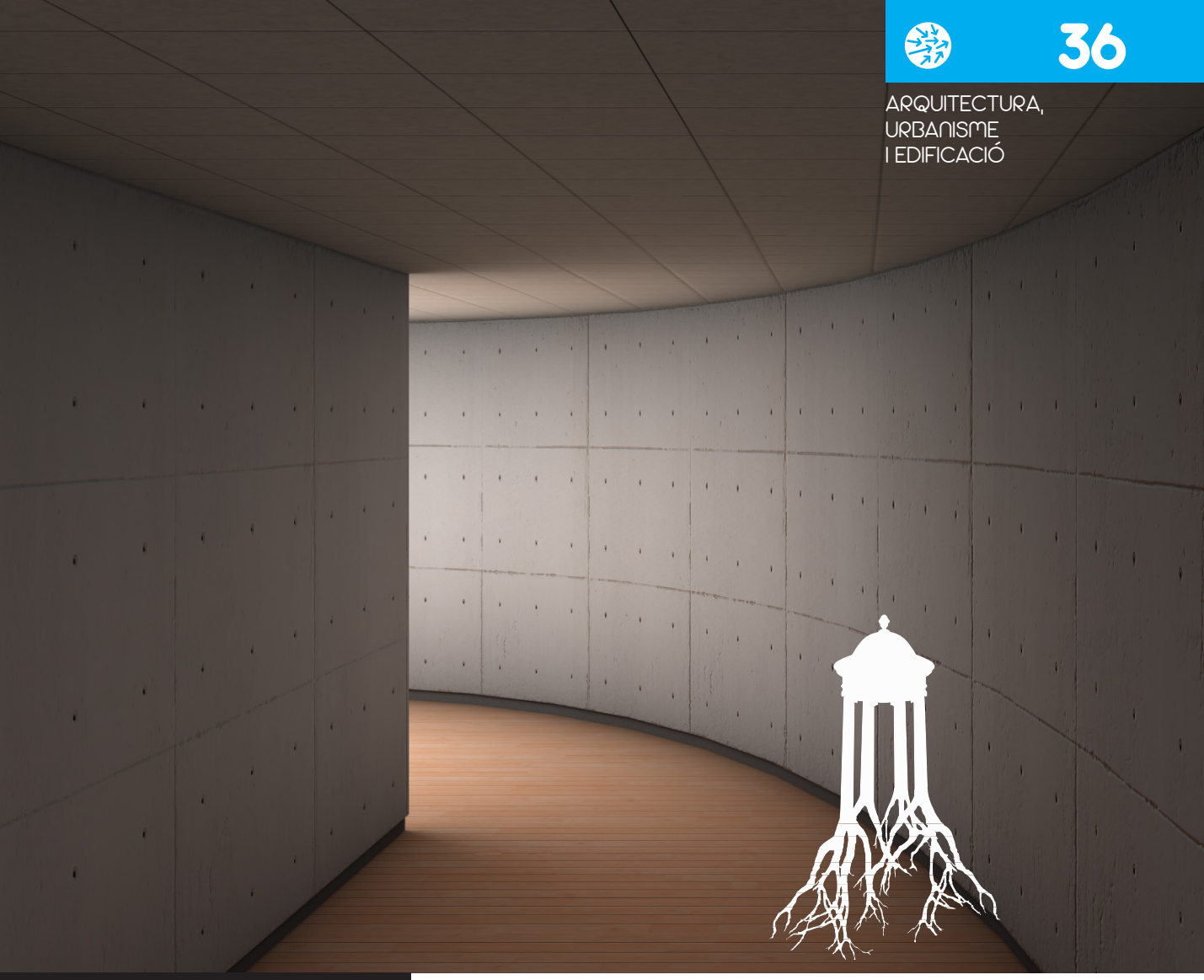
36

ARQUITECTURA,
URBANISME
I EDIFICACIÓ



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

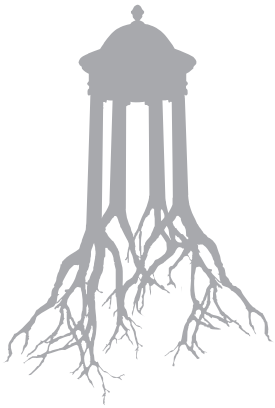
UPCGRAU

Simulación visual de materiales
Teoría. Técnicas. Análisis de casos

Javier Monedero

36

ARQUITECTURA,
URBANISME
I EDIFICACIÓ



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

iniciativa
digital politècnica
Publicacions Acadèmiques de la UPC

UPCGRAU

Simulación visual de materiales
Teoría. Técnicas. Análisis de casos

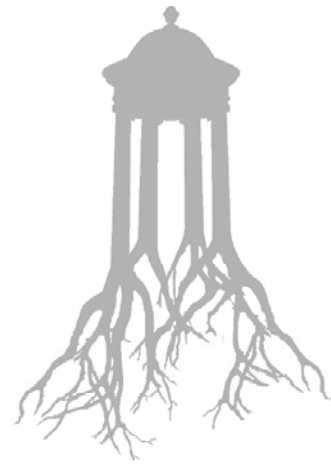
Javier Monedero

Primera edición: diciembre de 2015

Oficina de Publicacions Acadèmiques Digitals de la UPC
Jordi Girona 31,
Edifici Torre Girona,Planta 1, 08034 Barcelona
Tel.: 934 015 885
www.upc.edu/idp
E-mail: info.idp@upc.edu

DL: B. 29722-2015
ISBN: 978-84-9880-564-2

Índice



Introducción

La primera parte de esta introducción es común a dos libros que se publican en paralelo: *Simulación visual de la iluminación* y *Simulación visual de materiales*. A partir de aquí, los desarrollos se bifurcan aunque, en más de una ocasión me veré obligado a remitir el uno al otro y el otro al uno. Y, en algún caso, será inevitable alguna repetición pues las fronteras se solapan.

La creación de escenarios virtuales abarca muchos temas. Pero los dos principales son la representación o simulación de los materiales y de la iluminación. La interconexión es obvia pues si no hay materiales que redistribuyan la iluminación no podemos analizar ésta y, si no hay iluminación, poco podremos decir de las propiedades visuales de los materiales. Pero los problemas teóricos y técnicos implicados divergen.

Mi intención, que quizás atraiga a menos lectores aunque, espero, más exigentes, no ha sido elaborar un manual de cómo utilizar ciertas técnicas, sino situar estas técnicas en un contexto más amplio que pase, en primer lugar, por analizar los escenarios reales antes de intentar replicarlos por medios virtuales. Y esto supone un análisis extenso de las propiedades generales, físicas y perceptivas, que se dan en la realidad. Y, en segundo lugar, por lo que respecta a los escenarios virtuales, explicar los conceptos teóricos que hay detrás de las técnicas. Pues creo que el único modo de aplicar las técnicas de un modo adecuado es comprender bien la base teórica en que se apoyan.

El que sea imprescindible analizar a fondo las peculiaridades de las cosas reales antes de intentar simularlas por cosas virtuales parece que debería ser una obviedad. Pero a lo largo de muchos años de docencia he constatado que esto no es así. Y, hasta cierto punto, es lógico que no lo sea pues este análisis es complejo y abarca dos maneras muy distintas de abordarlo: una científica y otra artística. La obra de Leonardo da Vinci es un paradigma de lo que se puede conseguir cuando se atiende a estos dos extremos. En una época en la que parece que todo consiste en saber cuál es el botón que hay que apretar me parece saludable intentar, por difícil que sea, recuperar esta tradición.

Estas son las razones de esta división en técnicas de simulación de iluminación por un lado y de simulación de materiales por otro. Y ¿por qué "simulación" en lugar de, por ejemplo, "representación". No hay razones muy claras para elegir uno u otro término y confieso que he dudado entre ambos. Pero el hecho es que el término "simulación" se ha utilizado de modo preferente en los últimos años, por un lado para diferenciarse de los métodos tradicionales y, sobre todo, para recalcar el hecho de que los resultados se basan en procesos internos que implican bastantes más cosas que la simple proyección de un modelo sobre un plano.

En fin, para terminar esta introducción común, añadiré que no he incluido una sección de agradecimientos porque este libro es el resultado de más de veinte años dedicados a impartir clases y dirigir trabajos sobre estos temas en la ETS de Arquitectura de Barcelona. Principalmente en la asignatura optativa "Simulación Visual en Arquitectura por Medios Informáticos" (1994-2014). Pero también en otras asignaturas y cursos que se mencionan en la contraportada y de otros trabajos relacionados con estos temas de los cuales también he sido responsable. Hubiera sido necesario incluir docenas de nombres y seguro que me hubiera dejado alguno. Pero sí quiero dejar constancia de que he aprendido muchísimo de los cientos de estudiantes que he tenido a lo largo de estos años. En algunos

casos, particularmente brillantes, porque me han mostrado alternativas interesantes o me han dado información que no tenía. En otros porque me han hecho ver que lo que explicaba podía explicarse mejor o estaba basado en ambigüedades o en equívocos de los que no era consciente. A todos ellos mi más sincero agradecimiento.

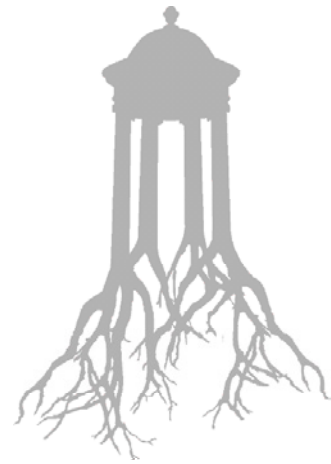
§ § §

La simulación visual de los materiales se enfrenta con tal diversidad de casos que, de entrada, desafía cualquier proyecto ordenado de síntesis. Los objetos que nos rodean ofrecen una riqueza inabarcable de colores y texturas, se modulan de mil modos con los cambios de luz, adquieren matices singulares por el reconocimiento y las resonancias simbólicas que proyectamos sobre ellos, aunque desbordan la memoria y la imaginación a nada que los observemos con suficiente atención. Sin embargo, la admiración invita, desafía, a la recreación. Y al igual que ha ocurrido con la pintura y la fotografía, la creación de imágenes sintéticas debe acompasar el deslumbramiento con el análisis. Así, pese a que suponga cierta violencia, es necesario separar, ordenar, clasificar, hacer acopio de recursos diversos para someter, hasta cierto punto, este extraordinario calidoscopio a leyes y técnicas que posibiliten esta recreación.

En los últimos cuarenta años, se han ido acumulando un sorprendente número de técnicas y recursos para emular esta riqueza natural. En unos casos, mediante la replicación técnica de propiedades físicas reales. En otros, mediante trucos ingeniosos que se han utilizado para construir réplicas artificiales con poco o ningún parentesco con los fenómenos reales. El concepto de *shader*, introducido a mediados de la década de 1980 por Robert Cook, ha sido una herramienta crucial en este desarrollo, al posibilitar la composición de algoritmos que servían para representar aspectos muy específicos, propiedades parciales, en un conjunto unitario que permitía generar superficies y fenómenos que, en unos casos, resultaban prácticamente indestinguibles de fenómenos reales y, en otros, daban lugar a resultados extraños para los que había que buscar alguna correspondencia natural.

La primera parte de este libro, *Teoría*, parte del análisis de lo real y, más exactamente, del dato inicial y final: los colores. Tanto desde el punto de vista de su percepción inmediata, fenoménica, como de una comprensión mínima de los procesos subyacentes, de las causas físicas de los colores. A partir de ahí se describen los principales desarrollos teóricos, los conceptos principales que han orientado la aparición de los *shaders* principales con que contamos en la actualidad y en que se han basado los múltiples métodos de simulación de las propiedades básicas de los materiales, tanto a nivel microscópico: absorción, reflexión, transparencia, dispersión, anisotropía… como a nivel mesoscópico: textura, relieve o variaciones generales de esas propiedades básicas en un entorno local.

La segunda parte, *Técnicas*, se centra en los modos específicos con que esos conceptos y recursos se pueden utilizar en la práctica para construir escenarios virtuales en los que las superficies materiales exhiban una riqueza de variaciones y matices equivalente a la que nos admira en los escenarios reales. Las técnicas disponibles son muy numerosas y es inevitable filtrarlas en función de las aplicaciones previstas. He dado preferencia a las que se relacionan directamente con la arquitectura y el diseño. Pero las fronteras no pueden ser rígidas y los métodos de simulación de objetos orgánicos pueden ser necesarios para la simulación de arquitecturas o elementos no convencionales o para la inclusión de elementos naturales en escenarios arquitectónicos. De ahí que también se hayan ampliado las explicaciones técnicas para incluir sistemas de partículas, medios participativos o proyecciones desplegadas (*unwrap*) de texturas complejas.



TEORÍA

→1

Materiales reales

Hay muchos libros de simulación visual que dan por supuesto que conocemos bien las características del mundo real y que no es necesario tratar de ellas. En este capítulo se parte de un supuesto contrario: que las conocemos muy mal. Y que sin un conocimiento más profundo de estas características, difícilmente podremos reproducirlas de un modo convicente o interesante.

1.1 Características generales de los materiales reales

Los objetos que nos rodean exhiben un sinfín de rasgos con los que estamos familiarizados. Las piedras parecen tener un color constante desde cualquier dirección. Los metales son brillantes y cambian de aspecto notoriamente si los miramos desde diferentes ángulos. El agua es transparente pero también azulada o verdosa. Los cristales transmiten la luz pero también la reflejan o la distorsionan. La leche es blanca porque dispersa la luz de un modo peculiar, al igual que ocurre con otros líquidos. Algunos materiales son luminiscentes. Otros, como las alas de las mariposas o los pétalos de algunas flores, son iridiscentes. Las hojas de los árboles tienen un aspecto metálico y opaco por uno de sus lados y luminoso y homogéneo por otro.

Si adoptamos unas determinadas condiciones de observación para contemplar un objeto y, manteniendo esas condiciones, nos olvidamos de su forma para concentrarnos en su superficie, resultará relativamente fácil describir sus características por comparación con las de otros objetos similares que podrían ocupar su lugar.

Antes de seguir hay que insistir en lo de "unas determinadas condiciones de observación", pues si, por lo menos, cambia la luz y cambia el punto de vista (cambia la distancia y cambia el ángulo de observación), todo lo que diremos a continuación dejará de ser cierto.

Sigamos entonces dando por supuesto que no cambia ni la distancia ni el ángulo (lo que implica, para que esto sea exactamente cierto, que estamos bien quietos y, preferiblemente, mirando al objeto con un solo ojo). ¿Qué vemos entonces? Colores. Ver una superficie quiere decir ver colores distribuidos de un modo u otro. Nada más. Si nada se mueve, ni la luz ni el objeto ni el ojo, las tres variables de la visión, todo lo que vemos son colores estáticos.

Si fijamos el ojo en un punto (lo que es bastante difícil) el color será único. Si desplazamos el ojo ligeramente en torno a ese punto pueden pasar dos cosas: que no haya cambios relevantes o que sí los haya. Si no hay diferencia decimos que el material no tiene textura, al menos en esa zona. Si hay diferencia, puede ser debido a que refleja la luz

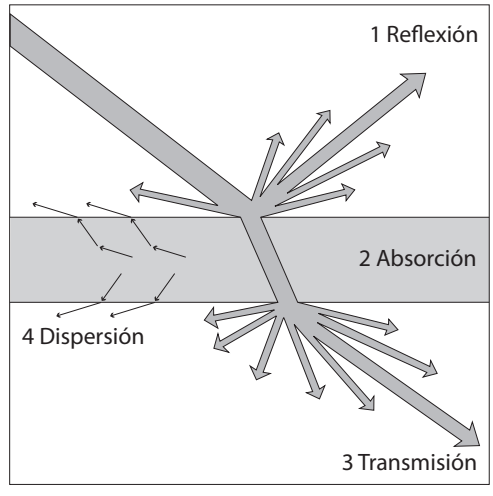


Figura 1.1 Interacción de la luz con una superficie semitransparente.

de modo diferente, aunque no tenga textura. Pero en general los objetos tienen algún tipo de textura.

Dejemos a un lado por ahora las variaciones locales (la textura). El hecho de que un objeto refleje la luz de un modo determinado es una propiedad puntual: tiene que ver con las propiedades intrínsecas del material considerando una región muy pequeña, casi microscópica. Para distinguirlas de otras propiedades, podemos denominar a estas, *básicas*, lo que nos permitirá relacionarlas, más adelante, con un grupo de parámetros que reciben también, en la mayoría de los programas de simulación, la denominación de *básicos*.

Desde este punto de vista, el de las propiedades básicas visibles, nos encontramos con las siguientes, que explican de modo muy general lo que le ocurre a una onda electromagnética al encontrar una superficie. Estas propiedades son fundamentalmente cinco: 1 reflexión, 2 absorción, 3 transmisión, 4 dispersión, 5 polarización.

La figura 1.1 muestra un esquema de las cuatro primeras. La reflexión hace que la luz se desvíe en una o varias direcciones, en el mismo lado con respecto a la superficie, que la luz incidente. La transmisión hace que la luz atraviese la superficie y, en general, se desvíe de su trayectoria. La absorción hace que la luz cambie su distribución espectral y que algunas longitudes de onda se anulen o se reduzcan, lo que se traduce en un cambio de color de la luz. La dispersión hace que parte de la luz reflejada o transmitida vuelva a ser emitida por puntos diferentes a los de entrada. La polarización cambia el plano de vibración de la luz, algo que solo es perceptible en circunstancias muy especiales: tiene una importancia bastante menor para lo que nos interesa aunque diremos algo más sobre ella más adelante.

Pero, en cuanto nos alejamos de la observación puntual, aparecen otras propiedades notables.

En primer lugar, la anisotropía. Si nos desplazamos en torno al punto de observación, en muchos objetos veremos que las características del reflejo cambian. Ocurre así porque en estos casos las propiedades dependen de la dirección. Esto puede ser debido a su estructura interna, como ocurre con las fibras de la madera o las vetas del mármol o bien al tratamiento superficial recibido, como ocurre con el bruñido o el repujado de algunos metales o la modificación de la forma o la rugosidad provocada por otros tratamientos en múltiples casos. Si las propiedades del material son independientes de la dirección el material es isótropo.

En segundo lugar, la textura. Podemos definir la textura como la variación de color local debido a dos causas muy diferentes: a la diferencia de color de los componentes puntuales, aunque no haya variación formal de la superficie (que es lo que pasa en, por ejemplo, una superficie de mármol pulido con vetas de diferente color) o a la diferencia de forma de los componentes puntuales, aunque no haya diferencia de color en la superficie (que es lo que pasa en, por ejemplo, la piel de una naranja o un sillar de piedra arenisca desbastada).

Todo lo que hemos dicho hasta aquí tiene que ver con las propiedades de los materiales desde el punto de vista de una ciencia concreta, la ciencia de la óptica. La óptica moderna se puede considerar dividida en tres grandes áreas: la óptica física, que se basa principalmente en la consideración de la luz como un fenómeno ondulatorio y su interacción con objetos de tamaño similar a su longitud de onda; la óptica cuántica, que se basa principalmente en la interacción de la luz con los átomos y las moléculas; la óptica geométrica, que se basa principalmente en el análisis de efectos macroscópicos.

La mayor parte de los métodos de simulación digital se basan en la óptica geométrica y en la representación de la luz mediante parámetros simples que representan la cantidad de luz que se propaga en una determinada dirección. Pero este punto de vista tiene sus limitaciones, y para una comprensión adecuada de determinados fenó-

menos que requieren métodos especiales habrá que hacer alguna incursión breve en las otras dos áreas.

Por último, hay que remarcar que, al centrarnos en las propiedades ópticas, hemos descartado las propiedades que tienen que ver con el comportamiento de los materiales, principalmente su comportamiento mecánico (resistencia, deformabilidad), térmico (capacidad de conducir y resistir el calor) o su capacidad aislante o resistente a diversos tipos de agentes. Sin embargo, hay una relación bastante cercana entre las características generales y las características visibles. Atendiendo a sus propiedades fundamentales, los tratados de ciencia de los materiales los clasifican en los cinco grupos siguientes, que no estará de más tener presentes:

a) Materiales metálicos. Se basan en elementos o en combinaciones de elementos metálicos. Disponen de electrones deslocalizados en su superficie en lo que, como veremos más adelante, se denomina "banda de conducción". Son buenos conductores del calor y la electricidad. Son resistentes y deformables.

b) Materiales cerámicos. Se basan en compuestos químicos que agrupan a metales y no metales. Son aislantes térmicos y acústicos en la gran mayoría de los casos. Son más resistentes que los metálicos y los poliméricos a la temperatura y los agentes químicos. Son duros y frágiles.

c) Materiales poliméricos. Se basan en compuestos, por lo general orgánicos, que incluyen cadenas moleculares de gran longitud. Tienen una densidad muy baja en comparación con otros materiales. Son flexibles y elásticos.

d) Materiales semiconductores. Se basan en elementos muy concretos (como el silicio o el germánico) y alterados por impurezas. Sus propiedades como conductores están a caballo entre los conductores (metales) y los aislantes (no metales).

e) Materiales compuestos. Se basan en combinaciones de otros materiales. En la gran mayoría de los casos se trata de materiales producidos artificialmente para sacar el mejor partido de las diferentes propiedades de sus compuestos. Esto incluye tanto materiales tradicionales como nuevos materiales diseñados específicamente para incorporar determinadas propiedades.

1.2 Causas inmediatas de los colores

Como he dicho en la sección anterior, si nos concentramos en el aspecto de un material, lo que vemos se reduce a colores. Y, como veremos más adelante, la simulación visual de materiales abarca un conjunto de técnicas que, en última instancia, sirven para generar una serie de puntos de color que son percibidos por los seres humanos como si correspondieran a puntos de color reales.

Vistas así las cosas, resulta que no estamos hablando de otra cosa que de colores. Y aunque la teoría de los colores es otro tema de estudio en sí mismo, algo habrá que decir acerca de algunos aspectos principales de los colores desde un punto muy abstracto o, según como se mire, muy concreto, porque es lo más inmediato, la causa inmediata de la percepción de lo que reconocemos como objetos de uno u otro material. Y, además, porque la simulación virtual de los materiales implica una reducción importante de la gama de colores reales y es necesario entender bien de qué orden es esta reducción.

La luz, los objetos y el ojo

Si distinguimos objetos es porque tenemos ojos. Y porque estos objetos están iluminados. Y porque estos objetos nos devuelven la luz que reciben de distintos modos. Si apagamos la luz no veremos objetos. Y si cerramos los ojos tampoco. Y si no hay objetos es obvio que tampoco veremos nada.

Estos tres elementos son indisociables. Un objeto se distingue de otro objeto porque absorbe una parte y refleja otra parte de la luz que recibe. Y porque lo hace de un modo pe-

culiar que no es el mismo para un metal que para una piedra pulida.

Por otro lado, la luz puede entenderse, en cierto modo, como un material y por eso decimos que hay materiales que emiten luz y otros que solo la reflejan en mayor o menor grado. El objetivo principal de este libro es analizar los objetos y, hasta cierto punto, la luz, en tanto que es producida por cierto tipo de objetos. Pero antes de adentrarnos en estos dos componentes de la visión habrá que decir algo sobre el tercero, sobre el modo en que nuestros ojos o, más exactamente, nuestro sistema visual, capta los colores, pues esto puede influir considerablemente en la eficacia de muchas técnicas de simulación.

Receptores cromáticos superficiales. El ojo

Esta sección incluye una descripción del sistema visual humano que es importante para comprender muchas de las características de las imágenes, tanto las que formamos a partir de la percepción de objetos reales como las que se generan sintéticamente. Gran parte de la descripciones son inevitablemente iguales a las incluidas en el libro sobre simulación de iluminación citado en la introducción. En este caso, sin embargo, se han relegado todos los temas relativos a la iluminación, a la formación de imágenes y a explicaciones más extensas sobre el sistema visual que se tratan más extensamente en dicho libro y, por el contrario, se ha dado mayor importancia a todos los temas relativos al color.

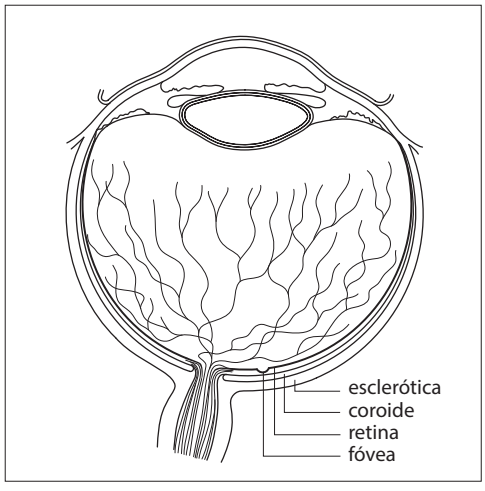


Figura 1.2 Sección horizontal del ojo humano.

El ojo humano tiene la forma aproximada de una esfera de unos 20 mm de diámetro. Está constituido por tres envolturas principales: una primera envoltura, más externa, la *esclerótica* que, al verterse al exterior forma una zona opaca, el blanco del ojo, y una zona transparente, la córnea; una segunda envoltura, la *coroide*, que recubre internamente la esclerótica y al prolongarse por detrás de la córnea forma el iris. El iris deja una abertura, de diámetro variable entre los 2 y los 8 mm, la pupila. Tras la pupila se sitúa el cristalino, constituido por capas superpuestas que forman algo similar a una lente convergente. El índice de refracción varía de una a otra de estas capas que forman el cristalino y su promedio es de 1,42 aproximadamente. La variación de la convergencia se produce por la modificación de estas capas bajo la acción de los músculos ciliares. Entre la córnea y el cristalino hay un medio líquido, el humor acuoso, de índice de refracción similar al del agua (1,33). La cavidad general del ojo, tras el cristalino, está llena por el humor vítreo, un líquido de índice de refracción también similar al del agua.

La tercera envoltura, la más interna, es la *retina*. La retina está cubierta por dos tipos de filamentos nerviosos que se conectan con el cerebro a través del nervio óptico situado en el "punto ciego". Hay alrededor de 120 millones de estas células terminales. Las más numerosas, y más abundantes en la periferia que en el centro, se denominan *bastones* y captan tan solo diferencias de luminosidad. Las menos numerosas, algo menos de un 10 % del total, entre 6 y 8 millones, y más abundantes en la zona central, se denominan *conos* y son las que captan colores. En esta zona central se sitúa la *fóvea*, la región

de mayor sensibilidad del ojo y el punto de fijación que orienta el enfoque de los objetos hacia los que se dirige la atención visual. La fóvea tiene unos 1,5 mm de diámetro, contiene unos 110.000 conos y abarca unos 5,2º de campo.

En el interior de la fóvea se distingue la *fovela*, una pequeña área de 0,4 mm de diámetro en la que prácticamente solo hay conos, unos 25.000, y que abarca aproximadamente 1,4º. En el interior de la fovela se distingue, por último, una región aún más limitada denominada *isleta central*, en la que no hay bastones y los conos tienen una longitud máxima. Esta área tiene unos 0,050 o 0,075 mm de diámetro y abarca entre 0,17º y 0,24º.

Alrededor de la fóvea se extiende la *mácula lutea*, un área amarillenta de unos 3 mm a 5 mm de diámetro y que abarca entre 10º y 17º. La zona que se extiende más allá de la mácula lutea contiene progresivamente mayor número de bastones y menor número de conos. A partir de los 10 grados aproximadamente los bastones comienzan a predominar sobre los conos.

La percepción de la luz y de los colores depende de los receptores fundamentales: los bastones y los conos. La existencia de estos receptores retinianos ya era conocida en 1850, cuando Helmholtz escribió su gran tratado en el que se dan descripciones precisas de la estructura microscópica de la retina, También se conocía por esta época el papel juzgado por la *rodopsina*, una proteina capaz de reaccionar químicamente a la luz y que es el principal constituyente de los pigmentos visuales de los bastones. La sensibilidad de los bastones a la luz es mucho mayor que la de los conos si bien, en condiciones normales, la visión se apoya fundamentalmente en los conos. Los bastones captan intensidades

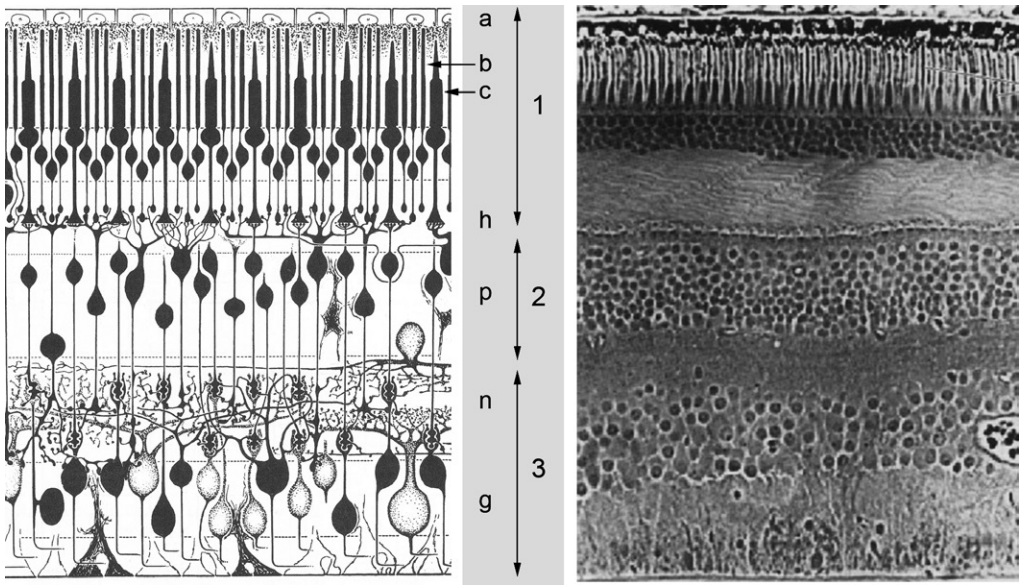


Figura 1.3 Sección de la retina mostrando las capas principales: 1 Capa de fotorreceptores: (a) epitelio pigmentado (fondo de la retina), (b) bastones, (c) conos. 2 Capa de células intermedias: (h) horizontales, (p) bipolares, (n) anacrinas. 3 Capa de células ganglionares (g). A la izquierda, un esquema adaptado de Polyak (1941). A la derecha, una imagen que muestra una sección de la retina, adaptada de Boycott y Dowling (1969) que se corresponde aproximadamente con el esquema de la izquierda. Obsérvese que el recorrido de la luz en ambos casos sería de abajo arriba, atravesando las capas celulares antes de activar los fotorreceptores de la retina.

de 2 a 5 quanta, 1.000 veces más pequeñas que las de los conos. Por otro lado, los bastones son ciegos al color. La percepción del color depende de los conos.

El cristalino del ojo juega cierto papel en la percepción de los colores. En primer lugar, absorbe las longitudes de onda más cortas, un efecto que afecta, en general, tan solo a las longitudes de onda más cortas pero que tiende a aumentar con la edad, llegando a afectar a ondas del orden de los 500 nm o más. Esto quiere decir que tan solo pasan las ondas correspondientes a los rojos y verdes cuya combinación da el amarillo, un color característico del color del cristalino de las personas ancianas.

En segundo lugar, hay que tener en cuenta que el cristalino no está corregido con respecto a las aberraciones cromáticas, como ocurre con las sofisticadas lentes de las cámaras de fotografía modernas. Esto hace que se tienda a compensar este efecto mediante variaciones en el enfoque según los diferentes colores. Las longitudes de onda más cortas (azules) que se desvían más al atravesar el cristalino, son enfocadas para que caigan más cerca. En una situación normal, con una distribución variada de colores, el ojo se enfoca para una longitud media que estaría situada en torno a los 580 nm. Los sujetos miopes pueden notar claramente las diferencias entre longitudes de onda corta y onda larga. Una luz azul puntual, contemplada a distancia, produciría un círculo de más de 1 cm de diámetro sobre la retina. Un efecto general, relacionado con este fenómeno, es el leve círculo azul que parece envolver las luces rojas traseras de los automóviles. Rodear objetos con límites de franjas rojas y azules, que parecerían emitir un leve resplandor, es una técnica que se ha utilizado ocasionalmente en las artes plásticas y que tenía un fundamento físico en este fenómeno.

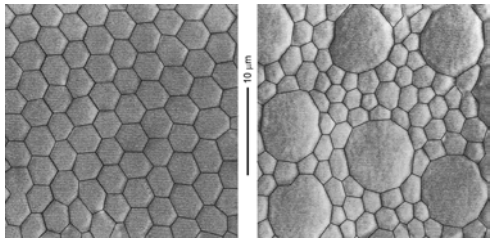


Figura 1.4 Sección de la retina mostrando conos en la fóvea (izquierda), menores y muy compactos, y en la periferia (derecha), mayores y más espaciados. La barra central muestra la escala (10 micras). Basado en Curcio et al.*, 1990.*

Hacia 1810, el científico inglés Thomas Young lanzó la hipótesis de que si con tres colores primarios se podían generar innumerables colores, el sistema visual humano debía contar también con tres receptores especializados capaces de generar los colores que percibimos a partir de algún tipo de mecanismo combinatorio.

La prueba de la existencia de receptores retininianos especializados en la percepción de los colores, que diese un fundamento real a la teoría planteada por Thomas Young a comienzos del siglo XIX, se convertió en la meta perseguida por un considerable número de investigadores durante el siglo XIX y, sobre todo, el siglo XX. Gracias a los trabajos de Rushton y Wald (este último galardonado con el premio Nobel en 1964 por sus descubrimientos en este campo), por no nombrar más que los dos científicos más conocidos que han investigado la fisiología de la retina, y a la aparición de nuevos instrumentos de medición, como los densitómetros o los espectrofotómetros microscópicos, pudo llegarse, durante la segunda mitad del siglo XX, a las siguientes conclusiones.

Existen al menos tres clases de conos, tres clases de receptores retinianos, que alcanzan su máxima sensibilidad en diferentes regiones del espectro. Estas tres clases de conos son: 1) Los sensibles principalmente a las ondas largas (rojo y amarillo), que estarían compuestos por una proteína denominada *eritrolabe* y que alcanzarían un máximo de receptividad en la región situada en torno a los 580 nm. 2) Los sensibles principalmente a las ondas medias (verde y amarillo verdoso), que estarían compuestos por una proteína denominada *clorolabe* y que alcanzarían un máximo en la región situada en torno a los

555 nm. 3) Los sensibles principalmente a las ondas cortas (azul), que estarían compuestos por una proteína denominada *cianolabe* y que alcanzarían un máximo a los 440 nm.

Esta distribución presenta algunas peculiaridades. Así, es posible estimular con longitudes de onda largas (rojas) a una sola clase de conos pero las longitudes de onda corta estimularían a los tres simultáneamente. Esto es debido, en parte, al hecho de que los máximos de cada clase de conos está más cerca del lado de las ondas cortas, pero, sobre todo, a la presencia de una banda secundaria de absorción (ondas beta).

Los conos de tipo L (sensibles a las ondas largas) y de tipo M (sensibles a las ondas medias) son mucho más abundantes que los de tipo S (sensibles a las ondas cortas). La proporción global está en torno a los 40/20/1 (L/M/S). Además, la sensibilidad a la longitud de onda de los pigmentos de tipo L y M es muy similar, como puede apreciarse en la figura 1.5, mientras que la de los pigmentos de tipo S es claramente mayor para longitudes de onda corta. La investigación sobre el papel de los pigmentos de tipo S se ha desarrollado durante muchos años. Se ha descubierto que están más separados que los otros dos tipos, del orden de 10' de arco visual. Su sensibilidad es máxima en la zona correspondiente a 1º desde la fóvea.

En la fóvea, la zona de fijación, hacia la que dirigimos la mirada, no se encuentran conos sensibles al azul. Esta área se considera, desde el punto de vista fisiológico, como dicromática (ciega al azul). Alrededor de la fóvea, en un sector comprendido entre los 20º y los 30º, se extiende un área tricromática en la que los tres tipos de conos están presentes. En la periferia de esta área el rojo y el verde dejan de distinguirse. A partir de aquí, y en una tercera área comprendida aproximadamente entre los 20º-30º y los 70º-80º, hay otra área dicromática en la que no se distinguen el rojo y el verde. A partir de los 80º aproximadamente la visión es monocromática. Tan solo hay bastones hasta los límites de la visión propiamente dicha que están comprendidos (para los dos ojos) entre los 180º y los 210º.

Es preciso recalcar el hecho de que la mayoría de estos resultados están obtenidos mediante experimentos estrictamente fisiológicos. Asi, una de las pruebas consideradas más convincentes es medir la absorción para diferentes longitudes de onda de conos extraidos de retinas disecadas. Hasta 1980 se habían obtenido datos relativos a siete hombres y un número bastante mayor de monos macacos (cuyo sistema de visión parece ser bastante parecido al del hombre). Otro método es el blanqueamiento selectivo. Y otro método de considerable importancia, pues parte de bases no estrictamente fisiológicas es el desarrollado por Stiles que consiste en mediciones de aumento del umbral. Se pueden ampliar estas explicaciones en obras especializadas como la del propio Stiles que se menciona en la bibliografía (véase Stiles y Wyszecki, 1982).

En cualquier caso, la mayoría de los científicos están de acuerdo en que puede hablarse de tres tipos de conos, si bien no está aún del todo claro, como veremos, la función que juegan estos tres tipos de receptores en la visión cromática.

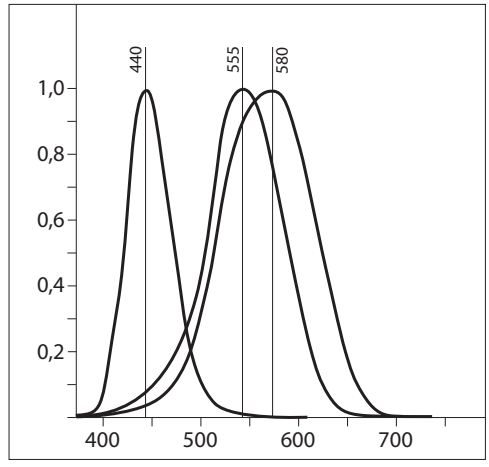


Figura 1.5 Respuestas espectrales de los conos S, M y L, normalizadas para facilitar la comparación (eje vertical) e indicando las frecuencias de máxima absorción.

Este resultado concuerda, como argumentaba Young, con el hecho experimental, fundamento de la teoria de los colores, y al que volveremos más adelante, de que casi cualquier color puede ser reproducido por mezcla aditiva de tres luces monocromáticas, de tres tonos primarios. Pero las longitudes de onda de los primarios utilizados en mezcla aditiva, desde Maxwell hasta la CIE, no coinciden con las longitudes de onda características de los conos. Aquellas están comprendidas entre los 400-450 nm (azul), 510-520 nm (verde) y 630-700 nm (rojo). Rangos bastante distintos que los picos que he citado para las absorciones características de las tres clases de conos.

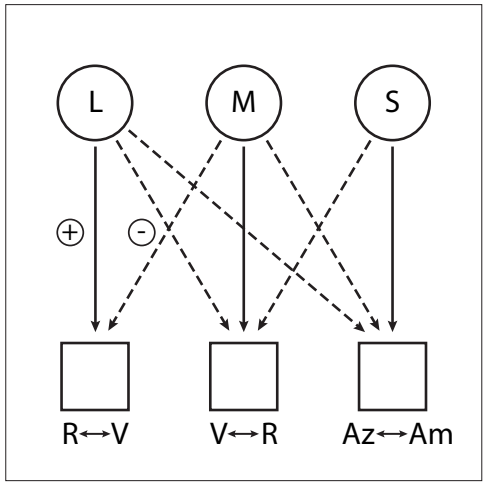


Figura 1.6 Salidas de colores de oposición desde los tres tipos de conos hasta las células ganglionares de la retina. Los conos L se oponen a los M en el canal rojo-verde y verde-rojo. Los conos S se oponen a los L y M en el canal azul-amarillo. Adaptado de Zeki, 1993.

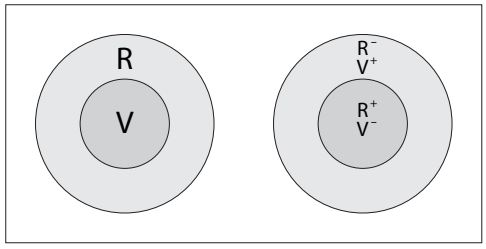


Figura 1.7 a*) Campo receptor de una célula oponente al color en el núcleo geniculado lateral que es excitada por una luz de onda media (V) en el centro y de onda larga (R) en la periferia;* b*) Campo receptor de una célula oponente doble que es excitada por una luz de onda larga (R) e inhibida por una luz de onda media (V) en el centro y a la inversa en la periferia. Adaptado de Zeki, 1993.*

Hasta quí lo que nos dice la fisiología. Con todo lo que se ha aprendido durante los últimos 150 años sobre los receptores superficiales, hay muchas características de la visión cromática que quedan sin explicar. Y esto ha llevado a pasar a otro nivel de explicación.

Procesamiento retiniano. Canales cromáticos. El córtex visual

En paralelo a los descubrimientos anteriores, a lo largo sobre todo de la segunda mitad del siglo xx, se ha ido ampliando nuestro conocimiento del modo en que la organización del cerebro está, en gran medida, basada en áreas funcionales especializadas. Un gran número de los trabajos de investigación que se han llevado a cabo para localizar con precisión estas áreas han tenido que ver con la visión y con el color.

La relación entre la retina, los receptores primarios del color y las áreas corticales en donde se procesa la visión, se ilustra en las figuras 1.6, 1.7 y 1.8.

En primer lugar, hay que decir que gran parte del procesamiento de las señales se da en la propia retina. Concretamente, en las células ganglionares que están situadas antes que los receptores básicos (curiosamente, la luz las atraviesa antes de llegar a estos, que luego reenvian sus señales a estas células), de tal modo que las señales que salen de la retina, por el punto ciego, hacia el córtex visual ya están preprocesadas. En la figura 1.3 ya hemos visto una sección de la retina en la que aparecen las diferentes capas que reaccionan a la luz. Los dos diferentes tipos de receptores retinianos envían sus señales a las diferentes capas de células en las que tiene lugar este procesamiento básico.

Tal como se explica con más detalle en el libro sobre Iluminación, las células gangliona-

res tienen un comportamiento centro/periferia caracterizado por una combinación de estímulos e inhibiciones. Este mecanismo permite garantizar que las variaciones de intensidad no afectan a las señales de un modo absoluto sino relativo: al aumentar la intensidad de la luz global el incremento de la señal del centro se ve compensada por un incremento proporcional de la inhibición del campo lateral asociado, con el resultado de que la señal significativa se mantiene igual.

Debe tenerse en cuenta que cuando una molécula de fotopigmento absorbe luz, el efecto es el mismo, con independencia de la longitud de onda. Es decir, que aunque un cuanto de luz de 400 nm posea más energía que uno de 700 nm, la secuencia de reacciones de la rodopsina es la misma. Rushton utilizaba el término *univariancia* para recordar que un fotopigmento da lugar a una única variable como respuesta a la luz que recibe. Es decir, que lo que cuenta es el índice de absorción (*rate of absorption*). No incorpora, por tanto ninguna información sobre la composición espectral de la luz. Sin embargo, puede absorber más luz de unas frecuencias que otras. La univariancia implica que, una vez absorbidos, los cuantos de luz tienen el mismo efecto visual.

En el caso de la percepción de los colores, la estructura es bastante más compleja y está lejos de ser bien comprendida, si bien hay algunos datos significativos que están bien fundamentados desde hace años. Las figuras 1.6 y 1.7 muestran la estructura de formación de canales en la retina. Los conos L envían señales excitatorias y los conos M señales inhibitorias al canal rojo-verde o verde-rojo de las células ganglionares.

Estas células tienen una estructura de su campo receptivo caracterizada por una zona central que es activada por longitudes de onda larga (o media) y una zona periférica que es activada por longitudes de onda media (o larga). Hay también células oponentes más complejas, células oponentes dobles, como las que se han encontrado en la retina de peces de colores que son excitadas por ondas largas e inhibidas por ondas medias en su zona central y a la inversa en su periferia.

Estos breves resúmenes y las figuras citadas pueden dar una cierta idea de la situación actual sobre la visión de los colores. Pero aún no se sabe con claridad de qué modo toda es-

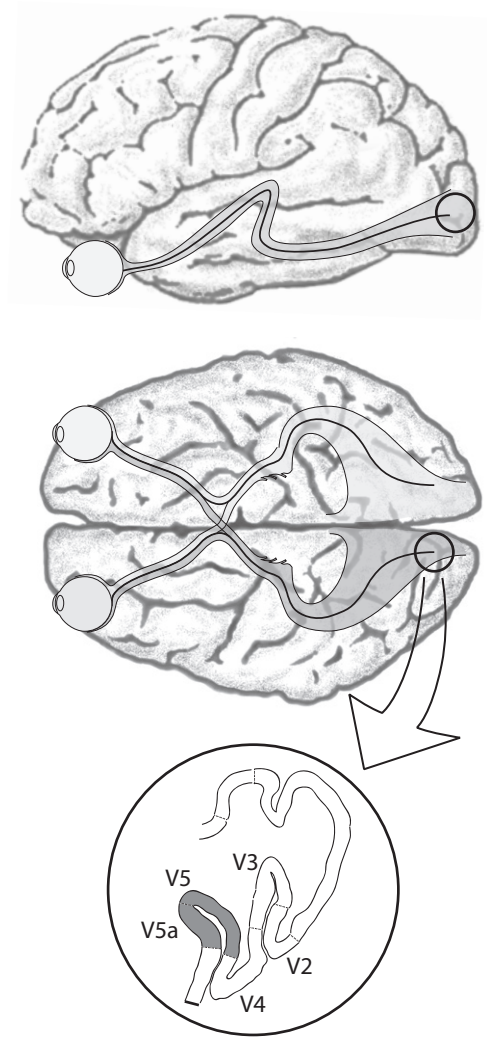


Figura 1.8 a*) Sección vertical del cerebro mostrando los ojos y su conexión con el córtex visual;* b*) Sección horizontal del cerebro mostrando los ojos, el quiasma, el núcleo geniculado lateral y el córtex visual;* c*) Detalle del córtex visual mostrando las áreas V1 a V4.*

tructura de células y canales en oposicion se traduce en nuestra percepción de los colores.

A lo largo del nervio óptico discurren las fibras que transmiten esta información preprocesada en la retina, el tracto óptico. Estas fibras se subdividen de tal modo que la zona lateral externa de cada ojo va a parar al hemisferio cerebral del mismo lado, al denominado *campo visual ipsilateral* pero la zona lateral interna va a parar al hemisferio opuesto, al denominado *campo visual contralateral*. Esto implica un cruce que se produce en el quiasma óptico.

Más allá del quiasma, se encuentra una primera zona fundamental, una subdivisión del tálamo subcortical denominada *núcleo geniculado lateral* (NGL). Este núcleo está formado por 6 capas que se relacionan de modos específicos con las zonas de la retina mencionadas. El campo visual ipsilateral se conecta con las capas 5, 3 y 2, y el contralateral, con las capas 6, 4 y 1. Y puntos adyacentes de la retina, como se ha comprobado en diversos experimentos, se corresponden con puntos adyacentes en cada capa del NGL. Por otro lado, y sin entrar en la descripción de la compleja estructura de este núcleo, también se puede mencionar que las cuatro capas superiores están formadas por pequeños cuerpos celulares, por lo que se las denomina capas P (parvocelulares), y las dos inferiores por grandes cuerpos celulares, por lo que se las denomina capas M (magnocelulares). Las células de las capas P están relacionadas con la visión del color pero las de las capas M no. Con todo, el papel del NGL, que actúa como una especie de filtro de la información que llega de las áreas retinianas, no se comprende demasiado bien. Se comprende mejor, sobre todo desde los descubrimientos de los últimos años, el papel de áreas más profundas.

Más allá del NGL se encuentra la corteza visual primaria o córtex visual, denominado así porque el procesamiento principal de las señales visuales se lleva a cabo en esta zona. El descubrimiento de estas áreas se debe, como decía al comienzo, al paciente trabajo de cientos de investigadores entre los que cabe citar en primer lugar a David Hubel y Torsten Wiesel (premios Nobel en 1981) por sus experimentos iniciados a finales de la década de 1950.

El resultado de estos trabajos se puede resumir apretadamente como sigue.

En primer lugar, hay una relación directa entre zonas de la retina y zonas de la corteza visual primaria, de tal modo que estas zonas reproducen un mapa de la retina en una zona determinada denominada V1 o zona visual primaria. La zona V1 es la que contiene el mapa más detallado de la retina, junto con una arquitectura funcional muy rica (neuronas especializadas en captar determinados rasgos tales como rectángulos, bordes y esquinas con determinadas orientaciones, etc.) que Hubel y Wiesel fueron los primeros en desvelar con precisión.

En segundo lugar, la zona V1 está estrechamente relacionada con otras cuatro áreas denominadas respectivamente, V2, V3, V4 y V5. La figura 1.8 muestra la ubicación de estas áreas. En los últimos veinte años se ha encontrado amplia evidencia de que la informacion que llega de la retina se representa de manera independiente y diferente en estas cuatro áreas. La visión es, por tanto, un proceso extraordinariamente complejo en donde la información puntual juega tan solo una parte. Estas áreas presentan cierto grado de especialización. Concretamente, la V2, aunque comparte muchas funciones de reconocimiento de patrones básicos con la V1, incorpora otras más complejas tales como la orientación de contornos virtuales o la separación entre figura y fondo y, probablemente, el reconocimiento de patrones complejos. La V3, cuya extensión exacta aún es objeto de discusión, parece jugar un papel importante en la percepción de la forma dinámica. La V4 tiene un papel fundamental (pero no único) en la percepción del color. A partir de los trabajos de Zeki, a finales de la década de 1970, se consideró que la función principal del área V4 era la percepción del color, pero trabajos posteriores han mostrado que en ella también están localizadas funciones de reconocimien-

to de patrones de complejidad intermedia. Por último, la V5 juega un papel tan fundamental en la percepción del movimiento que lesiones en esta área pueden ocasionar sorprendentes deficiencias tales como la ceguera al movimiento (los que padecen esta lesión ven cómo los objetos cambian de posición pero son incapaces de seguir su pista).

Es importante tener presente que hay una multitud de áreas que colaboran en la construcción del color aunque algunas de ellas puedan distinguirse de las otras por cierto grado de especialización funcional. La idea de construcción del color es importante pues también hay amplia evidencia de que el cerebro no percibe materiales o algo así como superficies con una etiqueta que dice de qué tipo de material son, sino que el cerebro construye esta información a partir de múltiples datos que se combinan de diversos modos. Esta idea de construcción activa resulta más clara cuando se analizan fenómenos fundamentales y aparentemente contradictorios como la interacción cromática y la constancia cromática. Pero antes hay que añadir algo más sobre nuestra capacidad de discriminación cromática.

Apariencia cromática. Categorías perceptivas. Nombres de los colores

Los mecanismos de percepción de los colores que he resumido en los apartados anteriores están directamente ligados a la producción y reproducción de los colores, un tema al que volveré más adelante. Sabemos que para generar la gran mayoría de los colores que podemos percibir, bastan tres o cuatro colores primarios que estimulen a nuestros receptores retinianos del mismo modo que lo harían los componentes principales del color reflejado por una determinada superficie, aunque un análisis espectrofotométrico de este color revelaría que su distribución espectral es más compleja.

Pero los colores que percibimos pueden no corresponder a los colores que producimos pues hay toda una variedad de fenómenos que influyen en la percepción. Algunos de estos fenómenos, que comentaré más adelante, tales como la interacción o la inducción cromática, son relativamente bien conocidos. Otros son más complejos y pueden encontrarse en la literatura especializada. Efectos tales como el efecto de Bezold-Brücke (desplazamiento tonal con la luminancia), el efecto Abney (desplazamiento del tono con la pureza cromática) o el efecto Hunt (aumento del colorido con la luminancia). La obra de Fairchild a la que me referiré a continuación contiene un buen resumen de los principales (véase Fairchild, 1997, cap. 6).

En la percepción de la apariencia cromática utilizamos, consciente o inconscientemente, categorías perceptivas globales que definen un espacio cromático independiente. Estas categorías perceptivas son el tono, la luminosidad y la saturación. Cualquier persona con la vista mínimamente educada es capaz de juzgar si un color está en una otra banda del espectro, si, por ejemplo, es más frío o más cálido, más azulado o más rojizo: es decir, cuál es su tono. O bien si es más claro o más oscuro: es decir, cuál es su luminosidad o su brillo relativo. O bien si es más saturado o menos saturado aunque en este

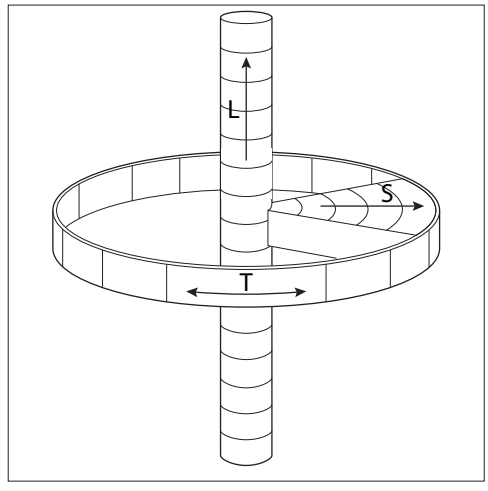


Figura 1.9 Apariencia cromática. El espacio de las tres categorías fundamentales.

caso el juico pueda mezclarse con la mayor o menor luminosidad.

Hacia 1900, el pintor americano Munsell (1858-1918) solucionó el problema de mantener la coherencia entre luminosidades correspondientes a los tonos puros y luminancias generales, con un modelo sólido muy sencillo que sigue siendo uno de los más utilizados hasta la fecha. Munsell partía, de un modo más explícito que la mayoría de sus predecesores, de tres categorías bien diferenciadas: el tono (*hue*), el valor (*value*) y el croma (*chroma*).

Estas categorías se han utilizado en otros sistemas de clasificación de los colores importantes, como el sistema DIN, utilizado en Alemania, el sistema NCS (Natural Color System), utilizado principalmente en Suecia, y el sistema OSA, propuesto por la Optical Society of America.

Pero son también la base de intentos más complejos y más ambiciosos de crear sistemas que permitan predecir con precisión el modo en que percibimos los colores bajo condiciones ambientales variables. La obra de Fairchild citada antes es una buena referencia para quien quiera adentrarse en este estudio. En este apartado me limito a mencionarla como introducción a una descripción más general de los límites de discriminación para cada una de estas variables.

La categorización de los colores entronca con otro tema de interés que también apuntaré brevemente.

El lenguaje es algo vivo en el que muchas palabras nacen y mueren o llevan una existencia marginal. Hay que precaverse por consiguiente contra las generaciones simplificadoras.

Se cita con cierta frecuencia, por ejemplo, que los esquimales cuentan con un gran número de palabras para designar el blanco, palabras que serían de imposible traducción a otros idiomas pues, se considera, tienen un vocabulario más amplio para nombrar a los tonos blancos.

Que los esquimales sean un pueblo alejado de nuestro entorno cultural hace que semejante dato pueda resultar exótico o excepcional. Pero los tintoreros, que existen en todas las ciudades occidentales, emplean o empleaban, un gran número de términos distintos para designar a diferentes clases de negro. Que esta especialización terminológica, ligada a una especialización técnica pueda llegar a sorprender es debido al hecho de que se acepta implícitamente que los colores son algo universal, una sensación compartida que no admitiría, en sí misma, una especialización tal como la que se da en otras actividades en las que admitimos naturalmente que existen términos relativamente extraños para designar objetos u operaciones sobre los objetos que no tenemos porque conocer.

En el siglo XIX aparecieron varios estudios sobre los términos utilizados a lo largo de la historia y en diferentes culturas para designar los colores. Algunos de estos estudios se debieron al primer ministro inglés W. Gladstone (*Studies on Homer and the Homeric Age*, 1858), otros muy citados fueron los de H. Magnus en 1877 y 1880 (hay traducción al español: *Evolución del sentido de los colores.* Hachette, Buenos Aires, 1976) o los de L. Geiger (*Contributions to the History of the Development of the Human Race*, 1880) por no mencionar sino algunos. Algunas de las conclusiones de estos primeros estudios, que luego se han desmentido, eran que los griegos debían ser ciegos al azul pues en ninguno de los textos antiguos se encuentran nombres genéricos para este color.

Estos estudios fueron continuados por dos investigadores americanos, Berlin y Kay, hacia 1967 en un seminario para graduados en Berkeley y se recogió en la monografía *Basic color Terms* (Berkeley, U. of California Press, 1969). Se analizaron 20 lenguajes pertenecientes a familias lingüísticas no relacionadas entre sí y se recogió documentación de otros 98. Establecieron una serie de criterios, tales como que el significado no pudiera deducirse del significado de alguna de sus partes (esto excluiría a “azulado”, “color limón”, “color salmón” o “óxido de hierro de la chapa de mi coche viejo”), que no estuviera incluido

en otro término más general (esto excluiría a “fucsia”, “carmín” o “escarlata” pues para la mayoría de las personas estos términos son “clases de rojo”), que no se aplicase a una exclusiva clase de cosas (esto excluiría a “rubio” que solo se aplica al color del pelo o al tabaco, aunque pueda ser utilizado excepcionalmente a otro tipo de entidades) y, en fin, que tuviera una clara autonomía como término.

Las conclusiones del estudio, lo que se conoce como la hipótesis de Berlin y Kay, fueron que hay 11 categorías universales que van apareciendo de modo progresivo a lo largo de la evolución. Estas 11 categorías son: blanco, negro, rojo, verde, amarillo, azul, marrón, púrpura, rosa, naranja y gris.

Por añadidura, los autores encontraron otra ley inesperada, y que muestra que cuando un lenguaje cuenta con un número inferior de términos generales para designar colores, la selección se produce con arreglo a ciertas reglas que pueden enunciarse del siguiente modo: 1. Todos los lenguajes contienen términos para el blanco y el negro. 2. Si un lenguaje cuenta con tres términos, uno de ellos es para el rojo. 3. Si un lenguaje cuenta con cuatro términos, cuenta con un término para el verde o el amarillo pero no para ambos. 4. Si un lenguaje cuenta con cinco términos, cuenta con términos para el verde y el amarillo. 5. Si un lenguaje cuenta con seis términos, cuenta con un término para el azul. 6. Si un lenguaje cuenta con siete términos, cuenta con un término para el marrón. 7. Si un lenguaje cuenta con ocho términos o más, cuenta con un término para púrpura, rosa, naranja, gris o alguna combinación de éstos.

Discriminación cromática

Conocer la capacidad de discriminación cromática es fundamental para poder llevar a cabo una representación o una simulación de escenarios reales. Una discusión completa sobre este asunto implicaría varias nociones complejas de las que solo podemos mencionar las principales.

Un concepto clave en psicología experimental, utilizado en muchas áreas, entre ellas la de la visión cromática, es el de la diferencia apenas perceptible, *JND* por sus siglas en inglés (*just noticeable difference*): la mínima diferencia que somos capaces de percibir conscientemente. Esta noción, que se remonta a las leyes de Weber y Fechner, en el siglo XIX, está en la base de la formación de escalas de color y del análisis de la capacidad de discriminación de los colores.

La CIE ha denominado a la distancia entre dos colores en el espacio cromático ideal, d*E* (o ΔE o delta *E*). En principio, este valor se escogió de tal modo que dos colores situados a una distancia de 1,0 serían indistinguibles. Sin embargo, este criterio se ha refinado con los años para integrar heterogeneidades perceptuales del espacio CIELAB (los humanos somos más sensibles a las diferencias de color en unos determinados rangos que en otros). Volveré más adelante sobre la CIE, los tipos de codificación y los diferentes espacios cromáticos normalizados.

En un estudio de Linhares *et al.* que mencionaré más adelante, se utilizaban valores de 0,3 a 0,6 para d*E* a la hora de analizar las respuestas de los sujetos.

La capacidad de discriminación de los colores es un tema muy complejo por lo que la mayoría de los científicos especialistas en la visión de los colores evita dar datos que pueden prestarse a confusión. Es necesario precisar muchos factores: las condiciones de observación, el contexto, o qué es lo que se entiende exactamente por distinguir colores, entre otras cosas. Algunos textos dan la cifra de 10 millones de colores como el número máximo de colores que un ser humano podría llegar a distinguir sin que esta cifra aparezca fundamentada de ningún modo.

Varios estudios recientes apuntan, por otro lado, a una cifra que estaría entre los 2 y 3 millones y que parece bastante mejor fundada (véase las referencias dadas para Linhares *et al*., 2008 y las previas de Pointer, 1998). Pero hay que introducir importantes matices. En primer lugar, que es imposible encontrar una

escena real en la que se dé un número tan considerable de colores pues todas las escenas reales están limitadas a un rango que depende de la reflectancia de las superficies y que depende, a su vez, de la iluminación.

El total de los colores que los seres humanos somos capaces de percibir estaría englobado en un espacio ideal en cuya superficie quedarían situados los denominados a veces colores óptimos, colores puros generados por luces monocromáticas en una longitud de onda claramente dominante o, en el caso de los tonos magentas (que no pertenecen al espectro visible sino que se generan por la acción simultánea sobre la retina de longitudes de onda largas y cortas), por dos longitudes de onda extremas dominantes. Los colores que percibimos corrientemente, Los colores de los objetos, formarían un subconjunto de este espacio ideal delimitado por los colores óptimos.

Cuando se habla del número de colores que un ser humano puede percibir, a menudo esta frase no se refiere a los colores de los objetos sino al límite teórico dado por el espacio ideal delimitado por los colores óptimos. Como este espacio puede ser representado con arreglo a diferentes modelos, es posible obtener una estimación ideal de este número. Los trabajos citados han calculado, como decía, que esta cifra ideal estaria en torno a los 2 millones de colores o algo más. Se ha utilizado para esto el espacio de color del CIE-LAB al que volveré más adelante.

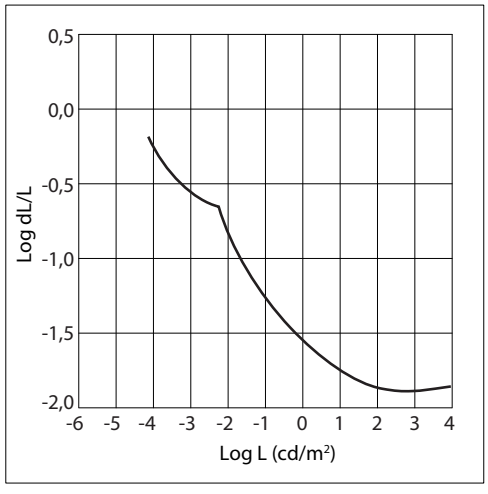


Figura 1.10 Discriminación de luminancias.

Ahora bien, diferentes experimentos con escenas reales (véase sobre todo la última referencia citada, de Linhares *et al*., 2008) han llegado a la conclusión de que el número máximo de colores que pueden percibirse en estos casos es mucho menor. Las estimaciones de estos investigadores para el número de colores incluido en el espacio ideal era similar a la de otros trabajos anteriores y se situaba también en torno a los 2,2 millones para escenarios naturales. Sin embargo, el volumen ocupado por los colores correspondientes al conjunto de escenas reales analizadas estaba en torno a los 690.000, es decir, del orden del 30 % del valor anterior. Pero, por otro lado, el número de colores que podían ser percibidos como diferentes por los sujetos que participaban en el experimento era aún más bajo. Por término medio estaba en torno a los 275.000.

Por otro lado, la gran mayoría de estas diferencias eran diferencias de luminosidad, a la que los seres humanos somos mucho más sensibles. Cuando, en el estudio anterior, se descontaba la luminosidad, de tal modo que solo las diferencias de tono y saturación eran contabilizadas, el número de colores percibidos descendía a una cifra que estaba en torno a 11.000, es decir, alrededor de un 4 % de la cifra anterior.

Otro modo de abordar esta misma cuestión es preguntarse, de modo independiente, por la capacidad de discriminación de las tres variables perceptivas ligadas a los colores: la luminosidad, la tonalidad y la saturación.

La **discriminación de la luminosidad** es difícil de precisar pues depende de las condiciones de adaptación. Hay varios estudios que se remontan a finales del siglo XIX y que se han recogido en la obra clásica de Wyszecki y Stiles (1982), que sintetizan los diferentes resultados en una curva como la de la figura 1.10. Como se observa en esta figura,

hay una ruptura en la continuidad de la curva para una luminancia de unos 5 x 10^{-3} cd/m^2. Esta ruptura deriva de que, para niveles bajos, la visión foveal (conos, visión fotópica) es reemplazada por la visión parafoveal (bastones, visión escotópica). En el extremo opuesto, para niveles superiores a los 100 cd/m^2, la relación *dL/L* (denominada corrientemente relación Weber-Fechner) alcanza sus niveles más bajos y permanece constante en torno a 0,01 (log dL/L: -2). Una estimación simplificada, pero a menudo utilizada para la construcción de escalas de grises, se basa en este valor.

Dicho de otra manera, en unas condiciones de adaptación determinadas, los seres humanos detectan variaciones de luminancia del orden del 1 % con respecto a la luminosidad del fondo. Es decir que si los valores de luminancia extremos son 1000 y 1 se perciben diferencias cuando la relación entre dos valores es más de 1,01. Por ejemplo entre 100 y 101 o entre 900 y 909.

El número de grises que somos capaces de distinguir depende, por consiguiente, del rango de intensidades extremas a que la visión esté adaptada, que es una fracción del rango total posible. Este rango está en torno a los 10^{10}, de los que un máximo de 10^4 correspondería a la visión escotópica y un máximo de 10^6 a la visión fotópica. En condiciones normales este valor es muy inferior. Los valores estimados pueden ir desde 50, que es un valor citado frecuentemente en relación con los valores reproducibles por un dispositivo (entre 5 y 6 bits), hasta un máximo de unos 450. En general, la discriminación sigue la ley de Weber que da una distribución exponencial (o logarítmica), pero en la región inferior los pasos son menores y en la región superior mayores.

Los fotógrafos se basan a menudo en el sistema de zonas utilizado por Ansel Adams (propuesto originalmente por Denman Ross en 1907) que utiliza 9 valores fáciles de recordar y que pueden jerarquizarse en escalas más simples de 3 valores básicos (1-5-9: negro/grismedio/blanco), 5 valores (los anteriores más 3-7, los grises medios de las zonas obscuras y claras) o 9 valores (los anteriores más 2-4-6-8, los grises intermedios entre los anteriores). Véase mi otro libro sobre simulación de la Iluminación, citado en la introducción, para un análisis más completo de la discriminación de la luminosidad, la construcción de escalas y el sistema de zonas.

La **discriminación de tonos** se conoce bien a partir de los experimentos clásicos llevados a cabo por Wright y Pitt en 1934 y por Wright en 1947, utilizando un colorímetro asociado a un campo bipartito. El sujeto tenía que modificar una de las dos mitades, manteniendo igual la luminancia, hasta que se percibía una modificación mínima del tono. Según estos

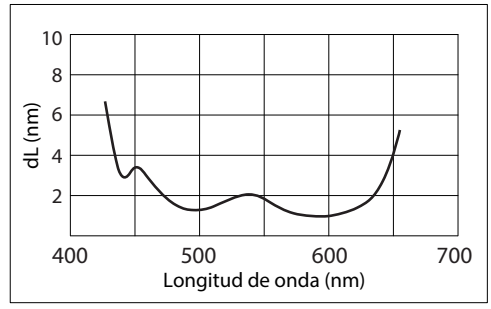


Figura 1.11 Discriminación de tonos.

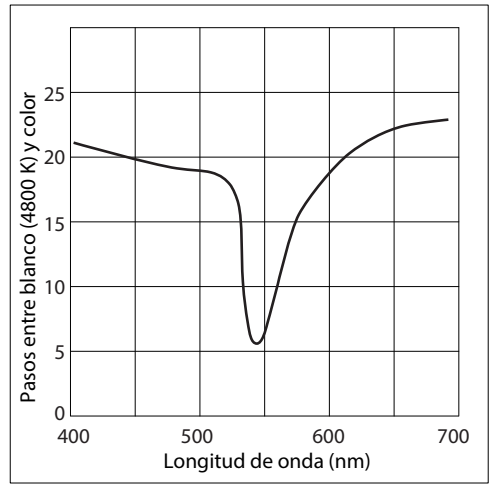


Figura 1.12 Discriminación de saturaciones.

experimentos, que se resumen en la figura 1.11, la discriminación de tonos estaría en torno a los 2 nm en la zona principal del espectro pero se haría más pobre en los extremos, esto es, resulta necesario aumentar el intervalo de longitud de onda para que se aprecien diferencias de tono. Hay unos mínimos en las zonas correspondientes aproximadamente a 480 (azul verdoso) y 590 (amarillo) lo que indica que somos más sensibles a diferencias en torno a estos tonos, algo que tiene que ver muy probablemente con la evolución y la importancia para la supervivencia de la identificación de unos tonos que son propios de alimentos básicos.

De estos valores resultaría que el número de tonos que pueden distinguirse en condiciones óptimas estaría alrededor de los 150, aunque esta cifra de referencia deben tomarse como una indicación muy aproximada.

La **discriminación de la saturación** es aún más difícil de precisar pues unos tonos admiten más niveles de saturación que otros. Los niveles de saturación del amarillo, el más claro de los colores puros, son menores que los de los demás.

La discriminación de saturación se conoce a partir de los trabajos de varios investigadores que también están resumidos por Wyszecki y Stiles (Jones y Lowry, 1926; Martin *et al*., 1933; Wright y Pitt, 1938; Mac Adam, 1942, Kaiser, Comerford y Bodinger, 1976). Los experimentos partían de mezclas aditivas de una longitud de onda dada y un estímulo cromático blanco. Se utilizaba un dispositivo similar al caso anterior que proporcionaba un campo bipartito. Las mezclas se mantenían en un nivel de luminancia determinado. Podían manipularse tanto el color puro como la luz blanca de tal modo que se obtenían gamas de una misma longitud de onda dominante que variaban entre el color puro y un tono acromático de luminosidad equivalente. Los resultados se han referido a dos escalas diferentes, una basada en la pureza relativa a la excitancia y otra basada en la pureza colorimétrica o pureza basada en la luminancia espectral. La diferencia entre ambas está referida a determinados computos técnicos relacionados con la utilización del diagrama cromático CIE 1931 y pueden pasarse por alto, entendiendo que son dos modos más precisos de referirse a la noción genérica de saturación. En general, tal como se muestra en la figura 1.12, se registra un mínimo de pasos, alrededor de 6, en torno a los 570 nm (amarillo) para pasar de un blanco de temperatura de color de 4.800ºK al color saturado, y este mínimo aumenta rapidamente hacia ambos lados hasta regularizarse en torno a los 20 pasos en ambos extremos del espectro. La capacidad de discriminación también depende del tamaño del campo. Los experimentos de la figura citada se hicieron para un ángulo de 2º. Por debajo de esta cifra la capacidad de discriminación disminuye como sería de esperar al disminuir la participación de los conos foveales.

Diferencias de discriminación en visión escotópica y fotópica

Entre los muchos factores que hay que tener en cuenta a la hora de hablar de discriminación de colores, hay que citar también el fenómeno descrito por el fisiólogo checo J.E. Purkinje (1787-1869) hacia 1825 y que ha dado lugar a una diferenciación corriente entre visión fotópica y estocópica.

En 1823 Purkinje publicó un opúsculo (*Commentatio de examine physiologico organi visus et systematis cutanei*), donde daba a conocer lo que actualmente se conoce como *efecto Purkinje*. Al disminuir la luminosidad, los colores violetas y azules mantienen su brillo en mayor medida que los rojos y naranjas; un jardín lleno de rosas y claveles comenzaría a apagarse al caer la tarde mientras que otro lleno de lirios y pensamientos mantendría su brillo en idénticas condiciones de iluminación. Purkinje observó también que en estas condiciones de baja iluminación o condiciones escotópicas como se denominan actualmente (del griego *skotoma*, “obscuridad”), los objetos de color débilmente iluminados son apenas perceptibles si se los mira de frente

pero resultan claramente visibles como objetos luminosos, sin color, si se desvía la mirada y se los ve con el rabillo del ojo. Un tercer grupo de observaciones, publicadas por esas mismas fechas, estaban referidas a la apariencia cambiante de los objetos coloreados al amanecer.

Las observaciones de Purkinje permanecerían sin explicación hasta que no se amplió el conocimiento de la anatomía de la retina y el papel jugado por los conos y los bastones. Y, con ello, el conocimiento de que el rango de visión normal se modifica al variar las condiciones de iluminación. En condiciones fotópicas, con abundancia de luz, donde la percepción corre fundamentalmente a cargo de los conos, la sensibilidad es máxima para los tonos de color situados en torno a los 555 nm. En condiciones escotópicas, con escasez de luz, donde la percepción corre fundamentalmente a cargo de los bastones, la sensibilidad es máxima para los tonos de color situados hacia los 507 nm, es decir, hay un desplazamiento de la sensibilidad hacia los tonos de onda corta, verdes azulados y azules. La figura 1.13 muestra las curvas de sensibilidad máxima del ojo en condiciones fotópicas y escotópicas.

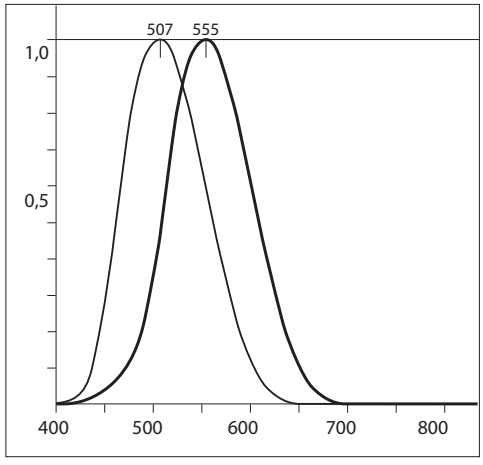


Figura 1.13 Sensibilidad visual en función de la longitud de onda en visión escotópica (línea más gruesa, máximo en 555 nm) y fotópica (línea menos gruesa, máximo en 507 nm).

Deficiencias en la discriminación cromática

Por último, hay que tener en cuenta que la discriminación del color puede venir afectada por deficiencias en la percepción del color. El estado actual de la cuestión es aproximadamente el que sigue. Las personas con deficiencias en la percepción de los colores se dividen en tres grupos: 1) tricromatos (alrededor de un 60 % de los casos anómalos). 2) dicromatos (alrededor de un 8 % de los casos anómalos). 3) acromatos o monocromatos (“ciegos para el color”, menos de un 0,0005 %)

Los tricromatos perciben todos los colores pero de modo deficiente. Se subdividen en tres tipos: protanómalos (diferencian con dificultad el color rojo), deuteranómalos (diferencian con dificultad el color verde) y tritánomalos (diferencian con dificultad el color azul, casos muy raros).

Los dicromatos solo tienen dos tipos de receptores. Se subdividen a su vez en protanopos, deuteranopos y tritanopos. Los protanopos carecen de fotorreceptores sensibles al rojo. Los deuteranopos carecen de fotorreceptores sensibles al verde. Y los tritanopos, una anomalía mucho menos frecuente, carecen de fotorreceptores sensibles al azul.

El término *daltonismo*, derivado de las primeras observaciones que hizo John Dalton (1766-1844), que padecía esta deficiencia, se utiliza de modo general para describir alguno de estos tipos de deficiencia. Se calcula que entre un 6 % y un 8 % de los varones padece esta anomalía y poco más de un 0,4 % de las mujeres. La mayoría de los que la padecen son tricromatos deuteránomalos (alrededor de un 6 % de hombres y un 0,4 % de mujeres) o tricromatos protanómalos (1 % de hombres y 0,01 % de mujeres). Los tritanopos son muy raros: tan solo 1 de cada 50.000 personas padecería esta anomalía. Los acromatos o monocromatos son aún más raros, menos del 0,003 % de la población.

La visión defectuosa es una anormalidad característicamente hereditaria; tanto es así, que resulta ser uno de los tipos de anomalías

que han aportado datos relevantes para el estudio de la transmisión genética de caracteres físicos. El estudio de los modos peculiares de visión de los sujetos con visión defectuosa es considerablemente difícil (particularmente en el caso de los tricromatos) debido a la propia esencia de la visión cromática que impide objetivar datos relativos a la visión normal. Hay ciertos sujetos, extremadamente raros, que han aportado datos decisivos para esta investigación; se trata de personas que tienen visión defectuosa en un solo ojo, lo que les permite comparar campos de visión cromáticamente distintos y comunicar con cierta exactitud las diferencias que observan.

Fenómenos de constancia cromática

Si el color se percibe por la interacción de la luz, los objetos y el ojo, es obvio que se trata de un fenómeno notoriamente inestable. Y, de hecho, basta con tomar un hoja de papel de color naranja, colocarla ante nuestro ojos y moverla suavemente hacia un lado y hacia otro mientras se la observa con atención para observar que los tonos, las intensidades, las saturaciones, varían constantemente. Ni siquiera es necesario que la movamos si está iluminada por luz natural: cualquier pequeño cambio, una nube que pasa, una oscilación de la cabeza, hará que cambie. El análisis científico nos dice que esto es así porque las longitudes de onda reflejadas se modifican constantemente y, en consecuencia, la percepción que recibimos también.

Sin embargo, esta experiencia está, paradójicamente, asociada a otra absolutamente opuesta y más familiar. Y es que, en la experiencia anterior, tendremos presente en todo momento que estamos viendo una hoja de color naranja. Y no solo esto, sino que identificaremos con bastante precisión este color y seremos capaces incluso, más tarde, de distinguirla de otras hojas de un color parecido.

Esta propiedad, que ha intrigado a todos los científicos que se han ocupado de la teoría de los colores, y que resulta vital para los animales que deben sobrevivir en un medio en el que es importante ser capaz de distinguir por su color unos alimentos de otros, no tiene explicación con arreglo a todo lo dicho hasta aquí. En términos de percepción de longitudes de onda, no estamos percibiendo un color estable sino una serie de matices cambiantes.

La explicación más satisfactoria, hasta la fecha, de este fenómeno la proporcionó Edwin Land (1909-1991) en una serie de experimentos famosos que se publicaron en 1959 y años posteriores (véase Land, 1959, 1971, 1977 y 1983). En un experimento denominado *experimento Mondrian* realizado hacia 1964, se presentó a una serie de sujetos una composición de rectángulos de diferentes colores planos, mates, “un Mondrian” iluminado por tres proyectores rojo, verde y azul, de intensidad graduable. El experimento, que no voy a explicar con detalle pero que puede encontrarse en el artículo original y en muchas publicaciones y también en internet, llevó a la notable conclusión de que, si se ajustaban las intensidades de los proyectores, el color que emite cualquiera de los rectángulos es variable. Y, observado de modo independiente, por medio de dispositivos que permitían aislarlo de los que le rodeaban, se reconoce como el color correspondiente a la composición de las intensidades de los tres proyectores. Así, un rectángulo verde puede aparecer como amarillo, rojo, azul, etc., en función de la luz con la que se le ilumine. Sin embargo, en cuanto se suprime el aislamiento la vista reconoce automáticamente el color original, verde en este ejemplo.

La conclusión inmediata es que cuando una superficie forma parte de una escena compleja con diferentes colores, el color percibido no se corresponde exclusivamente con la composición espectral del color recibido por los receptores retinianos. Una conclusión más elaborada es que el color percibido se corresponde con una evaluación llevada a cabo de algún modo por nuestro sistema visual en la que la distribución espectral de la luz reflejada por un fragmento de la escena, uno de los rectángulos de este experimento,

se compara con la distribución espectral reflejada por los fragmentos que le rodean.

El color de la superficie de un objeto depende de una constante principal que, de algún modo, nuestro sistema visual es capaz de percibir con notable precisión. Esta constante es la reflectancia de la superficie. Si la luz que recibe esta superficie es muy intensa, la luz reflejada será también más intensa, y si la luz es menos intensa la luz reflejada será menos intensa. Pero dado que los objetos circundantes proporcionan claves sobre la intensidad de esta luz y la reflectancia es constante, parece que nuestro sistema visual es capaz de descontar la mayor o menor intensidad, para mantener constante la percepción de la superficie. Y otro tanto ocurre con la tonalidad cromática de la luz: si la tendencia es azulada, fría, esta tendencia se captará por claves dadas por el entorno y se descontará de la correspondiente variación sufrida por la reflectancia. Y lo inverso ocurrirá si la tendencia es rojiza, cálida.

La teoría *retinex* de Land parte precisamente de esta hipótesis: que nuestro sistema visual (comprendido de algún modo entre la retina y el córtex, de ahí el nombre) es capaz de comparar registros de luminosidades de una escena entre tres bandas de longitudes de onda, larga, media y corta (correspondientes a nuestros tres receptores cromáticos) y, a partir de ahí, reconstruir el color de las superficies. Es decir, que el color percibido resulta, para decirlo en los términos utilizados por Zeki en otra obra de referencia fundamental sobre la percepción de los colores (véase Zeki, 1993, p. 275 de la trad. esp. de 1995), de *una comparación de comparaciones*: "la primera consiste en comparar la reflectancia de distintas superficies para una luz de la misma banda de ondas, generando así el registro de luminosidad de la escena para esta banda, y la segunda, en comparar los tres registros de luminosidad de la escena para diferentes bandas de ondas, dando así lugar al color". No hay por tanto, como se creía en las teorías clásicas, adición o mezcla sino tan solo relación, comparación. Esta capacidad del

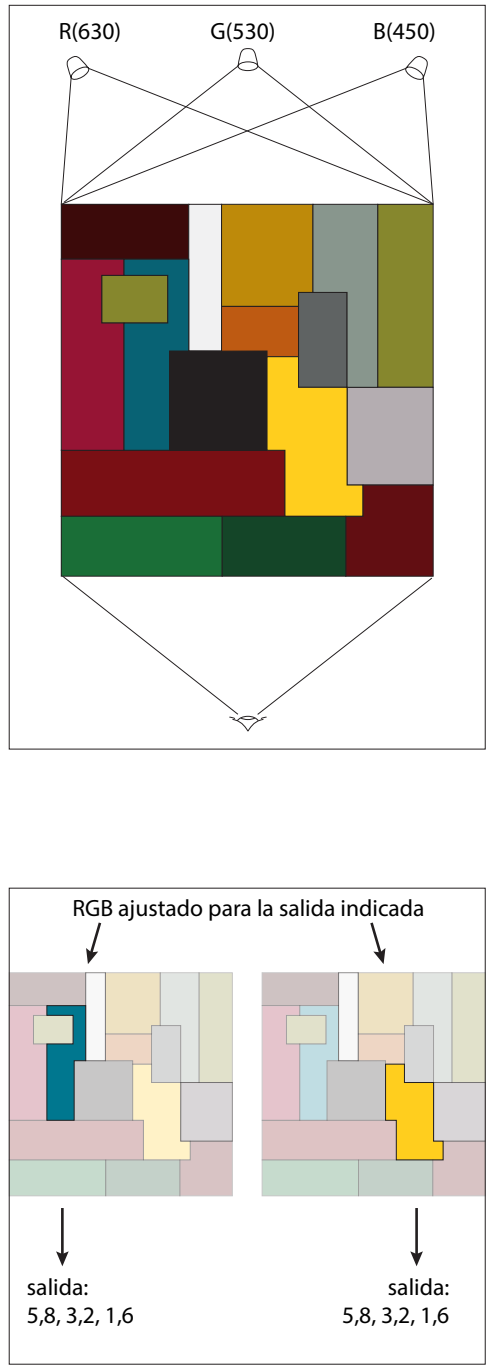


Figura 1.14 Constancia cromática. El experimento de Land.

cerebro de descontar la iluminación es lo que posibilita lo que se denomina constancia cromática.

Las investigaciones posteriores han confirmado los experimentos de Land con algunos matices. Análisis más precisos han mostrado que hay una cierta desviación de la constancia cromática inducida por las condiciones generales de iluminación. Alguno de los fenómenos mencionados por Fairchild a los que me he referido en el apartado sobre apariencia cromática tendrían que ver con esta desviación.

Pero la idea básica, que está estrechamente relacionada con todo lo que se ha aprendido a partir principalmente de las investigaciones de Hubel y Wiesel, es que las neuronas se disparan mediante mecanismos combinados de excitación/inhibición cuya función primordial, en este como en otros fenómenos, es comunicar relaciones. Y la constancia de las relaciones está en la base de los fenómenos de constancia cromática.

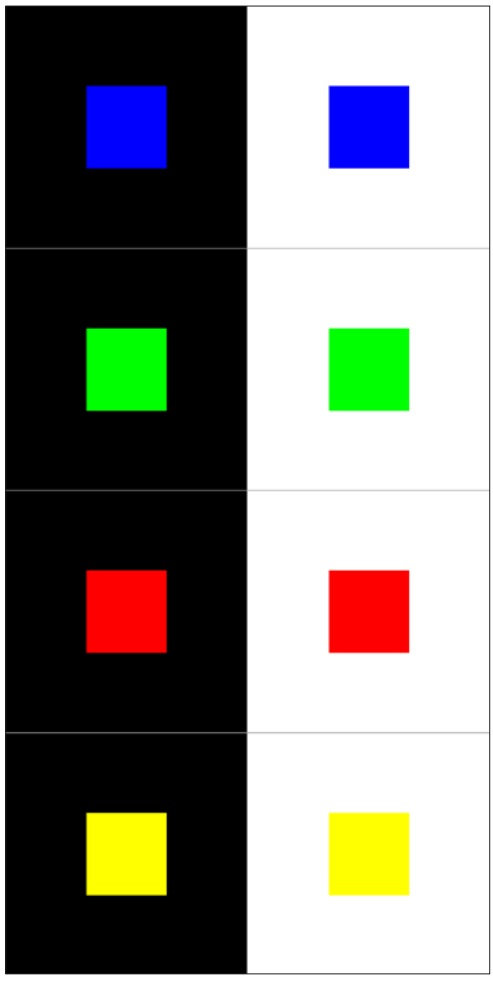


Figura 1.15 Interacción cromática. El cuadrado gris tiene la misma composición espectral en las cuatro figuras.

Fenómenos de interacción cromática

Sin embargo, los fenómenos de constancia cromática descritos en el apartado anterior deben conjugarse, por difícil que resulte, con los fenómenos de interacción cromática.

El color percibido de una superficie no está determinado solo por su color propio sino por el color de una estimulación previa o por el color de lo que rodea a la superficie. Estos efectos han sido ampliamente estudiados y tienen una gran importancia, tanto teórica como práctica.

Si se contempla durante cierto tiempo, por encima del medio minuto, un rectángulo de color rojo y después se desvía la vista hacia una superficie neutra, gris, se verá durante unos instantes un rectángulo de color azul verdoso: una imagen del color complementario al color observado. Este fenómeno, que ahora se denomina *inducción* o *contraste sucesivo*, se ha observado desde hace mucho tiempo pero no se ha explicado de un modo convincente hasta hace poco.

Hay dos procesos principales que intervienen en este fenómeno: el blanqueamiento y adaptación de los receptores superficiales y la reestructuración neuronal combinada con una organización cromática basada en colores oponentes. Lo primero ya fue avanzado en el siglo XIX por Helmholtz: al observar durante un período prolongado un estímulo intenso, como un color rojo, los pigmentos de la retina se saturan, se blanquean. Al modificar bruscamente el

estímulo, estos pigmentos no responden adecuadamente, lo que lleva a un desequilibrio que aumenta el papel de los otros receptores. Sin embargo esta explicación es insuficiente pues a menudo basta un período de observación no tan prolongado. Y, por otro lado, también se da el fenómeno opuesto como veremos a continuación. Hering, hacia 1872, avanzó la segunda causa. La percepción de los colores está basada en la actividad de células que trabajan por oposición: se activan ante la presencia de un color y se inhiben ante la presencia de su complementario. Cuando la excitación prolongada se suprime hay un rebote hacia la inhibición. Y a la inversa. Dado que las oposiciones son blanco/negro, rojo/verde, amarillo/azul, la supresión brusca de un estímulo provoca la activación del opuesto. La intuición de Hering ha sido confirmada por los avances más recientes en la investigación sobre el funcionamiento de las neuronas que se disparan o se inhiben en función del tipo de estímulo que reciben.

Por otro lado, las postimágenes también pueden ser del mismo color del original, lo que se conoce como *asimilación*. Las causas en este caso son más complejas, como lo es el propio fenómeno que solo se da en ciertas circunstancias, por ejemplo cuando el estímulo es rápido y brusco.

Otro fenómeno más corriente y más importante en la práctica es el de la *inducción* o *contraste simultáneo*. Fue analizado en profundidad, quizás por primera vez, por Goethe, en varios pasajes de su *Teoría de los Colores*, publicada a principios del siglo XIX y por Chevreul en una obra voluminosa, *De la loi du constraste simultanée des couleurs*, publicada en 1839 y que influyó considerablemente en la obra de muchos pintores de finales de siglo. Otro referente fundamental es la famosa obra de Josef Albers, profesor de la Bauhaus, *Interaction of Color*, publicada en 1963, donde analizaba las variaciones de tonos percibidos que se obtenían al combinar hojas de colores puros recortadas y compuestas de diferentes modos.

La obra de Hering, publicada por primera vez en 1874, es, como ya he dicho, una contribución más rigurosa a la explicación de este tipo de fenómenos. Las ideas de Hering fueron desarrolladas por G.F. Müller en la década de 1930 y por Hurvich y Jameson (1957) durante el siglo XX y, además de analizar con precisión los fenómenos de interacción, supusieron, como ya he dicho, una contribución muy importante a la comprensión del modo en que percibimos los colores.

La figura 1.15 muestra las notables diferencias que aparecen cuando un mismo color es rodeado por diferentes colores. Las interacciones laterales afectan de un modo evidente tanto a la intensidad (el nivel de gris equivalente percibido) como a la tonalidad, que se desplaza en dirección opuesta a la del estímulo lateral.

De hecho, esto es una prueba más de hasta qué punto la percepción de un color solo tiene sentido si consideramos el contexto en que se inscribe. Un rectángulo negro situado en un exterior a la luz del día se ve negro. Pero refleja mucha más luz que un rectángulo blanco situado en un interior iluminado por luz artificial. Lo que percibimos es la reflectancia que nuestro sistema visual “deduce” del entorno. Pero en casos muy concretos, como los que estamos comentando, hay un cierto desplazamiento del efecto que nuestro sistema visual no es capaz de compensar.

1.3 Causas físicas de los colores

Lo principal de esta sección está adaptado del libro de Kurt Nassau, *The Physics and Chemistry of Color. The Fifteen Causes of Color*, editado por primera vez en 1983. He intentado reducir al mínimo las precisiones técnicas.

Nassau clasifica las causas de los colores en 15 categorías agrupadas en 5 grupos (aunque ha modificado esta clasificación varias veces, en sucesivas ediciones). En la mayoría de los casos, la causa principal es la transición de estado de los electrones debido

a que la energía requerida para este cambio de estado se relaciona directamente con cambios visibles en la longitud de onda. Los 5 grupos son los siguientes:

I Colores debidos a vibraciones y excitaciones simples (colores producidos por 1. Incandescencia, 2. Excitaciones de gases, 3. Vibraciones y rotaciones).

II Colores que implican transiciones en campos ligantes (4. Causados por transiciones metálicas en compuestos, 5. Por transiciones metálicas con impurezas).

III Colores que implican transiciones en orbitales moleculares (6. En moléculas orgánicas, 7. En transferencias de carga).

IV Colores que implican la teoría de bandas (8. En metales. 9. En semiconductores puros, 10. En semiconductores dopados, 11 En centros de color).

V Colores que implican a la óptica geométrica y física (12. Dispersión refractiva, 13. Dispersión y efectos no lineales, 14. Interferencia, 15. Difracción.

La clasificación que he adoptado en los apartados siguientes está más orientada por la finalidad de este libro y simplifica y reordena la de Nassau.

La causa inmediata de los colores, como hemos visto en el apartado anterior, es que nuestro sistema visual es sensible a determinadas longitudes de onda, y que cuando la energía que reciben se encuentra distribuída de un determinado modo vemos diferentes colores.

Los colores simples pueden simularse por medio de tres colores primarios como veremos en el capítulo siguiente. Pero para simular el modo en que los materiales reflejan la luz esto es insuficiente. Una revisión sumaria de las causas físicas del color ayudará a ordenar los métodos de simulación y, sobre todo, a apreciar mejor sus limitaciones.

Estas causas dependen, en la gran mayoría de los casos, de complejas interacciones que hasta hace poco eran desconocidas y que solo recientemente se han comprendido suficientemente gracias a los avances de la física.

Al comienzo de este capítulo he recordado que los principales fenómenos son, entre otros, la reflexión, la refracción y la absorción. Pero ¿qué quiere decir exactamente que una superficie refleja, refracta o absorbe una determinada longitud de onda? Para que esto tenga más sentido habrá que ampliar algo más semejante explicación.

Podemos situar el inicio de esta explicación en 1905 cuando en un famoso artículo Einstein demostró que el efecto fotoeléctrico, esto es, la emisión de electrones por determinados metales como respuesta a la incidencia sobre su superficie de la luz (u otra radiacion electromagnética), que había llevado a contradicciones insuperables para las teorías clásicas, podía explicarse satisfactoriamente admitiendo que la luz se emite y se absorbe en forma de cuantos de energía o fotones.

En 1912, Niels Bohr incorporó la teoría cuántica de la radiación, iniciada por Planck en 1901 y confirmada por los trabajos de Einstein sobre el efecto fotoelétrico, a la presentación de un modelo del átomo en el que los electrones se situaban, sin radiar energía, en ciertas órbitas o estados estacionarios. Estas órbitas configuraban una serie de capas electrónicas discontinuas, con niveles de energía discretos (designadas desde entonces por las letras K, para n = 1, L, para n = 2, M, para n = 3, N, para n = 4, etc.), separadas por zonas en las que no podía haber electrones. La radiación se produce cuando un electrón pasa de una órbita a otra. El modelo de Bohr fue corregido en 1916 por Arnold Sommerfeld considerando órbitas elípticas y desplazamientos no regulares de los electrones en torno a estas órbitas, introduciendo números cuánticos secundarios que permitían explicar determinados fenómenos que aparecían al observar el espectro de algunos elementos con aparatos de gran resolución por medio del momento angular orbital del electrón. Esto fue seguido por ajustes posteriores que llevaron a establecer que la energía de un electrón en un átomo queda unívocamente determinada por cuatro números cuánticos (simplificando: el primero, *n*, es el índice de la órbita o nivel;

el segundo, *l*, el índice del subnivel; los dos siguientes cuantifican el momento angular y la orientación).

En 1925, Wolfrang Pauli enunció lo que se conoce como principio de exclusión, según el cual en un mismo átomo no pueden existir dos electrones con la misma energía, es decir, con los cuatro números cuánticos iguales. Este principio resultó fundamental para conocer la configuración electrónica de los átomos y, en consecuencia, la relación entre esta configuración y la tabla periódica de los elementos que había dado a conocer Mendeleiev en 1869, basada en la masa atómica. Como derivación de este principio se dedujo el número máximo de niveles de cada capa, que debe ser $2n^2$.

Así, para capa, resultan los siguientes números máximos. En el nivel 1 (capa K) el total de electrones admisibles es 2. En el nivel 2 (capa L) el total de electrones admisibles es 8. En el nivel 3 (capa M) el total de electrones admisibles es 18. En el nivel 4 (capa N) el total de electrones admisibles es 32. El número total de electrones se corresponde con el número atómico del elemento. Así, por ejemplo, el Cloro, de número atómico 17, tiene sus correspondientes 17 electrones distribuidos en tres niveles principales: 2-8-7. La distribución de los electrones en cada capa sigue una serie de reglas adicionales. La que más nos importa para lo que sigue es que el número de electrones de la capa más externa, la de máxima energía, no puede ser superior a 8.

Los electrones situados en la última capa se denominan electrones de valencia pues depende de ellos la menor o mayor facilidad del elemento para formar enlaces con otros elementos. Estos enlaces se pueden dar de varios modos, sea por intercambio, sea por compartición, sea por interacción, como ocurre en los enlaces metálicos.

Aunque los colores se asocian corrientemente a longitudes de onda o frecuencia, para lo que sigue será preferible convertir las longitudes de energía a unidades de energía, lo que implica pasar de la consideración de la luz como algo de naturaleza ondulatoria a algo de naturaleza corpuscular o, si se prefiere, de la mecánica ondulatoria a la mecánica cuántica, una dualidad que, como es bien sabido, está presente en las teorías fisicas sobre la luz durante todo el siglo XX. La menor cantidad de luz para cualquier longitud de onda es el fotón. La energía de un fotón se mide en electronvoltios y es directamente proporcional a la frecuencia e inversamente proporcional a la longitud de onda. Un electronvoltio se define como la energía cinética que adquiere un electrón al ser acelerado en el vacío por una diferencia de potencial de 1 voltio y su valor en julios es 1 eV = $1{,}6022 \times 10^{-19}$ J.

La tabla de la figura 1.18 da las equivalencias entre zonas de colores corrientes, longitudes de onda (nm, nanómetros: 10^{-9} metros), frecuencia (Hz, hercios) y energía (eV, electronvoltios).

En la interacción de la radiación electromagnética con la materia se dan tres procesos fundamentales: absorción, emisión espontánea y emisión estimulada.

La *absorción* tiene lugar cuando un electrón asimila la energía de un fotón y pasa a un estado excitado, de mayor energía. La *emisión espontánea* tiene lugar cuando, como consecuencia de estar en un estado excitado inestable, después de absorber un fotón, se emite un fotón de la misma frecuencia del absorbido aunque en dirección y fase aleatorias, y vuelve a su estado original. La *emisión estimulada* (prevista teóricamente por Einstein en 1916) se basa en la incidencia de un fotón sobre un electrón previamente excitado que estimula la emisión y tiene como resultado dos fotones de la misma frecuencia viajando en fase en la misma dirección. Si en su camino se encuentran con electrones excitados del mismo modo, el proceso se repetirá añadiéndose nuevos fotones a los iniciales. En este efecto se basa el láser (siglas de *Light Amplification by Stimulated Emission of Radiation*).

Todos estos conceptos básicos se hacen inevitablemente más complicados cuando consideramos una estructura de átomos compleja como la que se da en la superficie de

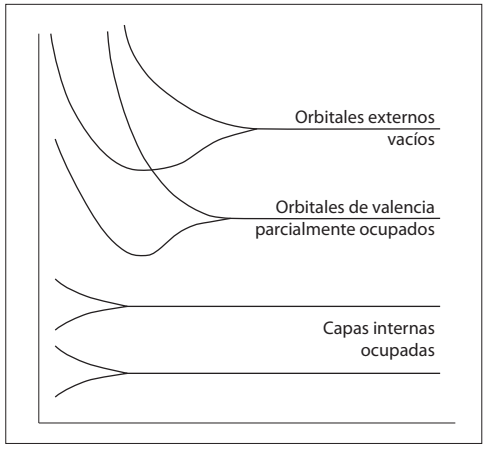


Figura 1.16 Orbitales.

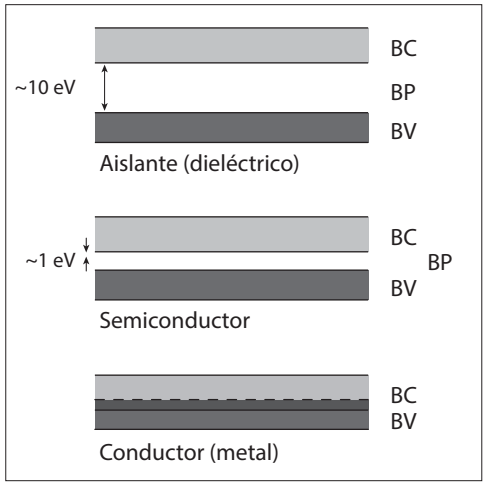


Figura 1.17 Bandas.

Color	*(nm)*	*(x10^{14} Hz)*	*(eV)*
Rojo	700	4,29	1,77
Rojo	650	4,62	1,91
Naranja	600	5,00	2,06
Amarillo	580	5,16	2,14
Verde	550	5,45	2,25
Cyan	500	5,99	2,48
Azul	450	6,66	2,75
Violeta	400	7,50	3,10

Figura 1.18 Tabla de equivalencias entre longitudes de onda, frecuencia y energía.

un material. Si en lugar de considerar un solo átomo consideramos dos, nos encontraremos con que la interacción entre ellos produce el desdoblamiento de un mismo nivel en dos niveles de energía distinta. Este desdoblamiento depende de la distancia y es tanto más intenso cuanto menor es la distancia entre los átomos. En un sólido, los electrones no se pueden mover libremente, como en el espacio libre, sino que su movimiento está restringido a ciertos niveles energéticos que se organizan en bandas.

Si consideramos una red cristalina, la interacción entre los átomos hace que los niveles básicos se separen en tantos niveles como átomos que, debido a su gran número, se puede considerar que forman una banda de energía casi continua. Para los niveles internos, esta perturbación es muy pequeña comparada con la interacción con el núcleo y la separación será irrelevante. Pero para los electrones externos, los electrones de valencia, la separación será grande, tanto mayor cuanto mayor sea la energía del nivel. Según las características del material, estas bandas pueden solaparse. Y el comportamiento de estas bandas explica la mayor parte de las propiedades de las superficies de los materiales.

Para contribuir a la conducción, un electrón debe poder desligarse de su átomo y acelerarse al recibir energía o bien ser excitado a un nivel de mayor energía. En una banda completamente llena esto no puede ocurrir. Los electrones solo pueden desplazarse a niveles próximos vacíos. A la banda de energía más baja que no está llena se le denomina *banda de conducción*. Y a la inmediatamente inferior se le denomina *banda de valencia*. La figura 1.17 muestra un esquema de esta situación. La estructura de bandas se relaciona directamente con la diferencia entre aislantes (dieléctricos), semiconductores y conductores (metales) que se representa también esquemáticamente en dicha figura.

La parte superior de esta figura representa una superficie en la que la banda de conducción, BC, está separada de la banda de

valencia, BV, por una amplia banda prohibida (*band gap*). En un aislante ideal la banda de conducción está vacía mientras que la banda de valencia está llena, con todos sus subniveles ocupados, de modo que no puede contribuir a la conducción. En un aislante no ideal, si algunos electrones recibieran energía suficiente podrían llegar a pasar a la banda de conducción aunque las intensidades generadas serían muy pequeñas.

La parte media de la figura representa una superficie en la que la separación entre la banda de conducción y la de valencia es pequeña, del orden de 1 eV. Esta disposición es característica de los semiconductores en los que sus propiedades varían con la temperatura. A 0 K, la banda de valencia está llena y la de conducción vacía. El material se comporta, por consiguiente, como un aislante. Sin embargo, al aumentar la temperatura, al ser pequeña la separación algunos electrones adquieren la energía suficiente para saltar de la banda de valencia a la de conducción. Y los electrones que abandonan su posición dejan huecos en la banda de valencia que también contribuyen a la conducción.

La parte inferior de la figura representa una superficie en la que la banda prohibida ha desaparecido pues las bandas de conducción y de valencia se solapan. En esta situación, característica de los metales, los electrones pueden moverse libremente bajo la influencia de un campo eléctrico.

Este es el marco general en que deben situarse la mayoría de los fenómenos que se describen en los apartados siguientes y que son percibidos como colores por nuestro sistema visual.

Incandescencia

El término *incandescente* se utiliza para designar específicamente a los objetos que emiten luz por la acción del calor y que tienen, por consiguiente, una determinada temperatura de radiación. En general, cuando un cuerpo se calienta por encima de una determinada temperatura, el electrón se aleja de su núcleo y cuando vuelve a su órbita emite fotones. Este proceso de oscilación se repite continuamente mientras el objeto reciba la suficiente energía para mantener esta temperatura. La velocidad de oscilación y la temperatura de color ligada a esta mayor o menor velocidad determinan el color de la luz. Hay otros objetos que pueden emitir luz no incandescente. Este tipo de emisión se denomina luminiscencia y se tratará en otro apartado.

Si calentamos en la obscuridad un trozo de hierro, sus partículas se irán agitando progresivamente hasta que llegará un momento en que comenzará a hacerse visible una coloración roja que culminará en un blanco brillante. Para un teórico cuerpo negro, que no reflejase ninguna radiación, al llegar alrededor de los 800 Kelvin (525 ºC) comenzaría a hacerse visible un color rojizo, que se haría más oscuro al llegar a los 1.000 K. A partir de aquí se iría haciendo progresivamente más claro, de un tono cereza al llegar a los 1.200 K, naranja hacia los 1.500 K y, finalmente, blanco brillante hacia los 1.800 K. La progresión no avanza directamente hacia los colores fríos sino que pasa por el blanco. Sería preciso calentar este teórico cuerpo negro hasta los 12.000 K aproximadamente para que comenzase a apreciarse claramente una tonalidad azul. Esto es así porque las radiaciones anteriores permanecen y, al mezclarse entre sí, sin que ninguna de ellas domine notoriamente sobre las otras, producen el color blanco, pues la luz blanca no es sino la manifestación de lo heterogéneo de sus componentes como ya he recordado al comienzo de este capítulo.

En 1900, Max Planck obtuvo unas ecuaciones que marcarían el inicio de la teoría cuántica, que permitían calcular la distribución de longitudes de onda en un cuerpo negro ideal. Un cuerpo negro es un objeto teórico que absorbe toda la energía que recibe. La radiación que emite se debe por consiguiente tan solo a la temperatura. Esto permite, entre otras cosas, expresar la distribución de longitudes de onda en términos de valores de temperatura. La figura 1.20 muestra las distribuciones espectrales del cuerpo negro a diferentes tem-

peraturas, normalizadas en torno a la longitud de onda de 560 nm para que puedan compararse pues los valores verticales serían muy dispares. Y la tabla de la figura 1.19 muestra las temperaturas de color de diferentes objetos de referencia.

La temperatura de color relaciona la calidad, el tono visible o la sensación cromática con una determinada cantidad física; decir que una bombilla corriente de 60 watios tiene una temperatura de color de 2.800 K, el Sol una temperatura de color en torno a los de 5.500 K y el cielo azul de un día de verano unos 10.000 - 20.000 K nos precisa la fuente de luz que estamos utilizando en función de la temperatura de un cuerpo ideal que tuviera una distribución espectral relativa conocida.

Las fuentes de luz incandescentes más primitivas son las debidas al fuego y están presentes en todo tipo de imágenes, antiguas y actuales: fogatas, chimeneas, antorchas, velas, etc. Los colores de estas fuentes de luz

Fuente de luz	*Temperatura de color (º K)*
Luz diurna (cielo azul)	12.000 - 20.000
Luz diurna, verano (sombra)	8.000
Luz diurna, verano (sol+cielo)	6.500
Fluorescente “luz de día”	6.300
Lámpara de xenón	6.400
Cielo cubierto	6.000
Lámpara de mercurio	5.900
Sol (mediodía, verano, latitudes medias)	5.400
Fluorescente blanco	5.200
Photoflood “luz de día”	5.000
Sol (amanecer/atardecer)	4.300
Lámpara de mercurio (White Deluxe)	4.000
Photoflood	3.400
Halógena 100 W	3.000
Incandescente 100 W	2.870
Incandescente 40 W	2.500
Sodio alta presión	2.100
Sol (amanecer/atardecer)	2.000
Llama de vela	1.900
Llama de cerilla	1.700

Figura 1.19 Temperatura de color de diversas fuentes de luz.

muestran todo tipo de matices que son muy difíciles de reproducir por medios artificiales. El color de la llama de una vela, en concreto, muestra varias tonalidades que van del amarillo al azul pasando por el rojo y el blanco. Matices similares pueden observarse en quemadores de gas. La figura 1.21 muestra la llama de una vela en la que se han señalado tres zonas características: 1) la zona interior, oscura, donde pequeñas partículas de carbón, de un tamaño que puede estar en torno a los 30 nm, se generan a través de reacciones complejas a temperaturas del orden de los 800º a 1000º; 2) la zona central superior donde la combustión alcanza temperaturas del orden de los 1.200º a 1.400º y hace que estas partículas entren popiamente en incandescencia y emitan luz de un tono amarillento, rojizo, blanco; 3) la zona exterior inferior tiene un característico tono azul que es debido a la mezcla de moléculas inestables con el oxígeno del aire.

Las lámparas incandescentes se basan en que algunos materiales, como el carbono, el platino, el tungsteno o el wolframio, tienen una distribución espectral que se aproxima a la del cuerpo negro. El material preferido desde que se empezaron a fabricar este tipo de lámparas (hacia 1900) era el tungsteno debido a que tiene un punto de fusión alto, por encima de los 3.650 K por lo que se puede utilizar a temperaturas muy altas, en el vacío o en un gas inerte. Su principal inconveniente es que da una luz más bien rojiza, como corresponde a su temperatura de color relativamente baja, que en la figura 1.20 que muestra las distribuciones correspondientes a diferentes temperaturas de color se ve que es más intensa en la región de las ondas largas. Las lámparas incadescentes actuales utilizan filamentos de wolframio. Seguían siendo las más utilizadas, pese a su bajo rendimiento (de 12 a 18 lm/W) y su corta vida (en torno a las 1.000 horas), hasta hace poco que se han eliminado del mercado.

Las lámparas halógenas son variantes de las incandescentes, introducidas a comienzos de la década de 1960, en las que el vidrio se

substituye por un compuesto de cuarzo, que soporta mejor el calor, el filamento puede ser de tungsteno o de wolframio y el interior contiene un gas halógeno (flúor, cloro, bromo o yodo) que evita que el metal evaporado se deposite en el filamento, como ocurre en las lámparas de tungsteno clásicas, que se ennegrecen con el tiempo. El gas utilizado en las primeras era yodo, que daba una coloración rosácea, y posteriormente fue substituido por bromo, que daba una luz más blanca. En cualquier caso, todo esto hace que el rendimiento sea mucho mayor y que las lámparas duren más, y también que la luz sea más blanca (aunque también emiten más en la región ultravioleta por lo que debe evitarse su uso para actividades corrientes, pues pueden acelerar la formación de cataratas.

La fuente de luz incandescente más importante de todas es evidentemente el Sol. Si el Sol no está oculto por las nubes podemos verlo como un objeto luminoso de color variable, filtrado por la atmósfera.

La constante solar es la cantidad de energía recibida, como radiación solar, por unidad de tiempo y unidad de superficie, sobre la parte externa de la atmósfera terrestre, en un plano perpendicular a los rayos del Sol. Se puede calcular dividiendo el flujo energético emitido por el Sol por la relación de áreas entre la superficie del Sol y la de la Tierra. O bien se puede medir por medio de satélites. No es propiamente una constante pues su valor fluctúa a lo largo de un año debido a que la distancia entre la Tierra y el Sol también fluctúa. Y también fluctúa a lo largo de los años debido a variaciones periódicas en el brillo del Sol y a otros factores como las variaciones de parámetros orbitales de la Tierra, particularmente de la excentricidad.

El valor medio de su irradiancia se considera que está en torno a los 1.366 W/m^2 aunque varía en cerca de un 7 % a lo largo del año (entre unos 1.412 W/m^2 a principios de enero y 1.321 W/m^2 a principios de julio).

La temperatura de color del Sol es de 5.770 a 5.780 K. Obviamente la temperatura del Sol en su interior es muy superior. El valor anterior se obtiene a partir de las emisiones espectrales del Sol.

Los objetos con temperaturas de color comprendidas aproximadamente entre los 5.000 y los 7.000 K se perciben generalmente como blancos.

En 1931, la CIE (Commission Internationale de l'Eclairage) emprendió la tarea de codificar los colores de modo que los diferentes industriales pudieran hablar un mismo idioma. Esto supuso normalizar los tres agentes principales de la percepción del color: la luz, los objetos y el ojo. Normalizar los objetos supuso estipular unas determinadas condiciones de observación e iluminación al medir su reflectancia a lo largo de toda la gama espectral. Normalizar el ojo supuso tomar una serie

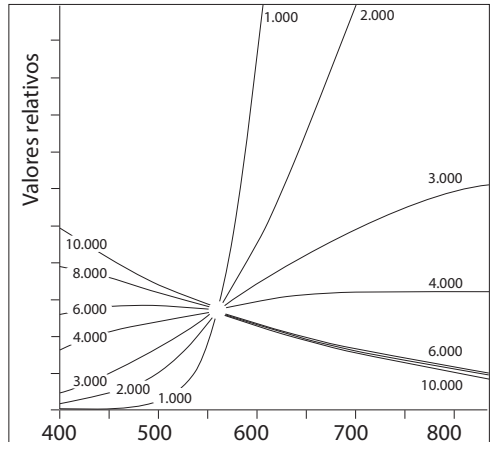


Figura 1.20 Cuerpo negro. Distribución espectral de diferentes temperaturas de color normalizadas tomando como referencia 560 nm.

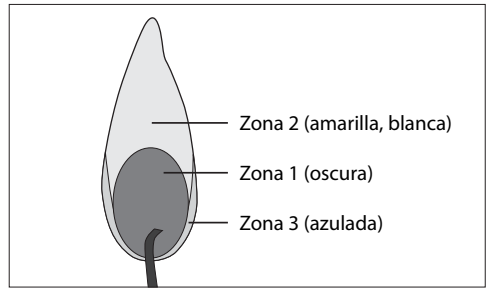


Figura 1.21 Zonas de color en la llama de una vela (véase la descripción en el texto).

de medidas colorimétricas a partir de los datos aportados por una muestra seleccionada de sujetos jóvenes que no sufrían de ningún tipo de anomalía cromática. Y normalizar la luz supuso tomar un determinado iluminante como referencia. El primer iluminante normalizado, el denominado *Standard Illuminant A*, se definió como la luz producida por una bombilla incandescente con una temperatura de color de 2.848 K. Posteriormente este valor se ha ajustado a 2.856 K. Los dos siguientes, el *Standard Illuminant B* (equivalente a la luz producida por la luz del Sol al mediodía, considerada equivalente al *Standard A* más un filtro líquido proporcionando conjuntamente una luz de temperatura de color de 4.870 K) y el *Standard Illuminant C* (definido como la luz del cielo en un día nublado, considerado equivalente al *Standard B* más un filtro especial, proporcionando conjuntamente una luz de temperatura de color igual a 6.770 K) están en desuso y se han substituido por el D65.

Posteriormente, a partir de 1964, se han adoptado los siguientes: El *Standard Illuminant D55*, equivalente a luz de día a 5.000 K. El *Standard Illuminant D65*, equivalente a luz de día a 6.500 K. Es el principal referente utilizado en la actualidad. Y el *Standard Illuminant D75*, equivalente a luz de día a 7.500 K.

La figura 1.22 muestra las curvas correspondientes a los iluminantes A, B, C y D65.

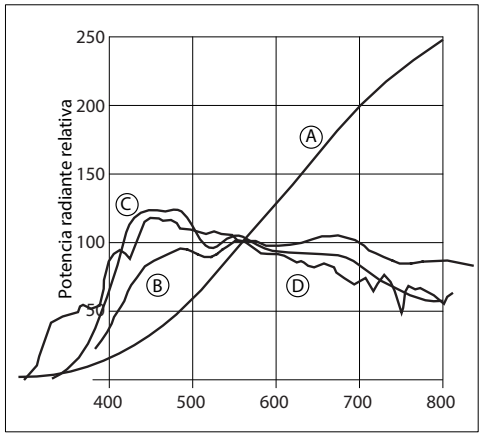


Figura 1.22 Iluminantes normalizados.

Excitación eléctrica de gases. Color de las lámparas de descarga

Hay otros tipos de fuentes de luz que se basan en la excitación de gases. Cuando un gas tal como el neón o un elemento vaporizado, como el sodio o el mercurio, se excita por el paso de una corriente eléctrica, la energía eleva los átomos a estados de energía más elevados, de los que vuelven a caer emitiendo fotones. La explicación del fenómeno es compleja pero el efecto es familiar pues muchos espacios públicos y privados están iluminados con lámparas de sodio o fluorescentes.

Según el grado de energía liberado, el color varía desde el tono blanco rojizo de las luces de neón hasta los tonos azulados de las lámparas de vapor de mercurio o fluorescentes o los tonos amarillentos de las lámparas de sodio. La excitación del gas también puede producirse por medios térmicos o por la interacción de partículas, como ocurre en el caso de las auroras boleares en las que el efecto es debido a la interacción entre partículas emitidas por tormentas solares y gases que se encuentran en la parte superior de la atmósfera terrestre.

Al igual que ocurre con otras causas del color, el fenómeno fundamental deriva de la absorción de energía por un electrón. Supongamos un átomo de sodio que se encuentra en estado de vapor. Este átomo tiene un núcleo con 11 elementos positivos equilibrados por 11 negativos, electrones situados en orbitales alrededor del núcleo, agrupados en 4 capas diferentes. La primera contiene 2 electrones, la segunda 8 (en dos orbitales denominados 2s y 2p) y la tercera 1, situado en la órbita de menor energía. Si se proyecta energía sobre este átomo, el electrón libre de la tercera capa puede excitarse con facilidad para pasar a un estado de mayor energía. Pocos instantes después vuelve a su estado anterior reemitiendo la energía absorbida en forma de luz. La energía característica que se emite en este proceso, en el caso del átomo de sodio es 2.103 eV o 2.105. La tabla de la figura 1.18 muestra que esta energía se corresponde con

589.6 y 589.0 nm, dos longitudes de onda que corresponden al color amarillo.

Las producción de luz como consecuencia de descargas de gases fue observada casualmente en laboratorios por varios científicos a principios del siglo XIX. Pero los primeros experimentos dirigidos a conseguir deliberadamente este efecto se llevaron a cabo por Michael Faraday hacia 1835 y por Heinrich Geissler hacia 1860. Se descubrió así que un tubo con gas en el que se había hecho el vacío mostraba una buena conductividad eléctrica y emitía luz de diferentes colores según el tipo de gas utilizado. En 1910, Georges Claude utilizó estos principios para fabricar los primeros tubos de neón que, desde entonces, se utilizan principalmente para anuncios. Si se utiliza solo neón o neón con argón el color resultante es rojizo. Si se utiliza argón con mercurio el resultado es azulado.

Hacia 1930 se descubrió que si se recubría el interior de estos tubos con polvo de fósforo se obtenía una luz más blanca y que la eficiencia y el coste de estas lámparas eran superiores a las de las incandescentes. Las lámparas flourescentes se utilizaron a gran escala por primera vez en la Feria Mundial de Nueva York de 1939 y desde esa fecha se han introducido todo tipo de mejoras en el rendimiento y en la calidad de la luz. Con todo, la distribución espectral, caracterizada por unos picos muy altos en determinadas regiones, no se corresponde con la de otras fuentes de luz que dan una distribución más similar a la natural.

Hacia 1948 se comenzaron a producir de modo regular, en Estados Unidos, lámparas de vapor de mercurio para alumbrado público. El tono azulado se intentó compensar por medio de recubrimientos interiores de fósforo. En la década de 1970 se introdujeron las lámparas de vapor de sodio a alta presión como alternativa. Los principios son los que se han explicado antes. Su principal ventaja es el alto rendimiento que puede ser de hasta 200 lúmenes por vatio (las lámparas incandescentes no llegan a 20). Su principal desventaja, el tono característico amarillento anaranjado se ha acabado por aceptar debido a sus considerables ventajas en términos económicos.

Las figuras 1.23 y 1.24 muestran las distribuciones espectrales características de estas lámparas.

Por último, hay que recordar que muchos fenómenos naturales obedecen al mismo principio. Los rayos que se producen durante las tormentas se originan por la diferencia de

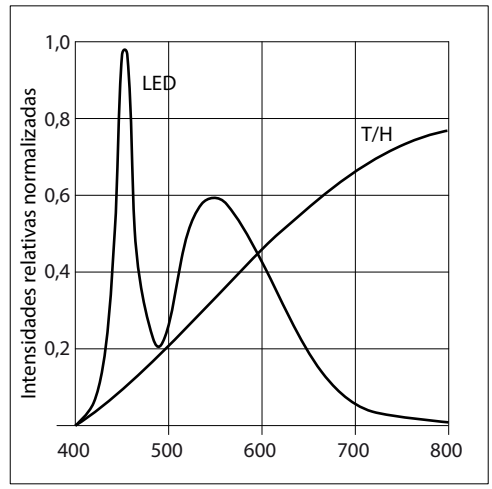


Figura 1.23 Distribuciones espectrales de lámparas de tungsteno/halógenas y led.

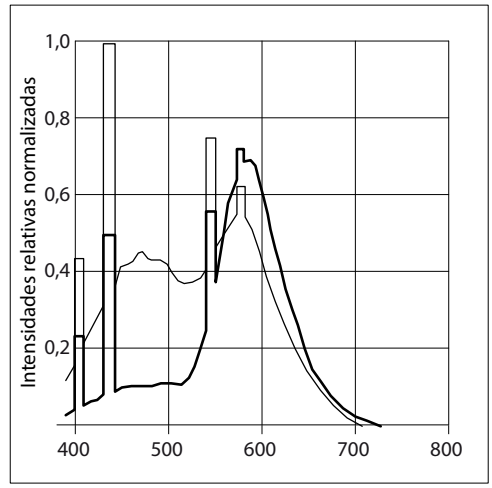


Figura 1.24 Distribuciones espectrales de lámparas fluorescentes y de mercurio.

potencial entre las nubes y la tierra, una diferencia que puede ser del orden de los 100 millones de voltios. En presencia de partículas de gas ionizadas, se produce una descarga que se manifiesta como luz visible. Las auroras boreales, entre otros fenómenos, también se originan por las mismas causas.

Luminiscencia. Color de otras fuentes de luz

En general, se denomina luminiscentes a los fenómenos luminosos no ligados al aumento de temperatura (incandescencia). Esto abarca a dos grupos principales, la fosforescencia y la fluorescencia.

Un material fosforescente (del griego *phóros*, "portador", "de luz", *phos*) es un material que absorbe energía y luego la reemite. Las diferencias tienen que ver con el tiempo que tarda en reemitir esta energía. La vida de un electrón excitado es del orden de 10^{-8} segundos. Si un material responde a la energía incidente volviendo a emitirla en un período de tiempo comprendido dentro de este intervalo de 10^{-8} segundos, hablamos de fluorescencia. Si tarda más, hablamos de fosforescencia, de persistencia en la emisión de la luz después de haber cesado la excitación.

La *fluorescencia* es la propiedad de ciertos cuerpos de absorber la luz y volver a emitirla en forma de radiación de mayor longitud de onda en un intervalo prácticamente inmediato. Es relativamente frecuente pues de 3.000 variedades de minerales conocidas, unas 500 tienen esta propiedad. Unos minerales tienen un factor constante de fluorescencia que contribuye a un color uniforme característico. Otros minerales tienen impurezas denominadas, en este contexto, activadores, que pueden modificar su coloración y su intensidad en la oscuridad. La fluorescencia puede apreciarse artificialmente con lámparas ultravioletas: las cicatrices aparecen de color violáceo, las manchas de nicotina de color rojo, el papel y los tejidos blancos aparecen de un tono blanco azulado. Las lámparas fluorescentes consisten en un tubo de vidrio dentro del cual hay vapor de mercurio. La cara interna del tubo está revestida de una capa de un material fluorescente que, cuando se produce una descarga, absorbe la luz ultravioleta emitida por el mercurio y la reemite como luz visible.

La *fosforescencia* es debida a materiales (como el fósforo), con una notable persistencia, materiales que mantienen durante mucho tiempo esta capacidad emisiva. El término se introdujo a finales del XVII para designar uno de los primeros casos reseñados, el descubrimiento (por parte del zapatero y alquimista de Bolonia, Vincenzo Cascariolo en 1603) de que, lo que se denominó piedra boloñesa (sulfato de bario), emitía luz durante la noche después de haber estado expuesto a la luz durante el día. La mayoría de los materiales fosforescentes son inorgánicos, no contienen carbono y están compuestos por cristales que contienen impurezas que distorsionan la regularidad de la red cristalina. Hay varios procesos que dan lugar a la emisión de luz por materiales que denominamos luminiscentes. La energía emitida es menor que la absorbida y la diferencia se transforma en calor (esto se conoce como la ley de Stokes). Esta pérdida se da tras haber absorbido un fotón y el electrón queda en un estado metaestable hasta que acaba por emitir la energía almacenada. Algunos materiales fosforescentes pueden retener del orden de la mitad de este potencial durante períodos de unos 6 meses, a temperatura ambiente o durante más tiempo si la temperatura es menor. El cristal de fluorita puede llegar a almacenar esta energía durante años. Los sulfuros de calcio, la orina o los huesos calcinados pueden presentar esta propiedad que también se da en algunos hongos o en la luciérnaga.

Además de estos dos grupos principales se pueden clasificar los fenómenos luminiscentes con arreglo a las causas que los producen. Así, se pueden encontrar denominaciones como quimiluminiscencia (cuando es debida a reacciones químicas), fotoluminiscencia (cuando la energía activadora es electromagnética), que correspondería principalmente a fluorescencia inducida por luz visible

o rayos ultravioletas), electroluminiscencia (cuando es causada por una corriente eléctrica), bioluminiscencia (cuando es causada por las funciones propias de un organismo vivo), triboluminiscencia (inducida por medios mecánicos), y unas cuantas más.

Impurezas metálicas en un medio aglutinante. Color de minerales y pigmentos

El color de la mayoría de minerales, gemas, pigmentos, pinturas, es debido a impurezas presentes en un medio que aglutina compuestos metálicos. Estos compuestos están formados por iones metálicos con electrones acoplados en sus órbitas externas que pueden excitarse dando lugar a variaciones en la absorción de determinadas longitudes de onda.

Si las capas orbitales de un átomo están completas, el estado es estable, como ya hemos visto, y se necesitan energías de alta intensidad, en la región ultravioleta, para modificar este estado. De ahí que la mayoría de los compuestos químicos sean incoloros, como ocurre con el cloruro sódico (sal común), el óxido de alumino o alúmina (corindón) o los diamantes. Sin embargo, en otros casos, las capas orbitales están incompletas y pueden ser alteradas por energías de baja intensidad relativa, en la región del espectro visible.

Por ejemplo, el óxido de aluminio cristalino o corindón se encuentra como componente de varios tipos de roca y, sin purificar, se utiliza como abrasivo. Pero el corindón puro, transparente, puede adquirir coloraciones rojas (rubí) o azuladas (zafiro). La estructura atómica está formada por un átomo trivalente de aluminio rodeado por seis átomos de oxígeno formando un octaedro irregular. El campo eléctrico formado por el aluminio se conoce como ligando (*ligand field*) o como campo cristalino (*crystal field*) en la teoría vigente hace algunos años. Si el óxido de aluminio contiene impurezas, corrientemente cromo en el caso del óxido de aluminio (pero que, en otros casos, pueden ser níquel, hierro, cobalto o manganeso), un 1 % de los átomos de aluminio es substituido por cromo y el ligando genera un cambio en los niveles energéticos que se traduce en una mayor absorción de las longitudes de onda corta y la correspondiente coloración rojiza característica del rubí. En el caso de los silicatos de aluminio, una substitución similar del aluminio, en un 1 % a 2 %, por cromo, da lugar a una absorción mayor de las ondas largas que se traduce en una intensificación de las medias (verdosas). Efectos similares dan lugar a los colores de otras piedras preciosas pero también al color

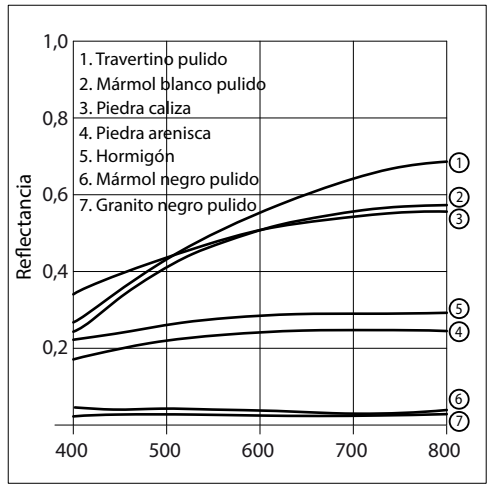


Figura 1.25 Distribuciones de reflectancias de piedras y materiales compuestos.

Material	*Reflectancia media*
Asfalto	0,07
Suelo de tierra	0,08
Suelo de hierba	0,25
Ladrillo rojo	0,15
Ladrillo ocre	0,25
Ladrillo blanco	0,75
Hormigón rugoso	0,20
Hormigón liso	0,30
Enlucido de yeso rugoso	0,40
Enlucido de yeso liso	0,80
Cobre	0,40
Mármol blanco	0,80
Nieve	0,80
Aluminio	0,85

Figura 1.26 Valores medios de reflectancias de algunos materiales de construcción corrientes.

de piedras como la arenisca roja, de algunos lásers y de algunos materiales fluorescentes.

En otros casos, el metal de transición puede estar presente, no como impureza sino como parte propia del compuesto. Esto es lo que ocurre con muchos pigmentos o con las capas que se forman por reacciones químicas sobre la superficie expuesta al aire libre de piedras de edificios o estatuas. Así ocurre en, por ejemplo, el óxido de cromo donde la debilidad del enlace del cromo hace que se requieran energías relativamente débiles para dar lugar a la absorción de ondas largas, lo que se traduce en una coloración verdosa (ondas medias). Algo similar ocurre con materiales de tonos azulados y verdoso-azulados tales como la malaquita, azurita o turquesa.

Un caso especial (que Nassau clasifica en un grupo independiente) se da en algunos compuestos químicos que contienen impurezas debido a fenómenos de transferencia de carga entre las impurezas. En el caso del óxido de aluminio, por ejemplo, puede ocurrir que haya una pequeña proporción de titanio y de hierro. Si ambas impurezas están presentes en la misma proporción, puede ocurrir que la absorción de la luz haga que un electrón se desplace de uno a otro ion, modificando temporalmente la valencia de ambos (aumentando la de uno y disminuyendo la del otro). Durante este proceso hay una absorción de energía en la región de las ondas medias lo que se traduce en la emisión de energía en la región complementaria dando lugar a las coloraciones azules intensas características del zafiro. Fenómenos de este tipo están en la base de colores tales como el pigmento conocido como azul de Prusia (ferrocianuro de hierro) desde el siglo XVIII, el lapislázuli o el azul ultramar.

Compuestos orgánicos. Colorantes naturales y sintéticos

La mayoría de los colores que nos rodean se deben a recubrimientos de la superficie de materiales por medio de tintes o pinturas. Las pinturas utilizan pigmentos minerales disueltos en un medio (clara de huevo en el caso del temple, resinas en el caso de las acuarelas, aceite de linaza en el caso de la pintura al óleo, resinas sintéticas en el caso de la pintura acrílica) que al secarse retiene el pigmento. El color se debe, por tanto, al pigmento y las pinturas no pertenecerían propiamente a este grupo sino más bien al grupo anterior. Pero los pigmentos son caros y por esta razón, entre otras, es mucho más frecuente, desde hace miles de años, el uso de colorantes orgánicos.

Algunos de estos, los que proporcionan un colorido más intenso, tienen una larga historia. El indigo (que actualmente se define como un color situado entre los 450 y los 420 nm), se conoce desde hace más de 3.000 años. El nombre deriva del griego *indikon* con el que se designaba a un tinte que provenía de la India, donde se extraía de ciertas plantas. Pero también se ha encontrado en los ropajes de momias en Egipto. Otro tinte famoso, de color muy similar, es la púrpura de Tiro, extraído de las mucosidades de un caracol marino y utilizado por los antiguos fenicios y, muchos años despúes, por senadores romanos o cardenales católicos para teñir sus túnicas. Y, en fin, otro colorante de tonalidad similar y también muy preciado se extraía de la cochinilla, un insecto que se utilizaba en el Mexico precolombino como tinte y que se envió a Europa a través de España a partir de 1526.

Desde 1856, con el descubrimiento del primer colorante sintético, la anilina, por W.H. Perkin, a partir de derivados del petróleo, comenzó una intensa actividad para obtener tintes más efectivos y más baratos y que cubriesen toda la región del espectro. Algunos de los más utilizados en la actualidad son los de la familia de las ftalocianinas (azules verdosos principalmente), cuya estructura molecular es muy similar a la de la familia de las porfirinas.

En paralelo, se desarrollaron diversas teorías, comenzando por las propuestas hacia 1876 por O.N. Witt, que postulaba la existencia de unos componentes básicos en la estructura molecular de los colorantes,

con dos grupos principales que afectarían de modo directo a la absorción de la luz, los cromóforos y los auxocromos y a cuya combinación se denómino *cromógeno*. Los *cromóforos*, que incluyen carbono con enlaces dobles, incorporan electrones que resuenan a determinada frecuencia y puede decirse que son los responsables del color propio del colorante en cuestión, de las frecuencias que se absorben o se reflejan. Los *auxocromos* son grupos cargados positivamente que intensifican la función de los cromóforos o que desplazan sus picos de absorción a determinadas longitudes de onda. Por otro lado, también contribuyen a mejorar la afinidad del colorante con su soporte, a hacerlo más estable. Estas teorías se han revisado recurrentemente hasta culminar en explicaciones más precisas, que se han dado a partir de la teoría orbital de moléculas, hacia 1950, pero que, en lo substancial, mantienen esta descripción más genérica basada en el papel de los cromóforos y auxocromos.

Metales y semiconductores

El comportamiento de los metales se explica por la teoría de bandas que he introducido anteriormente. Los metales se caracterizan por su baja resistencia eléctrica (son buenos conductores) y por su alta reflexividad. En general, la resistividad metálica, que es un criterio directo para distinguir aislantes, semiconductores y metales, aumenta con la temperatura en el caso de los metales y semiconductores.

Al combinarse los átomos de elementos para formar moléculas, se crean orbitales moleculares que se agrupan en bandas de energía. Los niveles de energía dependen de la distancia entre los átomos. Las figuras anteriores, 1.16 y 1.17, que ya he comentado, muestran el modo en que que estos niveles de energía varían con la distancia y con los tipos de bandas. En las capas internas, plenamente ocupadas y estables, no se subdividen en bandas, a no ser que las distancias entre átomos desciendan hasta valores muy bajos. Sin embargo, en las bandas más externas, parcialmente ocupadas, la reorganización en bandas se da a distancias mucho mayores. En el caso de los materiales metálicos, los electrones de valencia, situados en las bandas externas pueden moverse con libertad, lo que explica la alta conductividad de los metales. Cuando un fotón es absorbido por un electrón situado cerca del nivel superior de esta banda de energía, el electrón asciende a un nivel más elevado. La energía lumínica absorbida induce una corriente eléctrica en la superficie del metal que reenvía el fotón hacia el exterior, lo que es percibido como un brillo característico, propio de los metales: los materiales metálicos son muy reflectantes porque los electrones disipan con rapidez la energía que reciben. También por esta razón la mayoría de los metales tienen un color neutro, gris plateado, pues reflejan la mayor parte de la luz que reciben, con independencia de la frecuencia.

Sin embargo, los diferentes metales y aleaciones tiene diferentes factores de absorción que modifican estas características generales. La plata tiene un coeficiente muy bajo de absorción lo que se traduce en un reflectancia muy alta que disminuye muy ligeramente en la zona de las ondas cortas (azules y violetas), lo que hace que su tono general sea ligeramente cálido debido a la relativa preponderancia de las ondas largas (rojas). Su curva de reflectancia es similar a la del aluminio pero más alta y con una inclinación opuesta. Esto es mucho más notorio en el caso del cobre o del latón (con una curva de reflectancia similar a la del oro pero algo más baja), lo que se traduce en tonos más rojizos y más amarillentos respectivamente. El hierro, por otro lado, tiene un perfil similar al de la plata, bastante neutro aunque ligeramente cálido, pero su reflectancia general es más baja. La figura 1.27 muestra las distribuciones espectrométricas características de algunos metales. El cobre, el oro o el bronce tienen tonalidades cálidas

rojizas (predominio de ondas largas) o verdosas (bronce).

La explicación detallada de estos procesos se dio a partir de 1928 por medio de la teoría de la mecánica cuántica como ya he recordado antes y puede encontrarse en obras de referencia de física del estado sólido, como la de Kittel que se cita en la sección de referencias.

Los semiconductores, como el silicio, tienen características distintas que las de los metales. Pero la explicación de estas características es la misma, se describen igualmente a partir de los principios de la teoría de bandas. Los semiconductores principales tienen 4 electrones de valencia.

Como es bien sabido, lo que caracteriza a los semiconductores es que se comportan como aislantes o como conductores en función de la temperatura debido a que la banda prohibida es estrecha, del orden de 1 eV, lo que posibilita el salto de la banda de valencia a la de conducción.

Un pigmento como el amarillo cadmio tiene una banda prohibida de 2,6 eV que absorbe las longitudes de onda corta, reflejando las largas y medias lo que da lugar a su color amarillo y en otros casos naranja. Una banda similar, más estrecha, es característica del rojo bermellón, que se obtiene a partir del mineral cinabrio (sulfuro de mercurio). Y una banda aún más estrecha es propia de otro semiconductor, la galena (sulfuro de plomo). Esta secuencia de colores: amarillo, naranja, rojo y negro, es caracteristica de los semiconductores.

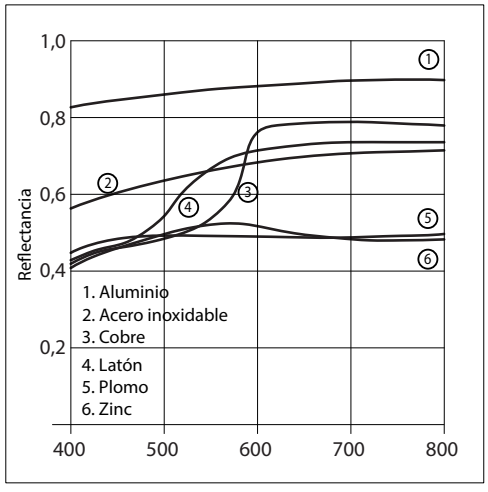


Figura 1.27 Distribuciones de reflectancia de varios metales.

Si en un semiconductor aparece un átomo de otra substancia extraña, una impureza, sus propiedades se modifican. Estos semiconductores denominados *dopados* tienen un número distinto de electrones de valencia, lo que hace que se puedan formar nuevos niveles de energía. Si el átomo impuro tiene más electrones que el átomo al que substituye, como ocurre con el nitrógeno que tiene cinco, en el caso de cristal de diamante (formado por carbono con cuatro valencias) se forma lo que se denomina un donante y el semiconductor será de tipo N (negativo, por exceso de electrones). Los electrones de este nivel pueden excitarse para pasar a la banda de conducción, absorbiendo fotones en la banda de longitudes de onda corta, lo que se traduce en una tonalidad amarillenta. Si la impureza tiene menos electrones que el átomo al que substituye, como ocurre con el boro, que tiene tres, se crea un nivel vacío, un hueco. Los átomos de este tipo se denominan *aceptores*. Y el semiconductor será de tipo P (positivo, por defecto de electrones). Los fotones excitados pueden pasar a este nivel, absorbiendo fotones en la zona media del espectro lo que da lugar a un color azulado característico de algunos diamantes especialmente valiosos.

Algunos materiales pueden tener tanto donantes como aceptores y pueden absorber energía eléctrica o ultravioleta y producir luz visible. Esto ocurre con el sulfuro de zinc contenido en el polvo de fósforo que contiene cobre y otras impurezas y se utiliza como revestimiento de las lámparas fluorescentes (y que sirve para convertir la energía ultravioleta generada por la descarga del mercurio en luz fluorescente), de la parte interior de los tubos de rayos catódicos que se utilizan en monitores de este tipo o de pinturas

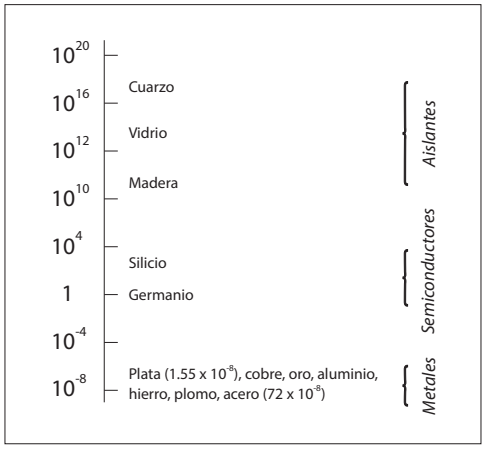


Figura 1.28 Resistividades de metales, semiconductores y aislantes.

fluorescentes. Un fenómeno similar está en la base de la electroluminiscencia que tiene múltiples aplicaciones, entre ellas la de los numerosos aparatos basados en led (*Light Emitting Diodes*).

Casos especiales

En cualquier material, las moléculas tienen cierto grado de energía debido a *vibraciones y rotaciones* debidas a los enlaces químicos. La frecuencia de esta vibración se puede incrementar en determinados casos, como ocurre en el caso del agua pura o del hielo, en donde la fuerza del enlace es superior que en una molécula aislada de H2O. Este aumento de energía puede ser suficiente para absorber las longitudes de onda larga (rojos y naranjas) lo que se traduce en una coloración azul característica.

Las vibraciones y rotaciones moleculares son en general de muy baja intensidad. Los niveles de vibración, denominados a veces *vibrónicos* pueden ser del orden de 100 niveles por cada nivel electrónico. Se manifiesta en casos muy especiales, tales como en las reacciones químicas asociadas a la combustión de hidrocarburos en las regiones de transición externas de llamas de velas o gases o en regiones periféricas de vapor de yodo, con coloraciones violetas o de vapor de bromo y cloro, con coloraciones verde amarillentas.

En el caso del agua y del hielo, los enlaces que se forman entre los átomos de hidrógeno y los dos átomos de oxígeno a los que está ligado, dan lugar a vibraciones complejas que se manifiestan en una mayor absorción de ondas largas y la correspondiente emisión de ondas cortas. Esta es la causa principal de los tonos azulados característicos del agua y del hielo y no, como se sostiene a menudo, el reflejo del azul del cielo, aunque este pueda contribuir a intensificar esta tonalidad básica.

El color puede ser también debido a defectos estructurales como ocurre con los *centros de color*. En algunos minerales puede haber un agujero, un ión ausente de lo que sería su posición normal, debido a causas diversas como una radiación de alta energía, una reacción química o la acción de un campo eléctrico. Estos defectos estructurales se denominan centros de color o centros F (del alemán, *Farbe*, “color”). La fluorita púrpura es un ejemplo. Como la estructura debe permanecer neutra, el hueco es ocupado por un electrón que se mantiene en su posición por el campo eléctrico (campo cristalino) de los iones que le rodean. Dentro de este hueco puede ocupar varios estados excitados y el movimiento entre estos estados es la causa del color y de la fluorescencia óptica de este tipo de materiales, entre los que, además de la fluorita púrpura, también se puede citar el color ahumado de algunos cristales de cuarzo. Las gafas de sol que se oscurecen también hacen uso de este efecto al captar energía ultravioleta del sol. Al igual que ocurre con la fluorita, el aumento de temperatura, en ausencia de rayos ultravioletas que mantengan el efecto, contribuye a restaurar la condición inicial. Otros minerales que muestran este efecto son la amatista, el vidrio de amatista y algunas fluorescencias.

1.4 Variaciones del color en la reflexión y refracción. Otras causas del color

Sin abandonar el análisis puntual de una pequeña zona de una superficie, el color tambien depende de otras causas, notoriamente los cambios de dirección de observación en vertical (variaciones del ángulo vertical de incidencia de la iluminación) y en horizontal (variaciones del ángulo horizontal de incidencia de la iluminación o del ángulo con que se observa una superficie anisotrópica).

Además, hay otras causas del color que no se pueden explicar por la óptica geométrica y que he incluido en este apartado para diferenciarlas con más claridad.

Reflexión, absorción, refracción

Cuando la luz incide en un punto de una superficie, parte de la luz se refleja, parte se absorbe y parte se transmite. Además de estos fenómenos básicos también hay otros, que veremos más adelante, como la dispersión y la polarización. Pero la reflexión y la refracción tienen una importancia principal y requieren un análisis más detallado. Podemos empezar por recordar las leyes clásicas de la reflexión y la refracción.

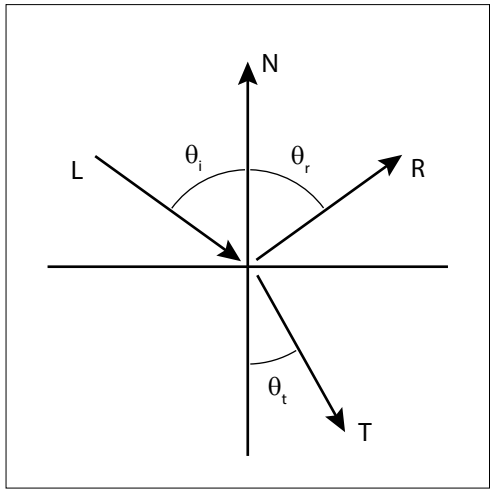


Figura 1.29 Reflexión y refracción clásicas.

La primera de estas leyes, referente a la reflexión, ya era conocida por los griegos aunque no conocemos el nombre del primero que la enunció (es posible que fuera descubierta por más de uno de modo independiente): cuando un rayo se refleja sobre una superficie plana ideal, tal como un espejo perfecto, el ángulo θ_i que la luz incidente forma con la normal a la superficie es igual al ángulo reflejado θ_r (véase la figura 1.30).

La segunda, referente a la refracción, también era relativamente conocida por los griegos pero no llegaron a formularla, aunque se conservan tablas de Ptolomeo y otros en los que se intentaba relacionar los ángulos de incidencia con los de refracción. En 1621, el matemático holandés conocido como Snell (Willebrord Snel van Royen, 1580-1626) descubrió la ley que lleva su nombre y que estipula que la relación entre los senos de los ángulos de incidencia y refracción es constante. Y que esta constante depende de los dos medios en que se propaga la luz. Dicho de otro modo, que en la figura 1.30 se cumple la relación

$$\text{sen } \theta_i / \text{ sen } \theta_r = n_1 / n_2$$

La tabla incluida en la figura 1.31 da los índices de refracción de algunos materiales corrientes.

La teoría ondulatoria propuesta por Huygens en el siglo XVII y retomada principalmente por Young y Fresnel a principios del XIX, da una explicación más compleja de esta relación, basada en la modificación de las velocidades de las ondas del frente que incide sobre la superficie. Mediante una elaboración más detallada, que implica que cada parte del frente que llega al plano se convierte en un emisor secundario, se llega a resultados coherentes con la ley de Snell. La teoría corpuscular clásica, por el contrario, no pudo llegar a establecer esta relación de un modo satisfactorio (Newton decía que los corpúsculos deberían tener “ataques de reflexión” y “ataques de refracción”, pasando periódicamente de uno a otro, pero sin que quedase determinado el porqué de estos “ataques” ni las leyes

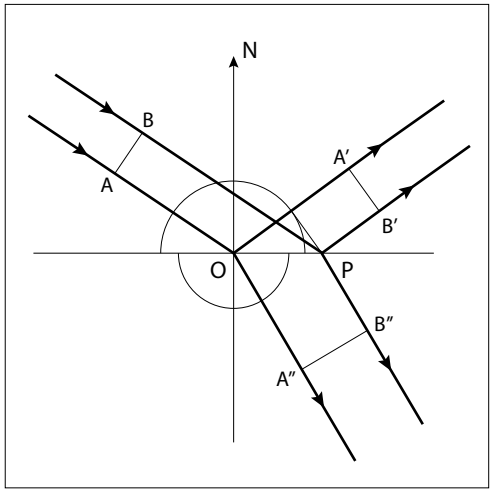


Figura 1.30 Reflexión y refracción según los principios de Huygens.

Material	*n*
Vacío	1.000
Aire (1,0003)	~1.000
Hielo	1.310
Agua	1.333
Leche	1.350
Alcohol etílico	1.360
Aceite (entre 1,46 y 1,52)	1.500
Vidrio impuro	1.450
Vidrio corriente	1.500
Cuarzo	1.550
Esmeralda	1.570
Vidrio flint (*)	1.700
Zafiro, rubí	1.770
Diamante	2.400

* (vidrio con alto contenido en plomo; según este contenido sea bajo o alto el índice puede variar entre 1,60 y 1,80)

Figura 1.31 Índices de refracción de materiales corrientes.

que explicaban la refracción). De hecho, esta fue una de las causas por las que la teoría ondulatoria se impuso a la corpuscular a partir de 1825 (con los trabajos de Young, Fresnel, Argo y Malus principalmente) hasta que esta última resurgió a principios del siglo XX.

La figura 1.30 muestra un frente de ondas AB que cae sobre la superficie que separa dos medios de diferente densidad. Cuando la onda alcance el punto O de la superficie se convierte en una fuente de ondas secundarias que se difunde tanto hacia el medio más denso, inferior, como hacia el superior. Si se asigna un tiempo *t* al tiempo que tarda el otro extremo del frente de ondas en llegar al punto P, durante este tiempo se habrá formado una onda secundaria de radio *v*1*t*, siendo *v*1 la velocidad de la luz en el primer medio y una onda secundaria de radio *v*2*t*, siendo *v*2 la velocidad de la luz en el segundo medio. Puede demostrarse con facilidad, analizando las relaciones entre estas magnitudes y los ángulos que forman los rayos de luz con la normal a la superficie, que el resultado al que se llega es el mismo que antes, es decir: sen θ_i /sen $\theta_r = n_1 / n_2$. No incluyo esta demostración pero puede encontrarse en un buen libro de óptica junto con más detalles de este análisis (véase por ejemplo el de Ditchburn, 1961, apartado 3.13).

Sin embargo, no hay ningún material que cumpla estas leyes ideales de la reflexión y refracción. Los apartados que siguen introducen explicaciones más complejas que, como veremos, están en la base de los principales algoritmos de simulación de materiales.

Variaciones del color con el ángulo de visión. Ecuaciones de Fresnel

Como ya he avanzado, uno de los factores que intervienen en el cálculo de los coeficientes de reflexión y refracción es el ángulo de visión.

Que la reflexión se modifica por el ángulo es algo que ya fue observado por Galileo y que, junto con otras reflexiones que merecen ser releidas, se encuentran en *Il Saggiatore*. Allí, Galileo escribía:

"[…]. En cuanto a la necesidad de pulidez, afirmo que aún sin ella se producirá la

reflexión de la imagen, unida y distinta: digo esto porque la fragmentada y confusa se produce en todas las superficies, por escabrosas y desiguales que se quieran; la imagen de un paño coloreado se ve muy clara en el espejo colocado en frente, pero muy confusa y fagmentada en el muro, en el que solamente se ve cierto ensombrecimiento. Pero si Vuestra Ilustrísima coge una piedra, o mejor un listón de madera, no tan liso que nos devuelva directamente las imágenes, y lo expone oblicuamente al ojo, como si quisiese saber si está recto y llano, verá sobre él la imagen de los objetos que se encuentran al otro extremo del listón, tan distinta que si se tratara de un libro podría leerlo cómodamente [...]" (p. 169 de la traducción de J.M. Revuelta de 1984, véase las referencias).

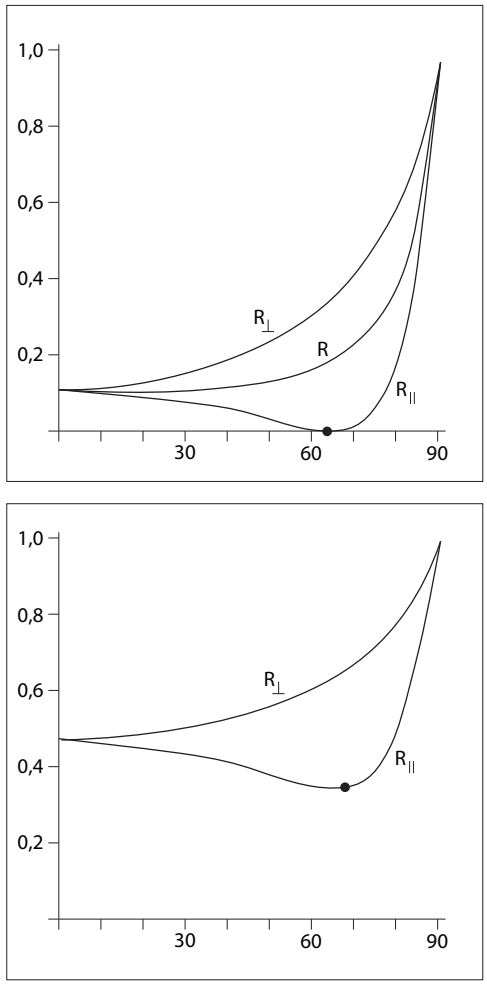


Figura 1.32 Coeficientes de reflexión para un material dieléctrico (arriba) y metálico (abajo). El ángulo señalado con un punto es el ángulo de Brewster.

Agustin-Jean Fresnel (1788-1827), que era hijo de un arquitecto, estudió ingeniería en la École des Ponts et Chaussées y aunque trabajó inicialmente como ingeniero, es famoso por sus contribuciones a la óptica, debido a las cuales fue elegido miembro de la Académie des Sciences de París en 1823 y de la Royal Society de Londres en 1825. Fresnel contribuyó a la consolidación de la teoría ondulatoria, propuesta inicialmente por Huyghens en el siglo XVII, eclipsada después por la de Newton y revitalizada por Thomas Young (con quien mantenía correspondencia) pocos años antes.

Hacia 1821, Fresnel dio a conocer un trabajo en el que se incluían lo que, desde entonces, se conoce como ecuaciones de Fresnel o fórmulas de Fresnel. Las ecuaciones de Fresnel son un conjunto de ecuaciones que describen de un modo completo y detallado las ondas reflejadas y refractadas en función de la onda incidente. Fresnel demostró que, además de (*a*) las constantes propias de los medios que afectan al índice de refracción, los coeficientes de reflexión y refracción dependían de (*b*) la frecuencia de la onda incidente, (*c*) los ángulos de incidencia y transmisión y (*d*) la polarización de la luz.

Las ecuaciones de Fresnel son un conjunto de relaciones matemáticas complejas que están fuera de lugar en un libro de estas características. El lector interesado las puede encontrar en un buen libro de óptica o en internet. Por otra parte, si se prescinde de la polarización, que es poco relevante para las aplicaciones corrientes, estas fórmulas se simplifican. En cualquier caso, puede bastar

con recordar que se trata de una función de cuatro variables:

$$R = f(\theta_i, \theta_t, n_1, n_2)$$

en donde (θ_i es el ángulo de incidencia, θ_t es el ángulo de transmisión, n_1 es el índice de refracción del primer medio y n_2 el índice de refracción del segundo medio.

La figura 1.32 muestra estos coeficientes para un material dieléctrico (aislante) y conductor (metal). Como puede apreciarse, la reflexión aumenta notablemente al aumentar el ángulo de incidencia. Y varía con los componentes de polarización, según estos sean paralelos o perpendiculares al campo eléctrico, lo que da valores extremos a los valores medios que se indican para el caso del material metálico.

El ángulo señalado con un punto en dicha figura es el ángulo de Brewster que, como puede verse, hace que la reflectancia se reduzca a 0 en el caso de un dieléctrico para la polarización paralela y a un valor mínimo en el caso de un metal.

En general, para cualquier superficie, tal como se refleja en dicha figura, la cantidad de luz que se refleja será mayor si el ángulo de incidencia es grande (si el rayo es rasante a la superficie), menor si es pequeño (si el rayo se acerca a la perpendicular a la superficie) y las cantidades relativas de reflexión y refracción variarán entre estos dos extremos.

Además de con el ángulo, el factor Fresnel está relacionado con el índice de refracción. Por otro lado, está también relacionado con la longitud de onda. Esto quiere decir que no solo variará la intensidad del reflejo sino también su color, algo que puede apreciarse con claridad en los reflejos metálicos.

Variaciones de reflexión debidas a la anisotropía

La anisotropía es lo opuesto de la isotropía (del griego *iso*, "igual" y *trópos*, "dirección"). Una determinada propiedad de un material es isotrópica si es igual para cualquier dirección que se considere. Y es anisotrópica si cambia según la dirección que se considere.

La anisotropía solo se encuentra en el estado sólido. Ni los gases ni los líquidos presentan, en general, propiedades que dependan de la dirección. El grado de anisotropía de un sólido depende, por otro lado, principalmente de su grupo cristalino. En los cristales cúbicos las propiedades deben ser las mismas en las tres direcciones. En otros casos ocurre lo contrario.

Desde el punto de vista de las características visuales, esta propiedad es importante pues el aspecto de muchos materiales cambia con la dirección, debido a la estructura propia del material. Esto es característico sobre todo de los tejidos, en los que el modo de fabricación, la particular estructura de la trama y la urdimbre afecta al modo en que reflejan la luz en una u otra dirección. Y es más notorio en el caso del satén, un tipo de tejido más compacto y en el que la urdimbre (las fibras longitudinales), de seda, predomina sobre la trama (las fibras transversales) que pueden ser, o no, también de seda. Debido a esto, reflejan la luz de un modo más continuo a lo largo de la dirección de la urdimbre, predominante, lo que da al satén un brillo direccional característico. Otro tanto cabe decir del pelo o del terciopelo, cuyo brillo cambia con la dirección de observación o la dirección de incidencia de la luz.

Pero una superficie también puede ser anisotrópica debido al tratamiento superficial que haya recibido. Así, una superficie de madera puede aparecer diferente si está barnizada con un pincel que deje diminutos surcos en una dirección o bien pulida con lana de acero formando círculos superpuestos. Y una superficie metálica también resultará diferente si está tratada con un cepillado metálico o acabada con otros medios.

Todo esto afecta a la reflexión pero también a la refracción pues en muchos materiales el índice de refracción es distinto para las diferentes direcciones del espacio.

Dispersión refractiva

La dispersión refractiva es un fenómeno que altera los colores, debido a que el paso de la luz por un medio distinto del vacío reordena las diferentes longitudes de onda presentes en un único rayo de luz y hace que percibamos estas diferentes longitudes de onda como colores distintos. Como es bien sabido, a partir de los experimentos llevados a cabo por Newton en 1666, la luz es una mezcla heterógenea de diferentes longitudes de onda que pueden separarse por medio de un prisma de vidrio o, en general, al pasar de un medio a otro (el vidrio del prisma modifica la mayor o menor desviación de las longitudes de onda al pasar del aire al cristal y viceversa). Este fenómeno es la causa de muchos colores de la naturaleza, desde el arco iris o los halos que vemos en torno al Sol o la Luna hasta los bordes coloreados que se aprecian en joyas, vidrios o sistemas ópticos debido a la aberración cromática.

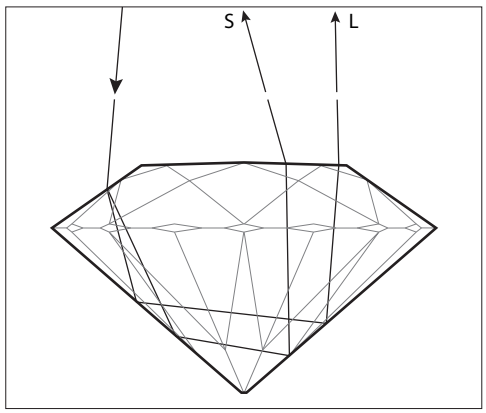


Figura 1.33 Dispersión refractiva. Gemas.

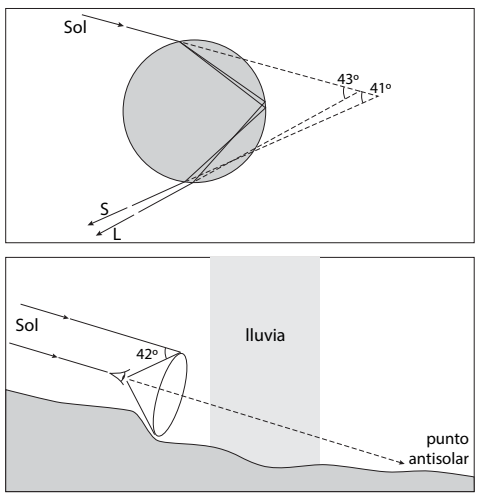


Figura 1.34 Dispersión refractiva. Arco iris.

Para que se dé este fenómeno es necesario que la luz pase por un medio transparente con un índice de refracción mayor que el aire, y que el ángulo de las caras de entrada y salida del objeto formen un cierto ángulo. Como esto último suele ser poco habitual, el fenómeno se puede ver en pocos casos. Puede observarse en las gemas, las piedras preciosas talladas, pues el ángulo de las facetas hace que los rayos de luz se refracten como en un prisma. El fenómeno es similar al del arco iris en donde las gotas de lluvia se dispersan tal como se indica en la figura 1.34. Es un fenómeno bastante complejo pero lo fundamental es que los rayos de Sol se dispersan al pasar por gotas de agua que flotan en la atmósfera. La desviación de las ondas cortas (violetas) es de unos 41º y las de las ondas largas (rojas) de unos 43º. La luz se refleja de vuelta hacia el Sol. Pero si estamos situados frente al Sol, y el Sol está por debajo de los 42º, lo que veremos es la dispersión de estos rayos que pasan por todas las longitudes de onda principales a las que el ojo es sensible (básicamente 6: rojo, naranja, amarillo, verde, azul y violeta, no 7 como quería Newton y se sigue explicando en algunos libros de bachillerato, si bien es cierto que hay un degradado continuo que puede subdividirse de múltiples maneras).

Polarización

La polarización de la luz se observó, antes de que pudiera explicarse en los términos actuales, a partir de materiales que exhiben

doble refracción, como el espato de Islandia. La cristales que pertenecen al sistema cúbico son ópticamente isotrópicos, tienen un único índice de refracción. Pero hay cristales ópticamente anisotrópicos que tienen más de un índice de refracción. En estos casos, los fenómenos de refracción son muy complejos y varían con la dirección de la luz y la orientación del cristal.

Si un rayo de luz atraviesa en una determinada dirección un cristal de espato de Islancia (una variante del mineral calcita) emergen dos rayos de luz diferentes. Al primero se le denomina rayo ordinario y sigue las leyes corrientes de la refracción; al segundo, rayo extraordinario y tiene un índice de refracción variable. Si se mueve el cristal, uno de los dos rayos permanece fijo mientras el otro rota en torno al anterior. La explicación de este fenómeno, en términos actuales, es que la luz se polariza y solo es visible una determinada dirección de vibración en cada caso. Hay otros fenómenos naturales en los que se puede apreciar este mismo fenómeno pero, en general, se requieren instrumentos ópticos especiales para observar la polarización de la luz. Algunos objetos corrientes, como el papel de celofán, también exhiben este mismo fenómeno. En general, los objetos brillantes y transparentes pueden exhibir este fenómeno en determinados casos.

A partir de los trabajos de Maxwell sabemos que la luz está formada por ondas electromagnéticas que vibran en dos direcciones: en la dirección de transmisión de la luz y en una plano perpendicular a esta dirección. Las ondas que vibran perpendicularmente a la dirección de propagación lo hacen en todas direcciones. La polarización hace que las direcciones de vibración queden restringidas de diferentes modos (que corresponden a diferentes tipos de polarización: lineal, circular o elíptico). Esto puede producirse por varias causas. Algunos materiales, como ciertos cristales minerales o los cristales líquidos, absorben selectivamente alguna de las componentes transversales, lo que se denomina dicroísmo.

Cuando la luz incide sobre una superficie no absorbente con cierto ángulo, la componente del plano transversal paralela al plano de incidencia no se refleja. Este ángulo, que ya se ha citado antes, conocido como ángulo de Brewster por el físico británico David Brewster (1781-1868), que hizo contribuciones notables al estudio de la polarización, se alcanza cuando el rayo reflejado es perpendicular al refractado.

Dispersión (*scattering*)

Otro fenómeno no menos importante es la dispersión (*scattering*) que es particularmente notorio en el caso de medios dispersivos tales como el cielo azul del amanecer, el cielo rojizo del atardecer, el color de las nubes, el del aire polucionado, la niebla, el humo o gases de diversos tipos.

Las primeras explicaciones correctas del fenómeno se deben al inglés John Tyndall (1820-1893), que demostró en su laboratorio que se debía a la presencia de partículas de un tamaño comparable al de la longitud de onda visible y que las longitudes de onda corta (azules) se dispersaban mucho más que las largas (rojas). Posteriormente, Lord Rayleigh (1842-1919) demostró que el fenómeno se da en todo tipo de materiales, en pequeña medida, debido a las fluctuaciones en el índice de refracción. Rayleigh demostró también que la intensidad de la luz dispersada, I_s, se relaciona con la intensidad de la luz incidente I_i por la siguiente relación

$$I_s / I_i = k / \lambda^4$$

donde k es una constante y λ es la longitud de onda.

Cuando las partículas son del orden de los 300 nm, algo más pequeñas que la longitud de onda correspondiente a los violetas, la dispersión azulada es notoria. En el caso del cielo, la dispersión es más intensa al amanecer y al atardecer debido a que los rayos del Sol, que inciden oblicuamente sobre la atmósfera, tienen que atravesar una porción de atmósfera mucho más larga. También el tono

es más rojizo, pues durante el día los rayos dispersados, vistos contra el fondo oscuro del espacio exterior, aparecen de color azul por estar compuestos principalmente de ondas cortas. Al amanecer o al atardecer este efecto queda atenuado y vemos principalmente ondas largas. El efecto es aún más intenso al atardecer debido a que durante el día se acumulan partículas de polvo y de todo tipo en la atmósfera.

Los tonos azulados debidos a la dispersión se dan con frecuencia en los animales. De hecho, prácticamente todos los tonos azulados, exceptuando la iridiscencia que se explica más adelante, son debidos a este fenómeno que es particularmente notorio en las plumas de algunos pájaros y que es causado por la dispersión provocada por una fibras transversales que refuerzan las alas. También los ojos de los recién nacidos se ven de color azul por las mismas causas, debido a que aún no se han formado la melanina, un pigmento parduzco, y el fondo oscuro del ojo se ve a través del iris.

También está presente el mismo fenómeno en las coloraciones azuladas de la piel blanca de los europeos, que se refuerza a veces por el color oscuro de los pelos afeitados. Por la misma razón las venas se ven azuladas pues proporcionan un fondo oscuro para la dispersión superficial.

Si el humo tiene pequeñas partículas húmedas (como ocurre cuando se fuma y se traga el humo antes de volver a expulsarlo) aparece, por la misma razón, más azulado.

Sin embargo, si el tamaño de las partículas se hace mayor que la longitud de onda y las partículas son de tamaño similar, se produce otra forma de dispersión conocida como dispersión de Mie, por el físico alemán Gustav Mie (1869-1957) que propuso la teoría que lleva su nombre en 1908. La dispersión se hace más intensa y se produce principalmente en la dirección de propagación. El color también cambia y puede ser de muy diversos tipos, generalmente en bandas de tonos rojizos y verdosos.

Interferencia. Iridiscencia. Difracción

Las ondas, en general, pueden anularse o reforzarse, como podemos ver en un estanque si tiramos una piedra y luego otra. Otro tanto ocurre con las ondas lumínicas que pueden anularse si están desfasadas o reforzarse si están en fase. Si la luz cae sobre una película transparente delgada, sea una lámina de plástico o un charco de aceite o de agua, la luz se refleja desde su capa superior y desde su capa inferior. Según el espesor de la capa se producirá uno u otro efecto. Y si la capa tiene un espesor variable, aparecerán bandas de diferentes colores, franjas de interferencia, un fenómeno ya observado por Newton. En las zonas de menor espesor aparecerán franjas oscuras de color neutro al cancelarse las longitudes de onda. Al ir aumentando el espesor aparecerán franjas de color blanco, luego amarillo, naranja, rojo, violeta, azul, verde, de nuevo amarillo, naranja, etc.

La figura 1.35 muestra lo que ocurre cuando la luz incide con un cierto ángulo sobre una lámina delgada de espesor *t*. Parte de la luz se refleja en la primera superficie (A) y parte se refracta para volver a reflejarse en la segunda superficie (B). La luz atraviesa de nuevo el espesor de la lámina para volver a salir refractándose (C). Este rayo saliente en C interfiere con el rayo saliente en A. Los patrones de interferencia dependerán del ángulo de incidencia, del espesor y de los índi-

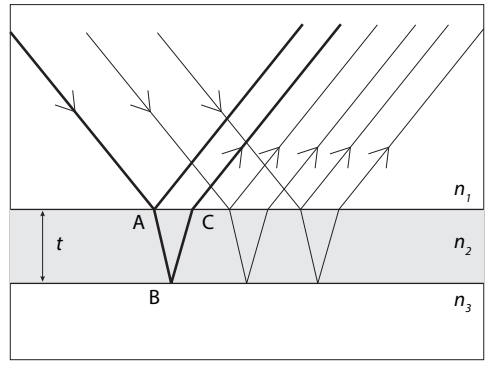


Figura 1.35 Patrones de interferencia en una lámina delgada.

ces de refracción del medio implicado. Para no complicar la explicación, cuando el ángulo es cercano a la perpendicular tendremos que la distancia adicional que recorre la luz (ABC) será aproximadamente igual al doble del espesor *t*, es decir 2*t*. La longitud de onda de la película variará en proporción inversa al índice de refracción, es decir, $\lambda' = \lambda/n$. En estas condiciones, habrá unos máximos en el patrón de interferencia que vendrán dados por la igualdad $2t = m\,\lambda'$ y unos mínimos que vendrán dados por $2t = (m + 1/2)\,\lambda'$ con $m = 0, 1, 2, 3$, etc.

Este fenómeno se puede ver en grietas de cristales, en burbujas de jabón, en aceite mezclado con agua, en las alas semitransparentes de algunas moscas o en minerales transparentes. También puede crearse, en la actualidad, por medios artificiales, utilizando filtros de interferencia que consisten en láminas pulidas con una precisión inferior a la de las ondas de luz y que se utilizan para aplicaciones especiales, tales como la observación astronómica o microscópica. También se utilizan en aplicaciones muy específicas, como la fabricación de tintas especiales empleadas en el papel moneda y que hacen que el color se modifique con el ángulo de observación.

Pero también está en la base de los fenómenos de iridiscencia. La iridiscencia es un fenómeno óptico por el que algunas superficies cambian de color según el ángulo desde el que se las observa. Esto ocurre, por ejemplo, con las manchas de aceite, las burbujas de jabón y las alas de mariposas y de algunos insectos. También puede apreciarse, en menor medida, en las uñas, el pelo y los ojos de los animales. El fenómeno básico es el mismo: la luz de refleja en múltiples capas superficiales semitransparentes y los subsecuentes cambios de fase y las interferencias que se crean en estas reflexiones modulan de diferente modo la luz incidente, variando según las longitudes de onda y los ángulos de observación.

La difracción es una variante de la interferencia que se produce cuando las ondas encuentran un obstáculo. Si se coloca un objeto entre una fuente de luz y un plano, por ejemplo un muro blanco, el objeto arrojará sombra sobre el plano. Pero si se observa la sombra de cerca se apreciarán fanjas oscuras y franjas claras alternadas. Esto quiere decir que, de algún modo, los rayos de luz se han curvado para penetrar en las zonas de sombra donde, según las leyes de la óptica geométrica, nunca podrían llegar.

Otro tanto ocurriría, pero al revés, si hacemos pasar la luz por una rendija hasta que ilumine un plano paralelo a la rendija. Teóricamente, la zona central proyectada sobre el plano, correspondiente a la rendija, aparecería iluminada y más allá de los bordes se vería una sombra. Pero en los bordes aparecerán de nuevo franjas claras y oscuras.

El fenómeno se explica porque la luz, al interaccionar con objetos de dimensiones comparables a su longitud de onda, deja de propagarse de modo rectílineo. El frente de ondas se modifica y cada punto se convierte en emisor de nuevas ondas que interfieren entre sí. El resultado es que algunas se potencian (refuerzo constructivo) y dan lugar a franjas más luminosas y otras se anulan (cancelación destructiva), dando lugar a franjas más oscuras.

La difracción fue descrita por primera vez por Grimaldi (1618-1663) y llegó a ser una de las pruebas decisivas para sostener la teoría ondulatoria de la luz. La difracción de Fresnel (descrita por Fresnel hacia 1818), también denominada difracción de campo cercano para diferenciarla de la de Fraunhofer, de campo lejano, se produce cerca del objeto causante de la refracción. La difracción de Fraunhofer (descrita por Fraunhofer hacia 1814) es similar pero la incidencia de los rayos es aproximadamente paralela, con lo que se forman ondas planas.

Una red de difracción consiste en una serie de aberturas (u obstáculos) que dispersan la luz produciendo fenómenos de difracción complejos que dan lugar a franjas luminosas variables. Este tipo de redes se observan en

la naturaleza, en las alas de algunos escarabajos o las pieles de algunas serpientes, o en algunas piedras preciosas como el ópalo. Fotografías con microscopio electrónico del ópalo revelan que su apariencia es debida a una red de difracción tridimensional formada por esferas de unos 250 nm de diámetro. También pueden crearse artificialmente redes de difracción haciendo hendiduras regulares en una superficie plana, que puede ser un espejo, para formar redes de reflexión, o un vidrio, para formar redes de transmisión.

1.5 Variaciones locales

Texturas

La palabra "textura" viene del mundo textil y designaba originalmente las diferentes maneras de disposición de los hilos de un tejido. Pero el término se utiliza de modo más general para designar las variaciones de color de una superficie por analogía con los tejidos.

Estas variaciones pueden ser debidas a dos causas principales. Si la superficie es lisa pero su color local cambia, por variaciones en la estructura o en la composición del material, que se traducen en diferencias de absorción como ocurre con la madera o con el mármol. En estos casos, la superfice tiene una textura característica aunque no haya variación geométrica. Si la superficie es cromáticamente uniforme pero su geometría local varía, porque tiene hendiduras o salientes de diferente tipo, como ocurre con superficies pintadas de un modo tosco, con la piel de naranja o con piedras de color muy uniforme pero a las que se ha aplicado un acabado abujardado, la superficie tiene una textura característicamente rugosa aunque no haya variaciones cromáticas notables.

Patrones causados por diferencias de color o por diferencias de microrelieve

En una primera aproximación podemos distinguir, por tanto, entre texturas planas y texturas rugosas, una distinción que tendrá relevancia inmediata desde el punto de vista de las técnicas de simulación.

Una textura es *plana* cuando no hay variaciones apreciables en su microestructura geométrica. Una manera más precisa de decir lo mismo es que las variaciones en la

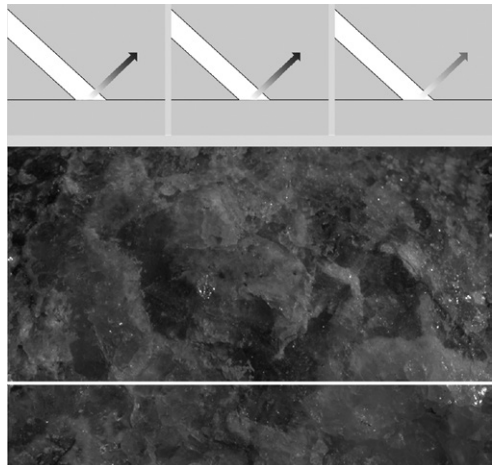


Figura 1.36 Textura plana. A lo largo de la línea blanca (abajo) se producen diferencias de absorción (arriba) que forman patrones característicos.

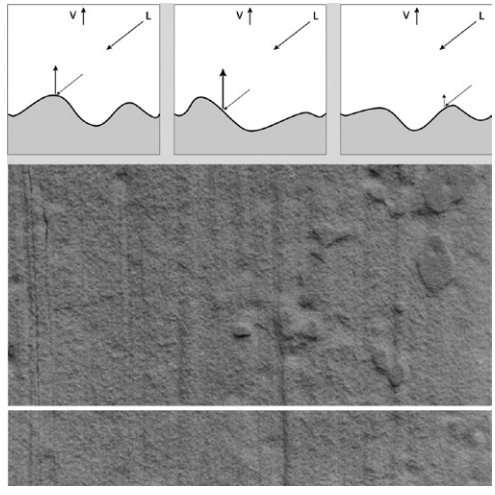


Figura 1.37 Textura rugosa. A lo largo de la línea blanca (abajo) se producen diferencias de orientación geométrica que dan lugar a variaciones en la iluminación local.

orientación del vector normal a la superficie estarían por debajo de un cierto umbral que no es necesario precisar. O bien, lo que es lo mismo, pues la reflexión especular se computa con respecto a la normal, que las variaciones en el ángulo de reflexión también están por debajo de un determinado umbral.

Como consecuencia de esto la textura no presenta variaciones en el reflejo y tampoco presenta microsombras que modificarían la percepción del color base aunque la luz incidente sea muy rasante.

¿Qué diferencia hay entre una textura plana y una superficie cubierta de patrones, como podría ser una superficie pintada? Para una descripción como la que estamos siguiendo, ninguna. La única diferencia es que, en la mayoría de los casos, las texturas planas se habrán generado mediante tratamientos, por lo general artificiales (cortes, operaciones de pulimentado) de materiales naturales, mientras que los patrones se habrán generado mediante tratamientos artificiales adicionales superpuestos a los anteriores.

Esta distinción es importante pues, como veremos más adelante, la simulación de texturas no se utiliza solamente para simular materiales sino también para incorporar imágenes a superficies y para generar lo que en el argot profesional se denomina "geometría falsa", patrones que parecen corresponder a diferencias de materiales o de elementos constructivos pero que no son sino patrones adecuadamente dispuestos sobre una superficie plana.

Una textura es *rugosa* cuando hay variaciones apreciables en su microestructura geométrica. Una manera más precisa de decir lo mismo es que las variaciones del vector normal a la superficie son superiores a un cierto umbral que tampoco es necesario precisar. Como veremos más adelante, una manera muy sencilla de simular una textura rugosa es, precisamente, alterar artificialmente la normal, aunque la geometría de la superficie base siga siendo plana.

Como en el párrafo anterior, hay que añadir que esto implica que la orientación del vector correspondiente al rayo reflejado sufirá alteraciones. Y las consecuencias serán que la superficie mostrará diferencias de intensidad correspondientes a "cumbres" y "valles". Y por añadidura, si la luz es suficientemente rasante o la superficie suficientemente rugosa o ambas cosas, aparecerán sombras propias y arrojadas que modificarán el color percibido que será más oscuro en determinadas zonas y aparecerán también micropatrones debidos a las sombras.

¿Qué diferencia hay entre una "textura rugosa" y una superficie "con relieves"? Para una descripción como la que seguimos, ninguna. La diferencia principal es también de origen: en un caso los relieves se habrán producido de modo natural y en otros casos de un modo artificial. Sin embargo, aquí hay una ambigüedad más evidente en la que tampoco necesitamos entretenernos: cuando los relieves son muy prominentes ya no hablamos de textura sino de una superficie "con relieves", al igual que cuando vemos una superficie esculpida tampoco hablamos de textura. Pero desde un punto de vista que se limite a describir diferencias de color y diferencias geométricas no hay diferencia.

También esta distinción es importante pues está estrechamente relacionada no solo con la modelización por medio de "geometría falsa", como en el caso anterior, sino con problemas técnicos de multirresolución que veremos más adelante.

Patrones generales. La percepción de texturas

La percepción de texturas se considera que está relacionada directamente con la percepción táctil, y en ella intervienen tanto el sistema visual como lo que se denomina sistema somatosensorial, un sistema complejo de percepción en el que intervienen diversas modalidades de estímulos y receptores. Por otro lado, en las investigaciones somatosensoriales, el espacio táctil se considera a veces dividido en tres polaridades: duro-blando (*hard-soft*), tosco-suave, (*rough-smooth*), ad-

herente-deslizante (*sticky-slippery*). Y estas claves no son exclusivamente espaciales sino temporales: distinguimos estas cualidades al desplazar los dedos sobre una superficie. Todo esto proporciona un marco de referencia complejo en el que claves exclusivamente visuales se asocian con claves de reconocimiento táctil. Esta distinción está claramente relacionada con la distinción previa que he establecido entre texturas planas y rugosas. Pero, al mismo tiempo, revela la insuficiencia de esta distinción. Por un lado, los patrones debidos a diferencias de absorción, como los de un mármol pulido, no serían perceptibles por el tacto. Y, por otro lado, los patrones debidos a diferencias de geometría local pueden ser confundidos con patrones debidos a diferencias de absorción.

Resulta, por tanto, evidente que los patrones visuales por si mismos aportan claves que parecen independientes. El que esta independencia sea el resultado de un largo proceso de evolución o de aprendizaje o de ambas cosas, y que la conexión entre el tacto y la vista siga estando en el origen de nuestra capacidad de reconocimiento de texturas es una cuestión abierta.

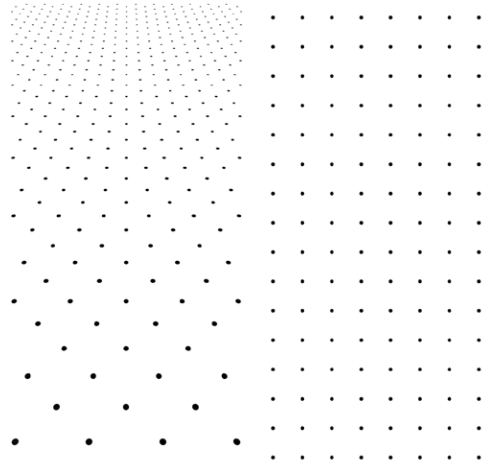


Figura 1.38 Distribuciones de puntos que dan la impresión de una superficie en perspectiva y de una superficie frontal. Adaptado de Gibson (1950).

Desde el punto de vista general de la investigación sobre la percepción visual de texturas, hay varios investigadores que han aportado análisis notables. Mencionaré solamente tres: Gibson, Marr y Julesz. La investigación actual está relacionada principalmente con los trabajos de los dos últimos pero se

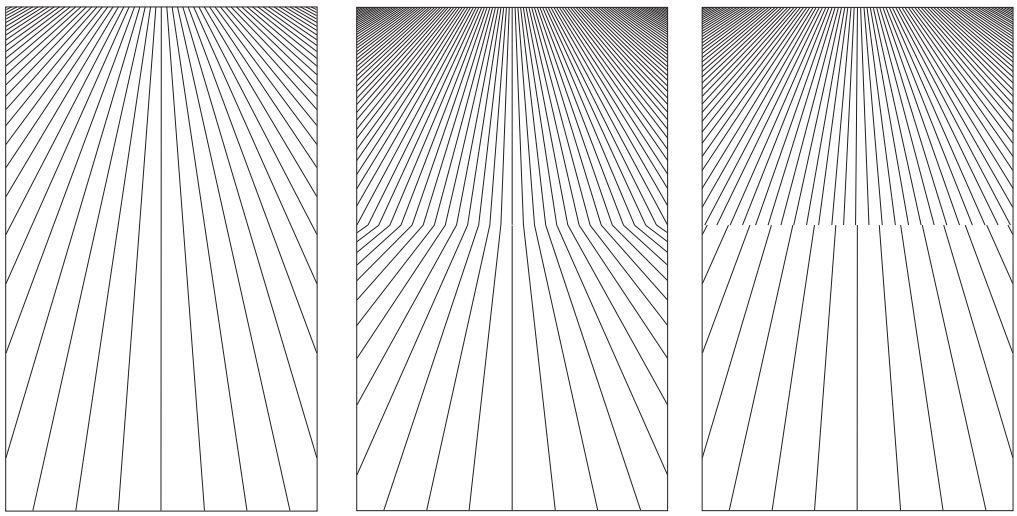


Figura 1.39 Cambios de gradiente que a) *se perciben como una superficie en perspectiva,* b) *se perciben como una esquina,* c) *se perciben como un borde en una superficie continua. Adaptado de Gibson (1950).*

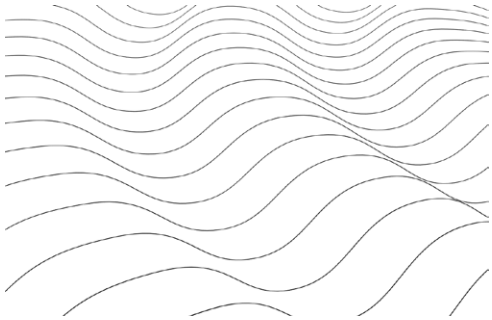


Figura 1.40 Ondulaciones regulares que se perciben como una superficie tridimensional con orientaciones definidas.

debe al primero el mérito de haber señalado por primera vez la importancia de las claves derivadas de la percepción de texturas en nuestra percepción visual.

James Gibson (Gibson, 1950) destacó el papel que juegan los gradientes de texturas en nuestra percepción del mundo visual a partir, entre otras cosas, de datos aportados por los aviadores, para los que las texturas terrestres proporcionaban claves vitales durante el aterrizaje pero que, como se apreció más tarde, eran datos universales sobre los que no se habia reflexionado hasta la fecha. La figura 1.38, adaptada de otras similares de la obra de Gibson, muestran de un modo sintético, cómo los gradientes de textura tienden a ser percibidos como superficies que se mantienen paralelas a la vista o se pierden en la distancia. En general, Gibson destacó que el reconocimiento de texturas incluye varios aspectos funcionales: *a*) nos permite identificar superficies conocidas, *b*) nos permite identificar bordes de transición en superficies, *c*) nos proporciona claves de profundidad por medio de gradientes de textura. La figura 1.39, igualmente adaptada de otras similares de Gibosn, ilustra estos aspectos.

David Marr (Marr, 1982), en su gran obra póstuma también destacó la importancia de las claves de reconocimiento de las superficies, entre las que figuran en un lugar destacado los diferentes tipos de texturas. En concreto, determinadas características de texturas se perciben corrientemente como claves visuales que nos permiten reconocer la propia orientación de la superficie. Una imagen como la de la figura 1.40, que representa una serie de ondulaciones bidimensionales, se percibe corrientemente como una superficie tridimensional, con ondulaciones superficiales. Es decir, nuestro sistema visual interpreta estas líneas como señales que actúan como generadores de contorno (Marr, p. 221 de la versión española de 1985). El análisis de cómo se lleva a cabo exactamente esta interpretación en nuestro sistema visual lleva a Marr a una discusión importante de las hipótesis de Gibson y de los trabajos de otro

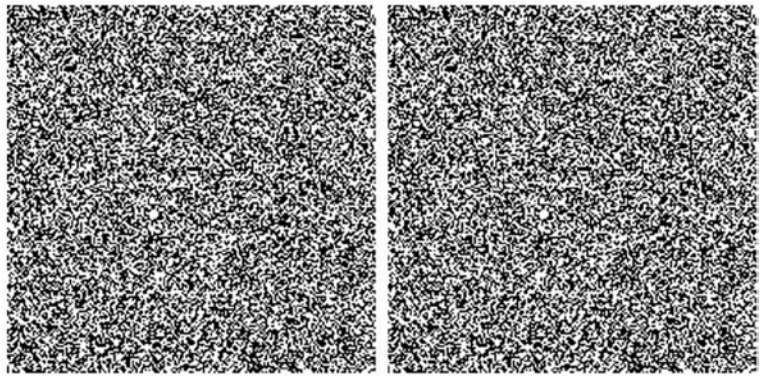


Figura 1.41 Estereograma de puntos aleatorios. Si se desenfoca la vista hasta que los dos cuadrados se fundan en uno se percibirá (aunque puede ser dificultoso) un pequeño cuadrado flotando sobre el mayor. Pese a que no hay ningún rasgo reconocible, el sistema visual humano busca similitudes que den claves sobre el reconocimiento de formas.

investigador notable, K. Stevens, que no cabe en este apretado resumen. En cualquier caso, es intuitivamente evidente que los contornos internos de las texturas, en determinadas circunstancias, son interpretados como claves tridimensionales de valor funcional evidente.

Y, en fin, Béla Julesz (1928-2003) es conocido principalmente por ser el creador de los estereogramas de puntos en 1959, utilizando patrones aleatorios de puntos sobre pares de imágenes con desplazamientos tales que, al visualizarse en una única imagen, generaban la ilusión de profundidad (posteriormente este método evolucionó para dar lugar a los autoestereogramas que utilizaban una única imagen y se pusieron muy de moda hacia el 2000). Pero fue también el primero en investigar de un modo sistemático cómo percibimos y discriminamos los patrones texturales.

Un estereograma como el que se muestra en la figura 1.41 se puede construir con facilidad. Hay que llenar una imagen (por ejemplo un cuadrado) con puntos aleatorios. Luego hacer dos copias y desplazar un pequeño cuadrado central hacia la izquierda en la imagen de la izquierda y hacia la derecha en la de la derecha. Los huecos que así quedan se rellenan con otros puntos aleatorios. Este desplazamiento es equivalente al que percibiríamos con objetos tridimensionales. Los puntos adicionales corresponden a puntos que un ojo vería pero el otro no. El resultado, que no acaba de contar con una explicación precisa pero probablemente se basa en la identificación de similitudes, es que determinadas zonas de la imagen se perciben como situadas sobre un fondo distinto.

Joulesz propuso en 1981 el concepto de *texton* como "la unidad putativa de percepción humana pre-atenta de las texturas" ("the putative units of pre-attentive human texture perception"), un concepto que se ha utilizado posteriormente en la investigación sobre la percepción de patrones por máquinas. Y sugirió que esto se relaciona con rasgos formales tales como aristas, finales de línea o abultamientos puntuales (*blobs*). Consideraba que la discriminación de texturas podía ser modulada por densidades de primer orden de tales *textons*. Esta observación está relacionado con el uso de procedimentos sistemáticos de reconocimiento de patrones, una línea de investigación que ha adquirido creciente importancia en los últimos años.

A diferencia de lo que ocurre con la identificación de formas, en el caso de texturas, la modificación (traslación, rotación, etc.) de partes de la textura no altera de modo determinante nuestra percepción de la textura como tal. Esta es la razón por la que se considera que las propiedades estadísticas (la distribución de las propiedades agregadas) son determinantes en el caso de la percepción de texturas).

Una de las características que mejor se pueden precisar es la distribución espacial de los valores (niveles de gris) de las texturas. El uso de estadísticas e histogramas ha sido por tanto uno de los primeros métodos utilizados de modo sistemático en su análisis.

Los análisis estadísticos revelan si *predominan* determinadas orientaciones, o determinadas agrupaciones o determinados rasgos de uno u otro tipo. Estos análisis se llevan a cabo, por lo general, eliminando datos redundantes. Es una alternativa a otros métodos tales como DWT (*Discrete Wavelet Transform*) que descompone las señales en sus componentes armónicos básicos mediante una discretización previa. Con el método estadístico se computa, en principio, el modo en que los diferentes valores (niveles de gris) se distribuyen en los píxeles de una determinada región. Un histograma de primer orden $P(I)$ se define como $P(i) = Ni / N$, siendo N el número de píxeles de la región y Ni el número de píxeles con un nivel de gris i. De aquí se deducen valores tales como las medianas u otros datos derivados, de interés. Sin embargo, las estadísticas de primer orden no dan información sobre la posición relativa de los diferentes grises en la imagen. Un análisis más complejo busca información sobre el número de píxeles con dos valores determinados que están separados por una determinada distancia en una determinada dirección. Esto es lo que se denominan estadísticas de segundo orden.

Figura 1.42 Pares de elementos con estadísticas iguales de segundo orden. La mitad inferior de las dos figuras está compuesta por elementos diferentes de la superior. En la figura de la izquierda es necesario una inspección más atenta que en la de la derecha para diferenciar las dos regiones. De M. Tuceryan y A.K. Jain, 1998.

La identificación de agrupaciones depende de las características de los elementos predominantes. Marr consideraba que los tipos de terminación son una clave principal para segmentar uno u otro grupo. Los *textons* serían eventos visuales tales como colinearidad, tipo de terminación, tipo de cierre, etc. cuya presencia se detecta espontáneamente y se utiliza para la discriminación de texturas. La figura 1.42 muestra un ejemplo característico en el que la identificación de agrupaciones depende de las características de los elementos de la "textura".

Variaciones, clasificaciones

Tipos de variaciones. Clasificaciones

La investigación actual sobre la percepción de texturas se puede considerar dividida en cuatro grandes grupos: *a*) clasificación (asignación a un grupo previamente establecido de clases de textura), *b*) segmentación (división en regiones con propiedades homogéneas) o segmentación supervisada (división en regiones con propiedades referidas a propiedades preestablecidas), *c*) síntesis (generación de texturas por medio de modelos), *d*) forma a partir de textura (análisis de la relación entre imágenes 2D e interpretaciones 3D).

Por lo que respecta a la clasificación que tiene un mayor interés para nosotros, lo poco que hay que decir es que las dificultades para definir con suficiente rigor científico qué se entiende por textura, discurren en paralelo a las dificultades para establecer un sistema de clasificación universal. No hay ninguna definición precisa, en términos de percepción visual, que haya sido aceptada corrientemente. Se utilizan descriptores generales tales como fina o tosca (*fine or coarse*), suave o irregular (*smooth or irregular*) u homogénea o no homogénea. Los siguientes términos se han utilizado con mayor o menor frecuencia para describir texturas: uniformidad, densidad, aspereza, rugosidad, regularidad, linearidad, direccionalidad, frecuencia, fase... Pero el caso es que no hay una clasificación universal de texturas. Las clasificaciones que se han utilizado en diferentes investigaciones han sido clasificaciones *ad hoc*, adaptadas al tipo de aplicación prevista.

La mayoría de las clasificaciones usan categorías tales como estructura espacial, contraste, tosquedad (*roughness*), orientación, etc. Pero el hecho es que, en la práctica, clasificamos las texturas a partir de su asignación inmedianta a una propiedad material. Decimos que se trata de una textura de madera, piedra, tela de un cierto tipo, etc. La cuestión está precisamente en ver cómo es posible relacionar estos modos particulares y espontáneos de clasificación con modos más universales, algo que no resulta nada sencillo. La detección de similaridades de textura es una de las claves de la organización perceptual y es, sin duda, esta detección de similaridades la que hace que asignemos una determinada textura a uno u otro material.

La segregación de patrones es un aspecto que se ha investigado extensamente en los últimos años en relación con la visión por medio de máquinas. Gran parte de esta investigación se ha llevado a cabo mediante patrones estandarizados, por ejemplo patrones que contienen las letras L, T, X o patrones similares a los de los ejemplos que he mostrado anteriormente. Pero se ha investigado menos en el análisis de texturas reales, naturales. Se ha llegado a poco más que el establecimiento de grupos basados en rasgos muy generales

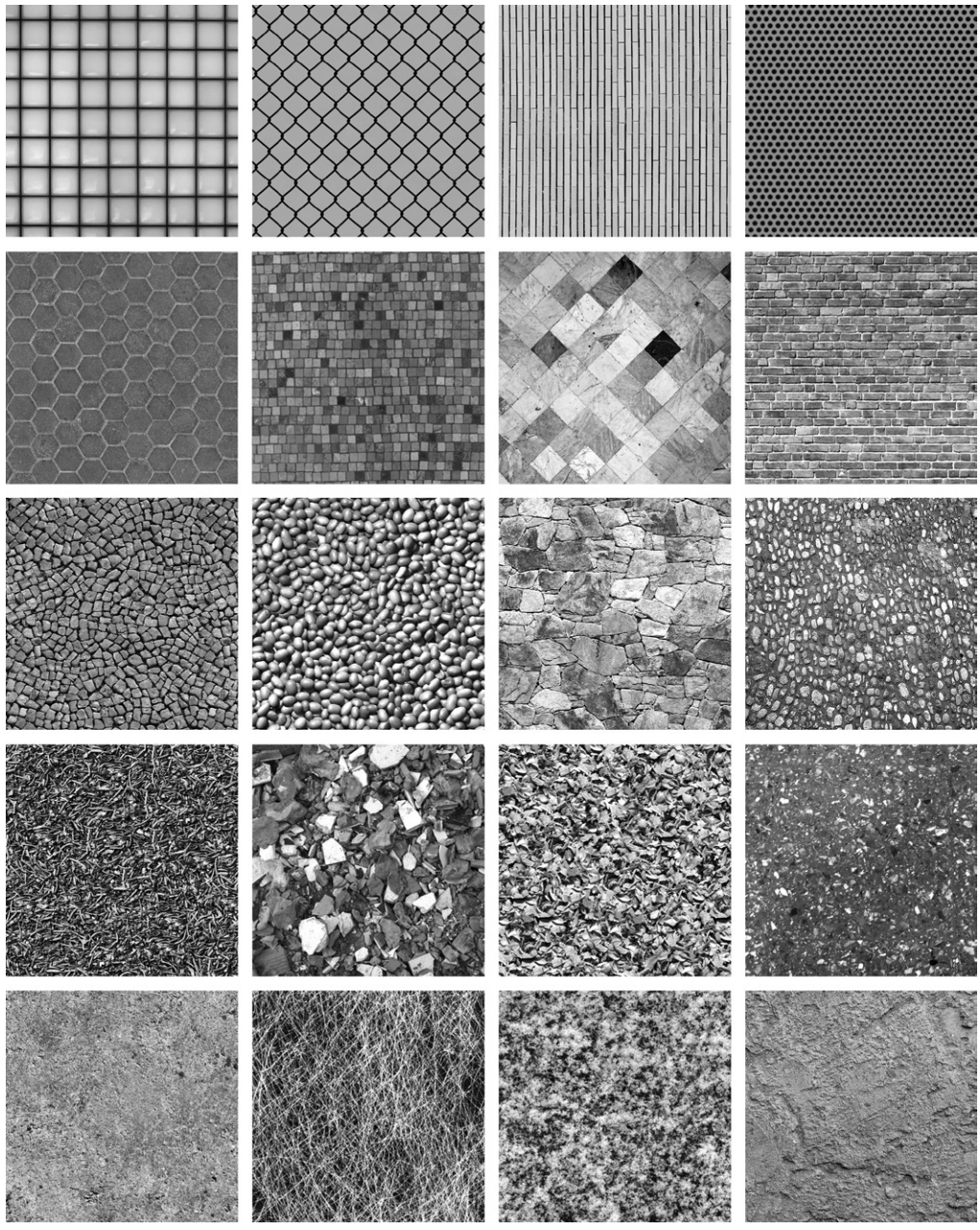


Figura 1.43 Ejemplos de texturas con periodicidad fuerte y débil (primera y segunda filas), estructuración fuerte y débil (tercera y cuarta) y no estructuración (quinta).

tales como la periodicidad, la estructuración o la falta de estructuración.

Los ejemplos que se muestran en la figura 1.43 están ordenados con arreglo a estos criterios básicos. Atendiendo a si los patrones presentan periodicidad más o menos manifiesta, a si están estructuradas a partir de elementos facilmente reconocibles, o bien si esta estructuración es muy débil o indetectable. Otras agrupaciones consideran en un grupo aparte aquellas texturas cuyo aspecto provoca sensaciones de relieve asimétricas, que cambian según la dirección de la luz, tales como las que he mostrado antes.

1.6 Variaciones debidas a otros factores

Variaciones debidas al deterioro

Hay múltiples causas que alteran la superficie de los materiales. En los ambientes rurales las superficies se deterioran por la fuerza del viento o la del agua. Tanto el uno como la otra van desgastando de modo irregularmente uniforme la superficie de muros o pavimentos hasta convertir una textura cuidadosamente pulimentada en una textura rugosa, con cavidades profundas.

En general, esto no implica alteraciones del color aunque el agua puede incorporar elementos que reaccionen químicamente con la piedra y el viento puede llevar insectos o partículas que se fijen a la superficie y alteren su color. Estos elementos pueden tener más importancia de lo que parece y quizás contribuyen a que las construcciones rurales mantengan una notoria armonía cromática con su entorno, si bien la razón de ser principal de esta armonía es, obviamente, que los materiales de construcción pertenecen al mismo entorno.

Los elementos metálicos también se oxidan o sufren diversas reacciones químicas y forman una pátina especial que se superpone a la superficie y altera su color.

En los ambientes urbanos las superficies se deterioran de un modo menos armónico y más radical. Las partículas de carbono que emiten los coches se depositan en las superficies y forman también una pátina pero que tiene poco que ver con la de los materiales metálicos.

Desde el punto de vista de la apreciación visual no hay diferencias entre el tipo de texturas generadas por características propias de los materiales y las generadas por el desgaste, del tipo que sea. Si las incluyo en un apartado diferente es para subrayar que el proceso por el que surgen es diferente y que esto, como veremos, puede estar ligado también a métodos diferentes de generación sintética.

Variaciones debidas a la distancia y a medios interpuestos

Los materiales reales se caracterizan por exhibir una diversidad de apariencias que se modifica con la distancia y que revela una estructura extraordinariamente compleja pero coherente con las diferentes escalas de observación.

Si nos acercamos a un material real lo que vemos cambia con la distancia. Una hoja de árbol, un ladrillo en un muro, un trozo de césped o la palma de la mano, pasan de ser una mancha de color a una distancia de 20 metros, luego a ser un conjunto armonioso de colores, luego a ser una serie de regiones independientes, con bordes internos que configuran otras manchas de colores que anticipan otros conjuntos armoniosos de colores y otra serie de regiones independientes.

No podemos proseguir porque nuestra capacidad de enfoque es limitada (más aún con la edad). Pero uno de los mejores modos de apreciar la complejidad de los materiales reales es revisar la magnífica obra *Powers of ten*.

En 1977, la oficina del arquitecto Ray Eames realizó una película de 9,5 minutos de dirección (para IBM) que mostraba el tamaño relativo de los objetos del universo al ir añadiendo (o quitando) un cero a la escala de observación. La película se plasmó en un libro del mismo título, preparado por Philip y

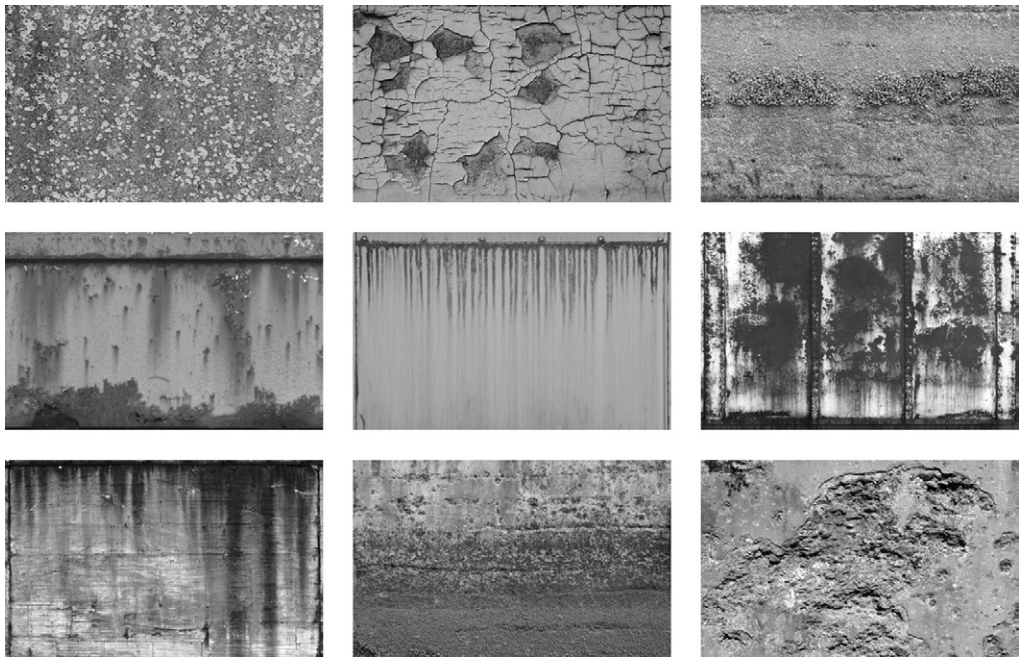
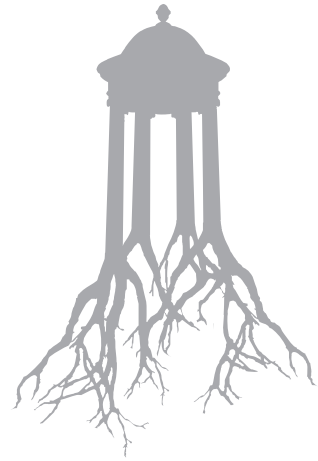


Figura 1.44 Ejemplos de texturas creadas por el desgaste o el deterioro de diversos tipos.

Phylis Morrison, que ya habían colaborado en la película, y se publicó en 1982 por Scientific American Books. Hay traducción española, véase las referencias bibliográficas.

Figura 1.45 Variaciones debidas a la distancia.



→2

Materiales virtuales

2.1 Características generales de los materiales virtuales

Una escena virtual es algo que cabe en un archivo de unos pocos Kb. Si se abre este archivo con un programa adecuado, lo que se encontrará es una larguísima enumeración de valores numéricos agrupados en categorías. Estas categorías comprenden todos los objetos que constituyen una escena: cámaras virtuales, luces virtuales, objetos virtuales. Y materiales virtuales relacionados con los objetos virtuales.

Los materiales virtuales, a su vez, están definidos por largas enumeraciones de valores numéricos también agrupados en categorías. Hay dos tipos principales de categorías que describen un material virtual: los parámetros básicos y las referencias a mapas que modifican estos parámetros. Los parámetros básicos son series de bytes que indican al programa qué propiedades se han asignado inicialmente al objeto.

En el caso más simple, todo lo que hay es una propiedad principal: el color asignado al material. Y el color se define por medio de tres valores (la intensidad de cada color primario rojo, verde y azul). Y estos tres valores se envían a la cámara que, a su vez, los envía a la pantalla del monitor.

Pero esta información elemental puede ser corregida de varios modos. Si se especifica que el material no es mate, los puntos sobre los que incide cualquier luz de la escena se aclararán debido a: *a*) el valor de de reflectancia del material, *b*) la intensidad de la luz, *c*) el ángulo de incidencia de la luz. O bien, si el material no es opaco, los puntos se modificarán debido a: *a*) al porcentaje en que el color propio sigue actuando, *b*) el color de los objetos que hay detrás para el punto de vista que sea. O bien, si se especifica que el material tiene textura, los puntos se modificarán debido al color correspondiente del mapa de textura que, según las especificaciones del tipo de proyección asignado al objeto, vendrán a substituir al color local.

Los materiales virtuales tienen muy poco que ver con los reales. Los materiales reales tienen un comportamiento extraordinariamente complejo pero este comportamiento es unitario: su interacción con la luz sigue las mismas leyes físicas en cualquier caso, aunque la descripción de esta interacción, como hemos visto en el capítulo anterior, pueda ser relativamente simple en algunos casos y muy complicada en otros. Los materiales virtuales simulan el comportamiento de los reales mediante abstracciones concatenadas que se van acumulando a medida que los investigadores que trabajan en este campo van descubriendo técnicas, trucos ingeniosos en muchos casos, que permiten simular el comportamiento de los materiales reales.

La primera y principal técnica es la simulación del color por medio de valores simples. Una técnica que, como veremos a continuación, tiene limitaciones importantes que es preciso conocer muy bien.

2.2 Generación de colores virtuales

Colores primarios y sistemas de mezcla

Desde tiempo inmemorial se sabe que es posible simular cientos o miles de colores a partir de unos pocos colores básicos. Cuando se pinta, la mayoría de los colores pueden obtenerse a partir de tres colores primarios, amarillo, rojo y azul. Los pintores saben, sin embargo, que muchos matices no se alcanzan con solo estos tres colores y un buen pintor tendrá de 12 a 20 colores en su paleta. Y, según su estilo, puede ser que incluya más pigmentos dentro de una determinada gama, no solo por sus tonalida-

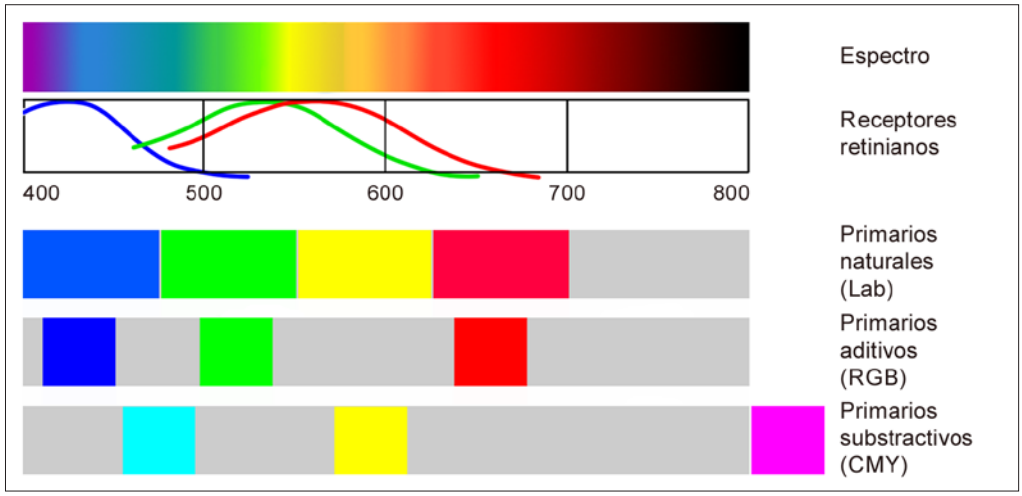


Figura 2.1 Colores primarios.

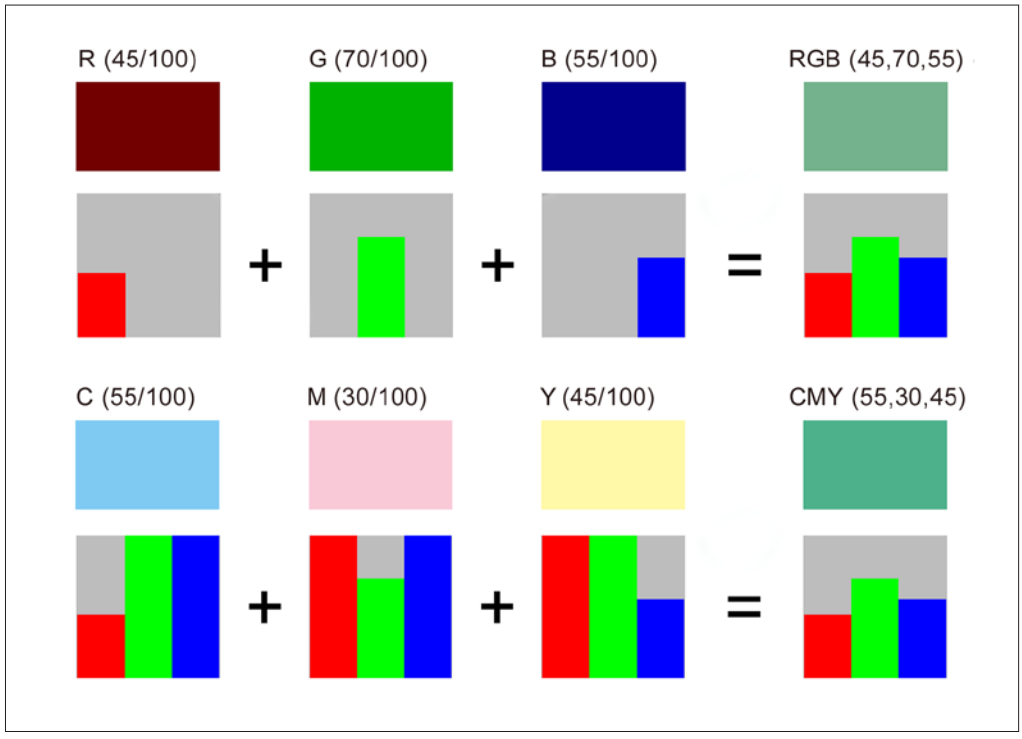


Figura 2.2 Mezcla aditiva (arriba): los primarios contribuyen directamente al resultado final. Mezcla substractiva (abajo): los primarios actúan como filtros que impiden el paso de su complementario. El resultado final es el mismo. En la práctica, los filtros no son perfectos y se requiere un cuarto color (negro).

des, sino por su comportamiento: el amarillo de Nápoles, por ejemplo, tiene unas tonalidades más suaves, menos saturadas, que el amarillo de cadmio, pero también se mezcla de modo diferente debido a que contiene plomo, lo que lo hace más pesado y más maleable. Los rojos acarminados son más puros desde el punto de vista de su rendimiento como color primario, facilitan las combinaciones; pero su comportamiento suele ser menos manejable que, por ejemplo, un rojo cadmio, también más pesado y más fácil de mezclar con otros pigmentos.

Pero solo desde el siglo XIX se comprende con claridad la diferencia entre mezcla aditiva y mezcla substractiva. Fue Thomas Young quien planteó como hipótesis, hacia 1810, que nuestro sistema visual debe incluir tres receptores especializados a partir de cuya combinación se puedan generar todos los colores, de modo similar a lo que hacen los pintores o los impresores, que no tienen que manejar miles de muestras para buscar la más adecuada para reproducir un color concreto. Como hemos visto en el capítulo anterior, esta hipótesis fue el comienzo de un largo proceso de investigación que culminó, a mediados de siglo XIX, con el descubrimiento de los tres pigmentos retinianos. Pero Young se equivocó al principio pues postuló que estos colores primarios retinianos debían ser especialmente sensibles al amarillo, el rojo carmín, y el azul claro, los colores primarios utilizados principalmente por los pintores y los impresores. Y luego rectificó y postuló, correctamente, unos primarios adecuados para mezcla aditiva.

Y fue James Maxwell quien lo confirmó, en 1861, creando imágenes en color mediante la superposición de colores primarios aditivos. Los colores primarios a los que los fotorreceptores de la retina son particularmente sensibles, son los los colores que corresponden a longitudes de onda larga (rojos), media (verdes) y cortas (azules). Los colores utilizados en mezcla substractiva se comportan realmente como filtros. La figura 2.2 explica este fenómeno, mal conocido, y que sigue desconcertando a quienes se han acostumbrado a pintar, es decir, a trabajar con mezclas substractivas, y pasan a trabajar con un programa informático, es decir, a trabajar con mezclas aditivas. Como se muestra en dicha figura, el fenómeno físico implicado es el mismo. Los filtros impiden el paso de las ondas correspondientes a su complementario y dejan pasar las demás. Pero la combinación de tres filtros da el mismo resultado que si mezcláramos directamente, por adición, las cantidades retenidas por cada uno de los tres filtros

Los trabajos iniciados por otro científico aleman, Hermann Grassmann (1809-1877) sentaron las bases de la colorimetria que permitía formular mezclas con precisión a partir de tres componentes normalizados. Maxwell,

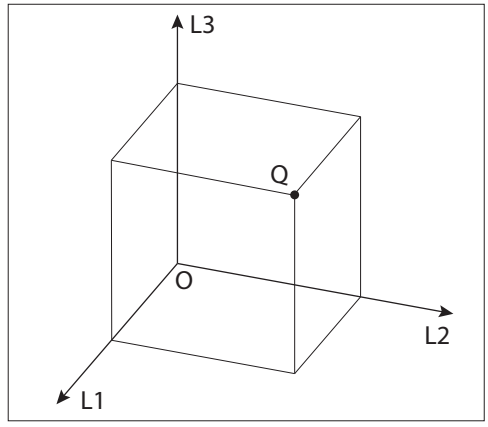


Figura 2.3 El espacio vectorial de colores de Grassmann. Cada color se representa mediante una suma de vectores luz primarios L1, L2, L3.

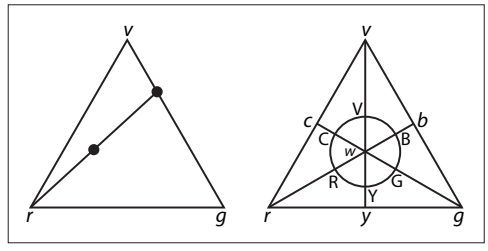


Figura 2.4 Triángulo de Maxwell. A la izquierda, el sistema de codificación: el punto sobre un lado se formula mediante su relación con los vértices; el punto interno, mediante su relación con los laterales. A la derecha, el ámbito de colores representables por el sistema (círculo interno). Adaptado del artículo original de Maxwell.

Grassmann y Helmholtz son, entre otros, los principales científicos que, en torno a 1850, proporcionaron los primeros fundamentos sólidos a la ciencia del color y las bases para la codificación de los colores.

Codificación de los colores

En 1931, La CIE (Comission Internationale de l'Éclairage), una organización fundada en 1913 por refundación de la Commission Internationale de Photometrie, a la que ya me he referido en el capítulo anterior, y que se ha consolidado a lo largo del siglo xx como el principal referente para la codificación de los colores, publicó el primer sistema de codificación riguroso que pronto se convirtió en una norma internacional. Sus resultados estaban basados en diversos experimentos llevados a cabo por varios investigadores (por David Wright, John Guild y otros), que se completaron hacia 1928.

Es importante tener presente que la especificación de la CIE se basaba en unos supuestos muy restrictivos que permitían superar la extraordinaria complejidad de la percepción real de los colores. Para que las especificaciones fueran universales se requería normalizar los tres agentes que intervienen en la percepción de los colores: el ojo, los objetos y la luz.

En primer lugar, se suponía un observador normalizado: una serie de sujetos jóvenes, sin defectos en la visión y cuyas respuestas se registraron para obtener valores de referencia medios. En segundo lugar, se suponían unas condiciones de observación concretas: el color debería ser observado aislado, bajo un ángulo determinado (de 2º) que abarcaría solamente los conos foveales (en 1964 se introdujo otra especificación menos restrictiva, para un ángulo más amplio, de 10º). Y, en tercer lugar, la muestra debería estar iluminada con un iluminante normalizado y bajo un determinado ángulo de incidencia de la luz. En el capítulo anterior también se han dado más detalles sobre estos iluminantes normalizados.

El triángulo de color o diagrama cromático de la CIE es un recurso similar al utilizado por Maxwell pero que simplifica su uso al especificar un color por medio de dos coordenadas referidas a un triángulo rectángulo y deducir la tercera, partiendo de la base de que la suma de las tres debe ser igual a 1. Tal como se muestra en la figura 2.5 (a), dados tres primarios específicos, un color cualquiera Q puede formularse mediante la suma de tres cantidades determinadas de estos tres colores primarios rojo (R), verde (G) y azul (B) mediante dos coordenadas *r*, *g*, que sitúan el color en el diagrama. Y la tercera cantidad resulta de la ecuación $r + g + b = 1$.

