UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

SISTEMA DE VISUALIZACIÓN MÉDICA TRIDIMENSIONAL SOBRE LA WEB.

TESIS EN OPCIÓN AL TÍTULO ACADÉMICO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Ernesto Gonzalez Martin

Tutor: MSc. Osvaldo Pereira Barzaga

Co-Tutor: Ing. Luis Guillermo Silva Rojas

Co-Tutor: Ing. Rubén Alcolea Núñez

Junio 2012

"Los que pueden, lo hacen; los que no, sólo saben quejarse."

Linus Torvalds

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco 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 2012.

Firma del Autor	**Firma del Tutor**
Ernesto Gonzalez Martin	MSc. Osvaldo Pereira Barzaga.

DATOS DE CONTACTO

Tutor: M.Sc. Osvaldo Pereira Barzaga.
Edad: 27.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: M.Sc. en Informática Aplicada.
Categoría Docente: Instructor.
Correo Electrónico: opereira@uci.cu

Co-Tutor: Ing. Luis Guillermo Silva Rojas.
Edad: 25.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ingeniero en Ciencias Informáticas.
Categoría Docente: Instructor Recién Graduado.
Correo Electrónico: lgsilva@uci.cu

Co-Tutor: Ing. Rubén Alcolea Núñez.
Edad: 25.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ingeniero en Ciencias Informáticas.
Categoría Docente: Instructor Recién Graduado.
Correo Electrónico: ralcolea@uci.cu

AGRADECIMIENTOS

Gracias:

A mi papá y mi mamá por ayudarme, apoyarme y estar al tanto de todo, en estos años de estudiante. De corazón gracias, los quiero mucho a los dos.

A mis hermanos Jose y Daniel, son los mejores hermanos del mundo.

A mis abuelos, maternos y paternos por ayudarme en todo lo que me hizo falta.

A toda mi familia GRACIAS.

Agradecer también:

A mis amigos de Contramaestre: Leonardo, Alberto, Yonnier y Risquiet.

A mis amigos que estuvieron en Venezuela y se graduaron junto conmigo.

A mis amigos que se graduaron en el 2011, que estuvieron al tanto de mi tesis.

A mi tutor Osvaldo por buscarme el tema de tesis más apropiado para mí, y ayudarme en todo lo que me hizo falta.

A los co-tutores Guille y Rubén también por ayudarme a realizar este trabajo.

Un agradecimiento especial:

A Julio por ser como otro hermano para mí y ayudarme a cumplir una de mis metas.

A Yausell y Yerandi por inculcarme todas sus experiencias.

En general:

A todas aquellas personas que tuve la dicha de ayudar en algún trabajo o proyecto, eso me ayudo a aprender algo nuevo todos los días.

DEDICATORIA

Dedico el resultado de este trabajo:

A toda mi familia, en especial a mi mama, mi papa, mis abuelos y mis hermanos por su apoyo constante durante estos años de la carrera.

A mis colegas Julio, Yausell y Yerandy que son como mis hermanos, por darme los consejos necesarios para ser un buen profesional.

A mis amigos de Contramaestre Leo, Yonnier, Risket y Alberto por siempre estar al tanto de mis resultados en la universidad.

RESUMEN

El constante desarrollo de las tecnologías de la información y el crecimiento que ha experimentado Internet en los últimos años, ha permitido que las tecnologías Web sean utilizadas en el desarrollo de las diferentes ramas de las ciencias. La medicina es una de las ramas que se ha visto favorecida con estos avances, un ejemplo fehaciente son las aplicaciones Web con entornos tridimensionales interactivos que permiten visualizar de una forma didáctica y muy precisa, imágenes y modelos de las estructuras anatómicas del cuerpo humano.

En el presente trabajo se propone realizar una aplicación Web que permita la visualización tridimensional de imágenes médicas obtenidas a partir de los dispositivos de exploración radiológica tales como Tomógrafos Computarizados y Resonadores Magnéticos. En la misma se implementa el algoritmo Raycasting basado en GPU utilizando la tecnología WebGL.

Palabras Claves: Entornos Tridimensionales, Imágenes Médicas, Raycasting, WebGL, Web.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con el desarrollo de las tecnologías de la información y el crecimiento de Internet como principal protagonista, ha propiciado que en los últimos años la información que reside en las páginas Web no solo sea representada a través de textos e imágenes en dos dimensiones. A partir de la décadas de los 90 las aplicaciones Web fueron tomando el espacio que antes ocupaban las aplicaciones de escritorio, los entornos tridimensionales (3D) no han sido la excepción, son muchos los software y lenguajes que se han desarrollado para lograr insertar en la Web los entornos 3D, que permitan al visitar un sitio Web, poder recorrer las instalaciones de un museo o navegar a través de la anatomía del cuerpo humano.

En la actualidad la visualización tridimensional en las aplicaciones Web es ya una realidad. Su uso en la medicina es un tema de interés constante, muchas de las aplicaciones desarrolladas con estas tecnologías tienen como objetivo principal mostrar de una forma didáctica y muy precisa, imágenes y modelos de las estructuras anatómicas del cuerpo humano, así estudiantes y especialistas pueden interactuar con dicha información y realizar diagnósticos más detallados.

En nuestro país la informatización del sistema de salud pública es una prioridad. En su desarrollo e implementación participan diferentes empresas y centros del Ministerio de la Informática y las Comunicaciones. La Universidad de las Ciencias Informáticas es uno de ellos, en ella existen varios centros productivos que se encargan de desarrollar soluciones para mejorar el sistema de salud, ejemplo de esto se encuentra el Centro de Informática Industrial, CEDIN por sus siglas en español. El mismo está compuesto por varios proyectos, uno de ellos es el de Visualización Científica que pertenece al área de Realidad Virtual. Este proyecto tiene a su cargo el desarrollo de software dirigidos al campo de la visualización médica, los cuales tienen como objetivo principal el diagnóstico quirúrgico y el entrenamiento de los especialistas en cirugías de mínimo acceso.

Dicho proyecto desarrolló una herramienta de visualización tridimensional, interactiva y en tiempo real, de imágenes DICOM y otros formatos utilizados para este mismo propósito como el RAW. La aplicación posee una versión de escritorio muy amigable y con un alto rendimiento gracias a las potencialidades del hardware gráfico actual, sin embargo en el proyecto no se cuenta con una aplicación Web que permita la visualización tridimensional de estas imágenes. Esto trae como consecuencia que diferentes usuarios no puedan acceder a la misma aplicación de manera simultánea desde diferentes estaciones de trabajo, lo que dificulta el trabajo colaborativo entre diferentes personas e instituciones.

Teniendo en cuenta la situación problémica anterior, se plantea como **problema científico:** ¿Cómo visualizar modelos anatómicos tridimensionales sobre plataforma Web a partir de imágenes DICOM?

De esta manera se define como **objeto de estudio** la visualización tridimensional de imágenes médicas.

Se plantea como **objetivo general**: Desarrollar una aplicación Web que permita visualizar modelos anatómicos tridimensionales obtenidos a partir de imágenes DICOM.

El **campo de acción** se enfoca en la visualización tridimensional de imágenes médicas sobre plataforma Web.

Como **idea a defender** se propone que: el desarrollo de la aplicación Web propuesta permitirá la visualización tridimensional de imágenes médicas, con la cual los usuarios podrán interactuar de forma simultánea o independiente desde diferentes ordenadores.

Para dar respuesta a los objetivos planteados se realizarán las siguientes **tareas de investigación**:

- Elaboración del marco teórico a partir del estado del arte actual referente al tema.
- Selección de la técnica de visualización de volumen más eficaz para el cumplimiento del objetivo propuesto.
- Selección de las herramientas y lenguajes de programación para el desarrollo del algoritmo.
- Implementación del algoritmo seleccionado.
- Validación de los resultados a través de pruebas de rendimiento.

Para darle cumplimiento a las tareas de investigación propuestas se hará uso de los métodos científicos de investigación:

Método teórico:

- **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 sobre la Web.

- **Modelación:** Mediante este método se representará toda la información obtenida hasta el momento mediante diagramas, ya sea de clases o de diseño, los cuales permitirán reflejar las relaciones y cualidades de la aplicación Web a desarrollar.

Métodos Empíricos:

- **Entrevista:** Es una conversación planificada para obtener información, mediante este método se consultará a diferentes personas del área de Realidad Virtual de la facultad 5 que ya hayan trabajado en el tema.

- **Observación:** Este método será aplicado en distintos momentos del proceso de investigación con el objetivo de evaluar el impacto que tienen las distintas técnicas de visualización 3D sobre la Web en el rendimiento de la computadora, con lo cual se podrá seleccionar en cada momento la más adecuada para emplear en la aplicación.

- **Pruebas:** Se realizarán diferentes pruebas al algoritmo implementado para determinar si se comporta según los resultados esperados.

A continuación se muestra la estructura de cada uno de los capítulos para conocer brevemente el contenido de este trabajo.

Capítulo 1: "Fundamentación teórica", en este capítulo se realiza un estudio de los aspectos relacionados con la visualización tridimensional en la Web, sus orígenes y evolución. Se hace referencia a los campos de aplicación en el cual está enmarcado la visualización tridimensional en la Web. Se abunda en el concepto de Visualización Médica y los tipos de aplicaciones que se desarrollan utilizando esta tecnología. Se presenta la descripción del estado de arte con las soluciones que existen alrededor del objeto de estudio, estableciendo comparaciones, analizando sus características, ventajas y desventajas.

Capítulo 2: "Solución propuesta", en este capítulo se realizará la descripción de la solución propuesta para dar respuesta a la situación problémica planteada. Primeramente se exponen algunos de los principales algoritmos más utilizados en la visualización de volumen. Acto seguido se explica todo el funcionamiento del algoritmo utilizado en el desarrollo de la aplicación, el Raycasting, desde la perspectiva de la Web utilizando la tecnología WebGL, se caracterizan cada uno de sus pasos y se proporcionan detalles de su implementación. Por último se mencionan las metodologías y herramientas de desarrollo empleadas en la realización de este trabajo.

Capítulo 3: "Características del sistema", en este capítulo se hace énfasis en las características de la solución propuesta donde se definen las reglas específicas del negocio y el modelo de dominio del problema. Se muestran los requisitos funcionales y no funcionales detectados durante la Captura de Requisitos y el modelo de Casos de Uso del Sistema. Dentro del modelo de Casos de Uso del Sistema se

muestran 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.

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 y de despliegue del sistema desarrollado. Posteriormente se toman casos de prueba para validar los resultados alcanzados, fundamentalmente relacionados con el rendimiento del sistema.

CAPÍTULO 1. Fundamentación Teórica.

1.1. Introducción.

En este capítulo se realiza un estudio de los aspectos relacionados con la visualización tridimensional en la Web, sus orígenes y evolución. Se hace referencia a los campos de aplicación en el cual está enmarcado la visualización tridimensional en la Web. Se abunda en el concepto de Visualización Médica y los tipos de aplicaciones que se desarrollan utilizando esta tecnología. Se presenta la descripción del estado de arte con las soluciones que existen alrededor del objeto de estudio, estableciendo comparaciones, analizando sus características, ventajas y desventajas.

1.2. La visualización tridimensional en la Web, sus orígenes y evolución.

1.2.1. Surgimiento de la Web

La *World Wide Web* o WWW o W3 o simplemente Web, es la tecnología que permite a los usuarios acceder a la inmensidad de los recursos de Internet. Es la forma más moderna de ofrecer información, el medio más potente. La información se ofrece en forma de páginas electrónicas. Permite saltar de un lugar a otro en pos de lo que interesa, lo más interesante es que con unas pocas órdenes se puede mover por toda la Internet.

La Web es una idea que se construyó sobre la Internet, empezó a principios de 1990 en Suiza, en el Centro de Estudios para la Investigación Nuclear (CERN) y la idea fue de *Tim Berners-Lee*[1], que se gestó observando una libreta que él usaba para añadir y mantener referencias de cómo funcionaban los ordenadores en el CERN.

Antes de la Web, la manera de obtener los datos por la Internet era caótica: había un sinfín de maneras posibles y con ello había que conocer múltiples programas y sistemas operativos. La Web introduce un concepto fundamental: la posibilidad de lectura universal, que consiste en que una vez que la información esté disponible, se pueda acceder a ella desde cualquier ordenador, desde cualquier país, por cualquier persona autorizada, usando un único y simple programa [1].

Ya han pasado más de dos décadas desde la creación de la Web, dentro de ese período los cambios que ha sufrido son considerables. Para poder caracterizar las diferentes evoluciones que ha tenido la Web se le

[1] Sir Timothy "Tim" John Berners-Lee, nació el 8 de junio de 1955 en Londres, Reino Unido, se licenció en Física en 1976 en el Queen's College de la Universidad de Oxford. Es considerado el padre de la Web.

ha denominado como Web 1.0, Web 2.0, Web 3.0 y ya se habla de la Web 4.0. En el siguiente epígrafe se aborda más sobre este tema.

