UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Deformación de Objetos para Sistemas de Realidad Virtual

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

Autores: Osley Bretau Camejo

Raissel Ramírez Orozco

Tutores: Dr. José Ignacio Guzmán Montoto

Ing. Yanoski Román Camacho

Ciudad de la Habana

Mayo del 2007

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:

Osley Bretau Camejo

_____

Raissel Ramírez Orozco

_____

Tutores:

Dr. José Ignacio Guzmán Montoto

_____

Ing. Yanoski Camacho Román

_____

Datos de Contacto:

Nombre y Apellidos: José Ignacio Guzmán Montoto
Edad: 40 años
Ciudadanía: cubano
Institución: SIMPRO
Título: Doctor en Informática
E-mail: joseigm@reduim.cu

Graduado de la CUJAE, en Ingeniería Mecánica, Master en Informática y Doctor en Informática, jefe de investigaciones de SIMPRO.

Nombre y Apellidos: Yanoski Rogelio Camacho Román
Edad: 26 años
Ciudadanía: cubano
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Informática
Categoría Docente: Profesor Instructor
E-mail: rcamacho@uci.cu

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

Dedicatoria:

…a mi familia y amigos por la incondicionalidad.

A la memoria de Ovidio Orozco, mi eterna fuente de inspiración.

Raissel

… a mis amistades, mi familia y en especial a mi hermana Odrys.

Osley

Agradecimientos:

Este trabajo es el resultado del apoyo de muchos amigos que incluso anónimamente han colaborado con su desarrollo. Mencionar todos los nombres sería una lista interminable, pero obviarlos sería una falta de reconocimiento a su esfuerzo. Sinceramente agradecemos la colaboración del Dr. Luis Otero por su preciado tiempo, al Ing. Fernando por sus conocimientos de RUP, a Alejandro Alonso Fuster por su inagotable espíritu optimista, a todos los profesores que nos han guiado en el mundo de la investigación y los gráficos por computadoras, a los integrantes del proyecto SIMPRO por nuestra constante retroalimentación, especialmente a los tutores Yanoski y José Ignacio, a los estudiantes Karel y Arian…

Agradecerles principalmente a mis padres por guiarme en la vida y fundamentalmente en los estudios.

A mis amigos de la universidad Sachel, Raissel, Yaima y Laura, por su constante apoyo docente, espiritual y la vida en general.

A mis amigos de la casa Yosbel y Elizabeth, por apoyarme desde hace más de 10 años en todos los aspectos de la vida.

A Lien, por estar a mi lado en más de 4 años de la carrera y poder encontrar en ella una persona casi perfecta.

A mis compañeros de 1er año, por darme la oportunidad de contar con ellos en cualquier situación, especialmente a Rubén.

Osley

A mis padres, por el amor, la ética y la fuerza de toda la vida.

A toda mi familia, mis abuelas, padrino, más que todo a mis primos, por ser los hermanos que la natura me negó.

A Ivett y Ronnie por absolutamente todo.

A Laura por la alegría y el calor de mis años de universitario.

A mis amigos, una larga lista que no puede obviar a Osley, Eduar, Dashiell, Lien, Sachel y Yaima, Arnaldo…

Raissel

…a todos nuestros agradecimientos más sinceros.

Resumen

Los sistemas de realidad virtual han devenido parte importante en el entrenamiento de las más diversas actividades humanas. Sin embargo, aún hoy el hardware representa 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 y principales técnicas usadas con este propósito, tanto matemáticas como son los Splines, NURBS, curvas Bezier, el algoritmo 3D Chain Mail y la Deformación de Libre Forma, como métodos basados en física, como el Método de Elementos Finitos, Método de Elementos de Frontera y Sistemas Masa-Resorte.

Por último se propone un módulo de software multiplataforma basado en Sistemas Masa-Resorte con amortiguación y conservación de volumen, acoplado a la Scene Toolkit (STK) y desarrollado usando el proceso unificado de software e implementado en C++ estándar.

Palabras claves:

Simulación basada en física, sistemas masa-resorte, simulación de órganos.

Summary

Virtual Reality Systems has become an important part of human training in many human activities Moreover, still today hardware represent an inconvenient for developing complex applications, like surgery simulation. The main goal of this work is offering a solution to the organs deformation problem, especially focused to the organs deformation.

A study has been developed to know the art state and main techniques used in the world with this purpose. Mathematic techniques like Splines, NURBS, Bezier curves, 3D Chain Mail algorithm and Free Form Deformation and physically based methods such as Finite Element Method, Boundary Element Method and Mass-Spring Systems.

Finally, a multiplatform software module is presented, based on Mass-Spring-Damper System and volume conservation. It has been linked to Scene Toolkit and developed using the software unified process implemented in standard C++.

Keywords:

Physically based simulation, mass-spring system, organs simulation.

Contenido

Introducción

Las aplicaciones de la Realidad Virtual han experimentado un vertiginoso salto en los últimos tiempos. Su alcance es hoy mucho mayor que los juegos interactivos, la representación estática de objetos o realización de imágenes y videos verdaderamente asombrosos. Hoy se habla de la simulación de procesos cada vez más complejos, del ahorro de costosos recursos y vidas en el entrenamiento del personal.

Los simuladores han surgido como una efectiva herramienta para el entrenamiento y prueba de las más diversas ramas de la actividad humana. Es ya clásico el ejemplo de los simuladores de vuelo, tiro, conducción…etc. Desde las maquetas digitales que realizan ingenieros para probar la resistencia de sus creaciones, hasta la minuciosa simulación quirúrgica, han evitado que se cometan irremediables errores y han ahorrado sumas incalculables. Sin embargo, estamos aún muy lejos de lograr la efectividad necesaria; “Un reporte del Instituto de Medicina en el 2002 indica que más de 44,000 personas mueren por errores médicos en los Estados Unidos cada año. El reporte indica que el entrenamiento de los profesionales de la salud no era adecuado y que era insuficiente la evaluación de su destreza.”[1]

Es sumamente necesaria la fiabilidad de estos sistemas, es decir, del apego a las leyes físico-matemáticas que rigen el comportamiento de los objetos depende el éxito de la simulación realizada. 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 **problemática** radica en la necesidad de un módulo que permita la representación de objetos deformables y su reacción ante agentes deformadores en tiempo real, haciendo énfasis en la representación de órganos para su posterior uso en la simulación quirúrgica.

En la Universidad de las Ciencias Informáticas (UCI), se producen simuladores, juegos y se preparan las condiciones para el desarrollo de un simulador quirúrgico. Se cuenta con una biblioteca de clases producida por el grupo “Herramientas de desarrollo para sistemas de realidad virtual” con el nombre de Scene Toolkit (STK), con funcionalidades que facilitan la producción de simuladores pero no dispone de recursos para el trabajo con objetos deformables.

Como **objetivo general**, este trabajo persigue desarrollar un módulo acoplado a la STK capaz de representar objetos que se deformen en tiempo real, según sus características físicas, ante la acción de fuerzas externas.

Existe todo un espectro de ideas en torno al tema, muchos expertos tienen distintas visiones de cómo afrontar el problema. El **objeto de estudio** de este trabajo es el análisis del comportamiento físico de los objetos no rígidos y el **campo de acción,** las técnicas y algoritmos utilizados para la simulación de este tipo de objetos y su eficiencia.

Existen en el mundo muchos y muy efectivos acercamientos al tema de la modelación de órganos, incluso, muchos creen que “Los cirujanos están presenciando la revolución más importante de la cirugía en toda su historia. La llegada de la informática permitió el desarrollo de la cirugía mínimamente invasiva. La realidad virtual y la simulación están invadiendo las salas de cirugía y han cambiado completamente la forma de entrenamiento quirúrgico” [2].

Varios de los países más desarrollados del mundo han fomentado desde mediados de los ’80 proyectos de simulación quirúrgica, resaltan el *KISMET 3D-Simulation Software* de Alemania (Fig. 1), el *Laparoscopic Cholecystectomy* desarrollado por Michael Downes *et al.* en la Universidad de California [3], *Laparoscopic Simulator* desarrollado por Srinivasan *et al.* en el *Laboratory for human and machine haptics* de la Universidad de Missouri-Columbia [4], *Hepatic Surgery Simulator* desarrollado por S. Cotin *et al.* en el centro de investigación INRIA de Francia [5], *Anastomosis Simulator* desarrollado por *Boston Dynamics Inc.* [6] “En 1996 un Grupo de Trabajo multidisciplinario en cooperación con en Departamento de Obstetricia y Ginecología de la Universidad de Zurich, el Instituto de Anatomía y el Instituto de Sistema Mecatrónica de la Universidad de Ciencias Aplicadas de Winterthur, comenzaron el proyecto *LaSSo (Laparoscopic Surgery Simulator)* con la intención de construir un Simulador Quirúrgico basado en las Técnicas de Realidad Virtual.” [7] En el propio 1996 surge *Mimic Technologies* y en 1997 nace *Simbionix*, actuales líderes globales de las tecnologías de simulación quirúrgica.

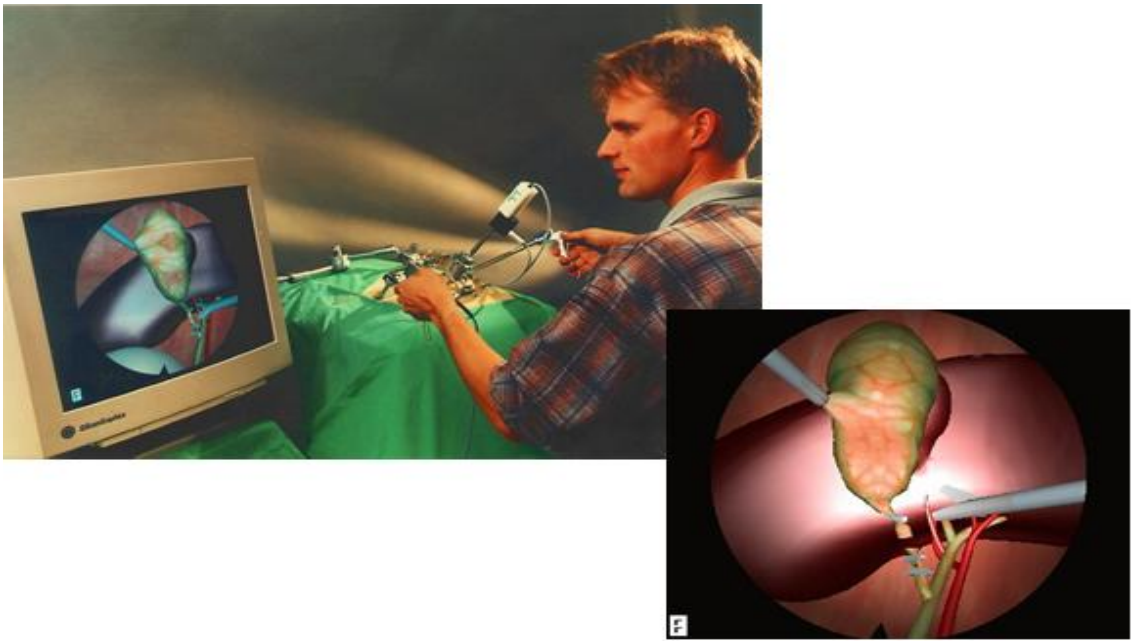


Fig. 1 Simulador de Cirugía de Mínimo Acceso. Proyecto KISMET, Alemania.

Al culminar este trabajo se espera contar con un módulo para ser integrado a la STK, capaz de proporcionar al programador una manera viable de representar objetos que reaccionen ante fuerzas externas, obedeciendo parámetros reales y con alta calidad de visualización, abriendo paso a aplicaciones de simulación quirúrgica.

Para satisfacer las necesidades planteadas es necesario ejecutar varias **tareas** que se relacionan a continuación:

- ✓ Estudiar los parámetros físicos que definen la rigidez y capacidad de deformación de cuerpos no rígidos.
- ✓ Estudiar las tendencias actuales de la simulación de objetos deformables y tejidos así como de técnicas y algoritmos que favorezcan la velocidad de render.
- ✓ Estudiar las características de la STK para la integración.
- ✓ Desarrollar la documentación del módulo a implementar.
- ✓ Desarrollar parcial o totalmente un módulo para solucionar el problema planteado.

Para una mejor comprensión, el documento está dividido en capítulos que estructuran el contenido 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 con el objetivo de esclarecer cual es posible candidata para una solución.

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 como indica el diagrama de componentes del sistema.

Capítulo 1 : Fundamentación Teórica

Introducción:

La simulación del comportamiento de los órganos no es tarea trivial, se requiere de una serie de ecuaciones matemáticas capaces de describir el comportamiento elasto-mecánico de los tejidos y comportamientos acorde con leyes de la mecánica de sólidos para obtener deformaciones con alto nivel de realismo, pero a su vez lo suficientemente poco complejas para garantizar su cálculo en tiempo real.

Existen muchas investigaciones enfocadas a la optimización de este proceso, así como de su modelación físico-matemática. La manera de representar gráficamente los objetos puede influir en la calidad de las aplicaciones, en esta área se ha desatado un debate acerca de cual es la más eficiente, la tradicional representación en mallas o la revolucionaria idea de usar los puntos como primitiva gráfica.

1.1 Características Generales y Antecedentes de las Deformaciones de Cuerpos

Cuando se habla de objetos deformables, se refiere a objetos que, debido a sus propiedades físicas, su forma puede ser modificada por la acción de otro cuerpo o agente externo (gravedad, viento, pinza, mano, etc.). Generalmente en los ambientes virtuales actuales, los objetos son representados por mallas triangulares o últimamente como una colección de puntos en el espacio, de manera que una modificación de su forma implica una variación en la ubicación de los vértices y aristas de la malla en correspondencia con la acción que la provoca. El cálculo de la nueva posición para cada vértice o arista de la malla es sumamente costoso para las PC actuales, más aún cuando se está en presencia de escenarios muy complejos, con cientos de miles de polígonos como es el escenario quirúrgico. “Aunque es posible el render en PC del comportamiento gráfico de sistemas comerciales, la simulación en tiempo real de la deformación de tejidos es aún el mayor obstáculo para el desarrollo de simuladores de cirugía” [8].

La idea de la deformación es aparentemente simple, bastaría sólo con determinar que parte del cuerpo sufre cambios ante una deformación dada y recalcular la posición de cada vértice del área afectada, sin embargo, esa nueva posición depende de una serie de parámetros físicos o no, dependiendo de la técnica de deformación usada, lo cual requiere de un cálculo muy costoso, debido a que cada vértice que sufre cambios deberá calcular todos estos parámetros para relocalizarse.

Muchos han sido los esfuerzos en esta área hasta hoy, a pesar de ser un campo relativamente joven, puede decirse que es sumamente dinámico. Uno de los primeros modelos para animar cuerpos deformables fue introducido por Dimitri Terzopoulos [9]. En el trabajo referenciado, el autor emplea la teoría de la elasticidad para generar ecuaciones diferenciales que modelan los sólidos en función del tiempo. Este primer modelo, basado en la Ley de Hooke para objetos perfectamente elásticos, ha sido mejorado sucesivamente y es considerado el pionero en este tema. A continuación se relacionan los hechos y tendencias más significativos hasta la actualidad.

Entre las técnicas de deformación están presentes fundamentalmente dos tipo, ellas son: métodos geométricos y métodos basados en física.

Los 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. Aún cuando su analogía con las deformaciones físicas no son del todo obvias, esta técnica se ha hecho efectiva en algunos simuladores de cirugía debido a la velocidad del cálculo de la deformación. Algunos métodos que responden a este tipo de deformación son: Splines y Ajuste de Curvas, Deformación de Libre Forma y 3D ChainMail.

Entre los métodos basados en física, los Sistemas Masa-Resorte han sido los más populares, pero existen otras alternativas como el Método de Elementos Finitos (FEM) y Método de los Elementos de Frontera (BEM), que se sustentan en la física de materiales y la mecánica continua.

Los modelos de superficies elásticamente deformables, calculan la deformación resolviendo las ecuaciones lineales de elasticidad, lo que permite la simulación basada en los principios de la física. La extensión de esta técnica a 3D es el primer intento en la modelación de tejidos partiendo de elementos finitos. [10]

1.2 Mecánica de Sólidos

La mecánica de sólidos deformables o resistencia de materiales, es la rama de la física encargada del estudio de las deformaciones de cuerpos ante la acción de una fuerza o un efecto térmico. Para resolver problemas de este tipo hay que determinar el campo de tensiones y el campo de deformaciones del cuerpo, para ello se utilizan tres tipos de ecuaciones:

- **Ecuaciones de Equilibrio:** Relacionan tensiones internas del sólido con las cargas aplicadas. Las ecuaciones de la estática son deducibles de las ecuaciones de equilibrio.
- **Ecuaciones Constitutivas:** Relacionan la tensión y la deformación, en las que pueden intervenir también otras magnitudes como temperatura, velocidad de deformación, deformaciones plásticas acumuladas, variables de endurecimiento, etc.
- **Ecuaciones de Compatibilidad:** a partir de la cual puede calcularse los desplazamientos en función de las deformaciones y las condiciones de contorno o enlace con el exterior.

