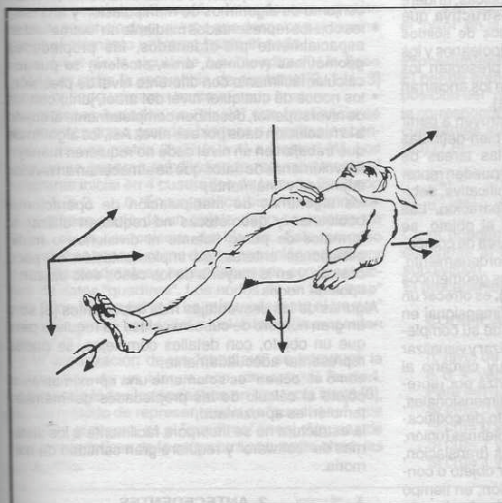


Visualización de modelos de sólidos basados en árboles octales ("Octrees")



RESUMEN

Un enfoque para el modelado de sólidos basado en la descomposición del espacio objeto en subespacios regulares, particularmente cubos, cuyo resultado es una estructura de árbol, es el "octree" (árbol octal).

Un objeto cualquiera (Universo) se descompone en 8 celdas, del mismo tamaño para cada una de las cuales se verifica si son o no homogéneas, es decir, si están completamente dentro o completamente fuera del objeto. Si esto ocurre, la descomposición termina; en caso contrario es heterogénea y por lo tanto, se subdivide en 8 subceldas. Cuando todas las celdas son homogéneas, para algún nivel de precisión, el proceso termina. La tarea de visualizar un "octree" puede hacerse de dos maneras básicas: frente-hacia-atrás y atrás-hacia-frente (en inglés: front-to-back y back-to-front, respectivamente).

El propósito de este artículo es presentar algunos enfoques básicos de visualización. En la primera parte se caracteriza el enfoque de codificación "octree", introduciendo algunos elementos de su antecesor, el "quadtree", y en la segunda se incluyen, en forma detallada, diferentes enfoques de visualización, incorporando además el tratamiento de superficies ocultas.

ABSTRACT

A geometric modeling technique called Octree Encoding is characterized based on the quadtree technique. Some display methods are presented including hidden surface removal.

□ Ana Esther Rodríguez
Título de postgrado y pregrado
Licenciada en informática
Universidad de Deusto España
Candidata a doctorado en informática
de la Universidad Politécnica de Madrid.

□ Marta Elena Millán, Ph.D.
Licenciada en Matemática-Física
Profesora Asociada
Departamento de Ciencias de la Computación
Universidad del Valle.

PALABRAS CLAVES

(KEYWORDS)

Octree, image Encoding Quad-tree, Object representation

1. INTRODUCCION

Dentro del modelado de sólidos se pueden identificar, entre otros, los siguientes tipos de sistemas [14]: los que se basan en el **instanciamiento de primitivas**, que definen un objeto con base en un tipo de forma y en un conjunto de valores de parámetros; los de **enumeración espacial**, que representan un objeto mediante una lista de las celdas espaciales cúbicas que este ocupa; los de **descomposición en celdas**, generalización del modelo anterior, en el que un objeto se representa por medio de un conjunto de celdas disjuntas que no son necesariamente cúbicas, ni idénticas; los basados en **geometría constructiva** que representan los objetos como conjuntos de sólidos primitivos conectados vía operadores booleanos y los de **representación de bordes**, que presentan los objetos por medio de las superficies que los encierran (planos, superficies cuadráticas, etcétera).

En estos sistemas, los objetos se construyen a partir de un número limitado de superficies bien-definidas matemáticamente o de primitivas y las tareas de manipulación y visualización de objetos pueden representar una carga computacional significativa, debido, básicamente, al proceso de comparación. Las primitivas, con las cuales se modela el objeto, se deben comparar unas con otras. Esta tarea de comparación usualmente requiere búsqueda u ordenamiento. El propósito fundamental de los modelos geométricos basados en **árboles octales** ("octree"), es ofrecer un esquema de modelado geométrico tridimensional en el que un objeto, independientemente de su complejidad, se pueda codificar, manipular, analizar y visualizar interactivamente en tiempo real o muy cercano al tiempo real [14]. El modelo se caracteriza por representar objetos 3-dimensionales y N-dimensionales, para cualquier resolución, en un formato de codificación común, trabajar con operaciones booleanas (unión, intersección y diferencia) y geométricas (translación, rotación, escalamiento) sobre cualquier objeto o conjunto de objetos y facilitar la visualización, en tiempo lineal, de cualquier número de objetos desde el punto de vista de un observador.

Esta representación describe un cuerpo sólido como un arreglo tridimensional que contiene información en donde se indica el material que se encuentra en cada punto del espacio [2]. Cuando una gran región del espacio contiene un solo material, entonces muchos elementos adyacentes del arreglo toman el mismo valor y por esta razón se puede utilizar un esquema de compresión.

