UNIDAD 3

3 Creación de Árbol Octal vía la definición CSG

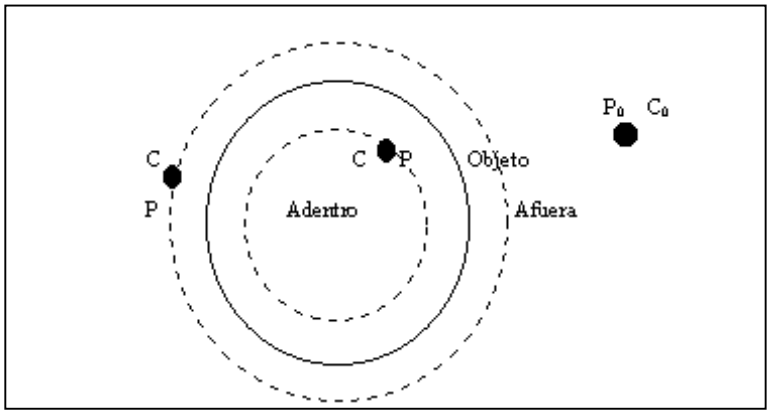
3.1 Creación del Árbol Octal de un objeto sólido vía la definición CSG

Un árbol CSG hace uso de un conjunto de primitivas sólidas, estas primitivas se pueden representar mediante árboles octales, y mediante la comparación de los octantes que forman las primitivas se logra obtener el árbol octal de un objeto sólido.

Obtener el Árbol octal de una primitiva tal como el cilindro, la esfera, el paralelepípedo y el Semi-Espacio es fácil, ya que la superficie de una primitiva se describe por medio de una función de la forma f(x,y,z)=0, únicamente se debe determinar si el octante está adentro o afuera de la primitiva. Lee y Requicha [11,12] aplicaron el método de aproximación de espacio para convertir el problema de la determinación de un octante en un sólido. Mediante este método se determina el color del octante, si se considera al octante como un punto y que el objeto puede crecer o no de acuerdo al color. Si el punto está adentro del sólido, el octante está dentro de la primitiva y se marca como negro; si el punto está afuera de la primitiva sólida, el octante se marca con blanco. De otra forma el octante esta parcialmente dentro y parcialmente fuera del objeto y se marcará con gris. En la figura 3.1 se muestra un ejemplo en el que se usa esta técnica. C_0 se reduce a un punto P_0 y la primitiva sólida se va a crecer o disminuir de acuerdo a la determinación del estado del octante

Este método se va a aplicar a una primitiva hasta llegar a la precisión deseada que es cuando el tamaño del octante es del tamaño del factor de escala. En el peor caso, la complejidad del algoritmo es proporcional al producto de número de primitivas y el número de unidades cúbicas en el espacio del Árbol Octal [3].

Figura 3.1 La Detección del color de un octante en una primitiva.



"Elaboración propia"

3.2 Creación de Árbol Octal vía la definición CSG por el método clásico

3.2.1 Características

Se puede pensar en un objeto definido vía la representación CSG, como un conjunto de primitivas sólidas las cuales se agrupan por un conjunto de operadores. Para convertir un objeto definido en CSG a un Árbol Octal, se construye un Árbol Octal por cada primitiva sólida, entonces, se unen los resultados de estas primitivas sólidas al usar el conjunto de operadores Booleanos [3]. Cuando el objeto final se obtiene, los Árboles Octales que se crearon para las primitivas y para los resultados parciales ya no se ocupan más.