Atendiendo a la ecuación constitutiva, los sólidos deformables pueden clasificarse como sigue:

- **Comportamiento elástico:** El sólido se deforma adquiriendo mayor energía potencial elástica y en consecuencia, aumenta su energía interna sin que se produzcan transformaciones termodinámicas irreversibles. Al valor máximo de la fuerza aplicada para el que la deformación es elástica, se le denomina límite elástico, a partir de este valor, el proceso será irreversible. Dentro del comportamiento elástico hay varios subtipos:
    - **Elástico lineal:** Existe una relación lineal entre las fuerzas incidentes y el desplazamiento del material**:**
        - **Isótropo:** Son aquellos que se deforman igualmente en cualquier dirección, su constante elástica es la misma en cualquier punto del cuerpo [47].
        - **No-Isótropo:** La magnitud de la deformación puede variar en dependencia de la dirección o la distancia del punto de contacto con la fuerza.
    - **Elástico no-lineal:** Las fuerzas incidentes y el desplazamiento del material no tienen una relación lineal.

- **Comportamiento plástico:** Este comportamiento se distingue por ser irreversible y puede presentarse de distinta forma:
    - Plástico puro.
    - Plástico con endurecimiento.
    - Plástico con ablandamiento.
- **Comportamiento viscoso:** Se produce cuando la velocidad de deformación entra en la ecuación constitutiva. Aquí se pueden distinguir los siguientes modelos:
    - **Visco-elástico**: Relaciona las propiedades elásticas y viscosas del material.
    - **Visco-plástico**: Se refiere a materiales que comparten propiedades tanto plásticas como elásticas.

Es válido aclarar que los tipos de deformaciones pueden coexistir en un mismo sólido según sea el rango de tensión y deformación que predomine. El comportamiento dependerá de la forma concreta de la ecuación constitutiva que relaciona la tensión, la deformación, la velocidad de deformación y la deformación plástica, junto con parámetros como las constantes elásticas, la viscosidad y parámetros termodinámicos como la temperatura o la entropía [25].

1.2.1 Deformaciones Elásticas

Debido a las características del tejido humano sólo resultan de interés inicialmente las deformaciones elásticas, es decir, aquellas en las que el cuerpo trata de conservar su estado inicial contrarrestando la fuerza aplicada, una vez retirada la misma, el objeto recupera su forma inicial. Los objetos deformables usualmente se definen por su estado inicial o de equilibrio y por una serie de parámetros que definen como este se comportará ante la acción de una fuerza. "La teoría de la elasticidad establece que la tensión en un tiempo t depende sólo de la deformación local en ese momento de tiempo y no de los antecedentes de la deformación" [47].

El conjunto de los desplazamientos de cada punto define el campo de deformación del objeto. La *deformación* ε mide el desplazamiento por unidad de área (en el caso de lineal, ε es simplemente $\Delta l / l$, donde l es la longitud del objeto). Un campo de desplazamiento espacialmente constante, representa una traslación del objeto sin deformación. Por tanto la deformación debe ser medida en términos de variación espacial del campo de desplazamiento u (m) = {u, v, w} ^{T} y para ello se emplean las siguientes expresiones:

$$\varepsilon_G = \frac{1}{2}(\nabla u + [\nabla u]^T + [\nabla u]^T \nabla u) \quad (1)$$

$$\varepsilon_C = \frac{1}{2}(\nabla u + [\nabla u]^T) \quad (2)$$

Donde $\varepsilon_G \in \Re^{3x3}$ es el tensor simétrico de desplazamiento no lineal de Green y $\varepsilon_C \in \Re^{3x3}$ su linearización, el tensor lineal de Cauchy. El gradiente del campo de desplazamiento ∇u es una matriz de 3 X 3.

$$\nabla u = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \\ \frac{\partial x}{\partial w} & \frac{\partial y}{\partial w} & \frac{\partial z}{\partial w} \end{bmatrix} \tag{3}$$

La ley de materiales (o constitutiva) usada para calcular el tensor de *tensión* $\sigma \in \Re^{3x3}$ (en el caso unidimensional es Fuerza/Área y tiene como unidad de medida el Pa, al igual que la presión) para cada punto del material basado en la *deformación* ε de ese punto es usualmente la Ley de Hooke [14]

$$\sigma = E \cdot \varepsilon \tag{4}$$

Donde E depende del módulo de elasticidad longitudinal o módulo de Young. Para caracterizar el comportamiento de un sólido elástico lineal e isótropo se requieren además del módulo de Young otra constante elástica, llamada coeficiente de Poisson (v). [18]

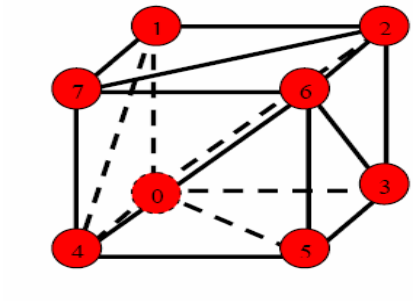
La energía de deformación, es el aumento de energía interna acumulada en el interior de un sólido deformable, como resultado del trabajo realizado por las fuerzas que provocan la deformación.

1.3 Representación Geométrica

En una escena virtual coexisten diferentes tipos de objetos (edificios, árboles, órganos, humo, fluidos…) que por su naturaleza no pueden ser representados del mismo modo. Existen varios modelos con este fin, pero sólo se mencionan a continuación los que por sus características pueden ser utilizados para la representación de objetos deformables.

1.3.1 Modelos Basados en Mallas

Tratando de conceptualizar una malla, pudiera afirmarse que no es más que una colección de vértices y aristas en forma poligonal que representa la forma de un objeto poliédrico. A lo largo de la historia de los gráficos por computadora se han usado diversos tipos de mallas, ya sea mallas triangulares, poligonales, de tetraedros etc. Sin embargo, las mallas triangulares son las más extendidas en el mundo de los gráficos por computadora. "La mayoría de los *3D engines* (tanto hardware como software) sólo se relacionan con triángulos. Las razones son muchas, entre otras, los triángulos son fáciles de representar en sistemas gráficos, tres puntos definen un plano y muchos de los algoritmos 3D trabajan mejor con mallas triangulares." [26] En comparación con otras formas de representación explícita, como las superficies Splines, las mallas triangulares no están especificadas en términos de una parametrización, sin embargo se considera 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 en triángulos. Para la mayoría de las mallas, el número de triángulos es aproximadamente el doble del número de vértices [37].



Coordenada de vertices

	x	y	z
0	-10	-10	-10
1	-10	10	-10
2	10	10	-10
3	10	-10	-10
4	-10	-10	10
5	10	-10	10
6	10	10	10
7	-10	10	10

Indice de las caras

1	0	1	2
2	3	0	2
3	4	5	6
4	7	4	6
5	0	3	5
6	4	0	5
7	3	2	6
8	5	3	6
9	2	1	7
10	6	2	7
11	1	0	4
12	7	1	4

Fig. 2 Malla triangular, sus vértices y conectividad.

Los triángulos han sido considerados por algunos como los reyes entre las primitivas gráficas por su simplicidad y el soporte que brindan las herramientas actuales para su *render*. Cuando se almacenan en una estructura apropiada, los triángulos almacenan datos de conectividad a través de relaciones entre los vértices, aristas y adyacencia. [20] Debido a su extendido uso, en diversos textos son considerados incluso como una norma, se habla de "geometría basada en triángulos". [26]

1.3.2 Modelos Basados en Puntos

La mayoría de las aplicaciones de simulación tanto 2D como 3D en las que intervienen deformaciones, están basadas en mallas, tanto Sistemas Masa-Resorte como sistemas motivados por métodos matemáticos como Método de Elementos Finitos (*Finite Element Method FEM*) o Elementos de Frontera (*Boundary Element Method BEM*) conjugados con la teoría de la elasticidad. Sin embargo las aplicaciones basadas en puntos se han convertido en una popular técnica en el mundo de los gráficos por computadoras. Una razón clave de este nuevo interés en los puntos es que la complejidad poligonal de los modelos gráficos se ha incrementado dramáticamente en la última década, lo que lógicamente hace mucho más complejos los enmallados tradicionales.

Una representación geométrica basada en puntos, puede ser considerada como un muestreo de una superficie continua (como se muestra en la 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 [38]. En animaciones y simulaciones físicas, complejos efectos físicos como deformaciones o fracturas, implantan un gran reto técnico en términos de mantener la consistencia topológica de la malla subyacente. [19] Por ejemplo, en simulación de procesos de fractura se necesita modelar la propagación de la fisura a través de caminos complejos y arbitrarios, este problema en particular no es fácil de abordar usando métodos convencionales basados en malla debido a que en las zonas afectadas por cortes es necesario enmallar dinámicamente para evitar simular la existencia del interior de los objetos.

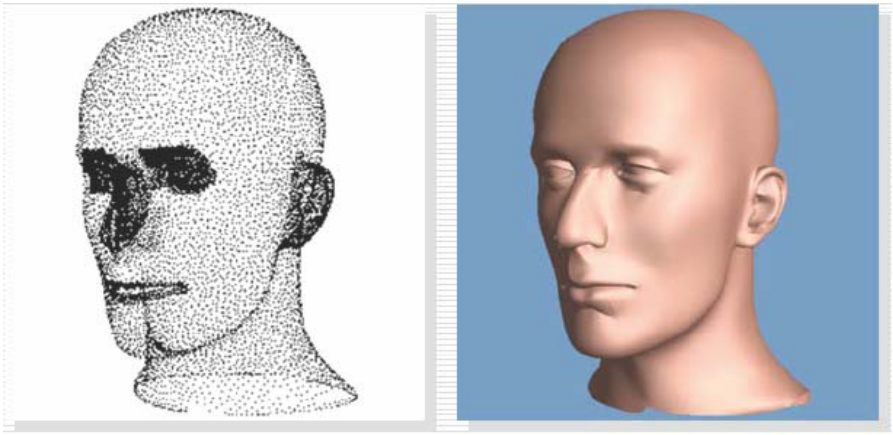


Fig. 3 Rostro representado a través de puntos.

Probablemente el artículo de Levoy y Whitted en 1985 sea el primer intento serio en el uso de puntos como primitiva básica en gráficos por computadoras, recientemente este tema ha recibido un fuerte impulso, algunos incluso creen que “Los puntos son claramente la primitiva gráfica más simple […] El punto constituye una primitiva de bajo nivel simple y versátil. Si bien no van a reemplazar las existentes, han probado su capacidad de complementarlas.” [20]

1.4 Métodos de Deformación Geométrica

Si bien la mayoría de las técnicas discutidas en este trabajo son basadas en parámetros físicos, no se puede obviar la existencia de técnicas puramente geométricas, que también han sido empleadas con distinto grado de éxito, sobre todo en el campo de la eficiencia computacional. Los métodos basados en la geometría modifican la forma de los objetos indirectamente, a través de puntos de control o ajustando parámetros en la función que define la superficie. Estos métodos garantizan un buen rendimiento, pero a su vez sus principales desventajas son:

1. No modelan las propiedades físicas del material, o al menos no en un grado capaz de generar soluciones eficaces físicamente.
2. No emplean leyes físicas para calcular la nueva forma del cuerpo.
3. Como la deformación es calculada indirectamente resulta complejo proporcionar la directa manipulación del objeto [17].

1.4.1 Splines y Ajuste de Curvas

Muchas de las técnicas de modelación de objetos deformables provienen del campo del diseño asistido por computadoras (*Computer Aided Design CAD*). Los diseñadores necesitan vías para especificar numéricamente curvas y superficies, así como para refinar los objetos que modelan. De allí el surgimiento de las curvas de Bezier y otros métodos de especificar curvas con pequeños vectores de números, incluyendo los Splines, B-Splines, NURBS... [16] "La representación clásica de figuras tiene características basadas en la teoría de la elasticidad" [9]. Sin embargo en la adopción del Spline para representar curvas y superficies, la literatura gráfica ha desechado las bases físicas de estos.

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 de manera predecible moviendo los puntos de control o variando la cantidad de ellos, como se muestra en la siguiente figura.

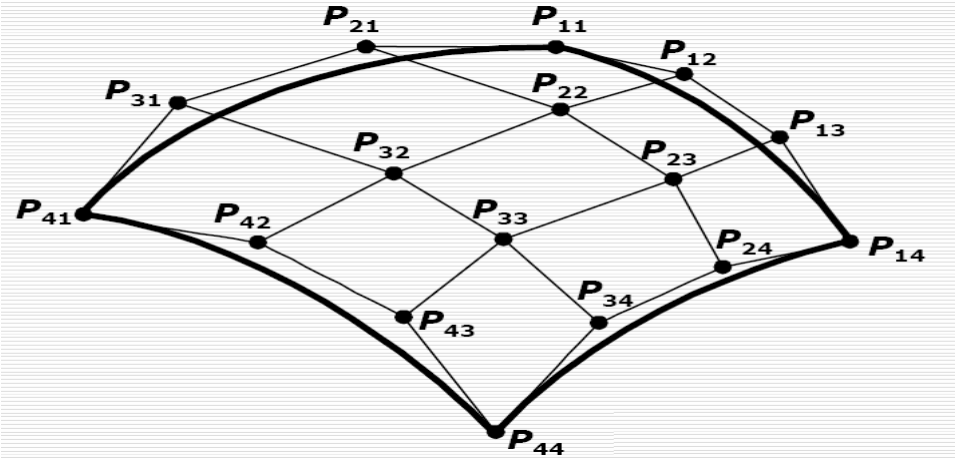


Fig. 4 Curva Spline, P_n: Puntos de Control

Esta vía de representar los objetos es eficiente desde el punto de vista computacional y soporta modificaciones interactivas pero el nivel de precisión alcanzado puede ser cuestionable. Sin embargo,

este nivel de control puede resultar muy trabajoso para modificaciones muy precisas, como pudieran ser las de cirugía, incluso cambios muy simples pudieran requerir del ajuste de varios puntos de control [16]. El uso de este método puede decirse que ha estado limitado a la modelación de órganos partiendo de imágenes médicas, y no a su deformación en tiempo real como es el caso de KisMo [36].

1.4.2 Deformación de Libre Forma

El método de Deformación de Libre Forma (*Free-Form Deformation FFD)* fue introducido por T. Sederberg y S. Parry en 1986, es un método muy rápido para representar y modelar objetos flexibles basada en deformaciones del espacio que los contiene. [17]

Las deformaciones son definidas por funciones paramétricas (3D Splines) cuyos valores son determinados a partir de puntos de control. Usualmente estos puntos de control conforman una malla o rejilla regular en forma de paralelepípedo y a cada uno de sus puntos le son asignados una serie de coordenadas locales, como muestra la figura 5.

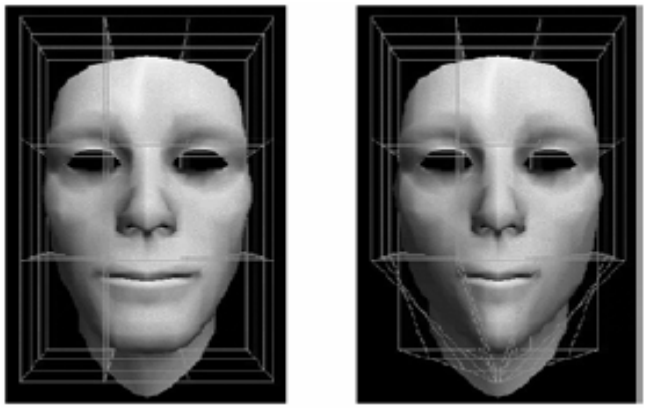


Fig. 5 Malla en forma de paralelepípedos asociada a un cuerpo

Algunos trabajos han propiciado que la interacción sea más intuitiva, como muestra el trabajo presentado por Hsu en 1992 que permite la directa manipulación de puntos en la superficie del cuerpo [33]. Más tarde, en 1996, en MacCracken y Joy proponen una solución para rejillas no regulares a través de un algoritmo de subdivisión [35]. Debido a su buen comportamiento y rapidez este método fue usado en un sistema de simulación laparoscópica implementado por C. Basdogan en 1998 [17] en este mismo año y con el mismo

propósito, este método fue usado por Baur [43]. Originalmente FFD es una técnica basada en geometrías, sin embargo algunos trabajos han integrado técnicas basadas en física. G. Hiriota en 1999 describió una extensión para FFD basada en la ley física de conservación de la masa, de manera que su algoritmo garantiza la preservación del volumen [32]. En el 2004 Guy Sela introdujo un nuevo tipo de FFD llamado *Discontinuous Free Form Deformation (DFFD)*, capaz de soportar discontinuidades y cambios en la topología en la malla debido a cortes o tensiones internas en tiempo real, tanto en superficies descritas por mallas como en objetos volumétricos.

Algunos expertos, como el Dr. Richard Paul Holbrey de la Universidad de Leeds, piensan que aunque este modelo es muy sencillo de implementar para tiempo real, la ausencia de propiedades internas significa que es difícil crear deformaciones realistas para simular interacción, aún cuando hay acercamientos para su integración con técnicas basadas en física, por lo que este modelo pudiera parecer convincente, sin embargo violaciones en la rejilla y 3D *aliasing* son aún comunes. [11] Otros como Sela creen que con la introducción de conceptos basados en física y la manipulación directa de los objetos, FFD es apropiado para la simulación quirúrgica [17].

