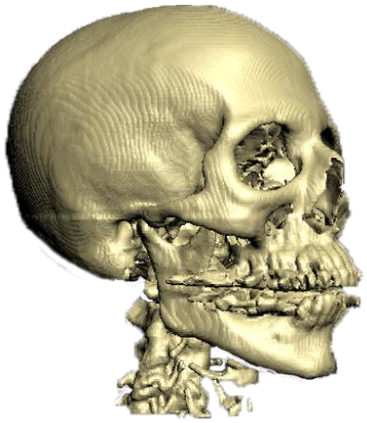
Universidad de las Ciencias Informáticas
Facultad 5

DECIMACIÓN DE GEOMETRÍAS OBTENIDAS POR MARCHING CUBE



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

Autor: Frank Rufino Nápoles

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

Marzo 2011

DECLARACIÓN DE AUTORÍA

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

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

_____

Frank Rufino Nápoles.

_____ _____

M.Sc. Osvaldo Pereira Barzaga. Ing. Ernesto Carrasco de la Torre.

DATOS DE CONTACTO

Tutor: M.Sc. Osvaldo Pereira Barzaga

Edad: 26

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título(s): Ingeniero en Ciencias Informáticas y Máster en Informática Aplicada.

Categoría Docente: Instructor.

E-mail: opereira@uci.cu

Graduado de la UCI, con cuatro años de experiencia en el tema de la Gráfica Computacional y líder de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

Co-tutor: Ing. Ernesto Carrasco De la Torre

Edad: 24

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título(s): Ingeniero en Ciencias Informáticas.

Categoría Docente: -

E-mail: ecarrasco@uci.cu

Graduado de la UCI, con dos años de experiencia en el tema de la Gráfica Computacional y profesor de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

DEDICATORIA

A la memoria de mi querida y adorada abuela

Por todo el cariño y el afecto que siempre me dio, por estar siempre a mi lado en los momentos que siempre la necesité.

A mi querido y adorado abuelo.

Por enseñarme que las cosas buenas de la vida siempre requieren de grandes esfuerzos y por el amor incomparable que siempre me dedicó.

A mi querida y adorada mamá.

A esa persona que me trajo al mundo y que siempre la llevo en mi corazón.

AGRADECIMIENTOS

A la memoria de mi querida y adorada abuela

A la persona que me dedicó toda su vida a formarme como una persona de bien y que siempre supiera identificar lo bueno de lo malo así como hacerme un hombre de bien.

A mi querido y adorado abuelo.

A la persona que de una manera incansable ha sabido ser ejemplo de padre, amigo y compañero, por su persistencia incorruptible ante el estudio y el trabajo.

A mi querida y adorada mamá.

A la persona que siempre me ha brindado su apoyo incondicional en todo momento sabiendo siempre como ser la mejor madre del mundo.

A mi tutor.

A la persona que ha sabido como guiarnos por el camino de la ciencia como todo un profesional. Siempre muy atento y cuya fuerza de voluntad y entusiasmo son casi incomparables.

A mi co-tutor.

A la persona que siempre ha estado atenta para cualquier ayuda que pueda brindar, siempre muy preocupado por el estado de las tareas investigativas.

A mis compañeros de trabajo.

A ese aguerrido equipo que siempre está ansioso por querer desarrollar cosas nuevas y por compartir conmigo los momentos buenos y los difíciles también.

A mis compañeros de cuarto y grupo.

A ese grupo entusiasta de compañeros y compañeras que siempre de una forma u otra ha sabido cómo hacer que la estancia en la universidad sea más agradable y divertida.

RESUMEN

Con el desarrollo de las tecnologías de visualización (tarjetas gráficas) en los últimos años la visualización científica se ha insertado en muchos campos de las ciencias; con el objetivo fundamental de lograr representaciones tridimensionales de grandes volúmenes de datos. Dentro de las técnicas más aplicadas se encuentra la Visualización Directa de Volumen, la cual permite obtener una representación geométrica de la superficie de una región de interés. Uno de los algoritmos más empleados dentro de dicha técnica es el Marching Cubes, el cual a partir de un isovalor genera la descripción geométrica de la superficie. El problema fundamental de dicho algoritmo es que para volúmenes de datos de alta resolución obtiene representaciones geométricas con un elevado número de triángulos; elemento que atenta contra la representación de la geometría en tiempo real. Es por ello que en este trabajo proponemos un algoritmo de decimación de la geometría que reduce el elevado número de polígonos de la malla inicial usando como criterio matemático la desviación normal promedio; obteniendo como resultado un balance entre la calidad visual de los modelos y el número de polígonos.

Palabras clave: Decimación, Desviación Normal Promedio, Isovalor.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Numerosas aplicaciones de gráfico por computadoras y otros campos relacionados, se basan en modelos de superficie poligonal para la visualización y simulación de fenómenos. Tradicionalmente los algoritmos para la reconstrucción de modelos superficiales empleados en las aplicaciones de visualización científica obtienen un único nivel de detalles de la geometría correspondiente a dicho modelo. Sin embargo, para diversos contextos este único nivel de detalles puede ser poco adecuado; ya que en todas las aplicaciones existe un equilibrio entre la precisión con la que una superficie es modelada y el tiempo requerido para procesarla; por lo que para lograr tiempos aceptables de procesamiento a menudo se debe sustituir el modelo original por aproximaciones más simples.

Desde su surgimiento en 1776, la simplificación poligonal se ha convertido en una poderosa herramienta que permite obtener diferentes niveles de detalles de un modelo geométrico; los cuales pueden ser posteriormente adaptados a las necesidades individuales de cada aplicación.

En la medicina, el uso de herramientas de visualización tridimensional de superficies anatómicas para agilizar y mejorar los procesos de diagnóstico clínico, han teniendo una gran aceptación tanto por los profesionales de la salud como por los pacientes. Este tipo de herramientas ha ido perfeccionándose con la introducción de técnicas de Visualización Científica que realizan la reconstrucción tridimensional de un paciente a partir de las imágenes adquiridas de modalidades como Tomografía y Resonancia Magnética.

En nuestro país, desde los primeros momentos del triunfo de la Revolución y como uno de los puntos a cumplir del programa del Moncada, la salud pública constituye un elemento que distingue el proceso revolucionario. Innumerables son los logros alcanzados por la medicina cubana en estos años de Revolución y los esfuerzos realizados por nuestro Estado Socialista para mantener una atención sanitaria a la altura de países desarrollados.

Muchos centros de investigación han dedicado parte de su trabajo a crear equipos computarizados de apoyo a la actividad médica. Un ejemplo fehaciente de esto es el Instituto Central de Investigaciones Digitales (I.C.I.D), creador de un número importante de equipos de la más alta tecnología, utilizando para ello las computadoras: el CardioCid,

el NeuroCid y el S.U.M.A (Sistema Ultra Micro Analítico) -por solo mencionar algunos ejemplos- los cuales constituyen aportes significativos al Sistema Nacional de Salud.

Actualmente en el Centro de Informática Industrial (CEDIN) de la Facultad 5 de la Universidad de la Ciencias Informática (UCI) la línea de desarrollo de visualización científica (VISMEDIC) está trabajando para desarrollar aplicaciones que emplean técnicas de visualización médica tridimensional que apoyen a los especialistas y radiólogos en las realizaciones de los diagnósticos; en las cuales se emplean algoritmos de reconstrucción de superficies como el Marching Cube.

La cantidad de triángulos de los modelos obtenidos mediante Marching Cube en volúmenes de datos de elevada resolución, son una limitante para la correcta visualización de los mismos en tiempo real. A partir de la problemática anteriormente expuesta se plantea el problema científico de este trabajo.

Problema Científico: ¿Cómo eliminar el exceso de primitivas redundantes en los modelos exportados por el algoritmo Marching Cube para mejorar el nivel interactividad y la visualización en tiempo real?

A raíz de esto el trabajo toma como **objeto de investigación** los algoritmos de simplificación de mallas poligonales y dentro de esta extensa gama de algoritmos y técnicas para el tratamiento de mallas poligonales se propone como **campo de acción** los algoritmos de simplificación de mallas basados en eliminación de geometrías, más conocidos como algoritmos de decimación.

El **objetivo** de esta investigación es elaborar un módulo que permita la simplificación de modelos poligonales obtenidos por Marching Cube eliminado el exceso de primitivas geométricas.

Este trabajo defiende la siguiente idea: la simplificación de los modelos poligonales obtenidos por Marching Cube, basándose en los principios de la decimacion, mediante la utilización del módulo de simplificación, permitirá obtener un balance entre la calidad de visualización y el rendimiento necesario para la interacción con los mismos.

Para dar cumplimiento a los objetivos planteados en este trabajo se necesita un grupo de tareas investigativas a las cuales se hará referencia:

1. Elaboración del marco teórico a partir del estado del arte existente sobre simplificación de mallas poligonales.

2. Selección de alguna de las distintas técnicas de decimación.
3. Selección de la estructura de datos más conveniente para la implementación.
4. Implementación del algoritmo seleccionado.
5. Validación de los resultados a través de los modelos exportados en el proyecto VISMEDIC, comprobando calidad y rendimiento.

Además, para todo el proceso de investigación y elaboración de este trabajo se tomará en cuenta la utilización de varios métodos científicos de investigación como:

Histórico – Lógico: método teórico mediante el cual se constatará como ha sido la trayectoria histórica real, la evolución y desarrollo de los aspectos principales de la investigación como: el tratamiento de las mallas y la simplificación de primitivas.

Analítico – Sintético: este método teórico será utilizado en la investigación para buscar la información más actualizada de lo que existe en el mundo acerca de las principales técnicas y algoritmos de simplificación de mallas poligonales así como el tratamiento sobre las superficies exportadas por algoritmos como Marching Cube.

Entrevista: método empírico para obtener información acerca de la simplificación poligonal y las características de las distintas técnicas para centralizar el resto de la investigación.

Experimento: método empírico mediante el cual, se realizarán experimentos y pruebas de los principales algoritmos y técnicas de simplificación, para examinar los resultados en busca de escoger el más adecuado.

Para tener un conocimiento general de lo que se abordará en este trabajo, se muestra a continuación una síntesis de cada capítulo:

Capítulo 1: Fundamentación Teórica.

En este capítulo, se indicaran las bases teóricas fundamentales de las técnicas y algoritmos fundamentales de simplificación sobre mallas poligonales. Así como los tipos de superficies sobre las cuales operan cada uno de estos algoritmos y el tipo de trabajo a realizar en dependencia de las necesidades, ya sea de refinamiento o de decimación sobre la geometría seleccionada.

Capítulo 2: Solución Propuesta.

En esta sección se propone una solución técnica del problema, teniendo en cuenta las técnicas y algoritmos seleccionados, las adaptaciones y aportes realizados por los autores.

Capítulo 3: Características del Sistema.

En este capítulo se definirán las reglas del negocio así como el modelo de dominio. Se expondrá la captura de requisitos. Se elaborará el Modelo de Casos de Uso que estará formado por los actores, el Diagrama de Casos de Uso y la descripción textual de cada Caso de Uso. Finalmente se elaborarán los diagramas de clases del diseño y los diagramas de secuencia del diseño.

Capítulo 4: Resultados y Validación.

En este capítulo se definirán los detalles más específicos de la implementación y se harán las pruebas para validar los resultados de la investigación realizada. Se modelará el diagrama de componentes para la implementación y se mostrarán los resultados de las pruebas de tiempo y calidad de los modelos simplificados.

Glosario de Términos.

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

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se abordarán los principales elementos teóricos de las técnicas y algoritmos para la decimación geométrica de acuerdo a la topología del mallado que describe la superficie de una región de interés dentro de un volumen de datos médicos. Se concluye el capítulo con una selección de las estructura de datos para un almacenamiento eficiente de la geometría que permita durante el procesamiento de la misma un uso eficiente de los recursos de hardware y tiempos de respuestas adecuados del algoritmo de decimación seleccionado.

