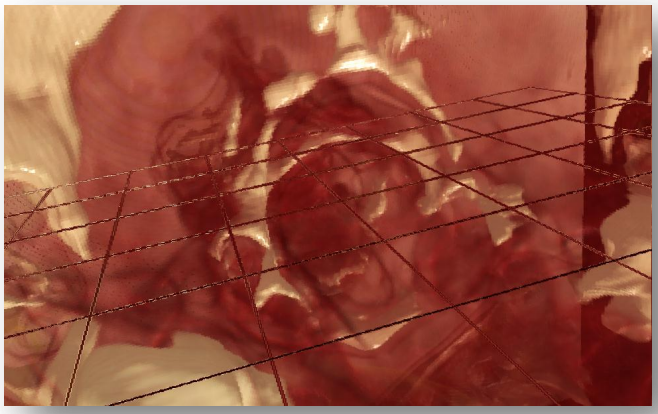
Universidad de las Ciencias Informáticas
Facultad 5

VISUALIZACIÓN DIRECTA DE VOLUMEN PARA ENDOSCOPIAS VIRTUALES



Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Luis Guillermo Silva Rojas

Tutor: M.Sc. Osvaldo Pereira Barzaga
Co-Tutor: Ing. Ernesto Carrasco De la Torre

Marzo 2011

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 _____.

Luis Guillermo Silva Rojas
Autor

M.Sc. Osvaldo Pereira Barzaga
Tutor

Ing. Ernesto Carrasco De la Torre
Co-tutor

DATOS DE CONTACTO

Tutor: M.Sc. Osvaldo Pereira Barzaga.
Edad: 26.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: M.Sc. en Informática Aplicada.
Categoría Docente: Instructor.
E-mail: opereira@uci.cu
Graduado de la UCI, con cuatro años de experiencia en el tema de la Gráfica Computacional y líder de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

Co-tutor: Ing. Ernesto Carrasco De la Torre.
Edad: 25.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ing. en Ciencias Informáticas.
Categoría Docente: Instructor Recién Graduado.
E-mail: ecarrasco@uci.cu.
Graduado de la UCI, con dos años de experiencia en el tema de la Gráfica Computacional y profesor de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

DEDICATORIA

A mis abuelos

A las personas que me observaron dar los primeros pasos, pronunciar las primeras palabras y levantarme de las primeras caídas. Reconocieron mis triunfos, lloraron conmigo en los fracasos y me cuidaron en todo momento. A los que cambiarían todo por ofrecerme un minuto de felicidad.

A mi madre

A la persona que me trajo al mundo, que ha sacrificado tantas cosas para verme crecer, que me enseñó a ver la realidad con mis propios ojos. A quien tiene la capacidad de localizarme en cualquier parte del mundo para saber si estoy bien. A quien siente mis logros como si fueran suyos.

A mis hermanos

A mis hermanos Alejandra y Enmanuel, aunque esté lejos, siempre pienso en ustedes, algún día podré dedicarles el tiempo que se merecen, compartir sus logros e indicarles el camino a seguir.

AGRADECIMIENTOS

Quisiera agradecer a todas las personas que de una forma u otra han contribuido a mi formación personal y profesional, en muchos casos, las palabras no son suficientes para expresar las dimensiones de mi agradecimiento.

A mis madres

Todas las personas son afortunadas por venir al mundo del interior de una mujer, mi dicha es doble, debo agradecer por igual a dos personas especiales que comparten la misma categoría. Por un lado la experiencia, por el otro la fuerza necesaria en los momentos más difíciles, pero de ambas la confianza suficiente para permitirme tomar las decisiones más importantes y el afecto necesario para levantarme de cada uno de los tropiezos. Gracias mamá, gracias mamita.

A mi papá:

Mi mejor amigo, mi primer maestro y mi ejemplo personal, la persona incansable que siempre quisiera ser. Una sola frase define toda nuestra relación y me enorgullece mucho escucharla de ti cuando dices que soy “el hijo varón que nunca tuviste”. Gracias papá.

A mi hermanita

Debe ser difícil tener un hermano que nunca está cuando más hace falta y ese ha sido uno de los mayores sinsabores que ha tenido la distancia de mis estudios. Sólo espero haber transitado por el camino correcto y que el tiempo me permita en algún momento reparar mi ausencia.

A mi novia Neysi

Por estar a mi lado durante casi toda la universidad, aún en los momentos más difíciles. Por entregarme tu amor, cariño y dedicación en todos los momentos posibles. Por ser mi alegría cuando debía estar triste, mi luz cuando todo se oscurecía y mi mejor enfermera.

A mi tutor Osvaldo

Por enseñarme el camino correcto durante la realización del presente trabajo, por ser ejemplo de trabajador incansable y mostrarme que no existen metas imposibles. Sólo conozco una persona que tiene tu fuerza de voluntad, mi papá.

A mi Co-tutor Ernesto

Por la constante preocupación acerca del estado de la presente investigación, por las sugerencias en los momentos precisos y el deseo de ayudar en todo lo necesario.

A mis compañeros

Por formar una gran familia en el aula y en el proyecto, por señalar las deficiencias y reconocer los resultados. Agradecimiento especial durante el proceso de investigación para Leonel y Rubén.

RESUMEN

El vertiginoso desarrollo de los dispositivos de exploración radiológica para la obtención de imágenes médicas digitales, tales como Tomógrafos Computarizados y Resonadores Magnéticos, magnificó la importancia de la información en forma de imágenes dentro de la medicina. Esto, unido al amplio avance del hardware gráfico y de los algoritmos para la visualización tridimensional de imágenes médicas, abre las puertas para la ejecución de procedimientos médicos no invasivos y de alta aceptación por los pacientes.

Las endoscopias virtuales permiten la exploración detallada de estructuras anatómicas, empleando imágenes médicas tridimensionales y gráficos generados por computadora. En el presente trabajo se propone el empleo del algoritmo Raycasting basado en GPU con algunas modificaciones que lo hacen extensible para la realización de endoscopias virtuales. Además se presentan varias técnicas para mejorar el rendimiento y la calidad de la imagen obtenida. Con el objetivo de visualizar escenas más complejas, se brinda la posibilidad de combinar la representación volumétrica con geometría poligonal.

El algoritmo se probó utilizando diferentes conjuntos de datos con el objetivo de mostrar su viabilidad. En todos los casos, la representación de la imagen tridimensional se realizó en tiempo real. Si bien, estas pruebas no son suficientes para utilizar los métodos en el ámbito clínico, los resultados muestran su potencial para alcanzar tal estado.

Palabras Clave: Endoscopia Virtual, GPU, Raycasting, Shader, Visualización de Volumen, Vóxel.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde el comienzo de la historia humana las imágenes han sido una herramienta de comunicación indispensable. Las primeras pinturas datan de más de 30 000 años, en estas, las personas de la época plasmaban interesantes bocetos relacionados con sus hábitos de caza. El sentido más desarrollado por los humanos es la vista, se estima que el 50% de las neuronas se encuentran dedicadas a la visión. Además, la densidad de información por unidad de área en una imagen es considerablemente mayor que la de un texto, por tanto, si los datos que se desean analizar son complejos o se presentan en grandes cantidades, es lógico pensar que se empleen imágenes.

La información en forma de imágenes se ha utilizado en prácticamente todas las ramas de la ciencia. La visualización médica es un campo especial dentro de la visualización científica que se establece como área de investigación a finales de la década del 80 del pasado siglo [1]. La visualización científica presenta tres raíces fundamentales: la primera de ellas consiste en la larga tradición de los científicos de ilustrar sus trabajos cuidadosamente, un ejemplo de esto son las detalladas ilustraciones anatómicas realizadas por el genio Leonardo Da Vinci (ver Fig. 1). Una segunda influencia proviene del procesamiento de imágenes, empleado fundamentalmente en el Análisis de Imágenes Médicas (MIA del inglés Medical Image Analysis), originalmente sólo se procesaban imágenes bidimensionales (2D), la representación de imágenes tridimensionales (3D) es tradicionalmente atribuida a la visualización médica. En tercer orden se encuentran los gráficos por computadora, estos proveen los algoritmos necesarios para realizar las representaciones.

Fig. 1 Ilustraciones anatómicas realizadas por Leonardo Da Vinci.

Los datos empleados en los algoritmos de visualización médica son adquiridos mediante dispositivos de exploración radiológica, tales como Tomografías Computarizadas (CT) e Imágenes por Resonancia Magnética (MRI). Estos dispositivos han experimentado una evolución impresionante en los últimos años. Aunque existen otras modalidades para la adquisición de estas imágenes como son los Ultrasonidos 3D, Tomografías por Emisión de Positrones (PET) y otras técnicas derivadas de la medicina nuclear; CT y MRI se destacan como las principales debido a la alta resolución de sus imágenes y a su buena tasa calidad-ruido.

Actualmente los equipos médicos poseen sistemas informáticos que permiten manipular imágenes médicas y visualizarlas de forma tridimensional (3D). Cada año un número superior de radiólogos sustituyen sus clásicas cajas iluminadas y las películas de rayos X por un software que les permita realizar tareas como ajustar el contraste y el brillo aunque el proceso de obtención de la imagen no haya sido el óptimo, mejorando convenientemente la calidad del diagnóstico.

Existe gran variedad a la hora de seleccionar un software de este tipo, desde algunos con características muy limitadas hasta otros con muchas funcionalidades pero extremadamente costosos. En el ámbito internacional se destacan los visualizadores de empresas como PHILIPS [2] y SIEMENS [3], además, existen herramientas para el desarrollo de aplicaciones de visualización de volumen como Voreen (Volume Rendering Engine) [4] y se hace imprescindible resaltar los trabajos realizados por el grupo VRVis [5] de la Universidad Tecnológica de Viena.

En Cuba, uno de los principales aportes en esta rama lo constituye el proyecto IMAGIS, desarrollado en el Centro Nacional de Biofísica Médica de la Universidad de Oriente y desplegado en la mayoría de los hospitales del país. En la Universidad de las Ciencias Informáticas (UCI), específicamente en la Facultad #7 se desarrolla el sistema ALAS PACS, el mismo se encarga de la manipulación y control de las imágenes médicas empleadas en los Centros de Alta Tecnología (CAT) de la República Bolivariana de Venezuela. Ambos productos carecen de un algoritmo de visualización de volumen que permita explorar detalladamente las estructuras anatómicas presentes en las imágenes médicas digitales.

En la Facultad #5 se creó el proyecto de Visualización Científica, una de sus principales tareas es la creación del software de visualización médica Vismedic, el mismo, a partir de imágenes médicas volumétricas (DICOM) es capaz de visualizar estructuras anatómicas de forma tridimensional pero no permite la exploración detallada de las mismas.

El software Vismedic no cuenta con un algoritmo de visualización de volumen que obtenga una representación realista del interior de las estructuras anatómicas, lo que trae consigo dificultades a la hora de examinar con el detalle requerido el interior de las mismas. En muchos casos sólo es posible visualizar el exterior de los órganos, perdiéndose la percepción de la naturaleza volumétrica que poseen.

Teniendo en cuenta la situación problémica anterior, se plantea el siguiente **problema de investigación**: ¿Cómo explorar con alto nivel de detalle las estructuras anatómicas empleadas en entornos quirúrgicos virtuales? A partir del problema planteado, se toma como **objeto de investigación** las endoscopias en entornos quirúrgicos virtuales y dentro de esta amplia área se propone como **campo de acción** la representación de endoscopias virtuales a partir de imágenes médicas digitales.

Se define como **objetivo general** implementar un algoritmo que permita la realización de endoscopias virtuales a partir de imágenes médicas digitales. Este trabajo defiende la siguiente idea: el empleo de Endoscopias Virtuales permitirá un análisis más detallado y una mejor interpretación de las imágenes médicas tridimensionales usadas en el proyecto Vismedic.

Para dar cumplimiento al objetivo planteado es necesario realizar un grupo de **tareas investigativas** tales como:

1. Elaboración del marco teórico a partir del estado del arte actual referente al tema.
2. Selección de la técnica de Visualización de Volumen más eficaz para el cumplimiento del objetivo propuesto.
3. Caracterización de las ventajas que ofrece la implementación del algoritmo seleccionado sobre la GPU con el objetivo de mejorar su rendimiento.
4. Selección de la metodología, herramientas de software y lenguajes de programación para el desarrollo del algoritmo.
5. Implementación del algoritmo seleccionado.
6. Incorporación de técnicas para la optimización del rendimiento del sistema y el mejoramiento de la calidad de la imagen.
7. Validación de los resultados a través de pruebas de rendimiento.
8. Integración del algoritmo seleccionado con la aplicación principal del proyecto Vismedic.

Para la realización de la investigación y elaboración del presente trabajo se emplearon varios **métodos científicos de investigación**, entre los cuales se pueden mencionar los siguientes:

Métodos teóricos:

- **Histórico – Lógico:** Mediante este método se analizará la evolución y desarrollo del objeto de investigación y sus elementos más importantes.
- **Analítico – Sintético:** Se usará para analizar la información de las técnicas existentes para visualización de volúmenes.
- **Modelación:** Se empleará para realizar una representación simplificada de la realidad a través de diagramas de clases, de flujo y de componentes.

Métodos empíricos:

- **Pruebas:** Se realizarán diferentes pruebas al algoritmo implementado para determinar si se comporta según los resultados esperados.
- **Observación:** Se empleará para constatar los resultados visuales alcanzados y determinar la influencia de estos sobre el rendimiento de la computadora.
- **Consulta a especialistas:** Se consultarán distintos profesionales nacionales y extranjeros con conocimientos sobre el tema con el objetivo de recibir orientaciones y validar los resultados.

A continuación se muestra la estructura del presente trabajo, incluyendo una síntesis de los capítulos y secciones fundamentales:

Capítulo 1. Fundamentación Teórica.

En este capítulo se definen los principales conceptos que serán empleados durante todo el trabajo y se presentan las bases teóricas fundamentales relacionadas con las endoscopias tradicionales y virtuales. Se abunda en los conceptos y algoritmos de Visualización Indirecta de Volumen y Visualización Directa de Volumen, mostrando las ventajas que presenta su implementación en la GPU. Se presentan los modelos teóricos para Visualización Directa de Volumen y el principio de funcionamiento del algoritmo Raycasting.

Capítulo 2. Características del sistema.

Se propone una solución técnica al problema planteado, haciendo uso del algoritmo Raycasting basado en GPU, así como las adaptaciones realizadas por el autor para resolver el problema existente, entre estas, especial atención merecen la Navegación dentro del volumen, presentada en el epígrafe 2.2.1 y la Intersección con geometría, perteneciente al epígrafe 2.2.2. Se proponen técnicas para el mejoramiento

de la calidad de la imagen y del rendimiento del sistema. Se mencionan las metodologías y herramientas de desarrollo empleadas en la realización de este trabajo.

Capítulo 3. Descripción de la Solución.

Durante este capítulo se describe el sistema desde la perspectiva de Ingeniería de Software, usando el Proceso Unificado de Desarrollo como metodología. Se presentan las reglas específicas del negocio y el modelo de dominio del problema. Se realiza la Captura de Requisitos y el modelo de Casos de Uso del Sistema. Posteriormente se muestra el Diagrama de Clases del Diseño y los Diagramas de Secuencia correspondientes a los Casos de Uso.

Capítulo 4. Implementación y validación de los resultados.

En este capítulo se abordan los temas relacionados con la implementación del sistema, el mismo se basa en el trabajo desarrollado en los capítulos anteriores. Posteriormente se toman casos de prueba para validar los resultados alcanzados, fundamentalmente relacionados con el rendimiento del sistema y la calidad de la representación.

Anexos: En esta sección se incluyen imágenes obtenidas con el algoritmo propuesto, integradas con la aplicación del proyecto Vismedic.

Glosario de Términos: Se elaboró un Glosario de Términos con el objetivo de facilitar la comprensión del lenguaje utilizado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se definen los principales conceptos que serán empleados durante todo el trabajo y se presentan las bases teóricas fundamentales relacionadas con las endoscopias tradicionales y virtuales. Se abunda en los conceptos y algoritmos de Visualización Indirecta de Volumen y Visualización Directa de Volumen, destacando las ventajas que presenta su implementación en la GPU. Especial atención debe prestarse a las secciones 1.5 y 1.6, donde se presentan respectivamente los modelos teóricos para Visualización Directa de Volumen y el principio de funcionamiento del algoritmo Raycasting.

1.1. Endoscopias tradicionales

Según el diccionario de la Real Academia de la Lengua Española [6] una endoscopia se define como una técnica de exploración visual de una cavidad o conducto del organismo, la palabra procede del griego ´endo´ que significa ´dentro´ y ´skopíā´ que significa ´exploración´ o ´examen visual´.

Dentro de la terminología médica, una endoscopia es un procedimiento en el que se usa un endoscopio para examinar el interior del cuerpo. Un endoscopio es un instrumento delgado con forma de tubo, con una luz y una lente para observar. También puede tener una herramienta para extraer tejido y observarlo bajo un microscopio con el objetivo de verificar si hay signos de enfermedad [7].

El endoscopio contiene canalizaciones en su interior que permiten:

- Inyectar aire o líquido para distender el tubo digestivo e inspeccionarlo, así como lavar la zona inspeccionada.
- Aspirar y tomar muestras de la superficie del tubo digestivo a estudiar.
- Introducir micropinzas para la realización de biopsias y tomar muestras de tejido para su estudio microscópico, extraer pequeños cuerpos extraños que se hayan ingerido accidentalmente, pólipos de la mucosa digestiva, cauterizar varices o lesiones hemorrágicas, etc.
- Introducir microtijeras, y otras herramientas para realizar intervenciones en el esófago, estómago o intestino (extracción de pólipos, tumores, etc.).

Hay muchos tipos de endoscopias y reciben el nombre de acuerdo con el área u órganos que exploran [7]. Por ejemplo:

- Artroscopia: empleada para examinar directamente las articulaciones.
- Broncoscopia: empleada para examinar los pulmones.

- Cistoscopia: empleada para visualizar el interior de la vejiga.
- Laparoscopia: empleada para examinar directamente los ovarios, el apéndice u otros órganos abdominales.

Como técnica quirúrgica permite resolver determinados problemas sin necesidad de abrir el abdomen y el tubo digestivo, reduciendo los riesgos y complicaciones, y permitiendo una recuperación mucho más rápida del paciente.

Sin embargo, ningún procedimiento médico está exento de complicaciones [7], algunas de estas son:

- Hemorragia digestiva.
- Perforación de la pared del tejido inspeccionado.
- Complicaciones secundarias a la anestesia general, si ésta ha sido necesaria.

1.2. Endoscopias virtuales

Una endoscopia virtual es una técnica mediante la cual, las imágenes adquiridas por CT o MRI, son procesadas por la computadora y reconstruidas de forma tridimensional para lograr una visualización similar a la obtenida a través de un endoscopio [8].

Endoscopia y Realidad Virtual se unen en diferentes puntos, desde la "telepresencia" (inmersión, interacción y navegación), hasta la posibilidad de lograr "quirófanos virtuales" que permitan al alumno entrenarse en un ambiente con similares condiciones y posibles herramientas, cavidades anatómicas y pasos técnicos del proceder antes de realizarlo en pacientes reales [9].

Con la realización de endoscopias virtuales el joven médico se entrenaría con más precisión, desarrollaría reflejos seguros, activaría su banco mental de imágenes propias del escenario de trabajo, evaluaría su temor a provocar daños y de paso ganaría confianza en él mismo [9].

Esto permitiría al profesor decidir si el alumno se encuentra capacitado para continuar su entrenamiento con menos riesgos de provocar daños en pacientes reales. De esta forma el estudiante se presentaría a la endoscopia diagnóstica y quirúrgica con el rigor técnico y ético exigido en la actualidad.

Para que una endoscopia virtual pueda considerarse útil en la medicina, es necesario usar técnicas de visualización de volumen que representen con exactitud la anatomía real de los pacientes que se van a examinar.

1.3. Volumen de datos

Los datos adquiridos para generar un volumen son usualmente representados como un conjunto de imágenes individuales. Cada imagen representa un corte de una parte del objeto y está compuesta por píxeles (elementos de imagen). Estos píxeles están organizados en una red bidimensional, 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 permite el cálculo de la posición actual, multiplicando el valor de la distancia con el índice del píxel. Si se asume que i indexa la posición horizontal y j indexa la vertical, la posición del píxel $P_{i,j}$ puede ser calculada [1]. La red 2D de la imagen se muestra en la Fig. 2 Izquierda.