1.2.2. Evolución de la Web

La Web no siempre ha tenido el mismo aspecto que hoy podemos observar, muchas tecnologías se popularizaron en ciertas etapas de la historia, que luego fueron remplazadas por otras más avanzadas y acordes a las necesidades que se planteaban los desarrolladores y usuarios.
Los términos Web 1.0, Web 2.0 y Web 3.0 han servido para caracterizar las diferentes etapas por las que ha ido evolucionando la Web.

⇒ **Web 1.0**

La Web 1.0 o también conocida como “Internet Básica” [2] empezó en los años 60 junto con la Internet, de la forma más básica que existe, con navegadores de solo texto, como el ELISA, bastante rápido pero muy simple. Después en los 90 surgió HTML (Hyper Text Markup Language) como lenguaje hipertexto e hizo que las páginas Web sean más agradables a la vista y puedan contener componentes como imágenes, formatos y colores [3]. El uso de la Web 1.0 era muy limitado, estaba dirigida fundamentalmente para publicar documentos y realizar transacciones. Con ella, las grandes empresas inauguraron su estrategia online. Crearon un sitio donde publicar información corporativa, y desarrollaron planes de marketing y ventas que incorporaban la Web como nexo con los clientes [3].
Esta Web era de solo lectura, el usuario no podía interactuar con el contenido de la página, es decir no podía realizar ningún comentario o dar respuesta a cualquier artículo publicado en el sitio Web, estando la información totalmente limitada a lo que el webmaster publicara.

⇒ **Web 2.0**

La Web 2.0 es un concepto que se originó en una sesión de *brainstorming* (o lluvia de ideas) entre Dale Dougherty de O’Really y Craig Cline, quienes estaban preparando una conferencia y decidieron hablar del renacimiento de la Web. A partir de estas ideas, en una conferencia en octubre de 2004, caracterizan a la Web 2.0 como una nueva actitud o evolución de la Internet, que se resumía en tres principios básicos: la Web como plataforma, la inteligencia colectiva y la arquitectura de la participación.
Dicho de otro modo, este término propuesto por Tim O'Reilly se refiere a una segunda generación en la historia del desarrollo de tecnología Web, basada en comunidades de usuarios y en una amplia gama de herramientas, como las redes sociales, blogs, wikis y otros, que fomentan la colaboración y el intercambio

de información. La Web 2.0 es también llamada Web social por el enfoque colaborativo y de construcción social de esta herramienta, donde el usuario es el ente más importante de información [4].

⇒ **Web 3.0 o Web semántica**

La Web 3.0 también conocida como Web semántica es un término que acuñó Tim Berners Lee, creador de la World Wide Web para referirse a un medio cibernético capaz de interpretar e interconectar un número mayor de datos, lo que permitiría el avance en el campo del conocimiento [5]. Otros escritos plantean que la Web 3.0, con lenguajes más potentes, redes neurales, algoritmos genéticos, pondrán el énfasis en el análisis y la capacidad de procesamiento, y en cómo generar nuevas ideas a partir de la información producida por los usuarios [3].

1.2.3. Entornos 3D en la Web

Los entornos 3D en la Web siempre han sido algo así como un sueño perenne desde la creación de la Internet. A partir de los años 90 se comenzaron a realizar varios intentos por conseguir integrar la "realidad virtual" en los navegadores Web, pero debido a cierto inmovilismo por parte de los desarrolladores de los principales navegadores de ese período tales como Netscape y Microsoft básicamente, la falta de potencia en los ordenadores y el poco ancho de banda en las conexiones, condujeron a que estos intentos fracasaran.

⇒ **Virtual Reality Modeling Language (VRML)**

Una de las primeras formas y quizás la más conocida que se utilizó para integrar el 3D en la Web fue *Virtual Reality Modeling Language* (VRML), un formato de fichero que permitía almacenar modelos y texturas en 3D, pensado para su uso en la Web, de forma que se asociasen objetos de una escena a hipervínculos, creando así cierta interactividad en un mundo 3D.
La primera versión de VRML fue especificada en noviembre de 1994, aunque no fue hasta 1997 cuando se presentó la versión VRML97 ó 2.0, que es la que alcanzaría relativa popularidad. Tuvo bastante éxito en el campo educacional y de investigación, aunque no llegó a utilizarse por el público mayoritario. En la época de apogeo de VRML, la mayoría de los usuarios contaba solamente con conexión por módem, lo cual provocaba que fuese necesaria una carga de varios minutos para poder visualizar, en muchos de los casos, un modelo de calidad bastante reducida.
Por si fuera poco, una de las empresas importantes detrás de VRML, Silicon Graphics, detuvo en 1998 todo nuevo desarrollo sobre la tecnología. Todos estos inconvenientes provocaron que VRML prácticamente cayera en el olvido recién comenzado el nuevo siglo [6].

⇒ **X3D**

A partir de los múltiples inconvenientes que presentaba el formato VRML, se comenzaron a buscar otras soluciones para lograr integrar el 3D en la Web, X3D (Extensible Tridimensional) fue una de ellas, es un estándar abierto XML que tiene un formato de archivo 3D que permite la creación y transmisión de datos 3D entre distintas aplicaciones y especialmente aplicaciones en red.
Para la visualización de escenas X3D se emplean los X3D *browsers* que son aplicaciones de software que pueden leer o procesar escenas X3D, no solo permiten mostrar a los objetos desde distintos puntos de vista sino que también permiten la animación de los mismos y la interacción con el usuario. Estos se suelen aplicar como *plugins* que funcionan como una parte integrada del lenguaje HTML que interpreta el navegador Web.

⇒ **Java3D y Flash3D**

Al mismo tiempo que VRML y X3D continuaban con su desarrollo para mejorar su rendimiento en los navegadores, otras compañías de software desarrollaban sus soluciones, basadas en las tecnologías o lenguajes creados por las mismas. Java3D una API (*Application Programming Interface*) orientado a objetos para el lenguaje Java, fue desarrollado por Intel, Silicon Graphics, Apple y Sun en el año 1997. Es una interfaz de programación utilizada para realizar aplicaciones y *applets* con gráficos en tres dimensiones. Proporciona a los desarrolladores un alto nivel para crear y manipular objetos geométricos 3D y para construir las estructuras utilizadas en el renderizado de dichos objetos [7].
A finales de 2005, entró en vigor Sandy 3D, el primer motor 3D para Flash escrito en ActionScript 2.0, el lenguaje de scripts para esta plataforma en ese período. Debido a las capacidades limitadas del procesador de gráficos en el intérprete Flash Player, los resultados que se obtenían eran bastante básicos, sin embargo, la liberación de Sandy 3D fue el punto de partida para la creación de los futuros motores 3D para Flash. El desarrollo del nuevo lenguaje ActionScript 3.0 incluido en la versión 9.0 del intérprete Flash Player, hizo posible que se crearan motores 3D más eficientes, capaces de lograr una nueva experiencia en el campo de la visualización 3D sobre la Web. Después de la salida de la actualización de Sandy 3D en el 2006, nació Papervision3D, Away3D [7] y otros motores gráficos como Alternativa3D, todos ellos capaces de generar resultados muy convincentes y de gran calidad visual dentro del navegador Web. Actualmente Adobe Systems Incorporated, la compañía encargada del desarrollo de la plataforma Flash, ha incorporado en la última versión del intérprete Flash Player una nueva API de bajo nivel llamada Molehill, capaz de trabajar con elementos 3D en Flash utilizando la aceleración de hardware proporcionada por las unidades de procesamiento gráfico (GPU).

⇒ **WebGL**

Sin embargo, el futuro de Flash, Java 3D y X3D no está muy claro, pues estos padecen el mismo problema que impidió la popularización de VRML: es necesario un *plugin* para interpretarlo. Por tanto, y debido a la arquitectura de *plugins* de los navegadores Web, esto lo limitaba a ser un rectángulo dentro de una página Web.

Para superar estas limitaciones, a principio del 2009 Mozilla y Khronos comenzaron el WebGL Working Group (Grupo de Trabajo de WebGL). Este grupo de trabajo compuesto también por Apple, Google y Opera lanzaron en marzo del 2011, WebGL 1.0, una especificación estándar y abierta que permite utilizar la aceleración del hardware gráfico para la visualización de gráficos 3D, en tiempo real y en el mismo navegador Web sin la necesidad de *plugins*, utilizando la etiqueta <canvas> de HTML 5 [8]. En la actualidad los navegadores Web: Safari (Apple), Google Chrome (Google), Mozilla Firefox (Mozilla) y Opera (Opera), desarrollados por las empresas miembros del Grupo de Trabajo de WebGL, traen esta tecnología de forma nativa lo que hace posible que aparezcan aplicaciones de calidad gráfica similar a videojuegos o aplicaciones de visualización científica complejas.

1.3. Uso de la visualización 3D sobre la Web.

La aplicación de la visualización 3D sobre la Web cada día se hace más amplia, varios son los campos que se han favorecido con el auge de esta nueva tecnología. El entretenimiento, la educación o la visualización científica son los que más se destacan, es extensa la gama de aplicaciones que se han creado por los desarrolladores, desde mapas 3D hasta paseos virtuales interactivos.

A continuación se exponen algunos ejemplos de aplicaciones que integran entornos 3D interactivos.

⇒ **Mapas**

MapsGL: Google siempre se ha destacado por estar siempre en la vanguardia, en lo que se refiere al desarrollo de aplicaciones Web. Este grande de la informática ha desarrollado una característica denominada MapsGL la que se apoya en WebGL para ofrecer vistas 3D en *Google Maps*[2]. Si se habilita esta característica, el usuario se puede acercar al objetivo y desplazarse por el entorno 3D apreciando una vista panorámica del paisaje elegido.

[2] Google Maps es un servicio de Google que ofrece imágenes vía satélite de todo el planeta, combinadas, en el caso de algunos países, con mapas de sus ciudades, lo que unido a sus posibilidades de programación abierta ha dado lugar a diversas utilidades ofrecidas desde numerosas páginas Web. (15)

⇒ **Paseos Virtuales**

La Capilla Sixtina en 3D: Este es un paseo virtual de la Capilla Sixtina (Vaticano) que ofrece un escenario 3D renderizado con flash en alta resolución, las escenas son bastante vividas y nítidas y permiten hacer zoom. Es una experiencia agradable visitar de forma virtual esta joya pictórica y arquitectónica [9].

1.4. Visualización Médica.

La Visualización Médica o también conocida como Visualización en la medicina es un área especial de la visualización científica que se estableció como un área de investigación a finales de 1980 [10]. La disciplina de la visualización médica surge como una necesidad para aprovechar y asimilar la enorme cantidad de datos producidos por los modernos sistemas de imágenes médicas [11]. Algunos autores definen a la visualización como el proceso de explorar, transformar y mostrar datos en forma de imágenes para entender y evaluar adecuadamente las características de los mismos, considerando así, que la visualización encuentra en la medicina un campo de aplicaciones muy adecuado.
Los objetivos y escenarios de investigación de la visualización científica son los siguientes:

- Explorar los datos de búsqueda (sin dirección, sin una hipótesis específica)
- Para poner a prueba una hipótesis basada en las mediciones o simulaciones y su visualización
- La presentación de los resultados

Todos estos escenarios se ponen de manifiesto en el área de la visualización médica. Un ejemplo relevante es cuando un especialista tiene al frente un paciente que está sufriendo una determinada enfermedad, lo primero que debe realizar es una exploración de los datos de búsqueda apoyándose en las investigaciones clínicas e imágenes médicas, los equipos de procesamiento de imagen en particular ayudan a mejorar el diagnóstico. Seguidamente el especialista cuando haya realizado el diagnóstico que especifica el escenario y la gravedad de la enfermedad, se lo presenta al médico remitente, estas visualizaciones incluyen medidas (medida de una estructura patológica) y anotaciones (regiones cercadas o flechas) para mejorar su interpretación. Por último estas visualizaciones y el informe adjunto apoyan las decisiones del tratamiento a realizar. También las visualizaciones serán utilizadas como tema de discusión entre los especialistas para emplearlos con fines educativos o como parte de una publicación.

1.4.1. Imágenes Médicas.

Se le llama imagen médica al resultado de aplicar un conjunto de técnicas y procesos, que son usados con el fin de crear imágenes del cuerpo humano, o parte de él, aprovechando en sí las características físicas y

químicas de estas, con propósitos clínicos (procedimiento médico que buscan revelar, diagnosticar o examinar enfermedades) o para la ciencia médica [10]**.**

Entre las modalidades de adquisición de imágenes médicas se destacan las de Tomografía Axial Computarizada (TAC), basadas en radiaciones electrónicas, la Resonancia Magnética (IRM), basada en la capacidad de algunos núcleos para absorber ondas de radiofrecuencia cuando son sometidos al efecto de un campo magnético y la "Tomografía Computarizada" (TC), la cual por su facilidad de realización, la precisión diagnóstica y la ausencia de riesgo, la ha convertido en la primera alternativa de diagnóstico que se les realiza a los pacientes.

Las imágenes médicas obtenidas utilizando las modalidades antes mencionadas son guardadas en archivos de formato diseñados para estas imágenes como el DICOM o pueden almacenarse en formato bruto dentro de los archivos RAW. A continuación se exponen algunas características de estos archivos.

