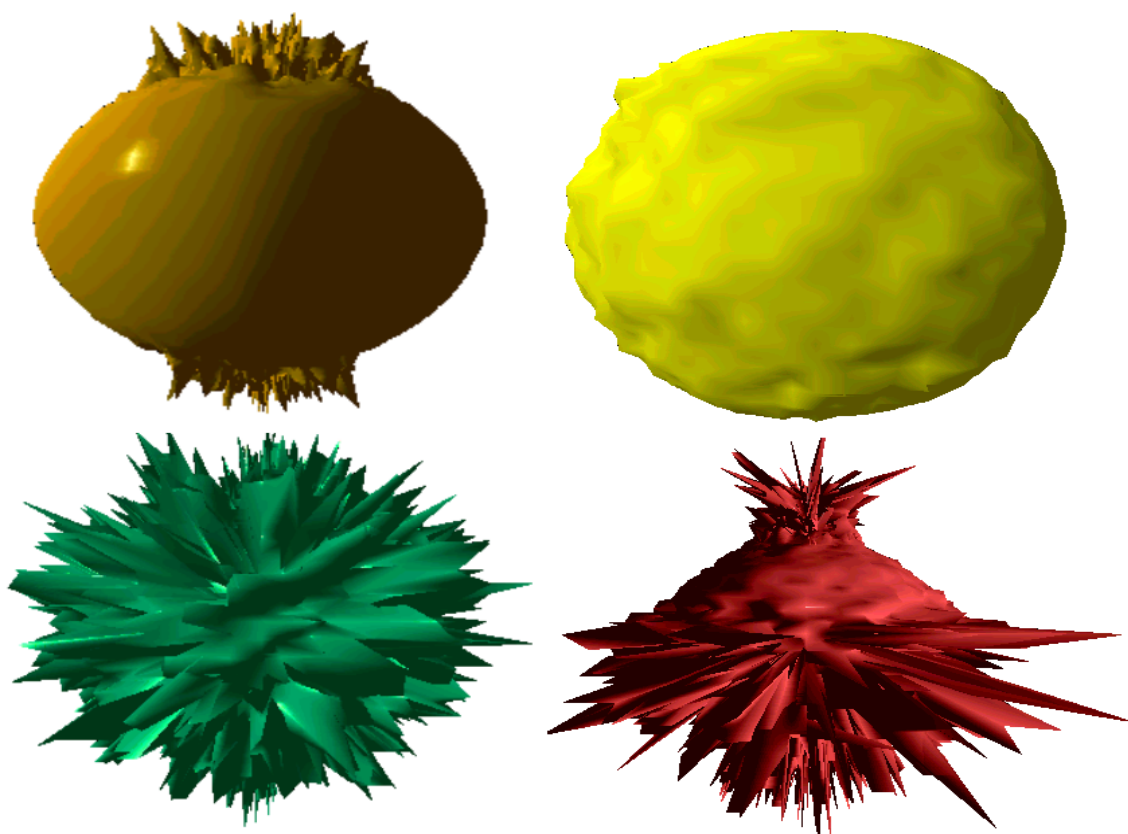

9

Temas Avanzados



9.1 Modelos de refracción

Los modelos de refracción constituyen el desafío más significativo en la representación de realismo gráfico, por la riqueza de los efectos que se producen, y por la enorme complejidad involucrada en el cálculo. La técnica de rendering más satisfactoria al respecto parece ser en la actualidad el ray-tracing, porque esta forma de computar los modelos de refracción es geoméricamente exacta por definición. Una objeción importante es que si bien el modelo es geoméricamente exacto, la representación de objetos en los cuales el modelo es efectivamente (o prácticamente) computable es muy limitada.

