UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Técnica de Corte para Mallas Superficiales

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores: Lester Oscar Rodríguez González

Leonardo Rafael Fernández Ruiz

Tutores: Ing. Osley Bretau Camejo

Ing. Raissel Ramírez Orozco

Dedicatoria de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al Proyecto Simulador Quirúrgico (Kheipros) de la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autores:

Lester O. Rodríguez González

_____

Leonardo R. Fernández Ruiz

_____

Tutores:

Ing. Osley Bretau Camejo

_____

Ing. Raissel Ramírez Orozco

_____

Datos de Contacto:

Nombre y Apellidos: Osley Bretau Camejo

Edad: 25 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero Informático

Categoría Docente: Profesor Adiestrado

E-mail: obretau@uci.cu

Graduado en la Universidad de las Ciencias Informáticas, 3 años de experiencia en proyectos de Realidad Virtual.

Nombre y Apellidos: Raissel Ramírez Orozco

Edad: 25 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero Informático

Categoría Docente: Profesor Adiestrado

E-mail: rramirez@uci.cu

Graduado en la Universidad de las Ciencias Informáticas, 3 años de experiencia en proyectos de Realidad Virtual.

Dedicatoria

...a mis padres Leonela y Rafael.

...a Josué, mi semillita que va creciendo.

.

Leo.

... a mami Lupe

… a mis padres Maritza y Oscar.

… a mi tía Grisel y mi tío Frey

Lester O.

Agradecimientos

Con este trabajo concluye un largo viaje de sacrificio y dedicación para nosotros, gracias a Dios. Esto no hubiera sido posible sin la colaboración de amigos que estuvieron presentes en las buenas y en las malas, unos partimos juntos, otros se incorporaron con el tiempo pero logramos fusionarnos como un buen equipo que supo sortear las dificultades. Este trabajo fue gracias a la colaboración de una lista infinita de personas, solo queda ofrecerles nuestros más sinceros agradecimientos a todos los que hicieron posible este sueño.

A nuestros tutores Osley y Raissel, por sus palabras y sabios consejos, nuestro trofeo es también suyo.

A Juan Manuel, por su preocupación desde el primer día, por la confianza que depositó en nosotros, lo recordaremos siempre.

A Dania, Yoisy, Lida por las impecables revisiones, consultas y consejos.

Al piquete: Yausell, Guille, Leonel "El helper", Inti, el Niche, Saylin por sus imprescindibles aportes.

A todo el grupo de estudio, a los socios del cuarto Ricar, Rainer, Candiman, el Flaco, Yaself, Pupi, Yera, Fácil, Marlon a todos los que estuvieron al tanto de nuestro trabajo.

A todos muchas gracias.

A cinco personas en especial porque su presencia ha sido y será siempre el motivo más grande que me impulse a vencer cualquier meta, sabiendo aún que jamás existirá una forma de agradecerles en esta vida lo que han hecho por mí, deseando que sientan este logro también de ustedes. Por su sacrificio conmigo, por su comprensión y confianza, por su amor y amistad incondicional, por iluminar mí camino, en fin por existir…….. Un millón de gracias. Los amo y amaré por la eternidad:

Mami Lupe, Papa, Mama, Tía Grisel y mi Tío Frey.

A Yali con todo mi corazón, has significado mucho en mi vida, gracias por tu cariño y ternura, por tenerte siempre al lado, por ser siempre mi compañera: en los momentos buenos y malos.

A Araceli que te quiero mi amor como una tía pues así has sido conmigo en todo momento.

A mi tía Eulalia, mi tío Omar, mi Abuela Rosa que siempre se han preocupado por mis resultados en la escuela.

A toda mi familia querida en general…. mi hermanito, mis primos, mis tías, Miguelito todos… todos son especiales.

A mi compañero de tesis Leo que ha sido más que un amigo y un compañero de tesis, un hermano.

A los compañeros de la vocacional Dairillo, Aenlys, Axel que siempre han estado para lo que los necesite.

A todas las personas queridas que siempre estuvieron al tanto de mi desempeño en la universidad, por sus consejos, sus maneras de darme ánimos, sus ganas de verme triunfar.

A todos muchas gracias.

Lester Oscar

A mis padres, por la educación, el apoyo y los valores que sembraron en mi personalidad, quienes me han heredado el tesoro más valioso que puede dársele a un hijo: amor.

A mi hermana Ibelis, por su preocupación incansable.

A toda mi familia quienes sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y educarme. A toda la gente que me vio crecer, a quienes la ilusión de su vida ha sido convertirme en una persona de provecho.

A Lili y el familión completo que me abrió sus brazos como un hijo más.

A mi primo Joe por ayudarme siempre que lo necesité desde el primer año de la carrera.

A Elizabeth por regalarme todo su amor y cariño.

A Endris, Errol, Jhonatan, Alexander amigos que nunca faltan cuando se les necesita.

A mi compañero de tesis Lester, cariñosamente Marko, has sido un amigo y un hermano.

A mis amigos del barrio, la vocacional, de la Uci, los que serán siempre.

A todos los que faltan gracias de todo corazón

Leonardo.

Resumen

La visualización de la información mediante computadoras ha revolucionado muchos campos de la ciencia. La simulación quirúrgica se alza como una prominente tecnología para el entrenamiento de estudiantes de medicina y la planificación de cirugías. Un requerimiento importante para tal sistema de simulación es ser capaz de efectuar cortes sobre tejidos, de forma tal que se pueda entrenar esta importante habilidad. Este trabajo propone una técnica de corte para modelos superficiales; se definen un conjunto de algoritmos para la subdivisión en mallas con el paso de una herramienta de corte.

En desarrollo de la investigación se ha realizado un estudio del estado del arte a nivel mundial de las principales técnicas usadas con este propósito, las cuales se dividen en tres tendencias fundamentales: destructivo, subdivisión y separación.

El módulo resultante de este proyecto, implementa una técnica de corte aplicable para mallas superficiales con el objetivo de brindar esta funcionalidad al proyecto Simulador Quirúrgico de la Facultad 5 en la Universidad en las Ciencias Informáticas. El proceso de desarrollo fue dirigido por el Proceso Unificado del Software (RUP) e implementado en C++ estándar.

Palabras claves

Simulación quirúrgica, técnica de corte, subdivisión, mallas superficiales.

Tabla de Contenidos

Introducción

El constante cambio en el mundo tecnológico y la creciente necesidad de nuevas tecnologías que emergen estocásticamente, dan como resultado el crecimiento en el desempeño de las supercomputadoras y en particular las computadoras con grandes alcances en capacidades gráficas, lo que ha permitido la incorporación de nuevas tecnologías de visualización y modelación como la Realidad Virtual.

Precisamente la medicina a lo largo de los últimos decenios se ha manifestado a favor de utilizar las tecnologías más avanzadas. Tal es el caso del aprendizaje y entrenamiento de los médicos con el fin de mejorar el progreso y la calidad de la salud. Como puede resultar obvio, un cirujano no puede entrar por primera vez a un quirófano con un paciente grave sin haber tenido anteriores experiencias operando a personas. Por ese motivo, la tradición médica ha venido utilizando diversos tipos de “entrenadores”, basados en reproducciones de zonas del cuerpo, para poder aportar una base realista al aprendiz. En nuestros días irrumpe otra vía para el entrenamiento, basada en simuladores virtuales, también llamados simuladores quirúrgicos a los que se les programan modelos de órganos y zonas del cuerpo humano para realizar las prácticas de los disímiles procedimientos quirúrgicos.

El corte es una manipulación o proceso común que se encuentra a diario en la cirugía, tanta importancia reviste que en simuladores quirúrgicos es un ejercicio de obligatoria inclusión. A menudo se escucha que entre las habilidades y destrezas que un cirujano debiera tener, figuran una serie de maniobras básicas, donde cortar destaca como una de las más importantes. Estas habilidades son difíciles de obtener en la práctica, ya que debe considerarse la seguridad del paciente por encima de las necesidades de enseñanza “las técnicas de realidad virtual aparecen a los ojos de muchos expertos como el medio definitivo de entrada de la informática en los procesos de formación y entrenamiento” [1].

Nuestro país está inmerso en un programa revolucionario de informatización, se perfila para incursionar en el tema de la animación por computadoras a un nivel que le permita ocupar un puesto en el creciente mercado internacional. Ya se han dado los primeros pasos, existe el “Centro de Investigación y Desarrollo de Simuladores” (SIMPRO), pionero en el desarrollo de este campo en nuestro país, ha surgido también como perfil investigativo y de producción de la Facultad 5, en la Universidad de la Ciencias Informática (UCI), donde ya se perciben logros en esta rama tan amplia de

la informática con el desarrollo de una biblioteca de clases llamadas Scene Tool Kit (STK) la cual facilita la creación y manipulación de entornos virtuales.

La Facultad 5 de la UCI cuenta con un grupo de proyectos tanto investigativos como de producción, enfrascados en el desarrollo de Sistemas de Realidad Virtual (SRV). Unos de estos proyectos está desarrollando un Simulador Quirúrgico (Kheipros), implementado con el uso de STK, el cual carece de un módulo que permita realizar modificaciones en la malla de un objeto tridimensional, convirtiéndose en una **necesidad** importante del proyecto, cuando necesita configurar ejercicios de entrenamiento para las habilidades de corte en tejidos.

Analizando la situación problémica existente el **problema** a resolver es: ¿Cómo crear una técnica de corte en mallas superficiales para que sea usada por el Simulador Quirúrgico en los ejercicios para el entrenamiento de cirujanos?

Para adentrarse en la ciencia que estudia esta investigación, aspecto importante para definir la línea de trabajo se define el **objeto de estudio** el comportamiento matemático de las mallas al someterlas al proceso de corte y el **campo de acción** son las técnicas y algoritmos utilizados para la simulación de este tipo de proceso y su eficiencia.

El **objetivo general** del trabajo es proveer de una técnica, o sea un conjunto de algoritmos que ofrezcan la posibilidad de subdividir las mallas, para así simular el proceso de corte. Esto aporta una de las funcionalidades más importantes en los simuladores quirúrgicos.

Para satisfacer las necesidades planteadas es necesario ejecutar las tareas de investigación que se relacionan a continuación:

Analizar y clasificar la bibliografía sobre el tema en los medios de información disponibles.

Estudiar las formas de representación geométrica de los objetos tridimensionales.

Clasificar los algoritmos para la detección de colisiones.

Estudiar y aplicar la metodología de desarrollo de software RUP.

Desarrollar un módulo para solucionar el problema planteado.

Para una mejor comprensión, este documento está dividido en capítulos que estructuran el contenido de la siguiente forma:

En el capítulo uno “Fundamentación Teórica” se hace un análisis del estado del arte de las soluciones de este tipo en el mundo, detallando las ventajas y desventajas con el objetivo de obtener cual es la posible solución al problema.

El capítulo dos “Soluciones Técnicas”, establece la solución resultante del análisis realizado en el capítulo anterior y se propone el algoritmo de corte a utilizar.

En el capítulo tres “Descripción de la Solución Propuesta”, se describe el sistema a desarrollar desde la perspectiva de las necesidades del cliente.

El capítulo cuatro “Diseño e Implementación del Sistema”, corresponde a los flujos de trabajo de diseño e implementación de RUP, donde se describen las clases de diseño y como se distribuyen en componentes de software.

Capítulo 1

Fundamentación Teórica

Introducción:

Para la realización del estudio concerniente al problema planteado, se realizó una revisión de la documentación existente acerca de un tema tan amplio y hasta hoy en estudio por muchos científicos de todo el mundo. En la investigación indudablemente están relacionadas varias ciencias como la Biomecánica, Matemáticas y la Informática Gráfica.

A continuación se expone el marco teórico conceptual en el cual está inmerso la investigación, cuenta con secciones donde se exponen; representación geométrica de la malla, métodos de solución para la deformación de las mallas y las tendencias actuales de los algoritmos de corte.

Teniendo en cuenta lo anterior existen actualmente un número de técnicas que permiten la simulación del corte en mallas superficiales y volumétricas. El número de pasos intermedios para obtener un procedimiento de corte limita con frecuencia el nivel de realismo que se obtiene al aplicar un método en específico, en eso radica la aproximación a lo que puede ser una acertada simulación del corte.

1.1 Visión General

Entre las principales acciones de los médicos cirujanos están cortar y suturar, definitivamente ellos han acumulado en sus manos un conjunto de habilidades que les permiten ejecutar procedimientos que a cualquiera se les pondrían los pelos de punta. El corte es vital tanto en la cirugía a cuerpo abierto como en la de mínimo acceso y es usado en diferentes partes del cuerpo, por ejemplo en el cerebro, la vesícula o los procedimientos ortopédicos. Es importante saber cuando y como cortar porque es un proceso irreversible. En el mundo se alzan hoy nuevas técnicas en el campo de la medicina y nuestro país no está al margen de tal desarrollo, por eso es notable el estudio en el arte de la cirugía abdominal, con el desarrollo alcanzado en la cirugía laparoscópica. El procedimiento comienza con tres pequeñas incisiones, el abdomen se infla con un gas que permita un espacio más abierto en el interior del paciente y una cámara de video es introducida por una de las incisiones, la imagen de video es transmitida a un video monitor de alta resolución permitiendo al cirujano ver la anatomía del paciente con gran claridad.

Hay que esclarecer que, si la cirugía mínimamente invasiva reduce el tiempo de recuperación del paciente, también es significativa la reducción de la zona de acceso al cuerpo del paciente, por lo que el cirujano debe ser altamente competente en este tipo de técnica quirúrgica. Precisamente, la falta de percepción del espacio y la necesidad de una nueva y específica coordinación mano-ojo, son los principales obstáculos que un cirujano debe superar. Motivados por lo novedoso de este tema y por la amplia gama de conocimientos que encierra el mismo, los investigadores del mundo han mostrado gran interés en desarrollar software para simulaciones quirúrgicas, con el objetivo de proveer de un sistema capaz de entrenar a los cirujanos.

Son varios los problemas en el desarrollo de simuladores quirúrgicos planteados por Ayache [14], en primer lugar, debe definirse los modelos geométricos y físicos de las estructuras anatómicas. Las geometrías pueden ser obtenidas de las diversas modalidades de imágenes médicas, en tanto que la naturaleza deformable de los tejidos se mide a través de estudios biomecánicos, además todo cálculo debe ser realizado con la suficiente rapidez para mantener interacciones en tiempo real. En el área de la simulación quirúrgica, tiempo real significa que la retroalimentación visual y táctil sean reproducidas en la frecuencia correcta. Ser capaz de proporcionar un alto grado de realismo e interacción en la simulación quirúrgica es el más difícil reto de la misma.

1.1 Antecedentes

Los algoritmos de corte en la simulación quirúrgica van acoplados con la deformación y visualización de mallas. Las cuales son utilizadas para representar tanto modelos volumétricos como superficiales, dada la complejidad de aplicar un algoritmo de corte sobre un modelo volumétrico, los algoritmos de corte primeramente fueron aplicados sobre un malla superficial en [15], Basdogan y otros, usan la estrategia de determinar y duplicar el punto por donde ha interceptado el triángulo la herramienta virtual de corte. Después emergen las ideas de Bruyns y Montgomery [16] introduciendo un esquema de subdivisión que ofrece un modelo más acertado de la representación de corte en superficies, además en su solución permite el trabajo con múltiples capas. Evidentemente los métodos basados en mallas de superficie, no permiten el corte en modelos volumétricos. Las investigaciones en este campo comenzaron con un algoritmo que describe la idea de remover los elementos que estaban en contacto con el instrumento de corte, planteado por Bro-Nielsen [17] y S. Cotin [18], más tarde retomado por Forest [19]. Lamentablemente, la idea de quitar los elementos, conllevaba a imperfecciones visuales, porque el camino que se representa no queda aproximado a la trayectoria del corte, conjuntamente es violado el principio de conservación de la masa. Sin embargo los métodos más atractivos surgieron después, que son los llamados métodos de subdivisión, estos tienen en común la clasificación del corte de acuerdo a los estados de intercepción de instrumento con la malla, en fin, estos algoritmos predefinen los estados posibles del corte y luego los ejecutan de acuerdo el correspondiente en cada caso. Esto fue introducido por primera vez en el contexto médico cuando Mazura y Seifert [20], plantean el corte de una malla tetraédrica con planos predefinidos. Este algoritmo fue refinado más tarde por Mor y T. Kanade [21], abordando temas como la incisión parcial de la malla y la estrategia del corte progresivo. Finalmente Bielser [22] plantea una máquina de estados, que conserva los diferentes tipos de incisiones en una malla tetraédrica.

Todos los algoritmos enunciados, tienen en común el incremento de los elementos en la malla, se ha reportado en un rango de 5 a 17 nuevos elementos por tetraedro interceptado en las mallas volumétricas. Por otra parte, la introducción de nuevos elementos extiende los cálculos en el modelo, otro aspecto negativo es la reducción del tamaño de los tetraedros que implica problemas en la estabilidad de la deformación, como los reportados por Mor y T. Kanade [21].

Algunas de estas dificultades fueron reducidas con la adaptación que hicieran Serby y otros [23], además de Nienhuys [24]. La idea consiste en acercar los nodos que componen los ejes, al camino

descrito por el instrumento de corte. Este acercamiento posibilita la incisión en la malla sin la creación de nuevos elementos, desafortunadamente surge otro problema, que es la degeneración de los elementos cuando el instrumento de corte atraviesa la malla cerca de un nodo.

1.2 Representación Geométrica de la Malla

En la representación geométrica de los cuerpos tridimensionales se utilizan modelos geométricos, para representar los objetos de modo que permita una visualización más exacta de diferentes elementos (nubes, árboles, rocas, edificios, órganos, mobiliario, etc.). Dentro de los modelos, se puede citar los de superficies poligonales y cuadráticas, superficie splines, modelos volumétricos, modelos procedurales y modelos basados en propiedades físicas. Debido al alcance de este trabajo, solo se describen los modelos superficiales poligonales y los modelos volumétricos.

1.3.1 Modelos de Superficie

Estos modelos se caracterizan por representar el modelo tridimensional solo en su superficie, a través de una colección de vértices y aristas, pudiendo formar a su vez polígonos, de forma que cada arista es compartida como máximo con dos polígonos, es lo que se denomina **mallas poligonales**.

Vértice: punto de coordenada (x, y, z), en tres dimensiones.

Arista: segmento de línea que une dos vértices.

Polígono: secuencia cerrada de aristas.

Las mallas poligonales tienen diferentes topologías, esto está en dependencia de la eficiencia que requiera la manipulación del modelo.

Cinta de triángulos *(triangle strip)***:** Para una cantidad de vértices n produce (n-2) triángulos conexos.

Abanico de triángulos: Para una cantidad de vértices n produce (n-2) triángulos conexos.

Malla de cuadriláteros *(quadrilateral mesh)*: Genera una malla de (n-1) por (m-1) cuadriláteros para m por n vértices.

Fig. 1: Topologías de mallas poligonales

Los modelos poligonales son de gran uso en los gráficos por computadoras y en los programas de diseño 3D. La mayoría de los algoritmos en los engines están soportados con la manipulación de la información de triángulos en estructuras de datos. Los triángulos son considerados la base de las primitivas gráficas por su simplicidad y el soporte que brindan las herramientas actuales para su visualización.

1.3.2 Generación de la Malla Volumétrica

Los objetos de una escena virtual, pueden ser construidos por softwares de diseño (3D StudioMax, Blender, etc.), estos brindan la posibilidad de enmallar solo superficialmente. Para visualizarlos en una escena, es necesario leer la estructura del fichero, que exporta las coordenadas de los vértices de cada triángulo que conforman el modelo, materiales, luces, entre otros. En la realidad virtual están estrechamente relacionados la geometría y la informática, así es que se han creados algoritmos para generar mallas superficiales y volumétricas. Los modelos de mallas volumétricas son ampliamente difundidos para la representación de órganos y reconstrucción 3D a partir de imágenes.