⇒ **DICOM**

El formato DICOM, está contemplado en el estándar DICOM 3.0, para el almacenamiento, procesamiento y transmisión de imágenes médicas, el cual fue creado por *American College of Radiology* (ACR) y *National Electrics Manufacturers Association* (NEMA), para lograr una estandarización dentro de los fabricantes de equipos de adquisición de imágenes médicas. Por su uso a nivel mundial por parte de los sistemas informáticos que implementan el estándar DICOM, constituye uno de los formatos de almacenamiento de imágenes médicas más extendidos y populares que existen.

Internamente, un fichero DICOM está formado por dos componentes, el primero es una cabecera que contiene un número variable de conjuntos de datos o *DataSet*. Cada *DataSet* se compone de varios elementos de datos o *DataElement*. Define la información asociada a la imagen médica, así por ejemplo se tienen elementos de datos tales como: nombre del paciente, edad, espacio entre píxeles, modalidad y dimensión de la imagen y otros, almacenados en un archivo único (.dcm). El segundo componente de un fichero DICOM contiene una o más imágenes, es decir, el flujo de bytes donde se codifican las imágenes en sí. No debe confundirse con los datos de la imagen (capas, bits dedicados, filas, columnas) que se codifican en la cabecera [12].

⇒ **RAW**

RAW es un formato de archivo que no comprime los datos de la imagen al archivarla. Los ficheros RAW contienen la totalidad de los datos de la imagen tal como ha sido captada. Debido a que RAW contiene la totalidad de los datos de la imagen, sus ficheros ocupan una cantidad elevada de memoria.

.

1.4.2. Aplicación de la Visualización Médica.

En los últimos años las aplicaciones dirigidas al área de la visualización médica han cobrado gran auge, todo gracias a los avances que ha tenido hardware gráfico en este siglo. Las endoscopias virtuales y la visualización médica para la educación son dos de los principales campos en donde se aplica este tipo de visualización.

⇒ **Endoscopia virtual**

Una endoscopia virtual es una técnica mediante la cual, las imágenes adquiridas a partir de la Tomografía Axial Computarizada (TAC) y Resonancia Magnética (RM), son procesadas por la computadora y reconstruidas de forma tridimensional para lograr una visualización similar a la obtenida a través de un endoscopio [13]. Las aplicaciones de este tipo permiten al cirujano planificar la trayectoria de acceso a las zonas a intervenir apoyándose de una interfaz con diferentes vistas de localización de una "cámara virtual" (endoscopio virtual) que se introduce dentro de un volumen de datos. (18) En este tipo de aplicaciones normalmente se integran también técnicas de tratamiento y análisis para la extracción de estructuras de interés, de las cuales se pueden elaborar modelos tridimensionales para su posterior manipulación y procesamiento.

⇒ **Visualización médica en la educación**

Las técnicas de visualización médica tienen un gran potencial para la educación de la anatomía, así como para la educación de la cirugía. Desde estudiantes de secundaria y fisioterapeutas hasta los médicos que quieren ensayar intervenciones terapéuticas. Las técnicas de la visualización médica son cruciales para fines educativos: las superficies eficientes y de alta calidad, la visualización de volumen, así como la interacción para explorar los datos médicos de volumen tales como el corte y la resección virtual son esenciales ingredientes para los sistemas educativos [10].

1.5. Visualización Médica tridimensional en la Web.

En epígrafes anteriores se hacía referencia al papel que juega la visualización médica en el campo de la educación. Gracias al alto grado de interactividad que ofrece el uso de gráficos tridimensionales, su integración en las aplicaciones Web puede convertir a estas en un potente recurso didáctico del cual los usuarios pueden valerse para lograr un alto grado de motivación y un mejor aprendizaje. Las tecnologías de visualización tridimensional sobre la Web, estudiadas en secciones anteriores han motivado el

desarrollo de múltiples aplicaciones Web cuyo objetivo es mostrar de forma didáctica e interactiva diferentes secciones del complejo cuerpo humano.

A continuación se exponen algunas de las principales aplicaciones Web que integran visualizaciones médicas tridimensionales. Es importante aclarar que en la investigación no se muestra ninguna aplicación desarrollada en nuestro país debido a que no se encontró ningún trabajo relacionado con este tema.

⇒ **Visible Body**

Visible Body es una aplicación Web en 3D con la que se puede estudiar la anatomía humana de forma interactiva. Permite ver cada uno de los aparatos, sistemas y los órganos del cuerpo humano de manera individual o conjunta y desde cualquier punto de vista. El usuario puede hacer transparentes algunos de ellos u ocultarlos simplemente pinchando en un botón, acercar o alejar el modelo y girarlo en cualquier dirección del espacio con unos sencillos controles. El manejo, aunque la Web esté en inglés, es muy intuitivo y la calidad y detalle de la imagen 3D sorprendente.

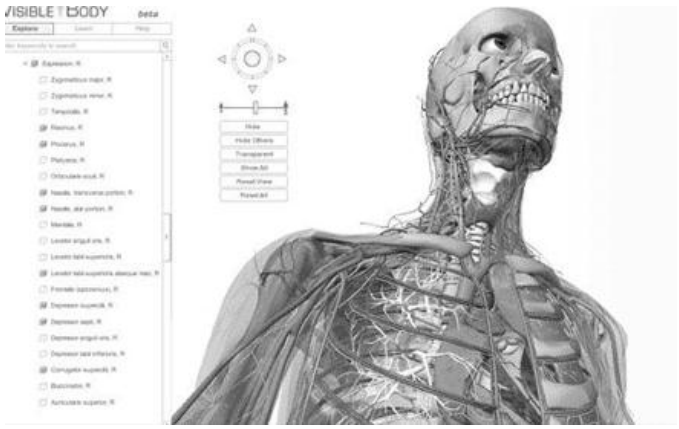

Figura 1. Visible Body

Ventajas: Permite buscar y localizar estructuras anatómicas por nombre, además de esconder, rotar, explorar y ver a través de las partes de la anatomía humana. También facilita al usuario mover, aumentar y disminuir en el espacio el modelo tridimensional.

Desventajas: Exige que se realicen subscripciones para poder usar su contenido mediante el pago de las mismas o sea es privado. Al ser una aplicación Web es necesario tener instalado en el navegador el *plugin* UnityWebPlayer para poder visualizarlo.

⇒ **Zygote Body**

Zygote Body es una aplicación Web gratuita que le permite al usuario recorrer el cuerpo humano de manera interactiva reconociendo las diferentes estructuras anatómicas de éste. En sus inicios este proyecto tenía el nombre de Google Body Browser, el cual estaba disponible a los usuarios desde los Labs[3] de Google, pero este proyecto dejó de estar en línea cuando Google cerró sus Labs en julio de 2011.
Gracias a Zygote (la empresa que proveía a Google las imágenes para su Body Browser) ha vuelto a ponerlo en línea bajo el nombre de Zygote Body [14]**.**

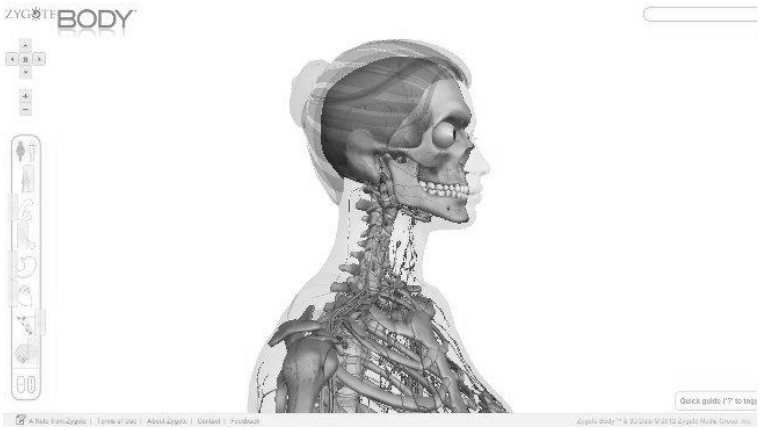

Figura 2. Zygote Body

Ventajas: Es un proyecto de código abierto. Cuenta con un buscador integrado de estructuras. Posee un navegador por capas mediante el cual se pueden activar o desactivar sistemas o estructuras de manera independiente. Puede ser visualizado desde cualquier navegador que soporte WebGL (Firefox, Google Chrome, Opera, Safari), lo que lo convierte en un proyecto multiplataforma.

⇒ **BodyMaps**

BodyMaps es un atlas anatómico interactivo realizado en Flash. Diseñado principalmente para la educación. Es un esfuerzo de colaboración con las redes GE HealthyImagination y HealthLine, una compañía de información de salud para los consumidores.

[3] Google Labs es un sitio web que muestra los nuevos proyectos de Google “que no están absolutamente listos para ser usados”. Sirve como una zona de prueba para los nuevos servicios que serán lanzados. Esto es una manera en que Google logra despertar interés en los usuarios antes de lanzar las versiones finales.

Además de proporcionarles a los pacientes la habilidad de aprender más sobre su anatomía, el sitio está lleno de enlaces a videos sobre enfermedades comunes que afectan a la parte del cuerpo que está seleccionada [15].

Figura 3. Body Maps

Ventajas: Se puede realizar la búsqueda a través del texto o haciendo clic en diferentes partes del cuerpo, una especie de Google Maps.
Desventajas: La resolución de los modelos no es muy alta.

⇒ **Visión Médica Virtual**

Es una compañía de reciente creación y su objetivo es generar un software, que permita visualizar en línea un Atlas 3D de Anatomía Humana, interactivo y de alto rigor científico-médico. Cualquier universidad de medicina o escuela de ciencias de la salud puede acceder a los modelos 3D del Atlas Anatómico, aunque para esto, es necesario ponerse en contacto con 3DVisioMédicavirtual y establecer las condiciones de uso y privacidad de los contenidos de la Web. Existe la posibilidad de acuerdos de acceso restringidos a grupos y se puede acceder mediante el pago de una cuota anual.

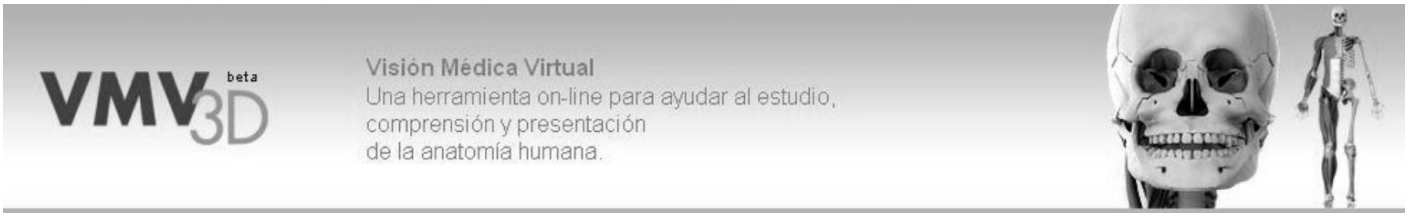

Figura 4. Visión Médica Virtual

1.6. Consideraciones parciales

Con la conclusión del capítulo se logró conformar un compendio de conceptos y caracterizaciones que ayudan a una mejor comprensión del contenido de este trabajo. Se caracterizaron las tecnologías existentes en el mundo para lograr la visualización tridimensional en la Web tales como VRML, X3D, Flash 3D, Java 3D y WebGL, donde se seleccionó esta última teniendo en cuenta que no depende de un *plugin* para su ejecución, pues viene integrado de forma nativa en los navegadores Web más modernos como Moxilla Firefox y Google Chrome. Se estudiaron las aplicaciones Web que permiten la visualización médica tridimensional y se identificó que no existe ninguna aplicación con este fin desarrollada en Cuba, a nivel internacional las aplicaciones que existen no permiten la visualización tridimensional de imágenes médicas, sino que visualizan modelos anatómicos, diseñados con herramientas de modelado 3D.

CAPÍTULO 2. Solución Propuesta

2.1. Introducción

En este capítulo se realizará la descripción de la solución propuesta para dar respuesta a la situación problémica planteada. Primeramente se exponen algunos de los principales algoritmos más utilizados en la visualización de volumen. Acto seguido se explica todo el funcionamiento del algoritmo utilizado en el desarrollo de la aplicación, el Raycasting, desde la perspectiva de la Web utilizando la tecnología WebGL, se caracterizan cada uno de sus pasos y se proporcionan detalles de su implementación. Por último se mencionan las metodologías y herramientas de desarrollo empleadas en la realización de este trabajo.

2.2. Algoritmos de visualización de volumen

La visualización de volumen consiste en la representación visual del conjunto completo de datos volumétricos [16], 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. La visualización directa (DVR) e indirecta (IVR) de volumen son las dos áreas principales en las que se divide este tipo de visualización. A continuación se exponen los algoritmos más populares pertenecientes a las áreas mencionadas anteriormente.

2.2.1. Algoritmos de IVR

Los algoritmos de IVR se basan fundamentalmente en el uso de planos (*Planar Rendering*) o superficies geométricas como representación intermedia (*Surface Rendering*).

⇒ **Algoritmo *Marching Cubes***

El algoritmo *Marching Cubes* fue propuesto por William E. Lorensen y Harvey E. Cline en 1987 [17] se basa en el estudio del volumen adquirido mediante subdivisión en pequeños vóxeles o cubos y consta además con una tabla de búsqueda (ver Figura 5) donde aparecen los 15 casos posibles en los que un cubo puede ser interceptado por la superficie. 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.