Estas consideraciones llevan a investigar modelos de iluminación no locales dentro del marco de los sistemas de rendering tradicionales, es decir, los sistemas scan-line. El estado del arte en el rendering scan-line de refracciones, sin embargo, está lejos de ser satisfactorio. Los modelos más usuales ignoran la geometría de la refracción, y realizan únicamente un modelo de la transparencia del objeto traslúcido. Esta transparencia se puede computar interpolando el color del pixel p en función del color del objeto traslúcido P_1 y del color del objeto (opaco) P_2 que se encuentra detrás: $I_f(p) = (1 - k)I_{P_1}(p) + kI_{P_2}(p)$, donde $0 \leq k \leq 1$ es el coeficiente de transmitancia del objeto traslúcido, normalmente dependiente de la longitud de onda. Este esquema es denominado *transparencia interpolada* en [33]. Otro esquema posible es utilizar *transparencia filtrada*, en la cual se considera que P_1 “filtra” el color de P_2 con un coeficiente O de transparencia: $I_f(p) = I_{P_1}(p) + O.k.I_{P_2}(p)$.

Claramente se observa que estos mecanismos son *ad hoc*, dado que —al margen de ignorar la geometría de la refracción— plantean una fórmula de interpolación que no proviene de un modelo de iluminación. Una ulterior mejora fue propuesta por Kay y Greenberg [56], donde el coeficiente de transmitancia k es una función no lineal de z_n , la componente z del normal. Kay y Greenberg sugirieron una función $k = k_{min} + [k_{max} - k_{min}].[1 - (1 - z_n)^m]$, donde k_{min} and k_{max} son las transparencias mínima y máxima respectivamente, y m es un coeficiente arbitrario, generalmente entre 2 y 3, donde un m mayor modela un objeto traslúcido de menor grosor (ver Figura 9.1(a)).

En [67] se propuso modelo de rendering scan-line de objetos traslúcidos con características superiores a otros modelos presentados, dado que no solo se computa un modelo de iluminación adecuado, sino que además se considera *la geometría* de la refracción. Básicamente, el modelo aproxima la forma de los objetos traslúcidos interpolando superficies esféricas de la expresión poligonal de los mismos (ver Figura 9.1(b)). Dicha aproximación permite considerar a cada objeto traslúcido como una *lente*, para la cual

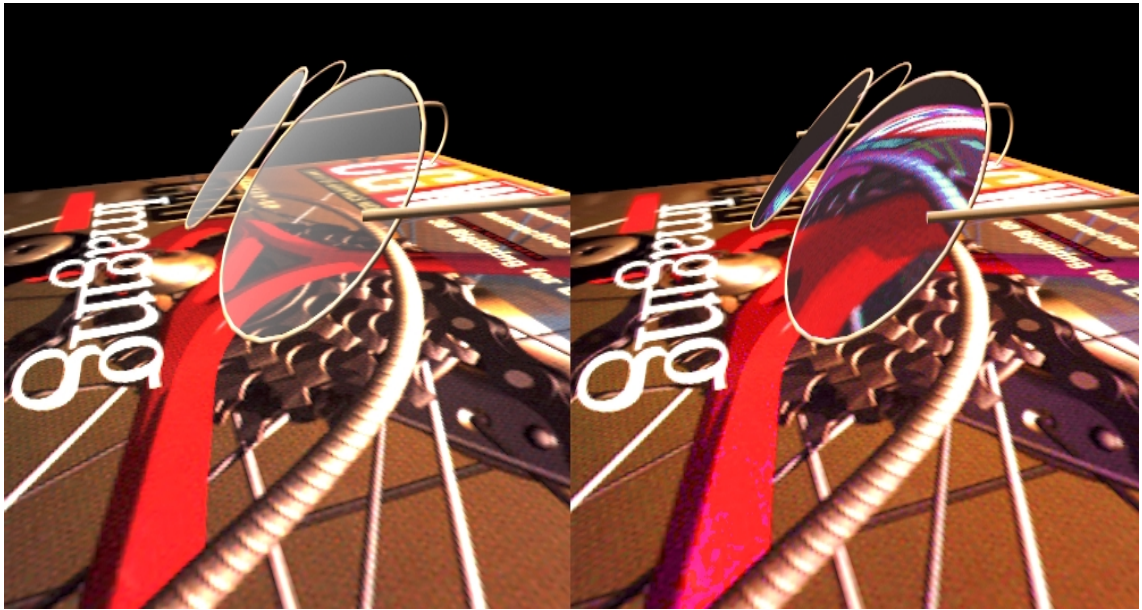


Figura 9.1 *Modelo de Kay-Greenberg vs. modelo de Patow.*

es sencillo encontrar una expresión analítica de la geometría de la refracción, expresión que puede simplificarse considerando la aproximación paraxial de los rayos de luz [52].

