Universidad de las Ciencias Informáticas

Facultad 5

Visualización Indirecta de Volumen mediante la técnica de Planar Rendering.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Javier Rogelio Hernández Romero

Tutor: Ing. Osvaldo Pereira Barzaga

Marzo, 2010

“Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.”

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Autor: Javier Rogelio Hernández Romero. Tutor: Ing. Osvaldo Pereira Barzaga

DATOS DE CONTACTO

Tutor: Ing. Osvaldo Pereira Barzaga

Edad: 25 años.

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Profesor Instructor

E-mail: opereira@uci.cu

Graduado de la UCI, con seis años de experiencia en el tema de la Gráfica Computacional y líder de un proyecto de Visualización Médica en la Universidad de las Ciencias Informáticas.

AGRADECIMIENTOS

A mis padres:

A ustedes, mis padres queridos, que tantas veces me han apoyado en mis decisiones, en mis estudios, en mi vida.

A mi novia:

A ti, Yaisa, por permitirme conocer el mundo de un modo distinto y enseñarme que el mismo está lleno de colores y matices.

A mis familiares:

Por estar siempre cuando los necesité y permitirme ser miembro de una familia tan unida.

A mis profes:

Por enseñarme tantas cosas que a lo largo de mi vida me han formado para ser un profesional comprometido con su profesión, sobre todo a mi tutor Osvaldo y a la profe Yaima, por apoyarme tanto en mi tesis.

A mis amigos:

Por permitirme conocer que es la amistad, sobre todo cuando a veces se está tan lejos del hogar.

A la UCI:

Por permitirme aprovechar estos cinco años para formarme como un profesional comprometido con la Revolución.

DEDICATORIA

A mi madre:

Por ser la flor que me trajo al mundo, por cobijarme bajo su sobra y darme su abrigo toda mi vida.

A mi padre:

Por ser mi amigo desde siempre, por caminar a mi lado cuando era pequeño, por guiarme y quererme tanto.

A mi novia:

Por ser parte de mis sueños y apoyarme siempre en todo lo que me propongo hacer.

RESUMEN

Las aplicaciones de visualización médica han adquirido un elevado auge en la medicina a nivel mundial, ya que les permite a los médicos especialistas realizar diagnósticos preoperatorios no invasivos y de alta precisión desde una perspectiva 3D. La idea principal de la misma es obtener un modelo tridimensional de alta resolución gráfica a partir de imágenes médicas digitales de las modalidades de Tomografía Axial Computarizada y Resonancia Magnética. Los usuarios de este tipo de aplicaciones demandan que las interacciones con los modelos tridimensionales sean en tiempo real y sobre un hardware convencional de bajo costo en el mercado.

En este trabajo de diploma se propone un algoritmo para la visualización de volúmenes conocido como Planar Rendering, aplicándolo a la visualización de modelos anatómico 3d a partir de imágenes DICOM. Con el objetivo de lograr una reconstrucción realista y en tiempo real de la anotomía específica de cada paciente.

Durante la implementación del algoritmo de Planar Rendering se realizaron algunas modificaciones al algoritmo clásico con el objetivo de aumentar el rendimiento del mismo, referidos a la optimización del uso de memoria y a lograr una frecuencia de muestreo constante durante la visualización del volumen de los datos. Además se combina esta técnica con la de planos de corte con el objetivo de dotar al médico especialista de una herramienta que le permita la exploración interior de las estructuras anatómicas que son de su interés durante el diagnóstico.

PALABRAS CLAVE

Frecuencia de Muestreo, Función de Transferencia, Geometría Proxy, Mapeado de Texturas, Planar Rendering, Vértice, Voxel.

ÍNDICE

DECLARACIÓN DE AUTORÍA I

DATOS DE CONTACTO II

AGRADECIMIENTOS III

DEDICATORIA IV

RESUMEN V

ÍNDICE VI

ÍNDICE DE FIGURAS IX

ÍNDICE DE TABLAS XI

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 4

1.1 VISUALIZACIÓN. 4

1.1.1 Surface Rendering. *5*

1.1.1.1 Marching Cubes. 5

1.1.1.2 Marching Tetrahedra. 7

1.1.2 Volume Rendering. *9*

1.1.2.1 Ray Casting. 9

1.1.3 Planar Rendering. *11*

1.1.3.1 Estados de los algoritmos de la técnica Planar Rendering. 12

1.1.3.2 Geometría proxy. 13

1.2 PIPELINE GRÁFICO Y GPUS PROGRAMABLES. 14

1.3 METODOLOGÍAS Y HERRAMIENTAS DE DESARROLLO. 16

1.3.1 Metodología de Desarrollo de Software. *16*

1.3.2 Herramientas de desarrollo. *16*

1.3.3 Lenguajes. *17*

1.3.3.1 Lenguajes de programación. 17

1.3.3.2 Lenguaje de modelado. 17

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA..... 18

2.1 ADQUISICIÓN DE LOS DATOS. 18

2.1.1 Interpolación Trilineal. *20*

2.2 PLANAR RENDERING. 21

2.2.1 Planar 2D. *21*

2.2.2 Planar 3D. *26*

2.3 FUNCIÓN DE TRANSFERENCIA. 29

2.4 CLIPPING PLANE. 30

2.5 CONSIDERACIONES PARA LA IMPLEMENTACIÓN. 31

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA..... 32

3.1 REGLAS DEL NEGOCIO. 32

3.2 MODELO DE DOMINIO. 32

3.2.1 Descripción del Modelo de Dominio. *33*

3.3 CAPTURA DE REQUISITOS. 33

3.3.1 Requisitos Funcionales. *34*

3.3.1 Requisitos No Funcionales. *34*

3.4 MODELO DE CASOS DE USO DEL SISTEMA. 36

3.4.1 Actores del Sistema. *36*

3.4.2 Diagrama de Casos de Uso del Sistema. *36*

3.4.3 Descripción de casos de Uso del Sistema. *37*

3.5 DIAGRAMA DE CLASES DE ANÁLISIS. 41

3.6 DIAGRAMAS DE COLABORACIÓN. 42

3.6.1 Diagrama de Colaboración del Caso de Uso Visualizar Modelo 3D. Sección Realizar Corte. *42*

3.6.2 Diagrama de Colaboración del Caso de Uso Visualizar Modelo 3D. Sección Rotar Volumen. *43*

3.7 DIAGRAMA DE CLASES DEL DISEÑO DEL SISTEMA. 44

3.7.1 Diagrama de Clases del CU Visualizar Modelo 3D. *44*

3.8 DIAGRAMAS DE SECUENCIA DEL DISEÑO. 45

3.8.1 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección realizar Corte. *45*

3.8.2 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección Rotar Volumen. *46*

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA 47

4.1 DIAGRAMA DE COMPONENTES. 47
4.1.1 Componentes del paquete de Implementación Visualización. *48*
4.2 VALORACIÓN DE LOS RESULTADOS. 49
4.2.1 Datos de entrada. *49*
4.2.2 Parámetros a medir. *50*
CONCLUSIONES **53**
RECOMENDACIONES **55**
BIBLIOGRAFÍA **56**
GLOSARIO **58**
ANEXOS **65**
ANEXO A: IMÁGENES DE LA APLICACIÓN. 65

ÍNDICE DE FIGURAS

Figura 1.1Técnicas de la visualización volumétrica. 5
Figura 1.2 Tabla de de búsqueda del algoritmo Marching Cubes. 6
Figura 1.3 Tabla de de búsqueda del algoritmo Marching Tetrahedra. 8
Figura 1.4 Pasos del algoritmo Ray Casting. 10
Figura 1.5 Pasos de implementación de los algoritmos de la técnica Planar Rendering. 12
Figura 1.6 Geometría proxy usada en el mapeo 3D, donde los planos de corte están orientados perpendicularmente al vector de visión. 13
Figura 1.7 Geometría proxy usada en el mapeo 2D, donde los planos de corte están orientados perpendicularmente al los ejes principales. 14
Figura 1.8 Pipeline gráfico del GPU. 15
Figura 2.1 Izquierda: Grid 2D de una imagen. Derecha: Volumen de datos, donde los vóxeles están agrupados en un grid 3D. 19
Figura 2.2 Volumen de la celda. 19
Figura 2.3 Interpolación trilineal en un voxel. 21
Figura 2.4 Objeto con cortes alineados usados como geometría proxy con mapeo de textura 2D. 22
Figura 2.5 Conjunto de pilas de cortes, cada una alineada con un eje principal. 22
Figura 2.6 Cambiando la pila de los cortes del objeto con cortes alineados en dependencia de la dirección de visión. Entre la imagen (C) y la (D) la pila usada para realizar el render ha sido cambiada. 23
Figura 2.7 Seudocódigo de la variante de implementación usando planos alineados a los ejes. 24
Figura 2.8 La frecuencia de muestreo no es constante. 25
Figura 2.9 La frecuencia de muestreo es constante. 25
Figura 2.10 Cortes alineados al punto de visión usados como geometría proxy con mapeo de textura 3D. Derecha: Cortes poligonales planos, Centro: textura 3D, Izquierda: imagen final. 26
Figura 2.11 Lugares de muestreo en cortes alineados al punto de visión para la proyección ortogonal (A) y la proyección de perspectiva (B). 27
Figura 2.12 La frecuencia de muestreo es constante. 27
Figura 2.13 Cortes alineados al punto de visión. 29
Figura 2.14 Editor de función de transferencia 1D. 30
Figura 2.15 Utilización del clipping plane para visualizar el interior de una estructura volumétrica. 31
Figura 3.1 Modelo de Dominio. 33
Figura 3.2 Diagrama de Casos de Uso del Sistema. 37

Figura 3.3 Diagrama de Clases del Análisis.....41
Figura 3.4 Diagrama de Colaboración del caso de uso Visualizar Modelo 3D. Sección Realizar Corte.....42
Figura 3.5 Diagrama de Colaboración del caso de uso Visualizar Modelo 3D. Sección Rotar Volumen.....43
Figura 3.6 Diagrama de Clases del Diseño del Caso de Uso Visualizar Modelo 3D.....44
Figura 3.7 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección Realizar Corte.....45
Figura 3.8 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección Rotar Volumen.....46
Figura 4.1 Paquete del diagrama de componentes. 47
Figura 4.2 Diagrama de componentes. Paquete Visualización. 48
Figura A.1 Visualización 3D de un modelo de Tórax utilizando el mapeo 2D. Se ha utilizado la función de transferencia para lograr un acercamiento a la realidad de la anatomía humana. 65
Figura A.2 Visualización 3D de un modelo de Tórax utilizando el mapeo 2D. 66
Figura A.3 Visualización 3D de un modelo de un motor de automóvil utilizando el mapeo 3D. La visualización se ha llevado a cabo utilizando la escala de grises.....66
Figura A.4 Visualización 3D de un modelo de un motor de automóvil utilizando el mapeo 3D. Obsérvese cómo se logra la transparencia en el modelo.....67
Figura A.5 Visualización 3D de un modelo de un bonsai utilizando el mapeo 3D. 67
Figura A.6 Utilización de los planos de recorte en un eje, para explorar el modelo 3D. 68
Figura A.7 Utilización de los planos de recorte en dos ejes, para explorar el modelo 3D. 68

ÍNDICE DE TABLAS

Tabla 3.1 Actor del Sistema. 36
Tabla 3.2 Descripción del CU Visualizar modelo 3D. 39
Tabla 3.3 Descripción de la sección Realizar Corte. 40
Tabla 3.4 Descripción de la sección Rotar Volumen. 41
Tabla 4.1 Datos de Entrada. 49
Tabla 4.2 Sumario de la técnica Planar Rendering en sus dos variantes de implementación. 51
Tabla 4.3 Resultados de la cantidad de cuadros por segundo, consumo de recursos de hardware y tiempo de pre procesamiento para los datos de entrada. 52

INTRODUCCIÓN

Las tecnologías de visualización tridimensional han demostrado ser una gran herramienta en el campo de la medicina. Hoy en día, la utilización de los modelos tridimensionales para representar las imágenes del cuerpo humano le permite a los especialistas en medicina tener una mejor apreciación de las patologías del paciente y poder hacer un examen más exacto del enfermo; ya que se logra una visualización muy aproximada a la realidad de la estructura anatómica que se quiere consultar, utilizándose para esto diversas técnicas de visualización de volúmenes.

Los equipos que actualmente se utilizan para obtener información en forma de imágenes médicas, por lo general poseen un sistema informático, que permite realizar la visualización 3D del interior de un paciente a partir de las imagines 2D obtenidas por resonancias magnéticas o tomografías computarizadas, el cual en muchos casos es caro y en otros, quedan anulados sus contratos, esto se evidencia en la ruptura del contrato de la Empresa Philips con Cuba, lo cual ha afectado el soporte tanto para el hardware como para el software de los productos suministrados por esta empresa al país.

Esta situación se ve agudizada dado a que Cuba es un país bloqueado económicamente. En el mismo, el servicio de Salud Pública a la población es priorizado, por lo que constantemente se buscan soluciones y alternativas. Ejemplo de esto es el proyecto productivo VISMEDIC, en el cual se pretende elaborar una herramienta de visualización médica 3D, interactiva y en tiempo real, basada en software libre y optimizada de tal manera que se pueda ejecutar en una estación de trabajo convencional con un mínimo de requerimientos de hardware.

En la actualidad este proyecto cuenta con una herramienta de visualización que, a partir de imágenes DICOM, realiza la reconstrucción y generación de la superficie geométrica de modelos anatómicos 3D, la cual no facilita la interacción en tiempo real por lo que disminuye el nivel de realismo de la simulación.

Sin embargo, con el desarrollo del hardware gráfico a nivel mundial se han propuesto diversos métodos y algoritmos de visualización 3D basados en GPU que proponen una solución a la problemática anterior, la cual puede ser utilizada para erradicar las deficiencias actuales del modulo de reconstrucción del proyecto VISMEDIC.

Dada la situación problemática descrita anteriormente surge el **problema:** ¿Cómo lograr una visualización interactiva en tiempo real de los modelos anatómicos 3D representados en el Proyecto VISMEDIC? siendo el **objeto de investigación:** Las técnicas y algoritmos de visualización de modelos 3D, y **el campo de acción:** Las técnicas y algoritmos de visualización de modelos anatómicos 3D basados en GPU. Donde como **objetivo** se pretende**:** Desarrollar un módulo para la visualización interactiva y en tiempo real de modelos anatómicos 3D a partir de imágenes médicas digitales, y como **idea a defender:** El empleo de la técnica del Planar Rendering como algoritmo de visualización y reconstrucción de modelos 3D a partir de modelos DICOM, producirá una visualización de dichos modelos realista, interactiva y en tiempo real.

Para el cumplimiento del objetivo se trazaron un conjunto de **tareas investigativas:**

- Caracterizar las distintas técnicas de visualización tridimensional basadas en GPU para valorar su posible utilización en el módulo a desarrollar.
- Seleccionar la técnica adecuada a implementar para lograr una visualización de gran calidad visual, interactiva y en tiempo real de modelos anatómicos 3D a partir de imágenes médicas digitales.
- Seleccionar las metodologías y las herramientas de desarrollo para garantizar que el ciclo de desarrollo del módulo se concluya en el tiempo establecido en el cronograma y cumpla con los requerimientos funcionales establecidos por el usuario final de la aplicación.
- Implementar un módulo para la visualización de las estructuras anatómicas a partir de estudios DICOM, utilizando las distintas variantes del Planar Rendering.
- Integrar el módulo a la aplicación ya existente para potenciar la capacidad de diagnóstico e interactividad en tiempo real.
- Realizar las pruebas de calidad para certificar la validez del trabajo realizado.

Durante la investigación se utilizaron diversos **métodos teóricos y empíricos:**

Métodos teóricos:

- **Histórico – Lógico:** Con la aplicación de este método se analizó la trayectoria histórica real y la evolución de la investigación, así como el desarrollo de los elementos más importantes de la misma, tales como: la visualización 3D de los modelos anatómicos.
- **Modelación:** Este método se utilizó para hacer una representación simplificada de la realidad, representando el conocimiento adquirido y diferentes diagramas que complementan el proceso de elaboración del módulo, tales como: diagramas de clases, diagramas de secuencia y diagramas de colaboración.

Métodos empíricos:

- **Observación**: Este método se utilizó para definir, a partir de la observación de los resultados en la caracterización e identificación de los algoritmos utilizados en la reconstrucción 3D, cuál o cuáles serán más adecuados.
- **Experimento**: Método empírico mediante el cual se realizó pruebas a los algoritmos que utiliza la técnica del Planar Rendering en la visualización 3D para, en dependencia de los resultados, escoger cuál o cuáles son los más adecuados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo abordará las características de técnicas de visualización de volúmenes más utilizadas para obtener una visualización interactiva y en tiempo real en la esfera médica, tales como el Volume Rendering, el Surface Rendering y el Planar Rendering. Se describirán además, algunas características de las unidades de procesamiento gráfico (GPU) y del conjunto de transformaciones que se le aplican a los vértices y a las texturas para determinar el valor de los píxeles de la imagen final (pipeline gráfico); y por último se abordarán las metodologías y herramientas de desarrollo que serán utilizadas durante toda la investigación e implementación del módulo.

1.1 Visualización.

Las técnicas de visualización tridimensional (3D) son muy utilizadas para el desarrollo de aplicaciones científicas e ingenieriles que requieran la visualización de datos volumétricos, como es el caso de la visualización de los estudios médicos adquiridos por los dispositivos de imágenes médicas, tales como: Resonancias Magnéticas (MRI) [1], Tomografías Computarizadas (CT) [2], Tomografías de Emisión de Positrones (TEP) [3], y Ultrasonidos [4], permitiendo la exploración de los pacientes para la obtención de un diagnóstico preciso.

Es por ello que la visualización médica se ha convertido en un proceso muy importante para el desarrollo de la medicina, donde se utiliza la visualización de volúmenes, que según [19] se divide en dos grupos de técnicas. Ver **Figura 1.1.**

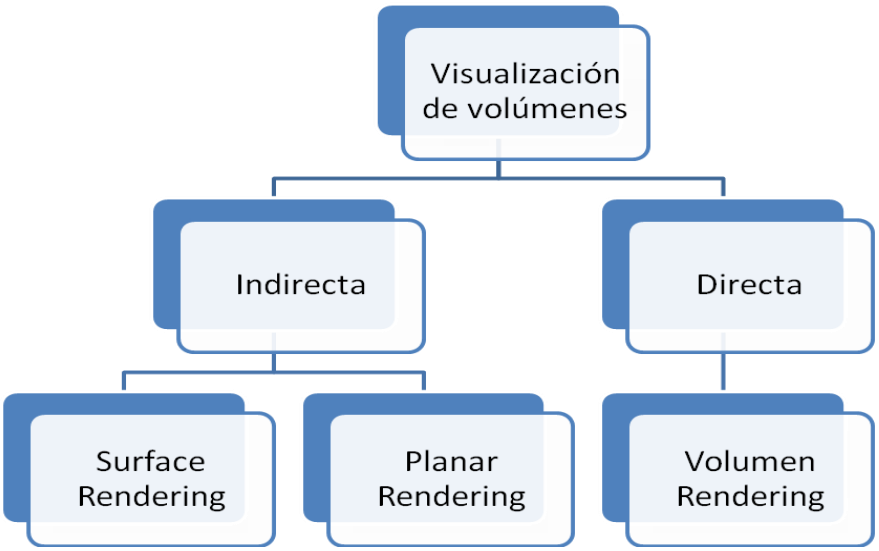

Figura 1.1Técnicas de la visualización volumétrica.

1.1.1 Surface Rendering.

Actualmente existen varios algoritmos para resolver el problema de la reconstrucción de superficies de un volumen de datos, dos de los primeros algoritmos más usados en la actualidad son: el algoritmo Marching Cubes (ver epígrafe **1.1.1.1**) propuesto por William E. Lorensen y Harvey E. Cline en 1987 [5] y el Marching Tetrahedra [6], descrito en el epígrafe **1.1.1.2**.

