Universidad de las Ciencias Informáticas

Facultad 5

Módulo de corte en mallas superficiales.

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

Autor: Yaniel Calviño Cruz

Tutor: MSc. Osvaldo Pereira Barzaga

Co-tutor: Ing. Ernesto Carrasco de la Torre

"*La Habana, junio 2011"*

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos 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 firmamos la presente a los ____ días del mes de _____ del año _____.

Yaniel Calviño Cruz
Autor

MSc. Osvaldo Pereira Barzaga
Tutor

Ing. Ernesto Carrasco De la Torre
Co-tutor

DATOS DE CONTACTO

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

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

AGRADECIMIENTOS

Quisiera agradecerle a cada una de las personas que han tenido que ver conmigo durante la carrera y especialmente a todos los que han hecho posible la conclusión del presente trabajo.

Especialmente a mis tres madres:

Estoy seguro de que sin estas tres personas no hubiera podido hacer nada de lo que he hecho en mi vida, mi madre Rebeca la mujer que me trajo al mundo, la mujer que más se ha preocupado por mí, la que me ha enseñado el camino a seguir, a mi abuela Mary que siempre me ha apoyado ni importa lo lejos que pueda estar, a mi tía Mirely que desde que empecé con el pre siempre ha estado más cerca de mí que mi propia madre debido a la ubicación de las escuelas y que ha sabido cubrir ese espacio que me faltaba para salir adelante en mis estudios.

A mi padre y mi familia por parte de padre:

Les agradezco de corazón todo el apoyo que siempre me han dado y su preocupación por todo lo relacionado con mis estudios y mi vida, por atenderme tan bien como siempre lo han hecho, y por estar siempre orgullosos de mí.

A mis primas y el marido de mi tía:

A mis primas Mirelita y Cludia por estar siempre cerca de mí y hacerme la vida más alegre, a Dagoberto por darme los consejos necesarios para sobrellevar la vida y ser lo más parecido a un padre que tuve durante mi carrera.

A mi tía:

A Tata por formar parte de mi crianza y siempre estar preocupada por mí y ayudarme en todo lo que ha podido.

A mi novia y toda su familia:

A mi novia Dayana por estar siempre a mi lado brindándome su amor y apoyarme constantemente y además soportarme durante este tiempo, a toda la familia por parte de madre en especial los que viven con ella en la finca que me han dado todo su cariño

como si fuera un miembro más de la familia, a toda la familia por parte de padre que igualmente me han dado un espacio entre ellos a todos muchas gracias.

A mis tutores:

Por estar siempre pendiente del desarrollo y conclusión del presente trabajo y por enseñarme como debe ser un profesional y como administrar el tiempo para lograr alcanzar todas las metas que nos tracemos, gracias a ellos por su apoyo. Agradecer a Ernesto Guevara también por ser casi un tutor ante la ausencia de los míos.

A Lester y Leonardo:

Por ayudarme mucho con la tesis, el apoyo de ellos fue muy importante.

A mis compañeros:

A mis compañeros de proyecto en especial a Rufino que me dio una mano cuando más lo necesitaba y siempre me apoyo en este trabajo, al Guille y Rubén que siempre que los necesite estaban ahí sin ningún tipo de problemas, a mis compañeros de aula desde primero año que siempre nos hemos preocupados los unos por los otros, especialmente a Juan Carlos, Alexis, Adruban, Evelio, Maceo, Andrés. A la gente de mi zona Jorgito y Raidel por venir desde la primaria junto conmigo.

Al piquete de programación:

Bueno a todo el movimiento de programación de la UCI que juntos pusimos el nombre de la UCI bien en alto, a mis dos compañeros de equipo Bankai Ray y Alkaid, a Yandri que fue nuestro coach, a los otros “rivales” pero que siempre nos llevamos como hermano fueras de las competencias Charchaval, Mario, Abel, Ballester, Boloy, Luis Mariano que con el compartí varias competencias perteneciendo al mismo equipo representando la facultad 5, a Leandro, a Dovier, a Toranzo y Argelio que una vez formamos equipo también y fueron los que me iniciaron en este mundo y me enseñaron mucho por hacer posible el sueño de que pudiéramos participar en eventos como ACM-ICPCE, y a todos los otros que llevan y tienen interés de seguirle dando resultados a la UCI.

A la facultad 5:

A la facultad 5 por siempre depositar en mi toda su confianza y apoyarme en todas las cosas que me hicieron falta siempre sin vacilar, a Liván el presidente de la FEU que siempre me brindo una mano cuando hacía falta, a la Millet que nunca vacilo en apoyar los movimientos de programación y de matemática en aras de que la facultad saliera adelante, a toda la dirección en particular por querer llevar la facultad adelante y haberla convertido en una de las mejores de la UCI.

A la UCI:

A esta hermosa universidad que me ha enseñado como debe ser una persona cuando quiere lograr sus sueños y por permitirme formarme como profesional.

A Fidel:

A nuestro máximo líder y conductor de la Revolución Cubana, por ser el principal creador de esta universidad que me ha dado la oportunidad de convertirme en un profesional listo para servir a la Revolución.

DEDICATORIA

A mi madre:

Esa persona tan especial que siempre me ha apoyado en todo momento, ha sabido ser madre y padre al mismo tiempo cosa que no es nada fácil, todo lo que soy hoy en día y todo lo que he logrado se lo debo a ella, gracias por tu apoyo.

A mi padre:

Aunque no esté presente físicamente siempre trató de ser un ejemplo para mí y darme consejos necesarios para la vida.

A mi abuela:

A mi abuela por parte de madre que valga la redundancia ha sido como una madre para mí, a ella que es una de las personas más luchadoras que he conocido y que me ha enseñado lo importante que es la familia y el cómo sacrificarse por el bienestar de la misma.

A mi tía:

A mi tía Mirely por ser mi otra madre sobre todo cuando no tenía a las otras dos cerca, siempre me ha tratado como un hijo más y se ha sacrificado mucho por mí.

A mi hermano:

A mi hermano Elier que es el único de mis hermano que lo vi nacer y espero ser un ejemplo para él y que siempre este orgulloso de su hermano que lo quiere mucho.

A mi tía y mis primas:

A mi tía Tata por estar siempre preocupada por mí y apoyarme siempre que pudo, a mis primas que siempre estuvieron cerca de mí y las he visto crecer al lado mío.

RESUMEN

La Realidad Virtual es uno de los temas más tratados en la actualidad, la misma abarca varias áreas de conocimiento dentro de las que se encuentra la medicina. Dentro de la misma el corte en mallas superficiales triangulares es un proceso fundamental, ya que sin este es imposible la realización del proceso de simulación de intervenciones quirúrgicas.

Debido a la gran importancia que tiene esta problemática dentro de los proyectos que se vinculan con la simulación médica de intervenciones, dentro del que se incluye el proyecto VISMEDIC el objetivo de esta investigación es realizar un profundo estudio de las técnicas de corte.

La solución propuesta está basada en el método de corte progresivo con generación de ranura interna, el cual permitirá crear la ranura interna del corte siguiendo la trayectoria del instrumento virtual de corte de forma dinámica. Utilizándose tres tipos de cortes para cada triángulo: triangulación inicial, final e intermedio. Como resultado de la implementación se obtuvo un módulo de simulación de corte para mallas superficiales triangulares que permite crear la ranura interna del corte mediante la subdivisión de cada triángulo.

PALABRAS CLAVE

Mallas triangulares, modelos superficiales, simuladores virtuales, técnicas de corte.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE LAS TABLAS

INTRODUCCIÓN

Las nuevas condiciones actuales, el desarrollo científico-tecnológico, la interrelación con otras ciencias y sus métodos, el modo de vida de una sociedad altamente desarrollada y muchos más factores, han cambiado cualitativamente la problemática de la medicina. Sin duda alguna, el más significativo desarrollo tecnológico durante el último siglo, ha sido la construcción de computadoras de finalidades generales, capaces de hacer cosas que en el hombre se consideran como comportamiento inteligente. Como un lógico proceso de desarrollo, la medicina ha ido asimilando la introducción de las computadoras para agilizar y mejorar los procesos de apoyo médico, teniendo una gran influencia, la que sigue aumentando más cada día con la introducción de la Inteligencia Artificial. Podemos hablar entonces del surgimiento de la Informática Médica, que comprende una amplia gama de cuestiones de la organización y del uso de la información biomédica. El objetivo de la Informática Médica es reforzar y mejorar la toma de decisiones médicas y la atención al paciente, mediante el uso de sistemas informáticos para el diagnóstico médico, operaciones asistidas por computadora, así como el entrenamiento de médicos y estudiantes con el uso de simuladores virtuales. En este tipo de aplicaciones un eslabón fundamental lo constituye la visualización científica de las estructuras médicas; las cuales se referencian en la bibliografía por el concepto de visualización médica.

La visualización médica consiste en transformar los datos científicos obtenidos a partir de técnicas médicas como Tomografías Axial Computarizadas (TAC), Resonancias Magnéticas (MRI) o Ultrasonidos en imágenes digitales que pueden ser estudiadas y analizadas por los especialistas con mayor profundidad y detalle. La visualización de estas imágenes necesita una calidad elevada para que los diagnósticos emitidos por los especialistas sean lo más efectivos posibles.

El proyecto de Visualización Médica (VISMEDIC), perteneciente al centro de desarrollo de informática industrial (CEDIN) que se encuentra dentro de la facultad 5 de la Universidad de las Ciencias Informáticas se encarga de la visualización de modelos tridimensionales obtenidos a partir de imágenes médicas digitales. El objetivo principal de este proyecto es realizar un sistema que les permita a los médicos la realización de diagnósticos desde una perspectiva 3D. El proyecto realiza dos tipos de visualizaciones: la directa y la indirecta. En la visualización directa el volumen de datos se visualiza directamente sin

necesidad de utilizar representaciones intermedias previas a la visualización. En la visualización indirecta ocurre lo contrario, previo al proceso de visualización se genera una representación geométrica del modelo que se desea visualizar y posteriormente se realiza el proceso de visualización.

Dentro de las funcionalidades con las que debe cumplir este sistema se encuentra la realización de corte específicamente para tejidos del cuerpo humano. Actualmente el proyecto no cuenta con un módulo de corte que permita generar la ranura interna después de realizado dicho proceso. Lo que trae como consecuencia que no se puedan realizar las visualizaciones de las mallas a las cuales se les realiza el fenómeno virtual de corte

Ante la **situación problémica** planteada anteriormente, se define como **problema de investigación:** ¿Cómo visualizar las mallas superficiales triangulares después de efectuar el corte? Como resultado del problema de investigación, se define como el **objeto de estudio** el comportamiento topológico de los objetos virtuales al someterlos al corte.

Como **objetivo general** del trabajo se propone desarrollar un módulo que permita realizar el proceso de corte mediante la generación de ranura interna, y la visualización del mismo a través de las mallas superficiales triangulares. Este trabajo defiende la idea: con el módulo elaborado se podrán hacer cortes mediante la generación de ranura interna sobre mallas superficiales triangulares usadas en el proyecto VISMEDIC.

Como **campo de acción** se define las técnicas y algoritmos empleados para la simulación del corte.

Para darle cumplimiento a los objetivos planteados se proponen una serie de **tareas investigativas** que se muestran a continuación:

1. Elaboración del marco teórico a través del estudio del estado del arte existente sobre el tema.

2. Selección de los algoritmos para la detención de colisiones que se utilizarán en la ejecución del corte.

3. Selección del método que se implementará en la solución así como la estructura de datos utilizará en la misma.

4. Selección y empleo de la metodología de desarrollo de software RUP para realizar el análisis y diseño de la solución.

5. Implementación de las técnicas y algoritmos seleccionados para el desarrollo de un módulo informático.

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 utilizado para analizar la evolución y las tendencias actuales del proceso de corte.

Analítico – Sintético: Método Teórico utilizado para extraer y analizar la información sobre los principales métodos del corte.

Consultas de fuente de información: Método Empírico utilizado para la consulta de fuentes bibliográficas durante la investigación.

Pruebas: Método Empírico utilizado para validar los resultados obtenidos con la solución propuesta.

Observación: Método Empírico utilizado para observar los resultados obtenidos en la caracterización e identificación de los principales algoritmos utilizados, para poder decidir luego cuál o cuáles serán más adecuados.

Capítulo 1: Fundamentación Teórica.

Se hace un análisis de las técnicas usadas para la simulación del corte desde sus inicios hasta la actualidad.

Capítulo 2: Solución Propuesta.

Establece la solución resultante del análisis realizado en el capítulo anterior y se describen las técnicas de corte a utilizar.

Capítulo 3: Diseño y Análisis de la Solución Propuesta.

En este capítulo se definen 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: Implementación y Validación de los Resultados.

En este capítulo se definen 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 realizadas al módulo elaborado.

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 las principales técnicas de mallado que existen a nivel mundial, los diferentes métodos de deformación basados en física, se realizará un estudio de las diferentes tendencias del corte así como una breve panorámica sobre las colisiones entre objetos.

1.1. Representación geométrica de la malla.

Existen diferentes objetos que conforman una escena virtual y proporcionan a la vez una mejor visualización de los elementos reales (nubes, arboles, edificios, rocas, órganos, entre otros.). Dentro de los modelos que se usan para la representación de estos objetos se encuentran los modelos superficiales, volumétricos, basados en física, entre otros. En el trabajo se abordan temas relacionados con modelos superficiales y los modelos volumétricos.

1.1.1. Modelos de Superficie.

Estos modelos se caracterizan por representar el modelo tridimensional solamente en su superficie a través de vértices, aristas y polígonos de forma que cada arista es compartida como máximo por dos polígonos [23].

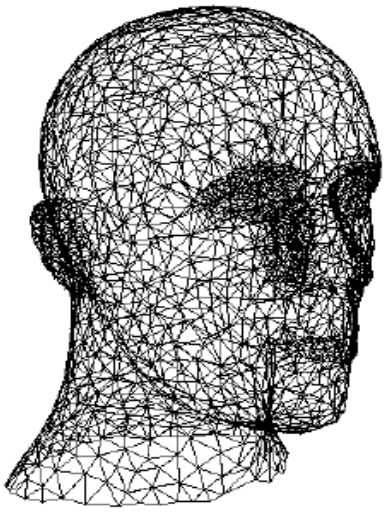

FIG 1: Representación de una malla superficial donde cada polígono es un triángulo.

1.1.2. Modelos Volumétricos.

Los modelos de mallas volumétricas son ampliamente utilizados para la representación de órganos y reconstrucción $3D$ a partir de imágenes. Estos simulan el comportamiento interior de los objetos, conformados por tetraedros, hexaedros o cubos entre otros elementos [23].