3.2.2 Función Met_Clásico

A continuación se muestra la subrutina Met_Clásico. Nodo contiene la información ya sea si se trata de un operador Booleano o de una primitiva. En caso de ser primitiva, el Nodo contiene las características de ésta, como su nombre, su ubicación y su proporción; en caso de ser operador Booleano contiene el nombre del operador: unión, intersección diferencia o complemento. En la subrutina, se puede observar que cuando el Nodo es una primitiva, se crea el árbol octal de ésta; y cuando el nodo es una operación Booleana, se crea el árbol octal de la operación al llamar a la subrutina Op_Booleana, la cual tiene como parámetros de entrada: el operador Booleano y los dos nodos hijos (hijo izquierdo e hijo derecho) para la unión, intersección y diferencia; en caso de que la operación Booleana se trate del complemento únicamente necesita al nodo izquierdo.

```
Met_Clásico(Nodo,mundo,ancho,precisión)
   {
   Con Nodo hacer
       {
       Si(opción=paralelepípedo ó opción=Esfera ó opción=Cilindro ó
       opción=Espacio ó opción=Mundo ó opción=SemiEspacio)
           Const_primitiva(Nodo, mundo, ancho, precisión);
       Si no: Si(opción=complemento)
           {
           Op_Booleana(complemento, Met_Clásico(Nodo.izq, mundo,
           ancho, precisión));
           }
       Si  no:  Si(opción=unión  ó  opción=intersección  ó
       Opción=diferencia)
           {
```

```
            Op_Booleana(opción,
            Mét_Clásico(Nodo.izq,mundo,ancho,precisión),
            Mét_Clásico(Nodo.der,mundo,ancho,precisión));
            }
        }
    }
```

3.3 Creación de Árbol Octal vía la definición CSG por el método Top Down

El método Top Down consiste en generar el Árbol Octal de un objeto desde su representación CSG, con el propósito de no generar los Árboles Octales de cada primitiva que comprenden al objeto ni las operaciones booleanas entre estos Árboles Octales [4], las ideas de este método se encuentran en [4].

Para realizar este método, se tiene que determinar cuál es el estado de cada octante que forma parte del objeto final con respecto a las primitivas que forman parte de su árbol CSG, puede ser negro si se determina que el octante esta adentro del objeto, blanco si se determina que el octante esta afuera del objeto o gris dependiendo si el octante tiene solo una parte adentro y la otra parte afuera del objeto. Entonces se puede ver que se está hablando ya de la determinación del estado de los octantes a nivel del objeto final, por medio del análisis de todas las primitivas y su relación entre ellas.

El número de hojas que comprende el Árbol Octal de una primitiva (o un objeto) es del orden del área de su superficie: 2^{2n}. (Los octantes más pequeños en la superficie se requieren para conservar la forma del objeto) De esta forma, el número de pruebas que se requiere es menor que con el método clásico, el cual es de 2^{3n} [4].

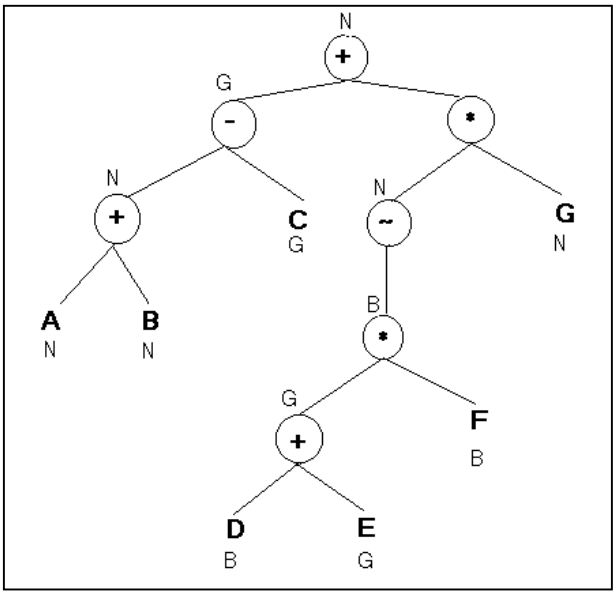
Los objetivos principales son:

- Hacer la evaluación directa de cada octante desde el árbol CSG, para obtener únicamente el árbol octal del objeto final.
- Usar sólo la cantidad de memoria que realmente se ocupará, puesto que al realizar el método clásico los árboles que se fueron creando para obtener el objeto final ya no se ocupan
- Al no hacer los árboles octales de las primitivas y ni los árboles octales de los operadores, se pretende tener una disminución de tiempo.