Para Jackins et al. [11], un "octree" es una forma de ordenamiento espacial que permite hacer operaciones más económicas, en términos de memoria. El enfoque es una representación tridimensional de los métodos "quadtree".

De acuerdo con Yamaguchi et al. [25] y Samet [18] [21], un conjunto "octree" CO se define formalmente:

(1) $0 \in CO$

(2) $1 \in CO$

(3) $(O_{0,0,0}, O_{0,0,1}, O_{0,1,0}, O_{0,1,1}, O_{1,0,0}, O_{1,0,1}, O_{1,1,0}, O_{1,1,1}) \in CO$ si $O_{x,y,z} \in CO$ para $x,y,z \in \{0,1\}$

(4) CO no incluye ningún elemento que no esté construido aplicando las reglas 1 a 3 repetidamente. 0 y 1 son hojas y los otros son nodos "octree".

Algunas de las ventajas de esta representación [2], [14] son las siguientes:

- Se utiliza únicamente una forma primitiva: el cubo,
- cualquier objeto se puede representar con la precisión del cubo de menor tamaño,
- para todos los objetos, se utiliza solamente un único conjunto de algoritmos de manipulación y análisis,
- los objetos representados mediante un "octree" están espacialmente pre-ordenados, las propiedades geométricas (volumen, área, etcétera) se pueden calcular fácilmente con diferente nivel de precisión,
- los nodos de cualquier nivel del árbol, junto con los de nivel superior, describen completamente el objeto a la resolución dada por ese nivel. Así, los algoritmos que trabajan en un nivel dado no requieren manejar los volúmenes de datos que se almacenan en niveles inferiores y finalmente,
- los algoritmos de manipulación de operaciones booleanas y geométricas no requieren utilizar ni formatos de punto flotante ni divisiones o multiplicaciones enteras, su implementación es poco costosa y en la mayoría de los casos solo utilizan 3 tipos de nodos.

Algunas de las desventajas más importantes [2] son:

- un gran número de cubos o celdas se requiere para que un objeto, con detalles complejos, se pueda representar adecuadamente,
- como el "octree" es solamente una aproximación al objeto el cálculo de las propiedades geométricas también es aproximado,
- la estructura no se incorpora fácilmente a los sistemas de "software" y requiere gran cantidad de memoria.

2. ANTECEDENTES

La codificación "octree" es una extensión de la codificación bidimensional quadtree [3]. En la estructura "quadtree" un grupo de "pixels" forma una región homogénea, si todos son del mismo color. Una región homogénea se describe completamente por medio del valor del "pixel" que la ocupa. Por otra parte, una región heterogénea contiene una mezcla de "pixels" y está subdividida de manera que cada subregión que la compone se puede describir homogénea o heterogéneamente. Una codificación "quadtree" se genera cuando la subdivisión de regiones produce cuadrantes de la región original.

La estructura de datos "quadtree" es una grafo dirigido

para el cual cada nodo contiene una lista de cuatro elementos de datos correspondientes a los cuadrantes de una región en la imagen [3]. Cada elemento de datos tiene una marca que indica si el cuadrante es o no homogéneo. Cuando la marca indica que ese cuadrante particular es homogéneo, el resto de los elementos de datos toman como valor el del "pixel" que llena la región. En un cuadrante heterogéneo un elemento de datos contiene un puntero a otro nodo que subdivide la región. Un elemento de datos heterogéneo de un nodo apunta a un descendiente del nodo. Cada nodo "quadtree" puede además tener 0, 1, 2, 3 ó 4 descendientes.

El **Elemento Universo**, es un elemento de datos que describe completamente al objeto y que tiene nivel 0. El nivel de un nodo en el árbol se determina por el número de subdivisiones anidadas desde éste al elemento universo. Los nodos de un "quadtree" tienen asociada una única **dirección de nodo**, que consiste en una secuencia de punteros que describen el camino desde el elemento universo hasta el nodo.

En el árbol de la siguiente (un "quadtree"), la línea gruesa muestra la secuencia de punteros [2, 1, 1, 3] desde el nodo universo hasta el nodo final.

Las regiones basadas en "pixels" con frecuencia se representan como grandes bloques o cuadrantes centrados en una matriz $2^n \times 2^n$ ($n \geq 1$) o "raster" [1]. Los bloques se generan subdividiendo recursivamente el cuadrante inicial en 4 cuadrantes de tamaño $2^{n-1} \times 2^{n-1}$ con $g = n-1, n-2, \dots, 0$. En un gráfico binario, cuando un bloque hace parte de un objeto dado se colorea de negro, si no, se colorea de blanco. Este proceso recursivo se conoce como descomposición regular por medio de cuadrantes y está relacionado con la estructura de datos "quadtree". Los nodos de un "quadtree" o no tiene hijos o tiene 4. La raíz representa el "raster" completo y las hojas del nivel inferior representan "pixels" [1].

