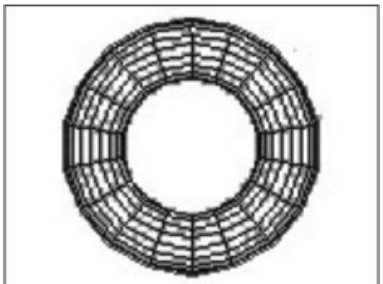
2. Teoría de los OCTREES

Los métodos de **descomposición espacial** se basan en formar modelos a partir de pequeños bloques, cuyas dimensiones dan la resolución del sólido, por lo tanto son métodos con gran flexibilidad. Aquí un sólido es descompuesto en un conjunto de elementos adyacentes que no se intersectan con primitivas que pueden variar de tipo, tamaño, posición y orientación. Según la clase del objeto base o primitiva y la forma de combinar este elemento ha generado diversas técnicas como:

- la descomposición de celdas,
- la enumeración espacial
- y los árboles octales



Módelos de descomposición

Figura 2.1

2.1 El Voxel

Un objeto es un conjunto continuo de puntos, sin embargo no podemos listar todos los puntos que pertenecen a un sólido, ya que son infinitos, podemos listar en pequeñas partes que sean contenidos dentro del sólido, sea de manera parcial o total[1].

Cada elemento, de la descomposición, se llama VOXEL o *elemento de volumen*, el más común es el cubo, entre más pequeño se mejora la representación, sin embargo, n^3 celdas son necesarias para un objeto de n voxels en 3D.

Los cubos son elementos que no se cubren, ni se extiende uno sobre otro, ni se pliegan o descansan de manera conjunta; son uniformes en forma, tamaño y orientación, haciendo una división regular del espacio. Así, cada celda o cubo en 3D tiene seis caras, con doce orillas o aristas y ocho vértices, que comparte con sus vecinos, en el caso extremo, su grado de conectividad puede variar y genera reglas de validez.

---

[1] *Hoffman Christoph M., Solid Modeling: an Introduction,p.121.*

Cada elemento voxel puede ser descrito en términos de sus esquinas, almacenando solamente sus coordenadas de una esquina de cada cubo, y codificarla en un arreglo o matriz de 3D de datos binarios. El arreglo nos representa la existencia del cubo, dentro del modelo, de tal manera que, el elemento a tratar será 1 si se encuentra dentro del modelo, sino sería 0, es decir, no pertenece.

Esta forma de representación binaria es parecida a la utilizada en el procesamiento digital de imágenes. Es aquí donde se unen dos campos de la computación, el modelado de sólidos y el procesamiento digital de imágenes; muchas de las técnicas utilizadas en el modelado de sólidos, dentro de la metodología de descomposición, tiene sus raíces en el procesamiento digital de imágenes.

Entonces describir un sólido significaría listar todos los cubos que pertenecen al modelo o forman parte de él, pero en el modelado de sólidos, no se tiene una imagen u objeto inicial, o es muy difícil describir un objeto real generalizado a cubitos.

2.2 La subdivisión del espacio

Al tratar de describir un objeto mediante todos los voxels que le pertenecen, cuando cada cubo puede ser descrito por sus esquinas, se generan grandes listas de datos. La enumeración completa de los modelos hechos por descomposición tiene muchas ventajas, tales como su simplicidad, su generalidad y permiten un gran número de algoritmos. Sin embargo, el consumo de memoria es alto; se puede cambiar la división regular del espacio por algo más eficiente, una división adaptativa,

A este esquema de trabajo basado en la subdivisión del espacio se logra por simple observación de la rejilla descriptiva de la enumeración completa de un espacio 3D, donde muchas veces un cubo blanco (1) tiene de vecinos otros cubos del mismo tipo ”1”. Al observar la combinación y codificarla en modelos de datos comparativamente similares, podemos reducir la cantidad de memoria utilizada por dicha rejilla.

La división adaptativa tiene una propiedad fundamental en la que el número de nodos necesarios para la representación de un sólido es proporcional al área de su superficie. De aquí que el número de elementos es proporcional al cuadrado r^2 de la resolución r que se utilice, en el otro caso, el tamaño de elementos en un esquema completo es proporcional a r^3.