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 de 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.

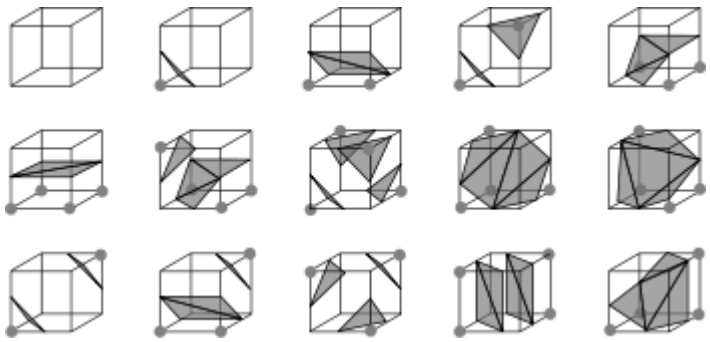

Figura 5. Tabla de búsqueda del Algoritmo *Marching Cubes*

⇒ **Planar Rendering**

Esta capacidad fue descrita por primera vez por Cullip y Neumann [18], consiste en aprovechar las interpolaciones bilineal y trilineal realizadas en la GPU. Ellos 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.

Con esta técnica se obtienen resultados satisfactorios en cuanto a la visualización de volúmenes, ya que ofrece, en dependencia de los recursos 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.

2.2.2. Algoritmos de DVR

Mediante esta técnica de visualización de volumen, los datos volumétricos son representados mediante su evaluación en un modelo óptico, que describe cómo el volumen emite, refleja, dispersa, absorbe y ocluye la luz [16]. Estos algoritmos se pueden clasificar en dos grupos: algoritmos basados en objetos y algoritmos basados en imágenes. 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.

2.2.2.1. Algoritmos basados en objetos

⇒ **Algoritmo Splatting**

En contraste con algunos algoritmos basado en imágenes que basan su funcionamiento en la toma de muestras a lo largo de rayos que viajan a través del volumen, el algoritmo Splatting proyecta los vóxeles hacia el plano de la imagen final [16]. 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.

⇒ **Texture-Mapping**

Uno de los más recientes algoritmos de DVR es Texture-Mapping, descrito en 1994 [19], 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.

2.2.2.2. Algoritmos basados en imágenes

⇒ **Algoritmo Shear Warp**

El algoritmo Shear Warp 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. Este basa su funcionamiento en que

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 [16].

⇒ **Raycasting**

Para dar cumplimiento a la solución propuesta en el Epígrafe 2.3 se profundiza el análisis de este algoritmo y su Implementación sobre la Web utilizando WebGL.

2.3. Implementación del Algoritmo Raycasting en la Web

2.3.1 Principio del algoritmo Raycasting

Raycasting es un algoritmo de DVR basado en imágenes que utiliza una evaluación directa de la Ecuación de la Visualización de Volumen [20]. 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 Figura 6 muestra gráficamente el principio básico del Raycasting.

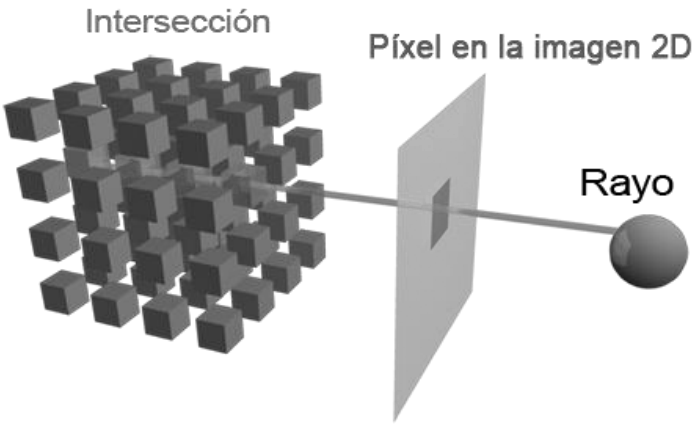

Figura 6. Principio del Algoritmo Raycasting

2.3.2 Algoritmo Raycasting utilizando WebGL

Para el desarrollo del algoritmo en cuestión se contará con cuatro fases fundamentales que son expuestas a continuación:

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.

En los epígrafes siguientes se exponen con más detalles cada fase del desarrollo del algoritmo.

2.3.2.1 Generación de las caras delanteras

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. 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. 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.

Las imágenes resultantes son dos simples cubos de colores, que se muestran en la Figura 7. Geometría limitante. Derecha: Caras delanteras, Izquierda: Caras traseras.. Cada imagen es almacenada en una textura independiente para su posterior consulta.

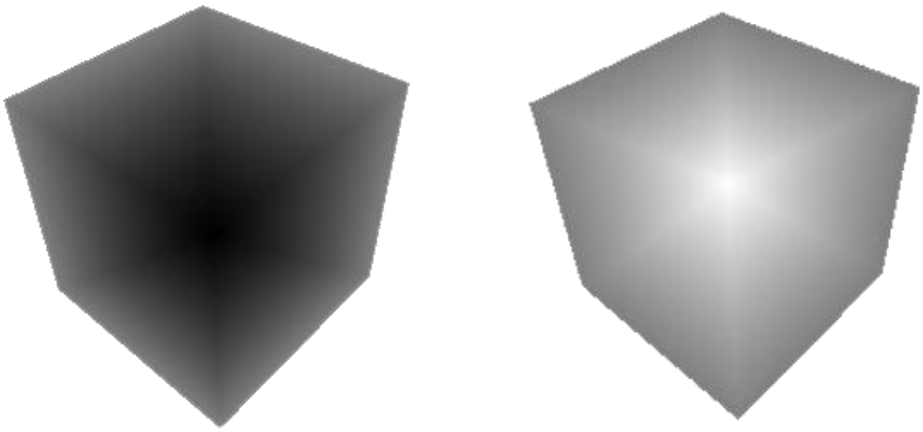

Figura 7. Geometría limitante. Derecha: Caras delanteras, Izquierda: Caras traseras.

2.3.2.2 Generación de la textura direccional

La diferencia de las dos imágenes anteriores 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.

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.

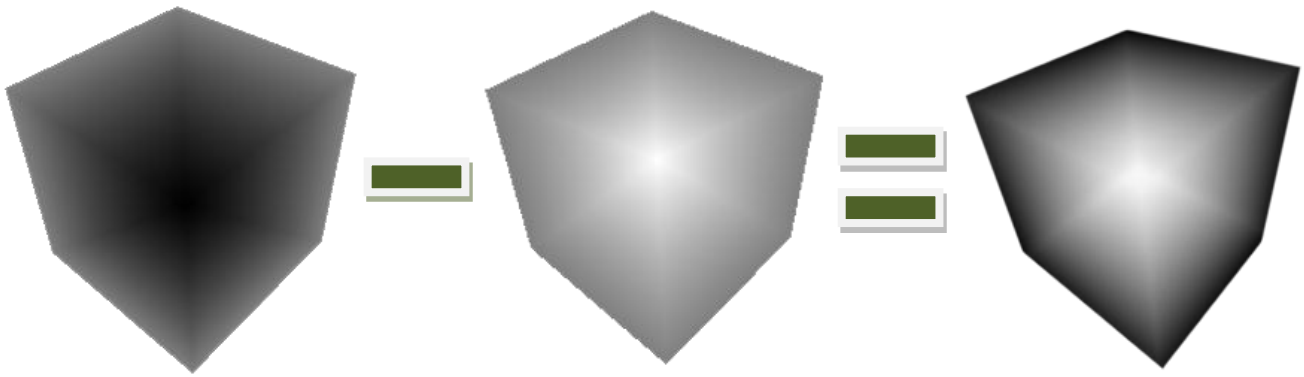

Figura 8. Creación de la textura direccional.

El *fragment program* tiene como entrada el valor de color de las caras traseras de la geometría limitante (ver Figura 8 izquierda) 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 se obtiene el vector de visión o vector direccional (ver Figura 8 derecha). 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. Toda la corrección de perspectiva y el cálculo del vector de visión se llevan a cabo por el hardware gráfico sin prácticamente consumir tiempo [21], la representación de dos cubos de colores no es ningún problema.

2.3.2.3 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.

- ⇒ **Búsqueda dentro de la textura 3D**

En una aplicación de escritorio la búsqueda dentro de la textura 3D ya vienen implementadas en OpenGL, en la cual se realiza automáticamente la interpolación trilineal. En la solución que se propone sucede todo lo contrario, debido a que la versión de WebGL 1.0 que se utiliza, está desarrollada basada en la especificación de OpenGL ES 2.0, donde esta funcionalidad no está implementada, se hace necesario

buscar una alternativa para completar el algoritmo. A continuación se expone una solución para realizar esa importante funcionalidad, la cual se divide en dos fases.

1ra Fase: Generación de una textura 2D a partir del volumen de datos.

El objetivo de esta fase es crear una textura compuesta por todos los cortes que contiene un volumen, tal como se muestra en la Figura 9. Este paso se lleva a cabo como un pre procesamiento ante de realizar el algoritmo. Después de haber creado la textura, se carga y se envía al fragment program utilizando las funcionalidades de WebGL, para ser utilizada en la siguiente fase.

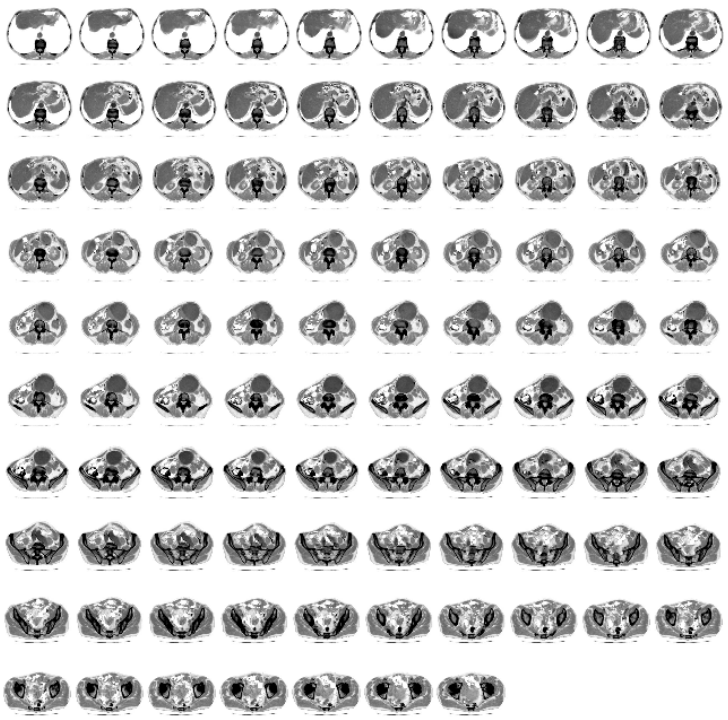

Figura 9. Imagen generada a partir de un volumen.

2da Fase: Extracción del valor en la textura 2D.

Para la extracción correcta del valor de un pixel en las coordenadas x, y, z contenido dentro de la textura 2D se llevaron a cabo las siguientes ecuaciones:

$$s_1 = floor(z * S)$$
$$s_2 = s_1 + 1$$
$$dx_1 = fract\left(\frac{s_1}{M_x}\right)$$
$$dy_1 = \frac{fract\left(\frac{s_1}{M_y}\right)}{M_y}$$
$$dx_2 = floor\left(\frac{s_2}{M_x}\right)$$
$$dy_2 = \frac{fract\left(\frac{s_2}{M_y}\right)}{M_y}$$
$$tx_1 = dx_1 + \frac{x}{M_x}$$
$$ty_1 = dy_1 + \frac{y}{M_y}$$
$$tx_2 = dx_2 + \frac{x}{M_x}$$
$$ty_2 = dy_2 + \frac{y}{M_y}$$
$$v_1 = tex2D(tx_1, ty_1)$$
$$v_2 = tex2D(tx_2, ty_2)$$
$$V_a(x, y, z) = mix(v_1, v_2, (x \times S) - s_1)$$

Donde S es el número total de imágenes en el mosaico, Mx y My son el número de imágenes en el mosaico por cada fila y columna como muestra la Figura 9. Las funciones que se presentan en las ecuaciones se definen utilizando la especificación de OpenGL Shading Language (GLSL) [22].

Con la ayuda de la función antes expuesta se obtienen las contribuciones de color de la muestra, estas se acumulan en una variable hasta que el rayo abandone el volumen (este comportamiento se muestra en la Figura 10), acto seguido se mezclan las contribuciones de todas las muestras tomadas a lo largo del rayo con lo cual se determina el color final del pixel. 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.

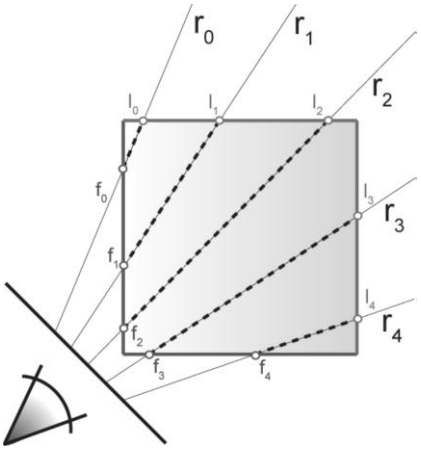

Figura 10. Contribución de las muestras a la imagen final

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. [21]

2.3.2.4 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.

