IMPLEMENTACIÓN DEL OCTREE Y LA VISUALIZACIÓN DE FIGURAS TRIDIMENSIONALES

Paul Antony Parizaca Mozo

—*Abstract* – This paper discusses the implementation of an Octree, a three-dimensional hierarchical data structure, and its integration with the VTK (Visualization Toolkit) visualization toolkit. The focus is on the efficient representation of three-dimensional data using Octree and its practical application in the visualization of three-dimensional figures by plotting the leaf nodes of the Octree using VTK highlighting its usefulness for intuitive graphical representation of the information contained in the Octree structure. Implementation details are explored, highlighting the ability to represent complex spatial structures. This work contributes to the understanding and effective application of Octree in three-dimensional visualization environments, offering a valuable tool for the representation and analysis of spatial data.

***Index Terms*—Octree, hierarchical data structure, three-dimensional visualization, Visualization Toolkit (VTK), spatial data.**

—*Resumen* – Este trabajo aborda la implementación de un Octree, una estructura de datos jerárquica tridimensional, y su integración con el kit de herramientas de visualización VTK (Visualization Toolkit). El enfoque se centra en la representación eficiente de datos tridimensionales mediante Octree y su aplicación práctica en la visualización de figuras tridimensionales graficando los nodos hoja del Octree utilizando VTK resaltando su utilidad para la representación gráfica intuitiva de la información contenida en la estructura Octree. Se exploran los detalles de la implementación, destacando la capacidad de representar complejas estructuras espaciales. Este trabajo contribuye a la comprensión y aplicación efectiva de Octree en entornos de visualización tridimensional, ofreciendo una herramienta valiosa para la representación y análisis de datos espaciales.

—*Palabras Clave* – Octree, estructura de datos jerárquica, visualización tridimensional, kit de herramientas de visualización (VTK), datos espaciales.

I. Introducción

La manipulación eficiente de datos tridimensionales es esencial en numerosos campos de la ciencia de la computación, desde la visualización hasta la simulación y análisis de entornos tridimensionales complejos. En este contexto, las estructuras de datos jerárquicas, como los octrees, han demostrado ser herramientas poderosas para organizar y gestionar información 3D de manera eficiente. Este trabajo aborda el desafío de implementar un octree con funciones de inserción y búsqueda, aprovechando la capacidad de la estructura para representar datos tridimensionales de manera compacta y eficaz. Además, para facilitar la comprensión y análisis de los resultados, se ha integrado el sistema de visualización VTK (Visualization Toolkit), permitiendo la representación gráfica de los cubos formados por los nodos hoja del octree. A lo largo de este artículo, presentamos en detalle nuestra implementación, destacando las funciones clave desarrolladas y evaluando el rendimiento de la estructura en escenarios prácticos.

II. Marco Teorico

A. Octree

Un Octree (o árbol octal) es una estructura de datos utilizada para organizar información tridimensional en un espacio cartesiano. Su nombre proviene de la palabra "octant", ya que el espacio tridimensional se divide recursivamente en ocho octantes. Cada octante puede contener datos o subdividirse en ocho octantes más pequeños.

1) Nodos:

- Cada nodo del Octree representa un cubo en el espacio tridimensional.
- Un nodo puede estar en uno de los siguientes estados:
  - Nodo hoja: Contiene información específica, como un punto o un conjunto de datos.
  - Nodo interno: Actúa como un contenedor para subdividir el espacio en ocho octantes más pequeños.

2) Subdivisión:

- Para los nodos internos, el espacio tridimensional se divide en ocho octantes más pequeños.
- Cada octante representa una región más pequeña dentro del nodo padre.

3) Jerarquía:

- Los nodos forman una jerarquía, con el nodo raíz representando todo el espacio tridimensional.
- La profundidad del árbol depende de la resolución requerida y la distribución de los datos.