1.1.1.1 Marching Cubes.

Este algoritmo se basa en el estudio del volumen adquirido mediante subdivisión en pequeños vóxeles o cubos y consta además con una tabla de búsqueda (ver **Figura 1.2**) donde aparecen los 15 casos posibles en que un cubo puede ser interceptado por la superficie.

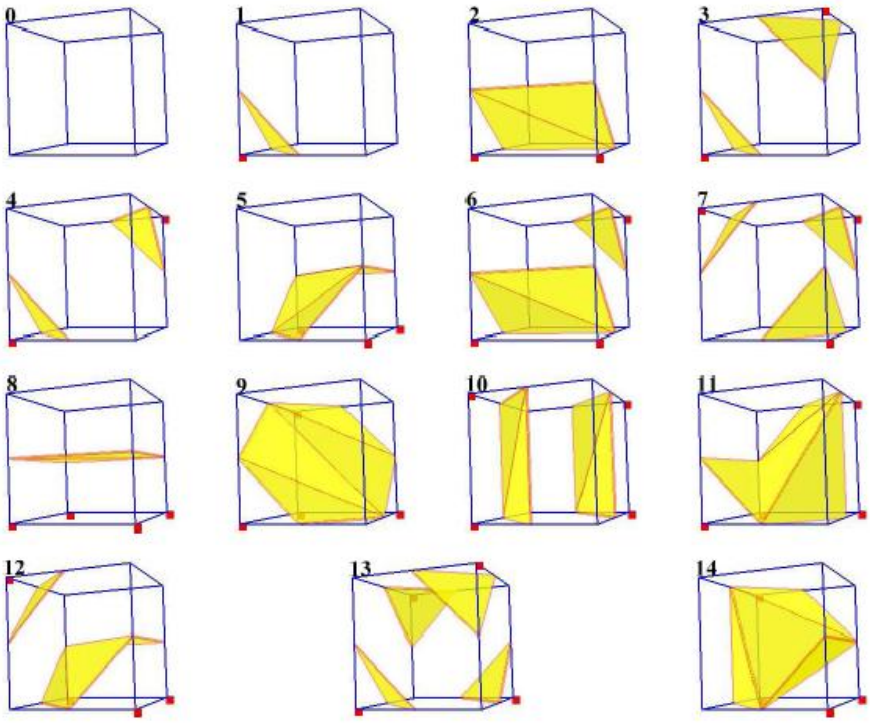

Figura 1.2 Tabla de de búsqueda del algoritmo Marching Cubes.

El Marching Cubes posee dos pasos principales. El primero es decidir cómo definir la sección o secciones de la superficie que cortan un cubo, donde existen 256 posibles combinaciones, y si se clasifican los vértices como dentro o fuera de la superficie. Existe la posibilidad que todos los puntos estén dentro o que todos estén fuera. Para el resto de las combinaciones es necesario calcular, para cada arista del cubo, si ésta corta la superficie. El segundo paso ocurre en una segunda pasada, donde, por medio de interpolación se calcula el punto por donde la superficie corta las aristas, también se puede calcular tomando el punto medio de las aristas como vértice, y una vez que se tienen todos los puntos por donde la superficie corta el cubo solamente se necesita unir estos vértices para formar los triángulos.

El algoritmo Marching Cubes, según [5], brinda una manera muy sencilla y eficiente de convertir los datos volumétricos en una malla triangular regular, pero éste presenta problemas con la topología de esta malla generada porque posee casos ambiguos en su tabla de configuración, los cuales pueden generar huecos en la malla final. Existen dos tipos de ambigüedades en ciertos casos de la tabla de configuración donde

existe más de una manera de triangular los cubos. El primero es la ambigüedad en la cara, ocurre cuando una cara tiene dos vértices diagonales opuestos positivos y otros dos negativos, lo que puede dar lugar a agujeros y a inconsistencia en la topología de la malla [7]. La otra la constituye las ambigüedades internas que ocurren en el interior de una celda. Una de las formas en la que puede ser resuelta este tipo de ambigüedad es adicionando un punto dentro de cada celda. [8,9]

A raíz de este algoritmo se han propuesto varias soluciones que resuelven de una forma u otra las dificultades que el mismo presenta, no obstante, dado que este algoritmo, a pesar de generar una imagen final de alta resolución, durante el proceso reconstrucción genera un elevado número de triángulos, lo cual afecta la interactividad en tiempo real por lo que se recomienda aplicar un método de reducción de la malla triangular.

1.1.1.2 Marching Tetrahedra.

La idea del Marching Tetrahedra [6], es muy similar al Marching Cubes excepto que en este caso la estructura fundamental para subdividir el volumen de datos son tetraedros, y no cubos, donde un tetraedro solamente tiene 16 triangulaciones posibles. Cuenta con una tabla de búsqueda (ver **Figura 1.3**) donde aparecen los 8 casos posibles en el que un tetraedro puede ser cortado por la superficie.

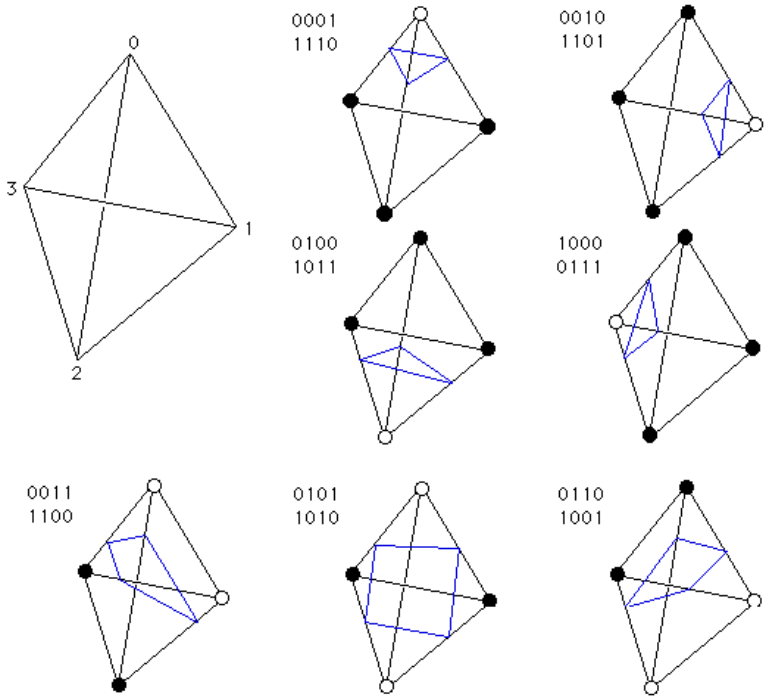

Figura 1.3 Tabla de de búsqueda del algoritmo Marching Tetrahedra.

Existen dos posibles divisiones de la malla de cubos, en una primera, el algoritmo divide cada celda en seis pequeños tetraedros y las aristas de estos se alinean con las celdas adyacentes, introduciéndose ambigüedades debido a que la simetría de las subdivisiones de los cubos debe alternarse entre cubos para alinear las caras de los tetraedros.

Una segunda forma de dividir la malla consiste en dividir los cubos en tetraedros centrados, los cuales se encuentran centrados respecto del cubo. En este caso los tetraedros siempre van a tener la misma forma, lo cual elimina las ambigüedades de la malla de triángulos resultante.

La principal desventaja de esta técnica es que crea una cantidad de triángulos mayor que el Marching Cubes para unos datos de entrada determinados, lo que tiene una influencia directa en la regularidad de la malla de la superficie resultante.

1.1.2 Volume Rendering.

Las técnicas de Volume Rendering han alcanzado un nivel elevado de utilización por la gran calidad e interactividad que se logra en las visualizaciones de los distintos volúmenes de datos en 3D, donde su complejidad depende de la cantidad de elementos del volumen (vóxeles) y el número de píxeles de la imagen donde serán proyectados los datos, o sea, la resolución del plano de la imagen.

En la visualización directa de volumen, basada en Volume Rendering, los datos volumétricos son representados mediante su evaluación en un modelo óptico, que describe cómo el volumen emite, refleja, dispersa, absorbe y ocluye la luz [10], lo que incrementa el nivel de realismo físico en la visualización, donde las propiedades ópticas como el color y la opacidad afectan la luz que pasa a través del medio debido a la absorción, dispersión y emisión de partículas. [11]

Una de las técnicas de visualización que se emplea en el Volume Rendering es la conocida como Ray Casting, la cual se describe a continuación.

1.1.2.1 Ray Casting.

Esta técnica para la visualización de volúmenes de datos 3D está basada en el concepto de un rayo de luz que penetra en el volumen e intercepta los vóxeles que encuentra a su paso. [12]

Por cada píxel de la ventana, a través de la cual se observa la representación tridimensional del estudio, se lanza un rayo dentro del volumen, recorriendo una trayectoria rectilínea que comienza desde la posición de la cámara virtual y paralelo al vector direccional de la misma. Es muy usado en las implementaciones, que los rayos viajen en sentido contrario al vector direccional de la cámara, proyectando la imagen resultante en la ventana del observador.

El algoritmo Ray Casting está compuesto por tres pasos: (1) determinar la dirección del rayo (ver **Figura 1.4 A**), (2) determinar los valores encontrados a lo largo del rayo (ver **Figura 1.4 B**) y (3) procesar estos valores y proyectarlos sobre la imagen final (ver **Figura 1.4 C**).

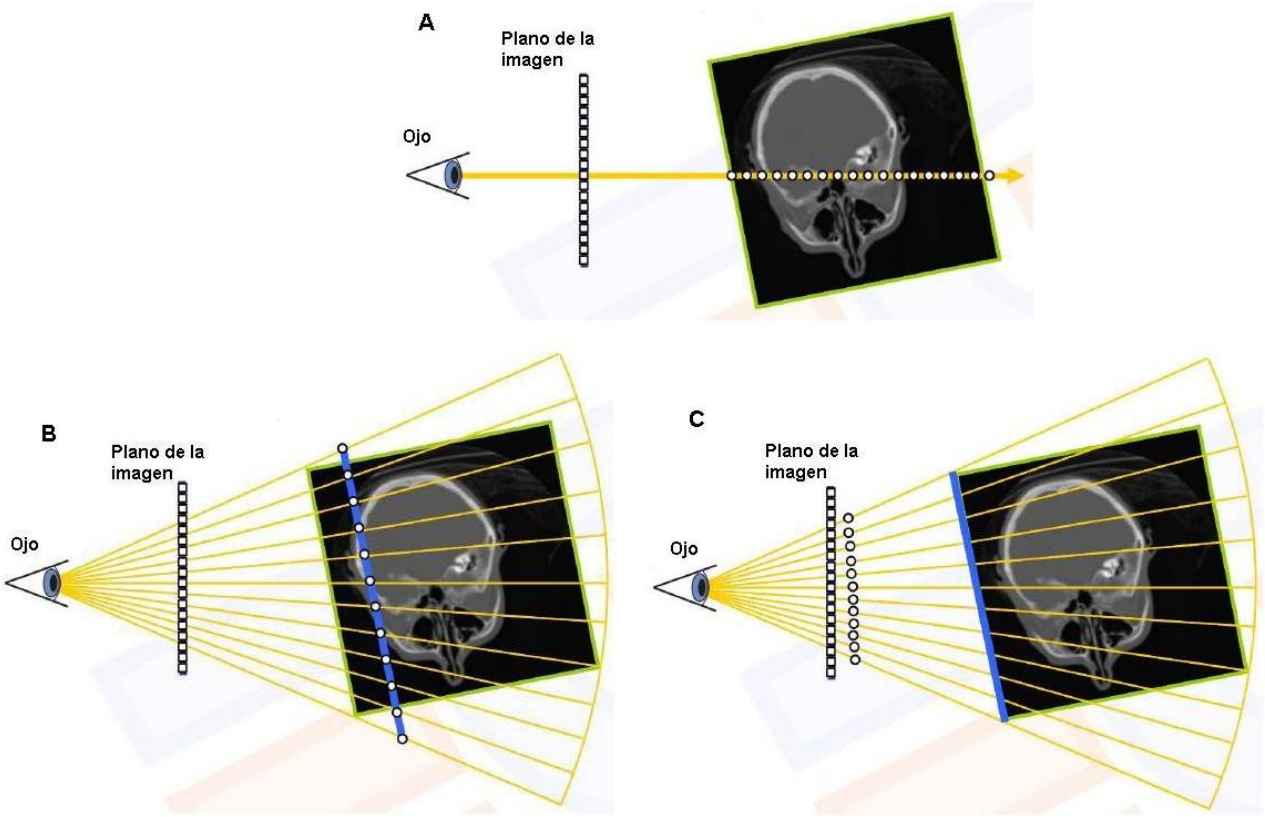

Figura 1.4 Pasos del algoritmo Ray Casting.

Existen varios métodos para extraer estos valores del volumen de datos:

- **Proyección de Intensidad Promedio:** Asigna el promedio de todos los valores encontrados a lo largo del rayo al correspondiente píxel en la imagen final. En esta función las estructuras que posean mayor intensidad aumentarán el promedio, siendo más visibles.

- **Composición:** Esta función determina el píxel final basado en una mezcla de todas las muestras de color y opacidad encontradas a lo largo del rayo. Para cada muestra, un color y una opacidad son determinados por una función especial llamada función de transferencia. Una importante característica de este método es que tiene la habilidad de visualizar diferentes estructuras con diferentes colores y opacidades y así obtener una visualización de alta calidad.

- **Proyección de Intensidad Máxima:** Por sus siglas en inglés (MIP), asigna el máximo valor encontrado a lo largo del rayo en el correspondiente píxel de la imagen, por lo que resultado final es una representación semitransparente del volumen de datos que no provee información acerca de la profundidad de las estructuras, lo que conduce a que dos proyecciones MIP desde direcciones opuestas para una misma orientación sean idénticas. Puede producir un alto consumo de tiempo ya que cada rayo debe atravesar completamente el volumen de datos en busca del mayor valor. Con la función MIP, estructuras con altas intensidades pueden ser visualizadas eficientemente, sin embargo, estructuras de un bajo valor de intensidad que sean de interés para el observador permanecerán oscuras en la imagen resultante.

- **Umbralización:** Es similar a la función MIP excepto que el máximo valor es escogido de antemano. Aquí el recorrido del rayo termina en cuanto el valor de umbral es encontrado, por esa razón no es muy usado.

- **Isosurface:** Es una forma de umbralización donde todos los vóxeles son mostrados con un determinado valor de umbral.

1.1.3 Planar Rendering.

Actualmente, las aplicaciones demandan la necesidad de poder visualizar grandes volúmenes de información con una elevada calidad de visualización en tiempo real, donde el uso de la técnica Planar Rendering se ha convertido en una solución a este problema, dado a que utiliza el GPU para el procesamiento gráfico, dejándole a la unidad central de procesamiento (CPU) menos carga de procesamiento, de esta forma se logra una mayor interactividad. [13]

La clave de esta técnica de aceleración consiste en la capacidad de aprovechar las interpolaciones bilineal y trilineal realizadas en la GPU. Esta capacidad fue descrita por primera vez por Cullip y Neumann [14], donde plantearon la necesidad de establecer unos esquemas de muestreos, usando planos alineados a los ejes y planos alineados al vector de visión. Además presentaron las cuestiones generales de implementación para el mapeo de textura por hardware y fueron los primeros en generar imágenes usando esta técnica basada en dos diferentes funciones de transferencia. El desarrollo de esta idea, así como la extensión de la visualización médica más avanzada fue descrito por Cabral [15]. Ellos

demostraron que tanto la reconstrucción como la visualización interactiva de volúmenes son posibles mediante la utilización de hardware que provea aceleración 3D de textura.

Esta técnica se compone de dos formas principales de aplicación, una utilizando mapeo 2D de texturas y la otra utilizando mapeo 3D de textura, donde Wilson [16], Van Gelder y Kim [17] han hecho grandes aportes en el desarrollo de las matemáticas para la generación de las coordenadas de textura.

Con ella se obtienen resultados satisfactorios en cuanto a la visualización de volúmenes, ya que ofrece, en dependencia de los recursos de que se dispongan, seleccionar entre una y otra variante, pues el mapeo 3D necesita la existencia de una GPU, la cual provee el soporte para la textura 3D, mientras que el mapeo 2D no necesariamente necesita de una GPU, pues los cálculos se pueden hacer todos en el CPU, aunque el uso de la tarjeta gráfica puede potenciar la interactividad.

1.1.3.1 Estados de los algoritmos de la técnica Planar Rendering.

Los algoritmos del Planar Rendering pueden dividirse en tres estados. Ver **Figura 1.5**. La Inicialización es usualmente ejecutada una sola vez. Los estados de Actualizar y Dibujar son ejecutados cada vez que la aplicación recibe entradas del usuario. [18].

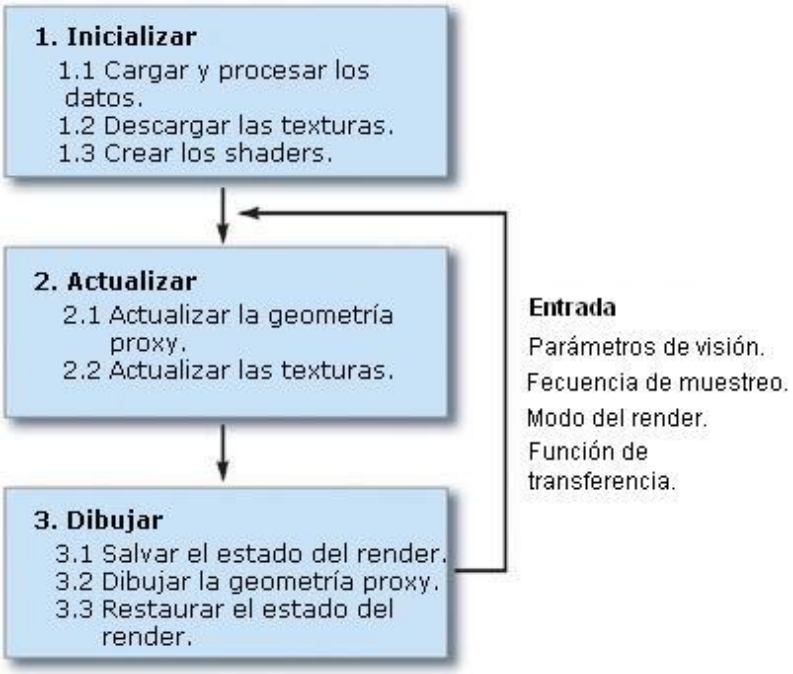

Figura 1.5 Pasos de implementación de los algoritmos de la técnica Planar Rendering.

En la inicialización, el volumen de datos es cargado en la memoria de la CPU, en algunos casos éste necesita ser procesado antes de ser creada la textura, dado a que la GPU no soporta cualquier primitiva para representar el volumen, tolerando solamente líneas, puntos y polígonos, entonces hay que convertir la representación volumétrica en primitivas que sean soportadas por la GPU.

Luego de la inicialización y cada vez que los parámetros de visión cambian, la geometría proxy es recalculada y almacenada en arreglos de vértices.

Durante el estado de actualización, las texturas son actualizadas si el modo de render o los parámetros de la función de transferencia cambian. Antes que los cortes poligonales sean dibujados en un orden, el estado del render debe ser guardado. El volumen y la función de transferencia de texturas deben estar vinculados a las unidades de textura, lo cual utiliza el fragment shader como entrada. Luego se prepara el conjunto de vértices para ser dibujado. Finalmente, luego que los cortes son dibujados en orden, el estado del render es restituido, y así el algoritmo no afecta la visualización de otros objetos en la escena.

1.1.3.2 Geometría proxy.

La técnica Planar Rendering posee dos formas principales de geometría proxy, una de ellas es cuando el volumen de datos es almacenado como un objeto de textura 3D, donde la geometría proxy estará representada por un conjunto de polígonos que cortan el volumen perpendicular a la dirección de visión. [21] En la **Figura 1.6** se muestra un ejemplo de esta geometría proxy.

