Universidad de las Ciencias Informáticas

Facultad 5

Entornos Virtuales

"MÓDULO DE DEFORMACIÓN BASADO EN EL ALGORITMO CHAINMAIL"

Trabajo para optar por el Título de Ingeniería en Ciencias Informáticas

Autores:

Yerandi Marcheco Díaz.

Yausell Ruiz Marine.

Tutores:

Ing. Raissel Ramirez Orozco.

Ing. Osley Bretau Camejo.

Ciudad de la Habana

Junio 2008

“Saber no es suficiente; tenemos que aplicarlo. Tener voluntad no es suficiente: tenemos que implementarla”

Johann Wolfgang von Goethe.

Declaración de Autoría.

Declaramos que somos los únicos autores de este trabajo y autorizamos al Proyecto “Herramientas de Desarrollo para Sistemas de Realidad Virtual”, 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:

Yerandi Marcheco Díaz

_____

Yausell Ruiz Marine

_____

Tutores:

Ing. Raissel Ramirez Orozco

_____

Ing. Osley Bretau Camejo

_____

Datos de Contacto.

Nombre y Apellidos: Raissel Ramírez Orozco.
Edad: 24 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.

Nombre y Apellidos: Osley Bretau Camejo.
Edad: 24 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.

Dedicatoria.

A mis padres por ser mi inspiración, mi soporte y mis guías.
A mi novia Eli.
A mi familia, en especial a mis hermanas Ivi y Lari, a Dailen, a mi tia Paula, a Lachi, a Lázaro, a Anel y su familia, a Raque y Dulce, a mis hermanos Yanier, Rolandito y Vitico.
A mi nueva familia, Magdia y Juan Miguel.
A mis suegros.
A todos ellos les dedico mi logro.
Yausell.

A mis padres por ser fuente inagotable de apoyo.
A mis hermanos por cuidarme y a apoyarme en todo.
A mi familia en general por todo su apoyo a lo largo de toda mi vida.
A mis amigos, a mis compañeros a todos ellos les dedico este momento.
Yerandi.

Agradecimientos.

Las personas que han contribuido a que esta encomienda sea una realidad han sido muchas, quizás no todos sus nombres queden plasmados en esta hoja, algunos de ellos son Dania, Leo, Lester y Juan Manuel gracias por brindarnos sus conocimientos. A las muchachas y muchachos del grupo, a los del proyecto que de alguna forma contribuyeron, a todos los profesores que nos inculcaron conocimientos y valores, y sobre todo gracias a Nuestro Comandante y esta gran Revolución por confiar en nosotros y darnos la oportunidad de realizar nuestros sueños de convertirnos en profesionales capaces.

Especial Agradecimiento: *A Raissel y Osley por su guía, paciencia, apoyo, dedicación y enseñanza en todo el transcurso de este trabajo y por también ser nuestros amigos.*

Yerandi y Yausell.

De Yausell.

A Yerandi por ser mi hermano y amigo incondicional.

A mi novia Elizabeth por siempre estar para mí y ser mi principal apoyo.

A mis padres por darme amor y fortaleza para avanzar en la vida.

A mis hermanas por ser como son conmigo, a mis hermanos por su apoyo, a mi familia de Alquízar por estar para mí.

A Magdia a Juan Miguel por cuidar de los míos mientras estoy lejos de casa.

Y a todos los que no mencione y que merecen mis agradecimientos.

Gracias.

De Yerandi

A mis padres Rolando y Esperanza de todo corazón, por confiar en mí, apoyarme en todo momento y ser ejemplos para mí.

A mis hermanos Yeylin y Yaini por ser mis segundos padres, por enseñarme que no hay nada imposible y ser ejemplos para mí.

A mi tío y mi tía Osvaldo y Juana por su apoyo en todo momento.

A Yausell que más que compañero de tesis ha sido mi hermano estos 5 años, y creo que sin él sería imposible realizar una tesis de este tipo.

A Lisandra por todo su cariño y apoyo estos 5 años.

A Eli por darnos apoyo, y guiar a mi compañero.

A mis compañeros de aula por ser mi segunda familia aquí en la UCI.

A mis tutores que más que tutores fueron amigos.

A esa persona tan especial; que no puedo mencionar su nombre, por su apoyo y su cariño.

A Dios por acompañarme todos estos años de estudios.

A todos mis amigos que de una u otra forma contribuyeron a la realización de esta Tesis.

Resumen.

En los últimos años los sistemas de realidad virtual se han convertido en una parte importante en el entrenamiento de las más diversas actividades humanas. Sin embargo, las limitaciones del hardware representan un obstáculo para el desarrollo de aplicaciones complejas, como es el caso de la simulación quirúrgica. El objetivo de este trabajo es brindar una solución al problema de la simulación de objetos deformables, prestando especial atención a la simulación de órganos.

Se ha desarrollado un estudio para conocer el estado del arte a nivel mundial de las principales técnicas usadas en la deformación de objetos, tanto geométricas; como son Splines, Curvas Bezier, la Deformación de Libre Forma y el Algoritmo 3D ChainMail. Así como métodos basados en física, entre los que se encuentran, el Método de Elementos Finitos, Método de Elementos de Frontera y Sistemas Masa-Resorte.

En este documento se propone una solución con el desarrollo de un módulo de software basado en la técnica 3D ChainMail, que será acoplado a la Scene Toolkit (STK) y desarrollado usando el proceso unificado de software e implementado en C++ estándar.

Palabras Claves.

Algoritmo, 3D ChainMail, Simulación.

Summary.

In the last years the systems of virtual reality have become an important part in the training of the diverse human activities. Nevertheless, the limitations of hardware represent an obstacle for the development of complex applications, as it is the case of the surgical simulation. The objective of this work is to offer a solution to the problem of the simulation of deformable objects, paying special attention to the organs simulation.

A study has been developed to know the state of the art at world level used in the deformation of objects, Studying geometric techniques as Splines, Bezier Curves, the Free Form Deformation and the 3D ChainMail Algorithm. As well as methods based on physics, like Finite Elements Method, Boundary Elements Method and Mass-Springs Systems. In this document a solution with the development of a module of software based on the technique sets out 3D ChainMail, that will be connected to Scene Toolkit (STK) and developed using the Rational Unified Process and implemented in standard C++.

Keywords.

Algorithm, 3D ChainMail, Simulation.

Índice:

Introducción.

Hasta la actualidad las técnicas de Realidad Virtual han progresado alcanzando un avance aún mayor que la habilidad siquiera para imaginar cómo utilizarlas. Hoy el mundo de la Realidad Virtual es tan amplio que va desde la representación de personajes en videos y películas hasta la representación de los procesos más complejos mediante simuladores.

El desarrollo de estos simuladores es cada vez más acelerado, así como su alto poder de representación de la realidad; ellos han traído numerosas ventajas a profesionales de varios sectores; los pilotos aprenden a volar en tierra, los arquitectos diseñan y muestran a los clientes sus edificios sobre maquetas virtuales, se simula el funcionamiento de átomos en investigaciones, los militares aprenden a utilizar sus carros de combate y armas, así lo muestran las últimas investigaciones [56]. Sin embargo los simuladores también tienen un especial uso en la medicina, principalmente en la cirugía. El uso de la simulación quirúrgica ha permitido el entrenamiento de cirujanos en determinadas técnicas quirúrgicas, fundamentalmente en las de cirugías de mínimo acceso.

El entrenamiento habitual de los cirujanos en las técnicas quirúrgicas se realiza utilizando cadáveres, ya sea de personas o de animales; lo que trae consigo una serie de dificultades, incluyendo el alto costo que representa [2], estas se han ido eliminando con el desarrollo de los Simuladores Quirúrgicos. De ahí la importancia de los mismos y de su capacidad de representar la realidad. Lograr representar objetos deformables en un entorno virtual ante la influencia de diversos agentes es un paso primario para el futuro desarrollo de simuladores de cirugía que permitan representar deformaciones sobre los órganos, cortes, cauterizaciones, etc.

La **situación problémica** está dada por el requerimiento de un módulo que posibilite la escenificación en tiempo real de elementos deformables y su respuesta ante factores deformadores de una forma más rápida que el algoritmo existente.

En la actualidad se hace un poco difícil la representación real de las deformaciones de los órganos debido a las limitaciones de hardware existentes, o a la falta de algoritmos eficientes. Pero gracias a las grandes ventajas que aportan los simuladores quirúrgicos al entrenamiento de cirujanos son varios los grupos de investigaciones que le dedican un gran esfuerzo a la producción de estos. Existen algunos países que ya han tenido resultados positivos en estas investigaciones, tal es el caso de España con el InsightMIST [3] y Francia con el Hepatic Surgery Simulator desarrollado por S. Cotin [4] (Fig. 1). Muchos son también los centros que se dedican a la investigación y desarrollo de los simuladores como es el Laparoscopic Cholecystectomy desarrollado por Michael Downes en la Universidad de California [5], el Laparoscopic Simulator desarrollado por Srinivasan en el Laboratory for Human and Machina Haptics de la Universidad de Missouri, Columbia; el Anastomosis Simulator desarrollado por Boston Dynamics Inc [6], y el Arthroscopic Knee Surgery Simulator desarrollado por Sarah Gibson en Mitsubishi Electric Research Laboratories (MERL) [7].

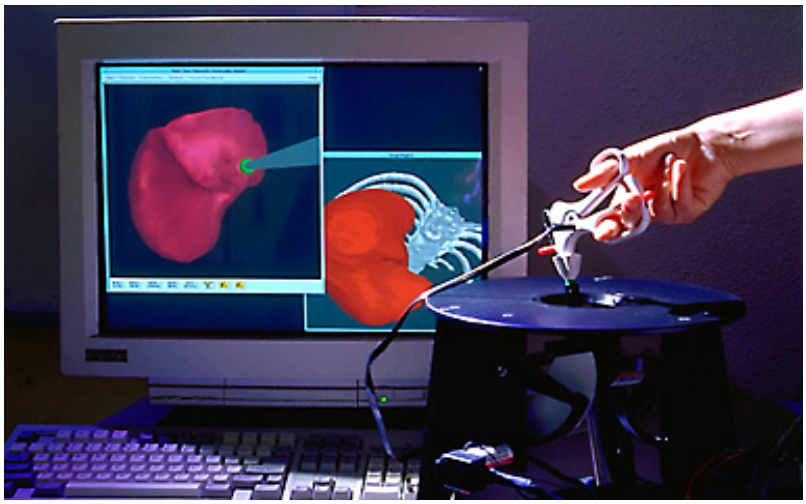

Fig. 1 Simulador de Cirugía en el Hígado. Centro de Investigación INRIA, Francia.

En la Universidad de las Ciencias Informáticas, se desarrollan aplicaciones en el área de virtual como juegos y simuladores dentro de ellos el Simulador Quirúrgico. Existe ya una biblioteca de clases desarrollada, llamada *SceneToolKit* (STK) [1], con implementaciones que facilitan el trabajo en la producción de simuladores. Se dispone de un módulo para

manipulación de objetos deformables, pero se requiere mayor velocidad en aras de representar escenas cada vez más complejas.

El **objetivo general** de este trabajo es desarrollar un módulo que se adjunte al STK con cualidades para representar objetos que alteren su forma ante la acción de factores externos, dependiendo de sus propiedades; basado en el algoritmo ChainMail y que garantice velocidades de render superiores a las existentes.

Varias son las ideas y las mejoras que presenta el algoritmo ChainMail. El **objeto de estudio** de este trabajo es el estudio e implementación de este como vía de solución, y el **campo de acción** está dado por los algoritmos de deformación que aporten elementos a la solución final, enfatizando en el algoritmo ChainMail y la arquitectura de la STK para garantizar una correcta integración a la herramienta.

Con el objetivo de satisfacer las necesidades planteadas se necesita realizar varias **tareas** que se mencionan a continuación.

- Estudiar las propiedades geométricas de los objetos virtuales.
- Analizar los diferentes algoritmos de deformación existentes, haciendo énfasis en las variantes del ChainMail.
- Estudiar las características del STK para la integración.
- Desarrollar la documentación del módulo.
- Implementar el módulo para dar solución al problema planteado.

Para una mejor comprensión, el documento está dividido en capítulos, donde el contenido queda estructurado de la siguiente forma:

En el capítulo uno “Fundamentación Teórica”, se hace un exhaustivo análisis de las técnicas usadas para la deformación de objetos desde sus inicios hasta nuestros días, destacando ventajas y desventajas de cada una de ella.

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

En el capítulo tres “Descripción de la Solución Propuesta”, se describe el sistema a desarrollar desde la perspectiva de las necesidades del cliente, en función de las dificultades, necesidades y características del cliente, y aplicando las técnicas a utilizar descritas en el capítulo anterior.

El capítulo cuatro “Diseño del Sistema”, corresponde a los flujos de trabajo de análisis e implementación de RUP, se describen las clases de diseño, se relacionan mediante el diagrama de clases de análisis y se distribuyen en componentes de software.

1

Capítulo 1 Fundamentación Teórica.

Introducción.

Lograr una representación de órganos lo más cercana a la realidad es una tarea nada fácil de lograr, pero no por ello se ha dejado de lado el estudio de métodos y algoritmos para lograr un resultado bastante exacto.

A continuación en este capítulo se dará una panorámica del estado del arte de los diferentes métodos y algoritmos para la deformación de objetos, y de sus principales ventajas y desventajas.

1.1 Características Generales y Antecedentes de las Deformaciones de Órganos.

Los objetos deformables son aquellos que de acuerdo a sus propiedades físicas alteran su forma ante la acción de un cuerpo o agente externo (gravedad, pinza, mano, etc.).

La representación de un objeto en un escenario virtual está dada generalmente por mallas triangulares o por la representación de un conjunto de puntos en el espacio, de tal forma que si se quiere lograr una deformación se tendría que alterar las posiciones de los vértices y de las aristas por las cuales está formado el objeto. En los sistemas de realidad virtual, usualmente se ejecutan varias acciones para deformar un objeto: primero se determina que parte del cuerpo del objeto se deformará y luego se calcula las posiciones de los vértices afectados para llegar a una nueva posición de estos, pero esta nueva posición depende de una serie de parámetros físicos ó no, dependiendo de la técnica utilizada.

Pero todo esto no es tan sencillo, lograr el cambio de ubicación de los vértices implica un alto costo en el rendimiento de las computadoras, y si a esto se le suma todo el escenario de un simulador quirúrgico y que se precisa de una frecuencia entre 300 Hz y 1000 Hz para tener una sensación realista en este tipo de aplicaciones [8] se hace mucho más complejo lograr una deformación en tiempo real.

La deformación de objetos se ha convertido en un importante campo de investigación dentro de los gráficos por computadora. Aunque son varias las técnicas de deformación que se han desarrollado todavía no puede asegurarse cuál de ellas es la más factible para lograr una escenificación lo suficientemente real y exacta de los objetos virtuales. Uno de los primeros modelos para la deformación de objetos fue introducido por Dimitri Terzopoulos [9], quien describió los modelos deformables usando las técnicas de minimización de la energía, tomadas de la ingeniería mecánica y usando la teoría de elasticidad, donde la fuerza aplicada a un elemento es absorbida o pasada a sus vecinos en variadas proporciones, gobernada por la elasticidad. Este primer modelo, es basado en la Ley de Hooke para objetos perfectamente elásticos, y es tomado como base para el desarrollo de las demás técnicas de deformación que se han desarrollado posteriormente.

Las técnicas de deformación se pueden dividir en dos grupos: en técnicas basadas en métodos geométricos y técnicas basadas en métodos físicos.

Las técnicas basadas en métodos geométricos son aquellos que emplean modelos de interpolación paramétrica (modelos polinomiales, splines…) para la estimación de la deformación, los cuales dependen sólo de relaciones no físicas. Esta técnica es muy efectiva en muchos simuladores quirúrgicos por su rapidez en el cálculo de la deformación. Entre este tipo de técnica se pueden encontrar Splines y Ajuste de Curvas, Deformación de Libre Forma y 3D ChainMail.

Entre las técnicas basadas en métodos físicos se pueden encontrar los Sistemas Masa-Resorte, Método de Elementos Finitos (MEF) y Método de los Elementos de Frontera (BEM) que se sustenta en la física de materiales y la mecánica continua. Aunque muchas personas son del criterio que estas técnicas aportan un mayor realismo a las aplicaciones que la usan, hoy en día son varios los Simuladores Quirúrgicos que eliminan la física de los modelos de órganos mediante la limitación de la interacción a una mera detección de choques entre objetos rígidos [10] o bien incorporan únicamente características físico elásticas [11] debido al alto costo computacional que implica el uso de las técnicas basadas en métodos físicos.

