UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5

"Editor de Propiedades Físicas para Sistemas de Realidad Virtual"

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

Autores: Susej Beovides Luis

Yurian Diaz Capote

Tutor: Lic. Juan M. Medero Martínez

Ciudad de la Habana

Junio 2008

“.....hay una sola forma de triunfar en algo y eso es dar todo de ti mismo.

.....la alegría está en la lucha, en el esfuerzo, y no en la victoria misma.

Nuestra recompensa se encuentra en el esfuerzo y no en el resultado.

Un esfuerzo total es una victoria completa."

Anónimo

Declaración de Auditoría

Declaramos que somos los únicos autores de este trabajo, y autorizamos a los proyectos 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 firmo la presente a los ____ días del mes de _____ del año _____.

Autores:

Susej Beovides Luis

_____

Yurian Diaz Capote

_____

Tutores:

Ing. Juan M. Medero Martín

_____

Datos de Contacto

Autores

Nombre: Susej Beovides Luis
Correo Electrónico: sbeovides@estudiantes.uci.cu

Nombre: Yurian Diaz Capote
Correo Electrónico: ydcapote@estudiantes.uci.cu

Tutor

Nombre: Juan Manuel Medero Martínez
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Licencido en Ciencias de la Matemática.
Categoría Docente: Profesor Instructor
Correo Electrónico: juanm@uci.cu

Dedicatoria

...a mi madre que además de darme la vida, ha estado siempre pendiente de mis luchas diarias,
...a mi padre por su ejemplo y cariño,
...a mi hermano lindo por ser la persona más dulce que he conocido,
Los AMO, gracias por ser mi luz aún en días de sol.

Yurian.

A la memoria de mi abuela Sara.

.....a mi madre, a mi hermana Suly y a mi padrastro Carly.

Por ser mis fuerzas y las razones por las que vivo y lucho en la vida.

Los quiero mucho.
Susej.

Agradecimientos Colectivos

Agradecer es querer abarcar con palabras las inmensidades del corazón, es un acto de nobleza, de reconocimiento...

Aprovechamos este espacio para dar las gracias a todos aquellos que de una forma u otra contribuyeron a la realización de este trabajo, a nuestra formación profesional y a ser cada día mejor persona.

A la Revolución por crear proyectos tan genuinos y dar la posibilidad del disfrute colectivo de los mismos, como lo es la Universidad de las Ciencias Informáticas.

A Fidel por ser el paradigma de todo hombre de bien, por ser nuestro siempre invicto Comandante en Jefe y tener ideas tan brillantes para no detenernos aún cuando el camino parezca difícil.

A la UCI por prepararnos para la Vida y permitirnos pasar en ella los mejores momentos de nuestras vidas como universitarios.

A Juan Manuel por darnos su apoyo incondicional en cada momento, por su carácter jovial, por su paciencia.

A chicos como Guille, Leonel, David, Alfredo que nos ayudaron siempre que lo necesitamos, y ¡fueron muchas!

A los profes que nos dedicaron parte de su preciado tiempo sin objeción, Alexito, Liudmila, Yanosky, Yoisy, Dania.

A los amigos del grupo, que nos han acompañado siempre durante estos 5 años llegando a convertirse en parte de nuestra familia.

A mi mamá, por ser mi vida, mis ojos, mi sol, mi todo, por guiarme siempre por el camino correcto, por su ejemplo, por su amor, por ser la luz que me guía en cada paso.....

A mi hermana Suley, por ser mi inspiración, por quererme tanto, por permitirme ser su ejemplo...

A mi padrastro, por quererme como una hija, por su apoyo constante, por estar siempre....

A mi abuela Sara, donde quiera que se encuentre, por ser parte de mí, por haberme enseñado a dar pasos fuertes en la vida, por educarme y adorarme a cada momento....

A mi abuelo Luis, por su ayuda incondicional, por su preocupación constante, por la ternura brindada....

A mi papá, por su inteligencia, por su audacia ante la vida, por su afecto.....

A mi hermana Sandra por su cariño infinito, por su inocencia, por su perseverancia.....

A mi tía Sarita, a mis abuelos paternos, a mamá, por apoyarme, por quererme.....

A mis tíos, mis primos, en general a toda mi familia que me ha apoyado en todo momento.

A mis amigos de siempre en especial a Medero, Osley, Yuniel, Leidys, Walkis, Yester, Bladimir, Yoelvys, por los fines de semanas distintos, por las vacaciones maravillosas, por las llamadas sorpresivas....

A todos mis amigos, en especial a Yaself, Papita, Daire, Niche, Tamara, por todos los momentos felices, por estar cada día y a cada hora dispuestos a todo, por quererme como soy.....

A Yurian, por enseñarme que la amistad verdadera existe, por cada segundo de alegría, por demostrarme que todo puede llegar a ser diferente, por quererme como a una hermana, por su apoyo constante y sincero, por ser siempre y por siempre amiga.....

A Yamel, por demostrarme que hay cosas que pueden volver a sentirse y encontrarse, por amarme, por aguantarme mis malacrianzas, por cada minuto de amor y alegría, por hacerme feliz, por estar en mi vida....

A mis suegros y a mi cuñada por el cariño, por haberme acogido como si fuera parte de su familia....

A mis amigos de la FEU en especial a Yunia, Javier, Serguey, César, Raicel, por aprender juntos a hacer muchas cosas en poco tiempo, por las tantas madrugadas, por hacerme mejor persona....

A mis amigos y compañeros de grupo desde primer año, por las fiestas, por la alegría compartida, por los momentos de apoyo.....

A mi tutor, por las noches de dudas y revisiones, por ser más que tutor, un buen amigo.....

A todos los profes que han contribuido en mi formación como profesional y en especial a la decana por su ternura, por su protección y respaldo.....

A todos los que en algún momento han formado parte de mi vida.....

A todos.....Gracias!!!!!

Susej

A mis padres Margarita y José por darme siempre la fuerza que necesito para seguir adelante, por su preocupación, cariño y ejemplo, por su constancia durante estos 5 años para que hoy pudiera comprender que en la vida para llegar a ser alguien hay que sacrificarse mucho,

...gracias a los dos por ser mi vida, no sé qué sería de mí sin ustedes.

A mi hermano lindo por todas las alegrías que me hace vivir, por estar siempre a mi lado, apoyándome y ser tan cariñoso conmigo.

A mis abuelos que ya no están conmigo por ser mi inspiración en algunos momentos.

A mima Migdalia por ser mi segunda mamá y quererme tanto.

A mis primas Darelys y Dianelys por estar siempre tan pendientes de mi y ayudarme tanto, las quiero.

...en fin, a toda mi familia.

A Wendy por ser mi amiga de todos los tiempos, por haberme enseñado que en la vida todo lo que sucede conviene, y que no importa la distancia solo tenemos que confiar, gracias mi hermana...

A Medardo por ofrecerme su amistad sin condición, y darme tantos consejos como si fuera un hermano.

A Dayamí por acompañarme desde la infancia.

A mis amigos de la vocacional René, Yany y Yurién... con nosotros no existe el olvido jeje.

A las amigas(os) especiales que he encontrado en esta universidad y que han estado conmigo en las buenas y malas, sin ustedes todo sería muy aburrido: Papita, Daire, Lily, Beatriz, Tamara, Yunia, Yariel, Yaself y Maikel...los quiero.

A Jhonatan por aguantarme todas mis malcriadeces, por quererme tanto y estar siempre que lo necesito.

A ti Susy por ser más que una amiga, una hermana, por estar siempre a mi lado, por ayudarme a ser fuerte, por tu confianza, tu cariño, tu alegría, por soportar mis cambios de ánimo que son muchos,

...a tus padres mis gracias, por quererme y guiarme como otra hija más.

...sin tu apoyo este sueño no se habría echo realidad.

A todos mis compañeros de aula por haber convivido juntos durante estos 5 años y ser parte de esta travesía hasta el final, siempre los recordaré.

A mi tutor Juan Manuel un millón de gracias por su paciencia conmigo.

A todos los profes que contribuyeron a mi formación profesional y personal.

A Dios por darme la fe. ...a todos mil gracias de todo corazón.

Yurian.

Resumen

El surgimiento y desarrollo de los Sistemas de Realidad Virtual en la última década promete la solución a problemas que parecían lejos del alcance humano. Simular un proceso que en condiciones normales requiere gran exactitud es algo sumamente complejo. Este trabajo tiene como objetivo el desarrollo de un Editor de Propiedades Físicas para los Sistemas de Realidad Virtual (SRV) que permita adicionarle indicadores físicos a objetos 3D útiles en la simulación de escenas virtuales de manera que los cuerpos mostrados se comporten tal y como se describen en tiempo real, logrando un mayor grado de realismo en todos los entornos que se representan. Actualmente los parámetros físicos se introducen de forma directa en el código fuente lo cual resulta bastante engorroso a la hora de hacer cualquier cambio, por tanto el Editor facilitará estas modificaciones de forma independiente y con la misma eficiencia.

Se utiliza el Visual Studio 2003, y la herramienta SceneToolKit, incluyendo un Módulo para la Simulación de la Dinámica de los Cuerpos Rígidos en Entornos Virtuales que esta tiene acoplado. Se tiene como proceso de desarrollo de software el Proceso Unificado Racional (RUP), que unido al Enterprise Architect, sustentado sobre las últimas especificaciones del diseño UML2.1, constituyen la metodología a utilizar.

Como resultado se obtuvo un editor de propiedades físicas para el trabajo con ficheros que representen cuerpos rígidos, donde a través de un modelo físico-matemático se actualiza el estado de los cuerpos a través del tiempo y atiende el proceso de acción y reacción a las colisiones que pueden ocurrir entre ellos o con el medio que le circunda, lo cual permite optimizar las animaciones virtuales en cuanto a visibilidad y grado de realismo en las simulaciones que se realizan.

Índice de Contenido

Introducción 1

Capítulo 1 : Fundamentación Teórica 6

Introducción 6
1.1 Principales características de un fichero gráfico 3D. 7
1.2 Formatos de almacenamiento más usados y ficheros que lo utilizan. 7
1.3 Principales características de algunos de los ficheros 3D en forma Binario. 8
1.3.1 STL (Standard Tessellation Language). 8
1.3.2 3DS. 9
1.3.3 3DX. 11
1.3.4 STK. 12
1.4 Características físicas de los cuerpos. 16
1.4.1 Comportamiento físico de cuerpos rígidos. 18
1.4.2 Comportamiento físico de cuerpos deformables. 20
1.4.3 Métodos de deformación basados en física. 21
1.4.4 Sistema Masa Resorte. 22
1.4.5 Método de Elementos Finitos (FEM). 24
1.4.6 Método de Elementos Frontera (BEM). 27
1.5 Principales características de algunos editores gráficos. 28
1.5.1 Antecedentes y funcionamiento de algunos editores de propiedades físicas. 29

Capítulo 2 : Soluciones Técnicas 32

Introducción 32
2.1 Consideraciones Específicas. 33
2.2 Propuesta Física. 33
2.2.1 Cuerpos Rígidos. 34
2.2.2 Cuerpos Deformables. 34
2.3 Fichero a cargar. 36
2.4 Forma de Salvado. 36
2.5 Estructura y nombre del fichero físico. 37
2.5.1 Características del bloque Header. 38
2.5.2 Características del bloque PhysicGeometry. 38
2.5.3 Características del bloque PhysicModel. 39
2.6 Consideraciones técnicas generales. 39
2.6.1 Metodología. 39
2.6.2 Enterprise Architect 40
2.6.3 Implementación. 40

Capítulo 3 : Descripción de la Solución Propuesta 42

Introducción 42
3.1 Modelo de Dominio 43
3.1.1 Glosario de términos del modelo de dominio. 43
3.2 Especificación de los requisitos del software. 45
3.2.1 Requisitos Funcionales. 45
3.2.2 Requisitos No Funcionales. 46
3.3 Modelo de Caso de Uso del Sistema 46
3.3.1 Actores del sistema 46
3.3.2 Casos de uso del sistema 47

3.3.3 Diagrama de Casos de Uso del Sistema. 48
3.3.4 Descripción de los Casos de Uso del Sistema 49
Capítulo 4 : Diseño e Implementación del Sistema 56
Introducción 56
4.1 Diseño lógico del sistema. 57
4.2 Diagrama de Paquetes de Diseño. 58
4.3 Diagrama de Clases de Diseño. 59
4.4 Diagramas de Secuencia. 61
4.5 Diagrama de Despliegue. 65
4.6 Diagrama de Componentes. 65
4.7 Nomenclatura y estándares de codificación. 66
Conclusiones 70
Recomendaciones 72
Referencias Bibliográficas 73
Bibliografía Consultada 75
Glosario de Abreviaturas 76
Glosario de Términos 77
Anexos 81
Índice de Figuras 102
Índice de Tablas 103

Introducción

La revolución tecnológica y científica más importante experimentada por la humanidad se está viviendo en la actualidad, y se inducen sentimientos cruzados de esperanza y temor. Los avances que ha sufrido el campo de la ciencia de la información y dentro del mismo la informática, ha permitido el comienzo de la investigación de varias ramas que evidencian cambios significativos. La Realidad Virtual (RV) es un ejemplo de ello como relevancia de la informática gráfica, donde se generan entornos sintéticos en tiempo real o representaciones de una realidad ilusoria, pues se trata de una realidad perspectiva sin soporte objetivo que solo vive dentro de un ordenador, haciendo uso de medios electrónicos y combinando ambientes tridimensionales e interactivos orientados a la visualización de objetos 3D generados por computadoras.

Actualmente, la RV se plasma en una multiplicidad de sistemas, tal es el caso de un quirófano virtual, ya en uso en la Facultad de Medicina de la Universidad Complutense de Madrid, también en la exploración espacial (en la futura estación espacial Alpha o la programación de un recorrido en Marte), en la meteorología y la arqueología, así como en la investigación genética y química (en la investigación de la estructura del átomo, hélice de ADN). También se benefician todas las profesiones que utilizan habitualmente maquetas o prototipos, ya que la RV permite al diseñador presentar a sus clientes simulaciones en tres dimensiones de la obra, antes de iniciar su realización.

Las actividades de alto riesgo constituyen otro campo donde la RV se puede convertir en un aliado invaluable, permitiendo, entre otros elementos, el entrenamiento de personal especializado sin poner en peligro su integridad física [1], el diseño y recorrido de modelos arquitectónicos, la visualización científica (Análisis de Sistemas Físicos), educación, ayuda a minusválidos, diversión y juegos electrónicos, así como también en la medicina. En esta última, la simulación digital multisensorial ha significado un cambio trascendental para complementar el desarrollo de sistemas virtuales, no sólo en lo que se refiere a los métodos de instrucción, sino también como herramienta para la práctica de nuevos procedimientos y técnicas terapéuticas y quirúrgicas, en la adquisición de experiencias en el tratamiento de patologías diferentes y en el conocimiento de las variaciones anatómicas por parte de los profesionales de la salud.