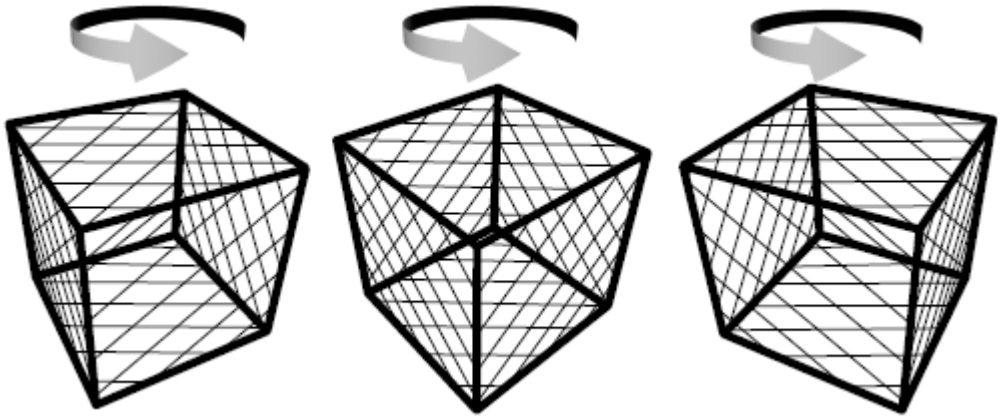

Figura 1.6 Geometría proxy usada en el mapeo 3D, donde los planos de corte están orientados perpendicularmente al vector de visión.

La otra variante es cuando el volumen de datos es almacenado en un conjunto de mapas de texturas en 2D, (ver **Figura 1.7**) donde la geometría proxy estará representada por rectángulos o cortes alineados a los ejes principales. [21]

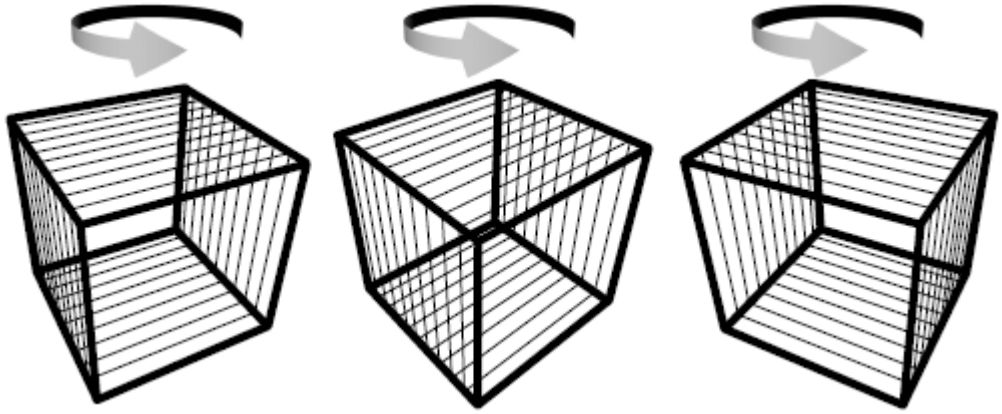

Figura 1.7 Geometría proxy usada en el mapeo 2D, donde los planos de corte están orientados perpendicularmente al los ejes principales.

1.2 Pipeline gráfico y GPUs programables.

Los procesadores gráficos fueron desarrollados primeramente para la industria de los juegos para computadoras, donde en la gran mayoría del hardware grafico antiguo, la geometría enviada al GPU era estática, y las operaciones que se podían realizar en el mismo eran limitadas, por lo que las funciones de los GPU se reducían a determinados pipelines, tales como texturas y luces.

Sin embargo, como los GPUs modernos están diseñados para procesar un gran flujo continuo de datos, en los últimos años las investigaciones los han clasificados como una solución de bajo costo y como una plataforma computacional de alto rendimiento. Entre sus rasgos más distintivos se encuentran que incrementaron la facilidad de programación y la alta precisión de en las operaciones aritméticas, permitiendo el desarrollo de nuevas aplicaciones que no precisamente son juegos.

La nueva generación de tarjetas gráficas ha evolucionado al punto de ser programables tanto el estado de interpretación de los vértices; como el estado de la creación de fragmentos de texturas. Estos elementos

comúnmente son referidos como vertex shader y fragment shader, los cuales pueden ser programados con lenguajes de alto nivel tales como OpenGL Shading Language (GLSL), creado por el OpenGL ARB (Architecture Review Board), donde se utilizan primitivas de hardware tales como vértices, polígonos, texturas y coordenadas de texturas, permitiendo el control del pipeline gráfico a un bajo nivel.

El procesamiento en paralelo de la información en los GPUs se logra precisamente gracias a la arquitectura que posee, donde todo el flujo de datos de entrada es procesado de la misma forma por el kernel **(**fragment program**)** para producir un flujo de salida. En el kernel se procesan un gran número de instrucciones donde se usan registros temporales para almacenar valores intermedios. El GPU toma ventaja del nivel de paralelismo dado a que posee un conjunto considerable de unidades idénticas de procesamiento que ejecutan las instrucciones a la vez.

En la **Figura 1.8** se muestra el pipeline gráfico de un GPU típico. Los vértices y las coordenadas de textura son procesados primeramente por el vertex procesor (vertex shader). Seguidamente en la rasterización se interpola a través de las primitivas definidas por los vértices y se generan fragmentos (píxeles), y por último, el fragment procesor (fragment shader) aplica las texturas y/o realiza el procesamiento que determina el valor final del píxel. Un pase de render es el conjunto de datos que pasan completamente a través del pipeline. [20]

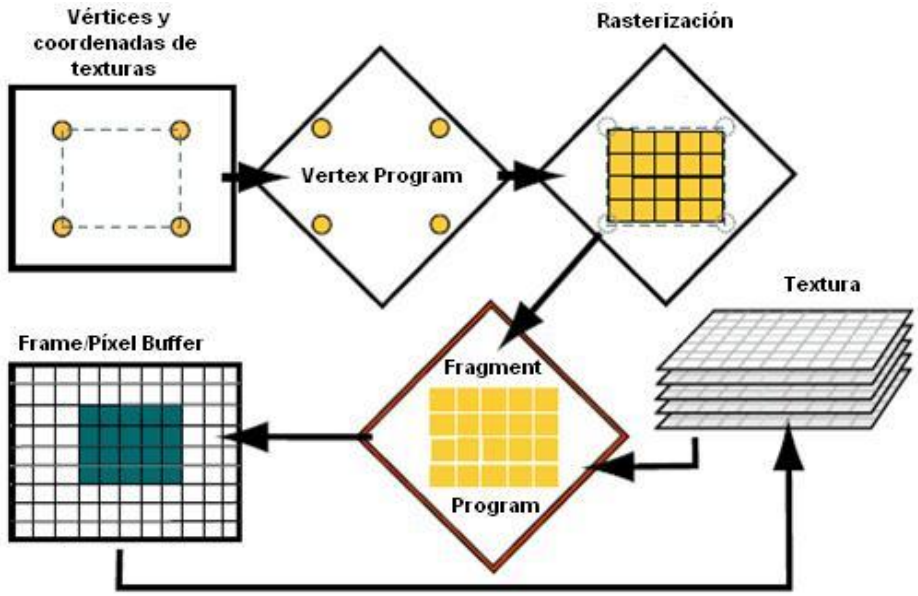

Figura 1.8 Pipeline gráfico del GPU.

1.3 Metodologías y Herramientas de Desarrollo.

1.3.1 Metodología de Desarrollo de Software.

Se utilizó la metodología RUP (Proceso Unificado de Desarrollo) en el desarrollo de la aplicación, dado que es una metodología robusta que unifica los mejores elementos de metodologías anteriores, es orientada a objetos y está preparada para guiar el desarrollo de grandes y complejos proyectos. A continuación se muestran las características que más influyeron en la selección de esta metodología:

1. Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

2. Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

3. Iterativo e Incremental: RUP propone que cada proyecto se desarrolle en fases y que cada fase se desarrolle en iteraciones, donde cada iteración resulta en un incremento del proceso de desarrollo, lo cual se realiza de forma planificada y culmina con el cumplimiento del punto de control trazado en la fase.

4. Orientado a Objetos (OO).

5. Preparado para desarrollar grandes y complejos proyectos.

1.3.2 Herramientas de desarrollo.

Se utilizó como herramienta de modelado Rational Rose Enterprise Edition, pues posee una interfaz con la cual se puede interactuar fácilmente, además de ofrecer un soporte total y sencillo al proceso de desarrollo.

El marco de trabajo (framework) que ofrece QT, fue seleccionado para la implementación del módulo, lo que facilita el diseño y la programación de las interfaces de la aplicación en el lenguaje C++, además, por ser multiplataforma permite la portabilidad posibilitando ser configurado tanto en Windows como en Linux.

1.3.3 Lenguajes.

1.3.3.1 Lenguajes de programación.

Como lenguaje de programación se utilizó C++, ya que es el lenguaje por excelencia para las aplicaciones de realidad virtual, donde se puede optimizar con gran facilidad el uso de la memoria y del CPU, siendo más eficiente el control de grandes volúmenes de datos.

1.3.3.2 Lenguaje de modelado.

Como lenguaje de modelado se escogió UML (Unified Modeling Language), el cual fue utilizado en el modelado del dominio del negocio, así como en el análisis y diseño de la aplicación, dado a que constituye un estándar mundial de modelado, es independiente de la plataforma, permite combinar diversos sistemas gráficos y crear diagramas, se usa solamente para modelar sistemas que usan tecnología OO y su objetivo es visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el actual capítulo se discutirá la solución propuesta, donde se describe las características de la rejilla (grid 3D), la cual, es la entrada que toma la técnica Planar Rendering en sus dos formas de implementación, o sea, con planos alineados a los ejes de coordenadas (Planar Rendering con textura 2D) y con planos alineados al vector de visión (Planar Rendering con textura 3D). De ambas formas de implementación se realiza una descripción detallada, así como algunas consideraciones que mejoran tanto la calidad visual final como la optimización del uso de memoria. Se analizará la estructura de la función de transferencia que se utilizará y algunas características de los planos de recortes (clipping plane).

2.1 Adquisición de los datos.

Un grid 3D es procesado como una secuencia de imágenes o cortes 2D, donde cada imagen representa un corte de una parte del objeto y está compuesto por píxeles (elementos de la imagen). Estos píxeles están organizados en un grid bidimensional, donde la distancia entre los píxeles es típicamente constante en cada dirección. Para la mayoría de las imágenes las direcciones horizontal (***x***) y vertical (***y***) tienen distancias idénticas, lo que es conocido como distancia del pixel (ver **Figura 2.2**), lo que permite el cálculo de la posición actual, multiplicando el valor de la distancia con el índice del píxel. Si asumimos que ***i*** indexa la posición horizontal y ***j*** indexa la vertical, la posición del píxel $P_{i,\,j}$ puede ser calculada. [19] La **Figura 2.1** (izquierda) muestra el grid 2D de la imagen.

Un dato volumétrico combina imágenes individuales en un grid 3D. Ver **Figura 2.1** (derecha). Los elementos del dato ahora son llamados vóxeles (elementos del volumen) y están localizados en un grid de puntos. En adición a las dimensiones horizontal (***x***) y vertical (***y***), ahora también tenemos una dimensión representando la profundidad (***z***).

La distancia entre dos imágenes vecinas es conocida como distancia de corte. Ver **Figura 2.2**

Si la distancia del pixel es idéntica a la distancia de corte se dice que estamos en presencia de un grid isotrópico, de lo contrario es un grid anisotrópico. Ocho vóxeles vecinos forman un paralelepípedo (cuboides), lo que se conoce como celda del grid. Ver **Figura 2.1** (derecha).

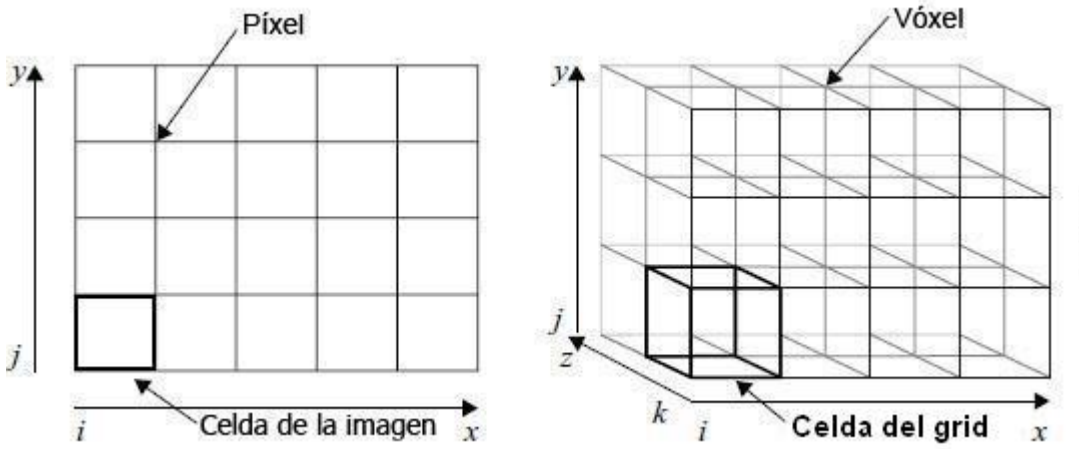

Figura 2.1 Izquierda: Grid 2D de una imagen. Derecha: Volumen de datos, donde los vóxeles están agrupados en un grid 3D.

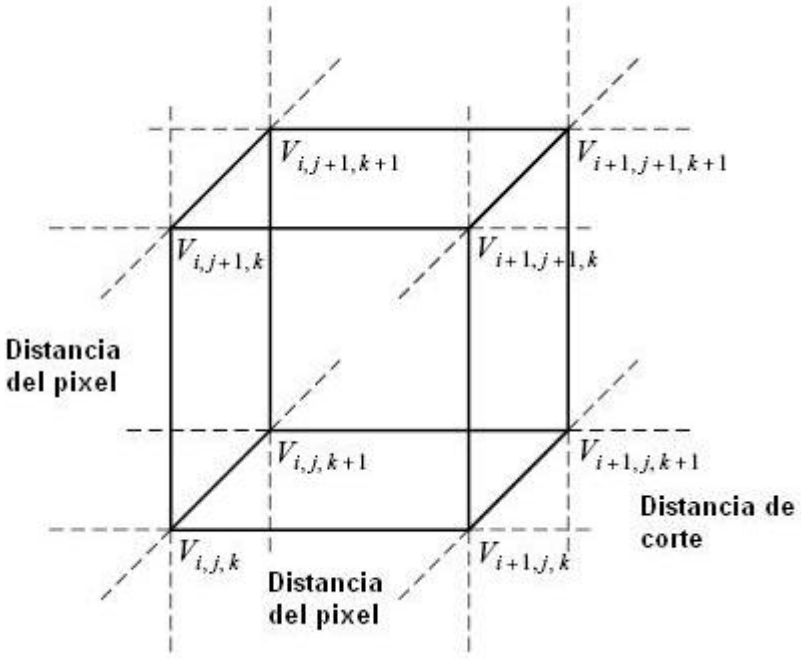

Figura 2.2 Volumen de la celda.

Desde que los puntos del grid se pueden alinear a un sistema de coordenadas cartesiano se le conoce a este tipo de grid como cartesiano o grid uniforme, en el cual:

- El espacio entre cada dimensión es constante.
- La geometría regular puede ser calculada por la cantidad de entradas del grid y la distancia de corte.
- Geometría regular (posee la misma conectividad para todos los puntos del grid).
- Sólo se compone de cuboides.

Un volumen de datos está solamente definido en posiciones discretas del grid 3D. Sin embargo, en muchos casos se necesita calcular puntos dentro de una celda del volumen. Uno de los métodos más usados para calcular dichos puntos es la interpolación trilineal, el cual se describe a continuación.

2.1.1 Interpolación Trilineal.

La interpolación trilineal está compuesta por siete interpolaciones lineales que se realizan en tres pasos, donde cada interpolación lineal está definida por una función donde se calcula el punto medio entre dos puntos del grid 3D. Ver **Figura 2.3**. En el primer paso, cuatro interpolaciones lineales calculan los puntos entre dos vóxeles vecinos en un borde de la celda del volumen a lo largo de una dirección (ya sea ***x***, ***y*** o ***z***). En el segundo paso, dos interpolaciones lineales entre vóxeles localizados en la misma cara de la celda del volumen son combinadas para computar el resultado de la interpolación bilineal. Finalmente, el resultado obtenido de las dos interpolaciones bilineales en las caras opuestas de la celda del volumen es combinado por la interpolación lineal para obtener el punto de muestreo trilineal.

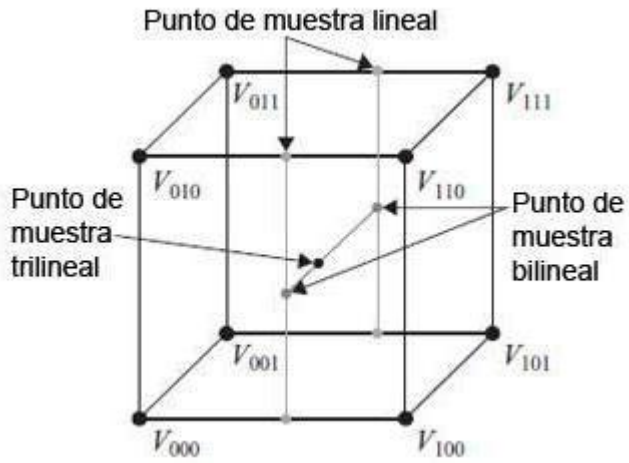

Figura 2.3 Interpolación trilineal en un voxel.

2.2 Planar Rendering.

2.2.1 Planar 2D.

Esta variante de implementación, utilizando el mapeo 2D, donde el volumen debe ser almacenado en un conjunto de mapas de textura de 2 dimensiones, implica que solamente se pueden visualizar planos en 2 dimensiones del volumen original, donde la geometría proxy en este caso, es una pila de cortes planos, los cuales deben estar alineados con uno de los principales ejes del volumen (cualquiera de los ejes ***x***, ***y*** o ***z***), interpolados y mapeados con textura 2D. Ver **Figura 2.4**.

La razón por la cual los cortes son alineados con un eje principal es que cada vez que un corte es dibujado, solamente dos dimensiones están disponibles para coordenadas de texturas, donde la tercera dimensión debe permanecer constante.

En vez de ser usada como una coordenada de textura, la tercera coordenada selecciona la textura a usar de la pila de cortes y las otras dos coordenadas se convierten en las coordenadas 2D de textura actual usada para visualizar el corte.

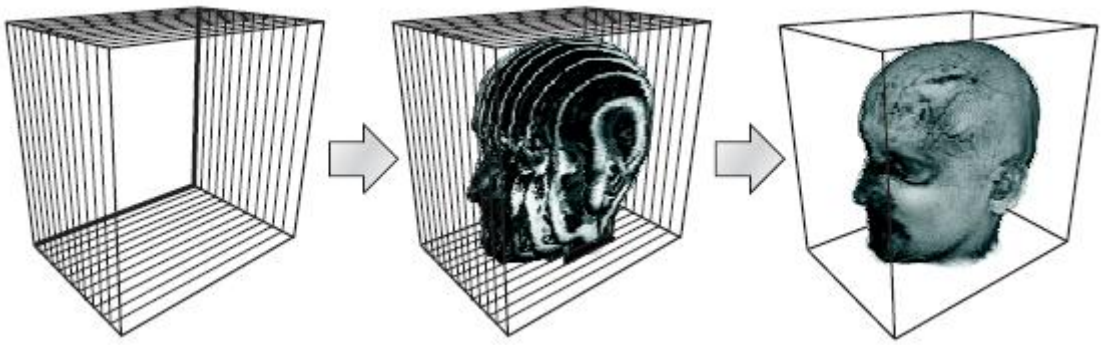

Figura 2.4 Objeto con cortes alineados usados como geometría proxy con mapeo de textura 2D.

Aunque una sola pila de cortes puede almacenar toda la información del volumen, una sola pila no es suficiente, debido a que cuando el punto de visión es rotado sobre el objeto es posible que se vean espacios grandes o no se visualicen correctamente los planos deseados. Para solucionar esto es necesario utilizar tres pilas de cortes, una para cada uno de los ejes principales. Ver **Figura 2.5**.