La determinación de los elementos refractados se efectúa utilizando mapas de entorno procesados por medio de dicha expresión analítica. El modelo de iluminación considerado en este caso extiende el modelo local de Phong para incorporar la componente refractada, por lo cual es más adecuado que el modelo de ray tracing dado que le da un tratamiento uniforme a las distintas fuentes de iluminación, sean locales o globales. Una ulterior simplificación, considerar la geometría de la refracción en una lente delgada, permite encontrar una expresión más simple. En particular, se demuestra que en este caso la geometría de la refracción es sencillamente obtenible a partir de la geometría de la reflexión, y que por lo tanto un preprocesamiento de un mapa de reflexiones estándar permite computar la refracción. Por último, se considera la representación de objetos complejos como series de lentes, lo cual constituye una primera aproximación muy económica (ver Figura 9.2).



Figura 9.2 Aproximación de objetos complejos como series de lentes.

9.2 Animación

Por animación en Computación Gráfica entendemos bastante más que modelar el movimiento de los objetos y la cámara dentro de la escena. Una animación por computadora normalmente involucra, además, cambios en la forma y color de los objetos, en la iluminación, en las texturas o en las estructuras. Actualmente es posible reconocer las grandes posibilidades de la animación por computadora en las publicidades y en los video juegos y aplicaciones interactivas. Sin embargo, las aplicaciones de la animación van más allá, dado que es posible encontrarla en sistemas educativos, simuladores de vuelo, usos industriales, visualización científica, y hasta en los cascos de realidad virtual.

Las técnicas básicas de animación por computadora se basan en la metodología convencional de los dibujos animados. Esencialmente se parte de lo que se denominan *key frames* o cuadros clave, donde cada cuadro es exactamente una de las tantas imágenes por segundo que se generan. Todo lo que sucede entre dos cuadros clave sucesivos puede interpolarse. Por lo tanto, dados dos cuadros clave sucesivos, y el tiempo que debe transcurrir entre ellos, es posible computar los cuadros intermedios por medio

de técnicas de interpolación. A esta operación se la denomina *in betweening* [61], (ver Figura 9.3).

Las técnicas que computan los estados intermedios de la escena reciben de los cuadros claves una descripción de cada elemento de la escena (objetos, iluminantes, posición de la cámara, etc.) por medio de una lista de parámetros, y su tarea consiste en calcular los valores de dicha lista para los cuadros intermedios, y luego graficar la escena con dichos valores. Esta interpolación no es necesariamente lineal, dado que los objetos pueden tener *leyes* de movimiento para conseguir determinados efectos. Por ejemplo, un objeto en caída libre describe una parábola en el tiempo, por lo que su ley de movimiento del primer cuadro clave al siguiente debe ser una parábola. Una aplicación importante de las curvas de Bézier es, entonces, la descripción de las leyes de movimiento en animación.

En los sistemas comerciales existe una estandarización de los parámetros que pueden animarse, y de las leyes que se utilizan. Nacen, de esa manera, los lenguajes para descripción de animaciones. Recomendamos al lector interesado que consulte el libro de N. Magnenat-Thalmann y D. Thalmann [61] o el Capítulo 21 del libro de Foley et. al. [34].

Las técnicas avanzadas de animación por computadora, sin embargo, no pueden quedarse en un nivel tan superficial de descripción. Por ejemplo, si en la animación de un objeto modelado con NURBS, la forma del mismo debe modificarse, lo lógico es modificar adecuadamente la posición de sus puntos de control cuadro por cuadro. Sin embargo, resulta imposible pensar que nuestro lenguaje de animación esté preparado para contemplar la interpolación de este tipo de parámetros.