Los modelos volumétricos se encuentran formados en el interior del objeto por tetraedros, hexaedros o cubos para mencionar los más comunes. Lograr esta característica implica usar algoritmos de enmallado que permitan representar al cuerpo sólido. Los principales algoritmos para este proceso se dividen en algoritmos triángulo/tetraedro y algoritmos cuadrilátero/hexaedro, por ser los primeros los más difundidos, a continuación se describen las categorías donde se encuentran las técnicas más usadas: Octree, Delaunay, Advancing Front.

Octree: Fue primeramente desarrollada en 1980 por Marck Shepard. Con este método, los cubos contenidos en el modelo geométrico son subdivididos recursivamente hasta alcanzar la resolución deseada. La figura 2. muestra la descomposición, el equivalente quadtree en dos dimensiones, celdas irregulares son creadas cuando el cubo intercepta la superficie. Muchas veces requiere de grandes cálculos en la intercepción con la superficie. Los tetraedros son generados en las celdas irregulares de la superficie y en las regulares internas.

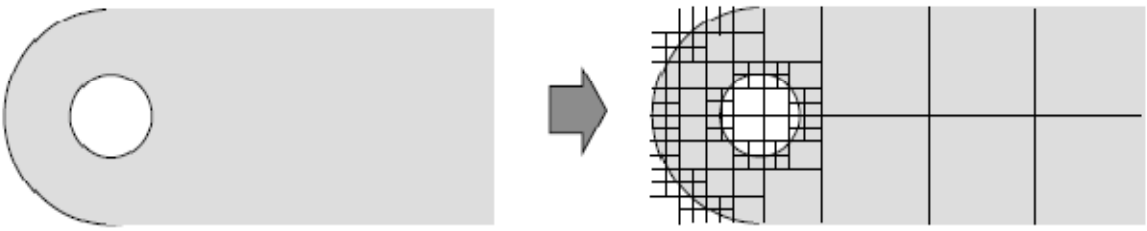

Fig. 2: Descomposición en Quatree de objeto 2D

Delaunay: Por mucho la técnica más popular para generar mallas triangulares, es el criterio de la Delaunay. Plantea que algún vértice de un tetraedro no debe estar contenido en la esfera circunscripta, si se traza una esfera por los nodos de un tetraedro y ésta no contiene a todos cumple con el criterio de Delaunay. Así se garantiza una triangulación dado un conjunto finito de puntos.

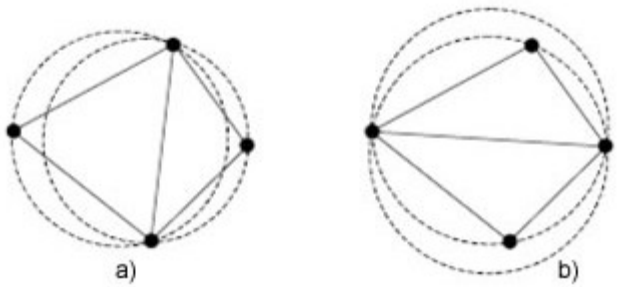

Fig. 3: Criterio de Delaunay a) cumple condición b) no cumple condición

Advancing Front: Es muy popular en la familia de algoritmos de generación de triángulo/ tetraedro. En este método el tetraedro es construido progresivamente hacia dentro desde la superficie triangulada del objeto. La figura 4. muestra un ejemplo en dos dimensiones de Advancing Front, donde los triángulos son formados desde la superficie, el algoritmo avanza desde el frente (dígase así en este caso a la superficie hasta rellenar todo 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 tetraedro para garantizar que no se superpongan al avanzar uno hacia el otro.

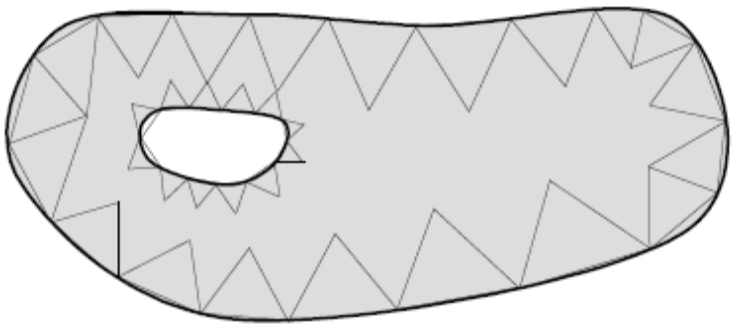

Fig. 4: Ejemplo de Advancing Front 2D

1.3 Modelos de Deformación Basados en Física

Para lograr un nivel aceptable de realismo al realizar el corte se necesita una solución físico – matemático que soporte las deformaciones, se han realizado disímiles estudios en este campo aportando novedosos resultados. “La potencia de los sistemas computacionales se incrementa a diario y muchas aplicaciones están demandando las técnicas más relevantes en realismo e interactividad. En particular tienen un gran interés en la simulación y manipulación de objetos deformables” [25].

En simuladores quirúrgicos el tejido humano generalmente es representado por un modelo geométrico y físico. Los cuerpos blandos son físicamente y geométricamente complejos en su representación, con frecuencia se dividen en un conjunto de pequeños elementos para facilitar su análisis. En las últimas dos décadas muchos modelos que están basados en el concepto de la discretización de los elementos dentro de los más prominentes están; sistema Masa **–** Resorte, Método de Elementos Finitos, Método de Elementos de Frontera, Método de Elementos Alargados entre otros. Para poder hacer un análisis exhaustivo de cada modelo hay que tener presente; el tiempo de interacción, estabilidad numérica y realismo. A continuación se describen los conceptos esenciales de los modelos físicos, poniendo al relieve ventajas y desventajas.

1.3.1 Sistema Masa – Resorte

El sistema Masa - Resorte es una técnica basada en física que ha sido usada ampliamente para modelar objetos blandos. El modelo físico consiste en la estructura de la red que está formada por nodos unidos por enlaces elásticos. La red del sistema Masa - Resorte es mapeada en la malla geométrica haciendo coincidir la masa de los nodos con los vértices y los muelles de la malla. El sistema de Masa-Resorte es usado para discretizar las ecuaciones de movimiento, por ejemplo, el par de nodos conectados permite la descripción de las propiedades elásticas del desplazamiento, cualquier cambio relativo a la longitud de los muelles produce una fuerza entre los nodos conectados.

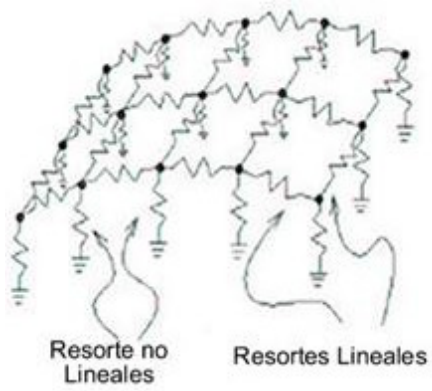

Fig. 5: Modelo de un Sistema Masa – Resorte

En un sistema dinámico la Segunda Ley de Newton gobierna el movimiento de cada punto en la rejilla:

$$Mi\,Xi + Di\,Yi + \sum_{j} Fi\,int = \sum Ri\,ext \qquad (1)$$

donde para cada nodo i , X_i y Y_i son las su velocidad y aceleración M_i es la masa, D_i es el coeficiente de amortiguación, $R_{i\,ext}$ es la fuerza externa resultante aplicada y $F_{i\,int}$ es la fuerza interna ejercida por el nodo vecino j al cual el nodo i está conectado. Generalmente esta fuerza interna es la respuesta viscosa elástica de los muelles conectados y está dada por:

$$F_i^{int} = (\lambda \Delta d + \mu \dot{d})k \qquad (2)$$

donde λ , es el factor de rigidez del muelle, μ indica el factor de amortiguamiento, Δd y $\dot{d}$ es la variación relativa de la distancia y la velocidad de dos partículas conectadas respectivamente, por último k es el vector unitario que une esos dos nodos.

El sistema Masa-Resorte es un simple modelo físico con una sólida fundamentación matemática. Es computacionalmente ligero y relativamente pequeño [26] 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.

Los sistemas Masa-Resorte han sido usados ampliamente en las animaciones faciales estáticas y dinámicas [27] [28]. También han sido manejadas para la simulación de ropa, video juegos y películas de animados. Muchos métodos se han propuesto para evadir la inestabilidad numérica [29][30][31]. Muchas investigaciones abordan el tema de las redes Masa-Resorte mejorando varios aspectos como el refinamiento adaptativos de parámetros [32] y el control de las características del material simulado [33]. Hace algunos años Brown y Montgomery desarrollaron un simple pero eficiente algoritmo basado en modelo Masa –Resorte para la simulación de microcirugía [34].

Fig. 6: Modificación de una topología 2D usando sistema masa-resorte. Tomado de [35].

1.3.2 Método de Elementos Finitos

El Método de Elementos Finitos *(Finite Element Method, FEM)* ha sido estudiado durante mucho tiempo, fe de ello son los trabajos de Zienkiewicz [36], Bathe [37], es uno de los más populares métodos en las Ciencias de la Computación. El 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, cuyo comportamiento se especifica mediante parámetros asociados a los puntos característicos denominados nodos, en cada uno de estos elementos del campo de la deformación es expresado por una función polinominal de interpolación en cada uno de los nodos de los elementos.

“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” [38].

Pudieran definirse una guía para el método como sigue:

Discretización geométrica: El cuerpo se subdivide en elementos más simples (usualmente tetraedros o hexaedros), en cada elemento se definen nodos que son puntos de control donde se evalúa el problema y describen localmente el material del objeto. La exactitud del método es directamente proporcional a la cantidad de subdivisiones hechas, ver la figura 7.

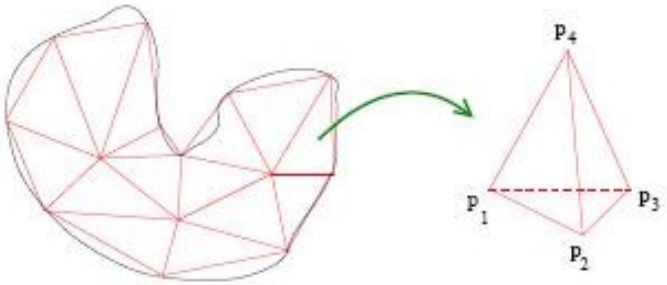

Fig. 7: Discretización de un dominio en elementos tetraédricos

Interpolación: Se usan funciones de forma (también conocidas como funciones base) para interpolar cualquier punto P del continuo, por tanto, cualquier función continua aplicada a un cuerpo continuo es aproximada por un sistema de ecuaciones finito en término de coordenadas de los nodos.

Aplicación de la Mecánica Continua: Una vez discretizado el cuerpo y las funciones de forma se deben encontrar expresiones que empleen las ecuaciones de elasticidad en términos de los nodos de la malla, esto permitirá calcular la distribución interna de tensiones del elemento. En dependencia de las necesidades del sistema su formulación puede ser estática o dinámica.

Formulación Dinámica: Usando la dinámica de Lagrange y la Teoría de la Elasticidad, la ecuación de movimiento del modelo deformable puede ser expresada por la ecuación diferencial de segundo orden:

$$Mx''+Dx'+Kx=F \tag{3}$$

Donde M , D y K son matrices de $3n \times 3n$ de masa, amortiguación y rigidez respectivamente, F es el vector fuerza aplicado al objeto. Una solución analítica no es posible para este sistema debido a su complejidad, por lo que su solución tendrá que ser numérica[39].

Formulación Estática: La ecuación 3 puede ser simplificada despreciando aspectos como la inercia, entonces la ecuación del movimiento puede ser expresada como:

$$Kx = F \tag{4}$$

En general FEM tiene cálculos muy densos, demasiado cargados para obtener resultados de interacción precisos. Globalmente FEM no es adecuado para aplicaciones interactivas en tiempo real por la capacidad de los procesadores. Pero si no hay cambios topológicos es posible obtener una simulación en tiempo real usando precómputos tal como Dimaio y Salcudean obtuvieron en [40]**.** No obstante está limitada a deformaciones pequeñas, la cual pudiera ser una desventaja en la simulación de tejidos deformables. Recientemente Mendoza y Laugier [41] presentaron una implementación de una formulación explícita de FEM tomando las largas deformaciones y los cambios topológicos, ellos manejan la idea de ejecutar el corte en humanos virtuales considerando el comportamiento humano [42].

1.3.3 Método Elementos de la Frontera

El Método de Elementos de la Frontera (*Boundary Element Method*, BEM) aparece en uno de los trabajos relacionados con la deformación de cuerpos es el realizado por James y Pai [43]. "Constituye una alternativa interesante al método estándar de Elementos Finitos, porque el cálculo se hace en la superficie del cuerpo en lugar del volumen" [44]. El propósito de usar este método casi estático para modelar la deformación describe como los objetos interactúan con el entorno en la superficie Γf . Por ejemplo la figura 8. presenta el desplazamiento en la frontera, el provocado por el usuario y el de la superficie, las restantes partes de la superficie del objeto quedan libres de movimiento.

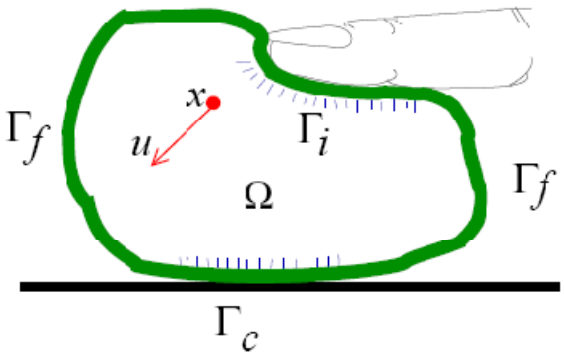

Fig. 8: Notaciones de BEM

Cuando el objeto se deforma, ocurre el desplazamiento n , del punto $x \in \Omega$. La deformación está basada en la ecuación de Navier, la cual es una generalización de la ley de Hooke:

$$Nu + b = 0 \tag{5}$$

donde N es la derivada de segundo orden y b es el término que representa las fuerzas, como gravedad o acción de otro agente externo sobre el objeto. Las condiciones de frontera junto con la ecuación de Navier constituyen el Problema de Valor en la Frontera (BVP por sus siglas en inglés). Para determinar el vector desplazamiento **u,** y el vector fuerza **T** en la frontera del cuerpo se usa la integral de superficie de Navier la cual queda expresada como:

$$\int_{\Gamma} U^{*}(x,y)\ p(y)\ d\Gamma(y) + \int_{\Omega} U^{*}(x,y)\ b(y)\ d\Omega(y) \tag{6}$$

A continuación se definen los pasos para la realización del Método de Elementos de la Frontera:

- Discretizar la frontera en un conjunto N no superpuestos de elementos n que representen el desplazamiento y las tracciones.
- Aplicar la ecuación integral Nu para determinar el desplazamiento de cada nodo N . Esto genera un sistema de $3N$ ecuaciones.
- Aplicar las condiciones de frontera deseadas al problema, fijando el valor de los nodos, desplazamiento y tracción por dirección.

BEM garantiza una velocidad sustancial porque el problema tridimensional lo reduce a dos dimensiones. Sin embargo el método solo funciona para objetos que en su interior estén compuestos por un material homogéneo. “En los cambios topológicos como corte y fractura son más difíciles de emplear que el método de FEM explicito” [45].

Fig. 9: Ejemplo de deformación usando BEM

1.3.4 Método de Elementos Alargados

Los modelos matemáticos para la deformación en entornos virtuales han adoptado disímiles formas. "El Método de Elementos Alargados (*Long Element Method,* LEM) ve el objeto en dos dimensiones distribuyendo los elementos llenos de un fluido incompresible. Las ventajas de este método es que el número de elementos en un orden de magnitud menos que la discretización basada en tetraedros o elementos cúbicos" [46]. En el método se asume que cada elemento está lleno con fluido, pero al mismo tiempo cada uno de los elementos obedece la Ley Hooke el Principio de Pascal y la Ley de Conservación del Volumen son usados en condiciones límites para establecer el equilibrio.

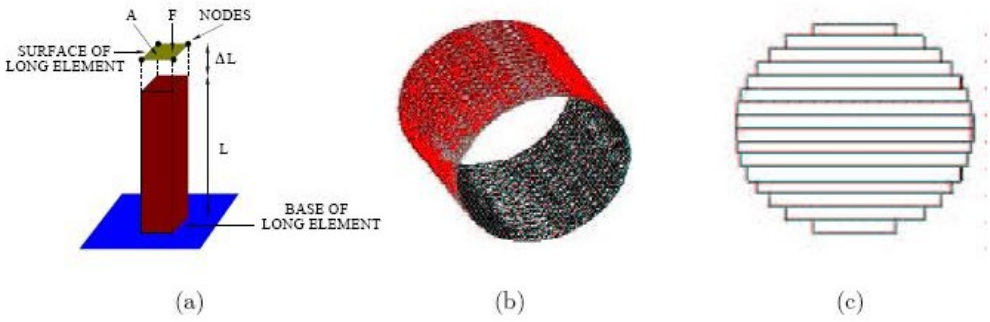

Fig. 10: a) Elemento alargado b) Malla cilíndrica con superficie discretizada c) Cilindro representado por un elemento alargado en 1D.

El problema de este método está básicamente dado porque no se conserva el volumen en grandes deformaciones. Una solución ingenua podría ser discretizar el objeto en cada intervalo de la deformación, pero evidentemente esta no es una solución muy eficiente. “La validez del LEM para aplicaciones interactivas en tiempo real es dudosa aunque muchos de sus planteamientos son razonables. Muchos aspectos de este método siguen siendo investigados profundamente” [42].

1.4 Tendencias de Corte

Una de las tareas básicas de la simulación quirúrgica es el fenómeno virtual del corte, este puede ser considerado como la interacción de un modelo virtual deformable y un cuerpo rígido, visto como herramienta de corte. En este proceso el modelo deformable es cortado y su topología geométrica sufre cambios. La mayoría de los trabajos realizados en el campo de la simulación virtual del corte, pueden ser clasificados en tres métodos fundamentales, las cuales se abordan a continuación:

- Método destructivo.
- Método de subdivisión.
- Método de separación.

1.5.1 Método destructivo

Considerado en la literatura del tema como el primer y más antiguo método de corte, basado en el principio de destrucción, es decir consiste en eliminar el elemento que está colisionado, con la herramienta virtual de corte.

Cotin en 1997 utiliza un modelo Masa-Resorte para la deformación y el corte [47] ver figura. 11. El corte fue implementado por la eliminación de los elementos que colisionaban con la herramienta virtual. Posteriormente Cotin, H. Delingette, y N. Ayache retoman este método [18], y en su trabajo describen el compartimiento entre algunos modelos de deformación como Masa-Resorte. También C. Forest, H.Delingette y N. Ayache realizan el corte por un algoritmo destructivo [19], su trabajo profundiza bastante el método, plantean soluciones y describen temas importantes, como el

refinamiento de la malla en las cercanías del corte para lograr una visualización más próxima a la realidad del proceso.

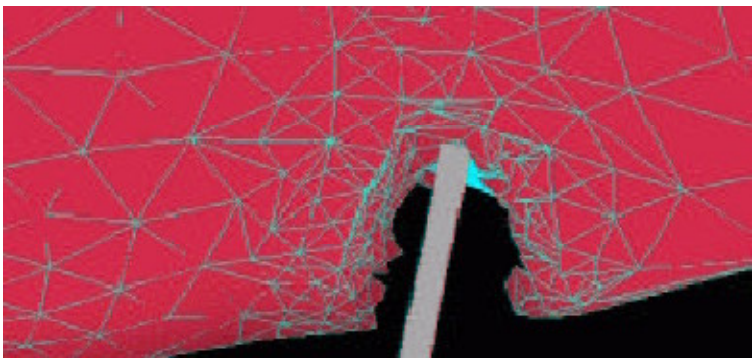

Fig. 11: Método Destructivo. Se destruyen los tetraedros colisionados por la herramienta de corte.