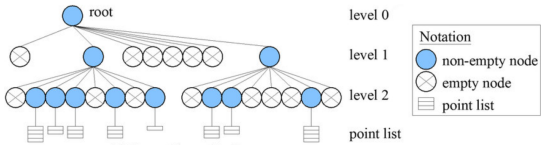

Figure 1: Octree

4) Eficiencia Espacial:

- El Octree es eficiente para estructurar datos en regiones espaciales donde la densidad de datos varía significativamente.

- Permite una búsqueda rápida en áreas específicas del espacio.

5) Aplicaciones:

- Utilizado en gráficos por computadora para la representación de escenas 3D.
- En procesamiento de imágenes para estructurar datos tridimensionales.
- En simulaciones físicas y juegos para la gestión eficiente de objetos en el espacio.

B. Algoritmos de Inserción y Búsqueda

1) Inserción en Octrees: La operación de inserción en un octree implica colocar un objeto tridimensional en la estructura, ajustando la jerarquía del árbol según sea necesario. Algoritmos eficientes de inserción consideran la adaptabilidad del octree para minimizar el número de subdivisiones y mantener la eficiencia en la recuperación de datos.

2) Búsqueda en Octrees: La búsqueda en un octree implica determinar la ubicación y relación espacial de los objetos dentro de la estructura. Algoritmos de búsqueda optimizados aprovechan la jerarquía del árbol para descartar rápidamente regiones irrelevantes y reducir el tiempo de búsqueda.

C. Visualización Tridimensional

1) Visualization Toolkit (VTK): VTK es una biblioteca de software ampliamente utilizada para la visualización 3D de datos científicos y médicos. Proporciona herramientas para representar gráficamente estructuras jerárquicas, como octrees, facilitando la comprensión visual de datos complejos.

2) Representación de Octrees en VTK: La representación de octrees en VTK implica la conversión de la estructura jerárquica en objetos gráficos tridimensionales. Esto permite la visualización clara de la distribución espacial de datos y la interacción visual con la estructura.

III. METODOLOGIA

En esta sección, se describe la metodología utilizada para llevar a cabo el presente trabajo. La metodología es fundamental para entender cómo se abordó la implementación del octree además contribuye a la claridad y transparencia del trabajo, permitiendo a los lectores replicar y comprender el proceso seguido para llegar a los resultados presentados.

A continuación, se detallan las clases implementadas, la estructura del nodo Octree y las funciones específicas del Octree. Cada componente desempeña un papel crucial en la organización y manipulación efectiva de datos tridimensionales.

A. Clases

1) Point: En la siguiente figura 2, nos muestra como se diseño la clase Point.

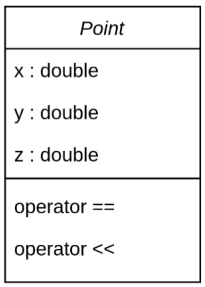

Figure 2: Clase Point

2) Octree: En la siguiente figura 3, nos muestra como se diseño la clase Octree.

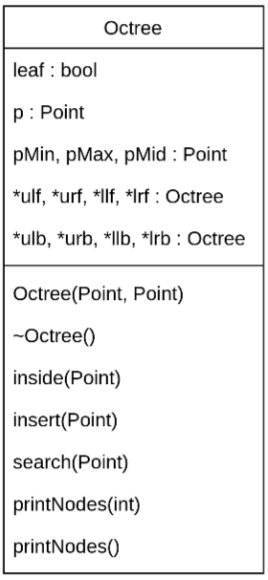

Figure 3: Clase Octree

B. Nodo del Octree

La distribución de los nodos sigue el principio de subdivisión recursiva, donde cada nodo se divide en ocho subnodos hasta alcanzar un nivel específico o cumplir ciertas condiciones. Esta estructura jerárquica permite una representación eficiente y organizada del espacio tridimensional, por ello en la figura 4 vemos como se nombra y divide el espacio del nodo actual en sus 8 particiones.