1.1. Simplificación de Geometrías.

Michael Garland, en su tesis doctoral [1] define simplifición poligonal como el proceso de obtener de forma automática a partir de una malla inicial M una representación M' con un menor número de polígonos que la geometría inicial pero con una calidad visual lo más próxima posible al modelo inicial M.

La simplificación poligonal es un concepto muy empleado en aplicaciones que requieren visualizaciones de grandes volúmenes de datos en tiempo real, para el cual se definen niveles de detalles a partir de la distancia que se encuentre el modelo del observador [1].

A continuación caracterizaremos las técnicas y algoritmos para la simplificación poligonal de acuerdo a la taxonomía de la geometría de la cual se desea obtener una representación aproximada de la original.

1.2. Tipos de Taxonomías.

Los tipos de topologías de mallas triangulares más empleados en aplicaciones de realidad virtual se clasifican en tres grandes grupo:

1.2.1. Campos de Alturas y Superficies Paramétricas.

Los campos de alturas y superficies paramétricas son las más sencillas clases de superficies (ver Figura 1). Dentro de esta clase de superficies, se agrupan los métodos de procesamiento de las mismas en las siguientes seis sub-clases: los métodos de redes

regulares, los métodos jerárquicos de subdivisión, métodos de ventajas, los métodos de refinamiento, los métodos de decimación, y los métodos óptimos [2].

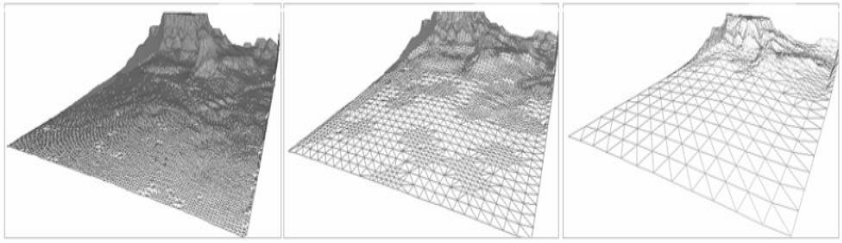


Figura 1. Campos de Alturas.

1.2.2. Superficies Múltiples.

Según Henle en su trabajo [3] una superficie se define como el conjunto de primitivas geométricas en una pequeña área circundante sobre la vecindad sombrilla [2] de todos los triángulos que componen la malla poligonal. **Figura 2**.

Estas superficies pueden tener género arbitrario y son más difíciles de simplificar que los campos de altura o superficies paramétricas porque no tienen una parametrización natural en 2D. Este tipo de superficie agrupa los algoritmos para su procesamiento en dos grandes grupos: los métodos de refinamiento y los métodos de decimación [2].

Figura 2. Mallas Múltiples.

1.2.3. Superficies No Múltiples.

La clase más general de las superficies son la superficie no-múltiples, que permite que tres o más triángulos compartan un lado y permiten intercepciones arbitrarias entre los polígonos que conforman la geometría. Pocos algoritmos de simplificación de superficie pueden manejar modelos geométricos de esta categoría y por la generalidad de geometría que procesan dichos algoritmos difíciles de generalizar por lo que no se

registra en la bibliografía sobre el tema una agrupación de los mismo como se hacen para las topologías de campos de alturas o superficies múltiples [2].

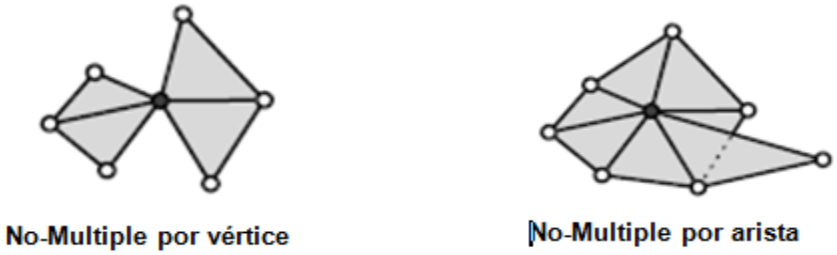


Figura 3. Mallas No Múltiples.

1.3. Caracterización de los Algoritmos de Decimación.

Como se enuncio en el Epígrafe 1.2 una vez clasificada la taxonomía de la geometría se procede a la selección de los algoritmos de procesamiento de acuerdo al objetivo que se persiga. En [2] se encuentran clasificados y detallados cada uno de los métodos de procesamiento según sus propósitos y de acuerdo a la taxonomía geométrica sobre la que operan. A continuación caracterizaremos los algoritmos de decimación que mayor aceptación han tenido por la comunidad científica según la bibliografía consultada [1, 2].

1.3.1. Algoritmos de Decimación para Campos de Alturas.

Algoritmo de Lee.

Un método heurístico para la simplificación de terrenos fue propuesto por Lee [5] en el cual en cada iteración del algoritmo se elimina un vértice. La idea fundamental de dicho algoritmo es formar cuadrículas de dimensiones 2 x 2 y generar una triangulación de la superficie donde cada cuadrícula se divida en dos triángulos [5]. Cuando Lee comparó su algoritmo con el método de Chen y Guevara [6] y con el método de triangulación ternaria de Floriani [6], demostró que su método arrojo mejores resultados visuales a pero de mayor costo computacional.

Algoritmo de Scarlatos – Pavlidis.

Scarlatos y Pavlidis desarrollaron un método de decimación en 1992 [2]. Su algoritmo toma una triangulación inicial y aplica tres pasos: reducción de triángulos con alta curvatura, la fusión de triángulos adyacentes coplanares, y el intercambio de los bordes para mejorar la forma y ajuste de los triángulos. Aunque en las pruebas expuestas para la

mayoría de los casos el método redujo el número de triángulos, pero no se dieron imágenes de las mallas resultantes, por lo que es difícil compararlo en cuanto a calidad visual de la triangulación obtenida con otros métodos.

Posteriormente, en 1993, Scarlatos desarrolló también un método de decimación de vértices para la construcción de triangulaciones jerárquicas [5]. El método comienza con una triangulación inicial y, para generar cada nivel de la jerarquía, calcula la "importancia" de cada vértice y elimina los vértices en orden creciente de importancia hasta que no se puede eliminar más. Donde la “importancia” es una función (sin especificar) del error entre un vértice y un promedio ponderado de sus vecinos. Sobre dicho algoritmo Scarlatos informó una duración de 7.75 minutos para construir una triangulación jerarquía completa de 900 vértices resultantes en un microprocesador VAX 8530.

Algoritmo de Hughes – Lastra – Sajonia.

El algoritmo de simplificación descrito por Hughes, Lastra, y Sajonia [7] apunta hacia la simplificación de mallas de iluminación global resultantes de los sistemas de radiosidad. En consecuencia, el algoritmo debe simplificar tanto la geometría de la malla como el color de los valores asociados a cada vértice de malla. En versión posterior del algoritmo optaron por una combinación de decimación de vértice locales y envolturas de simplificación como en [8]. Demostraron además que la selección al azar de los vértices en vez de su elección atendiendo al criterio del mayor error para su eliminación obtenía mejores resultados ya que se obtenían mallas más uniformes, lo que permite una representación de mayor calidad visual.

1.3.2. Algoritmos de Decimación para Superficies Múltiples.

Los métodos de decimación para geometrías con topologías de superficies múltiples se clasifican de acuerdo al elemento de la geometría que eliminan de la siguiente forma:

- Los métodos de ***decimación de vértice****:* eliminan un vértice y se retriangula su vecindad.
- Los métodos de ***decimación de borde****:* eliminan un borde y dos triángulos, y fusionan dos vértices.
- Los métodos de ***decimación de caras****:* eliminan un triángulo y tres bordes, fusionan tres vértices, y se retriangula su vecindad.

- Los métodos de ***decimación de parche****:* eliminan varios triángulos adyacentes y se retriangula su frontera.

Diversas variantes de este enfoque de decimación se han propuesto, de las cuales algunas se mencionan a continuación:

Algoritmo de Kalvin.

Kalvin desarrolló un método de dos fases para crear modelos de superficies a partir de datos médicos [11]. La primera fase se aproxima a una superficie con polígonos pequeños utilizando un algoritmo similar a Marching Cube [19], y la segunda a continuación, hace una decimación de parche en el modelo mediante la fusión de rectángulos adyacentes coplanares. Dado que sólo fusiona caras coplanares precisas, el método no permite el control sobre el grado de simplificación, por lo que es bastante limitado.

Algoritmo de Schroeder – Zarge Lorensen.

Schroeder y Zarge Lorensen desarrollaron un algoritmo general de decimación de vértices para usar principalmente en visualización científica [2]. Su método toma una superficie triangular como entrada e itera varias veces hasta que se alcanza el error deseado. En cada paso, todos los vértices que no están en un límite de error por debajo del umbral se suprimen, y sus polígonos circundantes son retriangulados (ver Figura 4). El error en un vértice es la distancia desde el punto al plano aproximado de los vértices circundantes [2]. Es importante destacar que en este algoritmo en particular el error se mide con respecto a la aproximación anterior, no en relación con la triangulación original. En los resultados expuestos por los autores se obtuvieron simplificaciones de 40 000 vértices para modelos originales de hasta 1 700 000 vértices, en tiempos aproximados a los 14 minutos sobre un procesador R4000 [15]. En relación con el método de Lee, la técnica de Schroeder y Zarge Lorensen es más general, puesto que no se limita a los campos de altura y elimina múltiples vértices por pasada. En consecuencia, es más rápido, pero probablemente tiene menor calidad.

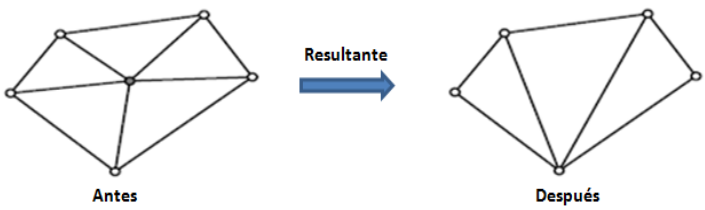


Figura 4. Colapso de Vértices.

Algoritmo de Hinker – Hansen.

Hinker y Hansen desarrollaron un algoritmo de decimación de parche para su uso en visualización científica [7]. Se trata de un método de una pasada que primero encuentra manchas de triángulos con los vectores normales casi paralelos, y retriangula cada parche. El método tiene complejidad $\boldsymbol{O(n\,log\,n)}$ en la práctica. Para modelos cercanos a los 510 000 vértices obtuvieron simplificaciones de hasta 321 000 vértices en tiempos aproximados de 9 minutos sobre microprocesadores CM-5. Este algoritmo es ineficaz frente a las superficies de alta curvatura [7].

Algoritmo de Gueziec.

Gueziec desarrolló un método de simplificación para superficies múltiples orientables que emplea decimación de borde [2]. El algoritmo propuesto cumple cuatro condiciones fundamentales: la topología de la superficie se mantiene, las normales de las caras modificadas cambian poco, los nuevos triángulos están formados correctamente y el error para el nuevo vértice está por debajo del umbral de error máximo establecido. En su artículo [2], Gueziec informa tiempos de ejecución del algoritmo de hasta 10 minutos para simplificar modelos cercanos a los 90 000 vértices, obteniendo simplificaciones de hasta 5 000 vértices en procesadores RS6000 de IBM 350.

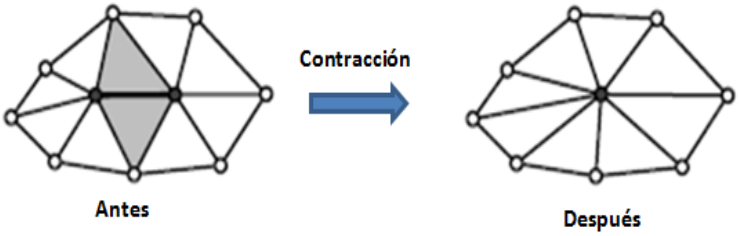


Figura 5. Colapso de Arista.