1.2 Sistemas de Partículas.

Sistemas de Partículas es un buen método para el modelado de objetos que cambian su forma, expandiéndose o contrayéndose como nubes, agua, humo, etcétera [40].

En un Sistema de Partículas un objeto es modelado a través de un conjunto de partículas. Cada partícula puede sujetar un juego de las propiedades como la vida, la forma, transparencia y el color. Por ejemplo, el agua es modelada a través de muchas gotas. Aun con su simplicidad, las partículas pueden mostrar comportamientos interesantes [41].

Este método es muy utilizado al modelar fluidos del cuerpo como la sangre, pero no solo es usado para simular fluidos, también es usado combinado con subdivisiones tetraédricas adaptativas para modelar la simulación de corte de tejidos y deformación en tiempo real con alto realismo para aplicaciones de simulación quirúrgica [42]. Eberhardt presentó su aplicación en un método para la representación de ropas, este permite modelar materiales de mucha suavidad, basado en las fuerzas no lineales para lograr dicha suavidad [43], debido a la gran flexibilidad de estos sistemas estos pueden ser usados en la simulación quirúrgica.

1.2.1 Dinámica Restringida.

En un modelo físico de partículas, estas son gobernadas por un conjunto de restricciones de movimiento además de sus propiedades físicas como las fuerzas que inciden. Por ejemplo restringiendo una partícula que se mueva en una curva determinada o entre dos partículas estas solo se pueden separar una distancia determinada, el problema de la dinámica restringida estaría en que las partículas obedezcan las leyes físicas de Newton y a la vez sus restricciones geométricas. Esto puede resultar complejo desde el punto de vista computacional además del costo de cálculo que lleva consigo hacer cumplir las restricciones geométricas junto con la aplicación de las fuerzas pudiendo llegar a ser numéricamente inestable.

Una forma de evitar este problema es calcular directamente las fuerzas necesarias para mantener las restricciones, estas se encargaran de evitar las fuerzas que actúen en contra de las restricciones; dado que las fuerzas aplicadas influyen en la aceleración específicamente, las fuerzas de restricción convierten las aceleraciones de las partículas en legales consecuentes con las restricciones.

1.2.2 Detección de Colisiones y Contacto.

La detección de colisiones es un aspecto de gran importancia al simular la interacción entre objetos virtuales, en la realidad los objetos no se compenetran pero si se deforman o cambian de posición en respuesta a una colisión, de aquí que lo primero

para una deformación precisa es detectar cuando están colisionando los objetos, y luego validar si esta colisión lleva a un estado límite del sistema y así deformarse el objeto según las propiedades del material que represente.

1.2.3 Modelado.

Los sistemas de partículas en general pueden ser usados para la representación de cuerpos deformables, aunque su campo de acción principal es la simulación de gases o fluidos como la sangre para el caso de la simulación quirúrgica. Ahora, los sistemas de partículas con restricciones o de Dinámica restringida son mejor usados para la modelación de sistemas deformables. Siendo utilizado tanto en sistemas basados en física como el Masa-resorte y otros basados en geometrías como el 3D ChainMail.

1.3 Representación Geométrica.

Una escena virtual puede contener distintos tipos de objetos (nubes, árboles, rocas, edificios, mobiliario, órganos, etc.) para los que existen una gran variedad de modelos de representación, entre los que se encuentran Modelos Alámbricos, Modelos de Superficies, Modelos Sólidos entre otros. Pero a continuación sólo se tomarán en cuenta los que debidos a sus características permiten la deformación de los objetos.

1.3.1 Modelos Basados en Mallas.

Dentro de los Modelos de Superficie se pueden encontrar la representación por mallas, la cual es una de las técnicas de modelado más populares que existe hoy en día. Tratando de ver las mallas desde su concepto, es una colección de vértices, aristas y polígonos conectados de tal forma que cada arista es compartida como máximo por dos polígonos representando así un objeto poliédrico. Las mallas pueden ser Estructuradas o No Estructuradas. En las primeras, la conectividad puede ser descrita por algún esquema de indexado; mientras que en las segundas, esa relación no existe y se hace necesario una estructura de datos especial para representar la información de dicha conectividad [12]. Entre los tipos de representaciones por mallas usados en los

gráficos por computadora se encuentran las mallas triangulares, las tetraédricas, las poligonales, entre otras, sin embargo las más populares son las mallas triangulares ya que son más fáciles de renderizar [51]. Las mallas triangulares consisten en un conjunto de puntos con coordenadas 3D, que representan una superficie, y una estructura que describe como estos puntos son conectados por triángulos [13] (Fig. 2). Este tipo de mallas no están especificadas en términos de una parametrización, aún así son consideradas como una representación explícita. Una malla triangular almacena básicamente las coordenadas de los vértices que la componen y los datos de conectividad que los enlazan. Para la mayoría de las mallas, el número de triángulos es aproximadamente el doble del número de vértices [14].

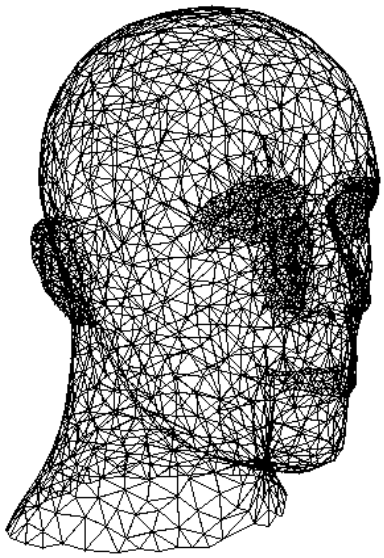

Fig. 2 Representación de un cuerpo mediante mallas triangulares.

1.3.2 Modelos Basados en Puntos.

Los sistemas de gráficos por computadora tradicionalmente han usado los triángulos como primitivas para la representación geométrica de los objetos. Con el objetivo de lograr un bajo costo del sistema en escenas que contienen gran cantidad de triángulos en sus geometrías se han realizado estudios en una nueva variante de representación geométrica que es basada en puntos. La representaciones geométricas basadas en puntos han experimentado un mayor renacimiento en los últimos años, aunque se puede decir que uno de los primeros trabajos donde se mencionan los puntos como

primitivas básicas en los gráficos por computadora fue el de Levoy y Whitted en 1985 [20], luego fue tratado por Grossman y Rally [21], en el año1998 y recientemente en el 2003 por Pauly M[57], así también Zwicker M [58] y su equipo en el 2004.

.

Las representaciones basadas en puntos han surgido como una variante eficiente para el proceso de modelado de objetos en escenas virtuales. Una de las principales razones del uso de este método es por su simplicidad y bajo costo en el rendimiento de las PC a la hora representar una escena virtual.

La superficie de un objeto 3D representada geométricamente por puntos está descrita por un conjunto de puntos de muestreo de una superficie continua (Fig. 3), una colección no necesariamente regular de posiciones 3D, opcionalmente con vectores normales asociados o propiedades auxiliares como color u otras propiedades del material representado. Se dice que este tipo de representación por sus propiedades presenta una adaptación natural a los cambios de topología de un objeto virtual. Hoy en día son varias las compañías que producen juegos de consola que utilizan esta técnica en la representación de personajes [22].También está siendo usada en la representación de objetos que se deforman mediante la técnica de *Free-Form Deformation* [52].

Fig. 3 Representación de un rostro mediante puntos.

La representación geométrica basada en punto está llamada a convertirse en una de las técnicas de representación más usada en el campo de los gráficos por computadora. Hoy en día los proyectos de desarrollo que existen en el campo de la representación

geométrica basada en puntos son muy ambiciosos; ya se habla de trabajos donde se representen objetos naturales como árboles y plantas.

1.4 Métodos basados en Geometrías.

Muchas son las aplicaciones en la que la técnica de deformación usada es basada en propiedades físicas, no por eso se dejan de desarrollar los métodos en el cual sus propiedades se basan en la geometría solamente. En estos métodos se cambian las energías por restricciones geométricas y las fuerzas son reemplazadas por distancias desde cada posición inicial hasta la posición final. Muchos de estos métodos tienen en cuenta una serie de propiedades geométricas, como son el área de la superficie y el área del volumen a la hora de efectuarse una deformación.

Las deformaciones que se logran con este tipo de método son muy eficientes si de velocidad en la deformación y de rendimiento se trata. Pero se ve muy limitado si lo que se desea es exactitud en las deformaciones debido a la carencia de los parámetros físicos.

1.4.1 Deformación de Libre Forma.

El método de Deformación de Forma Libre (*Free-Form Deformation FFD*) es un método general para la deformación de objetos mediante el ajuste de puntos de control. Este método basa su funcionamiento en alojar un objeto en una región del espacio o volumen deformable, de tal forma que cada punto del objeto presente una parametrización única en el espacio 3-D, que define su posición en la región. La región de deformación está descrita por un paralelepípedo al cual se le asocia un sistema de coordenadas locales. La región de deformación de coordenadas es mallada de manera que se establecen planos perpendiculares formando una malla o rejilla de puntos de control en el espacio 3-D. Esta rejilla es posteriormente alterada al asignársele propiedades de deformación elástica, causando un ajuste de la posición de los puntos del objeto, basado en su parametrización inicial (Fig. 4). Tal ajuste en la posición de los puntos del objeto, está definido por una función de deformación que se encarga de

establecer la correspondencia de los puntos del objeto inicial y de los puntos del objeto deformado. La función de deformación está definida por el producto tensorial de funciones polinomiales de Bernstein [53].

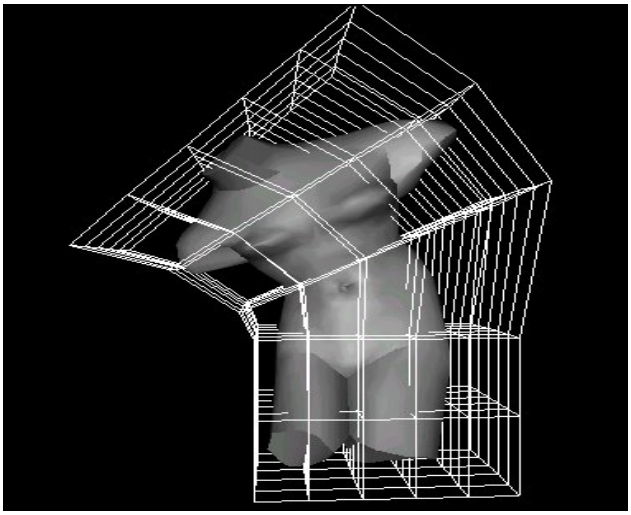

Fig. 4 Deformación de un cuerpo mediante FFD

Los primeros trabajos en deformación libre fueron introducidos por Sederberg y Parry [23] en 1986 y estaban basados en la deformación de objetos contenidos en una estructura espacial *lattice,* regularmente dividida y definida por una matriz tridimensional de puntos de control [27]. Posteriormente Griessmair y Purgathofer [24] presentaron una nueva técnica que estaba basada en estructuras tri-variadas B-Spline (B-Spline FFD). A continuación la técnica de FFD es extendida por Coquillart [25], ampliando las estructuras utilizadas a formas no paralelepípedas. Estas estructuras de deformación resultan más apropiadas para definir deformaciones cilíndricas, esféricas, etc. Otra técnica consiste en la utilización de estructuras espaciales racionales [26], donde los pesos asociados a los puntos de la estructura también intervienen en la deformación. En todos estos casos la técnica de deformación, consiste en asociar el objeto a la estructura de deformación. Otras técnicas de deformación libre se basan en la deformación directa del objeto [28]. En este caso no se utiliza una estructura espacial como herramienta de deformación sino que se modifica directamente el espacio que contiene el objeto utilizando una función general de deformación, generalmente definida

a través de restricciones en los desplazamientos de los puntos del objeto que se desea deformar [29].

La carencia de propiedades internas que impiden crear deformaciones realistas y las limitaciones que presenta a la hora de deformar estructuras complejas son alguna de las desventajas que presenta este método. Pero gracias a su facilidad de uso y el poder de crear muchos tipos de deformaciones con poca interacción por parte del usuario y por su rapidez en las deformaciones, este método ha sido usado en varios trabajos como en el de simulación de cavidades cardiacas, y en sistemas de simulación quirúrgica como el de *Laparoscópica* implementado por C. Basdogan en 1998 [30].

1.4.2 3D ChainMail.

La mayor desventaja de las representaciones volumétricas consiste en que los objetos pueden estar compuestos por millones de elementos. Esta cantidad de datos requiere de una mayor potencia en los ordenadores para poder realizar un render en tiempo real, así como el modelado realista de objetos, de ahí la necesidad de algoritmos que eliminen o moderen todas estas restricciones.

En 1997 es presentado un algoritmo de deformación por Sara F. F. Gibson [31], inicialmente para objetos deformables de escenarios quirúrgicos, aunque es un algoritmo basado en geometría, este algoritmo puede modelar una amplia variedad de materiales [32], incluyendo rígidos, deformables, elásticos y sustancias plásticas. Además el método puede modelar materiales anisótropos tales como músculos. La idea básica de este es que cada elemento del volumen es vinculado con sus seis vecinos más cercanos. Cuando un elemento de la estructura es presionado, los enlaces cercanos absorben el movimiento cumpliendo sus restricciones de movimiento (Fig. 5). Si un vínculo entre dos elementos es estirado o comprimido a su límite, los desplazamientos son transferidos a sus enlaces cercanos. De este modo los desplazamientos pequeños de un área seleccionada en un sistema relativamente suave resulta solamente en deformaciones locales del sistema, al igual que en una cadena los elementos vecinos solo responden si las restricciones de distancia son

violadas (Fig. 6). Cambiando estas restricciones se pueden modelar tanto cuerpos rígidos como deformables.

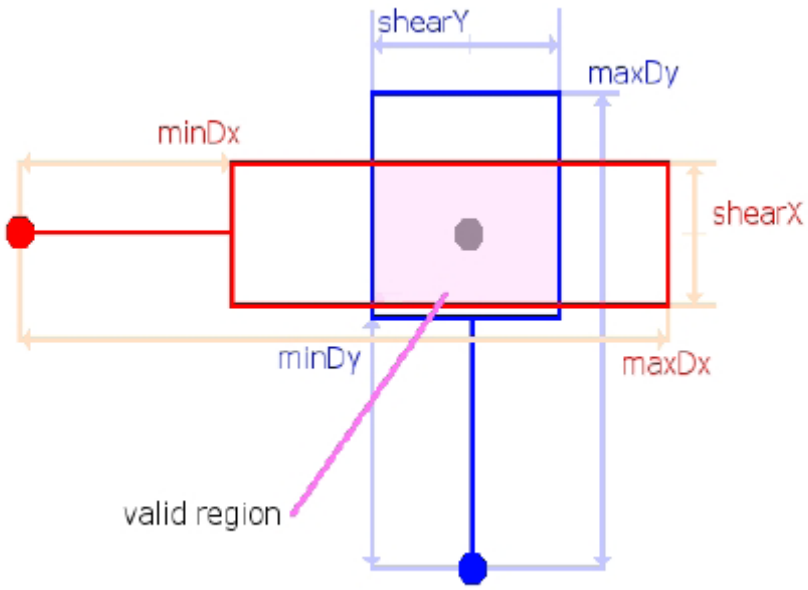

Fig. 5 Regiones válidas de un elemento ChainMail.

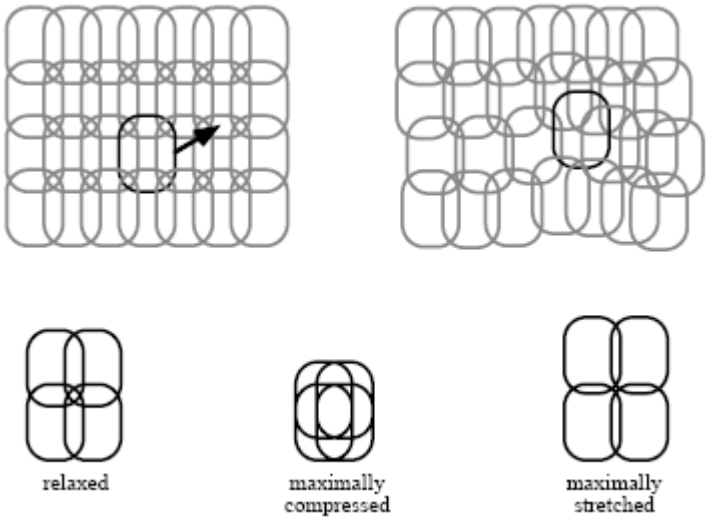

Fig. 6 Representación 2d del algoritmo ChainMail al ser movido un elemento.