Los datos volumétricos combinan imágenes individuales en una red 3D (ver Fig. 2 Centro). Los elementos individuales ahora son llamados vóxeles (elementos del volumen). En adición a las dimensiones horizontal (x) y vertical (y), se incrementa una dimensión representando la profundidad (z).

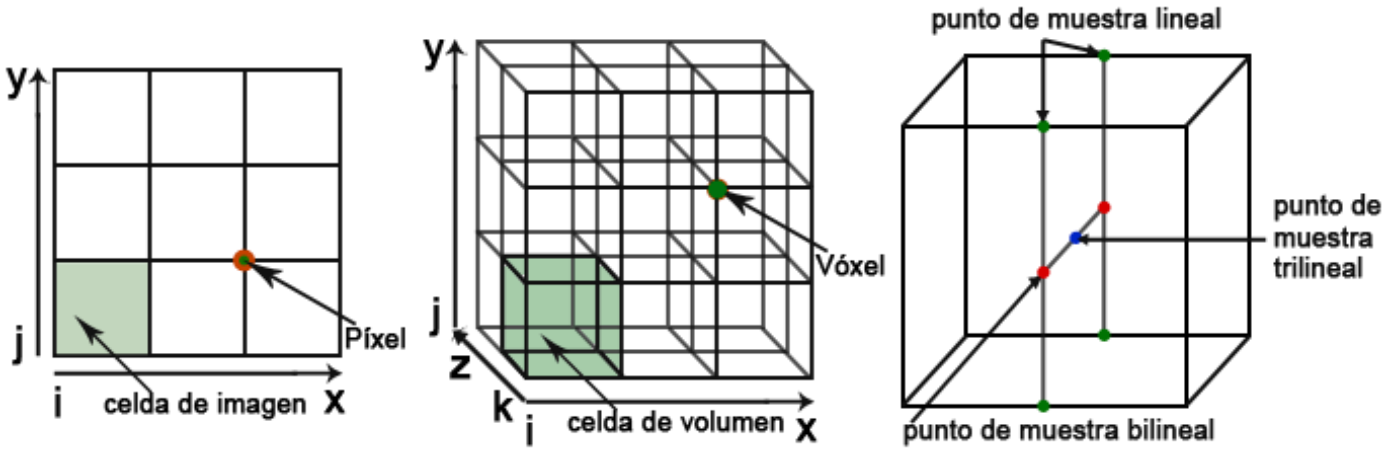


Fig. 2 Izquierda: Red de imagen 2D. Centro: Volumen de datos. Derecha: Interpolación trilineal.

Un volumen de datos está definido solo por posiciones discretas de la red 3D. Sin embargo, en muchos casos se necesita calcular puntos dentro de una celda del volumen. El método más empleado para calcular dichos puntos es la interpolación trilineal [1], la cual está compuesta por siete interpolaciones lineales (ver Fig. 2 Derecha). 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 muestra trilineal.

1.4. Visualización de volumen

La Visualización de Volumen consiste en la representación visual del conjunto completo de datos volumétricos [1], por tanto, de todas las imágenes al mismo tiempo. Los vóxeles individuales deben ser seleccionados, combinados y proyectados sobre el plano de una imagen. Esta imagen actúa literalmente como una ventana de los datos, representando la posición y la dirección desde donde el volumen es examinado por un observador.

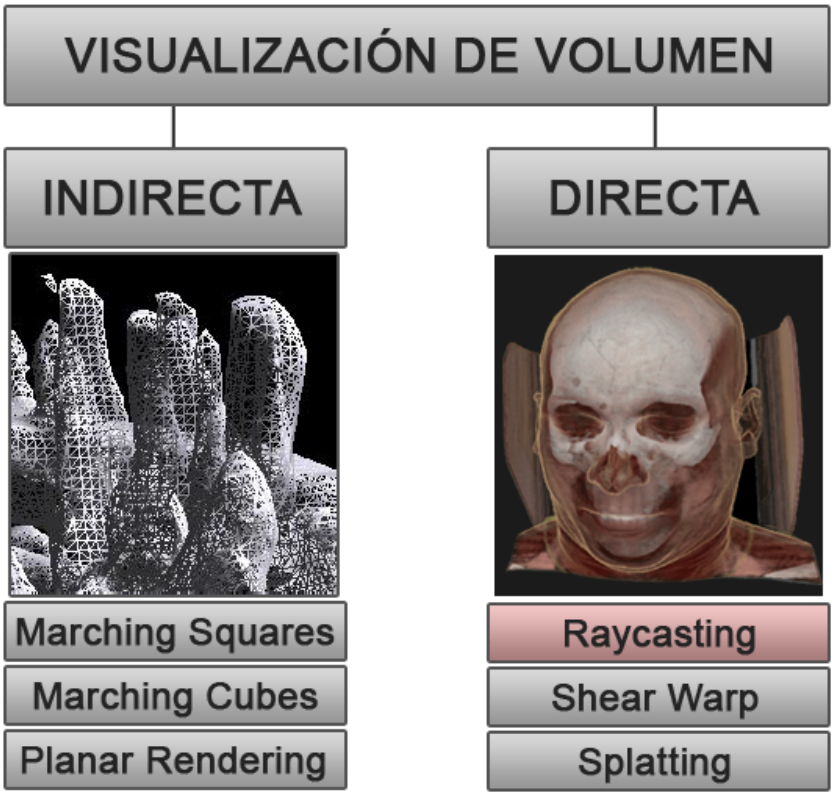


Fig. 3 Visualización de Volumen.

1.4.1. Visualización Indirecta de Volumen

La IVR es una popular forma de visualizar datos volumétricos, la cual genera una representación intermedia (Ej.: superficie de polígonos) que es mostrada posteriormente. Los algoritmos de IVR se basan fundamentalmente en el uso de planos (Planar Rendering) o superficies geométricas como representación intermedia (Surface Rendering).

Actualmente existen varios algoritmos para resolver el problema de la reconstrucción de superficies de un volumen de datos, dos de los más usados en la actualidad son: el algoritmo Marching Cubes propuesto por William E. Lorensen y Harvey E. Cline en 1987 [10] y el Marching Tetrahedra [11].

Algoritmo 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 Fig. 4) donde aparecen los 15 casos posibles en los que un cubo puede ser interceptado por la superficie.

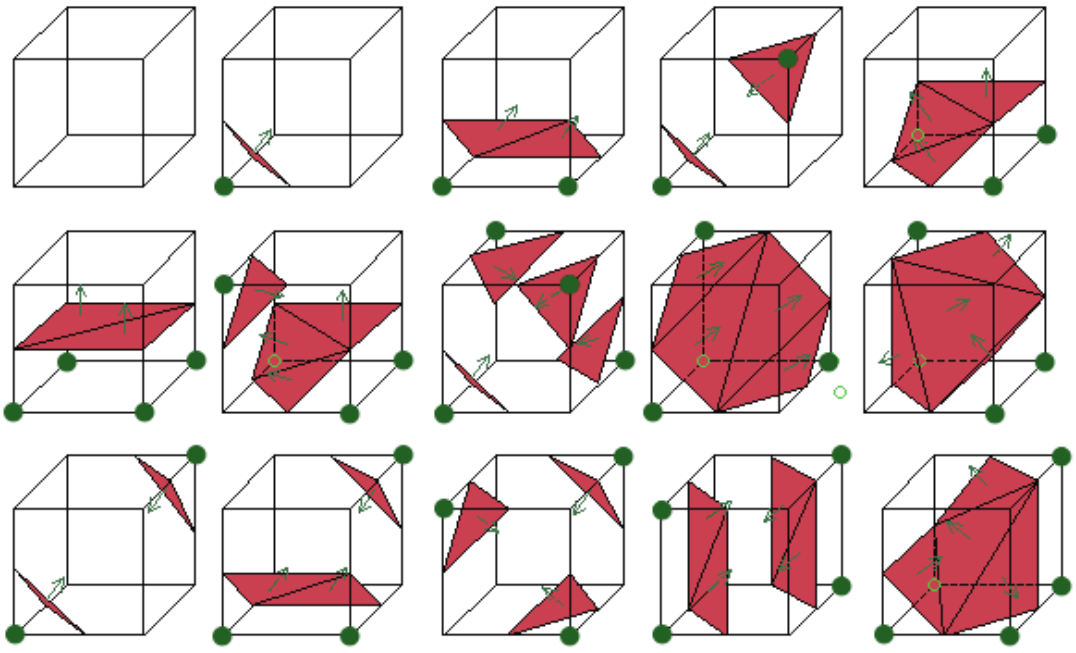


Fig. 4 Tabla de búsqueda del algoritmo Marching Cubes.

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, brinda una manera 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 [12]. 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 [13].

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.

Planar Rendering

La clave de esta técnica de representación 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.

1.4.2. Visualización Directa de Volumen

Mediante esta técnica de visualización de volumen, los datos volumétricos son representados mediante su evaluación en un modelo óptico, que describe como el volumen emite, refleja, dispersa, absorbe y ocluye la luz [1].

La complejidad de los algoritmos de DVR depende del número de vóxeles de la red y del número de píxeles de la imagen final donde será proyectada la visualización.

Los algoritmos de DVR se pueden clasificar en dos grupos: algoritmos basados en imágenes y algoritmos basados en objetos. Sin embargo, muchas variaciones avanzadas de estos algoritmos no pueden ser clasificadas estrictamente dentro de uno de los grupos, porque fusionan aspectos de ambos grupos en un mismo algoritmo.

Algoritmos basados en imágenes

El algoritmo clásico de visualización directa de volumen es el Raycasting [1]. Está basado en el concepto de un rayo de luz que penetra en el volumen e intercepta los vóxeles que encuentra a su paso.

Por cada píxel de la ventana a través de la cual se observa la representación tridimensional, se lanza un rayo hacia el interior del volumen, recorriendo una trayectoria rectilínea que comienza en la posición de la cámara virtual y es paralelo al vector direccional de la misma [18][19]. 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. En el **Epígrafe 1.5** se profundiza el análisis de este algoritmo.

Otro algoritmo frecuentemente usado es el Shear Warp, el mismo es una versión optimizada del Raycasting. Su idea básica es simplificar el proceso de toma de muestras para reducir los costos de acceso a memoria.

Shear Warp parte de un plano base que es siempre paralelo al eje del volumen de datos que más se ajusta al punto de vista del observador, lo que posibilita un patrón muy regular para la toma de muestras. Para mayor información sobre el algoritmo Shear Warp remitirse a [1], sección 9.2.

Algoritmos basados en objetos

Los dos algoritmos previos (Raycasting y Shear Warp) basan su funcionamiento en la toma de muestras a lo largo de rayos que viajan a través del volumen. El algoritmo Splatting, en contraste, proyecta los vóxeles hacia el plano de la imagen final [1]. Para esto es necesario aplicar un filtro de volumen que mantenga constante las propiedades geométricas de los objetos, este filtro es generalmente una función Gaussiana.

Uno de los más recientes algoritmos de DVR es Texture-Mapping, descrito en 1994 por [15], el mismo se sustenta en las funcionalidades para el manejo de texturas que presentan las tarjetas gráficas actuales. En esencia, el volumen de datos es cargado dentro de la memoria de textura y subsecuentemente encuestado por las funciones de manejo de textura hasta ser mapeado hacia un polígono que funcionará como geometría intermedia entre los datos y su representación en la pantalla. Una descripción detallada de este algoritmo se encuentra en [1], Sección 9.4.

1.4.3. Visualización de Volumen Híbrida

En muchas aplicaciones es necesario combinar objetos geométricamente definidos con otros que presentan características volumétricas. En los Simuladores Quirúrgicos, por ejemplo, se representan los objetos quirúrgicos, que se emplean para manipular los datos volumétricos, mediante geometría poligonal.

Existen dos aproximaciones básicas para alcanzar el resultado anterior:

- Se convierte una representación en la otra y se fusionan los resultados.
- Se representan independientemente y luego los resultados son mezclados en la imagen final.

Con respecto a la primera opción, el volumen de datos puede ser transformado en una representación poligonal, por ejemplo usando el algoritmo Marching Cubes, o la representación poligonal puede ser transformada en una red de vóxeles, lo que se conoce como voxelization [1] sección 7.3.4.
En la mayoría de los casos siempre es mejor preservar las representaciones originales y mezclarlas en una imagen bidimensional.

1.5. Modelos teóricos para Visualización Directa de Volumen

Los modelos teóricos incrementan el grado de realismo físico en la visualización directa de volumen. 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.
Si eliminamos la refracción y la reflexión se alcanza el modelo físico estándar, donde solamente se considera la emisión y la absorción [20]. En este modelo cada partícula contribuye al volumen como una pequeña fuente de luz atenuada por la absorción.
A continuación serán explicados los modelos emisión y absorción, para más detalles en el artículo de Nelson Max [20] son descritos los modelos ópticos más importantes.

Emisión

De este modelo físico se asume que cada partícula en el volumen de datos es una pequeña fuente luminosa que emite su luz a través del volumen, donde la luz no es atenuada ni dispersada; por lo tanto, existe sin ninguna interacción con el volumen [1] [20]. La fuente de luz es modelada por el término **Qλ(s)**, donde **λ** especifica la longitud de onda de la luz emitida y **s** es la dirección del rayo de luz que viaja desde el punto de vista del observador a través del volumen acumulando la luz. La ecuación **1** implementa el cálculo de la intensidad **I(s)** en la posición **s** a lo largo del rayo **S**, basado en **Q(s)** [1], por la siguiente ecuación diferencial:

$$\frac{dI}{ds} = Q(s) \qquad (1)$$

La solución de esta ecuación diferencial se obtiene mediante el cálculo de la integral

$$I = I_{s0} + \int_{s0}^{s} Q(t)dt \qquad (2)$$

Donde s_0 es el punto de entrada en el volumen, I_{s0} es el valor inicial de la intensidad de la luz cuando el rayo entra al volumen, el cual puede ser considerado como algún tipo de luz ambiental. El término **Q(s)** representa la contribución del volumen a la imagen final y puede ser especificado por **Q(s) = c * v(s)**, donde **c** es una constante y **v(s)** es el valor que tiene el volumen en la posición **s**.

Absorción

El modelo físico de solo absorción asume que cualquier intensidad de luz absorberá todas las partículas [1]. La ecuación **3** define esta absorción en una posición **s** a lo largo del rayo **S** con la siguiente ecuación diferencial:

$$\frac{dI}{ds} = -\tau(s) \cdot I(s) \tag{3}$$

Donde **I(s)** es la intensidad de la luz en la posición **s** a lo largo del rayo **S**. $\tau(s)$ es llamado coeficiente de extinción, el cual define la atenuación de la intensidad a lo largo del rayo [1]. Si se está empleando para la representación algún tipo de Función de Transferencia, se puede utilizar la opacidad como coeficiente de extinción. La transparencia y la opacidad son valores complementarios. Si asumimos una transparencia **τ ∈[0..1]** podemos calcular la opacidad por **α(s) = 1 − τ(s)**. Finalmente, la solución a la ecuación diferencial **3** sería:

$$I(s) = I_{s0} \cdot e^{\left(-\int_{s0}^{s} \tau(s)dt\right)} \tag{4}$$

donde I_{s0} es la intensidad para **s = 0**, cuando el rayo entra al volumen.

Ecuación de la Visualización de Volumen

El modelo físico estándar para la visualización directa de volumen emplea emisión y absorción [1] [20], combinando ambos términos:

$$\frac{dI}{ds} = Q(s) - \tau(s) \cdot I(s) \tag{5}$$

De la ecuación diferencial **5** se obtiene la ecuación de la visualización directa de volumen [1] [20]:

$$I(s) = I_{s0} \cdot e^{\left(-\int_{s0}^{s} \tau(s)dt\right)} + \int_{s0}^{s} Q(p) \cdot e^{\left(-\int_{p}^{s} \tau(t)dt\right)} dp \quad (6)$$

En la práctica el volumen de datos es discreto, por lo que la evaluación de la integral es aproximada numéricamente. Esta aproximación es hecha por la suma de Riemann y en un paso de tamaño fijo Δs que escala la muestra actual de la posición **k** a lo largo del rayo **S** con $k \cdot \Delta s$. Primero se procede a discretizar el siguiente término:

$$e^{\left(-\int_{s0}^{s} \tau(s)dt\right)} = e^{\left(-\sum_{k=0}^{n-1} \tau(k \cdot \Delta t)\Delta t\right)} = \prod_{k=0}^{n-1} e^{-\tau(k \cdot \Delta t)\Delta t} = \prod_{k=0}^{n-1} t_k \quad (7)$$

Donde asumimos que el punto de entrada del rayo **S** al volumen es ahora la posición 0 en vez de s_0, con $n-1$ como la posición $(n-1) \cdot \Delta s$ a lo largo del rayo **S** y con $\mathbf{t_k}$ como la transparencia del punto **k**. Si se discretiza la ecuación de la visualización directa de volumen (ver ecuación **6**) con $s_k = k \cdot \Delta s$, $Q_k = Q(s_k)$ y la discretización realizada en la ecuación **7** se obtiene:

$$I(s) = I_0 \prod_{k=0}^{n-1} t_k + \sum_{k=0}^{n-1} Q(k \cdot \Delta s) \cdot \Delta s \prod_{j=k+1}^{n-1} t_j \quad (8)$$

La ecuación describe como el rayo **S** con el primer sumando de la ecuación **8** como valor inicial atraviesa el volumen acumulando contribuciones en las posiciones discretas **k**. Estas contribuciones son basadas en el término Q_k que es atenuado por la transparencia que ha sido acumulada de t_j en las posiciones **j** a lo largo del rayo. Esta ecuación provee las bases teóricas para la visualización directa de volumen y define como los vóxeles del volumen de datos contribuyen a la imagen final [1].

1.6. Principio del algoritmo Raycasting

El algoritmo Raycasting fue propuesto inicialmente por Marc Levoy en 1988, su artículo Display of Surfaces from Volume Data [18] puede considerase como el punto de partida para cualquier implementación de este algoritmo.

Raycasting es un algoritmo de DVR basado en imágenes que utiliza una evaluación directa de la Ecuación de la Visualización de Volumen [21]. Por cada píxel de la imagen a representar, se proyecta un rayo hacia la escena y se integran las propiedades ópticas obtenidas del volumen a lo largo del rayo generalmente usando interpolación trilineal como filtro de reconstrucción. La Fig. 5 muestra gráficamente el principio básico del Raycasting.

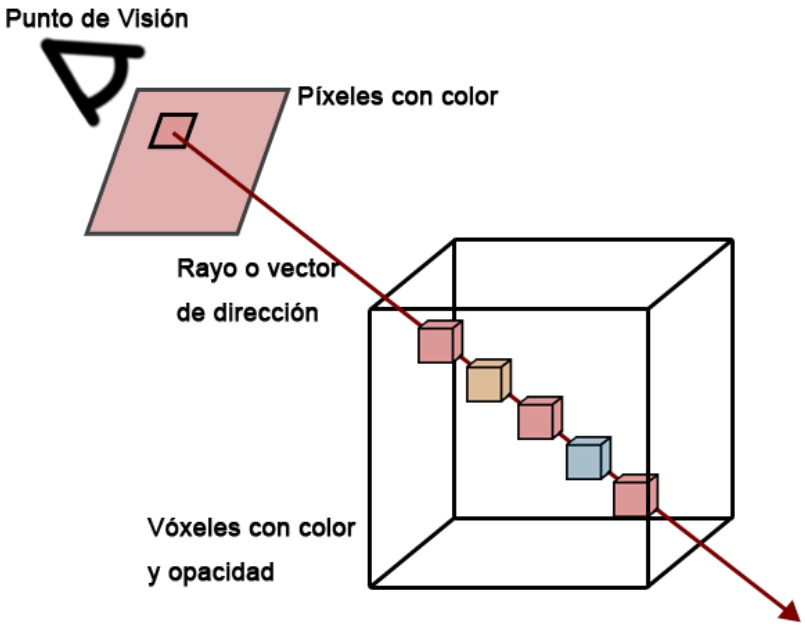


Fig. 5 Principio de funcionamiento del algoritmo Raycasting.

En la práctica, por supuesto, los datos volumétricos son discretos, por tanto la integral de DVR debe ser aproximada numéricamente. En combinación con otras simplificaciones, la integral es normalmente sustituida por una suma de Riemann.

1.7. Implementaciones basadas en CPU vs GPU

En los últimos años el hardware gráfico se ha desarrollado considerablemente, su uso se ha diversificado hasta alcanzar resultados impresionantes en áreas que no son precisamente gráficas, por ejemplo: bibliotecas de funciones matemáticas, sistemas de sonido y compresión de datos.