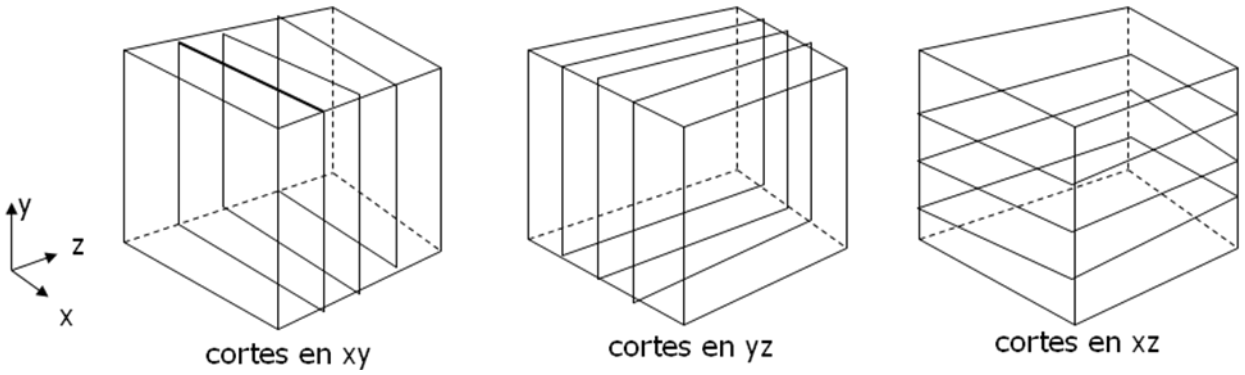

Figura 2.5 Conjunto de pilas de cortes, cada una alineada con un eje principal.

Durante la visualización se va a escoger la pila de cortes que se encuentre más paralela al punto de visión. Ver **Figura 2.6.**

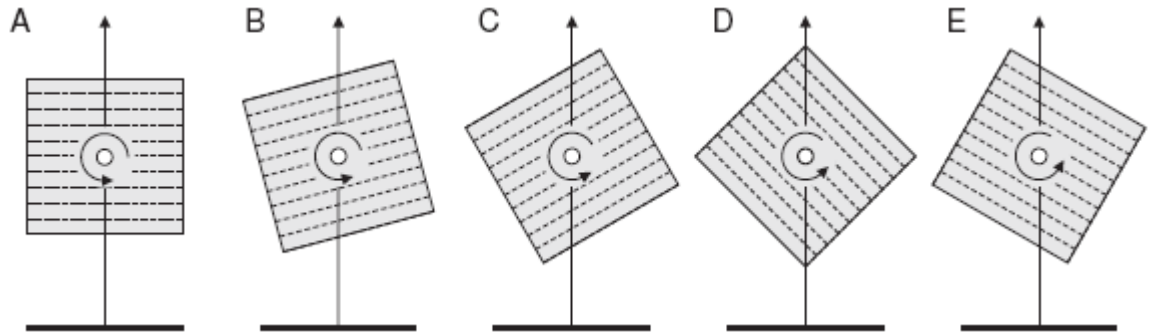

Figura 2.6 Cambiando la pila de los cortes del objeto con cortes alineados en dependencia de la dirección de visión. Entre la imagen (C) y la (D) la pila usada para realizar el render ha sido cambiada.

El uso de tres pilas de cortes, implica que el consumo de memoria supere tres veces la utilizada por una textura 3D. Este problema se puede resolver sobrecargando el procesamiento, recalculando la geometría proxy en cada pasada de render.

Cuando se selecciona una pila para realizar el render es necesario comprobar el sentido del render, para garantizar la correcta visualización de los objetos. Por ejemplo, si una pila es vista desde atrás (respecto a ella misma) se debe dibujar en orden inverso para mantener el resultado deseado.

El pseudocódigo mostrado en la **Figura 2.7** es una vía para el uso de esta técnica. [11]

```
GLfloat modelview_matrix[16];
GLfloat modelview_rotation_matrix[16];
    //obtener la matriz de transformación actual
    //desde el estado de OpenGL
glGet(GL_MODELWIEW_MATRIX,modelview_matrix);
    //extraer la rotación de la matriz
GetRotation(modelview_matrix,modelview_rotation_matrix);
    //rotar la dirección inicial de visión.
Glfloat view_vector[3] = {0.0f,0.0f,-1.0f};
MatVecMultiply(modelview_rotation_matrix,view_vector);
    //encontrar la componete con el mayor valor absoluto del vector
int max_componet = FindAbsMaximun(view_vector);
    //dibujar la pila de cortes de acuerdo con la dirección de visión
switch (max_componet){
    case X:
        if (view_vector[X]>0.0f)
            DrawSliceStack_PositiveX();
        else
            DrawSliceStack_NegativeX();
        break;
    case Y:
        if (view_vector[Y]>0.0f)
            DrawSliceStack_PositiveY();
        else
            DrawSliceStack_NegativeY();
        break;
    case Z:
        if (view_vector[Z]>0.0f)
            DrawSliceStack_PositiveZ();
        else
            DrawSliceStack_NegativeZ();
        break;
}
```

Figura 2.7 Seudocódigo de la variante de implementación usando planos alineados a los ejes.

Uno de los problemas que posee esta variante es que la frecuencia de muestreo depende de la resolución del volumen, dado a que cuando el ángulo formado por el vector de visión y el eje en que se encuentran alineados los planos visualizados, se acerca a 45^{O}, pueden aparecer efectos no deseados, dado a que quedan espacios oscuros entre cada corte. Ver **Figura 2.8.**

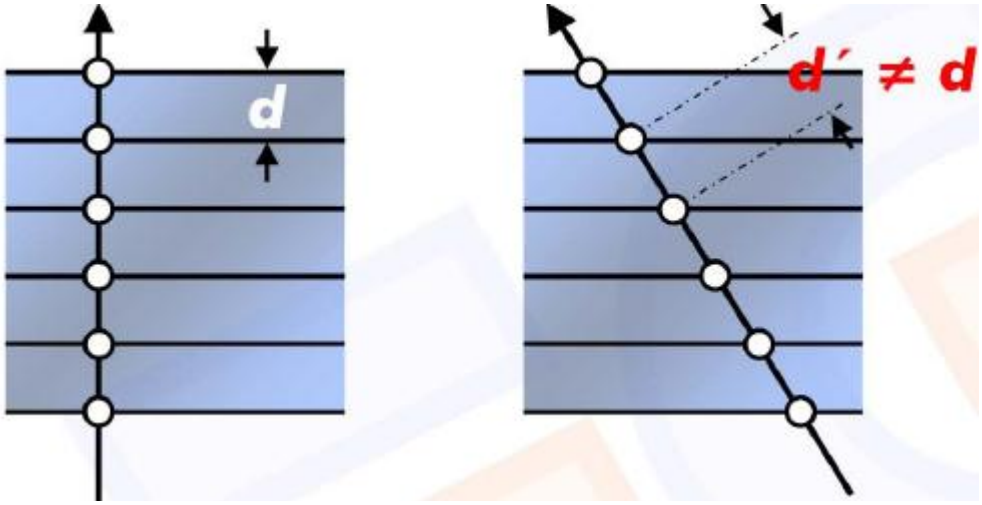

Figura 2.8 La frecuencia de muestreo no es constante.

La solución de este problema es incrementar la cantidad de planos sobre la marcha para mantener constante la frecuencia de muestreo. La idea es que la distancia inicial ***d*** (ver **Figura 2.9** derecha) entre los puntos de dos planos consecutivos determinados por la intersección de estos planos y el vector de visión, se mantenga constante (ver **Figura 2.9** izquierda).

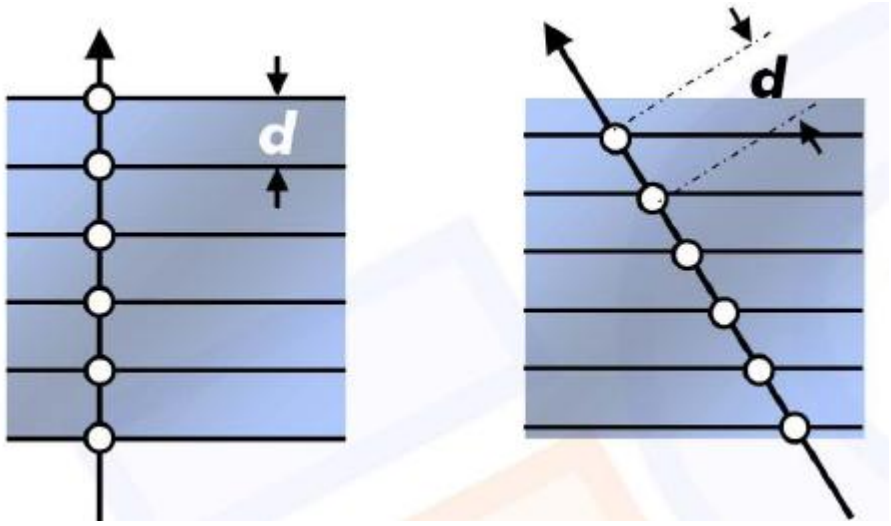

Figura 2.9 La frecuencia de muestreo es constante.

2.2.2 Planar 3D.

Otra variante de implementación está basada en el mapeo 3D, donde la geometría proxy está compuesta por cortes, cuyas normales están alineadas al vector de visión, texturizados con texturas 3D. (Ver **Figura 2.10**).

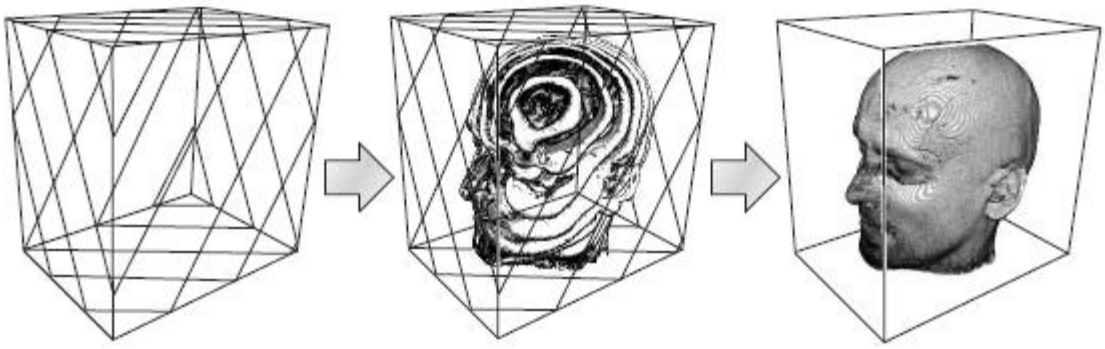

Figura 2.10 Cortes alineados al punto de visión usados como geometría proxy con mapeo de textura 3D. Derecha: Cortes poligonales planos, Centro: textura 3D, Izquierda: imagen final.

En esta variante, el volumen es almacenado en un mapa de textura 3D, donde las coordenadas de texturas 3D son interpoladas trilinealmente en el interior de los polígonos de la geometría proxy. Estas coordenadas de texturas son usadas directamente para indexar el mapa de textura 3D en la correspondiente ubicación y redibujar el volumen con las nuevas coordenadas calculadas.

La gran ventaja del mapeo de textura 3D es que permite que los cortes estén orientados arbitrariamente dentro del dominio de la textura. Así, es natural el uso de cortes alineados al viewport, ya que estos cortes imitan el muestreo utilizado por el algoritmo Ray-Casting [11]. Ofrece además, una distancia constante entre las muestras para la proyección ortogonal y todas las direcciones de visualización. Ver **Figura 2.11** (A). Por lo que la frecuencia de muestreo se mantiene constante. Ver **Figura 2.12**. En el caso de la proyección de perspectiva la distancia entre las muestras sucesivas es diferente para los píxeles adyacentes. Ver **Figura 2.11**(B).

Los efectos no deseados, causados por una incompleta compensación de opacidad, sólo son perceptibles para un campo de gran ángulo de vista dentro de la pirámide de visión.

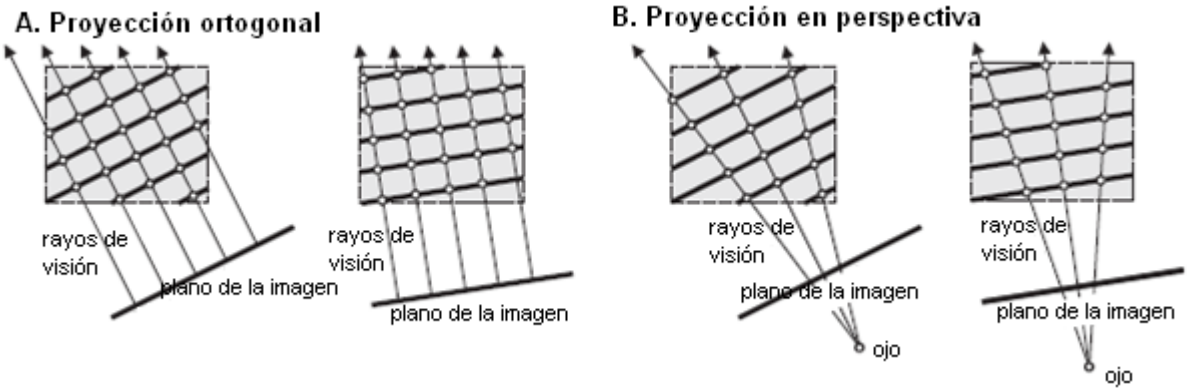

Figura 2.11 Lugares de muestreo en cortes alineados al punto de visión para la proyección ortogonal (A) y la proyección de perspectiva (B).

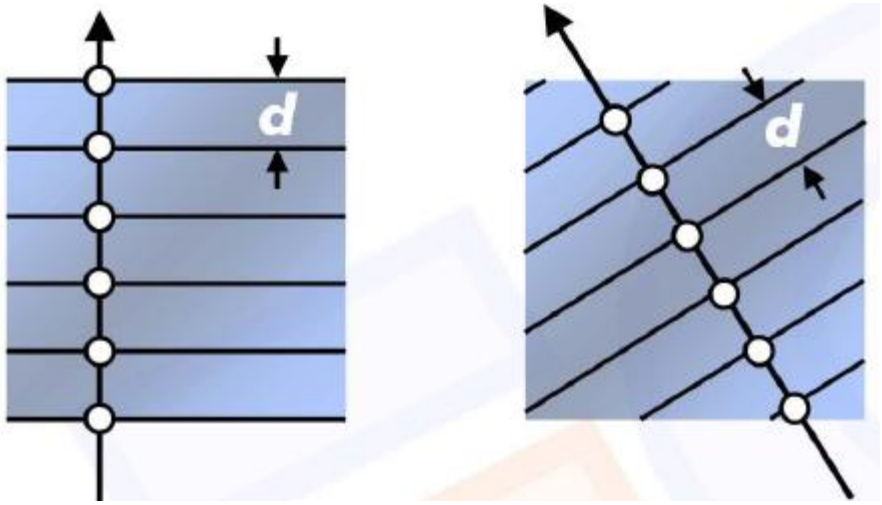

Figura 2.12 La frecuencia de muestreo es constante.

En la actualidad, como el hardware gráfico ya es capaz de realizar la interpolación trilineal completa, la geometría proxy se puede recalcular y el número de cortes se puede ajustar fácilmente sobre la marcha sin ninguna restricción.

El siguiente algoritmo analiza la geometría proxy en el espacio de visión utilizando la matriz de modelado para transformar vértices entre el objeto y el punto de visión. Los polígonos proxy son divididos en triángulos, y los vértices resultantes son almacenados en un arreglo de vértices para un mejor rendimiento. [18]

1-Transformar el volumen de la caja en coordenadas donde el origen estará en el punto de visión, usando la matriz de modelado.

2-Encontrar la mínima y la máxima coordenada en ***z*** de los vértices transformados. Asignar el número de planos usados entre estos dos valores usando espacios equidistantes. El espacio entre los planos es calculado dividiendo la distancia entre los valores extremos encontrados de ***z*** por la cantidad de planos.

3- Para cada plano desde el frente hacia el fondo o viceversa.

a) Chequear la colisión con la caja del volumen. Adicionar cada punto de intersección a una lista temporal de vértices. Se pueden generar hasta seis intersecciones.

b) Buscar el centro del polígono promediando los puntos de intersección. Ordenar los vértices del polígono en sentido horario o antihorario, proyectándolos en el plano *x-y* analizando el ángulo formado con el centro y el primer vértice o el eje **x** como referencia.

c) Dividir el polígono en triángulos y adicionar los vértices resultantes al arreglo de vértices de salida. El polígono puede ser dividido en un conjunto de triángulos en una estructura que semeje un ventilador, lo cual es recomendado dado a que como los triángulos están consecutivos, su procesamiento en el pipeline pudiera ser más rápido para un volumen elevado de datos.

La **Figura 2.13** ilustra el algoritmo descrito anteriormente con dos cortes en forma de polígonos. El primer polígono contiene tres vértices, el segundo está compuesto por seis vértices y es dividido en seis triángulos.

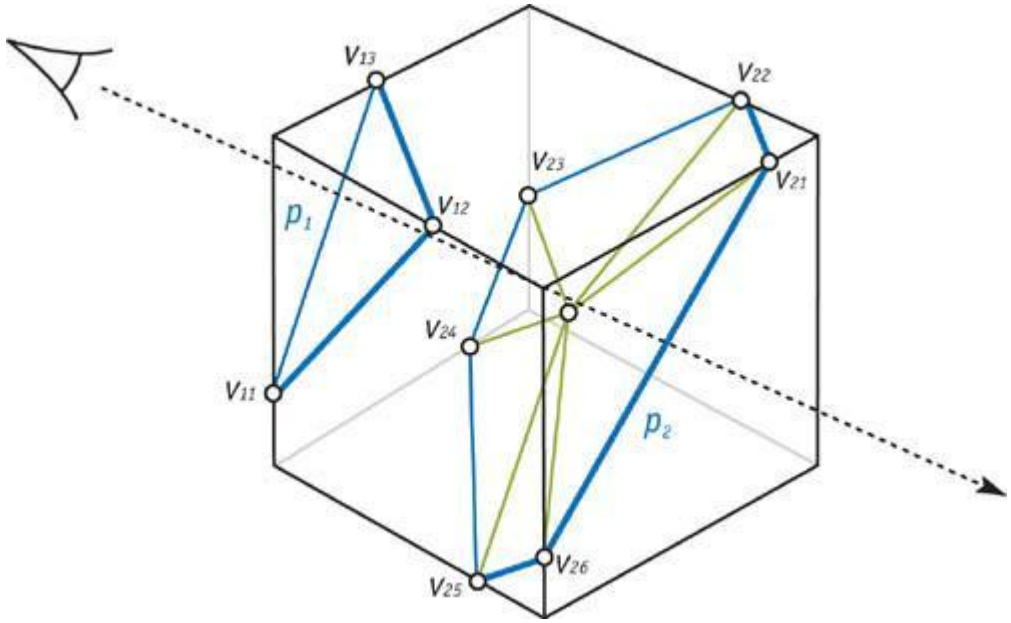

Figura 2.13 Cortes alineados al punto de visión.

Existen distintas formas de generar las coordenadas de texturas para los vértices de los polígonos. Por ejemplo, las coordenadas de texturas pueden ser calculadas en el CPU en el paso 3(c) del algoritmo propuesto anteriormente, desde la posición de los vértices calculados y el volumen de la caja. En este caso las coordenadas son enviadas a la memoria de la GPU en un arreglo separado de vértices o intercalados con la información del vértice.

Hay diferentes métodos para calcular las coordenadas de texturas en la GPU, incluyendo la generación automática de coordenadas de texturas, la matriz de textura o con un programa para el análisis de los vértices.