En la linearización de un "quadtree" se introduce la localización de un bloque con respecto al "raster", codificando todos los ancestros en una sola clave [6], [8]. Un método de representación de "quadtrees", que no utiliza punteros, es el "quadtree" lineal. Algunas de las características de la estructura son, entre otras, las siguientes:

- solo se almacenan nodos negros,
- en la codificación de los nodos se incorporan propiedades de adyacencia en las direcciones NW, NE, SW, SE (nor-oeste, nor-este, sur-oeste, sur-este)
- en la representación de los nodos se introduce, implícitamente, el camino desde la raíz hasta el nodo y
- se eliminan punteros.

2.1 Codificación de nodos negros

El cuadrante NW se codifica con 0, el NE con 1, SW con 2 y SE con 3.

Cada "pixel" negro se codifica con un entero en base 4, donde cada dígito representa la subdivisión del cuadrante desde el cual se origina. El orden de izquierda a derecha de los dígitos refleja el orden de subdivisión de mayor a menor.

Los dígitos tienen un peso 4^{n-h} , $1 \leq h \leq n$, identificando el cuadrante al que pertenece el "pixel" en la h-ésima subdivisión.

Procedimiento de codificación

El par de enteros (I, J) $I, J = 0, 1, \dots, 2^n - 1$ identifica la posición del "pixel" en el "array" $2^n \times 2^n$. A cada par (I, J) le corresponde un entero K (en base 4) que indica los cuadrantes sucesivos a los que pertenece el "pixel" (I, J) [7,8].

$$I = C_{n-1} 2^{n-1} + C_{n-2} 2^{n-2} + \dots + C_0$$

$$J = d_{n-1} 2^{n-1} + d_{n-2} 2^{n-2} + \dots + d_0$$

Ci, di pueden tomar los valores 0 ó 1 e indican el cuadrante al que pertenece (I, J) en cada nivel de subdivisión.

El procedimiento para calcular k es el siguiente:

CODIFICAR (n, c, d, k)

n: nivel de descomposición

c, d, k: vectores enteros

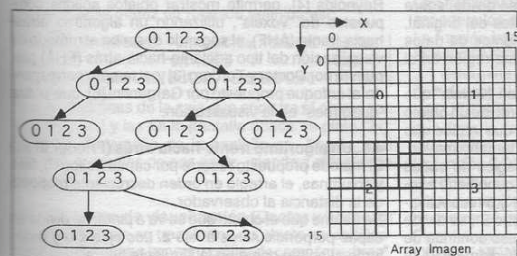
Para $l=n-1$ hasta 0, incremento = -1 hacer:

Si $c(l) = 0$ y $d(l) = 0$ entonces $k(l) = 0$

Si $c(l) = 0$ y $d(l) = 1$ entonces $k(l) = 1$

Si $c(l) = 1$ y $d(l) = 0$ entonces $k(l) = 2$

Si $c(l) = 1$ y $d(l) = 1$ entonces $k(l) = 3$



"Quadtree" lineales y regulares.

001	100	101	110	111
003	012	013		113
020				131
022				133
201				311
203				313
230		321	331	
223	232	233	322	332
				333

Región etiquetada con códigos en base 4

Después de codificar cada "pixel" negro, éstos se ordenan y se almacenan en una lista que se puede condensar en aquellos casos en los que cuatro "pixels" tienen la misma representación, excepto el último dígito. En este caso ese último dígito se reemplaza por una marca especial.

Para el ejemplo de la figura el "quadtree" lineal correspondiente es (001, 003, 012, 013, 02X, 03X, 1XX, 201, 203, 21X, 223, 23X, 30X, 31X, 320, 321, 322, 33X). Cualquier región basada en "pixels" formada de $2^n \times 2^n$ bloques negros se puede representar por medio de un "quadtree" lineal.

3. OBJETOS Y "OCTREES"

Codificación "octree"

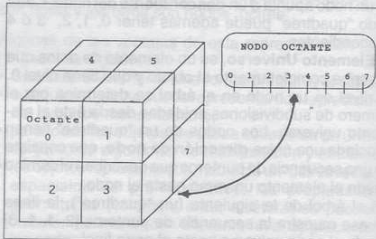
Este esquema es similar a los enfoques de enumeración espacial y descomposición en celdas [14]. Sin embargo, desde el punto de vista de almacenamiento, los datos se almacenan en una estructura jerárquica de árbol, en la cual los nodos representan cubos disjuntos decreciendo exponencialmente de tamaño. Cada nodo en el árbol, corresponde a una región del universo y tiene uno o más valores que definen la región. Un nodo es terminal u hoja si su valor describe completamente la región. Si no, el nodo apunta a 8 hijos que representan 8 subregiones u octantes del nodo padre.