Portar los algoritmos clásicos basados en la CPU hacia la GPU no tiene mucho sentido en la mayoría de los casos. Los algoritmos basados en hardware más eficientes intentan aprovechar las ventajas específicas que presenta la GPU sobre la CPU en la mejor forma posible, como es el caso del Raycasting. Algunas de estas ventajas son:

- Arquitectura masivamente paralela.
- Separación en dos unidades de trabajo distintas (vertex y fragment shader) que pueden duplicar el rendimiento si el diseño del algoritmo lo permite.

- Operaciones con vectores y matrices tan rápidas como operaciones escalares.
- Instrucciones dedicadas al trabajo gráfico.

Muchas otras ventajas pueden aparecer dependiendo de la naturaleza del algoritmo a implementar. El algoritmo Raycasting puede tomar ventajas específicas de las siguientes funcionalidades:

- Cálculo automático de las posiciones iniciales de los rayos, dejando al hardware interpolar los valores de color.
- Eficiente interpolación trilineal de Texturas 3D.
- Cambio entre proyección ortogonal y perspectiva sin esfuerzos adicionales.
- Cálculo automático de intersecciones en el buffer de profundidad.

En una implementación basada en CPU, las funcionalidades antes expuestas deben ser desarrolladas en software, lo que representa una sobrecarga adicional para la CPU y disminuye considerablemente el rendimiento del algoritmo. Por tanto, se puede afirmar que las implementaciones de este algoritmo empleando la GPU, sabiendo aprovechar las ventajas que esta presenta, ofrecerán mayor rendimiento y calidad que las realizadas en la CPU.

1.8. Pipeline Gráfico

El propósito fundamental de las tarjetas gráficas 3D es tomar una geometría de datos 3D y proyectarla en una imagen 2D para ser mostrada en la pantalla de la computadora [19]. Una geometría está compuesta fundamentalmente por vértices o puntos en un espacio 3D que agrupados forman primitivas de mayor nivel tales como puntos, líneas, triángulos y polígonos. La tarjeta gráfica toma esta lista de vértices describiendo los objetos de la escena y los proyecta hacia una imagen de píxeles discreta en un proceso llamado rasterización.

El término pipeline en español significa cañería o tubería. Aplicándolo a la informática, el pipeline gráfico está compuesto por un conjunto de etapas de procesamiento gráfico, donde se simula un fluido que debe ser procesado mientras circula por una tubería. En este proceso cada operación depende de la realización de la operación anterior y su orden simplificado es descrito en la Fig. 6.

Las tarjetas gráficas modernas, ahora mucho más programables y con lenguajes de alto nivel, permiten al programador reemplazar la etapa 1 y la 4 del pipeline con su propio programa (shader). El programa que reemplaza la etapa 1 es denominado vertex shader, el cual es ejecutado por cada vértice que se procesa,

mientras que el de la etapa 4 es ejecutado por cada fragmento (pixel) que es producido por el proceso de rasterización.

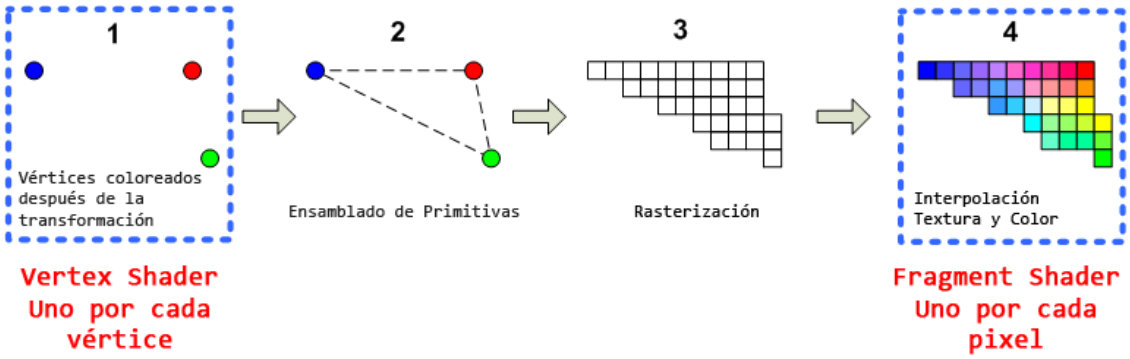


Fig. 6 Pipeline Gráfico simplificado.

Para una mejor comprensión de los temas relacionados con la programación usando hardware gráfico se recomienda el libro *OpenGL Shading Language* [22].

1.8.1. Vertex Shader

Los *vertex shaders* son ejecutados para cada vértice por el Procesador de Vértices (*Vertex Processor*). El *vertex shader* no puede crear o eliminar vértices, sólo modificarlos.

Tradicionalmente esta unidad ejecuta una serie de operaciones fijas en función de las capacidades de la tarjeta gráfica:

- Transformaciones de vértices.
- Transformación y normalización de vectores.
- Generación de coordenadas de textura.
- Mapeo de coordenadas de textura.
- Iluminación.

De todas estas funciones la más importante es la transformación de vértices, la misma se realiza mediante la multiplicación de la matriz de modelado (*Model-View Matrix*) y la matriz de proyección (*Projection Matrix*).

1.8.2. Fragment Shader

Los *fragment shaders* son ejecutados para cada fragmento o píxel por el denominado Procesador de Fragmentos (*Fragment Processor*).

Tradicionalmente esta unidad ejecuta una serie de operaciones fijas en función de las capacidades de la tarjeta gráfica:

- Operaciones con valores interpolados.
- Acceso, interpolación y mapeo de texturas.
- Suma, modulación y mezcla de colores.
- Representación de Niebla.

El aspecto más importante del Procesador de Fragmentos es su capacidad para interpolar valores. Si dos vértices A y B forman una línea, el vértice A tiene color rojo y B tiene color azul, entonces el Procesador de Fragmentos crea una serie de píxeles entre los vértices para crear una línea que comienza con el color de A y termina con el color de B, interpolando de forma suave los colores correspondientes.

1.9. Consideraciones parciales

Hasta este punto se analizaron los conceptos fundamentales concernientes a los procedimientos endoscópicos, tanto reales como virtuales. Se sentaron las bases necesarias para la comprensión del objeto de estudio y su campo de acción, se mostraron los principales algoritmos de Visualización de Volumen, seleccionándose el Raycasting basado en GPU para la realización del presente trabajo.

Se analizaron las principales ventajas que ofrece la implementación del algoritmo seleccionado en el hardware gráfico y se brindó una breve explicación sobre el funcionamiento del pipeline gráfico que permite comprender de manera general los lugares donde los programadores pueden incluir programas propios (shaders).

CAPÍTULO 2. SOLUCIÓN PROPUESTA

En el presente capítulo se propone una solución al problema planteado, como protagonista se toma el algoritmo Raycasting basado en GPU, de este se caracterizan cada uno de sus pasos y se proporcionan detalles de implementación. Seguidamente se muestran las extensiones necesarias al algoritmo original para su uso en aplicaciones de endoscopias virtuales (Navegación dentro del volumen e Intersección con geometría). Se proponen técnicas para el mejoramiento de la calidad de la imagen y del rendimiento del sistema. Se adicionan planos de corte paralelos a los ejes de coordenadas con el objetivo de mejorar la capacidad de exploración del algoritmo y se mencionan las metodologías y herramientas de desarrollo empleadas en la realización de este trabajo.

2.1. Algoritmo Raycasting basado en GPU

Para la construcción de los rayos, es necesario conocer la posición de entrada y salida del volumen. Estas posiciones se pueden obtener a través de la representación de una geometría limitante o frontera, la misma cumple con una simple característica: la posición de los vértices coincide con el color de cada uno de ellos. La geometría limitante más común es un simple cubo (observar Fig. 7).

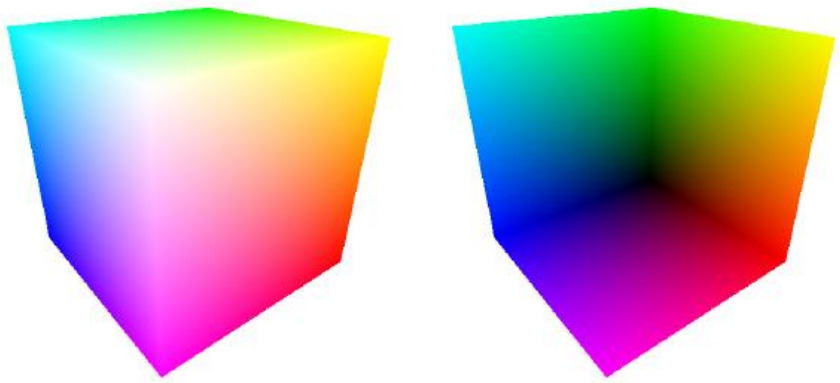


Fig. 7 Geometría limitante. Izquierda: Caras delanteras, Derecha: Caras traseras.

Representando sólo las caras delanteras de la geometría limitante se obtienen las posiciones iniciales de los rayos, mientras que la representación de las caras traseras, en una segunda pasada, obtiene las respectivas posiciones finales [19].

La diferencia de estas dos imágenes crea una **imagen direccional** o **textura direccional,** que representa el actual vector de visión (rayo) para cada uno de los píxeles en coordenadas de volumen. De esta forma, una simple búsqueda dentro de la textura direccional obtiene el vector de visión al cual se le incrementará en cada paso un pequeño delta. Este procedimiento se repite hasta que el rayo abandone el volumen[1].

Todos los valores obtenidos a lo largo del rayo son mezclados, almacenados en un *buffer* y visualizados en la pantalla en una última pasada de render. El color final del píxel es equivalente al cálculo de la integral a lo largo del rayo o al menos una buena aproximación a este valor si la frecuencia para la toma de muestras (frecuencia de muestreo o sampling rate) es suficiente (ver epígrafe 1.5).

Dicho lo anterior, se puede concluir que el algoritmo Raycasting que se propone en la presente investigación está compuesto por cuatro fases:

1. **Generación de las caras delanteras:** se representan las caras delanteras de la geometría limitante hacia un buffer, con los colores equivalentes a las posiciones de los vértices.
2. **Generación de la Textura Direccional:** de la misma forma se representan las caras traseras, se sustrae de la imagen de las caras delanteras y se guarda normalizada.
3. **Raycasting:** con las posiciones iniciales y finales obtenidas anteriormente se avanza por el rayo hasta que este abandone el volumen.
4. **Composición:** Se mezclan los resultados y se muestran en pantalla.

El orden de ejecución de cada uno de estos pasos se muestra en la Fig. 8. Nótese que la generación de las caras traseras y de la textura direccional se considera como un solo paso.

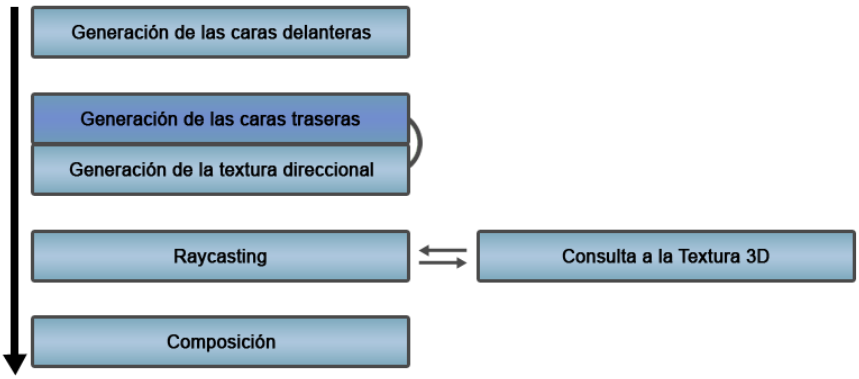


Fig. 8 Pasos fundamentales del algoritmo Raycasting.

[1] En implementaciones más avanzadas se detiene también cuando la opacidad acumulada alcanza el límite (Terminación Temprana del Rayo).

En los siguientes epígrafes se profundiza en cada uno de estos pasos.

2.1.1. Generación de las caras delanteras

La generación de las caras delanteras tiene el objetivo de proveer las posiciones iniciales de los rayos. Esto se logra representando sólo las caras delanteras de la geometría limitante, donde, como se menciona anteriormente, el color de los vértices coincide con su posición espacial. Esto significa que si el vértice se encuentra en la posición (x, y, z) = (0.0, 0.0, 0.0), entonces su color durante la representación será (r, g, b) = (0.0, 0.0, 0.0), el color resultante en este caso será el negro.
Como la geometría limitante es siempre convexa, no existe más de una cara frontal para un píxel en particular, haciendo innecesario cualquier test de profundidad [23].
La imagen resultante es un simple cubo de colores, que se muestra a la izquierda en la Fig. 7. Esta imagen es almacenada en una textura independiente para su posterior consulta.

2.1.2. Generación de la textura direccional

Para la generación de la textura direccional, sólo las caras traseras de la geometría limitante son representadas. Nuevamente, sólo existe una cara trasera para cualquier pixel. Esta vez, el resultado no es directamente guardado en una textura independiente, pero es proporcionado como entrada a un fragment program, que se encargará de calcular la textura direccional.

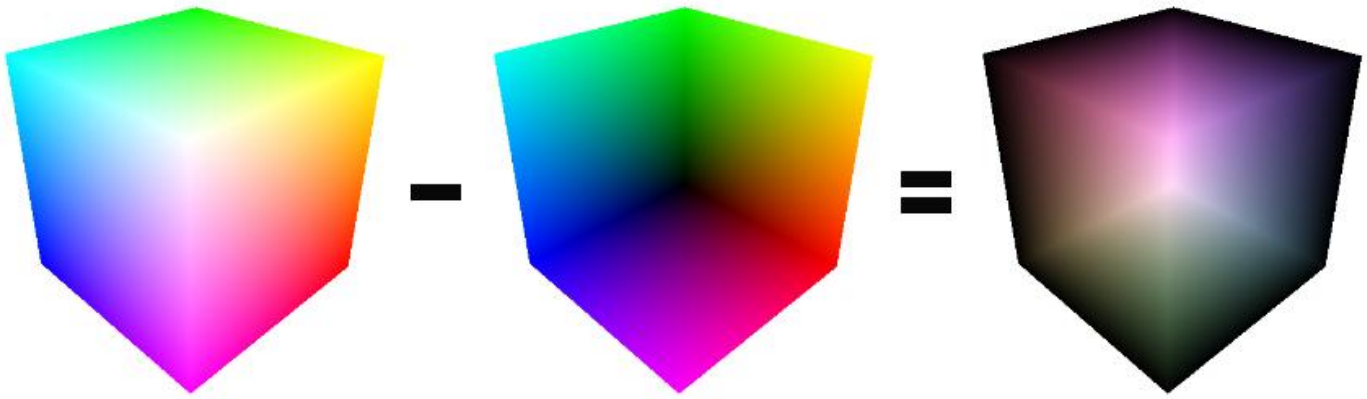


Fig. 9 Obtención de la textura direccional.

La imagen ubicada a la izquierda en la Fig. 9, muestra las actuales caras traseras de la geometría limitante.

El fragment program tiene como entrada el valor de color de las caras traseras y la posición del píxel actual. Luego realiza una búsqueda en la misma posición dentro de la textura de las caras delanteras para obtener la posición inicial. Restando estos dos valores (ver Fig. 9 derecha) se obtiene el vector de visión o vector direccional. La obtención y normalización de este vector puede ser realizada sencillamente en el fragment program debido a que presenta instrucciones dedicadas para este tipo de cálculos.

La diferencia entre las posiciones iniciales y finales de los rayos siempre resulta en correctos vectores de visión para cualquier posición dada y funciona para visualizaciones ortogonales y en perspectiva, lo que posibilita el uso de este algoritmo en un amplio rango de aplicaciones. Toda la corrección de perspectiva y el cálculo del vector de visión se llevan a cabo por el hardware gráfico sin prácticamente consumir tiempo [23], la representación de dos cubos de colores no es ningún problema para un procesador de gráficos moderno.

Con la generación de la textura direccional se tienen preparadas las condiciones necesarias para la realización del paso más costoso computacionalmente: Raycasting.

2.1.3. Raycasting

Para la ejecución de este paso es necesario representar algún tipo de geometría, esta será la encargada de llamar al respectivo *fragment program* para todos los píxeles. Una solución puede ser representar un rectángulo en toda la pantalla, pero como la geometría limitante empleada es bastante simple, se pueden representar nuevamente sus caras delanteras o traseras. De esta forma se asegura que todos los rayos tengan posiciones iniciales válidas y evita la realización de chequeos innecesarios.

El fragment program de este paso toma como entradas el valor de color de las caras delanteras (posición inicial del rayo) y la posición del píxel actual y realiza una búsqueda en la textura direccional, retornando el vector direccional normalizado y su longitud.

Todo lo que queda por hacer ahora es calcular las posiciones de las muestras a lo largo de los rayos, multiplicando el vector de visión o direccional por el desplazamiento (delta) tomado y adicionar este vector a la posición inicial, lo que determina la posición absoluta dentro del volumen. Una búsqueda dentro de la textura 3D devuelve la densidad en la posición dada, automáticamente el hardware gráfico realiza la interpolación trilineal correspondiente.

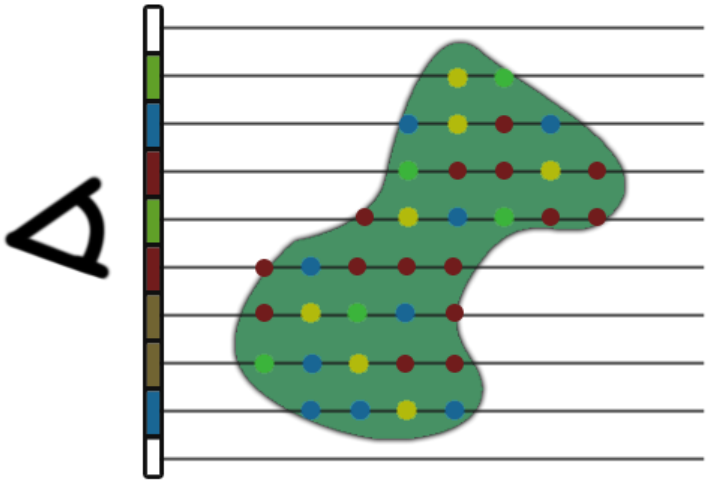


Fig. 10 Contribución de las muestras a la imagen final.

La contribución del color de la muestra es acumulada en una variable y se toma la siguiente muestra hasta que el rayo abandone el volumen (este comportamiento se muestra en la Fig. 10), el color final del píxel se determina mezclando las contribuciones de todas las muestras tomadas a lo largo del rayo. Para hacer uso de las ventajas de los algoritmos basados en imágenes se debe implementar una terminación temprana del rayo, terminando el rayo si la opacidad acumulada excede cierto umbral, generalmente próximo a 1.0.

En el caso de las representaciones de falsas superficies (iso-surface), se detiene la propagación del rayo en la primera intersección significativa, o sea, cuando el valor de la densidad es mayor que un umbral predefinido [23].

2.1.4. Composición

El paso final del Raycasting se encarga principalmente de mostrar los resultados en la pantalla, ejecutarlo en una pasada de render independiente le aporta flexibilidad al algoritmo, permitiendo efectos de post-procesado para ciertas aplicaciones. Adicionalmente brinda la posibilidad de mezclar los resultados de diferentes métodos o de diferentes partes de un mismo volumen que han sido representados de forma independiente por cualquier razón.

En el presente trabajo la intersección con geometría, presentada en el epígrafe 2.2.2, aprovecha la realización de este paso. De todas formas, separar la composición en otra pasada de render no tiene

ningún efecto notable en el rendimiento, por esta razón en la implementación se realiza de esta forma aunque la intersección con geometría esté desactivada.

2.2. Algoritmo Raycasting basado en GPU Avanzado

El algoritmo básico presentado hasta el epígrafe anterior es elegante, simple y razonablemente rápido, de todas formas presenta una serie de inconvenientes que no lo hacen ideal para cualquier tipo de aplicaciones. El primer inconveniente es su imposibilidad de penetrar el volumen de datos para observarlo desde su interior, esta técnica es habitualmente conocida como Endoscopia Virtual.

En segundo lugar, para mejorar la versatilidad del algoritmo es necesario incorporar la representación de otros tipos de geometrías junto con la Visualización Directa de Volumen que ofrece el Raycasting, lo que traerá como resultado escenas de mayor complejidad y realismo visual.