Eliminar aristas, nodos o tetraedros disminuye el número de elementos y con ello el tiempo de simulación disminuye. El método es adecuado solo para aplicaciones donde se destruye realmente el tejido, por otra parte el método viola el principio físico de la conservación de la masa, además requiere extensos procesos de refinamiento de la malla para mantener el realismo. Se han propuesto soluciones para el proceso de refinamiento, como por ejemplo realizar una refinación en el área donde se ejecutó el corte y en las otras llevar a cabo un procedimiento de refinamiento grueso, esto requiere que debe conocerse las áreas donde se realizó el corte, en el documento de C Forest, H.Delingette y N. Ayache [19] se analizan estos temas.

1.5.2 Método de subdivisión

Un nuevo procedimiento de corte es puesto en práctica por Bielser a finales de la década de los 90' empleando un modelo de bisturí como instrumento de corte. En el método, la malla es abierta exactamente donde el objeto de corte toca el modelo. El proceso de corte basa su principio en la subdivisión de los elementos colisionados por la herramienta virtual de corte.

Bielser utiliza un modelo Masa-Resorte en una malla tetraédrica para simular el corte y la deformación [48], en el trabajo se subdivide cada tetraedro interceptado por la herramienta de corte considerando las distintas formas en que un tetraedro pueda ser cortado, desde un corte parcial hasta un corte completo. De este análisis se guarda un conjunto de precortes en los tetraedros. Mor y Kanade

simulan el corte con un modelo de Masa-Resorte, aquí proponen un corte progresivo y minimizando el número de elementos a crear [21] ver figura. 12. Ellos basan el corte de un tetraedro en 5 formas posibles ver figura. 13, computan para cada uno de estos casos el conjunto mínimo de nuevos tetraedros necesarios para simular el corte. Para simular un corte progresivo mientras que la herramienta está dentro del modelo se crean tetraedros temporales, una vez que la herramienta sale del modelo los tetraedros son destruidos y sustituidos por un conjunto de mínima cantidad de tetraedros determinado por la trayectoria de la herramienta virtual. D. Bielser, P. Glardon, M. Teschner y M. Gross, plantean una máquina de estados que conserva los diferente tipos de incisiones en una malla tetraédrica, la máquina controla de manera eficiente el problema del corte progresivo, cada estado describe un conjunto de caras y vértices interceptados, definiendo así una topología particular para estos tetraedros [21].

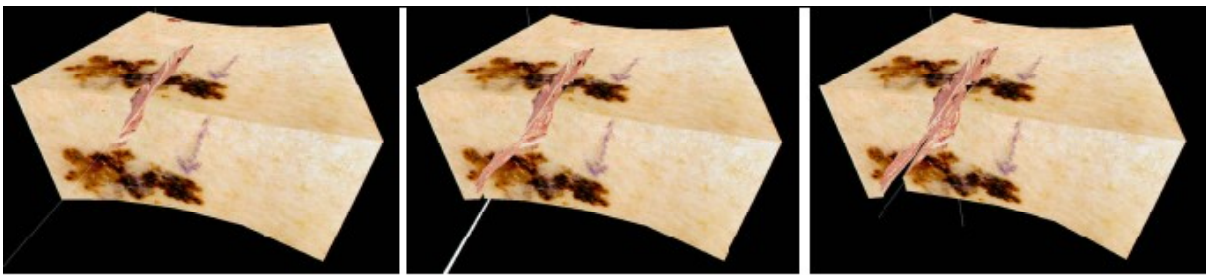

Fig. 12: Ejemplo de corte de Bielser, en un objeto de 576 tetraedros. Tomado de [49].

Fig. 13: Los 5 casos de subdivisión en tetraedros después de completado el corte.

El método de subdivisión cumple con el principio físico de conservación de la masa, pues, los tetraedros involucrados en el proceso de corte no son eliminados, sino sustituidos por subdivisiones de ellos mismos. No se requiere de extensos procesos de refinamiento de la malla para lograr el realismo en la simulación del corte, sin embargo para cada tetraedro en el proceso de corte, se crean

y adicionan nuevos al sistema, acción que provoca lentitud en el ciclo de la simulación. En el mejor de los casos se crean 6 nuevos tetraedros por cada uno que sea cortado, esto demanda un gran aumento del procesamiento de cálculos, lo que puede crear inestabilidad en el sistema, brindándole ventajas a técnicas precomputarizadas del método, como la máquina de estado [48].

1.5.3 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 en vez de eliminarlas o subdividirlas. Las primitivas que son colisionadas por la herramienta virtual, se duplican los nodos significativos para el corte y se procede a la separación apoyándose en estos.

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 laparoscópica [35]. Nienhuys basa su algoritmo de corte en el método de separación, utiliza un modelo linear de elementos finitos [49]. 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 [47] como se muestra en la figura 14. 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 [50].

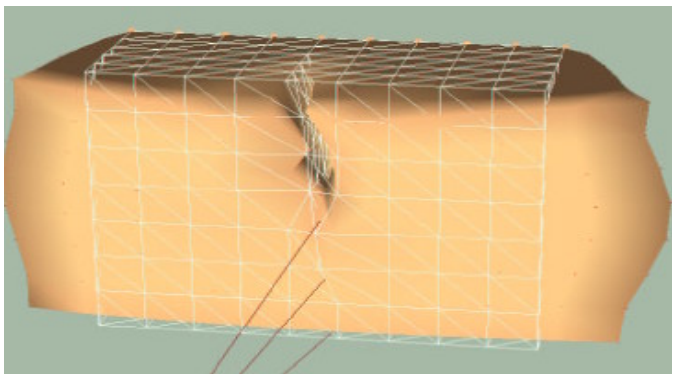

Fig. 14: Incisión en una malla de 480 nodos.

El método de separación garantiza que no incremente el número de elementos del modelo en el proceso de corte, lo que a diferencia ocurre en el método de subdivisión, evitando así perjudicar la velocidad de la simulación. La separación a diferencia del método destructivo cumple con las leyes físicas de conservación de la masa y mantiene las propiedades físicas del modelo. Desafortunadamente la degeneración de los elementos es un problema que puede ocurrir cuando el instrumento de corte atraviesa la malla cerca de un nodo.

1.5.4 Corte Progresivo en modelo superficial 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 [51] se propone un nuevo método para generar una estructura interna en un modelo superficial acorde con la interacción con la herramienta de corte.

En el trabajo se definen términos importantes como estado de contacto y estado de penetración, el primero cuando no hay suficiente fuerza para penetrar el modelo y en el segundo cuando el instrumento de corte penetra el modelo superficial. Para la deformación utilizan un modelo Masa-Resorte con una base sólida donde reposa el objeto [52] ver figura 15.

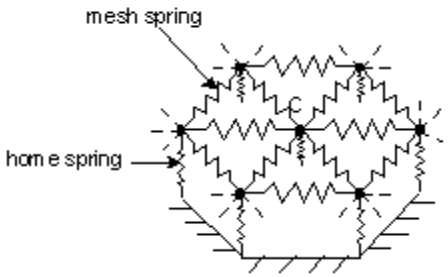

Fig. 15: Modelo Masa Resorte con base sólida. Tomado de [51]

La estructura generada en el interior de la superficie es llamada “Ranura” y la modificación de la topología se implementa basándose en la técnica de subdivisión, nuevos vértices son adicionados a la malla inicial y conectados con los vértices originales de esta.

Para la generación de la ranura se utiliza el comienzo y final del corte en el triángulo (ABC) para definir la profundidad ($G1\ G2$), en la figura 16. $G1$ corresponde a la profundidad del instrumento cuando es interceptado con el punto $P1$ y $G2$ en el punto $P2$. Como resultado, se pueden generar cuatro triángulos dentro de la apertura como ranura. Los triángulos serían ($V1V2G1$, $V1V3G1$, $G1G2V2$, $G1G2V3$).

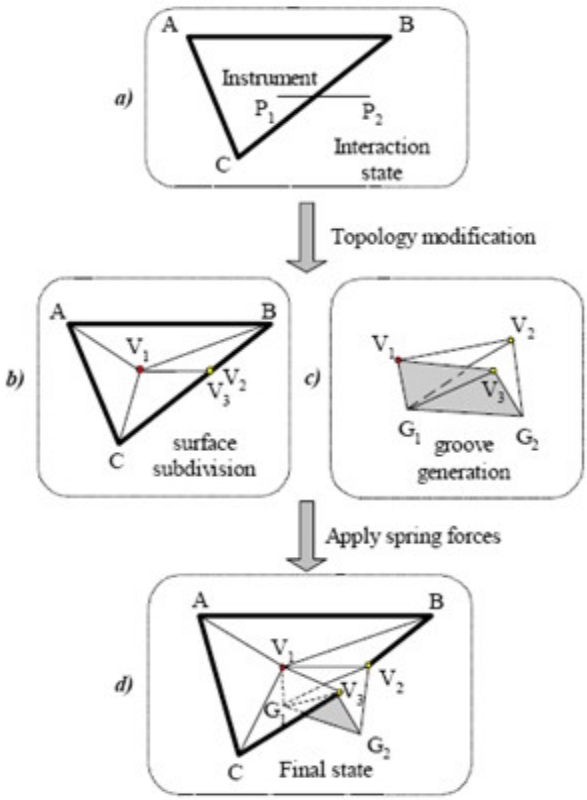

Fig. 16: Generación de ranura interna. Tomado de [51]

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

1. La generación de ranura interna temporal, en la cual se trata cada triángulo como un estado final del corte y si luego se actualiza este, en caso de adición de nuevos triángulos al proceso.
2. La unión de dos cortes es otra vía de solución, permite realizar un proceso cíclico de corte.

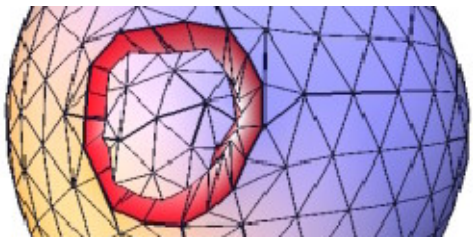

Fig. 17: Ejemplo de realización de un proceso cíclico de corte.

El método de separar una malla superficial y generar ranura interna es novedoso y eficiente comparado con el modelo volumétrico ampliamente utilizado. Aun así hay áreas en las que se debe trabajar como la división de un objeto en dos partes.

1.6 Interacciones

Detección de Colisiones: Este es el primer paso para ejecutar una interacción realista. De hecho es necesaria para saber donde colisionan los objetos. Sin embargo localizar la primitiva entre dos objetos puede ser costoso computacionalmente, especialmente si el objeto está compuesto por miles de polígonos.

Efectos de Colisión: Las interacciones físicas asumen que la colisión entre dos objetos causa cambios en la velocidad de los objetos involucrados, transformación en la cinemática, la energía potencial y la generación de fuerzas de repulsión. Estos efectos separan los objetos virtuales evitando que se interpongan.

Contacto de Deformación: La colisión entre dos objetos además de separarlos, crea una deformación en los mismos. De esta manera se usan modelos físicos del objeto para producir las fuerzas repulsivas como fuerzas externas para deformar el objeto.

Cortar Tejido Blando: La simulación quirúrgica incluye la ejecución de tareas complicadas en los órganos virtuales: desgarrar, cortar, coser, etc. Considerando esto, los órganos humanos como objetos virtuales, los mismos están compuestos por miles de polígonos, estas manipulaciones causan cambios en la forma que los polígonos están unidos. En otras palabras la topología original de la malla es modificada.

1.6.1 Detección de Colisiones

La detección de la interacción entre dos elementos virtuales es conocida como detección de colisiones. Diferentes enfoques para la detección de colisiones dependen de la representación del objeto que pueden ser modelos no poliédricos o modelos poliédricos. En general para determinar el contacto entre dos objetos virtuales se calculan las distancias entre ellos. Si la distancia es negativa entonces los objetos están en contacto.

Existen tres principales aspectos que determinan la interactividad cuando se trata de localizar el lugar exacto de la colisión

- ✓ La forma geométrica de los objetos (cubos, esferas, poliedros, etc.).
- ✓ El número de polígonos que define el objeto cuando el objeto es construido por miles de polígonos, el cálculo del punto de colisión podría ser muy costoso computacionalmente.
- ✓ La naturaleza física de los objetos (rígido o deformable).

Fases de la Detección de Colisiones

En general los entornos virtuales donde están presentes elementos rígidos o deformables, la estrategia para la detección de colisiones está dividida en tres fases: amplia, estrecha y exacta.

Fase Amplia: En esta primera fase de optimización el par de objetos que probablemente colisionan son seleccionados. Luego este par de objetos son testeados por interferencia. Los algoritmos usados en esta fase principalmente están basados en descomposición espacial.

Voxel: El algoritmo descompone el espacio en partes uniformes o celdas llamadas voxels [53]. El método más usado para la descomposición espacial es mediante octrees, las celdas nunca se sobreponen, nunca se extienden más allá del espacio del padre y su unión siempre cubre el espacio del padre. Por ejemplo se forma una cuadrícula de igual tamaño de celdas, para cada celda en la cuadrícula se almacenan la lista de objetos que comparten la misma celda. Para procesar la colisión sería identificando las celdas que comparten entre un par de objetos, si existe una celda en común entonces se procede a determinar la colisión.

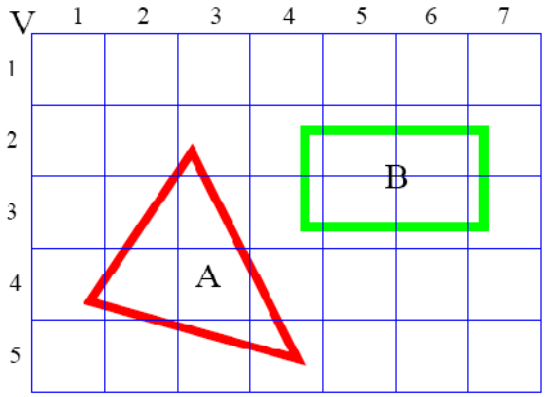

Fig. 18: Celdas de la cuadricula asociada a los objetos Ay B.

Fase Estrecha: En esta fase no se calcula con exactitud la entidad donde colisionan el par de objetos, pero da una idea preliminar acerca de la región donde los objetos pudieran colisionar. Esta zona se llama zona de colisión y los algoritmos están basados en la descomposición del objeto, esta es otra fase de optimización.

Jerarquía de Volumen Envolvente: El volumen envolvente encierra el objeto virtual en acción. Por ejemplo una caja o una esfera encierra un objeto, el problema de la colisión se reduce a determinar cuando los volúmenes envolventes se sobreponen. Las jerarquías de volumen son usadas cuando la complejidad del objeto es alta. La jerarquía es un árbol que contiene volúmenes envolventes, esferas o cajas, cada volumen encierra una o varias primitivas geométricas. El conjunto de los volúmenes es llamado padre pues es el que contiene todas las primitivas abarcadas por sus nodos hojas. La fortaleza de este método está en el tipo de jerarquía que se utilice para hacer las pruebas de colisiones. Cajas Envolventes de Ejes Alineados (AABBs siglas en inglés) [54], Cajas Envolventes Alineadas Arbitrarias (OBBs siglas en inglés) [55] o Jerarquía de Esferas [56].

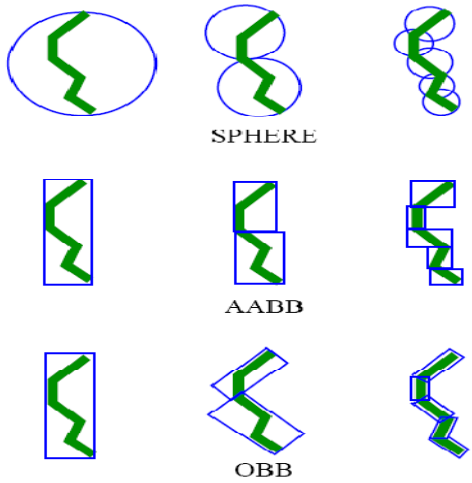

Fig. 19: Jerarquia de Volúmenes Envolventes. Tomado de [42]

Fase Exacta: Esta última fase obtiene la entidad exacta de colisión. El algoritmo de esta fase chequea la colisión entre dos polígonos

Es esta fase el algoritmo de colisión usado determina si al menos dos primitivas se interceptan. Se consideran fundamentalmente tres pares de primitivas esfera/esfera, caja/caja, triángulo/triángulo.

Consideraciones del Capítulo

En este capítulo se trataron temas fundamentales a tener en cuenta para simular el proceso de corte para simuladores quirúrgicos. Se destacaron aspectos concernientes a la topología y estructura de la malla tridimensional. Queda descrito el estado de arte de los métodos para dar solución a la deformación propiedad que está intrínsecamente relacionada con el corte. Al final se plantean las tendencias actúales para obtener un algoritmo de corte, donde se perfilan las ventajas y desventajas que acarrean cada uno de estos.

Con el estudio del marco teórico conceptual realizado se considera que existen las bases suficientes para proponer una solución en el siguiente capítulo.

Capítulo 2

Soluciones Técnicas

Introducción

El corte de mallas superficiales genera cambios en la topología de las estructuras que conforman la malla, el objetivo que compete este trabajo es modificar la malla por cada interacción de la herramienta de corte con el cuerpo, de manera que se brinde una solución visual, mostrando en la trayectoria de la herramienta de corte una abertura, lo que significaría un aporte, como por ejemplo en el corte de tejidos para un cuerpo u órgano, en un simulador quirúrgico.

En este capítulo se describe de manera general el proceso de corte, se explican los métodos para generar la subdivisión de los elementos, como se maneja la detección de colisiones y su seguimiento mientras se realiza el corte.

2.1 Procedimiento de Corte

En el capítulo anterior se describían tres tendencias existentes para efectuar el proceso de corte, el primer método planteaba que cuando un elemento del modelo era interceptado por la herramienta de corte, se eliminaba completamente, el segundo método comenzaba el corte a partir del punto exacto de la intercepción de la herramienta de corte y el modelo se subdividía insertando nuevos elementos por cada triángulo interceptado y el tercero consistía en adaptar los puntos de contacto de la herramienta de corte con los vértices que conforman los triángulos de la malla, separando los lados que unen los vértices más cercanos a la trayectoria del corte. El primero no preserva el volumen ni la masa del modelo, el segundo método necesita de altos procesos de cálculos, aún con las mejoras introducidas de máquinas de estados o precálculo de las posibles incisiones a realizar, fundamentalmente en cuerpos volumétricos, por último el método separación promueve eliminar la entrada de nuevos elementos a la malla mientras por su parte se pueden degenerar triángulos al pasar la herramienta de corte cerca de los vértices del los triángulos. Finalmente este trabajo propone una solución usando la tendencia de subdivisión para mallas superficiales. Está claro que se introducen nuevos triángulos en la malla pero se propone un algoritmo basado en lo planteado por Hui Zhang y otros en [51], para resolver el problema de simular el corte.

El procedimiento general para cortar elementos en el simulador basado en mallas triangulares consiste en los siguientes pasos. Primero la intersección inicial entre el modelo y el instrumento de corte es detectado. Para hacer esto, se comprueba los triángulos contra el movimiento del instrumento para determinar donde ha sido interceptado el cuerpo, una vez detectada la colisión se almacena una colisión inicial. Luego se mantiene el chequeo para todos los triángulos y aristas vecinas del primer triángulo interceptado contra el movimiento del instrumento de corte. Por último para cada elemento interceptado se subdivide creando nuevos triángulos en el modelo. El proceso se muestra en la figura 20.