1.4.3 3D ChainMail

El 3D ChainMail es un algoritmo que fue creado por Sarah Gibson en el año 1997, para deformar mallas uniformes. Está basado en geometría pero puede simular las propiedades de los materiales [17]. Su funcionamiento se basa en que los elementos son modelados por una cantidad X de eslabones enlazados entre sí, se le llama eslabón a un cubo en el caso del 3D ChainMail (cuadrados para el caso de 2D ChainMail) para modelación de elementos volumétricos, donde cada uno tiene un rango limitado de movimiento, de modo que el objeto que se esté simulando tenga un nivel de rigidez y elasticidad acorde con la capacidad de movimiento que contenga cada eslabón, cada uno de ellos está conectado con sus 6 vecinos más cercanos: frente, fondo, laterales izquierdo y derecho, superior e inferior. Usando este algoritmo, los materiales elásticos, rígidos, deformables y plásticos pueden ser ajustados según los límites de deformación, tratando cada objeto de forma individual, donde los vecinos más cercanos pueden estar en tres tiempos fundamentalmente: relajado, máximo comprimido y máximo extendido como se muestra en la figura 6.

Fig. 6 Grados de libertad de un eslabón en 3D ChainMail

Al mover un eslabón, sus vecinos se moverán sólo si el movimiento es mayor que el rango que cada elemento puede soportar, de esta manera se propaga la deformación por el volumen afectado como se muestra en la figura 7.

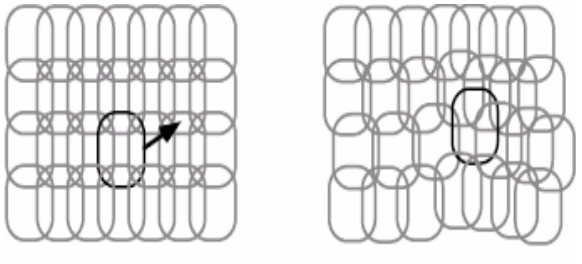


Fig. 7 Deformación del 3D ChainMail cuando un eslabón seleccionado es movido.

El proceso consta de dos pasos:

1. Calcular la deformación cuando un eslabón es desplazado.
2. “Paso de relajación” el objeto se relaja ajustando cada elemento hasta que se alcanza un estado válido de mínima energía.

La mayor ventaja de este algoritmo es su rendimiento, Gibson ha demostrado que cada elemento es procesado como máximo una vez. Esto permite que el algoritmo trabaje con grandes cantidades de datos y produzca una respuesta interactiva [17]. Sin embargo, su principal desventaja es que su uso está restringido al trabajo con mallas rectilíneas, la STK tiene implementado este tipo de representación, pero carece de la representación basada en puntos y si en versiones posteriores hace este cambio de topología en busca de mayor eficiencia, el algoritmo dejaría de ser útil.

El algoritmo ha sufrido cambios debido a sus restricciones, como que sólo podía trabajarse con datos homogéneos, una de las modificaciones fue hecha por Markus Schill en el año 1998, esta mejora del algoritmo se le llamó "*Enhanced ChainMail*" donde eliminaba dicha restricción, este algoritmo fue probado en un sistema 2D para cirugía oftalmológica [31]. Otra restricción que presentaba el 3D ChainMail es que sólo podía funcionar sobre una malla rectilínea, la cual fue resuelta por Ying Li y Ken Brodlie en el año 2003 llamándole al método "Generalised ChainMail", este se caracterizaba por su rapidez en los cálculos, tanto así como para hacer simples simuladores de entrenamiento quirúrgico en la Web. En el año 2005 se crea el algoritmo "Divod ChainMail" por Christopher Dräger, que es un algoritmo para la deformación directa del volumen basado en el "*Enhanced ChainMail*" y posteriormente se integró a un sistema virtual de endoscopia llamado STEP creado por VRV para la simulación de la cirugía de adenomas endonasales [17].

1.5 Métodos de Deformación Basados en Física

Los modelos deformables basados en física tienen dos décadas de historia en los gráficos por computadoras, desde la discusión de Lasseter's acerca de contracción y elasticidad en 1987 y las descripciones de Terzopoulos sobre modelos deformables elásticamente, numerosos investigadores han tomado parte en la visualización de objetos deformables y fluidos. Este campo combina la dinámica newtoniana, mecánicas continuas, computación numérica, geometría diferencial, cálculo de vectores, teorías de aproximación y gráficos por computadoras. [18]

Los métodos basados en física son una popular técnica para representar objetos deformables como goma, ropas, piel humana etc. mediante la incorporación de propiedades físicas al material modelado, por lo que brinda una solución muy intuitiva.

1.5.1 Sistemas Masa-Resorte

El Sistema Masa-Resorte es una técnica basada en física ampliamente usada para modelar objetos blandos, consiste en la discretización de los objetos como una malla de partículas y muelles. “Las partículas no son más que objetos que tienen masa, posición, velocidad y responden a la acción de fuerzas pero que carecen de extensión espacial. A pesar de su simplicidad, pueden mostrar comportamientos interesantes.” [12]

“De manera general un Sistema de Masa-Resorte consta de n puntos de masa (partículas) enlazadas con sus vecinos a través de muelles libres de masa y longitud natural mayor que cero.” [13] Pueden ser muchas las implementaciones que se encuentran de este modelo, pero todas sobre un tronco común: dos ecuaciones clásicas de la física mecánica, la Segunda Ley de Newton que define el movimiento de cada nodo y la Ley de Hooke que define la tensión que une a los nodos mediante muelles. La fuerza interna entre dos nodos es $f_{ij} = -k_{ij}\Delta u_{ij}$ donde Δu_{ij} es la diferencia entre la longitud del resorte en reposo y la longitud en un instante de tiempo dado, k_{ij} la constante elástica del resorte y u_{ij} es el vector unitario entre N_i y N_j siendo N los centros de masa. En cada partícula finalmente se calcula la sumatoria de las fuerzas que la modifican (internas o externas) y con ello se obtiene la nueva posición.

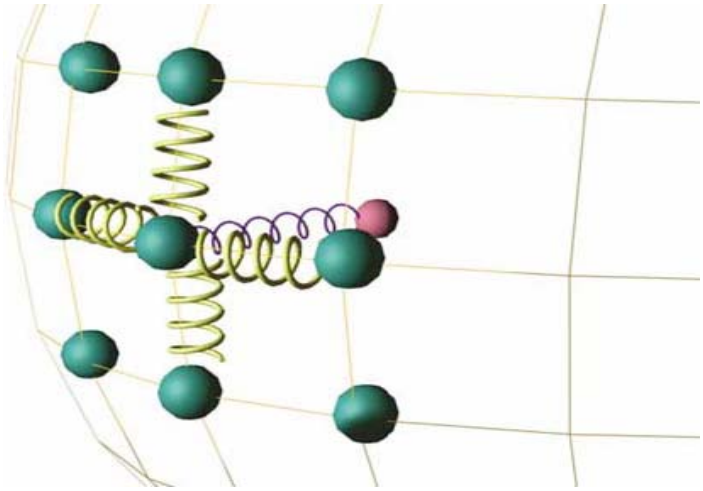


Fig. 8 Sistema Masa-Resorte

Uno de los trabajos pioneros en el uso de este método para soluciones dinámicas fue propuesto por D. Terzopoulos y K. Waters en 1990, su idea fue modelar capas de distintos materiales con diferentes capas de Masa-Resorte. Ellos usaron tres capas de Masa-Resorte para animación facial, simulando las capas de la piel [27]. En 1998 N. P. Nedel presentó un trabajo basado en Sistemas de Masa-Resorte para deformación de músculos en tiempo real. Su método no pretendía obtener una deformación perfecta, sino una herramienta provechosa para aplicaciones interactivas con una aproximación visual aceptable [29], este trabajo se enfrenta con algunas desventajas de los Sistemas de Masa-Resorte, incluyendo el comportamiento irrealista en deformaciones grandes así como los valores de las constantes de los muelles para lograr comportamientos apropiados, especialmente en objetos rígidos como los huesos, la modelación de objetos rígidos a través de este método es una pérdida de rendimiento, debido a que sus deformaciones son insignificantes o nulas [17]. También en este año el método fue usado en simulación laparoscópica por Michael Downes [44]. Hay ejemplos que evidencia la efectividad de este método en la simulación quirúrgica, como es el "Karlsruhe Endoscopic Surgery Trainer" presentado por U. Kühnapfel en el 2000, el software del sistema, llamado KISMET usa Sistemas de Masa-Resorte para simular la elastodinámica de los objetos [28]. Un trabajo de M. Teschner en 2004 aborda el tema de la deformación de objetos geométricamente complejos desde una perspectiva muy próxima a los Sistemas de Masa-Resorte. Está basado en mallas triangulares o tetraédricas y deriva tres fuerzas basadas en energías potenciales en cada centro de masa. Las tres fuerzas son modeladas para conservar la distancia entre los puntos de masa, el área de la superficie y el volumen de los tetraedros. Las propiedades del material son manipuladas mediante los coeficientes de rigidez. La gran ventaja de esta propuesta es que modela deformaciones tanto plásticas como elásticas [30].

Este tipo de modelación ha sido muy usada hasta hoy debido a que es la forma más intuitiva de imaginar un modelo deformable, es una técnica poco compleja para la implementación y garantiza velocidades superiores a métodos continuos [16]. Sin embargo, no es todo lo exacta que se necesita porque no se soporta en la elasticidad continua, además depende en gran medida de la topología y la resolución de la malla. [14] En un documento expuesto en EuroGraphics 2005 por varios investigadores del tema puede leerse que "… a diferencia del Método de Elementos Finitos, basado en la teoría de la elasticidad, los Sistemas de Masa-Resorte no son necesariamente exactos." [18]. Otros enfoques insisten en que "… a pesar de la simplicidad de su formulación, el sistema resultante puede ser de costosa solución; la

inexactitud del método implica un elevado número de nodos para simulaciones realistas (comparado con FEM o BEM)" [50]. Las constantes de los muelles son usualmente seleccionadas arbitrariamente, alejándose de las propiedades reales del material simulado, además, algunas restricciones no son mostradas en el modelo, como el caso de superficies finas resistentes a curvaturas. Según Gibson y Mirtich en un documento publicado en noviembre de 1997, afirma que los Sistemas de Masa-Resorte en ocasiones muestran problemas respecto a los objetos con constante elástica elevada, es decir, cuerpos cercanos a la rigidez por lo que sistemas que contienen elementos rígidos están expuestos a problemas de estabilidad cuando las partículas están poco limitadas (un muelle limita la distancia entre partículas). [16][46].

1.5.2 Método de Elementos Finitos (FEM)

El Método de Elementos Finitos (FEM por sus siglas en inglés) es uno de los métodos más populares en las ciencias de la computación para resolver ecuaciones diferenciales en rejillas irregulares. El artículo publicado por Turner, Clough, Martin y Topp en 1956 es reconocido como el inicio del actual Método de Elemento Finito [45]. En muchos artículos se habla de una forma simple de FEM para la simulación de deformaciones denominado FEM explícito que es una propuesta fácil de comprender e implementar [39] [40]. El FEM explícito no debe ser confundido con el FEM estándar integrado explícitamente. El FEM explícito puede ser integrado tanto implícita como explícitamente. FEM describe los objetos como un conjunto finito de elementos básicos (triángulos, tetraedros, cubos…) y transforma la mecánica continua de deformaciones (ecuaciones físicas, condiciones de contorno…) en un problema discreto que puede ser resuelto usando análisis numérico [48].

En esencia el método consiste en dividir el objeto en un conjunto finito de elementos mediante discretización geométrica y luego las propiedades físicas del objeto son interpoladas para cada elemento usando funciones de forma, de manera que la mecánica continua del objeto es expresada en términos de un conjunto de elementos [47].

Pudieran definirse una guía para el método como sigue:

1. **Discretización geométrica:** El cuerpo se subdivide en elementos más simples (usualmente tetraedros o hexaedros), en cada elemento se definen nodos que son puntos de control donde se

evalúa el problema y describen localmente el material del objeto. La exactitud del método es directamente proporcional a la cantidad de subdivisiones hechas, ver la figura 9.

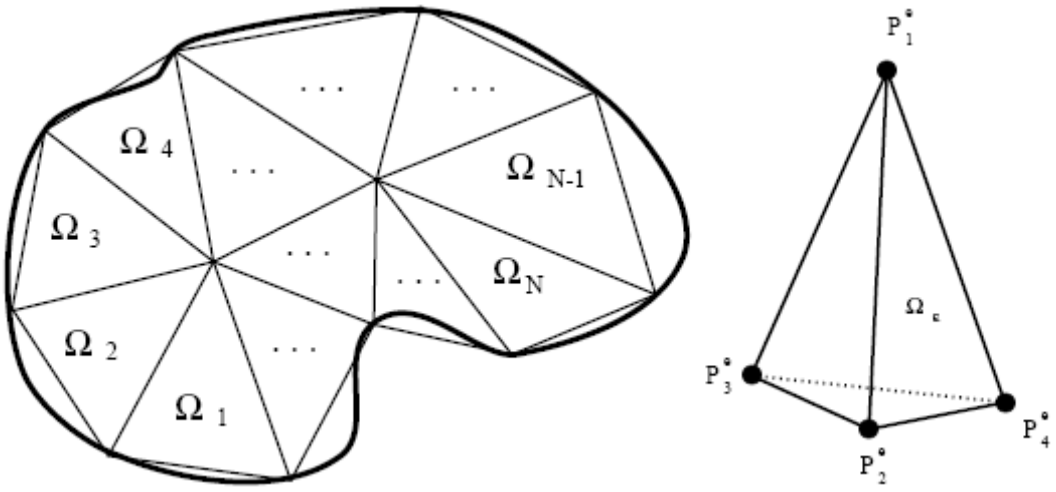


Fig. 9: Discretización de un dominio en elementos tetraédricos.

2. **Interpolación:** Se usan funciones de forma (también conocidas como funciones base) para interpolar cualquier punto P del continuo. Por tanto, cualquier función continua aplicada a un cuerpo continuo es aproximada por un sistema de ecuaciones finito en término de coordenadas de los nodos.
3. **Aplicación de la Mecánica Continua:** Una vez discretizado el cuerpo y las funciones de forma se deben encontrar expresiones que empleen las ecuaciones de elasticidad en términos de los nodos de la malla, esto permitirá calcular la distribución interna de tensiones del elemento. En dependencia de las necesidades del sistema su formulación puede ser estática o dinámica.

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

$$Mx''+Dx'+Kx=F \tag{5}$$

Donde M, D y K son matrices de 3n x 3n de masa, amortiguación y rigidez respectivamente, F es el vector fuerza aplicado al objeto. Una solución analítica no es posible para este sistema debido a su complejidad, por lo que su solución tendrá que ser numérica [49].

- Formulación Estática: La ecuación 5 puede ser simplificada dejando de tener en cuenta aspectos como la inercia, entonces la ecuación del movimiento puede ser expresada como:

$$Kx = F \tag{6}$$

A principios de la década de los '90 hubo varias propuestas de FEM pero ninguna en tiempo real, en 1994, Sagar desarrolló un ambiente virtual para la simulación de cirugía oftalmológica, donde la córnea era modelada como un material elástico no lineal. Otra aplicación de FEM para problemas quirúrgicos fue la propuesta por Keeve en 1996, que usó el método para predecir el resultado de cirugías faciales [17]. O'Brien en 1999 y en 2002 presentó una versión de FEM para simular ruptura de elementos frágiles y dúctiles, que produjo resultados realistas y visualmente convincentes, pero no para aplicaciones en tiempo real [18].

Bro-Nielsen y Cotin trabajan con FEM Linealizado para simulación de cirugía. Ellos lograron un aumento de velocidad simulando sólo los nodos visibles de la superficie (condensación), similar al Boundary Element Method (BEM). De esta manera se obtienen resultados en tiempo real [41]. En comparación con los Sistemas de Masa-Resorte, el FEM brinda una representación mucho más clara de los parámetros de los tejidos para calcular las fuerzas [11]. FEM fue usado para modelar deformación de tejidos para un simulador de endoscopia ginecológica por Székely y su equipo en 1999 [10].

La principal ventaja de FEM comparada con otras aproximaciones, es que puede producir simulaciones más realistas físicamente, debido a que las matrices de masa y rigidez permanecen constantes en el tiempo. Sin embargo requiere de muchos cálculos, que sólo pueden ser reducidos disminuyendo el número de elementos, lo que atenta contra la exactitud del modelo [42].

1.5.3 Método de los Elementos de Fontera

El Método de los Elementos de Frontera (*Boundary Element Method*, BEM), es una alternativa interesante al FEM estándar, porque todos los cálculos se realizan en la superficie del cuerpo elástico en lugar de su volumen como se representa en la figura 10. El método logra un aumento sustancial de la velocidad debido a que el problema tridimensional original, es reducido a dos dimensiones. Sin embargo, sólo puede ser aplicado en cuerpos cuyo interior esté compuesto por un material homogéneo, además, cambios de topología son más complejos de manipular que en FEM Explícito [18].