2.3 Función de transferencia.

Una vez mapeada con textura la geometría proxy utilizada, se hace necesario enfatizar las características específicas dentro del volumen, para ello se utilizó una función de transferencia unidimensional, en la cual se asigna color y opacidad a los valores del volumen, donde se emplea un editor de curva 1D para especificar la función de transferencia a través de un conjunto de puntos de control, donde el eje

horizontal provee la intensidad de gris de los pixeles del volumen de los datos y el eje vertical la frecuencia de aparición de los mismos en el volumen. Ver **Figura 2.14**. Esto permite asignar color y opacidad al conjunto de vóxeles en el rango seleccionado. [11]

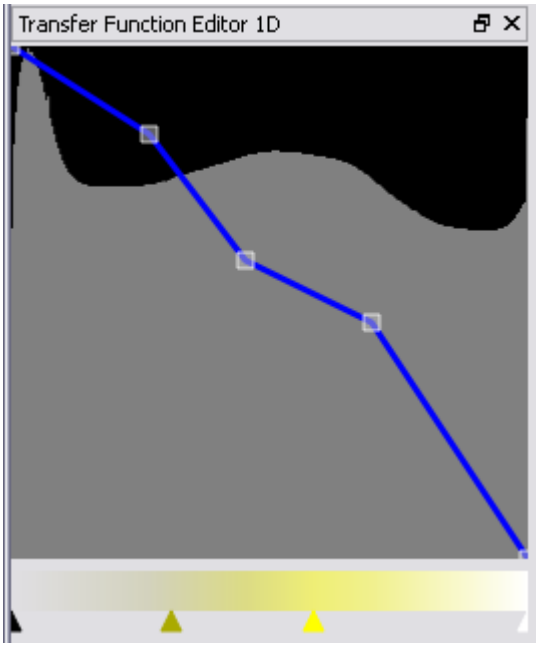

Figura 2.14 Editor de función de transferencia 1D.

2.4 Clipping Plane.

Para lograr comprender mejor la estructura interna de los modelos 3D obtenidos, se utilizaron planos de cortes, donde se considera solamente de interés para la visualización el volumen delimitado por éstos. Básicamente esta idea consiste en asignarle un valor alfa igual a 0 a las zonas que no son de interés, donde se utilizó la función de transferencia, llevando consigo un mayor manejo de la opacidad que se le asigna a cada píxel.

En la **Figura 2.15** se muestra un ejemplo de cómo se han hecho cortes en los planos sagital y coronal, donde, al volumen que queda fuera de los planos de corte se le asigna una opacidad total.

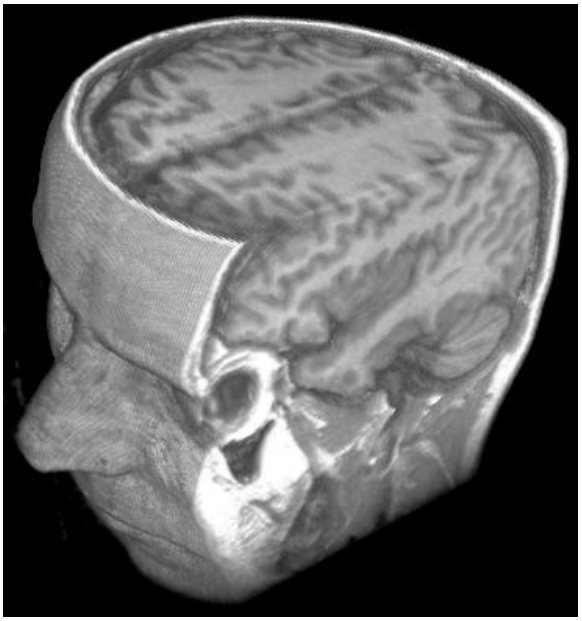

Figura 2.15 Utilización del clipping plane para visualizar el interior de una estructura volumétrica.

2.5 Consideraciones para la implementación.

La técnica del planar rendering utiliza, entre otras primitivas de hardware, los polígonos, los cuales se consideran como primitivas relativamente de gran extensión. Durante la rasterización se producen muchos fragmentos por cada primitiva, lo cual puede formar un cuello de botella en el pipeline, además, las secciones que son transparentes, o sea, que no contienen datos de interés, pueden influir en la generación adicional de fragmentos, que sobrecarguen el procesamiento.

Por todo esto se hace necesario dibujar una geometría proxy que genere solamente los fragmentos requeridos, utilizándose por ejemplo, rectángulos o triángulos, lo que es muy viable si se utiliza la variante 2D en el primer caso y la variante 3D en el segundo caso, otra vía sería la disminución del viewport, con lo cual se limitaría el área de procesamiento, decrementar la frecuencia de muestreo, esto último afectaría la interacción en tiempo real, por lo cual no es muy aconsejable si lo que se quiere es un sistema de interacción en tiempo real, y por último no dibujar las regiones vacías, lo que facilitaría que no se produzcan fragmentos adicionales que no contribuyen en la conformación de la imagen final.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo se hará una descripción de la solución propuesta en el capítulo anterior. Se definen las reglas del negocio, como vía para establecer las condiciones necesarias para el correcto funcionamiento del módulo. Se analizarán los conceptos fundamentales en el área de interés mediante el modelo de dominio. Se describirán tanto los requisitos funcionales como los no funcionales. Se definirán los casos de uso del sistema y los actores que interactúan con el mismo. Se identificarán temas del análisis, como los diagramas de clases del análisis y los diagramas de colaboración. Luego del análisis, donde se tiene ya una idea de la solución, se realizará el diseño del módulo a través de los diagramas de clases del diseño y sus respectivos diagramas de secuencia.

3.1 Reglas del Negocio.

Las imágenes de los estudios que se deseen visualizar deben estar en el formato *.DCM, que es el formato estándar para las imágenes médicas, dado que el módulo que se implementó solamente garantiza el soporte para este tipo de archivos.

3.2 Modelo de Dominio.

A continuación se muestra en la **Figura 3.1** una descripción del negocio, donde se pretende facilitar la comprensión del funcionamiento del mismo.

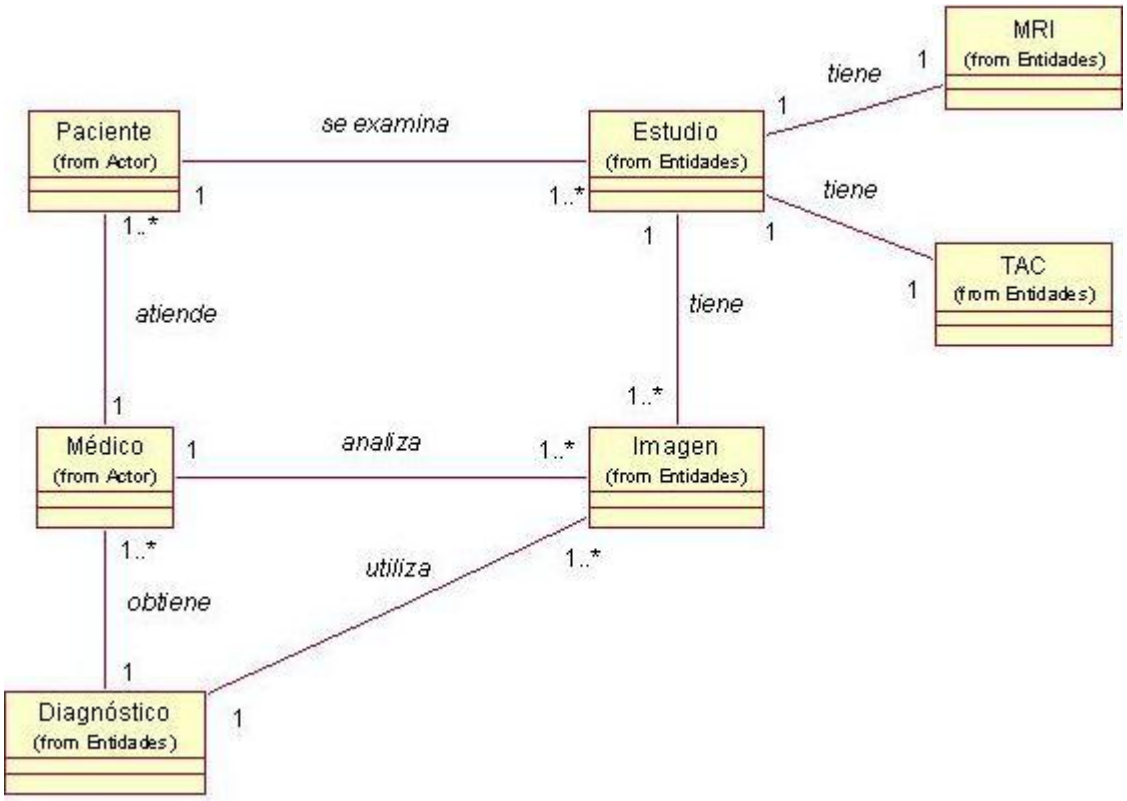

Figura 3.1 Modelo de Dominio.

3.2.1 Descripción del Modelo de Dominio.

Al paciente se le realizan una serie de estudios por parte de los médicos, lo cual puede ser a través de tomografías computarizadas (TAC) o resonancias magnéticas (MRI). Estos estudios adquiridos están formados por un conjunto de imágenes que pueden estar orientadas en el plano sagital, coronal o el axial. El médico especialista analiza las imágenes a través de una herramienta de visualización para obtener un diagnóstico de paciente.

3.3 Captura de Requisitos.

Un requerimiento es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, norma, especificación u otro documento formal, facilitando el

entendimiento entre clientes y desarrolladores. A continuación se exponen los requisitos funcionales y no funcionales, por los cuales se va a regir el desarrollo del módulo.

3.3.1 Requisitos Funcionales.

Los requisitos funcionales son capacidades o funciones que el sistema debe cumplir. A continuación se presentan los requerimientos del módulo a desarrollar.

R1. Cargar archivos.

El sistema debe permitir:

R1. Cargar

R1.1 Seleccionar los archivos a cargar.

R1.2 Cargar archivos DICOM (*.DCM).

R2. Visualizar Modelo 3D

R2.1 Generar texturas 3D.

R2.2 Realizar cortes al modelo tridimensional obtenido.

R2.3 Mostrar el modelo tridimensional obtenido.

R2.4 Rotar el modelo tridimensional obtenido.

3.3.2 Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener y representan las características que hacen al producto atractivo, usable, rápido o confiable. Seguidamente se enumeran los requisitos no funcionales del módulo a desarrollar.

1. ***De Software:***

   Sistema Operativo Linux o Windows XP.

2. ***De Hardware:***

   Tarjeta gráfica de 512 MB con soporte para textura 3D.

   Procesador Pentium 4 a 3.0 GHz o superior.

   Memoria RAM de 1GB de capacidad.

   Espacio en disco duro de 512 Mb.

3. ***De Seguridad:***

   Confidencialidad: Los modelos 3D obtenidos en la visualización deben representar lo más real posible la anatomía humana.

   Integridad: No se debe perder información del estudio inicial a la hora de generar la textura 3D y aplicarla para visualizar.

4. ***De Usabilidad:***

   Los usuarios deben tener un conocimiento básico sobre la visualización tridimensional aplicada a la medicina.

5. ***De Apariencia o interfaz externa:***

   La interfaz propuesta debe ser lo más atractiva y fácil de usar para el usuario final, implicando que todas las funcionalidades faciliten la interactividad en tiempo real.

6. ***De Soporte:***

   Soporte para los sistemas operativos Windows XP y versiones de Linux.

7. ***De Restricciones en el diseño e implementación:***

   Se utilizará el lenguaje de programación C++ utilizando el paradigma orientado a objetos.

3.4 Modelo de Casos de Uso del Sistema.

En esta sección se reconocen los actores del sistema, así como los casos de uso del módulo a desarrollar.

3.4.1 Actores del Sistema.

Los actores del sistema son agentes externos, que pueden cambiar información con él. Pueden representar el rol que juega una o varias personas, un equipo o sistema automatizado. En el caso particular quien interactuará con el módulo es un especialista médico, por lo que como actor del sistema se le llamará médico. En la tabla 3.1 se justifica la selección de este actor.

Actores	**Justificación**
Médico	El médico es quien interactúa con el sistema para ejecutar las funcionalidades de: cargar imágenes DICOM perteneciente a un estudio realizado a un paciente, visualizar el modelo 3D obtenido, rotarlo y realizarle cortes para navegar por dentro del mismo.

Tabla 3.1 Actor del Sistema.

3.4.2 Diagrama de Casos de Uso del Sistema.

El diagrama de casos de uso del sistema representa gráficamente los casos de uso y su interacción con los actores. En la **Figura 3.2** se muestra el diagrama de casos de uso del sistema para el módulo en cuestión.

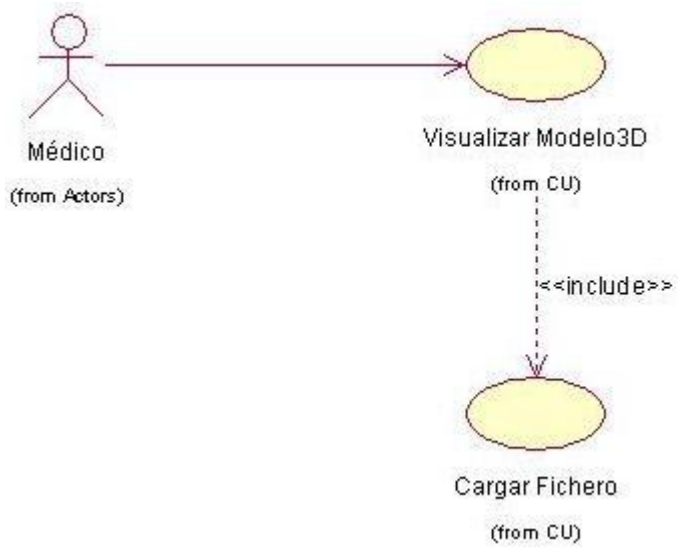

Figura 3.2 Diagrama de Casos de Uso del Sistema.

3.4.3 Descripción de casos de Uso del Sistema.

Cada caso de uso tiene una descripción de las funcionalidades que ejecutará el sistema propuesto como respuesta a las acciones del usuario. Las tablas presentadas a continuación argumentan los flujos operacionales del caso de uso Visualizar Modelo 3D. La descripción del caso de uso Cargar Fichero, así como los diagramas de secuencia y clases correspondientes al mismo se encuentran en la Trabajo de Diploma Reconstrucción Tridimensional de Modelos Anatómicos a partir de Imágenes Médicas Digitales [22]. Con el caso de uso Cargar Fichero se le da solución a los requisitos funcionales R1.1 y R1.2.

Caso de Uso:	Visualizar Modelo 3D
Actor:	Médico
Propósito:	Generar un modelo 3D y visualizarlo en pantalla.
Resumen:	El caso de uso se inicia una vez que se haya seleccionado el estudio a cargar, donde se interactúa con una interfaz, la cual es la encargada de cargar las imágenes y proveer

	el grid 3D, el cual es utilizado para generar el modelo 3D. Luego se visualiza el mismo. El usuario puede navegar por el volumen mediante el uso de cortes.
Referencia:	R1.1, R1.2, R2.1, R2.2, R2.3, R2.4, CU Cargar Fichero
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos

Acción del Actor	Respuesta del Sistema
1. El médico selecciona la opción Cargar imágenes.	1.1 Muestra un cuadro de diálogo donde el médico podrá seleccionar el directorio y las imágenes que desea cargar.
2. El médico selecciona las imágenes que desea cargar y oprime el botón aceptar.	2.1 Cierra el cuadro de diálogo y procede a cargar las imágenes seleccionadas. 2.2 Interactúa con la Interfaz Cargar Fichero para cargar las imágenes y generar el grid 3D. 2.3 Genera la textura 3D a partir del grid 3D creado. 2.4 Aplica la textura 3D a la geometría proxy. 2.5 Visualiza el modelo 3D con los valores de los cortes por defecto.

3. El médico puede cambiar los valores de los cortes o rotar el volumen.	3.1. El sistema ejecuta una de las acciones siguientes: • Si el médico desea cambiar los valores de los cortes consultar sección Realizar Corte. • Si el médico desea rotar el volumen consultar sección Rotar Volumen.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El médico selecciona las imágenes que desea cargar y oprime el botón aceptar.	2.1 Si no han sido seleccionadas las imágenes a cargar, muestra un error.
Poscondiciones:	Se realizan los cortes o se rota el volumen.
Prioridad:	**Crítico**

Tabla 3.2 Descripción del CU Visualizar modelo 3D.

Sección:	Realizar Corte
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El médico selecciona la opción realizar corte.	1.1 Muestra un formulario con los valores por defecto de los cortes.
2. El médico puede o no variar los	2.1. Visualiza el modelo 3D generado con

parámetros de los cortes.	los valores de los cortes introducidos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones:	Se visualizará en pantalla el modelo 3D generado.
Prioridad:	**Crítico**

Tabla 3.3 Descripción de la sección Realizar Corte.

Sección:	Rotar Volumen
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El médico selecciona la opción rotar volumen.	1.2 Habilita la opción de rotar el volumen.
2. El médico utiliza el mouse para realizar la rotación en los ejes ***x***, ***y*** o ***z***.	2.1. Captura el nuevo ángulo de rotación introducido por el mouse. 2.2. Visualiza el modelo 3D generado con la nueva rotación proporcionada.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Poscondiciones:	Se visualizará en pantalla el modelo 3D generado.
Prioridad:	**Crítico**

Tabla 3.4 Descripción de la sección Rotar Volumen.

3.5 Diagrama de Clases de Análisis.

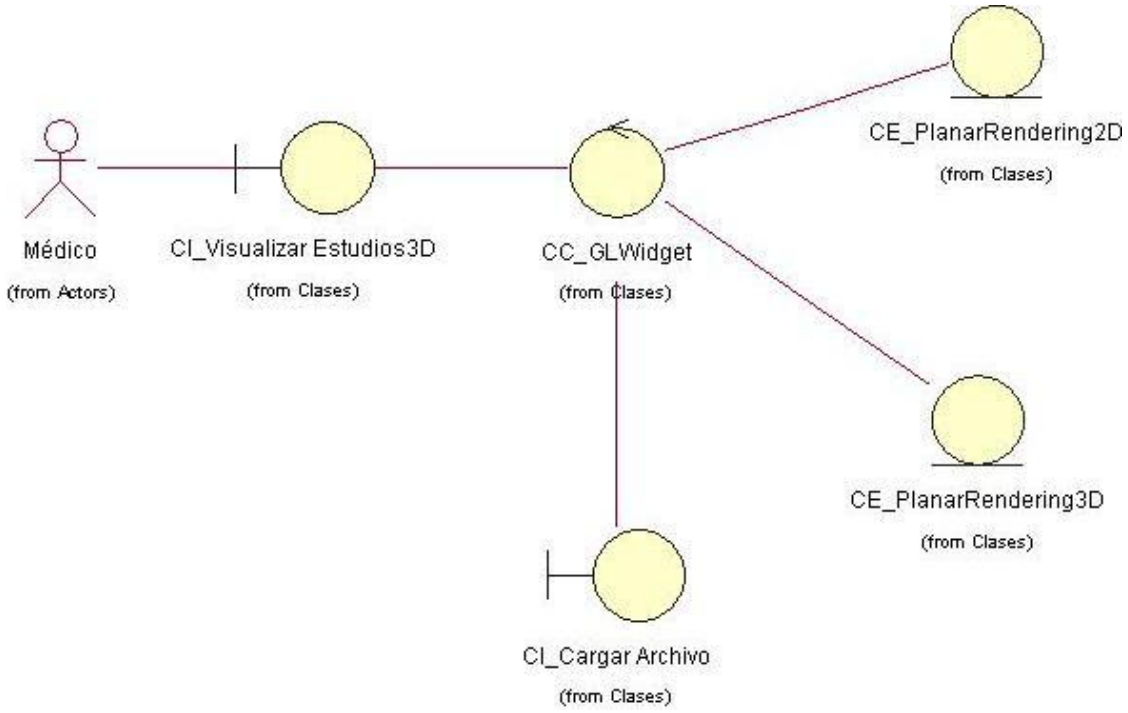

Figura 3.3 Diagrama de Clases del Análisis.