Este algoritmo cuenta con tres ventajas principales, lo primero es que cada elemento es considerado a lo máximo una vez en cada deformación, en segundo lugar cada

elemento es comparado con solo un vecino, para determinar como el debe moverse y por último la deformación se propaga de adentro hacia fuera y esta debe terminar lo antes posible. Esto hace que permita trabajar con gran cantidad de datos y seguir comportándose de forma interactiva. Este algoritmo soporta cambios topológicos en caso de una modelación dinámica, por lo que los vínculos de los elementos con sus vecinos pueden ser fácilmente desconectados al ser necesario, esto es muy utilizado en el proceso de corte de tejidos en la simulación quirúrgica.

No obstante este algoritmo también posee desventajas, por ejemplo fue diseñado para funcionar en mallas rectilíneas solamente y con datos homogéneos, pero desde entonces se han escrito otras variantes que arreglan esta situación, primeramente en 1998 M. A. Schill [33] introdujo el algoritmo permitiendo el procesamiento con datos heterogéneos, pero con el problema de que no asegura que cada elemento pueda ser verificado como máximo una vez, por lo que la ventaja de la velocidad se pierde un poco , M. A. Schill resuelve esto con listas ordenadas en el proceso de selección de los vecinos a mover.

En el 2002 Y. Li y k. Brodlie [34] escribieron una nueva variante del algoritmo, donde solucionaban el problema de la malla rectilínea en este momento los elementos pueden tener cualquier número de vecinos y no se supone acerca de la topología de los elementos vecinos. Para manejar las mallas no uniformes se cambian las restricciones absolutas por relativas para los elementos vecinos, cuando se dice malla no uniforme se quiere decir cualquier malla topológicamente equivalente a una uniforme que tiene conexiones con elementos de su interior (Derecho, Izquierdo, Arriba, Abajo, Delante y Detrás).

Recientemente en el año 2005 C. Drägner [30] publica una nueva variante del algoritmo ChainMail donde reutiliza algunas de las mejoras de estos trabajos además de utilizar la administración de la memoria explotando el hecho de que las deformaciones se propagan localmente y hacia fuera, por lo que se divide el volumen en mallas de macro cubos que son cargados según la demanda de los mismos minimizando el uso de memoria, esto repercute en velocidad para el algoritmo.

Este algoritmo puede ser comparado con el método basado en física Masa-Resorte y permite la realización de corte en tejidos presentando ventajas, además de representar sus deformaciones de manera interactiva.

1.4.3 Splines y Ajustes de Curvas.

Para la modelación de objetos, o la representación de objetos reales se tiene como una variante representar curvas y superficies. En el caso de la representación de objetos reales normalmente no existe un modelo matemático previo del objeto, por lo que este se aproxima con "pedazos" de planos, esferas y otras formas simples de modelar, requiriéndose que los puntos del modelo sean cercanos a los correspondientes puntos del objeto real. Para lograr estas representaciones se usan las curvas de Bézier y otros métodos de especificar curvas con pequeños vectores de números, entre los que están Splines, B-Splines, NURBS y otros [37], donde un Spline es comúnmente referida como una curva compuesta de secciones polinomiales satisfaciendo ciertas condiciones de continuidad entre ellas; y donde B-Splines son curvas de aproximación (no interpolan los puntos de control) que tienen la ventaja de permitir una manipulación local de la curva y requerir menos cálculos para la determinación de sus coeficientes.

Estos métodos se basan en controlar las deformaciones a objetos volumétricos mediante curvas (Splines, B-Splines, NURBS, Bezier o interpolación polinomial), de manera general estas curvas están formadas por una serie de puntos llamados puntos de control. La forma de los objetos puede ser ajustada moviendo los puntos de control o variando la cantidad de ellos (Fig. 7). La deformación estaría definida entonces por funciones paramétricas (3D Spline), las que se determinan por la ubicación de los puntos de control.

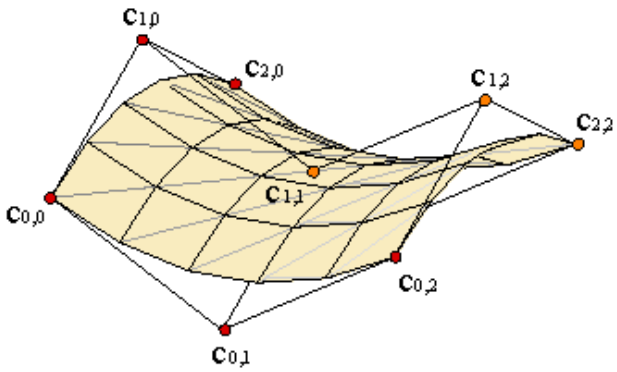

Fig. 7 Curva B-Spline.

Este método es muy ventajoso desde el punto de vista del bajo costo que implica su representación en el rendimiento de la PC, pero se ve muy limitado a la hora de realizar representaciones muy precisas de determinados objetos, debido a que la colocación de los puntos de un objeto determinado es muy difícil de lograr. La utilización de este método se ha visto mayormente en la representación de objetos como órganos, no así en el uso de deformaciones [38].

1.5 Métodos basados en Física y Soluciones Numéricas.

Los Métodos Basados en Física se han convertido en una poderosa herramienta para simular las deformaciones en telas [16], tejidos humanos o cualquier objeto con propiedades deformables.

Las propiedades elásticas restringen el movimiento y dinámica de los objetos en el mundo real, por lo tanto para modelar y simular las características físicas de los objetos es necesario obtener una representación real en los gráficos generados en las computadoras, así como su visualización y animación.

Los métodos basados en física han estado incluidos desde las primeras investigaciones para los gráficos por computadoras desde principios de los años 80, combinando la Dinámicas Newtonianas, Mecánica Continua, Computación Numérica, Diferencial Geométrico, Cálculo de Vectores, Teorías de Aproximación, por solo mencionar algunos, dentro de este campo que está siendo explorado y extendiéndose, en constante flujo, con avances visuales impresionantes [39].

Hay que mencionar que la simulación basada en física necesita un alto costo computacional, dado a la solución de las ecuaciones que pueda presentar, por lo tanto su interactividad requiere de técnicas especiales.

1.5.1 Sistemas Masa-Resorte:

Un sistema Masa-Resorte consiste en un conjunto de *n* puntos con masa, cada uno vinculado a sus vecinos por resortes sin masas y largo mayor que cero [19] (Fig. 8), dominados por la 2da Ley de Newton $f = m \times a$, donde m es la masa de cada punto y f es la sumatoria de las fuerzas incidentes sobre dicho punto, las fuerzas están divididas en dos categorías; internas, dada por la tensión de los resortes regida por la Ley de Hooke, que es el resultado de la rigidez de la vinculación de todos los resortes de este punto a sus vecinos, y externas, que pueden diferir en su naturaleza dependiendo de qué tipo de simulación se quiera modelar, frecuentemente son la gravedad y el amortiguamiento viscoso.

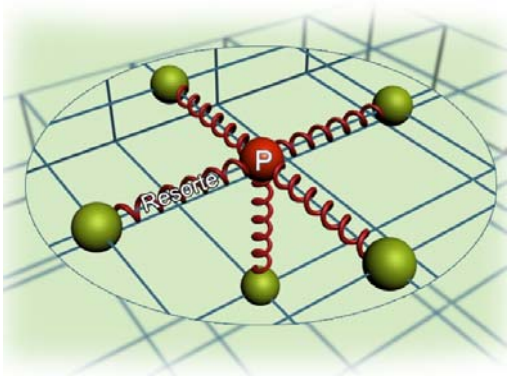

Fig. 8 Sistema Masa-Resorte Relajado

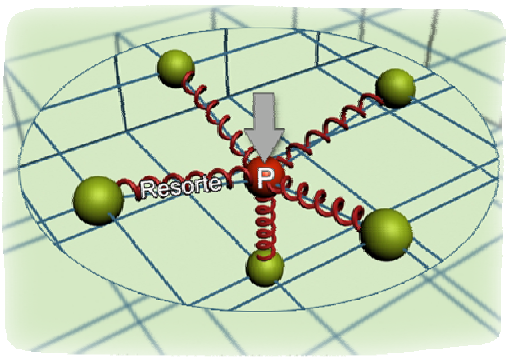

Fig. 9 Sistema Masa-Resorte Aplicando Fuerza

Si una deformación ocurre esta se propaga a través del objeto por sus tensores según las fuerzas aplicadas y las características de estos (Fig. 9), este método es de fácil implementación y provee al sistema con un cómputo de sus deformaciones rápidas, permitiendo la simulación de un rango de objetos en los que se incluyen los tejidos visco-elásticos en el caso de la cirugía.

Una manera eficiente de utilizar los sistemas masa-resorte es modelando varias capas del material como presentaron en su trabajo D.Terzopoulos and K. Waters. Ellos usaron tres capas de Masa-Resorte para animación facial, estas estaban asociadas a tres capas anatómicas del tejido facial (Dermis, Subcutánea y Músculos) [15].

El uso de sistemas Masa-Resorte ha sido difundido en trabajos como la animación facial antes mencionada, en la animación de telas [16], y en la modelación de

deformaciones de músculos, en el año 1998 Nedel y Thalman [35] desarrollaron un algoritmo de este tipo con resortes angulares adicionales para preservar la forma del músculo, pero su enfoque estaba basado en un volumen simplificado de cálculo y no garantizaba la preservación de volumen exacta, además puede modificar las propiedades de los materiales debido a los resortes adicionales, por lo que demuestra el comportamiento irrealista en deformaciones grandes. También tiene como dificultad los valores constantes de los muelles para lograr los comportamientos apropiados, especialmente en objetos rígidos como los huesos, haciendo que la modelación de objetos rígidos a través de este método es una pérdida de rendimiento, debido a que sus deformaciones son insignificantes o nulas. En el 2005 M. Hong [36] y su equipo publican una técnica de Masa-Resorte que posibilita no sólo la preservación de volumen sino también la capacidad de controlar la distribución del volumen dependiendo de las propiedades materiales y las características de cargas externas. También ha sido usado en la simulación quirúrgica, por ejemplo el software llamado “Karlsruhe Endoscopy Surgery Trainer” (KISMET) presentado por U. Kuhnapfel en el 2000 [17]. Además en este campo de cirugía fue utilizado por Roger Webster y su equipo integrando además un indicador para la aceleración del cálculo en cada paso [18].

1.5.2 Método de Elementos Finitos.

El Método de los Elementos Finitos (MEF) es un método de aproximación de problemas continuos, donde el continuo se divide en un número finito de elementos, cuyo comportamiento se especifica mediante un número finito de parámetros asociados a ciertos puntos característicos denominados nodos**.** Estos nodos son los puntos de unión de cada elemento con sus adyacentes, la solución del sistema completo sigue las reglas de los problemas discretos. El sistema completo se forma por ensamblaje de los elementos, las incógnitas del problema dejan de ser funciones matemáticas y pasan a ser el valor de estas funciones en los nodos, el comportamiento en el interior de cada elemento queda definido a partir del comportamiento de los nodos mediante las adecuadas funciones de interpolación ó funciones de forma [54]. El MEF, por tanto, se basa en transformar un cuerpo de naturaleza continua en un modelo discreto

aproximado, esta transformación se denomina discretización del modelo (Fig. 10). El conocimiento de lo que sucede en el interior de este modelo del cuerpo aproximado, se obtiene mediante la interpolación de los valores conocidos en los nodos. Es por tanto una aproximación de los valores de una función a partir del conocimiento de un número determinado y finito de puntos.

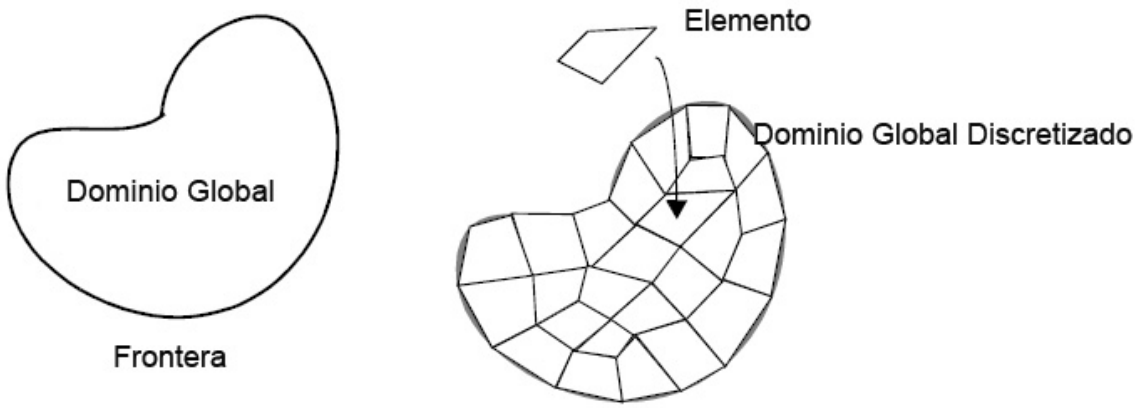

Fig. 10 Discretización de un dominio en elementos.

Aunque es una eficiente manera para calcular la representación de diferentes procesos físicos con un elevado realismo, dentro de ellos las deformaciones de objetos en las aplicaciones virtuales; dado su fuerte cálculo numérico, la resolución de sus ecuaciones resulta muy costosa computacionalmente por lo que limita su uso en aplicaciones que necesitan interactividad en tiempo real.

Muchos trabajos han encontrado solución con el Método de Elementos Finitos desde hace aproximadamente dos décadas atrás. Terzopoulos [15] lo uso para modelar deformaciones físicas como la animación facial basado en la teoría de la Elasticidad, Bro-Nielsen y Cotin [49] usan el método MEF linealizado en la simulación quirúrgica, donde utilizan la deformación elástica lineal sujetada a la Ley de Hooke, otros estudios aplicados a la simulación quirúrgica específicamente han sido realizados como el trabajo hecho en 1999 por Cotin y Delinguette [4] donde se genera el modelo volumétrico desde imágenes médicas escaneadas y se simula la deformaciones basado en la teoría de la elasticidad adecuando la forma del volumen con las fuerzas implicadas en la deformación, otros como Nienhuys y Van der Stappen usan un algoritmo interactivo para el método MEF linealizado para simulación de deformaciones proceso

de corte. En el 2004 Irving y Fedkiw [50] desarrollan un método basado en MEF que permite deformaciones estables.

1.5.3 Método de los Elementos de Frontera.

El Método de los Elementos de Frontera (*Boundary Element Method,BEM*) es una importante técnica matemática que da solución a diferentes problemas científicos. Se puede decir que el BEM es alternativa del MEF, aunque se puede considerar que el BEM para algunos casos es mucho más eficiente que el MEF pues sólo se requiere de una malla sobre el modelo de superficie al aplicarse este método, aunque nada más se puede aplicar este método a cuerpos donde su interior está compuesto por un material homogéneo.

En el Método de los Elementos de Frontera los cálculos se realizan en la superficie del cuerpo elástico en lugar de su volumen. El método logra un aumento sustancial de la velocidad debido a que el problema tridimensional original, es reducido a dos dimensiones. (Fig. 11).

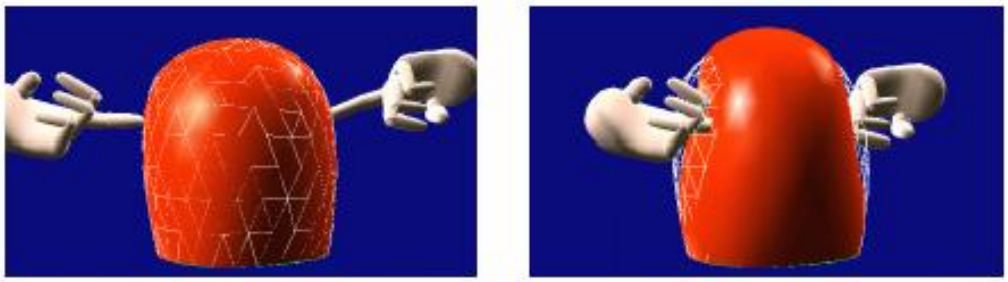

Fig. 11 El Método de los Elementos de Fronteras en la deformación de un objeto.