Son grandiosos los resultados que se han obtenido en el mundo gracias a la simulación 3D de procesos complejos, y al grado de exactitud de sus eventos que cada día se hace más puntual. El surgimiento y desarrollo de varios simuladores (Tiro, Juegos Virtuales, Quirúrgicos, Conducción, entre otros) que contribuyen al entrenamiento y a su vez aprendizaje, son un ejemplo de ello; y como aporte

ayudan a disminuir los riesgos de accidentes y sus consecuencias, los costos de las ejercitaciones en el terreno y prolongan los ciclos de vida útil del material provisto para cada esfera a la que pertenecen.

Dentro de los principales requisitos que debe tener un sistema de simulación se encuentra el hecho de que los objetos presentes en el entorno se comporten tal y como lo harían en la realidad al ser sometidos a fuerzas, al ser cortados o suturados, al efectuarse un movimiento que provoque cambios en su dirección y sentido, un aumento de velocidad o variabilidad en algún punto de referencia, o sea, los cuerpos que intervienen en el proceso de simulación deben tener inherentes el comportamiento físico que los identifica, con grandes exigencias tanto visual como en el sentido del tacto. Simular estos comportamientos de las entidades está determinado por el modelo físico intrínseco que se le asocie a cada cual y a su vez por un conjunto de variables o propiedades físicas características del modelo en cuestión.

En Cuba, poco a poco ha ido creciendo el desarrollo de Sistemas de Realidad Virtual (SRV), así como la investigación de ideas de avance tecnológico aplicadas en el campo de la Medicina, las Fuerzas Armadas Revolucionarias, el MININT, la Educación. Muchos centros están colaborando en pos de estos adelantos, tal es el caso del “Centro de Investigación y Desarrollo de Simuladores” (SIMPRO). La Universidad de las Ciencias Informáticas se ha unido a todo este desempeño, y la facultad 5 que tiene como perfil de desarrollo el campo de los SRV, tiene definidos varios proyectos dentro de los cuales se pueden citar: simuladores de conducción, desarrollo de juegos, simuladores para la defensa (tiro), y está concebido la creación de un simulador quirúrgico que ya se encuentra en su fase de elaboración.

Lograr un ambiente acorde al que se vive en la realidad en todos los sucesos que se simulan, incluyendo la reacciones que de ellos se derivan, con efectos, sonidos y un alto valor de credibilidad, es uno de los problemas más grandes que aqueja a los desarrolladores de estos proyectos.

Las características físicas propias de cada modelo, regidas por leyes físico-matemáticas que determinan el proceder de los cuerpos vinculados al proceso que se está representando son las que aportan el grado de realismo que debe encerrar cada escena para alcanzar el éxito en la simulación y lograr la sensibilidad humana de quienes disfrutarán del producto. Actualmente en los diseños que se han definidos soportados con la herramienta SceneToolKit, un framework desarrollado en la facultad 5 que sustenta el trabajo de varias aplicaciones de RV que se encuentran en desarrollo, estas propiedades están prefijadas con valores constantes en el código fuente, lo cual impide que puedan ser modificadas en algún momento previo, durante o luego de la ejecución; por lo que sería provechoso contar con un instrumento independiente que permita cargar los cuerpos que van a intervenir en un proceso de simulación, mostrar un comportamiento físico mediante nuevos valores de

propiedades físicas definidas, y crear un fichero donde se puedan salvar estas modificaciones. Esta situación precisamente es la que da lugar a un ***problema científico*** que se manifiesta a través de la siguiente interrogante: ¿Cómo lograr asociar, probar y salvar las propiedades físicas de cuerpos que intervienen en una simulación 3D con un comportamiento físico determinado, antes de que se simule la escena real donde estos cuerpos estarán finalmente involucrados?

El objetivo general que se propone en este trabajo es precisamente elaborar una herramienta (Editor de Propiedades Físicas) que permita asociar, probar y salvar las propiedades físicas de cuerpos 3D, según las características de los distintos modelos que permiten simular comportamientos, teniendo como ***objeto de estudio*** el análisis de las propiedades físicas de los cuerpos en su totalidad, así como algunos de los editores gráficos existentes en el mundo actual, teniendo como ***campo de acción*** el estudio de los editores de propiedades físicas para la edición de modelos 3D.

A continuación se mencionan un conjunto de tareas que permitirán dar cumplimento al objetivo formulado:

- ✓ Investigar las propiedades físicas de los modelos físicos que pueden ser utilizados para simular comportamientos en un entorno virtual.
- ✓ Estudiar características de los ficheros 3D.
- ✓ Caracterizar los distintos tipos de editores gráficos existentes.
- ✓ Implementar la funcionalidad que permita cargar los modelos 3D que representan cuerpos y entornos.
- ✓ Implementar las funcionalidades que permitan editar los valores físicos añadidos al cuerpo cargado según del modelo seleccionado, así como simular el proceso teniendo en cuenta los valores definidos y el comportamiento que se desea alcanzar.
- ✓ Definir un formato de salva de fichero.
- ✓ Implementar la funcionalidad de salvar en un fichero las propiedades físicas y sus valores definidos. Verificar que el fichero resultante contiene las propiedades físicas añadidas.

Los métodos científicos que se utilizan para el estudio precedente de este trabajo son los siguientes:

Métodos Teóricos:

Analítico-Sintético: Se usa este método ya que primeramente se busca lo que caracteriza y distingue a los proyectos de RV, los aspectos físicos que precisan la imagen en la escena; apoyados en la bibliografía y los estudios de quienes ya han abordado el tema, lo cual permitirá obtener el mayor cúmulo de información referente al problema.

Histórico-Lógico: Mediante este se grafica todo el proceso evolutivo del fenómeno en cuestión, o sea, ofrece una visión de cómo han venido manifestándose los sistemas de RV, en qué medida han aumentado su horizonte de realismo y a qué se debe precisamente este avance.

Inductivo-Deductivo: Permite llegar a un conocimiento o vía de solución al problema que se plantea, partiendo de toda la información que se ha encontrado.

Modelación Icónica: Permite hacer una reproducción simplificada de la realidad; o sea, definir un esquema de la herramienta con vista a refinarlo durante el desarrollo del proceso de investigación.

Métodos Empíricos

Observación: Permite hacer una observación minuciosa de la situación actual de los proyectos de RV, y extraer los verdaderos hechos que se derivan de la no existencia del Editor de Propiedades Físicas; esto hará posible precisar conceptos de trabajo, estructuras y posibles representaciones que vayan marcando el camino a seguir.

Entrevista: Se utiliza para poder obtener una información más específica sobre los rasgos físicos de cada modelo, apoyados en un experto en Física.

Para conocer brevemente el contenido de este trabajo a continuación se hace una pequeña descripción de cada uno de los capítulos.

-*Capítulo 1* “Fundamentación Teórica”, se hace un estudio de los diferentes formatos de ficheros 3D, profundizando en ficheros binarios, sus las particularidades. Además de analizar y definir las distintas características que rigen la dinámica física de los cuerpos, así como las características fundamentales de diferentes editores gráficos existentes.

-*Capítulo 2* “Soluciones Técnicas”, precisamente en esta sección, como su nombre lo indica, se presentan los rasgos que distinguirán el Editor de Propiedades Físicas. Se especifica él o los modelos físicos a utilizar, las propiedades que se le asignarán a los cuerpos en dependencia de su clasificación, así como se muestra la estructura y organización del fichero que se creará. Ofrece una propuesta de la posible solución del problema.

-*Capítulo 3* “Descripción de la solución propuesta”, se procede al levantamiento de requisitos del sistema haciendo un análisis más exhaustivo del objeto de estudio que se plantea.

-*Capítulo 4*, “Diseño e implementación del sistema”, se muestran los diagramas de clases de diseño agrupados por paquetes, los diagramas de secuencia, las descripciones de las clases de diseño, así como los diagramas de despliegue y de componentes.

Finalmente se expone un glosario de términos, un glosario de abreviaturas con el objetivo de facilitar la comprensión del lenguaje del que se ha hecho uso, un índice de figuras y otro de tablas para tener mejor organización de estos elementos, así como las referencias bibliográficas por si el lector siente la necesidad de ampliar sus conocimientos sobre el tema, además de un conjunto de anexos que amplían el contenido referenciado en algunos capítulos.

1

Capítulo 1 : Fundamentación Teórica

Introducción

El desarrollo de simuladores se ha convertido en uno de los mayores avances de la ciencia y la tecnología en los últimos tiempos. Simular un procedimiento implica la necesidad de que este proceso se de como realmente sucede en la vida, no basta con que la manipulación ocurra en tiempo real, sino que además la forma debe parecer lo suficiente similar al suceso cotidiano. Para ello es importante que los ficheros 3D que representan cuerpos geométricos, posean una estructura bien definida y que el comportamiento que se le defina en el ambiente de simulación tenga el mayor grado de credibilidad posible.

En este capítulo se hará un estudio de los diferentes formatos de ficheros 3D, profundizando en ficheros binarios y sus particularidades. Además se analizan y se definen las distintas características que rigen la dinámica física de los cuerpos rígidos y deformables que toman parte en un entorno simulado.

1.1 Principales características de un fichero gráfico 3D.

Un fichero gráfico 3D debe almacenar la información de la escena que representa, ya sea de un objeto en específico o de un mundo virtual en su totalidad. Debe contar con todos los atributos que faciliten el entendimiento de la estructura de la malla que lo define, sus vértices y caras en el caso más básico y además suelen tener un conjunto de características que brindan un mayor nivel de detalle de las escenas, entre los cuales podemos citar, los materiales, los colores, los vértices, las normales de las caras, los estados de render, entre otros.

Otro aspecto importante a tener en cuenta es el formato de almacenamiento y la organización de los atributos en el fichero, este último influye sin lugar a dudas en la complejidad, tiempo de carga y facilidad de entendimiento del mismo.

1.2 Formatos de almacenamiento más usados y ficheros que lo utilizan.

Se define un formato de almacenamiento como un conjunto de reglas (algoritmo) que precisa la manera de almacenar datos en memoria. [2]

Hoy en día son diversos los formatos de almacenamiento que se utilizan para el trabajo de ficheros 3D, entre los más utilizados encontramos el ASCII y el Binario, a continuación se citan algunos ejemplos de ficheros gráficos que emplean estos formatos de almacenamientos:

Un fichero **ASCII** o texto estándar es un fichero en el que la información que contiene está escrita en caracteres en código ASCII. Son ficheros que contienen texto sin formato, y que se pueden visualizar con cualquier editor. Por ejemplo, los ficheros cuyo nombre tiene la extensión .txt, .html o .htm son ficheros ASCII.

Dentro de los ficheros que usan este tipo de formato para almacenar la información de sus entornos están:

- ✓ STL
- ✓ OBJ
- ✓ ASE
- ✓ DIRECTX

Un fichero **Binario** (archivo binario), contrariamente a un fichero ASCII, más que simplemente texto, puede contener fotos, sonido, hojas de cálculo, o documentos concebidos para el procesamiento de texto. Los ficheros binarios están formados de unos y ceros.

Dentro de los ficheros que usan este tipo de formato para almacenar la información de sus entornos están:

- ✓ STL
- ✓ MD3
- ✓ DIRECTX
- ✓ 3DS
- ✓ 3DX

Otro fichero que también usa el formato binario para la organización de su información es el **.STK**, elaborado recientemente en la UCI con el fin de dar solución a uno de los problemas más grandes que se presenta a la hora de recrear un entorno virtual, que es el de generar y cargar archivos gráficos que cumplan con los requisitos necesarios, ya sea disminuir el tamaño de los archivos o facilitar la carga y puesta en escena de los mismos.

1.3 Principales características de algunos de los ficheros 3D en forma Binario.

1.3.1 STL (Standard Tessellation Language).

El formato de archivo estándar del prototipo rápido es el Archivo .STL. Este tipo de archivo utiliza una malla de pequeños triángulos sobre las superficies para definir la forma del objeto. Su uso principal está dado en la creación de prototipos físicos desde diseños generales en un ordenador o desde datos 3D procesados.

Debido a que los archivos ASCII STL pueden ser muy grandes debe ser escrito byte a byte por el software que haga uso de él, existe una versión **binaria** de STL que puede ser leído (y editado) con un simple editor de texto.

Esta forma usa la representación numérica de puntos enteros y flotantes de IEEE. [3]

Estructura del fichero.

- ✓ string Comentario: un comentario al inicio de fichero.
- ✓ unsigned long int: número de la cara.
- ✓ float i: normal en i.
- ✓ float j: normal en j.
- ✓ float k: normal en k.
- ✓ **float x**: coordenada para el primer vértice respecto al eje x.
- ✓ **float y**: coordenada para el primer vértice respecto al eje y.
- ✓ **float z**: coordenada para el primer vértice respecto al eje z.
- ✓ **float x**: coordenada para el segundo vértice respecto al eje x.
- ✓ **float y**: coordenada para el segundo vértice respecto al eje y.
- ✓ **float z**: coordenada para el segundo vértice respecto al eje z.
- ✓ **float x**: coordenada para el tercer vértice respecto al eje x.
- ✓ **float y**: coordenada para el tercer vértice respecto al eje y.
- ✓ **float z**: coordenada para el tercer vértice respecto al eje z.
- ✓ **unsigned int c**: cuando está en cero indica que terminó la información de la cara.

Lo que aparece en **negrita** se repite en dependencia de la cantidad de caras que tenga el objeto. Los dos últimos bytes se usan para especificar que aquí termina la información de una cara [3].

1.3.2 3DS.

El 3DS es creado originalmente para 3D StudioMax por Autodesk. Se encuentra en forma de binario y presenta una estructura muy completa. [4]

- ✓ **Estructura del fichero.**

Generalmente todos los tipos de ficheros están conformados por bloques y casi todos presentan una cabecera que indica donde localizar estos bloques y en qué orden se encuentran, pero el fichero 3DS

no muestra estas características, el formato 3DS puede tener bloques en cualquier orden y no presenta una cabecera que especifique donde encontrarlos. Veamos el siguiente ejemplo. [4]

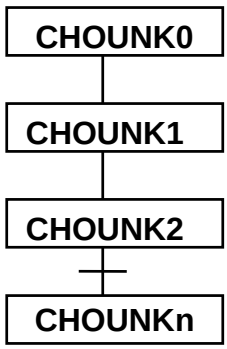


Fig 1. 1: Estructura del 3DS.