Sin embargo este triángulo nos da cantidades relativas. No tiene en cuenta las variaciones de luminosidad que se darían en la realidad. De hecho, el planteamiento del sistema se basa en la asunción de que todo color puede ser definido en un espacio cromático que cumple las leyes de Grassmann y donde resulta posible, por consiguiente, formular leyes de mezcla que permitan deducir cuál será el color resultante de la mezcla de unos colores dados. Uno de los innumerables espacios posibles sería un cubo normalizado, donde cada color vendría dado como suma de tres vectores cromáticos. En este espacio, un color dado vendría dado por la formula $Q = L1 + L2 + L3$ en donde *L*1, *L*2, *L*3 son tres colores luz adecuadamente escogidos con la condición de que sean independientes, es decir, que uno de ellos no pueda obtenerse por mezcla de los otros dos.

En paralelo, también en 1931, la CIE acordó los tres tipos normalizados de luz blanca que ya hemos visto en el capítulo anterior: el *Standard Illuminant A* (equivalente a la luz producida por una bombilla incandescente, definido como cuerpo negro a 2.848 K y que sigue vigente con su valor ajustado a 2.856 K), el *Standard Illuminant B* (en desuso, substituido por el D65) y el *Standard Illuminant C* (en desuso, substituido por el D65). A partir de 1964 se adoptaron los siguientes: el *Standard Illuminant D55* (luz de día a 5.000 K), el *Standard Illuminant D65* (luz de día a 6.500

K) que es el principal referente utilizado en la actualidad y el *Standard Illuminant D75* (luz de día a 7.500 K).

El diagrama cromático de la CIE debe verse, por consiguiente, como un corte en este espacio y cada punto del diagrama estaría atravesado por una recta que va del origen al infinito y que representaría cualquier color con las mismas características cromáticas pero diferente intensidad luminosa. El color *Q* sería, por tanto, un punto de la recta que une *O* y *Q*, y en la que se situarían los colores *Q*', *Q"*, etc., semejantes a *Q* pero más o menos luminosos.

Cuando se efectuaron los primeros experimentos para llevar a un diagrama de este tipo todos los colores que un sujeto normal puede percibir resultó, como ya le ocurrió a Maxwell, que había muchos colores que no podían ser reproducidos por medio de tres primarios. Sin embargo, también se comprobó que la coherencia de la teoría se podía mantener si se utilizaban colores negativos. ¿Y qué sentido real tiene un color negativo? La respuesta es muy sencilla para un científico aunque muy poco satisfactoria para un industrial: es un color que, si se lleva al otro lado de la ecuación, es decir, si se suma al color que se quiere reproducir, mantiene la consistencia de la fórmula. Aunque esto esté muy bien desde el punto de vista teórico, pues permite mantener la coherencia de las leyes de mezcla, es obvio que no resuelve el problema de reproducir un determinado color. La figura 2.5 (b) muestra el diagrama RGB que se obtuvo con estos primeros experimentos y donde queda claro que una gran parte de los colores naturales caerían en el lado negativo del diagrama.

Así que a partir de esta primera constatación se elaboró otro sistema que partía de unos primarios ideales, a los que se denominó X, Y, Z, que abarcarían todo el espacio cromático de los colores que un sujeto normal puede percibir. En este espacio se cumple la ley de que todo color puede obtenerse por mezcla aditiva de los constituyentes de la mezcla, es decir, que $Q = L1 + L2 + L3$. Es decir, la energía radiante de un color deter-

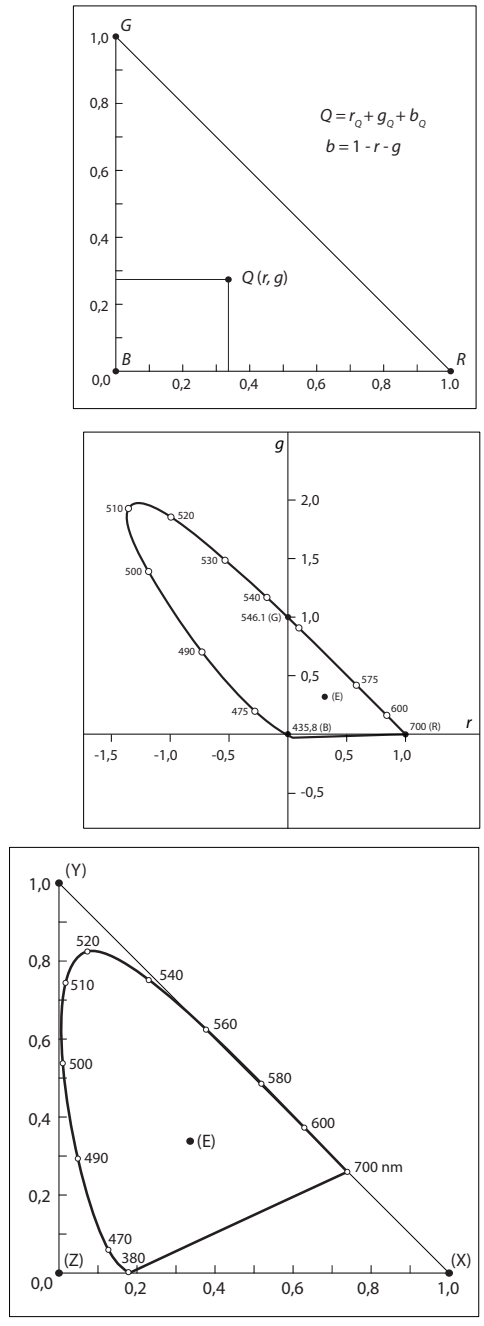


Figura 2.5 El sistema CIE XYZ: a) *Triángulo cromático, base del diagrama cromático de la CIE;* b) *Diagrama cromático, CIE RGB 1931;* c) *Diagrama cromático, CIE XYZ 1931.*

minado puede formularse como suma de la energía radiante de tres componentes dados. Si a esta ley se le añaden otra serie de leyes lineales (simetría, aditividad, transitividad, proporcionalidad), se lleva a cabo una generalización que permite operar de modo coherente en un espacio vectorial que constituye una representación adecuada del espacio cromático triestímulo.

El aspecto más sofisticado del sistema de la CIE viene dado por la independencia que establece entre la luminosidad de un color y el color en sí. Por diversas razones, interesa mantener separadas las cantidades que intervienen proporcionalmente en la especificación de color de la cantidad de luz correspondiente a dicho color. Una de estas razones es que el diagrama cromático representa un plano en el que los puntos tendrían diferente luminosidad. A igualdad de mezcla relativa, el color verde, por ejemplo, es más luminoso que el rojo o el azul.

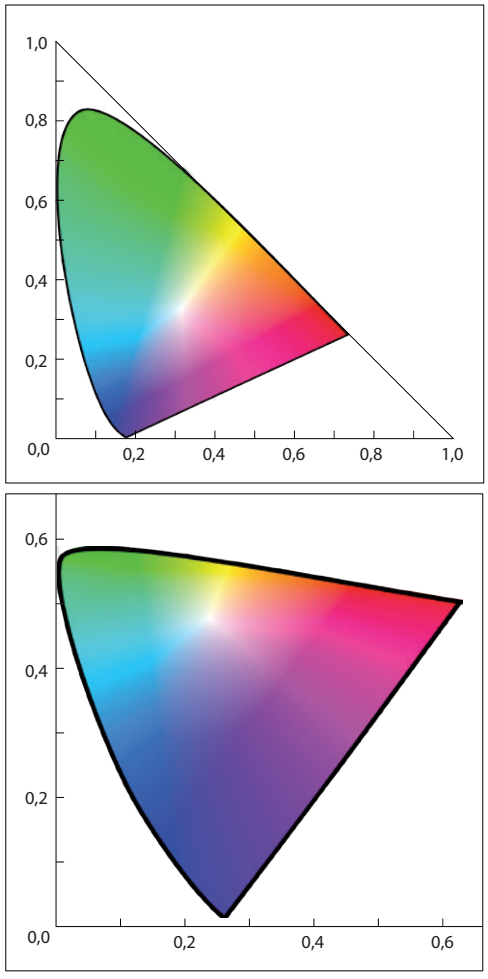


Figura 2.6 Los diagramas de cromaticidad CIE XYZ (arriba) y CIE LUV (abajo).

Se acordó así que el valor *Y* representaría la reflectancia o luminosidad del color objeto en forma de porcentaje. Un cuerpo perfectamente reflectante tendría el valor 100 y cuerpo perfectamente absorbente el valor 0. Alternativamente, se define *Y* como el factor de luminancia o luminosidad, definida como la relación entre la luminancia de un cuerpo dado y la de un reflector/difusor perfecto.

El color queda especificado por consiguiente, en el sistema CIE-XYZ, por dos coordenadas de cromaticidad *x*,*y*, más un valor de luminosidad *Y*. De hecho, otra denominación de este sistema es CIE xyY.

El plano del digrama básico no representa sino los colores espectrales y se considera, convencionalmente, que no tendría otra luminancia que la propia del color, como si tuviéramos la posibilidad de observar un objeto, en una cámara oscura, que emitiera energía luminosa en una longitud de onda determinada. Aunque esto implica la emisión de cierta cantidad de luz, se considera a este plano como plano de luminosidad nula, con Y = 0. A medida que se aumenta la luminosidad, dada por el valor de Y, puede considerarse que se asciende por un eje vertical donde la progresiva luminosidad de los colores va pareja con una disminución del ámbito del diagrama. El incremento de la luminosidad equivale a sumar luz blanca a todos los colores del diagrama, lo que se traduciría en una pérdida de saturación y en una disminución del rango de colores que pueden ser representados por el diagrama en unas determinadas condiciones de luminosidad. El espacio de color CIE xyY representa, por tanto, los colores mediante dos coordenadas, *x*,*y* que dan su posición en

el diagrama cromático y un valor Y que representa su luminosidad.
A partir de su implantación en 1931, el espacio de color XYZ o xyY se convirtió en una herramienta imprescindible para la codificación de los colores. Pero algunas de sus limitaciones fueron adquiriendo mayor importancia. Las dos principales darían lugar a dos nuevos sistemas de codificación.

La primera limitación se definió con precisión a partir de varios trabajos de investigación, desarrollados principalmente por David MacAdam y publicados en 1942, que mostraron que las diferencias de percepción de los colores de un sujeto normal no se correspondían con las diferencias de color del espacio de color CIE XYZ. Dicho de otro modo: este espacio de color no era uniforme desde el punto de vista perceptivo.

Después de diversas propuestas y una nueva codificación avanzada en 1964, se acabó por adoptar el espacio de color CIE 1976 (L*, u*, v*) denominado corrientemente CIE LUV. Este espacio puede derivarse del de 1931 mediante una transformación simple. No doy aquí las fórmulas de transformación pero el lector interesado las puede encontrar con facilidad en internet. La finalidad de este nuevo espacio era crear una distribución perceptual más uniforme, es decir, tal que una modificación numérica de uno de los valores se correspondiera con una modificación equivalente del valor percibido.

La segunda limitación era de más calado. Hacia esas fechas había quedado suficientemente probado que la teoría tricromática no podía explicar adecuadamente el modo en que percibimos los colores y que la teoría cuatricromática, o de primarios en oposición, avanzada por Hering a finales del siglo XIX era algo más que una interesante hipótesis alternativa. Como ya he mencionado en el capítulo anterior, esta teoría estaba siendo corroborada por los avances en el conocimiento del funcionamiento del cerebro y de los receptores neurológicos que se habían localizado en el córtex visual. Uno de los aspectos teóricos de los colores en oposición, de gran interés desde el punto de vista práctico, es el hecho de que dos colores opuestos se refuerzan mutuamente a una determinada distancia de observación pero se anulan mutuamente al aumentar esta distancia, como puede apreciarse en la figura 2.7.

En consecuencia, tras muchos años de discusiones, también en 1976, la CIE adoptó un nuevo sistema de codificación denominado CIE LAB, que puede considerarse la principal referencia actual y que está basado en tres coordenadas: *L*, que indica la luminosidad), *a*, que indica el predominio de rojo/verde (negativo/positivo) y *b*, que indica el predominio de azul/amarillo (negativo/positivo). Este nuevo espacio de colores también puede relacionarse con los anteriores por medio de fórmulas de transformación más complejas. El sistema de tres colores primarios se sigue utilizando en la práctica pues los métodos de generación obedecen a lógicas distintas que los métodos de especificación.
Desde la aparición de los televisores y, algo más tarde, de los ordenadores, la generación de colores por medio de tres primarios aditivos ha pasado de ser una teoría científica que

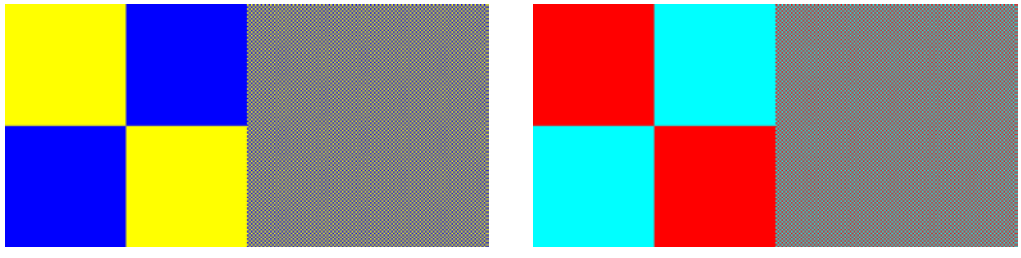


Figura 2.7 Complementarios amarillo/azul y rojo/verde a diferentes escalas.

solo tenía relación con la práctica de los escenógrafos (que iluminaban la escena de los teatros con tres focos: rojo, verde y azul) a ser una teoría relacionada directamente con la práctica de un número creciente de usuarios.

Por otro lado, la importancia que la fotografía y la impresión en color han adquirido a lo largo del siglo xx, ha dado lugar a múltiples sistemas de codificación de mezclas substractivas. Y, por añadidura, el desarrollo de la teoría de los colores y de la práctica profesional también ha propiciado la aparición de otros sistemas de codificación basados en variables perceptivas, así como sistemas de ordenación que puedan traducirse en catálogos o referencias manejables.

La consecuencia de todo esto es que hay decenas de sistemas de codificación y ordenación que se han utilizado y se siguen utilizando. Aunque en algunos casos la frontera sea difusa hay que distinguir entre sistemas de codificación y sistemas de ordenación.

Un *sistema de codificación* es un modo preciso de especificar un color, que puede estar ordenado o no. Un *sistema de ordenación* es un modo sistemático de ordenar los colores, que puede estar codificado o no.

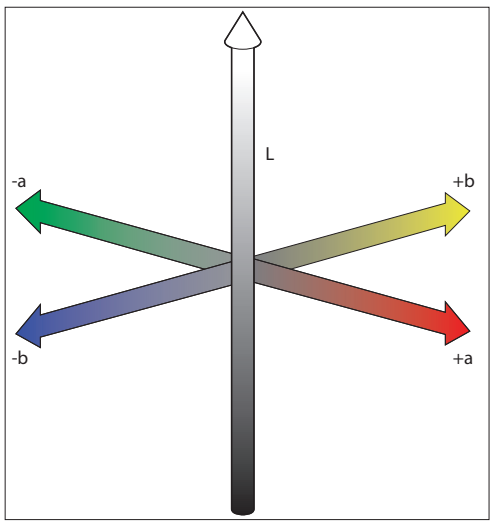


Figura 2.8 El espacio de color CIE Lab.

El sistema de codificación que he descrito hasta aquí, el sistema de la CIE con sus diferentes variantes, es un sistema de codificación universal que asegura que dos colores con la misma codificación serán indistinguibles si se observan en condiciones iguales. Pero, por razones prácticas, hay industrias que utilizan otros sistemas (que pueden y deben traducirse automáticamente a los de la CIE).

En televisión se han desarrollado métodos de codificación específicos que son deudores de la propia historia de la televisión, y que se remontan a los primeros ensayos de TV en color que datan de 1926 (los llevó a cabo J. L. Baird en la Royal Institution de Londres). Pero dado que la mayoría de los aparatos emitían en blanco y negro, la señal de *luminancia* tenía que transmitirse con independencia de la señal de color o señal de *crominancia*. Esto se hacía y se sigue haciendo así por diversas razones, la principal de las cuales es la compatibilidad y la retrocompatibilidad, esto es a la capacidad de un sistema de transmisión en color para ser captado por un receptor en blanco y negro, prescindiendo de la información en color y sin que esto afecte a la recepción monocromática de la imagen y, por otro lado, a la capacidad de un receptor en color para captar señales en blanco y negro, sin que sus características afecten a la recepción monocromática de la imagen. Una segunda razón por la que se envía por separado la señal de luminancia es que esto permite un considerable ahorro de banda. La discriminación de detalle depende mucho más del contraste de luminancia que del contraste del color. Esto quiere decir que si se transmiten por separado la señal de luminancia y la señal de color, se ahorraría ancho de banda enviando la primera a altas frecuencias y la segunda a frecuencias reducidas. Todo esto ha llevado a sistemas específicos que se han incorporado de diferentes modos a los tres principales sistemas actuales. En 1953 el NTSC (National Television Systems Committee) recomendó un sistema que fue adoptado en Estados Unidos y constituye la norma seguida por todas las emisoras de este país y la mayoría de las

de América Latina. Una década más tarde se implantaron en Europa los sistemas SECAM y PAL. El sistema SECAM (*Sequence and Memory*) se comenzó a utilizar en Francia, Rusia y paises de la órbita rusa de influencia, a partir de 1967. El sistema PAL (*Phase Alternation Line*) se comenzó a utilizar en Gran Bretaña y otros paises europeos, entre ellos España, pocos años después.

El sistema utilizado actualmente, tanto por NTSC (Estados Unidos y América Latina principalmente) como el PAL (Europa y otros países) es el *YUV* en donde el parámetro *Y* codifica la luminancia y los parámetros *U*, *V* codifican la crominancia. Hace años la NTSC utilizaba el sistema YIQ, ya en desuso (Y codificaba la luminancia, como el actual y los parámetros I, de *in-phase* la fase y Q, de *quadrature*, la crominancia). Y hay otros sistemas de codificación utilizados, por ejemplo, en imprentas, muchos de los cuales han sido progresivamente abandonados por sistemas más generales.

Conviene volver a insistir, antes de dejar el tema de la codificación, que especificar colores por medio de tres primarios es una referencia que, si se utilizan primarios reales en lugar de los primarios ideales de la CIE, no cubre la generación de colores reales. Una codificación RGB es una forma de comprimir drásticamente, con pérdida, lo que nos daría una medición espectrofotométrica. De hecho, también una medición de este tipo supone un recorte, aunque mucho menos drástico, pues se toman tan solo, por razones prácticas, una serie de muestras. Las organizaciones internacionales recomiendan tomar muestras cada 5 nm para que la medición sea suficientemente exacta. Entre 300 y 700 nm esto supondría 60 muestras, que es un valor muy alto. En la práctica se toman cada 10 o 20 nm, es decir, entre 30 y 15 muestras.

Un sistema de ordenación es un catálogo, un atlas, una referencia para la práctica, que ayuda a elegir y combinar colores. Pues en la práctica de diferentes profesionales ha pesado desde siempre la necesidad de contar con algún sistema de ordenación relativamente simple que permita ordenar y clasificar los colores y que vaya más allá de clasificaciones tales como los catalógos comerciales de fabricantes de pinturas o tintas. La lista de modelos propuestos es muy larga pero me referiré brevemente a tres o cuatro.

Hacia 1810, el pintor Otto Runge, amigo de Goethe, propuso un modelo de clasificación

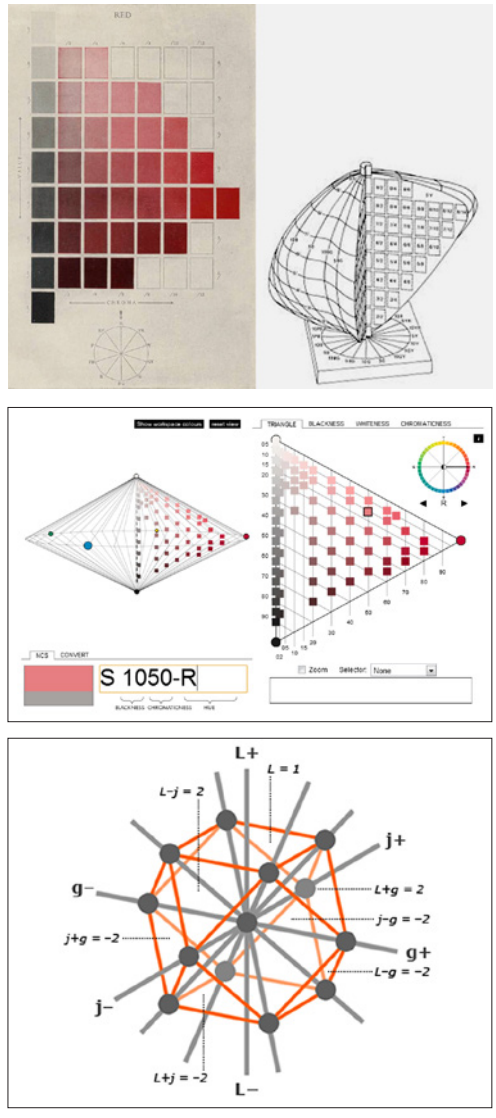


Figura 2.9 Los sistemas de ordenación de: a) *Munsell;* b) *NCS;* c) *OSA.*

de los colores en un libro editado con el título de *Die Farbenkugel* (La esfera de color). El modelo de Runge consistía en una esfera con un eje vertical acromático, blanco en el polo superior, negro en el polo inferior, y formando una escala regular de grises a lo largo de este eje. El círculo de colores puros o primarios formaba el ecuador y el resto de los colores terciaros quedaban en el interior de la esfera.

El modelo de Runge tenía una ventaja interesante pero un defecto no menos interesante. La ventaja, que era claro y unitario, fácilmente comprensible. El inconveniente, al igual que muchos otros modelos que habían aparecido antes (como uno propuesto por Lambert hacia 1772 y que consistía en un triángulo de primarios que se iba reduciendo y oscureciendo hasta formar una pirámide) y que aparecieron después, es que no era coherente. Si el eje vertical representa a los colores con diferente luminosidad y el ecuador a los colores con máxima saturación, ocurre que los colores con máxima saturación no tienen el mismo nivel de luminosidad: el amarillo más saturado es más luminoso que el azul más luminoso. Por otro lado, Runge tampoco daba criterios claros para construir sus escalas de luminosidad, saturación y tono.

Otros modelos que aparecieron durante el siglo xx no aportaban soluciones a estos problemas. Se pueden mencionar los de Chevreul (1868), un sólido semiesférico muy similar a la pirámide de Lambert, el de Rood (1879), un doble cono que en poco se diferenciaba de la esfera de Runge, el de Charpentier (1883), un cubo en el que cada vértice representaba los tres primarios, sus mezclas secundarias, el blanco y el negro o el de Höfler (1897), que recogía la teoría cuatricromática recién propuesta por Hering.

La solución, que daría lugar a un sistema muy utilizado durante el siglo xx y que sigue siendo una referencia importante, la dio, hacia 1900, el pintor americano Munsell (1858-1918) con un modelo sólido bastante sencillo. Munsell partía de tres categorías perceptivas bien diferenciadas: el tono (*hue*), el valor (*value*) y el croma (*chroma*). Cada una de estas tres categorías daba lugar a una escala de 10 grados. Los tonos se disponían en un círculo que estaba formado por 5 primarios: amarillo, rojo, púrpura, azul, verde (Y, R, P, B, G, según las iniciales inglesas) y sus 5 intermediarios. Los valores se disponían en una escala vertical de 10 grises que iban del negro 0, al blanco 9. Y los cromas en una escala de unos 5 grados de pureza que iba desde la máxima saturación disponible hasta el gris correspondiente en el eje vertical. A diferencia de las anteriores, la amplitud de esta escala no se especificaba sino que quedaba limitada por la pureza de los pigmentos disponibles que, teóricamente, siempre podría ser mejorada.

De este modo se podían nombrar algo menos de 1.000 colores con considerable precisión. Asi, por ejemplo, el término R5/8 describiría un color de tono rojo, valor 5 y grado de saturación 8. El término 2.5YR5/10 describiría un color de tono entre el amarillo y el rojo, 2.5YR, valor 5 y grado de saturación 10. Además de esta precisión en los términos, el modelo de Munsell, a diferencia de los que le precedieron y de los que le siguieron, era coherente, es decir, tenía en cuenta que los diferentes tonos del círculo cromático poseen, intrínsecamente, diferentes valores de luminosidad por lo que no pueden estar en el mismo nivel de la escala de grises vertical. Dicho de otro modo; en el modelo de Munsell se tenía en cuenta que los diferentes tonos alcanzaban su punto máximo de saturación, su pureza, en diferentes niveles de la escala acromática. El amarillo en el nivel más alto de la escala, más cerca del blanco, el violeta en el nivel más bajo, más cerca del negro. Y, por añadidura, se consideraba que ciertos colores cuentan con mayor cantidad de grados de saturación que otros. Así, por ejemplo, en condiciones similares, el rojo sería dos veces más fuerte que el azul verdoso.

La esfera de Runge dejaba de ser una esfera regular y se transformaba en una especie de patata. El idealismo alemán quedaba así corregido por el pragmatismo americano.

En 1943, la CIE completó una serie de análisis para especificar las muestras del sistema de Munsell por medio del sistema CIE y publicó un informe, elaborado por Newhall, Nickerson y Judd, "Final Report of the OSA Subcommittee on the spacing of Munsell colors", conocido como la "renotación Munsell de 1943" que ha servido como referencia para que los subsecuentes catálogos Munsell incluyeran especificaciones CIE junto con sus códigos.

Hay muchos otros sistemas que cabría citar pero baste con mencionar, en el contexto de este libro, cuatro sistemas importantes, el sistema de Munsell, el sistema DIN (bastante similar al de Munsell), el sistema NCS (*Natural Color System*) y el sistema OSA (*Optical Society of America*). Tres de ellos se ilustran en la figura 2.9 pero no los describo con detalle.

Rangos absolutos de formatos de grabación y dispositivos de reproducción

En el capítulo anterior hemos visto que el número de colores que el ojo humano puede distinguir, difícilmente superaría los 2 millones en condiciones ideales, cifra que sería todavía muy inferior, probablemente del orden de la cuarta parte o menos, en situaciones relativamente reales, es decir, menos de 500.000.

¿Qué sentido tiene, entonces, que la publicidad de cualquier ordenador nos diga que trabaja con "color real" y que por tanto puede generar 16,7 millones de colores (16.777.216, 2^{24} para ser exactos)?

Lo primero que hay que decir es que, de este total, por las razones dadas, distinguiríamos realmente, como mucho, no más de un 3 %. De hecho, en la mayoría de los casos nos basta con mucho menos. Un modo sencillo de comprobar hasta qué punto esto es así es tomar una imagen en color real (3 bytes por píxel) en la que no haya superficies continuas con degradados y grabarla en modo de color indexado. Para hacer esto en un programa como Photoshop, por ejemplo, todo lo que hay que hacer es ir al menú Imagen/Modo/Color indexado. La imagen disminuirá su tamaño drásticamente y el número de colores pasará de 16,7 millones a 256 (1 byte por píxel) con lo que tendremos aproximadamente 0,001 % de los colores que teníamos. Sin embargo no se apreciarán diferencias en pantalla.

Pero si la imagen contiene degradados continuos, al disminuir el número de colores se apreciarán discontinuidades, un efecto conocido como bandas de Mach por haber sido analizado por primera vez por el físico y filósofo austríaco Ernst Mach (1838-1916). Nuestro sistema de visión es especialmente sensible a diferencias de continuidad y exagera las discontinuidades cuando el número de niveles de gris es insuficiente. Por esta razón, se requiere un número mínimo de niveles de luminosidad para cada uno de los colores primarios y, si esto no ocurre, se pueden percibir discontinuidades.

Para que esto no ocurra necesitamos un número de valores que estaría en torno a 200. Si utilizamos valores digitales tenemos que elegir entre 7 bits por canal, lo que daría 128 valores (2^7) u 8 bits por canal, lo que daría 256 valores (2^8). Pero 7 bits por canal es una pésima especificación pues no se presta con facilidad a las operaciones digitales corrientes mientras que con 8 bits (1 byte) pasa lo contrario: puede subdividirse sistemáticamente en valores enteros inferiores hasta llegar a 1 bit. Esta es una de las razones principales por las que se utiliza este rango para codificar escalas de grises.

Si la imagen es en color, cada uno de los colores primarios deberá ser capaz de generar un mismo rango de valores para que, en su canal respectivo, los degradados también aparezcan continuos. Esto quiere decir que necesitamos 8 bits por cada uno de los tres canales y, por tanto, un total de 24 bits. Es decir, los 16,7 millones de colores de que hablábamos.

Si la imagen no incluye degradados o por razones prácticas queremos que su tamaño se mantenga dentro de un límite, aunque los degradados aparezcan discontinuos, podemos utilizar lo que se denomina "color in-

dexado", que se basa en un recurso interno denominado LUT (*Look Up Table*). La idea es sencilla. En lugar de generar los colores por medio de valores dados dentro de un rango, se selecciona un número determinado de colores, concretamente no más de 256, que se almacenan en una lista con sus tríadas rgb. Cada uno de los 256 valores apunta por tanto a una tríada, a un color específico.

Este último mecanismo pone a disposición del usuario una paleta limitada. ¿Cómo se genera esta paleta? Si se parte de una imagen en color real el procedimiento se puede llevar a cabo automáticamente mediante algún tipo de algoritmo que descarte colores cercanos y se quede tan solo con los más representativos. No hay ni puede haber un método único para llevar a cabo esta selección. Por esta razón, los programas de manipulación de imágenes ofrecen diversas alternativas. En Photoshop, por ejemplo, se encuentran, entre otras, las siguientes: Perceptual (da prioridad a los colores a los que el ojo humano es más sensible), Selectiva (similar a la anterior pero más amplia y dando prioridad a colores compatibles por web), Adaptable (da prioridad a los colores que aparecen con más frecuencia), A medida (permite adaptaciones manuales). Este tipo de conversión puede ser muy interesante para determinadas aplicaciones, incluidas las arquitectónicas.

Así que tenemos dos formatos principales, en primer lugar, un formato, color real, que nos da 24 bits por píxel lo que supone 256 valores por cada canal. Y, en segundo lugar, otro formato, color indexado que nos da un total de 256 colores referidos a una tabla creada de diversos modos posibles.

En tercer lugar, podemos encontrarnos, por el otro extremo, con rangos absolutos superiores, con 32 bits o 48 bits por píxel. Y, en cuarto lugar, con diferentes sistemas de codificación, en concreto con el formato HDR.

Un archivo con 32 bits por píxel es, por lo general, un archivo que incluye un cuarto canal, un canal alfa, que permite procesar transparencias. En este caso no hay diferencia con respecto a los formatos anteriores pues los otros tres canales utilizan el mismo rango, 8 bits que equivalen a 256 valores.

Si nos encontramos con un archivo de 48 bits por píxel que codifica 16 bits por canal lo más probable es que estemos en presencia de una variante del formato HDR.

El cuarto formato importante, HDR se trata extensamente en el libro sobre simulación de la iluminación, al igual que todo lo relativo a rangos dinámicos de dispositivos de entrada (cámaras y escáners) y de dispositivos de salida (monitores e impresoras). Aquí me limitaré a resumir lo principal desde el punto de vista de los colores. Un formato HDR (*High Dynamic Range*) nos permite almacenar muchos más valores que el formato en color real. Y esto tiene una gran importancia tanto desde el punto de vista de su uso en edición de imágenes como de su uso en simulación visual 3D. Las dos variantes del formato más utilizadas en la actualidad son HDR y EXR.

El formato HDR fue introducido por Greg Ward hacia 1985 y ha sido desarrollado principalmente por Paul Debevec hacia 1998. Las imágenes grabadas en este formato se obtienen combinando diferentes exposiciones. Como sabe cualquier aficionado a la fotografía si, por ejemplo, se saca una fotografía a pleno día con luz brillante y se ajusta la exposición para que las zonas iluminadas queden bien expuestas, los detalles de las zonas en sombra se fundirán en una zona oscura homogénea. Y si, a la inversa, se saca una fotografía de un interior y se ajusta la exposición para que los objetos del interior queden bien expuestos, los objetos que aparezcan a través de una ventana aparecerán quemados, sobreexpuestos, fundidos en una zona clara homogénea. Combinando diferentes tomas en un mismo archivo de más capacidad se obtienen rangos dinámicos internos mucho más extensos que permiten alcanzar distribuciones más precisas. Véase el libro sobre simulación de la iluminación para una descripción detallada de los métodos de obtención de este tipo de archivos o buen manual sobre este formato. Aquí me limitaré a resumir sus características básicas.

El formato HDR se presenta en dos modos principales: RGB y XYZE (tres primarios imaginarios, XYZ y un exponente, E). En los dos casos, los bits se organizan de modo que hay una mantisa para cada color y un exponente común (8 + 8 + 8 + 8). Los tres primeros canales almacenan la información de color primario pero el cuarto canal introduce un exponente que amplía los valores de luminancia. El valor de cada píxel se obtiene según la fórmula ((R,G,B)/255) x 2(E - 128). El rango dinámico resultante de utilizar 32 bits de esta forma es equivalente a utilizar 96 bits, lo que equivaldría a 76 órdenes de magnitud o más de 250 EV, mucho más de lo que podemos percibir pues ya los escenarios reales no suelen sobrepasar los 20 EV en condiciones extremas ni los 16 bits en condiciones corrientes.

El formato OpenEXR es una variante que proporciona un rango dinámico equivalente a 30 pasos EV. Su precisión es mayor que la del anterior y se comprime muy bien. Hay dos versiones del formato, la versión *half*, de 16 bits y la versión de 32 bits. La versión *half*, o *half RGB*, más utilizada, utiliza un formato de números de 1 bit para el signo, 5 bits para el exponente y 10 bits para la mantisa o valor significante. Esta mantisa permite guardar 210 = 1.024 valores por cada canal de color. Proporciona un rango dinámico equivalente a 30 pasos EV.

El uso de formatos HDR permite cálculos internos de una gran precisión. Pero tanto para visualizar el archivo de entrada como para visualizar el resultado, dado que los dispositivos de salida que utilizamos corrientemente (monitores o impresoras) son de bajo rango dinámico, necesitaremos convertir la imagen a LDR (*Low Dynamic Range*) por medio de alguno de los varios algoritmos de conversión denominados genéricamente de *Tone mapping*. Me remito de nuevo al libro sobre simulación de la iluminación donde también se trata más extensamente este tema.

Todo lo anteriores está referido al rango que nos ofrecen los formatos de los archivos que utilizamos para almacenar la información de color de un material virtual. Ahora bien, cada dispositivo que utilicemos para trabajar o para representar el resultado tendrá, a su vez, un determinado rango cromático, o bien, al igual que ocurre con los instrumentos musicales, un determinado registro será capaz de generar una determinada gama de colores que pueden ubicarse en posiciones precisas en el espacio de color propio del dispositivo.

Para empezar hay que subrayar que la gama de colores que un dispositivo es capaz de reproducir es siempre un subconjunto de todos los colores que somos capaces de percibir. Esta gama de colores puede representarse en un plano de dos dimensiones en donde se muestran tan solo los tonos y las saturaciones que corresponderían a un determinado nivel de luminosidad, aunque esto no sea del todo exacto pues las saturaciones máximas no se producen al mismo nivel de luminosidad (el amarillo y el azul por ejemplo, alcanzan su máximo en niveles más luminosos para el primero y menos para el segundo, como ya hemos visto) pero es suficiente para hacerse una idea adecuada de la gama.

Pero, además de que no existe ningún dispositivo que pueda generar todos los colores que podemos percibir, lo que resulta más grave desde el punto de vista de la práctica profesional es que la gama de colores propia de los diferentes dispositivos que utilizamos no es concordante ni puede serlo.

La figura 2.10 muestra la gama de varios dispositivos dentro del espacio de todos los colores posibles representado en el diagrama de la CIE XYZ. Como puede apreciarse en esta figura, hay muchos colores que podemos percibir pero que no pueden ser generados por ningún dispositivo. Y otros colores que pueden ser generados por unos dispositivos pero no por otros. En particular, es muy importante recordar que hay colores que se pueden reproducir en un monitor pero no en una impresora. Estos colores se dice que están fuera de gama. Muchos progamas, como Photoshop, activan un aviso cuando se está utilizando un color fuera de gama para

recordarle al usuario que lo que está viendo en pantalla no lo podrá ver en papel. Y ofrecen una alternativa (si, en Photoshop, desde el selector de color, se hace un clic sobre el aviso, el selector salta al color alternativo más próximo que esté en gama). Pero esta utilidad no se encuentra en los programas de simulación 3D, por lo que es recomendable tomar precauciones y evitar colores excesivamente saturados sobre todo en tonos que se reproducen peor sobre papel.

Por otro lado, el espacio de color de un mismo tipo de dispositivo, como puede ser un monitor, tampoco coincide con otro, a no ser que esté perfectamente calibrado y que utilice un mismo sistema de especificación del espacio cromático. Dada la importancia de este tema será conveniente añadir algo aunque tan solo sea como introducción a un tema bastante complejo.

Sistemas digitales de control y gestión de los colores. Modo. Espacio. Perfil. Módulo de gestión

Para controlar el color que vamos a reproducir se necesita manejar una serie de conceptos básicos ligados a técnicas específicas. El conjunto de técnicas utilizadas para controlar el color y para grabar la configuración de estos controles de tal modo que no tengamos que repetir el proceso una y otra vez se denomina genéricamente gestión del color (*Color Management*). Esto implica varios conceptos y términos que hay que comenzar por precisar.

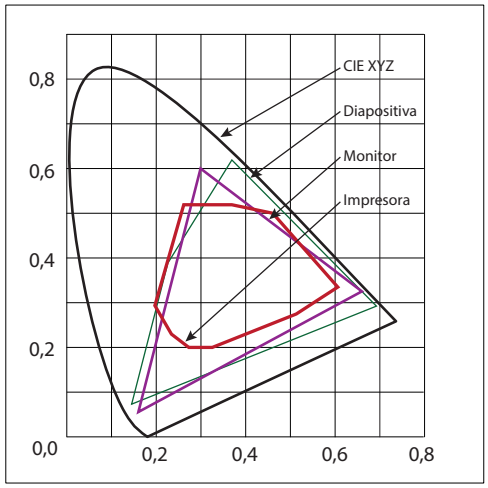


Figura 2.10 Gama de color de diferentes dispositivos.

El *modo* de color o modelo de color es ni más ni menos que la codificación utilizada para describir el color, en términos de 3 o más coordenadas, lo que está ligado a un espacio de color elemental y a un determinado formato de grabación de archivos. Es un concepto familiar pues hay al menos dos modos de color que están presentes en todos los programas: RGB y HSB. Pero hay algunos más, principalmente CMYK y CIE LAB. Todos estos modos ya se han descrito más arriba.

El *espacio de color* es el espacio concreto que se genera a partir de un determinado modelo y a partir de una determinada especificación de colores primarios, y que se caracteriza porque tiene dimensiones concretas, es decir, un número mayor o menor de colores que abarcarán una u otra zona del espacio ideal que se supone que representa a todos los colores posibles. Hay tres espacios principales de color que se encuentran en la práctica. El más utilizado es el espacio de color sRGB, un espacio de color seguro (pues la mayoría de los dispositivos pueden reproducirlo) pero, por esta misma razón, más limitado que otros. Este espacio de color se desarrolló por Hewlett-Packard y Microsoft, fue presentado en 1996 y fue adoptado por las principales industrias de impresión y gestión del color. Otro espacio de color bastante utilizado, más amplio que el anterior, sobre todo en las tonalidades verde y azul-verdoso, es el espacio de color Adobe RGB, desarrollado por Adobe Systems en 1998 con el objetivo de reproducir adecuadamente colores en modo CMYK desde un espacio RGB. Se considera que abarca alrededor del 50 % de la gama de colores visibles que se especifican en el CIE LAB. Y un tercer espacio de color importante en la práctica profesional es ProPhoto RGB, desarrollado por Kodak, con una gama más

amplia que los anteriores y que solo puede ser usado con formatos de 16 bits por canal, cuyo uso está más bien restringido a fotografos profesionales.

Es importante subrayar, de todas formas, que el espacio de color de una buena cámara es más amplio que cualquiera de los dos anteriores, más utilizados, sRGB o Adobe RGB, por lo que si un buen fotógrafo quiere sacar el máximo partido de su cámara tendría que trabajar con éste espacio que está disponible directamente si se utiliza el *Camera Raw* de Photoshop y se graba el resultado en un formato de 16 bits. Sin embargo no se podrá visualizar en un monitor corriente. La principal ventaja es que proporciona mayor flexibilidad a la hora de escoger uno u otro recorte o substitución en lugar de aceptar que este recorte o substitución se lleve a cabo automáticamente por uno de los dos espacios de color anteriores.

El *perfil* de cada dispositivo que utilicemos para captar, visualizar, manipular o reproducir un color (cámaras, escáners, monitores o impresoras) es una especificación que describe con precisión su modo, su espacio de color y otras características relevantes. Es decir, el lenguaje particular con que tal dispositivo describe los colores. Solo si conocemos este lenguaje, este perfil, podremos traducirlo a otros lenguajes, a otros perfiles. Estos perfiles pueden ser genéricos o, en casos más profesionales, creados por el propio usuario a partir de dispositivos tales como colorímetros o espectrofotómetros.

Pero para que los diferentes lenguajes o perfiles se puedan comunicar entre sí necesitamos un lenguaje compartido. Este lenguaje compartido es el ICC y, cada vez que se crea un perfil, se crea lo que se denomina un perfil ICC (*ICC profile*), es decir, una descripción del dispositivo en términos tales que puedan ser entendidos por otros dispositivos. Los perfiles ICC derivan de la existencia de este organismo cuyo origen se remonta a 1993, cuando Apple desarrolló Color Sync, un sistema de gestión de color incorporado a sus ordenadores que utilizaba perfiles de diferentes dispositivos para facilitar la traducción entre los diferentes espacios de color. Y se creó un consorcio de empresas que utilizaban este mismo sistema para evitar todos los problemas derivados de no contar con un lenguaje común. Posteriormente, este consorcio de empresas se estableció y fue reconocido como consorcio internacional con el nombre de ICC (*International Color Consortium*) y se extendió a los ordenadores que utilizaban el sistema operativo Windows y Unix. El documento principal generado por este consorcio es el perfil ICC (*ICC profile*).

Un perfil ICC tiene cuatro atributos principales: el rango o gama cromática, el factor *gamma*, el punto negro y el punto blanco. La gama cromática (*gamut*) describe el rango de colores que el dispositivo es capaz de reproducir. Y esto depende del substrato material del dispositivo: los puntos de fósforo de un monitor antiguo de tipo CRT, los cristales líquidos de un monitor LCD, los pigmentos o colorantes de una impresora, tienen un rango limitado y diferente de colores, que ya se ha mostrado en la figura 2.10. El factor *gamma* (un valor que es corrientemente de 2,2 en la mayoría de los monitores actuales) describe cómo se distribuyen las intensidades según una curva característica. Véase el libro sobre

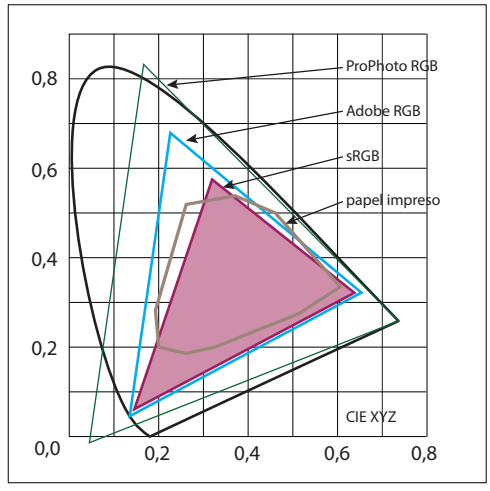


Figura 2.11 Espacios de color RGB utilizados por la industria como estándares convenidos.

simulación de la iluminación para una explicación más amplia de todo lo que está implicado en el factor gamma. El punto negro describe el negro más negro que el dispositivo es capaz de reproducir (y que siempre es superior al 0 % de luminancia). El punto blanco describe el blanco más blanco que el dispositivo es capaz de reproducir (y que es un valor muy bajo en comparación con fuentes de luz más o menos corrientes). Este valor se especifica en términos de intensidad y temperatura de color. Algunos perfiles ICC también dan como referencia el blanco medio denominado a veces blanco papel. Un valor de referencia habitual para el blanco es D65, que se refiere al blanco normalizado de la CIE que ya hemos visto.

Por último, para poder llevar a cabo la traducción de valores entre dispositivos, necesitamos un programa o una tecnología específica capaz de llevar a cabo esta conversión manejando los componentes anteriores (perfiles, espacios). En otras palabras, necesitamos un programa que incorpore lo que se denomina un *módulo de gestión del color* o CMM por sus siglas en inglés (*Color Management Module*) o CMS (*Color Management System*). Los programas avanzados de manipulación de imágenes, como Photoshop, ya lo incluyen y suelen emitir mensajes, familiares a los usuarios (aunque en muchos casos no sepan muy bien de qué les están hablando), sobre el espacio de color que incorpora un determinado archivo que se está abriendo. Es muy probable que quien lea esto, si tiene cierta experiencia con programas como Photoshop, se haya encontrado alguna vez con un mensaje en que se le informa de que "el archivo que se está abriendo tiene insertado un espacio de color tal y tal. ¿Desea conservarlo o prefiere substituirlo?".

Estos módulos también incluyen diferentes opciones para interpretar los colores que no caben en el espacio de color que estamos utilizando. Los perfiles ICC incluyen cuatro posibilidades de transformación para los colores que caen fuera del espectro o del espacio utilizado: *perceptual*, *saturación*, *relativa* y *absoluta*, muy similares a las que he citado a próposito de la conversión de un formato en color real a color indexado. Simplificando, puede decirse que las dos primeras quitan saturación de los colores de origen para encajarlos en el espacio de color (la perceptual intentando mantener las relaciones entre tonos), mientras que las dos segundas buscan el color más cercano al color que cae fuera del espacio

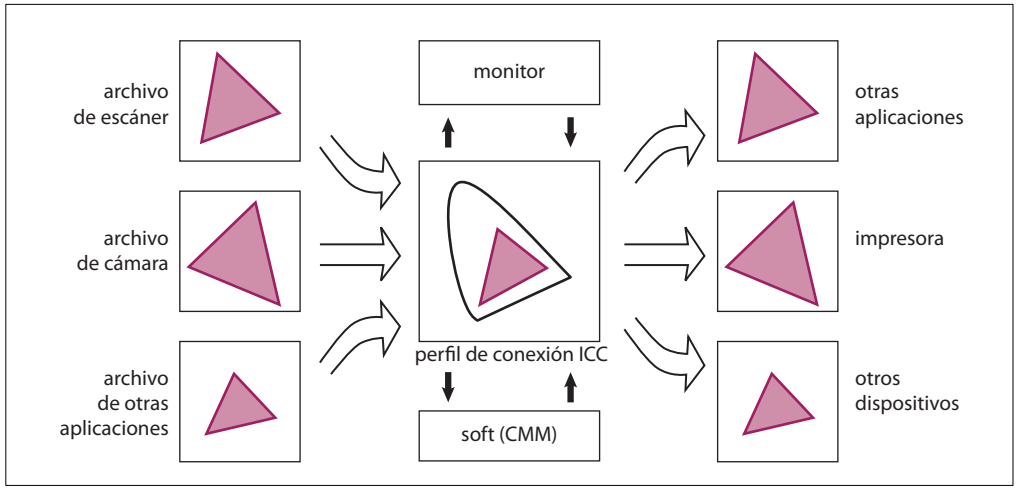


Figura 2.12 Gestión del color. Flujo de trabajo.

(la relativa convierte el blanco de origen en el de destino y ajusta los demás proporcionalmente, mientras que la absoluta intenta mantener al máximo los colores de origen, incluyendo el blanco). En Photoshop, por ejemplo, estas opciones de conversión se encuentran en el cuadro de diálogo Ajustes de color. Las diferencias entre estos cuatro modos de conversión a menudo son imperceptibles.

Resumiendo, el proceso simplificado sería el siguiente. El espacio de color relacionado con el perfil de entrada (por ejemplo, una cámara) se convierte a un espacio de color dado por un perfil de conexión intermediario (algo proporcionado, por ejemplo, por un programa de tratamiento de imágenes) y, de ahí, pasa a convertirse en un espacio de color relacionado con un perfil de salida (por ejemplo, una impresora). El espacio de color característico de un dispositivo tan corriente como una impresora es CMYK pues trabaja con mezcla substractiva. Sin embargo, la mayoría de las impresoras modernas aceptan RGB como modo de entrada.

Pero la historia no termina aquí pues, para una mejor comprensión del proceso, no nos basta con saber cuáles son los rangos absolutos, la gama cromática de un dispositivo, sino que necesitamos saber también cómo se distribuyen estos valores.

Rango dinámico. Factor gamma

El rango dinámico de un dispositivo es el número de valores que puede mostrar. Esto está directamente relacionado con la relación de contraste (*contrast-ratio*) que se define como la relación entre los valores máximos y mínimos de luminosidad que puede mostrar. Esta relación debe medirse en condiciones concretas por lo que no hay que fiarse de los valores dados por los vendedores de dispositivos que a menudo se refieren a valores nominales ideales que son imposibles de conseguir en condiciones corrientes.

La relación de contraste se mide corrientemente por referencia a valores utilizados en fotografía, concretamente a los f-stops o pasos de apertura del diafragma. Como da lo mismo abrir o cerrar el diafragma en un punto que disminuir o aumentar la velocidad en un punto, se utiliza un término más genérico, los valores EV (*Exposure Values*). El valor base, un EV = 0, corresponde a un tiempo de exposición de 1 segundo y una apertura relativa de f/1,0. Como en cada paso la exposición es la mitad de la anterior, la escala de valores EV se da en potencias de 2.

De este modo resulta sencillo calcular el rango dinámico de un dispositivo si conocemos cuantos pasos EV admite. En el caso de las cámaras, este cálculo es inmediato pues todo lo que necesitamos saber es cuantos pasos admite la ruedecilla de apertura de diafragma. Una cámara digital corriente puede contar con poco más de 7 u 8 EV (2^7, 2^8, es decir, 128, 256) mientras que una cámara analógica llega a los 11 o algo más y una cámara digital reflex puede llegar a los 10.

Resulta relativamente fácil hacer corresponder esta referencia con la de dispositivos corrientes. Así, el rango dinámico de las imágenes impresas es del orden de 200/1. El de las cámaras digitales, del orden de 300/1. Un monitor antiguo, CRT, tiene un rango dinámico de alrededor de 500/1 mientras que un LCD alcanza los 700/1 o más. Una buena cámara analógica podía llegar a los 2.000/1.

Ahora bien, el ojo humano tiene un rango dinámico estimado de unos 10.000/1 (algo menos de 14 pasos EV). Y en un exterior al Sol se pueden encontrar contrastes del orden de los 100.000/1 (entre 16 y 17 pasos EV). Esto quiere decir naturalmente que el ojo se adapta a estas situaciones y que percibirá tan solo una parte de estos valores.

Pero esta adaptación también tiene que hacerse al procesar imágenes para adaptarlas a los diferentes dispositivos. Las imágenes de 24 bits por píxel que se utilizan corrientemente tienen 8 bits por cada canal RGB, lo que significa que pueden asociar una determinada intensidad a un entero comprendido entre 0 y 255. Un color blanco, codificado como RGB 255, 255, 255 representará la

superficie más blanca que haya en la escena, sea una hoja de papel en un interior o una superficie blanca iluminada por el Sol en un exterior. Sin embargo, la medición de la iluminación que llegaría a estas superficies en un caso real podría variar fácilmente entre 100 y 10.000 luxes o más. Una vez que nuestro sistema visual se ha adaptado a un entorno más o menos luminoso (lo que puede durar más de 20 minutos), percibiríamos ambas superficies como blancas aunque su luminosidad real fuera muy distinta. Estos rangos se adaptan mal a las escalas utilizadas por un monitor.

La percepción de intensidades del ojo humano ha sido ampliamente estudiada y los razonamientos teóricos coinciden con los experimentales. Si tenemos dos valores extremos, blanco y negro, y se crea una escala de grises de modo que el ojo percibe intervalos regulares entre los diferentes escalones de esta escala, la distribución sigue una curva como la de la figura 2.13. Como puede observarse en esta curva, el gris medio no refleja el 50 % como pensaron los primeros teóricos del siglo XIX que trabajaron sobre estos temas (incluyendo a Chevreul, que hizo aportaciones muy notables a la teoría de los colores pero se equivocó al crear las escalas de sus sólidos de color) sino el 18 % aproximadamente.

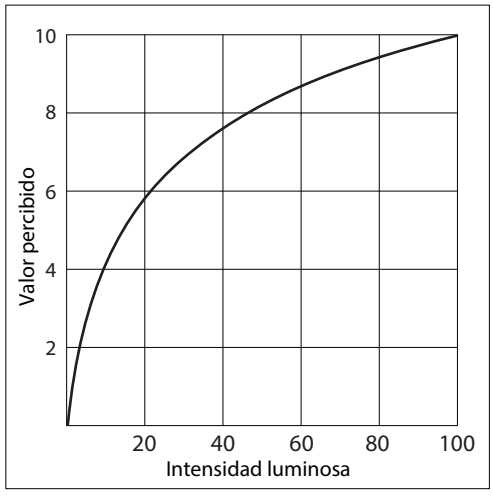


Figura 2.13 Relación entre intensidades emitidas e intensidades percibidas.

Esta relación puede expresarse numéricamente mediante una fórmula simple como la siguiente:

$$I = K^{\gamma}$$

que nos dice que la intensidad de salida se relaciona con la intensidad de voltaje mediante una constante elevada a un factor conocido como factor gamma (γ).

Los monitores corrientes de rayos catódicos tienen una relación de intensidad entre intensidad de salida y voltaje que corresponde a un factor γ que está entre 2,4 y 2,8 (en 2008). El sistema sRGB estándar parte de un valor algo más bajo, de 2,2, lo que tiene el efecto de que las imágenes corrientes adquieren algo más de contraste al representarse en un monitor corriente.

Pero, en cualquier caso, los monitores y las impresoras tienen un rango dinámico muy limitado. Por ello es necesario redistribuir los valores por medio de una curva de tonos que adapta los valores a los rangos dinámicos del dispositivo. Esta operación se denomina proyección de tonos (*tone mapping*) y se lleva a cabo por medio de parámetros especiales que están disponibles en todos los programas avanzados de simulación.

En un programa como 3ds Max esto se hace por medio de un panel específico que se denomina Control de exposición. Este panel permite escoger entre diferentes curvas de ajuste que corrigen la curva de proyección de tonos de una escena para adecuarla a los valores de iluminación que se han utilizado para el cálculo. Si, por ejemplo, tenemos unos materiales que están situados en una escena exterior y se han utilizado valores reales para el cálculo de la iluminación, pongamos que 80.000 luxes, estos valores deben ser redistribuidos, pues si no se hiciera así el resultado quedaría completamente distorsionado.

Las limitaciones de los formatos actuales de imágenes, desde el punto de vista de su comprimido rango dinámico, pueden superarse en parte gracias al formato HDR (*High*

Dynamic Range) al que ya me he referido anteriormente. Una imagen grabada en formato HDR guarda la información como valores en coma flotante. Esto quiere decir que podría guardar información tal que representara un color blanco de valor 255 por un valor tal como 10.000. Se mantiene por tanto la proporcionalidad real. Si, por el contrario, este rango se grabara en un formato corriente, que solo admite 256 valores dados por números enteros, los decimales se truncarían, lo que equivale a truncar la escala de luminosidades: como si una curva trazada con precisión se convierte en una serie de segmentos escalonados por falta de resolución. Me remito de nuevo al libro sobre simulación de la iluminación para ampliar este tema.

Gestión del color en la práctica

En la práctica real nos encontramos con situaciones características, con diversas variantes, pero que podemos resumir como sigue. Supongamos que hemos obtenido una serie de imágenes que vamos a utilizar para llevar a cabo la simulación de una escena, sea para utilizarlas como imágenes en fondos o partes de un objeto, sea para manipularlas desde un programa de tratamiento de imágenes, como Gimp o Photoshop, y convertirlas en texturas.

Esto quiere decir que las imágenes van a ser captadas por un dispositivo, una cámara (o un escáner, lo que implicaría otros problemas pero no demasiado diferentes), van a ser trabajadas desde un monitor, con un determinado programa, y van a ser llevadas, finalmente, a otro monitor o a una impresora.

Tenemos por tanto, en esta situación simplificada, un dispositivo principal de entrada, una cámara y dos dispositivos principales de salida, un monitor y una impresora.

Como he dicho antes, el análisis de los rangos dinámicos de dispositivos se desarrolla con cierta extensión en el libro sobre simulación de la iluminación mencionado en la introducción. Aquí me limitaré a resumir lo fundamental por lo que respecta a los principales dispositivos de salida que interesan en la práctica: los monitores y las impresoras.

En cada uno de estos estadios necesitamos contar con la suficiente garantía de que los colores que estamos utilizando se corresponden con los colores “reales”, es decir, con colores tal como los percibiríamos en condiciones óptimas. Pasemos por alto todas las complejidades y ambiguedades implicadas en esta descripción y que ya se han apuntado en las secciones anteriores. Baste con decir que tenemos, en cada estadio, dos posibilidades: controlar los colores mediante métodos caros y profesionales o controlarlos mediante métodos aproximados y menos costosos.

Comencemos por la *cámara*. Tenemos dos alternativas: la barata y sencilla y no demasiado fiable y la menos barata, menos sencilla pero más fiable. La primera consiste en tomar fotografías de alguna escena o imagen de referencia con colores destacados, comprobar el resultado y, si este no es correcto, estudiar a fondo el capítulo de balance de blancos del manual de la cámara para asegurarnos de que no estamos distorsionando excesivamente los colores reales. Y a partir de ahí confiar en que estos datos serán leidos adecuadamente por los programas que utilicemos. La segunda alternativa comienza por adquirir una carta de colores normalizada. Una opción relativamente sencilla y no demasiado cara (en torno a 100 € en 2014) es un *Color Checker* de los que distribuyen algunas empresas conocidas, como x-Rite. Lo que hay que hacer a continuación es sacar una fotografía de la escena incluyendo esta carta en un lugar claramente visible y luego retirarla. Esta primera toma nos servirá para crear un perfil ICC de la cámara pues los colores que hemos fotografiado están codificados mediante códigos ICC y, por tanto, cualquier programa podrá medir la desviación de los colores fotografiados con los reales e introducir las correcciones correspondientes. No voy a describir el proceso con detalle por razones de espacio y porque ya se describe con la guía del producto, pero es relativamente sencillo. Básicamente consta de los pasos siguientes: *a*)

Tomar una fotografía de la escena (que puede ser una escena genérica, con iluminación diurna a pleno Sol, diurna en sombra, con luz artificial, de modo que podamos reutilizarla en casos similares) con la carta claramente visible e iluminada de un modo uniforme. La fotografía debe tomarse en el formato *raw* propio de la cámara y luego guardarse como *Digital Negative* (DNG); *b*) Generar un perfil para la cámara que estemos utilizando con la ayuda de la aplicación incluida con la carta; *c*) Abrir la imagen en Camera Raw o Lightroom (dos programas ligados a Photoshop, el primero incluido en el programa corriente y el segundo con un coste adicional), y en la sección Calibración de la cámara, seleccionar el perfil que hemos creado y que se habrá añadido a la lista de perfiles incluidos en Camera Raw. Con esto la imagen se modificará ligeramente para adecuarse al perfil y, a partir de ahí, si es necesario, podemos utilizar las herramientas del programa para ajustar por añadidura el balance de blancos o los tonos de la imagen.

Figura 2.14 Color Checker.

La figura 2.14 muestra una escena sencilla ajustada con este método.

Continuemos por el *monitor*. De nuevo tenemos dos alternativas: la barata y aproximada y la cara y profesional. En cualquiera de los dos casos necesitamos asegurarnos de que trabajamos en condiciones adecuadas: sin cambios de iluminación, sin reflejos sobre la pantalla, con luces suaves que no incidan directamente sobre la pantalla. Para la primera podemos utilizar el sistema de calibración simple incorporado a los sistemas operativos de los PC o Max y que se basan en la apreciación subjetiva del usuario, es decir, como antes, fiarnos de nuestra buena vista. Para la segunda, la más recomendable si el coste no es demasiado problema, necesitamos adquirir también un sistema de calibración profesional (o un monitor caro que lo incorpore). Una opción recomendable es un calibrador Spyder que puede adquirirse, como el Color Checker, en cualquier tienda de fotografía profesional con un coste de unos 150 € o quizás algo más. El calibrador consiste en un aparato que se fija, como una araña, sobre el monitor, toma una serie de medidas automáticas, ajusta el monitor y crea un perfil ICC que se guarda en el propio ordenador. Este proceso debe repetirse periódicamente.

Terminemos con la *impresora*. También aquí tenemos las dos opciones anteriores: hacer pruebas personales con una determinada impresora o mandarlas a un servicio profesional. Pero la primera opción en este caso es más complicada: una vez que hemos comprobado las desviaciones que se producen al imprimir una imagen tendríamos que introducir curvas de corrección, ajustarlas cuidadosamente, guardarlas y revisarlas periódicamente, algo que no muchos usuarios estarán dispuestos a hacer. Es preferible, en este caso, guardar la imagen con el perfil ICC incorporado y utilizar un servicio que sepa de qué le estamos hablando cuando le preguntemos si su impresora cuenta con un sistema adecuado para leer estos perfiles y efectuar los ajustes correspondientes.

2.3 Simulación de la reflexión y refracción

Cantidades radiométricas básicas. Radiancia e irradiancia

En la actualidad, la descripción del modo como la luz incide y se refleja en la superficie de un material, se hace en el marco de la teoría física del análisis de la radiación térmica que incluye la radiación visible. Esto implica una serie de términos que son comunes a la radiometría y la fotometría y que conviene recordar brevemente: el flujo radiante, la intensidad, la irradiancia y la radiancia.

Si tenemos una fuente de energía, la primera medida que nos importa es el *flujo radiante*, que se mide en vatios y que describe la cantidad de esta energía emitida por unidad de tiempo (un vatio es igual a 1 julio por segundo). Una lámpara de 100 vatios emite esta energía en todas direcciones mientras esté encendida: pero la cantidad de esta energía que percibimos como luz depende del rendimiento luminoso de la bombilla (si el rendimiento está, como suele ser corriente para lámparas incandescentes sencillas, en torno a los 15 o 20 lúmenes por vatio diremos que el flujo luminoso es de 1.500 a 2.000 lúmenes).

Pero es evidente que este flujo no se distribuirá igual en la dirección del casquillo al que está sujeta la lámpara (que obstaculiza el flujo) que en la dirección opuesta (que es óptima) ni en la dirección perpendicular, hacia los lados. Si nos olvidamos de la bombilla y suponemos que, en general, el centro de emisión está rodeado de una esfera imaginaria de 1 metro de radio, lo que nos interesa es la *intensidad* en una dirección. Si la distribución es homogénea, nos bastará con dividir el flujo total por 4π, que es la superficie de esta esfera imaginaria ($4\pi r^2$ con radio 1). En general, tendremos que dividirla por el ángulo sólido considerado, que se mide en estereorradianes, sr (la superficie de una esfera tiene por tanto 4π sr). Si en lugar de considerar toda la esfera consideramos, en el análisis de una superficie, un hemisferio, tendremos un total de 2π sr.

Si consideramos una fuente de luz y queremos analizar cómo incide sobre un punto de una superficie bastará con que consideremos la proyección del área de esta fuente de luz sobre un hemisferio virtual de radio unidad pues el resultado será equivalente y podemos unificar los resultados. Los dos términos clave que debemos considerar para lo que sigue son la irradiancia y la radiancia. Son términos similares por lo que respecta al tipo de unidades consideradas, pero en un caso se trata de la energía inicial y en el segundo, de la energía derivada cuyo valor queremos obtener.

La energía que cae sobre una superficie se denomina *irradiancia* (E) y se mide en vatios/m^2. Es el equivalente a la densidad de flujo, y para el análisis de la iluminación de un punto sobre una superficie no varía pues partimos precisamente del flujo considerado según un determinado ángulo de incidencia.

La energía emitida desde una superficie se denomina *radiancia* (*L*) y se mide en vatios/m^2/sr. A diferencia de la anterior, disminuye al aumentar el ángulo de reflexión. Si una superficie recibe energía y la devuelve, la emisión será máxima en la dirección perpendicular a la superficie (θ_r = 90º) y nula a partir de la dirección coincidente con la superficie (θ_r = 0º). Dicho de otro modo, variará con el coseno del ángulo de reflexión (pues cos(90) = 1 y cos(90) = 0).

Todos estos términos se relacionan directamente con los cuatro términos equivalentes utilizados en fotometría: *flujo* luminoso (que se mide en lúmenes), *intensidad* luminosa (que se mide en candelas), *iluminación* (que se mide en luxes) y *luminancia* (que se mide en candelas por metro cuadrado, cd/m^2). Para pasar de unidades radiantes a unidades fotométricas necesitamos saber la eficiencia luminosa de la luz que estemos utilizando. Dado que en simulación de materiales lo que nos importa principalmente es la relación entre radiación recibida y la emitida, es corriente obviar esta traducción de valores radiantes a valores luminosos.

La reflexión de la luz en un punto de una superficie dependerá por consiguiente de dos factores, cada uno de los cuales, a su vez, implica otros dos valores, la dirección y la intensidad. Como los dos factores iniciales dependen de la dirección, la reflexión se expresa como una relación entre estos dos factores y se expresa como una relación entre ambas

$$\rho \rightarrow L_r / E_i$$

A esta relación se le conoce con el nombre de *reflectividad bidireccional*. Su cálculo implica una función de cuatro variables (dos por cada uno de los dos términos). Como veremos más adelante, a lo largo de los últimos 30 o 40 años el progreso en simulación de materiales ha estado estrechamente ligado a los diferentes métodos de calcular esta función, la función BRDF (*Bi-directional Reflectance Distribution Function*).

En general no solo hay reflejo sino que también transparencia. La transparencia depende del espesor y de las características de los materiales. La intensidad de una onda al entrar en un medio disminuye en un factor variable cuando la onda ha recorrido una determinada profundidad de penetración o profundidad pelicular del material. Un material es transparente a una radiación de una determinada frecuencia si esta profundidad es mayor que su espesor. Este factor depende de la conductividad. Los materiales metálicos tienen una conductividad no nula y en consecuencia su profundidad de penetración es pequeña. El cobre, por ejemplo, en la zona de frecuencias ultravioletas, en torno a los 100 nm tiene una profundidad de penetración de 0,6 nm y en la de los infrarrojos, en torno a los 10.000 nm, de 6 nm. Otros metales tienen valores similares. Y esto se corresponde con el hecho de que aunque los materiales metálicos sean opacos, si se cortan en láminas muy delgadas se hacen transparentes.

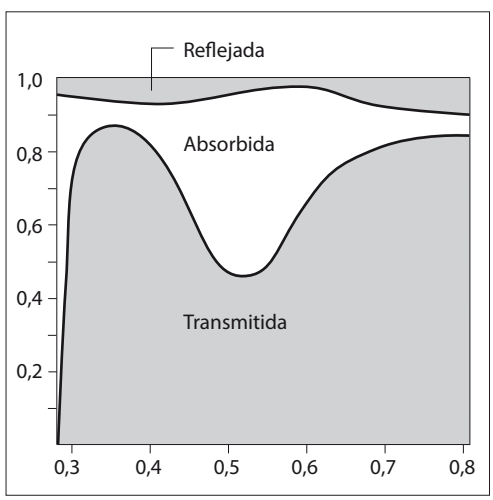


Figura 2.15 Balance de la energía luminosa de un vidrio de color verdoso.

Como ya he dicho, cuando una onda de intensidad I_i penetra en un sólido transparente, parte se refleja, con una intensidad I_r, parte se absorbe, con una intensidad I_a, y parte se transmite, con una intensidad I_t. La conservación de la energía requiere que se cumpla que $I_r + I_a + I_t = I_i$. Si dividimos todo por I_i resultará que $R + A + T = 1$, siendo R el coeficiente de reflexión o reflectancia, A el coeficiente de absorción o absorbancia y T el coeficiente de transmisión o transmitancia. Si se lleva a cabo un análisis de este tipo para todas las frecuencias que inciden sobre un determinado sólido, se obtendrá una figura como la 2.15, que muestra la distribución de estos tres factores para un vidrio de color verdoso. Si la absorción fuera la misma en todas las frecuencias el color sería transparente.

Una formulación más precisa del modo en que la luz se modifica al atravesar un sólido viene dada por la ley de Beer-Lambert (denominada así porque fue enunciada por Johann Lambert en 1760 y por August Beer en 1852. De hecho, ya había sido anticipada por otro físico, Pierre Bouguer en 1729). Esta fórmula la relaciona la absorbancia con la intensidad incidente y transmitida, en función del coeficiente de absorción, el espesor del material y un factor variable según las versiones que ajusta la fórmula en función de la concentración del coeficiente de absorción o de un factor de extinción más genérico.

Albedo. Reflectividad. Reflectancia

Los términos que encabezan este apartado aparecen con frecuencia en la literatura especializada pero tienden a confundirse por lo que será conveniente precisarlos antes de continuar.

El término *albedo* viene de la astronomía y se utiliza para caracterizar a los planetas y los elementos de la propia superficie de la Tierra. Se denomina albedo a la relación entre la luz reflejada y la luz incidente. Aunque el rango ideal es de 1,0 a 0,0, el albedo de los materiales corrientes varía entre 0,9 (nieve) y 0,04 (carbón). El albedo medio de la Tierra es de 0,3. En simulación de materiales se utiliza a veces para referirse al comportamiento global de la superficie de un objeto, a su coeficiente medio de reflexión, principalmente en simulación de exteriores.

El término *reflectividad* viene de la física y se utiliza para caracterizar la capacidad de una superficie para reflejar la radiación que recibe de una determinada longitud de onda. Es, por tanto, una denominación genérica que debe ser analizada para cada caso y que se subdivide en varios tipos. Pues la reflectividad puede ser, como veremos, especular, difusa o mixta y depende de diversos factores.

El término *reflectancia* se utiliza para designar la cantidad específica que mide la reflectividad o la reflexión de una superficie en determinadas circunstancias. Se define como la relación entre el flujo (o energía) radiante reflejado y el flujo (o energia) radiante incidente. Se analizará con detalle en los apartados siguientes.

La función BRDF

La literatura especializada actual engloba todos los modelos que han ido surgiendo en los últimos 35 años bajo una descripción más general, la *función de distribución de la reflectancia bidireccional*, BRDF por sus siglas en inglés (*bidirectional reflectance distribution function*), que representa la cantidad relativa de energía que se refleja en una dirección, a partir de otra dirección, para una determinada longitud de onda. El término bidireccional se refiere a los dos ángulos involucrados. También puede entenderse como la probabilidad de que un fotón se emita en una determinada dirección cuando llega de una dirección dada.

La función BRDF se definió por primera vez en 1970 por Nicodemus (Nicodemus, 1970 y Nicodemus *et al*. 1977) y actualmente está incorporada a todos los métodos actuales de iluminación global, por lo que es importante entender sus características principales. Cada superficie de una escena virtual incorpora una función de este tipo que informa al sistema sobre cómo deben redistribuirse los rayos que llegan a un punto de dicha superficie. La mayoría de los sistemas actuales incluyen, como veremos más adelante, alguna variante de métodos de Montecarlo por la que se escogen muestras significativas de una determinada función probabilística.

En general se expresa como la relación diferencial entre dos cantidades: L_r, que representaría la radiancia reflejada, y E_r, que representaría la irradiancia, el flujo incidente por unidad de área por unidad de ángulo sólido. En el apartado anterior hemos visto los términos principales de esta relación, la irradiancia (E) y la radiancia (L).

La BRDF es una función que tiene cuatro variables. Puede expresarse de diferentes modos equivalentes (para una determinada longitud de onda):

$$f_r(w_i, w_r) = f_r(\theta_i, \phi_i, \theta_r, \phi_r) = dL_r(w_r)/\, dE_i(w_i)$$

en donde la función f_r tiene como variables w_i, el vector que representa la irradiancia incidente en la superficie desde una determinada dirección, y w_r, el vector que representa la radiancia reflejada que sale de un punto de una superficie en una determinada dirección (es decir que $w_i = (\theta_i, \phi_i)$ y $w_r = (\theta_r, \phi_r)$). O bien (segundo término) puede definirse a partir de los ángulos incidente y reflejado, θ_i, θ_r y de los flujos indicente y reflejado, ϕ_i, ϕ_r. O bien (tercer término), a partir de diferencial de la cantidad de luz reflejada, $L(w_r)$ y la cantidad de luz incidente $E(w_i)$. Todas estas formas pueden

encontrarse en la literatura sobre este tema.

La función BRDF cumple varias propiedades importantes.

La primera es que si se intercambian los valores de w_r y w_i se llega al mismo resultado. Esto es lo que se denomina la reciprocidad de Helmholtz, al haber sido observado por Helmholtz a mediados del siglo XIX (y es la base de una serie de técnicas actuales de reconocimiento de objetos que se denominan globalmente Helmholtz Stereopsis).

Una segunda propiedad importante de esta función es que, por la ley de conservación de la energía, la suma normalizada de las intensidades incidentes y reflejadas debe ser ≤ 1.

No debe perderse de vista que esta formulación general es válida para una determinada longitud de onda, que prescinde de la polarización y que considera una superficie isotrópica, es decir, que mantiene sus propiedades de reflectancia para cualquier dirección. Como veremos después esto excluye muchas superficies anisotrópicas importantes para las que habrá que introducir parámetros adicionales.

Todos los modelos de simulación visual de materiales utilizan alguna variante simplificada de función BRDF. En función del tipo de simplificación propuesta podemos clasificar estos modelos en tres categorías principales:

a) Los modelos analíticos que podemos denominar *empíricos* y que se basan en la introducción de parámetros sin significado físico preciso, pero que pueden ser ajustados para obtener resultados adecuados. A partir del momento en que ha sido posible contar con mediciones (véase *c*) también ha sido posible conseguir que el ajuste sea más preciso por medio de comparaciones entre los resultados del modelo computacional y el modelo real. La principal ventaja de este tipo de mo-

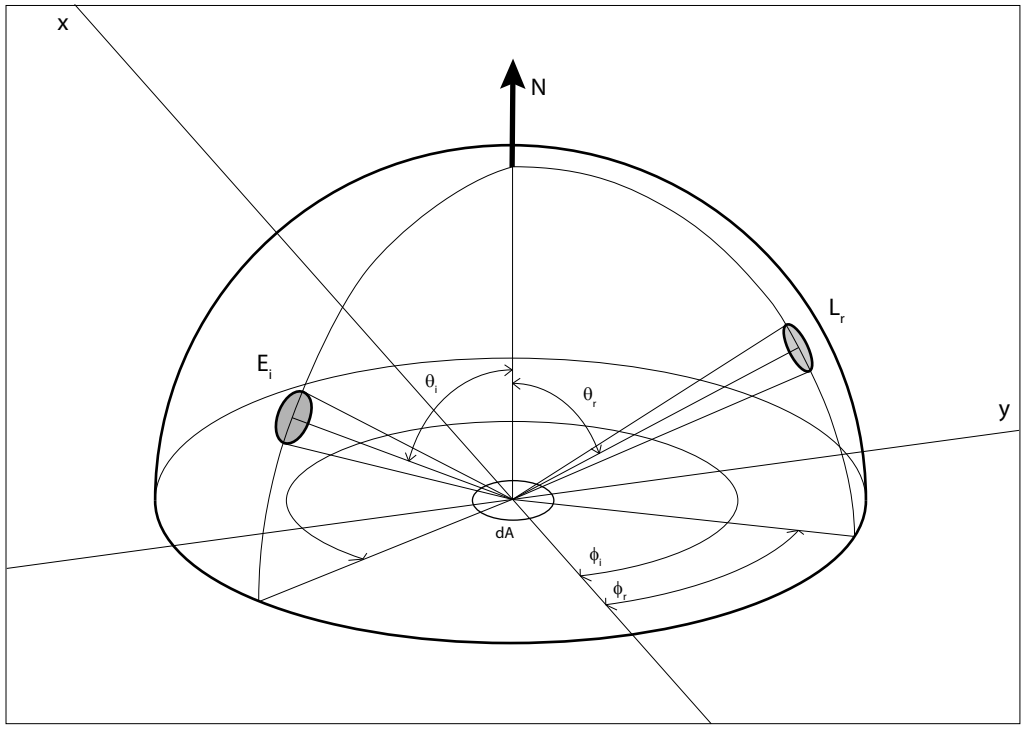


Figura 2.16 Términos fundamentales de la función BRDF.

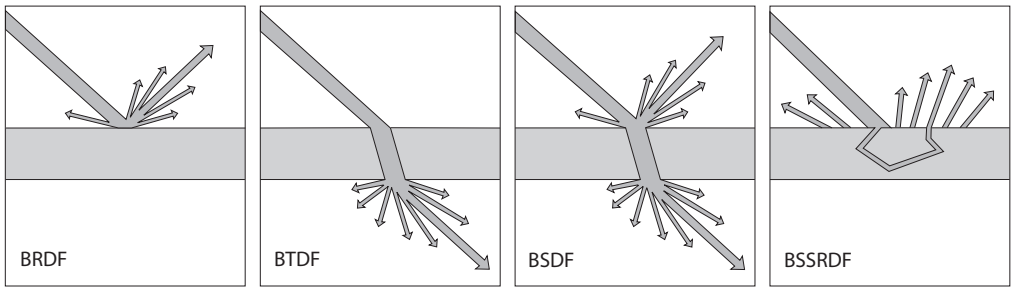


Figura 2.17 Las funciones BRDF, BTDF, BSDF y BSSRDF.

delos es que son más sencillos de calcular y resultan más eficaces en la práctica.

b) Los modelos analíticos que podemos denominar *físicos* y que se basan en la introducción de parámetros con significado preciso basados en alguna versión más o menos simplificada de la teoría física que proporciona ecuaciones coherentes. En la práctica, se eliminan algunos parámetros o se introducen algunas constantes adecuadas para hacer el modelo más manejable. El principal inconveniente, sin embargo, es que son más complicados de calcular y resultan menos eficaces en la práctica.

c) Los basados en mediciones de materiales reales y que almacenan los resultados en tablas de valores. Estos modelos se basan en la utilización de aparatos especiales que permiten tomar medidas en diferentes direcciones y almacenar los resultados en tablas. Tienen una gran valor tanto como referencia para comprobar la eficacia de los modelos anteriores como porque proporcionan una vía alternativa que puede tener una creciente importancia en el futuro. Hay que destacar también que, pese a que los resultados de los modelos teóricos deberían aproximarse más a las mediciones, no siempre es así y puede resultar más fácil ajustar los parámetros de los modelos empíricos para que se aproximen a los resultados experimentales.

En las secciones que siguen describiré en primer lugar los dos componentes básicos, especular y difuso, que se combinan de diferentes modos en todos los modelos que seguirán, por orden de aparición histórica. En todos estos modelos se prescinde de la polarización y de la luminiscencia y, salvo indicación expresa, de las variaciones debidas al ángulo de incidencia (factor Fresnel).

Pero antes de comenzar este resumen de modelos hay que mencionar otras funciones importantes desde el punto de vista teórico aunque sean menos utilizadas en la práctica.

Las funciones BTDF, BSDF, BSSRDF

La función BRDF da una fórmula que permite computar la distribución de la reflectancia para un determinado tipo de material. Como ya he dicho antes, para simplificar la exposición solo he considerado el caso más general, y también más corriente, en que el material no es transparente.

Si el material es transparente hay que tener en cuenta, no un hemisferio, sino toda la esfera que rodea al punto y el rayo reflejado se convierte en el rayo transmitido. En estos casos se utilizaría la *función de distribución de la transmitancia bidireccional*, BTDF por sus siglas en inglés (*Bidirectional Transmittance Distribution Function*). Y la ley principal que se aplica en estos casos es la ley de Snell. La única diferencia es que los ángulos considerados están en diferentes hemisferios.

En algunos casos se utiliza una función combinada para la reflexión y la transparencia, denominada, un tanto equívocamente, *función de distribución de la dispersión bidi-*

reccional, BSDF por sus siglas en inglés (*Bidirectional Scattering Distribution Function*) que toma en consideración la esfera que rodea a un punto y combina la BRDF y la BTDF en una única ecuación. La unica diferencia es que se consideran los valores absolutos de los senos en la formulación de la función o se reorientan las normales en la formulación.

Por último, hay que citar otra variante que se utiliza en contextos específicos: la *función de distribución de la reflectancia de la dispersión superficial,* BSSRDF por sus siglas en inglés (*Bidirectional Surface Scattering Reflectance Distribution Function*) que computa el modo en que la luz se dispersa en determinados medios (líquidos oleosos, leche, piel de poco espesor, ciertos tipos de mármol, etc.). Esta función fue introducida por Henrik Wann Jensen en 2001, en Sigradi. La intención de Jensen era desarrollar un *shader* físico más exacto que permitiera simular de un modo preciso la dispersión interna, que es algo que se da en la mayoría de los materiales con la excepción principal de los metales. Resultó que esto tenía ventajas técnicas adicionales pues los *shaders* que utilizaban la función BSSRDF se representaban con mayor rapidez sin necesidad de recursos técnicos artificiosos.

En la actualidad se utiliza principalmente para simular la piel humana por lo que, en principio, queda fuera del campo de aplicación a que va dirigido este libro. Y también queda fuera del campo de aplicación de muchos programas de simulación que no incluyen *shaders* de este tipo. Pero no debe perderse de vista que muchos materiales, como el mármol, el alabastro o el jade, además de muchos líquidos, presentan características visuales que se basan en este mismo tipo de fenómenos. Por esta razón se incluirán algunos ejemplos de aplicación. En este apartado me limitaré a una breve introducción teórica que complementa el marco de referencia general.

Tal como se muestra en la figura 2.17, la función BSSRDF se basa en que, en muchos casos como los citados, la luz incidente no se refleja directamente desde la capa externa de la superficie sino que la atraviesa, se refleja y se dispersa internamente (*in-scattering*), y parte de estos reflejos son absorbidos mientras que otra parte vuelven a salir al exterior, es decir, que se dispersan externamente (*out-scattering*).

Para computar adecuadamente la forma de dispersión se necesitan conocer datos empíricos sobre diferentes tipos de materiales. Puede encontrarse información adicional en las siguientes referencias (véase el capítulo correspondiente): Jensen; Jensen, Marschner, Levoy, Hanrahan; Jensen y Donner; Koutajoki (2002).

En el capítulo sobre técnicas se proporcionarán ejemplos adicionales sobre los *shaders* con que se puede contar en la actualidad para este tipo de simulaciones.

2.4 Modelos principales para la función BRDF

Los apartados que siguen resumen los modelos principales, por orden de aparición histórica y con una terminología y simbología unificadas para facilitar el seguimiento de esta recopilación.

Hay otros modelos importantes que no incluyo pues solo pretendo describir el marco teórico general que conviene conocer, aunque tan solo sea superficialmente, para los objetivos de este libro. Entre ellos, los de Kajiya (1985), que fue quizás el primero en introducir variantes anisotrópicas, el de Strauss (1990), que introdujo variantes empíricas importantes para simular metales y que simplificaban el modelo de Cook-Torrance para estos casos. O el de Banks (1994), un modelo empírico que suponía una buena aproximación práctica mediante modificaciones del modelo de Phong que introducían variantes anisotrópicas. En la bibliografía se dan referencias sobre estos y otros, para que el lector interesado amplíe esta información.

Especularidad perfecta

Si simplificamos al máximo la función BRDF asumiendo una especularidad perfecta, resultará que toda la luz que incide por un lado se refleja por el otro. Por consiguiente, la función será 0 en todas direcciones excepto en la dirección de reflexión, es decir, en la dirección en que los dos ángulos son iguales. Tendremos entonces que:

$$f_r = \rho \text{ si } \theta_r = \theta_i;\ \phi_r = \phi_i + \pi$$
$$f_r = 0 \text{ en caso contrario}$$

Difusión perfecta (superficie de Lambert)

Si volvemos a simplificar al máximo pero asumiendo esta vez una difusión perfecta, tendremos que, para una superficie perfectamente difusa (denominada superficie de Lambert o Lambertiana, por Lambert, 1728-1777, que fue el primero en describir las leyes de la difusión perfecta) la función es constante y por tanto:

$$f_r = \rho / \pi$$

pues la función cubre 180º (un hemisferio) que en radianes es π.

Dado que la función es constante, el brillo de la superficie solo dependerá de la dirección de la luz incidente y de su intensidad y, por consiguiente es independiente del punto de vista, a diferencia de la reflexión especular y de los modelos que siguen. La luz reflejada varía con el coseno del ángulo de incidencia: $I_R = I_L \rho cos\theta$. En notación vectorial y asumiendo que L es un vector normalizado que representa la dirección de la luz y N un vector normalizado que representa la dirección de la normal, esto puede reescribirse:

$$I_R = I_L \rho (N.L)$$

que es otra notación habitual para trabajar con superficies Lambertianas.

La difusión perfecta puede considerarse como un caso especial de *sub-surface scattering* donde la luz entra y vuelve a salir en el mismo punto, debido a la dispersión, a una escala inapreciable.

Phong (1975)

El modelo de Phong (desarrollado por Bui Tuong Phong, un investigador vietnamita que desarrolló el modelo que lleva su nombre en Utah y que murió de leucemia el mismo año que lo publicó, a los 33 años), es una formulación empírica que permite asignar un valor arbitrario a la intensidad y el diámetro del resalte especular. La figura 2.20 (izquierda) muestra los elementos básicos de este modelo que durante muchos años ha sido el más utilizado. Si L es la dirección de incidencia de la luz sobre una superficie, en un punto cuya normal es N, con un ángulo θ, la dirección de especularidad perfecta vendría

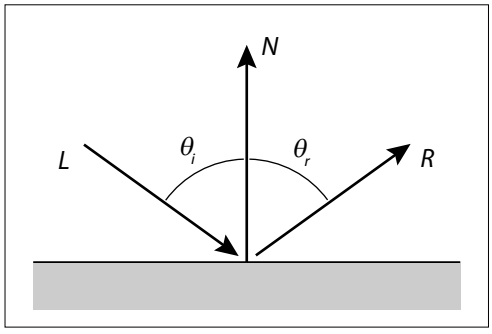


Figura 2.18 Especularidad perfecta.

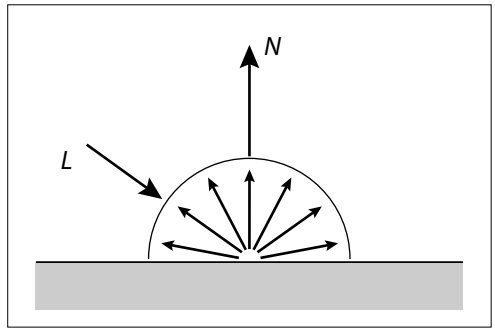


Figura 2.19 Difusión perfecta.

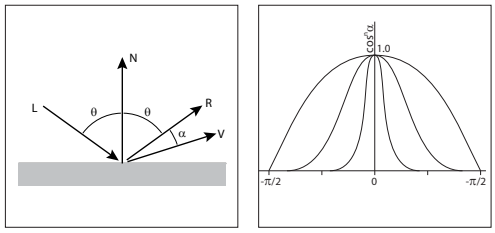


Figura 2.20 Modelo de Phong.

dada por *R*, que forma el mismo ángulo θ con respecto a la normal. Si la superficie fuera un objeto perfecto y la luz fuera diminuta, su reflejo solo sería visible si la dirección de la visual, *V*, coincidiera con *R* o estuviera muy próxima a él. Pero en casos reales, con superficies reflectantes pero no especulares, el reflejo sería percibido dentro de un determinado ángulo α.

Cuanto más brillante fuese la superficie más pequeña sería la zona en que el reflejo sería perceptible y viceversa. Phong decidió hacer que la intensidad reflejada fuera una función de (cos α)*n*, con $n \geq 200$ para superficies muy brillantes (y tendiendo a infinito para superficies muy especulares) y *n* mucho más pequeño para superficies mates (tendiendo a 0 para superficies completamente mates). La figura 2.20 (derecha) muestra como varía este valor con el ángulo y con el valor de *n*.

La formulación inicial era la siguiente:

$$I = I_a k_a + dI_d + sI_s$$

donde *I* era la intensidad resultante percibida, $I_a k_a$ era la contribución ambiental (un factor corriente en esa época para simular la luz reflejada por los objetos circundantes que no se podría simular adecuadamente hasta que no aparecieran los métodos de iluminación avanzada, unos 10 años después), dI_d la contribución difusa que se obtenía mediante el factor $d = k_d$.max (0, *N*.*L*) y sI_s, el último término, que es el que nos interesa, la contribución especular que venía dado por el factor $s = k_s(V.R)^n$, siendo k_s el factor de reflexividad de la superficie que podemos hacer equivalente al valor *r* de las fórmulas anteriores, y *V*, *R* los vectores correspondientes a la dirección de visión y de reflexión y *n* el exponente mencionado. El producto de los dos vectores es, por tanto, función directa del cos α pues $V.R = |V|.|R|.\cos\alpha$. En términos similares a los anteriores, por consiguiente, la función BRDF puede considerarse dada por el factor principal $(V.R)^n$, si bien la comparación con otros modelos posteriores resultaría equívoca debido a que, entre otras cosas, no se normalizan los vectores y los coeficientes k_d y k_s pueden escogerse con bastante libertad.

Lafortune *et al*. (véase mas adelante) reformulan el modelo de Phong para adaptarlo a las formalizaciones corrientes, y como introducción a su propia extensión del modelo del modo siguiente:

$$f_r(w_i, w_r) = \rho_s C_s \cos^n\alpha$$

donde C_s sería un factor de normalización dado por $(n + 2) / 2\pi$. Esta reformulación también puede escribirse vectorialmente utilizando el factor principal que hemos dado anteriormente, así:

$$f_r(w_i, w_r) = \rho_s C_s (V.R)^n$$

Las implementaciones posteriores del modelo de Phong, a medida que fueron apareciendo modelos más coherentes desde el punto de vista físico y a medida que se empezaron a utilizar sistemas de iluminación avanzada, han utilizado preferentemente esta fórmula. El lector interesado en profundizar en estos temas puede ver los artículos de Lewis (1993), para un análisis más preciso de las carencias de este y otros modelos desde el punto de vista físico, el modelo de Lafortune (1997), que se resume más adelante y que desarrolla el modelo de Phong, o el artículo de Lawrence (2002), para un análisis de la adecuación del modelo de Phong a sistemas de cálculo de iluminación basados en métodos de Montecarlo.

Blinn (1977)

El modelo propuesto por Blinn dos años después es una variante del de Phong, que utiliza

el vector *H*, bisector de *L* y *V* (véase la figura 2.21) y que se calcularía mediante la simple ecuación

$$H = (L + V) / |L + V|$$

Esto hace que el cálculo sea más rápido cuando las luces y el observador están en el infinito pues con la formulación de Phong hay que recalcular el ángulo *R.V* continuamente mientras que en estos casos *H* sería constante.

En la formulación de Blinn se puede substituir el término especular *R.V* por el nuevo término *N.H*. El vector *H* es también la dirección de máxima reflexión especular pues si *H* coincidiera con *N*, entonces *R* coincidiría con *V*. Hay que tener en cuenta que el factor empírico *n* se aplica a un ángulo que no es exactamente el mismo, por lo que los resultados son ligeramente diferentes (de hecho, son algo más precisos con la formulación de Blinn). En cualquier caso el término especular sería:

$$(N.H)^n$$

Por otra parte, en el mismo artículo, Blinn plantea la alternativa de desarrollar un modelo más adecuado a los materiales reales a partir de una distribución de microfacetas basado en el modelo propuesto por Torrance y Sparrow en 1967. El tipo de distribución propuesta por Blinn era menos eficiente que el propuesto pocos años después y que se conoce como el modelo de Cook y Torrance.

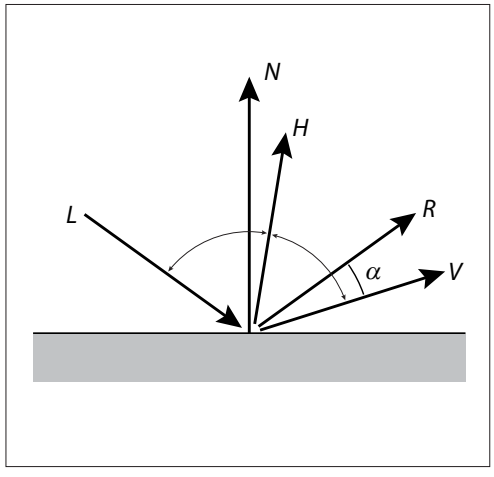


Figura 2.21 Modelo de Blinn.

Cook y Torrance (1982)

Este modelo deriva de modelos utilizados en la comunidad de físicos (una formulación inicial se publicó en artículos de Torrance y Sparrow en 1966 y 1967, en el *Journal of the Optical Society of America*). A diferencia del de Phong, que busca un resultado convicente y adapta los parámetros al resultado buscado, este modelo buscaba una simulación real basada en una modelización de la superficie como una serie de microfacetas, que luego sería repetida con diversas variantes. Cada faceta se considera como un reflector isotrópico perfecto.

Se desarrolló por Robert Cook, que trabajaba en aquella época en Lucasfilm, y Ken Torrance, que trabajaba en la Universidad de Cornell. Lo presentaron conjuntamente en un famoso artículo publicado en 1982 (véase Cook, R; Torrance, K., 1982). La idea principal es simular la rugosidad de la superficie por medio de una serie de microfacetas regulares, simétricas, en forma de V y de tamaños dados por una determinada función de distribución.

Las facetas se consideran como reflectores perfectos, de tamaños cercanos a los de las longitudes de onda de la luz, con una distribución estadística que simplifica los cálculo, y se considera tan solo el efecto de oclusión entre las facetas.

La función bidireccional contiene cinco variables

$$f_r(w_i, w_r) = \frac{D(w_h)G(w_i, w_r)F_r(w_r)}{4\cos\theta_i \cos\theta_r}$$

donde:

– *D* es una función de distribución estadística de las microfacetas. Hay varios modelos posibles. Cook y Torrance utilizaron la distribución de Beckmann: $D(N.H) = (1 / 4$

m^2 $(N.H)^4$) e exp $(-(1-(N.H)^2)/(N.H)^2.m^2$. En esta fórmula, *m* es una constante [0,1] que controla la suavidad de la superficie.

– *G* es un factor geométrico de atenuación que representa la oclusión de parte de las facetas con respecto a la luz incidente o con respecto al punto de vista. Esto se representa por $G(N.w_i, N.H, N.w_o)$ = min (1, $(2(N.H)\ (N.w_o)\ /\ w.H)$, $(2(N.H)\ (N.w_i)\ /\ w.H)$, donde *H* es el ángulo mitad entre w_i y w_o por lo que $(H.w_i) = (H.w_o)$.
– *F* es el factor Fresnel para una determinada longitud de onda, λ y que depende del ángulo de incidencia de la luz, θ_i.
– El valor de $\cos(\theta_i)$ da cuenta de la cantidad de superficie (macroscópica) que es visible para la luz. Este factor también puede venir dado, en otras formulaciones, por *N.L*.
– El valor de $\cos(\theta_r)$ da cuenta de la cantidad de superficie (macroscópica) que es visible para el punto de vista. Este factor también puede venir dado, en otras formulaciones por *N.V*.

Las diferencias principales con respecto a otros modelos están, por tanto, en primer lugar, en el cómputo de la variable *D*. Ya he mencionado que Blinn también se basó en el modelo de Torrance y Sparrow pero con una distribución gaussiana. Cook y Torrance la substituyeron por otra más eficaz (elaborada por P. Beckman en 1963).

Y, en segundo lugar, en que, en este modelo, se incorpora una consecuencia importante de las fórmulas de Fresnel: que el color y la intensidad de la reflexión se modifican al variar el ángulo de incidencia de la luz. El desarrollo de las fórmulas de Fresnel muestra que si la luz incide perpendicularmente a la superficie, el factor *F* y, en consecuencia, la reflectancia, dependen del índice de refracción del material. Pero cuando el ángulo de incidencia se aproxima a 90º, es decir, cuando la luz es rasante con respecto a la superficie, el factor *F* y la reflectancia se aproximan a 1 y dejan de depender del índice de refracción. Esto es lo que ocurre con los metales, en los que para ángulos extremos la especularidad es máxima y no depende del color del material.

En el modelo de Phong no se da esta diferencia. El modelo de Phong es adecuado para representar un material pintado con pintura plástica pues, en la realidad, las partículas de este tipo de acabado están embebidas en un medio transparente que dispersan la luz de un modo que anula la influencia del ángulo recogida en las fórmulas de Fresnel.

Las fórmulas de Fresnel permiten calcular el índice de refracción para diferentes longitudes de onda, pero el cálculo es complicado y retarda el proceso. Por esta razón se utiliza un valor medio de *F* derivado de un valor medio del índice de refracción. Con este valor se

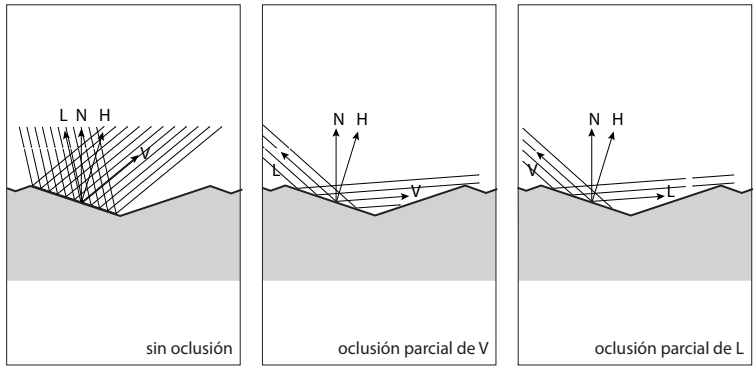


Figura 2.22 Modelo de Cook y Torrance.

interpola entre el color del material a 0º y el color a 90º para el que *F* es 1 y el color del material es el correspondiente a la luz.

Poulin y Fournier (1990). Anisotropía

Los modelos anteriores parten del supuesto de que el material es isotrópico, lo que no suele ser cierto en un número importante de casos. Un metal bruñido en una dirección principal, las superficies acabadas por medio de tratamientos superficiales en una dirección dominante, las telas de satén en las que la urdimbre resalta sobre la piel, son ejemplos corrientes de superficies anisotrópicas. Pero otro tanto ocurre a una escala mayor con las superficies del agua, de las nubes, de la hierba o de los bosques vistos a gran distancia, de la tierra labrada o moldeada por el viento, etc. En todos estos casos, a gran escala, los detalles geométricos no pueden simularse directamente, lo que sería prohibitivo, sino que deben aproximarse por medio de modelos que puedan simular virtualmente las asimetrías por distribuciones estadísticas adecuadas.

Uno de los primeros modelo de referencia que tuvo en cuenta la anisotropía es el modelo propuesto por Pierre Poulin y Alain Fournier en un artículo de 1990. Antes de este artículo, Kajiya había propuesto un primer modelo anisotrópico (Kajiya, 1985) basado en la teoría de la difracción de Kirchoff, que Poulin y Fournier citan en la introducción, indicando también sus importantes restricciones. Otra referencia importante citada por los autores es la de Cabral *et al*. (1987) que almacenaban los resultados de reflectancia en una tabla pero representándolas por medio de *spherical harmonics*, una variante importante pero más costosa. Otra referencia más cercana, pues introduce la idea de simular la anisotropía por medio de cilindros, fue debida a G.Miller, en 1988, pero que también implicaba limitaciones importantes.

El modelo de Poulin y Fournier se basa en representación de las alteraciones geométricas locales que dan lugar a reflejos anisotrópicos por medio de pequeños cilindros, alargados y paralelos localmente (aunque puedan no ser paralelos a escala de toda la superficie sobre la que, por ejemplo, podrían formar círculos concéntricos). Esta simulación resulta efectiva para casos característicos como los que hemos comentado. Es importante remarcar que la geometría, como ocurre en la realidad, no es visible: lo que vemos son los reflejos que destacan y que nos hacen intuir la geometría.

Los cilindros pueden ser tanto positivos, representando protuberancias (caso del satén), o negativos (caso de los surcos dejados por un cepillo sobre el metal). Hay dos parámetros que permiten modificar las características de la superficie: la distancia *d*, entre centros de cilindros, y la altura *h*, a la que sobresalen o se hunden con respecto al plano de la superficie.

El primer parámetro afecta a la anisotropía del modo siguiente. Si suponemos un radio 1 para los cilindros, la anisotropía, como puede observarse en la figura 2.23, será nula para $d = 0$ y máxima para $d = 2$ y, para $d > 2$, aparecerá un suelo entre los cilindros. En este caso la anisotropía será una combinacion entre las normales, variables para los cilindros y la normal constante correspondiente al suelo. Un valor bajo de *d* es adecuado para superficies con anisotropía poco

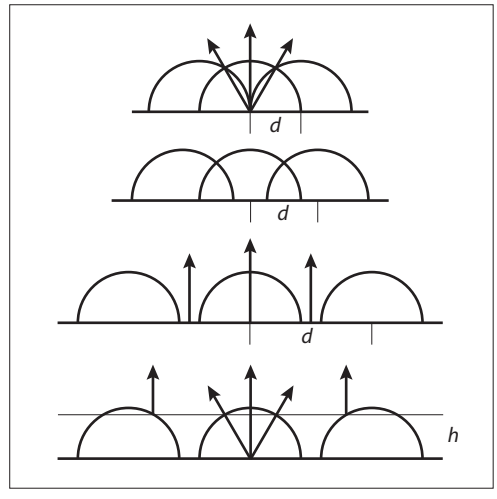


Figura 2.23 Modelo de Poulin y Fournier.

pronunciada como las superficies bruñidas. Un valor alto de *d* es adecuado para superficies con anisotropía muy marcada como telas muy rugosas o superficies naturales. Pueden simularse techos metálicos ondulados con valores muy altos.

El segundo parámetro, *h*, varía entre 0 y 1 y afecta a la altura a la que puede elevarse el suelo de tal modo que pueda modificarse la influencia del primero. Un valor alto de *h* reduce la variación de las normales y aumenta la importancia relativa de las normales correspondientes al suelo. Y viceversa.

A partir de aquí se llega a una serie de fórmulas similares a las del modelo de Torrance-Sparrow que incluyen los efectos de sombreado y oclusión que en aquel modelo se integraban en el parámetro *G*. De hecho, puede considerarse este modelo como una variante del de Torrance/Sparrow en donde las microfacetas son de diferente tipo.

He-Torrance-Sillion-Greenberg (HTSG, 1991)

Estos autores desarrollaron un modelo analítico basado en la óptica física y en las teorías de difracción de Kirchhoff. Se trata de un modelo que extiende el modelo de Coook y Torrance, basado en la óptica geométrica para incluir fenómenos de difracción, interferencia y polarización. Es un modelo importante desde el punto de vista teórico pero poco utilizado en la práctica.

Básicamente distingue tres tipos de reflexión: especularidad ideal (*sp*), difusa direccional (*dd*) y uniformemente difusa (*ud*). Los dos primeros componentes corresponden a las reflexiones primarias (véase la figura 2.24) y el tercero, a la dispersión debida a la difracción por la rugosidad de la superficie. La función BRDF se divide en tres partes, a cargo de cada una de estas formas de reflexión. Es decir:

$$f_r = f_{r,sp} + f_{r,ud} + f_{r,\,dd}$$

El modelo se basa, como el de Cook y Torrance pero de un modo más elaborado, en

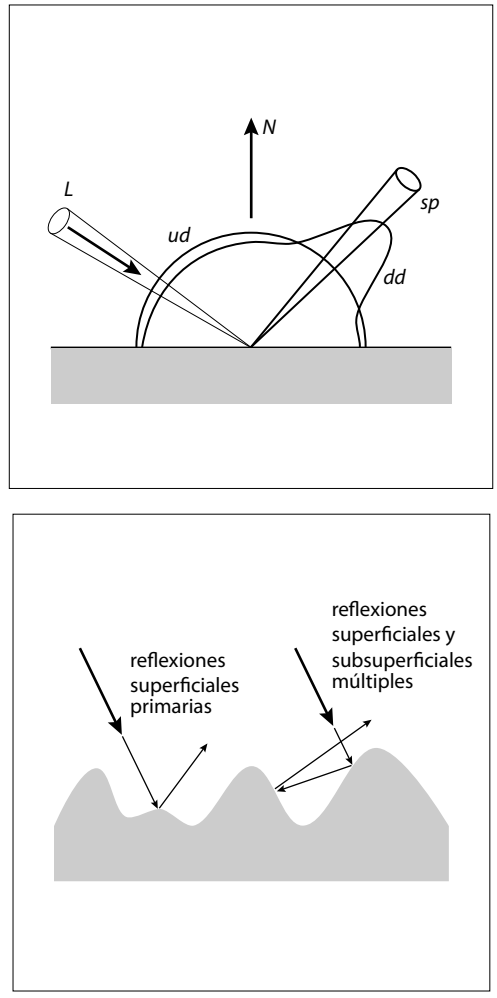


Figura 2.24 Modelo HTGS.

el análisis de las reflexiones y subreflexiones (véase la figura) a que da lugar la microgeometría de la superficie considerada. Esta geometría se representa por medio de funciones de distribución estadística y se basa en dos parámetros principales. Con el parámetro σ_0 se representa la rugosidad de la superficie (de modo similar al parámetro *m* en el modelo de Cook y Torrance). Con el parámetro τ se representa la longitud de autocorrelación, esto es, la distancia entre picos de la superficie. A partir de aquí se desarrollan una serie

de ecuaciones muy complejas que no voy a incluir aquí. El lector interesado las puede encontrar en el artículo original (véanse las referencias) o en el resumen incluido en la obra de Glassner (1995), en el apartado 15.5, pp. 744-747.

Ward (1992)

El modelo propuesto por Gregory Ward en 1992 contiene dos variantes, una isotrópica y otra anisotrópica. Al igual que el de Phong es un modelo empírico, no físico. Y su objetivo explícito era encontrar la fórmula empirica más sencilla que diera resultados satisfactorios. Los resultados se contrastaron con las mediciones dadas por un gonioreflectómetro.

En este modelo se utiliza una distribución gaussiana para la distribución geométrica de microfacetas de la superficie (la distribución más sencilla utilizada en modelos previos al de Cook y Torrance, que asume variaciones mínimas en la altura local de las facetas). Se prescinde también del término de Fresnel. En lugar de estos factores, Ward utiliza un único factor de normalización ($1/4\pi\alpha^2$ en la fórmula que sigue) que asegura una integración regular de la distribución de reflectancias sobre el hemisferio que rodea el punto de la superficie a calcular.

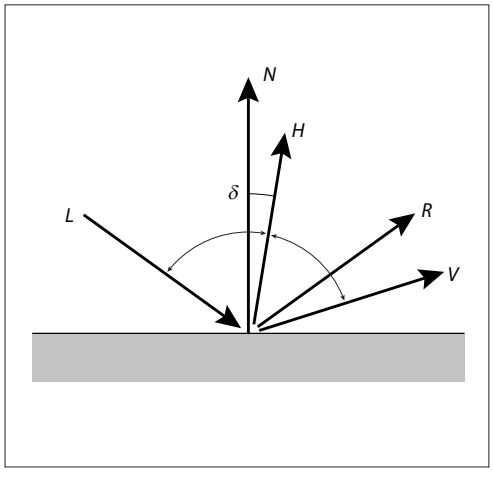


Figura 2.25 Modelo de Ward.

La función BRDF de la versión isotrópica se obtiene a partir de la fórmula siguiente:

$$f_r = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{\exp\left[-\tan^2(\delta/\alpha^2)\right]}{4\pi\alpha^2}$$

donde ρ_d y ρ_s son los coeficientes de reflexión difusa y especular respectivamente y dependen de la longitud de onda por lo que en principio podrían incorporar el factor de Fresnel; θ_i y θ_r son los ángulos de incidencia y reflexión, δ es el ángulo entre *N* y *H* (véase la figura 2.25) y α es la desviación estándar (RMS, *Root Mean Square*) de la pendiente de la superficie.

El modelo viene a ser similar al de Phong pero, a diferencia de este, sus desviaciones con respecto a los modelos físicos son menores. Así, en la mayoría de las aplicaciones del modelo de Phong, las dos propiedades principales de la función BRDF, la simetría con respecto a la incidencia y la reflexión y la conservación de la energía, no se cumplen, mientras que en este modelo sí.

La función BRDF de la versión anisotrópica introduce el ángulo azimutal ϕ, correspondiente a la dirección de incidencia de la luz y divide el parámetro α, que representa la desviación estándar de la pendiente de la superficie, en dos parámetros, α_x, α_y que representan la misma desviación pero desdoblada en las dos direcciones principales. La fórmula modificada queda así:

$$f_r = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{e^{\left[-\tan^2\delta\left(\frac{\cos^2\phi}{\alpha_x^2}+\frac{\sin^2\phi}{\alpha_y^2}\right)\right]}}{4\pi\alpha_x\alpha_y}$$

donde ρ_d, ρ_s, θ_i, θ_r, δ son, como antes, los coeficientes de reflexión difusa y especular, θ_i y θ_r los ángulos de incidencia y reflexión y δ el ángulo entre *N* y *H*. El parámetro ϕ es el ángulo azimutal correspondiente a la dirección de incidencia, α_x es la desviación estándar de la pendiente de la superficie en la dirección del eje X, α_y es la desviación estándar de la pendiente de la superficie en la dirección del eje Y.

Oren-Nayar (1993)

El modelo de Lambert da resultados incorrectos cuando se aplica a superficies difusas pero rugosas. Una esfera de reflexividad uniforme, iluminada frontalmente, muestra, en el modelo de Lambert, los bordes más oscuros que la parte central. Sin embargo una esfera real, rugosa, tiene un aspecto plano, con los bordes tan luminosos como la parte central debido a que la intensidad de la reflexión es mayor en la dirección del observador. Un ejemplo notorio es la Luna que aparece plana cuando hay luna llena (es decir cuando está iluminada por el Sol sin que la Tierra le haga sombra). De hecho, gran parte del trabajo previo, que se remonta a alrededor de 100 años, en que se basa este modelo, ha estado motivado por análisis de la superficie de la Luna.

La figura 2.26 da una explicación intutiva de este fenómeno. Si se analiza una distribución de microfacetas como la utilizada en el modelo de Cook y Torrance (en la que también se basa este modelo), se puede comprobar que la reflexión es más intensa desde el punto de vista del observador cuando este punto de vista coincide con la dirección de la fuente de luz, debido a que la orientación de las microfacetas favorece la reflexión en esta dirección. Sin embargo los modelos propuestos hasta la fecha no tenían en cuenta este hecho a la hora de calcular la contribución del factor difuso o lambertiano.

El modelo propuesto por Michael Oren y Shree Nayyar corrige esta limitación y, por esta razón, es más adecuado para simular telas o superficies mates con cierto grado de rugosidad. Puede ser considerado como una generalización del modelo de Lambert. Se basa en la estructura de microfacetas en forma cavidades simétricas en forma de V asumida en el modelo de Torrance-Sparroy y otros, que va unida a un tipo de distribución estadística que, en los casos más sencillos, es gaussiana.

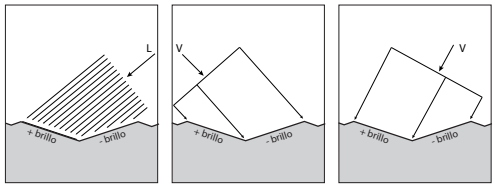


Figura 2.26 Modelo de Oren-Nayar.

La fórmula propuesta es:

$$f_r(w_i,w_r) = \frac{\rho}{\pi}(A + B\max(0,\cos(\phi_i - \phi_r)\sin\alpha\tan\beta)$$

donde:

A = 1 - 0.5(σ^2/(σ^2 + 0,33)
B = 1 - 0,45(σ^2/(σ^2 + 0,09)
α = max(θ_i, θ_r)
β = min(θ_i, θ_r)

Como en los casos anteriores, ρ es el albedo de la superficie. Y σ es un parámetro que determina la rugosidad de la superficie y que varía entre 0 y 1. En el caso de que σ = 0, resulta que A = 1 y B = 0 con lo que la fórmula se reduce a:

$$f_r(w_i,w_r) = \rho/\pi$$

Que es la del modelo de Lambert que ya hemos visto.

Lafortune *et al*. (1997)

El modelo propuesto por Eric Lafortune, Sing-Choong Foo, Kenneth Torrance y Donald Greenberg, debe entenderse en un doble contexto. Por un lado, en una situación en que, pese a los grandes avances teóricos, el modelo de Phong, con todas sus limitaciones, seguía siendo el método más utilizado en la práctica debido a su sencillez y a su eficacia. Por otro lado, en una situación en que se estaba avanzando en el campo de las mediciones reales y era factible refinar el modelo empírico de Phong para ajustarlo mejor a estas mediciones.

El más joven de los cuatro autores (los otros tres ya eran nombres ilustres en la investigación en este campo), presentaba, ese mismo año, una tesis de máster precisamente sobre este tema: “A gonioreflectometer for measuring the bidirectional reflectance of materials for use in illumination computations”

(Master's thesis, Cornell University, Ithaca, Nueva York, julio de 1997).

El propósito de este algoritmo era, por tanto, mejorar los resultados del de Phong para adecuarlo a las mediciones reales. En particular, los autores hacen ver algunas discrepancias notables con lo que ocurre en la realidad. Si se representa una escena con una superficie relativamente brillante, a medida que el ángulo de visión se hace más rasante, la especularidad decrece. Pero en la realidad, como es fácil comprobar, sucede justamente todo lo contrario: a medida que el ángulo se hace más rasante, la especularidad aumenta y el detalle de la textura se pierde.

Esto ya se había tenido en cuenta en los algoritmos previos pero la respuesta había sido, como ya hemos visto, intentar mejorar el modelo teórico para acercarlo al comportamiento físico real. Lo que se propuso este artículo fue mantener las ventajas de una aproximación empírica pero buscando adecuarlo mejor a los datos reales. Para ello, a partir de una reformulación del algoritmo de Phong, que también hemos visto anteriormente (reescrita en los mismos términos que estamos utilizando para otros algoritmos: $f_r(w_i, w_r) = \rho_s C_s (V.R)^n$) se llega a la siguiente fórmula:

$$f_r(w_i, w_r) = \rho_s (C_x w_{ix} w_{rx} + C_y w_{iy} w_{ry} + C_z w_{iz} w_{rz})^n$$

en donde C_x, C_y, C_z son factores de normalización en tres dimensiones que permiten ajustes más precisos y que incluyen la anisotropía (el caso isotrópico vendría dado por C_x=C_y) y en donde el modelo de Phong sería un caso particular de este modelo más general en el que

$$-C_x = -C_y = C_z = \sqrt[n]{C_s}$$

La figura 2.27 muestra como el ajuste de estos parámetros permite modificar el modo en que la reflexión especular aumenta para ángulos más rasantes, uno de los problemas principales que presentaba el modelo de Phong que incluía un único lóbulo. La BRDF

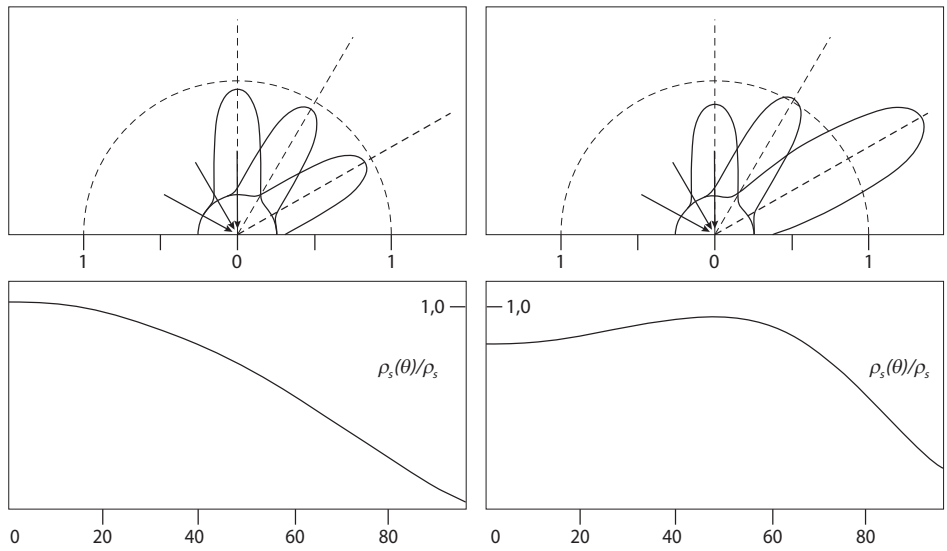


Figura 2.27 Modelo de Lafortune. Diagramas polares. A la izquierda, modelo clásico de Phong con ρ_s = 0,2, n = 20 y un término lambertiano ρ_d= 0,8, para ángulos de incidencia de 0º, 30º y 60º. El diagrama inferior muestra la disminución relativa del albedo del término direccional/difuso en función del ángulo de incidencia. A la derecha, modelo generalizado con ρ_s = 0,2, n = 20, C_z/C_x= 0,95 y un término lambertiano ρ_d= 0,8, para ángulos de incidencia de 0º, 30º y 60º. El diagrama inferior muestra como el albedo del término direccional/difuso disminuye solo para ángulos de incidencia muy grandes. Adaptado de Lafortune et al. *(1997).*

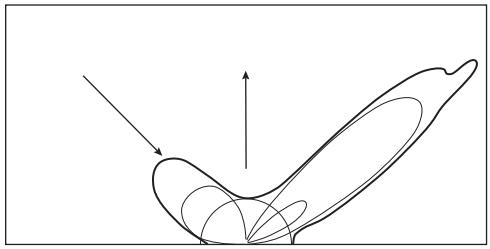


Figura 2.28 Modelo de Lafortune. Diagrama polar que muestra la superposición de tres lóbulos característicos del modelo de Lafortune, junto con un componente difuso uniforme, en línea más fina, y del resultado global, en línea más gruesa.

obtenida con este método puede considerarse como la suma de múltiples lóbulos de tipo Phong. La tercera figura, 2.28, muestra como un ajuste adecuado de los lóbulos permite simular prácticamente cualquier tipo de distribución.

Ashikhmin y Shirley (2000)

El modelo propuesto por Michael Ashikmin y Peter Shirley en 2000 y 2002 es un modelo empírico, inspirado en el modelo de Ward de 1992, y que tiene ventajas importantes sobre otros modelos empíricos anteriores: cumple las leyes de conservación de energía y reciprocidad, permite un control directo y muy eficaz de la anisotropía, utiliza parámetros intuitivos, da un buen control del efecto Fresnel combinado con un término difuso no constante que puede decrecer para ángulos de visión rasante, algo que ocurre en la realidad pero que no se cumplía en los modelos anteriores y, por añadidura, se adapta bien a métodos de Montecarlo que desde hacía ya unos años se habían convertido en el método principal de cálculo de sistemas de iluminación avanzada. Otra diferencia importante es que, al igual que el modelo de Ward, está desarrollado a partir de comprobaciones con mediciones de materiales reales, principalmente metales y plásticos.

La función BRDF es una suma clásica de un término difuso y un término especular. En el caso de metales el término difuso se reduce a cero, en el caso de superficies mates el término especular se mantiene pero con valores muy bajos, y en el caso de superficies brillantes los dos términos se utilizan con valores variables según los casos. Una característica notable de este método es que la evaluación del componente difuso es más precisa y elaborada que en modelos previos. El modelo presupone que el material tiene dos capas: una superior, pulida, muy reflectante, y otra inferior, el substrato propio del material. La figura 2.29 ilustra el modo en que, con ángulos no rasantes, el reflejo es una combinación del componente difuso y el componente especular mientras que a partir de un cierto ángulo solo actúa el componente especular.

El modelo depende de cuatro parámetros: *a*) R_s, que controla la reflectancia especular con ángulos de incidencia normales de la capa superior, *b*) R_d, que controla la reflectancia difusa del substrato, *c*) n_u, un exponente similar al de Phong que controla la forma del lóbulo especular en la dirección *u*, d) n_d, un exponente similar al anterior para la dirección *v*. Los vectores *U*, *V* forman una base ortonormal con *N*, la normal a la superficie en el punto considerado. Si $n_u = n_v$ el material es isotrópico. El rango de valores de n_u y n_v es de 0 a 10.000. El diagrama de la figura 2.29, derecha, muestra el modo en que actúan estos exponentes: si la distribución de las microfacetas se orienta en la dirección *u* o en la dirección *v*, se toma en cuenta uno de estos exponentes. Con otras orientaciones, el exponente se obtendría encontrando el valor correspondiente al ángulo azimutal ϕ.

Las ecuaciones son las siguientes (los términos utilizados por los autores son k_1, k_2 en lugar de w_i, w_r, que mantengo para facilitar la comparación con otros BRDF y ρ en lugar de f_r.; otras versiones de este artículo utilizan los términos *L*, *V*):

$$f_r(w_i, w_r) = f_{rs}(w_i, w_r) + f_{ds}(w_i, w_r)$$

donde la ecuación correspondiente al término especular es:

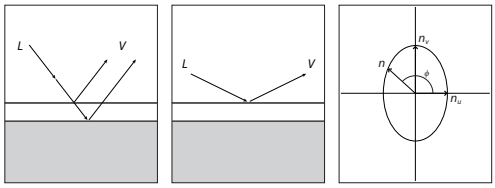


Figura 2.29 Modelo de Ashikhmin y Shirley.

$$\rho_s = \frac{\sqrt{(n_u+1)(n_v+1)} \times (N.H)^{n_u \cos^2\phi + n_v \sin^2\phi}}{8\pi (H.W)\max((N.Wi)(N.Wr))} F(H.W)$$

y la correspondiente al término difuso:

$$\rho_d = \frac{28R_d}{23\pi}(1-R_s)\left(1-\left(1-\frac{N.W_i}{2}\right)^5\right)\left(1-\left(1-\frac{N.W_r}{2}\right)^5\right)$$

donde R_s, R_d, n_u, n_v son los términos citados más arriba, N y H son los vectores correspondientes a la normal a la superficie y al ángulo mitad entre L y V, ϕ es el ángulo azimutal que también hemos visto en modelos anteriores, W_i, W_r son los vectores correspondientes a la dirección de incidencia L y salida V, y F es el término de Fresnel que los autores utilizan mediante una fórmula aproximada que no incluyo y que se puede encontrar en el artículo original.

BRDF obtenida por mediciones

La alternativa a obtener la función BRDF por medios analíticos es obtenerla a partir de mediciones, tomando medidas sistemáticas de una muestra real. Esto es bastante complicado desde un punto de vista técnico y puede llevar, como de hecho ha llevado, a importantes discrepancias en los resultados. Una de las primeras propuestas fue debida a Ward que, en el mismo artículo que he citado y en que se proponía un modelo concreto de BRDF (1992), presentaba un método para obtener valores por medio de un reflectofotómetro alternativo. El método clásico para obtener estos valores es por medio de un gonioreflectómetro o gonioespectrómetro (de *gonia*, "ángulo" en griego, una variante del sextante clásico que se utiliza para medir ángulos) para enviar la luz en una determinada dirección y recoger la reflectancia en otra dirección.

La figura 2.30 muestra un ejemplo simple de un aparato de estas características. Consta básicamente de una plataforma giratoria donde se coloca la muestra a medir, una fuente de luz que puede estar incorporado al propio dispositivo, como en la figura, pero que también puede enviarse desde un foco externo situado con precisión con respecto al dispositivo y un sensor o detector de reflectancias. Modificando los ángulos de emisión y de recepción se pueden elaborar diagramas del modo en que la reflectancia cambia en cada caso.

Sin embargo, el número de tomas necesarias requiere métodos complementarios para interpolar adecuadamente los datos. Si la resolución angular fuera de 0,5º, el número de medidas necesarias sería de más de 45 millones. Dado que esto es impracticable, la pregunta crucial que hay que hacerse es ¿cuál debe ser la resolución mínima para que la muestra de datos sea significativa y pueda ser tratada posteriormente, de modo que se acerque suficientemente al caso real?

Pero esta pregunta tampoco tiene fácil respuesta pues las distribuciones de reflectancia de muchos materiales son notoriamente complejas. Es necesario contar con algunas funciones básicas, como en los métodos propuestos por Matusik (2003) y, en cualquier caso, el número de datos requeridos sigue siendo muy elevado.

Estos datos se almacenan para cada material, lo que plantea el problema adicional de almacenamiento que se superpone al de tratamiento de los datos. La utilización directa de los datos no es práctica porque requiere

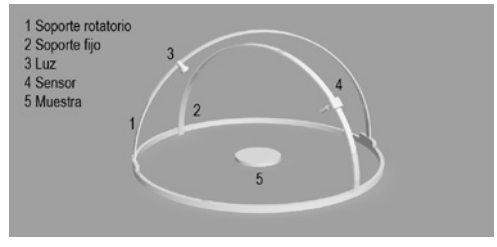


Figura 2.30 Esquema de un gonioreflectómetro.

mucha memoria. El modelo basado en mediciones propuesto por Matusik, por ejemplo, requiere del orden de 17 mb por material, lo que en una escena con varios cientos de materiales supondría una carga excesiva.

Se han propuesto métodos de representación que permitan comprimir adecuadamente la información almacenada en grandes cantidades de puntos. Cuatro métodos han mostrado ser suficientemente eficaces: *splines*, *spherical harmonics*, *wavelets* y *polinomios de Zernike*. Otros métodos de interés son los propuestos por Lawrence *et al*. (2004) basados en la factorización de datos BRDF, lo que tiene ventajas para ciertos métodos de computación, como los basados en *importance sampling*. Sin embargo, se pierde precisión con respecto a los datos originales.

Por ahora ninguno de estos métodos parece ser suficientemente eficaz, por lo que el principal uso que se ha dado hasta ahora a este tipo de mediciones es que sirvan como referencia para comparar los resultados y comprobar la precisión de modelos más sencillos y más utilizables en la práctica.

En cualquier caso, hay varias universidades y laboratorios que han emprendido la labor de medir y catalogar materiales con alguna variante de estos métodos. Los principales son los siguientes. En el laboratorio de Cornell se han tomado medidas de diversos materiales y los resultados pueden consultarse directamente por internet (véase Cornell en las referencias). En Columbia y Utrecht se ha creado una base de mediciones de unos 60 materiales a partir de 200 tomas (diferentes direcciones de observación e iluminación) para reproducir la BRDF de estas muestras de materiales. Véase Curet (Columbia-Utrecht) en las referencias. Una tercera referencia importante es el NIST (National Institute of Standards and Technology) uno de cuyos muchos proyectos está también dedicado a este tema en conexión, entre otras, con una base de datos creada en Oregón, la *Oregon BRDF Library* (OBL), que incluye una amplia colección de funciones BRDF gracias a la colaboración de físicos y especialistas en computación gráfica.

La función BTF

Estos métodos de medición incluyen también la medición de la variación espacial de la reflectancia, es decir, la textura.

De aquí ha surgido una función adicional, la función bidireccional de textura o BTF por sus siglas en inglés (*Bidirectional Texture Function*). Los primeros trabajos en esta área se remontan a finales del siglo pasado, véase Dana *et al*. (1999) o Kautz y Seidel (2000), Suykens *et al*. (2003), McAllister *et al*. (2002).

La función BTF es una función de 6 dimensiones que depende, por un lado, como la función BRDF, de la dirección del punto de vista y del ángulo de iluminación y, por añadidura, de las coordenadas *x*, *y* de las texturas planas asociadas al material en cuestión. Los valores de esta función se obtienen de varios miles de imágenes tomadas de la muestra del material de que se trate, a partir de diferentes combinaciones de iluminación y observación. Estas imágenes se procesan y se combinan con varios métodos que aún están en fase de investigación para llegar a resultados manejables en la práctica.

En Bonn hay una base de datos BTF, iniciada en 2002 y basada en 6 materiales y 6.561 combinaciones de diferentes puntos de vista y direcciones de luz. Véase Bonn en las referencias.

Aunque este tema corresponde propiamente a la sección siguiente lo incluyo aquí porque por ahora tiene más interés teórico que práctico y el tema principal es la posibilidad de obtener funciones ligadas a tomas de datos reales.

2.5 Simulación de texturas. Métodos principales

Tipos de texturas y tipos de aplicaciones

La palabra *textura*, como ya hemos visto en el capítulo anterior, se aplica en dos sentidos principales, como texturas planas que se ca-

racterizan porque no hay variación geométrica local pero sí variación de absorción (color), y texturas rugosas en las que ocurre básicamente lo contrario: puede no haber variación de color pero hay variación geométrica que modifica el reflejo.

Las texturas planas se simulan por medio de proyecciones de imágenes planas. Son las más fáciles de simular y las que dan resultados más satisfactorios. Sin embargo hay que tener en cuenta varios aspectos. Primero, el término "textura plana" es una descripción de variación local a pequeña escala, que vale tanto para un muro plano como para un muro curvo. Pero el método de proyección no será el mismo en los dos casos. Segundo, la resolución de la imagen que utilicemos debe ser "adecuada", esto es, ni demasiado "grande" ni demasiado "pequeña". Pero no es posible dar una respuesta sencilla de lo que quiere decir "grande" o "pequeña". En tercer lugar, si la textura abarca una superficie muy grande, tendremos que hacer todo lo posible para poder abarcarla con una textura más pequeña pero que se repita y, al mismo tiempo, hacer todo lo posible para evitar que estas repeticiones sean visibles. Estos problemas técnicos característicos se abordarán con más detalle en los capítulos siguientes.

Las texturas rugosas se pueden simular de tres modos principales. Primero, mediante una simulación del relieve al hacer el *render*, es decir, procesando el resultado en la propia imagen, sin modificar la geometría de la escena. Esto se hace superponiendo a la textura un tipo especial de mapa de proyección denominado mapa de relieve que, como veremos, puede ser de varios tipos. Segundo, mediante una simulación ligada a un mapa que lleva a cabo una modificación transitoria de la geometría del objeto durante la representación de la escena, pero que no altera la geometría de la escena de modo permanente. Tercero, mediante una modificación real y permanente de la geometría de la escena. El primer método es el más sencillo pero también el más limitado y puede resultar insuficiente si el relieve va a verse con suficiente detalle. El segundo consume más recursos y no funciona igual de bien con todos los programas. Es recomendable para aquellos casos en los que el relieve va a verse en primer plano, con suficiente detalle. El tercero solo está justificado en casos en los que el detalle se va a ver muy de cerca.

Las texturas se utilizan para una variedad de aplicaciones que también se analizarán en los capítulos siguientes. Las principales son las siguientes:

Modificación del color local o color difuso. En el caso más corriente se produce una simple substitución: el color dado por los parámetros básicos del material asignado a la superficie se substituye por el color dado por la textura.

Modificación del color o la intensidad especular. Es una variante del caso anterior, para superficies reflectantes, en donde la textura se aplica solo a las partes brillantes (para simular efectos de reflexión del entorno) o se utiliza para modificar otras características del reflejo.

Modificación del relieve. En este caso el mapa se utiliza para alterar la geometría de la superficie de diversos modos tal como se ha avanzado antes.

Modificación de la transparencia. Si el mapa incorpora un canal alfa (un cuarto canal añadido a los tres primarios, RGB) los valores de este canal pueden utilizarse para anular los valores superficiales de modo que se tengan en cuenta los de cualquier otra superficie que está en la misma línea de representación.

Creación de *Lightmaps*. En aplicaciones interactivas, principalmente en juegos de vídeo, no es posible llevar a cabo un cálculo de iluminación de la escena a la velocidad requerida si la escena debe modificarse con rapidez para adaptarse a la interacción con el usuario. La solución es activar un único cálculo previo y convertir el resultado, los gradientes de iluminación, en textura y superponer esta textura a la del material. Es un método que tiene bastante que ver con los métodos primitivos de simular reflejos por medio de mapas de

entorno (*environment maps*) que se describen también en los apartados siguientes.

Hay otras aplicaciones que también veremos pero estas son las principales.

Texture mapping (Catmull, 1974)

La proyección de texturas para incorporar detalle a las superficies se comenzó a aplicar desde los comienzos de los métodos de simulación de materiales, concretamente a partir de la tesis doctoral presentada por Edwin Catmull (nacido en 1945 y hasta hace poco presidente de Walt Disney Animation Studios y de Pixar Animation Studios) en la Universidad de Utah, en 1974, y publicada al año siguiente.

El procedimiento básico sigue siendo el mismo aunque hay varios aspectos que convendrá recordar. El método presentado por Catmull estaba aplicado a parches paramétricos bicúbicos, lo que permitía solucionar limpiamente muchos problemas que aparecen cuando la proyección de texturas se aplica a polígonos planos. De hecho, muchos de estos problemas se siguen evitando si se trabaja con NURBS (cuya parametrización interna es similar a los parches paramétricos utilizados por Catmull). El problema es que la mayoría de los modelos no utilizan NURBS ni parches paramétricos, que son herramientas sofisticadas y que solo se utilizan para modelar superficies de curvatura compleja, lo que en muchos modelos es algo innecesario.

El problema básico se resume en la figura 2.31. En el caso más general el problema es desarrollar una transformación que proyecta un espacio 2D (el espacio de textura) sobre otro espacio 2D (el espacio de la imagen de salida) a través de un espacio 3D (el espacio del objeto). Como veremos más adelante, en el caso de mapas o texturas procedurales, también se puede proyectar un espacio de textura 3D sobre un objeto 3D, lo que elimina muchos de estos problemas a costa de tener que utilizar un algoritmo puramente artificial.

El método de Catmull ligaba las coordenadas *u*, *v*, propias del espacio textura (o las coordenadas *s*, *t*, como las denominan otros autores para evitar confusión con las de los parches geométricos) con las coordenadas *u*, *v*, propias del parche paramétrico. Y, a partir de ahí, se llevaba a cabo una subdivisión recursiva de ambos espacios hasta que la subdivisión del parche paramétrico se correspondía con un único píxel. Al llegar a este estadio, el color del píxel se obtenía a partir del área correspondiente en el espacio de textura. Este método es directo y fácil de aplicar. Lo sigue siendo, como decía, si contamos con elementos paramétricos.

Parafraseando un artículo posterior (véase Cook *et al.*, 1987) se podría introducir una distinción entre texturas de acceso coherente (aquellas en las que la parametrización del espacio de texturas se puede relacionar directamente con el espacio del objeto) y tex-

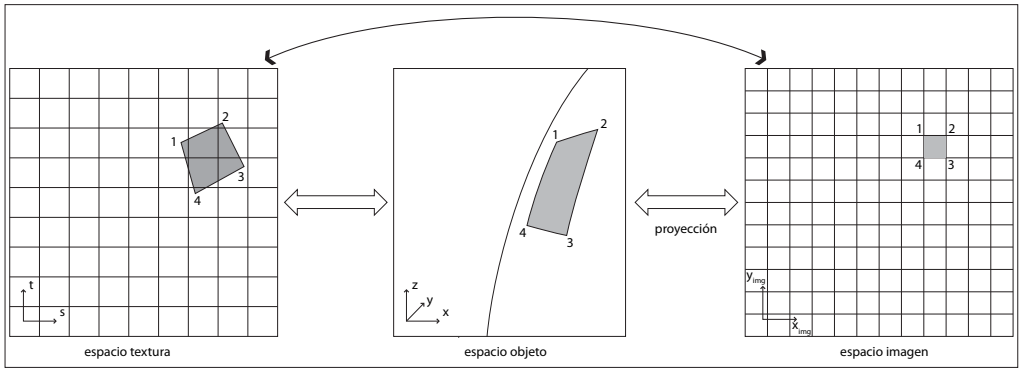


Figura 2.31 Texture mapping.

turas de acceso no coherente (aquellas que requieren una reparametrización).

También hay que mencionar que, debido a que el método utilizado subdividía los objetos de un modo no adecuado para los sistemas corrientes de representación, Catmull desarrolló un sistema alternativo, que denominó *Z-Buffer*, y que utilizaba una memoria adicional para guardar la coordenada *z* del objeto desde el punto de vista de la cámara. De este modo, para cada elemento, se comparaba su coordenada *z* con las previamente almacenadas, lo que permitía descartar cualquier elemento cuya coordenada *z* fuera mayor que otro pues esto implicaria que no sería visible, quedaría oculto por el de coordenada *z* inferior. Por añadidura, facilitaba la combinación de resultados obtenidos por un algoritmo con resultados obtenidos por otros algoritmos, lo que abría la puerta a sistemas más flexibles que incorporasen diferentes técnicas. Este método, que en la época se consideró muy eficaz pero costoso en términos computacionales, pasó a convertirse en el más utilizado a medida que los ordenadores aumentaron su potencia de cálculo y está incorporado (con variantes, como el *w buffer*) a todas las tarjetas gráficas actuales.

Si en lugar de esto tenemos un modelo basado en caras poligonales, hay que resolver otras dificultades. Dado que toda la información con que contamos son las coordenadas de los vértices (*x*, *y*, *z*), será necesario parametrizar de algún modo el interior de las caras de los polígonos para poder establecer una correspondencia con el espacio de texturas. Si la cara es plana, esto no supone otra dificultad importante que la posible falta de correspondencia de las resoluciones, de las proporciones respectivas o de ambas, un problema que veremos con más detalle en la segunda parte de este libro. Sin embargo, si la cara no es plana, será necesario encontrar un método adecuado de proyectar el espacio de textura, plano, sobre la superficie. Una primera revisión sistemática de los tipos de proyecciones requeridos según los casos se encuentra en Heckbert (1986), que es una buena introducción, aún vigente en muchos aspectos, a este tema. Los programas actuales proporcionan amplios recursos para intentar abordar este problema, recursos que también veremos más adelante. Pero es importante tener presente que en muchos casos no existe una solución perfecta.

La proyección de texturas se ha aplicado de múltiples maneras y para conseguir todo tipo de efectos. En los capítulos siguientes analizaremos y pondremos ejemplos de todas las técnicas apropiadas para cada caso.

Por último, también hay que recalcar que los métodos de proyección de texturas dan lugar a varios tipos de "artefactos" o *alias*, que también se revisarán más adelante. En general, las transformaciones son perspectivas por lo que la proyección final no cae en una retícula regular, tal como se muestra en la figura 2.31 (centro). Esto requiere filtros que varían espacialmente (*space variant filters*) y que son más difíciles de aplicar pues deben modificarse según la posición del elemento al que se aplican.

Environment mapping (Blinn y Newell, 1976)

La proyección de texturas se utilizó, un par de años después de la aportación de Catmull, para simular reflejos. Este fue el primer ejemplo de cómo la aplicación de mapas podía servir para algo más que para simular materiales con textura. Aunque actualmente los reflejos se simulan por métodos de iluminación avanzada, este método se sigue utilizando y está estrechamente relacionado con toda una serie de técnicas de importancia, entre ellas *Light mapping*, que ya he mencionado. Por otra parte introdujeron mejoras técnicas importantes al aplicar el método a casos más generales que los abordados por Catmull.

Blinn y Newell desarrollaron un método que consistía en dos pasos principales. En primer lugar, se obtenía una representación de la escena desde una cámara virtual situada en el centro del objeto cuyos reflejos

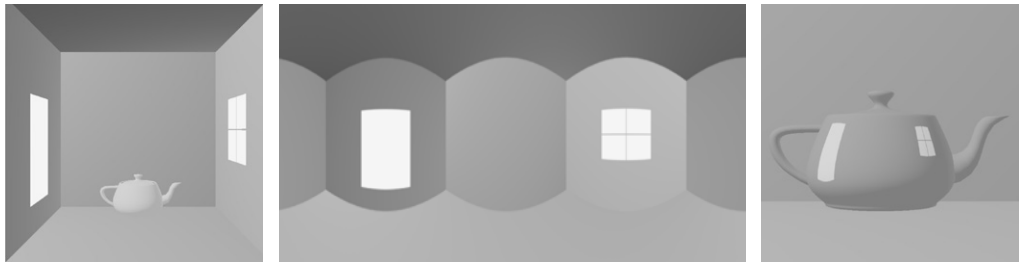


Figura 2.32 Environment mapping*: a) Vista general de la escena;* b*) Mapa de entorno obtenido desde la posición del objeto;* c*) Mapa reproyectado sobre el objeto como textura para simular reflejos.*

se querían simular. Esto resulta muy sencillo en escenarios virtuales pues todo lo que hay que hacer es una transformación del espacio de coordenadas. Esta representación puede desplegarse sobre una esfera o, alternativamente, sobre las seis caras de un cubo virtual. En cualquiera de los dos casos se puede obtener una correspondencia exacta entre los puntos de la superficie del objeto y la esfera o el cubo virtual que lo rodea.

En segundo lugar, el resultado de esta representación se reinterpreta como un mapa 2D que se proyecta sobre el objeto como si fuera una textura, siguiendo los mismos procedimientos que en el método de Catmull. El resultado es que el objeto parece reflejar la escena que le rodea con algunas restricciones que llevan a algunas complicaciones técnicas, que aquí no será necesario recordar.

El método funcionaba muy bien para objetos más o menos similares a una esfera pero bastante peor para objetos planos. Esto dio lugar a técnicas alternativas a veces denominadas *flat reflection mapping*, muy utilizadas a finales de la década de 1990 pero que prácticamente han desaparecido debido a la difusión de métodos de iluminación avanzada y al aumento de la potencia de los ordenadores.

Relieve con *Bump mapping* (Blinn, 1978)

La simulación de relieve por medio de *Bump mapping* se introdujo en 1978 por James Blinn (vése Blinn, 1977) y se ha utilizado ampliamente desde principios de la década de 1990 con la difusión de los PC y la incorporación de todas estas técnicas al software comercial y, posteriormente, a la GPU (*Graphics Processing Unit*).

La idea básica consiste en perturbar la dirección de la normal a una superficie antes de que sea computada por el sistema de cálculo de iluminación y enviada al sistema de representación. De este modo no es necesario modificar la geometría del objeto.

Para ello se usa un mapa que codifica desplazamientos virtuales por medio de patrones en blanco y negro o valores intermedios. Las zonas negras se asocian a desplazamientos negativos, a huecos, por debajo de la superficie real, y las zonas blancas a desplazamientos positivos, a relieves, por encima de la superficie real. Los valores grises se computan como valores de desplazamiento intermedio entre estos dos extremos.

La figura 2.33 muestra un esquema que resume el proceso. A una superficie con diferentes orientaciones, dadas por sus vectores normales, se le asocia una función (una tabla de valores que corresponden a diferentes intensidades del mapa asociado). El resultado de esta combinación es una alteración virtual de las normales que simula rugosidades en la superficie sin alterar la geometría real.

Los resultados son muy eficaces desde un punto de vista computacional pues se puede simular con un mínimo gasto el relieve de objetos. Sin embargo, pueden no serlo tanto

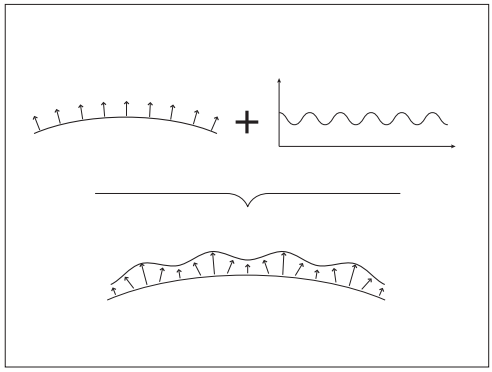


Figura 2.33 Bump mapping.

desde el punto de vista de la calidad de los resultados. Los principales inconvenientes son los siguientes: *a*) la silueta de los objetos se seguirá viéndo con su contorno real, *b*) otro tanto ocurre con sombras arrojadas que también mostrarían la silueta real del objeto, *c*) las oclusiones debidas al relieve no se computan.

Si la silueta no es visible o no hay sombras, el resultado puede ser bastante convincente. Pero si no es así, se puede arruinar el efecto. La única solución es ocultar los contornos o recurrir a los procedimientos más sofisticados que siguen.

Relieve con *Displacement mapping* (Cook, 1984)

La simulación de relieve con mapas de desplazamiento se introdujo en 1984 por Robert L. Cook. Con este método se altera la altura de la superficie según los valores dados por un rango de valores de grises dados por el mapa asociado, de modo similar a *Bump mapping*. Sin embargo, la superficie se modifica geométricamente con lo que los problemas mencionados al final del apartado anterior se solucionarían.

No obstante, los problemas técnicos son considerables. En el artículo original, Cook no dio detalles sobre cómo podía implementarse el método, solo describió su integración en un tipo especial de *shader*. En un artículo posterior (1987) dio algún detalle adicional de un tipo particular de implementación utilizando micropolígonos. Esta implementación estaba ligada al famoso algoritmo *Reyes* (*Renders Everything You Ever Saw*), desarrollado por Carpenter y Cook para el Computer Graphics Research Group de Lucasfilm, que posteriormente se convertiría en Pixar, y se utilizó por primera vez en 1982 para las imágenes del efecto Génesis en la película *Star Treck II: the wrath of Kahn*. Fue también uno de los puntos de partida principales del lenguaje RenderMan.

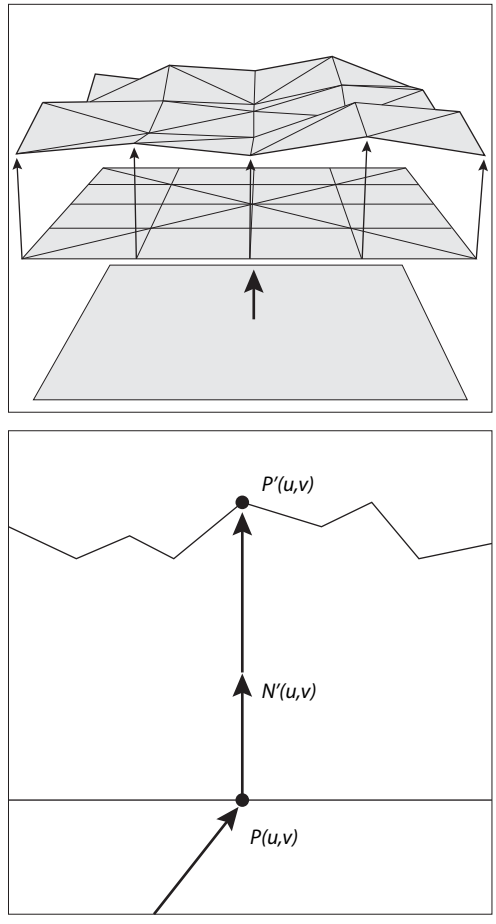


Figura 2.34 Displacement mapping. *Arriba: vista en perspectiva del desplazamiento virtual. Abajo: esquema de la modificación de posición de un punto.*

Sin embargo no se ha utilizado mucho, hasta hace poco, debido a su coste computacional, pese a los diversos intentos de mejorar el procedimiento básico. En su versión más simple, la superficie de base se subdivide recursivamente, hasta llegar a un tamaño de micropolígono relacionado con el tamaño de la imagen de salida, generalmente del orden de un píxel o menos. A partir de esta teselación básica, los vértices se empujan según los valores dados por un campo de alturas (*height field*), un campo escalar que se asocia a la superficie.

El desplazamiento se computa mediante un cálculo simple que define la nueva posición *P'* de los puntos de una superficie *P*:

$$P'(u,v) = P(u,v) + N'(u.v)H(u,v)$$

donde $P'(u,v)$ son las coordenadas del punto desplazado, $P(u,v)$ son las coordenadas del punto original, $N'(u,v)$ es el vector normal correspondiente al punto que define la dirección de desplazamiento y H es el valor correspondiente al campo escalar de alturas que define la altura a que se desplaza el punto en cuestión. La figura 2.34 ilustra estas operaciones básicas.

El campo escalar de altura se da corrientemente por medio de imágenes de mapas de bits en blanco y negro en los que los valores de altura están asociados al nivel de gris. Hay diversos métodos recientes que permiten aumentar la precisión por medio de mapas de bits en color con información más precisa sobre la altura y que se han utilizado para modelar terrenos.

El principal problema de este método es que el número de polígonos necesarios para conseguir resultados adecuados era, y es, muy alto. Y, en consecuencia, el coste computacional es muy elevado. Por esta razón se han desarrollado varios métodos alternativos que buscan en todos los casos reducir este coste. Entre ellos cabe citar los trabajos de Michael Doggett (2000) sobre teselación adaptativa e integración en hardware por diversos métodos, lo que puede resolver el problema del alto coste de computación.

Desde hace pocos años, con la integración de procedimientos de teselación en las tarjetas gráficas avanzadas, estos problemas se han reducido considerablemente.

Relieve con *Normal mapping* (varios, 1992, 1996, 1998)

Un método más eficaz que *Bump mapping* es lo que se denomina *Normal mapping*. El término fue introducido por Alain Fournier (el mismo autor que con Poulin presentó un algoritmo anisotrópico para calcular la BRDF en 1990, que hemos visto anteriormente) en un artículo de 1992, a partir de una interesante reflexión sobre los diferentes niveles de detalle de un modelo, de las técnicas requeridas en cada caso y del flujo que se requiere establecer en la práctica entre estos tres niveles.

Estos tres niveles dependen de la escala a que consideremos el material. El nivel superior y mejor definido es el geométrico (superficies definidas por polígonos, parches paramétricos o NURBS) que se puede denominar *macroscópico*. El nivel inferior, que también ha sido estudiado con gran detalle, es el nivel correspondiente a los modelos de reflexión local para los que se computa la BRDF y que puede ser denominado *microscópico*. El nivel intermedio ha sido menos explorado: es el nivel en que se encuentran detalles característicos apreciables a cierta escala tal como el relieve local. Es un nivel que por analogía con los anteriores puede denominarse *mesoscópico*.

A lo largo de este artículo, Fourier hacía ver que el cómputo adecuado del relieve implicaría un análisis del comportamiento real de las microfacetas, tal como se proponía en el modelo BRDF de Cook y Torrance, en el que realmente no nos encontraríamos con una normal correspondiente a una superficie entendida como la media de las variaciones locales, sino como una serie de diferentes normales correspondientes a las diferentes orientaciones microscópicas. Y que, tal como se analiza matemáticamente en el artículo, esto llevaría a

una función de distribución de normales como la que denomina en el artículo, NDF (por sus siglas en inglés, *normal distribution function*), una función que puede relacionarse directamente, mediante una serie de transformaciones, con la propia función BRDF.

Sin entrar en el detalle de este análisis, es claro que si, para cada punto de una superficie se guarda, en un lugar de una única normal, un triplete de vectores que den más información sobre las variaciones microscópicas de orientación del punto, esto puede asociarse a un cómputo de la iluminación que preserve las variaciones de iluminación del punto que pueden asociarse a su relieve.

A finales de la década de 1990 aparecieron una serie de artículos (Krishnamurthy y Levoy, en el Siggraph de 1996, Cohen *et al*., en el Siggraph de 1998, Cignoni *et al*. en el Congreso IEEE Visualization de 1998) que aportaban métodos para transferir información de polígonos con alta resolución a polígonos de baja resolución por medio de *normal maps*.

No debe perderse de vista este contexto: los nuevos métodos de simulación de relieve aparecieron como técnicas colaterales de una búsqueda en la que la motivación principal era reducir las decenas o centenares de miles de caras poligonales de un modelo elaborado con detalle, a cantidades más manejables, del orden de unos cuantos miles nada más, pero que preservaran la calidad aparente. Durante los primeros años de la década de 2000, estas ideas se desarrollaron y comenzaron a utilizarse extensamente en la industria de juegos de vídeo, como un método particularmente eficaz que podía substituir a los mapas de desplazamiento para simular relieves complejos con poco coste de computación.

Hay muchas técnicas de simplificación de polígonos y a algunas de ellas me referiré más adelante, en el apartado sobre multirresolución y nivel de detalle. Pero con independencia de la técnica utilizada para reducir el número de caras geométricas, en todos los casos se presenta el problema de preservar los atributos de color y textura del original. El artículo citado de Krishnamurty y Levoy pro-

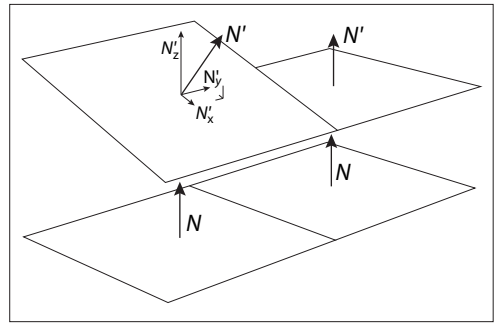


Figura 2.35 Normal mapping. *Esquema en perspectiva del desplazamiento virtual.*

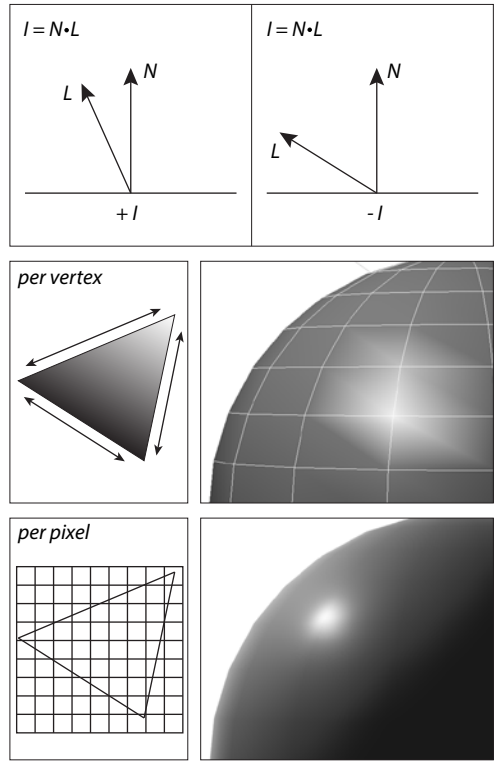


Figura 2.36 Normal mapping. *Detalle de los procesos implicados. Arriba: cálculo de la intensidad a partir de los vectores* L *y* N. *La intensidad será mayor cuanto menor sea el ángulo entre los dos vectores y viceversa. Abajo: cálculo por vértice y cálculo por píxel.*

ponía asociar al modelo derivado, valores correspondientes a la orientación en las tres direcciones pero codificados como valores RGB. La idea básica es incorporar información más precisa sobre el relieve geométrico guardándola en un mapa de bits especial. Esta información se codifica de tal modo que los píxeles rojos representan las coordenadas en *X*, los verdes en *Y* y los azules en *Z*. El resultado es una mezcla de estos tres colores que representará el relieve relativo de la superficie. La información sobre el relieve queda por tanto contenida en el mapa que, procesado adecuadamente, da como resultado una transformación de una superficie sin relieve, en la escena, a una superficie con relieve, en la representación.

Hay otra ventaja adicional importante y es que la utilización de *Normal mapping* con tarjetas gráficas modernas, optimizadas para procesar el cálculo de representación por medio de *pixels shaders* (véase más adelante el apartado sobre el proceso general de *rendering*), hace que el uso de este tipo de mapas sea más preciso y más rápido.

La figura 2.36 resume los conceptos principales que hay detrás de este método. El cálculo básico de iluminación de las caras se lleva a cabo, en todas sus variantes, a partir de un cómputo inicial por el que se obtiene el producto vectorial (producto punto) de los dos vectores fundamentales, el vector normal a la superficie, *N*, y el vector correspondiente a la dirección de la luz, *L*. Si estos dos vectores están normalizados, este producto es igual al coseno del ángulo que forman los dos vectores. Como puede apreciarse intuitivamente en la figura, si el ángulo se aproxima a 0º la intensidad se aproximará a su máximo, y si el ángulo se aproxima a 90º la intensidad se aproximará a 0. Entre estos dos extremos habrá mayor o menor intensidad.

El método más elemental de cálculo de iluminación, introducido por Gouraud en 1971 y que está en la base de otros métodos derivados, como los de Phong o Blinn, lleva a cabo este cálculo para cada vértice. Y los valores intermedios de una cara triangular se interpolan a partir de estos valores. El resultado básico es tosco aunque puede refinarse de diferentes modos. Pero con las mejoras técnicas de las tarjetas gráficas este cálculo puede realizarse para cada píxel lo que aumenta extraordinariamente la precisión y, si este tipo de cálculo está incorporado al hardware (es decir, si contamos con un tarjeta gráfica especializada, de las que se mencionan en el apartado citado sobre *rendering*), también es más rápido.

Relieve con *Parallax mapping* (Kaneko *et al.*, 2001)

Esta técnica de simulación de relieve, poco utilizada en la práctica por ahora, se entenderá mejor en el contexto de un conjunto de técnicas aparecidas a lo largo de la década de 1990 y que se conocen genéricamente como *Image-Based Rendering* (IBR). Por razones de espacio no puedo resumir sino lo principal pero puede encontrarse una buena introducción en Shum y Kang (2002).

La idea básica de estas técnicas es utilizar una serie de imágenes para modelar

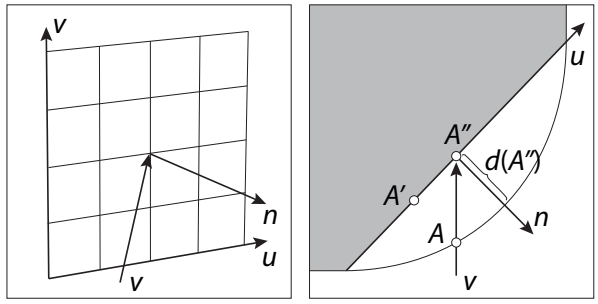


Figura 2.37 Parallax mapping.

una escena. La primera pregunta práctica que cabe hacer es, obviamente, cuál es el número de imágenes necesarias para poder interpolar de modo adecuado los valores resultantes. Y la respuesta es que no se puede responder a tal pregunta de modo general pues las variables son infinitas. Pero es posible dar una respuesta téorica que sirva como referencia para las diferentes variantes con que se ha buscado abordar este asunto. Esta respuesta teórica se da en un artículo publicado en 1991 por Adelson y Bergen, aunque hay otras referencias anteriores y posteriores que también habría que mencionar. En dicho artículo se planteaba una función ideal, la función plenóptica (*plenoptic function,* de *plenus*, "pleno", "lleno", y *optics*, "óptica") que definiría la intensidad de los rayos de luz que pasarían por el centro de una cámara en cada posición (V_x, V_y, V_z), para todo ángulo posible (θ, ρ), para una longitud de onda λ y en cada tiempo *t*. Es decir, una función de 7 variables. Si se eliminan las variaciones del tiempo y longitud de onda, tendríamos una función de 5 variables. Y si se acepta que la cámara está situada en el exterior de la escena, tenemos el *Light Field* (Levoy, 1996) o el *Lumigraph* (Gortler, 1996), dos propuestas teóricas que se basan en la generación de escenarios virtuales a partir de múltiples imágenes (reales o sintéticas). Con todo, la respuesta a cuántas imágenes son necesarias para simular adecuadamente una escena sigue sin poder responderse, en general. Pero sí puede decirse que este número se reduce si los movimientos de la cámara que explora la escena se reducen a, por ejemplo, seguir anillos concéntricos. En el extremo opuesto, hay versiones simples y muy conocidas como los panoramas generados con QuickTime. Pero hay versiones intermedias donde se pueden generar imágenes interpoladas que den claves tales como el cambio de paralaje que proporciona una clave de profundidad importante y a la que estamos muy acostumbrados. La figura 2.37 muestra una ilustración esquematizada de este tipo de claves.

Este es el punto de partida de las técnicas propuestas inicialmente por Kaneko, Takahei *et al*. (2001). La idea fundamental es que sería posible ampliar la calidad de los efectos de relieve si se modifica la posición de los píxeles de salida de una textura, de tal modo que este desplazamiento corresponda a los cambios de paralaje correspondientes a la modificación del punto de vista. Esto soluciona un problema intrínseco a los *Bump maps* pero que tampoco se solucionaba satisfactoriamente con los *Normal maps*. Y el interés de esta técnica no está solo en que se mejora la calidad del resultado sino que ésto se hace con un coste de computación bajo, pues nos ahorra las engorrosas modificaciones virtuales de la geometría que se requieren con los mapas de desplazamiento.

La idea básica es distorsionar la textura dinámicamente de tal modo que el desplazamiento se corresponda con la forma virtual que se quiere simular. La figura 2.37 ilustra el procedimiento que es relativamente sencillo, aunque su aplicación técnica implique varios problemas. En el lado izquierdo de la figura se muestra una vista en perspectiva de una textura plana, con sus coordenadas locales, *u*, *v*, y con los vectores de visión, *v* y normal a la superficie, *n*. A la derecha, se muestra una vista en planta de la misma textura a la que se ha superpuesto una línea que indica el desplazamiento virtual cuyo valor es *d*(*A"*). Para que el punto *A* aparezca sobre el contorno virtual, se necesita que el punto *A*' se desplace a la posición *A"*. Dado que conocemos el ángulo entre θ_u, entre el eje *u* y el vector *v*, se puede calcular facilmente este desplazamiento mediante fórmulas trigonométricas simples.

Esta primera versión presentaba unos cuantos problemas que se fueron resolviendo en los años siguientes. El principal era que, a medida que el ángulo aumentaba, los valores de desplazamiento tendían a infinito con lo que las zonas más alejadas en perspectiva quedaban deformadas. Una solución sencilla (propuesta por Welsh, 2004) era limitar el desplazamiento (una variante de *Parallax mapping* bautizada como

offset limiting) reduciendo su valor a partir de un determinado ángulo. Otros problemas técnicos se han ido mejorando también mediante diversas contribuciones, entre ellas la modificación del ángulo en función de la inclinación de la superficie (*steep parallax mapping*) o el cómputo adecuado de la oclusión. Hay técnicas similares que han recibido la denominación de *relief mapping* y que utilizan otros algoritmos de implementación, aunque los resultados a los que se llega son similares. Puede encontrarse un buen resumen de estas mejoras en Szirmay-Kalos y Umenhoffer (2008), junto con una revisión de las técnicas anteriores de simulación de relieve. Otra técnica posterior que da mejores resultados, al combinarla con técnicas de *Ambient occlusion* (véase una descripción de esta técnica en mi libro sobre simulación de la iluminación) es *Parallax Occlusion Mapping* (POM) que surgió por esas mismas fechas a partir de un artículo de Zoe Brawley y Natalya Tatarchuck (véase Brawley, Z.; Tatarchuk, N., 2004).

Sistemas de partículas (Reeves, 1983)

Los sistemas de partículas se utilizan principalmente en animación, un tema que este libro no incluye. Pero también se pueden utilizar sistemas de partículas para imágenes fijas pues todo lo que hay que hacer es activar la animación en un momento adecuado y grabar el resultado.

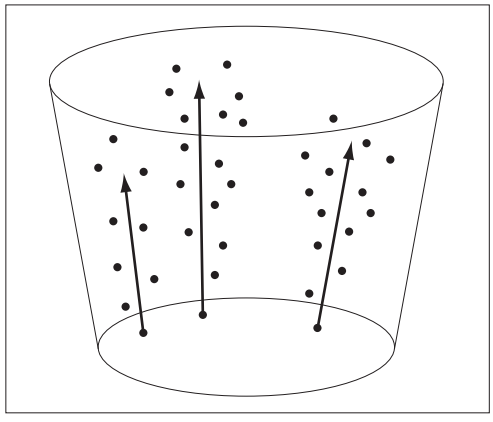


Figura 2.38 Sistemas de partículas. Los puntos indican las partículas, las flechas su dirección de desplazamiento y el cono truncado, la región afectada.

Los sistemas de partículas surgieron a partir de los trabajos de Reeves (1983), principalmente, y se utilizaron por primera vez en la película *Star Trek II: The Wrath of Khan* para simular una pared de fuego. Después se han usado mucho para generar fenómenos tales como el fuego, la lluvia, nieve o cualquier otro efecto que requiera modelar cientos o miles de pequeñas partículas que se dispersan por el espacio. También se han utilizado, con la denominación "sistemas estructurados de partículas" (*structured particle systems*), para modelar hierba y árboles, a partir de los primeros artículos de Reeves y Blau (1985). Reeves utilizó este sistema por primera vez para modelar árboles en una película corta, *The Adventures of Andre and Wally B*.

También pueden utilizarse sistemas de partículas como alternativa a la modelización de volúmenes cuando los gases son muy dispersos, como ocurre por ejemplo con el humo de un cigarrillo, de una chimenea o un fuego poco intenso. Como veremos más adelante hay otras alternativas para estos casos.

Un sistema de partículas consiste básicamente en dos cosas: *a*) Un tipo de partícula, con sus características formales (forma básica, esférica, poliédrica...) y materiales (color, textura, opacidad...) a las que se suman otras características dinámicas: posición inicial, tamaño inicial, velocidad, duración… En el primer sistema propuesto por Reeves, cada objeto tenía los siete atributos siguientes: posición inicial, velocidad y dirección iniciales, tamaño inicial, color inicial, transparencia inicial, forma y tiempo de vida; *b*) Un algoritmo que describe su generación, su movimiento (la dinámica propia del sistema) y su extinción.

Cada uno de estos atributos cuenta con una serie de parámetros propios. Además de estos atributos, un sistema de partículas también cuenta con una forma de generación que define una región en el espacio en la que

las nuevas partículas se irán distribuyendo aleatoriamente. Estas formas son volúmenes simples tales como esferas, cilindros o conos, y su elección depende del tipo de efecto que se quiere generar. La figura 2.38 muestra alguno de estos parámetros en un sistema de partículas destinado a simular una explosión y cuya forma de generación es un cono truncado invertido.

El algoritmo que describe la dinámica del sistema incluye procesos estocásticos que hacen que el resultado resulte muy complejo aunque los parámetros que definen esta complejidad sean relativamente sencillos. Los parámetros propios del algoritmo principal permiten controlar el número de partículas creado inicialmente, su densidad, y el modo en que pueden quedar afectadas por otros factores que pueden incluirse en la serie de parámetros de la simulación, tales como la fuerza del viento, la gravedad, la atracción de otros objetos o sistemas, la inclusión de turbulencias, etc.

En un sistema estructurado de partículas, tal como el que se utiliza para simular hierba o bosques, las partículas mantienen relaciones entre sí, formando una estructura cohesionada. Cada partícula representa un elemento del sistema (una rama, una hoja) por lo que, en este sentido, el sistema es similar a otros sistemas de modelado procedural basados en lenguajes formales. A diferencia de estos, sin embargo, la precisión geométrica no es fundamental y se dedica más esfuerzo a la representación del color con variantes que dependen de diversos factores. Por otra parte, al igual que los sistemas de partículas generales, pueden incluir factores tales como la fuerza del viento (las hierbas y los árboles pueden moverse suavemente debido a estos factores), la influencia de la orientación del Sol (el crecimiento puede ser mayor en una determinada dirección), etc.

Al igual que ocurre con los volúmenes de substancias difusas, se requieren procedimientos especiales de *rendering* que se describen brevemente en el apartado correspondiente, más adelante, sobre *volume rendering*.

También se han utilizado partículas para simular animales en movimiento. El primer ejemplo de este uso se debe a Reynolds (1987) que lo utilizó para simular el vuelo de enjambres de pájaros. En este caso, cada pájaro era un objeto poligonal específico con un sistema de coordenadas propio y su movimiento conjunto se controlaba por medio de algoritmos especiales. A diferencia de los sistemas de partículas, no había generacion y extinción y los métodos de *rendering* eran los tradicionales. En estos casos los problemas son de otro orden, muy interesantes pero que se alejan de la temática de este libro, por lo que los mencionaré telegráficamente. El problema principal en este caso es simular el movimiento de un enjambre que se caracteriza por una sofisticada relación entre movimientos individuales y movimientos de conjunto. El resultado a que se llegó partía de la especificación de una serie de parámetros que permitían controlar el movimiento individual (velocidad, aceleración, rotación en diferentes ejes, etc.) combinados con una serie de parámetros que controlaban la interacción de cada individuo con el grupo, por medio de tres reglas principales: *a*) la prevención de colisiones (cada individuo incluía un detector de distancias mínimas a partir del cual se modificaba su vuelo); *b*) la igualación de la velocidad (la velocidad media se contrastaba recurrentemente con la velocidad individual); *c*) la centralidad (cada individuo buscaba mantenerse centrado con respecto a sus vecinos inmediatos). Los resultados se han extendido posteriormente a la simulación de todo tipo de movimientos colectivos con resultados espectaculares que nos resultan familiares por sus muchas aplicaciones en el cine y la publicidad.

Procedural mapping (Perlin, Peachey, 1985)

Para entender la importancia de esta técnica, imaginemos que tenemos un “mapa” 3D que pueda representar, por medio de una función

matemática, cuál es el color de un bloque de mármol. Si fuera posible contar con una función así no tendríamos que preocuparnos de otra cosa que de establecer una relación directa entre la función que representa la textura y el objeto geométrico. Cada punto del espacio geométrico propio del objeto quedaría relacionada con cada punto del espacio material con lo que su color quedaría perfectamente especificado. No tendríamos que preocuparnos por el manejo de técnicas engorrosas de proyección y gestión de mapas como las que veremos en los capítulos siguientes. Técnicas tales como definir una proyección adecuada para cada objeto, como ligar cada definición de propiedades materiales a una textura que debe almacenarse en algún lugar de nuestro disco duro y que debe gestionarse adecuadamente para que no se pierda la relación con el archivo de la escena, como ajustar los parámetros del mapa para que se coordinen adecuadamente con la proyección definida para cada objeto, lo que no siempre es posible.

A esta ideal han intentado acercarse los muchos métodos de generación de texturas algorítimicas 3D, denominados corrientemente de obtención de texturas sólidas (*solid textures*) o mapas procedurales 3D (*procedural 3D maps*) o, integrando los dos términos, texturas sólidas procedurales (*procedural solid texturing*). Los pioneros fueron Ken Perlin y Darwyn Peachey que publicaron, el mismo año, 1985, de modo independiente, dos artículos en que describían métodos hasta cierto punto similares para obtener texturas sólidas por métodos matemáticos. Véase las referencias, Perling (1985) y Peachey (1985).

Pueden citarse de todas formas varios precedentes importantes. Blinn y Newell utilizaron, en 1976, métodos basados en síntesis de Fourier para generar patrones artificiales, un método que fue ampliado por Shacter y Ahuja en 1979 y por Schacter en 1980 para generar imágenes que pudieran servir en simulaciones de vuelo. Otros investigadores aplicaron técnicas basadas en fractales para generar patrones artificiales que podían ser utilizados como texturas, entre otros, Fourier, Fussell y Carpenter en 1982 o Haruyama y Barsky en 1984.

Un paso técnico importante que permitió utilizar texturales procedurales durante el proceso de *rendering* se debe a Cook (1984), que introdujo el concepto fundamental de *shader*, un concepto al que también volveremos. El uso de paquetes algorítmicos relacionados implicado en este concepto abriría el camino para el uso de texturas ligadas a transparencias, reflejos, relieves y texturas sólidas.

También hay que mencionar otras aportaciones importantes como la simulación de nubes por métodos procedurales, debida a Gardner (1984, 1985) y que aún se sigue utilizando. A diferencia de los métodos posteriores, se basaba únicamente en la combinación de funciones sinusoidales (análisis de Fourier) ajustando adecuadamente las diferentes frecuencias, amplitudes y fases. Es un método que requiere una considerable habilidad para evitar la aparición de patrones regulares, algo que es bastante fácil que ocurra cuando las funciones de base son uniformemente periódicas. Gardner consiguió resultados notables por medio de una serie de “números mágicos” colocados estratégicamente en su algoritmo para romper esta tendencia a la uniformidad. Los desarrollos posteriores han conseguido generar todo tipo de formas de nubes mediante variaciones de estos patrones básicos.

Con todo, el artículo de Perlin (1985), publicado cuando todavía era un estudiante de doctorado, un año antes de publicar su tesis doctoral, fue y sigue siendo la principal aportación a la generación de texturas procedurales.

La utilización de una función en lugar de una tabla de valores, en el caso de una textura 2D, no supone, en principio, ninguna dificultad pues todo lo que hay que hacer es substituir los valores correspondientes a una determinada celda por los valores dados por la función. Y en el caso de una textura 3D, la única dificultad es que aumenta la memoria

pues en lugar de, por ejemplo, 512 x 512 valores, tendremos 512 x 512 x 512. La verdadera dificultad está en encontrar una función interesante. La gran ventaja de las texturas 2D es que se pueden obtener con facilidad digitalizando texturas reales. Cabría la posiblidad de hacer lo mismo con texturas 3D digitalizando secciones de un material determinado pero esto es, por ahora, impracticable. Se trata, por tanto, de encontrar una función que simule adecuadamente lo que obtendríamos con tal serie de secciones.

Perlin utilizó una función estocástica, una función que generaba ruido aleatorio (*noise function)* que asignaba un escalar a un vector tomado como argumento de la función. Partía de una serie de puntos en el espacio dados por valores enteros, coordenadas *x*, *y*, *z* (que podía extenderse a una cuarta dimensión, *t*, en el caso de animaciones), una red de enteros (*integer lattice*). A cada punto de la red se le aplicaba la función que modificaba su color. Los puntos intermedios se interpolaban a partir de una red de vectores de gradiente precalculados.

El método introducido por Perlin ha conocido muchas versiones que buscan tanto crear mapas más interesantes como optimizar el procedimiento haciéndolo más rápido y más eficaz. En lugar de gradientes pueden utilizarse otras funciones y otros métodos de interpolar los valores. En el artículo citado, Perlin utilizaba variantes de la función, como la función *turbulence*, para simular la apariencia de mármol o funciones de onda para simular la apariencia de agua.

El artículo de Peachey partía de los mismos conceptos básicos, el interés práctico de crear texturas sólidas, y proponía diversos ejemplos utilizando funciones más simples o más conocidas que las utilizadas por Perlin, principalmente funciones sinusoidades proyectadas ortogonalmente o combinadas con otras funciones para simular madera u otros materiales.

El mármol, por ejemplo, se puede obtener mediante una función de ruido que distorsiona aleatoriamente una función de degradado que se aplica a una tabla de colores que simula una sección horizontal del mármol en la que se alternan los colores de las vetas. Al aumentar la frecuencia se forman distorsiones en la amplitud de las vetas que simulan la distorsión ocasionada en las rocas reales por efecto de la presión y los cambios de temperatura. La madera, otro ejemplo, se obtiene mediante una función cuadrática que ordena el espacio en cilindros concéntricos que simulan la sucesión de vetas. La ondulación y la variación de grosor de estas se puede crear modulando una función de ruido. Sin embargo, los resultados de estos dos ejemplos característicos resultan aún demasiado artificiales.

Los métodos procedurales requieren menos memoria que las imágenes almacena-

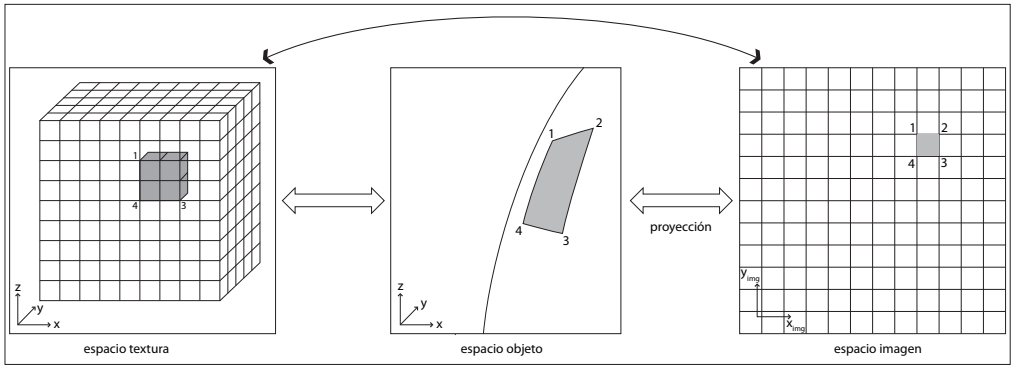


Figura 2.39 Procedural mapping.

das. Y, también, a diferencia de estas, su resolución depende tan solo de la precisión del cómputo. Pueden extenderse hasta el infinito sin repetición. Si la cámara se acerca a una textura generada por medio de una función de ruido, el detalle puede hacerse más complejo si el número de funciones utilizado en su generación es adecuado.

Por otro lado, el acceso a una textura procedural requiere más tiempo que a una textura almacenada en memoria, dado que debe generarse duante el proceso de cálculo de la representación, por lo que el tiempo de salida puede ser mayor. Otro inconveniente es que los procedimientos de *antialiasing* son más complicados. Dos de los métodos más efectivos de gestión de texturas, *MIP Maps* y *Summed-Area Tables*, pueden ser precomputados y almacenados para texturas basadas en imágenes. Pero debido a que las texturas procedurales se generan durante la representación, esto no es posible. El supermuestreo (*supersampling*) es una técnica efectiva tanto con texturas basadas en imágenes como con texturas procedurales, pero aumenta considerablemente el tiempo de cálculo.

Otra variante popular que complementa las anteriores es la función *noise cellular texturing*, introducida por Steven Worley (1996). El algoritmo se basa en la distribución aleatoria de una serie de puntos en el espacio. A partir de aquí se construye una función escalar basada en esta distribución. De este modo, se generan una serie de células yuxtapuestas cuyos planos de separación se computan por medio de un diagrama de Voronoi (una partición del espacio tal que todos los puntos de la célula están más cerca del punto base que de otro punto de la distribución). Mediante la adición de una serie de parámetros que afectan al color y el espesor de los límites, etc., se generan infinitos patrones que pueden simular de un modo bastante eficaz algunas texturas naturales.

Veremos ejemplos concretos de todos estos tipos de mapas procedurales en los últimos capítulos.

Procedural modeling de volúmenes difusos (varios, 1985)

El humo, el vapor, la niebla, las nubes… Todos estos fenómenos, que se perciben como patrones de variaciones del color, también son debidos a propiedades materiales. Pero, como ocurre con el relieve de las texturas, la simulación visual de estos casos implica procedimientos que están en frontera entre lo que es simulación geométrica y simulación de materiales.

Este tipo de fenómeno se puede modelar por medio de una técnica denominada, en general, modelización procedural de volúmenes (*volumetric procedural modeling*), o de gases (*procedural modeling of gases*), o de nubes (*volumetric cloud modeling*), para el caso específico de las nubes. Pero son técnicas de modelado que difieren radicalmente de la modelización geométrica corriente que se basa principamente en facetas planas triangulares. Y la simulación de los colores materiales se integra en el propio proceso de modelado por lo que estas técnicas dan lugar a problemas de integración con el flujo de trabajo habitual.

La modelización procedural de volúmenes se basa en funciones de densidad de volúmenes, *VDF* por sus siglas en inglés (*volume density functions*). Estas funciones se han utilizado por varios investigadores, como una extensión natural de las funciones de Perlin para generar texturas sólidas, principalmente por David Ebert, a partir de 1990, que es también el editor principal de una de las mejores obras de referencia de métodos procedurales de generación de texturas (véase Ebert en las referencias).

Hay otros trabajos anteriores en esta misma línea que ya he citado, como el uso por Gardner (1985) de composición de funciones sinusoidales para simular nubes. Pero el principal inconveniente de estos métodos es que, como decía, no son realmente 3D lo que limita considerablemente el rango de posibles aplicaciones. Los trabajos posteriores de Ebert han mejorado

substancialmente este tipo de representaciones. Ebert utiliza el término *Solid Space* para englobar las texturas sólidas, hipertexturas y funciones de densidad de volumen, en un mismo marco general. Un espacio sólido sería un espacio 3D asociado a un objeto de tal modo que permita especificar atributos adicionales para ese objeto. El uso más simple es, como ocurre con las texturas sólidas que hemos visto en el apartado anterior, para asociar un color determinado a todo punto del espacio asociado. En general, consiste en la aplicación de una función *n*-dimensional a un punto tridimensional (o a un punto cuatridimensional si queremos animar el objeto).

En el caso de la modelización de gases (humo, vapor) se utilizan funciones de densidad de volumen basadas en funciones de turbulencia, una variante de las funciones de ruido. Ebert (véase Ebert, 2003, p. 211) utiliza una interpolación trilineal sobre una malla regular (en los ejemplos descritos, de 64 x 64 x 64). Sobre esta malla se aplica una función *noise* y una función *turbulence*. Las funciones básicas que dan la forma primaria del gas dependen de los diferentes casos. La función puede ser tan simple como una función exponencial con valores diferentes para el exponente (cuanto mayor sea el valor del exponente mayor será el contraste local de los patrones resultantes). Esto puede combinarse con funciones de otros tipo, tales como funciones sinusoidades que pueden dar lugar a variaciones tales como finas "venas" en el gas.

En la práctica, como se verá en los capítulos siguientes, el procedimiento consiste en escoger una forma básica, tal como un cubo, una esfera, un cilindro, un cono, etc., que actúan como objetos contenedores iniciales, y aplicarles un algoritmo de este tipo que creará patrones difusos debido a las características de las funciones implicadas y al sistema de representación que, como también se verá en lo que sigue, debe ser específico para este tipo de objetos.

En el caso de las nubes las dificultades son mayores pues la simulación de la forma debe ser lo suficientemente precisa como para dar cuenta de variedades familiares como los tipos denominados *cúmulos*, *cirros* o *estratos* y variantes intermedias. Por otra parte, estas formas básicas varían con el viento, la altura, el clima, la hora del día. La altura, por ejemplo, hace que las nubes formadas por encima de los 5.000 metros (cirros, cirroestratos) sean más delgadas y más tenues y estén compuestas predominantemente de cristales debido a las altas temperaturas. Las situadas en posiciones intermedias, entre los 2.000 y los 5.000 metros (altocúmulos) son más esponjosas y ondulantes debido a que predominan gotas de agua. Las situadas más cerca del suelo, por debajo de los 2.000 metros (estratos, estratocúmulos) se distribuyen en capas horizontales. Por otro lado, las variaciones de luminosidad son más sutiles y, por añadidura, las nubes tienen sombras propias muy características.

Por todas estas razones la simulación de nubes ha sido un desafío para los investigadores. Algunas de las primeras soluciones que ya se han mencionado se debieron a Gardner (1984) por medio de funciones sinusoidales. Pero tenían limitaciones importantes. El método propuesto por Ebert en 1997 conseguía resultados muy efectivos, que luego se han incorporado a varios sis-

Figura 2.40 Nubes simuladas con métodos procedurales por medio del programa Terragen variando diferentes parámetros de densidad, altura, espesor, etc.

temas comerciales, algunos relativamente populares y sencillos, por medio de una combinación de métodos. Se basaba en el uso de funciones implícitas (*blobs*, *metaballs*), métodos procedurales con funciones de turbulencia y *volume rendering*. Por ejemplo, para simular una nube de tipo cúmulus, Ebert utilizaba 9 esferas implícitas de radios variables y funciones de fusión que daban la forma básica de la nube. Esta forma básica se refinaba por medio de funciones de turbulencia que añadían más detalle y permitian variar la densidad, así como modificar la mayor o menor suavidad de los bordes. Por último, dado que los sistemas de representación corrientes no pueden procesar este tipo de formas, se recurría a métodos alternativos de representación.

Este último problema es común a la simulación de nubes y de gases. Dado que todos los sistemas de *rendering* se basan en el cómputo de facetas triangulares con vértices que especifican exactamente su posición en el espacio y vectores normales que especifican exactamente su orientación, es evidente que estos sistemas no son adecuados para representar gases. Se requiere, por tanto, una técnica de representación alternativa. Esta técnica es *volume rendering*, una forma especial de representación que se utiliza ampliamente en ciertas aplicaciones, principalmente en medicina, para obtener representaciones del interior de órganos o funciones de distribución 3D de diversos tipos. En el anteúltimo apartado de este capítulo, sobre técnicas generales de *rendering*, se dan más detalles sobre esta variante.

Figura 2.41 Hipertexturas.

Hipertexturas (Perlin y Hoffert, 1989)

Como ya hemos visto, las texturas se pueden generar también por alteraciones de la geometría local. Y también hemos visto como uno de los primeros modelos eficaces, tanto desde el punto de vista teórico como práctico, fue el modelo de Cook y Torrance que simulaba la geometría local por medio de una función de distribución que computaba la frecuencia y las características de formas simétricas en "V" cuyo efecto sobre la reflexión podía tomarse en cuenta sin necesidad de modelar físicamente esta geometría.

Esta alteración virtual del entorno puede llevarse más lejos hasta abarcar los casos en que una región envolvente en torno a la superficie ideal, de espesor cero, puede ser modelada por medio de funciones que describan situaciones más complejas.

Ken Perlin desarrolló las ideas que había publicado en su famoso artículo de 1985, tres años más tarde, en un nuevo artículo publicado junto con Eric Hoffert (Perlin, K.; Hoffert, E., 1989) y que supuso el inicio de un método eficaz para generar texturas complejas. Es una extensión de los métodos procedurales utilizados para crear texturas sólidas. Pero la diferencia es que la evaluación se lleva a cabo en torno a una región volumétrica, en torno a la superficie del objeto en lugar de exclusivamente sobre la superficie del objeto. El resultado es, como dicen los autores, una forma-textura (*shape-texture*) o hipertextura (*hypertexture*) que permite sintetizar la apariencia de texturas que se dan en casos especiales: pieles, pelo, hierba, llamas, fluidos o materiales erosionados.

El método se basa en modelar objetos por medio de distribuciones de densidad. Se

formulan una serie de formas básicas que incorporan una región sólida y una región indeterminada. A esta región indeterminada se pueden incorporar todo tipo de funciones que permiten manipular su volumen. Y, por añadidura, se pueden utilizar operadores booleanos que combinen los diferentes efectos.

Es una región que se puede considerar "como maleable, en la que el usuario puede empujar o estirar o retorcer o deformar de otros modos materia simulada de un modo controlable" (ob. cit., p. 254 de las actas del congreso). Es decir, se parte de una función de densidad de objeto (*Object Density Function*), *D*(*x*), definida en el entorno [0,1] que describe los puntos situados en este entorno blando (comprendidos entre los límites 0 y 1) y una función de modulación de la densidad, DMF (*Density Modulation Function*), que se utiliza para modular la anterior, de tal modo que la aplicación sucesiva de esta función de modulación, con parámetros convenientemente escogidos, dé lugar a hipertexturas que pueden simular diferentes tipos de objetos. Estas funciones se dan según variantes específicas: *bias* (que se utilizan para empular o estirar la densidad del objeto), *gain* (para hacer el gradiente de densidad más suave o más agudo), *noise* (para introducir variaciones aleatorias), *turbulence* (para incrementar el efecto de las funciones anteriores), etc.

Por medio de variaciones de esta función se obtienen geometrías alteradas abstractas pero en las que se pueden encontrar similitudes con fenómenos reales. En el artículo citado se presentan ejemplos de simulación de esferas deformadas localmente con ruido, de fuego, de materiales semitransparentes con variaciones aleatorias de la refracción, de materiales erosionados (obtenidos mediantes combinaciones booleanas de prismas y esferas). Y de pelo y pieles. En el caso del pelo (que se puede extender a hierba, etc.), se utilizan tres funciones combinadas (*noise*, *bias* y *gain*) para modelar la longitud y el espaciado. Luego se suporpone otra variante de la función de *noise* que crea rizos mediante torcimientos aleatorios más o menos suaves. Ejemplos en color de estas primeras hipertexturas pueden encontrarse en las láminas en color de la obra clásica de Foley *et al*. (1990). En los capítulos sobre técnicas de este libro también se dan algunos ejemplos adaptados a la simulación de hierba.

Uno de los principales problemas que se abordaron es el relativo a la representación (*rendering*), pues los objetos resultantes no tiene caras bien definidas y por consiguiente no se pueden procesar por los métodos corrientes que se basan en el cómputo de normales asociados a facetas poligonales. Perlin y Hoffert proponen utilizar técnicas propias de lo que se conoce en la literatura especializada como *volume rendering*, que he mencionado en el apartado anterior y de lo que se dirá algo más al final de este capítulo.

Perlin y Hoffert usan una variante de estas técnicas para computar sus hipertexturas, que denominan *ray marching* y que, como en los métodos clásicos de *ray tracing*, envía un rayo desde cada píxel de salida hacia el modelo. Los rayos que penetran en el modelo se computan a intervalos determinados para obtener valores adecuados que se integran en un resultado final, que es realmente notable pese a la relativa indefinición de la geometría que se está computando. Con todo, el proceso era prohibitivamente lento en la época en que se propuso esta técnica. Pero con el crecimiento constante de la potencia de los ordenadores ahora es posible aplicarlo en programas especializados que funcionan con PC tal como veremos en la segunda parte y tal como cualquier espectador que haya visto películas como *Monsters* (Pixar, 2001), que incluyen pioneras y espectaculares simulaciones de pelo, habrá podido apreciar por su cuenta.

Texturas naturales generadas con fractales

Simular la textura de amplias extensiones de terreno es prácticamente imposible por medio de proyecciones de imágenes digitales. Una alternativa que se ha probado con creciente

eficacia, para los casos en los que el modelo no debe reproducir un terreno real con precisión, es el uso de fractales.

Los fractales son, en la actualidad, una técnica tan conocida que bastarán algunas notas para resumir lo que más nos puede interesar. El término y las ideas principales fueron introducidos por Benoit Mandelbrot en 1975 (véase Mandelbrot, 1975, 1982) y se han aplicado desde entonces a todo tipo de disciplinas. En la actualidad se utiliza principalmente para referirse a objetos geométricamente complejos que exhiben autosimilaridad a diferentes escalas y que pueden generarse artificialmente por algoritmos recursivos. Una de las razones del interés que han despertado los fractales es que hay muchos objetos naturales que también exhiben muchas de sus características. Entre ellos, y de un modo notorio, las ramas de los árboles, los ríos, las fomaciones rocosas o los rayos.

Precisamente por esta razón se han utilizado extensamente para simular objetos naturales por medio de procedimientos artificiales. En el caso de las texturas destinadas a ser vistas a corta distancia, de simulación de texturas con patrones singulares, como ocurre con la madera o el mármol, los resultados son aún insuficientes. Pero en el caso de texturas más amorfas o en el caso de texturas y objetos que vayan a ser vistos a distancia, como ocurre con los paisajes que incluyen montañas, o vegetación o texturas de rocas, etc., los resultados son notables.

Algunos de los ejemplos más espectaculares de fractales los ha desarrollado F. Kenton Musgrave, que trabajó durante años con Mandelbrot y es uno de los coautores de la obra citada de Ebert (2002). Merece la pena ver las imágenes incluidas en los capítulos 14 ("A brief Introduction to fractals"), 15 ("Fractal solid textures") y 16 ("Procedural fractal terrains") de esta obra. Musgrave comenzó a trabajar con Mandelbrot en Yale en 1987 y presentó su tesis doctoral "Methods for Realistic Landscape Imaging" en 1993 en la que incluyó una serie de artículos notables publicados durante los años de trabajo con Mandelbrot. También hay que citar otra obra que apareció durante este período, la de Heiz-Otto Peitgen (1988) aunque solo incluye imágenes abstractas, poco aplicables a la simulación de materiales. Con posterioridad a estas fechas han aparecido docenas de trabajos notables que pueden buscarse por internet.

Algunos de los conceptos básicos que convendrá recordar son los siguientes.

Un fractal tiene una determinada *dimensión fractal* donde la parte entera corresponde a la dimensión de la base euclídea y la parte fraccionaria, a la densidad de ocupación de la dimensión superior. Por ejemplo, una dimensión de 2,3 indica que la dimensión de base es 2D, es decir, un plano. Pero el incremento de 0,3 indica que se está ocupando en mayor o menor grado la dimensión superior, es decir, que el plano mostraría protuberancias repetitivas de una cierta densidad, tanto mayor cuanto más alto sea el valor fraccionario.

Los fractales se generan a partir de una determinada *función de base*. Esta función de base proporciona la forma básica que se irá repitiendo con variantes a diferentes escalas. Una eleción adecuada de la función de base es fundamental para conseguir una determi-

Figura 2.42 Montañas generadas con fractales por medio del programa Terragen variando diferentes parámetros de escala, octavas y funciones asociadas.

nada configuración formal. Por ejemplo, una función utilizada extensamente en texturas sólidas es la función *Noise*, que ya hemos visto y que permite crear y escoger distribuciones aleatorias simples que pueden modificarse y hacerse más intensas por medio de fractales. Otra función utilizada ampliamente es la función *fBm* que simula el movimiento browniano (*Brownian Motion*, BM) pues se ha encontrado que muchas texturas naturales exhiben un aspecto similar al generado por este tipo de distribución.

Otro concepto fundamental es el de *lacunarity* (lagunaridad o lacunaridad: no hay traducción corriente). Puede entenderse la lacunaridad como una medida de la distribución de los espacios vacíos, de los huecos, de una estructura fractal. Si el valor de la lacunaridad es alto, quiere decir que hay muchos huecos, y a la inversa. Es una noción propuesta por Mandelbrot (1982) para caracterizar estructuras fractales con la misma dimensión fractal pero con diferente textura. Puede considerarse como una medida de la heterogeneidad de la textura o estructura fractal resultante.

El valor de la lacunaridad determina la cantidad de cambios en escala a través de los sucesivos niveles de detalle. Estos ciclos de repetición se denominan *octavas* y el número de octavas de un fractal es un modo de referirse al número de veces que la función base se repite generando nuevas distribuciones. Con 1 octava solo se utiliza el patrón básico. Con 2 octavas y una lacunaridad de 2, un patrón mitad se superpone al patrón básico. Con 3 octavas, otro patrón mitad del anterior (un cuarto del inicial) se superpone de nuevo y así sucesivamente. Aunque a menudo se piensa erróneamente que el número de octavas de un fractal es infinito, lo cierto es que en la mayoría de los casos no suele ir más allá de 3. De hecho, todos los fractales de la naturaleza tienen limitaciones de banda, es decir, que solo son fractales dentro de un cierto rango de octavas.

Los fractales se han utilizado con éxito para simular, como dice Musgrave, "los cuatro elementos de los antiguos: aire, fuego, agua y tierra". Pueden encontrarse ejemplos y código para estos casos en el capítulo 15 de la obra citada de Ebert, Musgrave, Peachey, Perlin y Worley (2003). Y pueden verse las imágenes incluidas en este libro y algunas más en http://www.kenmusgrave.com/, además de su *Mojoworld*: un proyecto iniciado hace ya unos cuantos años para construir planetas procedurales.

Los ejemplos de agua, tierra y paisajes son espectaculares y, afortunadamente, se pueden utilizar con facilidad desde varios programas como también veremos en los capítulos siguientes. El fuego y las nubes (aire) pueden simularse más eficazmente con otros métodos, principalmente con sistemas de partículas en el caso del fuego y con los métodos que se han descrito en el apartado anterior sobre simulación de volúmenes difusos.

Figuras planas simuladas con *sprites* y *billboards*

Una alternativa simple pero efectiva para simular figuras es el uso de lo que se denomina *sprites* o, más recientemente, *billboards*, en la jerga de los juegos de vídeo.

Un *sprite* ("duendecillo" en inglés) es un tipo de mapa de bits que se usa para simular figuras de un modo ágil. Se utilizan desde 1980 aproximadamente, principalmente en juegos 2D, tanto para simular figuras como para incluir elementos de interfaz. Los *sprites* son generalmente de pequeñas dimen-

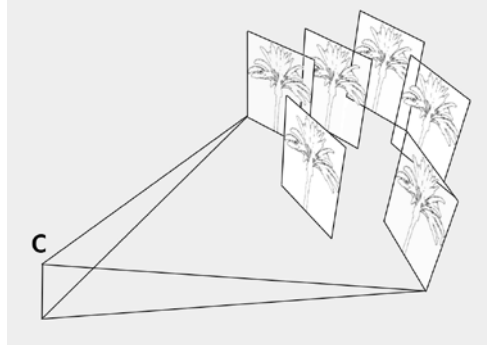


Figura 2.43 Billboards.

siones e incluyen un fondo transparente (un canal alfa) que actúa como una máscara y permite simular figuras recortadas. Eran recursos que se combinaban con la imagen almacenada en el *frame buffer* de tal modo que podían moverse con independencia de esta imagen. Un cursor es, de hecho, un ejemplo elemental de *sprite*. En los primeros videojuegos permitían animar figuras simples sobre un fondo fijo, de modo similar a lo que se hacía en los dibujos animados tradicionales para animar los personajes principales sin tener que redibujar el fondo.

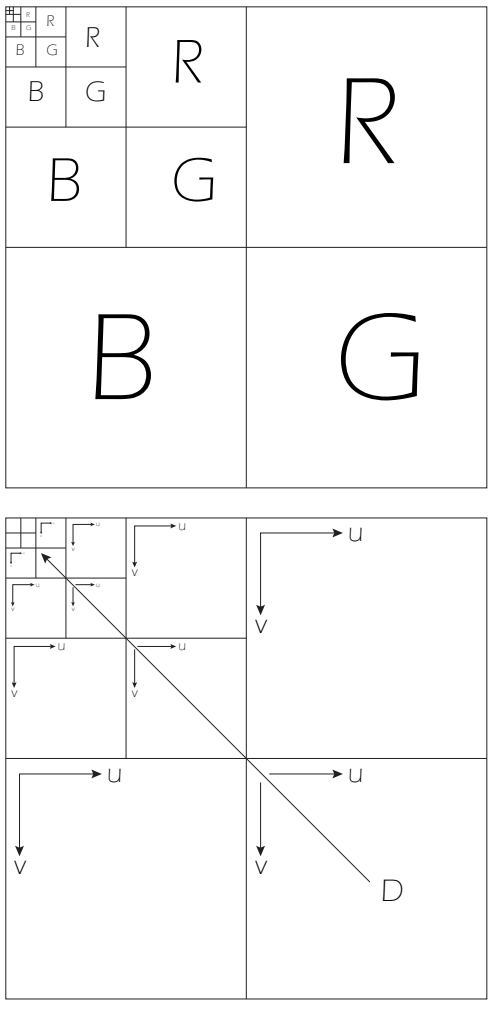


Figura 2.44 Mipmaps.

Un *billboard* es un *sprite* y algo más. El término deriva de los grandes paneles de anuncios, utilizados sobre todo en autopistas de Estados Unidos (*billing boards*). La diferencia es ambigua porque también se puede encontrar el término *billboard sprite*. Pero se utiliza a menudo, desde hace algunos años, para designar a simulaciones de figuras tridimensionales por medio de una imagen plana, situada en un espacio 3D, que incorpora un fondo con un canal alfa de modo que la figura aparece recortada y está ligada a un controlador que hace girar el plano de tal modo que siempre quede enfrentado a la cámara. Puede decirse que las diferencias son más bien históricas dado que los *sprites* tenían un uso más limitado, restringido generalmente a juegos 2D, y los *billboards* se utilizan para escenas 3D y permiten incorporar comportamientos colectivos.

Los *billboards* se utilizan para simular figuras sobre un escenario pero también en simulación de partículas y en sistemas de multirresolución, de tal modo que a corta distancia el elemento sea realmente 3D pero a partir de un distancia determinada se substituya el 3D por un *billboard* para optimizar recursos de computación.

Variaciones debidas a la distancia. *Mipmaps*. Multirresolución

Como hemos visto en el capítulo anterior, una diferencia crucial entre los materiales reales y los virtuales es que los primeros cambian de aspecto con la distancia, tal como se muestra en la espectacular película dirigida por los Eames *Potencias de 10* a la que se hacía referencia en ese capítulo. El aspecto, los detalles de un material, cambian a medida que nos acercamos o nos alejamos y aún más si recurrimos a microscopios o telescopios.

Sin embargo, esto no ocurre así con los materiales virtuales. Lo que caracteriza, por el contrario, a los materiales virtuales es que su aspecto se degrada con la distancia, en

ambas direcciones. Si nos acercamos, los detalles se convierten en píxeles: rectángulos de un color único. Si nos alejamos, en muchos casos nos encontraremos con patrones artificiosos que revelan que una textura aparemente única se ha obtenido por copia de texturas más simples. O bien perderemos sutiles efectos que dependen del modo en que grandes zonas reflejan el color de diferente modo al variar la distancia de observación.

Por otro lado, aparece un problema técnico adicional. A determinada distancia necesitamos texturas de alta resolución, adecuadas para esa distancia. A una distancia más corta esa resolución puede ser innecesaria o inadecuada, con lo que estamos desperdiciando recursos.

Una solución parcial a este problema pero que ha pasado a formar parte de los recursos corrientes de los videojuegos y los programas de animación y se ha incorporado a los recursos básicos de las tarjetas gráficas, fue propuesta por Lance Williams en 1983 (Williams, 1982). Williams bautizó a su método *Mip Mapping*: MIP son las siglas de la frase latina *multum in parvo* (mucho en poco).

La técnica está basada en lo que se denomina genéricamente estructura piramidal de datos (*piramydal data structure*) que proporciona una estructura jerárquica que puede ser recorrida con rapidez. Hay varios tipos de estructuras de este tipo: árboles binarios, quads, octales, *n*-dimensionales (*binary*, *quads*, *oct*, *n-dimensional trees*) que se han utilizado en todo tipo de aplicaciones gráficas. Williams utilizó esta estructura para organizar los datos de tal modo que admitieran interpolaciones internas y externas.

Si tenemos una imagen dada por una matriz de muestras y a los dos parámetros propios del espacio de la imagen, *u*, *v*, se añade un tercer parámetro *D*, tal como se muestra en la figura 2.44, la modificación de este parámetro nos permitirá recorrer con facilidad y rapidez la estructura de datos. Este recurso se utililiza para filtrar versiones que tienen una cuarta parte del tamaño original. Para optimizar el recurso se utilizan siempre tamaños de imagen que son potencias de 2. Esta es una de las razones por las que se recomienda utilizar dimensiones de textura basadas en estos valores pues, en cualquier caso, se reconvertirán internamente a potencias de 2, con lo que evitamos esta conversión si ya les damos inicialmente un valor adecuado.

Si, por ejemplo, tenemos una imagen RGB de un tamaño original de 512 x 512, se genera una imagen de doble tamaño lineal (4/3 total) es decir, de 1024 x 1024. Cada cuarto se llena con el canal R, G o B de la imagen original y el cuarto restante se llena con versiones filtradas por un factor de 4. Este proceso se repite recursivamente hasta que el mapa se llena por completo con un último mapa de 1 x 1 píxel. El tamaño guardado es mayor pero el procesamiento es mucho más rápido pues solo se procesan los píxeles necesarios en cada caso.

Otra alternativa es utilizar texturas de diferente tamaño, ligadas a objetos también con diferente densidad geométrica, agrupar estos objetos y utilizar uno u otro en función de la distancia. Hay programas que permiten utilizar este recurso ligándolo a una determinada distancia o un determinado tamaño de salida de tal modo que, al cambiar esta distancia o este tamaño, se utiliza una u otra de las versiones agrupadas. Este recurso se denomina LOD (*Level of detail*) y permite gestionar estas substituciones. La utilización del recurso es sencilla y veremos algun ejemplo más adelante.

2.6 Otros métodos

Variaciones debidas al deterioro

Los diferentes modelos presentados hasta aquí se basan en propiedades de materiales que no tienen en cuenta la acción del tiempo, el desgaste sufrido por los materiales reales. Las imágenes que resultan de la aplicación de esos modelos, exhiben una precisión y una pulcritud características de los modelos virtuales generados por ordenador.

Esto puede estar bien en algunos casos pero, en otros, el resultado es una notoria falta de realismo, pues los materiales reales se modifican a medida que pasan los años por causas muy diversas. A menudo muestran manchas de todo tipo, provocadas por el roce, por accidentes o por el desgaste debido a reacciones diversas a nivel superficial. O bien el conjunto de la superficie cambia de color y a la textura natural se superpone otra textura debida a capas de materiales derivados, semitransparentes, cuyos colores y patrones se funden con los de capas inferiores.

Hay múltiples razones por las que puede ser necesario incorporar estos efectos a una simulación. Pero podemos resumirlas en una: la necesidad de obtener resultados más realistas debido a los requisitos de ciertas aplicaciones. Requisitos que pueden tener que ver con el arte, para resultar más convincentes, o con la ciencia, para representar mejor las propiedades variables de los materiales reales.

En cualquier caso, los métodos que permiten integrar estos efectos se pueden clasificar en dos grandes categorías: los que se aplican a nivel estrictamente superficial, con espesor cero, por medio de mapas 2D; y los que se aplican por medio de capas superpuestas a la superficie original, simulando de un modo más exacto lo que ocurre en la realidad.

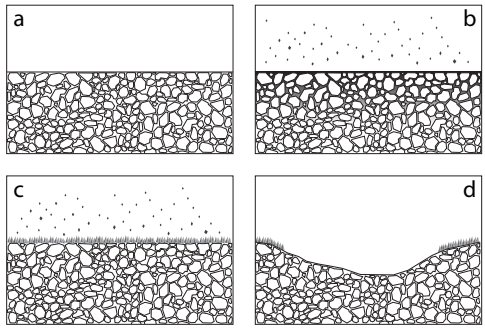


Figura 2.45 Erosión de la piedra: superficie original (a) atacada por la lluvia (b), lo que da lugar a la formación de una costra por cristalización de óxidos (c), que posteriormente se erosionan (d), lo que reinicia el proceso.

En el primer caso también podemos distinguir dos tipos de técnicas. La primera y más corriente, y que quedaría englobada dentro de las técnicas corrientes de *texture mapping*, es crear una imagen digital que incorpore el tipo de manchas o alteraciones que interesen, lo que se puede conseguir fotografiando texturas reales o modificando texturas reales con mapas superpuestos como veremos en la segunda parte. Todo lo que se requiere es una cierta habilidad artística para conseguir resultados convincentes y un conocimiento suficiente de técnicas básicas que veremos en el capítulo dedicado a ejemplos de aplicación de texturas. Pero conseguir buenos resultados puede ser trabajoso o puede requerir una habilidad con la que no siempre se cuenta.

De ahí que, la segunda, menos corriente, sea utilizar métodos automáticos, procedurales, de alteración de texturas dadas o de generación de patrones de manchas que puedan superponerse a texturas dadas.

Uno de los primeros y escasos trabajos que se pueden citar en esta última línea es el debido a Norman Badler y Welton Becket (Badler y Becket, 1990). Estos investigadores propusieron un algoritmo basado en fractales que iba unido a un “lenguaje natural” de especificación de parámetros que facilitaba al usuario la utilización del algoritmo. Como esta aplicación no se ha difundido, que yo sepa, podemos pasarla por alto y anotar alguna de las ideas que son interesantes y que pueden aplicarse por otros medios.

Las manchas algorítimicas pueden generarse por varios procedimientos corrientes tales como la aplicación de fractales o distribuciones gaussianas y combinarse y recombinarse entre sí. El tipo de manchas puede orientar la elección del método. Como veremos más adelante, estos procedimientos pueden incorporarse a varios programas de uso más o menos corriente, como Photoshop, 3ds Max o MapZone. Si se ajustan adecuadamente los límites de variación pueden obtenerse todo tipo de efectos, tal como se muestra en la figura 2.45 que es una de las muchas variaciones que analizare-

mos más adelante. Un análisis de la tipología de manchas puede orientar adecuadamente la elección de una u otra técnica.

En el segundo caso mencionado más arriba, la simulación por medio de la generación de capas de espesor distinto de cero, también podemos distinguir varios casos de interés:

a) La formación de pátinas en los metales, por reacciones químicas. Estas pátinas varían de color y espesor, y tardan más o menos tiempo en formarse según el entorno sea marino, rural o urbano.

b) La erosión de la superficie de piedras y otros materiales o la corrosión de los metales, que tienen el efecto contrario al anterior, penetrando en la superficie y alterando la coloración y la rugosidad de la textura propia del material.

c) El depósito de partículas diversas o de agentes extraños, como pueden ser las manchas de sales de color blanquecino debido a la cristalización de sales presentes en el material (eflorescencias) o a la aparición de hongos debida a humedades o la incrustación de insectos o partículas de otros materiales arrastradas por el viento.

d) La combinación de las causas anteriores.

Para no extender innecesariamente este tema que es relativamente secundario para las finalidades de este libro, me referiré solo a dos aportaciones interesantes que tendrían que ver con los dos primeros casos.

La formación de pátinas en los metales se aborda en un artículo presentado por Julie Dorsey y Pat Hanrahan (Dorsey y Hanrahan, 1996). Los autores se limitan a un caso particularmente interesante: el cobre. Cuando el cobre queda expuesto al exterior se forma una fina capa de color marrón que con el tiempo se va haciendo más rojiza debido a la formación de óxido de cobre. Sobre esta capa, que se forma con relativa rapidez, van creciendo poco a poco otras capas que incorporan sulfatos, nitratos y sales orgánicas y que van tiñendo el cobre de un tono verdoso característico, pero que se funde con los tonos rojizos de la capa inicial. El grosor de estas capas y el tiempo de formación depende del medio ambiente: se forma con mayor rapidez en medios urbanos, más lentamente en medios rurales y a una velocidad intermedia en medios marítimos.

El método propuesto por estos autores permite la creación de estas capas por medio de una serie de operadores (*coat*, que añade grosor a las capas, *erode*, que quita grosor, *fill*, que rellena con un determinado material, *polish*, que elimina el material que supere una determinada altura) que afectan al color, al acabado y al espesor de dichas capas. Estos operadores se combinan con un mapa fractal que hace variar el espesor con el tiempo, a partir de tres modelos de crecimiento (marino, rural o urbano). Los resultados que se pueden ver en color en el artículo citado son notables si bien resultarían indistinguibles de texturas 2D obtenidas a partir de fotografías digitales por lo que el interés de este trabajo es más científico o dirigido a futuras aplicaciones de análisis arquitectónico de superficies que adecuado para aplicaciones de simulación.

El desgaste de las piedras se aborda en otro artículo presentado por varios autores y encabezado también por Julie Dorsey (que estudió arquitectura y ciencias de la computación, hizo su doctorado en Cornell en 1993 y es profesora en la facultad de Yale desde 2002 (Dorsey *et al*., 1999).

En este artículo se busca, como en el anterior, reproducir el proceso real que, en el caso del desgaste de las piedras, es bastante complejo pues lo que tiene lugar no es solo un proceso de erosión, sino un complejo proceso que se resume en la figura 2.45: la piedra expuesta a las inclemencias del tiempo es atacada por la lluvia y el viento y por óxidos transportados por estos agentes que penetran en la piedra y se recristalizan formando una costra, un recubrimiento que se degrada y se erosiona dejando de nuevo expuesta parte de la superficie.

Para simular este proceso se recurre a tres componentes: un volumen alineado con la superficie; la simulación del flujo de agua y minerales a través de este volumen; técnicas

especiales de *rendering* (derivadas de técnicas de dispersión, de *subsurface scattering*) para simular este tipo de elementos que no admiten los procedimientos corrientes de representación basados en facetas poligonales. El volumen alineado con la superficie es una estructura de datos especial, una *slab*, como se denomina en el artículo, una pequeña región alrededor del borde externo de la superficie.

Volume rendering. Participating media. Ray marching

En muchas aplicaciones se necesita obtener representaciones 2D de un volumen sólido. Lo que se denomina, genéricamente, *volume rendering*, se utiliza desde mediados de la década de 1980 para visualizar las características internas de cuerpos previamente modelados con técnicas especiales. Se ha utilizado principalmente en medicina, para mostrar el interior de zonas del cuerpo a partir de la computación de las diferentes densidades de los tejidos o de las diferencias de temperatura. En estas aplicaciones, cada punto del volumen tiene asociado un determinado valor y la serie de estos valores define un campo escalar que se puede interpolar para dar lugar a superficies que definenen niveles y fronteras internas en el volumen. Por medio de aplicaciones específicas de *volume rendering* es posible visualizar estos campos escalares del modo más adecuado para cada caso. Las diferencias entre los tipos de valores y el modo en que se distribuyen han dado lugar a diversas variantes que han surgido en los últimos años.

Las técnicas de generación de estos datos se engloban genéricamente bajo la denominación de tomografía (del griego *tomos*, "corte", "sección") que se puede definir como la técnica de registro gráfico de imágenes de un cuerpo correspondientes a un corte o plano determinado. Hay varias técnicas de tomografía (rayos X, ultrasonido, emisión de positrones, resonancia magnética), entre ellas la tomografía computerizada (CT, por sus siglas en inglés, *Computer Tomography*). Los resultados tienen grandes ventajas con respecto a otras técnicas tradicionales pues permiten visualizar secciones del cuerpo desde cualquier punto de vista.

Además de en medicina también se utilizan en muchas otras aplicaciones (arqueología, geografía, astrofísica...) y, entre ellas, en ciencias de materiales. Por lo general, la información se almacena por medio de valores asociados a puntos del espacio distribuidos en una red regular. El conjunto de estos valores forma un campo escalar y lo que se requiere en todas las aplicaciones mencionadas es obtener representaciones adecuadas de este campo escalar.

Hay varias técnicas específicas de *volume rendering*. Las tres principales son, probablemente, la representación de elementos de volumen en un *bps*, un espacio binario particionado sistemáticamente (*rendering voxels in binary partitioning space*), el procesamiento secuencial de cubos (*marching cubes*) y el trazado secuencial de rayos (*ray marching*). Cada una tiene ventajas y desventajas. En el tipo de aplicaciones que nos interesan, la representación de gases, humo, nubes, etc., el método más utilizado, y que Perlin describe en su artículo de 1989 sobre hipertexturas, es lo que se conoce como *ray marching* y que describiré algo más extensamente a continuación como introducción a las técnicas concretas que veremos más adelante.

A diferencia de los métodos corrientes de simulación, en los que lo que se procesa es exclusivamente la superficie envolvente de un cuerpo, en estos métodos lo que se procesa es lo que está comprendido entre determinadas superficies. De este planteamiento derivan dos tipos de aplicaciones: las que consideran volúmenes globales y las que consideran volúmenes locales. En el primer caso, se procesa todo el espacio de una determinada escena por medio de una envolvente general. En el segundo caso, se procesa el espacio que está comprendido entre los límites geométricos de un objeto determinado. Lo primero da lugar a simulaciones de, por ejemplo, niebla y otros efectos atmosféricos

que abarcan toda la escena. Lo segundo da lugar a simulaciones de fuego, gases, humo o hipertexturas.

Como decía, en las aplicaciones de simulación más generales, como las que se utilizan principalmente en arquitectura, las técnicas más utilizadas son las que consideran los objetos representados como constituidos por superficies envolventes. Y computan sus valores a partir de técnicas que se basan principalmente en el envío de rayos trazadores (*ray tracing*) que solo tienen en cuenta estas superficies. Por esta razón, resulta más sencillo adaptar las técnicas de *volume rendering* a este punto de partida: representaciones de superficies, no de volúmenes y envío de rayos.

Ray marching es una técnica específica de *volume rendering* que puede considerarse como una variante de *ray tracing*, y que consiste en enviar rayos hacia el objeto desde la cámara y computar las intersecciones del rayo con los objetos y los valores correspondientes a los atributos de las superficies con que se encuentre el rayo en su camino. Pero, en *ray marching*, a diferencia de *ray tracing*, cuando el rayo encuentra una superficie, no se refleja o se refracta, sino que continúa su camino a pasos predeterminados. Y en cada paso, evalúa el valor de una función de distribución general del volumen para, a partir de ahí, calcular la densidad, el color, la opacidad, la iluminación y el sombreado de la porción de volumen correspondiente. La denominación *ray marching* se utiliza para enfatizar esta diferencia con *ray tracing*, aunque también se encuentran otras denominaciones como *volume ray casting* o *volume ray tracing*. Este último término, como decía, puede resultar confuso pues las técnicas habituales de *ray tracing* son bastante distintas, razón por la que muchos prefieren evitar tal denominación.

En cualquier caso, es una técnica cuyos algoritmos se basan en el procesamiento de píxeles, lo que la distingue netamente de las técnicas habituales de *volume rendering*, que se basa en el procesamiento de objetos, de los objetos incluidos en el volumen que se quiere representar. Se utiliza principalmente para integrar la iluminación y otros efectos, a lo largo de una determinada distancia, en el cálculo del volumen. Un ejemplo característico es el de un rayo de luz que penetra en una habitación oscura, y con partículas de polvo, por una ventana. El rayo de luz es visible porque las partículas de polvo se iluminan por el rayo y reflejan la luz.

Para calcular este efecto es necesario recurrir a métodos más sencillos de los teóricos pues un cálculo preciso sería impensable. El procedimiento característico seguido por este algoritmo es evaluar una serie de puntos a lo largo del recorrido teniendo en cuenta su posición con respecto al volumen en que están incluidos.

La figura 2.46 muestra un esquema de este proceso. El rectángulo representa una vista en planta del volumen de cálculo. Cuando un rayo de luz penetra en este volumen, se seleccionan una serie de puntos a lo largo de su recorrido por el interior del volumen y se evalúa la iluminación que recibe, que puede estar condicionado por obstáculos internos, tal como también se ilustra en la figura en la que algunos de los puntos más lejanos no reciben luz porque esta queda interrumpida por un elemento interno.

Los puntos se pueden escoger a intervalos regulares o a intervalos que dependan de la distancia. El número de puntos es uno de los

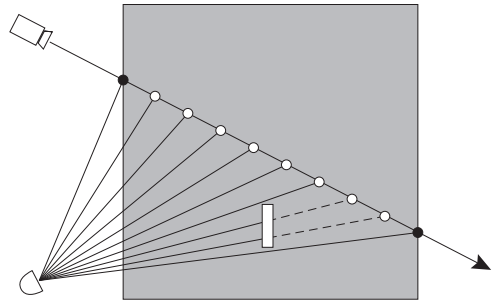


Figura 2.46 Ray marching. *Los puntos del volumen procesado, cuya sección horizontal se muestra tramada, se procesan a lo largo de un rayo enviado desde la cámara, a intervalos dados por la configuración y computando su posición y su iluminación.*

parámetros principales que habrá que ajustar indirectamente, por lo general especificando valores mínimos y máximos. Por otro lado, el medio encerrado por el volumen puede considerarse como uniforme o no uniforme, y su grado y tipo de variación también pueden especificarse.