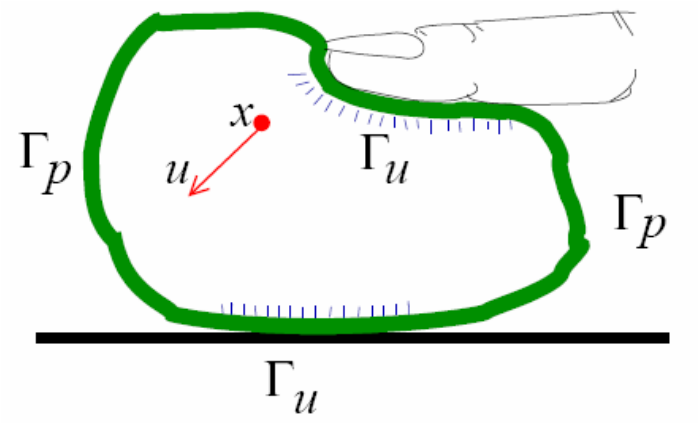


Fig. 10 Notación de BEM y sus condiciones de frontera

Este método tiene una larga historia en análisis de ingeniería, pero fue propuesto por primera vez para simular objetos deformables por Doug L. James y Dinesh K. Pai en 1999, en este trabajo se propone un método quasi-estático para modelar cómo un objeto interactúa con el ambiente en su frontera, Γ. Por ejemplo, en la Fig. 10 el objeto está sujeto a dos condiciones de desplazamiento en la frontera: Γ_u debido al contacto con el dedo y el suelo fijo, y Γ_p libre para moverse, en su interior, ante una deformación, cualquier punto x puede sufrir un desplazamiento u. De las ventajas de este método respecto a FEM para determinado tipo de soluciones, puede decirse que: “BEM es más preciso que FEM para calcular fuerzas de contacto y quizás la mejor opción que ofrece BEM es que usa la misma discretización usada para el render, es decir, no se necesita otro enmallado, en FEM el interior del cuerpo debe ser enmallado” [50]. Otro trabajo fue propuesto en 2003 agregando posibilidades para el cambio de las condiciones de la frontera, como colisiones órgano – órgano [51].

Conclusiones

Al finalizar este capítulo se cuenta con una reseña del estado del arte, así como las tendencias actuales y técnicas existentes en cuanto a los métodos de representación y deformación de cuerpos en ambientes virtuales según diferentes bibliografías, todos con un punto coincidente: su alto costo computacional.

Para lograr entender el proceso de deformación, fue preciso definir en términos físicos el comportamiento de las deformaciones, sus clasificaciones y especialmente las deformaciones elásticas como base para el cumplimiento de los objetivos fijados por este trabajo.

Capítulo 2 : Soluciones Técnicas

Introducción

De acuerdo con el capítulo anterior, la simulación de objetos deformables se hace costosa para lograr un alto nivel de realismo, de ahí que el estudio general que se realizó de los diferentes métodos de representación y deformación sirva para llegar a la solución que se presenta a continuación, esta solución está compuesta por un método de deformación para dar cumplimiento al objetivo planteado inicialmente.

2.1 Método de Deformación

Definitivamente, entre todos los métodos estudiados el que brinda herramientas más exactas es el FEM, teniendo en cuenta la importancia que tiene la precisión en el tipo de simulación que se pretende desarrollar, este aspecto conjuntamente con el factor tiempo es la principal premisa, sin embargo su complejidad físico-matemática obliga a un largo período de estudio antes de su implementación, sobre todo en busca de optimizaciones para posibilitar su ejecución en tiempo real, debido a que sin reducir los cálculos a realizar por el FEM tradicional, no es posible una solución interactiva.

Otra solución viable es los Sistemas de Masa-Resorte, que si bien no son del todo físicamente correctos, son una propuesta que incorpora parámetros físicos y proporciona resultados lo suficientemente correctos y en tiempo real. Algunos autores [11] aseguran que, para deformaciones pequeñas, los modelos de muelles se comportan similar a un modelo elástico de elementos finitos como probara Keeve et al [53].

Finalmente, este trabajo propone una solución que incorpore ambas técnicas de manera que el usuario pueda decidir cual usar en función de sus necesidades. En una primera versión se implementará sólo la parte que usa Sistemas de Masa-Resorte.

2.1.1 Sistema Masa-Resorte

En la STK se usan mallas triangulares para la representación de los objetos, esto brinda la posibilidad de usar los vértices como partículas y las aristas como muelles. Partiendo de la malla se construye un Sistema Masa-Resorte como una colección de partículas que tienen propiedades físicas como masa, fuerza, aceleración y velocidad enlazadas entre sí, por muelles libres de masa pero con constante de rigidez y factor de amortiguación para controlar las posibles vibraciones, lo que permite el cálculo de la fuerza que ejerce el muelle entre las partículas que une.

El comportamiento de los muelles se asume como visco-elástico, debido a que los cuerpos de manera general no son perfectamente elásticos, entonces la fuerza que cada uno de ellos ejerce sobre las partículas i j se calcula de la siguiente forma:

$$F_{elast} = \left[k_s\left(\left|x_j - x_i\right| - l\right) + k_d\left(v_j - v_i\right)\right]\frac{x_j - x_i}{\left|x_j - x_i\right|} \quad (7)$$

Donde k_s es el coeficiente de rigidez del muelle, x las posiciones de las partículas enlazadas por el muelle, l la longitud en reposo, k_d el coeficiente de amortiguación utilizado para simular la disipación de energía que ocurre durante la deformación [14] y v la velocidad de los centros de masa.

Atendiendo a la segunda ley de Newton,

$$x''(t + \Delta t) = \frac{\sum F(t + \Delta t)}{m} \quad (8)$$

Cada partícula se ve afectada por determinada cantidad de fuerzas, en dependencia de la cantidad de muelles que la vinculen con otras partículas y otras fuerzas como la gravedad o sencillamente una fuerza que se aplique, en cada momento de la simulación se tiene la suma de las fuerzas que afectan a cada partícula y su masa, aplicando (8) se obtiene la aceleración y luego integrando con el método de Euler las demás variables cinemáticas [18]:

$$x'(t + \Delta t) = x'(t) + \Delta t x''(t + \Delta t) \quad (9)$$

$$x(t + \Delta t) = x(t) + \Delta t x'(t + \Delta t) \quad (10)$$

2.1.2 Conservación de Volumen

Un Sistema Masa-Resorte de superficie no garantiza que un cuerpo se comporte elásticamente ante una deformación porque los muelles sólo ofrecen resistencia a la deformación en el sentido axial, muchos trabajos generan mallas volumétricas para garantizar elasticidad en el interior de cuerpos volumétricos, sin embargo, esta solución genera mallas con gran cantidad de vértices y aristas [52].

Debido a que muchos de los órganos del cuerpo humano conservan su volumen ante las deformaciones, este es un importante aspecto a tener en cuenta [54]. Suponiendo que los cuerpos volumétricos están llenos de un gas, que se opone a la deformación a través de la presión que ejerce sobre las paredes del mismo, es razonable pensar que una vez retirada la causa de la deformación, esta presión hará al cuerpo retornar a su forma inicial. Para simular este fenómeno, se aplicará la Teoría del Gas Ideal, de manera que el cuerpo simula estar lleno de un gas que cumple con la siguiente relación, que se conoce como ecuación del Gas Ideal y garantiza la proporción entre la presión del gas en interior y el volumen del cuerpo:

$$P * V = nR\Delta T \tag{11}$$

En esta ecuación, n (cantidad de partículas) y R (constante del gas) son constantes, y para este caso, también la temperatura T, de manera que sólo P y V pueden variar de manera inversamente proporcional, de modo que si el volumen disminuye aumenta la presión, aumentando de este modo la resistencia a la deformación del cuerpo volumétrico en cuestión.

Para el cálculo del volumen del cuerpo utilizamos un método propuesto en [54]:

$$V = \frac{1}{3}\sum_{i=1}^{n}\frac{A_i}{3}\begin{Bmatrix}(P_1x + P_2x + P_3x) * N_x + \\ (P_1y + P_2y + P_3y) * N_y + \\ (P_1z + P_2z + P_3z) * N_z\end{Bmatrix} \tag{12}$$

Siendo n el número de caras triangulares en la malla, P_n los vértices de cada cara, N la normal y A_i el área de la iésima cara.

Finalmente pudiera escribirse a manera de algoritmo:

En cada ciclo de render:

- Calcular Δt

- Para cada muelle del sistema

    1. Calcular la fuerza que este ejerce sobre las partículas en sus extremos usando (7).

- Para cada triángulo de la superficie del cuerpo:

    1. Calcular la normal y su módulo.

- Calcular el volumen del cuerpo usando el Método de Gauss.

- Para cada triángulo de la superficie:

    1. Calcular presión que actúa a modo de fuerza desde el interior sobre las caras.
    2. Aplicar fuerza resultante de la presión sobre las partículas.

- Para cada partícula del sistema:

    1. Calcular sumatoria de las fuerzas que la afectan.
    2. Calcular la aceleración usando (8).
    3. Calcular la velocidad según (9).
    4. Calcular la nueva posición usando (10).

2.2 Consideraciones Técnicas Generales

Para guiar el proceso de desarrollo del sistema se utilizará el proceso unificado de desarrollo de software (RUP) y la herramienta CASE Rational Rose Enterprise Edition 2003.

El sistema será implementado en su totalidad siguiendo el paradigma de programación orientada a objetos y en lenguaje C++, usando Microsoft Visual C++ .NET 2003 al igual que el resto de la STK.

El código estará escrito en C++ estándar para garantizar que el sistema pueda ser usado tanto en Windows como en Linux. El convenio de nomenclatura y estándares de codificación que se sigue, se adjuntan en los anexos y es el mismo usado en la STK para no entrar en contradicciones de notación.

Conclusiones

En este capítulo se propuso una solución al problema de las deformaciones, según la investigación desarrollada en el capítulo anterior. Las decisiones se basaron en lograr un equilibrio entre realismo y velocidad de render. Se especificó el lenguaje y la metodología de desarrollo de software a utilizar.

Capítulo 3 : Descripción de la Solución Propuesta

Introducción

Este capítulo describe el sistema a desarrollar desde la perspectiva de las necesidades del cliente, con una visión práctica del sistema a construir, en función de las dificultades, necesidades y características del cliente, y aplicando las técnicas a utilizar descritas en el capítulo anterior.

A continuación se relacionan las reglas del negocio, modelo de dominio, requisitos funcionales y no funcionales así como los requerimientos del sistema capturados como casos de uso según establece el Proceso Unificado de Software.

3.1 Reglas del Negocio

El sistema sólo soporta mallas triangulares.

La malla no debe tener más de 1800 polígonos.

3.2 Modelo Del Dominio

El modelo que aparece a continuación refleja la relación entre los conceptos involucrados en la deformación de cuerpos incluidos los conceptos relativos al FEM que no será implementado en esta versión, pero debido a su influencia en la arquitectura del producto final, se consideró pertinente incluirlo en este modelo.

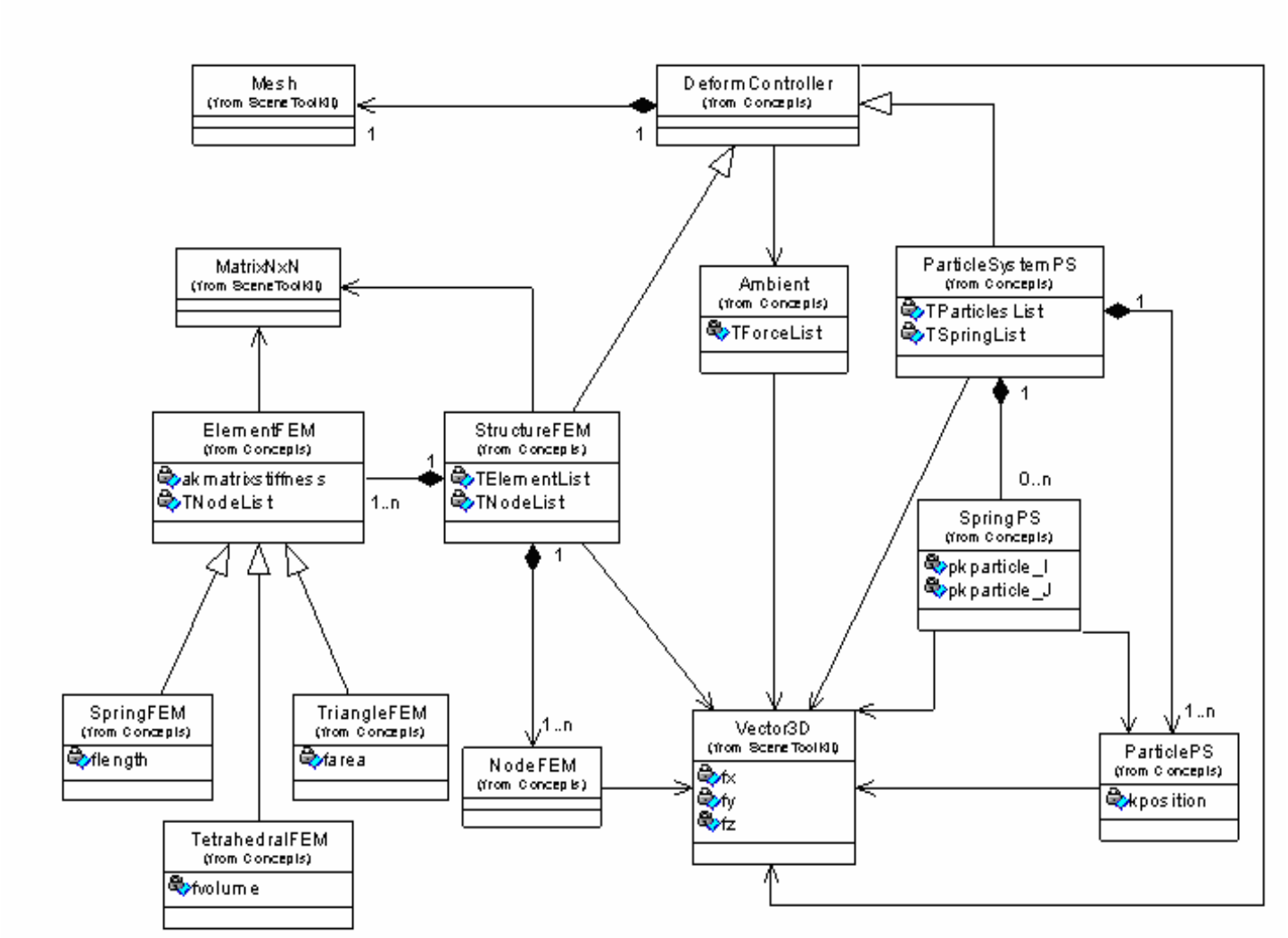


Fig. 11: Modelo de Dominio.

3.3 Captura de Requisitos

A continuación se expondrán los requisitos funcionales y no funcionales del sistema.

3.3.1 Requisitos Funcionales

1. Definir un nodo como objeto deformable.
    1.1. Verificar si es un nodo de geometría.
    1.2. Tomar la información de la malla del nodo.
    1.3. Crear un controlador de deformación para el nodo.
    1.4. Proporcionar las constantes según el método.
    1.5. Agregar el controlador de deformación al nodo.
2. Crear modelo matemático.
    2.1. Recorrer la lista de vértices de la malla.
    2.2. Recorrer la lista de conectividad de la malla.
    2.3. Generar partícula.
    2.4. Generar muelle.
    2.5. Generar cara triangular.
3. Calcular volumen.
4. Aplicar presión sobre las caras.
    4.1. Calcular presión sobre cada cara.
5. Calcular parámetros de las caras.
    5.1. Calcular normal de cada cara.
    5.2. Calcular módulo de la normal de cada cara.
6. Calcular deformación.
    6.1. Calcular fuerza elástica de cada muelle.
    6.2. Calcular la aceleración de cada partícula.
    6.3. Calcular la velocidad de cada partícula.
    6.4. Calcular la nueva posición de cada partícula.
7. Actualizar Malla
    7.1. Recorrer lista de partículas.
    7.2. Actualizar posición de vértices en la malla.
8. Aplicar fuerza.

8.1. Buscar partículas a afectar.

9. Actualizar fuerza
   9.1. Identificar la partícula.
   9.2. Sumar a la fuerza resultante.

3.3.2 Requisitos No Funcionales

Usabilidad: Los futuros usuarios del sistema serán programadores con conocimientos básicos de programación gráfica y de la terminología afín. El producto debe estar concebido para que el usuario piense en qué desea hacer y no en cómo hacerlo, por lo que éste requerimiento debe estar presente en alto grado en el producto final.

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: Compatibilidad con tarjetas gráficas de la familia NVIDIA (Quadro FX 500/FX 600).

Diseño e implementación: Debe utilizar transparentemente la biblioteca gráfica OpenGL y DirectX, en su primera versión, 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 Modelado de Caso de Uso del Sistema

En esta sección se reconocen los posibles actores del sistema a desarrollar y se conciben, a través de la agrupación de los requisitos funcionales anteriormente hallados, los posibles resultados de valor que le pueda brindar a sus actores, o lo que es lo mismo, los casos de uso del sistema.