Chounk: significa bloque de datos.

El fichero 3DS no sólo está dividido en pequeños bloques, sino que cada bloque está dividido en subbloques y lo mejor de todo es que no tiene que existir un orden en el fichero, por ello es que a simple vista es muy difícil entender su estructura. [4] Cada uno de estos bloques que contienen algunos subbloques se pueden ver en el **Anexo 1**.

El fichero presenta una cabecera en cada bloque que contiene un identificador y el tamaño del bloque, entonces dado el tipo de bloque y su tamaño se maneja en el programa que carga el fichero. Un bloque puede contener mallas, luces y cámaras, y los subbloques vértices, coordenadas de texturas, etcétera. En caso de que un bloque contenga información que no es de interés, simplemente se salta por encima de él. A continuación se muestra la estructura de la cabecera de un bloque.

Header

unsigned short m_usID: contiene un identificador de 2 bytes que específica qué información contiene este bloque, ejemplo: datos de vértices, entre otros.

unsigned int m_uiLength: muestra el tamaño del bloque incluyendo la cabecera.

Los atributos son representados por estructuras de datos y cada uno presenta los datos necesarios para los mismos, por ejemplo: la estructura de la cara es como se muestra a continuación. [4]

Face

unsigned short m_usIndices[3]: índices de los vértices.

CVector3 m_vecNormal: normal de la cara.

Es muy parecida a la forma en que todos los ficheros representan una cara, primero por los índices de los vértices y luego la normal de la cara.

El formato 3DS es muy extenso, presenta un gran número de información, por lo cual se hace un análisis general de cómo funciona y qué organización presenta. Para profundizar más en este formato consultar la bibliografía propuesta. [4]

Ventajas

Este formato exporta escenas completas, es decir, un mismo fichero puede contener varias mallas.

Desventajas

No muestra un orden en los bloques de datos y su estructura es compleja y difícil de cargar.

1.3.3 3DX.

El 3DS analizado anteriormente es un formato que optimiza muy bien el tamaño de los ficheros ya que se encuentra en forma binario. Exporta escenas completas, es decir que un mismo fichero puede contener varias mallas, pero infelizmente no muestra un orden en los bloques de datos, aunque esto es recompensado con una cabecera en cada bloque, que presenta un Id para representarlo. La estructura que demuestra es compleja y difícil de cargar. Por tal motivo en la UCI se elaboró un fichero que guarda todos sus datos en sistemas de coordenadas de 3DSMax, y que posee una estructura semejante a los ficheros estructurados en bloques; el mismo lleva el nombre de **3DX**. La herramienta SceneToolKit con la cual se va a laborar para el desarrollo del editor sólo cuenta con la implementación y respaldo de este fichero nuevo, por tanto se decide que este será el candidato selecto para cargar y operar por la aplicación.

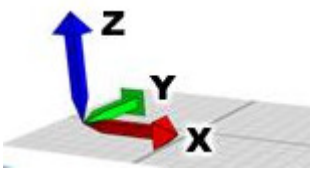


Fig 1. 2: Sistema de Coordenadas 3DSMax.

✓ Estructura de un 3DX.

El fichero 3DX está estructurado de la siguiente forma: Un encabezamiento y luego una serie de **Tags** que identifican qué tipo de objeto está almacenado y su tamaño, el orden de los objetos es arbitrario. Luego del encabezamiento se encuentra el primer **Tag** como un DWORD (4 bytes) y **Size** también DWORD (4 bytes). Este tag identifica los datos que vienen a continuación y ocupan un tamaño de **Size** bytes en el fichero. Esto permite poder recorrer todo el fichero buscando los objetos del tipo deseado de una forma rápida. En el **Anexo 2** se explica de forma más detallada la estructura de cada uno de los bloques que se simbolizan a continuación.

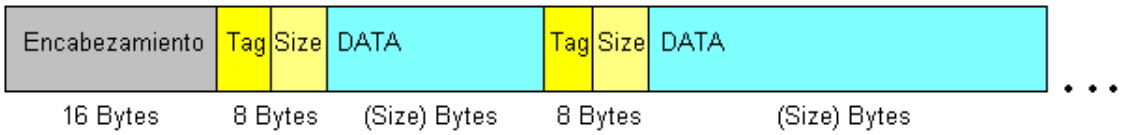


Fig 1. 3: Estructura de un fichero 3DX.

1.3.4 STK.

El formato .STK lleva ese nombre debido a la herramienta para la cual se elaboró SceneToolKit creado en la UCI como parte de la herramienta. El fichero es exportado en **binario** pues los ficheros de este tipo son de menor tamaño que los que se encuentran en texto plano, ejemplo de ello son el **md3** y el **unreal**, que son propios de dos grandes empresas que desarrollan proyectos de realidad virtual con gran impacto en el mundo de los videojuegos y otros productos de los SRV. Además de poseer esta forma (binario), este formato brinda la posibilidad de tener confidencialidad con la información contenida en el fichero 3D. [5]

✓ Estructura del fichero.

A continuación se propone la estructura que define un fichero .STK. Su organización está dada en bloques de datos, en la siguiente figura se muestra claramente la estructura que presenta:

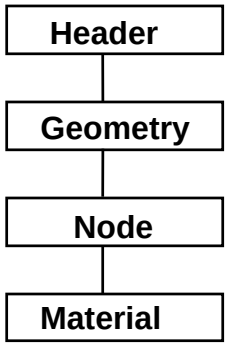


Fig 1. 4: Estructura del fichero STK.

El fichero estará organizado de esta manera, primero el encabezado (Header), después las geometrías (Geometry), los nodos (Node) y finalmente los materiales (Material). La mayoría de los formatos ya existentes usan una organización en forma de bloques, este tipo de estructura permite que a la hora de leer el fichero, se cargue un bloque de datos por completo, el cabezal del disco duro sólo tendrá que moverse en una sola dirección una vez que se esté leyendo un bloque, logrando mayor velocidad de lectura cuando se está cargando el fichero. [5]

Características del bloque Header.

Header es el bloque que representa el encabezado del fichero 3D, brindando informaciones generales tales como la versión del formato, el nombre de la escena, el tamaño del fichero, entre otros. Véase el **Anexo 3** donde se describe la estructura del encabezamiento (**Header**) para un mejor entendimiento.

El encabezado es un contenedor de informaciones generales que describen aspectos básicos del fichero, dando la posibilidad de conocer qué versión de formato es con la que se está trabajando, cuántos elementos de un tipo de bloque están en el fichero, entre otros, que bien pueden incluirse en los espacios reservados que se dejan en la cabecera para un uso futuro.

Características del bloque Geometry.

Geometry contiene toda la información respecto a las geometrías presentes en una escena. A continuación se muestra la figura donde se ven los tipos de geometrías que contiene este bloque y las características que presentan cada una de ellas.

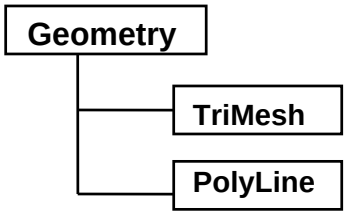


Fig 1. 5: Estructura del bloque Geometry.

Existen dos tipos de geometrías a exportar, una malla (TriMesh) y una polilínea (Polyline).

Los parámetros que contienen el bloque geometría son comunes para los dos tipos de geometría:

- ✓ char g_acName: nombre de la geometría.
- ✓ unsigned int g_iID: Id de la geometría.
- ✓ unsigned int g_iVertexQuantity: cantidad de vértices que contiene la geometría.
- ✓ sVertex g_TVertexList [g_iVertexQuantity]: arreglo de estructuras de vértices dados en los tres ejes *x, y, z*.

Las geometrías se exportan en ese orden, primero todas las mallas (TriMesh) y después todas las polilíneas (Polyline), logrando así mayor comodidad a la hora de cargar el fichero una vez exportado. [5]

El bloque geometría constituye la principal parte de información del fichero, sin el mismo no existiría nada que visualizar una vez exportada la escena, por lo cual es el más importante dentro del fichero. Al tener dentro dos subbloques, brinda mayor comodidad y organización, ordenando primero todas las geometrías tipo malla y después todas las tipo polilínea.

Si se remite al **Anexo 4** encontrará las descripciones de cada subbloque representado anteriormente en el bloque Geometry.

Características del bloque Node.

El bloque Node brinda toda la información respecto a los nodos en una escena. Existen dos tipos de nodos, los nodos geometría y los nodos grupos, en la siguiente figura se muestra de forma clara la estructura de este bloque. [5]

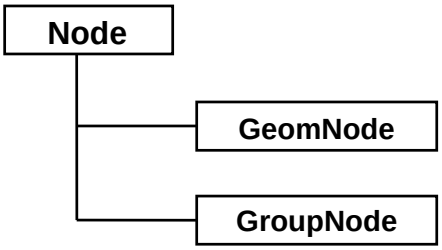


Fig 1. 6: Estructura del bloque Node

El bloque nodo trata a cada objeto en la escena como un todo, analiza sólo las características generales y propiedades desde el punto de vista de cómo se van a representar en la escena, datos tales como: el grado de visibilidad, la matriz de transformación, entre otros. Está divido en dos subbloques, lo cual presenta gran ventaja, a continuación se analizan las mismas en cada subbloque.

- ✓ **GeomNode**: optimiza el tamaño del fichero, ya que en caso de que en la escena existan objetos que sean instancias o copias idénticas de otro objeto, sólo se exporta en el fichero un único objeto y un nodo por cada instancia, los cuales contienen la matriz de transformación del objeto que representa dónde están ubicados en la escena.
- ✓ **GroupNode**: optimiza el dibujado de la escena y el trabajo con la misma una vez que se quiera cargar el fichero.

Uno de los aspectos más importantes es conocer cómo se ubican los nodos de la escena y posteriormente como se sitúan en el árbol de nodos geométricos. Examinar **Anexo 5**.

Características del bloque Material.

El bloque Material contiene todo lo referente a los materiales que se utilizan en la escena y las características de las texturas. A continuación se muestra una figura que representa la estructura de este bloque.

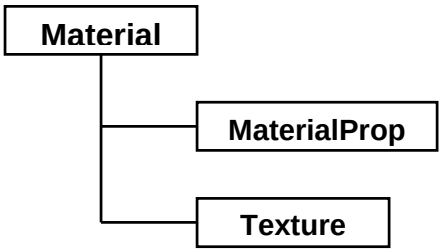


Fig 1. 7: Estructura del bloque Material

El objetivo de este bloque es el de tratar a un material como un objeto con sus respectivas propiedades. Se exportarían todos los materiales que se usan en la escena, en caso de no haber separado este bloque, por cada nodo tendría que dar las características del material que tiene asignado, lo que sería una desventaja ya que en caso de que un nodo use el mismo material que otro, se tendría que repetir la misma información nuevamente. Por otra parte, el tener dividido el bloque material en dos subbloques persigue un objetivo semejante al descrito anteriormente, el primer subbloque contiene las propiedades de los materiales y los índices de las texturas que usan, y el segundo bloque contiene los caminos de las texturas, por lo que se puede tener una cantidad *n* de materiales que utilicen una misma textura sin tener que volver a definirla, sólo haciendo referencia a ella. [5]

Las características de cada uno de los subbloques del bloque material representados en la Fig. 1.7 se exponen en al **Anexo 6** de manera que puede conocerse los atributos correspondientes a cada cual.

1.4 Características físicas de los cuerpos.

El comportamiento físico de los cuerpos en un marco virtual se basa en la aplicación de modelos físicos de la realidad para definir la evolución de los elementos animados en la escena. Todos los objetos se mueven de acuerdo con las leyes físicas: gravedad, elasticidad, fluidez y por tanto reaccionan a estas leyes.

En un entorno se tendrán disímiles modelos para simular su movimiento. Todos estos cuerpos en la literatura de los gráficos por computadoras pueden ser clasificados en dos categorías fundamentales, los *cuerpos deformables* y los *cuerpos rígidos*. No obstante del tipo de objeto, el movimiento de estos generado por la simulación física se basa en la dinámica. Sin embargo la simulación de los cuerpos deformables o cuerpos no rígidos consume mucho más tiempo que la de los cuerpos rígidos. [6]

Se entiende por **sólido rígido** ó **cuerpo rígido** a un conjunto de puntos del espacio que se mueven de tal manera que no se alteran las distancias entre ellos, sea cual sea la fuerza actuante. [7] Los cuerpos rígidos no se deforman, están totalmente rígidos en su estructura y permanecen así aunque otro cuerpo interactúe con él, también se asume que no existe penetración en la interacción aunque si pueden rebotar en respuesta a la colisión con otros cuerpos.

Los cuerpos **no rígidos o deformables**, son aquellos que sufren una deformación (cambio del tamaño o la forma) debido a la aplicación de una o más fuerzas sobre él. [8], es decir que debido a sus

propiedades físicas, su representación pueda ser modificada por la acción de otro cuerpo o agente externo (gravedad, viento, pinza, mano, etc.). Generalmente en los ambientes virtuales, los objetos son representados por mallas triangulares, de manera que una modificación de su forma implicaría una variación en la ubicación de los vértices y aristas de la malla en correspondencia con la acción que provoca dicho cambio.

Posteriormente se hace una breve descripción de las características que identifican a los cuerpos tanto rígidos como deformables en tiempo real, las cuales pueden ser llevabas a la escena a través de la aplicación de algún método definido, o especificadas por el usuario a la hora de la simulación.

Propiedades generales de los cuerpos.

Masa: Es la cantidad de materia que tiene un cuerpo, su unidad fundamental en el Sistema

Internacional de Unidades es el kilogramo (kg) y en el Sistema Inglés es la libra (lb). Para medir masas muy pequeñas, como la del átomo, se emplea la uma (u) que es la unidad atómica de masa.

Volumen: Es el lugar o espacio que ocupa un cuerpo en un medio determinado, lo cual varía en dependencia del tamaño del cuerpo. Para expresar el volumen de un cuerpo se utiliza el metro cúbico (m^3) y demás múltiplos y submúltiplos.

Inercia: Es la resistencia que presenta un cuerpo a cambiar su estado de reposo o de movimiento, mientras no exista una fuerza que lo modifique.

Impenetrabilidad: Es la propiedad que tienen los cuerpos de no poder ocupar el mismo lugar o espacio al mismo tiempo.

Divisibilidad: Propiedad en virtud de la cual los cuerpos sólidos pueden fraccionarse en partículas muy pequeñas hasta el límite molecular en algunos casos. [9]

Propiedades especificas de los cuerpos.