2.3 Árboles octales u “octrees”

El principal ejemplo de la división adaptativa son los árboles octales dentro de la representación de sólidos y su análogo son los quadtree para la representación de objetos en 2D.

Estos árboles son una variante jerárquica de la enumeración espacial, como resultado de un derivado de los quatrees. Los octrees utilizan la división recursiva del espacio en ocho octantes, por ello el nombre de octree, que son colocados es una estructura lógica de árbol octario (de ocho hojas).

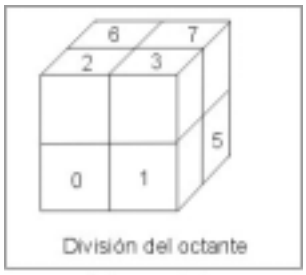


Figura 2.2

Usualmente, el árbol es colocado alrededor del origen del sistema de coordenadas, con su primer nivel de octantes que corresponden a los octantes del espacio general, a manera de cuadrantes del sistema de coordenadas. El octante número 3 es el que corresponde a la parte positiva cuando x, y, z > 0. Esto es un acuerdo para una apropiada transformación y procesado de información.

La idea fundamental es un algoritmo divide-y-vencerás. En el quadtree el plano se divide en cuadrantes. Si se usa el cuadrante completo se describe como *"full"*, si no como "e*mpty"*. Si es parcialmente usado se subdivide, hasta una profundidad determinada. Un octree es semejante con tres Dimensiones, subdividido en octantes de manera recursiva. El número de nodos es proporcional al perímetro en un quadtree o superficie del objeto en un octree, en ambos casos es proporcional a la medida de la frontera.

Figura 2.3

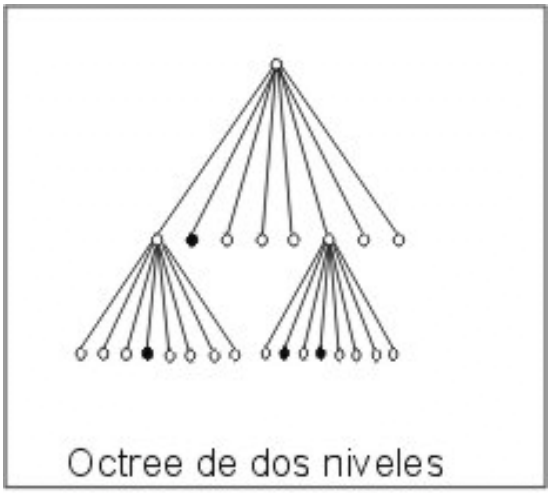


Figura 2.4

Cada nodo del árbol consiste en un código y ocho apuntadores hacia ocho "hijos", enumerados del 0 al 7 (000 al 111); El código indica si el cubo es negro o blanco, es decir, sus valores son 0 y 1 respectivamente, lo que significa, si el código es negro, el espacio representado por el nodo está lleno, es decir, todo el cubo pertenece al modelo, y el nodo en si se convierte en una "hoja" y todos sus hijos se encuentran con valor de nil; cuando el código es blanco, ese espacio se encuentra vacío y no pertenece al modelo, el nodo también es una "hoja"; la tercera posibilidad es que el nodo sea gris, esto indica parcialmente ocupado o escasamente

vacío; en este último caso, los ocho apuntadores hijos corresponden a la división regular del padre. Por ejemplo el objeto de la figura 2.3 corresponde al octree de la figura 2.4, utilizando solamente dos niveles.

2.4 Estructura de datos

La estructura de datos que representa la forma de un árbol "octree" es la siguiente:

```
struct octreeroot
{
     float xmin, ymin, zmin;  /*limites del espacio
     float xmax, ymax, zmax;
     struct octree  *root;    /* raíz del árbol
};

struct octree
{
     char code;/*Codigo para Negro, Blanco o gris
     struct octree *oct[8];/*hijos para el caso de gris
};
```

El espacio de interés en representar es una caja ortogonal en el espacio XYZ, colocado a partir del nodo especial, la raíz.