3.6 Diagramas de Colaboración.

3.6.1 Diagrama de Colaboración del Caso de Uso Visualizar Modelo 3D. Sección Realizar Corte.

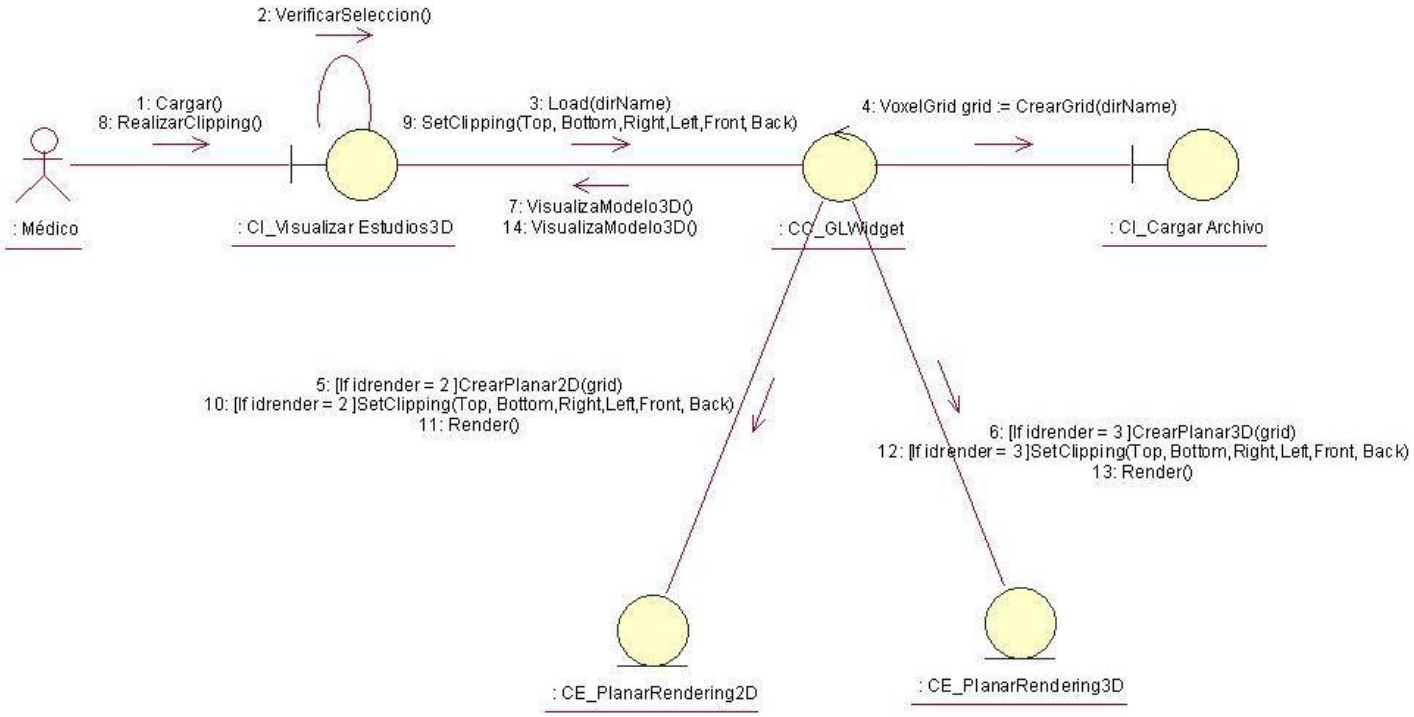

Figura 3.4 Diagrama de Colaboración del caso de uso Visualizar Modelo 3D. Sección Realizar Corte.

3.6.2 Diagrama de Colaboración del Caso de Uso Visualizar Modelo 3D. Sección Rotar Volumen.

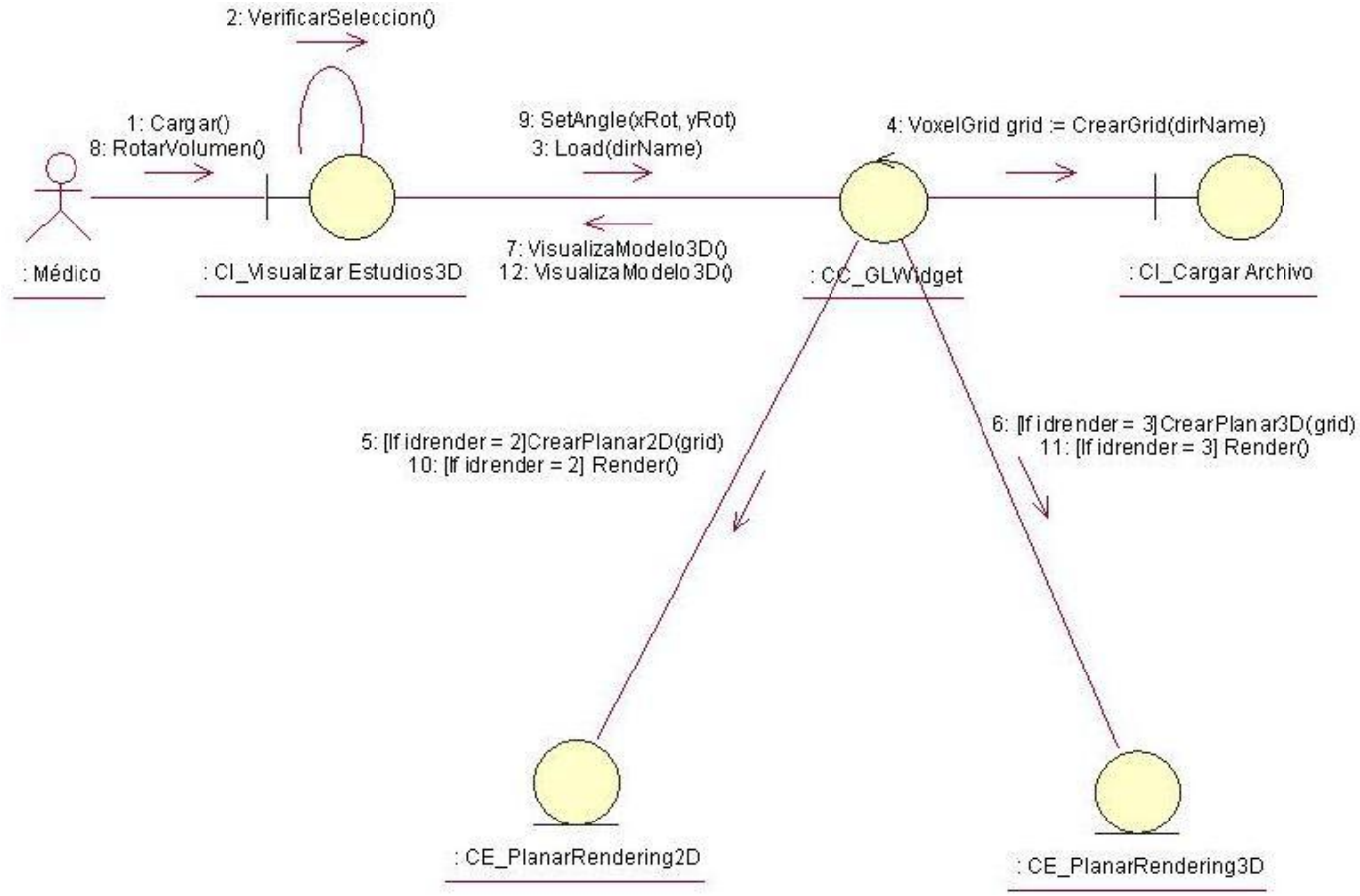

Figura 3.5 Diagrama de Colaboración del caso de uso Visualizar Modelo 3D. Sección Rotar Volumen.

3.7 Diagrama de Clases del Diseño del Sistema.

3.7.1 Diagrama de Clases del CU Visualizar Modelo 3D.

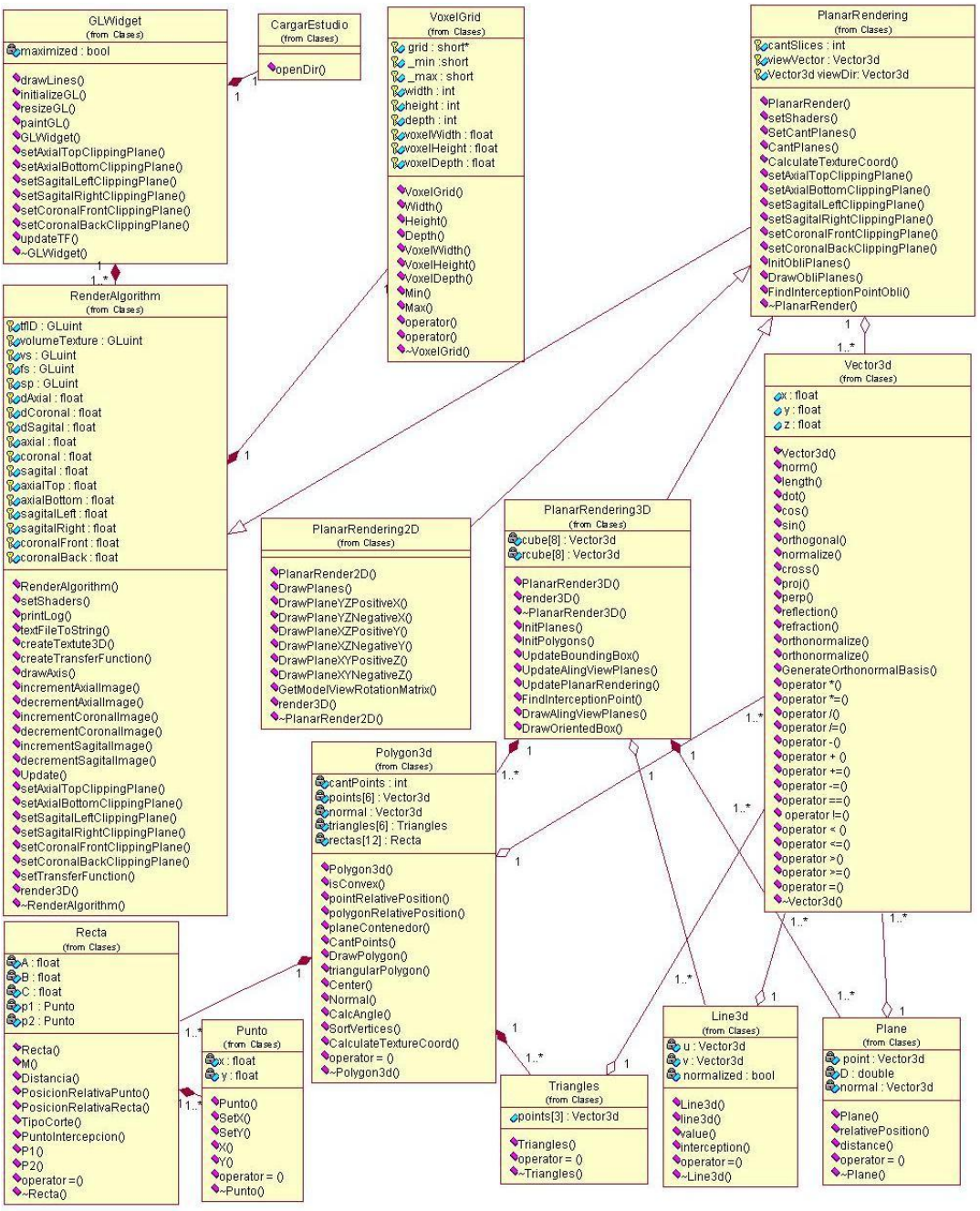

Figura 3.6 Diagrama de Clases del Diseño del Caso de Uso Visualizar Modelo 3D.

3.8 Diagramas de Secuencia del Diseño.

3.8.1 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección realizar Corte.

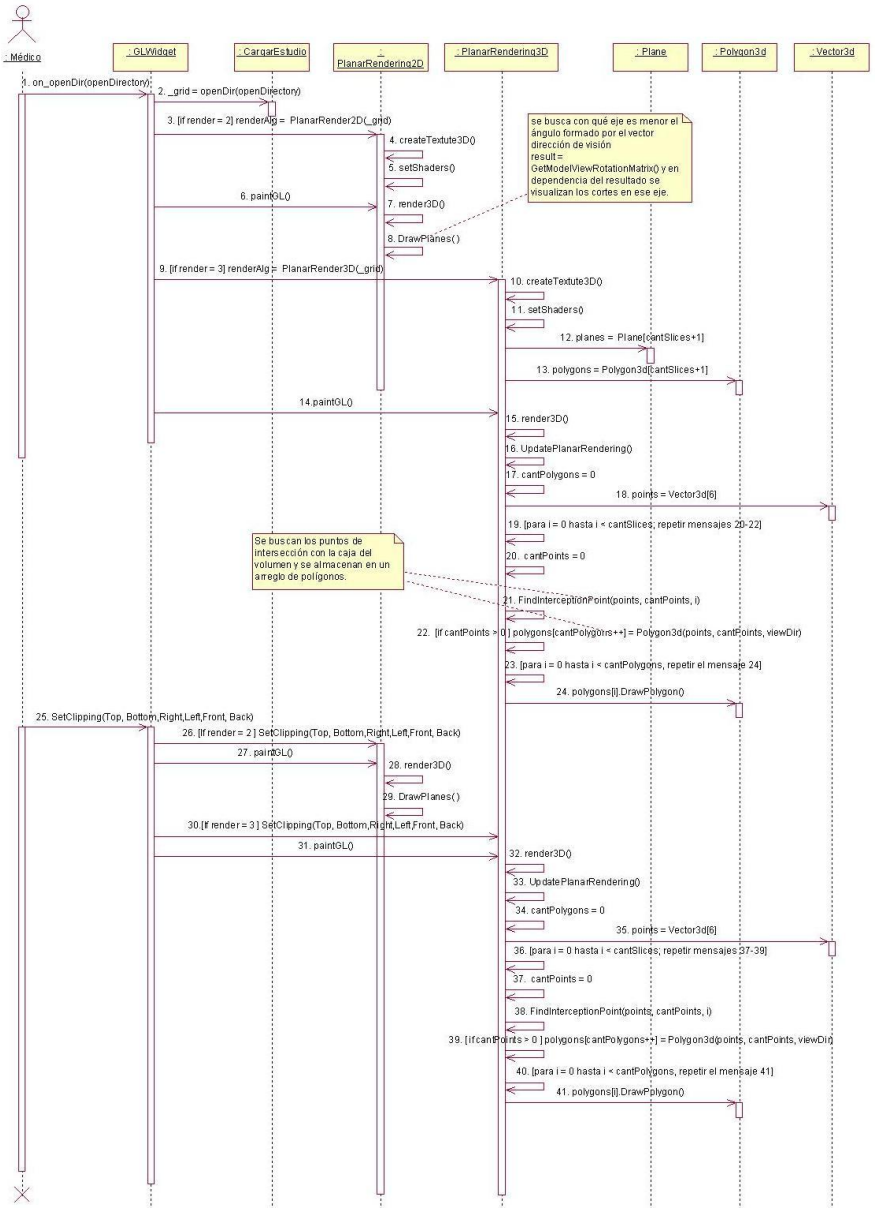

Figura 3.7 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección Realizar Corte.

3.8.2 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección Rotar Volumen.

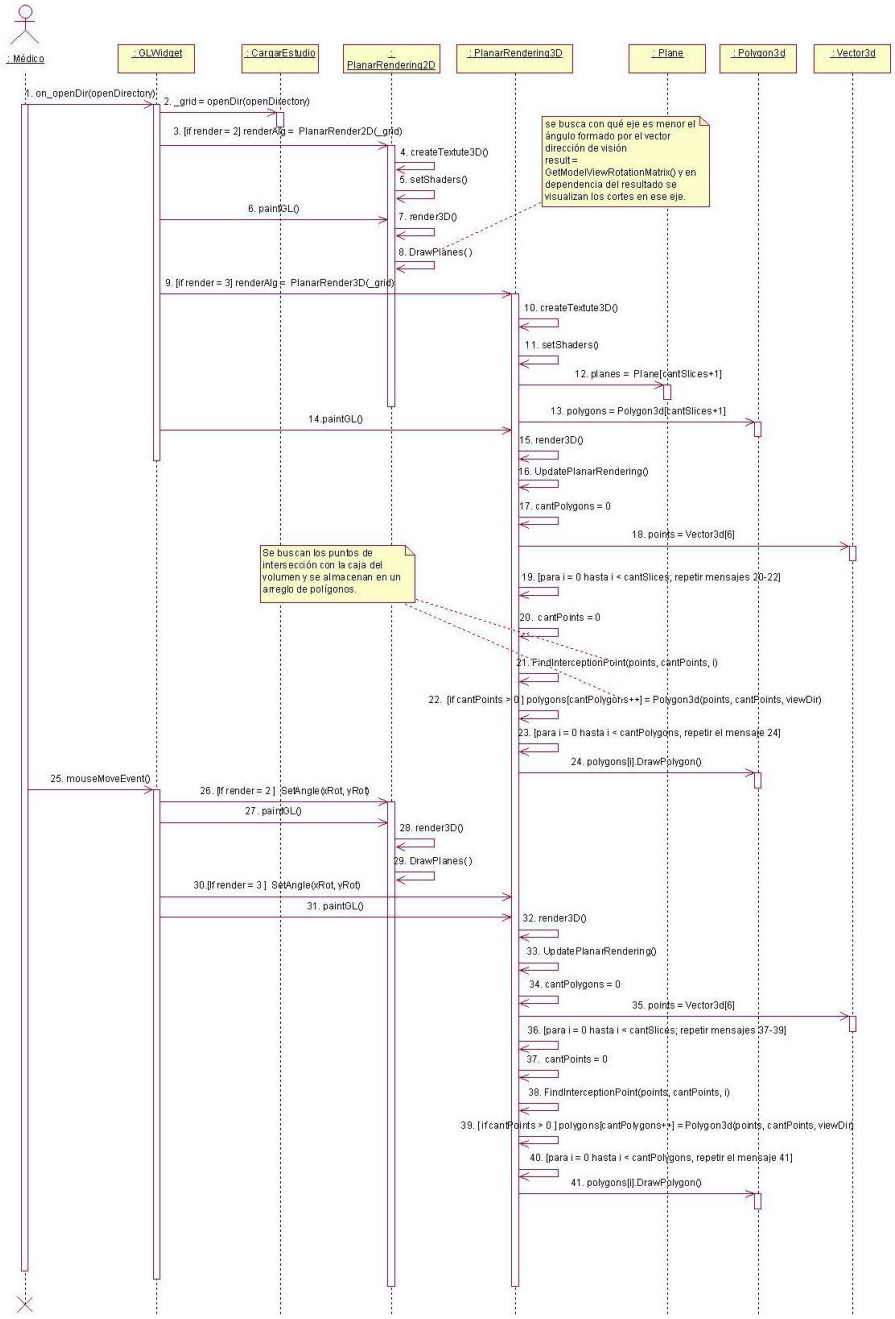

Figura 3.8 Diagrama de Secuencia del Diseño del Caso de Uso Visualizar Modelo 3D. Sección Rotar Volumen.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se analizan los temas de la implementación del módulo basado en lo descrito en los capítulos anteriores, se hará un análisis de los resultados del sistema, en cuanto al cumplimiento de los objetivos propuestos y al rendimiento de la aplicación.

4.1 Diagrama de Componentes.

Las clases resultantes del análisis y el diseño se hacen físicas mediante componentes. A continuación se muestran los componentes resultantes en la **Figura 4.2**, los cuales han sido agrupados en el paquete Visualización (ver **Figura 4.1**).

Figura 4.1 Paquete del diagrama de componentes.