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

3.4.1 Actor del sistema

Tabla 1: Actores del Sistema.

Actores	**Justificación**
Programador	Es el que se beneficiará con las funcionalidades que brinda el módulo: definir objetos deformables, aplicarle una fuerza a dicho objeto y deformar dicho objeto.

3.4.2 Casos de Uso del Sistema

Tabla 2: CU1 Definir nodo como deformable.

CU1	Definir nodo como deformable.
Actor	Programador.
Descripción	Determinar que un objeto se pueda deformar según la acción de una fuerza externa.
Referencia	R1

Tabla 3: CU2 Crear modelo matemático.

CU2	Crear modelo matemático.
Actor	CU1
Descripción	Crear un modelo matemático según el método de deformación escogido.
Referencia	R2, CU1(*include*)

Tabla 4: CU3 Calcular volumen.

CU3	Calcular volumen.
Actor	CU STK Ciclo de Escena.
Descripción	Calcular el volumen del sistema de partículas.
Referencia	R3, CU5(*include*)

Tabla 5: CU4 Calcular presión.

CU4	Calcular presión.
Actor	CU STK Ciclo de Escena.
Descripción	Calcular presión en el interior del sistema de partículas.
Referencia	R4, CU5 (*include*), CU9(*include*)

Tabla 6: CU5 Calcular parámetros de las caras.

CU5	Calcular parámetros de la cara.
Actor	CU3, CU4.
Descripción	Calcular la normal y su módulo para cada cara del sistema.
Referencia	R5

Tabla 7: CU6 Calcular deformación.

CU6	Calcular deformación.
Actor	CU STK Ciclo de Escena
Descripción	Determinar la deformación que sufre el objeto.
Referencia	R6, CU9(*include*)

Tabla 8: CU7 Actualizar Malla.

CU7	Actualizar Malla.
Actor	CU STK Ciclo de Escena
Descripción	Actualizar la posición de cada vértice de la malla que conforma al objeto deformable.
Referencia	R7

Tabla 9: CU8 Aplicar Fuerza.

CU8	Aplicar fuerza
Actores	Programador
Descripción	Modificar la fuerza que actúa sobre una o varias partículas del sistema.
Referencia	R8, CU9(*include*)

Tabla 10: CU9 Actualizar Fuerza.

CU9	Actualizar fuerza
Actores	
Propósito	Modificar la fuerza que actúa sobre la partícula del sistema.
Referencia	R9

3.4.3 Diagrama de Casos de Uso del Sistema

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

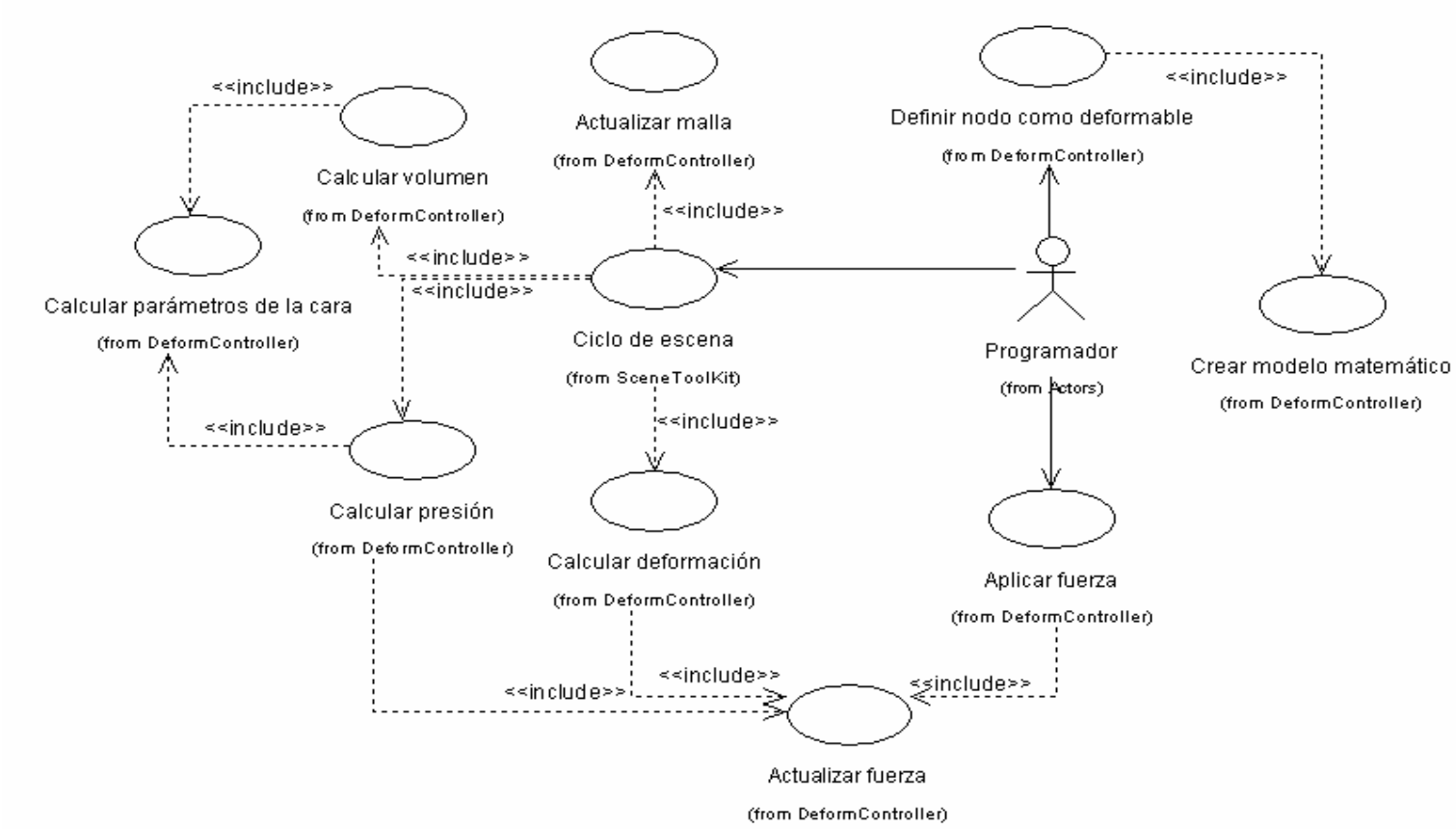


Fig. 12 Diagrama de Casos de Uso del Sistema

3.4.4Expansión de Casos de Uso

Tabla 11: Expansión CU1.

Caso de uso		
Nombre	Definir nodo como deformable.	
Actores	Programador	
Propósito	Determinar que un objeto se pueda deformar según la acción de una fuerza externa.	
Resumen:	El caso de uso se inicia cuando el programador determina que nodo de la escena va a ser deformado.	
Referencias	R1	
Precondiciones	Que estén cargados los nodos de la escena.	
Poscondiciones	Un controlador de deformación controlando al nodo.	
Curso Normal de los Eventos		
Acción del actor		**Respuesta del sistema**
1- El programador solicita establecer un nodo como objeto deformable proporcionando el nodo que desea controlar.		
		2-Verificar si el nodo contiene una geometría.
		3-Tomar la información de la malla del nodo.
		4-Se construye un controlador de deformación para el nodo.
5- Proporciona las constantes según el método de deformación.		
		6- Valida las constantes.
		7-Se le agrega el controlador de deformación al nodo.
Curso Alternativo:		
		3- Termina el caso de uso sin establecer

	objeto deformable porque el nodo no contiene geometría.
Prioridad:	Crítico.

Tabla 12: Expansión CU2.

Caso de uso	
Nombre	Crear modelo matemático.
Actores	CU2 Definir método de deformación
Propósito	Crear un modelo matemático según el método de deformación escogido.
Resumen:	El caso de uso se inicia cuando se define el método para deformar y este hace la llamada a ajustar malla para crear el modelo matemático que solucionará la deformación.
Referencias	R2, CU1 (*include*)
Precondiciones	Que exista un objeto deformable que tenga definido el método con el cual se trabajará.
Curso Normal de los Eventos	
Acción del actor	**Respuesta del sistema**
	1- Recorre la lista de vértices de la malla.
	2- Crea partículas a partir de los vértices, sin repetir.
	3- Recorre la lista de conectividad de la malla.
	4- Crea muelles a partir de las aristas, sin repetir.
	5- Crear cara a partir de la conectividad y las partículas.
Prioridad:	Crítico.

Tabla 13: Expansión CU3.

Caso de uso		
Nombre	Calcular volumen.	
Actores	CU STK Ciclo de escena.	
Propósito	Calcular el volumen del cuerpo definido por el sistema de partícula.	
Resumen:	El caso de uso se inicia en cada ciclo de render con el objetivo de obtener el volumen del cuerpo.	
Referencias	R3, CU5(*include*)	
Precondiciones	Que el objeto deformable sea volumétrico.	
Curso Normal de los Eventos		
Acción del actor		**Respuesta del sistema**
		1- Ejecutar CU5.
		2- Recorrer lista de caras.
		3- Aplicar Método de Gauss.
Prioridad:	Secundaria.	

Tabla 14: Expansión CU4.

Caso de uso		
Nombre	Calcular presión.	
Actores	CU STK Ciclo de escena.	
Propósito	Calcular la presión en el interior del sistema de partículas.	
Resumen:	El caso de uso se inicia en cada ciclo de render con el objetivo de aplicar la fuerza que se ejerce como presión desde el interior del cuerpo.	
Referencias	R4, CU5 (*include*), CU9(*include*)	
Precondiciones	Que el objeto deformable sea volumétrico.	
Curso Normal de los Eventos		
Acción del actor		**Respuesta del sistema**
		1- Ejecutar CU5.
		2- Recorrer la lista de caras
		3- Calcular la fuerza sobre cada cara.
		4- Calcular fuerza sobre cada partícula de la cara.
		3- Ejecutar CU9.
Prioridad:	Secundaria.	

Tabla 15: Expansión CU5.

Caso de uso		
Nombre	Calcular parámetros de la cara.	
Actores	CU3, CU4.	
Propósito	Calcular la normal y su módulo para cada cara del sistema.	
Resumen:	El caso de uso se inicia cuando se calcula el volumen del cuerpo y se ejecuta nuevamente para aplicar la presión sobre las caras del sistema de partículas.	
Referencias	R5	
Precondiciones	Que el objeto deformable sea volumétrico.	
Curso Normal de los Eventos		
Acción del actor		**Respuesta del sistema**
		1- Calcular dos vectores partiendo de dos aristas.
		2- Calcular normal como producto cruzado sobre los vectores de las aristas.
		3- Calcular el módulo de la normal.
Prioridad:	Secundaria.	

Tabla 16: Expansión CU6.

Caso de uso		
Nombre	Calcular deformación.	
Actores	CU STK Ciclo de Escena	
Propósito	Determinar la deformación que sufre el objeto.	
Resumen:	El caso de uso se inicia cuando se ejecuta una aplicación que contiene nodos deformables, este CU garantiza el cálculo de la deformación del cuerpo según las fuerzas que lo afectan y el tiempo de la aplicación.	
Referencias	R6, CU9(*include*)	
Precondiciones	Que esté definido un objeto deformable.	
Curso Normal de los Eventos		
Acción del actor		**Respuesta del sistema**
		1- Calcular la fuerza elástica que ejercen los muelles sobre las partículas que enlazan mediante la Ley de Hooke.
		2- Recorrer lista de partículas.
		3- Ejecutar CU9.
		4- Calcular la aceleración de cada partícula según la Segunda Ley de Newton.
		5- Calcular la velocidad de cada partícula.
		6- Calcular la nueva posición de cada partícula.
Prioridad:	Crítico.	

Tabla 17: Expansión CU7.

Caso de uso		
Nombre	Actualizar Malla.	
Actores	CU STK Ciclo de Escena.	
Propósito	Actualizar la posición de cada vértice de la malla que conforma al objeto deformable.	
Resumen:	El caso de uso se inicia cuando ha ocurrido una deformación en el objeto lo que implica que hubo cambio en la posición de los vértices que conforman la malla y deben ser relocalizados.	
Referencias	R7	
Precondiciones	Que haya ocurrido deformación.	
Curso Normal de los Eventos		
Acción del actor		**Respuesta del sistema**
		1- Verificar si hubo deformación.
		2- Recorre la lista de partículas
		3- Para cada partícula recorre la lista de vértices que controla.
		4- Cada partícula actualiza la posición de los vértices que controla.
Curso Alternativo:		
		2- Si no hubo deformación, no continua el caso de uso.
Prioridad:	Crítico.	

Tabla 18: Expansión CU8.

Caso de uso		
Nombre	Aplicar fuerza	
Actores	Programador	
Propósito	Modificar la fuerza que actúa sobre una o varias partículas del sistema.	
Resumen:	El caso de uso se inicia cuando el programador ejerce una fuerza externa al sistema de partículas, definiendo la o las partículas afectadas.	
Referencias	R8, CU9	
Precondiciones		
Acción del actor		**Respuesta del sistema**
1- Solicitar aplicar fuerza a una o varias partículas, definiendo la fuerza que se le ejerce a las mismas.		
		2- Ejecuta CU9.
Prioridad:	Crítico	

Tabla 19: Expansión CU9.

Caso de uso		
Nombre	Actualizar fuerza.	
Actores		
Propósito	Modificar la fuerza que actúa sobre la partícula del sistema.	
Resumen:	El caso de uso se inicia cuando algún CU requiere de modificar la fuerza que actúa sobre una partícula.	
Referencias	R6, R3 (*extend*), R5 (*extend*)	
Precondiciones		
Acción del actor		**Respuesta del sistema**
		1- Identificar la partícula que se le aplica la fuerza.
		2- Suma a la fuerza resultante de la partícula la fuerza ejercida.
Prioridad:	Crítico	

Conclusiones

En el capítulo que concluye se definieron las expectativas que el cliente tiene respecto al sistema que se construye. Quedaron establecidos los requisitos funcionales y no funcionales y descritos los casos de uso que reflejan las necesidades del cliente.

Capítulo 4 : Diseño e implementación del Sistema

Introducción

En este capítulo se encuentran el diagrama de clases de diseño del sistema propuesto y los diagramas de secuencia que reflejan la realización de los casos de uso descritos en el capítulo anterior así aspectos a tener en cuenta para la integración del producto a la STK, se incluye además el diagrama de componentes de implementación.

4.1 Características de la Scene Toolkit (STK)

Para la comprensión del sistema será imprescindible el estudio de algunas características de la STK en aras de un acoplamiento eficiente. Una escena de la STK puede estar compuesta por objetos de diferentes tipos contenidos todos en nodos del grafo de escena, que es la estructura encargada del manejo de los objetos del ambiente a simular, dígase cámara, luces, objetos estáticos, dinámicos, deformables, animados… 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. Los controladores en la STK están asociados a nodos y son quienes pueden alterar sus diferentes estados.

Resultan de interés las clases “CGeometryNode” de la STK que es la clase de nodos que contienen las geometrías de los objetos en forma de mallas triangulares o “CTriMesh” necesarias para la construcción de un Sistema Masa-Resorte, así como la clase “CController” de quien heredará el controlador de deformación encargado de alterar el estado geométrico del objeto CTriMesh, contenido en un CGeometryNode, en función de la deformación realizada.