Un ejemplo de cómo se calculan los octantes se muestra en la Figura 3.2 donde A, B, C, D, E, F G son primitivas y N es octante negro, B es octante blanco y G es octante gris. + es el operador Booleano unión, * es el operador Booleano intersección, - es el operador Booleano diferencia y ~ es el operador complemento. Como puede verse en ésta figura, cada nodo tiene el color de octante que le corresponde con respecto al objeto. La forma en como se almacena el octante está condicionada, y depende de la relación final que se obtenga al relacionar a las diferentes primitivas por medio de los operadores Booleanos, de acuerdo a la definición CSG del objeto. En la figura 3.2 después de haber realizado la evaluación del árbol CSG se obtiene que el octante es negro, entonces el octante almacenará un 01, suponiendo que hubiese sido blanco, almacenaría un 00, o en caso de que hubiese sido gris se almacenaría un 10 y se procedería a entrar en una recursión dividiendo el octante en 8 sub-octantes para determinar el estado de dichos sub-octantes.

Figura 3.2 Ejemplo de como se calcula un octante

de un objeto Final por medio del método Top Down



"Elaboración propia"

3.3.1 Funciones usadas

Para implementar la técnica Top Down, se ocupan las siguientes subrutinas: Octante_en Primitiva, Octante_en_Operación_Booleana, Octante_en_Objeto y Crear_Árbol_Octal

3.3.1.1 Octante_en _Primitiva

Esta función determina la relación entre un octante dado y una primitiva específica, recibe como parámetros de entrada la primitiva, las coordenadas del octante, el ancho del octante y la precisión. La definición de la primitiva incluye nombre de la primitiva (Paralelepípedo, esfera, cilindro, semi-espacio, etc.) su ubicación y cuáles son sus

proporciones, por ejemplo una esfera tiene su punto donde se ubica X,Y y Z y los radio en cada eje.

Ejemplos de subrutinas de primitivas

```
Octante_en_Paralelépido(Primitiva,octante,ancho,precisión) Relación.
Octante_en_Esfera(Primitiva,octante,ancho,precisión) Relación.
Octante_en_Cilindro(Primitiva,octante,ancho,precisión) Relación.
Octante_en_SemiEspacio(Primitiva,octante,ancho,precisión) Relación.
Octante_en_Mundo(Primitiva,octante,ancho,precisión) Relación
```

Algoritmo para Octante_en_Primitiva

```
Octante_en_Primitiva(Primitiva,octante,ancho,precisión) Relación
    {
    Si(Primitiva.opción = Paralelepípedo)
        regresar(Octante_en_Paralelepípedo(Primitiva,octante,ancho,
        precisión))
    Si no: Si (Primitiva.opción = Cilindro)
        regresar(Octante_en_Cilindro(Primitiva,octante,ancho,precisi
        ón))
    Si no: Si (Primitiva.opción = Esfera)
        regresar(Octante_en_Esfera(Primitiva,octante,ancho,precisión
        ))
    Si no: Si (Primitiva.opción = mundo)
        regresar(Octante_en_Mundo(Primitiva,octante,ancho,precisión)
        )
    Si no: Si (Primitiva.opción = EspacioMedio)
        regresar(Octante_en_EspacioMedio(Primitiva,octante,ancho,pre
        cisión))
}
```