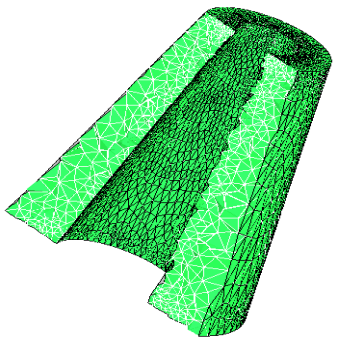

FIG 2: Representación de una malla volumétrica.

1.2. Algoritmos de mallado.

Lograr el almacenamiento de estos objetos virtuales en las mallas estudiadas en los epígrafes anteriores implica usar algoritmos de mallado. Los principales algoritmos para realizar este proceso se dividen en dos grupos fundamentales, los basados en triángulo/tetraedro y los basados en cuadrilátero/hexaedro. El primero es el más utilizado [24], por tanto en él encontramos las técnicas más usadas: Grafo Dual, Octree, Delaunay, Advancing Front, las cuales se describen a continuación.

1.2.1. Grafo Dual.

El grafo dual es una estructura eficiente para almacenar una malla con una forma muy peculiar. El grafo consta de una lista de vértices, otra de aristas y otra de caras. Estas 3 listas se entrelazan entre sí para conseguir búsquedas en aceptables tiempos de ejecución [1]. Cada cara tiene el índice de sus vértices y los índices de sus aristas, las aristas tienen los índices de cómo máximo dos caras y tiene los índices de sus dos vértices y cada vértice tiene las listas de los índices de las aristas al que pertenece así como las caras a las cuales él pertenece y sus vértices vecinos.

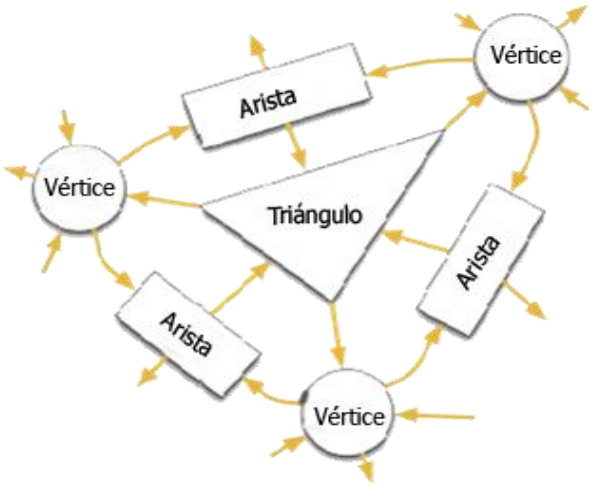

FIG 3: Representación de un Grafo Dual.

1.2.2. Octree.

Marck Shepard fue el primero en desarrollarlo en 1980. 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 Fig. 4). 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 esté 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 [2].

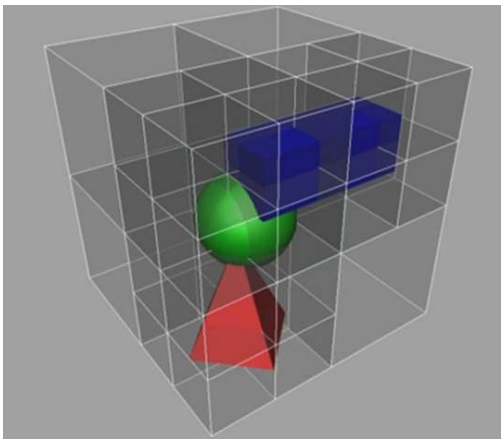

FIG 4: Representación de un Octree.

1.2.3. Delaunay.

Es la técnica más popular para generar mallas triangulares. La idea de Delaunay es realizar una triangulación sobre un conjunto finito de puntos de manera tal que esta triangulación quede de la forma más regular posible o sea que los ángulos que se formen sean agudos. Para lograr esto se utiliza el criterio de Delaunay que plantea que si se traza una circunferencia sobre un triángulo esta debe ser vacía, o sea no debe contener a ningún otro vértice del conjunto de puntos en su interior [24]. En la Fig. 5 se visualiza este criterio.

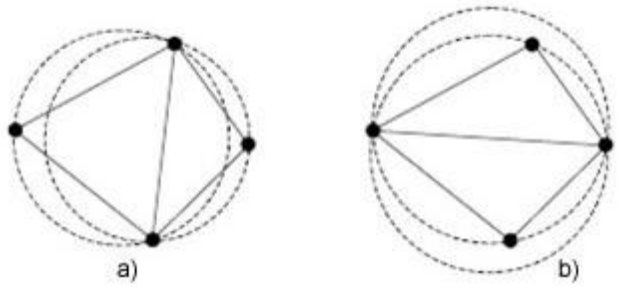

FIG 5: Criterio de Delaunay.

1.2.4. Advancing Front.

En este método el tetraedro es construido progresivamente hacia dentro desde la superficie triangulada del objeto. La Fig. 6 muestra un ejemplo en dos dimensiones de Advancing Front, donde los triángulos son formados desde la superficie, el algoritmo avanza desde el frente hasta rellenar toda el área del objeto con triángulos. En tres dimensiones, por cada cara triangular en el frente, es calculada la posición ideal de un

cuarto nodo que formaría el tetraedro. El algoritmo selecciona el cuarto nodo creado o de un nodo existente para formar el nuevo tetraedro, basado en cual podría formar el mejor tetraedro. También son necesarios los controles de intersección entre tetraedros para garantizar que no se superpongan al avanzar uno hacia el otro [24].

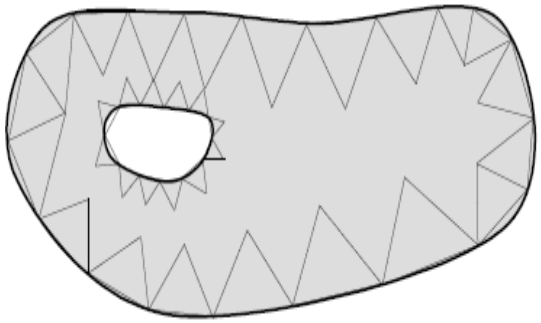

FIG 6: Ejemplo de Advancing Front 2D.

1.3. Modelos de Deformación Basados en Física.

Para lograr un aceptable realismo al realizar el corte se necesita de soluciones físico-matemáticas que soporten las deformaciones, sobre esto se han realizado muchos trabajos aportando novedosos resultados en sus más de 3 décadas de surgimiento. A continuación se describen los conceptos esenciales de los modelos físicos.

1.3.1. Sistema Masa-Resorte.

El sistema masa-resorte es un modelo físico con una sólida fundamentación matemática. Es computacionalmente ligero y relativamente pequeño [3] y apropiado para aplicaciones interactivas. Se plantea que con la estructura de los sistemas masa–resorte se pueden llevar a cabo largas deformaciones y modificaciones topológicas. Es usado ampliamente para modelar objetos blandos, consiste en la discretización de los objetos como una malla de partículas y muelles, así como en animaciones faciales estáticas y dinámicas [4] [5], también han sido utilizadas para la simulación de ropa, video juegos y películas de animados.

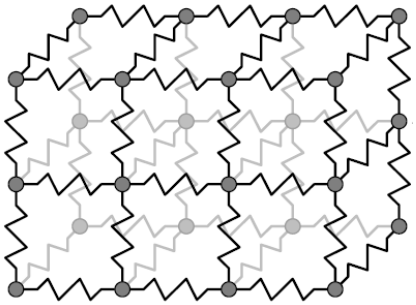

FIG 7: Sistema masa-resorte.

1.3.2. Método De Elementos Finitos (FEM).

Es uno de los métodos más populares en las ciencias de la computación para resolver ecuaciones diferenciales en rejillas irregulares. El artículo publicado por Turner, Clough, Martin y Topp en 1956 es reconocido como el inicio del actual Método de Elementos Finitos [6].

En esencia el método consiste en dividir el objeto en un conjunto finito de elementos mediante discretización geométrica y luego las propiedades físicas del objeto son interpoladas para cada elemento usando funciones de forma, de manera que la mecánica continua del objeto es expresada en términos de un conjunto de elementos. Cada uno de los elementos es asociado para obtener un conjunto de ecuaciones que representan la física del objeto [7].

Bro-Nielsen y Cotin trabajan con FEM linealizado para simulación de cirugía. Ellos lograron un aumento de velocidad simulando solo los nodos visibles de la superficie, similar al método de Elementos de la Frontera (*Boundary Element Method, BEM*).De esta manera se obtienen resultados en tiempo real [8].

Este método transforma la mecánica continua de la deformación en un problema individual que puede ser resuelto usando el análisis numérico. Este método descompone el modelo en pequeños polígonos o poliedros: triángulos en $2D$ y tetraedros en $3D$, ver en Fig. 8.

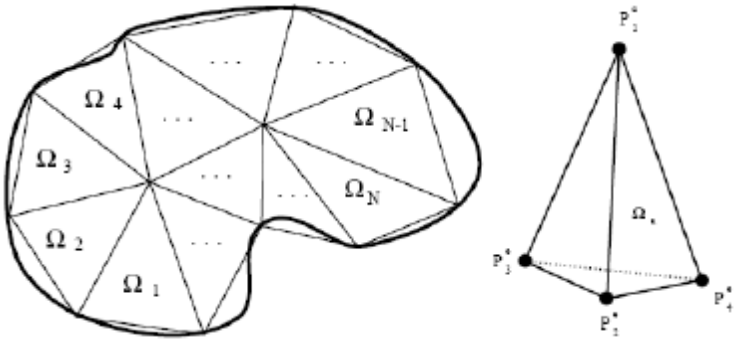

FIG 8: Discretización del Dominio.

La principal ventaja de FEM comparada con otras aproximaciones, es que puede producir simulaciones más realistas físicamente, debido a que las matrices de masa y rigidez permanecen constantes en el tiempo. Sin embargo, requiere de muchos cálculos, que sólo pueden ser reducidos disminuyendo el número de elementos, lo que atenta contra la exactitud del modelo [9].

1.3.3. Método De Elementos de Frontera.

El Método de los Elementos de Frontera (*Boundary Element Method*, BEM), es una alternativa interesante al FEM estándar, porque todos los cálculos se realizan en la superficie del cuerpo elástico en lugar de su volumen como se representa en la Fig. 9. El método logra un aumento sustancial de la velocidad debido a que el problema tridimensional original, es reducido a dos dimensiones. Sin embargo, solo puede ser aplicado en cuerpos cuyo interior esté compuesto por un material homogéneo, además, cambios de topología son más complejos de manipular que en FEM Explícito [7].

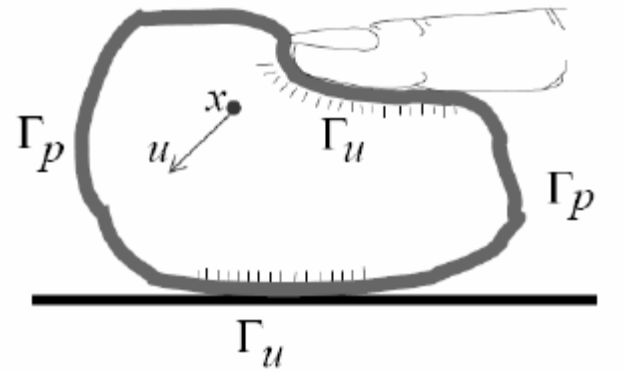

FIG 9: Notación de BEM y sus Condiciones de Frontera.

Este método fue propuesto por primera vez para simular objetos deformables por Doug L. James y Dinesh K. Pai en 1999 [10], en el año 2003 se propone otro trabajo agregando posibilidades para el cambio de las condiciones de la frontera, como colisiones órgano – órgano [11].

Entre las ventajas que ofrece este método vale resaltar que es más preciso que FEM para calcular fuerzas de contacto y quizás la mejor opción que ofrece BEM es que usa la misma discretización usada para el render, es decir, no se necesita otro enmallado [10].

1.4. Colisiones.

La interacción entre los objetos virtuales dentro de un ambiente dinámico es inevitable. A esta interacción, choque o contacto entre dos elementos virtuales se le denomina colisión, la detección de estas colisiones es el primer paso para realizar una interacción realista [23].

Fases de la Detección de Colisiones.

En un ambiente virtual donde están presentes objetos rígidos o deformables la estrategia para detectar colisiones se divide en tres fases.

Fase Amplia: Son seleccionados el par de objetos que probablemente colisionan en esta primera fase de optimización. Los algoritmos empleados en esta fase principalmente están basados en descomposición espacial.

Fase Estrecha: Es otra fase de optimización pero en esta se da una idea preliminar acerca de la región donde los objetos pudieran colisionar, a lo que denominados zona de colisión.

Fase Exacta: En esta se obtiene la entidad exacta de colisión. El algoritmo de esta fase chequea la colisión entre dos polígonos. En esta fase el algoritmo de colisión usado determina si al menos dos primitivas se interceptan. Fundamentalmente son considerados estas primitivas: esfera, caja, triángulo.

1.5. Tendencias Del Corte.

Como comentamos anteriormente una de las tareas básicas de la simulación quirúrgica es el fenómeno virtual del corte, este es visto como la interacción de un modelo deformable y un cuerpo rígido que sería la herramienta virtual de corte. A continuación se explican los 4 métodos que han sido desarrollados mundialmente.

1.5.1. Método Destructivo.

Es el método más antiguo de corte, basado en el principio de la destrucción, es decir que se elimina el elemento que esta colisionando con la herramienta virtual de corte. Este método es muy fácil de implementar, ahora la refinación de las mallas después de realizar dicho corte si tiene gran complejidad sin embargo en el trabajo realizado por Forest, Delingette Ayache [12] se aborda bastante el refinamiento de la malla sobre todo en mallas volumétricas, estos algoritmos son rápidos ya que se eliminan elementos en las estructuras de datos, decir en contra del método que no mantiene las leyes físicas de la conservación de la masa. Es recomendable utilizar este método donde realmente se destruye el tejido.

En 1997 Cotin utilizó un modelo Masa-Resorte para la deformación y el corte [13]. Años más tarde, en el trabajo realizado por S. Cotin, H. Delingette, y N. Ayache [14], retoman este método y describen tres diferentes modelos físicos, dentro de los cuales uno es muy similar al modelo masa-resorte y la operación de corte que se le realiza puede ser simulado en tiempo real. Además C.Forest, H.Delingette y N.Ayache en su documento profundizan en este método para mallas volumétricas, describiendo todo el proceso de refinamiento de las mismas, planteando soluciones y dándose a la tarea fundamental de disminuir el tiempo computacional y lograr tetraedros de óptima calidad.