Dentro de las propiedades específicas de los cuerpos se encuentran las *propiedades físicas*: dureza, tenacidad, maleabilidad, ductibilidad, punto de fusión, punto de ebullición, densidad, elasticidad, plasticidad, resistencia mecánica, entre otras.

Dureza: es la resistencia de los cuerpos a ser rayados.

Tenacidad: es la resistencia de la materia a ser fraccionada por tensión.

Maleabilidad: es la capacidad que tienen los metales para formar láminas.

Ductibilidad: es la propiedad de los metales para formar alambres o hilos muy delgados.

Elasticidad: Designa la propiedad mecánica de los cuerpos de sufrir deformaciones reversibles cuando se encuentran sujetos a la acción de fuerzas exteriores y de recuperar la forma original si estas fuerzas exteriores se eliminan.

Plasticidad: Propiedad mecánica de un material, biológico o de otro tipo, de deformarse permanentemente e irreversiblemente cuando se encuentra sometido a tensiones por encima de su rango elástico.

Densidad: Es la cantidad de sustancia contenida en una unidad de volumen determinado, es una unidad derivada. Se obtiene al dividir la cantidad de su masa entre el volumen que ocupa. La unidad en el sistema internacional es kg/m^3, utilizándose más en la práctica las siguientes unidades g/cm^3 ó kg/dim^3.

Punto de ebullición: es la temperatura a la que hierve un líquido y pasa al estado de gas o vapor.

Punto de fusión: es la temperatura en la que un cuerpo sólido pasa al estado líquido.

Resistencia mecánica: Capacidad que tiene un material de soportar los distintos tipos de esfuerzo que existen sin deformarse permanentemente. [10]

1.4.1 Comportamiento físico de cuerpos rígidos.

Existen diversas formas de lograr que un cuerpo se comporte con un alto grado de realismo en un medio totalmente simulado, y cuyas propiedades sean configuradas o manejadas por un ordenador. La idea de delegar un modelo físico, utilizado por los juegos o simuladores en una biblioteca es algo que en los últimos tiempos se ha vuelto muy común. Son numerosos los sistemas informáticos modernos y juegos de video que utilizan “motores físicos” para producir animaciones realistas del movimiento de los objetos y personajes, que no son más que conjuntos rígidos dentro del marco del diseño para la simulación. Entre los motores físicos más conocidos se pueden mencionar: NovodeX, Newton, Tokamak, ODE, PhysX, Endorphin y Ragdoll. [11]. Adelante se explican algunas de sus características generales.

NovodeX Physics SDK: es una librería muy completa de simulación de entornos físicos en tiempo real. Provee una demostración llamada *Rocket* que es muy completa e ilustrativa así como un SDK provisto de gran cantidad de ejemplos y buena documentación. Recientemente la biblioteca cambió de

tipo de licencia permitiendo ser utilizada libremente para fines no comerciales. Es soportado en Windows, Linux, y consolas. [11]

Una característica a destacar de este motor es la posibilidad de especificar una clase para la asignación de memoria (*allocator*); esta posibilidad es bienvenida particularmente en el desarrollo para consolas debido a que usualmente se suele utilizar un asignador propio que gestiona toda la memoria y no los operadores clásicos *new/delete*. Otra característica interesante es la posibilidad de especificar una clase para la depuración visual. En pocas palabras es una clase que especifica métodos para el dibujo de ciertas primitivas que luego utiliza el motor de física para mostrar distintas fuerzas, *bounding boxes*, etc. Esto facilitaría mucho la depuración de un juego.

Newton Game Dynamics: es una solución integrada para la simulación de entornos físicos en tiempo real. [11] Es libre, pero de código cerrado (*closed source)* para la simulación realista de cuerpos rígidos en juegos y otras aplicaciones en tiempo real. La licencia de *Newton Game Dynamics SDK* permite a los desarrolladores incorporar libremente el motor dentro de su proyecto personal o productos comerciales siempre que sea dado en los créditos y sea distribuido solamente como parte del programa de software compilado, que no es en sí mismo el motor físico. [12]

El API provee gestión de escenas, detección de colisiones, comportamiento dinámico de objetos. Es pequeña, estable y fácil de utilizar. [11] Es soportado en Windows.

Tokamak Game Physics SDK. *Tokamak* [11] es una biblioteca física de alto rendimiento diseñada específicamente para juegos con una interfaz de programación muy sencilla (a costa de pérdida de flexibilidad). Como todas las bibliotecas, basan sus modelos en cuerpos rígidos que especifican restricciones y soporta detección de colisiones. El asunto es que las primitivas de colisión son bastante pocas y su implementación no facilita la creación de particiones de espacio (para implementación de *quadtrees*, *octrees*, etc.). La interfaz general de la biblioteca está escrita en C++. Es soportado por Windows. Es bastante limitada en funcionalidades, sin embargo esto no impide implementar las funcionalidades más deseadas de una biblioteca física (colisiones, automóviles y *ragdolls*). A diferencia de Novodex, que hace un uso intensivo de métodos virtuales, la biblioteca Tokamak los evita como decisión de diseño para priorizar el rendimiento. En su lugar, cuando la biblioteca debe realizar llamados al motor o al juego se utilizan funciones callback tipo C.

Open Dynamics Engine (**ODE**) es una biblioteca de código abierto, excelente para la simulación de la dinámica de cuerpos rígidos articulados pues permite dotar a los objetos de masa y solidez física, aplicarles fuerzas y hacer que todos se comporten de una forma más realista. [11]

Por ejemplo, es buena para la simulación en vehículos, criaturas con articulaciones, y el movimiento de objetos en entornos de Realidad Virtual. Es rápido, flexible, y robusto, incorpora la detección de colisiones. Es además bastante estable, se encuentra escrita en C++, posee una interfaz en C que facilita la integración. Es posible descargar binarios para Windows con precisión simple (uso de floats) y precisión doble (uso de doubles). Es soportado en Windows, Linux, MAC OS X. Se distribuyen las fuentes de la biblioteca que están escritos en ANSI C++.

Esta biblioteca agrupa familia de funciones por prefijo, lo cual facilita la búsqueda de funcionalidades comunes a este tipo de bibliotecas. Usualmente cuando se crea algún tipo de entidad en ODE esta retorna un identificador que es un número entero, luego para modificar algunas de sus propiedades se debe utilizar una función y pasarle dicho identificador como primer parámetro. Se pueden poseer cuantos mundos se quieran, sin embargo no es necesario crear distintos mundos para separar dominio de colisiones ya que ODE soporta múltiples espacios por mundos.

En conclusión, ODE se encarga de asociar, a cada objeto que se le indique una entidad física, que normalmente coincidirá con su forma visible lo más posible, y que será la responsable del comportamiento físico del objeto. A partir de ese momento, la que mande será la entidad física, independientemente de la forma visible del objeto.

1.4.2 Comportamiento físico de cuerpos deformables.

El trabajo con las deformaciones de los cuerpos ha tomado mayor auge en los últimos tiempos debido al proceso de simulación de sistemas reales a través de entornos virtuales, con el fin de obtener métodos, técnicas o productos que contengan un alto nivel de realismo.

Teniendo en cuenta la ecuación de constitución de los sólidos deformables, estos pueden clasificarse en:

- ✓ **Comportamiento elástico**: El sólido se deforma adquiriendo mayor energía potencial elástica y consecuentemente, aumenta su energía interna sin que se produzcan transformaciones termodinámicas irreversibles. El valor máximo de la fuerza aplicada en la deformación elástica es nombrado límite elástico, y a partir de ese valor el proceso es irreversible. Dentro de este comportamiento encontramos varios subtipos: elástico lineal (isótropo, no-isótropo) y elástico no-lineal.

- ✓ **Comportamiento plástico**: Se distingue por ser irreversible y puede presentarse de distintas formas:
    - Plástico puro.
    - Plástico con endurecimiento.
    - Plástico con ablandamiento.
- ✓ **Comportamiento viscoso:** Se produce cuando la velocidad de deformación entra en la ecuación constitutiva. Pueden distinguir varios modelos: **visco-elástico** y **visco-plástico**.
    - Visco-elástico: Relaciona las propiedades elásticas y viscosas del material.
    - Visco-plástico: Se refiere a materiales que comparten propiedades tanto plásticas como elásticas.

Es importante 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. [12]

Deformaciones Elásticas

Las deformaciones elásticas o reversibles, teniendo en cuenta las características del tejido humano, resultan de gran interés, debido a que estas hacen que el cuerpo trate de recuperar su estado original contrarrestando la fuerza que le provoca la distorsión una vez retirada la misma.

Generalmente los cuerpos no rígidos se definen por su estado inicial o de equilibrio, además de tener en cuenta una serie de parámetros que determinan como este se comportará ante la acción de una determinada fuerza.

1.4.3 Métodos de deformación basados en física.

Durante años la modelación física y animación de objetos deformables ha sido un problema de gran interés para la sociedad de gráficos por ordenadores. Desde sus inicios con 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. [13]

Los modelos deformables son una clase de primitivas para modelado de curvas, superficies y sólidos generales que obedecen a la dinámica de cuerpos no rígidos. Estos modelos físicos idealizan amplias gamas de respuestas de materiales bajo diversas condiciones ambientales, son capaces de demostrar una gran variedad de comportamientos naturales incluyendo elasticidad, visco-elasticidad, plasticidad, fractura, comportamiento, tipo fluido, entre otras.

Los métodos basados en física constituyen una exitosa técnica para representar objetos blandos tales como gomas, ropas, tejidos, etc. a través de la incorporación de propiedades físicas al material que se modela, por lo que brindan una solución muy apropiada para los entornos de simulaciones virtuales. Entre estos métodos, 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 contínua.

1.4.4 Sistema Masa Resorte.

Un 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.” [14]

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.

Se pueden encontrar muchas implementaciones 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 **Fij= -kij*Δij*uij**

Donde:

Δij: es la diferencia entre la longitud del resorte en reposo y la longitud en un instante de tiempo dado.

kij: la constante elástica del resorte.

uij: es el vector unitario entre Ni y Nj.

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. [14]

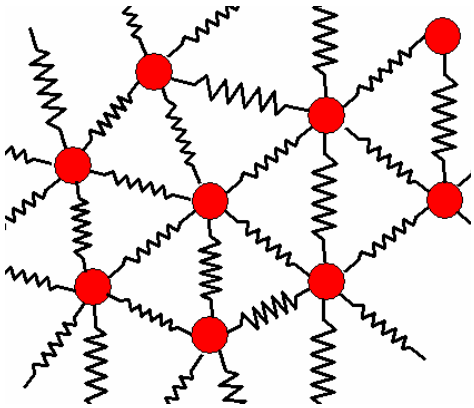


Fig 1. 8: Sistema Masa-Resorte

Las propiedades físicas que presenta este método de deformación son las siguientes:

- ✓ m: masa de las partículas que conforman el modelo.
- ✓ k: constante elástica de los muelles que unen las partículas o centros de masa en el modelo.
- ✓ d: constante de Dumpin.

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 los métodos continuos que serán descritos posteriormente. Sin embargo, no es todo lo exacta que necesitamos porque no se soporta en la elasticidad continua, además depende en gran medida de la topología y la resolución de la malla. Debido a la versatilidad y efectividad de esta técnica, es usada en un amplio espectro de aplicaciones. Los modelos masa-resorte están entre las técnicas más usadas para la modelación de tejidos. Aun sin obviar que presenta algunos inconvenientes, su velocidad, unido a su capacidad de modelar los atributos físicos del material la convierten en una buena opción para la simulación de deformaciones. [14]

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).

1.4.5 Método de Elementos Finitos (FEM).

El Método de Elementos Finitos (MEF en castellano o FEM en inglés) es uno de los métodos más populares en las ciencias de la computación para resolver ecuaciones diferenciales en rejillas irregulares. Comenzó a utilizarse hace más de 30 años; en un principio se utilizaba solamente para el análisis estructural, los modelos eran muy simples y las condiciones de carga que se podían considerar eran limitadas. Posteriormente, y debido al avance por una parte de los conocimientos, y por otra de la capacidad de las computadoras, el método comenzó a aplicarse en otros problemas de la física, citando por ejemplo, transferencia de calor, dinámica de fluidos, magnetismo, acústica, etc. Además se incrementó notablemente el "realismo" que se podía lograr en las simulaciones.

Este método permite convertir un problema continuo en un problema discreto y por tanto con un número finito de grados de libertad, es decir define los elementos 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 que puede ser resuelto a través de un análisis numérico. En varios 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. [15]

Teniendo en cuenta lo planteado anteriormente, pueden definirse un conjunto de pasos para conformar el FEM:

- ✓ **Discretización del continuo**: 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, como se muestra a continuación.

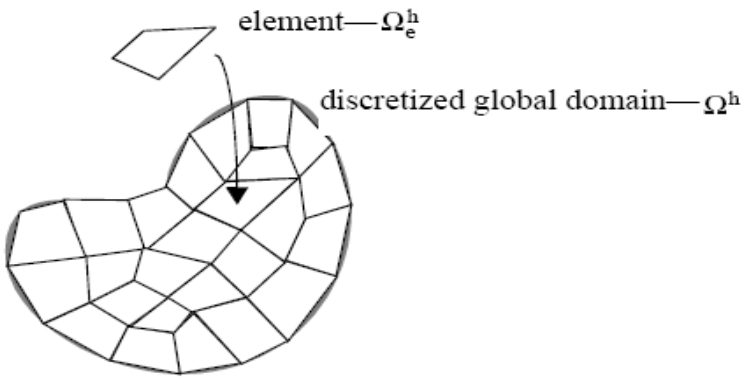


Fig 1. 9: Discretización de un dominio global.

- ✓ **Interpolación:** Se usan funciones de forma (también conocidas como funciones base) para interpolar cualquier punto P del continuo. Por tanto, cualquier función continua aplicada a un cuerpo continuo es aproximada por un sistema de ecuaciones finito en término de coordenadas de los nodos.
- ✓ **Aplicación de la Mecánica Continua:** Una vez discretizado el cuerpo y las funciones de forma se deben encontrar expresiones que empleen las ecuaciones de elasticidad en términos de los nodos de la malla, esto permitirá calcular la distribución interna de tensiones del elemento. En dependencia de las necesidades del sistema su formulación puede ser estática o dinámica.
  - *Análisis Estático*: Permite determinar los componentes de los nodos por efecto de una solicitud estática y, en una segunda fase, se determina el estado en ciertos puntos característicos de cada elemento. Acota la deformación del componente de estudio y localiza zonas altamente solicitadas o zonas de solicitación baja.
  - *Análisis Dinámico:* 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:

$$Mx''+Dx'+Kx = F \tag{1}$$

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. [16]

Las propiedades físicas que presenta este método de deformación son las siguientes:

- ✓ m: masa de los elementos simples discretizados.
- ✓ k: constante elástica.
- ✓ Y: módulo de Young.
- ✓ v: coeficiente de Poisson.