Una de las ventajas adicionales, a las enunciadas en la introducción, es que las operaciones de visualización de caras ocultas y la detección de interferencias son de complejidad lineal porque todos los objetos se mantienen preordenados espacialmente [14]. Si se recorre el árbol, en una secuencia adecuada y las regiones del espacio se visitan en una dirección uniforme, los algoritmos de visualización de caras ocultas no requieren búsqueda, ni ordenamiento. Los árboles que representan los objetos que se van a visualizar, se recorren simplemente en un orden específico, dependiendo del punto de vista del observador.

En este caso, el concepto de codificación "quadtree" se extiende directamente a tres dimensiones (codificación "octree") [3], en el que se utilizan regiones del espacio, en lugar de áreas. Por consiguiente, una región del espacio es homogénea cuando todos los "voxels" (elementos volumen) son del mismo material o heterogénea en el caso contrario, para el cual ésta se divide, sobre los ejes primarios para producir octantes del original. Los nodos octantes contienen 8 elementos de datos que corresponden a los octantes de una región en el espacio.

Las regiones vacías se representan con "voxels" que contienen el valor especial, vacío. El borde de un objeto es un límite entre dos tipos de material que impide que "voxels" adyacentes formen cubos de tamaño mayor. Los objetos en consideración son arreglos de cubos unitarios [12], $2^n \times 2^n \times 2^n$ en donde cada cubo tiene asociada una etiqueta: LLENO o VACÍO (1 y 0 respectivamente). Un objeto ordenado, de diámetro 2^n , se divide en 8 cubos de diámetro 2^{n-1} . Este proceso continúa de manera que en cada paso un arreglo de diámetro 2^n se divide en 8 arreglos en forma de cubos de diámetro

2^{n-1} [11] [12]. La región cúbica que se obtiene de esta forma es un **OCTANTE**. Usando "octrees", un objeto se divide en octantes, cada uno de los cuales, a su vez, se dividen en octantes, hasta que los cubos dentro de cada octante tengan el mismo valor de etiqueta. Un octante tiene asociada una etiqueta con valor: LLENO (L) si todos los cubos unitarios que se encuentran dentro de él tienen etiqueta L o VACÍO (V), si todos los cubos unitarios que están dentro de él tienen como etiqueta V o MEZCLA (M), en cualquier otro caso.



Un objeto que se representa por medio de un "octree", es un árbol cuyos nodos o son hojas o tienen 8 hijos. El nodo raíz corresponde al objeto completo, los hijos del nodo raíz corresponden a los octantes del arreglo objeto y el proceso de división continúa hasta que los octantes alcanzan etiquetas L o V. Los hijos se enumeran 0, 1, 2, 3, 4, 5, 6, 7 y cada uno de los nodos tiene una etiqueta que, a su vez, es la etiqueta que le corresponde al octante. En un "octree" la etiqueta V representa transparente o BLANCO y L representa NEGRO.

4. VISUALIZACIÓN DE "OCTREES"

Los algoritmos de visualización transforman una representación "octree" de un objeto en una imagen o gráfico. De acuerdo con Carlbom [2], en los algoritmos de visualización se puede recorrer un "octree" de **atrás-hacia-frente** o de **frente-hacia-atrás**. En esta parte se presentan algunos métodos de visualización. Un primer método, propuesto por Frieder, Gordon y Reynolds [4], permite mostrar objetos sólidos compuestos de "voxels", utilizando un algoritmo atrás-hacia-frente (AHF), el segundo describe el método de visualización del tipo adelante-hacia-atrás (FHA) propuesto por Doctor y Torborg [3] y el tercero corresponde al enfoque propuesto por Gargantini [6], que utiliza "quadrees" en la visualización.

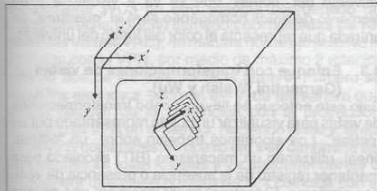
4.1. Un algoritmo frente-hacia-atrás (Frieder et al.)

El método propuesto recorre por capas ("slices"), filas y columnas, el arreglo en orden decreciente respecto de la distancia al observador.

Se asume que el objeto que se va a pintar se divide en capas perpendiculares al eje z. Los ejes x e y están junto al objeto como en la siguiente figura.

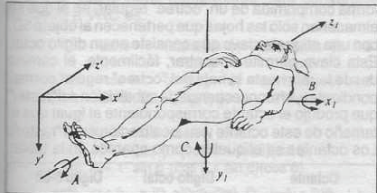
El usuario especifica la orientación del objeto por

medio de tres ángulos A,B,C, que son el resultado de rotaciones sobre los ejes x_1, y_1, z_1 . En aplicaciones médicas estos ejes coinciden con la posición de un paciente cuando se le hace una entomografía.