Una de las primeras aplicaciones exitosas del BEM, a problemas no lineales independientes del tiempo en mecánica de sólidos con la formulación para plasticidad fue debido a Swedlow y Cruse [44]. Esto fue seguido por una implementación numérica por Riccardella [45]. Otra formulación para plasticidad basada en esfuerzos iniciales se debe a Banerjee y Mustoe [46], otros como Mukherjee [47] han estado trabajando en la

aplicación del BEM para problemas no lineales dependientes del tiempo en deformaciones inelásticas. Otras aplicaciones del BEM incluyen la elástica y muy recientemente la mecánica de fractura inelástica [48].

El Método de Elementos de Frontera ha demostrado su eficacia para resolver algunos tipos de problemas. Por ejemplo, permite la simulación de problemas que tienen variaciones muy pronunciadas de esfuerzos, como es el caso de la fractura o agrietamiento de un sólido, así como simulación de fluidos, y de objetos deformables. Permite además el modelado de problemas complejos, en especial aquellos que requieren la resolución en regiones abiertas o una definición precisa de los contornos.

En comparación con otros métodos el BEM es muy eficiente en términos de discretización y de recursos computacionales para problemas donde existe una pequeña proporción entre volumen y superficie de radio. Sin embargo para muchos problemas este método es menos eficiente que los métodos de volúmenes discretos (FEM, *Finite Difference Method, Finite Volume Method*). En cuanto a la discretización en el BEM se limita específicamente a la frontera, mientras que en el MEF el dominio completo tiene que ser discretizado, haciendo este último mucho más costoso al hablar del rendimiento de la PC.

Aunque el BEM puede llegar a ser muy útil cuando se habla de ahorro de tiempo en la creación y modificación de mallas, tiene una aplicación práctica relativamente reciente.

Conclusiones.

Al finalizar este capítulo se cuenta con una noción de los principales métodos de deformación de objetos. Aquí se han visto las formas de representación de estos objetos deformables y las principales características de cada uno de ellos. Por otro lado se ha hecho un amplio bosquejo de los métodos de deformaciones de objetos, clasificándolo en dos grupos particulares, los métodos que sus soluciones se basan en técnicas geométricas y los que se basan en física exponiendo las principales ventajas de cada uno de estos, así como las principales dificultades que pueden implicar su uso.

2

Capítulo 2 Soluciones Técnicas.

Introducción.

Luego de lo visto en el capítulo anterior, se cuenta con el conocimiento acerca de una serie de algoritmos de deformación; Los basados en física como son, el Masa-Resorte, Elementos de Frontera, Soluciones Numéricas como MEF, y otros algoritmos basados en geometrías como el Deformación de Libre Forma o el ChainMail por citar algunos. Con lo que queda demostrado que la representación y deformación de objetos virtuales resulta bastante costosa en su procesamiento, mientras más preciso con la realidad sea el método de deformación que se implemente, sirviendo este conocimiento como premisa en la solución que se propone a continuación.

2.1 Método de Deformación.

Después de haber analizado y estudiado los distintos métodos y soluciones para la solución del problema de la simulación de deformaciones en entornos virtuales, y visto ya que los métodos físicos aunque muestran representaciones bastantes reales estos al contrario de los geométricos necesitan más recursos en términos de procesamiento que los antes mencionados dado los complejos cálculos que llevan consigo.

Existe un método basado en física que resulta bastante apropiado en términos de velocidad de procesamiento y acercamiento a deformaciones reales, el Masa-Resorte pero con la dificultad que su aplicación solo es recomendable en deformaciones pequeñas [55] para poder representar deformaciones con velocidades en tiempo real.

Este trabajo propone una solución que aunque no resulte lo bastante real, dado que se basa en cálculos simplemente matemáticos, sí resulta menos costosa y posibilita velocidades interactivas para deformaciones de proporción mayor, dicha solución está dada por el algoritmo 3D ChainMail de Sarah F. Frisken Gibson [33].

2.1.1 Generación de la Rejilla Regular.

La herramienta STK permite la representación de mallas triangulares, lo que posibilita acceder a los vértices de estas. A partir de la información que brinda la herramienta se puede generar una Rejilla Regular, donde los nodos que conforman dicha rejilla contendrán información como: los vértices de la malla, la dirección de sus nodos vecinos y otros datos necesarios.

Tomando la información de los vértices de la malla se calcula los extremos máximo y mínimo del objeto representado, recorriendo la lista de vértices y escogiendo los valores máximos y mínimos de cada eje de coordenada; con este resultado se obtiene la longitud de las aristas $a1, b1, c1$ del paralepípedo envolvente.

Para la creación de los nodos que conforman la rejilla se calcula el Área promedio de los triángulos que forman la malla, esta Área se iguala al Área de un cubo, despejando el lado se obtiene la longitud de la arista de los nodos. Como sigue:

$$Atriángulo = \frac{b.h}{2} \quad (1)$$

Donde b, h son la base y la altura del triángulo respectivamente.

Sumando las Áreas de los triángulos y dividiéndola entre la cantidad total de triángulos se obtiene el Área promedio. Como sigue:

$$Apromtriángulo = \frac{\sum (A1 + A2 + .. + An)}{n} \quad (2)$$

Donde A es el área de un triángulo y n es la cantidad total de triángulos.

El área del cubo se obtiene de la siguiente manera:

$$Acubo = 6a^2 \quad (3)$$

Cada cubo que forma parte de la rejilla regular debe de contener la menor cantidad posible de los vértices que forman la rejilla, para ello se iguala el área del cubo $Acubo$ al área promedio de los triángulos $Apromtriángulo$ con el objetivo de conocer el valor de la longitud que tendrá la arista a de los cubos que formarán la rejilla regular, como sigue:

$$Apromtriángulo = 6a^2 \quad (4)$$

Despejando para obtener el valor de la arista a se obtiene:

$$a = \sqrt{\frac{Apromtriángulo}{6}} \quad (5)$$

Para Calcular la cantidad de cubos que se generaran en cada eje coordenada se hace la división entera de las longitudes de los ejes de coordenadas $a1, b1, c1$ pertenecientes al paralepípedo entre la longitud de la arista a de los cubos.

Con estos datos se generará una rejilla regular tri-dimensional que será la representación espacial del objeto ChainMail en cubos de tamaños iguales como se muestra en la Fig. 12.

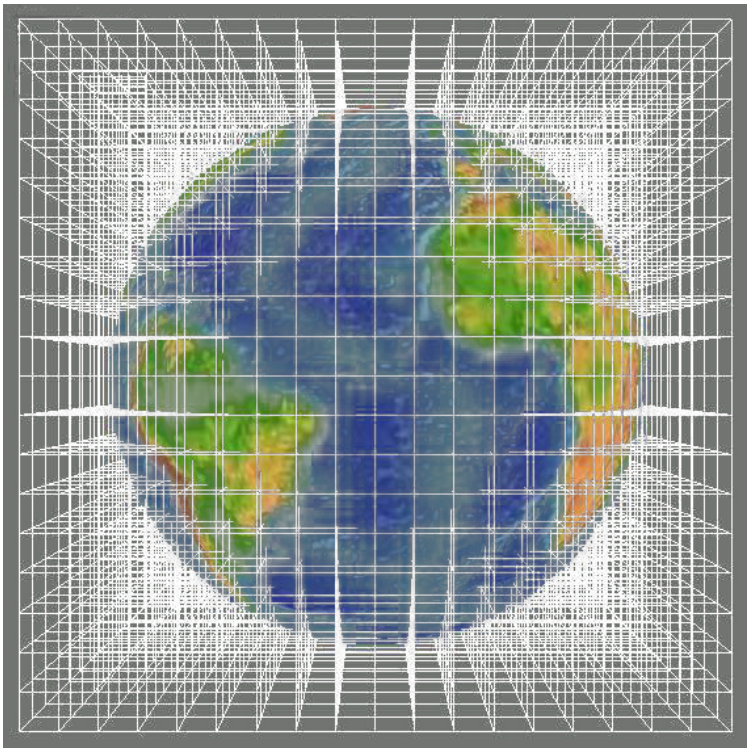

Fig. 12 Rejilla Regular generada para el Objeto Deformable.

Cada cubo tendrá una posición dentro de la matriz tridimensional que forma la rejilla regular, es decir (Fila, Columna, Profundidad). Por ejemplo el cubo cuya posición es (3, 3, 3) sería el que se muestra en la Fig. 13.

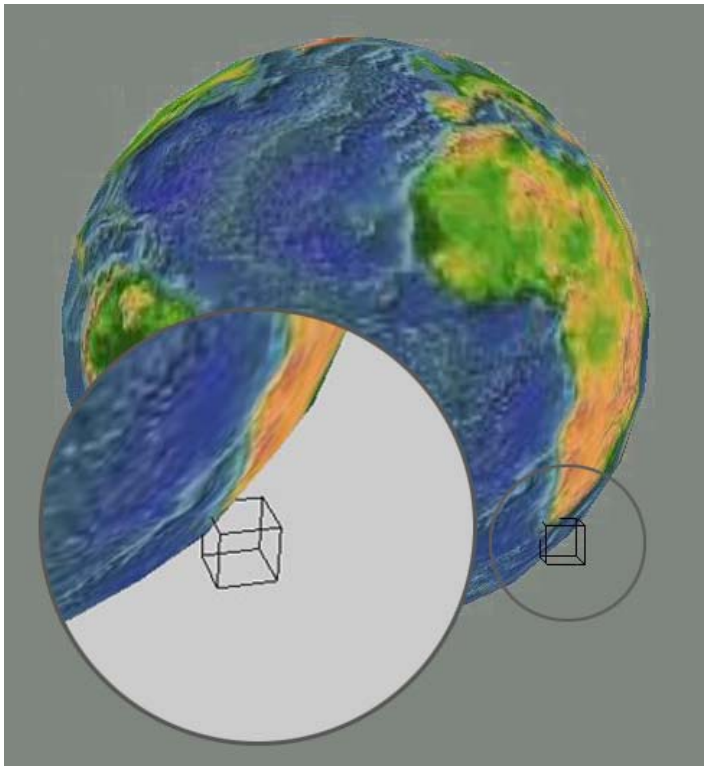

Fig. 13 Posición del Cubo (3, 3, 3) en la Rejilla Regular generada.

.

Cada cubo contendrá la lista de vértices que pertenecen a su dominio espacial, así como sus extremos máximo, mínimo y su respectivo centro, además de conocer a sus seis vecinos: Derecho, Izquierdo, Superior, Inferior, Delantero y Trasero.

2.1.2 3D ChainMail.

Con la rejilla regular formada cada nodo de ella deberá cumplir con un rango horizontal $MinDx$ y $MaxDx$ para sus vecinos izquierdo y derecho, un rango vertical $MinDy$ y $MaxDy$ para sus vecinos superior e inferior y dentro de un rango de profundidad $MinDz$ .y $MaxDz$ para sus vecinos anterior y posterior respectivamente.

Existen 6 listas de nodos candidatos a movimiento para tratar la deformación, cuando un cubo es atraído o empujado sus vecinos son automáticamente agregados a sus respectivas listas de candidato, se verifican sus restricciones para cada eje coordenada, con respecto a su vecino referencia, que seria quien le transfirió el

movimiento. Por ejemplo si un nodo es comprimido el elemento y su vieja posición son adicionados en una lista de elementos movidos, y sus vecinos cercanos son adicionados a sus respectivas listas de candidatos.

La lista de los elementos candidatos es procesada una a la vez en el siguiente orden: derecho, izquierdo, superior, inferior, anterior y posterior. La lista de candidatos de los vecinos derechos es procesada de la siguiente manera: comenzando con el primer elemento en la lista, sus restricciones de deformación son chequeadas entre el elemento de la lista y el elemento que lo adicionó (siempre es el vecino izquierdo). Si las restricciones son violadas el elemento es movido una mínima distancia hasta que cumpla con las restricciones. La nueva posición del cubo se calcula de la siguiente manera:

$$(x - xneigh) < MinDx \ , \ \ x = xneigh + MinDx \qquad (6)$$

$$(x - xneigh) > MaxDx \ , \ x = xneigh + MaxDx \qquad (7)$$

Donde x es el valor de las coordenadas del nodo comprimido, $xneigh$ es el valor de la coordenada de su vecino izquierdo.

Si el elemento es movido, sus vecinos son adicionados a sus respectivas listas de candidatos. Cada candidato en la lista es procesado hasta que no quede ninguno.

La lista de los vecinos izquierdo es procesada de forma similar excepto que el elemento izquierdo es introducido por su vecino derecho, entonces el movimiento del nodo izquierdo causa que sus vecinos izquierdo, frente, trasero, superior e inferior sean adicionados a la lista de candidatos. Ahora la nueva posición del cubo se calcularía usando (6), (7); pero ahora $xneigh$ sería el valor de la coordenada del vecino derecho.

Las demás listas de vecinos son procesadas de forma similar para cada caso; solo que con los vecinos superior e inferior la nueva posición del cubo se calcularía de la siguiente manera:

$$(y - yneigh) < MinDy \ , \ \ y = yneigh + MinDy \qquad (8)$$

$$(y - yneigh) > MaxDy \ , \ \ y = yneigh + MaxDy \qquad (9)$$

Donde y es el valor de las coordenadas del nodo comprimido y $yneigh$ es el valor de la coordenada de su vecino superior o inferior, depende del vecino que se esté tratando.

Para el caso de la lista de vecinos anterior y posterior la nueva posición de calcularía de la siguiente forma:

$$(z - zneigh) < MinDz\ ,\ z = zneighs + MinDz \quad (10)$$

$$(z - zneigh) > MaxDz\ ,\ z = zneigh + MaxDz \quad (11)$$

De esta forma cada nodo que es chequeado cambia su posición y propaga la deformación para cada uno de sus seis vecinos.

De manera general pudiera escribirse el algoritmo como sigue:
Para cada triángulo calcular el área usando (1).
Calcular el área promedio de los triángulos usando (2).
Determinar los valores de los extremos máximos y mínimos de la malla.
Calcular longitud de la arista de los cubos de la rejilla usando (3), (4), (5).
Generar la rejilla regular.
En cada ciclo de render:
Para cada nodo de la rejilla:
Determinar si fueron violadas sus restricciones de deformación.
Adicionar nodo a la lista de vecinos para ser procesados.
Calcular la nueva posición usando (6), (7), (8), (9), (10), (11).

2.2 Consideraciones Técnicas Generales.

El proceso de Desarrollo estará guiado por el Rational Unified Process (RUP) soportado por la herramienta CASE Rational Rose Enterprise Edition 2003 y el Enterprise Architect versión 7.0.

La implementación se realizará siguiendo el paradigma de programación orientada a objeto en lenguaje C++ con Microsoft Visual C++ .NET 2003 para garantizar su compatibilidad con el STK.

Las nomenclaturas y estándares de codificación son las mismas que las del STK, la escritura del código fuente será en C++ estándar para su compresión por Windows así como Linux.

Conclusiones.

Como se pudo apreciar en este capítulo se propone la solución al problema de las deformaciones planteado anteriormente, esta solución está basada en la búsqueda de una velocidad de render superior a la del algoritmo ya existente. Se planteó la metodología de desarrollo de software a utilizar y el lenguaje de programación.

3

Capítulo 3 Descripción de la Solución Propuesta.

Introducción.

Contando ya con la propuesta de la solución al problema de las deformaciones, se dará a continuación en este capítulo un enfoque de la solución, sobre la base de las dificultades, necesidades y características del cliente con la aplicación de la técnica descrita anteriormente.

3.1 Reglas del negocio:

El sistema solo soporta mallas triangulares.

El sistema solo soportará hasta 30 000 cubos dentro de la matriz tridimensional.

3.2 Modelo del Dominio:

En el modelo que se verá a continuación se presenta los eventos que suceden en el entorno en el que trabaja el sistema, y se muestran también los objetos más importantes en el contexto del sistema.