Otros problema propios de la animación por computadora son la animación de objetos articulados y la animación de objetos no rígidos. Una técnica que ha ganado gran popularidad es la animación por medio de sistemas de partículas. Todos estos temas son de un nivel de complejidad considerable. Recomendamos al lector interesado que consulte los Capítulos 15, 16, 17, y 18 del libro de Watt y Watt [84].

9.3 Visualización Científica

Como ya dijéramos en la Introducción, la Visualización Científica significó todo un cambio de paradigma dentro de la Computación Gráfica, al buscar la representación gráfica de entidades abstractas, elaboradas a partir de conjuntos de datos. La Visualización Científica constituye actualmente una verdadera revolución en la metodología

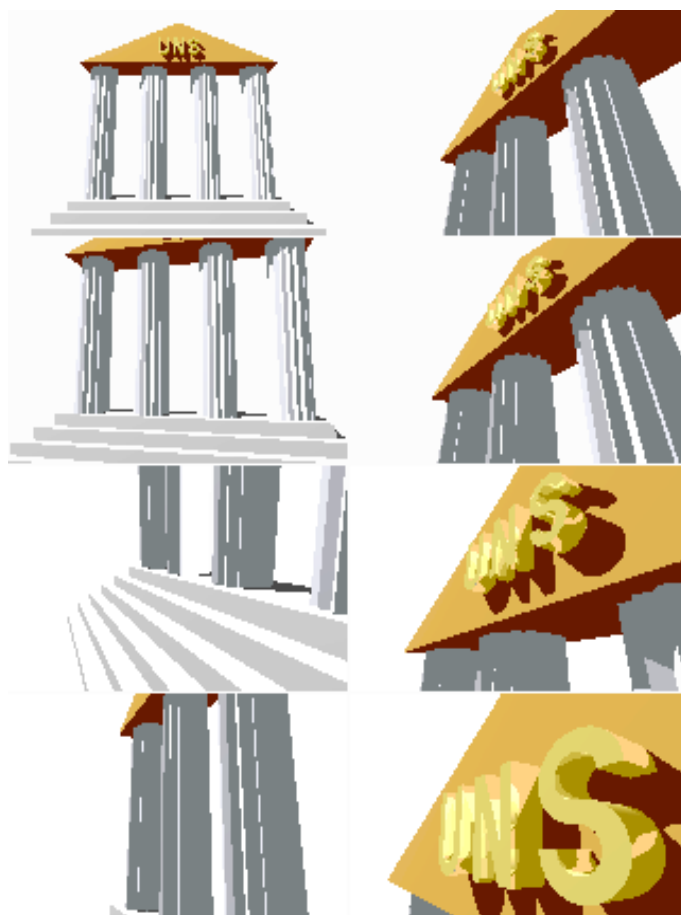


Figura 9.3 Algunos cuadros de una secuencia animada.

de investigación científica, tanto básica como aplicada, comparable al invento del microscopio o el telescopio. Con el tiempo, la comunidad que estudia estos temas se fue apartando de la Computación Gráfica tradicional, y actualmente constituye de hecho una disciplina aparte, con sus propios congresos y publicaciones. Sin embargo, dada la enorme relevancia del tema, no queremos dejar de considerarlo aquí.

En la esencia de la Visualización está la representación de cantidades enormes de datos (probablemente multidimensionales) por medio de alguna metáfora visual que transmita adecuadamente la información. Se utilizan los sistemas computacionales no para simular sino para *representar* enormes conjuntos de datos, apelando a la vasta capacidad de interpretación visual del cerebro. En este sentido, la Visualización Científica representa la culminación de las actuales posibilidades de los sistemas de computación gráfica, dado que exigen una capacidad enorme de procesamiento. Por ejemplo, para manipular una base de datos tridimensional de $1000 \times 1000 \times 1000$ se requieren 1000M de datos. Estos datos pueden provenir de sensores, como en el caso de tomógrafos o de satélites, o bien pueden provenir de tareas computacionales anteriores, como por ejemplo de simulaciones o de análisis por elemento finito. El resultado gráfico que se espera de la visualización de estos datos no es meramente *cuantitativo* -no se busca necesariamente la representación fiel de valores- sino *cualitativo* -se busca un entendimiento global de determinadas propiedades de los datos [24, 74].