En los epígrafes siguientes se profundiza en los detalles para la eliminación de estos inconvenientes.

2.2.1. Navegación dentro del volumen

Para muchas aplicaciones resulta interesante mover el punto de visión dentro del volumen y explorarlo en modo de vuelo libre, especialmente en aplicaciones de Endoscopias Virtuales, para las cuales ha sido diseñado el algoritmo discutido en el presente trabajo.

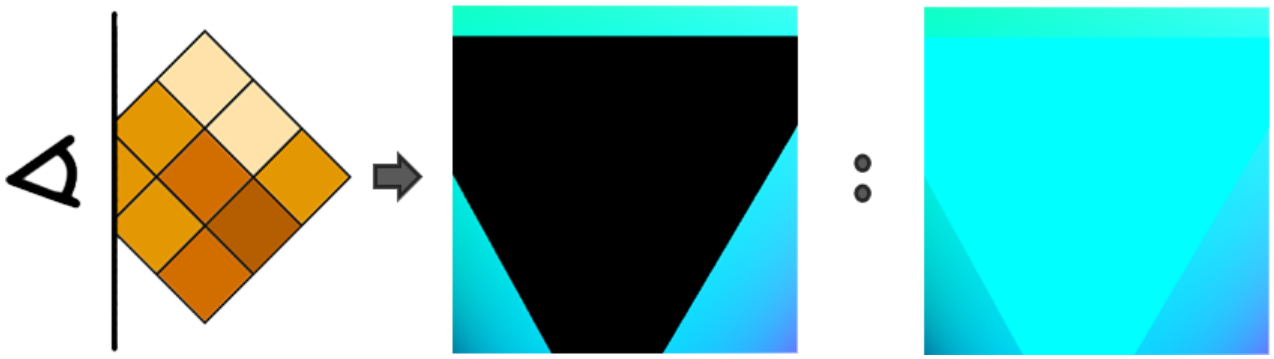


Fig. 11 a) Intersección. b) Huecos resultantes. c) Corrección de la representación.

Considerando el funcionamiento de la geometría limitante representada, no hay ningún problema mientras la cámara virtual no intersecte esta geometría. Pero en cuanto el plano de corte cercano de la cámara (*near clipping plane*) intersecta la geometría limitante, se generan huecos en la representación de las

caras delanteras, trayendo como resultado posiciones iniciales incorrectas para la creación de los rayos. En la Fig. 11 a) y b) se ilustra este comportamiento.

Para solucionar este problema es necesario que en cuanto ocurra una intersección del plano cercano de la cámara con la geometría limitante, se sustituya su posición inicial con la correspondiente posición de la intersección. En la práctica esto significa que los huecos deben ser rellenados con el color correspondiente a la posición donde ocurre la intersección con el plano cercano de la cámara en el píxel actual (ver Fig. 11 c)).

Durante la elaboración del presente trabajo se implementaron dos soluciones distintas a este problema, la primera basada en la transformación de la geometría limitante cuando ocurre alguna intersección, adicionando los triángulos necesarios a la geometría para rellenar los agujeros en cuestión. Esta solución, aunque ofrece buenos resultados visuales, es más compleja algorítmicamente y más propensa al uso de CPU por lo que su ejecución presenta menor rendimiento.

La segunda solución, más elegante y eficiente, implementada en el fragment shader correspondiente a la tercera fase del Raycasting, se encarga de proporcionar posiciones iniciales válidas a los rayos cada vez que ocurre una intersección.

Cuando el valor correspondiente a la textura que representa las caras delanteras en el píxel actual se encuentra por debajo del umbral admitido, generalmente próximo a 0.0, se realiza una proyección inversa que obtenga la posición en coordenadas de volumen que ocupa el plano cercano de la cámara. Para esto es necesario conocer las coordenadas (X, Y) del píxel o fragmento actual utilizando la función gl_FragCoord.xy, luego se transforman las coordenadas de pantalla obtenidas a coordenadas espaciales, multiplicando la posición resultante del paso anterior por las inversas de las matrices de modelado y proyección, esta posición se toma como el color de las caras delanteras.

Hasta este paso se tienen posiciones iniciales válidas para todos los rayos y el algoritmo permite la exploración desde cualquier parte interior del volumen representado.

2.2.2. Intersección con geometría

La combinación de las técnicas de visualización de volumen con las técnicas tradicionales para la representación de los objetos (mallas triangulares, NURBS, etc.) amplía en gran medida la variedad de aplicaciones posibles a desarrollar mediante el uso del Raycasting. Para las aplicaciones de Endoscopias Virtuales resulta necesario combinar la representación volumétrica con el endoscopio y las herramientas

que generalmente se utilizan para estos procedimientos. Para aplicaciones más avanzadas, como simuladores quirúrgicos, se deben representar también las pinzas de corte, cauterizadores, agujas e hilo para suturas.

Un objeto geométrico frecuentemente empleado en las aplicaciones de endoscopias virtuales es una red (grid) uniforme, la misma brinda al usuario mejor orientación espacial en el mundo virtual, principalmente a la hora de estimar distancias y mejora la percepción de la profundidad.

Por otra parte, la introducción de un nuevo componente oclusor contribuye a la optimización del algoritmo original, producto a que la propagación del rayo se detendrá en cuanto intersecte la nueva geometría. Entonces, antes de pensar en adicionar geometría es necesario buscar la forma de no representar el volumen que será ocluido por esta, en este paso vuelve a tomar vital importancia la separación de la representación de las caras delanteras y traseras pertenecientes a la geometría limitante.

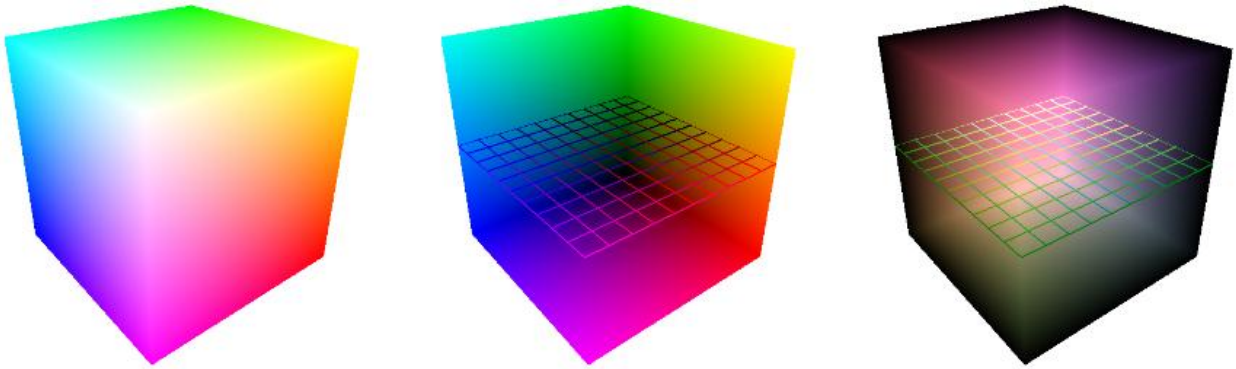


Fig. 12 Izquierda: Representación de las caras delanteras. Centro: Representación de las caras traseras adicionando la intersección con geometría. Derecha: Textura direccional obtenida por la resta de las dos texturas anteriores.

Como se observa en la Fig. 12, cuando ocurra una intersección, la representación de las caras traseras debe ser modificada, este comportamiento se consigue adicionando los nuevos objetos a la representación de las caras traseras de forma que si el objeto se encuentra más cerca del punto de vista, este es el que se representa, con un ligero cambio: los vértices tendrán como color, la posición espacial que ocupan dentro del volumen (igual a la representación de la geometría limitante) [23]. Para el caso particular del grid de orientación, las representaciones de la geometría limitante y la textura direccional se muestran en la siguiente figura.

Después de esto, los vectores direccionales pueden ser calculados de la misma forma, proporcionando longitudes y direcciones correctas aunque se intersecte algún objeto (ver Fig. 12 Derecha). El resultado de esta operación producirá un volumen cortado por una pared invisible, todo lo que resta por hacer es representar la nueva geometría hacia el buffer de pantalla antes de la representación del volumen, y mezclarlos de forma tradicional.

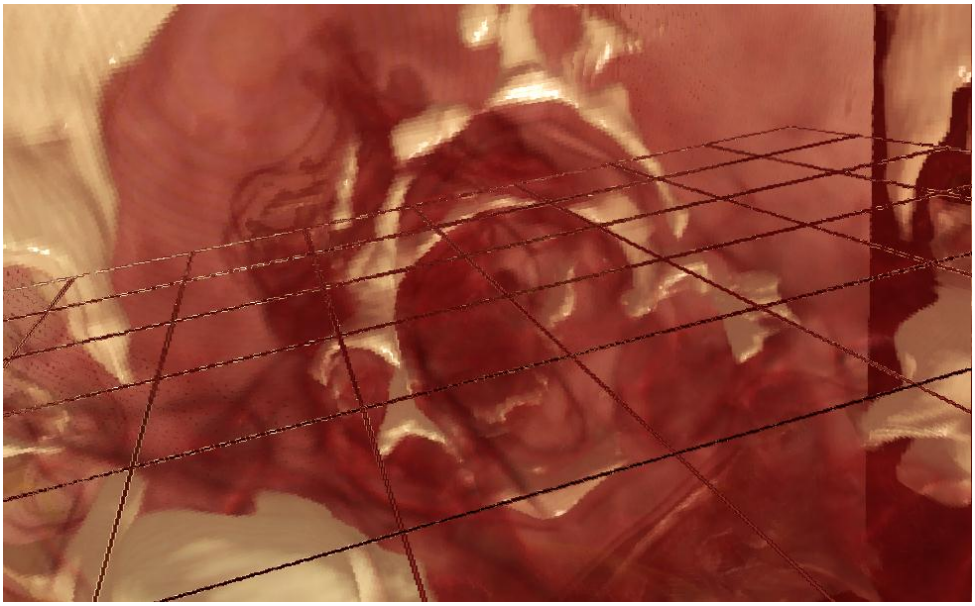


Fig. 13 Endoscopia Virtual con un grid de orientación representado por geometría.

Los resultados de esta modificación del algoritmo se aprecian en la Fig. 13, la percepción de la profundidad y el sentido de la distancia se han mejorado considerablemente.

2.3. Mejorando la calidad de la imagen

El principal problema que presenta cualquier algoritmo basado en Raycasting es la aparición de artefactos debido a la baja frecuencia para la toma de muestras, estos artefactos son claramente visibles en la Fig. 14. Raycasting es sólo una aproximación al cálculo real de la integral a lo largo del rayo, siempre va a existir al menos una pequeña diferencia entre el valor del cálculo real y el aproximado de forma discreta. Las diferencias disminuyen cuando se toma una distancia entre las muestras (delta) muy pequeño, trayendo consigo gran impacto en el rendimiento del sistema en general [23].

Fig. 14 Aparición de artefactos no deseados debido a una baja frecuencia de muestreo.

Disminuir la frecuencia de muestreo durante toda la representación no es una solución aceptable, de esta forma se representaría con alto nivel de detalle zonas que no son precisamente de interés. Una posible solución es representar con mayor detalle las áreas más cercanas al punto de visión y degradar el detalle en las zonas más alejadas, desafortunadamente esta técnica, conocida como toma de muestras adaptativa (adaptative sampling), no hace uso de la principal ventaja presente en la GPU: su capacidad de ejecutar tareas simultáneamente y destruiría el rendimiento alcanzado hasta el momento.

Entonces deben emplearse técnicas que aprovechen la naturaleza del algoritmo Raycasting e introduzcan menos condicionales en el fragment shader. Se seleccionó la técnica: Refinamiento de los puntos de intersección, la misma será explicada en el siguiente epígrafe, esta incrementa la calidad visual de la imagen sin tener gran impacto en el rendimiento.

2.3.1. Refinamiento de los puntos de intersección

La idea del Refinamiento de los puntos de intersección (Hitpoint Refinement) [23] se basa en mantener un paso largo y constante durante las zonas del volumen que no precisan de gran nivel de detalle, fundamentalmente los espacios en blanco, en cambio, donde el rayo intersecta objetos de interés, se disminuye considerablemente el paso para tomar un número mayor de muestras.

Cuando se detecta la primera intersección, el algoritmo retrocede la mitad del paso actual en dirección contraria, si aún se encuentra dentro de una zona de interés, el próximo paso lo ejecuta hacia atrás igualmente siempre a una distancia de la mitad del paso actual. Si por el contrario, se encuentra dentro de una zona de interés, se desplaza hacia delante de la misma forma, siempre tomando la mitad de la distancia anterior para su avance, la Fig. 15 ilustra gráficamente este procedimiento.

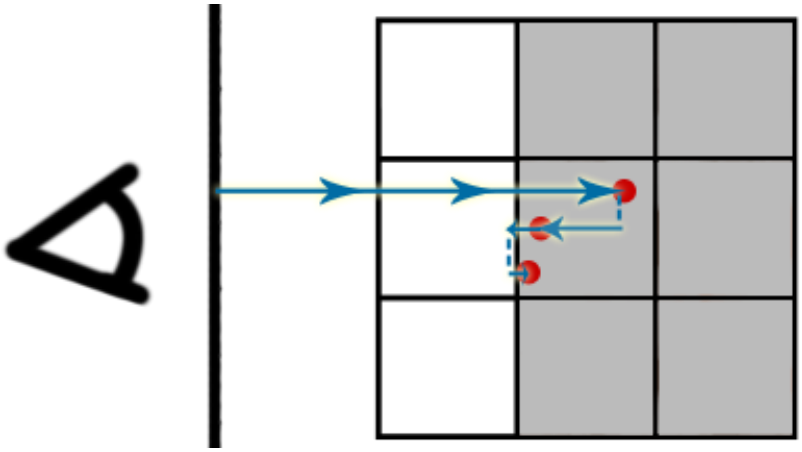


Fig. 15 Refinamiento de los puntos de intersección.

El procedimiento anterior se ejecuta un número finito de veces, en el presente trabajo con 6 iteraciones fue siempre suficiente.

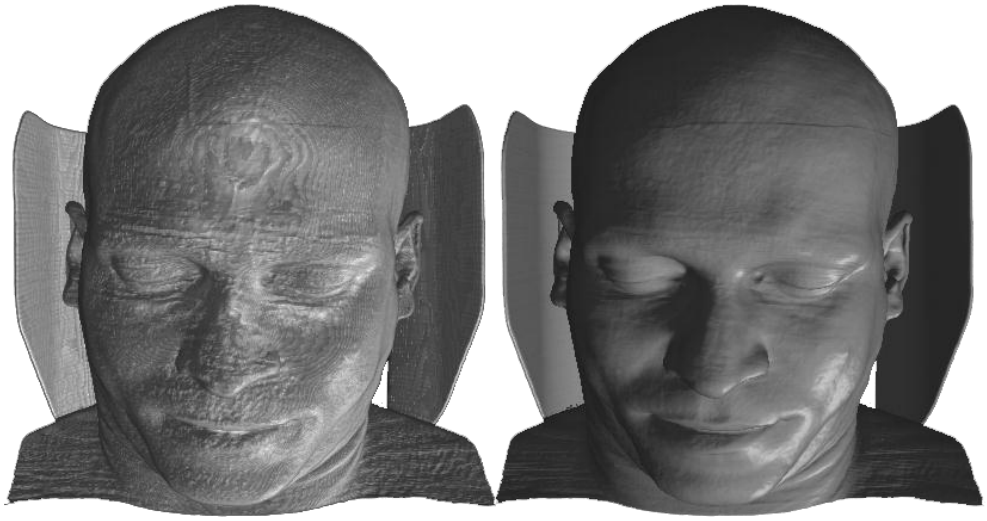


Fig. 16 Refinamiento de los puntos de intersección empleando 6 iteraciones.

En la Fig. 16, se muestran los resultados obtenidos refinando los puntos de intersección, se puede observar que los artefactos presentes en la imagen ubicada a la izquierda han sido eliminados y su calidad visual es superior.

2.4. Técnicas de Optimización

Tanto como la calidad y precisión de la imagen, el rendimiento del sistema es uno de los pilares fundamentales para un buen algoritmo de visualización de volumen. Raycasting se puede optimizar de diferentes formas, siempre respetando su principio de funcionamiento y aprovechando las características de la arquitectura paralela de la GPU.

En los siguientes epígrafes se describen dos modificaciones al algoritmo original que incrementan el rendimiento general del sistema. Los mismos son la terminación temprana de los rayos y una forma simple de saltar o esquivar los espacios en blanco.

2.4.1. Terminación temprana del rayo

Durante la propagación de los rayos se alcanzan zonas que no contribuyen a la imagen final, estas zonas aparecen cuando la opacidad acumulada supera el máximo permisible, generalmente 1.0. Cuando esto sucede, para evitar cálculos innecesarios, es preciso detener el avance del rayo y pasar hacia el próximo [19], la siguiente imagen (Fig. 17) ilustra el comportamiento antes mencionado.

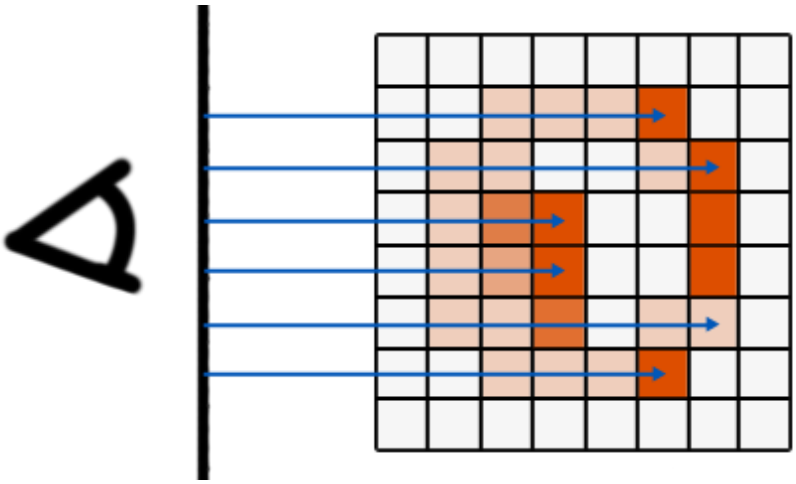


Fig. 17 Terminación temprana del rayo.

2.4.2. Saltar los espacios en blanco

Aunque esta técnica para la optimización del algoritmo Raycasting parece bastante simple y lógica, sus resultados son claramente apreciables. El volumen de espacios en blanco depende de la escena en particular, pero en la mayoría de los casos equivale al 50% del volumen total.

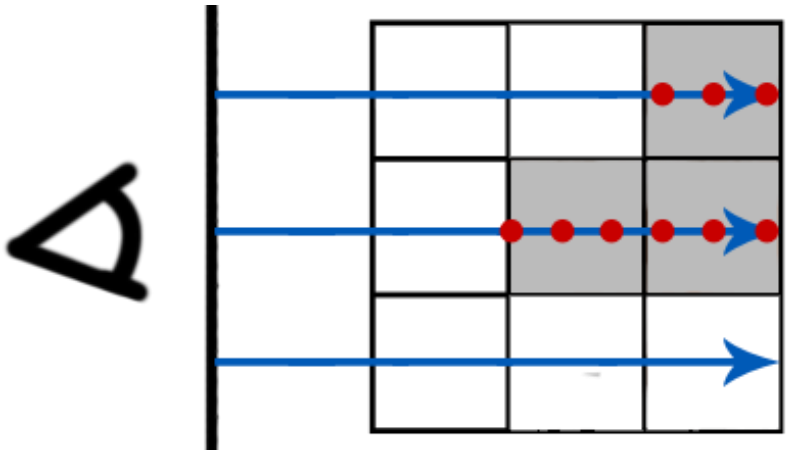


Fig. 18 Saltando los espacios en blanco.

Su implementación es también relativamente sencilla: cuando la propagación del rayo se encuentre en una zona en blanco o por debajo del umbral considerado de interés, se detienen todos los cálculos posteriores y se pasa automáticamente a la siguiente iteración, este comportamiento concluye cuando se alcance una zona significativa dentro del volumen o el rayo en cuestión abandone el volumen [23] (observar Fig. 18).

2.5. Planos de corte

Con el objetivo de mejorar la exploración de áreas específicas dentro del volumen, se adicionaron seis planos de corte al algoritmo, los mismos al ser movidos ocultan o descubren una parte del volumen, como se muestra en la Fig. 19.