3.3.1.2 Octante_en_Operación_Booleana

La subrutina Octante_en_Operación_Booleana es utilizada para ejecutar la operación Booleana entre dos relaciones (primitivas). La subrutina recibe como parámetro de entrada un operador Booleano (unión, intersección, diferencia o complemento), el operador 1 y el operador 2, y da como salida la relación total (Relación). De esta forma, la subrutina únicamente evalúa un nodo interno (no terminal) del árbol CSG. La subrutina es de la forma:

```
Octante_en_Operacion_Booleana(operador, op1, op2)
        {
        Sí(operador = unión)
                {
                Sí(op1 = blanco y op2 = blanco)regresa(blanco);
                Sí no: Sí(op1 = negro ó op2 = negro) regresa(negro);
                Sí no: Sí(op1 = gris ó op2 = gris) regresa(gris);
                }
        Sí no: Sí(operador = intersección)
                {
                Sí(op1 = blanco ó op2 = blanco) regresa(blanco);
           Sí no: Sí(op1 = negro y op2 = negro) regresa(negro);
                Sí no: Sí(op1 = gris ó op2 = gris)regresa(gris);
                }
        Sí no: Sí(operador = diferencia)
                {
                Sí(op1 = blanco ó p2 = negro) regresa(blanco);
                Sí no: Sí(op1 = negro y op2 = blanco)
regresa(negro);
                Sí no: Sí regresa(gris);
                }
        Sí no: Sí(operador = complemento)
                {
           Sí(op1 = negro) regresa(blanco);
                Sí no: Sí(op1 = blanco) regresa(negro);
                Sí no: Sí(op1 = gris) regresa(gris);
                }
        }
```

Como se puede observar en el caso del complemento, solo se usa Relación1. Las relaciones detalladas están contenidas en la tabla 3.1

Tabla 3.1 Resultados entre dos relaciones dependiendo de la operación Booleana

Relación1	Relación2	Diferencia R1-R2	Intersección R1*R2	Unión R1+R2	Complemento R1
B	B	B	B	B	N
B	G	B	B	G	N
B	N	B	B	N	N
G	B	G	B	G	G
G	G	G	G	G	G
G	N	B	G	N	G
N	B	N	B	N	B
N	G	G	G	N	B
N	N	B	N	N	B

La tabla se obtuvo de [4]. Pág. 56

3.3.1.3 Octante_en_Objeto

Para crear la función Octante_en_Objeto se utilizan las subrutinas Octante_ en_Primitiva y Octante_en_Operación_Booleana, esta nueva función regresa negro, blanco o gris dependiendo de la relación entre un octante dado y un objeto arbitrario que está definido como árbol CSG. Cada nodo no terminal en el árbol CSG posee dos hijos Hijo_izquierdo e Hijo_Derecho para el caso de las operaciones intersección, unión y diferencia; para el caso del nodo complemento, solo posee un hijo.

La función Octante_en_Objeto va a hacer su recorrido en post-orden si la relación es gris, entrando en un proceso recursivo, esto significa que va a visitar a los hijos e inmediatamente después se evalúa el nodo padre. A continuación se muestra la subrutina:

```
Octante_en_Objeto(Apuntador_al_Objeto_CSG,octante,ancho,precisión)
Relación
    {
    Si(opción = primitiva)
```

```
        Regresa(Octante_en_Primitiva(Apuntador_al_Objeto_CSG.nodo,
        octante, ancho, precisión));
    Si no: Si (opción = Operación_Booleana)
        {
        Si (opción = Complemento)
            Regresa(Octante_en_Operacion_Booleana(
            Apuntador_al_Objeto_CSG.opcion,Octante_en_Objeto(
            Apuntador_al_Objeto_CSG.izq,octante,ancho,precisión)))
        Si no
            Regresa(Octante_en_Operacion_Booleana(
            Apuntador_al_Objeto_CSG.opcion,Octante_en_Objeto(
            Apuntador_al_Objeto_CSG.izq,octante,ancho,precisión),
            Octante_en_Objeto(Apuntador_al_Objeto_CSG.der,octante,an
            cho,precisión)));
        }
} //Octante_en_Objeto
```

Con este método se puede obtener la relación entre cualquier octante que forme al objeto y sin importar la forma de dicho objeto.