Algoritmo de Gourdon.

Gourdon exploró un método de simplificación de mallas resultantes de superficies orientables a partir de la reconstrucción de la superficie [2]. Su algoritmo se diferencia de casi todos los algoritmos de simplificación, ya que no asume la malla de superficie como una triangulación. El algoritmo está diseñado para preservar la característica Euler del modelo, la característica de Euler de un modelo se define como $\boldsymbol{X = F - E + V}$, donde $\boldsymbol{F}$, $\boldsymbol{E}$ y $\boldsymbol{V}$, son respectivamente, el número de caras, aristas y vértices, lo que implica que la topología se conserva. El algoritmo iterativo elimina los bordes sobre la base de un criterio

de curvatura no especificada. Debido a que el algoritmo admite facetas no triangulares; a raíz de esto la simplificación obtenida requiere de un paso de "regularización"; con el cual se intenta mejorar la malla resultante [2].

Otros algoritmos

Muchas han sido las variantes de algoritmos propuestas al pasar de los años, autores como Klein, Liebich, y Strasser en sus trabajos [2] han propuestos algoritmos de decimación basados en triangulación restringida de Delaunay (CDT por sus siglas en inglés). Otros como Algorri y Schmitt en 1996 desarrollaron un algoritmo para la simplificación de triangulaciones cerradas obtenidas a partir de una reconstrucción de superficie [2].

Es importante destacar que en los últimos años la mayorías de las soluciones propuestas al problema de la decimación de geometría han perdido su capacidad de generalización; ya que los autores, con el objetivo de lograr mejores tiempos de ejecución han particularizados sus soluciones a sus problemas concretos [1].

1.3.3. Algoritmos de Decimación para Superficies No Múltiples.

Como la simplificación de superficies múltiples es un problema poco común no existen algoritmos que se puedan considerar como bases generales dentro de la bibliografía. Los autores con mayores aportes en este campo podemos mencionar a Rossignac y Borrel en un trabajo publicado en el año 1993 [1, 2], Low y Tan en 1997 [1, 2] y Maichel Garland y Heckbert en 1997 [1, 2].

1.4. Estructuras de Datos para el Manejo Eficiente de Geometrías.

Las estructuras de datos pueden en gran medida hacer la diferencia entre una aplicación en tiempo real o una aplicación muy lenta en dependencia de la correcta utilización de las mismas. Para los trabajos de tratamientos de mallas, la selección de la estructura de datos forma parte de la base fundamental durante el desarrollo de cualquiera de los algoritmos anteriormente descritos.

Los trabajos sobre mallas triangulares, como se muestra en los epígrafes anteriores, vienen desarrollándose desde hace muchos años atrás y en la mayoría de ellos se ven reflejadas un grupo de estructuras de datos que facilitan el trabajo de la implementación

así como la rapidez en tiempo de ejecución del algoritmo. Dentro de las más utilizadas se encuentran: Octree, DCELL y Grafo Dual.

1.4.1. Octree.

Un octree es una estructura jerárquica que comienza con un cubo contenedor del conjunto total de los datos. Este cubo se subdivide en 8 cubos más pequeños, con los que se comprueba la intercepción de los elementos de la malla (vértices, aristas o triángulos). Aquellos cubos con los que intercepte algún elemento son nuevamente subdivididos en 8 cubos más pequeños (ver Figura 6). Este proceso se repite hasta que se satisfaga una cierta condición de parada. Esta función de parada puede venir en función de los objetivos finales de la aplicación final. Ejemplos de estas condiciones de paradas podría ser el hecho de que el tamaño del cubo este por debajo de un determinado umbral, que el número de elementos que interceptan con el cubo sea lo suficientemente pequeño o que el nivel de profundidad alcanzado llegue a un máximo predeterminado [9].

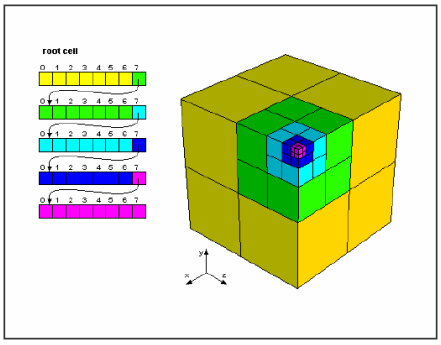


Figura 6. Octree.

1.4.2. DECEL.

El DCEL es una estructura de datos espacial para representar subdivisiones planares que contiene una referencia por cada cara, arista y vértice de una subdivisión. Para cada referencia se almacenan diferentes tipos de informaciones como:

- Información Geométrica: que seria las coordenadas de la geometría.
- Información Topológica: Que no es más que la relación entre vértices y aristas.
- Información Adicional: Que serían las etiquetas asignadas a cada cara.

Cada arista del DCEL tiene una referencia a su arista previa y siguiente, las aristas delimitan las caras de la geometría, pero se debe tener en cuenta que cada arista delimita dos caras. Cada arista está compuesta dos seminaristas gemelas orientadas en sentidos opuestos (ver Figura 7).

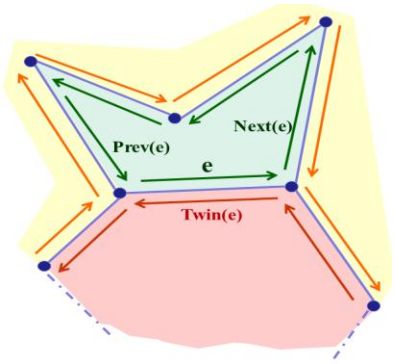


Figura 7. DCEL.

1.4.3. Grafo Dual.

La estructura de datos Grafo Dual fue creada a partir de diversos criterios de diferentes estructuras de datos y la misma está basada sobre la representación de listas entrelazadas guardando la topología de la malla con una peculiaridad muy importante, almacena los índices de los vértices para la creación de cada triángulo, formando una relación única de posiciones que evita recorridos innecesarios [8].

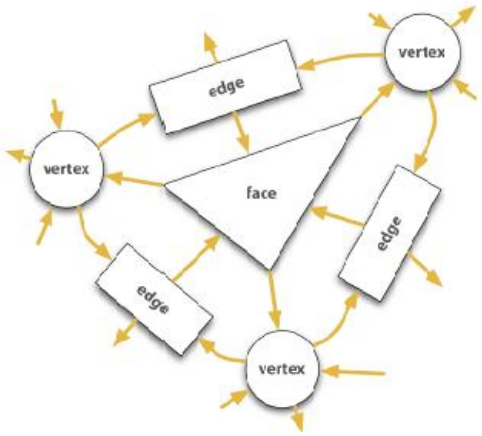


Figura 8. Grafo Dual.

La cantidad de listas utilizadas para la representación de las geometrías viene dada en dependencia del tipo de información que se desee almacenar o la finalidad del trabajo para la cual se utilice. Para representar solo la topología geométrica de cualquier superficie, que es el objetivo de nuestro trabajo, basta almacenar la información de dicha malla en tres listas con la información correspondientes los vértices, aristas y triángulos que conforman dicha geometría. La relación establecida entre estas listas proporcionan la obtención de cualquier información de vecindad o composición geométrica de cada una de las primitivas de dicha malla (ver Figura 8).

1.5. Consideraciones Generales del Capítulo.

El estudio teórico realizado durante este capítulo nos permitió identificar que la geometría obtenida a partir de la ejecución del algoritmo Marching Cubes se encuentra dentro del grupo de geometrías clasificadas como Superficie Múltiples; las cuales soportan algoritmos de decimación de vértices, aristas y caras. Estos algoritmos de decimación, de acuerdo a las características geométricas de la malla obtenida tendrán un rendimiento óptimo sobre una estructura de datos que permita una consulta de vecindad con tiempos de ejecución cercanos a $\boldsymbol{O}(\mathbf{1})$ por lo que asumimos que la estructura de datos idónea en este caso para la gestión de la información geométrica de la misma será el Grafo Dual. Sobre esta base en el siguiente capítulo proponemos un algoritmo de simplificación geométrica basado en la decimacion caras (triángulos).

CAPÍTULO 2. SOLUCIÓN PROPUESTA

Después del estudio teórico realizado en el capítulo anterior, en este capítulo se describe en detalle el algoritmo desarrollado para la simplificación de geometrías generadas a partir de Marching Cubes. El proceso de decimación propuesto estará compuesto por las siguientes etapas:

- Obtención de la geometría de entrada.
- Transformación de la triangulación inicial en un grafo dual.
- Algoritmo de decimación.
- Persistencia y visualización de la geometría simplificada.

2.1. Obtención de la geometría de entrada

Las geometrías de entrada de nuestro trabajo fueron obtenidas a partir del algoritmo Marching Cube utilizado en el proyecto de VISMEDIC, en el cual del volumen de datos correspondiente a imágenes DICOM genera las superficies geométricas de los modelos anatómicos.

Como el volumen inicial del cual se obtuvo la geometría de las superficies de regiones de interés es de elevada resolución, se hace necesario como primera etapa del algoritmo un pre-procesamiento de la misma. Esta etapa cuenta con dos pasos fundamentales:

1. Eliminar las ambigüedades de las caras.
2. Eliminar vértices repetidos.

Una vez que la geometría fue validada correctamente siendo eliminadas las ambigüedades entre puntos y triángulos se estructura la información de la triangulación inicial en un Grafo Dual.

2.2. Transformación de la triangulación en un grafo dual

Luego del análisis realizado en el capítulo anterior sobre las estructuras de datos más utilizadas para almacenamiento de la información correspondiente a una malla triangular, se escogió el Grafo Dual para la realización del presente trabajo debido a la facilidad que

brinda el mismo para realizar las consultas necesarias durante la implementación del algoritmo y la optimización de los tiempos de ejecución de dichas consultas.

El seudocódigo del proceso de construcción del grafo se describe a continuación:

Algoritmo 1:

Entradas:

$Lista\ de\ Vértices$: $\boldsymbol{P} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}, \ldots, \boldsymbol{p_n}\}$.

$Lista\ de\ Triángulos$: $\boldsymbol{T} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \boldsymbol{T_3}, \ldots, \boldsymbol{T_n}\}$ *Donde* $\boldsymbol{T_i} = \{\boldsymbol{p_{3i}}, \boldsymbol{p_{3i+1}}, \boldsymbol{p_{3i+2}}\}$ *y* $\boldsymbol{p_{3i}}, \boldsymbol{p_{3i+1}}, \boldsymbol{p_{3i+2}} \in \boldsymbol{P}$.

Salida:

$Grafo\ Dual$: $G(T, P, A)$ *Donde* T *es la lista de triángulos,* P *la lista de vértices y* A *la lista de aristas.*

Para Todo T_i en T **hacer**:

Se crean y se adicionan las aristas pertenecientes a cada triangulo T_i en un arreglo *A* actualizando la información de vecindad para cada elemento.

fin

Para la creación y la adición de cada una de las aristas de la geometría y la actualización de cada elemento de la geometría se procede como se muestra a continuación en el seudo-código del Algoritmo 2.

Algoritmo 2:

Entradas:

$Lista\ de\ Vértices$: $\boldsymbol{P} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}, \ldots, \boldsymbol{p_n}\}$.

$Lista\ de\ Triángulos$: $\boldsymbol{T} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \boldsymbol{T_3}, \ldots, \boldsymbol{T_n}\}$ *Donde* $\boldsymbol{T_i} = \{\boldsymbol{p_{3i}}, \quad \boldsymbol{_{3i+1}}, \boldsymbol{p_{3i+2}}\}$ *y* $\boldsymbol{p_{3i}}, \boldsymbol{p_{3i+1}}, \boldsymbol{p_{3i+2}} \in \boldsymbol{P}$.

Para todo P_i, P_j de T_k **hacer**:

si no existe la arista correspondiente para el par vértices P_i y P_j **entonces**

Se crea la arista.