Los planos propuestos son siempre paralelos a los ejes de coordenadas, por tanto, para su implementación sólo es necesario almacenar la posición donde se encuentra cada uno. A la hora de tomar las muestras durante la propagación de los rayos, si la posición de la muestra se encuentra fuera del área

que definen los planos (inicialmente todo el volumen), simplemente no se representa y se prosigue al siguiente paso.

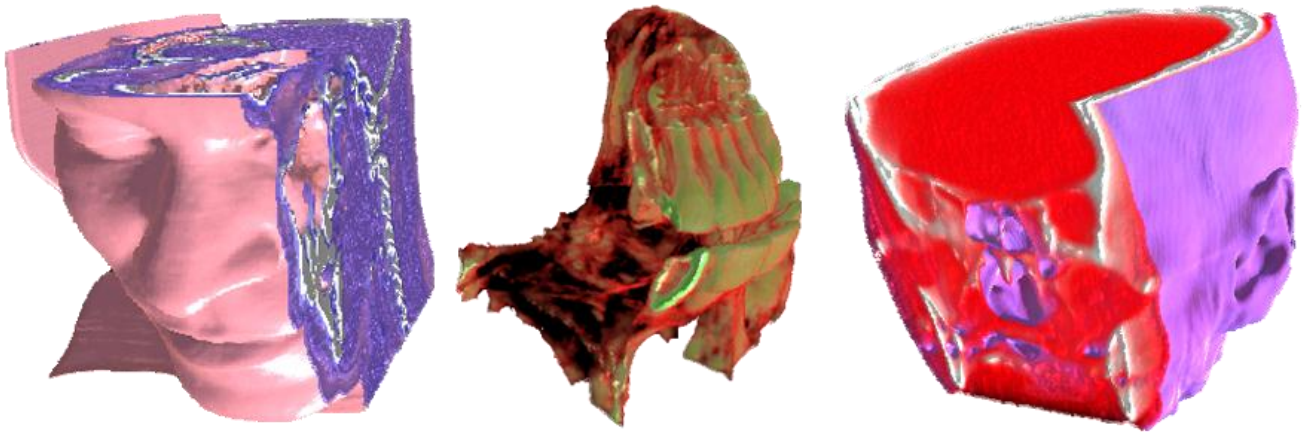


Fig. 19 Planos de corte alineados a los ejes de coordenadas.

2.6. Metodologías y herramientas de desarrollo

Los siguientes epígrafes muestran la Metodología de Desarrollo de Software empleada durante la realización del presente trabajo, así como las principales herramientas que asistieron el proceso de creación de los diagramas y la programación de la solución propuesta.

2.6.1. Metodología de Desarrollo de Software

Se escogió como metodología de desarrollo de software el Proceso Unificado de Desarrollo (RUP). Esta robusta metodología unifica los mejores elementos de las metodologías anteriores y está preparada para guiar el desarrollo de prácticamente todo tipo de proyectos. Su diseño orientado a objetos facilita la comprensión a alto nivel para su posterior implementación usando este paradigma de programación. Dentro sus principales características se encuentran:

Dirigido por casos de uso

Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, estos se obtienen durante el modelado del negocio. El proceso de desarrollo de software avanza a través de una serie de flujos que

parten de los casos de uso, se puede afirmar que estos proporcionan un hilo conductor y una guía para todo el proceso [24].

Centrado en la arquitectura

La arquitectura muestra la visión común del sistema completo y 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. La arquitectura debe diseñarse para que el software evolucione, no solo en su desarrollo inicial, sino también a lo largo de las futuras generaciones [24].

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.

2.6.2. Herramientas de desarrollo

Como herramienta de modelado se empleó Visual Paradigm, creado para asistir el proceso de Ingeniería de Software, este se encuentra basado en UML y soporta el ciclo de vida completo del desarrollo de software, además cuenta con funcionalidades más avanzadas que las presentes en el Rational Rose, lo que permite agilizar considerablemente el trabajo. A continuación se describen sus principales características.

- Presenta licencia gratuita y comercial.
- Soporta aplicaciones web.
- Disponible en varios idiomas.
- Fácil de instalar y actualizar.
- Compatible entre versiones.
- Entorno gráfico amigable para el usuario.
- Disponible en múltiples plataformas (Windows/Linux/Mac OS X).

Para la creación de la interfaz gráfica de usuario (GUI) se utilizó la plataforma QT, la misma permite la portabilidad de la aplicación hacia diferentes sistemas operativos y facilita en gran medida el desarrollo de nuevos componentes gráficos. Entre sus principales características se encuentran:

- Con el mismo código base, permite desplegar el sistema en múltiples plataformas.
- Producir aplicaciones de alto rendimiento con apariencia nativa.
- La concentración de los desarrolladores en la producción de código y no en las particularidades del sistema operativo.
- Acceso total al código fuente para su revisión y modificación.

2.6.3. Lenguaje de modelado

UML es un lenguaje de modelado visual que se usa para especificar, construir, documentar y visualizar artefactos de un sistema de software. El mismo está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Su objetivo es visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo.

Este lenguaje de modelado está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

2.6.4. Lenguaje de programación

Como lenguaje de programación se utilizó C++, lenguaje por excelencia para las aplicaciones de realidad virtual que hace uso eficiente del paradigma de Programación Orientada a Objetos. Permite un excelente control de la memoria y una buena administración de los recursos de la computadora. Dentro de las principales ventajas que presenta el lenguaje C++ se encuentran:

- **Difusión:** al ser uno de los lenguajes más empleados en la actualidad, posee gran número de usuarios y tiene una excelente bibliografía.

- **Versatilidad:** C++ es un lenguaje de propósito general, se puede emplear para resolver cualquier tipo de problema.
- **Portabilidad:** se encuentra estandarizado, por tanto, el mismo código fuente puede ser compilado en diferentes plataformas.
- **Eficiencia:** C++ es uno de los lenguajes más rápidos en tiempo de ejecución.
- **Herramientas:** existe gran cantidad de compiladores, depuradores y bibliotecas de clases basadas en este lenguaje.

2.7. Consideraciones parciales

En este capítulo se propone la solución científico-técnica al problema planteado, la misma reside en la implementación del algoritmo Raycasting basado en GPU, mejorado a partir del empleo de los avances científicos publicados en las fuentes bibliográficas consultadas. Se describen los pasos necesarios para la implementación del algoritmo (generación de las caras delanteras, generación de la textura direccional, raycasting y composición), así como las técnicas empleadas para el mejoramiento de la calidad de la imagen (Hitpoint Refinemet) y el rendimiento del sistema (Empty Space Skipping y Early Ray Termination).

CAPÍTULO 3. CARACTERÍSTICAS DEL SISTEMA

Durante este capítulo se describe el sistema desde la perspectiva de Ingeniería de Software, usando el Proceso Unificado de Desarrollo como metodología. Se presentan las reglas específicas del negocio y el modelo de dominio del problema. Posteriormente se muestran los requisitos funcionales y no funcionales detectados durante la Captura de Requisitos y el modelo de Casos de Uso del Sistema; dentro de este último, los Actores del Sistema, los Casos de Uso y sus respectivas descripciones. Del flujo de trabajo Diseño del Sistema se muestra el Diagrama de Clases del Diseño y los Diagramas de Secuencia correspondientes a los Casos de Uso.

3.1. Reglas del Negocio

Para la realización del presente trabajo se tuvieron en cuenta las siguientes reglas del negocio:

1. Las imágenes médicas que se deseen visualizar deben estar en el formato *.dcm o en el *.raw.
2. Las imágenes con el formato *.dcm deben tener la estructura original, las que no cumplan con este requisito no podrán ser visualizadas de forma correcta.
3. Las imágenes con el formato *.raw deben tener las dimensiones del estudio sin modificaciones. En caso de errores en las dimensiones originales, la visualización no será siempre correcta.

3.2. Modelo de Dominio

El modelo del dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software [25].

El entorno se puede describir cuando a un paciente se le realizan una serie de estudios por parte de los médicos, estos pueden ser a través de tomografías computarizadas (TAC) o resonancias magnéticas (MRI). Los estudios adquiridos empleando estas modalidades están formados por un conjunto de imágenes DICOM. El médico especialista analiza las imágenes a través de un software de visualización para realizar un procedimiento endoscópico virtual. En la Fig. 20 se muestra la descripción del negocio con el objetivo de facilitar la comprensión del funcionamiento del sistema.

El **Médico Especialista** es aquel que esté adecuadamente capacitado en el manejo de los métodos de imagenología diagnóstica y en los procedimientos endoscópicos.

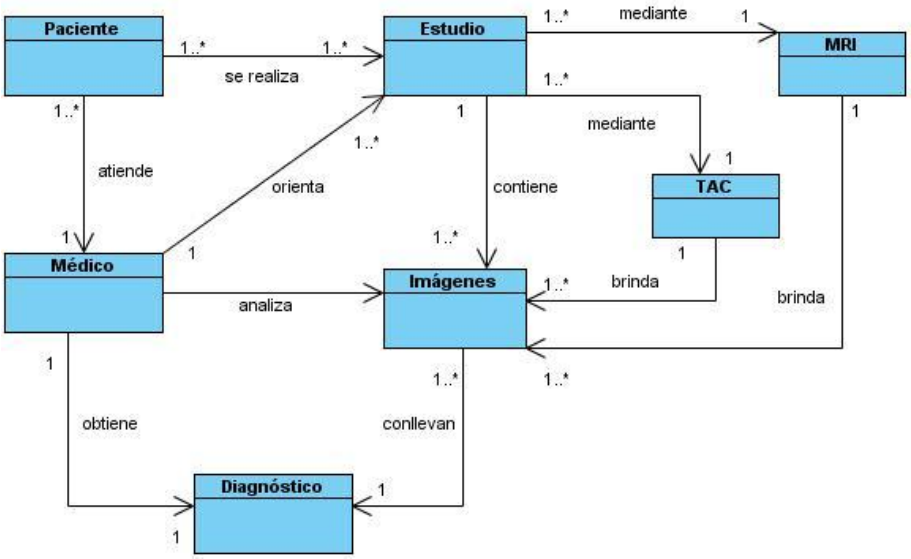


Fig. 20 Modelo de Dominio.

Las **MRI** son una modalidad de diagnóstico radiológico que utiliza tecnología de resonancia magnética nuclear en la que los núcleos magnéticos (especialmente los protones) del paciente se alinean en un campo magnético potente y uniforme, absorben energía de impulsos afinados de radiofrecuencia y emiten señales de radiofrecuencia a medida que decae su excitación. Una **TAC** es un procedimiento de diagnóstico médico que utiliza rayos X con un sistema informático que obtiene imágenes radiográficas en secciones progresivas de la zona del organismo estudiada.

Un **Estudio** es el examen orientado por el médico al paciente para obtener una patología y emitir un diagnóstico, este se encuentra compuesto por una serie de imágenes. Las **Imágenes** constituyen el resultado del estudio orientado por el médico al paciente a través del TAC y MRI. El **paciente** es aquel que recibe los servicios de un médico u otro profesional de la salud, sometiéndose a un examen.

El **Diagnóstico** es el resultado que emite el médico luego de realizar el estudio a un 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. Seguidamente se exponen los requisitos funcionales y no funcionales, por los cuales se rige el desarrollo del módulo.

3.3.1. Requisitos Funcionales

Los **requisitos funcionales** especifican acciones que debe poder realizar un sistema, sin tener en cuenta las restricciones físicas [25], además definen su comportamiento de salida y entrada. Además se mantienen invariables sin importar con que propiedades o cualidades se relacionen. A continuación se presentan los requerimientos funcionales tomados en cuenta durante el desarrollo del presente trabajo.

RF1. Cargar archivo.

- **RF1.1** Seleccionar archivo a cargar.
- **RF1.2** Cargar archivo.

RF2. Visualizar modelo tridimensional.

- **RF2.1** Generar la Textura 3D.
- **RF2.2** Mostrar el modelo tridimensional obtenido.

RF3. Realizar Endoscopia Virtual.

- **RF3.1** Mover cámara de forma Longitudinal.
- **RF3.2** Mover cámara de forma Vertical-Transversal.
- **RF3.3** Rotar cámara de forma Vertical-Transversal.
- **RF3.4** Rotar cámara de forma Longitudinal.

RF4. Configurar visualización.

- **RF4.1** Establecer Iso-valor.
- **RF4.2** Establecer la frecuencia de muestreo.
- **RF4.3** Establecer la calidad de la visualización.
- **RF4.4** Realizar cortes al volumen.

3.3.2. Requisitos no Funcionales

Los **requisitos no funcionales** sólo describen atributos del sistema o atributos del entorno del sistema [25]. Los requisitos no funcionales tomados en cuenta en el presente trabajo son los que se describen a continuación:

1. **De Software**

   Sistema Operativo Windows XP, Windows 7, Ubuntu 10.04 o superior.

2. **De Hardware**

   Microprocesador Intel Pentium IV a 3.0 GHz o superior.

   Memoria RAM de 1GB.

   Tarjeta Gráfica NVidia GeForce 9800 GT de 512 MB o superior.

3. **De Seguridad**

   Fiabilidad: Los modelos tridimensionales visualizados deben representar con alta precisión la anatomía real del paciente.

   Integridad: Los datos originales no deben sufrir pérdidas durante su representación.

4. **De Apariencia o Interfaz Gráfica de Usuario**

   La interfaz gráfica de usuario debe proporcionar, de forma coherente y sencilla, interactividad para todas las funcionalidades de la aplicación.

5. **De Soporte**

   Se brindará soporte para los sistemas operativos Windows XP, Windows 7 y Ubuntu 10.04 o superiores.

6. **De Restricciones en el Diseño e Implementación**

   Se empleará el lenguaje de programación C++ bajo el paradigma de Programación Orientada a Objetos.

3.4. Modelo de Casos de Uso

La forma en que los actores usan el sistema es representada a través de los casos de usos. Estos son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. En los siguientes epígrafes se muestran los actores y los casos de uso identificados en el presente trabajo.

3.4.1. Actores del Sistema

Los actores de un sistema son agentes externos, roles que las personas (usuarios) o dispositivos juegan cuando interactúan con el software. En este caso particular quien hará uso del sistema será el Médico Especialista con conocimientos sobre procedimientos endoscópicos.

Tabla 1 Actores del Sistema.

Actores	Justificación
Médico Especialista	Interactúa con el sistema durante la ejecución de sus funcionalidades. Carga las imágenes pertenecientes a un estudio realizado al paciente, ordenando al sistema visualizarlas de forma tridimensional para posteriormente realizar la endoscopia virtual.

3.4.2. Diagrama de Casos de Uso del Sistema

El Diagrama de Casos de Uso del Sistema (DCUS) representa gráficamente los casos de uso y su interacción con los actores (ver Fig. 21).

Los casos de uso tomados en cuenta en el presente trabajo están fuertemente relacionados, por ejemplo para la ejecución del Caso de Uso Realizar Endoscopia Virtual es necesaria la correcta visualización del modelo tridimensional (Caso de Uso Visualizar Modelo Tridimensional) y este último depende explícitamente del Caso de Uso Cargar Datos.

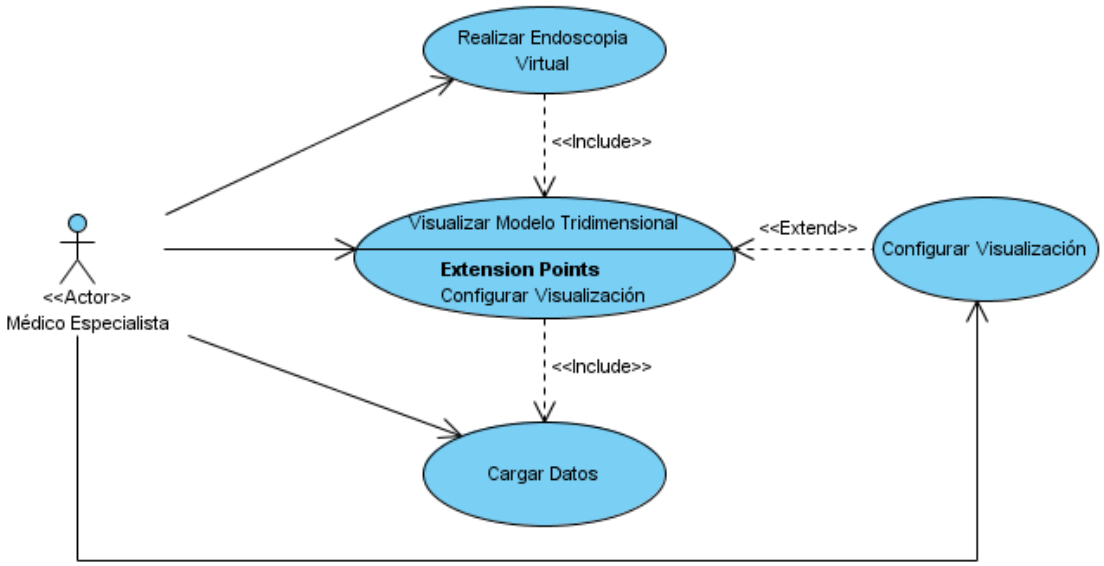


Fig. 21 Diagrama de Casos de Uso del Sistema.

Aunque el Caso de Uso Visualizar Modelo Tridimensional se puede ejecutar independientemente de Configurar Visualización, este último aporta al usuario las herramientas necesarias para corregir la

representación en función de la calidad y el rendimiento. Es necesario aclarar que el actor interactúa directamente con todos los casos de uso.

3.4.3. Descripción de los Casos de Uso del Sistema

Cada Caso de Uso posee una descripción de las acciones que realizará el sistema como respuesta a las peticiones del usuario. A continuación se relacionan las tablas correspondientes a las descripciones de los Casos de Uso detectados y se argumentan los flujos operacionales de cada uno, los casos de uso más complejos se dividen en secciones.

Tabla 2 Descripción del Caso de Uso Cargar Datos.

Caso de Uso:	Cargar Datos
Actores:	Médico
Propósito:	Importar los ficheros con extensión *.dcm o *.raw.
Resumen:	Se inicia cuando el actor selecciona la opción de cargar ficheros. Se selecciona el directorio donde están los ficheros DICOM o el fichero RAW deseado. Finaliza con la carga de todos los ficheros seleccionados para su posterior visualización.
Referencia:	RF1.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción de Cargar Ficheros.	1.1 Muestra un cuadro de diálogo que le permite al usuario seleccionar el o los ficheros que desea cargar. 1.2 El cuadro de diálogo solo mostrará los ficheros con las extensiones especificadas anteriormente.
2. Selecciona el directorio o los ficheros que desea cargar y acepta.	2.1 Se cierra el cuadro de diálogo y se procede a cargar los ficheros seleccionados.
Postcondiciones:	Se cargaron los ficheros seleccionados.
Prioridad:	Crítica.

Tabla 3 Descripción del Caso de Uso Visualizar Modelo Tridimensional.

Caso de Uso:	Visualizar Modelo Tridimensional
Actores:	Médico
Propósito:	Crear una representación visual con sensación de volumen a partir de los datos previamente cargados.
Resumen:	Se inicia luego de la carga de los ficheros seleccionados por el Médico. El sistema muestra una representación tridimensional de las estructuras anatómicas contenidas dentro de los ficheros seleccionados. Finaliza con el cierre del programa o la selección de un nuevo fichero.
Referencia:	RF1, RF2.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción de Cargar Ficheros.	1.1 Crea la textura tridimensional correspondiente al conjunto de datos cargado. 1.2 Realiza la representación de los datos empleando el algoritmo Raycasting Basado en GPU. 1.3 Muestra los resultados en pantalla.
Postcondiciones:	Se visualizó en pantalla la representación tridimensional del modelo cargado.
Prioridad:	Crítica.

Tabla 4 Descripción del Caso de Uso Realizar Endoscopia Virtual.

Caso de Uso:	Realizar Endoscopia Virtual
Actores:	Médico
Propósito:	Desplazar la cámara virtual por el interior del volumen para una mejor exploración de las estructuras anatómicas de interés.
Resumen:	Con el movimiento y el uso de los botones del mouse, el Médico manipula los movimientos permisibles de la cámara virtual.

Referencia:	RF3.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Utiliza alguno de los botones del mouse y los combina con movimientos (izquierda, derecha, arriba y abajo).	1.1 Realiza un conjunto de acciones que se detallan a continuación: a) Si el actor emplea el botón central del mouse: consultar Sección Mover Cámara. b) Si el actor sostiene el botón derecho del mouse y arrastra: consultar Sección Rotar Vertical-Transversal. c) Si el actor sostiene el botón izquierdo del mouse y arrastra: consultar Sección Rotar Longitudinal. 1.2 Se actualiza la escena, desplazando la cámara virtual según los eventos recibidos.
Postcondiciones:	Se representó la escena con la nueva posición y orientación.
Prioridad:	Crítica.