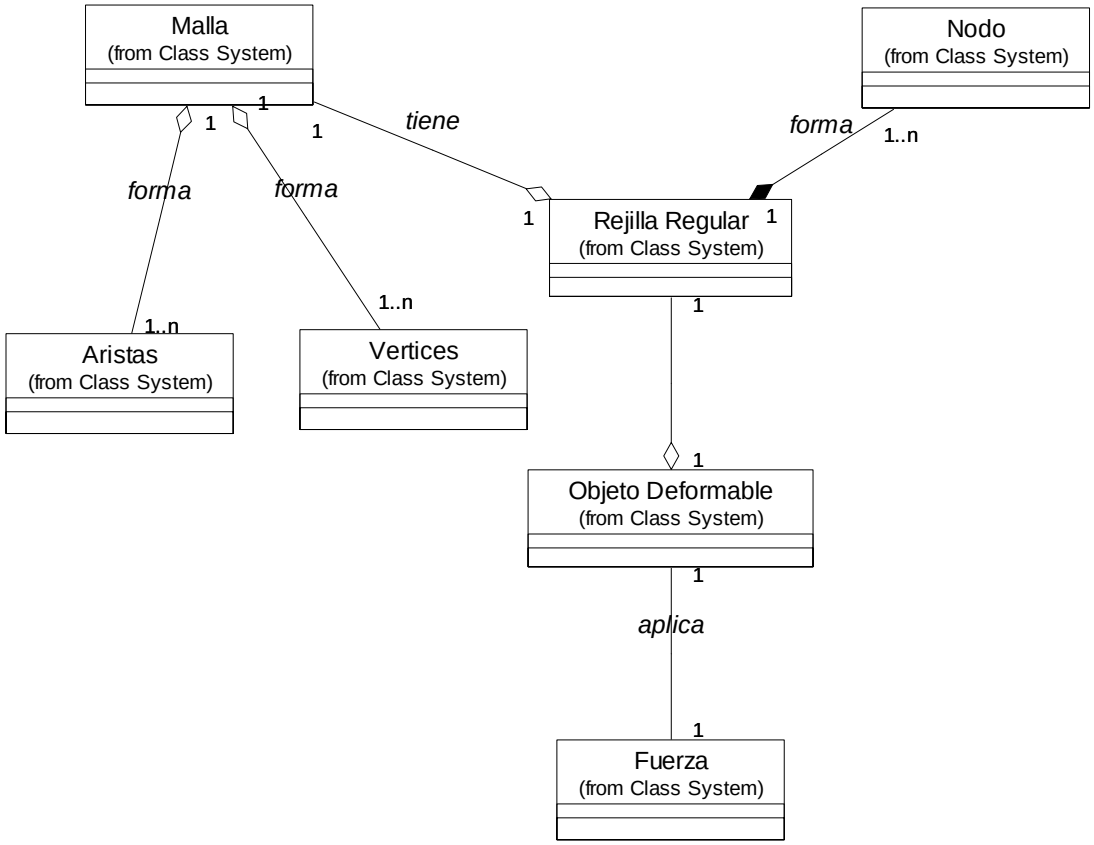

Fig. 14 Modelo de Dominio.

3.2.1 Glosario de términos del Modelo de Dominio.

Malla: Representa la malla triangular que será deformada como resultado final de todo el proceso.

Rejilla Regular: Estructura encargada de posicionar los nodos de forma tal que envuelvan la malla para su posterior deformación.

Nodo: Estructura encargada de almacenar los valores de las posiciones de los vértices que conforman la malla.

Objeto Deformable: Objeto que alterará su forma ante la acción de una fuerza externa.

3.3 Captura de Requisitos:

A continuación se exponen los requisitos funcionales y no funcionales que el sistema debe cumplir.

3.3.1 Requisitos Funcionales.

1. Definir nodo como objeto deformable.

1.1. Verificar si un nodo es geometría.

1.2. Tomar la información de la malla del nodo.

1.3. Proporcionar las constantes del método.

2. Crear modelo matemático.

2.1. Obtener lista de vértices de la malla.

2.2. Obtener lista de triángulos de la malla.

3. Crear Rejilla Regular.

3.1. Calcular arista de un cubo.

3.2. Calcular área promedio de los triángulos.

3.3. Calcular coordenadas de los extremos de la malla.

3.4. Calcular cantidad de cubo por eje de coordenada.

3.5. Generar Rejilla.

3.6. Asignar vértices a nodos.

4. Aplicar presión sobre la malla.

5. Establecer parámetros de deformación.

6. Calcular deformación.

6.1. Calcular restricciones de deformación.

6.2. Determinar nodos afectados.

7. Actualizar la malla.

7.1. Recorrer lista de nodos.

7.2. Actualizar posición de los vértices.

8. Restablecer malla.

8.1. Recorrer lista de nodos.

8.2. Restablecer posición de los vértices.

3.3.2 Requisitos No Funcionales.

Usabilidad: Los usuarios del sistema serán programadores con conocimientos básicos de programación gráfica.

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

Soporte: En una versión inicial deberá ser compatible con la plataforma Windows, pero debe estar preparado para que con rápidas modificaciones pueda migrar para Linux.

Hardware: Se necesita una PC Pentium 4 con 1 GB de RAM como mínimo.

Diseño e implementación: Debe utilizar transparentemente la biblioteca gráfica OpenGL y DirectX, y ser adaptable a trabajar con otras bibliotecas. Se harán llamadas a dichas bibliotecas desde el lenguaje C/C++. Se regirá por la filosofía de Programación Orientada a Objetos.

3.4 Modelo de Caso de Uso del Sistema.

A continuación se muestran los posibles actores del sistema a desarrollar, además se describe bajo la forma de acciones y reacciones el comportamiento del sistema, y su interacción con el usuario final.

También, se seleccionan los casos de uso correspondientes al primer ciclo de desarrollo para hacerles sus especificaciones textuales en formato expandido.

3.4.1 Actores del Sistema.

Tabla 1 Actores del Sistema.

Actores	Justificación
Kheipros	Nombre del Simulador Quirúrgico que se beneficiará con las funcionalidades del módulo.

3.4.2 Casos de Uso del Sistema.

Tabla 2 CU1 Gestionar Deformación.

CU1	Gestionar Deformación
Actor	Kheipros
Descripción	Generar la deformación que sufrirá el objeto.
Referencia	R4, R5, R6, R7

Tabla 3 CU2 Generar Rejilla Regular.

CU2	Generar Rejilla Regular
Actor	CU1
Descripción	Crea la rejilla regular, a partir de los datos calculados.
Referencia	R3, CU1 (*include*)

Tabla 4 CU3 Reconstruir Malla.

CU3	Reconstruir Malla.
Actor	CU1
Descripción	Reconstruye la malla, llevándola a su forma inicial.
Referencia	R8, CU1 (*include*)

3.4.3 Diagrama de Casos de Uso del Sistema.

El siguiente diagrama muestra la relación entre los casos de uso del sistema y el actor.

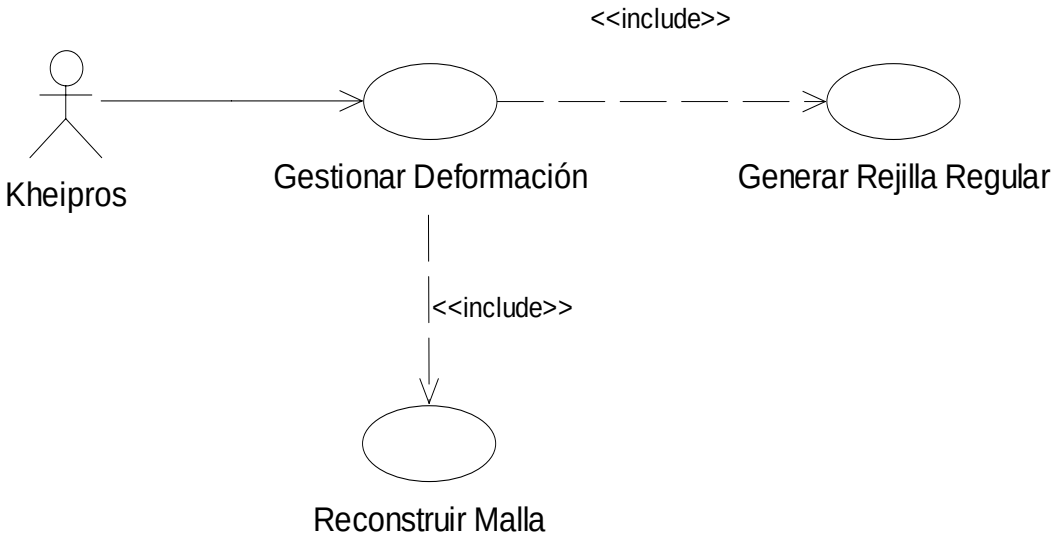

Fig. 15 Diagrama de Casos de Uso del Sistema

3.4.4 Expansión de los Casos de Uso.

Tabla 5 Expansión CU1.

Caso de Uso		
Nombre	Gestionar Deformación	
Actores	Kheipros	
Propósito	Generar la deformación del objeto.	
Resumen	El caso de uso se inicia cuando el actor demanda una nueva deformación.	
Referencias	R4, R5, R6, R7	
Precondiciones	Que se haya creado la Rejilla Regular.	
Poscondiciones	La malla debe quedar deformada.	
Curso Normal de los Eventos		
Acción del Actor		**Respuesta del Sistema**
1- Verifica que se genere una nueva deformación.		
		2- Toma los datos necesarios de la malla
		3- Se llama al CU Generar Rejilla.
		4- Se adicionan los nodos de la rejilla a la lista de nodos
		5- Se inicializan los parámetros de deformación.
		6- Se recorre la lista lineal de nodos
		7- Para cada nodo se chequea sus parámetros de deformación y sus vecinos.
		8- Se determinan los nodos que serán afectados por la deformación, de acuerdo a los parámetros de deformación.
		9- Se recorre la lista de vértices de los nodos chequeados.
		10- Se actualiza la posición de los vértices. Termina el Caso de Uso.
Curso Alternativo:		
		2- No se creó la rejilla regular.
		3- No continúa el CU.
Prioridad:	Crítico.	

Tabla 6 Expansión CU2.

Caso de Uso		
Nombre	Generar Rejilla	
Actores	CU1	
Propósito	Tomar la información de los vértices de la malla y generar la rejilla.	
Resumen	El caso de uso se inicia cuando se ha demandado una deformación, el caso de uso se encarga de construir la rejilla regular.	
Referencias	R3, CU1 (*include*)	
Precondiciones	Que se haya realizado la demanda de una deformación.	
Poscondiciones	Debe quedar creada la rejilla.	
Curso Normal de los Eventos		
Acción del Actor		**Respuesta del Sistema**
1- Verifica la creación de una nueva rejilla regular.		2- Calcula el Área Promedio de la lista de triángulos.
		3- Calcula el valor de la arista de los nodos.
		4- Calcula las coordenadas de los extremos de la malla.
		5- Calcula la cantidad de nodos que tendrá la rejilla regular por eje de coordenada.
		6- Forma la rejilla regular.
		7- Asigna los vértices que forman la malla a los nodos de la rejilla regular. Termina el caso de uso
Curso Alternativo:		
		2- No se demandó crear una rejilla regular.
		3- No continúa el CU.
Prioridad:	Crítico.	

Tabla 7 Expansión CU3.

Caso de Uso		
Nombre	Reconstruir Malla	
Actores	CU1	
Propósito	Llevar la malla a su forma original después de que se haya deformado.	
Resumen	El caso de uso se inicia cuando se ha generado una deformación. El caso de uso se encarga de reconstruir la malla.	
Referencias	R8, CU1 (*include*)	
Precondiciones	Que se haya producido una deformación en la malla.	
Poscondiciones	La malla debe quedar en su forma original.	
Curso Normal de los Eventos		
Acción del Actor		**Respuesta del Sistema**
		1- Verifica si hubo una deformación.
		2- Recorre la lista lineal de nodos.
		3- Para cada nodo chequea sus vecinos y sus parámetros de deformación.
		4- Para cada nodo recorre su lista de vértices.
		5- Actualiza la posición de los vértices de los nodos.
		6- El caso de uso culmina cuando se actualiza la malla.
Curso Alternativo:		
		2- No existió deformación.
		3- No continúa el CU.
Prioridad:	Crítico.	

Conclusiones.

Con la conclusión de este capítulo quedaron definidas las funcionalidades del sistema a partir de la recogida de los requisitos funcionales y no funcionales más importantes, a los que la aplicación debe responder para dar un total cumplimiento a los objetivos propuestos.

4

Capítulo 4 Diseño e implementación del Sistema.

Introducción.

Después de tener definidos los requisitos funcionales y no funcionales del sistema, solo quedaría traducir estos requisitos a una especificación que describa cómo implementar el sistema, y esto se logrará con la realización del diagrama de clases de diseño, el diagrama de componentes de implementación y los diagramas de secuencia.

4.1 Características de la Herramienta SceneToolkit.

Para poder comprender bien el sistema será necesario el estudio de algunas de las características de la herramienta STK. Una escena de esta herramienta está compuesta por objetos de diferentes tipos, contenidos todos en nodos del grafo escena, que es la estructura encargada del manejo de los objetos del ambiente a simular, ya sea cámara, luces, objetos estáticos, dinámicos, deformables, animados, etc. Los nodos de la escena están especializados según su contenido en “nodos geometría”, “nodos cámara”, “nodos luz”, “nodos grupo”, “nodos hueso”, etc. Para su actualización, cada nodo almacena la información necesaria en forma de estados, ejemplo: estado geométrico global, que almacena la información relativa a la posición orientación y escala del nodo respecto a la escena.

Es importante analizar las clases “CGeometryNode” de la STK que es la encargada de almacenar las geometrías de los objetos en forma de mallas triangulares o “CTriMesh” necesaria para la construcción del objeto a deformar, y la clase “CController” de quien heredará el controlador de deformación encargado de alterar el estado geométrico del objeto “CTriMesh”.

4.2 Modelo del Sistema.

Para una mejor comprensión del sistema en cuestión, y por la complejidad del Diagrama de Clases de Diseño se han agrupado las clases en paquetes. El paquete SceneToolkit encapsula algunas clases que ya están implementadas, y aunque no se tomaran en cuenta para el desarrollo de los artefactos del sistema en construcción, su representación permite comprender mejor la arquitectura del sistema y su relación con la STK. El paquete ChainMail contiene las clases encargadas de generar la deformación.

4.2.1 Diagrama de Paquetes.

En el siguiente diagrama se muestra de manera general como queda estructurado el sistema mediante los paquetes SceneToolkit y ChainMail.

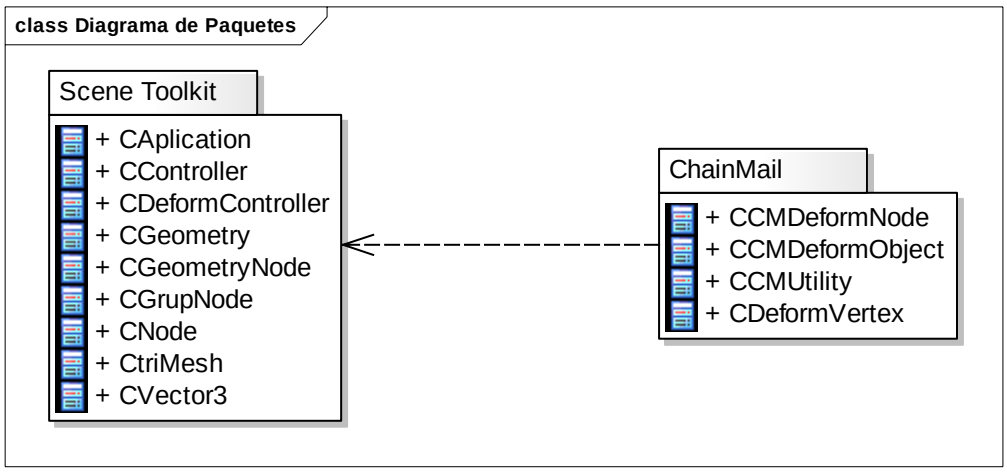

Fig. 16 Diagrama de Paquetes.

En el siguiente diagrama se muestran las clases y las relaciones por las que está compuesto el paquete SceneToolkit.