Por lo que respecta a las características de los efectos de dispersión, también hay que considerar cuatro propiedades: *Emisión*: las partículas pueden emitir luz propia si sus propiedades incluyen características de autoiluminación, lo que aumentará la radiancia. *Absorción*: las partículas absorben iluminación, el caso más corriente, lo que disminuirá la radiancia. *Dispersión externa*: las partículas pueden reflejar la luz hacia el exterior tal como ocurre con la luz de la atmósfera, lo que disminuirá globalmente la radiancia. *Dispersión interna*: las partículas reciben luz que reflejan internamente, lo que aumenta la radiancia en una determinada región tal como ocurre con los fenómenos de SSS (*SubSurface Scattering*). La absorción y la dispersión externa, por tanto, hacen que disminuya la radiancia mientras que la emisión y la dispersión interna hacen que aumente.

Por lo que respecta a los tipos de dispersión, hay que distinguir entre medios que favorecen la dispersión en una u otra dirección, lo que depende de varios factores que, en general, están ligados a los tipos y tamaños de las partículas. El punto de partida de estos análisis son los modelos propuestos por el físico inglés Lord Rayleigh (1842-1919) y por el físico alemán Gustav Mie (1869). En el campo de los análisis más recientes y de las técnicas que se verán más adelante, tiene particular importancia el modo de computar la intensidad (p) de la dispersión en una determinada dirección (θ) o lo que se conoce como la fase característica de un medio dispersivo.

El modelo de Rayleigh está basado en partículas de tamaño inferior a la longitud de onda media de la luz y se caracteriza por una distribución isotrópica. La longitud de onda afecta en este caso a la dispersión. Un ejemplo característico es el azul del cielo durante el día que es debido a un predominio de la dispersión de ondas cortas (frecuencias altas), un color que va cambiando al atardecer pues la dispersión de estas ondas cortas es absorbida por la atmósfera en ángulos que ya no son visibles para nosotros y quedan las ondas medias o largas, lo que resulta en una coloración más cálida. La intensidad de la dispersión en estos casos viene dada por la fórmula:

$$p\,(\theta) = 3\,/(16\,p\,(1 + \cos^2\theta))$$

El modelo de Mie está basado en partículas de tamaño igual o mayor que la longitud de onda media de la luz y se caracteriza por una distribución anisotrópica, con un predominio de la dispersión hacia delante. A diferencia del modelo de Rayleigh, en este caso la longitud de onda no afecta a la dispersión y el color es predominantemente neutro, como ocurre en la niebla debida a causas naturales o artificiales, como la polución, el humo o las nubes.

Los astrónomos americanos Louis G. Henyey (1910-1970) y Jesse Greenstein (1909-2002) desarrollaron un modelo más preciso y basado en datos experimentales que permitía calcular la intensidad de la dispersión para un medio determinado en función de tres parámetros principales. El modelo de Henyey y Greenstein, presentado en un artículo de 1941, se ha reelaborado por Christophe Schlick en otro artículo de 1993 (con otros autores, véase las referencias) en una versión simplificada para adaptarlo a la simulación digital y es la base de las técnicas que veremos más adelante. La fórmula general de Henyey y Greenstein es:

$$p\,(\theta) = (1/4\pi)\,((1 - g^2)/(1 + g^2 - 2g\cos\theta)^{1,5})$$

El parámetro g, en esta fórmula, computa la intensidad relativa en una determinada dirección, en relación con el ángulo de incidencia de la luz, tal como se muestra en la figura 2.47.

Valores positivos indican que la dispersión es dominante en la dirección de la luz y valores negativos indican que la dispersión es dominante en dirección opuesta a la luz,

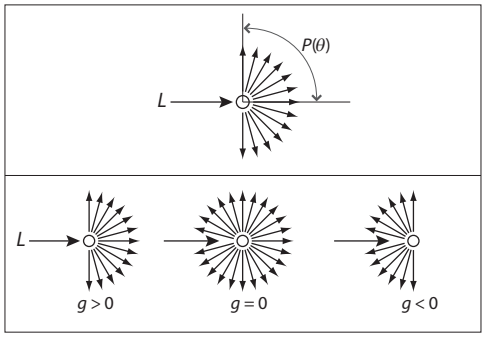


Figura 2.47 Dispersión. La función de fase (a) y los parámetros g1, g2 *(b).*

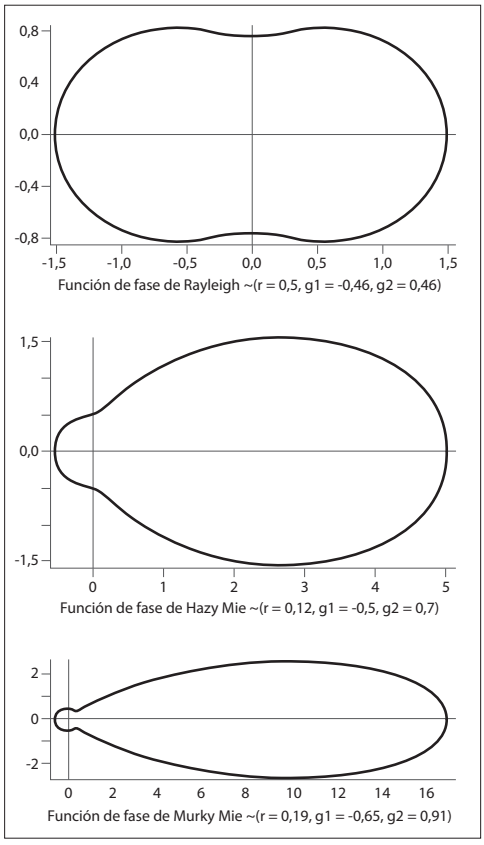


Figura 2.48 Dispersión. Representación y parámetros de los tipos de dispersión de Rayleigh, Hazy Mie y Murky Mie.

mientras que un valor nulo indica que la dispersión es igual en todas direcciones. Los dos primeros casos son propios de la dispersión anisotrópica y el tercero, de la dispersión isotrópica.

La fórmula original de Henyey-Grenstein está referida a un hemisferio, es decir, a la dispersión en una u otra dirección. Para facilitar su uso en aplicaciones de simulación se ha substituido por una función de fase única con dos parámetros, *g*1, *g*2 y un tercer parámetro, *r*, que los combina. Si se denomina la fórmula anterior HG (Heyley-Greenstein) la función de dos términos puede resumirse así (esta reformulación es debida a Schlick):

$$HG(g_1) + (1 - r)\, HG(g_2)$$

Como puede verse en está fórmula, si $r = 1$ el segundo término se anula y solo cuenta el valor *g*1. Y si $r = 0$, el primer término se anula y solo cuenta el valor *g*2. En general, si el valor de *r* está comprendido entre 0 y 1, la fórmula será una combinación lineal de los dos términos. Corrientemente *g*1 tiene valores negativos y *g*2 positivos. En la segunda parte veremos como se aplican estos términos en la práctica, por medio de *shaders* especiales que los incorporan a sus parámetros.

Los valores adecuados de *g*1 y *g*2 para diferentes medios se dan también en el artículo citado de Schlick. Los diagramas de la figura 2.48 muestran una representación de la función sin simplificar, junto con los valores de la formulación simplificada para los tres casos que fueron estudiados en su día por Rayleigh y Mie. Una configuración *r* - *g*1 - *g*2 de 0,50, -0,46, 0,46 genera el efecto de dispersión Rayleigh. Una configuración de 0,12, -0,50, 0,70 genera el efecto Hazy Mie. Y una configuración 0,19, -0,65, 0,91 genera el efecto Murky Mie.

Representación de sistemas de partículas

Tanto los sistemas de partículas regulares como los estructurados requieren métodos especiales de *rendering* debido al gran núme-

ro de elementos implicados.

En los primeros se ha recurrido a varios métodos, principalmente al primero de los que siguen. Los segundos plantean dificultades mayores. En su primer uso Reeves recurrió a sistemas de representación más o menos tradicionales. Se han utilizado algoritmos probabilísticos para calcular los valores de los píxeles de la imagen final. Posteriormente han ido apareciendo técnicas diversas en las que, por lo general, se utilizan *billboards* o representaciones directas 3D, es decir, el tercero y cuarto de los métodos que se resumen en el párrafo siguiente. Las sombras plantean dificultades específicas. Para calcular las sombras propias de las ramas o las hojas de los árboles, por ejemplo, se recurre a envolver virtualmente cada árbol con un cono virtual. Desde cada punto que representa un elemento del árbol se envía un rayo hacia la luz y se computa su distancia al cono envolvente. Cuanto mayor sea esta distancia, mayor será la probabilidad de que el elemento esté en sombra pues otros elementos interpuestos le arrojarán sombra. Para calcular las sombras arrojadas por otros árboles vecinos se envían rayos desde la parte superior del árbol hacia la luz. Antes de representar una partícula se comprueba si está por debajo (con lo que la probabilidad de que reciba sombras será tanto mayor cuanto más abajo esté) o por encima de esta línea (con lo que no recibirá sombras).

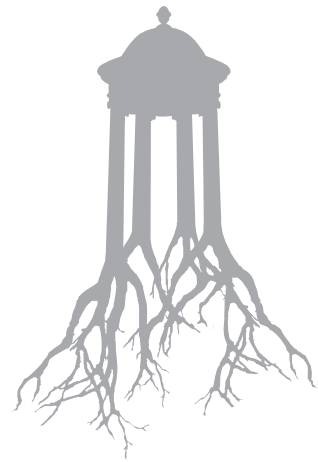
Las técnicas de representación principales son las siguientes (puede encontrarse un análisis de estos métodos en Walsh, 2005):

a) Primitivas simples sin texturas. Es el sistema utilizado en los primeros casos debido a las limitaciones de los ordenadores aunque ahora se utiliza muy pocas veces. Las partículas se representan como puntos simples de color que se almacenan en la memoria temporal del sistema de representación y luego se componen con la imagen global. En general, se prescinde del orden para simplificar más el proceso, haciendo que los colores de cada partícula sumen su contribución si van a caer en el mismo píxel. Sin embargo, en algunos casos el orden de representación puede afectar al resultado por lo que se requiere algún método de ordenación previo, lo que puede complicar el proceso.

b) *Sprites* puntuales. En estos casos las partículas pueden incorporar más detalles gracias a esta técnica que ya se ha descrito anteriormente.

c) *Quads* texturizados o *Billboards*. Se utilizan planos simples (*quads*) con texturas que incorporan canales alfa para recortar los contornos y que se pueden mover y girar con facilidad, de modo que se pueden reorientar automáticamente para mirar hacia la cámara.

d) Objetos poligonales. En este caso, la particula es un objeto 3D que se representa con procedimientos corrientes, lo que implica un gran coste computacional, por lo que se han utilizado raramente aunque es de suponer que lo serán más en el futuro a medida que aumente la potencia de computación corriente.



TÉCNICAS

→3

Recursos básicos

3.1 Las tuberías de la representación

La base física. Características y evolución de las tarjetas gráficas y la GPU

Las tarjetas gráficas han experimentado una evolución tan espectacular en los últimos años, y lo siguen haciendo a un ritmo tan vertiginoso, que resulta difícil seguir esta evolución acelerada y aún más intentar resumir las características principales que se siguen modificando cada pocos años. Sin embargo, una comprensión mínima de estas características es importante para entender adecuadamente las técnicas que se utilizan a un nivel superior, en el software, más aún cuando muchas de estas técnicas se están integrando, progresivamente, en el nivel inferior, en el hardware.

En esta sección intentaré abordar esta difícil tarea a partir de unos principios relativamente simples. En primer lugar, no hay que perder de vista que, con toda su complejidad, las tarjetas gráficas cumplen una misión muy sencilla: indicar a los dispositivos de salida de qué color deben pintar cada uno de los puntos de los marcos (*frames*) de una animación o de una sesión interactiva o de una imagen fija. Y los colores de estos puntos de salida se corresponden, en general, con los colores asignados por el usuario a los polígonos de los objetos que forman la escena, modificados por los cálculos de iluminación que se hayan activado. En segundo lugar, cumplen otra misión que también puede enunciarse de un modo simple: pasar por todos los estadios de un proceso complejo a la mayor velocidad posible. De ahí el nombre de "aceleradoras gráficas" con que también se las denominaba.

Si se tienen presentes estos principios básicos, resultará más fácil aceptar un resumen algo apretado y que comenzará por describir la evolución histórica de estas herramientas, lo que también ayudará a entender cuáles son las características principales que se han mantenido, o se han modificado o han ido apareciendo a lo largo de esta evolución.

Utilizaré la denominación *tarjeta gráfica* (*graphic card*) para referirme al conjunto de dispositivos y que es prácticamente sinónima de otras denominaciones igualmente corrientes como "placa gráfica" (*graphic board*) o "tarjeta de vídeo" (*video card*). También se utiliza muy a menudo como equivalente la denominación GPU (*graphic processing unit*), si bien este término debería reservarse en sentido estricto para el procesador principal instalado en una tarjeta gráfica. Dado que las descripciones de esta sección se basarán sobre todo en estas características, las del procesador principal, este será el término que utilizaré con más frecuencia.

Evolución

Los inicios de esta historia parten de un difícil equilibrio entre aplicaciones especializadas y aplicaciones generales, que se mantendrá a lo largo de todos estos años y que se sigue renovando en la actualidad. Es un equilibrio difícil pues si se deja a la CPU (*control processing unit*), la unidad principal de la placa base de un ordenador, todo el procesamiento gráfico, se gana en flexibilidad debido a que la manera de gestionar los procesos básicos sirve para cualquier tipo de programa y para cualquier tipo de aplicación. Y si muchos de estos procesos se incorporan a una tarjeta gráfica, a la GPU (*graphic processing unit*) se gana en eficacia y en velocidad, pero a costa de ligarlos a un tipo especial de aplicación, concretamente a los programas avanzados de simulación visual, a los vídeojuegos, a los

escenarios interactivos, y las aplicaciones de todo tipo relacionadas con estos programas.

Los antecesores de las primeras tarjetas gráficas eran dispositivos elementales cuya misión era generar imágenes a partir de datos muy generales y mostrarlos en un dispositivo de salida adecuado, tal como un monitor. El dispositivo que apareció con el primer PC, producido por IBM en 1981, no representaba imágenes, sino tan solo texto en color verde brillante sobre un fondo negro. Este "adaptador" (denominado MDA, *monochrome display adapter*, el término *tarjeta gráfica* aún no había aparecido), traducía la información generada por el ordenador a bloques de letras que se disponían en una retícula de 80 x 25 líneas. A este le siguieron otros con capacidad para representar elementos gráficos simples y que contaban con una gama limitada de colores: el CGA (*color graphics adapter*), en 1981, el EGA (*enhanced graphics adapter*) en 1984 o el adaptador 8514 de IBM en 1987.

En 1987, IBM introdujo lo que todavía sigue existiendo como adaptador o tarjeta gráfica básica, la VGA (*video graphics array*). En su primera versión, la VGA tenia una resolución de 80 x 25 celdas para texto y de 640 x 480 con 16 colores para imágenes, y una memoria de 256 Kb. La VGA sigue formando parte de algunos ordenadores, como un recurso básico instalado en la placa base y que permite al usuario gestionar gráficamente por medios simples la instalación de otros recursos más avanzados.

A mediados de la década de 1990, los ordenadores con mayores capacidades gráficas tenían varios chips especializados, más o menos desperdigados sobre la placa base, y que colaboraban para generar imágenes a la mayor velocidad posible. Por esas fechas se comenzó a agrupar estos chips en una única placa o "tarjeta" especializada. De este modo, además de optimizar el resultado de una colaboración más o menos dispersa, esto permitió que el desarrollo de las tarjetas se fuera especializando progresivamente.

El término GPU se introdujo a mediados de la década de 1990 (por NVIDIA) para describir a estas nuevas unidades independientes, con capacidades muy superiores a las de la VGA. A partir de estas fechas comenzó una evolución vertiginosa que, entre otras cosas, dejaría a tan solo dos compañías, NVIDIA y ATI (actualmente AMD), con el control de la práctica totalidad del mercado profesional. Pero esta evolución también es deudora, sobre todo por lo que hace a muchos conceptos y desarrollos técnicos cruciales, como la integración en hardware de transformaciones de vértices o proyección de texturas, del trabajo de grandes compañías especializadas, como Evans & Sutherland, que producían las mejores (y más caras) tarjetas gráficas de la época o de Silicon Graphics, que también se dirigía a un mercado muy especializado.

La primera generación de estas nuevas tarjetas se puede ubicar temporalmente entre 1990 y 1998. Sus características principales fueron la capacidad de integrar la rasterización de triángulos (previamente transformados por la CPU) y de aplicar algunos tipos simples de texturas. El resto del trabajo de computación necesario para generar imágenes como resultado de un cálculo de simulación de una escena 3D se seguía dejando en manos de la CPU. Parte de esas cosas que se dejaban para la CPU eran importantes por su carga computacional, como todo lo relativo a la transformación de vértices 3D en triángulos proyectados sobre el plano de salida. Tampoco incorporaban funciones matemáticas adecuadas para llevar a cabo el tipo de operaciones que se requieren para el procesamiento gráfico. Pero con todas estas limitaciones implicaban un trabajo considerable pues debían adaptar sus funciones elementales a cada tipo de ordenador. El coste de esta independencia era caro.

Las complicaciones que implicaba programar estas nuevas tarjetas independientes, para que se adaptaran a todo tipo de ordenadores, llevó a la aparición de aplicaciones genéricas que se situaban a un nivel intermedio entre los dos dispositivos, el ordenador y la tarjeta. De este modo, un núcleo importante de funciones se podía estandarizar, liberando al programador de tener que repetir con

pequeñas variantes largos bloques de código. Así surgió OpenGL en 1992 a la que le seguiría DirectX en 1995, de los que se tratará más extensamente en las siguientes secciones. Los fabricantes y modelos principales de esta época eran NVIDIA (TNT), ATI (Rage) y 3dfx (Vodoo).

Las nuevas tarjetas que aparecieron durante lo que a veces se denomina segunda generación, entre 1999 y 2000, se caracterizarían por la incorporación de nuevas funciones, principalmente para descargar a la CPU del procesamiento de vértices 3D y de cálculos básicos de iluminación. Tanto OpenGL como DirectX proporcionaban recursos para implementar este tipo de cálculos. También se incluyó un conjunto limitado de funciones matemáticas. La adición de funciones de transformación e iluminación fue el primer paso hacia una integración más compleja que llevaría posteriormente a una mayor flexibilidad, con la separación de unidades de programación de vectores (*vertex shaders*) y píxeles (*pixel shaders*), que también se describirán más adelante. Todo esto supuso un salto considerable en la velocidad de procesamiento de las nuevas tarjetas. La NVIDIA GeForce 256 (o NV10), lanzada en 1999, fue la primera tarjeta gráfica que introdujo este tipo de funciones.

La siguiente generación, a partir del año 2001, permitió por primera vez que las transformaciones de vértices fueran programables con lo que se abría el camino a las tarjetas actuales cuyo principal rasgo distintivo es la capacidad de programación independiente de diversas funciones de hardware.

A partir de 2002, esta tendencia hacia la especialización interna se fue incrementando con tarjetas que permitían no solo la programación independiente de procesos geométricos, procesando vértices, sino la programación de otros procesos de visualización más complejos, procesando píxeles. Esto abrió nuevas posibilidades, incorporadas a DirectX 9 y a las extensiones proporcionadas por OpenGL. Los fabricantes y modelos principales por esas fechas eran NVIDIA (serie GeForce FX) y ATI (serie Radeon 9700), que durante este período se consolidaron como los dos grandes fabricantes que acabaron por acaparar la práctica totalidad del mercado. La ATI Radeon 9700 (o ATI R300) fue la primera tarjeta gráfica que integró funciones dadas por Direct3D 9.0, entre ellas *pixel shaders* y *vertex shaders*, que podían incorporar bucles condicionales y cálculos complejos en coma flotante. A partir de estas fechas, las GPU se convirtieron en herramientas cada vez más parecidas a las CPU y que proporcionaban velocidades muy superiores para operaciones sobre imágenes.

Dado que desde mediados de la década de 2000 el mercado de las tarjetas profesionales quedó prácticamente dominado por NVIDIA y ATI (actualmente AMD), no estará de más añadir algo sobre estas dos empresas.

ATI (*array technologies incorporated*) se fundó en 1991 a partir de otra compañía fundada en 1985 (por Lee Ka Lau, Benny Lau y Kwok Yuen Ho) en Canadá. Producía tarjetas gráficas integradas para PC (IBM y Commodore principalmente). En 1987 se independizó de clientes concretos a los que estaba ligada y amplió su mercado. En 1991 sacó una tarjeta (Mach8) que podía procesar gráficos con independencia de la CPU y ofrecía más memoria y aceleración. A mediados de la década de 1990 sacó al mercado la primera tarjeta (3D Rage), que combinaba aceleración 2D y 3D. En 1999, la Rage 128 introdujo, además de mejoras en la aceleración, comandos que permitían sacar pleno partido del DirectX que ya estaba en su versión 6.0. En 2000 apareció la serie Radeon, una GPU que incorporaba aceleración 3D con DirectX 7.0, además de aceleración para 2D y vídeo. En paralelo amplió su mercado a la sincronización con TV (desde 1996) y consolas para Nintendo y otros juegos. En 2006, ATI se fusionó con AMD (Advanced Micro Devices) y se renombró como AMD Graphics Product Group (conocida también con el nombre ATI Technologies ULC) aunque el nombre ATI se mantuvo para las tarjetas gráficas. En 2015 sigue siendo el principal fabricante

de tarjetas gráficas profesionales, junto con NVIDIA.

NVIDIA (el nombre deriva por lo que parece de un juego de palabras entre "n", inicial de número, "vidia", relacionado con "vídeo" y "envidia", alusión a sus competidores), se fundó en 1993 (por Jen-Hsun Huang, Chris Malachowsky y Curtis Premium, los dos últimos provenientes de Sun Microsystems). Actualmente es una compañía multinacional con base en Santa Clara, California. Su primera tarjeta gráfica, la NV1, salió al mercado en 1995 e incorporaba el procesamiento de superficies cuadráticas. Esta novedad dejó de serlo cuando aparecieron las especificaciones de DirectX basadas en polígonos. Pero la compañía se adaptó a la irrupción de DirectX y OpenGL con una serie de nuevos productos, principalmente con una nueva tarjeta, la RIVA 128, que apareció en 1997 y que incorporaba especificaciones superiores a las de cualquier otro producto similar con un coste relativamente bajo, lo que la convirtió con rapidez en un producto popular. La GeForce 256 (NV10), lanzada en 1999, que introducía transformaciones e iluminación integradas en hardware, con 4 *pipelines* para píxeles, diversas mejoras en otros aspectos y una velocidad de reloj de 120 MHz, se convirtió en poco tiempo en la mejor tarjeta de sus características, superando a las de sus competidoras, incluida ATI. Los siguientes modelos de GeForce mantuvieron esta supremacía. Y estos modelos posteriores fueron incorporando nuevas *pipelines* para píxeles, texturas múltiples, *Z-buffers* y *Stencil-buffers* mejorados, filtrado anisotrópico, etc. Toda una serie de mejoras técnicas que también se describirán más adelante. La potencia de sus procesadores siguió aumentando hasta sobrepasar la de la CPU de ordenadores corrientes. Y en los siguientes años, una serie de adquisiciones (3dfx en 2000, ULI electronics en 2005, Hybrid Graphics en 2006, Ageia en 2008), reforzaron aun más su posición en el mercado. Tan solo las ATI y su serie Radeon aguantaron, entre otras cosas debido a sus mejores relaciones con Microsoft durante el desarrollo de DirectX9. Su línea de productos más conocida es la de las series de tarjetas gráficas GeForce, que a partir de la serie 100 ha cambiado su denominación por GT, GTS o GTX. Estas series incluyen, por un lado, una serie de chips especializados en gráficos y, por otro, varias tecnologías especializadas que también comentaré brevemente en los apartados siguientes.

Según algunos estudios de analistas de mercado estadounidense, que pueden encontrarse por internet, la cuota de mercado del sector de las tarjetas gráficas dirigidas a videojuegos correspondiente a NVIDIA era, en junio de 2010, de alrededor de un 60 % frente a alrededor de un 33 % para ATI. Si se considera los ordenadores corrientes, que utilizan tarjetas gráficas "no especializadas", las ventas se repartirían (datos de 2008) entre Intel (49 %), NVIDIA (28 %) y ATI (21 %). Todas estas cifras difieren relativamente según las diferentes fuentes. Pero, en cualquier caso, resulta claro que la venta de tarjetas gráficas de gama alta está dominada por estas dos compañías. Y, de hecho, en 2006 ambas recibieron advertencias del Departamento de Justicia de Estados Unidos por posible violación de la leyes Antitrust en el mercado de las tarjetas gráficas, lo que es un indicador bastante evidente de la posición dominante que ambas habían alcanzado.

Hay que aclarar que aunque ATI/AMD y NVIDIA son los principales fabricantes, hay muchas marcas que ofrecen tarjetas gráficas, lo que puede resultar confuso. Pero estas marcas se dedican realmente a ensamblar las tarjetas, utilizando los chips y la tecnología suministrada por los dos grandes fabricantes. Algunas marcas conocidas en la primera y segunda década de este siglo son ASUS (www.asus.com), Club3D (www.club3d.nl), Gainward (www.gainward.com), Gigabyte (www.giga-byte.es), HIS (www.hisdigital.com), Leadtek (www.leadtek.com), MSI (www.msi.com.es), PNY (www.pny.com), Zotac (www.zotac.com).

Componentes principales y especificaciones técnicas

Si se buscan las características de una determinada tarjeta gráfica nos encontraremos con una serie de términos y datos aproximadamente semejantes a los que se resumen en la tabla de la figura 3.1. Es decir, datos relativos a: *a*) el procesador principal, *b*) la memoria y la tasa de transferencia, *c*) la tecnología soportada, *d*) la potencia de consumo y, e) las conexiones externas. Algunos de estos datos son relativamente estables mientras que otros se renuevan cada cierto tiempo. La intención de este apartado es describir mínimamente el sentido de estas especificaciones técnicas.

El componente principal de una tarjeta gráfica es la unidad de procesamiento gráfico, la GPU, el *motor principal*, un procesador especializado que ha ido incorporando nuevas funciones en los últimos años. El primer dato que conviene anotar es la velocidad del procesador básico. Pero este dato puede no ser demasiado significativo por sí mismo, pues la frecuencia de reloj del procesador de la CPU suele seguir siendo muy superior, en general, al de la GPU. La potencia de la GPU deriva de otros factores como, por ejemplo, de la diversificación de sus unidades fundamentales de cálculo, principalmente unidades que procesan vértices y unidades que procesan píxeles. Por esta razón, frecuencias relativamente bajas, del orden de 800 MHz en las tarjetas más recientes (2014), bastante inferiores a los 3.000 o 4.000 MHz de la CPU por estas mismas fechas, van ligadas a rendimientos mucho más altos que sacan partido de una arquitectura muy distinta que favorece el *procesamiento en paralelo* de un número creciente de unidades funcionales. Con todo, es uno de los primeros datos que hay que anotar.

La velocidad y potencia de procesamiento de la GPU se mide de varios modos: *a*) por el reloj de gráficos (*graphics clock*); *b*) por el reloj del procesador (*processor clock*), la velocidad de reloj (*clock speed*), el reloj del motor (*engine clock*) o el reloj central (*core clock*), pues se utilizan todos estos términos; *c*) por valores de rendimiento tales como la tasa de relleno de texturas en miles de millones por segundo (*texel fill rate*, en gigatexels por segundo), el número de píxeles que la tarjeta es capaz de dibujar por segundo o bien la tasa de dibujo de polígonos también medida en miles de millones por segundo.

Por ejemplo, la NVIDIA Geforce GTX 480, un modelo bastante sencillo, tiene las características siguientes: 700 MHz de velocidad de gráficos, 1.401 MHz de velocidad del procesador, 42 GT/s de *texel fill rate* . El "reloj" es un modo sintético de referirse a la frecuencia o la velocidad de proceso. Se mide en herzios (ciclos por segundo). Una velocidad de 800 MHz quiere decir que se pueden hacer 800 millones de "cosas" por segundo, por lo general, operaciones matemáticas. Esto no es del todo exacto pues algunas operaciones requieren 1 ciclo mientras que otras pueden requerir 8 o incluso más. Pero es una referencia convenida que da una idea de la capacidad de procesamiento. En algunos casos se dan referencias más precisas, como el número de operaciones en coma flotante (*FLOPS*) o, como decía antes, el número de polígonos que pueden ser procesados por segundo.

El número de transistores es otra medida clásica de la potencia de procesamiento. Pero en las tarjetas más recientes también hay que tomar esta cifra con reservas. Su crecimiento ha sido espectacular en los últimos años y ha estado ligado a otro factor importante: la tecnología de fabricación, que actualmente (2011) permite trabajar a escalas de 40 nanómetros (10^{-9} metros). Las cifras siguientes dan una idea de esta progresión. En 1998, las mejores tarjetas contaban con alrededor de 7 millones de transistores. En 1999 ya se había saltado a 23 millones (la GeForce 256), lo que estaba ligado a una capacidad de procesamiento de 15 millones de triángulos por segundo. En 2003, la Geoforce FX contaba con 125 millones de transistores y una capacidad de procesamiento de 200 millones de triángulos por segundo. Hacia 2010, las cifras corrientes eran las que siguen. La Radeon

HD 5770 tenía 627 millones de transistores. La Radeon HD 5870 tenía 2.150 millones de transistores con tecnología de proceso de 40 nm. La Geforce GTX 480 y la GTX-570 tenían 3.000 millones de transistores con tecnología de proceso de 40 nm.

Sin embargo, los modelos posteriores han reducido estas cifras pero aumentando otras prestaciones, principalmente la velocidad de proceso, lo que ha resultado en tasas mayores de procesamiento pero con incrementos reducidos en el número de transistores.

Estos cambios están ligados a otros de más calado que se basan en una redefinición de la microarquitectura de estas tarjetas, un proceso complejo en el que participan otras compañías. Tanto NVIDIA como AMD tienen como proveedor a TSMC (Taiwan Semiconductor Manufacturing Company), la mayor compañía independiente del mundo que fabrica chips para AMD y NVIDIA, entre otros fabricantes. Esta redefinición está ligada a la nueva arquitectura Fermi o GF100, en el caso de NVIDIA o a la nueva arquitectura basada en los chips Barts o Cayman en lugar de los Cypress, en el caso de AMD. Pero este es un tema excesivamente complejo del que solo se puede dar esta breve indicación para hacer ver que por debajo de las descripciones sintéticas que aquí se dan hay niveles inferiores de gran complejidad.

Por todas estas razones, puede ser más significativo comprobar datos de rendimiento que pueden medirse por medio de otras referencias características como las correspondientes GT/s o GFLOPS.

El primer valor citado, GT/s (giga transfers por segundo o 10^9 operaciones de transferencias por segundo), se refiere al número de datos transferidos, agrupando tanto la velocidad como la cantidad transferida, pues la velocidad por sí misma tiene poco significado dado que importa más la cantidad de datos que se mueve a una velocidad determinada. Es evidente que una tarjeta gráfica con 1 Gb de memoria será mucho más rápida transfiriendo datos si cuenta con un bus de 256 bits de anchura, aunque su frecuencia sea más lenta que si contara con una frecuencia más rápida pero un bus de 64 bits. La frecuencia se mide en MHz (mega ciclos por segundo) mientras que GT/s es la cantidad de datos transferidos.

Un valor significativo es el número de unidades de textura que se proyectan sobre la pantalla por segundo. Esto se mide por el *Texell Fill Rate* que se calcula multiplicando el número de *texture mapping units* por la velocidad de reloj del procesador central (*core clock*). Y el valor resultante se mide igualmente en GT/s.

El segundo valor citado, GFLOPS, es otra medida corriente, más general, dada por los millones de operaciones por segundo que lleva a cabo una unidad de procesamiento. Esto se mide en FLOPS (*Floating Point Operations per Second*). Como los valores actuales son muy grandes se utilizan las unidades derivadas MFLOPS (*megaflops* o 10^6 flops), GFLOPS (*gigaflops* o 10^9 Flops) o, en grandes computadores, TFLOPS (*teraflops* o 10^{12} Flops) o PFLOPS (*petaflops* o 10^{15} Flops). En las especificaciones actuales se puede encontrar también la indicación adicional GFLOPS (FMA). Las siglas FMA (*Fused Multiply Add*) indican que las operaciones de multiplicación y suma están integradas en un solo ciclo si el dispositivo lo permite (por ejemplo d = a + bc) con lo que el rendimiento es mayor que si no lo permite (dos ciclos en lugar de uno).

Otro componente principal es la *memoria* que, en las modernas GPU, se dedica principalmente a almacenar resultados intermedios de las operaciones. En las tarjetas antiguas la memoria iba ligada muy directamente a la resolución. Una resolución como la de las tarjetas actuales, que suele alcanzar un máximo del orden de 2.560 x 1.600, requiere procesar algo más de 4 millones de píxeles (4.096.000). Para 32 bits por píxel esto supone 15.6 Mb. Esto es lo que requiere el *frame buffer*, la memoria intermedia en donde se almacenan las imágenes antes de ser enviadas al monitor. Pero todas las tarjetas actuales incorporan al menos un *back buffer* que mantiene la siguiente imagen preparada para

evitar parpadeos, lo que supone almacenar la misma cantidad. Y todas incoporan al menos un *Z-buffer*, un *Stencil-buffer* que guardan la misma información para calcular la profundidad de los vértices correspondientes a cada píxel o las máscaras que ocultan parte de estos píxeles, si bien esta información puede comprimirse con lo que resulta ser algo menor. Y todas incorporan unos cuantos *buffers* más tan solo para el estadio previo a la salida.

Y esto no es sino una parte de la memoria de las tarjetas actuales, pues cada *vertex shader* y *pixel shader* guarda su propia información sobre la escena, a lo que se suma la correspondiente a los más recientes *shaders* que veremos más adelante. Los mapas de textura se almacenan también en memoria de un modo variable, que dependerá de su tamaño y profundidad de color y otras características. Así, resulta que los requisitos de memoria acaban por ser del orden de 100 veces más que los requeridos por la resolución de salida, si bien todos ellos son múltiplos de esta resolución inicial, otro factor que no debe perderse de vista. Por estas razones, la memoria de las tarjetas gráficas de gama baja/media es del orden de los 1.500 Mb por lo menos.

Pero un factor más importante es la tasa de transferencia, es decir, la velocidad con que se transfiere esta información. Y esto implica varios parámetros importantes: *a*) la capacidad y el tipo, *b*) el reloj de memoria en MHz, *c*) el interfaz de memoria en bits, *d*) el ancho de banda de memoria en GB/s.

Hasta los primeros años del siglo XXI la memoria de las tarjetas gráficas era un tipo especial de memoria RAM (*Random Access Memory*), la "memoria dinámica con interfaz síncrona", SDRAM (*Synchronous Dynamic Random Access Memory*), que se diferencia de la memoria DRAM (*Dynamic Random Access Memory*) en que esta tarda un cierto tiempo en responder a los cambios de entrada, tras un número determinado de pulsos, un retraso que se denomina técnicamente "latencia". La memoria SDRAM responde en el momento señalado por la señal del reloj y está, por tanto, sincronizada con el bus de sistema del ordenador.

Hacia 2003 (se había ido desarrollando entre 1996 y 2000 por la compañía JEDEC) apareció un nuevo tipo de memoria denominado DDR SDRAM (*Double Data Rate Synchronous Dynamic Random Access Memory*) o *DDR* sin más (*Double Data Rate*), que permitía aumentar los tiempos de transferencia hasta duplicar aproximadamente la anchura de banda de la anterior. Con una transferencia de 64 bits por pulso, se conseguía así una anchura de 128 (64 x 2) bits que, a una frecuencia de bus de 100 MHz, daba valores máximos de transferencia de 1.600 MB/s. Poco después apareció la versión DDR2, que permitía mayores velocida-

Modelo	*Motor principal (1)*				*Memoria (2)*				*Varios (3)*			
	Core (MHz)	*Trns (m)*	*Txt (GT/s)*	*GFLOPS*	*Tipo*	*Bus datos (bits)*	*Band-width (MB/s)*	*Clock rate (MHz)*	*Conn.*	*Tecn.*	*TDP (w)*	*Coste (€)*
Gama baja	800	600	10	300	GDDR3	128	28,8	1.800	PCIe 2.0	750	65	100
Gama media	900	1.000	100	2.000	GDDR5	256	134,0	1.500 (6.000)	PCIe 3.0	1.500	150	300
Gama alta	900	7.000	200	3.000	GDDR5	384	288,0	1.500 (6.000)	PCIe 3.0	2.500	250	600
(1)	*Core* indica la frecuencia del procesador principal. Trns (m) los millones de transistores. Para Txt (GT/s) y GFLOPS véase el texto.											
(2)	No se ha incluido la resolución máxima que sería de 2.560 x 1.600 px en todos los casos											
(3)	En *Conn*. (conexiones) no se ha incluido los conectores externos que son al menos VGA, DVI y HDMI. En *Tecn*. (tecnología) las cifras se refieren a (muy aproximadamente) núcleos CUDA. No se mencionan las versiones de DirectX y Open GL, que serían generalmente 11 y 4.4, ni OpenCL (1.1) En *TDP* (*Thermal Design Power*) se indica la potencia térmica equivalente. Véase el texto. El coste en euros es muy aproximado y no incluye los modelos de gama muy baja ni muy alta. Los tres modelos indicados pueden corresponder, en 2014 a modelos intermedios de una Geforce GTX de la serie 400, una Geforce GTX de la serie 600 y una Geforce de la serie 700.											

Figura 3.1 Componentes y especificaciones características de tarjetas gráficas.

des y alcanzaba una tasa de transferencia de 3.200 MB/s. Y poco después la versión DDR3, que permitía trabajar con menores voltajes, lo que reducía el consumo, y que introducía un nuevo recurso para aumentar las tasas de transferencia, los *prefetch buffers*, que gestionaban de un modo más eficaz las matrices internas de datos. Así se alcanzaron tasas de transferencia de 6.400 MB/s (con 100 MHz de *memory clock*) que en poco tiempo aumentaron hasta 17.066 MB/s, con mayores frecuencias de reloj de memoria (266 MHz).

Un poco antes, en 2002, se publicaron las especificaciones de un nuevo estándar, el *GDDR3 (Graphics Double Data Rate*), desarrollado por ATI en colaboración con JEDEC (aunque la primera tarjeta que utilizó este tipo de memoria fue la GeForce FX 5700 Ultra de NVIDIA). La tecnología era similar a DDR2 pero el consumo se reducía, permitiendo un mejor rendimiento de los módulos de memoria, y utilizaba 4 bits de datos por pin en ciclos de reloj lo que, con una anchura de 32 bits, le permitía alcanzar más de 5.6 GB/s.

En 2005 se desarrollaron, por Samsung, los primeros estándares de la memoria de *GDDR4*, basada en DDR3, con una anchura de banda de 2,5 GB/s, que fue aumentando en las siguientes versiones a 3,2 por pin o 12,8 GB/s por módulo (2006), hasta 16 GB/s por módulo (2007).

En 2008 se comenzó a utilizar la memoria *GDDR5* (con la tarjeta Radeon HD 4870 de ATI). Este nuevo tipo, con una mayor anchura de banda, se basaba en la memoria DDR3 que tiene el doble de líneas de datos (líneas DQ) que la DDR2 pero le añadía un *prefetch buffer* de 8 bits, como la GDDR4. La GDDR5 opera con dos tipos de reloj, uno que gestiona los comandos de entrada (CK, *command clock*) y otro que gestiona las operaciones de lectura y escritura de datos (WCK, *write clock*). Todo esto hace que la velocidad de transferencia de datos sea notoriamente superior a los anteriores.

Un tercer factor importante es la *tecnología soportada* que aparece en las especificaciones de las tarjetas gráficas modernas, y se refiere a los programas o API (*application programming interface*) que incorpora una tarje-

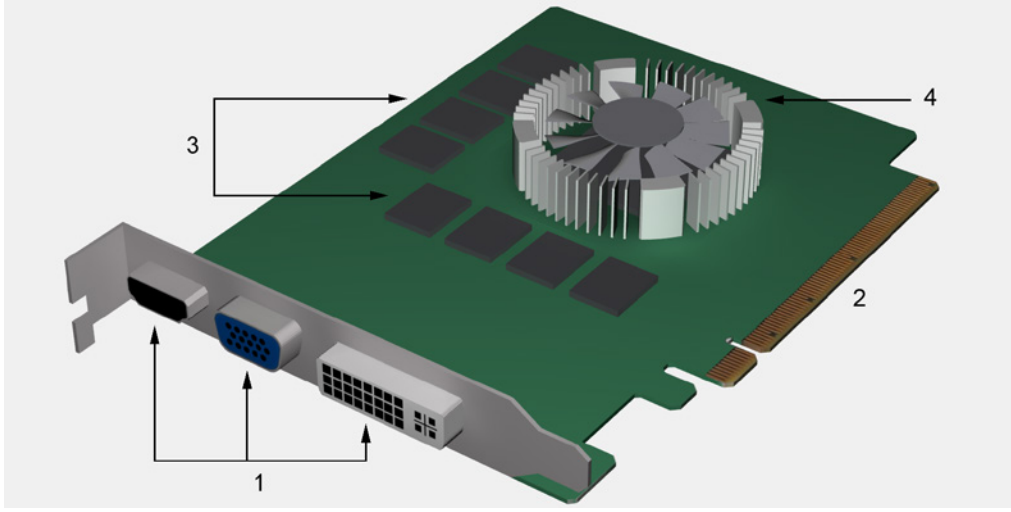


Figura 3.2 Vista simplificada de una tarjeta gráfica de gama baja/media (dimensiones aproximadas: 11,5 x 15 cm): 1) conectores externos (HDMI, VGA, DVI), 2) conexión PCIe con la placa base (CPU), 3), chips de memoria, 4) ventilador sobre el chip principal. No se muestran condensadores, resistencias, transistores ni los buses de conexión que van a parar al PCIe.

ta. Es habitual que se incorporen versiones concretas de *DirectX* y *OpenGL*. Pero puede incorporar otras que han ido surgiendo en los últimos años, a partir de 2006 principalmente. En las tarjetas actuales es corriente encontrar al menos *OpenCL* y, en las tarjetas de AMD la tecnología *AMD Stream Processor*. Y en las de NVIDIA, la tecnología *CUDA* que, al igual que las anteriores, permite sacar el máximo rendimiento del procesamiento en paralelo. Estas nuevas aplicaciones han surgido en buena parte a raíz de la extensión de las aplicaciones gráficas a otras aplicaciones (financieras, médicas, etc.), para sacar partido de la enorme potencia de computación en paralelo de las modernas tarjetas, una extensión que se conoce genéricamente con la denominación *computación de propósito general sobre GPU*, *GPGPU* por sus siglas en inglés (*general purpose computing on graphics processing units*). Sobre este tema volveremos más adelante.

Otra referencia que puede aparecer es la versión del *shader* utilizado, por ejemplo *Shader Model 4.0*. Pero esto es prácticamente equivalente a indicar la versión de DirectX u OpenGL utilizada, como también veremos más adelante. Es una especificación propia del lenguaje HLSL (que también se comenta en la sección siguiente), ligado a DirectX y está relacionada directamente con la versión correspondiente. Los modelos de *vertex shaders* y *pixels shaders* son los siguientes: VS 2.0 y PS 2.0 (DirectX 9.0, que incluía la primera especificación del *Shader Model*), VS 3.0 y PS 3.0 (*Shader Model* 3), VS 4.0 y PS 4.0 (*Shader Model* 4). Esta denominación proviene de que a lo largo de las diferentes versiones se ha buscando unificar el lenguaje de *shaders* para simplificar la tarea de los programadores. A partir de la versión DirectX 10, el *Shader Model 4.0* se considera equivalente al *Unified Shader Model*, una denominación utilizada por OpenGL pero también por muchos fabricantes o montadores de tarjetas y que se refiere a un conjunto de instrucciones consistente, más o menos común para todos los tipos de *shaders* (aunque sigue, como es lógico, instrucciones diferentes pues, por ejemplo, solo los *geometry shaders* pueden crear primitivas geométricas y solo los *pixel shaders* pueden leer texturas de determinadas características), y que también abren el camino a utilizar *shaders* más versátiles, no ligados estrictamente a una fase del procesamiento.

Así, en las especificaciones de tecnología soportada, se pueden encontrar referencias a unidades especializadas tales como *shaders unificados* (US, *unified shaders*, *shaders* que sirven tanto para procesar vértices como píxeles con lo que aumenta la flexibilidad de programación), *tuberías de proyección de texturas* (TMP, *texture mapping pipelines*, que se encargan de los procesos de proyección y filtrado de texturas), "tuberías de salida de render" (ROP, *render out pipelines*, que se encargan del proceso de salida a pantalla y de realizar operaciones finales como la combinación de valores de píxeles provenientes de varias vías), denominaciones que se refieren en todos los casos a unidades especializadas que descargan a la GPU de realizar determinadas tareas, con lo que también aumenta la velocidad, y que están ligados a las API mencionadas. De este modo, por ejemplo, los *shaders* pueden dedicarse a hacer tareas principales y, una vez finalizadas, pasan los resultados a las otras unidades para que las completen, con lo que pueden dedicarse a otro grupo de tareas.

Las tarjetas recientes también incluyen otras unidades especializadas en el procesamiento de funciones más especializadas, como *PhysX* (simulación de efectos físicos) o *Eyefinity*, que se utiliza para manejar múltiples pantallas, entre otras.

Por último, la *potencia* en vatios de la tarjeta, que va unida a la *temperatura* que puede alcanzar la GPU es otro factor que no se debe pasar por alto. Las tarjetas que se detallan al final, y que cubren la gama media-alta de ordenadores de sobremesa, pueden requerir potencias que están entre los 120/220 y 350/700 (el primer valor corresponde a actividad mínima y el segundo, a actividad máxi-

ma). Además de que esto implica una alimentación secundaria solo para la tarjeta, algunas pueden requerir nominalmente hasta 500 o 600 vatios pero al alcanzar picos de trabajo pueden necesitar algo más y, si no cuentan con esta potencia, el ordenador se cuelga, por lo que es recomendable suministrarle una potencia algo superior a la nominal. Y todo esto está ligado al consumo, a la temperatura y al ruido.

La mayor o menor temperatura de la tarjeta se debe a que el movimiento de la corriente eléctrica genera calor, tanto mayor cuanto mayor sea la velocidad. Hay una relación básicamente lineal entre las frecuencias de reloj de un procesador y lo que se denomina potencia térmica de diseño, o *TDP* (*thermal design power*). Y hay un límite para las frecuencias posibles que solo puede superarse aumentando la potencia de voltaje suministrada. Pero la relación entre el voltaje de los chips y la TMD es exponencial, debido a que al aumentar el calor del chip aumenta la resistencia. Este aumento del calor requiere enfriar de algún modo el chip para evitar que resulte dañado. Y el aumento del calor puede hacer que la velocidad disminuya. De ahí que la potencia y la necesidad de refrigerar las tarjetas se hayan convertido en un problema importante a medida que ha ido aumentando su potencia en los últimos años. Hay muchos medios que se han probado: enfriar los ordenadores con agua, con nitrógeno líquido, con tubos de cobre. Y con el método más corriente, que es colocar un gran ventilador sobre los chips, lo que genera otro problema: el ruido. Cuando trabajan al máximo, estas tarjetas pueden producir entre 45 y 55 dB de ruido, lo que es una cifra digna de consideración.

Conexiones externas

Un componente muy importante en el rendimiento de los ordenadores personales es el *bus de conexión* entre las tarjetas gráficas y la CPU, lo que justifica que le dediquemos un apartado.

En general, un bus es una vía de transferencia de datos. Esto implica varios parámetros que no son demasiado diferentes a los que se utilizan para describir la circulación en las autopistas: la anchura (en bits) de la vía, la velocidad (de sincronismo) y la existencia de controladores que regulen el tráfico. Físicamente se relacionan con trazas impresas en la placa base, que pueden verse a simple vista y que han substituido al cableado de los primeros ordenadores. Pero no se deben identificar demasiado literalmente los buses con estas marcas físicas que no son sino el soporte para conexiones lógicas bastante más complejas y que dependen de la organización de los datos: aquí se acabaría la analogía con las autopistas (aunque podemos prolongarla si pensamos que los vehículos pueden transportar cosas muy diversas y que podríamos medir el tráfico en función, no de los vehículos sino en función de las personas y cosas que estos transportan).

Dado que las tarjetas gráficas evolucionaron hasta convertirse en un componente independiente que se debe conectar a las placas base, es evidente que la vía de conexión entre la tarjeta gráfica y el ordenador (la GPU y la CPU) tiene una gran importancia pues se pueden crear cuellos de botella que ralenticen todo el proceso.

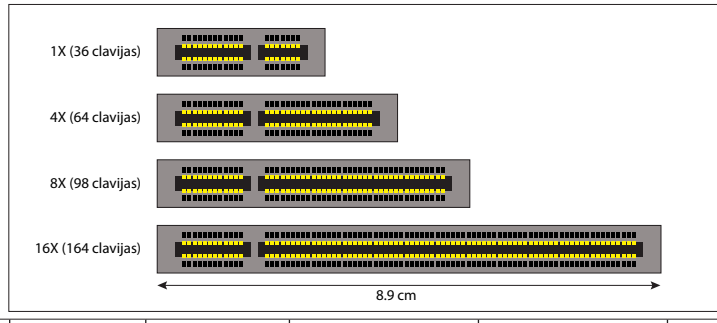
El primer bus de conexión directa entre la GPU y la CPU se produjo en 1974 (era un componente de un ordenador pionero en la época, el Altair 8800). Pero el mercado de los PC adoptó inicialmente el bus ISA (*Industry Standard Architecture*) de IBM, que apareció también en 1981 y tenía una anchura de 8/16 bits con una velocidad de 8 MHz. Esto supuso una innovación fundamental pues permitiría en breve plazo ampliar la capacidad de los ordenadores en función de las diferentes necesidades de los usuarios. Pronto siguieron otros tipos que introdujeron mejoras sucesivas: el NuBus de los Macintosh, el MCA (*Microchannel Architecture*) de IBM, o el EISA (*Extended Industry Standard Architecture*), lanzado en 1988 por una asociación de competidores de IBM.

A partir de 1996 aproximadamente, se impusieron tres tipos de bus para substituir a los citados: el primero fue VESA, que pronto cedió su posición a PCI y AGP. Se caracterizaban, en primer lugar, por una conexión dedicada de modo exclusivo al flujo de información entre la CPU y la GPU. Y, después, por un crecimiento sostenido de sus características principales: la memoria, la velocidad y el ancho de banda. Tanto VESA com PCI se denominaron *buses locales* debido a que ampliaban "localmente" la CPU por medio de ranuras de expansión. VESA (*Video Electronics Standards Association*), aparecido en 1992, era fundamentalmente una extensión de ISA, con un conector de 16 bits agregado para llegar a 32 bits, muy grande, una característica que llevó a su rápida substitución por otros modelos más pequeños y con prestaciones similares.

En 1993 se introdujo el PCI (*Peripheral Components Interconnect*). Su anchura era de 32 bits, el *clock rate* de 33 a 100 MHz y la anchura de banda de 132 a 800 MB/s. El PCI-X apareció en 1998 como una extensión del PCI, basada principalmente en la ampliación de la anchura del bus a 64 bits y la frecuencia del reloj a 133 MHz.

En 1997 apareció AGP (*Accelerated Graphics Port*) como solución a los cuellos de botella que se producían con los buses PCI. La denominación "puerto" indicaba que solo se podía conectar un dispositivo (en un bus se pueden conectar varios). Su anchura era también de 32 bits, el *clock rate* de 66 MHz y la tasa de transferencia de 266 MB/s. Las diferentes versiones que fueron apareciendo se denominaron 2X (pues la velocidad, 133, y la tasa de transferencia, 532 MB/s, eran dobles que en el anterior), 4X (266 MHz, 1 GB/s), 8X (533 MHz, 2 GB/s). La duplicación de las tasas de transferencia se conseguía sin modificar las características físicas principales, aumentando los ciclos del reloj. El último tipo, el AGP 8X, se lanzó hacia 2000.

Con todo, ya a principios de la década de 1990 comenzó a apreciarse cada vez más claramente que las necesidades siempre crecientes de aumento de la tasa de transferencia llegarían pronto a un límite si se ligaban exclusivamente al aumento de la anchura del bus paralelo. Este aumento repercutía en el coste pero también en ruido y en otros problemas técnicos. Así se planteó una nueva tecnología que utilizaba transferencia en serie pero ligándola a un aumento de la frecuencia y a un tipo nuevo de recurso compartido que dejaba de ser propiamente el bus para pasar a ser el *switch*: un punto de conexión entre varias líneas. Cada dispositivo del sistema tiene una conexión directa a este *switch*



Fecha	*Bus*	*Reloj (GHz)*	*Ancho de banda por vía (MB/s)*	*Ancho de banda total para x16*	*Tasa de transferencia*
2003	PCIe 1.0	2,5	~250 MB/s	~8 GB/s	2,5GT/s
2007	PCIe 2.0	5,0	~500 MB/s	~16 GB/s	5,0GT/s
2010	PCIe 3.0	8,0	~1 GB/s	~32 GB/s	8,0GT/s

Figura 3.3 Esquema (arriba) y características principales de las distintas versiones del PCIexpress.

que conecta múltiples entradas con múltiples salidas.

En 2003, apareció este nuevo modelo de bus, el PCIe o PCI Express (*Peripheral Component Interconnect Express*) con interfaz punto a punto, lo que quiere decir que, a diferencia de los anteriores, la transferencia era en serie. Pero la diferencia fundamental era que la frecuencia era mucho mayor, con lo que se alcanzaban anchuras de banda efectiva muy superiores a las anteriores. Por añadidura, los datos podían ser enviados por dos direcciones a la vez (en paralelo los datos van en una sola dirección) pues cada conexión incluía dos cables, uno en una dirección y otro en la opuesta. Otra ventaja adicional es que la vía de transmisión no se compartía como en los PCI, por lo que había menos congestión en el tránsito. Los PCIe que surgieron a partir de esta fecha se distinguen por el número de *lanes* (calles) que utilizan. Las conexiones son escalables y se denomina x1, x2, x4, x16 o x32: la cifra a continuación de la x indica el número de carriles de datos de conexión con la placa base. La anchura de banda dual está relacionada directamente con estos valores que para el PCIe 1.0 son: 5 Gbps/400 MB/s (x1), 20 Gbps/1.6 GBps (x4), 40 Gbps/3.2 GBps (x8), 80 Gbps/6.4 GBps (x16). En comparación con los anteriores, un carril simple es aproximadamente el doble de rápido que un PCI normal.

En 2007 apareció la versión 2.0, que duplicaba las tasas de transferencia del anterior, hasta unas 5 GT/s (gigatransfers/segundo), con un ancho de banda de 500 MB/s. En noviembre de 2010 se publicaron las especificaciones de la versión 3.0, que alcanzaría los 8 GT/s. Con una frecuencia de 8 GHz la anchura de banda alcanzaría cerca de 1 GB por segundo para una sola pista (*lane*) y en una dirección, lo que equivale a 32 GB/s de ancho de banda total para un x16.

La tabla de la figura 3.3 muestra los principales datos de los buses PCIexpress.

En las tarjetas actuales todos los buses de conexión entre la GPU y la CPU son PCIexpress. Y también, en prácticamente todos los casos, la versión es 2.0 y el tipo x16. Las especificaciones del PCIexpress 3.0, recientemente aparecido, son de 1 GB/s direccional y 2 GB/s bidireccional, por lo que para x16 supone un máximo teórico de 16 GB/s direccional y 32 GB/s bidireccional.

Además de esta conexión principal, la mayoría de las tarjetas gráficas tienen al menos dos *conectores*, un conector *DVI* (*Digital Visual Interface*) que soporta pantallas de tipo LCD, y un conector VGA, que soporta las clásicas pantallas CRT que ya están en vías de desaparición por lo que algunas tarjetas han optado por dos conectores DVI en lugar de uno de cada, lo que no implica necesariamente abandonar la opción clásica pues es posible conectar un CRT mediante un adaptador. Otro conector relativamente corriente desde 2003 es el *HDMI* (*High Definition Multimedia Interface*), que se utiliza para conectar consolas de juego y DVD a un monitor. Además de estos tres conectores habituales algunas tarjetas incluyen conectores para TV, para cámaras de vídeo o para cámaras digitales. Y, desde 2007, salidas *DP* (*Display Port*) que puede acabar desplazando a los anteriores. Por ejemplo, la ATI Fire Pro (2010) tiene hasta 4 salidas de tipo *Display Port* que, con 2 Gb GDDR5 de memoria, permite manejar hasta 4 pantallas de 30" con resoluciones de 2.560 x 1.600 cada una.

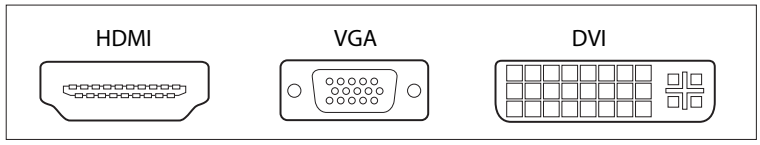


Figura 3.4 Conectores básicos de una tarjeta gráfica.

La base algorítmica. El concepto de *shader*

Los lenguajes de programación por hardware se denominan también *lenguajes de shaders* (*shading languages*) debido a que se basan en la programación de bloques de código que pueden compartirse entre aplicaciones y que se denominan *shaders*, un término que podría traducirse, con reservas, por "sombreadores".

El concepto de *shader* ha evolucionado tanto en los últimos años que su uso actual puede resultar equívoco debido a que se utiliza a dos niveles bastante distintos. Un *shader* puede estar embebido en hardware, formando parte de la GPU, o estar a disposición de un programador, como parte del software que se comunicará en su momento con la GPU de diferentes modos. Volveremos a esta distinción al final de esta sección, después de haber explicado sus fundamentos y su origen, y también en el capítulo siguiente.

Shader trees

El concepto de *shader* fue introducido por Robert Cook en un artículo publicado en 1984 (véase Cook, 1984). El trabajo de Cook en Lucasfilm, que posteriormente se convertiría en Pixar, llevó a la clasificación de los *shaders* en diferentes categorías: *shaders* de superficies, de luces, de efectos atmosféricos, etc. Sobre esta aportación original se desarrolló el primer lenguaje de *shaders*, RenderMan en cuyo desarrollo tuvo también un papel importante otro gran investigador que ha aportado múltiples técnicas a la simulación de materiales, Pat Hanrahan.

Otra aportación fundamental dentro de esta evolución fue la aparición, por estas mismas fechas, de los lenguajes procedurales en el que jugaron un papel principal investigadores como Ken Perlin, David Peachey o Steven Worley, entre otros, a quienes ya me he referido en el capítulo anterior.

Un *shader* es básicamente un procedimiento invocado por un programa más general para computar el valor o conjunto de valores requeridos durante el proceso de *rendering* (véase Upstill, 1990, p. 211). Dicho de otro modo: es un algoritmo que se utiliza para calcular todo tipo de efectos de representación. El concepto introducido por Cook ha pasado a ser una parte fundamental de los lenguajes de programación por hardware pues proporciona una abstracción extraordinariamente útil y flexible que puede ser utilizada a todos los niveles y para todo tipo de finalidades. El término alude a la función principal de un *shader* que es modificar de algún modo las características de una superficie ("sombrearla" o "modularla"), al igual que lo haría una luz que se moviese por encima de ella.

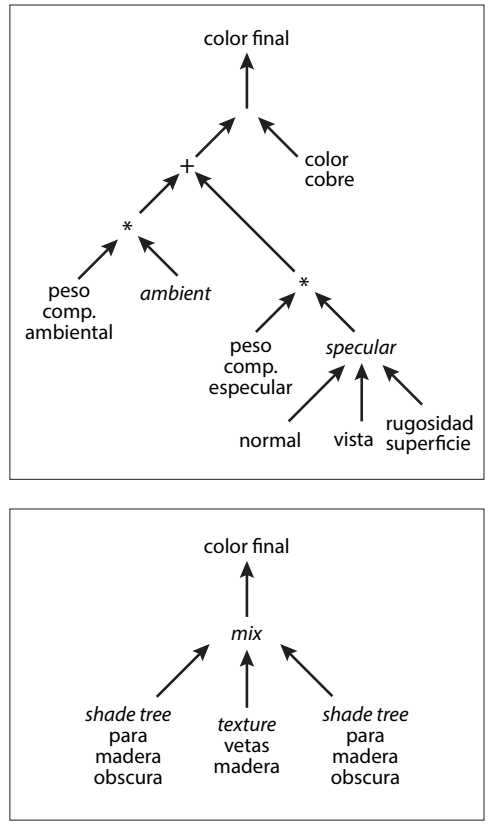


Figura 3.5 Ejemplos de shaders *para simular cobre y para simular madera. Adaptados del artículo de Cook de 1984.*

La figura 3.5, reelaborada a partir de un grupo de figuras que aparecen en el artículo original de Cook, muestra las ideas básicas: dierentes funciones se combinan en una estructura jerárquica que, en la cúspide, las integra en un resultado único.

Los *shaders* se utilizan para programar las GPU y han substituido a secciones completas de lo que se denomina, desde los inicios de la computación gráfica, la *rendering pipeline* que, hace muchos años, estaba formado por algoritmos fijos, embebidos en las placas de las tarjetas y que no podían modificarse con facilidad. Los *shaders* han permitido subsitutir estas secciones rígidas por secciones programables, más flexibles.

Hay dos tipos de *shaders* que están incorporados a las dos *libraries* gráficas que utilizan todas las tarjetas gráficas: *vertex shaders*, y *pixel* o *fragment shaders*. También puede encontrarse un tercer tipo: *geometry shaders*, que se ha introducido con la versión 10.0 de Direct3D.

Vertex shaders

A medida que se han ido desarrollando las dos principales API de programación por hardware, OpenGL y DirectX, se han ido desarrollando y estandarizando algunos estadios principales de la *rendering pipeline*. El primero de estos estadios es el correspondiente a un conjunto de operaciones corrientes que se llevan a cabo sobre los datos principales recibidos de la aplicación: los vértices de los objetos de que está compuesta la escena y que incorporan la información principal, geométrica y no geométrica: coordenadas de posición, vectores normales, colores y coordenadas de textura.

Para cada vértice se ejecutan una serie de operaciones de transformación geométrica: operaciones de transformación de coordenadas locales en globales, de transformación perspectiva del espacio del modelo en el espacio del visor y de normalización de este espacio. Los valores procesados se almacenan en una memoria temporal hasta que son enviados a la imagen de salida.

Las transformaciones principales de este estadio son las que se engloban bajo la denominación genérica de *vertex shader*. Concretamente, son operaciones de transformación geométrica de espacios de coordenadas, de iluminación directa que afecta a los atributos de los vértices y, con algunas variantes según las API, de eliminación de vértices cuyas normales apuntan en dirección contraria a la dirección de visualización (*culling*) o que están fuera del espacio propio del *frustrum* (*clipping*). Más adelante se ampliará esta información.

El ejemplo siguiente, extremadamente simple, muestra, en pseudocódigo simplificado, el aspecto que tendría un programa para crear un vértice de color azul en la posición 1, 2. Los números tras las barras son comentarios y delimitadores de bloques que remiten al párrafo siguiente.

```
struct vertex {
/ 1
    tipoApi position;
    tipoApi color;
    tipoApi normal;
    tipoApi textura;
}
vertex v;   / 2
tipoApi matrizTransf;

vs (v.position)
/ 3
{
    nuevaPos = position * matrizTransf;
    return nuevaPos;
}
```

El primer bloque (1) define un tipo de dato complejo (*struct*) que incorpora tipos de datos básicos propios de la API. Por ejemplo, en Direct3D el tipo sería D3DVECTOR3, es decir, un vector de tres elementos.

El segundo bloque (2) define un vértice “v” a partir de la estructura anterior y una matriz, a partir de un tipo predeterminado en la API. Por ejemplo, en Direct3D, el tipo sería WORLDVIEWPROJECTION, una matriz de 4 x 4 *floats*. Cuando se transforma un vértice desde el espacio modelo al espacio de la vista se necesita llevar a cabo tres transformaciones canónicas: primero, por la matriz glo-

bal (*world matrix*), luego, por la matriz de la vista (*view matrix*) y, finalmente por la matriz de proyeccción (*projection matrix*). Pero estas transformaciones se pueden concatenar por medio de una matriz compuesta como la indicada.

El tercer bloque (3) es el *vertex shader*, "vs", propiamente dicho que, simplificado al máximo, toma un dato de entrada (la posición del vértice dado), lo multiplica por la matriz de transformacion y devuelve el resultado.

Para simplificar la exposición hemos eliminado palabras clave que ligarían estos valores al conjunto de la *rendering pipeline*. Si se utiliza, por ejemplo, el lenguaje CG de NVIDIA, la declaración primera deberían haber sido

```
float4 position : POSITION
float4 color : COLOR
```

Una sintaxis como esta indica que el valor de la variable "position" está ligado a la semántica propia de una determinada zona de la *rendering pipeline*, lo que permitiría ligar este fragmento de código a formas más complejas de manejo de los vértices.

Los programas que afectan a los vértices son procesados por una unidad especializada de hardware, el procesador de vértices programable (*programmable vertex processor*). El flujo de datos comienza cargando todos los atributos de los vértices: posición, color, coordenadas de textura, normales, etc. El procesador de vértices va cargando sucesivamente las instrucciones almacenadas en memoria y las va ejecutando hasta que el programa termina. Las instrucciones acceden sucesivamente a diferentes registros que contienen los valores correspondientes a los atributos de los vértices que son de solo lectura y han sido especificados por la aplicación principal. Los registros temporales son de lectura-escritura y se utilizan para almacenar valores intermedios de los cálculos. Los registros de salida son de solo escritura. Al terminar el trabajo del procesador, los registros de salida contienen los valores modificados de los vértices. Tras el proceso de rasterización, esos valores son pasados al procesador de píxeles/ fragmentos.

Las operaciones matemáticas que incluye el procesador son operaciones básicas de adición, multiplicación, producto vectorial punto, mínimos y máximos. Estas operaciones se han ampliado en los últimos años a producto vectorial cruzado, negación y reordenación de vectores (*component-wise swizzling*) y otros. Y más recientemente, a la inclusión de controles condicionales de flujo y otras operaciones.

Pixel / Fragment shaders

El segundo estadio que se ha estandarizado en los últimos años es el relativo al procesamiento de "píxeles" o "fragmentos". La diferencia de terminología viene de las dos API: DirectX utiliza la denominación *pixel shader* mientras que OpenGL utiliza la denominación *fragment shader*. Algunos teóricos prefieren esta última denominación, que es más correcta pues no hay una relación directa entre los píxeles de la imagen final de salida y los píxeles que se procesan, a menudo de modo iterativo, antes de la salida final: un ejemplo característico es la serie de valores correspondientes a objetos que compartirían un mismo píxel pero que están situados a diferentes distancias (con diferentes valores z), con lo que solo se conserva el valor del más cercano. Sin embargo, la denominación más extendida es la primera, que es la que utilizaré en lo que sigue.

Los *pixel shaders* describen datos relativos a los píxeles de la imagen de salida: color, profundidad (coordenada z), valores alfa. El dato principal es obviamente el color. Pero el color enviado desde la escena se transforma en función de información adicional, que se procesa en el estadio previo a la salida final. Entre la información que se procesa en este estadio está la información relativa a la iluminación o a datos que alteran virtualmente la geometría, como el relieve dado por *bumps* o *normal maps*,

por ejemplo. Esta información puede llevar a descartar ciertos valores o a combinarlos entre sí.

Las operaciones principales de este estadio son la rasterización de triángulos, la aplicación y procesamiento de texturas y, con algunas variantes según las API, operaciones complementarias de control como *antialiasing* o eliminación de píxeles en función de la información suministrada por diversos buffers (*depth*, *alpha*, *stencil*).

El ejemplo siguiente, extremadamente simple, como el anterior, muestra, en código simplificado, el aspecto que tendría un programa para procesar un píxel.

```
struct pixel {
/ 1
    tipoApi color;
    tipoApi otraInformacion;
}
pixel p;
/ 2
colAdd colorlum (datosCalculo);

ps (p.color, colAdd)
/ 3
{
    nuevoColor = p.color * colAdd;
    return nuevoColor;
}
```

El primer bloque (1) define un dato complejo (*struct*) que incorpora tipos de datos propios de la API. Por ejemplo, en Direct3D el tipo sería COLOUR, que es un vector de 4 elementos (*red*, *green*, *blue*, *alpha*) en coma flotante.

El segundo bloque (2) define un píxel "p" a partir de la estructura anterior. Y a continuación, un color adicional que podría ser el resultado de un cálculo de iluminación avanzada. O que podría ser otro píxel que se quiere combinar con el anterior para interpolar el resultado. U otras operaciones.

El tercer bloque (3) es el *pixel shader*, "ps", propiamente dicho. Toma dos valores como argumento, dos colores, y los combina para devolver el resultado "nuevoColor".

Este color se pasará al *frame buffer* donde, si no se modifica debido a procesamientos finales que veremos en la descripción general de la *rendering pipeline*, acabará por ser enviado al *driver* de la tarjeta gráfica y, de ahí, al monitor.

La estructura de los programas de procesamiento de fragmentos/píxeles son más simples que los de vértices pues, en definitiva, reciben un color de entrada y devuelven otro color. Naturalmente, en casos más complejos este código es más largo y acepta datos diversos así como diversas funciones de control, acceso a registros, etc. Pero la idea básica es crear estructuras programables que, en el fondo, son bastante simples para que se procesen de modo independiente y a gran velocidad.

Los programas que afectan a los fragmentos son procesados por una unidad especializada de hardware, el procesador de fragmentos programable (*programmable pixel/fragment processor*). El flujo general es similar al que se ha descrito antes para los *vertex shaders*.

Otros tipos de *shaders*

Además de estos tipos principales hay muchos otros tipos de *shaders* especializados. En el capítulo siguiente veremos múltiples ejemplos de diferentes *shaders* que se utilizan en programas de simulación para todo tipo de efectos de representación de materiales complejos o de efectos especiales.

A medida que estos *shaders*, que nacen por lo general en una determinada aplicación, en el software, van adquiriendo importancia, se van integrando en las tarjetas gráficas, en el hardware. Con esto, la velocidad de procesamiento se multiplica de un modo espectacular. Naturalmente, esto solo ocurre cuando el *shader* se ha consolidado, pues la inversión requerida para integrar un determinado algoritmo en el hardware es bastante más costosa.

Algunos ejemplos recientes notables, a los que me referiré más adelante, son los *shaders* de teselación, que permiten subdividir un objeto geométrico, virtualmente, en múltiples microfacetas para simular superficies curvas o para otras finalidades. Otros, más antiguos, que también se describirán más adelante, son

los correspondientes a procesos tales como enmascaramiento (*stencil shaders*), lectura de valores en profundidad (*depth shaders*) o fusión de valores provenientes de diferentes canales (*blending shaders*).

Las tuberías de la representación (*rendering pipeline*)

Descripción genérica. Fases principales

El término *rendering pipeline* se puede traducir, literalmente, como "tubería de la representación" y, aproximadamente, como "la serie de procedimientos que se encadenan para llegar a la representación". Es decir, la serie de operaciones normalizadas que se deben llevar a cabo para pasar de una estructura virtual, una descripción de entidades geométricas y atributos físicos, a una imagen sintética. Otra descripción aún más sintética es la siguiente: es el conjunto de operaciones que hay que llevar a cabo para convertir un modelo virtual 3D en una imagen 2D. La figura 3.6 resume este proceso simplificado.

Es un proceso similar al que seguiríamos para sacar una fotografía de una escena real pero con una diferencia importante. La escena que vamos a fotografiar no existe: no es sino una colección de números que solo pasarán a tener existencia virtual al final de este proceso.

En este apartado se resumen los estadios principales de dicho proceso sin entrar en excesivo detalle, pero procurando que queden claras las peculiaridades más generales de cada estadio. Es una descripción de la secuencia básica, dejando a un lado los avances recientes que afectan principalmente a la paralelización. Para simplificar la descripción se supondrá que la secuencia de operaciones es consecutiva. Como veremos más adelante, esto no es realmente así y la potencia de las GPU recientes y los avances en su programación se basan en que muchas de estas operaciones se realizan en paralelo.

La mayor parte de las operaciones implican especificar un espacio virtual, por medio de un sistema de coordenadas, con respecto al cual quedan definidos los objetos virtuales, y transformar este espacio, con su sistema de coordenadas, a otro espacio más adecuado para la fase correspondiente de este proceso.

Partimos de la base de que ya existe un modelo. Este punto de partida es un tanto anómalo pues en realidad se corresponde con el final del proceso: no podríamos crear un modelo, interactuando con el ordenador para definir geometría, colores, luces, cámaras, etc, si no contásemos con un programa que, a su vez, cuenta ya con una *rendering pipeline* que es, precisamente, la que hace posible esta labor.

Pero partamos de esta base: existe un modelo que ya hemos creado (o, si se prefiere, que existe misteriosamente) y que se envía a la aplicación que se encargará de enviarlo, a su vez, a la tarjeta gráfica que lo enviará, finalmente, al monitor.

Esto implica que existe una aplicación que actúa de intermediaria entre el programa que utilizamos para modelar (pongamos que 3ds Max, Maya, SketchUp o el que sea) y los dispositivos mecánicos (CPU, GPU y monitor) que transformarán esta información en pun-

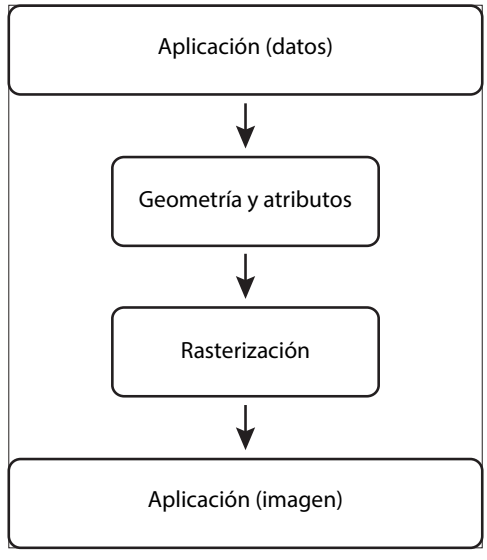


Figura 3.6 *La* rendering pipeline. *Esquema elemental.*

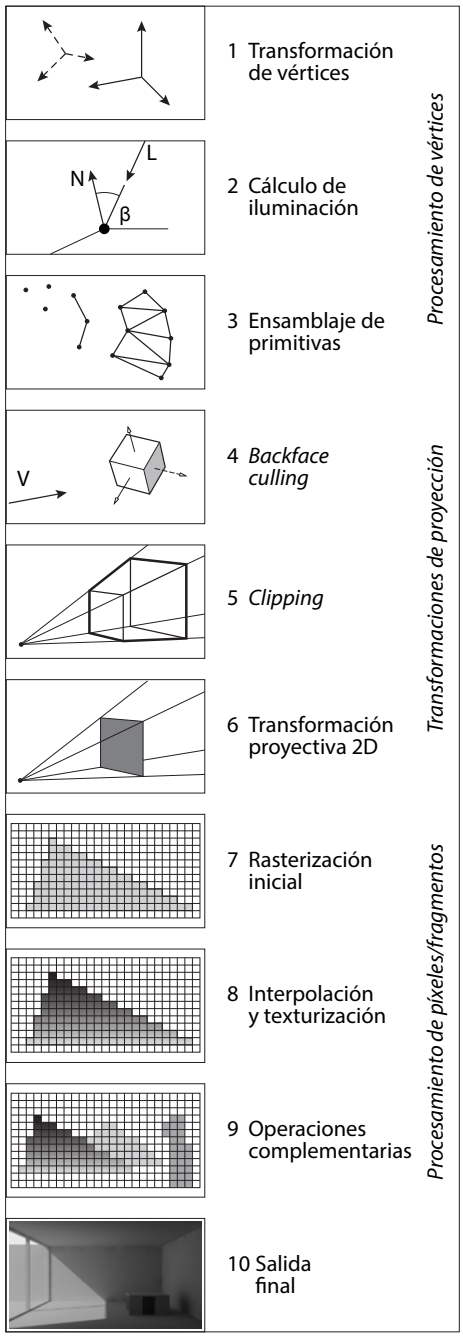


Figura 3.7 La rendering pipeline. *Esquema general.*

tos de color visibles. Como ya he avanzado y veremos con algo más de detalle, esto corre a cargo de una API (*application programming interface*) que se encarga de la tarea de hacer esta traducción.

Lo primero que hace la API es traducir los datos abstractos de alto nivel que hemos estado manejando, puertas, muros, terrenos, etc., a datos abstractos de bajo nivel: concretamente a *vértices*. Pues para la *rendering pipeline* este es el dato fundamental: una puerta no es sino una colección de vértices con atributos. De hecho, cuando se comenzó a modelar arquitectura en los departamentos más avanzados de Europa, en la década de 1970 aproximadamente, la información se introducía vértice a vértice.

Comenzando por tanto por aquí, los apartados siguientes irán describiendo las operaciones principales a un nivel muy general. La figura 3.7 ilustra los estadios principales que se describen algo más extensamente en los siguientes apartados.

Procesamiento de vértices y ensamblaje de primitivas

0 Procesamiento y evaluación de la información de entrada. Los datos recibidos por el programa situado a nivel superior deben procesarse adecuadamente para entrar en la *pipeline*. Hay toda una serie de procedimientos previos que definen el contexto en que se llevará a cabo toda la secuencia, y que concluyen con la recopilación normalizada de la información que se pasará al primer estadio propiamente dicho.

1 Transformación de vértices. La información principal por la que comienza el proceso es la posición geométrica de cada vértice y la conectividad con que se relaciona con otros vértices. Por añadidura, cada vértice incorpora información adicional, además de su posición geométrica, tal como el color, que resultará del color asignado por la aplicación superior, la normal derivada de la normal a la superficie en que está incluido o las coorde-

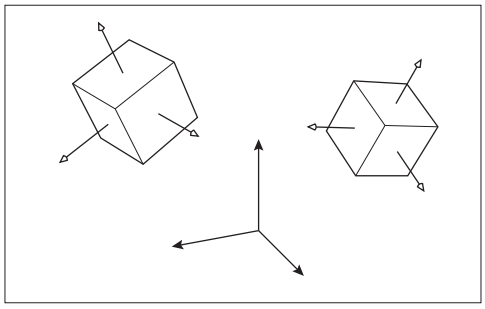


Figura 3.8 Transformación de vértices. Coordenadas de objeto y coordenadas globales.

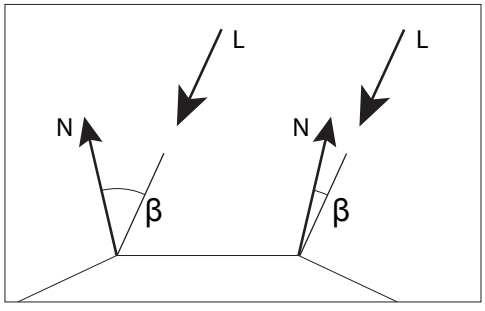


Figura 3.9 Transformación de vértices. Cálculo básico de iluminación.

nadas de textura que se hayan definido para este vértice, si es el caso. Estos datos vienen definidos corrientemente en el espacio objeto (*object space*). En este espacio hay un origen de coordenadas y unos valores que presuponen unas unidades determinadas. El origen (de coordenadas 0,0,0) puede ser el centroide del objeto (como ocurriría en un objeto esferoide) o, si resulta más conveniente, el centro de una cara (como ocurriría en un prisma que representa un muro apoyado en el suelo). Las unidades dependerán de las finalidades del modelo: pueden ser nanómetros, si se está trabajando en diseño microscópico, milímetros o centímetros, si se está modelando un objeto doméstico, centímetros o metros, si se está modelando una casa, metros o kilómetros, si se está diseñando un barrio o una ciudad, o parsecs, si se está diseñando una galaxia. Para el procesamiento interno esto es irrelevante pues, sean las que sean, acabarán siendo normalizadas, es decir, reducidas a un rango de 0,0 a 1,0.

Además de las coordenadas del espacio objeto, en una escena en la que hay varios objetos se necesita un espacio común a partir del cual se puedan relacionar los diversos objetos. Este espacio común se denomina espacio global (*world space*) que cuenta con su propio sistema de coordenadas, el sistema de coordenadas global (*world coordinate System*), con un origen que se especifica automáticamente por el programa de modelado que se esté utilizando y que se pasa a la API correspondiente. Las coordenadas de los objetos se transforman a las coordenadas del espacio global mediante una transformación específica denominada transformación de modelado (*modeling transformation*). Esta transformación puede implicar no solo cambios de posición sino también cambios

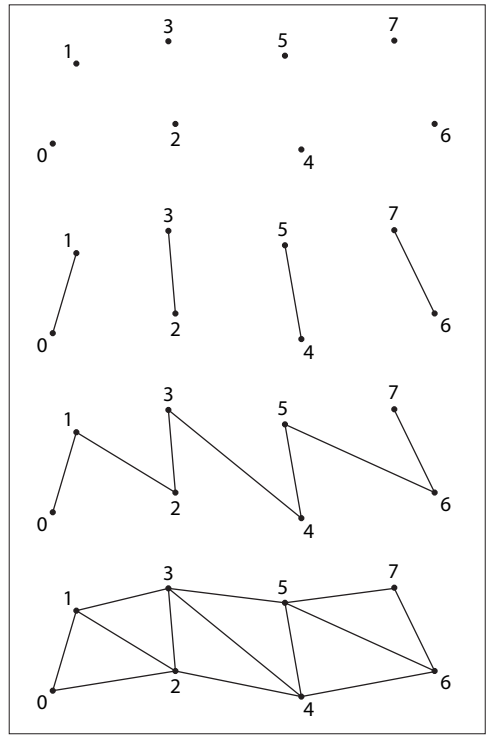


Figura 3.10 Ensamblaje de primitivas.

de orientación y cambios de escala, lo que implica una serie de transformaciones concatenadas.

2 Cálculo básico de la iluminación. La información suministrada por la aplicación incluye corrientemente la posición, intensidad y color de luces. Estas luces modifican el color de los vértices a partir de un cálculo elemental en el que se obtiene el producto vectorial punto del vector unificado correspondiente a la luz y el vector unificado correspondiente a la normal. Como el producto punto de dos vectores unificados es igual al coseno del ángulo que forman, es posible asociar directamente este resultado con la intensidad de iluminación recibida. Es evidente que si el ángulo que forman los dos vectores es 90º, el vértice no recibirá iluminación, lo que se corresponde con el valor del coseno, que es 0 en este caso. Y es evidente que si el ángulo es 0º, el vértice recibirá la máxima iluminación, lo que se corresponde con el valor del coseno que es 1 en este caso. Entre 0 y 1 se dan todas las variaciones de ángulos e intensidades de iluminación.

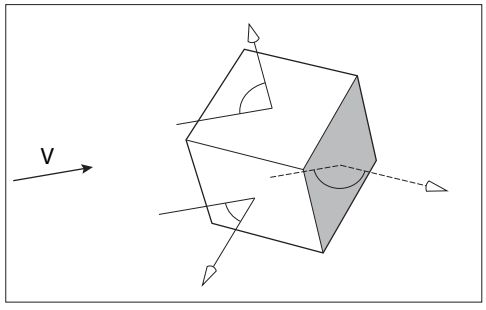


Figura 3.11 Culling.

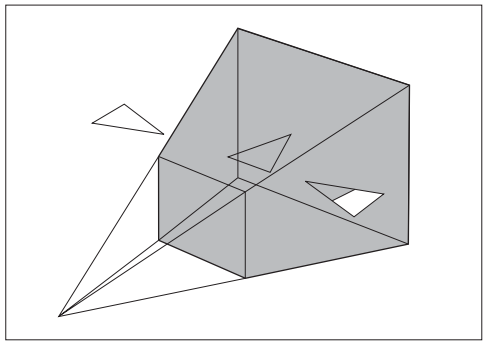


Figura 3.12 Clipping.

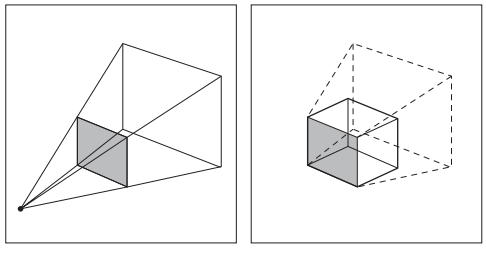


Figura 3.13 Viewport mapping.

El cálculo de iluminación y las transformaciones del modelo y las que siguen se integran, como veremos más adelante, en las aplicaciones recientes, en un único estadio de procesamiento de vértices, el *vertex shader*, que integra este cálculo y estas transformaciones por lo que también se le denomina de transformación e iluminación o T&L (*transform & lighting*).

3 Ensamblaje de primitivas. A partir de la información incorporada a la secuencia de vértices, estos se agrupan en primitivas: triángulos en la mayoría de los casos pero también líneas o puntos si la escena los incluye.

Estas primitivas se recortan o se eliminan a partir de operaciones adicionales de reordenación. Son operaciones que incluyen sistemas sofisticados de indexación para evitar redundancias y que en las aplicaciones recientes se apoyan en un *buffer* especial creado en el primer estadio de la *pipeline*, el *index buffer*. Si los triángulos que definen una superficie se computaran directamente, cada vértice se contaría dos o más veces. En una escena con decenas o centenares de miles de triángulos esto supone una carga de memoria considerable e innecesaria. Para evitarlo, se utilizan tablas que eliminan la redundancia pero definen perfectamente la topología del objeto.

A continuación siguen una serie de operaciones fundamentales que pueden ejecutarse en diferente orden del que sigue. Las opera-

ciones de *culling*, por ejemplo, pueden especificarse una vez que se conozca la posición del punto de vista, pues esto basta para definir la relación entre los vectores correspondientes al punto de vista y la normal a una cara. Otro tanto ocurre con las operaciones de *clipping*. Direct3D y OpenGL las ejecutan en diferente orden, por razones técnicas en las que no es necesario entrar pero que describiremos de modo general más adelante. Para que sean más comprensibles las diferentes fases aquí se presentan de un modo diferenciado.

Transformaciones de proyección

4 Transformación perspectiva. *Culling*. Tras este proceso, el espacio modelo se proyecta sobre un espacio perspectivo con un nuevo origen: el correspondiente al punto de vista. Los triángulos más alejados del punto de vista resultarán más pequeños, después de esta transformación, que los más cercanos. Las coordenadas vienen dadas en el espacio "del ojo" o "de la camara" y se denominan coordenadas de ojo o de cámara (*eye space* o *camera space* y *eye coordinates* o *camera coordinates*).

Por añadidura, las coordenadas de todos los objetos se transforman en coordenadas homogéneas (se les añade un cuarto componente, *w*) y se reproyectan sobre un espacio normalizado (un cubo unitario). Así, todos los objetos tendrán coordenadas comprendidas entre (-1,-1,-1) y (1,1,1). Este espacio se denomina NDS, *normalized device space* y sus coordenadas correspondientes NDC*, normalized device coordinates*.

En este estadio se lleva a cabo también un proceso denominado *culling* o *backface culling*. Los polígonos cuyas normales forman un ángulo mayor de 90º con el vector correspondiente al punto de vista se eliminan, pues corresponden a caras no visibles de un volumen cerrado o a caras que, por alguna razón (incluyendo errores del usuario), tienen esta orientación.

5 *Clipping* (*frustrum clipping*). El espacio resultado de la transformación perspectiva está limitado por los cuatro planos correspondientes a la pirámide de visión. Todos los objetos o partes de objetos que queden fuera del espacio dado por esta pirámide, se eliminan. Por añadidura, el usuario puede haber especificado dos planos de recorte perpendiculares a la línea de visión, uno más cercano y otro más lejano, para que los objetos situados fuera de estos planos también se eliminen.

La escena visible corresponde, por tanto, a todo lo que queda dentro de este espacio delimitado por 6 planos, cada uno de los cuales define un hemiespacio, y que se denomina *frustrum* (del latín "fragmento", "trozo"). Todos los polígonos que queden fuera del *frustrum* se eliminan o se recortan.

6 Transformación proyectiva 2D (***viewport mapping***). Tras todas estas operaciones y transformaciones, el modelo está listo para ser proyectado sobre el marco de salida con lo que los objetos 3D se convierten en objetos 2D. Los píxeles definidos por el marco de salida también se normalizan y quedan definidos en el espacio de la ventana (*window space*) con sus correspondientes coor-

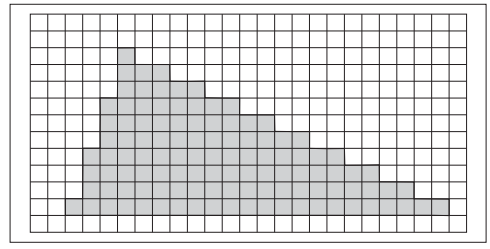


Figura 3.14 Rasterización inicial.

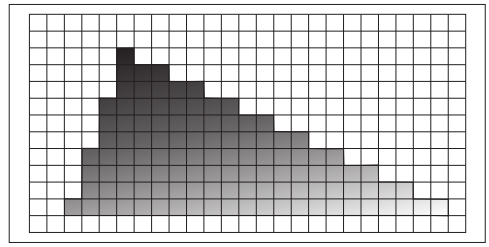


Figura 3.15 Interpolación y texturización.

denadas de ventana (*window coordinates*). Los valores están comprendidos entre (0,0) y (x-1, y-1), siendo x e y las dimensiones de la ventana. El origen varía según las API. En OpenGL está en la parte inferior izquierda, en Direct3D está en la parte superior izquierda de la pantalla.

Esta última transformación se lleva a cabo por una matriz específica, la matriz de visión y la operación que transforma valores dados en el *normalized device coordinates* a *window coordinates* se denomina proyección o transformación de visor o *viewport mapping* o *viewport transformation*.

Procesamiento de píxeles/fragmentos

7 Rasterización inicial. El resultado de todo el proceso anterior es que a cada primitiva le corresponderá un conjunto de fragmentos/píxeles. Píxeles (*pixels*, el término utilizado por Direct3D) y fragmentos (*fragments*, el término utilizado por OpenGL) son términos casi sinónimos pero hay una diferencia importante de matiz que ya hecomentado y que está relacionada con la diferencia de terminología de Direct3D y OpenGL. Un píxel se corresponde exactamente con una posición y un valor almacenados en memoria (en el *frame buffer*) para ser proyectados en la imagen final. Un fragmento se corresponde con la serie de valores a partir de los cuales se actualiza un píxel. Viene a ser, por tanto, como un píxel potencial, como el valor de un píxel que puede variar en función de los cálculos que se llevan a cabo por los *shaders* correspondientes en este nivel de procesamiento. Los píxeles/fragmentos pasan por una serie de operaciones antes de ser enviados al *buffer*.

8 Interpolación y texturización. Una vez que se cuenta con una colección de píxeles/fragmentos que corresponden a los colores previos de los vértices, se pasa a un proceso de interpolación y, en su caso, de proyección de texturas almacenadas en memoria. La interpolación consiste en la deducción de los valores de los píxeles intermedios que se crean entre dos píxeles cuyos valores son conocidos pues se corresponden con vértices proyectados directamente desde la escena. La gestión de texturas es una de las funciones principales de las tarjetas gráficas en la medida en que introducen aceleración por hardware en elementos muy grandes y que consumen una gran cantidad de recursos. Es el aspecto en que probablemente más avances se han llevado a cabo en los últimos años. Las texturas se aplican por medio de funciones progresivamente más complejas a medida que evolucionan las aplicaciones y que permiten llevar a cabo substituciones, recortes, mezclas, combinaciones diversas, transformaciones de traslación, rotación, inversión o cambio de escala, por medio de matrices específicas, etc. Los valores resultantes del acceso a las texturas almacenadas en memoria y manipuladas o combinadas de diversos modos se fusionan con un proceso de aplicación de la textura en función de otras entradas en el proceso debidas al cálculo de iluminación avanzada o a otras modificaciones. El resultado de este proceso es que cada fragmento y, en consecuencia, cada píxel recibirá una nueva asignación de color.

Operaciones complementarias y salida final

9 Operaciones complementarias y finales de rasterización. Si la escena incluye especificaciones especiales se procesan en este estadio previo al final. Si, por ejemplo, la escena incluye niebla (*fog*), el valor de cada fragmento se modifica en función de la distancia a la cámara y en función de los parámetros asociados a este efecto.

El último estadio es procesar una secuencia de pruebas de control que están incorporadas tanto a DirectX como a OpenGL. Estas pruebas han evolucionado en las últimas aplicaciones por lo que daré una información breve que se matizará más adelante:

a) *Alpha test*. Los píxeles con un determinado valor alfa son eliminados o atenuados. Esta prueba se ha integrado en otras en versiones más recientes;

b) *Depth test*. Los píxeles cuyas coordenadas de proyección *x* e *y* coinciden con las de otro píxel pero cuya coordenada *z* es mayor son eliminados (pues quiere decir que se proyectan sobre la misma posición respecto al punto de vista y quedarían ocultos por el de coordenada *z* menor). Normalmente este test se realiza en colaboración con un sector de memoria especial, el *z-buffer,* que almacena estos valores para acelerar este procesamiento. Las tarjetas modernas utilizan otro *buffer* más preciso y más efectivo denominado *w-buffer*;

c) *Stencil test*. Los píxeles se enmascaran en función del contenido del *stencil buffer*. Un *stencil buffer* almacena valores enteros, por lo general de 1 byte per píxel, de tal modo que el píxel se representaría con uno u otro color en función de este valor. El término deriva de que actúa como una plantilla superpuesta al área de *rendering*. Se utiliza en combinación con el *depth-buffer* y sus aplicaciones más características están ligadas al cálculo de sombras en aplicaciones 3D (principalmente *shadow volumes*) y también a reflexiones planas, pues permite enmascarar las zonas de las superficies reflectantes evitando errores (como reflejos que "se saldrían" de la superficie). Hay muchos otros usos, tales como ocultar un objeto o representar una parte del conjunto, etc. En OpenGL es un test que se lleva a cabo antes del *depth test*.

d) *Blending*. Un píxel puede pasar las pruebas anteriores de un modo total o parcial. En este último caso, los valores del píxel deben fusionarse con los valores de otros píxeles. En el caso característico de un objeto semitransparente, parte de sus valores se retienen después del *depth test* y deberán combinarse con los del objeto que esté en su misma línea de visión. Esta mezcla corre a cargo de esta operación. Hay varios métodos de *blending*. Los principales son los siguientes: a) adición (los valores de los píxeles se suman), b) substracción (los valores de los píxeles se restan), c) multiplicación (los valores de los píxeles se multiplican), d) fusión alfa (*alpha blending*: se utilizan los valores alfa de una de las fuentes para combinarlos con los de la otra fuente).

Además de estas operaciones principales, pueden encontrarse también las siguientes

— *Pixel ownership test*. Se utiliza para averiguar si el fragmento especificado (en el *frame buffer*) pertenece al contexto GL. Si no es así, el sistema de ventanas decide la suerte final de este píxel. Puede que se descarte o que pase al siguiente test.

— *Scissor test*. Es un recorte (*clipping*) adicional, en 2D, que se lleva a cabo para comprobar si los fragmentos quedan dentro del rectángulo de la ventana de salida.

— *Multisample fragment operations*. Se lleva a cabo para eliminar el efecto de *aliasing* en los bordes de las primitivas. Se denomina así porque el *buffer* contiene varias muestras por píxel, cada uno de ellos con sus valores propios de *color*, *depth* y *stencil*.

— *Occlusion query*. Este test comprueba el número de fragmentos o muestras que han pasado el *depth test*. Esta información es útil para determinar la visibilidad de los objetos. Si, por ejemplo, un objeto determinado no incluye ningún fragmento que haya pasado el *depth test*, esto significa que esta completamente oculto por otros objetos.

— *sRGB Conversion*. En caso de que esté activado este espacio de color, los colores resultantes de las operaciones anteriores se convierten a este espacio de color.

— *Dithering*. Cuando los colores no son colores reales (24 bpp o más) sino que están basados en tablas de color (LUT, *Look Up Tables*, 8 bpp), la representación se basa en esta técnica que distribuye los puntos con arreglo a determinados patrones para simular la apariencia

de color real. En general, el programador no decide este patrón (basado en algoritmos estandarizados) sino que esto corre a cargo de la GPU que se esté utilizando.

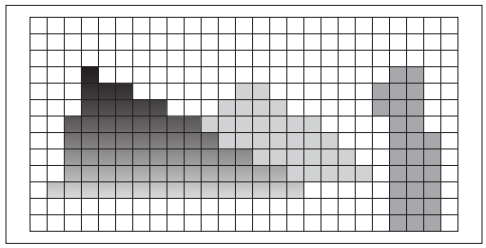


Figura 3.16 Operaciones complementarias.

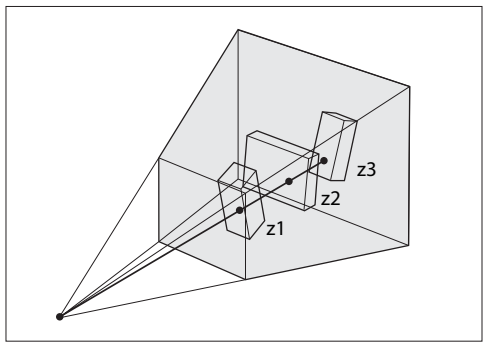


Figura 3.17 Depth test.

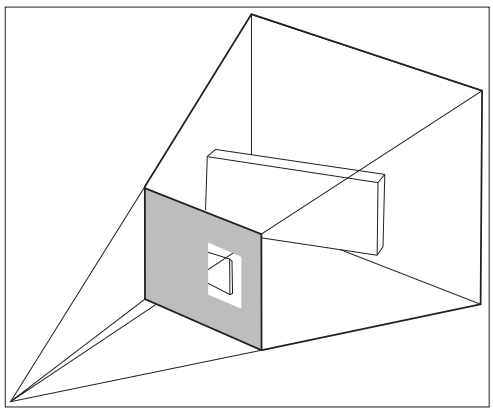


Figura 3.18 Stencil test.

— *Logical operations*. Es una alternativa que, si se activan las opciones correspondientes, substituye a *blending* y lleva a cabo operaciones de tipo *and*, *or*, *xor*, etc., sobre los fragmentos que llegan aeste estadio.

Por otro lado, hay tests que se han eliminado o se han desplazado. Por ejemplo, la versión 9.0 de Direct3D incluía un *alpha test* que descartaba píxeles con valores alfa (transparentes). En la versión 10.0 este test desapareció y se incorporó en parte al *pixel shader* y, en parte, al *depth* / *stencil shader*.

En Direct3D 10 el *output merge* incorpora principalmente las operaciones de *depth test* y *stencil test* que se basan en un *buffer* unificado, así como el *blending* de los diferentes *render targets*. El *pixel shader* puede dar como salida diferentes valores para diferentes *render targets*. Estas *render targets* comparten una misma función de *blending*, pero esta función puede activarse o desactivarse para cada una de las *render targets*.

En OpenGL 3.0 las operaciones que se agrupan en la categoría *fragment operations* siguen el siguiente orden estricto: 1 *Pixel ownership test*, 2 *Scissor test*, 3 *Multisample fragment operations*, 4 *Stencil test*, 5 *Depth test*, 6 *Occlusion query*, 7 *Blending*, 8 *sRGB conversion*, 9 *Dithering*, 10 *Logical operations*.

10 Salida final. Por último, los resultados que sobreviven a todas estas pruebas se envían al *back buffer* que mantiene la información preparada para ser enviada al *front buffer*, a la salida final. Contiene información que es substituida por esta nueva entrada. Pero también es posible fusionarla (*blending*) con la nueva. Esta operación se denomina, en Direct3D, *render target blending*. Pueden especificarse varios *render targets* (*buffers* de atributos). El *pixel shader* produce valores independientes para cada *render target*. La fusión (*blending*) puede activar o desactivarse de modo independiente por cada *render target*.

En cualquier caso, el resultado de todo este proceso es una imagen que se envía al *frame buffer* y, de este, a la pantalla del dispositivo concreto que se vaya a utilizar.

3.2 Estructuras mediadoras entre hardware y software

Las dos grandes API

Una API (*application programming interface*) es, en general, un programa especial que incorpora un conjunto de funciones y procedimientos que actúan como intermediarios para comunicarse con otro programa. Estas funciones y procedimientos se organizan en bibliotecas (*libraries*) que quedan a disposición de los desarrolladores y que facilitan su trabajo al permitirles pasar por alto detalles tediosos.

En el caso de la programación de funciones gráficas, esta necesidad surgió en el momento en que había que repetir una y otra vez grandes cantidades de código para adaptarlos a dispositivos específicos. La primera empresa que adoptó este procedimiento fue Silicon Graphics, y así nació lo que luego sería OpenGL. Pronto le siguió otra empresa, Microsoft, y así nacieron DirectX y Direct3D. Estas dos API han dominado el mundo de las aplicaciones gráficas desde entonces. En este apartado se describe a grandes rasgos su evolución y sus características más generales.

Visión conjunta de la evolución histórica de OpenGL y DirectX

Para crear un juego, un escenario interactivo o una aplicación que permita simular procesos de todo tipo había, hace unos cuantos años, dos alternativas. La primera consistía en programarlo todo desde el comienzo. Esto suponía escribir enormes cantidades de código, una buena parte del cual estaría dedicado a hacer cosas tan básicas como agrupar vértices, computar sus modificaciones de color, relacionarlos con texturas, aplicarles transformaciones básicas y proyectivas, etc. La otra alternativa, bastante más lógica si se tiene en cuenta que gran parte de este código es el mismo para cualquier tipo de aplicación, era utilizar funciones predeterminadas y adaptarlas a la maquinaria concreta que se estuviera utilizando. Pero este último paso se puede separar en dos partes aunque, en algún caso, sea algo menos eficiente: funciones genéricas que sirven para cualquier máquina y funciones específicas que sirvan para una máquina particular. Esta separación es algo menos eficiente, porque hay que escribir más código y el resultado es menos compacto e implica más llamadas a funciones. Pero tiene la enorme ventaja de que podemos reutilizar un enorme número de funciones genéricas. Esto es lo que hacen las API. Proporcionan funciones genéricas que pueden ser entendidas de un modo automático por los *drivers* de los diferentes fabricantes. Un *driver* es un pequeño programa que se encarga de tomar las funciones genéricas y aplicarlas a una máquina particular por medio de funciones específicas para esa máquina.

OpenGL apareció en 1992. Era la respuesta de la empresa líder en aplicaciones 3D en aquellas fechas, Silicon Graphics, al problema principal que se había creado con el desarrollo de estas aplicaciones. Los ordenadores tenían componentes distintos y las empresas de desarrollo de software tenían que escribir versiones diferentes de sus aplicaciones insertadas en programas especiales, en *drivers* que traducían los modos en que las funciones generales tenían que aplicarse en cada caso.

Se habían propuesto soluciones basadas en estándares institucionales, independientes de marcas concretas. El más famoso era PHIGS (*Programmer's Hierarchical Interactive Graphics System*), que se diseñó en la década de 1980 y se convirtió en un estándar ISO en 1989, y del que se proporciona una amplia descripción en una de las obras principales sobre computación gráfica, la de Foley *et al*. (1990). Pero es notorio que estos es-

fuerzos siempre han ido por detrás de la evolución comercial. La aplicación propia de Silicon Graphics en aquella época (IRIS GL) era muy superior, entre otras cosas porque era de uso más simple, y se había convertido en un estándar de hecho. Pero otras empresas poderosas, como Sun y Hewlett-Packard, tenían también sus propias aplicaciones gráficas, basadas en extensiones propias de PHIGS. Esto llevó a Silicon Graphics a sacar un estándar abierto (no lo podía hacer con IRIS por problemas técnicos y legales).

Al dejar disponible este primer estándar, Silicon cedía parte de su trabajo a otras empresas. Pero como contrapartida se liberaba de la responsabilidad de mantener actualizados lo *device drivers* para todo tipo de tarjetas gráficas y programas comerciales. Ese mismo año Silicon, creó un consorcio, el OpenGL ARB (Architectural Review Board), un grupo de compañías que se dedicarían a mantener y revisar este primer estándar y los que le siguieran.

Esto tuvo unas consecuencias muy importantes pues a partir de esa fecha todos los desarrolladores podían hablar en un lenguaje común. Sin embargo, las funciones comunes que quedaban disponibles eran muy limitadas. A partir de ahí comenzó una evolución que, en paralelo con la propia evolución de las tarjetas gráficas, fue incrementando progresivamente el abanico de funciones y la propia accesibilidad a estas funciones. Y comenzó también la competencia con Microsoft, pues esta compañía sacó su propio estándar, DirectX, en 1995, como respuesta a la decisión de OpenGL.

Hubo un intento fallido de elaborar un estándar común. En 1997, Microsoft y Silicon iniciaron un proyecto, denominado proyecto Fahrenheit, para elaborar una interfaz común, un proyecto al que también se unió Hewlett-Packard en 1998. Pero los problemas financieros y la realidad de una competencia muy dura entre las empresas acabaron por abortar este proyecto en 1999.

La NVIDIA GeForce 256, que salió en 1999, fue la primera tarjeta gráfica que introdujo funciones ligadas a este tipo de desarrollos. La integración de funciones de transformación e iluminación, que ya existían en OpenGL, fue el primer paso hacia una integración más compleja que llevaría a la integración de unidades de *pixel shader* y *vector shader*, más flexibles y programables.

Las primeras funciones que se integraron fueron la proyección de texturas y la representación de polígonos. Posteriormente se añadieron funciones para acelerar los cálculos geométricos implicados en operaciones básicas de traslación y rotación de vértices. El paso principal fue el soporte para *shaders* programables que permitían manipular vértices y texturas del mismo modo que se haría con programas que actuaban desde la CPU. A esto se añadieron también mecanismos de soporte de técnicas de *supersampling* e interpolación.

A principios de la década de 2000, el líder era OpenGL seguido a corta distancia por Direct5 y luego por muchos otros (Apple Quickdraw 3D, Argonaut Brender, Criterion Renderware, Rendermorphics, Reality Lab…), incluido Intel que sacó su propia API, 3DR, que se suponía que daba soporte común a través de los Pentium, hasta que Microsoft le convenció de sacar un producto común para Windows 95.

Sin embargo, a partir de la aparición de Direct3D 9, en 2002, la situación se modificó. Los críticos tuvieron que reconocer por primera vez que esta versión era al menos tan buena como la de OpenGL y que Microsoft había hecho algo más que replicar muchas de las funciones de OpenGL. Entre otras cosas, introducir un nuevo lenguaje, HLSL, que facilitaba aún más el trabajo de los programadores. De hecho, OpenGL tuvo que introducir importantes cambios en su estructura para, entre otras cosas, llegar a gestionar las texturas con eficiencia. La versión 2.0 que apareció en 2004 recogió parte de estos necesarios cambios pero que no iban mucho más allá que mantenerse a la misma altura que Direct3D 9. En 2006, ARB, el consorcio creado por Silicon, dejó la gestión de estos cambios en manos de Khronos, otro consorcio que

nació en 2000, por iniciativa de una serie de líderes en el campo de la computación gráfica (3Dlabs, ATI, Discreet, Evans&Sutherland, Intel, NVIDIA, SGI y Sun) para crear estándares abiertos que sirvieran para todo tipo de plataformas y dispositivos. En 2008 apareció la versión 3.0, que suponía una cierta ruptura con las versiones anteriores y una renovación importante. Pero, según los críticos, no suficiente, pues se debería haber acometido una renovación radical lo que, por lo que parece, no fue posible debido a desacuerdos entre los promotores.

En cualquier caso, a lo largo de los últimos años las posiciones se han equilibrado y han quedado como las dos API hegemónicas si bien la aparición de nuevas tecnologías hace prever cambios muy importantes en los próximos años. En los apartados siguientes se resume la evolución de cada una de las dos grandes API.

Diferencias principales

Direct3D es una aplicación creada por una empresa, Microsoft, para funcionar sobre una plataforma concreta, Windows. Puede utilizarse en otras plataformas pero no de modo directo ni con toda la funcionalidad. Es un subconjunto de DirectX que se describe más adelante. OpenGL es una aplicación abierta que está disponible para todos los sistemas operativos corrientes (Windows, Mac, Linux…). Tambien acepta extensiones desarrolladas por otros programadores que se registran oficialmente en la OpenGL Extension Registry. El registro especifica convenciones de nomenclatura y recomendaciones para desarrollar extensiones. Sin embargo, a diferencia de Direct3D, requiere bibliotecas específicas para hacer cosas tales como crear una ventana y eliminarla al final. Esto corre a cargo de la GLUT, que se describe más abajo.

Direct3D utiliza un lenguaje de programación orientado a objetos y basado en clases. Mantiene compatibilidad desde sus inicios con COM (*component object model*), el componente introducido en 1993 por Microsoft para facilitar la comunicación entre programas. OpenGL no es orientado a objetos. Inicialmente, OpenGL se construyó en torno a otro concepto muy poderoso, el de “máquina de estados finitos”, aunque posteriormente ha evolucionado hasta ser muy similar a un sistema basado en objetos. Una máquina de estado finito, o FSM (*finite-state machine*) o “automáta de estado finito”, es una abstracción matemática que se utilizaba, y se utiliza, para diseñar programas digitales. Se basa en la idea de un modelo concebido como una serie finita de estados, de transiciones entre esos estados y de acciones. Las operaciones comienzan a partir de un estado inicial (*start state*), pasa por una serie de transiciones y termina con uno de los posibles estados incorporados al sistema.

Direct3D está diseñado de tal modo que el hardware está virtualmente presente y el programador puede olvidarse en parte de la necesidad de acomodar el programa a los dispositivos, si bien complica algo más la programación pues debe acomodar las funciones al hardware virtual. La implementación resulta más sencilla pues parte del trabajo ya está hecho. OpenGL está diseñado de tal modo que el hardware no se tiene en cuenta y el programador puede olvidarse por completo de este asunto. Pero la implementación resulta más complicada pues es necesario acomodar la aplicación al dispositivo concreto que se vaya a utilizar.

Direct3D está especialmente preparado para juegos y aplicaciones interactivas pues nació con esta finalidad claramente presente. OpenGL nació con un amplio abanico de posibles aplicaciones que no incluían los juegos y aplicaciones interactivas como prioridades. Su gama de funciones es más amplia y esta es una de las razones, junto con su independiencia de Windows, por la que es preferido en muchas aplicaciones científicas y académicas.

Direct3D no tiene un sistema de extensiones como el de OpenGL. Pero tiene un sistema de diferentes API con un núcleo fijo de

características, y mantiene la compatibilidad con versiones previas por medio del COM de Microsoft. OpenGL cuenta con un sistema sofisticado de extensiones que permite a los desarrolladores de hardware implementar sus propias extensiones a las especificaciones de OpenGL lo que, a su vez, permite a los desarrolladores de OpenGL acceder antes a nuevas funciones de hardware. El OpenGL ARB puede hacer que una extensión sea semioficial dándole soporte antes de que se incorpore oficialmente a las especificaciones. A partir de la versión 3.0 se eliminaron muchas funciones y características obsoletas.

DirectX se distribuye de dos modos: *runtime* y *SDK*. El primer modo, *runtime*, se descarga por los usuarios para poder utilizar aplicaciones que usan DirectX o viene incorporado directamente al sistema operativo. El segundo, *SDK* (*software development kit*) es un módulo completo que incluye todas las herramientas y bibliotecas de funciones necesarias para desarrollar aplicaciones propias. El programa y la documentación completa de Direct3D puede descargarse de: http://www.microsoft.com/games/en-us/aboutgfw/pages/directx.aspx.

OpenGL está compuesta por tres módulos principales: GL, el núcleo básico de funciones de OpenGL; GLU (*GL Utilities*), utilidades y funciones auxiliares; GLUT (*GL Windows interface utility toolkit*), incluye diversas herramientas para la gestión de la interfaz. Y hay otras como GLX, AGL, WGL. Las convenciones de nomenclatura son sencillas. Un nombre como *glColor3f*… implica cuatro campos: *a*) "gl", la biblioteca de funciones utilizada, *b*) "Color" la raíz principal, *c*) "3" el número de argumentos, *d*) "f" el tipo de dato (f float, i integer, v vector…). La documentación oficial de OpenGL puede descargarse de: http://www.opengl.org/.

Evolución histórica de las diferentes versiones de OpenGL y Direct3D

A partir de 1992 en que, como ya he dicho, apareció OpenGL, fueron apareciendo diferentes versiones y variantes de versiones. Las características principales se resumen brevemente en los apartados siguientes. La numeración de versiones es algo equívoca, pues hubo un gran número de subversiones de la primera versión que abarcaron 11 años. Pero lo mantengo para ordenar la exposición. Aunque la fecha inicial que se da corrientemente es 1992, hasta 1993 no estuvieron realmente disponibles los primeros productos. En 1992 se creó también el OpenGL ARB para seguir el desarrollo de este estándar y prevenir, entre otras cosas, conflictos legales. Las primeras comisiones incluían representantes de SGI (Silicon), Intel, Microsoft, Compaq, Digital Equipment Corporation, Evans & Sutherland e IBM. En los últimos años los miembros participantes han sido 3DLabs, Apple, ATI, Dell, IBM, Intel, NVIDIA, SGI y Sun Microsystems. La primera versión incluía todas las herramientas básicas que se mantienen en la actualidad: transformaciones de vértices y primitivas, modificación de colores de vértices por asignaciones directas y por modificaciones derivadas de cálculos de iluminación, transformaciones de objetos y proyecciones, *clipping*, rasterización, operaciones sobre fragmentos y píxeles, operaciones sobre los *frame buffers*, etc. En 1997 apareció la versión 1.1. Su principal innovación fue la adición de soporte para *arrays* de vértices y texturas así como funciones para copiar y gestionar texturas de diversos tipos. La versión 1.2 (1998) añadía recursos para texturas 3D y algunas funciones de procesamiento de imágenes (matrices de gestión de colores, funciones de convolución, de control de histograma, de fusión de colores) y algunas funciones especiales tales como control de LOD (*level of detail*) para texturas. Una variante posterior (1.2.1) incluyó también control de texturas múltiples. La versión 1.3 (2001) incorporaba funciones adicionales para la gestión de texturas múltiples, mapas cúbicos, muestreo múltiple, y funciones adicionales paa combinar texturas. La versión 1.4 (2002) añadia gestión de sombras por hardware, coordenadas para procesamiento de efectos de niebla, gene-

ración automática de *mipmaps* y algunas funciones adicionales para el control de texturas. También incluía funciones tales como *stencil wrapping*. La versión 1.5 (2003) incluía funciones adicionales para el control de sombras, soporte para *vertex buffers* y control de oclusiones. También permitía utilizar texturas con dimensiones que no fueran potencias de 2.

OpenGL 2 (2004-2008) apareció en septiembre de 2004 y su principal diferencia con las anteriores fue que incluía soporte para un lenguaje ensamblador realmente basado en la GPU, denominado ARB pues fue desarrollado por el consorcio del mismo nombre al que he aludido antes. Este lenguaje se convertiría en un estándar de hecho para la implementación de *vertex shaders* y *fragment shaders*. Los responsables principales del diseño de este lenguaje fueron 3DLabs, una compañía que fabricaba tarjetas gráficas y que era pionera en la adaptación de las tarjetas gráficas a las aplicaciones 3D. Aunque el lenguaje ARB es de bajo nivel, se relaciona directamente con lenguajes posteriores basados en C con los que es posible escribir *shaders*, concretamente con lo que también apareció por estas fechas y que acabaría por ser otro nuevo estándar de OpenGL, el lenguaje GLSL (*OpenGL shading language*) y que permitiría a los programadores substituir las funciones fijas para vértices y fragmentos por *shaders* personalizados escritos en un lenguaje muy similar a C. Las tarjetas gráficas que se fabricaron con soporte para este nuevo estándar fueron las primeras que admitían la programación de *shaders* a nivel de hardware lo que supuso una nueva revolución pues abría el camino para superar la rigidez de las tarjetas anteriores que solo incluían las funciones selecionadas por los fabricantes. Las limitaciones, sin embargo, seguían siendo importantes. Esta versión incorporaba, además de funciones mejoradas para *shader objects*, *vertex shaders* y *fragment shaders*, otras funciones adicionales tales como *point sprites*, *separate stencils* y funciones adicionales de fusión (*blending*) y *rendering*. En 2006 apareció la versión 2.1; esta versión incluía soporte para *pixel buffers* y texturas sRGB (texturas con corrección de color según este espacio de color). Incluía también una versión revisada (1.20) del lenguaje GLSL.

OpenGL 3 (2008-2010) apareció en julio de 2008. Incluía complementos para cosas que ya habían aparecido en la versión anterior, como *frame buffer objects*, *vertex array objects* o sRGB. Además incluía funciones matemáticas más precisas basadas en formatos de números en coma flotante de 32 bits. Y otra versión revisada (1.30) del lenguaje GLSL. Esta versión, desarrollada en colaboración con Khronos, supuso una ruptura importante pues, hasta entonces, la compatibilidad con versiones anteriores se había mantenido. A partir de esta versión varias funciones y recursos se eliminaron como obsoletos. En marzo de 2009 apareció la versión 3.1; en agosto de 2009 la versión 3.2, y en marzo de 2010 la 3.3. Estas nuevas versiones incorporaban cosas tales como nuevas funciones para la gestión más eficaz de texturas o instancias de objetos basadas en la reutilización de datos de vértices; soporte para *geometry shaders*, funciones adicionales de control de *fragment shaders*, filtrado de mapeados cúbicos sin costuras (*seamless*) y *fragment depth clamping*. Y nuevas versiones (1.40, 1.50) del GLSL.

OpenGL 4 apareció en marzo de 2010. Incluía nuevos estadios de *shaders* que permiten descargar teselación geométrica de la CPU, mejoras en la salida de *rendering* y *antialiasing*, mayor flexibilidad de organización de *shaders* por medio de subrutinas, separación de datos de textura mediante un nuevo tipo de objeto denominado *sampler object*, operaciones matemáticas basadas en datos en coma flotante con precisión de 64 bits y una nueva revisión del lenguaje GLSL

§ § §

Como ya he dicho, Microsoft reaccionó a la publicación en abierto de la primera versión

de OpenGL sacando a la luz su propio estándar en 1995. Pero esta decisión también estuvo motivada por problemas que emergieron a partir de la salida de Windows 95, que no estaba preparado para dar soporte adecuado a aplicaciones de videojuegos o de simulación 3D, debido principalmente a restricciones de acceso a la memoria base y a que el propio sistema estaba orientado a otro tipo de aplicaciones. La solución provisional fue DirectX, que fue evolucionando hasta convertirse en un conjunto de componentes o API dirigidas a diferentes tipos de aplicaciones.

Unos de los principales componentes de DirectX es *Direct3D*. Es el producto principal al que me referiré en lo que sigue; y proporciona un conjunto de funciones que permiten programar algoritmos de diversos tipos para aplicaciones de dibujo de entidades gráficas y de modelado 3D y simulación visual, que proporcionan aceleración por hardware, en función de las disponibilidades de la tarjeta gráfica con que se cuente. Otros componentes de DirectX eran DirectDraw y Direct2D. En aplicaciones 2D se utilizaba el módulo DirectDraw para acelerar la representación de gráficos por medio de funciones incorporadas al hardware que podían acceder directamente a la memoria de vídeo o utilizar toda una serie de algoritmos básicos. A partir de la versión 8.0 de DirectX pasó a integrarse en un nuevo módulo denominado DirectX Graphics que realmente era Direct3D con una serie de funciones adicionales propias de DirectDraw. DirectDraw se sigue incluyendo pero sin actualizaciones, y está en desuso y ha sido reemplazado por Direct2D o Direct3D. Con Windows 7 ya se utiliza Direct2D. Y otros componentes, para música, productos multimedia, etc., sobre los que se puede encontrar más información en internet, son DirectInput (proporcionaba un interface para interactuar con dispositivos de entrada de datos como teclados, ratones, joysticks, etc), DirectSound , DirectMusic, DirectShow, DirectX Media, etc.

Las cuatro primeras versiones de DirectX aparecieron en un par de años. La primera versión, ***DirectX 1*** (1995), era poco más que un *kit* para desarrolladores de juegos y que permitía incorporar a Windows funciones multimedia. La siguiente***, DirectX 2 (***1996), era ya un componente del propio Sistema Operativo Windows (Windows 95 y Windsows NT). Fue una versión a la que se le dio una gran publicidad, para captar a los usuarios de juegos de vídeo, entre otras cosas. A finales del mismo año 1996, aparecieron diferentes versiones de ***DirectX 3***, las versiones 3.0a y 3.0b. La siguiente versión, ***DirectX 4***, no llegó a aparecer pues Microsoft puso a trabajar a dife-

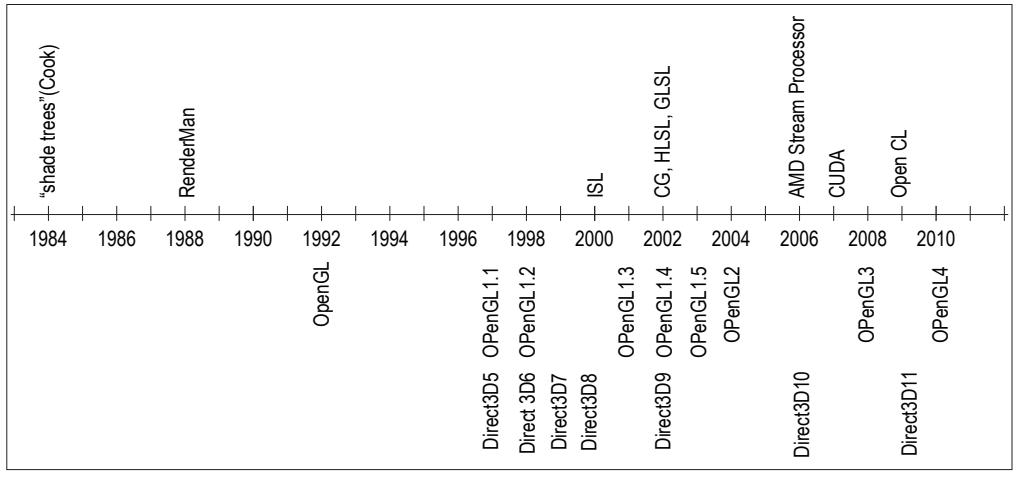


Figura 3.19 Evolución histórica de las diferentes versions de Direct3D y Open GL.

rentes equipos en las versiones 4 y 5 simultáneamente con la idea de sacar la versión 4 como una versión a corto plazo mientras se completaba la siguiente. Los desarrolladores de juego criticaron las características de las versiones de prueba de la versión 4 con lo que se saltó directamente a la versión 5.

La primera versión de Direct3D que apareció con estas últimas versiones fue acogida con un gran número de críticas, hasta el punto de que algunos de los críticos más respetados recomendaron a Microsoft que abandonara el proyecto y adoptara OpenGL. Las críticas iban dirigidas principalmente a que las operaciones más simples, tales como cambios de estado, requerían la creación y gestión de objetos denominados *execute buffers*, a diferencia de OpenGL que llevaba a cabo estas operaciones mediante simples llamadas a funciones.

La versión ***DirectX 5*** apareció en 1997. Al año siguiente, 1998, apareció la siguiente versión, ***DirectX 6***, al año siguiente, 1999, ***DirectX 7***, y en 2001, la ***DirectX 8*** que duraría algo más. Todas estas versiones iban acompañadas de variantes que salían a los pocos meses.

Microsoft no aceptó las críticas a la primera versión de Direct3D ni las recomendaciones de que abandonara el proyecto sino que sacó una segunda versión (aunque la denominó "5" para mantener la sincronía con DirectX) notoriamente mejorada, la ***Direct3D 5.0*** que es considerada como la primera versión efectiva, pues a partir de esa fecha fue aceptada y las posiciones con respecto a OpenGL comenzaron a equilibrarse. Esta versión remplazó los *execute buffers* con una API específica denominada DrawPrimitive. Sin embargo, las críticas continuaron pues se le consideró como una copia de las funciones equivalentes de OpenGL pero sin la estructura de base adecuada. Con la versión ***Direct3D 6.0*** se introdujeron mejoras en la estructura y funciones adicionales para gestionar multitexturas y *stencil buffers*, así como funciones renovadas para gestionar texturas. También se incorporaron métodos de apoyo para la compresión de texturas y para la aplicación de mapas de relieve básicos (*bump mapping*).

Con ***Direct3D 7.0*** (1999) se introdujo el formato de texturas DDS así como apoyo adicional para aceleración de transformaciones y cálculo de iluminación integrados en hardware (lo que era realizable a partir de la aparición de la NVIDIA Geforce) así como la capacidad para asignar *buffers* de vértices a la memoria física de la tarjeta gráfica (*vertex buffers*) uno de los primeros avances de Direct3D con respecto a OpenGL. También se mejoraron las funciones de soporte a texturas múltiples. En general, fue una versión que supuso un avance fundamental para Direct3D aunque también complicó considerablemente la programación, lo que llevaría a otra renovación en el lenguaje.

Con ***Direct3D 8.0*** (2001) se introdujeron nuevos conceptos de programación fundamentales, los *vertex shaders* y *pixel shaders*, lo que facilitaría el trabajo de los programadores al permitirles desentenderse del estado del hardware, concentrándose en la tarea a desarrollar en una determinada fase del flujo de estados de la *rendering pipeline*. El *driver* se encargaba de traducir las instrucciones correspondientes a instrucciones de máquina. Esto supuso un avance fundamental con respecto a OpenGL, que seguía anclada en una serie de transiciones de estados basados en su *finite state machine* pero que estaba comenzando a quedar obsoleto ante la generalización de los conceptos derivados de la programación basada en objetos. Algunos problemas de facilidad de uso de esta versión se mejoraron con la siguiente sub-versión, la 8.1. Con esta versión también se abandonó la dependencia de funciones heredadas de DirectDraw, que pasó a convertirse en otra API de la familia de DirectX y se integró en las nuevas funciones de Direct3D.

DirectX/Direct3D 9 (2002-2006) aparecieron en diciembre de 2002. Con esta nueva versión de Direct3D los críticos aceptaron con toda claridad que era al menos tan bueno como OpenGL, tanto por lo que respectaba a la funcionalidad como a la facilidad de uso.

Suponía el afianzamiento de una renovación radical de muchos aspectos estructurales e introducía nuevas funciones para texturas y *renders* así como optimizaciones específicas para Windows Vista. También incluía una versión definitiva del HLSL (*high level shader language*). En estas últimas versiones se mantenían características que ya estaban incorporadas en la primera versión, notoriamente su relación con el COM de Microsoft que facilitaba la relación con cualquier programa que utilizara este componente, tal como C++, C#, Visual Basic.Net o Delphi. Ya desde finales de la década de 1990 DirectX fue popularizándose como una alternativa entre los usuarios de Windows. A diferencia de OpenGL, Microsoft buscó desde el principio una relación más directa del lenguaje con el hardware, lo que inicialmente retardó su uso directo pues los fabricantes de hardware no daban facilidades para este acceso directo, pero a la larga le proporcionó mayores ventajas a medida que la posición dominante de Windows le permitió trabajar directamente con los fabricantes de tarjetas gráficas.

DirectX/Direct3D 10 (2006-2009) aparecieron en noviembre de 2006 para el nuevo sistema operativo Windows Vista y posteriores. Incorporaba nuevas estructuras, como el *shader model* 4.0, el conjunto de funciones agrupados en lo que se denominaba *geometry shader*. El SDK de esta nueva versión se ofreció a partir de febrero de 2007.

La versión ***Direct3D 11.0*** (2008-) se distribuyó junto con Windows 7, y se presentó en julio de 2008 como un superconjunto de Direct3D 10.1. Sus principales novedades eran la inclusión de *tesselation*, la capacidad para incrementar en tiempo real el número de divisiones poligonales de un modelo, el *rendering* con *multithread*, la capacidad para representar escenas procesando diversas líneas en paralelo, los *compute shaders*, la capacidad de utilizar la *rendering pipeline* para procesar tareas no gráficas tales como aceleración física y procesamiento de flujos (*stream processing*), de modo más o menos similar a los procesadores de flujos en paralelo CUDA de NVIDIA y FireStream de ATI/AMD, buscando métodos unificados de programación. También se incluían nuevos algoritmos de compresión de texturas.

Procesos básicos en OpenGL y Direct3D

Supongamos que queremos crear un programa muy simple, en una API, que dibuje una pequeña imagen en la pantalla. El código requerido implica una serie de palabras clave, tanto en DirectX como en OpenGL, junto con una colección de parámetros que puede resultar un poco apabullante. Pero si lo reducimos a lo esencial, escribiéndo las llamadas principales en pseudócodigo, el resultado sería algo así:

```
Main
{
    CrearVentanaAplicacion          // 1
    CrearObjetosComplementarios     // 2
    MantenerBucleMensajes           // 3
    RepresentarEscena               // 4
    CerrarSalir                     // 5
}
```

Veamos qué es lo que se hace en cada uno de los pasos anteriores de un modo algo más detallado. La estructura de este pseudocódigo es la características del lenguaje C y sus derivados: una función principal (*main*) que llama a otras funciones incluidas en un bloque delimitado por corchetes angulares. Lo que hacen este conjunto de funciones es, en otras palabras, arrancar y organizar el sistema, poner en marcha una serie de procesos básicos y salir. En el apartado que sigue se describe con un poco más de detalle esta estructura superior. En los siguientes, algunos detalles adicionales sobre los procesos básicos. Para simplificar la exposición me referiré principalmente a Direct3D.

Inicialización, procesos básicos, cierre

1. Crear una ventana para la aplicación
Los primeros pasos son más complicados y

menos intuitivos y son también los que varían más entre las dos grandes API.

En Direct3D, el código básico, simplificado, tendría este aspecto:

```
WNDCLASS wnd ={…};
RegisterClass (&wnd);
HWND hWnd = CreateWin (texto, tamaño, varios…);
```

En la primera línea se crea un objeto ("wnd"), una "ventana", a partir de una clase predefinida por medio de la palabra clave de Direct3D "WNDCLASS". En la segunda línea se registra este objeto. En la tercera línea se le da un tamaño, un texto y otros atributos y se le asigna un *handle* por medio de otra palabra clave, "HWND". El *handle* (literalmente "mango" o "asa") nos permitirá referirnos al objeto creado por medio de un puntero (el signo "&" indica que guardamos una dirección en memoria para apuntar al objeto, lo que ocupa mucho menos espacio que el propio objeto).

En OpenGL, el código básico tendría más o menos este aspecto:

```
glutInit (&argc, argv);
glutInitDisplayMode (…);
glutInitWindowSize (500, 500);
glutCreateWindow ("My First Program");
```

En la primera línea se inicializa la biblioteca de utilidades que permite adaptar la API a las diferentes plataformas. Esto implica crear un *handle* a la ventana y al contexto en que funcionará el programa. La segunda línea especifica el modo de representación. Los argumentos afectan al tipo de *buffer* (único, doble) y al color. El primer argumento dentro del paréntesis podía ser GLUT_SINGLE (otra alternativa sería GLUT_DOUBLE, que indica otro tipo de *buffer*). Y el segundo argumento podría ser GLUT_RGB para indicar que se trabajará con color real. Otra alternativa sería GLUT_INDEX para indicar que se trabajará con color indexado. La tercera línea especifica el tamaño de la ventana. La cuarta línea crea la ventana con un determinado título.

En OpenGL, habría por añadidura una línea, al comienzo, que activaría la inicialización específica del módulo GL y que sería algo así:

```
void myinit(void)
{
glClearColor(1.0, 1.0, 1.0, 1.0);
glColor3f(1.0, 0.0, 0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 500.0, 0.0, 500.0);
glMatrixMode(GL_MODELVIEW);
}
```

Con este código se inicializa propiamente OpenGL y se establece un determinado *estado*: un color de fondo, un color para los objetos, se inicializan las matrices básicas que se utilizarán para la transformación de los objetos del modelo y de proyección…

2. Crear objetos complementarios

Aquí también aparecen diferencias importantes entre las dos API. En Direct3D hay que crear específicamente un "objeto Direct3D" y a partir de ahí, un "dispositivo Direct3D". El código básico sería una primera línea como las que siguen:

```
DIRECT3D  d3d;
```

La primera línea crea un objeto "d3d" (el nombre es arbitrario) de la clase DIRECT3D (el nombre no es arbitrario).

Una aplicación Direct3D necesita crear un objeto de tipo Direct3D que es una interfaz de tipo COM. Todas las aplicaciones necesitan dar este paso: se inicializa un objeto que, a su vez, crea otros objetos que engloban todas las funciones que vayamos a utilizar y, al final, se destruye, liberando memoria y recursos.

Un objeto Direct3D permite crear otros objetos que cumplen diferentes cometidos. El primero y principal es "un dispositivo" de tipo *Direct3Ddevice* que representa la tarjeta gráfica que vayamos a utilizar. No sabemos qué tarjeta gráfica utilizará nuestro programa por lo que necesitamos una interfaz específica, independiente, para relacionarse con la tarjeta que sea.

```
a = D3Dadapter_Default;
b = DeviceTypeHAL;
c = Windowed;
d = BackBuffer;
e = etc
DIRECT3DDEVICE d3Ddevice (a, b, c, d, e,…);
```

Las cinco primeras líneas representan los parámetros del dispositivo que se crea en la última línea y que se pasan como argumentos al dispositivo "d3Ddevice" (el nombre es arbitrario) de la clase DIRECT3DDEVICE (el nombre no es arbitrario).

La primera línea establece como tarjeta gráfica la dada por una variable que apunta a la tarjeta concreta que utilice el ordenador en que se vaya a utilizar la aplicación. La segunda especifica el tipo de dispositivo. En Direct3D hay dos tipos principales de dispositivos: por hardware y por software. Al configurar las preferencias de un programa corriente de simulación 3D se encontrarán estas dos alternativas. Por ejemplo, en 3ds Max se puede escoger uno u otro si se va a menú *Customize/ Preferences/ Viewports/ Display Drivers*. Un dispositivo de hardware representa una tarjeta gráfica real. Este tipo de dispositivo se denomina corrientemente HAL (*hardware abstraction layer*). Es el dispositivo principal. Un dispositivo de software simula un dispositivo por hardware. Es más lento y suele utilizarse por los desarrolladores para probar funciones sin comprometerse con una determinada tarjeta. O puede utilizarse por usuarios si por alguna razón tienen problemas con el principal. La tercera línea especifica si queremos que la aplicación corra en modo ventana (la alternativa sería a toda pantalla) y la cuarta línea especificaría el tamaño del *buffer* que comentaremos más adelante. Y habría otros parámetros más que no necesitamos comentar pero que describirían otras características que debe cumplir nuestro dispositivo virtual.

En OpenGL, la inicialización debe completarse de varios modos. Puede ser necesario, por ejemplo, registrar las funciones de devolución de llamada (*callback functions*). Los programas interactivos reaccionan a eventos (movimientos o clics del ratón, entradas de teclado, etc.). En esta fase se configuran las funciones que se vayan a utilizar para manejar estos eventos, la funciones "de devolución de llamada" (*callback*). Las funciones disponibles son, entre otras, las siguientes:

```
glutDisplayFunc()                [1]
glutReshapeFunc()                [2]
glutKeyboardFunc()               [3]
glutMouseFunc()                  [4]
glutPassiveMouseFunc()           [5]
glutIdleFunc()                   [6]
…
```

[1: El contenido de la ventana necesita regenerarse; 2: La ventana se ha cambiado de tamaño o se ha movido. 3: Teclado accionado; 4: Ratón accionado; 5: Ratón no accionado pero movido; 6: Llamada aunque no haya ningún evento]

Las funciones seleccionadas deben registrarse previamente mediante procedimientos específicos (por ejemplo glutReshapeFunc (resize) para cambiar el tamaño de la ventana; glutKeyboardFunc (key) para responder al teclado). Una vez registradas pueden utilizarse mediante procedimientos para representar la escena (que invoca a una función principal denominada *display*), para cambiar el tamaño de la ventana (que invoca de nuevo a la función *display* con otros parámetros), para recoger datos de entrada del usuario, etc.

3. Configurar el bucle de mensajes

Si se arrancara la aplicación, y se enviaran datos como los que se describen más adelante al dispositivo virtual que se ha creado en el paso anterior, que a su vez los enviaría a la ventana virtual que hemos creado en el primer paso, estos datos se mostrarían un instante y luego se borrarían. Para que los resultados se mantengan en pantalla hay que renovarlos constantemente. Y, por añadidura, hay que procesar los mensajes derivados de eventos que proceden de la interacción con el usuario por medio de las funciones que se han registrado en el paso anterior (entradas de teclado, movimientos de ratón, etc.).

Para ello, la aplicación debe entrar en un bucle, denominado "bucle de mensajes" (*message loop*) que recibe mensajes derivados de eventos diversos (una entrada de

teclado, un clic de ratón, un objeto que aparece al invocar una orden, etc.) hasta que recibe un mensaje concreto de salir de la aplicación.

El pseudocódigo correspondiente sería algo así:

```
While(GetMessage(mensajeTalycual)
{
TraducirMensaje (mensajeTalycual)
EnviarMensaje (mensajeTalycual)
}
```

Mientras "mensajeTalyCual" no sea "Salir" la aplicación se regenerará constantemente y será visible en pantalla.

En la práctica esto es más complejo, sobre todo con juegos que necesitan sacar el máximo partido de los recursos del sistema e incluyen cláusulas condicionales para que, si no hay mensajes que procesar, se lleven a cabo otras tareas. Pero lo esencial es que hay un paso de este tipo que debe ser tenido en cuenta.

4. Representar la escena

El contenido de la escena no solo debe mantenerse en pantalla sino que debe renovarse constantemente. Si, por ejemplo, hacemos un zoom, el contenido previo debe borrarse para mostrar el nuevo contenido. Y si desplazamos la ventana por la pantalla ocurrirá otro tanto. Y si rotamos un objeto, otro tanto.

Para llevar a cabo estas tareas Direct3D utiliza *surfaces*. Una *surface* almacena una imagen, del tipo que sea. Y se guardarán, en el *buffer* correspondiente (una zona de memoria en espera de ser procesada), tantas imágenes como la memoria lo permita. Hay al menos dos *buffers*, un *front buffer* y un *back buffer*. El *front buffer* guarda la imagen que aparecerá en pantalla, la imagen final. El *back buffer* está oculto pero preparado para saltar al *front buffer*. Así, en cada marco (*frame*), las dos imágenes, la guardada en el *front* y *back buffer*, se alternan con tal rapidez que se evita el leve parpadeo que si no fuera así podría ser perceptible. Hay otros *buffers* que veremos más adelante.

El pseudocódigo básico (en Direct3D) podría ser algo así:

```
clear d3Ddevice (clearBackBuffer, clearZBuffer…)
d3Ddevice->IniciarEscena
{
…[funciones para dibujar elementos, etc]
d3Ddevice->FinalizarEscena
}
```

5. Cerrar y salir

Al finalizar el proceso hay que destruir todos los objetos que se hayan creado para liberar recursos del ordenador.

El pseudódigo sería algo así:

```
d3d->Release();
d3Ddevice->Release();
```

La primera línea libera el objeto Direct3D que hemos creado en el paso 2 y la segunda línea libera el dispositivo virtual que hemos creado en el paso 3.

En OpenGL no es necesario, como en Direct3D, gestionar la ventana de la aplicación. Pero al terminar el proceso también hay que liberar los objetos que se hayan creado y salir explícitamente, mediante funciones concretas.

Creación y representación de primitivas

En este caso, al igual que en los siguientes, no hay diferencias importantes entre Direct3D y OpenGL, por lo que daré una descripción conjunta en pseudocódigo indicando en algún caso estas diferencias.

La primitiva gráfica más sencilla que podemos dibujar es un triángulo. El pseudocódigo, que se desarrollará algo más en lo que sigue, sería algo así:

```
{
   CrearEstructuraVértices          // 1
   CrearBufferVértices              // 2
   VertexShader                     // 3
   EnsamblarPrimitivas              // 4
   ClipCullRasterize                // 5
   PIxelShader                      // 6
   DibujarPrimitiva                 // 7
}
```

1. Crear la estructura de vértices

El primer paso es leer los valores de la aplicación e integrarlos en una estructura adecuada. Los valores principales que se van a leer son vértices, entendidos como datos que incorporan tanto valores de posición geométrica como atributos diversos (color, normales, coordenadas de textura, etc.).

Esta estructura puede ser un tipo de dato corriente en cualquier lenguaje de programación, una estructura (*struct*), un tipo de dato compuesto que el usuario puede definir para almacenar con comodidad valores de diferentes tipos. En el ejemplo simple que estamos siguiendo, como necesitamos guardar información sobre, al menos, dos tipos de datos relativos a los vértices, su posición (que implica también tres valores) y su color, lo primero que se necesita es crear esta estructura y rellenarla con los valores del triángulo que vamos a dibujar. El código tendría el siguiente aspecto:

```
struct MisVertices
{
    tipoApi xyz;
    tipoApi color;
};
MisVertices vertices [] =
{
    {0,0,0, RGB (255,0,0)}
    {0,2,0, RGB (0,255,0)}
    {0,1,0, RGB (0.,0,255)}
}
```

El primer bloque define la estructura que consta de 4 datos: las tres coordenadas (que serían de tipo *float* para tener suficiente precisión aunque no lo hemos indicado) y el color que sería un tipo de dato preestablecido (también con tres valores internos). Por “tipoApi” indicamos el tipo utilizado en la API correspondiente, que sería en ambos casos un vector de tres o cuatro elementos. El segundo bloque rellena esta estructura con valores que corresponden a un triángulo determinado (en la práctica, estos valores serían variables que el usuario especificaría) y a tres colores, rojo, verde y azul, para cada uno de los tres vértices.

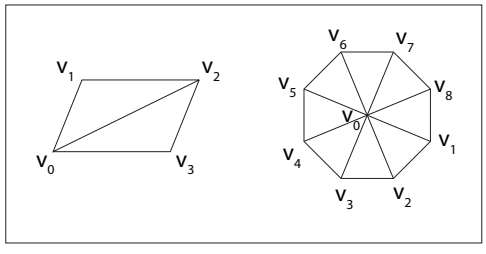


Figura 3.20 Indices a polígonos.

Hay varias cosas que se han eliminado en este pseudocódigo. Por ejemplo, Direct3D requiere que, además de lo anterior, se rellene un descriptor denominado FVF (*flexible vertex format*). Este descriptor admite una serie de valores para los vértices, los dos de la estructura y algunos más (por ejemplo el valor de la normal, el color especular o las coordenadas de textura que utilizaremos más adelante). Esta especificación es necesaria pues Direct3D no puede procesar cualquier tipo de datos. En cualquier caso, lo que nos importa de este paso es que contamos con una serie de datos estructurados que se pasan al siguiente estadio.

2. Crear el buffer de vértices y el buffer de Indices

El paso siguiente es meter los vértices que hemos creado en dos *buffers* específicos, un *vertex buffer* y un *index buffer*, prefijados por la aplicación. Estos *buffers*, cuando se trabaja con decenas de miles de datos, son fundamentales para acelerar el proceso. Deben ser entendidos, por tanto, como un recurso de optimización aunque en el contexto de este ejemplo simple aparezcan como una complicación innecesaria..

Para que la GPU pueda acceder al listado de vértices, estos necesitan ser colocados en una zona específica de memoria, en un *buffer* especial denominado *vertex buffer* (hay varios *buffers* de este tipo en las tarjetas modernas que quedan disponibles para alimentar la *pipeline*). Los *vertex buffers* se crean con métodos también específicos y que representan las operaciones que llevaría efectivamente a cabo la tarjeta gráfica real que se usaría en la aplicación. Este método

requiere una serie de argumentos de los que nos basta con mencionar dos: la estructura que contiene los vértices que hemos definido previamente y un puntero al *vertex buffer* (pVB). Por añadidura, una vez creado, se requiere bloquear un área de memoria para guardar estos valores y, una vez hecho esto, desbloquearlo. Esto es lo que se resume en el pseudocódigo siguiente:

```
CrearVertexBuffer(sizeof(MisVertices)*3, ...pVB)
pVB->Lock(...)
pVB->UnLock(...)
```

Un *vertex buffer* almacena los vértices pero no dice nada sobre cómo se relacionan estos vértices entre sí para formar primitivas geométricas. Esto requerirá especificaciones adicionales que definan la topología de agrupación de los vértices. Para ello se utilizan estructuras predefinidas que almacenan diversas topologías que quedan disponibles para estadios posteriores: listas de puntos, listas de líneas dadas por dos puntos, listas de puntos concatenados, listas de triángulos de diversos tipos, etc. Las denominaciones correspondientes en Direct3D son *PointList*, *LineList*, *LineStrip*, *TriangleList*, *TriangleStrip*, etc. Y las correspondientes en OpenGL son *gl_point*, *gl_lines*, *gl_line_strip*, *gl_line_loop*, *gl_triangle*, *gl_triangle_fan*, etc. Hay variantes en las que no vamos a entrar sobre el modo en que se crean las primitivas.

Pero si agruparamos vértices que posteriormente se definirían, por ejemplo, como un polígono de varios lados formado por triángulos, nos encontraremos con vértices duplicados: un cuadrilátero tendrá 6 vértices (2 x 3) y un octógono 24 vértices (8 x 3). Esto duplica innecesariamente el número de vértices. Para evitarlo se recurre a índices de modo que se eviten duplicaciones y estos índices se almacen en un *index buffer*. El procedimiento de creación es similar al de un *vertex buffer*. También como en el caso anterior este *buffer* requiere invocar una función específica para crear el *buffer* e insertarlo en la *pipeline*. El código siguiente muestra un ejemplo de cómo se aplica este método. El orden en que se especifican los vértices (denominado *winding order*) es importante aunque no vamos a entrar en los detalles de cómo hacerlo y porqué.

En los listados que siguen el vértice v0 está repetido 2 veces en el caso del cuadrilátero (los triángulos correspondientes se indican entre corchetes, como “T0”, T1”, etc) y el vértice v0 está repetido 8 veces en el caso del octógono.

```
Vertex quad[6] = { v0, v1, v2 [T0] v0, v2, v3, [T1]};
Vertex octagon[24] = {v0, v1, v2 [T0] v0, v2, v3 [T1] v0, v3,
v4 [T2]  v0, v4, v5 [T3] v0, v5, v6 [T4] v0, v6, v7 [T5] v0, v7,
v8 [T6] v0, v8, v1 [T7] };
```

Para evitar esta repetición se utilizan índices, de tal modo que hay una única lista de vértices y la lista de índices se remite a esta lista de vértices. La duplicación se lleva en realidad a la lista de índices, pero esto es mucho mejor pues esta lista solo contiene números enteros que ocupan mucha menos memoria. Y el uso de *vertex cache* evita también el procesamiento doble. El código quedaría como sigue para el triángulo (la lista de índices viene a decir: “usa los vértices 0,1,2 para construir el triángulo 0 (T0) y los vértices 0,2,3 para construir el triángulo 1 (T1)”.

```
Vertex v[4] = {v0, v1, v2, v3};
indexList[6] = {0, 1, 2 [T0] 0, 2, 3 [T1]};
```

Y como sigue para el octógono:

```
Vertex v[9] = {v0, v1, v2, v3, v4, v5, v6, v7, v8};
indexList[24] = { 0, 1, 2 [T0] 0, 2, 3 [T1] 0, 3, 4 [T2] 0, 4, 5
[T3] 0, 5, 6 [T4] 0, 6, 7 [T5] 0, 7, 8 [T6] 0, 8, 1 [T7] };
```

El almacenamiento de los vértices (es decir, de un índice que apunta a una serie de propiedades geométricas y no geométricas) en *buffers* independientes es una de las claves de la aceleración por hardware de los programas de simulación. En relación con esto hay varios detalles técnicos sobre el uso más adecuado de *buffers* dedicados en los que no podemos entrar y que afectan también el manejo de objetos geométricos más complejos como curvas y superficies.

3. Procesamiento de vértices (vertex shader)
El modo más sintético de explicar lo que hace un *vertex shader* es decir que recibe una serie de vértices como entrada y devuelve otra serie de vértices como salida. Entre estos dos extremos se llevan a cabo una serie de operaciones que modifican los valores originales de los vértices. Estas operaciones son, por lo general, operaciones afines, operaciones proyectivas que preservan relaciones pero modifican valores concretos. Este estadio también se denomina "procesamiento de vértices" (*vertex processing*) e incluye diversos tipos de operaciones por vértices (*per-vertex operations*).

Es importante insistir en que el término vértice puede resultar equívoco y sería más adecuado substituirlo por "datos básicos de entrada". Pues, como ya se ha dicho, los vértices almacenan información geométrica pero también atributos diversos. Y, en este estadio, no solo las posiciones de los vértices se transforman por las matrices de transformación del modelo y las matrices de proyección,

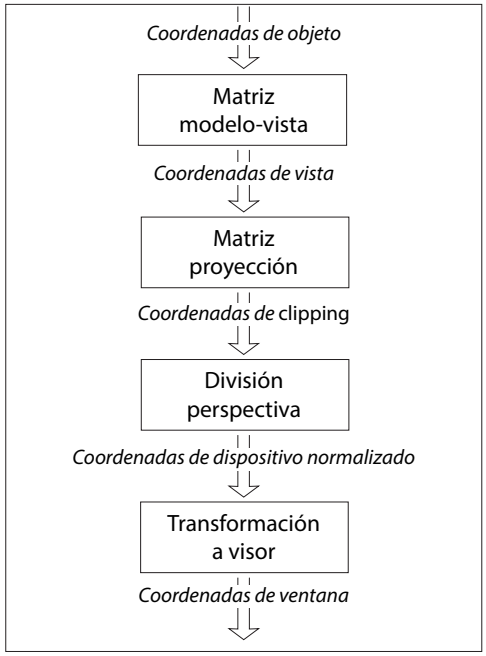


Figura 3.21 Transformaciones de vértices.

sino que otro tanto ocurre con las coordenadas de texturas y cualquier otro valor ligado a las coordenadas de los vértices.

Una de las transformaciones que tienen lugar en este estadio es la de coordenadas de espacio local a coordenadas de espacio modelo. En muchas aplicaciones, los objetos se crean en coordenadas locales lo que simplifica considerablemente el procedimiento de creación. Una vez creados en su propio espacio (*local space*) se colocan en posiciones variables en el espacio global (*world space*). Esto requiere el uso de las tres transformaciones clásicas: traslación, rotación y cambio de escala. Estas transformaciones son complejas y requieren el uso de matrices de transformación. Pero afortunadamente, Direct3D se encarga de llamar a funciones predefinidas que se encargan de automatizar el proceso por medio de una matriz de transformación del modelo (*modeling transformation matrix*).

La parte principal del procesamiento de vértices es la transformación de coordenadas, que venía dada en el espacio modelo, a las coordenadas de proyección que dependen del punto de vista escogido. Para esto se requiere ejecutar tres transformaciones canónicas que ya hemos visto en la sección sobre la *rendering pipeline*. Estas transformaciones también requieren el uso de matrices de transformación. Pero es un proceso muy estandarizado que funciona prácticamente igual en las dos API, si bien el orden y otros detalles pueden cambiar. Estas tres transformaciones canónicas se automatizan por medio de tres matrices: *a*) de vista (*viewing transformation*), que sitúa y orienta la cámara virtual en la escena; *b*) de proyección (*projection transformation*), que define el campo visual; *c*) de visor (*viewport transformation*), que ajusta el resultado a las dimensiones de la ventana de salida. En cada caso hay que inicializar las matrices como matriz de identidad y asignarle después los valores correspondientes. Las matrices son de 4 x 4 (coordenadas homogéneas). Hay diferencias entre D3D y Open GL pues en este último caso las matrices de transformación son parte del estado (*OpenGL*

state) y deben ser definidas antes de los vértices a los que se van a aplicar, pero no vamos a entrar en estos detalles.

La primera matriz que hay que invocar es, por tanto, la matriz de transformación ("matrizTransf" en el ejemplo que se dará más adelante) que sitúa los vértices en la posición del espacio que se requiera según la escena que estemos creando.

La segunda matriz sitúa la cámara, esto es, el punto a partir del cual se llevará a cabo la proyección de la escena. Esto requiere declarar previamente la matriz y, a continuación, definir tres vectores: el primero, que define la posición de la cámara, el segundo, que define el punto al que apunta y el tercero, que define su orientación. Estos cuatro valores se pasan como argumento a un método predefinido (por ejemplo "claseMatrixLookAt") que implementa la transformación. El código podría tener un aspecto como lo que sigue:

```
claseMatrix matrizVista;
claseVector3 Ojo (0,-10,0);
claseVector3 LookAt (0,0,0);
claseVector3 Up(0,1,0);
claseMatrixLookAt(&matrizVista, &Ojo, &LookAt, &Up);
dispositivo->SetTransform(...);
```

La tercera matriz ("matrizProyección") del pseudocódigo que sigue, lleva a cabo esta proyección por la que todo el sistema de coordenadas cambia de tal modo que el nuevo origen de coordenadas pasa a ser el punto de vista dado por la segunda matriz. Afortunadamente, la API se encarga de los detalles de esta transformación y el código básico es muy simple. En la primera linea se declara la matriz, en la segunda se define el ángulo de visión ("fov": *field of view*) y en la tercera se define la relación de aspecto ("aspect") dividiendo la anchura por la altura de la ventana que hemos definido previamente:

```
fov(anguloVision);
aspect(anchuraVentana, alturaVentana);
frustrum(left, right, bottom, top, zNear, zFar )
claseMatrix matrizProyeccion(fov,aspect, zNear, zFar)
```

Las matrices de espacio modelo, vista y proyección se combinan en una única matriz, que a veces se denomina WVP (*World-View-Projection*). El resultado es lo que se denomina el "espacio de proyección" (*projection space*). Este resultado se normaliza para que sea independiente de cada caso particular. Esto es, las coordenadas de anchura y altura deben estar comprendidas en el intervalo [-1,1] y las coordenadas de profundidad en el rango [0,1]. Después de esta conversión las coordenadas se denominan NDC (*normalized device coordinates*, "coordinadas normalizadas del dispositivo"). Tanto este último paso como los anteriores se corresponden con la descripción general y las ilustraciones que se han dado en la sección anterior.

El *vertex shader* solo necesita invocar a esta matriz para procesar el vértice que sea. Es decir, repitiendo el ejemplo que hemos dado en la sección anterior, todo el proceso descrito que en versiones previas tenía que especificarse, ahora queda condensado.

```
struct vertex { tipoApi position; tipoApi color; tipoApi normal; tipoApi textura;}
vertex v;
tipoApi matrizTransf;

vs (v.position) {nuevaPos = position * matrizTransf; return nuevaPos;}
```

El *buffer* correspondiente almacenará la matriz combinada "matrizTransf", pues dado que esta matriz varía con cada objeto se necesita actualizarla en cada caso: almacenarla en un *buffer* facilita esta actualización. Las API modernas cuentan con *buffers* que se utilizan en función de la frecuencia de actualización prevista para cada caso. Pueden crearse *buffers* para, por ejemplo, variables de luces o efectos de niebla o cualquier otro conjunto de parámetros que requieran ser actualizados con frecuencia según las aplicaciones previstas.

4. Ensamblaje de primitivas/geometry shader

Después del procesamiento de vértices sus posiciones y atributos quedan fijados y se puede pasar a otro estadio denominado "ensamblaje de primitivas" (*primitive assembly*). En este estadio, los vértices se agrupan en

primitivas completas con arreglo a la lista de topologías admisibles que hemos visto en el primer estadio: puntos, líneas con dos o más vértices, triángulos, polígonos y, en general, mallas poligonales con un número arbitrario de vértices. Este agrupamiento depende de cómo se ha especificado la creación de los vértices: en los casos más simples el agrupamiento ya viene definido por el tipo dado en el momento de la creación, en otros casos puede ser necesario un reagrupamiento debido a que, por ejemplo, una malla poligonal se ha modificado debido a operaciones de recorte definidas por la pirámide de visión.

La principal razón de ser de este estadio es que determinadas operaciones, notoriamente *clipping*, modifican las primitivas al cortar el conjunto en determinados puntos, lo que requiere rehacer la estructura del conjunto de vértices.

De hecho, las funciones específicas que se realizan en este estadio pueden variar según las versiones y según las aplicaciones. En algunas descripciones de las operaciones que se llevan a cabo en el estadio de *primitive assemby* se incluyen las siguientes, además de la reagrupación de vértices en primitivas predeterminadas tras el recorte (*clipping*) de las primitivas con el *frustrum*: conversión a coordenadas homogéneas; transformación a coordenadas de visor; *back-face culling*. Son operaciones que, como veremos, se describen más corrientemente como propias de otros estadios. Hay que recordar que la serie de operaciones que se llevan a cabo no sigue una secuencia lineal y pueden incorporarse a diferentes fases de un procesamiento que cada vez se ejecuta con más intensidad en paralelo.

Por otro lado, en las *rendering pipelines* recientes de Direct3D y OpenGL ha aparecido un nuevo estadio denominado *geometry shader*. Este estadio, que se etiqueta como "opcional", se sitúa entre el ensamblaje de primitivas y el proceso final de rasterización. El *geometry shader* recibe como entrada el resultado del ensamblaje de primitivas. Así, siguiendo con el ejemplo anterior puede recibir una tira de 8 triángulos, y devuelve como salida otra primitiva, que, en este caso, sería la misma, por lo que se podría prescindir de este estadio. ¿Para qué sirve entonces? Para los casos en que se definan primitivas más complejas que solo pueden ser procesadas por este *shader*. Pues también puede recibir una primitiva y devolver otras varias reestructuradas según los requerimientos del usuario o del propio sistema. Esta reestructuración puede consistir en algo tan simple como la creación de nuevos vértices por interpolación. Pero también puede modificar los tipos de primitiva: una serie de puntos puede convertirse en una serie de triángulos o una serie de líneas pueden colapsarse en puntos. Algunas aplicaciones típicas de este nuevo *shader* son las siguientes: la creación de *billboards* o *sprites* a partir de puntos, la creación de líneas con grosor a partir de líneas simples, la teselación como fase previa para aplicar mapas de desplazamiento (si bien esta función ha sido ampliada con el nuevo *shader* de teselación), la creación de extrusiones por hardware, la multirresolución automatizada, la creación de pelo, hierba, etc. Todo esto repercute en la reducción de los *vertex buffers*, que no necesitan gestionar geometría adicional y, por tanto, en una aceleración general de estos procedimientos.

5. Clipping. Culling. Proyección 2D. Rasterización

Tras todas las transformaciones anteriores, los datos de la escena están preparados para ser enviados a la ventana de salida. Esto requiere toda una serie de operaciones finales que se agrupan en este apartado, para seguir las descripciones generales de la *rendering pipeline*, pero que pueden haberse producido antes.

La información de la escena que llega a este estadio es la de una serie de datos estructurados (vértices con atributos que componen primitivas) situados en el interior de un volumen limitado por 6 planos, el *frustrum*. A partir de ahí, se realizan a cabo las siguientes operaciones principales.

Clipping. Como decía, esta operación puede haberse efectuado, según las diferentes aplicaciones, durante la fase de ensamblaje de primitivas. En cualquier caso, lo fundamental es que cada primitiva se compara geométricamente con el *frustrum* de lo que resultan tres posibilidades: *a*) la primitiva queda completamente dentro del *frustrum* y se conserva, *b*) la primitiva queda completamente fuera del *frustrum* y se descarta, *c*) la primitiva queda parcialmente dentro del *frustrum* con lo que se computa la intersección y se crean nuevos vértices que hay que volver a ensamblar.

Culling o *backface culling*. Cada cara triangular o poligonal tiene una normal que se deriva de las coordenadas de los vértices que lo forman y de cómo están ordenados. La convención corriente es que si el orden es horario (*clockwise*) con respecto al observador, el lado es frontal y visible, y si es antihorario (*counterclockwise*), el lado es posterior e invisible. Esta convención deriva de que la mayoría de los objetos que hay en una escena son sólidos cerrados, por lo que los polígonos cuyas normales forman un ángulo mayor de 90º con el vector correspondiente al punto de vista se eliminan pues corresponderían a caras no visibles de un volumen cerrado o a caras que, por alguna razón (incluyendo errores del usuario) tienen esta orientación. Dicho de otro modo, el observador no puede ver los triángulos posteriores de un objeto porque quedan ocultos por los frontales. Por esta razón, todos los triángulos con esta orientación se eliminan de la *pipeline*. Esto reduce potencialmente el número de polígonos a procesar aproximadamente a la mitad. Véase las ilustraciones correspondientes a estos estadios en la sección anterior.

Transformación a coordenadas del visor (*viewport transform*). Después de la proyección y la transformación a coordenadas normalizadas del dispositivo (NDC) con que finalizaba el *vertex shader*, las coordenadas 2D que conformarán la imagen de salida se transforman en un rectángulo que se almacenan en un *back buffer* dedicado al visor de salida (*viewport*). Después de esta transformación las unidades son píxeles. Las coordenadas de profundidad *z* se conservan pues son utilizadas por el *depth-buffer*. Esta transformación viene dada generalmente por una función simple que admite cuatro comandos, los dos primeros dan la posición del visor (la esquina inferior izquierda) en la pantalla general en caso de que la salida no sea a pantalla completa y los dos siguientes, las dimensiones:

```
viewportTransform ( x, y, anchura, altura )
```

Conviene recordar que las coordenadas de pantalla de OpenGL son diferentes que las de Direct3D (Windows). En Windows el origen está en la parte superior izquierda de la pantalla y en OpenGL está en la parte inferior izquierda.

Interpolación de valores de vértices. Los vértices incorporan información sobre propiedades adicionales a la información geométrica. Durante el proceso de rasterización los vértices se convierten en píxeles, y se crean píxeles en todas las posiciones intermedias. Los valores de estos píxeles creados en posiciones intermedias son desconocidos. Sus valores se deducen por interpolación entre los vértices adyacentes. Esto incluye también los valores de profundidad (*depth values*, *z-values*). De este modo, cada nuevo píxel contará con la misma serie de atributos que los vértices originales. En el caso de los valores de profundidad, esto requiere cómputos adicionales para tener en cuenta la distorsión perspectiva, pero esto se lleva a cabo mediante funciones internas automatizadas en las que tampoco es necesario entrar.

La salida del proceso de rasterización es, por consiguiente, una serie de píxeles o fragmentos (el término de OpenGL) en una determinada posición en 2D y que incorporan valores derivados de los vértices originales. Si, por ejemplo, una línea va a cubrir una docena de píxeles en la pantalla, este proceso la convierte en 12 fragmentos. Cada fragmento almacena la información proveniente de los vectores (color, coordenadas de textura, posición en profundidad, etc.) además de sus co-

rrespondientes coordenadas de ventana (*window coordinates*). Dado que, para seguir con el ejemplo de la línea, estos valores solo se conocían para dos fragmentos, el correspondiente al inicio y al final, los otros diez valores de este ejemplo se determinan por interpolación, si el modo de representación es suavizado (*smooth shading*), o reciben de modo uniforme el valor del último vértice si el modo de representación es plano (*flat shading*).

Cada tipo de primitiva incorpora por añadidura reglas adicionales que se utilizan para enriquecer la representación. Los polígonos pueden incluir patrones, o contornos resaltados, las líneas pueden tener grosor y los puntos pueden tener grosor u otros atributos (estas cosas se controlan, por ejemplo, en OpenGL por funciones específicas, glPolygonMode, glLineWidth, glPointSize y glPointParameter) o estar asociados a texturas con canales alfa que las recortan para simular iconos o partículas (lo que se denomina corrientemente *sprites*). Por otra parte, todas estas primitivas fragmentadas se tratan por medio de funciones adicionales de *antialiasing*, otro tema que también dejaremos para más adelante.

6. Pixel/Fragment shader

Los valores interpolados con que finaliza el estadio anterior se pasan al estadio siguiente como datos de entrada. El *pixel/fragment shader* lee los datos de entrada y los convierte en datos de salida. Estos datos de salida son píxeles/ fragmentos que pueden tener hasta 8 atributos de color, más un valor de profundidad opcional. El *pixel shader* es semejante al *vertex shader* pues debe procesar cada uno de los elementos de entrada que recibe para devolver un valor similar. Al igual que el *vertex shader* procesa todos y cada uno de los vértices que le llegan, el *pixel shader* procesa todos y cada uno de los fragmentos que han surgido de la interpolación de los valores de todos y cada uno de los vértices. Su función es calcular el valor del fragmento a partir de la información de entrada y la información adicional que pueda modificar esta información. O suprimirla, como ocurriría si se descarta porque otro fragmento coincide con su posición pero según la información almacenada en el *depth buffer* tiene un valor de profundidad menor o porque el *stencil buffer* especifica un valor de substitución previamente almacenado.

Aunque estamos suponiendo un caso simple, sin texturas, podemos recordar que, como se verá en el apartado siguiente, una de las principales operaciones que se llevan a cabo en este estadio es la proyección de texturas. Las coordenadas de textura también se han transformado en el *vertex shader* y son uno de los datos que se mantienen en la proyección de los vértices sobre el espacio 2D de salida.

7. Cargar el buffer. Dibujar la primitiva. Liberar el buffer

Cuando todo está listo para ser representado en la pantalla de salida, pasando por alto toda una serie de operaciones adicionales que pueden tener que llevarse a cabo antes de este paso final y que se describirán sumariamente más adelante, el último paso es cargar la información en el *frame buffer*. Esto implica su inicialización mediante métodos específicos que tampoco vamos a detallar y que utilizan formatos específicos preestablecidos.

Los fragmentos que han sobrevivido de todos los anteriores se convertirán finalmente en píxeles virtuales que, cuando todo el programa se lleve a cabo en un dispositivo real, pasarán a ser píxeles reales. Al terminar el proceso hay que liberar el *buffer* y dejarlo disponible para otros procesos.

La *rendering pipeline* en Direct3D y OpenGL

Las figuras 3.22 y 3.33 resumen las operaciones que se han descrito con más detalle en los apartados anteriores pero situándolas en el orden en que se presentan en las dos API principales. Esta descripción está basada en las versiones más recientes (2012) de estas API. Las diferencias principales con respecto a versiones anteriores están en la inclusión de un nuevo estadio, *Tessellation*, de otros esta-

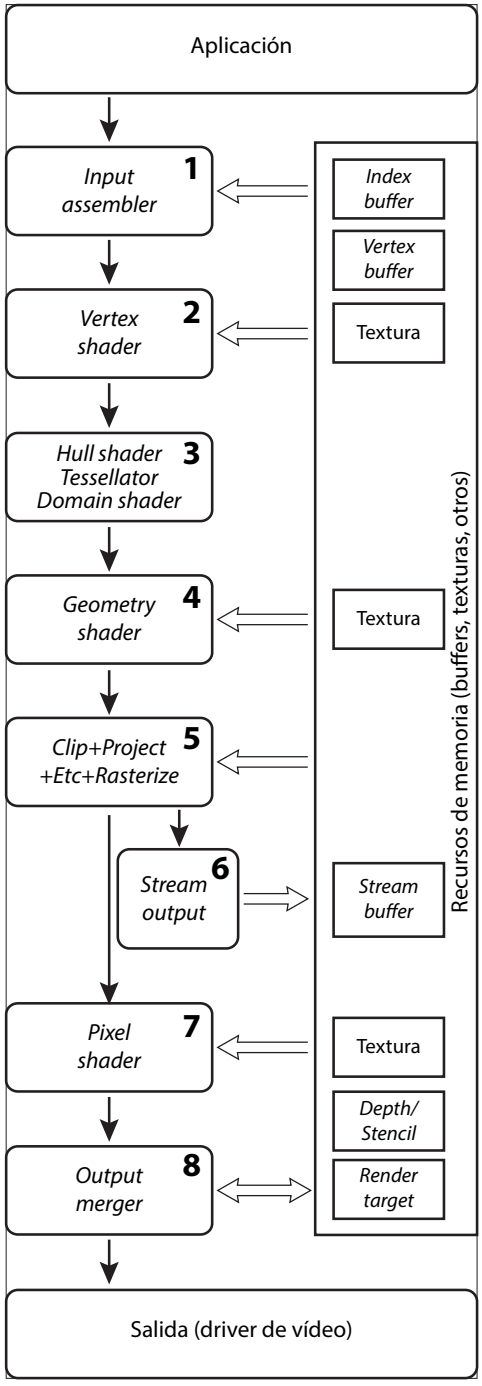


Figura 3.22 Rendering Pipeline *de Direct3D 11.*

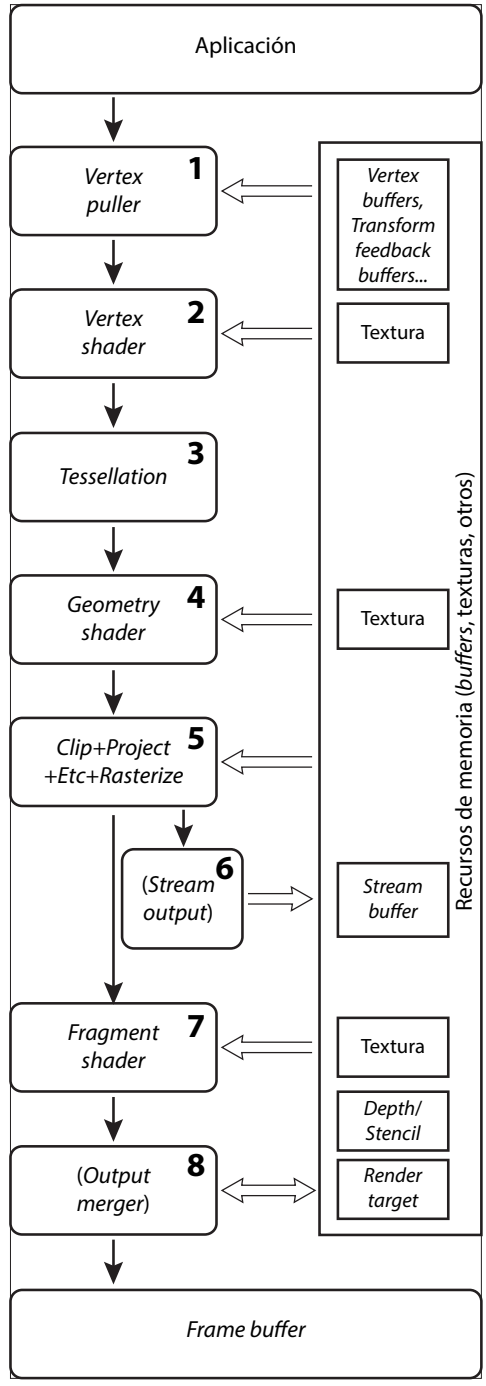


Figura 3.23 Rendering Pipeline *de OpenGL 4.*

do algo menos reciente pero también relativamente nuevo, *Geometry Shader* (que apareció en la versión 10 de Direct3D) y en algunas diferencias de detalle importantes en los últimos estadios. Esto cambiará en los próximos años (ya habrá cambiado cuando este libro se publique), pero todo lo que interesa aquí es hacerse una idea del modo particular en que los esquemas generales que se han presentado antes se aplican en casos particulares.

1. *Recopilación y reestructuración de datos* (*input assembler/vertex puller*). El primer estadio es recopilar la información enviada por la aplicación. Esto corre a cargo del primer estadio del procesamiento, el ensamblador de entradas (*input assembler*, en Direct3D) o el "tirador" de vértices (*vertex puller*, en OpenGL). En este estadio se leen datos geométricos, vértices e índices que se utilizan para combinarlos en primitivas geométricas tales como triángulos y líneas. Otra forma de describirlo es decir que es el estadio en el que se procesan los comandos enviados por la aplicación a la que se sirve. Hay, por tanto, todo un proceso complejo implicado por el que los comandos deben ser interpretados y traducidos a los tipos y funciones que pueden ser procesadas por la API, sea Direct3D u OpenGL. Entre estos procesos hay que incluir transformaciones de formato y operaciones de desempaquetado (*unpacking*) de datos básicos.

2. *Vertex shader*. El *vertex shader* se ejecuta una vez por cada vértice. Recibe un vértice como dato de entrada y devuelve un vértice como dato de salida. Su función es transformar cada vértice desde las coordenadas dadas en espacio objeto o espacio modelo a coordenadas dadas en espacio de recorte (*clip space*). Estadios anteriores diferenciados, como el ensamblaje de primitivas (*primitive assembly*), que reestructura las primitivas en función de los resultados de los recortes operados en el volumen de la escena que importa para el punto de vista considerado, el *frustrum*, se integran en este estadio.

3. *Teselación*. Las versiones recientes han añadido este estadio que implica *shaders* adicionales. En Direct3D 11 hay tres nuevos estadios que aquí hemos reunido en uno: *a*) *hull shader*, *b*) *tessellator*, *c*) *domain shader*. La función general de este estadio es subdividir de un modo eficiente la geometría en el momento en que se necesita restaurándola después a su estado original. Las superficies que llegan a este estadio general están compuestas por parches (*patches*), triangulares o cuadrangulares, definidos por una serie de puntos de control que han sido previamente transformados en el *vertex shader*. El *hull shader* es un estadio programable que tiene dos funciones principales. La primera, opcional, es convertir los parches de un tipo de representación (una determinada base matemática) a otra. La segunda es computar factores de teselación adecuadas que se pasan al siguiente estado, el *tessellator*. Esto permite que la teselación sea adaptativa y se acomode a los diferentes casos (por ejemplo, a la mayor o menor distancia del punto de vista). Los factores de teselación (*TessFactors*) se especifican para cada arista del parche y en la versión 11 pueden variar de 2 a 64, lo que implica las correspondientes subdivisiones de los triángulos o cuadrángulos. El *tessellator* es un estadio configurable pero no programable. Esto quiere decir que su función es fija, aunque admite diversas configuraciones previas. A partir de los datos recibidos por el *hull shader* subdivide cada parche en nuevos triángulos o cuadrángulos. No tiene acceso a los puntos de control sino que se limita a ejecutar la subdivisión especificada. Cada vértice resultante se pasa al estadio siguiente. El *domain shader* es un estadio programable que tiene una función principal: evaluar la representación resultante de los datos anteriores, opcionalmente, aplicar determinadas operaciones tales como desplazamientos y pasar los resultados al estadio siguiente, el *geometry shader*, o bien, si este estadio no se aplica, al siguiente, el procesamiento de proyecciones (*clipping*).

4. *Geometry shader*. Es un estadio opcional que toma como entrada los vértices de una o más primitivas y genera ninguno o nuevos vértices. Es decir, que puede reducir o aumentar y reorganizar las primitivas recibidas. También puede añadir atributos adicionales a una primitiva. Se utiliza para casos tales como la creación de *sprites* a partir de puntos, de líneas con grosor a partir de líneas simples, de la simulación de pelo o hierba a partir de líneas o la generación de mapas de desplazamiento multiplicando las primitivas a partir de una base simple.

5. *Procesamiento de proyecciones y rasterización* (*clipping*, *project*, *rasterize*). En este estadio se completan las operaciones de transformación que proyectan los vértices que han pasado los estadios anteriores sobre el visor de salida (*viewport*) tras someterlos a operaciones adicionales de recorte (*clipping*), eliminación de caras no visibles (*culling*), proyección perspectiva, normalización del *frustrum* y, en algunos casos, pruebas previas de profundidad (*depth testing*). El resultado es una serie de datos 2D, fragmentos candidatos a convertirse en píxeles de salida.

6. *Procesamientos adicionales* (*stream output. transform feedback*). Tanto Direct3D (*stream output*) como OpenGL (*transform feedback*) cuentan con recursos adicionales para escribir los resultados de los estadios anteriores en memoria, en *buffers* especiales. Esto permite que los resultados del procesamiento de vértices en diversos estadios vuelvan a los *shaders* que lo necesiten e incluso, si es preciso, a la CPU.

7. *Pixel shader. Fragment shader*. El *pixel/fragment shader* se ejecuta una vez por cada fragmento. Recibe un color como entrada y, en función de información adicional que puede venir de diversas vías, entre ellas el procesamiento de texturas, devuelve un color como salida. Con variantes según las API, pueden incluir operaciones de *antialiasing* o eliminación de píxeles en función de los datos almacenados en *buffers* especiales. Los valores procesados se envían a un *buffer* que almacena colores preparados para ser enviados a la salida final y que se denomina corrientemente *render target*.

8. Operaciones finales (*Tests. Output merger. Pixel transfers*). En Direct3D el estadio *output merger* toma los fragmentos recibidos del *pixel shader* y los somete a una serie de operaciones finales tales como pruebas de comprobación de la profundidad (*depth test*) o de supresión de zonas por plantillas (*stencil test*), además de la principal operación final que es la fusión de las imágenes destinadas a la salida con datos recibidos de otras vias (*render target blending*). Hay que tener en cuenta que el *pixel shader* puede emitir diferentes valores para diferentes *render targets*, que deben fusionarse en un único valor final. En OpenGL hay operaciones finales similares que se basan en lo que se engloba bajo la denominación de "empaquetado y desempaquetado de píxeles" (*pixel packing and unpacking*), y que se refiere al modo en que los datos de imagen se escriben en la memoria. Estos datos se guardan en *buffers* especiales para datos de píxeles (*pixel unpack buffer*, *pixel pack buffer*). Las operaciones (*pixel transfer operations*) se definen para cuatro clases de grupos de píxeles: *RGBA component* (cuatro componentes, *red*, *green*, *blue*, *alpha*), *Depth component* (un único componente de profundidad), *Color index* (un índice a una tabla de color), *Stencil index* (un índice a una plantilla). Las operaciones denominadas *pixels transfer* que se llevan a cabo sobre estos datos básicos pueden consistir en conversiones de formato o modificación de los rangos de estos grupos pero también en operaciones más complejas de tratamiento de imágenes (filtros basados en convoluciones, etc.).

La explotación del paralelismo, los nuevos lenguajes y las nuevas tecnologías

Una API es, como ya hemos visto, una aplicación que proporciona recursos para especifi-

car el modo en que diferentes componentes de una aplicación general interactúan entre sí y con los componentes de la máquina concreta sobre la que estas aplicaciones generales funcionarán. Además de esto, y en paralelo a la evolución de las tarjetas gráficas y las propias API, pueden proporcionar recursos adicionales de programación que permiten añadir nuevas funciones a estos recursos básicos, ampliando las posibilidades para sacar el máximo partido del hardware y hacer que los programas funcionen de modo más efectivo.

A medida que estas funciones adicionales se fueron desarrollando surgió la necesidad de utilizar un lenguaje compartido entre los programadores. Las primeras API, como ya hemos visto, se creaban en lenguaje ensamblador, adaptado a cada dispositivo específico y, después, por medio de lenguajes generales (C, C+++, Java, lenguajes propios de la API, etc.) ligados a clases y bibliotecas de funciones específicas.

Pero la escritura de *shaders* se facilita considerablemente si se utilizan lenguajes diseñados especificamente para ellos y que no tengan que preocuparse por especificaciones técnicas que pueden dejarse en manos de la API. Por esta razón surgieron los lenguajes de programación de *shaders*.

Por añadidura, como también veremos más adelante, han surgido tecnologías específicas (Stream Processor, CUDA, OpenCL) dirigidas específicamente a explotar el paralelismo y hacer más eficiente el funcionamiento de las tarjetas gráficas posibilitando que se ejecuten múltiples secuencias en paralelo.

Lenguajes de programación de *shaders*

El primer lenguaje de *shaders* interactivo se desarrolló durante la década de 1990 en un sistema gráfico de procesamiento en paralelo denominado *PixelFlow* en la universidad de North Carolina. Uno de los principales investigadores que trabajaron en este proyecto fue Marc Olano, que posteriormente pasó a trabajar con SGI y, junto con otros investigadores de esta empresa, contribuyó a crear OpenGL Shader que se considera el primer lenguaje comercial de alto nivel para la programación de gráficos en tiempo real, que apareció en 2000 y que se dio a conocer en una publicación de Dave Baldwin en octubre de 2001.

Posteriormente aparecieron el lenguaje CG de NVDIA, cuya primera especificación se publicó en junio de 2002. Y el lenguaje HLSL de Microsoft, cuya primera especificación se publicó en noviembre de 2002, como parte de la primera versión (beta) de DirectX 9.0.

Estos tres lenguajes, herederos de RenderMan, se han influenciado entre sí considerablemente. La descripción que sigue de cada uno de ellos es necesariamente sintética y se centra en las diferencias principales.

RenderMan, el padre de todos los lenguajes de *shaders*, apareció en 1988 después de varios años de desarrollo y de varias pruebas parciales en diferentes contextos. Con él apareció también la plena integración del concepto de *shader* en un lenguaje de programación relacionado directamente con una aplicación gráfica de simulación visual. En los años posteriores sería la base de películas tan conocidas como Parque Jurásico (Steven Spielberg, 1993), La guerra de las galaxias" (Georges Lucas, 1999, *Star Wars Episode1: The Phantom Menace*) , El señor de los anillos (Peter Jackson, 2001, *Lord of the Rings: The Fellowship of the Ring*), por no citar sino algunas de las más famosas. Y de otras películas desarrolladas integramente por ordenador como *Toy Story* (1994), *A Bug's Life* (1998), *Monsters Inc* (2001) o *Finding Nemo* (2003), entre muchas otras.

La principal diferencia entre RenderMan y los lenguajes que le seguirían es que estos actúan como un lenguaje independiente que deja que la GPU se relacione con el programa de aplicación por medio de una interfaz que consiste en un módulo independiente. Esto está basado en razonamientos industriales y comerciales (descargar a los programadores de la tarea de tomar en consideración las variantes de cada plataforma). Pero esto no era así con RenderMan, que

relacionaba directamente los programas de modelado y los módulos de representación. Estaba concebido como un *interface* de *rendering* completo.

Aparte de esta diferencia fundamental, proporcionaba toda una serie de recursos tales como la descripción de primitivas geométricas, organización jerárquica de los objetos, transformaciones automatizadas, atributos de materiales, luces o cámaras por medio de *shaders* especializados, que ahora ya están integrados en OpenGL o Direct3D, por lo que los lenguajes de más alto nivel que se han construido por estas y otras firmas no requieren tenerlos en cuenta.

La parte más interesante es la correspondiente al *RenderMan Shading Language*, que es un módulo integrado en RenderMan y que permite describir libremente cualquier tipo de *shader* que puede posteriormente ser pasado al módulo de representación a través de la interfaz de RenderMan.

Es un lenguaje basado en C, como los que siguen, y por tanto muy similar a estos. La diferencia principal tiene que ver con el nivel de abstracción. Los lenguajes que siguen utilizan dos tipos principales de abstracción del *shader*: *vertex shaders* y *pixels* o *fragment shaders*. RenderMan utiliza cinco diferentes niveles de abstracción: *light shaders*, *displacement shaders*, *surface shaders*, *volume shaders* e *image shaders*.

Hay también otras diferencias de menor importancia como un mayor número de tipos de datos o variables predefinidas en RenderMan. De hecho, algunos tipos especiales utilizados por OpenGL, como los calificadores *uniform* y *varying* fueron inventados por RenderMan y la mayoría de las funciones internas son también similares.

La diferencia principal, ya mencionada, es que RenderMan se relaciona peor con todo tipo de tarjetas gráficas, razón por la que, pese a seguir siendo una referencia notable por su alta calidad, no se ha utilizado en ordenadores más o menos corrientes. Pixar lo ha desarrollado para su propio software, RenderMan Pro Server.

En 2000 apareció ISL ***(***OpenGL Interactive Shading Language***)*** que fue el primer intento de crear un lenguaje con las características de RenderMan pero adaptado a OpenGL. En 2002 fue substituido por GLSL y en la actualidad ya no está disponible. Su principal inconveniente desde el punto de vista de su difusión comercial era que requería utilizar OpenGL API como lenguaje ensamblador. Por otro lado, podía funcionar en una variedad de GPU incluso en las no programables. Esto es otra diferencia fundamental con GLSL, que se dirige en exclusiva a GPU programables. Dicho de otro modo, GLSL apunta al futuro en el que se supone que todas las GPU serán programables mientras que ISL apunta al pasado, a GPU con menos capacidades. Al igual que OpenGL, está basado en C. Pero el código es bastante diferente debido al requisito del lenguaje ensamblador

Como ya he dicho más arriba, ***GLSL (OpenGL Shading Language***, a veces denominado GLSlang) se considera el primer lenguaje comercial de alto nivel para la programación de gráficos en tiempo real. Apareció en 2000 (desarrollado por 3DLabs junto con OpenGL ARB) y se dio a conocer en una publicación de Dave Baldwin en octubre de 2001. Aunque se introdujo como una extensión de la versión 1.4 de OpenGL, su desarrollo oficial se dio a partir de la versión 2.0 de OpenGL. Esto facilitó considerablemente la tarea de los programadores, no solo por ser un lenguaje más intuitivo y sencillo sino porque permitía escribir para diferentes plataformas (Windows, Mac OS, Linux…). Las sucesivas versiones de OpenGL fueron incluyendo diferentes versiones del lenguaje. La última versión de GLSL, correspondiente a OpenGL 4.0, es la GLSL 4.0. Es muy similar al C, lo que también facilita considerablemente su uso.

HLSL (High-Level Shader Language) fue presentado por Microsoft en 2002, junto con Direct3D9. Sus características generales, sintaxis, tipos de funciones, organización general, son similares al lenguaje GLSL de OpenGL. Al igual que este, se basa extensamente en la programación de *vertex shaders*

y *fragment shaders*, que en este lenguaje se denominan *pixel shaders*. Una diferencia importante es que el compilador no tiene en cuenta el entorno dado por DirectX sino que envía el código directamente al lenguaje ensamblador de la GPU o bien lo traduce a código binario en módulos denominados también *vertex shaders* y *pixel shaders*. Estos módulos tienen sus propias versiones de implementación (Vertex Shader 1.0, 2.0, 3.0, etc., o Pixel Shader 1.1, 1.4, 2.0, 3.0, etc.). Esto independiza estos módulos con respecto al *driver*, lo que puede tener ventajas pero también inconvenientes. En GLSL la compilación está integrada en el *driver*, lo que da más responsabilidad a los fabricantes de la GPU pero también les da más libertad para optimizar y mejorar la arquitectura de sus productos. Otra diferencia notoria es que HLSL está diseñado por Microsoft para Direct3D, lo que asegura que funcione bien en máquinas y programas con estos productos pero probablemente no tanto en otros casos.

CG (C for Graphics) es un lenguaje de programación de *shaders* desarrollado por NVIDIA en colaboración con Microsoft para la programación de *vertex shaders* y *pixel shaders*. La primera versión apareció en 2002, el mismo año que HLSL. Es muy similar a este lenguaje, tan similar que según algunos son lo mismo y solo se diferencian por razones comerciales. Y porque HLSL solo puede compilarse en plataformas que incorporen DirectX mientras que CG se puede compilar tanto con DirectX como con OpenGL.

La función de CG es desarrollar un *shader* que luego se compila en el lenguaje ensamblador de la GPU adaptándolo a la plataforma específica, DirectX u OpenGL. Para adaptarse a los diferentes tipos de tarjetas sin exponer el código se utiliza el concepto de *perfil* que compila el código según el perfil propio de la plataforma anfitriona y optimiza el *shader* correspondiente para sacar partido del perfil correspondiente. Está basado en C y la principal diferencia es que hay tipos de datos ampliados para hacer que el lenguaje sea más adecuado a la programación de gráficos (por ejemplo “sampler”, que representa una textura).

CG puede utilizarse tanto con DirectX como con OpenGL. Sin embargo, al ser neutral con respecto a la plataforma hay ciertas cosas que no están incluidas, tales como el manejo de *vertex buffers* y *pbuffers* o el *rendering to textures*. De hecho, un programa escrito en CG es un *vertex shader* o un *pixel shader* y requiere que todo el resto de las funciones propias del procesamiento de la *rendering pipeline* se desarrollen por uno de estos dos programas.

Las aplicaciones gráficas de propósito general (GPGPU)

Hacia 2005 comenzó a resultar cada vez más evidente que las API tradicionales estaban basadas en modelos de un nivel de abstracción que ya no era adecuado para la computación de propósito general.

Los grandes supercomputadores habían desarrollado técnicas específicas de computación en paralelo en la última década del siglo xx, técnicas como la aplicación de instrucciones únicas a múltiples datos mediante unidades trabajando en paralelo (SIMD, *single instruction, multiple data*), técnicas que se acabaron por denominar genéricamente como propias de procesadores de flujos, *stream processors*. Las GPU, a medida que iban aumentando su potencia y sus recursos, estaban especialmente preparadas para adoptar este tipo de técnicas, y los fabricantes tenían el incentivo comercial de que, de este modo, se ampliaría su clientela a diversos campos científicos, como la biotecnología o las finanzas, además de las aplicaciones gráficas tradicionales. Así surgió, en torno a la década de 2000, lo que se denomina ya corrientemente “Computación de propósito general sobre GPU”, ***GPGPU*** (*general purpose computing on graphics processing units*).

En 2006, ATI sacó un primer producto destinado a estas nuevas aplicaciones, que se denominó *close to metal* y que proporciona-

ba acceso directo a elementos de bajo nivel y recursos para llevar a cabo métodos de computación paralela masiva. Este producto tuvo una vida corta debido, por un lado, a que ATI (AMD) decidió apoyar el nuevo estándar OpenCL y al desarrollo de FireStream.

FireStream (ATI) / Stream processor (AMD)

Las investigaciones de ATI sobre aplicaciones generalizadas de la GPU se concretaron en la nueva tecnología ATI FireStream. En 2006, con la adquisición de ATI por AMD, esta tecnología se reconstruyó y se renombró como AMD Stream processor. Estaba basado en la serie Radeon X1900 y se aplicaba a las GPU codificadas como Radeon R580. Se utilizó en campos científicos y financieros. Según AMD, la capacidad de los nuevos procesadores equipados con esta nueva tecnología era ocho veces superior a la de los anteriores.

Hasta la fecha (2014) han aparecido cuatro generaciones. La primera, a mediados de 2007. La segunda, el AMD FireStream 9170, poco tiempo después, estaba basada en tecnología de 55 nm, al igual que la siguiente, para la RV670. La tercera, el AMD FireStream 9250 y 9270, estaba basada en la RV770 y apareció en junio de 2008 (la 9250) y noviembre de 2008 (la 9270). Pueden encontrarse datos más recientes, incluida la cuarta y última versión, en internet.

Puede programarse con varios lenguajes, basados en C, con extensiones al lenguaje Brook+, que es a su vez una extensión de BrookGPU, un lenguaje de programación de flujos desarrollado por la universidad de Stanford, dirigido principalmente a GPGPU, es decir, a facilitar la programación en paralelo de GPU para aplicaciones de propósito general. Su primera versión beta importante apareció en 2004 y le siguió otra versión beta mejorada en 2007, que incluía interfaces similares a las de OpenGL. Las versiones posteriores se denominaron Brook+ e incluían computaciones de doble precisión especialmente adaptadas a las GPU de AMD.

CUDA (NVIDIA)

Por su parte, en 2007, NVIDIA lanzó la primera versión de ***CUDA*** (*computer unifed device architecture*), un conjunto de herramientas de desarrollo de algoritmos para sus GPU (a partir de la serie G8X), basadas en el lenguaje C y ligadas a un compilador específico. Su motivación principal era explotar las posibilidades de la GPU explotando el paralelismo ofrecido por sus numerosos núcleos que permiten el lanzamiento de varios hilos (*threads*) simultáneos.

La mayoría de los lenguajes de programación no cuentan con estructuras nativas que permitan hacer que los procesos se desarrollen en paralelo. En el mejor de los casos pueden iniciar varios hilos de ejecución dejando en manos de la CPU la gestión de estos hilos, pero con el requisito de tener que modificar el código fuente según la arquitectura de hardware en que se vaya a ejecutar el programa.

La gran aportación de CUDA es que cuenta con estas estructuras y que permite explotar el código desde la GPU, lo que también permite aumentar de un modo espectacular el rendimiento pero sin que el programador tenga que ocuparse de este asunto. El programador escribe el código como si se fuera a ejecutar en un único hilo sin preocuparse de la gestión, la sincronización, etc. El código será ejecutado descomponiéndolo en el número de hilos que sea necesario y asignando cada uno a un núcleo del proceso pero sin que el programador tenga que preocuparse de la gestión de estas descomposiciones.

CUDA se proporciona como un conjunto de tres componentes principales que pueden descargarse gratuitamente de la url de NVIDIA.

El controlador CUDA es el primer componente a instalar y facilita la ejecución de los programas y la conexión entre la GPU y la CPU. El segundo componente es el *toolkit*, que incluye un compilador (*nvcc*), un perfilador de código y una serie de bibliotecas de funciones así como una guía introductoria. El tercer componente es el *CUDA Developer SDK*, que incluye códigos de ejemplo y documentación

adicional. Los códigos de ejemplos incluyen, entre otros, varios proyectos que muestran cómo integrar CUDA en DirectX y OpenGL

Una de sus limitaciones principales era que no contemplaba el *rendering* de texturas. También tiene algunas limitaciones desde el punto de vista de la programación, como que no se puede utilizar recursividad o punteros a funciones o funciones con parámetros variables. Con todo, la aceleración de los cálculos proporcionada por el alto paralelismo compensa sobradamente estas limitaciones.

Todas las tarjetas gráficas de NVIDIA de los últimos años incluyen varios "núcleos CUDA", tantos más cuanto más potente sea la tarjeta en cuestión. Y cuantos más núcleos se incluyan, tantas más tareas se podrán ejecutar de modo simultáneo, lo que aumenta extraordinariamente la eficacia de la GPU. Así que puede asociarse, con ciertas reservas, el mayor número de núcleos con la mayor velocidad de procesamiento. Pero las reservas son debidas a que no en todos los casos se puede sacar partido de la ejecución simultánea de varias tareas.

OpenCL

En 2009 apareció OpenCL (*open computing laguage*), un lenguaje de computación abierto que se basa en un lenguaje de programación (basado en C) ligado a una interfaz de programación de aplicaciones. OpenCL está especialmente preparado para crear aplicaciones que hagan un uso extenso del paralelismo para el procesamiento de datos y tareas que puedan ejecutarse en unidades independientes. Estas unidades pueden residir tanto en la CPU como en la GPU. La especificación original se llevó a cabo por Apple pero se la cedió al grupo Khronos (un consorcio industrial creado para apoyar las API de estándares abiertos, al que se unió OpenGL en julio de 2006) que creó un grupo especial, el Compute Working Group, en 2008, para convertirla en un estándar abierto (ver www.khronos.org).

OpenCL es particularmente adecuado para acceder a recursos heterogéneos. Soporta la ejecución en paralelo tanto en procesadores únicos como múltiples y admite computación en coma flotante con la precisión especificada por el lenguaje. Está preparado para trabajar con API gráficas, como OpenGL. Se basa en un nivel de abstracción del hardware que tiene presente los desarrollos previsibles a corto plazo.

Su estructura básica es similar a la de OpenGL y DirectX: *a*) una capa de relación con la plataforma en la que se crea un contexto, a través de la cual se seleccionan los dispositivos presentes en el sistema, se inicializan los dispositivos virtuales y se crea un bucle de procesamientos de mensajes, *b*) una estructura de ejecución en la que se gestionan los recursos y se ejecutan las instrucciones principales ligadas a las diferentes unidades de procesamiento, los *compute kernels*, que potencian la ejecución en paralelo de datos y tareas específicas.

La ejecución está ligada a la existencia de un número determinado de dominios de computación (*computational domains*). Cada elemento de un dominio se denomina "elemento de trabajo", *work-item*. El dominio define el número total de *work-items* que se ejecutarán en paralelo, el tamaño global del *work*. Los *work-items* se pueden agrupar y comunicar entre ellos y sincronizar sus ejecuciones de modo que se controle el acceso a la memoria.

La memoria está también gestionada de un modo flexible, con múltiples direcciones de espacios accesibles de diferentes modos y que pueden ser etiquetados con distintos cualificadores (*private*, *local*, *constant*, *global*…).

Los tipos de datos son los corrientes en lenguajes dirigidos a la computación gráfica: vectores y escalares, *structs*, punteros y también imágenes, como tipos específicos. También las funciones principales son las que se encuentran corrientemente en estos lenguajes: funciones mátemáticas (la biblioteca math.h de C), funciones de lectura y escritura de matrices ligadas a imágenes, funciones geométricas, etc., y funciones específicas de sincronización.

3.3 Procesamiento de materiales e iluminación

Estructuras generales

Definición y registro de parámetros básicos de materiales

En Direct3D y OpenGL los **materiales** son, en principio, estructuras simples que definen el color del material y el modo en que este color puede quedar modificado por la luz, en caso de que esta incida sobre la superficie en una dirección tal que resulte visible desde el punto de vista en que se vaya a procesar la escena.

En general estas estructuras admiten al menos cuatro valores: *diffuse*, *ambient*, *emissive* y *specular*. Son los mismos valores que se encuentran en cualquier programa de simulación: corresponden al color local, al color de la zona en sombra, al color de la emisión en caso de que el material sea emisivo y al color del reflejo en caso de que el material sea reflectante.

Una vez que se han definido estos valores se utilizan métodos específicos para, en su caso, modificar los valores de los vértices. Los valores son de tipo RGBA (*red*, *green*, *blue*, *alpha*). El pseudocódigo podría ser algo así:

```
tipoApiMaterial nuevoMaterial;
... [asignación de memoria]
nuevoMaterial.Diffuse = RGBA (1.0, 1.0, 1.0, 0.0);
nuevoMaterial.Ambient = RGBA (0.0, 0.0, 0.0, 0.0);
nuevoMaterial.Specular = RGBA (0.0, 0.0, 0.0, 0.0);
nuevoMaterial.Emissive = RGBA (0.0, 0.0, 0.0, 0.0);
dispositivo >> SetMaterial(&nuevoMaterial);
```

La primera línea define un material (“nuevoMaterial”); la segunda le asigna un lugar en la memoria; las siguientes líneas asignan un color a los cuatro valores citados, y la última línea inicializa el material.

Los colores se asignan a los vértices como datos de entrada en el estadio a cargo del *vertex shader* y se interpolan para obtener los valores intermedios de cada superficie triangular en el estadio a cargo del *pixel shader*.

Durante el estadio del *vertex shader* también se pueden modificar los colores del material por la acción de luces virtuales directas. Dado que estas son las principales operaciones que tienen lugar, junto con las transformaciones de coordenadas, a veces se denomina a este estadio T&L (*transformation and lighting*).

También en este estadio tiene lugar un primer procesamiento de texturas que, por su importancia, se tratará con algo más de detalle en el apartado siguiente.

Modificación de parámetros básicos por la iluminación de la escena

Por lo que respecta a la **iluminación**, lo primero es recordar que los programas de simulación cuentan con recursos avanzados para crear luces que simulan las luces reales y para desarrollar cálculos de iluminación avanzada (iluminación global), que computen como quedaría iluminada una escena determinada con esos tipos de luces. Pero este tipo de simulación de la iluminación, que se describe con detalle en el libro que he citado en la introducción, consume muchos recursos, es muy lento y requiere procesos que, en general, tienen que llevarse a cabo externamente a la GPU.

En casos más generales o simplemente para visualizar la escena correctamente, hay que utilizar métodos más sencillos como los que están incorporados desde las primeras versiones a Direct3D y OpenGL. Aquí, la iluminación se basa en la asignación de valores a los vértices. Cada vértice recibe un determinado peso (*weight*) que puede estar activado o desactivado y, en caso de que esté activado, puede ser ajustado por un sistema general muy básico o modificado por luces simples que se incluyan en la escena.

Es decir, que todo se basa en tres mecanismos muy simples. En primer lugar hay que activar o desactivar la iluminación. El resultado aparentemente paradójico es que, si se activa la iluminación pero no hay luces en

la escena, todo aparecerá negro. Pero si se desactiva, sin luces en la escena, aparecerá iluminado, pues el programa utiliza un sistema de luces por defecto, al igual que ocurre en muchos programas comerciales de *rendering*.

Suponiendo por tanto que queramos utilizar nuestras propias luces, el primer paso es activar la iluminación. En pseudocódigo esto supondría escribir algo así:

```
dispositivo->SetRenderState(Lighting, True)
```

Una vez que la iluminación está activada, se pueden utilizar dos modos simples de iluminación: *ambient lighting* y *direct lighting*. Con *ambient lighting*, un método que ya está en desuso en programas avanzados de simulación pero se mantiene por razones de compatibilidad y se utiliza en algún caso en videojuegos, todo queda iluminado por igual lo que evita que aparezcan superficies completamente negras. Con *direct lighting* todo queda iluminado según una dirección dada por un vector. Cada uno de estos dos modos puede activarse mediante funciones simples semejantes a la anterior.

Por añadidura pueden crearse (o utilizarse) tres tipos de luces: luces puntuales (*point lights*), focales (*spot lights*) y direccionales (*directional lights*). Las luces puntuales tienen una posición dada en el espacio e irradian por igual en todas direcciones. Las luces focales tienen una posición dada e irradian en una dirección principal. Además, tienen un cono interno y un cono externo que determinan cómo se distribuye su iluminación. En la zona que abarca el cono interno la intensidad es máxima. En la zona comprendida entre el cono interno y externo la intensidad pasa de ser máxima a ser nula. Las luces direccionales no requieren una posición en el espacio e irradian en una dirección única.

Para crear estas luces se utilizan funciones específicas que piden los valores que hemos citado en cada caso: posición, dirección, ángulo de los conos internos y externos, junto con otro valor común a las tres, la intensidad y el color, dados por tres valores RGB.

Tanto en Direct3D como OpenGL hay dos métodos básicos de cálculo de la iluminación: *flat shading* y *smooth shading* (o *Gouraud shading*) .

En el primer caso se computa el producto vectorial punto del vector Luz normalizado con el del vector normal a la superficie igualmente normalizado. Como este producto es el coseno del ángulo formado por los dos vectores y este coseno es 0 cuando el ángulo es 90º y 1 cuando el ángulo es 0º, resulta sencillo ajustar la intensidad en función de un valor lineal comprendido entre 0 y 1, lo que se corresponde con la menor o mayor incidencia de la luz sobre la superficie. Véase la figura 3.9, correspondiente al cálculo de iluminación que se ha dado en secciones anteriores.

En el segundo caso se aplica el mismo procedimiento pero se envía la información adecuada al *pixel shader* para que interpole estos valores entre los vértices correspondientes. El resultado es que la superficie mostrará un degradado continuo en lugar de facetas planas lo que producirá el efecto de que la superficie tiene una curvatura continua (un efecto traicionado por los contornos).

Cada uno de estos modos depende por añadidura de las propiedades de los materiales que ya he citado en el apartado anterior: color difuso, ambiente, emisivo y especular que modifican esta modelación básica. El cálculo de iluminación da lugar, por consiguiente, a que cada vértice cuente con un color primario (correspondiente a la definición de material) y un color secundario (correspondiente, en principio, a su modificación por la luz).

Texturas

Estructura y registro de los datos

Un mapa de textura se crea a partir de la lectura de un archivo o a partir de un algoritmo que activa un proceso de generación artificial o a partir de operaciones *per-pixel* que se aplican a datos que ya residen en el *frame buffer* y se reenvían a la *pipeline*. En cualquier

caso, el mapa de textura es un conjunto de datos estructurados muy importante, tanto por su finalidad como por su tamaño.

El código para gestionar estos datos tendría un aspecto como lo que sigue:

```
apiTextura2D (clase, width, height, format, type, miplevel, border, datos)
```

donde "apiTexture2D" sería la función de creación propia de cada API y los argumentos tendrían el significado siguiente: *clase*, el tipo de objeto predeterminado, en su caso (por ejemplo "textura2D"), *width*, *height* las dimensiones en píxeles, *format* el formato de los datos (por ejemplo RGBA, es decir *red*, *green*, *blue*, *alpha*), *type* el tipo de datos (normalmente *unsigned_byte*), *miplevel* el nivel de *mipmap* (véase las explicaciones que se dan más adelante), *border*, el tipo de borde (corrientemente 0) y *datos*, la colección de valores que constituyen concretamente la textura.

La *lectura de un archivo* implica, en primer lugar, una *descodificación* de los datos, que llegarán estructurados en un formato específico (tif, jpg, gif, png, etc.), para convertirlos en datos estructurados en un *array*, un tipo de datos corriente en cualquier lenguaje. Dado el volumen y la importancia de estos datos, la creación de *buffers* dedicados a las texturas fue una de las primeras mejoras de las tarjetas gráficas. Estos *texture buffers* pueden rellenarse con datos provenientes de esta decodificación inicial pero, también, de datos reprocesados que ya estaban almacenados y habían seguido su curso en la *pipeline* pero que pueden haber sido sometidos a operaciones específicas (*per-pixel operations*). Un *buffer* de textura es un *array* de *m* x *n* x *k* en donde *m*, *n* son las dimensiones espaciales (correspondientes en principio a píxeles) y *k* la profundidad en bpp (*bits per pixel*).

Una textura 2D es, por tanto, una matriz de datos. Aunque la idea general es que estos datos se corresponden con imágenes esto no es así en muchos e importantes casos. Por ejemplo, un *normal map*, que se utiliza para simular relieves, también contiene los mismos tipos de datos (3 colores de 8 bytes) que una textura corriente, una imagen. Pero en el caso del *normal map* estos colores representan información geométrica que se traduce en una apariencia de relieve, como se verá en otros capítulos, más adelante. Y, por otro lado, además de texturas 2D, que es el tipo más corriente, hay también texturas 1D y texturas 3D.

La aplicación de mapas de textura para finalidades distintas que la representación de materiales, finalidades tales como simulación de relieve (*bump map*, *normal map*), simulación de reflejos (*environment map*) y otras, se lleva a cabo con funciones específicas durante la fase de rasterización.

Para que las texturas puedan relacionarse con la geometría, los vértices deben incluir un atributo específico que ya he citado: las *coordenadas de textura*. Estas coordenadas se procesan durante el estadio correspondiente a las transformaciones generales, el *vertex shader*, como un dato geométrico más y no se relacionan con el mapa hasta los estadios finales. Estas coordenadas, denominadas convencionalmente (*u*,*v*), identifican un elemento de la textura que se denomina un *texel*. Lo que se denomina el "espacio de textura" (*texture space*) está normalizado y va de 0,0 a 1,1. El valor 0,0 corresponde generalmente al extremo superior izquierdo. Cada *texel* queda situado en este espacio 2D, el espacio de textura.

Las coordenadas de textura vienen dadas desde la aplicación principal, sea mediante especificaciones del usuario, sea mediante especificaciones automáticas y, como decía, es parte de la información incluida en los vértices, independiente del mapa específico que vaya a ser proyectado sobre la geometría. Esto quiere decir que la estructura de vértices, en caso de incluir coordenadas de textura, tendría este aspecto:

```
struct MisVertices
{
    vector3 xyz;
    vector3 color;
    float u, v
};
```

Es decir, un vector 3D que se corresponde a la posición geométrica, un vector 3D que se corresponde al color (prescindimos del posible cuarto canal, alfa), y un decimal en coma flotante que corresponde a las coordenadas *u*, *v*.

Los datos correspondientes al mapa o mapas se almacenan, como ya hemos visto, en un *texture buffer*, la zona de memoria local que se utiliza para almacenar mapas de textura. Si el espacio es insuficiente, se mantienen en la memoria del sistema hasta que son requeridos. Esto supone un retardo importante. Las aplicaciones basadas en la CPU no necesitan preocuparse de dónde se ponen las texturas pues prácticamente solo hay dos opciones: registros y memoria global (a la que se accede a través de *caches* denominadas L1, L2, L3…). Pero la programación de la GPU implica utilizar múltiples recursos de memoria para acelerar los procesos. De ahí que una GPU será tanto más potente cuanto más *texture buffers* tenga disponibles para evitar este tráfico entre la CPU y la GPU que retrasa el cálculo. Pese a todo, esto puede ocurrir y se puede reducir de varios modos. Un modo es contar con *caches*, zonas de memoria que atrapan los datos más frecuentes y reducen así el tiempo de acceso y que también están incorporadas a las modernas GPU. Otro recurso importante que veremos más adelante es el uso de *mipmaps* para reducir el nivel de detalle y, en consecuencia, el tamaño efectivo de los mapas. Esto implica utilizar operaciones denominadas *texture fetch* que, en pocas palabras, implican la carga de una imagen y su reconstrucción por medio de remuestreos (*resampling*) a partir de operaciones de filtrado como las que veremos más adelante y que se aplican a las diferentes unidades de textura (*texture units*) cargadas en la *pipeline*. Una *texture unit* es la pieza de hardware que lleva a cabo las operaciones sobre texturas. La idea básica de este tipo de operaciones es muy simple: se toma un *texel*, es decir el valor (color) de una imagen en una determi-

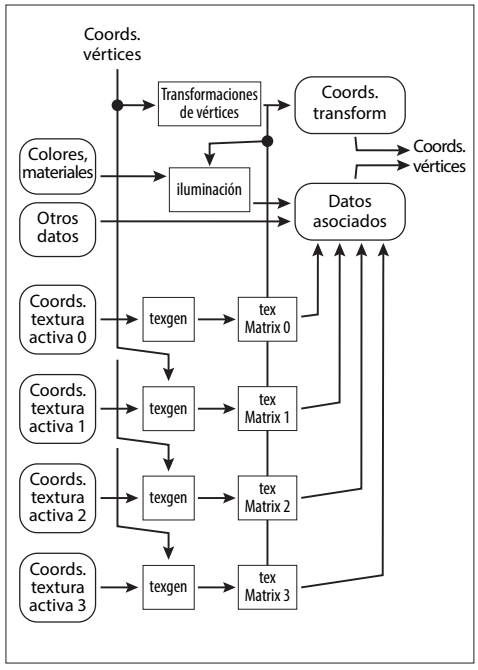


Figura 3.24 Texturas. Relación con el vertex shader.

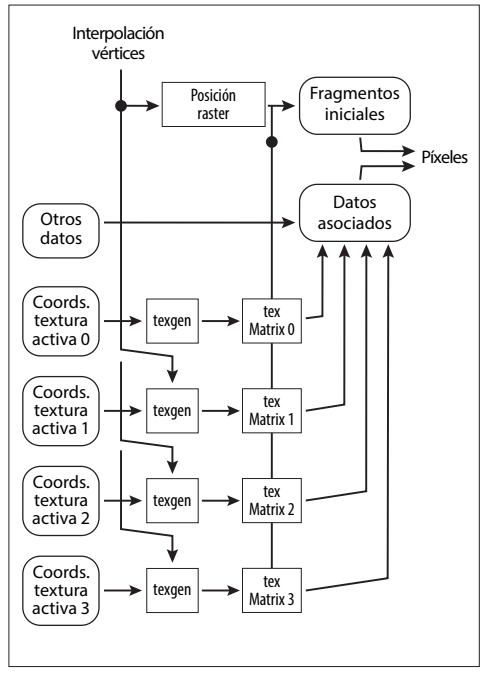


Figura 3.25 Texturas. Relación con el pixel shader.

nada posición, se combina con los valores derivados de otros recursos y se envía al estadio siguiente.

Pero para entender la importancia de estas operaciones, en cuyo detalle no podemos entrar, baste decir que una instrucción de GPU de tipo *texture fetch* se considera que es aproximadamente dos órdenes de magnitud más compleja que cualquier otra de la GPU.

Las texturas se acoplan al objeto de varios modos que se denominan genéricamente *wrap modes* (o *wrapping*), "modos de envoltura" y están gestionados por el recurso *texture sampler*. Los modos principales son: *a*) repetir (*wrap* en Direct3D, *repeat* en OpenGL) y sus variantes, tales como la repetición simétrica (*mirror*), *b*) recortar (*clamp*) y sus variantes: recortar hasta el borde (*clamp* en Direct3D, *clamp to edge* en OpenGL, *border* en Direct3D, *clamp to border* en OpenGL). La primera fuerza el mapa a repetirse a intervalos regulares, de modo constante o con simetrías alternadas. La segunda recorta el mapa sobre el objeto de diferentes modos que se ilustran en la figura 3.26: *clamp* / *clamp* to *edge* hace que los valores fuera del rango 0,1 reciban el valor de los límites de la textura; *border* / *clamp to border* hace que los valores fuera del rango 0,1 reciban un valor de borde único.

Las texturas pueden estar ligadas a diferentes *estadios* de la *pipeline*. Hay dos estadios principales a los que se ligan las texturas: al *vertex shader* y al *rasterize* / *pixel* / *fragment shader*, que es el principal. También pueden ligarse, como una *render target*, al estadio final de la *pipeline*. Pero también, ocasionalmente, desde el *geometry shader* o en las versiones recientes, desde los *shaders* ligados al estadio de *tessellation*.

Cuando las texturas se ligan al *vertex shader* es para proporcionar información que podría afectar a la reestructuración de los vértices. Un recurso reciente, por ejemplo, son los *vertex texture fetch*, que se utilizan para crear nuevos vértices ligados, por ejemplo, a mapas de desplazamiento.

Cuando las texturas se ligan al proceso de rasterización, que es lo más usual, las proyecciones de las texturas y los complejos procesos de recombinación que veremos a continuación solo se aplican a los píxeles que hayan sobrevivido a lo largo de los estadios anteriores (*clipping*, *culling*, *depth texts*, etc.) con lo que el proceso es más eficiente. En este estadio, las coordenadas de textura ya se han procesado y convertido a lo largo de los estadios anteriores, por lo que ya están incorporadas a la información con que se cuenta.

El modo de gestionar las texturas sigue evolucionando y varía entre las API. La tendencia general es flexibilizar los modos de asignación para dar más libertad a los programadores, de modo que puedan enviar un mismo mapa a los *shaders* en estadios intermedios según lo que se necesite. Direct3D utiliza el concepto técnico de "vista" (*view*). Cada textura se asocia a una "vista" (*resource view*) que puede utilizarse de diferentes modos. En la versión 10 había 3 interfaces de este tipo (*ID3D10DepthStencilView*, *ID3D10RenderTargetView* y *ID3D10ShaderResourceView*), las dos primeras utilizadas para

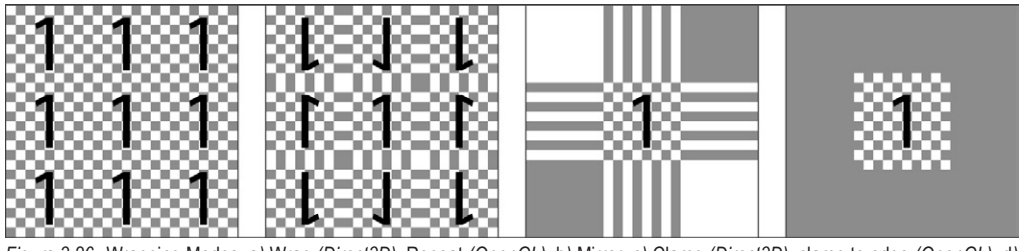


Figura 3.26 Wrapping Modes*: a)* Wrap *(Direct3D),* Repeat *(OpenGL),* b) Mirror, c) Clamp *(Direct3D),* clamp to edge *(OpenGL),* d) Border *(Direct3D),* Clamp to border *(OpenGL).*

ligarlas (*bind*) al último estadio (*output merge*) y la segunda para ligarla a los *shaders*. Dado que el mapa permanece en su estado original en otro buffer pueden crearse múltiples "vistas" que pueden configurarse de diferentes modos, por ejemplo con diferentes parámetros de *mipmap* que como veremos a continuación, es una de las especificaciones más corrientes para gestionar la aplicación concreta de una determinada textura a una determinada salida.

Filtrado básico de texturas

La operación principal que debe llevarse a cabo con la aplicación de las texturas a objetos es el filtrado de texturas. Esta operación es necesaria cada vez que se aplica una textura a un objeto cuya salida por pantalla se produce sobre un número de píxeles que no coincide con el número de *texels*. Es decir: en la práctica totalidad de los casos.

Hay varias operaciones estandarizadas de filtrado. Las dos grandes categorías son *magnification* y *minification*. Los métodos concretos son *nearest-point*, *linear*, *mipmap* y *anisotropic*.

Para ver con algo más de detalle cómo funcionan estas operaciones de filtrado, supongamos que tenemos una textura que ocupa 256 x 256 píxeles. Y que esta textura va a proyectarse sobre una zona de salida que es más grande o que es más pequeña. Para concretarlo con un ejemplo muy simple (y que raramente se daría), supongamos que en un caso se va a proyectar sobre una zona 4 veces mayor, de 1.024 x 1.024 y en otro caso se va a proyectar sobre una zona 4 veces menor, de 64 x 64.

En el primer caso, denominado genéricamente *magnification*, tendríamos un problema que se puede resolver tomando un valor único para los varios píxeles que corresponden a un único *texel*. En el segundo caso, denominado genéricamente *minification*, tendríamos un problema inverso que se podría resolver tomando un valor único para los varios *texels* que se van a proyectar sobre un único píxel.

El modo de resolver este problema puede ser más o menos preciso y más o menos rápido. Las GPU utilizan corrientemente algunos de los siguientes filtros que se aplican durante el estadio de rasterización, cuando se invocan los mapas de textura situados en zonas específicas de memoria. Todos estos filtros pueden utilizarse tanto en los procesos de magnificación como de minificación. La especificación comienza generalmente por declarar un "estado de muestreo" (*sampler state*) que puede ser de uno de estos dos tipos, a los que se puede añadir un tercero para especificar si se va a utilizar *mipmapping* (que está implicado en el filtrado trilineal que se describe más adelante).