2.4. 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.

2.4.1 Metodología de Desarrollo de Software

Se escogió como metodología de desarrollo de software el Proceso Unificado de Desarrollo (RUP) porque constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. 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 [23].

⇒ **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 [23].

⇒ **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.

2.4.2 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).

2.4.3 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.

2.4.4 Lenguaje de programación

Como lenguajes de programación se utilizaron PHP y JavaScript, de los cuales se presentan sus características más esenciales.

⇒ **PHP**

PHP son las siglas de *Hypertext Preprocessor* (Procesador de Hipertextos), es generalmente muy usado por su flexibilidad. Es un lenguaje de código abierto de *scripting* (escritura) especialmente adecuado para el desarrollo de Web.
PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Además permite la conexión a diferentes tipos de servidores de base de datos tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird, SQLite logrando aplicaciones Web robustas y sus archivos cuentan con la extensión (.php) [16].
Dentro de las principales ventajas que presenta este lenguaje se encuentran:

- ⇒ Es identificado como un lenguaje muy rápido y manuable.
- ⇒ Soporta de cierta forma la orientación a objeto. Clases y herencia.
- ⇒ Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- ⇒ Capacidad de conexión con la mayoría de los manejadores de base de datos.
- ⇒ Puede expandir su capacidad de potencial utilizando módulos.
- ⇒ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos y su código está disponible bajo la licencia GPL.
- ⇒ Posee una biblioteca nativa de funciones sumamente amplia e incluida.
- ⇒ No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

⇒ **JavaScript**

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y

desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios [24].

JavaScript está basado en objetos, tiene un tipado relajado y vinculación dinámica. Además, está interpretado por el explorador del cliente, aunque el término "compilador JavaScript" se utiliza frecuentemente para hacer referencia al mecanismo incorporado del explorador que lee el código y, o bien lo ejecuta, o bien devuelve mensajes de error [25].

2.4.5 Técnicas de programación

Se utiliza Ajax para enviar las peticiones del cliente hacia el servidor. Ajax es el acrónimo de ***Asynchronous JavaScript And XML*** (JavaScript y XML asíncronos). Ajax no es una tecnología porque es la unión de un conjunto de tecnologías para el desarrollo WEB que se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano.

Ajax tiene una presentación basada en estándares usando XHTML y CSS. Además realiza una interacción dinámica usando el DOM y realiza el intercambio y manipulación de datos usando XML. Esta tecnología posee recuperación de datos asincrónica usando XMLHttpRequest. Los datos adicionales que carga del servidor no sobrecargan la página ni afecta su comportamiento porque no interfiere con la visualización de la misma [26].

2.5. Consideraciones parciales

En este capítulo se caracterizaron los principales algoritmos de visualización de volumen, donde se seleccionó el algoritmo Raycasting, por su rendimiento, calidad, extensibilidad y robustez a la hora de realizar la visualización. Se propuso la solución científico-técnica al problema planteado, se explicó con detalles la implementación del algoritmo seleccionado utilizando WebGL, donde se describieron los pasos necesarios para la implementación del mismo (generación de las caras delanteras, generación de la textura direccional, Raycasting sobre la Web y composición). También se mencionaron las principales metodologías y herramientas para el desarrollo de la solución propuesta.

CAPÍTULO 3. Características del sistema

3.1. Introducción

En este capítulo se hace énfasis en las características de la solución propuesta donde se definen las reglas específicas del negocio y el modelo de dominio del problema. Se muestran los requisitos funcionales y no funcionales detectados durante la Captura de Requisitos y el modelo de Casos de Uso del Sistema. Dentro del modelo de Casos de Uso del Sistema se muestran 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.

3.2. Reglas del Negocio

Las reglas del negocio describen con un nivel adecuado de detalle las políticas, normas, operaciones, definiciones y restricciones presentes en una organización, estas son de vital importancia para alcanzar los objetivos planteados. Para el desarrollo de la aplicación 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.
3) Las imágenes con el formato *.dcm deben estar comprimidas en formato *.zip.
4) Las imágenes con el formato *.raw deben tener las dimensiones del estudio sin modificaciones.

3.3. Modelo de Dominio

Cuando no se pueden identificar claramente los procesos de negocio se elabora un modelo de dominio el cual mediante la representación de un conjunto de conceptos y las relaciones que se establecen entre ellos ayudarán tanto a usuarios como a desarrolladores a usar un vocabulario común que les permita entender el contexto en que se emplaza el sistema a desarrollar. A continuación se muestra el modelo del dominio utilizado en la solución propuesta.

A un paciente se le realiza una serie de estudios por parte de los médicos, estos pueden ser a través de tomografías computarizadas (TC) o resonancias magnéticas (RM). 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 obtener un diagnóstico final. En la Figura 11 se muestra la descripción del negocio con el objetivo de facilitar la comprensión del funcionamiento del sistema.

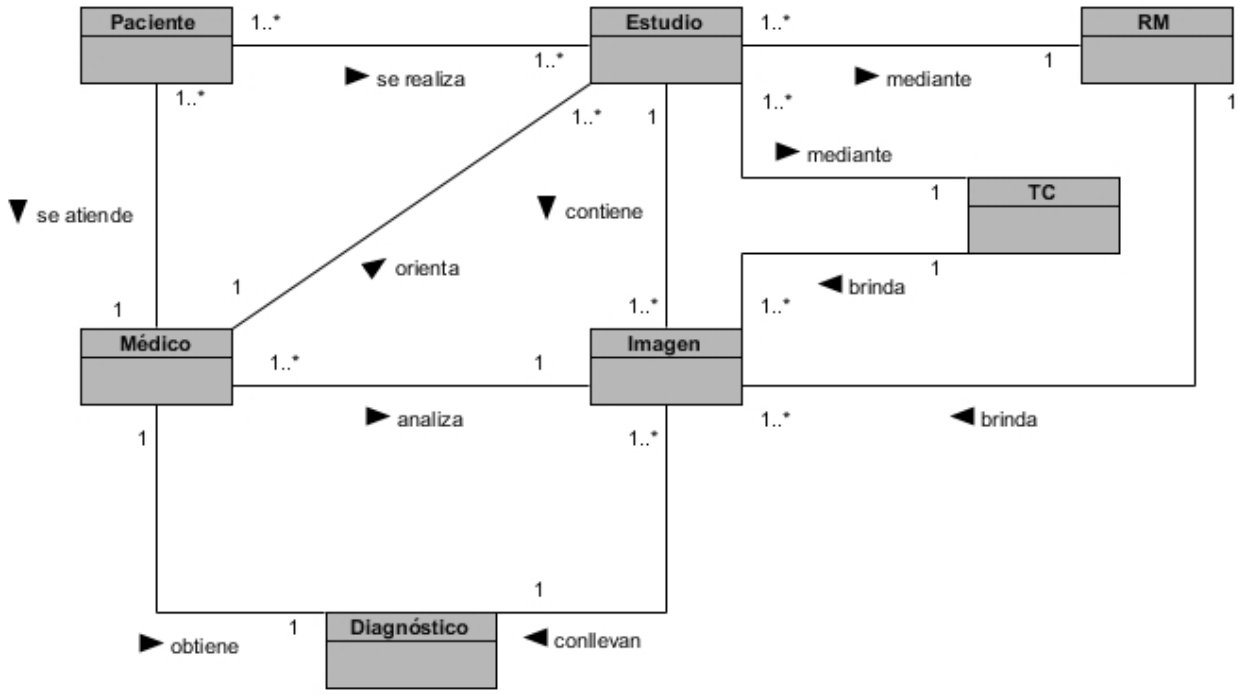

Figura 11. Modelo de Dominio

Médico Especialista: Persona capacitada en el manejo de los métodos de imagenología diagnóstica y en los procedimientos endoscópicos.

Paciente: Persona que recibe los servicios de un médico u otro profesional de la salud, sometiéndose a un examen.

Estudio: Examen orientado por el médico al paciente para obtener una patología y emitir un diagnóstico.

RM: 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.

TC: 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.
Imagen: Resultado del estudio orientado por el médico al paciente a través del TC y RM.
Diagnóstico: Resultado que emite el médico luego de realizar el estudio a un paciente.

3.4. Captura de Requisitos

Los requisitos son una especificación de lo que el sistema debe cumplir para satisfacer al cliente. Estos pueden dividirse en requisitos funcionales y requisitos no funcionales. En los próximos epígrafes se especifican los requisitos tanto funcionales como no funcionales.

3.4.1 Requisitos Funcionales

Los requisitos funcionales definen el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación. Los requisitos funcionales son capacidades o funciones que el sistema debe cumplir. A continuación se presentan los requerimientos para desarrollar la solución propuesta.

RF1. Cargar archivo.

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

RF2. Visualizar modelo tridimensional.

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

RF3. Manipular modelo.

- **RF3.1** Mover Modelo en el Eje de coordenada Z.
- **RF3.2** Rotar Modelo en el Eje de Coordenada X.
- **RF3.3** Rotar Modelo en el Eje de Coordenada Y.

RF4. Visualizar modelo anterior.

3.4.2 Requisitos no Funcionales

Los requisitos no funcionales son las cualidades o propiedades que la aplicación debe tener para lograr características que lo hagan rápido, confiable, atractivo, usable y seguro, etc. En ellos se evidencian las restricciones del entorno, de la implementación, el rendimiento del sistema, así como las dependencias de plataformas entre otros aspectos. Los requisitos no funcionales tomados en cuenta en el presente trabajo son los que se describen a continuación:

1. **RNF 1 Software**

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

⇒ Firefox 4 o superior, Google Chrome 12 o superior.

2. **RNF 2 Hardware**

2.1. Desarrollo y ejecución

⇒ 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. **RNF 3 Seguridad**

⇒ *Fiabilidad:* Si ocurre una falla de conexión la visualización del modelo no se afectara, al menos que el usuario vuelva a seleccionar la opción Visualizar modelo.

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

4. **RNF 4 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. **RNF 6 Restricciones en el Diseño e Implementación**

⇒ Se empleará el lenguaje de programación PHP bajo el paradigma de Programación Orientada a Objetos.

6. **RNF 7 Rendimiento**

⇒ Cantidad de cuadros que es capaz de representar en un segundo (FPS).

⇒ Consumo de memoria RAM.

⇒ Consumo de CPU.

⇒ Consumo de la GPU.

⇒ Tiempo que demora en crear la textura 2D.

3.5. Modelo de Casos de Uso

En este epígrafe se muestran los actores del sistema que van a interactuar con los Casos de Usos definidos a partir de los requisitos funcionales antes expuestos. Esto posibilitará un mejor entendimiento de las funcionalidades que se requieren del sistema y darán un resultado de valor para los actores identificados. A continuación se muestran los actores y los casos de uso identificados.

3.5.1 Actores del Sistema

Los actores son agentes que pueden interactuar con el sistema. Un actor en términos de Ingeniería de Software, representa el rol de una o varias personas, un equipo o sistema automatizado. En el caso particular de la aplicación que se desarrolló, quien interactuará con el sistema es una persona que interpreta el rol de especialista médico con conocimientos de imagenología, por lo que como actor del sistema se le llamará Especialista Médico.

Tabla 1. Actores del Sistema

Actores	Justificación
Especialista Médico	Interactúa con el sistema durante la ejecución de sus funcionalidades. Carga los archivos con las imágenes pertenecientes a un estudio realizado al paciente y visualiza los modelos anatómicos tridimensionales.

3.5.2 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 Figura 12).

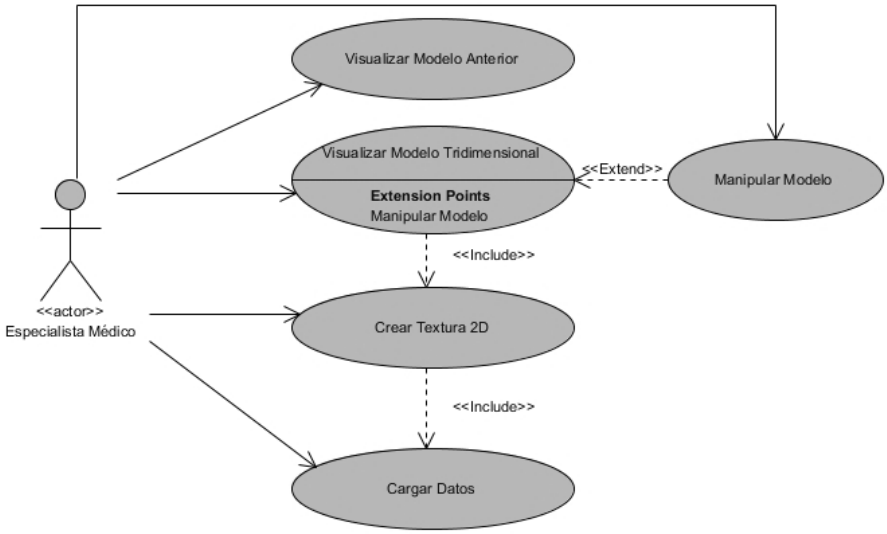

Figura 12. Diagrama de Casos de Uso del Sistema.