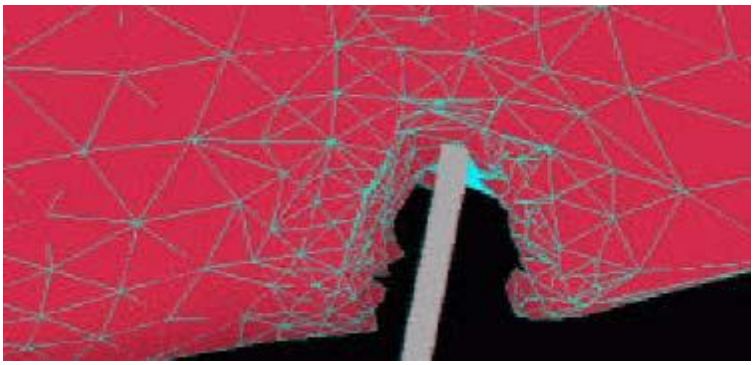

FIG 10: Ejemplo de Método Destructivo.

1.5.2. Método de Separación.

Otra idea para simular el corte es el método de separación, este ajusta la superficie de corte a nodos significativos en el modelo, el método se basa en el principio de separar las primitivas. Las primitivas que son colisionadas por la herramienta virtual, se duplican los nodos significativos para el corte y se procede a la separación. Boux de Cason utiliza este corte para una malla superficial $2D$. El modelo físico usado fue Masa-Resorte y más tarde aplica este trabajo en un framework de operaciones [15]. Nienhuys basa su algoritmo de corte en el método de separación, utiliza un modelo linear de elementos finitos [16]. Luego Nienhuys profundiza en este método basándose en un modelo de elementos finitos y con una malla volumétrica representada por tetraedros, este trabajo describe tres partes importantes del desarrollo del corte por la vía de la separación: La selección de la superficie de corte, la ruptura de los nodos y la eliminación de las degeneraciones [13]. Mendoza y Laugier desarrollan un importante trabajo sobre el método de separación generalizando el método para distancias largas de desplazamiento, pues la idea de Nienhuys no consideraba hasta el momento este aspecto requerido para el proceso de corte [17].

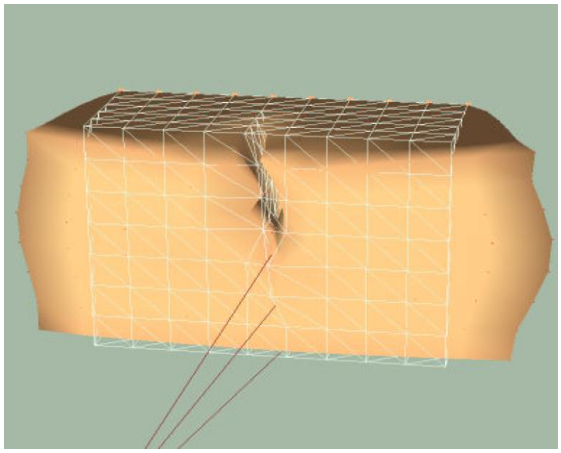

FIG 11: Ejemplo del Método de Separación.

1.5.3. Método de Subdivisión.

A finales de los 90, Bielser pone en práctica un nuevo método de corte, el cual basa su principio en la subdivisión de los elementos colisionados por la herramienta virtual de corte, en este caso el bisturí es modelado como instrumento de corte.

Este método cumple con el principio físico de conservación de la masa, no requiere de extensos procesos de refinamiento de la malla, sin embargo, aumenta el número de elementos, lo que provoca lentitud en el ciclo de la simulación y baja calidad en los nuevos elementos.

Fue introducido por primera vez en el contexto médico cuando Mazura y Seifert [18], plantean el corte de una malla tetraédrica con planos predefinidos. Bielser en su trabajo [19] usa el modelo masa-resorte en una malla tetraédrica para simular el corte y la deformación. Este algoritmo fue refinado más tarde por Mor y T. Kanade [20], abordando temas como la incisión parcial de la malla, emplean un modelo masa-resorte, proponen un corte progresivo y minimizan el número de elementos creados. Posteriormente, D Bielser, P. Glardon, M. Teschner, M. Gross [21] plantean una máquina de estados, la cual sigue el patrón topológico y controla actualizaciones necesarias de cada tetraedro. Esto da lugar a un algoritmo rápido para la simulación dinámica de la trayectoria volumétrica muy exacta en tiempo real.

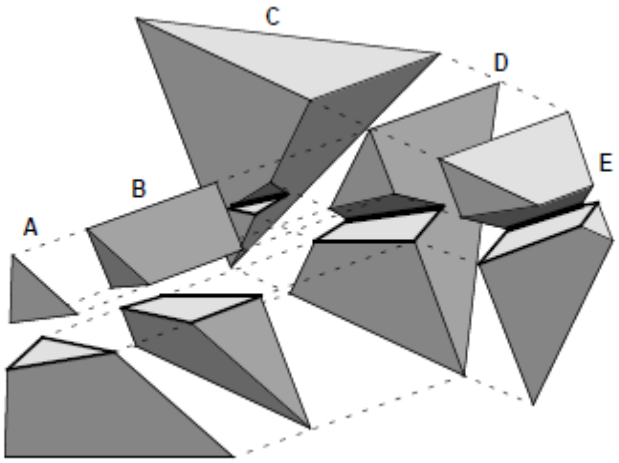

FIG 12: Los 5 estados posibles en el método de subdivisión.

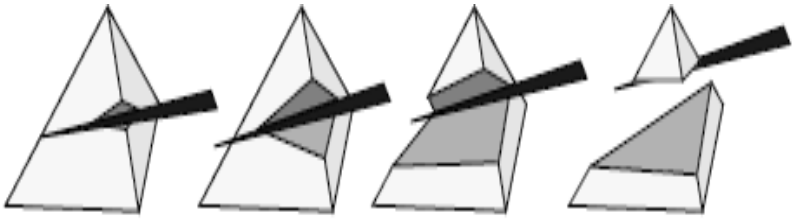

FIG 13: Ejemplo de corte.

1.5.4. Método de Corte Progresivo con Generación de Ranura Interna.

Un trabajo novedoso sobre el campo de la simulación del corte es la adaptación del corte en un modelo superficial a un modelo volumétrico. En el trabajo realizado por Hui Zhang, Shahram Payandeh y John Dill [22], se propone un modelo masa-resorte, con una base sólida, para simular el funcionamiento de un corte virtual usando un dispositivo de entrada. Se introduce un novedoso y nuevo algoritmo para subdividir la superficie y generar la estructura interna siguiendo el movimiento del dispositivo empleado. Cuando el instrumento cortante penetra el objeto estamos en presencia de una operación de corte, en el trabajo referenciado se definen dos estados para la misma:

• Estado de contacto: cuando el objeto no ejerce fuerza suficiente para penetrar el objeto. Aquí se deforma el objeto.

• Estado de penetración: cuando es penetrado el objeto, es decir, cuando se logra el corte.

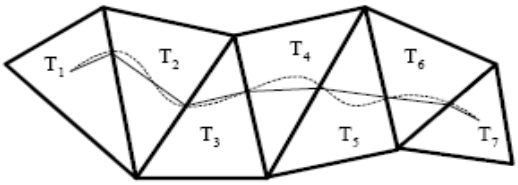

FIG 14: Ejemplo de trayectoria de un corte.

En la Fig. 15 podemos ver cómo se lleva a cabo el proceso del corte, los puntos G_1 y G_2 dos puntos que hay que crear que son el estado de penetración del instrumento de corte, también el punto donde se corta a BC es duplicado para formar la triangulación que se muestra y que sea creada la ranura, después se le aplica el modelo masa resorte para alcanzar la visualización mostrada en el estado final.

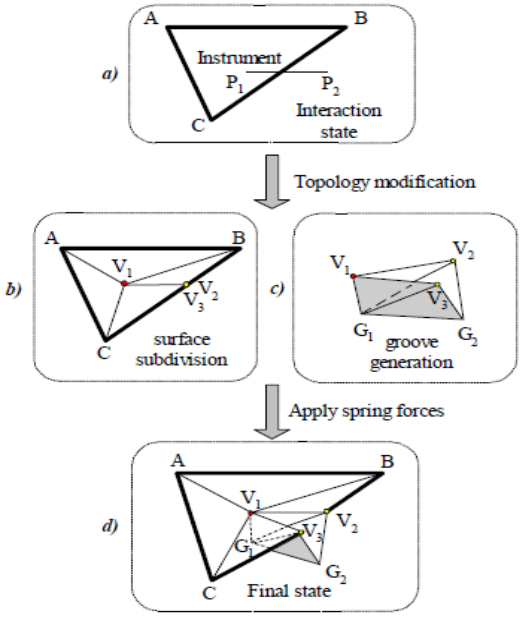

FIG 15: Estado de los triángulos inicial y final.

En la Fig. 16 podemos ver cómo se lleva a cabo el proceso del corte, los puntos G_1 y G_2 son el estado de penetración del instrumento de corte, también los puntos P_1 y P_2 son duplicados, se triangula y después se le aplica el modelo masa resorte para alcanzar la visualización mostrada en el estado final.

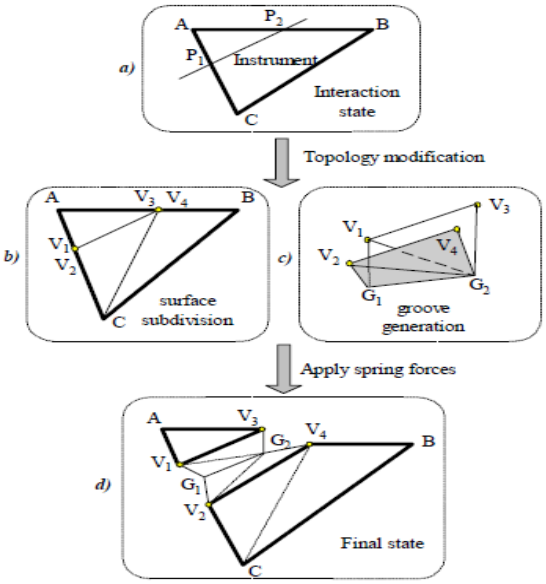

FIG 16: Estado de los triángulos intermedios.

Para lograr el proceso de corte de forma progresiva se plantean dos soluciones:

• Corte progresivo con subdivisión temporal, donde se genera una ranura interna temporal, en la cual se trata cada triángulo como un estado final del corte, subdividiéndolo temporalmente según donde pase el instrumento cortante.

• La unión de dos cortes, permitiendo realizar un proceso cíclico de corte.

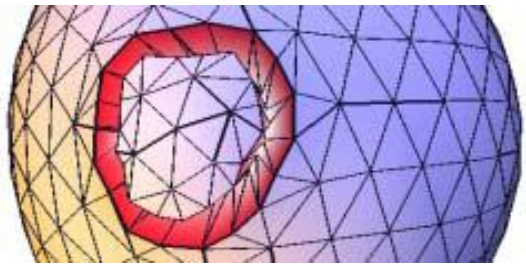

FIG 17: Proceso cíclico de corte.

1.6. Conclusiones Parciales.

Con el estudio realizado en este capítulo se llega a las conclusiones parciales que en el presente trabajo se utilizarán las mallas superficiales triangulares, para el almacenamiento de estas mallas se escogió el Grafo Dual debido a su fácil adaptación a las mallas superficiales y las ventajas que esta brinda, anteriormente explicadas. De los tres

aspectos fundamentales para realizar el corte, como son deformaciones basadas en física, técnicas para realizar el corte y detención de colisiones el trabajo se basa en la segunda de las tres y se utilizará el mouse como instrumento de corte. De las cuatro tendencias de corte estudiadas se utilizará la última tendencia, o sea se desarrollará un módulo que permita generar la ranura interna después de realizado el corte.

CAPÍTULO 2: SOLUCIÓN PROPUESTA

En este capítulo se describe en detalles el proceso de corte en mallas superficiales triangulares, el cual consta de tres pasos que se mencionan a continuación:

1. Transformar la geometría de entrada en un grafo dual.
2. Detección de colisiones.
3. Proceso general del corte.

2.1. Transformar la Geometría de Entrada en un Grafo Dual.

En la realización del trabajo se opta por el grafo dual para el almacenamiento de la malla ya que dicho grafo permite hacer buenas optimizaciones y consultas rápidas, además hace un uso eficiente de memoria al no repetir vértices, inicialmente la malla se carga y nos brinda una lista de vértices y una lista de triángulos y el grafo dual se quiere que quede de la siguiente manera:

Debe contener una lista de vértices, una lista de triángulos y una lista de aristas. La relación que debe existir entre estas listas son las siguientes:

Triángulo: Contiene los índices de los vértices que conforman dicho triángulo así como los índices de las aristas y también los índices de los triángulos vecinos a él o sea que comparten una arista.

Vértice: Contiene una lista de vértices vecinos a dicho punto que son los que comparten aristas con este, una lista de arista que lo contienen, así como una lista de triángulos donde este punto es común para todos ellos.

Arista: Contiene los índices de los 2 vértices que la forman así como el o los triángulos a la que ella pertenece.

Para el mejor entendimiento de cómo sería la construcción del grafo se muestra el siguiente seudocódigo:

Algoritmo 1:

Entrada:

Lista de vértices: $V = \{V_1, V_2, \ldots, V_n\}$.

Lista de triángulos: $T = \{T_1, T_2, \ldots, T_n\}$. *Donde* $T_i = \{V_j, V_k, V_l\}$ *y* $0 <= j, k, l < n$.

Salida:

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

Para todo T_i en T **hacer**:

Se realiza las actualizaciones de vecindad para los triángulos, vértices y las aristas de T_i*, en caso de que se cree alguna arista nueva se almacena en lista de aristas A.*

Fin

Para entender cómo se adicionan las aristas y se maneja el tema de las vecindades se muestra el siguiente seudocódigo:

Algoritmo 2:

Entrada:

Lista de vértices: $V = \{V_1, V_2, \ldots, V_n\}$.

Lista de Triángulos: $T = \{T_1, T_2, \ldots, T_n\}$. *Donde* $T_i = \{V_j, V_k, V_l\}$ *y* $0 <= j, k, l < n$.

Salida:

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

de aristas.

Para todo V_j, V_k de T_i **hacer**:

Si *no existe la arista correspondiente entre* V_j, V_k **entonces**:

Se crea la arista y se adiciona en A.