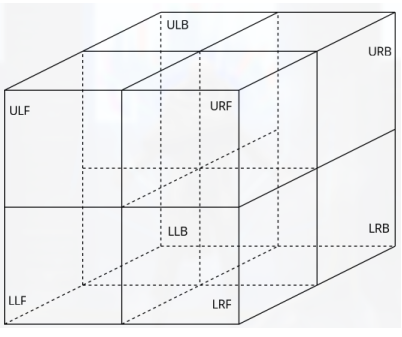

Figure 4: Distribución de nodos en el Octree

C. Funciones del Octree

1) Constructor: El contructor inicializa y calcula los límites iniciales del octree al asignar los puntos mínimos y máximos, y luego calcula el punto medio para facilitar futuras operaciones y divisiones en el espacio tridimensional.

Algorithm 1: Constructor de la clase Octree

Data: Puntos *min* y *max*
Result: Inicialización de *pMin*, *pMax*, y *pMid*
begin
$pMin \leftarrow$ min;
$pMax \leftarrow$ max;
$pMid.x \leftarrow (pMin.x + pMax.x)/2$;
$pMid.y \leftarrow (pMin.y + pMax.y)/2$;
$pMid.z \leftarrow (pMin.z + pMax.z)/2$;

2) Destructor: El destructor libera la memoria asignada dinámicamente a cada uno de los nodos hijos. Este proceso se realiza de manera recursiva, ya que cada nodo hijo también invocará su propio destructor para liberar la memoria de sus propios nodos hijos.

Algorithm 2: Destructor de la clase Octree

Result: Liberación de memoria de nodos hijos
begin
delete ulf;
delete urf;
delete llf;
delete lrf;
delete ulb;
delete urb;
delete llb;
delete lrb;

3) Función inside: La función inside se utiliza para determinar si un punto específico está contenido dentro del volumen representado por un nodo del octree.

Algorithm 3: Función `inside` de la clase Octree

Data: Punto *p*
Result: Verdadero si *p* está dentro del octree, falso en caso contrario
begin
return $((pMin.x \leq p.x) \wedge (p.x < pMax.x) \wedge (pMin.y \leq p.y) \wedge (p.y < pMax.y) \wedge (pMin.z \leq p.z) \wedge (p.z < pMax.z))$;

4) Función insert: La función insert inserta puntos en un octree tridimensional, dividiendo recursivamente el espacio en octantes y distribuyendo los puntos en los subnodos correspondientes. Además, maneja casos especiales para nodos perfectos y evita la inserción de puntos duplicados.