Los Casos de Uso tomados en cuenta en el presente trabajo están fuertemente relacionados, por ejemplo para la ejecución del Caso de Uso Visualizar Modelo es necesario que el Caso de Uso Crear Textura 2D se realice correctamente, este último depende explícitamente del Caso de Uso Cargar Datos.

3.5.3 Descripción de los Casos de Uso del Sistema

Con la descripción de los Casos de Usos del sistema se brinda una mayor información y entendimiento de la aplicación a implementar. Las tablas presentadas a continuación argumentan los flujos operacionales y las descripciones de los Casos de Usos detectados en el presente trabajo.

Tabla 2. Descripción del Caso de Uso Cargar Datos.

Caso de Uso:	Cargar Datos
Actores:	Especialista Médico
Propósito:	Importar los ficheros con extensión *.zip o *.raw.
Resumen:	Se inicia cuando el actor escoge el tipo de archivo y selecciona el

	archivo. Se selecciona el directorio donde están los ficheros ZIP con las imágenes DICOM comprimidas o el fichero RAW deseado. Finaliza con la carga de todos los ficheros seleccionados para usarlos posteriormente y habilita la opción Crear Textura.
Referencia:	RF1.
Precondición:	N/A
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción de Tipo de Archivo.	1.1 Realiza las acciones que se detallan a continuación: a) Si el usuario seleccionó el tipo de archivo RAW consultar Sección Seleccionar tipo de archivo DICOM. b) Si el usuario seleccionó el tipo de archivo RAW consultar Sección Seleccionar tipo de archivo RAW.
2. Selecciona el opción Seleccionar el Fichero.	2.1 Muestra un cuadro de diálogo que le permite al usuario seleccionar el o los ficheros que desea cargar. 2.2 El cuadro de diálogo solo mostrará los ficheros con las extensiones especificadas anteriormente.
3. Selecciona el directorio o los ficheros que desea cargar y acepta.	3.1 Se cierra el cuadro de diálogo y se procede a cargar los ficheros seleccionados.
Postcondiciones:	Se cargaron los ficheros seleccionados. Se habilitó la opción Crear Textura.
Prioridad:	Crítica.

Tabla 3. Descripción de la Sección Seleccionar tipo de archivo DICOM

Sección:	Seleccionar tipo de archivo DICOM.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:

1. Selecciona el tipo de archivo DICOM.	1.1 Activa el campo para seleccionar el archivo.
Postcondiciones:	Se visualiza activo el campo para seleccionar el archivo.
Prioridad:	Crítica.

Tabla 4. Descripción de la Sección Seleccionar tipo de archivo RAW

Sección:	Seleccionar tipo de archivo RAW.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona el tipo de archivo DICOM.	1.1 Muestra un campo de texto para insertar el tamaño de la imagen dentro del archivo.
Postcondiciones:	Se visualiza el campo de texto tamaño de la imagen.
Prioridad:	Crítica.

Tabla 5. Descripción del Caso de Uso Crear Textura 2D.

Caso de Uso:	Crear Textura 2D
Actores:	Especialista Médico
Propósito:	Crear la textura 2D a partir de los archivos cargados.
Resumen:	Se inicia cuando el actor ejecuta la opción Crear Textura 2D. Crea una imagen con los cortes ordenados en forma de matriz. Finaliza habilitando la opción Visualizar Modelo.
Referencia:	RF2 ,RF2.1
Precondición:	Los datos deben estar cargados
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción de Crear Textura 2D.	1.1 El sistema muestra un mensaje con el texto “Creando la Textura” y un icono animado representativo de la acción que se está realizando.
Postcondiciones:	Se habilitó la opción Visualizar Modelo.
Prioridad:	Crítica.

Tabla 6. Descripción del Caso de Uso Visualizar Modelo Tridimensional.

Caso de Uso:	Visualizar Modelo Tridimensional
Actores:	Especialista 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 y la correcta creación de la textura. El sistema muestra una representación tridimensional de las estructuras anatómicas contenidas dentro de los ficheros seleccionados. Finaliza con el cierre de la aplicación o la selección de un nuevo fichero.
Referencia:	RF1, RF2.1
Precondición:	Debe haberse creado la textura.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción de Visualizar Modelo.	1.1 Crea la textura tridimensional a partir de la imagen creada utilizando el conjunto de datos cargado. 1.2 Realiza la representación de los datos empleando el algoritmo Raycasting utilizando WebGL. 1.3 Muestra los resultados en pantalla.
Postcondiciones:	Se visualizó en pantalla la representación tridimensional del modelo cargado.
Prioridad:	Crítica.

Tabla 7. Descripción del Caso de Uso Manipular Modelo.

Caso de Uso:	Manipular Modelo
Actores:	Especialista Médico
Propósito:	Movimiento del modelo para una exploración de las estructuras anatómicas en la escena.
Resumen:	Con el movimiento y el uso de los botones del mouse, el Médico manipula los movimientos permisibles en la escena.

Referencia:	RF3.
Precondición:	El modelo debe estar visualizado.
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). 2. Utiliza algunas teclas para aplicar movimiento (acercar y alejar).	1.1 Realiza un conjunto de acciones que se detallan a continuación: a) Si el actor sostiene el botón izquierdo del mouse y lo arrastra horizontalmente: consultar Sección Rotar Modelo en el Eje de Coordenada Y. b) Si el actor sostiene el botón izquierdo del mouse y lo arrastra verticalmente: consultar Sección Rotar Modelo en el Eje de Coordenada X. 1.2 Se actualiza la escena, rotando el modelo según la dirección recibida. 2.1 Realiza un conjunto de acciones que se detallan a continuación: a) Si el actor presiona las teclas “Num +” y “Num -”: consultar Sección Mover Modelo en el Eje de Coordenadas Z. 2.2 Se actualiza la escena, trasladando el modelo según la dirección recibida.
Postcondiciones:	Se representó la escena con la nueva posición y orientación.
Prioridad:	Crítica.

Tabla 8. Descripción de la Sección Rotar Modelo en el Eje de Coordenada Y.

Sección:	Rotar Modelo en el Eje de Coordenada Y.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Sostiene el botón izquierdo del mouse y lo arrastra horizontalmente.	1.1 Recibe los eventos correspondientes y rota el modelo en el Eje de Coordenada Y. O sea movimientos hacia la izquierda y hacia la derecha.

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

Tabla 9. Descripción de la Sección Rotar Modelo en el Eje de Coordenada X.

Sección:	Rotar Modelo en el Eje de Coordenada X.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Sostiene el botón izquierdo del mouse y lo arrastra verticalmente.	1.1 Recibe los eventos correspondientes y rota el modelo en el Eje de Coordenada X. O sea movimientos hacia arriba y hacia abajo. 1.2 Se actualiza la escena a partir de la nueva posición del modelo.
Postcondiciones:	Se representó la escena con la nueva orientación.
Prioridad:	Crítica.

Tabla 10. Descripción de la Sección Mover Modelo en el Eje de Coordenadas Z.

Sección:	Mover Modelo en el Eje de Coordenadas Z.
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Presiona las teclas “Num +” y “Num -”.	1.1 Recibe los eventos correspondientes y realiza las siguientes acciones: a) Si se presiona la tecla “Num +”, modifica el Eje de Coordenada Z acercando el modelo. b) Si se presiona la tecla “Num -”, modifica el Eje de Coordenada Z alejando el modelo. 1.2 Se actualiza la escena a partir de la nueva posición del modelo.
Postcondiciones:	Se representó la escena con la nueva orientación.
Prioridad:	Crítica.

Tabla 11. Descripción del Caso de Uso Visualizar Modelo Anterior

Caso de Uso:	Visualizar Modelo Anterior
Actores:	Especialista Médico
Propósito:	Visualizar el modelo creado en la última ejecución de la aplicación.
Resumen:	El sistema muestra una representación tridimensional de las estructuras anatómicas a partir de los datos generados en la última ejecución de la aplicación. Finaliza con el cierre de la aplicación o la selección de un nuevo fichero.
Referencia:	RF1, RF2.1
Precondición:	N/A
Flujo Normal de Eventos:	
Acción del Actor:	Respuesta del Sistema:
1. Selecciona la opción de Visualizar Modelo.	1.1 Crea la textura tridimensional a partir de la imagen creada utilizando el conjunto de datos cargados. 1.2 Realiza la representación de los datos empleando el algoritmo Raycasting utilizando WebGL. 1.3 Muestra los resultados en pantalla.
Postcondiciones:	Se visualizó en pantalla la representación tridimensional del modelo cargado.
Prioridad:	Crítica.

3.6. 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. 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.

3.6.1 Diagrama de Clases del Diseño

Los diagramas de clases del diseño son una representación concreta, representan la parte estática del sistema, las clases y sus relaciones. En la figura se muestra el diagrama de clases del diseño, el cual resultó ser el punto de partida para comenzar a elaborar la solución en términos de los lenguajes de programación seleccionados.

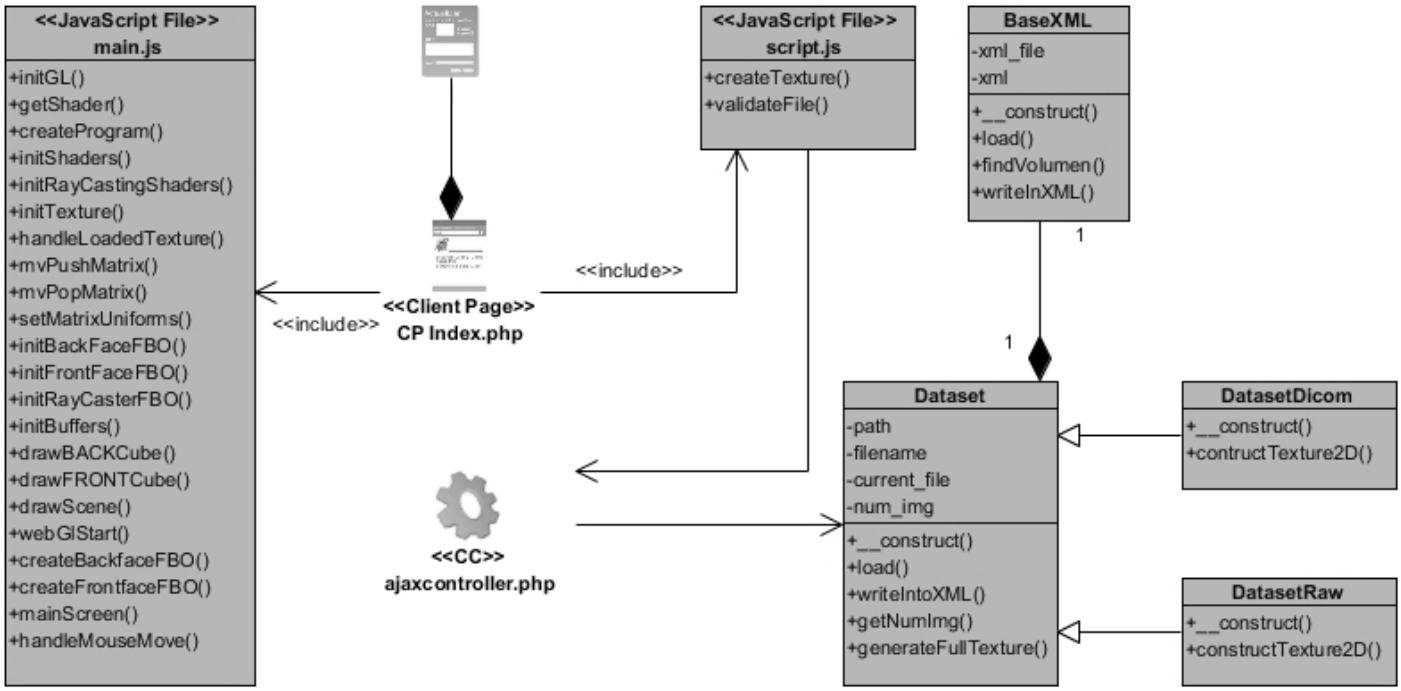

Figura 13. Diagrama de Clases del Diseño

3.6.2 Diagramas de Secuencia del Diseño

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. En este epígrafe se muestran los diagramas de secuencia correspondientes a los casos de uso descritos en el epígrafe 3.5.3.