Se crea la relación de vecindad entre V_j, V_k.

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

Se crean la relaciones entre T_i *con* V_j, V_k.

Fin

Sino

Se toma la posición de la arista formada entre los dos puntos V_j, V_k.

Se crea la relación de dicha arista con T_i *y viceversa.*

Se crean la relaciones entre T_i *con* V_j, V_k.

Se crea la relación de vecindad entre T_i *y el otro triángulo que contiene a la arista.*

Fin

Fin

2.2. Detección de Colisiones.

En el presente trabajo la detección de colisiones se realiza de forma sencilla, solo para dar soporte a una interfaz visual que permita observar el proceso del corte. El instrumento virtual del corte es el mouse y el corte se realizará mientras el scroll se mantenga pulsado. La colisión entonces entre el objeto cortante y la malla superficial sería cuando este se encuentre sobre algún triángulo de la malla. El mouse arroja unas coordenadas (x, y) que son la posición en píxeles de la pantalla, luego se procede a dada esta posición de la pantalla capturar la posición (x, y) del mundo mediante una función de OpenGl.

Para realizar la colisión inicial se recorre la lista de triángulos de la malla para cada triángulo se chequea en el plano $2D$ si la posición (x, y) del mundo está dentro de este triángulo, en caso de que esté se calcula la intersección del plano formado por los 3 vértices del triángulo con la recta que pasa por el punto (x, y) y que es paralela al eje z, para cada punto calculado se obtiene el triángulo que arroje una mayor z y este sería el triángulo de la malla con el cual el objeto de corte estará colisionando inicialmente. Luego solo para ver con que triángulo colisiona el instrumento de corte al continuar el mismo sería solo verificar en $2D$ si la posición (x, y) del mundo real que representa el objeto de corte está contenido en algún triángulo vecino del último triángulo colisionado.

Para saber si un punto P está dentro de un triángulo con tres vértices P_1, P_2 y P_3 se procese a calcular el giro de los triángulos P_1P_2P, P_2P_3P y P_3P_1P, si estos 3 valores tienen signos iguales entonces el punto P se encuentra dentro del triángulo. Para calcular el giro entre 3 puntos $P\,(x, y)$, $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$ se calcula el valor del determinante de la siguiente matriz:

$$D = \begin{vmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ 1 & 1 & 1 \end{vmatrix}$$

El valor D sería el valor del giro y a la vez el módulo de $D/2$ es el valor del área del triángulo formado por estos vértices.

Para calcular la intersección del plano formado por los vértices de un triángulo y una recta que pasa por las coordenadas (x_1, y_1) y a la vez es paralela al eje Z, se procede de la siguiente manera, se sustituyen x_1 y y_1 en (1) y se despeja z_1, luego la intersección sería $P_1(x_1, y_1, z_1)$.

$$ax_1 + by_1 + cz_1 = d \qquad (1)$$

Para obtener a, b y c se halla la normal del triángulo y la primera componente es a, la segunda es b y la tercera c.

Dado un triángulo con puntos P_1, P_2 y P_3 para calcular la normal se crean dos vectores $v1$ y $v2$ donde $v1 = P_1 - P_2$ y $v2 = P_1 - P_3$, luego el vector normal $v = v1\ cross\ v2$.

2.3. Proceso General del Corte.

El proceso general de corte es dinámico, cada triángulo con que el objeto virtual del corte colisiona es tratado como un estado final del corte en caso de que el corte continúe entonces este triángulo pasa a ser un estado de corte intermedio, en el triángulo inicial si se realiza un solo corte que es el corte inicial, los pasos para realizar dicho proceso son los siguientes:

Paso 1: Se chequea colisión entre el objeto virtual del corte y la malla.

Mientras no exista colisión se continúa en el Paso 1 si no se pasa a la ejecución del Paso 2.

Paso 2: Se chequea colisión entre el objeto virtual del corte y el triángulo seleccionado.

Mientras exista colisión se continua este paso en caso contrario entonces se pasa a la ejecución del Paso 3.

Paso 3: Se chequea colisión entre el objeto virtual del corte y los vecinos del triángulo seleccionado.

En caso de que no exista la colisión entre el objeto virtual del corte y algún vecino entonces terminamos el corte y pasamos al Paso 1 en espera de un nuevo corte.

Paso 4: Realizar corte inicial en caso de que el triángulo seleccionado sea el primer triángulo del corte o eliminar corte final y realizar corte intermedio en otro caso, luego el nuevo triángulo vecino que esta colisionando con el objeto de corte pasa a ser el triángulo seleccionado y se realiza sobre este el corte final.

Cuando se termine de ejecutar este paso retornamos a la ejecución del Paso 3.

2.3.1. Corte Inicial para cada Triángulo.

Bueno el corte inicial solo se realiza en el primer triángulo con que el objeto virtual del corte colisiona con la malla. Inicialmente este triángulo tiene su forma normal o sea no se ha hecho ninguna trasformación en este, entonces se realiza el corte desde P_1 hasta P_2, se crea el vértice V_1 que sería el punto inicial de donde se comienza el corte luego se obtiene un punto temporal G_1 que es la intersección del segmento P_1P_2 con BC, con este

punto se crean V_2, V_3, a continuación obtenemos un vector que es la suma de las normales del triángulo ABC y el otro triángulo que tiene a BC como arista común, luego este vector se multiplica por -1 y se normaliza entonces con este vector y el punto G_1 le damos profundidad, el punto G_2 que quedaría de la siguiente forma $G_2 = G_1 + un\ escalar * el\ vector\ calculado$. Después de la creación de los puntos se crean las aristas $V_1A, V_1B, V_1C, V_1V_2, V_1V_3, V_1G2, V_2B, V_3C$ y a continuación se crean los triángulos $V_1AB, V_1AC, V_1V_2B, V_1V_3C, V_1V_2G2\ y\ V_1V_3G2$. Ver en la Fig. 18.

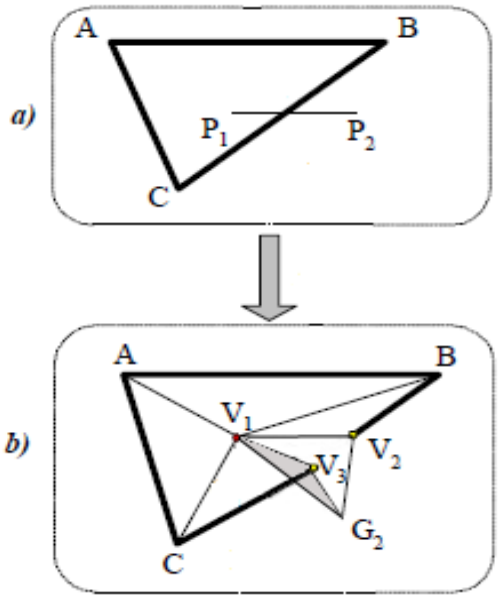

FIG 18: Proceso de Corte Inicial.

2.3.2. Corte Final para cada Triángulo.

El corte final se realiza en todos los triángulos con que el objeto de corte tiene colisión excepto el primero pero al final del corte solo uno o sea el último triángulo colisionado es el que queda con la configuración que se explicará a continuación. El procedimiento en este corte es bastante similar al corte inicial lo que con la única diferencia de que los vértices $V_2, V_3\ y\ G_2$ están creados al igual que las aristas $V_2B, V_3C, V_2G_2 y\ V_3G_2$. Se crea el vértice V_1 que sería el vértice donde el objeto virtual de corte esta colisionando el triángulo ABC, se crean las aristas $V_1A, V_1B, V_1C, V_1V2, V_1V_3\ y\ V_1G_2$, se crean entonces los triángulos $V_1AB, V_1AC, V_1V_2B, V_1V_3C, V_1V_2G_2\ y\ V_1V_3G_2$. Ver en la Fig. 18.

2.3.3. Corte Intermedio para cada Triángulo.

Este corte se realiza para todos los triángulos excepto para el primer y último triángulo del corte. Cuando se va a realizar el corte el triángulo se encuentra como se muestra en la Fig. 19 o sea este triángulo anteriormente fue tratado como un estado final del corte lo que al continuar el mismo como es dinámico hay que eliminar este estado y ponerlo en estado intermedio, primeramente se elimina el vértice V_1, las aristas V_1A, V_1B, V_1C, V_1V_2, V_1V_3 V_1G_2 y los triángulos V_1AB, V_1AC, V_1V_2B, V_1V_3C, $V_1V_2G_2$ y $V_1V_3G_2$. Ver en la Fig. 19.

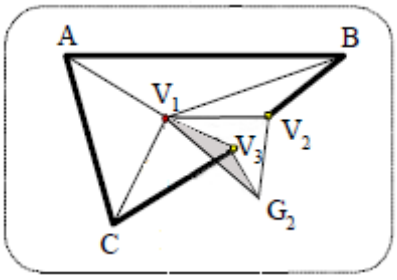

FIG 19: Estado del triángulo ABC después de realizar un Corte Final.

Luego tenemos que los vértices V_3 , V_4 y G_2 y las aristas AV_3, V_3G_2, G_2V_4 y V_4B están creadas entonces se crean los puntos V_1, V_2 y G_2 este último se crea de la misma manera que se explicó en el corte inicial para cada triángulo, luego se crean las aristas AV_1, V_1G_1, G_1V_2, V_2C, V_1V_3, V_2V_4, V_1G_2,V_2G_2, G_1G_2 y CV_4 y los triángulos AV_1V_3, V_1V_3G2, $V_1G_1G_2$, $V_2V_4G_2$, $V_2G_1G_2$, V_2V_4C y BCV_4. Ver en la Fig. 20.

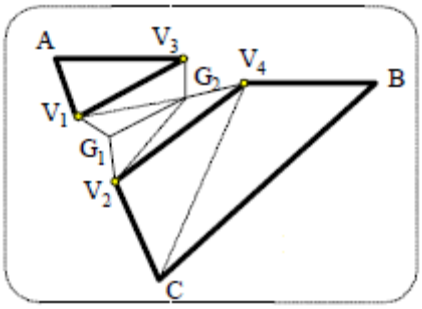

FIG 20: Estado del triángulo ABC después de realizar un Corte Intermedio.

2.4 Herramientas.

Para implementar la solución propuesta se utilizaron algunas herramientas que se describirán en este epígrafe para tener una idea más profunda sobre la implementación realizada.

Como herramienta de modelado se utilizó Visual Paradigm. Esta es una herramienta que utiliza UML como lenguaje de modelado. Es considerada una herramienta muy completa, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Ofrece además la capacidad de ingeniería directa (versión profesional) e inversa. Es un producto de probada calidad que soporta varios idiomas y con licencia gratuita y comercial.

Como entorno de desarrollo integrado se utilizó *Microsoft Visual Studio.NET 2008*. Este entorno de trabajo ofrece a los desarrolladores un conjunto unificado, orientado a objetos, jerárquico y extensible de bibliotecas de clases (API). Este cuenta además con herramientas como el Visual Assist que permiten el completamiento de código y contribuyen a la codificación de la aplicación.

Para el diseño y programación de la GUI de la aplicación se utilizó el framework Qt. Este es un framework de alta compatibilidad, tiene características que lo hacen muy versátil y el código C++ que utiliza tiene un alto rendimiento. El código fuente está disponible y así como una muy buena documentación que hace de este una opción muy buena para los desarrolladores. Tiene una arquitectura muy flexible que permite diseñar aplicaciones sin mucho esfuerzo y con una gran calidad. Tiene un apoyo técnico de alta calidad y sigue el principio de reutilizar el código para crear más y hacer despliegues sin importar el lugar.

2.5 Lenguajes.

El lenguaje de programación seleccionado fue C++. Este es uno de los lenguajes más potentes para desarrollar aplicaciones porque permite programar a un alto nivel y tiene mecanismos muy buenos como la herencia y el polimorfismo que le brindan al desarrollador gran flexibilidad para diseñar. Al ser un lenguaje orientado a objetos permite encapsular los datos y los métodos en clases, esto posibilita obtener un código más seguro y con una mejor organización. Otra ventaja muy importante que se tuvo en cuenta es que este tipo de aplicaciones manejan grandes volúmenes de datos y necesitan

interacción en tiempo real y no todos los lenguajes posibilitan esto: en el caso de C++ es un candidato ideal por su gran rapidez.

El lenguaje seleccionado para modelar el análisis y diseño de la aplicación fue UML. Este permite verificar y validar el modelo realizado. Es un lenguaje independiente de la plataforma y constituye un estándar mundial en el modelado. Este brinda además la posibilidad de modelar sistemas utilizando técnicas orientadas a objetos (OO) así como la documentación de todos los artefactos de desarrollo. Puede conectarse con los lenguajes de programación y posibilita la ingeniería inversa.

2.6 Metodología de desarrollo de software.

Como metodología de desarrollo de software se escogió Proceso Unificado para Desarrollo de Software (RUP). Es una metodología que acumula muchos años de experiencia por lo que está probada a nivel mundial.

Guiado por Casos de Uso: La razón de ser de un sistema software es servir a usuarios ya sean humanos u otros sistemas. Un caso de uso es una facilidad que el software provee a sus usuarios. Los casos de uso constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.

Centrado en la arquitectura: La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas de software, sistemas operativos, manejadores de bases de datos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Los casos de uso guían el desarrollo de la arquitectura y la misma se retroalimenta en los casos de uso.

Iterativo e incremental: Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un miniproyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Este capítulo tendrá como objetivo el análisis y diseño de la solución propuesta. Definiéndose varios aspectos dentro de los que se encuentran las reglas del negocio, modelo de dominio, la especificación de requisitos funcionales y no funcionales. Esta última será utilizada en la elaboración del Modelo de Casos de Uso (CU) y en la descripción de los CU. Además se definirán los actores del sistema. Se generaran diferentes artefactos dentro de los que se encuentran el diagrama de dominio, diagrama de CU, y los diagramas de secuencias.

3.1. Reglas del Negocio.

A continuación se mencionan las restricciones correspondientes al módulo que se implementará.

1. Los modelos que se deseen visualizar para realizar el corte deben estar en formato (*.stl) o en formato (*.ase).

2. 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 sometidos al fenómeno virtual del corte.

3.2. Modelo del Dominio.

Como parte de la solución propuesta no es necesario Modelar el negocio, ya que los procesos que componen el mismo no se definen claramente. Por lo que se utilizó el Modelo de Dominio.

El Modelo de Dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de software. En la Fig. 21 se muestra el Modelo del Dominio elaborado.