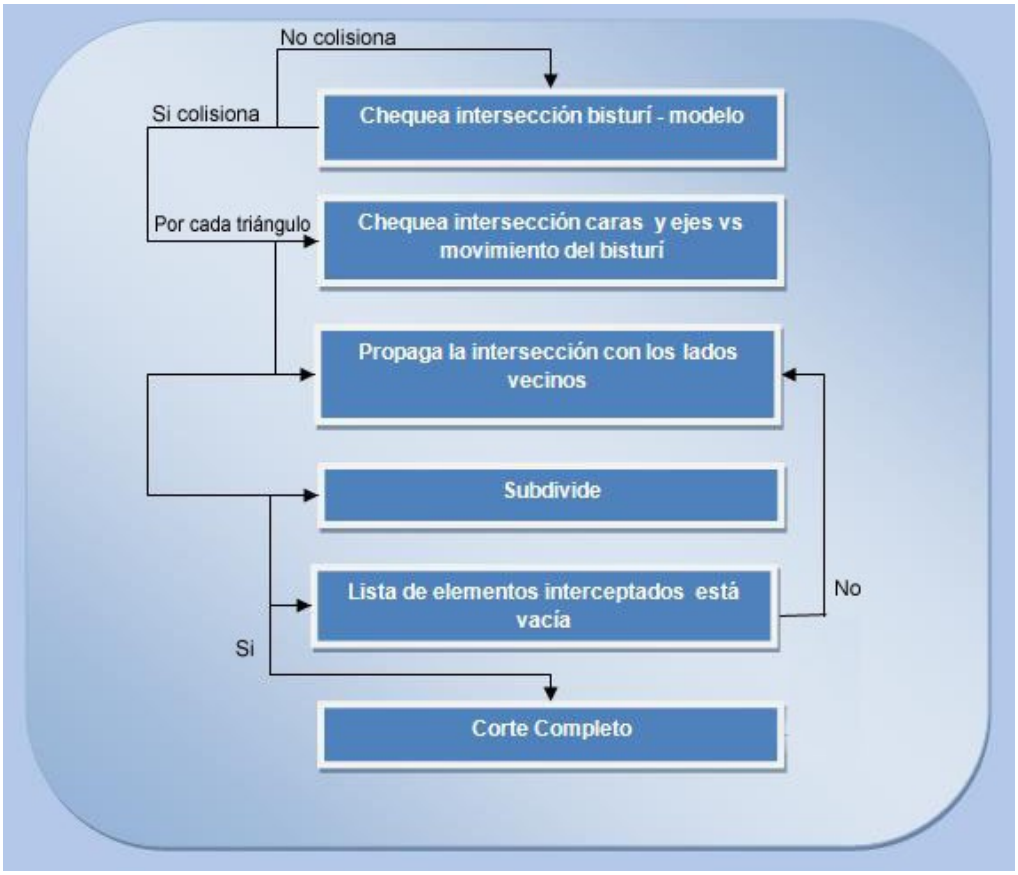

Fig. 20: Proceso general de corte

2.1.1 Subdivisión de los Triángulos

El algoritmo define dos fases esenciales la primera se encarga de subdividir los elementos que conforman la malla en el contacto inicial o final, la segunda definida como contacto intermedio, de esta forma se manejan los cambios de la topología en la malla en el transcurso de la operación de corte.

Como se muestra en la figura. 21 el camino descrito por el instrumento de corte $P1\,P2$ ($P1$ es el contacto inicial del instrumento con el modelo) solo con interceptar un lado del triángulo ABC , este triángulo estaría en cualquiera de los estado inicial o final del corte. Este triángulo es subdividido en cuatro nuevos elementos, dígase ($ABV1, ACV1, BCV1, BV1V2, CV1V3$) en b), donde $V1$ está en la posición de $P1$ y $V2V3$ corresponden a $P2$.

Si la trayectoria del instrumento intercepta dos lados (AB y AC en la figura 22a) del triángulo, este triángulo estaría en el proceso intermedio del corte. El triángulo ABC es subdividido en tres nuevos triángulos (AV_1V_3, V_3V_4C, BCV_4). Solo dos nuevos vértices son creados $V_3\ V_4$ en orden con los $V_1\ V_2$ generados como complemento de la previa subdivisión.

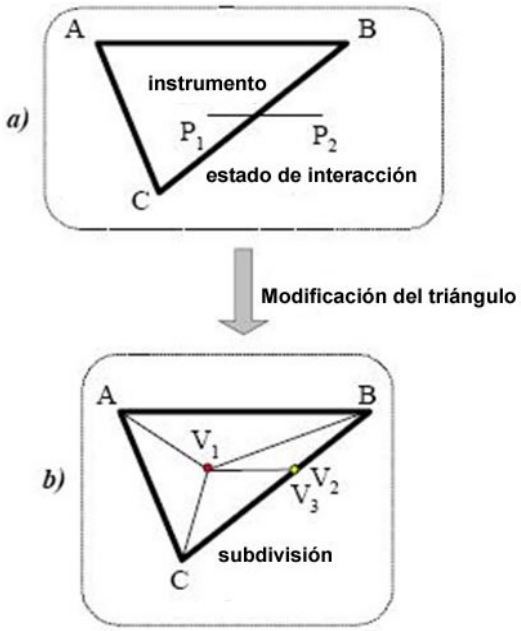

Fig. 21: Subdivisión de la superficie para los triángulos inicial – final.

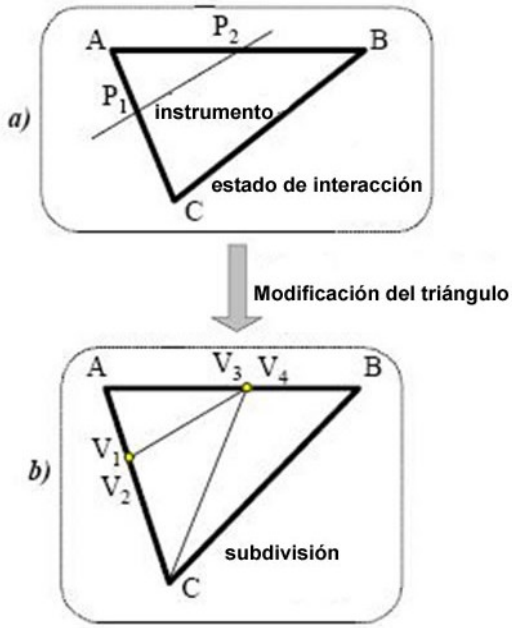

Fig. 22: Subdivisión de la superficie para los triángulos intermedios.

2.2 Detección de Colisiones y Propagación

El proceso de detección de colisiones se realiza localmente consultando a los triángulos vecinos de los interceptados en cada momento, garantizando así que la detención global de colisiones se utilice para encontrar la primera interacción; con la intención de lograr eficacia en el funcionamiento del sistema. Basados en la fase exacta de los algoritmos de detección de colisiones se obtienen los puntos donde interactúa el cuerpo con el instrumento de corte. El corte puede ser observado como el movimiento de un segmento cortante a través de un objeto, si la hoja del instrumento es ignorada y reduciendo el problema, a seguir el paso del segmento correspondiente al lado afilado del instrumento de corte en tiempo y espacio por una malla triangular superficial.

Las técnicas usadas para las pruebas de intersecciones, son soluciones de sistemas de ecuaciones vectoriales que dan como resultado si hubo intersección o no, seguidamente se muestran las ecuaciones de las rectas en la ecuación (7) en su forma vectorial.

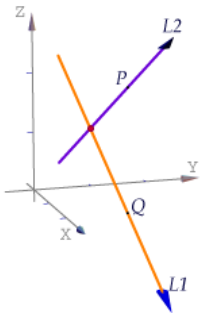

Fig. 23: Intersección de dos rectas

Sea $L1$ y $L2$ dos rectas en el espacio, sus ecuaciones estarían dada de la siguiente forma:

$$L1:(x,y,z) = P + t\vec{v}_1;\ t \in \Re \ \wedge\ L2:(x,y,z) = Q + s\vec{w}_1;\ s \in \Re \qquad (7)$$

Y la solución del sistema como se muestra:

$$P + tv = Q + sw, \text{ o sea,}$$

$$\begin{cases} tv_1 - sw_1 = q_1 - p_1 \\ tv_2 - sw_2 = q_2 - p_2 \\ tv_3 - sw_3 = q_3 - p_3 \end{cases}$$

Otra prueba extendida, es el tratamiento de las intersecciones segmento plano, figura 24, en este caso el plano debe ser acotado a un triángulo para determinar que el punto está intersecando un triángulo en específico, para darle solución se presentan las ecuaciones (8) de la recta y (9) el plano que es interceptado.

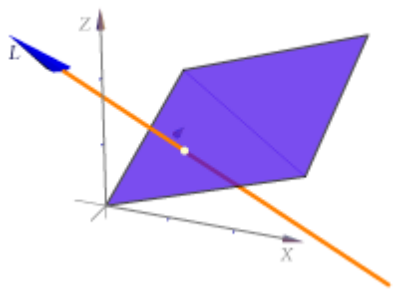

Fig. 24: Intersección de una recta L y el plano d.

Sean $L1$ una recta y un plano $\pi 1$ en el espacio sus ecuaciones están regidas a las siguientes expresiones matemáticas

Por ejemplo una recta

$$L1: (x, y, z) = P + t\vec{v} \quad (8)$$

y la ecuación cartesiana del plano $\Pi 1 : a1x + b1y + c1z = d$ (9)

Para obtener una intersección entre la recta y el plano se despeja $x\,y\,z$ de de la ecuación de la recta y sustituyendo $x\,y\,z$ en la ecuación del plano. Luego se resuelve para t , si la solución es única, con este valor de t obteniéndose el punto de intersección sustituyendo en la ecuación de la recta.

2.3 Metodologías, Herramientas y Lenguaje Utilizado

2.3.1 Metodología

La metodología aplicada fue el Proceso Unificado de desarrollo de Software (RUP) por ser un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software con diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Sin embargo, hay tres características fundamentales que lo hacen una metodología robusta y poderosa: es dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. Cuando se caracteriza RUP como dirigido a casos de uso se refiere a que se sigue un hilo, avanzando a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan, se implementan y a partir de ellos se obtienen los casos de prueba. El desarrollo de los casos de uso no se hace de forma aislada sino que se desarrollan con la arquitectura del sistema, por lo que tanto la arquitectura del sistema maduran a medida que avanza el sistema. Centrado en la arquitectura se refiere a que se incluyen los aspectos estáticos y dinámicos más significativos del sistema. Además recoge una serie de factores como la plataforma en la cual funciona el software, los bloques de construcción reutilizables que se tienen, consideraciones de implementación, sistemas heredados y requisitos no funcionales. Es iterativo e incremental porque el proyecto o el desarrollo de una aplicación se pueden dividir en partes y desarrollarlas de manera iterativa, incrementándose a medida que se integran unas con otras hasta llegar a formar la tarea final. Las iteraciones hacen referencia a pasos en el flujo de trabajo y los incrementos en el crecimiento del producto.

2.3.2 Visual Studio 2003

El Visual Studio 2003, es una herramienta poderosa, fuerte y voluminosa que tiene una gran integración de varios lenguajes entre ellos el C++, C#, y Asp.Net. Tiene la posibilidad de implementar aplicaciones para soluciones integrales que aprovechen de manera óptima la ventaja de cada lenguaje. Acelera de manera significativa la producción de software. Mejora en un alto grado los resultados finales y optimiza el desempeño. La documentación del Visual Studio está entre las mejores. La interfaz es altamente amigable con el usuario permitiendo que el tiempo en implementar una solución o aplicación determinada sea mucho menor. Los ejecutables desarrollados por esta herramienta son generalmente de menor tamaño lo que hace que ocupe un lugar cimero en la producción de software al lograr aplicaciones óptimas y de poco volumen. Otro de los motivos por los cuales se usa esta herramienta para desarrollar el software propuesto es que esta es la plataforma de desarrollo usada en el proyecto Simulador Quirúrgico para las realización de sus productos.

2.3.3 Enterprise Architect

Enterprise Architect (EA) combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer un modelado avanzado al escritorio, con un conjunto de características, EA puede equipar a un equipo entero, incluyendo analistas, evaluadores, administradores de proyectos, personal del control de calidad, equipo de desarrollo y más. Enterprise Architect es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multiusuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.

2.3.4 Lenguaje de Programación C++

El C++ se escogió por ser un lenguaje de programación estandarizado por la norma ISO/IEC 14882:1998. Entre sus principales características está el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica, además de ser un lenguaje de alto nivel que está considerado como un lenguaje potente al poder trabajar tanto en bajo, como en alto nivel. Posee dos propiedades fundamentales que son difíciles de encontrar en otros lenguajes que son la posibilidad de redefinir los operadores, comúnmente conocido como la sobrecarga de operadores, y la identificación de tipos en tiempo de ejecución. Además presenta una biblioteca estándar muy poderosa y es muy usado tanto en el ámbito educacional como profesional.

Consideraciones del Capítulo

Este capítulo concluye con una solución científica al problema que se plantea al inicio de la investigación, definiendo entre varios algoritmos expuestos en el capítulo anterior, el más conveniente para desarrollar, en aras de lograr alto nivel de realismo, eficiencia y velocidad de render. Se especifica además la metodología y herramientas de desarrollo de software a utilizar.

Capítulo 3

Descripción de la Solución Propuesta

Introducción:

El propósito de este capítulo es guiar el desarrollo hacia un sistema correcto. Esto se consigue mediante la descripción de los requisitos necesarios, que no son más que condiciones o capacidades que el sistema debe cumplir suficientemente buenas como para que se pueda llegar a un acuerdo con el cliente. El reto fundamental de este proceso consiste en que tanto desarrolladores como el cliente puedan entenderse en base a los resultados del levantamiento de requisitos. Como cada proyecto es singular, esto depende también del tipo de aplicación que se desea desarrollar, la manera de contextualizar el negocio del problema en cuestión fue a través de un modelo de dominio con la intención de comprender qué conceptos están relacionados en la descripción del negocio, se plantean los requisitos funcionales, no funcionales y se presenta un diagrama de caso de uso para resumir el proceso de captura de requisitos.

3.1 Reglas del Negocio

El sistema está basado mallas triangulares en la superficie de los modelos.

El desplazamiento de segmento cortante debe ser menor que el área de los triángulos.

3.2 Modelo del Domino

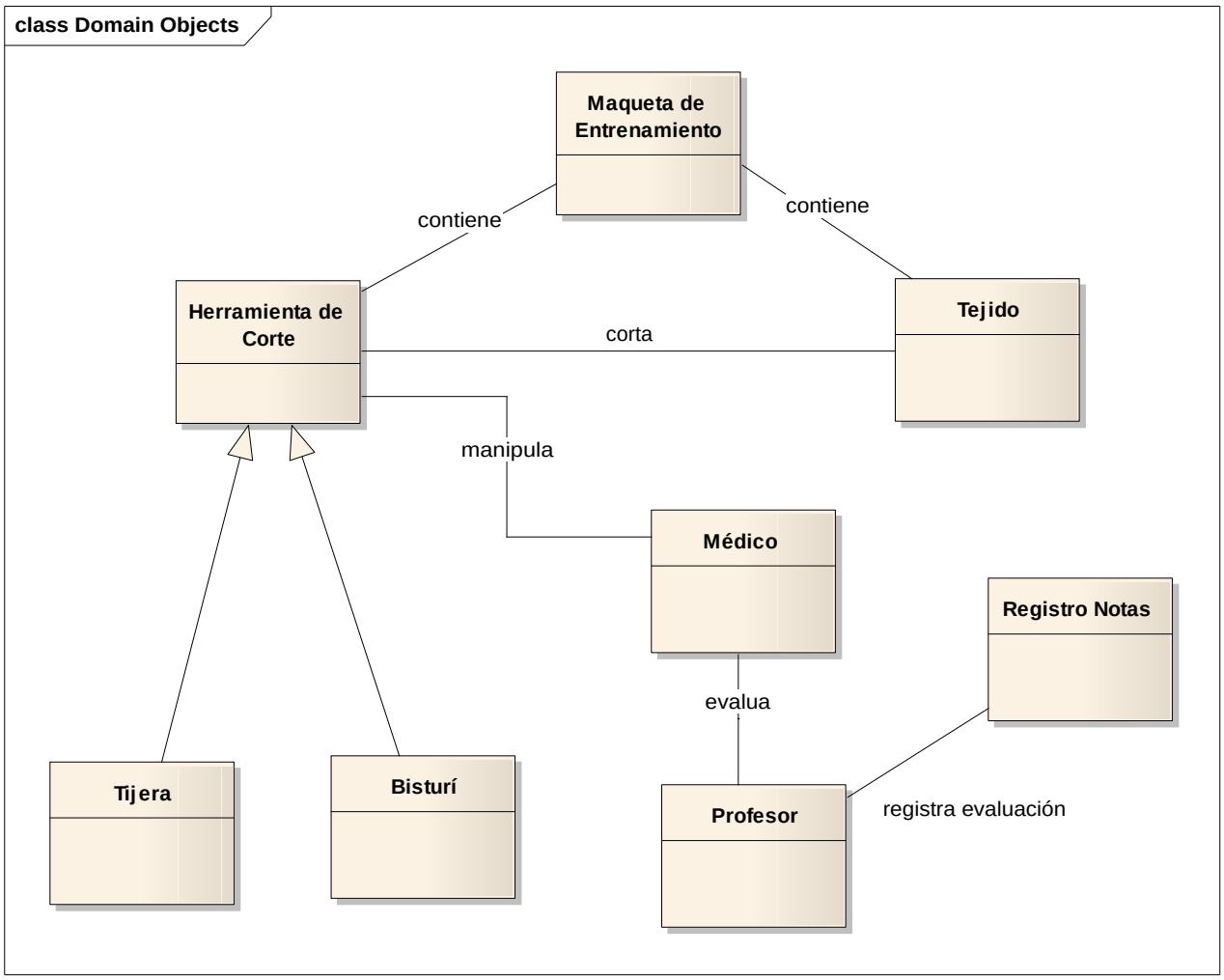

Fig. 25: Modelo de Dominio

Glosario de términos

Maqueta de entrenamiento: Es una caja donde se introducen las herramientas quirúrgicas por dos orificios simulando la cavidad abdominal humana. Se colocan dentro de esta los objetos para realizar los entrenamientos de los cirujanos que se especializan en la cirugía mínimamente invasiva.

Herramienta de corte: No son las herramientas quirúrgicas convencionales, sino que tienen un cuerpo auxiliar alargado, lo que permite al médico manipular desde fuera.

Tejido: Consiste en una porción de tejido la cual será sometido al corte.

Médico: Es quien se entrenará en la maqueta, con ejercicios que desarrollarán sus habilidades quirúrgicas para la cirugía de mínimo acceso.

Profesor: Es un médico con conocimientos avanzados, quien evaluará los procedimientos de médico en adiestramiento.

Registro de notas: Es donde se almacena todos los resultados, para controlar el progreso del los médicos en las habilidades de la cirugía mínimamente invasiva.

3.3 Levantamiento de Requisitos

El flujo de trabajo Requerimientos define las condiciones o capacidades que el sistema debe cumplir como los requisitos funcionales y las propiedades del sistema o restricciones impuestas como no funcionales. A continuación se describen las funcionalidades y características del sistema en desarrollo.

3.3.1 Requisitos Funcionales

R1. Subdividir la malla

R1.1 Duplicar vértices.

R1.2 Insertar vértices.

R1.3 Modificar vértices de la malla.

R2 Gestionar triángulos de la malla

R2.1 Adicionar triángulos en la malla.

R2.2 Buscar triángulos en la malla.

R2.3 Eliminar triángulos en la malla.

R2.4 Almacenar la lista de triángulos que son vecinos para cada triángulo.

R2.5 Retornar la lista de triángulos actualizada.

R3. Detectar intersecciones con el modelo

R3.1 Determinar puntos intersección entre herramienta y la malla triangular.