Se crea la relación de vecindad entre cada par de vértice.

Se crea la relación de la nueva arista con el triángulo T y viceversa.

Se crea la relación de vecindad con T_k *y los vértices* P_i y P_j.

fin

sino

Se toma la posición que ocupa la arista en el arreglo A.

Se crea la relación de la nueva arista con el triángulo T_i y viceversa.

Se crea la relación de vecindad con T_k *y los vértices* P_i y P_j

Se crea la relación de vecindad entre los triángulos T_k y el anterior relacionado con la arista existente.

fin

fin

Una vez creado el Grafo Dual y actualizada la información de vecindad entre cada uno de los elementos de la geometría se procede a realizar la decimación.

2.3. Algoritmo de Decimación

Nuestra solución está basada en la implementación de un algoritmo basado en el colapso de triángulos [26, 27] cuyo principio de funcionamiento es unificar los tres vértices de un triángulo son unidos en un solo vértice (ver Figura 9).

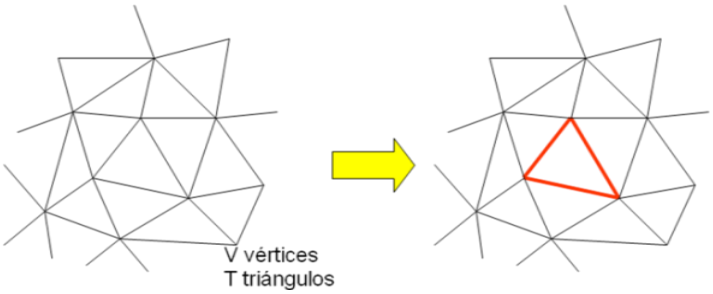


Figura 9. Selección del Triángulo.

Es importante destacar que en el algoritmo propuesto la ubicación del nuevo vértice está dada por el punto promedio de los tres vértices que conforman el triángulo eliminado (ver Figura 10).

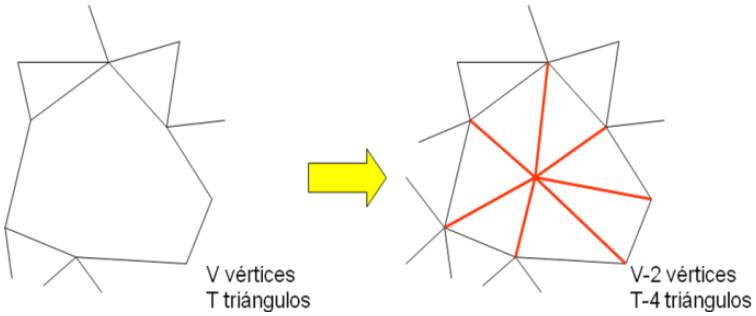


Figura 10. Eliminación del Triángulo.

El algoritmo del proceso de decimación propuesto se describe en el Algoritmo 3:

Algoritmo 3:
Entradas: $Grafo\ Dual$: $G(T,P,A)$ *Donde* T *es la lista de triángulos,* P *la lista de vértices y* A *la lista de aristas.* ***Salida:*** Puntos P' y triangulación T' perteneciente al nuevo modelo simplificado.
Paso 1: Se identifican los puntos p_i que conforman el borde : ***Algoritmo 4*** ***Paso 2:*** Estimación de la desviación normal promedio: ***Algoritmo 5*** ***Paso 3:*** Se simplifica la malla del modelo: ***Algoritmo 6*** ***Paso 4:*** Si se desea, volver al paso 1

2.3.1. Extracción de Bordes

La identificación de los puntos que conforman los bordes de la malla es la etapa del algoritmo que evita una deformación brusca de la malla (ver Figura 11). Se considera un vértice como borde cuando su vecindad sombrilla es incompleta. En una vecindad sombrilla completa [27] (ver Figura 12).

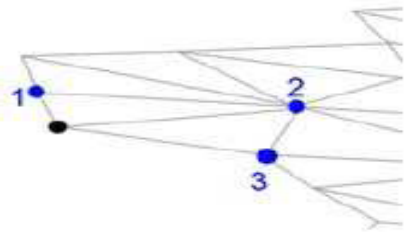


Figura 11. Punto Borde

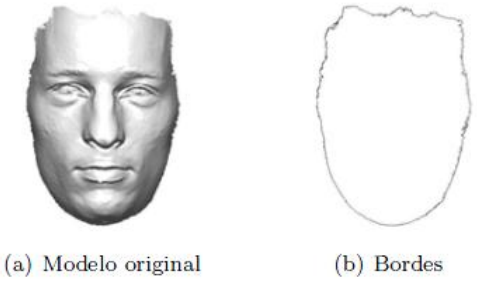


Figura 12. Bordes de una Malla

En esta etapa del algoritmo se recibe como parámetros de entrada un grafo $\boldsymbol{G}$ ($\boldsymbol{P}$, $\boldsymbol{T}$) donde P es el conjunto de vértices y $\boldsymbol{T}$ el conjunto de triángulos; y se obtiene como resultado el mismo $\boldsymbol{G}$ ($\boldsymbol{P'}$, $\boldsymbol{T}$) donde $\boldsymbol{P'}$ es el mismo conjunto de vértices inicial etiquetados con una bandera booleana en con valor verdadero en los vértices que son bordes dentro de la triangulación inicial. (Algoritmo 4)

Algoritmo 4:
Entradas: $Grafo\ Dual$: $G(T, P, A)$ *Donde* T *es la lista de triángulos,* P *la lista de vértices y* A *la lista de aristas.* ***Salida:*** Puntos P' que conforman el borde de la malla.
Paso 1: Obtener los puntos vecinos a un punto p_i ***Paso 2:*** **si** V_{pi} es vacío **entonces** p_i es un borde **fin** ***Paso 3:***

para todos los p_k Є V_{pi} **hacer**
  si p_k no se repite en V_{pi} **entonces**
    p_i es un borde
  fin
fin

2.3.2. Extracción de la Curvatura.

En la implementación y al igual que para la extracción de bordes, el análisis de curvatura se lleva a cabo para cada vértices de la malla. Se calcula la desviación normal promedio según la siguiente ecuación: $\boldsymbol{\theta = cos^{-1}\left(\vec{N}i \cdot \vec{N}prom\,(Rk)\right)}$ donde $\vec{N}i$ es el vector normal de cada vértice y $\vec{N}prom\,(Rk)$ es el vector normal de la vecindad sombrilla del vértice i (ver Figura 13).

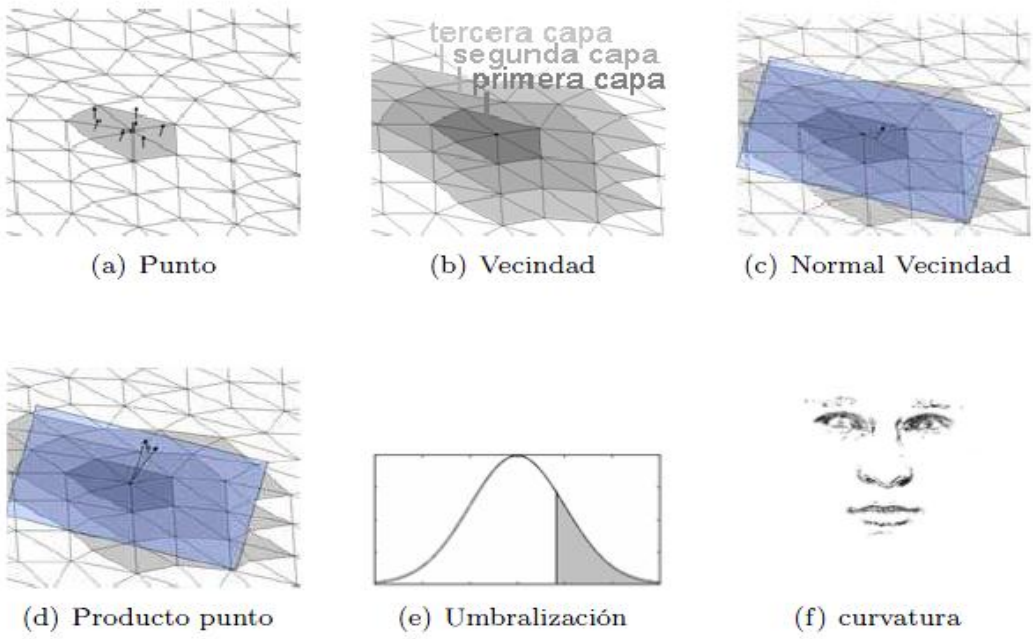


Figura 13. Extracción de la Curvatura. (a) Selección de la normal de un vértice. (b) Vecindad sombrilla de un vértice. (c) Vector normal de la vecindad sombrilla. (d) Angulo entre la normal del vertice y la normal de la vecindad sombrilla. (e) Umbralizar la curvatura. (f) Resultado final.

Para este proceso de extracción de la desviación normal promedio se necesitan como parámetros de entrada la nube de puntos y los triángulos que componen la malla y se obtiene como resultado la curvatura para cada punto de la superficie. En el mismo se llevan a cabo 5 procesos fundamentales para su correcta implementación, los cuales se describen a continuación en el seudo-código del Algoritmo 5.

Algoritmo 5:

Entradas:

$Grafo\ Dual$: $G(T, P, A)$ *Donde* T *es la lista de triángulos,* P *la lista de vértices y* A *la lista de aristas.*

Salida:

Curvatura C de cada vértice de P.

Paso 1:

Se calculan todas la normales de los triángulos t_i: $\overrightarrow{Ni} = \overrightarrow{(ai - bi)} \cdot \overrightarrow{(ai - ci)}$

Paso 2:

Se asigna como vector normal a cada punto p_j el promedio de las normales de los triángulos a los cuales pertenece.

Paso 3:

Se determinan las tres capas de la región sombrilla de cada punto p_j agregando en una lista V_j los triángulos t_i a los que éste pertenece y haciendo lo mismo con los primeros y segundos puntos vecinos. (ver **Figura 13-b**)

Paso 4:

Se calcula el vector normal a la vecindad sombrilla de cada punto p_j.

$$\vec{N}Vecindad = \left[norm\left(\sum_{j|Pj \in Rk} A_j \vec{N}_j \right) \right]^{-1} \sum_{j|Pj \in Rk} A_j \vec{N}_j$$

Donde A_j y $\vec{N}_j$ son el área y el vector normal de cada triángulo perteneciente a la vecindad.

Paso 5:

Se calcula la desviación normal promedio para cada punto p_j como el ángulo entre su vector normal y el de su vecindad $Vecindadj$.

$$\theta = \cos^{-1}\left(\vec{N}_i \cdot \vec{N}Vecindadj\ (R\mathrm{k}) \right)$$

2.3.3. Proceso de Decimación.

Una vez identificados los vértices bordes y calculada la desviación normal promedio de cada vértice se procede a eliminar los triángulo dentro de la malla inicial que no contengan dentro de sus tres vértices ningún vértice borde y que todas las desviaciones normales de los vértices que lo constituyen estén por debajo de un valor umbral definido, (ver Algoritmo 6). Dicho valor umbral se puede definir siguiendo donde criterios fundamentales:

1. Asumir como umbral el valor medio de las desviaciones normales promedios de todos los vértices.
2. Asumir como umbral el promedio de las desviaciones normales más dos veces la desviación estándar de las mismas.

Algoritmo 6:

Entradas:

$Grafo\ Dual$: $G(T, P, A)$ *Donde* T *es la lista de triángulos,* P *la lista de vértices y* A *la lista de aristas.*

Cada t $\in T$ *tiene 3 vértices* $t = (a, b, c)$

Salida:

Puntos P' y triangulación T' pertenecientes al nuevo modelo simplificado.