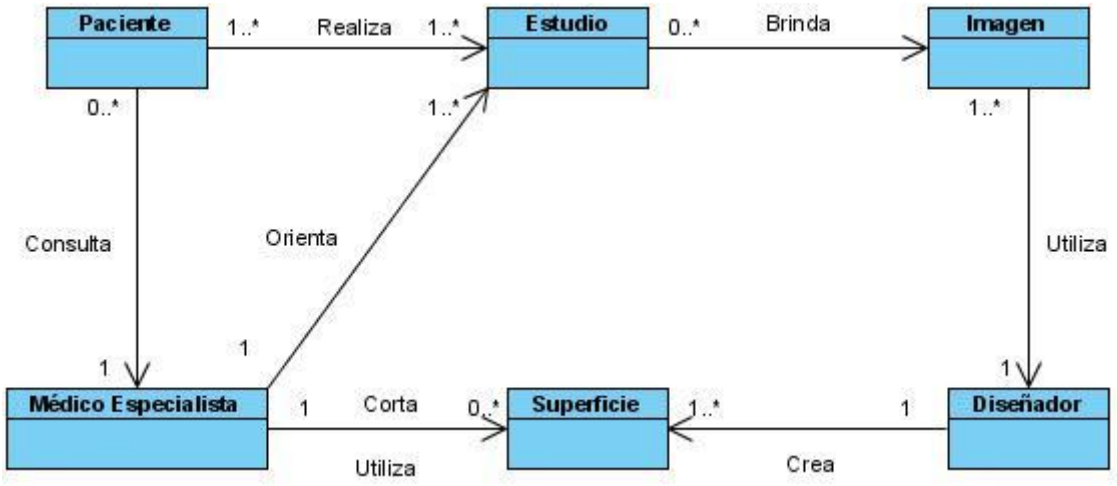

FIG 21: Modelo del Dominio.

3.2.1. Descripción del diagrama del Modelo de Dominio.

El Médico especialista le orienta al paciente un estudio el cual brinda una serie de 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 médico especialista puede realizar un corte sobre la superficie en caso de que el paciente tenga que ser intervenido quirúrgicamente.

3.3. Captura de Requisitos.

La captura de requisitos es quizás la etapa más importante en el desarrollo de software, y una de las más descuidadas. Esta disciplina va adquiriendo relevancia cuando se examinan los indicadores frente a fracasos de proyectos por la ineficiente educción de requisitos. La Ingeniería de Requisitos según Loucopoulos (Loucopoulos et al., 1989) y la IEEE (IEEE, 1990) es un proceso cuyo objetivo final es plasmar en un documento las necesidades reales que clientes/usuarios esperan satisfacer con la construcción de un sistema. De otro lado, Zave y Jackson (Zave et al, 1997) la definen como una rama de la Ingeniería de software que apoya al analista de sistemas en su tarea de traducir los objetivos del mundo real a funciones, restricciones y requisitos de manera que dicha traducción sea consistente con las necesidades de los clientes, y que no se obtengan resultados equivocados o de mala calidad.

3.3.1 Requisitos Funcionales.

Un requerimiento es una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, norma, especificación u otro documento formal, facilitando el entendimiento entre clientes y desarrolladores. A continuación se mencionan los requisitos funcionales y no funcionales, por los cuales se rige el desarrollo del módulo.

RF1. Cargar modelo.

-Permite que se cargue el modelo con formato *.stl y *.ase.

RF2. Mostrar modo *Wireframe.*

-Permite visualizar la triangulación de la malla.

RF3. Rotar modelo.

-Permite que se rote el modelo seleccionado por los tres ejes de coordenadas.

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.

-Permite trasladar el modelo seleccionado por los tres ejes de coordenadas.

RF4.1 Trasladar modelo sobre el eje X.

RF4.2 Trasladar modelo sobre el eje Y.

RF4.3 Trasladar modelo sobre el eje Z.

RF5. Administrar corte.

-Permite cortar el modelo anteriormente cargado. Esta es la principal funcionalidad con la que deberá contar el módulo a implementar.

3.3.2. Requisitos no Funcionales.

Los requisitos no funcionales sólo describen atributos del sistema o atributos del entorno del sistema. Los requisitos no funcionales con los que deberá contar el módulo son los siguientes:

RnF1. Software

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

RnF2. Hardware

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

La memoria RAM será de 1GB.

RnF3. 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 lograr representar la anatomía humana.

Integridad: No debe haber pérdidas de información ni de calidad en las imágenes obtenidas durante el proceso de corte.

RnF4. Interfaz Externa

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

RnF5. Restricciones en el 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.

La forma en que los actores usan el sistema es representada a través de los casos de usos. El artefacto que responde a esta necesidad es conocido como descripción de casos de uso.

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.

Actor	Justificación
Médico Especialista	Interactúa con el sistema para ejecutar las funcionalidades de: Cargar modelo, Rotar modelo, Trasladar modelo y Cortar modelo.

Tabla 1: Actores del sistema.

3.4.2. Diagramas de Casos de Uso.

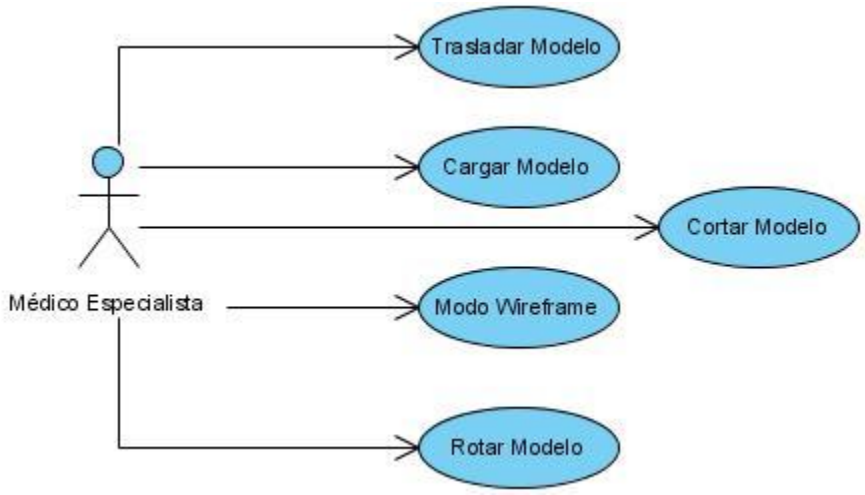

FIG 22: Diagrama de Casos de Uso.

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.

A continuación se relacionan las tablas que se corresponden a las descripciones de los CU, argumentándose los flujos operacionales de cada uno.

Caso de Uso:	Cargar Modelo
Actor:	Médico Especialista (inicia)

Propósito:	Seleccionar el modelo a cargar donde se realizará el corte.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de cargar el modelo y termina cuando se visualiza el modelo seleccionado.
Referencia:	RF1
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción cargar modelo.	1.1 Muestra un cuadro de diálogo que le permite al usuario seleccionar el o los ficheros que desea cargar. 1.2 El cuadro de diálogo solo mostrará los ficheros con las extensiones especificadas anteriormente.
2. Selecciona el fichero que desea cargar y oprime el botón de aceptar.	2.1 Se cierra el cuadro de diálogo y se procede a cargar el modelo seleccionado. Termina el caso de uso.
Post-condiciones:	Se visualiza el modelo previamente seleccionado.
Prioridad:	Crítica.

Tabla 2: Caso de Uso Cargar Modelo.

Caso de Uso:	Mostrar modo Wireframe
Actor:	Médico Especialista (inicia)
Propósito:	Mostrar el mallado del modelo visualizado.

Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de Wireframe y termina cuando se muestra el mallado del modelo.
Referencia:	RF2
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción Wireframe.	1.1 Se visualiza el mallado del modelo. Termina el caso de uso.
Post-condiciones:	Se visualiza el mallado del modelo.
Prioridad:	Secundario.

Tabla 3: Caso de Uso Mostrar modo Wireframe.

Caso de Uso:	Rotar Modelo
Actor:	Médico Especialista (inicia)
Propósito:	Su propósito es rotar el modelo por los tres ejes de coordenadas.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de Rotar, brindándole la oportunidad de que lo realice por los tres ejes. Y termina después de haber rotado el modelo por cualquiera de los ejes seleccionados.
Referencia:	RF3.1, RF3.2, RF3.3
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. Selecciona la opción Rotar	1.1 Muestra los ejes de coordenadas por los que se puede rotar la malla. Rotar: -X. -Y. -Z.
2. Selecciona un eje de rotación.	2.1 Rota la malla por el eje seleccionado. Termina el caso de uso.
Post-condiciones:	Se va visualizando la rotación de la malla.
Prioridad:	Secundaria.

Tabla 4: Caso de Uso Rotar Modelo.

Caso de Uso:	Trasladar Modelo
Actor:	Médico Especialista (inicia)
Propósito:	Su propósito es trasladar el modelo.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción de Trasladar, brindándole la oportunidad de que lo realice por los tres ejes. Y termina cuando el modelo es trasladado por cualquiera de los ejes de coordenadas.
Referencia:	RF4.1, RF4.2, RF4.3
Flujo Normal de Eventos	
Acción del Actor	**Respuestadel Sistema**
1. Selecciona la opción Trasladar	1.1 Muestra los ejes de coordenadas por los que se puede trasladar la malla. Trasladar: -X.

	-Y. -Z.
2. Selecciona un eje de traslación.	2.1 Traslada la malla por el eje seleccionado. Termina el caso de uso.
Post-condiciones:	Se va visualizando la traslación de la malla.
Prioridad:	Secundaria.

Tabla 5: Caso de Uso Trasladar Modelo.

Caso de Uso:	Cortar Modelo
Actores:	Médico Especialista (inicia)
Propósito:	Su propósito es realizar el proceso de corte.
Resumen:	El caso de uso se inicia cuando el usuario desplaza el mouse sobre el modelo y comienza a ejecutarse el proceso de corte. Y termina después de que se realicen varios cortes.
Referencia:	RF5.1, RF5.2, RF5.3 RF5.4, RF5.5, RF5.6
Flujo Normal de Eventos	
Acción del Actor	**Respuestadel Sistema**
1. Desplaza el mouse sobre el modelo.	1.1 Realiza el corte.
Post-condiciones:	Se visualiza el modelo con el corte realizado.
Prioridad:	Crítica.

Tabla 6: Caso de Uso Cortar Modelo.

3.5. Diagrama de Clases.

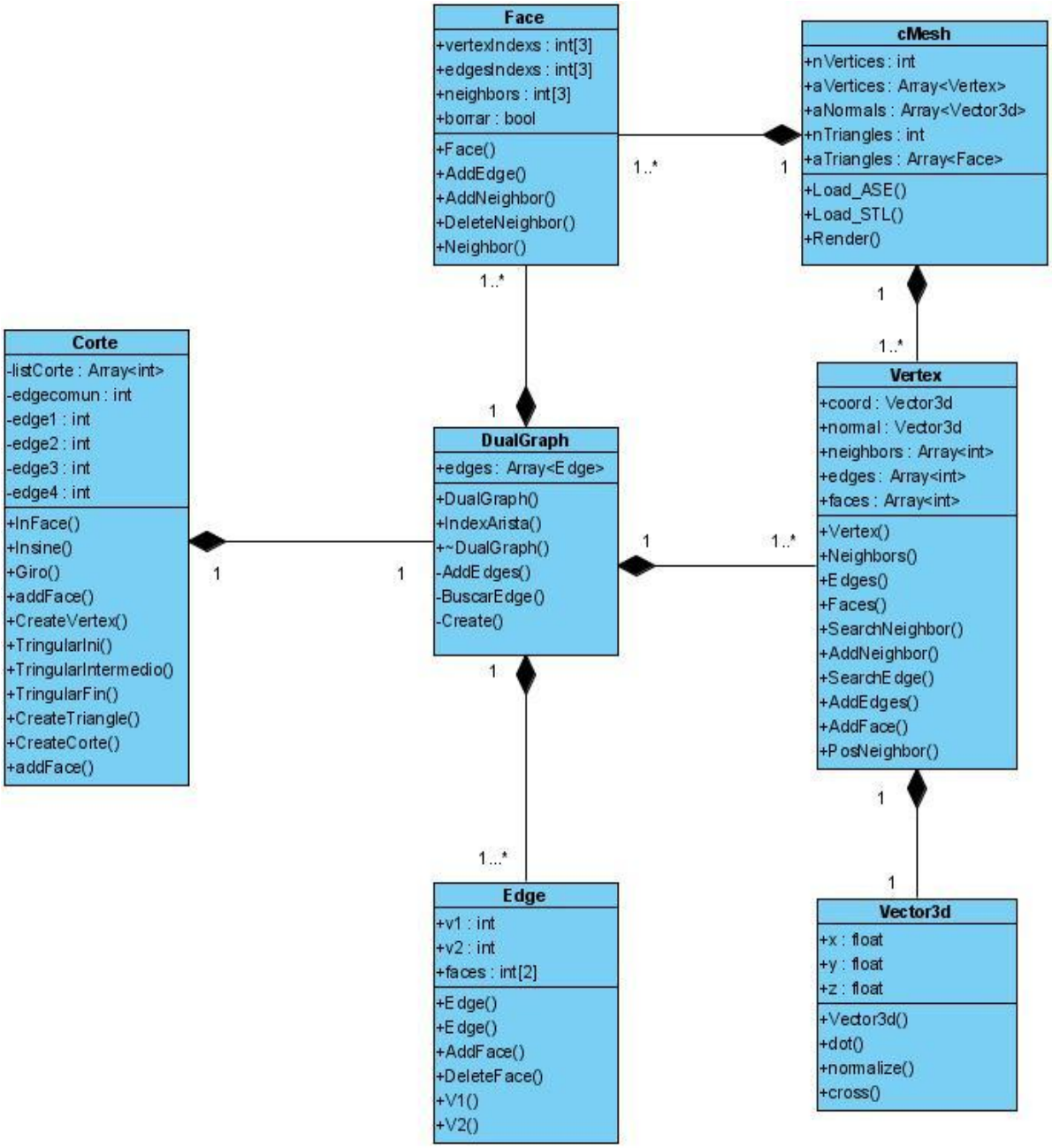

FIG 23: Diagrama de Clases.

3.6. Diagramas de Secuencia del Diseño.

A continuación se representarán los diagramas de secuencia del diseño, que representan el flujo de actividades que se realiza entre las clases del diseño. Esto posibilita comprender mejor el módulo elaborado en términos de implementación.