La Visualización involucra el empleo de técnicas derivadas de la Computación Gráfica utilizadas para la representación de datos científicos de los tipos más diversos. Dentro de la investigación en Visualización Científica, la representación de datos volumétricos se destaca por las dificultades computacionales que plantea, pero al mismo tiempo concentra la mayor atención en la investigación actual, dado que es actualmente de gran utilidad en la investigación científica en temas tan diversos como en matemática, medicina, ciencias naturales e ingeniería [37, 53], y es utilizada para representar datos que pueden provenir de diversas fuentes.

Los volúmenes de datos pueden pensarse abstractamente como matrices tridimensionales, en los que cada celda contiene valores uni o multivaluados. Estos datos fueron eventualmente preprocesados para extraer y enfatizar adecuadamente las características intuitivamente adecuadas en una determinada aplicación y para un determinado propósito en su visualización (por ejemplo, destacar determinadas áreas en la representación visual de una tomografía). Una vez que los datos volumétricos están adecuadamente preparados para el rendering, el mismo procede según algoritmos de mayor o menor sofisticación. Las técnicas usuales de rendering de volúmenes están normalmente asociadas a una representación de los mismos en estructuras de celdas volumétricas o en *voxels*. De esa manera, un voxel en particular representa el factor de ocupación que el sólido posee en una determinada fracción del espacio tridimensional.

Una de las primeras técnicas de rendering de volúmenes [32] consiste en graficar por capas el volumen de datos. Normalmente el volumen de datos se hace coincidir con los ejes del sistema de coordenadas del mundo, de modo que el eje z (hacia donde mira el observador) coincida con uno de los ejes del volumen de datos. Planos perpendiculares a dicho eje son entonces procesados de adelante hacia atrás. El procesamiento es sencillo, consistiendo en una proyección paralela de los datos al buffer de pantalla, utilizando alguna técnica de *pseudocoloring* para asociar los valores a representar con colores de una paleta predeterminada (por ejemplo, asociar un determinado color a un determinado tejido). Cada voxel, en función de su valor, tiene a su vez una determinada transparencia, es decir que no es necesariamente opaco, permitiendo que se visualice parcialmente las partes del volumen que se encuentran detrás. La transparencia en cada dirección visual se computa acumulándola en un α -buffer de pantalla [15]. Para emular una proyección tridimensional, los datos de las capas se van desplazando una determinada distancia en x e y a medida que éstas son más distantes en el eje z . Esta técnica es bastante primaria, pero por esa misma razón es implementable directamente con hardware específico. Su mayor limitación consiste en que, al no existir un sólido propiamente dicho en ningún momento del procesamiento, no es posible una representación con realismo [28, 84], por ejemplo, la interacción con iluminantes o con otros objetos.

Otros métodos más sofisticados buscan extraer la *representación* de un objeto tridimensional a partir del volumen de datos. Una de las primeras técnicas [38] consiste en procesar capa por capa al volumen de datos, en función de un determinado valor umbral. De esa manera, es posible identificar en una capa dada aquellos voxels en los cuales ocurre una transición cercana al valor umbral. Dichos voxels conforman un contorno. Entre dos capas adyacentes, entonces, es posible vincular los contornos para determinar un esqueleto de polígonos. El conjunto de polígonos encontrado entre todas las capas procesadas de esta manera constituye una representación del sólido con una estructura “intermedia”, en este caso, una superficie. Esta estructura de polígonos permite la visualización de los datos originarios, y tiene la ventaja de ser una estructura “tradicional” en el sentido de la computación gráfica, es decir, es un conjunto de polígonos, el cual puede graficarse con los algoritmos usuales, utilizando cara oculta, sombreado, iluminación, etc. Sin embargo, esta técnica encuentra problemas cuando no es directo encontrar el esqueleto de polígonos entre dos capas sucesivas (por ejemplo si ocurren discontinuidades topológicas).