Tabla 5 Descripción de la Sección Mover Cámara, perteneciente al Caso de Uso Realizar Endoscopia Virtual.

Sección:	Mover Cámara
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Realiza las operaciones que permite el botón central del mouse.	1.1 Realiza un conjunto de acciones que se detallan a continuación: a) Si el actor rota el botón central (scroll): hace retroceder o avanzar la cámara según su eje longitudinal. b) Si el actor arrastra el mouse con el botón central oprimido: mueve la cámara por el plano que forman sus ejes Vertical y Transversal. O sea movimientos hacia la izquierda, derecha, arriba o abajo.

	1.2 Se actualiza la escena a partir de la nueva posición de la cámara.
Postcondiciones:	Se representó la escena con la nueva posición.
Prioridad:	Crítica.

Tabla 6 Descripción de la Sección Rotar Vertical-Transversal, perteneciente al Caso de Uso Realizar Endoscopia Virtual.

Sección:	Rotar Vertical-Transversal
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Sostiene el botón izquierdo del mouse y arrastra.	1.1 Recibe los eventos correspondientes y rota la cámara por sus ejes Vertical y Transversal. O sea movimientos hacia la izquierda, derecha, arriba o abajo. 1.2 Se actualiza la escena a partir de la nueva orientación de la cámara.
Postcondiciones:	Se representó la escena con la nueva orientación.
Prioridad:	Crítica.

Tabla 7 Descripción de la Sección Rotar Longitudinal, perteneciente al Caso de Uso Realizar Endoscopia Virtual.

Sección:	Rotar Longitudinal
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Sostiene el botón derecho del mouse y arrastra.	1.1 Recibe los eventos correspondientes y rota la cámara por su eje Longitudinal (hacia la izquierda o la derecha). 1.2 Se actualiza la escena a partir de la nueva orientación de la cámara.
Postcondiciones:	Se representó la escena con la nueva orientación.
Prioridad:	Crítica.

Tabla 8 Descripción del Caso de Uso Configurar Visualización.

Caso de Uso:	Configurar Visualización
Actores:	Médico
Propósito:	Configurar los parámetros de la visualización para obtener una imagen con el nivel de detalle deseado en correspondencia con el rendimiento.
Resumen:	Se inicia cuando el actor selecciona la opción Configurar Visualización. Se muestra la interfaz gráfica de usuario que permite configurar los componentes relacionados con la calidad de la representación y el rendimiento general del sistema. Finaliza con el cierre del formulario Configurar Visualización.
Referencia:	RF4.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción Configurar Visualización.	1.1 Muestra la ventana correspondiente a la opción seleccionada.
2. Interactúa con los componentes de la interfaz gráfica de usuario.	1.1 Realiza una de las siguientes acciones: a) Si el actor modifica el iso-valor: consultar Sección Establecer Iso-valor. b) Si el actor modifica la frecuencia para la toma de muestras: consultar Sección Establecer Frecuencia de Muestreo. c) Si el actor modifica el estado del algoritmo Hitpoint Refinement: consultar Sección Establecer estado del algoritmo Hitpoint Refinement. d) Si el actor modifica la Cantidad de Iteraciones del algoritmo Hitpoint Refinement: consultar Sección Establecer Cantidad de Iteraciones del algoritmo Hitpoint Refinement. e) Si el actor modifica el estado del algoritmo Empty Space Skipping: consultar Sección Establecer estado del algoritmo Empty Space

	Skipping. f) Si el actor modifica el valor de alguno de los planos de corte: consultar Sección Realizar Cortes al Volumen. 1.2 Actualiza la escena, creando nuevamente la representación con los nuevos valores.
Postcondiciones:	Se representó la escena con los parámetros actualizados.
Prioridad:	Crítica.

Tabla 9 Descripción de la Sección Establecer Iso-valor, perteneciente al Caso de Uso Configurar Visualización.

Sección:	Establecer Iso-valor
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Mediante la GUI cambia el Iso-valor.	1.1 Recibe los eventos correspondientes y cambia el Iso-valor. 1.2 Actualiza la escena con más o menos sustancia.
Postcondiciones:	Se actualizó la escena con el nuevo valor.
Prioridad:	Crítica.

Tabla 10 Descripción de la Sección Establecer Frecuencia de Muestreo, perteneciente al Caso de Uso Configurar Visualización.

Sección:	Establecer Frecuencia de Muestreo
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Mediante la GUI cambia el valor de la frecuencia para la toma de muestras.	1.1 Recibe los eventos correspondientes y cambia la Frecuencia de Muestreo. 1.2 Se actualiza la escena con mayor o menor detalle, dependiendo del valor introducido por el Médico.
Postcondiciones:	Se actualizó la escena con el nuevo valor.
Prioridad:	Crítica.

Tabla 11 Descripción de la Sección Establecer estado del algoritmo Hitpoint Refinement, perteneciente al Caso de Uso Configurar Visualización.

Sección:	Establecer estado del algoritmo Hitpoint Refinement
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Mediante la GUI cambia el estado del algoritmo Hitpoint Refinement.	1.1 Recibe los eventos y cambia el estado del algoritmo correspondiente. 1.2 Activa o desactiva la opción de modificar la cantidad de iteraciones del mismo algoritmo.
Postcondiciones:	Se actualizó la escena con el nuevo valor.
Prioridad:	Crítica.

Tabla 12 Descripción de la Sección Establecer Cantidad de Iteraciones del Hitpoint Refinement, perteneciente al Caso de Uso Configurar Visualización.

Sección:	Establecer Cantidad de Iteraciones del Hitpoint Refinement
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Mediante la GUI cambia la Cantidad de Iteraciones del Hitpoint Refinement.	1.1 Recibe los eventos y cambia la Cantidad de Iteraciones del Hitpoint Refinement. 1.2 Se actualiza la escena mejorando el detalle en la representación de la imagen o dejando el mismo según la opción seleccionada (Activar o Desactivar).
Postcondiciones:	Se actualizó la escena con el nuevo valor.
Prioridad:	Crítica.

Tabla 13 Descripción de la Sección Establecer estado del algoritmo Empty Space Skipping, perteneciente al Caso de Uso Configurar Visualización.

Sección:	Establecer estado del algoritmo Empty Space Skipping
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Mediante la GUI cambia el estado del algoritmo Empty Space Skipping.	1.1 Recibe los eventos y cambia el estado del algoritmo Empty Space Skipping. 1.2 Se actualiza la escena, si la elección fue Activar: se saltan los espacios en blanco que presente el volumen, mejorando el rendimiento del sistema, en otro caso, la escena se representa normalmente.
Postcondiciones:	Se actualizó la escena con el nuevo valor.
Prioridad:	Crítica.

Tabla 14 Descripción de la Sección Realizar cortes al volumen, perteneciente al Caso de Uso Configurar Visualización.

Sección:	Realizar cortes al volumen
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Mediante la GUI realiza cortes al volumen según los planos definidos con ese propósito.	1.1 Recibe los eventos y cambia los valores correspondientes a los planos de corte modificados. 1.2 Oculta o muestra el volumen acotado por los planos axiales, sagitales y coronales definidos.
Postcondiciones:	Se actualizó la escena con los cortes realizados.
Prioridad:	Crítica.

3.5. Diseño del Sistema

En el flujo de trabajo de Diseño la elaboración de los diagramas de clases de diseño juega un papel fundamental, estos muestran las clases finales para la realización de los casos de uso modelados con anterioridad.

Los diagramas de clases de diseño muestran las clases con sus atributos y métodos y la forma en que se relacionan entre sí. En este flujo también se realizan los diagramas de interacción que muestran la comunicación y las relaciones entre los objetos, con el objetivo de dar cumplimiento a los requerimientos.

En la Fig. 22 se muestra el Diagrama de Paquetes general del sistema. Para obtener un sistema modular, el mismo se ha dividido en tres paquetes fundamentales (Visualización, Función de Transferencia e Iluminación), el presente trabajo se centra en el paquete de **Visualización**.

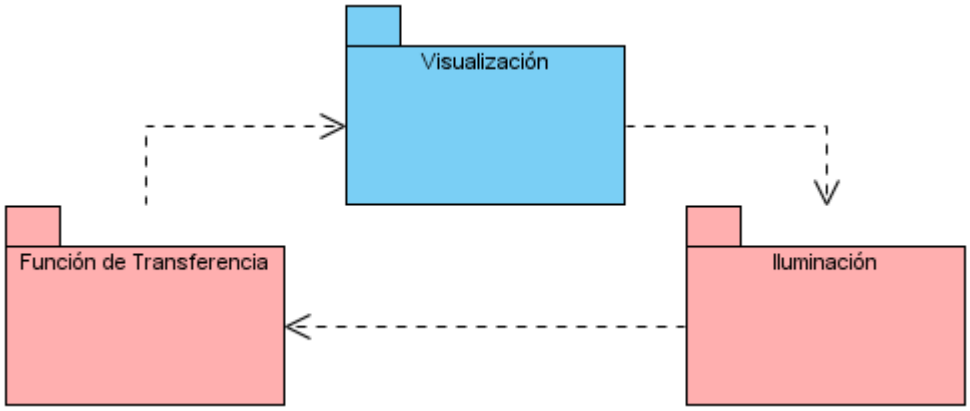


Fig. 22 Diagrama de Paquetes del Sistema.

3.5.1. Diagrama de Clases del Diseño

En la Fig. 23 se muestra el Diagrama de Clases del Diseño correspondiente al sistema desarrollado, nótese que la implementación del algoritmo propuesto se ha realizado principalmente dentro de la clase VolumeRender mientras que las funciones necesarias para el manejo de los movimientos de la escena se encuentran en la clase Camera. Las clases antecedidas por el prefijo ui, pertenecen a la interfaz gráfica de usuario y sólo se encargan de coordinar las acciones que realiza el actor y transmitirlas hacia el núcleo del sistema.

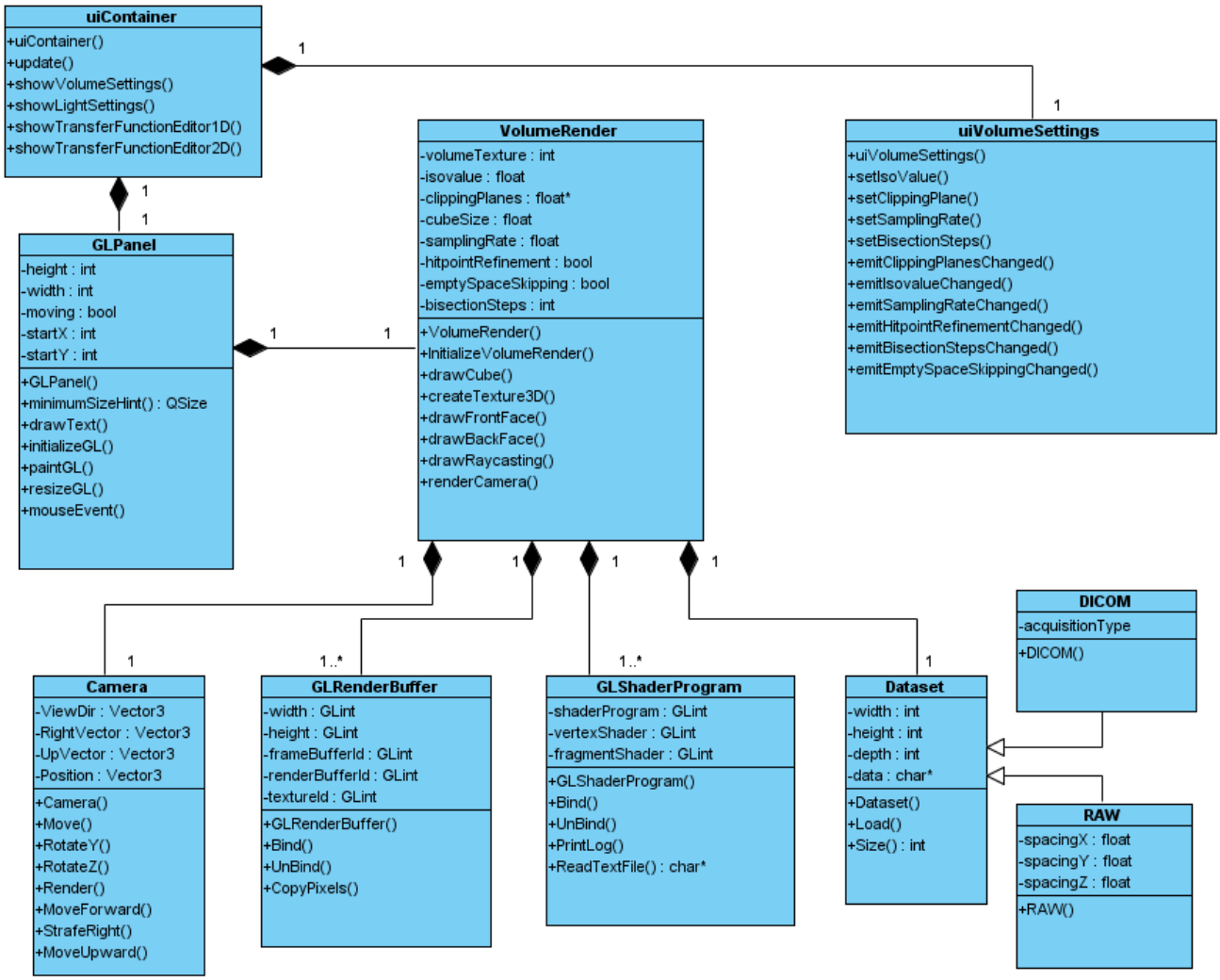


Fig. 23 Diagrama de Clases del Diseño del Paquete Visualización.

3.5.2. Diagramas de Secuencia del Diseño

A continuación se muestran los diagramas de secuencias empleados en el diseño del sistema.

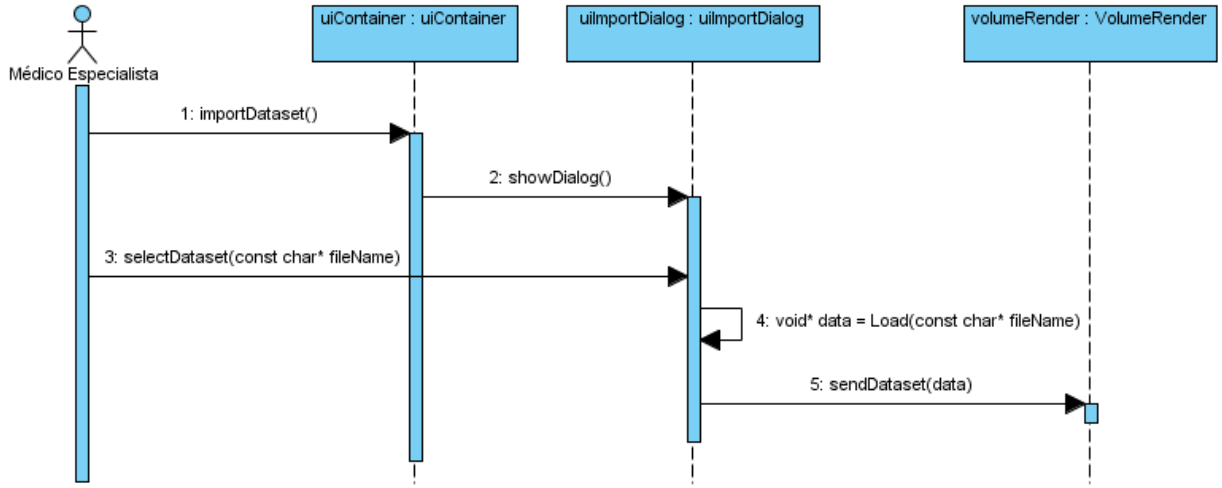


Fig. 24 Diagrama de Secuencia del Caso de Uso Cargar Datos.

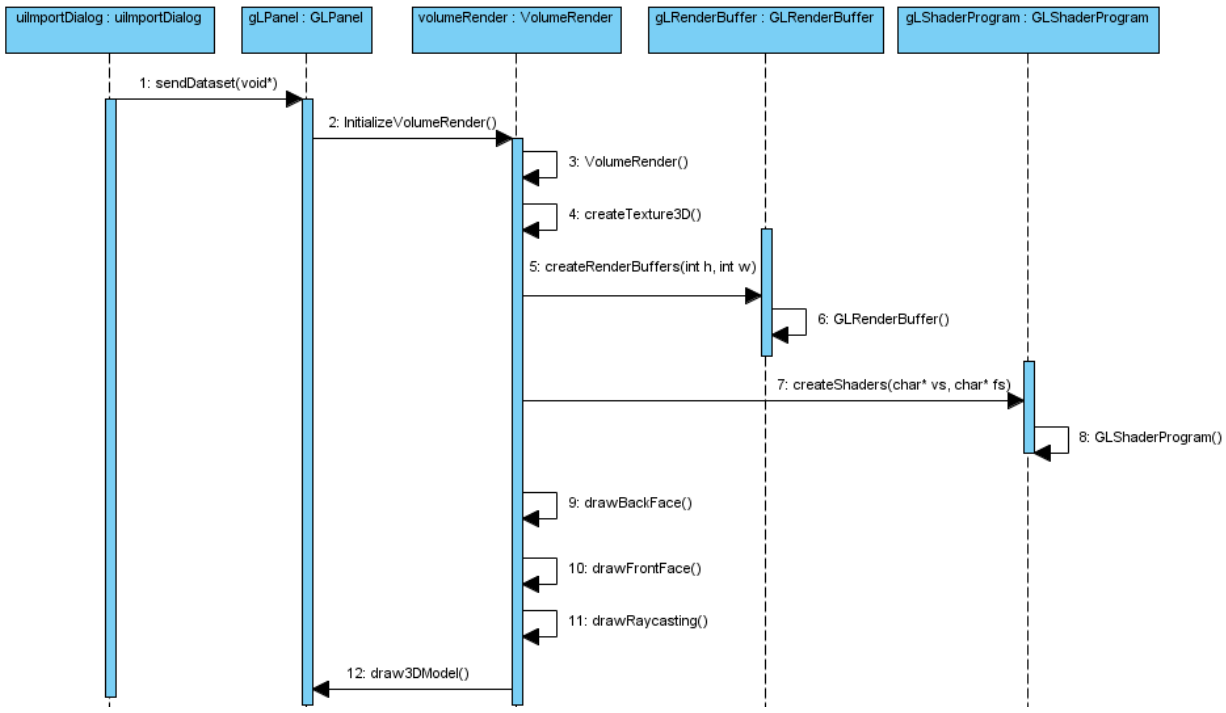


Fig. 25 Diagrama de Secuencia del Caso de Uso Visualizar Modelo Tridimensional.

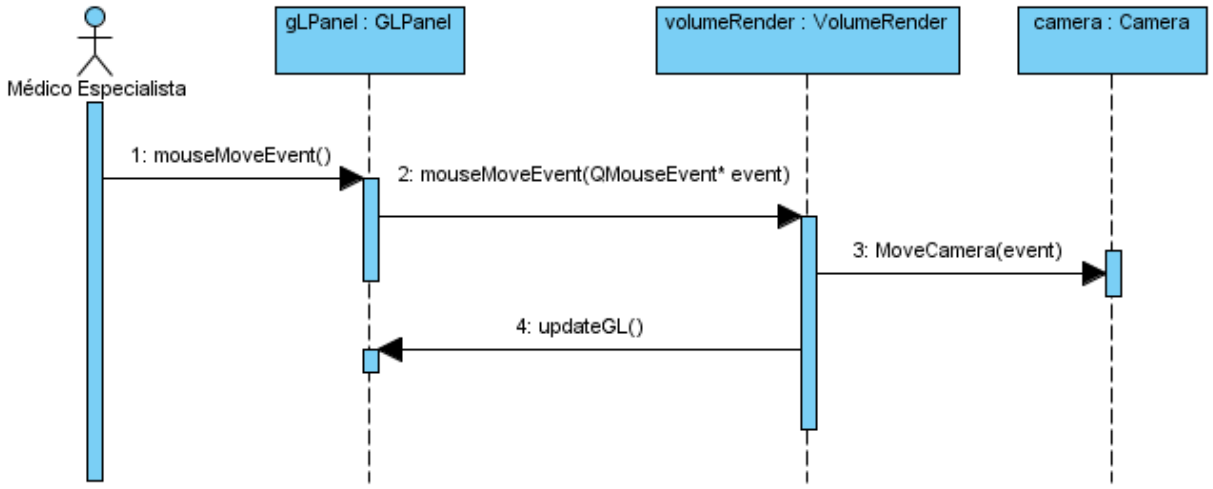


Fig. 26 Diagrama de Secuencia de la Sección Mover Cámara, CU Endoscopia Virtual.