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

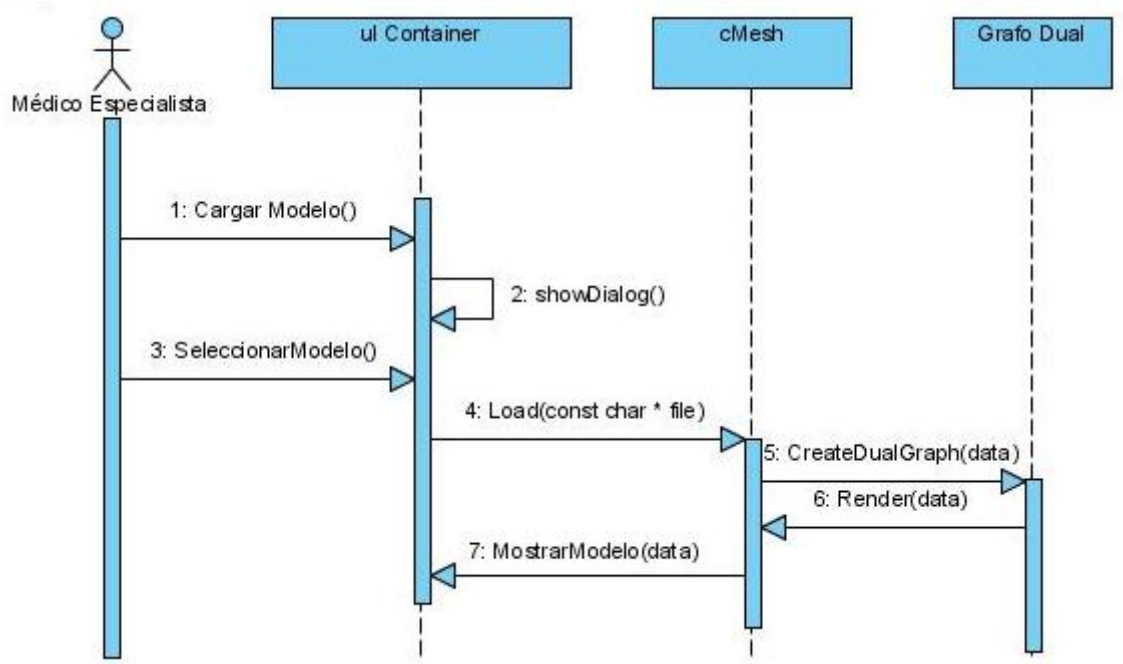

FIG 24: Diagrama de Secuencia de Cargar Modelo.

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

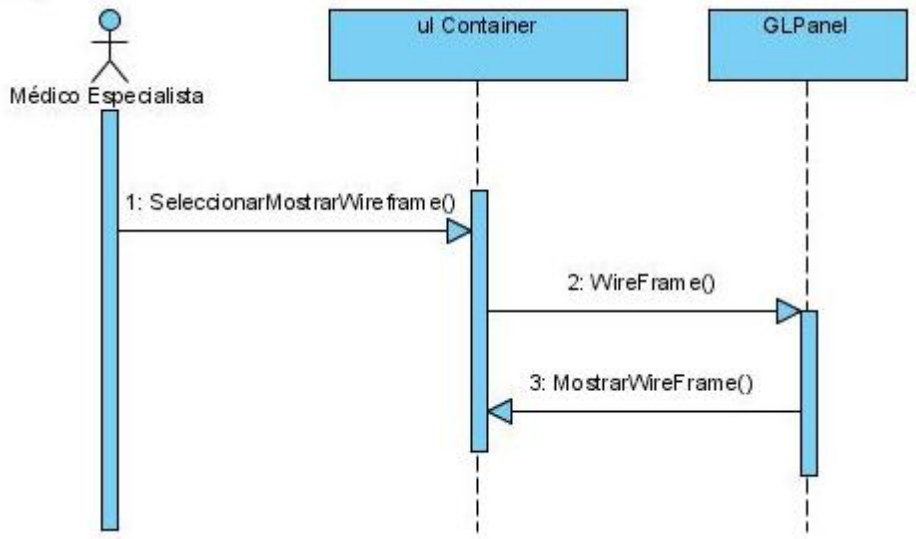

FIG 25: Diagrama de Secuencia de Mostrar Modo Wireframe.

3.6.3. Diagrama de Secuencia del Caso de Uso Rotar Modelo.

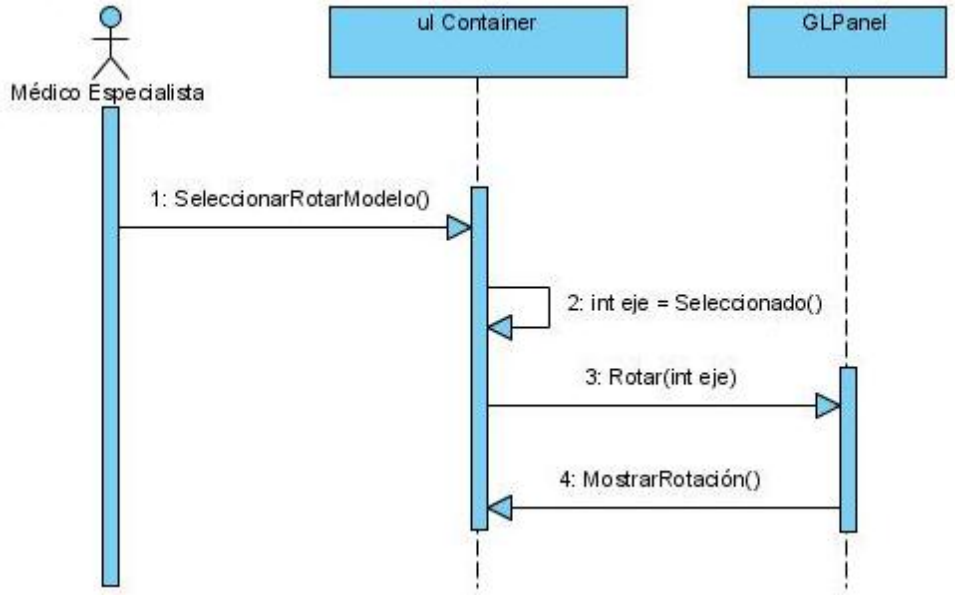

FIG 26: Diagrama de Secuencia de Rotar Modelo.

3.6.4. Diagrama de Secuencia del Caso de Uso Trasladar Modelo.

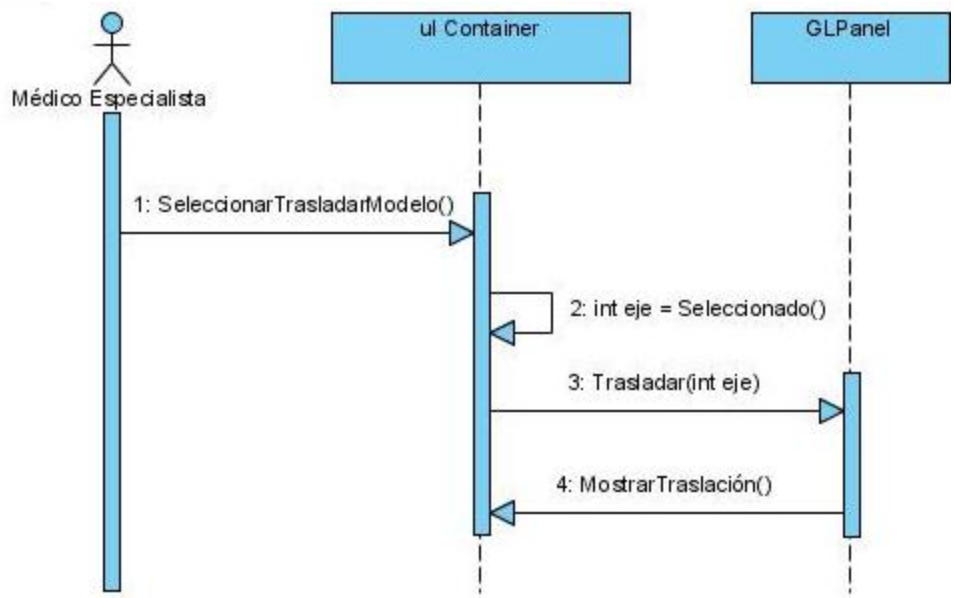

FIG 27: Diagrama de Secuencia de Trasladar Modelo.

3.6.5. Diagrama de Secuencia del Caso de Uso Cortar Modelo.

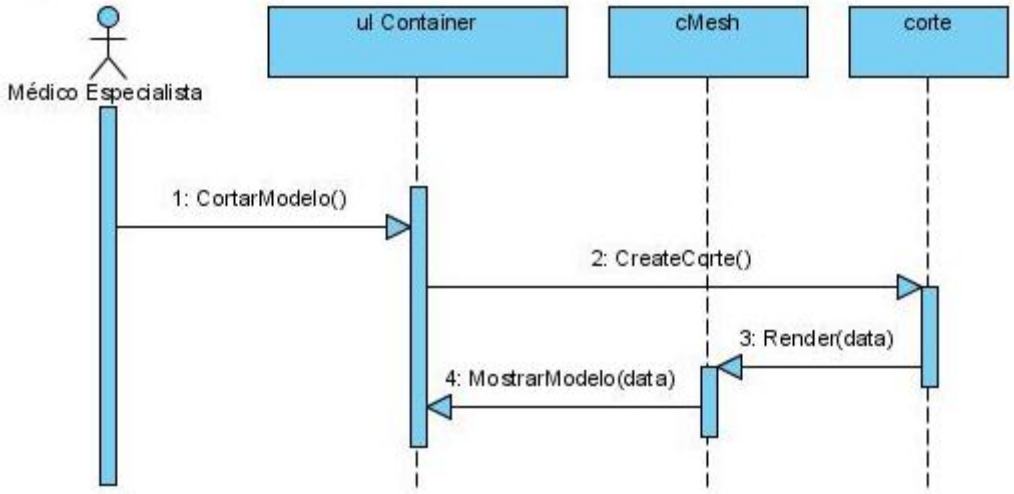

FIG 28: Diagrama de Secuencia de Cortar Modelo

CAPÍTULO 4: IMPLEMENTACIÓN Y VALIDACIÓN DE LOS RESULTADOS.

En este capítulo se abordan temas relacionados con la implementación y validación del módulo obtenido. Generándose el artefacto que se corresponden con el flujo de ingeniería por el que está transitando el módulo y que lo constituye el diagrama de componentes. Además se validan los resultados a través de las funcionalidades que se corresponden con los requerimientos capturados.

4.1. Implementación.

El principal resultado del proceso de implementación, es la obtención de componentes, dentro de los que se pueden incluir ficheros, ejecutables y las dependencias existentes entre estos. En este flujo se especifica cómo van a estar ubicadas físicamente las distintas partes del sistema.

4.2. Diagrama de componentes.

Un componente por sí solo representa una parte física del sistema, por ejemplo, una biblioteca de clases, un ejecutable, una tabla, entre otros., que engloba la implementación de las clases definidas en el diseño. Este diagrama le permite conocer a los desarrolladores y clientes la estructura física con la que cuenta el módulo y la relación entre sus partes. En la Fig. 28 se muestra el diagrama de componentes correspondiente al módulo elaborado.

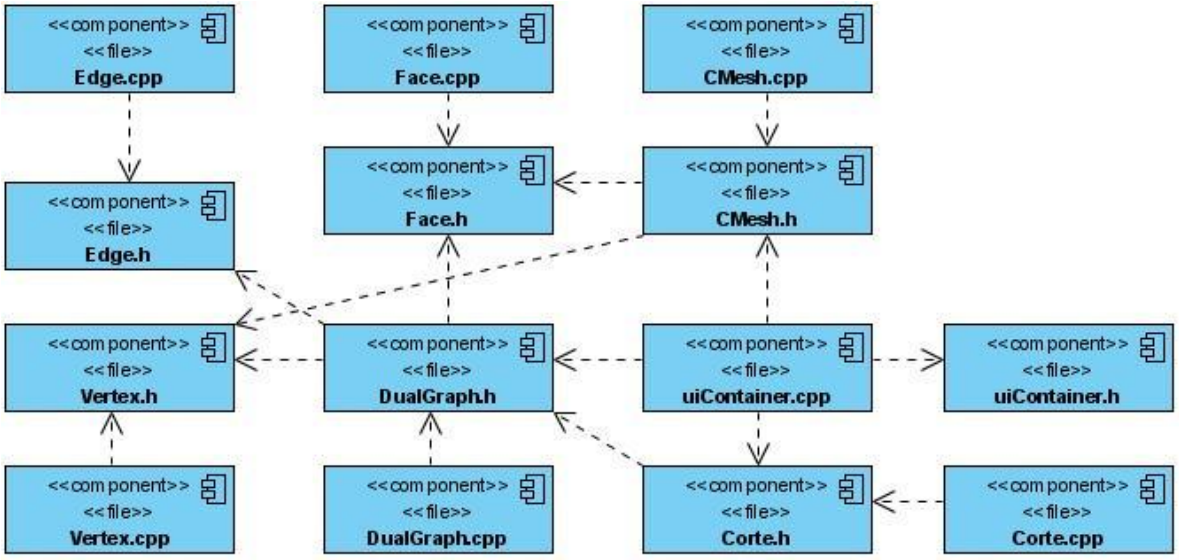

FIG 29: Diagrama de Componentes.

4.3 Validación.

Para la validación del módulo que se obtuvo se realizaron comparaciones funcionales, que consisten en probar el módulo contra los requerimientos capturados. Todas las comparaciones se realizaron sobre una computadora con un procesador Intel Core2 Quad Q6600 a una frecuencia de 2.4 GHz, 1 GB de memoria RAM y una tarjeta gráfica NVidia 9800 GT con 512 MB de RAM para video.

Para realizar las comparaciones funcionales se siguió una estructura que fue definida por el autor. La que se menciona a continuación:

Entrada:

Descripción:

Resultado esperado:

Prototipo no funcional:(interfaz que se corresponde con la imagen definida en la aplicación).

Resultado obtenido:

Las validaciones se realizan en orden lógico, que se corresponde con la secuencia de pasos que debe ejecutar el usuario final para la utilización del módulo de corte.

4.3.1 Cargar Modelo.

Entrada:

El usuario selecciona la opción cargar modelo, brindándose la posibilidad de cargar modelos en el formato *.stl y *.ase.

Descripción:

Se inicia cuando el usuario selecciona la opción de cargar el modelo, luego sale una ventana para seleccionar el modelo en los formatos correspondientes a la entrada y termina cuando se visualiza el modelo seleccionado.