4.2 Diseño del Sistema

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

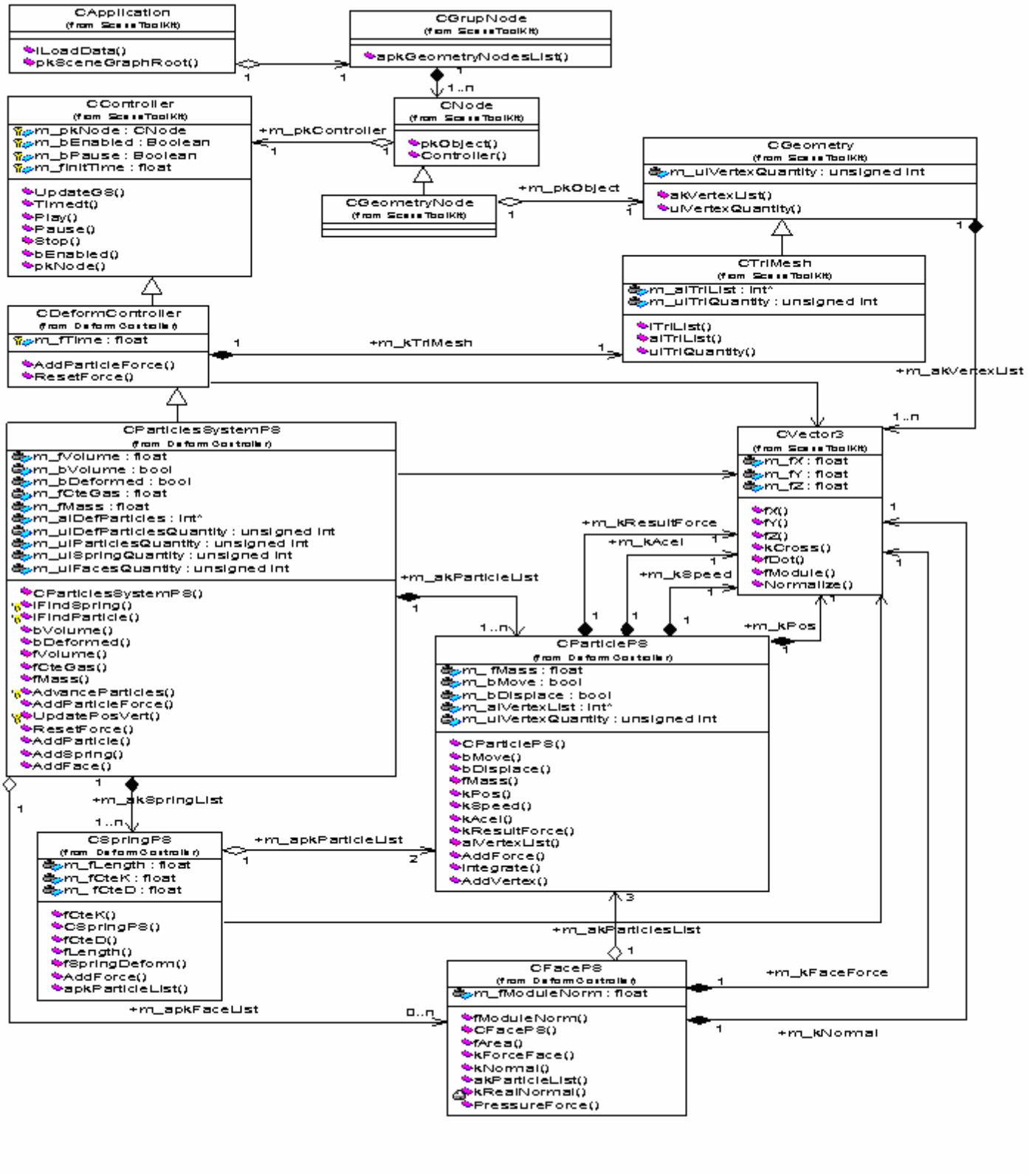


Fig. 13 Diagrama de Clases de Diseño

4.3 Descripción de las Clases de Diseño

Tabla 20: Descripción de la clase CDeformController

Nombre: CDeformController	
Tipo de clase: Controladora	
Atributo	**Tipo**
m_pkMesh	CTriMesh*
m_fDTime	float
Para cada responsabilidad:	
Nombre:	CDeformController()
Descripción:	Constructor de la clase, inicializa m_fDTime en 0.0.
Nombre:	fDt()
Descripción:	Método de acceso a miembro, m_fDTime es la variación del tiempo entre frames consecutivos.
Nombre:	AddParticleForce(unsigned int arg_uiPosPart, CVector3 arg_kForce)
Descripción:	A implementar el la clase CParticleSystemPS con el objetivo de agregar una fuerza arg_kForce sobre una partícula arg_uiPosPart.
Nombre:	ResetForce()
Descripción:	A implementar el la clase CParticleSystemPS con el objetivo de anular todas las fuerzas que afectan el cuerpo.

Tabla 21: Descripción de la clase CParticlesSystemPS

Nombre: CParticlesSystemPS	
Tipo de clase: Controladora	
Atributo	**Tipo**
m_bVolume	bool
m_fVolume	float
m_bDeformed	bool
m_fCteGas	float
m_fMass	float
m_aiDefParticles	int*
m_uiDefParticlesQuantity	unsigned int
m_uiParticlesQuantity	unsigned int
m_uiSpringQuantity	unsigned int
m_uiFaceQuantity	unsigned int
m_akParticleList	CParticleSP*
m_akSpringList	CSpringPS*
m_akFaceList	CFacePS*
Para cada responsabilidad:	
Nombre:	iFindSpring(CVector3 arg_kPos1, CVector3 arg_kPos2)
Descripción:	Función encargada de retornar la posición en la lista de muelles de un muelle que una a las partículas en las posiciones que se pasan como parámetro y y en caso de no existir, retornar -1.
Nombre:	iFindParticle(CVector3 arg_kPos)
Descripción:	Función encargada de retornar la posición en la lista de partículas de una partícula en la posición que se pasan como parámetro y en caso de no existir, retornar -1.
Nombre:	AdvanceParticles(float arg_fDt)
Descripción:	Según una variación de tiempo arg_fDt entre frames ejecuta el algoritmo de deformacion para cada muelle, partícula y cara que conforman el modelo en cada ciclo de render.
Nombre:	UpdatePosVert()

Descripción:	Función responsable de la actualización de los vértices de la malla según las posiciones de los nodos del sistema masa-resorte luego de una deformacion.
Nombre:	CParticleSystemPS(CTriMesh* arg_pkMesh, float arg_fCteGas, bool arg_bVolume, float arg_fCteK, float arg_fCteD, float arg_fMass);
Descripción:	Constructor del objeto. Crea el modelo de masa- resorte según pkMesh, creando una partícula en cada posición de la malla sin repetir posiciones, un muelle en cada arista y una cara sobre cada cara. Asigna las constantes de elasticidad, amortiguación y masa, especifica además si el sistema es cerrado (arg_bVolume == true) o si no posee volumen.
Nombre:	akParticles()
Descripción:	Método de acceso a miembro m_akParticleList, lista de partículas del sistema.
Nombre:	akSprings()
Descripción:	Método de acceso a miembro m_akSpringList, lista de muelles del sistema.
Nombre:	akFaceList ()
Descripción:	Método de acceso a miembro m_akFaceList, lista de caras del sistema.
Nombre:	uiDefParticlesQuantity ()
Descripción:	Método de acceso a miembro m_uiDefParticlesQuantity, cantidad de partículas que han sufrido deformaciones.
Nombre:	uiParticlesQuantity ()
Descripción:	Método de acceso a miembro m_uiParticlesQuantity, cantidad de partículas del sistema.
Nombre:	uiSpringQuantity ()
Descripción:	Método de acceso a miembro m_uiSpringQuantity, cantidad de muelles del sistema.
Nombre:	uiFaceQuantity ()
Descripción:	Método de acceso a miembro m_uiFaceQuantity, cantidad de caras del sistema.
Nombre:	bVolume ()
Descripción:	Método de acceso a miembro m_bVolume, que especifica si el cuerpo es volumétrico o no.
Nombre:	bDeformed ()

Descripción:	Método de acceso a miembro m_bDeformed, que especifica si el cuerpo ha sufrido deformación o no.
Nombre:	fCteGas ()
Descripción:	Método de acceso a miembro m_fCteGas, constante del gas a utilizar en la ecuación del gas ideal, para mantener proporción entre presión interna y volumen.
Nombre:	fVolume ()
Descripción:	Funcion encargada de calcular el volumen del cuerpo representado según el Metodo de Gauss y actualizar la variable m_fVolume.
Nombre:	AddParticleForce(unsigned int arg_uiPosPart, CVector3 arg_kForce);
Descripción:	Es responsable de sumar la fuerza arg_kForce a la fuerza resultante de la partícula en la posición arg_uiPosPart.
Nombre:	ResetForce()
Descripción:	Se encarga de eliminar todas las fuerzas externas que influyen en el sistema poniendo la fuerza resultante de la partícula en cero.
Nombre:	UpdateGS ()
Descripción:	Definida en la clase base CController, se encarga de actualizar el estado geométrico del cuerpo, para ello ejecuta el algoritmo de deformación y luego el método UpdatePosVert().

Tabla 22: Descripción de la clase CParticlePS

Nombre: CParticlePS	
Tipo de clase: Entidad	
Atributo	**Tipo**
m_fMass	bool
m_bMove	bool
m_bDisplace	float
m_kPos	CVector
m_kSpeed	CVector
m_kAccel	CVector
m_kResultForce	CVector
m_aiVertexList	Int *
m_uiVertexQuantity	Unsigned int
Para cada responsabilidad:	
Nombre:	CParticlePS(CVector3 arg_kPos, bool arg_bMove)
Descripción:	Constructor del objeto, asigna una posición a la partícula y establece si es fija o no.
Nombre:	fMass()
Descripción:	Método de acceso a miembro m_fMass.
Nombre:	bMove()
Descripción:	Método de acceso a miembro m_bMove, indica si la partícula es fija o puede moverse.
Nombre:	bDisplace ()
Descripción:	Método de acceso a miembro m_bDisplace, indica si la partícula sufrió desplazamiento.
Nombre:	aiVertexList ()
Descripción:	Método de acceso a miembro m_aiVertexList, arreglo de las posiciones que ocupan en la lista de vértices de la malla los vértices que corresponden a esta partícula.
Nombre:	uiVertexQuantity ()
Descripción:	Método de acceso a miembro m_uiVertexQuantity, cantidad de vértices que corresponden a esta partícula.

Nombre:	kPos ()
Descripción:	Método de acceso a miembro m_kPos, posición de esta partícula.
Nombre:	kSpeed ()
Descripción:	Método de acceso a miembro m_ kSpeed, velocidad de esta partícula.
Nombre:	kAcelerate ()
Descripción:	Método de acceso a miembro m_ kAcelerate, aceleración de esta partícula.
Nombre:	kResultForces ()
Descripción:	Método de acceso a miembro m_ kResultForces, sumatoria de las fuerzas que afectan a esta partícula.
Nombre:	AddForce(CVector3 arg_kForce)
Descripción:	Suma arg_kForce con m_ kResultForces, actualizando así la sumatoria de las fuerzas que afectan a esta partícula.
Nombre:	ResetForces ()
Descripción:	Establece m_ kResultForces como cero, eliminando así las fuerzas que afectan esta partícula.
Nombre:	Integrate(float arg_fDt)
Descripción:	Calcula m_kPos partiendo del valor de arg_fDt y de m_kResultForce mediante la Segunda Ley de Newton y las leyes de la cinemática.

Tabla 23: Descripción de la clase CSpringPS

Nombre: CSpringPS	
Tipo de clase: Entidad	
Atributo	**Tipo**
m_apkParticleList[2]	CParticlePS*
m_fLength	float
m_fCteK	float
m_fCteD	float
Para cada responsabilidad:	
Nombre:	CSpringPS(CParticlePS *arg_pkFirst, CParticlePS *arg_pkLast, float arg_CteK, float arg_CteD)
Descripción:	Constructor del objeto, establece un muelle con constantes de elasticidad y amortiguación arg_CteK y arg_CteD respectivamente, enlazando las partículas arg_pkFirst y arg_pkLast.
Nombre:	fLength ()
Descripción:	Método de acceso a miembro, m_fLength es la longitud inicial del muelle.
Nombre:	fCteK ()
Descripción:	Método de acceso a miembro, m_fCteK es la constante de elasticidad del muelle.
Nombre:	fCteD ()
Descripción:	Método de acceso a miembro, m_fCteD es la constante de amortiguación del muelle.
Nombre:	pkParticle (int arg_ipos)
Descripción:	Método de acceso a miembro, m_apkParticleList es el arreglo que contiene las dos partículas y retorna la partícula en 0 < arg_ipos < 2
Nombre:	fSpringDeform ()
Descripción:	Calcula la longitud real del muelle después de una deformación como la distancia entre las dos partículas que une.
Nombre:	AddForce ()
Descripción:	Actualiza la resultante de las fuerzas de las partículas que une luego de calcular la fuerza que ejerce mediante la Ley de Hooke.

Tabla 24: Descripción de la clase CFacePS

Nombre: CFacePS	
Tipo de clase: Entidad	
Atributo	**Tipo**
m_apkParticlesList[3]	CParticlePS*
m_fModuleNorm	float
m_kForceFace	CVector3
m_kNormal	CVector3
Para cada responsabilidad:	
Nombre:	kRealNormal()
Descripción:	Calcula la normal de la cara, a ser usado en el cálculo de volumen.
Nombre:	CFacePS(CParticlePS* arg_pkPart1, CParticlePS* arg_pkPart2, CParticlePS* arg_pkPart3);
Descripción:	Constructor de la clase, construye una cara triangular con vértices en las posiciones de arg_pkPart1, arg_pkPart2 y arg_pkPart3.
Nombre:	fModuleNormal ()
Descripción:	Calcula el modulo de la normal de la cara obtenida mediante kRealNormal().
Nombre:	kNormal()
Descripción:	Retorna un vector unitario en la dirección de la normal de la cara.
Nombre:	kForceFace ()
Descripción:	Método de acceso a miembro, m_kForceFace es la fuerza que ejerce el gas en el interior del cuerpo sobre la cara.
Nombre:	fArea ()
Descripción:	Calcula el área de la cara.
Nombre:	PresureForce(float arg_fCteGas, float arg_fVolume)
Descripción:	Actualiza la resultante de las fuerzas de las partículas luego de calcular m_kForceFace, cada particular recibe 1/3 de esa fuerza.
Nombre:	pkParticle (int arg_ipos)
Descripción:	Método de acceso a miembro, m_apkParticleList es el arreglo que contiene las tres partículas que definen una cara y retorna la partícula en 0 < arg_ipos < 3

4.4 Diagramas de secuencia

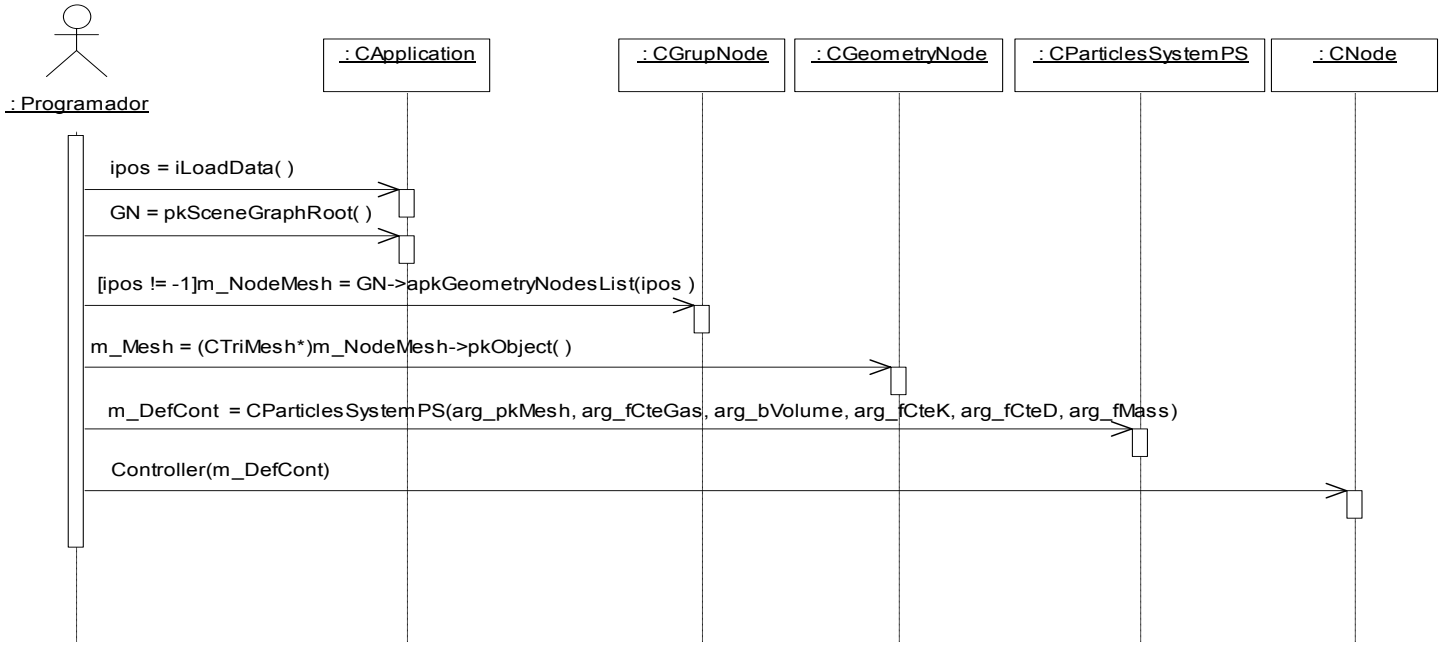


Fig. 14: Diagrama de Secuencia para el CU Definir nodo como deformable.