Nearest-point. También se denomina *point filtering* o *nearest neighbor*. Este método, "el punto más cercano", es más rápido pero también más impreciso y puede dar lugar a diversos artefactos. Consiste en escoger un *texel* del mapa de texturas que corresponda al valor más cercano del píxel correspondiente según venga dado por las coordenadas de textura. El valor escogido es el dado directamente por

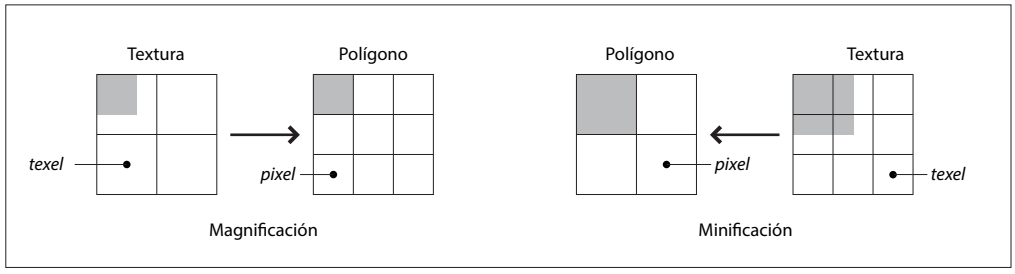


Figura 3.27 Magnificación y minificación.

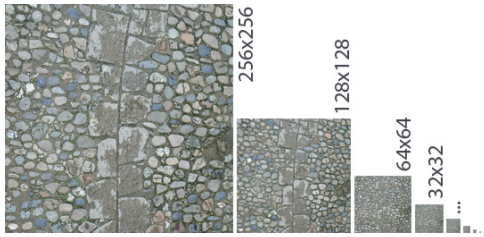


Figura 3.28 Filtro mipmap.

este *texel*. Si el proceso es de magnificación, los píxeles se multiplican. Si el proceso es de minificación, varios píxeles son descartados, lo que da lugar a *aliasing*.

Linear. Al igual que el anterior, parte del valor más cercano al definido por las coordenadas de textura. Pero en lugar de escoger el valor dado directamente por el *texel* toma una media lineal, ponderada, de los valores que le rodean. Hay dos variantes de este método. La más corriente es el filtrado bilineal (*bilinear filtering*), donde se utilizan 4 *texels* para producir 1 píxel siguiendo un orden tipo norte-sur-este-oeste. Una extensión de este procedimiento es el filtrado trilineal (*trilinear filtering*) utilizado con *mimaps* y denominado también filtrado trilineal de *mipmaps* (*mipmap trilinear filtering*), en el que el proceso se repite en dos *mipmaps* sucesivos lo que reduce las fronteras visibles en *mipmaps* sobre objetos grandes, aunque es más lento pues se utilizan 8 *texels* para producir 1 píxel. Si el proceso es de magnificación, se crean gradientes suaves entre los píxeles. Si el proceso es de minificación bilineal, se eliminan píxeles tomando sus valores medios, lo que también produce *aliasing* pero menos notorio que en el caso anterior. Si el proceso es de minificación trilineal (*mipmapping*) se reduce el *aliasing* a costa de menor rendimiento.

Anisotropic. Los métodos anteriores son isotrópicos, no tienen en cuenta diferencias de orientación. Pero en muchos casos estas diferencias son importantes pues la textura puede quedar más inclinada con respecto a la vista en una dirección que en otra, con lo que su proyección no es homogénea. En estos casos se utiliza un filtro anisotrópico que implica cálculos adicionales previos a la proyección, entre ellos la medida del ángulo de desviación con respecto a la cámara: cuanto mayor sea el ángulo mayor será la anisotropía y más muestras se tomarán para corregirla. La anisotropía puede ser también debida a las propias características de la textura, pero esto es otra historia que se tratará más adelante en el capítulo sobre técnicas. La anisotropía de la que estamos hablando se refiere solamente a distorsiones creadas por el punto de vista.

Los filtros anisotrópicos pueden, a su vez, ser de varios tipos que se distinguen por el número de *texels* procesados, que pueden especificarse en el *driver* en función de las disponibilidades. Un filtro anisotrópico 1x procesa 8 *texels*. Y cada nivel sucesivo procesa el doble: 2x (16 *texels*), 4x (32 *texels*), 8x (64 *texels*), 16x (128 *texels*). Y cada nivel sucesivo implica un aumento correspondiente del tiempo de procesamiento.

Durante el proceso de minificación se puede utilizar *mipmapping*. Esta técnica se usa para yuxtaponer diferentes versiones de una textura a diferentes tamaños, de tal modo que las partes más distantes se representen con un tamaño optimizado y, además, se aplique un filtro adecuado para evitar artefactos. Puede utilizarse en realidad con cualquiera de los tres tipos de filtros citados, punto (*nearest*), bilineal y trilineal, aunque da mejores resultados con el tercero. Los modos principales son *mipmapping bilineal*, *mipmapping trilineal* y *mipmapping trilineal anisotrópico*.

La API genera automáticamente diferentes versiones de la misma textura a la mitad de resolución. Por ejemplo (ver la figura 3.28), una textura de 256 x 256 daría lugar a texturas agrupadas de 128 x 128, 64 x 64, 32 x 32, 16 x 16, 8 x 8, 4 x 4, 2 x 2 y 1 x 1. Según la distancia de visualización y los píxeles cubiertos se utilizaría una u otra versión de dicha textura. Al haber una correspondencia exacta entre los píxeles, minimizada en una potencia de 2, la transición entre uno y otro, con la ayuda del filtro, será imperceptible.

El problema, resumido, es que si se hace un zoom para acercarse a una superficie texturizada se verá cada vez más pixelada. Y si se hace un zoom para alejarse, los detalles aparecerán y desaparecerán de un modo aleatorio.

Para solucionar este problema las tarjetas gráficas incorporan, como solución principal, *mipmaps*, por lo general prefiltrados para que se vean mejor a mayores distancias. Pero a corta distancia la pixelación permanecerá y la única solución será utilizar mapas mayores.

La solución más general es la misma que se usa con *antialiasing*: supermuestreo seguido de fusión de los valores. En esto consiste básicamente el filtrado de texturas. El problema es que se necesita más ancho de banda para manejar las grandes texturas temporales.

El sistema más efectivo es el filtrado bilineal. En lugar de tomar una muestra correspondiente al centro del píxel que se va a representar se toman cuatro muestras y se computa el valor ponderado correspondiente. Así se reduce el pixelado, pero a costa de cierto grado de desenfoque. Funciona mejor en combinación con *mipmaps* aunque tiene el inconveniente de que se crean discontinuidades en los bordes, donde la textura pasa de enfocada a desenfocada.

El filtrado trilineal funciona de modo similar al bilineal pero reduce el problema de los bordes de los *mipmaps* tomando muestras de los bordes de los niveles de *mipmap* próximos (8 muestras por píxel en total). Pero eso es todo, los detalles se pierden igualmente.

El filtrado anisotrópico se utiliza con texturas que aparecen en superficies 3D y cuyas propiedades dependen de la deformación perspectiva y de la dirección de esta deformación. Se basa en tomar muestras múltiples, bilineales o trilineales, para cada píxel de la imagen de salida para prevenir la pérdida de resolución debida a la deformación que aumenta con el ángulo de visión. El mayor número de muestras tomadas en la vecindad del píxel y en la dirección de la pendiente, atenúa esta pérdida.

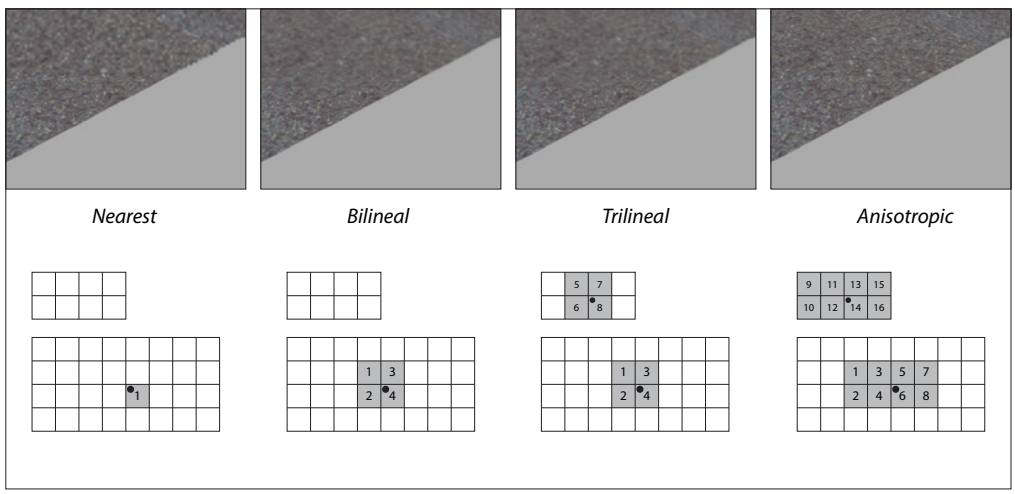


Figura 3.29 Las imágenes superiores muestran de modo esquemático los resultados de una textura proyectada con mip-mapping *y filtrada de diferentes modos. Las fronteras del* mip-map *son apreciables en las dos primeras mientras que quedan difumidas en las siguientes. El esquema inferior muestra un esquema de dos niveles del* mip-map *donde cada celda corresponde a un* texel *y, en cada caso, se muestra, con un punto, la posición equivalente del píxel procesado y, con trama y números, los* texels *muestreados para ese píxel.*

La figura 3.29 ilustra esquemáticamente el uso de estos filtros.

Por último, hay que añadir que las últimas versiones de las dos API admiten texturas que no sean potencias de 2, a diferencia de versiones anteriores que no las admitían y remuestreaban las texturas que no cumplieran esta condición para adaptarlas a los procesos incorporados a la GPU. Sin embargo, debido a que los procesos internos siguen optimizados para estos tamaños, es recomendable utilizar valores de este tipo o combinaciones de estos valores, como 256 x 512 o 1024 x 1576, etc.

3.4 Aliasing y antialiasing

Notas sobre procesamiento de imágenes

Para comprender mínimamente los procesos de *aliasing* y *antialiasing* es necesaria una breve incursión en la teoría de señales. Quien quiera ampliar esta información puede consultar alguna de las obras citadas en la bibliografía. Por ejemplo, las obras de Pratt (1978, 2007), Foley *et al*. (1990, 2013), Glassner (1995) o Wandell (1995).

Imágenes y señales

Las imágenes son tipos particulares de señales. La imagen que contemplamos en un monitor está creada por una única señal, continua, que recorre la pantalla de izquierda a derecha y de arriba abajo unas 60 veces por segundo, lo suficientemente rápido para que el ojo no capte las discontinuidades de la señal. Si esa imagen ha sido captada por otro dispositivo, un dispositivo de entrada, como un vídeo, debe sufrir previamente un proceso de transformación para convertirse en una señal y poder ser enviada por los aires o por los cables o ser almacenada hasta que llegue al dispositivo de salida.

A lo largo de ese proceso la señal sufre diversas transformaciones. Estas transfor-

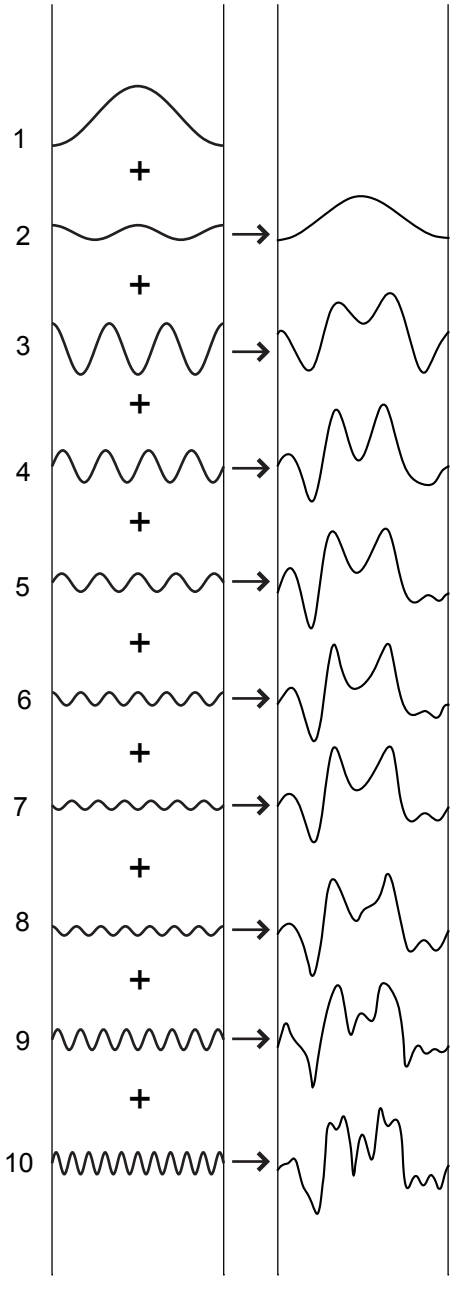


Figura 3.30 Descomposición de una señal en componentes armónicos simples (series de Fourier).

maciones se estudian en lo que se denomina corrientemente procesamiento de imágenes (*image processing*) y que trata del análisis y reconstrucción de imágenes reales. Las técnicas de simulación visual por medios digitales tratan de la generación de imágenes sintéticas. La distinción es hasta cierto punto ambigua pues a menudo las imágenes sintéticas están basadas parcialmente en la manipulación de imágenes reales.

Pero las técnicas básicas son comunes a los dos procesos. El objetivo común es generar una serie de píxeles que produzcan en un observador corriente la misma impresión que produciría la imagen real que se quiere simular, exista o no exista. Conviene subrayar que partimos de algo continuo, pasando por algo discontinuo para llegar a algo continuo. Y que el término "píxel" se utiliza a veces de un modo que se presta a confusión pues puede referirse a cosas distintas. La imagen real o ideal de la que partimos no consta de puntos sino que se percibe como un continuo. Las muestras que tomamos de esa imagen pueden ser consideradas como puntos asociadas a valores discretos. La imagen sintética, que reconstruye la imagen real o ideal a partir de las muestras puntuales, también se percibe como un continuo, aunque conste de píxeles. Pero estos píxeles pueden corresponder a agrupaciones de muestras por lo que no deben identificarse los "puntos" de la imagen sintética con los "puntos" procesados. El término *fragmentos* utilizado por OpenGL se presta menos a confusión, en este sentido, que el término *píxel* utilizado por Direct3D. Los píxeles entendidos como valores, como muestras abstractas no tienen porque ser lo mismo que los píxeles entendidos como parte de la imagen sintética final, como pequeños rectángulos que emiten luz y que se confunden en la imagen sintética final.

En cualquier caso, lo que se procesa son señales. Y trabajar con señales es trabajar con ondas sinusoidales. Jean-Baptiste Fourier (1768-1830) demostró hacia 1810 que las funciones periódicas pueden descomponerse en una suma de funciones sinusoidales simples (Dirichlet demostró, algunos años después, que estas funciones deben cumplir ciertas condiciones tales como ser integrables, estar acotadas o contar con un número finito de discontinuidades). Esto permitió el tratamiento sistemático de toda una serie de funciones de importancia capital en varias disciplinas, entre ellas la telecomunicación, pues permitía su formulación y reformulación matemática a partir de estas funciones simples derivadas de la función $y = \text{sen}(x)$. La figura 3.30 ilustra este principio básico.

Una imagen es, por tanto, un caso particular de una señal, una función que asocia un determinado valor a un determinado punto. La posición del punto viene dada en un determinado espacio o dominio que, en principio, es unidimensional. Pero que, en el caso de las imágenes, se proyecta en un espacio bidimensional por lo que también podemos hablar de una imagen como una función que relaciona un par de valores (las coordenadas del punto) con un valor (la intensidad de la señal en ese punto).

Dado que, internamente, la señal se descompone en ondas sinusoidales regulares, es corriente en la literatura técnica sobre imágenes hablar de frecuencias que irían asociadas a variaciones mayores o menores de la intensidad de un punto o una zona. Cuantos más detalles, ángulos, bordes tenga una imagen, mayor será el número de componentes con altas frecuencias, lo que puede apreciarse de un vistazo en una representación del dominio de frecuencias. Y si la imagen tiene discontinuidades, esto se traduce en frecuencias infinitas.

Una imagen con discontinuidades y, por tanto, con frecuencias infinitas no puede ser representada por ningún número discreto y finito de muestras. De ahí que las imágenes sintéticas tengan una serie de defectos característicos que se analizan en el apartado siguiente y cuya prevención o corrección son el objetivo principal de esta sección.

Teorema del muestreo y límite de Nyquist

De entrada, habría que decir que los defectos parecerían ser inevitables pues la fidelidad de la reproducción solo podría garantizase si tomásemos un número infinito de muestras y valores, lo que es obviamente imposible. La cuestión clave que surge, por consiguiente, es determinar *a priori* el número de muestras suficientes para que los errores sean aceptables. Con esto aparecen dos cuestiones complejas: por un lado, qué se quiere decir con "errores aceptables" y, por otro lado, cómo determinar el número de muestras "suficiente". Para lo primero la respuesta es que depende del contexto en que se vaya a utilizar esta imagen, de las condiciones de observación y de la mayor o menor necesidad de fidelidad requerida en este contexto. Para lo segundo hay una respuesta muy concreta gracias a un teorema fundamental, el teorema de muestreo (*sampling theorem*). A veces se le conoce con el nombre de teorema de Nyquist-Shanonn, pero aparece bajo varias denominaciones pues fue descubierto y desarrollado de modo independiente por otros autores además de los dos mencionados. Harry Nyquist (1889-1976) fue un científico sueco que se doctoró en Yale y trabajó en los laboratorios Bell. Hizo importantes contribuciones a la teoría de la información antes de que existiese con esta denominación. Concretamente, en 1924 publicó un artículo ("Certain Topics in Telegraph Transmission Theory") donde mostraba que el número de pulsos independientes que podían ser enviados por un canal telegráfico por unidad de tiempo estaba limitado por el doble de la amplitud de banda del canal. Este límite, que ha pasado a ser conocido como el *límite de Nyquist* (*Nyquist limit*) quedó propiamente integrado en la teoría general de la información desarrollada por Claude Shannon (1916-2001), y cuyas ideas generales fueron presentadas en 1948 aunque muchos de los principios se remontaban a más de diez años atrás. Cuando Shannon demostró y presentó el teorema del muestreo (*sampling theorem*) en 1949 se refirió al intervalo de muestreo crítico T = 1/2 W como el "Nyquist interval", correspondiente a la anchura de banda W en reconocimiento a la importancia de la aportación de Nyquist en el campo de la telegrafía. Otros autores se refirieron en la década de 1950 a este intervalo como "el límite de Nyquist".

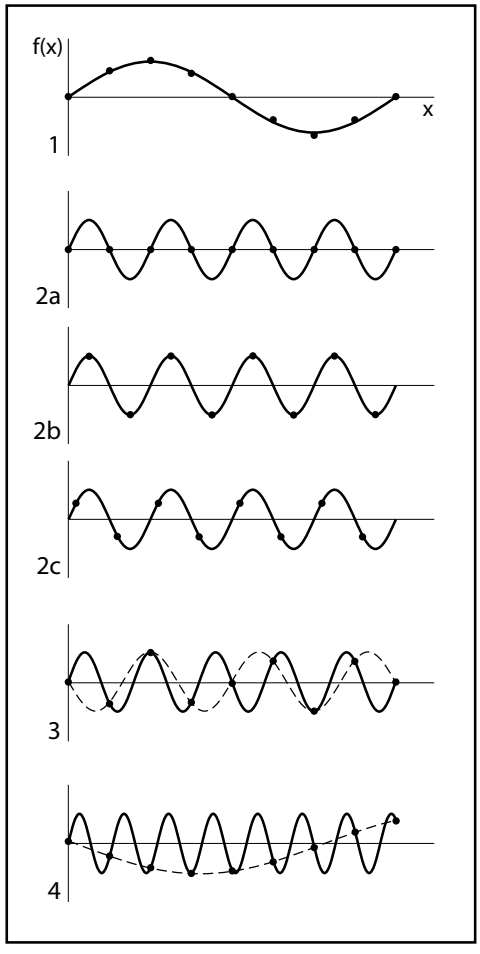


Figura 3.31 Límite de Nyquist. Un mismo conjunto de muestras, los puntos marcados en la figura, pueden corresponder a diferentes señales.

El teorema del muestreo establece que una señal que ha sido muestreada puede ser reconstruida a partir de una secuencia teóricamente infinita de muestras, si la frecuencia de muestreo es superior a 2B, siendo B

la frecuencia más alta de la señal original. Otro modo de decir lo mismo, más orientado a la práctica, es que el período de muestreo no debe ser mayor que la mitad del detalle más fino de la función que vamos a muestrear. Esta es la regla práctica que conviene recordar. La figura 3.31 ilustra la idea básica: si la frecuencia está por debajo del límite de Nyquist, diferentes señales podían ser representadas por una misma función con la subsecuente pérdida de información.

Operaciones fundamentales. Transformada de Fourier. Convolución

Como veremos más adelante, tanto para corregir errores de las imágenes, para manipularlas realzando algunos aspectos que interesan más que otros, como para introducir variaciones que interesan por motivos artísticos o científicos, se utilizan filtros, una herramienta fundamental en el procesamiento de imágenes.

Dado que las señales se manipulan por medio de fórmulas que combinan de diversos modos funciones armónicas sinusoidales, utilizar filtros quiere decir utilizar un tipo de operación adecuada para trabajar con este tipo de funciones. Y por razones técnicas complejas, pero que pueden comprenderse intuitivamente, estas operaciones se realizan mejor en el dominio de las frecuencias pues es precisamente sobre las frecuencias sobre lo que vamos a operar. Para poner un ejemplo muy simple: si el sistema que vamos a utilizar no puede resolver adecuadamente la representación de frecuencias altas (líneas muy finas, detalles poco perceptibles de objetos), la solución más sencilla es reducirlas o incluso eliminarlas.

Para poder operar sobre las frecuencias necesitamos transformar la representación de la señal, que inicialmente viene dada en el dominio espacial por la representación de la señal en otro dominio, el dominio de las frecuencias.

Una transformación es, en general, una proyección de un conjunto de valores sobre otro conjunto de valores, y una transformación de una función es otra función que se obtiene multiplicando la función original por una función especial e integrando el producto entre límites adecuados. *La transformada de Fourier* parte de un conjunto de valores, las intensidades de una serie de puntos o muestras en un determinado ámbito (el tiempo en el caso unidimensional o el espacio en el caso bidimensional), y los transforma en otro conjunto de valores, los correspondientes a diferentes frecuencias.

Si nos restringimos, como estamos haciendo en todo este texto para simplificar la exposición, al ámbito temporal y si tomamos como referencia, por claridad, el ejemplo del sonido, esta conversión se corresponde con el modo en que está organizado nuestro propio aparato auditivo, pues distinguimos naturalmente frecuencias altas o bajas hasta un cierto límite y diferencias de frecuencias, y esto se corresponde con el hecho de que también resulta más fácil analizar una señal en términos de frecuencias.

Este es el sentido del uso de la transformada de Fourier y de la transformada inversa de Fourier. Se trata de representar la señal de otro modo que sea más adecuado al caso. Los dos modos de representación se denominan, a veces, pares de transformadas de Fourier, pues puede pasarse de uno a otro según sea más conveniente.

A partir de estas consideraciones muy básicas podemos volver a los filtros. La operación fundamental que se utiliza cuando se aplican filtros en el dominio de la frecuencia, es una *convolución*: una operación matemática por la que dos funciones, f, g, se combinan para dar lugar a una tercera función, una versión modificada de las funciones originales. Se utiliza a menudo el símbolo * para indicar esta operación, de modo que $h(x) = f * g$ denotaría la convolución de las funciones f, g.

El concepto de convolución se remonta a los estudios de Laplace sobre las órbitas de cometas, que le llevaron a combinar funciones de modos originales, pero el término parece que se usó por primera vez en el siglo

XX en relación con estudios sobre ecuaciones integrales. Hilbert utilizaba el término alemán *faltung* (pliegue, doblado, trenzado…) y Norbert Wiener utilizó este término alemán en un libro de 1933 sobre la integral de Fourier y sus aplicaciones a falta de un término inglés más adecuado. Pero pocos años después se introdujo el término *convolution* como traducción del término alemán. En la actualidad, es un término fundamental en la teoría del procesamiento de imágenes.

Aunque la convolución no es una operación que pueda ser descrita con facilidad en términos corrientes podemos simplificarla, para los fines de este capítulo, diciendo que es la base matemática a partir de la cual es posible manipular las frecuencias de una imagen de diversos modos. Dicho de otro modo: es la herramienta matemática que nos permite utilizar todo tipo de "filtros" lo que, intuitivamente, se correspondería con "dejar pasar" ciertas características que nos interesan pero bloquear el paso de otras características que no nos interesan.

Aliasing y *antialiasing*. Tipos generales

En el procesamiento de señales, en general, y en el procesamiento de imágenes digitales, en particular, se denomina *aliasing* a un defecto inherente al sistema por el que aparecen distorsiones, diferencias más o menos importantes con respecto a la señal original o a la señal ideal que se quiere generar. El término viene principalmente de que, en determinados casos, aparecen "alias", señales indistinguibles que interfieren entre sí. Pero, en general, se refiere a todo tipo de distorsiones. Estos defectos pueden ser globales, afectar a toda la imagen, o locales, afectar principalmente a determinados objetos o partes de la escena.

El primer artículo importante sobre este tema, el de Franklin C. Crow (véase Crow, 1977) ya comenzaba por destacar las tres principales fuentes de problemas que afectaban, y siguen afectando, a las imágenes sintéticas: *a*) Los bordes de objetos o resaltes finos en las superficies; *b*) Los objetos muy pequeños; *c*) Las zonas con detalles intrincados. En todos estos casos puede ocurrir que haya líneas o puntos que desaparecen o reaparecen en la imagen o que se representen de modo discontinuo.

En el mismo artículo se citaban las técnicas principales para solucionar estos problemas:

a) Aumentar la resolución para que aumente el número de muestras y los puntos o líneas finas queden suficientemente cubiertos. El inconveniente de este recurso es que el tamaño de la imagen aumenta y, con ello, el coste de computación. Con todo, aumentar la resolución es la primera y más sencilla solución para eliminar el *aliasing*. Pero la siguiente pregunta es, obviamente, ¿hasta cuánto? Y la respuesta viene dada, hasta cierto punto, por el teorema fundamental en la teoría del muestreo que ya hemos visto: la frecuencia de muestreo debe ser superior a 2B siendo B la frecuencia más alta de la señal original. Dicho de un modo más sencillo: si tenemos una escena con detalles que ocuparían 1 píxel, convendrá aumentar la resolución hasta que ocupen 3 píxeles para curarse en salud.

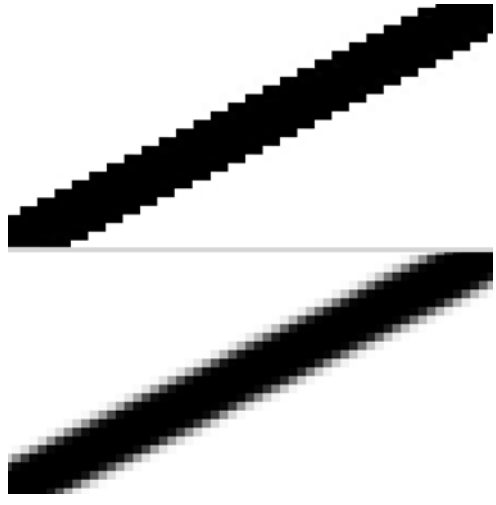


Figura 3.32 Antialiasing *por desenfoque de bordes.*

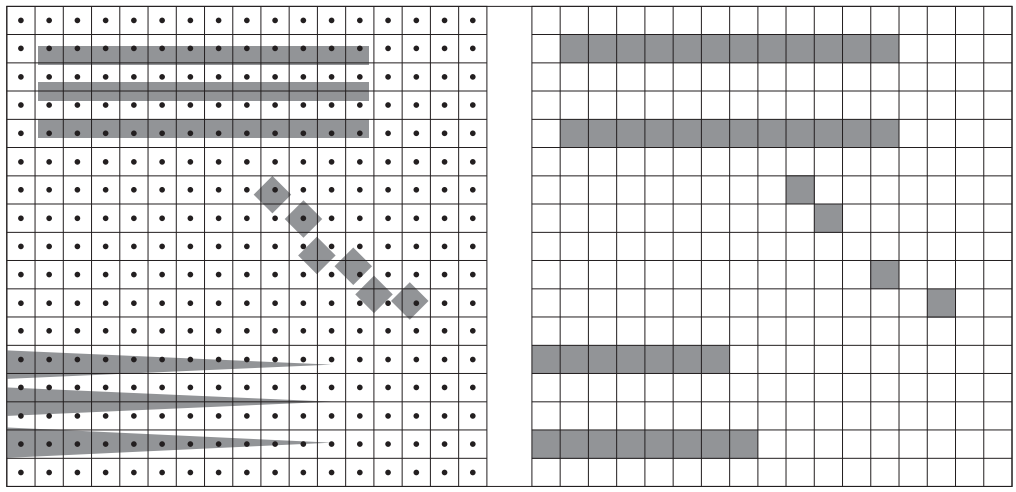


Figura 3.33 Aliasing. *La figura de la izquierda muestra las zonas muestreadas (puntos centrales por píxel). La figura de la derecha muestra cómo parte de la información puede perderse o distorsionarse debido a un muestreo insuficiente.*

b) Desenfocar los bordes de los objetos o líneas internas para disimular el efecto de escalonamiento y de discontinuidad con algoritmos específicos. Este recurso puede disimular este defecto pero no sirve para el caso de objetos que desaparecen por ser demasiado pequeños. Y por añadidura, hace que la imagen pierda calidad de contraste. Desenfocar los bordes es una técnica que está incorporada al procesamiento básico de las imágenes y que se puede incrementar mediante filtros de desenfoque. El inconveniente obvio de este recurso es que se eliminan los defectos a costa de perder precisión en los detalles.

c) Supermuestrear la imagen, es decir, hacer que la muestra represente, no un punto, sino un área en torno al punto, y en lugar de computar un valor único, computar una media ponderada del área que rodea al punto. Esta técnica es superior a las anteriores y permite introducir diferentes variantes según los casos. Sin embargo, introduce una carga adicional en la computación de la imagen y, aunque reduce el *aliasing*, no lo elimina. Pese a todo, es la técnica principal que se describe en el apartado siguiente.

Supermuestreo. Submuestreo

El ***supermuestreo*** (*supersampling*), consiste en tomar más muestras de las correspondientes a la resolución, es decir, tomar, virtualmente, más de una muestra por píxel y luego ponderar el resultado para asignar el valor adecuado a cada píxel. Esto implica dos series de cuestiones: la primera es cómo distribuir las muestras, la segunda, cómo ponderar adecuadamente los resultados.

La distribución de las muestras puede hacerse de varios modos. Una de las primeras soluciones para aumentar el muestreo, pero sin recargar excesivamente el cómputo (debida a Whitted, hacia 1985, en el contexto de nuevas técnicas de *ray tracing*) fue tomar muestras en las esquinas de los píxeles además de los centros. Otra solución es subdividir sistemáticamente los píxeles en 4, 16, 64, etc. Y una tercera solución es utilizar distribuciones aleatorias que reducen los defectos causados por tramas regulares. Estas dos últimas son las más utilizadas en la actualidad.

La ponderación de las muestras puede igualmente hacerse de múltiples modos. En general, dependerá de si envolvemos los va-

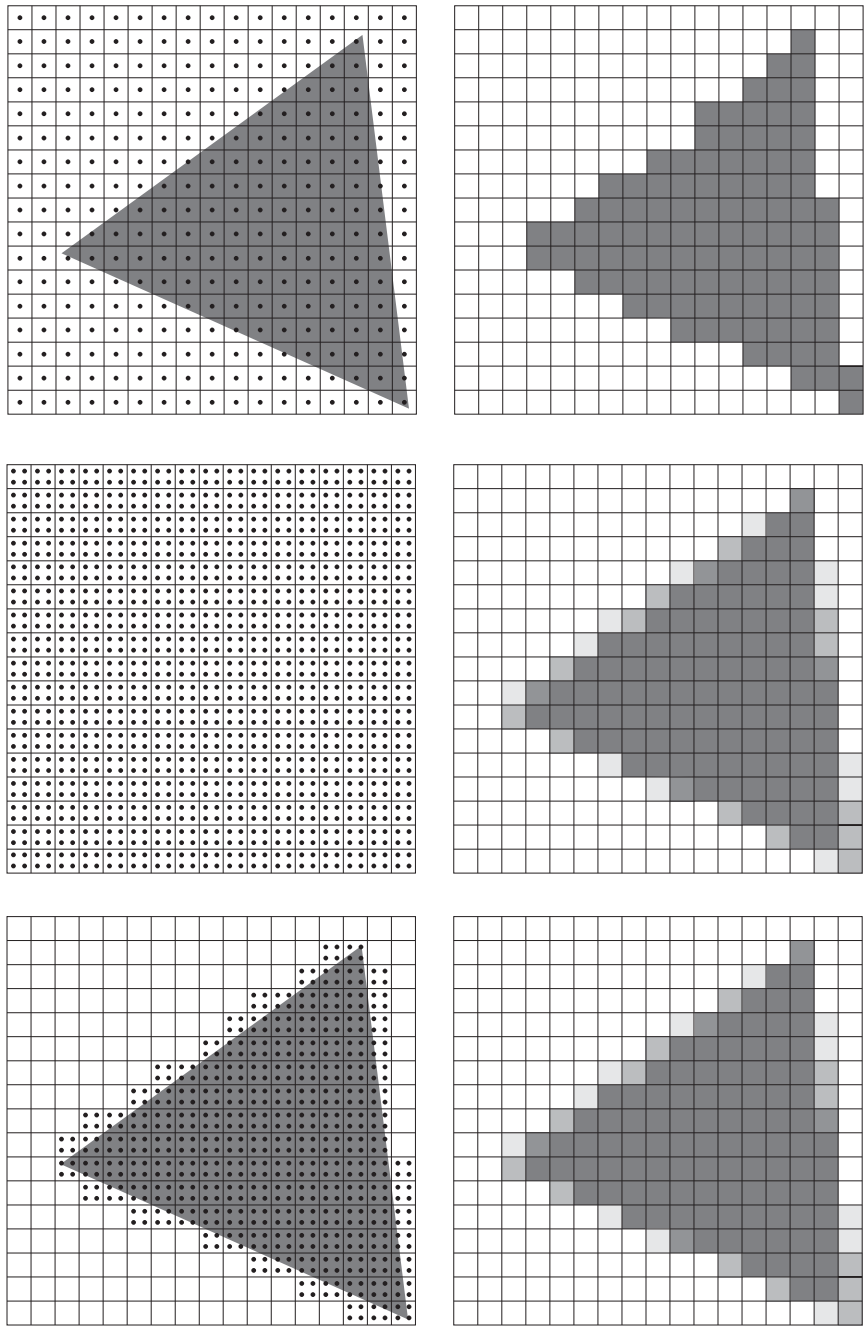


Figura 3.34 Aliasing. *Las figuras de la izquierda representan las muestras como puntos centrados sobre el píxel de salida. Las de la derecha la salida:* a) *Una muestra por píxel,* b) *Cuatro muestras por píxel y ponderación de la intensidad, c) Cuatro muestras adaptativas por píxel.*

lores de un modo simple, como un prisma, fácil de aplicar pero de resultados toscos, o con una curva adecuada que dé mayor importancia a las regiones centrales y menos a las periféricas. En el apartado sobre filtros veremos algunos ejemplos de este tipo de curvas.

Las figuras agrupadas en la 3.34 muestran algunas soluciones características. Una sola muestra por píxel produce bordes escalonados que pueden ser más o menos perceptibles según la distancia, y que en la primera de las tres figuras, se muestran a muy corta distancia para que se aprecie con claridad el efecto. La figura de la izquierda muestra los puntos muestreados y la de la derecha, el resultado que se obtiene con este muestreo. La segunda figura de este grupo muestra la misma imagen pero utilizando una forma mínima de supermuestreo, con 4 muestras por píxel en lugar de 1, como en la figura anterior. Cuando el borde del polígono cae dentro de un píxel, coincidirá con 1, 2, 3 o 4 submuestras. En función de este número de coincidencias se asigna una intensidad al borde, que es una media ponderada de las intensidades de las áreas adyacentes. Si la coincidencia es de 1, el píxel se representa con una intensidad cercana al fondo, si es 3, con una intensidad cercana al polígono, y si es 2, con una intensidad media entre los dos. La tercera figura de este grupo muestra la imagen pero utilizando muestreo adaptativo. En este caso, como puede apreciarse en la figura de la izquierda, solo se toman muestras de las zonas críticas, los bordes del triángulo en este caso.

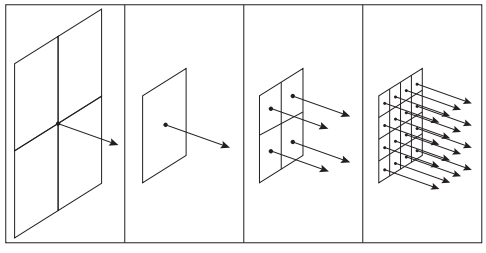


Figura 3.35 Supermuestreo y submuestreo: a) *1/4 (1 rayo cada 4 píxeles),* b) *1 (1 rayo por 1 píxel),* c) *4 (4 rayos por píxel),* d) *16 (4×4 rayos por píxel).*

En la actualidad, la aplicación de técnicas básicas de *supersampling* corre a cargo principalmente de las tarjetas gráficas que incorporan, por lo general, dos técnicas principales: *supersampling* 4x y *multisampling*. La primera técnica consiste sencillamente en hacer un *render* interno al doble de resolución línea (lo que supone 4 veces la resolución total) y luego hacer un *render* externo al tamaño original suavizando las transiciones. Es decir, que si nuestro monitor tiene, por ejemplo, una resolución de 1.440 x 900 (1,29 mp), se hace un render interno a 2.880 x 1.800 (5,18 mp) y se remuestrea de nuevo hacia abajo a la resolución original. El resultado es mejor, pero el coste en computación es obvio.

La segunda técnica, que surgió para mejorar la anterior, es el *multisampling* (MSAA, *MultiSampling AntiAliasing*), que utiliza dos variantes. La primera se basa en suavizar exclusivamente los bordes de los objetos. Se hace un *render* sin ningún tipo de *antialiasing* pero se recomputan las muestras correspondientes a los contornos de los objetos. Para esto, se hace una prueba de detección de aristas (denominada Z-Text) para localizar todas las aristas críticas. La segunda se basa en utilizar los diferentes *shaders* y memorias incorporados a la tarjeta gráfica para llevar a cabo algunos cálculos una vez por píxel para elementos tales como los *pixel shaders*, tablas de textura o muestras de color, y remuestrear otros elementos más exigentes como los valores correspondientes a valores de profundidad (*depth buffers*) y *stencil buffers*.

Los dos fabricantes principales de tarjetas han desarrollado técnicas específicas que, en esta misma línea de reducir la carga bruta del supermuestreo, utilizan recursos sofisticados que reducen el tiempo de cálculo. No las voy a describir aquí por razones de espacio y porque nos basta con saber que existen este tipo de técnicas que hacen más eficiente la técnica básica de muestreo. Baste un par de indicaciones: NVIDIA, por ejemplo, ha introducido la CSAA (*Coverage Sampling Anti-Aliasing*) como una características propia de la serie GeForce 8. AMD, la otra gran competidora,

introdujo la CFAA (*Custom Filter Anti-Aliasing*) para la serie ATI Radeon HD 4870 y posteriores. Puede encontrarse información adicional en las páginas web de estas empresas.

El extremo opuesto del supermuestreo es el **submuestreo** (*undersampling* o *infrasampling*). En las fases iniciales de un trabajo, para obtener pruebas con rapidez, se hace lo contrario de lo que se acaba de explicar: tomar menos muestras por píxel. El resultado, obviamente, es una imagen de peor calidad que la que se obtendría en condiciones normales. Pero en este caso lo que interesa no es la calidad sino la rapidez.

Las especificaciones de submuestreo se hacen, por lo general, mediante números negativos. En mental ray, por ejemplo, se puede especificar un mínimo y un máximo. Y el mínimo puede ser de -4, lo que significa que se toma una muestra por cada 4 píxeles. En el apartado siguiente se dan más detalles sobre alternativas básicas de configuración de cálculo.

Otra forma de supermuestreo es el *supermuestreo adaptativo* (*adaptive oversampling*) que ya he mencionado en relación con las figuras anteriores. En este caso, en lugar de tomar un determinado número de submuestras, el número de muestras se adapta a las características de la escena. En las zonas en las que no hay bordes no se toman muestras adicionales. En las zonas en las que se detectan bordes, se aumenta el número de bordes en función del grado de ocupación del píxel por el borde. El sistema comienza por tomar muestras en las intersecciones de la malla superpuesta a los polígonos de la escena. A partir de aquí, se utilizan diferentes sistemas para decidir donde tomar más muestras: un método corriente es comparar los niveles de contraste de puntos adyacentes. En el apartado siguiente también se dan más ejemplos.

Por último, hay que recordar que los programas de simulación visual incluyen también diversos tipos de controles para ajustar el *antialiasing* por software. Estos controles se pueden aplicar a nivel global o a nivel local. Si se aplican a nivel global afectan por igual a toda la escena. Si se aplican a nivel local afectan a los materiales asignados a los objetos. El procedimiento y los recursos son similares aunque deberán consultarse en cada caso.

Filtros

Generalidades

Un filtro se puede definir como cualquier sistema que modifica una señal. En teoría de señales, esto implica encontrar una función y una operación matemática adecuadas que pueda aplicarse automáticamente a la señal para modificarla de un modo característico. Como ya he indicado antes, la operación matemática principal que se utilizan en filtros es la convolución.

Hay dos tipos principales de operaciones de filtrado aplicadas a imágenes digitales: filtrado en el dominio de la frecuencia y filtrado en el dominio espacial.

El filtrado en el dominio de la frecuencia implica tres pasos principales: *a*) Se aplica a la imagen $f(x,y)$ la transformada de Fourier para obtener la nueva función $F(u,v)$; *b*) Se escoge una función adecuada, $H(u,v)$ (un filtro) y este filtro se combina con la transformada por medio de una convolución. De este modo, se obtiene la nueva función $G(u,v) = F(u,v) * H(u,v)$; *c*) Se aplica a este última función una transformada inversa de Fourier para llevar de vuelta el resultado al dominio espacial.

En el dominio de frecuencias hay tres tipos principales de filtros que se aplican corrientemente: filtros de paso bajo (atenúan las frecuencias altas), filtros de paso alto (atenúan las frecuencias bajas) y filtros de paso medio (atenúan ambos extremos). El resultado característico de un filtro de paso bajo es que se suaviza la imagen eliminando defectos debidos a cambios bruscos de intensidad pero a costa de difuminar la imagen. El resultado característico de un filtro de paso alto es que se refuerzan los contrastes debidos a cambios bruscos de intensidad, lo que puede ser útil

para detectar o enfatizar bordes si bien puede tener efectos colaterales indeseados.

En el dominio espacial se aplican operaciones de filtrado a píxeles concretos de la imagen lo que implica definir un entorno, denominado *kernel* o máscara de convolución, en la vecindad del píxel considerado. Como en el caso anterior, la imagen resultante es el resultado de la convolución en el dominio espacial del *kernel* sobre la imagen para obtener otra imagen $G(x,y) = H(x,y) * F(x,y)$.

Además de los filtros de paso bajo o de paso alto, hay múltiples variantes de detección o resalte de bordes por medio de operadores laplacianos o de gradientes direccionales o de contorno. Un programa de manipulación de imágenes, como Photoshop o Gimp, cuenta con docenas de filtros que pueden utilizarse para todo tipo de aplicaciones. Pero aquí nos concentraremos en los que se utilizan para corregir defectos característicos que aparecen tras un cálculo de iluminación avanzada o de procesamiento de texturas complejas.

El problema con que nos encontramos al utilizar filtros automáticos es que la transformada de Fourier opera sobre el conjunto de la imagen, sobre la totalidad de la señal, no permite, por tanto, el análisis local. Tampoco nos da información sobre la localización de las frecuencias. Nos dice que en una determinada imagen hay frecuencias altas pero no dice dónde. Esto es útil si lo que se necesita es una operación global que, por ejemplo, elimine todas las frecuencias excesivamente altas. Pero a menudo lo que se necesita es un sistema que discrimine entre zonas en las que esto es necesario y zonas en las que no.

Una alternativa importante a la transformada de Fourier que ha surgido en relación con este problema es la *wavelet transform* (que se basa en trabajos del matemático húngaro Alfred Haar de 1909; el término fue propuesto en 1984 por Jean Morlet y Alex Grassmann). Lo que hace esta operación es proyectar la señal original sobre una serie de nuevas funciones básicas que se denominan *wavelets*. Estas funciones operan localmente y admiten diferentes resoluciones (diferentes muestreos sobre la señal original). Se ha utilizado extensamente en sistemas de compresión (como jpeg2000) pero también en reconocimiento de bordes o patrones.

Filtros principales utilizados en simulación visual

Un filtro de paso bajo sirve, como decía en el apartado anterior, para recortar frecuencias demasiado altas. Esta operación puede hacerse fácilmente en el dominio de las frecuencias mediante un recurso técnico, una operación que consiste en multiplicar el espectro de la señal por una función pulso (*pulse function*). Esto equivale ni más ni menos que a eliminar todas las frecuencias que queden más allá de determinado límite. Pero también puede hacerse en el dominio espacial utilizando la señal correspondiente al pulso en el dominio espacial. Esta señal es una función especial, de considerable importancia en teoría de señales, la función *senc* ("seno cardinal", *sinc* en la literatura inglesa) definida por $y = \text{senc}(x) = \sin(x) / x$. Esta función puede definirse también como un filtro ideal que elimina todas las frecuencias que estén más allá de un determinado límite, del ancho de banda dado. La figura 3.6 muestra un diagrama de esta función.

Hay una relación especial entre la altura y la anchura de esta función y las que se derivan de ella. Si la frecuencia a la cual queremos recortar la banda es W y la amplitud es A ,puede demostrarse que A = 2W. Esta relación se mantiene en los filtros que se utilizan en la práctica y que pueden considerarse como simplificaciones basadas en la función senc cuyo uso práctico es poco viable. Los filtros más utilizados desde los tiempos de

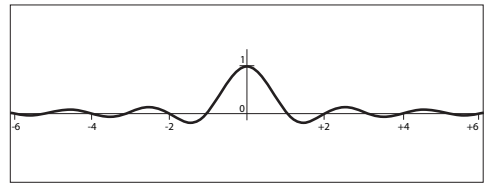


Figura 3.36 La función y = sinc(x).

RenderMan hasta mental ray son los que se derivan de recortes progresivos sobre esta función.

Antes hemos visto un empleo característico de muestreo que se basa en enviar una determinada cantidad de rayos por cada píxel (supermuestreo o *supersampling*) o por cada grupo de píxeles (submuestreo o *undersampling*). Las muestras que se toman para los píxeles se combinan para formar un único valor. Esta combinación puede ser una media simple o ponderada por medio de un tipo determinado de filtro. El tipo de filtro escogido puede tener una influencia relativamente importante en el resultado.

Muchos programas de simulación, entre ellos mental ray solo usan Box, Triangle, Gauss, Mitchell y Lanczos. El tamaño de estos filtros se especifica en píxeles. Si solo se especifica un valor se supone que la altura es igual a la anchura. La lista que sigue los presenta en un orden dado por el tamaño, lo que se corresponde aproximadamente con la velocidad.

Box. Es el filtro utilizado por defecto. Suma todas las muestras del área y las promedia sin ponderarlas. El valor predeterminado suele ser 1,0 (anchura y altura). Es el más simple y más rápido, y el resultado desenfoca el área y suprime los defectos. Pero también puede suprimir los detalles.

Triangle. Pondera las muestras mediante una pirámide centrada en el píxel, lo que equivale a una redistribución lineal. Es algo mejor que Box, las diferencias de velocidad son prácticamente imperceptibles y da resultados suficientes en la mayoría de los casos. El valor predeterminado es 2,0.

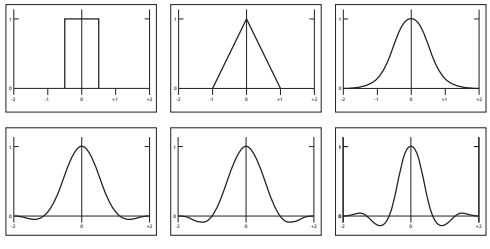


Figura 3.37 Los cinco filtros principales (de izquierda a derecha y de arriba abajo): Box, Triangle, Gauss, Mitchell-Netravali, Lanczos *(dos variantes).*

Gauss. Pondera las muestras mediante una curva de Gauss, lo que hace que las muestras cercanas al centro tengan un peso netamente superior a las situadas en la periferia. Da resultados mejores que el anterior aunque es algo más lento. El valor predeterminado es 3,0.

Lanczos. Pondera las muestras mediante una curva similar a la de la función senc, al igual que el filtro Mitchell-Netravali y disminuye el efecto de las muestras en las aristas del área. El valor predeterminado es 4,0. Proporciona resultados mejores y más definidos aunque puede provocar defectos característicos de anillo (*ringing*).

Mitchell-Netravali. Pondera las muestras mediante una curva que es similar a la función estándar senc pero recortada a partir del segundo *lobe*. El valor predeterminado es 4,0. Valores superiores a los especificados por defecto suavizan el resultado y pueden ser ligeramente más rápidos. Valores inferiores aumentan el contraste y la definición pero pueden producir artefactos. Por debajo de 1,0 pueden eliminar muestras y no son aconsejables. En mental ray tienen dos parámetros (de hecho son 3, pero el tamaño predeterminado, 4,0, no es modificable): *blur* (0,333) y *ringing* (0,333) que permiten equilibrar estos dos efectos. Es el filtro que, en general, da mejores resultados aunque es más lento. Corrige mejor las líneas inclinadas que Lanczos, pero puede desenfocar las texturas.

Por añadidura, hay que mencionar el factor *jittering*, que ya he citado y que introduce variaciones en las muestras locales, desplazándolas aleatoriamente. Esto reduce los artefactos. Para activar *jittering* basta especificar valores superiores a 0,0 cuando este parámetro está disponible.

Además de los anteriores, que son los más recomendables en la actualidad, se pueden citar otros que se encuentran en varios programas. El motor predeterminado de 3ds Max utiliza los siguientes que describo muy brevemente, véase la ayuda si interesa cono-

cer más detalles: *Area* (un filtro genérico muy sencillo que puede servir tanto para agudizar como para suavizar los detalles); *Blackman* (similar al filtro de enfoque de Photoshop pero sin realce en los bordes); *Blend* (combina un filtro de area que aumenta el contraste en los bordes pero sin realzarlos y un filtro gaussiano que suaviza el resultado); *Catmull-Rom* (similar al anterior pero añade un resalte en los bordes, era uno de los filtros más utilizados hasta la aparición de los Mitchell-Netravali y Lanczos); *Cook Variable* (enfoca o desenfoca el resultado en función del valor del parámetro "filter size", entre 1,0 y 2,5 se enfoca y por encima de 2,5 se desenfoca); *Cubic* (un filtro de suavizado que abarca 25 píxeles y utiliza una *spline* cúbica para computar los valores); *Plate Match*/*Max* 2,5 (utilizado desde las primeras versiones de 3ds Max para controlar desajustes entre mapas de fondo y materiales *matte*/*shadow*); *Quadratic* (un filtro de suavizado, que abarca 9 píxeles, y está basado en una spline cuadrática); *Sharp Quadratic* (un filtro de enfoque que abarca 9 píxeles); *Soften* (un filtro gaussiano ajustable de 1,0 a 20,0 que proporciona desenfoques medios); *Video* (un filtro de desenfoque de 25 píxeles adaptado especialmente para vídeo NTSC y PAL).

Además de los citados, pueden encontrarse diversos filtros adicionales para materiales problemáticos, por lo general materiales que incluyen alguna forma de *ray tracing*. Estos materiales ya suelen incluir sus propios controles de *antialiasing* pero este tipo de filtros se suman a estos recursos propios. Lo corriente es utilizar este recurso cuando se detectan artefactos en el *render*, en casos tales como realces especulares suaves o mapas de relieve (*bump mapping*).

Algunos de los filtros de supermuestreo que pueden encontrarse son los siguientes: *Adaptive Halton*: el número de muestras depende del parámetro "Quality" (de 0,0 a 1,0, con 0,5 por defecto) y puede variar entre 4 y 40 muestras. La muestra en el centro de un píxel se promedia con los 4 píxeles que le rodean. El patrón para las muestras es aleatorio. *Adaptive Uniform*: el número de muestras se distribuye regularmente, desde un mínimo de 4 hasta un máximo de 36. El patrón se distorsiona ligeramente para mejorar la precisión en horizontal y vertical. *Hammersley*: las muestras se distribuyen regularmente en la dirección X y de modo cuasi aleatorio en la dirección Y, siguiendo un patrón que depende del parámetro "Quality" (de 0,0 a 1,0, con 0,5 por defecto) y puede variar entre 4 y 40 muestras.

3.5 Organización de la escena

Aunque la finalidad de este libro es la descripción de las técnicas de simulación de materiales, esto no puede separarse con facilidad de otras técnicas generales de simulación, como ya he dicho varias veces. En esta sección resumo los aspectos básicos que hay que tener en cuenta para crear un escenario virtual.

Cámaras

Cámaras básicas

Las cámaras virtuales de los programas más o menos corrientes de simulación visual son cámaras ideales que proyectan la escena sobre un plano virtual y prescinden de otras consideraciones. Esto quiere decir que las únicas propiedades iniciales que nos importan son, prácticamente, el campo de visión y el formato de salida.

En la mayoría de los programas, las cámaras se crean como un objeto más, que se sitúa en la escena y apunta en una dirección. Una vez que se ha creado una cámara puede configurarse un visor para que muestre una "vista de cámara".

Lo que muestra el visor depende, por tanto, de la posición y dirección de la cámara pero también de su ángulo de visión y de su relación de aspecto. Lo primero se controla editando las propiedades de la cámara y modificando este ángulo, que suele denominarse corrientemente FOV por sus siglas en inglés (*field of view*). Lo segundo se controla más frecuentemente desde la configuración de sa-

lida (*render setup*), especificando unas determinadas dimensiones de anchura y altura de la imagen. En la mayoría de los programas de simulación (pero no en todos) también se puede activar algún parámetro especial para que el marco de salida se muestre en el visor de trabajo de tal modo que se pueda componer adecuadamente la escena sin tener que representarla una y otra vez.

Otras propiedades que se pueden encontrar en las cámaras virtuales básicas son: la activación de planos de recorte, la activación de ajustes de profundidad de campo, la activación de ajustes de rangos de entorno o rangos atmosféricos.

Los planos de recorte (*clipping planes*) permiten excluir parte de la geometría de la escena. Hay un plano "cercano" (*near*), que suprime todo lo que hay entre él y la cámara, y otro plano "lejano" (*far*), que suprime todo lo que queda más allá de su ubicación. Dicho de otro modo: solo se representará lo que hay entre estos dos planos. Es un recurso muy útil para ver el interior de una escena sin necesidad de forzar excesivamente el ángulo de visión.

La profundidad de campo o DOF por sus siglas en inglés (*depth of field*) está disponible, según los programas, como recurso propio de una cámara básica, como recurso propio de una cámara física (véase el apartado siguiente) o como recurso independiente que hay que cargar, sea desde la cámara, sea desde la configuración de salida (*render setup*). En cualquier caso, cuenta con unos pocos parámetros relativamente sencillos que permiten especificar la distancia a partir de la cual se desenfocarán los objetos.

Los rangos de entorno (*environment ranges*) están disponibles en algunos programas para ligarlos a efectos atmosféricos tales como la simulación de niebla. De modo similar a los planos de recorte, se especifica una distancia cercana y otra lejana que definen el espesor sobre el que se aplicarán estos efectos, en su caso.

Algunos programas incluyen un operador adicional que permite rectificar las verticales, sea como parte de los controles de la cámara (en V-Ray, por ejemplo, se incluye el parámetro *vertical shift*), sea como recurso independiente que añadir a la cámara (en 3ds Max, por ejemplo, se encuentra en el menú *Modifiers*/*Camera*). Es un recurso importante en arquitectura pues permite simular el procedimiento clásico de los fotógrafos antiguos que utilizaban cámaras de fuelle con las que era posible hacer que la lente y el negativo estuviesen orientados de modo diferente, no paralelo, con lo que se podía invertir la fuga de las verticales y hacer que quedaran captadas sin distorsión. Un recurso que ahora se lleva a cabo por medio de programas de ajuste profesional (como DXO Optics o Photoshop, entre otros).

Por último, en todos los programas de simulación se pueden situar diferentes cámaras e ir pasando de una a otra según lo que interese.

Cámaras físicas. Control de exposición

Para reproducir algunos efectos característicos de la cámaras reales y, sobre todo, para trabajar con sistemas de iluminación avanzada, que utilizan rangos de intensidades muy superiores a los corrientes, lo anterior es insuficiente. Muchos programas de simulación avanzada utilizan el concepto de "cámara física" (*physical camera*) para referirse a cámaras que incluyen ajustes adicionales y que, en general, tienen propiedades similares a las de las cámaras reales. Por ejemplo, en V-Ray nos encontramos con controles para ajustar la exposición por medio de los tres controles clásicos: apertura del diafragma (*f-stop*), velocidad de obturación (*shutter speed*) y velocidad de la película (ISO) y, además, efectos de distorsión, simulación del viñeteado o controles adicionales para el balance de blancos.

En otros programas, como 3ds Max, el control de exposición se ajusta de modo independiente, como un control que se aplica a la salida de la imagen, con independencia de si esta salida se hace a través de una cámara o

no. Pero si se utiliza un control de exposición como el incorporado a mental ray nos encontraremos con parámetros similares para ajustar la exposición.

Algunas cámaras físicas incorporan un efecto adicional denominado *bokeh* en el mundo de la fotografía. Es una palabra japonesa que se puede traducir por "desenfoque" pero implica algo más. Cuando se desenfoca el fondo de una imagen para que resalte mejor el sujeto que está en primer plano, los objetos lejanos aparecen como manchas más o menos circulares. Pero estas manchas pueden tener diferentes formas por razones que están relacionadas con las láminas del diafragma. Una apertura que esté formada por cinco láminas creará formas ligeramente hexagonales mientras que una apertura con más láminas o menos desenfocada creará formas circulares. Estas formas pueden afectar notablemente a la calidad del fondo y muchos fotógrafos las ajustan para que se adapte a lo que les interesa. Algunas cámaras virtuales incoporan la capacidad para crear este efecto y ajustar la forma y la proporción de estas manchas de desenfoque.

El término *aperture* puede ser motivo de confusión pues a veces se utiliza con diferentes sentidos. Su sentido principal es que determina la cantidad de luz que pasa a través de la lente y que, a su vez, se controla por medio de un mecanismo ligado a los *f-stops*. En este sentido, el término estaría referido al diámetro del diafragma medido en *f-stops* y se obtendría dividiendo el valor de la distancia focal por el valor del *f-stop*. Así, una lente de 50 mm de distancia focal, con un valor *f* de 11, tendría una apertura de 4,54 mm (50/11). Y una de 36 mm por el mismo valor *f* tendría una apertura de 3,27 mm. Sin embargo en algún caso se utiliza para referirse a la relación de aspecto de la película utilizada o del formato especificado. En muchos programas y lenguajes (por ejemplo en CG) se refiere a la anchura, es decir, a la dimensión horizontal del formato.

El valor dado por la relación de aspecto de la imagen (*image aspect ratio*) se refiere al cociente entre la dimensión horizontal y vertical. Los dos valores más corrientes en la actualidad son 4:3 (1,333...) y 16:9 (1,777...).

Otro valor que se encuentra en las especificaciones del *render setup* es el *pixel aspect ratio*. Es un valor que está fijado en 1,0 (píxeles cuadrados) y no hay ninguna razón para cambiar. Pero no está de más saber que hay dos formatos en los que los píxeles no son cuadrados: NTSC (*national television system committee*) el formato propio de la TV de Estados Unidos y buena parte de América Latina, que tiene una relación de aspecto de 0,9 y PAL (*phase alternating line*), el formato propio de la TV de la mayoría de los países de Europa, entre otros, que tiene una relación de aspecto de 1,0 para vídeo de 768 x 576 y de 1,06667 para vídeo D-1 de 720 x 576. Esto es algo que conviene recordar en caso de que haya que trabajar con estos formatos.

El control de exposición es importante cuando se trabaja con sistemas de iluminación avanzada, como ya he dicho, pues, en estos casos, el rango dinámico de intensidades replica el real que es miles de veces superior al predeterminado, lo que altera la distribución de intensidades que se basa inicialmente en los ajustes de gama propios de un monitor calibrado para trabajar adecuadamente con programas que utilizan rangos dinámicos simples.

Los problemas y ajustes implicados son independientes de que el control se haga a través de parámetros propios de la cámara o de parámetros propios de un recurso externo. Lo que importa es qué valores de corrección aplicar para que la distribución de intensidades se ajuste a las intensidades utilizadas por las luces avanzadas del sistema o, en su caso, por el uso de archivos en formato HDR. Este es un tema complejo que tiene que ver principalmente con los sistemas de iluminación avanzada y, como en casos anteriores, me remito al libro de simulación de la iluminación en que se trata más amplliamente de estos temas.

Iluminación básica

Organización elemental

La iluminación correcta de una escena y la elección de un sistema de cálculo de iluminación son temas lo suficientemente complejos como para requerir otro libro. Así que me remito de nuevo a mi otro libro sobre simulación de la iluminación que ya he citado varias veces.

Pero se puede simplificar al máximo este tema si todo lo que nos interesa es comprobar que los materiales que hemos preparado se representan correctamente y si nos basta con una simulación básica de una determinada escena. Para esto no se necesita un sistema de iluminación avanzada sino todo lo contrario: bastan dos requisitos simples.

El primer requisito es comenzar por aislar el objeto al que hemos aplicado un material, escoger una representación frontal que lo muestre a toda pantalla e iluminarlo con una única luz uniforme y perpendicular a su cara principal. Si la luz es de tipo directo, su intensidad es la predeterminada (1,0 en unidades convencionales normalizadas), no arroja sombras y abarca todo el objeto, el requisito de uniformidad se cumplirá con facilidad y podremos juzgar adecuadamente si la aplicación del material es correcta. Bastará seguir este procedimiento para todos los materiales de la escena y luego eliminar esta luz auxiliar.

El segundo requisito es preparar una o varias cámaras que cubran adecuadamente la escena y crear una disposición lumínica canónica: una luz primaria y, al menos, una luz secundaria. La luz primaria debe apuntar en la misma dirección que la cámara pero desde su izquierda. Y la luz secundaria debe apuntar en la misma dirección pero desde la derecha. La luz primaria será de tipo simple pero preferentemente extensa para que las sombras sean suaves y con una intensidad de 1,0 o algo menos para compensar la adición de la secundaria (en principio, ambas deberían sumar 1,0 aproximadamente). Una buena elección, si se utiliza 3ds Max es utilizar una luz estándar de tipo *mr area spot*. La luz secundaria será del mismo tipo pero puntual pues sus propiedades de generar sombras deben estar desactivadas y de una intensidad de 0,5 o menos. La finalidad de la luz secundaria es evitar que las sombras queden excesivamente oscuras, simulando el rebote de la luz en los objetos cercanos como ocurriría en la realidad o si utilizáramos sistemas de iluminación avanzada. La figura 3.38 ilustra esta disposición elemental. La luz primaria se ha etiquetado como L1 y la secundaria como L2.

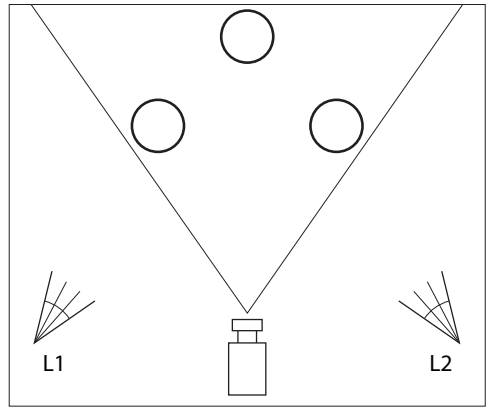


Figura 3.38 Organización básica de la iluminación de una escena con una luz primaria (L1) y una luz secundaria (L2).

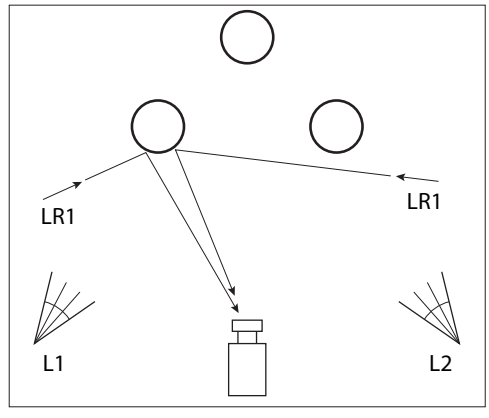


Figura 3.39 Organización básica de la iluminación de una escena con luces complementarias, que no afectan a la intensidad global, para realzar los reflejos.

Si los materiales son reflectantes se pueden utilizar, sobre todo si los objetos a los que se aplican tiene superficies curvas, luces secundarias adicionales para realzar los reflejos. Dado que el uso de varias luces desequilibraría la intensidad global, que habría que reajustar con cuidado, una buena solución, si el programa lo permite, es desactivar la contribución de la luz a la iluminación de la escena y activar tan solo su contribución al realce especular. Por ejemplo, en 3ds Max, si se selecciona una luz y se abre el menú contextual con el botón derecho del ratón, aparece un pequeño menú con opciones para apagar la luz, para desactivar su contribución a la iluminación de las superficies (*affect diffuse*) y para desactivar su contribución a la especularidad (*affect specular*). Bastará, por tanto, con desactivar la primera y dejar la segunda.

Por último, hay que subrayar que para que el resultado final sea óptimo deben utilizarse sistemas de iluminación avanzada. No solo porque la calidad general de la escena mejorará de un modo notorio sino también porque las propiedades de materiales avanzados solo se apreciarán adecuadamente con este tipo de sistemas. En esta fase final, por ejemplo, si se utiliza el material Arch&Design de mental ray (o materiales equivalentes de otros programas) será más que preferible substituir las luces de tipo estándar por luces fotométricas, pues estos materiales avanzados están calibrados para este tipo de luces y su rendimiento y fidelidad será mejor.

Configuración básica de cálculo

Como en el caso anterior y dado que las configuraciones de salida dependen principalmente del sistema de cálculo de iluminación escogido, me remito al libro sobre simulación de la iluminación.

Pero, también como en el caso anterior, si partimos de un sistema sencillo como el descrito en el apartado anterior, la configuración adecuada se puede resumir en ajustar: *a*) la resolución, *b*) el formato de salida, *c*) la elección del motor de render, *d*) la calidad de salida que depende de la configuración de *antialiasing*, *e*) en su caso, otros ajustes más avanzados.

Por lo que respecta a la ***resolución*** lo primero que hay que recordar es que la finalidad más inmediata de un proceso de modelado y simulación visual es obtener una imagen sintética. Pero esta imagen puede tener destinos muy diferentes. Puede ser una prueba para una futura filmación, puede estar destinada a mostrarse en un monitor con una calidad ajustada al tipo de monitor o puede estar destinada a ser impresa sobre papel, con mayor o menor calidad y mayor o menor tamaño. Esto significa que, en cualquier caso, será necesario, además de componer adecuadamente la escena y ajustar la iluminación, especificar una resolución y un formato.

La resolución depende, por tanto, del destino previsto. Si la imagen está destinada a salida por pantalla, sin ampliaciones, una resolución de 72 dpi (unos 28 ppc, puntos por centímetro), para un proyector de tamaño medio, lo que equivale a entre 1.024 y 1.600 píxeles de ancho, será suficiente. Si se prevé que se van a realizar ampliaciones, está resolución deberá multiplicarse por el mismo factor que la ampliación o ampliaciones previstas. Si está destinada a ser impresa sobre papel la resolución debería de ser del orden de los 300 dpi (unos 120 ppc, puntos por centímetro), un valor de referencia que varía con las características de la impresora, del papel, de la calidad y del tamaño, que tiene que ver con la distancia de observación pues si esta va a ser, en cualquier caso, mayor que 25 cm, nos permitirá reducir proporcionalmente la resolución gráfica. Para una estimación correcta es preferible ponerse en contacto con quienes vayan a hacer la impresión o estudiar bien el manual de la impresora que se vaya a utilizar. El factor crítico, sin embargo, es el tamaño previsto. Si, por ejemplo, se va a imprimir a un tamaño de 18 x 24 cm, con una resolución de 120 ppc, el tamaño de la imagen deberá ser de 2.160 x 2.880 píxeles o valores cercanos a estos. Y si el tamaño de salida previsto es tres veces mayor la imagen tendría, en principio, tamaños del orden de 8.000 o 9.000 píxeles de ancho, lo que puede

reducirse considerando que la distancia de observación de referencia para 300 dpi es 25 cm y que tendemos a contemplar un cuadro a una distancia que es más o menos igual a la de su diagonal, aunque el observador podría acercarse a menor distancia para ver algunos detalles.

La palabra **formato** se puede referir a dos cosas: las proporciones de salida (relación de aspecto) o el tipo de archivo (reconocible por su extensión). Por lo que respecta a lo primero, si el destino de la imagen es vídeo, las dimensiones deberán cumplir con las especificaciones del sistema que se vaya a utilizar. En este caso, escoger los formatos predeterminados, si el programa los incluye en una lista, como ocurre en muchos que, por ejemplo, incluyen salidas prefijadas para vídeo NTSC (720 x 480), vídeo PAL (720 x 576) o HDTV (1.920 x 1.080) entre otras. O bien introducir directamente el valor que interese. Si el destino de la imagen es una pantalla de ordenador o una impresión sobre papel, las dimensiones son libres. En este caso, todo lo que hay que hacer es escribir la anchura y altura en píxeles en las casillas correspondientes. Luego convendrá bloquearlo (clic en pequeño candado adjunto) para mantener la relación de aspecto aunque se cambie el tamaño para hacer pruebas.

Hay que tener en cuenta que el formato de salida especificado no tiene porqué coincidir con el que aparece en el visor. Para que este muestre un marco que coincida con el de salida hay que activar las opciones correspondientes si el programa lo incluye. Por ejemplo, en 3ds Max basta con presionar *Shift+F* (o buscar las opciones de menú de visor correspondientes). Por lo que respecta a lo segundo, el tipo de archivo, la elección depende sobre todo del tipo de salida previsto. Si la salida será por pantalla, un formato jpg con calidad media será suficiente. Pero el formato jpg, un formato de compresión con pérdida, no es adecuado para imprimir pues aparecerán errores, sobre todo en zonas con degradados suaves, que no son visibles en pantalla debido al efecto de irradiación. Por esto, si se va a imprimir es preferible utilizar un formato de comprensión sin pérdida como tif o tga.

Por lo que respecta a la elección del ***motor de render*** lo mejor será, para no meterse en complicaciones, dejar el predeterminado por el programa que utilicemos. De nuevo me remito al libro sobre simulación de iluminación donde se trata este tema ampliamente. Baste con recordar que los programas más utilizados incluyen múltiples opciones cada una de las cuales implica un sistema de cálculo de iluminación diferente además de otros aspectos que también afectan a la configuración de salida. Por ejemplo, 3ds Max incluye, en sus últimas versiones, Default scanline, el propio de 3ds Max que incorpora dos subsistemas de iluminación avanzada, *light tracer* o *radiosity* (que en 2014 ya no son muy “avanzadas”); mental ray, que utiliza *final gather*, una variante propia de irradiancia; iRay (también ligado a mental ray) que utiliza una variante propia de *path tracing*; Quick Silver, un motor de render basado en hardware que da resultados menos exactos pero muy rápidos, utilizando una combinación de la CPU (que procesa los datos básicos de la escena y compila los *shaders*) y de la GPU, que recibe los datos y *shaders* compilados por la anterior y genera la imagen utilizando recursos similares a los utilizados en videojuegos. Es necesario contar con una tarjeta gráfica adecuada que soporte como mínimo la versión SM3 (Shader Model 3.0). También incluye Vue, un motor de render que genera un archivo de tipo .vue, en formato ASCII, que puede abrirse y editarse con el bloc de notas y que se usaba para transferencia de archivos entre aplicaciones. O bien, si se utiliza V-Ray se puede elegir entre diferentes combinaciones de cuatro sistemas: *irradiance*, *photon mapping*, *light cache* y *QMC* (o *DMC* o *brute force* en las últimas versiones).

La calidad de salida se ajusta con una serie de parámetros que dependen en gran medida del motor utilizado. Pero en todos ellos habrá un sistema de *antialiasing* que incluirá parámetros claramente relacionados con los que he descrito, con carácter general, en la sección correspondiente. En muchos casos se incluyen etiquetas tales como calidad “muy baja”, “baja”, “media” o “alta”, y escogiendo las pri-

meras para pruebas y las últimas o penúltimas para resultados finales no tendremos que preocuparnos demasiado de ajustar parámetros concretos. Pero los usuarios exigentes preferirán hacer ajustes más finos dentro de alguna de estas opciones, por lo que conviene conocer el sentido de estos parámetros.

Por último, hay que tener en cuenta que hay varios parámetros y configuraciones que afectan también a la velocidad de cálculo. Los programas de *rendering* utilizan dos modos de procesamiento: por software y por hardware. El procesamiento por *software* se basa de modo principal en dos tipos de algoritmos principales: *scanline* y *raytrace*. El procesamiento por hardware se basa de modo principal en los recursos propios de OpenGL y Direct3D integrados en la GPU de la tarjeta gráfica del ordenador, que se basan en la resolución especificada y procesa la escena triángulo por triángulo pero a una velocidad enorme.

El procesamiento por software evalúa los valores de los píxeles a partir de muestras que se toman de los dos modos descritos. Si se utiliza un software como mental ray (sea a través de Maya, XSI Soft Image o 3ds Max), el programa utiliza solo hasta cierto punto los recursos de la GPU debido a que su sistema de muestreo depende, en alto grado, de un sistema de cálculo sofisticado propio que evalúa el color final de un píxel a partir de múltiples muestras de la escena, que se ponderan y se interpolan de diferentes modos. Y a los que, por añadidura, se aplican diferentes sistemas de filtrado para corregir errores e integrar los resultados con los de los píxeles adyacentes.

Con mental ray se utilizan tres algoritmos de cálculo: *a*) *scanline*. Este método, que se describe más adelante, se aplica a través de tres tipos de algoritmos: el propio del motor predeterminado por la aplicación anfitriona (Maya, XSI, Max); el *rasterizer* (un método más rápido y preciso, en determinados casos, que el estándar, que no se basa exclusivamente en los datos derivados de las coordenadas de vista de cámara); algoritmos propios de OpenGL (basados en *software* y con apoyo específico de *hardware* en ciertos casos) que están disponibles en Maya pero no en XSI ni en 3ds Max; *b*) *ray tracing*. También está basado en software, está ligado de modo más característico a los modos de cálculo de iluminación de mental ray y se describe ampliamente en los capítulos correspondientes del libro de iluminación; *c*) *Hardware rendering*. A la inversa de los anteriores, se apoya principalmente en la GPU con apoyo de software.

Estos tres modos se pueden utilizar de modo independiente o combinados entre sí. Así, en 3ds Max, se pueden activar o desactivar los dos primeros desde *Render Setup* / *Renderer*. Algunos cálculos básicos, como el sombreado elemental de superficies, se pueden remitir al hardware mientras que los cálculos más complejos y característicos se harán por *ray trace*. Gran parte de la decisión sobre si utilizar uno u otro sistema recae sobre el usuario. En algunos casos la decisión es obligada. Por ejemplo, si se utilizan planos de recorte para que una vista de cámara pueda ver el interior de una escena desde el exterior, en 3ds Max, será necesario activar *scanline*, pues en caso contrario no se aplicarán adecuadamente estos recortes y el resultado será opaco. Por otro lado, para llevar a cabo ciertos cálculos, como la reflexión de un material, el sistema activará automáticamente los algoritmos de *ray trace*. Los primeros rayos se trazan, por lo general, por el algoritmo de *scanline* mientras que los secundarios requieren *ray tracing*. Con *scanline*, por otro lado, se utilizan todos los recursos de hardware posibles para cargar datos en el *frame buffer*. Si se necesitan cálculos de *ray trace* se activan por software y los resultados se pasan a la GPU que, a su vez, los pasa de nuevo al *frame buffer* donde se combinan con los ya existentes. Un ejemplo característicos son los valores dados por cálculos de reflexión por *ray trace* que se combinan con los datos de color local computados previamente.

Con *scanline* solo se computan los polígonos que están situados directamente frente a la línea de la vista. Ordena los triángulos proyectados sobre las coordenadas *xy* de vista de cámara y computa las coordenadas *z* correspondientes, descartando aquellos triángu-

los cuya coordenada *z* sea mayor que otro de iguales coordenadas *xy*. Por tanto, cualquier polígono que quede oculto por otros o que esté fuera de la pirámide de visión no se tiene en cuenta, se borra de la memoria inmediata (*cache memory*).

Con *ray tracing*, por el contrario, todos los polígonos que los rayos encuentran al desviarse se computan, aunque estén ocultos por otros polígonos o situados detrás de la cámara o fuera del rango especificado por los planos de recorte.

Con *hardware*, por otro lado, nos encontraremos con algunas limitaciones importantes pese a su velocidad de proceso. Depende exclusivamente de la resolución y no puede llevar a cabo *supersampling*, muestreos a nivel de subpíxel. Tampoco puede procesar *shaders* complejos, que dependen de algoritmos no incorporados a la GPU. Sin embargo, los beneficios en términos de velocidad son considerables y se van incrementando con el tiempo. Un concepto importante que se debe tener presente es el de *layering*. Este término se refiere al modo en que ciertos elementos se procesan por la GPU, ejecutando ciertas tareas en el mejor orden posible (mapas de sombras, de fotones, de irradiancia, de texturas, de entorno, etc.) y guardándolas en *buffers* especializados, que luego se combinan como capas superpuestas que combinan sus valores con los derivados de otros procesos para llegar a obtener un determinado valor para un píxel. Con mental ray solo está disponible en Maya.

Un último aspecto de interés tiene que ver con los *buckets*. Cuando se utiliza mental ray como motor de *render*, el proceso se lleva a cabo, no línea a línea, como con *scanline*, sino a través de paquetes rectangulares, denominados *buckets* ("cubos", "cestas"), que van apareciendo en la pantalla en un orden particular y un determinado tamaño. El orden se puede modificar: en 3ds Max las alternativas son *Hilbert*, *spiral*, *left to right*, *right to left*, *top down* y *bottom up*. El más eficiente es el primero, *Hilbert*, que utiliza la base de datos de la escena para encontrar el orden más lógico y más rápido de procesamiento pero, en determinados casos, nos puede interesar alguna de las otras opciones para que aparezcan antes las zonas que nos interesa y cancelar el proceso si el resultado no es correcto. Por lo que respecta al tamaño, si este es más pequeño hay más regeneraciones de la imagen, lo que consume un pequeño número de ciclos de CPU. Esto puede afectar ligeramente al tiempo de cálculo, algo que es inapreciable en imágenes pequeñas pero que se puede notar en imágenes grandes. En estos casos puede convenir aumentar el tamaño del *bucket* a algo así como 1/10 de la dimensión mayor. Por ejemplo, para una imagen de 4.000 x 3.000 sería preferible que el tamaño del bucket fuese de 400 píxeles.

Hay otros ajustes de algoritmos que también pueden afectar a la velocidad de cálculo. Con *ray trace*, el algoritmo funciona comprobando, en primer lugar, si cada rayo trazador encuentra algún triángulo en su recorrido lo que, en principio, obliga a computar la posición de todos los triángulos de la escena. Este cómputo puede reducirse notablemente si se descartan los triángulos que no pueden estar en la ruta de los rayos por medio de otro cómputo previo. Este cómputo previo se lleva a cabo, con mental ray, por medio de dos algoritmos específicos: *hierarchical grid* y *binary space partition* (BSP). El más utilizado es el segundo que incluye una variante, *large BSP*. Lo que hacen estos algoritmos es dividir la escena en regiones o "contenedores" o *voxels* (*volume elements*), de tal modo que puedan descartarse los contenedores que no están en la ruta del rayo, lo que permite descartar los triángulos que contiene. La diferencia entre los algoritmos se deriva del modo particular en que subdividen y organizan la escena. BSP utiliza una subdivisión adaptativa que acelera el análisis de las ramas para comprobar si hay elementos en la ruta del rayo. El parámetro principal es la máxima subdivisión que se puede alcanzar para no recargar excesivamente la memoria. Otro parámetro importante es el máximo número de triángulos que un *voxel* puede almacenar. Hay diversas técnicas que optimizan de diferentes modos estos algoritmos. Estos dos parámetros están disponibles,

en 3ds Max, en el grupo *raytrace acceleration* si se escoge como método BSP. El parámetro *size* afecta al tamaño del contenedor y el parámetro *depth*, a la profundidad de la subdivisión. Si se aumenta el tamaño aumenta el tiempo de cálculo pero se reduce la memoria porque se necesitan menos *voxels*. Y el efecto inverso se da si se aumenta el valor de *depth* pero se reduce el de *size*. Son parámetros que merecerá la pena ajustar principalmente en el caso de animaciones. Pueden analizarse los resultados por medio de herramientas de diagnóstico, que permitirán comprobar la memoria y el tiempo de cálculo. En 3ds Max estas herramientas se pueden encontrar en *Diagnostics* (en *Render setup* / *Processing*) activando la opción correspondiente a BSP y escogiendo *size* o *depth* de la lista desplegable adjunta. Los resultados se mostrarán gráficamente al hacer un *render*, como colores codificados superpuestos a la imagen: azul (mínimo), verde, amarillo, naranja y rojo (áreas que han llegado al máximo especificado).

Si se escoge BSP2 no hay parámetros, el proceso es automático. Este algoritmo está optimizado para escenas muy grandes, con más de un millón de triángulos. Requiere menos memoria que BSP, pero puede resultar más lento que éste para escenas relativamente pequeñas.

3.6 Organización del proyecto

La organización adecuada de los materiales y mapas de un proyecto y la reorganización y gestion de esta organización previa son una parte fundamental, a menudo descuidada (lo que da lugar a pérdidas de tiempo en el mejor de los casos y problemas graves en otros casos), de todo lo que tiene que ver con la simulación de materiales.

Organización general

Pasos previos

Para desarrollar un proyecto de simulación de materiales, sea antes o después de haber completado el modelo geométrico, se necesita hacer lo siguiente:

1 Analizar los materiales del modelo y decidir el método de simulación más conveniente.
2 Preparar texturas adecuadas para los materiales que lo necesiten, siguiendo las recomendaciones que se dan más adelante, en el capítulo sobre este tema.
3 Copiar todas las texturas requeridas para el proyecto en una carpeta concreta. Aunque se produzcan repeticiones, es preferible crear y mantener una carpeta con las texturas que se piensa utilizar en el proyecto. De este modo no hay que cambiar la ruta de acceso. Y, para hacer copias de seguridad,

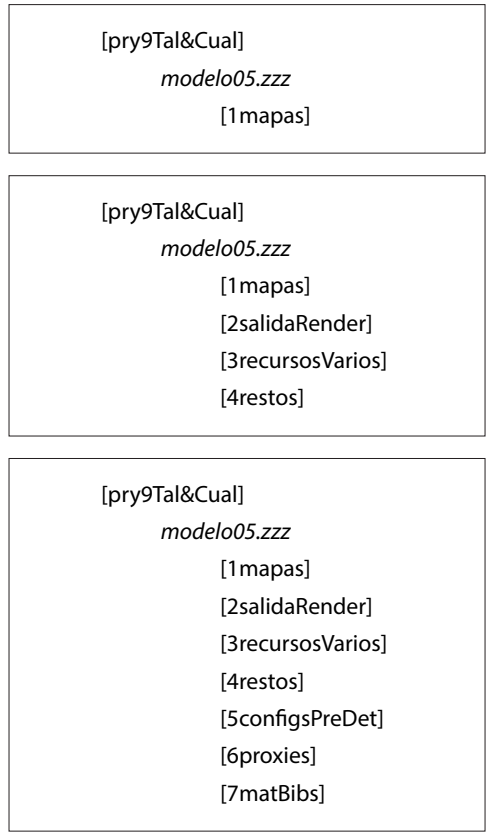


Figura 3.40 Organización del proyecto: a) estructura mínima, b) estructura media, c) estructura avanzada.

bastará con copiar la carpeta de proyecto con todo su contenido, incluyendo los mapas utilizados.

4 Organizar el proyecto de un modo similar a lo que se muestra en los esquemas de la figura 3.40.

El primer esquema muestra la estructura mínima: el nombre del proyecto en el nivel superior, el archivo del modelo y una carpeta con los mapas.

El segundo esquema incluye tres carpetas más: "salidaRender" para guardar las imágenes de prueba o definitivas; "recursosVarios", que puede incluir archivos con componentes del proyecto o scripts, o cualquier otra cosa que pueda ser útil para el desarrollo de la simulación; "restos", que incluirá archivos que se van desechando pero que conviene guardar en lugar de eliminarlos. Por ejemplo, como se puede apreciar en la imagen, el modelo lleva un sufijo, un número ("05"), lo que indica que es la quinta versión y hay cuatro modelos previos. Conviene guardar estos modelos antes de terminar pues a menudo hay arrepentimientos y llegamos a la conclusión de que algo que hicimos en una versión anterior era preferible. Otro tanto ocurre con los mapas: puede ser que hayamos corregido un mapa pero nos interese guardar una versión anterior, por si acaso.

El tercer esquema incluye otras tres carpetas: "configsPreDet" para guardar configuraciones de cálculo predeterminadas (si el programa lo permite, como ocurre en la mayoría), lo que nos ahorrará tener que modificar varios parámetros cada vez que queremos pasar de, por ejemplo, una configuración de salida de baja resolución y con *antialiasing* de poca calidad para hacer comprobaciones elementales a otra configuración superior; "proxies", para guardar las imágenes de baja resolución asociadas a materiales y objetos que utilicen este tipo de recurso, que se explicará con más detalle en los siguientes capítulos; "matBibs" para guardar los materiales del proyecto como copia de seguridad, tal como se ha explicado antes.

Puede haber unas cuantas más. Por ejemplo, si se utiliza 3ds Max y se establece una nueva ubicación formal para el proyecto (véase la ayuda del programa para más explicaciones), el programa creará automáticamente una carpeta para el proyecto con una docena de subcarpetas (muchas de las cuales no se utilizarán sino en casos muy especiales).

Asignaciones. Nomenclatura

La asignación de un material es muy simple e implica unos pocos clics. Los pasos que hay que dar son: *a*) Definir el material desde el editor de materiales siguiendo los procedimientos que veremos en los capítulos siguientes; *b*) Dar un nombre al nuevo material siguiendo los criterios que se dan a continuación; *c*) Seleccionar el objeto de la escena al que se quiere asignar este material y asignarlo siguiendo el procedimiento del programa (presionar un botón, arrastar sobre el objeto, activar un comando específico, etc.).

Así, cada vez que se crea un material hay que bautizarlo (y, si no, el programa lo bautizará por nosotros con un nombre tal como "material1, "material2", etc.). Y dar nombres adecuados a todos los materiales de una escena compleja no es una operación trivial. La mayoría de los que comienzan a trabajar en un programa de simulación pierden mucho tiempo, *a posteriori*, por ahorrar unos pocos segundos, dejando que el programa nombre las cosas como le parece.

¿Qué criterio seguir para la nomenclatura? Esto depende de los gustos de cada cual y de los casos, pero hay algunas consideraciones bastante obvias que pueden orientar estos criterios. Si designamos un material con un nombre tal como "maderaRoble" nos costará recordar a qué elemento se ha asignado (¿un pavimento? ¿una puerta?). Y si designamos un material con un nombre tal como "pavimento" también nos puede costar recordar a cuál se refiere si hay varios pavimentos con materiales distintos. Y, por otro lado, nos puede interesar almacenar variantes para hacer pruebas.

Un modo seguro de saber a que atenerse es nombrar los materiales con el mismo nombre que los objetos a los que se van a asignar y, si es necesario, con números que indiquen variantes y, si también se considera necesario, con nombres de materiales que precisen el tipo de variante. Así, si la escena contiene objetos tales como "muroExterno" o "PavimentoSala" (pavimento correspondiente a la sala principal), los materiales correspondientes pueden recibir el mismo nombre o nombres como "muroExterno1" o bien "muroExternoLdr1" (ladrillo, versión 1), "PavimentoSala_marmolBlanco", etc.

Gestión

Control de las referencias externas

Es posible que nos hayamos pasado mucho tiempo definiendo y ajustando la definición de un material y, en un instante de descuido o por un error del programa o del ordenador, el material se pierda. Para evitar esto, se debe tener presente lo que sigue:

1 En muchos programas, entre ellos 3ds Max, los materiales se guardan en 3 lugares: *a*) en el editor de materiales, *b*) en la escena, *c*) en una biblioteca externa.
   Los dos primeros se guardan en el archivo, junto con el modelo. El tercero es independiente y se guarda en otro formato (.mat en 3ds Max), que puede ser abierto desde cualquier otro archivo de modelo.
2 En el primer sitio, el editor de materiales, hay un límite. En el editor clásico de 3ds Max, solo caben 24.
3 En el segundo sitio, la escena, caben tantos como objetos pueda contener la escena. Pero pueden modificarse con peligrosa facilidad. Si, por ejemplo, se asigna por descuido un material a un objeto seleccionado, la asignación anterior se pierde.
4 Lo más seguro, por tanto, es el tercer lugar: una biblioteca externa. Lo que quiere decir que es muy recomendable, a medida que el proyecto avance, grabar los materiales que estamos utilizando en una biblioteca que debe guardarse en la misma carpeta en que esté situado el modelo principal. Los métodos para hacerlo son sencillos y se resumen en el apartado siguiente.

También es posible que un material no desaparezca pero quede cojo porque el mapa que forma parte de su definición no se encuentra. Esto pasa habitualmente cuando se cambia de ordenador y se traduce en un mensaje también bastante habitual al hacer una representación: "Archivos externos ausentes".

Esto es debido a que la "propiedad" de definición de mapa que se incluye en la definición de un mapa es fundamentalmente una dirección. Y sí, por ejemplo, hemos trabajado en un ordenador en donde nuestro mapa estaba en la dirección "C:/…/Proyecto33/…/zzz.jpg" y nos hemos cambiado a un ordenador en donde está en "C:/…/MisProyectos/…/zzz.jpg", aunque el nombre final "zzz.jpg" coincida el resto no coincide y por consiguiente el nombre completo no cuadra y el programa no puede encontrar un archivo con ese nombre.

Esto se puede solucionar reasignando los nombres del mapa, lo que lleva menos de un minuto. Pero si hay que hacerlo con muchos mapas son muchos minutos; y si hay que hacerlo muchas veces pueden ser horas.

Hay recursos que ayudan a solucionar con facilidad estos problemas. Por ejemplo, se puede indicar al programa que busque todos los archivos en una ubicación determinada. Pero la mejor recomendación es guardarlo en una carpeta adjunta al archivo del modelo. La mayoría de los programas buscan cualquier mapa asociado a los materiales del modelo en esta ubicación.

También hay programas que incorporan los mapas al archivo con lo que este problema desaparece. Por ejemplo, en SketchUp, no hay que preocuparse por este asunto pues los mapas se guardan con el archivo. Pero la sencillez tiene un coste y aunque a muchos usuarios no les preocupe demasiado, resulta que es bastante habitual que un archivo de Sketchup ocupe 50 o 100 Mb cuando un ar-

chivo de 3ds Max, con las mismas características ocupa menos de 1 Mb.

Bibliotecas de materiales

En todos los programas de simulación, una biblioteca (*library*) de materiales es, en realidad, una base de datos (si, por ejemplo, se hace un doble clic sobre un archivo tipo *xxx.mat*, tal como los que se generan en 3ds Max, en un ordenador que tenga el programa de gestión de base de datos MS Access, se abre este programa aunque no se puede acceder a la base). En ella se guardan las propiedades de los materiales, que consisten básicamente en un nombre propio, una serie de valores relacionados con parámetros y la dirección en que están los mapas de bits, en el caso de que el material los incluya.

Debido a que los materiales se guardan con el propio archivo de 3ds Max y que son fáciles de crear, la mayoría de los usuarios no se preocupa demasiado por crear y guardar bibliotecas de materiales, sino más bien de su principal y más valioso componente, las bibliotecas de mapas de bits a las que se refiere el apartado siguiente. Pero, como he dicho en el apartado anterior, es aconsejable contar, al menos, con una biblioteca de materiales por proyecto dado que ocupa poco espacio y puede ahorrar redefiniciones innecesarias. Esta es la recomendación principal. Lo que sigue tiene un interés relativo.

Si se quiere crear una nueva biblioteca de materiales hay varios procedimientos aparte del ya mencionado, crear una biblioteca por cada proyecto. Puede también crearse una biblioteca genérica que sirva como base para otros proyectos. En este caso, se puede utilizar el mismo procedimiento, pero en lugar de guardar la biblioteca en la carpeta del proyecto activo, guardarla en una ubicación general e ir introduciendo materiales genéricos. O bien copiar la biblioteca básica del programa, borrar los materiales que no interesen y guardar o modificar los que interesen.

Bibliotecas de mapas

Como ya hemos visto, la definición del material no incluye los mapas de bits asociados al material sino solo su dirección.

La gestión de mapas es la tarea principal de la organización de una biblioteca de materiales. Definir un material es una tarea relativamente sencilla pero contar con una buena colección de texturas no es algo que se pueda improvisar. Hasta tal punto es así que todo profesional que trabaje en simulación cuenta con buenas bibliotecas de mapas, pero puede que no utilice bibliotecas de materiales excepto en casos especiales o como medida de seguridad en ciertos proyectos.

En el capítulo sobre técnicas de aplicación de mapas se amplía este tema. Pero podemos avanzar que aunque se cuente con una amplia colección de imágenes, lo más probable es que, si uno es exigente, no se encuentre la que estamos buscando y haya que crearla a partir de fotografías, retoques y ajustes con Photoshop.

Debe tenerse en cuenta el modo de almacenamiento, el tipo de ordenación, la ubicación, para que sean accesibles desde el programa, la facilidad de acceso para comprobar su contenido. Los criterios de ordenación en el caso de la arquitectura pueden ser algo así como: piedras, maderas, mármoles, fondos, metales, muros, pavimentos, etc. Puede interesar también almacenar diferentes resoluciones. En la duda, optar por la más grande o guardar su referencia (foto, etc.). La configuración puede ser permanente o variar para cada proyecto. En el caso de aplicaciones para arquitectura será necesario guardar imágenes de paramentos de grandes dimensiones antes que piezas aisladas.

Un criterio recomendable es el siguiente:

a) Crear una biblioteca general bien organizada.

b) En cada proyecto, copiar los mapas que interesen de la anterior y crear nuevos mapas según las necesidades del proyecto. Y, al terminar el proyecto, copiar los nuevos mapas a la biblioteca general que, de este modo, se irá enriqueciendo con cada proyecto.

→4

Técnicas básicas

Nota sobre técnicas y software

En este capítulo se describen técnicas concretas para aplicar los procedimientos generales y los principios teóricos, descritos en los capítulos anteriores, a un proyecto de simulación visual de materiales. Estas técnicas están incorporadas de diferentes modos en diferentes programas comerciales. No tiene sentido dar una descripción completa sin referirse con algún detalle a los pasos que hay que dar en un entorno en el que es indisociable la técnica general, basada en conceptos, de la técnica específica, basada en una interfaz compleja que puede variar relativamente de uno a otro programa.

En la medida de lo posible se incluyen descripciones que valen para prácticamente cualquier programa de simulación disponible en la actualidad, pues todos ellos se basan en algoritmos similares a los descritos de modo genérico en los capítulos previos. Pero en la mayoría de los casos será necesario dar descripciones referidas a un programa determinado para que se entienda adecuadamente el método.

El programa de referencia que utilizaré en estos casos es 3ds Max con mental ray. La razón principal es que es el programa que conozco mejor pues es en el que he realizado trabajos profesionales y he impartido cursos desde hace más de 20 años. También he impartido cursos utilizando otros programas más sencillos como SketchUp o Artlantis. Y también he probado durante períodos del orden de unos 6 meses otros programas como Maya, Blender, Maxwell y, durante más tiempo, V-Ray. Conozco muy superficialmente, sin embargo, otros programas importantes como SoftImage, Lightwave o Cinema 4D. Por otra parte, la razón para elegir 3ds Max en trabajos profesionales o cursos no es solo personal sino que se debe a que, como profesor de una escuela de arquitectura, he de adaptarme al software disponible y la decisión que se tomó hace años fue utilizar el software más extendido en nuestro entorno profesional, entre arquitectos o diseñadores. Esto no quiere decir que sea mejor que otros pues la elección de uno u otro programa depende de muchos factores. Pero es uno de los más completos, lo que me permitirá explicar con detalle todo tipo de técnicas. Por otro lado, al utilizar 3ds Max con mental ray, que es también el motor utilizado por otros programas importantes, como Maya o SoftImage, las descripciones técnicas valen prácticamente igual para estos programas.

Por todas estas razones, aunque intentaré, como decía, mantener las explicaciones al nivel más general posible, los ejemplos y las descripciones técnicas de detalle estarán referidas a este programa.

4.1 *Shaders* básicos

Shaders

Como ya hemos visto en capítulos anteriores, el concepto de *shader* fue introducido por Robert Cook en 1984 y se desarrolló por la compañía Pixar para normalizar las descripciones de imágenes 3D. Implicó la colaboración directa o indirecta de los nombres más famosos en el desarrollo de la computación gráfica: Ed Catmull, Loren Carpenter, Robert L. Cook, Pat Hanrahan, James F. Blinn o William Reeves, entre otros. Su antecedente principal fue el desarrollo de otro famoso motor de *render*, REYES (siglas de Renders Everything You Ever Saw y alusión a Pont Reyes, la zona de California en que estaba situada Lucasfilm), por la Computer Division de la compañía Lucasfilm, a partir de la cual se formaría Pixar y que en 1982 ya produjo fragmentos animados en la película *Star Trek*

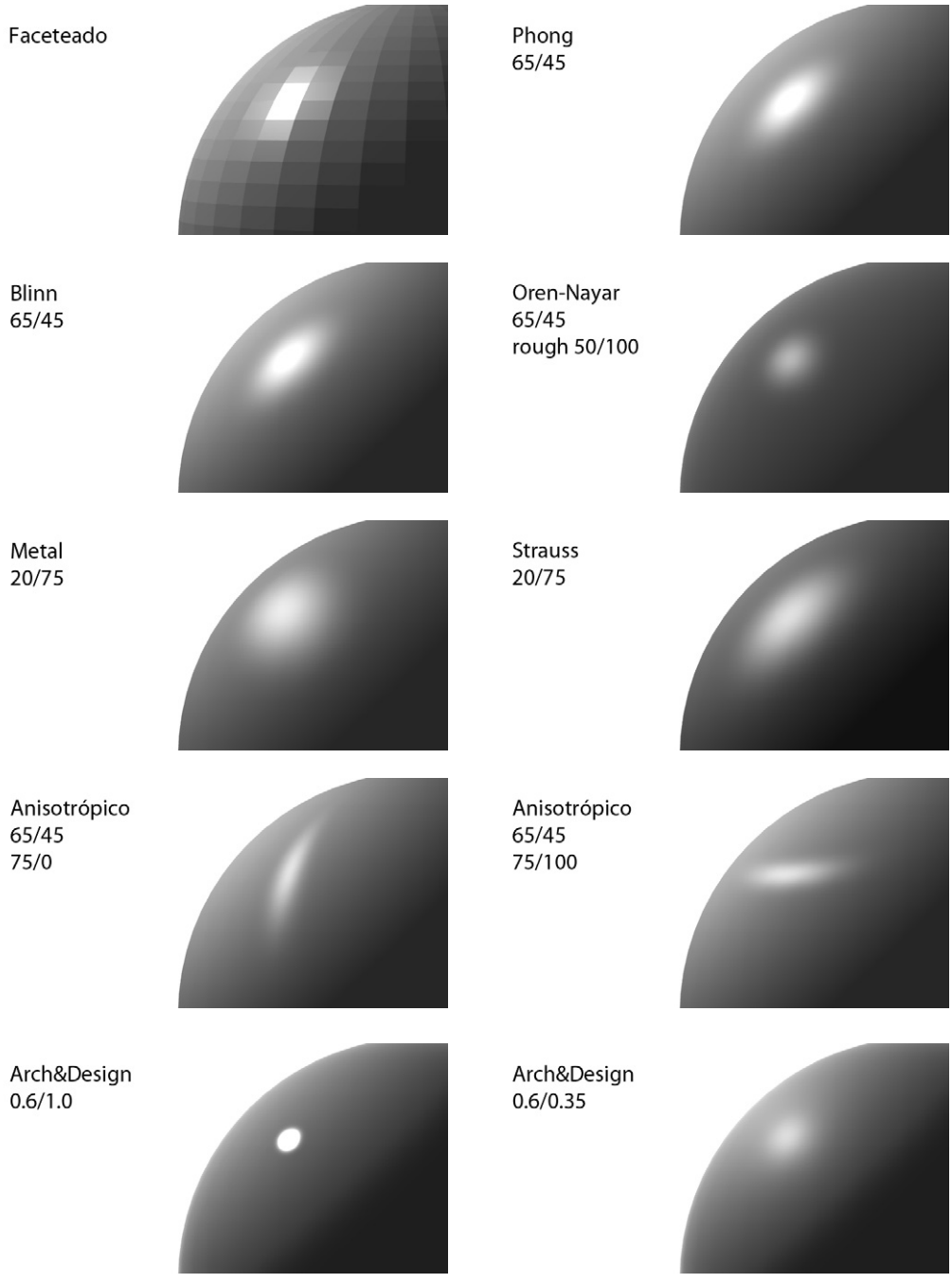


Figura 4.1 Shaders *básicos. Los valores corresponden a intensidad especular/dispersión especular (*glossiness*).*

II: The wrath of Kahn. En 1986 Steve Jobs adquirió esta división de Lucasfilm y fundó Pixar. A partir del trabajo previo de esta compañía se desarrolló el RenderMan Interface que sacó su estándar definitivo en 1989 y que incluía el RenderMan Shading Language, al que también me he referido en el capítulo anterior, y que contribuyó a fijar el concepto de *shader*.

Un *shader* es un pequeño programa que resuelve o modifica la representación de un objeto pero que también se utiliza para crear efectos especiales y, en general, para modificar la representación de una escena. Los *shaders* se utilizan de modo prioritario y principal para la representación de materiales, tal como veremos a continuación. Pero también se utilizan para otro tipo de efectos, incluyendo la modificación de las características de una luz.

Los *shaders* pueden estar integrados en un programa o incluirse como plug-in. Los principales desarrolladores de *shaders* dan facilidades para que los usuarios avanzados puedan crear sus propios *shaders*. En mental ray, por ejemplo, puede crearse un *shader* escribiendo el código adecuado en C o C++ y copiándolo en una carpeta que contenga bibliotecas compartidas tal como las *Dynamic Link Libraries* (DLL) de Windows. En las últimas versiones, por añadidura, se incluye un módulo especial, Meta SL, que facilita la creación de *shaders*. Y también es posible descargar un programa gratuito, *mental mill*, de Nvidia, que permite crear *shaders* para mental ray.

Por ejemplo, el siguiente fragmento de código de mental ray muestra la definición de un material, "mat1" y el *shader* básico que incluye esta definición:

```
...
material "mat1"
    opaque
    "mib_illum_phong" (
        "ambient"    0.0 0.0 0.0
        "diffuse"    0.7 0.7 0.7
        ...
        "specular"   1.0 1.0 1.0
        "exponent"   5
        ...
        "lights"     ["luz1"]
    )
end material
...
```

Esta definición incluye un *shader* estándar, "mib_illum_phong", que incorpora el modelo desarrollado por Bui Tuong Phong, el investigador vietnamita que desarrolló el modelo que lleva su nombre y al que ya me he referido en el capítulo 2, y una serie de valores predeterminados para los parámetros básicos de este *shader*.

Recientemente han aparecido métodos de creación de *shaders* aún más directos. Quien tenga un mínimo conocimiento de programación y quiera experimentar con *shaders* y poder probar los resultados con facilidad, puede, por ejemplo, descargar gratuitamente un programa reciente de edición de juegos, Unity 3D, que permite crear con relativa facilidad *shaders* simples utilizando versiones simplificadas de Java Script.

En el capítulo 2 también hemos visto los algoritmos principales que se utilizan para resolver la función reflectante de distribución bidireccional (BRDF), así como su evolución histórica, que está ligada a los nombres de los investigadores que han ido ampliando las posibilidades de esta función básica. En las aplicaciones corrientes se utilizan muchos de los *shaders* desarrollados por estos investigadores, en unos casos con el propio nombre de su inventor y en otros casos con nombres genéricos que no dan indicación sobre el algoritmo concreto utilizado.

La figura 4.1 muestra ejemplos de los diferentes resultados que se obtienen con los *shaders* básicos que se encuentran en la gran mayoría de los programas de simulación.

La geometría real se muestra en la primera de esta serie de figuras, activando el modo *Faceteado* o *Flat Shading*. Es una representación elemental que no figura en la lista de *shaders* pero se puede activar de diferentes modos: lo más sencillo, en cualquier programa, es seleccionar todas las caras del objeto y desactivar la opción "AutoShade" que suele estar activada por defecto (en 3ds Max, también se puede marcar la casilla "faceted", en los parámetros básicos del material). Con esto se eliminan las interpolaciones incorporadas a todos los *shaders* básicos y se mues-

tra la geometría real del objeto. Es decir, que una esfera, por ejemplo, aparecería como un conjunto, más o menos denso, de caras planas. El volumen viene dado por una función muy simple que colorea de diferente modo las caras. Se dio a conocer por primera vez en un artículo de Bouknight de 1970 y consiste, como ya hemos visto, en computar el ángulo que forma el vector orientación de la fuente de luz con el vector normal a la superficie que se quiere simular y asignar una intensidad a la cara en función de ese ángulo: si el ángulo es 0º la intensidad será máxima y si es 90º, nula. Proporciona un faceteado plano que se corresponde con la geometría real del objeto.

Si se desactiva la representación por facetas y se activa el modo *AutoShade*, la curvatura aparente de la esfera que se muestra en los visores se procesa por un método conocido como *Gouraud shading* (Gouraud, 1971) o modelado local por interpolación lineal (*incremental shading*, *interpolated shading*), que modifica los valores de salida interpolando las intensidades entre píxeles adyacentes.

Estos son los dos modos más simples de representación. Los ejemplos siguientes corresponden a los *shaders* básicos que hemos visto en el capítulo 2 activados desde el editor de materiales de 3ds Max a través del material *standard* y que son similares a los que se encuentran en muchos otros programas.

Phong y Blinn, que ya he descrito en el capítulo 2, dan resultados casi idénticos, probablemente inapreciables en la mayoría de los casos. Los valores indicados en la figura, 65/45, están referidos a los dos parámetros principales de estos *shaders*: nivel o intensidad especular (*specular level*) y brillo o dispersión del reflejo (*glossiness*).

Oren-Nayar, que también hemos visto en el capítulo 2, es una variante de los anteriores que corrige el error en la distribución del reflejo para superficies rugosas (véase el apartado correspondiente en el capítulo 2 para una explicación más amplia de este error). Incluye un parámetro adicional, *roughness*, para ajustar el grado de rugosidad de la superficie. Los materiales avanzados, como *Arch&Design* de mental ray que veremos más adelante, incorporan esta variante con un parámetro denominado así (*roughness*).

Metal, el término utilizado en 3ds Max, es una variante de Cook y Torrance que también hemos visto. El color del reflejo se adapta al color del material y los valores de los parámetros básicos dan resultados más intensos que, en la figura, se han atenuado para facilitar la comparación con los anteriores: 20/75 son los valores de nivel especular (rebajado) y de *glossiness* (más concentrado). Es un *shader* poco utilizado en la actualidad pues se consiguen resultados mucho mejores con materiales avanzados, como veremos más adelante.

También se utiliza cada vez menos la otra variante incluida en las figuras, *Strauss*, que también se ha mencionado en el capítulo 2 y que, en su momento, supuso una variante más cómoda con respecto a la anterior. El valor de 20 indicado en la figura se refiere a un parámetro específico de este *shader*: “metalness”. El otro valor, 75, es el mismo que en metal, *glossiness*.

Un *shader anisotrópico* permite alargar el reflejo en una determinada dirección, cuya orientación también puede modificarse. Si el valor de anisotropía se aumenta hasta 75 (en un rango de 0 a 100) como en la figura, el reflejo se alarga notoriamente. Y si se cambia el valor de orientación de 0 a 100, la dirección gira 90º. Véanse las descripciones del capítulo 2 correspondientes a algunos de los algoritmos principales para simular este efecto: Poulin y Fournier, Ward o Ashikhmin y Shirley. Los reflejos anisotrópicos los veremos con mayor detalle en un apartado específico, más adelante, referidos a materiales avanzados que utilizan otros rangos de valores pero que tienen el mismo sentido.

Hay otros *shaders* que no están incluidos en estas figuras, sea porque la utilidad es relativa sea porque los analizaremos a fondo más adelante.

Si se utiliza 3ds Max con mental ray hay disponibles, en las versiones recientes, materiales avanzados que incluyen *shaders* más

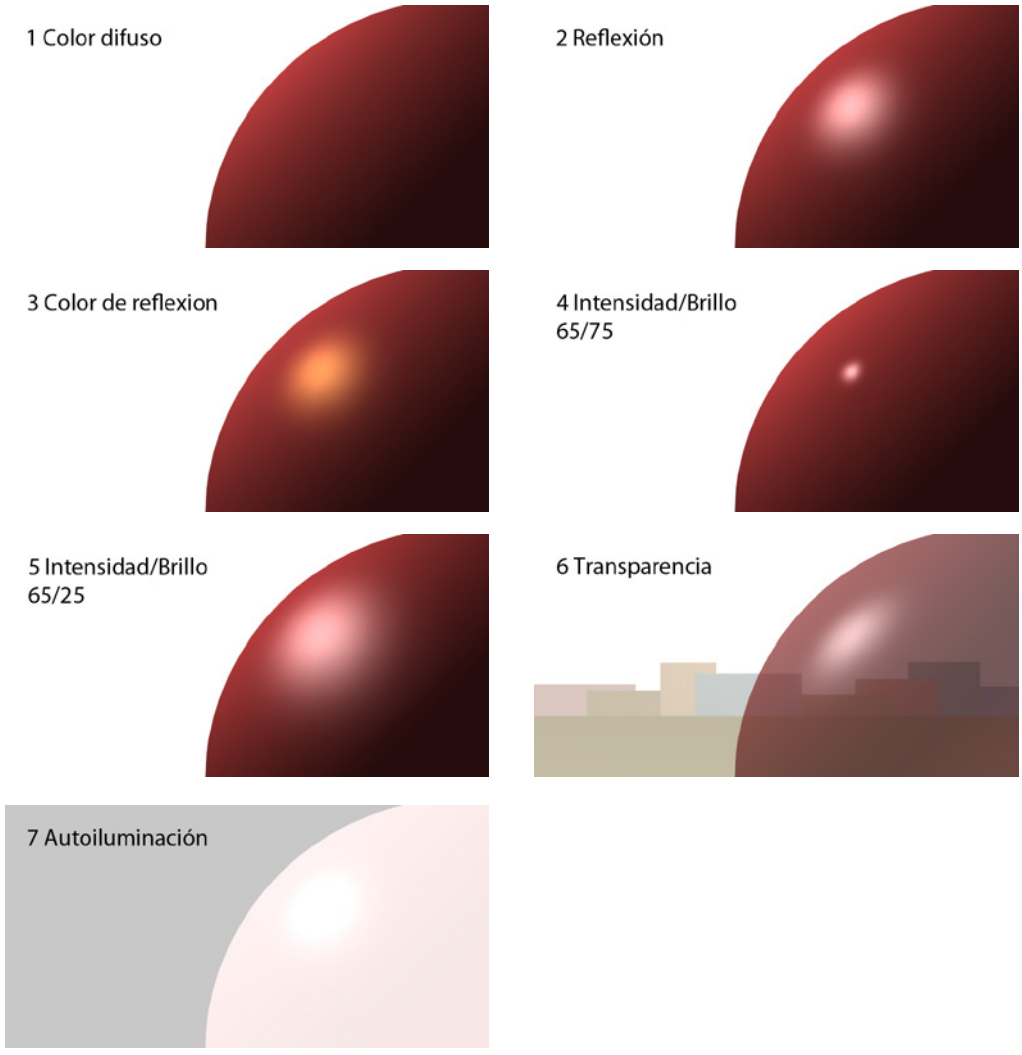


Figura 4.2 Parámetros básicos.

modernos y más parámetros de control. En particular, el material *Arch&Design*, diseñado específicamente para simulaciones de arquitectura y diseño y que utilizaremos extensamente en los ejemplos que seguirán, da resultados incomparables desde el punto de vista de la calidad.

¿Entonces, por qué no olvidarse de todos los *shaders* anteriores?

Hay varias razones. Por un lado, este libro pretende ser una introducción a todo tipo de programas de simulación. Y muchos programas muy eficaces pero más modestos, no incluyen *shaders* avanzados sino variantes de los indicados. Por otro lado, para escenas simples, estos *shaders* son más sencillos de manejar pues solo incluyen unos pocos parámetros. Y, en tercer lugar, por las mismas

razones que se ha incluido un largo capítulo dedicado a la teoría, quien quiera profundizar en todos estos temas es bueno que conozca cómo han evolucionado los algoritmos de simulación y que hay detrás de unos cuantos botones que no siempre hacen lo que esperamos.

Por desgracia los programas comerciales rara vez indican los algoritmos utilizados en sus materiales más avanzados. Mental ray utiliza con seguridad los BRDF de Phong, Blinn, Cook y Torrance y Ward, pues así está explicado en algunas de sus publicaciones principales, como el manual de Thomas Drienmayer en el que se indica que el algoritmo utilizado para anisotropía es un *shader* que parte de la propuesta inicial de Ward pero la desarrolla por medio de derivadas superficiales de los vectores u,v (véase Drienmayer, 2005, p. 107). Y es muy probable que, en sus versiones más recientes utilicen variantes de los de Lafortune *et al*. (1997) o Ashikhmin y Shirley (2000).

En cualquier caso, tampoco se trata de averiguar el método concreto utilizado en un programa comercial sino de tener presente que están basados en los métodos principales que hemos visto y, sobre todo, en los conceptos y las propiedades que se integran en unos parámetros determinados.

Parámetros de *shaders* básicos

Si solo consideramos una región puntual, como hemos visto en los primeros capítulos, el color dependerá de una serie de propiedades básicas (absorción, reflexividad, transparencia) que dependen de las características del material y que afectan a ese punto concreto.

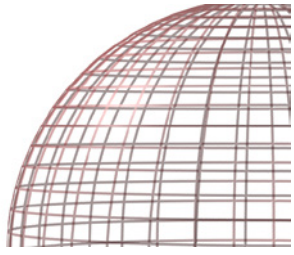


Figura 4.3 El parámetro Wire.

Estas propiedades se pueden incorporar a los *shaders* por medio de valores literales que ocupan unas pocas líneas, a diferencia de los mapas que requieren una referencia que remite a una información externa que puede ocupar mucho más espacio que el propio *shader*. Estos valores son prácticamente idénticos en todos los programas comerciales y se denominan “parámetros básicos”. Son los siguientes.

Color local o *Color difuso*. Es el color propio del objeto sin modificar por ningún reflejo, relieve, etc. Se define por medio de un selector de color que lo codifica con tres valores, corrientemente en modo RGB o HBS. Conviene recordar lo dicho en los primeros capítulos sobre las limitaciones en la representación de los colores.

Color especular. Es el color del resalte especular (*highlight*). En sistemas simples, que no computan la interacción entre objetos, es blanco pues se supone que corresponde a la iluminación de una luz blanca. Puede cambiarse este blanco a gris claro o bien copiar el color local y rebajar la saturación para simular un material metálico o mate. En el caso de materiales avanzados se substituye por reflejos que computan el entorno del objeto con lo que el color varía en función de estos reflejos.

Controles de especularidad. En los materiales simples, se simula por medio de dos controles que afectan a la intensidad y a la extensión del realce especular: el nivel especular (*specular level*), que afecta a la intensidad y el brillo (*glossiness*), que afecta a su tamaño. En los materiales avanzados estos parámetros vienen dados por la intensidad del reflejo (*reflectance*) y por la dispersión del reflejo o el lustre (*glossiness*). No debe perderse de vista que estos controles carecen de sentido si la superficie no refleja luz en dirección del espectador.

Transparencia/Opacidad/Refracción. El color local puede combinarse con el color de los objetos que estén en la misma dirección

que el punto de vista para simular un efecto de transparencia simple. El rango va de 0 % (total transparencia) a 100 % (total opacidad). Hay parámetros adicionales, más o menos corrientes en programas de simulación. Los materiales estándar de 3ds Max incorporan una serie de parámetros (Atenuación, Filtro...) que han quedado bastante obsoletos pues es preferible utilizar los materiales avanzados para simular transparencias.

Autoiluminación. El color de las partes no iluminadas de un objeto puede manipularse de tal modo que no se tenga en cuenta el efecto de las luces que inciden sobre él y que crearían sombras propias. El efecto resultante es que el material parece que irradia luz, que es "autoiluminante". Este recurso se utiliza para crear luminarias u objetos luminosos. El valor de autoiluminación se puede especificar linealmente, con números asociados al valor relativo, que es la opción más sencilla y más corriente, o asociados a un determinado color.

La figura 4.2 ilustra estos parámetros básicos.

Otros parámetros

Además de estos parámetros principales hay otros más o menos secundarios que se utilizan, o se utilizaban en algunos casos, para crear efectos especiales o para corregir insuficiencias.

Ambiente. Es un parámetro que se usaba en sistemas de iluminación local para que las partes de un objeto que no recibiesen iluminación no se representasen completamente negras. Se mantiene para no romper la continuidad con estos sistemas pero es mejor olvidarse de que existe.

Alámbrico (*Wire*). Se encuentra en algunos programas y crea un efecto de calado, representando únicamente las aristas de la malla con un grosor que puede especificarse en la sección de parámetros extendidos. El "tamaño alámbrico" (*wire size*) puede especificarse en unidades o píxeles; en general, se utiliza la opción "unidades" para dar al grueso del alambre valores coherentes con la geometría de la escena. Véase la figura 4.3.

2 lados (*2-sided*). No es propiamente un parámetro de material sino un corrector de la convención por la que tan solo una de las caras de un modelo, la que tiene la normal apuntando en la dirección "correcta", es visible. Se usa en los casos siguientes: *a*) Cuando se trabaja con modelos en los que las normales pueden haber quedado desordenadas por diversas causas y desaparecen caras en la representación y no se tiene tiempo (o ganas) de corregirlas manualmente; *b*) Cuando se utiliza el parámetro anterior, alámbrico, para que resulte real el efecto de calado en objetos con volumen. Si no se utilizase esta opción no se verían las caras del interior del objeto; *c*) Con el parámetro "opacidad" inferior al 100 % para producir efectos de transparencia en objetos con un volumen visible, por la misma razón; *d*) Cuando se quiere ver el interior del objeto sin modificar las normales.

4.2 *Shaders* arquitectónicos

Los principales programas de simulación que se utilizan en la actualidad utilizan materiales denominados "monolíticos" (*monolithic materials*), materiales que no están compuestos de agrupaciones variables de *shaders* sino de una serie de propiedades principales (*shaders* complejos o combinaciones internas de *shaders*), que se asemejan de modo más intuitivo a las propiedades de los materiales reales aunque también cuentan con la posibilidad de ampliar estas propiedades por medio de todo tipo de *shaders* complementarios.

El más conocido y probablemente el más completo es el material *Architectural* (*mia*) *Material*. Las siglas *mia* corresponden a *mental images architectural* y es un material desarrollado por mental ray y que se ha integrado de diversos modos en las tres aplicaciones principales que utilizan mental ray como motor de *render*: Maya, Soft Image/XSI y 3ds Max. En Maya se denomina *mia_material*. En XSI se denomina *Architectural*. En 3ds Max

se denomina *Arch&Design*. En otros programas se encuentran materiales similares; así ocurre con el material V-Ray, cuyas características son muy similares a este material.

Tal como dice la propia documentación de mental ray "el mental ray mia_material es un *shader* monolítico diseñado para dar soporte a la mayoría de los materiales utilizados en simulación visual (*rendering*) de arquitectura y diseño de productos. Permite simular la mayoría de los materiales de superficies duras tales como metales, madera y vidrio. Está especialmente ajustado para simular de un modo eficaz las reflexiones y refracciones con mayor o menor grado de dispersión (*glossy reflections and refractions*), substituyendo al material DGS, así como al vidrio de alta calidad, substituyendo al material dieléctrico".

Los dos materiales a que se refiere este párrafo, el material DGS (*Diffuse-Glossy-Specular*) y el material dieléctrico (materiales aislantes, no metálicos), son dos tipos de materiales avanzados que se utilizaban en versiones previas de mental ray y que daban excelentes resultados para simular de modo más preciso los efectos a que aluden sus nombres.

Aunque los detalles de sus algoritmos internos no son del todo conocidos se pueden apuntar algunas referencias. Las variaciones de rugosidad de los colores difusos dependen en mayor o menor grado de ajustes de rugosidad que están relacionados con el *shader* Oren-Nayar. Las variaciones anisotrópicas dependen de variantes del *shader* de Ward. Y las reflexiones y refracciones están probablemente basadas en métodos avanzados derivados de algoritmos como los de Lafortune y otros que hemos visto en el capítulo 2. Por añadidura, incluye métodos para ajustar automática y manualmente las curvas BRDF. Y cuenta con métodos adicionales para simular la translucidez.

En los apartados que siguen se describen y se discuten las características principales de este material. Estas descripciones están referidas principalmente a mental ray con 3ds Max, pero pueden extenderse facilmente a los otros programas mencionados.

Estructura y parámetros básicos del material *Arch&Design* (*mia material*)

El material Arch&Design tiene (en 3ds Max) 10 secciones de importancia variable. Las describo brevemente en lo que sigue, remitiendo la explicación a otros apartados en donde se amplía la descripción de su uso con análisis y ejemplos concretos.

La primera sección, ***Templates***, presenta una serie de plantillas para simular acabados genéricos (mate, satinado, brillante) o acabados específicos (madera, hormigón, cerámica, plástico, varios tipos de metales, etc.). Al seleccionar una de estas plantillas, los parámetros principales, de la siguiente sección, se ajustan a una serie de valores predeterminados que luego se pueden refinar.

La segunda sección, ***Main material parameters***, es la principal e incluye una serie de parámetros básicos que se organizan en los siguientes cinco grupos. Doy entre paréntesis los valores predeterminados:

Color local (*Diffuse color*). Al igual que con los materiales estándar, el color se especifica directamente mediante el icono de acceso al selector de colores que hay junto al parámetro *Color*, o mediante un mapa asociado al que se acccede mediante el pequeño icono situado a la derecha, que está en blanco si no hay mapa o muestra una "M" si hay una mapa asociado (o una "m" si hay mapa pero está desactivado). *Diffuse level* (1,0) controla la contribución del componente difuso que es total por defecto (1,0) o puede reducirse hasta anularse (0,0). *Roughness* (0,0) controla la rapidez con que el componente difuso se mezcla con el componente ambiental lo que se traduce en cierto grado de rugosidad (apreciable sobre todo en las zonas de transición de iluminación a sombra). Si se escoge *glossy finish* en la lista de *Templates*, este valor cambia a 0,0. Si se escoge *matte finish* cambia a 0,2.

Conviene recordar que se utilizan *valores decimales en el selector de color*. Cuando

se utilizan materiales avanzados, los valores RGB no son números enteros sino valores decimales que dan más precisión y permiten trabajar con valores de alto rango dinámico (como los formatos HDR). Si se introducen valores RGB que vienen de otros contextos hay que hacer una regla de tres para pasar de un rango 0 – 255 a un rango 0,00 – 1,00. Pero en el selector de color de 3ds Max hay un evaluador que facilita el trabajo. Para usarlo, hacer un clic sobre un cuadro numérico del selector y presionar ctrl+n. Esto abre un cuadro denominado "numerical expression evaluator". Si se quiere traducir, por ejemplo, un color de valor 105 a decimal, escribir 105/255 (o, si se ha copiado de otro lugar, presionar el botón *Paste*). En la parte inferior aparece el valor traducido (0,411765). Presionar Intro y el nuevo valor se aplicará automáticamente (truncándose según el número de decimales dado en preferencias).

Reflexión (*Reflection*). Las reflexiones de este material, a diferencia de los *shaders* básicos, dependen no solo de la fuente de luz sino también de la escena y se procesan por medio de rayos trazadores (*ray tracing*). El resultado es más exacto desde el punto de vista físico y también más convincente desde el punto de vista perceptivo pues los "reflejos" de un *shader* estándar solo simularían el caso en que una luz es tan fuerte que su reflejo inhibe todos los demás. Los parámetros de este grupo los analizaremos con detalle en el apartado "Reflejos", más adelante.

Refracción (*Refraction*). Los parámetros de refracción son similares a los de reflexión y, como en el caso anterior, los veremos con más detalle en el apartado "Transparencias".

Translucidez (*Translucency*). La traslucidez es una variante de la transparencia en que se produce una dispersión interna. También lo veremos con detalle más adelante, en el apartado del mismo nombre, a continuación del apartado sobre transparencia.

Anisotropía (*Anisotropy*). El parámetro principal, *anisotropy*, tiene el mismo sentido que con los materiales estándar de 3ds Max: hace que los reflejos discurran a lo largo de una dirección determinada más que en otra. Como en los casos anteriores, me remito al apartado "Reflejos anisotrópicos".

La tercera sección, ***BRDF*** (*Bidirectional Reflectance Distribution Function*) incluye controles para ajustar las variaciones de las reflexiones según el ángulo de observación. La analizaremos, con ejemplos adecuados, en el apartado sobre "Reflejos".

La cuarta sección, ***Self Illumination*** **(*Glow*)** se utiliza para crear materiales que emiten luz. Veremos sus parámetros y ejemplos de utilización en el apartado "Autoiluminación".

La quinta sección, ***Special effects***, incluye dos tipos de efectos que pueden utilizarse para aumentar el realismo sin excesivo coste de computación.

Ambient occlusion permite simular el efecto de oclusión ambiental, lo que es un método sencillo para simular la iluminación indirecta. Se estudia con más detalle en el libro sobre simulación de la Iluminación y me remito a este libro para una explicación más completa de sus parámetros y de alternativas de utilización, más allá de su incorporación al material *Arch&Design*. El parámetro principal es *Max distance* que define el radio, en unidades de la escena, con que se rastrean los objetos que pueden contribuir a la oclusión. Valores en torno a los 20 o 50 cm pueden dar resultados adecuados, aunque habrá que experimentar según los casos. *Samples* (16) controla el número de muestras que se toman para simular este efecto. Puede ser necesario elevarlo a 32 o 64 en casos especiales. Si se aumenta este valor mejora el resultado pero también el tiempo de cálculo. *Use color from other materials* (*Exact AO*) hace que se tengan en cuenta los valores de materiales cercanos, lo que da resultados más exactos pero más lentos. *Shadow color* controla la obscuridad de las sombras propias de AO. Suele ser preferible un tono gris oscuro antes que un negro aunque todo depende de los casos concretos. *Custom*/*Global* permite cambiar el valor predeterminado para el color de la luz ambiente (*custom*) por *global*. Es mejor dejarlo como está.

Round corners se utiliza para redondear virtualmente las aristas de los objetos a que se aplique este material con un radio de fileteado especificado por el parámetro "Fillet radius". Convendrá empezar con valores bajos, tal como 0,1 y subirlos gradualmente. Si se activa la casilla *Blend with other materials* este efecto de redondeado se aplica también a materiales adyacentes. Este parámetro hace que la normal a la superficie se modifique en las esquinas lo que crea un efecto de curvatura, es decir, funciona de un modo muy similar al método utilizado por los mapas de tipo *Bump*. Esto puede funcionar relativamente bien con formas simples aunque no tanto en formas más complejas en las que será preferible modificar la geometría real. Algunas de las figuras que se incluyen en los apartados sobre reflejos y transparencias, más adelante, incluyen este efecto.

La sexta sección, ***Advanced rendering options***, contiene parámetros adicionales para reflexiones y refracciones, principalmente, *Max distance* y *Max trace depth* que controlan la profundidad a que se aplican las reflexiones y/o refracciones. También incluye controles adicionales para transparencias y multiplicadores ligados a los cálculos de iluminación indirecta. Probar a variarlo únicamente en casos de escenas con muchos rebotes debidos a reflexiones y refracciones en las que el color pueda resultar alterado. En el ejemplo de simulación de agua, que se da en el capítulo siguiente, se analiza con detalle el efecto de este parámetro.

La séptima sección ***Fast glossy interpolation***, incluye parámetros adicionales que afectan principalmente a las reflexiones o refracciones dispersivas (*glossy*). Depende de que se haya activado la casilla correspondiente en la sección de parámetros principales, en reflexiones o refracciones. Al activar esta opción, los valores de las reflexiones dispersas se interpolan con lo que el resultado será, en principio, más rápido y más regular aunque menos exacto. Cuanto mayor sea la *grid density* (1/2 predeterminado) más preciso será el cálculo. Y a la inversa. Pero las diferencias serán muy a menudo inapreciables, por lo que podemos olvidarnos de esta sección en la mayoría de los casos.

La octava sección, ***Special purpose maps***, incluye mapas para efectos específicos que no están incluidos en los parámetros básicos. Pues todos los parámetros principales permiten introducir mapas en lugar de valores, sea para el color local, la rugosidad, la reflexión, la dispersión de la reflexión, la transparencia, la dispersión de la transparencia y la refracción, la translucidez y sus características propias, y la anisotropía y la orientación de la anisotropía. Pero hay algunas propiedades adicionales que no están incluidas en estos parámetros básicos y que se incluyen en esta sección. Son las siguientes:

Mapas de *Relieve* (*Bump*), que funciona básicamente igual que el método de simulación equivalente del material *Standard*. Pero hay una diferencia importante y es que se puede aplicar a los componentes especular o de dispersión especular en lugar de al componente de color local si se marca la opción "Do not apply bumps to the diffuse shading". Esto hace que el aspecto parezca deberse a una propiedad interna de la superficie y puede resultar más convincente en determinados casos. Véase el apartado sobre mapas de relieves en el capítulo siguiente.

Mapas de *Desplazamiento* (*Displacement*), que modifica virtualmente la geometría real con arreglo a los métodos que se han descrito en el capítulo sobre teoría y métodos generales. Véase también el apartado citado sobre mapas de relieves en el capítulo siguiente.

Mapas de *Recorte* (*Cutout*) que remueven partes del objeto y se aplica principalmente para la simulación de figuras. Véase el apartado sobre mapas de recortes en el siguiente capítulo.

Mapas de *Autoiluminación* (*Self illuminating color*) utilizado en conjunción con las propiedades de autoiluminación, ampliándolas mediante el uso de mapas. Véase el apartado sobre mapas de aplicación a la autoiluminación en el capítulo siguiente.

Mapas de *Entorno* (*Environment*). Se utiliza para añadir un mapa de entorno que permite simular el reflejo de una escena virtual a par-

tir de una imagen ligada de modo exclusiva al material. Véase el apartado sobre mapas de aplicación a reflejos en el capítulo siguiente.

Mapas de asignación de *Color adicional* (*Additional color*). Se puede utilizar para añadir un *shader* que se superpone a los definidos para el material. Se utiliza a veces para superponer un determinado *shader*, por ejemplo de tipo *Ambient occlusion* (con una lista de parámetros más completa que la incluida en la sección citada anteriormente), al material.

Las dos últimas secciones tienen menos interés. La novena sección, ***General maps***, agrupa los mapas utilizados en las propiedades principales lo que permite desactivarlos o activarlos si interesa hacerlo. La décima sección, ***mental ray connection***, permite incorporar al material Arch&Design algunos *shaders* propios del material mental ray que era, hasta la aparición del material arquitectónico, el material con mayores posibilidades, con acceso a un mayor número de *shaders*, alguno de las cuales quizás puede interesar recuperar.

Reflejos

La simulación de los reflejos resulta compleja por varias razones. En primer lugar, porque implica al conjunto de la escena, es un efecto que no puede aislarse de este conjunto. En segundo lugar, porque dependen en gran medida de las características del objeto, tanto desde el punto de vista geométrico como material. En tercer lugar, porque hay un número mayor de parámetros que influyen en el resultado y que dependen de las condiciones de observación y del tipo de cálculo y de la precisión que se quiere alcanzar.

Por estas razones, esta sección está dividida en diferentes apartados que intentan ordenar esta diversidad de factores que hay que tener en cuenta a la hora de simular los reflejos de un objeto.

Es importante tener presente, en esta como en las secciones siguientes, que en esta introducción a los reflejos prescindimos de todo lo relativo al uso de mapas, un componente muy importante en la simulación de reflejos. Retomaré, por tanto, el tema de los reflejos (y de transparencias y autoiluminación) en el capítulo siguiente, después de analizar los diversos modos de aplicar mapas a la simulación de un objeto.

Criterios generales

Al igual que ocurre al organizar una escena real, para que los reflejos de un objeto resulten adecuados para las intenciones generales de la composición, tan importante como ajustar las propiedades del material, es organizar adecuadamente el entorno. Cuando un fotógrafo profesional saca una fotografía de un objeto real reflectante, lo rodea de un fondo uniforme para que los reflejos sean uniformes e intensos. Y luego dispone las luces de tal modo que creen puntos luminosos en posiciones clave.

Lo mismo vale para un objeto virtual. Lo primero que habrá que controlar es el escenario general. Si el objeto está aislado, habrá que cambiar el color de fondo. Si está apoyado sobre un plano, habrá que tener muy presente que los bordes del plano aparecerán como líneas marcadas más o menos visibles. Si hay otros objetos cercanos, habrá que tener en cuenta que estos objetos aparecerán en las zonas reflectantes.

En algunos casos puede interesar crear objetos especiales para el fondo, con curvas

Figura 4.4 Reflejos. Nudo reflectante.

suaves de transición entre el plano vertical y el horizontal, para evitar que aparezcan líneas marcadas, y de colores suaves y uniformes. Es decir, organizar la escena de modo similar a como lo haría un fotógrafo profesional pero sacando partido de la facilidad de un programa de simulación para crear fondos adecuados de cualquier dimensión y configuración.

En los últimos ejemplos daré mayor importancia a la preparación del escenario. En los primeros, para simplificar la exposición y para que los ejemplos se puedan reproducir con facilidad, la complejidad del escenario estará reducida al mínimo.

La figura 4.4, de un nudo reflectante, ilustra lo anterior. En la superficie del nudo podemos distinguir claramente las líneas correspondientes al borde de la base de madera, el fondo, que corresponde a un cielo con nubes, y la luz principal, un rectángulo blanco.

Parámetros básicos

En el grupo *Reflection* de la sección *Main material parameters* nos encontramos con los parámetros siguientes. El parámetro *Reflectivity* (0,0 o 0,6 según el material predeterminado que se elija, pues hay dos tipos) define el nivel de reflexión del material (0,0 es mate y 1,0 especular). El parámetro *Color* afecta al color de la reflexión: el color predeterminado es blanco y no conviene cambiarlo. Si se presiona el pequeño botón cuadrado que hay junto al icono que muestra el color y se escoge un mapa, el reflejo quedará afectado en el sentido de que las zonas blancas mantendrán el reflejo y las negras lo anularán (y las grises lo afectarán en mayor o menor grado). Como veremos más adelante, a veces se utilizan mapas en blanco y negro, derivados de una textura, para realzar o rebajar determinadas partes del reflejo. El parámetro *Glossiness* (1,0) concentra o dispersa el brillo especular. Con 0,0 la dispersión es completa y al aumentar este valor la concentración irá aumentando, hasta el valor máximo, 1,0, perfectamente especular. Si el valor de este parámetro es inferior a 1,0 hay que tener en cuenta el parámetro siguiente, *Glossy samples* (8,0) que controla el número de muestras que se toman para calcular las reflexiones. Para obtener mejores resultados, más suaves, en vistas cercanas, puede convenir subirlo hasta 16 o 32, tanto más cuanto más dispersa sea la reflexión. Esto puede implicar manejar algunos parámetros de optimización que ya he mencionado: *a*) *Fast* (*interpolate*), que acelera el cálculo a costa de la precisión; *b*) *Highlights+FG only*, que fuerza al motor de *render* a no tener en cuenta más que los cálculos que afectan a las zonas brillantes y a los valores dados por *Final gather* (el sistema de cálculo de iluminación predeterminado en mental ray). Es decir, hace que el objeto se comporte como un *shader* básico; si no hay objetos reflectantes en primer plano, acelera los cálculos, pues se inhibe el procesamiento por *ray trace*, sin pérdida apreciable de calidad. Es recomendable utilizar esta opción con valores bajos de *Glossiness*, por ejemplo por debajo de 0,4 o 0,3 ya que los reflejos serían inapreciables en esos casos; *c*) *Metal material,* que, si se activa, hace que se tome más en cuenta la contribución del color local a la reflexión especular, tal como ocurre con los metales.

Los reflejos se controlan también desde otras secciones del material *Arch&Design*. En la tercera sección, *BRDF*, encontramos los siguientes parámetros que se utilizan para controlar la curva de distribución de los reflejos en función del ángulo de incidencia de la luz y del ángulo de visión. La opción *By IOR* (Fresnel) hace que este ajuste dependa exclusivamente del índice de refracción, como ocurre en materiales dieléctricos (no metálicos, malos conductores). La opción *Custom reflectivity function* permite ajustar directamente la curva que controla la BRDF. El valor dado para 0º define la intensidad de la reflexión cuando la cámara (o la vista) está situada perpendicularmente a la superficie y el dado para 90º cuando la cámara es paralela a la superficie. La curva define el tipo de transición entre ambos extremos. En materiales como los metales se requiere una curva tal que con 0º el valor sea alto (0,8 a 1,0) pues los metales son, en general, casi tan reflectantes en direcciones cercanas a la perpendicular como en ángulos

rasantes, a diferencia de lo que ocurre con los materiales reflectantes no metálicos. En materiales como, por ejemplo, la madera laqueada, se requiere una curva con un valor a 0º más bajo (de 0,1 a 0,3). Los ejemplos que se dan más adelante ilustran el uso de estos controles.

Muchos usuarios expertos pasan por alto el comportamiento real de los materiales y manipulan la curva BRDF de modo que los resultados sean los que interesan. O introducen valores irreales para el índice de refracción (por ejemplo 25,0) que dan lugar a una curva con perfiles muy pronunciados para ciertos ángulos. Aunque esto puede escandalizar a los científicos no hay que perder de vista que la mayoría de los métodos de simulación tienen un objetivo principal: conseguir un determinado efecto que sea percibido como verosímil (aunque sea irreal). El ojo bien educado y unos objetivos claros son, en estos casos, un criterio más importante que una "objetividad científica", que a menudo pierde de vista las múltiples trampas y supuestos inverosímiles que hemos ido desperdigando por el camino durante un proceso de simulación.

Reflejos sobre una esfera

El análisis de los reflejos sobre una esfera nos permitirá presentar los tipos más generales de recursos para generar reflejos. Este análisis está basado en una escena (véanse las figuras adjuntas) que está organizada del siguiente modo. Hay dos objetos: una esfera de 15 cm de radio (y 64 segmentos) y un plano de 3 x 3 m, ambos situados en 0,0,0. El plano base tiene asignado un material con un mapa procedural tipo *checker* (con colores RGB 25,4,4/ 220, 210, 210). Aunque aún no hayamos hablado ni de mapas en general ni de mapas procedurales en particular, este mapa es muy simple (también es posible saltar al capítulo correspondiente antes de continuar si se quiere reproducir exactamente estos ejemplos). Los materiales de la esfera se detallan en los párrafos siguientes pues son los que vamos a analizar. Hay una cámara que apunta a la esfera. Y dos luces: una luz primaria, "L1", de tipo mrAreaSpot, con un área de unos 12 x 12 cm para que las sombras sean más suaves y otra luz secundaria, "L2", también mrAreaSpot, de área nula y que no arroja sombras ni reflejos (desactivar su contribución especular, seleccionándola y desmarcando la propiedad "Affect Specular" en el menú contextual) y que está situada en posición opuesta a

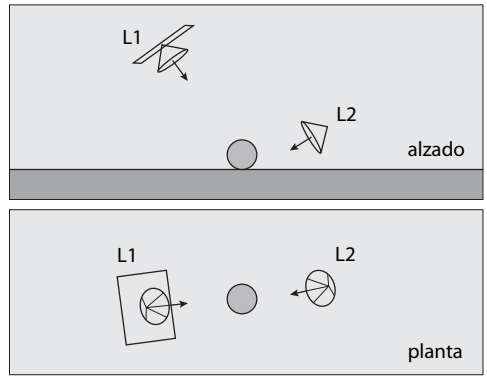


Figura 4.5 Reflejos sobre una esfera. Esquema inicial de la disposición de luces de la escena.

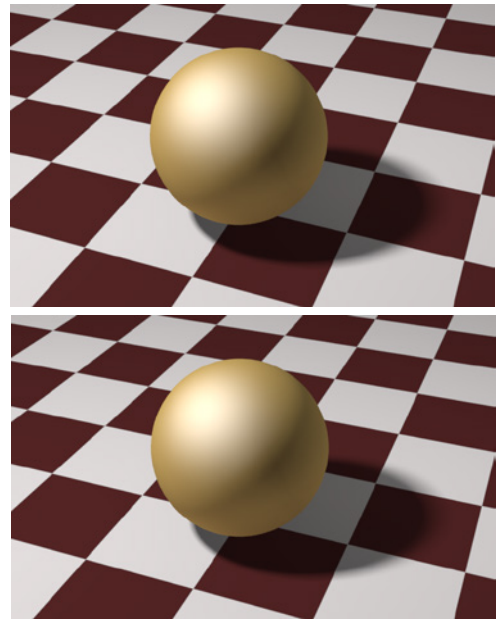


Figura 4.6 Reflejos sobre una esfera: a) *Material* Standard/ Blinn *con especularidad 85, 10, 0,25 (*level, gloss, soft*), b) Material* Standard/Metal *con especularidad 155, 65 (*level, gloss*).*

la anterior para aclarar sus sombras. Las intensidades (multiplicador) de estas luces son de 0,85 y 0,15 respectivamente. El esquema de la figura 4.5 resume esta disposición.

En el primer ejemplo se ha asignado a la esfera un material *Standard* con el *shader Blinn* (figura 4.6, a). En este caso, la especularidad se controla por medio de parámetros que ya hemos visto y que simulan una luz especular pero sin que el material refleje realmente lo que tiene alrededor. Las características del realce especular dependen de la intensidad y el color de la luz que ilumine la esfera, y los parámetros modifican este valor inicial por medio de, *a*) el nivel de especularidad (*specular level*), *b*) el grado de dispersión o concentración del realce (*glossiness*) y, *c*) un parámetro adicional que suaviza más (1,0) o menos (0,0) el resultado. Los valores utilizados en el ejemplo se indican en el pie de la figura. El color difuso de la esfera en este ejemplos y los que siguen es RGB 202, 135, 40. Como puede apreciarse en la imagen, este método es adecuado para objetos satinados pero que solo reflejan en zonas de su superficie situadas directamente frente a la luz. Puede utilizarse otro *shader*, *Metal*, derivado del *shader* de Cook y Torrance (fi-

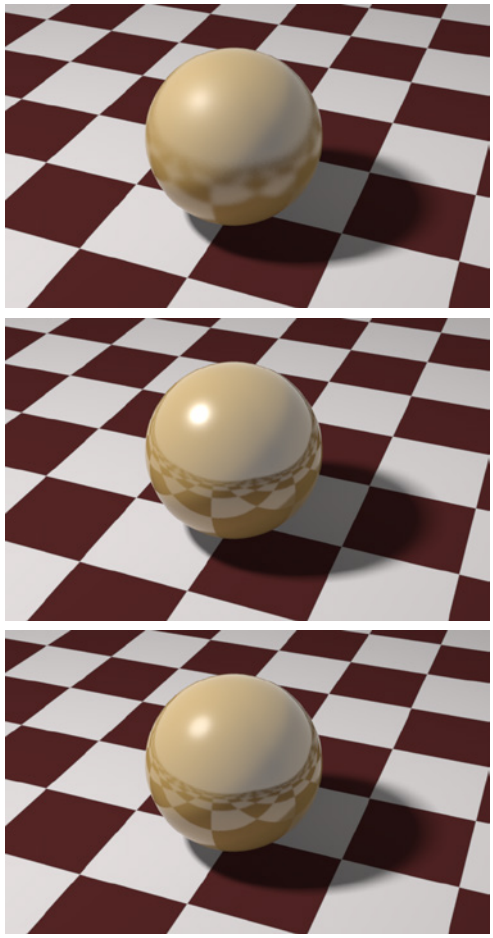


Figura 4.7 Reflejos sobre una esfera. Material A&D*:* a*) Diffuse* level *1,0. Reflectividad 0,65, 0,35;* b*) Diffuse* level *1,0. Reflectividad 0,75, 0,65. Al aumentar el valor de* glossiness*, el reflejo de la luz es muy visible;* c*) Misma configuración que la anterior pero se ha desdoblado la luz, tal como se indica en el texto, para controlar el reflejo.*

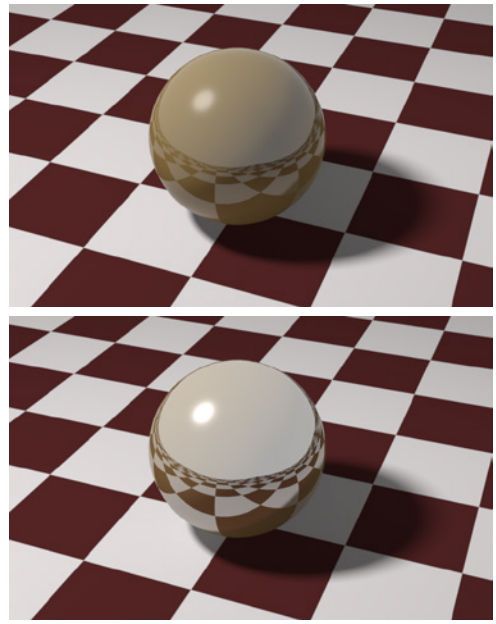


Figura 4.8 Reflejos sobre una esfera. Material A&D *con* Diffuse level *0,75 y reflectividad 0,65 (*gloss *1,0). Influencia de la función BRDF:* a*) BRDF predeterminada (0,2 para 0º, 1,0 para 90º, curva 5,0);* b*) BRDF modificada: 0,8 para 0º, 1,0 para 90º, 5,0).*

gura 4.6, b), que concentra más el reflejo y da resultados más simples que los ejemplos más avanzados que veremos más adelante. Su única ventaja es que es más sencillo de utilizar pues admite pocas variaciones.

En el segundo ejemplo se ha asignado a la esfera un material avanzado de tipo arquitectónico, *Arch&Design*. Este material refleja realmente lo que le rodea por lo que los resultados serán más exactos pero también requieren ajustes más cuidadosos. Las figuras 4.7 y 4.8 muestran ejemplos relevantes de las diferencias que se obtienen ajustando los principales parámetros que incluye este tipo de material.

Al aumentar el valor de *glossiness*, a partir de unos 0,3 o 0,4, el reflejo se concentra y la luz de área es claramente visible. Esto da lugar a que se altere la distribución de las luces, un problema de difícil solución y que es característico de las fotografías de estudio y al que volveremos más adelante. Pero con medios digitales puede solucionarse mediante un truco técnico sencillo. Las luces virtuales, como ya he mencionado, incluyen recursos especiales que nos permiten utilizarlas para que afecten solo a la iluminación difusa o solo a la especular. Por tanto, podemos clonar la luz que estamos utilizando, desactivar la contribución especular de la primaria y desactivar la contribución difusa de la auxiliar que, así, solo afectará al reflejo. Modificando a nuestro gusto el tamaño y la intensidad de esta luz auxiliar podemos controlar el aspecto de este refejo, que es lo que se ha hecho en los ejemplos siguientes. Todos los parámetros, excepto los indicados, se han dejado con los valores predeterminados. Es decir, que tenemos las siguientes luces en la escena, todas de tipo mr Area spot:

a) Luz primaria. Ilumina la escena y arroja sombras de área. El área es de unos 8 cm de radio y está situada a unos 90 cm de la esfera. Contribución especular desactivada. Multiplicador 0,85.

b) Luz secundaria. Ilumina la escena, apuntando a las sombras. No arroja sombras. Contribución especular desactivada. Multiplicador 0,15.

c) Luz de reflejos. No ilumina la escena ni arroja sombras. Área de unos 25 cm de radio y situada a unos 105 cm de la esfera. Contribución especular activada. Multiplicador 0,25.

Las propiedades especulares y difusas (y de transmisión en su caso) deberían estar adecuadamente equilibradas para que el equilibrio energético se mantenga aunque por ahora prescindiremos de esto, que se comenta más adelante. Y los valores predeterminados de la curva BRDF harán que la parte central de la esfera refleje mucho menos (un 20 %) que los laterales. Si variamos estos parámetros, como en las figuras citadas, podremos comprobar cómo afectan al resultado final. En la figura 4.8, en que la BRDF se ha llevado a valores altos para 0º, he reducido la intensidad de la luz de reflejo a 0,15 en lugar de 0,25. Si la contribución del nivel difuso es 0,0 y la BRDF para 0º está en el 100 %, la bola se comporta como un espejo perfecto muy poco realista, un caso que no ilustro pero que se puede comprobar con facilidad.

Si los reflejos son muy visibles la importancia de lo reflejado aumentará. Como es de suponer que estos ejemplos simplificados se aplicarán a casos en los que existirá una escena compleja, la prosecución del análisis dependerá de cada caso particular y, por tanto, no podemos ir mucho más allá. En el apartado sobre aplicaciones de mapas a reflejos ampliaré los ejemplos y el análisis de los casos.

Reflejos sobre un plano

La escena utilizada para los ejemplos que siguen incluyen varios cilindros de unos 20 x 60 cm (diámetro x altura), dispuestos a diferentes distancias de la cámara. He asignado a los cilindros un material de color que pueda distinguirse fácilmente en el reflejo sobre el plano. A partir de aquí, para analizar los reflejos sobre el plano, continuar como sigue:

1 Asignar al plano base un material *Arch&Design*. con un color gris poco satu-

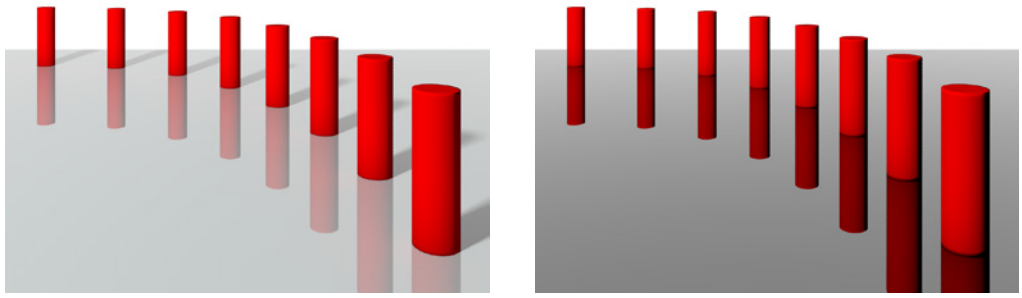


Figura 4.9 Reflejos sobre un plano. Variaciones de la relación Diffuse level/Reflection level*: a) 1,0/1,0; b) 0,0/1,0.*

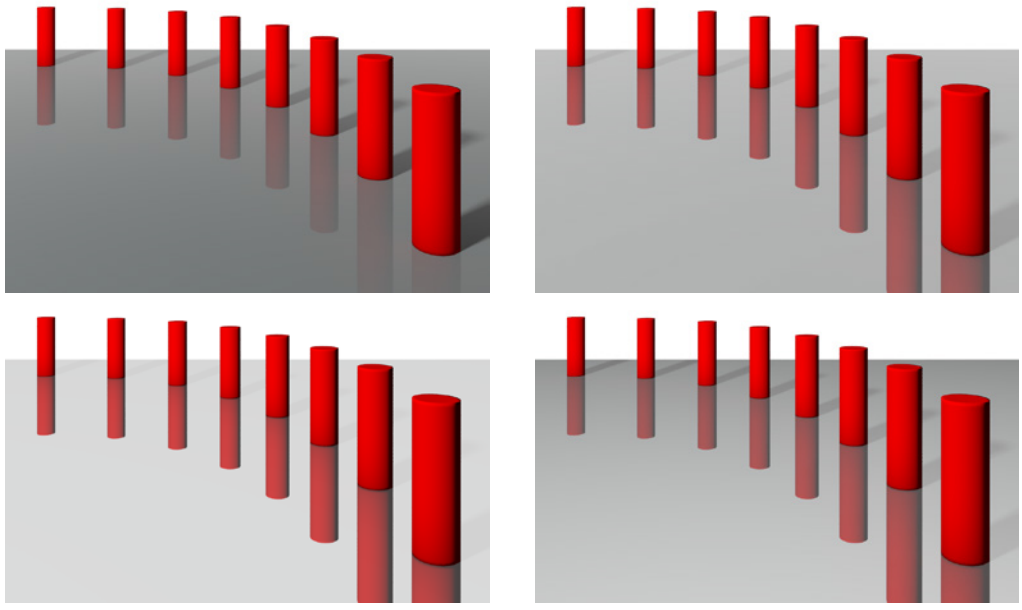


Figura 4.10 Reflejos sobre un plano. Variaciones de la curva BRDF (para una relación Diffuse level/Reflection level *de 0,6/0,4) y la BRDF con valores predeterminados de 0,0 para 90º y una* curve shape *de 5,0 y con un valor, para 0º, según se indica (de izquierda a derecha y de arriba abajo: a) 0,0, b) 0,5, c) 1,0. La (d) se ha obtenido con valores BRDF 1,0, 0,0, 2,5 (10º, 90º,* curve shape*).*

rado (RGB 0,65, 0,72, 0,72 en las figuras de ejemplo).

2 Cambiar los dos valores principales del grupo *Reflection*, *Reflectivity* y *Glossiness*. Las figuras adjuntas muestran resultados característicos. Si el valor de *glossiness* es inferior a 1,0 habrá que aumentar también el valor de *Glossy samples* para reducir el ruido.

3 Cambiar los valores de la curva BRDF y comprobar también como afectan al reflejo: si se aumenta el parámetro correspondiente a 0º la reflectancia será máxima y no quedará afectada por la dirección de la

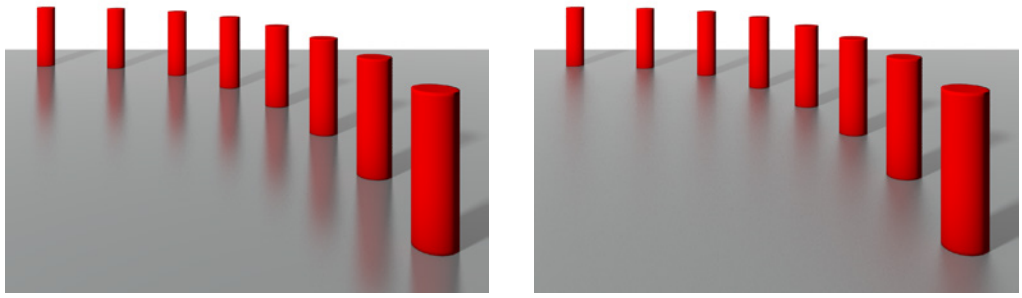


Figura 4.11 Reflejos sobre un plano. Variaciones del parámetro glossines *(para una relación* Diffuse level/Reflection level *de 0,6/0,4 y la BRDF con valores predeterminados). Valores de* glossiness *(con 16* samples*): a) 0,40, b) 0,15.*

vista. Las figuras adjuntas también muestran diferencias características.

Reflejos anisotrópicos

En el capítulo 1 ya he resumido las características generales de las reflexiones anisotrópicas. Como se recordaba allí, una propiedad anisotrópica es una propiedad que no se produce por igual en todas direcciones, al contrario de lo que ocurre con una propiedad isotrópica. En el caso de las reflexiones, esta asimetría está producida por la preponderancia de resaltes o huecos en una determinada dirección, lo que puede ser debido a todo tipo de causas: el proceso de fabricación (como ocurre con los tejidos), el proceso de acabado (como ocurre con la madera o el metal bruñido), las propias características del material (como ocurre con ciertas maderas), el desgaste, etc.

La explicación de estas características es relativamente fácil con la ayuda de una lupa o un microscopio, o bien analizando ejemplos similares pero que se producen a una escala mayor. Cualquier superficie que presente alteraciones regulares en una dirección hará que el reflejo se repita en esa dirección. Si las irregularidades son muy amplias, como ocurre con las ondulaciones del agua, el reflejo se interrumpe a intervalos regulares y podemos apreciar claramente las razones del fenómeno. Si las irregularidades son muy pequeñas solo percibimos que el reflejo se alarga en una dirección. En este caso no percibimos las causas sino sus consecuencias visuales. Es decir, que cualquier superficie que, sea por sus características naturales, sea por el tratamiento a que ha sido sometida, presente rugosidades predominantes en una determinada dirección, tendrá propiedades anisotrópicas que se manifestarán principalmente en los reflejos.

Las imágenes y esquemas de la figura 4.12 resumen lo dicho hasta aquí. La figura (*a*) muestra una fotografía de un metal bruñido, la figura (*b*) muestra una fotografía de un reflejo sobre el agua. La figura (*c*) muestra una imagen de un cilindro formado por varios anillos superpuestos donde puede apreciarse como este efecto depende de la geometría de la superficie. Y la figura (*d*) resume los ejemplos anteriores con un esquema de lo que está ocurriendo en estos casos y donde puede apreciarse cómo el efecto del reflejo se alarga pues el ángulo correspondiente al punto donde se produce este reflejo va aumentando.

Se puede simular la anisotropía de dos modos principales: *a*) mediante realces virtuales con *shaders* anisotrópicos, *b*) mediante mapas que simulan relieves en una determinada dirección o que aplican mapas específicos que simulan distorsiones en una determinada dirección. El primer modo se explica en este

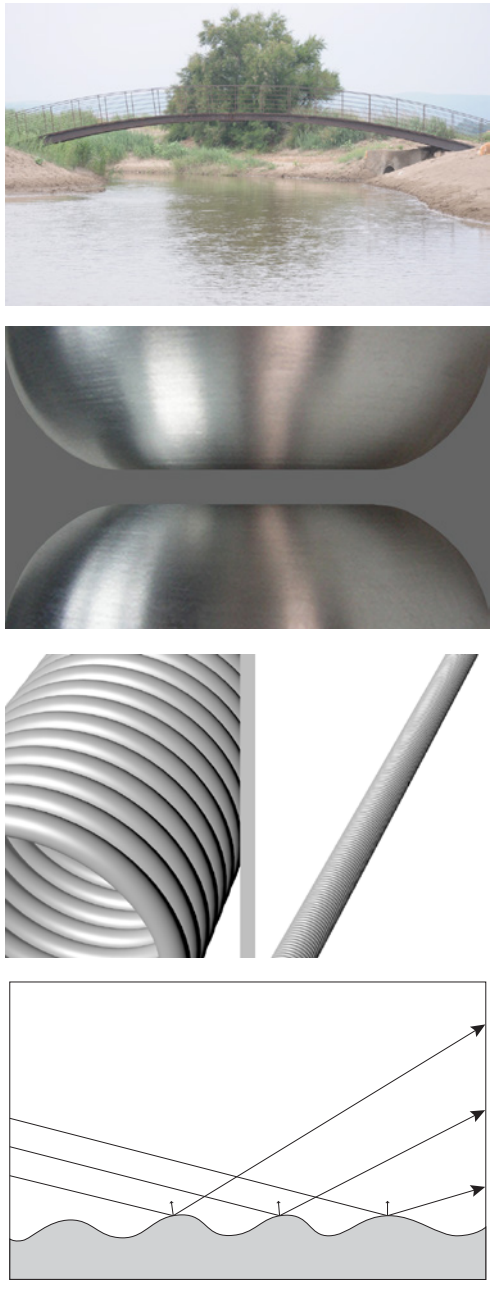


Figura 4.12 Ejemplos de superficies con reflejos anisotrópicos: a) agua, b) metal bruñido, c) cilindro formado por anillos superpuestos, d) esquema de recorrido de los rayos reflejados en los ejemplos anteriores.

apartado. El segundo modo, dado que requiere el uso de mapas, se explicará más adelante, en el apartado sobre mapas de reflexión.

En el capítulo 2 ya he introducido los principales *shaders* que computan reflexiones anisotrópicas: los de Poulin y Fournier (1990), Ward (1992) o Ashikhmin y Shirley (2000).

Los *shaders* anisotrópicos pueden ser específicos, con este mismo nombre, o estar integrados en materiales avanzados, lo que resulta preferible en la gran mayoría de los casos. Aunque la ayuda de los programas proporcionan escasa información sobre los métodos utilizados, debido entre otras cosas a que a menudo se combinan o se ajustan con métodos internos, parece que uno de los más utilizados es el de Ward. En mental ray con 3ds Max hay poca información aunque probablemente el *shader* que se utiliza con el material *Arch&Design* es una adaptación del de Ward. En mental ray con Maya se utiliza de modo explícito pues Maya tiene dos versiones del *shader* de Ward: *mib_illum_ward* y *mib_illum_ward_deriv* (que se utiliza principalmente con superficies nurbs).

Parámetros de reflejos anisotrópicos

Los parámetros disponibles cuando se utiliza el material *Arch&Design* están en *Main Material Parameters*, en el grupo *Anisotropy*. Hay dos parámetros, *Anistropy* y *Rotation*.

Con el primero, ***Anisotropy***, se controla la intensidad de la anisotropía. La anisotropía máxima se da con 0,01. Entre 0,01 y 0,1 los resultados se modifican con rapidez. Entre 0,1 y 1,0 la intensidad del efecto disminuye más lentamente. Con 1,0 no hay anisotropía. De 1,0 a 10,0 hay una ligera variación que alarga la anisotropía en sentido opuesto.

Con el segundo, ***Rotation***, se controla la dirección de la anisotropía. Este parámetro proyecta valores de 0,0 a 1,0 en el rango 0º a 360º. Esto quiere decir que con 0,0, 0,5 y 1,0 obtendremos la misma orientación. Y con 0,25 o 0,75 el efecto girará 90º. Este parámetro, *Rotation*, equivale al parámetro *orienta-*

tion de los materiales estándar de 3ds Max que modifica igualmente la dirección de la anisotropía.

Además, hay un tercer parámetro que es propiamente una alternativa entre utilizar "Automatic" o utilizar "Map Channel". La opción predeterminada es la primera, que aplica la rotación a las coordenadas locales del objeto. Si se escoge la segunda opción, la anistropía puede ligarse a un canal determinado en el caso de proyecciones múltiples. En general, no será necesario cambiar la opción predeterminada.

Pueden comprobarse los efectos de la anisotropía en el editor de materiales, tanto en una esfera como en un cilindro (cambiando el tipo de muestra). Para un cilindro, la intensidad máxima del efecto, con valores predeterminados del material *Arch&Design* se obtiene con 0,04.

Reflejos sobre un cilindro

Los ejemplos que siguen se basan en una escena como la de la figura 4.13 que incluye un plano (de unos 10 x 10 m) y un cilindro sobre este plano (de 42 cm de radio y 84 de altura, con 64 segmentos en los lados y 2 en la parte superior). He editado la malla para modificar los segmentos del borde superior que se han achaflanado ligeramente (0,8 cm). También he añadido un grupo de suavizado a todos los polígonos excepto los de la base (para suavizar las aristas superiores). El plano base tiene asignado un material *Arch&Design* mate con un mapa procedural de tipo *checker* con los colores que se pueden apreciar en la figuras citada. Esto nos servirá para comprobar los reflejos.

1 Crear un sistema de luces como el del ejemplo anterior: una luz primaria, *mrArea spot*, que ilumine el cilindro desde la izquierda y con unas dimensiones tales que se creen sombras de bordes suaves. Y una luz secundaria que no arroje sombras y de baja intensidad, en posición opuesta, que suavice las sombras de la primaria.
2 Asignar al cilindro un material Arch&Design con las características siguientes:
Diffuse level 0.1. *Color* RGB 0,55, 0,60,0,65.
Reflection: 0,9, 0,7, 16 (*reflectivity*, *gloss*, *samples*). *Metal*.
3 Configurar el cálculo con el esquema habitual recomendado en el capítulo 3 (FG, de baja calidad para pruebas y de media calidad para el final). *Render*

El resultado será muy pobre. Esto es debido a dos razones. Primero a las insuficiencias del material y segundo a que no tenemos luces adecuadas. Para remediar lo segundo, hacer lo siguiente:

4 Crear una luz focal que apunte al mismo lado que la luz anterior, pero dirigida desde abajo y desde un lado hacia el cilindro para que el reflejo llegue a la cámara. Será más fácil conseguirlo con el comando *Align Highlight*. Luego crear otra similar, de me-

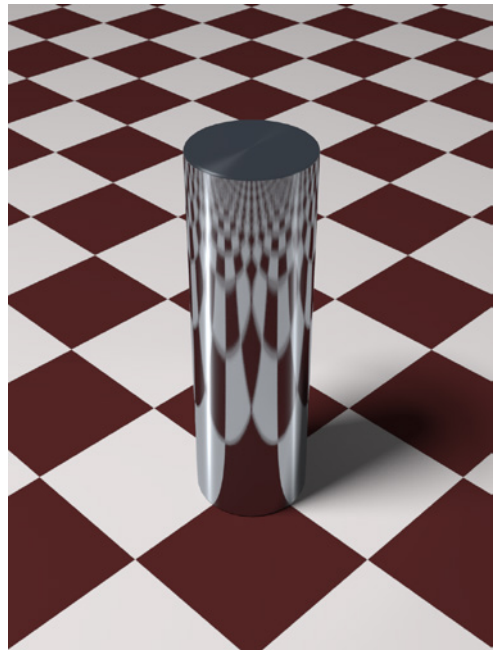


Figura 4.13 Reflejos anisotrópicos. Reflejos sobre un cilindro.

nos intensidad, que cree resaltes especulares en el otro lado.

5 Seleccionarla y con el menú contextual desmarcar su propiedad “afectar al color difuso” (*BDR/Affect diffuse*). De este modo la luz no afecta a la iluminación global sino que solo contribuye al reflejo especular. Y a la inversa, seleccionar la luz primaria y desmarcar la opción *Affect specular*.

6 Ajustar las intensidades de todas las luces según lo que interese. En la figura citada se han utilizado las siguientes:
Primaria: 0,9 (sin especularidad)
Secundaria: 0,1 (sin especularidad ni sombras)
Refl1 (izquierda): 0,3.
Refl2 (derecha): 0,1.
Refl3 (desde atrás): 0,2.

7 Editar el material y hacer los siguientes cambios:
Anisotropy: 0,1
BRDF: 0,85, 1,0, 3,0 (0º, 90º, curva)

Experimentar con variantes del material y diferentes valores de anisotropía. Más adelante veremos otras alternativas y efectos más sofisticados usando mapas.

Valores de reflectancia para diferentes materiales. Conservación de la energía

Cuando se utilizan materiales avanzados y sistemas de iluminación avanzada, que computan los rebotes de la luz sobre la superficie de los materiales y, por tanto, la contribución de estos a la iluminación global de la escena, los valores de reflectancia deben corresponderse con valores de reflectancia reales. Para evitar distorsiones y efectos falsos conviene tener en cuenta lo que sigue.

Una pared blanca no refleja más del 85 % de la luz que recibe. Y un objeto muy blanco y muy reflectante no sobrepasará el 90 %. Asignar a un objeto un valor de 100 % es un error pues distorsionará los cálculos. En general, los máximos no deberán superar valores de rgb 0,85, 0,85, 0,85 (217, 217, 217 en un rango de 0 - 255).

Los valores de reflectancia se muestran en algunos programas. Por ejemplo, en el editor de materiales de 3ds Max puede aparecer algo así como “Reflectance Avg: 50 % Max: 50 %” (valores medios y máximos de reflectancia) y “Transmittance Avg: 0 % Max: 0 %” (valores medios y máximos de transmitancia). Conviene acostumbrarse a echar un vistazo a esta información para recordar qué valores estamos manejando.

El rango de reflectancias varía con el color del material, pero esta variación está limitada según el tipo de material. Muy aproximadamente podemos decir que se limitan a los rangos siguientes:

20 % a 50 % Madera
50 % a 70 % Madera barnizada
20 % a 70 % Cerámica, piedra, telas
30 % a 70 % Papel
30 % a 80 % Plástico
30 % a 90 % Metales, pintura

En la mayoría de estos casos la reflectancia es dispersa por lo que se debería reducir el valor de *Glossiness*. Y aumentar el número de muestras, el parámetro *Glossy samples*, para prevenir defectos, manchas, etc, al menos, a 16 para resultados medios y valores tales como 24, 32, 48 o incluso más, para resultados finales, aunque convendrá anotar los tiempos de cálculo. Si esto no es suficiente y siguen apareciendo defectos o los tiempos son excesivos, utilizar la opción *Highlights+FG*. El resultado es menos nítido y los objetos cercanos no se reflejarán con claridad pero será bastante más rápido y con menos defectos notorios. Otra alternativa que ya he citado es utilizar *Fast glossy interpolation* que es más rápida aunque menos precisa. Esta opción es más adecuada para grandes superficies planas (como pavimentos) pues no hace el cálculo a partir de superficies concretas sino del conjunto de la imagen. No es adecuada para superficies curvas. Tampoco es adecuada si se quieren reflejos realistas: será preferible aumentar los valores de *glossy samples*.

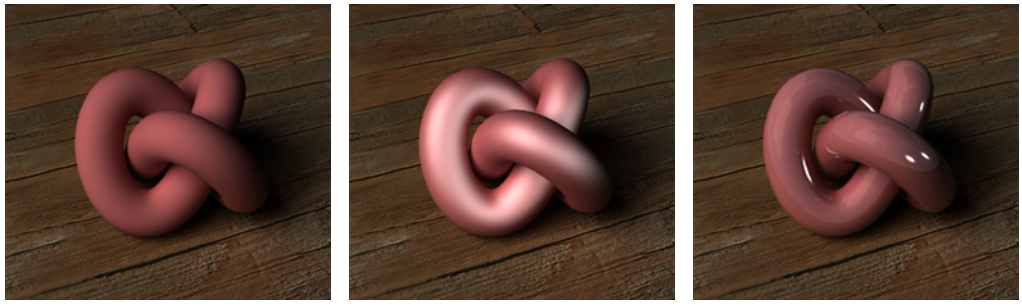


Figura 4.14 Reflejos sobre diferentes materiales: a) *Plástico mate;* b) *Plástico brillante;* c) *Cerámica.*

Figura 4.15 Reflejos metálicos: a) *Aluminio anodizado;* b) *Aluminio lacado;* c) *Aluminio lacado brillante.*

Los metales se caracterizan por reflejar la luz en todos los ángulos, a diferencia de los dieléctricos que tienden a reflejar muy poco, cerca de los 0º (vistas perpendiculares). En el caso de los metales debe modificarse la curva para simular este efecto. Puede hacerse manualmente pero es más práctico y quizás más consistente hacerlo por medio del IOR, activando esta opción en la sección BRDF. Los valores adecuados, aunque sean absurdos desde un punto de vista puramente físico, deberán estar por encima de 8,0 (que supone una reflexividad de alrededor de un 60 % para 0º). Las plantillas utilizadas para metales por *Arch&Design* utilizan un valor de 25 (que da una reflexividad de alrededor del 90 % para 0º) para los materiales cromados y de hasta 40 para el cobre.

Todos los parámetros anteriores se pueden utilizar libremente. Sin embargo, conviene tener presente, como decía, que un *shader* que pretenda llevar a cabo una simulación física correcta debería mantener la conservación de la energía. Esto quiere decir ni más ni menos que no puede ser que, por ejemplo, refleje y transmita toda la energía que recibe pues la suma normalizada debe mantenerse igual a 1. Esto se ajusta automáticamente en algunos programas pero no en todos, y el usuario debe preocuparse de revisar estos valores de modo que el resultado final se mantenga dentro de unos límites correctos.

Esto también afecta en principio a las reflexiones Fresnel que pueden estar configuradas de tal modo que las reflexiones en una dirección sean muy superiores a las reflexiones en otra dirección, algo que también puede perderse de vista.

Si se especifican valores máximos (1,0) de color difuso, reflectancia y transparencia, la suma será superior a 1,0 lo que es incorrecto físicamente y dará resultados distorsionados. Una ventaja importante de utilizar el material *Arch&Design* de mental ray es que corrige internamente los valores para que la suma no rebase la unidad. Esta es también una razón para no utilizar los viejos *shaders* de mental ray (DGS, Dielectric, Glass, etc.)

pues, aunque algunos de ellos incorporaban parámetros adicionales, la mayoría de sus propiedades se han incorporado de un modo más compacto y más manejable al material *Arch&Design*.

Si se quiere llevar a cabo una simulación precisa puede medirse la reflectancia, si se cuenta con un luxómetro o un fotómetro y un patrón blanco (una hoja de cartón con un blanco normalizado que puede adquirirse en una tienda de fotografía). Todo lo que hay que hacer es tomar una medida del material en una dirección y otra del patrón blanco sin modificar la dirección (ni las condiciones de iluminación). Luego dividir el primer resultado por el segundo. Si, por ejemplo, el primero ha dado un valor de 24 cd/m^2 y el segundo 58 cd/m^2 la reflectancia es 0,41 (24/58). A partir de este valor, el color que asignemos al material, si se quiere que represente adecuadamente el material real, no debería ser muy distinto de un HSV con V = 0,41 (equivalente a 104 sobre una escala de 255).

En la práctica no será necesario hacer mediciones precisas y es de suponer que un ojo educado sabrá calibrar si una determinada configuración da resultados adecuados o no. Las figuras 4.14 y 4.15 muestran diferentes configuraciones de un nudo reflectante para simular materiales corrientes. En el último capítulo (apéndice 6), en la sección sobre metales, se amplían estos ejemplos.

Transparencias

Esta sección abarcaría en principio casos muy diversos y, en algunos casos, muy complejos, pues las características de la escena son tan importantes o más que en el caso de los reflejos. Lo primero que habrá que controlar, por consiguiente, es el escenario general: si no hay objetos que se vean a través de otros objetos o que se reflejen sobre estos, no tendrá sentido la simulación.

Sin embargo, a diferencia de los reflejos, que de uno u otro modo aparecen en muchas escenas de interiores, los fenómenos más complejos de transparencias son más bien inhabituales. Lo más corriente son las transparencias simples, que son muy fáciles de simular, y las transparencias con reflejos, que tampoco implican grandes dificultades.

En los apartados siguientes analizarémos los casos principales que se pueden presentar en la práctica. Los valores dados están referidos en todos los casos al material *Arch&Design* de mental ray pero son similares en otros programas.

Parámetros básicos

Con el parámetro principal, *Transparency* (0,0) se especifica el grado de transparencia. *Color* afecta al color de la transparencia y, si se añade un mapa, funciona igual que con los reflejos: las zonas blancas mantendrán la transparencia, las negras lo anularán y las grises tendrán un efecto intermedio. Veremos algún ejemplo de aplicaciones de mapas en el capítulo siguiente. Los parámetros *glossiness* (1,0) y *glossy samples* (8,0) tienen el mismo sentido que con reflexión pero en este caso aplicado a la dispersión de la transparencia. Otro tanto cabe decir de *Fast interpolate*. Por último, IOR (*Index Of Refraction*) define el índice de refracción.

Si se quiere especificar con precisión los valores convendrá obtenerlos de los fabricantes. Y si se quiere simular el efecto de vidrios dobles con diferentes valores de transmitancia hay que multiplicar los valores correspondientes. Por ejemplo, si un vidrio transmite 0,85 y otro 0,75 el total será 0,64 (0,85 x 0,75).

Transparencias simples

El caso más elemental es el de un panel de vidrio incoloro visto a distancia y con un ángulo no muy alejado del frontal.

En este caso tan elemental, la solución más sencilla y más recomendable es no hacer nada. Es decir, no colocar ningún objeto que simule el vidrio. Un vidrio corriente en estos casos es invisible. Y al igual que no nos preocupa que los objetos virtuales que creamos en una situación no tengan peso y floten

tampoco tiene porqué preocuparnos que estén representados por un vacío.

Si nos interesa que el color del vidrio sea visible la solución también es muy sencilla pues todo lo que hay que hacer es cambiar el color predeterminado para la transparencia (blanco) por el color que nos interese. Para un caso como este, donde se supone que estamos representando un vidrio muy delgado también convendrá reducir el valor de parámetro IOR a 1,0 para que no haya distorsión y para no retardar innecesariamente el cálculo. La figura 4.16 ilustra estos casos elementales.

Las figuras 4.17 a 4.19 ilustran casos algo más complejos con una escena en la que hay un plano base, un pequeño muro al fondo y un muro inclinado en primer plano cuyos valores de transparencia iremos modificando.

Las primeras ilustran el caso elemental en que el muro inclinado recibe valores de transparencia simple y las únicas diferencias provienen de modificar ligeramente el índice de refracción. La configuración que se ha utilizado es la siguiente. Tipo de material: *Arch&Design*. *Diffuse color*: mismo color que transparencia. En principio, sin reflexión. Grupo *Refraction*: valores indicados en los pies de las figuras citadas, es decir, *Transparency* entre 1,0 y 0,75, *Glossiness* entre 1,0 y 0,45, *Color* de transparencia gris verdoso claro, *IOR* entre 1,0 y 1,5.

Si el valor de IOR es alto y el objeto tiene cierto grosor hay que comprobar que, en el

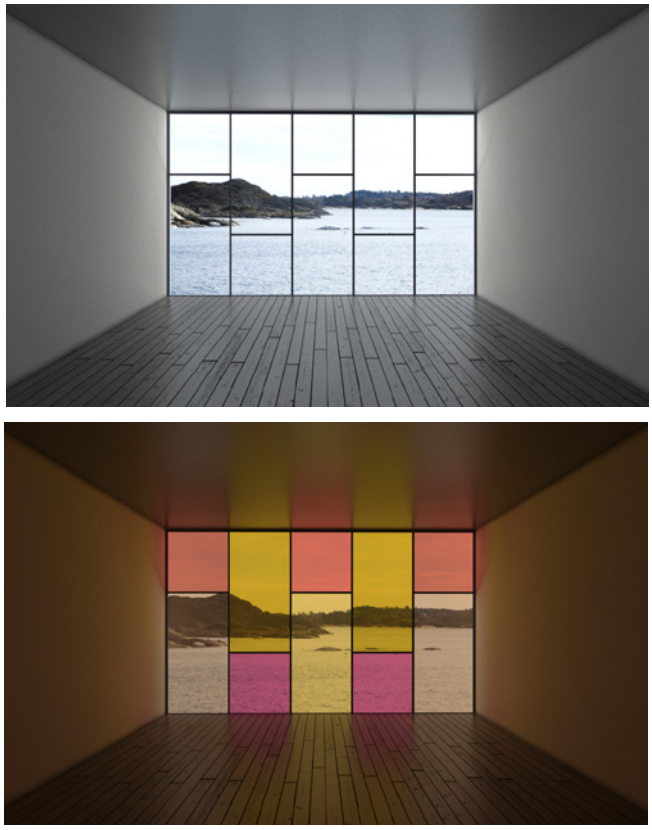


*Figura 4.16 Transparencias. En vistas frontales, en las que ni los reflejos ni las refracciones son determinantes, lo más sencillo es eliminar el vidrio (*a*) o asignar un color de transparencia a los objetos que simulan el vidrio (*b*).*

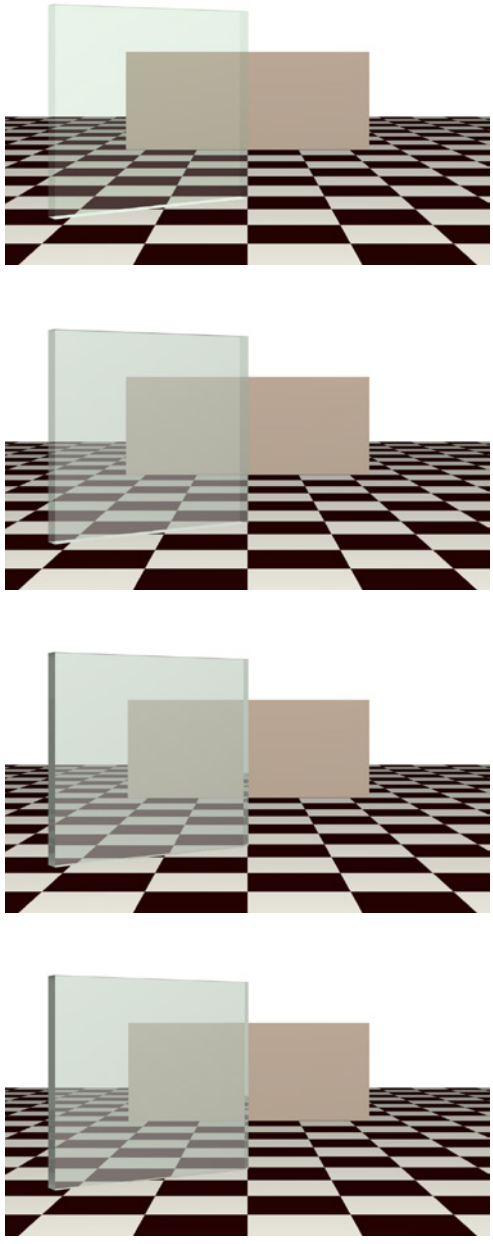


Figura 4.17 Transparencias: a) *Grupo* Refraction*: 1,0, 1,0, 1,0 (*transparency, gloss, IOR*);* b) *Igual que la anterior con* transparency *en 0,75;* c) *Igual que el anterior con* IOR *1,5, d) Igual que el anterior con* IOR *2,0.*

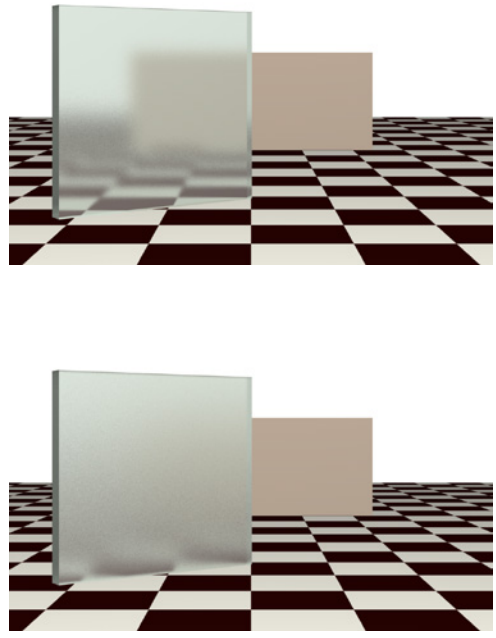


*Figura 4.18 Transparencias. Variación de la dispersión (*glossiness*):* a) *0,8, 0,75, 1,0 (*transparency, glossiness, IOR*),* b) *0,8, 0,45, 1,0.*

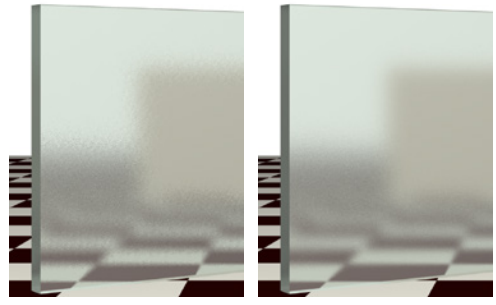


Figura 4.19 Transparencias: glossines *0.75 con:* a) *4* samples, b) *24* samples.

grupo *Advanced rendering options/ Advanced transparency options* está marcada la opción *solid* (en lugar de *thin*) y desactivada la opción *Back face culling*.

Transparencias con dispersión

Si el elemento transparente no está perfectamente pulido, la transparencia se dispersa de diversos modos que dependerán del tipo de material y del tipo de tratamiento. Esto se puede simular de dos modos que por lo general se usan en combinación: por medio de parámetros básicos y por medio de mapas. En esta sección veremos solo los parámetros básicos y retomaremos el tema después de presentar las técnicas generales de proyección de mapas.

Para controlar la dispersión a partir de los parámetros básicos hay que modificar el parámetro *glossiness*, de modo similar a lo que hacíamos con reflejos. Un valor máximo de *glossiness* (1,0) representa una transparencia perfecta. Un valor mínimo (0,0) una dispersión perfecta, es decir que no hay transparencia. Entre estos dos extremos se puede especificar un valor de *glossiness* que, por lo general, estará entre 0,3 y 0,7 pero que dependerá de cada caso.

Si hay dispersión convendrá tomar más muestras para que la representación sea más precisa. Si el número de muestras, *samples*, es bajo (en torno a 8, que es el valor predeterminado, o menos), el cálculo será más rápido pero el resultado será imperfecto, aparecerá ruido. Si el número de muestras es alto, el resultado será mejor pero a costa de más tiempo de computación. Como en casos similares, lo que se debe hacer es comenzar por valores bajos y aumentar este parámetro tan solo en las salidas finales. La figura 4.19 muestra un par de ejemplos que pueden dar alguna indicación de estas diferencias aunque solo se apreciarán bien con alta resolución y un buen monitor.

También se puede utilizar la plantilla "frosted glass" incluida con el material *Arch&Design* que da resultados similares a los anteriores. Utiliza los siguientes valores para el grupo *Refraction*: 0,8, 0,5, 12 (*transparency*, *glossiness*, *samples*), con un IOR de 1,5 y algo de *Translucency* (activado con un peso de 0,2). Véase también los parámetros de *Advanced options* que afectan al color, algo más azulado que en los ejemplos adjuntos.

Transparencias con reflejos

Un vidrio observado con un cierto ángulo, combina transparencias con reflejos. El peso de uno u otro fenómeno, es decir, si hay más transparencia o más reflejo, depende de la luminosidad de los objetos que se transparenten o se reflejen.

Si miramos las ventanas de un edificio en un día luminoso, las ventanas se comportarán casi como un espejo y veremos las casas y el paisaje que las rodea. Pero si miramos las mismas ventanas al atardecer, a nada que haya una pequeña luz encendida en el interior al que da la ventana, lo que veremos principalmente es ese interior. Ahora bien, entre estos dos extremos hay infinitos casos en los que se combinan de múltiples modos transparencias y reflejos.

La combinación de transparencias y reflejos se puede controlar de un modo bastante completo con el material *Arch&Design*. El siguiente ejemplo, que corresponde a la figura 4.20, ilustra el procedimiento básico.

La escena consiste en dos muros, situados frente a frente, y un pavimento. La cámara está dirigida al primer muro que incluye una ventana y, tras la ventana, un recinto interior en cuyo muro lateral derecho (desde la cámara) hay unas letras minúsculas de color rojo. El segundo muro, no visible desde la cámara, tiene colgadas de su superficie unas letras mayúsculas de color azul. De este modo, podemos comprobar el reflejo (que mostrará las letras mayúsculas azules del muro no visible desde la cámara) y la transparencia (que mostrará las letras minúsculas rojas del recinto interior) o bien la superposición del reflejo y la transparencia (que mostrará los dos grupos de letras mezclados).

A estos tres resultados podemos llegar de dos modos. El primer modo, consistiría en crear luces que iluminen solo lo que nos interesa (el reflejo, la transparencia o ambos). El segundo modo, más realista, consistirá en crear luces exteriores e interiores y aumentar la intensidad de unas u otras. Como ocurre en la realidad, si la luz exterior es dominante

solo veremos el reflejo y, si la luz interior es dominante, solo veremos la transparencia. Y en una situación intermedia (que también dependería de la dirección de la vista) veremos la superposición de ambas.

En las imágenes de la figura 4.20 se ha utilizado tan solo el segundo modo. Porque es más realista, porque es suficientemente ilustrativo de lo que aquí interesa y, en tercer lugar, porque la única ventaja del primer modo, más artificioso, es que permite mayor control de los resultados pero a costa de una serie de recursos técnicos que se explican en el libro sobre simulación de la Iluminación pero que que no caben en éste.

Figura 4.20 Transparencias con reflejos: a*) solo reflejo,* b*) solo transparencia,* c*) reflejo y transparencia.*

Para utilizar este modo, crear un sistema de luz diurna, con el Sol dirigido hacia el muro no visible, iluminando las letras mayúsculas azules y una luz fotométrica que ilumine el interior dirigida hacia las letras minúsculas rojas. La configuración de intensidades la daré más adelante.

1 Crear tres materiales, de tipo *Arch&Design*, uno solo reflejante, “vidrio1R”, otro solo transparente, “vidrio2T” y otro refejante y transparente, “vidrio3RT”. Ajustar sus parámetros como sigue:
“vidrio1R”. *Reflection* 1,0, 1,0 (*Reflectivity*, *Glossiness*), BRDF 0,85, 1,0, 5,0 (0º, 90º, curva), *Transparency* 0,0.
“vidrio2T”. *Reflection* 0,0, *Transparency* 1,0, 1,0.
“vidrio3RT”. *Reflection* 0,5, 1,0, BRDF 0,85, 1,0, 5,0, *Transparency* 0,75, 1,0.

Podríamos utilizar solo el tercer material pues el efecto depende básicamente de la intensidad de las luces exteriores e interiores, como ocurre en la realidad. Pero el efecto será más controlable con estos tres materiales.

2 Asignar a la ventana el material solo reflectante (“vidrio1R”). Desactivar la luz fotométrica y activar el sistema de luz diurna con los valores predeterminados (intensidad del sol y *sky* 1.0). Ajustar el control de exposición a EV:14. Activar *Final gather* con valores medios y la calidad (*antialiasiang*) también a valores medios. Así se obtendrá un resultado similar al de la primera imagen (*a*) de la figura 4.20.
3 Asignar a la ventana el material solo transparente (“vidrio2T”). Activar la luz fotométrica con una intensidad de unas 20.000 cd y reducir los valores del sistema de luz diurna como sigue: desactivar el sol y reducir la intensidad de *sky* a 0,01. Aumentar la exposición reduciendo el valor de EV a 10,0. Dejar el resto de la configuración

(*FG*, *antialisaing*) como antes. Así se obtendrá un resultado similar al de la segunda imagen (*b*) de la figura 4.20.

4 Asignar a la ventana el material refectante y transparente ("vidrio3RT"). Activar la luz fotométrica con la misma intensidad de unas 20.000 cd y reducir los valores del sistema de luz diurna como sigue: reducir la intensidad del sol a unos 0,02 y la de *sky* a unos 0,2. Reducir un poco la exposición haciendo EV igual a unos 12,0. Dejar el resto de la configuración (*FG*, *antialiasing*) igual. Así se obtendrá un resultado similar al de la tercera imagen (*c*) de la figura 4.20. Si, a partir de esta configuración, aumentamos la intensidad de la luz externa, el reflejo dominará sobre la transparencia. Y viceversa. Es decir que obtendremos resultados similares a los de las dos primeras figuras pero menos nítidos.

Además de los métodos resumidos, si se prefiere, se pueden utilizar materiales ya preparados como los proporcionados por Autodesk, que incluyen vidrios sencillos que funcionan bastante bien aunque las posibilidades de ajuste son más limitadas. No incluyen refracción y se pueden aplicar a objetos sin espesor, como planos o caras poligonales. Tienen muy pocos parámetros y solo permiten cambiar el color y el grado de reflectancia. El parámetro *Sheets of glass* está pensado para ventanas con doble vidrio, para compensar la transparencia en estos casos.

Transparencias con refracción. Parámetros adicionales

En el primer apartado, transparencias simples, ya he introducido los parámetros de refracción pues, en general, no hay transparencia sin refracción, y un rayo de luz, al atravesar un objeto transparente se desvía en mayor o menor grado, una desviación que viene dada por el índice de refracción del material. Pero en la mayoría de los casos de interés práctico, como también he dicho, esto tiene escasa importancia pues los vidrios que se utilizan en la construcción son muy delgados y planos y el efecto es imperceptible.

Pero hay objetos transparentes en los que la refracción juega un papel importante debido a que el espesor es mayor. En estos casos, cuando el rayo atraviesa el cuerpo, su intensidad se atenúa, con el resultado de que el color se modifica y pierde intensidad a medida que atraviesa el cuerpo. En el capítulo 1 he incluido los índices de refracción de materiales corrientes. Una referencia fácil de recordar es que el agua tiene un índice de 1,33, el vidrio está en torno a 1,5 y el diamante alrededor de 2,4.

Si se utiliza el material *Arch&Design* de mental ray hay un medio para simular este efecto de pérdida de intensidad con el espesor que es utilizar los controles que se encuentran en la sección *Advanced rendering options/ Refraction*. En este grupo hay dos parámetros principales: "distancia máxima" y "color a la distancia máxima". Si no se especifica un color se asigna el color negro a esta distancia máxima. Y si se especifica un color, este será el color que recibe el cuerpo a la distancia especificada. A la mitad de distancia la influencia del color será la mitad y, al doble, la influencia será doble.

Si se fija el valor de distancia máxima a un valor relacionado con el grosor del cuerpo que estamos simulando, y se especifica el color que debe alcanzarse a esta distancia, el efecto es más convincente.

Este control está particularmente indicado para casos como la simulación del agua. Para

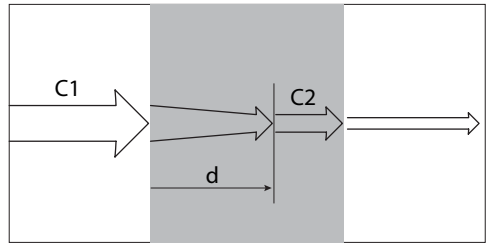


Figura 4.21 Transparencias con refracción. El color se atenúa al atravesar un cuerpo con espesor. El parámetro "color al max distance" se puede utilizar para modificar el color C1, en función de la distancia (d), que se convierte en C2.

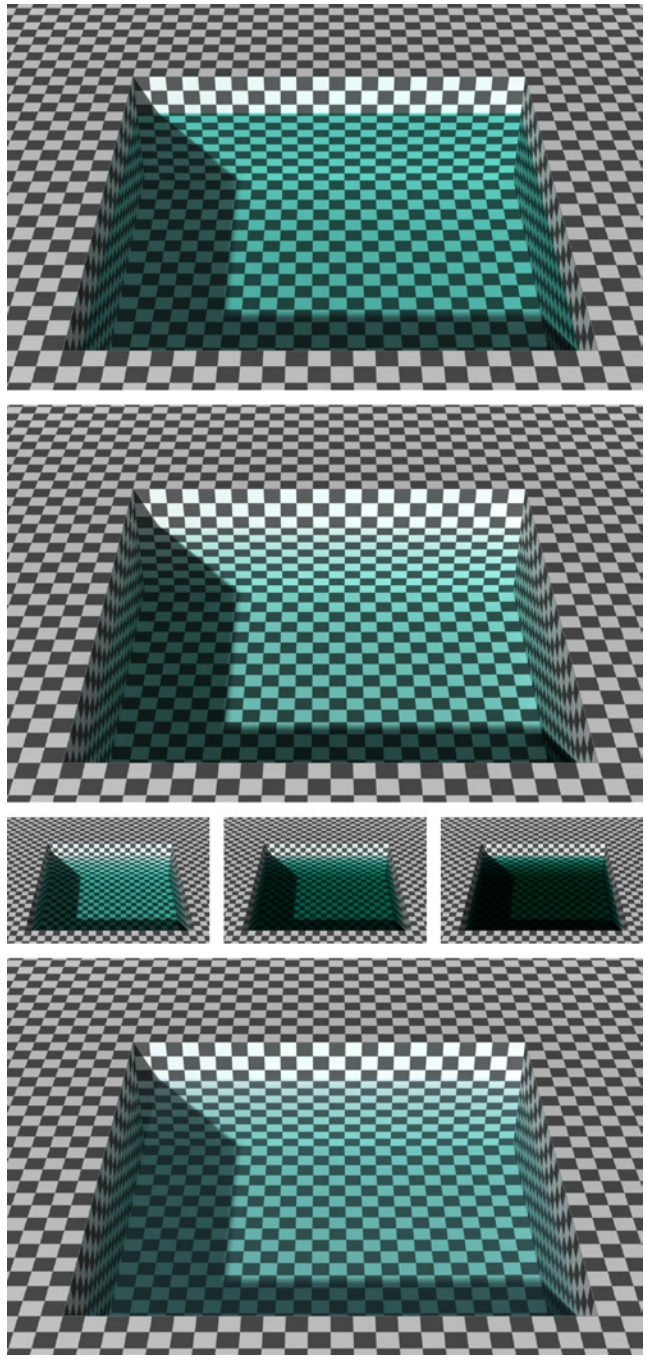


Figura 4.22 Transparencias con refracción: a) *color dado por el parámetro* transparency color, b) *Color dado por el parámetro* color at max distance *con un valor d=50 cm,* c) *resultados para d=25, d=10, d=5.*

que la simulación sea completa y reproducir los diversos efectos que tienen lugar en un medio como el agua, se necesitan recursos adicionales, tales como mapas de relieve y caústicas, como veremos más adelante. Pero el primer control complementario que conviene utilizar es este parámetro. Las imágenes de la figura 4.22 ilustran su uso por medio de una escena muy sencilla que representa una mini piscina de 1 metro de lado y 0,5 metros de profundidad. A 10 cm por debajo del nivel del suelo hay un plano sin espesor que es el que utilizaremos para simular el agua. Si prescindimos de los efectos debidos a los reflejos y la ondulación del agua y nos concentramos en este caso tan elemental podemos proceder como sigue (usaré los colores siguientes: "agua", HSV 0,48, 0,39, 0,85 y "cielo", HSV 0,55, 0,39, 0,95).

1 Asignar al plano un material *Arch&Design* con la siguiente configuración:
Diffuse: 0,1, "cielo" (level, color).
Transparency: 1,0 / 1,0 / "agua" (*refract*/*gloss*/*color*).
BRDF: 0,6/1,0/5,0 (0º/90º/*shape*).

Así se obtiene la imagen (*a*) de la figura 4.22, con solo transparencia y sin parámetros adicionales. El resultado es un poco falso pues la transición es demasiado rápida y el color, uniforme. Para hacer más suave la transición cambiaremos el color de la transparencia a blanco y llevaremos el color anterior, "agua", al parámetro mencionado más arriba, *color at max distance*. La distancia se puede hacer igual a la profundidad de la piscina que, en este ejemplo es de 0,5 m. El resultado ahora debería ser que la parte superior es muy transparente, sin color, y el fondo se va tiñendo de color. Las imágenes más pequeñas en la parte central de la figura 4.22, muestran los resultados obtenidos con diferentes valores.

2 Cambiar el color de la transparencia a blanco.
3 En *Advanced options*/*Refraction*, activar *Max distance*. Dar como valor de distancia 50 cm y como color "agua" (cortarlo y pegarlo del canal anterior, junto a los parámetros de transparencia).

El resultado es más interesante. Las figuras más pequeñas citadas muestran variaciones correspondientes a diferentes valores del parámetro *max distance*. Sin embargo ahora la superficie es casi invisible. Para recuperar este color podemos elevar el nivel de *diffuse* a 1,0 y bajar un poco el valor de transparencia a 0,9, como en la última imagen (*c*). Pero las mejoras vendrán realmente de completar la simulación con mapas adicionales, como veremos más adelante.

Translucidez

Una superficie translúcida permite que la luz la atraviese pero la dispersa internamente. Un caso especial de translucidez es el que presenta la piel de los animales y de los seres humanos cuando tiene poco espesor y puede ser vista a contraluz. Un ejemplo característico son las orejas que, en sus bordes, pero también en el lóbulo, si no es muy grueso, presentan un color especial, unas características que vienen dadas por la luz que las atraviesa y se dispersa. También las aletas de la nariz, en algunos casos, pueden mostrar este efecto.

La importancia creciente que tiene la simulación de caracteres humanos en la industria del espectáculo ha llevado al desarrollo de materiales que utilizan algoritmos denominados genéricamente de *Sub Surface Scattering* (dispersión subcutánea). Los parámetros de este tipo de algoritmos son una versión más compleja de los parámetros propios de la simulación de translucidez. Esto se aborda de un modo resumido en la sección siguiente, que trata de este tema.

Sin embargo hay casos más simples como los que se dan cuando la translucidez se reduce al efecto de dispersión de una superficie homogénea y muy delgada, como puede ser un panel de vidrio o de plástico o de papel o de tela, visto a distancia. Estos casos pueden simularse con los parámetros incor-

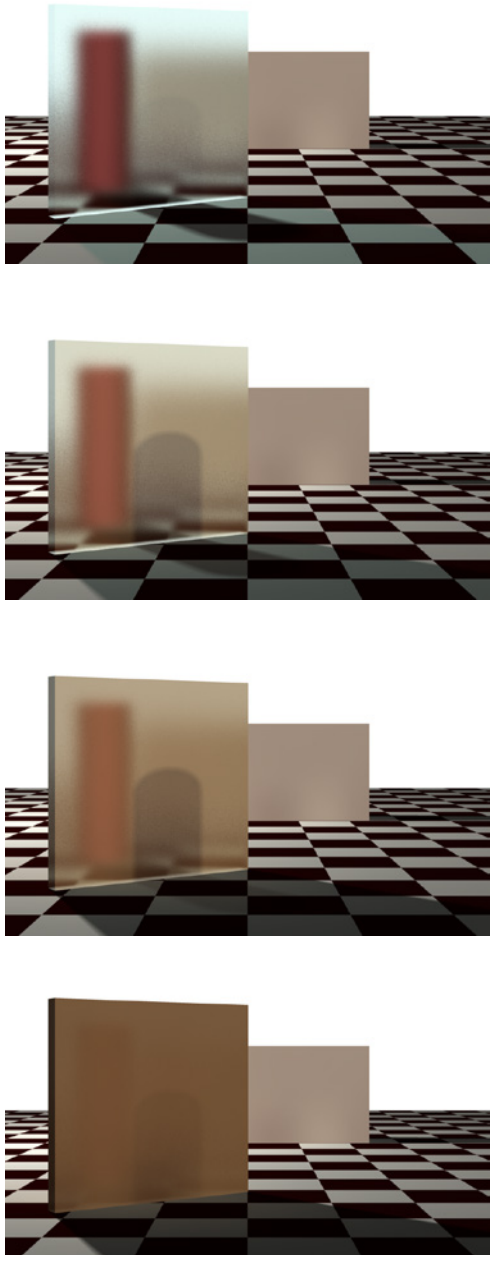


Figura 4.23 Translucidez. El panel de vidrio tiene activado el parámetro de translucidez con los siguientes valores para el parámetro weight*:*
a*) 0,05,* b*) 0,35,* c*) 0,65,* d*) 0,95.*

porados a *shaders* avanzados como el material *Arch&Design* que incluye, en la sección *Translucency*, algunos parámetros básicos que pueden ser suficientes para algunos casos simples como los que se presentan a continuación. También hay otros métodos, que veremos en otros apartados, como el uso de mapas *falloff* para simular cortinas o el uso de materiales autoiluminantes.

En las imágenes de la figura 4.23 se ha utilizado la misma escena que en el apartado anterior pero se ha incluido un cilindro rojo detrás del vidrio y una luz dirigida hacia este cilindro que proyectaría una sombra sobre el vidrio y el plano de base.

Para añadir un efecto de translucidez basta con activar la opción *Translucency*, que aparece a continuación de *Transparency*, y ajustar dos parámetros.

Con *Weight* (0,5) se especificia la cantidad de transparencia que se usa como factor de translucidez (si es 0,0 toda la transparencia es translúcida, si 0,5, la mitad). Y con *Color*, el color de la translucidez.

El material que se ha asignado al panel de vidrio en el ejemplo que se ilustra en la figura 4.23 es un material de tipo *Arch&Design*, como los anteriores, con la siguiente configuración:

Diffuse: 0,0
Reflectivity: 0,0
Refraction. 1,0 / 0,8 / 24 / RGB 0,9,1,0,1,0 / 1,5 (*Transp* / *Gloss* / *Samples* / *Color* / *IOR*)
Translucency: activado, con un color ocre (RGB 0,8, 0,6, 0,4 aprox.). Los valores de *weight* variarán tal como se indica en el pie de la figura 4.23.

El efecto de translucidez hace que la transparencia se disperse internamente, lo que produce un efecto que no es el mismo que se obtiene aumentando el parámetro de *glossiness*. Afecta principalmente a las sombras arrojadas por objetos próximos.

Sin embargo, como ya he dicho, estos efectos son sutiles y en casos más complejos será necesario utilizar materiales SSS, como los que se describen en la sección siguiente.

Sombras sobre vidrios

Aunque las sombras sobre los vidrios no serían visibles si el vidrio fuera perfectamente transparente, a menudo son visibles, sea porque el vidrio tiene algo de color o porque no está limpio. Y si el plano del vidrio está retirado, las sombras añaden profundidad al hueco por lo que es un efecto que a menudo interesa forzar un poco.

Sin embargo, si el vidrio se hace reflectante no mostrará sombras. Para solucionar este problema se puede añadir un material *Matte/Shadow* al plano del vidrio. Como este material requiere el uso de mapas de fondo y está ligado a una discusión más general sobre procesos especiales de simulación con otras finalidades, me remito a la sección correspondiente del capítulo siguiente.

Autoiluminación

La mayoría de los programas incluyen esta opción, como ya hemos visto. En el caso del material *Arch&Design* se utiliza la sección *Self illumination* que incluye parámetros bastante eficaces.

Si, en esta sección, se activa la casilla *Self illumination,* el color del material se substituye por un color, en principio blanco, que irradia desde todos los puntos de la superficie. El *Color* se controla mediante una lista colgante que presenta diversos blancos normalizados. Si se escoge la opción *Kelvin* se puede especificar el tipo de blanco por la temperatura de color. Este color también puede modificarse por medio de otro parámetro, *Filter*, que lo varía con valores dados desde el selector de color o desde un mapa, como veremos en el capítulo siguiente. El grupo *Luminance* permite controlar la intensidad de la iluminación. Si se deja la opción predeterminada, *Unitless*, los valores son similares a los del multiplicador y se basan en el rango del monitor, con el valor predeterminado (1,0) referido a un "blanco" basado en este rango. Si se cambia la opción por *Physical units* (cd/m^2) la intensidad se puede especificar en unidades fotométricas. El grupo *Glow options* tiene dos opciones. Si se mantiene activada la opción *Visible in reflections* el resplandor afecta a las reflexiones. Y si se activa la opción *Iluminates the scene (when using FG)* afecta al propio cálculo de iluminación.

La propiedad de autoiluminación se utiliza corrientemente para simular luminarias. En la mayoría de los programas de simulación las luces son invisibles: son objetos situados en

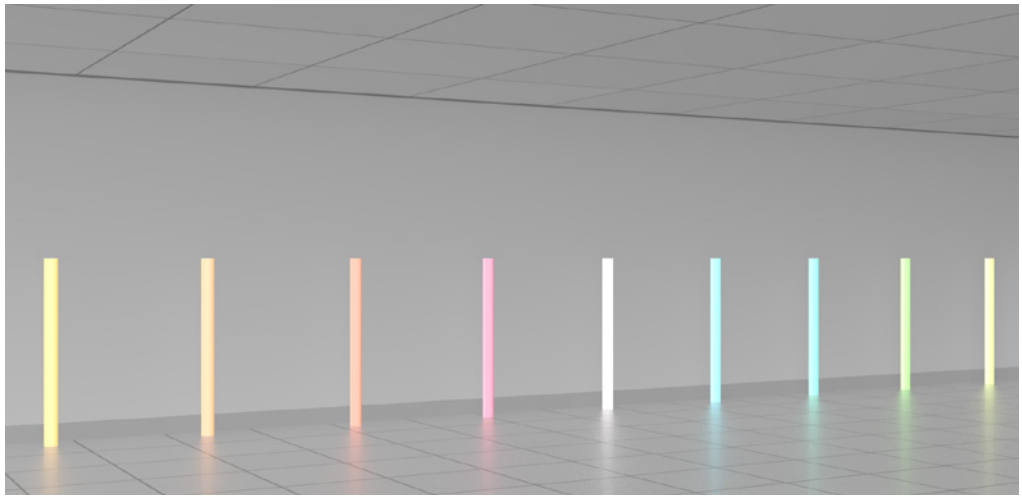


Figura 4.24 Autoiluminación. Escena con tubos de color luminosos.

una determinada posición y que emiten luz. Pero "emitir luz" quiere decir que afectan directamente a las superficies que les rodean, alterando su color en función de las propiedades de la luz y, en el caso de sistemas de iluminación avanzada, alterando indirectamente el color de otras superficies que no están en línea con la luz virtual pero a las que llegaría la luz si las superficies que esta encuentra en su camino tienen determinadas propiedades de reflexión o transparencia y estas pueden ser computadas por el sistema utilizado. Pero la luz virtual no es visible pues no es sino una serie de propiedades numéricas, incluyendo su posición, que se usan para modificar las superficies de la escena.

Dado que resultaría extraño que una determinada zona pareciese irradiar luz sin que haya ninguna fuente de luz visible, lo que se hace es crear un objeto que parezca que irradia luz y situarlo en la misma posición que la luz, haciendo que la luz excluya este objeto (pues si estuviera situada en su interior solo iluminaría el interior del objeto). Esto puede hacerse, en 3ds Max, mediante un parámetro especial incluido con las luces, *Exclude*, que permite excluir a los objetos seleccionados del cálculo de la iluminación. También puede hacerse editando las propiedades del objeto y desactivando la opción *Cast shadows*. De este modo, las luces que encuentren las superficies de este objeto no las tendrán en cuenta.

Dado que la principal característica de un objeto luminoso es que no tiene sombras pues irradia en todas direcciones, por lo general basta con hacer que el objeto sea blanco (o de un color muy claro) y que no tenga en cuenta la luz que le llegue, para que todas sus caras tengan la misma intensidad, lo que crea la apariencia de irradiación luminosa.

Hay programas que utilizan métodos de cálculo de iluminación (principalmente derivados del sistema de cálculo *path tracing*), que pueden toman en cuenta las propiedades emisoras de objetos concretos de la escena y, en este caso, basta con crear un único objeto que actúa como luminaria real. Pero incluso en estos casos puede merecer la pena trabajar con objetos dobles pues las luces virtuales tienen más controles que permiten ajustar su distribución. En muchos casos, por ejemplo, conviene ligar una luz fotométrica a un diagrama concreto de distribución por medio de archivos de tipo IES u otros.

Cuando se usan materiales avanzados, como *Arch&Design*, se puede hacer que el objeto al que se aplica el material contribuya

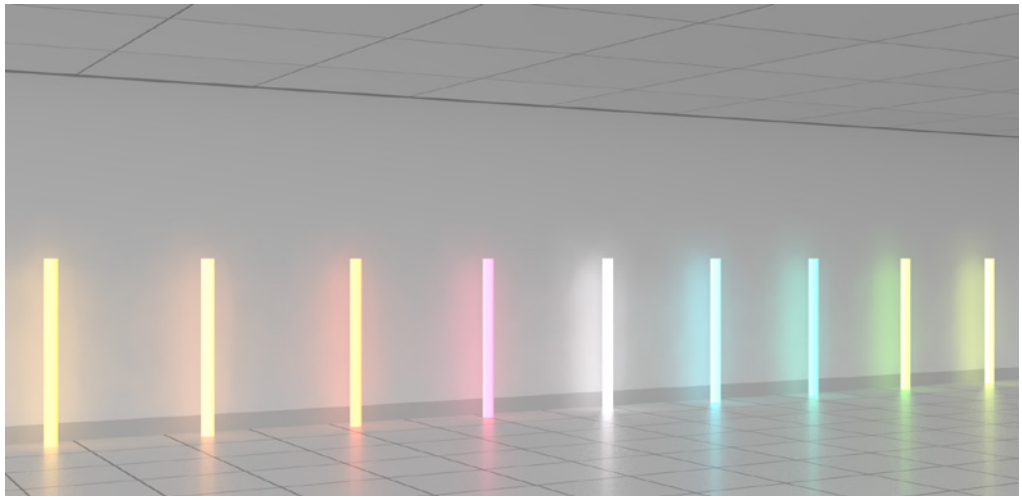


Figura 4.25 La misma escena anterior con la opción de "contribuir a la iluminación de la escena" activada.

a la iluminación de la escena. Esto facilita la simulación pero tiene limitaciones pues si la intensidad es muy alta se producen distorsiones que es difícil eliminar, como veremos en uno de los dos ejemplos que siguen. Por esta razón sigue siendo preferible ligar el objeto a una luz determinada.

Todos estos temas se tratan más extensamente en el libro de simulación de la iluminación. La simulación de objetos emisores está a caballo entre la simulación de materiales y la simulación de iluminación, por lo que es particularmente difícil de explicar de un modo independiente. En este apartado me limitaré, por tanto, a dar un par de ejemplos básicos que ampliaré en el apartado de mapas de autoiluminación. Y me remito al libro citado para ampliar el tema.

El primer ejemplo es muy simple y consiste tan solo en una escena en que se han insertado una serie de objetos autoiluminantes que se perciben como luminarias aunque en realidad no están emitiendo luz de ningún tipo. Este ejemplo sirve para ilustrar cómo, en muchos casos, la simulación depende tanto o más de expectativas psicológicas que de hechos reales.

Se trata de una escena en la que se han colocado varios cilindros que simulan tubos fluorescentes, a la manera de Dan Flavin. Los objetos iluminantes tienen asignado un material *Arch&Design*. Todo lo que se ha hecho es, en la sección de *Self illumination*, activar este efecto y dar un color al parámetro *Filter*. Será necesario, en este misma sección, ajustar los valores de las unidades de intensidad de la iluminación, según los casos, para que la intensidad del objeto iluminante sea la adecuada. En este ejemplo, en que se han utilizado dos luces simples *mr Area Spot*, sin control de exposición, se ha mantenido el valor predeterminado, *Unitless* (la opción que debe escogerse cuando se utilizan luces estándar) con un multiplicador de 1,0.

En el segundo ejemplo se ha activado, en esta misma sección, la opción *Illuminates the Scene (when using FG)*. Si se utiliza como motor de *render* mental ray con *Final Gather*, el color del objeto se suma a los rebotes de iluminación avanzada y tiñe ligeramente de color las superficies cercanas. Al activar esta opción habrá que revisar los ajustes de intensidad hasta conseguir el resultado que interese. Valores bajos harán que el efecto sea poco perceptible, pero valores altos crearán ruido y distorsiones difíciles de eliminar.

4.3 *Shaders* orgánicos. Otros *shaders*

Materiales de tipo SSS

En el capítulo 2 ya se han introducido estos materiales, en la sección correspondiente a las funciones BTDF, BSDF, BSSRDF. Como se explicaba allí, estos materiales simulan el efecto característico de algunos materiales, principalmente orgánicos, que reflejan la luz después de que esta haya penetrado las capas superficiales y se haya dispersado parcialmente. Estos efectos complejos, pero muy familiares, pues son los que presenta la piel humana, no pueden simularse correctamente si no es mediante este tipo de materiales especiales. Y aunque la simulación de la piel no entre dentro de los objetivos de este libro, hay materiales cuya calidad de representación mejora si se aplican estas técnicas, lo que justifica que le dediquemos un apartado. Otra característica notoria de los fenómenos de dispersión es que las sombras y los relieves quedan teñidos de luz, lo que les da una suavidad característica que es prácticamente imposible de simular con otras técnicas.

Los *shaders* de tipo SSS (*Sub Surface Scattering*) derivan, como también hemos visto en el capítulo 2, de los trabajos pioneros de Henrik Jensen publicados por primera vez en 2001. Tal como se ilustraba en estos primeros artículos, utilizando un rayo láser que atravesaba medios tales como la leche u otros similares, la luz penetra en estos medios de tal modo que se dispersa de un modo más o me-

nos regular en todas direcciones. Esto resultaba difícil de simular con técnicas previas. Lo que se hizo a partir de esta fecha fue integrar las técnicas y combinar un parámetro de difusión, de dispersión múltiple (*multiple scattering*), que dispersa la luz, con un parámetro de dispersión simple (*single scattering*) que mantiene relativamente focalizada la dispersión.

Estos *shaders* están incorporados en la mayoría de los programas principales. Al igual que en los ejemplos anteriores, en los que sigue utilizaré los parámetros de mental ray para que las explicaciones sean más precisas. Los parámetros que se encuentran en otros programas son muy similares.

Con mental ray, hay dos grandes grupos o versiones de este tipo de materiales.

Los grupos o versiones ***fast*** o "no físicas", utilizan recursos más eficientes de simulación pero que no reproducen con fidelidad los fenómenos físicos en que se basan. Funcionan mejor con objetos de poco espesor, como los lóbulos de las orejas o la piel que hay entre los dedos, y son adecuados para simular cosas tales como hojas de hierba o de árboles, láminas de plástico, la cera que rodea las llamas de las velas o láminas finas de materiales translúcidos. O bien cuando el coste de computación, en términos de velocidad o de memoria que requiere el otro grupo, no se puede o no se quiere asumir.