Este método requiere de un modelo geométrico del sistema continuo. Para ello se pueden emplear diferentes tipos de elementos:

- ✓ Elementos unidimensionales.
- ✓ Elementos planos.
- ✓ Elementos tridimensionales.
- ✓ Elementos tipo placa.
- ✓ Elementos de longitud infinita.
- ✓ Elementos tipo contacto.

La correcta selección del tipo de elemento geométrico es fundamental para una buena representación del sistema. Un mismo elemento puede tener diferentes mallados:

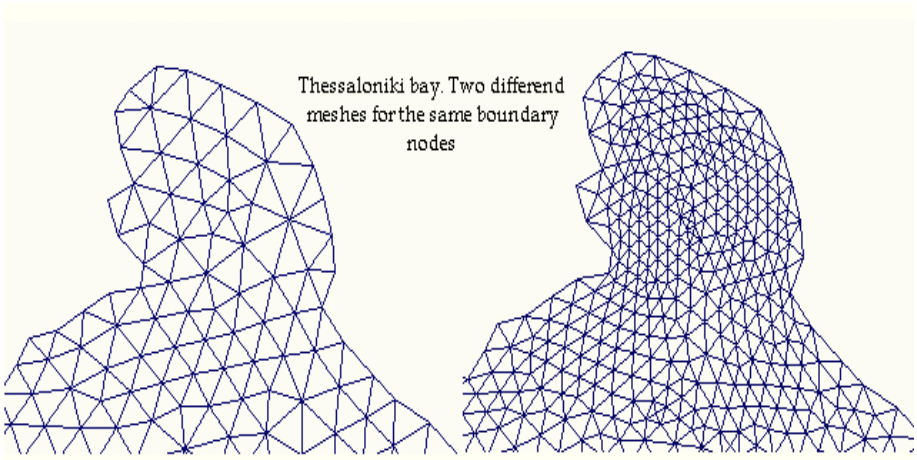


Fig 1. 10: Representación de un sistema por dos mallados diferentes haciendo uso de FEM.

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 [19]. 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. FEM fue usado para modelar deformación de tejidos para un simulador de endoscopia ginecológica por Székely y su equipo en 1999 [20].

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. [21]

Finalmente, la utilización del método de elementos finitos se presenta como una valiosa herramienta, tanto para el diseño, como para evaluar equipos o componentes en servicio. El mismo permite no solo representar geometrías complicadas, sino también considerar las más variadas condiciones de carga, así como propiedades mecánicas del material: comportamiento elástico, plástico, hiperelástico, materiales compuestos, etc.

1.4.6 Método de Elementos Frontera (BEM).

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 1.11. 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, los cambios de topología son más complejos de manipular que en FEM Explícito. [13]

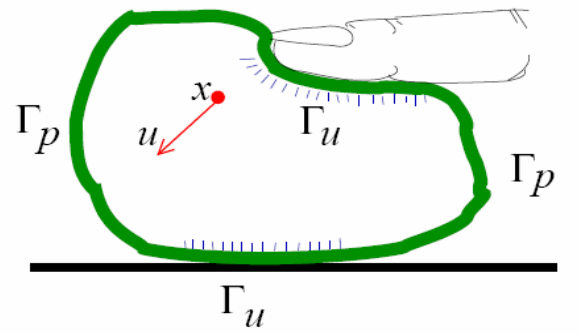


Fig 1. 11: Notación de BEM y sus condiciones de frontera.

1.5 Principales características de algunos editores gráficos.

Un editor es una de las herramientas de mayor uso hoy en día, pues facilitan la navegación entre las distintas categorías de los tipos de objetos, desde la raíz, la figura, controles, ejes y sus respectivas características. Ya sea para adquirir o establecer alguna propiedad de cualquiera de los aspectos que caracterizan los cuerpos en los cuales se está trabajando. Existe una gran diversidad de editores: Editores de Texto, Editores HTML, Editores Gráficos, entre otros. De estos últimos, los editores de escenas 3D juegan un papel importante, la inmensa mayoría de ellos presentan características similares, a las cuales se refieren a continuación.

✓ Objetos Básicos:

- Puntos, rectas, rayos, planos, esferas.
- Paralelogramos, segmentos, circunferencias, polígonos regulares.
- Vectores (afín, por escalar, entre dos puntos, unitario).

✓ Objetos derivados:

- Operatoria entre vectores (suma, resta, producto punto y cruz).
- Operatoria entre vectores y otros objetos (normal, directriz, reflejado).
- Medición de ángulo entre tres puntos.
- Punto sobre superficie y punto sobre curva.

✓ Otras características relevantes:

- Creación de vistas predefinidas.

- Etiquetas visuales para los objetos.
- Definir objetos como no movibles por el usuario.
- Modificar colores, visibilidad y transparencia, tanto para los objetos como para sus etiquetas.
- Importa modelos tridimensionales WaveFront.
- Activar, para interacciones, propiedades de los objetos.

✓ Limitaciones:

- No permite sistemas de coordenadas relativas.
- No incorpora funciones.
- Máximo de 300 objetos e Iconografía no definida.

1.5.1 Antecedentes y funcionamiento de algunos editores de propiedades físicas.

El desarrollo tecnológico en la actualidad ha permitido crear interfaces "inteligentes", que pueden "recordar" o guardar en la memoria algunas preferencias del usuario, a partir de lo cual la interfaz de productos interactivos pueden adaptarse con el fin de establecer una conexión cognitiva y emocional con el usuario. Por otro lado, tanto la costumbre y cotidianidad de los ambientes tecnológicos como los diseños más recientes parecen ir tomando una apariencia de menor artificiosidad y por tanto, adquieren una mayor "naturalidad".

El mayor grado de interactividad se encuentra representado por la realidad virtual; estos entornos virtuales se presentan a través de interfaces inteligentes, lo cual permite al usuario experimentar un alto grado de interactividad al ofrecerle la libertad de modificar y/o generar contenidos, sean estos, elementos audiovisuales, ambientes, situaciones o personajes.

Con el nacimiento de la RV como una interfaz de computadora que utilizaba la percepción humana para facilitar la comunicación hombre-máquina, y su progenitor Ivan Shutherland, se concibió en 1961 por el mismo Shutherland el primer aparato de visualización de tubos catódicos que representaba imágenes producidas por computadoras (imágenes digitales) al cual denominó ”generador de escenas”.

Los procesos de simulación en entornos virtuales generalmente cuentan con dos componentes claramente diferenciados: *un generador de escenas o editor de propiedades* como también podría

llamársele y el simulador real quien hará visibles los cambios en el medio que se desee simular en un momento dado. Si bien el primero de ellos permite generar escenas que se desean en la práctica, el segundo permite obtener las propiedades que contiene la escena y reflejar su dinámica.

Para la concepción de estas escenas virtuales, el *generador de la escena* toma como insumo elementos puramente geométricos (sin propiedades físicas asociadas), o sea la estructura 3D de los cuerpos con las cuales se realizará la operación. El único objetivo de las entidades meramente geométricas 3D es crear un entorno inmersivo para el cliente, pues son geometrías (mallas), en las que ningún tipo de interacción se puede llevar a cabo, y en caso de ser posible algún procedimiento que sea similar encierra gran complejidad. Los cuerpos 3D con propiedades dinámicas (que responden a algún modelo (deformable o rígido), a su vez, son los que serán manipulados por el encargado de realizar la edición durante la fase de formación del nuevo modelo para la simulación. Ellos responden en tiempo real a la interacción con su superficie, y la deformación o conservación de su estado de manera similar a como sucedería en un ambiente real, en las mismas condiciones. En un generador de escenas pueden coexistir dos o más tipos de modelos deformables, lo que haría variable las posibles soluciones de edición. Se conoce que los más usados para deformaciones superficiales se sustentan sobre la base de un Modelo de Deformación Masa-Resorte (Mass Spring Deformable, en inglés), y un segundo modelo que se refiere a la deformación volumétrica sustentado sobre la base del Método de Elemento Frontera (BEM, siglas en inglés). La selección del modelo a usar para un determinado cuerpo está en dependencia de las características físicas y de la precisión que se requiera.

En la actualidad existen varias herramientas destinadas a la edición de modelos respaldados por parámetros físicos-matemáticos. Véase lo siguiente.

- ✓ *Asagrad (Advanced Spectral Scene Generator)*: Proporciona imágenes con un contenido físico avanzado, de gran aplicación en sistemas de estudios de sensores tácticos. Proporciona alta fidelidad de simulación de escenarios dinámicos y cuenta con espectros altamente optimizados, se fundamenta sobre la física basada en la fenomenología de modelado de firmas espectrales. Cuenta con una interfaz gráfica fácil de usar, que puede cargar en su base de datos las estructuras 3D del terreno, especificar la cantidad de sensores a usar, condiciones ambientales, vehículos y modelos humanos en la escena.
- ✓ *Carrara Studio 2*: Las cualidades de este software son varias, no intenta ser un software para usuarios muy avanzados aunque se permite el lujo de incluir características que otro software todavía hoy no presentan como es el comportamiento físico de objetos sólidos, dígase velocidad

inicial y/o angular (si la tiene), introducir una fuerza (ya predeterminadas), dotar a cada objeto físico de cualidades (fricción, dureza, capacidad de rebote, entre otras). Lo anterior se complementa con la selección de los objetos que seguirán el comportamiento que se define a través de las propiedades, y luego solo un clic para lograr la incorporación de la dinámica deseada. [22]

En el **Anexo 7** se referencian algunos ejemplos de interfaces de editores gráficos existentes en el mundo.

2

Capítulo 2 : Soluciones Técnicas

Introducción

En el presente capítulo se proponen las soluciones técnicas para el desarrollo del Editor de Propiedades Físicas, partiendo de la fundamentación teórica que se ha planteado anteriormente. Se define el fichero a cargar por la herramienta que se desarrollará, y se concreta también la estructura que va a tener el nuevo fichero que se va a crear, en el cual se editarán las propiedades físicas que se desea que tenga el objeto cargado, en correspondencia con el modelo físico-matemático a utilizar, así como la forma de exportarlo. Es importante precisar que toda esta labor estará acoplada a la herramienta SceneToolKit.

2.1 Consideraciones Específicas.

Diseñar e implementar un generador de escenas que regule determinadas cualidades físicas requiere de varias exigencias. ¿Por qué seleccionar la SceneToolKit como soporte para el desarrollo del Editor de Propiedades Físicas?

SceneToolKit es la primera herramienta brindada por el Proyecto de Herramientas de Desarrollo para Sistemas de Realidad Virtual a la UCI para los proyectos de este perfil, surge como respuesta a la necesidad de mejorar los Sistema de Realidad Virtual (SRV) que están actualmente en desarrollo, pues para hacer un simulador, juego o cualquier sistema de este tipo se necesita tener motores gráficos que proporcionen mayor rapidez y eficiencia en su construcción; existen muchos engine pero la documentación es escasa o trabajan sustentados en otros principios, o sencillamente hay que pagar licencias que no están al alcance.

La SceneToolKit agrupa las funcionalidades comunes de cualquier SRV, de manera que se les facilite el trabajo a los programadores de juegos y simuladores a través de la reutilización de código y el uso de sus componentes. Esta herramienta, en perfeccionamiento actualmente, permite manipular geometrías, polilíneas, mallas triangulares, luces, cámaras, personajes, componentes visuales (botones, menúes, edits, paneles, etiquetas, entre otros); permite no sólo la visualización de los entornos sintéticos sino además la aplicación de leyes físicas y matemáticas, animaciones, usando las librerías gráficas OpenGL y DirectX, sobre plataforma Windows y Linux. A la vez, se retroalimenta con las necesidades de los programadores que la utilicen, así como de sus investigaciones.

Mientras numerosos editores en el mundo trabajan sus estructuras a nivel de puntos, rectas, rayos, planos, esferas, segmentos, circunferencias, polígonos regulares, el Editor de Propiedades Físicas implementado sobre la SceneToolKit tendrá la posibilidad de trabajar con geometrías, componentes visuales, modelos físicos, visualizar objetos y entornos, asi como utilizar las funcionalidades que esta herramienta provee para cargar y salvar ficheros, trabajar con su estructura interna y probar las soluciones obtenidas de forma dinámica.

2.2 Propuesta Física.

Determinar el comportamiento de un cuerpo rígido o deformable de acuerdo a la dinámica que lo caracteriza resulta bastante complejo cuando se requiere de un alto grado de exactitud y veracidad. En el capítulo 1 se exponen varios motores físicos que ayudan al tratamiento de cuerpos duros así como diferentes procedimientos que determinan el comportamiento de los sistemas deformables. Resulta

interesante luego de ese estudio la selección de qué elementos serán utilizados en el desarrollo del Editor de Propiedades Físicas.

2.2.1 Cuerpos Rígidos.

Actualmente, para soportar la dinámica de los cuerpos duros (rígidos) se han creado lo que en secciones anteriores se define como motores físicos. Con el fin de incorporar aspectos de esta índole resultaría provechoso hacer uso de uno de estos motores que reduciría el coste de recursos por parte del trabajo manual. La SceneToolKit recientemente acopló un módulo que vincula su funcionamiento con un motor físico de los ya mencionados en la fundamentación teórica, **ODE**, que actualiza el estado de los cuerpos rígidos a través del tiempo, atiende el proceso de acción y respuesta a las colisiones que pueden ocurrir entre ellos, facilita hacer visible la edición de los objetos de forma más óptima, con mayor velocidad, relaciona varios parámetros (posición inicial, velocidad (angular, lineal), masa, fuerzas, coeficientes de fricción, un ambiente o mundo de la escena que también cuenta con sus especificidades (gravedad, fricción, resistencia al medio, entre otras), así como permite asociar volúmenes de geometrías (cúbicas, cilíndricas, esféricas) a los objetos cargados teniendo con anterioridad las dimensiones del mismo.

Por tal razón se utiliza este motor físico ya integrado a la SceneToolKit para la simulación de cuerpos rígidos en el desarrollo de la aplicación que se propone.

Partiendo de todo lo anterior se toman los datos posibles a editar en esta iteración:

- ✓ Masa (m).
- ✓ Velocidad (lineal, angular) ambas representadas a través de un vector tridimensional (x, y, z).
- ✓ Posición o ubicación inicial del cuerpo en la escena (coordenadas x, y, z).