4.1.1 Componentes del paquete de Implementación Visualización.

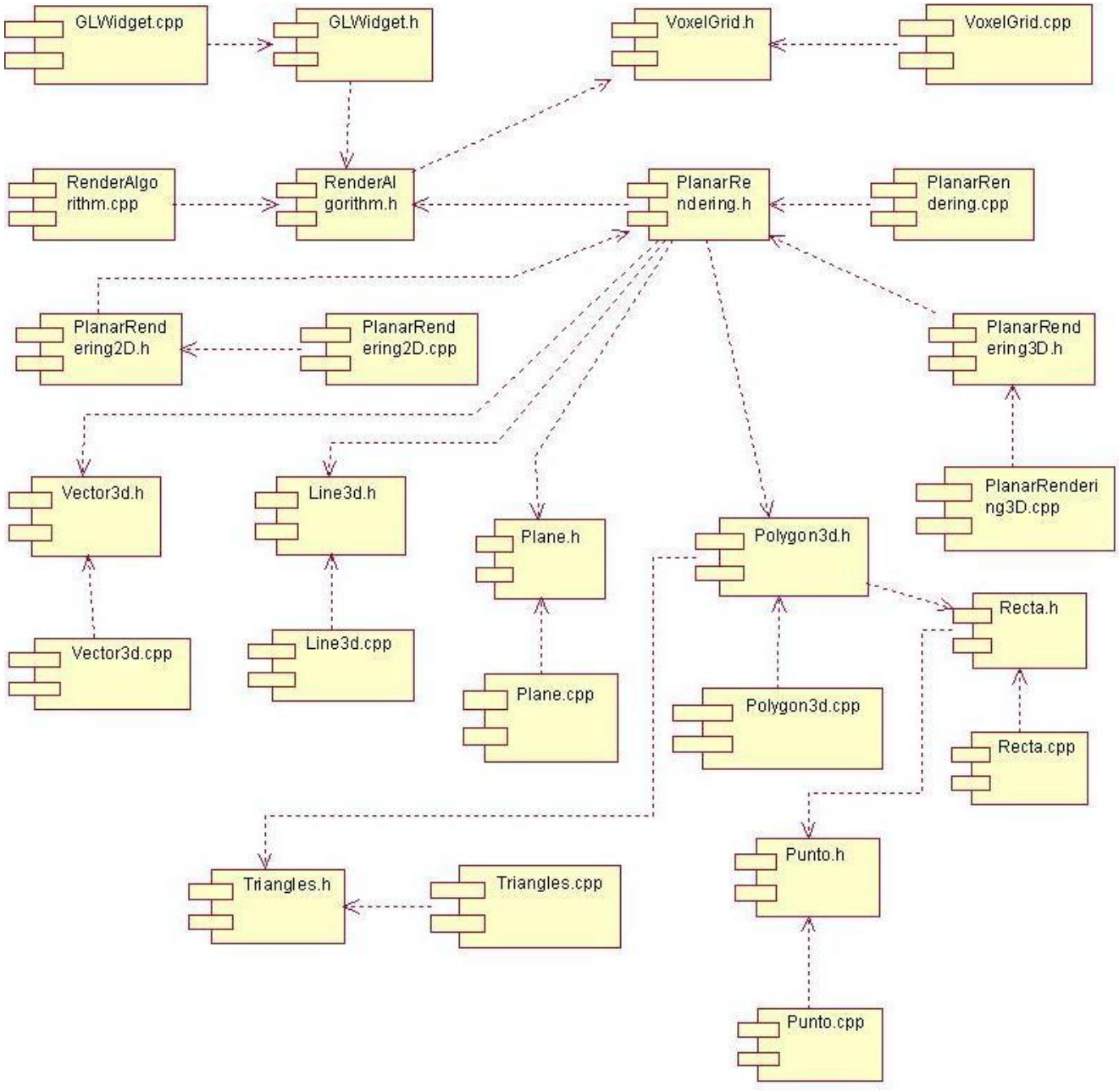

Figura 4.2 Diagrama de componentes. Paquete Visualización.

4.2 Valoración de los resultados.

4.2.1 Datos de entrada.

Para realización de las pruebas se tomaron como datos de entrada imágenes de estudios médicos aplicados a pacientes, los cuales se reflejan en la tabla 4.1. La misma está compuesta por tres campos, el primero refleja el ID del Caso de Estudio o prueba, el segundo corresponde a la modalidad en que fue realizado el estudio y el último consiste en las dimensiones del estudio, el cual está definido en un formato **X*Y*Z**, donde **X**, **Y** y **Z** corresponden al ancho, alto y cantidad de imágenes que componen el estudio respectivamente.

Nro. de Prueba	Modalidad	Dimensión del estudio
1	TAC	128x128x30
2	TAC	256x256x110
3	TAC	256x256x128
4	TAC	256x256x256
5	MRI	512x512x46
6	MRI	512x512x347

Tabla 4.1 Datos de Entrada.

4.2.2 Parámetros a medir.

Para evaluar los resultados de los dos algoritmos desarrollados se tendrán en cuenta tres criterios fundamentales para cada uno de los casos de prueba de la tabla 4.1: cantidad de cuadros por segundo, uso de memoria RAM y tiempo de carga y creación de la textura 3D.

Estos criterios fueron seleccionados teniendo en cuenta que la mayoría de los sistemas de visualización 3D en tiempo real, realizan sus pruebas sobre la base de estos parámetros, dado a que influyen directamente en los resultados esperados, sobre todo si el volumen de datos que se manipula es elevado, donde la calidad de la visualización y la interactividad en tiempo real deben ser una premisa de la solución del problema.

También se tuvieron en cuenta otros indicadores que resumen las características principales de forma cualitativa de los dos algoritmos tras las pruebas realizadas, donde se tuvieron en cuenta algunos parámetros tales como el rendimiento, referido a la cantidad de cuadros por segundo, la disponibilidad, que tiene en cuenta si no necesita algún hardware en específico, como es el caso de una GPU, los requisitos de memoria, que analiza el uso de memoria de cada algoritmo, el tipo de interpolación, que puede ser bilineal o trilineal, la frecuencia de muestreo, que analiza si se ven saltos o espacios entre los planos de cortes en la visualización del modelo y por último la calidad de la imagen obtenida. Todo esto quedó reflejado en la tabla 4.2.

Parámetros	**Objetos con cortes alineados texturizados con texturas 2D**	**Objetos con cortes alineados texturizados con texturas 3D**
Rendimiento	Alto.	Alto.
Disponibilidad	Alta.	Limitada. Necesita una GPU que soporte textura 3D.
Requisitos de memoria	Medio.	Medio.

Tipo de Interpolación	Bilineal.	Trilineal.
Frecuencia de muestreo	Consistente	Consistente para la proyección ortogonal e inconsistente para la proyección de perspectiva.
Calidad de la imagen	Alta.	Alta.

Tabla 4.2 Sumario de la técnica Planar Rendering en sus dos variantes de implementación.

En las pruebas realizadas, cuyos resultados cuantitativos se visualizan en la tabla 4.3, la cantidad de planos de corte se hizo coincidir con la cantidad de imágenes de cada uno de los caso de prueba, para aumentar el nivel de realismo de la simulación.

	Mapeo 2D			**Mapeo 3D**		
Nro. de Prueba	**Tiempo de carga y creación de la textura (s)**	**Cantidad de cuadros por segundo**	**Uso de memoria(Kb)**	**Tiempo de carga y creación de la textura (s)**	**Cantidad de cuadros por segundo**	**Uso de memoria(Kb)**
1	2.27	59.34	37232	1.44	58.6	35016
2	1.6	58.74	48268	1.56	56.41	48548
3	1.68	55.69	51148	1.66	53.50	50668
4	2.28	53.70	66828	3.45	52.3	66984
5	2.74	25.81	81388	3.98	12.9	80796

6	22.00	15.70	266432	12.23	9.74	391980

Tabla 4.3 Resultados de la cantidad de cuadros por segundo, consumo de recursos de hardware y tiempo de pre procesamiento para los datos de entrada.

Las tablas 4.2 y 4.3 evidencian que la solución propuesta, logra resultados de muy buena calidad en tiempo real, tanto para el mapeo 2D como para el mapeo 3D, en un tiempo relativamente corto y con requerimientos mínimos de hardware.

CONCLUSIONES

Con la realización de este trabajo se logró desarrollar un módulo que permite la visualización 3D de modelos anatómicos, a partir de imágenes médicas digitales, facilitando la interacción en tiempo real del usuario con los mismos; utilizándose para ello la técnica del Planar Rendering en sus dos variantes de implementación, seleccionada a partir de un profundo estudio bibliográfico realizado. Lográndose de esta forma el cumplimiento de los objetivos planteados, debido a que:

- Se hizo una descripción de las principales técnicas y algoritmos de visualización 3D que existen en el mundo, centrando la atención en las variantes de implementación de la técnica Planar Rendering, quedando claro las ventajas y desventajas de la misma.

- Se describió la estructura del grid 3D, el cual es utilizado como entrada de las variantes de implementación, o sea, del mapeo 2D y el mapeo 3D.

- Se analizó los requerimientos de hardware de ambas variantes de implementación, donde el mapeo 3D requiere del uso de una GPU, la cual provee el soporte para la textura 3D, mientas que el mapeo 2D no requiere del uso de este hardware, dado a que todo el procesamiento se puede hacer en el CPU, no obstante, como el uso del mismo puede potenciar la capacidad de procesamiento, se decidió utilizarlo.

- En el mapeo 2D para eliminar la ocurrencia de efectos no deseados, en el momento de cambio entre las pilas de cortes, se aumentó en tiempo de ejecución la cantidad de planos, lográndose mejores resultados visuales, los cuales se aproximan bastante a la calidad visual del mapeo 3D, el cual soporta interpolación trilineal, contrario al primero, que solamente soporta interpolación bilineal.

- Se establecieron planos de cortes adicionales, mediante los cuales se puede explorar el volumen arbitrariamente, de esta forma la anatomía humana puede ser analizada en más detalle y llegar a un diagnóstico más preciso.

- Para reducir la cantidad de memoria utilizada por el mapeo 2D, de tres pilas de cortes a una, se decidió sobrecargar el procesamiento, recalculando la geometría proxy en cada pasada de render, con lo cual se mejora la calidad de la visualización y se optimiza el uso de la RAM, todo si afectar la interactividad en tiempo real.

- Se realizaron pruebas al módulo desarrollado para garantizar que su desempeño cumpla con los objetivos trazados.

De esta forma quedó demostrado que el empleo de la técnica Planar Rendering como algoritmo de visualización y reconstrucción de modelos 3D a partir de estudios DICOM, produce una visualización de los mismos, realista e interactiva, en tiempo real.

RECOMENDACIONES

Luego de la investigación realizada y de los resultados obtenidos en la misma surgen algunas recomendaciones para investigaciones futuras, entre ellas se pueden mencionar:

- Investigar cómo, con la utilización de planos oblicuos como geometría proxy, se pueden lograr mejores resultados visuales, eliminando la cantidad de efectos no deseados producidos durante la visualización.

- Investigar cómo se puede compactar el tamaño de las texturas cuando sobrepasan la capacidad de la memoria de la GPU.

BIBLIOGRAFÍA

1. **Martínez López, Manuel.** Resonancia magnética. [Online] Agosto 2001. [Cited: 01 16, 2010.] http://www.ciberhabitat.gob.mx/hospital/rm/.

2. **Biblioteca Nacional de Medicina de EE.UU.** MedLinePlus. *Tomografía computarizada.* [Online] Octubre 29, 2008. [Cited: 01 16, 2010.] http://www.nlm.nih.gov/medlineplus/spanish/ency/article/003330.htm.

3. **Universidad de Virginia.** Tomografía por Emisión de Positrones. [Online] Diciembre 2007. [Cited: 01 16, 2010.] http://www.healthsystem.virginia.edu/UVAHealth/adult_radiology_sp/pet.cfm.

4. **Biblioteca Nacional de Medicina de EE.UU.** MedLinePlus. *Ultrasonido.* [Online] Marzo 2009. [Cited: 01 16, 2010.] http://www.nlm.nih.gov/medlineplus/spanish/ency/article/003336.htm.

5. **William E. Lorensen, Harvey E. Cline.** *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* Nueva York, USA : General Electric Company, Corporate Research and Development, 1987, Vol.

6. **Bourke, Paul.** Polygonising a scalar field . [Online] 1994. http://local.wasp.uwa.edu.au/~pbourke/geometry/polygonise/.

7. **Hamann, Gregory M. Nielson y Bernd.** The asymptotic decider: resolving the ambiguity in marching cubes. Arizona, USA : Arizona State University, 1991. Vol. 9.

8. **Natarajan, B. K.** Generating Topologically Correct Isosurfaces from Uniform Samples. University of Michigan, Michigan, USA : s.n., 1991. Vol. 20.

9. **Chernyaev, Evgeni V.** Marching Cubes 33: Construction of Topologically Correct Isosurface. Institute for High Energy physics. Russia : s.n., 1995. Vol. 8.

10. **Max, Nelson.** Optical Models for Direct Volume Rendering. s.l. : IEEE Transactions on Visualization and Computer Graphics, 1995. Vol. 1, 99-108.

11. **Klaus Engel, Markus Hadwiger, Joe M. Kniss y Christof Rezk-Salama.** Real-Time Volume Graphics. 2006.

12. **Brecheinsen, Ralph.** Real-time volume rendering with hardware-accelerated raycasting. 2006.

13. **Chen Shihao, He Guiqing y Hao Chongyang.** Rapid Texture-based Volume Rendering. 2009.

14. **T. Cullip, and Neumann, U.** Accelerating volume reconstruction with 3D texture hardware. University of North Carolina, Chapel Hill N.C : s.n., 1993.

15. **B. Cabral, N. Cam et. al.** Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. s.l. : Proceedings ACM Symposium on Volume Visualization, 1994. Vol. 94, 91–98.

16. **O. Wilson, A. Van Gelder, y J. Wilhelms.** Direct volume rendering via 3D textures. University of California, Santa Cruz : s.n., 1994.

17. **Kim, A. Van Gelder y K.** Direct volume rendering with shading via three-dimensional textures. s.l. : Volume Visualization Symposium, IEEE, 1996. 23–30.

18. **Milan Ikits, Joe Kniss, Aaron Lefohn y Charles Hansen.** GPU Gems. Programming Techniques, Tips and Tricks for Real-Time Graphics. 2007.

19. **Bernhard Preim, Dirk Bartz.** Visualization in medicine. Burlington, Elsevier : s.n., 2007. 13-14.

20. **Lefohn, A., J. Kniss, C. Hansen y R. Whitaker.** A Streaming Narrow-Band Algorithm: Interactive Deformation and Visualization of Level Sets. s.l. : IEEE Transactions on Visualization and Computer Graphics, 2004.

21. **Ertl, Klaus Engel y Thomas.** Interactive High-Quality Volume Rendering with Flexible Consumer Graphics Hardware. Visualization and Interactive Systems Group, University of Stuttgart : s.n., 2002.

22. **Pereira Barzaga, Osvaldo y Kindelan Nuñez, Rolando.** Reconstrucción Tridimensional de Modelos Anatómicos a partir de Imágenes Médicas Digitales. 2008. 57, 66, 71.

GLOSARIO

A

Algoritmo: Es una lista que, dado un estado inicial y una entrada, propone pasos sucesivos para arribar a un estado final obteniendo una solución.

C

Cantidad de cuadros por segundo: Es la cantidad de veces que se dibujan los planos en un segundo.

Clipping plane: Técnica utilizada, aplicando planos de recortes, para explorar el interior de los modelos 3D generados en la visualización.

CPU: Unidad de procesamiento central.

D

Diagnóstico: Etimológicamente el concepto diagnóstico proviene del griego, tiene dos raíces, dia- que es a través de, por. Y gignoskein que es conocer, así etimológicamente diagnostico significa conocer a través de. El concepto de este significado es la identificación de la naturaleza o esencia de una situación o problema y de la causa posible o probable del mismo, es el análisis de la naturaleza de algo.

Digital: Quiere decir que utiliza o que contiene información convertida al código binario, el lenguaje de números (ceros y unos) que emplean los ordenadores para almacenar y manipular los datos.

E

Efectos no deseados: Alteración en la visualización, que se produce en una zona determinada de la imagen resultante, como resultado de inconsistencias en la interpolación.

F

Fotón: Cada una de las partículas que constituyen la luz y, en general, la radiación electromagnética en aquellos fenómenos en que se manifiesta su naturaleza corpuscular.

Fragmento: Es usado para describir los datos de los elementos antes de su salida como píxeles hacia la pantalla.

Fragment shader: Programa que calcula el color de los píxeles individuales. Controla cómo las texturas son aplicadas a los fragmentos.

Frecuencia de muestreo: Es la distancia promedio que existe entre los planos que se utilizan para hacer el muestreo, que varía o no en función de la calidad que se quiere obtener.

Frustum: Es un volumen de espacio 3D, definido como la parte de una pirámide rectangular que se encuentra entre dos planos perpendiculares a la línea de centro. Se utiliza a menudo para representar lo que una "cámara virtual" ve en el espacio 3D.

Función de transferencia: Es un modelo matemático que a través de un cociente relaciona la respuesta de un sistema a una señal de entrada o excitación.

G

GPU: Unidad de procesamiento gráfico.

Geometría proxy: Es el conjunto de primitivas de hardware que representan a un objeto volumétrico.

Gradiente: Denota una dirección en el espacio según la cual se aprecia una variación de una determinada propiedad o magnitud física.

Grid 3D: Estructura para almacenar la información de las imágenes médicas contenidas en los estudios.

I

Imagen: Figura, representación, semejanza y apariencia de algo.

Interactividad: Cualidad de interactivo. Que permite una interacción, a modo de diálogo, entre el ordenador y el usuario.

Interpolación: Algoritmo matemático que a partir de varios puntos en el espacio, describe una función que contiene a los puntos intermedios.

Invasivo: Que invade o tiene capacidad para invadir. Irrumpir, entrar por la fuerza. Ocupar anormal o irregularmente un lugar.

M

Mapeado de texturas: Proceso por el cual se asignan los téxeles a cada píxel de la imagen final.

Matriz de modelado: controla la posición de la cámara respecto a las primitivas que se utilizan en el render.

Metodología: Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.

Modelo 3D: Modelo tridimensional, real o virtual para representar un cuerpo o una parte del mismo.

Módulo: Pieza o conjunto unitario de piezas que se repiten en una construcción de cualquier tipo, para hacerla más fácil, regular y económica.

O

Optimización: Acción y efecto de optimizar. Buscar la mejor manera de realizar una actividad.

P

Paralelismo: Es una forma de computación en la cual varios cálculos pueden realizarse simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños, que son posteriormente solucionados en paralelo.

Pase de render: Es el proceso en el cual los datos pasan completamente por las diferentes etapas del pipeline.

Patología: Conjunto de síntomas de una enfermedad.

Pipeline: División de un proceso en etapas, asignando a cada una de ellas distintos recursos.

Píxel: Abreviatura de “picture element”. Es la mínima unidad de información dentro de una imagen bidimensional.

Primitiva geométrica: Formas geométricas consideradas primitivas por su básica constitución en las partes que la conforman, que pueden ser combinadas para crear formas geométricas más complejas.

Polígono: Porción de plano limitada por líneas rectas.

R

RAM: Es la memoria de acceso aleatorio (en inglés: *random-access memory*), desde donde el procesador recibe las instrucciones y guarda los resultados. Es el área de trabajo para la mayor parte del software de un computador.

Reconstrucción 3D: Es el proceso mediante el cual objetos reales son reproducidos en la memoria de un computador, manteniendo sus características físicas (dimensiones, volumen y forma).

Render: el proceso de convertir modelos geométricos 3D en escenas 2D para presentarlas a los usuarios.

Resolución: Es el número de píxeles que se muestran en una pantalla. Al ser ésta una matriz de filas y columnas de píxeles, primero se nombra la cantidad de columnas (resolución horizontal) y luego la cantidad de filas (resolución vertical).

Resonancia magnética (RM): La RM es una técnica de diagnóstico que posee la capacidad de generar finas secciones de modo no invasivo y proporciona imágenes estáticas y dinámicas de alta calidad de las estructuras internas de todo el cuerpo, desde cualquier ángulo y dirección. En su funcionamiento utiliza magnetos superconductivos y ondas de radio-frecuencia para producir las imágenes, donde el análisis computarizado, de los cambios de energía de los tejidos cuando reaccionan a fuerzas magnéticas, crea una imagen compuesta de los tejidos. En la visualización 3D permite la visualización de las arterias y venas empleando la técnica denominada angiografía por resonancia magnética, la visualización del corazón con exquisito detalle anatómico empleando la técnica del trazado electrocardiográfico, el rastreo de componentes bioquímicos que corresponden a cualquier corte anatómico del cuerpo humano. Es la técnica de diagnóstico de elección esencial para todos los procesos del cerebro y del sistema nervioso central.