Los grupos o versiones ***physical*** intentan simular con fidelidad los fenómenos de penetración y dispersión que ocurren en el mundo real. Su coste de computación es bastante mayor y pueden ser adecuados para simular cosas tales como el jade, el alabastro o materiales altamente translúcidos, líquidos como la leche o la sangre, el agua jabonosa o láminas gruesas de materiales translúcidos.

La biblioteca general de mental ray incluye cuatro tipos de *shaders*: *misss_fast_simple_phen*, *miss_fast_skin_phen*, *miss_fast_skin_d*, *miss physical*. El prefijo "miss" viene de *mental images subsurface scattering*. El sufijo "phen" viene de *phenomena* pues estos *shaders* pertenecen al grupo de *shaders* que mental ray clasifica así para distinguirlos de los ligados estrictamente a propiedades superficiales.

Estos tipos se aplican de diferentes modos según las diferentes aplicaciones sobre las que trabaja mental ray aunque las diferencias son irrelevantes. En 3ds Max se encuentran como 4 diferentes tipos de materiales SSS. El *shader Physical* se puede aplicar también como mapa mientras que los otros tres, que en realidad son variantes del tipo más simple, *fast material*, solo se pueden aplicar como material. Los cuatro tipos son los siguientes (entre paréntesis doy el nombre genérico de mental ray).

SSS Fast Material (*misss_fast_simple_phen*). Es el más sencillo y puede utilizarse para todo tipo de materiales. Genera automáticamente los mapas de luz (*light maps*) utilizados por este *shader* y asigna una serie de propiedades genéricas para que los objetos se comporten como objetos translúcidos, dispersando la luz que les llega a nivel subsuperficial. Divide la dispersión en dos partes, una para la parte frontal y otra para la parte posterior del objeto.

SSS Fast Skin Material (*miss_fast_skin_phen*). Incluye *shaders* especiales para simular la piel humana. Puede considerarse como un material basado en la superposición, en dos capas, del anterior, más simple. Estas dos capas, la "epidérmica" y la "subdérmica", permiten incluir efectos más complejos y más aptos para simular la piel, pero que se basan en los mismos parámetros que el *Fast Material*.

SSS Fast Skin Material + Displace (*miss_fast_skin_d*). Igual que el anterior, *Fast Skin*, pero con la posibilidad adicional de incluir mapas de desplazamiento.

SSS Physical Material (*miss physical*). Es el más complejo y simula una dispersión volumétrica real. Requiere el uso de fotones, con activación de cáusticas, y controles propios de sistemas de iluminación avanzada, además de un ajuste muy fino de los parámetros por lo que resulta más complicado obtener resultados correctos.

Dadas las finalidades de este libro me limitaré a describir el primero que, por otro lado,

tampoco tiene demasiadas aplicaciones importantes para proyectos de arquitectura o diseño, si bien en algunos casos, como decía, merecerá la pena utilizarlo pues da resultados imposibles de conseguir con otros materiales.

El material *SSS Fast*. Parámetros

Como ya he dicho, esta versión simplifica el proceso a costa de una reproducción menos precisa. No se simula realmente la dispersión volumétrica sino tan solo la apariencia de esta dispersión.

Tiene varias ventajas importantes: el cálculo es más rápido debido, entre otras cosas, a que no utiliza fotones ni *ray tracing*, es más sencillo de configurar y de asimilar y sus resultados son suficientemente correctos en la mayoría de los casos prácticos. Su menor precisión se manifiesta principalmente en las reflexiones, pues el efecto de dispersión depende del punto de vista. Así, daría peores resultados si se quisiera utilizar para simular, por ejemplo, tubos o superficies curvas que arrojasen reflejos combinados con dispersión subsuperficial.

Es importante entender adecuadamente los conceptos implicados en los parámetros para poder ajustarlos correctamente.

Los términos *Front surface* y *Back surface* se refieren a *shaders* especiales que usan *lightmaps* para simular la dispersión. Estos *lightmaps* se generan en el espacio de la imagen, desde el punto de vista de la cámara, y almacenan valores que corresponden a la irradiancia difusa de las superficies del objeto junto con su coordenada de profundidad desde la cámara (*z-depth*). El término *Front* se refiere a la primera superficie que se encuentra desde el punto de vista de la cámara y que está orientada hacia ella (es decir, con su normal apuntando hacia la cámara). El término *Back* se refiere a la segunda superficie que se encuentra desde la cámara y que está orientada en dirección contraria a la anterior (es decir, con su normal apuntando en dirección contraria a la cámara). Los términos son, por tanto, relativos y la superficie "frontal" será la que se ve desde la cámara y la "posterior" la que está detrás de esta y que, si es el caso, podría dispersar luz hacia la frontal. Durante el proceso de *rendering* estos mapas se muestrean para crear dos capas de luz que incorporan información sobre el color y la profundidad. Estas capas se suman para producir el efecto final que se aplica a la representación de la superficie.

Los efectos de dispersión dependen, por tanto, del punto de vista y están programados para funcionar desde esta dirección. La dispersión generada por la superficie frontal se verá directamente y la generada por la superficie posterior se verá a través del objeto. Esto quiere decir que solo funcionará correctamente si el objeto es simple y presenta un obstáculo regular al recorrido de la luz. Deben evitarse objetos con pliegues internos irregulares que podrían complicar este efecto aunque la alternancia regular de huecos y vacios también puede dar resultados aceptables.

El color se controla desde la segunda sección de parámetros, *Diffuse SSS*. En esta sección hay básicamente tres grupos de parámetros, que afectan al color local sin dispersión (*unscattered diffuse*), al color de la superficie anterior (*front surface*) y al color de la superficie posterior (*back surface*). Estos tres grupos pueden entenderse como tres capas de material que se combinan entre sí. Cada una tiene al menos dos parámetros: *color*, que controla el color y *weight*, que controla el peso o cantidad de esta capa que se tiene en cuenta en el resultado final. Las capas anterior y posterior tienen por añadidura otro parámetro, *radius*, que controla la extensión de la dispersión de la luz dentro del material. Y la capa posterior tiene aún otro parámetro, *depth*, que controla la profundidad de penetración de la luz en el material. La figura 4.26 ilustra estos parámetros básicos.

Todos estos valores se dan en unidades absolutas que dependen de la escena. Para no tener que modificar todos los valores predeterminados que, por lo general, dan buenos resultados iniciales, puede ser preferible cam-

biar el factor de escala, el *scale conversion factor*, reduciéndolo o aumentándolo hasta que se obtengan resultados adecuados. Este parámetro se encuentra en la sección de opciones avanzadas.

El material incluye 5 secciones que se describen a continuación de un modo muy compacto y ampliando solo las explicaciones más interesantes en la práctica y que se desarrollarán en el siguiente ejemplo.

a) ***Parameters***. El primero, *Scatter group* (*A*), permite crear diferentes grupos de dispersión pero, en general, no será preciso cambiarlo pues podemos suponer que todos los objetos que nos interesen, que vayan a dispersar la luz, ya estarán en un mismo grupo. Solo habría que modificarlo en casos de objetos complejos a los que se aplica un mismo material de este tipo. El segundo, *Lightmap size* (50 %) afecta al tamaño del mapa de luz en porcentaje del tamaño de salida de *render*. El valor predeterminado es suficiente y podría reducirse, aunque si aparecen errores en los bordes quizás habría que aumentarlo, a costa de aumentar también el tiempo de cálculo. Pero no suele ser necesario. El tercero, *Number of samples*, afecta al número de muestras que se toman para el *lightmap* por cada rayo. Es recomendable, como de costumbre, usar un valor que sea potencia de 2. El valor predeterminado (64) se puede reducir a 32 para pruebas y aumentarlo si aparecen artefactos. Un valor como 128 debería ser más que suficiente para resultados finales. El cuarto, *Bump shader*, es el recurso habitual para añadir relieve, si se necesita. Se aplica a la capa externa y por tanto no afecta a la dispersión (*scattering*) sino tan solo al color de la capa inicial (*diffuse*) y la especularidad.

b) ***Diffuse sub-surface scattering***. Este grupo es el principal pues desde él se controla el color y la dispersión. Detallo todos los parámetros con los valores predeterminados entre paréntesis.

— *Ambient* / *Extra light* (negro) Afecta al color de las zonas en sombra, de modo similar a lo que se puede hacer con un material estándar. Esto afectará también al *ligthmap* y puede ser útil para añadir interesantes matices al objeto, más aún si se utiliza un mapa HDR.

— Capa *Diffuse*. Incluye los siguientes: 1) *Overall diffuse coloration* (blanco). Se utiliza para colorear todo el conjunto, tanto zonas dispersadas como no dispersadas. Puede utilizarse un mapa que añada zonas de distinto color a todo el objeto. Todos los colores difusos que hay en este grupo se

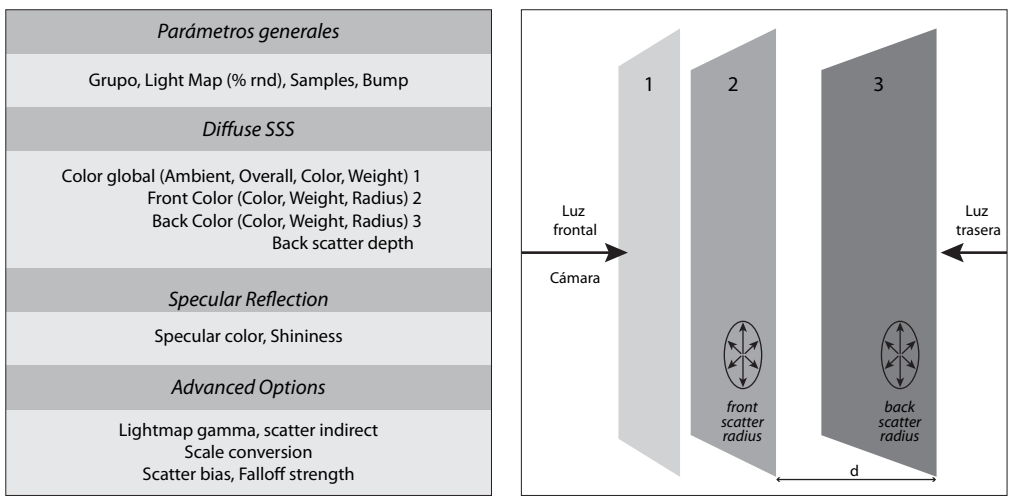


Figura 4.26 SSS Fast. *Ilustración de los parámetros principales.*

multiplicarían por este; 2) *Unscattered diffuse color* (blanco). Es el color de la zonas de superficie que no dispersan la luz. El efecto será muy similar al anterior pero menos intenso; 3) *Unscattered diffuse weight* (0,5). Controla el peso del anterior, cuanto afecta a la luz no dispersada. Un valor de 1,0 hará que cuente al 100 % y un valor de 0,0 lo anula. Los pesos que siguen tienen el mismo sentido.

— Capa de dispersión frontal (*Front surface scatter*): 1) *Color* (anaranjado). Determina el color de dispersión de la capa frontal; 2) *Weight* (0,5). Controla la influencia, el peso; de esta capa; 3) *Radius* (1,0). Este radio controla la amplitud con que la luz se dispersa en las zonas que están frente a la luz.

— Capa de dispersión posterior (*Back surface scatter*): los tres primeros parámetros, *Color* (un anaranjado más oscuro que el anterior), *Weight* (0,5) y *Radius* (1,0) tienen el mismo sentido que en la capa frontal. El cuarto, *Scatter depth* (1,0) controla la profundidad con que la luz penetra en el objeto desde la superficie posterior.

c) ***Specular reflection***. Se utiliza para ajustar el color e intensidad especular. Tiene dos parámetros que funcionan de modo semejante a los de un material *Standard*. El primero, *Specular color* (gris), controla la intensidad del reflejo (nulo para negro y máximo para blanco). Con *Shininess* (0,33) se controla la amplitud del reflejo: valores mayores lo concentran y valores pequeños lo expanden. Los resultados son similares a los que se obtendrían variando el parámetro *glossiness* de un *shader* de tipo Phong o Blinn.

d) ***Advanced options***. No suele ser necesario modificarlos aunque en algunos casos puede ser más cómodo usar el parámetro de conversión de escala para hacer ajustes, como decía más arriba.

Con *Lightmap gamma curve* (0,75) se controla el factor *gamma* para el *lightmap*. Si es 1,0, se almacena luz difusa normal (Lambertiana). Si es menos de 1,0 (con lo que la curva se aplana), la luz se dispersa más con respecto a la dirección de la luz. Si es mayor de 1,0 (con lo que la curva aumenta su pendiente), la luz se concentra más en las áreas que reciben luz directa. Los valores corrientes para que la dispersión sea notoria están entre 0,4 y 0,8. Con *Scatter indirect illumination* (desactivado) se controla la contribución de luz indirecta. Si se activa, el material incluirá luz indirecta (FG, fotones) en el *lightmap*. Esto aumentará el tiempo de cálculo y es dudoso que el resultado mejore mucho, excepto en algún caso, concretamente si se utilizan mapas HDR vía iluminación indirecta o cáusticas.

Con *Scale conversion factor* (1,0) se pueden variar a la vez todos los valores de los radios, lo que, como ya he repetido, puede resultar práctico para hacer pruebas o en casos en los que las dimensiones de los modelos requieran conversión. Es simplemente un multiplicador que afecta a todos los parámetros ligados a unidades de la escena.

Con *Scatter bias* (+0,1) se puede reequilibrar la dispersión frontal y posterior. Con 0,0 la dispersión será completamente uniforme. Por encima de 0,0 aumentará la dispersión hacia delante. Por debajo de 0,0 aumentará la dispersión hacia atrás. Aunque el rango es de -1,0 a +1,0 los valores adecuados están más bien en el rango de -0,1 a +0,1. Con *Falloff strength* (2,0) se puede controlar la distribución de la atenuación a lo largo del radio de dispersión. Valores altos (de 1,0 a 10,0) harán que sea más brusca y valores bajos (de 0,1 a 1,0) más suaves. En este último caso es posible que haya que compensarlo aumentando la distancia de dispersión para que la impresión de suavizado sea más convincente. Con *Screen* (*soft*) *compositing of layers* (desactivado) se puede controlar el modo en que se combinan las capas. Si se activa, el programa buscará un modo más suave de composición para las capas subsuperficiales. Pero en general debe mantenerse desactivado para no interferir con luces fotométricas ni con el control de exposición fotográfico.

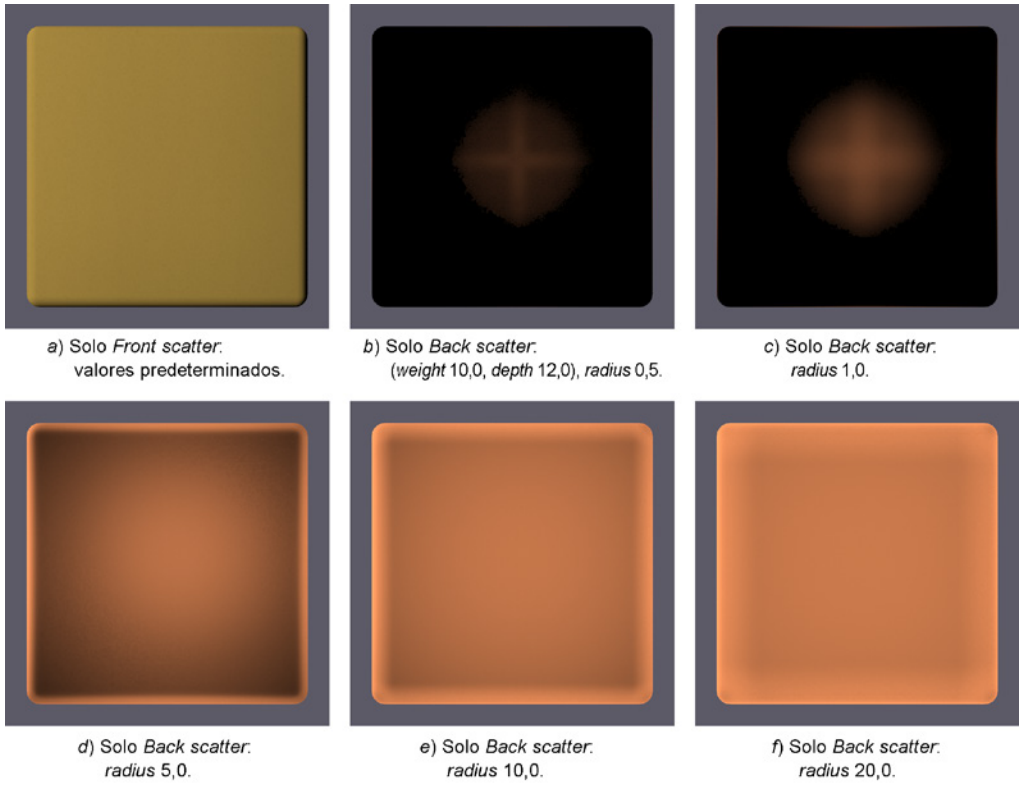


Figura 4.27 Material SSS Fast *aplicado a un prisma de 100 x 100 x 10 cm con los parámetros que se indican en cada caso.*

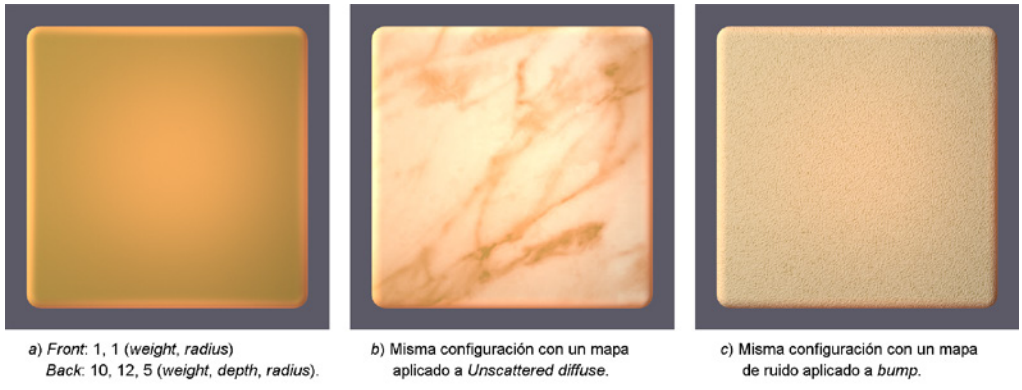


Figura 4.28 g) Material SSS Fast *aplicado a un prisma de 100 x 100 x 10 cm con los parámetros que se indican en cada caso y mapas añadidos.*

Figura 4.29 Material SSS Fast *con* Front scattering *0,5, 1 (*weight, radius*)*, Back scattering *2, 5, 20 (*weight, radius, depth*) y una tetera colocada a unos 40 cm del prisma.*

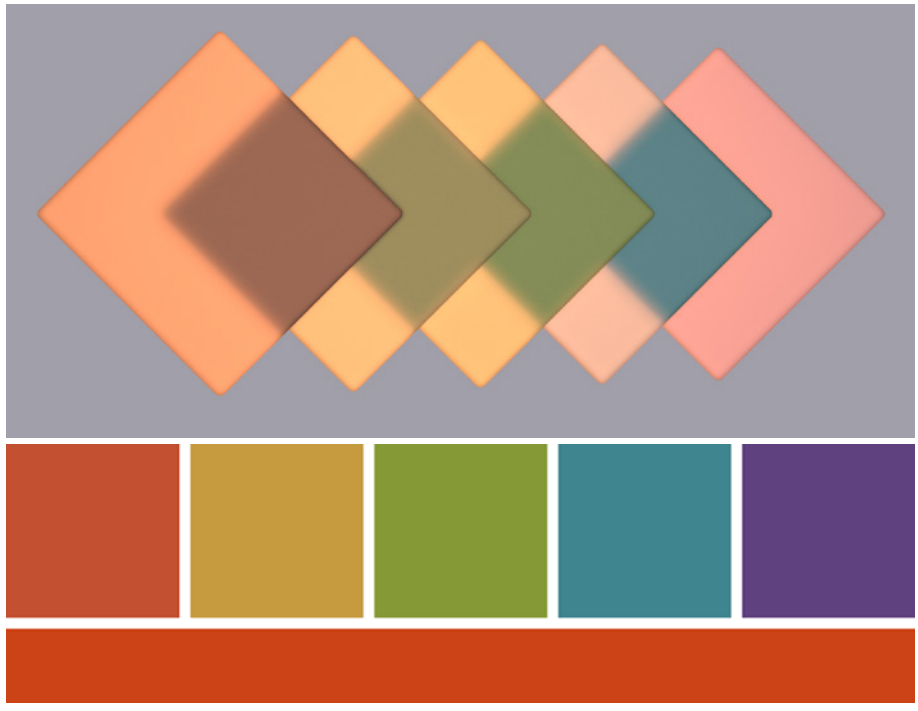


Figura 4.30 Material SSS Fast *aplicado a prismas semejantes a los de las figuras anteriores. El color de la superficie frontal de cada rombo se muestra en cada uno de los cuadrados de la figura inferior y el color de la superficie posterior en la banda que corre bajo estos cuadrados.*

El material *SSS Fast*. Ejemplo de aplicación

Para comprender mejor el sentido de los parámetros principales, convendrá aplicarlos a dos ejemplos como los de la figuras adjuntas. En ambos casos hay una luz dirigida hacia el objeto desde la posición de la cámara y otra dirigida en sentido opuesto, desde detrás del objeto.

El primer ejemplo (figuras 4.27 y 4.28) es un prisma de bordes redondeados de 100 x 100 x 10 cm. El prisma está iluminado frontalmente (desde arriba y desde la izquierda) con una luz estándar y por detrás con otra luz de iguales características pero apuntando directamente al objeto.

Para comprobar el efecto de los parámetros lo que he hecho es ir desactivando los tres parámetros principales (haciendo su peso, *weight*, igual a 0,0) e ir variando el radio. El parámetro de profundidad también afecta al resultado pero, para que resulte más claro el efecto, lo he dejado en 12,0 cm, es decir, algo más del espesor. Debe tenerse en cuenta que por debajo de este valor, hasta el límite de 10,0 o algo menos y por encima de este valor, afectaría a la intensidad.

También he desactivado el reflejo (haciendo que el *specular color* sea negro) para que no interfiera con el análisis.

En la anteúltima figura de este primer ejemplo, la 4.29, he colocado también una tetera para que se aprecie como su sombra se integra en el material. La nitidez del resultado depende principalmente de los valores de *back scatter radius* y también es afectado hasta cierto punto por el valor de *back scatter depth*.

Figura 4.31 Ejemplo 2: a) *Material corriente* (A&D)*;* b) *Material* SSS Fast *con parámetros predeterminados .*

Y en la última figura, la 4.30, se muestran cinco pastillas semejantes pero giradas y con distintos colores, que se muestran también en la parte inferior, para que se aprecie el modo en que la dispersión afecta a sus colores propios. En la configuración de los materiales, el color de la superficie frontal varía y el de la superficie posterior (banda inferior) es igual.

El segundo ejemplo es un objeto tridimensional con espesor variable. Este ejemplo nos permitirá comprobar mejor las diferencias que resultan de aplicar este material a un objeto con diferentes espesores que varían gradualmente. La iluminación consiste en dos luces frontales, una desde la izquierda y arriba, con una intensidad inicial de 0,4 y otra desde la derecha y abajo, con una intensidad inicial de 0,2 y que no arroja sombras. Y una luz posterior, que apunta directamente al objeto y con una intensidad de 1.0.

El material inicial es un material simple de color gris claro. Le he añadido también un mapa de ruido, *bump* (*size* 0.25, *fractal*) para hacer la superficie menos uniforme.

Así se obtiene un primer resultado cuya finalidad es únicamente comprobar que la iluminación y la configuración iniciales son correctas. Véase la figura 4.31 (a). Luego he asignado al objeto un material SSS Fast con los parámetros predeterminados. El resultado es el que se muestra en la figura 4.31 (b).

A partir de aquí vamos a analizar los diferentes parámetros. Para comprender su sentido los iremos aislando, haciendo que su peso sea 0,0 y ajustando el sentido del parámetro aislado con diferentes valores.

Hay que tener en cuenta que las unidades y los valores aplicados son fundamentales pues pequeñas variaciones en estos valores pueden hacer que aparezcan o no ciertos efectos. En el ejemplo, la altura es de unos 10

cm y la profundidad global está en torno a los 4 cm, aunque disminuye considerablemente según las zonas

El primer parámetro que habría que tener en cuenta es *Unscattered diffuse*. El peso (*weight*) de este parámetro determina en qué cantidad se mezclan los valores de zonas que reflejan la luz del modo tradicional con los valores que dispersan la luz internamente. Nos lo podemos imaginar como la combinación de dos materiales, uno que refleja toda la luz hacia el exterior y en todas direcciones y otro que permite que esta luz penetre su superficie antes de reflejarla en todas direcciones. Pero si el resto de los parámetros están desactivados, el resultado que obtendremos será prácticamente el mismo que obtendremos con un material corriente. La principal diferencia la obtendremos en los reflejos, pero volveremos a este tema más adelante.

Empezaremos por tanto por analizar la influencia de los parámetros de *Back surface*. Sin cambiar el color modificaremos los otros tres parámetros, *weight*, *depth* y *radius*, para estimar su valor más adecuado.

1 Anular la influencia de todos los parámetros, excepto los de *Back*, haciendo que su peso (*weight*) sea 0,0. Anular también los valores de reflexión para que no interfieran con este análisis y los que le siguen. Para ello, cambiar el color predeterminado del color (HSV 0,0,128) en la sección *Specular reflection* a negro.

2 Dejar los valores predeterminados para *scatter depth* y *scatter radius* y variar los valores de *scatter weight* de modo similar a lo que se indica en la figura 4.32, correspondiente a este parámetro.

3 Asignar a ***scatter weight*** un valor de 3.0, dejar el valor predeterminado (1.0) para *scatter radius* y variar los valores de ***scatter depth*** como en la figura 4.33.

4 Asignar a *scatter weight* , como antes, un valor de 3.0 y a *scatter depth* un valor de 3.0 y variar los valores de ***scatter radius*** de modo similar a lo que se indica en la figura 4.34.

A partir de este análisis podemos estimar los parámetros adecuados. En el ejemplo he

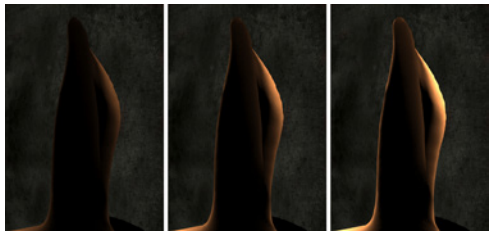


Figura 4.32 Material SSS Fast. *Ejemplo 2. Solo* Back surface. *Influencia del peso* (scatter weight)*:* a) *1,0,* b) *5,0,* c) *15,0.*

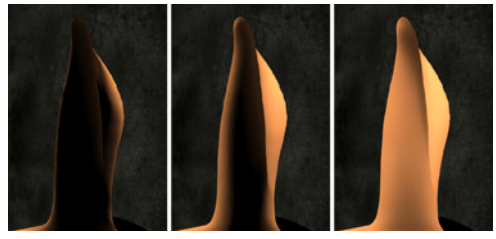
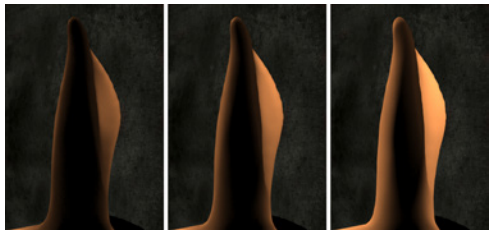


Figura 4.33 Material SSS Fast. *Ejemplo 2.* Back surface. *Influencia de la profundidad* (scatter depth) *para un espesor global de 4 a 5 cm, con* weight *3,0:* a) *1,0,* b) *2,5,* c) *5,0.*



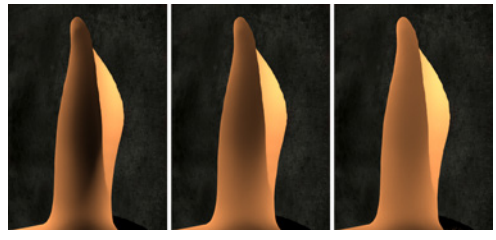


Figura 4.34 Material SSS Fast. *Ejemplo 2.* Back surface. *Influencia del radio de dispersión* (scatter radius) *con* weight *3,0 y* depth *3,0:* a) *0,2,* b) *0,4,* c) *0,8,* d) *2,0,* e) *4,0, f) 8,0.*

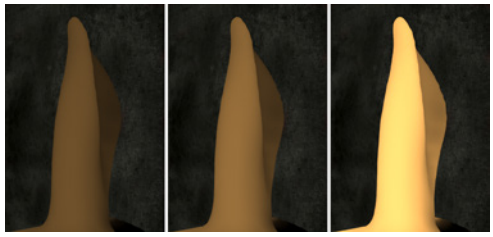


Figura 4.35 Material SSS Fast. *Ejemplo 2. Solo* Front surface. *Influencia del peso para un radio de 1,0:* a) *0,5,* b) *1,0,* c) *5,0.*

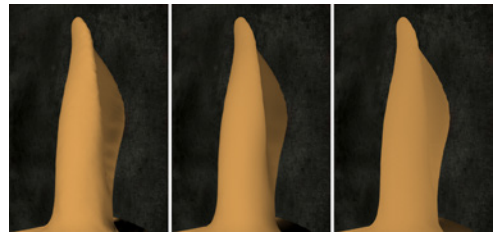


Figura 4.36 Material SSS Fast. *Ejemplo 2.* Front surface. *Influencia del radio para un peso de 2,0:* a) *0,1,* b) *1,0,* c) *6,0.*

Figura 4.37 Material SSS Fast. *Ejemplo 2.* Front surface. *Influencia de la intensidad (color especular) para un brillo (*shininess*) de 33,0:*
a) *65,* b) *128,* c) *225.*

Figura 4.38 Material SSS Fast. *Ejemplo 2.* Front surface. *Incluencia del brillo (*shininess*) para una intensidad (color especular) de 128:*
a) *5,* b) *20,* c) *100.*

Figura 4.39 Material SSS Fast. *Ejemplo 2. Resultado con la siguiente configuración:*
Front*: 0,5, 0,8 (*weight, radius*).* Back*: 3,0, 0,8, 3,0 (*weight, radius, depth*).* Specular*: 128, 75,0 (v de* hsv color, shininess*).*

adoptado los valores 3,0, 0,5 y 3,0 (*weight*, *radius*, *depth*). Es decir, un peso mayor que el predeterminado para que la influencia de la luz posterior sea importante, un radio menor para que haya más detalle en estas zonas y una profundidad de 3,0 cm para que penetre plenamente en las partes de poco espesor y parcialmente en las partes de mayor espesor.

El siguiente paso será analizar los valores más adecuados para *Front surface* lo que es más sencillo pues solo necesitamos comprobar la influencia, para un color dado, de dos parámetros, *weight* y *radius*. También podemos integrar en este análisis la recuperación de la especularidad.

1 Anular la influencia de todos los parámetros, excepto los de *Front*, haciendo que su peso (*weight*) sea 0,0.
2 Probar diferentes valores. En las figuras 4.35 y 4.36, se muestran resultados para tres configuraciones características de cada uno de estos dos parámetros.
3 Cambiar el valor de la especularidad, modificando la intensidad, dada por el color especular, que habíamos reducido a negro y probar diferentes valores, en un rango de 0 a 255, como los que se dan en la figura 4.37. Comprobar igualmente las diferencias que se obtienen modificando el valor de brillo (*shininess*), predeterminado a 33,0. En este caso, que se ilustra en la figura 4.38, no hay límites superiores ni inferiores pero por debajo de 10,0 habrá excesiva dispersión y aparecerán defectos notorios, y, por encima de 100,0 no se apreciarán demasiadas diferencias en la concentración del brillo.

De este análisis podemos también concluir valores adecuados para lo que interese conseguir. No hay ninguna otra regla pues podemos hacer que el peso del efecto de translucidez o la influencia de la especularidad sean mayores o menores, algo que dependería de las características concretas del material, de la iluminación y de otros factores difíciles de ponderar. La última figura, 4.39, muestra un posible resultado entre muchos. Los valores utilizados se dan en el propio pie de la figura.

El material *SSS Fast skin*

Si el *shader* que estamos utilizando fuera de tipo *skin*, nos encontraríamos con dos capas en lugar de una, una capa de piel "epidérmica" y otra capa "subdérmica". Esto permite introducir efectos más complejos que permiten simular los que se dan realmente en el caso de la piel humana. Para comprender adecuadamente estos efectos el procedimiento sería similar: activar una y desactivar la otra, invertir esta activación y combinar las dos de diversos modos. En el caso del *shader* simple que estamos utilizando esto se reduce a una capa "general", sin dispersión y una capa de dispersión. Los efectos son obviamente más limitados.

Como he dicho antes no voy a incluir ejemplos ni análisis pues las aplicaciones rebasan las previstas por este libro. El lector interesado puede encontrar bastante información sobre este tema en internet.

El material *SSS Physical*

Este último *shader* (que en mental ray se incluye como material), el *SSS Physical*, permite simulaciones más exactas de los efectos de dispersión. Tiene un considerable interés teórico pero, por ahora, poco interés práctico pues los parámetros son difíciles de ajustar, los tiempos de cálculo se disparan y las diferencias en los resultados son muy sutiles y prácticamente inapreciables en muchos casos.

Por otro lado, utiliza mapas de fotones para registrar los valores de transmisión por lo que habría que adentrarse en estos temas que, como ya he dicho, se tratan más extensamente en el libro sobre simulación de la iluminación. Los utiliza en combinación con una técnica denominada *ray marching* de la que trataré algo en el apartado que sigue a este, correspondiente a medios participativos.

Los fotones se dispersan en el interior del volumen a partir de los parámetros del *shader* especificados por el usuario y el algoritmo de *ray marching* se utiliza para calcular la contribución de los fotones a la iluminación a medida que nos adentramos en el volumen. La iluminación de este volumen también se suma a la iluminación global de la escena.

Puede llevar a cabo tres tipos de simulaciones de dispersión que pueden activarse de modo independiente o conjunto: *a*) *Múltiple* (*multiple-scattering estimation*). Es el método más exacto y utiliza una combinación de mapas de fotones y *ray marching* para estimar la irradiancia de la dispersión; *b*) *Única* (*single-scattering approximation*). Da un valor aproximado a partir de la intensidad de las luces, sin recurrir a la proyección de fotones; *c*) *Diffusion*. Se utiliza para calcular la contribución de los fotones a la capa más profunda.

En mental ray hay una única sección con los parámetros siguientes: *Material* (el color o textura del material), *Transmission* (valor que controla la cantidad de luz que penetra en el objeto), *Index of refraction* (índice de refracción del material), *Absorption coefficient* (0,1, 0,1, 0,1, predeterminado para cada uno de los tres canales RGB, especifica la atenuación de la luz a medida que atraviesa el medio), *Scatter coefficient* (1,0, 1,0, 1,0, predeterminado para cada uno de los tres canales RGB, especifica el grado de dispersión de la luz a medida que atraviesa el medio), *Scale conversion* (como en el *Fast material*), *Scattering anisotropy* (anisotropía de la dispersión, varía entre -1,0 y 1,0), *Depth* (profundidad de penetración de los fotones), *Max samples* (máximo número de muestras), *Max photons* (máximo número de fotones), *Max radius* (radio de una esfera centrada en un punto muestreado a partir de la cual se recolectarán más o menos fotones), *Lights* (permite especificar luces en lugar de utilizar todas las que inciden sobre el material).

Los valores de los coeficientes de absorción y dispersión se pueden obtener de la ayuda del programa para varios materiales característicos.

Medios participativos (*Participating media*). El *shader Parti volume* de mental ray

La simulación de materiales orgánicos por medio de técnicas especificas de dispersión superficial, SSS, como las que acabamos de ver, y la simulación de materiales gaseosos por medio de otras técnicas de dispersión presentan bastantes similaridades, pues los diferentes casos se distribuyen de tal modo que no se puede trazar una frontera clara entre ellos. La frontera, si hablamos de técnicas, la trazan las propias técnicas, pues unas serán más adecuadas que otras según los casos. Como veremos, se pueden simular efectos similares a los de dispersión superficial, SSS, por medio de técnicas de dispersión volumétrica que se utilizan para simular volúmenes gaseosos, humo, atmósferas polucionadas o niebla.

También resulta difusa la frontera por lo que respecta a la distinción entre simulación de materiales o de iluminación pues los efectos dependen de la luz que esté iluminando la escena y requieren especificar luces concretas. De hecho, este apartado podría haberse incluido en el libro sobre simulación de la iluminación en lugar de estar aquí. Pero me ha parecido preferible incorporarlo a esta sección por la mayor cercanía a técnicas como la simulación de materiales de diferentes densidades y consistencia.

En el capítulo 2 ya he introducido las ideas básicas en que se basan toda una serie de técnicas para simular el efecto de dispersión que tienen lugar cuando los rayos de luz atraviesan un medio gaseoso como puede ser el aire con partículas en suspensión, el humo o medios más densos. Como ya he avanzado en ese capítulo, la aplicación de técnicas de *volume rendering* a casos de interés en el campo de las aplicaciones de simulación de escenarios virtuales se basa, en primer lugar, en una diferenciación en dos grandes grupos de técnicas: las que se aplican al conjunto de la escena (*global volumes*) para simular cosas tales como niebla o volúmenes de luz y

las que se aplican a objetos concretos (*local volumes*) para simular cosas tales como fuego, humo o hipertexturas.

También hemos visto que la técnica más utilizada para este tipo de simulaciones es lo que se denomina *ray marching*. Esta técnica se implementa de diversos modos pero una de las más extendidas y más eficaces es la incorporada a un *shader* específico de mental ray, *Parti volume*. En lo que sigue describiré el funcionamiento de este *shader* específico. Es recomendable revisar los conceptos avanzados en el capítulo 2 con respecto a este tema antes de continuar.

El *shader Parti volume* se aplica a un volumen geométrico y utiliza las luces que afectan a este volumen (incluye un parámetro específico que permite seleccionar las luces que interesa que actúen sobre el volumen). Hay que tener en cuenta que, en principio, solo funciona si se encuentra con algún tipo de geometría al final de los rayos. Si al atravesar el volumen se encuentra con el vacío no se verá el volumen. Para evitar esto hay varias soluciones. La más sencilla es envolver la escena en un volumen geométrico y asignar a este volumen un *shader* especial denominado *Transmat*, que no tiene parámetros y es invisible pero que se computa como si fuera geometría real. Otra opción es modificar las propiedades del objeto para que sea transparente y no visible en el *render* pero, si se cuenta con ello, el uso de *Transmat* es más sencillo. Por otro lado, cuando se utiliza el *shader* como un efecto de cámara y ligándolo a una luz no será necesario utilizar este volumen envolvente. Veremos todos estos recursos en los ejemplos que siguen.

El *Parti volume shader* utiliza un modelo que simula la dispersión hacia delante y hacia atrás con respecto a la dirección de la luz. El procedimiento se puede resumir en cuatro pasos principales:

a) Trazado de rayos (*ray casting*). Por cada píxel computado desde la cámara se envía un rayo que atraviesa el volumen al que se ha asignado el *shader*. La primera intersección define el inicio y la segunda, el final del trazado. Es posible incluir volúmenes dentro de volúmenes pero me limitaré al caso más simple y más corriente.
b) Muestreo (*sampling*). A lo largo del rayo se seleccionan una serie de puntos. En el caso del *Parti volume* de mental ray este muestreo viene dado por dos parámetros importantes: *Minimum step length* y *Maximum step length*. Los valores de los puntos intermedios se obtienen por interpolación.
c) Sombreado (*shading*). Cada punto recibe un valor que se deriva del cálculo de *ray marching* y que computa el color inicial dado por el parámetro *Scatter color*, así como por la posición y la iluminación reci-

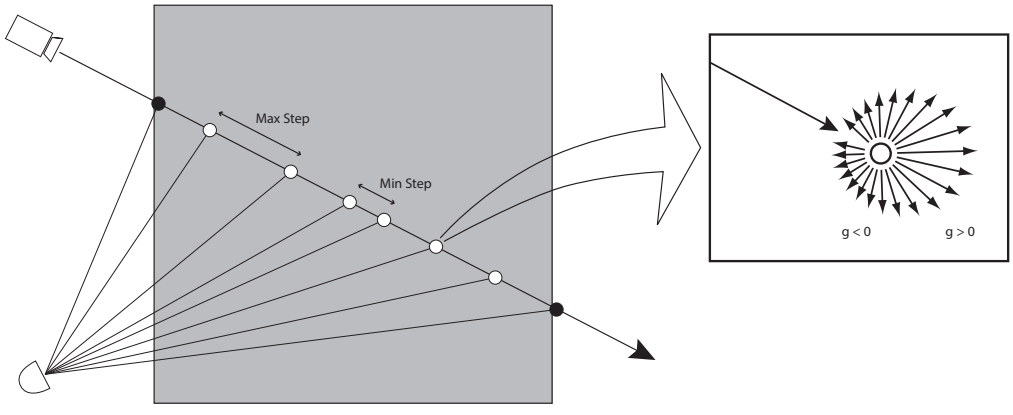


Figura 4.40 Ray marching. *Parámetros principales.*

bida y por otros parámetros importantes, principalmente *Extinction*, que especifica básicamente la densidad del medio y, finalmente, por los parámetros *r*, *g1*, *g2*, descritos en el capítulo 2 y revisados más adelante, que definen la anisotropía del medio. También puede especificarse la mayor o menor uniformidad general de dicho medio.

d) Integración final. Los valores resultantes se combinan con los colores iniciales dados por los valores propios del material asignado al volumen y otros valores presentes en la escena.

Se puede aplicar generalmente por medio de uno a tres *shaders* combinados aunque en lo que sigue trataremos solo del primero:

a) Un *volume shader* que computa el efecto a partir de la vista de cámara y de, al menos, una luz directa presente en la escena que ilumine directamente las partículas virtuales. En este caso no se requiere activar métodos de GI (*Global Illumination*). Puede asignarse a cámaras (desde *Render setup*), con lo que el efecto se aplica a toda la escena visible o bien a materiales, con lo que el efecto se aplica a los objetos que reciban este material. Cuando se asigna a materiales es necesario hacer el material transparente para que el volumen sea visible.

b) Un *photon shader* adicional que computa, por añadidura, la contribución de iluminación indirecta. Esto requiere que se activen los métodos básicos de GI disponibles en el programa.

c) Un *photon shader* que computa, por añadidura, efectos de reflexión y refracción. Esto requiere que se activen métodos complementarios de GI con cáusticas.

Por último, conviene recordar que, dado que es difícil ajustar la intensidad de este efecto, y que el cálculo es muy lento, muchos profesionales que lo utilizan prefieren computarlo como un pase independiente y luego componerlo externamente.

Parámetros del Parti volume shader

Todo lo anterior se concreta, cuando se utilizar el *shader* de mental ray, en una serie de parámetros que se pueden agrupar en tres categorías: los que afectan al color y densidad del medio (*scatter*, *extinction*), los que afectan a la calidad del resultado (*min*, *max step length*) y los que afectan a la isotropía o anisotropía de la dispersión (*r*, *g*1, *g*2). Los demás pueden considerarse como opciones adicionales que en muchos casos no será necesario tener en cuenta.

Mode. Solo hay dos modos que importen, 0 (*Fills entire volume*) y 1 (*Fills only below height*). En la mayoría de los casos hay que dejarlo en 0 (el filtro se aplica a todo el volumen del objeto que lo emite). El valor 1 (el filtro se aplica hasta un cierto nivel) se utiliza cuando se quiere crear niebla a nivel del suelo y en este caso se liga al valor del parámetro *Height*.

Scatter color. Es el color del medio de dispersión, de la niebla, humo o polución, creado por las partículas suspendidas. Cuanto más claro sea este color más luminoso y denso será el resultado. Puede substituirse el color por un mapa, y un modo interesante (aunque puede retardar aún más el cálculo) de variar el resultado es aplicar el color de dispersión por medio de un mapa de ruido. Estos mapas también pueden animarse si interesa complicar aún más la simulación.

Extinction. Es un coeficiente que controla la cuánta luz es absorbida y dispersada por el medio, las partículas suspendidas. Cuanto más alto sea este valor más denso y visible será el volumen y cuanto menor, más diáfano (0 es aire puro o vacío y elimina el efecto de dispersión). Debe mantenerse en valores muy bajos, en torno a 0,01 o 0,001 en la mayoría de los casos.

r*, *g2*, *g2. Estos parámetros afectan a la isotropía o anisotropía del medio. Véanse las explicaciones generales dadas en el capítulo 2 sobre sus fundamentos teóricos que están basados en los trabajos de los astrónomos Henyey y Greenstein. Si *g*1 y *g*2 son iguales

el medio es isotrópico. Si no, es anisotrópico. Como ya hemos visto en el capítulo 2, su sentido deriva de la fórmula *HG*(*g*1) + (1 - *r*) *HG* (*g*2) en donde *HG* representaría la ecuación general de cálculo de la dispersión. De esta ecuación resulta que si *r* = 1,0, el segundo término de la ecuación será 0 por lo que solo se utiliza el parámetro *g*1. Y a la inversa, si *r* = 0, por la misma razón solo se utilizará *g*2. Por otro lado, los valores *g*1 y *g*2 controlan la distancia a la que se produce el *scattering*. Si son positivos la dispersión es mayor hacia delante. Si son negativos la dispersión es mayor hacia atrás. Según la documentación de mental ray, basada en trabajos de Glassner, una configuración *r* - *g*1 - *g*2 de 0,50, -0,46, 0,46 genera el efecto de dispersión Rayleigh. Una configuración de 0,12, -0,50, 0,70 general el efecto Hazy Mie. Y una configuración 0,19, -0,65, 0,91 genera el efecto Murky Mie. Veremos algunos ejemplos más adelante. Con Rayleigh hay una atenuación relativamente lineal en todas direcciones. Con Hazy Mie y Murky Mie, la dispersión hacia delante (*forward scattering*), cerca de la luz, es mayor y la atenuación es más rápida.

Non uniform. Hace que el volumen se visualice como no uniforme, añadiendo ruido al resultado final. El rango es 0,0 a 1,0. Valores bajos hacen que la distribución sea uniforme y valores altos introducen variaciones en la distribución del efecto, a costa de mayor tiempo de computación. Y también convendrá aumentar el número de muestras, lo que aumentará de nuevo el tiempo de cálculo.

Height. Solo funciona cuando el Mode = 1 (*Fills only below light*), y en este caso controla la altura del volumen de niebla sobre el nivel del suelo (en el suelo el valor de *extinction* es 0,0). Se utiliza en raras ocasiones.

Minimum, Maximum step length. Controlan el número de muestras con que se procesa el volumen o, más exactamente, la longitud de los pasos computados con *ray marching*. Cuanto mayores sean estos valores (en unidades de la escena) menos muestras se toman y más rápido es el cálculo. Los valores predeterminados son 0,1 y 5,0, es decir, que el valor máximo es 50 veces superior al mínimo. En cualquier caso, es conveniente que el máximo sea al menos 10 veces superior al mínimo. Para pruebas y resultados rápidos se pueden utilizar 1 y 50 (min, max). Para resultados medios se pueden utilizar valores en torno a 0,4 y 20. Para resultados más finos habrá que volver a los valores predeterminados o algo menos, aunque no conviene bajarlos demasiado. Si el valor de *Non uniform* es mayor que 0,0 el tamaño de los pasos varía aleatoriamente entre los valores mínimos y máximos.

Light distance. Se utiliza para optimizar el muestreo de las fuentes de luz: especifica la distancia de atenuación de la calidad del muestreo en el caso de luces de área y está relacionado con el propio *sampling value* de la luz de área.

No GI where direct. Si se activa, el volumen de luz no se tiene en cuenta al hacer un cálculo de GI (iluminación avanzada). Hay que tener en cuenta que cuando está activado el cálculo global es bastante más lento por lo que convendrá desactivarlo de un modo u otro (también se puede desmarcar el efecto desde *Render setup*) si no se necesita.

Lights. Permite añadir o quitar luces a las que afecte este *shader*. Si se desactiva, el *shader* se aplicará a todas las luces de la escena. Si se activa y se escoge una determinada luz o luces, solo estas se tendrán en cuenta. Si hay muchas luces en la escena será necesario activarlo y escoger la o las que interese. Si solo hay una no es necesario.

Puede utilizarse, en principio, cualquier tipo de luz aunque los ajustes serán diferentes, tal como se indica en los ejemplos que siguen. Y en versiones anteriores a la 2012 parece que las luces fotométricas daban algún problema.

Ejemplo 1. *Parti volume* aplicado a un objeto envolvente

La escena de la figura 4.41 muestra una serie de pilares en perspectiva iluminados por un sistema de luz diurna.

En este caso es más apropiado utilizar un volumen envolvente si se quiere mantener el sistema de luz diurna con esta configuración de luces pues no podemos restringir el efecto a la zona envuelta por la luz, como en los ejemplos que siguen. Y nos sirve para ilustrar el caso un efecto de medios participativos que requiere quedar ligado a un volumen concreto.

Para ello, hay que envolver la escena con un objeto. Una ventaja importante de utilizar el *shader Parti volume*, frente a otros métodos más simples, es que podemos utilizar cualquier tipo de objeto. Para este ejemplo bastará con un prisma simple tal como el que se muestra en la figura citada. Pero podría haberse utilizado cualquier otro tipo de geometría.

Dado que el uso de medios participativos requiere, como hemos visto, que los rayos trazadores encuentren algún tipo de geometría al final de su recorrido, necesitamos utilizar un *shader* especial, *Transmat*, que es invisible pero se conmuta como geometría. Es un *shader* que no tiene parámetros, todo lo que hay que hacer es asignarlo.

Por tanto, hay que aplicar al objeto envolvente un material con las características siguientes:

Type: mental ray.
Surface: Transmat.
Shadows: Transmat.
Volume: Parti volume

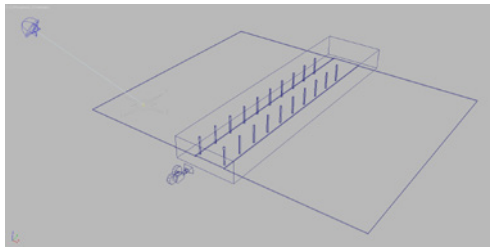


Figura 4.41 Parti Volume. *Ejemplo 1. Vista general de la escena que incluye una serie de pilares, un plano de base y un prisma alrededor de los pilares que actúa como volumen envolvente.*

Y configurar el *Parti volume* con los parámetros que se indican en las figuras.No indico los valores de *Min*, *Max*. Como en todos los ejemplos que siguen, lo primero que hay que hacer es reducir los valores de *Min*, *Max steps* para hacer pruebas pues esto reduce el tiempo de cálculo, y luego aumentarlos. Los valores utilizados para el cálculo final en estos ejemplos han sido de 0,8, 4,0.

Lo siguiente que se debe hacer es buscar valores adecuados para el color de dispersión (*scatter color*) y para el tamaño de la dispersión (*extinction*). Los valores asignados se indican en los pies de las figuras.

Ejemplo 2. *Parti volume* aplicado a una luz focal

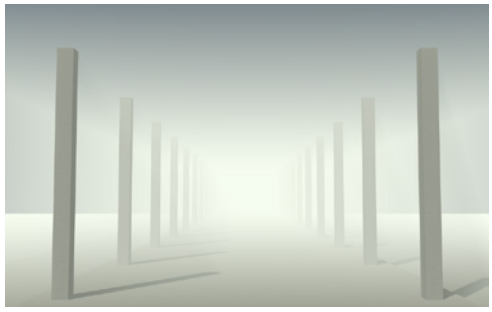
La siguiente escena (figura 4.44) incluye una luz principal situada directamente sobre el objeto, de 45 x 45 cm, que arroja sombras, y dos luces secundarias que iluminan el objeto por los dos lados para suavizar el contraste. Las intensidades (multiplicadores) de estas luces son 5,0 (principal), 0,5 (secundaria izquierda) y 0,2 (secundaria derecha).

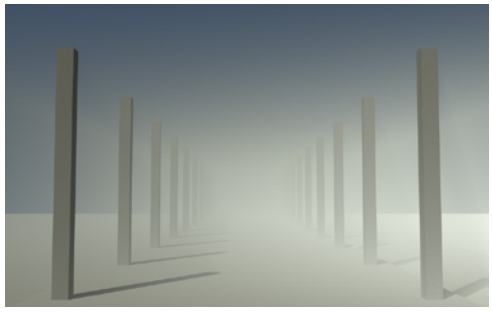
El *shader Parti volume* se ha configurado así:

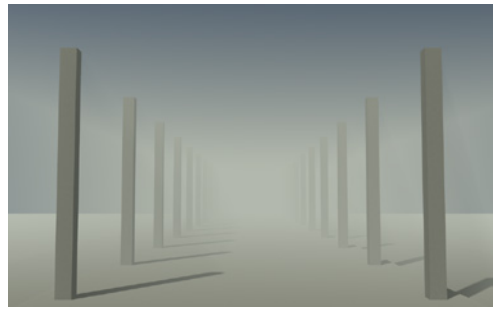
Mode: 0.
Scatter color: gris claro (hsv 0, 0, 0,5).
Extinction: según lo indicado en el pie de la figura.
Min/*Max step length*: 0,1/10,0 para pruebas y 0,1/1,0 para el resultado final.
Lights: presionar el botón *Add* y seleccionar la luz principal.
El resto de los parámetros (*r*, *g*1, *g*2, *Non uniform*, *Height*, *Light distance*) se han dejado con sus valores predeterminados (0,0).

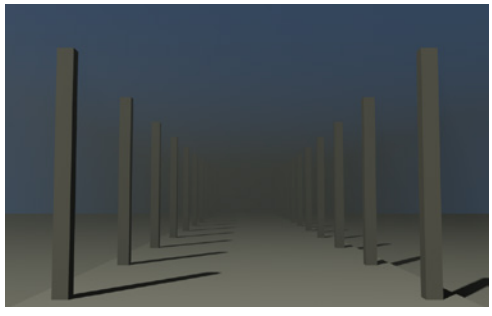
Ejemplo 3. *Parti Volume* aplicado a una luz directa

La escena de la figura 4.45 representa un interior con una ventana y una luz que entra por ella. La luz es una luz directa de tipo *Standard*









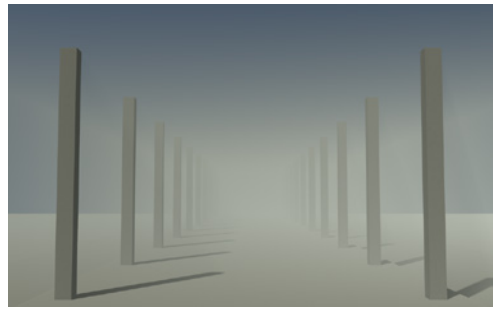


Figura 4.42 Parti volume. *Ejemplo 1. Resultados con r, g1, g2:* a) *0,0, 0,0, 0,0,* b) *1,0, -0,95, 0,0,* c) *0,0, 0,0, 0,95.*

Figura 4.43 Parti volume. *Ejemplo 1. Resultados con la misma configuración anterior y r, g1, g2:* a) *0,50, -0,46, 0,46 /Rayleigh),* b) *0,12, -0,50, 0,70 (Hazy Mie),* c) *0,19, -0,65, 0,91 (Murky Mie).*

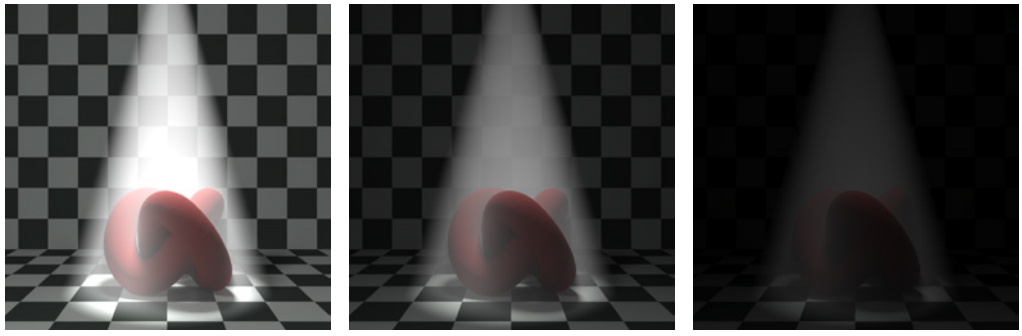


Figura 4.44 Parti volume. *Ejemplo 2. Objeto iluminado por un foco. Diferencias con la configuración general indicada en el texto y valores de extinction de:* a*) 0,005,* b*) 0,012,* c*) 0,025.*

(puede ser de cualquier tipo). También he utilizado un par de luces secundarias de baja intensidad para que las paredes, suelo y techo sean visibles. A partir de ahí, la configuración se establece como sigue:

1 En *Render setup/ Renderer/ Camera effects*, en el grupo *Camera shaders* asignar a la entrada *Volume* un *shader Parti volume*.
2 Llevar el *shader* al editor de materiales y editarlo. Probar valores adecuados. Los del ejemplo son:
Scatter color: azul muy oscuro (RGB 0,19, 0,20, 0,23)
Extinction: 0,004
*Valores r, g*1*, g*2: 1,0, -0,25, 0,0
Non uniform: 0,65
Lights: presionar el botón *Add* y seleccionar la luz directa que entra por la ventana.

Figura 4.45 Parti volume. *Ejemplo 3. Luz directa a través de una ventana.*

Resto de parámetros: valores predeterminados.

Sistemas de partículas

La descripción detallada de las técnicas que se engloban en el uso de sistemas de partículas queda fuera de los límites de este libro. Sin embargo lo incluyo aquí por su cercanía con los *shaders* que se utilizan para simular materiales tales como humo, fuego, gases, etc. Y por sus amplias posibilidades, pues se utilizan para simular los siguientes tipos de materiales y fenómenos: agua, nieve, gases, polvo, nubes, niebla, fuego, humo, explosiones, hierba, árboles, guijarros y algunas cosas más.

En este apartado daré una descripción básica de cómo utilizar estos sistemas y de cómo incorporarles especificaciones de materiales sin mapas. En el capítulo siguiente ampliaré esta descripción para incluir materiales con mapas, lo que amplía considerablemente las posibilidades de simulación de estos sistemas. Las descripciones serán muy breves por lo que, si el lector no las considera suficientes, deberá ampliarlas con la ayuda del programa o tutoriales específicos.

Generalidades

En el capítulo 2 ya se han resumido los conceptos y principios básicos a partir de los cuales se crea un sistema de partículas. Conven-

drá revisar el apartado correspondiente de ese capítulo antes de seguir con las técnicas específicas que se dan aquí y que, como en otros casos, están basadas en 3ds Max, si bien los procedimientos son muy similares en otros programas.

Cualquier implementación de un sistema de partículas consta de los siguientes elementos básicos: *a*) un **emisor** (*emitter*), que genera partículas en la escena desde una posición determinada, *b*) un conjunto de **parámetros**, que controlan la cantidad, el tamaño, el lapso de vida de las partículas, su velocidad, su color o características materiales y otras características secundarias, *c*) un conjunto de especificaciones adicionales que afectan a su **representación** final.

En 3ds Max los emisores son básicamente de dos tipos: *a*) emisores clásicos (de importancia decreciente) no basados en eventos: *Spray*, *Super spray*, *Snow*, *Blizzard*, *PArray* y *pCloud*, b) un emisor especial, *PF Source* (*Particle flow source*, fuente de flujo de partículas) que permite crear sistemas más complejos por medio de un panel especial, *Particle view*, que facilita la adición de todo tipo de eventos y operadores, con parámetros para controlar las características y flujo de las partículas. La mayoría de los sistemas actuales se crean con este último tipo y tanto la descripción que se da en el siguiente apartado como los ejemplos que siguen estarán basados en él.

Además de los elementos básicos, los sistemas de partículas se complementan con recursos adicionales que simulan fuerzas de diferentes tipos que actúan sobre el sistema y modifican su comportamiento. Los más habituales son fuerzas que simulan el viento, la gravedad, la restricción a un recorrido o la acción de un impulso indeterminado. Estos recursos adicionales se agrupan en una categoría general denominada *Space warps* que se describe más adelante.

Por lo general, los sistemas de partículas se utilizan en animaciones, por lo que uno de los parámetros que hay que especificar es el rango de la animación en que serán activas, es decir, en que *frame* se inician y en que *frame* se extinguen. También es relativamente corriente hacer que empiecen antes y terminen después, especificando, por ejemplo, valores de -100 y 500 cuando el rango es 0 a 400. Pero también se puede hacer una simulación estática, especificando que los valores del *frame* de inicio y final sean 0 y 0. Sin embargo, incluso en representaciones estáticas como las que más nos importan, merecerá la pena utilizar un determinado rango de *frames*, pues esto nos permitirá seleccionar con facilidad la vista más adecuada.

Procedimiento básico. Parámetros principales de *PF Source* y *Particle view*

El procedimiento más general para crear un sistema de partículas es crear un emisor genérico, *PF Source*, y luego añadirle operadores y eventos por medio del panel *Particle view*. También se puede entrar directamente en *Particle view* y crear el emisor genérico desde allí. Pero como mi intención no es hacer una descripción exhaustiva describiré tan solo la primera vía. Para crear un sistema de partículas con un procedimiento básico, en 3ds Max, debemos hacer lo siguiente:

1 Crear un emisor de tipo *PF Source*. Ir al panel *Create*/ *Particle systems*, escoger el tipo *PF Source* y pinchar y arrastrar sobre la escena. Aparecerá un rectángulo con un logo en su interior y una flecha apuntando hacia abajo.
2 Mover y rotar el icono de *PF Source* para situarlo y orientarlo según interese. Las partículas siguen la flecha, por lo que si queremos que se emitan hacia arriba habrá que girarlo 180º y si queremos que emitan horizontalmente habrá que girarlo 90º.
3 Editarlo y cambiar sus parámetros de emisión. La emisión depende también de la forma y del tamaño del icono por lo que habrá que cambiar el *icon type* (puede ser rectangular, circular, un prisma o una esfera) y sus dimensiones. También podemos

cambiar el número de partículas que se muestran en el visor (un 50 % por defecto).

4 Comprobar que se emiten partículas desplazando el deslizador de tiempo (*time slinder*) desde la barra de animación. Si interesa, cambiar el rango predeterminado de 100 *frames* entrando en configuración de la animación (el pequeño botón situado a la derecha de esta barra).

5 Entrar en *Particle view* presionando el botón de este nombre que también se encuentra en la sección *Setup* de *PF Source*. O desde el menú *Graph editors*. O presionando la tecla 6, que es el método más directo y más fácil de recordar.

Al entrar en *Particle view* después de haber creado este icono nos encontraremos con dos cosas. Un recuadro etiquetado *PF Source*, con un único operador, *Render* (*Geometry*), del que no tenemos que preocuparnos mucho pues todo lo que hace es proporcionar un punto de partida mínimo con parámetros básicos que podemos especificar desde otros operadores.

Y, a continuación, una serie de eventos y operadores predeterminado que corresponden al *Standard flow* que aparece en la parte inferior, el "Depósito" (*Depot*).

El *Depot* (que resultará más cómodo ocultar cuando estemos familiarizados con él e insertar operadores con el botón derecho del ratón) incluye los operadores disponibles, en cinco grupos con diferentes colores y tipos de iconos. La descripción que doy es mínima, si no se considera suficiente habrá que ir a la ayuda del programa:

Flows (rectángulos blancos). Hay cuatro tipos: *Preset* (carga un *flow* con una configuración guardada previamente); *Empty* (crea un único evento global con tan solo un operador de *Render*); *Standard* (crea un evento con una serie de operadores básicos: *Birth*, *Position*, *Speed*, *Rotation*, *Shape*, *Display*); *One click* (utiliza datos guardados en *cache* que comprenden los siguientes operadores: *Birth file*, *Material static*, *Shape*, *Display*).

El procedimiento más habitual es crear un *Standard flow* y luego borrar los que sobran y añadir los que interesen.

Birth operators (círculos verdes). Hay cinco tipos: *Birth* (activa la generación de partículas con la configuración especificada en el panel ligado a este operador); *Birth file* (utilizado para interactuar con otras aplicaciones), *Birth script* (basado en scripts) y *Birth texture* (basado en texturas en blanco y negro). El procedimiento habitual es utilizar un operador *Birth* para fijar el rango (*Emit start*/*Emit stop*) y la cantidad (*Amount*).

General operators (cuadrados azules). Hay más de 20 tipos. *Delete*, *Force*, *Group*, *Group Selection*, *Mapping*, *Mapping object*, *Material dynamic*, *Material frequency*, *Material static*, *Position icon* (liga el sistema de partículas al icono creado en la escena: *surface* emite de modo aleatorio desde la superficie, *volume* de modo aleatorio desde el volumen, *pivot*, *vertices*, *edges*, desde estos subelementos), *Position object* (liga el sistema de partículas a un icono o un objeto), *Rotation*, *Scale*, *Script*, *Shape*, *Shape facing* (crea caras ligadas a un *look at*, tal como una cámara, para que las caras se orienten siempre con respecto al objeto escogido y darles un tamaño y una orientación), *Shape instance* (permite especificar un objeto de la escena), *Shape mask*, *Speed*, *Speed by icon*, *Speed by surface* y *Spin*. Veremos ejemplos de usos de los principales en los ejemplos que siguen.

Test (rombos amarillos). Hay 15 tipos. *Age test*, *Collision* (comprueba si las partículas chocan con deflectores u otras partículas), *Collision spawn* (crea nuevas partículas a partir de choques con deflectores), *Find target* (envía partículas a destinos específicos), *Go To rotation* (rota las partículas en orientaciones específicas), *Lock*/*Bond* (enlaza partículas con objetos), *Scale test* (controla los cambios de escala), *Script Test* (usa scripts para comprobar propiedades específicas de las partículas), *Send out* (envía partículas al siguiente evento), *Spawn* (crea nuevas partículas a partir de otras), *Speed test*, *Split amount*, *Split group*, *Split selected*, *Split*

source (proporcionan diferentes controles para subdividir el flujo).

Miscelánea (cuadrados azules). Hay tres: *Cache* (registra estados de partículas en sistemas complejos), *Display* (especifica el tipo: dots, ticks, círculos, líneas, geometría…), *Notes* (es un operador no ejecutable que se utiliza para añadir notas a la escena). En principio solo nos importa el segundo, *Display*.

Al seleccionar un operador se muestra una descripción breve en el panel derecho. Para añadir una acción a un evento se arrastra el operador desde el *depot* al panel principal. También se puede arrastrar a un área vacía para crear un operador independiente que luego se conectará a otros por medio de *Wires*. Y también se puede añadir por medio del menú contextual como veremos a continuación.

El panel central o ***Event display*** contiene la estructura de eventos. Al seleccionar un elemento del panel central la parte de la derecha muestra sus propiedades y sus parámetros. Estos cambios son los principales y los que veremos con detalle, a partir de ejemplos, más adelante. También se puede seleccionar un operador e invocar el menú contextual (botón derecho del ratón), que incluye comandos tales como *Insert* y *Append* que dan acceso a submenús de operadores, ordenados, en este caso, en tres categorías. Hay que tener en cuenta que *Insert* substituye el operador seleccionado mientras que *Append* lo conserva y añade otro al final.

Parte de los *Space warps* están incluidos en el nuevo sistema de modo directo, lo que también facilita el trabajo. Hay dos categorías principales que deben tenerse presentes: *Forces* y *Deflectors*. Las fuerzas, que se introducen por medio de un operador específico, afectan al comportamiento de la partícula, principalmente a su movimiento y dirección. Los deflectores, las afectan a través de un operador tipo test que determina en primer lugar si se produce una colisión y qué hacer en este caso. Los principales *Space warps* se describen brevemente en el apartado siguiente.

Space warps (*Forces*, *Deflectors*)

Los *Space warps* se crean desde el panel *Create* / *Space warps*. La lista desplegable de este panel permite seleccionar cinco tipos. Los que nos interesan son los dos primeros, *Forces* y *Deflectors*. La lista que sigue describe brevemente los principales tipos de *Forces* (los siete primeros) y de *Deflectors* (agrupados en el octavo miembro de esta lista). Si interesa profundizar en este tema convendrá ampliar esta información con la ayuda del programa, que incluye buenas ilustraciones.

1) ***Push***. ***Motor***. ***Vortex***. *Push* genera una fuerza lineal, es decir, acelera (o retarda) el movimiento de las partículas en la dirección dada. Se pueden modificar los valores básicos con los parámetros de *Amplitude* y *Period*. *Motor* genera un movimiento rotacional en torno a su eje. La dinámica de la rotación se controla con los parámetros *Torque* (torsión) y *Revolutions per minute*. También se puede ajustar el efecto ajustando los valores de fase y amplitud en *Periodic variation*. *Vortex* puede entenderse como una combinación de *Motor* y *Push*, es decir, una combinación de empujes rotacionales y lineales. Produce efectos similares a los de un tornado. Los parámetros principales son *Radial pull* y *Orbital speed*.

2) ***Drag***. Se utiliza principalmente para atenuar el movimiento de las partículas a medida que evolucionan.

3) ***PBomb***. Impulsa las partículas hacia fuera, como si se hubiera producido una explosión (de ahí el nombre). La dirección depende de la posición de las partículas y de los parámetros *Blast symmetry* y *Explosion*. Se suele utilizar en combinación con *Gravity* y *Wind*.

4) ***Gravity***. Se utiliza corrientemente en combinación con otros operadores para simular la fuerza de la gravedad. Se supone que la flecha apunta hacia abajo pero puede modificarse su dirección al igual que puede activarse el parámetro *Spherical* para que las partículas sean atraidas hacia el centro del *space warp*.

5) ***Wind***. Es otro de los *space warps* más utilizados. Crea una fuerza, similar a la de *Gravity*, pero en la dirección dada por la flecha incorporada a su icono. La intensidad se controla con el parámetro *Strength* y, si se controla desde *Particle view*, con el parámetro *Influence*%. Se pueden utilizar parámetros adicionales para hacer algo más complejo el efecto: *Turbulence*, que altera la distribución de las partículas, *Frequency* (que modifica esta distribución) y *Scale* (otro tanto).

6) ***Displace***. Se utiliza para controlar la distribución de las partículas por medio de un mapa asociado, un recurso habitual en muchas aplicaciones digitales. Veremos ejemplos del uso de este *space warp* en el apartado sobre uso de mapas en sistemas de partículas.

7) ***Path follow*/ *Speed by icon***. Hacer que las partículas sigan un recorrido definido por una *spline* es, obviamente, un recurso muy útil para un gran número de aplicaciones. Esto se consigue, en principio, mediante este recurso que es uno de los que están disponibles en el panel *Space warp*/ *Forces*. Pero no puede activarse desde *Particle view* por lo que habría que utilizarlo a través de las herramientas clásicas de versiones anteriores. Sin embargo, no es necesario hacerlo pues es posible conseguir lo mismo mediante otro recurso igual o mejor que puede parecer algo más complicado pero que realmente es muy sencillo. Lo explicaré mediante un ejemplo concreto.

Supongamos que queremos que un determinado sistema de partículas, formado por pequeños conos, siga un recorrido dado por una *spline*. Por tanto, comenzamos por dibujar la *spline*, en planta, y un pequeño cono, tal como se ilustra en la figura 4.46. Una vez que contamos con estos dos objetos hacer lo siguiente.

1 Crear un *PF Source* en un visor frontal. Cambiar su tipo a circular y su radio a un valor proporcionado a la *spline*, el cono y el tipo de distribución que nos interese, y moverlo hasta situarlo de tal modo que la flecha apunte al inicio de la *spline*, como en la figura citada (podíamos hacer que esto sirviera para crear una animación a lo largo de un túnel y, en este caso, el diámetro del icono debería coincidir con el del túnel).

2 Abrir el *Particle view*, que mostrará un *PF Source* con sus eventos predeterminados. Para introducir de un modo más directo operadores que substituyan a otros, como *Speed by Icon* en lugar de *Speed* y *Shape instance* en lugar de *Shape* de un modo más directo, seleccionar el operador y hacer BDR (botón derecho ratón)/*Insert*/*Operator*. Configurar todo como sigue (los valores pueden modificarse según interese):

Birth. Emit Start/Stop> 0/200. Amount> de 100 a 200.
Position icon. Predeterminado (volume).
Speed by icon. (Dejar en principio los valores predeterminados pero luego probar a modificar la aceleración y otros parámetros).
Rotation. Orientation matrix> Speed space follow (esto hace que se mantenga la orientación dada por Speed by icon). Modificar la rotación sobre los ejes xyz (-90, 180, 0 en el ejemplo) para que se oriente como interese.
Shape instance. Particle geometry object: (seleccionar el objeto). Si es necesario cambiar la escala (65 % en el ejemplo).
Display. Type: Geometry.

Al crear el operador *Speed by icon* habrá aparecido en la escena el icono correspondiente. Según las características de la escena es posible que no se visualice inmediatamente. Si es así, seleccionarlo por nombre y cambiar su tamaño y su posición.

El último paso será enlazar este icono con la *spline* siguiendo procedimientos corrientes en animación para enlazar objetos con trayectorias. Para ello continuar como sigue:

3 Seleccionar el icono *Speed by icon*. Ir al panel *Motion* / *Parameters*. En la sección *Assign controller*, en la lista de controles de transformación, seleccionar *Position* y presionar el botón *Assign controller*. Se

abrirá otro panel, *Assign position controller* con una lista de controladores. Seleccionar *Path constraint*. Con esto aparecerá un nueva sección, *Path parameters*. Presionar el botón *Add path* y seleccionar la *spline*. La *spline* aparecerá en el cuadro inferior y, a continuación, una serie de parámetros adicionales para ajustar la posición del objeto en el *path* que, en este caso, no será necesario modificar.

4 Desplazar el *time slider*. Las partículas (instancias del cono que hemos creado) se crearán a lo largo del recorrido.

8) ***Deflectors***. En general, un deflector (del latín *de-flectere*, "doblar, curvar") es un dispositivo creado para desviar un flujo indeterminado (de aire, agua, partículas).

Como decía anteriormente, los 7 grupos anteriores son *Forces* y, en 3ds Max, se encuentran agrupados en la primera categoría de *Space warps*. La segunda categoría de la lista de recursos disponibles es la de *Deflectors*. E incluye los seis siguientes: *POmniFlect*, *SOmniFlect*, *UomniFlect*, *Udeflector* (*Universal deflector*), *Sdeflector* y *Deflector*. Las diferencias se basan en la forma y en la funcionalidad.

Los deflectores que se incluyen en 3ds Max (y otros programas) pueden ser de tres formas básicas: planos, esféricos y universales. La forma de los dos primeros es obvia. La forma del tercero depende de la geometría a la que se enlaza. En este último caso debe tenerse en cuenta que el coste computacional será bastante mayor.

La funcionalidad puede ser también de tres categorías: *ordinaria*, *omni-flect* y *dyna-flect*. La primera es, como su nombre indica, la más corriente y es la que utilizaremos en los ejemplos que siguen. La segunda incluye varias características que pueden ser de interés, entre las cuales están: *Reflection parameters* (controles para ajustar cómo rebotan las partículas), *Refraction parameters* (controles para ajustar cómo se desvían las partículas que penetren en el deflector), *Friction parameters* (controles para ajustar la velocidad de las partículas cuando inciden en el deflector con ángulos rasantes), *Spawn reactivity* (controles para ajustar el lapso de vida de las partículas después de topar con el deflector). La tercera requiere que, para que la respuesta sea más compleja, se utilicen reactores, otro tipo de recurso de 3ds Max que pasaremos por alto en este resumen que pretende ser breve y adecuado a las finalidades generales de este libro.

Hay tres tipos de la primera categoría, deflectores corrientes, y tres tipos de la segunda, deflectores omni. Estos tipos son concreciones de las categorías genéricas que he resumido en el párrafo anterior.

Los deflectores corrientes tienen capacidades limitadas. El tipo *Deflector* actúa como una barrera plana que detiene el flujo de partículas, sin más. *SDeflector* es una versión esférica de este mismo tipo, y *UDeflector*, una versión universal, es decir, que se puede enlazar a cualquier tipo de geometría.

Los deflectores omni tiene capacidades adicionales para refractar partículas y generar partículas derivadas con un lapso de vida determinado. El tipo *POmniFlect* es una versión plana, *SOmniFlect*, una versión esférica

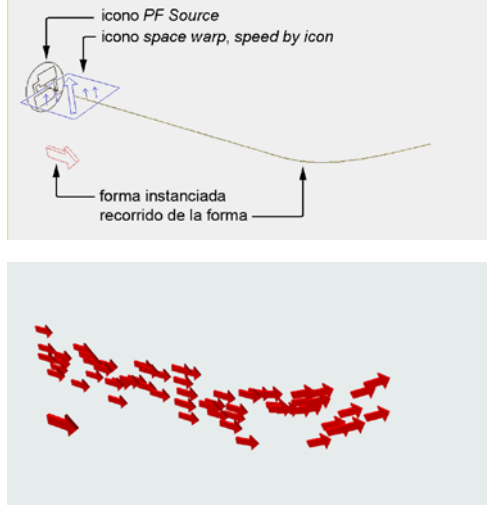


Figura 4.46 Partículas. Operadores básicos. Uso del space warp, speed by icon*: a) esquema interno, b) resultado.*

y *UOmniFlect*, una versión universal, enlazabla a cualquier geometría.

El procedimiento general es: *a*) crear un deflector, es decir, seleccionar un tipo y arrastrar en la escena para colocar su icono, *b*) si es de tipo universal, escoger el objeto a que se aplicará y si no, no hacer nada, *c*) enlazar el deflector con un sistema de partículas. Si se utiliza un *Particle flow*, especificarlo con un operador de tipo *Collision test* o *Collision spawn*. Si se utiliza un sistema clásico, no basado en eventos, enlazarlo con el sistema (*Bind*), *d*) colocar el deflector en una posición adecuada para interrumpir el flujo de partículas, e) reajustar sus parámetros si fuera necesario.

Ilustraré todo esto con el primer ejemplo, después del apartado siguiente sobre materiales.

Materiales (sin mapas) y sistemas de partículas

Para asignar materiales a un sistema de partículas, el método más inmediato es crear un material, en principio de cualquier tipo, y asignarlo al icono que representa el sistema de partículas.

Las descripciones que siguen utilizan materiales multisubobjeto. Véase el apartado sobre proyecciones múltiples, más adelante, para una descripción más completa de esta técnica, aunque para seguir los ejemplos bastará con ceñirse a lo que se indica.

Si se utiliza *Particle view* será preferible utilizar un operador pues tendremos más opciones de ajuste. Hay tres tipos de operadores de materiales:

Material static mantiene la asignación a lo largo del evento en que está incluido, tanto del material principal como, en su caso, de los submateriales incorporados a este por medio de identificadores (ID). Véase el ejemplo que se da más delante de un emisor simple.

Material dynamic se utiliza para asignar materiales animados a las partículas. La animación puede llevarse a cabo mediante texturas animadas o mediante técnicas de rotoscoping asignadas a submateriales. Pero su utilización mas corriente es ligándolo a un mapa de tipo *Particle age* que liga la aparición del material a la vida de las partículas. Véase el ejemplo sobre simulación de humo que se da en el apartado de partículas con mapas, en el capítulo siguiente.

Material frequency se utiliza para redistribuir submateriales en función de sus ID y de frecuencias de aparición controladas por paramétros específicos. En principio no lo utilizaremos. Si se quiere comprobar cómo funciona, crear un ejemplo simple, una escena con un plano y un objeto. Luego crear un sistema de partículas estándar, utilizar un operador *Position object* y escoger el plano como referencia, añadir un operador *Shape instance* y escoger el objeto como instancia. Dar un número adecuado de partículas para que se dispersen sobre el plano, mover el *time slider*, ajustar valores de velocidad (0) y rotación (*random horizontal*), etc. Luego crear un material multisubobjeto con 10 materiales y submateriales *standard* con varios colores y asignarlo al objeto original. Añadir un operador *Material frequency* y arrastrar el material que hemos creado al botón *Assign*. Marcar las opciones *Assign material ID* y *Show in viewport*. En el panel se mostrarán los 10 submateriales con un valor ligado al *frame* en que aparecen. Llevar el *time slider* hacia delante (o al final) y hacer que los dos o tres primeros tengan porcentajes altos y el resto 0. Solo se mostrarán estos tres y con la frecuencia especificada. Luego aumentar el valor de otro que comenzará también a aparecer en función de la frecuencia asignada.

En general, hay que tener en cuenta que si el material asignado es multisubobjeto la fuente del material afecta a cómo se representan las partículas. Si se ha aplicado al icono, las partículas, en principio, reciben un diferente submaterial a medida que se generan, recorriendo los subobjetos hasta que se agoten, es decir, que la primera partícula recibe el submaterial 1, la segunda el 2, y la *n* el *n*. Se pueden visualizar los ID de las partículas si, desde *Particle view*/ *Display*, se marca la op-

ción *Show particle Ids*. Aparecerá un número junto a cada partícula.

Las posibilidades aumentan si utilizamos mapas, por lo que este apartado debe verse como una introducción general que habrá que completar con el apartado correspondiente del capítulo siguiente.

Los ejemplos que siguen muestran las diferentes posibilidades que tenemos utilizando solo materiales, sin mapas.

Ejemplo 1. Emisor elemental

El ejemplo siguiente ayudará a entender mejor las descripciones anteriores. Se trata de crear un sistema simple, un objeto que emite partículas y las envía sobre un plano en el que rebotan hasta quedar en reposo. Este ejemplo ilustra el uso de emisores básicos, de fuerzas (*wind*) y deflectores (el plano base).

1 Crear una escena similar a la de la figura 4.47.
2 Crear un *PF Source* siguiendo el procedimiento básico que hemos visto. Cambiar su tipo a círculo. Luego moverlo y girarlo hasta que quede alineado con el agujero del objeto emisor (véase la figura citado). O bien crearlo directamente sobre el lado inclinado con *Autogrid* y luego moverlo en coordenadas locales. Por último, ajustar sus dimensiones para que coincidan más o menos con las del agujero.
3 Crear un *Deflector* (desde *Create/Space Warps/Deflectors*) de dimensiones similares a las del suelo y alinearlo con él.
4 Crear también una *Gravity force* y una *Drag force* (desde *Create/Space Warps/Forces*). En este caso no importa la posición, basta con que sean visibles y no molesten. Completar *Drag force* con *Gravity* nos permitirá hacer ajustes más precisos.
5 Entrar en *Particle view* desde el *PF Source* (botón *Setup*).
6 Configurarlo como sigue (borrar o insertar operadores según se necesite):
Birth. Emit Start/Stop: 0/100. Amount: 300.
Position Icon. Location: Volume (o Surface, tanto da pues hemos escogido un icono plano).

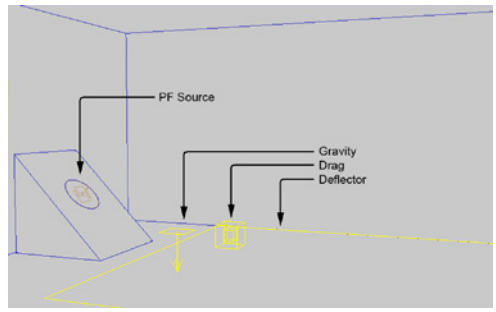


Figura 4.47 Sistemas de partículas. Emisor elemental. Vista general de la escena con los iconos de partículas y fuerzas.

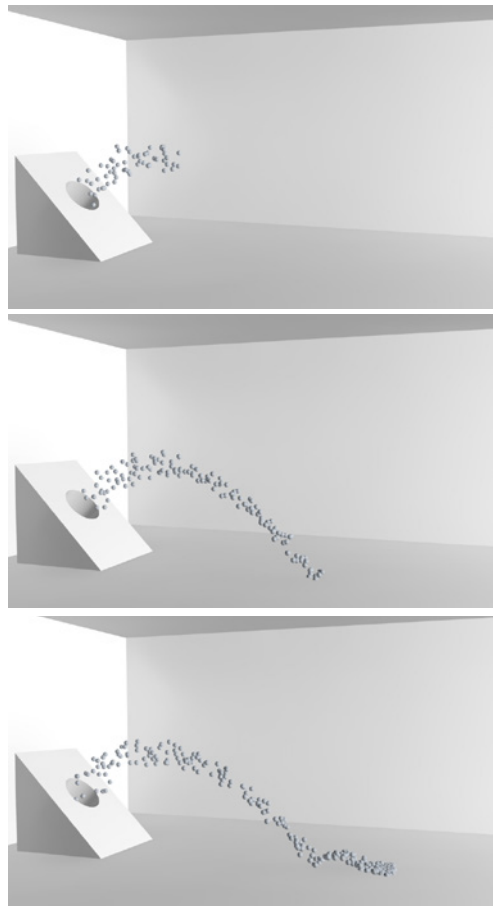


Figura 4.48 Sistemas de partículas. Ejemplo 1. Emisor elemental. Resultado final: a) Frame *20/100,* b) Frame *50/100,* c) Frame *100/100.*

Speed. Speed/Variation: 300/0 (pero experimentar con diferentes valores).
Shape. 3D/Sphere, 80-sides. Size: 6,0.
Force. Seleccionar las dos que hemos creado. Reducir la influencia a un 100 %.
Display. Geometry. 100 % visible. Para experimentar con la dinámica escoger dots y un 50 % visible para sobrecargar menos la máquina.

7 Desde la escena, editar los iconos de *Gravity* y *Drag* hasta conseguir que las partículas sigan una curva más o menos parabólica desde que salen del agujero hasta que tocan el suelo (desplazar el *time slider* para comprobarlo). En el ejemplo he usado los parámetros siguientes: *Gravity Strength*: 2,0. *Drag*: *Linear damping* con 0,0 %, 5,0 %, 35 % (X Axis, Y Axis, Z Axis).

Las partículas se moverán correctamente pero al llegar al suelo lo atravesarán, por lo que necesitamos que lo detecten y reboten. Para ello hay que añadir un detector de colisiones. Tenemos dos disponibles: *Collision* y *Collision spawn*. El primero detecta la colisión y rebota con mayor o menor impulso, según los parámetros. El segundo crea nuevas partículas después de la colisión, en una cantidad que dependerá también de los ajustes que introduzcamos.

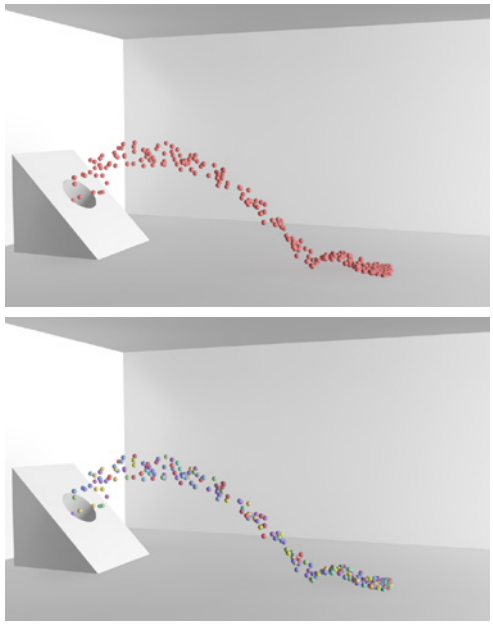


Figura 4.49 Sistemas de partículas. Ejemplo 1. Emisor elemental. Asignación de materiales: a) *Material único,* b) *Material multisubobjeto 12 submateriales.*

8 Añadir un test de tipo *Collision*. Editar sus parámetros y seleccionar el *Deflector* que hemos creado. En el grupo *Test true if particle*, seleccionar la opción *Collides* y la velocidad predeterminada, *Bounce* (otra opción sería *Is slow after collision*" y asignarle una velocidad menor que la especificada, o bien *Is fast after...*, etc). Con la opción que hemos escogido, *Bounce*, la velocidad será la especificada en el parámetro correspondiente del *Deflector* que, en el ejemplo, se ha fijado en 0,5, con una variación de 12 % y una fricción de 35 %.
9 Comprobar el resultado y hacer ajustes adecuados para que, después de rebotar, las partículas rueden un poco y se detengan.

El último paso es asignar materiales a las partículas. Podemos asignar un material único, con lo que todas las partículas tendrán el mismo aspecto, o un material con diferentes submateriales que pueden mostrarse de diferentes modos.

10 Entrar en el editor de materiales y crear un material de tipo multisubobjeto de 12 submateriales (o una cifra mayor o menor, según lo que interese). Asignar al primero un material de tipo *Arch&Design*, mate y con un color cualquiera. Copiar este material a los restantes submateriales y cambiar el color de cada uno de ellos (esto puede hacerse desde el nivel superior que muestra un icono con el color principal de cada submaterial).
11 Mantener el editor de materiales abierto. Entrar en *Particle view* y añadir un operador de tipo *Material static* a la lista de eventos, antes de *Display*. Editar sus parámetros. Arrastrar el material *multi* que hemos

creado, como instancia, sobre el botón *None* de *Material static*.

12 Marcar la opción *Assign material ID* y probar las tres alternativas principales que hay disponibles: *a*) con la opción *Material ID* podemos dar un número y todas las partículas se representarán con el submaterial correspondiente a ese número, *b*) con la opción *Cycle*, los submateriales se asignarán a las partículas siguiendo el orden de submateriales y el orden de emisión de las partículas (podemos ver el ID correspondiente a cada partícula si, en *Display*, marcamos la opción *Show particle IDs*) y según el número asignado en el parámetro inferior *sub-materials* (en principio dar el mismo número, 12, que los que hemos creado, *c*) con la opción *Random* las asignaciones son aleatorias entre el número de submateriales dados por el siguiente parámetro (de nuevo, 12 como máximo en nuestro ejemplo).

Ejemplo 2. Distribución de partículas sobre una superficie

Lo que sigue es un procedimiento general para dispersar objetos sobre una superficie. Se pueden utilizar *proxies*, como veremos en un ejemplo similar pero más complejo, utilizando mapas, pero en este ejemplo utilizaré un objeto simple. En la primera parte del ejemplo se puede utilizar indistintamente un plano o una superficie irregular pero crearé una superficie irregular para que el ejemplo sea más general.

1 Crear una superficie “base” y otra “baselrregular”), como la de la figura 4.50 (en el ejemplo he creado un plano de 500 x 500, con 32 segmentos por lado y le he añadido un modificador *Noise* con una fuerza de 30 unidades en la dirección *z*). Seleccionar la primera y ocultar la segunda para las primeras pruebas y luego hacer lo contrario, ocultar la primera y seleccionar la segunda para las siguientes pruebas.

2 Crear un cono para simular un tallo de hierba. El del ejemplo tiene 6 lados, un radio base de 0,6 y una altura de 25 unidades (cm). Crear una copia y añadirle un modificador *bend* para inclinarlo. Copiarlo otras dos veces y cambiar los ángulos y la dirección. Renombrar estos objetos como “cono”, “hierba1”, “hierba2” y “hierba3”.

3 Abrir *Particle view*. Hacer BDR (botón derecho ratón) / *New / Particle system / Standard flow*. Configurarlo como sigue, substituyendo o borrando los operadores que no interesen:
Birth. Emit Start/Stop > 0/0 (todas las partículas se generarán al inicio). Amount > 500.
Position Object. Add > “base”.
Rotation. Orientation Matrix > Random Horizontal. Divergence 7,5.
Shape Instance. Particle Geometry Object > “cono”. Scale > 100 %. Variation > 50 %.
Display. Type > Geometry. Al escoger esta opción, el plano deberá aparecer cubierto por instancias del objeto con variaciones de rotación y tamaño. Para verlos todos, seleccionar PF Source y, en Quantity multiplier cambiar el valor de Viewport (50 %) por 100 %.

Este es el procedimiento básico, que será suficiente en casos en los que la superficie sea plana. Lo que sigue introduce posibilidades adicionales de ajuste.

4 En *Position object/ Location*, probar diferentes posibilidades además de la predeterminada (*surface*). Antes, aumentar el número de segmentos del plano y convertirlo a malla poligonal: *a*) En lugar de la opción predeterminada (*surface*), probar *Vertices*, *Edges* o *Selected vertices* y *Selected edges*. Una opción corriente es utilizar la opción *Selected faces* y crear un dibujo de distribución añadiendo caras a la selección (con clic+Ctrl o la herramienta de selección de lazo); *b*) Con *Surface offset* se controla la distancia máxima y mínima a que se dispersan los objetos; *c*) Con *Spacing* se controla la distancia entre las partículas a partir de los valores *dis-*

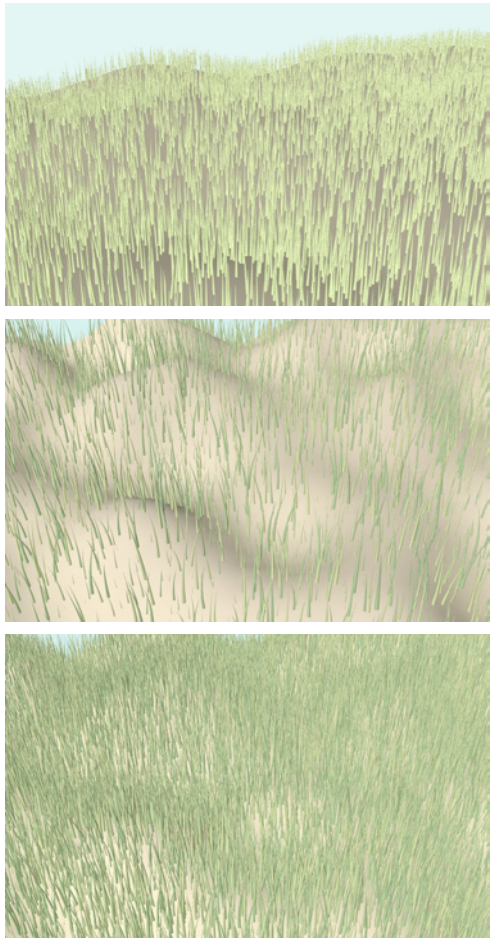


Figura 4.50 Sistemas de partículas. Ejemplo 2. Distribución de un objeto sobre una superficie irregular: a) *Conos simples,* b) *Tallos,* c) *Tallos con mayor densidad.*

tance (al aumentar este valor aumenta la regularidad de la distribución) y *attemps max* (también aumenta la regularidad). La cantidad de partículas también influye, obviamente, en la distribución. Jugar con estos tres valores; *d*) con *Density by material* podemos utilizar un mapa de bits en blanco y negro para controlar la distribución. Las zonas negras actuarán como zonas de exclusión. Utilizaremos esta alternativa más adelante, con mapas.

5 Para hacer que la orientación de los objetos dispersados se oriente con la normal a la superficie, cambiar el objeto de referencia "base" (plano) por "base irregular". Luego insertar un operador *Speed by surface* a continuación de *Position object*. Editar sus parámetros y, en *Surface geometry*, seleccionar "baseIrregular". En la sección *Direction*, seleccionar "Surface Normals".
6 Editar el operador *Rotation* y cambiar *Random horizontal* por *Speed space*. Cambiar el ángulo del eje Y por 90.

Con estos dos cambios el objeto se orientará según las normales a la superficie. Ahora que hemos visto cómo controlar la posición y la orientación del objeto, podemos substituir el cono por los tallos de hierba y añadir un material más adecuado.

7 Editar el operador *Shape instance* y substituir el objeto anterior, "cono", por "hierba1". La superficie quedará cubierta por el primer tallo que hemos creado.
8 Seleccionar el "Event 001" y hacer *BDR/Copy* y, luego, *BDR/Paste*. Así se creará un "Event002" idéntico al anterior. Borrar el operador *Birth* pues conectaremos este evento al *Birth* del anterior.
9 Añadir un operador *Split amount* al *Event 001*, justo a continuación del operador *Birth*. Crear una conexión desde este operador al *Event 002*.
10 En el *Event 002* seleccionar el operador *Shape instance* y cambiar la forma "hier-

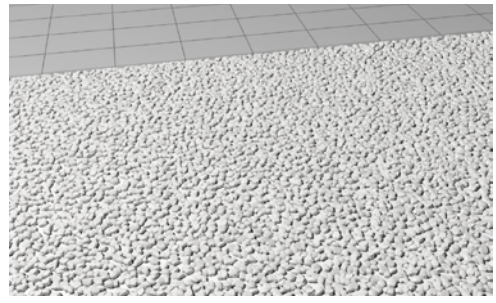


Figura 4.51 Sistemas de partículas. Ejemplo 2. Distribución de guijarros sobre un plano.

ba1" por "hierba2". En *Position object/location*, activar *Surface offset* y dar valores adecuados a *Min*, *Max*, para que el segundo tallo se desplace con respecto al anterior. Para ver mejor los resultados en el visor cambiar el color de *Display* en los dos eventos.

11 Editar los parámetros de los dos eventos y ajustar la escala y la variación de la forma.

Podría repetirse la operación y crear otro "Event 003" y cambiar la forma por "hierba3". Por otro lado, hay más variaciones que se pueden introducir probando diferentes parámetros. Pero me limito a describir lo principal. El último paso será crear un material adecuado para la hierba.

12 Crear un material multisubobjeto con unos 4 submateriales. Asignarles colores verdosos con variaciones de tono.

13 Arrastrar un operador de tipo *Material static* a los eventos. Luego arrastrar el material que hemos creado, desde el editor de materiales, sobre el botón "none" de este material. Marcar la opción *Assign material ID*. Y en las opciones, escoger *Random* con un número de submateriales igual al que hemos creado, 4.

14 Comprobar el resultado y cuando sea satisfactorio, aumentar el número de partículas a un valor adecuado al caso. En la figura 4.50 se han utilizado 12.000.

Más adelante veremos cómo pueden mejorarse estos resultados utilizando mapas de distribución y mapas que simulen hierba.

El mismo procedimiento se puede utilizar para simular cosas tales como gravilla. En este caso habrá que crear previamente tres o cuatro guijarros diferentes buscando un compromiso entre el menor número posible de caras y una representación convincente. Una vez que se cuenta con estos modelos el procedimiento es idéntico que en el caso anterior. La figura 4.51 muestra un ejemplo simple del tipo de resultados que se pueden conseguir.

Ejemplo 3. Lluvia y nieve

Este ejemplo muestra cómo utilizar un sistema de partículas para simular lluvia. También puede utilizarse el mismo procedimiento para simular gotas de agua, del tipo que sea, o nieve o fenómenos similares.

En este caso, en lugar de *Particle view*, utilizaremos un sistema simple que tendrá la ventaja, en este caso, de que podemos añadir con mayor facilidad un desenfoque de movimiento (*motion blur*) a las partículas sin necesidad de recurrir a métodos más complejos.

1 Crear un sistema de partículas de tipo *Spray* (panel *Create*/ *Particle systems*) pinchando y arrastrando sobre un visor en planta. Editarlo y darle unas dimensiones adecuadas para que abarque la escena que vamos a representar. Seleccionarlo y moverlo para colocarlo centrado sobre la zona que sea. Rotarlo ligeramente para que las partículas caigan con una cierta inclinación con lo que no será necesario añadir fuerzas. Comprobar que el número de *frames* es 100.

2 Editarlo y configurarlo más o menos como sigue (los valores son más o menos los utilizados en este ejemplo, con dimensiones en centímetros. Modificarlos para adaptarlos a las unidades utilizadas y a lo que interese):
Viewport count> 500. Render count> 4000
Drop size> 2.5. Speed> 18. Tipo> Drops, Tetrahedron.
Timing > -100/100 (Start/Life)

3 Desplazar el *time slider* para comprobar el resultado y hacer un *render*. En función de los resultados, hacer ajustes (tamaño de las partículas, inclinación, cantidad, etc). Repetir el *render* para comprobar el resultado que deberá ser más o menos similar al de la primera imagen de la figura 4.52.

4 Cambiar el color de fondo por un azul oscuro.

5 Crear un material *Standard* con las siguientes características y asignarlo al sistema de partículas: *Diffuse* > blanco (HSV 0,0,248), *Self illumination* > 95.

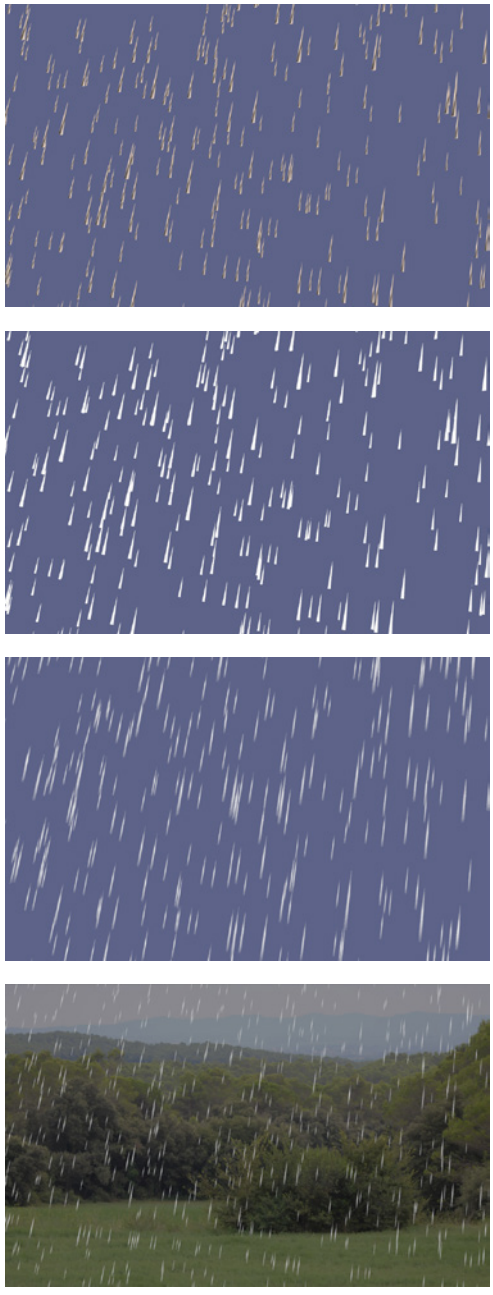


Figura 4.52 Sistema de partículas. Ejemplo 3. Simulación de lluvia: a*) resultado inicial,* b*) resultado con el material asignado,* c*) resultado con el material asignado y* motion blur, d*) resultado con un mapa de fondo.*

6 En *Render setup* / *Renderer* / *Camera effects*, en el grupo *Motion Blur*, marcar la casilla *Enable* y la casilla *Blur all objects*. Hacer un prueba con los valores predeterminados, 0,5, -0,25, 1 ,5 (*Shutter duration*, *Shutter offset*, *Motion segments*, *Time*). Luego ajustarlos en función de los resultados. El principal valor a ajustar es *shutter duration* que afecta al mayor o menor desenfoque. En el ejemplo se ha cambiado a 0,7. El resto de parámetros se han dejado tal cual.

Shutter duration tiene el mismo efecto que tendría mantener el obturador de velocidad de una cámara real abierto durante más o menos tiempo. Con 0,0 no hay desenfoque. Con valores mayores el desenfoque aumenta. *Shutter offset* (*frames*) sitúa el inicio del efecto. En principio conviene hacer que empiece un poco antes, de ahí el valor predeterminado negativo (-0,25) que no suele ser necesario cambiar. *Motion segments* afecta a la precisión del resultado pero en un ejemplo sencillo la diferencia será inapreciable. *Time samples* afecta principalmente a los materiales pero, como en el caso anterior, no es necesario ajustarlo a no ser que estemos elaborando una animación. Con texturas complejas o reflejos puede ser necesario aumentarlo. Pero el tiempo de cálculo también aumentará.

Si se quiere utilizar este sistema con los métodos más generales de *Particle view* una posibilidad que funcionará bien en la mayoría de los casos es alargar artificialmente el objeto y aplicar un desenfoque de movimiento global. Esto se puede comprobar con facilidad con el siguiente ejercicio elemental:

1 Crear un sistema de partículas simple y cambiar el tipo de forma a esfera.
2 Añadir un operador *Scale*. Desactivar la opción *Constraint proportions*. Cambiar la escala en X, o en el eje que convenga para que las esferas se alarguen en la dirección del flujo.
3 En *Render setup/ Renderer/ Camera effects*, activar *Motion blur* con la opción

Blur all objects y dar un valor adecuado a *Shutter duration*.

Objetos proxy. Con sistemas de partículas. Con herramientas de pintura de objetos

Otra posibilidad que puede aplicarse a diferentes casos de interés en simulación de modelos complejos con diferentes materiales, es utilizar objetos proxy.

Un **proxy** es, en general, un substituto de un objeto que lo representa de modo "aproximado" o "próximo" (de ahí el término). En los programas orientados a objetos es una clase que actúa como interfaz de un conjunto de objetos o recursos diversos. Y en programas de simulación es un objeto que representa de un modo simplificado un objeto más complejo, con el subsecuente ahorro de memoria y facilidad para interactuar con la escena.

En mental ray, como en otros programas de simulación, es frecuent el uso de un *binary proxy*, una serie de datos ligados a un prisma envolvente (*bounding box*) que no se procesan hasta que un rayo trazador no entra en contacto con el prisma, lo que supone que podemos interacturar con facilidad con el proxy hasta que se invoca el proceso de *rendering*.

Utilizar proxys requiere algo de trabajo previo pues hay que crear el objeto real y, luego, el proxy con la conexión al objeto real. Pero la ganancia en tiempo y eficacia, cuando se van a utilizar múltiples objetos de características similares, compensa de sobras el esfuerzo. El ejemplo que sigue ilustra el uso de proxys como alternativa a la creación de hierba con sistemas de partículas. El mismo procedimiento puede aplicarse a otro tipo de simulaciones, guijarros, árboles (con mapas), etc.

1 Utilizar la misma escena que en el ejemplo anterior o crear una variante sencilla. Por ejemplo, un plano de unos 400 x 400 cm, y un tallo de hierba creado a partir de un cono con un radio de unos 0.5 cm, una altura de 15 cm y unos 10 segmentos para los lados y unos 8 para la altura. Añadirle un modificador *Bend* y darle algo de inclinación. Luego crear varias copias, cambiar la inclinación, moverlas y rotarlas alrededor del tallo original.

2 Añadirles tres o cuatro materiales con diferentes tonos de verde. El resultado final deberá ser similar al de la figura 4.53.

3 Seleccionar uno de los tallos. Convertirlo a malla poligonal. Presionar el botón *Attach* y seleccionar el resto de los tallos. Así obtendremos un objeto poligonal complejo con un material multisubobjeto de tres o cuatro submateriales (también se puede asignar el material multisubobjeto *a posteriori*: véase el apartado correspondiente más adelante). Renombrarlo como "tallos". No estará de más en este punto guardar el archivo con otro nombre pues a partir de lo que sigue no se podrán hacer cambios en este objeto.

4 Ir al panel de *Create / Geometry / mental ray*. Presionar el botón *mr Proxy* y pinchar y arrastrar en la escena, junto a los tallos, para crear un objeto proxy que aparecerá como un cubo alámbrico. Mantener seleccionado este objeto. Ir al panel *Modify*. En el grupo *Source object* presionar el botón *None* y seleccionar "tallos". El nombre del botón cambiará para mostrar el nombre del objeto seleccionado como fuente.

4 Pulsar el botón *Write object to file*. Dar un nombre y una ubicación para el proxy (por ejemplo "tallos.mib" en la carpeta del proyecto activo). Después de grabar este archivo, con la extensión .mib, se abrirá un cuadro de diálogo con varias opciones. Dejar las predeterminadas: *Current frame* (las otras opciones se utilizan para animaciones), *Include only source object in preview* y *Automatic zoom extents*. Al terminar (se hará un pequeño *render*), el cubo que representaba el objeto proxy cambiará para mostrar una serie de puntos que representan el objeto original. Puede aumentarse el número de puntos editando el proxy y variando el valor de *ViewPorts* (128) desde el grupo *Display*.

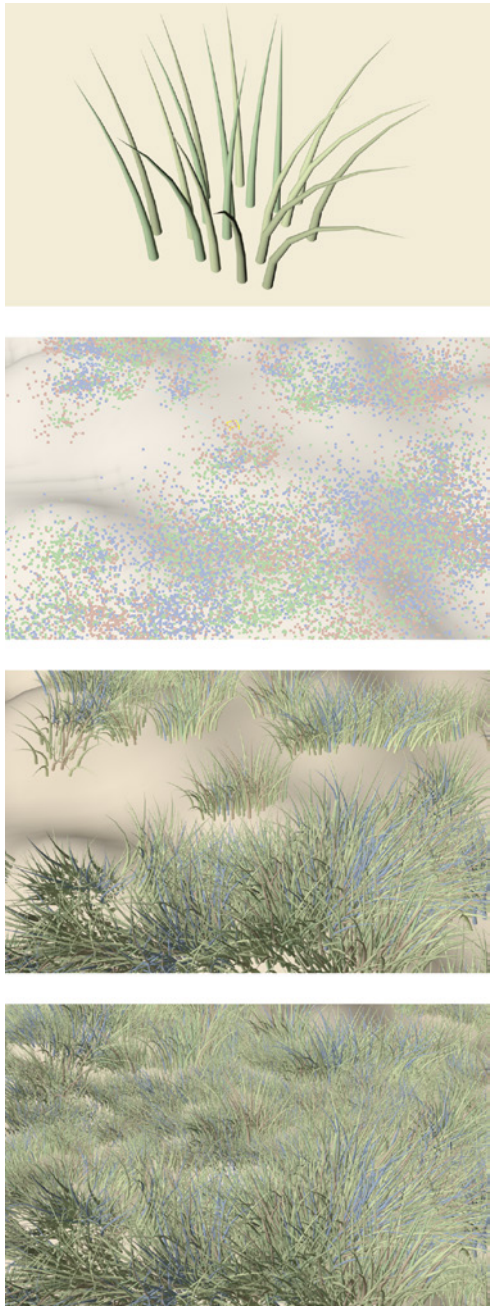


Figura 4.53 Distribución de tallos por medio de proxys *y la herramienta* Object paint*: a) vista del objeto original, b) vista de la escena con los* proxys *distribuidos, c) resultado final.*

A partir de aquí, desde este panel, se podría cambiar el tamaño con el parámetro *Scale* además de la apariencia (haciendo que se muestren más o menos puntos o la caja envolvente, *bounding box*, o ambos). También se pueden hacer múltiples copias o asignarle diferentes materiales. Pero todo esto lo haremos con métodos más interesantes y más eficaces.

La primera posibilidad es, simplemente, repetir el proceso que ya hemos visto y dispersar los tallos de hierba proxy igual que hemos hecho con los tallos simples del ejemplo anterior utilizando *Particle view*. No hay mucho que añadir a lo anterior así que doy por asimilado este método.

Pero hay una segunda posibilidad: pintar directamente con objetos proxy. Para ello podemos utilizar un recurso específico de 3ds Max, que es *Object paint*. Esta herramienta permite "pintar" objetos 3D sobre una superficie. Se pueden pintar objetos no proxy pero, en casos como este, para descargar al programa del procesamiento interactivo de miles de caras es recomendable usar objetos proxy.

La descripción que sigue es muy breve por lo que recomiento al lector que consulte la ayuda del programa antes de seguir pues las indicaciones telegráficas que daré se entenderán mucho mejor si se ha hecho una prueba previa.

Si se quiere contar con más objetos y más materiales, seleccionar el objeto "mrProxyTallos1" y copiarlo dos o tres veces y renombrarlos como "mrProxyTallos2", "mrProxyTallos3", etc. Luego, desde el editor de materiales, captar el material del primer proxy dos o tres veces ("tallos2", "tallos3", etc.), asignarlos a los nuevos proxys y cambiar los tonos de la hierba. Pero para simplificar la exposición partiré de la base de que solo tenemos un proxy. A partir de aquí, continuar como sigue.

1 Abrir la herramienta *Object paint* que forma parte de una barra de herramientas especial, la "cinta" (*Ribbon*). Si no estuviera abierta, ir al menú *Customize/Show UI / Show ribbon*. Aplicar los comandos como sigue (de izquierda a derecha):

a) *Edit object list*. Abrir este panel y seleccionar el proxy (o, en su caso, los diversos proxys) que hemos creado. El objeto u objetos seleccionados se incluirá en esta lista y podemos ir seleccionando uno u otro, según lo que interese en cada caso.
b) Marcar la opción adjunta, *All, Randomly* para que al pintar se creen aleatoriamente instancias del objeto u objetos seleccionados.
c) Descolgar la lista encabezada por *Align*. Desactivar *Follow stroke*.
d) Dar un valor adecuado a *Spacing* (habrá que hacer tanteos para que los tallos no queden ni muy juntos ni muy separados).
e) En el grupo *Scattering* se puede dar un valor aleatorio de dispersión para que los objetos no caigan sobre el punto marcado. Pero en este caso no será necesario.
f) En el grupo *Rotate settings*, en Z, activar la opción *Random Z*.
g) En el grupo *Scale settings*, descolgar la llista adjunta y escoger *Random*. Con esto cambiarán los valores *xyz* que mostrarán un rango (predeterminado a 20 % / 200 %) Ajustar este rango a lo que interese o hacer pruebas previas.

4 Activar el pincel (a la izquierda de la cinta) y comenzar a pintar. Al desplazar el pincel sobre la superficie se irán generando instancias de los proxys con variaciones aleatorias de color, rotación y escala.
5 Al hacer un *render*, los *proxys* serán substituidos por el objeto original con resultados que deberán ser similares a los de la figura 4.53.

Otros *shaders*

La variedad de *shaders* que pueden encontrarse en los diferentes programas de simulación desafía cualquier resumen. En este apartado me he centrado en los principales que se encuentran con mental ray. Pero si se hace una búsqueda por internet con las palabras clave “software” y “shader”, donde “software” sería un programa determinado, por ejemplo V-Ray, Maxwell, Blender, etc., se encontrarán un gran número de materiales muy similares a los que hemos visto y que pueden agruparse en categorías semejantes.

También hay *shaders* que nacen y mueren pues son substituidos por otros más eficaces. Un buen ejemplo sigue siendo el propio mental ray, que ha creado un gran número de *shaders* que en el pasado proporcionaban alternativas muy eficaces para simular materiales especiales. Estos *shaders* se han ido integrando en materiales más complejos como los que ya hemos visto.

Hay una fuerte tendencia a utilizar *shaders* con nombres de materiales concretos, tales como “cerámica”, “metal”, “vidrio”, etc. Y múltiples bibliotecas, ligadas a programas específicos que proporcionan estos materiales. Sin embargo, la gran mayoría de estos *shaders* no son sino configuraciones de parámetros como los que ya hemos visto, de un material genérico. Quienes los utilizan no tienen acceso directo a estos parámetros sino al modo en que se han definido por los creadores de las bibliotecas, por lo que su uso resulta más rígido. Es preferible crear alternativas propias pues, aunque de algo más de trabajo al principio, el control que se obtendrá a medio plazo será mucho mayor.

Por otro lado, la simulación de gases, nubes, humo, niebla, etc., es difícil de llevar a cabo por medio de un sistema basado en polígonos, es decir, en representación de superficies. Se requiere, en general, un sistema que utilice *voxels*, particiones sistemáticas de un volumen. Medios tales como *Parti volume* son una aproximación que nunca puede dar tan buenos resultados como los que se consiguen con estos sistemas.

Quien esté interesado en estos temas debería probar programas especializados en este tipo de efectos. Algunos de los más conocidos por estas fechas (2014) son FumeFX, que es particularmente efectivo para simular fuego, humo y gases de todo tipo o Vue 8 xstream que es particularmente efectivo generando nubes. También conviene recordar que los mejores efectos especiales

que vemos en el cine o la televisión suelen ser debidos a *shaders* escritos expresamente por programadores profesionales.

Hipertexturas

Como ya hemos visto en la 1ª parte, las "hipertexturas" introducidas por Perling y Hoffert en 1989 permiten modelar objetos tales como fuego, relieve aleatorio, deterioro superficial o pelo. En estos casos nos encontramos en una zona fronteriza entre lo que es simulación geométrica y simulación material. Y de hecho, tal como se explicaba con algo más de detalle en el apartado correspondiente, el procedimiento es delimitar una estrecha zona adyacente a la superficie y especificar una función que defina una densidad inicial básica, que es es modificada por diversas funciones de modulación de la densidad.

Las hipertexturas se utilizan a menudo para simular pelo. Y variando un poco los parámetros y el color, sirven también para simular hierba.

Pero acabamos de ver que esto también puede simularse con sistemas de partículas. Así que, en muchos casos, no quedará claro cuál es el sistema más recomendable y, en última instancia, puede que dependa no solo de los casos sino de las preferencias personales.

Doy un ejemplo simple para ilustrar el tipo de procedimiento que se puede seguir con esta técnica.

El modificador *Hair*&*Fur* que ha incluido 3ds Max en sus últimas versiones y que es similar al utilizado en Maya, da resultados lo suficientemente buenos como para que siga siendo una alternativa interesante.

Este modificador está diseñado expresamente para simular todo tipo de pelos y peinados pero se puede utilizar facilmente para algo relativamente más simple como simular hierba que, en arquitectura, es un tipo de simulación bastante mas corriente que la del pelo. Para comprobar su funcionamiento, hacer lo siguiente.

1 Crear un plano de unos 100 x 100 m y 4 x 4 segmentos.
2 Añadir a una luz directa que simule el Sol por la izquierda, con sombras de *shadow map* y una luz focal de poca intensidad, sin sombras, por la derecha para suavizar las sombras de la primera.
3 Aplicar al plano un material de un color gris amarillento muy claro.
4 Aplicar al plano un modificador *Hair*&*Fur*. El plano aparecerá cubierto de líneas curvas de altura e inclinación aleatorias. Editarlo para modificar los valores predeterminados.

Este modificador incluye varias secciones. Para lo que nos interesa se puede prescindir de las tres primeras: *Selection*, *Tools* y *Styling*, que se utilizan para simular pelo con varios estilos de peinado. También podemos prescindir de las secciones *mr Parameters* (que solo incluye un parámetro, para aplicar un *shader* de mr), *Dynamics* (que se utiliza principalmente en animación). La sección *Display* sirve tan solo para que el visor muestre más o menos elementos que los que aparecerán en la representación final y puede convenir aumentarlo algo por encima de su valor predeterminado (2 %) o hacer que muestre la geometría real, en lugar de un esquema, activando la opción correspondiente.

Hay otras dos secciones que tienen un interés muy relativo para simular hierba. *Kink parameters* incluye parámetros para hacer que la hierba se rice ligeramente en sentido longitudinal, lo que tampoco nos interesa. *Multi strand* podría tener algo más de interés pues crea grupaciones, como matas, generando elementos suplementarios alrededor del principal. El valor predeterminado del contador es 0 en ambos casos por lo que no hay que preocuparse de ellas si no se usan.

Y en las tres secciones que quedan solo nos interesarán los parámetros que se describen a continuación.

En la sección *General*, el parámetro *Hair count* establece el número de elementos (el valor es aproximado y la densidad se ajusta

al tamaño de los polígonos); *Hair segments* la resolución de los elementos (un valor inferior los hará más rígidos y un valor superior los hará más flexibles pero aumentará el tiempo de cálculo), *Hair passes* controla la transparencia lo que redunda en mayor o menor suavidad; *Density* es un multiplicador del contador que varía entre 0 y 100; *Scale*, la longitud de las hierbas; *Random scale*, la aleatoriedad de la distribución (si se aumenta el desorden puede hacer que algunas hierbas desaparezcan); *Root thick* y *Tip thick* el grosor de la base y la punta.

En la sección *Material parameters* hay parámetros para controlar directamente el material. Con *Root color* y *Tip color* se define el color de la raíz y la punta. *Hue variation* y *Value variaton* establecen grados de variación del tono y la luminosidad.

En la seción *Frizz parameters* hay parámetros adicionales para modificar la curvatura y la orientación de los elementos. *Frizz root* y *Frizz tip* fijan el ángulo con que se curva el elemento, desde la raíz y en la punta. Los valores de frecuencia de *Frizz* (*XYZ*) *freq*. determinan la frecuencia con que esta curvatura se da en cada eje.

5 Configurar el modificador del siguiente modo para la primera prueba:
General parameters.
Hair count: 1.000.
Hair segments: 1. *Hair passes*: 1.
Scale: 15. *Rand*. *scale*: 0.
Root/*Tip Thick*: 2,0
Material parameters.
Tip color, *Root color*: marrón verdoso.
Hue variation: 0. *Value Variation*: 0.
Frizz Parameters.
Frizz root: 45. *Frizz tip*: 90.
Frizz (*XYZ*) *freq*: 14,0.

6 Hacer un *render*. El resultado debería ser como el de la primera imagen de la figura 4.54. Con valores tan bajos puede apreciarse la forma y la distribución de los elementos y ajustarlos.

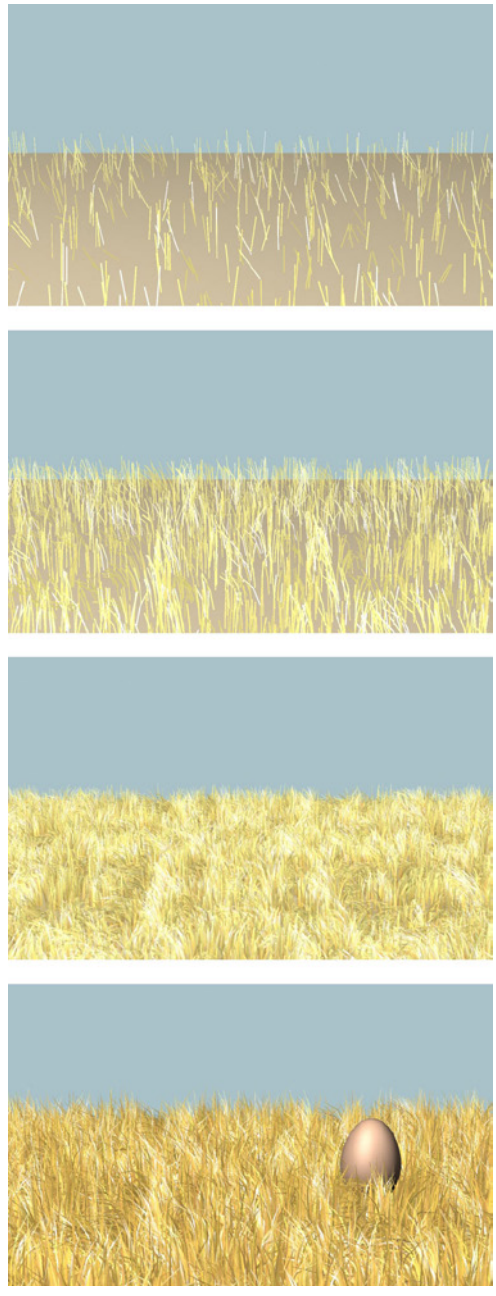


Figura 4.54 Hipertexturas. Resultados obtenidos variando progresivamente el número de elementos y la densidad.

7 Probar a ir aumentando los valores. El resultado debería ser como el de la segunda imagen de la figura citada.
8 Configurarlo de modo definitivo con algo parecido a lo que sigue (solo indico los valores modificados) para obtener un resultado como el de la tercera imagen.
General parameters.
Count: 20.000. *Density*: 100
Se*gments*: 4. *Passes*: 2.
Scale: 15. *Rand. scale*: 40.
Root thick: 4,0. *Tip thick*: 2,0
Material parameters.
Tip color: marrón verdoso oscuro,
Root color: marrón verdoso claro.
Hue variation: 10. *Value variation*: 50.
Frizz parameters.
Root: 15. *Tip*: 130.
XYZ freq: 14,0.

9 Colocar un objeto en el medio (una especie de huevo de dinosaurio en la figura) para comprobar que el modificador resuelve adecuadamente los solapamientos. Así se obtendría algo similar a la imagen final de la figura 4.54.

Esta última imagen se ha obtenido con la siguiente configuración y dos luces: una mr Spot de 0,5 de intensidad (multiplicador) con sombras por la izquierda y una igual, secundaria, de 0,2 sin sombras por la derecha.

General parameters
Count: 35.000
Segments: 5
Scale: 6,0. Rand scale: 12,0
Root thick: 2,5. Tip thick: 0,0
Material parameters
Tip color: hsv 58, 137, 145
Root color: hsv 61, 145, 95
Hue var: 20,0. Value var: 20,0
Specular: 10,0. Gloss: 12,0
Spec tint: hsv 44, 44, 250
Frizz Parameters
Root: 12,0. Tip: 120,0
X freq, Y freq, Z freq: 15,0, 24,0, 18,0

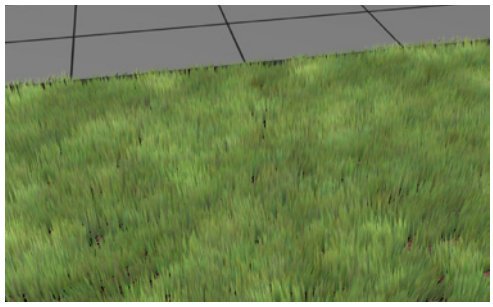


Figura 4.55 Otro ejemplo de simulación de hierba con hipertexturas.

4.4 Mapas

El término *mapa* es una traducción algo equívoca del inglés *map*, que significa "mapa" pero también "proyección". En castellano un mapa es también una proyección pero este sentido queda más bien oculto por su significado habitual que se relaciona con los viajes y la geografía antes que con la geometría y las matemáticas.

Aunque no se corresponda con el uso habitual en castellano, en lo que sigue utilizaré el término *mapa* para referirme a proyecciones geométricas del tipo que sea. Tanto por las razones citadas, como porque es un término más general y menos ambiguo que *imágenes* o *texturas*, otros dos términos alternativos corrientes.

Utilizar mapas, como recurso asociado a la simulación de un material por medio de un *shader* básico, implica tres cosas.

En primer lugar, para poder utilizar mapas con un objeto este debe contar, previamente, con una especificación del modo en que va a incorporar las proyecciones que reciba. Esta especificación es una propiedad del objeto, independiente del mapa concreto que se le aplique.

En segundo lugar, habrá que decidir que tipo de mapa queremos utilizar, pues una opción corriente es utilizar un mapa de bits obtenido a partir de imágenes reales. Pero pueden utilizarse también mapas procedurales, generados artificialmente o combinados

de diferentes modos entre sí, como también se verá más adelante.

En tercer lugar, habrá que decidir el tipo de aplicación, a qué nivel se quiere actuar. La opción más corriente es utilizar un mapa como textura, para substituir el color propio del objeto por el color correspondiente a la parte del mapa que se relaciona con la parte correspondiente de la superficie del objeto. Pero también podemos utilizar un mapa para modificar el relieve o los reflejos o las transparencias, como hemos visto en el capítulo anterior.

En los programas de simulación se acostumbra a hablar de "tipo de mapa" en dos sentidos diferentes, lo que puede resultar confuso. Como "tipo de mapa" por su contenido, por sus características intrínsecas, o como "tipo de mapa" por su nivel de aplicación. Utilizaré la denominación "tipo" o "nivel de aplicación" cuando haya riesgo de confusión entre estos dos sentidos.

Coordenadas UVW

Por convenio, las coordenadas de proyección del espacio del mapa se denominan coordenadas UVW, donde la U equivale a la X, la V a la Y y la W a la Z del espacio geométrico del objeto. El sentido principal de esta convención es subrayar que el espacio en que se representan las proyecciones es independiente del espacio en que se representa el objeto. Y el espacio del mapa a menudo es 2D, con lo que la coordenada W no representa otra cosa que la capacidad de hacer transformaciones tales como rotar el mapa en torno a este eje. Esto cambiará el modo en que se relaciona con el objeto (la textura aplicada aparecerá girada) pero el contenido de la proyección, mientras no cambiemos los valores correspondientes a U y V, será el mismo.

Si un objeto va a recibir una proyección, lo primero que se necesita es saber cómo la va aplicar. En el caso más simple, un objeto plano y un mapa igualmente plano, la relación será directa pero aún así nos podemos encontrar con infinitas posibilidades pues los dos planos pueden tener proporciones y orientaciones distintas. Se requiere, por tanto, especificar esta relación de algún modo. Esto se hace, en los programas de simulación, por medio de métodos gráficos o numéricos o de una combinación de ambos, y con la ayuda de una representación que permita visualizar el mapa como si fuera un objeto geométrico superpuesto al objeto. En algunos programas, como 3ds Max, esta representación virtual del mapa se denomina *gizmo*.

La especificación de esta relación queda incorporada al objeto como una propiedad adicional, independiente, como decía, del mapa concreto que vaya a recibir. Antes de proseguir con los diferentes tipos de mapas, en el siguiente apartado analizaremos los diferentes tipos de proyecciones y los problemas que se pueden presentar en cada caso. Para ilustrarlos, utilizaré un mapa simple, un mosaico de cuadrados negros y blancos.

Esta especificación puede estar predeterminada o puede no existir. Los programas que generan objetos por métodos paramétricos asignan automáticamente un tipo de proyección "adecuada" a cada objeto: un plano recibirá una proyección de tipo plano, una caja recibirá una proyección de tipo caja (6 mapas planos sobre cada una de sus caras), un cilindro, una proyección cilíndrica y una esfera, una proyección esférica. Y objetos más complejos también pueden recibir asignaciones especiales, como veremos. Pero, sea porque la proyección es inadecuada para lo que interesa, sea porque el objeto ha sufrido transformaciones, sea porque se ha importado como malla poligonal desde otros programas que no incluyen esta propiedad, puede ocurrir que haya que cambiar o definir la proyección.

En este último caso, si la proyección no está definida y se hace una representación de la escena y al objeto se le ha asignado un material con un mapa, el programa no sabe cómo gestionar la proyección y o bien se representa el material con el color dado por sus parámetros básicos pero sin mapa o bien

aparecerá un aviso, un cuadro de diálogo con un texto tal como:

> "Los siguientes objetos requieren
> coordenadas de mapa y
> no se representarán
> correctamente".

Proyecciones explícitas y no explícitas

Las especificaciones de coordenadas UVW incorporadas al objeto reciben a veces el nombre de **proyecciones explícitas**. En la mayoría de los programas la asignación de un mapa a un objeto incluye varias opciones. La primera y principal es como una textura proyectada a través de un *Explicit map channel*. Esto quiere decir que la textura se proyectará a través del canal indicado por el usuario. Si el usuario no especifica ningún canal se asigna el predeterminado que, en principio, es el canal 1. Pero un objeto puede incorporar múltiples especificaciones UVW que, entre otras cosas, se utilizan, como veremos más adelante, para asignar diferentes texturas a un mismo objeto. En 3ds Max, por ejemplo, se pueden asignar hasta 99 canales a un mismo objeto.

Además de *Explicit map channel* puede haber otras opciones cuando se escoge "textura" como tipo de proyección. Por ejemplo: *Vertex color channel* (utiliza los colores de los vértices como canal: véase el apartado sobre color de vértices más adelante), *Planar from object XYZ* (usa una proyección plana que se basa en las coordenadas locales del objeto), *Planar from world XYZ* (usa una proyección plana que se basa en las coordenadas globales de la escena). Y otras que dependen del tipo de programa y que se dan como alternativas a la manipulación directa del espacio UVW, sea porque se considera preferible en algún caso (poco frecuente), sea porque el programa no cuenta con recursos para este tipo de edición (más frecuente).

Las **proyecciones no explícitas** no dependen de las especificaciones UVW. La asignación de un mapa a un objeto puede ha-

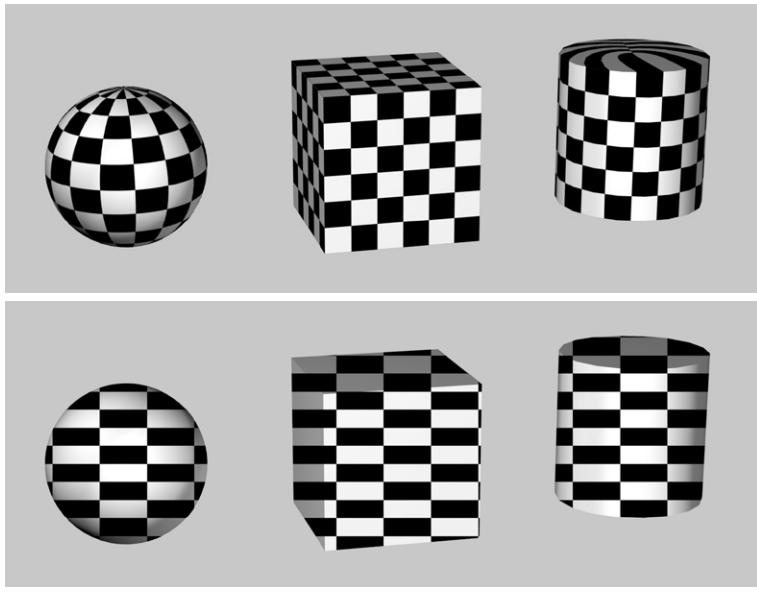


Figura 4.56 Proyecciones explícitas y no explícitas: a) *Cada objeto tiene asignada una especificación adecuada a su forma,* b) *Cada objeto tiene asignada una proyección de entorno plana (*screen*).*

cerse también escogiendo la opción “Entorno” (*Environment*) en lugar de “textura”. En este caso las opciones corrientes son *Screen* (la proyección se aplica sobre el fondo de pantalla), *Spherical environment* (la proyección se aplica sobre una esfera virtual que envuelve la escena), *Shrink-wrap environment* (la proyección se aplica sobre una esfera virtual con corrección en los polos), *Cylindrical environment* (la proyección se aplica sobre un cilindro que rodea la escena). Es decir, las mismas que tenemos, en general, para un objeto pues también se basan en un mapa de bits plano, pero en este caso se aplican al conjunto de la escena y dependen del punto de vista. Y no hay posibilidad de utilizar diferentes canales o de cambiar la orientación de los ejes UVW.

Las proyeciones explícitas pueden combinarse y, posteriormente, si interesa, desplegarse para formar una proyección unificada, como también veremos. Las proyecciones no explícitas dependen de una proyección externa y las capacidades de modificar el tamaño y la orientación son limitadas. Por otra parte, se evalúan únicamente durante el proceso de *rendering* (o de previsualización), por lo que también son más difíciles de controlar.

Las proyecciones no explícitas se utilizan generalmente para fondos. Pero también se pueden aplicar a un objeto que, en este caso, se comportaría como una máscara a través de la cual veríamos el mapa. La figura 4.56 muestra un ejemplo de este uso, que es poco habitual pero ilustra con más claridad estas diferencias.

Tipos de proyecciones

Si la imagen de que partimos es plana, como ocurre cuando se utiiza un mapa de bits, tenemos tres posibilidades principales: *a*) que el mapa se conserve plano, *b*) que se adapte a una superficie simple tal como un cilindro o la esfera, *c*) que se adapte a una superficie de curvatura compleja.

La primera posibilidad implica varias variantes que son muy importantes en la práctica pero que no ofrecen ninguna dificultad especial. La segunda tiene la dificultad que puede ser más o menos importante según los casos, y que, estrictamente, no puede solucionarse, de que habrá zonas que no quedarán cubiertas adecuadamente por el mapa. La tercera presenta dificultades técnicas especiales y las veremos más adelante.

Desplegando las variantes principales de las dos primeras posibilidades nos encontraremos con los siguientes casos principales que se ilustran en las figuras 4.57, 4.58 y 4.59:

1) Mapa plano aplicado a una superficie plana con la misma orientación. La proporción y la orientación se ajustan por medio de los controles propios de los mapas UVW.

2) Mapa plano aplicado a una superficie plana con diferente orientación. La proporción y la orientación se ajustan, como antes, por medio de los controles propios de los mapas UVW.

3) Mapa plano de tipo “box”, aplicado a un prisma desdoblándose automáticamente en seis caras (mapas planos) que se aplican a las seis caras del prisma. En este caso, si las caras no son iguales podremos ajustar las parejas paralelas pero las otras quedarán desajustadas como se puede apreciar en la imagen inferior de la figura 4.57.

Hay una posibilidad adicional que es utilizar mapas planos diferentes aplicados de modo independiente a las seis caras de un prisma mediante multiproyección. En este caso volveremos a tener pleno control del ajuste, pero a costa de utilizar los métodos de multiproyección que veremos más adelante.

4) Mapas cilíndrícos. Si el objeto es un cilindro o un objeto asimilable a un cilindro, el ajuste puede controlarse tan perfectamente como en los casos anteriores, aunque la base y la tapa superior quedarán desajustadas. Esto puede modificarse mediante variantes que incluyan un mapa adicional automático para la tapa, aunque según los casos esta modificación no servirá de nada y habrá que recurrir a proyecciones múltiples.

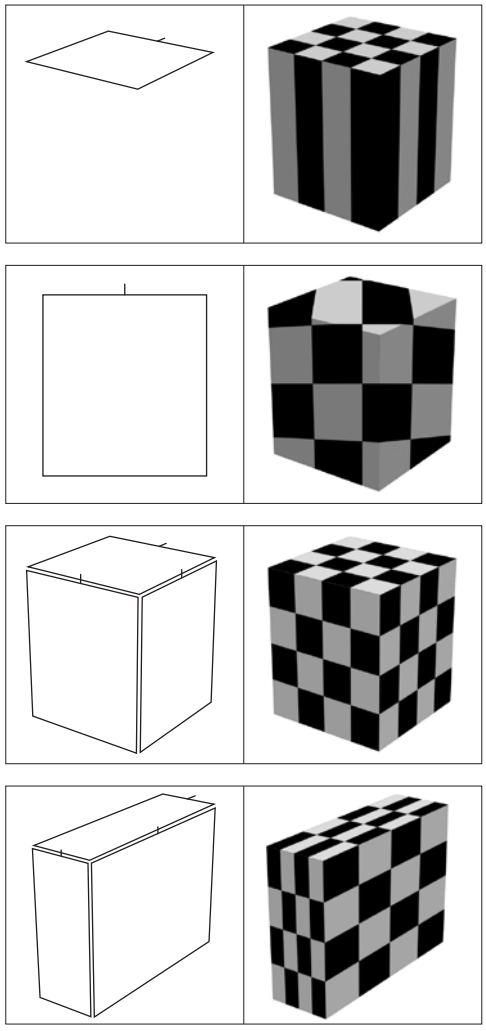


Figura 4.57 Proyecciones planas sobre un prisma.

5 Mapas esféricos. Si el objeto es una esfera o un objeto cercano, el ajuste puede controlarse bastante bien excepto en los polos. En varios programas pueden escogerse al menos dos variantes: una que crea una singularidad en los dos polos y otra que concentra la singularidad en uno de los dos polos nada más.

Además de todos estos casos pueden aparecer otras variantes, sin demasiado interés, como los tipos “por cara” (otra variedad de la proyección plana que aplica el mapa sobre cada cara triangular de la malla interna propia del objeto) o XYZ a UVW (se utiliza para hacer que un mapa procedural 3D se adapte a la superficie del objeto aunque este cambie, por ejemplo, con una deformación).

Métodos básicos de ajuste de proyecciones

Para ajustar adecuadamente un mapa a un objeto, una vez que se ha asignado como material que incorpora un mapa, hay dos métodos básicos: 1) desde la escena, editando la proyección, 2) desde el editor de materiales, editando el mapa.

1) Desde la escena, como una propiedad del objeto, editando el *gizmo* desde el panel de modificar. En 3ds Max esto se hace del modo siguiente:

Desde el panel *Modify*, asignar un modificador mapa UVW al objeto o editarlo si ya estuviera asignado. Desplegar el único subobjeto propio de este modificador, el *gizmo*, que representa el espacio del mapa, y seleccionarlo. Al seleccionarlo, queda disponible como un objeto más con lo que se pueden cambiar sus dimensiones y moverlo o rotarlo con las herramientas corrientes de transformación. El panel del modificador ofrece por añadidura, en la parte inferior, en la sección *Alignment*, varias herramientas que facilitan el ajuste: Botones *X / Y / Z* (alinean el gizmo con uno de estos ejes; *Fit* (centra el *gizmo* en el objeto y cambia sus dimensiones para que coincidan con la envolvente); *Center* (centra el *gizmo* sin cambiar las dimensiones); *Bitmap fit* (ajusta las proporciones del gizmo a las de un determinado mapa de bits); *Normal align* (al desplazar el cursor por las caras del objeto se ajusta la orientación del *gizmo* a la normal a la cara); *View align* (el *gizmo* se alinea con la vista); *Región fit* (permite dibujar directamente el *gizmo* en pantalla); *Reset* (restituye la orientación y las dimensiones ini-

ciales dadas por defecto); *Acquire* (tras activar esta opción, si se lleva el cursor sobre un objeto válido, cambia de aspecto y, al hacer un clic, se abre un cuadro de diálogo: la opción “adquirir relativo”, la más corriente, hace que el *gizmo* se sitúe en nuestro objeto con la misma posición que tenía, en coordenadas locales, en el objeto de referencia; la opción “adquirir absoluto” hace que el *gizmo* se sitúe en nuestro objeto en la misma posición, en coordenadas globales, que tenía en el objeto de referencia). Otra posibilidad muy utilizada es copiar el modificador UVW, situándose sobre y haciendo BDR (botón derecho ratón)/ *Copy* y luego editando el objeto que sea y haciendo BDR/*Paste*.

La figura 4.60, arriba, muestra el resultado (a la derecha) de aplicar un mapa de bits a un plano modificando el gizmo (izquierda) mediante las tres transformaciones básicas, mover, rotar y escalar.

2) Desde el propio editor de materiales, en el nivel de coordenadas de mapa, como una propiedad del material. De este modo se modifica la posición del mapa en relación con el *gizmo* pero el resultado, en la mayoría de los casos, sobre todo cuando las operaciones de recolocación del mapa son simples, será equivalente a lo anterior. En 3ds Max esto se hace del modo siguiente:

Desde el editor de material, seleccionar el material que hemos creado y entrar en el nivel correspondiente al mapa, en la sección coordenadas de mapa. En esta sección hay diversos controles que afectan al tamaño, a la colocación y a la orientación del mapa. Los parámetros disponibles son los siguientes. *Mosaico (Tiling)*. Por defecto está activada

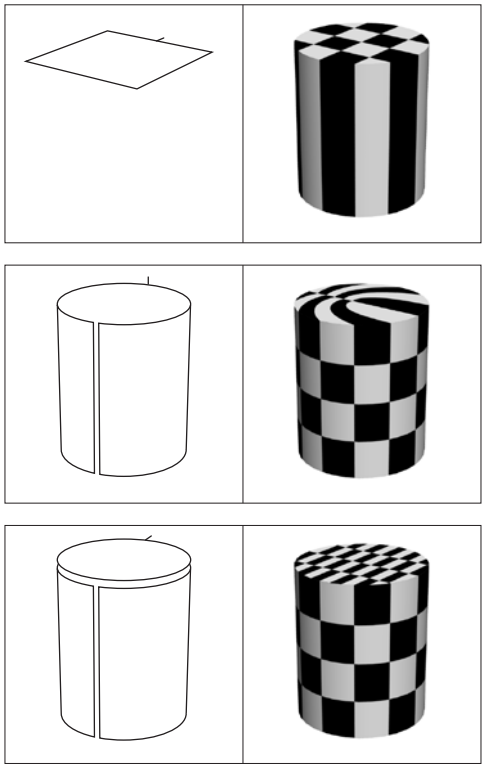


Figura 4.58 Proyecciones planas/cilíndricas.

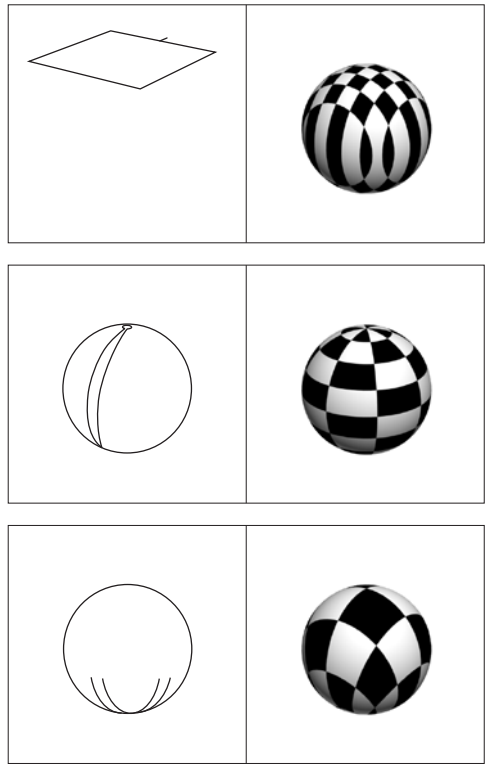


Figura 4.59 Proyecciones planas/esféricas.

con lo que el mapa se repite hasta recubrir el objeto. Si se desactiva, el mapa se proyecta como una calcomanía recortada sobre la cara correspondiente ("decal" o "crop", en otros programas). Si el valor es inferior a 1,0, el mapa rebasa el objeto (equivale a aumentar el tamaño del mapa). Si es superior a 1,0, el mapa ocupa solo parte del objeto (equivale a disminuir el tamaño del mapa). La opción *Tile* repite el mapa pero invirtiendo la repetición. Con *Desfase (Offset)* y *Ángulo* se modifican la posición y el ángulo del mapa. En el caso de un mapa 2D los valores de rotación se aplican al eje W, que corresponde al eje vertical, para rotarlo sobre el propio plano.

Hay que tener en cuenta que si se modifica el ángulo W en un mapa no cuadrado se distorsiona el resultado como puede apreciarse en la figura 4.60, abajo. Para evitar este problema hay que editar el *gizmo* y hacer que sus dos lados sean iguales. Pero en estos casos, cuando la posición del mapa debe estar dada con precisión es preferible rotar el *gizmo* directamente.

En muchos casos los mapas deben ajustarse con precisión al objeto de un modo que solo es posible si se crean mapas a medida del objeto. La figura 4.61 muestra, en la parte superior, el mapa de bits original. En la parte media se puede apreciar lo que ocurre si se inserta el mapa de modo que se ajuste al objeto: las letras quedan deformadas. Y en la parte inferior, lo que ocurre si se inserta el mapa de modo que se ajuste a la imagen original: el mapa no se deforma pero no se ajusta al objeto.

Para solucionar este problema la única solución es crear mapas *ad hoc*. Esto reduce al máximo las operaciones de ajuste y garantiza que habrá una perfecta adecuación entre el mapa y el objeto. Esto se puede hacer de dos modos:

a) Generar un mapa de bits basado en las dimensiones exactas del objeto. Por ejemplo, supongamos que tenemos un objeto del que conocemos su geometría exacta y queremos generar mapas que se ajusten exactamente a

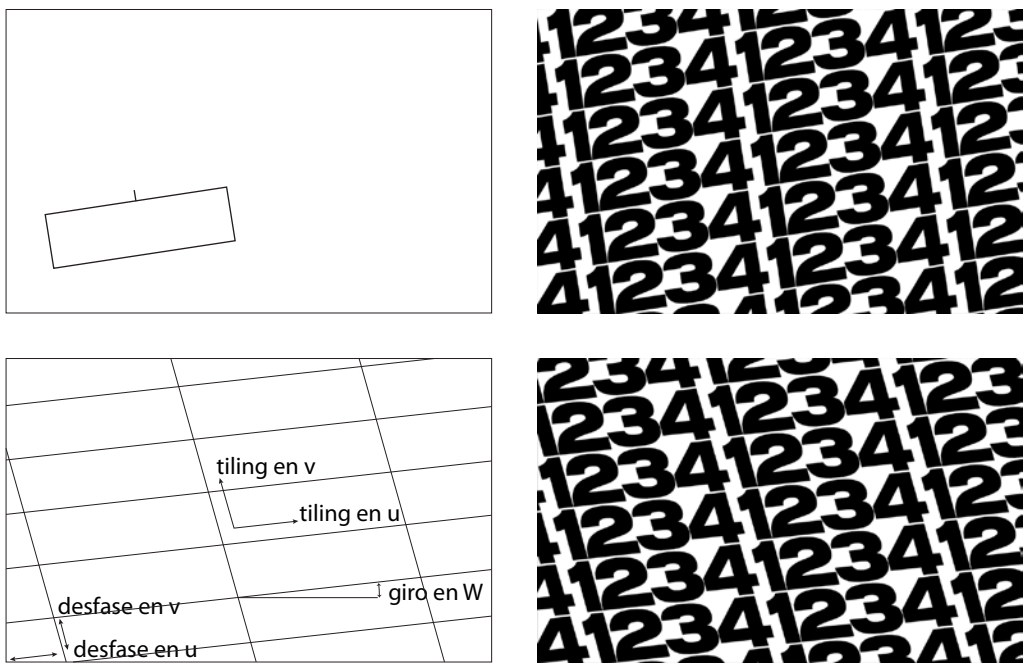


Figura 4.60 Ajuste de proyección desde la escena (arriba) y desde el editor de materiales (abajo).

esta geometría. Pongamos que se trata de un prisma de 300 x 400 x 100 unidades (ancho, largo, alto) y queremos generar una textura a la medida de cada una de sus caras visibles. Desde Photoshop o un programa similar crear (o modificar) tres mapas de bits de 300 x 400, 300 x 100 y 400 x 100 píxeles o de dimensiones proporcionales a estas, por ejemplo 1.200 x 1.600. Estos tres mapas de bits se ajustarán exactamente a las caras del objeto.

b) Generar un mapa de bits basado en una captura de pantalla a partir de una vista ortogonal al objeto desde el programa que se esté utilizando para modelar, o bien, si se necesita más resolución, mediante un *render* de un vista ortogonal a dicho plano. Luego abrir la imagen resultante desde un programa de pintura digital y recortar, editar y ajustar el tamaño de la imagen. Luego generar un mapa que se adapte a esta imagen auxiliar.

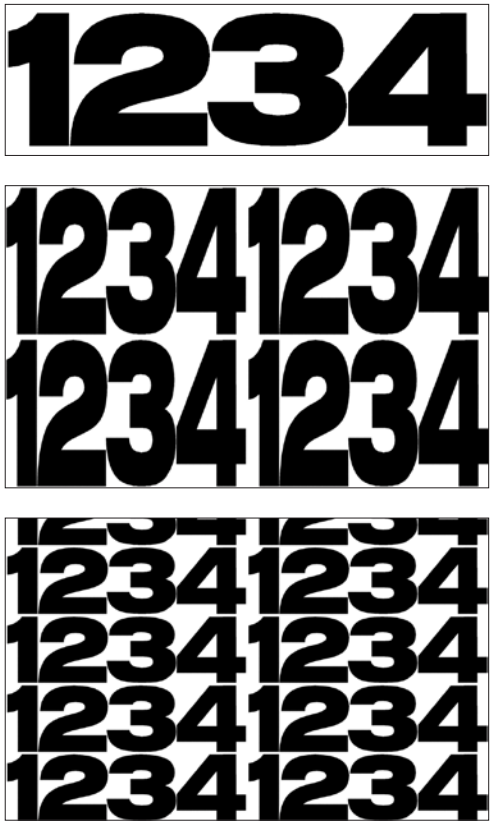


Figura 4.61 Mapa original (arriba), ajustado al objeto (centro) y ajustado al original (abajo).

Si contamos con texturas de dimensiones conocidas es posible utilizar una opción que también está incorporada en muchos programas: *Real world map size*. Esta opción está disponible tanto en el modificador UVW como en el editor de materiales y permite especificar el tamaño del mapa de bits en unidades de la escena. En este caso no tenemos necesidad de manipular las dimensiones del *gizmo* pues este se ajustará automáticamente a estas dimensiones.

Proyecciones de un mismo mapa sobre diferentes planos

Si queremos proyectar una misma textura sobre un objeto con diferentes planos sin que su aspecto se altere, tendremos que recurrir, en principio, a proyecciones múltiples, que se tratan en el apartado siguiente. Esto implica tener que recurrir a técnicas más complicadas que, en algunos casos, es posible evitar. Si los planos sobre los que vamos a proyectar la textura son ortogonales, podemos utilizar dos recursos sencillos para que la textura se proyecte exactamente igual sobre los dos planos.

Una solución es editar el operador UVW girándolo a 45º. Si, previamente, hemos adaptado su tamaño a una de las caras bastará con multiplicar su anchura por la raíz de 2 para que, al proyectarse, quede como estaba. Y si la precisión no es importante podemos obviar este cálculo. En cualquier caso, la proyección quedará igual por ambas caras, que es lo que nos interesa.

Otra solución es editar el operador UVW, cambiar su tipo a *box*, ajustar una de las dos caras y hacer que la anchura de la otra cara sea la misma. De este modo la proporción se mantendrá en ambas.

La figura 4.62 ilustra estas dos posibilidades para un caso tal como la proyección de un balcón, en donde nos encontraríamos con este problema.

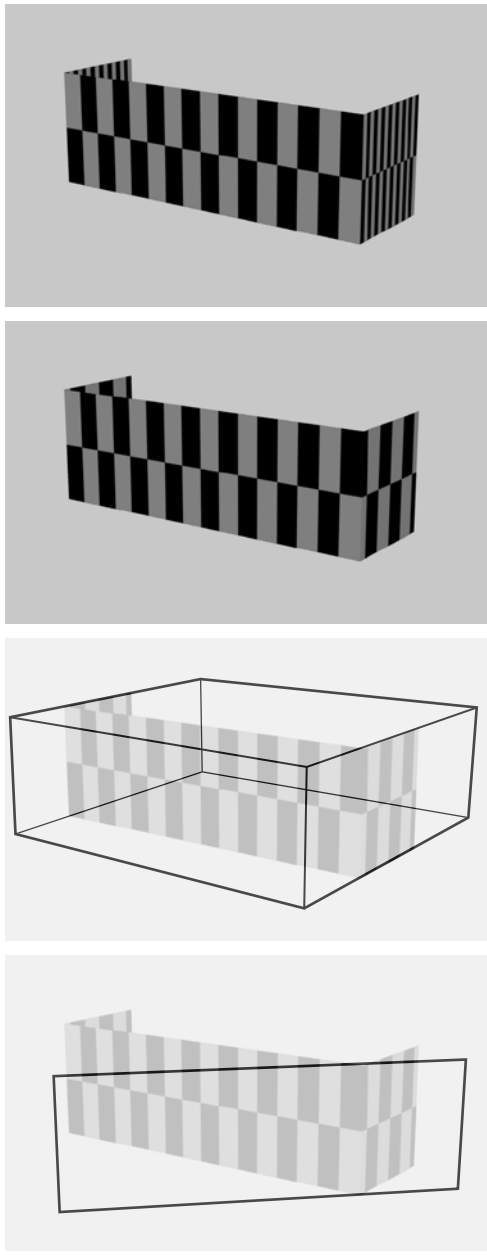


Figura 4.62 Proyecciones de un mismo mapa sobre diferentes planos: a) *Proyección deformada sobre dos planos;* b) *Proyección corregida mediante alguno de los dos tipos que siguen;* c) *Tipo box de lados iguales;* d) *Tipo plano girado 45º.*

Pero ¿qué pasa si los planos no son ortogonales? ¿hay alguna alternativa antes de recurrir al procedimiento más controlable, pero más complicado, de las proyecciones múltiples?

Hay una solución que se encuentra en 3ds Max y quizás también en otros programas, que es utilizar un modificador especial denominado *map scaler*. Este modificador hace dos cosas importantes que permiten abordar con facilidad este tipo de problemas. En primer lugar, despliega las proyecciones adaptándolas a las caras poligonales del objeto. Y, en segundo lugar, mantiene la misma escala sobre todas las caras (de ahí el nombre). La figura 4.63 ilustra su aplicación a un caso similar, un prisma, que podía ser un balcón, pero con las esquinas achaflanadas. Todo lo que hay que hacer es aplicar este modificador que tiene tres parámetros principales: *scale*, *uv offset* y *wrap*. Con el primero, escala, se puede ajustar el tamaño del mapa; con el segundo, *offset*, se puede desplazar el mapa a lo largo de las caras y con el tercero, activado por defecto, se hace que las proyecciones envuelvan (*wrap*) las caras del objeto. Hay otras opciones de interés pero menos usadas para las que se puede consultar la ayuda del programa.

La figura 4.63 muestra dos casos de interés práctico en los que puede merecer la pena utilizar este recurso. El primero, ya descrito, es un prisma achaflanado en donde el uso de este mapa resuelve el problema de las proyecciones en las esquinas. El segundo es una serie de objetos de diferentes tamaños y proporciones pero a los que nos puede interesar aplicar una textura con la misma escala sin tener que hacer ajustes en cada caso.

Proyecciones múltiples

Si queremos que las proyecciones sobre las diferentes caras de un objeto sean distintas, sea porque vamos a utilizar diferentes texturas o porque vamos a utilizar la misma textura con diferentes orientaciones (y los recursos indicados en el apartado anterior son insuficientes), tenemos dos soluciones principales:

separar las caras de modo que se conviertan en objetos independientes o utilizar proyecciones múltiples. Aunque el procedimiento es más complejo en el segundo caso, resulta bastante más recomendable porque mantiene la unidad del objeto y evita tener que gestionar un número innecesario de objetos en la escena. También cabe la posibilidad de separar los objetos y luego volverlos a juntar. El resultado final será el mismo. Pero se entenderán mejor los procedimientos internos si analizamos el segundo caso.

En general, esto implica tres pasos principales: *a*) asignar identificadores de material a las caras poligonales del objeto, *b*) definir proyecciones adecuadas para cada una de estas caras y *c*) utilizar un material compuesto con varios canales, haciendo que la numeración de cada canal coincida con el identificador de material.

En 3ds Max, los pasos concretos que habría que dar, para asignar tres mapas diferentes a un mismo objeto como el de la figura 4.64, serían los siguientes:

1. Desde la escena, seleccionar el objeto, convertirlo a malla poligonal o asignarle un modificador que permita gestionar las caras (*Edit mesh*). Seleccionar la cara poligonal que recibirá el material 1. Desde la sección *Surface properties / Material* cambiar el número que aparece junto al identificador de material, ID por un 1 (escribir un "1" y confirmar con Intro). Hacer otro tanto con las otras dos caras asignándoles los identificadores 2 y 3. Comprobar el resultado con el botón *Select by ID* que resalta las caras cuyo ID se ha escrito en la caja de texto.

2. Definir las proyecciones para cada una de estas tres caras. Para ello, añadir 3 modificadores *UVW Map*. En cada caso, ajustar la proyección a cada cara y asignar a cada uno un número de canal, en el grupo *Channel*, junto a *Map channel*, que coincida con el número de material que se quiere aplicar. Tener en cuenta que los modificadores aparecerán en el catálogo en el orden inverso a como se han creado. El número de canal nos servirá también para recordar cuál es el primero y cuál el tercero.

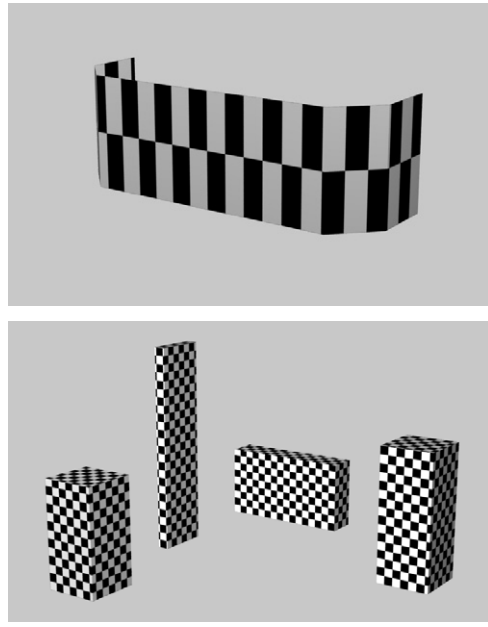


Figura 4.63 Proyecciones de un mismo mapa sobre diferentes planos con el modificador map scaler*: a) sobre un mismo objeto con planos no ortogonales,* b*) sobre diferentes objetos que compartirían una misma textura.*

Así tenemos definidas las proyecciones. El siguiente paso consiste en asignar un material especial compuesto por diferentes submateriales, un material que en 3ds Max se denomina multisubobjeto.

3. Desde el editor de materiales, crear un material *Multi/Sub-Object* de tres materiales. Definir cada uno de los tres materiales como un material corriente (*Standard* o *Arch&Design*) con su textura correspondiente. En la sección de coordenadas del mapa, en *Map channel*, asignar el canal 1 al mapa correspondiente al material 1, el canal 2 al del material 2 y el canal 3 al del material 3.

Por último, asignar este material al objeto. El resultado deberá ser que sobre cada cara se muestra un mapa diferente, tal como se muestra en la figura 4.64 que presenta un esquema simplificado con texturas procedurales de cuadrados, rombos y círculos.

Como decía al comienzo de este apartado también se pueden separar las caras del ob-

jeto y luego volver a juntarlas. Para ello, seleccionar el objeto, seleccionar las caras que interese separar y, en cada caso, presionar el botón *Dettach* del editor de la malla y darle un nombre adecuado. Así tendremos diferentes planos a lo que podemos asignar mapas con facilidad siguiendo los procedimientos que ya hemos visto.

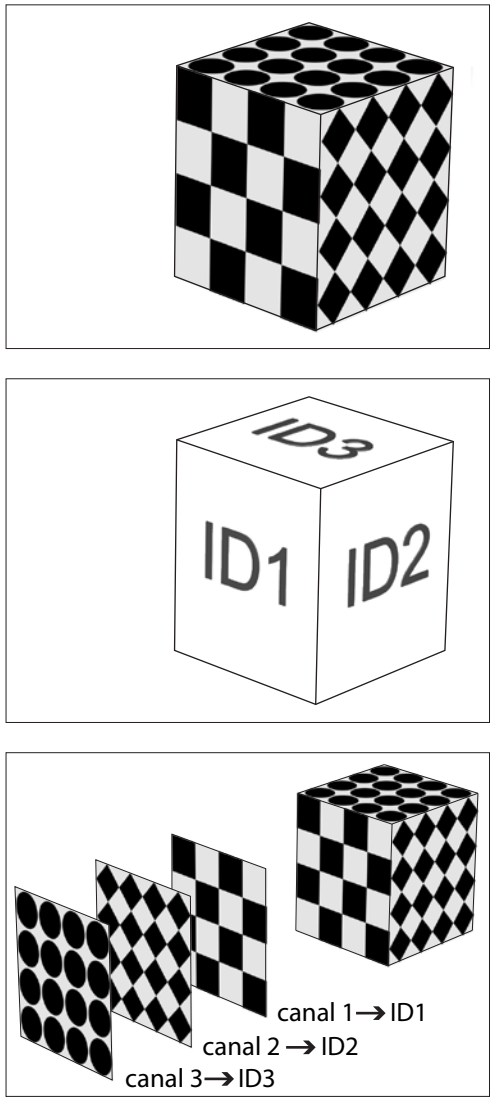


Figura 4.64 Proyecciones múltiples.

Luego seleccionar cualquiera de las caras, presionar el botón *Attach* y hacer un clic sobre las caras que hemos separado. Probablemente aparecerá un mensaje como el que sigue:

Match Material IDs to Material (predet)
Match Material to Material IDs
Do not modify Mat IDs or Material
"condense Material and IDs"

El significado de estas opciones es que el programa necesita hacer, automáticamente, lo que hemos hecho con el procedimiento anterior manualmente. La opción más corriente y recomendable es la primera, es decir, que liguemos los identificadores de materiales de las caras poligonales al material asignado. El significado de esta opción es que el número de identificadores de material del objeto que se está anexando se modificará para que no sea mayor que el número de submateriales con que cuenta el objeto al que se va a juntar. Un ejemplo característico es el de una *box*, que cuenta en principio con 6 ID. Si esta *box* se anexa a un objeto que tiene solo 3 submateriales, su número de ID se reducirá para adaptarse al del objeto de destino.

La opción "condense material and IDs" está relacionada con la anterior y hace que los submateriales duplicados o no utilizados se eliminen del material multi al que van a ir a parar.

Proyecciones sobre superficies paramétricas con especificaciones internas UVW

Si la superficie es curva y su curvatura no es simple, como ocurre en el caso de un cilindro o una esfera que pueden considerarse como plegamientos regulares de un plano, no será posible utilizar ninguno de los métodos anteriores.

En el caso de una superficie que pertenece a la serie de primitivas con que cuenta el programa y que está generada a partir de parámetros y fórmulas internas de generación, las proyecciones pueden adaptarse a esta formulaciones internas y el programa asigna por defecto esta especificación al objeto. La

figura 4.65 muestra un toro generado mediante una invocación directa a la primitiva de ese nombre en 3ds Max. Si se aplica a este toro un mapa de cuadros, el mapa se adapta a la superficie de un modo que sería imposible conseguir con alguna de las proyecciones estándar (planos, cilindros o esferas) que hemos visto y sin necesidad de aplicar expresamente un mapa.

Otro tanto ocurre con las extrusiones simples. Al generar una extrusión por medio de una spline, como ocurre en la figura 4.66 que podría utilizarse para modelar una silla, ocurre, en principio, lo mismo. El programa genera definiciones UVW que se adaptan a la curvatura.

Y otro tanto ocurre con las extrusiones generalizadas o *lofts*. En este caso, al menos en 3ds Max, hay parámetros de edición que permiten ajustes interactivos. Al editar un *loft*, en la sección *Surface parameters*, hay un grupo, *Mapping*, con controles para, en primer lugar, aplicar una proyección propia, *Apply mapping*. Y si se activa esta opción quedan disponibles los parámetros siguientes: *a*) *Real world map size*. Tiene el significado habitual; *b*) *Length repeat*. Controla el número de veces que el mapa se repite a lo largo del *loft*, con la parte inferior del mapa situada en el primer vértice del *path*; *c*) *Width repeat*. Controla el número de veces que el mapa se repite a lo largo de la *shape*, con el lado izquierdo del mapa situado en el primer vértice de la *shape*; *d*) *Normalize*. Si se activa, se ignoran los vértices y la distribución del mapa se hace en función de la longitud del *path* y de la *shape*. En caso contrario, la posición de los vértices afecta a esta distribución.

Cuando se utiliza 3ds Max para modelar objetos, una alternativa a los *lofts*, que son más complejos, es utilizar *splines* con grosor (*renderable splines*) o el modificador *Sweep*, que permite asignar de modo sencillo secciones de todo tipo a una *spline*. Estas dos alternativas son mucho más sencillas de utilizar aunque no admiten modificaciones de sección a lo largo del recorrido, como los *lofts*. Pero tampoco incluyen estas opciones de control de las coordenadas de proyección UVW. Lo que puede ser una buena razón para utilizar *lofts* en lugar de *splines*, pese a su mayor complejidad de configuración.

Proyecciones sobre superficies de curvatura libre y definidas por NURBS

En el caso de una superficie curvada de forma libre, generada mediante modificaciones interactivas, como sería el caso de un plano

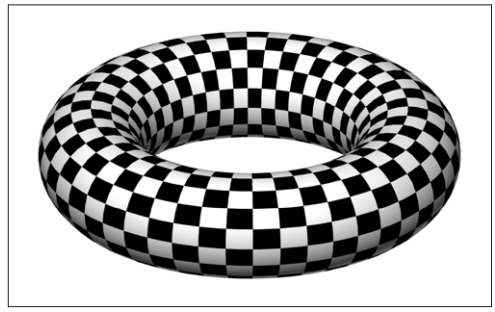


Figura 4.65 Proyecciones sobre un toro.

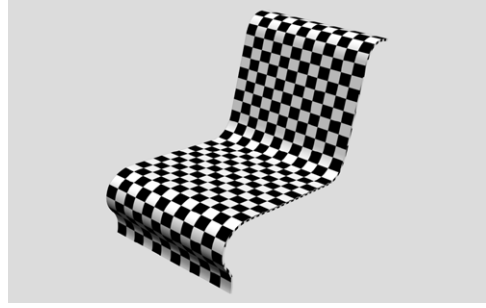


Figura 4.66 Proyecciones sobre una spline *extrusionada.*

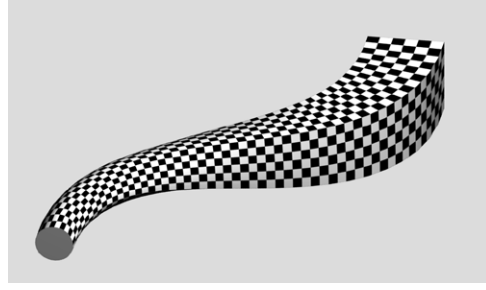


*Figura 4.67 Proyecciones sobre una extrusión generalizada (*loft*).*

convertido en malla y modificado con operadores especiales, no es posible adaptar la proyección a la superficie de un modo sistemático por los métodos complementarios que he resumido en el apartado anterior.

En este caso, la única solución es crear una superficie por medio de *NURBS* algo que es posible hacer en 3ds Max pero no tan bien como en programas especializados en *NURBS*, como Rhino. La figura 4.68 muestra una superficie generada por medio de una *NURB* (que se muestra en modo alámbrico en la parte superior con las curvas de generación internas) a la que se ha aplicado un mapa de cuadros. El mapa se adapta a los parámetros internos UV de un modo que sería imposible de conseguir por medio de una proyección plana.

El inconveniente de este sistema es que obliga a trabajar con *NURBS* en programas que no cuentan con ellas o no las manejan suficientemente bien. En el apartado siguiente se dan otras alternativas que dan resultados menos precisos que los que se conseguirían por medio de *NURBS* pero pueden ser más que suficientes en muchos casos en los que todo lo que se necesita es adaptar una textura a una superficie irregular de un modo que resulte convincente para una determinada distancia de observación. Y proporcionan métodos de control mucho más directos.

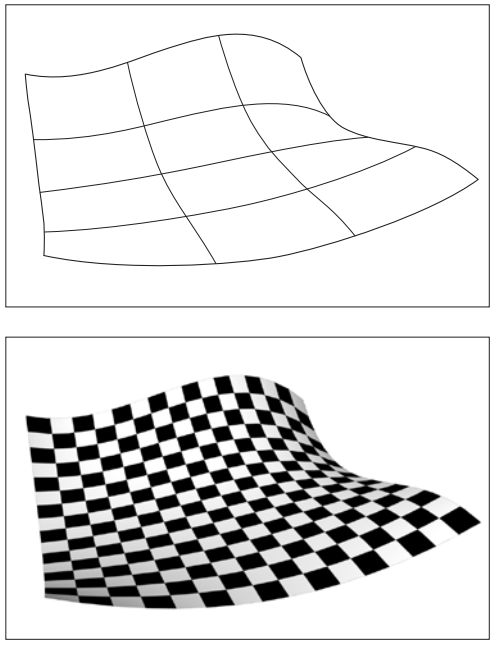


Figura 4.68 Proyecciones sobre NURBS.

Proyecciones desplegadas y métodos avanzados de edición de mapas

El método más utilizado, en el mundo de la creación profesional de escenarios virtuales, para adaptar texturas a objetos geométricos irregulares, es utilizar métodos manuales de modificación de las texturas. Sin embargo, estos métodos son bastante menos utilizados en el mundo de la arquitectura y algo menos en el mundo del diseño. Pero merece la pena saber que existen y que es el medio más eficaz de controlar las texturas irregulares. Los profesionales de la simulación visual, que trabajan para crear apariencias, no productos precisos, saben perfectamente que la búsqueda de la precisión geométrica es perfectamente inútil pues lo que cuenta es la precisión del efecto.

En 3ds Max, de modo similar a Maya o Blender u otros programas, estos métodos se aplican por medio de un modificador específico: *Unwrap UVW* (*Desenvolver* o *Desajustar UVW*). Es un modificador bastante complejo y el mejor modo de entender su funcionamiento será por medio de un par de ejemplos de complejidad progresiva. También pueden utilizarse programas independientes, como UVLayout, que permiten importar y exportar objetos y texturas desde programas corrientes.

En los últimos capítulos se darán más ejemplos del uso de *Unwrap*. El primer ejemplo de los que incluyo a continuación será muy simple para que se comprenda el funcionamiento básico. Habría que hacer lo siguiente:

1 Crear un plano de 4 x 4 segmentos. Convertirlo a malla poligonal.

2 Asignarle un material con un mapa de cuadros. En coordenadas de mapa, cambiar el valor de *tiling UV* a 4.
3 Añadir un modificador *Unwrap UVW*. En la sección de parámetros, presionar el botón *Edit*.

Con esto se abre el cuadro de diálogo del modificador. Es un cuadro de diálogo bastante complejo, con muchas herramientas y dividido en tres partes principales: una serie de menús y barras de herramientas en la parte superior, un gran panel de visualización en el centro, y herramientas adicionales en la parte inferior. Pero solo utilizaremos lo principal. En la figura 4.71 se muestra un esquema simplificado. El lector interesado en entender mejor el funcionamiento de este Editor puede ir a la ayuda de Max y, en el Índice, escribir "unwrap" y en la lista de entradas que aparecerán, seleccionar "Edit UVWs dialog".

Disponer el visor de este editor de tal modo que se vea también el visor de la escena, para comprobar los resultados. La figura 4.69 muestra, a la derecha, el plano con el mapa de cuadros asignado y, a la izquierda, lo que aparecerá en el visor del *Edit UV* (ocultar el fondo presionando el icono *Show map*, en la barra de herramientas superior, a la derecha).

Como se puede apreciar en esta figura, el Editor UV muestra unos puntos blancos que se corresponden con los vértices del plano. Cuando coloquemos la textura como fondo de este editor, los puntos se relacionan directamente con los colores que tienen debajo por lo que, al desplazar los puntos, estamos cambiando esta relación.

4 Comprobar que en la parte inferior del editor *Unwrap*, bajo *Selection modes*, está activado *Vertex*. Luego activar la herramienta *Move* (barra de herramientas superior, a la izquierda) y seleccionar algunos vértices y moverlos. La textura de cuadros del plano se deformará para seguir el movimiento de los vértices. Fijarse en que el efecto es aparentemente el inverso al esperado inicialmente pero que basta comprobar que parte de textura se está cubriendo con cada desplazamiento para asimilar la lógica de estas deformaciones.
5 Desplazando grupos de vértices se puede redistribuir libremente el mapa para obtener resultados como los del ejemplo de la figura 4.69.

El segundo ejemplo es más elaborado. Se trata de obtener un resultado similar al que obtendríamos con proyecciones múltiples editando una textura compleja. La figura 4.70 muestra, en la parte superior, el mapa que se utilizará y, en la parte inferior, el resultado final. Las imágenes del medio corresponden a fases intermedias del proceso.

1 Crear, en 3ds Max, un objeto con varias partes sobre las que queremos proyectar diferentes texturas, como en la figura. Aunque no es estrictamente necesario, para facilitar el trabajo que sigue, convertirlo a malla editable y asignar identificadores a las caras: asignar un ID 1 a las caras verticales, un ID 2 a las horizontales, un ID 3 a uno de los extremos y un ID 4 a otro de los extremos. Esto facilitará la selección posterior.
2 Crear, en Photoshop, un mapa con texturas que se correspondan con estas diferentes partes, como en la figura citada.
3 De vuelta en 3ds Max, asignar al objeto un material *Standard* o *Arch&Design* que tenga como mapa, en el nivel de *Diffuse*, el que acabamos de crear.
4 Asignar al objeto un modificador *Unwrap UVW*. En la sección *Parameters*, presionar el botón *Edit*, como antes, para que se abra el *UV Editor*. Desactivar el botón *Show map* para ver con más claridad las caras.

El visor mostrará todas las caras del objeto pero superpuestas. Como nos interesa ajustarlas individualmente para poder colocarlas manualmente sobre el mapa, lo primero que haremos será desplegarlas y, lo siguiente, moverlas y manipularlas para que

coincidan con las partes del mapa que nos interesa.

5 En la barra inferior, bajo *Selection modes*, presionar el botón *Face*. Luego ir al menú *Mapping / Flatten mapping*. En el cuadro de diálogo que se abrirá, dejar las opciones por defecto pero activar la casilla *By Marial IDs*.

Ahora el visor mostrará todas las caras desplegadas, agrupadas según los 4 IDs que hemos creado. Si se selecciona una cara en el visor, la cara correspondiente aparecerá resaltada en la escena, tal como se puede apreciar en la figura 4.70 en donde una de las caras se ha dejado seleccionada. El editor incluye filtros (en la parte inferior del visor principal) para mostrar solo las caras seleccionadas o los grupos de ID seleccionados. Utilizaremos este último recurso para manipular con más facilidad las caras.

6 En la barra de herramientas superior, desplegar la lista que permite escoger un mapa de fondo y seleccionar el mapa que está asignado al objeto. Si no apareciese (o si se quisiera escoger otro), presionar el botón *Pick map* y seguir las instrucciones. Luego presionar el botón *Show map* para que el mapa aparezca en el fondo.

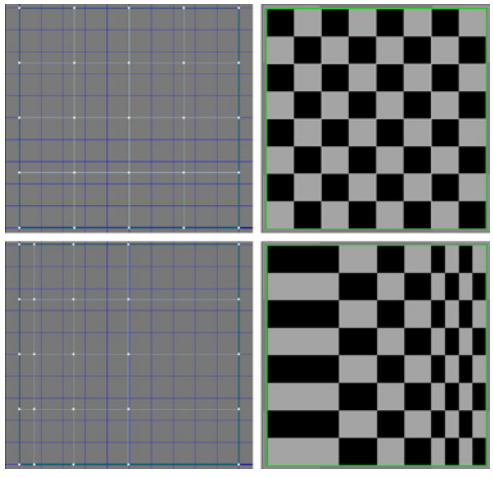


Figura 4.69 Proyecciones desplegadas. Deformación de la proyección de un mapa plano.

7 Utilizando las herramientas de mover, rotar y escalar en modo de selección *Face* o la herramienta de mover en modo de selección *Vertex*, mover las caras y los vértices en el editor hasta que coincidan con las partes del mapa que nos interesa para conseguir un resultado similar al de la figura.

Este ejercicio debería realizarse antes de utilizar estos métodos de despliegue de mapas para crear texturas complejas como los que veremos más adelante.

Como ya he dicho, el editor de *Unwrap* es bastante complejo. Está pensado para aplicaciones profesionales de creación de personajes con piel, pelo, ropa, etc., integrados en una única textura que se aplica a un objeto de forma libre. Para el tipo de aplicaciones a las que va dirigido este libro, la mayoría de estas herramientas resulta excesiva.

Sin embargo, hay algunas herramientas que son muy útiles para estas aplicaciones. Mi recomendación, por tanto, es: *a*) hacer una pequeña incursión en la ayuda del programa o en tutoriales sencillos que pueden encontrarse en internet; *b*) probar las herramientas generales aplicándolas a un objeto con diferentes caras y diferentes orientaciones como en los ejemplos que hemos visto; *c*) fijarse particularmente en las que destaco a continuación como principales.

La figura 4.71 muestra un esquema de la interfaz del editor. La describo brevemente subrayando las herramientas que más nos importan para las aplicaciones propias de este libro. Los números están referidos a los de la figura citada:

1 Menús. Incluye varios comandos que en muchos casos son alternativos al uso de herramientas. El que más nos interesará es *Mapping*/ *Flat mapping* que, como ya hemos visto, sirve para desplegar las caras del objeto y asociarlas a la textura de

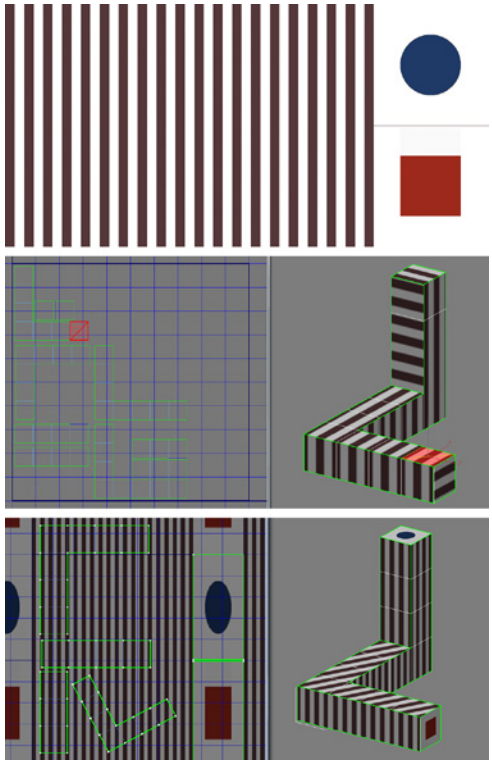


Figura 4.70 Proyecciones desplegadas. Asignación de una textura compleja a las diferentes caras de un mismo objeto.

fondo. Puede aplicarse a todas las caras o a una selección.

2 Herramientas de transformación (*move*, *rotate*, *scale*, *freeform*, *mirror*). La más útil es la cuarta, *freeform*, marcada con un aspa en la figura, que combina mover, rotar y escalar. Conviene fijarse en que la herramienta de mover incluye tres herramientas: mover en ambos sentidos o mover solo en vertical o en horizontal. Lo mismo ocurre con la herramienta de simetría (*mirror*).
También puede restringirse el movimiento o el cambio de escala a una dirección horizontal o vertical si se presiona la tecla *Shift* al mismo tiempo que se aplica.

3 Los tres primeros botones sirven para: *a*) Activar o desactivar la visualización del fondo; *b*) Cambiar el espacio predeterminado (UV) por VW o UW; *c*) Acceder al panel de opciones. La cuarta es una lista colgante que permite elegir la textura que se mostrará en el fondo.

4 Panel principal de visualización.

5 *Quick transform*. Incluye herramientas muy útiles para alinear horizontal o verticalmente los elementos seleccionados, para distribuir de modo regular los elementos seleccionados en horizontal o vertical y para rotar la selección 90º o -90º.

6 *Reshape*. Incluye una herramienta particularmente útil para enderezar el elemento seleccionado (rota las aristas del polígono hasta que quedan horizontales o verticales, lo que quede más cerca).

7 *Stitch*. Es un método de edición avanzado que no usaremos (sirve para "coser" o "recoser" las uniones de vértices o aristas a otros subobjetos).

8 *Explode*. También es un método de edición avanzado que, en principio, no usaremos aunque puede ser útil en algún caso pues permite separar los elementos seleccionados de otros a los que han quedado unidos. También incluye herramientas para hacer lo contrario, fusionar elementos (*weld*).

9 *Peel*. Otro método de edición avanzado que permite reestructurar las coordenadas de textura por métodos intuitivos muy ágiles que se usan extensamente en edición profesional.

10 *Arrange*. Incluye herramientas para reestructurar los elementos, empaquetándolos (*packing*) de diferentes modos semiautomáticos.

11 *Properties*. Se utiliza principalmente con las herramientas anteriores para hacer que determinados elementos se mantengan juntos durante las operaciones de reestructuración.

12 *SubObject selection*. Incluye herramientas de selección de subobjetos. Los tres primeros botones seleccionan los modos vértice, arista y polígono y los siguientes incluyen herramientas para ampliar la

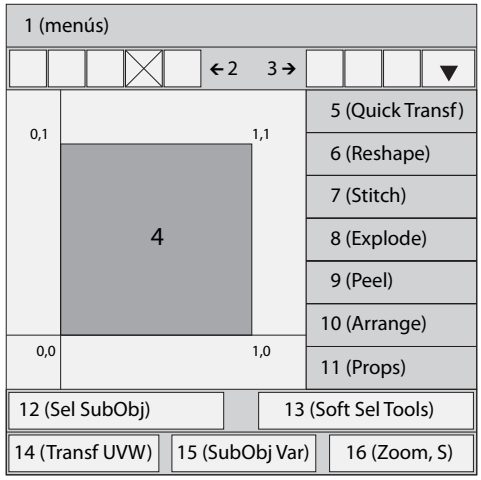


Figura 4.71 Editor UV. Esquema de la interfaz.

selección automáticamente de diversos modos.

13 *Soft selection tools*. Incluye herramientas para que las transformaciones se apliquen de modo ponderado a vértices cercanos según los parámetros especificados en este grupo.

14 *Transform UVW*. En esta sección se muestran las coordenadas UVW de vértices y aristas, que se pueden modificar numéricamente de modo absoluto o relativo.

15 *SubObject* (varios). En este grupo central se incluyen varias herramientas para bloquear la selección, para ocultar o desocultar elementos, para aislarlos o para congelarlos. La lista desplegable *All IDs* permite seleccionar ID de materiales para que solo se muestren las caras con estos identificadores. También se pueden ocultar, desocultar, congelar o descongelar elementos desde el menú *Display*.

16 *Zoom*, *Pan*, *Snaps*. Este último grupo incluye herramientas de navegación típicas (desplazamiento, zooms, zooms a extensión, etc.) y *snaps* para mover elementos a posiciones fijas.

4.5 Tipos de mapas

Todo lo anterior era independiente del contenido, pues la especificación de las proyecciones, como ya hemos visto, es una propiedad del objeto que define el modo en que se va a proyectar un mapa sobre un objeto, sea cual sea su contenido y sea cual sea su nivel de aplicación. En los apartados y capítulos siguientes veremos que los contenidos pueden ser muy diversos y que implican varias técnicas adicionales. Y en el capítulo siguiente veremos que los tipos o niveles de aplicación, también lo son. Pero antes será preciso analizar mínimamente los tipos de mapas que podemos utilizar.

Mapas de bits

Para comenzar, partiremos de la base de que el contenido será el más corriente, un mapa de bits. Y que el modo de aplicación será tambien el más corriente: la substitución de los parámetros básicos de definición de color del material por el color correspondiente a cada punto del mapa de bits que se corresponda con el punto geométrico del objeto, según el tipo de proyección que se haya definido. Los ejemplos que se han dado en los apartados anteriores presuponían también este tipo de contenido y proyección. Aunque, como veremos más adelante, esto no tiene porqué ser siempre así.

Comencemos por recordar que un mapa de bits es una matriz de puntos ordenados, cada uno de los cuales tiene un valor que indica su color. Y que puede obtenerse digitalizando una imagen real o que puede generarse internamente, creando una imagen sintética con un programa de pintura digital o haciendo un *render* o combinando ambas cosas. Hay dos parámetros fundamentales que deben tenerse en cuenta en cualquier caso: el tamaño (número de píxeles) y el valor (bpp, bits por píxel).

La decisión sobre el rango de estos parámetros dependerá en última instancia del tipo de imagen que se pretenda conseguir y de la

salida prevista para esta imagen (impresión de mayor o menor calidad, filmación, etc.). Depende, por tanto, del caso concreto de que se trate por lo que lo abordaré con más extensión en el capítulo siguiente.

Resolución y multirresolución

La decisión principal es la relativa a la resolución adecuada de la textura que vayamos a utilizar. Esta decisión, que solo puede plantearse correctamente en función del caso concreto, como acabo de decir, tiene que ver con el tamaño de salida previsto para la imagen. Si el tamaño de salida en ningún caso va a ser superior a, por ejemplo, 1.200 píxeles de altura y tenemos que decidir el tamaño adecuado de la textura de un muro que en ningún caso va a ocupar más de la mitad de altura de la imagen, la textura utilizada podrá ser del orden de los 600 píxeles o algo más.

Pero dejando a un lado todo lo relacionado con la resolución adecuada para los diferentes casos, que también se discutirá más adelante, los mapas de bits plantean problemas técnicos adicionales.

En la vida real, la apariencia de las cosas cambia a medida que nos acercamos a ellas. Y una interesante demostración de hasta que punto esto es así, que también he mencionado, se encuentra en el documental *Potencias de 10*, creado por los Eames en 1968 y reeditado en numerosas ocasiones. Pero en la vida digital esto no ocurre así. Si nos acercamos a una textura virtual el resultado es que, a partir de cierta distancia, solo vemos píxeles cada vez más grandes.

Dicho de otro modo. Si un objeto con textura, pongamos que un muro, va a ser visto siempre a varios metros de distancia, no tenemos que preocuparnos demasiado por estos cambios y tampoco tenemos que preocuparnos demasiado por la calidad de la textura. Pero si va a ser visto a media distancia, necesitaremos el doble del valor previsto. Y si va ser visto en primer plano, es posible que nos haga falta una textura no solo del doble del tamaño previsto sino también de mayor calidad.

Una solución poco eficiente es utilizar la máxima calidad, tanto para las vistas lejanas como para las cercanas. Una solución más eficiente es trabajar con multirresolución.

El concepto de multirresolución se aplica tanto al modelado geométrico como a la representación de materiales. Pues lo dicho hasta aquí vale tanto para las imágenes como para los modelos geométricos: un muro que vaya a ser visto a gran distancia puede ser modelado de un modo muy tosco pues los detalles no se apreciarán. Pero si va a ser visto en primer plano tendremos que ocuparnos de todo tipo de detalles.

Desde el punto de vista geométrico, la multirresolución significa que tendremos dos o tres versiones de un mismo objeto, unas con muchas caras poligonales y otras con pocas. Esto se puede conseguir manualmente, elaborando dos o tres veces el mismo modelo, o mejor, por medio de herramientas especiales que reducen el número de caras de un modo automático. En este caso todo lo que hay que hacer es crear el modelo con el máximo detalle y luego aplicar una de estas herramientas a las diferentes versiones del elemento.

En 3ds Max esto se puede hacer por medio de tres modificadores: *Optimizer*, *MultiRes* o *ProOptimizer*. Los tres hacen lo mismo, reducir caras automáticamente, pero con algunas diferencias. Hasta la versión 2010 había dos modificadores de reducción de la densidad de una malla, *Optimize* y *MultiRes*, con características complementarias. *Optimize* está basado en el ángulo entre dos caras y, en función de este valor, reduce o no el número de polígonos. Su principal inconveniente es que no conserva las coordenadas de mapeado UV de las caras. *MultiRes* permite especificar el número de vértices del objeto resultante, en porcentaje y en números absolutos. Pero no tiene en cuenta el ángulo entre caras. Esto tiene el efecto de que puede crear un teselado adicional innecesario. Pero tiene la ventaja de que conserva el mapeado

UV. Por esta razón a veces se utilizaban los dos en sucesión.

Desde la versión 2010 está disponible una tercera alternativa, *ProOptimizer*, que parece reunir las ventajas de los dos anteriores. Es algo más complejo pero su primera y principal sección, *Optimization Level*, es prácticamente igual a la de *MultiRes*: una vez que se ha inicializado el modificador presionando el botón "Calculate", los parámetros disponibles permiten especificar la reducción en porcentaje y en valores absolutos del número de vértices. En la segunda sección, *Optimization options*, está disponible, entre otras, una opción para preservar las texturas. Debe tenerse en cuenta que los parámetros de esta sección no son interactivos: después de cada cambio hay que volver a presionar el botón "Calculate" de la primera sección. El grupo *Mode* da alternativas para preservar o no los bordes durante la optimización (un borde es una arista no compartida por dos caras). El grupo *Materials and UV* da opciones para preservar los mapas de distintos modos. Luego hay otros grupos de menor interés: *Normals* (da alternativas adicionales para gestionar las normales); *Merge tools* (si hay vértices o caras desconectados se pueden conectar a partir de un cierto umbral); *Sub-Object / Preserve Vertices* (si se mantiene activado los vértices seleccionados se mantienen sin optimizar; *Symmetry* (da opciones para preservar la simetría del objeto con respecto a un plano XY, YZ, XZ y con un determinado grado de tolerancia); *Favor compact faces* (preserva caras que formen un triángulo cercano a equilátero); *Prevent flipped normals* (controla que al eliminar un triángulo no se invierten las normales de otro). Resumiendo: si solo se va a utilizar la primera y principal sección da igual utilizar *MultiRes*, que es más sencillo.

En cualquier caso, la modificación de la resolución geométrica debe ir a la par de la modificación de la resolución del mapa asociado.

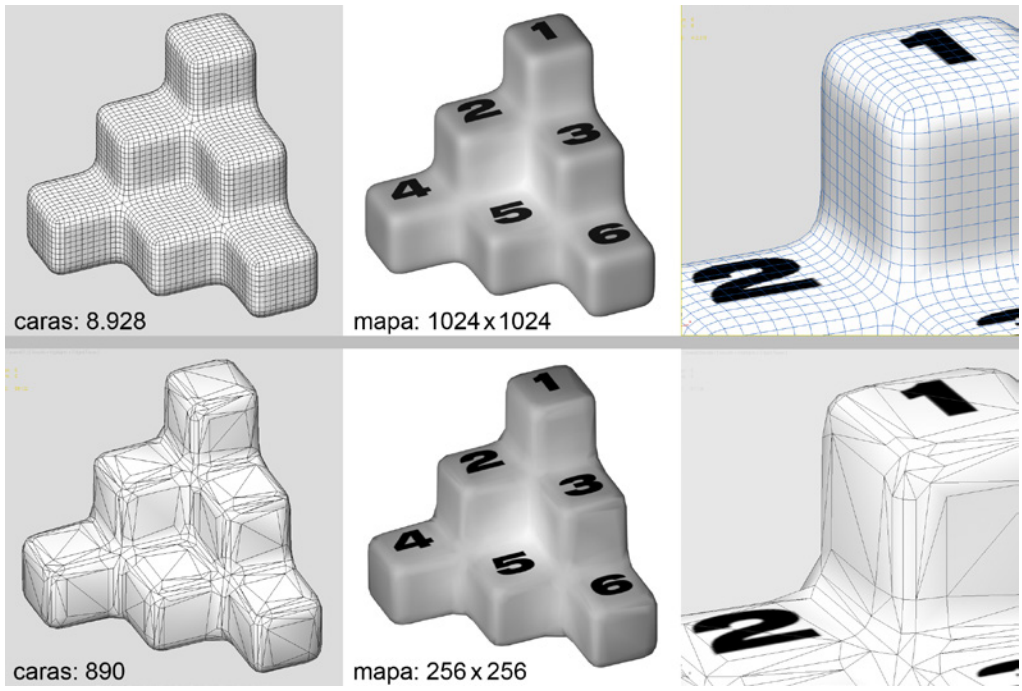


Figura 4.72 Un mismo objeto con diferentes resoluciones geométricas y de mapa.

En principio, esto quiere decir que se tendrán que utilizar mapas de diferentes resoluciones para asociarlos a los objetos de diferentes resoluciones.

Las imágenes de la figura 7.72 muestran el resultado de modificar un objeto con alta resolución geométrica (cerca de 9.000 caras poligonales) y un mapa de alta resolución (1.024 x 1.024) para reducir sus caras a un 10 % (cerca de 9.000) con un mapa de baja resolución (251 x 256). Las diferencias son claramente apreciables a corta distancia (véase la ampliación de la derecha) pero serían inapreciables a larga distancia. Las técnicas concretas para combinar estas diferentes versiones se detallan en el apartado siguiente.

Objetos de substitución. Recursos LOD

Si hemos obtenido dos objetos con diferente resolución a partir de alguno de los métodos anteriores, nos interesará substituir uno por otro según los casos. Esto puede hacerse de varios modos y utilizando diferentes recursos. Uno de estos recursos es utilizar objetos de substitución de modo que se trabaje con un objeto simplificado y, al final del proyecto o cuando sea necesario, substituir el objeto de baja resolución por la versión a mayor resolución. Otro recurso es utilizar dos o más objetos agrupados y hacer que, a partir de determinada distancia, se substituya uno por otro.

Para cualquiera de las dos opciones lo primero que se necesita es contar con al menos dos objetos con diferente resolución, geométrica, material o ambas. Lo normal será lo último, ambas, que es lo que se explica a continuación. Pero si solo interesa una de las dos, nos podemos saltar los pasos correspondientes.

1 Crear un objeto, “ObjAltaRes”, con una definición geométrica precisa, como el primer objeto del ejemplo anterior.
2 Hacer una copia en la misma posición, renombrarla “ObjBajaRes” y utilizar alguno de los recursos mencionados en el apartado anterior para reducir su definición geométrica.
3 Asignar al primero un material con un mapa de alta resolución (por ejemplo, de 1.024 x 1.024 px, como en el ejemplo anterior).
4 Asignar al segundo un material con un mapa de baja resolución (por ejemplo de 256 x 256, como en el ejemplo anterior).

Así tendremos dos objetos superpuestos, con diferente resolución (habrá que ocultar uno para ver el otro y viceversa).

Para utilizar el primer recurso, un *objeto de substitución* (un modificador *Substitute en* 3ds Max), hacer lo siguiente:

1 Seleccionar el objeto “ObjAltaRes”, ir al menú *File / Save selected* y grabarlo con el nombre “ObjAltaRes.max” (u otro que resulte más conveniente). Luego borrarlo de la escena.
2 Seleccionar el objeto “ObjBajaRes” y aplicarle un modificador *Substitute*. Desactivar la opción *Display / In viewport* y dejar activada la opción *Display /In render*.
3 Presionar el botón *SelectXRef object*. Escoger el archivo recien grabado “ObjAltares.max”.
4 Hacer un *render*. El resultado corresponderá al objeto de alta resolución que substituye al de baja resolución en la imagen de salida. De este modo, la escena no se sobrecarga pero la imagen final tiene la calidad que interesa.

Para utilizar el segundo recurso, un grupo ligado a una utilidad *LOD* (*Level of detail*) hay que crear, como en el caso anterior, dos o más objetos superpuestos, en la misma posición, uno de alta resolución y otro de baja resolución. Luego seleccionar los objetos y crear un grupo. Ir al panel *Utility*, presionar el botón *More* y escoger la utilidad *Level of detail*. Seleccionar el grupo recién creado y presionar el botón *Create New Set*. Los dos

miembros del grupo aparecerán en una lista, abajo, en el panel de *Level of Detail*, en orden de complejidad, el menos complejo arriba, con un número tal como 25,0 a la izquierda y el otro con un número tal como 100,0. Estos valores por defecto se refieren al tamaño relativo de la imagen (medido diagonalmente) en relación al tamaño total de salida. Consultar la ayuda para una explicación más completa de los parámetros. La idea básica es que al cambiar el tamaño de la imagen se mostrará uno u otro de los objetos que forman el grupo.

Otra alternativa, en versiones más recientes, es utilizar un recurso similar que se encuentra en el panel *Create* / *Helpers* / *VRML97* / *LOD*. En este caso se utiliza un objeto ayudante al que se añaden los objetos que sean. El resultado es similar. Para más detalles, consultar la ayuda del programa.

Mapas procedurales

La importancia de los mapas procedurales como técnica alternativa al uso de mapas de bits, ya la he comentado ampliamente. Baste con recordar que, con los mapas procedurales, los problemas se invierten; no hay ninguna dificultad por lo que respecta a la proyección, puesto que puede establecerse una relación biunívoca entre puntos del patrón y puntos del objeto. Por el contrario no es posible, por ahora, obtener una imagen de un objeto en 3D; la alternativa es utilizar funciones matemáticas que simulen la particular distribución de patrones cromáticos de un elemento natural. Los resultados son tanto más insatisfactorios cuanto más característica y familiar sea la textura a simular, y tanto más aceptables cuanto más indefinida. Los mapas procedurales producen resultados que, en general, hay que considerar como inaceptables, cuando intentan simular texturas de mármoles o maderas, y resultados que, con cierta habilidad por parte de quien los usa, pueden considerarse como aceptables, cuando se les utiliza para simular estucos, texturas rugosas o amorfas.

En este apartado describiré los principales tipos de mapas procedurales que están disponibles en programas corrientes y las técnicas concretas de ajuste de los valores de estos mapas. En el capítulo siguiente proporcionaré ejemplos concretos de aplicación de este tipo de mapas, incluyendo, pese a las limitaciones indicadas, la simulación de texturas naturales de granitos o mármoles por combinaciones relativamente complejas de este tipo de mapas.

Mapas procedurales 2D

Aunque los mapas procedurales característicos son 3D, por las razones dadas (no hay que preocuparse del ajuste de las proyecciones) también pueden utilizarse mapas procedurales 2D. En este caso la ventaja no tiene que ver con las dificultades de la proyección sino con la posibilidad de conseguir con facilidad determinados patrones o efectos y con la comodidad de no tener que depender de archivos externos para la aplicación de estos efectos.

No se debe olvidar que esta comodidad es una desventaja desde el punto de vista de la calidad de la imagen. Nada más sencillo que crear un patrón para un pavimento, un muro o un techo con procedural 2D de tipo *checker* o *tiles*. Pero, también, nada más artificioso: el resultado será geométricamente perfecto pero perceptualmente imperfecto pues ningún pavimento, muro o techo real presenta tal regularidad. Por eso conviene utilizar este recurso únicamente para elementos que se vayan a ver a distancia, o con poca luz (como muchos techos) o para representaciones estilizadas, con cierto grado de abstracción, en las que el realismo no interese.

Puede decirse que hay dos grupos principales de mapas procedurales 2D que interesan desde estos puntos de vista prácticos:

1 Mapas que generan patrones regulares. Hay dos variantes de este grupo que se encuentran en muchos programas. Los mapas tipo *Checker* generan un patrón ajedrezado con dos colores (o dos texturas). Los mapas tipo *Tiles* generan patrones basados en rectángulos de color, separados

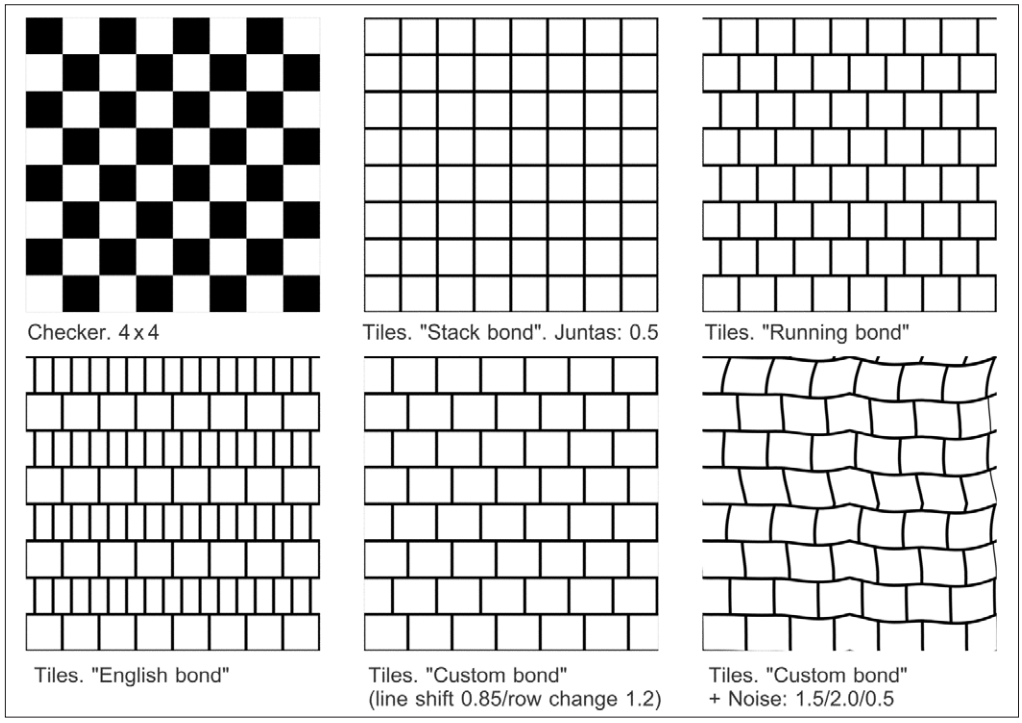


Figura 4.73 Patrones procedurales 2D. Seis variantes sobre un plano de 100 x 100.

por juntas, distribuidos como aparejos de paramentos. Entre los parámetros de este último hay varias opciones para escoger el tipo de aparejo, el color de la junta y el color (o textura) de la superficie principal. Estos dos tipos de mapas cuentan con controles adicionales para distorsionar el resultado de diversos modos.

2 Mapas que generan degradados. También se pueden encontrar dos tipos con varios subtipos que difieren únicamente en que el primero, *Gradient*, crea un degradado simple, con tres colores y un único punto de transición y dos subtipos (lineal y radial), y el segundo, *Gradient ramp*, crea un degradado más complejo, con tantos colores como se quiera, con puntos de transición especificados libremente y con más subtipos (lineal, radial, en caja, en diagonal, en espiral...). Es un tipo de mapa familiar para quien haya utilizado también programas de pintura digital, como Photoshop o Gimp.

Además de estos tipos básicos hay otros más que tienen un interés muy relativo o muy específico. Véaser el apartado sobre otros tipos de mapas procedurales, más adelante.

Mapas procedurales 3D

Los mapas procedurales principales son 3D pues su utilidad es más evidente en estos casos. Como ya he repetido, las ventajas técnicas que se derivarían de contar con mapas 3D adecuados son más que notables. Nos liberaríamos de la mayoría de las tareas engorrosas que implica utilizar mapas de bits: el ajuste, a veces imposible, de las proyecciones, la grabación y mantenimiento de bibliotecas de mapas, la coordinación entre dos espacios incompatibles, el espacio 2D del material y el espacio 3D del objeto, etc.

Sin embargo, como también hemos visto, la búsqueda de funciones que representen adecuadamente texturas 3D por medio de algoritmos matemáticos ha tenido un éxito relativo. Si se acepta una representación convencional y limitada, como es el caso en la cada vez más floreciente industria del cine de animación, que usa casi exclusivamente este tipo de texturas por razones técnicas evidentes, los avances espectaculares de los últimos años ponen a nuestra disposición una larga lista de mapas procedurales eficaces. Si se busca una representación de alta calidad, para imágenes estáticas, esto resultará insuficiente.

En esta sección resumiré los principales tipos de mapas procedurales 3D disponibles y que en general son variantes de un tipo de mapa principal, "Ruido" (*Noise*) que, como ya hemos visto en el capítulo 2, fue introduci-

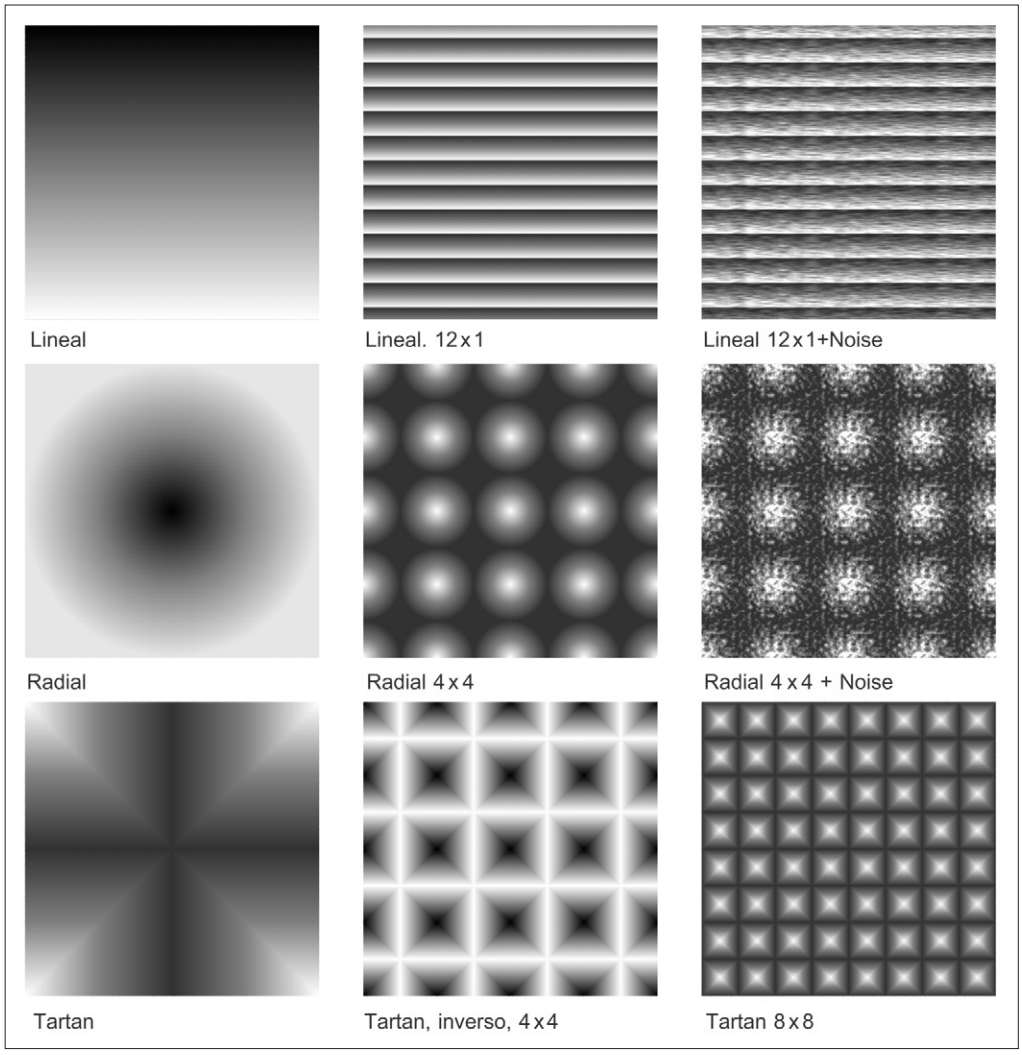


Figura 4.74 Degradados procedurales 2D. Seis variantes sobre un plano de 100 x 100.

do hacia 1985 por Perlin. El orden en que se presentan es alfabético con la excepción del primero, que es el principal dado que es la base teórica de los siguientes. Los parámetros que se indican son los de 3ds Max pero son prácticamente idénticos a los de otras aplicaciones aunque puedan variar los rangos numéricos, pues los algoritmos de base, que ya se han presentado en el capítulo 2, son los mismos.

También daré ejemplos algo más detallados de los dos principales, *Noise* y *Cellular*, para que se entienda mejor el sentido de los parámetros utilizados por estos mapas. Y, por añadidura, en el capítulo siguiente, en la última sección, en el apartado sobre texturas artificiales daré algunos ejemplos más elaborados de estos dos mapas procedurales.

Noise (Ruido). Crea patrones aleatorios de dos tonos y de formas similares. Los parámetros permiten cambiar el tamaño, los colores y el tipo de cálculo interno. El tamaño, *Size*, se da en unidades de la escena y hay que ajustarlo, por consiguiente, para cada caso. Los *Colors* predeterminados son blanco y negro y pueden ser uniformes o bien, si se les asigna un mapa (que puede ser también otro mapa de ruido procedural), variables. La función utilizada puede ajustarse con tres alternativas que aumentan el efecto: *Regular*, *Fractal* (utiliza un algoritmo fractal que da mayor contraste entre las formas de los dos tonos) y *Turbulencia* (utiliza un algoritmo fractal modificado que proporciona aún mayor contraste entre los dos tonos). Hay otros parámetros que convendrá utilizar. Si se producen defectos notorios en la representación (*aliasing*) puede reducirse reduciendo algo el rango dinámico, acercando los valores de *Low threshold* (0,0 por defecto) y *High threshold* (1,0 por defecto). Estos parámetros también tienen el efecto de aumentar el contraste. Por último, cuando se activan la modalidad fractal o turbulencia, puede ajustarse la función con los parámetros de *Level* (3,0 por defecto) y *Phase* (principalmente en animaciones). Véanse los ejemplos que se dan más adelante.

Celular. Otra variante de Ruido que también se ha introducido a nivel teórico en el capítulo 2 y que genera un patrón irregular a partir de dos colores. No interpola gradientes sino que se basa en valores esparcidos aleatoriamente. Los valores de otros puntos se evalúan en función de su distancia a los puntos iniciales. Puede ser útil para crear efectos de mosaico irregular. Véase la figura 4.76 y los ejemplos más elaborados que se dan en el siguiente capítulo.

Dent (Cavidad, Muescas). Es una variante de Ruido, con parámetros fractales que funcionan relativamente bien para mapas de relieve que busquen simular cavidades en una superficie, de ahí el nombre.

Smoke (Humo). Genera patrones fractales difusos que pueden servir para simular humo o para combinarlos con otros mapas como veremos más adelante pues a menudo se utiliza, superpuesto a otros mapas procedurales, para suavizar el resultado de los patrones previos. También se puede usar para simular nubes o manchas suaves sobre superficies.

Splat (Salpicadura). Genera patrones similares a los de una superficie manchada con pintura salpicada.

Speckle (Moteado). Genera patrones similares a superficies moteadas, como las del granito. También puede resultar útil como mapa de relieve asociado a otros mapas.

Stucco (Estuco). Otra variante de Ruido que genera patrones similares a los de una superficie estucada.

Otros mapas que también están basados en funciones de ruido y que, por ahora, tienen más interés teórico que práctico, pues apuntan a formas alternativas mucho más eficaces de generar texturas naturales, son los mapas que simulan madera o mármol. Sin embargo, los resultados, si se quieren obtener resultados realistas, son, por ahora, muy pobres. Los mapas de ondas pueden ser eficaces para crear efectos de ondulación simple. Los disponibles corrientemente son los siguientes:

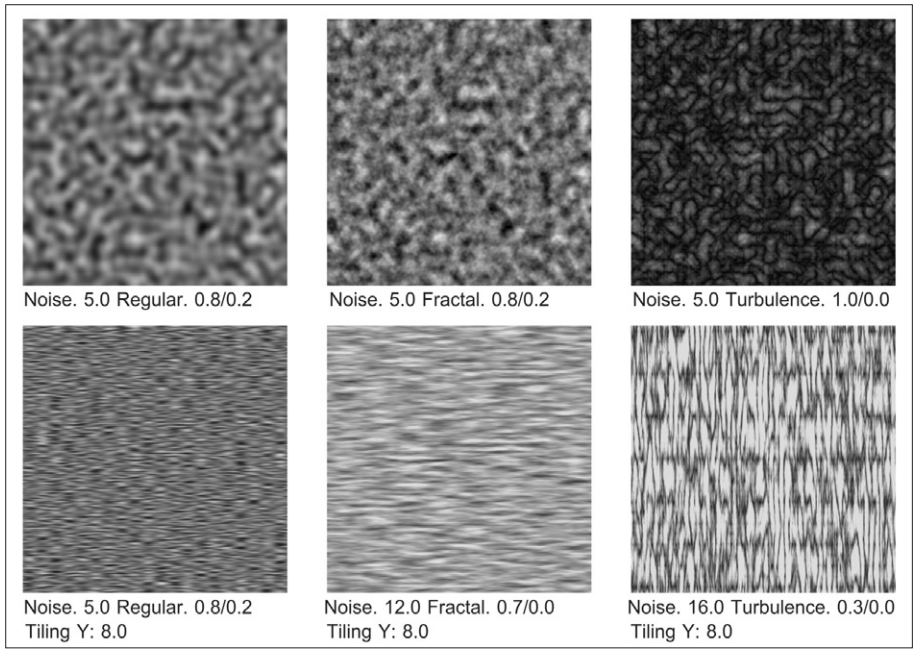


Figura 4.75 Noise. *Seis variantes sobre un plano de 100 x 100 (el primer valor es el del parámetro* size*).*

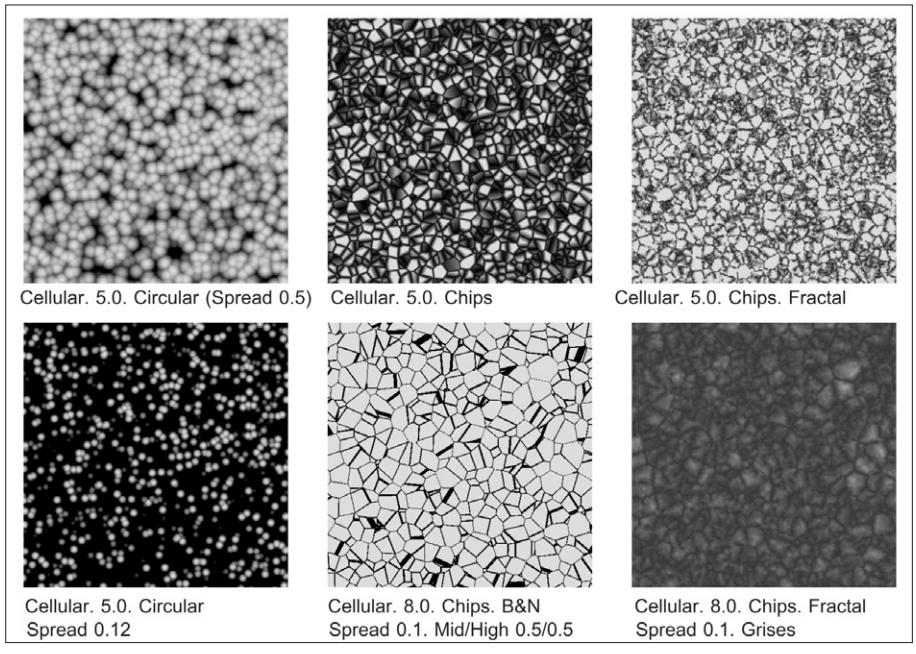


Figura 4.76 Cellular. *Seis variantes sobre un plano de 100 x 100 (el primer valor es el del parámetro* size*).*

Wood (Madera). Genera patrones similares a las de una superficie de madera por medio de funciones cilíndricas que simulan las vetas.

Marble (Mármol). Genera patrones similares a las vetas del mármol, con dos colores principales y uno intermedio, por medio de funciones parecidas a las del anterior.

Perlin marble (Mármol Perlin). Similar a mármol pero con vetas más elaboradas por medio de fractales.

Waves (Ondas). Genera patrones similares a las de la superficie del agua, con centros de ondas esféricas. Suele utilizarse conjuntamente como mapa difuso y de relieve, y como mapa de opacidad en combinación con otros. Si se quiere utilizar para simular agua es mejor utilizar los materiales y mapas más avanzados, que se describen en el capítulo sobre agua. Pero puede utilizarse para introducir variaciones aleatorias superpuestas a otros mapas, igual que humo o algún otro de los mapas mencionados antes.

Otros tipos de mapas procedurales

Hay muchos otros tipos de mapas que se utilizan para casos muy concretos. Menciono solo los principales.

Hay varios mapas utilizados para corregir o ajustar el color o la intensidad de salida. El principal es *Color correction* pero en casos muy concretos puede ser conveniente utilizar otros como *Output* o *Gamma and gain*.