R3.2 Determinar la intersección entre el segmento de trayectoria y las aristas de los triángulos implicados.

3.3.2 Requisitos no Funcionales

Usabilidad: Los futuros usuarios del sistema serán programadores con conocimientos básicos de programación gráfica y de la terminología a fin. El módulo debe estar concebido para que el usuario piense en qué desea hacer y no en cómo hacerlo.

Rendimiento: Como aplicación de tiempo real, debe tener alta velocidad de procesamiento o cálculo, tiempo de respuesta, de recuperación y disponibilidad.

Soporte: Es compatible con la plataforma Windows y Linux.

Hardware: Se necesita 512MB memoria RAM. Mantiene compatibilidad con tarjetas gráficas de la familia NVIDIA.

Diseño e implementación: Debe utilizarse la biblioteca de clases STK la cual tiene definidas las funcionalidades básicas para trabajar con gráficos por ordenador, soporta OpenGL y DirectX. Se harán llamadas a dicha biblioteca desde el lenguaje C/C++. Se regirá por la filosofía de Programación Orientada a Objetos.

3.4 Modelo de Caso de Uso

En esta sección se reconocen los posibles actores del sistema a desarrollar y se conciben a través de la agrupación de los requisitos funcionales anteriormente especificados. Los casos de uso son los resultados de valor que les pueda brindar a sus actores, una vez interaccionen con el sistema. A continuación se exponen los casos de uso correspondientes al primer ciclo de desarrollo.

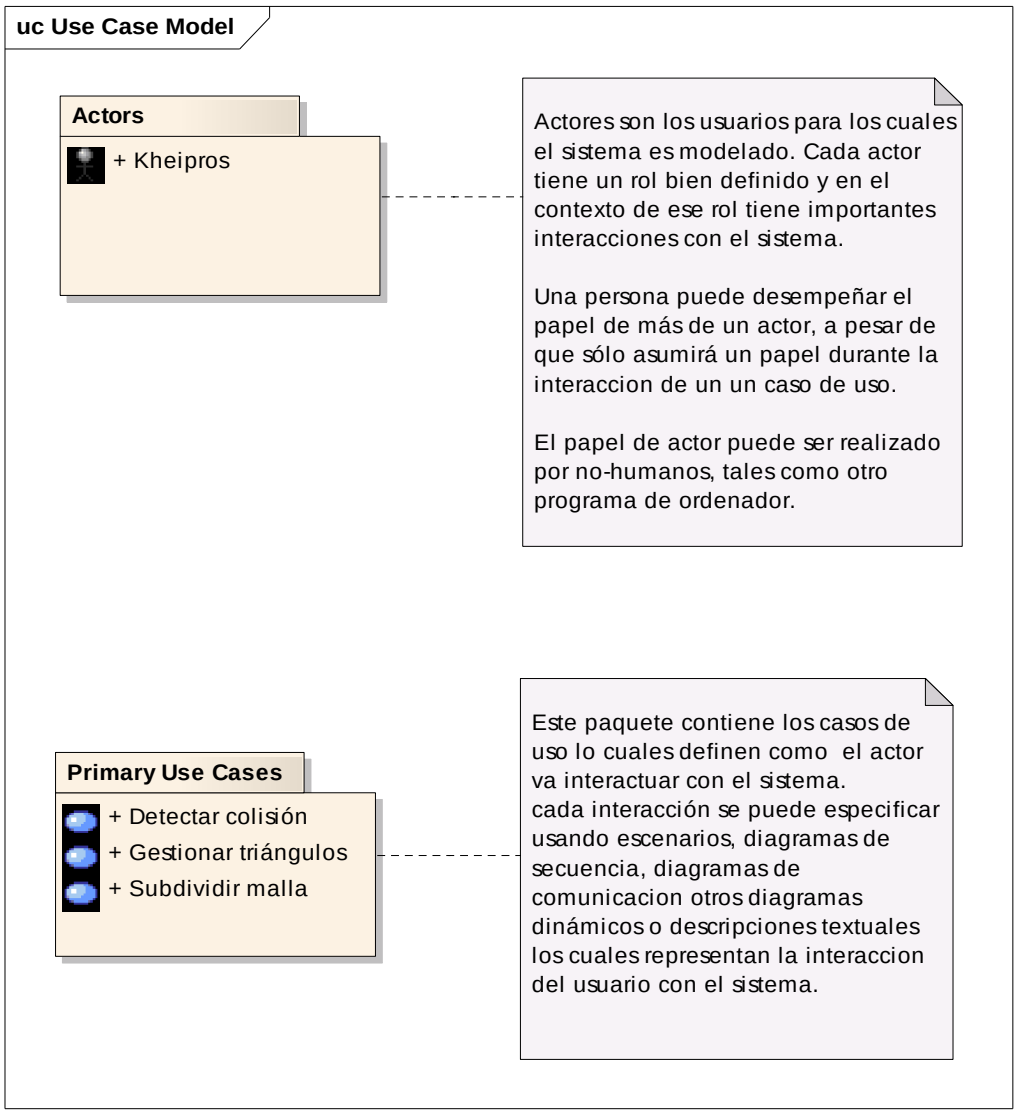

Fig. 26: Modelo de Caso de Uso del Sistema.

3.4.1 Actor del Sistema

Actores	Justificación
Kheipros	Es el simulador quirúrgico quien interactúa con las funcionalidades que presta el sistema

Tabla 1: Actor del Sistema

3.4.2 Casos de Uso del Sistema

CU1	Subdividir malla
Actor	Kheipros
Descripción	Modifica la estructura de la malla generando una abertura.
Referencia	R1.1, R1.2, R1.3, CU2 (include), CU3 (include)

Tabla 2: CU1 Subdividir Malla.

CU2	Detectar colisiones
Actor	CU1
Descripción	Maneja las interacciones de la herramienta de corte y la malla a cortar.
Referencia	R3.1,R3.2

Tabla 3: CU2 Detectar Colisiones.

CU3	Gestionar triángulos
Actor	CU1
Descripción	Inserta, busca y elimina los triángulos de la malla
Referencia	R2.1, R2.2, R2.3, R2.4, R2.5

Tabla 4: CU3 Gestionar triángulos.

3.4.3 Diagrama de Caso de Uso del Sistema

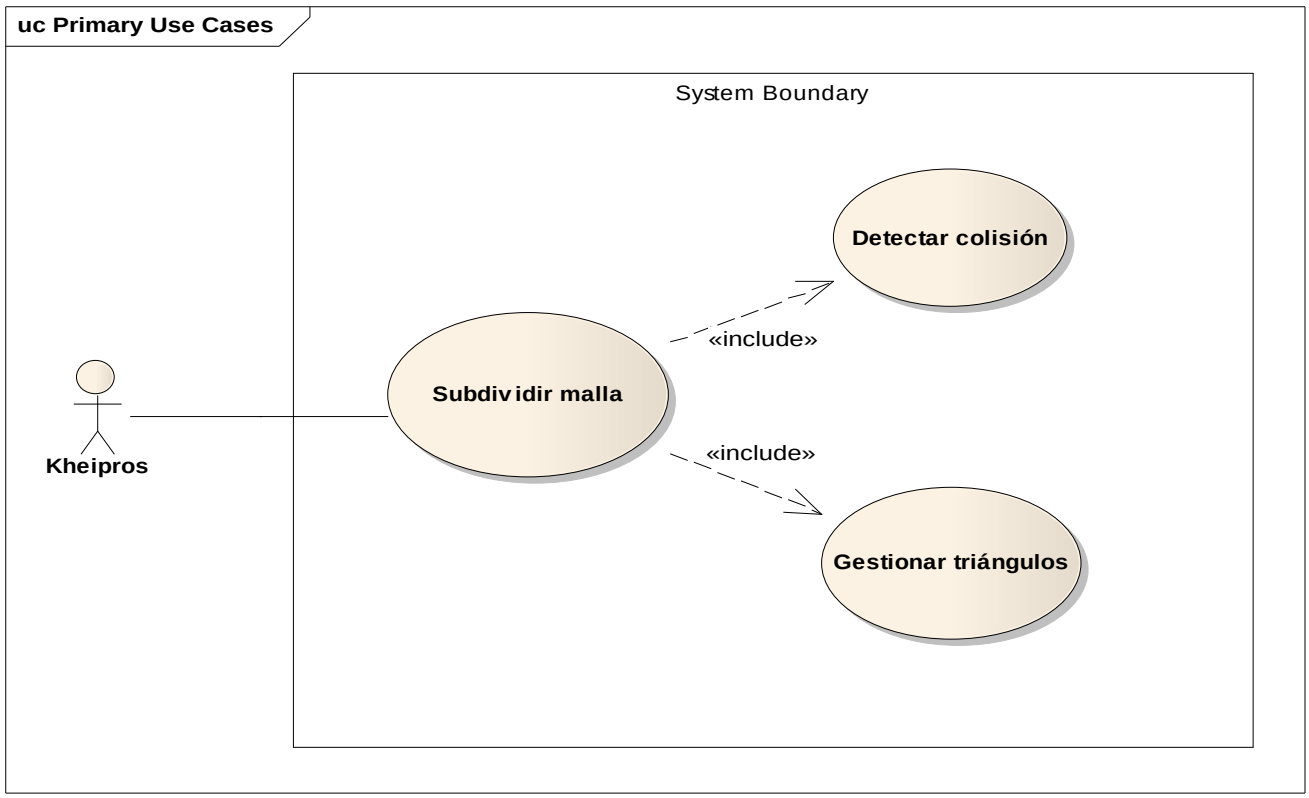

Fig. 27: Diagrama de Caso de Uso del Sistema.

3.3.4 Expansión de Casos de Uso

Caso de uso	
CU1	Subdividir malla
Propósito	Modifica la estructura de la malla generando una abertura
Actor	Kheipros
Resumen: El caso de uso se inicia cuando el actor quiere subdividir la malla. Donde crea una abertura al pasar una herramienta de corte por los triángulos que forman la malla.	
Referencias	R1.1, R1.2, R1.3, CU2 (include), CU3 (include)
Precondiciones	Detectar contacto inicial entre la herramienta y la malla a cortar.
Poscondiciones	Subdivisión de la malla.
Curso Normal de los Eventos	
Acción del actor	Respuesta del sistema
1-El simulador desea subdividir una malla, se realiza movimiento de la herramienta de corte en la malla a subdividir.	1.1- Al detectar contacto con la malla **ver descripción CU2** se adiciona el punto donde hubo el primer contacto.
	1.2- Sigue detectando puntos de contacto, construye la abertura subdividiendo los triángulos en dependencia si son triángulos del inicio ó fin de la subdivisión **ver 1.2.1** o triángulos interceptados en la trayectoria de la herramienta **ver 1.2.2.**
	1.2.1- Los triángulos se modifican creando cuatro nuevos triángulos, por cada triángulo de inicio o fin del la subdivisión de la malla.
	1.2.2- Los triángulos se modifican creando tres nuevos por cada triángulo interceptado en la trayectoria intermedia de la herramienta por la malla.

Prioridad	Crítico

Tabla 5: Expansión CU1 Subdividir malla.

Caso de uso		
CU2	Detectar colisión	
Propósito	Maneja las interacciones de la herramienta de corte y el la malla a cortar.	
Actor	CU1	
Resumen: El caso de uso se inicia cuando la herramienta de corte se mueve en el entorno virtual, el caso de uso se encarga de detectar los puntos de contacto con la malla que se desea subdividir.		
Referencias	R3.1, R3.2	
Precondiciones	Tener la herramienta de corte y la malla a cortar.	
Poscondiciones	Los puntos de contactos al interactuar.	
Curso Normal de los Eventos		
Acción del actor		Respuesta del sistema
1-CU1 manda ejecutar el caso de uso que brinda una información necesaria para la su realización.		1.1-Chequea mientras se mueva la herramienta detectando los puntos de contacto con los triángulos de la malla y los segmentos que conforman los lados de estos.
		1.2-Una vez detectado el primer contacto se mantiene chequeando contacto solo con los triángulos vecinos del triángulo interceptado.
		1.3-Termina el caso de uso cuando se termine el proceso de corte y la herramienta no tenga contacto con la malla.
Prioridad:	Crítico.	

Tabla 6: Expansión CU2 Detectar colisión.

Caso de uso		
CU3	Gestionar triángulos	
Propósito	Se encarga de adicionar y eliminar triángulos de la malla.	
Actor	CU1	
Resumen**:** El caso de uso se inicia cuando el CU1 demanda crear, eliminar de la malla.		
Referencias	R2.1, R2.2, R2.3, R2.4, R2.5	
Precondiciones	Tener una estructura de datos para gestionar los triángulos	
Curso Normal de los Eventos		
Acción del actor		Respuesta del sistema
1-CU1 manda ejecutar el caso de uso que brinda una información necesaria para su realización, modificar la estructura de la malla.		1.1-Permite realizar las siguientes acciones. a) Si decide crear triángulos ver sección “Crear triángulo”. b) Si decide eliminar triángulos ver sección “Eliminar triángulo”.
Sección “Crear triángulo”		
Acción del actor		Respuesta del sistema
2-Se necesita introducir nuevos triángulos en la estructura de la malla que se está subdividiendo.		2.1-Obtiene los vértices para crear un nuevo triángulo.
		2.2-Crea los triángulos y los inserta en la lista de triángulos que gestiona la malla.
		2.3- Se actualizan los vecinos de cada triángulo.
Sección “Eliminar triángulo”		
3-Se necesita eliminar triángulos en la estructura de la malla que se está subdividiendo.		3.1-Obtiene los índices de los triángulos a eliminar.
		3.2- Busca el triángulo por el índice y lo encuentra lo elimina.
Prioridad:	Crítico.	

Tabla 7: Expansión CU3 Gestionar triángulos.

Consideraciones del Capítulo

En el capítulo que concluye con un modelo de dominio para establecer el contexto del sistema, el levantamiento de requisitos funcionales y no funcionales así como también se realiza el modelo de caso de uso mediante descripciones textuales y el diagrama de casos de uso del sistema. Estos resultados reflejan las necesidades cliente además de ser un muy buen punto de partida para el diseño e implementación del sistema.

Capítulo 4

Diseño e Implementación del Sistema

Introducción

En este capítulo se define la estructura del sistema en estudio, siempre basándose en los requisitos funcionales y no funcionales que fueron seleccionados en las etapas anteriores. Se presentan las características de la herramienta Scene Tool Kit imprescindible para el sistema, los artefactos involucrados en el diseño como, Diagramas de Paquetes, Diagramas de Clases y Diagramas de Secuencia para la realización de los casos de uso. La implementación del sistema se describe con diagramas de componentes en los que se definen como se harán físicas las clases de diseño en el lenguaje seleccionado. Se especifican además otros aspectos para la comprensión de los diagramas y se describen en tablas las clases de diseño correspondientes al módulo.

4.1 Características de Scene Tool Kit (STK)

La Herramienta Básica para Desarrollo de Sistemas de Realidad Virtual (SceneToolKit, STK), surge ante la necesidad de concebir herramientas de trabajo propias para los proyectos de Realidad Virtual de la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), que permitieran desarrollar productos con rapidez y calidad, a las que se les pudiera incluir fácilmente nuevas funcionalidades y módulos.

Esta plantea obtener una biblioteca gráfica de tiempo real para Sistemas de Realidad Virtual (SVR) que soporte:

- ✓ El empleo de técnicas para la optimización de la visualización de la escena.
- ✓ La utilización transparente de varias bibliotecas gráficas.
- ✓ La flexibilidad a actualizaciones futuras.
- ✓ Que sea multiplataforma.

Para el almacenamiento de los datos se utiliza un grafo general de escena. El grafo se forma a través de nodos que heredan de una clase Node, y que permiten agrupar otros nodos (GroupNodes, nodos intermedios), o almacenar objetos como geometrías (GeometryNodes), cámaras (CameraNodes) y luces (LightNodes). A cada nodo de la escena se le pueden asociar controladores que rigen el comportamiento del objeto contenido por dicho nodo. Como caso especial se tiene el controlador físico, el cual modifica el cuerpo según sus propiedades físicas y las fuerzas que incidan sobre él en cada instante de tiempo. Esta solución permite que el usuario pueda definir nuevos controladores a través de la herencia y redefinición de métodos virtuales. El proyecto simulador quirúrgico aprovecha las bondades de STK para la visualización de los ejercicios de entrenamiento para médicos. Como el fenómeno de subdividir una malla modifica las listas de triángulos que el renderer utiliza para pintar la escena, este sistema obtiene el CTriMesh de la STK y lo actualiza constantemente. La arquitectura de la herramienta está compuesta por tres capas Application, Engine y Renderer a continuación se especifica la estructura de STK y se presenta algunos de los paquetes del la capa Engine de la cual se obtienen las clases que permiten la modificación de la malla.

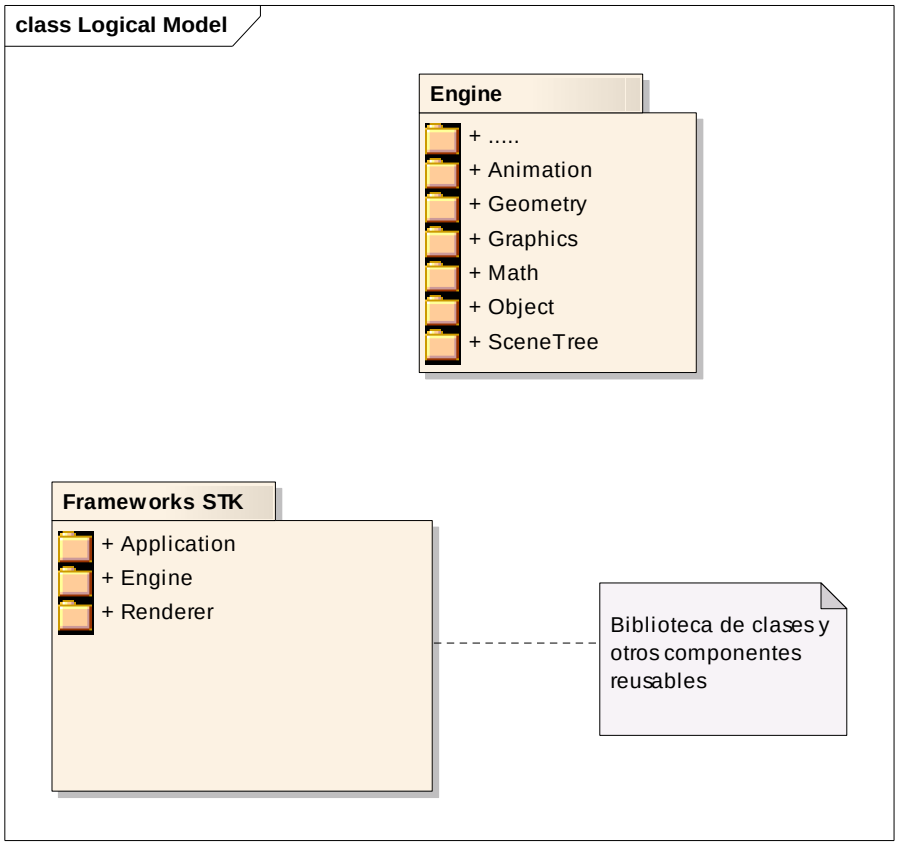

Fig. 28: Arquitectura STK. Ampliación de la Capa Engine

4.2 Diseño del Sistema

En el diagrama de la página siguiente se muestra la relación entre las clases de diseño, nótese que refleja además algunas clases implementadas ya en la Scene Toolkit, pero sin las cuales sería muy complejo entender la arquitectura del sistema y su relación con la STK. En lo adelante solo se desarrollarán los artefactos de nuestro sistema, aunque se mencionen o representen elementos de la STK que son arquitectónicamente significativos.