Algorithm 4: Función `insert` de la clase Octree

Data: Punto *newP*
Result: Inserción del punto *newP* en el Octree
begin
if *no es hoja y* `ulf` *no está inicializado* **then**
hoja ← verdadero;
`p` ← *newP*;
return;
if *newP es igual a* `p` **then**
`imprimir` "Duplicado";
return;
hoja ← falso;
if `ulf` *es nulo* **then**
`ulf` ← **nuevo** Octree({pMin.x,pMid.y,pMin.z},{pMid.x, pMax.y, pMid.z});
`urf` ← **nuevo** Octree({pMid.x,pMid.y,pMin.z},{pMax.x, pMax.y, pMid.z});
`llf` ← **nuevo** Octree({pMin.x, pMin.y, pMin.z}, {pMid.x, pMid.y, pMid.z});
`lrf` ← **nuevo** Octree({pMid.x, pMin.y, pMin.z}, {pMax.x, pMid.y, pMid.z});
`ulb` ← **nuevo** Octree({pMin.x,pMid.y,pMid.z},{pMid.x, pMax.y, pMax.z});
`urb` ← **nuevo** Octree({pMid.x, pMid.y, pMid.z}, {pMax.x, pMax.y, pMax.z});
`llb` ← **nuevo** Octree({pMin.x, pMin.y, pMid.z}, {pMid.x, pMid.y, pMax.z});
`lrb` ← **nuevo** Octree({pMid.x, pMin.y, pMid.z}, {pMax.x, pMid.y, pMax.z});
if `ulf->inside(p)` **then**
`ulf->insert(p)`;
if `urf->inside(p)` **then**
`urf->insert(p)`;
if `llf->inside(p)` **then**
`llf->insert(p)`;
if `lrf->inside(p)` **then**
`lrf->insert(p)`;
if `ulb->inside(p)` **then**
`ulb->insert(p)`;
if `urb->inside(p)` **then**
`urb->insert(p)`;
if `llb->inside(p)` **then**
`llb->insert(p)`;
if `lrb->inside(p)` **then**
`lrb->insert(p)`;
if `ulf->inside(newP)` **then**
`ulf->insert(newP)`;
if `urf->inside(newP)` **then**
`urf->insert(newP)`;
if `llf->inside(newP)` **then**
`llf->insert(newP)`;
if `lrf->inside(newP)` **then**
`lrf->insert(newP)`;
if `ulb->inside(newP)` **then**
`ulb->insert(newP)`;
if `urb->inside(newP)` **then**
`urb->insert(newP)`;
if `llb->inside(newP)` **then**
`llb->insert(newP)`;
if `lrb->inside(newP)` **then**
`lrb->insert(newP)`;

5) Función search: La función search busca un punto dentro del octree, navegando recursivamente a través de los nodos y subnodos según sea necesario. Si el punto se encuentra, devuelve true; de lo contrario, devuelve false.

Algorithm 5: Función `search` de la clase Octree

```
Data: Punto auxP
Result: Verdadero si auxP está en el Octree, falso en
        caso contrario
begin
    if es hoja then
        if this->p == auxP then
            return verdadero;
        else
            return falso;
    if ulf y ulf->inside(auxP) then
        return ulf->search(auxP);
    if urf y urf->inside(auxP) then
        return urf->search(auxP);
    if llf y llf->inside(auxP) then
        return llf->search(auxP);
    if lrf y lrf->inside(auxP) then
        return lrf->search(auxP);
    if ulb y ulb->inside(auxP) then
        return ulb->search(auxP);
    if urb y urb->inside(auxP) then
        return urb->search(auxP);
    if llb y llb->inside(auxP) then
        return llb->search(auxP);
    if lrb y lrb->inside(auxP) then
        return lrb->search(auxP);
    return falso;
```

6) Función printNodes: La función printNodes tiene el propósito de recorrer los nodos del octree, y en el caso de ser nodos hoja mostrar la información del punto que contiene incluyendo el nivel en el que se encuentra.

Algorithm 6: Función `printNodes` de la clase Octree

```
Data: Int level
begin
    if es hoja then
        imprimir "level: p";
        return;
    if ulf then
        level ← level + 1;
        ulf->printNodes(level);
        urf->printNodes(level);
        llf->printNodes(level);
        lrf->printNodes(level);
        ulb->printNodes(level);
        urb->printNodes(level);
        llb->printNodes(level);
        lrb->printNodes(level);
```

D. Integración con VTK para Visualización de Nodos

En este apartado, se describe las funciones de integración del Octree con la biblioteca de visualización VTK (*Visualization Toolkit*). El objetivo es representar gráficamente los nodos del Octree para facilitar la comprensión de su estructura y distribución en el espacio tridimensional.

1) Función `calculateCubePoints`: Esta función toma dos puntos en el espacio tridimensional, $p1$ y $p2$, que definen las esquinas inferior izquierda y superior derecha de un cubo, respectivamente. Luego, calcula las coordenadas de las otras seis esquinas del cubo y las almacena en una matriz `pointsArray`.