Con *Color correction* se puede modificar el tono, saturación o luminosidad de una textura sin tener que volver a editarla en un programa de pintura digital. Volveré a este mapa en el capítulo siguiente. Es un mapa que substituye con ventaja a otros mapas anteriores que se conservan por razones de compatibilidad

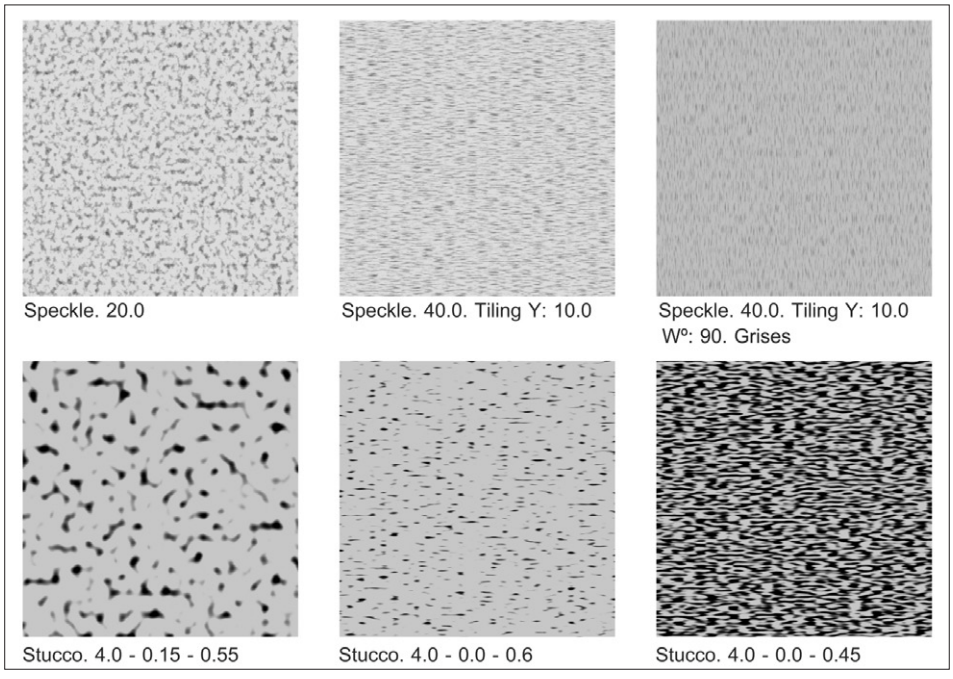


Figura 4.77 Speckle *y* Stucco. *Seis variantes sobre un plano de 100 x 100 (el primer valor es el de* size*).*

o porque son más sencillos de aplicar, como *Output*, *RGB multiply* o *RGB level*.

Con *Gamma and gain* se puede modificar tanto la intensidad como el valor *gamma* de un mapa de bits. En ocasiones puede ser útil para ajustar los valores de un mapa LDR (*Low Dynamic Range*) de modo que pueda usarse en un contexto HDR (*High Dynamic Range*). Véase el apartado sobre fondos del capítulo siguiente.

Hay un mapa muy útil en ciertos casos, el mapa *Falloff*, que modifica el color del mapa en función de la orientación de la superficie. También dedicaré un apartado especial a este mapa en el capítulo siguiente.

En esa misma sección del capítulo siguiente también veremos mapas que permiten combinar diferentes mapas entre sí, concretamente, el mapa *Composite*, la versión más reciente y más avanzada de este tipo de mapas, pero también otros mapas más antiguos pero que siguen siendo útiles y son más sencillos de utilizar, como *Mix* o *Mask*.

También veremos usos muy específicos de otro mapa que permite pintar con colores y que también es útil para crear cierto tipo de máscaras, como *Vertex color*.

En el contexto, muy específico, de sistemas de partículas, hay dos mapas que también veremos en el apartado correspondiente, *Particle age* y *Particle mblur*.

Y en el contexto, aún más específico, de medios participativos, también hemos visto el *shader Parti volume* y un *shader* asociado a este, *Transmat*, que no tiene otra función que encerrar la escena en un volumen virtual. No he mencionado, otro *shader* de mental ray, *Mist*, que sirve para crear niebla y es una alternativa más sencilla pero con menores posibilidades que el anterior.

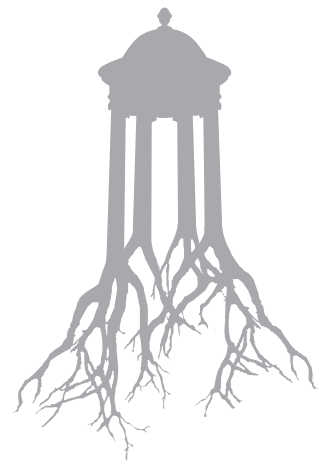
Los mapas que se utilizaban para simular de un modo más preciso la reflexión o la refracción, como *Reflect*/*Refract* o mapas similares que se utilizan en otros programas, han perdido sentido si se utilizan materiales arquitectónicos avanzados como *Arch&Design*. Pero para simulaciones de reflejos muy exigentes puede interesar utilizar mapas que simulan reflejos en varias capas, como el *shader Car paint* de mental ray.

Para simular efectos de resplandor y autoiluminación se pueden utilizar dos mapas, *Glow* y *Glare*, que también veremos en el apartado correspondiente del capítulo siguiente.

Para crear mapas de entornos específicos y para ajustar el modo en que estos entornos se procesan en reflejos hay toda una colección de *shaders*, desde el *shader mr Physical Sky* hasta los diferentes tipos de *Environment shaders* de mental ray que también veremos en el apartado sobre fondos del capítulo siguiente.

Y, en fin, hay mapas (y materiales) específicos para hacer que la representación simule un dibujo lineal, por medio de contornos y tratamientos planos de las formas, lo que se denomina genéricamente *Toon shading*, como el material *Ink'n Paint* y los diferentes tipos de *contour shaders* de mental ray. También dedicaré un apartado a este tema en el capítulo siguiente.

Hay otros tipos de mapas procedurales que no veremos pero al lector quizás le interese utilizar, por lo que le remito a la ayuda. Por ejemplo (en 3ds Max) *Combustión*, que se utiliza para pintar sobre un mapa de bits y para crear efectos especiales y requiere un *plug-in* específico, el Combustion de Discreet. O bien *Swirl*, que permite generar efectos de remolino con dos colores o dos mapas. O muchos otros que me he dejado por el camino por no considerarlos de suficiente interés o por desconocimiento, y que pueden encontrarse en la lista de mapas de los programas respectivos.



→5

Aplicaciones y combinaciones de parámetros y mapas

5.1 Recursos generales

En el capítulo anterior hemos visto los parámetros básicos que se utilizan para simular materiales, así como el uso de mapas para proyectar valores que substituyan o modifiquen parámetros básicos.

Los mapas se utilizan corrientemente para simular texturas o, dicho de un modo más preciso, para substituir un parámetro básico, el color, por patrones de colores variables asociados a un mapa. Pero también hemos visto que, tras la introducción de la idea general de *texture mapping* por Edwin Catmull en 1974, con la intención original de simular texturas, pronto se vio que esta técnica podía servir para muchas otras aplicaciones tales como la simulación de reflejos, transparencias o relieves. Y que, por añadidura, estas diferentes aplicaciones podían combinarse entre sí de múltiples modos.

En este capítulo veremos las técnicas y recursos que están ligadas a todas estas aplicaciones. La última sección estará dedicada a la aplicación más corriente, la aplicación de texturas.

Editores de materiales y mapas

La diversidad de estas aplicaciones requiere, en primer lugar, un instrumento adecuado que facilite al usuario poder manejar con facilidad todos estos parámetros y mapas y combinarlos libremente para definir un material digital. El propio concepto de *material* debe verse, en este contexto, como algo bastante distinto de un material real: como el contenedor de una serie de *shaders* relacionados entre sí.

El *editor de materiales* es, por consiguiente, un recurso fundamental en cualquier programa de simulación. La descripción pormenorizada del editor se puede encontrar en la ayuda de cada programa y no tendría sentido repetirla aquí. En lo que sigue me limitaré por consiguiente a algunos comentarios generales sobre las características comunes de los editores antes de analizar con más detalle los tipos de técnicas correspondientes a los diferentes tipos de aplicación y a las combinaciones de aplicaciones entre sí.

Los editores de materiales cuentan con varias secciones que en lo esencial se reducen a lo que sigue: *a*) ***Visores*** que permitan previsualizar mínimamente los resultados de aplicar los parámetros con que cuenta el editor; *b*) ***Propiedades*** especificadas en secciones correspondientes a los diferentes parámetros, que variarán en función del *shader* seleccionado y del mapa o mapas asociados al *shader*; *c*) ***Navegadores externos*** para acceder a bibiotecas de materiales y mapas predeterminados; *d*) ***Navegadores internos*** que muestren la estructura de un material compuesto por varios *shaders* y permitan acceder con facilidad a uno u otro nivel. A esto habría que añadir una variedad de recursos secundarios para asignar o reasignar materiales a objetos de la escena o para hacer que los visores muestren uno u otro aspecto de la estructura interna y unos cuantos más relativamente importantes y que varían bastante con los diferentes editores. Pero los principales son los citados.

La figura 5.1 muestra el aspecto general un editor que se corresponde con la descripción anterior y que puede relacionarse con facilidad con la de editores conocidos como los de 3ds Max, Maya, V-Ray, Maxwell, Blender, etc.

La representación digital de un material puede implicar una estructura bastante compleja, con diferentes niveles y subniveles (o “capas” y “subcapas”). La estructura más corriente es aquella en la que en el nivel superior están los parámetros básicos y en el nivel inferior, los mapas. Pero en cada uno de

estos niveles se pueden introducir otras definiciones de parámetros y otros mapas, sin que exista límite teórico a esta ramificación, aunque es relativamente raro encontrar materiales con más de 3 o 4 niveles de definición.

Esta estructura se puede manejar de dos modos principales. Mediante el uso de una interfaz que facilite la combinación de parámetros y mapas, es decir, mediante un editor de materiales que proporciona los medios para especificar y combinar valores y comprobar el resultado. O bien mediante el uso de "tipos de materiales" predeterminados que ya incorporan esta estructura, por lo que el usuario no tiene que preocuparse de construirla.

En los dos apartados que siguen ampliaré un poco estos dos aspectos. El uso del editor para crear nuestras propias estructuras implica la utilización de recursos adicionales entre los que ocupan un lugar especial los canales y las máscaras. Y el uso de tipos de materiales predeterminados implica un mínimo conocimiento de lo que podemos y no podemos esperar de estas estructuras predeterminadas.

Canales y máscaras

Un canal es, en general, un medio para enviar información desde un emisor a un receptor. Puede entenderse, en términos convencionales (analógicos, continuos), como un contenedor por el que se *transmite* información. Pero también, de un modo menos habitual, como un contenedor que *almacena* información para transmitirla más adelante.

En términos informáticos (digitales, discretos), un canal es también un medio para transmitir o almacenar información pero el

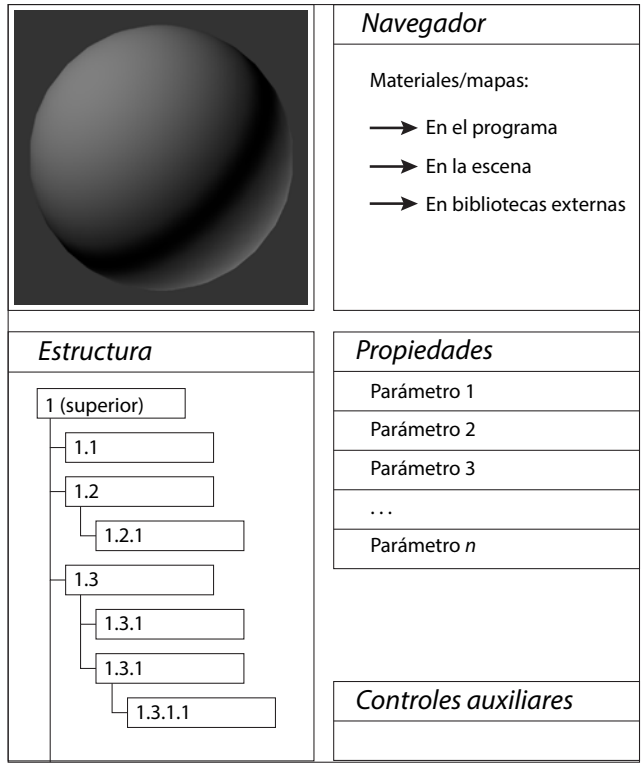


Figura 5.1 Editores de materiales. Estructura general.

"contenedor" es, en este caso, un grupo de bits, de un tamaño fijo y que ocupan una posición convenida dentro de una estructura más amplia.

También puede entenderse, de un modo más general, como un término convencional que se refiere a un determinado componente de una imagen que puede ser un valor pero también un índice que remite a un valor.

Así, en una imagen monocromática, el color de un píxel está asociado a un canal. Esto es, a un grupo de 8 bits, que permiten representar 256 valores (2^8) y que ocupan una posición convenida en la estructura global de la imagen codificada a la que pertenecen. Si se edita un archivo de imagen, una de las primeras cosas que encontraremos es una cabecera que precisa que cada grupo de 8 bits debe entenderse como la representación de un único valor. El programa que descodifique este archivo, por consiguiente, leerá todos los valores en grupos de 8.

En una imagen cromática, el color de un píxel está asociado a 3 canales, cada uno de los cuales es un subgrupo de 8 bits que permite representar el valor de cada color del mismo modo que en una imagen monocromática, ocupa, por tanto, 24 bits.

También podemos encontrar formatos de 4 canales, los tres anteriores más un cuarto que puede representar un canal alfa, un canal que representa la transparencia por medio de una escala de grises que asocia al color blanco máxima opacidad, al negro máxima transparencia y a los valores intermedios transparencias variables, y que ocupa, por tanto, 32 bits.

Pero también puede estar asociado a un único canal que codifica un único grupo que corresponde a una selección de 256 colores, como ocurre, por ejemplo, en el formato GIF, y que ocupa 8 bits. En este caso, el valor es un índice a una tabla (LUT, *Look Up Table*) que almacena una selección de colores.

Esto no se acaba aquí pues hay múltiples posibilidades que están asociadas al uso de canales adicionales y que iremos viendo más adelante, pues las diferentes aplicaciones de mapas pueden entenderse como la aplicación de mapas a través de canales que aplican el efecto a una propiedad diferente y que deben, por tanto, "circular" por rutas diferenciadas.

Los canales también se utilizan, en muchos casos, para crear máscaras de diversos tipos. Una ***máscara*** es un mapa que se superpone a otros mapas de tal modo que, en general, las partes blancas retienen los valores del mapa que está en primera posición (o última, según se mire), mientras que las partes negras hacen que los valores del mapa que está en primera posición se lean como transparentes, con lo que los valores representados serán los del mapa que está en segunda posición (o primera, según se mire). Pero también es posible componerlos de diversos modos por medio de combinacio-

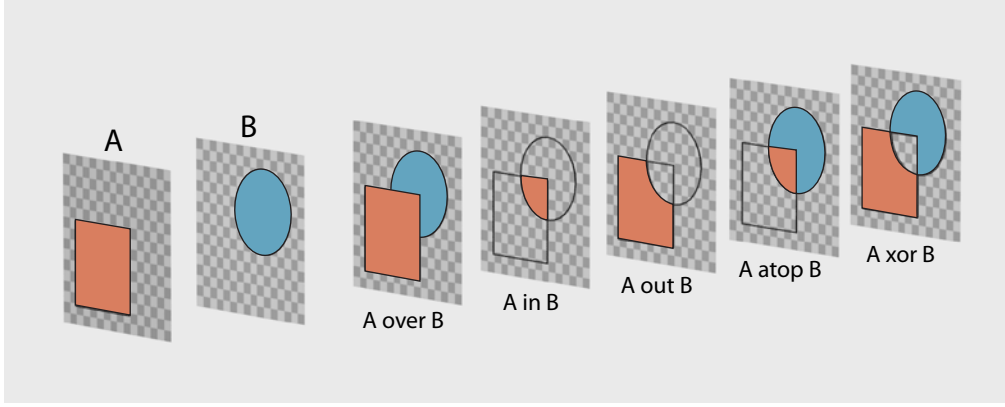


Figura 5.2 Canales. Operaciones básicas de composición de dos imágenes, A y B, con canales alfa.

nes de operadores, tal como se ilustra en la figura 5.2.

El concepto de *canal alfa* (*alpha channel*) fue introducido por Alvy Ray Simith a finales de la década de 1970. Alvy Ray fue uno de los fundadores, junto con Edwin Catmull, de Pixar, la compañía que nació como una división de Lucas Film hacia 1979 y financiada por Steve Jobs a partir de 1986 (y adquirida por Walt Disney Company en 2006). El uso de este tipo de recursos de composición, que veremos con detalle más adelante, se denomina genéricamente *alpha compositing*. En la sección sobre mapas de aplicación a recortes volveremos sobre estos temas que tienen gran importancia práctica.

Estructuras predeterminadas. "Tipos de materiales"

La programación de *shaders* y el uso de mapas que modifican parámetros básicos, da una enorme flexibilidad para simular todo tipo de efectos, combinando adecuadamente diferentes codificaciones o introduciendo estructuras condicionales en el propio *shader*.

Podríamos, por ejemplo, escribir un *shader* que leyera la información del objeto a que va a aplicarse y, en función de la información almacenada en los identificadores asociados a las caras poligonales, aplicase unos parámetros u otros. O bien un *shader* que, en función de la información sobre el ángulo de las normales correspondientes a las diferentes caras, aplicase uno u otro parámetro. O bien un *shader* que, en función de la orientación de esta normal, aplicase también uno u otro parámetro.

Ahora bien, si es previsible que estos casos se van a repetir con cierta frecuencia, merecerá la pena incorporar esta estructura a una plantilla genérica que reciba directamente los parámetros correspondientes de modo que quede disponible con facilidad. Esta estructura predeterminada es lo que se denomina en algunos programas un "tipo de material". Pero no hay que dejarse confundir por una terminología que puede resultar algo equívoca. Estos "tipos de materiales" no son sino *shaders* complejos que se han estructurado de modo que puedan responder de modo inmediato a diferentes casos. Los tres ejemplos que he puesto antes corresponden precisamente a tres de estos tipos en 3ds Max: el material "Multisubobjeto", el material "Top/Bottom" y el material "2-Sided". El mapa "Falloff", que veremos con detalle más adelante, también está basado en recursos similares, la orientación de las normales, aunque de un modo bastante más complejo.

Por ejemplo, si en el editor de materiales de 3ds Max se activa el botón "tipo", junto al nombre del material, se pueden escoger del orden de unos 14 tipos diferentes. Los más utilizados en las aplicaciones de arquitectura y diseño son los tres primeros. Doy un breve resumen de estos tipos para ilustrar las diferencias.

Standard. Es el tipo predeterminado que incluye los *shaders* y parámetros básicos que he resumido en la primera sección del capítulo anterior (Phong, Blinn, Metal, Strauss, etc.). Los parámetros están directamente relacionados con estos *shaders* históricos.

Arch&Design. Es el tipo predeterminado con mental ray, que también hemos visto y que es un ejemplo de material "monolítico", que incluye propiedades más avanzadas tales como reflejos que procesan, por medio de *ray tracing*, los rebotes provenientes de superficies cercanas.

MultiSubObjeto. Permite combinar varios materiales en un mismo objeto. Es el material que debe utilizarse con el sistema de multiproyección que hemos visto antes. Con el parámetro "Establecer número" se define el número de materiales que se pueden mostrar. Veremos ejemplos de aplicación de este tipo más adelante.

Además de estos tres, también se utilizan en algunos casos, pero menos frecuentes, los siguientes.

Mezcla (*Blend*). Permite combinar dos materiales en proporciones determinadas o por medio de una máscara de mezcla. También se describe más adelante aunque a menudo

resulta suficiente utilizar mapas de mezcla ligado a un material *Standard* o *Arch&Design* para conseguir el mismo efecto.

Compuesto (*Composite*). Se utiliza para combinar, por métodos de fusión, diferentes materiales. Como en el caso anterior también es posible conseguir el mismo efecto y de un modo más sencillo con el mapa del mismo nombre y a través de un material *Standard* o *Arch&Design*. Veremos ejemplos más adelante.

Superior/*Inferior* (*Top*/*Bottom*). Permite asignar distintos materiales a una superficie en función de la orientación de las normales. Véase el apartado siguiente.

Mate/*Sombra* (*Matte*/*Shadow*). Se utiliza para combinar objetos con imágenes de fondo. Un objeto con un material mate/sombra se convierte en un objeto "mate", más exactamente, en un agujero en la escena a través del cual se ve el fondo asignado como proyección de entorno. También veremos ejemplos de estos tipos en la sección sobre mapas de entorno.

Otros "materiales" que también veremos más adelante son los denominados genéricamente *Toon shaders*, que se utilizan para "representación no realista" o NPR por sus siglas en inglés (*Non Photorealistic Rendering*). De este tipo es el material *Ink'n Paint* que sirve para crear materiales propios de cómics, con colores planos y bordes resaltados. Otra alternativa es utilizar los mapas de tipo *Contours* de mental ray, o bien los equivalentes en otros programas. También veremos ejemplos de estos tipos en el apartado "Simulación no realista. Contornos con mapas".

Hay otros tipos de aplicación mucho más esporádica o que han quedado obsoletos. Véase la ayuda de los diferentes programas de aplicación. Por ejemplo, en 3ds Max el material *Ray trace* era un material muy útil para simular reflexiones y refracciones cuando no había otras alternativas mejores como *Arch&Design*. Tanto el material *Ray trace* como el mapa *Ray trace* fueron creados por Scott Kirvan y Steve Blackman cuando trabajaban para Blur Studio, una firma de California que producía diversos tipos de componentes para 3ds Max. Luego se independizaron para fundar Splutterfish, la empresa que comercializa el *plug-in* Brazil.

Otros tipos que se pueden encontrar son *Shellac*, un material de composición aditiva más bien sofisticado; *Morphing*, que se utiliza en animaciones para crear transiciones en el tiempo, de uno a otro material sobre un mismo objeto o sobre objetos con igual topología, y unos cuantos más, según el programa utilizado.

5.2 Mapas especiales de aplicación genérica

Hay diversos modos de combinar mapas. Uno de los más corrientes, y que ya hemos visto en el capítulo anterior, es asignarlos a diferentes partes (caras poligonales o agrupaciones de caras poligonales) de un mismo objeto. Esto requiere separar estas partes por medio de identificadores de material y utilizar materiales especiales, "multisubobjeto", que permitan canalizar las diferentes propiedades hacia los diferentes identificadores. Por añadidura, si se utilizan mapas, habrá que asignar también a las diferentes caras otros tantos controladores de la proyección.

Pero, como también he comentado, este método es equivalente a lo que resultaría de separar las caras, asignarles diferentes materiales y luego compactar el resultado en un objeto único que estaría compuesto internamente por diferentes objetos. Aplicar directamente un material "multisubobjeto" es un recurso que resulta más eficiente (cuando se conoce bien), pero no hay ninguna diferencia técnica específica por lo que respecta al procedimiento corriente de asignación de materiales a un objeto.

Hay otros métodos alternativos en los que los materiales se mezclan mediante técnicas que sí introducen modalidades diferentes. Podemos agruparlos en dos categorías: los que asignan diferentes materiales a partir de diferencias de orientación geométrica y los que asignan diferentes materiales mediante su-

perposición de propiedades que se controlan por medio de máscaras. Estos mapas pueden utilizarse para diferentes tipos de aplicaciones a la modificación de propiedades, razón por la que los he incluido en esta sección.

Mapas ligados a la orientación de las caras. El mapa *Falloff*

La primera categoría agruparía técnicas muy utilizadas como los materiales de *Doble cara*, *Superior/Inferior* o, principalmente, el mapa *Falloff*. Los dos primeros tienen la ventaja de que son muy sencillos de aplicar. La desventaja es que sus posibilidades son más limitadas.

El material *Doble cara* (*Double Sided*) permite asignar con facilidad dos materiales a una misma superficie a partir de la orientación de las normales. Es decir, hace lo mismo que si creáramos un identificador para cada uno de los lados y utilizáramos estos identificadores para asignar diferentes materiales. No es muy utilizado pues la mayoría de los objetos que se utilizan en simulación están formados por caras dobles pero puede ser útil en algún caso especial.

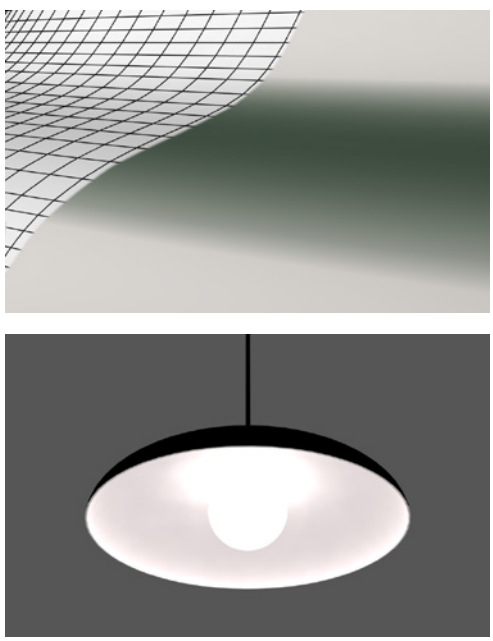


Figura 5.3 El material Top/Bottom *aplicado a la simulación de* a*) un terreno con diferentes pendientes y* b*) una luminaria con diferentes materiales en la parte superior y la inferior.*

El material *Superior*/*Inferior* (*Top*/*Bottom*) permite asignar con facilidad dos materiales a un mismo objeto a partir, también, de la orientación de las normales. Pero en este caso se basa en el ángulo de inclinación, lo que resulta bastante más útil para casos prácticos. Es un recurso que se ha utilizado extensamente para, por ejemplo, simular montañas con nieve o para asignar materiales a una luminaria. En el primer caso, todo lo que hay que hacer es ligar la transición a una determinada pendiente tal como se muestra de modo simplificado en la imagen superior de la figura 5.3 donde la parte izquierda muestra la malla geométrica, para que pueda apreciarse el cambio de pendiente, y la parte derecha la aplicación de este material, que incluye la posibilidad de fusionar con un degradado más o menos amplio la zona de transición. En el segundo caso, el recurso es el mismo y se saca partido de que determinados objetos, como ocurre con algunas lámparas, cambian de material en las zonas en que la superficie gira bruscamente, con lo que el cambio de material ligado a la orientación de la normal permite una aplicación muy sencilla. Esto se ilustra en la imagen inferior de la figura 5.3.

El mapa *Falloff* funciona de un modo similar pues permite asignar un color o un mapa a un objeto a partir de la orientación de la superficie del objeto. Pero incluye más posibilidades y, en este caso, la diferencia puede basarse en diferentes criterios, lo que hace este recurso particularmente útil para un gran número de aplicaciones (incluyendo las anteriores, que podrían replicarse exactamente igual con este mapa). La aplicación de este mapa hace que el color del objeto varíe (en principio entre blanco y negro) en función de la variación del ángulo de la normal a la superficie del objeto al que se aplica, pero relacionándolo con la orientación de la superficie de otro objeto, de la cámara, de la iluminación o de la distancia.

Hay, por tanto, varios tipos pero el más corriente es el *Perpendicular/Parallel* ligado a la dirección de la vista de cámara. Esto quiere decir que, con los colores predeterminados (*Front*: blanco, *Side*: negro) la zona en la que las normales son paralelas a la vista aparecerían blancas y en la que son perpendiculares, negras. Un uso también muy corriente es ligar este mapa a un canal de opacidad. De este modo, las zonas periféricas se hacen paulatinamente transparentes (el valor negro se registra como canal alfa, transparencia total y valores menores como transparencia parcial). Para que el efecto sea más convincente, a menudo habrá que modificar la curva de mezcla para que la distribución sea tal que se mantenga el color blanco (opacidad) en la zona central y se acelere con rapidez hacia el negro (transparencia) en la periferia. La figura 5.5 muestra el resultado con un material *Arch&Design* al que se ha aplicado un mapa *Falloff* predeterminado a la entrada *CutOut* (véase el apartado sobre recortes más adelante) pero ajustando la curva en la variante.

Con 3ds Max se pueden utilizar cinco tipos que son también los más corrientes en otros programas: *a*) *Perpendicular/Parallel* (*Front*: blanco, *Side*: negro; *b*) *Towards/Away* (*Towards*: blanco, *Away*: negro). En este caso, el cambio se basa en un ángulo que va de 0º (*towards*) a 180º (*away*), por lo que la transición es similar pero más lenta que con *Perpendicular*/*Parallel*; *c*) *Fresnel* (*Front*: blanco, *Side*: negro). En este caso, que se aplicaba a modificaciones de reflexión, el cambio se basa en el IOR (*Index Of Refraction*) que quedará disponible como un parámetro que se superpone al asignado al material en los parámetros principales. Era un recurso muy utilizado cuando no había materiales arquitectónicos, como *Arch&Design* que proporciona controles muy efectivos (curva BRDF) para este ajuste. Será preferible en la gran mayoría de los casos utilizar los controles de este material. Pero puede utilizarse para casos especiales como el ejemplo que se da más adelante, aplicado a un objeto con autoiluminación; *d*) *Shadow*/*Light* (*Shaded*: blanco, *Lit*: negro) se utiliza para asignar un color a la zona iluminada y otro a la que está en sombra lo que puede servir para realzar la iluminación. Cuando se utiliza este tipo se desactivan los parámetros de dirección que se resumen en el párrafo siguiente; *e*) *Distance Blend* (*Far*: blanco, *Near*: negro) hace que los colores cambien y se fundan con la distancia a la cámara. Al activar este tipo el panel presenta dos parámetros adicionales: *Near* y *Far*. Los valores que se asignan a estos dos parámetros hacen que el primer color se asigne hasta la distancia especificada en *Near*, el segundo a la distancia especificada en *Far* y entre ambos se cree un degradado de transición. Pero no funciona correctamente con mental ray, para probarlo, será mejor cambiar el motor de render a *default*.

La dirección corriente es la dirección de la vista de cámara (*Camera Z-Axis*). Hay otras opciones menos usuales para la cámara (*Camera X-Axis*, *Camera Y-X*). Y también puede ligarse al objeto. En este caso, los parámetros que hay que ajustar son las coordenadas locales del objeto (*Local X*, *Local Y*, *Local Z*) o bien las globales de la escena (*World X*, *World Y*, *World Z*).

Hay varias aplicaciones posibles. Véanse los ejemplos que se dan más adelante, en la sección sobre transparencias, para simular cortinas. Las figuras adjuntas muestran los tipos genéricos más corrientes. La 5.4 muestra

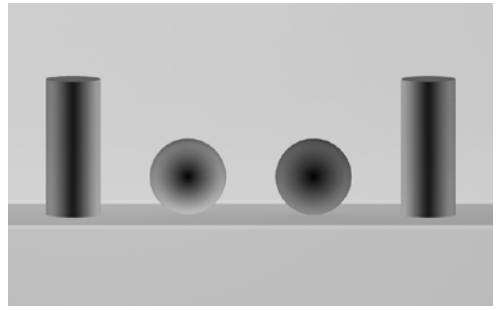


Figura 5.4 Mapa Falloff *aplicado a dos cilindros y dos esferas. Los dos cilindros y la esfera de la izquierda tienen asignado el tipo predeterminado:* Perpendicular/Parallel. *La esfera de la derecha, el tipo* Towards/Away.

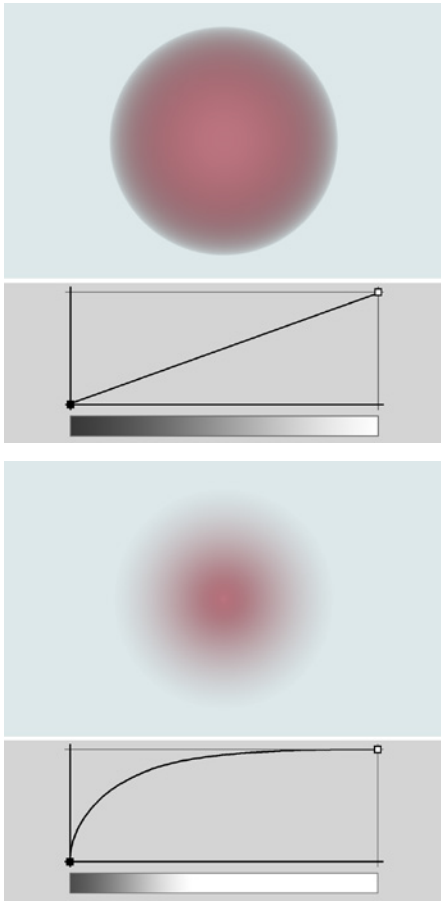


Figura 5.5 Mapa Falloff *aplicado a una esfera como mapa de recorte:* a) *valores predeterminados y curva de mezcla sin modificar,* b) *mismos valores y curva de mezcla modificada.*

las diferencias entre usar el modo predeterminado, *Perpendicular*/*Parallel* (los cambios se dan en un arco de 90º, entre la cara frontal y la lateral) o *Towards*/*Away* (los cambios se dan en un arco de 180º, entre la cara frontal y la posterior) que se traducen en un cambio más suave.

Pero para controlar mejor esta transición será mejor editar la curva, tal como se ilustra en la figura 5.5, ya mencionada, que muestra uno de los usos más característicos de este mapa: utilizarlo para crear una figura de recorte de tal modo que los contornos se desvanezcan. Para conseguir este efecto es necesario que la transición entre las zonas opacas (blancas) y las transparentes (negras) se de con mayor rapidez. Para ello, basta editar la curva de recorte, convertir los puntos extremos en puntos de una curva Bezier (situándose encima y activando el menú contextual) y mover las tangentes para obtener una curvatura como la de la figura. La parte central retendrá el color (rojizo en este ejemplo) o la textura del material al que se aplica el mapa si fuera el caso.

La tercera figura de este grupo, la 5.6, ilustra otro uso relativamente frecuente. Si se aplica este mapa para controlar la autoiluminación (junto al parámetro *Filter* si se utiliza un material tipo *Arch&Design*, en la sección *Self Illumination*) se pueden conseguir diferentes efectos, por ejemplo, que se cree un borde luminoso en los contornos, lo que, entre otras cosas, puede servir para destacar la figura con respecto al fondo. En este caso se ha utilizado el tipo Fresnel. La rapidez del cambio se puede controlar en este caso con el índice de refracción. El pie de la figura citada indica los valores que se han utilizado para este caso.

Por último, la cuarta figura de este grupo, la 5.7, resume otro uso característico, similar al del material *Top*/*Bottom* (que como ya he dicho puede considerarse como una versión simplificada y más sencilla de este recurso). Si se escoge, para *direction*, la opción *World Z-Axis* (o *X-Axis*, *Y-Axis* en otros casos) podemos hacer que un material se aplique a de-

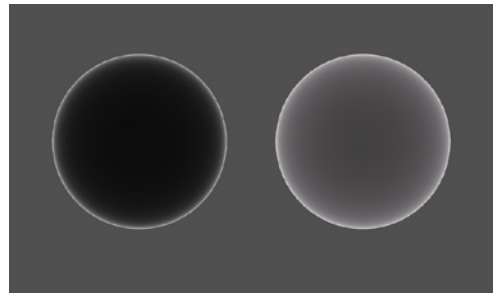


Figura 5.6 Mapa Falloff *aplicado a dos esferas a nivel de autoiluminación. Tipo Fresnel con índice de refracción:* a) *1,1,* b) *1,8.*

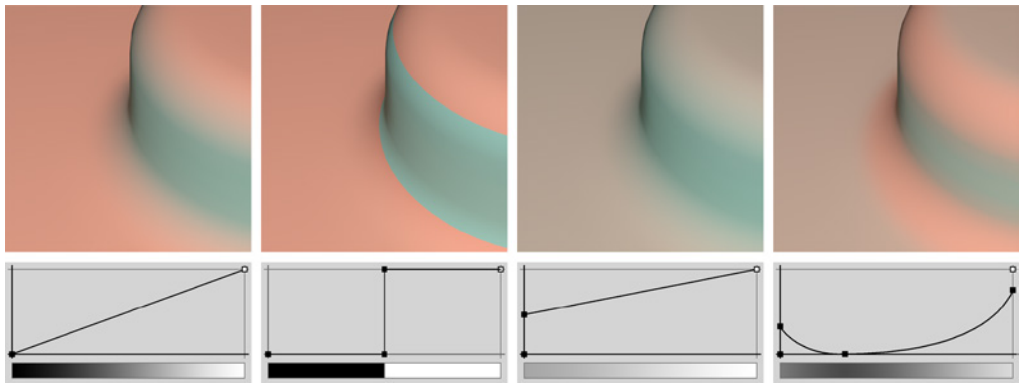


Figura 5.7 Mapa Falloff *aplicado en modo* Perpendicular/Parallel *a la dirección* Z/World *con cuatro curvas de transición diferentes.*

terminadas orientaciones y controlar también la transición entre estas orientaciones. Este recurso se puede utilizar para representar terrenos u objetos con cambios de material que coincidan con una cambio brusco de orientación.

En las figuras citadas se ha utilizado un material con un mapa de tipo *Mix* asignado al canal *Diffuse*. En el siguiente apartado se da más información sobre el uso de este mapa. De momento nos basta con indicar que, para este ejemplo, el mapa *Mix* se ha configurado del siguiente modo:

Color 1: rojo
Color 2: verde
Mix: mapa Falloff

Con esta configuración, el mapa *Falloff* actúa como una máscara de tal modo que, si se mantienen los colores predeterminados, negro y blanco, el color negro deja paso al color 1 y el color blanco retiene el color 2. El tipo que se ha escogido es *Perpendicular/Parallel* y la dirección *World Z-Axis*. Por tanto, las caras frontales, de orientación perpendicular a esta dirección, mostrarán el primer color, rojo en nuestro ejemplo. Y las laterales, el segundo color, verde en nuestro ejemplo.

Pero, además, tenemos la posibilidad de manipular la curva de transición entre los dos colores, lo que da amplias posibilidades para combinarlos. En las figuras adjuntas se han mostrado algunas de estas muchas posibilidades cuyo interés dependerá de los casos y de la imaginación del usuario. Para que sea más comprensible el resultado se ha utilizado un objeto con aspecto de sombrero, pero que sería asimilable a un terreno más complejo, con zonas planas horizontales y verticales y una zona curva, regular, de transición entre ambas.

Mapas de mezcla

Hay varios tipos de materiales y mapas que permiten combinar propiedadees y texturas de varios modos. Los principales, por orden de complejidad, son los siguientes, que se utilizan, de modo muy similar, en diferentes programas: los mapas de máscara (*Mask*), los materiales o mapas de mezcla (*Blend*) y los materiales o mapas de tipo *Composite* o, según la terminología de cada programa, de *capas superpuestas*. Hay otros tipos pero muchos de ellos han quedado obsoletos o son innecesariamente complejos.

Un mapa de tipo *Mask* es el recurso más sencillo que se puede utilizar. Solo tiene dos entradas, un mapa normal y una máscara. Si se aplica un mapa de este tipo al canal *Diffuse* de un material *Standard*, las partes negras del mapa actuarán como un agujero a través del cual se ve el mapa y las partes blancas retendrán el color asignado a *Diffuse*. Pero este uso es excesivamente limitado (solo funciona con el material Standard y no admite otras texturas) por lo que es mejor utilizar una variante un poco más sofisticada, como un mapa de mezcla.

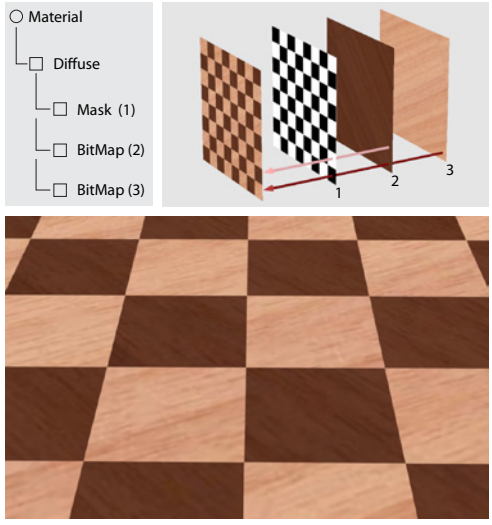


*Figura 5.8 Mapa de mezcla (*Mix*). El esquema superior muestra la estructura del material y la imagen inferior, el resultado de la combinación.*

*Figura 5.9 Mapa de mezcla (*Mix*). En este caso los dos mapas (una textura metálica y un gradiente con repetición) se han combinado superponiendo los valores por medio de un parámetro de mezcla del 18 % (*mix amount*).*

Los mapas o materiales de mezcla (*Blend*) permiten combinar dos mapas o dos materiales. Si todo lo que se quiere hacer es combinar dos texturas basta con utilizar un mapa de este tipo, que, en 3ds Max se denomina *Mix*. El ejemplo de la figura 5.8 muestra un caso muy simple: un tablero de ajedrez que se ha generado con dos texturas de madera, una más clara y otra más oscura, combinadas por medio de una máscara: otra textura que puede ser, como en este caso, un mapa procedural simple de tipo *Checker*. Las partes blancas de la máscara retienen el primer mapa y las partes negras hacen que se traspase el primer mapa para leer los valores del segundo mapa. Otro ejemplo similar se da en la figura 5.9.

Si se quieren combinar propiedades del material, y no solo texturas, necesitamos combinar dos materiales pues los mapas solo incorporan valores de píxeles. En este caso se puede utilizar un material de tipo *Blend* que tiene la misma estructura que el mapa de mezcla pero admite como entrada dos materiales en lugar de dos mapas.

En la figura 5.10 se muestra otro ejemplo de este tipo de combinación. En este caso se ha utilizado un *material* de mezcla (*Blend*) con la estructura siguiente:

Material 1. Tipo *Arch&Design*. Color azul. Reflectividad 1,0, 0,7, 16, metal (*level*, *gloss*, *samples*, relación con *diffuse*). Anisotropía 0.1. Mapa *bump*: mapa de tipo *noise*, blanco y negro, con tamaño aproximadamente un 0,3 % del diámetro del objeto, *threshold* 0,8/0,2 (*high*/*low*). El resultado con este material aislado se muestra en la figura citada (*a*).

Material 2. Tipo *Arch&Design* con un mapa de bits de un metal oxidado como textura. Puede intensificarse el efecto añadiendo una versión en blanco y negro de este mapa como mapa bump. El resultado con este material aislado se muestra también en la figura (*b*).

Mask. Un mapa en blanco y negro generado con una serie de combinaciones de filtros en Photoshop (o escaneado de un patrón que resulte interesante). La figura (*c*) muestra el resultado de filtrar, por medio de esta máscara, los valores del material anterior.

El resultado final de combinar estos dos materiales y la máscara se muestra en la figura 5.10 (*d*).

Material Mezcla con *Vertex paint*

Otro modo interesante de utilizar el material *Blend* es por medio de un modificador especial denominado *Vertex paint*. Este recurso, disponible en 3ds Max, como en otros programas, desde hace varias versiones y que es el antecesor de otras herramientas de pintura 3D más avanzadas, permite pintar directamente sobre los vértices de una malla. Se

Figura 5.10 Material de mezcla: a*) Material 1 aislado;* b*) Material 2 aislado;* c*) Material 2 y máscara;* d*) Resultado conjunto.*

utilizaba (ahora menos, debido a la potencia de otros programas, como ZBrush) para pintar texturas o *Lightmaps* (superficies con gradientes que simulan efectos de iluminación avanzada) en videojuegos, lo que requiere una habilidad considerable que cae fuera de los objetivos de este libro. Pero puede utilizarse para cosas más sencillas como la generación directa de máscaras.

Para comprobar su funcionamiento se puede hacer un ejercicio simple como el que sigue, en el que se pinta un camino sobre un terreno. Antes de empezar el ejercicio habrá que conseguir un par de texturas de hierba y arena o grava fina. Luego, crear un pequeño terreno por medio de un plano con al menos 64 x 64 segmentos, convirtiéndolo a malla editable y moviendo los vértices con *Soft selection*. Así obtendremos un resultado básico como el de la primera imagen de la figura 5.11. A partir de ahí, continuar como sigue.

1 Crear un material de tipo *Blend*. Asignar al primer material un tipo *Standard* o *Arch&Design* con una textura de hierba aplicada a *Diffuse*. Asignar al segundo un material del mismo tipo con una textura de arena (o grava o lo que sea) aplicada a *Diffuse*.
2 Presionar el botón *Mask* y escoger un mapa de tipo *Vertex color*. Dejar los valores predeterminados (que solo afectan a los canales, en caso de que quisiéramos utilizar varios).
3 Seleccionar el objeto terreno y asignarle un modificador *Vertex paint*. Al hacer esto se abrirá una paleta especial con herramientas para pintar sobre la malla.

En esta paleta hay muchas herramientas y su descripción completa puede encontrarse en la ayuda de 3ds Max. Pero nos basta con saber hacer lo siguiente: activar el primer icono de la parte superior para que se visualicen los trazos. Activar la herramienta pincel y ajustar su tamaño a lo que interese. También se puede usar la herramienta de borrar para rectificar lo hecho. Tener en cuenta que,

al pintar, se pintan las caras adyacentes del vértice sobre el que se presiona, con un degradado que también se puede ajustar, aunque para este ejemplo bastará con los valores predeterminados.

4 Asignar el material *Blend* al objeto terreno. El resultado será que las zonas pintadas actuarán como una máscara, tal como se muestra en la segunda imagen, central, de la figura 5.11, y los dos mapas se combinarán tal como se muestra en la figura adjunta con lo que, al hacer un *render*, el resultado será como el de la tercera imagen, abajo.

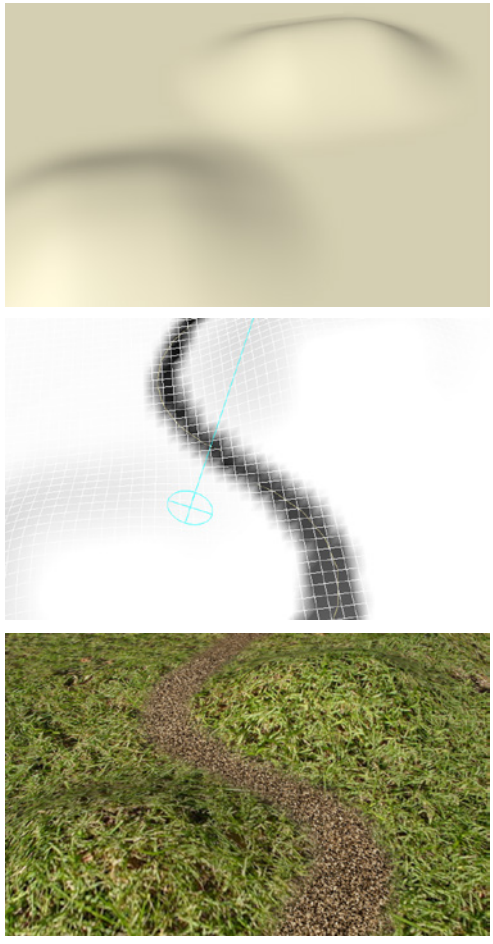


*Figura 5.11 Material Mezcla (*Blend*) con* Vertex paint*: a) Vista del terreno sin materiales añadidos;* b*) Vista del terreno después de pintar la máscara (el camino) con* vertex paint*;* c*) Resultado final.*

Mapas modificadores de color

Si la textura que utilizamos tiene una tonalidad que no es la que interesa es muy sencillo cambiar esta tonalidad desde Photoshop o Gimp pues no hay ninguna dificultad para aplicar una herramienta como Tono/Saturación/Luminosidad (menú Imagen/Ajustes o atajo de teclado ctrl+u) a la totalidad de la imagen.

Pero para esto no es necesario salir del programa si este cuenta con recursos para hacer este tipo de correcciones. En 3ds Max hay una herramienta muy útil, que ya ha aparecido en algunos ejemplos previos, que permite ajustar el color de un mapa sin necesidad de editarlo desde fuera del programa.

Esta herramienta es un mapa especial denominado ***Color correction*** (corrección de color) que se superpone a un mapa dado (al aplicarlo, responder que "sí" a la pregunta de si se quiere conservar el mapa anterior como submapa"). Tiene tres secciones *a*) *Canales*, con controles para convertir el mapa a blanco y negro, invertir los tonos o modificar manualmente los canales principales (R, G, B, alfa); *b*) *Color*, con controles para modificar el tono y la saturación y para superponer un tono a todo el mapa; *c*) *Luminosidad*, con controles para modificar el brillo y el contraste y controles avanzados para ajustar la exposición y los valores de *gamma*. Todos estos ajustes son iguales a los que utilizaríamos con Photoshop o Gimp.

Es un mapa que substituye con ventaja a otros que había en versiones anteriores como *Output* (aplica funciones de modificación global de la salida del mapa, una sección corriente en mapas de bits, a mapas paramétricos que no las tienen) o *RGB Tint*, que tiñe el color de un mapa con valores RGB especificados.

Este mapa está integrado, en 3ds Max, en los mapas de composición que se descri-

ben en el apartado siguiente, por lo que si se utilizan este tipo de mapas no es necesario cargarlo.

Mapas de composición

Los materiales y mapas descritos hasta aquí no permiten más que dos combinaciones. Si se quieren utilizar estructuras más complejas hay que utilizar materiales o mapas de tipo "compuesto", que incluyen recursos para combinar un número mayor de materiales y mapas. En este caso, las posibilidades varían más según los diferentes programas. En 3ds Max, se encuentra el material de tipo *Composite*, más antiguo, que funciona de modo diferente que el mapa del mismo nombre, más moderno y, en general, más práctico. El material permite combinar diferentes materiales diferentes mediante la superposición aditiva, substractiva o mixta de sus colores. Al utilizar un material de este tipo aparecen hasta 10 posibilidades de combinación con una letra/parámetro para elegir el tipo de combinación: aditiva (A), substractiva (S) o mixta (M). La principal ventaja del material compuesto con respecto al mapa compuesto es que permite combinar también las propiedades del material, además de las de textura. Pero esta ventaja es pequeña pues en muy pocos casos interesa y se puede esquivar mediante otros recursos, por lo que me limitaré a describir las posibilidades del mapa.

Un mapa *Composite* solo permite combinar mapas pero cuenta con una serie de recursos de control muy útiles. Se pueden combinar diferentes mapas de modo semejante a como funciona la superposición de capas con modos de fusión (*Blending modes*) en Photoshop o Gimp. La estrategia básica es la misma: capas que se superponen, modos diversos de fusión y máscaras que permiten ver parcialmente las capas inferiores (también pueden utilizarse imágenes que ya incorporen canales alfa). Cada capa actúa sobre la que le precede. La última es dominante.

El editor de 3ds Max incluye varios controles: para crear nuevas capas, para eliminarlas, para renombrarlas, duplicarlas, ocultarlas, etc. Los principales, que se condensan en el esquema superior de la figura 5.12, son tres: *a*) un icono para asignar una textura a la capa (izquierda), *b*) un icono para asignar una máscara (derecha) y, *c*) un botón central que muestra el modo de fusión activo y que, al descolgarlo, da acceso a otros modos de fusión que se resumen en el párrafo siguiente. También hay otros dos controles importantes que no se muestran en esta figura: un control de la opacidad de la capa y un control de ajuste del color para modificar el color, luminosidad, etc., de la capa sin necesidad de editarla en un programa externo. Este control es idéntico al mapa *Color Correction* que he comentado antes.

Los modos de fusión son los corrientes en Photoshop y Gimp y otros programas, con alguna variante adicional. Los resumo telegráficame representando por "A" la capa activa y por "B" la capa que la sigue y sobre la que se aplica el modo de fusión: *Normal* (predeterminado), *Average* (A+B/2), *Addition* (A+B), *Subtract* (B-A), *Darken* (compara A y B y usa el píxel más oscuro), *Multiply* (A*B: como el rango es 0,0 a 1,0 el resultado es siempre más oscuro), *Color burn* (da a los píxeles más oscuros de B el color de A), *Linear burn* (como *Color burn* pero con menos contraste), *Lighten* (compara A y B y usa el píxel más claro), *Screen* (hace las zonas claras más claras y las más oscuras algo más claras), *Color dodge* (da a los píxeles más claros de B el color de A), *Linear dodge* (como *Color dodge* pero con menos contraste), *Spotlight* (como *Multiply* pero dos veces más brillante), *Spotlight blend* (como *Spotlight* pero añade iluminación ambiental), *Overlay* (oscurece o aclara los píxeles según el color de B), *Soft light* (si el color de A es más claro que un gris medio la imagen se aclara y si es más oscuro que un gris medio la imagen se obscurece), *Hard light* (si el color de un píxel es más claro que el gris medio se aplica el modo *Screen*, si es más oscuro se aplica el modo *Multiply*), *Pinlight* (substituye el color de B en función del brillo del color de A: si este es más claro que el

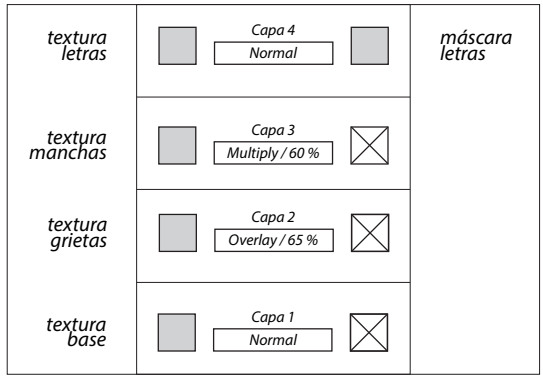


Figura 5.12 Mapa compuesto. Arriba, la estructura de capas. En el centro, las diferentes capas utilizadas. Abajo, el resultado final.

gris medio los colores de B más oscuros que A se substituyen, y viceversa), *Hard mix* (produce blanco o negro según las semejanzas entre A y B), *Difference* (para cada par de píxeles substrae el más oscuro del más claro), *Exclusion* (similar a *Difference* pero con menos contraste), *Hue* (usa el color de A y el valor y la saturación de B), *Saturation* (usa la saturación de A y el valor y el tono de B), *Color* (usa el tono y la saturación de A y el valor de B), *Value* (usa el valor de A y el tono y la saturación de B).

El ejemplo de la figura 5.12 muestra cómo pueden componerse diferentes texturas para crear una textura compleja.

Hay que tener en cuenta que, en este ejemplo, las texturas principales tienen el mismo tamaño, por lo que pueden discurrir por un mismo canal combinando sus colores por medio de modos de fusión. Excepto en el caso de las letras, que se han aplicado con un mapa de menores dimensiones y girado. Dicho de otro modo, el objeto, un plano, tiene asignados dos modificadores UVW, uno que se adapta a todo el objeto y otro que se adapta a las letras. Para que no haya interferencias, en el primer modificador UVW se debe especificar, en el panel del modificador, el canal 1 y en el segundo modificador UVW, el canal 2.

El objeto tiene asignado un material *Arch&Design* (igual podría ser un material *Standard*) y con un mapa de tipo *Composite* asignado a *Diffuse*. La estructura del mapa, que se resume en el esquema de la figura 5.12, es la siguiente (si no se indica la opacidad es que es del 100%):

Capa 1 (textura base). Mapa con una textura de mortero de cemento.
Capa 2 (grietas). Mapa con una textura de grietas, editada para aumentar el contraste y aplicada en modo *Overlay* con una opacidad de un 65%.
Capa 3 (manchas). Mapa con una textura de manchas aplicada en modo *Multiply* y con una opacidad del 60%.
Capa 4 (letras). Mapa con letras de color rojo oscuro, con el *tiling* desactivado para que no se repita. Ligado a una máscara copia del anterior con las letras en blanco y el fondo negro. Las zonas blancas retienen la textura y las negras la vuelven transparente.

En la última sección de este capítulo se muestran otros ejemplos.

5.3 Mapas de aplicación a relieves

El relieve puede simularse por medio de mapas de patrones aplicados específicamente al canal de relieve, que variará según el tipo de material utilizado y según el tipo de mapa de relieve. Hay cuatro variantes principales: mapa de tipo *Bump*, mapa de desplazamiento, mapa de tipo *Normal mapping* y mapa de tipo *Parallax mapping*. Las características principales así como las ventajas e inconvenientes de cada una de ellas ya se han descrito en los apartados correspondientes del capítulo 2. Aquí se describe el procedimiento técnico básico y las peculiaridades del proceso.

Mapas de tipo *Bump*

El mapa es un patrón en blanco y negro. Las zonas blancas alteran las normales creando el efecto de que la geometría sobresale, las zonas negras crean el efecto de que la geometría se hunde y las grises crean efectos intermedios. Véase el capítulo 2 para más detalles sobre el algoritmo utilizado.

El procedimiento básico es el siguiente. En primer lugar, se requiere contar con un mapa de bits con zonas blancas que correspondan a los resaltes, zonas negras que correspondan a los hundimientos y zonas grises intermedias. Las figuras 5.13 y 5.14 muestran dos mapas de bits, uno con solo dos colores, blanco y negro, y otro con grises intermedios en los bordes (lo que será poco apreciable en la impresión).

Este mapa se asocia a un material y el modo de hacerlo puede variar según los programas y los tipos de material. En 3ds Max, si el material es de tipo *Standard* la asignación se hace desde la sección *Maps*, que incluye varios tipos de mapas entre ellos *Bump*, con una intensidad predeterminada, 30 sobre un rango de 100. Si el material es de tipo *Arch&Design* la asignacion se hace desde la sección *Special purpose maps* que incluye igualmente una entrada para este tipo con una intensidad predeterminada, 0,3 sobre un rango de 1,0. Si se asigna este material a

Figura 5.13 Mapa de relieve de tipo Bump *asignado a un plano.*

Figura 5.14 Mapa de relieve de tipo Bump*, suavizado, asignado a un plano.*

*Figura 5.15 Mapa procedural (*Gradient*) de relieve, de tipo* Bump*, asignado a una esfera.*

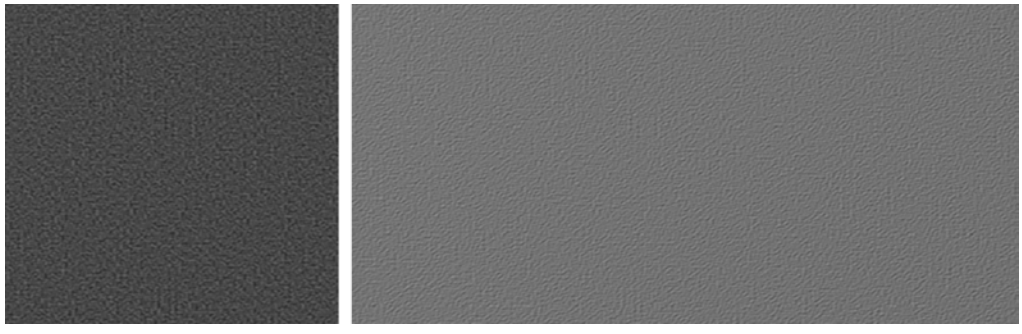


*Figura 5.16 Mapa procedural (*Noise*) de relieve, de tipo* Bump*, asignado a un plano.*

una superficie se crea un efecto de relieve. El efecto se combina con el color local del material o con una textura asignada al color local.

Los mapas de tipo *Bump* resultan particularmente efectivos cuando no hay demasiado riesgo de que se vean los bordes de la geometría real, sea porque quedan ocultos sea porque el relieve se produce en direcciones paralelas a los bordes visibles, sin cruzarlos. La figura 5.15 muestra un mapa procedural (un degradado repetido en la dirección U) que se ha aplicado a una cuerpo esférico para simular relieve. En este caso particular no se aprecia esta limitación.

Los mapas de *Bump* también proporcionan un método sencillo de crear texturas rugosas por medio de mapas procedurales de tipo *Noise* que, si se aplicasen como texturas, darían resultados toscos, pero aplicados como relieve dan resultados convincentes. La última figura de este grupo, la 5.16, muestra el efecto de aplicar un mapa de tipo *Noise* con *Bump* a un plano de color gris.

Si se utilizan en combinación con mapas de bits pueden realzar el efecto de la textura. En la última sección de este capítulo se mostrarán algunos ejemplos adicionales.

Mapas de tipo desplazamiento

Con mapas de *Bump* la simulación falla en los bordes pues la geometría real no refleja el relieve, como ya he comentado anteriormente. Si estos bordes van a ser visibles la única solución es (aparte de modelar el relieve propiamente dicho) utilizar un mapa de desplazamiento.

Con los mapas de desplazamiento el mapa es un patrón en blanco y negro, como en *Bump*. Y como en *Bump*, las zonas blancas alteran las normales hacia arriba, las negras hacia abajo y las grises crean desplazamientos intermedios. Pero a diferencia de los mapas de *Bump*, con este tipo se genera un desplazamiento real de la geometría durante la representación. Véase el capítulo 2 para más detalles sobre los principios teóricos de este tipo de mapas.

El principal inconveniente de este recurso es que resulta costoso en términos de computación. Sin embargo, este inconveniente se va solucionando a medida que pasan los años y las tarjetas gráficas de gama media van incorporando capacidad que hasta hace poco solo estaban disponibles en las de gama alta. Y los principales programas de simulación van incorporando nuevas versiones más eficaces. Dado que las anteriores se mantienen por compatibilidad, esto se presta a cierta confusión. En 3ds Max hay cuatro modos diferentes de generar mapas de desplazamiento que se han ido incorporando a las diferentes versiones: *a*) el primero, que ha mejorado con las sucesivas versiones, está incorporado al motor de *render* corriente y funciona exactamente igual que un mapa de *Bump* con la única diferencia que el tipo es *Displacement map*. Los otros tres están disponibles si se utiliza mental ray, los dos que siguen desde la sección “Mental Ray Connection” del material *Standard* o *Arch&Design* y el último desde la sección indicada de *Arch&Design*; *b*) *3D Displacement*, que incluye parámetros adicionales para modificar la dirección de desplazamiento; *c*) *Height map displacement*, que solo incluye parámetros para la altura de extrusión máxima y mínima; *d*) *Displacement*, que se encuentra en la sección “Special Purpose Map”, del material *Arch&Design*, con un único parámetro para ajustar la distancia de desplazamiento (predeterminada a 1,0). En el ejemplo que sigue se ha utilizado este último que es más reciente y da mejores resultados en menos tiempo.

En cualquiera de estos casos, en 3ds Max, no basta con aplicar el mapa al material sino que también hay que configurar los parámetros de *rendering*. Para ello, ir a *Render setup / Renderer / Shadows&Displacement* y activar el grupo *Displacement* (*Global settings*). Hay dos parámetros importantes (el tercero, “desplazamiento máximo”, especifica el valor máximo, en unidades de la escena, que se desplazará un vértice y no necesita ser modificado a no ser que se quiera un desplazamiento muy intenso o que las unidades no se

correspondan con el valor predeterminado): *Edge length* que determina el tamaño mínimo, en píxeles, de subdivisión de una arista; al alcanzar este valor se detiene la subdivisión. Valores menores hacen que disminuya la longitud de los segmentos subdivididos, lo que equivale a aumentar el número de subdivisiones de las aristas. El otro parámetro es *Max subdiv* (16 K) que controla la extensión de la recursión de la malla geométrica original. Cada paso de la recursión subdivide, en principio, una cara en otras cuatro. El rango va de 4 (4 subdivisiones por cada cara) a 64 K (65.536 nuevas caras por cada cara). En versiones anteriores este parámetro se denominaba *Max level* y tenía un significado distinto (1 *level* era equivalente a 4 subdivisiones, 2 a 16, etc.). Hay que tener el cuenta que la resolución del mapa también influye por lo que, a partir de un cierto valor (0,25 en el ejemplo de las figuras adjuntas) da igual reducir el parámetro *Edge length* y aumentar *Max subdiv*.

Por tanto, para utilizar un mapa de desplazamiento en este programa se requieren

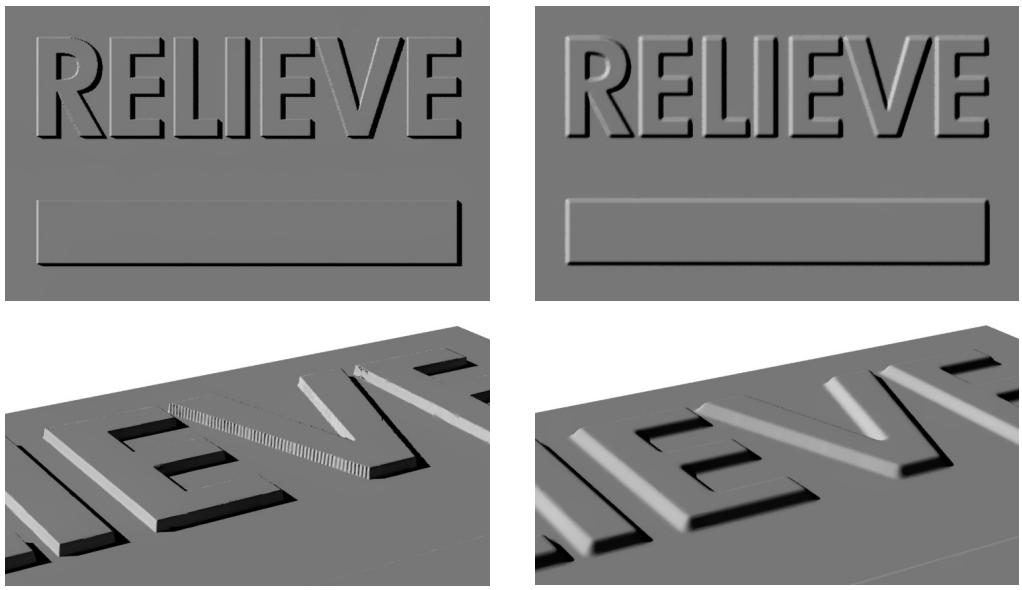


Figura 5.17 Mapa de relieve, de tipo desplazamiento, asignado a un plano. A la izquierda, vista en planta y detalle de una versión sin suavizar, y a la derecha de una versión suavizada.

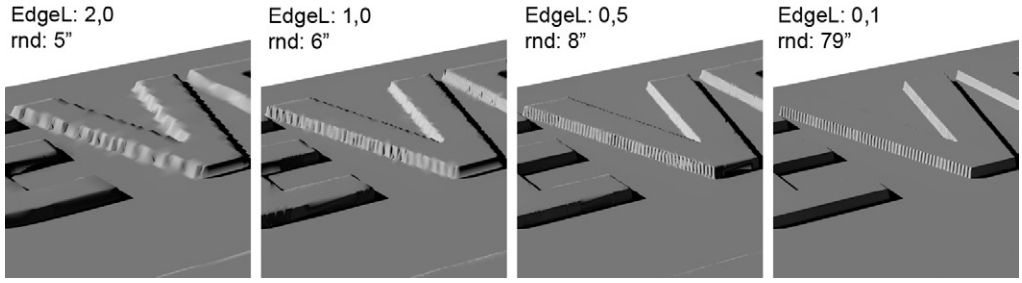


Figura 5.18 Mapa aplicado a nivel de relieve, de tipo desplazamiento, asignado a un plano. A la izquierda, vista en planta y detalle de una versión sin suavizar, y a la derecha de una versión suavizada.

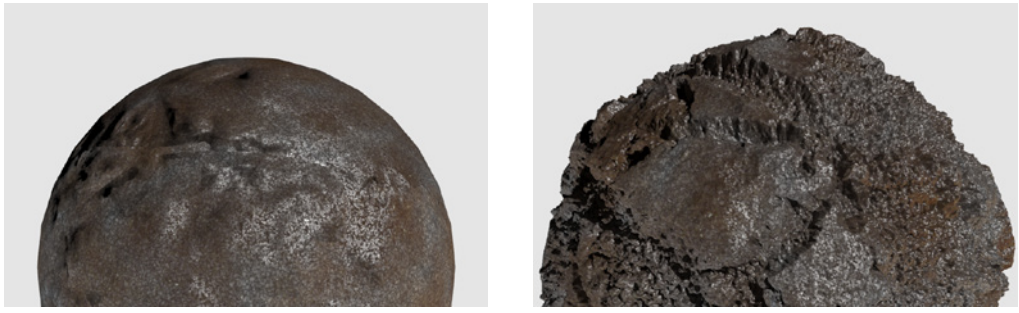


Figura 5.19 Comparación entre la utilización de un mapa de tipo Bump *y un mapa de tipo* Displacement *para simular el relieve de una roca.*

dos pasos: *a*) asignar al objeto un material que incluya un mapa de desplazamiento adecuado, *b*) modificar los parámetros de *render* para que tengan en cuenta cualquier mapa de desplazamiento que esté presente en la escena.

Como puede apreciarse en la figura 5.17, en el detalle (derecha), los resultados son mejores si se utiliza un mapa con los bordes suavizados. Por otra parte, en la figura de detalle (izquierda) con el mapa no suavizado se aprecia mejor la repercusión de los parámetros. La siguiente figura, 5.18, muestra los resultados y el tiempo de *render* al modificar el parámetro principal, *Edge length*. Los resultados de la figura anterior están obtenidos con un valor de *Edge length* de 0,25 que como se puede apreciar es prácticamente idéntico, con menos tiempo de computación, que el valor de 0,1.

La última figura de este grupo, la 5.19, muestra las diferencias de aplicar un mapa *Bump* o un mapa de desplazamiento a un objeto que, en este caso, podría simular una roca o un asteroide. El material es bastante complejo pues incluye varios mapas procedurales combinados con un mapa de tipo *Composite* para simular una textura más interesante. No voy a detallar su configuración pues todo lo que interesa es ilustrar cómo, a partir de una configuración similar, el mapa de desplazamiento permite simular el relieve de un modo bastante más detallado, aunque el tiempo de procesamiento también es considerablemente mayor.

Mapas de tipo *Normal mapping*

Como en los casos anteriores, me remito al capítulo 2 para los principios teóricos en que se basa este tipo de mapa que simula mejor el relieve que los mapas de tipo *Bump*, sin los costes de los mapas de desplazamiento, si bien requiere mayor preparación previa. El mapa, como allí se ha visto, es un patrón de colores rojo, verde y azul que proporcionan información sobre la variación de dirección de la normal y que requiere ser generado por métodos especiales.

Un *Normal map* se puede utilizar de dos modos. En primer lugar, como alternativa a un *Bump map*, pues su procesamiento es más eficiente con las tarjetas gráficas modernas. En segundo lugar, para simular detalles de relieve que son imposibles de simular con los métodos anteriores.

Para lo primero, todo lo que hay que hacer es convertir un *Bump map* en un *Normal map*. Para ello hay varios métodos pero lo más sencillo es utilizar un plug-in para Photoshop que proporciona gratuitamente NVIDIA. El procedimiento, detallado paso a paso, es el siguiente:

1 Descargar un plug-in gratuito de NVIDIA que se puede obtener (en 2014) de http://developer.nvidia.com/object/photoshop_dds_plugins.html.

Si esta dirección hubiera cambiado, hacer una búsqueda con las palabras clave adecuadas. En cualquier caso, una vez que se

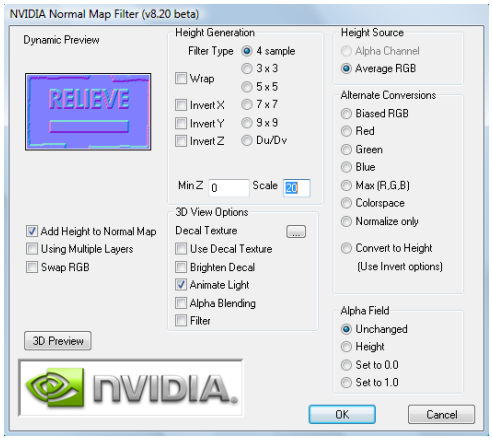


Figura 5.20 Normal maps. *Cuadro de diálogo del* plug-in *de Nvidia para Photoshops que permite convertir un patrón de grises en un* Normal map.

cuenta con este plug-in, desde Photoshop, continuar así:

2 Utilizar un *Bump map* ya existente o crear uno nuevo que consistirá en cualquier caso en una imagen en blanco y negro de tal modo que las partes negras correspondan a zonas hundidas y las blancas a partes resaltadas.
3 Ir a menú *Filtros*. Después de la instalación del plug-in habrá aparecido una nueva entrada, *NVIDIA Tools / Normal map filter*.
4 Al abrirlo aparecerá un cuadro de diálogo como el de la figura 5.20.
5 Rellenarlo con los siguientes valores o probar diferentes alternativas: *Height generation / Filter type*: 4 samples. *Height source*: Average RGB. *Min Z*: 0 (valor mínimo que se asignará a la altura). *Scale*: 8 (valor máximo que se asignará a la altura). *Alpha Field*: Unchanged. Dejar como estén el resto de los parámetros.

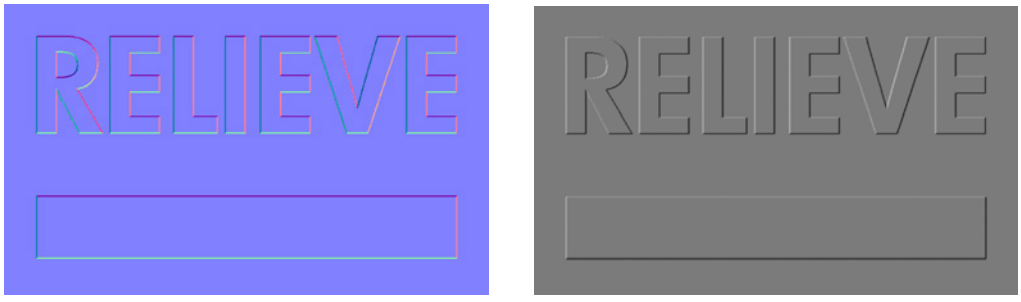


Figura 5.21 Mapa aplicado a nivel de relieve, de tipo Normal map, *asignado a un plano.*

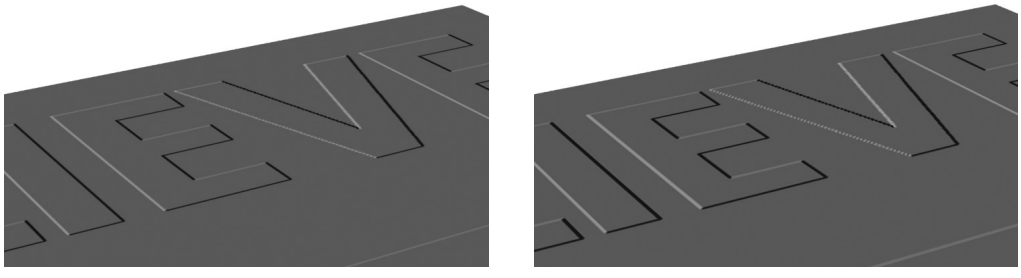


Figura 5.22 Detalle de los mapas de relieve Bump *(izquierda) y* Normal map *(derecha).*

Así se obtendrá un mapa como el de la figura 5.21 (izquierda).

Una vez que se cuenta con este mapa, desde 3ds Max:

6 En el editor de materiales crear un nuevo material *Standard* y, desde la sección de Mapas Básicos, en *Bump*, asignar a *Bump* un mapa de tipo *Normal Bump*. El editor mostrará los parámetros de este tipo. Dejar activado el método por omisión "Tangent" y asignar a "Normal" el mapa que acabamos de crear.

El resultado será como el de la figura 5.21 (derecha), prácticamente indistinguible del de *Bump* pero (si se cuenta con una buena tarjeta gráfica) algo más rápido.

Pero un uso más interesante es utilizar un *Normal map* para hacer que un objeto de baja resolución aparezca como un objeto de alta resolución y, sobre todo, para hacer que el mapa muestre detalles del relieve que serían imposibles de obtener con los métodos anteriores.

Obsérvese la figura 5.23 que muestra un cuadrado con dos rombos a cada lado. ¿Cómo conseguir que un mapa de relieve de tipo *Bump* o *Displacement* simule las variaciones que se aprecian en los niveles de gris de un relieve que corresponden a variaciones en la inclinación de los planos? Obviamente esto no es posible a no ser que incorporemos la información geométrica de algún modo en el mapa de relieve, que es precisamente lo que hace un *Normal map*.

Pero para incluir esta información geométrica no podemos utilizar el método anterior, el plugin de NVIDIA, pues este se genera a partir de un mapa de grises que no puede incorporar esta información. Se necesita crear un

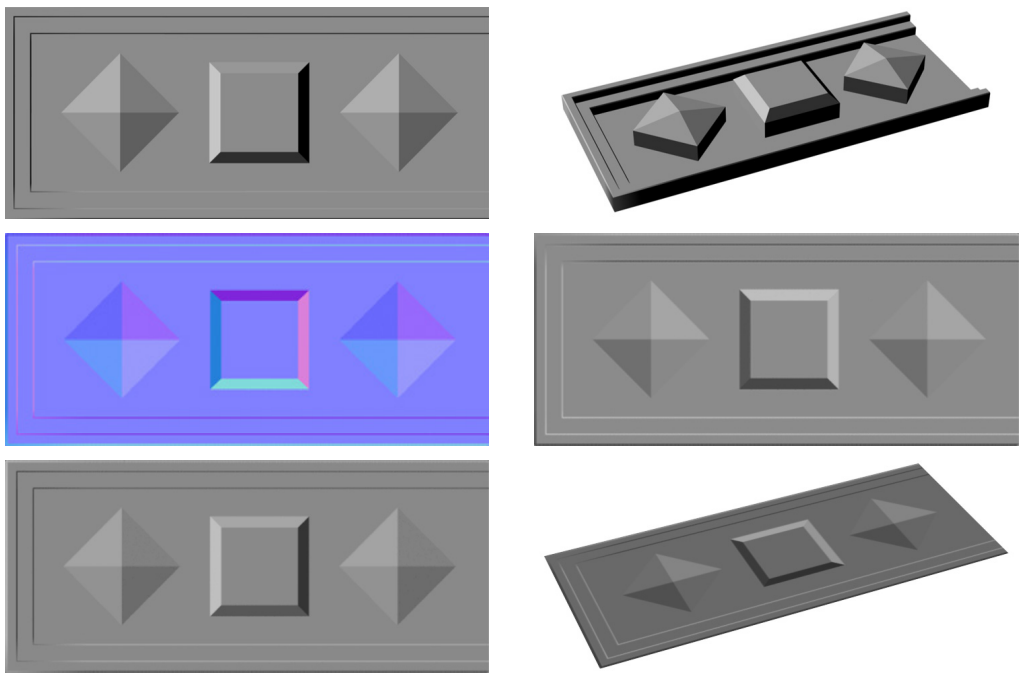


Figura 5.23 Mapa aplicado a nivel de relieve, de tipo Normal map*, asignado a un plano. El mapa se ha generado a partir del objeto real que se muestra en la primera fila, en planta y alzado. La segunda fila muestra el mapa, a la izquierda, y su aplicación a un plano, a la derecha. La tercera fila muestra la misma aplicación con otra dirección de luz y, a la derecha, una vista del plano en perspectiva.*

Normal map a partir de un modelo real. Pero entonces ¿de qué nos sirve el mapa si tenemos que modelar el elemento? La respuesta es que no nos ahorrará tiempo de trabajo de modelado pero sí tiempo de procesamiento. En una escena con múltiples detalles de relieve, el tiempo de procesamiento de estos detalles, si están modelados con geometría real, puede ser muy alto. Al incorporarlos a mapas de tipo *Normal map* estaremos ahorrando horas de tiempo de cálculo. Esto es particularmente importante en animación y en escenarios interactivos. Debe tenerse esto bien presente pues si todo lo que nos interesa es crear imágenes fijas, es bastante probable que la diferencia de tiempo no compense el tiempo que se tarda en crear un *Normal map* a partir de un modelo real aunque, como se verá a continuación, el procedimiento es bastante sencillo.

El proceso resumido consiste en: *a*) crear dos versiones de un mismo elemento, una a baja resolución y otra a alta resolución, *b*) situar las dos versiones en la misma posición geométrica, *c*) utilizar un programa especial (que en nuestro caso será *Render to texture*, incluido en 3ds Max, aunque hay otras alternativas) para generar el *Normal map*. Este programa asignará un mapa vacío al modelo de baja resolución y, por cada píxel del mapa, enviará un rayo perpendicular a la superficie del modelo de baja resolución hacia el modelo de alta resolución para, en el punto de intersección con este último, registrar la normal correspondiente al modelo de alta resolución. Revisar la descripción teórica de este proceso que se ha dado en el capítulo 2 antes de continuar. Y revisar también el apartado sobre *Render to texture* al final de este capítulo.

Para comprender bien el proceso, comenzar por un modelo simple pero adecuado, como el de la figura 5.23, y seguir los siguientes pasos:

1 Crear dos versiones de un mismo objeto con relieve: una detallada y otra plana. Bautizarlos con nombres tales como “objAltaRes” y “objBajaRes”. Superponerlos de modo que ocupen exactamente la misma posición en la escena. En la figura se muestran separados para poder visualizarlos pero deben estar superpuestos
2 Asignar al objeto con baja resolución un *Unwrap map* (seleccionar el objeto y, en el panel *Modify*, desplegar la lista de modificadores y escoger *Unwrap map*).
3 Manteniendo el objeto con baja resolución seleccionado, entrar en *Render to texture* (presionar 0 o ir a menú *Rendering* / *Render to texture*). Este cuadro de diálogo tiene varias secciones cuya descripción completa se puede encontrar en la ayuda. Aquí me limitaré a indicar qué hay que hacer en cada sección para lo que aquí interesa.
   – En *General settings*, presionar *Setup*. Con esto se entra en *Render setup*. Ir a la sección *Renderer* y marcar la casilla “Enable global supersampling” (esto no es estrictamente necesario pero mejorará el resultado).
   – En *Objects to bake*, comprobar que aparece el objeto seleccionado en la lista de objetos. Luego presionar el botón *Pick*, en el grupo *Projection mapping*, y seleccionar el objeto de alta resolución en el panel que se abrirá y que mostrará los objetos presentes en la escena.
   – En *Output*, *a*) presionar el botón *Add* para escoger un tipo de *Texture element* y en la lista de elementos posibles escoger *Normals Map*, *b*) junto a “File Name and Type”, dar un nombre y una ubicación al mapa de bits que se generará al final de proceso, por ejemplo, “pruebaNormalMap.tga”, *c*) asignar una resolución de salida al archivo, escoger valores bajos para pruebas (por ejemplo 256) y valores superiores (por ejemplo 1.024) para la salida final aunque esto dependerá de cada caso particular. Para este ejemplo, escoger un valor bajo (256 o 512).
   – En *Baked material* y en *Automatic mapping* no hacer nada.
4 Antes de continuar, comprobar, desde la escena, las proyecciones del elemento. Al

escoger el objeto de alta resolución como fuente de proyección se habrá creado automáticamente un modificador *Projection*. Si se despliega, y se selecciona el subobjeto *Cage*, aparecerá una caja envolvente que deberá recubrir el objeto (en casos complejos puede ser necesario editar manualmente esta caja para asegurarse de que todo el objeto queda bien envuelto).

5 Por último, comprobar que está seleccionado un visor que muestre la vista frontal (en planta o en alzado) del objeto que nos interesa y, en la parte inferior del cuadro de diálogo *Render to texture*, común a todas las secciones, presionar *Render*. Probablemente se emitirá un mensaje del tipo "The following elements do not specify a Target Map slot:", seguido del nombre del objeto a baja resolución. No tenerlo en cuenta y presionar el botón "Continue".

6 El objeto se procesará y el resultado aparecerá en la ventana de salida corriente de *Render*. El aspecto no tendrá nada de particular. Pero en paralelo se ha creado otro archivo. Para comprobar que este archivo que se ha creado en paralelo contiene información del relieve, codificada en gamas de rojo, verde y azul, localizar el archivo y abrirlo. El resultado deberá ser un mapa con píxeles de diferentes tonos que representan direcciones de normales, codificadas en tonos intermedios entre rojo, verde y azul que representan otros tantos desplazamientos parciales en *xyz*.

Una vez que contamos con este mapa, el resto del proceso es similar a lo que ya hemos visto. Desde el editor de materiales, crear un nuevo material, asignar a mapa básico *Bump* un mapa de tipo *Normal bump*, escoger el mapa que hemos creado y asignarlo al objeto de baja resolución. El resultado será como el de la figura 5.23, que simula perfectamente el relieve aunque cambiemos la posición de la luz. Habrá que evitar de todos modos las vistas muy inclinadas para evitar que se muestre la geometría real, que seguirá siendo plana.

Mapas de tipo *Parallax* y *Parallax occlusion mapping*

Este mapa, que combina hasta cierto punto las ventajas de los mapas de tipo *Bump* y de tipo *Displacement*, pues da mejores resultados que el primero en menos tiempo que el segundo, es relativamente reciente y no está disponible en los editores corrientes aunque sí en algunos programas de juegos de vídeo, como Unity o Cry Engine entre otros. Al igual que los anteriores, requiere dos mapas, uno que represente la textura y otro, en blanco y negro, que represente las alturas a que debe desplazarse la textura al aplicar el mapa.

Pero *Parallax mapping* tiene limitaciones que conviene tener en cuenta. El proceso de implementación resulta costoso en términos de computación. Por otro lado, el resultado resulta plano y poco natural debido a que el algoritmo original no computaba la oclusión y las sombras autoarrojadas (*Self-shadowing*).

Como ya he indicado en el capítulo 2, esto ha dado lugar a desarrollos alternativos como *Relief mapping* y *Parallax occlusion mapping* (POM) que, en particular esta última, parecen claramente preferibles, entre otras cosas porque sacan más partido de la programabilidad de las GPU modernas e incluye sombras autoarrojadas, sombras suaves y mejores controles de *aliasing*. *Parallax occlusion mapping* utiliza *ray tracing*, desde la GPU, por cada píxel, para estimar la visibilidad de los puntos computados junto con sistemas de estimación del nivel de detalle (LOD) para optimizar el cálculo y reducir el tiempo de cómputo. Los resultados, que pueden verse en algunos foros de internet (hacer una búsqueda en particular con el nombre de la técnica y el de uno de sus autores, Natalya Tatarchuk, que ha hecho presentaciones de resultados en Siggraph) son claramente superiores en calidad a los de *Bump mapping* o *Normal mapping*.

Sin embargo, sea porque además de ser costoso, su eficacia no está garantizada (en algunos foros se sostiene que, habida cuenta de las capacidad de las GPU modernas, pue-

de ser preferible modelar la geometría real, sobre todo en algunos casos), sea porque la integración de este tipo de técnicas es lenta, el caso es que, hasta la fecha (2014) no se encuentra en ninguno de los grandes programas de simulación. Lo que no descarta que la situación cambie en poco tiempo, razón por la que incluyo este apartado. Hay algunos plug-ins para Maya pero su uso tampoco parece estar muy extendido.

En mental ray (para 3ds Max, Maya o SoftImage) es posible utilizar un recurso, *mental mill*, que se podía descargar, en 2012 de NVIDIA. Con este recurso se pueden generar *shaders* que se aplican a través de MetaSL. Y entre los *shaders* disponibles está *Parallax mapping* (pero no *Parallax occlusion mapping* aunque sería posible incluir el algoritmo, que está disponible por Internet). Pero solo es recomendable esta vía para quien tenga algunos conocimientos de programación y bastante tiempo y paciencia.

5.4 Mapas de aplicación a recortes

Ya hemos visto en el capítulo 2 que los mapas de recorte se han utilizado desde los inicios de la computación gráfica para crear *sprites* y, posteriormente, *billboards*, figuras que aparecen recortadas sobre los objetos o fondos de la escena en que se insertan.

En las versiones antiguas de los programas de simulación esto requería utilizar dos mapas: uno con la imagen normal y otro que consistía en una versión en blanco y negro de la misma imagen, de tal modo que las partes blancas correspondiesen a la imagen que se quiere conservar y las negras, al fondo sobre el que se recorta la figura y que se quiere eliminar. Este segundo mapa se utiliza como máscara y el procedimiento se detalla en el apartado que sigue.

Sin embargo, como este uso es muy corriente, en muchos programas se puede cargar directamente un formato de mapa de bits que incorpore canales alfa y el sistema puede reconocer este canal y aplicar una transparencia al fondo. En algunos casos, por ejemplo con SketchUp, este reconocimiento es automático. En otros casos, hay que activar algunas opciones para que este reconocimiento sea efectivo. En el apartado siguiente se detallan ambos procedimientos.

Procedimientos básicos de aplicación

a) Mapas de recorte aplicados al canal de opacidad.

El procedimiento antiguo, pero que puede seguir siendo de interés en algún caso, es aplicar un mapa combinado, asignando el mapa que incluye la imagen de la figura a *Diffuse* y el mapa que contiene el patrón en blanco y negro al canal de *Opacity* de un material *Standard* (o al canal *CutOut* de un material *Arch&Design*). Por tanto, para esta alternativa clásica, habría que preparar dos mapas de bits: el primero, que podemos llamar “figura.jpg” representa la figura que queremos simular sobre un fondo de color uniforme; el segundo, que podemos llamar “figuraBN.jpg” (figura en blanco y negro) se genera fácilmente a partir de la anterior, en Photoshop o Gimp, seleccionado el fondo uniforme y pintándolo de negro y luego invirtiendo la selección y rellando la parte correspondiente a la figura de blanco.

Una vez que contamos con estos dos mapas, el procedimiento sería el siguiente:

1 Crear un material de tipo *Standard*.
2 Asignar “figura.jpg” a *Diffuse*. Comprobar las propiedades para que la reflexión sea nula pues podría interferir en el efecto.
3 Asignar “figuraBN.jpg” a *Opacity*.
4 Crear un plano de dimensiones adecuadas y de la misma proporción que la imagen, orientarlo de tal modo que su cara principal quede perpendicular a la cámara y asignarle este material. O bien utilizar los recursos que se describen más adelante para que se oriente automáticamente hacia la cámara.

Eso es todo. Al hacer un *render* la figura quedará recortada contra el fondo.

b) Mapas de recorte integrados

El procedimiento más corriente, desde hace unos cuantos años es utilizar un único mapa con un canal alfa integrado. El formato más corriente para esto es PNG aunque también se puede utilizar GIF, TIFF o TGA de 32 bits.

En este caso, habrá que preparar un único archivo que incluya un canal alfa. Si se trabaja con Photoshop esto supondrá hacer algo tan sencillo, en principio, como cambiar el nombre de la capa *Fondo* para que el programa reconozca que no es una capa universal sino una capa propia de Photoshop y que, por tanto, admite canales alfa. Luego seleccionar el fondo y borrarlo (aparecerá, en lugar del color original, un patrón característico de pequeños cuadros blancos y grises). Por último, guardarlo en alguno de los formatos indicados, por ejemplo, como "figura.png".

Una vez que se cuenta con este mapa, el procedimiento sería el siguiente:

1 Crear un material de tipo *Arch&Design*.
2 Asignar "figura.png" a *Diffuse*. Comprobar que la reflexión y la transparencia son nulas para que no interfieran en el efecto.
3 Asignar el mismo mapa a *Cutout*, en la sección S*pecial purpose maps*. Es decir, repetir el proceso o, mejor, copiar el mapa (BDR (botón derecho ratón) / *Copy* sobre el icono con una "M", junto a *Diffuse*) y pegarlo sobre el botón correspondiente de *Cutout*.
4 Editar el mapa copiado y, en los grupos *Mono channel output* y *RGB channel output*, marcar las opciones *Alpha* y *Alpha as gray* respectivamente. De este modo se procesarán los valores del canal alfa y se recortará la figura.

También pueden utilizarse los mismos procedimientos para crear una figura con solo color. En este caso, bastará con utilizar el mapa de recorte (o un mapa en blanco y negro).

Si interesa que la figura arroje sombra, nos encontraremos con el problema de que la orientación de la luz no tiene porqué coincidir con la orientación de la cámara. Y, por consiguiente, la sombra producida por una luz en dirección más o menos perpendicular a la cámara será una figura alargada que acabaría por convertirse en una línea si la orientación es perpendicular.

Para solucionar este problema un recurso efectivo es duplicar el plano al que está asignado el material con el mapa recortado y hacer que este duplicado quede orientado hacia la luz mientras que el original quede orientado hacia la cámara. Luego editar las propiedades de cada uno de estos objetos y hacer que el plano orientado hacia la luz no reciba ni arroje sombras mientras que el orientado hacia la luz arroje sombras pero sea invisible en la representación. La lista de propiedades que se encuentra (en 3ds Max) al activar el menú contextual (botón derecho del ratón) permite modificar estas propiedades y unas cuantas más.

Para hacer que estos planos se orienten automáticamente hacia la luz o hacia la cámara, seguir las indicaciones que se dan en el apartado siguiente. También convendrá modificar el punto de pivote para que coincida con el centro del lado inferior del plano y hacer también que la figura de referencia tenga el control centrado. Esto nos ahorrará varios ajustes manuales.

La figura adjunta muestra un ejemplo del tipo de resultado que se obtiene con estos métodos.

Control de la orientación del plano de la figura (restricción *Look at*)

Para que los recursos anteriores funcionen adecuadamente, el plano sobre el que se aplica el material con el mapa debe quedar orientado hacia la cámara. Esto puede hacerse manualmente, lo que no requiere grandes explicaciones, o automáticamente, lo que depende del programa utilizado.

En algunos programas simples, como SketchUp, esto se resuelve de un modo muy sencillo aunque con menos posibilidades de control adicional. Todo lo que hay que hacer es convertir el plano en un componente y marcar la opción “Mirar a la cámara” (*Look at camera*).

En otros programas puede ser algo más complicado debido a que este recurso se inserta en un contexto con posibilidades adicionales.

En 3ds Max puede hacerse con facilidad de dos maneras: por medio de sistemas de partículas, como veremos en el apartado correspondiente más adelante, un método particularmente recomendable cuando hay muchas figuras iguales. O bien mediante un procedimiento como el que sigue, más adecuado si no hay demasiadas figuras.

1 Seleccionar el plano.
2 Ir al panel *Motion / Assign controller*. En la lista de transformaciones que aparecerá desplegada, seleccionar *Rotation* (que por defecto mostrará Euler XYZ). Presionar el botón *Assign controller* que hay en la parte superior para escoger otro tipo de control. En el panel que se abrirá escoger *LookAt constraint*. Con esto, *Rotation* substituirá el control predeterminado (Euler XYZ) por el escogido. Y en la parte inferior de este panel aparecerá una nueva sección “LookAt constraint”.
3 Desplegar este panel y activar el botón *Add LookAt target*. Seleccionar la cámara (puede seleccionarse cualquier objeto).
4 Al seleccionar la cámara el plano girará para orientarse en función de la cámara. Pero la orientación no será la que nos interesa pues lo más probable es que los ejes predeterminados no sean los adecuados. Para cambiarlos, en el grupo “Select LookAtAxis” y en el grupo “Aligned to Upnode” escoger como opción *Z*. Dejar el resto de los parámetros tal como estaban (en el grupo *Select Upnode* mantener como opción *World*, en el grupo *Upnode control* mantener como opción *Axis alignment* y en *Source Axis* mantener como opción Y).

De este modo, al mover la cámara el plano se mantendrá perpendicular a la dirección de la visión.

Si queremos controlar la dirección de la sombra será necesario repetir el mismo procedimiento pero escogiendo, al activar *AddLookAtTarget*, la luz (o el *daylight system*, en su caso).

Como decía más arriba, convendrá que el punto de pivote del plano esté situado, en cualquiera de los dos casos, en el centro del lado inferior para facilitar la alineación.

Por añadidura, como decia en el apartado anterior, hay que cambiar las propiedades de los planos. Para ello:

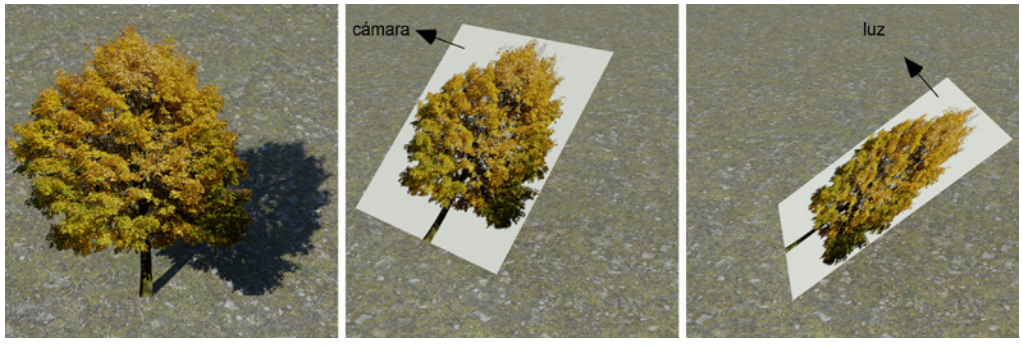


Figura 5.24 Mapas de recorte: a) *Vistas del resultado final desde la cámara;* b) *Vista lateral mostrando solo el plano orientado hacia la cámara que no arroja ni recibe sombras;* c) *Vista lateral mostrando solo el plano superpuesto orientado hacia la luz que arroja sombras pero es invisible en la vista final.*

1 Seleccionar el plano orientado hacia la cámara y editar sus propiedades (*BDR* / *Object properties*).
2 En la sección *General*, en el grupo *Rendering control*, pulsar el botón *By layer* para que queden disponibles las propiedades por objeto (*By object*).
3 Desmarcar las opciones *Cast shadows* y *Receive shadows* (esto último para que el otro plano no arroje sombras sobre este).
4 Seleccionar el plano orientado hacia la luz y repetir la misma operación, pero en este caso desmarcar solo la opción *Visible to camera*.

De este modo veremos el objeto orientado hacia la cámara (que no arrojará sombras) y la sombra correspondiente al objeto orientado hacia la luz (que no será visible).

Canales alfa y fusión de imágenes. Valores alfa premultiplicados

En la industria cinematografíca es corriente la composición de imágenes por medio de un conjunto de técnicas que reciben varias denominaciones aunque su finalidad es la misma: *green* o *blue screening*, *keying*, *color differences* o *chroma key*. En estos procedimientos de composición, una de las dos imágenes representa a los actores y la otra a un exterior sobre el que se quiere fusionar las imágenes filmadas, en estudio, de los actores.

La primera película filmada con esta técnica parece que fue *El ladrón de Bagdad* de Alexander Korda, en 1940. Luego fue desarrollada por Lawrence Butler (que obtuvo un Oscar por su trabajo). Butler utilizó una pantalla azul como fondo para filmar las secuencias y luego consiguió separar el color azul con técnicas relativamente primitivas que dejaban un rastro perceptible en los bordes de los personajes (*color fringe*). En la década de 1950, Petro Vlahos desarrolló un sistema denominado *difference processing* que simplificó notablemente la separación del color y eliminaba casi totalmente

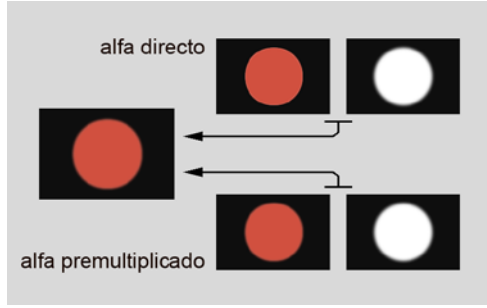


Figura 5.25 La imagen que se muestra a la izquierda puede ser el resultado de la combinación de dos imágenes con valores RGB y alfa separados o de dos imágenes en las que la primera ya ha sido combinada con la segunda (los bordes del círculo rojo inferior son difusos aunque quizás no se aprecie en la impresión).

los defectos de los bordes y que se ha mantenido hasta no hace mucho. Las técnicas digitales actuales facilitan la tarea aunque hay muchos aspectos complejos que deben tenerse en cuenta. Actualmente se tiende a utilizar el verde en tomas exteriores para que no se confunda con el cielo, de ahí el nombre alternativo *green screening* que ha tendido a substituir a las “pantallas azules” o *blue screening*. Es la misma técnica que se utiliza en televisión para, por ejemplo, dar los partes metereológicos.

Estas técnicas se han ido adaptando a los sistemas de simulación visual y, en ellas, juega un papel fundamental, como ya hemos visto, el canal alfa, que es el que posibilita la fusión de imágenes al “abrir agujeros” en la imagen que se encuentra en primer plano, “agujeros” que permiten visualizar la imagen que está en segundo plano.

Sin embargo, hay un problema de difícil solución, debido a que los contornos de las imágenes reales son de un color en el que participan tanto el color del objeto que está en primer plano como el de fondo. Es un fenómeno que ya observó Masaccio, en la primera mitad del siglo xv, y que le llevó a fusionar los contornos de un modo característico que se insertaba en una evolución histórica hacia un mayor realismo en la pintura.

Como veremos en los apartados siguientes, hay diversas técnicas que se pueden utilizar para conseguir este efecto y prevenir la aparición de halos en torno a la figura en primer plano. Pero la idea básica es, en cualquier caso, conseguir que la transparencia en los bordes se produzca de un modo gradual para mantener este efecto de fusión con el fondo, sea cual sea, sobre el que se va a colocar la imagen.

Ahora bien, esta fusión puede producirse directamente, al activar la representación de la imagen, o puede ajustarse *a posteriori*, mediante efectos de postproducción. Si este va a ser el caso, puede interesar mantener separados los valores alfa para facilitar este trabajo de postproducción. Y, en muchos casos, por ejemplo en los parámetros del mapa *Cutout* de mental ray que hemos visto en el apartado anterior, nos podemos encontrar con una opción para mantener activada, o no, la opción de "canales alfa premultiplicados" (*premultiplied alpha channels*).

Aunque para las finalidades de este libro tanto da, en general, activar o no esta opción, es importante saber en qué consiste pues en determinados casos puede ser necesario llevar a cabo un encargo o un trabajo del tipo que sea que implique la postproducción posterior. El significado de esta opción está relacionado con el modo en que los valores alfa se almacenan en los archivos lo que, en el argot de este tipo de aplicaciones se denomina *Straight alpha* frente a *Premultiplied alpha*.

Una imagen con un canal alfa no premultiplicado aplica el valor del canal alfa a cada uno de los canales RGB antes de generar la imagen. Una imagen con un canal alfa premultiplicado almacena los valores RGB con el valor de cada canal ya multiplicado por el valor del canal alfa, con lo que la gestión es más eficiente. Y, por añadidura, algunas operaciones, como la combinación lineal de valores, también funcionan mejor. Sin embargo, esto puede afectar a la postproducción.

Por ejemplo, si un archivo tiene valores RGBA y un determinado píxel tiene el valor (1,0, 0,8, 0,6, 0,5) el almacenamiento premultiplicado sería el correspondiente a multiplicar los tres canales por el último valor, el valor alfa, manteniéndolo en el resultado. Es decir que tendríamos como valor RGBA resultante (0,5, 0,4, 0,3, 0,5). O dicho de otro modo, la intensidad de cada color primario se habrá reducido, en este caso, a la mitad tal como indica el valor alfa. Esto no afectará a los valores blancos (pues se multiplicarán por 1,0) o los valores negros (pues se multiplicarán por 0,0) pero sí a los valores intermedios. Y, como decía, esto no afecta al resultado inmediato pero sí afecta al caso en que queramos la imagen para componerla posteriormente pues resulta más complicado extraer el valor alfa si está premultiplicado.

Preparación de imágenes. Técnicas básicas de ajuste y suavizado de los bordes

Todo lo anterior depende, por tanto, de que contemos con imágenes bien recortadas y que incluyan canales alfa (o imágenes asociadas en blanco y negro que para el caso es lo mismo). Y, por tanto, nos encontraremos con dos alternativas: que estas imágenes las hayamos adquirido de terceros o que tengamos que generarlas nosotros mismos porque no encontramos exactamente lo que buscamos.

En este segundo caso, el principal problema con que nos podemos encontrar con los recortes es la aparición de imperfecciones en los bordes. Estamos ante un problema clásico, el mismo que se encuentran, como decía en el apartado anterior, quienes trabajan en la industria del cine o de animación y deben extraer el *chroma key* el color de fondo, azul o verde, de las pantallas sobre las que se ruedan escenas que serían imposibles o muy dificultosas de rodar y que dan lugar a contornos defectuosos, *fringes*, debido a que el fondo se cuela entre pequeños huecos. Un ejemplo típico es el pelo de los actores que, cuanto más suelto esté más mostrará fragmentos del color del fondo muy

difíciles de eliminar. Y otro tanto ocurre con las ramas delgadas de los árboles o cualquier elemento fino.

Para eliminar estos defectos hay programas especializados y métodos manuales relativamente sofisticados. Y términos como *Light wrapping* o *Light spill* que aluden a técnicas específicas para recuperar el modo en que los contornos de los objetos quedan envueltos por la luz que proviene de los objetos que están tras ellos, con lo que su color es un sutil degradado que va del color del objeto al color del fondo. Cuando falta este efecto o, peor aún, cuando es substituido por un fondo que ya no existe (el fondo del que se ha extraido la imagen), el resultado es una imagen de mala calidad, como puede verse a veces en cine o televisión, en montajes antiguos o de bajo presupuesto.

Si no se quiere entrar en este terreno, la primera recomendación es evitar figuras con bordes de este tipo. Y la segunda recomendación, si esto no es posible, es utilizar programas que cuenten con recursos adecuados para extraer los bordes y dedicar unas cuantas horas a familiarizarse con estos recursos.

Con Photoshop se pueden utilizar varias técnicas. Si el fondo es relativamente uniforme se obtendrán resultados adecuados con relativa facilidad utilizando la herramienta *Borrador de fondos*, que selecciona el color que cae en el centro del cursor (marcado con una pequeña cruz) y borra tan solo los colores similares, en un rango dado por el parámetro de transición. Otra herramienta más sofisticada, que ha mejorado considerablemente en las versiones recientes, es *Perfeccionar borde* que incluye varias herramientas complementarias para rastrear una determinada zona detectando transiciones bruscas que se interpretan como transiciones entre figura y fondo. Puede ajustarse este radio de búsqueda y puede dejarse al programa que haga este rastreo de modo automático o dibujando sobre las zonas que nos interesa separar.

Si estos recursos son insuficientes se puede intentar seleccionar el canal RGB en donde la transición es mayor (que por lo general suele ser el azul), duplicarlo, aumentar el contraste con las herramientas de ajuste (niveles, mapa de curva) y convertir el resultado en una selección inversa que se puede utilizar para borrar el fondo. Véase la figura adjunta en que se ilustra la fase principal de este proceso para un árbol con ramas muy finas.

En este caso, como en los anteriores, puede convenir convertir el resultado en una máscara de capa y retocar el resultado pintando con blanco (zonas opacas, que se mantendrán) y negro (zonas transparentes). Véanse los comentarios que añado en el apartado siguiente sobre las máscaras de capas.

Por último, si ninguno de estos métodos es suficiente, como por desgracia ocurre con

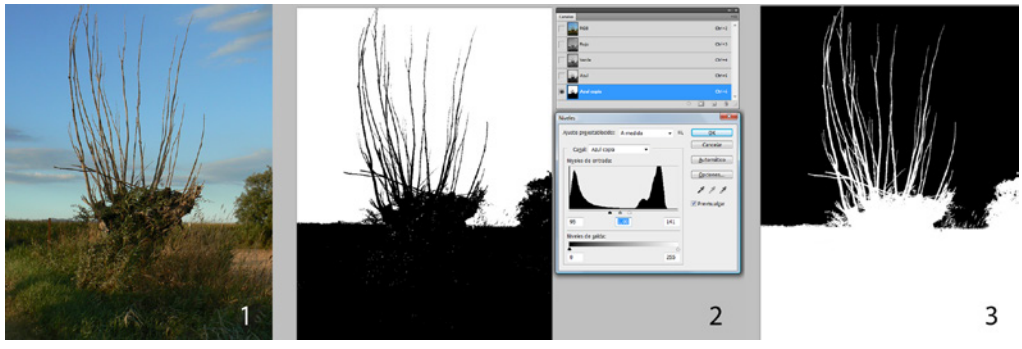


Figura 5.26 Recorte de una imagen con elementos muy finos, difíciles de separar del fondo, duplicando uno de los canales RGB, aumentando el contraste y convirtiendo el resultado en una máscara de capa.

fondos no uniformes que se mezclan con las figuras de primer plano de tal modo que solo cabe decidir discrecionalmente qué es figura y qué es fondo, habrá que armarse de paciencia, utilizar la herramienta pluma para crear un trazado marcando puntos y luego convertir el resultado en una selección.

No puedo entrar en el detalle de todos estos métodos pero pueden encontrarse muchos ejemplos y tutoriales en internet pues se trata de un problema bien conocido por quienes se dedican al retoque de imágenes.

El ***suavizado de los bordes*** consiste ni más ni menos en crear una transición suave entre los bordes de la figura y el fondo transparente. Aunque se puede hacer manualmente será preferible utilizar máscaras de capa que facilitan el trabajo. Y las máscaras de capa se pueden controlar de un modo mucho más eficaz, en Photoshop, con la ayuda del panel de máscaras. También se puede utilizar el panel anterior, *Perfeccionar borde*, ajustando los controles del grupo *Ajustar borde* para suavizar más o menos las transiciones.

Si no se sabe bien como trabajar con máscaras de capa, hacer el ejercicio siguiente, muy simple. Y si se conoce, saltar al siguiente párrafo: *a*) Abrir una imagen que incluya alguna figura que destaque claramente sobre un fondo; *b*) Abrir el panel de capas y duplicar la capa; *c*) Seleccionar la figura de un modo aproximado; *d*) Aplicar una máscara de capa: aparecerá otra miniatura al lado de la de la capa mostrando, en blanco, la zona seleccionada y, en negro, el resto de la imagen; *e*) Ocultar la capa original. Se visualizará tan solo la parte seleccionada (el blanco de la miniatura) de la capa duplicada pues el resto (la zona negra) es transparente y ahora no hay nada que transparentar pues hemos ocultado la capa original que había quedado debajo. Si pintamos con blanco se recupera opacidad y si pintamos con negro se recupera transparencia. Podemos visualizar la máscara en blanco y negro pulsando Alt+clic sobre su icono. Y convertir la máscara en una selección pulsando Ctrl+clic.

También se puede entrar en el panel de *Perfeccionar borde* (de la máscara en este caso) situándose sobre el icono y activando el menú contextual. Nos encontraremos con las opciones que ya he comentado antes y que, entre otras cosas, nos permitirán hacer los bordes más o menos suaves para controlar las transiciones.

Después de hacer todos estos ajustes, activar el comando *Aplicar máscara* (menú contextual en el panel de capas). La máscara desaparecerá y el efecto de lo que hemos hecho quedará integrado en la capa, lo que nos permitirá usar otros recursos adicionales.

Otra opción simple que puede aplicarse en casos más sencillos es aplicar un desenfoque de tipo gaussiano a toda la imagen, con un radio no demasiado alto. Luego, con las herra-

Figura 5.27 Suavizado de los bordes de una figura con un canal alfa: a*) figura con bordes no suavizados,* b*) la misma figura después del suavizado.*

mientas de *Desenfoque local* y *Dedo* (*smudge*), retocar los bordes en puntos críticos para fundirlos suavemente con el fondo.

Por último, eliminar las capas auxiliares si las hubiera y guardar el archivo en un formato que preserve el canal alfa, como PNG.

El procedimiento anterior implica que la figura se fusionará con suavidad sobre cualquier fondo sobre el que se coloque. Esto mejora el resultado considerablemente. Al igual que ocurre en una fotografía digital de una escena real, los bordes de la figura se fusionarán con las diferentes partes del fondo.

Representación directa sobre un fondo alfa

Otra alternativa que se puede utilizar en algunos casos, para evitar el problema de los bordes intricados que puede que resulte muy difícil de separar del fondo, es representar directamente la imagen sobre un canal alfa. Pero tiene el obvio inconveniente de que solo se puede hacer con imágenes sintéticas y modelos virtuales 3D, lo que limita considerablemente el campo de aplicación. Sin embargo hay algunos casos en los que merece la pena utilizar esta vía, particularmente en el caso de árboles y arbustos. Hay muchos programas que permiten simular de un modo extraordinariamente realista todo tipo de vegetación, como Onyx Tree. Y hay otros programas, entre ellos 3ds Max, que también incluyen este tipo de recursos aunque no con la riqueza de posibilidades del mencionado. Naturalmente, se pueden usar directamente los elementos vegetales en 3D que se incluyen en 3ds Max como en otros programas, lo que nos ahorra tener que utilizar mapas de recorte. Pero si el número de elementos de este tipo es grande el coste de computación puede ser inasumible por lo que merecerá la pena obtener imágenes 2D a partir de los elementos 3D. Y otra ventaja de utilizar imágenes 2D es que se pueden retocar y ajustar con facilidad.

El siguiente procedimiento es muy simple y se puede adaptar a cualquier tipo de programa que incluya estos recursos. En 3ds Max, habría que hacer lo siguiente:

a) Crear un árbol 3D (panel *Create / AEC Extended / Foliage*). Escoger un tipo y darle unas dimensiones adecuadas. En el ejemplo he utilizado un olmo (*american elm*) con una altura de 12 metros.

b) Situarse en una vista frontal. Cambiar el tamaño de salida a 512 x 512. Activar *Safe frames* y centrar el árbol en el marco, procurando que la base del tronco quede en el centro y pegada al borde inferior. Editar el árbol y modificar los valores de *density*, *pruning* y *seed* hasta que la distribución de ramas y hojas sea satisfactoria.

c) Desde el editor de materiales, captar el material del árbol. Nos encontraremos con 5 submateriales que corresponden al tronco, las ramas y las hojas). Asignar al color del

Figura 5.28 Mapas de recorte. Mapas de bits de árboles 2D generados a partir de árboles 3D.

tronco y las ramas un mapa de ruido de tonos marrones. En el ejemplo se ha utilizado un mapa *Noise*, *size* 0,8, *fractal* con un tono marrón claro y otro oscuro. Asignar a las hojas otro mapa de ruido similar pero con un tamaño algo menor y más contraste (*size* 0,3, *fractal*, *threshold* 0,8/0,2 en ejemplo) y colores verde oscuro y verde claro.

d) *Render*. Comprobar que se ha grabado correctamente el canal alfa (presionar el botón alfa: el árbol deberá quedar blanco y el fondo negro). Guardar en un formato TIF, marcando la opción "store alfa" o PNG marcando la opción "alpha channel".

e) Repetir el procedimiento 3 o 4 veces variando la configuración (*density*, *pruning*, *seed*) y los colores.

f) Guardar el resultado en un formato adecuado que preserve el canal alfa.

Las imágenes de la figura 5.28 muestran tres variantes que se han obtenido con este método.

5.5 Mapas de modificación de reflexiones

Se utilizan mapas para modificar reflejos por dos razones principales: para alterar las propiedades reflectantes de zonas determinadas de la superficie de un material y para substituir las imágenes reflejadas por otras imágenes. En el primer caso se utilizan imágenes en blanco y negro que, de modo similar a lo que hemos visto con los relieves y lo que veremos con transparencias y otras propiedades, hacen que las partes blancas computen el reflejo, las negras no lo computen y los valores intermedios lo computen parcialmente. En el segundo caso se utilizan imágenes en color para substituir a las imágenes que corresponderían al reflejo real, sea para corregir defectos debidos al incorrecto procesamiento de imágenes de fondo, sea porque interesa utilizar diferentes imágenes para el reflejo. Los dos apartados siguientes desarrollan estos modos principales de aplicación de mapas a reflejos.

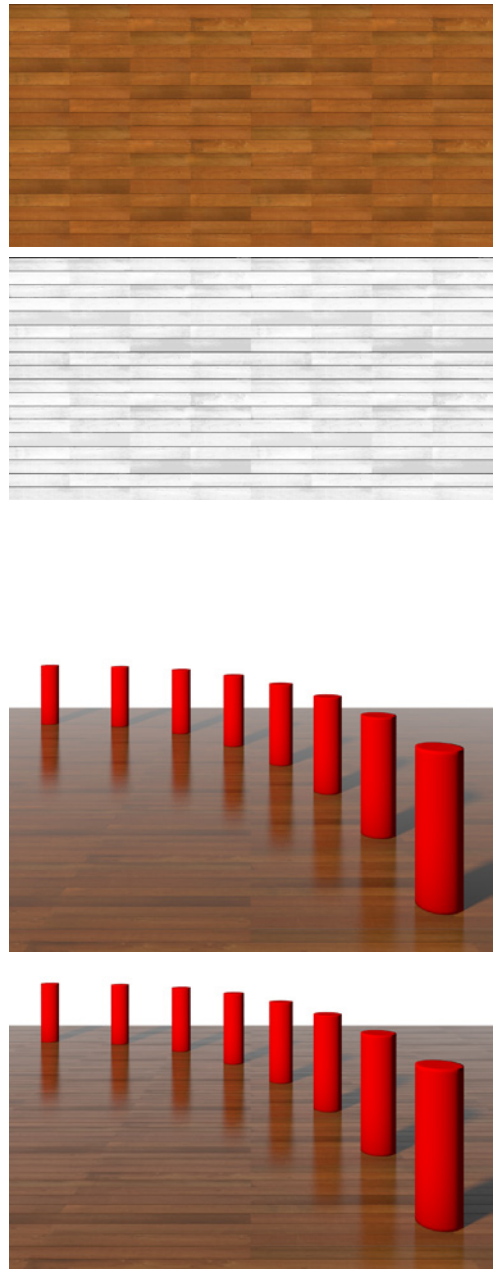


Figura 5.29 Mapas de reflexión. Reflejos sobre un plano con mapa de reflexión monocromático: a*) mapa utilizado como textura,* b*) mapa monocromático de corrección del reflejo,* c*) resultado sin el segundo mapa,* d*) resultado corregido con el segundo mapa.*

Modificación de reflejos por mapas monocromáticos

Si utilizamos un patrón en blanco y negro asignado al canal de reflejos el efecto será, como ya hemos visto, que las partes blancas retendrán el reflejo y las negras lo anularán (y las grises lo retendrán o anularán parcialmente). El efecto es similar a un *Bump map* y, en muchos casos, puede conseguirse un resultado de interés combinando ambos modos, pues un mapa de relieve virtual también altera el reflejo haciendo que se anule en unas zonas y se resalte en otras. Pero el control sobre este efecto es mayor si se asigna el mapa al canal de reflejos pues estamos actuando directamente sobre esta propiedad.

Las aplicaciones posibles dependen de los casos que se puedan presentar y de la imaginación del usuario. Pero el uso más corriente y, a menudo, imprescindible, es para modificar adecuadamente una superficie que, por sus propiedades físicas reales, puede ser reflectante en unas zonas y mate en otras. El ejemplo más corriente es el de un pavimento brillante con juntas. Por ejemplo, un pavimento de madera pulimentada será reflectante. Pero las juntas entre las piezas no lo son. Si se aplica un mapa que represente una textura y se hace que el material sea reflectante, tanto las partes correspondientes a la madera como las correspondientes a las juntas reflejarán lo que tengan enfrente, lo que no ocurre en la realidad. La solución es utilizar dos mapas en lugar de uno: un mapa que corresponda a la textura del material y otro, en blanco y negro, derivado del anterior para que la correspondencia sea exacta, en el que las partes correspondientes a las juntas sean negras y el resto, blanco. Si la madera es vieja y tiene grietas, estas también deberán dejarse de color negro pues tampoco son reflectantes. Y, en general, hacer que las zonas correspondientes al reflejo no sean completamente blancas sino que mantengan algo de la textura del original de un modo variable contribuirá a aumentar el contraste y hará más interesante la simulación.

Las imágenes de la figura 5.29 ilustran este uso básico por medio de la misma escena que se ha utilizado en el capítulo anteriormente en la sección correspondiente a reflejos. El material también es el mismo, un material *Arch&Design* con las mismas características que la última figura de aquella serie. Pero, en el grupo *Diffuse* se ha substituido el color por una textura de madera y, en el grupo *Reflection*, se ha substituido el color de reflexión blanco por un mapa (pinchar en el pequeño cuadrado a la derecha de este parámetro) en blanco y negro, derivado del mismo mapa de textura, convirtiendo los colores a blanco y negro, aumentando el contraste y retocando manualmente algunas zonas.

Como puede comprobarse en la figura 5.29 (y como se comprobará aún mejor en un monitor si se rehace este ejemplo), el resultado es claramente mejor, más contrastado y más realista en el segundo caso. Puede replicarse este ejemplo con diferentes variantes, haciendo, por ejemplo, que las zonas negras correspondan a grietas para simular madera vieja, etc.

Modificacion de reflejos en mapas de entorno

Cuando no existían los programas de simulación basados en técnicas de *ray tracing* u otras técnicas capaces de simular la interacción de la iluminación con los diferentes objetos de una escena, los reflejos se simulaban por medio de proyecciones de entorno (*environment mapping*). En el capítulo 2 he descrito las caracteristicas de este método que aún se sigue utilizando en algunos casos.

Pero los mapas de entorno se utilizan también para corregir problemas derivados del uso de las técnicas corrientes de simulación de reflejos, y para ampliar las posibilidades de estas técnicas. En este apartado resumiré tres usos característicos: el uso de mapas asignados al canal de entorno del material para corregir, localmente, errores en el procesamiento de los reflejos del fondo; el uso de mapas asignados al mapa de entorno de

Figura 5.30 Mapas de reflexión. Reflejos sobre un plano con mapa de entorno: a) *Vista de la escena con un mapa de fondo y un plano, no reflectante, situado en primer plano;* b) *La misma escena haciendo el plano reflectante (reducido ligeramente para que se aprecie la posición del plano): el fondo se transparenta en lugar de reflejar;* c) *Resultado con el mismo mapa asignado al canal* environment *del material del plano y con los parámetros reajustados, tal como se indica en el texto.*

toda la escena para corregir globalmente estos errores, y el uso de mapas de entorno diferentes para controlar los reflejos de un determinado material.

Consideremos el caso siguiente que es una prolongación del ejemplo anterior. En lugar de utilizar un fondo blanco será más normal que queramos utilizar una imagen de fondo. Esto puede hacerse de dos maneras. Una consistiría en crear un plano vertical, más allá del plano horizontal del pavimento y asignarle una textura. De este modo simularemos un fondo que se procesará como un objeto más. Esto no planteará ningún problema técnico pero es un procedimiento engorroso: tenemos que crear el plano en la posición correcta, asignarle un material con la textura adecuada, iluminarlo de modo exclusivo para no modificar la iluminación de la escena, etc. Es obvio que resulta mucho más sencillo asignar a la escena un fondo virtual sin tener que preocuparse por todos estos detalles.

Esto es lo que se ha hecho en la segunda imagen de la figura 5.30. Pero, como puede comprobarse en esta figura, el resultado es incorrecto: el fondo no se refleja adecuadamente. Y esto es bastante lógico pues estamos utilizando técnicas de *ray trace* con las que se envían rayos desde la cámara, que rastrean la escena y desvían los rayos en función de las propiedades de las superficies con que se encuentran hasta toparse con otras superficies y volver a desviarse. Pero el fondo virtual no está en la escena. No es un objeto como los demás con una superficie que se cruce en el camino del rayo trazador. Y los rayos que la cámara computa para modificar el color de la imagen en función del reflejo no provienen de la escena sino del fondo virtual.

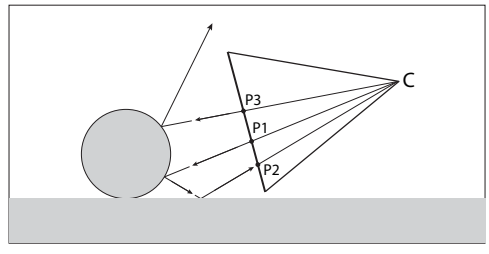


Figura 5.31 Mapas de reflexión. Esquema que muestra el desfase del punto de reflejo dado por un mapa de entorno ligado a la pantalla (P1) y el punto de reflejo real (P2).

El esquema que se incluye en la figura 5.31, muestra como el punto que correspondería a un reflejo real no tiene porque coincidir con el punto computado por la imagen. El punto P1 es el punto de intersección del rayo enviado desde la cámara hacia la escena con la pantalla virtual. Dado que el reflejo depende del mapa de entorno, un mapa de bits que coincide con la pantalla, este punto es el que se utilizará para cualquier cómputo ligado a este rayo, incluido el cálculo del "reflejo" proveniente del fondo, lo que es claramente incorrecto pues si este reflejo correspondiese a un objeto real, como la esfera que se muestra en la imagen, el punto correspondiente a su reflejo sería el punto P2. Por otro lado, la adición de estos puntos daría lugar a una imagen invertida, cosa que tampoco ocurre.

El problema con que nos encontramos puede solucionarse asignando al material reflectante, el pavimento de base en nuestro ejemplo, una copia (no una instancia) del mapa de fondo a través de un canal especial, un canal de entorno que está incorporado a los materiales avanzados como *Arch&Design*. Y manipulando sus parámetros para que se ajuste a lo que necesitemos. Esto es lo que se ha hecho en la tercera imagen de la figura 5.30 . El procedimiento que he seguido en este ejemplo ha sido el siguiente:

1 Desde el editor de materiales, editar el material *Arch&Design* asignado al pavimento. Desplegar la sección *Special purpose maps*.
2 Desde el panel *Environment* (menú *rendering* o tecla de atajo 8) desplegar la zona en que se muestra el mapa asignado al fondo. Arrastrar este mapa de entorno desde el panel *Environment and effects* al material del pavimento que se muestra en el editor de materiales, concretamente sobre el botón que hay junto a *Environment*, en la sección *Special purpose maps*. En el pequeño cuadro que se abrirá escoger la opción *Copy* (no *Instance*), pues cambiaremos sus parámetros de modo independiente.
3 Editar el mapa. Comprobar que, está asignado a *Environment* y que el tipo de *mapping* es *Screen*. En *Coordinates*, hacer *V Tiling* igual a -1.0 para que la imagen se invierta. Luego, si fuera necesario (no lo ha sido en este ejemplo) cambiar el valor de *V Offset* de tal modo que el mapa se sitúe adecuadamente. Esto dependerá de cada caso.

Así hemos corregido este problema. Pero ¿qué pasaría si, en lugar de un material al que afecta este problema tenemos varios? Tendríamos que hacer la misma operación para todos los objetos y todos los materiales a los que afectara este problema. Lo que es poco eficiente.

Figura 5.32 Mapas de reflexión: a) *Vista de un piso del Ensanche de Barcelona (representado con un sistema de iluminación avanzada);* b) *Esfera con material* A&D *reflectante situada en el entorno anterior.*

La solución, para este segundo caso, más general, y que tiene ventajas adicionales importantes es utilizar un *shader* especial de mental ray que se denomina *Environment / Background switcher*. Este *shader*, muy sencillo, tiene dos canales o dos entradas principales: una que afecta al fondo y otra que afecta a los reflejos. Esto nos permite utilizar un mapa para el fondo y otra para los reflejos, es decir, hacer de un modo global lo que acabamos de hacer de un modo particular. El mapa utilizado para los reflejos puede ser el mismo que el utilizado para el fondo pero con los parámetros modificados del modo que interese. O bien puede ser otro, lo que, entre otras cosas, nos permitiría asignar a objetos con sus zonas reflectantes orientadas hacia la cámara un mapa correspondiente a otra parte del fondo que rodea la escena.

El procedimiento, en este caso, sería el siguiente (no lo ilustro porque el resultado final sería muy similar al de las imágenes anteriores):

1 Asignar como mapa de entorno (panel *Environment*) un mapa de tipo *Environment / Background switcher*.
2 Arrastrar (en este caso como *instancia*) este mapa al editor de materiales para editarlo.
3 Asignar el mapa de fondo a *Background* y a *Environment*. Editar el primero y comprobar, en la sección *Coordinates*, que está asignado correctamente (*Environment / Screen*). Dejar los parámetros predeterminados. Editar el segundo y hacer las mismas comprobaciones pero modificar los valores de *V tiling* del mismo modo que en el caso anterior (invirtiendo y desplazando). El resultado será el mismo pero ahora se ha aplicado a nivel general.

Con todo, determinados objetos podrían requerir ajustes específicos. Por esta razón, hay que tener en cuenta que el uso de este *shader* va más allá de lo explicado pues, en general, se utiliza en conjunción con un mapa de entorno esférico que permite computar con precisión valores globales para toda la escena. Como esto se utiliza muy a menudo para procedimientos avanzados de iluminación con HDR, concretamente sistemas de tipo IBL (*Image Based Lighting*) se explica con más detalle en los capítulos correspondientes del libro de simulación de la lluminación por lo que me remito a esas explicaciones.

Para casos específicos, para solucionar problemas concretos como los descritos aquí, lo más sencillo será utilizar un mapa de entorno asignado a un objeto concreto y un material concreto.

Esto nos lleva al tercer caso, un tercer caso igualmente importante en la práctica, en que interesa utilizar mapas diferentes para el fondo y para el reflejo. Esto puede ser debido a dos razones distintas: que los reflejos sean efectivamente distintos pues vienen de partes muy diferentes del mapa de fondo y no nos interesa utilizar un mapa de fondo que abarque los 360º del entorno que rodea a la escena, entre otras cosas porque necesitaríamos una resolución muy alta, o bien, sencillamente, que queremos economizar tiempo de cálculo, y en una escena en la que el fondo reflejado por un objeto correspondería a objetos situados detrás de la cámara resultará más eficiente, en términos de computación, utilizar un entorno pregrabado que no hacer que se envíen cientos de miles de rayos alrededor para computar el reflejo proveniente de estos objetos.

Considérese el ejemplo que se ilustra en las figuras 5.33 y 5.34. La escena representa una habitación de un piso del Ensanche de Barcelona. Incidentalmente, en la sección sobre texturas, más adelante se explica también cómo se han generado las texturas del suelo

Sobre el suelo se ha colocado una esfera y la cámara está situada de tal modo que la esfera refleja el balcón que se ve de frente en la figura superior. A la esfera se le ha aplicado un material *Arch&Design*, reflectante, y por tanto refleja correctamente este balcón que está situado del lado de la cámara.

Ahora supongamos que por las razones que sea no contamos con el modelo de este

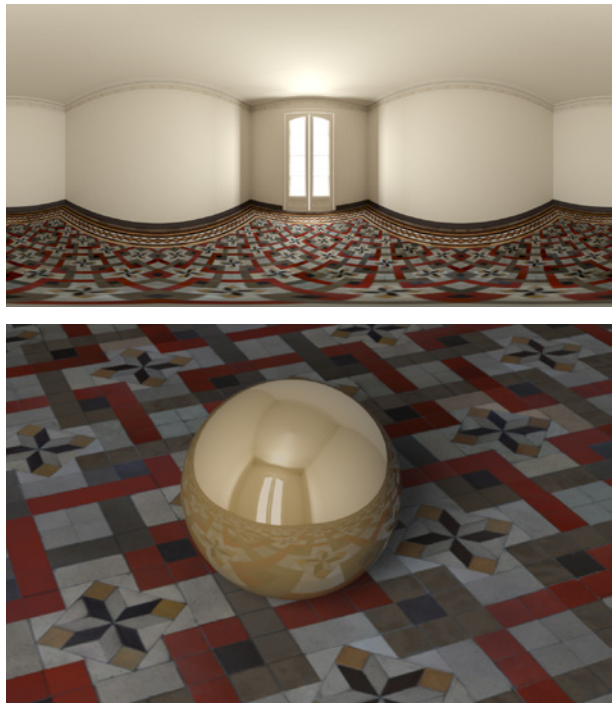


Figura 5.33 Mapas de reflexión: a) *Mapa panorámico derivado del piso del Ensanche anterior y utilizado como mapa de entorno en;* b) *Esfera con material* A&D *reflectante. Mismas luces que en las figuras anteriores (tipo* mr Area spot *para primaria, secundaria y reflejo).*

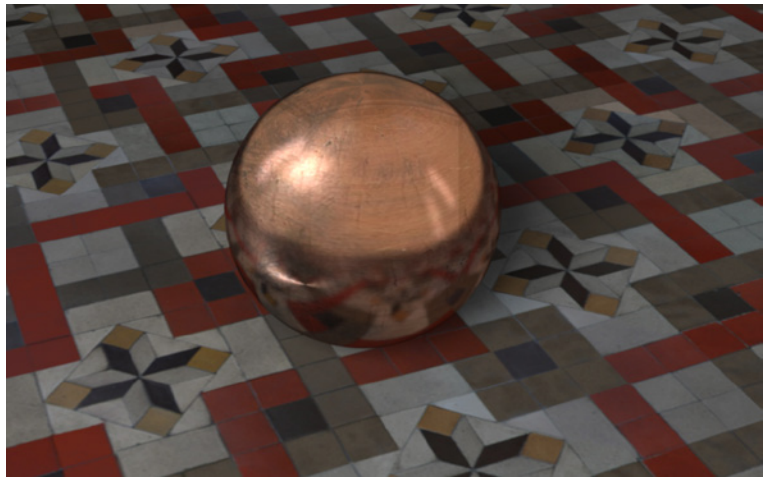


Figura 5.34 Mapas de reflexión. Resultado final añadiendo objetos especulares y modificando el material para simular cobre, con los parámetros indicados en el texto.

entorno o no queremos contar con él. Podríamos substituir el modelo real, la geometría real, por un mapa. Para que la comparación sea más efectiva he generado un panorama de este entorno, que es el que se muestra en la imagen superior de la figura 5.33 y, en la siguiente imagen, he eliminado el entorno. Lo que queda, por tanto, es simplemente una esfera situada sobre un plano. Tenemos, por tanto, un escenario mucho más simple aunque, en este caso, no supone una gran diferencia. Pero que si lo extrapolamos a una escena interior con un gran número de objetos o a una escena exterior con docenas de edificios es evidente que supondrá una enorme diferencia, no solo en términos de tiempo de cálculo sino, sobre todo, en términos de tiempo dedicado a la elaboración del modelo.

Para procesar esta escena el material asignado a la esfera será el mismo que en el ejemplo anterior, es decir, un material *Arch&Design* con los mismos valores de reflectividad y *glossiness* (1,0 en el ejemplo). No es la configuración que más nos interesaría, pues es poco realista, pero he hecho que el objeto sea plenamente especular para que se aprecien mejor las diferencias.

El resultado es similar al obtenido en el caso anterior, con el escenario real. Sin embargo, si se observa con atención se comprobará que no es exacto lo que resulta lógico pues en el primer caso tenemos un rastreo real de la escena y en el segundo, una aproximación mediante un mapa de entorno desplegado en un panorama que no mantiene una correspondencia exacta con lo que se obtendría trazando rayos desde la esfera. Pero es más que suficiente para lo que queremos, sobre todo si ajustamos los reflejos para que no sean totalmente especulares, como ocurriría en la realidad.

Esto es lo que se ha hecho en la imagen de la figura 5.34. Si se añade una textura, si se añaden realces especulares mediante luces virtuales que no alteren la iluminación (seleccionarlas y desactivar, con el menú contextual, la contribución a *Diffuse*) y si se reduce el porcentaje de reflectividad y *glossiness*, el resultado será más real y la percepción del fondo quedará más dispersa, por lo que su exactitud geométrica tiene menor importancia.

Fuga o sangrado de color (*color bleeding*)

Otro problema relativamente corriente, relacionado con los reflejos es lo que se denomina *fuga de color* o *sangrado de color* o *color bleeding* en la literatura técnica anglosajona. Si un objeto de un color muy saturado está situado cerca de otro objeto de colores más claros, este último quedará teñido de los colores del primero. Este efecto es real, pero puede resultar excesivo y a menudo interesar reducirlo, sea porque la mayor intensidad sea debida a desajustes técnicos difíciles de controlar, sea por razones estilísticas, porque no siempre interesa que el realismo interfiera con el efecto que se quiere conseguir.

Dado que este efecto depende del sistema de iluminación avanzada utilizado, pues dará diferentes resultados según el método utilizado, no se puede desligar su corrección del sistema escogido. Si, por ejemplo, se utiliza radiosidad, hay un material especial (*architectural material*), adecuado para radiosidad que incluye un control específico para este efecto. Para no complicar la exposición partiré por tanto de dos supuestos. En primer lugar, que uno de los sistemas más efectivos en la actualidad es *Final gather* con mental ray, equivalente a *Irradiance* con otros programas, y que esto justifica que me limite a este caso. En segundo lugar, que el lector tiene un conocimiento mínimo de esta técnica que también se explica extensamente en el libro de simulación de la iluminación citado varias veces.

Partiendo de estos supuestos, el *color bleeding* con mental ray se puede controlar por medio de un *shader* especial, *Color overide / Ray type switcher* o, en versiones anteriores, por el *mr Raytype switcher* que, al igual que otros *shaders* creados por Zap Anderson y agrupados en los *production.mi*, estaba oculto desde la versión 2008 (para desocultarlos había que localizar el archivo produc-

tion_max.mi, buscar “rayswitch” y escribir el signo “#” delante de “hidden” para que esta palabra clave se procesase como un comentario y no se tuviera en cuenta al arrancar 3ds Max. Una vez hecho esto, al arrancar el programa y buscar un mapa desde el editor de materiales, aparecía este *shader*, *mr Raytype switcher* en la lista de mapas de mental ray).

Este *shader* consta de nueve controles para rayos: *Eye*, *Transparency*, *Reflected*, *Refracted*, *Final gather*, *Environment*, *Shadows*, *Photon* y *Default*. Cada uno de estos controles permite modificar el color de los rayos secundarios que se utilizan en los cálculos de iluminación. El primero corresponde a los colores básicos que llegan al observador: lo primero que hay que hacer, por consiguiente, es asignar a este control el mismo color o mapa que se quería utilizar para el material original. A partir de ahí, lo más sencillo es copiar este mismo color o mapa al tipo de rayo cuyo color se quiere ajustar y modificarlo ligeramente (por ejemplo, rebajar la saturación).

Puede resultar algo desconcertante, si se utiliza el *shader Color override* / *Ray type switcher*, que cada uno de estos controles no tenga asociado un selector de color sino tan solo un botón de acceso a un mapa. ¿Qué hacer entonces si se está creando un material que solo tiene color, no mapa? La respuesta es utilizar un “mapa que asigna colores” (lo que es un tanto retorcido pero así lo han decidido los diseñadores de este *shader*). Este “mapa que asigna colores” es el mapa *Color correction*, un *shader* que ya hemos visto, muy útil para modificar los colores de cualquier mapa que tenga por debajo pero que también sirve para asignar directamente colores a un material. Todo lo que hay que hacer es utilizar el selector de color que aparece en la primera sección de este mapa, en *Basic parameters*, entre “Map” y el botón de asignación de mapa. Más exactamente, si ya se contaba con un material con un color determinado, copiar el color del material de que partíamos (situarse encima del icono de color y hacer BDR/Copy) y pegarlo encima del color de *Color correction* (con BDR/Paste).

El procedimiento básico, cuando se utiliza cualquiera de estos dos *shaders*, se puede ilustrar con el ejemplo siguiente:

1 Crear una escena muy simple, como la de la figura 5.35, que es una especie de frontón con tres objetos: “muros” (las paredes del frontón), “suelo” y “bola”. Crear también una luz que ilumine la escena desde un lado.
2 Asignar un material con un color neutro a los muros. En el ejemplo se ha utilizado un

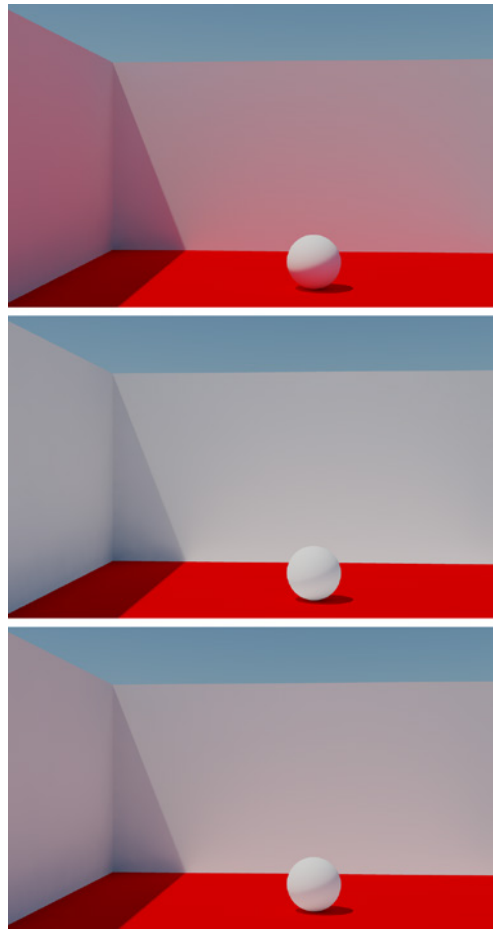


Figura 5.35 Control del sangrado de color con el shader Ray switcher*: a) Sin aplicar controles de reducción de sangrado;* b*)* Ray switcher *con saturación 0,2;* c*)* Ray switcher *con saturación 0,5.*

material *Arch&Design*, con la reflectividad y transparencia en 0 y un color blanco roto (RGB 235,235,235 o 0,92,0,92,0,92) asignado a *Diffuse*. Asignar un material de color vivo al suelo. En el ejemplo he utilizado el mismo material que para los muros pero con un color rojo (RGB 255, 0, 0) asignado a *Diffuse*.

3 Hacer un *render* con FG (*Final gather*) configurado con valores bajos para no perder tiempo (solo nos interesa el color). El resultado será como el de la figura 5.35 (*a*) en la que se aprecia claramente que los muros han quedado teñidos con el color del suelo.
4 Asignar al material *Arch&Design* un mapa a nivel de *Diffuse* y escoger el *mr Color override / Ray type switcher* (recordar que con este *shader* la modificación de color debe hacerse como se ha explicado antes, a través de un mapa).
5 Copiar el color rojo original asignado a *Diffuse* (o anotar sus valores) y asignarlo al control *Eye rays*.
6 Copiar este mismo color rojo original y asignarlo al control *Final gather*. Luego editarlo y manteniendo su tonalidad (o no, si se quiere experimentar), rebajar la saturación hasta un valor adecuado que habrá que decidir haciendo pruebas. En la figura (*b*) la saturación se ha rebajado de 1,0 a 0,2 para que se aprecie mejor la diferencia. En la figura (*c*) se ha dejado en 0,5 que parece un valor más adecuado.

Hay que tener en cuenta que este *shader* afectará a todos los rayos que se procesen en una escena. Precisamente por esta razón hay tantos tipos de rayos incluidos en este *shader*. En la mayoría de los casos esto no tendrá consecuencias. Pero si hay objetos reflectantes, transparentes o refractados, o si estamos utilizando fotones, habrá que tener esto en cuenta y copiar el color o el mapa de *Eye rays* a los otros componentes.

Por ejemplo, si la escena incluye rayos que atraviesan un medio transparente (o refractante o reflejante, etc.) habrá que añadir el mapa a los parámetros correspondientes del *mr Ray type switcher*. Todo lo que hay que hacer es arrastrar el mapa desde la casilla de uno a la de otro escogiendo la opción *Copy*.

Modificación de reflejos anisotrópicos por mapas

Ya hemos visto en el capítulo 2 las características generales de los reflejos anisotrópicos y en el capítulo 4, las técnicas básicas de simulación de este efecto. En este apartado ampliaré esa información para incluir el uso de mapas.

El metal bruñido es un caso frecuente de reflejos anisotrópicos. La anisotropía es debida al tipo de acabado que crea pequeñas hendiduras, no perceptibles directamente, que hacen que el reflejo se disperse en una dirección.

Esto puede simularse con relativa facilidad creando un mapa de relieve alargado. Un modo aún más sencillo de hacerlo, aunque no quede tan perfecto, es utilizar un mapa de ruido con las coordenadas UV alteradas para que el ruido se deforme en una dirección. Pero esto crea excesivas irregularidades, debido a las limitaciones de los mapas de tipo *Bump* y se obtendrán mejores resultados si se asigna este mapa a la reflexión en lugar de a *Bump*. Para comprobarlo, hacer lo siguiente:

1 Preparar una escena como la de la figura 5.36: una base sencilla a la que se ha aplicado un material con un mapa *checker* de colores blanco y rojo oscuro y una plancha metálica representada por un prisma cuadrado de poca altura (de 120 x 120 x 1 cm en el ejemplo). Crear también una cámara que apunte hacia la plancha y una luz directa situada en la dirección opuesta a la cámara. Para controlar mejor el modo en que el reflejo afecta a la plancha, crear una *sky light* que ilumine el conjunto de la escena y desactivar la contribución de la luz directa a la iluminación (BDR / *Affect diffuse*, desactivado).
2 Si se utilizan primitivas del programa (*box*), el prisma ya contará con coordenadas pro-

pias adecuadas. Si no fuera así, asignar un mapa *UVW* a la cara superior. Es importante que esta cara sea independiente de las demás para que el reflejo anisotrópico funcione adecuadamente.

3 Crear un material Arch&Design con las características siguientes:

Diffuse: gris claro (rgb 0,8,0,8,0,8).
Reflection:
0,85/0,75/16/metal (reflectivity/glossiness/samples/metal).
Color: mapa 1.
mapa1: Noise. Tiling UVW: 1,100,100.
Size 0.5. Colores gris (rgb 135,135,135) y blanco.
Anisotropy: 0,05

Con esta configuración se obtiene el resultado de la primera imagen de la figura 5.36.

Para mejorar este resultado habría que crear un mapa geométricamente más preciso, pues la aletoriedad del mapa *Noise* hace difícil conseguir el resultado que queramos. Pero si en lugar de asignarlo a *Bump*, que sería lo correcto desde un punto de vista físico, lo asignamos como mapa a *reflection*, como hemos hecho el resultado mejorará con menos coste de computación aunque sea algo menos realista. Puede comprobarse la diferencia llevando el mapa a *Bump* en lugar de a *Reflection*.

Otra posibilidad es asignar al mapa de reflexión un mapa distorsionado de modo similar a lo que ocurriría con un reflejo anisotrópico. El mapa de la segunda imagen de esta misma figura, se ha manipulado en Photoshop por medio de un filtro *Motion blur* . Aplicando este mapa en lugar del anterior, en el grupo *Reflection*, y aplicando en este caso el mapa *Noise* a *Bump* se obtendrá un resultado como el de la imagen de la figura citada.

En el caso de objetos que presenten anisotropía circular, como un disco, el procedimiento es algo más complejo. Necesitamos algún modo de redirigir el desplazamiento del realce especular. Esto puede conseguirse por medio de un mapa procedural, un degradado que permita crear un gradiente con la dirección que nos interesa. Para conseguir un resultado como el de la figura seguir los pasos siguientes.

1 Preparar una escena con un disco (en el ejemplo, un cilindro de 12 cm de radio y 0,2 de altura y 64 segmentos para los lados), una cámara y una luz mr Area Spot con un tamaño de unos 8 cm de ancho

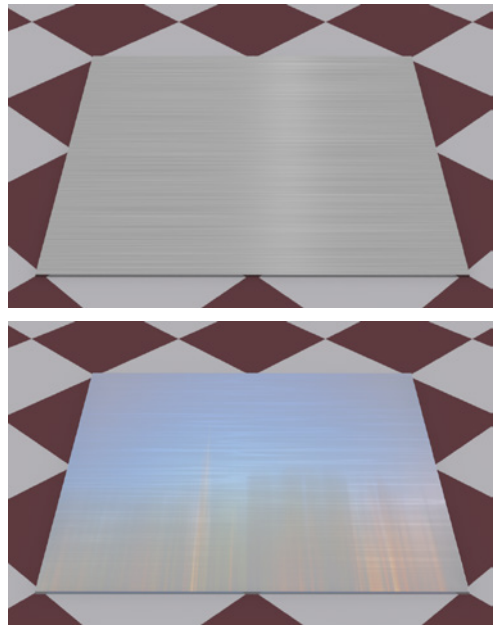


Figura 5.36 Reflejos anisotrópicos modificados por mapa: a) mapa de ruido aplicado al canal reflection*, b) mapa de ruido aplicado al canal de relieve (*bump*) y mapa de fondo distorsionado con* motion blur *aplicado al canal* reflection.

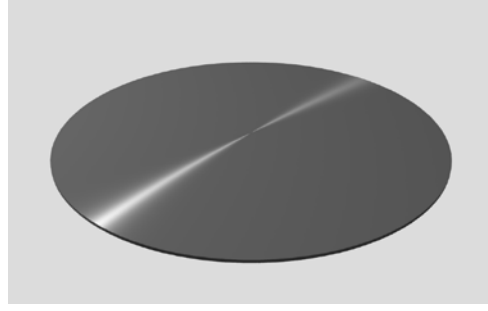


Figura 5.37 Reflejos anisotrópicos modificados por mapa. Reflejos sobre un disco metálico.

y que apunte en dirección contraria a la cámara.

2 Asignar al disco un *UVW Map* plano. Si el disco fuera la parte superior de un cilindro habría que crear una doble proyección y un material multisubobjeto. También es importante que la cara a la que se aplica el procedimiento que sigue sea independiente. Si el borde estuviera asignado a un *Smoothing group* que integrase las caras laterales se producirían resultados extraños.

3 Crear un nuevo material, de tipo *Arch&Design* y configurarlo como sigue:

Figura 5.38 Reflejos anisotrópicos modificados por mapa. Charcos en una calle.

Diffuse: 1,0, color gris medio
Reflection:
0,75, 0,85, 16 (reflectivity, gloss, samples),
activar “metal” para un efecto más atenuado.
BRDF aumentado a 0,35, en el ejemplo, para 0º
(esto intensifica los reflejos por lo que habría
que ponderarlo según lo que interese)
Anisotropy: 0,01, Rotation: mapa 1.
mapa 1 Gradient Ramp de negro a blanco. Tipo Spiral.

También pueden utilizarse recurso similares a los descritos en este apartado para simular charcos sobre una calle, como en el ejemplo de la figura 5.38 que se ha elaborado con un mapa de manchas blancas irregulares sobre un fondo negro, superpuesto al mapa con la textura de la calle.

5.6 Mapas de modificación de transparencias

Procedimientos básicos

Los mapas de transparencia y refracción funcionan de modo similar a los de reflexión. Al igual que en estos, y en otros casos similares de aplicación de mapas, la transparencia se puede modificar por medio de mapas en blanco y negro que hacen que las zonas blancas retengan la propiedad correspondiente, la transparencia en este caso, y las negras lo anulen. Y si hay zonas grises la propiedad se aplica de modo parcial.

No hay diferencia entre aplicar un mapa de transparencia o un mapa de recorte en la gran mayoría de los casos. La única diferencia importante es que si aplicamos un mapa al canal de transparencia podemos jugar con los parámetros propios de esta propiedad, que es lo que haremos en los ejemplos que siguen. Pero si todo lo que se quiere hacer es simular un objeto que es perfectamente transparente en una zonas y opaco en otras lo más sencillo es aplicar un mapa de recorte siguiendo los procedimientos ya vistos.

Tampoco hay gran diferencia, en algunos casos, entre simular efectos tales como vidrios grabados por medio de mapas de relieve (*Bump*) o por medio de mapas de transparencia si el efecto afecta principalmente a la alteración de algunas zonas de la superficie.

Modificación de transparencias por mapas monocromáticos. Ejemplos

La figura 5.39 muestra ejemplos del uso básico de este procedimiento. Hay que recordar que las partes blancas mantienen la transparencia especificada con el parámetro transparencia y las negras la anulan. Esto puede llevar a confusión pues los canales alfa funcionan al revés. La lógica general que hay que tener presente es que un mapa aplicado a una propiedad siempre funciona de este modo: lo blanco retiene la propiedad y lo negro la anula.

La primera figura de este grupo muestra un ejemplo de simulación de persianas venecianas por este procedimiento. En lugar de un mapa de bits he utilizado mapas procedurales pues el uso de mapa de bits resultará obvio si se asimila este procedimiento que puede resultar algo mas complejo técnicamente aunque es más sencillo de poner en práctica una vez que se comprende el sentido de la técnica.

El esquema adjunto a las tres imágenes de la figura 5.39 que ilustran este caso resume el procedimiento. He utilizado un material *Arch&Design* con un mapa de gradiente aplicado al parámetro de transparencia. El mapa de gradiente se ha girado 90º para que actúe de arriba abajo, se ha configurado como se muestra en la figura, con dos zonas blancas y negras claramente diferenciadas y un degradado suave y rápido en uno de los extremos para que el borde más luminoso quede atenuado, y se ha repetido 24 veces. Es decir, en coordenadas de mapa se han hecho los cambios siguientes: *U Tiling*: 25,0, *V Tiling*: 0,0, *W Angle*: 90º.

El efecto es que aparecerán franjas verticales negras (opacas) y blancas (transparentes).

Las partes opacas retendrán el color del material. Pero como la persiana está a contraluz este color resultará casi negro, lo que no

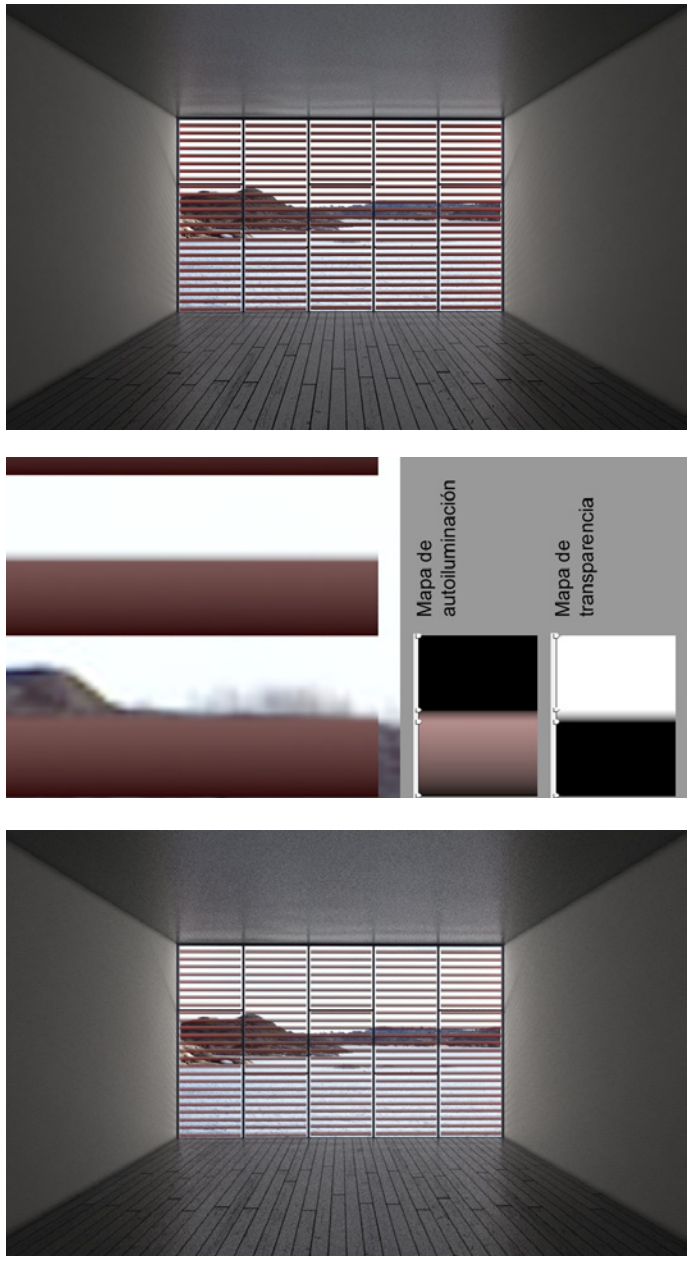


Figura 5.39 Transparencias por mapa. Simulación de persianas venecianas: a*) Lamas opacas generadas con una combinación de dos mapas procedurales, uno para la transparencia y otro para el color local (con autoiluminación);* b*) Esquema de los mapas procedurales utilizados: a la izquierda un detalle del resultado y a la derecha el gradiente vertical utilizado en cada caso y repetido 24 veces;* c*) Lamas semiopacas generadas con una variante del ejemplo anterior cambiando el negro del mapa de transparencia por un gris oscuro.*

se corresponde con el efecto de translucidez simple que se daría en este caso. Por esto, en lugar de asignar un color al parámetro *Diffuse* lo he asignado al filtro de *Self illumination* (que tiene un valor de luminancia acorde con la iluminación de la escena). Y para que los dos mapas se correspondan, he copiado el mapa de transparencia, lo he pegado en la entrada del filtro de autoiluminación y lo he configurado de tal modo que las partes negras (no autoiluminantes y, por tanto, no visibles) correspondan a las zonas blancas del mapa de transparencia (transparentes) y las partes de color (visibles) correspondan a las partes zonas negras (opacas) del mapa de transparencia.

La tercera imagen de esta figura muestra una variante de la anterior haciendo que las zonas negras del mapa de autoiluminación sean grises oscuras en lugar de blancas. Esto hace que sean ligeramente transparentes lo que mejora el resultado pues es más real e introduce cierto grado de variación.

El efecto se controla mejor y puede ser de mayor calidad utilizando un mapa de bits. Pero en muchos casos la diferencia de calidad será inapreciable y los mapas procedurales simplifican considerablemente la gestión y resultan particularmente adecuados para casos como este.

Para simular vidrio con la transparencia alterada por tratamientos diversos, como ocurre, por ejemplo, en el caso de vidrio esmerilado, la solución más simple, que ya hemos visto, es disminuir el parámetro *glossiness* que dispersa la transparencia, tanto más cuanto más bajo es el valor de este parámetro. Esto funciona bien para vistas relativamente lejanas. Pero si la vista es cercana y el detalle del vidrio esmerilado (o del tipo que fuese) resultara visible, este recurso no es suficiente y hay que recurrir a mapas.

Se puede generar un mapa que simule vidrio esmerilado utilizando filtros especiales de Photoshop o Gimp.

En el ejemplo que sigue he utilizado un mapa generado en Photoshop con el procedimiento siguiente:

1 Crear un archivo de dimensiones no muy grandes (luego lo ampliaremos), por ejemplo 512 x 512.
2 Escoger como colores de frente y fondo dos grises cercanos. En el ejemplo he utilizado dos grises hsb 0,0,60 y 0,0,40.
3 Aplicar un filtro de tipo “nubes” para que se creen patrones aleatorios suaves.
4 Aplicar a este resultado un filtro de tipo “Distorsión/Cristal”. En el panel que se abrirá, descolgar la lista junto a Textura y escoger “Esmerilado”. Otras opciones que también merecerá la pena probar son “Cubos” (para simular pavés) y “lente pequeña” (para simular vidrio grabado de modo regular). Ajustar los parámetros de distorsión y suavizado hasta que el resultado sea adecuado a lo que interese conseguir.
5 Aumentar el contraste del resultado. Abrir niveles (Ctrl+L o menú Imagen/Ajustes) y llevar los extremos del histograma hacia el

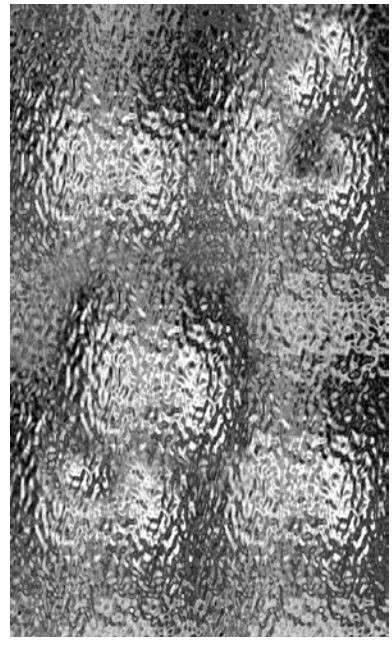


Figura 5.40 Transparencias por mapa. Vidrio esmerilado. Mapa utilizado en las figuras que siguen.

centro. En el ejemplo he llevado el extremo izquierdo (negros, 0) a unos 120 y el extremo derecho (blancos, 255) a unos 140.

6 Guardar el archivo, que servirá como base para otros ejemplos similares. Luego volver a guardarlo con otro nombre y aumentar el tamaño de la imagen y su proporción para que coincida con los paneles que queremos simular.

7 Copiar partes de los patrones a otras posiciones para evitar repeticiones. El resultado debería ser similar al de la imagen inferior de la figura 5.41.

Podrían añadirse más ejemplos pero el uso de mapas para simular transparencias no es demasiado frecuente y los dos ejemplos incluidos creo que cubren la mayoría de los casos de interés que se pueden presentar en la práctica.

Figura 5.41 Transparencias por mapa: a*) vidrios planos,* b*) vidrio esmerilado.*

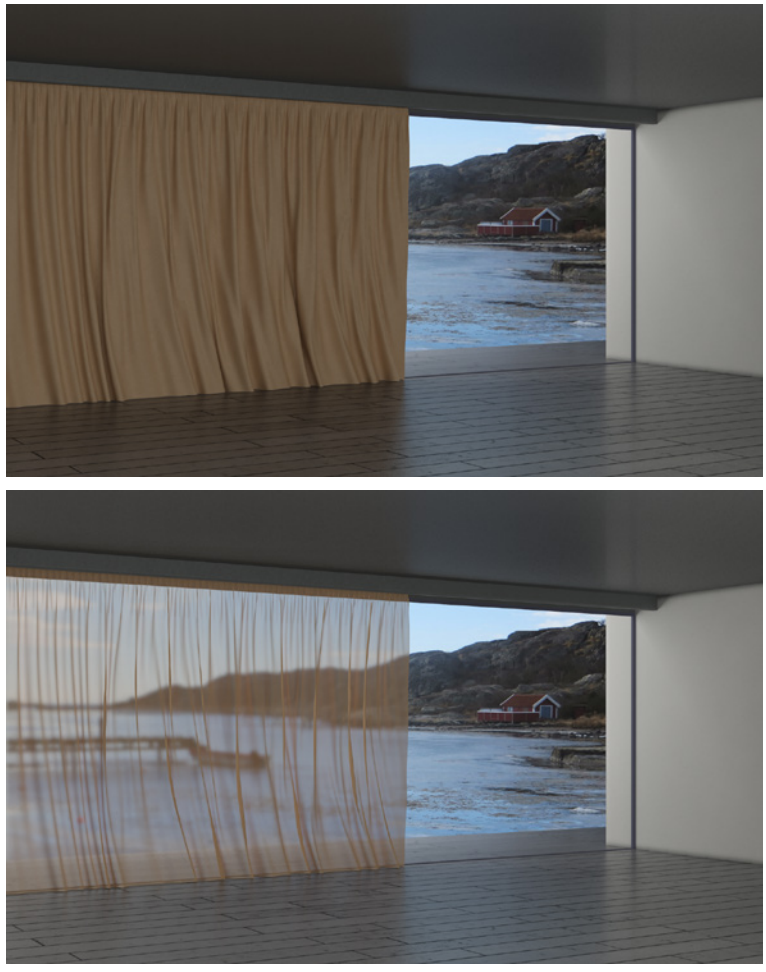


Figura 5.42 a) *Cortina opaca;* b) *Cortina con transparencia parcial y mapa* FallOff *aplicado a transparencia.*

Simulación de cortinas semitransparentes con mapas *Falloff*

En el caso de objetos que presentan pliegues que alteran la transparencia, como ocurre con las cortinas, el método más eficaz de simular este efecto es utilizar un mapa *Falloff* ligado a la transparencia. Al usar este mapa en el modo predeterminado, las partes que queden orientadas frente a la cámara recibirán un color blanco (máxima transparencia), las partes que queden orientadas en paralelo recibirán color negro (mínima transparencia), y las partes que queden orientadas en direcciones interemedias recibirán colores grises (diferentes grados de transparencia).

La figura 5.42 muestra un ejemplo de aplicación de este efecto. Conviene recordar que la parte más difícil de simular cortinas corresponde al modelado. Intentar modelar los pliegues manualmente es una tarea larga, pesada y condenada, en el 95 % de los casos, al fracaso. Cualquier procedimiento de modelado que intente reproducir un proceso

físico debe plantearse reproducir el propio proceso, no el resultado. Afortunadamente, en muchos programas, entre ellos 3ds Max, existen herramientas muy potentes de simulación de procesos físicos. Y, por tanto, quien quiera simular una cortina similar a la del ejemplo adjunto deberá hacer dos cosas. Primero, dedicar un par de horas a entender el funcionamiento del modificador *cloth* de 3ds Max (u otro programa similar). Segundo, leer el resumen que sigue que es necesariamente telegráfico pues este libro trata de materiales, no de modelado, y que se entenderá mejor cuando se cuente con esta base mínima.

Las cortinas del ejemplo se han modelado del siguiente modo: *a*) He creado un plano de dimensiones iguales a toda la abertura (unos 12,5 x 3,6 m) con un gran número de segmentos (unos 400 x 100) y con subdivisiones dobles en los 20 cm superiores; *b*) He creado un objeto auxiliar, una barra de unos 6 cm de diámetro, situada un poco por debajo del borde superior. Esta barra se ha animado para que al comienzo de la simulación tenga la misma anchura que el hueco (unos 12,5 m) y al final una dimensión menor (unos 8 m); *c*) He añadido al plano un modificador *cloth*, he añadido la barra como objeto de colisión y he creado un grupo con vértices alternados ligados a la barra. De este modo, al activar la simulación, el plano se desplaza por efecto de la gravedad mientras que los vértices superiores se encogen a medida que la barra cambia de escala y se mantienen en su posición. Es decir, básicamente lo mismo que ocurriría si corriésemos una cortina real. Una vez completada la simulación, se escoge el momento más adecuado, desplazando el *time slider* y se convierte el objeto a malla editable. Luego puede editarse el resultado, borrando algunos pliegues o deformando el conjunto, si interesa.

Al objeto así creado le he asignado un material con la siguiente configuración: tipo *Arch&Design*. *Diffuse*: 1,0, color ocre. *Refraction*: 0,4, 0,95, 16 (*transparency*, *glossiness*, *samples*), color por mapa. Mapa a transparencia: *Falloff* con valores predeterminados (tipo: *perpendicular/parallel*, dirección: *camera z-axis*, colores *front*/*side*: blanco y negro)

Simulación de agua con adición de mapas

En el capítulo anterior hemos visto métodos básicos de simulación de agua con métodos de control de la refracción y el color de la transparencia en profundidad. Si a estos métodos le añadimos mapas de relieve el resultado mejora considerablemente pues la superficie del agua raras veces es lisa sino que, por el contrario, presenta ondulaciones más o menos amplias. Estas ondulaciones se traducen en distorsiones características del reflejo y la transparencia que requieren métodos adicionales de simulación.

Un método sencillo de incorporar estos métodos es generar un mapa con transiciones suaves de franjas blancas y negras que se aplique como mapa de tipo *Bump*. Pero como la simulación de agua es frecuente en muchos contextos, hay *shaders* que facilitan el trabajo. Uno de estos *shaders* es el mapa *Ocean* de mental ray que incorpora varios parámetros para conseguir lo mismo pero sin necesidad de rehacer el mapa. Es decir que es como si tuviéramos un mapa con transiciones más o menos suaves de franjas blancas y negras pero del que podemos variar todas sus características con facilidad.

Este mapa tiene una serie de parámetros básicos y otros complementarios que se utilizan para animación y que pasaré por alto. Los parámetros básicos (doy entre paréntesis los valores predeterminados) son: *Largest*, máximo tamaño de las olas (100); *Smallest*, mínimo tamaño de las olas (1,0); *Quantity*, pasos entre las olas más grandes y las más pequeñas (10); *Steepness*, controla si las olas son suaves (valores pequeños) o abruptas (valores grandes) (5;0).

Para ilustrar su utilización he incluido dos ejemplos sencillos. El primero representa un río con dos fondos superpuestos, una imagen de una ciudad y una imagen de un árbol, asignados a dos planos orientados hacia la

Figura 5.43 Simulación de un río con un shader *de tipo* Ocean *con los diferentes valores para los parámetros básicos que se indican en el texto.*

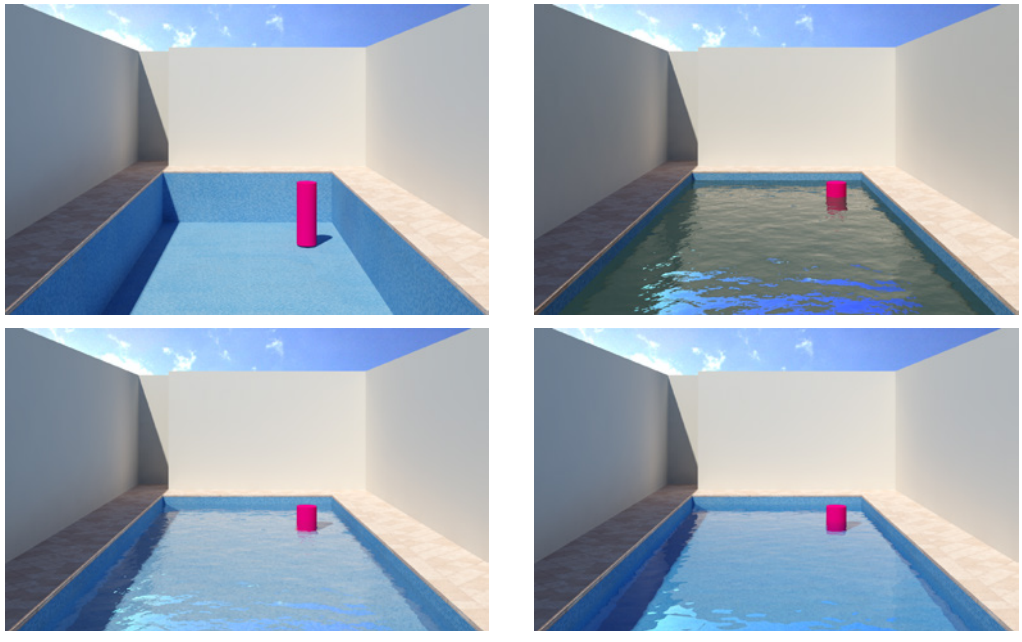


Figura 5.44 Simulación de un piscina con un shader *de tipo* Ocean *con los diferentes valores para los parámetros básicos que se indican en el texto.*

cámara y un borde modelado con una *spline* extruida. El plano del agua es un plano sin espesor. Las dos imágenes de la ciudad y el árbol incluyen canales alfa para que sea visible el plano del río en la parte inferior.

Las figuras correspondientes a este ejemplo muestran diferentes resultados, a partir de asignar al plano del río un material *Arch&Design* en que se ha aplicado a *Bump* (con una intensidad de 0,1), un mapa de tipo *Ocean* (*lume*) con los siguientes valores para *Largest*, *Smallest*, *Quantity* y *Steepness* (unidades en cm): *a*) 20, 10, 2, 0,2; *b*) 20, 1, 5, 1; *c*) 20, 1, 5, 5; *d*) 150, 15, 5, 1,0. Los tres primeros introducen pequeñas variaciones a partir de un tamaño máximo de olas igual. El cuarto aumenta considerablemente este tamaño.

El segundo ejemplo representa una piscina con un cilindro de color rojo en su interior para que se aprecien mejor los efectos de transparencia. La primera imagen de la figura 5.44 muestra la piscina sin agua. En las siguientes he añadido un plano al que he asignado un material *Arch&Design* con un mapa de tipo *Ocean* (*lume*) en *Bump* (con una intensidad de 0,1) con los valores siguientes: 20, 2, 4, 0,8 (*Largest*, *Smallest*, *Quantity*, *Steepness*) para la segunda y la tercera. La segunda no tiene transparencia. La tercera tiene un valor de transparencia de 0,9. La cuarta tiene transparencia y, por añadidura, activadas las opciones de color a máxima distancia (sección *Advanced rendering options/ Refraction* de *Arch&Design*) y se han modificado los valores de *Ocean* del modo siguiente: 16, 8, 2, 0,5).

5.7 Mapas de aplicación a la autoiluminación

Procedimientos básicos

Como ya he dicho en el apartado correspondiente a *shaders* arquitectónicos, autoiluminación (sin mapas), no es posible tratar adecuadamente este tema sin adentrarse en técnicas de iluminación que he preferido tratar por separado en otro libro. En el caso de los mapas, esto vuelve a ocurrir si bien en este caso merecerá la pena que me extienda un poco más debido a la importancia de las texturas y del modo en que se relacionan con esta propiedad.

La aplicación de mapas a la propiedad de autoiluminación es sencilla y muy similar a la de los apartados anteriores. Como en estos casos, si se asigna un mapa en blanco y negro a la propiedad de autoiluminación, las partes blancas retendrán la propiedad y, por tanto, tendrán autoiluminación, mientras que las partes negras retendrán el color (o la textura) propio del material.

Para ilustrarlo he desarrollado cuatro ejemplos sencillos que abordan diversas posibilidades del uso de mapas con autoiluminación.

Ejemplo 1. Cubo

El primer ejemplo es muy simple pero servirá para introducir las ideas básicas. Se trata de aplicar un mapa de autoiluminación a un cubo (con las aristas achaflanadas) situado sobre un plano. El prisma y el plano están iluminados por dos luces fotométricas de área, una de 1.200 cd por la izquierda y otra de 600 cd por la derecha. Pueden iluminarse de un modo más simple, lo suficiente para que el plano reciba iluminación. En la figura 5.45 también se ha utilizado un cálculo de *Final Gather* y un control de exposición mr de 8,0 EV (debido al uso de luces fotométricas), pero también se puede prescindir de este cálculo.

El cubo tiene asignado un material de tipo *Arch&Design* con un color *diffuse* rojo oscuro. A partir de aquí, lo que habría que hacer es desplazarse a la sección de *Self illumination* (*Glow*), activar la opción del mismo nombre, dar a la intensidad de autoiluminación un valor de unas 500 cd y pulsar el botón junto a *Filter* para asignar un mapa.

El mapa utilizado es simplemente un mapa procedural de tipo *Tiles*, con las juntas de color negro y la parte principal de color blanco. Al asignar este mapa al objeto las partes blancas serán autoiluminantes y las negras no lo

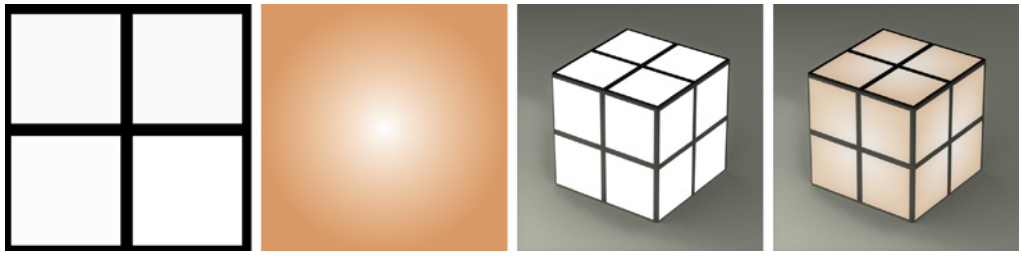


Figura 5.45 Mapas de autoiluminación. Ejemplo 1: a) *Mapa procedural que genera un patrón de juntas;* b) *Mapa procedural que genera un gradiente circular;* c) *Cubo autoiluminante con el primer mapa aplicado;* d) *Cubo autoiluminante con los dos mapas aplicados en combinación.*

serán, con lo que el color resultante para estas zonas será el del material (rojo oscuro en el ejemplo).

Se puede hacer más interesante el ejemplo si combinamos el mapa anterior con un degradado radial que vaya de blanco a un color cálido para simular el efecto de una luminaria interior que afectaría más a las partes centrales que a las periféricas. Para ello hay varios métodos, quizás el más simple es utilizar un mapa *Composite* con el mapa de degradado como textura del material y el mapa anterior como máscara.

Las imágenes de la figura 5.45 muestran los resultados que se obtienen con este procedimiento.

Ejemplo 2. Prisma

Este ejemplo muestra cómo asignar un material autoiluminante y con texturas a un prisma. Puede adaptarse a la simulación de pantallas o de cualquier otro objeto similar. La escena consiste en un prisma de 60 x 60 x 45 cm sobre un plano. El prisma está iluminado por dos luces fotométricas iguales a las del ejemplo anterior.

1 Asignar al objeto un material *Arch&Design* de color gris claro. El resultado de esta asignación simple se muestra en la primera imagen de la figura 5.46 (*a*).
2 Modificar el material activando la sección *SelfIllum* y activando la opción “illuminates the scene”. Ajustar la intensidad para que sea visible la contribución a la iluminación del plano de base. Así se obtiene la segunda imagen (*b*).
3 Desactivar *SelfIllum*. Modificar el material asignando una textura al parámetro *Diffuse*. Así se obtiene un resultado como el de la tercera imagen (*c*).

Ahora se trata de lograr que la textura que asignemos al material sea visible pero manteniendo la intensidad global de este último material. Para ello habrá que compensar con una mayor intensidad de las partes más iluminantes (claras) de la textura, la reducción de las partes menos iluminantes (oscuras). Esto variará según las características de la textura aplicada.

4 Eliminar el mapa anterior (cortar y pegar) y colocarlo como parámetro de *SelfIllum* (*glow*) asignándolo al color del filtro. De este modo las partes claras serán más luminosas que las oscuras y la textura parecerá emitir luz.

Hacer un *render* de prueba con esta última configuración: el resultado será demasiado luminoso y la textura quedará desvaída. Para que sea más visible habrá que oscurecerla y aumentar el contraste. Esto puede hacerse editando la textura con Photoshop pero también se puede recurrir a un mapa de corrección de color incluido en Max.

5 Editar el mapa y presionar el botón “bitmap”. Seleccionar el mapa *Correction*

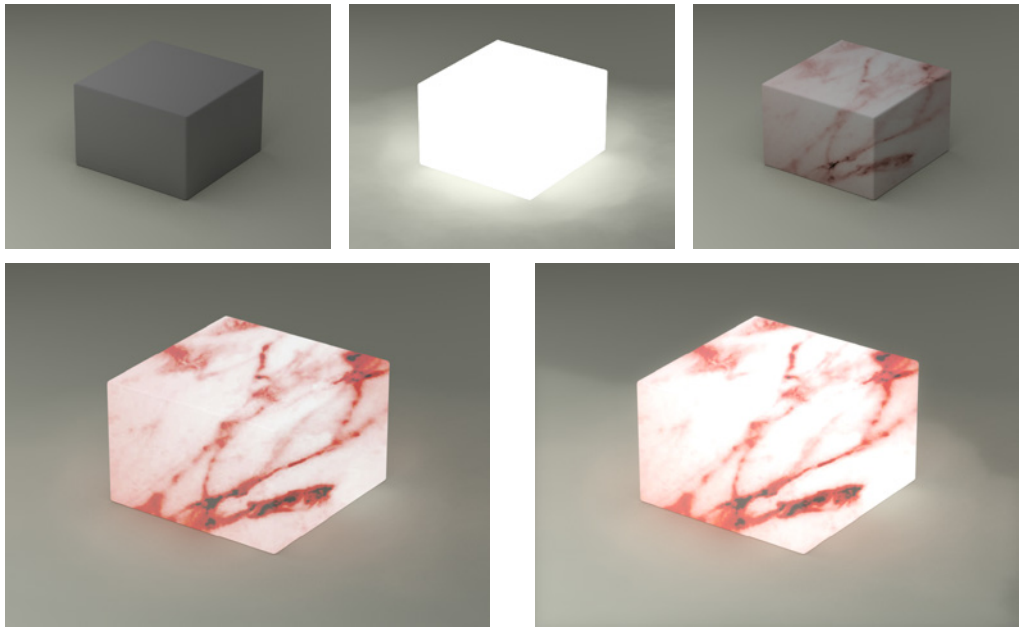


Figura 5.46 Mapas de autoiluminación. Ejemplo 2. Prisma irradiante con textura: a) *Asignación de un material con color mate;* b) *Material con autoiluminación simple;* c) *Material sin autoiluminación y con textura;* d) *Material con autoiluminación y textura;* e) *Adición de resplandor (*glare*).*

color. Responder "Sí" a la pregunta de si se quiere conservar el mapa previo como submapa. Luego modificar la luminosidad y el contraste hasta que el resultado sea adecuado. La cuarta imagen de la figura 5.46 (*d*) se ha obtenido con un valor de *brightness*/*contrast* de -68,0/+12,0.

6 Por último, se puede añadir un efecto de *glare* con un valor bajo para aumentar el resplandor pero sin afectar demasiado a la textura. Para ello, todo lo que hay que hacer es, desde *Render setup / Renderer/Camera effects,* en el grupo *Camera shaders*, activar "Output" y asignarle un mapa de tipo *Glare*. Luego llevar este mapa al editor de materiales para ajustar sus parámetros. La última imagen de la figura 5.46 (*e*) se ha obtenido con un valor de mapa *Glare* de 4/0,6 (*quality* / *spread*).

Ejemplo 3. Plancha metálica

El tercer ejemplo es similar al anterior pero con algunas variantes de interés.

La escena incluye un interior con texturas corrientes y un panel irradiante al fondo. Para simular la iluminación del panel se ha utilizado una luz *mr Area Spot*, situada ligeramente por delante del panel, con unas dimensiones aproximadamente iguales a las de este panel, una abertura muy amplia (133º/155º *Hotspot* / *Falloff*) y una intensidad de 1,0. El cálculo es *FG medium* con 2 rebotes para que el muro de fondo y los planos de los pilares paralelos al muro de fondo no quedan completamente negros (si se quisiera que quedasen más claros se podría hacer la textura ligeramente autoiluminante).

El panel irradiante tiene asignado un material *Arch&Design* con *Self Illumination* activado, un mapa que simula una plancha metálica

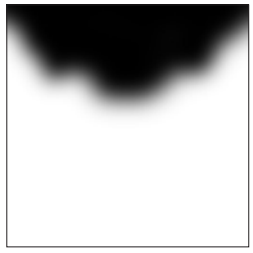
de textura irregular aplicado a *Filter*, una *Luminance* de 400 cd/m^2 y la opción "Illuminates the scene when using FG" activada aunque su efecto será bastante leve.

El interés de este caso es que es necesario ajustar con cuidado la iluminación para que el efecto sea convincente. Como en el caso anterior y en el caso que sigue, este es un ejemplo que está en el límite de los temas principales que se tratan en este libro, pues aunque el sistema de iluminación sea relativamente simple requiere cierto grado de práctica con la disposición de las luces.

Ejemplo 4. Vela

Para simular una vela, lo primero que se requiere es modelar la vela con el programa de modelado favorito de quien esté siguiendo este ejemplo. Pueden obtenerse resultados más interesantes con programas especiales que permiten esculpir objetos orgánicos, como *Sculptris* (gratuito) o *ZBrush* (similar al anterior pero más complejo y no gratuito). En el ejemplo adjunto me he limitado a utilizar recursos elementales con 3ds Max, suavizados con *Turbo smooth* para el cuerpo de la vela. La mecha se simula fácilmente con una *spline* curva a la que se le puede dar grosor simplemente activando la opción "renderable". Y la llama con una *spline* que dibuja la curva, un modificador *Lathe* para convertir este perfil en un cuerpo de revolución y un modificador *Noise* para dar alguna complejidad al resultado (este modificador también se puede animar si interesa). La primera imagen de la figura 5.48 muestra el resultado con los materiales básicos que se describen a continuación. A partir de aquí continuar como sigue.

Figura 5.47 Mapas de autoiluminación. Ejemplo 3. Panel irradiante.



1 Asignar al cuerpo de la vela un material con la configuración siguiente:

material: “cuerpoVela” (Standard)
Diffuse > mapa 1 (Grad Ramp), Spec level 100/10
mapa 1 > Coords/Rotation UVW: 0,0,-90.
Tipo Lineal. Colores:
negro rojizo en 1,
naranja en 50,
amarillo anaranjado en 100

2 Asignar al cuerpo de la vela un modificador *UVW Map* de tipo cilíndrico ajustado al envolvente general.
3 Crear una luz sencilla a un lado para que ilumine el cuerpo de la vela.
4 Asignar a la mecha un material mate negro.
5 En Photoshop o Gimp crear un mapa sencillo como el de la segunda imagen de la figura 5.48 que usaremos para hacer la punta de la llama más transparente.
6 Asignar a la llama un material con la configuración siguiente:

material: “llama” (Standard)
Diffuse > mapa 1 (Grad ramp),
Spec level 100/10,
SelfIllum 100.
Opacity > mapa 2 (Gradient ramp)
mapa 1 > Coords/Rotation UVW: 0,0,-90.
Tipo Lineal. Colores:
rojo azulado en 1,
naranja en 50,
amarillo en 100
mapa 2 > Coords/Rotation UVW: 0,0,-90.
Tipo Lineal. Colores:
rojo azulado en 1,
naranja en 50,
amarillo en 100

La imagen (*c*) de la figura 5.48 muestra el resultado de añadir este mapa.

7 Para completar la simulación, añadir un efecto de resplandor. No voy a entrar en los detalles que se pueden encontrar en la ayuda del programa. Básicamente, lo que hay que hacer es asignar a la llama un

Figura 5.48 Autoiluminación con mapas. Ejemplo 4. Vela: a) *Modelo inicial con materiales básicos;* b) *Mapa de opacidad;* c) *Resultado con el mapa de opacidad añadido;* d) *Resultado con un efecto de resplandor añadido.*

identificador (*Object properties* / *G-Buffer* / *Object ID*), luego entrar en *Video post* (menú *Rendering*), crear un *Scene event* ligado a una cámara, añadir un *Image filter event* y escoger un tipo *Lens effects glow* y entrar en *Setup* para configurar las características del efecto ligándolas al identificador escogido para la llama. Una vez configurado volver al panel principal y pulsar el botón *Execute sequence* para que se represente la escena y se superponga al resultado el efecto escogido.

Y si se quiere que ilumine los objetos adyacentes, colocar una luz omni en el centro de la llama, que excluya la llama y la mecha, y editar los parámetros de la luz para hacer que se atenúe en el rango que interese en torno a la vela.

5.8 Mapas con sistemas de partículas

Introducción

En el capítulo anterior he resumido los métodos para crear sistemas de partículas, ilustrados con algunos ejemplos básicos. En este apartado doy por conocidos estos métodos básicos y ampliaré la revisión de estas técnicas para incluir el uso de mapas.

El uso de mapas con sistemas de partículas amplía considerablemente las posibilidades de simulación de un gran número de objetos que se distribuyen en el espacio o sobre una determinada superficie, así como la simulación de fenómenos tales como el fuego, humo, nubes, etc.

El recurso principal es el uso de *billboards*, un término ya introducido genéricamente en el capítulo 2 para designar imágenes con canales alfa que se aplican a un plano orientado hacia la cámara y se perciben como siluetas de objetos y cuyas técnicas de aplicación hemos visto también en el apartado sobre recortes de este capítulo. Si las partículas están constituidas por facetas simples dirigidas hacia la cámara, se puede asignar a estas facetas un material con un mapa de estas características, lo que permite simular objetos distintos pero, también, si usamos mapas con degradados difusos, objetos continuos con variaciones de color y textura.

Por otra parte, el uso de mapas nos permite ampliar las posibilidades de controlar la distribución de las partículas sobre una superficie como también he indicado en el capítulo anterior.

Todo esto se entenderá mejor por medio de ejemplos. Pero antes de pasar a los ejemplos resumo muy brevemente las técnicas disponibles para utilizar mapas con partículas.

Operadores de mapas

Ya hemos visto que, en 3ds Max, hay tres tipos de operadores de materiales, *Material static*, *Material dynamic* y *Material frequency*. Y ejemplos de uso del primero. En este apartado veremos algún ejemplo adicional que utiliza el segundo. El tercero tiene sentido principalmente en animación aunque, como ya he resumido en el apartado correspondiente, puede utilizarse también para controlar la frecuencia de distribución de los diferentes submateriales de un material multisubobjeto.

Para controlar los mapas hay dos operadores, el primero y principal es el *Mapping operator*. Este operador permite asignar las mismas proyecciones de coordenadas a cada partícula. Las coordenadas de mapa se pueden animar, lo que hace que las partículas cambien de color a lo largo del tiempo utilizando, por ejemplo, un mapa de gradiente. Este operador funciona con un operador *Material* definido para el evento activo. También es posible utilizar un mapa de bits, junto con un operador tipo *Dynamic* para asignar diferentes marcos de una secuencia de imágenes a las partículas basándose en la vida de las partículas (*particle age*) por medio del mapa complementario que describiré más adelante. El sentido de este operador se entenderá mejor con un ejemplo, por lo que me remito al que se da más abajo.

El otro operador es el *Mapping object operator*, que aplica proyecciones a las partículas tomando la información de las proyecciones de uno o más objetos de referencia. Para cada partícula, el operador encuentra el punto más cercano de la referencia y asigna este punto a la partícula. Incluye los siguientes parámetros principales: *a*) *Type*. La opción predeterminada es "Once" (adquiere al mapa al iniciarse el evento, a partir del objeto de referencia). La otra opción es "Continuous" (varía el punto de referencia en función de la posición de la partícula, lo que aumenta el tiempo de cálculo); *b*) *Mapping channels*. Permite asignar hasta 32 diferentes canales; *c*) *Mapping variation*. Proporciona controles para variar en porcentaje las proyecciones UVW; *d*) *Blends*. Proporciona controles para fusionar la proyección según la duración o la distancia.

Todo lo anterior está referido a los operadores que se encuentran en *Particle view* y que forman parte de este recurso general que, como hemos visto, facilita la gestión de los sistemas de partículas. En el editor de materiales, a la hora de asignar un mapa a un objeto siguiendo los procedimientos habituales, nos encontraremos, por añadidura, con dos tipos de mapas, diseñados específicamente para trabajar con sistemas de partículas: *Particle age* y *Particle mblur*.

Particle age es un mapa con el que se definen tres colores ligados a la vida de las partículas. El color 1 corresponde al color en el momento del nacimiento, 0 % de la duración de la secuencia, el color 2 a la mitad de esta duración, 50 % y el color 3 al final, 100 %. Estos porcentajes se pueden modificar, si interesa hacerlo. El mapa se utiliza con el material que se asignará a las partículas. Véase el ejemplo de simulación de "humo" para comprobar como se usa en la práctica este mapa.

Particle mblur se utiliza para alterar la opacidad de las partes finales del recorrido de una secuencia y simular así el efecto de movimiento. Los parámetros son muy simples: dos colores que se relacionan en principio con valores de opacidad, uno para velocidad baja (blanco, opaco) y otro para velocidad alta (negro, transparente). Y un parámetro de enfoque (*sharpness*, 2,0 por defecto) que controla la mayor o menor transparencia según la velocidad. Al igual que en el caso anterior, el mapa se utiliza en el material al que se asignan las partículas. Pero en la mayoría de los casos que nos pueden interesar es más sencillo utilizar el método general de *Motion blur* que he descrito en el primer apartado sobre partículas por lo que no lo incluiré en los ejemplos que siguen.

Ejemplo 1. Procedimiento básico

Para introducir las técnicas básicas se puede hacer el siguiente ejercicio. Tener en cuenta que necesitamos crear una forma instanciada pues las partículas propias de *Particle view* no reciben todo tipo de mapas.

1 Crear un sistema de partículas, un *PF source rectangular* de 50 x 50, situado en 0,0,0 y girado 180º para que las partículas emitan hacia arriba. Si no se recuerda el procedimiento revisar lo explicado en el capítulo anterior.
2 Crear una cámara, "camera1" dirigida hacia este sistema.
3 Entrar en *Particle view* y configurarlo así:
   Birth. Emit Start/Stop > 20/40. Amount 20.
   Position Icon, *Speed*. Dejar los valores predeterminados.
   Shape Facing. Look at camera > "camera1". Use parallel Direction. Units in world space > 10,0.
   Con esto se crearán partículas cuadradas de 10 x 10 giradas hacia la cámara.
   Display. Dejar los valores predeterminados.
4 Desplazar el *time slider* hasta el *frame* 30 y continuar avanzado *frame* a *frame*. Las 20 partículas nacerán en el *frame* 30 e irán ascendiendo hasta desaparecer hacia el *frame* 40 (dependiendo de la vista de cámara). Dejar el *slider* en un *frame* en el que las partículas queden claramente visibles (en el *frame* 37 en mi ejemplo). Hacer un *render* en esta posición para comprobar el resultado previo.

Figura 5.49 Partículas con mapas. Procedimientos básicos. Ejemplo 1. Resultado en un frame *intermedio.*

En paralelo, con Gimp o Photoshop, crear un mapa de bits que simule un globo como los de la figura 5.49 haciendo que el fondo sea transparente. Guardarlo en un formato que preserve el canal alfa (por ejemplo png). Hacer otras dos versiones del mismo con diferentes colores. Luego, en 3ds Max, crear un material multisubobjeto, "globos", de tres materiales. Asignar al primero un material de tipo *Arch&Design* con el mapa de bits en *Diffuse*. Copiar este mapa y pegarlo en la entrada de *CutOut* (sección *Special purpose maps*). Editarlo y, en *Bitmap parameters*, en *Mono channel output*, marcar la opción *Alpha*. Luego copiar dos veces este material pegándolo en los otros dos materiales y cambiar el mapa por las diferentes versiones de colores. Una vez que contemos con este material, continuar como sigue.

5 En *Particle View*, añadir un operador *Material static*. Asignarle el material que hemos creado, "globo", arrastrándolo desde el Editor de Materiales sobre el botón de *Assign Material*. Luego marcar la opción "Assign Material ID", escoger la opción "Cycle" (o "Random") y especificar, en "#Sub-Materials", el número de submateriales que queremos mostrar (3 en este ejemplo).
6 El resultado deberá ser como el de la figura 5.49.

Ejemplo 2. Uso del operador de mapeado

El siguiente ejemplo elemental ilustra el uso del *Mapping operator*.

1 Crear un material corriente (*Arch&Design* o *Standard*) y asignar a *Diffuse* un mapa de tipo *Gradient ramp*.
2 Editar el mapa, comprobar que el tipo es lineal y asignarle diferentes colores desde la posición 0 a la 100. En el ejemplo se ha asignado un color rojo a la posición 0, naranja a la 35, azul claro a la 65 y azul a la 100.

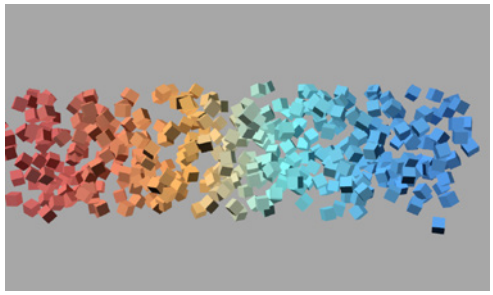


Figura 5.50 Partículas con mapas. Ejemplo 2. Uso del operador de mapeado.

3 Crear un *PF Source*. Abrir *Particle view*. En *Birth*, cambiar *Emit start* / *Stop* por el mismo rango que la animación, 0/100, y el número de partículas a unas 400. Cambiar la velocidad a unos 120 o el valor adecuado para que, al final, en el *frame* 100 se muestren todas. En *Shape*, escoger un cubo 3D (como en mi ejemplo) u otra forma claramente visible para comprobar el resultado.
4 Añadir un operador *Material static* y arrastrar el material que hemos creado sobe el botón de asignación.
5 Añadir un operador *Mapping*. En "Sync By", escoger la opción "Particle Age".
6 Mantener abierto *Particle view* y los parámetros de *Mapping* y atendiendo a los valores de *Map value* que en principio serán 0,0,0 (U,V,W). En la barra de animación activar *Autokey* y comprobar que el valor de U es 0,0 en el *frame* 0. Luego desplazarse hasta el frame 100 y cambiar el valor de U a 1,0. Desactivar *Autokey*. De este modo hemos creado una clave de animación que aplicará el mapa de modo diferente entre estos valores.
7 Comprobar que al desplazar la barra del tiempo los valores de U se desplazan entre 0.0 y 1.0. Y que, en consecuencia, los cubos irán adquiriendo todos los colores del mapa de gradiente. La figura 5.50 muestra el resultado hacia el final de la animación.

Ejemplo 3. Simulación de árboles con mapas de bits

Ya hemos visto, en el apartado sobre mapas de aplicación a recortes de este capítulo, que la idea básica del uso de *billboards* para simular figuras es utilizar mapas de bits con canales y asignar estos mapas, incluidos en un material, a un plano que queda siempre orientado hacia la cámara. También hemos visto que si queremos que la figura arroje sombras, esto se puede completar con un segundo plano, en la misma posición pero no visible y orientado hacia la luz. Y que, según los casos, también puede interesar hacer que el mapa de bits tenga un cierto grado de autoiluminación para que sea claramente visible aunque esté orientado contra la luz, para simular la irradiación atmosférica por la que quedaría iluminado.

Ligar *billboards* a sistemas de partículas es especialmente adecuado para escenas que incluyan múltiples objetos que si se simulasen con modelos 3D cargarían excesivamente la memoria. En el ejemplo siguiente se describe esta posibilidad.

Lo primero que necesitamos es tres o cuatro imágenes de árboles. Esto puede conseguirse por diversos medios, utilizando mapas descargados de internet, generados con un programa especializado como Onyx Tree o generados a partir de árboles 3D como también hemos visto en el apartado sobre mapas de recortes. Me remito a ese apartado y doy por supuesto que, sea como sea, ya contamos con estas imágenes.

A partir de ahí, contando con estos recursos, hacer lo siguiente.

1 Crear una escena que incluya un plano de unos 200 x 200 metros, al que denominaremos "base". Para simplificar la descripción trabajaremos con una superficie plana. El procedimiento será similar con una superficie más compleja (véase la última imagen de la figura 5.51). Añadir una luz simple a la escena y una cámara. Añadir

también una *skylight* de baja intensidad para que las sombras no queden demasiado oscuras. O bien crear un sistema de luz diurna en lugar de las dos luces anteriores.

2 Entrar en *Particle view*. Crear un *Standard flow*. Configurarlo como sigue (borrando o substituyendo los operadores para que solo queden los que se detallan):

Birth. Emit Start/Stop > 0/0. Amount > 12 (de momento crearemos tan solo una docena de árboles).
Position Object. Objecto > "base".
Shape facing. Look at camera > "camPrincipal". Units > 20,0 (o lo que convenga). Pivot at > Bottom.
Display. Type > Geometry.

Con esta configuración el resultado inicial será como el de la primera imagen de la figura 5.51.

3 Crear un material "arbol1" con un mapa de recorte siguiendo los procedimientos que ya hemos visto, es decir, con la siguiente configuración: Tipo *Arch&Design*. *Diffuse* > mapa "arbol.png. Copiar este mapa, pegarlo en *Cutout* (*special maps*), editarlo y, en parámetros del mapa de bits, cambiar las opciones de *Monochannel output* a "Alpha" y de *RGB Channel output* a "Alpha as gray".

4 Desde *Particle view* añadir un operador de tipo *Material static* y situarlo antes de *Shape facing*. Arrastrar el material "arbol1" que hemos creado sobre el botón junto a *Assign material*. Comprobar el resultado que deberá ser similar la de la tercera imagen de la figura 5.51.

Esto puede ser suficiente para muchos casos. Pero la imagen con que se ilustra el resultado, que muestra sombras, está obtenida con una luz situada algo a la izquierda de la cámara, con lo que el resultado es coherente con la orientación de las partículas. Pero si la luz estuviera situada en otra dirección, cerca de la perpendicular a la dirección de la cámara, lo que obtendríamos es una sombra muy estrecha o una mera línea pues el objeto que simula el árbol es un plano. Aunque el control de la iluminación no es el tema de este libro,

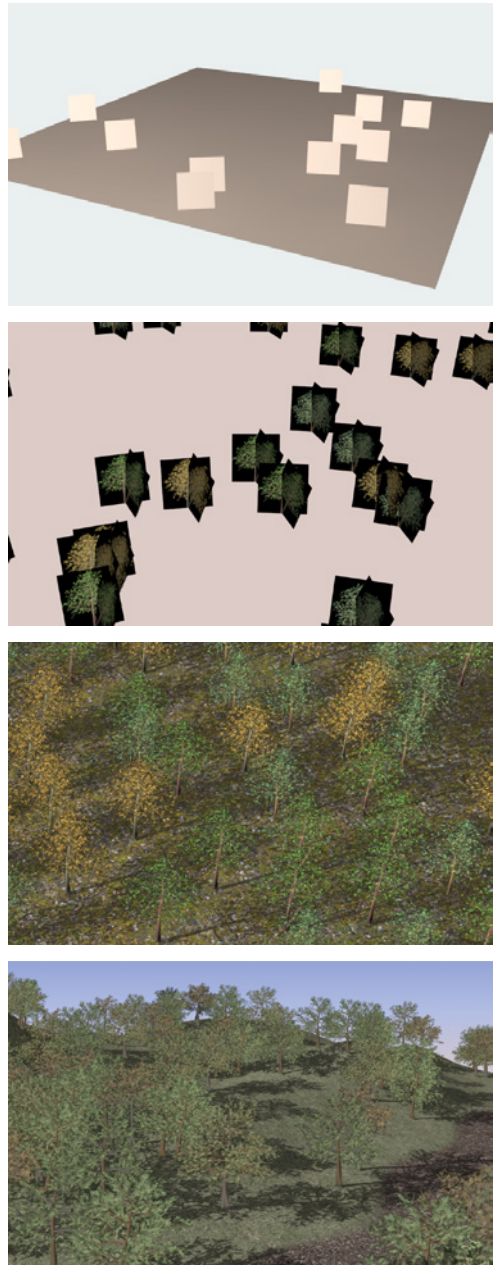


Figura 5.51 Partículas con mapas. ejemplo 3. Arboles: a) *Vista inicial, sin el mapa añadido;* b) *Vista con los mapas añadidos y las partículas orientadas hacia la cámara y las orientadas hacia la luz;* c) *Resultado final (detalle);* d) *Resultado con superficie irregular.*

para completar este ejemplo convendrá ilustrar el método corriente para solucionar este problema que consiste en crear un segundo sistema de partículas, basado en el anterior pero orientado hacia la luz y haciendo que las partículas no sean visibles en el render sino solo su sombra.

Para generar un segundo sistema de partículas a partir del que estamos utilizando podemos utilizar un operador denominado *Spawn* (engendrar). Hay dos operadores de este tipo, *Collision spawn* y *Spawn*. El primero genera partículas a partir de una colisión y se utiliza para cosas tales como simular gotas de agua que se crean al chocar una gota con un plano y rebotar formando nuevas gotas. El segundo es más genérico y simplemente crea un nuevo sistema de partículas a partir de otro dado y pasa este nuevo sistema de partículas a otro evento que hay que enlazar con el anterior.

1 Entrar de nuevo en *Particle view*. Añadir un operador de tipo *Spawn* al final, después de *Display*. Este operador incluye una conexión que utilizaremos para enlazarlo con un nuevo envento. Editar sus parámetros y marcar la opción *Once* (solo se generará un conjunto de partículas).
2 Crear un nuevo evento. Podemos crearlo siguiendo el procedimiento habitual o, más sencillamente, copiando un operador del evento anterior que nos interese y pegándolo en el panel. Para esto, situarse encima del operador *Shape facing*, hacer BDR *(*botón derecho ratón*)* / *Copy* y luego, en un área libre del panel de *Particle view*, *BDR* / *Paste*. Así se creará un nuevo evento, *Event 002*, con un operador *Shape facing 002* y un operador *Display 002*. O copiar y pegar el evento anterior y borrar todo menos estos dos operadores.
3 Editar los parámetros del operador *Shape facing 002* y cambiar el *Look at*… por la luz en lugar de la cámara. En *Display*, cambiar el tipo a mostrar por “Geometry”.
4 Para que el material del *Event* 001 se aplique igual al *Event 002* podemos copiarlo a éste último y situarlo, como antes, antes de *Shape facing*. Pero será más eficaz copiarlo a continuación de *Render 01*, en el cuadro inicial de *PF Source* (véase el esquema resumen que se da en la figura 5.52).
5 Conectar el *Evento 002* con *Spawn 001* (arrastar desde el círculo de salida del primero al cuadradito de entrada del segundo. Al establecer esta conexión deberá aparecer un nuevo sistema de partículas orientado hacia la luz. Si no se ve bien, cambiar los colores de los dos *Display* para que se distingan bien y, quizás, mover el *time slider* más allá del *frame* 0. Mover la luz y comprobar cómo las partículas del segundo evento giran para seguir el movimiento de la luz.

Pero si ahora hacemos un *render* lo que obtendremos será un lío (comprobarlo). Pues tendremos dos árboles y dos sombras super-

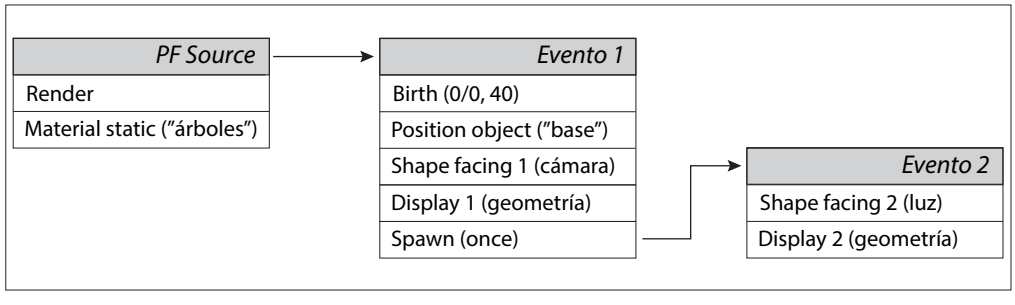


Figura 5.52 Partículas con mapas. Árboles. Esquema del sistema de partículas utilizado.

puestas. Así que, por un lado, tenemos que ocultar las sombras del primer sistema de partículas dejando los árboles y, por otro lado, ocultar los árboles del segundo sistema de partículas dejando las sombras.

6 Seleccionar el *Event 001* y hacer *BDR*/*Properties*. En el panel que se abrirá, en la ficha *General*, grupo *Render control*, presionar el botón "By layer" para que pase a "By object" y desactivar las opciones "Receive shadows", "Cast shadows" y "Apply atmospherics". Así este sistema no recibirá ni proyectará sombras.
7 Seleccionar el *Event 002* y hacer *BDR*/*Properties*. Repetir la operación pero, en este caso, desactivar todas las opciones ("Inherit visibility", "Visible to camera", "Visible to reflection/refraction", "Receive shadows", "Apply atmospherics") excepto "Renderable" y "Cast shadows". Así este sistema no será visible pero arrojará sombras.

De este modo hemos conseguido una simulación suficiente de los árboles. Pero los árboles son todos del mismo tipo lo que no resulta muy interesante. El siguiente y último paso será introducir más variedad de tipos y dimensiones. Para introducir mayor variedad de tipos utilizaremos un recurso que ya hemos visto: un material multisubobjeto con tantos submateriales como tipos nos interese utilizar. Véase el apartado sobre partículas sin mapas para recordar el proceso básico.

1 Desde el editor de materiales crear un nuevo material "arboles" de tipo *Multisubobject* con 3 submateriales. Arrastrar el material "arbol1" sobre el primer submaterial. Y luego desde este a las otras dos entradas y renombrarlas como "arbol2" y "arbol3". Ahora tenemos un material *multi* con tres copias idénticas a las del material que habíamos utilizado hasta aquí.
2 Editar los submateriales "arbol2" y "arbol3" y substituir los mapas de bits por los correspondientes a los diferentes tipos de que habíamos partido.
3 Desde *Particle view*, editar el operador *Material static* y substituir el material asignado previamente por "arboles". Marcar la opción "Assign material ID". Dar un valor de 3 a la opción "Sub-Materiales". Marcar la opción "Material ID" y dar sucesivamente a esta opción los valores 1, 2, 3. Activar *render* para comprobar que se asignan los submateriales correspondientes. Por último, cambiar esta última opción por "Cycle" o "Random" para que los submateriales se asignen por orden o aleatoriamente, según lo que interese. Probar también a cambiar el valor de aleatoriedad presionando el botón "New", en el grupo *Uniqueness*.
4 Ajustar todos los valores y, por último, redistribuir adecuadamente las partículas para que solo haya árboles en la zona de la cámara. Hay varios modos de hacer esto pero el más sencillo es, en *Position object*, escoger la opción "Selected faces". Convertir el plano a una malla editable con un número suficiente de caras y seleccionar las que convenga. Aumentar el número de partículas a un valor en torno a 40 o algo más, en función de lo que interese pero también de las capacidades del ordenador que se esté utilizando pues el resultado final puede implicar procesar más de 1 millón de caras poligonales según los casos. Así se llegará a un resultado final que deberá ser similar al de la última imagen (*d*) de la figura 5.51.

Ejemplo 4. Simulación de humo con mapas procedurales

Crear una escena adecuada para que aparezca algún objeto que emita humo, como la chimenea de la figura 5.53. Crear también una cámara adecuada para las vistas que obtendremos. En la descripción que sigue, primero trataremos de la configuración del flujo general para que el movimiento y densidad de las partículas sea el que nos interese, así como de la forma de las partículas para que se adecúen al tipo de simulación. A continuación veremos la configuración de los materia-

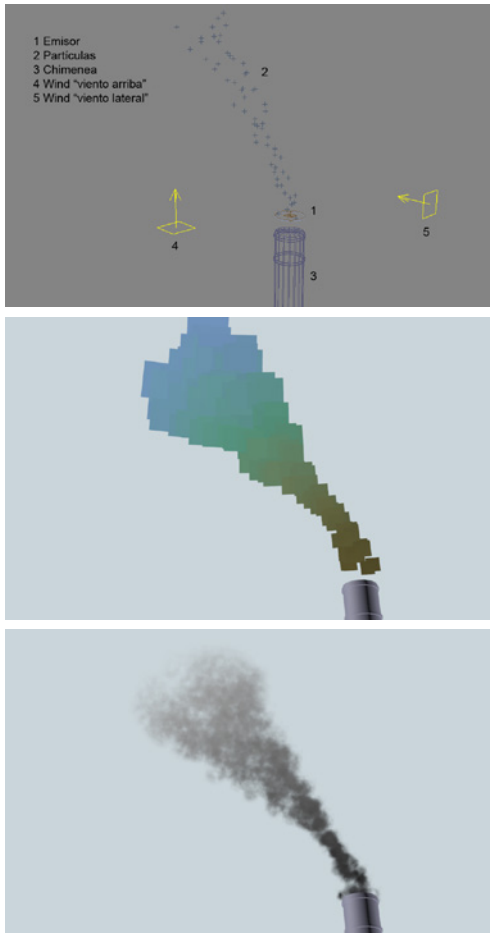


Figura 5.53 Partículas con mapas. Humo: a) *Vista general de la configuración de la escena;* b) *Vista intermedia con las partículas visibles y los colores de comprobación dados por el mapa particle age;* c) *Resultado final.*

les y mapas, que es el punto principal al que vamos a parar y en el que me extenderé más.

1 Crear un *PF Source*. Situarlo junto al objeto emisor. Girarlo 180º en vertical para que las partículas emitan hacia arriba. Cambiar su tipo a círculo y hacer su tamaño algo más pequeño que el objeto. En *Emission*, hacer que el número de partículas que se muestren en el visor sea 100 %. Mover el *time slider* para comprobar que se emiten partículas hacia arriba.

2 Abrir *Particle view*. Configurarlo así:

Birth. Emit Start/Stop > -100/300 (para un rango de 0/300; cambiarlo si es necesario). Amount > 2.000.
Position Icon. Sin cambios (volume).
Speed. Speed > 50. Variation > 1,0.
Shape. Type > 3D/Cube. Size: 4 (para que las partículas se vean con claridad pero separadas).
Display. Type > Geometry. Cambiar también el color para que se distingan mejor.

Comprobar el resultado. Para que el movimiento sea más interesante añadiremos dos fuerzas de viento, una hacia arriba y otra desde un lado.

3 Crear dos fuerzas de viento (*Create* / *Space warps* / *Forces* / *Wind*). Renombrarlas "vientoArriba" y "vientoEste" y situarlas tal como se muestra en la figura 5.53 (*a*). Volver a editar *Particle view* y añadir un operador *Force* antes de *Shape*. Editar sus parámetros y adjuntarle las dos fuerzas que hemos creado.

El penúltimo paso de la configuración del flujo será ajustar estas dos fuerzas para que el movimiento del humo sea menos rígido. Con la fuerza vertical hacia arriba controlamos principalmente la dispersión a medida que el humo subre. Con la fuerza horizontal, controlamos su movimiento hacia un lado. Hay que tener en cuenta que los valores dependen de las unidades (centímetros en el ejemplo) y del efecto que interese conseguir.

4 Editar los parámetros de las dos fuerzas y configurarlos más o menos así:

"vientoEste":
Force > 100, 0.03 (Strength, Decay).
Wind > 0,01, 3,0, 0,1 (Turbulence, Frequency, Scale).
"vientoArriba":
Force > 0.0, 0.0 (Strength, Decay).
Wind > 0,02, 10, 0,25 (Turbulence, Frequency, Scale).

5 Volver a comprobar el resultado y ajustar todos los parámetros utilizados hasta aquí. Por ejemplo, aumentar el número de partículas a unas 5.000.

Y el último paso será configurar adecuadamente la forma de las partículas.

6 Reducir el tamaño de las partículas a unas 200 para poder visualizar claramente las formas.
7 Substituir el operador *Shape* por un operador *Shape facing*. Editar sus parámetros. Cambiar el tamaño a unos 25 cm. Asignarle la cámara principal como "Look at camera…". Comprobar que las partículas cambian de aspecto y se orientan hacia la cámara al moverse.
8 Añadir un operador *Scale* después de *Shape facing*. Editar sus parámetros. Cambiar el tipo por "Relative succesive". Manteniendo activado "constraint proportions" cambiar algunos de los valores de "scale variation". Comprobar que las partículas aumentan de tamaño mientras ascienden.

Con esto hemos completado la primera fase de configuración general del flujo. En fase la siguiente configuraremos los materiales adecuados para simular humo. La idea básica es crear un material básico con una textura adecuada y un mapa de recorte con todos sus bordes negros para que el fondo se transparente.

1 Abrir el editor de materiales. Crear un nuevo material "humo", con la siguiente estructura:

"Humo" (Standard)
Diffuse color: mapa 1 (Particle Age).
Color 1: amarillo.
Color 2: rojo.
Color 3: azul

Los colores son temporales y nos servirán para comprobar mejor el resultado. Al final los cambiaremos.
2 Desde *Particle view*, añadir un operador de tipo *Material dynamic* al final, antes de *Display*. Editar sus operadores y asignarle el material "humo".

Si ahora comprobáramos el resultado no aparecería ningún cambio pues el mapa está ligado a la vida de las partículas y esto está por definir. Necesitamos romper el flujo con un operador *Delete*.
3 Insertar un operador de tipo *Delete* detrás de *Birth*. Editarlo y escoger la opción *Remove* / *By particle age*. Dar a *Life span* un valor de 350 y a *Variation* 50.
4 Comprobar el resultado en el *frame* 300. Las partículas deberán empezar su vida con el color 1 que hemos definido, pasar por el color 2 y terminar con el color 3.
5 Reeditar el mapa de *Particle age* como sigue de modo que el ruido se vaya haciendo algo más disperso a medida que la partícula crece:

…
Diffuse color: mapa 1 (Particle age).
Color 1 > mapa 11 (Noise):
fractal, levels 10, threshold 0.65/0.35, size 20.0,
color 1: gris muy oscuro (v 30),
color 2 gris oscuro (v 85).
Color 2 > mapa 12 (Noise):
fractal, levels 10, threshold 0.80/0.20, size 30.0,
color 1: gris oscuro (v 85),
color 2 gris medio (v 160).
Color 3 > mapa 13 (Noise)
fractal, levels 10, threshold 0.8/0.2, size 20.0,
color 1: gris medio (v 160),
color 2 gris claro (v 220).

"humo" (Standard)
Diffuse: map 1 (Particle age)
color 1: map 11 (Noise)
color 2: map 12 (Noise)
color 3: map 13 (Noise)
Opacity: map 2 (Particle age)
color 1: map 21 (Gradient)
color g1: negro
color g2: map 212 (Noise)
color g3: (map 213 (Noise)
color 2: map 22 (Gradient)
color g1: negro
color g2: map 222 (Noise)
color g3: map 223 (Noise)
color 3: map 23 (Gradient)
color g1: negro
color g2: map 232 (Noise)
color g3: map 233 (Noise)

Figura 5.54 Partículas con mapas. Estructura final del material "humo".

Comprobar el resultado antes de continuar. La textura (el humo) debe ser más oscuro y más detallado en el arranque y más claro y más disperso a medida que asciende. El siguiente paso será añadir un mapa de opacidad para ocultar los bordes rectos de las partículas. Para ello, añadiremos otro mapa de *Particle age* a *Opacity*. Luego añadiremos mapas de gradiente a las entradas de color gris y blanco. Al hacer el mapa de gradiente radial, el centro (blanco) será opaco y la periferia (negra) será transparente. Si, además, añadimos ruido, los bordes de la periferia serán irregulares. Después de configurar el color 1 de *Particle age* nos bastará con copiarlo y modificarlo para los otros dos.

6 Reeditar el material "humo" y modificar *Opacity* como sigue (véase el esquema de la figura 5.54).
7 Añadir otro mapa *Particle age*. Asignar a cada uno de los tres colores, inicio (0 % de la edad), medio (50 % de la edad) y final (100 % de la edad), de este mapa otro mapa de tipo *Gradient*, una variante sencilla de gradiente con solo tres colores.
8 Configurar cada uno de los mapas de tipo *Gradient* dejando el color 1 en negro y asignando un mapa de tipo *Noise* a los otros dos.
9 Los dos mapas *Noise* tienen una configuración similar. En el ejemplo son de tipo fractal, con mayor contraste (*high*/*low* 0,7, 0,3) y un tamaño de unos 20 cm. Pero la diferencia está en el color. Los correspondientes al color 1 de la edad de partículas, es decir, al arranque del humo, son de color negro y blanco con lo que el resultado es mayor opacidad. Los correspondientes al color 2, es decir, a la vida media, y por tanto, a la zona media del humo son de color negro y gris medio, con lo que serán algo más transparentes. Y los correspondientes al final son de color negro y gris oscuro con lo que serán todavía más transparentes.
10 Añadir luces y sombras. *Render*.

La estructura final del material se muestra en la figura 5.54.