Otra solución, más estable con respecto a este tipo de problemas, es la denominada “marching cubes” [60], en la cual se clasifican los voxels que pertenecen a una superficie umbral. Un voxel pertenece a la superficie umbral si por lo menos uno de sus vértices está por debajo del valor umbral y por lo menos otro está por encima. En este caso, cada uno de los ocho vértices de un voxel puede asumir un valor por debajo o por encima del umbral. El total de todos los casos posibles es $2^8 = 256$, pero por consideraciones

de simetría se reducen a solo 14. Para cada uno de dichos casos es posible aproximar la superficie umbral con polígonos sencillos (normalmente triángulos) que cortan al voxel, y al mismo tiempo ubicar los voxels vecinos en los cuales dicha superficie debe continuar.

Una solución más completa (y también más compleja) para el rendering de volúmenes consiste en arrojar rayos desde el observador hacia el sólido, de una manera similar al ray tracing pero computando el comportamiento de la luz a través del volumen. Esta técnica, denominada ray casting, comienza por considerar que cada voxel es el modelo de un objeto físico, en el cual ocurre un fenómeno de interacción con la luz y con los rayos visuales provenientes de los demás voxels. Por lo tanto, es necesario establecer un modelo de iluminación que, a diferencia de los modelos tradicionales en Computación Gráfica como el de Phong, considere la interacción de la luz con una densidad volumétrica. Estos modelos fueron estudiados por Blinn [9] y por Kajiya [54, 55], llegando ambos a una formulación matemática similar.

Dada una densidad volumétrica $D(x, y, z)$ y un rayo visual \vec{v} que la atraviesa entre dos puntos t_1 a t_2 , consideraremos por un lado la iluminación acumulada en su interacción con una distribución de energía luminosa I que representa una determinada condición de iluminación, y por otro lado la densidad acumulada por el rayo desde que ingresa al sólido en t_1 . Sea entonces un punto t entre t_1 y t_2 . La iluminación que recibe dicho punto es la sumatoria de las intensidades de las fuentes luminosas puntuales. El modelo considera que la densidad volumétrica puede pensarse como una distribución gaussiana de partículas idealmente especulares. Los algoritmos basados en esta técnica, normalmente simplifican esta ecuación según ciertas consideraciones. Por ejemplo, si los rayos visuales son paraxiales, es decir, con pequeña desviación angular respecto del eje z , entonces la distancia de t_1 a t_2 es constante en todos los voxels, y por lo tanto la integral puede aproximarse con una productoria.

Otra forma de graficar un volumen de datos es el procesamiento *cell by cell* (por celdas). Esta técnica consiste en ir recorriendo la base de datos en una forma ordenada de adelante hacia atrás (según la posición del observador) y se va proyectando dato por dato, éste requiere un buffer con la información de los datos ya proyectados (color y transparencia acumulados). En arquitecturas computacionales complejas, como por ejemplo en las máquinas Silicon Graphics, el buffer de pantalla está pensado como para brindar soporte a este tipo de operaciones, es decir, se opta por la solución más natural del hardware dedicado. En máquinas PC, sin embargo, el buffer de pantalla no tiene capacidad para soportar estos cálculos intermedios, por lo que hay que recurrir a la memoria de propósito general (RAM), con la consiguiente complicación en la programación, y los tiempos de cómputo mayores.

El procesamiento por celdas puede pensarse en forma diferente si se considera que los datos están formando voxels o formando celdas. El voxel es la mínima cantidad de información en 3D y vendría a ser lo que es el pixel en 2D. En cambio, la celda está formada por 8 datos que serían los vértices, entonces se puede realizar un promedio o bien una interpolación (trilineal) para obtener la información dentro de la celda y de esta forma evitar que en la imagen se perciban los pequeños cubos que la componen, cuando la matriz de datos es pequeña. La resolución de los gráficos está dada por el tamaño de la matriz de datos. Las etapas más importantes de este proceso son:

- Preprocesamiento de datos, etapa que comprende la generación y acondicionamiento de la matriz de datos o buffer volumétrico (*V-Buffer*).
- Adecuación a las transformaciones de proyección. De acuerdo con el punto de observación se elige el orden en que se van a ir tomando los datos para su proyección y se toma el dato correspondiente.
- *Voxelización*, etapa que consiste en armar un voxel con una cierta ubicación en el espacio, de acuerdo a la posición que tiene el dato en la matriz.
- Mapeo de color y transparencia, es decir, cada voxel va a tener un color y una transparencia asignada, de acuerdo al valor numérico del dato de la matriz que ha sido tomado.
- Proyección de las caras de cada voxel que son visibles sobre la pantalla computando la contribución de color y transparencia que aporta el voxel sobre la imagen.