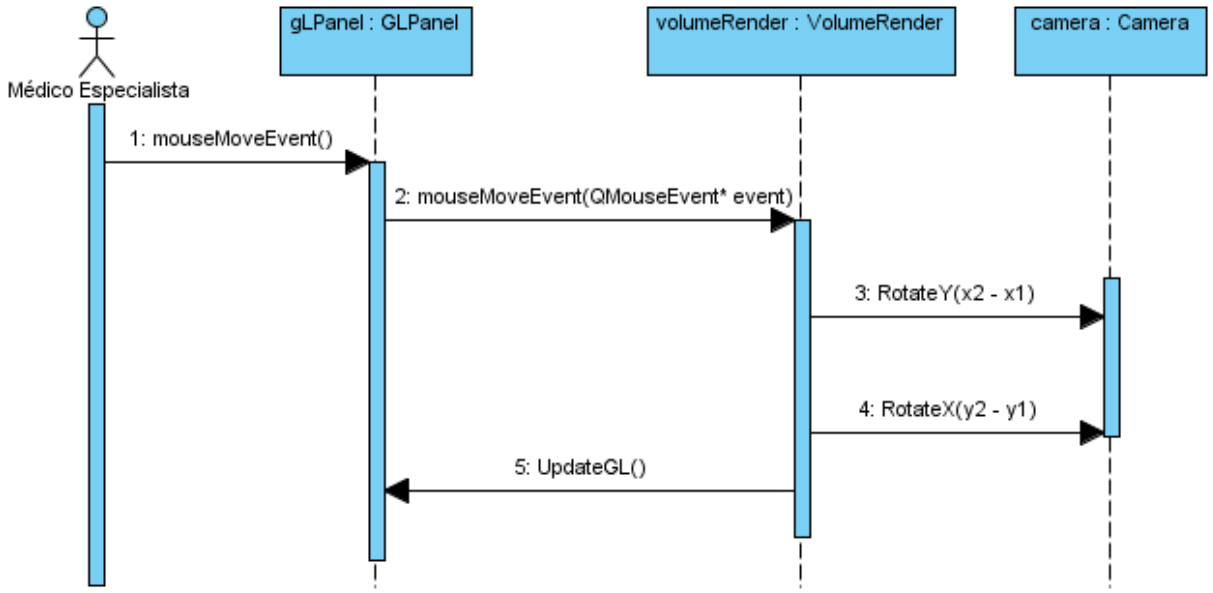


Fig. 27 Diagrama de Secuencia de la Sección Rotar Vertical-Transversal, CU Endoscopia Virtual.

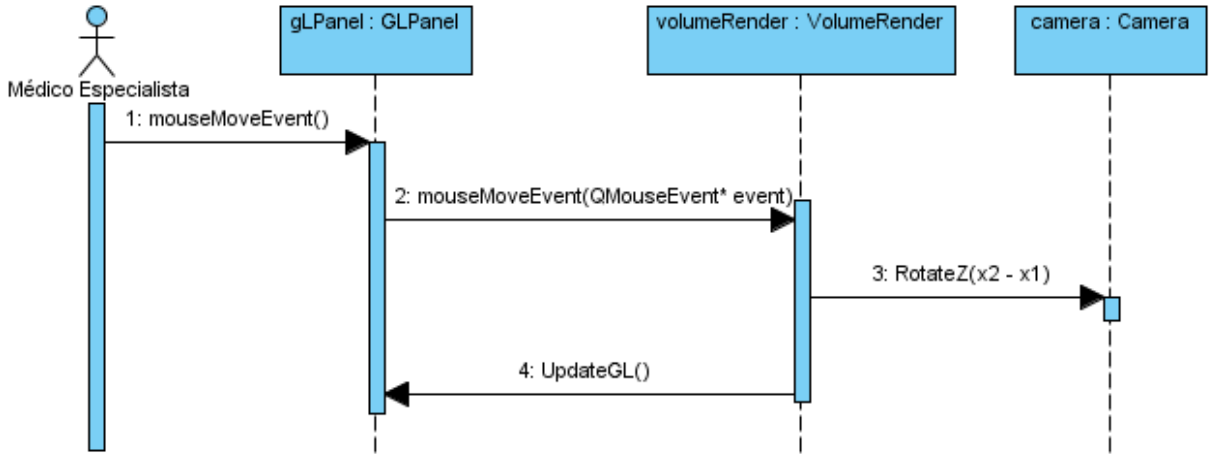


Fig. 28 Diagrama de Secuencia de la Sección Rotar Longitudinal, CU Endoscopia Virtual.

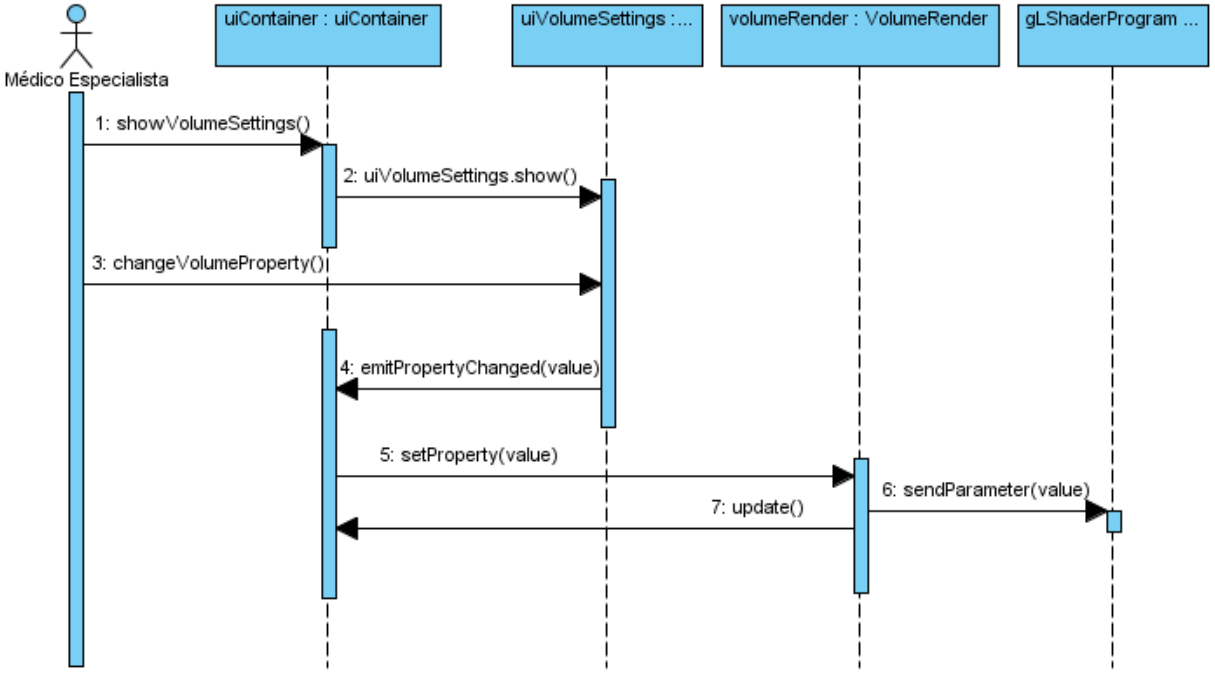


Fig. 29 Diagrama de Secuencia del Caso de Uso Configurar Visualización.

CAPÍTULO 4. IMPLEMENTACIÓN Y VALIDACIÓN DE LOS RESULTADOS

En este capítulo se abordan los temas relacionados con la implementación del sistema, el mismo se basa en el trabajo desarrollado en los capítulos anteriores, esta sección centra su contenido en el diagrama de componente del sistema desarrollado. Posteriormente se toman casos de prueba para validar los resultados alcanzados, fundamentalmente relacionados con el rendimiento del sistema y la calidad de la representación, estos se expresan respectivamente mediante tablas que contienen las variables fundamentales para futuras comparaciones con otros algoritmos y mediante imágenes comparativas, resultado de la aplicación de técnicas para el mejoramiento de la calidad de la imagen.

4.1. Implementación

El resultado principal de la implementación, es la obtención de componentes, dentro de los que se incluyen ficheros, ejecutables y las dependencias existentes entre estos. Además, este flujo especifica cómo van a estar ubicadas físicamente las distintas partes del sistema. En los casos necesarios se incluye también los protocolos que se emplearán para la comunicación entre los componentes.

4.1.1. Diagrama de componentes

Un componente representa una parte física del sistema, por ejemplo, una biblioteca de clases, un ejecutable, una tabla, etc., que engloba la implementación de un grupo de clases del diseño. Cada componente define una interfaz que describe su funcionalidad y forma de empleo. El diagrama de componentes, permite conocer a los desarrolladores y clientes la estructura física que tiene el sistema y cómo se relacionan sus partes. A continuación se presenta el diagrama de componentes del sistema propuesto (ver Fig. 30).

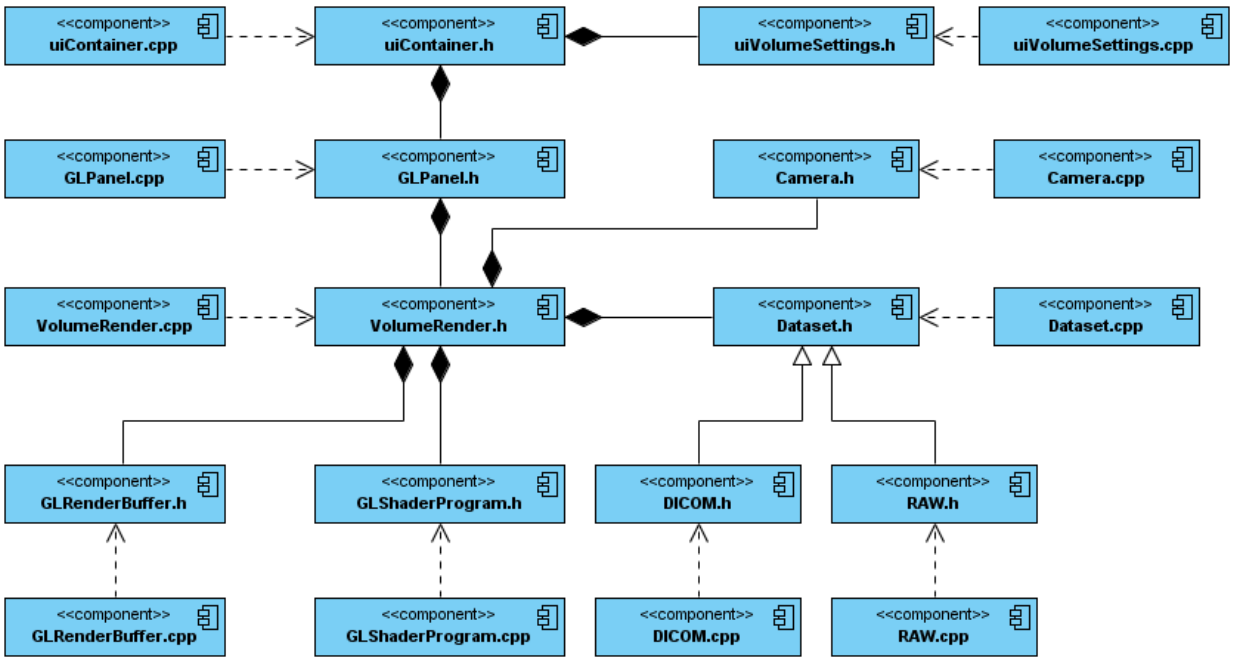


Fig. 30 Diagrama de Componentes.

4.2. Validación de los Resultados

Esta sección muestra los principales resultados alcanzados con el algoritmo propuesto y puede ser tomada como punto de partida para hacer comparaciones con los algoritmos similares existentes. Todas las pruebas se realizaron sobre una computadora personal con un procesador Intel Core2 Quad Q6600 a una frecuencia de 2.4 GHz, 1 GB de memoria RAM y tarjeta gráfica NVidia 9800 GT con 512 MB de RAM para video.

4.2.1. Datos de entrada

Para la realización de las pruebas se tomaron tres imágenes tridimensionales con diferentes dimensiones, cada una de estas guarda un valor de 8 bits para sus vóxeles. La resolución de la imagen que se representó en todos los casos fue de 600x600 píxeles y como distancia para la toma de muestras se estableció la longitud de un vóxel.

4.2.2. Parámetros a medir

Con el objetivo de evaluar los resultados del algoritmo propuesto se tendrán en cuenta dos criterios fundamentales: el rendimiento del sistema y la calidad de la representación.
El primer caso se expresa mediante la cantidad de cuadros que es capaz de representar en un segundo (fps) y por el consumo de memoria RAM. El segundo caso se evalúa por la precisión y realismo de la imagen final.
Los criterios escogidos se emplean en la validación de la mayoría de los sistemas dedicados a la visualización tridimensional en tiempo real. En [26] se establecen los puntos de referencia (benchmarks) para medir el comportamiento de los principales algoritmos actuales de Visualización Directa de Volumen.

4.2.3. Rendimiento

La siguiente tabla muestra el rendimiento normal del sistema y los resultados de la aplicación del algoritmo de optimización Empty Space Skipping para saltar los espacios no significativos dentro del volumen. La explicación de este algoritmo se encuentra en la sección 2.4.2 del presente trabajo.

Tabla 15 Rendimiento del algoritmo usando Empty Space Skipping.

Estudio	Dimensiones (X, Y, Z)	Empty Space Skipping Desactivado (fps)	Empty Space Skipping Activado (fps)	Consumo de Memoria RAM (MB)
Bonal	(512, 512, 346)	53	61	105.656
Cráneo	(256, 256, 256)	85	85	31.432
Molécula	(64, 64, 64)	85	85	11.056

Con volúmenes de datos relativamente pequeños, los resultados no son apreciables; sin embargo, cuando las dimensiones del volumen crecen la diferencia es bastante notable, lográndose en todos los casos tasas de refresco superiores a los 60 cuadros por segundo. El consumo de memoria RAM se mantiene constante con la optimización activada o desactivada.
La Tabla 16 muestra los efectos que produce el algoritmo Hitpoint Refinement en el rendimiento del sistema, aunque estos son claramente notables, aún se mantiene en tiempo real la visualización obtenida [26].

Tabla 16 Rendimiento del algoritmo usando Hitpoint Refinement.

Estudio	Dimensiones (X, Y, Z)	Hitpoint Refinement Desactivado (fps)	Hitpoint Refinement Activado (fps)	Consumo de Memoria RAM (MB)
Bonal	(512, 512, 346)	53	31	105.656
Cráneo	(256, 256, 256)	85	61	31.432
Molécula	(64, 64, 64)	85	75	11.056

4.2.4. Calidad de la imagen

La Fig. 31 muestra los resultados visuales aplicando el algoritmo Hitpoint Refinement usado para mejorar la calidad de la imagen. Las imágenes rotuladas con números impares muestran las representaciones con el algoritmo desactivado y las del rótulo par con el mismo activado. Se han señalado con elipses las partes donde mejor se aprecian los cambios que produce la activación del algoritmo:

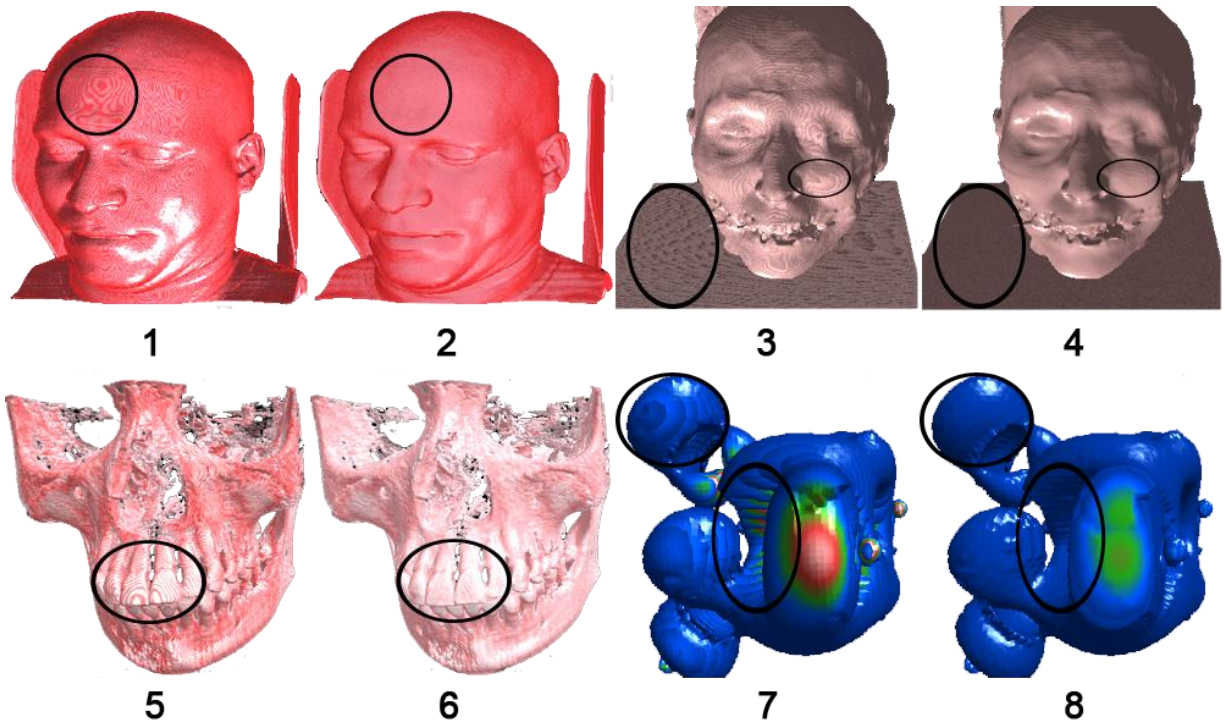


Fig. 31 Resultados de la aplicación del algoritmo Hitpoint Refinement.

Resulta lógico que una mejora significativa en la calidad de la imagen traiga como consecuencia un mayor costo computacional, producto a la toma de más muestras durante la propagación de los rayos, este impacto se muestra en la Tabla 16 de la sección anterior.

4.3. Integración

La arquitectura propuesta para el módulo de visualización permitió que se integrara de forma natural con la aplicación principal del proyecto Vismedic. Durante el proceso de integración solo fue necesario rediseñar las interfaces gráficas de usuario, conservando intacto el núcleo algorítmico del módulo elaborado.

Para la comunicación del módulo de Visualización con el resto de los módulos se definió la clase VolumeRender. Esta clase contiene las funciones CreateTransferFunction1D y CreateTransferFunction2D, que se encargan de crear la función de transferencia unidimensional y bidimensional respectivamente.

Para la integración con el módulo de Iluminación se hizo necesario crear en la clase VolumeRender las funciones AddLight y DestroyLight, estas adicionan y destruyen las luces respectivamente.

En el Anexo A se muestran las imágenes correspondientes a los resultados obtenidos con la aplicación principal del proyecto Vismedic luego de la integración.

CONCLUSIONES

Con la realización de este trabajo, se desarrolló un algoritmo que permite la visualización directa del volumen contenido en imágenes médicas digitales y la exploración de las estructuras anatómicas presentes en estas mediante endoscopias virtuales. De esta forma se da cumplimiento al objetivo propuesto al inicio de la investigación, además:

- Se adicionaron diferentes técnicas al algoritmo inicial que mejoraron el rendimiento del sistema y la calidad de la imagen representada, lográndose en todos los casos la ejecución en tiempo real.
- El algoritmo implementado se integró satisfactoriamente a la aplicación principal del proyecto Vismedic.
- El uso de endoscopias virtuales integradas al software Vismedic, permitió explorar con alto nivel de detalle el interior de los modelos tridimensionales empleados en el proyecto y mejoró considerablemente la percepción de la naturaleza volumétrica de estos.
- La ampliación del algoritmo para la intersección con geometría abrió las puertas para futuras aplicaciones endoscópicas más avanzadas y para su empleo en simuladores quirúrgicos.
- Se comprobó la eficiencia del hardware gráfico a la hora de realizar tareas paralelas. El empleo de modernos lenguajes de alto nivel para la programación en la GPU como GLSL, facilitó en gran medida la implementación del algoritmo propuesto.