5.9 Fondos y mapas de entorno

Fondos y mapas de entorno. Generalidades. Problemas en la práctica

Un componente fundamental de la simulación de muchas escenas es el fondo. Utilizar una imagen como fondo es importante tanto si aparece directamente, pues enriquece el resultado sin necesidad de ampliar el modelo, como si aparece indirectamente, a través de los reflejos o del modo en que modifica la distribución de la iluminación en una escena.

En general, los fondos pueden crearse y utilizarse de dos maneras:

a) Mediante mapas asignados a una envolvente virtual, plana, esférica o cilíndrica. Con 3ds Max el procedimiento es similar al de otros programas y consiste básicamente en asignar un mapa al fondo entrando en el panel *Environment* (menú *rendering* o tecla 8). Para editarlo, se arrastra al editor de materiales, en donde se activará, en la sección *Coordinates*, como tipo de proyección, la opción *Environment* en lugar de *Texture*. El proceso también puede hacerse a la inversa, crear el mapa de entorno desde el editor de materiales y de ahí arrastrarlo al panel *Environment*. Pero en este caso no se activará automáticamente la opción de proyección adecuada y deberá hacerse manualmente. En cualquier caso, convendrá comprobar cuál de las opciones de la lista colgante que incluye los tipos de proyección está seleccionada: 1) *Screen*: crea 6 versiones del mapa asignadas a cada lado de un cubo virtual envolvente y hace que su relación de aspecto coincida con la de salida especificada para el *render*; 2) *Spherical Environment*: aplica el mapa a una esfera virtual que rodea la escena; 3) *Cylindrical environment*: aplica el mapa a un cilindro virtual que rodea la escena; 4) *Shrink-wrap*: igual que *Spherical* pero con la proyección de los polos corregida.

b) Mediante mapas asignados a un objeto plano o cilíndrico situado en una determinada posición, en el exterior de la escena. En este caso, hay que crear el objeto en posición ade-

cuada y asignarle el mapa como si fuera una textura. Esta opción está más bien en desuso pero hay quien prefiere seguir utilizándola pues proporciona mayor control y evita muchos de los problemas que se comentan más adelante.

Los mapas utilizados pueden ser de cualquier tipo, es decir, mapas de bits corrientes, mapas de bits de tipo HDR o EXR, mapas procedurales que creen degradados o patrones diversos. En esta sección me centraré principalmente en los mapas de bits. El uso de mapas HDR o EXR proporciona posibilidades adicionales importantes pero que tienen que ver sobre todo con el uso de fondos como medio de iluminar una escena, y se trata extensamente en el libro sobre simulación de la iluminación.

El fondo se incorpora a la imagen al hacer un *render* pero también puede visualizase en el visor para ajustar su tamaño o su posición. En algunos programas el fondo que se muestra en el visor es el mismo que se mostrará al hacer un *render*. En otros, como 3ds Max, puede ser el mismo o no. En general, para componer adecuadamente un fondo nos interesará que sea el mismo. Y para ello, en 3ds Max hay que hacer lo siguiente. Abrir el cuadro de diálogo de configuración del *Viewport background*, pulsando las teclas Alt+B o desde el menú *Views / Viewport background* o desde el menú de visor *Configure viewports*. En este panel nos encontraremos con dos opciones principales: 1) Utilizar un archivo independiente para el fondo del visor, presionando el botón *Files*; 2) Utilizar el mismo fondo que hemos asignado como mapa de entorno, marcando la opción “Use Environment Background”.

Hay varios problemas que pueden surgir al utilizar fondos. Los principales son los siguientes: *a*) El fondo sale de color demasiado oscuro o demasiado claro; *b*) Los reflejos no se procesan bien; *c*) Los colores del fondo no se integran bien con los de la escena. Estos problemas tienen diferentes causas que requieren diferentes tratamientos y que se discuten en los apartados siguientes.

a) Fondos demasiado oscuros o claros

En la mayoría de los casos esto puede ser debido a un problema de exposición cuando se utilizan sistemas de iluminación avanzada (y que se discute también más extensamente en el libro sobre simulación de la iluminación) similar al que ocurre en la vida real. Si nuestra vista está adaptada a un interior oscuro, el exterior parece excesivamente brillante. Y viceversa. Algo similar ocurrirá si utilizamos una cámara: tendremos que ajustar la exposición para visualizar adecuadamente el interior o el exterior. Pero no podremos ajustarla para que se vean los dos bien a la vez. Y lo mismo ocurre en las escenas virtuales con luces que simulan luces reales. Si la escena está iluminada por luces virtuales potentes, como las que se utilizan para simular el Sol y la irradiación celeste, al utilizar una imagen de bajo rango dinámico para el fondo puede resultar completamente negra pues su valor se traduce al rango de la escena y el resultado es un valor muy bajo.

Una solución simple, que en muchos casos puede ser suficiente (a partir de la version 2010 de Max), es, desde el panel *Environment* / *Exposure control*, desactivar la opción “Process Background and Environment Maps”. Al hacer esto, el algoritmo de control de exposición no aplica la redistribución de intensidades al fondo.

Sin embargo, esto puede provocar otro tipo de problemas por lo que no hay que descartar otras opciones algo más complicadas. En el caso particular que veremos más adelante, el uso de un material de tipo *matte*/*shadow*, el fondo queda incorporado a la escena a través de un plano al que se asigna este material que retiene las sombras pero hace que el fondo se visualice a través de las partes que no reciben sombras. Como, en este caso, el fondo “está en la escena”, el recurso anterior no sirve y el resultado será que aparecerá negro. Para esto, una solución efectiva es, si se utiliza el control más recomendable, el *mr Photographic exposure control*, desactivar la opción predeterminada *Physical scale* e introducir un

valor manual adecuado a través de la opción *Unitless*. Este valor, que substituirá al predeterminado (1,0) debe corresponderse con los valores más altos de la escena que si se utiliza un sistema de luz diurna, y estará en torno a los 80.000 o 100.000 (el valor de luxes generados por el Sol) o algo más. Puede comprobarse cuál es este valor, aunque no sea estrictamente necesario, cambiando el tipo de sol a *IES Sun* lo que permitirá visualizar la cantidad de luxes correspondiente a la posición del Sol y luego volviéndolo a dejar como estaba, probablemente, como tipo *mr Sun*).

Esto indicará al programa que procese el fondo con valores adecuados al rango utilizado y se solucionará el problema, aunque puede ser neceario hacer pruebas hasta encontrar el valor más adecuado. Otra posibilidad sería asignar al mapa de bits del fondo un mapa *Gamma* & *Gain* para modificar su salida aumentando sus valores de irradiación. Pero la solución anterior será más controlable.

Hay algunos usuarios que, para evitar estos problemas y otros que veremos más adelante, relativos a los reflejos, prefieren el viejo método de utilizar un plano, situado adecuadamente para simular el fondo al que ya me he referido antes. En este caso también puede ocurrir que no resulte visible o que quede muy oscuro. Pero la explicación es más simple y más fácil de evitar. Lo más probable es que no resulte visible por la sencilla razón de que no recibe luz, pues las luces virtuales estarán dirigidas hacia objetos de la escena. En este caso, para no alterar la iluminación general, lo más sencillo es iluminarlo frontalmente con una luz directa y utilizar el botón *Exclude*, de los parámetros de la luz, para excluir todos los objetos de la escena excepto este plano que simula el fondo.

Si se quiere comprobar exactamente todo lo que puede ocurrir en estos casos, podemos provocar estos problemas haciendo lo siguiente.

Crear una escena sencilla, un objeto sobre un plano (o abrir una escena que incluya, por ejemplo, un edificio sobre un terreno). Iluminar la escena con un *daylight system* de tipo mr (*mr Sun* y *mr Sky*). Utilizar el fondo predeterminado *mr Physical sky* (un mapa procedural que se adapta a los cambios de posición del Sol). Comprobar que el control de exposición es adecuado para esta escena: *mr Photographic exposure control* con EV 15. Configurar el cálculo con *Final gather* en modo *Draft* para que vaya más rápido. Hacer un *render* que deberá ser correcto para estos valores. Ahora editar el *mr Physical sky* (arrastrarlo desde el panel de *Environment* al editor de materiales). Marcar la opción “Use Custom Background Map” y escoger un JPG corriente que represente, por ejemplo, un cielo con nubes. Desde el control de exposición marcar o desmarcar la opción “Process Background and Environment Maps”. Comprobar que, si se desactiva esta opción, la imagen de fondo aparece correctamente. Pero que si se activa, sale de color negro. Y si se desactiva el control de exposición, la imagen de fondo aparece correctamente pero la escena aparece blanca, quemada.

Esto es debido a que, si se activa el control de exposición, la imagen aparecerá negra al procesarse, pues su rango dinámico queda muy por debajo del de la escena. Por otro lado, si se excluye del procesamiento mediante la opción citada, aparecerá correctamente. Y si se desactiva el control de exposición también aparecerá correctamente; pero ahora es la escena la que aparece distorsionada hacia el blanco pues su rango dinámico queda muy por encima de los valores convencionales.

Por otro lado, si mantenemos activado tanto el control de exposición como la opción citada (“process background...”) pero, en el grupo *Physical scale* cambiamos la opción predeterminada, *Physical units*, por *Unitless* y le asignamos un valor muy alto, del orden de 150.000 o 200.000 (el máximo que admitirá), el resultado será correcto. En un caso como este no tiene mucho sentido usar este recurso pero, como veremos, en casos especiales en que el fondo está incorporado a la escena mediante materiales especiales sí lo tendrá.

b) Fondos que no se reflejan bien

Otro problema corriente es que los reflejos no muestran adecuadamente el fondo que se supone que deberían reflejar. Si se crea una escena con, por ejemplo, un suelo o un muro reflectantes, y hay un mapa de bits como mapa de entorno que simula un fondo, este fondo no aparece en los reflejos o aparece a través de los objetos, como si estos fueran transparentes.

Ya hemos visto las causas de este problema (el sistema no puede aplicar adecuadamente los algoritmos de cálculo de reflejos, que se basan en el envío de rayos trazadores que van rebotando en los objetos de la escena, a un objeto que no está en la escena), y el modo de solucionarlo, en la sección sobre reflejos. Hay básicamente dos soluciones: o bien utilizar como fondo un objeto plano o cilíndrico al que se aplica un mapa, como ya he indicado o bien utilizar un procedimiento más efectivo pero algo más complicado: un grupo de *shaders* especiales de mental ray que se agrupan en la colección denominada *Production shaders*, principalmente el *Environment / Background switcher* combinado con *Environment / Background / Camera map* y para conseguir efectos adicionales, *Matte / Shadow / Reflection*. En los apartados siguientes veremos con más detalle cómo utilizar estos *shaders* con nuevos ejemplos.

c) Fondos de colores inadecuados para la escena

En muchos casos el desajuste entre fondo y escena será notorio. Para esto hay dos soluciones bastante obvias. La primera es buscar un fondo que se relacione suficientemente bien con la escena en lugar de utilizar el primero que encontremos. Algo que puede llevar más tiempo del que a uno le gustaría pero que es inevitable para conseguir un resultado digno.

La segunda es ajustar los valores de los fondos creando en la escena algunos objetos de referencia con colores relacionados con algunos tonos del fondo y comprobando que las variaciones de tono e intensidad en la escena sean adecuadas. También abordaremos estos problemas en los apartados siguientes.

Composición de escenas con fondos. Concordancia de cámaras y luces

Cuando se utilizan imágenes de fondo de tal modo que partes del modelo parezcan ser parte de ese fondo, como veremos en los ejemplos que siguen, será necesario que se dé suficiente concordancia entre los elementos de la escena y las imágenes del fondo. Esta concordancia deberá ser de tres tipos principales: *a*) La proyección geométrica de la escena debe coincidir con la proyección perspectiva del fondo; *b*) Las sombras que aparezcan en la escena deben coincidir con las del fondo; *c*) Los colores, la distribución tonal de la escena deberán coincidir con los del fondo.

Cada uno de estos tipos de concordancia requiere técnicas específicas que se resumen en este apartado.

a) Ajuste de cámara

Podemos ajustar la posición y el ángulo de la cámara de varios modos. Uno de los más sencillos y más efectivos es derivar el ángulo de la cámara a partir de los datos EXIF (*Exchangeable Image File*) y luego ajustar su posición manualmente. Este es el método en que detallaré en este apartado.

Si la imagen no tiene datos EXIF podemos calcular su ángulo mediante una reconstrucción geométrica manual. Puede encontrarse este método en un buen manual de geometría descriptiva. En lo esencial se reduce a lo siguiente: *a*) En un programa adecuado, como AutoCad, insertar la imagen y redibujar sobre ella prolongando las aristas de objetos de caras ortogonales hasta encontrar sus puntos de fuga; *b*) Así se formará un triángulo: el ortocentro de este triángulo es el punto principal de la imagen, P, que, si la imagen no ha sufrido manipulaciones, deberá coincidir con su centro; *c*) si se abate este punto por cualquiera de sus lados, trazando un semicírcu-

lo cuyo diámetro sea un lado del triángulo y una perpendicular a este lado desde P, habremos abatido uno de los lados del triedro rectangular cuyo vértice V es el punto de vista. A partir de ahí es sencillo calcular la distancia de V a P por métodos manuales o numéricos.

Otra posibilidad, que también sirve para imágenes que no tengan datos EXIF, es utilizar un programa con recursos adecuados para obtener la posición de la cámara. Y una alternativa sencilla y bastante práctica es utilizar el *PhotMatch* de SketchUp que no voy a explicar con detalle pues esta explicación se puede encontrar en tutoriales de Google o la ayuda del programa. Bastará con, desde SketchUp, ir al menú *Cámara / Adaptar nueva fotografía* y seleccionar la imagen en el cuadro de diálogo que se abrirá. La imagen seleccionada se mostrará como fondo junto con unas líneas rojas (eje X), verdes (eje Y) y azules (eje Z). Hay que mover estas líneas hasta que coincidan con líneas de fuga de elementos ortogonales de la imagen. Cuando se haya conseguido esta coincidencia, pulsar el botón *Finalizado* del pequeño cuadro adjunto. Luego situarse encima de la pestaña correspondiente a la escena y activar el comando *Actualizar* para que se grabe la vista. Para llevar este resultado a otro programa grabar el archivo e importarlo desde el programa marcando tan solo la opción “importar cámara”. Luego asignar la cámara a un visor y continuar tal como se explica a continuación para ligar la cámara al fondo, pues ya tenemos la cámara con su ángulo que es lo que buscábamos. En este caso, por añadidura, la vista ya está adaptada y los elementos que se creen desde la vista de cámara quedarán orientados en el mismo espacio perspectivo que el fondo.

Si la imagen tiene datos EXIF podemos calcular su ángulo con facilidad y más precisión mediante un cálculo trigonométrico simple. Las imágenes digitales actuales incorporan, en la gran mayoría de los casos, este tipo de datos que, entre otras cosas, incluyen la distancia focal y el modelo de cámara. Para obtener los datos EXIF de una imagen todo lo que hay que hacer es, desde el explorador de Windows, seleccionarla y hacer BDR (botón derecho ratón) / *Propiedades*. Con esto se abrirá el cuadro de diálogo de *Propiedades* de la imagen. Ir a la sección *Detalles* y en el grupo *Imagen* econtraremos las dimensiones de la imagen. En el grupo *Cámara* encontraremos el modelo de cámara y la distancia focal a la que fue sacada la fotografía. Anotar estos tres datos: dimensiones, modelo de cámara y distancia focal. El modelo de la cámara nos servirá para averiguar, si no lo conocemos, el tamaño del sensor. Buscar por internet con las palabras “especificaciones modelo tal y tal” o ir directamente a una página web especializada en fotografía como www.dpreview.com.

Para que resulte más clara la explicación pondré un ejemplo con datos reales. Supongamos que contamos con una fotografía de 3.216 x 2.136 px obtenida con una cámara Nikon D90 y una distancia focal de 21 mm. Las especificaciones de este modelo nos indican que el tamaño del sensor es de 23,6 x 15,8 mm.

A partir de estos datos podemos obtener directamente el ángulo horizontal de la cámara. La tangente de la mitad de este ángulo será igual a la mitad de la anchura del sensor dividida por la distancia focal. Y el ángulo horizontal (FOV: *Field Of View*) será, por tanto el doble del arco tangente correspondiente a este ángulo. Es decir:

$FOV_{HORIZONTAL} = 2atan(a/2d)$
$= 2atan(23,6 / 21 \times 2) = 58,7º.$

Conocido este ángulo el procedimiento será el siguiente:

1 Abrir un nuevo archivo en 3ds Max. Ir a *Environment* (tecla 8). Asignar un mapa al fondo y escoger la imagen de que estamos partiendo. Configurar el tamaño de salida con *Render setup*. Dar a la salida las mismas dimensiones que la imagen (3.216 x

2.136), luego bloquear la relación de aspecto y reducir al valor que se prefiera para la salida final, por ejemplo, 1.600 px de ancho o un valor más pequeño para pruebas. El valor de altura cambiará automáticamente. Y desde el visor, activar *Show frames* (Shift+F) para que se visualice el marco de salida.

2 Crear una cámara en planta que apunte ortogonalmente hacia el visor frontal (por ejemplo con su *target* en 0, 0, 0 y con la cámara en 0, -100,0). Luego activar una vista de esta cámara en un visor frontal. Hacer Alt+B para abrir el cuadro de diálogo *Viewport background*, activar la opción "Use environment map". Al salir de este cuadro deberemos ver en el visor el mapa de fondo ajustado al marco.

3 Si ahora se crea un objeto de referencia, un plano o un cubo, preferiblemente centrado inicialmente en 0, 0, 0, al mover la cámara comprobaremos que se puede adaptar la vista con facilidad para que tres líneas de fuga del objeto se adapten a las de elementos ortogonales del fondo pues nos estamos moviendo en un espacio perspectivo igual. Una vez que la cámara esté ajustada correctamente al fondo, podemos crear otros objetos que se adaptarán directamente a este espacio perspectivo.

b) Ajuste de sombras

El siguiente paso, después de ajustar la cámara al fondo con algunos de los métodos anteriores, es hacer que las sombras arrojadas sobre la escena coincidan con las de la imagen de fondo. Evidentemente, eso solo será necesario si en la imagen aparecen con claridad sombras o indicaciones explícitas de la dirección de la luz. Supondremos, por tanto, que este es el caso.

Para conseguir este objetivo es preferible ocultar el modelo virtual sobre el que vamos a trabajar y utilizar dos objetos auxiliares que luego podemos eliminar o esconder: un plano, situado en la misma cota que los objetos principales del modelo, y un prisma (o un cilindro), situado en el centro de este plano. Esto nos permitirá ver bien la dirección de las sombras.

Luego crear un sistema de luz diurna. Configurarlo en modo manual. Activar la visualización de sombras en el visor y mover el sol virtual hasta que las sombras estén dirigidas de modo que coincidan con otras sombras de la escena. Si fuera necesario aumentar la altura del prisma y la amplitud del plano. O bien crear una copia de la imagen de fondo y dibujar sobre ella líneas que destaquen y que indiquen la dirección de las sombras.

En el ejemplo que se dará más adelante se muestran estos resultados parciales.

c) Ajustes de intensidad y color

El tercer y último paso es conseguir que los colores del modelo queden bañados por una luz de las mismas características que los de la imagen para que la concordancia de tonos sea adecuada. El mejor método será captar algún color de la imagen que esté orientado igual que nuestros planos auxiliares y ajustar la intensidad para que el color resultante sea el mismo.

Para esto, tenemos tres parámetros con los que jugar: la intensidad del Sol, la intensidad del cielo y el control de exposición. Pero es preferible utilizar solo los dos primeros. El procedimiento resumido es el siguiente.

1 Abrir la imagen de fondo desde el editor de materiales (arrastrar el mapa desde *Environment* hasta un visor vacío si no se ha hecho antes). En la sección *Bitmap parameters* presionar el botón *View image*. La imagen se abrirá en un panel independiente con varios controles. Haciendo BDR sobre ella se obtiene información sobre el color de un punto. Captar un punto adecuado y anotar sus valores. O, mejor, utilizar el *Color clipboard* en el panel *Utilities* que permite grabar colores de cualquier panel del programa.

2 Crear un nuevo material y asignar a *Diffuse* este mismo color (escribir sus valores

o, desde el selector de colores, utilizar el cuentagotas para volver a captarlo, esta vez del *Color Clipboard*).

3 *Render*. Analizar el resultado y ajustar los valores que sean necesarios en función de este análisis.

Si el resultado es demasiado oscuro o demasiado claro modificar la Intensidad del Sol cambiando el valor del multiplicador. Si las sombras quedan demasiado oscuras o demasiado claras cambiar también la intensidad de la luz celeste del mismo modo, modificando el multiplicador. También se puede ajustar el control de exposición pero esto afecta a los dos valores a la vez y probablemente es más difícil de controlar. También se puede variar la tonalidad de las sombras ajustando ligeramente los valores de la irradiación celeste. Para ello, editar el sistema de luz diurna y, en *mr Sky advanced parameters*, en el grupo *Non-Physical tuning*, aumentar ligeramente el valor predeterminado (0,0) de *Red* / *Blue tint* para hacer los tonos más cálidos (más rojizos) o disminuirlos ligeramente para hacerlos más fríos (más azulados). Y reducir el valor predeterminado (1,0) de *Saturation* para hacer los tonos más neutros o aumentarlos para hacerlos más saturados

En el ejemplo que se dará más adelante también se muestran estos resultados parciales.

Integración de sombras y reflejos con imágenes de fondo. Material *matte*

Con los métodos anteriores podemos conseguir que nuestro modelo quede integrado en la escena con la misma orientación y las mismas características de iluminación que la imagen de fondo. Pero, en la mayoría de los casos, esto no será suficiente pues en las zonas de transición entre el modelo y el fondo habrá un salto brusco que echará a perder la simulación. Podemos editar el resultado en Photoshop o Gimp y disimular este salto. Pero el procedimiento será laborioso y pesado y costará conseguir resultados convincentes.

Dado que este trabajo es habitual en las aplicaciones profesionales, muchos programas cuentan con un *shader* especial que, al aplicarlo a un objeto, hace que se retengan las sombras o los reflejos, o ambos, y se transparente el fondo. El resultado es una perfecta integración del modelo y el fondo.

Este material se encuentra con diferentes denominaciones en algunos programas. En V-Ray, por ejemplo, se denomina *VRayMtlWrapper*. En Blender se consiguen algunos resultados similares con la opción “shadow only” pero con bastantes limitaciones. En los programas que utilizan mental ray se denomina *Matte*/*Shadow*/*Reflection* y es un *shader* muy eficaz. De hecho, es un material que pertenece a un grupo de *shaders* especiales, utilizados para la composición de fondos y que mental ray denomina *Production shaders* (estos *shaders* no han estado abiertamente disponibles hasta la versión 2009. En versiones anteriores podían utilizarse desocultando las líneas correspondientes de las bibliotecas de mental ray que se instalan con el programa pues no aún no habían pasado los requisitos de Autodesk, del *Autodesk’s Quality Assurance Team*). Como en los apartados siguientes también se utilizan otros *shaders* de este grupo, doy un breve resumen de estos *shaders* y sus parámetros. Puede encontrarse más información en la ayuda del programa.

En total hay unos nueve *productions shaders*: un material *Matte/Shadow/Reflection*, dos *Output shaders* para *Motion blur*, usados principalmente en animación, un *Lens shader* que permite hacer un *render* de un subconjunto de objetos en la escena definidos por objeto y por material, y cinco *Texture shaders*: *Utility Gamma&Gain* (se utiliza para ajustar la intensidad y distribución de la intensidad), *Environment / Background switcher*, *Environment / Background camera map*, *Environment probe / Chrome ball* y *Environment probe / Gray Ball.*

Los que nos interesan principalmente son los cuatro últimos que describo brevemente a continuación (ampliar información con la

ayuda del programa). A menudo se utilizan en combinación, como veremos en los ejemplos que siguen.

Environment / Background camera map. Este *shader* funciona como un mapa de entorno corriente pero que mejora las reflexiones al calcular transformaciones inversas. En realidad está ligado al *Environment / Background switcher* o al *Matte*/*Shadow*/*Reflection* y no tiene mucho sentido por sí mismo pues se utiliza principalmente para permitir utilizar otros mapas en combinación. Incluye los parámetros siguientes: *Map*. Es el parámetro principal y asigna un mapa de fondo a la cámara. *Multiplier*. El valor prefijado es 1,0. Si se aumenta o disminuye, aumenta o disminuye la luminosidad del fondo (y en consecuencia la de los reflejos). Los restantes parámetros tienen menos importancia: *Reverse gamma correction* (aplica una corrección de gama inversa al mapa); *Per-pixel matching* (ajusta la imagen al fondo mediante una proyección píxel a píxel); *Force transparent alpha* (si se activa, fuerza el alfa del fondo a 0, si no se activa utiliza el valor alfa del mapa); *Off-screen ray return environment* (puede ocurrir que algunas superficies reflejen puntos en una dirección que no están en pantalla y que el mapa, por tanto, no incluye); al activar esta opción se fuerza al programa a utilizar en cualquier caso puntos del mapa; *Off-screen color* (si la opción anterior está desactivada se puede escoger un color para los puntos que quedan fuera del mapa.

Environment / Background switcher. Este *shader* solo tiene dos entradas que dan acceso a usar un mapa como mapa de fondo y otro como mapa de reflexión. Véase la figura que se adjunta al ejemplo que usa este mapa, más adelante.

Environment probe / Chrome ball. Este *shader* se utiliza en combinación con una imagen panorámica de un tipo concreto que se genera a partir de una bola cromada. Véase el ejemplo que se da más adelante. El parámetro principal es el propio mapa que se carga pulsando el botón junto a *Chrome*/*Mirror*. *Multiplier* y *Reverse gamma correction* tienen el mismo sentido que en el primer *shader* citado. *Blur* se puede utilizar para ajustar el desenfoque en algunos casos.

Además, hay un material especial, ***Matte / Shadow / Reflection***. Este material, como ya he avanzado antes, permite integrar una forma específica de lo que se denomina técnicamente *differential rendering* o *differential shading*. Esto quiere decir que calcula la cantidad de luz que recibiría un punto si no estuviera en sombra, la compara con la cantidad de luz que recibe y asigna al material la diferencia. De este modo se representa el color propio del material, sin influencia de la luz de la escena. Esto facilita la integración con el fondo. De hecho es una colección de *shaders* con parámetros que están divididos en las siguientes siete secciones. La principal es la primera:

a) *Matte / Shadow / Reflection parameters*. Incluye cuatro parámetros: *CameraMapped Background* asigna un color o un mapa al objeto (la opción habitual es escoger el mismo mapa que el fondo y en modo *Screen*); si se incluye una máscara que oculte parte del fondo, el parámetro *Mask/Opacity* controla la opacidad del material al disminuir el valor prefijado, 1,0; también se pueden asignar mapas de relieve con *Bump* y controlar su efecto con el cuarto parámetro de esta sección, *Bump amount*.

b) *Shadows*. El principal control es *Receive shadows* que está activado por defecto y que hace efectivos los siguientes. El parámetro *Ambient* / *Shadow intensity* (0,2) controla la cantidad de luz ambiente lo que afecta a la oscuridad de las sombras. Es preferible controlar esta intensidad a partir de la sección *Ambient occlusion*. El color predeterminado de *Ambient* / *Shadow color* es blanco y, como antes, es preferible controlarlo desde *Ambient occlusion*. El último parámetro, *Shadow casting lights list* puede activarse si hay varias luces y se desea seleccionar las que crean sombras sobre este material.

c) *Ambient occlusion*. El principal control es *Use ambient occlusion* (*AO*) que está ac-

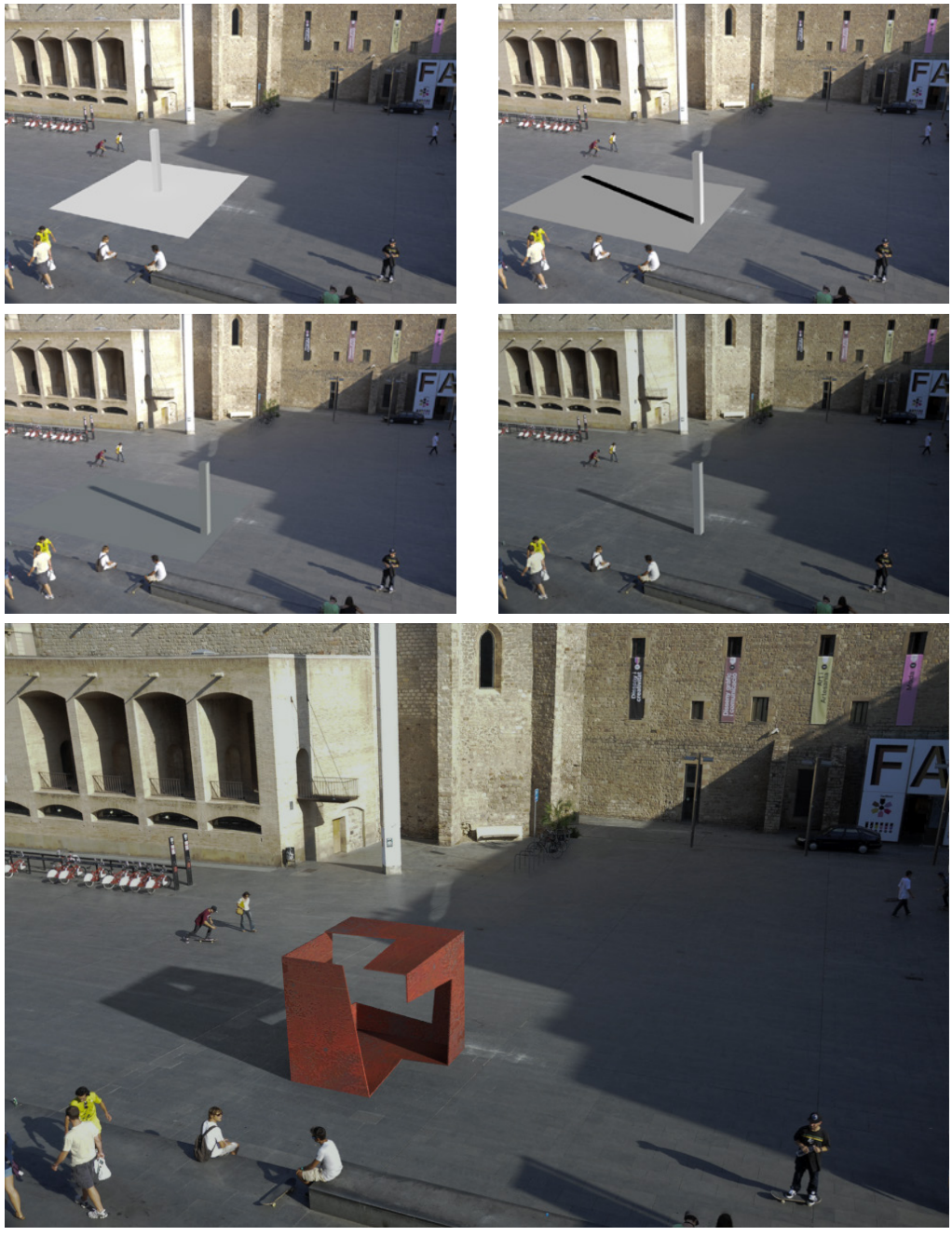


Figura 5.55 Integración de modelo y fondo. Ejemplo 1. Iluminación directa: a*) Ajuste de la vista de cámara;* b*) Ajuste de la dirección de las sombras;* c*) Ajuste de la intensidad;* d*) Aplicación de un material* Matte/Shadow *a la base;* e*) Substitución de los objetos auxiliares por el modelo.*

tivado por defecto y que hace efectivos los siguientes. Con *AO Samples* (8) se aumenta o disminuye el número de muestras y con ello la calidad del efecto. El valor de *AO Max distance* (0,0 = infinite) afecta la distancia a la que actúa el efecto y que depende de las características de la escena. El valor predeterminado es 0,5 m. La intensidad de la sombra se controla con *AO Shadow strength* (negro). En muchos casos convendrá modificar este valor haciéndolo algo más claro.

d) *Reflection*. El principal control es *Receive reflections* (*AO*), desactivado por defecto. Si se activa, quedan disponibles los siguientes. Con *Reflection color* (gris) se modifica el color de las reflexiones si interesa hacerlo por alguna razón, aunque es conveniente dejar el color neutro predeterminado. Con *Reflections* (*Subtractive color*) se controla la cantidad de color que se elimina del fondo antes de añadir los reflejos. Con blanco (predeterminado) el color del fondo es substituido al máximo por los reflejos del objeto con lo que el resultado es más oscuro y más notorio. Con negro no se substituye nada y el color de los reflejos se suma a los existentes con lo que el resultado puede llegar a ser imperceptible, una leve alteración de los tonos. Con valores intermedios se puede controlar la intensidad del reflejo. El valor de dispersión del brillo, *Glossiness*, puede disminuirse por debajo de su valor predeterminado (1,0, especular) para simular reflejos dispersos. En este caso será conveniente aumentar el número de muestras (*Glossy samples*, 8). El parámetro *Max distance* (0,0) permite controlar la distancia a la que se aplican los reflejos si el valor es distinto de 0.0. En este caso, con *Max distance falloff* se puede controlar la velocidad de atenuación con un parámetro que controla la forma de la curva: valores inferiores al predeterminado (2,0) hacen que la caida sea más rápida y viceversa.

e) *Indirect illumination*. El principal control es *Receive indirect illumination* que está activado por defecto. Incluye un multiplicador que intensifica localmente el efecto sin afectar al resto de la escena.

f) *Direct illumination*. El principal control es *Receive direct illumination* que está desactivado por defecto. Si se activa, computa la iluminación directa de todas las luces o de las seleccionadas por medio del control *Illuminating lights list* que permite incluir una selección de las luces de la escena.

g) *Maps*. Incluye todos los mapas utilizados en secciones anteriores.

El ejemplo siguiente permitirá entender mejor el funcionamiento de estos *shaders*. Está basado en dos archivos que el lector puede adaptar a casos similares.

El primero es una fotografía de la plaza dels Àngels, en el casco viejo de Barcelona, tomada desde una terraza del MACBA (Museu d'Art Contemporani de Barcelona). El segundo es un modelo de una escultura de Oteiza. La idea es insertar esta escultura en la plaza de modo convincente.

Siguiendo los pasos anteriores, la primera fase consistirá en ajustar la cámara y crear objetos auxiliares.

1 Seguir el procedimiento que hemos visto antes para obtener el ángulo de la cámara. Luego situar la cámara en el modelo haciendo que apunte a 0, 0, 0, asignar a la cámara este ángulo y asignarle un visor. Entrar en *Render setup* y dar como tamaño de salida los mismos que los de la imagen, luego bloquear la relación de aspecto y dar un tamaño adecuado para pruebas que al final cambiaremos por el definitivo. En el visor de cámara activar la visualizacion de los marcos (Shift+F) con lo que aparecerá sobre el visor un marco con la misma proporción que la imagen. Ir al panel *Environment* y asignar la imagen como mapa de entorno. Por último, hacer que el fondo de visor sea el mismo que el de salida. Revisar las explicaciones más detalladas que he dado anteriormente si esta descripción es demasiado condensada.
2 Crear un plano de dimensiones adecuadas (que abarque la escultura y su sombra). Situarlo en 0, 0, 0. Mover y rotar la cámara

hasta conseguir que las aristas del plano fuguen igual que líneas de referencia de la imagen.

3 Añadir un prisma de unos 20 x 20 x 200 cm. Luego mover el plano y el prisma hasta que se sitúen en la zona en la que queremos colocar la escultura. Darles colores que ayuden a distinguirlos del fondo. Así se llega a un primer resultado como el de la figura 5.55 (*a*).

La segunda fase consiste en ajustar las luces y los materiales.

1 Crear una luz directa auxiliar (también se puede crear directamente el sistema de luz diurna y ajustarlo pero con una luz directa iremos más rápido). Hacer que el *target* coincida con la base del prisma. Configurar el visor para que muestre las sombras (ir a *Viewport configuration*, desde el menú de visor y, en la sección *Visual style & appearance* comprobar que el estilo visual es "Realistic", que está activada la opción "Illuminate with Scene Lights" y que están activadas las sombras). Mover la luz hasta que las sombras concuerden con las del fondo. Si es necesario, substituir el fondo por una imagen auxiliar con líneas dibujadas que ayuden en este proceso. La finalidad es llegar a un resultado como el de la figura 5.55 (*b*). Luego substituir la luz directa por un sistema de luz diurna y ajustar el control de exposición a *mr Photographic* con EV:15 (o aceptar que el programa lo haga automáticamente). Hacer que la posición de la rosa de los vientos (el *target* del sistema diurno) coincida con el *target* de la luz auxiliar. Cambiar su tipo de posición a manual y mover la luz para que coincida con la luz auxiliar. Luego borrar o desactivar la luz auxiliar. Como decía antes también se puede prescindir de la luz auxiliar y trabajar desde el comienzo con el sistema diurno.
2 *Render* para una primera comprobación del resultado. Revisar la sección anterior sobre "Problemas con fondos..." si el resultado no fuera el que esperamos. Luego captar el color de elementos cercanos al plano y anotarlos o llevarlos a la utilidad *Color clipboard* (panel *Utilities*). Y luego asignar al plano y al prisma un material *Arch&Design* con este color.
3 *Render* de nuevo, que deberá ser similar a la imagen de la figura 5.55 (*c*) y analizar el resultado. Ajustarlo como sigue:
   Si el color del plano es muy distinto de lo que le rodea cambiar la intensidad del Sol. En el ejemplo se ha aumentado el multiplicador a 1,6.
   Si el color de la sombras es demasiado oscuro aumentar la intensidad de la luz celeste. En el ejemplo se ha aumentado el multiplicador a 3,5.
   Si la tonalidad de las sombras es demasiado fría o cálida o demasiado saturada, modificar los valores (en *mr Sky advanced parameters*) de *Red* / *Blue tint* y de *Saturation*. En el ejemplo se ha reducido el valor de saturación a 0,55.

Ahora que ya tenemos controlada la dirección y la intensidad de la luz podemos substituir el material del plano por el material MSR (*Matte/Shadow/Reflection*). Al aplicar este material, las partes opacas del objeto iluminadas se asignarán a un canal alfa y aparecerán transparentes dejando ver el fondo mientras que las partes que reciben sombras (o reflejos si se activa esta opción) las retendrán.

Sin embargo, esto no es tan inmediato si estamos utilizando un sistema de iluminación avanzada, pues para que el efecto funcione adecuadamente habrá que ajustar la intensidad y el control de exposición tal como hemos visto en el primer apartado de esta sección. Al activar el *mr Photographic exposure control* el fondo que se muestra a través del plano aparecerá negro pues la imagen que estamos utilizando ha quedado incorporada a la escena y no está preparada para recibir las altas intensidades del sistema de luz diurna, como ya hemos visto en el apartado sobre problemas con fondos. Así que, además de añadir el material *Matte/Shadow* habrá que hacer ajustes adicionales:

1 Desde el editor de materiales crear un nuevo material de tipo mr MSR (*Matte / Shadow / Reflection*). Dejar los valores predeterminados y asignar a *Camera mapped background* el mismo mapa que usamos para el fondo (arrastralo desde un visor o desde *Environment* sobre el pequeño cuadrado junto a *Camera mapped background*.
2 *Render*. El resultado será que el plano aparecerá de color negro.
3 Cambiar el tipo de *mr Sun* a IES para averiguar la intensidad en luxes. En el ejemplo son 80.000 luxes. Deshacer el cambio.
4 Ir al panel del mr *Photographic exposure control*. En el grupo *Physical scale* cambiar la opción predeterminada, *Physical units* por *Unitless*. Introducir un valor igual al obtenido, 80.000. Comprobar que está activado “Process Background and Environment Maps”.
5 *Render*. El resultado ahora deberá ser correcto aunque quizás haya que hacer algunas pruebas adicionales y subir el valor de 80.000 hasta conseguir que la integración entre plano y fondo sea perfecta. También se puede ajustar el resultado variando la intensidad, como ya hemos visto o modificando ligeramente, desde el panel *Exposure control*, los valores del grupo *Image control* (anotar antes los valores predeterminados por si hubiera que restituirlos: 0,25, 1,0, 0,2, *high*, *mid*, *shadows*) haciendo que los valores altos, medios o bajos sean más o menos intensos. Así se obtendrá un resultado como el de la figura 5.55 (*d*).
6 Por último, substituir el prisma por la escultura y, si fuera necesario, revisar los ajustes que hemos hecho y que se recopilan con alguna adición en el párrafo siguiente. Así se llega al resultado de la figura 5.55 (*e*).

En general, conviene afinar el resultado final con los controles siguientes:

Para ajustar la iluminación del objeto y su sombra propia, modificar las intensidades en el sistema de luz diurna.

Para ajustar la intensidad de la sombra arrojada, editar el material MSR y, en la sección *Shadows*, cambiar el valor de *Intensity* (0,2). Un valor algo inferior (0,10, 0,15) hará la sombra más oscura; un valor algo superior (0,4, 0,5) la hará más clara.

Para ajustar el gradiente de la sombra, si fuera necesario, editar el material MSR y, en la sección *Ambient occlusion*, reducir o aumentar el valor de *AO shadow strength* (que está predeterminado al máximo, negro). Si aparecieran defectos aumentar el número de muestras (*AO samples*).

§ § §

El ejemplo siguiente ilustra el caso en que no hay sombras debidas a la iluminación directa sino sombras difusas debidas solo a iluminación indirecta. Utilizaré un ejemplo muy simple pues el procedimiento es similar, con las diferencias que se indican. Partimos de una imagen como la de la figura 5.56 (*a*). Seguir el procedimiento de igualación de cámara que ya hemos visto y crear un prisma flotante sobre un plano para llegar a un primer resultado como el de la segunda imagen de la figura 5.56 (*b*).

A diferencia del caso anterior, ahora no nos interesa crear un sistema de luz diurna. Nos bastará con luces de área que arrojen sombras difusas y que permiten mayor control del resultado. Según los diferentes casos habrá que seguir un proceso de prueba y error a partir de un análisis aproximado de la escena. En el ejemplo he utilizado dos luces de área muy amplias, aproximadamente el doble que el tamaño de la mesa central y situadas a ambos lados. También he utilizado una tercera luz dirigida exclusivamente al lateral derecho de la mesa para evitar que quede excesivamente oscura.

Una vez creadas las luces, configurar el cálculo con valores bajos de *Final gather* para hacer las primeras comprobaciones. La figura final de este ejemplo se ha obtenido con *Final gather* con calidad media y dos rebotes.

A partir de ahí, crear, como en el ejemplo anterior, un material *mr MSR* y asignarlo al plano de base. Arrastrar, desde el cuadro *Environment*, el mapa de fondo sobre la casilla de mapa de los parámetros de *Camera mapped background*. Editar el mapa y comprobar que, en *Coordinates*, está seleccionada la opción *Environment* en lugar de *Texture* y que el *Mapping* es a *Screen*.

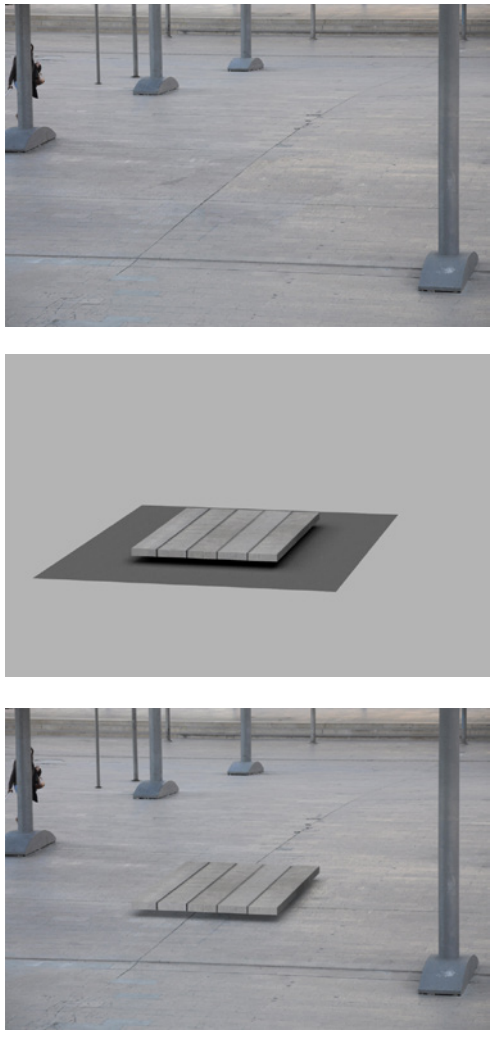


Figura 5.56 Integración de modelo y fondo. Ejemplo 2. Iluminación indirecta: a*) Imagen de fondo;* b*) Modelo aislado;* c*) Modelo integrado.*

Se puede simular la iluminación indirecta activando, desde la sección *Indirect illumination*, la opción "Receive Indirect Illumination". Sin embargo, en un caso como este se obtendrán mejores resultados, más controlados, activando en su lugar *Ambient occlusion*. Esto nos permite ajustar la intensidad de la sombra que, en este caso, se ha suavizado dando al parámetro *AO shadow strength* un valor gris medio (hsv 0, 0, 0,45) en lugar del predeterminado (negro).

§ § §

El tercer ejemplo de este grupo ilustra el caso en que nos interesa que aparezcan reflejos sobre el fondo como si fueran debidos a los objetos virtuales del modelo. También en este caso el procedimiento es similar con las variantes que se indican y el ejemplo que he escogido para ilustrarlo también es muy sencillo, para que sea fácil de reproducir.

El primer paso es, como en los anteriores, crear una cámara, ajustarla a la vista y luego crear un prisma y un plano como objetos auxiliares para las primeras comprobaciones. Como en este caso la iluminación es interior, habrá que crear luces que simulen aproximadamente las de la escena.

Por último, crear un material *mr MSR*, arrastrar el fondo a la casilla de *Camera background map*. Editarlo y cambiar sus opciones a *Environment*/*Screen*. En la sección *Reflections*, activar "Receive Reflection". *Render* para comprobar que el resultado inicial es correcto y luego volver a editar esta sección y ajustar los valores para que el reflejo se adecúe a la escena. En la figura 5.57, la configuración es: *Glossiness* 0,65, *Glossy samples* 16, resto de valores sin modificar.

§ § §

El último ejemplo de este grupo es similar al que ya hemos visto en la sección sobre reflejos, pero en este caso lo ampliaremos con

Figura 5.57 Integración de modelo y fondo. Ejemplo 3. Reflejos: a) *Imagen de fondo;* b) *Modelo situado sobre el fondo;* c) *Modelo integrado.*

los *shaders* que estamos utilizando y que amplían las posibilidades. Se trata de simular reflejos no sobre el propio fondo, como en el ejemplo anterior, sino sobre los objetos del modelo.

La simulación de reflejos sobre el objeto se puede solucionar de un modo más efectivo si se utilizan mapas de entorno esféricos. El problema es que estos mapas deben crearse con métodos especiales para que tengan la suficiente calidad y a menudo se utilizan con formato HDR para que la calidad sea aún mayor. Me remito de nuevo al libro de simulación de la iluminación donde se tratan más extensamente estos temas.

Otra alternativa aún más eficaz, probablemente la mejor desde el punto de vista de la calidad conseguida, y que veremos en el apartado siguiente es utilizar un bola reflectante. El inconveniente de este método es que la preparación previa es más laboriosa.

Volviendo a lo anterior. Dado que es difícil conseguir una resolución adecuada para los mapas esféricos, un recurso habitual y bastante efectivo (que también se amplía en el libro sobre simulación de la iluminación) es utilizar el *mr Environment / Background switcher* con dos mapas, uno para el fondo, normal y proyectado como tipo *Screen*, y otro para los reflejos, proyectado como tipo *Spherical*. Este último puede ser de una resolución mucho menor y también interesa que quede algo desenfocado pues esto evita problemas de *aliasing*. En el ejemplo que sigue utilizaremos dos mapas que corresponden a diferentes vistas pero también se puede usar el mismo con variantes.

Partimos de las dos imágenes que se muestran en la figura 5.58 (*a*) y (*b*). La primera es una imagen normal que utilizaremos como en los ejemplos anteriores para montar la cámara deduciendo su ángulo, etc. La segunda imagen es un minipanorama que he montado (con Hugin) a partir de tres imágenes tomadas en dirección opuesta a la anterior. Esta segunda imagen, a la que me referiré como “panorama”, se utilizará para los reflejos. El procedimiento es como sigue.

1 Comenzar como en los ejemplos anteriores. Es decir, ajustar el tamaño de salida, crear una cámara, ajustar su ángulo horizontal para que coincida con el de la fotografía, luego asignar al visor de cámara el mapa “fondo” y activar los marcos del visor de cámara y visualizar el fondo.
2 Crear un objeto prismático que nos ayude a ver las sombras y un plano situado bajo este objeto auxiliar. Asignarles materiales

Figura 5.58 Integración de modelo y fondo. Ejemplo 4: a) *Imagen de fondo;* b) *Imagen panorámica de contrafondo;* c) *Montaje previo con prisma y plano auxiliar;* d) *Montaje previo con material* Matte / Shadow / Reflection *asignado al plano.*

mates, simples, con colores captados de las superficies cercanas.

3 Crear un sistema de luz diurna y ajustarlo hasta llegar a un resultado como el de la figura 5.58 (*c*) en la que se ha utilizado un sistema de luz diurna de posición manual, con *mr Sun* y *mr Sky* con sus valores predeterminados. Ajustar también el control de exposición (en la figura adjunta está en EV 15,0 con *Unitless* 75.000) y marcar la opción "Process background and environment maps".

4 Por último, substituir el material del plano base por un material *mr MSR* (*Matte* / *Shadow* / *Reflection*). Copiar o arrastrar el mapa de fondo a la casilla de *Camera mapped background* de este material. Repetir el *render* y, si es necesario, ajustar valores de *Ambient occlusion* para que el plano se integre bien con el fondo.

Hasta aquí el procedimiento es el mismo que hemos seguido en los ejemplos anteriores. En lo que sigue utilizaremos un *Switcher* que nos permitirá utilizar dos mapas, uno para el fondo y otro para los reflejos.

5 Ir al panel *Environment* (tecla 8). Escoger como mapa de fondo *mr Environment / Background switcher*.

6 Abrir el editor de materiales. Arrastrar el mapa anterior, *Environment / Background switcher*, como instancia, sobre un visor del editor de materiales. Este mapa es muy sencillo y solo incluye dos entradas que se muestran en el esquema de la figura 5.59.
Asignar a la primera, *Background*, el mapa que estábamos utilizando como fondo.
Asignar a la segunda, *Environment*, el mapa de bits "panorama". Editarlo y, en *Coordinates*, cambiar la opción "Texture" por "Environment" y escoger como tipo de proyección "Cylindrical".

La estructura final de los *shaders* deberá ser como la del esquema citado de la figura 5.59 (*a*).

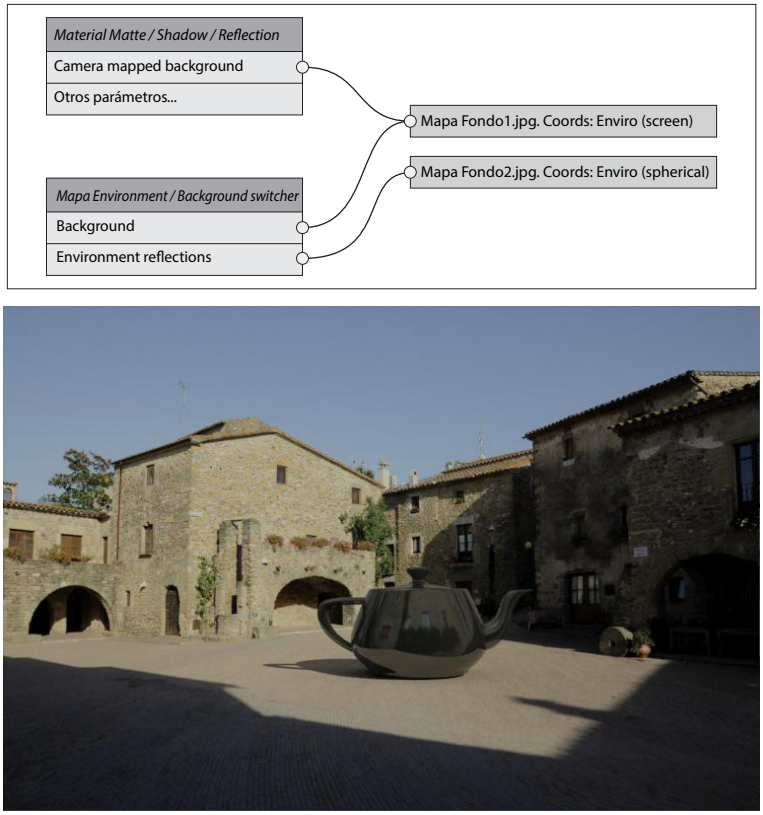


Figura 5.59 Integración de modelo y fondo. Ejemplo 4: a*) Esquema de la relación entre los* shaders *principales utilizados;* b*) Resultado final.*

7 Por último, asignar al objeto un material *Arch&Design* oscuro y con la reflexión al máximo para comprobar las reflexiones. Para que el panorama se visualice con más claridad crear un esfera como objeto de pruebas (la imagen se distorsionará en los polos) y luego substituirlo por el objeto que sea (una tetera en el ejemplo de la figura 5.59).

Integración de reflejos con imágenes de fondo. Bola cromada

Para llevar a cabo este ejercicio se necesita una bola cromada y seguir una serie de procedimientos que se explican con más detalle en el libro sobre simulación de la iluminación (incluyendo cómo conseguir una bola cromada de características adecuadas). A pesar de que estos procedimientos son más trabajosos, el resultado capta de un modo prácticamente perfecto la iluminación del entorno y los reflejos que resultarían en un objeto curvo muy reflectante.

Límitándome a lo principal el procedimiento resumido consiste en lo siguiente.

1 Tomar dos fotografías de la escena real: Una fotografía del entorno que se quiere simular, con la bola situada en una posición que, idealmente, debería coincidir con el centro del modelo virtual que se quiere colocar en el sitio. Pero, en la práctica, basta con que esté más o menos cerca de esa posición

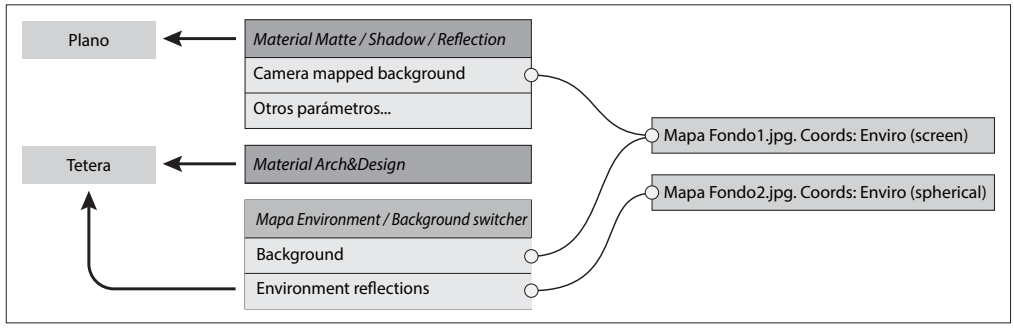


Figura 5.60 Integración de modelo y fondo con bola cromada: a) *Esquema de aplicación de materiales y mapas;* b) *Fotografía original;* c) *Fotografía con la bola cromada;* d) *Bola recortada;* e) *Resultado final: la tetera es el modelo virtual con los reflejos correspondientes a la bola cromada.*

para que refleje lo que tiene alrededor, que es el objetivo que se persigue. Guardarla con un nombre tal como *fondoBolaCromada.jpg*. Y otra fotografía sin la bola del entorno. Guardar esta otra imagen con un nombre tal como *fondoGeneral.jpg*. Véase la figura 5.60 (*b*, *c*).

2 Recortar la imagen *fondoBolaCromada.jpg*. Esto se puede hacer en HDR Shop o en Photoshop. En cualquiera de los dos casos, convendrá editar la imagen resultante, comprobar que las dimensiones de altura y anchura son iguales y que el marco es tangente a los extremos. También convendrá rellenar el fondo con un color gris medio para comprobar mejor que el recorte es adecuado. Véase la figura 5.60 (*d*).

Contando con estas dos imágenes y con el modelo que queremos situar en este entorno, y que puede ser tan complicado como se necesite, aunque en este ejemplo sea muy sencillo para que resulte fácil de reproducir y explicar, los pasos siguientes son:

1 Desde 3ds Max, crear una escena con las características siguientes:
Un plano de base.
Un objeto o un conjunto de objetos (la tetera del ejemplo) situados sobre este plano.
Una cámara que apunte al modelo con un ángulo similar al de la vista que hemos obtenido. El procedimiento de ajuste de la cámara, en caso de que sea necesario un ajuste preciso, es el mismo que ya hemos visto en el ejemplo de la escultura de Oteiza en la plaza frente al MACBA.
Una luz *mr Area spot* de área amplia para que arroje sombras difusas de la tetera sobre el plano y otra luz en dirección opuesta que suavice estas sombras buscando que sean similares a las de la imagen de fondo.
2 Abrir el editor de materiales. Crear los siguientes mapas y materiales:
Un mapa, "fondo1", con la imagen *fondoGeneral.jpg* como mapa de tipo *Bitmap*
Un mapa, "fondo2", con la imagen *fondoBolaCromada.jpg* como mapa de tipo *Environment probe / Chrome ball*.
Un mapa de tipo EBS (*Environment* / *Background switcher*) con el mapa "fondo1" arrastrado como instancia a *Background* y el mapa "fondo2" arrastrado como instancia a *Environment* / *Reflections*. Editar ambos mapas y, en *Coordinates*, hacer que el primero sea de tipo *Environment* / *Screen* y el segundo de tipo *Environment* / *Spherical*. Luego arrastrar este mapa a la entrada de mapa del panel de *Ennvironment* (menú *Rendering/Environement* o tecla 8).
Un material de tipo MSR (*Matte*/*Shadow*/*Reflection*). Una vez creado, arrastrar (como instancia) el mapa "fondo1" al botón junto a *Camera mapped background*. Activar las secciones *Shadows*, *Ambient occlusion* y *Reflections* (con un valor bajo de *glossiness*, en torno a 0,1). Asignar este material al plano base.
Un material de tipo *Arch&Design*, de color gris claro y con la reflectancia al máximo (ajustar también la curva BRDF, para 0º, 0,5 aproximadamente para que el reflejo se incremente).
Véase el esquema de la figura 5.60 (*a*) que resume lo principal de estas asignaciones.
3 Configurar la salida con FG y 2 rebotes. Hacer una prueba con poca resolución y calidad baja y, cuando el resultado sea correcto, aumentar la resolución y la calidad. El resultado final deberá ser como el de la figura 5.60 (*e*).

Pueden ajustarse los parámetros específicos del mapa de la bola cromada para, por ejemplo, aumentar o disminuir la intensidad (subiendo o bajando el valor del multiplicador) y otros ajustes. Pero en general no será necesario.

Integración de sombras con fondos uniformes

En muchos casos interesa utilizar un fondo uniforme para resaltar un objeto. Los fotó-

grafos profesionales consiguen este efecto mediante telas o cartones pintados y una iluminación muy dispersa, que ilumine de modo uniforme el conjunto, con algunas luces dirigidas a puntos clave del objeto.

En los dos ejemplos que siguen replicaremos este proceso mediante tres tipos de fondo: uno de color uniforme, otro degradado y otro con una textura global.

Para crear imágenes similares seguir los pasos siguientes.

1 Crear una escena parecida a la de la figura 5.61. La escena consiste en un objeto sobre un plano (un cáliz diseñado por Desny, hacia 1925), una luz principal de tipo *mr Area spot*, situada a la izquierda del objeto, con un área muy grande para que cree sombras muy dispersas, y otra luz secundaria, en el lado opuesto, con las sombras desactivadas y más o menos la mitad de intensidad para que suavice las sombras de la otra luz. La contribución al reflejo de estas luces está desactivada. Los reflejos visibles en el cáliz son debidos a cinco planos con autoiluminación situados frente al cáliz. El plano tiene asignado provisionalmente un material *Arch&Design* mate de color gris. con valores por defecto. El objeto tiene asignado un material *Arch&Design* de tipo metal, brillante. El cálculo de iluminación se lleva a cabo con *Final gather* (con poca calidad para las pruebas iniciales y mayor calidad para las finales).
2 Abrir el editor de materiales. Crear tres mapas:
   a) "fondo Uniforme", de tipo *Color correction*. Dar a este mapa un color simple. Otra alternativa para usar un color como un mapa es hacerlo de tipo *Gradient ramp* pero con colores iguales en los extremos.
   b) "fondo Degradado", de tipo *Gradient ramp*. Editarlo y crear un degradado entre dos colores similares, uno claro y otro más oscuro. Comprobar que el *mapping* es de tipo *Environment / Screen.*
   c) "fondo Textura", de tipo mapa de bits, con una textura que habrá que haber creado previamente en Photoshop o Gimp o sacando una fotografía adecuada. Comprobar que el *mapping* es de tipo *Environment / Screen.*

Si quisiéramos controlar los reflejos de modo independiente tendríamos que haber utilizado un *Environment* / *Background switcher* para contar con una entrada independiente para el reflejo. Pero en este caso no será necesario.

3 *Render*. Comprobar y ajustar el resultado preliminar. Probablemente habrá que mover y ajustar la intensidad de los paneles que se reflejan en el objeto. También puede interesar, en los parámetros del mapa de bits de entorno, aumentar un poco (de 0,0 a 0,01 o 0,02 será notorio) el valor de *Blur offset* para desenfocar algo el reflejo. Cuando el resultado sea satisfactorio habrá que hacer desaparecer el plano de base. Para ello, continuar como sigue.
4 Asignar el primer mapa de fondo como mapa de entorno: copiarlo y pegarlo como instancia (o arrastrarlo), desde el editor de materiales al panel de *Environment*.
5 En el editor de materiales escoger un material MSR (*Matte* / *Shadow* / *Reflection*). Copiar uno de los mapas de fondo que hemos creado y pegarlo como instancia (o arrastrarlo) en la entrada de *Camera mapped background* del material MSR. Así, el fondo procesado por este material será el mismo que el asignado al fondo. Asignar el material MSR al plano.
6 *Render*. El resultado deberá ser similar al de las imágenes de la figura 5.61, al substituir uno de los mapas por otros mapas para obtener variantes alternativas.

5.10 Simulación no realista. Contornos con mapas

Introducción. *Toon Shading*

En la literatura anglosajona se denomina generalmente *Toon shading* a las técnicas de

Figura 5.61 Integración de modelo y fondo con fondos uniformes: a) *Resultado con fondo de color uniforme;* b) *Con fondo degradado;* c) *Con mapa de bits de manchas.*

"representación no realista" o NPR por sus siglas en inglés (*Non Photorealistic Rendering*) que se utilizan, como un fin en sí mismo o para desarrollar una representación estilizada, más abstracta, de un escenario virtual. También se encuentra el término *Cel shading*, que deriva de las hojas de acetato denominadas *cels* que se utilizaban en animación tradicional para calcar los personajes, introduciendo variantes entre secuencias. O el propio término *Cartoon shading*. Pero el primer término es más utilizado.

Este término, *toon*, parece que se popularizó a partir de la película *Who framed Rogger Rabbit* (1988) basada a su vez en la novela de Gary K. Wolf, *Who Censored Roger Rabbit?* (1981) donde se utilizaba para referirse a un *cartoon character* (personaje de película de dibujos animados) que, en la novela, habitaba un mundo humano pero no era humano. Por lo que parece, de *cartoon* se derivó *toon*. El uso se popularizó aún más a partir del juego de rol de Disney *Toontown Online* en el que los habitantes de Toontown se denominan *toons* (los jugadores pueden crear nuevos *toons* eligiendo la especie, el género y el color o colores).

Con independencia de todo esto, es un hecho que, además de su uso en el contexto de los dibujos animados y de los cómics, la técnica tiene un considerable interés en el campo de la arquitectura y el diseño. Pues el realismo, como bien saben los arquitectos, puede ser un medio de comunicar ciertas características del proyecto. Pero también puede ser un estorbo, pues nuestra percepción del mundo de modo alguno está necesariamente mediada por el "fotorrealismo". Las representaciones conceptuales, esquemáticas, pueden ser tanto o más informativas que las realistas. Y hay un término medio entre la fidelidad fotográfica y el esquema abstracto, donde siempre ha brillado el dibujo, la representación basada exclusivamente en los contorno, desde los inicios de lo que conocemos como representación gráfica. El dibujo representa la forma, desnuda de cualidades secundarias, y la ofrece a la vista como se ofrecería al tacto, resaltando los rasgos característicos que facilitan el reconocimiento.

De ahí el interés de estas técnicas que son relativamente sencillas aunque presentan algunas peculiaridades técnicas que conviene conocer.

Shaders disponibles

Con 3ds Max se utiliza el material *Ink'n Paint*. Con mental ray hay toda una serie de *shaders* que permiten crear este efecto, equivalentes al anterior pero con más opciones, y que son utilizados por los programas anfitriones de mental ray, es decir 3ds Max, Maya y Soft Image. En los siguientes apartados me centraré en estos *shaders* que dan una amplia gama de opciones y son una buena introducción a este tema.

En VRay está disponible como un efecto atmosférico (un *plug-in*) denominado *VrayToon*. ¿Por qué un efecto atmosférico? Según la página oficial de Chaos Group, porque la implementación es muy simple, porque funciona con varios tipos de geometría utilizada por VRay, incluyendo casos especiales como *VRayFur* y *VRayProxy*, y porque funciona con los varios tipos de cámaras utilizados por VRay. Tabién hay un *shader*, *VRayEdges* que permite generar directamente representaciones alámbricas, una técnica que también veremos más adelante. En Blender se encuentra un *Edge* (*Toon*) *Rendering* que funciona como un efecto de postproducción. En Cinema 4D también se encuentra un módulo, *Sketch and Toon*, con varias opciones de trabajo que dan resultados similares a los anteriores.

Si se quiere probar el funcionamiento básico del material *Ink'n Paint* de 3ds Max crear dos o tres objetos en la escena. Luego, desde el editor de materiales crear un material de tipo *Ink'n Paint* y aplicarlo a los objetos. Utilizar los controles básicos de la sección *Paint controls*, para obtener una representación en blanco y negro, con líneas resaltadas. Por ejemplo, en *Paint controls* / *Paint*, marcar la casilla *Lighted* y el color en blanco. Y en *Ink controls* / *Ink width*, ajustar el valor *Min* a 2,0

o algo menos o algo más en función de lo que interese.

Los resultados serán similares a los obtenidos con los *shaders* de mental ray que veremos a continuación y que incluyen más opciones.

Contornos simples

El uso más corriente y probablemente de mayor interés como alternativa a las representaciones realistas es generar imágenes que sean como dibujos. Como esto implica una detección automática de bordes hay variantes más o menos complicadas, según la propia complejidad de la escena.

a) Contornos simples con líneas rectas

Para crear una imagen con contornos, utilizando mental ray, es necesario, en cualquier caso, activar dos series de órdenes: *a*) una dirigida al editor de materiales, para aplicar un material específico, que simule contornos, a todos o parte de los materiales de la escena, *b*) otra dirigida al motor de *render* para que utilice procedimientos específicos de representación y haga determinados ajustes a partir de configuraciones que son, hasta cierto punto, modificables por el usuario.

Cuando se activa el motor de *render* para que aplique los procedimientos de contorno nos encontraremos con que se cargan automáticamente tres *shaders* predeterminados. La función de estos *shaders*, que se procesan en el mismo orden en que se presentan es: *a*) Rastrear la imagen para buscar aristas y zonas de contraste. Esto corre a cargo de primero, *Contour contrast*; *b*) Almacenar los contornos así encontrados en un registro especial. Esto corre a cargo del segundo, *Contour store*, que no tiene parámetros; *c*) Procesar estos contornos y componerlos o más bien superponerlos, con la imagen de salida. Esto corre a cargo del tercero que, a diferencia de los anteriores, puede ser uno de tres: *Contour composite* (predeterminado), *Contour only* y *Contour PS* (salida vectorial).

Para editar los parámetros de estos *shaders* será necesario ligarlos a un material del editor de materiales. Y esto puede hacerse para diferentes objetos de la escena o para todos los objetos, sea asignando un mismo material a toda la escena, sea utilizando el recurso *material override*, que se encuentra en *Render setup* y que ya hemos visto.

Los dos ejemplos que siguen ilustran estos procedimientos básicos. Como veremos más adelante hay otras variantes de interés.

Para aplicar un *shader* de contornos sencillo a toda la escena hacer lo siguiente:

1 Crear una escena simple como la de la figura 5.62.
2 Ir a *Render setup* / *Renderer* / *Camera effects* / *Contours*, marcar la casilla *Enable* y configurar así los mapas (los dos primeros se añadirán automáticamente):
Contour contrast: “Contour Contrast Function Levels”.
Contour store: “Contour Store Function”.
Contour output: Cambiarlo por “Contour Only”.
3 Desde el editor de materiales crear un material de tipo *Arch&Design* o *Standard* (se requiere un color de base aunque luego no se visualice). Ir a la sección *Mental ray connection*, que se encuentra al final de estos dos tipos de materiales y, en *Advanced shaders*, pulsar el botón junto a *Contour*.

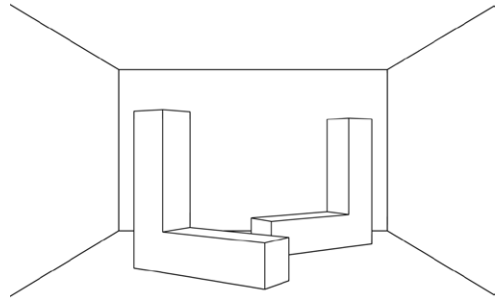


Figura 5.62 Resultado de aplicar un contorno simple a un recinto con dos objetos.

Aparecerá una larga lista de tipos de contornos que comento muy brevemente (ampliar información en la ayuda del programa): *Combi*: es una combinación de los *shaders Depth fade*, *Layer thinner* y *Width from light* que se describen más adelante. *Curvature*: crea contornos cuyo espesor depende del ángulo entre normales de las caras adyacentes. *Depth fade*: crea contornos cuyo grosor depende de la distancia. *Factor color*: usa el color del material para los contornos. *Layer thinner*: crea contornos cuyo grosor depende de los valores encontrados por la función de contraste. *Simple*: crea contornos del mismo grosor. *Width from color*: crea contornos que varían según el color del material, cuanto más claro más fino y viceversa. *Width from light*: crea contornos que varían según la dirección de la luz. *Width from light direction*: crea contornos que varían según la dirección de una luz virtual definida por el *shader*.

4 Escoger el más sencillo, *Contour simple*. Este tipo solo tiene dos parámetros, el color y el grosor. El grosor viene dado en porcentaje de la imagen de salida por lo que deberemos ajustarlo en función de este tamaño. Por el momento dejar su valor predeterminado.
5 Arrastrar este material, como instancia, a *Render setup / Processing / Material override*.
6 *Render*.

El resultado será suficiente como primera comprobación pero las líneas serán demasiado gruesas. Como ya he dicho, el grosor depende de la resolución. Si, por ejemplo, queremos imprimir esta imagen en una anchura de 6,4 cm, como las de las columnas de este libro, nos interesará una anchura de salida de 768 píxeles (6,4 x 120 ppc). Y si queremos que las líneas tengan un grosor de 1 píxel esto supondrá un porcentaje de 0,15 (100 x 1/768).

6 Modificar el valor de anchura del *Contour simple* a 0,15 y repetir el *render*. Ahora el resultado será como el de la primera figura de esta sección, la 5.62. Tener en cuenta que el valor tampoco debe ser demasiado bajo pues se producirán desigualdades.

b) Contornos simples con líneas curvas

Una escena como la anterior no supone ninguna dificultad pues el primer *shader*, el *Contour contrast*, encuentra fácilmente las aristas que debe almacenar en el segundo. Sin embargo, cuando nos encontramos con un objeto con curvas el procesamiento no es tan obvio pues pueden aparecer zonas ambiguas, como ocurre en la escena de la figura 5.63 que representa una silla, diseñada por Mies Van der Rohe, situada en una esquina de un recinto simple.

1 Crear una escena que incluya objetos con curvas como los de la figura 5.63.
2 Asignar a los objetos un material *Arch&Design* o *Standard*. Ir como antes a la sección *mental ray connection* y asignar a *Contour* un *shader* de tipo *Curvature*. Editar el *shader*. Dejar el color en negro pero reducir los valores de *Min width* y *Max width* a 0,01 % y 0,1 % respectivamente (probar otros valores para comprobar las diferencias).
2 Ir a *Render setup / Renderer / Camera effects / Contours*, marcar la casilla *Enable* y configurar los mapas como antes (*Contour contrast* y *Contour store* y *Contour only*).
3 *Render*. El resultado, con los valores anteriores, será bastante correcto pero se perderán algunas líneas de curvatura interna.
4 Editar el primer *shader*, *Contour contrast* (arrastrarlo desde *Render setup* al editor de materiales) y reducir drásticamente el valor de *Z step threshold* y *Angle step threshold*. Con valores muy bajos tardará más tiempo y el resultado será que se creará un número excesivo de líneas. Ir probando valores hasta encontrar el adecuado. En el ejemplo de la figura 5.63,

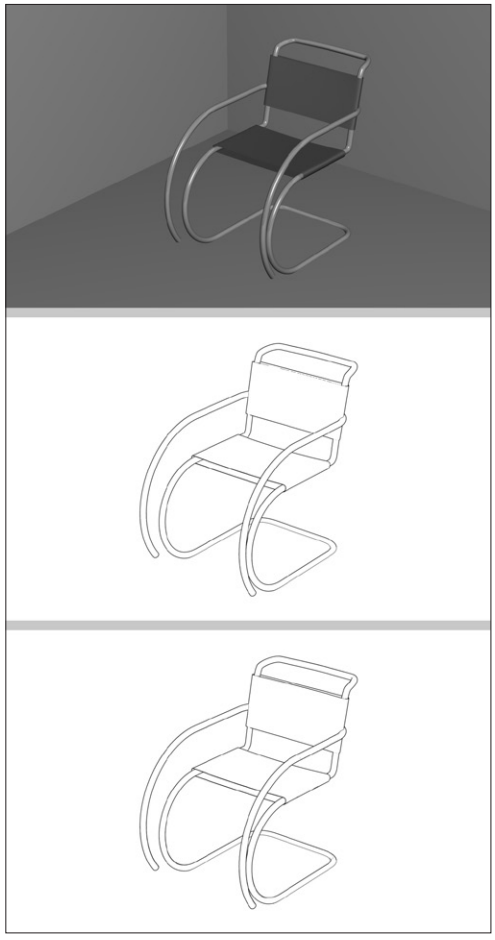


Figura 5.63 Resultado de aplicar un contorno simple a un objeto con líneas curvas: a) *Aspecto original;* b) *Contorno de tipo* Curvature *con parámetros 0.01 y 0.1 (*Min, Max width*); c) Ajuste de la del contraste con los valores indicados en el texto.*

con unas dimensiones de anchura de la silla de unos 50 cm, los valores que he utilizado han sido 0,1 para *Z step* y 5,0 para *Angle step*.

c) Contornos variables con la distancia

Otro *shader* que puede resultar útil en algún caso es *Contour depth fade* que es como el primero que hemos utilizado, *Contour simple* pero haciendo que las líneas más lejanas se representen de un modo más ligero (líneas más finas y de color menos negro).

El procedimiento es similar aunque habrá que hacer algún ajuste para que funcione correctamente. Para probarlo, crear una escena con objetos a diferentes distancias de la cámara y anotar las distancias más cercanas y más lejanas porque afectarán a la configuración. Para la figura 5.64 se ha utilizado una escena simple en la que el objeto más cercano está a unos 7 metros de la cámara y el más lejano a unos 70. A partir de ahí:

1 Asignar a toda la escena un material *Arch&Design* o *Standard* (aplicándolo a todos los objetos o, mejor, como *Material override* desde *Render setup*) y, desde la sección *mental ray connection* del material, asignar a *Contour* un *shader* de tipo *Depth fade*. Editar el *shader*. Asignar a *Near Z* el primer valor de distancia que hemos tomado o dejarlo en 0,0 para que el degradado sea más suave, dejar *Near color* en negro y *Near width* en unos 0,25 %. Asignar a *Far Z* un valor superior al segundo valor de distancia que hemos tomado (130 en el ejemplo) para que el degradado sea más suave; a *Far color* un gris muy claro (0,75 sobre 1,0 en el ejemplo y a *Far width* 0,01 %.
2 Ir a *Render setup* / *Renderer* / *Camera effects* / *Contours*, marcar la casilla *Enable* y configurar los mapas como antes (*Contour contrast* y *Contour store* y *Contour only*).
3 Desactivar el cálculo por FG para que sea más rápido pues no se necesita. Aumentar la resolución, pues afecta a la precisión de las líneas (2048 de alto en el ejemplo).
4 *Render*. El resultado deberá ser similar al de la figura 5.64.

Contornos y colores planos

Los contornos pueden combinarse con los colores dados por los materiales asignados a los objetos. Pero esto tiene escaso interés pues lo que pretendemos es dar un resultado

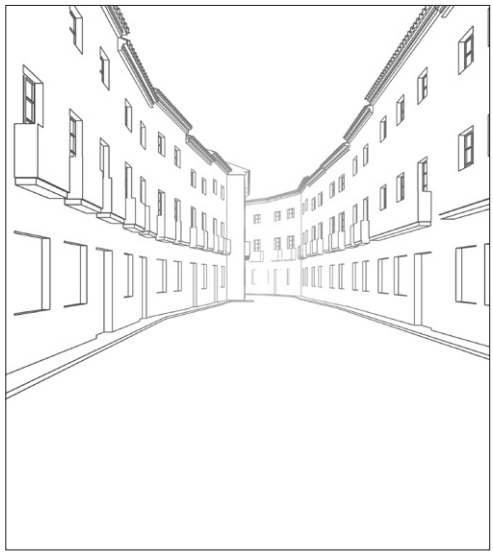


Figura 5.64 Contornos variables en profundidad. Parámetros de Contour depth fade *(z, color, %).* Near*: 0,0, negro, 0,25.* Far*: 130, gris claro, 0,01.*

simplificado y el realismo de los materiales interfiríra con esta intención. Por tanto, lo que habrá que hacer es simplificar los degradados y sombreados de estos materiales para reforzar esta intención.

Para seguir este proceso de simplificación paso a paso comenzaremos por añadir contornos a los materiales sin simplificar y luego iremos introduciendo simplificaciones sucesivas.

1 Crear una escena como la de la figura 5.65 con diferentes tipos de objetos, rectos y curvos.
2 Asignar materiales corrientes, *Arch&Design* o *Standard* a los objetos de la escena.
3 En la sección *mental ray connection*, al final, en el grupo *Advanced shaders*, asignar a *Contours* un *shader* de tipo *Contour curvature*. Este *shader* servirá tanto para superficies rectas como curvas. También podríamos haber asignado un *Contour simple* pues no habrá gran diferencia en este caso. Si ahora hiciéramos un *render* no pasaría nada pues se necesita, en cualquier caso, configurar la salida a través del *Render Setup*, como ya hemos visto.
4 Ir a *Render setup / Renderer / Camera Effects / Contours*, marcar la casilla *Enable* y dejar los *shaders* que se asignan por defecto: *Contrast*, *Store* y *Composite*. Este último es necesario para que se mantengan los colores de los materiales. Si utilizáramos, como antes, *Contour only*, estos colores se substituirían por el color dado por el parámetro *Background* de este último *shader*.
5 *Render*. Así se obtendría un resultado como el de la figura 5.65 después de ajustar los grosores del contorno. En la figura se ha utilizado 0,01, 0,25 (Min%, Max%).

Para simplificar los colores el mejor método es utilizar un mapa *Falloff* con el mapa de curvas manipulado para forzar los gradientes a que se desplieguen en pasos uniformes. Este método es independiente del uso de *shaders* de contornos, pero es evidente que implica una simplificación radical que solo tendrá sentido para generar resultados no realistas por lo que se combina bien con las técnicas de contorneado.

Se comprenderá mejor el funcionamiento de esta técnica si se analiza de modo independiente, con un objeto tal como una esfera antes de continuar. Esto es lo que se ha hecho en los ejemplos que se ilustran en la figura 5.67. En estos ejemplos se ha partido de una escena elemental: una esfera sobre un plano iluminada por una luz directa.
A esta esfera se le ha asignado un material *Standard* con dos propiedades modificadas: la *Self-Illumination*, aumentada al máximo (100) y un mapa *Falloff* asignado a *Diffuse*. También se podría haber utilizado un material de tipo *Arch&Design* o *mental ray* pero es más sencillo controlar la autoiluminación con el material *Standard*.

El mapa *Falloff* está configurado como sigue: *Type* > *Shadow* / *Light*. *Colores* > dos tonos similares, el de *Shade* algo más oscuro que el de *Lit*. *Mix Curve* > modificada tal como se muestra en la figura 5.67 para forzar una

transición no gradual de tonos. De este modo, las partes de la superficie orientadas hacia la luz recibirán el color de *Lit* y las partes en sombra el color de *Shade*. Y la transición entre ellas, en lugar de ser gradual, será más o menos brusca.

Como decía, esto es independiente del uso de *shaders* de contornos y, por tanto, podríamos obtener este resultado sin necesidad de ningún recurso especial. Para complementarlo con contornos, todo lo que tenemos que hacer es seguir el mismo procedimiento indicado más arriba, es decir, asignar un *shader* de contornos al material desde la sección *mental ray connection* y, en paralelo, activar los contornos desde *Render setup*.

De este modo se ha obtenido un resultado como el de la anterior figura de este grupo, la 5.66. También puede combinarse este efecto con *Ambient occlusion*, entre otras alternativas interesantes.

Representación alámbrica de la geometría interna

Otra posibilidad que puede interesar en determinados casos es obtener una representación de un objeto que muestre su geometría interna, las líneas que definen sus caras poligonales. Es decir, lo que sería equivalente a una representación alámbrica pero con un mínimo control sobre los colores y grosor de las líneas, y sobre el color de las caras, algo que no podríamos conseguir con una captura de pantalla.

Para ello se podrían utilizar las técnicas anteriores activando, en el *shader Contrast function levels* la opción *Face*. Sin embargo

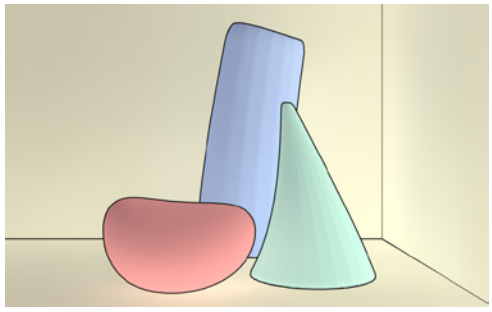


Figura 5.65 Contornos con colores planos (color local más autoiluminación).

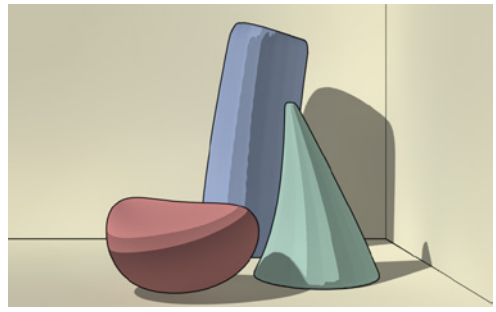


Figura 5.66 Contornos con colores planos (color local con mapa falloff*).*

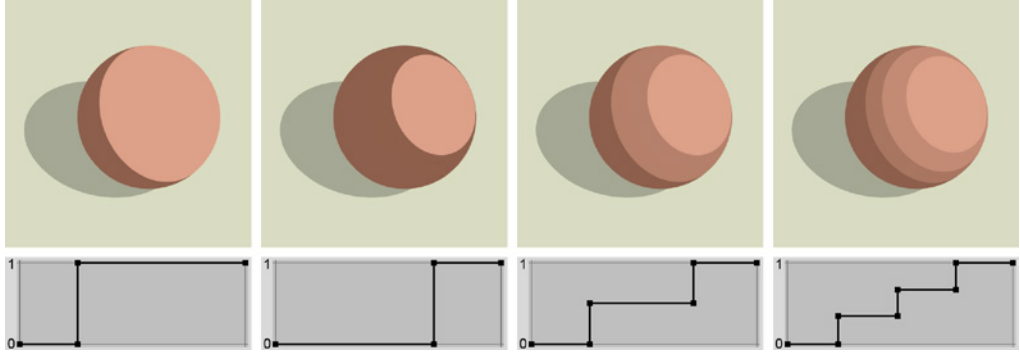


Figura 5.67 Uso de Falloff *para crear gradientes. Esfera con* Falloff *de tipo* Shadow / Light.
El diagrama inferior muestra diferentes configuraciones de Mix Curve *que dan lugar a transiciones más o menos graduadas.*

es probable que esto dé resultados indeseados pues se repesentarán también los triángulos internos y, por otra parte, hay otros modos más sencillos de llegar al mismo resultado.

De hecho, hay varias maneras de conseguirlo. Resumo brevemente dos métodos que tienen cierto interés intrínseco y, con más detalle, un tercer método que parece más efectivo para este objetivo concreto.

El primero, aunque implica derivaciones que pueden resultar útiles en casos muy concretos, tiene el inconveniente de que no muestra los materiales ya asignados. El procedimiento consiste en lo siguiente: *a*) Aplicar un mapa UVW al objeto. Escoger *Face* como tipo de *Mapping*; *b*) Aplicarle un material *Standard* o *Arch&Design* y añadir a *Diffuse* un mapa de tipo *Gradient ramp*; *c*) Configurar el *Gradient ramp* como tipo *Box*, en blanco y negro. Ajustar los *flags* del gradiente más o menos como sigue (hacer *BDR*/*Properties* sobre el *flag* para introducir un valor numérico si resulta más cómodo): Pos = 0 > blanco, Pos = 95 > blanco, Pos = 96 > negro, Pos = 100 > negro. Esto creará, como se puede comprobar en el editor de materiales, un cuadrado blanco con un fondo negro. Si la distancia entre los dos últimos *flags* se reduce, la línea negra será más estrecha y si se aumenta, más ancha. Al aplicar un material con este mapa al objeto, cada cara recibirá este patrón y el resultado será que las aristas internas se representarán con líneas negras.

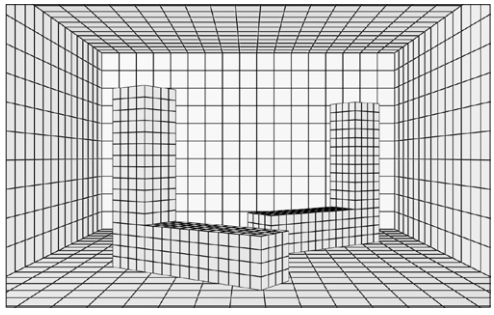


Figura 5.68 Representación alámbrica de la geometría interna de una escena.

El segundo método consistiría en extraer todas las aristas del objeto (aplicarle un modificador *Edit poly* si no fuera una malla poligonal), luego entrar en modo subobjeto arista y hacer Ctrl+A para seleccionarlas todas o seleccionar manualmente las que interese). Editar estas aristas extraídas como *splines* y hacer que sean "renderizables" (sección *Rendering* de los parámetros de edición de *splines*) ajustando su grosor y otras características. La ventaja de este método es que podemos hacer que una parte del modelo se represente con aristas ocultas y otras no. La desventaja, que resulta un tanto engorroso, por lo que habría que reservarlo para casos como el citado, cuando nos interese controlar las aristas que se muestran. O bien cuando interese que su grosor o su color sean diferentes.

El tercer método es el que se ilustra en la figura 5.68 y que describo con algo más de detalle a continuación. Tiene la ventaja de que es sencillo, nos evita tener que asignar un modificador UVW al objeto y, si interesa, preserva el material original.

1 Asignar un material de tipo *Composite* al objeto. Si interesa, confirmar que se quiere mantener el material anterior que aparecerá como "Base material". Si no, editar el material *Standard* asignado por defecto a "Base material" y darle el color que sea. En el ejemplo que se ilustra en la figura citada lo he hecho de color blanco y algo autoiluminante, para que el resultado sea uniforme.
2 Para el siguiente material, escoger un material de tipo *Standard*. En *Shader basic parameters*, marcar la casilla *Wire*. En *Blinn basic parameters* asignar un color (por ejemplo negro), que será el color de las líneas. En *Extended parameters* dar un valor a *Size* para controlar el grueso de las líneas.
3 Así se obtendrá un resultado como el de la figura 5.68. Puede volver a editarse el material para ajustar el grosor de las líneas u otras características.

5.11 Texturas

Criterios

La mayoría de los materiales utilizados en arquitectura se basan en texturas obtenidas mediante proyecciones de mapas de bits. Esto requiere tener en cuenta al menos los siguientes aspectos: *a*) Controlar la **calidad** de las imágenes con las que se va a trabajar; *b*) Controlar la **resolución** adecuada, que dependerá en cada caso de la finalidad del trabajo que se está llevando a cabo; *c*) Controlar la **repetición** de las texturas en el caso, frecuente, de que no se utilicen texturas que abarquen la totalidad del objeto sobre el que se van a proyectar; *d*) Controlar la relación entre los archivos que incluyen las texturas y los archivos que incluyen los modelos a los que estas texturas se van a aplicar, dicho de otro modo, utilizar métodos eficaces de **organización** del proyecto.

Todos estos aspectos se tratan en los apartados siguientes.

a) Calidad inicial

La calidad de las imágenes se controla desde dos extremos: en el origen y en el tratamiento posterior.

Sobre el origen poco hay que decir. Podemos obtener texturas digitales a partir de dos vías principales: fotografías tomadas por nosotros mismos y fotografías tomadas por terceros que hemos obtenido por internet, sea por medios gratuitos, sea por medios de pago, sea porque alguien nos las ha regalado. Hay una gran cantidad de sitios web que ofrecen texturas y, como sería de esperar, las mejores (texturas que cubren zonas amplias y con buena resolución) son de pago. Hay sitios web (como CG Textures, probablemente el más conocido) que ofrecen texturas gratuitas de calidad limitada y otras de mejor calidad mediante una subscripción anual. Y otros que ofrecen texturas concretas (de madera, metal, fondos, etc.) mediante pagos por cada una de ellas o por cada paquete, que incluye una pequeña colección. También es posible escanear libros o revistas, utilizando una resolución mayor de la necesaria y luego pasando un filtro de destramado para eliminar las tramas de impresión. Conviene no perder de vista, en cualquiera de estos casos, que la originalidad del proyecto se puede resentir y que, en casos extremos, se puede estar violando derechos de autor.

En el caso de imágenes obtenidas personalmente, será más que conveniente contar con una buena cámara y cierta experiencia en fotografía, cuidar las condiciones de iluminación, evitando sombras, etc. Y tener la paciencia suficiente como para buscar modelos adecuados y esperar a que las condiciones de la toma sean convenientes. No es un trabajo que se pueda improvisar sino un hábito que hay que cultivar.

Las imágenes obtenidas por alguno de los modos citados tendrán que ser tratadas de diferentes modos que también conviene discutir con detalle, para lo que me remito al apartado *e*) de esta sección.

b) Resolución

El control de la resolucion es otro aspecto crucial que afecta a los dos extremos, el origen y el final. En este caso, la recomendación previa es guardar imágenes que tengan buena resolución, del orden de los 2.048 a 4.096 píxeles y luego hacer una copia (véase el apartado sobre organización) para el proyecto concreto y reducir la resolución al mínimo necesario. Si es posible, utilizar valores que sean potencias de 2 (512, 1.024, 2.048, etc.) o bien valores intermedios (768, 1.536, etc.). Las tarjetas gráficas están optimizadas para estos valores por lo que, en general, se tarda menos en hacer un *render* de una textura de 1.024 x 1.024 o de 1.024 x 512 que de texturas mayores pero de valores arbitrarios. En las tarjetas gráficas más recientes esta limitación tiene menos importancia pero no está de más seguir teniéndola en cuenta.

La resolución adecuada depende de dos cosas: del tamaño global de salida previsto

y de la posición de la textura en la escena. Un criterio adecuado para saber cuál es el tamaño necesario para una textura es, por tanto, hacer una doble estimación. En primer lugar, calcular qué tamaño ocupará la textura en la imagen de salida. Si la textura se va a aplicar a un muro que aparecerá al fondo de una imagen de 1.200 píxeles de alto, ocupando una cuarta parte de la altura, bastaría en principio con una cuarta parte del tamaño aunque es mejor multiplicar este valor por dos y redondearlo al valor múltiplo de potencia de 2 más cercano, por ejemplo, en este caso, 640. En segundo lugar, tener en cuenta que el detalle más fino (por ejemplo, las juntas de un muro o un pavimento) debe ocupar al menos un píxel si queremos que sea visible. En la práctica es preferible 3 píxeles. Y a partir de aquí deducir el tamaño total necesario. Si el tamaño es menor de lo requerido las juntas pueden desaparecer.

c) Repetición. Texturas *seamless*

Lo más seguro para que el resultado sea correcto es utilizar texturas a medida de los objetos. Pero esto no se suele cumplir, sea porque no es estrictamente necesario sea porque requiere un trabajo de reparación que no podemos asumir. Así que, para simular un gran paramento, un pavimento o un muro, suele ser bastante habitual utilizar una textura que abarque una parte del paramento y dejar que esta textura se repita para cubrirlo todo.

Cuando se hace esto, es decir, cuando se admite que va a haber **repetición**, es necesario comprobar que no aparecen huellas de "cosido" donde se repite la imagen pues el efecto es penoso y, en muchos casos, arruina la calidad del resultado. Eliminar estas costuras no elimina la repetición de los patrones, que también puede afectar negativamente a la calidad del resultado pero, al menos, alivia un poco esta limitación.

La creación de texturas *seamless* se puede hacer automáticamente en algunos casos pero, más frecuentemente, por métodos manuales. Se puede hacer con relativa facilidad editando la imagen en Photoshop y aplicando un filtro de *desplazar* (*offset*) que se encuentra en el menú *Filtros*/*Otros*. El ejemplo siguiente, ilustrado en la figuras 5.69, explica con algo más de detalle como se puede hacer.

1 La primera imagen muestra un muro de unos 4,80 m de largo por 2,80 m de alto al que se le ha asignado una textura de ladrillo que encaja bien en estas dimensiones si se repite dos veces en anchura y altura. Pero la repetición no funciona pues son visibles las juntas de repetición, como puede apreciarse en la imagen. Para solucionarlo, hacer lo siguiente.
2 Editar la imagen en Photoshop. Ajustar el tamaño a valores adecuados (en el ejemplo se ha cambiado 800 x 547 a 800 x 540: esto hace algo más estilizados los ladrillos, lo que no va mal y nos permitirá trabajar con números más adecuados.
3 Ir a menú *Filtros/Desplazar*. Desplazar la imagen en horizontal la mitad de su anchura (400 píxeles en el ejemplo). Luego retocar las uniones con el pincel clónico hasta obtener un resultado como el de la figura.
4 Repetir la operación en vertical pero solo a la distancia adecuada (unos 16 o 18 píxeles en el ejemplo) para absorber el desfase de las hiladas horizontales. Así se llega al resultado final. Al utilizar esta textura las uniones no serán visibles.

d) Organización

Las texturas se utilizan en la mayoría de los programas como referencias externas. Esto quiere decir que la parte de la definición de un material que incluye texturas solo consiste en una dirección. Y al activar el cálculo de *render*, el programa irá a buscar esa textura a la dirección indicada. Si no la encuentra, porque ya no está en esa dirección, sea porque el usuario ha cambiado de

ordenador, sea porque ha decidido organizar sus carpetas de otro modo, sea por lo que sea, dará un mensaje de error. Algo que ocurre con, para los principiantes, irritante frecuencia.

Para que esto no ocurra es necesario organizar adecuadamente el proyecto. Y la mayoría de los programas utilizan un mismo criterio. Van a buscar las texturas en la misma carpeta del archivo de modelo que se está utilizando o en una carpeta situada al lado.

Todo esto quiere decir que es muy importante (como debería serlo en cualquier caso) organizar adecuadamente el proyecto en que se esté trabajando. Y los criterios son semejantes en todos los programas.

El modo de organización adecuado de un proyecto ya lo hemos visto en el capítulo 3, concretamente en la sección sobre "Organización del proyecto". Así que me limito a recordar que esta organización puede ser de complejidad "mínima", "media" o "avanzada". Y que en el primer caso, lo "mínimo" es situar todas las texturas que se vayan a utilizar en una carpeta con un nombre adecuado tal como "mapas" (o "texturas" o "imágenes", si se prefiere). Y, si es necesario (que en general no lo será), indicar al programa estas rutas de acceso. Por ejemplo, en 3ds Max, desde el menú *Customize / Configure user paths*. Esto sirve principalmente para que al buscar una textura el programa vaya en primer lugar a la carpeta indicada, lo que nos ahorrará muchos clics y paseos por el ordenador.

Hay toda una serie de cuestiones relativas a la nomenclatura más adecuada, a las que ya me he referido en el capítulo 3, en el apartado sobre asignaciones y nomenclatura de la sección sobre "Organización del proyecto" , que convendrá revisar. Y también al uso de bibliotecas de mapas, que pueden copiarse o referenciarse, según los gustos, aunque es más efectivo contar con una biblioteca general e ir copiando las texturas necesarias para cada proyecto, aunque esto implique repe-

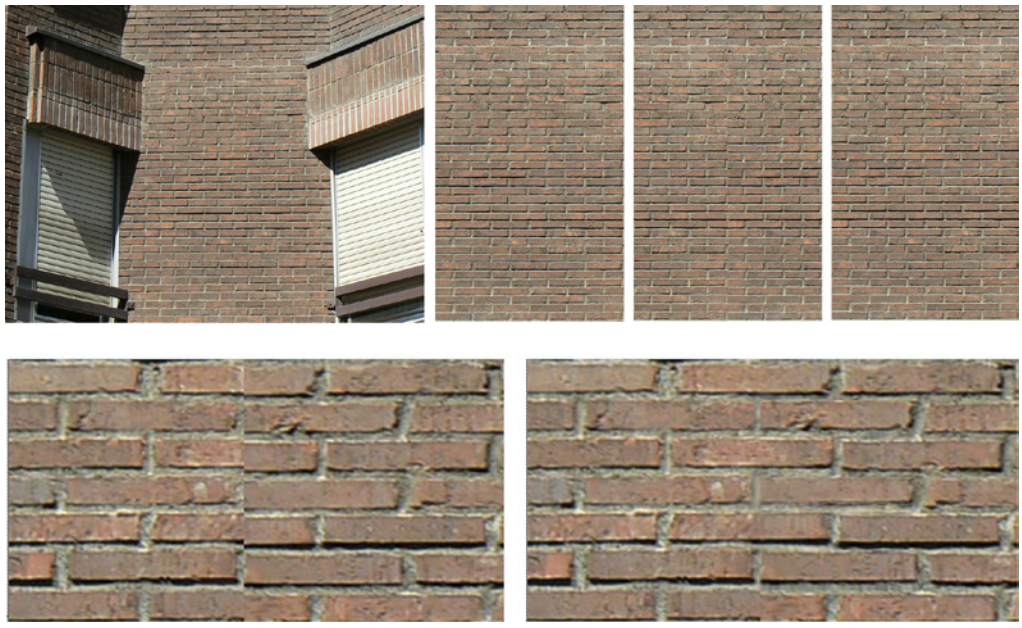


Figura 5.69 Ajuste de texturas para ocultar repeticiones (texturas seamless*): a) Foto original; b) Recortada y rectificada; c) Desplazada; d) Resultado final; e) Detalle del desplazamiento sin corregir; f) Detalle del desplazamiento corregido.*

ticiones. O bien utilizar una combinación de carpetas generales y carpetas específicas. Esto depende de los casos y, en gran medida, de las preferencias de cada cual.

e) Calidad final. Ajustes y corrección de defectos

Por lo que respecta al tratamiento posterior hay una serie de pasos básicos, que comentaré muy rápidamente pues es de suponer que son conocidos, y otros que merecen comentarios más detallados.

1) El primer paso será eliminar la distorsión de la imagen. Este es un tema incómodo pero importante pues las distorsiones debidas a las imperfecciones de las lentes se acentúan cuando se utilizan cámaras con sensores pequeños.

La mejor recomendación es utilizar un programa especializado. Uno de los más utilizados por los fotógrafos profesionales es DxO Optics. Este programa, además de contar con una base actualizada de cámaras profesionales, que permite eliminar la distorsión automáticamente, cuenta con varios recursos muy útiles para eliminar defectos habituales. Pero aunque su precio no sea muy alto es bien posible que muchos prefieran utilizar otros recursos. Las versiones de Adobe Photoshop posteriores a la CS5 también incluyen bases de datos de cámaras que permiten hacer esta corrección automática. Y, en fin, hay algunos programas más sencillos de bajo coste que también incluyen estos recursos, aunque es difícil que mantengan sus bases al día.

Si no se cuenta con estos recursos la única solución será hacer esta corrección manualmente. No es muy laborioso si solo hay que hacerlo para unas pocas imágenes. En Photoshop se puede utilizar el filtro “Corrección de lente” que permite eliminar la distorsión haciendo la imágen más cóncava o más convexa de un modo controlado. En Gimp se encuentra un filtro del mismo nombre que hace la misma función aunque de un modo algo más limitado.

En cualquier caso, no se debe caer en la tentación de pasar por alto este ajuste pensando que “no se notará” o que “si se nota, ya lo corregiremos después”. En la gran mayoría de los casos *sí* se notará y resultará mucho más difícil, si no imposible, corregir esta distorsión cuando ya se ha manipulado la imagen.

2) Una vez que se ha corregido la distorsión, hay que enderezar y recortar la imagen. Si se utiliza Photoshop esto implica dos operaciones muy sencillas: rotar y recortar. Para rotar la imagen de modo que una línea de referencia quede horizontal (o vertical), medirla con la herramienta “Regla”. Si, inmediatamente después de haber tomado esta medida, se activa el comando de “Rotación libre” el panel mostrará el ángulo que se acaba de medir. Basta confirmar este valor para que la imagen rote y la línea de referencia pase a ser horizontal (o vertical). Si se utiliza Gimp el procedimiento es básicamente el mismo aunque habrá que anotar el ángulo medido e introducirlo manualmente. Para recortar la imagen utilizar la herramienta de recortar (*Crop*) en cualquiera de los dos programas.

3) El siguiente paso, que será necesario en bastantes casos, es hacer ajustes básicos de color. La herramienta más utilizada, si las correcciones son simples, es, también en Photoshop, el ajuste de Tono/Saturación (menú Ajustes o Ctrl+U). Y en Gimp, la herramienta del mismo nombre (menú Colores). Hay que tener en cuenta que si no se tiene muy claro el destino de la imagen o cómo va a armonizar con otros tonos, este paso se puede dejar para más adelante pues es posible ajustar el color de los mapas de bits desde 3ds Max por medio de un mapa especial denominado *Color correction* al que ya me he referido anteriormente.

4) En muchos casos, sin embargo, nos encontraremos con problemas más difíciles. Los principales son la aparición de degradados o inconsistencias de color en la imagen. En estos casos se pueden utilizar los siguientes recursos:

a) Si el degradado es lineal y está limitado, como ocurre con piezas rectangulares (maderas, baldosas, etc., superponer, en otra capa, un degradado lineal de gris claro a gris oscuro que vaya en sentido contrario al degradado de la imagen. Cambiar el modo de

fusión a multiplicar (para oscurecer) o trama (para aclarar) y reducir la opacidad. Con la intensidad de opacidad de la capa se controla la intensidad del efecto y se puede utilizar el panel de información para leer los valores resultantes. La figura 5.70 muestra estas tres fases características

b) Si el degradado no es lineal, en algunos casos puede bastar con recurrir a herramientas características de programas como Photoshop o Gimp: pincel clónico, pincel corrector, reducción de intensidad en determinadas zonas, sea con Ajustes/Brillo, sea con herramientas como las de exponer (*burn*) y subexponer (*dodge*). En otros casos habrá que recorrer a métodos algo más elaborados. La figura 5.71 muestra un paramento en el que parte de la imagen incluye un degradado debido a la influencia de luces en una determinada dirección. Un procedimiento para corregirlo, en Photoshop (en Gimp hay métodos equivalentes), sería el siguiente:

1 Duplicar la capa.
2 Aplicar un filtro de desenfoque gaussiano para suavizar la aplicación de lo que sigue y reducir los errores visibles. En el ejemplo de la figura citada se ha utilizado un radio de 12,0.
3 Invertir los valores de la capa (menú *Imagen/Ajustes* en Photoshop o *Colores*/*Invertir* en Gimp).
4 Desaturarla (menú *Filtros*/*Desaturar*) para que el filtro de paso alto afecte solo a la luminosidad. Escoger la opción “luminosidad” aunque no hay diferencias importantes con respecto a las otras dos.
5 Cambiar el modo de fusión a *solapar* (*overlay*). También se consigue un efecto similar con claridad suave.

Así se conseguirá ajustar las luminosidades. Pero es posible que la tonalidad se desajuste. Si es así, continuar como sigue:

6 Captar el color de una zona media de la imagen original con el cuentagotas en modo “Promedio de 11x11” (o menos o más, según el tamaño de la imagen).
7 Crear una capa adicional y rellenarla con este color. Cambiar su modo a “color”. Así se teñirán con un color uniforme las capas inferiores.
8 Ajustar la tonalidad, la saturación y la iluminación de esta capa con *Ajustes*/*Tono-Saturación* (Ctrl+U).

En otros casos puede ser necesario eliminar sombras arrojadas. La tercera figura de este grupo, la 5.72, muestra una imagen con unas sombras muy marcadas y en una dirección determinada. Para utilizarla como textura necesitamos reducir al máximo estas sombras que nos obligarían a utilizar luces en una dirección que coincidiera con ellas. El procedimiento, en Photoshop (en Gimp hay métodos equivalentes), sería el siguiente:

Figura 5.70 Ajuste de texturas. Corrección de degradados lineales: a) original; b) degradado superpuesto; c) el mismo degradado con modo de fusión tipo trama y opacidad de la capa en un 36 %.

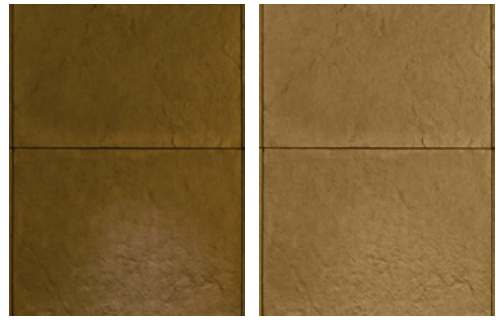


Figura 5.71 Ajustes de texturas. Corrección de degradados no lineales con procedimientos de superposición de capas indicados en el texto.

1 Ir al menú *Imagen/Ajustes/Sombras e iluminaciones*.
2 En *Sombras*, aumentar al máximo el parámetro *Cantidad* (100 % en este ejemplo) y aumentar también el parámetro *Anchura tonal* (80 % en este ejemplo).
3 Si se considera necesario, aumentar un poco el contraste del resultado con los ajustes de curvas o niveles y retocar algunos puntos en los que aún se vieran sombras arrojadas con el pincel clónico.

Hay varios recursos similares, como el ajuste *Reemplazar color* que es bastante conocido. Pero otro método importante, menos conocido y que he utilizado en el ejemplo que sigue, es utilizar la herramienta *Igualar color*. Lo explico con un ejemplo sencillo. Supongamos que tenemos una imagen en la que hay un sector (por ejemplo, un grupo de baldosas) que sabemos que tiene el mismo color que otro sector pero que, debido a la iluminación ha quedado más oscuro o más claro o más o menos saturado. Podemos ajustar a ojo, manteniendo abiertas dos ventanas del programa, una que muestra el color que damos por bueno y otra que muestre el color que queremos modificar. Pero es más eficaz hacer lo siguiente:

1 Guardar una copia de la imagen con otro nombre.
2 Seleccionar la zona de esta imagen copiada que queremos tomar como referencia. Intentar que la zona de referencia incluya valores en toda la gama de luminosidades.

Figura 5.72 Ajustes de texturas. Reducción de sombras arrojadas.

3 Mantener la anterior abierta y volver a la imagen original y seleccionar con precisión la zona que queremos modificar (utilizar las herramientas de refinar borde para que la transición sea más suave).
4 Ir al menú *Imagen/ Ajustes/ Igualar color* y:
a) En el grupo *Estadística de la imagen*, en "Origen", descolgar la lista, que mostrará las imágenes disponibles, y seleccionar la imagen copiada. Mantener marcada la opción "Usar selección de origen". El color de la original se ajustará automáticamente.
b) En el grupo *Opciones de Imagen*, utilizar los controles de *Luminancia*, *Intensidad de color* y *Transición* para acabar de refinar el efecto.

Ejemplo de recomposición de textura

Para que se entiendan mejor los procesos que puede ser necesario seguir en la práctica, comento con algún detalle un ejemplo más completo. Se trata de una textura obtenida a partir de fotografías de un piso del Ensanche de Barcelona.

El primer paso es obtener una serie de fotografías de suficiente calidad, procurando que no haya variaciones importantes en la iluminación y que cubran una parte importante del suelo. Como es obvio, si estamos fotografiando un pavimento iluminado por un balcón, lo que creará un gradiente continuo, esto es una empresa imposible a la que solo cabe aproximarse de un modo suficiente para poder empezar a trabajar. Las primeras imágenes de la figura 5.73 muestran las diferencias iniciales de distribución de iluminación de las imágenes.

El siguiente paso es crear una retícula geométrica que se corresponda con la retícula ideal del paramento y situar las imágenes seleccionadas sobre esta retícula, en otra capa y bajando su opacidad para ver la trama de la capa inferior. Lo siguiente, que en la mayoría de los casos será lo más complicado, es ajustar la distribución de la iluminación siguiendo alguno de los métodos que se han descrito antes. A partir de ahí, se

trata de ir copiando, clonando, retocando y ajustando las partes copiadas para que queden bien integradas en el fondo y, al mismo tiempo, preservando las variaciones, pues lo que nos interesa en un caso como este es mantener el carácter artesanal del paramento. Es decir, resumiendo y generalizando el proceso por pasos:

1 Obtener una serie de fotografías de buena calidad, procurando que no haya grandes variaciones de luz (y, obviamente, evitando sombras arrojadas).
2 Seleccionar las mejores. Corregir la distorsión, preferentemente por métodos automáticos (programas como DXO o bases de datos que reconozcan el modelo de cámara).
3 Calcular la resolución mínimima de tal modo que las juntas o las líneas más finas cuyo color o textura queremos que resulte visible tengan una resolución mínima de 3 píxeles.
4 Crear una trama geométrica que responda a la geometría subyacente y cuyos ejes coincidan con las juntas de los elementos y que abarque una parte principal de la textura que se vaya a modelar. En el ejemplo (el piso del Ensanche) se ha creado una imagen que cubre la cuarta parte y luego se ha copiado para completar la totalidad. En otros casos puede bastar con una proporción menor y no será necesario copiarla, bastará con ajustar los bordes para que la repetición no sea visible (véase el apartado sobre texturas *seamless*).
5 Ajustar la iluminación de las partes principales con los métodos citados (degradados superpuestos, recortes manuales con las herramientas de subexposición o sobreexposición local, reemplazar o igualar color, etc.).
6 Clonar las partes retocadas a otras posiciones rellenando todos los huecos.
7 Por último, utilizar las herramientas de suavizado o desenfoque local, dedo (*smudge*) para ocultar las transiciones entre zonas copiadas.

Figura 5.73 Textura de un suelo de un piso del Ensanche de Barcelona. Proceso de recomposición.

Texturas artificiales

Noise. Análisis

En el capítulo anterior hemos visto los parámetros básicos del mapa procedural *Noise* y cómo estos parámetros se pueden manipular para crear patrones irregulares que pueden servir, entre otras cosas, para crear efectos de relieve, como también se ha visto en este capítulo.

Pero pueden forzarse estos parámetros para crear otro tipo de patrones que pueden combinarse con otros mapas procedurales para crear diferentes tipos de textura. Por ejemplo, se pueden simular grietas o nubes con algunas modificaciones algo más sofisticadas de estos parámetros.

Las grietas se pueden simular con un mapa *Noise* de dos colores, manipulando el mapa de *Output*. Las figuras adjuntas están aplicadas a un plano de 100×100 cm y tienen las configuraciones indicadas en las propias imágenes. Las dos primeras de la figura 5.74 sirven para crear grietas o contornos. Si la manipulación de la curva es simple, como en el caso de la segunda imagen, el resultado sería el mismo que cambiar los valores de *High*/*Low* a 0.49 y 0.51 respectivamente. Pero utilizar el mapa permite controlar mejor las transiciones, como vemos sobre todo en las imágenes que siguen. Pues los efectos de "nubes" se pueden conseguir también manipulando las curvas de salida como en las dos últimas imágenes de la figura 5.74 lo que no es posible mediante una variación simple como la dada por los parámetros *High*/*Low*.

Los efectos de "nubes suaves" o "aguas" (transiciones suaves) se consiguen, así, fácilmente utilizando un mapa de ruido de poco contraste. Pero para conseguir efectos más interesantes se puede utilizar un mapa *Mix* (o *Composite*) mezclando dos mapas de ruido de diferentes tamaños y colores y cambiando la fase de uno de los dos para que se mezclen adecuadamente. Las imágenes de la figura 5.75 se han obtenido con un material corriente (*Standard* o *Arch&Design*) en el que

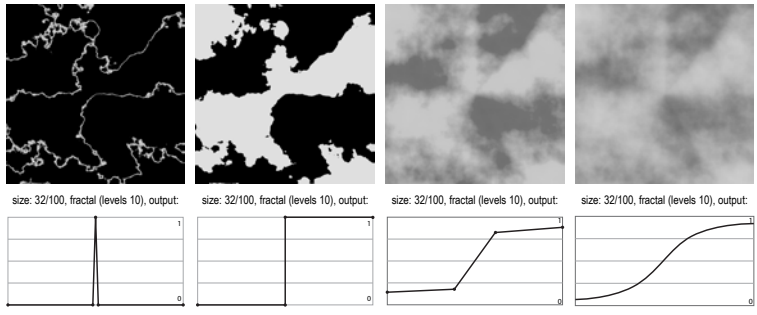


Figura 5.74 Noise. *Mapa* Noise *aplicado a un plano de 1 m de lado, con la configuración y las curvas de salida reajustadas tal como se muestran en los esquemas inferiores.*

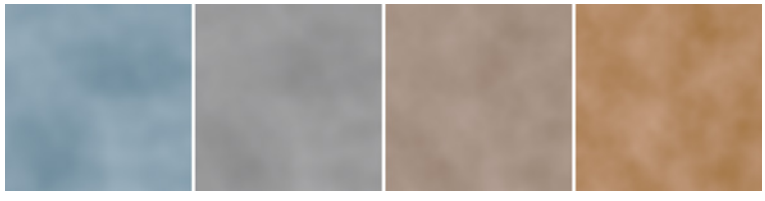


Figura 5.75 Noise. *Material con mezcla de dos mapas (*Mix*). Mapa 1: colores azul oscuro y claro. Tipo fractal.* Size *45/100. Mapa 2: colores ocre oscuro y claro. Tipo fractal.* Size *25/100.* Mix amount*:* a*) 0 %,* b*) 35 %,* c*) 65 %,* d*) 100 %.*

se ha asignado un mapa *Mix* a *Diffuse* con las configuraciones indicadas en las figuras.

Cellular noise. Análisis

Al igual que en el caso de *Noise*, los parámetros básicos ya se han descrito en el capítulo anterior. Pero también en este caso se puede sacar más partido de este mapa procedural forzando estos parámetros básicos. Este mapa es más complejo y requerirá un análisis algo más detallado.

La idea básica del mapa procedural *Cellular noise* de Steven Worley, es situar una serie de puntos en el espacio por medio de algún tipo de distribución semialeatoria (el algoritmo original de Worley utilizaba una distribución de Poisson). Para cada punto de esta distribución, hay otro punto que está más cerca de dicho punto que de cualquier otro y, si se consideran todos los puntos posibles, se definen una serie de celdas que forman un diagrama de Voronoi (una estructura que es dual de una triangulación de Delaunay), una estructura en la que los contornos de las celdas están a la distancia mínima de los centros de las celdas adyacentes.

La figura 5.76 muestra una distribución característica de este diagrama. A partir de esto, en el algoritmo de Worley se define una función $F(x)$ dada por la distancia del punto considerado al más cercano. Al variar la posición de x se obtienen diferentes valores de F que varían de modo continuo, aunque su derivada varía de modo discontinuo, al saltar el cálculo de la distancia desde un punto cercano a otro más cercano. Esta función se puede evaluar de muy diversos modos y, según su definición y los parámetros que intervengan, se obtienen diferentes resultados.

Si el algoritmo de Worley se aplica directamente, el usuario puede definir la función y sus variantes y puede hacerse una idea más cabal del procedimiento que se está activando. Sin embargo, esto requiere unos conocimientos matemáticos y de programación que no están al alcance de los usuarios corrientes de los programas de simulación por lo que, en la práctica, nos encontramos con algún tipo de aplicación de este algoritmo a través de una serie de parámetros que intentan ofrecer una interactividad más intuitiva.

Desafortunadamente esto no suele ser exactamente así. En el caso de 3ds Max (como en otros programas), las variaciones se controlan por medio de algunos parámetros principales que pretenden ser intuitivos pero cuyo sentido no es excesivamente claro. No hay ninguna dificultad por lo que respecta a los parámetros correspondientes al tipo de célula base (*circle* o *chip*), el tamaño (*size*), el color de la célula base y los dos colores de transición entre células, o a la opción fractal. Hay otros parámeros secundarios de importancia menor pero que también tienen un sentido claro: *Adaptive* hace que el número de recursiones dependa de la distancia a la cámara, lo que reduce el *antialiasing* y el tiempo de *render*, por lo que conviene dejarlo activado; *Roughness* tiene sentido si se utiliza el mapa celular como mapa de relieve, *Bump*: en este caso, al aumentar el valor de *roughness* cada iteración se acerca al valor de la iteración previa; si *roughness* es 0 cada iteración es la mitad del valor de la anterior y un valor de 1,0 equivale a desactivar el modo fractal; con *Bump smoothing* ocurre algo similar: solo tiene sentido cuando se usa el mapa celular como mapa de *Bump* y, en este caso, mejora los resultados, suavizando las irregularidades, si se aumenta el valor predeterminado, 0,1.

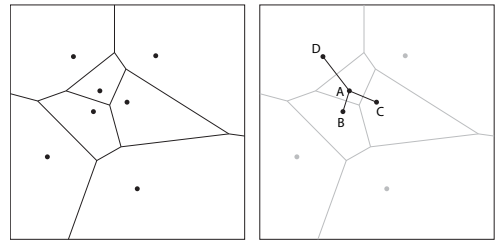


Figura 5.76 Cellular noise. a*) Diagrama de Voronoi;* b*) Distancia del punto A, a puntos circundantes. La distancia más corta, base de la función asociada, es con el punto B.*

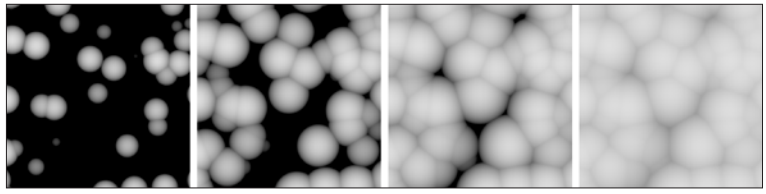


Figura 5.77 Cellular noise. *Círculos.* Size *25 sobre un plano de 100 x 100. Variaciones de* Spread*:* a*) 0,10;* b*) 0,25;* c*) 0,50;* d*) 1,00.*

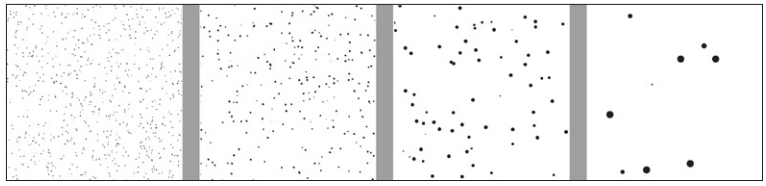


Figura 5.78 Cellular noise. *Círculos.* Spread *0,01,* Threshold *1,0. Variaciones de* Size*:* a*) 2,5;* b*) 5;* c*) 10;* d*) 20.*

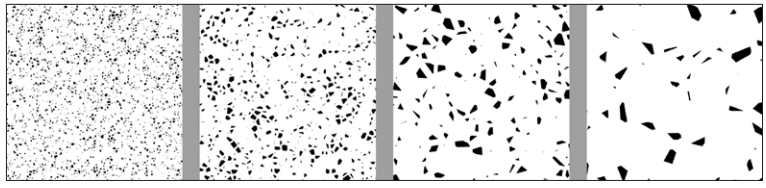


Figura 5.79 Cellular noise. Chips. Spread *0,5.* Threshold *0, 0, 1. Variaciones de* Size*:* a*) 2;* b*) 4;* c*) 8;* d*) 16.*

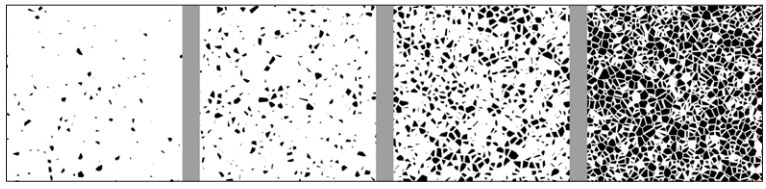


Figura 5.80 Cellular noise. Chips. Size *5,0.* Threshold *0, 0, 1. Variaciones de* Spread*:* a*) 0,75;* b*) 0,55;* c*) 0,35;* d*) 0,15.*

Figura 5.81 Cellular noise. Chips. Cell color*: blanco;* Division colors*: negros,* Size *10,* Spread *0,* Threshold *0.0,0.5,1.0:* a*)* Cell color variation *25;* b*) Idem fractal;* c*)* Cell color variation *75;* d*) Idem fractal.*

Pero las variaciones dependen principalmente de otros dos parámetros fundamentales, *Spread* y *Threshold*, cuyo sentido, como decía, es menos evidente y que están relacionados de modo más directo con la función principal del algoritmo de Worley. Se definen del modo siguiente en la ayuda (doy entre paréntesis los valores predeterminados):

Spread "altera el tamaño de las células individuales" (0,5).

Threshold incluye tres parámetros: "*Low*, ajusta el tamaño relativo de las células (0,0 %). *Mid* ajusta el tamaño relativo del primer color de división con respecto al segundo (50 %). *High* ajusta el tamaño relativo de las divisiones (100 %)."

Esto es todo lo que dice la ayuda. Pero la dificultad principal es que estos cuatro parámetros interactúan entre sí de modos que no son fáciles de controlar.

Por otro lado, debe tenerse en cuenta que lo que muestra el visor del editor de materiales no se corresponde del todo con lo que aparecerá en la representación final, por lo que los resultados deberán ajustarse a cada caso y a las dimensiones concretas del objeto y hacer un *render* para comprobarlo.

El análisis mínimamente sistemático de las posibilidades de este algoritmo nos llevaría demasiado lejos por lo que me limitaré a algunas indicaciones básicas que el lector puede ampliar por su cuenta.

Para generar texturas de materiales puede ser más interesante obtener, mediante ensayos de prueba y error, valores adecuados para aislar fragmentos que puedan sugerir incrustaciones minerales. Las figuras 5.77 a 5.82, muestran cómo, al introducir valores adecuados de *Spread* y *Threshold*, con variaciones del tamaño base, *Size*, se pueden obtener resultados de bastante interés para combinarlos con otras texturas, tal como veremos más adelante. Para simplificar el análisis me he limitado a algunos ejemplos elementales en blanco y negro. Si se varían los colores las posibilidades se multiplican.

Todos los ejemplos están aplicados sobre un plano de 100 x 100 unidades.

Si se mantienen los principales valores fijos, un modo sencillo de aislar los fragmentos es, como se puede ver en las figuras citadas, igualar los colores de división y hacer que los valores de *Threshold* sean 0, 0, 1 (*low*, *mid* y *high*). Otras variantes que se indican en los pies de dichas figuras también permiten crear configuraciones simples pero que, en algunos casos, pueden servir directamente para simular ciertos materiales, como metal galvanizado.

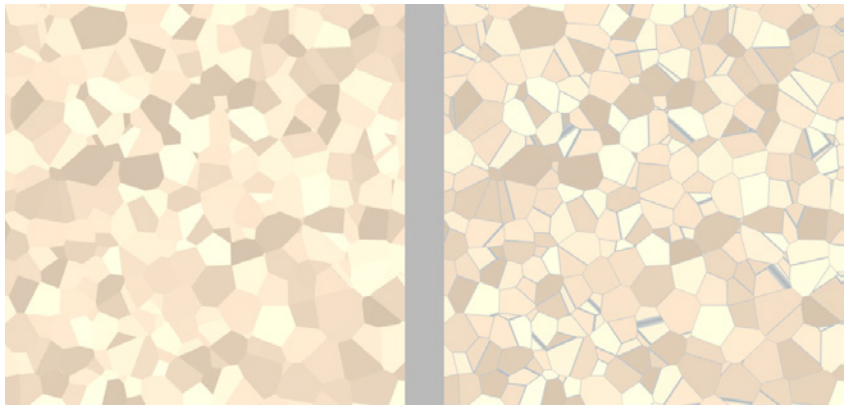


Figura 5.82 Cellular noise. Chips. *Configuración similar a la de la figura anterior variando los colores y (derecha) aumentando el valor de* Spread *a 0,15.*

Ejemplos de aplicación de *Noise* y *Cellular noise*

A partir del análisis anterior podemos contar con un abanico más amplio de posibilidades, de recursos ligados a una serie de configuraciones básicas que pueden combinarse entre sí de diferentes modos. Como ya he dicho, estas posibilidades son infinitas y solo dependen de la imaginación, de la paciencia y del interés de quien las utilice. La ventaja, que no debe perderse de vista, es que si se consigue un efecto adecuado, este efecto puede modificarse con facilidad sin más que cambiar unos pocos parámetros y no requiere almacenar mapas de bits en carpetas que hay que ir arrastrando con cada proyecto.

Me limitaré a tres ejemplos relativamente sencillos. El primero es una aplicación prácticamente directa de la última configuración utilizada en la figura 5.82 y se puede utilizar para simular metal galvanizado aunque también es fácil de adaptar para crear mosaicos irregulares. El segundo es una aplicación algo más compleja de mapas celulares superpuestos y combinados con ruido que puede utilizarse para simular un pavimento de terrazo o granito. El tercero es una aplicación también relativamente compleja de mapas celulares y de ruido que puede servir para simular mármol.

El primer ejemplo se basa en un material con la configuración que se indica en el pie de la figura 5.83. La idea básica es crear un material con un mapa que represente adecuadamente la textura característica de un metal galvanizado. Como hemos visto en el apartado anterior, esto es bastante sencillo de conseguir con la configuración indicada. Una vez que se cuenta con un material de este tipo puede aplicarse directamente a objetos como los de la figura 5.83. Para que la distribución sea regular por todos los lados, he añadido a los objetos un modificador UVW en modo *box* con los tres lados iguales. También conviene añadir luces complementarias que realcen puntos especulares de los objetos. El color y el mayor o menor contraste de la textura se controla con facilidad sin más que ajustar los valores de *Division colors* y el parámetro *Variation* de *Cell color*.

El segundo ejemplo se basa en el otro esquema incluido en la figura 5.84. La generación del mapa en este caso es más compleja pues se ha superpuesto una textura de manchas suaves (*Noise*), sobre una textura granulada (*Cellular noise*), sobre otra capa de grietas finas y sobre otra capa de grietas más gruesas, obtenidas en ambos casos con los métodos que he descrito al comienzo. A diferencia del ejemplo siguiente, en este caso no podemos aplicar directamente este mapa a un objeto como un pavimento pues las piezas resultarían iguales y se echaría a perder el efecto. Sería posible aplicarlo automáticamente por medio de un script que incoporara variaciones y cambios de posición, pero esta posibilidad se sale de los límites de este libro. Sin embargo, cabe otra posibilidad que es muy sencilla y no lleva más de unos minutos. Se trata de crear un mapa de bits a partir de este mapa procedural. Para ello, todo lo que hay que hacer es grabar el resultado a una resolución suficientemente alta, llevarlo a Photoshop o Gimp, hacer varias copias de diferentes zonas, rotarlas, yuxtaponerlas y grabar el resultado. Una vez que se cuenta con este mapa, se aplica al objeto, un pavimento en este caso, a través de un tipo de material corriente (*Standard* o *Arch&Design*) al que se ha aplicado al canal *Diffuse* un mapa compuesto. El uso de un mapa compuesto en este caso es, simplemente, para poder superponer las juntas según lo que interese. Un modo sencillo de hacerlo es, como se ha hecho en este ejemplo, utilizar un mapa procedural de tipo *Tiles*, con las juntas de color gris oscuro y las zonas correspondientes a la “baldosa” de color blanco. Si se aplica con el modo de fusión en *Multiply*, las zonas blancas retendrán el color de la textura anterior y las zonas oscuras se superpondrán a esta textura. Esta misma textura se copia y se lleva a la propiedad de reflexión para, como ya hemos visto, anular el efecto de reflexión en estas zonas y

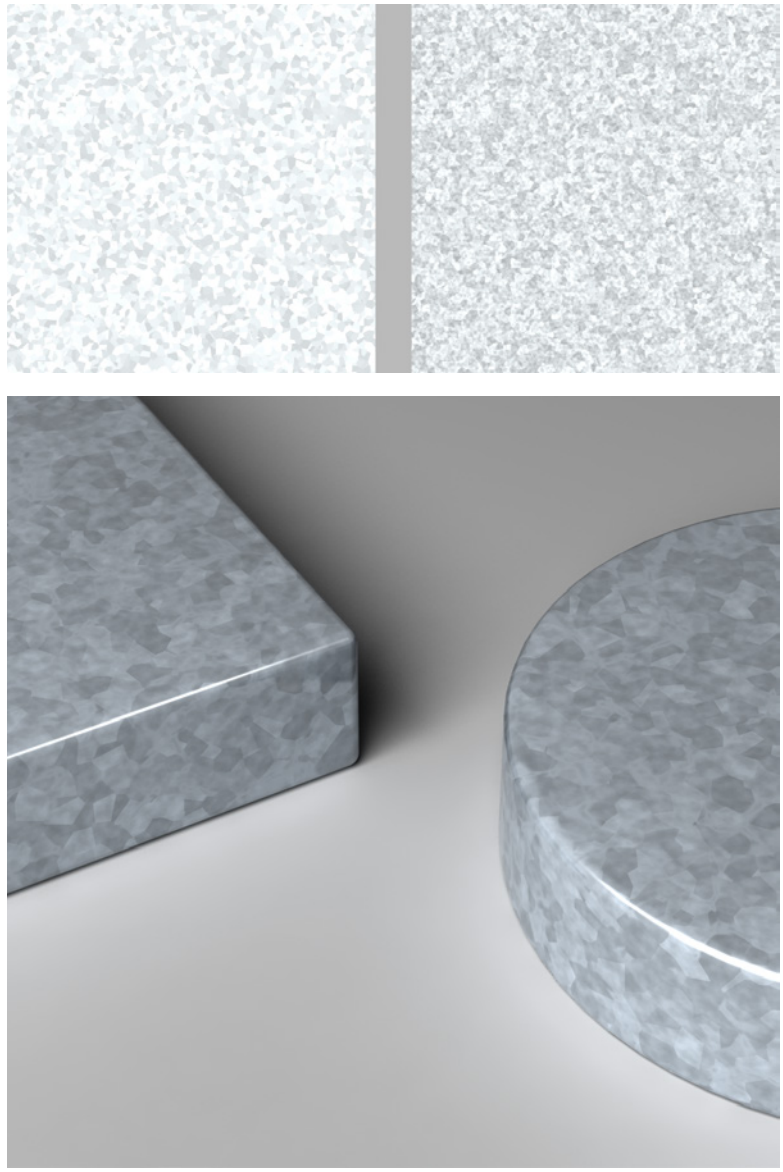


Figura 5.83 Aplicación de Cellular noise *a la simulación de metal galvanizado. En la parte superior se muestran dos variantes del mapa:* a) Cellular noise. Chips. Cell color *gris azulado,* variation *25.* Division colors *grises azulados.* Size *20 (sobre un plano de 100 x 100).* Spread *0.15.* Thresholds *predet (0,0, 0,5, 1,0);* b) *Igual pero substituyendo el* Cell color *por un mapa de tipo* Noise *con la siguiente configuración: fractal 3 iteraciones,* Size *1.0,* High / Low *0,65 / 0,35, colores grises azulados. En la parte inferior se muestra la aplicación de este mapa a dos objetos por medio de un material reflectante con el mapa aplicado como textura.*

“mármol” (A&D)
color difuso: mapa compuesto
capa 1: mapa 1 (noise) (textura base)
color 1 noise: mapa 11 (cellular)
color 2 noise: mapa 12 (cellular)
capa 2: mapa 2 (noise) (manchas globales)
capa 3: mapa 3 (noise) (grietas finas)
capa 4: mapa 4 (noise) grietas grandes)

“pvm mármol” (A&D)
color difuso: mapa compuesto
capa 1: mapa 1 (bitmap) (”pvmMarmol.jpg”)
capa 2: mapa 2 (tiles) (juntas)
mapa reflexión: mapa 2 (tiles)

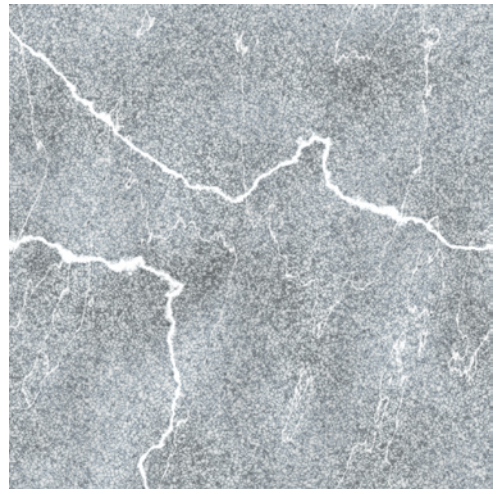
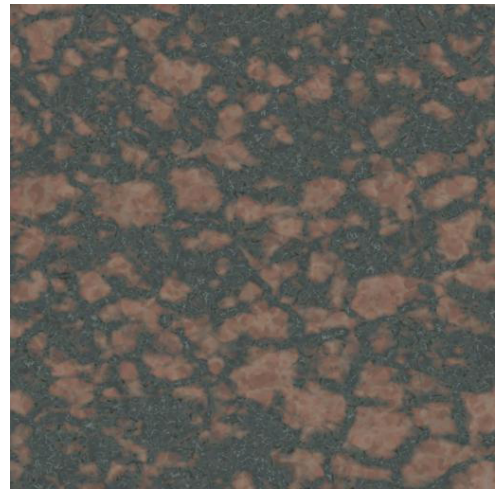


Figura 5.84 Aplicación de una combinación de mapas de tipo Noise *y* Cellular noise *para la simulación de un pavimento de mármol:* a) *Esquema del material utilizado;* b) *Vista del mapa aislado;* c) *Aplicación de una serie de estos mapas a la simulación de un pavimento por medio de un material reflectante con la combinación de mapas aplicada como textura.*

“granito” (*A&D*)
- color difuso: mapa compuesto
  - capa 1: mapa 1 (Cellular)
    - cell color: mapa 11 (Cellular)
    - division color1 : mapa 12 (Cellular)
    - division color2 : mapa 13 (Cellular)
  - capa 2: mapa 2 (tiles) (juntas)
- mapa reflexión: mapa 2 (tiles)



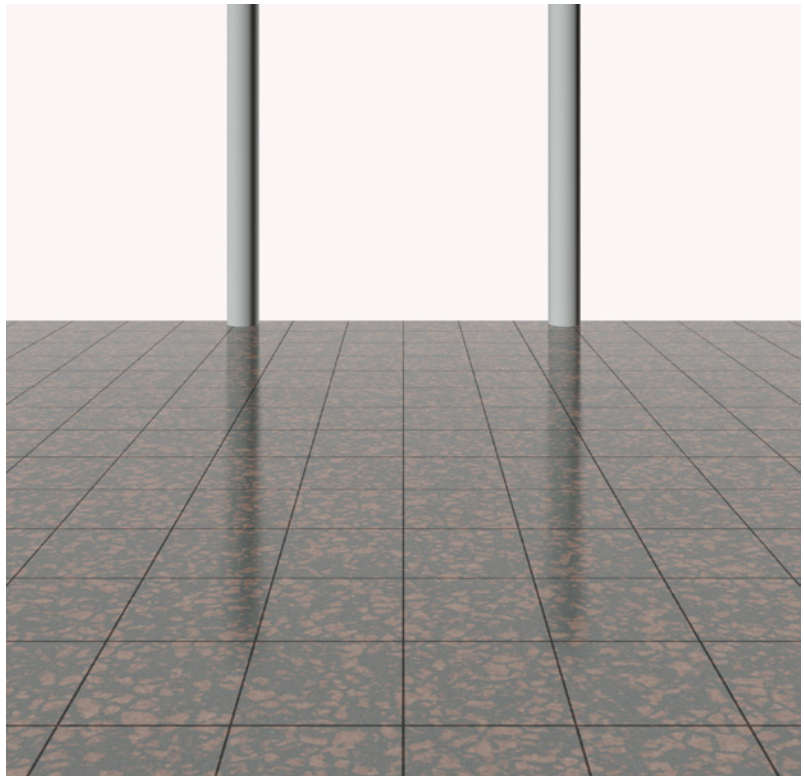


Figura 5.85 Aplicación de una combinación de mapas de tipo Noise *y* Cellular noise *para la simulación de un pavimento de granito o terrazo:* a) *Esquema del material utilizado;* b) *Vista del mapa aislado;* c) *Aplicación de este mapa a la simulación de un pavimento por medio de un material reflectante con el mapa compuesto aplicado como textura.*

de este modo conseguir un efecto más real y más contrastado.

El tercer ejemplo es similar al anterior y se basa igualmente en un material con la configuración resumida en el esquema de la figura 5.85 y que simula una pieza de granito o terrazo. En este caso, la idea básica es superponer varias texturas que simulan fragmentos de diversos tamaños y colores, por medio de mapas procedurales, para simular la textura de un material de estas características. Las configuraciones utilizadas son similares a las que hemos visto en los apartados anteriores, es decir, utilizar el tipo *chips*, en modo fractal, con diferentes tamaños y colores, y ajustes de *Spread* y *Threshold* adecuados para mantener estos fragmentos separados. Una vez que se cuenta con este mapa, a diferencia del caso anterior, puede aplicarse directamente a todo el plano base pues es un material informe sin divisiones claras. Para ello bastará con utilizar un esquema similar al del caso anterior, es decir, un material corriente (*Standard* o *Arch&Design*), con el mapa aplicado a *Diffuse* y, también como en el caso anterior, con juntas superpuestas por medio de otra capa asignada con el modo de fusión en *Multiply* y con este mismo mapa asignado a la propiedad de reflexión para anular el reflejo en las zonas correspondientes a estas juntas.

Otros recursos procedurales. Ejemplo de simulación de cuero

El siguiente ejemplo utiliza dos texturas "procedurales" de diferente tipo que las anteriores. Las comillas se deben a que no son texturas procedurales en sentido estricto pues se aplicarán como mapas de bits. Pero tampoco son mapas de bits corrientes pues no requieren sacar fotografías de texturas reales y pueden generarse con facilidad sin salir de casa.

La primera sí merecería el calificativo de procedural pues utiliza un algoritmo de Photoshop que podía estar disponible en 3ds Max u otro programa de simulación. De hecho, hay múltiples algoritmos que están disponibles en un programa y no en otro y un recurso sencillo para sacar partido de esta variedad de recursos es generar una textura en un programa y luego convertirla en mapa de bits para utilizarla en otro. Es evidente, pese a todo, de que con este procedimiento estamos perdiendo una de las principales ventajas de los mapas procedurales, el no tener que preocuparnos ni de la proyección ni de la gestión. Pero esta pérdida se compensa hasta cierto punto porque ampliamos la gama de recursos disponibles.

La segunda es más heterodoxa pues se basa en un "procedimiento" mecánico para crear una textura y luego reutilizarla como

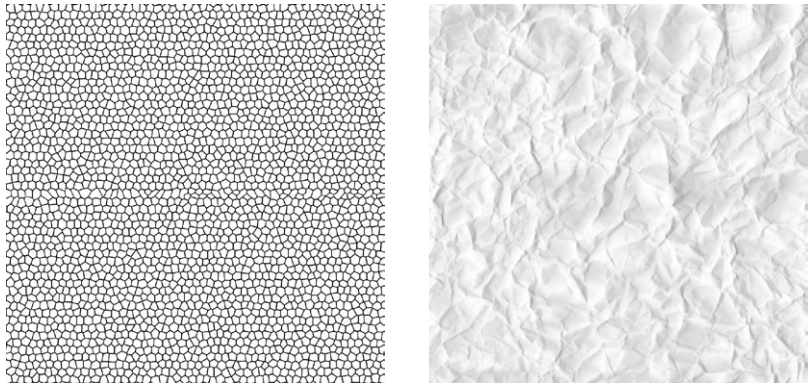


Figura 5.86 Simulación de cuero. Texturas utilizadas: a) *Retícula generada mediante un algoritmo de pintura;* b) *Textura obtenida con un papel arrugado.*

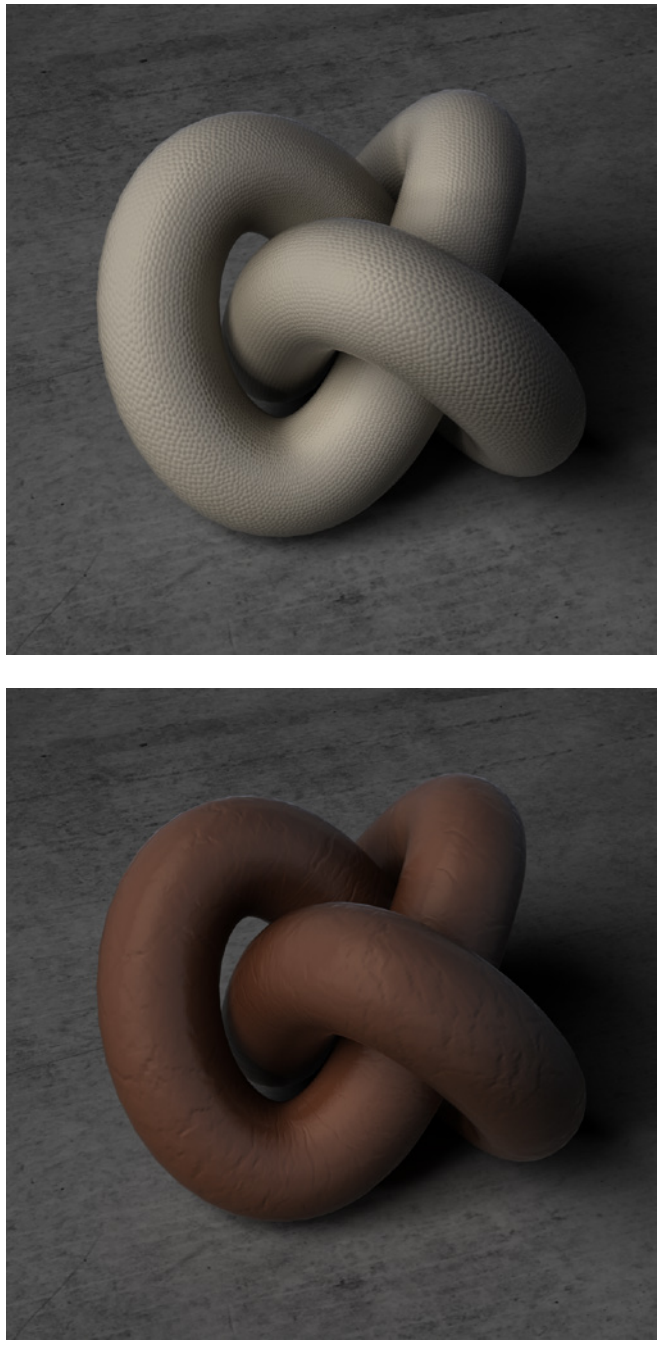


Figura 5.87 Simulación de cuero: a) *Resultado obtenido con la primera textura;* b) *Resultado obtenido con la segunda textura.*

mapa de bits. El procedimiento es, en este caso, algo tan sencillo como arrugar un papel y escanearlo. Calificar a una textura de este tipo como "procedural" es una licencia que espero que se disculpe pues apunta a otra vía de recursos muy interesante: la posibilidad de generar texturas con métodos caseros que pueden resultar muy interesantes y dar lugar a resultados valiosos sin necesidad de hacer kilómetros, cámara fotográfica en ristre, para encontrar una textura que se adapte a lo que nos interesa. Este tipo de recursos se puede utilizar también para generar fondos. Por ejemplo, se puede manchar una hoja de papel con pinceladas o con una esponja, ponerla debajo de la ducha, esperar a que se seque, volver a mancharla con diferentes tonos, volver a ponerla debajo de la ducha y repetir el proceso hasta que el resultado sea de interés. El fondo que he utilizado en algunas de las imágenes de otros apartados se han obtenido de este modo.

Para el primer ejemplo hacer lo siguiente:

1 Desde Photoshop crear un nuevo documento de lados iguales de unos 1.024 o 1.200 píxeles. Comprobar que los colores frontal/fondo son negro y blanco.
2 Aplicar un filtro de tipo textura/vidriera con un tamaño de celda de unos 9 píxeles y una anchura de borde de unos 3 px. Grabar el resultado, que deberá ser similar al de la 5.86 (*a*), en modo escala de grises.
3 Para la segunda textura, escanear un papel arrugado, recortarlo, ajustarlo a 1.024 x 1.024 px y guardarlo también en escala de grises. Véase la figura 5.86 (*b*).
4 En 3ds Max (u otro programa), aplicar estas texturas como mapa de relieve de tipo *Bump*. Así se obtendrán resultados semejantes a los de la figura 5.87.

En ambos casos, si el objeto lo requiere porque es muy grande y es necesario repetir la textura con *Tiling*, convendrá trabajar los bordes, tal como se ha explicado en apartados anteriores para crear una textura "sin costuras" (*seamless*). Pero en otros casos (como en los ejemplos de la figura 5.87) esto no será necesario.

En ambos casos, el color del material es el que se asignará como propiedad básica y, dado que la textura corre a cargo del mapa de *Bump*, esto será suficiente. Pero si se quiere, se puede añadir una textura adicional a la propiedad *Diffuse*.

Texturas combinadas

Las texturas pueden combinarse de varios modos por medio de técnicas que ya hemos visto en apartados anteriores. Este apartado debe verse, por tanto, como una revisión de lo ya explicado pero poniendo el acento en la idea de que las texturas aplicadas a un paramento no deben verse solo como el resultado directo de la aplicación de un mapa de bits sino como el resultado de una combinación de mapas que puede efectuarse de muchos modos. Los tres ejemplos que siguen repasan estas diferentes técnicas para combinar texturas.

a) Texturas recortadas (*decals*) sobre texturas continuas

En la jerga técnica, a una imagen colocada sobre otra sin repetición se denomina un *decal*. Es una abreviación de *decalcomanía* que deriva a su vez del francés *decalquer*, una operación que se puso de moda hacia 1800, que fue introducida por Simon Ravenet hacia 1750 y que consistía en transferir un patrón a una superficie por medio de agua o calor.

Normalmente se utiliza una imagen junto con una máscara en blanco y negro, de tal modo que las partes blancas de la máscara retienen la imagen y las partes negras actúan como canales alfa que dejan ver la superficie sobre la que se aplica. En 3ds Max esto se hace con facilidad por medio de un mapa de tipo *Mix* o con un material *Blend* (mezcla) como ya hemos visto en el apartado sobre mapas y materiales de mezcla.

Partiendo de la base de que contamos con dos mapas de bits como los que se incluyen en la figura 5.88, “muro.jpg” y “cruz.png”, el segundo con un canal alfa incorporado como fondo, la imagen mayor se puede obtener, en principio, de varios modos. Lo más sencillo es utilizar un material *Arch&Design* o *Standard* con un mapa de tipo *Mix*. En este caso la estructura sería:

“muro compuesto” (Arch&Design)
Diffuse: mapa Mix
Color 1: mapa “muro.jpg”
Color 2: mapa “cruz.png”
mapa Mix: “cruz.png”

Y para que el resultado funcione correctamente hará falta hacer lo siguiente:

a) El objeto que recibe el mapa debe tener asignado dos modificadores de mapa UVW. El primero, ajustado a la primera textura, es decir, en el ejemplo, a la anchura y la altura del muro. El segundo, ajustado a la segunda textura, es decir, en el ejemplo, la posición y el tamaño en que queremos que caiga la cruz. Para cada uno de estos modificadores de mapas hay que especificar un canal diferente, pongamos que el canal 1 para el primero y el canal 2 para el segundo.

b) El mapa correspondiente al color 2, “cruz.png” debe tener el *Tiling* desactivado (para que no se repita) y su asignación de canal debe ser la correspondiente al mapa UVW, es decir, en nuestro ejemplo, el 2.

Figura 5.88 Texturas combinadas. Arriba: a) *Textura de piedra;* b) *Pinceladas;* c) Graffiti; d) *Manchas sobre un muro. Abajo: superposición de las cuatro texturas con los métodos descritos en el texto.*

c) El mapa correspondiente a *Mix*, debe tener , como el anterior, el *Tiling* desactivado, su asignación de canal debe ser también la 2 y, por añadidura, debe tener marcada la opción "Alpha" en *Monochannel output* y "Alpha As Gray" en *RGB output*, para que se procese adecuadamente el canal alfa.

Otra alternativa, en principio prácticamente idéntica, pero con más posibilidades, es utilizar un material *Arch&Design* (o *Standard*) con un mapa compuesto de varias capas. En este caso, la estructura básica sería:

material: "muro compuesto" (Arch&Design)
Diffuse: mapa Composite
Layer 1: mapa "muro.jpg"
Layer 2: mapa "cruz.png"
Layer 2: mask "cruz.png"

Como en el caso anterior, el mapa "cruz.png" debe tener el *Tiling* desactivado, estar asignado al canal 2 y, en el caso de la máscara, marcadas las opciones "Alpha" correspondientes.

Por añadidura, se ha hecho que la capa 2, correspondiente a la cruz, tenga la opacidad algo rebajada (85 % en el ejemplo), para que se transparente ligeramente la textura a través de las pinceladas. Y se ha ajustado el color de la textura con el mapa *Color correction* que está integrado en el mapa *Composite*, pues basta hacer un clic sobre el icono correspondiente que está a la izquierda del icono de textura. Para hacer cambios mayores modificar el color por medio del parámetro *Hue tint* y cambiando el valor predeterminado de *Strength* (0 %) a 100 %, y ajustar los valores de saturación, luminosidad y contraste.

b) *Decals* superpuestos

Como decía, la ventaja de utilizar el mapa *Composite* es que nos permite combinar más cosas. Por ejemplo, podemos añadir facilmente otro *decal* al anterior. Todo lo que habría que hacer es repetir el proceso seguido hasta aquí: crear un nuevo modificador UVW asignándole el canal 3, crear una nueva capa sobre las anteriores, asignarle otra textura con un canal alfa y hacer los ajustes finales.

Pero para ver las posibilidades que ofrece el mapa *Composite* seguiré otra vía: crear una nueva capa y asignar directamente la textura a través del mismo canal que la textura del muro pero en modo *Multiply*. Dado que la nueva textura utilizada consiste en letras negras sobre un fondo blanco, al utilizar este modo se retendrán los valores negros mientras que los blancos quedarán transparentes: un valor de 0,5 multiplicado por 1,0 quedará igual, multiplicado por 0,5 (gris), quedará reducido a la mitad, 0,25, y multiplicado por 0,0 o por 0,1 quedará reducido a 0,0 (negro) o 0,05 (casi negro).

Esto nos ahorra manipulaciones adicionales. Y la posición del nuevo *decal* la podemos controlar desde el editor de materiales, desactivando el *Tiling* y cambiando los valores de *U offset* y *V offset*.

La estructura del material quedará, por tanto como sigue (prescindiendo de *Color correction* para simplificar). Tener en cuenta que en la interfaz del editor de materiales aparece en sentido inverso, con la última arriba y que esto se corresponde con el modo en que actúan pues, en principio, la que está arriba oculta todas las demás:

material: "muro compuesto" (Arch&Design)
Diffuse: mapa Composite
Layer 1: mapa "muro.jpg"
Layer 2: mapa "cruz.png"
Layer 2: mask "cruz.png"
Layer 3: mapa "graffiti.jpg"

Las dos primeras capas tienen las mismas características que en el ejemplo anterior (modo de fusión *normal*, canal 1 y 2 con *Tiling* desactivado en la segunda, opacidad al 100 % en la primera y al 85 % en la segunda).

La tercera capa tiene el modo de fusión en *Multiply*, va por el canal 1 con el *Tiling* desactivado y los parámetros *UV offset* ajustados para que caiga donde nos interesa, y la opacidad en un 90%. Así se obtendría el resultado de la figura 5.88 prescindiendo de las manchas que añadiremos en el siguiente apartado.

c) Texturas adicionales superpuestas

La adición de suciedad, polvo, etc., a texturas las hace más convincentes y, para cierto tipo de aplicaciones en que el realismo es necesario, es una técnica básica. El procedimiento es el mismo que hemos utilizado en los ejemplos anteriores. Pero en este caso, debido a las dificultades para recortar adecuadamente algo tan amorfo y disperso como unas manchas, será más sencillo utilizar el último recurso, es decir, partir de una textura con zonas oscuras sobre un fondo claro y con el modo de fusión en multiplicar. Así, las partes blancas quedarán transparentes. Y pueden incorporar zonas de degradados de blanco a negro que suavizan el paso de una a otra textura y pueden resultar particularmente adecuadas para casos tales como manchas en muros.

Esto puede hacerse facilmente utilizando un mapa de tipo *Composite* que ya hemos visto en la segunda sección de este capítulo. También puede hacerse desde Photoshop o Gimp siguiendo exactamente los mismos procedimientos de superposición de capas y métodos de fusión. La ventaja de hacerlo con un mapa de tipo *composite* es que no necesitamos salir del programa y que es más interactivo. La ventaja de hacerlo con Photoshop o Gimp es que puede unificarse el resultado en una textura en formato JPG más compacta.

Si partimos de la estructura anterior todo lo que nos hará falta es, suponiendo que contamos con una textura adecuada, añadir una nueva capa y asignar a esta capa, por el mismo canal, superpuesto al canal 1 por el que va la textura del muro, una capa en modo *Multiply*.

La estructura del material quedará, por tanto, como sigue:

material: “muro compuesto” (Arch&Design)
Diffuse: mapa Composite
Layer 1: mapa “muro.jpg”
Layer 2: mapa “cruz.png”
Layer 2: mask “cruz.png”
Layer 3: mapa “graffiti.jpg”
Layer 4: mapa “manchas.jpg”

Las tres primeras capas tienen la misma estructura que en el ejemplo anterior.

La cuarta capa tiene el modo de fusión en *Multiply* y la opacidad rebajada hasta un 60 %. Así se obtiene el resultado final de la figura 5.88. Junto a esta figura también he incluido todas las texturas utilizadas.

Por último, hay que recordar que para que las texturas superpuestas en modo *decal* puedan colocarse de modo independiente tendrán que discurrir por otro canal. Esto implica que habrá que añadir otros dos modificadores UVW, asignarles los canales 2 y 3 y llevar esta misma asignación al editor de materiales, a la sección de *Coordinates* del mapa, cambiando el canal para que coincida con esta asignación.

5.12 Texturas desplegadas. *Render to texture*

Texturas desplegadas (*unwrap*). Casos

Antes de adentrarse en esta sección convendrá volver a leer la sección sobre este tema del capítulo 4, principalmente los métodos de edición de texturas desplegadas.

Además de esto, tampoco estará de más insistir en algunos conceptos básicos de lo que en algunos programas se denomina *UV Spaces* y en otros, como 3ds Max, *UV Channels*. En 3ds Max podemos incluir hasta 99 canales en un mismo objeto. Esto quiere decir que el objeto almacenará información sobre hasta 99 formas distintas de proyectar cualquier mapa que reciba, por medio de una asignación de un material, siempre que el canal especificado por el mapa coincida con el canal correspondiente a esa proyección. Por otro lado, cuando se colapsa un objeto (convirtiéndolo a malla poligonal), las propiedades de proyección quedan incorporadas a las caras, en un único canal, aunque sean diferentes, mientras no reciban otra especificación

El número de canales que un objeto incorpora puede averiguarse desde el menú *Tools* / *Channel info*. Para comprobarlo, crear un

objeto simple y activar este comando. Nos encontraremos con una tabla con varias columnas (nombre del objeto, número de caras, vértices, etc.). Una de estas columnas, *ID*, muestra el identificador del canal. Y encontraremos al menos los siguientes identificadores: *mesh*/*poly* (datos sobre los vértices y caras del objeto), *vsel* (vertex selection), *-2*:*alpha* (el canal alfa), *0*:*vc* (vertex color), *1*:*map*, el canal predeterminado que, incorpora las proyecciones ya definidas y que se utilizará al añadir un modificador UVW mientras no se modifique. Así, si añadimos sucesivos modificadores UVW a un mismo objeto y, en cada caso, modificamos su canal, estos nuevos canales se irán añadiendo al final de esta lista. En la parte superior hay botones para copiar, pegar y añadir o eliminar canales, entre otras cosas.

Cuando se aplica un modificador *Unwrap* a un objeto que ya tiene varias proyecciones incorporadas, este modificador se sitúa por encima de las proyecciones previas y, en principio, no tiene porqué interferir con ellas, sino integrarlas. Las coordenadas de proyección se almacenan en los vértices de la geometría. El modificador *Unwrap* extrae esta información y la presenta de modo que puedan modificarse las proyecciones sin alterar la geometría subyacente.

Lo primero que debe recordarse es que hay que asignar a este modificador un canal libre pues, si no, lo que haría es substituir a una de las proyecciones previas. De hecho, ocurre lo mismo con un modificador corriente, como es fácil de comprobar. Supongamos que tenemos un objeto con proyecciones previas definidas, por ejemplo, una esfera convertida a malla poligonal con proyecciones envolventes, y un material con texturas asignado. Si aplicamos a este objeto un modificador UVW, como, por defecto, el canal que se asigna a este modificador es 1 y el tipo de proyección es plano, el aspecto del objeto cambiará para adaptarse a una proyección plana. Pero si cambiamos el canal a 2 volverá a su aspecto anterior. Y si le asignamos un material *multi-subobjeto* de dos submateriales, con dos texturas diferentes, al cambiar de canal cambiaremos de textura (lo que nos permitiría, si nos interesara por alguna razón, tener un objeto con varios materiales superpuestos e ir pasando de uno a otro cambiando de canal, como quien sintoniza una emisora).

El modo más seguro de saber qué canal deberemos asignar es activando el comando *Channel info* (menú *Tools*), tomando nota del último canal de la lista y escogiendo un núméro superior. Al cambiar el canal en los parámetros de *Unwrap* nos encontraremos con un aviso, “Channel Change Warning”, que nos obliga a escoger entre dos opciones: “Move” (predeterminada, mover los UV al canal elegido) o “Abandon” (no aplica cambios al modificador). Elegir la segunda, “Abandon”, aunque en realidad tanto da mientras se desplieguen adecuadamente las caras. Al aplicar el modificador *Unwrap* las coordenadas previas se almacenan en el modificador. Y si no hay coordenadas previas se crean nuevas en modo plano. Al evaluar el modificador, el programa procesa las coordenadas en el orden en que están creadas.

Todo esto se entenderá mejor por medio de un ejercicio simple en el que asignaremos diferentes proyecciones a las cinco caras de un objeto desplegado hacia la cámara.

Pero antes debemos tener en cuenta lo siguiente. Como vamos a aplicar diferentes texturas a las cinco caras tendremos que asignar a cada cara un modificador de proyecciones de coordenadas adecuada. Para ello, podemos seguir dos procedimientos que ya hemos visto.

Uno, consistiría en separar todas las caras, aplicar a cada una de ellas un operador UVW con la orientación adecuada, luego aplicar a cada una de ellas un material con un mapa ligado a la textura correspondiente y, luego, colapsar una de las caras para que el operador UVW quede incrustrado (si no se hace así, al hacer la operación de *Attach* las caras que se integren en esta recibirán las proyecciones de la cara de que se ha partido). Luego volver a juntarlas (editando la malla, presionando el botón *Attach* y seleccionando los diferentes objetos separados). Al juntarlas, las proyec-

ciones se incrustran en el objeto integrado y las distintas caras reciben, automáticamente, diferentes ID de material.

Otro procedimiento consistiría en editar el objeto y asignar a cada una de las caras, manualmente, un diferente ID de material, aplicar al objeto cinco operadores UVW, editar cada uno de ellos y darles la orientación adecuada y asignarles un canal diferente, cuyo número coincida con el identificador correspondiente y, por último, asignarles un material multisubobjeto de cinco materiales con sus correspondientes mapas, cada uno de ellos con una asignación de canal diferente y que se corresponda con el ID de las caras y con el canal del operador UVW. El resultado al que llegaríamos sería el mismo en los dos casos: un único objeto con un material múltiple compuesto por cinco mapas de bits diferentes.

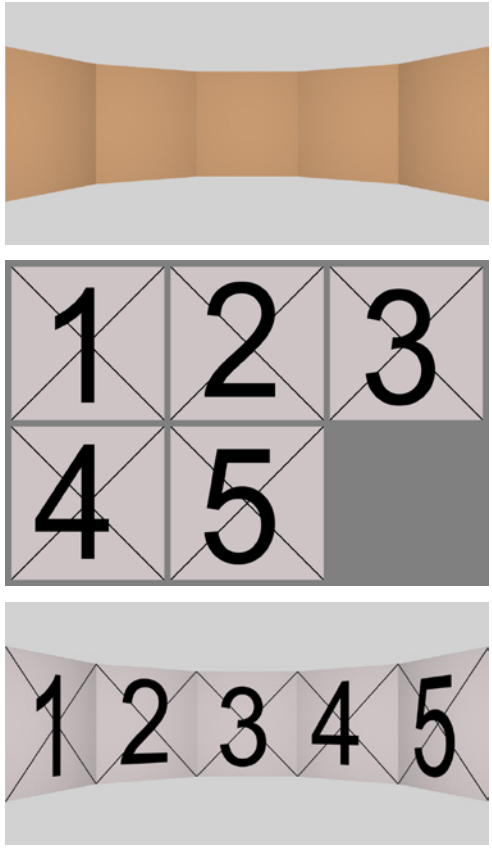


Figura 5.89 Texturas desplegadas con Unwrap. *Ejercicio inicial:* a*) Escena original;* b*) Mapa de la textura desplegada;* c*) Escena final, común a los dos métodos descritos en el texto.*

Hay, con todo, una diferencia entre ambos métodos. Con el primer método, integrando las proyecciones, habrá 1 solo canal (se puede comprobar desde *Channel info*) y no será necesario cambiar los canales del material multisubobjeto pues las propiedades de proyección han quedado asociadas a las diferentes caras que comparten un mismo canal. Con el segundo método, dejando explícitas las proyecciones, con cinco modificadores *UVW Map* superpuestos, habrá cinco canales (se puede comprobar igualmente desde *Channel info*) y será necesario cambiar los canales del material multisubobjeto, es decir, que el mapa del submaterial 1 tendrá que ir por el canal 1 (modificando esta asignación en la sección *Coordinates* de los parámetros del mapa), el mapa del submaterial 2, por el canal 2, etc. Y estos canales y el requisito de cambiar los canales del material multi se mantendrán aunque se colapsen.

En cualquier caso, estos dos métodos tienen dos inconvenientes. El primero, que no es demasiado importante, es que resultan algo laboriosos. No es demasiado importante porque muchos lectores pensarán que el método que voy a describir a continuación tampoco es demasiado sencillo (aunque es más largo de explicar que de ejecutar, una vez que se asimila adecuadamente). El segundo, más importante, es que nos encontramos con que tenemos que gestionar cinco texturas para un mismo objeto cuando sería bastante más práctico tener que gestionar solo una. En una escena con múltiples objetos y, más aún, en modelos que vamos a compartir con otros por medio de referencias externas, esto es una fuente de problemas pues cualquier ajuste será difícil de localizar y ajustar correctamente.

Procediendo con el ejemplo, se trata de llegar a un mismo resultado por dos vías. Previamente, crear un objeto como el de la figura

5.89 (un polígono de 10 lados, convertido a *spline* y extruido). Situar una cámara y una luz de modo que se muestren las cinco caras iluminadas de un modo uniforme. Continuar como sigue, probando las dos alternativas, una con varios mapas y la otra, con *Unwrap* y un único mapa:

a) Cinco proyecciones y un mapa multisubobjeto con cinco mapas:

1 Desde Photoshop o Gimp, crear cinco mapas cuadrados diferentes, darles nombres adecuados y situarlos en la carpeta de proyecto. En la figura del ejemplo son mapas sencillos con un número en el centro.
2 Desde 3ds Max, asignar proyecciones planas a las caras de alguna de las dos maneras que se han descrito antes (cinco caras diferentes unidas o cinco modificadores aplicados a los subobjetos poligonales).
3 Crear un material multisubobjeto de cinco materiales. Hacer que cada submaterial sea de tipo *Standard* con un mapa en *Diffuse* diferente para cada cada submaterial, utilizando los mapas creados en el primer paso. Asignar este material multi al objeto.
4 *Render*. Así se obtendrá un resultado como el de la figura.

b) Una única proyección y un único mapa:

1 Desde Photoshop o Gimp, crear un mapa que agrupe a los cinco anteriores, tal como se muestra en la figura 5.89 (*b*).
2 Desde 3ds Max, asignar al objeto original un modificador *Unwrap*. Editarlo (botón *Open UV Editor*). Seleccionar todas las caras (subobjeto *Faces*, Ctrl+A) y aplanarlas (menú *Mapping* / *Flatten mapping*).

El modo *Flatten mapping* hace que las coordenadas UVW se proyecten sobre elementos cuyas caras se encuentran en un ángulo mayor de 45º. Las aristas de estos elementos pasan a ser las nuevas *seams*. Si se aumentara el ángulo, pongamos que a 75º, habrá menos elementos pequeños dispersos en el caso de formas con caras no ortogonales. Otra variante más útil en otros casos, y que veremos más adelante, es menú *Mapping* / *Normal mapping*. En este caso, se puede elegir la dirección de proyección sobre uno de los tres ejes.

3 Crear un material *Standard* con un mapa en *Diffuse*. Escoger como mapa el que hemos creado en el primer paso. Asignar este material al objeto. Si se visualiza el resultado inicial será caótico.
4 Volver a editar el *Unwrap*. En la lista desplegable de la parte superior derecha, seleccionar como textura el mapa que acabamos de asignar para que aparezca como fondo. Activar el modo subobjeto *Faces*. Desde el visor de la escena, seleccionar la cara 1. En el editor *UV* de *Unwrap* esta cara aparecerá resaltada. Moverla para que caiga aproximadamente en la zona correspondiente del mapa que aparece como fondo y, si es necesario, rotarla 90º con las herramientas del panel lateral *Quick transform*. El visor de la escena mostrará los cambios que se aproximarán al resultado que buscamos. Hacer lo mismo para las otras cuatro caras. Cuando estén encajadas, acercarse a las esquinas con zoom y, con la herramienta *Freeform*, ajustar las caras para que coincidan con el fondo.
5 *Render*. El resultado será idéntico al anterior. Pero ahora tenemos una única proyección, con un único material estándar y con un único mapa.

Integración de texturas. Ejemplo 1. Casos simples (misma orientación)

El uso de texturas desplegadas con *Unwrap* es particularmente adecuado cuando nos encontramos con varios elementos similares pero con texturas distintas. En principio, esto nos obligaría a gestionar varios objetos, con varios materiales y con varias texturas, lo que complica la organización.

Si los objetos tienen la misma orientación, el uso de *Unwrap* es muy simple, más aún de lo que hemos visto en el ejercicio previo del apartado anterior. Un ejemplo característico es el de un edificio con varias puertas, ventanas o persianas. Partiremos de este último ejemplo: un modelo de un edificio que incluye diferentes persianas en las ventanas. El procedimiento para este caso sería como sigue.

1 Desde Photoshop o Gimp, crear una textura compuesta, "persianas.jpg" unificando varias texturas, con una pequeña separación entre ellas. Darle un tamaño adecuado para que las juntas entre lamas tengan al menos un par de píxeles. El resultado debería ser similar al de la figura 5.90 (b).
2 Desde 3ds Max, unificar todas las persianas a las que se quiere aplicar esta textura en un único objeto "persianas": convertir una de ellas a malla editable y añadirle todas las demás pulsando el botón *Attach* y haciendo un clic sobre cada una de ellas (o pulsando *Attach list* y seleccionándolas por nombre).
3 Asignar a este objeto un material *Arch&Design* o *Standard* con la textura compuesta que hemos creado ("persianas.jpg") aplicada a *Diffuse*. Asignarlo al objeto "persianas". No hacer caso del resultado o de los mensajes que puedan aparecer.
4 Seleccionar el objeto y asignarle un modificador *Unwrap UVW*. Editarlo (botón *Open UV Editor*). Seleccionar todas las caras (subobjeto *Faces*, Ctrl+A) y aplanarlas (menú *Mapping* / *Flatten mapping*). En la lista lista desplegable de la parte superior derecha, seleccionar como textura el mapa "persianas.jpg" para que aparezca como fondo. Proceder como en el caso anterior (activar *Faces*, seleccionar una de las caras, moverla y rotarla hasta que el visor de la escena muestre la posición correcta, repetir el procedimiento con todas las caras). De este modo se conseguirá relacionar libremente las caras geométricas con zonas determinadas de una textura general.

Figura 5.90 Integración de texturas con Unwrap. *Objetos con la misma orientación. Un solo objeto tiene asignado un solo material con un mapa.*

5 *Render*. El resultado deberá ser como el de la figura 5.90 (a).

Así hemos llegado a este resultado final con un único material y un único mapa, lo que resulta mucho más sencillo de gestionar.

Integración de texturas. Ejemplo 2. Casos generales (diferente orientación)

El mismo método se puede aplicar a un caso más general, en que los planos tengan diferentes orientaciones y que es algo más complicado de desarrollar. Antes de describir un ejemplo particularmente importante, el de una fachada con diferentes elementos, ménsulas, cornisas, losas de balcones, etc., convendrá analizar un ejemplo muy simple pero muy ilustrativo.

Crear un objeto como el de la figura 5.91, un prisma con una zona extruida. Crear también en Photoshop o Gimp (o descargar algo similar de internet) un patrón cuadrado con 8 x 8 casillas, con letras y números, A1…A8 en la primera fila y H1…H8 en la última. Este patrón nos servirá para identificar con facilidad donde se coloca la textura sobre el modelo.

1 Crear un material *Standard* o *Arch&Design* y asignar este mapa a *Diffuse*. Aplicarlo al objeto. El resultado será como el de la figura 5.91 (*a*), es decir, una proyección que solo sirve para el plano principal pero no para los otros.
2 Aplicar al objeto un modificador *Unwrap*. Editarlo (botón *Open UV Editor*).
3 Desde el panel *Modify* / *Unwrap*, activar el modo subobjeto *Face* (descolgar los subobjetos disponibles de *Unwrap*) y, desde el visor de la escena, seleccionar las caras frontales (en un caso más complejo convendrá utilizar herramientas de selección automática, por ejemplo, seleccionando una cara central y presionando el botón *Grow*, en el grupo *Selection*).

Si, tras seleccionarlas, desde el UV Editor intentamos mover las caras seleccionadas veremos que, por un lado, no están orientadas como nos interesan y, además, al moverlas, arrastran a otras caras a las que están pegadas. Por tanto, lo que nos interesa es reorientarlas e independizarlas.

4 Desde el panel *Modify* / *Unwrap*, en la sección *Projection*, presionar el botón *Planar mapping*. Con esto, se aplica a una proyección plana, según el eje seleccionado, a la selección, que, por añadidura, se convertirá en un grupo independiente. Volver a presionar el botón para liberar la selección.
5 Las caras seleccionadas aparecerán resaltadas en el *UV Editor*. Ahora podemos mover estas caras de modo independiente. Sacarlas fuera de la zona central.
6 Seleccionar la cara superior, horizontal, del elemento central extruido y hacer otro tanto (*planar mapping* y desplazamiento fuera de la zona).
7 Seleccionar todas las caras restantes y hacer otro tanto.

Así tendremos tres grupos de caras, las correspondientes a “la fachada”, las correspondientes a la “losa superior del balcón” y el resto que, para este ejemplo, no nos importa, por lo que podremos minimizarlo.

8 Activar el fondo, seleccionando (de la lista desplegable, arriba a la derecha) el mapa con la textura del patrón que hemos asignado al material y al objeto.
9 Seleccionar las caras del primer grupo y llevarlas de nuevo a la zona central. Luego seleccionar las tres caras superiores y, utilizando la herramienta *Freeform*, moverlas y reescalarlas hasta que caigan sobre las partes de la imagen de fondo que nos interesa (filas que comienzan por A, B y C). Luego seleccionar las tres caras inferiores y hacer otro tanto (filas que comienzan por F, G y H).
10 Seleccionar las caras del segundo grupo y hacer otro tanto (filas que comienzan por D y E).

11 Por último, seleccionar las caras restantes, superponerlas, reescalarlas hasta que ocupen menos de uno de los cuadrados del patrón y llevarlos a una zona gris.
12 Así llegamos a una reconfiguración como la de la figura 5.91 (*b*) que se traduce en un resultado final como el de la figura 5.91 (*c*).

§ § §

En el caso de una fachada, podemos hacer más interesante el proceso si integramos en la textura efectos de iluminación indirecta para que la fachada no quede uniforme sino con zonas más oscuras que corresponderían a zonas a las que, en cualquier caso, llegaría menos la luz. Pero esta es precisamente la utilidad de *Render to texture*, que veremos más adelante.

El caso de una fachada de estas características es, por tanto, un caso límite en el que no está claro si es preferible generar previamente una textura desplegada y luego adaptar las proyecciones a esta textura o bien aplicar diferentes texturas a todos los elementos y obtener una textura desplegada por métodos automáticos, con *Render to texture*. Para que se puedan apreciar mejor estas diferencias desarrollaré este ejemplo con texturas desplegadas. Con los ejemplos que se dan más adelante será fácil entender qué habría que hacer para llegar a un resultado similar con *Render to texture*. El lector deberá decidir, en cada caso, que método será preferible.

El proceso que seguiremos será el siguiente. En primer lugar, hay que crear una textura a la medida del muro incorporando los colores y las texturas de todos los elementos que la integran. En segundo lugar, se puede enriquecer esta información con un método muy sencillo de "iluminación" indirecta sin luces: *Ambient occlusion*. Este método que se describe más ampliamente en el libro sobre simulación de la iluminación, computa la oclusión de las superficies, enviando rayos en torno a un punto y genera un mapa con zonas más o menos oscuras en función de la mayor o menor oclusión. Luego integraremos estos dos mapas y añadiremos, si interesa hacerlo, otras texturas o mapas de elementos tales como puertas o ventanas.

Para que el ejemplo se pueda reproducir con facilidad he simplificado la fachada, que se ha reducido a un muro con solo dos huecos, una sola losa y una sola cornisa. Pero el mismo método se aplicaría a fachadas más complejas. Partimos, por tanto, de un objeto único como el que se muestra (en su estadio final) en la 5.92. El método se desarrolla en cuatro fases. En las dos primeras se generan dos mapas desde 3ds Max. En la tercera se combinan y se complementan estos mapas desde Photoshop o Gimp. En la cuarta, de nuevo desde 3ds Max, se aplica el mapa

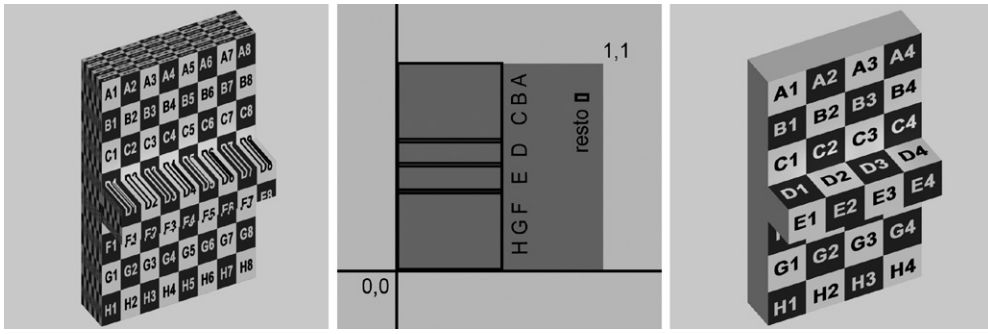


Figura 5.91 Integración de texturas con Unwrap. *Ejemplo 2 (previo):* a) *Resultado de aplicar directamente una textura con un patrón regular a un modelo con caras de diferente orientación;* b) *Reorganización de las proyecciones con* Unwrap*;* c) *Resultado final.*

del modo habitual a través de un material corriente.

a) Generación de un mapa con los colores y texturas de la fachada.

1 Asignar al muro los materiales que queremos incluir en la representación (jambas, ménsulas, cornisas, elementos decorativos, etc.) pero con colores simples, para que puedan distinguirse con facilidad a la hora de generar la textura.
2 Crear una luz simple que ilumine la fachada con un ligero ángulo, desde arriba, para que la distribución sea algo más luminosa en la parte superior que en la inferior. Desactivar sus sombras y comprobar que abarca bien toda la fachada.
3 Preparar un visor para representar el muro. Ajustar la resolución y la proporción de salida para que su proporción sea la misma que la del muro. Activar los marcos de visor (*Show safe frames*, por teclado Shift+F) para controlar este ajuste. Crear una cámara ortográfica que apunte al centro del muro (crear una cámara normal dirigida al muro y luego cambiar su tipo a "orthographic"), y ajustar la posición y el ángulo adecuado para que abarque el muro, ajustándose lo más posible al marco. Se podría partir de una vista frontal pero esta es más inestable y vamos a grabar dos vistas que luego deberán superponerse, por lo que es preferible usar una cámara.
4 Desactivar *Final gather* y el control de exposición.
5 *Render*. Grabar el resultado con un nombre tal como "FachadaColor.jpg".

b) Generación de un mapa a medida con sombras integradas.

1 En el editor de materiales crear un material de tipo *mental ray* (también se puede utilizar un material *Standard* o *Arch&Design* y utilizar la última sección *mental ray connection* que tendrá exactamente el mismo aspecto que la de un material de tipo mental ray). En la sección *Basic shaders*, pulsar el botón *Surface* y escoger un mapa de tipo *Ambient / Reflective occlusion*.
2 Configurar inicialmente este mapa modificando tan solo el valor de *Max distance* a unos 25 o 50 cm. Este parámetro tiene como efecto que los rayos de rastreo del mapa de oclusión lleguen más o menos lejos. También habrá que ajustar el valor de *Spread* que dispersa o concentra los rayos. En principio, el valor prederminado (0,75) es aceptable. El resto de valores se pueden dejar de momento como están (blanco y negro para el mapa de oclusión) y 16 para el número de muestras (habrá que subirlas al final).
3 Abrir *Render setup* por la sección *Processing* / *Translator options*. Activar *Material override*. Arrastrar el material que acabamos de crear, desde el editor de materiales sobre el botón que hay junto a *Material override*. Escoger *Instance* para el modo de copia. Con esto, este material se superpondrá a todos los materiales de la escena mientras esté activada esta opción de *Render setup*.
4 Hacer un *render* de prueba y analizar el resultado. Modificar los valores según lo que interese. Quizás haya que aumentar o reducir el parámetro *Max distance* o hacer que el color negro sea un gris oscuro para reducir el contraste. Y modificar ligeramente el valor de *Spread* para que las muestras se dispersen más o menos. Por último, aumentar el valor de *Samples* para que el resultado tenga más calidad, aunque tardará algo más. La figura 5.92 está configurada con *Max distance* 45,0 cm, *Spread* 0,8, *Dark* 0,25 / 1,0 (HSV 0, 0, 0,25) y *Samples* 32.
5 *Render*. Grabar el resultado con un nombre tal como "FachadaBN.jpg".

c) Integración, edición y ampliación de los mapas.

1 Desde Photoshop o Gimp abrir las dos imágenes y copiar la de *Ambient occlusion* sobre la de color.
2 Cambiar el modo a *Multiplicar* y reducir la opacidad hasta el 65 % más o menos (probar valores adecuados).
3 Acoplar la imagen.
4 Revisar el resultado, retocar zonas que no hayan quedado como queramos. Añadir, si interesa, otros efectos.

Si solo nos interesara utilizar el mapa de la fachada, lo que habría que hacer a continuación es recortar y guardar el resultado como mapa listo para ser aplicado. Pero como vamos a ampliar el mapa esto no es necesario. Por tanto, continuar así:

5 Añadir texturas adecuadas para el suelo y el techo de las losas de los balcones y, si se considerara necesario, otros elementos perpendiculares al plano de fachada.

Así llegamos a un mapa como el que se muestra en la figura 5.92 (a)

En este ejemplo no he incluido otros elementos que el suelo y el techo de las losas. Esto quiere decir que los laterales de elementos secundarios quedarán barridos por el color del píxel del borde. Más adelante explico métodos adicionales para solucionar este problema. Pero los métodos citados hasta aquí serán suficientes en la gran mayoría de los casos. Si ni estos ni los adicionales fueran suficientes, habría que añadir elementos laterales pero entonces nos encontraríamos en un caso límite en el que, como decía antes, puede ser preferible utilizar métodos automáticos como los que se describen en *Render to texture*.

d) Aplicación del mapa a la fachada.

Una vez que contamos con este mapa, desde 3ds Max, hacer lo siguiente.

1 Aplicar a la fachada un modificador *Unwrap UVW*.
2 Comprobar, en *Channel info* (menú *Tools*) cuál es el último canal asignado al objeto y asignar al canal de *Unwrap* el siguiente número. Responder *Abandon* al mensaje de advertencia.
3 Crear un material de tipo *Standard* o *Arch&Design* y asignar a *Diffuse* el mapa desplegado que hemos creado en el paso anterior. Aplicar este mapa a la fachada (previamente, si no se ha hecho antes, integrar en un único objeto todos los elementos que la componen y cuyas texturas o colores hemos combinado en el mapa desplegado). Asignarle el mismo canal que hemos utilizado para *Unwrap*.
4 Entrar en el *UV Editor*. Seleccionar todas las caras.
5 Desde el panel *Modify* / *Unwrap*, en la sección *Projection*, pulsar el botón *Planar mapping*. Con esto, se aplica a una proyección plana, según el eje seleccionado X o Y, a la selección, que se convertirá en un grupo independiente. Comprobar que el eje elegido es el adecuado para que el plano de proyección quede paralelo a la fachada.
6 Volver a pulsar este botón para liberarlo (mientras no se desactive no se puede continuar).
7 Desde el *UV Editor*, seleccionar la cara correspondiente a una de las puertas. Si no se pudiera mover de modo independiente, repetir la misma operación (*Projection* / *Planar mapping*). Luego desplazar esta cara fuera de la zona principal.
8 Repetir esta misma operación para la otra puerta.

Pero no podemos hacer lo mismo con los elementos correspondientes a la parte superior o inferior de la losa del balcón porque habrán quedado orientados según la proyección paralela a la fachada y todo lo que veremos será una línea. Y si quisiéramos desplazarlos, tampoco podríamos porque han quedado ligados a las caras que les rodean. Así que tenemos que repetir el procedimiento anterior para reorientarlos y desplazarlos.

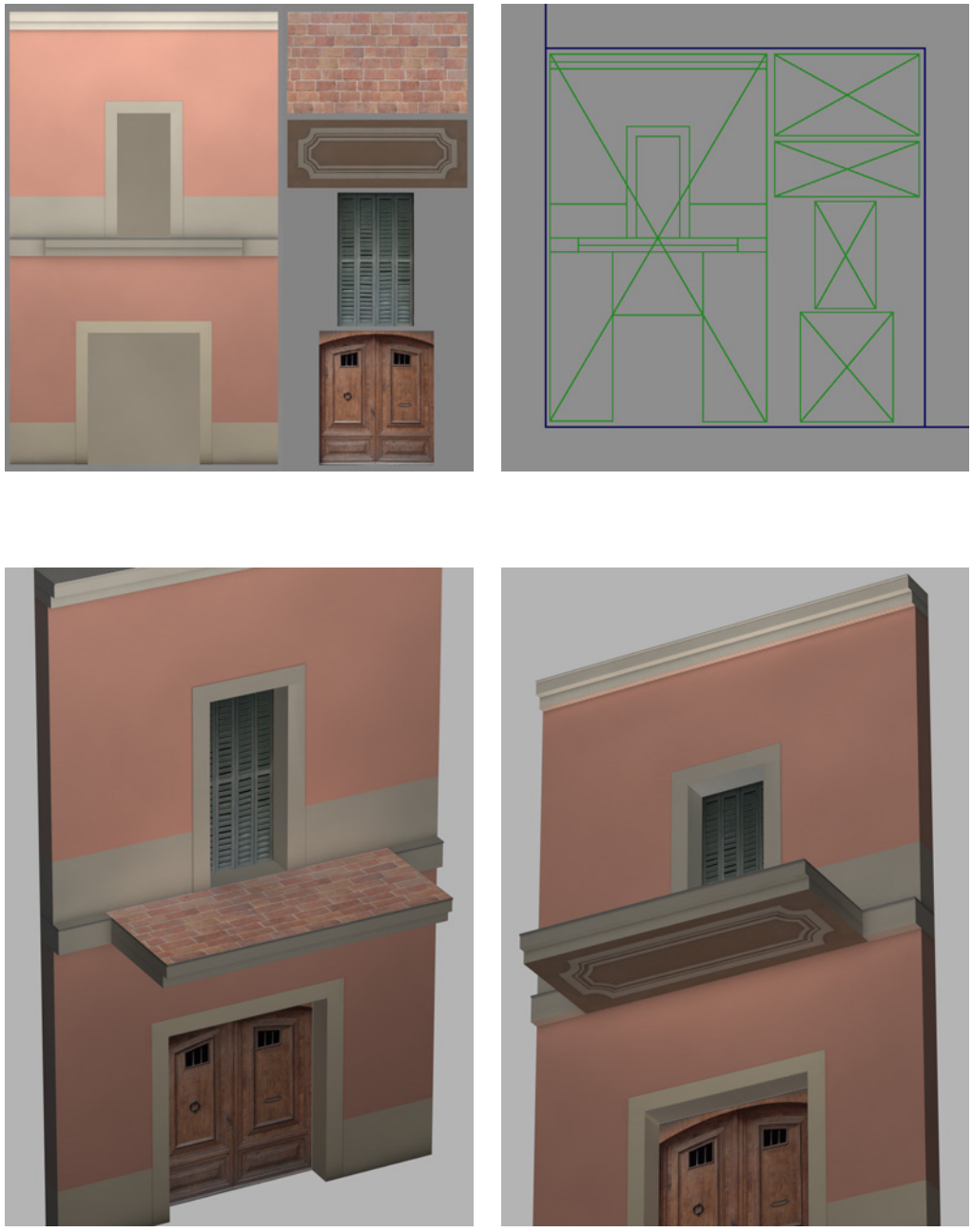


Figura 5.92 Integración de texturas con Unwrap. *Ejemplo 2:* a*) Mapa utilizado, con las texturas desplegadas;* b*) Estructura de* Unwrap*, con grupos que se corresponden con las texturas principales;* c*) Vista superior del modelo;* d*) Vista inferior del modelo.*

9 Desde la escena, seleccionar la cara superior de la losa.
10 Desde el panel *Modify* / *Unwrap*, en la sección *Projection*, pulsar el botón *Planar mapping*. Marcar, en este caso, el eje *Z*. Volver a pulsar este botón para liberarlo.
11 Desde el *UV Editor*, seleccionar la cara correspondiente y desplazarla fuera de la zona principal.
12 Repetir el mismo procedimiento para la cara inferior de la losa.

Así tendremos cinco grupos de elementos. Todo lo que queda por hacer es disponerlos de modo que se correspondan con el mapa de bits, tal como se muestra en la figura 5.92 (*b*).

13 Seleccionar cada uno de los cinco grupos y escalarlos, rotarlos (seleccionar una arista para comprobar su orientación) y desplazarlos para que queden aproximadamente tal como se muestra en la 5.92 (*b*).
14 Activar el fondo y, desde la lista desplegable del *UV Editor*, seleccionar la textura desplegada que hemos asignado al material aplicado al objeto.
15 Mover y ajustar cada uno de los grupos hasta que la correspondencia con la textura desplegada sea suficiente.

De este modo será sencillo conseguir una buena correspondencia entre coordenadas de proyección, textura y geometría.

Sin embargo, los retranqueos pueden no quedar bien pues la textura se proyecta arrastrando el color del último píxel a lo largo del plano. Como ya he dicho, en algunos casos esto es imperceptible y no merece la pena ajustarlo. Pero en otros es visible y malogra el resultado.

En casos extremos, si el retranqueo es muy amplio, habrá que utilizar proyecciones adicionales, como hemos hecho con la losa de los balcones. Pero si hay pequeños retranqueos hay que buscar soluciones más expeditivas. Y una solución sencilla es escalar ligeramente las caras adyacentes. Al hacer esto, como las caras siguen unidas en el grupo que hemos creado, el resultado indirecto es que las caras transversales se inclinan hacia dentro y reciben la proyección de la textura con lo que se elimina el efecto de barrido. Otra alternativa es seleccionar las arisas o vértices internos y desplazarlos ligeramente hacia dentro.

Para este tipo de operaciones hay que trabajar con dos vistas: la que muestra en el *UV Editor* los vértices, aristas o caras correspondientes a las coordenadas de proyección y la que muestra, en la escena, estos mismos elementos. Puede convenir, para la escena, utilizar el atajo de teclado F2 para visualizar o no las caras seleccionadas, según lo que interese para hacer el ajuste. Y, desde el *UV Editor*, utilizar atajos como +Shift para que el movimiento sea ortogonal, o +Alt+Ctrl, para escalar uniformemente hacia adentro o hacia afuera.

Hay múltiples variantes en que no puedo entrar por falta de espacio. Puede, por ejemplo, ocurrir que después de haber ajustado un elemento, nos interese ocultarlo o desplazarlo para poder trabajar con otros elementos cercanos. Una alternativa eficaz para estos casos es seleccionar el elemento o elemento y desplazarlos una distancia fija (+1,0 en U, por ejemplo) con lo que, si el *tiling* está activado, la proyección se mantendrá y, si no, será sencillo devolverlas a su posición previa introduciendo el valor inverso.

Control de la posición en texturas sobre superficies curvas irregulares. Ejemplo 3

El uso de modificadores de tipo *Unwrap* también se utiliza para casos en los que no es posible ajustar la posición de las texturas por los métodos corrientes (desplazando el *gizmo* o cambiando numéricamente los valores de *tiling* u *offset* desde el editor de materiales). Esto ocurre principalmente cuando la superficie es irregular y carecemos de puntos de referencia adecuados.

En general, hay dos vías generales de utilizar texturas desplegadas: adaptar las coorde-

nadas UV al mapa de bits o adaptar el mapa de bits a las coordenadas UV. En los ejemplos anteriores hemos utilizado la primera vía. En los casos en que no es posible ajustar la textura con facilidad, como lo que ocurre cuando la superficie es irregular, suele ser preferible utilizar la segunda vía.

El ejemplo que sigue ilustra este caso. Hay una cuestión previa importante y es que, en estos caso, es conveniente que el modelo geométrico de base sea regular pues esto facilita el trabajo. Más concretamente, que esté formado por *quads* (polígonos cuadrados) no por triángulos. Pero muchos modelos irregulares se crean a partir de deformaciones de una primitiva básica, como una esfera. Y, si partimos de una esfera, la zona de los polos quedará formada por triángulos pequeños con sus vértices unidos en los polos. Esto es una fuente de todo tipo de problemas, incluyendo efectos de simulación de reflejos que pueden dar lugar a imperfecciones. Y también nos dificultará considerablemente el trabajo con mapas desplegados que es a lo que vamos. Necesitamos partir de una esfera compuesta por *quads*.

Un modo de hacerlo, desde 3ds Max, es crear un cubo con un segmento por lado. Aplicarle un modificador *Turbo smooth* con tres o cuatro iteraciones (con tres será suficiente pero cuatro aguantará mejor las deformaciones). Y añadirle un modificador *Spherify*. De este modo conseguiremos una esfera casi perfecta, indistinguible de una esfera paramétrica, pero cuya geometría interna está formada por *quads*.

Una vez que contamos con este objeto básico, será fácil deformarlo según interese aplicando modificadores de tipo FFD (*Free Form Deform*) . El objeto de la figura 5.93 se ha generado de un modo similar, aunque no exactamente igual (no he utilizado el modificador *spherify*). Lo que he hecho es crear un prisma de anchura doble que la longitud, aplicarle un modificador *Turbo smooth* con tres iteraciones y luego deformarlo con un *FFD* de 3 x 3 x 3.

Al aplicar *Unwrap* a un objeto de este tipo no será muy útil utilizar las técnicas que hemos visto en los ejemplos anteriores. Pues si aplanamos las caras por medio de proyecciones planas estas proyecciones no se corresponderán más que con una pequeña parte de la superficie curva. Además, nos encontraremos con docenas de superficies que no podremos controlar.

Si la textura es informe y se extiende por toda la superficie podemos utilizar métodos algo más sencillos de los que siguen. Pero si es regular y nos interesa controlar la posición de determinadas partes de esta textura, necesitaremos algún medio de hacer dos cosas: dividir la superficie en unas pocas zonas que se correspondan con zonas de textura que queremos controlar y "planchar" estas zonas para poder relacionarlas adecuadamente con la textura que vamos a crear.

En 3ds Max hay una serie de herramientas, que forman parte del modificador *Unwrap*, que nos permiten hacer precisamente esto. Son herramientas que permiten crear con facilidad "costuras" (*seams*) sobre la superficie, a partir de las cuales se pueden crear grupos y que permiten utilizar los grupos así creados como si fueran "pieles" (*pelts*) que se pueden "planchar" o "estirar" (*stretch*) para relacionarlas con mapas de bits o para crear mapas de bits a partir de ellas.

El procedimiento será, por tanto, sintetizando lo principal, como sigue: *a*) Crear costuras (*seams*) sobre la superficie que se correspondan con la textura que queremos aplicar (véase las imágenes de la figura 5.93); *b*) Crear grupos a partir de estas costuras; *c*) Aplanar estos grupos; *d*) Situar los grupos así aplanados sobre la zona principal del *UV Editor* y hacer un *render* para captar esta estructura final; *e*) Desde un programa externo de pintura digital crear una textura a partir de esta estructura; *f*) Utilizar esta textura como mapa aplicado al objeto.

Las explicaciones que daré serán muy sintéticas, por razones de espacio, aunque espero que suficientes para comprender lo

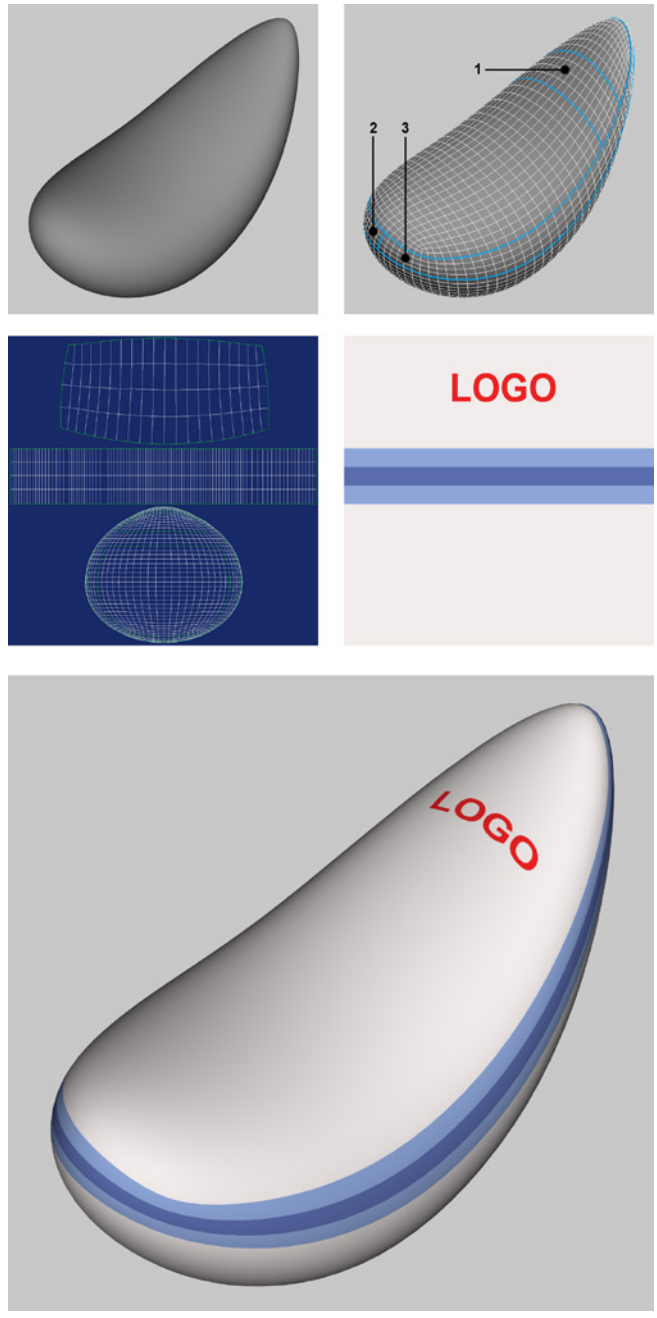


Figura 5.93 Texturas sobre superficies de curvatura libre con Unwrap. *Ejemplo 3:* a) *Objeto inicial;* b) *Líneas de costura* (seams) *que crean 3 zonas diferenciadas sobre el objeto;* c) *Vista del* UV Editor *con las 3 zonas desplegadas y encajadas en la zona principal;* d) *Mapa de bits generado a partir de la estructura anterior;* e) *Resultado final.*

principal de esta técnica. Si el lector las encuentra excesivamente concisas puede consultar la ayuda del programa o los tutoriales que se pueden encontrar en internet sobre este tema.

1 Crear un objeto como el de la figura 5.93, que representa una superficie de curvatura libre.
2 Aplicar al objeto un modificador *Unwrap*. Inicialmente es posible que muestre unas líneas verdes que se corresponden con los bordes de las proyecciones predeterminadas. Se pueden desactivar desde *Unwrap* / *Configure* / *Display*, "Map Seams". Pero puede ser más conveniente volver atrás y, antes de aplicar *Unwrap*, borrar las trazas de mapas anteriores aplicando un modificador *UVW mapping clear* y luego volver a convertir el objeto a malla editable.

En cualquier caso, vamos a crear *seams* a lo largo de las aristas que nos interesen. Para ello podemos utilizar varios métodos que se agrupan en dos principales: *i*) seleccionar grupos de aristas desde la sección *Selection* (añadiendo con Ctrl o barriendo con una ventana o usando la opción *Loop selection*) y luego convertirlas a *Seams*, desde la sección *Peel* con la herramienta "Convert Edge Selection to Seams"; *ii*) Directamente desde la sección *Peel*, con la herramienta "Point-to-Point Seams" con la que basta con marcar puntos que estén situados a lo largo de una serie de aristas yuxtapuestas. Cualquier de los dos métodos es muy sencillo y rápido.

3 En modo subobjeto *Edge* (del panel del modificador *Unwrap*) seleccionar las aristas correspondientes a las zonas que queremos separar, con alguno de los procedimientos citados en el párrafo anterior. Las zonas así creadas se resaltarán en color azul. Véase la figura 5.93 (*b*).
4 En modo subobjeto *Face* seleccionar una cara cualquiera de esta zona y, desde la sección *Peel*, pulsar el botón "Expand Face Selection to Seams". Todas las caras de la zona quedarán seleccionadas.
5 Entrar en el *UV Editor* donde las caras aparecerán resaltadas en rojo.
6 Desde el panel *Unwrap* (o desde el *UV Editor*), activar *Quick peel*. Las caras se expandirán aunque no del todo. Desplazarlas fuera de la zona principal. Desde la sección *Peel* del panel *Unwrap* pulsar el botón, que habrá aparecido tras esta operación, *Pelt map*. Con esto se abrirá un nuevo panel, *Quick pelt*. Desde este nuevo panel, pulsar el botón *Start pelt*. Como este ejemplo es muy sencillo el proceso durará solo unos instantes y se estabilizará (en casos más complejos lleva cierto tiempo aplanar las proyecciones). El botón *Start Pelt* habrá cambiado a *Stop Pelt*. Pulsarlo para detener el proceso. El botón *Start relax* se utiliza a veces para eliminar caras superpuestas, en este estadio, pero lo utilizaremos más adelante. Presionar el botón *Commit* para concluir el proceso.
7 Repetir este procedimiento para las demás zonas.
8 En el caso de la franja más larga, aplicar *Relax* a una selección de los vértices extremos hasta asegurarse de que no hay caras superpuestas y luego, en modo *Face*, seleccionar todo el conjunto y presionar el botón *Straighten shape* (menú lateral *Reshape elements*). De este modo será más sencillo pintar una franja uniforme como haremos más adelante.
9 Comprobar la orientación seleccionando una cara y viendo a cuál correspone en el modelo. Reorientar y escalar todos los grupos y organizarlos para que queden dentro de la zona principal hasta obtener un resultado similar al de la figura 5.93 (*c*).
10 Grabar el resultado. Desde el *UV Editor* activar el comando de menú *Tools* / *Render UV Template*. El panel que se abrirá muestra varias opciones y una de ellas es útil para detectar solapamientos. Desplegar la lista junto a *Mode*, encabezada por *None* y activar provisionalmente *Solid render*. Comprobar si, en las dos zonas que

nos interesan, hay caras resaltadas en rojo (lo que indica solapamiento). Si fuera así, selecionarlas barriendo una ventana sobre ellas y corregirlas con la herramienta de menú *Tools* / *Relax*. Y, si fuera necesario, moviendo manualmente algunos vértices.

11 Volver a activar *Render* con la opción *None* y grabar el resultado con una resolución adecuada (1.024 x 1.024 en este ejemplo) y en un formato con un canal alfa, por ejemplo PNG (de 24 bits y marcando la opción "alpha"). Esto nos facilitará el trabajo posterior.

12 Abrir el PNG desde Photoshop o Gimp. Duplicar la capa, rellenarla de un color y situarla debajo de la original para que se visualicen bien las líneas. Pintar, dibujar o crear texto sobre las zonas que interesen. Luego guardar el mapa en formato JPG. También se podría guardar otro mapa, partiendo del anterior, convirtiéndolo a grises y aumentando el contraste, para utilizar como *Bump* si fuera el caso.

13 De vuelta en 3ds Max, crear un material, aplicar el mapa anterior a *Diffuse* y asignarlo al objeto.

14 *Render*. Así llegamos a un resultado final como el de la figura 5.93 (parte inferior).

Render to texture. Aplicaciones. Proceso general. Parámetros principales

Como es bien sabido por cualquiera que haya pasado por ello, una representación realista, bien afinada, con materiales y sistemas de iluminación avanzada, a una resolución no demasiado pequeña tarda de minutos a horas en generarse. Pero si probamos un juego de vídeo de calidad en el mismo ordenador en que se ha generado esta imagen, nos encontraremos con escenas de calidad no muy inferior y que se están procesando a una velocidad del orden de 30 fps (*frames* por segundo).

Es evidente que esto no sería posible si se utilizase el mismo método para el juego de vídeo que para el cálculo de iluminación avanzada. La respuesta a este misterio es que los creadores de estos juegos utilizan toda una serie de trucos y técnicas ingeniosas para reducir el tiempo de cálculo. Y la técnica principal es prescindir de los cálculos de iluminación y captar los resultados como textura, más exactamente, como *Light maps*, que se incorporan a los objetos de la escena, que aparecen iluminados con degradados o sombras difusas que parecen provenir de luces que no vemos pero que, en realidad, están integradas en el material que se les aplica.

Para conseguir este resultado hay diversos métodos, el más obvio de los cuales es hacer un *render* para cada una de las caras de un objeto y luego utilizar el resultado como mapa reproyectado sobre el objeto. Esto sería efectivo pero muy pesado. Afortunadamente este procedimiento se puede automatizar. Y los medios de automatizarlo están integrados en varios programas, entre ellos 3ds Max, mediante un recurso especial denominado *Render to texture*.

Aunque la aplicación más conocida y principal sean los juegos de vídeo, se puede utilizar la misma técnica para otras aplicaciones, entre ellas para enviar modelos interactivos a terceros de modo que puedan visualizarlos interactivamente con facilidad. Y es indudable que sus aplicaciones arquitectónicas son, por esta razón, muy importantes, aunque aún estén poco desarrolladas.

Otro uso importante de *Render to texture* es que nos permite integrar diferentes texturas en una única textura, lo que posibilita una gestión más eficiente del modelo y, en muchos casos, una aplicación más sencilla de diferentes texturas a diferentes caras de un modelo complejo.

Otra posibilidad es utilizar este recurso para grabar diferentes soluciones de iluminación. Si guardamos la información sobre la iluminación de la escena como un *Light map*, como un mapa de grises que solo mantiene información acerca de cómo se distribuye la iluminación en la escena, este mapa puede combinarse con facilidad con otro mapa, un *Diffuse map*, que solo guarde información sobre los colores y texturas de las superficies. Esto implica que

podemos tener un único *Diffuse map* que se combine con diferentes *Light maps* pregrabrados sin necesidad de repetir los cálculos.

Hay que tener en cuenta que el procedimiento tiene algunas limitaciones importantes. Dado que su finalidad es que podamos navegar por la escena con facilidad, cualquier efecto que dependa del punto de vista, como ocurre con los reflejos, no podrá ser integrado en la solución. Otro tanto vale para mapas que simulan efectos que dependen del punto de vista, como los mapas de tipo *Falloff*. Tampoco podremos cambiar la iluminación o mover objetos que afecten al modo en que se distribuye la luz. Y, en principio, tampoco podríamos mover objetos aunque esto admite soluciones (en las que no vamos a entrar), mientras los objetos dinámicos no afecten a la distribución global de la luz.

Resumiendo. Se utiliza *Render to texture* por las razones siguientes. Para ahorrar tiempo de cálculo. Para crear aplicaciones interactivas, modelos con los que un usuario corriente puede interactuar, desplazándose y orbitando en torno al modelo. También puede utilizarse para almacenar en un mismo modelo diferentes soluciones de iluminación. Y, en fin, para reducir la complejidad del modelo, algo que, como ya hemos visto en los apartados sobre texturas desplegadas (*Unwrap*), también puede hacerse sin necesidad de recurrir a *Render to texture* pero utilizando técnicas muy similares.

§ § §

El procedimiento se basa en los pasos siguientes:

1) Preparar la escena del modo habitual: asignar proyecciones UVW y materiales a los objetos, preparar una iluminación adecuada, etc.

2) Guardar el archivo con otro nombre para preservar la edición de los parámetros. En el nuevo archivo, integrar al máximo los objetos y los materiales. Esto quiere decir que habrá que agrupar los objetos, convirtiéndolos a mallas editables y anexándolos entre sí. El objetivo es contar, idealmente, con un único objeto al que se ha asignado indirectamente, a través del proceso de conversión y anexión, una proyección compleja y un material multi-subobjeto. En la práctica esto puede no ser deseable y habrá que dividir la escena en zonas y capas para simplificar el siguiente paso. El primer ejemplo que daré utiliza un único objeto integrado y un único material multi-subobjeto. El segundo ejemplo utiliza cuatro objetos y otros tantos materiales.

3) Seleccionar un objeto y aplicarle un modificador *Unwrap UVW*. Luego editar este modificador y aplanar el mapa. Cuando se utiliza *Render to texture* solo puede haber un mapa UVW para que todas las caras del objeto reciban una misma información del *texture element* correspondiente. Esto quiere decir que la proyección debe desplegarse de tal modo que no haya superposiciones de caras. Habrá que modificar también el canal de este modificador para que no interfiera con los mapas internos.

4) Abrir el módulo *Render to texture* (en 3ds Max, tecla 0 o menú *Rendering*). En este módulo, los parámetros y pasos principales, que también se detallan más adelante, son: *a*) Asignar un nombre y una vía de salida a los archivos que se generarán; *b*) Seleccionar el objeto u objetos que se quieren procesar (puede aplicarse el procedimiento a cada uno de los objetos o a varios a la vez); *c*) Especificar los *Texture elements* que se quieren generar para el objeto u objetos seleccionados y, opcionalmente, *d*) Especificar el tipo de material al que se aplicarán las texturas generadas, un *Shell material*, un contenedor que agrupa el material original del objeto y el nuevo "baked material". Pues tras este proceso se crea un nuevo material que deberia asignarse al objeto antes de exportarlo a otra aplicación para su uso interactivo. Esto implica que los materiales originales de los objetos se perderán. Para prevenir esto hay dos soluciones principales. La primera sería guardar los materiales o toda la escena con otro nombre para permitir la restauración posterior. La otra, proporcionada por 3ds Max, es utilizar un material que se crea automáticamente

tras el proceso, denominado *Shell material* y que agrupa a dos materiales, el original y el nuevo. Pero no hay que perder de vista que al enviar el resultado a otras aplicaciones habrá que colapsar este material a un tipo corriente que sea reconocido por la aplicación receptora. Una tercera opción es crear un nuevo material con los mapas desplegados que hemos creado.

Una vez completado este proceso se pueden eliminar todos los elementos sobrantes: materiales y mapas que se han utilizado para llegar al resultado final, luces y configuraciones de iluminación avanzada. A partir de ahí, es posible utilizar un sistema de iluminación simple que se procesará con mucha mayor rapidez. Y otra solución es hacer el material autoiluminante para que sea visible aunque no haya luces.

§ § §

La descripción completa de los parámetros del módulo *Render to texture* se puede encontrar en la ayuda de 3ds Max (en *Rendering / Render to texture*). En este apartado me limitaré a lo principal, referido al diagrama de la figura 5.94.

1. General settings. El grupo *Output* permite especificar la carpeta donde se grabará el archivo. Esto es lo que nos importa. El resto (*Skip existing files*, *Rendered frame window*, *Render settings*, *Setup*) son opciones que facilitan el trabajo. En general bastará con lo primero, especificar la carpeta de destino.

2. Objects to bake. La *tabla central* se irá rellenando con los objetos seleccionados en la escena. Los valores que aparecen en esta tabla no se editan directamente sino que se cambian automáticamente o desde los grupos de opciones que siguen. Hay cuatro columnas: *Name* (que muestra el nombre del objeto seleccionado), *Object channel* (muestra el canal del objeto y se modifica desde el grupo *Mapping coordinates*, más abajo), *Sub-Object channel* (muestra, si se seleccionan subobjetos, los canales de los subobjeto), *Edge padding* (muestra el valor asignado en el contador del grupo que sigue). El grupo *Selected objects settings* incluye una casilla para activar la configuración, un único parámetro, *Enabled padding* y un contador para especificar el valor de *padding* (predeterminado a 2). Véanse los comentarios que añado al final de este apartado sobre *padding*. El grupo *Projection mapping* se utiliza para generar *Normal maps* (véase la sección sobre relieves) y no lo utilizaremos en los ejemplos que siguen. El grupo *Mapping coordinates* controla las vías por las que se aplican las proyecciones UV. Hay dos opciones principales para los objetos selecccionados: "Use Existing channel" (si hemos creado manualmente un *Unwrap*) o "Use automatic unwrap" (si hemos utilizado esta opción). En el primer ejemplo que sigue utilizaremos las dos para comparar las diferencias.

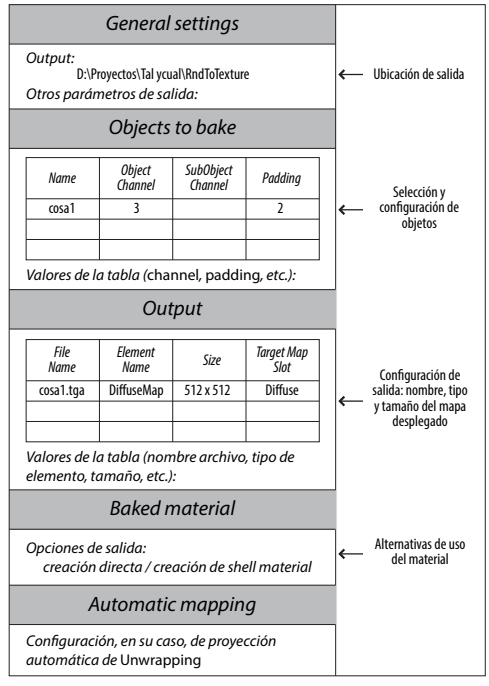


Figura 5.94 Render to texture. *Esquema de los parámetros principales en 3ds Max.*

3. Output. La *tabla* de esta sección muestra el nombre del mapa que se generará (columna *File name*), el nombre del elemento correspondiente (columna *Element name*) y el tamaño del mapa. Con *Add* se accede a un cuadro de diálogo con todos los tipos de mapas, o *texture elements*, disponibles. Véanse los comentarios que añado al final de este apartado. El grupo *Selected elements common settings* se utiliza para configurar los parámetros del elemento seleccionado. En *File name and type* se especifica el nombre y tipo de mapa de bits que se generará y que se guardará en la carpeta de destino especificada al comienzo. *Target map slot* despliega una lista con todos los mapas disponibles para ese elemento que se aplicarán al material del que se parte. Véanse también los comentarios que añado al final de este apartado. *Element background* permite cambiar el color del fondo. Puede merecer la pena escoger un color similar al del material principal para que los defectos, si los hay, se noten menos. El *Map size* puede especificarse de modo manual (escribiendo valores de anchura y altura) o seleccionando alguno de los predeterminados (de 128 x 128 a 2.048 x 2.048).

En general, en esta sección hay que hacer, al menos, lo siguiente: *a*) presionar el botón *Add* para escoger un *Texture element*, *b*) dar un nombre al archivo de salida, *c*) especificar un tipo de propiedad a la que se asignará el mapa en el nuevo material que se creará, por lo general *Diffuse color*, *d*) escoger una resolución de salida.

4. Baked material. En el grupo *Baked material settings*, las opciones *Output into source / Save source (Create shell)* regraban el material de que hemos partido o crean un nuevo *shell* material. Es preferible mantener la segunda opción (aunque también se podría copiar el material de que hemos partido a otro *slot* y dejar que se reasigne la nueva textura desplegada en lugar de la textura simple asignada previamente). Si se escoge esta segunda opción quedarán disponibles las alternativas *Duplicate source to baked / Create new baked*, que permiten escoger entre hacer un duplicado del material existente o crear uno nuevo. Si se escoge esta última opción, la lista presenta una lista de los *shaders* disponibles. La opción más corriente suele ser "Standard: Blinn", es decir, un material *Standard* con el *shader Blinn*. El botón *Update baked material* crea un nuevo *shell material* para todos los objetos seleccionados. El botón *Clear shell material* elimina los *shells materials* de los objetos seleccionados. En este caso hay dos opciones: *Keep source materials*, que conserva el material original, y *Keep baked materials*, que lo substituye por el nuevo material. *Render to files only* (desactivado), lleva el resultado directamente a los archivos, sin visualizarlos en el *frame buffer*. En general, es preferible activar la opción *Save source* (*Create shell*). Y, con esta opción, escoger la opción *Create new* o bien, si se quiere corregir una prueba previa, la opción *Duplicate source*.

5. Automatic mapping. Las opciones de este grupo son similares a las que se encuentran al aplanar los mapas desde el editor de *Unwrapping*. Si se escoge la opción de proyección automática en lugar de aplicar el modificador *Unwrapping* antes de entrar en *Render to texture* habrá que ajustar estos valores desde aquí.

6. Sección final. El botón *Render* hace un *render* de los objetos seleccionados. *Unwrap only* aplica un modificador *Automatic flatten UV* a los objetos seleccionados pero sin hacer un *render* y debe usarse solo si se ha escogido la opción *Automatic mapping*. Con *Close* se cierra el cuadro de diálogo grabando la configuración. Las opciones *Original / Baked* hacen que si se escoge "Views" el original o el *baked* se muestran en el visor y si se escoge *Render* se muestran en el *render*. En general, bastará con presionar el botón *Render* con las opciones predeterminadas para concluir el proceso.

Añado algunos comentarios sobre algunos de los parámetros citados.

Sobre *padding*. Como ya hemos visto en capítulos anteriores, los filtros de textura in-

corporados a la GPU incluyen por lo general un filtro de tipo *mipmap*. Esto implica que, al aumentar la distancia del objeto a la cámara se substituye automáticamente una textura de un tamaño determinado por otra de la mitad del tamaño (por ejemplo 1.024 x 1.024 pasa a ser 512 x 512, o bien 256 x 256, o bien 128 x 128, etc.). En cada uno de estos pasos los bordes de las imágenes se van alterando y pueden aparecer inconsistencias debido a estos cambios. El valor de *padding* hace que se produzca un recubrimiento entre aristas o bordes adyacentes para que estas inconsistencias se reduzcan. Para encontrar el valor adecuado la única solución correcta, pues hay muchos factores en juego, es probar y rectificar. Pero algunos profesionales recomiendan utilizar un valor que esté en torno al 10 % de la anchura o algo menos. Esto quiere decir que, si aparecen defectos internos en bordes, el valor predeterminado (2 píxeles) sería adecuado para un tamaño de 256 x 256 (que es también el valor predeterminado de tamaño) pero no tanto para valores superiores. Para 512 x 512 sería probablemente mejor aumentarlo hasta cerca de 5 píxeles y para 1.024 x 1.024 hasta 10 píxeles. Pero insisto en que esto puede variar considerablemente según los casos. En la gran mayoría de los casos el valor predeterminado, 2, es más que suficiente. Y, si se prueban valores superiores en el primer ejemplo, muy simple, que sigue, es fácil comprobar que no es en absoluto recomendable utilizar valores superiores a 2 si se va a utilizar *Automatic Unwrapping* pues los bordes de algunas superficies podrían superponerse a los de superficies vecinas y el resultado sería aún peor.