S

Segmentación: Se utiliza en el Procesamiento de Imágenes para el reconocimiento de objetos o estructuras de interés en la imagen.

Shader: conjunto de instrucciones capaces de ser ejecutadas por un procesador gráfico.

T

Tetraedro: Sólido determinado por cuatro planos o caras.

Texel: (contracción del inglés *texture element*, o también *texture pixel*) es la unidad mínima de una textura aplicada a una superficie, usada en gráficos por computador. De la misma forma que una imagen digital se representa mediante una matriz de píxeles, una textura se puede representar mediante un matriz de téxeles.

Textura: Son imágenes que pueden ser mapeadas en cualquiera de las primitivas gráficas para adicionar detalles a la escena.

Tomografía Computarizada (TC): La TC es también conocida como Tomografía Axial Computada (TAC), es un método de diagnóstico que muestra imágenes bidimensionales de las estructuras internas del cuerpo de una persona sin el empleo de procedimientos invasivos. Esta técnica utiliza rayos-x para obtener vistas de cortes cruzados verticales y horizontales de zonas internas del cuerpo. Para aplicaciones de visualización 3D, es usada frecuentemente para la visualización de la estructura de los huesos además de algunos tejidos blandos.

Tomografía de emisión de positrones (TEP): Constituye una técnica utilizada en la medicina nuclear en la obtención de imágenes de los tejidos corporales internos. Requiere un ciclotrón como fuente local de positrones emitidos por isótopos. Los isótopos se inyectan al paciente junto con un compuesto relacionado con la glucosa, y los positrones chocan con los positrones en los tejidos corporales para producir fotones. Los fotones son seguidos por un contador tomográfico de centelleo, y la información es procesada por una computadora que proporciona imágenes y datos sobre el flujo sanguíneo y los procesos metabólicos de los tejidos observados. En la visualización 3D es utilizada en la visualización de sangre en diferentes estructuras ya que puede mostrar la presencia de ésta con mayor nivel de detección que la TC, en el

diagnóstico de tumores cerebrales y del efecto de los accidentes cerebrovasculares, así como de varias enfermedades mentales. También se utilizan en investigación sobre el cerebro y en el estudio del mapa de las funciones cerebrales.

Topología: Es el método matemático – lógico usado para definir las relaciones espaciales entre los objetos espaciales. Hace referencia a las propiedades de vecindad o adyacencia, inclusión, conectividad y orden, es decir, propiedades no métricas.

U

Ultrasonidos: También conocido con ecografía, constituye una técnica de diagnóstico en la que un sonido de frecuencia muy alta es dirigido hacia el organismo, donde las interfaces tisulares reflejan el sonido, y el patrón de reflexión del sonido resultante es digitalizado para producir una imagen móvil en una pantalla o una fotografía. Los ultrasonidos se utilizan para explorar el sistema arterial, el corazón, el páncreas, la cavidad peritoneal, el tracto urinario, los ovarios, el sistema venoso y la médula espinal. Su aplicación más conocida es la exploración del feto durante el embarazo. Cuando se utilizan para explorar el corazón, se denomina ecocardiografía. La ecocardiografía se emplea en el estudio de cardiopatías congénitas, enfermedades de las arterias coronarias, tumores del corazón y, de forma especial, para las alteraciones de las válvulas cardiacas. Los ultrasonidos son también útiles para guiar intervenciones quirúrgicas, por ejemplo durante la amniocentesis o para introducir una aguja de biopsia en una región determinada.

Unidimensional: Término utilizado para describir figuras que sólo se pueden medir en una dirección, o sea, que poseen una sola dimensión, como una línea, que sólo tiene longitud.

V

Vertex shader: Es una función de procesado gráfico que manipula los valores de un vértice en un plano 3D mediante operaciones matemáticas sobre un objeto. Estas variaciones pueden ser diferencias en el color, en las coordenadas de la textura, en la orientación en el espacio o en el tamaño del punto. Permite el control sobre las transformaciones sobre los vértices.

Vértices: son puntos en el espacio 3D que definen primitivas gráficas tales como triángulos, polígonos y rectángulos, usados para construir la geometría de la escena que será dibujada.

Viewport: Rectángulo definido en el sistema de referencia de la pantalla, cuyo objetivo es seleccionar que área del mundo se desea ver en un sub-área de la pantalla.

Voxel: (del inglés: *volumetric pixel*) es la unidad cúbica que compone un objeto tridimensional. Constituye la unidad mínima procesable de una matriz tridimensional y es, por tanto, el equivalente del píxel en un objeto 2D.

ANEXOS

Anexo A: Imágenes de la Aplicación.

A continuación se muestran varias imágenes del módulo desarrollado.

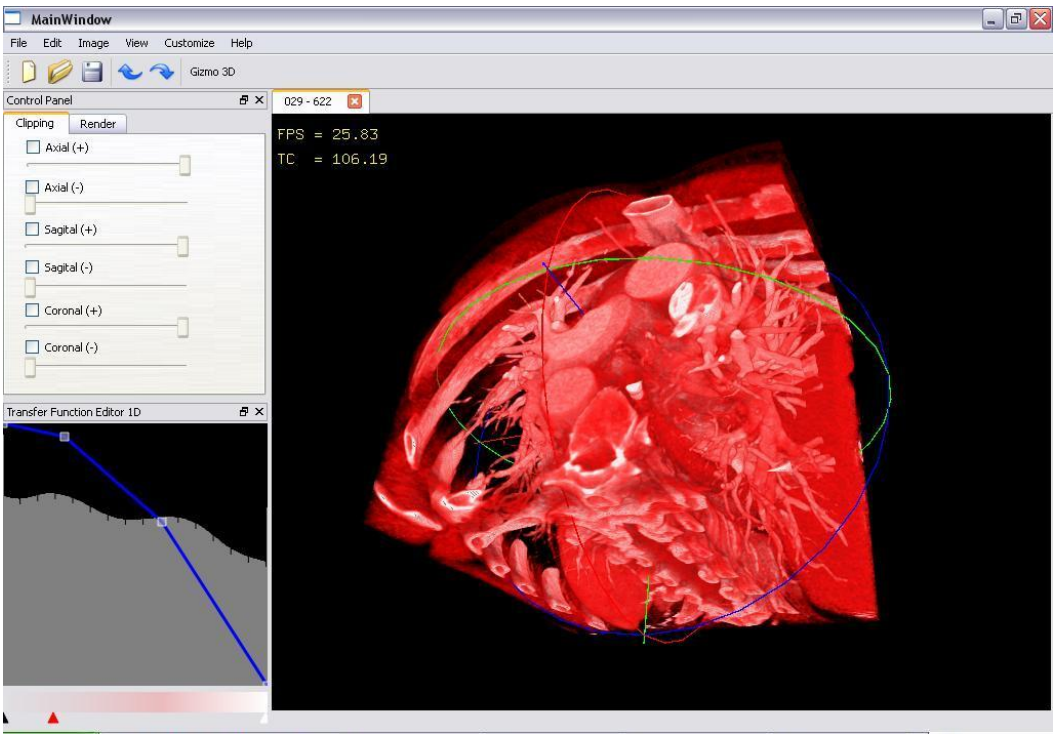

Figura A.1 Visualización 3D de un modelo de Tórax utilizando el mapeo 2D. Se ha utilizado la función de transferencia para lograr un acercamiento a la realidad de la anatomía humana.

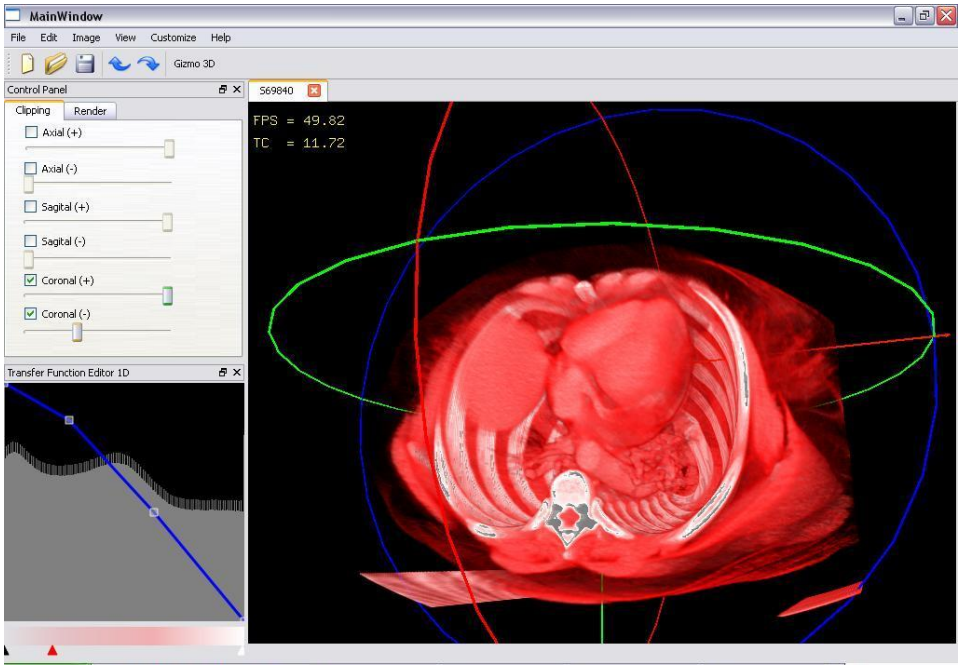

Figura A.2 Visualización 3D de un modelo de Tórax utilizando el mapeo 2D.

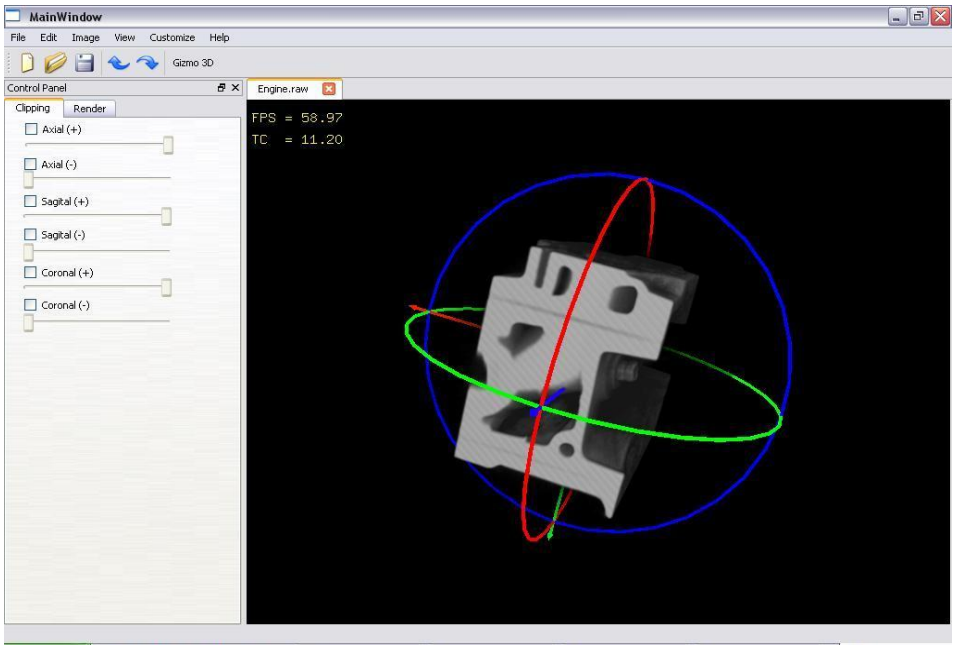

Figura A.3 Visualización 3D de un modelo de un motor de automóvil utilizando el mapeo 3D. La visualización se ha llevado a cabo utilizando la escala de grises.

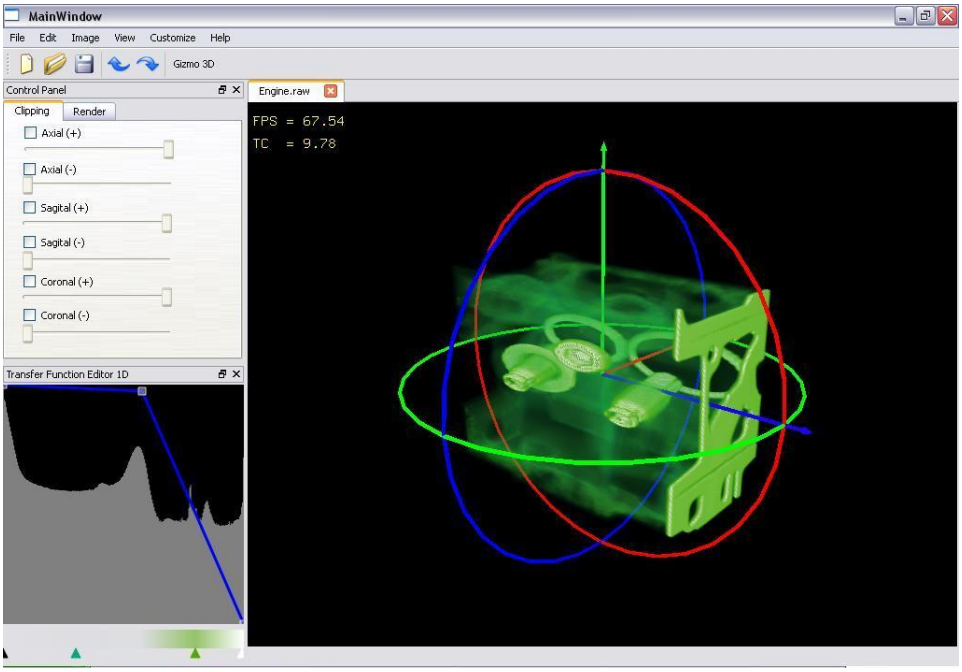

Figura A.4 Visualización 3D de un modelo de un motor de automóvil utilizando el mapeo 3D. Obsérvese cómo se logra la transparencia en el modelo.

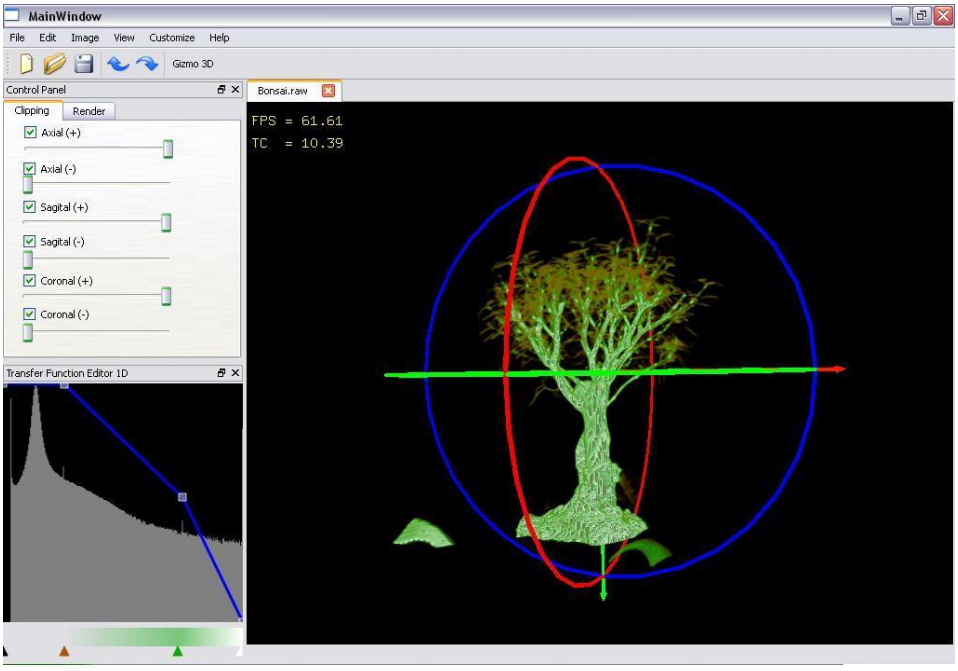

Figura A.5 Visualización 3D de un modelo de un bonsai utilizando el mapeo 3D.

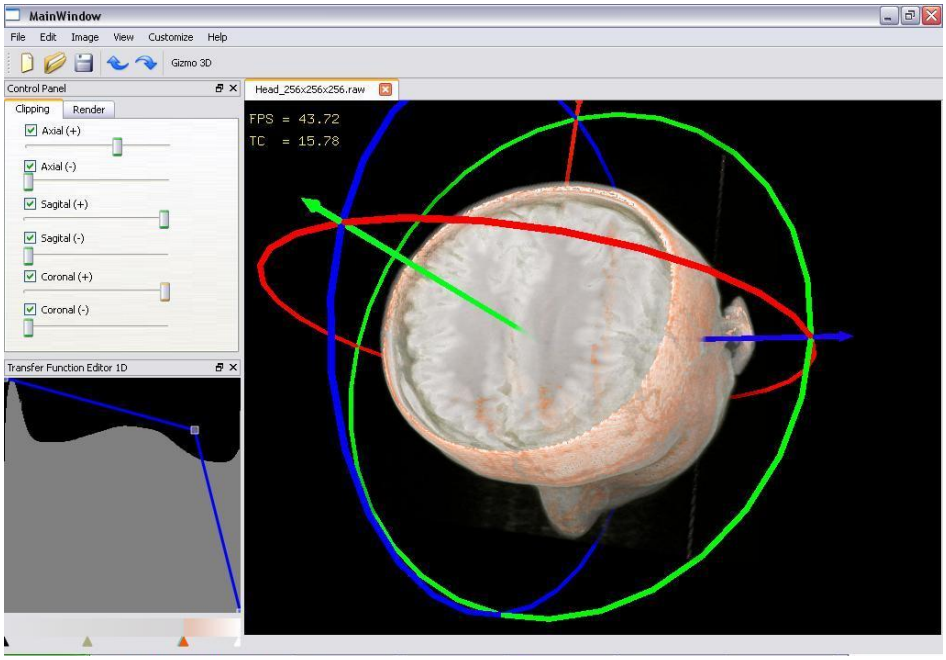

Figura A.6 Utilización de los planos de recorte en un eje, para explorar el modelo 3D.

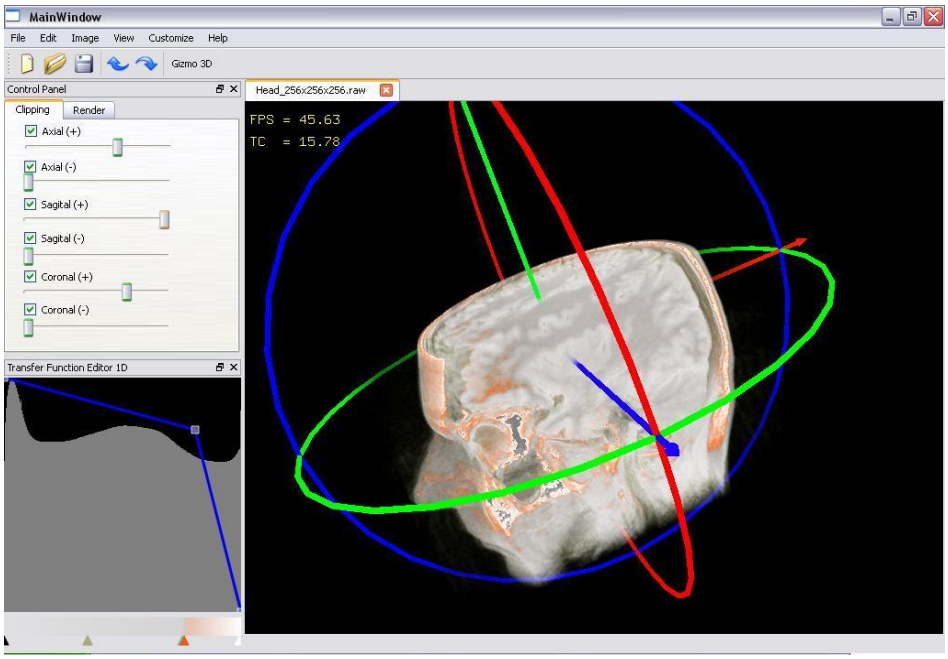

Figura A.7 Utilización de los planos de recorte en dos ejes, para explorar el modelo 3D.