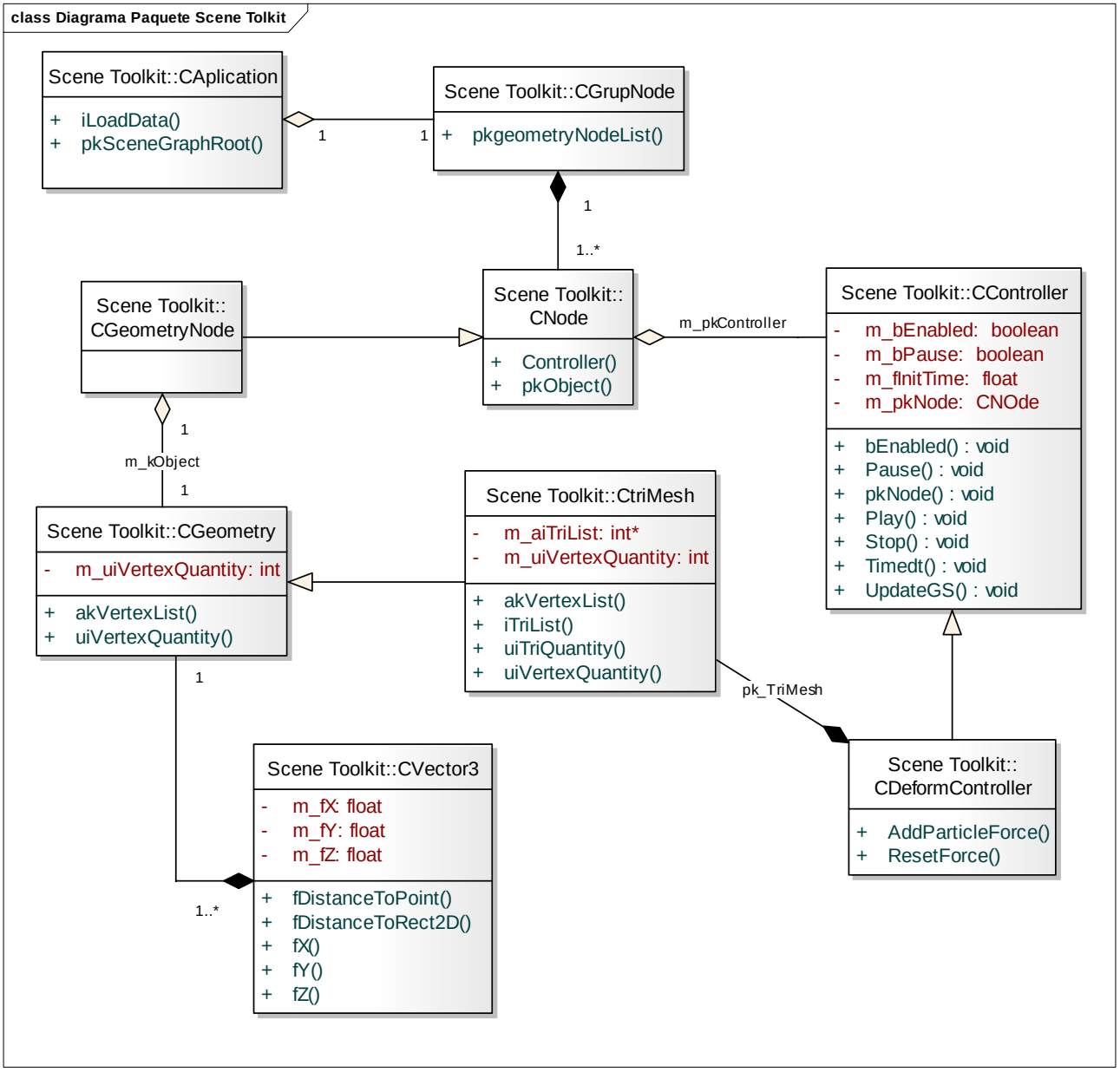

Fig. 17 Diagrama de Clases del Paquete SceneToolkit.

El siguiente diagrama muestra las clases y las relaciones por las que está compuesto el paquete ChainMail.

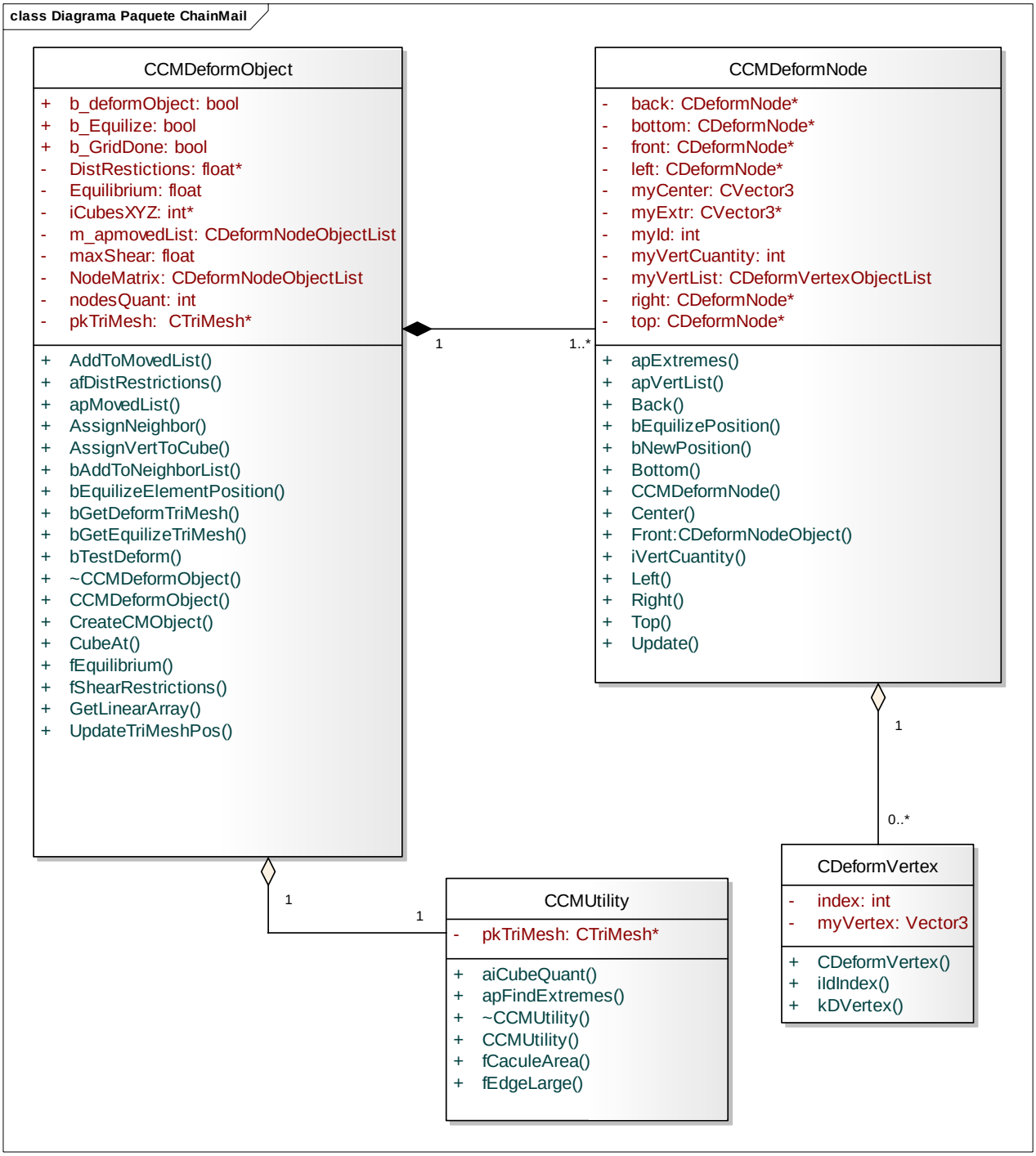

Fig. 18 Diagrama de Clases del Paquete ChainMail.

Es válido destacar que la clase de este diagrama que tiene por nombre CCMDeformObject tiene una relación de herencia con la clase CDeformController del paquete SceneToolkit.

El siguiente diagrama muestra de manera general como quedan estructuradas las clases del paquete ChainMail y su relación con el paquete SceneToolkit.

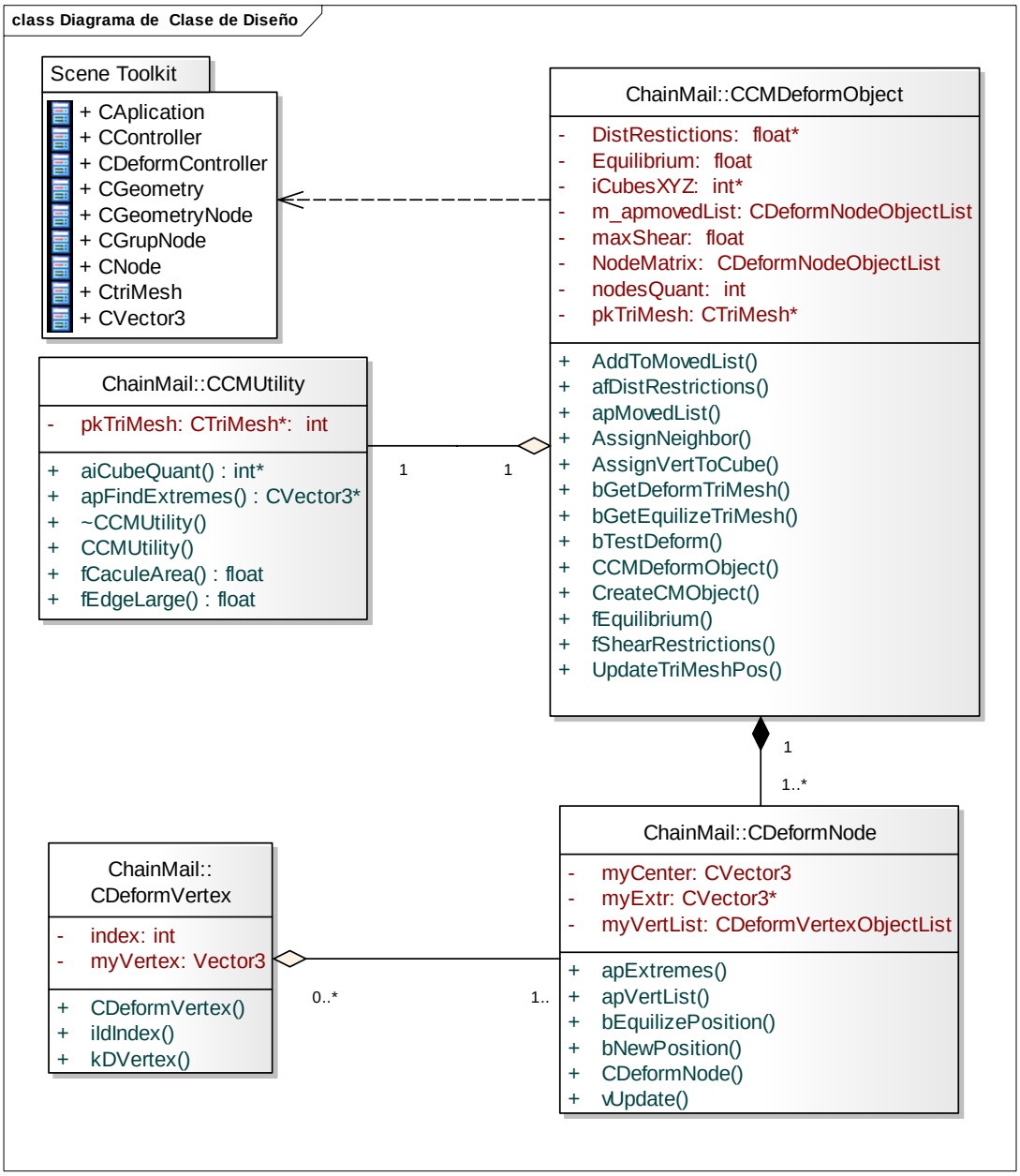

Fig. 19 Diagrama General de Clase de Diseño.

4.3 Descripción de las clases del Diseño

A continuación se realizará la descripción de las clases de diseños que forman parte del sistema que se desarrolla, se especificarán y describirán los atributos y las operaciones que contienen cada una de las clases para un mayor entendimiento del sistema.

Tabla 8 Descripción de la clase CCMDeformObject.

Nombre: CCMDeformObject	
Tipo de clase: Controladora	
Atributo	**Tipo**
pkTriMesh	CTriMesh*
m_apmovedList	CDeformNodeObjectList
NodeMatrix	CDeformNodeObjectList
iCubesXYZ	int*
nodesQuant	int
DistRestictions	float*
maxShear	float
Equilibrium	float
b_GridDone	float
b_Equilize	float
b_deformObject	float
Para cada responsabilidad:	
Nombre:	CCMDeformObject()
Descripción:	Constructor de la clase, encargado de inicializar los atributos pkTriMesh, iCubesXYZ, Equilibrium, y el nodesQuant.
Nombre:	~CCMDeformObject()
Descripción:	Destructor de la clase.
Nombre:	CreateCMObject()

Descripción:	Función que a partir de cantidad de cubos por ejes de coordenadas, de los extremos mínimo y máximo de la malla y de el valor de la arista de los cubo, va creando los nodos y, adicionándolos al arreglo tridimensional NodeMatrix de tal forma que quede creada la rejilla regular.
Nombre:	AssignVertToCube()
Descripción:	Función encargada de asignarle los vértices del objeto TriMesh a los nodos. Le resta el valor de la posición del extremo mínimo a los valores de los vértices en cada eje de coordenada y dividiendo este resultado entre el valor de la arista del cubo, se obtiene la posición en la Matriz Tridimensional del nodo al que se le asignará el vértice.
Nombre:	AssignNeighbor()
Descripción:	Función responsable de asignarle a cada nodo su vecino correspondiente en la rejilla regular.
Nombre:	GetLinearArray()
Descripción:	Función encargada de convertir la Matriz Tridimensional a un arreglo lineal y retornar dicho arreglo.
Nombre:	iCubes()
Descripción:	Función encargada de devolver la cantidad de cubos por cada uno de los ejes de coordenadas.
Nombre:	afDistRestrictions()
Descripción:	Función encargada de acceder a los valores de las restricciones de deformación.
Nombre:	fShearRestrictions()
Descripción:	Función encargada de acceder a maxShear.
Nombre:	fEquilibrium()
Descripción:	Función encargada de acceder al miembro Equilibrium.
Nombre:	iNodes()
Descripción:	Función encargada de acceder a nodesQuant, cantidad total de nodos.
Nombre:	CubeAt()
Descripción:	Función encargada de obtener y devolver un nodo dado una

	posición en la Matriz Tridimensional
Nombre:	bTestDeform()
Descripción:	Función que se encarga de deformar el nodo seleccionado y de procesar la listas de los vecinos que son afectados por la deformación.
Nombre:	UpdateTriMeshPos()
Descripción:	Función encargada de actualizar los vértices del objeto TriMesh.
Nombre:	apMovedList()
Descripción:	Función encargada de retornar la lista de nodos que fueron movidos.
Nombre:	AddToMovedList()
Descripción:	Función encargada de adicionar un nodo a la lista de elementos movidos.
Nombre:	bAddToNeighborList()
Descripción:	Función encargada de adicionar un nodo a la lista de vecinos correspondiente si este no ha sido chequeado aún, si lo adicionó retorna 1, de lo contrario devuelve 0.
Nombre:	bGetDeformTriMesh()
Descripción:	Función responsable de actualizar el TriMesh después de haberse deformado el objeto.
Nombre:	bEquilizeElementPosition()
Descripción:	Función que después que ocurrió la deformación se encarga de que para cada nodo equilibre su posición con respecto a sus vecinos.
Nombre:	bGetEquilizeTriMesh()
Descripción:	Función responsable de actualizar la lista de vértices del TriMesh a partir de los vértices del objeto equilibrado.

Tabla 9 Descripción la clase CDeformNode.

Nombre: CDeformNode	
Tipo de clase: Entidad	
Atributo	**Tipo**
myExtr	CVector3*
myVertList	CDeformVertexObjectList
myVertCuantity	int
myId	int
myCenter	CVector3
left	CDeformNode*
right	CDeformNode*
bottom	CDeformNode*
top	CDeformNode*
front	CDeformNode*
back	CDeformNode*
Para cada responsabilidad:	
Nombre:	CDeformNode()
Descripción:	Constructor de la clase, encargado de inicializar myExtr, los vecinos left, right, bottom, top, front y back, y la lista myVertList.
Nombre:	apExtremes()
Descripción:	Función encargada de acceder al miembro myExtr, extremos del nodo.
Nombre:	apVertList()
Descripción:	Función encargada de acceder al miembro myVertList, lista de vértices.
Nombre:	Center()
Descripción:	Función encargada de acceder a myCenter, valores de coordenadas del centro del nodo.
Nombre:	Right()
Descripción:	Función encargada de acceder al miembro right, vecino derecho del nodo.

