5. Análisis y desarrollo de algoritmos y estructuras de datos de modelos de Sólidos

Los conceptos principales que debemos tener en cuenta son dos: primero, el objetivo de crear un sólido virtual dentro de la computadora están basados en la necesidad del observador; y segundo la generación de una estructura de información que almacene la información requerida para generar ese sólido virtual, debe ser una estructura relacional, que guarde una estructura jerárquica dinámica de las entidades.[1]

Este modelo de datos conectará a los diferentes niveles de resolución.

Esta conectividad permitirá navegar a través de su estructura y como resultados cambiaremos de resolución tanto como el usuario lo requiera, es decir, navegaremos sobre las entidades de un mundo virtual.

5.1 Tipos de Información

Debemos hacer una importante distinción en la información que se manejará:

- Información paramétrica, son los datos concernientes a una entidad y está almacenada.

- Información de simulación, es aquella que es inicializada al tiempo de ejecución.

Los datos de simulación son inicializados cuando creamos una nueva entidad, a continuación se crean otras entidades en una estructura dinámica de *árbol.*[2]

La información necesaria es:

- Status: agregado, desagregado, media-resolución, etc.

---

[1] *Heiny Loren, Advanced Graphics Programming using C/C++, p. 221.*
[2] *Angell Ian O. y Griffith Gareth, High-resolution using C++, p. 324.*

- Umbrales de desagregación y de agregación: los límites en ambos procesos para evitar posibles oscilaciones alrededor de las transiciones.

- Observador responsable: es el dueño de la base de datos, requerido para generar y crear entidades.

- Referencia al padre de la entidad.

- Referencia al primer elementos de la entidad: cuando ocurre un evento de desagregación, los parámetros de la base de datos usada deben hallar el primer "hijo" o elemento de referencia a la instancia del primer "hijo" que es guardado.

La siguiente es la definición generalizada de una estructura de árbol[3]:

```
struct OCTREEROOT
{
   float xmin, ymin, zmin;
   float xmax, ymax, zmax;
   AnsiString Nombre;
   struct OCTREE *root;
};

struct OCTREEROOT *Raiz; // Estructura Mayor
```

En este caso, nos da las características del espacio a describir, y el inicio del sólido a modelar, cuya estructura es la siguiente[4]:

```
#define HIJOS 8
struct OCTREE {
   int nivel;     // nivel del arbol
   int codigo;   // tipo W, B o G
   int cubonum;   // consecutivo de cubo
   int figura;    // identificador de figura
   struct OCTREE *oct[HIJOS];
};

struct OCTREE *octree;  // Estructura de trabajo
```

El código, se interpreta como la característica del nodo, que puede pertenecer al sólido de tres formas: de manera completa FULL o Black, de manera incompleta Semicompleta o Grey y por último, no pertenecer al sólido, White o Vacío.

Esta definición es:

[3] *Heiny Loren, Advanced Graphics Programming using C/C++, p. 291.*
[4] *Hoffman Christoph M., Solid Modeling: an Introduction, p.187.*