Es fácil observar que 7/8 de los nodos en un octree son hojas, entonces se podría representar de una manera especial y evitar que sus ocho hijos se pongan con valor de nil, y hacer más eficiente esta representación.

2.5 Proceso de construcción de un octree

Los modelos hechos en base a un árbol octree, es un sólido de primitivas. Para cada tipo de primitiva, se tiene un procedimiento de clasificación entre la instancia de la primitiva y el nodo arbitrario del octree. Este proceso de clasificación debe ser capaz de distinguir entre los casos de:

1. El nodo esta en el exterior de la primitiva.
2. El nodo esta en el interior de la primitiva.
3. El nodo pertenece parcialmente a la primitiva.

Este método es usado en forma recursiva; al inicio, el primer nodo (raíz) es blanco o vacío, éstos son los datos iniciales del algoritmo. Siguiendo el algoritmo, va clasificando cada nodo contra la primitiva, si el primer o segundo caso aplica, el nodo es marcado como blanco o negro, y la recursión termina; en caso contrario, el algoritmo procede a marcar el nodo como gris, subdividiendo en ocho octantes y llamando recursivamente cada uno de ellos. La subdivisión es continuada hasta que la resolución deseada es alcanzada, generalmente se usan de seis a doce niveles.

El algoritmo de construcción de un octree es el siguiente:

```
make_tree(p, t, depth)
primitive *p;  /* la primitiva a modelar
octree *t;/* nodo inicial del octree, blanco
int depth;     /* maxima recursion
{
     int i;
     switch(classify(p, t))
     {
          case WHITE:
               t->code = WHITE;
               break;
          case BLACK:
               t->code = BLACK;
               break;
          case GREY:
               if (depth == 0)
               {
                    t->code = GREY;
               }
               else
               {
                    subdivide(t);
                    for (i=0; i<8; i++)
                         make_tree(p,    t->oct[i],
depth-1);
               }
               break;
     }
}

/* clasificar nodo octree contra primitiva
classify(…)

/* divide el nodo en ocho octantes
subdivide(…)
```

Las primitivas de fácil implementación son las formas regulares como cubos, esferas, cilindros y conos, y algunos objetos de un orden mayor como los llamados supercuadráticos[2]. Los octrees y quatrees pueden ser construidos a partir de imágenes digitales cuando se tiene la información.

[2] *Martti Mäntyla, An introduction to solid modeling, p. 68.*

2.6 Algoritmos de los árboles octales

Consiste en la metodología sistemática para generar y desplegar el modelo, donde la enumeración de atrás hacia adelante permite que los nodos sean listados, desplegando del más lejano al más cercano, desde un punto de VISION, luego sus tres vecinos, en cualquier orden y así recursivamente.

Un modelador de sólidos con las funciones completas debe incluir los algoritmos que realicen las tareas siguientes:

1. *Generador del árbol*, crea el octree desde las primitivas parametrizadas o desde otros tipos de modelos geométricos.
2. *Operaciones de Conjunto*, que pueden llevarse a cabo entre dos octrees, con un espacio idéntico de interés, con el cálculo del octree resultante dado por la unión, intersección o diferencia de conjunto de los dos argumentos.
3. Operaciones geométricas, que se hacen sobre un octree y calcular un nuevo octree como resultado de una traslación, rotación o escalamiento de los objetos modelados. Otro tipo de operación geométrica calcula el nuevo octree obtenido a partir de operaciones como las transformaciones de perspectivas. Algunos de estos problemas requieren un algoritmo complejo, muchas veces no intuitivo. La traslación de un octree es una operación bastante compleja y difícil.
4. *Procedimiento de análisis,* calcula ciertas propiedades del objeto tales como su volumen o área de superficie del octree; un procedimiento de componentes conectados que etiqueta cada nodo del octree con el identificador del objeto conectado, como el ejemplo anterior (figura 2.3), este análisis es una operación muy compleja.
5. *Generador de desplegado*, crea la imagen gráfica del objeto modelado por el octree.