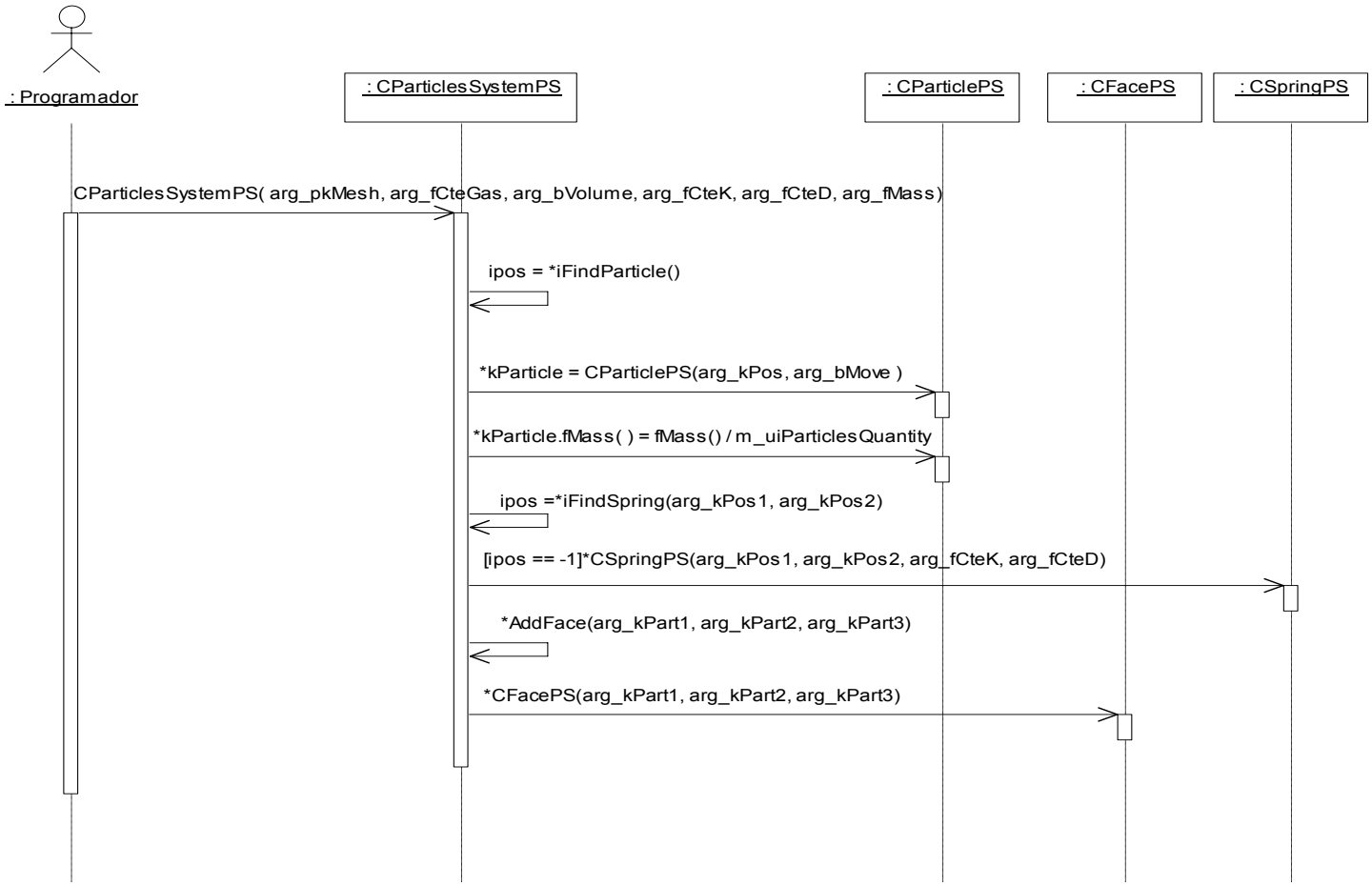


Fig. 15: Diagrama de Secuencia para el CU Crear modelo matemático.

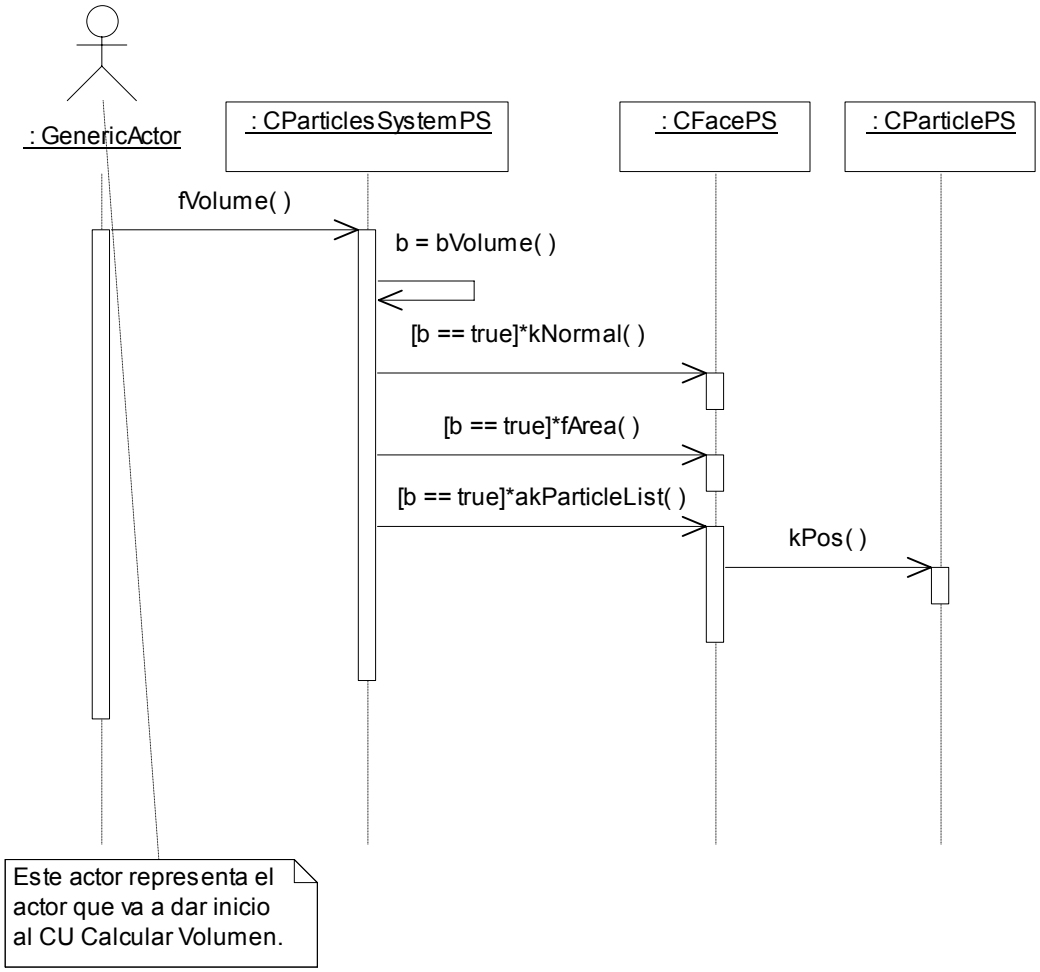


Fig. 16: Diagrama de Secuencia del CU Calcular volumen.

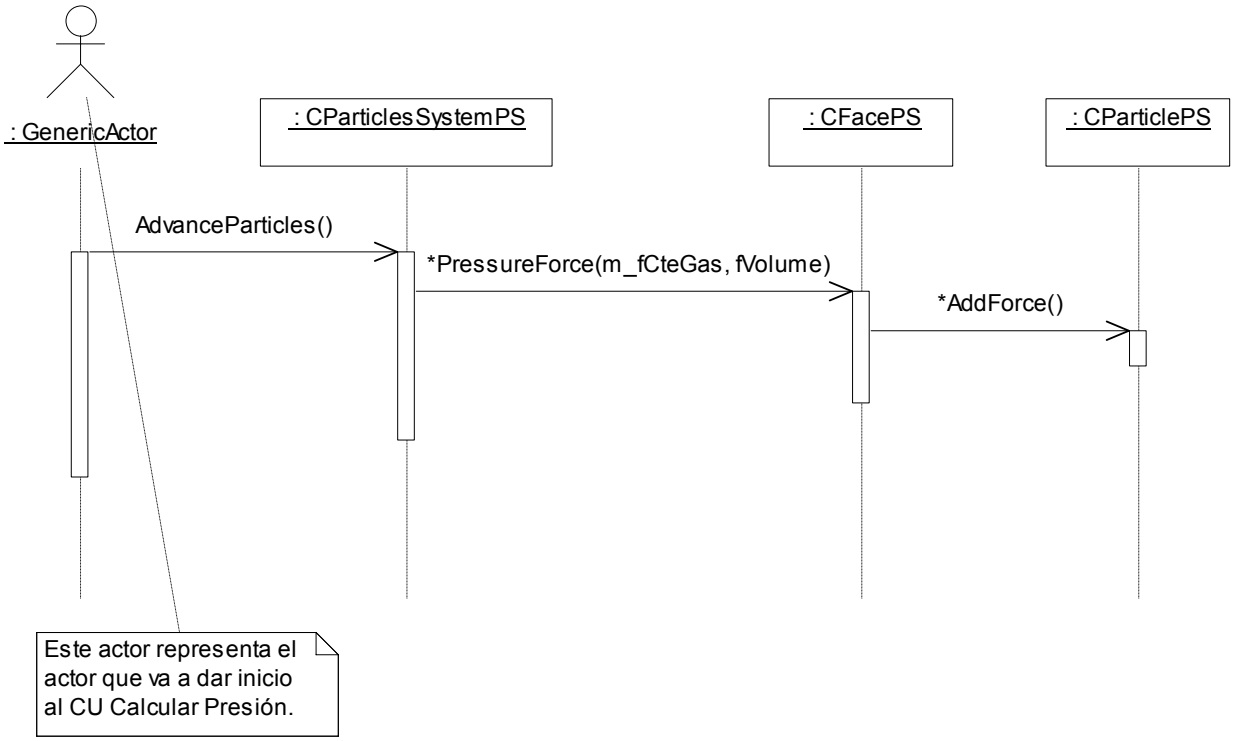


Fig. 17: Diagrama de Secuencia del CU Calcular presión.

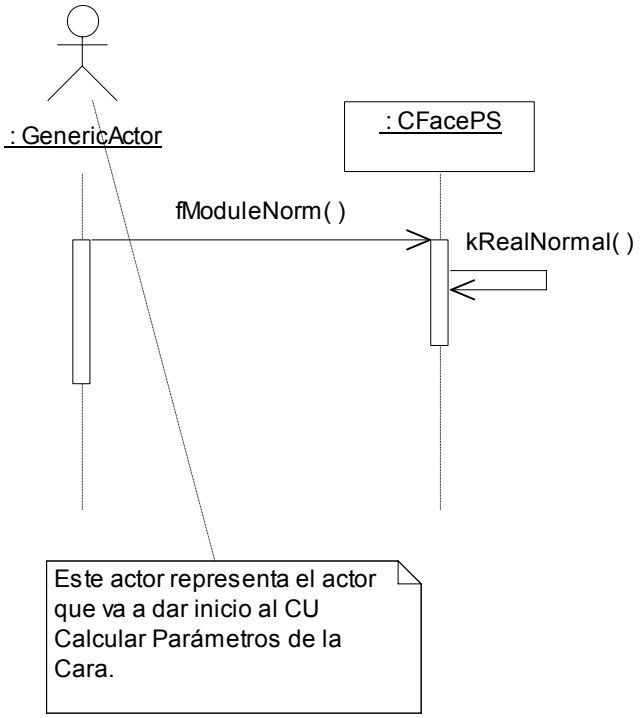


Fig. 18: Diagrama de Secuencia del CU Calcular parámetros de la cara.

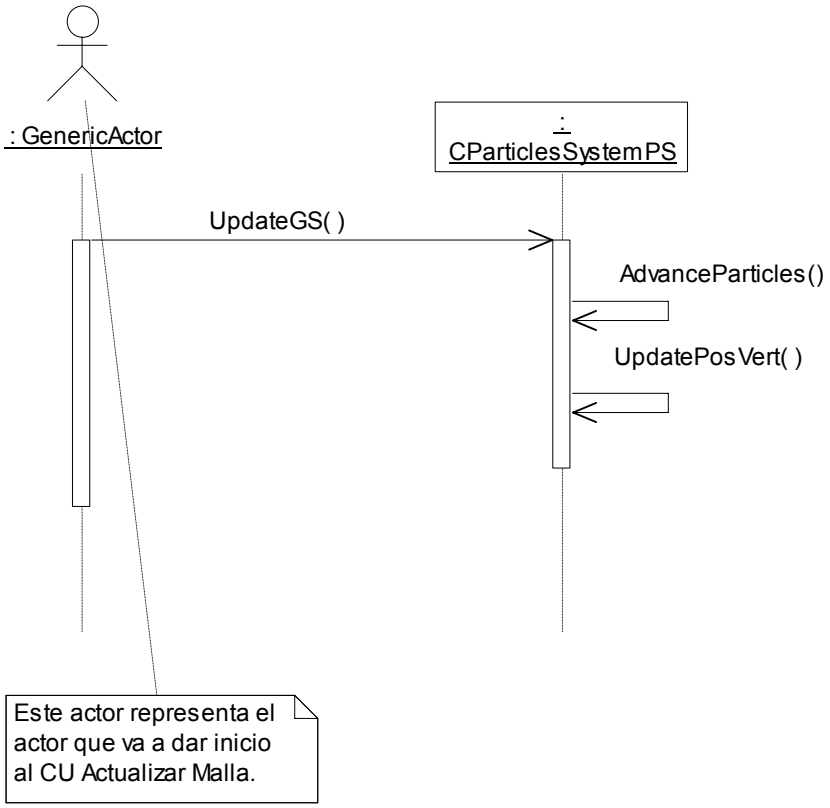


Fig. 19: Diagrama de Secuencia del CU Actualizar malla.

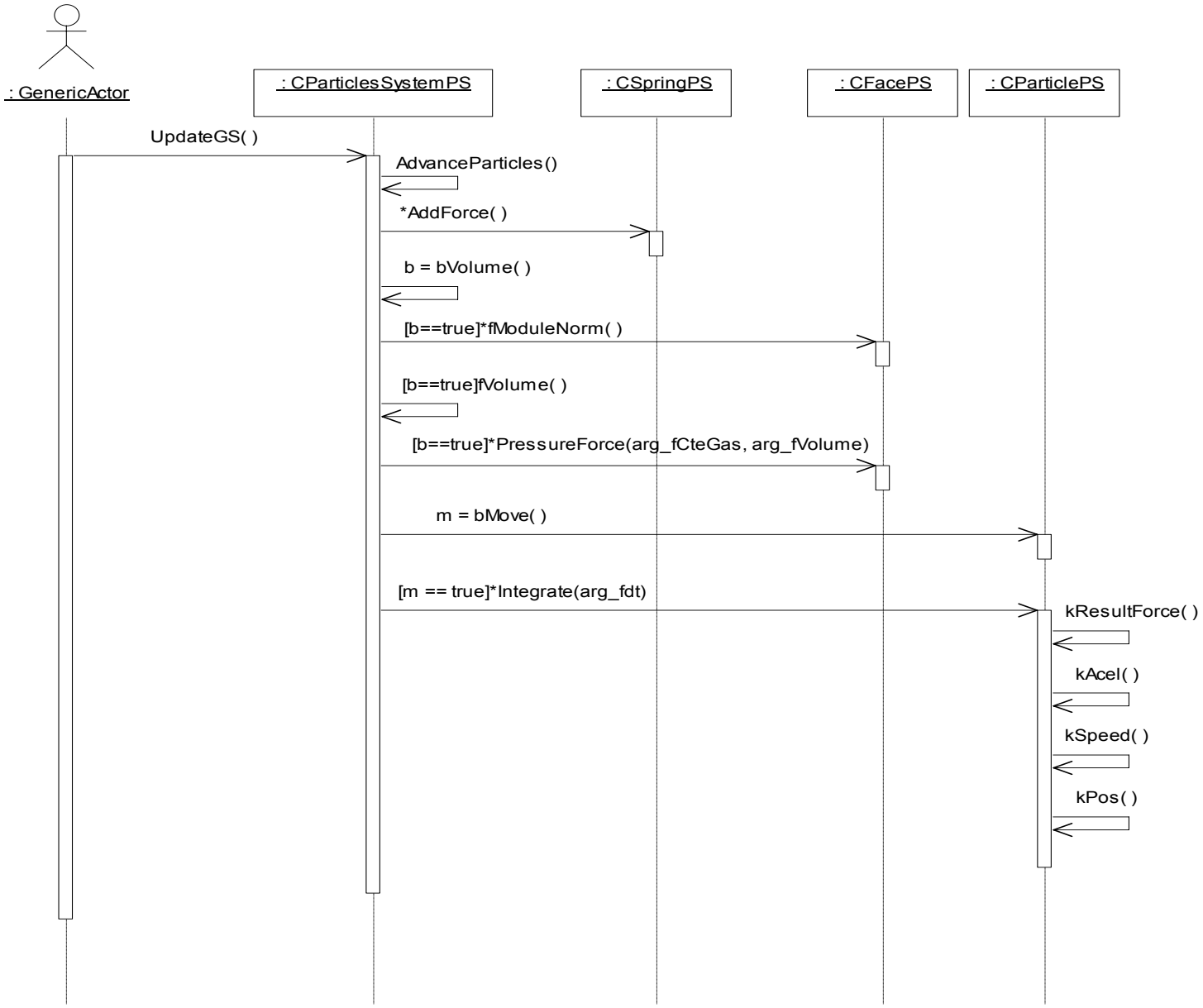


Fig. 20: Diagrama de Secuencia del CU Calcular deformación

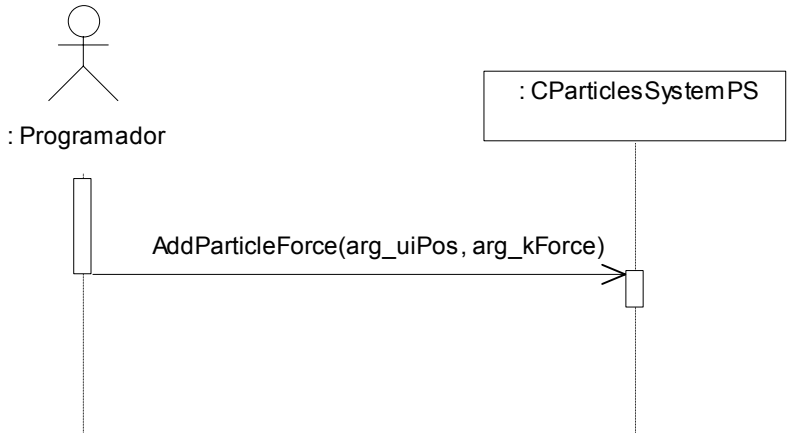


Fig. 21: Diagrama de Secuencia del CU Aplicar Fuerza.