Sobre los *texture elements* disponibles. Cuando se presiona el botón *Add* para añadir un elemento de textura aparecerá una lista con diferentes posibilidades. Cada una de estas posibilidades implica que al hacer un *render* se representará (y se convertirá en textura) un aspecto de la escena: la iluminación, las sombras, el color y la textura de las superficies, etc. Y al seleccionar uno de estos elementos, en la parte inferior aparecen, además de las opciones corrientes (activar o desactivar, *target mapa slot* para especificar el tipo de salida, tamaño del mapa…), un grupo, *Selected elements unique settings,* con otros elementos que pueden sumarse, o no, al escogido. Las que más nos interesan y las que utilizaremos en los ejemplos que siguen son *Complete* o *Blend*. La información que incorporan estos dos elementos es la misma: la totalidad de la información de la escena, es decir, todo lo que aparece al hacer un *render*. La única diferencia es que el segundo es más flexible. Al utilizar *Complete* solo tenemos una opción adicional: activar o desactivar las sombras. Al utilizar *Blend* podemos desactivar las siguientes propiedades: *Lighting*, *Shadows*, *Diffuse*, *Ambient*, *Specular*, *Self-Ilum*, *Reflection* y *Refraction*. Si en el modelo no hay reflejos, que es la propiedad principal que interesa desactivar (pues depende del punto de vista) es más sencillo utilizar *Complete*. Y si queremos desactivar los reflejos o alguna otra de estas propiedades es mejor utilizar *Blend*.

La lista completa de *texture elements* disponibles es la siguiente: *Complete* (ya lo he descrito); *Blend* (ídem); *Specular* (guarda solo los valores especulares y mantiene la opción de activar la iluminación y sombras); *Diffuse* (guarda solo los valores de color superficial y mantiene la opción de activar la iluminación y sombras); *Shadows* (guarda solo los valores de sombras, sin otra opción); *Lighting* (guarda los valores de iluminación directa, indirecta y sombras que pueden desactivarse por separado); *Normals* (guarda un gradiente de color que indica las direcciones de las normales); *Alpha* (guarda solo el canal alfa, se utiliza a veces para guardar máscaras de las zonas transparentes del objeto); *Height* (guarda un mapa de grises que almacena las alturas relativas del objeto cuando la representación se basa en una proyección normal como las que se utilizan al crear *Normal maps*); *Ambient occlusion* (guarda un gradiente de grises que representa la oclusión del objeto). Hay que tener en cuenta que algunos de estos elementos, como el último, *Ambient occlusion*, solo funcionan con mental ray. Re-

sumiendo: en la gran mayoría de los casos, para aplicaciones corrientes, nos bastará con *Complete*.

Sobre el *Unwrap* automático o manual. Lo primero es más cómodo. Lo segundo es más eficiente. Veremos ejemplos de ambos usos en los ejemplos que siguen.

Sobre *Target map slot*. Esta lista desplegable presenta una serie de mapas corrientes en cualquier definición de material (*Diffuse*, *Specular*, *Bump*, etc.). El tipo que se escoja de esta lista (por lo general *Diffuse*) aplicará el mapa desplegado, como instancia, a la propiedad correspondiente del material de que hemos partido o al nuevo material que se creará, según cuál sea la opción escogida en la sección *Baked material*.

Ejemplo 1. Objeto simple sobre un plano

Preparar una escena adecuada para hacer una prueba simple. Para este ejemplo he escogido un objeto sencillo, en forma de "T" colocado sobre un plano. Tanto el plano como el objeto tienen aplicados dos mapas de textura. La escena está iluminada por una luz *mr Area spot* y una *skylight* que suaviza las sombras. Para el cálculo de iluminación he utilizado *Final gather* pero con el control de exposición desactivado. Este resultado inicial se muestra en la figura 5.95 (*a*).

1 Convertir uno de los dos objetos a malla editable y luego anexarle el otro. Al hacerlo, el programa preguntará si queremos combinar los ID de material ("*match material IDs to material*") y si queremos condensar los ID de material, es decir, eliminar los no usados ("*condense material and IDs*"). Confirmar. De este modo hemos creado un objeto único con un material multisubobjeto interno (se puede comprobar captándolo con el cuentagotas desde el editor de materiales).
2 Seleccionar el objeto y entrar en *Render to texture* (menú *Rendering* o tecla 0). Configurarlo así:

a) *General settings*. En *path*, especificar la carpeta donde se creará el archivo.
b) *Objects to bake*. El objeto selecionado aparecerá en la tabla. No hace falta hacer ningún cambio. Dejar el valor por defecto para *Object channel* (1) y *Edge padding* (2). Dejar desmarcada la opción de *Projection mapping* (solo se usa para crear *Normal maps*). Dejar la opción predeterminada "Use Automatic Unwrap" con el canal 3 (luego veremos otra alternativa no automática). Dejar la opción "All Selected" para que se procesen todos los objetos seleccionados (aunque en nuestro caso solo hay uno).
c) *Output*. Presionar *Add*. Escoger "Complete Map" y dejar los valores predeterminados. En *File name and type* dar un nombre y un tipo, por ejemplo, "escenaCompleteMap.png" que se guardará automáticamente en la carpeta predeterminada. También podríamos guardar el resultado del *render* pero es preferible guardar el archivo que graba todos los elementos. En este caso nos dará igual pero en otros no. Se puede dar un valor de salida a la lista de *Target map slot*, por ejemplo "Diffuse Color", pero no es necesario aunque nos lo recordará al iniciar el proceso de *render*. En *Size*, escoger como resolución de salida 1.024 x 1.024. Dejar el resto de los parámetros por defecto.
d) *Baked material*. No lo vamos a utilizar pero, si se quiere probar, marcar la opción *Save source* (*Create shell*) y *Create new baked* (*Standard*: *Blinn*) así como la opción *Keep source materials*.
3 En la parte inferior del cuadro de diálogo presionar el botón *Render*. Aparecerán los elementos de la textura desplegados en el visor, tal como se muestra en la imagen de la figura 5.95 (*c*).

Como puede apreciarse en esta imagen, el modo automático desperdicia bastante espacio y, si nos interesara agrupar de otro modo las caras, no podríamos hacerlo. Por estas razones

suele ser preferible no utilizar la opción de desplegado (*Unwrap*) automático sino la manual que, si se cuenta con un mínimo de experiencia, no lleva más de unos pocos minutos.

Para ello, repetir el proceso y hacer lo siguiente.

Antes de entrar en *Render to texture* seleccionar el objeto y (antes de seguir, eliminar el *Unwrap* automático si fuera necesario) aplicarle un modificador *Unwrap UVW*. Desplegar el subobjeto *Polygons* y seleccionarlas todas (Ctrl+A). Entrar en el editor UV (botón *Open UV Editor*). Activar el comando de menú *Mapping* / *Flatten mapping*. Dejar los valores predeterminados que, en general serán adecuados, y confirmar. Con estos valores, que pueden ajustarse según se necesite, las caras se desplegarán de modo similar a lo que hemos obtenido antes pero de un modo más compacto que, por añadidura, si nos interesa, podemos manipular para que aún se compacten más.

Entrar en *Render to texture* y repetir el proceso anterior pero, en la sección *Objects to bake*, en el grupo *Mapping coordinates*, en lugar de *Automatic*, seleccionar la opción *Use existing channel* con el valor predeterminado para el canal (1). Así se obtendrá un resultado como el de la última imagen de la figura 5.95 (*d*).

En cualquiera de los dos casos hemos obtenido una textura que incorpora la iluminación de la escena y que nos permitirá prescin-

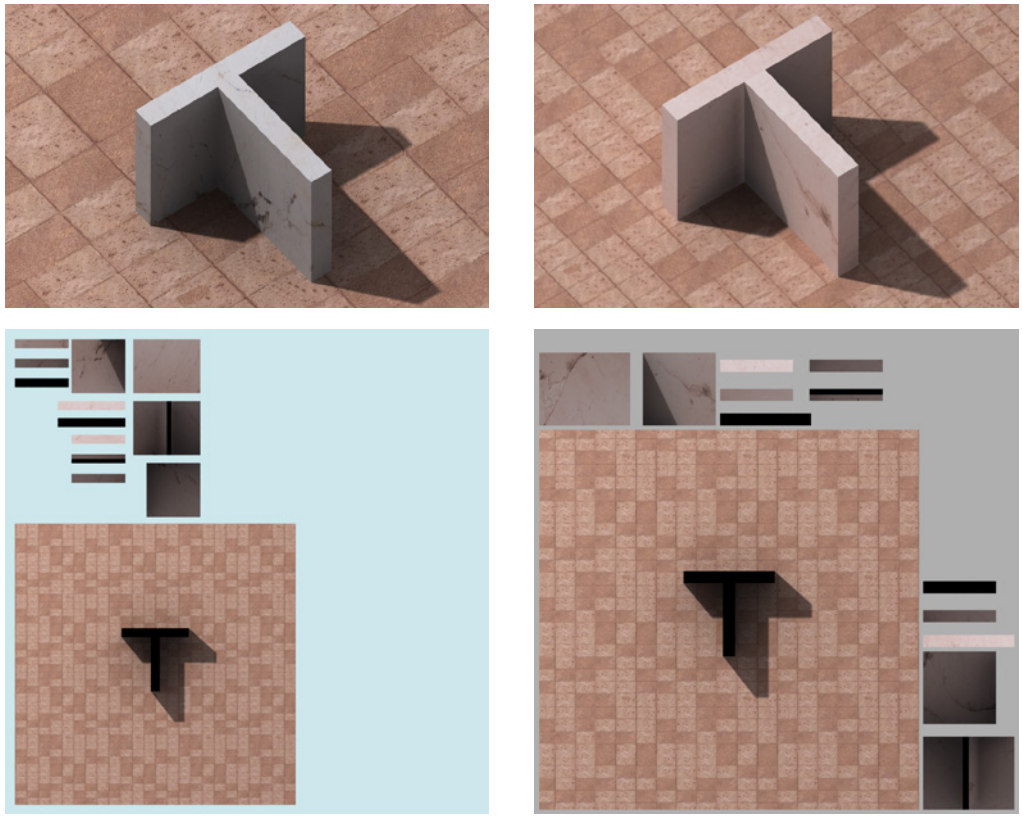


Figura 5.95 Render to texture. *Ejemplo 1:* a) *Escena inicial con luces;* b) *Escena final con las luces desactivadas;* c) *Elementos desplegados con el modo de* Unwrap *automático;* d) *Elementos desplegados con el modo de* Unwrap *manual.*

dir de las luces. Podríamos utilizar el material que se crea automáticamente, pero quizás se entienda mejor el sentido de todo esto si hacemos lo que sigue, que es algo más largo pero que resultará más claro.

1 Guardar la escena con otro nombre.
2 Desactivar las luces.
3 Crear un material *Standard*. Asignar un mapa a *Diffuse* y escoger como mapa de bits la textura desplegada que hemos creado. Hacer el material autoiluminante. Asignarlo al objeto "escena".
4 No modificar el *Unwrap UVW* pues para que este mapa encaje adecuadamente debe haber quedado tal como lo hemos dejado.
5 *Render*. El resultado será como el de la figura 5.95 (*b*) de este grupo que es muy similar a la figura de la que hemos partido aunque incluye algunas pequeñas diferencias que se podrían ajustar revisando la configuración anterior o editando el mapa en Photoshop o Gimp.

Como decía antes estas comprobaciones se pueden hacer sin necesidad de entrar en otro archivo pues, con *Render to texture* se crea automáticamente un *Shell material* que agrupa a dos materiales: el original del que hemos partido y el nuevo. Uno de ellos se muestra en el visor y otro en el *Render*. Esto depende de las opciones que hemos dejado por defecto. Si se presiona uno de los botones se accede a la definición del material que será la misma en los dos casos (la definición de que hemos partido). Para comprobarlo todo lo que hay que hacer es, desde el editor de materiales, captar el material que se habrá asignado al objeto.

Ejemplo 2. Galería

Preparar una escena como la que se muestra en la figura 5.96. La escena incluye los objetos: "suelo", "techo", "muros", que representan el recinto general de una galería y tienen asignados materiales de tipo *Arch&Design* con mapas que simulan un pavimento de madera en el caso del suelo y un muro de hormigón en el caso de los muros, proyectado en modo cilíndrico para el muro curvo y plano y en diagonal para el recto. Además, hay otros cuatro objetos, "cuadro1", cuadro2", "cuadro3", "cuadro4", que representan cuatro pinturas colgadas del muro y tienen asignados materiales *Arch&Design* con mapas de bits que representan otros tantos cuadros. Por último, hay otros cuatro objetos, "lamp1", lamp2", "lamp3" y "lamp4", que representan luminarias cilíndricas empotradas en el techo.

En este tipo de modelos que van a ser procesados con *Render to texture*, conviene eliminar las caras innecesarias de los objetos (caras posteriores de cuadros, suelo o techo si las hubiera) para que el resultado sea más limpio. Y, en nuestro caso, más fácil de seguir.

Cada una de las luminarias va unida a una luz fotométrica de distribución bastante focalizada, dirigida hacia el suelo y los cuadros. Hay también otras dos luces fotométricas, situadas fuera del recinto y que no arrojan sombras para que puedan afectar al interior, dirigidas en direcciones opuestas, hacia el suelo y los muros y hacia el techo y los muros opuestos, para proporcionar una iluminación general. El cálculo de la iluminación es con *Final gather*, el control de exposición está en EV 1,0 y la intensidad de las luces es muy baja, 5,0 cd, para los focos que iluminan los cuadros, 50 cd para la que ilumina el pavimento y el muro curvo y 10 cd para la que ilumina el techo y el muro recto.

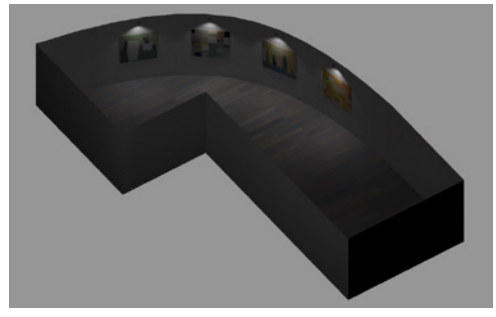


Figura 5.96 Render to texture. *Ejemplo 2. Galería. Vista general del modelo.*

La estrategia general será la siguiente. Aplicaremos *Render to texture* por separado a los muros, el pavimento, el techo y los cuadros. Para no alargar demasiado el ejemplo no he incluido reflejos en el pavimento. Si interesasen, porque se prevé utilizar el resultado en el propio 3ds Max o en un programa que procese reflejos, habría que hacer lo siguiente, pues los reflejos no se pueden incorporar a *Render to texture*, como ya he dicho. En lugar de utilizar un mapa de tipo *Complete*, utilizar un mapa de tipo *Blend* que da la opción de desactivar los reflejos (y las refracciones, si las hubiera). Luego, al completar el proceso, por el que se generará un material *Standard* con el mapa de la textura desplegada aplicada a *Diffuse*, ir a la sección de mapas de este material y, en *Reflection*, escoger un mapa de tipo *Ray trace*. Esto proporcionará reflejos, con menos controles que los del material *Arch&Design*, pero suficientes para aplicaciones más limitadas.

Por lo que respecta a los cuadros, también podrían utilizarse otras alternativas a las que describo a continuación. La primera sería, en lugar de procesarlos por separado, integrarlos en el muro con lo que tendríamos un objeto más complejo y un material multisubobjeto con más materiales.Y la segunda sería utilizar los recursos de proyección de *Render to texture* para proyectar los mapas en la textura del muro sin necesidad de integrarlos previamente. Doy solo algunas indicaciones sobre esto último, que habría que completar, si interesa hacerlo, desde la ayuda del programa. El lector no interesado en este procedimiento puede saltarse todo lo que sigue, hasta el siguiente párrafo. Lo primero que se necesitaría es aplicar al muro un modificador de tipo *Projection* que se utiliza principalmente para generar *Normal maps* pero también se puede utilizar para otras finalidades. Este modificador envuelve al objeto con una especie de jaula (*cage*) que representa rayos que empiezan en este envolvente y terminan en los vértices del objeto. Lo primero que hay que hacer es, desde la sección *Reference geometry*, seleccionar (con *Pick*) los objetos

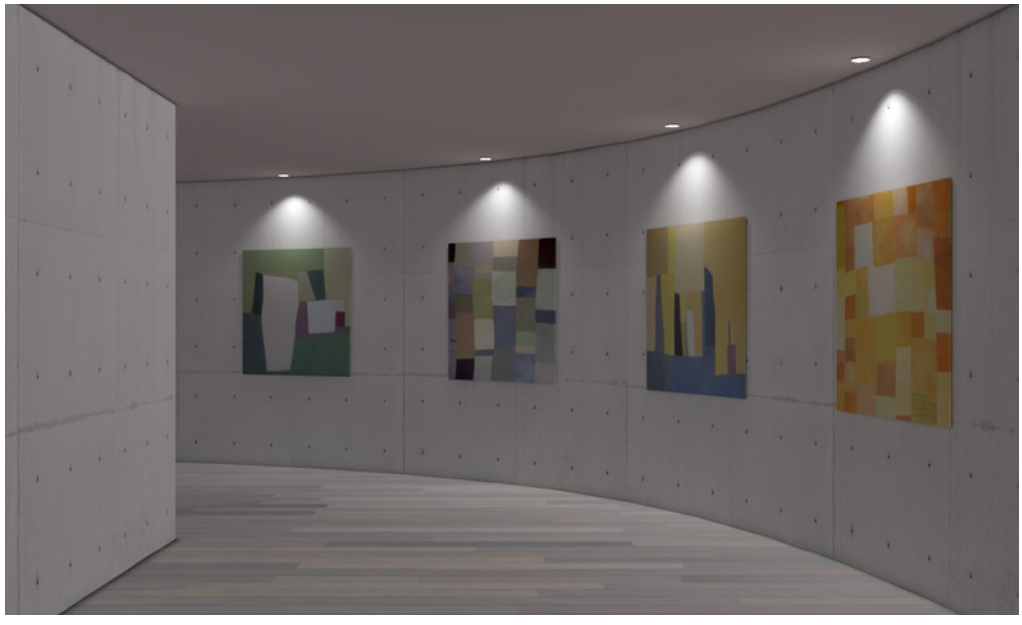


Figura 5.97 Render to texture. *Ejemplo 2. Galería. Vista interior.*

que se quiere proyectar. Se puede ajustar la envolvente desde la sección *Cage*, con los parámetros incluidos en el grupo *Push*. Una vez que se cuenta con este modificador, desde el panel *Render to texture* / *Objects to bake*, activar *Projection* y presionar el botón *Options*. En el siguiente panel, en el grupo *Resolve hit*, dejar la opción predeterminada "furthest" y marcar la opción "Include Working Model" para que se incluya la textura en la proyección. Y desactivar la opción "Ray miss check" que asigna un color a los rayos perdidos. De este modo, los cuadros se proyectarán sobre el muro. Y su grosor desaparecerá, lo que es otra razón para no utilizar este método en nuestro caso. Pues, como decía, en este ejemplo utilizaremos la alternativa de combinarlos en un único objeto "cuadros" con un material multisubobjeto que se creará automáticamente al colapsar uno de ellos y luego unir el resto.

Lo que haremos, por tanto, será lo siguiente, comenzado con los muros.

1 Seleccionar el objeto "muros" y, si se quiere, aislarlo (u ocultar el resto) para trabajar más cómodamente. Colapsar el muro si fuera necesario para incrustrar los mapas. Recordar que deben colapsarse los modificadores UV **antes** de añadir otros objetos pues, si no, el programa "entiende" que queremos aplicar los modificadores activos a los nuevos objetos lo que desharía su asignación previa. De este modo, tendremos un único objeto compuesto con un único material multisubobjeto.
2 Como vamos a aplicar un nuevo modificador UVW a un objeto que ya cuenta con dos modificadores UVW incrustrados, para que este nuevo modificador no interfiera con los anteriores habrá que asignarle otro canal que, en este caso, tendría que ser el 3. Pero no está de más comprobarlo. Así que, abrir *Channel info* (menú *Tools*). En la columna ID aparecen los nombres de los canales utilizados. El último utilizado será "2:map". Por tanto, podemos utilizar el siguiente, el canal 3, que está libre.
3 Aplicar al muro un mapa *Unwrap UVW*. En *Parameters/Channel*, cambiar el valor del canal por 3. Al hacer esto aparecerá un aviso ("Channel change Warning, This modifier contains edit to only one UVW channel at a time…") que ya he comentado anteriormente. Escoger "Abandon".
4 En modo subobjeto *Polygon*, seleccionar todas las caras (hacer Ctrl+A o seleccionarlas una por una con clic+Ctrl por el lado visible). Presionar el botón *Open UV editor* para entrar en edición de mapas. Aplanar el mapa activando el comando de menú *Mapping* / *Flatten mapping* con los valores predeterminados pero marcando la opción "By Material IDs" que distribuye mejor las caras. Reorganizarlo un poco de modo que las caras principales ocupen la práctica totalidad del mapa y la cara del fondo una esquina pues tiene muy poca importancia en la escena y partimos de la base de que en ningún momento se verá con detalle. Utilizar para ello la herramienta *Freeform mode* (cuarto icono de la barra de herramientas superior) para mover y deformar a la vez. El resultado, tras completar todos estos pasos, deberá ser similar al que se muestra en la figura 5.98 (*a*).
5 Con el objeto seleccionado, abrir *Render to texture* (tecla 0 o menú *Rendering*). Configurarlo de modo similar al ejemplo anterior: *a*) En *General settings*, especificar una carpeta de destino para los mapas que se van a crear; *b*) En *Objects to bake* dejar los valores predeterminados, comprobar que está activada la opción "Use Existing Channel y cambiar el canal a 3; *c*) En *Output*, presionar el botón *Add* y escoger como tipo de mapa, *Complete*. Cambiar el nombre y formato del archivo de salida según lo que interese (por ejemplo, cambiar TGA por PNG de 24 bits). Y de la lista junto a *Target map slot* escoger "Diffuse Color Map". Hacer también un clic en el icono de color junto a *Element*

background y, desde el selector de color, captar con el cuentagotas alguno de los colores del muro para substituir al negro de fondo pues esto hará que los posibles desajustes de aristas sean menos perceptibles. Por último, escoger una resolución de salida adecuada, en este ejemplo bastará con 1.024 x 1.024; *d*) En *Baked material*, aunque no vayamos a utilizarlo, dejar la opción predeterminada "Save Source (Create Shell)" pero cambiar la opción siguiente (Duplicate…) a *Create new baked* y, en la lista adjunta, escoger "Standard: Blinn". De este modo la textura generada se aplicará a un material *Standard*, en lugar de mental ray, que es más genérico y más facilmente reconocible por todo tipo de aplicaciones.

6 Salir del modo de aislamiento o desocultar los objetos ocultos antes de hacer el *render* para que se tenga en cuenta toda la escena. Presionar el botón *Render* en la

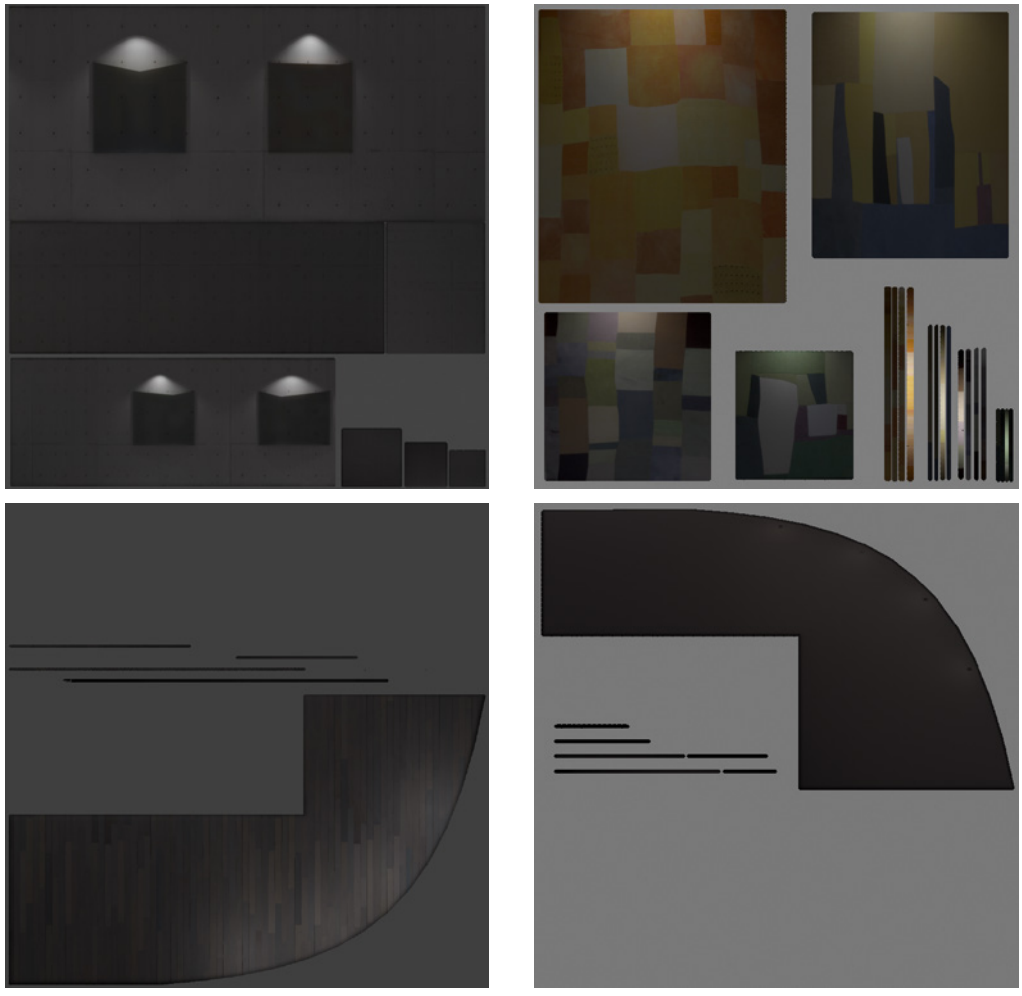


Figura 5.98 Render to texture. *Ejemplo 2. Galería. Mapas desplegados:* a*) Muros;* b*) Cuadros;* c*) Suelo;* d*) Techo.*

parte inferior del cuadro de diálogo *Render to texture*. Se creará y se grabará una textura desplegada correspondiente al muro.

Así se completa el proceso correspondiente al muro. Repetir el mismo procedimiento para los cuadros (unificarlos y colapsarlos en una única malla), el suelo y el techo. Los cuadros se han grabado con una resolución de 768 x 768 pues pueden reagruparse adecuadamente en este formato. El pavimento y el techo son sencillos de procesar.

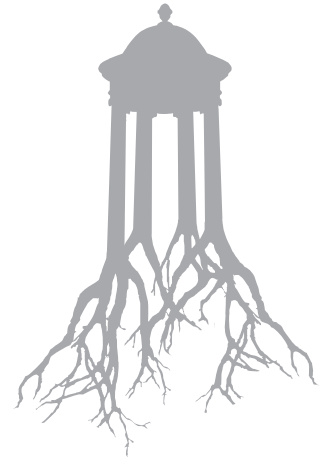
En las demás imágenes de la figura 5.95 correspondientes al resto de los elementos, he reordenado y girado los mapas para que se entienda mejor su relación con la escena. Esto no es necesario ni conveniente hacerlo pues se podrían alterar las relaciones establecidas entre el espacio UV de *Unwrap* y los espacios UV subyacentes.

De este modo, hemos terminado con una escena con cuatro objetos integrados y otros tantos materiales.

El último paso es utilizar este resultado para procesar el archivo sin necesidad de calcular la iluminación con lo que el resultado será mucho más rápido. Este nuevo archivo se puede utilizar para las diferentes aplicaciones que he comentado al comienzo (cálculos más rápidos, interacción con el usuario, creación de animaciones, etc.). En este caso, nos limitaremos a utilizarlo desde el mismo programa con que se ha creado. Continuar, por tanto, como sigue.

1 Guardar el archivo con otro nombre.
2 Eliminar las luces (dejar una apagada para que el programa no active automáticamente las luces predeterminadas). Desactivar el control de exposición.
3 Desde el editor de materiales, crear cuatro materiales, “muros”, “suelo”, “techo” y “cuadros”, de tipo *Standard* y asignar a cada uno de ellos el mapa desplegado generado en el proceso anterior (o bien, si se prefiere, captar el material creado por el programa, el *Shell material* compuesto por dos, el *Original* y el *Baked*, y arrastrar este último a un visor vació).
4 Editar estos cuatro materiales y aumentar la autoiluminación a 100.
4 En la configuración del cálculo de iluminación, desactivar *Final gather*.
5 *Render*.

El resultado será el mismo que hemos obtenido con el cálculo de iluminación previo, y que se muestra en la vista interior de la figura 5.97, al comienzo, pero mucho más rápido e independiente de cualquier sistema de cálculo de iluminación avanzada.



Apéndice. Materiales de construcción

Introducción

En las ocho secciones de este capítulo hay un resumen de las características de los materiales más utilizados en arquitectura. Están ordenados aproximadamente de mayor a menor antigüedad y al comienzo de cada sección se incluye un breve resumen de su historia como materiales de construcción.

A continuación sigue un resumen de sus características principales. Dado que este libro trata de las apariencias de los materiales, no de sus propiedades físicas reales, las características a las que se da mayor importancia son las visuales. Pero también he incluido un breve compendio de las características físicas pues no está de más tenerlas presentes. Entre otras cosas, porque el comportamiento físico se traduce, a lo largo del tiempo, en variaciones visibles. Esto ocurre de modo notorio en el caso de propiedades tales como el desgaste debido a la corrosión, al roce o a fuerzas diversas ejercidas en diferentes direcciones, que se traducen en grietas o deformaciones. Es menos notorio en el caso de otras propiedades como la densidad o la conductividad térmica o eléctrica. Pero también las he incluido para redondear la descripción y porque puede ser útil tener a mano un conjunto de referencias más amplias.

Aunque la mayoría de las unidades con que se describen estas propiedades son corrientes y las doy por conocidas, supongo que más de un lector me agradecerá que recuerde el sentido de otras no tan corrientes.

La ***dureza*** se mide de diferentes maneras. La primera escala se debe al geólogo alemán Friedrich Mohs (1773-1839) y, aunque en la actualidad hay otras escalas más precisas (Rosiwal, Knoop), sigue siendo un referente muy utilizado debido a su sencillez y su carácter intuitivo. Se basa en 10 grados definidos por un material de referencia y su resistencia al rayado dada en términos coloquiales: 1 talco (se puede rayar fácilmente con la uña), 2 yeso (se puede rayar con la uña con más dificultad), 3 calcita (se puede rayar con una moneda de cobre), 4 fluorita (se puede rayar con un cuchillo de acero), 5 apatito (se puede rayar difícilmente con un cuchillo), 6 ortosa (se puede rayar con una lija para el acero), 7 cuarzo (raya el vidrio), 8 topacio (se puede rayar por herramientas de carburo de wolframio), 9 corindón (se puede rayar por herramientas de carburo de silicio), 10 diamante (no se altera con nada excepto otro diamante). Otras referencias útiles son las siguientes: la uña tiene una dureza de unos 2,5, poco más que el yeso, una moneda de cobre 3,0 poco más que una uña, una hoja de cuchillo corriente 5,0, el vidrio de una ventana 5,5.

La **densidad** es la relación entre masa y volumen y la indico en todos los casos en kg/cm^3. Tampoco está de más recordar que es una magnitud escalar que representa la cantidad de materia de un cuerpo mientras que el peso es una magnitud vectorial que representa la fuerza con que un cuerpo es atraido hacia la tierra y, por consiguiente, puede variar según la altura a que esté situado el cuerpo en cuestión.

La **conductividad térmica** es la capacidad de una substancia de transferir la energía cinética de sus moléculas a otras moléculas adyacentes de otras substancias. Se representa por la letra K y se mide en W/m.K. Una conductividad térmica de 1.0 W/m.K indica que una cantidad de calor de 1 julio se propaga en 1 segundo, a través de una superficie de 1 m^2, por un grosor de 1 m, cuando la diferencia de temperatura entre las dos caras es de 1 K. Cuanto más alto sea este valor más conductor de calor será el material y a la inversa. El coeficiente de conductividad térmica, λ, mide la cantidad de calor por m^2 para que, en la unidad de tiempo, 1 m de material homogéneo varíe su temperatura 1 ºC entre las dos caras.

La **resistencia mecánica** la doy por lo general por referencia tan solo a la compresión y la tracción, para no complicar la exposición, e indicando en algún caso el módulo de elasticidad. Los valores los doy en kg/cm^2, que no es lo más usual desde el punto de vista técnico. Quien esté interesado en pasar estos valores a otras unidades que se encuentran en libros sobre resistencia de materiales, principalmente newtons y pascales, debe recordar las equivalencias siguientes.

1 kilopondio (kp) = 1 kilogramo-fuerza (kgf) = peso de una masa de 1 kilogramo (masa inercial en la gravedad terrestre) (=) 9,8 newtons (N). Es decir que 100 kg/cm^2 son aproximadamente iguales a 9,8 N/mm^2. El paréntesis que doy en algún caso indica que la igualdad es conceptualmente incorrecta. Otras fórmulas de conversión útiles son las siguientes: 1 N/mm^2 = 1 MPa (Megapascal) = 10.2 kp/cm^2 (=) 10 kg/cm^2. Es relativamente habitual pasar de datos en N/mm^2 a kg/cm^2, si solo se requiere una estimación, multiplicando por 10 (100 N/mm^2 serán 1.000 kg/cm^2).

1 Piedras

En arquitectura, la denominación *piedra natural* se refiere, de modo general, a todas las rocas que pueden ser utilizadas como elementos de construcción tras ser extraidas de yacimientos y ser sometidas a diversos tratamientos, que van desde operaciones elementales de corte y desbastado hasta el pulido por diferentes métodos.

Al igual que la madera y la tierra cocida, el uso de la piedra como elemento de construcción tiene más de 10.000 años de antigüedad y ha sido y sigue siendo, en menor medida debido a la progresiva implantación de otros materiales de menor coste, un elemento de construcción principal, aunque su uso como elemento estructural ha quedado relegado casi por completo.

Se acostumbra a clasificar las piedras en dos grandes grupos: piedra de cantería y rocas ornamentales.

El término *piedra de cantería* se aplica principalmente a su uso como elemento estructural (muros, columnas, arcos, vigas…), como elemento de revestimiento de suelos y paredes y como elemento de usos diversos menos refinados que los ornamentales (balaustres, bancos, elementos decorativos relativamente rústicos, etc.). En todos estos casos las piedras se extraen de la cantera en bloques manejables mediante cortes poco precisos y se someten a un tratamiento superficial relativamente rústico que no va mucho más allá de un desbastado rudimentario.

El término *roca ornamental* se aplica principalmente a su uso como elemento de revestimiento más refinado, lo que implica cortes más precisos y un acabado más refinado. Hay tres grupos principales de rocas ornamentales en el mercado: granitos, mármoles y pizarras.

Características físicas

Las rocas se clasifican corrientemente, a partir de su modo de formación, en tres grandes grupos. Las primeras rocas se formaron a partir de la cristalización del magma líquido que constituía la Tierra y se denominan rocas **magmáticas** o **ígneas** y están formadas por material volcánico en el que han penetrado gases minerales y líquidos diversos que crean nuevas formaciones cristalinas de colores variados. Las principales rocas magmáticas son las distintas variedades de *granito* (compuesto principalmente por feldepasto, cuarzo y mica).

Por alteraciones, erosiones y depósito de las rocas magmáticas más antiguas, se formaron las rocas **sedimentarias** que fueron arrastradas por movimientos tectónicos diversos hasta que los diferentes fragmentos se acumularon y consolidaron en fondos rocosos que se fueron compactando a lo largo de millones de años. Los varios modos de consolidación (temperatura, presión, reacciones químicas, aglutinantes presentes, etc.) están directamente relacionados con diferentes propiedades constructivas. Los tipos de rocas sedimentarias más comunes son la *caliza* (formada principalmente por cal), la *arenisca* (formada principalmente por cuarzo erosionado) o el *travertino* (formado principalmente por calcita y minerales y materia orgánica).

Por transformación de otros tipos de rocas, debida a cambios naturales, aparecieron las denominadas rocas ***metamórficas*** que presentan formaciones cristalinas y modificaciones características de la textura y el color. Los tipos más comunes son la *esteatita* (una roca muy blanda compuesta principalmente de talco, esteatita, magnetita y clorita), el *mármol* (que se divide en tres grupos según la mayor o menor proporción de magnesio: dolomita, magnesio y calcita) y la *pizarra*.

Las propiedades generales de todos estos tipos son similares, con las variaciones que se indican a continuación.

La ***densidad*** de las rocas es superior a la de otros materiales, con la excepción de los metales. Los valores más altos son los de granitos y mármo-

les (2.500 a 2.800 kg/m^3 en ambos casos) y pizarras (2.000 a 2.800 kg/m^3), y menores en el caso de areniscas (2.200 - 2.600) y calizas (2.000 a 2.200 en calizas duras y 1.600 a 1.800 en calizas blandas).

Su ***resistencia mecánica*** es en general muy alta, pero bastante variable pues depende mucho de su composición mineralógica y, en particular, de la presencia de cuarzo, que aumenta su resistencia. Las rocas con presencia de cuarzo pueden alcanzar valores de compresión superiores a los 5.000 kg/cm^2, mientras que la calcita no pasa de los 100 a 200 kg/cm^2. Algunos valores de referencia, de mayor a menor, son los siguientes: basalto (4.000 a 2.000 kg/cm^2), granito (de 1.000 a 2.700 kg/m^2), pizarra (2.000 a 2.500), arenisca (100 a 2.000), mármol (800 - 1.500), calizas (800 - 1.500). La resistencia a la tracción es del orden del 10 % al 15 % de la de compresión. El módulo de elasticidad está comprendido muy aproximadamente entre los 200.000 y los 950.000, valores inferiores a los de los metales pero superiores a los de las maderas. Algunos valores de referencia: calizas (210.000-690.000), granito (380.000-690.000), mármol (480.000 - 965.000), pizarra (620.000 - 960.000).

La ***conductividad térmica*** es relativamente alta por lo que su capacidad aislante es baja. En orden decreciente y con valores aproximados medidos en watios por metro lineal por grados kelvin (W/m.K): basalto (3.5), mármol (3.50), arenisca (3.00), granito (2.80), pizarra (2.20), caliza dura (2.30 a 1.70), caliza blanda (1.10 a 0.85).

La ***dureza,*** referida a la clásica escala de Mohs, es bastante variable. Las rocas más duras son las ígneas, como el basalto (en torno a 6,0) o el granito (entre 5 y 7). Las sedimentarias y las metamórficas son más blandas, con valores en torno a 3 para las calizas, areniscas, mármoles y pizarras.

Piedras más utilizadas en la construcción

Arenisca. Es una roca sedimentaria formada principalmente por cuarzo. Se trabaja con facilidad. Si el contenido de cuarzo es elevado es muy resistente mientras que las variedades que contienen calcita se disgregan con facilidad. La porosidad varía del 0,5 al 35,0 % y la absorción, del 0,2 al 9,0 %. Los colores dominantes son ocres. Se utiliza principalmente como roca de cantería con distintos tipos de acabado.

Caliza. Es una roca sedimentaria formada principalmente por cal. Su superficie es granular, bastante lisa pero que no se presta a ser pulida excepto en el caso de algunas calizas de mayor dureza. La porosidad varía entre el 0,6 y el 31,0 % y algunas variedades presentan poros muy amplios. El coeficiente de absorción varía del 0,2 al 12,0 %. Se presenta en una amplia gama de colores, del blanco al crema, del amarillo al rojo, del verde al azul, del gris al negro, con motivos igualmente variables. Como la arenisca, se utiliza principalmente como roca de cantería con tipos de acabado variables.

Travertino. Es una roca sedimentaria que se forma en manantiales geotérmicos y se caracteriza por múltiples cavidades formadas por el flujo del agua. Su color depende principalmente de la mayor o menor presencia de materia orgánica. Es una roca blanda, con una porosidad del 5,0 al 12,0 %, absorción del 2,0 al 5,0 %. Los colores dominantes suelen ser ocres y cremas y puede perder color con el tiempo si está expuesto a la luz directa del sol.

Granito. Es una roca solidificada perteneciente al grupo de rocas ígneas, principalmente plutónicas. Se compone de feldespasto (que afecta principalmente a su color), cuarzo (que afecta principalmente a su dureza) y mica. Se considera la piedra más resistente tanto a los agentes naturales como a la polución aérea. Sus valores de porosidad están entre el 0,4 y el 1,5 % y la absorción entre el 0,2 y el 0,5 %. Se presenta en una gran variedad de colores y texturas.

Mármol. Es una roca metamórfica que se forma a partir de la transformación de rocas sedimentarias calcáreas. El mármol puro es blanco y cristalino y no contiene fósiles. Hay una enorme variedad de mármoles que se caracterizan por la gran variedad de dibujos y colores de la vetas. La porosidad está entre el 0,5 y el 2,0 % y la absorción entre el 0,2 y el 0,6 %.

Pizarra. Las pizarras son rocas cuya característica principal es que se pueden extraer en láminas delgadas a lo largo de un plano de esfoliación que ha sido causado por un metamorfismo de bajo grado debido a la compresión tectónica. Esta característica se mantiene en planos de foliación muy mar-

cados que, entre otras cosas, facilitan la obtención de láminas muy delgadas. Se utiliza principalmente en cubiertas pero también en revestimientos de suelos y paredes. Se caracterizan visualmente por ser rocas de un grano muy fino, de color gris oscuro bastante homogéneo y por una textura que revela su estructura en planos horizontales paralelos a la superficie del revestimiento. Los valores de porosidad están entre el 0,4 y el 5,0 % y los de absorción, entre el 0,1 y el 1,7 %.

Otras piedras. Entre las metamórficas se utilizan a veces el gneiss, la cuarcita o el esquisto. El **gneis** es muy similar al granito (tiene sus mismos componentes) pero con capas alternadas muy marcadas. Se utiliza a veces para hacer peldaños o adoquines. La **cuarcita** es una roca dura con alto contenido en cuarzo. Se utiliza a veces como roca ornamental y también en caminos y vías férreas. Es una roca muy dura y que se trabaja con dificultad. El **esquisto** es similar a la pizarra pero tiene unos valores de absorción de agua mucho más altos (en torno al 10 %, similares a los de la arenisca o la caliza) que los de la pizarra y colores más variados.

Entre las rocas ígneas, principalmente plutónicas, otra rocas importantes son el basalto, la diorita o la sienita. El ***basalto*** es una roca eruptiva de color oscuro o negro. Es muy resistente pero su coste es muy superior al de otras piedras por lo que su uso es menos frecuente. Por otra parte, resiste menos los agentes atmosféricos que el granito, una de las razones por las que se emplea menos aunque su uso en pavimentos, por su gran dureza, se remonta a la época de los romanos que lo utilizaban (*lapis silex*) para pavimentar sus vías principales. En la actualidad, el basalto y la diorita se utilizan a veces en túneles y partes inferiores de puentes en lugar de granito. La **diorita** es una piedra suave, compuesta por múltiples minerales que a veces se utiliza como agregado y no muy frecuentemente como substituto del granito o la sienita. Se pule muy bien y es muy dura por lo que a veces se emplea en aplacados y en pavimentos interiores. La **sienita** es una roca similar al cuarzo pero menos dura, de color gris, verde o rojizo, que se pule con facilidad por lo que suele emplearse como material decorativo.

Piedras artificiales. La denominación es ambigua y no hay una terminología clara. Pero las incluyo en este apartado pues su uso es cada vez más importante. Los componentes principales son naturales (principalmente fragmentos de mármoles o de rocas silíceas) pero el proceso de fabricación se basa en la adición de colorantes, aditivos y otros componentes, embebidos en una resina de poliéster. El proceso, muy resumido, consiste en *a*) preparación de las materias primas (triturado de rocas, selección de colores, lavado, secado, cribado según diferentes granulometrías); *b*) mezcla de los fragmentos con la resina poliéster; *c*) formación de moldes por presión, vibrado y vacío, con lo que se obtienen grandes bloques (de un volumen de más de 3 m^3 y alrededor de ocho toneladas de peso); *d*) curado a temperatura ambiental durante varios días; *e*) corte y acabado de los bloques siguiendo procedimientos similares a los de la piedra natural.

Terrazos. Esta denominación abarca una variedad de productos que se utilizan exclusivamente en pavimentos desde hace más de cinco siglos. Probablemente se comenzó a utilizar en Venecia, en el siglo XV, para sacar partido de los fragmentos que quedaban tras construir pavimentos de mármol. El término deriva de *terrazza*, que a su vez deriva del latín *terraceus*, pues al principio se utilizaron para pavimentar las terrazas de viviendas corrientes. En la actualidad se siguen utilizando fragmentos de mármol pero a menudo combinados con otros tipos de roca y utilizando cemento como conglomerante, a veces coloreado con pigmentos o, más recientemente, resinas de diversos tipos.

Características visuales. Métodos de simulación

En el caso de las rocas ornamentales, las diferencias fundamentales vienen de los diversos minerales que las componen y que proporcionan las variedades de colores y texturas características de los granitos y los mármoles. Dado que estas rocas se presentan en general con la superficie pulida, los métodos de simulación se basan principalmente en obtener fotografías adecuadas. Esto vale sobre todo para los mármoles, cuyas caractersísticas visuales son muy difíciles de reproducir por métodos procedurales. En el caso de rocas más informes, pero con minerales de diferentes colores claramente visibles, como los granitos, es posible, con considerable esfuerzo y habilidad, obtener texturas

Figura 6.1 Piedras. Texturas propias: a*) mármoles,* b*) granitos,* c*) arenisca, caliza, pizarra.*

similares. En el capítulo 5 he desarrollado algún ejemplo que puede servir de orientación. Las rocas de menor variedad de textura, como las pizarras o las areniscas, pueden simularse algo mejor con métodos procedurales combinados con mapas de relieve simple.

En el caso de la roca de cantería, los tipos de *acabado* caracterizan el aspecto de la piedra. Los métodos principales, de menor a mayor refinamiento, son: *lajado* (se aplica solo a rocas exfoliables como pizarras y algunas cuarcitas y areniscas, por medio de instrumentos como cuñas o cinceles que permiten separar placas por planos naturales); *partido* (separación por cuñas y mazos, aprovechando planos más débiles, lo que da un acabado con protuberancias marcadas); *cortado* o *serrado* o *telar* (mediante el uso de discos o sierras especiales, telares de flejes de acero o diamante o sierras especiales, lo que da un acabado liso, algo rayado y mate, aunque corrientemente es un paso previo a otros acabados); *apiconado* o *picado* (se aplica

Figura 6.2 Piedra. Patrones de acabado: a*) partido, apiconado, apiconado acanalado;* b*) abujardado, abujardado fino, abujardado y pulido;* c*) escafilado, flameado, tratado con chorro de arena.*

a rocas no muy duras, con incisiones o muescas alargadas y paralelas, practicadas sobre superficies cortadas con un puntero o pica); *escafilado* (se aplica a granitos y algunas calizas, por medio de incisiones o muescas aleatorias sobre superficies cortadas que dan un acabado rugoso natural); *abujardado* (incisiones redondeadas y regulares, como pequeños cráteres, producidos por la bujarda, un martillo metálico con pequeños dientes piramidales); *raspado* (se aplica principalmente a areniscas blandas, alisando la superficie mediante rascadores o lijas que dan una superficie lisa y rugosa); *flameado* (se aplica a rocas de estructura granítica, por medio de la llama de un soplete que fragmenta parcialmente los diferentes minerales superficiales, a lo que le sigue un enfriamiento rápido con agua, lo que da un acabado característico rugoso y muy estable); *envejecido* (mediante el uso de abrasivos especiales, mediante el uso de tratamientos químicos o mediante chorro de arena o metal pulverizado); *apomazado* (similar al pulido aunque con abrasivos de grano menos fino, se aplica a rocas que no

Figura 6.3 Piedras. Ejemplos de tipos de distribución.

admiten bien el pulido, como algunas calizas); *pulido* (se aplica solo a rocas cristlinas como mármoles y algunos granitos y calizas, por medio de muelas abrasivas de grano progresivamente más fino).

La simulación visual de los tipos de piedra principales resumidos anteriormente, debe tener en cuenta tres factores principales que se ilustran en las figuras adjuntas:

a) La textura propia de las diferentes rocas. Esto afecta principalmente al mármol y los granitos, que tienen características muy destacadas, fácilmente distinguibles y que las singularizan de un modo único. Esto hace también que sean difíciles de reproducir por métodos procedurales pero, como hemos visto en el capítulo anterior, es posible obtener resultados aproximados que funcionan suficientemente bien a distancias relativamente cortas. Con todo, la mejor opción será obtener fotografías de grandes paramentos y rectificarlas y limpiarlas con métodos que también hemos visto.

b) El acabado de fábrica. En el caso de acabados pulidos (brillantes) o apomazados (mates) la textura propia se muestra nítidamente y no hay que hacer otra cosa que asegurarse de que la imagen utilizada es fiel a esta textura. Esto afecta principalmente a las piedras citadas en el párrafo anterior, que son precisamente las que se trabajan de este modo para mostrar la calidad intrínseca de la textura. En el caso de otros acabados relativamente corrientes, como los que se ilustran en la figura 6.2, y que se aplican a piedras más informes y, por tanto, más fáciles de simular por métodos procedurales, es posible obtener resultados adecuados combinando un color base con un patrón adecuado. Esto puede hacerse tanto en un programa de pintura digital como en un programa de simulación con la ayuda de mapas de relieve. La ventaja del primer método es que los resultados son más controlables y de mayor calidad (bastará, en general, con superponer la capa de patrón, en modo multiplicar, con la capa de color y aplicar a la primera un filtro de desenfoque gaussiano suave). Su desventaja es que las sombras de la textura vendrán dadas por la orientación del patrón y pueden no coincidir con la textura que produciría una determinada luz virtual. La ventaja del segundo método es que se adaptará a la dirección de la luz. Su desventaja es que el efecto, con un relieve de tipo *Bump* será más plano y menos controlable.

c) La distribución de la piezas y las características de las juntas. También en este caso tenemos dos posibilidades. La primera, que dará mejores resultados pero puede ser difícil de obtener, es conseguir fotografías de grandes paramentos y luego rectificarlas y limpiarlas para que puedan ser utilizadas en un proyecto de simulación. También hemos visto los procedimientos y ajustes por los que hay que pasar. La segunda es obtener una distribución aproximada, que puede incluso crearse artificialmente, y luego superponerle un patrón de juntas. Esto implica contar con un patrón de juntas sobre fondo blanco (al aplicarse en modo multiplicar el blanco, pues se multiplica por 1, mostrará el color de la capa inferior), que no es muy difícil de "fabricar" y del que se pueden derivar otros patrones. Al superponer este patrón sobre la textura base el resultado será notoriamente convincente.

Para completar estas indicaciones con una magnífica selección de obras de arquitectura clásica y moderna con acabados en piedra, véase el libro de Alfonso Acocella (Acocella, 2004).

2 Maderas

Actualmente, los bosques representan en torno al 30 % de la superficie de los continentes (excluyendo la Antártida) según datos de la FAO (Food and Agriculture Organization), de 1997. Esto supone algo más de 3.500 millones de hectáreas. Se estima que hace 8.000 años eran unos 6.000 millones de héctareas, es decir que (principalmente desde la Revolución Industrial) se han perdido casi la mitad. Y cada año siguen desapareciendo en torno a 15 millones de hectáreas. Esto afecta principalmente a los bosques tropicales.

Antes de la Revolución Industrial el material principal utilizado en la construcción era, con gran diferencia, la madera. Tras la explotación intensiva de los bosques para cubrir las nuevas demandas energéticas, la superficie se ha ido reduciendo espectacularmente y la construcción con madera se ha reducido.

Pero, por otro lado, esta situación está evolucionando positivamente por diversas razones, entre las que figura la reducción del uso de combustibles fósiles, la mayor utilización de elementos biode-

gradables y el menor gasto energético que genera su producción. A lo que cabe añadir sus diversas ventajas como elemento de construcción y su gran atractivo como material, lo que aumenta su demanda. Por tanto, la madera sigue siendo un material de construcción importante aunque su utilización se ha desplazado y está más presente como revestimiento interior que como material de construcción propiamente dicho.

Características físicas

Las maderas se clasifican de muy diversos modos, pero una división inicial adecuada es la que distingue entre **maderas blandas**, principalmente coníferas (denominadas así porque llevan las semillas en envolventes en forma de cono y que son el grupo principal de las gimnospermas) tales como el abeto, pino, ciprés o cedro, y **maderas duras** o **frondosas**, principalmente dicotiledóneas (denominadas así por tener dos cotiledones durante las primeras etapas de su desarrollo y que son el grupo principal de las angiospermas) tales como el roble, chopo, haya o cerezo. Los términos *blando* y *duro* son relativos y no siempre indican una propiedad claramente detectable. Las maderas duras son más heterogéneas y más densas y resistentes y dan mejor resultado en construcción.

La madera es anisotrópica, sus propiedades varían notablemente según la dirección en que se midan. Es también muy heterogénea, tanto por lo que respecta a la gran variedad de una especie determinada como a la gran variedad entre especies. Esta heterogeneidad se manifiesta de un modo más notorio en los anillos anuales.

La **densidad** (para un 12 % de humedad) varía entre los 350 y los 650 kg/m^3 aproximadamente. Las maderas blandas están en torno a los 400 o 500 y las maderas duras en torno a los 600 o 700. La lista siguiente muestra densidades, de mayor a menor, de algunas especies características: abeto (380), cedro (400), castaño (430), cerezo (500), nogal (550), roble (630), haya (640). En la manufactura de muebles se utilizan preferentemente especies de baja densidad, por debajo de los 400. En construcción se utilizan preferentemente especies de densidad media (en torno a los 500). En ambientes de humedad constante es preferible utilizar maderas de alta densidad (por encima de los 750).

La **conductividad térmica** es baja por lo que su capacidad aislante es alta. Es algo mayor en las frondosas pesadas, como los robles, con valores en torno a los 0,25 W/m.K, y menor en las coníferas, con valores en torno a los 0,15 que pueden aumentar hasta los 0,23 en las coníferas más pesadas, como el fresno y bajar hasta 0,12 en coníferas ligeras, como el pino blanco.

La ***dureza*** de una madera es otro factor importante que afecta principalmente a la resistencia de un suelo al desgaste con el tiempo. Se mide frecuentemente por medio de la escala de dureza de Janka (*Janka Hardness Scale*). La lista siguiente muestra, de menor a mayor, la dureza de algunas especies características: pino blanco (380), castaño (520), cerezo americano (950), nogal (1.010), abedul (1.050), roble (1.200), fresno (1.320), roble blanco americano (1.350), arce (1.450), wenge (1.630), caoba (2.200), cerezo (2.600), ébano (3.600).

Por lo que respecta a la ***resistencia,*** si la madera se somete a cargas de baja intensidad se deforma, comportándose como un cuerpo plástico antes de llegar al límite de rotura. Las deformaciones se miden en función del módulo de elasticidad, que depende del tipo de madera y de la cantidad de humedad y que oscila entre valores del orden de los 75.000 kg/cm^2 (77.000 cedro, 85.000 castaño, 99.000 pino) hasta los 120.000 (103.000 cerezo, 113.000 alerce, 116.000 nogal, 119.000 haya, 123.000 roble). La flexibilidad indica la capacidad de una madera para curvarse sin romperse, una propiedad característica de la madera que se utiliza en la fabricación de ciertos tipos de muebles pues las maderas jóvenes y húmedas son más flexibles y la flexibilidad puede aumentarse mediante tratamientos específicos con vapor. Los valores de resistencia a compresión y tracción son bastante variables. En general, están comprendidos entre unos 300 y 900 kg/cm^2 para compresión y entre unos 400 y 1.100 kg/cm^2 para tracción. En muchas maderas el valor de resistencia a la tracción es superior al de compresión. Así ocurre en las maderas siguientes (doy en primer lugar el valor de compresión y, a continuación, el de tracción): teca (900 / 1.100), boj (880 / 1.300), caoba (680 / 1.200), abedul (650 / 400), fresno (630 / 725), haya (530 / 1.080), ro-

ble (450 / 740), abeto rojo (435 / 760), cedro (380 / 780), pino (310-520 / 600-900). Esto hace que los valores de resistencia a la flexión estática sean superiores a los de otros materiales. De modo muy general (todos los valores que estoy dando son muy aproximados) puede decirse que los valores de compresión están comprendidos entre los 600 y 1.000 para las maderas duras (ébano, encina, tejo, teca), en torno a 500 y 600 para las semiduras (roble, nogal, arce, fresno, álamo, acacia, cerezo, almendro, castaño, haya) y entre 300 y 500 para las blandas (abeto, alerce, sauce, tilo, álamo blanco).

Maderas más utilizadas en la construcción

Abedul (*Betula alba*, betuláceas). Densidad: 600-750 kg/m^3. Zonas de producción: norte de Europa, Norteamérica, Asia. El tronco, a veces tortuoso, tiene de 4 a 10 m de altura y de 0,30 a 0,60 m de diámetro. La madera de abedul se caracteriza por los anillos apretados y el color pálido. Se utiliza principalmente en la industria de los enchapados (proporciona los enchapados más fuertes y sólidos) y de la fabricación de muebles. No es apta para usos exteriores.

Abeto blanco (*Abies amabilis*, pináceas). Densidad: 450-600 kg/m^3. Zonas de producción: Francia, España, Italia, Serbia. El tronco tiene de 30 a 45 m de altura y un diámetro máximo que puede llegar a los 1,2 m. La madera es blanda y no muy resistente y se emplea comúnmente en la elaboración de papel, cajas de embalaje y otros elementos funcionales y económicos para la construcción.

Abeto rojo (*Picea abies*, pináceas). Es otra variedad de abeto que se emplea principalmente para fabricar armazones, muebles y madera laminada.

Álamo temblón (*Populus iremula*, salicáceas). Densidad: 500 kg/m^3. Zonas de producción: Alaska, Canadá, zonas occidental y nororiental de Estados Unidos. El tronco puede alcanzar los 48 m de altura y unos 1,6 m de diámetro. La corteza es variable por lo que se refiere al color y al grado de surcos. El tronco tiene una textura uniforme con una fibra recta, ligera y blanda. Se trabaja fácilmente. Su pasta se emplea en la fabricación del papel y la madera en recubrimientos, plataformas, muebles, enchapados y paneles.

Arce americano o ***arce de azúcar*** (*Acer saccharum*, sapindáceas). Densidad: 700-800 kg/m^3. Zonas de producción: Norteamérica. El tronco tiene de 4 a 10 m de altura y de 0,30 a 0,60 m de diámetro. Tiene una veta ondulada muy característica y más notoria en ciertas variedades como el denominado “arce de ojo de pájaro”. Es una de las maderas más duras y densas. Suele emplearse en la fabricación de suelos industriales especiales, como pistas de bolos. También se usa para fabricar instrumentos musicales como violines, guitarras o tambores.

Caoba africana (*Khaya ivorensis*, meliáceas). Densidad: 450-600 kg/m^3. Zonas de producción: Ghana, Costa de Marfil, Camerún, Guinea, Gabán, Angola, Liberia, Nigeria. Su tronco tiene entre 15 y 25 m de altura y de 0,60 a 1,20 m de diámetro. La caoba africana es una madera noble caracterizada por uniones verticales de brillo variable y marcas de vasos xilemáticos (células tubulares). Muy duradera y sólida, la caoba africana se emplea para la fabricación de muebles, paneles o enchapados.

Castaño (*Castanea sativa*, fagáceas). Densidad: 600-700 kg/m^3. Zonas de producción: Francia, Reino Unido, Alemania, Italia. El tronco tiene de 3 a 8 m de altura y de 0,30 a 1,00 m de diámetro. Es una madera dura, resistente, de color claro. Se emplea en la manufactura de muebles, barriles y vigas de tejados, sobre todo en el sur de Europa. Debido a la tendencia de los ejemplares más viejos a romperse, doblarse mal y debilitarse, no es apta para la realización de grandes piezas. Se utiliza principalmente en la fabricación de ventanas y puertas y algunos tipos de suelos.

Cedro americano (*Cedrefa odorata*, meliáceas). Densidad: 400-700 kg/m^3. Zonas de producción: México, Surinam, Guyana, Nicaragua, Honduras, Brasil, Antillas. El tronco tiene de 15 a 20 m de altura y de 0,50 a 1,10 m de diámetro y una gruesa corteza marrón grisácea, con una veta longitudinal irregular. Conocido como cedro de España en el comercio inglés, tiene una gran demanda en los trópicos por su resistencia natural a las termitas y al deterioro. Es una madera ligera y atractiva que se emplea a menudo para la fabricación de artículos domésticos.

Cerezo (*Prunus avium*, rosáceas). Densidad: 600-700 kg/m^3. Zonas de producción: Francia, Rei-

no Unido, Alemania. El tronco tiene de 3 a 12 m de altura y de 0,30 a 0,60 m de diámetro. El duramen es marrón rojizo medio, con vetas rectas y una textura fina. Es una de las maderas con más demanda después de las maderas nobles para la fabricación de muebles refinados, revestimientos de suelos. instrumentos musicales, esculturas y piezas torneadas, obras de carpintería, enchapados y contrachapados. Es fácil de trabajar y apta para cualquier tipo de acabado.

Ébano de macassar (*Diospyros celebica*, ebenáceas). Densidad: 1.100-1.200 kg/m^3. Zonas de producción: Indonesia. El tronco puede alcanzar una altura de hasta 10 m y de 0,30 a 0,60 m de diámetro. Es una especie muy bella y muy apreciada pero en la actualidad muy escasa y, por consiguiente, bastante cara. El ébano de macassar es una de las especies más preciadas del mundo en la actualidad, quizás porque su área de crecimiento es muy limitada.

Embero o nogal africano (*Lovoa trichilioides*, meliáceas). Densidad: 500-550 kg/m^3. Zonas de producción: Costa de Marfil, Congo, Ghana, Nigeria, Camerún, Guinea. El tronco puede alcanzar una altura de entre 15 y 20 m y un diámetro de 0,50 a 0,55 m. Tiene un hermoso color bronceado con líneas negras irregulares, es dura, fuerte, con anillos apretados y muy resistente al agua. Se emplea en la construcción de postes de puentes y traviesas de ferrocarril, y para la producción de carbón. También se utiliza para fabricar muebles, paneles, enchapados.

Eucalipto (*Eucalyptus maculata*, mirtáceas). Densidad: 750-850 kg/m^3. Zonas de producción: Australia, Brasil, Sudáfrica. El tronco tiene de 10 a 15 m de altura y de 0,50 a 0,80 m de diámetro. Presenta una veta recta y una textura gruesa, relativamente dura. Tiene la notable cualidad de renovarse con gran rapidez. Se utiliza en los mismos campos de aplicación que las maderas nobles, en muebles de todo tipo, paneles, construcción naval, revestimiento de suelos y enchapados. Es fácil de trabajar tanto con herramientas manuales como mecánicas.

Fresno común (*Fraxinus excelsior*, oleáceas). Densidad: 650-800 kg/m^3. Zonas de producción: Francia, Reino Unido, Alemania e Italia. El tronco puede alcanzar una altura máxima de 10 m y tiene de 0,30 a 1,00 m de diámetro. Tiene una veta recta y más bien gruesa. Es resistente y dura y destaca por su gran capacidad para el plegado. Se utiliza mucho en el revestimiento de suelos y en la fabricación de partes curvadas necesarias en la construcción de barcos, contrachapados, muebles, obras de carpintería, paneles y enchapados. La madera de fresno se trabaja muy bien tanto con herramientas manuales como mecáncias.

Haya (*Fagus sylvatica*, fagáceas). Densidad: 600-750 kg/m^3. Zonas de producción: Francia, Reino Unido. Alemania. El tronco tiene entre 5 y 15 m de altura y un diámetro que oscila entre los 0,40 y los 0,90 m. La corteza es homogénea y de una tonalidad gris clara. La especie europea produce una madera funcional dura pero de dimensiones algo inestables. Se emplea ampliamente en la fabricación de armazones para muebles, carcasas, revestimientos de suelos, contrachapados y algunos artículos domésticos tales como bandejas, aunque raras veces con fines decorativos.

Imbuia (*Phoebe porosa*, lauráceas). Densidad: 960 kg/m^3. Zonas de producción: Brasil, Argentina, Paraguay. Puede alcanzar los 40 m de altura con un diámetro de 1,80 m aproximadamente. La imbuia sudamericana es de color marrón oscuro con un veteado excepcionalmente fino. Es una de las maderas con mayor densidad. Y una de las principales especies comerciales en Brasil debido a su valor en la manufactura de muebles de gama alta. También se emplea en la realización de contrachapados decorativos y de revestimientos de suelos.

Nogal (*Juglans regia*, juglandáceas). Densidad: 600-800 kg/m^3. Zonas de producción: Francia, Reino Unido, Alemania, Italia, España, Turquía, Irán, Cachemira. El tronco tiene de 2 a 5 m de altura y de 0,40 a 0,80 m de diámetro. El duramen es de color variable pero generalmente muestra un fondo marrón grisáceo con rayas oscuras, con una textura gruesa de vetas alternativamente rectas y onduladas. Se distingue entre otras cosas porque se presta muy bien al curvado por vapor. Su disponibilidad actual es más limitada y se emplea principalmente en la realización de muebles de gama alta, piezas torneadas y enchapados.

Nogal negro (*Juglans nigra*, juglandáceas). Densidad: 500-600 kg/m^3. Zonas de producción: Estados Unidos. Su tronco tiene entre los 8 y los 15 m de altura y de 0,35 a 0,90 m de diámetro. Es una

madera fuerte y sólida, de color marrón violáceo, similar al chocolate y de una textura más bien gruesa pero muy uniforme. El nogal negro es fácil de trabajar tanto con herramientas manuales como con máquinas. Es una de las maderas más utilizadas en la fabricación de muebles y también en enchapados, elementos torneados o instrumentos musicales.

Olivo (*Olea europaea*, oleáceas). Densidad: 750-900 kg/m^3. Zonas de producción: cuenca mediterránea. El tronco puede alcanzar una altura de 2 a 3 m y un diámetro de 0,30 a 0,80 m. Es resistente y duradero y se caracteriza por un grano irregular y de tacto algo aceitoso. Esta aceitosidad hace que no sea fácil encolarla por lo que hay que perforarla y asegurarla con clavos y tornillos. Pero, por otro lado, su acabado es muy bueno. Se emplea en la carpintería de interior, principalmente en suelos, en muebles y en enchapados decorativos.

Olmo (*Ufmus campestris*, ulmáceas). Densidad: 600-850 kg/m^3. Zonas de producción; Francia, Reino Unido, Alemania. El tronco tiene de 3 a 10 m de altura y de 0,40 a 1,30 m de diámetro. Tiene un color marrón más bien apagado, con una textura gruesa de vetas irregulares bastante atractiva. Es difícil de trabajar pero se presta bien al curvado por vapor. Se emplea comúnmente en la construcción de barcos, en la fabricación de muebles, enchapados, revestimientos de suelos y tornería.

Pino. Es uno de los árboles más comunes y existen alrededor de 110 especies. La más corriente es quizás el **Pino silvestre** (*Pinus sylvestris,* pináceas), también denominado pino serrano, pino albar, pino del norte o pino rojo. Crece en zonas muy extensas del norte de Eurasia y su tronco puede alcanzar los 30 m de altura. Tiene una densidad de 500-540 kg/m^3. Se utiliza principalmente para la fabricación de armazones, muebles sencillos y madera laminada pero sus nodos muy visibles descartan su utilización en revestimientos o muebles de mayor calidad. El **Pino Oregón** (*Pseudotsuga menziessi*) es la segunda conífera más alta del mundo (después de la secuoya roja) y llega a alcanzar los 60 o 75 m, con diámetros del tronco de 1,5 a 2,0 m. Tiene una densidad algo menor que el silvestre, 470-520 kg/m^3). Se utiliza principalmente para la construcción de estructuras, armazones y madera laminada.

Roble. Este término designa un gran número de especies del género *quercus*, que en latín abarcaba a los robles y las encinas y que actualmente incluye entre 400 y 600 especies. El **roble común** (*Quercus pedunculata*, *Quercus robur*) designa generalmente al roble europeo, un árbol robusto y majestuoso con un tronco que puede alcanzar los 40 m de altura y que se encuentra en Europa, Asia Menor y norte de África. El roble se caracteriza por sus definidos y amplios anillos de crecimiento de color marrón oscuro brillante. Su madera posee unas excelentes propiedades para el curvado por vapor y produce un corazón duradero, muy demandado en los trabajos de carpintería interior y en la realización de muebles. Otras variantes importantes son el roble blanco americano (*Quercus alba*), el roble rojo americano (*Quercus rubra*). La densidad está en torno a los 600-800 kg/m^3.

Secuoya roja o siempreverde (*Sequoia sempervirens*, cupresáceas). Densidad: 400-500 kg/m^3. Zonas de producción: Estados Unidos (California). El tronco puede alcanzar una altura de más de 80 m y un diámetro que alcanza los 8 m. El duramen es marrón rojizo oscuro, con vetas rectas y una elevada estabilidad de dimensiones. La secuoya roja es fácil de trabajar tanto con herramientas manuales como con máquinas. Es una madera muy preciada en la construcción y se usa para la fabricación de vigas resistentes, pilares y postes, puentes y revestimientos exteriores. Y también para puertas, enchapados y muebles.

Teca (*Tectona grandis*, lamiáceas). Densidad: 600-800 kg/m^3. Zonas de producción: la India, Birmania, Tailandia, Vietnam. El tronco tiene de 8 a 15 m y de 0,40 a 0,80 m de diámetro. Se caracteriza por un duramen de color marrón dorado con una fibra recta, algo ondulada, y una textura oleaginosa y gruesa que se hace más bella con los años. Posee asimismo unas excelentes propiedades de resistencia. Es relativamente fácil de trabajar aunque puede llegar a estropear las máquinas. Se emplea comúnmente en la construcción naval y en la realización de tablones, muebles, paneles, tallas, piezas torneadas, obras de carpintería, enchapados y contrachapados.

Tuya occidental o ***árbol de la vida*** (*Thuja occidentalis*, cupresáceas). Densidad: 350-440 kg/m^3. Zonas de producción: Canadá, Estados Unidos. El duramen de la tuya occidental es marrón claro y desprende un perfume aromático y picante. Tiene

una textura fina, de grano uniforme. El corazón es resistente al deterioro y a las termitas subterráneas. La tuya occidental es más fácil de trabajar con herramientas de mano que con máquinas. Se emplea en la construcción de cercados, en la fabricación de tejas de madera, en contenedores para el transporte marino, en la construcción de barcos, en muebles que repelen las polillas y también en la fabricación de guitarras.

Wenge (*Mülettia Laurentii*, papilionáceas). Densidad: 800-950 kg/m^3. Zonas de producción: Camerún, Congo, este de África, Mozambique. El tronco tiene de 8 a 15 m de altura y un diámetro de 0,50 a 1,00 m. El wenge se caracteriza por un exótico duramen de color marrón oscuro con venas muy tupidas y finas, casi negras. La textura es gruesa y el hilo es recto tirando a ondulado. La madera de wenge se emplea para revestir suelos, en obras de carpintería, piezas torneadas, enchapados, armarios y tallas. Funciona bien con herramientas para mecanizado y, una vez pulida, presenta un óptimo acabado.

Características visuales. Métodos de simulación

En una madera cortada pueden distinguirse fácilmente varias partes: la madera juvenil, la albura, el duramen y las vetas. La madera juvenil es la *médula*, la zona central, más ligera y poco resistente. La *albura* es el tejido leñoso con células vivas que transportan nutrientes, joven, poroso, que también es poco resistente y puede ser atacado con facilidad por organismos vivos. Tiene un color más claro que el duramen. El *duramen* es la parte más vieja, sin células vivas, la transformación de la albura en algo más denso, más oscuro e impermeable y más resistente a los ataques. Esta es la parte más utilizada en construcción. Las vetas muestran la dirección de las fibras en relación con el eje. Pueden ser rectas, paralelas al eje, más fáciles de trabajar, o bien onduladas, en espiral o inclinadas, más atractivas o interesantes visualmente pero más difíciles de trabajar.

Las características físicas de anisotropía y heterogeneidad son también características visuales, pues la madera se distingue por sus vetas dominantes en una dirección. El color es muy variable según las especies, desde el blanco al negro pasando por todo tipo de gamas ocres. No solo depende de la especie sino también de la edad. También el brillo es variable y no solo depende de la especie sino del corte y del pulido.

Las dificultades para simular los diferentes tipos de madera son, en cierto modo, similares a las de las piedras, pues las características principales dependen de la textura propia de la madera que, en muchos casos, solo puede simularse a partir de fotografías reales. En el caso de la madera, esta dificultad se incrementa porque la textura depende también del tipo de corte. La figura 6.4 muestra el modo en que el tipo de corte afecta a la textura. En el caso de tableros largos, como los que se utilizan corrientemente en revestimientos de suelos o pare-

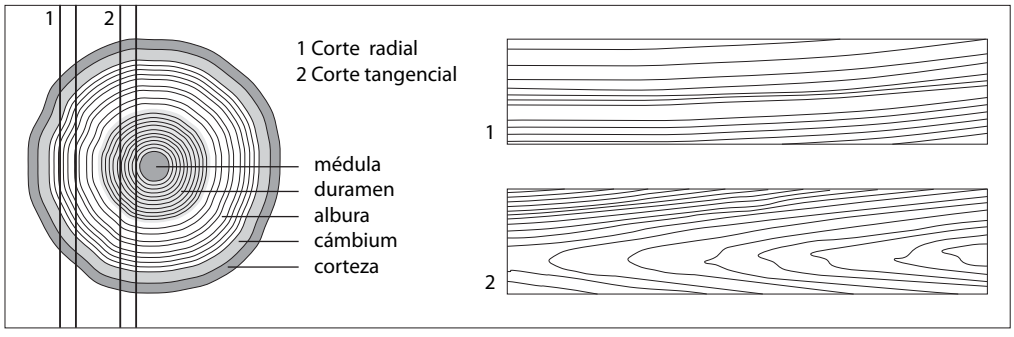


Figura 6.4 Maderas. Esquema de las zonas principales de un tronco de madera que afectan a sus características visuales. A la izquierda se muestra un corte transversal con indicación de las zonas de corte radial y tangencial y de las zonas principales del tronco. A la derecha se muestran las vetas características de un corte radial y un corte tangencial.

Figura 6.5 Maderas. Diferentes pavimentos con diferentes tipos de madera: a*) haya, arce, cerezo;* b*) fresno, roble claro, roble marrón;* c*) roble obscuro, nogal, teca;* d*) maderas envejecidas.*

des, la diferencia principal depende de si el corte es radial o tangencial. El corte transversal, que también se muestra en esta figura, es raramente visible. En el caso de un corte radial, cercano al centro del tronco, el tablero mostrará principalmente vetas longitudinales en bandas paralelas. En el caso de un corte tangencial, el tablero mostrará cómo las vetas se repliegan sobre sí mismas mostrando unos arcos superpuestos característicos que son más acentuados en unas especies que en otras, así como nudos (que no se muestran en la figura) y que también varían considerablemente según los casos.

En general, las características visuales dependen de los factores siguientes:

a) La textura propia de la madera, que depende de su estructura interna y del tipo de corte. Desde este punto de vista, se puede distinguir entre maderas que presentan una textura fina, con pocos nudos y vetas poco marcadas, como el haya o el abedul; maderas con vetas marcadas que pueden apreciarse claramente en los arcos superpuestos que aparecen en cortes tangenciales, pero comparativamente pocos nudos, como el roble o el castaño; y maderas con textura gruesa y nudos claramente visibles, como el pino.

b) El color. Esto tiene que ver, en primer lugar, con el tipo de madera. Hay maderas claras, como el haya, el abedul y el arce; rojizas, como el cerezo, y oscuras, como el nogal, la teca o la caoba. Pero, en segundo lugar, y cada vez más, con el tratamiento a que haya sido sometida la madera y que puede incluir el uso de tintes de todo tipo.

c) El acabado superficial. Si la madera se ha pulido y barnizado resultará bastante reflectante, con un tipo de reflejo que viene muy condicionado por el ángulo de visión (reflexiones Fresnel), como ocurre, en mayor grado, con los metales.

d) La distribución de piezas y las características de las juntas. La problemática es similar a la de las piedras pero más sencilla pues las variantes son mucho menores. Pero es importante tener en cuenta que la textura de un pavimento de madera reflectante debe quedar complementada por una versión en blanco y negro de la misma textura para anular el reflejo en las juntas (y, en su caso, si es una madera vieja, en las vetas agrietadas), tal como hemos visto en el capítulo anterior y se ha hecho en las imágenes de la figura 6.5 que acompañan esta sección.

Estas imágenes muestran algunos ejemplos de pavimentos de madera (parquets, tarimas) de diferentes tipos. En todos los casos se ha aplicado una textura similar, con un mapa UVW plano cuadrado, aplicada a través del canal *Diffuse* de un material de tipo *Arch&Design*. En las figuras con texturas más marcadas (roble, nogal) también se ha utilizado el mismo mapa con las juntas negras y el resto blanco para eliminar el reflejo en las zonas de las juntas, como ya hemos visto en el capítulo 5, en la sección sobre reflejos. Y en las últimas imágenes, de madera envejecida, la reflectividad es nula y se ha utilizado un mapa similar, en blanco y negro, pero aplicado como *Bump* para enfatizar la rugosidad.

La reflectividad varía según los tipos. En el caso de la tarima de haya es de 0,75, 0,1, 16 (*reflectivity*, *gloss*, *samples*) con *brdf custom* 0,5, 1,0, 5,0 (0º, 90º, *curve shape*). En el caso de las tarimas de arce y cerezo es similar, con alguna pequeña variación. Los otros ejemplos son menos reflectantes, aunque los valores son similares, debido entre otras cosas a que se ha utilizado un mapa de reflexión en blanco y negro para atenuar el reflejo en mayor o menor medida, como ya he dicho.

3 Cerámicas

La palabra *cerámica* deriva del griego *kéramos* que significa “cosa quemada”, derivada a su vez de un término sánscrito que se aplicaba a los procesos de calcinación. En Grecia se aplicó al arte de la alfarería y este término se extendió a otros pueblos que ya utilizaban la cerámica desde el neolítico. En la actualidad se aplica a la producción de un material no métalico mediante un proceso que aglutina sus componentes, calentándolos a altas temperaturas.

La mayoría de los materiales cerámicos se producen principalmente a partir de la arcilla aunque la industria actual ha ampliado esta base tradicional. La denominación *arcilla* abarca a un grupo muy amplio de materiales sedimentarios cuyas partículas son menores de 2 micras pero cuya composición mineralógica puede ser muy variada. Pero la pasta cerámica incorpora tres tipos de ingredientes con funciones determinadas: *a*) los elementos plásticos,

como la arcilla y el caolín, que facilitan el moldeado; *b*) los elementos magros, como la sílice, la arena o las arcillas silíceas, que reducen o aumentan la porosidad y facilitan el secado; *c*) los elementos fundentes, como los feldespatos, las micas, la cal o los vidrios pulverizados, que determinan la temperatura de fusión.

La mayor o menor proporción de estos tres elementos está directamente relacionada con las diferencias entre los diferentes tipos de cerámica. A grandes rasgos puede decirse que hay dos categorías principales de materiales cerámicos: los porosos y los relativamente impermeables.

En el primer grupo se incluirían dos subgrupos. Por un lado, lo que se denomina corrientemente **loza** y que abarcaría productos tales como baldosas, ladrillos corrientes, tejas y elementos domésticos como cazuelas o recipientes diversos. La loza común es de un color rojizo característico debido al óxido de hierro de las arcillas y se produce con una temperatura de cocción que varía entre 700º y 1.000º. Si la arcilla es de mayor calidad, eliminando el óxido de hierro y añadiendo otros componentes (sílex, yeso, feldepasto), y la cocción aumenta hasta los 1.200º o 1.300º, se obtiene un producto similar a la porcelana pero no impermeable al que se denomina “loza inglesa”. Por otro lado, tenemos los refractarios, que se obtienen añadiendo otros productos tales como óxido de aluminio, torio, berilio y circonio, aumentando la temperatura de cocción hasta los 1.300º o 1.600º y enfriando el producto lentamente. Así se obtienen ladrillos refractarios o electrocerámicas, que pueden soportar grandes temperaturas de hasta 3.000º.

El segundo grupo principal, los productos impermeables o semiimpermeables, se obtienen por temperaturas más altas que dan lugar a una vitrificación de la arena de cuarzo que proporciona un producto muy duro y con diferentes grados de impermeabilidad. Así se obtienen el **gres común** (a partir de arcillas ordinarias y temperaturas de unos 1.300º), el **gres fino** (a partir de arcillas refractarias y feldespato como fundente, temperaturas de unos 1.300º y una impregnación final con sal marina) y las **porcelanas** (a partir de una arcilla muy pura, el caolín, con adición de feldespato como fundente y cuarzo o sílex como desengrasantes, y dos cocciones, una a 1.000º o 1.300º y otra a una temperatura que puede llegar a los 1.800º).

Aunque se han presentado en otro apartado, los vidrios pueden ser también considerados como materiales cerámicos con la particularidad de que sus constituyentes se calientan hasta llegar a la fusión y luego se enfrían para llevarlos a un estado rígido sin cristalización, a una estructura no ordenada, al contrario de lo que ocurre con los sólidos cristalinos.

Características físicas

Los materiales cerámicos son inertes químicamente y tienen una estructura interna compleja que se deriva de la unión de diferentes iones mediante fuertes enlaces, lo que contribuye a una gran resistencia estructural. De aquí derivan sus propiedades más conocidas y más utiles: su resistencia mecánica, su resistencia térmica a altas temperaturas, su baja conductividad eléctrica y su estabilidad en ambientes agresivos.

La mayor o menor proporción de arcilla y agua afecta a la plasticidad de la masa o pasta cerámica. Esta es la característica más notable y que permite moldearla con facilidad. La plasticidad, que se define como la capacidad de un material para ser deformado sin que se rompa, depende no solo de la proporción entre arcilla y agua sino también de la calidad de la arcilla (composición mineralógica, granulometría, contenido de coloides y sales solubles, etc.).

La ***densidad*** es muy alta, con valores que van desde los 1.700 kg/m^3 a los 2.800 kg/m^3. Algunos valores de referencia, de menor a mayor, son los siguientes: baldosas (1.750), baldosín catalán (1.750), ladrillo macizo (2.300), azulejo (2.300 kg/m^3), gres de sílice (2.200 a 2.600), gres de cuarzo (2.600 a 2.800).

La ***conductividad térmica*** es relativamente baja por lo que su capacidad aislante es buena, algo menor que la de la madera. Algunos valores característicos, en W/m.K, de mayor a menor, son los siguientes: gres de cuarzo (2,60), gres de sílice (2,30), azulejo (1,30), baldosín catalan (1,05), baldosas (0,81), ladrillo macizo (0,85), ladrillo perforado (0,35), ladrillo hueco (0,32).

La **resistencia mecánica** de los materiales cerámicos es, en general, elevada para la compresión

uniaxial por lo que se utilizan generalmente bajo cargas de compresión. Por otro lado, su fragilidad y resistencia a tracción es mucho más baja, entre 5 y 10 veces inferior a la de compresión. Los ladrillos macizos corrientes tienen una resistencia que siempre es mayor que los morteros con que se unen. La resistencia mínima debe ser superior a los 35 kg/cm^2, pero normalmente está comprendida al menos entre 70 y 200 kg/cm^2 y puede alcanzar los 300 en los ladrillos de primera calidad. Hay, por otro lado, ladrillos especiales que pueden alcanzar hasta los 1.400 kg/cm^2.

La ***dureza*** es una característica importante pues puede marcar la diferencia entre su uso como revestimiento para suelos o para otras aplicaciones. Las baldosas blandas, con valores entre 3,0 y 4,0 en la escala de Mohs, no son muy recomendables para suelos (en general se recomienda una dureza Mohs de al menos 5,0 para suelos residenciales). Las baldosas vidriadas, con mayor contenido en feldespatos, están en torno a 6,0, las baldosas vidriadas, con mayor contenido en cuarzo, en torno a los 7,0. Las porcelanas sin esmaltar o productos equivalentes tienen valores en torno a 7,0 o más y las de sin esmaltar de alta calidad o esmaltadas pueden alcanzar valores de 8,0 o más, razón por la que se han utilizado tradicionalmente como un producto preferente para el revestimiento de paramentos interiores.

Productos cerámicos más utilizados en la construcción

Cerámica estructural. La utilización de la cerámica como elemento estructural es tan antigua como la civilización: los primeros muros de adobe, que combinaban arcilla, arena, paja y otros elementos vegetales, se construyeron hace al menos 8.000 años aC. Las primeras referencias al uso de ladrillo cocido vienen de Mesopotamia, unos 2.800 años aC. A partir de esas fechas, y hasta hoy, se convirtieron en el material más utilizado en construcción.

Los elementos más corrientes son: *a*) los *ladrillos*, que se producen en una amplísima gama de tipos, tanto por lo que hace a la composición, al grado de cocción o a las dimensiones, si bien la temperatura de cocción, como la de los demás productos de este apartado, es inferior a la de la mayoría de los productos cerámicos, entre los 900 y los 1.000º; *b*) las *tejas* cerámicas, obtenidas por diversas técnicas específicas de moldeado; *c*) las *bovedillas* cerámicas, que se utilizan como elementos, raramente visibles, de relleno de forjados; *d*) los *bloques* cerámicos, como los bloques de termoarcilla, de dimensiones mayores y destinados a ser recubiertos por lo que, en principio, tampoco suelen ser visibles.

Baldosas. La utilización de baldosas para revestimientos y pavimentos, con fines tanto funcionales como decorativos, se remonta a unos 1.000 años aC pues se han encontrado restos en Egipto, Caldea, Asiria, Grecia y Roma, si bien su uso no se extendió de modo significativo hasta los siglos x y xi y de modo principalmente en el mundo islámico. El término *alicatar* viene del árabe y se basa en el corte con una herramienta especial, una *al-laqqat*, de donde deriva también *alicate*, las piezas de cerámica vidriada.

Las baldosas actuales están constituidas por un cuerpo arcilloso cocido (el "bizcocho"), de porosidad variable. Si la porosidad y, por tanto, su capacidad de absorción de agua es alta, se denomina *loza* y se utiliza principalmente para revestimientos rústicos, y si el cuerpo arcilloso está vitrificado se denomina *gres* o *porcelana*, como ya hemos visto antes. Sobre este cuerpo arcilloso se puede aplicar un recubrimiento de naturaleza vítrea, el esmalte cerámico, lo que lleva a la distinción corriente entre baldosas *no esmaltadas* (*ugl*, *unglazed*, en inglés) y *esmaltadas* (*gl*, *glazed*, en inglés).

El color en cocido del cuerpo cerámico lleva también a la distinción entre baldosas con soporte de pasta roja o de pasta blanca, según el tipo de materia prima utilizado en su fabricación.

Las baldosas se clasifican según tres características principales: el tipo de moldeo (prensado o extruido), la porosidad (alta o baja) y si son o no esmaltadas. A partir de esta clasificación se distinguen los siguientes tipos más corrientes.

a) *Azulejos*. Es la denominación tradicional (del árabe *az-zulaiy*, piedra pulida) de las baldosas cerámicas porosas, prensadas en seco y esmaltadas. Su absorción de agua es del orden del 11 % al 15 % lo que permite, por su parte trasera, no esmaltada, que se adhieran fácilmente al muro. La cara vista está cubierta de un esmalte vitrificado con colores

o patrones impresos muy variados. Los espesores característicos son menores de los 10 mm. Se utilizan principalmente en revestimientos de paredes interiores.

b) *Gres*. Se denominan así las baldosas porosas, prensadas en seco y esmaltadas, es decir, de características similares a los azulejos. Pero son más densos y con mayor resistencia a la abrasión, con una absorción del agua menor, entre el 2 % y el 6 %, debido a una mayor vitrificación por un proceso de cocción con temperaturas de entre 1.000º a 1.200º. Por estas razones se utilizan corrientemente como pavimentos en suelos interiores. Su espesor suele ser del orden de los 8 mm y este mayor espesor aumenta su dureza y resistencia.

c) *Gres porcelánico*, *baldosín catalán*. El término gres porcelánico se aplica a un producto más reciente, prensado en seco y no esmaltado, pero con muy baja absorción de agua, del orden de 0,1 %. También hay un "gres porcelánico esmaltado" con la capa exterior tratada con esmalte por razones principalmente estéticas. El gres porcelánico no esmaltado puede tener su cara vista tratada de muy diferentes modos, desde acabados pulidos, para interiores, hasta acabados más rugosos, antideslizantes. El "baldosín catalán" es la denominación tradicional para un tipo de baldosa con capacidad de absorción de agua media/alta o alta, que se moldea por extrusión y que se comercializa generalmente sin esmaltar. El cuerpo de la baldosa es de color rojo o rojo/pardo, de textura poco homogénea. La cara vista es, en principio, igual que el cuerpo pero puede ser tratada para aumentar su resistencia, lo que afecta a su aspecto.

d) *Gres rústico*, *terracota*. El gres rústico es una denominación que se reserva para baldosas no porosas, moldeadas por extrusión y no esmaltadas. La absorción de agua está entre el 1,5 % y el 6 %. Se utiliza para revestimiento de fachadas y pavimentos industriales o espacios públicos. La terracota o barro cocido se aplica a una gran variedad de baldosas, porosas, moldeadas por extrusión o a mano, no esmaltadas. La absorción de agua está entre el 6 % y el 15 %. Tienen una textura muy heterógenea y un aspecto atractivo. La cara vista es, en principio, igual a la del cuerpo aunque puede

Figura 6.6 Cerámicas. Escenario con diferentes tipos de revestimientos cerámicos.

verse alterada por tratamientos que aumenten su resistencia.

Cerámica sanitaria. Esta denominación se aplica a productos tales como lavabos, bañeras, platos de ducha, inodoros, fregaderos, etc., es decir, aparatos de aseo o de cocina. Se frabrican con materias primas en las que predominan el caolín y las arcillas refractarias, el feldespato y el cuarzo, pastas blancas cocidas a altas temperaturas hasta alcanzar un estado de semivitrificación, para asegurar una alta resistencia mecánica, y posteriormente esmaltadas para aumentar su resistencia al desgaste de todo tipo. Como sus formas son complejas, la producción se lleva a cabo por medio de moldes especiales.

Cerámica decorativa. Cerámica artesanal. Bajo la denominación de cerámica decorativa se agrupa generalmente todo tipo de productos que van desde platos, tazas o lámparas a jarrones o esculturas. El material más utilizado es la porcelana, que se crea a partir de pastas blancas constituidas principalmente por caolín, feldepasto y cuarzo con procedimientos de cocción a muy altas temperaturas, hasta 1.400º. En el otro extremo, se enlaza con productos similares pero elaborados con loza, barro cocido a temperaturas más bajas y que caerían, por oposición, bajo la denominación general de "cerámica artesanal" o "cerámica tradicional".

Características visuales. Métodos de simulación

Los materiales cerámicos constituyen uno de los tipos de acabados más frecuentes en todo tipo de paramentos, pavimentos, muros o techos. Desde el punto de vista de su apreciación visual, la principal diferencia entre la cerámica estructural y la cerámica utilizada para revestimientos es que la primera tiene, por lo general, un acabado mate y relativamente tosco mientras que la segunda tiene un acabado pulido y brillante. El primer caso no presenta otra dificultad técnica, para su simulación visual, que encontrar una imagen adecuada de suficiente amplitud. Y los tipos de problemas y técnicas son muy similares a los de la piedra, por lo que ahorraré comentarios al respecto. Hay múltiples texturas por internet para simular ladrillos o baldosas rústicas, y ya hemos visto unos cuantos ejemplos en capítulos anteriores por lo que me remito a esas referencias y a esos ejemplos.

En el caso de la cerámica utilizada como revestimiento, las posibilidades son infinitas y la historia de la arquitectura y, sobre todo, de la arquitectura islámica, muestra la riqueza de posibilidades de combinaciones cromáticas que se pueden obtener con estos materiales. Por otra parte, el hecho de que puedan utilizarse piezas de pequeño tamaño facilita su adaptación a todo tipo de superficies curvas.

Desde el punto de vista técnico, el modo más adecuado de simular una cerámica brillante es mediante un material *Arch&Design* o similar, con valores de reflectancia altos, combinados con mapas de relieve para enfatizar los resaltes y, sobre todo, con una iluminación adecuada que potencie los reflejos.

La figura 6.6 muestra un escenario en el que se han utilizado revestimientos cerámicos de varios tipos y colores. En el caso del suelo y del plano de fondo, todo lo que he hecho es preparar texturas adecuadas combinando digitalmente texturas diversas. En el caso del muro curvo he utilizado un material *Arch&Design* con la configuración siguiente: *Diffuse color*: mapa de bits (piezas de cerámica roja que abarcan aproximadamente la mitad de la altura para proporcionar suficiente variación y evitar repeticiones); *Reflection*: 0,85, 0,85, 16 (*reflectivity*, *glossiness*, *samples*); BRDF: 0,65, 1,0, 3,5 (0º, 90º, *curve shape*); *Relieve*: *bump* a 1,2 y mapa de bits (versión en blanco y negro con el contraste aumentado del mismo mapa de bits de las cerámicas).

4 Vidrios

El vidrio puede considerarse también un material cerámico pues se obtiene a partir de la cocción de arena de sílice (con carbonato de sodio y caliza) a unos 1.500 ºC. Pero a diferencia de aquellos, sus componentes se llevan a temperatura de fusión y luego son enfriados hasta que alcanzan un estado rígido sin cristalización, lo que da como resultado una estructura amorfa, con las moléculas dispuestas de modo aleatorio. En su estado más corriente es un material transparente, que puede moldearse con facilidad si se calienta en hornos especiales a

temperaturas del orden de los 975º (vidrio de plomo) a 1.020º (vidrio común o sodocálcico). A temperaturas intermedias es bastante plástico y puede cortarse con un cuchillo. A temperatura ambiente se quiebra con facilidad.

El vidrio se utiliza en construcción desde la época neolítica, unos 10.000 años aC, y los primeros vidrios tallados se han encontrado en la época de los faraones del Antiguo Egipto y se frabricaron con arena fundida y algas. El vidrio en bruto, cuyos orígenes se remontan quizás más allá de los 4.000 años aC, tiene un color verdoso o acaramelado y no es transparente. Para que lo sea es preciso utilizar otras técnicas de fabricación que disminuyen su espesor y añadir otras substancias, como determinados óxidos metálicos. Los primeros vidrios más o menos transparentes aparecieron hacia 1.500 aC en Siria. Y se extendieron por Grecia y por Roma, donde las familias ricas utilizaban vidrio plano para las ventanas que, aunque no era transparente, dejaba pasar la luz. Hacia el siglo VII aparecieron, en Siria, los primeros métodos de fabricación con una caña hueca, no soplada sino que se hacía girar rápidamente. En la Edad Media, en los siglos XII y XIII se comenzó a utilizar el método de soplar a través de una caña hueca, haciéndola oscilar, para dar forma al vidrio. El uso de técnicas de coloración del vidrio, descubiertas por los árabes y llevadas a Europa a través de España, desde Damasco y Aleppo, en Siria, se aplicó a las grandes vidrieras de las catedrales y surgieron algunos centros famosos de fabricación del vidrio, como los que había en la isla de Murano, en Venecia, hacia el siglo XIV. A finales del siglo XVII, en Inglaterra, se empezó a elaborar el cristal de plomo o vidrio Flint, mediante la agregación de plomo a la mezcla. Posteriormente las industrias de vidrio se extendieron a muchos países y la lista de las industrias notables en diferentes países europeos se incrementó espectacularmente.

Características físicas

La mayoría de los vidrios tienen como componente principal el sílice. También abundan los vidrios de borato (B2O3) aunque no se utilizan corrientemente en construcción. Los más utilizados en construcción son los de tipo cálcico sódico, a partir de la mezcla y fusión de arena, silíce, caliza y sosa, que suponen en torno a un 90 % del total.

La **densidad** del vidrio corriente está en torno a los 2.500 kg/m^3 o algo más Los vidrios que incorporan plomo pueden llegar a los 6,0 g/cm^3.

Su **resistencia** a la compresión es muy alta. Se requieren del orden de los 10.000 kg para romper un cubo de vidrio de 1 cm de lado. La resistencia a la tracción es mucho más baja, entre 300 y 700 kg/cm^2 y disminuye aún mas, en torno al 40 %, para cargas permanentes. También disminuye con la temperatura.

La durabilidad es alta y resiste bien los ácidos aunque es atacado por substancias básicas y alcalinas. Los valores de ***dureza*** Mohs del vidrio corriente que se utiliza en ventanas está en torno a los 5,5 y los valores generales están entre 5,0 y 7,0.

La **conductividad térmica** es algo superior a 1,00 W/m.K, con valores en torno a 1,40 para el vidrio de cuarzo y valores similares para el vidrio prensado, superiores a los de la mayoría de las cerámicas. El vidrio no es un buen aislante térmico por lo que debe recurrirse a ventanas de vidrio doble para mejorar su capacidad aislante.

Vidrios más utilizados en la construcción

Vidrio aislante. Se denomina así al vidrio formado por al menos dos piezas de vidrio separadas por una cámara de aire. El aislamiento puede reforzarse rellenando la cámara de aire con gases nobles, principalmente argón o kriptón.

Vidrio armado. Se obtiene mediante un proceso de colado durante el cual se inserta una malla metálica, una trama formada por alambres metálicos. Si el vidrio sufre un impacto, los fragmentos quedan unidos al alambre lo que evita que puedan herir a alguien. No se utiliza en casos en los que sea previsible que se alcancen temperaturas muy altas pues la dilatación y el comportamiento general del metal y el vidrio son diferentes y se podrían producir fracturas espontáneas.

Vidrio laminado. Consiste en varias láminas de vidrio unidas mediante una película plástica situada entre ellas. Si el vidrio sufre un impacto la película mantiene los fragmentos unidos lo que crea un agrietamiento característico en forma de tela de araña y evita que estos fragmentos puedan herir

Figura 6.7 Vidrios: a) *Vidrio simple;* b) *Vidrio coloreado;* c) *Vidrio esmerilado;* d) *Pavés.*

a alguien. La lámina intermedia puede ser transparente, translúcida, de colores diferentes, de papel o de tela que incluyan dibujos o motivos decorativos. Puede incorporar diodos LED, puede recibir tratamientos que aumenten su capacidad de aislamiento acústico o puede incluir elementos que capten y redistribuyan la energía solar. Por todas estas razones ha adquirido una importancia considerable en la construcción contemporánea.

Vidrio templado. El vidrio puede templarse mediante tratamientos químicos (se sumerge el vidrio en una solución salina a temperatura elevada y con alta concentración de iones de potasio) o térmicos (se calienta el vidrio ya cortado hasta temperaturas del orden de los 700º y se enfría bruscamente en cámaras especiales). Esto aumenta su resistencia estructural y su resistencia al impacto. Si el vidrio sufre un impacto se fragmenta en trozos pequeños y redondeados que no son peligrosos. Por esta razón se utiliza habitualmente en espacios públicos tales como fachadas, vestíbulos, barandas de escaleras, etc.

Bloque de vidrio (***pavés***). Los bloques de vidrio (*glass bricks* en inglés) o pavés se comenzaron a utilizar en el Reino Unido en la década de 1880, unidos inicialmente con mortero y con resultados no demasiado satisfactorios. Sin embargo, en Alemania, las compañías Luxfer-Prismen y Siemens fabricaron bloques de mejores características y su uso se extendió, principalmente en la década de 1930, para separar zonas en las que interesaba preservar la visibilidad pero permitir la entrada de luz. La Maison de Verre, de Pierre Chareau, con la colaboración del arquitecto holandés Bernard Bijvoet, construida entre 1928 y 1932 en París y que incluía un gran muro de pavés, fue un hito importante que marcó su integración en otras obras de arquitectura notables.

Características visuales. Métodos de simulación

El porcentaje de transmisión de la luz directa de un vidrio corriente de 4 mm está en torno al 92 % y el de 6 mm, en torno al 90 %. La transmisión de luz indirecta es algo inferior, en torno a un 87 % para 4 mm y 85 % para 6 mm. El índice de refracción del vidrio corriente es de 1,52 aproximadamente.

El vidrio puede colorearse mediante la adición de óxidos metálicos durante la fase de cocción, con lo que pasan a formar parte permanente del vidrio. Ejemplos de colores que se pueden obtener con facilidad mediante la adición de compuestos metálicos son los siguientes: rojo (cobre), azul (cobalto), verde (hierro) o amarillo (manganeso).

Otra posibilidad es pintar sobre el vidrio y sellar el resultado con resinas o calor para mejorar la estabilidad. Los colores resultantes tienen menos pureza que con el método anterior.

Otro recurso tradicional es la unión de piezas pequeñas de vidrio mediante tiras de plomo, generalmente con una sección en H. Una vez colocadas las piezas, se sueldan las juntas superiores por cada uno de los lados y se rematan las juntas con un cemento especial que contiene plomo, para asegurar la estanqueidad. Y otro uso relativamente frecuente, al que ya me he referido anteriormente, es la creación de bloques de vidrio unidos por gruesas juntas de cemento.

Ya hemos visto en el capítulo anterior que la simulación visual del vidrio, en los casos más corrientes que se presentan en la práctica, no ofrece ninguna dificultad importante. En la mayoría de los casos nos encontraremos con casos simples como los que se resumen en las primeras tres imágenes de la figura 6.7: vidrio simple, vidrio coloreado y vidrio que dispersa la luz de diversos modos (esmerilado, etc.). Si se utiliza el material *Arch&Design* u otro similar, las propiedades básicas proporcionan directamente estos efectos con facilidad. En el caso de la cuarta imagen de la figura 6.7, un murete de pavés, el principal problema es conseguir una textura adecuada. Luego puede generarse una segunda textura en blanco y negro, a partir de esta, de tal modo que las partes negras correspondan a las partes no transparentes (las juntas en este ejemplo), y las blancas a las transparentes o, como en este caso, semitransparentes.

5 Metales

La utilización de los metales en la fabricación de utensilios diversos ha afectado de modo directo a la civilización como lo prueba las denominaciones de Edad del Cobre (entre 9500 y 6000 aC en Irak y

Anatolia), Edad de Bronce (antes del 3.000 aC en Asia Menor) o Edad de Hierro (hacia el 1100 aC en Oriente Próximo, India y Grecia) que marcaron su uso, entre otras cosas como arma hegemónica. Pero su aplicación en construcción fue insignificante, debido principalmente a su coste. Los metales no se encuentran en la naturaleza en estado puro sino en compuestos minerales y se necesita utilizar técnicas complejas para extraerlos. El hierro no se utilizó como elemento estructural hasta el siglo XVIII (en 1706 se fabricaron columnas de fundición para la construcción de la Cámara de los Comunes en Londres). Pero se empleó extensamente en el siglo XIX con la Revolución Industrial y la construcción de vías de ferrocarril con técnicas que luego se exportaron a la arquitectura. En 1836 aparece el perfil en doble T que comienza a substituir a la madera como elemento estructural. Y en 1855 Henry Bessemer inventó el método que lleva su nombre y que permitiría producir acero a un coste relativamente bajo. El Palacio de Cristal de Joseph Paxton en Londres (1851), la Biblioteca de Sainte Geneviève de Henri Labrouste en París (1851), la Galerie de Machines de Ferdinand Dutert y Victor Contamin, en París (1889), o la Torre Eiffel de París (1889) son algunos ejemplos históricos que marcan esta evolución.

También durante el siglo XIX se encuentran métodos eficaces de extracción del aluminio a partir del primer paso dado en 1825 por el físico danés Hans Christian Oersted y refinado dos años después por Friedrich Wöhler, que permitía aislarlo aunque a un coste elevado. Diferentes aportaciones durante el último cuarto del siglo XIX fueron reduciendo este coste hasta que la producción de aluminio fue aumentando espectacularmente, de poco más de 2 toneladas anuales en todo el mundo, en 1900, a 6.700 en 1939, 2 millones en 1943 y cifras muy superiores en los años posteriores, que han llevado a que después del hierro sea el metal más utilizado en la actualidad.

El uso de los metales se ha extendido debido a una serie de propiedades determinadas: su dureza y resistencia, combinadas con un peso inferior al de otros materiales con propiedades similares, junto con una flexibilidad mayor. Otras propiedades importantes son su facilidad de reciclado y su relativa facilidad de manipulación. Se ha utilizado y se utiliza cada vez más en estructuras, instalaciones auxiliares, elementos constructivos diversos y en revestimientos decorativos de muy diversos tipos.

Características físicas

Los metales se caracterizan principalmente por poseer una estructura cristalina con electrones libres. De esta estructura derivan sus propiedades más notables, como la alta resistencia y la elevada conductividad térmica y eléctrica. Las principales características físicas de los metales se pueden agrupar en mecánicas, químicas, térmicas y eléctricas.

Desde el punto de vista mecánico, los rasgos principales son su resistencia a la compresión y, sobre todo, a la tracción, muy superior a la de cualquier otro material. También tienen una elasticidad superior a la de otros materiales, con mayor capacidad para deformarse sin romperse y de volver a su estado primitivo al cesar las cargas. Esta característica puede ser modificable mediante tratamientos térmicos o mecánicos.

Desde el punto de vista químico, los metales tienen un comportamiento que depende de múltiples variables. Les afectan principalmente las condiciones ambientales tales como la temperatura, la humedad y la presencia de agentes diversos en el aire. Los efectos principales a que estos dan lugar son la oxidación y la corrosión. La oxidación se ve frenada por dos factores: el primero es debido a que la cantidad de oxígeno activo en el aire es muy pequeña, y el segundo, que los metales, al oxidarse, se recubren de una capa de óxido que impide que la oxidación progrese. La corrosión es más grave. Se produce cuando el metal entra en contacto con electrolitos, sobre todo a través del agua, que cataliza las reacciones químicas y en mucho mayor grado si además hay ácidos o sales. Pero en este caso no se detiene y puede provocar daños irreparables en el material. Para prevenirla se utilizan pinturas especiales que protegen el metal.

Desde el punto de vista térmico y eléctrico las propiedades principales son la dilatación y la conductividad. La dilatación varía considerablemente entre los diferentes metales y se mide por el coeficiente de ***dilatación lineal*** que está entre 1,1 (x 10^{-5}) para el acero, hasta 3.0 (x 10^{-5}) para el plomo (otros valores son hierro 1,2, cobre 1,7, aluminio 2,4, zinc 2,6, titanio 2,8 a 3,5). La conductividad de

los metales es muy alta y varía con el espesor, la temperatura y el tiempo. La importancia de la conductividad de los metales es importante desde el punto de vista de las instalaciones y las construcciones eléctricas pero menos desde el punto de vista de la arquitectura y la construcción.

Por su ***densidad*** se distinguen a veces entre metales pesados (más de 4.500 kg/m^3) y ligeros (menos de 4.500 kg/m^3). Los valores de densidad son muy altos, con bastantes variaciones. El metal más ligero, el aluminio, tiene aproximadamente la misma densidad que el granito. Algunos valores de referencia, de menor a mayor, son: aluminio (2.700), titanio (4.500), cromo (7.160), zinc (7.200), estaño (7.300), hierro de fundición (7.500), hierro forjado (7.690), acero (7.800), acero inoxidable (8.000), latón (8.400), bronce (8.700), cobre (8.900), plomo (11.300), oro (19.300).

Su ***resistencia mecánica*** es muy alta, los metales son los materiales que presentan mayor resistencia a la compresión y la tracción. Con todo, sus valores son muy variables en función de las aleaciones. Así, los aceros dulces y semidulces (con porcentajes de un máximo de 0,25 % y 0,35 % de carbono) tienen resistencias de compresión del orden de los 4.800 a 6.200 kg/cm^2 y de tracción de 4.200 a 5.100. Los semiduros y duros (con porcentajes de hasta un 0,45 % y 0,55 % de carbono) alcanzan resistencias a compresión de 6.200 a 7.500 kg/cm^2. El acero templado puede alcanzar los 10.000 kg/cm^2. Otro tanto ocurre con el aluminio, que en estado puro tiene una resistencia a la compresión relativamente baja, en torno a los 1.000 kg/cm^2 y entre 1.600 a 2.000 para tracción. Pero con aleaciones puede aumentarse esta resistencia hasta los 6.000 kg/cm^2 con valores característicos del orden de 1.900 kg/cm^2 (magnesio), 4.420 kg/cm^2 (cobre) o 5.000 kg/cm^2 (zinc). El acero tiene también un módulo de elasticidad muy alto, del orden de los 2.100.000 kg/m^2 o algo menos, lo que equivale a decir que es muy rígido (el módulo de elasticidad o módulo de Young se define por la relación entre la presión y la deformación), más de tres veces superior a las aleaciones de aluminio. Los de otros metales son inferiores: níquel (2.000.000), cobre (1.200.000), bronce (1.050.000), zinc (965.000). El plomo tiene un módulo muy bajo (es muy deformable), en torno a los 950.000.

La ***conductividad térmica*** es muy alta pues los metales son muy malos aislantes. Los siguientes valores aproximados de referencia, en W/m.K, están ordenados de mayor a menor conductividad (menor a mayor capacidad aislante): cobre (380), oro (308), aluminio (230), bronce (180-120), latón (110-80), zinc (140-110), cromo (95), hierro (80), estaño (65), acero (50), plomo (35), titanio (22).

La ***dureza*** es muy variable y puede ser muy alta en casos especiales. El carburo de tungsteno, utilizado en joyería, alcanza los 8,5 a 9,5 en la escala de Mohs, es decir, que solo puede ser rayado por el diamante. Las aleaciones de acero con carbón son tanto más duras cuanto más carbón incluyan y según, también, determinados procesos especiales de fabricación. Hay aleaciones de acero al carbón que alcanzan los 8,5 a 9,0 en la escala de Mohs y se utilizan para fabricar herramientas especiales. El acero endurecido mediante métodos diversos que van desde el martilleo clásico de los herreros hasta procedimientos industriales para ondularlo y someterlo a presión, está también en torno a los 8,0. El acero templado tiene una dureza de 7 a 8 en la escala de Mohs y se utiliza para herraduras, clavos, vigas de ferrocarril, etc. Otros valores de referencia son los siguientes: titanio (6,0), hierro (4,0 a 5,0), níquel (4,0), cobre, latón, bronce (3,0).

Aunque los metales no son inflamables, pierden su estabilidad a temperaturas altas. Los datos siguientes son valores de referencia para el punto de fusión, de menor a mayor, en grados centígrados: estaño (240), plomo (330), zinc (420), aluminio (630), bronce (900), latón (950), plata (960), oro (1.064), cobre (1.050), acero (1.370 a 1.430 según contenido de carbono), acero inoxidable (1.430), níquel (1.450), hierro forjado (1.580), titanio (1.660).

Metales más utilizados en la construcción

En 2008, la producción total de metales en el mundo era de unos 1.400 millones de toneladas. De este total, el mayor porcentaje, con diferencia, correspondía al acero, con un 95 %, en torno a las 1.320 millones de toneladas. El segundo, a gran distancia, era el aluminio, con una producción estimada de 39,7 millones de toneladas por esas fe-

chas. El tercero era el cobre y el cuarto, el zinc. Este volumen total ha crecido notablemente en los últimos años pasando a ser del doble de lo que era a finales de la década de 1970 y siete veces lo que era hacia 1950. Gran parte de este aumento de producción ha sido debida a la irrupción de China (con una cuota del 38 % de la producción del acero en 2008, seguida a gran distancia por Japón con un 9 % y Estados Unidos con un 7 %, y con una cuota de un 34 % de la producción de aluminio).

Acero. El hierro y sus aleaciones constituyen la parte principal del uso de los metales en construcción. Pero el hierro, que comenzó a emplearse hace más de doscientos años, no se utiliza en la actualidad debido sobre todo a su fragilidad, por lo que la mayoría de las aplicaciones se basan en el acero. El acero es básicamente una aleación de hierro y carbono con un porcentaje de este último inferior al 2 % (entre el 0,03 % y el 1,76 %). Por encima de este valor se denomina *fundición* (*iron cast*, normalmente entre un 2,5 % y un 4,5 %) y es mucho más frágil y no puede forjarse sino que debe moldearse. Aunque el hierro sea el metal dominante, la adición de carbono mejora sus propiedades fisico-químicas. Hay una gran variedad de aceros que se clasifican por lo general a partir de su composición, que puede incluir cromo y níquel (que mejoran también la resistencia al desgaste y la corrosión). El *acero inoxidable* contiene al menos un 10 % de cromo y otros metales como el níquel, el molibdeno, el titanio o el wolframio, y su contenido en carbono es inferior al 1.2 %.

Alumino. Se produce a partir de la bauxita (principalmente en Australia, China, Brasil e India). El nombre proviene la localidad francesa de Les Baux, en la Provenza francesa, donde fue descubierto en 1921. El proceso de extracción es complejo y requiere un considerable gasto de energía. Tiene propiedades muy útiles en la construcción: baja densidad, en torno a 2.700 kg/m^3 y alta resistencia a la corrosión. Puede aumentarse su resistencia mecánica (hasta los 690 MPa) por medio de aleaciones, principalmente con magnesio, manganeso, cobre, zinc y silicio. Aunque requiere una elevada cantidad de energía eléctrica para producirse, resulta comparativamente muy barato por su elevada vida útil y su bajo coste de reciclado. Se utiliza en una gran variedad de aplicaciones, principalmente para la creación de perfiles laminados de carpinterías metálicas y revestimientos,

El *aluminio anodizado* es aluminio protegido por una capa de óxido obtenida mediante un tratamiento especial por el que se forma una capa de óxido de aluminio sobre la superficie del metal. Esta capa proporciona una gran resistencia a los agentes químicos y se puede teñir de diferentes colores.

Cobre. Fue uno de los primeros metales utilizado por los humanos, tanto directamente, al ser uno de los pocos metales que puede encontrarse en la naturaleza en estado casi nativo (los únicos que se encuentran en estado puro son los metales preciosos: el oro, la plata y el platino), como en aleación con el estaño para formar el *bronce*. También se ha utilizado extensamente en aleación con el zinc para formar el *latón*. Es dúctil y maleable, muy duradero, se puede reciclar indefinidas veces sin que pierda sus propiedades principales. Aunque perdió importancia en el siglo XIX con el desarrollo de la siderurgia, la adquirió de nuevo en aplicaciones eléctricas por su alta conductividad y maleabilidad y sigue siendo el material más utilizado para fabricar cables y otros componentes eléctricos y electrónicos. Se sigue utilizando como revestimiento por sus notables cualidades visuales, su color dorado rojizo de gamas variables.

Zinc. Las aleaciones de zinc, principalmente el latón, se han utilizado desde hace siglos. Se han encontrado piezas datadas entre 1.000 y 1.500 años aC en Canaan. La fundición y extracción de zinc impuro se hizo hacia el año 1.000 en la India y, unos 300 años más tarde, los indios lo distinguieron como el “octavo metal”. Pero no sería aislado hasta mediados del siglo XVIII y no se fabricaría industrialmente hasta principios del XIX. Actualmente se utiliza de un modo muy variado: para proteger el acero mediante el galvanizado o como aislante térmico, para la fabricación de canalones y bajantes pluviales. Su principal uso en la construcción es para la protección de cubiertas en forma de chapas lisas u onduladas, de color gris azulado, y para todas las variantes derivadas de este uso: canalones, cornisas, limahoyas, etc.

Titanio. Es un metal de color gris plata cuyo uso era, hasta hace poco, inhabitual en la construcción, pese a que es uno de los metales más abundantes en la naturaleza, el cuarto (después del aluminio, el

hierro y el calcio, un metal alcalinotérreo). Se extrae principalmente del rutilo, abundante en las arenas de las costas, y se utiliza en forma de aleaciones con el hierro y el aluminio, principalmente. Posee propiedades similares a las del acero pero es mucho más ligero. Tiene un gran resistencia mecánica y una elevada resistencia a la corrosión. También tiene un coeficiente de expansión térmica muy bajo por lo que es muy adecuado para revestimientos. Otra ventaja importante es que sus láminas pueden alcanzar grandes longitudes. Si embargo, su coste era muy superior al de otros materiales más o menos similares, y hasta 1946 no se encontraron métodos para producirlo industrialmente, algo que se logró por el método de Kroll. Pero la extensión de su uso en la industria aeroespacial, debido a su capacidad para soportar temperaturas extremas, y en la industria química por su resistencia a los ácidos, principalmente a partir de la década de 1960, ha hecho que se extendiera a otros sectores, incluido el de la construcción.

Otros metales. Hay varios metales que se utilizan en aleaciones con otros metales principales. Hay varios que ya he citado, como el *estaño*, *molibdeno*, *wolframio*, *vanadio*, *manganeso* y *magnesio*. Otros metales, como el *plomo*, fueron muy utilizados, principalmente para conducciones, pero en la actualidad están prácticamente en desuso debido, entre otras cosas, a que se ha descubierto que resulta tóxico. Los metales preciosos como el oro, la plata o el platino se utilizan raramente en construcción por razones obvias.

Características visuales. Métodos de simulación

La simulación visual de los metales es compleja debido a que intervienen múltiples factores. En primer lugar, las variaciones de color y de textura, que son considerables y abarcan una amplia gama de colores y modalidades locales. En segundo lugar, los metales son muy reflectantes y, por añadidura, lo que reflejan se modifica considerablemente con el ángulo de observación: no solo cambia la intensidad del reflejo, de modo más acusado que con otros materiales, sino también el color. Por añadidura, los metales tienen una inestabilidad superficial, inherente a su estructura atómica, que ya he comentado en el capítulo 1. Una de las consecuencias de esta inestabilidad es que pueden deteriorarse con facilidad debido a la corrosión, que adopta diversas formas. Si el metal no está adecuadamente protegido, su exposición a la atmósfera da lugar a reacciones químicas que absorben oxígeno y el metal "se oxida", lo que se traduce en una serie de manchas características. Este proceso se detiene después de un tiempo pues la capa oxidada actúa de capa protectora y las manchas se estabilizan.

Todos estos factores hacen que sea bastante más difícil dar recomendaciones generales. No queda otro remedio que elaborar una casuística detallada para los diferentes tipos de metales, detalles que se modificarán notoriamente según los casos. Para resumir en la medida de lo posible alguno de estos casos, he elaborado los 12 ejemplos que se muestran en la figura 6.8, aplicando diferentes configuraciones a un objeto, un nudo tórico, que presenta propiedades de anisotropía, que también se dan en muchos metales.

La escena está preparada con un pavimento de base que incorpora un material con una textura de madera y con un mapa de fondo que representa un cielo azulado con nubes, más o menos visible en alguno de los reflejos. La iluminación corre a cargo de una luz principal (*mr Area spot*) situada a la izquierda del lado de la cámara, y dos luces secundarias a la derecha que realzan los reflejos.

En todos los casos se ha utilizado un material *Arch&Design*. Solo indico las configuraciones utilizadas para los casos más reflectantes. En el resto no hay prácticamente reflexión y la simulación depende principalmente de la textura. Los números que siguen se refieren a la figura 6.8, de izquierda a dercha y de arriba abajo:

2) Aluminio lacado. *Diffuse*: 0,3, blanco. *Reflection*: 0,60, 0,35, 16, "metal". *BRDF*: 0,9, 1,0, 5,0 (0º, 90º, *curve shape*). *Anisotropy*: 0,5;

3) Aluminio lacado brillante. Como el anterior pero *Reflection*: 0,60, 0,45, 16, "metal";

4) Cobre rojizo. *Diffuse*: 0,2, color por mapa (textura de cobre). *Reflection*: 0,60, 0,40, 16, "metal". *BRDF*: 0,9, 1,0, 1,65. *Anisotropy*: 0,8;

5) Cobre dorado. Como el anterior con variaciones en el color de la textura;

6) Latón. *Diffuse*: 1,0, color por mapa (textura de latón). *Reflection*: 0,60, 0,40, 16, "metal". *BRDF*:

Figura 6.8 Metales: a*) Aluminio anodizado. Aluminio lacado. Aluminio lacado brillante;* b*) Cobre rojizo. Cobre dorado. Latón;* c*) Hierro. Hierro oxidado. Bronce;* d*) Acero inoxidable. Acero corten. Acero corten.*

0,9, 1,0, 1,65. *Anisotropy*: 1,0;

10) Acero inoxidable. *Diffuse*: 0,8, color por mapa. *Reflection*: 0,50, 0,40, 16, "metal". *BRDF*: 0,9, 1,0, 5,0. *Anisotropy*: 0,5.

6 Hormigón

El hormigón es un material "moderno" en un sentido literal pues comenzó a utilizarse hace poco más de 150 años, a raíz del descubrimiento del cemento Portland en 1824 (por el constructor inglés Joseph Aspdin: el nombre viene de la semejanza con las rocas de la isla de Portland, en el sur de Inglaterra). Sin embargo, el uso de conglomerados materiales se remonta mucho más atrás. Los romanos utilizaban un cemento natural cuyo principal componente eran las cenizas volcánicas provenientes de Pozzuoli, cerca del Vesubio, de donde viene la denominación de cemento puzolánico, utilizando cal como elemento conglomerante. Pero la incorporación del acero para fabricar el hormigón armado fue un paso determinante para convertir el hormigón en el protagonista indiscutible de la construcción moderna. Un paso que se reforzó con el descubrimiento de todo tipo de aditivos que han ido contribuyendo a mejorar su comportamiento y su aspecto.

Buena parte del éxito del hormigón es debida a que sus componentes básicos, áridos, cemento y agua, son abundantes y baratos. Y a que su proceso de fabricación, que puede adaptarse al sitio y a todo tipo de circunstancias, es también relativamente sencillo. El principal inconveniente es que la estabilización se prolonga durante mucho tiempo. Los hormigones corrientes comienzan a fraguar a los 30 o 45 minutos de haberse colocado en los moldes y este proceso continúa durante 10 o 12 horas. Luego comienza el proceso de endurecimiento, a un ritmo relativamente rápido durante los primeros días. A partir del primer mes el proceso continúa muy lentamente pero puede durar hasta un año.

Los *áridos* que componen el hormigón son granos inorgánicos, por lo general grava y arenas de origen natural, aunque también se utilizan áridos artificiales y áridos reciclados, con granulometrías muy variadas pero inferiores a los 100 mm. Es el constituyente principal (entre el 80 % y el 90 %) del peso pues el cemento actúa como aglutinante y el agua se evapora o se integra en los constituyentes básicos. Las características de los áridos son objeto de análisis muy detallados, en los que naturalmente no vamos a entrar pero que pueden encontrarse en cualquier buen manual sobre estos temas.

El *cemento* es un conglomerante hidráulico, aunque también hay cementos no hidráulicos que se endurecen en contacto con el aire, al igual que lo son la cal o el yeso. Es decir, es un material inorgánico, pulvurento y que tiene la propiedad de poder unir fragmentos de otros materiales debido a las transformaciones químicas que experimenta su masa y que dan lugar a otros compuestos cuando se mezcla con el agua. El cemento Portland se obtiene al calcinar mezclas de calizas y arcillas a unos 1.500º. El producto resultante, el clínker, se muele junto con un retardador de fraguado (por lo general piedra de yeso). Hay múltiples variedades que se caracterizan por la presencia de elementos adicionales tales como escoria siderúrgica, humo de sílice, puzolana natural, ceniza silícea, ceniza calcárea o caliza.

El *agua* utilizada para fabricar el hormigón es de dos tipos. La que se denomina "agua de amasado" tiene mayor importancia y debe ser limpia y no presentar impurezas, pues cumple diferentes funciones: hidratar los componentes activos del cemento, permitir que la masa se pueda trabajar o crear huecos en la pasta para los productos resultantes de la hidratación del cemento. El "agua de curado" está en contacto con el hormigón durante poco tiempo y su única función es mantener hidratado el hormigón durante el proceso de fraguado y su primer endurecimiento, por lo que su importancia relativa es menor. El agua de amasado debe mantener una proporción perfectamente regulada pues una cantidad excesiva, al evaporarse, crearía huecos que disminuirían la resistencia del hormigón y una cantidad insuficiente resultaría difícil de trabajar.

Además de estos tres constituyentes principales hay diversos tipos de *aditivos*, compuestos muy variados, que se añaden en pequeñas cantidades para mejorar características concretas del hormigón. Y hormigones especiales que dependen de usos específicos y que se detallan en el apartado sobre tipos.

Características físicas

Las principales características físicas del hormigón derivan de su resistencia a la compresión, la tracción y la flexión. La más elevada de estas tres es la resistencia a compresión, del orden de diez veces superior a la de tracción. La resistencia de un hormigón concreto depende de múltiples factores que son objeto de estudios especializados basados en las características de los componentes principales, de la relación cemento/agua y del propio proceso de fabricación.

Otra característica importante es la permeabilidad que se puede definir como la facilidad de un material para ser atravesado por un fluido, líquido o gaseoso. Y que depende de la porosidad y de las conexiones entre los poros, lo que depende a su vez de las características de los áridos y de los procesos de fabricación, más concretamente, de los procedimientos de compactación.

Durante el fraguado el hormigón experimenta cambios en su volumen debido a procesos de retractación (contracciones debidas al fraguado) y entumecimiento (expansiones debidas a la absorción de agua por el cemento). Esto puede dar lugar a grietas o fisuras visibles y requiere, entre otras cosas, la previsión de juntas de dilatación para absorber estos movimientos.

El hormigón está expuesto al deterioro debido a múltiples causas que van desde la aparición de grietas debidas a movimientos no debidamente absorbidos por la previsión de un diseño adecuado, a la corrosión de las armaduras que aflora a la superficie, a las reacciones químicas con el medio ambiente, a la humedad y a otros factores.

Resumo las propiedades generales como en los casos anteriores.

La ***densidad*** está en torno a los 2.300 kg/m^3. Los denominados hormigones ligeros, en masa, están entre los 1.200 y 2.000 kg/m^3, más corrientemente entre 1.600 y 1.800. Los normales, entre 2.000 y 2.500. El hormigón armado tiene una densidad comprendida aproximadamente entre los 2.300 y los 2.500 kg/m^3. Otros valores de referencia son el mortero de cemento o cal, que está entre los 500 y los 1.800, y los bloques de hormigón, entre los 520 y los 1.300 kg/m^3.

La ***resistencia mecáncia*** a la compresión es la base de clasificación de los hormigones según su resistencia a los 28 días y varía entre 125 y 500 kg/cm^2 para el hormigón ordinario (H-125, H-150… H-450, H-500). Pero hay hormigones que alcanzan hasta 2.000 kg/cm^2. La resistencia a la tracción es del orden de unas siete u ocho veces menor. Pero se incrementa notablemente, como es notorio en el caso del hormigón armado, sacando partido de la resistencia a tracción del hierro. Los morteros de cal y cemento tienen valores comprendidos entre los 5 y los 175 kg/cm^2.

La ***conductividad térmica*** es baja. En W/m.K y de mayor a menor (menor a mayor capacidad aislante): hormigón armado (2,30), hormigón en masa (2,00), hormigón con áridos ligeros (1,30), bloque de hormigón (1,20).

La ***dureza*** del hormigón fresco es muy baja. A los dos días de fraguado está en torno a 2,5 de la escala de Mohs, es decir, entre el yeso (2,0) y la calcita (3,0). A medida que avanza el proceso de curado se va endureciendo pero las propiedades finales dependen considerablemente de los agregados utilizados lo que, a su vez, depende del uso previsto. Si el hormigón se va a utilizar para suelos estos agregados deberán asegurar que los valores de dureza están por encima de 5,0 que es un requisito mínimo habitual para estos elementos. Pero puede alcanzar valores bastante mayores en aplicaciones especiales, siempre en función del tratamiento específico, razón por la que es difícil dar valores generales.

Hormigones más utilizados en la construcción

Hormigón en masa y hormigón armado. El hormigón se utiliza principalmente *in situ*, lo que permite adaptarlo a las características singulares de la obra. Puede utililizarse en masa, sin armaduras de acero, o puede incluir armaduras de acero de dimensiones adecuadamente calculadas para aumentar su resistencia.

Hormigones especiales. Los más importantes son los hormigones *ligeros* (fabricados con áridos especiales de muy baja densidad, como la arlita o perlita o incorporando burbujas a la pasta, como en los hormigones celulares), que se utilizan como aislantes térmicos y en algunos tipos de estructuras; *pesados* (fabricados con áridos pesados, rocas

mineralizadas o fragmentos metálicos), que se utilizan como escudos contra las radiaciones; *refractarios* (fabricados con áridos y cementos especiales resistentes a las temperaturas elevadas), que se utilizan para resistir altas temperaturas; *reforzados* (con fibras de muy diversos tipos, de acero, vidrio, carbón, asbestos, nailon, poliéster…), con mayor resistencia a la tracción, la fatiga o los impactos; *porosos o drenantes* (sin áridos finos para aumentar su permeabilidad), que se utilizan en capas de drenaje de pavimentos o carreteras; *secos* (con mayor dosificación de agua para consolidarlos posteriormente mediante compactación con rodillos), que se utilizan en pavimentos o aplicaciones donde la superficie predomina sobre el espesor; *de alta resistencia* (con baja relación de agua-cemento favorecida por el uso de autoplastificantes); *autocompactables* (mediante el uso de escorias granuladas y superplastificantes a partir de métodos desarrollados en Japón en 1988).

Un tipo de hormigón especial, muy popular desde hace varios años por sus cualidades constructivas y visuales, es el hormigón reforzado con fibra de vidrio o GRC (por sus siglas en inglés, *Glass Reinforced Concrete*) o GFRC (*Glass Fiber Reinforced Concrete*). La incorporación de la fibra de vidrio mejora la respuesta a los esfuerzos de tracción y permite utilizarlo para fabricar paneles de poco grosor que pueden utilizarse como revestimiento en fachadas y cerramientos exteriores. Y, por añadidura, tiene buenas propiedades de aislamiento térmico, resiste bien los ácidos y otros agentes externos potencialmente nocivos y tiene un coste relativamente bajo.

Prefabricados. Hay una gran variedad de elementos que se producen sin relación con el sitio: bovedillas, casetones, conductos de todo tipo, forjados de placas alveolares, viguetas, vigas especiales, pilares, escaleras, paneles, etc.

Dentro de este grupo, por razones de espacio, se pueden incluir también morteros y materiales bituminosos. En ambos casos se trata de productos informes que se adaptan a las bases a las que se aplican y se endurecen con el tiempo. Los ***morteros*** de cal y cemento son mezclas de conglomerantes inorgánicos, agua y áridos de diversos tipos que sirven, desde tiempos inmemoriales, de aglutinante para elementos de construcción muy variados: cerámicas, rocas o bloques de hormigón y para rellenar los espacios que quedan entre elementos. Los ***materiales bituminosos*** son substancias de color negruzco, que se ablandan por el calor. Incluyen los derivados del petróleo y los obtenidos por destilación de substancias de origen carbonoso, utilizados desde el 3.800 aC en Mesopotamia y el valle del Indo. Los dos grupos principales son los betunes y los alquitranes. Los betunes son mezclas de hidrocarburos que se presentan de diversos modos. Los alquitranes son productos que se obtienen por destilación en ausencia de aire. Tanto los betunes asfálticos como los alquitranes se utilizan extensamente en la construcción.

Características visuales. Métodos de simulación

En el caso del hormigón y elementos similares, no hay ninguna dificultad técnica especial más allá de la obtención de texturas adecuadas, lo que tampoco supone un problema debido a que las variantes singulares son pequeñas. De hecho, puede ser más práctico contar con diferentes texturas informes por un lado y con patrones como marcas de encofrado (agujeros, juntas, etc.) por otro. Esto nos permitirá adaptar la distribución a los diferentes casos.

Esto es lo que se ha hecho en la primera imagen de la figura 6.9: un patrón de marcas se ha combinado (por medio de capas superpuestas en Photoshop) a una textura informe de hormigón para que la modulación se adaptase con facilidad a lo que interesa. En la segunda imagen de esta misma figura se ha utilizado una textura de encofrado horizontal modificando su tono con mapas correctores de color.

7 Plásticos

A diferencia de todos los materiales resumidos en los capítulos anteriores, con la excepción del hormigón, los plásticos se utilizan en la construcción de un modo relativamente limitado y desde hace pocos años. Sin embargo, su uso se ha ido expandiendo con gran rapidez, con un crecimiento de un 5 % anual en los últimos años, y es muy probable que este ritmo se mantenga o aumente en el futuro

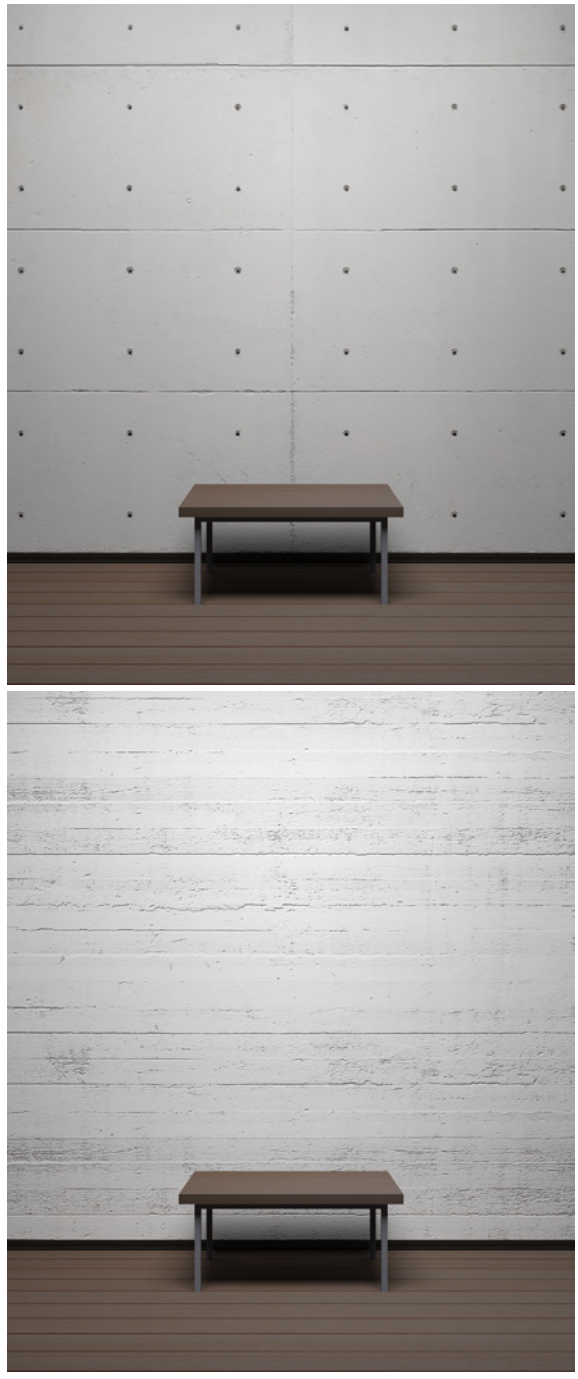


Figura 6.9 Hormigón. Muro con dos tipos de tratamiento de hormigón visto.

inmediato. En 2010 se produjeron 265 millones de toneladas en todo el mundo, de los cuales 57 correspondían a la UE27, la segunda del mundo, con un 21,5 % del total mundial (la primera fue China, con un 23,5 %).

Esta expansión ha afectado principalmente a las industrias de embalajes y envases, con una cuota del 39 % del total producido en la UE en 2010, seguidas por un 20,6 % en construcción, 7,5 % en automoción y 5,6 % en equipos eléctricos y electrónicos. En España las cifras fueron similares aunque con mayor porcentaje de las primeras (algo más de un 45 %) y menos de las segundas (algo menos de un 15 %).

La palabra *plástico* viene del griego *plastikos* que significa "capaz de ser moldeado o adquirir forma". Este sentido original se mantiene pues un material plástico, que pertenece a un grupo muy amplio de combinaciones de carbono con otros elementos (oxígeno, nitrógeno, hidrógeno y otros componentes orgánicos o inorgánicos), se caracteriza principalmente porque en algunas fases del proceso de fabricación se da en estado líquido y puede ser moldeado de diversos modos antes de pasar al estado sólido. Esta versatilidad es una de sus principales ventajas y la que permite que se adapte a todo tipo de aplicaciones.

A lo largo del siglo XIX aparecieron algunos de los principales antecesores de los plásticos modernos, productos parcialmente sintéticos pero que se conseguían utilizando materias primas orgánicas, como el *caucho* (obtenido casualmente en 1839, por vulcanización del caucho natural con azufre), la *ebonita* (un caucho endurecido, obtenida en 1851 añadiendo azufre en altas concentraciones a caucho puro), el *celuloide* (disolviendo celulosa, un material de origen natural, en una solución de alcanfor y etanol, inicialmente denominado *parkesita* por su inventor, el inglés Alexander Parker, que se olvidó de patentar su invento, y mejorado por Hyatt que lo patentó en 1870, con el nombre actual, ganando un premio ofrecido por un fabricante de bolas de billar). Otros productos similares que también se pueden citar, aunque no propiamente plásticos en el sentido general que estamos tratando, son el *linóleo* (1844, no propiamente un plástico sino aceite de linóleo solidificado con aditivos varios y utilizado para fabricar suelos baratos hasta la década de 1950 en que se le encontraron mejores substitutos) o la *caseína* (en 1897, al mezclar leche agria con formaldehído).

Por otro lado, también durante este periodo, se descubrieron muchos materiales que actualmente son fundamentales en la fabricación de los plásticos. Pero sus inventores o descubridores no estaban en situación de apreciar la importancia industrial de lo que habían encontrado y no se explotaron hasta bien entrado el siglo XX. Así ocurrió con el *estireno* (1831), la *melanina* (1834), el *cloruro de vinilo* (1835), el *poliestireno* (1845), el *poliéster* (1847), el *cloruro de polivinilo* (1872), el *metacrilato* (1880), la *urea-formaldehído* (1884), el *acetato de celulosa* (1894), similar al celuloide pero menos inflamable, el *polietileno* (1898) o el *policarbonato*, descubierto en 1898 pero que no se pondría en el mercado hasta 1959 y no se utilizaría a gran escala en la construcción hasta hace pocos años.

Los primeros plásticos sintéticos propiamente dichos se comercializaron en 1909, a partir del descubrimiento de la *baquelita*, el primer plástico sintético termoestable. El nombre deriva de su descubridor, Leo Baekeland (1863-1944), que la descubrió casualmente al intentar resolver un problema de síntesis química. Con este nuevo material se fabricarían artículos de escritorio, ceniceros o teléfonos, generalmente de color negro brillante, productos que todavía nos resultan familiares por el cine de la década de 1920.

En 1915 se descubrió la formación de polímeros por el encadenamiento de dos o más monómeros de distinto tipo, lo que se denominó *copolimerización*, que amplió considerablemente la variedad de plásticos y abrió la posibilidad de diseñarlos en función de finalidades predeterminadas, una posibilidad que ha ido adquiriendo cada vez más importancia. En 1928 se desarrollan las *poliamidas*, que se comercializarán principalmente con el nombre de nailon (*nylon*).

En la década de 1930 se desarrollan industrialmente muchos de los polímeros más utilizados actualmente: el *policloruro de vinilo*, el *poliestireno* o el *polimetacrilato* de *metilo* (lanzado al mercado en 1936 con el nombre de *plexiglás* y que en la Segunda Guerra Mundial se utilizaría para las ventanas de aviones). Y en esa misma década na-

ció la técnica de los termoplásticos. El *poliestireno* se produjo industrialmente en 1930, en la fábrica badense BASF, que es actualmente la empresa química más grande del mundo. La tendencia del *estireno* a endurecerse ya se había detectado desde su descubrimiento en 1831 pero no se había sacado partido de ello hasta que Hermann Staudinger lo sintetizó en laboratorio y lo explicó mediante una polémica "teoría de la polimerización" (hacia 1920, que fue rechazada por sus colegas pero que le valdría el Nobel de Química en 1953). En 1933 se descubrió por accidente, en Inglaterra, otra variante del *polietileno* (que ya se había descubierto también por accidente en 1898). Un par de años después se desarrolló este descubrimiento y se comenzó a producir industrialmente (en 1939), aunque la producción se detuvo por la Guerra y no se reanudaría, a gran escala, hasta la postguerra, principalmente a partir de finales de la década de 1950 y con aportaciones cruciales de Karl Ziegler, entre otros. En 1937, Otto Bayer y sus colaboradores sintetizaron el *poliuretano*, que se utilizó inicialmente para la producción de fibras y colchones flexibles y, posteriormente en recubrimiento de aviones durante la Segunda Guerra Mundial. En 1938 se descubrió casualmente el *politetrafluoretileno*, comercializado con el nombre de *teflón*, capaz de soportar temperaturas de hasta 300º.

En la década de 1940 siguen apareciendo nuevos productos. En 1941 se patentó en Inglaterra (a partir de investigaciones previas desarrolladas en Estados Unidos hacia 1929 por DuPont, que las abandonó para concentrarse en las poliamidas y el nailon) el *polyester*, una variante (PET, *PolyEthylene Terephthalate*) del producto ya descubierto en 1847, a partir del cual se fabricaron fibras sintéticas y también pinturas y barnices y, más recientemente, botellas de plástico. También por estas fechas se dispara, principalmente en el Reino Unido, la investigación y la experimentación con plásticos, principalmente poliéster, reforzado con fibra de vidrio o GRP por sus siglas en inglés (*Glass Reinforced Polyester*).

En la década de 1950 se descubre el *polipropileno* (por varios equipos de investigadores entre los que figuran en lugar destacado el alemán Karl Ziegler, en 1953, y el italiano Giulio Natta, en 1954). Este producto no se utilizaría mucho hasta mediados de la década de 1980, pero a partir de esas fechas su crecimiento se ha ido disparando. En 1951 los laboratorios Bask alemanes hallan el modo de producir espuma rígida calentando poliestireno dentro de un horno con agentes especiales y patentan el *poliestireno expandible*.

A mediados del siglo xx la utilización de los plásticos se había extendido a varios sectores industriales, principalmente de electrotécnica, aviación y automoción. En 1955 se construyó la primera casa edificada completamente con materiales plásticos, diseñada por un equipo de diferentes profesionales (principalmente los arquitectos Ionel Schein y René-André Couloun y el ingeniero Yves Magnant), presentada en la exposición de París de 1955 y que utilizó principalmente paneles de GFRP (*Glass Fiber Reinforced Plastic*). Otra referencia más notable fue la famosa "casa del futuro", diseñada por Alison y Peter Smithson y presentada en la exposición del "hogar ideal", en Londres. A estas experiencias les siguieron un buen número de otras similares en las que el plástico se fue incorporando parcial o totalmente al diseño. A partir de la década de 1960 se comenzó a utilizar cada vez más en la construcción. Un hito importante fue la primera IKA (*International Plastic Housing Exhibition*), en 1971, en Lüdensheid, Alemania, que en sucesivas ediciones mostraría diversos prototipos de viviendas construidas con plástico.

Por estas fechas comenzó a modificarse el desarrollo en una dirección extraordinariamente significativa, pues los primeros plásticos se desarrollaron a partir del descubrimiento de un nuevo material, de la observación de sus propiedades y de la búsqueda de aplicaciones comerciales. Pero ahora el proceso comenzó a invertirse en un giro sin precedentes. Se enunciaban las propiedades deseadas y se buscaba cómo sintetizar un material que incluyese estas propiedades. A partir de la década de 1970 esta tendencia se ha intensificado y junto a los nombres más conocidos aparecen periódicamente nuevas variantes. La obra de Engelsmann, Spalding y Peters (2010), citada en las referencias, presenta un buen panorama reciente de estos desarrollos para ampliar esta breve introducción.

Características físicas

Los plásticos son fundamentalmente estructuras macromoleculares, polímeros formados por la agregación de monómeros cuyo componente principal es el carbono. El carbono tiene cuatro valencias distribuidas simétricamente a las que pueden acoplarse otros átomos. La facilidad del carbono para combinarse con otros elementos es la base de la variedad de monómeros que entran como constituyentes básicos de los polímeros. A partir de esta descripción básica, intentar clasificar los plásticos de un modo coherente pero no excesivamente complicado es una tarea difícil y pesada. Las clasificaciones más simples sirven de muy poco para ordenar los tipos que se utilizan en la práctica constructiva, que es la referencia que nos importa. Y las clasificaciones sistemáticas son excesivamente complejas y requieren descripciones químicas. Por obvias razones de espacio me referiré muy brevemente a las clasificaciones más simples y añadiré alguna indicación sobre las sistemáticas.

Hay una primera distinción que cabe hacer aunque no sea demasiado importante, pues aunque cuando hablamos de plásticos nos referimos generalmente a los plásticos sintéticos, también se habla de plásticos "naturales" y "artificiales". Los plásticos naturales son también polímeros que se forman de modo espontáneo, sus propiedades son conocidas desde tiempos inmemoriales y son muy similares a las de las resinas y acrílicos artificiales. Entre estos polímeros naturales se pueden citar el *ámbar* (una resina de coníferas), la *goma laca* (secreciones de la hembra de chinche, el *lac*, originario de la India) o la *gutapercha* (una goma vegetal similar al caucho extraida de ciertos árboles de la India y de Indonesia).

Una segunda agrupación, también muy simple pero más relevante, es la que agrupa a los polímeros en tres grandes grupos: *termoplásticos*, constituidos por cadenas lineales flexibles; *termoestables*, por redes tridimensionales rígidas; *elastómeros*, por cadenas lineales con enlaces cruzados. Los dos primeros grupos permiten dividir a los plásticos utilizados en la construcción en grupos con propiedades claramente diferenciadas a las que ahora volveré. Los elastómeros se han utilizado a partir de la Segunda Guerra Mundial como substitutivos del caucho natural. El más conocido y utilizado es el *neopreno*, pero no se utilizan en construcción.

Los ***termoplásticos*** se caracterizan porque, al calentarse, sus cadenas lineales y flexibles se deslizan con facilidad entre sí, lo que se traduce en un ablandamiento que permite su moldeo y una recuperación posterior, al enfriarse, que se traduce en un endurecimiento que facilita su utilización práctica. Ejemplos de este grupo son los diferentes tipos de polietileno, el polpropileno, el PVC (cloruro de polivinilo), el poliestireno, los acrílicos, el nailon (poliamidas), el acetato de celulosa, el PTFE (politetrafluoroetileno), el policarbonato… Los ***termoestables*** (también denominados durómeros o duroplastos) se caracterizan por una mayor estabilidad debida a su estructura tridimensional muy firme que impide el deslizamiento y la deformación de sus componentes. Como lo indica su nombre, mantienen su forma aunque aumente la temperatura. Los termoestables se fabrican con una forma definitiva y si se calientan se degradan o se descomponen. Ejemplos de este grupo son el urea-formaldehído, la melamina, las resinas epoxi o las resinas de poliéster.

Un tercer criterio de interés es ordenar los plásticos por su volumen de utilización. El número de familias de plástico que se comercializan es muy grande pero, desde el punto de vista de su uso, puede decirse que hay siete grupos principales. Los cinco primeros, todos ellos termoplásticos, abarcaban el 74 % del consumo anual en Europa en 2010 y son también los más utilizados en construcción. Estos son, por orden de mayor a menor consumo en Europa: *a*) *polietileno* (PE), con dos grupos principales, polietileno de baja densidad (PEBD) y polietileno de alta densidad (PEAD). Su cuota es del orden del 29 %; *b*) *polipropileno* (PP), con una cuota de alrededor del 19 %; *c*) *policloruro de vinilo* (PVC), alrededor del 12 %; *d*) *poliestireno* sólido (PS) y expandido (PSE), alrededor del 8 %; e) *polietileno tereftalato* o *tereftalato de polietileno* (PET), alrededor del 6 %.

Por último, hay que citar, aunque sea muy brevemente, otras clasificaciones más sistemáticas que agrupan a los plásticos a partir de su composición química. Así, dentro de los termoplásticos se distinguen cinco familias principales:

1) *poliofelinas* (PE polietileno, PP polipropileno, PB polibuteno…), que incluyen a los plásticos más corrientes y económicos; 2) *polimerizados de estireno* (PS poliestireno, ABS, ASA); 3) *homopolímeros* (PVC, PTFE…); 4) *ésteres de polivinilo* y *poliacrilo* (acetato de polivinilo, PMMA polimetilmetacrilato, vidrios acrílicos); 5) *heteropolímeros* (acetato de celulosa, policarbonato, poliamidas…). Y dentro de los termoestables y elastómeros, se distinguen otras cinco más: 6) *fenoplastos* (resinas de fenol-formaldehído); 7) *aminoplastos* (resinas de rea-formaldehído, resinas de melamina formaldehído); 8) *resinas reactivas* (poliésters no saturadas, resinas epoxi); 9) *poliuretanos* principalmente; 10) *siliconas*.

En fin, otros sistemas de clasificación se limitan a distinguir los plásticos a partir del monómero del que provienen. En algunos casos, esto coincide con la denominación más corriente, como el polietileno (basado en el etileno) o el poliestireno (basado en el estireno).

§ § §

A la estructura fundamental de los plásticos, macromoléculas cuyo principal componente es el carbono y que agrupan a monómeros unidos mediante un proceso químico, se añaden ingredientes que refuerzan determinadas propiedades. Las tres grandes categorías de ingredientes que se encuentran en los plásticos utilizados en construcción son: *a*) *aditivos*. Son substancias químicas que refuerzan propiedades muy específicas: antioxidantes, estabilizantes térmicos, lubricantes, conservantes, colorantes, retardadores de llama, estabilizantes de rayos ultravioleta, etc. (hay más de 4.000 tipos de aditivos en el mercado). Un plástico corriente, como el PVC, puede incluir hasta un 55 % en peso, de aditivos; *b*) *refuerzos*. Son componentes que se añaden a los polímeros para mejorar su solidez, su resistencia al impacto o su rigidez. Hay dos grandes grupos de refuerzos, en estratos y fibrosos. El material más utilizado como refuerzo es la fibra de vidrio, que ha adquirido una importancia creciente en los últimos años hasta llegar a substituir al metal en determinadas aplicaciones; *c*) *cargas*. Son materiales inertes que se añaden a los plásticos para modificar su resistencia o su comportamiento físico y también para reducir el coste. Pueden ser orgánicos o inorgánicos y su forma (esferas, láminas, fibras irregulares) y tamaño influyen directamente en las características finales.

De toda esta variedad de componentes fundamentales y complementarios resulta un conjunto de propiedades dificiles de resumir, pero podemos intentar destacar las principales.

Además de la facilidad de moldeo y, como consecuencia, de las múltiples formas en que pueden fabricarse (espumas, paneles, tubos, formas de todo tipo) y manipularse, los plásticos tienen otras características muy adecuadas para ciertos usos, principalmente su ligereza, su resistencia a la corrosión, su impermeabilidad térmica, acústica y eléctrica, y que requieren muy poco mantenimiento.

Pero también tienen inconvenientes. Los principales, como ocurre a menudo, están relacionados directamente con sus ventajas pues uno de estos inconvientes es que se deforman con facilidad. Y, por añadidura, pueden reblandecerse y dilatarse, y son inflamables. Otro inconveniente es que algunos se decoloran con el tiempo, como ocurre con el PVC, si bien hay técnicas recientes que están reduciendo este tipo de desventajas.

Por último, y para facilitar la comparación con los materiales anteriores, resumo algunas de las características físicas más generales.

La ***densidad*** está en torno a los 1.000 kg/m^3 con los siguientes valores de referencia de menor a mayor: polietileno (920-980), poliestireno, PS (1050), linóleo (1.200), policarbonato (1.200), polipropileno, PP, con un 25% de fibra de vidrio (1.200), poliuretano, PU (1.200), PVC (1.350).

La ***conductividad térmica*** es baja por lo que en general son buenos aislantes. De mayor a menor (menor a mayor capacidad aislante): polietileno de alta densidad (0,50), polietileno de baja densidad (0,33), poliuretano (0,25), polipropileno (0,22), policarbonato (0,20), PVC (0.18), PMMA, polimetilmetacrilato (0,18), poliestireno (0,16).

Los valores de ***resistencia mecánica*** de los plásticos son muy variables como sería de esperar dada la gran variedad de tipos. Algunos valores de referencia muy aproximados para la resistencia a compresión, que pueden servir para situarlos con respecto a otros materiales, son los siguientes: el polietileno estaría en torno a los 200 (baja densidad) y los 400 (alta densidad) kg/cm^2, con un módulo de

elasticidad del orden de los 8.000 kg/cm^2; el PVC rígido, entre los 500 y los 900 kg/cm^2, con un módulo de elasticidad en torno a los 30.000 o 40.000; el policarbonato puede tener valores de resistencia a compresión de entre 600 a 800 y su módulo de elasticidad está en torno a los 23.000 kg/cm^2; los plásticos reforzados pueden alcanzar resistencias del orden de los 1.500 y hasta los 5.000 kg/cm^2.

Además de la resistencia mecánica, los plásticos son, por lo general, resistentes a agentes químicos y corrosivos, si bien el PE puede ser atacado por algunos compuestos alcalinos y disolventes orgánicos, el PVC es sensible a algunos ácidos y disolventes orgánicos y el PS es soluble en disolventes orgánicos y aceites. Tienen buena resistencia al agua marina y las aguas residuales. Muchos de ellos se degradadan por la exposición al sol, debido a que los rayos ultravioletas provocan oxidaciones y escinden las cadenas poliméricas lo que se traduce en oscurecimiento, pérdida de flexibilidad y grietas. El PVC expuesto al exterior se decolora considerablemente a lo largo de los años aunque esta exposición no afecta a su resistencia, ni mecánica ni química. Su resistencia al fuego es generalmente baja aunque variable según el tipo. Los poliuretanos desprenden substancias tóxicas al quemarse. El PE y el PP arden con facilidad. El PVC no propaga la llama pero también puede desprender gases tóxicos. El PS expandido arde con facilidad y desprende un humo relativamente tóxico.

Plásticos más utilizados en la construcción

Desde el punto de vista del volumen total de producción, los plásticos más utilizados en construcción son los siguientes (entre paréntesis doy los porcentajes del total en miles de toneladas, redondeados, para España, en 2000, según datos de ANAIP): PVC (policloruro de vinilo, 61 %), PE (politetileno, 14 %), PS (poliestireno, 8 %), PU (poliuretano, 8 %), PP (polipropileno, 1 %).

Desde el punto de vista de sus aplicaciones, la principal, con gran diferencia, es su uso para tuberías (más del 60 %) seguido por las ventanas y persianas de PVC (9 %), espumas rígidas de Poliuretano (8 %), sanitarios, depósitos, piscinas, etc. con Poliéster no saturado (6 %) y aislamiento de diversos tipos con PS (5 %).

El ***PVC*** (*polivynil chloride*) es menos inflamable que otros plásticos debido a la presencia de cloro que también influye en otras propiedades notables, como su resistencia mecánica y su resistencia a agentes químicos y atmosféricos. Es el más utilizado en la construcción debido a su bajo coste y a su buen comportamiento general, aunque últimamente se ha puesto en cuestión debido a los problemas ecológicos que plantea pues su combustión emite gases muy contamientantes. También tiende a decolorarse debido a la luz ultravioleta, como ya he dicho, aunque este inconveniente puede reducirse mediante aditivos. Se comercializa de cuatro modos principales: rígido, flexible, en espuma y en pasta. Las diferencias dependen principalmente de los aditivos incorporados. El PVC rígido se utiliza para la fabricación de marcos de ventana, persianas, tuberías de todo tipo, cisternas y contenedores. Comenzó a utilizarse en Alemania en la década de 1950 y su uso para esta aplicación se ha extendido considerablemente desde entonces. El PVC flexible se utiliza para suelos de vinilo, en aplicaciones domésticas, comerciales e industriales. El PVC en espuma se utiliza en piezas de revestimiento o en canaletas.

El ***politetileno*** (PE) es uno de los termoplásticos más simples. Se producen dos tipos, polietileno de alta densidad, más cristalino y en consecuencia más resistente y más rígido que el otro tipo, de baja densidad. Ambos tipos tienen una densidad muy baja, inferior a los 1.000 kg/m^3 y, por tanto, flotan en el agua. Son sensibles a la radiación ultravioleta por lo que suelen incorporar aditivos protectores (carbono negro). Por esta razón, aunque su aspecto natural es incoloro y translúcido el producto comercial suele ser de color negro opaco. Se utiliza principalmente en tuberías y en láminas para suelos de distintos tipos y también como barrera hidrófuga en suelos o como protector de materiales en distintas aplicaciones, provisionales o permanentes. También se utiliza para sistemas de ventilación y calefacción subterránea.

El ***poliestireno*** (PS) es similar al polietileno pero contiene cadenas de benzeno, lo que hace que sea más rígido pero más quebradizo. Es muy inflamable por lo que a menudo incorpora aditivos

que reducen este riesgo. Se utiliza en cuatro variantes principales: el PS cristal (producto puro de la polimerización del estireno, GPPS, *General Purpose Polystirene*), transparente, rígido, quebradizo, que se utiliza para productos tales como vasos de plástico; el PS de alto impacto (por adición de hasta un 14 % de caucho para mejorar la resistencia mecánica, HIPS, *High Impact Polystirene*), resistente y opaco, que se utiliza principalmente para la fabricación de envases pero también para carcasas de contenedores o puertas de frigoríficos; el PS expandido (por adición de un 5 % de gas, generalmente pentano EPS, *Expanded Polystirene*), muy ligero y que se utiliza principalmente como aislante y para el embalaje de productos frágiles. En construcción se utiliza extensamente para el aislamiento de paredes, pisos y techos; el PS extrusionado (mediante inyección de gas, XPS*, Extruded Poystirene*), similar al anterior pero con burbujas cerradas, lo que lo hace más denso e impermeable, y que también se utiliza principalmente como aislante.

El ***polipropileno*** (PP) es similar al polietileno pero incluye un grupo metílico en lugar de un átomo de hidrógeno lo que le confiere mayor resistencia y estabilidad, con un punto de fusión también más alto. Se utiliza en casetones para forjados sanitarios, en tuberías sometidas a altas temperaturas, para sistemas de ventilación, en planchas aislantes, en paneles solares y en espumas basadas en plásticos. También se ha utilizado en la fabricación de sillas y muebles.

El ***poliuretano*** (PUR) se utiliza, principalmente en forma de espuma, para aislar y sellar puertas, ventanas, muros, etc.

Las resinas de poliéster se utilizan generalmente en combinación con fibras de vidrio o fibras de vidrio para formar el ***poliéster reforzado (***GRP, *Glass Reinforced Polyester* y GF-UP, *Glass Fibre reinforced Unsaturated Polyester*). Es un material ligero, muy duro y resistente, que se está utilizando, desde las décadas de 1960 y 1970, para construir estructuras ligeras, placas para cubiertas o depósitos, en paneles de revestimiento exteriores y en múltiples aplicaciones de construcción.

El ***metacrilato*** (PMMA, polimetil metacrilato), con una transparencia del 92% y con muy buena resistencia a la rotura (del orden de siete veces superior al vidrio del mismo espesor) y a los agentes atmosféricos, se ha utilizado ampliamente como alternativa al vidrio para paneles y ventanas, por su menor peso, mejor transparencia y menor fragilidad, aunque se raya con facilidad y su calidad visual es inferior.

El ***policarbonato*** (PC) es un plástico flexible, muy transparente, utilizado desde hace pocos años en planchas lisas para construir lucernarios, ventanas, etc., o en planchas alveolares embutidas en placas de revestimiento de fachadas. Se utiliza a menudo como substituto del vidrio debido a dos ventajas importantes: es mucho más ligero (su densidad es del orden de la mitad de la del vidrio) y mucho más resistente al impacto (más de 200 veces superior). Se suministra con diferentes grados de transparencia. Tiende a amarillear con el tiempo por lo que requiere un tratamiento adecuado.

El ***Polietileno tereftalato*** (PET), resulta familiar pues es el material más utilizado para las botellas de plástico transparente que se usan en todo tipo de líquidos por sus buenas cualidades de transparencia y resistencia. Pero también se han construido recientemente casas de plástico con muros, tabiques y cubiertas hechos con PET reciclado.

El ***ETFE*** (etileno tetrafluoro etileno) es otro producto que se ha comenzado a utilizar recientemente, desde la década de 1980 aproximadamente (aunque la patente se remonta a la de los 1940, por du Pont) como variante mejorada de otro plástico, el ***PTFE*** (politetrafluoroetileno), mucho menos transparente aunque con buenas propiedades de resistencia a los agentes químicos y atmosféricos. La adición de etileno para formar el ETFE mejora sus características de resistencia y transparencia y se ha utilizado para cubrir grandes estructuras. Uno de los pioneros en su uso fue Vector Foilec, un diseñador y fabricante de yates que adaptó su experiencia en este campo a la arquitectura, diseñando un zoo en Arnhem, Holanda, en 1982.

Características visuales. Métodos de simulación

Como ya hemos visto en el capítulo 2, las primeras simulaciones de materiales con cierto grado de realismo se llevaron a cabo hacia 1975 gracias

a los algoritmos desarrollados por Phong y Blinn, principalmente. Tras este éxito inicial, las primeras críticas y los subsecuentes intentos de superación por algoritmos más elaborados iban dirigidas invariablemente al carácter "plástico" de estas primeras simulaciones. Pues, efectivamente, los plásticos reflejan la luz de un modo plano, sin matices, sin variaciones apreciables debidas a los diferentes ángulos de observación, como ocurre con los metales o los materiales orgánicos cuya estructura superficial más compleja crea estos efectos que llevaron a incorporar las reflexiones Fresnel y las curvas BRDF más complejas a los métodos de simulación de materiales.

Pero lo que esta crítica implica, por otro lado, es que la simulación de materiales plásticos es muy simple, pues no tenemos que preocuparnos ni tan siquiera de utilizar materiales avanzados. Un material *Standard* con controles para ajustar el color, la intensidad de la reflectividad y de la transparencia será suficiente para recrear los efectos característicos de los materiales plásticos.

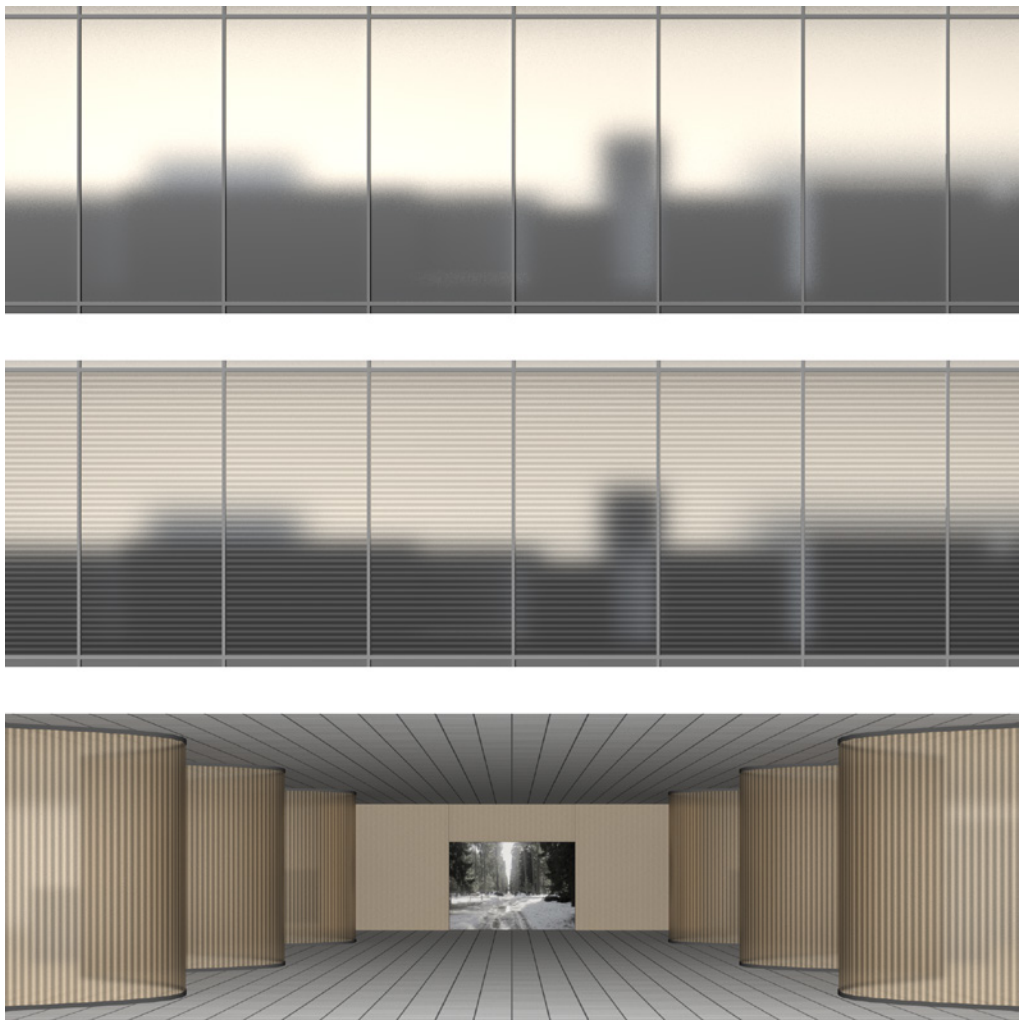


Figura 6.10 Plásticos. Diferentes tipos de paneles translúcidos.

Figura 6.11 Plásticos. Paneles translúcidos de diferentes colores.

En el caso de plásticos transparentes, los recursos proporcionados por un material avanzado para variar tanto la intensidad de la transparencia como su dispersión (*glossiness*), son suficientes, junto con la variación de color, para crear los efectos característicos que no son muy diferentes de los de los de los vidrios. El primer grupo de imágenes de la figura 6.10 ilustra este caso. El efecto de un panel translúcido de metacrilato, ETFE u otro plástico similar, puede simularse fácilmente ajustando los parámetros de transparencia de un material *Arch&Design* que, en la primera imagen de dicha figura, se han dejado en 0,65, 0,40, 24 (*transparency*, *glossiness*, *samples*). La adición de un mapa de degradado al color local es suficiente, en general, para simular el efecto de, por ejemplo, paneles de policarbonato (segunda imagen de la figura 6.10) o de paneles de plástico corrugado (tercera imagen de la figura 6.10).

Con valores similares pero modificando el color se pueden simular casos interesantes en los que se utilizan plásticos traslúcidos para crear gamas de color en ventanales, paneles o montajes de tipos diversos. La figura 6.11 muestra un ejemplo de algunas de las miles de combinaciones que pueden crearse con poco más que variando estos efectos elementales.

Hay que tener en cuenta que, aunque en las secciones anteriores he presentado diferentes tipos de materiales plásticos utilizados en arquitectura, desde el punto de vista de la simulación visual no hay diferencias importantes pues las diferencias principales tienen que ver con su comportamiento físico. Si el polietileno se utiliza con frecuencia para tuberías de color negro no es porque su aspecto sea diferente al de otros plásticos, como ya he remarcado antes, pues es incoloro y translúcido como muchos otros. Pero se le añaden aditivos que lo oscurecen y lo protegen pues es sensible a la radiación ultravioleta.

Otro tanto ocurre con muchos otros plásticos, cuyo color no depende de otra cosa que de los aditivos que se le añaden para colorearlo artificialmente. En el caso de los plásticos utilizados corrientemente como substitución del vidrio, la única diferencia remarcable es que la transparencia es menos pura y algo más dispersa, un efecto que ya hemos visto que se puede controlar con facilidad.

En el caso de grandes paneles, como los que a veces se construyen con GRP habría que tener en cuenta otros factores. Un objeto de pequeñas dimensiones puede simularse fácilmente, como ya he dicho, sin más que variar su color y su mayor o menor reflectancia. Sin embargo, en el caso de paneles de grandes dimensiones y formas complejas, hay que tener en cuenta dos cosas más. En primer lugar, debido al proceso de fabricación, aparecen irregularidades características que, si no se incor-

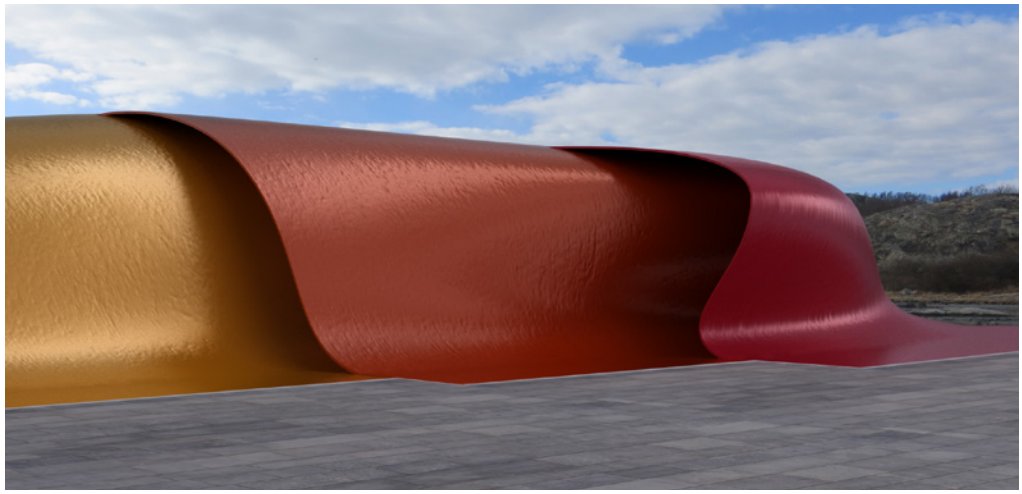


Figura 6.12 Plásticos. Paneles de plásticos reforzado con fibra de vidrio (GRP).

poran a la simulación, darían resultados excesivamente perfectos y artificiosos. Para incorporar este efecto, en la tercera figura de este grupo, la 6.12, he incluido un mapa de relieve (*Bump*) con una textura informe. En segundo lugar, como ya he recalcado en el capítulo anterior, la simulación de reflejos implica que haya algo que reflejar. Esto es un problema que se da exactamente igual en la realidad, razón por la que los fotógrafos que quieren obtener una imagen convincente de un objeto reflectante dedican mucho tiempo a organizar la escena que rodea al objeto y que no será visible en la fotografía. En un caso como el de la figura 6.12 esto se ha resuelto de dos modos. En primer lugar, añadiendo el mismo mapa de fondo a la entrada de mapas especiales del material *Arch&Design* (sección *Special Maps* / *Environment*) para prevenir errores que ya he comentado en el capítulo anterior, en la sección sobre Reflejos. Y, en segundo lugar, añadiendo a la escena unos objetos elipsoidales de color blanco (no visibles en la image), situados de tal forma que refuercen el reflejo de determinadas zonas de los paneles que se trata de simular.

8 Varios

En marzo de 2014 se concedió el premio Pritzker de arquitectura al japonés Shigeru Ban. Este premio ha supuesto una inflexión respecto a las tendencias imperantes en versiones anteriores, pues no se premiaba al autor de edificios singulares construidos para empresas o instituciones, sino a un arquitecto que ha destacado por su compromiso con proyectos donde, en palabras del jurado, "la sostenibilidad no es un concepto, sino un hecho, algo intrínseco". Shigeru Ban se ha hecho mundialmente conocido por sus intervenciones en regiones desvastadas, utilizando materiales locales simples, materiales alternativos o materiales de deshecho.

Pero también implicaba algo más, pues los edificios singulares que realzan la importancia social de instituciones públicas o privadas se construyen tradicionalmente con materiales nobles que estén a la altura de la significación de la obra. Pero los materiales utilizados en las obras de Shigeru Ban son todo lo contrario: materiales de muy bajo coste, cotidianos, fáciles de conseguir y de trabajar.

El premio a Shigeru Ban no supuso realmente un giro brusco en las tendencias citadas pues en 2012 se había concedido a Wang Shu, un arquitecto y escritor chino que se dio a conocer en la Xª Bienal de Venecia, en donde presentó un "Jardín de Baldosas", un marco de bambú que encerraba 66.000 azulejos chinos reciclados. Ha investigado en la arquitectura vernácula china

y también ha experimentado con materiales alternativos de bajo coste, como en el Museo de Historia de Ningbo, una obra en la que algunos de los muros están decorados con miles de azulejos reciclados, recogidos de los alrededores de la construcción.

Como ocurre a menudo, estas innovaciones radicales hunden sus raíces en antiguas tradiciones. Son originales en un sentido profundo, revitalizan los orígenes olvidados de la disciplina, pues las construcciones más primitivas que conocemos, las chozas de tribus africanas, las tiendas de tribus nómadas, las viviendas de todo tipo de poblados dispersos por todo el planeta, no utilizan piedras, ni piezas cerámicas cocidas, ni maderas cortadas cuidadosamente, sino tierra, paja, telas, cuerdas, materiales de deshecho reutilizados.

También hay que subrayar que muchos de los materiales y métodos de construcción que se recogen en este apartado son, de hecho, los que siguen siendo más utilizados en la actualidad en la arquitectura real, si entendemos por arquitectura cualquier forma de construcción de un espacio privado o social. Al igual que se pierde de vista que tan solo un ínfimo porcentaje de las viviendas que se construyen en el mundo son proyectadas por arquitectos y construidas bajo su supervisión, también se tiende a perder de vista que la gran mayoría de estas viviendas se construyen con materiales que no se inscriben en los procesos de producción de los materiales que se han resumido en las siete secciones anteriores.

Para no alargar demasiado la exposición he organizado esta última sección en tres apartados: tierra combinada con otros materiales; materiales orgánicos como el bambú, la hierba o la paja; y materiales corrientes como el cartón, las telas o los plásticos reciclados. El primer apartado está claramente relacionado con la sección 3, sobre cerámica. Pero la diferencia es que dicha sección se refiere a materiales obtenidos mediante procesos industriales complejos que requieren la inversión previa en medios de producción complejos. Otro tanto cabría decir de la madera o los plásticos, que también aparecen en los apartados que siguen, pero que se diferencian de los productos reseñados en las secciones 2 y 7, respectivamente, por las mismas razones.

Tierra combinada con otros materiales

La construcción de edificios sencillos con adobe y tierra es tan antigua como la historia de las ciudades y sigue siendo un método de construcción habitual en muchos lugares del mundo, principalmente en África, hasta tal punto que se estima que la mitad de la población mundial vive en casas de tierra. Las ventajas son indudables: la tierra es barata, fácil de obtener y de manejar, y proporciona un buen aislamiento acústico y térmico. Las principales diferencias vienen de los componentes principales. Las tierras graníticas dan buenos resultados pero las basálticas son más difíciles de compactar. Las basadas en bentonitas (una archilla muy fina cuyo nombre deriva de los yacimientos en Fort Benton, en Estados Unidos) no funcionan bien debido a la retracción causada por las arcillas que contienen. Las tierras calizas necesitan adiciones de arcilla, cemento o cal.

Aparte de estas diferencias de los componentes, otra diferencia importante está relacionada con los métodos de construcción que han dado lugar a diferentes denominaciones: adobe, cob, tapial, bajareque o tierra compactada, entre otros.

El ***adobe*** es una mezcla de barro y paja que se moldea para crear piezas manejables y se seca al aire libre durante unos 10 o 15 días, los primeros 5 días sin sol. Se ha utilizado desde tiempos inmemoriales y se sigue utilizando extensamente en África, América Central y América del Sur. La tierra utilizada debe tener entre un 15 % y un 30 % de arcilla y el resto suelen ser áridos o arena. La proporción es importante pues un exceso de arcilla puede dar lugar a fisuras y una deficiencia de arcillas daría lugar a fragmentación de las piezas. Se adhiere bien a la madera y otros elementos orgánicos. El principal inconveniente es que puede disgregarse, principalmente por la lluvia, por lo que requiere mantenimiento y protección, generalmente por medio de enlucidos elásticos que pueden ser tan simples como una capa de barro. Otro inconveniente, que depende de las zonas en que se utilice, es que es muy vulnerable a los movimientos de tierra.

El ***cob*** o ***cobb*** (el término es de origen inglés y significaba un “terrón” o “masa redondeada”) es otra variante del adobe que consiste en mezclar arcilla, arena y barro con paja. La principal diferencia con el adobe es que se utiliza para conformar

directamente los muros en lugar de fabricar piezas lo que, entre otras cosas, permite crear con facilidad formas orgánicas, curvas. Al igual que el adobe, se ha utilizado desde tiempos prehistóricos en varios lugares del mundo. En Inglaterra hay varios ejemplos, principalmente en el oeste, sobre todo en la zona de Gales (en donde el término usado es *clom*). Los muros construidos con *cob* suelen ser muy gruesos, de unos 50 o 60 cm, y se construyen sobre unos cimientos excavados hasta encontrar una base rocosa suficientemente firme. Los muros se levantan poco a poco, en capas sucesivas, esperando a que la capa inferior esté suficientemente seca antes de continuar. Cuando los muros están suficientemente consolidados se insertan elementos tales como jambas o dinteles para formar las aperturas. A veces se protejen o decoran con un revoco o directamente con cal.

El ***tapial*** es otra técnica similar pero que utiliza un encofrado de tablas para conformar la construcción. Los componentes son similares a los del adobe, tierra arcillosa, pero la tierra se compacta por diversos medios (desde los pies hasta martillos de piedra) una vez que se ha vertido en los tablones que forman el encofrado. El encofrado es generalmente de madera pero también se pueden utilizar planchas metálicas. Al igual que los métodos anteriores, se ha utilizado en todos los lugares del mundo desde hace siglos, desde China hasta Iberoamérica o el Oriente Próximo, donde sigue siendo un sistema de construcción muy utilizado, a veces con equipos mecanizados (martillos neumáticos) que facilitan la compactación de la tierra. En algunos casos se incluyen en el interior de la tierra compactada elementos verticales de madera o bambú para aumentar la resistencia.

El ***bahareque*** o ***bajareque*** (palabra que parece ser de origen taíno, una lengua que se hablaba en las Antillas precolombinas) es otro sistema de construcción que se basa en la formación de una estructura previa con palos o cañas y rellenada con barro. La estructura se construye corrientemente a partir de un zócalo de piedras o ladrillos que afianza la estructura y la aísla de la humedad del suelo. El entramado de cañas puede ser sencillo o doble, formando una especie de encofrado. En cualquier caso, una vez formado este entramado se rellena con barro y pajas y, a veces, pequeñas piedras.

Todos estos métodos de construcción se están recuperando por su bajo coste y sus excelentes propiedades bioclimáticas pues funcionan como los botijos o las vasijas de barro, manteniendo una temperatura muy estable en su interior, tanto en invierno como en verano.

En varias construcciones recientes se han utilizado variantes de las técnicas anteriores, principalmente los ladrillos de tierra sin cocer, los ladrillos de adobe o la tierra compactada.

Los ***ladrillos de tierra sin cocer*** se han utilizado en algunos edificios recientes. Por ejemplo en el Mercado Central de Koudougou, en Burkina Fasso, proyectado por Laurent Séchaud para la Agencia Suiza para la Cooperación y el Desarrollo (SDC). En este caso se buscó expresamente la recuperación de técnicas tradicionales utilizando ladrillos de tierra estabilizada, fabricados a mano con la adición de un 4 % a un 12 % de cemento industrial, para la construcción de paredes y techos. Los cimientos se hicieron de hormigón. Otro ejemplo es el museo Buddha Repository, en Toyogura-gun, Yamaguchi, Japón, proyectado por Kengo Kuma & Associates, cuyas paredes están construidas con ***bloques de adobe*** de 35 cm elaborados con métodos tradicionales y que proporcionan un buen aislamiento climático. Y otro ejemplo de edificio construido con bloques de adobe es el complejo reconstruido Jahili Fort, en Al Ain, Abu Dhabi, proyectado por Eike Roswag y Ziegert, donde se ha mantenido la estructura de adobe original completada con bloques de barro y hojas de palmera. El suelo de tierra compactada se ha encerado y también se ha mantenido.

También hay edificios contemporáneos que se han construido con ***tierra compactada.*** Un ejemplo es el Taller de Arquitectura en Oaxaca, México, proyectado por el Taller de Arquitectura Mauricio Rocha. Muchos de los edificios de este complejo están construidos con tierra compactada, tierra mezclada con un 15 % de cemento, formando muros de 60 a 70 cm de espesor que favorece el aislamiento climático y acústico. Los costes y tiempo de ejecución también se redujeron considerablemente con este sistema de construcción. Otro ejemplo del uso de tierra compactada es la residencia en Scottsdale, AZ, Estados Unidos, proyectada por Kendle Design Collaborative, donde se uso la misma tie-

rra excavada del terreno para la construcción de la residencia. O la residencia Rauch, Vorarlberg, Austria, proyectada por Martin Rauch, Boltshauser Architekten. También en este caso se construyeron los muros, suelos (baldosas) y hasta techos con tierra compactada del propio terreno. O la villa Yasmin, en Kfar Shmaryahu, Israel, proyectada por Elie Mouyal, ADAMA Building & Architecture, con muros de carga macizos construidos de este modo. Se pueden ver fotografías de estos y otros ejemplos citados en el párrafo anterior y en los que siguen en el libro de Cristina Paredes (2011).

Bambú. Mimbre. Hierba. Paja

El ***bambú*** , denominación corriente de la *bambusoideae*, es una planta que pertenece a la familia de las gramíneas. Crece con una gran variedad de tamaños, desde troncos de menos de 1 metro de altura con tallos de 0,5 cm de diámetro hasta, menos corrientemente, troncos de hasta 25 metros de alto y tallos de 30 cm de diámetro. Crece de modo natural en todos los continentes excepto en Europa. Los troncos pueden ser herbáceos (blandos y flexibles debido al predominio de colénquima, un tejido de sostén formado por células vivas, como ocurre en las hierbas corrientes) o leñosos (duros y rígidos debido al predominio de esclerénquima, células muertas con paredes endurecidas, como ocurre en la madera) y estos últimos son los que se utilizan en arquitectura. Son elementos huecos pero con tabiques transversales que aseguran su rigidez y resistencia. Es un material natural, reciclable, con buenas propiedades de resistencia y aislamiento térmico y acústico. Su mayor desventaja es su poca duración relativa pues es sensible a los ataques biológicos y también a los vientos huracanados y al fuego.

Se ha utilizado desde tiempos inmemoriales en regiones del planeta en donde abunda: en China, Japón, Filipinas o la mayoría de las islas de Indonesia. Y se sigue utilizando en la actualidad, no solo para la construcción popular sino también en diferentes regiones de planeta. Algunos ejemplos recientes notables son una residencia privada en Guilford, CT, Estados Unidos, proyectada por Gray Organschi Architecture donde los muros, tabiques, suelos y techos están realizados con bambú. Un centro de educación para niños en Six Senses, Koh Kut, Tailandia, proyectado por 24H>architecture, construido con troncos de bambú tratado con boro para protegerlo de los insectos y elevado unos 30 cm del suelo para evitar la humedad. El WNW Bar en Thu Dau Mot, Vietnam, proyectado por Vo Trong Nghia, que incluye una gran cúpula formada por arcos de bambú prefabricados y un recubrimiento cruzado de cañas de bambú unidas entre sí. Todo el conjunto está recubierto por lo que parece ser paja. Una escuela, la Green School en Badung, Indonesia, junto al río Ayung, proyectada por John y Cynthia Hardy (premio Aga Khan en arquitectura), arquitectos pioneros de construcciones ecologistas que fomenten modos de vida sostenible en sus comunidades y que promueven la enseñanza de construcciones con bambú para las comunidades que quieren participar en estas iniciativas. El proyecto está construido enteramente con troncos de bambú. Pueden verse imágenes de los edificios citados en la obra citada de Cristina Paredes.

El ***mimbre*** es una fibra vegetal que se obtiene de un arbusto de la familia de los sauces (principalmente del género Salix) y que crece en los márgenes de los ríos y acequias de muchas zonas de Europa y otros países. Se puede cultivar para controlar mejor el producto pues crece en troncos gruesos de los que se arrancan periódicamente tallos más finos. Estos tallos se tejen para formar entramados que se utilizan sobre todo en la fabricación de sillas, cestos y otro tipo de utensilios domésticos. En arquitectura se ha utilizado a veces como recubrimiento de elementos, tabiques o muros y permite crear pantallas ligeras que filtran la luz del sol. Un ejemplo reciente muy famoso, de construcción con mimbre, es el pabellón de España en la Expo de Shangai de 2010, proyectado por Miralles Tagliabue EMBT. El pabellón está recubierto de paneles formados por unas estructuras metálicas deformadas recubiertas por mimbre trenzado.

La ***paja*** se ha utilizado tradicionalmente en cubiertas. La paja es el tallo seco de ciertas gramíneas que queda después de haber separado el grano o la semilla. Es un material de deshecho pero que se reutiliza tradicionalmente de muy diversos modos, principalmente en usos agrícolas, como elemento de protección del terreno o de plantaciones. Sus principales características físicas son las

siguientes. En combinación con otros materiales, y siempre que se garantice la estanqueidad (que no penetren el viento ni el aire ni la humedad) es un buen aislante térmico y acústico. La paja proporciona un buen aislamiento durante el invierno de un modo natural, es barato y requiere poco mantenimiento, aunque se deteriora con cierta facilidad. Se ha utilizado principalmente en cubiertas debido a sus cualidades aislantes y a que, en condiciones normales, la lluvia resbala sobre ella pero no la penetra.

Pero también hay ejemplos recientes. Uno particularmente notable es la residencia en Steigereiland, en Holanda, proyectada por el estudio MOPET Architecten. El diseño está influido por la Escuela de Amsterdam de principios del siglo xx y la gran cubierta está forrada completamente de paja. Otro ejemplo reciente es un edificio "Barn", en Alkmaar, Holanda, proyectado por el estudio 24>architecture. El tejado está formado por pequeñas cañas de paja distribuidas en niveles que proporcionan una estructura muy original y termina con un gran voladizo ondulado que protege las pequeñas terrazas.

Cartón. Telas. Plásticos. Otros materiales

Hay un gran número de casas y edificios construidos con ***cartón,*** un material barato con algunas propiedades interesantes y que se puede reemplazar con facilidad. Aparte del bajo coste, es ligero y fácil de trabajar. Si va a quedar expuesto conviene protejerlo con cera o algún barniz impermeable. Los ejemplos más conocidos, ya citados al comienzo, son los debidos a varias obras de Shigeru Ban, como la casa de papel, construida tras el terremoto de Kobe de 1995 que dejó a muchas personas sin hogar, creada por medio de una estructura simple utilizando cajas de cerveza llenas de arena como cimientos, paredes con tubos de papel impermeabilizados y pegados entre sí y un techo abatible que podía separarse del cielo raso en verano para permitir la ventilación. Por añadidura, todos los materiales eran reciclables. En muchos otros edificios Shigeru Ban ha utilizado tubos de cartón. Otro ejemplo notable es el auditorio que diseñó en L'Aquila, tras el terremoto de 2009 (y que tuvo que sortear todo tipo de dificultades administrativas para llevarse a cabo), y cuya acústica resultó ser magnífica, según el testimonio de los músicos que lo inauguraron el 7 de mayo de 2011.

Otro ejemplo famoso es el pabellón finlandés en la exposición de Shangai de 2010, de JKMM Architects, diseñado con la intención de que tenga una vida más larga de lo que es usual en este tipo de eventos y que se puede desmontar y trasladar con facilidad. Se basó en ***material reciclado,*** principalmente una **mezcla de papel y plástico** con el que se han construido las escamas que conforman la fachada. Otro ejemplo en este mismo contexto es el pabellón alemán (también para la exposición de Shangai de 2010) debido a Schmidhuber & Partners. En este caso es la fachada la que está formada por una ***membrana transparente de material textil*** que protege del sol.

La tela se ha utilizado también desde tiempos inmemoriales, sobre todo por tribus nómadas, y se sigue utilizando con todo tipo de formas, sacando partido de los avances de la tecnología textil. Algunos de los ejemplos más originales pueden ser las que se encuentran en el pabellón para la empresa textil Kvadrat, por Ronan y Erwan Bouroullec, Estocolmo, construido con paneles formados por ***tejas de telas*** con aislante acústico. O el pabellón para la empresa de servicios Uboot.com, en Düsseldorf, diseñado por el estudio Hackenbroich Architekten. Incluía 1.200 ***tiras textiles*** de diferentes longitudes colgadas del techo. También pueden verse ilustraciones en el libro de Cristina Paredes citado.

Para terminar de nuevo con Shigeru Ban, se puede recordar que este arquitecto también ha utilizado contenedores, bambú, fibra de papel reciclado, plástico reciclado y otros materiales. En la "casa desnuda", en Saitama, Japón (2000), revistió las paredes externas con plástico transparente ondulado y otras zonas con acrícilico blanco estirado sobre marcos de madera. En el Centro Pompidou de Metz, Francia (2003), creó una cubierta con una celosía ondulante aireada de lamas de madera. También ha utilizado tubos de cartón para formar columnas o vigas, por ejemplo en construcciones temporales hechas en Ruanda (1999) tras el conflicto de 1994, promovidas por la UNHCR (United Nations High Commissioner for Refugees).

Otros arquitectos que cabe citar y que se han especializado en construir con materiales y técnicas autóctonos o alternativos son la arquitecta austría-

ca Anna Heringer (n 1977), una de cuyas primeras obras fue una escuela en Rudrapur (Bangladesh), construida con barro y bambú en 2006. La arquitecta india Anupama Kundoo (casada con el arquitecto español Luis Feduchi), entre cuyas obras cabe citar su casa en la India (Wall House, construida en 2000 y reproducida a escala real en la Bienal de Venecia de 2012), en la que, entre otras cosas, utiliza excedentes de vasijas artesanales de terracota para aligerar la losa de hormigón; o las casas construidas levantando una estructura abovedada de barro que luego se cuece desde el interior durante cuatro días utilizando polvo de carbón con arcilla para hacer el fuego. O bien los arquitectos Carlos Andrés Restrepo y Elizabeth Milagros que levantaron, en colaboración con los habitantes del caserío de Piedritas, en el norte del Perú, la Escuela Santa Elena, utilizando caña brava y reciclando materiales metálicos. Pueden verse estos trabajos y otros similares, en el reportaje de Anatxu Zabalbeascoa publicado en el suplemento "Babelia" de *El País*, el 21 de junio de 2014, o la versión ampliada que se encuentra en los enlaces del mismo artículo en versión digital.

Simulación visual

La simulación de los materiales citados no presenta ninguna dificultad especial. En general se trata de materiales que se remiten a los que ya hemos visto (cerámica, plásticos, madera) y que pueden simularse fácilmente con las técnicas desarrolladas a lo largo de los capítulos anteriores. Tampoco requieren, en general, obtener imágenes de texturas singulares pues en muchos casos se trata de materiales informes que pueden simularse con colores planos combinados con texturas procedurales que generen algo de ruido.

Por estas razones no incluyo ejemplos, que deberían ser muy variados, para no alargar innecesariamente este capítulo.

Referencias

Acocella, A. 2004. *L'architettura di pietra: antichi e nuovi magisteri costruttivi*. Firenze, Alinea Editrice.

Adelson, E.H; Bergen, J.R. 1991. "The Plenoptic Function and the Elements of Early Vision". En: M. Landy and J. A. Movshon (eds), *Computational Models of Visual Processing*, pp 3-20. Cambridge, MA, MIT Press.

Albers, J. 1975. *Interaction of color*. Yale University Press, New Haven. Trad. Esp.: *La interacción del color*. Madrid, Alianza, 2003.

Artigas, J.M.; Capilla, P.; Pujol, J. (coords). 2002. *Tecnología del color*. Universidad de Valencia.

Ashikhmin, M.; Shirley, P. 2000. "An Anisotropic Phong BRDF Model". *Journal of Graphics Tools*. Vol 5, n 2, pp 25-32. Véase también: http://www.cs.utah.edu/~michael/brdfs/.

ASTM (American Society for Testing and Materials): http://www.astm.org/.

Banks, D. 1994. "Illumination in Diverse Codimensions". *Proceedings SIGGRAPH*. July 1994, pp 327-334.

Becker B. G., Max N. L. 1993. "Smooth transitions between bump rendering algorithms". *SIGGRAPH '93 Proceedings* (1993), ACM Press, pp 183–190.

Becket W., Badler N. I. 1990. "Imperfection for Realistic Image Synthesis". *The Journal of Visualization and Computer Animation*. Vol 1, pp 26-32 (1990).

Berlin, B.; Kay, P. 1969. *Basic Color Terms: Their Universality and Evolution*. Univ. of California Press, Berkeley.

Blinn, J.F. 1977. "Models of light reflection for compuer synthesized pictures". *Computer Graphics (Proc. Siggraph '77)*. Vol 11 (2), pp 192-198, july 1977.

Blinn, J.F. 1978. "Simulation of Wrinkled Surfaces", *Computer Graphics (SIGGRAPH '78 Proceedings)*. Vol 12, n 3, pp 286-292.

Blinn, J.F.; Newell, M.E. 1976. "Texture and Reflection in Computer Generated Images". *Communications of the ACM*. Vol 19, pp 542-546.

Biliouris, D.; Verstraeten, W.W.; Dutré, P.; Van Aardt, J.; Muys, B.; Coppin, P. 2007. "A Compact Laboratory Spectro-Goniometer (CLabSpeG) to Assess the BRDF of Materials". *Sensors*, 2007, 7, 1846-1870.

Bonn. Universität Bonn. Institute of Computer Science II. *Computer Graphics*. Ver: http://cg.cs.uni-bonn.de/en/projects/btfdbb/.

Brawley, Z.; Tatarchuk, N. 2004. "Parallax Occlusion Mapping: Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing". en *ShaderX3: Advanced Rendering with DirectX and OpenGL*, Engel, W., Ed., Charles River Media, pp. 135-154.

Bustillo, M.; Calvo, J.P. 2005. *Materiales de construcción*. Madrid, Fueyo.

Cabral, B.; Max, N.; Springmeyer, R. 1987. "Bidirectional reflection functions from surface bump maps". *Computer Graphics*. 21(4), pp 273-281, July 1987.

Catmull, E.E. 1975. "Computer Display of Curved Surfaces". *Proceedings of the IEEE Conference on Computer Graphics, Pattern Recognition, and Data Structure*. May, 1975, pp 11-17. Versión reducida de la tesis doctoral "A Subdivision Algorithm for Computer Display of Curved Surfaces", presentada en el Dept. of CS, U. of Utah", 1974.

CIE Commission Internationale de l'Eclairage. Ver: *www.cie.co.at*.

Cignoni et al. 1998. "A general method for preserving attribute values on simplified meshes". *IEEE Visualization '98*.

Cohen et al. 1998. "Appearance Preserving Simplification". *SIGGRAPH 1998*.

Cook, R. L.; Torrance, K.E. 1981. "A reflectance model for computer graphics". *Computer Graphics (SIGGRAPH '81 Proceedings)*, Vol.15, n 3, july 1981, pp 301-316. Publicado también en *ACM Transactions on Graphics*. Vol 1, n 1, january 1982, pp 7-24.

Cook, R.L. 1984. “Shade Trees”. *Computer Graphics*. Vol 18, n 3, pp 223-232.

Cook,R.; Carpenter, L.; Catmull, E. 1987. “The Reyes Image Rendering Architecture”. *Computer Graphics*. Vol 12, n 4, july 1987, pp 95-102.

Cook, R.L.; DeRose, T. 2005. “Wavelet Noise”. *Proceedings of SIGGRAPH 2005*/ Pixar Animation Studios.

Cornell Light Measurement Laboratory. Ver: http://www.graphics.cornell.edu/research/measure/.

Crick, F. 1993. *The Astonishing Hypothesis: The Scientif Search for the Soul*. Scribner, New York. Trad. Esp: *La búsqueda científica del alma: una revolucionaria hipótesis para el siglo XXI*. Madrid, Debate, 1994.

Crow, F.C. 1977. “The Aliasing Problem in Computer Generated Shaded Images”. *Comm ACM*. Vol 20, n 11, nov 1977, pp 799-805.

Crow, F.C. 1984. “Summed-area tables for texture mapping”. *Computer Graphics (SIGGRAPH ’84 Proceedings)*, 18, pp 207-212.

Curet: Columbia-Utrech reflectance and texture database. Ver: http://www.cs.columbia.edu/ CAVE/software/curet/.

Dana, K.J.; Van Ginneken, B.; Nayar, S.K.; Koenderink, J.J. 1999. “Reflectance and Texture of Real World Surfaces”. *ACM Transactions on Graphics*. Vol.18, n 1, pp 1-34, jan, 1999.

De Valois, R.L.; De Valois, K.K. 1988. *Spatial Vision*. Oxford, New York.

Dietrich S.: *Elevation Maps*. NVIDIA Corporation, 2000.

Ditchburn, R.W. 1961. *Light*. Reed. Dover, New York, 1991. Trad. Esp.: *Óptica*. Barcelona, Reverté, 1982.

Doggett, M.; Hirche, J. 2000. “Adaptive View Dependent Tessellatin of Displacement Maps”. *Eurographics/SIGGRAPH workshop on graphics hardware*. September 2000. Ver también otros trabajos de Doggett sobre este tema en: http://www.cs.lth.se/home/Michael_Doggett/.

Dorsey, J.; Hanrahan. 1996. “Modeling and Rendering of Metallic Patinas”. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp 387 – 396.

Dorsey, J; Edelman, A.; Jensen, H.W.; Legakis, J.; Pedersen, H. K. 1999. “Modeling and rendering of weathered stone”. *ACM SIGGRAPH 2006. Proceedings of the 26th International Conference on Computer Graphics and Interactive Techniques*. July 1999, pp 225-234.

Drebin, R.A.; Carpenter, L; Hanrahan, P. 1988. “Volume Rendering”. *Computer Graphics (Proceedings of SIGGRAPH 88)*. Vol 22, n 4, pp 65-74.

Drienmayer, T. 2005. *Rendering with mental ray*. Springer/Wien/NewYork. 3ª edición (1ª edición 1986).

Ebert, D.S. 1997. “Volumetric Modeling with Implicit Functions (A Cloud is Born)”. *SIGGRAPH ’97 Proceedings. ACM Siggraph*.

Ebert, D.S. 1994. Musgrave, K.; Peachey, D.; Perlin, K.; Steven Worley, S. *Texturing and Modeling: A Procedural Approach*, 3ª ed. Morgan Kaufman, 2002.

Engelsmann, S; Spalding, V; Peters, S. 2010. *Plastics in Architecture and Construction*. Birkhauser, Basel.

Fairchild, M.D. 1997. *Color Appearance Models*. Addison Wesley, Reading, Massachusetts.

Fox, M. *Optical Properties of Solids*. 2001. Oxford University Press.

Foley, J.D.; Van Dam, A.; Feiner, S.K. Steven; Hughes,J.F. 1990. *Computer Graphics. Principles and Practice*. Reading, Mass., Addison and Wesley. 2ª ed, 1996. 3ª ed. 2013.

Fournier, A. 1992. “Normal distribution functions and multiple surfaces”. *Graphics Interface ’92 Workshop on Local Illumination*, pp. 45–52.

Galileo, G. 1623. *El Ensayador*. Madrid, Sarpe, 1984 (Trad. De J.M.Revuelta. Ed Orig.: *Il Saggiatore*, c.1623).

Gardner, G. 1985. “Simulacion of natural scenes using textured quadric surfaces”. *Computer Graphics (SIGGRAPH ’84 Proceedings)*. 18 pp 11-20.

Gardner, G. 1985. “Visual simulation of clouds”. *Computer Graphics (SIGGRAPH ‘85 Proceedings)*. 19 (3), pp 297-304.

Gibson, J.J. 1950. *The Perception of the Visual World*. Boston, Houghton-Mifflin. Trad. Esp.: *La percepción del mundo visual*. Buenos Aires, Ediciones Infinito, 1974.

Gibson, J. J. 1979. *The Ecological Approach to Visual Perception*. Boston, Houghton Mifflin.

Glassner, A.S. 1995. *Principles of Digital Image Synthesis*. San Francisco, Morgan Kaufman.

Gondek, J.S.; Meyer, G.W.; Newman, J.G. 1994.

"Wavelength dependent reflectance functions. *SIGGRAPH 94 Conference Proceedings*, pp 213-220, Orlando. Florida, July 1994.

Greene, N. 1986. "Environment mapping and other applications of world projections". *IEEE Comput. Graph. Appl*. 6, 11, nov 1986, pp 21-29.

Gregory, R.L. 1966. *Eye and Brain*. Princenton University Press, Princenton, New Jersey. Trad. Esp.: *Ojo y cerebro. Psicología de la visión*. Madrid, Guadarrama, 1965 (sic).

Hanrahan, P.; Lawson, J. 1990. "A language for shading and lighting calculation". *Computer Graphics 24(4), (Proc.SIGGRAPH 90)*. Aug, 1990, pp 289-298.

Hanrahan, P.; Krueger, W. 1993. "Reflection from layered surfaces due to subsurface scattering". *SIGGRAPH 93 Conference Proceedings*, pp 165-174, Anaheim, California, August 1993.

He, X.D.; Torrance, E.; Sillion, F.X.; Greenberg, D.P. 1991. "A comprehensive physical model for light reflection". *Computer Graphics (Proc. Siggraph '91)*. 25(4), July 1991, pp 175-186.

He, X., Heynen, P., Phillips, R., Torrance, K., Salesin, D., and Greenberg, D. 1992. "A Fast and Accurate Light Reflection Model", *Siggraph 1992*.

Henyey, L.G.; J.L. Greenstein, J.L. 1941. "Diffuse radiation in the galaxy". *Astrophysical Journal*. 93, pp 70-83, 1941.

Hering, E. 1905. *Grundzüge der Lehre vom Lichtsinn*. Trad Ingl. de Hurvich and Jameson: *Outlines of a Theory of the Light Sense*, 1964, Harvard University Press.

Homan, J.P. 2009. *Digital Color Management. Principles and strategies for the Standardized Print Production*. Springer, Berlin.

Hubel, D.H. 1988. *Eye, Brain and Vision*. W.H. Freeman, New York.

ICC (International Color Consortium). Véase: *www.color.org*.

ISO (International Standards Organization). Véase: *www.iso.org*.

Jensen, H.W.; Legakis, J.; Dorsey, J. 1999. "Rendering of Wet Materials". *Eurographics Workshop on Rendering*.

Jensen, H.W.: "Subsurface Scattering". Véase: http:// graphics.ucsd.edu/ ~henrik/images/subsurf.html.

Jensen, H.W; Marschner, S; Levoy, M; Hanrahan, P. wwwSubSurfLT. 2001. "A Practical Model for Subsurface Light Transport". *Proceedings of Siggraph'2001*. Disponible en: http://graphics.stanford.edu/papers/bssrdf/.

Jensen, H.W.; Donner, C.: "A Spectral BSSRDF for Shading Human Skin". Disponible en: http://graphics.ucsd.edu/~henrik/papers/skin_bssrdf/.

Judd, D.; Wyszecki, G. 1975. *Color in Business, Science and Industry*. Wiley, NY.

Julesz, B. 1971. *Foundations of Cyclopean Perception*. University of Chicago Press.

Kaneko, T., et al. 2001. "Detailed Shape Representation with Parallax Mapping". *Proceedings of ICAT 2001*, pp. 205-208.

Kanizsa, G. 1979. *Organization in Vision. Essays on Gestalt Perception*. Praeger, NY. Trad. Esp.: *Gramática de la Visión: Percepción y Pensamiento*. Barcelona, Paidós, 1986.

Julesz, B. 1981. "Textons, the Elements of Texture Perception, and their Interactions". *Nature*. March 1981, n 290 (5802), pp 91–97.

Kajiya, J.T. 1985. "Anisotropic reflection models". *Computer Graphics*. Vol 19 (3), pp 15-21.

Kajiya, J.T. 1986. "The rendering equation". *Computer Graphics*. Vol 20 (4), pp 143-50.

Kajiya, J.T.; Kay, T.L. 1989. "Rendering Fur with Three Dimensional Textures". *Computer Graphics* (*SIGGRAPH '89 Proceedings*). Vol 23, pp 271-280.

Kautz J. and Seidel H. 2000. "Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. *Siggraph/Eurographics Workshop On Graphics Hardware*, pp 51-58.

Kittel, C. 1953, 2003. *Introducción a la Física del Estado Sólido*. Barcelona/México, Reverté, 2003 (reimpresión de la 3ª ed, 1997 basada en la 6ª edición del original: *Introduction to Solid State Physics*, de 1953).

Koenderink, J.J.; Van Doorn, A.J.; Stavridi, M. 1996. "Bidirectional reflection distribution function expressed in terms of surface scattering modes". *European Conference on Computer Vision*, pp 28-39.

Koutajoki, K. 2002. "BSSRDF (Bidirectional Surface Scattering Distribution Function". Disponible en: http://www.tml.tkk.fi/Opinnot/Tik-111.500/2002/paperit/kalle_koutajoki.pdf.

Krishnamurthy, R; Levoy, M. 1996. "Fitting Smooth Surfaces to Dense Polygon Meshes". SIG-

GRAPH 1996.

Kurt, M.; Edwards, D. 2009. “A Survey of BRDF Models for Computer Graphics”. *ACM Siggraph. Computer Graphics Quarterly*. Vol 43, num 2, may 2009.

Lafortune, E.P.F.; Foo, S.; Torrance, K.E.; Greenberg, D.P. 1997. “Non-linear approximation of reflectance functions”. *Proceedings of Siggraph ‘97*, pp 117-126.

Land, E.H. 1959. “Experiments in color vision”. *Scientific American*, 200(5), pp 84-99.

Land, E.H. & McCann, J.. 1971. “Lightness and retinex theory”. *Journal of the Optical Society of America*, 61, pp 1-11.

Land, E.H. 1977. “The retinex theory of vision”. *Scientific American*, 237(6): 108-28. Trad. esp.: “La teoría retinex de la visión del color”. *Investigación y Ciencia*, 1978, n.17, febrero, p 64 y ss.

Land, E.H. 1983. “Recent advances in retinex theory and some implications for cortical computations: color vision and the natural imagery”. *Proceedings of the National Academy of SciencesUSA*, 80: 5163-69.

Lawrence, J. 2002. “Importance Sampling of the Phong Reflectance Model”. https://www.cs.virginia.edu/~jdl/importance.doc.

Lawrence J.; Rusinkiewicz, S.; Ramamoorthi, R. 2003. “Efficient BRDF importance sampling using a factored representation”. *ACM Transactions on Graphics 2004*. Vol 23(3), pp 496-505.

Lewis, R.R. 1993. “Making shaders more physically plausible”. *Fourth Eurographics Workshop on Rendering*. June 1993, pp 47-62.

Linhares, J.M.; Pinto, P.D.; Cardoso, S.M. 2008. “The number of discernable colors in natural scenes”. *Journal of the Optical Society of America*. Vol 25, n 12, december 2008.

Livny, B. 2008. *Mental Ray for Maya, 3ds Max and XSI*. Indiana, Wiley.

Mandelbrot, B. 1975. *Les objets fractals: forme, hasard, et dimension*. Paris, Flammarion, 1975.

Mandelbrot, B. 1982. *The Fractal Geometry of Nature*. New York, W. H. Freeman and Company, 1982.

Marr, D. 1982. *Vision*. W.H. Freeman, San Francisco. Trad. Esp.: *La Visión*. Madrid, Alianza, 1985.

Matusik, W.; Pfister, H.; Brand, M.; McMillian, L. 2003. “A data driven reflectance model”. *ACM Transactions on Graphics 2003*. 22(3), pp 759-769.

mental ray. *Architectural Shader Library*. 2007. Véase: http://www.nvidia-arc.com/ fileadmin/ user_upload/ PDF/ arch_and_design.pdf. Véase también: http://docs.autodesk.com/ MENTALRAY/2014/ENU/mental-ray-help/.

Miller, G.S.; Hoffman, R.C. 1984. “Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments”. *Course Notes for Advanced Computer Graphics Animation*, *SIGGRAPH 84*.

Mitchell, D.P.; Netravali, A. 1988. “Reconstruction filters in Computer Graphics”. *Computer Graphics (Proc.Siggraph ’88)*. Vol 22, n 4, pp 221-228, july 1988.

McAllister D., Lastra A. and Heidrich W. 2002. “Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions”. *Graphics Hardware 2002*, pp 78-88.

Morrison, P.; Morrison, P; Eames Office. 1982. *Powers of Ten*. Scientific American Books. Trad.Esp.: *Potencias de diez. Sobre el tamaño relativo de los objetos del universo*. Madrid, Labor, 1984. (Basado en la película del mismo título, preparada en la Oficina de Charles y Ray Eames, en 1968 y reeditada en 1977. Puede verse el documental en http://www.powersof10.com/).

Musgrave, F.; Kenton, F.; Mandelbrot, B. 1989. “Natura Ex Machina”. *IEEE Computer Graphics and Applications* 9 (1). January, 1989, pp 4-7.

Musgrave, F.F. Ver muestras de sus paisajes fractales y otros trabajos en: http://www.kenmusgrave.com/.

Narasimhan, S.; Gupta, M.; Donner, C.; Ramamoorthi, R.; Nayar, S.; Jensen, H.W. 2006. “Acquiring Scattering Properties of Participating Media by Dilution”. *Proceedings Siggraph 2006*. Disponible en: http://graphics.ucsd.edu/~henrik/papers/acquiring_scattering_properties/.

Nassau, K. 1983. *The Physics and Chemistry of Color. The Fifteen Causes of Color*. John Wiley, 2001, Second Edition.

Nayyar, S.K; Oren, M. 1993. “Generalization of the Lambertian Model and Implications for Machine Vision”. *International Journal on Computer Vision*. Vol.14, n 3, pp 227-251, apr, 1995 (publicado originalmente en 1993).

Ngan, A.; Durand, F.; Matusik, W. 2005. “Experimental Analysis of BRDF Models”. *Rendering Techniques ‘05 (Proceedings of the Eurographics Symposium on Rendering)*. Konstanz, Eurographics Association, 2005, pp 117-226.

Nicodemus, F. E. 1963. “Directional Reflectance and Emissivity of an Opaque Surface”. *Applied Optics*, Vol 4, issue 7, pp 767-773.

Nicodemus, F. E. 1970. “Reflectance nomenclature and directional reflectance and emissivity”. *Applied Optics*. Vol 9, pp 1474–1475.

Nicodemus, F.E.; Richmond, J.C.; Hsia, J.J. 1977. “Geometrical Considerations and Nomenclatue for Reflectance”. *National Bureau of Standards*, Washington, D.C., October 1977 (con la colaboración de I.W.Ginsberg y T.Limperis).

NIST (National Institute of Standards and Technology): Véase: http://www.nist.gov/index.html. Proyecto de Fern Hunt sobre “Computer Graphic Rendering of Material Surfaces”. Véase: http://math.nist.gov/~FHunt/appearance/index.html. Véase también: *Nist reference Reflectometer: Starr Facility*: http://Physics.Nist.Gov/.

Oliveira M. M., Bishop G., Mcallister, D. 2000. “Relief texture mapping”. *SIGGRAPH 2000 Proceedings*, pp. 359–368.

Paredes, C. 2011. *La biblia de los materiales de arquitectura*. Loft Publications, Barcelona.

Peachey, D.R. 1985. “Solid Texturing of complex surfaces”. *Computer Graphics* (*Proc*, *SIGGRAPH 85*). Vol.20, n 4, pp 55-64.

Peitgen, H.O. 1988. *The Science of Fractal Images*. Springer.

Perlin, K. 1985. “An Image Synthesizer”. *Computer Graphics*, Vol.19, n 3, pp 279-286.

Perlin, K.; Hoffert, E. 1989. “Hypertexture”. *Computer Graphics (proceedings of ACM SIGGRAPH)*. Vol 23, n 3.

Phong, B.T. 1975. “Ilumination for Computer-generated pictures”. *Communications of the ACM*. I8, june I975, pp 3II-3I7. (Tesis, University of Utah, July I973).

Pointer, M.R.; Attridge, G.G. 1998. “The number of discernible colours”. *Color Res. Appl*. Vol 23, pp 52-54.

Policarpo F., Oliveira M. M., Comba J. 2005. “Real-time relief mapping on arbitrary polygonal surfaces”. *ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pp. 155–162.

Polyak, S.L. 1941. *The Retina*. University of Chicago Press, Chicago.

Poulin, P.; Fournier, A. 1990. “A model for anisotropic reflection”. *Computer Graphics (Proc. Siggraph ‘90)*. Vol 24(4), pp 273-282, august 1990.

Pratt, W.K. 1978, 2007. *Digital Image Processing*. 1ª ed., McGraw-Hill, Nueva York, 4ª ed. revisada, 2007.

Reeves, W.T. 1983. Particle systems: A technique for modelling a class of fuzzy objects”. *ACM Transactions of Graphics*. Vol 2, pp 91-108.

Reeves, W.T.; Blau, R. 1985. “Approximate and probabilistic algorithms for shading and rendering structured particle systems”. *Computer Graphics*. Vol 19, pp 313-322.

Reynolds, C.W. 1987. “Flocks, Herds, and Schools: A Distributed Behavioral Model”, *Computer Graphics*. Vol 21, n 4, pp 25-34.

Schlick, C.; Blasi, P.; Le Saec, B. 1993. “A rendering algorithm for Discrete Volume Density Objects”. *Eurographics ’93*. Vol 12, n 3.

Schröeder, P.; Sweldens, W. 1995. “Spherical wavelets: Effciently representing functions on the sphere”. *SIGGRAPH 95 Conference Proceedings*, pp 161-172, Los Angeles, California, August, 1995.

Shirley, P.; Smits, B; Hu, H.; Lafortune, E. 1997. “A practitioners’ assessment of light reflection models”. 1997. *Proceedings of the 5th Pacific Conference on Computer Graphics and Applications*, p 40, october 13-16.

Shum, H-Y; Kang, S.B. 2000. “A Review of Image-based Rendering Techniques”. *IEEE/SPIE Visual Communications and Image Processing 200*, pp 2-131.

Stiles, W.S.; Wyszechi, G. 1982. *Color Science. Concepts and Methods, Quantitative Data and Formulae*. New York, Wiley.

Strauss, P. S. 1990. “A realistic lighting model for computer animators”. *IEEE Computer Graphics & Applications*. Vol 10 (11), november 1990, pp 56-64.

Suykens F., vom Berge K., Lagae A. and Dutré P. 2003. “Interactive rendering of bidirectional texture functions. *Eurographics 2003*, pp 463-472.

Szirmay-Kalos, L.; Umenhoffer, T. 2008. “Displa-

cement Mapping on the GPU-State of the Art". *Computer Graphics Forum*. Vol 27 (1).

Tatarchuk, N. 2005. "Practical Dynamic Parallax Occlusion Mapping". *Siggraph presentation*.

Torrance, K.E., Sparrow, E.M. 1967. "Theory for off-specular reflection from roughened sufaces"". *Optical Society of America*. Vol 57 (9), pp 1105-14.

Upstill, S. 1990. *The RenderMan Companion*. Addison-Wesley, Reading, MA, USA, 1990.

Wandell, B.A. 1995. *Foundations of Vision*. Sinauer Ass., Massachusetts.

Ward, G. 1992. "Measuring and modeling anisotropic reflection". *Computer Graphics (Proc. Siggraph '92)*. Vol 26 (4), july 1992, pp 265-272.

Welsh T. 2004. "Parallax Mapping with Offset Limiting: A PerPixel Approximation of Uneven Surfaces". *Tech. rep., Infiscape Corporation*.

Westin, S., Arvo, J., and Torrance, K. 1992. "Predicting Reflectance Functions from Complex Surfaces", *Computer Graphics*. Vol 26(2), pp 265-264, July 1992.

Whitted, T. 1978. "A scan line algorithm for computer display of curved surfaces". *ACM SIGGRAPH Computer Graphics*. Vol 12, august 1978, pp 8-13.

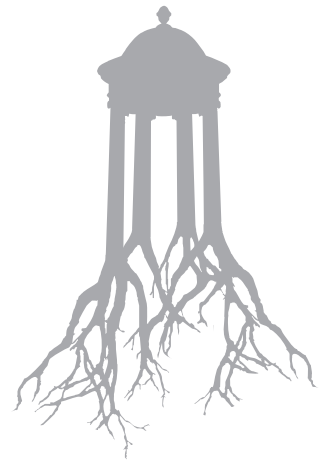
Williams, L. 1983. "Pyramidal parametrics". *Computer Graphics. (Proc. SIGGRAPH 83)*. Vol 17 (3), july 1982, pp 1-11.

Wolfgang Heidrich, W.; Seidel, H. 1999. "Realistic, hardware-accelerated shading and lighting". *Siggraph 1999, Annual Conference Proceedings*, 1999.

Worley, S. 1996. "A cellular Texture Basis Function". *Computer Graphics* (*SIGGRAPH '96 Proceedings*), pp 291-294.

Zahner, L.W. 1995. *Architectural Metals. A Guide to Selection, Specification and Performance*. New York, J.Wiley.

Zeki, S. 1993. *A Vision of the Brain*. Trad.Esp.: *Una Visión del Cerebro*. Barcelona, Ariel, 1995.



Índice analítico