2) Función `printNodesVtk`: Esta función imprime los nodos de un Octree en un objeto VTK para su visualización. Toma como entrada un puntero a un Octree (`q`), un objeto VTK Renderer (`renderer`), y un tipo que indica cómo imprimir los nodos (dimensiones del cubo, punto medio del cubo, o el propio punto del nodo).

3) Función `printVTK`: Esta función realiza la integración del Octree con VTK para la visualización de nodos. Toma como entrada un puntero al nodo raíz del Octree (`root`), un nombre de archivo (`nameFile`) y un tipo que indica cómo imprimir los nodos. La función utiliza un renderizador VTK para representar gráficamente los nodos y muestra la ventana de visualización.

En resumen, el código integra el Octree con el entorno de Visualización de Toolkit (VTK) para permitir la representación gráfica de los nodos del Octree. Utiliza las funciones de VTK para crear puntos, definir caras de cubos y agregar actores al renderizador para cada nodo del Octree.

Todo el código se puede encontrar en el siguiente repositorio de Github: https://github.com/PaulParizacaMozo/Octree

IV. Resultados

En esta sección, se presentan los resultados obtenidos durante la implementación del Octree en cuanto a la inserción de datos, la búsqueda eficiente y la visualización de nodos hoja. La sección se organiza en subsecciones dedicadas mostrar los resultados obtenidos al visualizar los cubos formados por la dimension de los nodo hoja, el punto medio del cubo, y el punto exacto que almacena dicho cubo(nodo hoja).

Para ello tuvimos 2 datasets, uno que representa a una figura de un gato y otro que representa la figura de un dragón.

A. Gato

Para esta figura se tuvo un dataset de 1136 puntos, recordemos que los puntos duplicados no se insertan, teniendo esto en cuenta tuvimos los siguiente resultados:

1) Cubo con dimensión nodo hoja: Se toma en cuenta la dimensión total del nodo hoja, tomando las dimensiones del punto mínimo hasta el punto máximo.

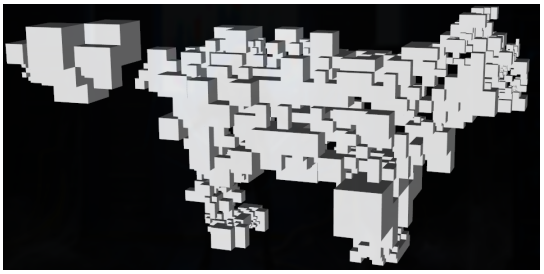

Figure 5: Gato - Nodos hoja

2) Punto medio del cubo del nodo hoja: Se toma en cuenta el punto medio del cubo con el mismo punto aumentado en 3 unidades.

Figure 6: Gato - Puntos medio de Nodos hoja

3) Punto exacto en el cubo del nodo hoja: Se toma en cuenta el punto exacto del punto en el nodo hoja con el mismo punto aumentado en 3 unidades.

Figure 7: Gato - Puntos exactos de Nodos hoja

B. Dragón

Para esta figura se tuvo un dataset de 10193 puntos, recordemos que los puntos duplicados no se insertan, teniendo esto en cuenta tuvimos los siguiente resultados:

1) Cubo con dimensión nodo hoja: Se toma en cuenta la dimensión total del nodo hoja, tomando las dimensiones del punto mínimo hasta el punto máximo.

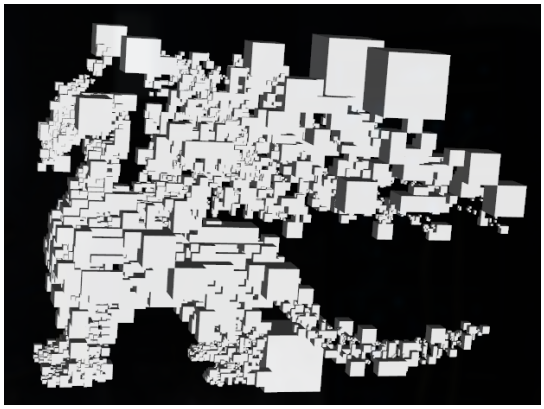

Figure 8: Dragón - Nodos hoja

2) Punto medio del cubo del nodo hoja: Se toma en cuenta el punto medio del cubo con el mismo punto aumentado en 3 unidades.