En la Figura 9.4 podemos ver el resultado de procesar matrices volumétricas de datos con este procedimiento.

Deseamos dedicar en este último Capítulo por lo menos un par de hojas a cada uno de los temas que consideramos de gran importancia. De esa manera, el lector interesado puede consultar la bibliografía recomendada. Un tratamiento más profundo de estos temas puede ser motivo para la segunda parte de este libro.

9.4 Modelos no Determinísticos y Fractales

Una de las razones más importantes de la gran difusión de las técnicas de modelado con objetos fractales reside en su capacidad para producir una representación económica

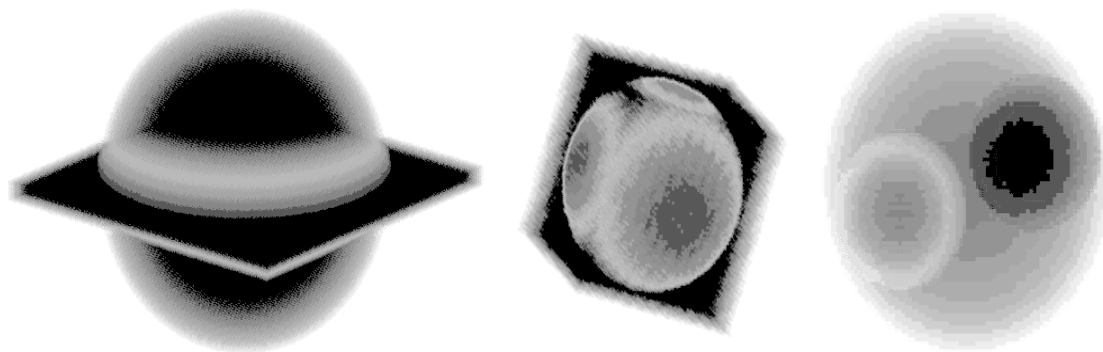


Figura 9.4 Matrices volumétricas de datos procesadas por celdas.

en tiempo y espacio de modelos o fenómenos naturales. El inconveniente clave de la geometría tradicional que se utiliza en la Computación Gráfica para representar objetos naturales es su falta de *simetría a escala*. Una costa podría describirse mediante segmentos, y si se tomara un número grande de ellos, se vería bastante real. Sin embargo, cuando nos acerquemos lo suficiente, esta costa revelaría su aspecto geométrico subyacente. Es por este motivo que los *objetos fractales* [62] están teniendo un uso creciente en la computación gráfica, especialmente aquellos basados en el modelo del ruido Browniano fraccional (fBm) [63].

Existen principalmente dos tipos de fractales: los determinísticos y los no determinísticos. Los primeros resultan generalmente de iterar sistemas de ecuaciones, y no incluyen al azar en su cálculo. Es decir, cada vez que se grafican, su aspecto es el mismo. Podemos citar muchos ejemplos notables, aunque el más popular es sin duda el conjunto de Mandelbrot (el mapa de complejos c cuyo conjunto de Juliá al iterar $z \leftarrow z^2 + c$ es conexo).

Los fractales no determinísticos, en cambio, utilizan explícitamente números aleatorios en determinadas etapas del cálculo. Por lo tanto, el resultado final es imprevisible. En la representación de fenómenos naturales, como por ejemplo relieves montañosos, costas, nubes, etc., suelen usarse los fractales no determinísticos. El cálculo de los mismos depende solamente de un parámetro denominado *dimensión fractal* D , que intenta reflejar el aspecto global del resultado. Los demás aspectos quedan librados a lo que el generador de números aleatorios entregue en cada paso.