3.3.1.4 Crear_Árbol_Octal

Con las subrutinas anteriores el Árbol Octal de un objeto se puede crear directamente desde el Árbol CSG, por medio de la subrutina Crear_Árbol_Octal, la cual tiene los siguientes parámetros: El Árbol CSG que define al objeto (Apuntador_al_Objeto_CSG), un apuntador el cual define el Árbol Octal, el octante en cuestión (cuyo tamaño se da en la primera llamada), el ancho del octante y la precisión requerida. Usando el Árbol CSG, la subrutina Crear_Árbol_Octal prueba y marca (Apuntador_al_Árbol_Octal.Estado) la relación entre cada octante y el objeto con respecto a lleno, vacío o parcial. Estas últimas relaciones corresponden al Árbol Octal mismo y se usan para crear las hojas del Árbol Octal (Cuando la relación es

lleno o vacío) o define los nodos cuando la relación es parcial. La subrutina Crear_Árbol_Octal [4] se muestra abajo.

```
Crear_Árbol_Octal(Apuntador_al_Objeto_CSG,Apuntador_al_Árbol_Octal,oc
tante,ancho,precisión)
    {
    caso
    Octante_en_Objeto(Apuntador_al_Objeto_CSG,octante,ancho,precisió
    n)
        Blanco: Apuntador_al_Árbol_Octal.Estado=Blanco     {hoja}
        Negro Apuntador_al_Árbol_Octal.Estado=Negro   {hoja}
        Gris:
           {
           Apuntador_al_Árbol_Octal.Estado=Gris       {Nodo}
           Para k=1 hasta 8
               {
               Crear_Árbol_Octal(Apuntador_al_Objeto_CSG,
               Crear_SubOctante(Octante,Apuntador_al_Árbol_Octal.hi
               jo(k),ancho,precisión));
               }
           }
    }
```

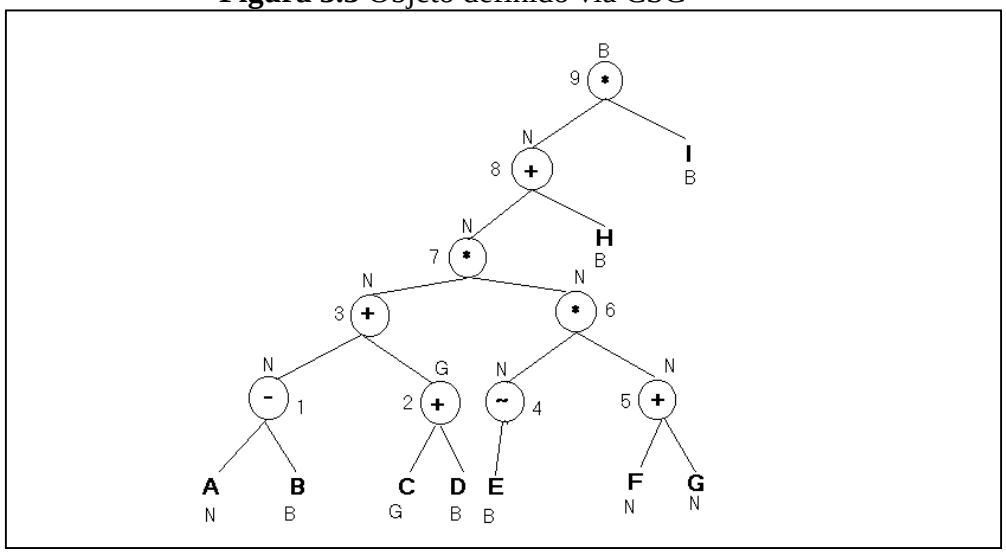
3.4 Ejemplo del recorrido de un Árbol CSG por medio del método Top Down

En la Figura 3.2 se muestra un objeto definido por medio de un Árbol CSG, este árbol esta formado por 9 primitivas llamadas A, B, C, D, E, F, G, H e I y nueve operadores Booleanos: diferencia, unión, unión, complemento, unión, intersección, intersección, unión e intersección. Las relaciones se obtienen por medio de las subrutinas Octante_en_Primitiva: Negro para las primitivas A, F y N; Blanco para B, D, E e I y Gris para la primitiva C, los nodos de los Operadores Booleanos del árbol CSG se enumeran del 1 al 9. Esta enumeración

se hace de acuerdo a como se evalúan los operandos, en forma post-orden. La expresión (((((A-B)+(C+D))*((~E)*(F+G)))+H)*I) describe al objeto definido en dicha figura.