Paso 1:

para todos los triángulos ti **hacer**
si ni a ni b ni c son bordes y a, b y c tienen curvatura baja y t_i puede ser removido y no ha sido borrado **entonces**

$$d = (a + b + c)/3$$

a, b y c son reemplazados por d
t_i ha sido borrado

todos los triángulos que compartían un punto con t_i no pueden ser removidos
Los triángulos que compartían dos puntos con ti son borrados
fin si
fin para

Paso 2:

se corrige la malla del modelo decimado: **Algoritmo 7**

2.3.4. Corregir.

Una vez obtenida la triangulación correspondiente a la simplificación del modelo, es necesario realizar un proceso de corrección de la misma, ya que para algunos casos; al eliminar un triángulo se pueden generar nuevas triangulaciones afectadas por tres fenómenos fundamentales:

1. No se mantenga la orientación original de las normales de la malla incial.
2. Generación de pirámides [26, 27].
3. Generación de espigas [27].

En el Algoritmo 7 se propone un seudo-código para el tratamiento de cada una de estas triangulaciones no deseadas.

Algoritmo 7:

Entradas:

$Grafo\ Dual$: $G(T, P, A)$ *Donde* T *es la lista de triángulos,* P *la lista de vértices y* A *la lista de aristas.*

Cada $t \in T$ *tiene 3 vértices* $t = (a, b, c)$

Salida: Puntos P' y triangulación T' pertenecientes al nuevo modelo simplificado.

Paso 1:

Remover espigas
para todos los triángulos t_i **hacer**
si algún triángulo vecino posee los mismos puntos **entonces**
ambos son borrados
fin si
fin para

Paso 2:

remover pirámides
para todos los puntos p_i **hacer**
si pertenece a tres triángulos y el ángulo entre uno de los triángulos y los

otros dos es mayor a 90 **entonces**
borrar los tres y conformar uno nuevo
fin si
fin para

Paso 3:

Cambiar la orientación de los triángulos que están al revés

2.4. Persistencia y Visualización de la Geometría Simplificada.

La geometría obtenida es exportada en formato (*.stl) para su posterior uso en el sistema de visualización medica VISMEDIC y por otros softwares. La visualización de la misma se realiza utilizando como API de visualización gráfica OpenGL con el objetivo de que la aplicación propuesta sea funcional en las plataformas Windows y Linux. En el próximo capítulo se realizamos una descripción detallada del sistema de software propuesto.

CAPÍTULO 3. CARACTERÍSTICAS DEL SISTEMA

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

3.1. Reglas del Negocio.

Los modelos que se deseen visualizar para su posterior simplificación deben estar en formato (*.stl) o en el formato (*.ase). Los modelos con estos formatos deben tener la estructura original, los mismo con los datos correspondientes a los vértices, las caras y las normales para cada uno de ellos, los que no cumplan con este requisito aunque hayan sido visualizados no podrán ser simplificados correctamente.

3.2. Modelo del Dominio.

El Modelo de Dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software. En el presente trabajo no es necesario realizar un complejo Modelo de Negocio, el Modelo de Dominio es suficiente para modelar todo el flujo del entorno de trabajo. En la Figura 14 se muestra el Modelo de Dominio elaborado.

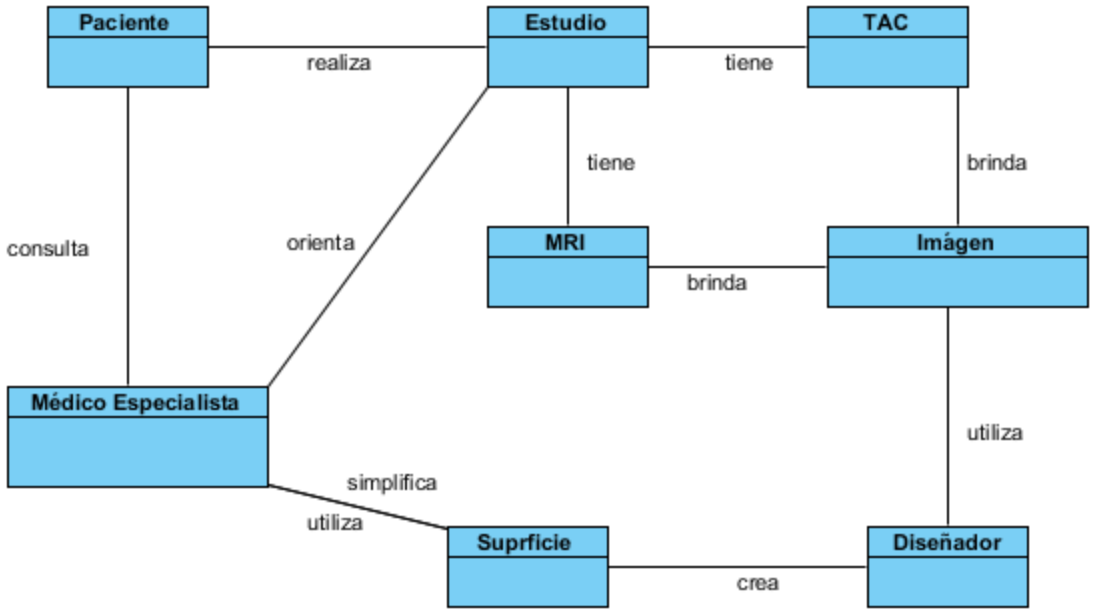


Figura 14. Modelo del Dominio

3.2.1. Descripción del Dominio.

El médico le orienta al paciente un estudio que puede ser una Tomografía Axial Computarizada (TAC), una Imagen de Resonancia Magnética (MRI) o ambas; los cuales brindan imágenes que son aprovechadas por el diseñador para modelar una superficie que le sirva al médico para dar un diagnóstico de los padecimientos del paciente. Además el especialista médico podría simplificar la superficie para lograr un balance entre la calidad visual y la rapidez de interacción con la misma.

3.3. Captura de Requisitos.

Un requerimiento es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, norma, especificación u otro documento formal. Estos facilitan el entendimiento entre usuarios y desarrolladores del sistema a elaborar. A continuación se exponen los requisitos funcionales por los que se regirá el sistema y los no funcionales que exponen las características de la aplicación.

3.3.1. Requerimientos funcionales.

Los ***requisitos funcionales*** representan la funcionalidad del sistema y se modelan mediante diagramas de casos de uso. Los siguientes requisitos responden a las funcionalidades que el sistema debe tener una vez concluida la implementación.

RF1. Cargar Modelo

- RF1.1 Seleccionar el modelo a cargar
- RF1.2 Simplificar modelo
- RF1.3 Exportar modelo

RF2. Mostrar modo *Wireframe*

RF3. Rotar modelo

- RF3.1 Rotar modelo sobre el eje X
- RF3.2 Rotar modelo sobre el eje Y
- RF3.3 Rotar modelo sobre el eje Z

RF4. Trasladar modelo

- RF4.1 Trasladar modelo sobre el eje X
- RF4.2 Trasladar modelo sobre el eje Y

RF4.3 Trasladar modelo sobre el eje Z

3.3.2. Requerimientos No Funcionales.

Los requisitos no funcionales representan aquellos atributos que debe tener el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de facilidad de uso, fiabilidad, eficiencia, portabilidad. Los requisitos no funcionales que debe de tener el módulo elaborado son los siguientes:

1. Software

El sistema operativo sobre el cual debe ejecutarse la aplicación será Windows XP, Windows 7 o Ubuntu.

2. Hardware

El modelo de microprocesador será Intel Pentium IV a 3.0 GHz o superior.

La memoria RAM será de 1GB.

3. Seguridad

Fiabilidad: Los modelos tridimensionales visualizados deben tener una gran calidad para permitir un análisis lo más exacto posible para los especialistas.

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

Integridad: No debe haber pérdidas de información ni de calidad en las imágenes obtenidas.

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

4. Interfaz Externa

La interfaz de usuario debe ser sencilla y amigable para permitir al usuario una rápida y cómoda interacción con las funcionalidades del módulo.

5. Soporte

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

6. Diseño e Implementación

Se empleará como lenguaje de programación C++ y el Framework Qt para el diseño de las interfaces gráficas.

3.4. Modelos de Casos de Uso del Sistema.

En este epígrafe se identificarán los actores del sistema que se desea desarrollar así como los casos de uso. Además se hará la descripción textual de los casos de uso del sistema, permitiendo así una comprensión más precisa y una mejor idea de la lógica del funcionamiento del módulo.

3.4.1. Actores del Sistema.

Los actores del sistema son entidades externas al sistema que guardan una relación con este y que le demandan una o más funcionalidades. Esto incluye a los operadores humanos, pero también incluye a todos los sistemas externos. En este caso particular quien hará uso del sistema será un especialista médico, que como actor del sistema se llamará médico especialista.

Actores	Justificación
Médico Especialista	El médico especialista es quien interactua con el sistema para ejecutar las funcinalidades de: Cargar modelo, Rotar el modelo, Trasladar el modelo y Simplificar el modelo.

Tabla 1. Actores del Sistema

3.4.2. Diagrama de Casos de Uso del Sistema.

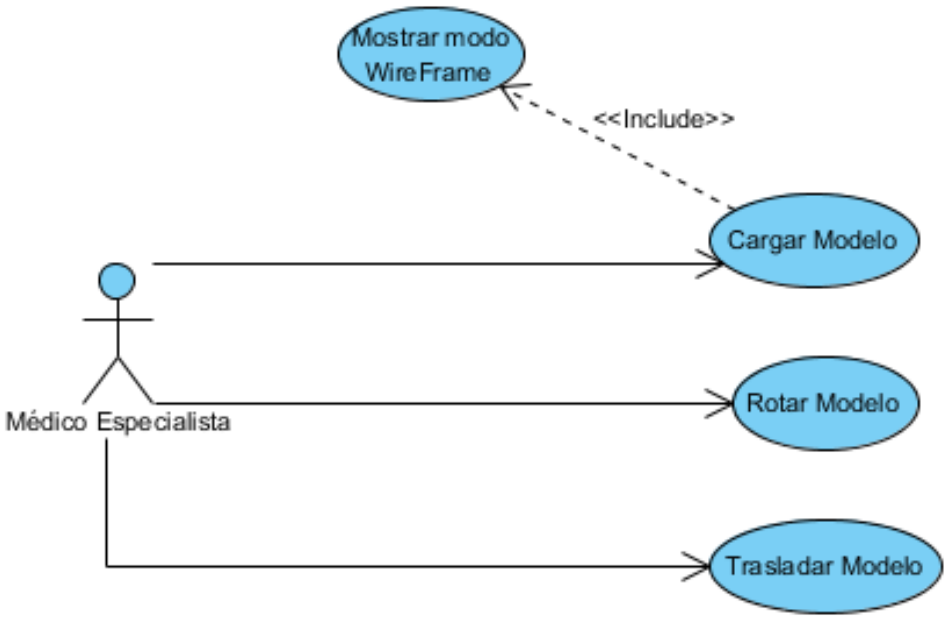


Figura 15. Casos de uso del Sistema

3.4.3. Descripción de los Casos de Uso del Sistema.

Cada caso de uso tiene una descripción de las funcionalidades que ejecutará el sistema propuesto como respuesta a las acciones del usuario. Las tablas siguientes muestran los flujos operacionales de cada caso de uso.

Caso de Uso:	Cargar Modelo
Actores:	Médico Especialista
Propósito:	Su propósito es seleccionar el modelo a cargar, simplificar y exportar el resultado deseado.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de cargar el modelo.
Referencia:	RF1.1, RF1.2, RF1.3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario puede seleccionar la opción adicionar	1.1 El sistema ejecuta una de las siguientes acciones: a) Si el actor desea cargar un modelo consulta la sección Abrir Archivo. b) Si el actor desea simplificar el modelo consulta la sección Aplicar Decimación. c) Si el actor desea exportar el modelo consulta la sección Salvar Archivo.
Post-condiciones:	Se visualiza el modelo, se simplifica o se exporta según se desee.
Prioridad:	Crítica.