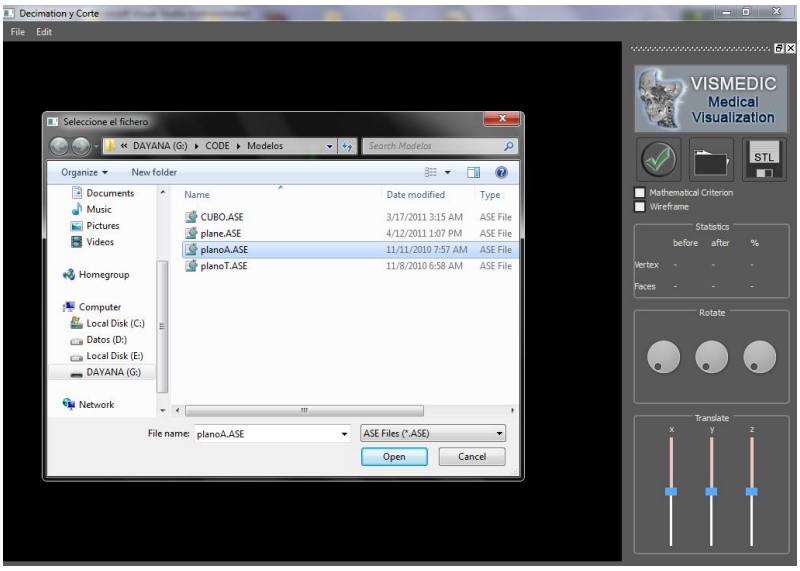

FIG 30: Ejemplo de cargar modelo.

Resultado esperado:

Se carga el modelo seleccionado.

Resultado obtenido: Se corresponde con el resultado esperado. Por lo que la comparación es válida cumpliéndose así con el rF especificado.

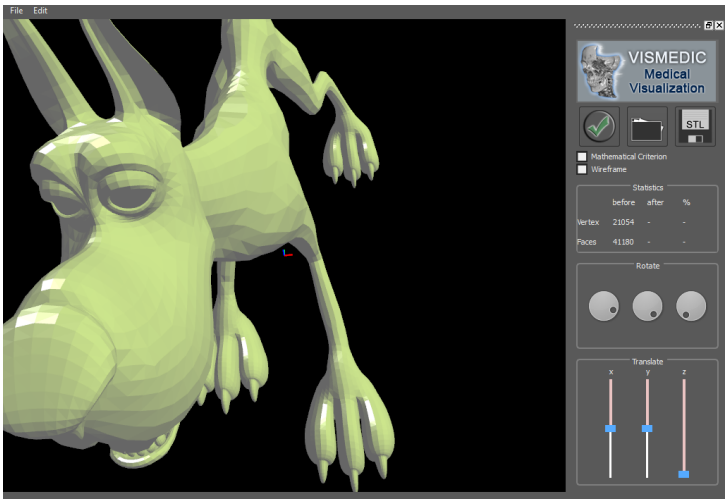

FIG 31: Modelo cargado.

4.3.2 Mostrar Modo Wireframe.

Entrada:

El usuario selecciona la opción WireFrame.

Descripción:

Se inicia cuando el usuario selecciona la opción de Wireframe y termina cuando se muestra el mallado del modelo.

Resultado esperado:

El modelo se muestra en Modo Wireframe o sea la triangulación del modelo.

Resultado obtenido: Se corresponde con el resultado esperado. Por lo que la comparación es válida cumpliéndose así con el rF especificado.

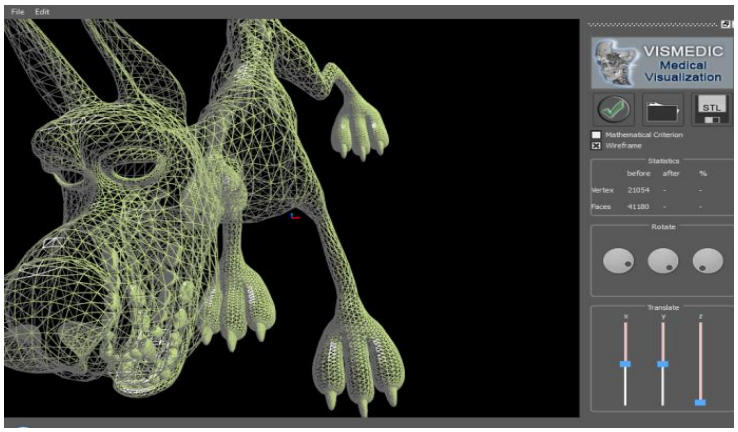

FIG 32: Modelo en modo Wireframe.

4.3.3 Cortar Modelo.

Entrada:

El usuario desplaza el mouse sobre el modelo para realizar el corte.

Descripción:

Se inicia cuando el usuario desplaza el mouse sobre el modelo y comienza a ejecutarse el proceso de corte. Y termina después de que se realicen varios cortes.

Resultado esperado:

Se realiza el corte.

Resultado obtenido: Se corresponde con el resultado esperado. Por lo que la comparación es válida cumpliéndose así con el rF especificado.

FIG 33: Modelo después de haberse realizado 3 cortes.

4.3.4 Rotar Modelo.

Entrada:

El usuario selecciona las opciones de rotar modelo por cualquiera de los ejes de coordenadas.

Descripción:

Se inicia cuando el usuario selecciona la opción de Rotar, brindándole la oportunidad de que lo realice por los tres ejes. Y termina después de haber rotado el modelo por cualquiera de los ejes.

Resultado esperado:

Se rota el modelo por el eje X, Y o Z según la selección realizada.

Resultado obtenido: Se corresponde con el resultado esperado. Por lo que la comparación es válida cumpliéndose así con el rF especificado.

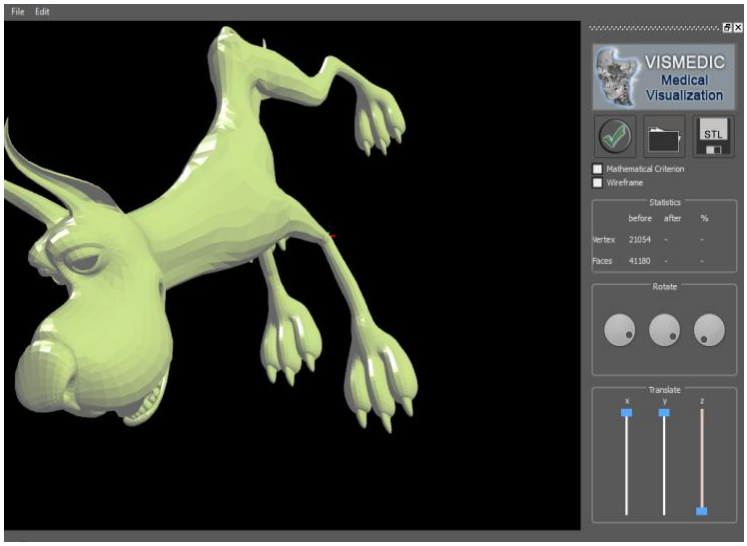

FIG 34: Modelo después de realizar varias rotaciones.

4.3.5 Trasladar Modelo.

Entrada:

El usuario selecciona las opciones de trasladar modelo.

Descripción:

Se inicia cuando el usuario selecciona la opción de Trasladar, brindándole la oportunidad de que lo realice por los tres ejes. Y termina después de haber trasladado el modelo por cualquiera de los ejes.

Resultado esperado:

Se traslada el modelo por el eje X, Y o Z según la selección anterior.

Resultado obtenido: Se corresponde con el resultado esperado. Por lo que la comparación es válida cumpliéndose así con el rF especificado.

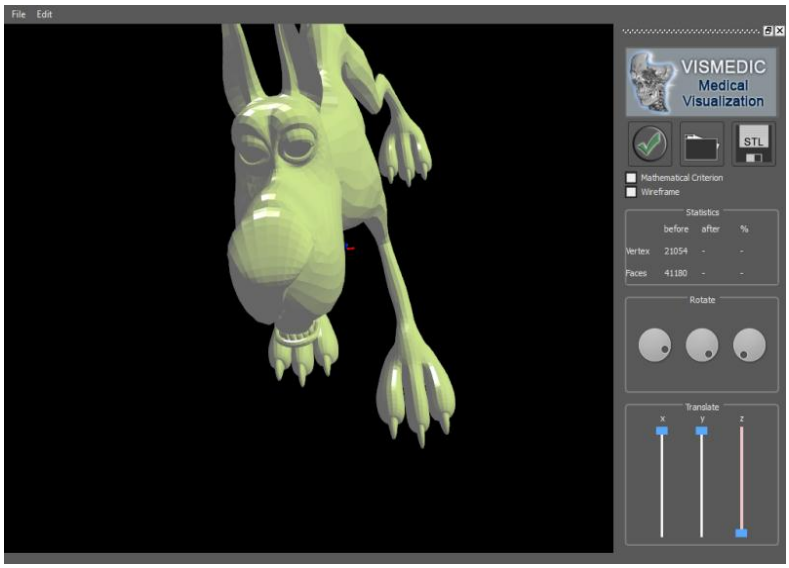

FIG 35: Modelo después de realizar varias traslaciones.

De las cinco comparaciones realizadas se obtuvo que las cinco se corresponden con el resultado que se quería obtener por lo que se pudo validar el módulo propuesto.

CONCLUSIONES

Con la realización de la investigación se logró desarrollar un módulo de corte sobre mallas superficiales triangulares. El trabajo desarrollado cumple con las especificaciones requeridas y le da cumplimiento a las tareas de la investigación.

1. La solución propuesta para la realización del corte brinda ejecución en tiempo real al ser el mismo realizado de forma dinámica.
2. Se logró representar el interior de los cuerpos con la generación de la ranura interna para cada triángulo de la malla.
3. Se integró el módulo de corte con el módulo de decimación del proyecto Visualización Médica.

RECOMENDACIONES

En el presente trabajo se alcanzaron los objetivos planteados pero este está expuesto a varias recomendaciones con el objetivo de perfeccionar el mismo:

1. Integrar el módulo de corte realizado con un modelo físico que permita un mayor realismo en la visualización de la malla.
2. Incorporar al módulo actual un objeto de corte más complejo y manejar las colisiones entre este objeto virtual de corte y la malla.
3. Tratar los casos degenerados como son cuando el instrumento de corte se encuentre sobre un punto de la malla.

BIBLIOGRAFÍA

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

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

3. Aulignac, D. 2001. *Modelisation de linteractionavec des objets deformables en temps-réelpour dessimulateurs medicaux.* s.l. : PhD thesis, Institute NationalePolytechnique de Grenoble, France, 2001.

4. Waters, K. 1987. *A muscle model for animating three dimensional facial expression.* s.l. : In Proceedings of ACM SIGGRAPH, 1987.

5. Terzopoulos D, Waters K. 1990. *Physically-Based Facial Modeling, Analysis, and Animation.* s.l. : Journal of Visualization and Computer Animation, 1(2):73–80, 1990.

6. Turner, M. J., Clough, R. W., Martin, H. C., Topp, L. J. 1956. *Stiffness and deflection analysis of complex structures.* s.l. : J. Aero, 1956.

7. C., Mendoza Serrano. 2003. *Soft Tissue Interactive Simulations for Medical Applications Including 3D Cutting and Force Feedback.* s.l. : Instituto Nacional Politécnico de Grenoble, Francia.

8. Bro-Nielsen M., Cotin S. 1996. *Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation.* s.l. : Technical University of Denmark, Lyngby, Denmark, 1996.

9. Al-khalifah A., Roberts D. 2004. *Survey of modeling approaches for medical simulators.* s.l. : Centre for Virtual Environments, the University of Salford, Manchester, UK, 2004.

10. Doug L. J., Dinesh K. P. 1999. *Accurate Real Time Deformable Objects.* s.l. : University of British Columbia,Canada, 1999.

11. Koppel D., Wang Y., Chandrasekaran Sh. 2003. *Toward Real-Time, Physically-Correct Soft Tissue Behavior Simulation.* s.l. : University of California, USA, 2003.

12. C. Forest, H. Delingette N. Ayache. 2002. *Cutting simulation of manifold volumetric mehses.* s.l. : Proceedings Medical Image Computing and Computer Assisted Intervention (MICCAI), 2002.

13. Cotin, S. 1997. *Modéles anatomiques déformables en temps-réel.* s.l. : Thése de doctorat, INRIA Sophia Antipolis – Université de Nice, Sophia Antipolis,cited on pages 9, 10, 12, 23, 37, 46,50, 50, 54, 54, 55, 55, 68, 68, 69, 73, 172, 1997.

14. Cotin, H. Delingette, and N. Ayache. 2000. *A hybrid elastic Model allowing real-time cutting, deformations and force-feedback for surgery training and simulation.* s.l. : The Visual Computer, 16(8):437 452, 2000.

15 Boux de Casson. 2000. *Simulation Dynamique de Corps Biologiqueset.* s.l. : PhD thesis, Universite de Savoie, France, 2000.

16. Han-Wen Nienhuys, A. Frank Van Der Stappen. 2001. *A surgery simulation supporting cuts and finite element deformation.* s.l. : MICCAI, 2001.

17. Faure, C. Mendoza - C. Laugier - O. Galizzi - F. 2003. *Simulating Cutting in Surgery Applications using Haptics and Finite Element Models.* 2003.

18. Seifert, A. Mazura and S. 1997. *Virtual cutting in medical data.* s.l. : In Proc. of Medicine Meets Virtual Reality,pages 420 429, 1997.

19. D. Bielser, V. A. Maiwald, and M. H. Gross. 1999. *Interactive cuts through 3-dimentional soft tissue.* s.l. : In EUROGRAPHICS'99, volume 18, pages C31–C38., 1999.

20. Kanade, A. Mor and T. 2000. *Progressive cutting with minimal new element creation.* s.l. : In MICCAI Proc, pages 598, 607, 2000.

21. Bielser, D. 2003. *A state machine for real-time cutting of tetrahedral meshes.* s.l. : In Proc. of Pacific Graphics,pages 377, 386, 2003.