El Árbol se va a recorrer de izquierda a derecha, entonces el primer nodo a evaluar es el que esta etiquetado con uno y es una diferencia entre las primitivas A y B, apoyándose en la tabla 3.1 se obtiene el resultado de la relación como "N", el siguiente nodo a evaluar es el nodo 2, que esta dado por la unión de las primitivas C y D y la relación es "G", a continuación se procede a evaluar la relación del nodo 3, que es la unión entre relación del nodo 1 con la relación del nodo 2 dando la relación "N", y así sucesivamente se continúan evaluando los nodos restantes hasta llegar al nodo 9 que es un nodo blanco.

Figura 3.3 Objeto definido vía CSG



"Elaboración propia"

En la tabla 3.2 se puede ver la secuencia de este recorrido, de acuerdo a la enumeración de los nodos. La razón puede ser una primitiva o el resultado de algún nodo ya evaluado, en el caso del complemento solo se tiene una razón.

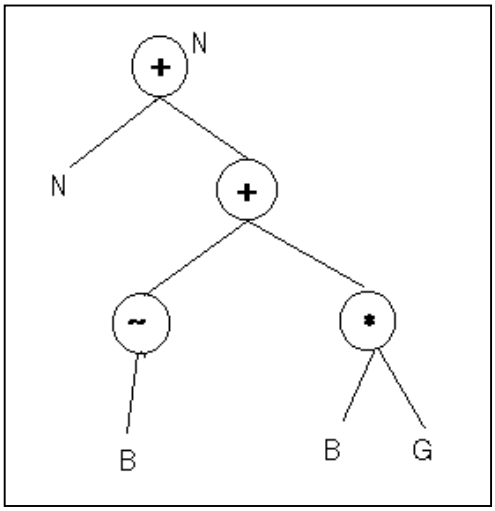
Tabla 3.2 Recorrido del árbol CSG de la Figura 3.3, por el método Top Down

Nodo No.	Razón 1	Razón 2	Relación 1	Relación 2	Operación	Resultado Actual
1	A	B	Negro	Blanco	-	Negro
2	C	D	Gris	Blanco	+	Gris
3	Nodo 1	Nodo 2	Negro	Gris	+	Negro
4	E		Blanco		~	Negro
5	F	G	Negro	Negro	+	Negro
6	Nodo 4	Nodo 5	Negro	Negro	*	Negro
7	Nodo 3	Nodo 6	Negro	Negro	*	Negro
8	Nodo 7	H	Negro	Blanco	+	Negro
9	Nodo 8	I	Negro	Blanco	*	Blanco

"Elaboración propia"

En [4] nos dice que la expresión puede optimizarse si en la Intersección el primer operando es Blanco, automáticamente se sabe que es Blanco el resultado y en la unión si el primer operando es negro se sabe que el resultado de la relación sea negro, entonces ya no hay necesidad de analizar el segundo operando.

Figura 3.4 Objeto definido vía CSG



"Elaboración propia"

En la figura 3.4 hay una relación entre un nodo negro y un subárbol, en este caso, como es una unión y el primer operando es negro, el sub árbol del operando dos ya no se evalúa.

3.5 Conclusiones

El método propuesto no requiere la creación de los Árboles Octales de las primitivas que comprenden al objeto, ya que el método Top Down involucra el análisis de las relaciones entre octantes y primitivas. Entonces los requerimientos de memoria disminuyen en gran parte cuando se genera el objeto. Además, La reducción de memoria se hace más notoria entre más primitivas formen al objeto. Ya que solo se ocupará la memoria requerida para generar el objeto final. Mientras que en método Clásico entre más primitivas formen al objeto más memoria se ocupará. Este método es proporcional al área de la superficie del objeto final, más que a las sumas de las áreas de las superficies de las primitivas [4].