Una de sus características interesantes es que satisfacen la propiedad de *autosimilitud*, o sea, se ven estadísticamente similares (esto es “parecidos a primera vista”) a

sí mismos a diferentes escalas. Un *zoom* a un objeto fractal no causará una pérdida de naturalidad de su aspecto, sino que, por el contrario, incorporará nuevos detalles no visibles a menor escala. La autosimilitud puede concebirse como una simetría a escala. Si el objeto es un fractal no determinístico, esta similitud se da en términos estadísticos, es decir, las propiedades geométricas del objeto tienen distribuciones semejantes a diferentes escalas. En algunos casos particulares pueden existir modelos matemáticos para determinados fenómenos naturales cuya fractalidad es un fenómeno emergente. El interés de la Computación Gráfica en los fractales, sin embargo, no es realizar una simulación exhaustiva de dichos fenómenos, sino en producir herramientas computacionales que los imiten adecuadamente en un tiempo razonable.

Un modelo matemático de gran utilidad para un gran conjunto de series de tiempo es el *movimiento Browniano fraccional* (fBm) [63]. La computación del fBm puede hacerse por medio de la transformada rápida de Fourier (FFT), pero en la práctica resultan más adecuados los métodos en el dominio tiempo. Las técnicas más comunes para efectuar dichos cálculos son

Cortes Independientes [68]: Es el primer método históricamente implementado para reproducir terrenos. Consiste en generar números aleatorios de importancia decreciente, los cuales son sumados a un arreglo en una cantidad de direcciones que sigue la proporción $\Delta V = \alpha(\Delta t)^{2-D}$. La *fase* de dichos cortes (la dirección de comienzo) es determinada aleatoriamente.

Cortes Secuenciales: Es una variación del anterior que tiene la ventaja de poder ser calculado en secuencia, sin requerir calcular todo el arreglo. Por ello es aplicable a la generación de números aleatorios que sigan una distribución fraccional. Consiste en tener un acumulador al que en cada tirada se le agrega una cantidad aleatoria. Esta cantidad tiene componentes proporcionales a períodos irracionales, cuyas amplitudes son inversamente proporcionales a los períodos. Una metáfora para entenderlo mejor es suponer que tenemos varios dados, algunos con números pequeños, otros medianos y otros grandes. En cada tirada se arrojan ciertos dados. La probabilidad de que un dado sea arrojado es inversamente proporcional a los números que figuran en él.

Desplazamiento Aleatorio del Punto Medio [35]: Este método es computacionalmente más eficiente que los anteriores, y algorítmicamente es más sencillo de entender. Se toma como comienzo y como final dos puntos dados $V(0)$ y $V(1)$. Se calcula el V promedio en el punto medio, y a ese valor se le agrega un ΔV proporcional al Δt según la ecuación ya vista. Luego, cada mitad es tratada recursivamente de forma similar.



Figura 9.5 Terrenos fractales computados con el algoritmo de desplazamiento del punto medio.

La extensión del algoritmo del punto medio a dos dimensiones puede aplicarse para generar una distribución bidimensional fractal de alturas. En este caso el punto medio tiene en cuenta las alturas de sus cuatro vecinos. Es decir, cada punto medio está en el centro de una cara, cuyos vértices son tenidos en cuenta para calcular su altura esperada. Esta altura es desplazada aleatoriamente en función de la escala. Con los puntos medios en el centro de cada cara es posible ahora calcular la altura en los puntos medios de cada arista. Estos dos pasos permiten pasar de una retícula de 1×1 y cuatro puntos a una retícula de 2×2 caras y nueve puntos. A cada cara de la retícula generada se le aplica recursivamente el mismo algoritmo.

La Figura 9.5 muestra dos mapas de alturas fractales desarrollados con el método de desplazamiento del punto medio. Estos resultados, además de permitir la simulación de terrenos, pueden utilizarse como mapas de texturas para otro tipo de objetos. Por ejemplo, en la Figura 5.17 se utilizaron fractales como mapa de desplazamientos en un algoritmo scan-line, y en la Figura 7.21 se utilizaron como mapa de normales en un algoritmo de ray tracing.