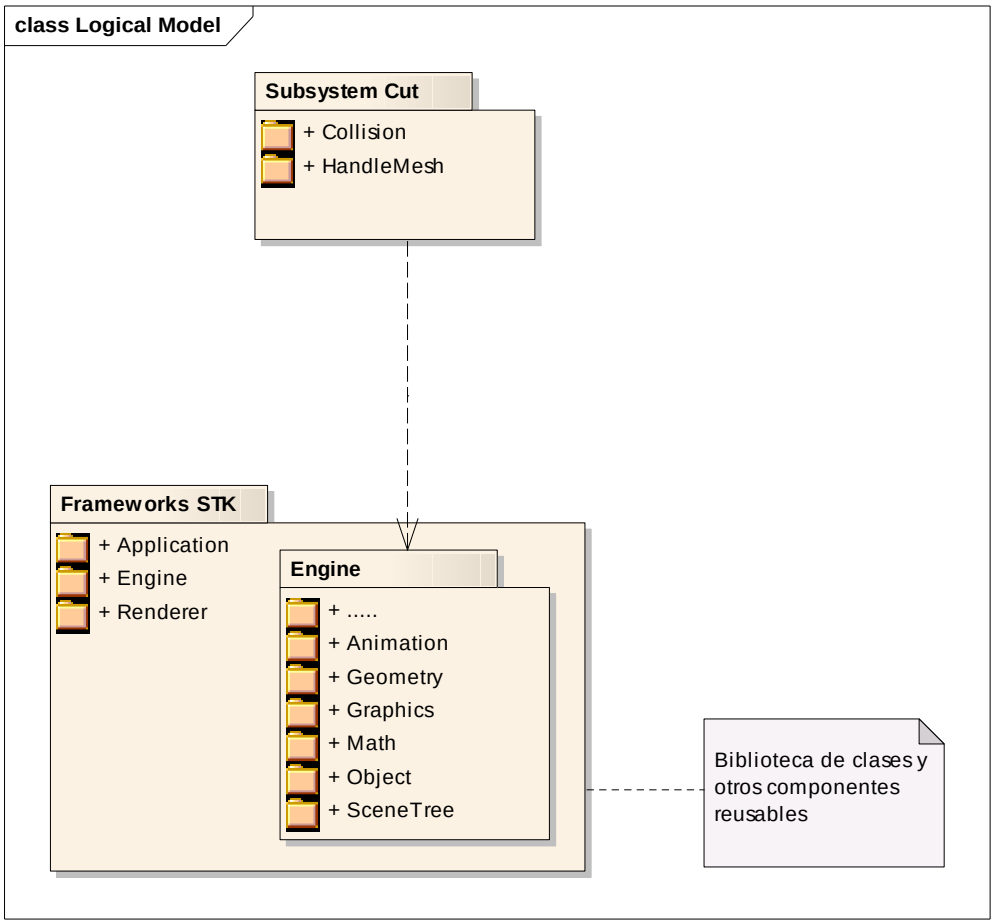

Fig. 29: Modelo Lógico del Sistema

4.3 Diagramas de Paquetes

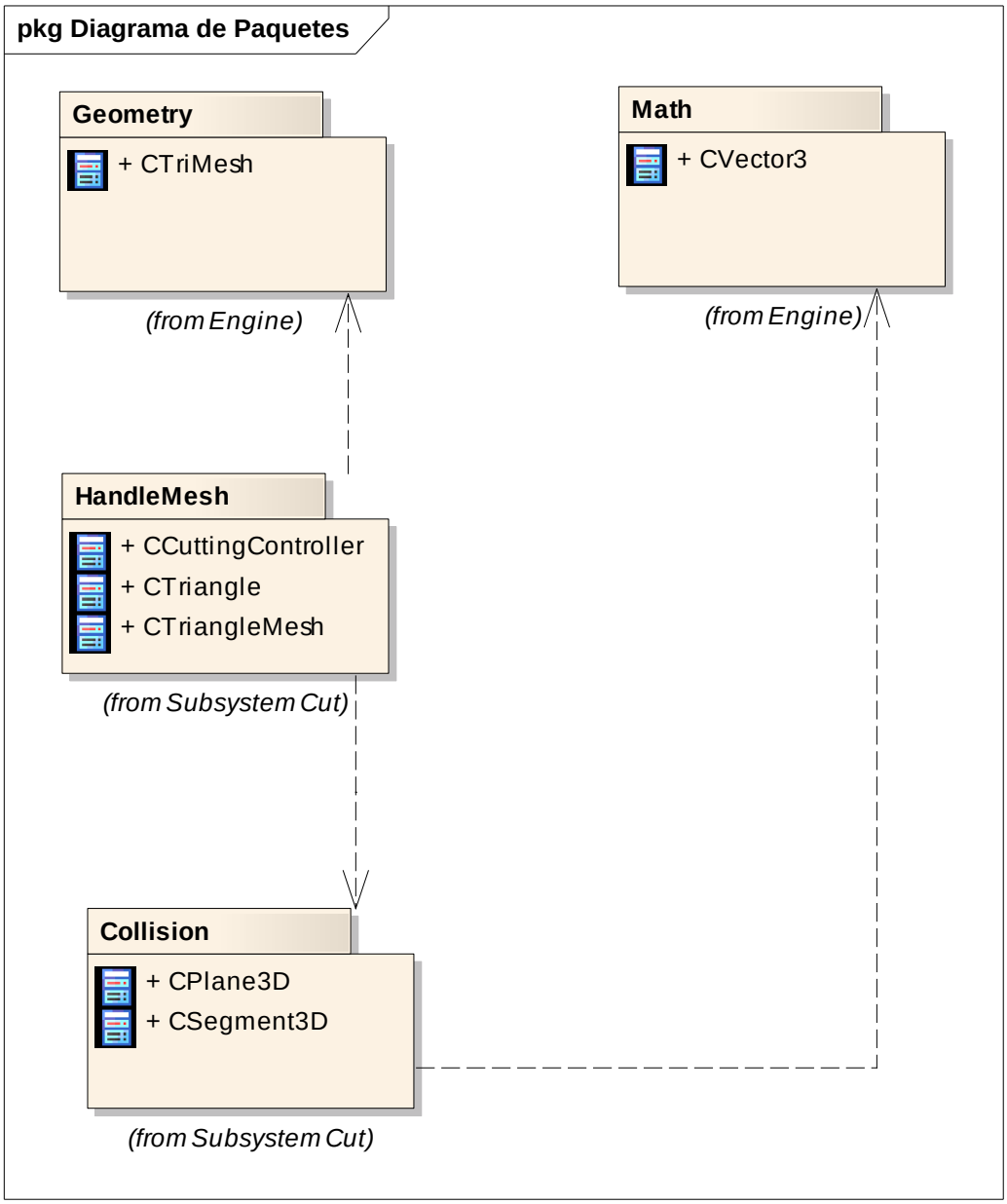

Fig. 30: Diagrama de Paquetes

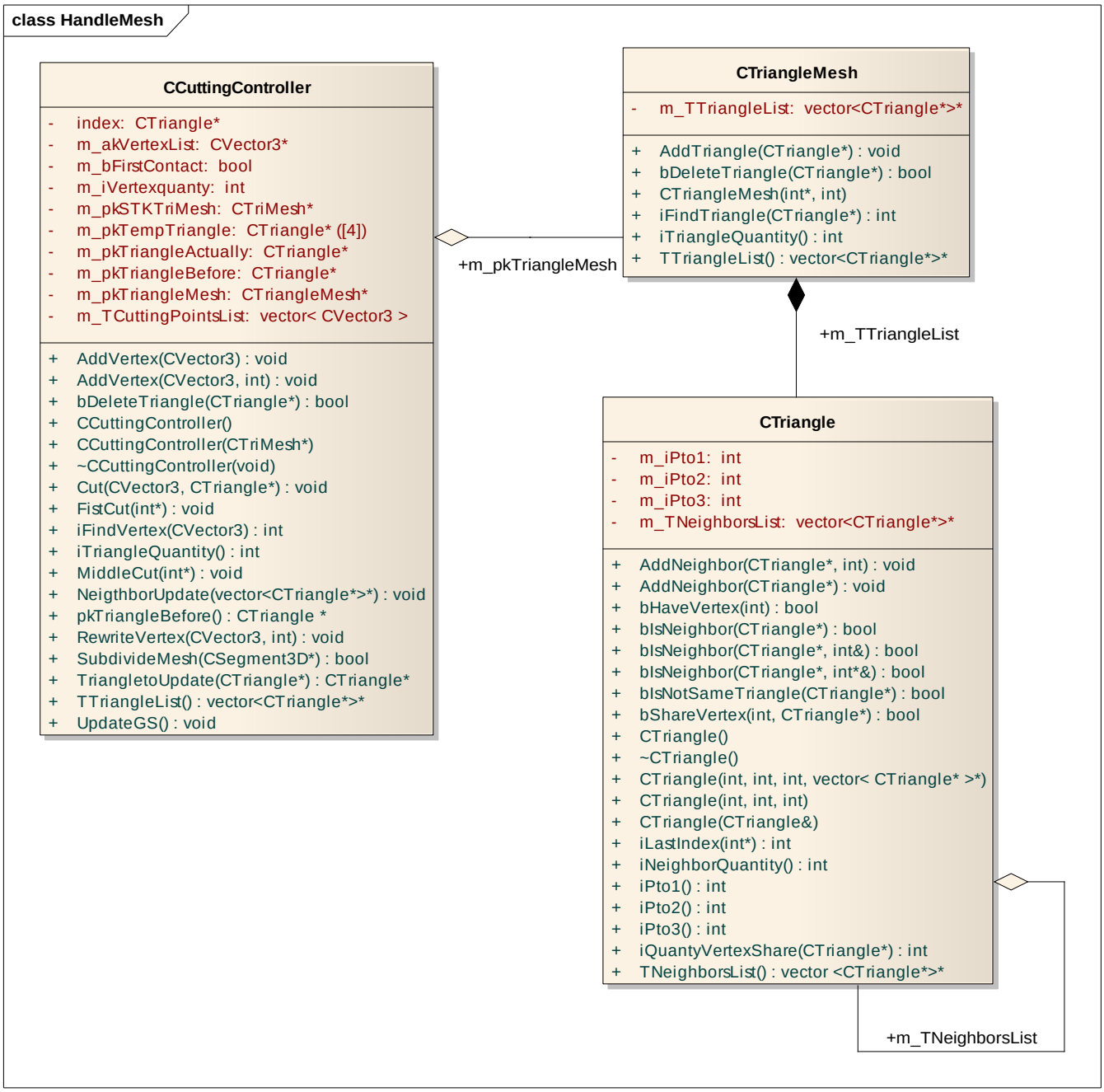

Fig. 31: Diagrama de clases paquete HandleMesh

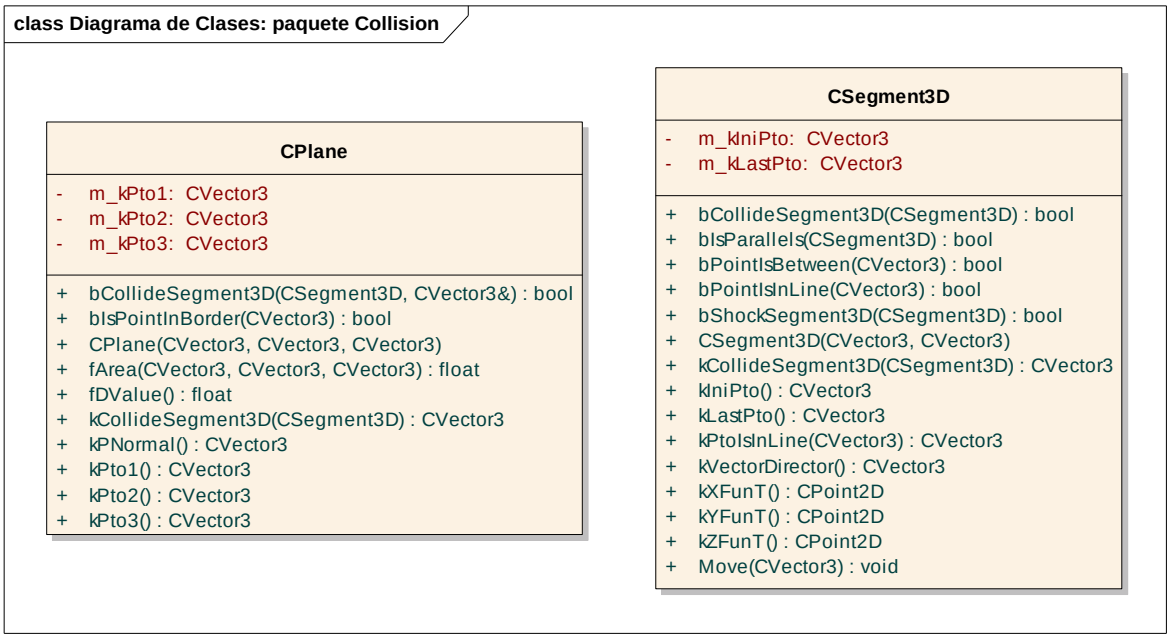

Fig. 32: Diagrama de clases paquete Collision

4.4 Diagrama de Clases General

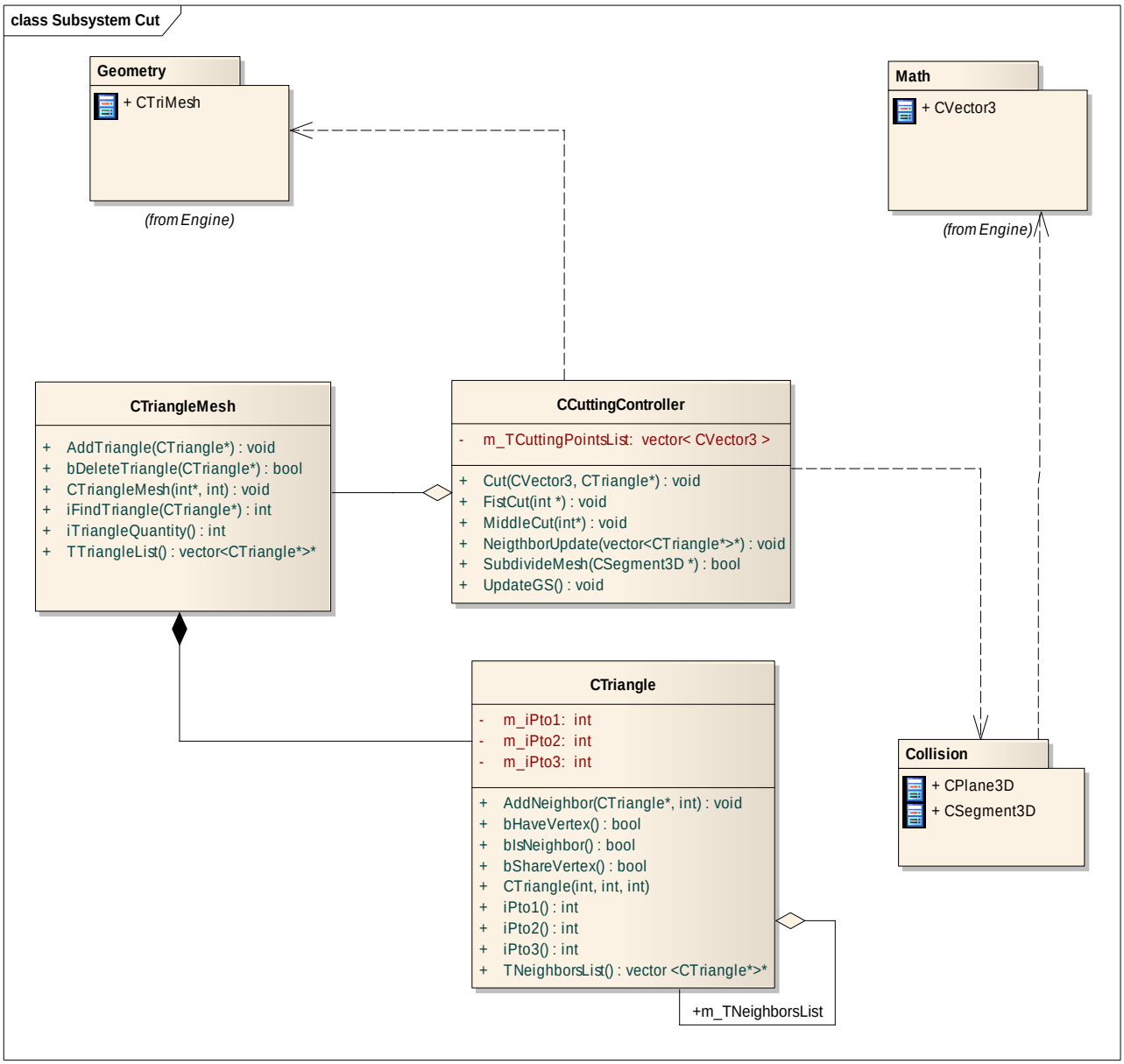

Fig. 33: Diagrama de clases general

4.5 Descripción de las Clases de Diseño

Nombre: CCuttingController	
Tipo de clase: Controladora	
Atributo	Tipo
m_pkSTKTriMesh	CTriMesh*
m_pkTriangleMesh	CTriangleMesh*
m_TCuttingPointsList	vector< CVector3 >
m_pkTriangleBefore	CTriangle*
m_pkTriangleActually	CTriangle*
index	CTriangle*
m_pkTempTriangle[4]	CTriangle*
m_akVertexList	CVector3*
m_iVertexquanty	int
m_bFirstContact	bool
Para cada responsabilidad:	
Nombre:	CCuttingController(CTriMesh* arg_pkSTKTriMesh)
Descripción:	Constructor de la clase se encarga de inicializar el objeto de la clases que manipula las modificaciones en la malla m_pkTriangleMesh.
Nombre:	TTriangleList()
Descripción:	Duevuelve la lista de triángulos de que están en la malla.
Nombre:	iFindVertex(CVector3 kVertex)
Descripción:	Devuelve la posición del vértice en la lista.
Nombre:	pkTriangleBefore ()
Descripción:	Devuelve el triángulo con el cual hubo contacto anteriormente.
Nombre:	AddVertex(CVector3 arg_kVector,int iPos)
Descripción:	Adiciona un vértice a la lista de vértices del TriMesh.
Nombre:	RewriteVertex(CVector3 arg_kVector,int iPos)
Descripción:	Cambia el valor de un vértice en la lista de vértices del TriMesh.
Nombre:	bDeleteTriangle(CTriangle* arg_kTriangle)
Descripción:	Devuelve verdadero o falso si se puede eliminar un triángulo en la lista de

	triángulos que modifica la malla.
Nombre:	iTriangleQuantity()
Descripción:	Devuelve la cantidad de triángulos de la malla que se modifica.
Nombre:	FistCut(int * arg_aiPtos)
Descripción:	Maneja el proceso de corte para los estados inicial y final.
Nombre:	MiddleCut(int* arg_aiPtos)
Descripción:	Maneja el proceso de corte para los estados intermedios.
Nombre:	Cut(CVector3 kVMyVCut, CTriangle* kVMyTriangleCut)
Descripción:	Se encarga de controlar recibiendo los puntos de contacto con la malla cuando hacer estado inicial - final del corte o intermedios.
Nombre:	SubdivideMesh(CSegment3D *arg_pkMySegment)
Descripción:	Maneja la detección de colisiones invocando el método Cut una vez se detecten dos puntos de contacto el método se mantiene chequeando un evento externo.
Nombre:	UpdateGS()
Descripción:	Se responsabiliza de actualizar las listas de vértices y triángulos que luego son usados para pintar la escena.

Tabla 8: Descripción de la clase CCuttingController.