Espacio Objeto (x, y, z) y Espacio Imagen (x', y', z')

El ángulo A permite girar al paciente sobre el eje longitudinal z_1 , B es la inclinación sobre el eje x_1 y C es la rotación sobre el eje y_1 . Debido a que los datos, en este tipo de aplicaciones, se almacenan por capas y dentro de cada capa por filas, los "voxels" se recorren de manera que el bucle más externo es el de las capas (dirección z_1), dentro de cada capa se define otro bucle externo para las filas (dirección y_1) y un bucle más interno se define en la dirección x_1 . Las coordenadas de un "voxel" en el Espacio Objeto (EO) corresponden en el Espacio Imagen (EI) a:



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = T \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} \quad \text{donde} \quad \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

donde T es la matriz de rotación definida así:

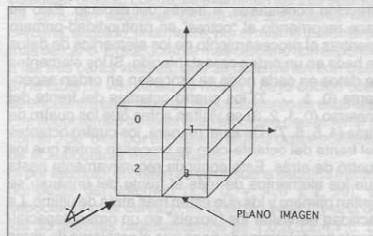
$$\begin{bmatrix} \cos A \cos C + \sin A \sin B \sin C & -\sin A \cos C + \cos A \sin B \sin C & \cos B \sin C \\ \sin A \cos C & \cos A \cos C & -\sin B \\ -\cos A \sin C + \sin A \sin B \cos C & \sin A \sin C + \cos A \sin B \cos C & \cos B \cos C \end{bmatrix}$$

Las dimensiones de la caja que encierra el objeto son $[2x_s, 2y_s, 2z_s]$ y las de la pantalla donde se pinta $[2x_p, 2y_p, 2z_p]$. El tamaño de la pantalla es $2x_p \times 2y_p$ y el número de niveles de gris es $2z_p$, siendo s el factor de escala.

4.2. Algoritmo de visualización (Doctor y Torborg)

En este enfoque se transforma un objeto en codificación "octree" en una imagen codificada como "quadtree". La imagen que genera representa una proyección no-

perspectiva del arreglo 3-D que emula un observador distante que está viendo directamente la cara frontal del universo. Desde este punto de vista, cada "pixel" en la imagen está alineado con una cadena de "voxels" en el arreglo espacial. Adicionalmente, cada cuadrante del "quadtree" está alineado con un par de octantes, uno directamente en frente del otro. Una visualización perspectiva se obtiene llevando a cabo una transformación geométrica sobre un objeto "octree", antes de iniciar el proceso de visualización.



El problema de visualización de objetos se divide en problemas simples: el de eliminación de caras ocultas en el que se identifican los "voxels" visibles con base en un punto de vista, el problema de sombras, en el que la orientación de las caras determina la cantidad de iluminación reflejada al observador y el de semitransparencia que no se detalla aquí.

4.2.1. Desarrollo del algoritmo

El algoritmo utiliza un conjunto de funciones para relacionar la base de datos que almacena, la estructura de árbol y la aplicación.

La función SON (OCTREE.DATA.ELEMENT, INDEX) se utiliza para acceder a los descendientes del nodo OCTREE.DATA.ELEMENT, en el "octree" del octante especificado por INDEX. Si la región es homogénea, cada uno de los descendientes tiene el mismo valor del elemento de dato dado. En otro caso, se utiliza un puntero de OCTREE.DATA.ELEMENT, para ir a un nodo particular.

Una función similar

QSON(QUADTREE.DATA.ELEMENT, INDEX)

accede a los descendientes de nodos "quadtree".

La función booleana

HOMOGENEOUS(DATA.ELEMENT.1, DATA.ELEMENT.2

...) determina si la región o regiones representadas

por el (los) elemento(s) de datos son homogéneas.

VALUE(DATA.ELEMENT) devuelve el valor del dato que llena la región descrita por DATA.ELEMENT. Si esta región es heterogénea el valor es indefinido.

La función COLOR(VALUE) es una tabla que contiene el color de cada tipo de material a visualizar. VALUE para una región "octree" sirve de índice en la tabla para encontrar el color de la región descrita.

Finalmente,

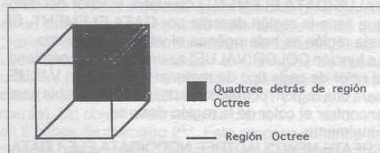
CREATE.NEW.QUADTREE.NODO(DATA.ELE.1, DATA.ELE.2, DATA.ELE.3, DATA.ELE.4) toma como entra-

da 4 elementos de datos que representan los cuadrantes de un área en la imagen. Si el área descrita por estos es heterogénea se crea un nuevo nodo "quadtree", que se llena con estos elementos y un puntero al nuevo nodo. En otro caso, se devuelve el elemento de datos homogéneo correspondiente.