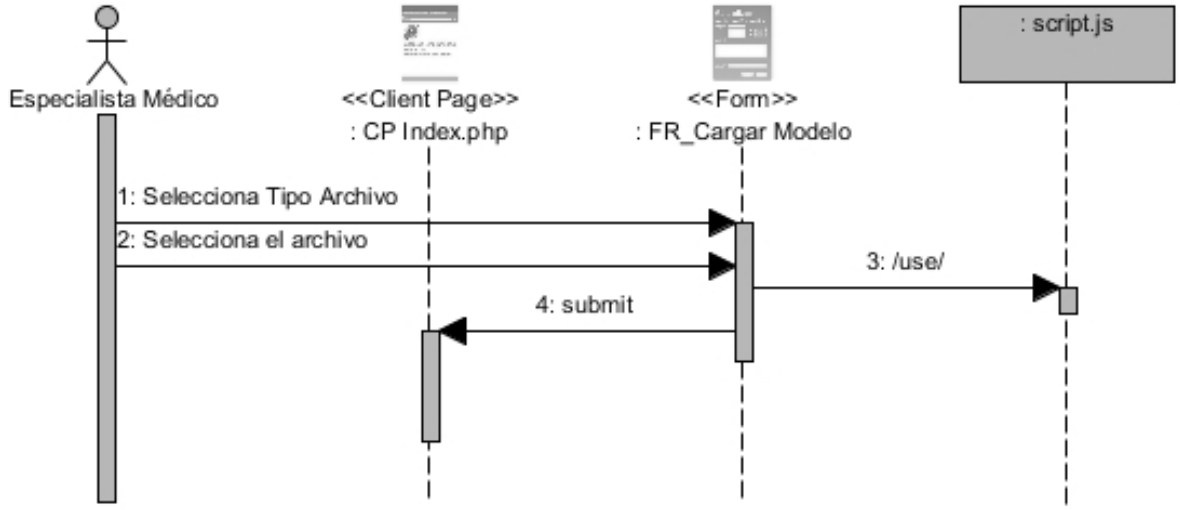

Figura 14. Diagrama de Secuencia Caso de Uso Cargar Datos

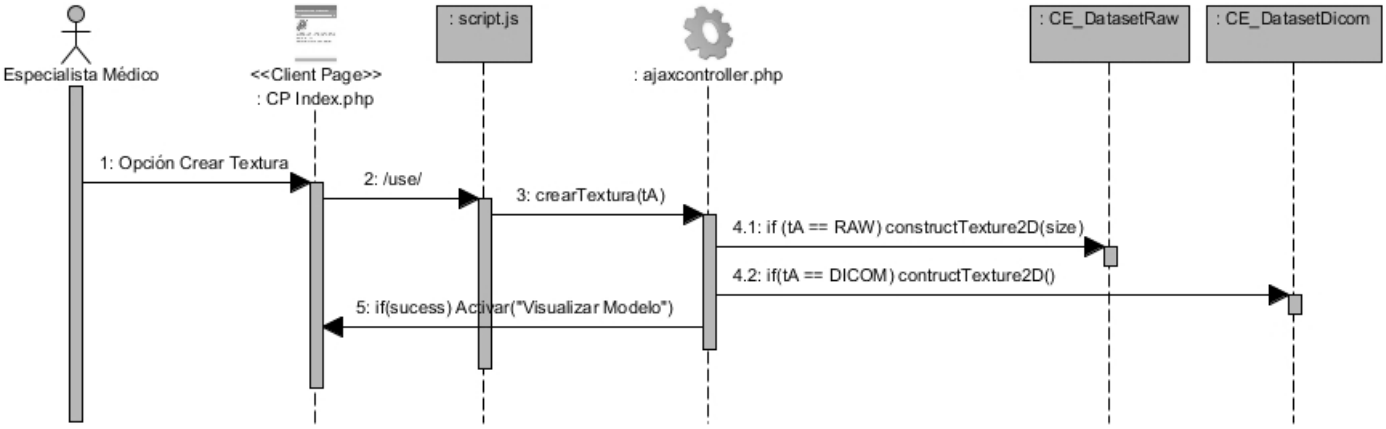

Figura 15. Diagrama de Secuencia Caso de Uso Crear Textura 2D

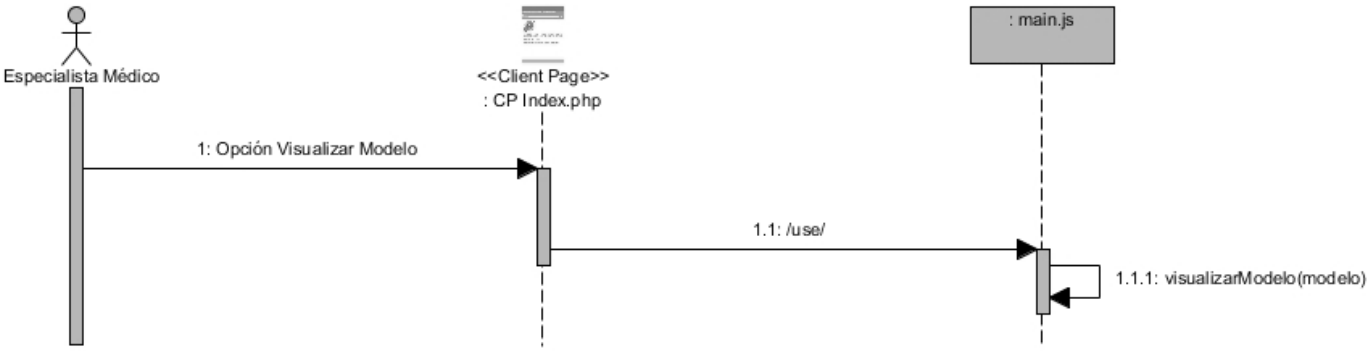

Figura 16. Diagrama de Secuencia Caso de Uso Visualizar Modelo

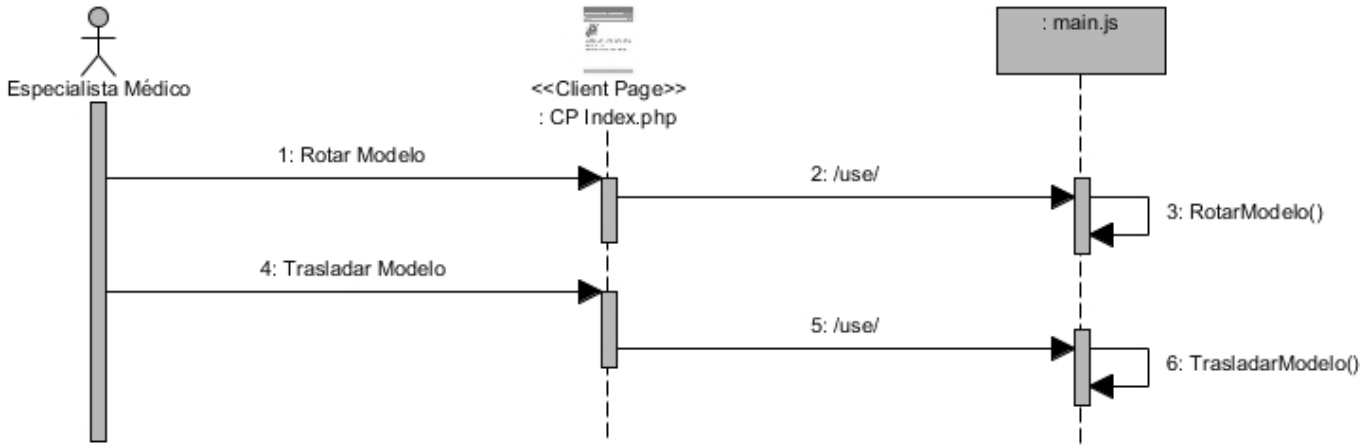

Figura 17. Diagrama de Secuencia Caso de Uso Manipular Modelo

CAPÍTULO 4. Implementación y validación de los resultados

4.1. Introducción

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 y de despliegue del sistema desarrollado. Posteriormente se toman casos de prueba para validar los resultados alcanzados, fundamentalmente relacionados con el rendimiento del sistema, estos se expresan respectivamente mediante tablas que contienen las variables fundamentales para futuras comparaciones con otros algoritmos.

4.2. Implementación

El resultado principal de la implementación es describir cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. Estos componentes representan la composición física de la implementación del sistema.

4.2.1 Diagramas de componentes

El diagrama de componentes es usado para estructurar el modelo de implementación y mostrar las relaciones entre los elementos de implementación. Este diagrama muestra la estructura de alto nivel del modelo de implementación, lo que permite a los desarrolladores y clientes conocer 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 Figura 18), y se muestra una breve descripción de las partes que lo conforman.

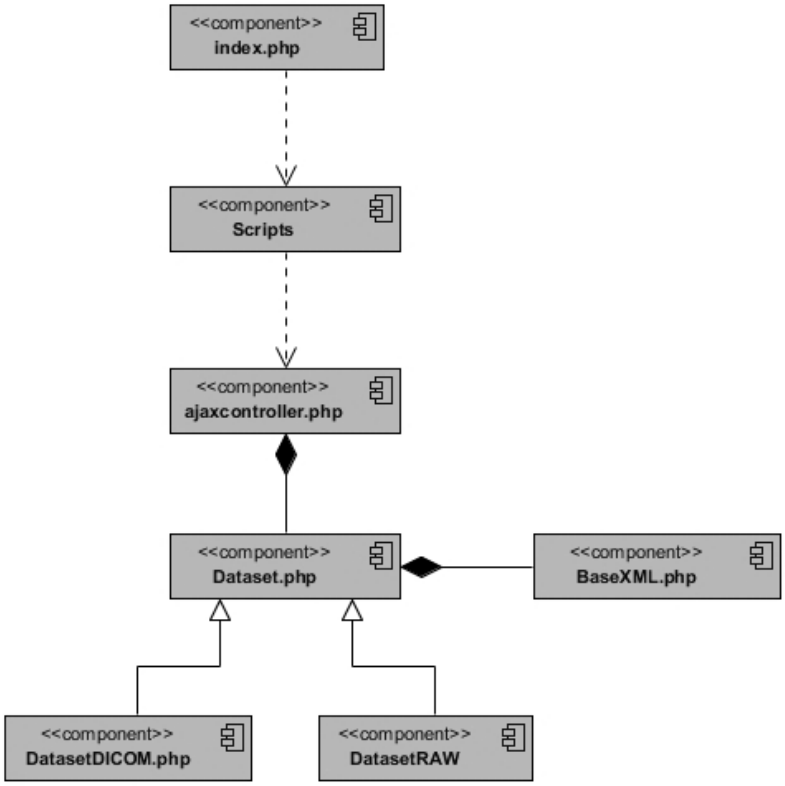

Figura 18. Diagrama de Componentes

Los componentes están divididos de la siguiente forma, los que se ejecutan en el cliente y los que están del lado del servidor.

En el cliente se encuentran:

- ⇒ **index.php:** Muestra la interfaz de la aplicación, con los formularios y las opciones para realizar las funcionalidades.
- ⇒ **Scripts:** Son los archivos JavaScript que se encargan de realizar la visualización tridimensional y enviar las peticiones del cliente hacia el servidor vía Ajax.

En el servidor:

- ⇒ **ajaxcontroller.php:** Es el encargado de controlar las peticiones que envía el cliente.
- ⇒ **Dataset.php, DatasetDICOM.php, DatasetRAW.php:** Son las clases encargadas de trabajar con las imágenes médicas.
- ⇒ **BaseXML.php:** Es la clase que maneja los archivos XML.

4.2.2 Diagrama de despliegue

En el siguiente diagrama se muestra la configuración de hardware del sistema, así como los nodos físicos por los que está compuesto y sus respectivas conexiones. El sistema está constituido siguiendo la arquitectura cliente servidor, donde se interactúa con el usuario a través de una estación de trabajo cliente desde donde se solicita una determinada petición al servidor. Del lado del servidor estará funcionando el servidor Web Apache en su versión 2.0 o superior y PHP 5.0.0 o superior. La comunicación con el nodo del cliente que es donde se van a mostrar las interfaces para interactuar con el sistema se va a realizar por medio del protocolo HTTP. Del lado del cliente se necesita tener instalado uno de los sistemas operativos más utilizados actualmente tales como Microsoft Windows, Linux o Mac. Para acceder a la aplicación el usuario podrá utilizar los navegadores Mozilla Firefox 4.0 o superior, Google Chrome 12 o superior, u Opera 11 o superior (ver Figura 19).

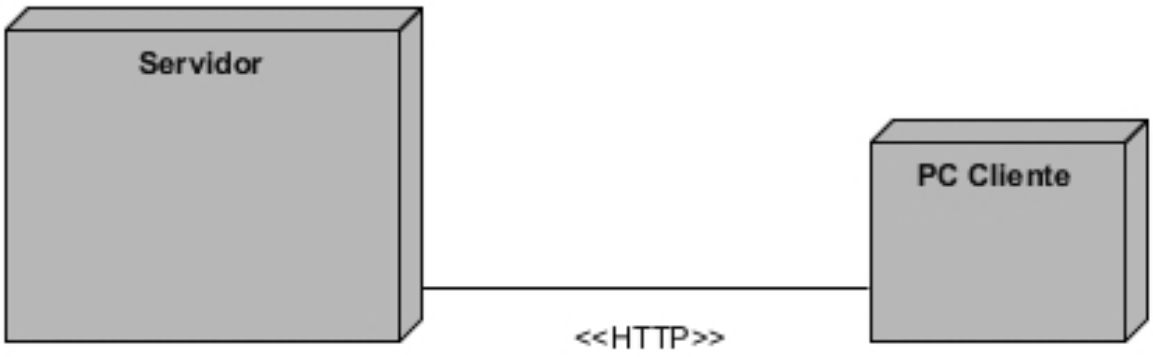

Figura 19. Diagrama de despliegue.