Nombre: CTriangleMesh	
Tipo de clase: Entidad	
Atributo	Tipo
m_TTriangleList	vector<CTriangle*>*
Para cada responsabilidad:	
Nombre:	CTriangleMesh(int* arg_piIndexList,int arg_iTriangleQuantity)
Descripción:	Constructor de la clase se encarga de inicializar los triángulos que contienen la malla para modificarlos, formar la lista de vecinos para cada triángulo de la malla.
Nombre:	iTriangleQuantity()
Descripción:	Devuelve la cantidad de triángulos del la lista.
Nombre:	AddTriangle (CTriangle* arg_kTriangle)

Descripción:	Adiciona un triángulo a la lista.
Nombre:	iFindTriangle (CTriangle* arg_kTriangle)
Descripción:	Busca un triángulo determinado en la lista.
Nombre:	bDeleteTriangle(CTriangle*arg_kTriangle)
Descripción:	Elimina un triángulo en la lista.

Tabla 9: Descripción de la clase CTriangleMesh.

Nombre: CTriangle	
Tipo de clase: Entidad	
Atributo	Tipo
m_iPto1	Int
m_iPto2	Int
m_iPto3	Int
m_TNeighborsList	vector<CTriangle*>*
Para cada responsabilidad:	
Nombre:	CTriangle(int arg_iPto1,int arg_iPto2,int arg_iPto3,vector< CTriangle* > *arg_TNeighbors)
Descripción:	Constructor de la clase se encarga de inicializar los triángulos, contiene la información de los triángulos vecinos del triángulo.
Nombre:	iPto1();
Descripción:	Método de acceso a miembros m_iPto1
Nombre:	iPto2();
Descripción:	Método de acceso a miembros m_iPto2
Nombre:	iPto3()
Descripción:	Método de acceso a miembros m_iPto3
Nombre:	TNeighborsList()
Descripción:	Método de acceso a miembros m_TNeighborsList
Nombre:	iNeighborQuantity()
Descripción:	Devuelve la cantidad de vecinos del triángulo
Nombre:	AddNeighbor(CTriangle* arg_akTriangle,int arg_iPos)

Descripción:	Adiciona un vecino a la lista de vecinos del triángulo
Nombre:	bIsNeighbor(CTriangle* arg_kTriangle)
Descripción:	Devuelve verdadero o falso si dos triángulos son vecinos
Nombre:	bIsNotSameTriangle(CTriangle* arg_kTriangle)
Descripción:	Devuelve verdadero o falso si no es el mismo triángulo
Nombre:	iQuantyVertexShare(CTriangle* arg_kTriangle)
Descripción:	Devuelve la cantidad de vértices que comparten con el triángulos que se le pasa por parámetros
Nombre:	bHaveVertex(int arg_iVertex)
Descripción:	Devuelve verdadero o falso si se encuentra en la posición que se pasa por parámetro como uno de los vértices en la lista de vértices.
Nombre:	bShareVertex(int arg_iVertex,CTriangle* arg_kTriangle)
Descripción:	Devuelve verdadero o falso si dos triángulos tienen un vértice en común
Nombre:	iLastIndex(int* arg_aiIndex)
Descripción:	Devuelve el index del vértice que no tiene contenido en el arreglo que recibe por parámetros

Tabla 10: Descripción de la clase CTriangle.

Nombre: CPlane3D	
Tipo de clase: Controladora	
Atributo	Tipo
m_kPto1	CVector3
m_kPto2	CVector3
m_kPto3	CVector3
Para cada responsabilidad:	
Nombre:	CPlane3D(CVector3 arg_kPto1,CVector3 arg_kPto2,CVector3 arg_kPto3)
Descripción:	Constructor de la clase, inicializa los tres puntos con los que se define un plano.
Nombre:	KPto1()
Descripción:	Método de acceso a miembros m_kPto1

Nombre:	KPto2()
Descripción:	Método de acceso a miembros m_kPto2
Nombre:	Kpto3()
Descripción:	Método de acceso a miembros m_kPto3
Nombre:	kPNormal()
Descripción:	Devuelve el vector perpendicular al plano, la Normal.
Nombre:	fArea(CVector3 arg_Pto1,CVector3 arg_Pto2,CVector3 arg_Pto3)
Descripción:	Determina el área de de un plano.
Nombre:	bCollideSegment3D(CSegment3D arg_kSegment, CVector3& arg_kInterceptionPoint)
Descripción:	Devuelve verdadero o falso si dos segmentos se interceptan, esta función comprueba además que el punto esté dentro del triángulo -plano interceptado-. Utiliza las funciones kCollideSegment3D() la que devuelve el punto exacto de la colisión y fArea() la cual comprueba el sentido de orientación de los triángulos formados con el punto de contacto determinando si está dentro o fuera del triángulo en cuestión.
Nombre:	kCollideSegment3D(CSegment3D arg_kSegment)
Descripción:	Determina el punto exacto de intersección entre dos segmentos.
Nombre:	fDValue()
Descripción:	Es el valor D de la ecuación ax + by + cz = D la cual se utiliza para hallar la intercepción entre un plano y un segmento.

Tabla 11: Descripción de la clase CPlane3D.

Nombre: CSegment3D	
Tipo de clase: Controladora	
Atributo	Tipo
m_kIniPto	CVector3
m_kLastPto	CVector3
Para cada responsabilidad:	
Nombre	CSegment3D(CVector3 arg_kIniPto,CVector3 arg_kLastPto)

Descripción	Constructor de la clase se encarga de inicializar los puntos extremos de un segmento.
Nombre:	kIniPto()
Descripción:	Método de acceso a miembros m_kIniPto
Nombre:	kLastPto()
Descripción:	Método de acceso a miembros m_kLastPto
Nombre:	kVectorDirector()
Descripción:	Devuelve el vector director de un segmento.
Nombre:	kXFunT()
Descripción:	Devuelve el valor de x despejado en función de t para la ecuación paramétrica de la recta
Nombre:	kYFunT()
Descripción:	Devuelve el valor de y despejado en función de t para la ecuación paramétrica de la recta
Nombre:	kZFunT()
Descripción:	Devuelve el valor de z despejado en función de t para la ecuación paramétrica de la recta
Nombre:	bPointIsInLine(CVector3 arg_kVector)
Descripción:	Devuelve verdadero o falso si un punto pertenece a una recta.
Nombre:	bPointIsBetween(CVector3)
Descripción:	Devuelve verdadero o falso si un punto está en un segmento determinado.
Nombre:	bCollideSegment3D(CSegment3D arg_kSegment3D)
Descripción:	Devuelve verdadero o falso si dos segmentos colisionan.
Nombre:	kCollideSegment3D(CSegment3D arg_kSegment3D)
Descripción:	Determina el punto exacto de colisión entre dos segmentos.
Nombre:	Move(CVector3 arg_kVDistance)
Descripción:	Método de acceso a miembros donde se cambia los valores de m_kIniPto y m_kLastPto

Tabla 12: Descripción de la clase CSegment3D.

4.6 Diagramas de Secuencia

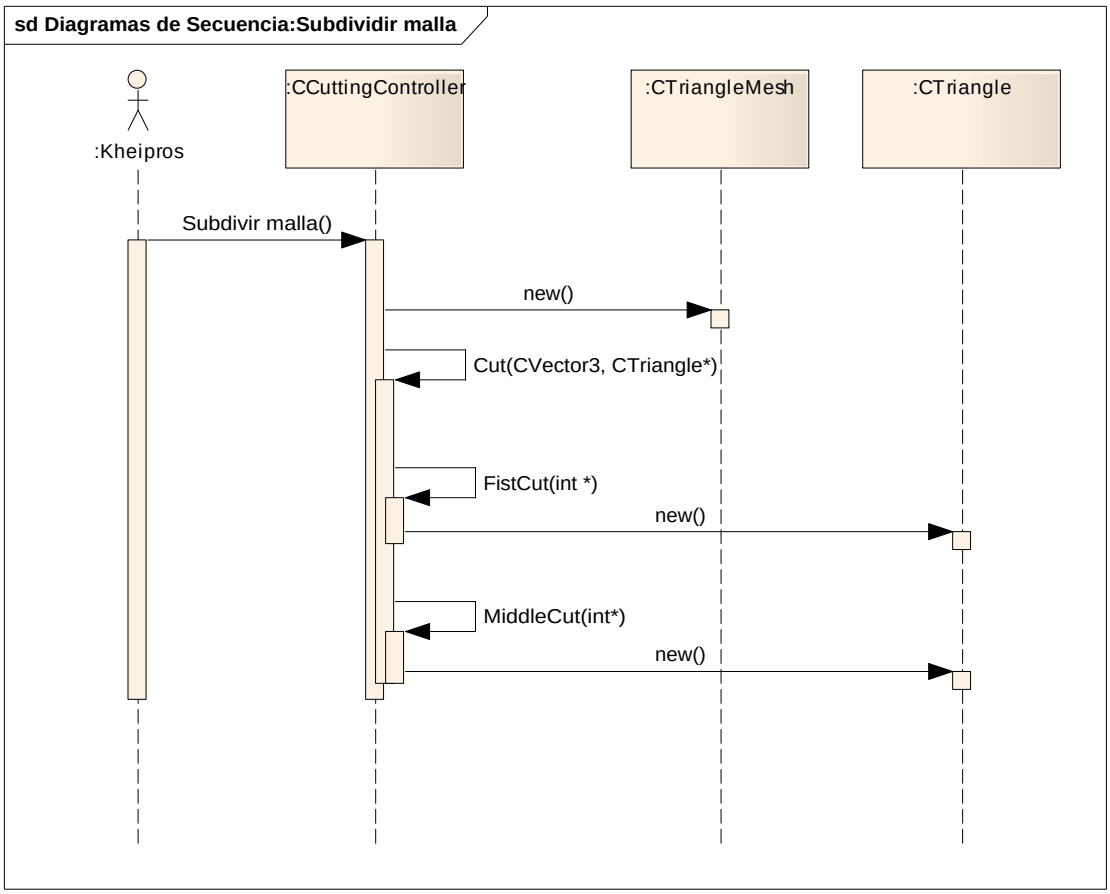

Fig. 34: Diagrama de Secuencia CU Subdividir Malla

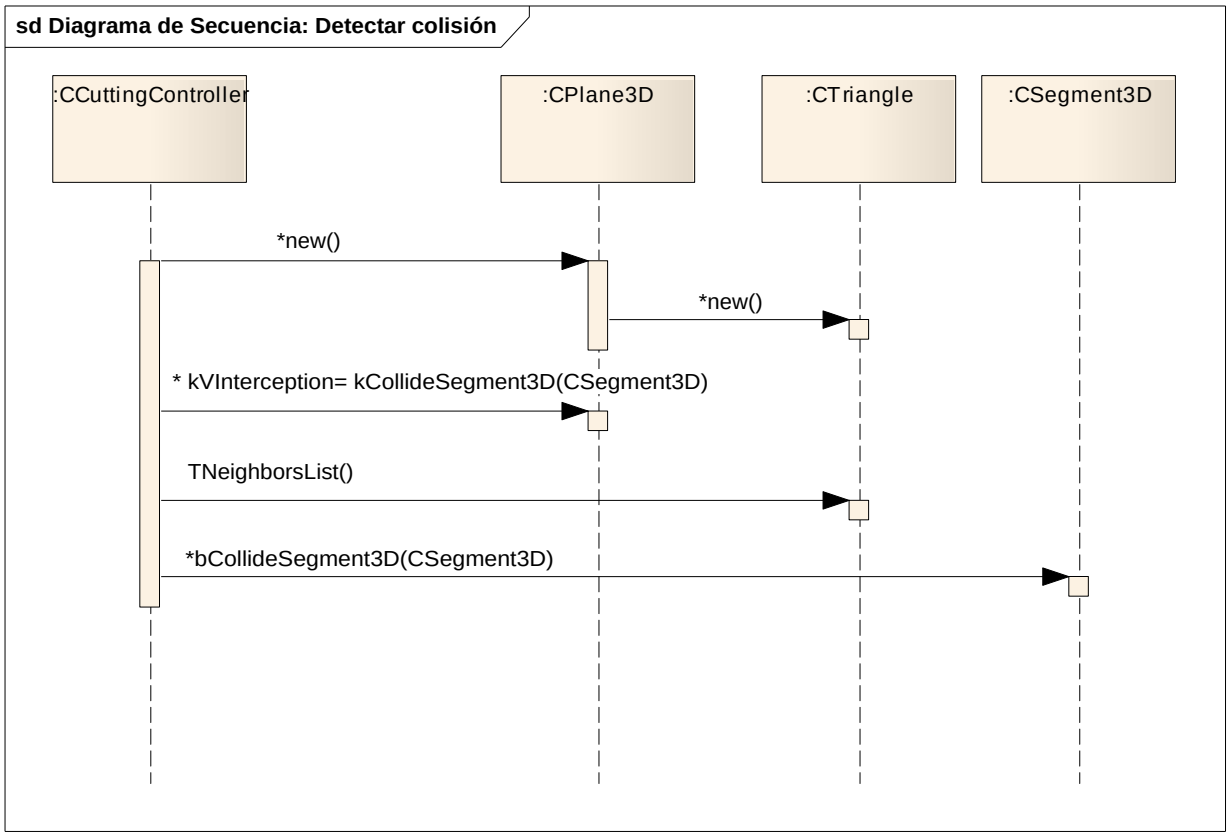

Fig. 35: Diagrama de Secuencia CU Detectar colisión

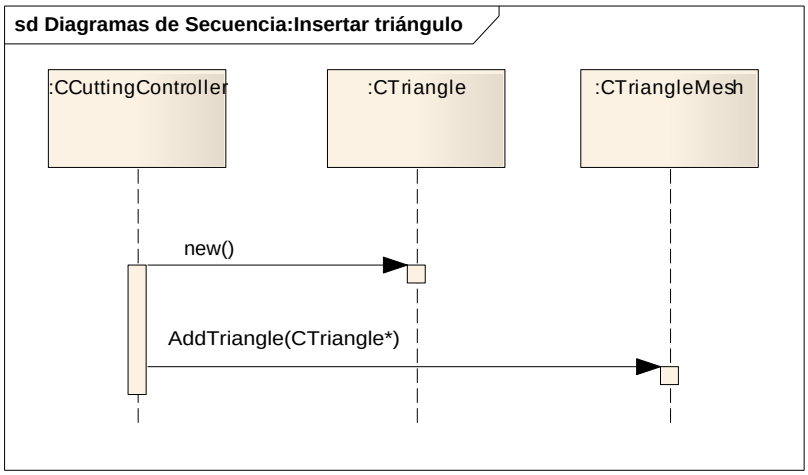

Fig. 36: Diagrama de Secuencia CU Gestionar triángulos, sección insertar.

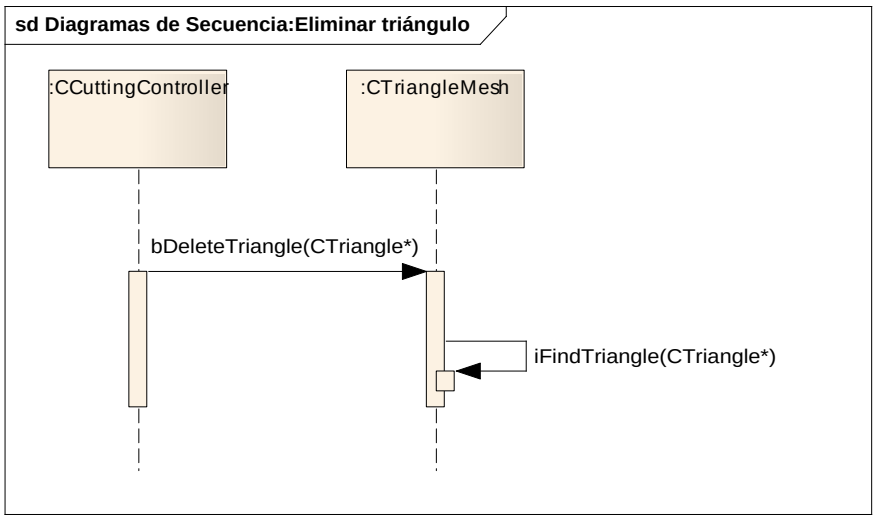

Fig. 37: Diagrama de Secuencia CU Gestionar triángulos, sección eliminar.

4.7 Diagrama de Componentes

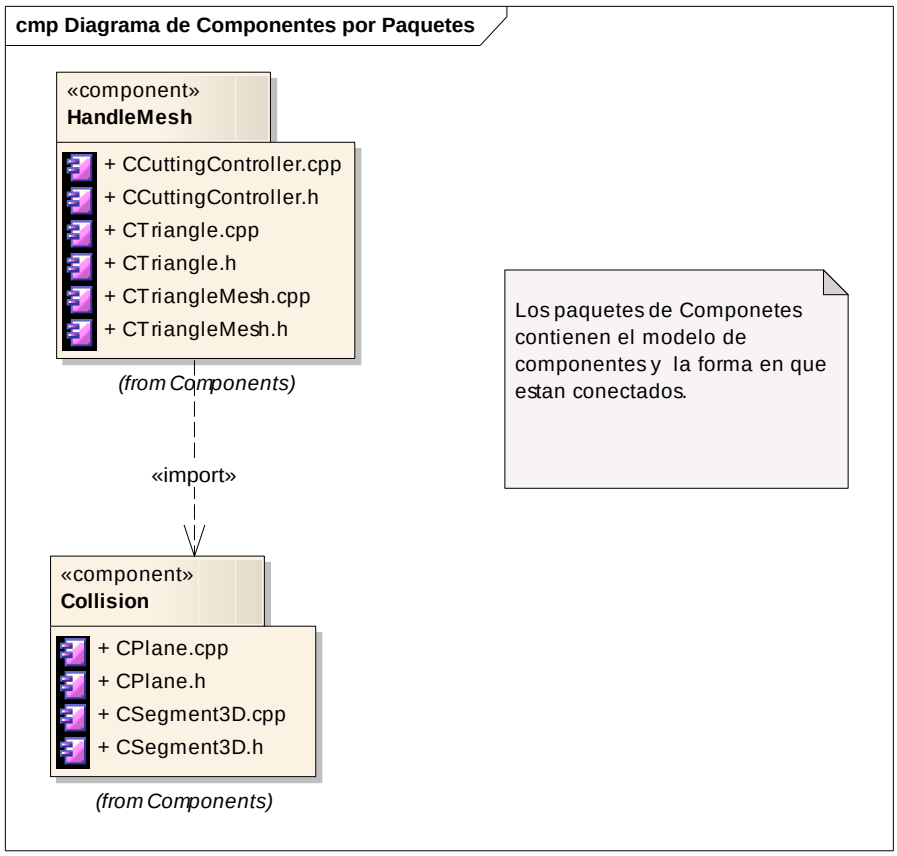

Fig. 38: Diagrama de relación entre paquetes de componentes

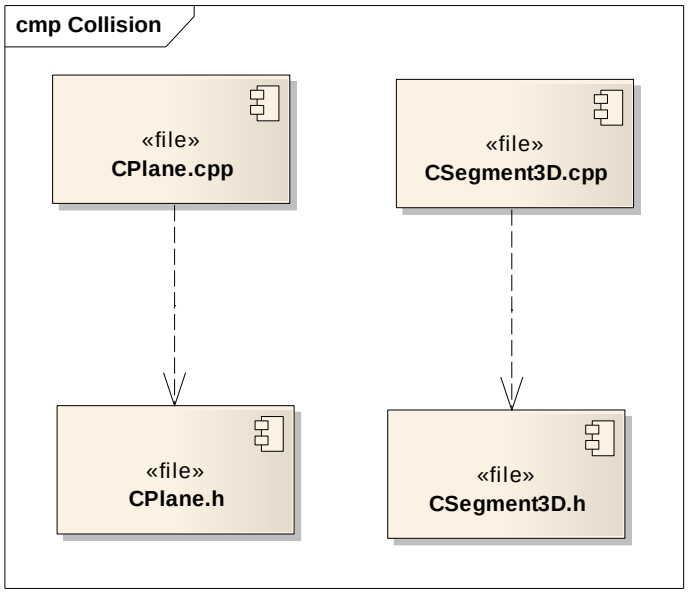

Fig. 39: Diagrama de componentes paquete Collision

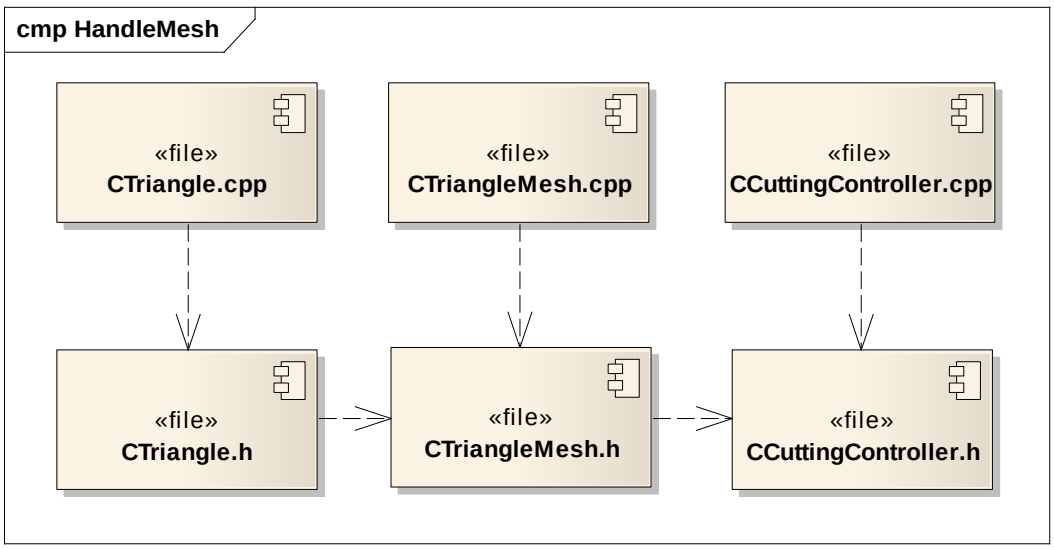

Fig. 40: Diagrama de componentes paquete HandleMesh

Consideraciones del Capítulo

A lo largo del este capítulo se mostraron los artefactos necesarios para el desarrollo del módulo de corte, pertenecientes a las etapas de diseño e implementación. Fundamentalmente se definió una arquitectura compatible con la Scene Tool Kit y la mensajería entre las clases en forma secuencial.