4.2.2. Eliminación de caras ocultas

Un "octree" es una estructura que facilita recorrer un árbol de manera que los nodos se visiten en una dirección consistente, a través del espacio. Esto se hace recorriendo el "octree" en profundidad-primero mientras el procesamiento de los elementos de datos se hace en un orden preestablecido. Si los elementos de datos en cada nodo se procesan en orden ascendente (0, 1, ..., 7) los cuatro octantes del frente del universo (0, 1, 2, 3) se visitan antes que los cuatro de atrás (4, 5, 6, 7). De igual manera, los cuatro octantes del frente del octante-cero se procesan antes que los cuatro de atrás. Esto continúa recursivamente hasta que los elementos de más al frente del universo se visitan primero y los que están más atrás de último. La facilidad de visitar los "voxels" en un orden espacialmente predeterminado hace que el problema de caras ocultas sea sencillo de resolver.

El conjunto de "voxels" visibles se extrae de un "octree" de entrada a través del procedimiento recursivo que transforma el "octree" en un "quadtree", que representa la imagen directamente en el frente de la región "octree". El resto del "octree" permanece sin cambios. El procedimiento trabaja recursivamente así: Si el "octree" representa una región homogénea, el "quadtree" está siempre definido. Si la región "octree" no es vacía, entonces, la imagen última se oscurece y se reimprime. El nuevo "quadtree" tiene el color correspondiente al material que se reimprime. Si la región "octree" es vacía, el "quadtree" se devuelve sin cambios. Si es heterogénea, el "quadtree" es indefinido y se crea un nuevo nodo "quadtree". El valor de cada cuadrante del nuevo nodo "quadtree" está determinado por el par de octantes alineados con ese cuadrante. Los octantes 0 y 4 del "octree", determinan el resultado del cuadrante 0 del "quadtree". Similarmente, los octantes 1 y 5 del "octree" determinan el resultado del cuadrante 1 en la imagen y así para los otros dos cuadrantes. Como el valor de cada elemento de datos "quadtree" depende de dos octantes del "octree", se hacen dos llamadas al procedimiento por cada elemento de datos "quadtree" del nuevo nodo. Los octantes de atrás (4, 5, 6, 7) se procesan primero que los del frente (0, 1, 2, 3) para todos los cuatro cuadrantes.



Parámetros de entrada al procedimiento

Después de que ambos, los del frente y los de atrás, han sido procesados el "quadtree" mantiene el resultado visible de la región "octree".

Este procedimiento se invoca pasando el elemento universo del "octree" que se va a visualizar y un elemento de datos homogéneo para el "quadtree" de entrada que representa el color del fondo del universo.

4.3. Enfoque con transformaciones de vistas (Gargantini, Walsh y Wu).

Bajo este enfoque se llevan a cabo transformaciones de vistas para visualizar un objeto representado por un "octree". Los algoritmos trabajan sobre un "octree" lineal, utilizando un mecanismo (BIT) asociado para mantener registro de la ausencia o presencia de vecinos similares. El "octree" lineal incorpora una prueba de visibilidad ordenada en profundidad para los "voxels", mientras que el mecanismo BIT incorpora información sobre visibilidad de las caras de cada "voxel".

4.3.1. Definiciones

Un "raster" 3D, es un "array" de "voxels" $2^x \times 2^y \times 2^z$ donde r es la resolución. Un gráfico tridimensional es un "raster" 3D en el que cada "voxel" tiene asignado un atributo prioritario (nivel de gris, color, textura etcétera). El conjunto de "voxels" con este atributo es el fondo y los demás son el objeto. Cuando hay grandes bloques de "voxels" con el mismo atributo se representan por medio de un "octree". Un "octree" lineal es una forma comprimida de un "octree" regular, en el que se almacenan solo las hojas que pertenecen al objeto 3D, con una etiqueta clave que consiste en un dígito octal. Esta clave permite encontrar, fácilmente, el camino desde la raíz hasta la hoja del "octree" regular correspondiente y, en consecuencia, la subdivisión del "raster" que produjo el octante correspondiente al igual que el tamaño de este octante y su localización en el "raster". Los octantes se etiquetan como aparece en la tabla:

Octante	Dígito octal	Dígito 3-bit
atrás-sur-oeste	0	000
atrás-sur-este	1	001
atrás-norte-oeste	2	010
atrás-norte-este	3	011
delante-sur-oeste	4	100
delante-sur-este	5	101
delante-norte-oeste	6	110
delante-norte-este	7	111

Etiquetamiento de octantes

El "octree" lineal de un objeto es un vector ordenado de códigos octales. N es el número de nodos en el "octree" lineal de un objeto.

Los **vecinos** de un "voxel" Q son aquellos "voxels" con los que Q comparte una cara, en cualquiera de las seis principales direcciones (sur, norte, oeste, frente, este, atrás). El borde de un objeto tridimensional es el conjunto de "voxels" del objeto que tienen al menos un vecino en el fondo o están en el borde del "raster".