RECOMENDACIONES

Al algoritmo propuesto en el presente trabajo se le pueden añadir numerosas mejoras de rendimiento y calidad de la imagen. Entre estas se pueden mencionar las siguientes:

- Incorporar un módulo para el pre-procesamiento y segmentación de las imágenes médicas, con el objetivo de disminuir su ruido e identificar con mayor detalle las estructuras anatómicas.
- Incorporar otras técnicas de optimización y manejo de memoria, usando estructuras de datos más apropiadas que permitan la representación de grandes volúmenes de datos.
- Agregar el cálculo de la ruta óptima para procedimientos endoscópicos virtuales, sentando las bases para su posterior uso en la planificación de endoscopias reales.
- Incorporar generación automática de materiales a partir de la intensidad en niveles de grises expresada en Unidades de Hounsfield.

BIBLIOGRAFÍA

1. Preim, Bernhard y Bartz, Dirk. Visualization in medicine, theory, algorithms and applications. Burlington : Elsevier Inc., 2007, 06.

2. Philips. Visualization Software.[En línea] [Citado el: 10 de 12 de 2010.] Philips. https://www.healthcare.philips.com/us_en/products/ct/products/Visualization _Software_Clinical_Applications/index.wpd.

3. SIEMENS. syngo fastView.[En línea] [Citado el: 10 de 12 de 2010.] SIEMENS. http://www.medical.siemens.com/webapp/wcs/stores/servlet/CategoryDisplay ~q_catalogId~e_-11~a_categoryId~e_1032039~a_catTree~e_100010,1008631,1029622,1017865, 1032039~a_langId~e_-11~a_storeId~e_10001.htm.

4. Voreen. Volume Rendering Engine. [En línea] [Citado el: 12 de 12 de 2010.] http://www.voreen.org/.

5. VRVis. Willkommen bei VRVis! . [En línea] [Citado el: 12 de 12 de 2010.] http://www.vrvis.at/.

6. Real Academia Española. Diccionario de la lengua española - Vigésima segunda edición. [En línea] [Citado el: 12 de 12 de 2010.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA= endoscopia.

7. Instituto Nacional del Cáncer en España. Definición de endoscopia - Diccionario del Cáncer. [En línea] [Citado el: 12 de 12 de 2010.] http://www.cancer.gov/diccionario/?CdrID=45678.

8. Free Online Medical Dictionary. Medical Dictionary. [En línea] [Citado el: 12 de 12 de 2010.] http://medical-dictionary.thefreedictionary.com/virtual+endoscopy.

9. Cantera Oceguera, Dra. Dolores T. Endoscopia y Realidad Virtual. [En línea] [Citado el: 12 de 12 de 2010.] http://fcmfajardo.sld.cu/jornada/conferencias/reumatologia/realidad_virtual.htm.

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

11. Bourke, Paul. Polygonising a scalar field. [En línea] [Citado el: 10 de 11 de 2010.] http://local.wasp.uwa.edu.au/~pbourke/geometry/polygonise/.

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

13. Chernyaev, Evgeni V. *Construction of Topologically Correct Isosurface.* s.l. : Institute for High Energy physics, 1995. Vol. 8.

14. Cullip, T. y Neumann, U. *Accelerating volume reconstruction with 3D texture hardware.* s.l. : Chapel Hill N.C, 1993.

15. Cabral, Brian, Cam, Nancy y Foran, Jim. *Accelerated volume rendering and tomographic reconstruction using texture mapping hardware.* 1995. págs. 91-98.

16. Wilson, Orion, VanGelder, Allen y Wilhelms, Jane. *Direct Volume Rendering Via 3D Textures.* Santa Cruz : University of California, 1994.

17. Van Gelder, Allen y Kim, Kuansik. *Direct volume rendering with shading via three-dimensional textures.* Santa Cruz: University of California, 1996.

18. Levoy, Marc. *Display of Surface from Volume Data.* Chapell Hill: s.n., 1988.

19. Brecheisen, Ralph. *Real-time volume rendering with hardware-accelerated raycasting.* 1996.

20. Max, Nelson. *Optical Models for Direct Volume Rendering.* California: University of California, 1995.

21. Engel, Klaus y Hadwiger, Markus. *Real-Time Volume Graphics.* 2006.

22. Rost, Randi. *OpenGL Shading Language.* s.l. : Addison Wesley Professional, 2006.

23. Scharsach, Henning. *Advanced Raycasting for Virtual Endoscopy on Consumer Graphics Hardware.* Vienna : s.n., 2005.

24. Booch, Grady, Jacobson, Ivar y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. 0-201-57169-2.

25. *Rational Unified Process.* s.l. : IBM Corporation, 2006.

26. Bartz, Dirk y et., al. *A Practical Evaluation of Popular Volume Rendering Algorithms.* Salt Lake City : s.n., 2000.

ANEXOS

ANEXO A: IMÁGENES DE LA APLICACIÓN

A continuación se muestran diferentes imágenes obtenidas con el algoritmo propuesto en esta investigación y su integración con la aplicación principal del proyecto Vismedic.

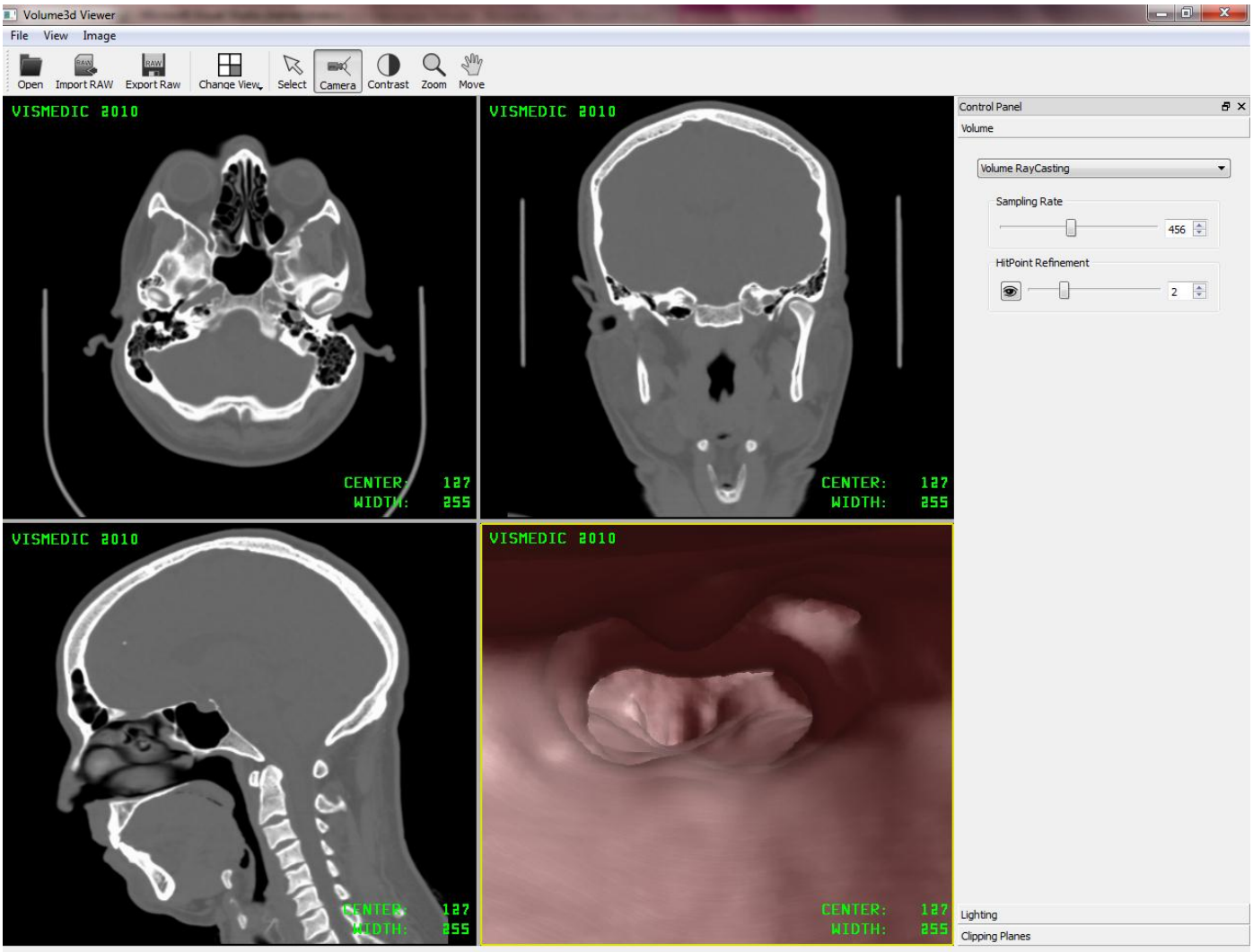


Fig. 32 Endoscopia Virtual en Vismedic empleando Raycasting.

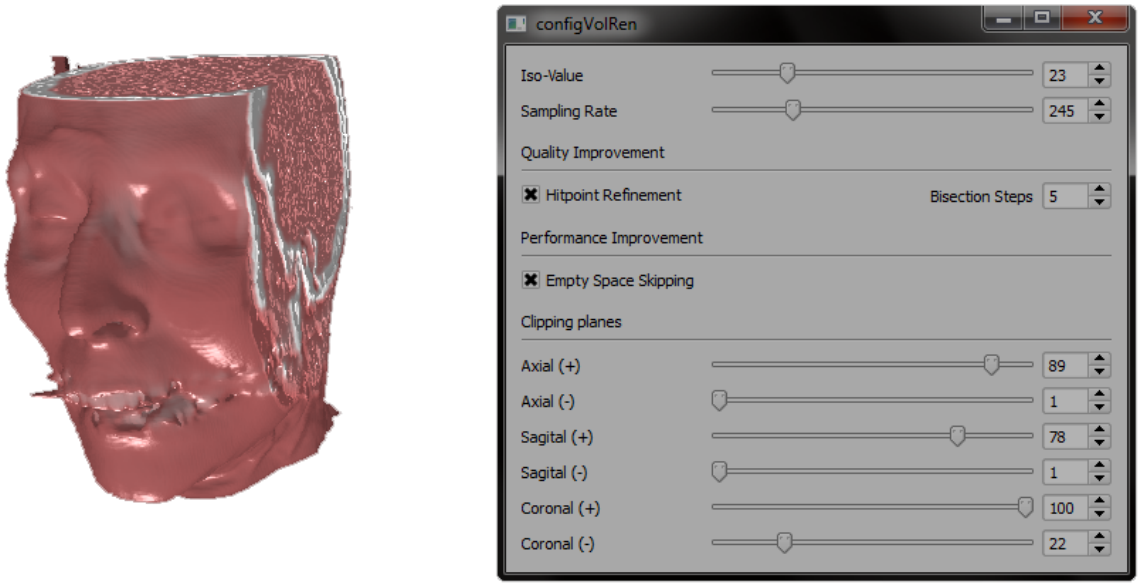


Fig. 33 Interfaz gráfica de usuario para configurar la visualización y los planos de corte.

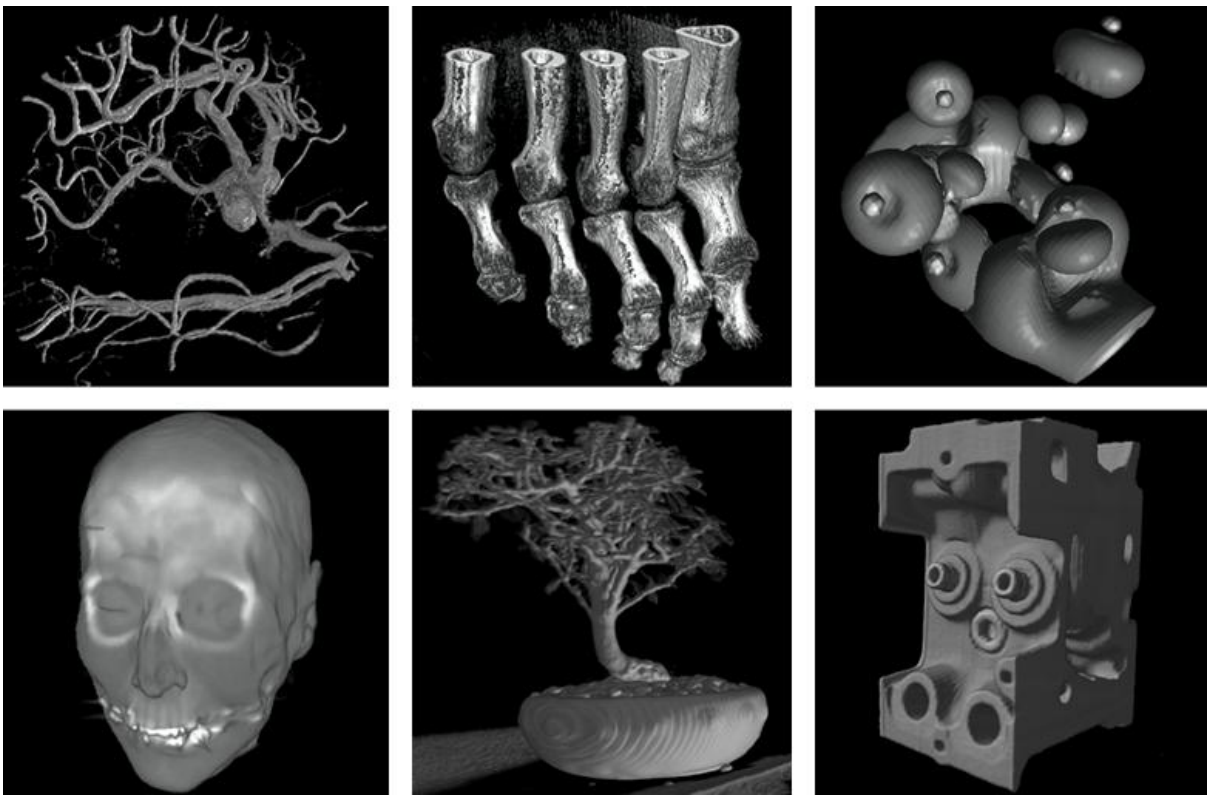


Fig. 34 Ejemplos de visualización de estructuras anatómicas y no anatómicas.

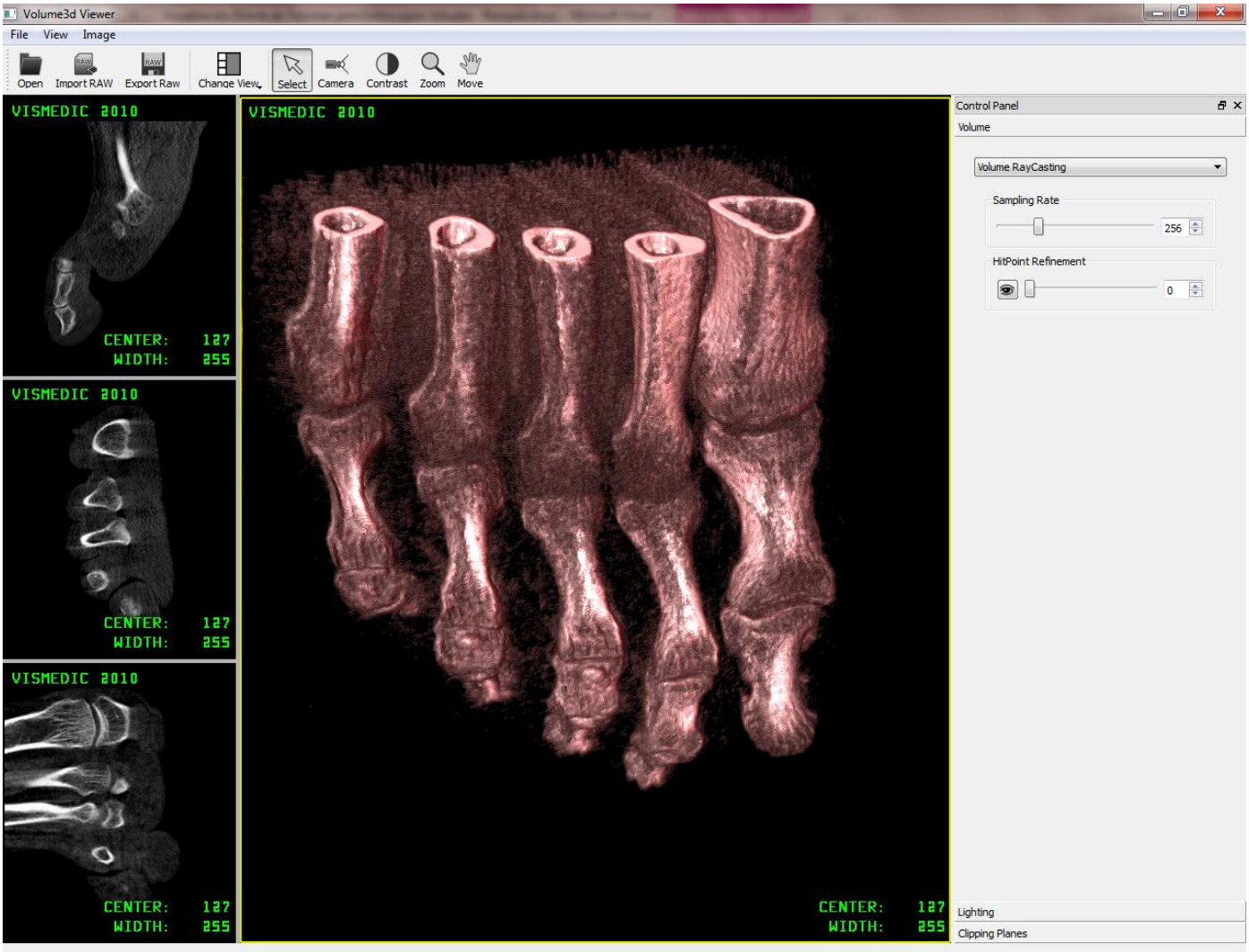


Fig. 35 Visualización de un pie con Hit Point Refinement desactivado.

GLOSARIO DE TÉRMINOS

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.

D

DICOM: En Inglés (Digital Imaging and Communication in Medicine) es el estándar reconocido mundialmente para el intercambio de imágenes médicas, pensado para el manejo, almacenamiento, impresión y transmisión de imágenes médicas. Incluye la definición de un formato de fichero y de un protocolo de comunicación de red. El protocolo de comunicación es un protocolo de aplicación que usa TCP/IP para la comunicación entre sistemas. Los ficheros DICOM pueden intercambiarse entre dos entidades que tengan capacidad de recibir imágenes y datos de pacientes en formato DICOM.

F

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

I

Imagenología: Comprende la realización de todo tipo de exámenes diagnósticos y terapéuticos en los cuales se utilizan equipos que reproducen imágenes del organismo. Los siete servicios de Imagenología son Ecotomografía, Imagenología Mamaria, Medicina Nuclear, Radiología, Rayos Infantil, Resonancia Magnética y Tomografía Computada o Scanner.

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

P

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

R

Realidad Virtual: Simulación generada por computadora de imágenes o ambientes tridimensionales interactivos con cierto grado de realismo físico o visual.

Resonancia Magnética: Modalidad de diagnóstico radiológico que utiliza tecnología de resonancia magnética nuclear en la que los núcleos magnéticos (especialmente los protones) del paciente se alinean en un campo magnético potente y uniforme, absorben energía de impulsos afinados de radiofrecuencia y emiten señales de radiofrecuencia a medida que decae su excitación. Estas señales, que varían en intensidad según la abundancia nuclear y el entorno químico molecular, se convierten en imágenes tomográficas (cortes seleccionados) mediante el uso de gradientes de campo en el campo magnético, lo que permite la localización tridimensional de las fuentes de las señales.

S

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

T

Tomografía Axial Computarizada (TAC): Es un examen médico no invasivo ni doloroso que ayuda al médico a diagnosticar y tratar enfermedades. Las imágenes por TAC utilizan un equipo de rayos X especial para producir múltiples imágenes o visualizaciones del interior del cuerpo, a la vez que utiliza conjuntamente una computadora que permite obtener imágenes transversales del área en estudio. Luego, las imágenes pueden imprimirse o examinarse en un monitor de computadora.

V

Vertex shader: Función del procesador 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 de las transformaciones sobre los vértices.

Vóxel: (la palabra proviene de la contracción del término en inglés "volumetric píxel") 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 (o pixel) en un objeto 2D.