Tabla 2. Caso de uso Cargar Modelo

Sección:	Abrir Archivo
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Abrir Archivo.	1.1 El sistema muestra en el cuadro de diálogo para buscar el modelo a visualizar.

2. El usuario selecciona el modelo y acepta	2.1 El sistema visualiza el modelo seccionado.
Post-condiciones:	Se carga y se visualiza el modelo seleccionado.
Prioridad:	Crítica.

Tabla 3. Sección Abrir Archivo

Sección:	Aplicar Decimación
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Criterio Matemático. 2. El usuario selecciona la opción Aplicar Decimación	1.1 El sistema crea la simplificación del modelo cargado utilizando el criterio matemático y muestra el resultado.
Flujo Alterno de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Aplicar Decimación	1.1 El sistema crea la simplificación del modelo cargado sin el criterio matemático y muestra el resultado.
Post-condiciones:	Se simplifica el modelo.
Prioridad:	Crítica.

Tabla 4. Sección Aplicar Decimación

Sección:	Salvar Archivo
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Salvar	1.1 El sistema muestra en el cuadro de diálogo para seleccionar donde se desea

Archivo.	guardar el modelo.
2. El usuario selecciona la dirección para guardar el modelo.	2.1 El sistema guarda el modelo en dicha dirección.
Post-condiciones:	Se guarda el modelo.
Prioridad:	Crítica.

Tabla 5. Sección Salvar Archivo

Caso de Uso:	Mostrar modo Wireframe
Actores:	Médico Especialista
Propósito:	Su propósito mostrar el mallado del modelo visualizado.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de Wireframe.
Referencia:	RF2
Precondición:	Debe haberse realizado el Caso de Uso Cargar Modelo.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Wireframe	1.1 El sistema visualiza el mallado del modelo
Post-condiciones:	Se visualiza el mallado del modelo.
Prioridad:	Secundaria.

Tabla 6. Caso de uso Modo Wireframe

Caso de Uso:	Rotar Modelo
Actores:	Médico Especialista

Propósito:	Su propósito rotar el modelo.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de Rotar.
Referencia:	RF3.1, RF3.2, RF3.3
Precondición:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Rotar	1.1 El sistema rota la escena.
Post-condiciones:	Se visualiza la escena rotada.
Prioridad:	Secundaria.

Tabla 7. Caso de uso Rotar Modelo

Caso de Uso:	Trasladar Modelo
Actores:	Médico Especialista
Propósito:	Su propósito es trasladar el modelo.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de trasladar.
Referencia:	RF4.1, RF4.2, RF4.3
Precondición:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción Trasladar	1.1 El sistema traslada la escena.
Post-condiciones:	Se visualiza la escena trasladada.
Prioridad:	Secundaria.

Tabla 8. Caso de uso Trasladar Modelo

3.5. Diagrama de Clases.

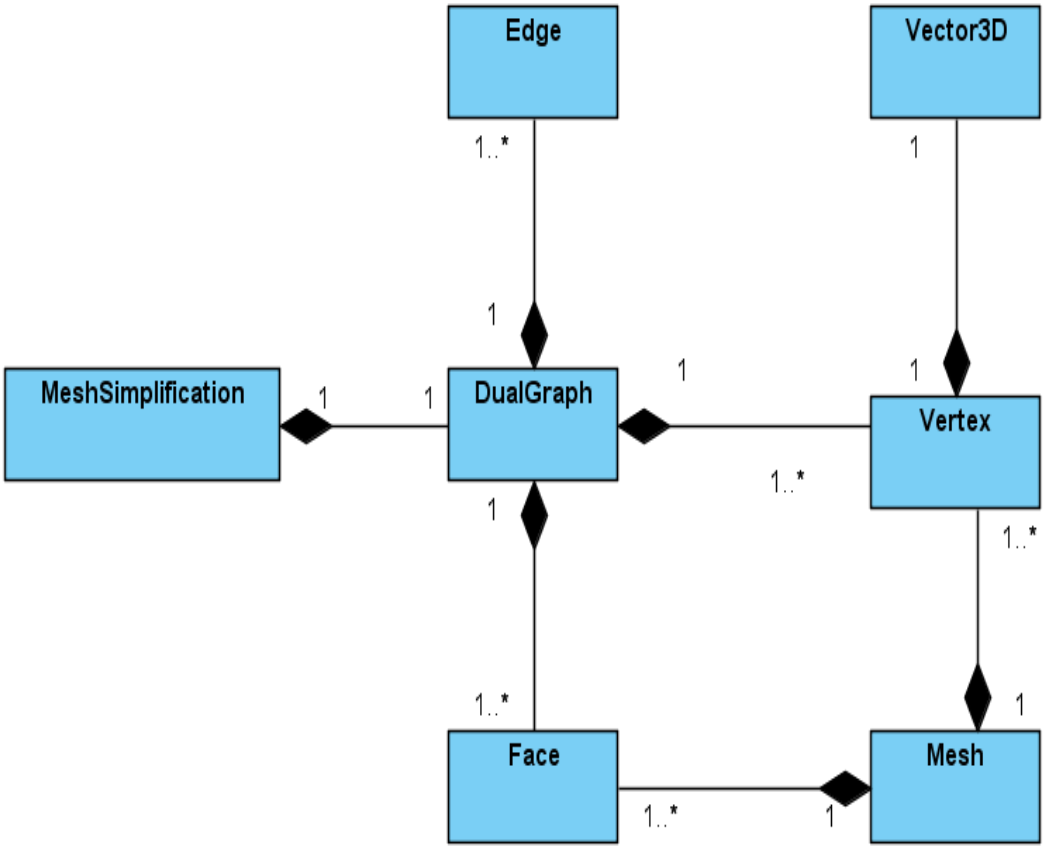


Figura 16. Diagrama de Clases

3.6. Diagramas de Secuencias.

A continuación se representarán los diagramas de secuencia del diseño para tener una idea más general sobre el flujo que se realiza entre las clases del diseño y que posibilita comprender mejor el módulo elaborado en términos de implementación.

3.6.1. Diagrama de Secuencia del Caso de Uso Cargar Modelo.

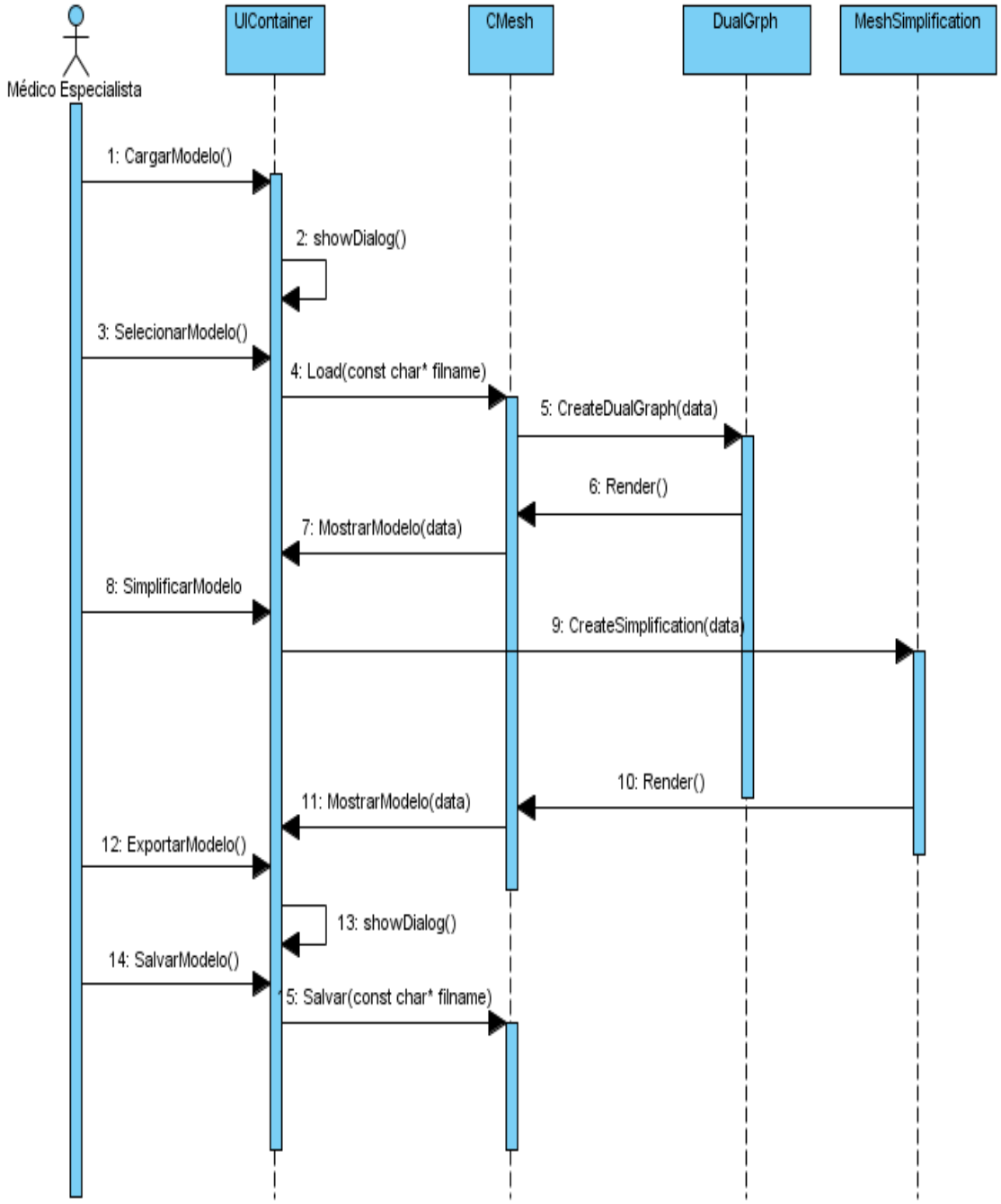


Figura 17. Diag. Sec. Cargar Modelo

3.6.2. Diagrama de Secuencia del Caso de Uso Mostrar Modo Wireframe.

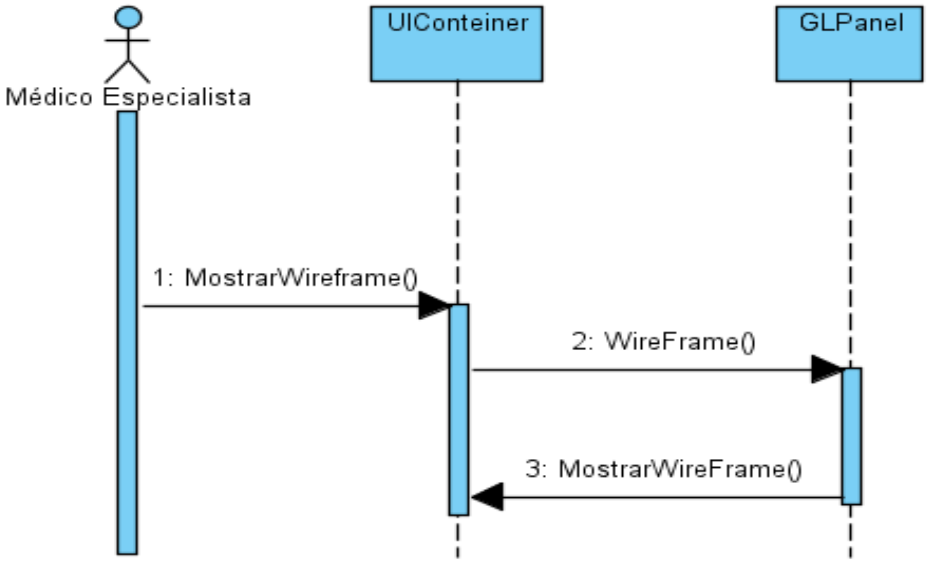


Figura 18. Diag. Sec. Modo Wireframe.