4.3.2. Transformación de coordenadas 3D

El objeto se localiza originalmente en un sistema de

coordenadas de mano-derecha (x,y,z), llamado **coordenadas del mundo (WC3)**, y la dirección desde el origen de WC3 al observador está dada por el vector (x_v,y_v,z_v), llamado **plano normal (VPN)**. El propósito es convertir WC3 en otro sistema coordenado de mano-derecha (U,V,N) llamado **coordenadas de referencia visual (VRC)**, de manera que el eje positivo N coincida con el VPN, por medio de máximo 2 rotaciones de dos de los ejes sobre el tercero. Primero se rotan los ejes x y z sobre el eje y, de tal manera que el eje positivo de la z coincida con la proyección (x_v,0,z_v) del VPN en el plano z, haciendo el eje y, el eje z y el VPN coplanarios. El nuevo eje x se denomina eje U. Después se rotan los ejes y y z alrededor del eje U de manera que el eje positivo de la z coincida con el VPN (que está ahora en el plano y-z).

Hasta ahora se han rotado los ejes alrededor del origen, pero el origen está, convencionalmente, localizado en una esquina del "raster", y se quiere rotar el objeto alrededor del centro del "raster" para que éste permanezca en el "raster".

Para hacer esto, primero se trasladan los ejes para que el origen esté en el centro, se ejecutan las rotaciones y finalmente se trasladan de nuevo los ejes a la posición original de manera que el WC3 y el VRC sean idénticos para el centro del "raster". Para un "raster" 2^x2^x2 el centro tiene coordenadas (x^c,y^c,z^c) y la matriz "viewing" llamada VIEW-MATRIX, por la cual cualquier vector debe multiplicarse para convertirse de WC3 a VRC es:

$$\text{VIEW-MATRIX} = \begin{bmatrix} \sin \theta & -(\cos \theta)(\sin \phi) & (\cos \theta)(\cos \phi) & 0 \\ 0 & \cos \phi & \sin \phi & 0 \\ -\cos \theta & -(\sin \theta)(\sin \phi) & (\sin \theta)(\cos \phi) & 0 \\ 2^{a-1}g(\theta, \phi) & 2^{a-1}g(\theta, \phi) & 2^{a-1}h(\theta, \phi) & 1 \end{bmatrix}$$

donde: $f(\theta, \phi) = 1 - \sin \theta + \cos \theta$
 $g(\theta, \phi) = 1 - \cos \phi + (\cos \theta + \sin \theta)(\sin \phi)$
 $h(\theta, \phi) = 1 - \sin \phi - (\cos \theta + \sin \theta)(\cos \phi)$

$$\sin \theta = z_c/D_c, \cos \theta = x_c/D_c, (0 \leq \theta < 2\pi), D_c = (x_c^2 + z_c^2)^{1/2}$$

$$\sin \phi = y_c/D_c, \cos \phi = D_y/D_c, (-\pi/2 \leq \phi \leq \pi/2), D_c = (x_c^2 + y_c^2 + z_c^2)^{1/2}$$

Los nuevos ejes x,y,z se llaman U,V y N respectivamente.

4.3.3. Caras ocultas

Una de las ventajas de la representación de objetos 3D, por medio de octantes, es que la mayoría de los problemas de visibilidad se resuelven simplemente viendo las coordenadas del VPN y el código octal. De nuevo sea VPN=(x_v,y_v,z_v) y supongamos que x_v,y_v,z_v ≠ 0. De esta manera, exactamente 3 caras de un cubo son visibles como se muestra en la tabla siguiente:

Visibilidad de un nodo respecto de las coordenadas VNP

VNP	Caras Visibles
x _v > 0 (> 0)	oeste (este)
y _v < 0 (> 0)	sur (norte)
z _v < 0 (> 0)	atrás (frente)

Si exactamente una coordenada es 0, entonces exactamente dos caras son visibles; si dos coordenadas son 0 exactamente una cara es visible. Se puede definir un orden de prioridad (basado en la codificación de "octrees" lineales) permitiendo establecer, recursivamente, el orden en el cual los suboctantes de un octante dado se deben considerar en la forma atrás-hacia-frente. El suboctante más lejano se examina primero, seguido por sus tres vecinos de cara (en cualquier orden). Luego, se examinan los tres vecinos de cara (en cualquier orden) del suboctante más cercano, seguido, finalmente, por el mismo suboctante más cercano. El suboctante más cercano corresponde a las tres caras visibles, que se determinan con base en la tabla de visibilidad de nodos y cuyo código octal se encuentra en la tabla de etiquetamiento de octantes. Los suboctantes se examinan en orden decreciente al número de "bits" por el cual sus dígitos octales difieren de el suboctante más cercano.