2.2.2 Cuerpos Deformables.

Definitivamente, entre todos los métodos de deformación basados en física estudiados en el capítulo anterior 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. [23] 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 aseguran que, para deformaciones pequeñas, los modelos de muelles se comportan similar a un modelo elástico de elementos finitos como probara Keeve et al [24].

Igualmente la herramienta SceneToolKit cuenta con un módulo de Deformación de Objetos para Sistemas de Realidad Virtual que aún no está acoplado en su totalidad, que controla el proceder de los objetos no rígidos y sustentado bajo el método de deformación Masa-Resorte, pues en la SceneToolKit 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.

En este módulo 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:

$$F_{elast} = \left[k_s\left(\left|x_j - x_i\right| - l\right) + k_d\left(v_j - v_i\right)\right]\frac{x_j - x_i}{\left|x_j - x_i\right|} \quad (2)$$

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 [25] y v la velocidad de los centros de masa.

Atendiendo a la segunda ley de Newton se define $$x''(t + \Delta t) = \frac{\sum F(t + \Delta t)}{m} \quad (3)$$

Además se examina cada partícula y cómo se afecta por determinada fuerza, en dependencia de la cantidad de muelles que la relacionen con otras partículas, y otras fuerzas como la fuerza de gravedad, o fuerzas que se apliquen en el momento de la simulación. Conjuntamente se tiene en cuenta la conservación del volumen del cuerpo en la escena.

Atendiendo las características anteriores podrían hacerse editables las siguientes particularidades o propiedades:

- ✓ Masa. (m)
- ✓ Constante Elástica. (k)
- ✓ Coeficiente de Amortiguación. (K_d) [25]

Luego, debido a que este último modelo no está integrado completamente a la SceneToolKit, en esta primera iteración solo se abordará lo relacionado con cuerpos no deformables, proponiendo el diseño para ambos casos pero la implementación no permitirá la prueba del fichero generado para entidades blandas, a diferencia de los rígidos.

2.3 Fichero a cargar.

Cargar un objeto es uno de los aspectos fundamentales que debe realizar el Editor de Propiedades Físicas, sin mencionar el resto de las operaciones que encierran la edición de una estructura desde el punto de vista físico-matemático. Antes de seleccionar el fichero que se cargará resultaría sustancial analizar las definiciones de cada archivo que se citaron anteriormente en la fundamentación teórica, y luego de destacar sus ventajas y desventajas obtener como resultado una propuesta de fichero a cargar.

Los archivos .STL son usados rara vez por los programas de modelado y animación 3D, y habitualmente se hace necesario una exportación de los modelos de éste formato antes de comenzar la “impresión” 3D. Dadas las diferencias entre el formato .STL y los formatos 3D de origen, las conversiones suelen dejar fallos estructurales en el modelo. Por lo tanto, los archivos .STL deben ser comprobados usando un software especial antes de utilizarse para fabricar un modelo, por tanto ese formato no es el más idóneo para usar en este trabajo. Solo faltaría detenerse en el fichero 3DX, que como bien se expresa en la fundamentación teórica propone una estructura bastante sencilla, fácil de manejar a la hora de tratar con sus componentes, con un formato propio, ajustado a las necesidades de la SceneToolKit, por lo cual se decide escoger este como propuesta a cargar y operar en el editor.

2.4 Forma de Salvado.

Como resultado se obtendrán dos ficheros, el que inicialmente se carga, sin modificación alguna, y se define uno nuevo con todas las exigencias desde el punto de vista físico que se le han agregado al

objeto inicial, en correspondencia con el proceder que se quiere lograr en este y el modelo físico seleccionado.

El nuevo fichero se exportará en binario debido a que los formatos de ficheros de este tipo son de menor tamaño que los que están en forma de texto plano que son propios de las grandes empresas que desarrollan proyectos de RV con gran impacto en el mundo de los videojuegos. Además el poseer esta forma de salvado brinda la posibilidad de tener mayor confidencialidad en la información contenida en el fichero.

2.5 Estructura y nombre del fichero físico.

El formato del nuevo fichero que se obtiene como producto de los cambios físicos aplicados tendrá por extensión **.phs**, debido a que este será el que almacenará todas las propiedades físicas (**Ph**ysic**s**, en inglés) y matemáticas como producto de la edición de la escena. La organización está dada en bloques de datos, en la siguiente figura se muestra de forma clara y precisa la estructura que presenta.

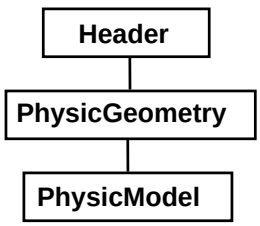


Fig 2. 1: Estructura del fichero físico .phs

El fichero estará organizado de la siguiente manera: primero el bloque encabezado (Header), después la geometría física (PhysicGeometry), y finalmente el del modelo físico (PhysicModel). Básicamente se reutilizó el mismo formato que se propone en el fichero .STK, uno de los tipos de ficheros analizados en la fundamentación teórica, que también propone una estructura en bloques, pues esta reporta grandes ventajas ya que permite que a la hora de leer el fichero, se cargue un bloque de datos por completo, el cabezal del disco duro sólo tendrá que moverse en una sola dirección una vez que se está leyendo un bloque, logrando mayor velocidad de lectura cuando se está cargando el fichero.

2.5.1 Características del bloque Header.

Header es el bloque que representa el encabezado del fichero .phs, brindando informaciones generales. A continuación se muestra la estructura del encabezamiento (**Header**) para una mejor comprensión.

- ✓ float fversion: contiene la versión del fichero que se está trabajando.
- ✓ EShapeVolumeTypes eVolumeGeometry: tipo de geometría que se utiliza.
- ✓ unsigned int uiReservedSpace: reserva espacios para posibles extensiones a otros modelos que no se implementen en este momento.

El encabezado de un fichero es de gran importancia pues este es un contenedor de informaciones generales que describen aspectos básicos del fichero, dando la posibilidad de conocer qué versión de formato es con la que se está trabajando, cuántos elementos de un tipo de bloque están en el fichero, entre otros, en dependencia del fichero con el que se esté trabajando, que bien pueden incluirse en los espacios reservados que se dejan en la cabecera para un uso futuro.

2.5.2 Características del bloque PhysicGeometry.

El bloque GeometryPhysics contiene toda la información respecto a las geometrías físicas presentes en una escena. A continuación se describen los parámetros que contienen este bloque en dependencia de la geometría a utilizar.

- ✓ Geometría Cúbica (SGeometryBox)
  - unsigned float fwidth: ancho del cubo.
  - unsigned float flenght: largo del cubo.
  - unsigned float fhigh: altura del cubo.
  - EPhysicModel m_EPhysicModel: tipo de modelo físico que se va a utilizar.
- ✓ Geometría Esférica (SGeometrySphere)
  - unsigned float fradio; radio de la esfera.
  - EPhysicModel m_EPhysicModel: tipo de modelo físico que se va a utilizar.

- ✓ Geometría Cilíndrica (SGeometryCylinder)
  - unsigned float fradio: radio del cilindro.
  - unsigned float flengh: longitud del cilindro.
  - EPhysicModel m_EPhysicModel: tipo de modelo físico que se va a utilizar.

2.5.3 Características del bloque PhysicModel.

Este bloque contiene los parámetros físicos que se quieren añadir al comportamiento del objeto que representa el fichero inicial en correspondencia con el modelo al que pertenece. Posee dos estructuras, una para guardar los indicadores para los cuerpos blandos y otra para los cuerpos rígidos.

Estructura para los Cuerpos Blandos:

- ✓ unsigned float fmasa: valor de la masa que va a tener el cuerpo.
- ✓ unsigned float fCteElast: constante elástica que tiene el modelo deformable masa-muelle.
- ✓ unsigned float fAmortig: valor de la amortiguación que tiene el modelo deformable masa-muelle.

Estructura para los Cuerpos Rígidos:

- ✓ unsigned float fmasa: valor de la masa que va a tener el cuerpo.
- ✓ CVector3 kVelocL: velocidad lineal del cuerpo (x, y, z).
- ✓ CVector3 kVelocA: velocidad angular del cuerpo (x, y, z).
- ✓ CVector3 kPossition (x, y, z): vector posición del cuerpo que determina la posición inicial del mismo en la escena.

2.6 Consideraciones técnicas generales.

2.6.1 Metodología.

Para guiar el proceso de desarrollo del sistema se utilizará el proceso unificado de desarrollo de software (RUP). Esta metodología además de ser un proceso de desarrollo de software bastante complejo, ofrece numerosas posibilidades de adaptación a sistemas de software sin tomar en cuenta el área a la que pertenece, organización, niveles de aptitud o dimensiones del proyecto. Otros aspectos que la hacen distintiva dentro del resto de las metodologías y la convierten en una más robusta es que:

es dirigida por casos de uso lo cual facilita que se siga el curso de los eventos a través de una serie de flujos de trabajos que tienen como inicio los casos de uso, los cuales van dirigiendo todo el desarrollo del sistema con su maduración; centrada en la arquitectura pues describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente; e iterativo e incremental lo que supone que cada proyecto o producto pueda dividirse y subdividirse en partes y que cada una de ellas se desarrolle de manera iterativa encerrando actividades de todos los flujos de trabajo, evolucionando de forma incremental a medida que se van uniendo los resultados para obtener el producto final, todo de forma planificada.

2.6.2 Enterprise Architect

La herramienta Enterprise Architect (EA) combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación. EA puede equipar a su equipo entero, incluyendo analistas, evaluadores, administradores de proyectos, personal del control de calidad, así como también verifica el rango completo de las herramientas y características case en detalle.

Enterprise Architect es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. El manual de usuario está disponible en línea. Soporta generación e ingeniería inversa de código fuente para muchos lenguajes populares, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP.

El Lenguaje Unificado de Modelado provee beneficios significativos para ayudar a construir modelos de sistemas de software rigurosos y donde es posible mantener la trazabilidad de manera consistente. Enterprise Architect soporta este proceso en un ambiente fácil de usar, rápido y flexible.

2.6.3 Implementación.

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 SceneToolKit.

El lenguaje de programación escogido es C++ por ser un lenguaje de alto nivel, versátil, potente y general, estandarizado por la norma ISO/IEC 14882:1998, soporta la programación orientada a objetos y la programación genérica. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores, dando la posibilidad de redefinirlos, comúnmente conocido como la sobrecarga de operadores, y expresiones, flexibilidad, identificación de tipos en tiempo de ejecución, concisión y eficiencia. Además presenta una biblioteca estándar muy poderosa y es muy usado tanto en el ámbito educacional como profesional.

Visual Studio 2003 es una herramienta poderosa, fuerte y voluminosa que tiene una gran integración de varios lenguajes entre ellos el C++, C#, y Asp.Net. Tiene la posibilidad de implementar aplicaciones para soluciones integrales que aprovechen de manera óptima la ventaja de cada lenguaje.

El convenio de nomenclatura y estándares de codificación que se sigue es el mismo usado en la SceneToolKit para no entrar en contradicciones de notación.

3

Capítulo 3 : Descripción de la Solución Propuesta

Introducción

El presente capítulo se enmarca en la conceptualización de la solución propuesta, desde la perspectiva de las necesidades del cliente, y con una visión práctica del sistema a construir, teniendo en cuenta las necesidades del cliente y aplicando las técnicas a utilizar propuestas en el capítulo anterior.

Seguidamente se detalla el modelo de dominio, los 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 Modelo de Dominio

A continuación se muestra el diagrama de dominio que servirá como punto de partida para los posibles usuarios, ofreciendo mayor comprensión de todos los términos utilizados en la elaboración del Editor de Propiedades Físicas y que se utilizará como guía para el futuro diseño del sistema.

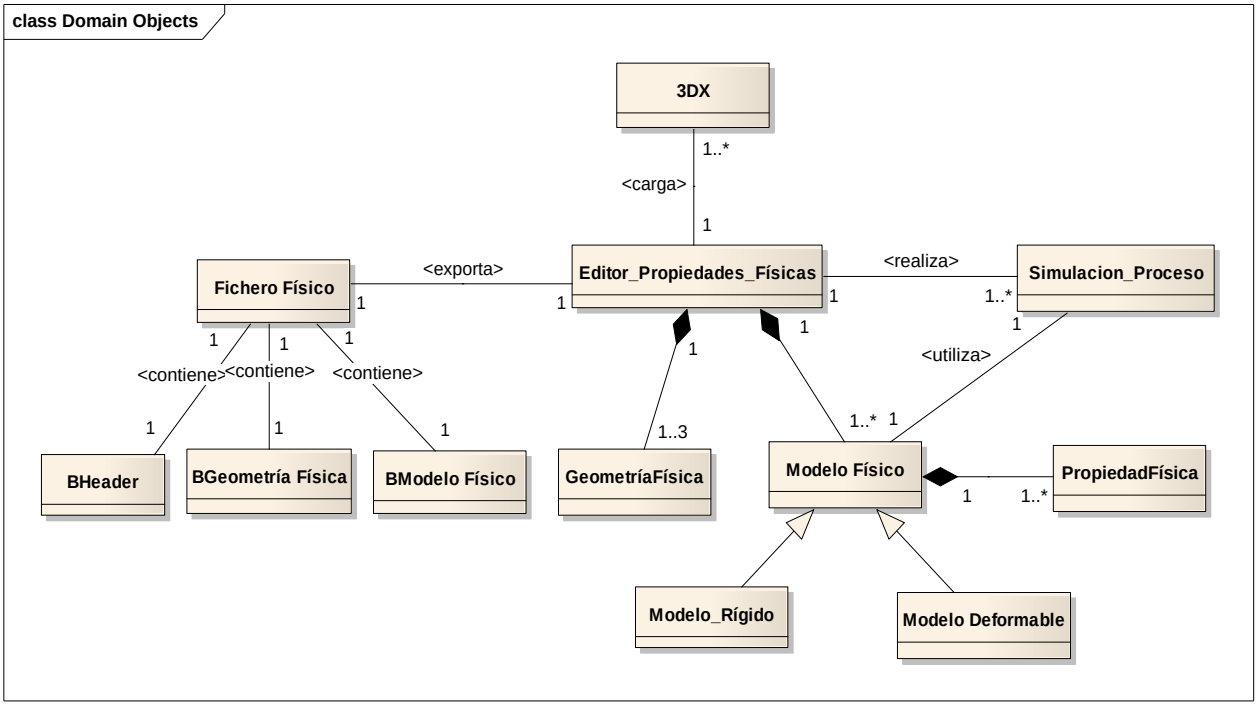


Fig 3. 1: Modelo de Dominio de la aplicación.

3.1.1 Glosario de términos del modelo de dominio.

Adelante se plantea una breve descripción de los principales conceptos que se manejan en el modelo de dominio para facilitar la familiarización con los términos que en él se tratan.