3.6.3. Diagrama del Caso de Usos Rotar Modelo.

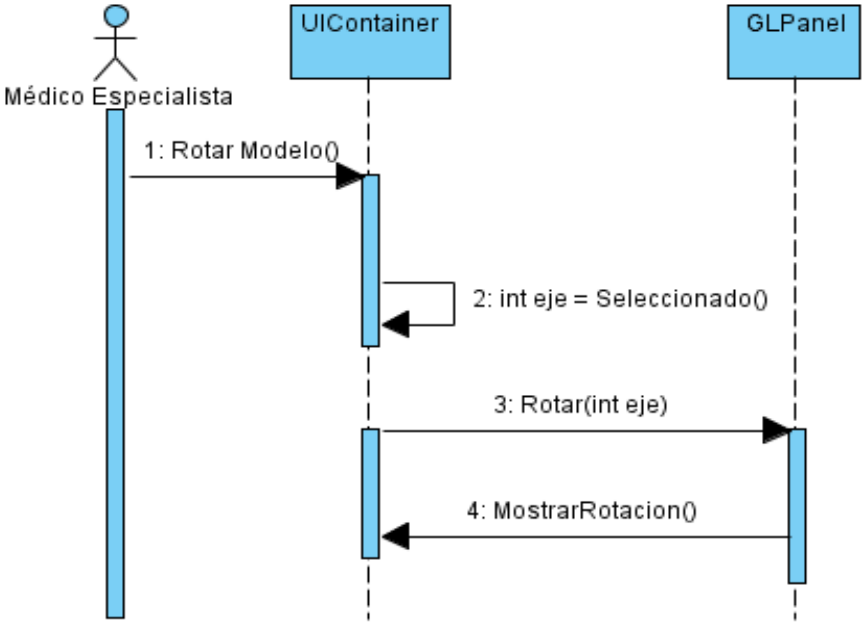


Figura 19. Diag. Sec. Rotar Modelo.

3.6.4. Diagrama del Caso de Uso Trasladar Modelo.

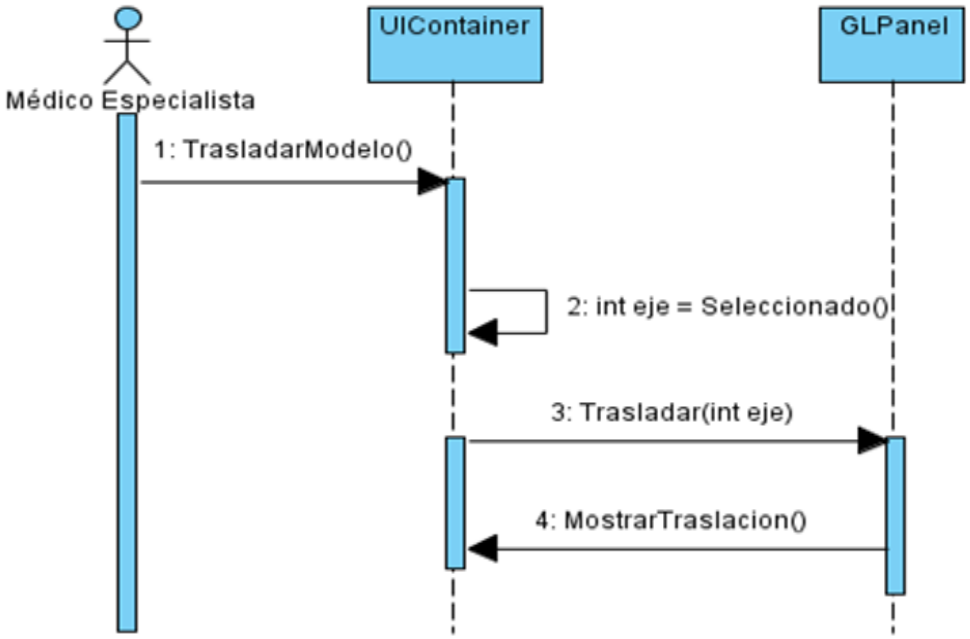


Figura 20. Diag. Sec. Trasladar Modelo

CAPÍTULO 4. RESULTADOS Y VALIDACIÓN

En este capítulo se abordan los temas de la implementación del módulo basados en todo el trabajo acumulado a lo largo de los capítulos anteriores y, además, se hará un análisis de los resultados obtenidos en cuanto a calidad y cantidad de información almacenada, cumplimiento de los objetivos propuestos, comparaciones entre los modelos originales y los modelos decimados.

4.1. Resultados

En este epígrafe se muestran los resultados obtenidos con el módulo desarrollado. Estos resultados permitirán hacer los análisis y las comparaciones de los resultados obtenidos al realizar las simplificaciones deseadas, basados en la calidad de los modelos resultantes y la cantidad de primitivas que presentan después de la decimación de los mismos.

4.2. Datos de Entrada.

Para la realización de las pruebas se tomaron modelos exportados en el proyecto Visualización Científica por el algoritmo Marching Cube con diferentes tipos de complejidades de acuerdo con la cantidad de información almacenada en los mismos. Se tomaron modelos de distintas ramas para su comparación.

4.3. Parámetros a Medir.

Las pruebas que se le realizaron al módulo se enfocaron en dos aspectos fundamentales por la importancia de los mismos en este tipo de trabajo. La cantidad de primitivas geométricas resultantes después de la decimacion de los modelos y la calidad de los mismos.

4.4. Resultados Obtenidos.

A continuación se presentan los modelos seleccionados para realizar las pruebas del módulo desarrollado comprobando la calidad y cantidad de información resultante luego de realizarse la decimación de los mismos.

La representación de los modelos está dada en el modo wireframe de manera que se pueda apreciar cómo queda la malla luego de ser simplificada.

Modelo	No. de Triángulos Inicial	No. de Triángulos Final	%	No. de Iteraciones
Head256	784 121	209 217	26%	2
Pelvis	100 850	29 362	29%	2
BostonTeapot	479 838	133 714	27%	2
Bonsai	451 732	126 600	28%	2
Engine	582 942	171 675	29%	2

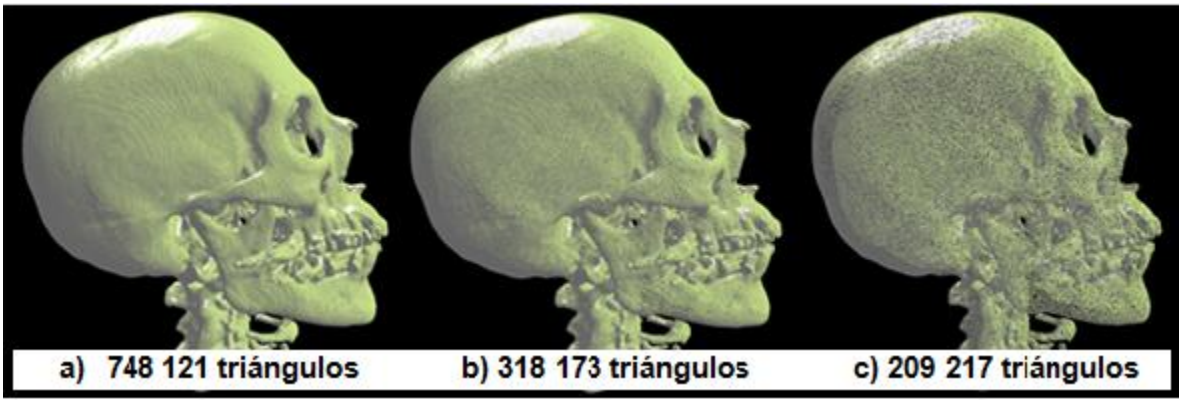


Figura 21. Modelo Head256

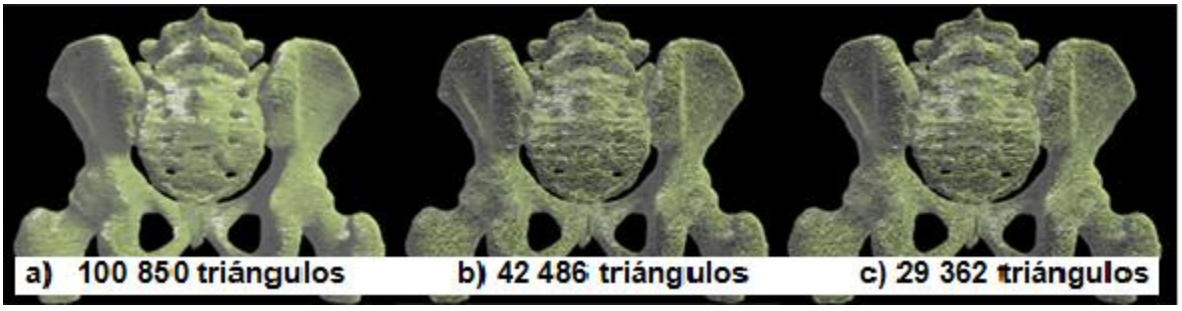


Figura 22. Modelo Pelvis

Figura 23. Modelo Boston Teapot

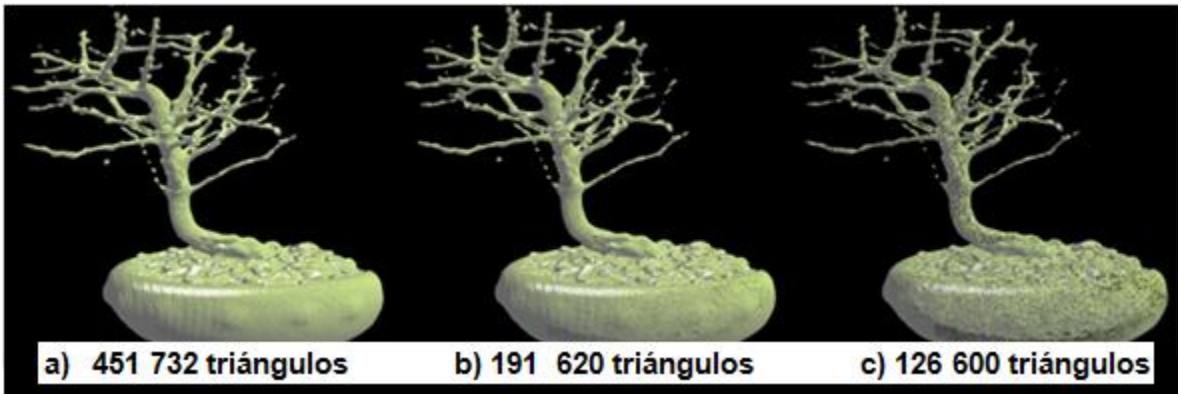


Figura 24. Modelo Bonsai

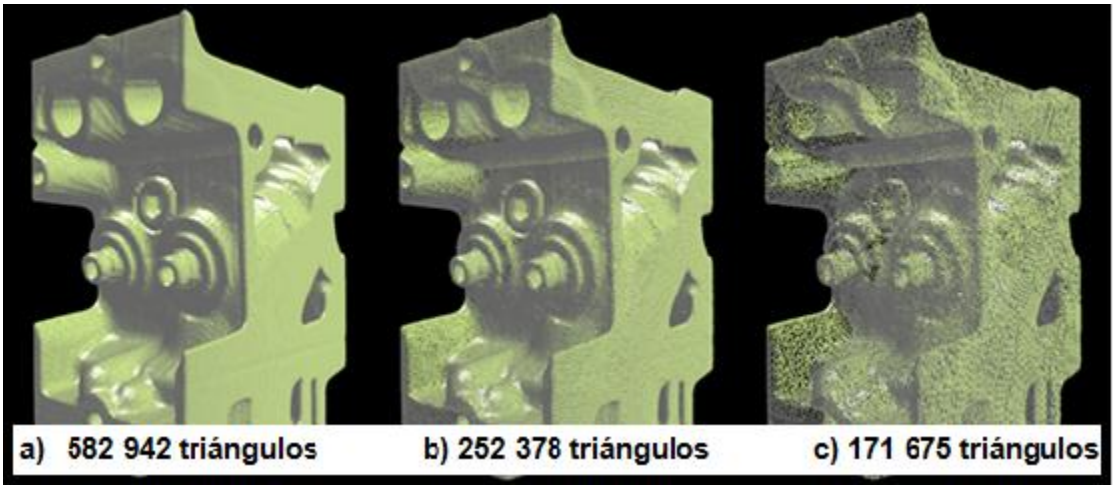


Figura 25. Modelo Engine

CONCLUSIONES

Con la realización de este trabajo se elaboró un módulo de simplificación que permite la decimación de las geometrías generadas por el algoritmo Marching Cube utilizadas en el proyecto Visualización Médica (VISMEDIC) logrando obtener un balance entre la calidad de las superficies y la interacción con las mismas en tiempo real.