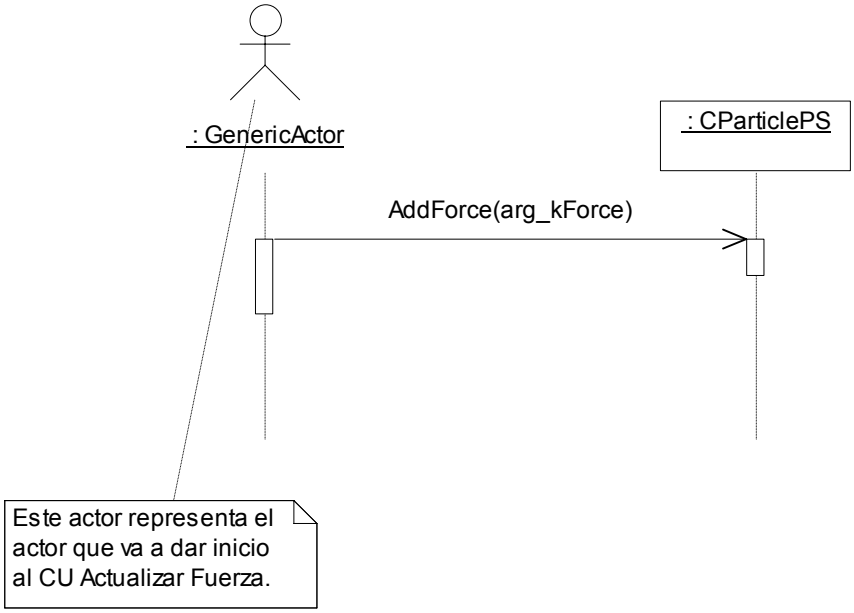


Fig. 22: Diagrama de Secuencia del CU Actualizar fuerza.

4.5 Diagrama de Componentes

Fig. 23: Diagrama de relación entre paquetes.

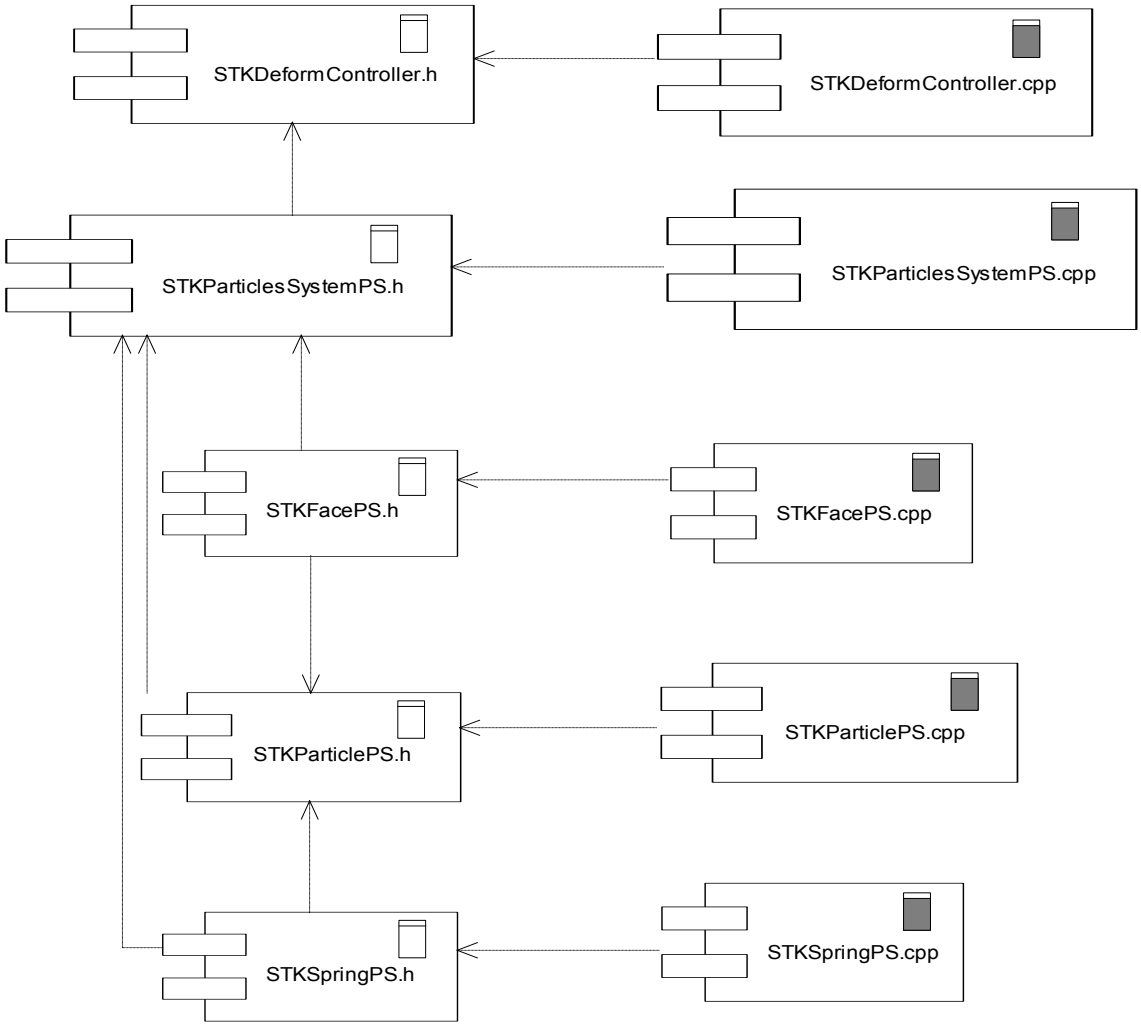


Fig. 24: Diagrama de Componentes.

Conclusiones

Al concluir este capítulo se tiene el diseño completo del sistema detalladamente concebido y el algoritmo de deformación traducido a mensajes entre clases, contiene además elementos de implementación del proyecto como el diagrama de componentes.

Conclusiones

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

Finalmente se propone una solución para cuerpos deformables con comportamiento en tiempo real y factible para ser usada en simulación quirúrgica. Descrita en términos de los flujos de trabajo de requerimientos, diseño e implementación establecidos por RUP y acoplada a la STK.

Recomendaciones

Para complementar este trabajo y obtener aproximaciones cada vez más reales, recomendamos el estudio profundo del FEM para lograr la implementación de alguna variante en tiempo real, implementar métodos de integración de las partículas que converjan mejor. Estudiar el enmallado volumétrico eficiente de los cuerpos en busca de un mejor comportamiento.

Referencias bibliográficas

[1] Haptica ” About simulation” 2006 http://www.haptica.com/id55.htm

[2] Góngora Arango J C, “La cirugía del siglo XXI. Nuevas tecnologías” 2005 http://www.encolombia.com/medicina/cirugia/ciru16401pre-cirugia.htm

[3] Downes M, Hsu A, Steele M. “A virtual environment for training laparoscopic cholecystectomy.” University of California, Berkely. CS294-5 Virtual Reality, Spring Semester, 1997

[4] Basdogan, Ho, Srinivasan, Small and Dawson. “Force Interactions in Laparoscopic Simulations: Haptic Rendering of Soft Tissues.” Proceedings of the Medicine Meets Virtual Reality Conference, San Diego, CA, Jan 1998.

[5] Cotin S, Delingette H, Ayache N. “Real Time Volumetric Deformable Models for Surgery Simulation.” Visualization in Biomedical Computing (Proc. VBC ‘ 96), K.H. Hhne, R. Kikinis (eds), Lecture Notes in Computer Science, vol. 1131, Springer- Verlag, 1996.

[6] Player R, Blank B, Cornelius N et al. “Integrated Haptics Applications: Surgical Anastomosis and Aircraf Maintenance.” Preprints of The First Phantom User’s Group Workshop, Sept. 27 – 30, Cambridge, M.A, 1996.

[7] LaSSo (Laparoscopic Surgery Simulator) Project http://www.vision.ee.ethz.ch/projects/Lasso/start.html

[8] Székely G et al. “Modelling of Soft Tissue Deformation for Laparoscopic Surgery Simulation.” Swiss Federal Institute of Technology. Zürich, Switzerland. February 27, 1999.

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

[10] Székely G et al. “Virtual Reality-Based Simulation of Endoscopic Surgery”. Swiss Federal Institute of Technology. Zürich, Switzerland. June 3, 2000.

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

[12] Witking A. “Physically Based Modeling. Particle System Dynamics.” Pixar Animation Studios 2001.

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

[14] Prietoni N. “Physically Based Deformable Objects in Computer Graphics.” Universidad de Geneva, Diciembre 2005.

[15] Gibson S. F. F. “3D ChainMail: a Fast Algorithm for Deforming Volumetric Objects”. A Mitsubishi Electric Research Lab, 1997.

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

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

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

[19] Guo X, Qin H “Point-Based Dynamic Deformation and Crack Propagation” Stony Brook University, New York.

[20] Gross M. “Getting to the Point?” Computer Graphics Laboratory ETH Zürich, Switzerland 2006.

[21] Sense8 CORPORATION TEAM, “WorldToolkit Release 9”*, Reference Manual.* Sense8 Corporation (www.sense8.com). USA. 1999.

[22] LUNA, Frank D, “Introduction to 3D Game Programming with DirectX” WordWare Publishing, Inc. USA. 2003.

[23] BIRN, Jeremy, “Digital lightning and rendering” New Riders Publishing, USA 2000.

[24] ASTLE, Dave y HAWKING, Kevin, “OpenGL Game Programming” Prima Tech Publishing, USA 2001.

[25] Bucciarelli Jr. Louis L. “Engineering Mechanics for Structures” http://web.mit.edu/emech/dontindex-build/index.html 2002.

[26] LaMothe A, “Tricks of the 3D Game Programming Gurus” Indianapolis, SAMS, 2003, 1601.

[27] Terzopoulos D, Waters K “Physically-Based Facial Modeling, Analysis, and Animation” Journal of Visualization and Computer Animation, 1(2):73–80, 1990.

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

[29] Nedel L. P., Thalmann D. “Real Time Muscle Deformations Using Mass-Spring Systems” Swiss Federal Institute of Technology, Computer Graphics Lab. Lausanne Switzerland 1998.

[30] Teschner M., Heidelberger B., Müller M., Gross M.” A Versatile and Robust Model for Geometrically Complex Deformable Solids” Computer Graphics Laboratory, ETH Zurich, Switzerland, 2004.

[31] Schill M. A., Gibson S. F. F., Bender H. J., Männer R. “Biomechanical Simulation of Vitreous Humor in the Eye Using Enhanced ChainMail Algorithm”, MERL, 1998.

[32] Hirota G., Maheshwari R., Lin M.”Fast Volume-Preserving Free Form Deformation Using Multi-Level Optimization” Department of Computer Science, University of North Carolina, USA 1999.

[33] Hsu M., Hughes J, Kaufman H. “Direct Manipulation of Free-Form Deformations” Cambridge Research Lab Brown University, 1992.

[34] Da Silva S. “A Virtual Surgery Environment” Institute of Computational Science, ETH Zürich, Switzerland 2003.

[35] MacCracken R., Joy K."Free-Form Deformations With Lattices of Arbitrary Topology" Computer Graphics Research Laboratory, Department of Computer Science, University of California, USA.

[36] 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

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

[38] Kobbelt L., Botsch M." A Survey of Point-Based Techniques in Computer Graphics" Computer Graphics Group, RWTH Aachen University, Germany 2004.

[39] Debunne G., Barr A., Desbrun M., Cani M. "Dynamic Real-Time Deformations using Space & Time Adaptive Sampling", In Computer Graphics Proceedings Annual Conference Series, ACM SIGGRAPH 2001.

[40] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B. "Stable real-time deformations" In Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2002.

[41] Bro-Nielsen M., Cotin S. "Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation" Technical University of Denmark, Lyngby, Denmark 1996.

[42] Al-khalifah A., Roberts D. "Survey of modeling approaches for medical simulators" Centre for Virtual Environments, the University of Salford, Manchester, UK 2004.

[43] Baur, C., Guzzoni, D., & Georg, O. "Virgy: A virtual reality and force feedback based endoscopy surgery simulator" EPFL (Swiss Federal Institute of Technology) 1015 Lausanne, Switzerland 1998.

[44] Downes,M., Cavusoglu, M., Gantert, W.,Way, L., & Tendick, F. "Virtual environments for training critical skills in laparoscopic surgery." Proc. MMVR'98, IOS Press 1998.

[45] Turner, M. J., Clough, R. W., Martin, H. C., Topp, L. J.“Stiffness and deflection analysis of complex structures”, J. Aero., 1956.

[46] Frisoli A, Borelli L F, Stasi C, Bellini M, Bianchi C, Ruffaldi E, Di Pietro G, Bergamasco M.” Simulation of real-time deformable soft tissues for computer assisted surgery” The International Journal of Medical Robotics & Computer Assisted Surgery 14/4/04.

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

[48] O’Brien J. F. “Graphical Modeling and Animation of Fracture” Georgia Institute of Technology, USA July 2000.

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

[50] Doug L. J., Dinesh K. P. “Accurate Real Time Deformable Objects” University of British Columbia, Canada, 1999.

[51] Koppel D., Wang Y., Chandrasekaran Sh.” Toward Real-Time, Physically-Correct Soft Tissue Behavior Simulation” University of California, USA 2003.

[52] Chen Yan, Zhu Qing-hong, Kaufman Arie. “Physically-based Animation of Volumetric Objects” Center for Visual Computing and Department of Computer Science, State University of New York at Stony Brook, 1999.

[53] Keeve E, Girod S, Pfeifle P, Girod B. “Anatomy-Based Facial Tissue Modeling Using the Finite Element Method”. Telecommunications Institute, Department of Oral and Maxillofacial Surgery, University of Erlangen-Nuremberg Glückstr, Germany, Noviembre 1996.

[54] Hong M, Welch S, Jung S, Choi M. “Fast Volume Preservation for a Mass-Spring System” Soonchunhyang University, South Korea, University of Colorado at Denver and Health Sciences Center, 2006.

Anexo 1: Glosario de Términos

A

Aliasing: En gráficos por ordenadores, es el proceso por el cual curvas y otras líneas aparecen separadas, debido a una reducción de la resolución del dispositivo gráfico

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.

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.

Elasto-mecánico: Combinación de los comportamientos elástico y mecánico de un cuerpo.

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

H

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

L

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

M

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

O

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

P

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

R

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

Render: Proceso de obtención de imágenes por computadora.

S

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

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

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

T

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.

Anexo 2: Estándares de codificación

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

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

Nombre de los ficheros:

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

STKNameOfUnits.cpp

Constantes:

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

Tipos de datos:

Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados:

enum EMyEnum {ME_VALUE, ME_OTHER_VALUE};

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

enum ENodeType {NT_GEOMETRYNODE,…};

Estructuras: struct SMyStruct {…};

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

Clases: class CClassName;

Indicando con “C” que es una clase

Interfaces: IMyInterface

Indicando con “I” que es una interfaz.

Listas e iteradotes STD: vector<> TNameList;

TNameList::iterator TNameListIter;

map<> TNameMap;

TNameMap::iterator TNameMapIter;

multimap<> TNameMultiMap;

TNameMultiMap::iterator TNameMultiMapIter;

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 bVarName;

int iName;

unsigned int uiName;

float fName;

char cName;

char* acName; // arreglo de caracteres

char* pcName; // puntero a un char

char** aacName; // bidimensional

char** apcName; // arreglo de punteros

bool m_bMemberVarName; //variable miembro

char gcGlobalVarName; //variable global, no se le antepone ""

short sName;

Instancias de tipos creados:

EMyEnumerated eName;

SMyStructure kName;

CClassName kObjectName;

CClassName* pkName; //puntero a objeto

CClassName* akName; //arreglo de objetos

CClassName* akName; // variable miembro de clase

IMyInterface* pIName; //puntero interfaces

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

Constructor y destructor:

CClassName (bool arg_bVarName, float& arg_fVarName);

~CClassName ();

Funciones:

bool bFunction1 (…);

int* piFunction2 (…);

CClassName* pkFunction3 (…);

Procedimientos:

void Procedure4 (…);

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

int iMyVar; //variable

Obtención del valor:

int iMyVar();

{

```
  return iMyVar;
}
```

Establecimiento del valor:

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

Obtención y establecimiento del valor:

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

Índice de figuras

Índice de tablas