3DX: Se denomina al fichero que se carga inicialmente y al objeto que este represente se le aplicarán las características físicas en dependencia del modelo que se seleccione.

Editor de Propiedades Físicas: Se denomina a la herramienta que se obtendrá como producto final.

Fichero Físico: Se denomina al fichero que se exportará y que contiene todo los aspectos físicos que le han sido adicionados al modelo de cuerpo inicial, así como la versión con la que se trabaja en ese momento, la geometría que se representa, el tipo de modelo físico utilizado, entre otros.

BHeader: Se denomina al bloque cabecera del fichero físico, que contiene los atributos expuestos en la solución propuesta.

BGeometría Física: Se denomina al bloque geométrico del fichero físico donde se almacena el tipo de modelo que se aplicó, las dimensiones de la geometría cargada, entre otros atributos ya mostrados con anterioridad.

BModelo Físico: Se denomina al bloque del modelo del fichero físico donde se guardan las estructuras correspondientes a cada uno de ellos y dentro de estas todos los atributos físicos correspondientes.

Simulación de Proceso: Se denomina a la simulación del comportamiento de la geometría, asociada al objeto cargado, con el modelo que se ha seleccionado y sus constantes físicas introducidas. Es preciso aclarar que en esta primera versión solo se simulará un solo modelo, el “Rígido”.

Modelo Físico: Se le denomina al procedimiento que encierra un conjunto de parámetros para describir el comportamiento de un objeto en el tiempo.

Modelo Rígido: Se denomina al modelo físico que define las características y el comportamiento de los cuerpos rígidos en un entorno determinado. Contiene atributos como: masa, posición inicial, velocidad (lineal, angular).

Modelo Deformable: Se denomina al modelo físico que define las características y el comportamiento de los cuerpos deformables o blandos en un entorno determinado. Contiene atributos como: masa, constante elástica, amortiguación.

Propiedad Física: Se le denomina a los atributos físicos distintivos de cada modelo físico.

Geometría Física: Se le denomina a las estructuras geométricas que se podrán asignar para representar el comportamiento físico del objeto cargado, estas geometrías pueden ser cubos, cilindros o esferas.

3.2 Especificación de los requisitos del software.

Los requisitos constituyen capacidades o condiciones que el sistema debe cumplir. Estos facilitan el entendimiento entre usuarios y desarrolladores del sistema a elaborar. A continuación se exponen los requisitos funcionales por los que se regirá el sistema y los no funcionales que exponen las características de la aplicación.

3.2.1 Requisitos Funcionales.

Los siguientes requisitos establecen las funcionalidades e instrucciones que el sistema debe cumplir en su implementación.

El sistema debe permitir:

R1. Cargar fichero

R1.1 Seleccionar el directorio donde se encuentran guardados los ficheros.

R1.2 Cargar el entorno en el que desea trabajar (que también sería un fichero 3DX).

R1.3 Cargar objeto 3DX (que sería él o los cuerpos a tratar en ese entorno).

R2. Seleccionar Objeto.

R3. Asignar modelo físico.

R3.1 Seleccionar modelo físico.

R3.2 Introducir valores físicos correspondientes al método seleccionado.

R4. Simular Comportamiento Físico.

R5. Crear fichero físico.

R6. Exportar el fichero físico.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

3.2.2 Requisitos No Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Requerimiento de Usabilidad: Los futuros usuarios del sistema serán especialistas con conocimientos básicos de física y de la terminología afín. El producto debe estar concebido con una interfaz amigable que posibilite que el usuario haga lo que desea hacer sin grandes dificultades, por lo que este requerimiento debe estar presente en alto grado en el producto final.

Requerimiento de Portabilidad: El producto debe estar preparado para que con pequeñas modificaciones pueda ser multiplataforma.

Requerimiento de Software: Se requiere el SO Windows XP o superior.

Requerimiento de Hardware: Compatibilidad con tarjetas gráficas de video de 128 MB. Tarjeta RAM 512 MB en adelante

Requerimiento de Diseño e Implementación: Se utilizará el lenguaje de implementación C++, regido por la filosofía de Programación Orientada a Objetos. Se utilizará el IDE Microsoft Visual Studio .NET 2003 sobre el sistema operativo Windows y la herramienta SceneToolKit. El diseño de la interfaz será en Photoshop y otros componentes en 3DMaxStudio.

3.3 Modelo de Caso de Uso del Sistema

En esta sección se reconocen los actores del sistema a desarrollar y se conciben a través de la agrupación de los requisitos funcionales anteriormente hallados los resultados de valor que le pueda brindar a sus actores, o lo que es lo mismo, los casos de uso del sistema y se representan mediante un diagrama de casos de uso utilizando las facilidades que brinda UML.

3.3.1 Actores del sistema

Los actores representan entidades que interactúan con el sistema. Un actor no es parte del sistema, es un rol de un usuario, que puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema, y se beneficia con los resultados de las funcionalidades del mismo. En este caso con el Editor de Propiedades Físicas interactúa sólo un actor que se puntualiza a continuación:

Tabla 3. 1: Actor del Sistema.

Actores	Justificación
Usuario Especialista	Es el que se beneficiará con las funcionalidades que brinda el Editor de Propiedades Físicas: asociar comportamiento físico a los cuerpos (rígidos o deformables) de forma independiente al código del simulador y probar dinámicamente dichos cambios para saber si son los esperados para el proceso de simulación que se desea.

3.3.2 Casos de uso del sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Un caso de uso puede "incluir" la funcionalidad de otro caso de uso con su propio comportamiento, en el modelo de casos de uso de este sistema se pueden encontrar relaciones de este tipo.

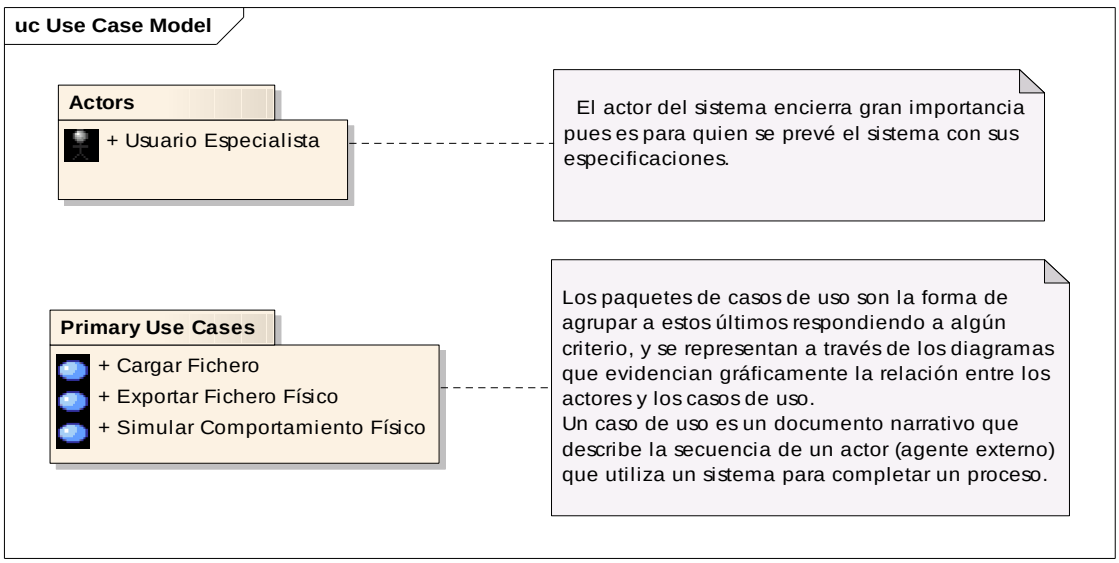


Fig 3. 2: Empaquetamiento del los Casos Usos y Actores del Sistema.

A continuación se describen los casos de usos expuestos anteriormente:

Tabla 3. 2: CU1 Cargar Fichero.

CU1	Cargar fichero
Actor	Usuario Especialista

Descripción	Tiene como propósito obtener e interpretar el fichero 3DX.
Referencia	R1.1, R1.2, R1.3

Tabla 3. 3: CU2 SimularComportamiento Físico.

CU2	Simular Comportamiento Físico
Actor	Usuario Especialista
Descripción	Permite mostrar el comportamiento del objeto que se selecciona en la escena, de acuerdo con la geometría que representa, en relación con el modelo físico escogido y las constantes de este, introducidas por el usuario.
Referencia	R2, R3.1, R3.2, R4

Tabla 3. 4: CU3 Exportar Fichero Físico.

CU3	Exportar Fichero Físico
Actor	Usuario Especialista
Descripción	Su propósito es crear un fichero con extensión .phs, y guardar dentro de él los bloques Header, GeometryPhysic y ModelPhysic ya definidos anteriormente con las propiedades añadidas por el usuario y luego exportarlo.
Referencia	R5, R6

3.3.3 Diagrama de Casos de Uso del Sistema.

A delante se presenta la relación entre los casos de uso del sistema y el actor que interactúa con ellos mediante el diagrama de casos de uso del sistema, donde se describe la funcionalidad propuesta, apoyada en la captura de los requisitos funcionales.

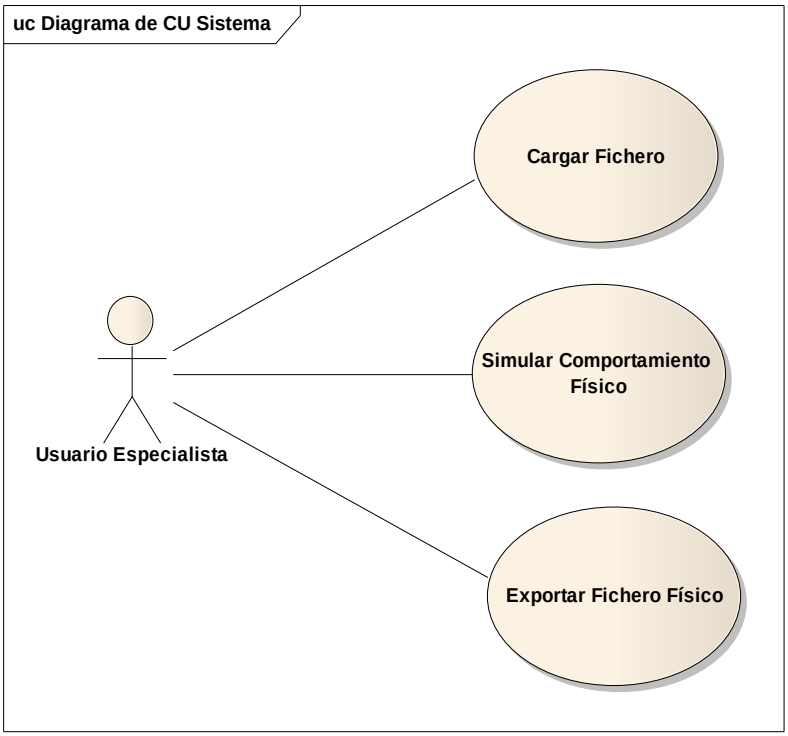


Fig 3. 3: Diagrama de CU del Sistema.

3.3.4 Descripción de los Casos de Uso del Sistema

¿Por qué resulta importante una descripción textual minuciosa de cada caso de uso? Para lograr entender con mayor facilidad la funcionalidad asociada a cada caso de uso no es suficiente una representación gráfica mediante un diagrama de casos de uso, sin lugar a dudas se precisa de una descripción detallada de cada uno de ellos por independiente lo cual se logra a través de la expansión de los mismos; a continuación se muestran una serie de tablas que agrupan los flujos operacionales de cada funcionalidad definida anteriormente.

Tabla 3. 5: Descripción CU Cargar Fichero.

Caso de uso	
Nombre	Cargar fichero.
Actores	Usuario Especialista.
Propósito	Tiene como propósito obtener e interpretar el fichero 3DX.

Resumen	El caso de uso se inicia cuando el usuario especialista selecciona la opción “archivo” y luego selecciona el entorno que desea cargar así como los objetos de los cuales se desea probar su comportamiento en el entorno seleccionado y consecutivamente la herramienta los muestra en escena.
Referencias	R1.1, R1.2, R1.3
Curso Normal de los Eventos	
Acción del actor	Respuesta del sistema
1. El usuario especializado selecciona la opción “Archivo”.	1.1. Se muestra un menú con las opciones de los distintos tipos de ficheros que se pueden abrir.
2. El usuario selecciona la opción “Cargar Entorno”.	2.1. Se muestra un cuadro de diálogo con posibles ficheros a cargar.
3. Selecciona el fichero que desea cargar y selecciona la opción abrir fichero.	3.1. Cierra el cuadro de diálogo y procede a cargar el fichero seleccionado por el usuario. 3.2. Muestra en pantalla el fichero cargado.
4. El usuario selecciona la opción “Cargar Objeto”.	4.1. Se muestra un cuadro de diálogo con posibles ficheros a cargar.
5. Selecciona el fichero que desea cargar y selecciona la opción abrir fichero.	5.1. Cierra el cuadro de diálogo y procede a cargar el fichero seleccionado por el usuario. 5.2. Muestra en pantalla el fichero cargado.
Curso Alternativo de los Eventos	
3. Selecciona el fichero que desea cargar y selecciona la opción abrir fichero.	3.1 Si la extensión del fichero seleccionado no es .3dx el sistema muestra un mensaje

	de error.
5. Selecciona el fichero que desea cargar y selecciona la opción abrir fichero.	5.1 Si la extensión del fichero seleccionado no es .3dx el sistema muestra un mensaje de error.
Postcondiciones	Se visualiza en pantalla cada uno de los ficheros .3dx cargados.
Prioridad	Crítico.

Tabla 3. 6: Descripción CU Simular Comportamiento Físico.

Caso de uso	
Nombre	Simular Comportamiento Físico.
Actores	Usuario Especialista.
Propósito	Permite mostrar el comportamiento del objeto en dependencia de la geometría que tiene asociada y en correspondencia con el modelo físico que haya sido seleccionado por el usuario.
Resumen	Se inicia cuando el usuario especialista selecciona el objeto al cual se le aplicarán los cambios físicos, selecciona además el modelo físico a utilizar e introduce sus valores correspondientes, y por último selecciona la opción “Ver” que visualiza el comportamiento del fichero en el entorno.
Referencias	R2, R3, R4.
Precondiciones	Debe haberse realizado el CU Cargar Fichero.
Curso Normal de los Eventos	
Acción del actor	Respuesta del sistema
1. Selecciona el objeto del cual quiere ver su comportamiento.	1.1. Muestra un mensaje comunicándole al usuario el objeto que ha sido seleccionado.