Nombre:	Left()
Descripción:	Función encargada de acceder al miembro left, vecino izquierdo del nodo.
Nombre:	Top()
Descripción:	Función encargada de acceder al miembro top, vecino superior del nodo.
Nombre:	Bottom()
Descripción:	Función encargada de acceder al miembro bottom, vecino inferior del nodo.
Nombre:	Front()
Descripción:	Función encargada de acceder al miembro front, vecino anterior del nodo.
Nombre:	Back()
Descripción:	Función encargada de acceder al miembro back, vecino posterior del nodo.
Nombre:	bNewPosition()
Descripción:	Función que en el proceso de deformación controla si las restricciones de deformación del nodo fueron violadas y calcula la nueva posición del mismo.
Nombre:	bEquilizePosition()
Descripción:	Función que en el proceso de relajación controla si el nodo esta comprimido o estirado y calcula la nueva posición del mismo.
Nombre:	vUpdate()
Descripción:	Función encargada de actualizar los valores de los extremos mínimos y máximo del nodo y los valores de los vértices que contiene el nodo, en el proceso de deformación y relajación.
Nombre:	calculeCenter()
Descripción:	Función que a través de los extremos de coordenadas mínimos y máximos del nodo calcula la posición del centro del nodo.

Tabla 10 Descripción la clase CCMUtility

Nombre: CCMUtility	
Tipo de clase: Entidad	
Atributo	**Tipo**
pkTriMesh	CTriMesh*
Para cada responsabilidad:	
Nombre:	CCMUtility()
Descripción:	Constructor de la clase encargado de inicializar el pkTriMesh.
Nombre:	fCaculeArea()
Descripción:	Función encargada de calcular el área promedio de los triángulos que forman la malla, para cada triangulo le calcula su área y luego las promedia.
Nombre:	fEdgeLarge()
Descripción:	Función encargada de calcular el valor de la arista de los cubos que formarán la rejilla regular, se iguala el área promedio de los triángulos al área del cubo y se calcula la arista del cubo.
Nombre:	apFindExtremes()
Descripción:	Función encargada de determinar y devolver los valores del os extremos de cada eje de coordenada, para cada eje busca los valores mínimos y máximos.
Nombre:	aiCubeQuant()
Descripción:	Función encargada de calcular la cantidad total de cubo que llevara la rejilla regular en cada eje de coordenada. Con los valores de las coordenadas mínimas y máximas de los eje obtiene el tamaño de cada eje y luego lo divide entre el valor de la arista de los cubos e incrementa en uno el resultado final.

Tabla 11 Descripción la clase CDeformVertex

Nombre: CDeformObject	
Tipo de clase: Entidad	
Atributo	**Tipo**
index	int
myVertex	CVector3
Para cada responsabilidad:	
Nombre:	CDeformVertex()
Descripción:	Constructor de la clase encargado de inicializar las variables index y myVertex.
Nombre:	iIdIndex()
Descripción:	Función encargada de acceder a la variable index, índice de la posición del vértice en la lista de vértices del TriMesh.
Nombre:	kDVertex()
Descripción:	Función encargada de acceder a la variable myVertex, vértice que corresponde a la posición del index en lista de vértices del TriMesh.

4.4 Diagramas de Secuencia:

Con el objetivo de tener una clara visión del flujo de control del sistema que se desarrolla y conocer el comportamiento dinámico del mismo, se muestran a continuación los Diagramas de Secuencia correspondiente a cada uno de los casos de usos ya seleccionados.

El siguiente Diagrama de Secuencia muestra la interacción entre los objetos correspondientes al caso de uso Gestionar Deformación.

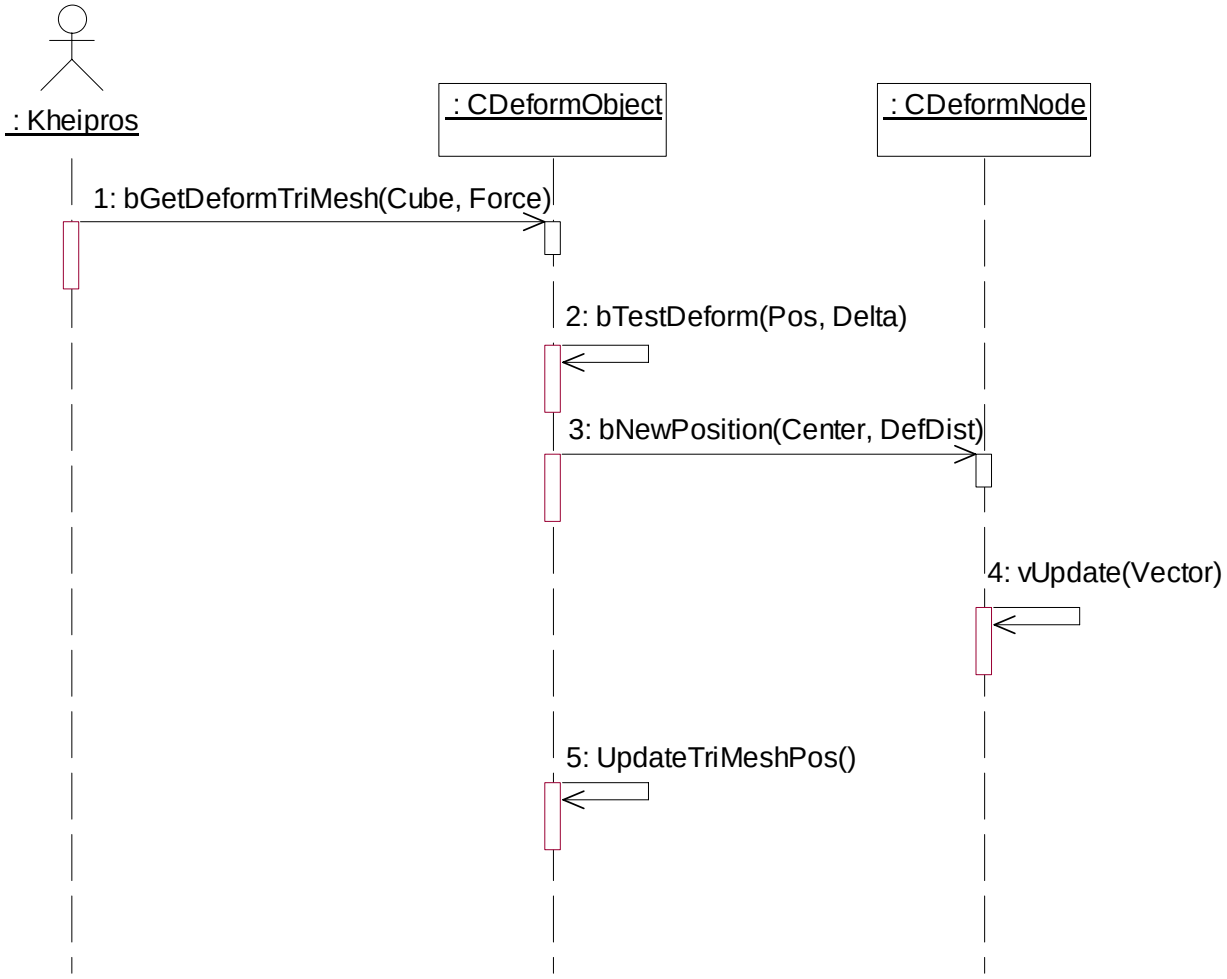

Fig. 20 Diagrama de Secuencia CU Gestionar Deformación.

El siguiente Diagrama de Secuencia muestra la interacción entre los objetos correspondientes al caso de uso Crear Rejilla.

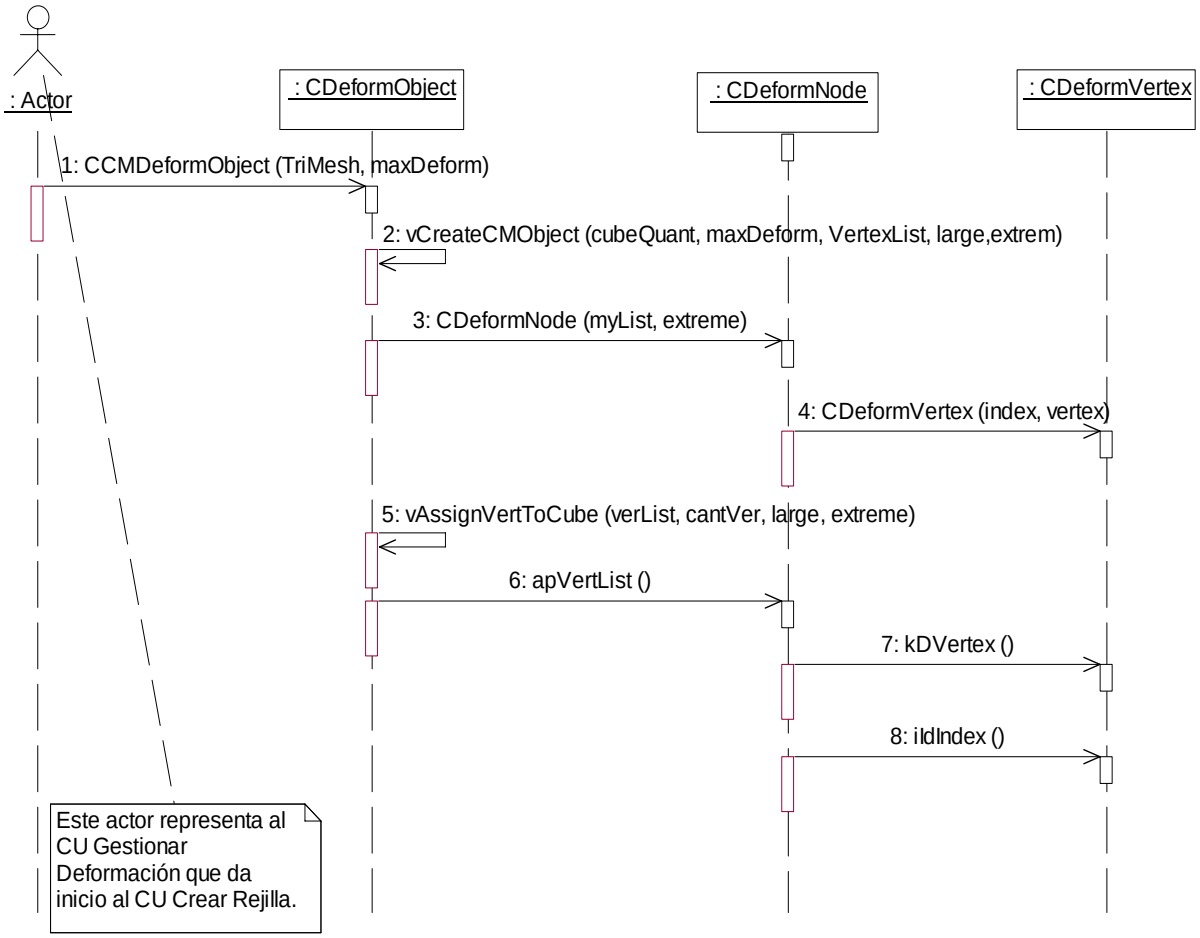

Fig. 21 Diagrama de Secuencia CU Crear Rejilla.

El siguiente Diagrama de Secuencia muestra la interacción entre los objetos correspondientes al caso de uso Reconstruir Malla.

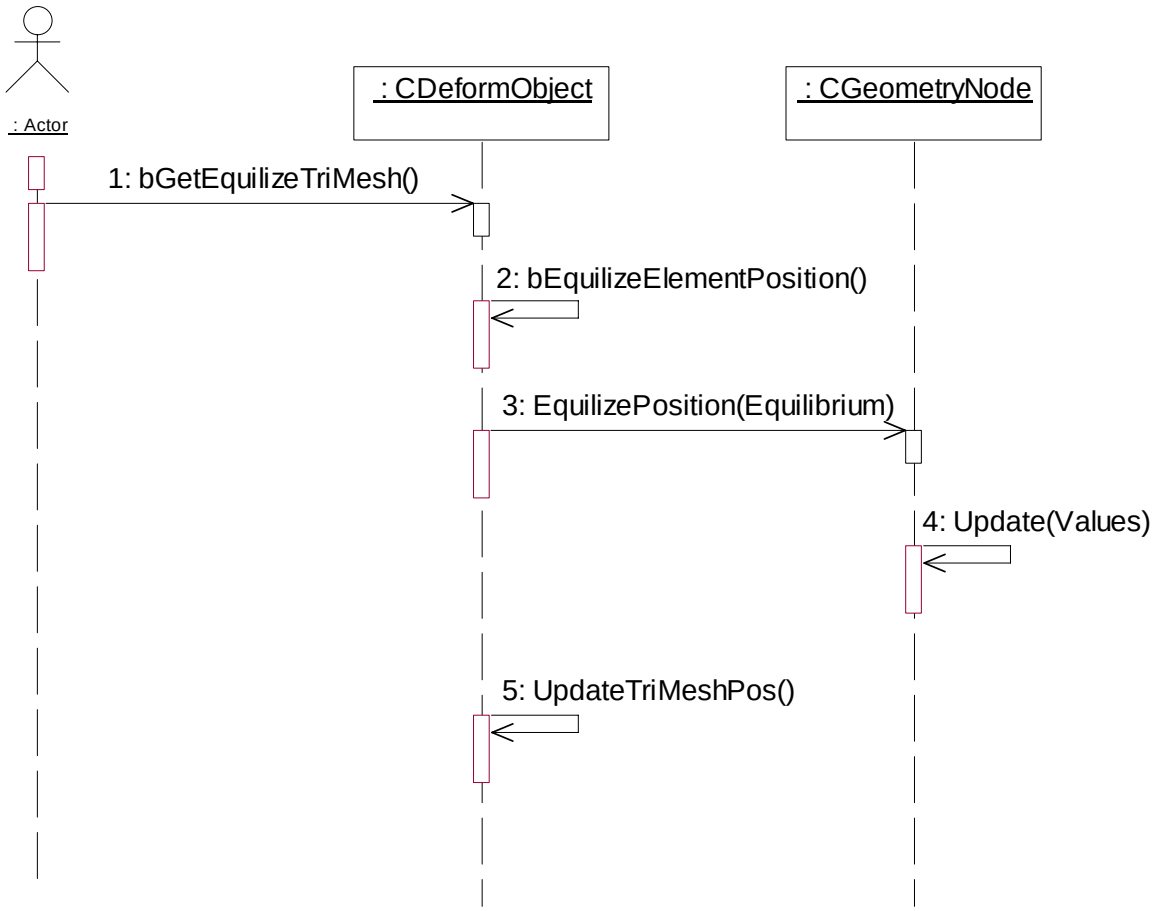

Fig. 22 Diagrama de Secuencia CU Reconstruir Malla.

4.5 Diagrama de Componentes.

Con todo lo visto anteriormente, solo quedaría tener una vista estática del sistema, para poder tener una idea de la organización y las dependencias entre los conjuntos de componentes. Para ello a continuación de muestran los diagramas de componentes.

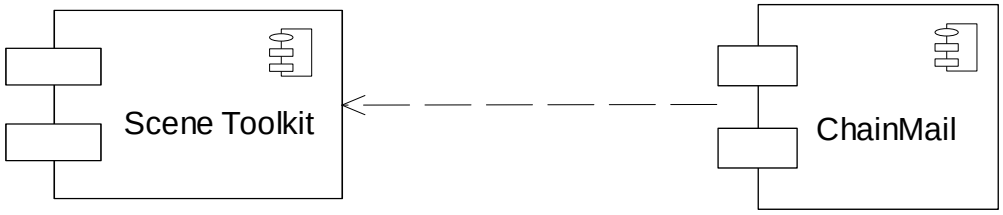

Fig. 23 Diagrama de Relación entre Paquetes.