2.7 Presentación gráfica

Una de las características esenciales de este método, es que guardan de manera implícita la información sobre el orden espacial, lo que se puede explotar en el algoritmo de diseño, haciendo posible crea programas con un proceso más simple ue solo pasen por el árbol obteniendo los argumentos necesarios y ejecutar cada operación en cada nodo. Y no es necesario un reordenamiento para esconder las superficies removidas o cerradas[3].

El orden apropiado depende de la ubicación del observador con respecto al árbol que se mostrará. Si el observador se encuentra en el octante 3 donde X, Y, Z > 0, del espacio, el orden de desplegado sería 4, 0, 5, 1, 6, 2, 7, 3.

[3] *Martti Mäntylä, An Introduction to Solid Modeling, p. 75.*

2.8 Análisis de operaciones

Muchas de las operaciones que se pueden hacer, son ejecutadas de acuerdo al paradigma nodo-a-nodo, por ello, las propiedades integrales como volumen, centro de masa y momentos de inercia se pueden calcular por un simple algoritmo de árbol transversal.

2.9 Operaciones de conjunto para los octrees

También para este caso, se puede utilizar un algoritmo de árbol transversal; las operaciones de conjunto implican dos árboles octree, resultando un tercer octree que representa la operación booleana deseada, tal como la unión, intersección o diferencia de conjunto de los argumentos[4]. Por ejemplo, para el cálculo de la intersección, al procesar los nodos n_1 y n_2, ocurrirá que:

1. Los nodos n_1 y n_2 son ambos hojas. En este caso, el nodo resultante del octree es negro si ambos lo son, de otra manera es blanco.
2. Uno de los nodos es hoja. Si el nodo hoja es negro, el sub-árbol de la que no es hoja es copiado al resultado del octree, de otra manera el nodo resultado es blanco.
3. Ambos no son hojas, entonces el algoritmo debe considerar resursivamente los hijos de esos nodos.

Como los argumentos ya están presentes su construcción no es tomada en cuenta; en la mayoría de los casos la complejidad del algoritmo es proporcional al árbol más pequeño.

No todos los algoritmos pueden ser llevados a una manera transversal, en algunos casos es necesario conocer a los nodos vecinos, como en los casos de efectos visuales como transparencias, el accesar a un nodo vecino puede ser complicado y requiere subir un nivel en el árbol o hasta la raíz en el peor de los casos. Para lo cual se puede incluir más apuntadores a ciertas áreas del árbol, y requiere un cuidado de balance en el árbol.

2.10 Propiedades de los octrees

- *Modelo poderoso*, los octrees son modelos de representación aproximada, y pueden ser exactos para ciertos objetos.
- *Validez*, no requiere de una conectividad especial, todos los octrees son representaciones válidas de algún sólido.

[4] *Martti Mäntylä, An Introduction to Solid Modeling, p. 78.*

- *No ambigüedad y unicidad,* hasta los límites de resolución, todos los octrees no ambiguos, definen un sólido.
- *Lenguajes de descripción,* los octrees son formados mediante una conversión de otras representaciones, como las constructivas; y en el procesamiento de imágenes, los octrees y los quatrees son formados directamente de imágenes digitales de datos, por media de un proceso de "raster".
- *Consistencia,* en general el número de nodos que un octree representa es proporcional al área del objeto. En promedio un octree fácilmente puede medir más de un millón de bytes de memoria.
- *Operaciones cerradas,* un octree soporta de algoritmos cerrados para los problemas de translación, rotación y operaciones booleanas.
- *Sencillos computacionalmente, muchos algoritmos de los octrees toman la forma de transversal, donde las operaciones son relativamente fáciles para cada nodo del árbol.*

2.11 Subdivisión binaria del espacio

Una alternativa, al procesamiento de los octree, es la subdivisión de un espacio binario. Esto implica, un enfoque que divide los nodos grises en dos hojas en vez de ocho octantes. Lo que los hace mucho más pequeños, de esto surge la versión lineal de los octrees.