Si por ejemplo, el objeto se ve desde la esquina fronto-nor-este del volumen (x_v < 0, y_v > 0 y z_v > 0), el suboctante más cercano tiene dígito octal 6 y uno de los 36 posibles ordenamientos es 1,0,3,5,2,4,7,6. El orden que se adopta para todos los 8 casos se describe en la tabla siguiente:

Orden de prioridad para etiquetamiento de octantes de acuerdo al VPN=(x_v,y_v,z_v)

z _v	y _v	x _v	Dígito octal	Orden
<0	<0	<0	0	7,6,5,3,4,2,1,0
<0	<0	>0	1	6,7,4,2,5,3,0,1
<0	>0	<0	2	5,4,7,1,6,0,3,2
<0	>0	>0	3	4,5,6,0,7,1,2,3
>0	<0	<0	4	3,2,1,7,0,6,5,4
>0	<0	>0	5	2,3,0,6,1,7,4,5
>0	>0	<0	6	1,0,3,5,2,4,7,6
>0	>0	>0	7	0,1,2,4,3,5,6,7

4.3.4. El Procedimiento

Un primer algoritmo, divide recursivamente el "octree" lineal en octantes, los examina en orden de prioridad y cuando encuentra un suboctante, que es un solo nodo, éste se incluye en un "pipeline" de visualización. Este algoritmo determina todas las caras visibles de un nodo. Un enfoque más económico que se desarrolló para crear un "pipeline" de visualización, se basa en el hecho de que cuando una cara de un nodo objeto se comparte con el fondo, o con un nodo de color diferente, se puede detectar automáticamente, mientras se ejecuta el algoritmo de detección del borde 3D para encontrar los "voxels" superficie.

El algoritmo para detectar los bordes o límites de un objeto, entrega todos los "voxels" superficie como un conjunto de códigos octales. Para indicar cuales caras de un "voxel" superficie comparten con el fondo, el algoritmo asigna al nodo "voxel" 6 "bits" de bloque-de-fondo, uno por cada una de las seis direcciones principales, denotadas por E,S,B,W,N,F. Utiliza, además, un "bit" adicional, bloque-de-fondo, para un

"voxel" que puede tomar valor 1, si éste comparte su cara este con el fondo (o si está en el borde "raster" este) y 0 en caso contrario. Otro algoritmo posterior determina el nuevo "pipeline" de visualización en donde aparecen aquellos "voxels" que tienen al menos una cara visible con un 1 en el "bit" de bloque-de-fondo; los "voxels" superficie o cara se procesan en el mismo orden en que se procesan los nodos en el primer algoritmo. Finalmente, después de calcular el "pipeline" de visualización se determinan las coordenadas de la pantalla que corresponden a los vértices proyectados.

5. BIBLIOGRAFIA

- [1] Atkinson, Gargantini and Walsh, Filling by Quadrants or Octants. Computer Vision, Graphics and Image Processing, 33, 1986.
- [2] Carlborn, I, Chakravarty, I, Vanderschel D, A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects. IEEE CG&A, April 1985.
- [3] Doctor, L, Torborg, J, Display Techniques for Octree-Encoded Objects. IEEE CG&A July 1981.
- [4] Frieder, G, Gordon, D, Reynolds A, Back-to Front Display of voxel-Based Objects, IEEE CG&A, January 1985.
- [5] Fujimoto, A, Takayuki T, Kansei I, ARTS: Accelerated Ray-Tracing System. IEEE CG&A April 1986.
- [6] Gargantini, T.R, Walsh and Wu O.L, Viewing Transformations of voxel- Based Objects via Linear octrees. IEEE CG&A, October 1986.
- [7] Gargantini, I, Linear Octrees for Fast Processing of Three-Dimensional Objects. Computer Graphics and Image Processing, Vol. 20, Nº 4, Dec, 1982.
- [8] Gargantini, I, An Effective Way to represent

quadrees Communications of the ACM, Vol, 25, Nº 12, Dec 1982.

- [9] Glassner, A, An Introduction to Ray tracing. Edited by Andrew Glassner, Academic Press, 1990.
- [10] Glassner, A, Space Subdivision for Fast Ray Tracing. IEEE CG&A, October 1984.
- [11] Jackins, Ch, Tanimoto, S, Quad-trees, Oct-trees and K-trees: A Generalized Approach to Recursive Decomposition of Euclidean Space. IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI-5, 5 September 1983.
- [12] Jackins, C, Tanimoto, S, Oct-tress and their Use in Representing Three- Dimensional Objects. Computer Graphics and Image Processing Vol. 14, Nº 3 Nov. 1980.
- [13] Mantyla, M, An introduction to Solid Model
- [14] Meagher D, Geometric Modeling Using octree Encoding. Jan 1984

AUTORES

Ana Esther Rodríguez es candidata a doctorado en informática en la Universidad Politécnica de Madrid (España). Actualmente trabaja en el área de Bases de datos del proyecto de Política Agraria Comunitaria de la Unión Europea.



Marta Elena Millán, Doctor en Informática de la Universidad Politécnica de Madrid es profesora del Departamento de Ciencias de la Computación de la Universidad del Valle.