La investigación demostró que para obtener tiempos de respuestas aceptables es necesario adaptar los algoritmos básicos a los requerimientos de la aplicación a partir de una clasificación topológica correcta de la geometría que se desea simplificar.

Se identificó que la malla obtenida como salida del algoritmo Marching Cube implementado en el proyecto VISMEDIC presenta problemas de ambigüedades (vértices repetidos) las cuales se eliminaron con la etapa de validación que se propuso realizar antes de realizar el proceso de decimación, la cual consistió en la realización de un “welding” de los vértices.

RECOMENDACIONES

Como trabajos futuros abiertos por esta investigación identificamos los siguientes:

1. Primeramente incluir dentro del módulo desarrollado algoritmos de decimación basados en la eliminación de vértices y aristas.
2. Implementar otras métricas de selección de primitivas a eliminar (criterio matemático).

Las cuales permitirán extender la usabilidad del sistema propuesto a otros tipos de aplicaciones que requieran la obtención de diferentes niveles de detalles de un mismo modelo geométrico.

BIBLIOGRAFÍA

1. **Garland, Michael.** *Quadric-Based Polygonal Surface Simplification.* Excelente : s.n., 1999.

2. **Paul S. Heckbert, Michael Garland.** *Survey of Polygonal Surface Simplification Algorithms.* 1997.

3. **Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk.** *Simplification Envelopes.* 1996.

4. **Paul Ning, Jules Bloomenthal.** *An Evaluation of Implicit Surface Tilers.*

5. **William J. Schroeder, Jonathan A. Zarge, William E. Lorensen.** *Decimation of Triangle Meshes.*

6. **W.J. Schroeder, Boris Yamrom.** *A compact cell structure for scientific visualization.*

7. **Paul Hinker, Charles Hansen.** *Geometric optimization.*

8. **Robert F. Tobler, Stefan Maierhofer.** *A Mesh Data Structure for Rendering and Subdivision.*

9. **Hernandez, Susana Martha.** *Simplificación de mallas triangulares mediante clustering adaptativo.* 2009.

10. **Hoppe, Hugues.** *Progressive Meshes.*

11. **Cohen, Jonathan D.** *Concepts and Algorithms for Polygonal Simplification.* 2000.

12. **Eric Shaffer, Michael Garland.** *Efficient Adaptive Simplification of Massive Meshes.*

13. **Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle.** *Mesh Optimization.*

14. **Fernández, Susana Mata.** *Modelado Multirresolucion.* 2010.

15. **Li Guangming, Tian Jie, Zhao Mingchang, He Huiguang, Zhang Xiaopeng.** *A New Mesh Simplification Algorithm Combining Half-edge Data Structure with Modified Quadric Error Metric.*

16. **Carlos González, Jesús Gumbau, Miguel Chover, Pascual Castelló.** *Simplificación de mallas para juegos.*

17. **Erikson, Carl.** *Polygonal Simplification: An overview.* 1996.

18. **María V. Cifuentes, Juan P. D'Amato, Cristian García Bauza, Marcelo Vénere.** *Tecnicas de simplificacion de modelos topograficos.*

19. **Belón, Mauricio Cele López.** *Visualización Interactiva de Volúmenes.* 2007.

20. **Renato Pajarola, Marc Antonijuan, Roberto Lario.** *QuadTIN: Quadtree based Triangulated Irregular Networks.* 2002.

21. **Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Levy, Stephan Bischoff, Christian Rossl.** *Geometric Modelling Based on Polygonal Meshes.* 2008.

22. **Michael Garland, Andrew Willmott, Paul S. Heckbert.** *Hierarchical Face Clustering on Polygonal Surfaces.*

23. **Paul S. Heckbert, Michael Garland.** *Optimal Triangulation and Quadric-Based Surface Simplification.* 1999.

24. **Lindstrom, Peter.** *Out-of-Core Simplification of Large Polygonal Models.*

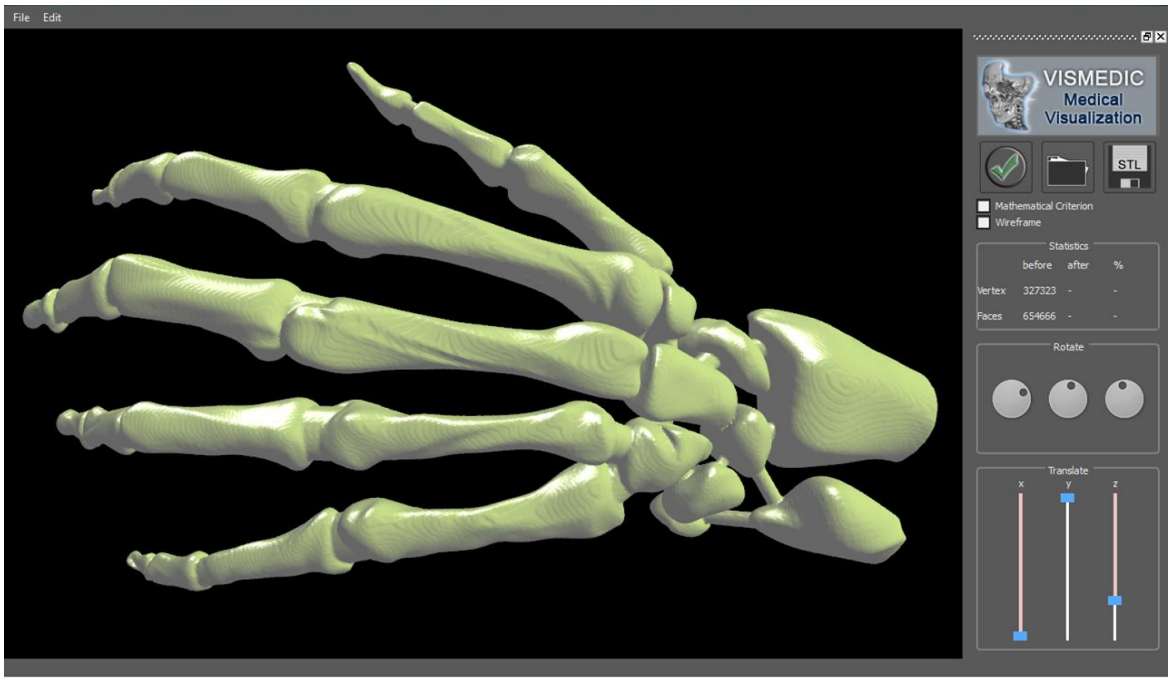
25. **Michael Garland, Paul S. Heckbert.** *Surface Simplification Using Quadric Error Metrics.*

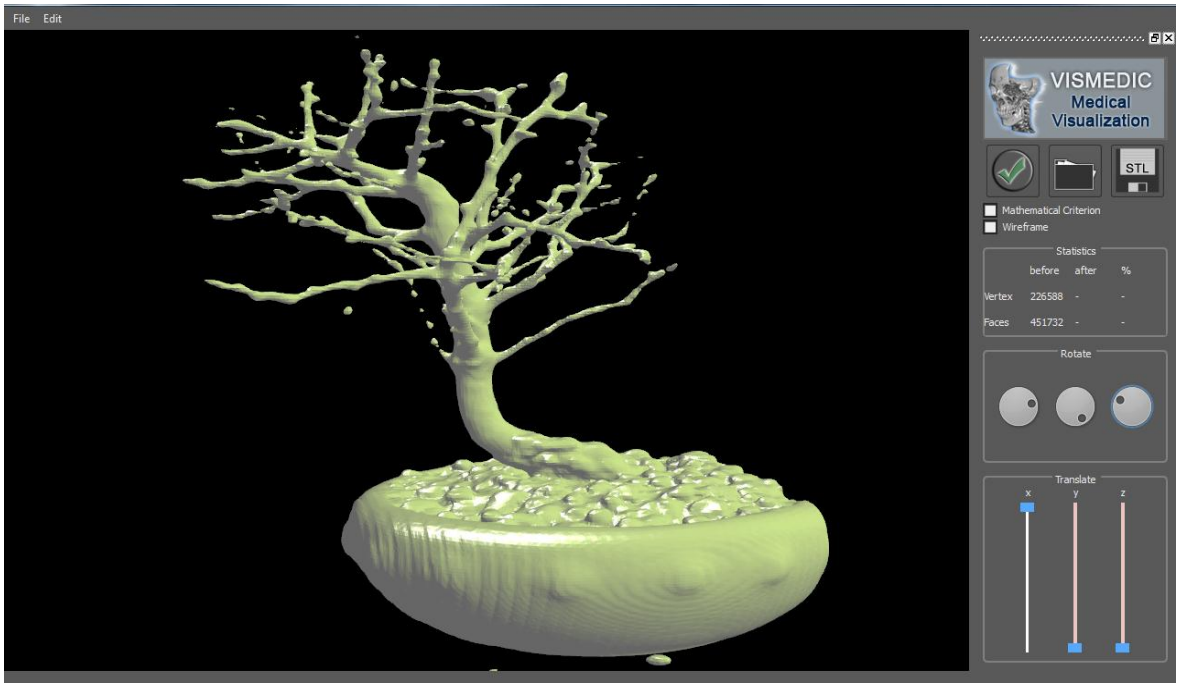
26. **Alexánder Ceballos, Jorge Hernández, Flavio Prieto.** *Multirresolución Adaptativa de Mallas Triangulares Basado en Criterio de Curvatura.* 2007.

27. **Alexander Ceballos, Jorge Hernández, Flavio Prieto.** *Multirresolución Adaptativa de Mallas Triangulares Basado en Criterio de Textura.* 2007.

ANEXO

A continuación se muestran imágenes de la aplicación correspondiente al módulo desarrollado.





GLOSARIO DE TÉRMINOS

A

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

C

CPU: Unidad de procesamiento central.

Coplanares: Puntos o figuras que están situados en el mismo plano, en caso contrario se dice que son no coplanares.

D

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

E

Envolturas geométricas: Capa envolvente de un modelo geométrico.

H

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

M

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

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

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

N

Normal: Vector tridimensional, perpendicular a la cara de un objeto, determinando la dirección en que ella apunta.

Nivel de detalles: Niveles de visualización en una escena virtual.

O

OpenGL: Librería de Gráficos de Código Abierto (del inglés Open Graphics Library). API para diseñar gráficos en 2D y 3D creada por Silicon Graphics en 1992. La soporta cualquier sistema operativo y sólo hace falta una tarjeta gráfica con soporte para OpenGL. Es el principal competidor de Direct3D de Microsoft.

P

Primitivas geométricas: Formas geométricas consideradas primitivas por su básica constitución en las partes que conforman. En un software 3D pueden ser editadas para conseguir formas geométricas más complejas agregando nuevos vértices, aristas y polígonos.

R

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

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

Resonancia Magnética: 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 de tomografías (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.

T

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

Tridimensional: Se dice de lo que se desarrolla en las tres dimensiones espaciales, altura, anchura y profundidad.

Triangulación: La triangulación de superficies es un método de obtener áreas de figuras poligonales, normalmente irregulares, mediante su descomposición en formas triangulares.

V

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