Figure 9: Dragón - Puntos medio de Nodos hoja

3) Punto exacto en el cubo del nodo hoja: Se toma en cuenta el punto exacto del punto en el nodo hoja con el mismo punto aumentado en 3 unidades.

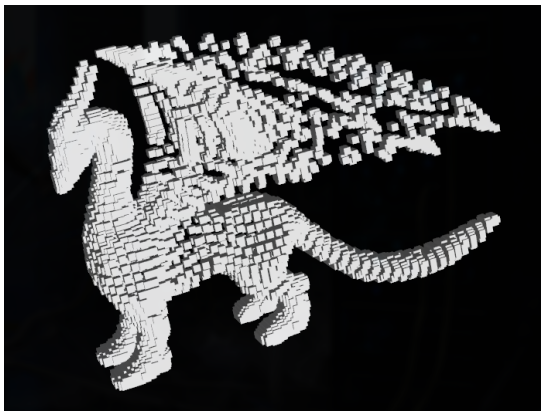

Figure 10: Dragón - Puntos exactos de Nodos hoja

En el siguiente repositorio de Github pueden encontrar el proceso que se llevo para la ejecución de estos resultados. https://github.com/PaulParizacaMozo/Octree

V. Conclusiones

En este trabajo, se abordó la implementación del Octree. Los objetivos iniciales incluían entender el funcionamiento del Octree e implementar las funciones principales como Insertar y Buscar ,además de la visualización de datos en un entorno tridimensional. A través de la implementación y pruebas realizadas, se lograron varios resultados destacados.

La visualización de nodos hoja a través de la integración con VTK proporcionó una herramienta valiosa para comprender la distribución espacial de los datos almacenados en el Octree. Se exploraron diferentes enfoques para la representación gráfica, desde las dimensiones del cubo hasta el punto medio y la ubicación específica del nodo.

Aunque los resultados son prometedores, se identificaron desafíos, como la gestión de grandes conjuntos de datos y ciertas limitaciones en la representación visual para conjuntos de datos muy densos. Nos encontramos con que hay sectores en donde se puede ver una diferencia entre el nivel mínimo de un nodo hoja y el nivel máximo de un nodo hoja, ello se puede apreciar en el apartado de resultados donde vemos cubos de diferentes dimensiones.

En el futuro, se sugiere explorar estrategias para optimizar el rendimiento con conjuntos de datos más grandes y abordar las limitaciones observadas. Además, el Octree podría extenderse para incluir funcionalidades adicionales, como la adaptación dinámica a cambios en la densidad de datos.

En conclusión, este trabajo destaca el valor del Octree en la organización espacial eficiente de datos tridimensionales. La combinación de inserción, búsqueda y visualización proporciona una herramienta versátil con aplicaciones potenciales en campos como gráficos por computadora, procesamiento de imágenes y simulaciones físicas.

References

[1] Sebastian Keller, Aurélien Cavelan, Rubén Cabezon, Lucio Mayer, Florina M. Ciorba, *Cornerstone: Octree Construction Algorithms for Scalable Particle Simulations*, 2023, pp. 10,

[2] Hanana Samet, *Foundations of Multidimensional and Metric Data Structures*, 2006, pp. 0-1022,

[3] GeeksForGeeks, *Octree | Insertion and Searching*, 2022, https://www.geeksforgeeks.org/octree-insertion-and-searching/, Consultado el 25 de Noviembre de 2023.

[4] Youtube, *Quadtrees in 30 minutes*, 2022, https://youtu.be/ybCdeu6amQc, Consultado el 25 de Noviembre de 2023.