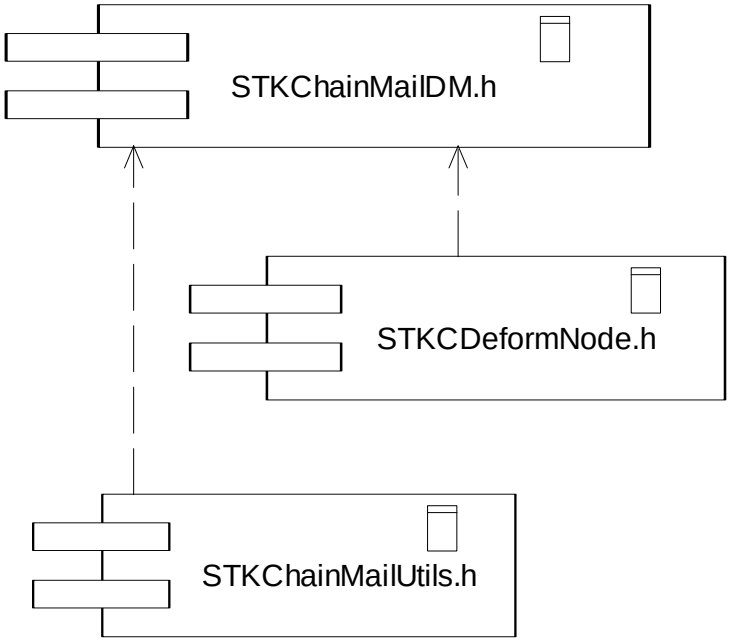

Fig. 24 Diagrama de Componentes.

4.6 Resultados.

Finalmente se realizó una comparación en igualdades de condiciones entre el módulo de deformación existente, que está basado en el método de Masa Resorte y el módulo de deformación obtenido; arrojando como resultado que el módulo obtenido supera en velocidad de render al existente, como se muestra a continuación.

Tabla 12 Comparación entre Técnicas de Deformación.

Técnica	Cantidad de Polígonos	Prestaciones de la PC.	Velocidad de Render
3D ChainMail	8170	Intel PIV, 3.0GHz,1.0 GB RAM	44.0 fps
		Intel PIV, 3.0GHz, 1.0 GB RAM, NVIDIA Quadro FX 500/FX 600.	75.5 fps
Sistema Masa- Resorte	8170	Intel PIV, 3.0GHz,1.0 GB RAM	28.5 fps
		Intel PIV, 3.0GHz, 1.0 GB RAM, NVIDIA Quadro FX 500/FX 600.	45.0 fps

Conclusiones.

Al terminar este capítulo queda definido el diseño completo del sistema, y la secuenciación de pasos traducida a mensajes entre clases de los casos de usos a desarrollar en el primer ciclo de vida del software. La estructura del sistema quedo descrita con el diagrama de clases de diseño así como la interacción del conjunto de objetos que participan en el desarrollo del sistema fueron mostrada con los diagrama de secuencia.

Conclusiones Generales.

A lo largo de este trabajo se realizó un estudio de las diferentes técnicas y algoritmos de deformación de cuerpos para poder dar solución a los objetivos planteados, en correspondencia con las exigencias del cliente. A partir de esta investigación se elabora la propuesta de solución al problema científico planteado.

Posteriormente se realizó el proceso de Ingeniera de Software, donde se describió la solución propuesta en términos de los flujos de trabajo de requerimientos, diseño e implementación, ajustándose a la metodología de desarrollo de software utilizada; que en este caso fue el Proceso Unificado del Software (RUP).

Finalmente se obtuvo un módulo de deformación funcional, que se ajusta a los objetivos planteados y que será acoplado a la herramienta SceneToolkit.

Recomendaciones

Para dar continuidad a este trabajo y complementar su funcionalidad, recomendamos el estudio y posible incorporación de propiedades físicas al módulo obtenido en pos de lograr un mayor realismo en las deformaciones cuidando su principal virtud que sería la velocidad de Render. Además analizar a fondo las demás variantes existentes del Algoritmo ChainMail para posteriores mejoras al módulo obtenido.

Referencias Bibliográficas.

[1] Ramirez R. Bretau O. “Deformación de Objetos para Sistema de Realidad Virtual”, UCI, 2007.

[2] Dawson S.L., Kaufman J.A., “*The Imperative for Medical Simulation”*, Proc. IEEE, 86 (3), pp. 479-483, 1998.

[3] Álvarez Clemente “Simulador quirúrgico con sentido del tacto http://www.elpais.com/articulo/futuro/Simulador/quirurgico/sentido/tacto/elpepusocfut/20070711elpepifut_5/Tes

[4] S.Cotin, H.Delingete, N. Ayache, *Real Time Volumetric Deformable Models for Surgery Simulation,* Visualization in Biomedical Computing (Proc. VBC ’96), K.H. Höhne, R. Kikinis (eds.), Lecture Notes in Computer Scince, vol. 1131, Springer-Verlag, 1996

[5] Downes, M.: Hsu, A. A virtual environmet for training laparoscopic cholecystectomy. University of California, Berkeley. CS294-5 Virtual Reality, Spring Semester, 1997.

[6] Playter, R., B.Blank, N. Cornelius. *Integral haptics Applications: Surgical Anastomosis and Aircraft Maintenance,* in Preprints of The First Phantom User’s Group Workshop, Sept. 27-30, Cambridge, MA, 1996.

[7] Gibson, S., Samosky,J., Mor,A. Simulating Arthroscopic Knee Surgery Using Volumetric Object Rendering with Real-Time Volume Rendering, in CVMRMed-MRCAS’97, pp. 369-378, Springer-Verlag, 1997.

[8] M. Downes et al., Virtual Environments for Training Critical Skills in Laparoscopic Surgery, Proc. Medicine Meets Virtual Reality (MMVR: 6), J.D. Westwood et al. (eds.), Studies in Health Technology and Informatics, 50, pp. 316-322, IOS Press, Amsterdam, 1998

[9] Terzopoulos D, Platt J, Barr A, and Fleischer K. “Elastically deformable models.” July 1987. Proceedings of SIGGRAPH’87 (Anaheim, California).

[10] Geiger, B.; Kikinis, R. *Simulation of endoscopy.* In Ayache, N. (ed), Proceedings of CVRMed’95, pp.277-281. Springer-Verlag, Berlin.

[11] Cover, S.A.; *Interactively deformable models for surgery simularion,* IEEE Computer Graphics & Applications, pp. 68-75, Nov. 1993.

[12] Shewchuk, J. (1999), Lectures notes on Delaunay mesh generation, Technical report, Department of Electrical Engineering and Compurter Science, University of California at Berkeley.

[13] Campbell, R. y Flynn, P. (2001), ‘A survey of free-form objects representation and recognition techniques’, Comput. Vis. Image Underst. 81(2), 166–210.

[14] Nachiappan S. Kapoor S. Prem K. “Geometry Based Connectivity Compression of Triangular Meshes” Illinois Institute of Technology, USA 2006.

[15]D. Terzopoulos, K. Waters (1990). Physically-Based Facial Modeling, Analysis, and Animation. The Journal of Visualization and Computer Animation, 1:73,80.

[16] Villard, Julien; Borouchaki, Houman(2005). Adaptive Meshing For Cloth Animation.

[17] Kühnapfel U. (2000) “Endoscopy Surgery Training Using Virtual Reality and Deformable Tissue Simulation.” Karlsruhe, Alemania.

[18] Webster, Roger. (2002) “Elastically Deformable 3D Organs for Haptics Surgical Simulation.”

[19] Vassilev T, Spanlang B.(2001) “A Mass-Spring Model for Real Time Deformable Solids”.

[20] Levoy, M. and Whitted, T. “The Use of Points as a Display Primitive,” *Technical Report TR 85-022*, University of North Carolina at Chapel Hill, 1985.

[21] Grossman, J. and Dally, W. “Point Sample Rendering,” *Proc. Eurographics Rendering Workshop*, 1998.

[22] AnimaTek International, Inc., “Caviar Technology,” Web page: http://www.animatek.com/products_caviar.htm

[23] Sederberg, T.W., Parry, S.R. *Free-form deformation of solid geometric models*. Siggraph'86. Computer Graphics 20 (4), 151-196, 1986.

[24] Griessmair J., Purgathofer W. *Deformation of solids with trivariate B-spline.* Proceedings of Eurographics'89, 137-148, 1986.

[25] Coquillart, S. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. Computer Graphics 24 (4), 187-196, 1990.

[26] Lamousin, H.J., Waggenspack, W.N. *NURBS-Based Free-Form Deformations.* IEEE Computer Graphics & App, 59-65, 1994.

[27] Bechmann, D. *Space Deformation Models Survey.* Computers & Graphics 18 (4), 571-586, 1994.

[28] Borrel, P., Bechmann, D. *Deformation of n-dimensional objects.* Journal Computational Geometry Apply 1 (4), 427-453, 1991.

[29] Hu, S., Zhang, H., Tai, Chiew, Sun, J. *Direct manipulation of FFD: efficient explicit solutions and decomposible multiple point constraints. The Visual Computer 17, 370-379, 2001.*

[30] Drägner Ch. “A ChainMail Algorithm for Direct Volumen Deformation in Virtual Endoscopic Simulation”. Vienna University of Technology. Mayo 2005.

[31] Sarah F. F. Gibson, 3D ChainMail: a fast algorithm for deforming volumetric objects. In Michael Cohen and David Zeltzer, editors, 1997 Symposium on Interactive 3D Graphics, pages 149-154. ACM SIGGRAPH, April 1997. ISBN 0-89791-884-3.

[32] Web-Based Surgical Simulators and Medical Education Tools.http://synaptic.mvc.mcc.ac.uk/simulators.html

[33] M. A. Schill, S. F. F. Gibson, H.-J. Bender, R. Manner (1998). Biomechanical Simulation of the Vitreous Humor in the Eye Using Enhanced ChainMail Algorithm. Proceedings of Medical Image Computation and Computer Assisted. Interventions (MICCAI) 1998, S. 679{687.

[34] K. Brodlie Y. Li (2003). Soft Object Modeling with Generalized ChainMail Extending the Boundaries of Web-based Graphics. Proceedings of Computer. Graphics Forum, 22(4):717{727.

[35] Nedel, L. P., And Thalmann (1998), D., Real Time Muscle Deformations using Mass-spring Systems, Proceedings of the Computer Graphics International.

[36] Hong Min, Jung (2005), Fast volume preservation for realistic muscle deformation. International Conference on Computer Graphics and Interactive Techniques.

[37] Gibson S. F. F., Mirtich B. “A Survey of Deformable Modeling in Computer Graphics” A Mitsubishi Electric Research Lab, Noviembre 1997.

[38] Tait R.J., Schaefer G., Kühnapfel U., Çakmak H.K. “Interactive Spline Modeling of Human Organs for Surgical Simulators” School of Computing and Mathematics, The Nottingham Trent University, U.K. Institut für Angewandte Informatik, Forschungszentrum Karlsruhe, Germany

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

[40] D. Hearn, M. P. Baker (1997). Computer Graphics C Version. Prentice Hall.

[41] A. Witkin (2001). SIGGRAPH Coursenotes - Physically Based Modeling, Particle System Dynamics. Pixar Animation Studios.

[42]M. Nakao, T. Kuroda, H. Oyama, M. Komori, T. Matuda, T. Takahashi (2003). Physically-Based Fine and Interactive Soft Tissue Cutting. IPSJ JOURNAL4, 44(8).

[43] B. Eberhardt, O. Etzmuss, M. Hauth (2000). Implicit-Explicit Schemes for Fast Animation with Particle Systems. In Eurographics Computer Animation and Simulation Workshop 2000, 2000.

[44] Swedlow, J. L. and Cruse, T. A. Formulation of the boundary integral equation for three-dimensional elastoplastic flow, International Journal of Solids and Structures, 7, 1673-1681 (1971).

[45] Riccardella, P. An Implementation of the Boundary Integral Technique for plane problems of Elasticity and Elastoplasticity, PhD Thesis, Carnegie Mellon University, Pitsburg, PA (1973).

[46] Banerjee, P. K. and MUSTOE, G. C. W. The boundary element method for two-dimensional problems of elastoplasticity. Recent Advances in Boundary Element Methods, C. Brebbia (ed.), Pentech Press, Plymouth, Devon, UK, 283-300 (1978).

[47] Mukherjee, S. A. Boundary Element Methods in Creep and Fracture. Applied Science Publishers LTD, Barking Essex, England (1982)..

[48] Kumar, V. and Mukherjee, S. A boundary-integral equation formulation for time-dependent inelastic deformation in metals, International Journal of Mechanical Sciences, 19, 713-724 (1977).

[49] Bro-Nielsen M. Cotin S. Real-time volumetric deformable models for surgery Simulations using finite elements and condensation.Computer Graphics Forum, 1996.

[50] Irving G. Teran J. Fedkiw R. Invertible finite elements for robust simulations of large deformations. Siggraph, 2004.

[51] Germán Sánchez Torres, Sandra P. Mateus Santiago y John William Branch. “Construcción de Mallas Triangulares no Estructuradas Aplicado al Ajuste de Superficies de Objetos Tridimensionales".

[52] Mark Pauly, Richard Keiser, Leif P. Kobbelt, Markus Gross. Shape Modeling with Point-Sampled Geometry.

[53] Bernd Hamann, DonhuaWu, and Robert J. Moorhead II. On particle path generation based on quadrilinear interpolation and Bernstein-B´ezier polynomials. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):210–217, 1995.

[54] Eduardo Frías Valero. El Método de los Elementos Finitos,Departamento de Ingenieria Electrica UPC, 2004.

[55] Holbrey R. P.. “Virtual Suturing for Training in Vascular Surgery” School of Computing, University of Leeds. Mayo 2005.

[56] McDonough James “Últimos avances y aplicaciones en realidades virtuales” http://www.campusred.net/telos/anteriores/num_032/actuali_noticias9.html

[57] Pauly M., Keiser R., Kobbelt L. P., GrossM.: Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 641–650.

[58]Zwicker M., Rsnen J., Botsch M., Dachsbacher C., Pauly M.: Perspective accurate splatting. In *Proceedings of Graphics Interface* (2004).

Anexo1: Glosario de Términos

B

Biblioteca de clases: Clases de programación orientada a objetos suministradas por terceros.

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.

Curvas de Bezier: Método matemático de interpolación paramétrica de curvas de superficie.

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.

D

Discretización: Proceso de dividir el objeto en cuestión en pequeñas partes.

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.

F

Fluidos: Sustancia o medio continuo que se deforma seguidamente en el tiempo ante la aplicación de una solicitación o tensión tangencial sin importar la magnitud de ésta.

H

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

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.

Malla Triangular: Conjunto de vértices y aristas que es su composición forman triángulos y estos a su vez la malla.

O

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

P

Paralepípedo: Poliedro de seis caras, cada una de las cuales es un paralelogramo, que son paralelas e iguales dos a dos

Parametrización: Representación de una curva o superficie como imagen de una función vectorial.

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.

Rejilla Regular: Matriz tridimensional formada por cubos iguales.

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.

Software: Equipamiento o soporte lógico de un computador digital, que comprende el conjunto de los componentes necesarios para hacer posible la realización de una tarea específica.

Splines: Método matemático de representación de curvas tanto bidimensionales como de superficie.

T

Tetraedro: Es un poliedro formado por cuatro caras que son triángulos equiláteros, y cuatro vértices en cada uno de los cuales concurren tres caras. Es uno de los cinco poliedros perfectos llamados sólidos platónicos.

Tiempo Real: Termino 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.

Anexo2: Estándares de Codificación

La codificación será guiada por el estándar de codificación propuesto por la herramienta SceneToolkit. Se programará en inglés debido que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático. Se respetarán los estándares de codificación para C++ (indexado, uso de espacios y líneas en blanco, etc.).

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:

Los ficheros *.h y *.cpp de nombrarán de la siguiente forma:

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 de la siguiente manera:

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 iteradores STD: vector<T> NameList

Indicando con “T” el tipo de dato.

Anexo 3: 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 **b**VarName;

int **i**Name;

unsigned int uiName;

float **f**Name;

char **c**Name;

char** **apc**Name; // arreglo de punteros

bool **m_b**MemberVarName; //variable miembro

char **g**cGlobalVarName; //variable global, no se le antepone “”

short **s**Name;

Instancias de tipos creados:

CClassName* akName; //arreglo de objetos

CClassName* akName; // variable miembro de clase

SMyStructure kName;

CClassName kObjectName;

CClassName* pkName; //puntero a objeto

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;

}
```

Índice de figuras.

Índice de tablas.