22. Hui Zhang, ShahramPayandeh y John Dill. 2008. *Simulation of Progressive Cutting on Surface Mesh Model.* 2008.

23 Yaima Soto Avello. 2009. *Técnica de Corte para Mallas Superficiales*.

24 Lester Oscar Rodriguez González, Leonardo Rafael Fernandez Ruiz. 2008. *Técnica de Corte para Mallas Superficiales*.

ANEXOS

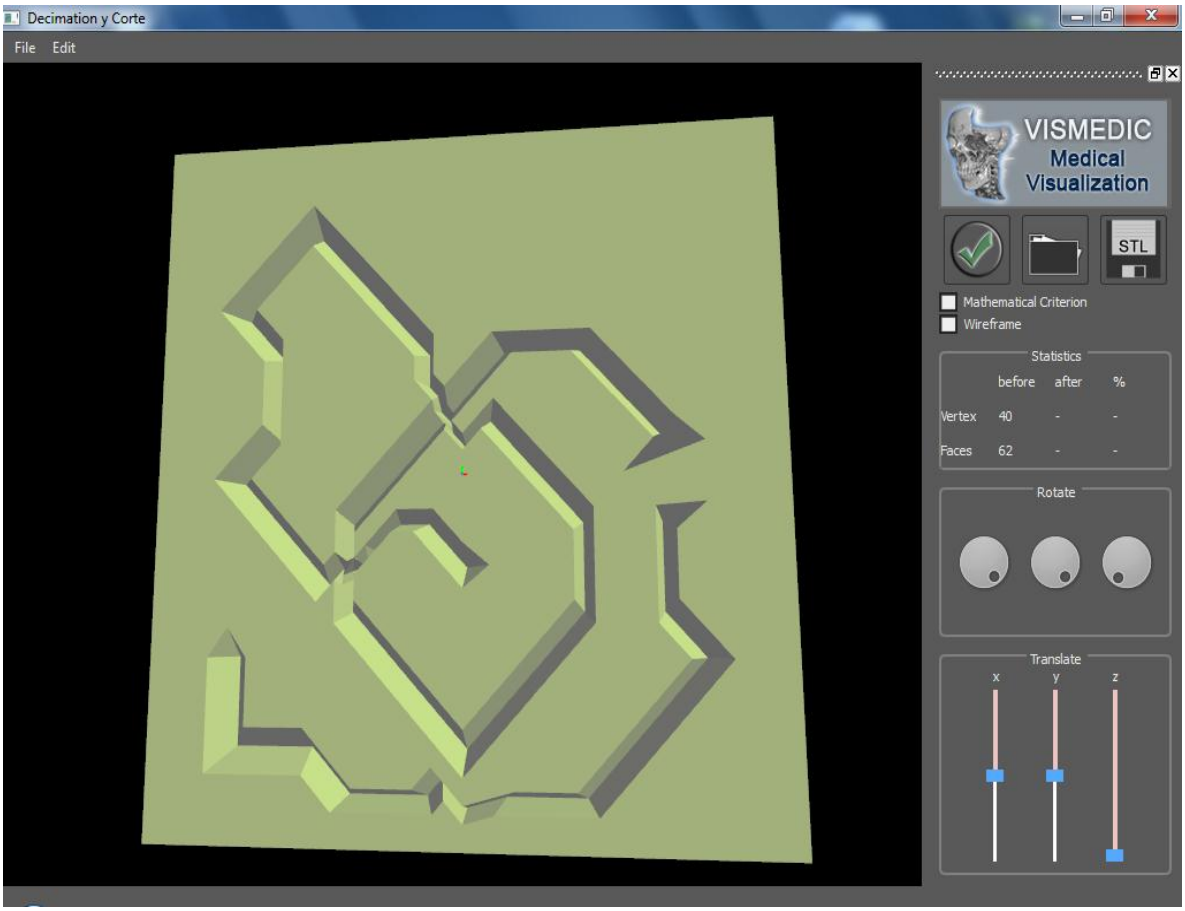

FIG 36: Cortes sobre un modelo en dos dimensiones.

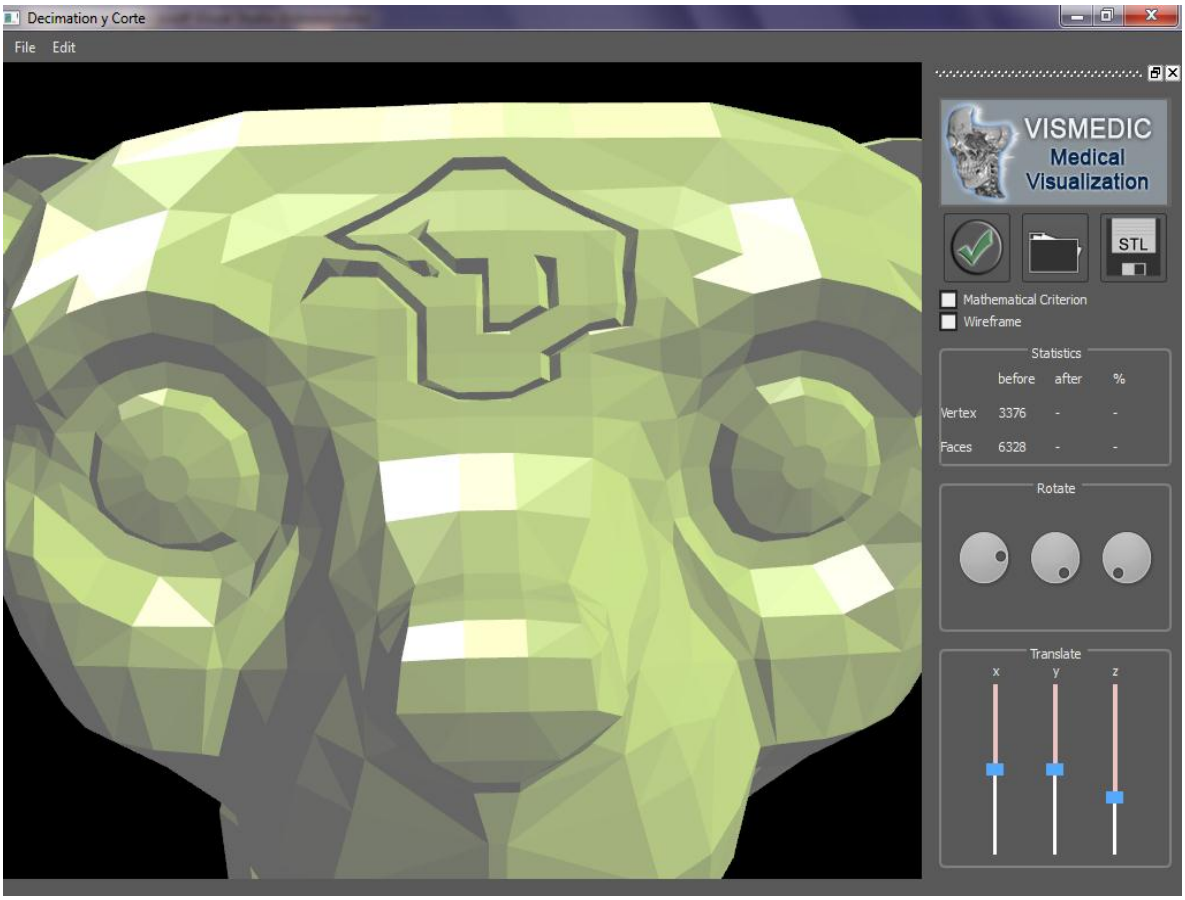

FIG 37: Corte sobre la cabeza de un leopardo.

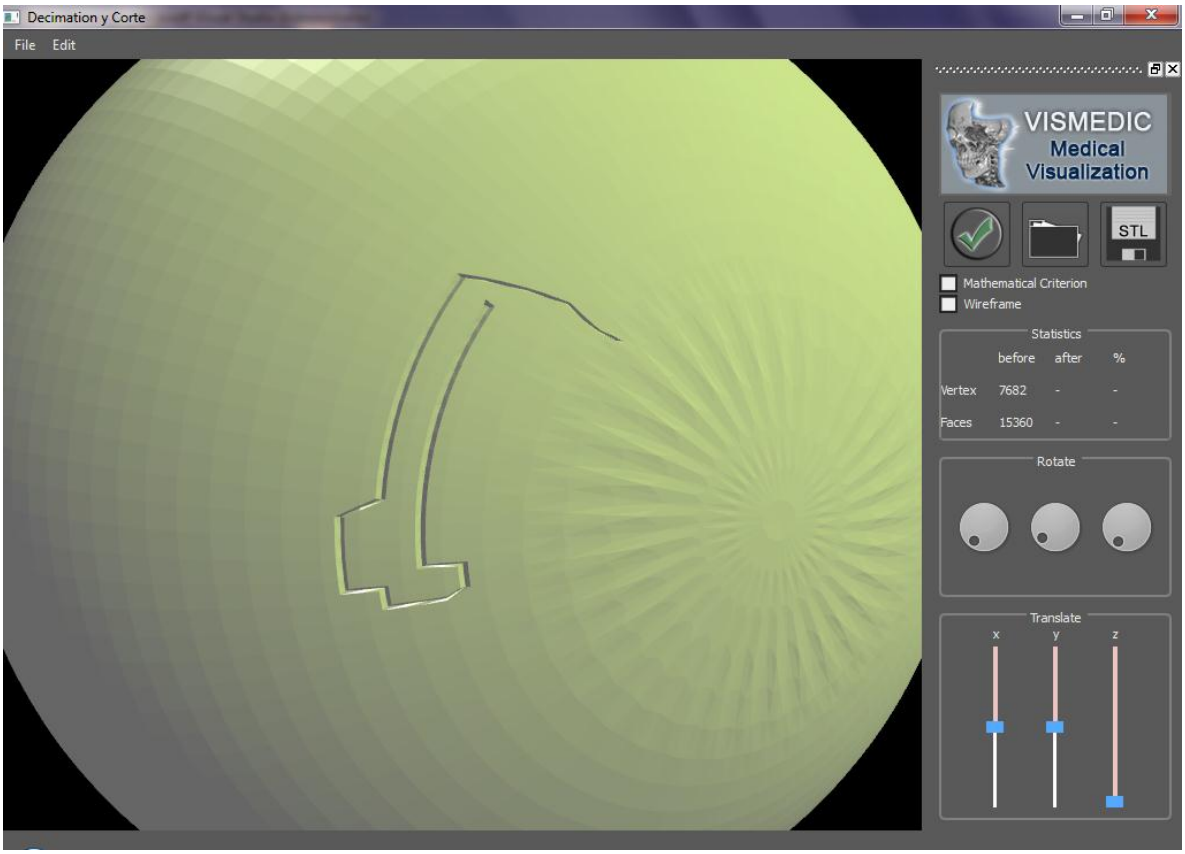

FIG 38: Corte sobre una esfera.

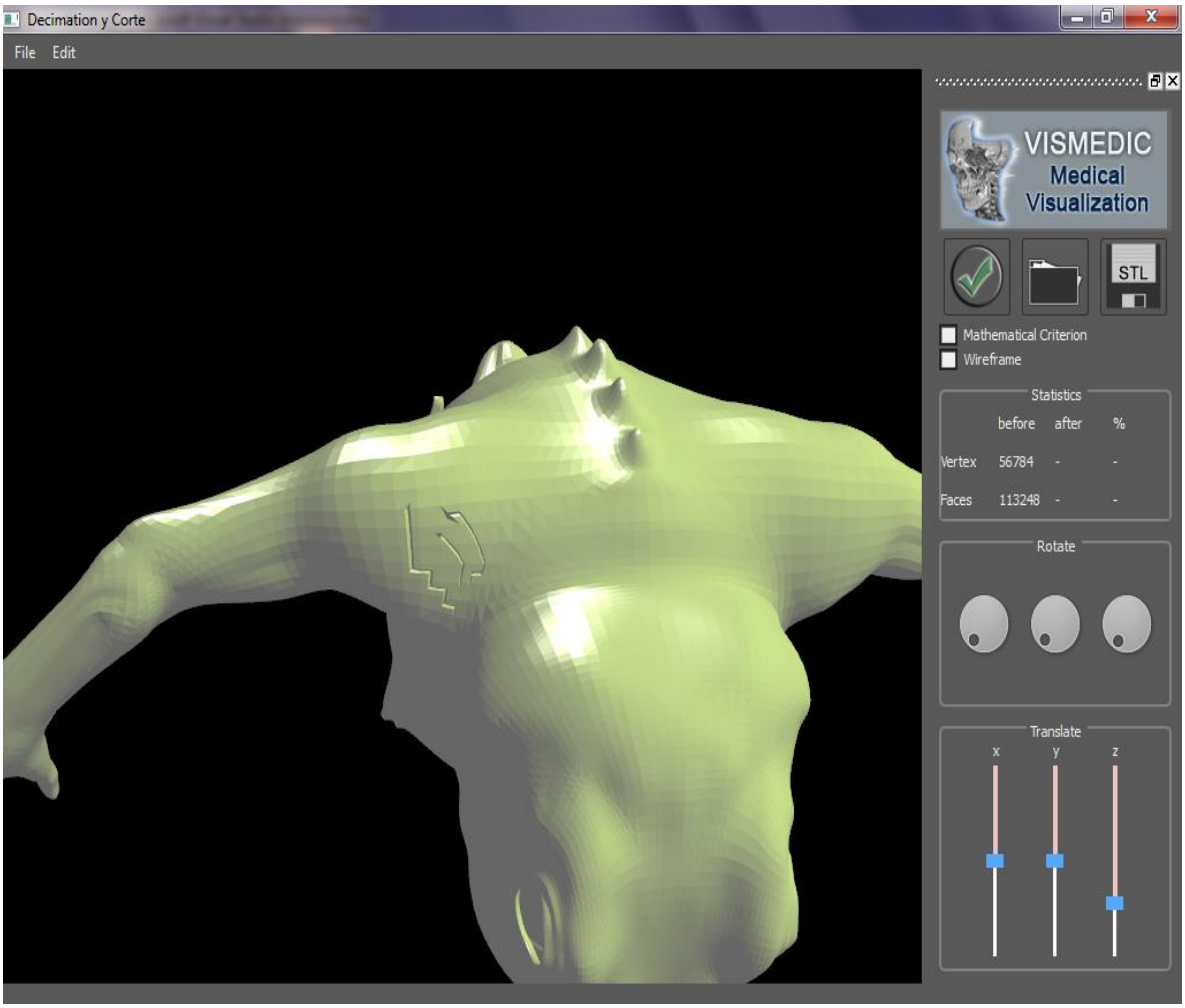

FIG 39: Corte sobre una rana.

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.

Arista: Segmento compuesto por 2 vértices.

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.

N

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

O

OpenGL: Biblioteca 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

Polígonos: Conjunto de vértices y aristas.

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

Render: Proceso de obtención de imágenes por computadora.

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 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

Tarjeta gráfica: Es una tarjeta de circuito impreso encargada de transformar las señales eléctricas que llegan desde el microprocesador en información comprensible y representable por la pantalla del ordenador.

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

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

V

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.

W

Wireframe: Algoritmo de renderización del que resulta una imagen semitransparente, de la cual sólo se dibujan las aristas de la malla que constituye al objeto