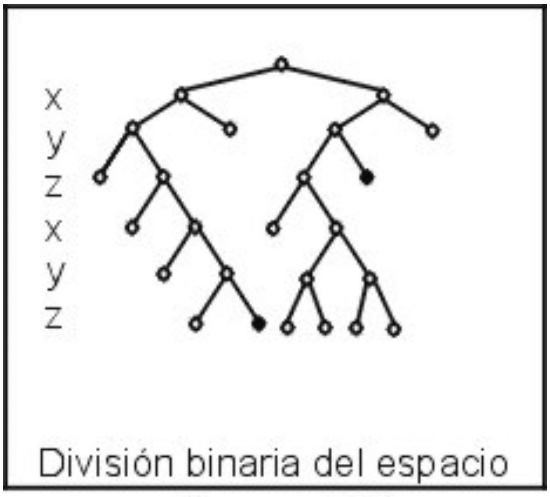


Figura 2.5

Aunque el espacio sea dividido en dos o en ocho como un octree normal, la cantidad de memoria utilizada siguen siendo demasiada, pero menor al utilizada por la enumeración espacial.

Lo anterior ha provocado investigaciones para comprimir la representación de árboles, muchas alternativas de representación reemplazan la estructura del árbol por un apuntador libre, una estructura de datos lineal.

Esta nueva propuesta se le llama octrees lineales.

Los números de los octantes pueden ser utilizados para encontrar la ruta de cualquier nodo, excepto la raíz misma, ya que cada octante tiene asignado n número identificador del 0 al 7, agregando el nivel i al que corresponde; agregando un dígito llamado número final, puede ser incluido para marcar número que tiene menos dígitos que la resolución máxima.

Basándose en estas observaciones, se generó el *árbol lineal de Gargantini,* en su enfoque, el octree lineal es una simple lista ordenada de rutas y direcciones de todos los bloques y nodos. Del ejemplo anterior, su lista sería:

$$\{03,1X, 51, 53\}$$

Donde X es el número final.

Otro método de compresión lineal de un octree es llamado representación-DF; es formulado al transversar el árbol octree en preorden y almacenar cierta información de los nodos encontrados, además utilizada una B, W y (, para denotar el tipo de nodo, ya sea negro, blanco o hijo. Para el ejemplo anterior tenemos:

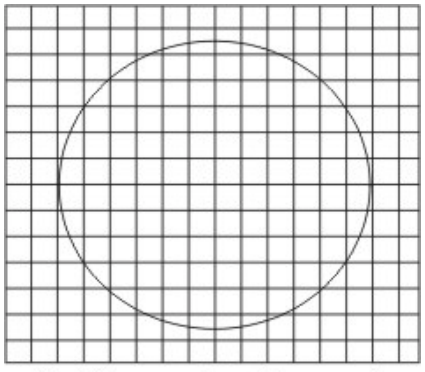
$$((WWWBWWWWBWWW(WBWBWWWWWW$$

Como sólo se tienen tres tipos de caracteres, se puede utilizar dos bits para decodificar los datos.

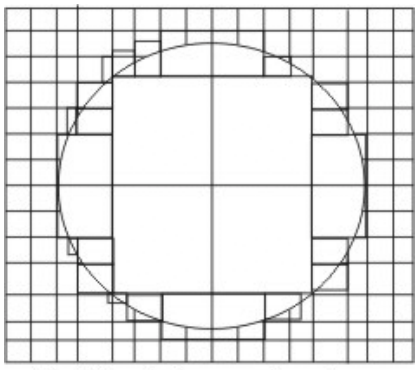
Las representaciones lineales son suficientes para muchos algoritmos. Las operaciones booleanas en octrees lineales se hacen al mezclar las dos cadenas de caracteres, lo cual es una operación fácil de realizar.

Un *bintree*, está basado en la subdivisión binaria lineal y es muy similar a la compactación de los octrees, pero son más pequeños porque el número de hojas es mucho menor, lo que se puede notar en las figuras 2.4 y 2.5, las hojas terminales pueden decodificarse con un solo bit ya que solo se tienen dos posibilidades W o B.

Las búsquedas geométricas por subdivisión en un árbol octree explícito para examinar a un nodo en particular, si se encuentra o no en el espacio de interés es muy fácil.



División regular del espacio
Figura 2.6



División del espacio octree
Figura 2.7