4.2.3 Implementación del Caso de Uso Cargar Modelo

Figura 20. Implementación del Caso de Uso Cargar Modelo

4.2.4 Implementación del Caso de Uso Visualizar Modelo

Figura 21. Implementación del Caso de Uso Visualizar Modelo

4.3. Validación de los resultados

Esta sección muestra los principales resultados alcanzados con la aplicación Web desarrollada, los que pueden ser tomados como punto de partida para realizar comparaciones con otras aplicaciones similares. El tipo de prueba utilizada fue eficiencia, esta sirve para medir el nivel de rendimiento de la aplicación propuesta. Las pruebas que se realizaron van dirigidas a las dos principales funcionalidades de la aplicación: la visualización tridimensional de las imágenes médicas utilizando el algoritmo propuesto y la construcción de la textura 2D a partir de los archivos cargados. La primera funcionalidad se realiza del lado del cliente y la segunda en el servidor. A continuación se muestran las prestaciones de la computadora a utilizar para realizar las pruebas:

⇒ Procesador Intel Pentium 4 a una frecuencia de 3.0 GHz, 2 GB de memoria RAM y tarjeta gráfica NVIDIA GForce 9800 GT con 512 MB de memoria dedicada a video.

Para la realización de estas pruebas se utilizaron dos imágenes tridimensionales nombradas Moléculas y Pie, sus dimensiones en (x, y, z) son (64, 64, 64) y (256, 256, 256) respectivamente. Cada una de estas imágenes guardan un valor de 8 bits para sus vóxeles. Además se utilizarán los Sistemas Operativos Microsoft Windows y Linux, en cada uno de ellos la aplicación se probará en los navegadores Web, Mozilla Firefox 12 y Google Chrome en su versión 17.0.

4.3.1 Parámetros a medir

Con el objetivo de evaluar los resultados se tendrá en cuenta el rendimiento del sistema a la hora de realizar las funcionalidades mencionadas anteriormente, en cada Sistema Operativo utilizando los navegadores descritos en la sección anterior.

Para la primera funcionalidad que se realizará del lado del cliente se comprobará para cada imagen la cantidad de cuadros que es capaz de representar en un segundo (*fps*), el consumo de memoria RAM y el uso en porcientos del CPU y la GPU. En la segunda funcionalidad que se realizará en el servidor, se evalúa el tiempo en segundos que demora la aplicación en construir la textura 2D, el consumo de la memoria RAM y el uso en porcientos del CPU.

4.3.2 Rendimiento

Las tablas que se muestran a continuación muestran los resultados que se obtuvieron después de ejecutar la aplicación en los sistemas operativos Windows y Linux.

Tabla 12. Pruebas de rendimiento del lado del cliente usando el sistema operativo Windows.

Navegador Web	**Estudio**							
	Molécula				Pie			
	FPS	RAM (MB)	CPU (%)	GPU (%)	FPS	RAM (MB)	CPU (%)	GPU (%)
Mozilla Firefox	30-32	248	50	30-35	30-35	250	50	30-35
Google Chrome	55-60	130	10	30-35	50-60	128	15	30-35

Tabla 13. Pruebas de rendimiento del lado del cliente usando el sistema operativo Linux.

Navegador Web	Estudio							
	Molécula				Pie			
	FPS	RAM (MB)	CPU (%)	GPU (%)	FPS	RAM (MB)	CPU (%)	GPU (%)
Mozilla Firefox	25-30	135	30	35	35-40	140	30	35
Google Chrome	50-60	109	25	35	50-60	110	30	35

En las tablas se aprecia la diferencia en rendimiento que generan los navegadores Web utilizados en las pruebas, donde Google Chrome muestra un rendimiento estable en ambos sistemas operativos, comparado con Mozilla Firefox que mejora el uso del CPU cuando se ejecuta en Linux.

Las pruebas de rendimiento del lado del servidor se muestran en las siguientes tablas (ver Tabla 14 y Tabla 15).

Tabla 14. Pruebas de rendimiento del lado del servidor usando el sistema operativo Windows.

Estudio	Dimensiones (x,y,z)	Tiempo (s)	Consumo de Memoria RAM (MB)	Uso del CPU (%)
Molécula	(64,64,64)	10	106	20
Pie	(256,256,256)	60	109	20

Tabla 15. Pruebas de rendimiento del lado del servidor usando el sistema operativo Linux.

Estudio	Dimensiones (x,y,z)	Tiempo (s)	Consumo de Memoria RAM (MB)	Uso del CPU (%)
Molécula	(64,64,64)	7	105	25
Pie	(256,256,256)	50-55	108	25

Con volúmenes de datos relativamente pequeños, el tiempo que demora en construir la textura 2D es bastante aceptable, sin embargo cuando las dimensiones del volumen crecen el tiempo de demora es bastante notable. Destacar que este tiempo puede disminuir si se utiliza un servidor con un procesador superior al que se utilizó en la realización de las pruebas.

CONCLUSIONES

Con la realización de este trabajo se hizo un estudio de las tendencias actuales de la visualización tridimensional de imágenes médicas sobre la plataforma Web, lo que ayudo a dar cumplimiento a los objetivos propuestos en esta investigación.

- ⇒ Se implementó el algoritmo de visualización de volumen Raycasting haciendo uso de la tecnología WebGL, lográndose la visualización tridimensional de imágenes médicas en la Web, a partir de los formatos DICOM y RAW.
- ⇒ Se realizaron pruebas a la aplicación desarrollada para garantizar que su desempeño cumpla con los objetivos trazados.

RECOMENDACIONES

A la aplicación desarrollada 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 los planos de corte para analizar el modelo desde los planos axial, sagital y coronal.
- ⇒ Incorporar técnicas de optimización, manejo de memoria y mejoras en la calidad de la imagen, ejemplo, una forma simple de saltar o esquivar los espacios en blanco y el refinamiento de los puntos de intersección.

REFERENCIAS BIBLIOGRÁFICAS

1. Monografias.com. [En línea] Monografias.com S.A., 2007. http://www.monografias.com/trabajos55/visualizacion-cientifica/visualizacion-cientifica.shtml.

2. Sabia. [En línea] Noviembre de 2010. http://sabia.tic.udc.es/gc/Contenidos adicionales/trabajos/Tutoriales/TutorialWebGL/index.htm.

3. **Fernandez, Fernando.** el cazador de TIC. *el cazador de TIC.* [En línea] http://cazadordetics.blogspot.com/.

4. **Ortega, Lucio Marcelo Quispe.** De la Web 1.0 a la Web 4.0 . *Consultora de Servicios Informáticos DEVIAN S.R.L.* [En línea] 2011. http://www.consultora-devian.net/inicio/noticias.

5. **Villamil, Jenaro.** La Web-3-0 y la web semantica. *Jenaro Villamil.* [En línea] 2011. http://jenarovillamil.wordpress.com/2011/11/03/la-web-3-0-y-la-web-semantica/.

6. *Monografias.com.* [En línea] Octubre de 2003. http://www.monografias.com/trabajos5/laweb/laweb.shtml.

7. **Aldana, Luis Antonio Fernández.** Introducción a Java3D. [En línea] Enero de 2007. http://www.monografias.com/trabajos43/java-tres-d/java-tres-d.shtml.

8. **Eduardo Antuña Díez, Aitor Díaz Solares, Daniel González Losada.** *Desarrollo en WebGL.* 2011.

9. La Capilla Sixtina en 3D. *InfoNucleo.* [En línea] 2010. http://www.infonucleo.com/2010/03/24/la-capilla-sixtina-en-3d/.

10. **Bernhard Preim, Dirk Bartz.** *VISUALIZATION IN MEDICINE.* 2007.

11. **Crespo, José.** Procesamiento y Visualización de Imágenes 3D. [En línea] 2010. http://www.saber.ula.ve/redtelemedicina/TallerTelemedicina/ponencia-j_crespo.html.

12. **Cesar J. Acuña, Esperanza Marcos, Valeria de Castro, Juan A. Hernández, Marcos López Sanz.** *Gestión de Imágenes Médicas a Través de la Web.* 2010.

13. **Free Online Medical Dictionary.** Medical Dictionary. [En línea] http://medical-dictionary.thefreedictionary.com/virtual+endoscopy.

14. Google Body Browser vuelve: Zygote Body. *Cual es mi IP.* [En línea] 2012. http://cualesmiip.com.es/es/noticias/Google_Body_Browser_vuelve_Zygote_Body/17513.html.

15. BodyMaps, un atlas anatómico 3-D. [En línea] 2011. http://medgadget.es/2011/06/bodymaps_un_atlas_anatomico_3d.html.

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

17. **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.

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

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

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

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

22. **John Congote, Alvaro Segura,Jorge Posada,Oscar Ruiz,Luis Kabongo,Aitor Moreno.** *Interactive visualization of volumetric data with WebGL in real-time.* 2011.

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

24. **López, Xavier Farré.** *Rich Internet Applications.* 2005.

25. **Fain, Yakov, Rasputnis, Victor y Tartakovsky, Anatole.** *Aplicaciones Ricas de Internet con Adobe Flex y Java.* 2007.

26. **James Garrett, Jesse.** Ajax: Un nuevo acercamiento a las aplicaciones WEB. *Ajax: Un nuevo acercamiento a las aplicaciones WEB.* [En línea] 2005. http://www.willydev.net/Descargas/WillyDev_AJAX.pdf..

BIBLIOGRAFÍA

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

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

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

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

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

AlternativaPlatform-Alternativa3D. Disponible en: http://alternativaplatform.com/ alternativa3d/

Van Der Henst S. Christian ¿Qué es la Web 2.0?. Disponible en: http://www.maestrosdelweb.com/

Torossi, Gustavo. El Proceso Unificado de Desarrollo de Software. 2005.

Brutzman, Don and Daly, Leonard. X3D: Extensible 3D Graphics for WEB authors. San Francisco: Elsevier Inc, 2007.

Edwards, Lin. Superior 3D Graphics for the WEB a Step Closer. 2009.

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

Macías, Jiménez. Escenarios virtuales WEB3D: Simulación con VRML, JAVA3D y X3D. 2004.

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

GLOSARIO DE TÉRMINOS

Ajax: *Asynchronous JavaScript And XML* no es una tecnología porque es la unión de un conjunto de tecnologías para el desarrollo Web que se ejecutan en el cliente.

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

Apache: Es el servidor que se encargado de resolver las peticiones de los clientes de páginas Web.

Aplicación Web: Es una aplicación informática que los usuarios utilizan accediendo a un servidor Web a través de un navegador o browser. Estas son muy populares debido a la habilidad para actualizar y mantener la información manipulada sin distribuir e instalar el software en miles de potenciales clientes.

Away3D: Es un framework de código abierto basado en ActionScript, motor de renderizado 3D para el desarrollo de aplicaciones flash.

DICOM: *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.

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

Google O3D: Es una API de código libre basada en JavaScript para crear mundos 3D.

Hardware: Componentes físicos de una computadora o de una red (a diferencia de los programas o elementos lógicos que los hacen funcionar).

HTML: Lenguaje de Marcas de Hipertexto, es un lenguaje para elaborar páginas Web.

IDE: Entorno integrado de desarrollo.

Interfaz: Es uno de los componentes más importantes de cualquier sistema computacional, funciona como el vínculo o comunicación entre el humano y la máquina.

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

JavaScript: Es un lenguaje interpretado que está del lado del cliente.

Java 3D: Es un API (Application Programming Interface) orientado a objetos para el lenguaje Java, que permite la confección de aplicaciones tridimensionales.

Modelo 3D: Es la unión de un conjunto de mallas, es como tal el resultado de todo el proceso de reconstrucción en la memoria.

PaperVision3D: Es un motor de renderizado 3D en tiempo real, hecho en ActionScript 3.

PHP: El lenguaje PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de script interpretado utilizado para la generación de páginas Web dinámicas, embebido en páginas HTML y ejecutado en el servidor.

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

Plugins*:* Aditamento para agregar a un equipo.

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.

RUP: Proceso de desarrollo de software, metodología utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Glosario de Términos Autores: Leandro Raúl Ramírez Batista, Reinier Talancón Díaz 128

Sandy3D: Librería open-source desarrollada en ActionScript para la visualización 3D en el entorno de flash.

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

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.

UML: “Unified Modelling Language”, lenguaje de modelado gráfico que permite especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.

Unity3D: Es una potentísima herramienta de creación de entornos 3D, aplicaciones interactivas y aplicaciones Web.

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.

VRML: Virtual Reality Modeling Language es un lenguaje de especificaciones, tridimensional e interactivo, orientado a la modelación y la visualización de objetos, situaciones y mundos virtuales en Red.

WEBGL: Es una tecnología de visualización 3D que combina JavaScript, OpenGL y la etiqueta canvas, incorporada a HTML5.

W3C: Consorcio World Wide Web es una comunidad internacional y la misión es guiar la Web hacia su máximo potencial.

XML: Extensible Markup Language es un perfil de aplicación o forma restringida de (SGML).

X3D: Extensible Tridimensional es un estándar abierto XML que tiene un formato de archivo 3D que permite la creación y transmisión de datos 3D entre distintas aplicaciones.