Después de ser concebido y detallado el diseño se pasó a detallar cómo se harán físicas las clases de diseño, consolidando el proceso de desarrollo para pasar a la programación de los casos de uso correspondientes al primer ciclo de desarrollo.

Conclusiones Generales

Para el cumplimiento de los objetivos de este proyecto en función con las necesidades del cliente, fue imprescindible un estudio de las técnicas de corte así como sus ventajas y desventajas en aras de brindar una solución eficiente.

Finalmente se propone una solución de una técnica de corte para modelos superficiales con comportamiento en tiempo real y factible para ser usada en simulación quirúrgica. La solución queda descrita en términos de los flujos de trabajo de requerimientos, diseño e implementación establecidos por RUP.

Recomendaciones

Se recomienda seguir en próximos ciclos de desarrollo del sistema incluir los siguientes aspectos.

- ✓ Generar la ranura interna con el objetivo de representar el interior en una incisión.
- ✓ Tratar degeneraciones con triángulos, mediante algoritmos de triangulación.
- ✓ Permitir separar las geometrías
- ✓ Implementar todas las técnicas de corte.
- ✓ Trabajar el mapeo de texturas, haciendo el cálculo de la coordenada de texturas para cada nuevo vértice a la malla.
- ✓ Integrar la solución con un método deformación basado en Masa - Resorte.

Bibliografía Consultada

1. **Pairó, Joan Trias.** *Geometría para la Informática y CAD.* 2003.

2. **Mendoza C., Sundaraj K., Laugier C.** *Issues in Deformable Virtual Objects Simulation with Force Feedback.* Francia : SHARP Project, Saint Martin, 2002.

3. **Walter Mora F, Geovanni Figueroa M.** Vectores, rectas y planos. *http://www.cidse.itcr.ac.cr/cursos-linea/Algebra-Lineal/algebra-vectorial-geova-walter/node1.html.* [Online]

4. **A Al-khalifah, D Roberts.** *Survey of modelling approaches for medical simulators.* UK : s.n., 2004.

5. **David J. WEISS, Allison M. OKAMURA.** *Haptic Rendering of Tissue Cutting with Scissors.* Newport Beach, CA : s.n., 2004.

6. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* La Habana : Editorial Félix Varela, 2004.

7. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado de Desarrollo de Sofware .* La Habana : Felix Varela, 2004.

Referencias Bibliográficas

[1] Levis Diego, “Realidad Virtual y Educación”

http://www.diegolevis.com.ar/secciones/Articulos/master_eduvirtual.pdf

[2] Downes, M., A. Hsu, and M. Steele, “A virtual environment for training laparoscopic cholecystectomy”, University of California. Berkeley, 1997.

[3] Basdogan, H., S. Srinivasan, and Dawson. “Force Interactions in Laparoscopic Simulations: Haptic Rendering of Soft Tissues”, in Proceedings of the Medicine Meets Virtual Reality Conference, San Diego, 1998.

[4] Cotin, H., et al. “Real Time Volumetric Deformable Models for Surgery Simulation, Visualization in Biomedical” in Computing (Proc. VBC ’96), España: Computer Science, 1996.

[5] Playter, R., B. Blank., and N. Cornelius. “Integrated Haptics Applications: Surgical Anastomosis and Aircraft Maintenance”. in Preprints of The First Phantom User’s Group Workshop. Cambridge, MA.1996.

[6] Gibson, S., J. Samosky, and A. Mor. “Simulating Arthroscopic Knee Surgery Using Volumetric Object Rendering with Real-Time Volume Rendering”. in *CVMRMed-MRCAS’97*. Centro de Investigación MERL Springer-Verlag. 1997.

[7] Corporation, I. “Inmersión médica”. 2006

https://www.immersion.com/medical/

[8] Things, L. “Limbs & Things: skill trining products for healtcare professional”,2006

http://www.limbsandthings.com/uk/index.php

[9] FAQs, C. Medical Education Technologies, Inc. 2006 http://www.meti.com/.

[10] COMPANY, M.S. Medical Simulator. 2006 http://www.medical-simulator.com/index.htm.

[11] Company, S. Simbionix. 2006 http://www.simbionix.com/index.html.

[12] Corporation, S. Simulab Corporation. 2004 http://www.simulab.com/AboutUs.htm.

[13] Science, S. Surgical Science 2006 http://www.surgical-science.com/main/default/default.cfm.

[14] Ayache, N, Cotin, S., Deligette, H, Clement H-M., Marescaux, J., and Nord, M. (1998)

“Simulation of endoscopy surgery”. Jornal of Minimally Invasive Therapy and Allied Technologies (MITAT).

[15]C. Basdogan, Chih - Hao, M.A. Srinivasan, “Simulation of tissue and bleending for laparascopic surgery ausing auxiliary surfaces”, in : Proc. Medicine Meets Virtual Reality, 1999, pp 38 - 44, 1999

[16]C.D. Bruyns, K. Montgomery, “Generalized intersections using virtual tools within the spring framework: Probing, piercing, cuterizing and ablating” in: Proc. Medicine Meets Virtual Reality 2002, pp 74- 78, 2002

[17]M. Bro-Nielsen. “Finite element modeling in medical VR”. Journal of the IEEE, 86(3):490–503, 1998.

[18]S. Cotin, H. Delingette, and N. Ayache. “A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation”. The Visual Computer, 16(8):437–452, 2000.

[19] C. Forest – H. Delingette –N. Ayache,”Cutting simulation of manifold volumetric mehses”, Proceedings Medical Image Computing and Computer Assisted Intervention (MICCAI), 2002

[20]A. Mazura and S. Seifert. “Virtual cutting in medical data”. In Proc. of Medicine Meets Virtual Reality, pages 420–429., 1997.

[21]A. Mor and T. Kanade. Modifying soft tissue models: “Progressive cutting with minimal new element creation”. In MICCAI Proc., pages 598–607, 2000.

[22] D. Bielser et al. “A state machine for real-time cutting of tetrahedral meshes”. In Proc. of Pacific Graphics, pages 377–386, 2003.

[23]D.Servy, M. Harders, G. Szekely. A new approach to cutting into finite element models. In MICCAI'01 Proc, páginas 425 - 433, 2001

[24] w. w. Nienhuys y A. F van der Stappen. “A surgery simulation supporting cuts and finite element deformation”. In MICCAI'01 Proc, páginas 153-160,2001

[25] Zhaosheng Bao “Interactive Simulation of Physically-Based Deformation, Plasticity, and Fracture”, May13th,2003

[26] Aulignac, D. (2001). “Modelisation de l'interaction avec des objets deformables en temps-réel pour des simulateurs medicaux”. PhD thesis, Institute Nationale Polytechnique de Grenoble, France.

[27] Waters, K. (1987).”A muscle model for animating three dimensional facial expression”. In Proceedings of ACM SIGGRAPH.

[28] Terzopoulos, D. and Waters, K. (1990).”Physically-based facial modeling analysis and animation”. Journal of Visualization and Computer Animation.

[29] Baraff, D. and Witkin, A. (1992).“Dynamic simulation of non-penetrating exible bodies”. In Computer Graphics.

[30] Baraff, D. and Witkin, A. (1998).”Large steps in cloth simulation”. In Computer Graphics.

[31] Desbrun, M., Schr• oder, P., and Barr, A. (1999). Interactive animation of structured deformable objects. In Proceedings of Graphics Interface.

[32] Hutchinson, D., Preston, M., and Hewitt, T. (1996).”Adaptive refinement for mass/spring simulation”. In Proceedings of EACG Conference on Eurographics.

[33] Bourguignon, D. and Cani, M. P. (2000).”Controlling anisotropy in mass-spring systems”. In Proceedings of EACG Conference on Eurographics.

[34] Brown, J. and Montgomery, K. (2001).”A microsurgery simulation system”. In Proceedings of Medical Image Computing and Computer Assisted Intervention - MICCAI.

[35] Boux-de-Casson, F. (2000).”Simulation Dynamique de Corps Biologiques et Changements de Topologie Interactifs” (in French). PhD thesis, Universit_e de Savoie, France.

[36] O.C. Zienkiewicz.”The Finite Element Method”. McGraw-Hill, 1977.

[37] K. Bathe.” Finite element procedures”. Prentice Hall, Inc., 1996.

[38] Mendoza Serrano C. “Soft Tissue Interactive Simulations for Medical Applications Including 3D Cutting and Force Feedback” Instituto Nacional Politécnico de Grenoble, Francia. Mayo 2003, 190.

[39] Mendoza C., Sundaraj K., Laugier C. “Issues in Deformable Virtual Objects Simulation with Force Feedback” SHARP Project, Saint Martin, Francia 2002.

[40] Dimaio, S. P. and Salcudean, S. E. (2002).” Simulated interactive needle insertion”. In Proceedings of International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator System.

[41] Mendoza, C. and Laugier, C. (2003). “Simulating soft tissue cutting using finite element models”. In Proceedings of IEEE International Conference on Robotics and Automation.

[42] Kenneth, Sundaraj “Real-TimeDynamic Simulation and 3D Interaction of Biological Tissue : Application to Medical Simulators”. 2004. Phd Thesis.

[43] James, D. and Pai, D. (1999). “Accurate real-time deformable objects”. In Proceedings of ACM SIGGRAPH.

[44] Andrew N, Matthias M, Richard K, Eddy B, Mark C “Physically Based Deformable Models in Computer Graphics”

[45] Nealen A, Müller M, Keiser R, Boxerman E and Carlson M “Physically Based Deformable Models in Computer Graphics” EUROGRAPHICS 2005.

[46] Costa, I. F. and Balaniuk, R. (2001b).”Static solution for real time deformable objects with uid inside”. In ERCIM News.

[47] S. Cotin. “Modéles anatomiques déformables en temps-réel”. Thése de doctorat, INRIA Sophia Antipolis – Université de Nice, Sophia Antipolis, 1997.

[48] D. Bielser, V. A. Maiwald, and M. H. Gross. “Interactive cuts through 3-dimentional soft tissue”. In EUROGRAPHICS’99, volume 18, pages C31–C38, 1999.

[49] H. Nienhuys and F. Vanderstappen.”Combining finite element deformation with cutting for surgery simulations”. In *EUROGRAPHICS 2000*, 2000

[50] C. Mendoza - C. Laugier - O. Galizzi - F. Faure, “Simulating Cutting in Surgery Applications using Haptics and Finite Element Models”, Proceedings MICCAI, 2003

[51] Hui Zhang, Shahram Payandeh and John Dill,” Simulation of Progressive Cutting on Surface Mesh Model”, Robotics and Computer Graphics Laboratories,School of Engineering Science Simon Fraser University Burnaby, BC V5A 1S6, Canada,2002

[52] P. Meseure et C. Chaillou, "Deformable Body simulation with Adaptive Subdivision and Cuttings", Proceedings of the WSCG'97

[53] Garcia-Alonso, A., Serrano, N., and Flaquer, J. (1994). Solving the collision detection problem . In IEEE Transactions on Computer Graphics and Applications.

[54] Van der Bergen, G. (1997). Effcient collision detection of complex deformable models using AABB trees . Journal of Graphic Tools.

[55] Gottschalk, S., Lin, M. C., and Manocha,D. (1996). OBB-Tree: A hierarchical structure for rapid interference detection .In Proceedings of ACM SIGGRAPH.

[56] Hubbard, P. M. (1996). Approximating polyhedra with spheres for time-critical collision detection . In ACM Transactions on Graphics.

Anexos 1: Estándares de Codificación

El código de la biblioteca “Scene Toolkit” sigue algunos estándares propuestos por el grupo de desarrollo, respetando los estándares de codificación para C++ (indexado, uso de espacios y líneas en blanco, etc.). Está programado en inglés, debido que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático.

El conocimiento de los estándares seguidos para el desarrollo de la misma permitirá un mayor entendimiento del código, y es una exigencia de los autores de la misma que cualquier módulo que se añada debe estar codificado siguiendo estos estándares.

Nombre de los ficheros:

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

STKNameOfUnits.cpp

Constantes:

Las constantes se nombrarán con mayúsculas, utilizándose el “_” para separar las palabras: MY_CONST_ZERO = 0;

Tipos de datos:

Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados:

enum EMyEnum {ME_VALUE, ME_OTHER_VALUE};

Indicando con “E” que es de tipo enumerado. Nótese que las primeras letras de las constantes de enumerados son las iniciales del nombre del enumerado. Véase otro ejemplo:

enum ENodeType {NT_GEOMETRYNODE,…};

Estructuras: struct SMyStruct {…};

Indicando con “S” que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

Clases: class CClassName;

Indicando con “C” que es una clase

Interfaces: IMyInterface

Indicando con “I” que es una interfaz.

Listas e iteradotes STD: vector<> TNameList;

TNameList::iterator TNameListIter;

map<> TNameMap;

TNameMap::iterator TNameMapIter;

multimap<> TNameMultiMap;

TNameMultiMap::iterator TNameMultiMapIter;

Anexos 2: Declaración de Variables

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepondrá el identificador “” (en minúscula), si son globales se les antepondrá la letra “g”, y en caso de ser argumentos de algún método, se les antepondrá el prefijo “arg_”.

Tipos simples:

bool bVarName;

int iName;

unsigned int uiName;

float fName;

char cName;

char* acName; // arreglo de caracteres

char* pcName; // puntero a un char

char** aacName; // bidimensional

char** apcName; // arreglo de punteros

bool m_bMemberVarName; //variable miembro

char gcGlobalVarName; //variable global, no se le antepone””.

short sName;

Instancias de tipos creados:

EMyEnumerated eName;

SMyStructure kName;

CClassName kObjectName;

CClassName* pkName; //puntero a objeto

CClassName* akName; //arreglo de objetos

CClassName* akName; // variable miembro de clase

IMyInterface* pIName; //puntero interfaces

Métodos: En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada. Solamente los constructores y destructores comenzarán con “”. En el caso de los argumentos se les antepone el prefijo “arg_”

Constructor y destructor:

CClassName (bool arg_bVarName, float& arg_fVarName);

~CClassName ();

Funciones:

bool bFunction1 (…);

int* piFunction2 (…);

CClassName* pkFunction3 (…);

Procedimientos:

void Procedure4 (…);

Métodos de acceso a miembros: Los métodos de acceso a los miembros de las clases no se nombrarán “Gets” y “Sets”, sino como los demás métodos, pero con el nombre de la variable a la que se accede y sin “m_”:

int iMyVar; //variable

Obtención del valor:

int iMyVar();

{

```
  return iMyVar;
}
```

Establecimiento del valor:

```
void MyVar(char* arg_iMyVar)
{
  iMyVar = arg_iMyVar;
}
```

Obtención y establecimiento del valor:

```
int& iMyVar();
{
   return iMyVar;
}
```

Anexo 3: Glosario de Términos

B

Biblioteca: En Inglés *library,* en términos informáticos, refiere al conjunto de rutinas que realizan las operaciones usualmente requeridas por los programas. Las librerías pueden ser compartidas, lo que quiere decir que las rutinas de la librería residen en un fichero distinto de los programas que las utilizan. Los programas enlazados con bibliotecas compartidas no funcionarán a menos que se instalen las bibliotecas o librerías necesarias.

Biomecánica: Es una disciplina científica que tiene por objeto el estudio de las estructuras de carácter mecánico que existen en los seres vivos (fundamentalmente del cuerpo humano). Esta área de conocimiento se apoya en diversas ciencias biomédicas, utilizando los conocimientos de la mecánica, la ingeniería, la anatomía, la fisiología y otras disciplinas.

C

Cirugía de Mínimo Acceso: La cirugía laparoscópica, sin ingreso o mínimamente invasiva es una técnica quirúrgica practicada a través de pequeñas incisiones, asistida de una cámara de video que permite cirujano accionar sobre el campo quirúrgico, evitando los grandes cortes de bisturí requeridos por la cirugía abierta o convencional y posibilita un periodo post-operatorio mucho más rápido y confortable.

C++: Lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos (POO). C++ está considerado por muchos como uno de los lenguajes más potentes, debido a que permite trabajar tanto a alto como a bajo nivel.

E

Elasticidad: Propiedad mecánica de ciertos materiales de sufrir deformaciones reversibles cuando se encuentra sujetos a la acción de fuerzas exteriores y de recuperar la forma original si estas fuerzas exteriores se eliminan.

3D Engines: Herramientas que facilitan la visualización de escenas 3D en ordenadores.

H

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

L

Laparoscopia: Es una técnica de endoscopia que permite la visión de la cavidad pélvica-abdominal con la ayuda de un tubo óptico

M

Multiplataforma: Término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

O

Objeto Deformable: Cuerpo que por sus características físicas están expuestos a cambios en su forma debido a la acción de agentes externos.

P

Plasticidad: Propiedad mecánica de un material de deformarse permanente e irreversiblemente cuando se encuentra sometido a tensiones por encima de su rango de elasticidad.

R

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

Render: Proceso de obtención de imágenes por computadora.

S

Simulación: Intento de recrear el comportamiento real de sistemas a través de modelos aproximados.

Sistemas de Realidad Virtual: Sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real.

T

Tiempo Real: Término usado en el mundo de los gráficos por computadora para las aplicaciones interactivas con respuesta en intervalos de tiempo que parecen instantáneos al usuario, usualmente más de 16 fps.

V

Virtual: Término utilizado para hacer referencia a algo que no tiene existencia física o real, sólo aparente.

Índice de Figuras

Índice de Tablas