2. Selecciona la opción “Modelo Físico”.	2.1. Muestra los modelos posibles a utilizar.
3. Selecciona el Modelo Físico a utilizar.	3.1. Se ejecuta alguna de las siguientes acciones: a) Si el usuario especializado selecciona el Modelo Rígido consultar sección Modelo Rígido. b) Si el usuario especializado selecciona Modelo Deformable consultar sección Modelo Deformable.
Postcondiciones	Se visualizará en la pantalla el comportamiento que propone el modelo seleccionado, también en dependencia de la geometría que tenga asociada el objeto.
Prioridad	Crítico.

Tabla 3. 7: Descripción de la Sección Modelo Rígido del CU Simular Comportamiento Físico.

Sección	Modelo Rígido.
CU al que pertenece	Simular Comportamiento Físico.
Actores	Usuario Especialista.
Propósito	Permite introducir valores físicos que harán que el fichero cargado tome un comportamiento como un cuerpo rígido.
Resumen	Se inicia cuando el usuario especialista selecciona el “Modelo Rígido”, y luego procede a introducir todos los valores de cada parámetro que exige el modelo.
Referencias	R3, R4
Precondiciones	Debe haberse realizado el CU Cargar Fichero.

Curso Normal de los Eventos	
Acción del actor	Respuesta del sistema
1- El usuario especializado selecciona la opción “Modelo Rígido”.	1.1. Se muestra un panel que contiene todos los atributos físicos propios del modelo.
2. Introduce los valores de la masa, el vector velocidad lineal (x, y, z), el vector velocidad angular (x, y, z), y el vector posición inicial (x, y, z) y se selecciona la opción “Ver”.	2.1. Muestra el comportamiento físico del objeto que describe el modelo seleccionado.
Postcondiciones	Se visualiza el comportamiento del objeto que debe ser similar al de un cuerpo rígido en tiempo real.
Prioridad	Crítico.

Tabla 3. 8: Descripción de la Sección Modelo Deformable del CU Simular Comportamiento Físico.

Sección	Modelo Deformable.
CU al que pertenece	Simular Comportamiento Físico.
Actores	Usuario Especialista.
Propósito	Permite introducir valores físicos que harán que el fichero cargado tome un comportamiento como un cuerpo deformable.
Resumen	Se inicia cuando el usuario especialista selecciona el “Modelo Deformable”, y luego procede a introducir todos los valores de cada parámetro que exige el modelo.
Referencias	R3, R4
Precondiciones	Debe haberse realizado el CU Cargar Fichero.
Curso Normal de los Eventos	
Acción del actor	Respuesta del sistema
1- El usuario especializado selecciona la opción “Modelo Deformable”.	1.1. Se muestra un panel que contiene todos los atributos físicos propios del modelo.

2. Introduce los valores de la masa, la amortiguación, y la constante elástica y se selecciona la opción “Ver”.	2.1. Muestra el comportamiento físico del objeto que describe el modelo deformable.
Postcondiciones	Se visualiza el comportamiento del objeto que debe ser similar al de un cuerpo deformable en tiempo real.
Prioridad	Crítico.

Tabla 3. 9: Descripción del CU Exportar Fichero.

Caso de uso	
Nombre	Exportar Fichero.
Actores	Usuario Especialista.
Propósito	Su propósito es definir el fichero .phs y guardarlo en la misma dirección donde se cargó el objeto seleccionado.
Resumen:	Se inicia cuando el usuario selecciona la opción de “Editar”, luego se crea un fichero físico con extensión .phs, en el se guardan los bloques Header, PhysicGeometry y PhysicModel anteriormente definidos y se exporta al mismo directorio donde se cargó el objeto seleccionado.
Referencias	R5, R6
Precondiciones	Debe haberse realizado el CU Simular Comportamiento Físico.
Curso Normal de los Eventos	
Acción del actor	Respuesta del sistema
1. El usuario especialista selecciona la opción “Editar”.	1.1. Se crea un fichero físico .phs con el nombre de la geometría que tiene asociada el objeto cargado. 1.2. Se define y se copia al fichero creado los bloques Header, PhysicGeometry y

	PhysicModel ya definidos con anterioridad. 1.3. Se guarda el fichero creado en el mismo directorio donde fue cargado el objeto seleccionado.
Poscondiciones	Se genera el fichero físico con extensión *.phs correspondiente al modelo seleccionado.
Prioridad	Crítico.

4

Capítulo 4 : Diseño e Implementación del Sistema

Introducción

En este capítulo se define la estructura que tendrá el sistema, tomando como base todos los requisitos planteados en la sección anterior. Se muestran los artefactos involucrados en el diseño e implementación de la herramienta que se pretende realizar a través de diagramas relacionales como son: Diagrama Lógico del Sistema, Diagrama de Paquetes, Diagrama de Clases de Diseño, Diagramas de Secuencia y por último el Diagrama de Componentes en el que se define como se harán físicas las clases de diseño en el lenguaje seleccionado. Se especifican también otros aspectos para la comprensión de los diagramas.

4.1 Diseño lógico del sistema.

El diseño es un refinamiento del proceso de análisis del sistema enfocado en cómo el sistema cumple los objetivos propuesto y partiendo pues, de los requisitos no funcionales. El diagrama de clases del diseño se concibe a través de paquetes para prever una mayor claridad y comprensión de las clases que lo integran.

En la fig.4.1 se muestra una representación lógica del sistema donde se relacionan las clases del diseño y además se plasman otras ya definidas en la SceneToolKit, que son reutilizadas en la implementación y sin las cuales sería bastante complejo entender la arquitectura de la misma y su relación con esa herramienta. Posteriormente se desarrollan el resto de los artefactos aun cuando se mencionen o representen elementos de la SceneToolKit que son arquitectónicamente significativos para el sistema.

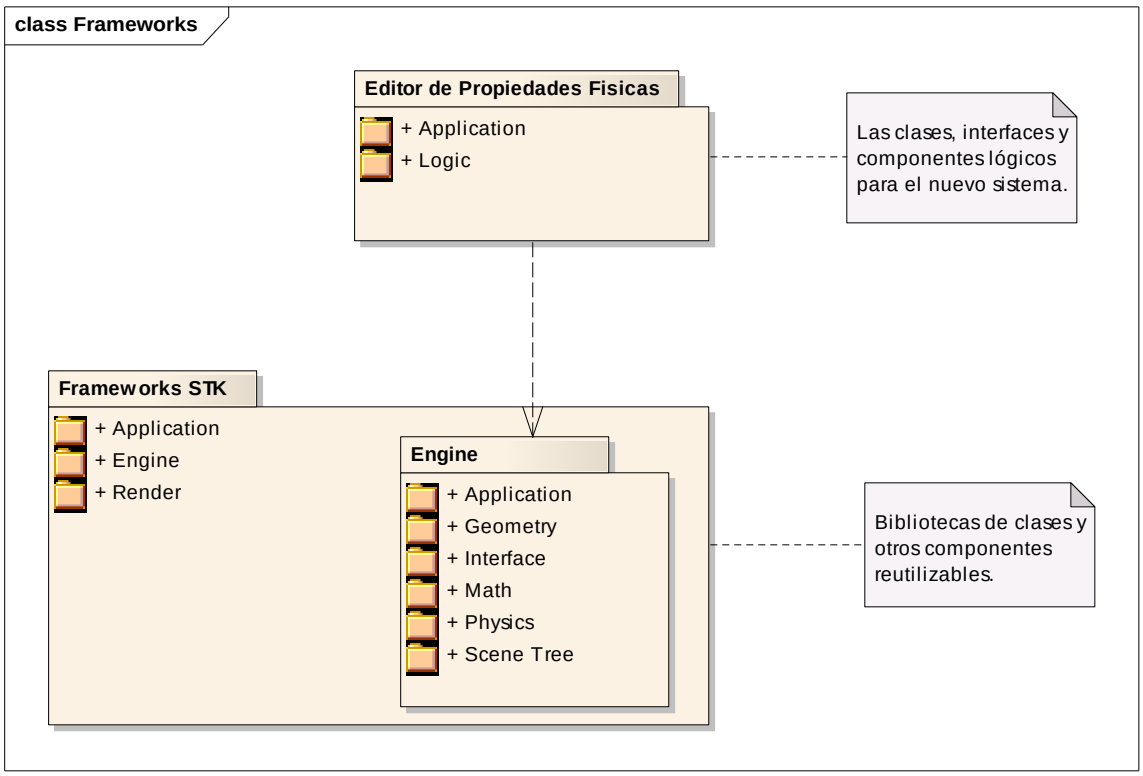


Fig 4. 1: Diseño Lógico del Sistema.

El paquete Application encierra la clase *Editor* que es la rectora de todo el proceso de edición que se pretende con el sistema, y el Logic involucra todas las clases que determinan la lógica del sistema: *EPhysicModel c*ontiene enumerativamente los modelos físicos que se pueden usar en el proceso de edición y simulación, *SRigidBody* guarda la estructura del Modelo Rígido con sus indicadores, *SDeformBody* comprende la estructura del Modelo Rígido con sus indicadores; *SGeometryBox, SGeometryCylinder, SGeometrySphere* involucra las estructuras correspondientes a cada geometría con sus atributos propios; *SHeader* define una estructura para la cabecera del fichero que se exportará como resultado final de la edición.

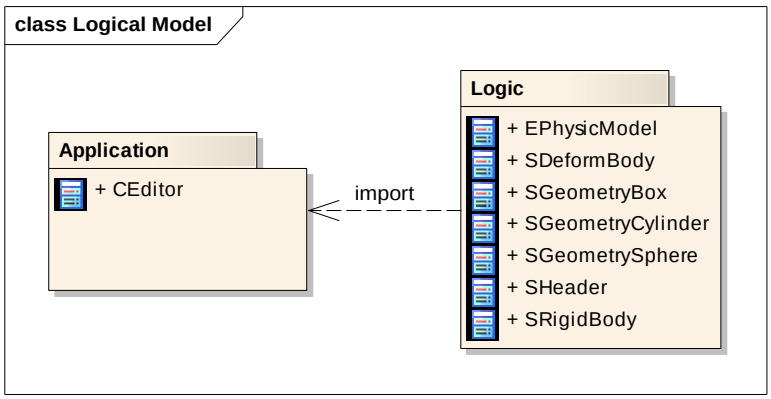


Fig 4. 2 Clases del Paquete Application y Logic.

4.2 Diagrama de Paquetes de Diseño.

Los paquetes de diseño son usados para agrupar elementos del modelo de diseño relacionados con propósitos organizacionales y frecuentemente para administración de configuración, pueden ser relacionados a través de un diagrama de paquetes.

En la fig.4.3 se destacan los paquetes de diseño del sistema de forma general, las trazas que existen entre ellos y las clases que los componen.

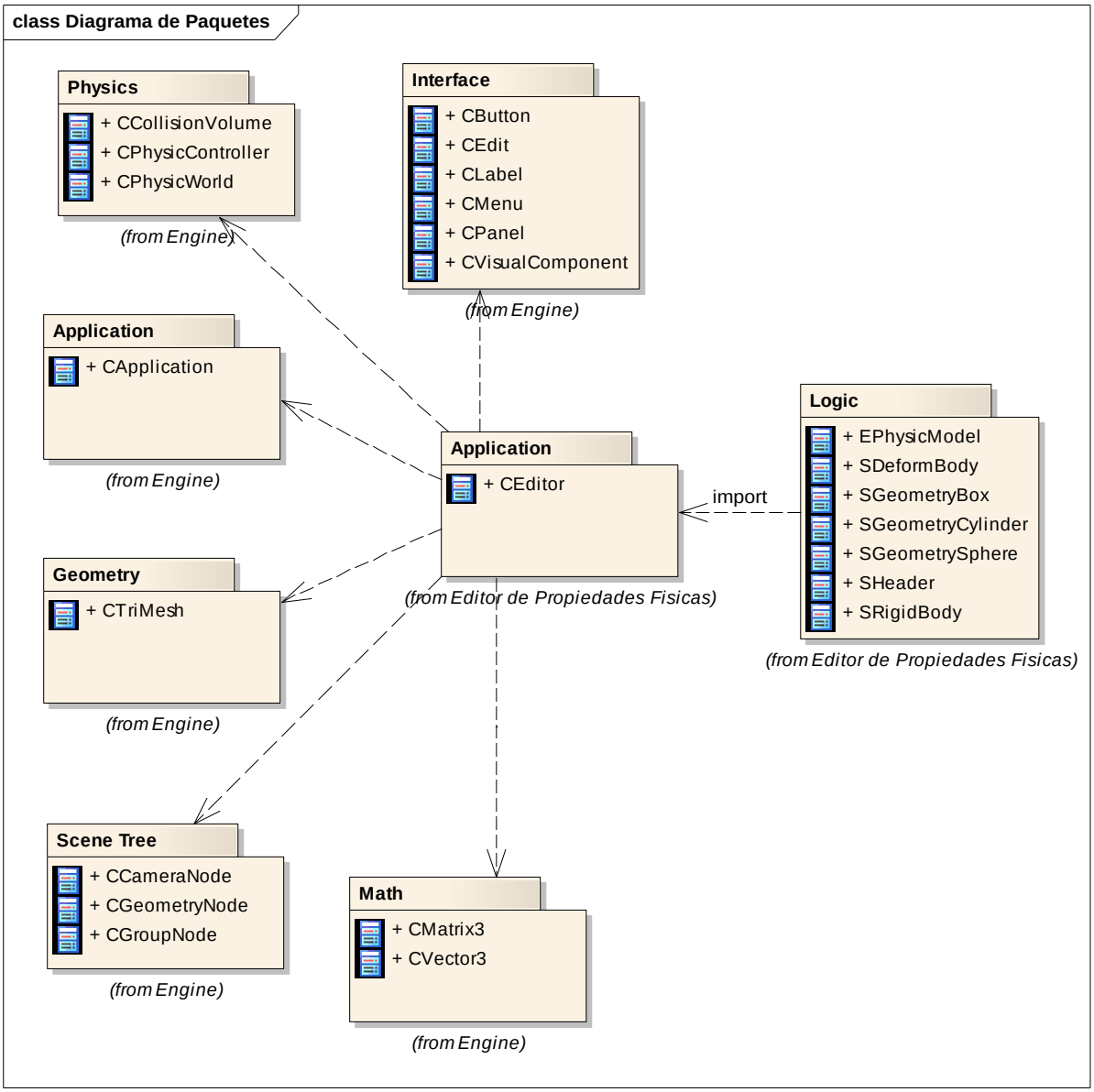


Fig 4. 3: Diagrama de Paquetes General del Sistema.

4.3 Diagrama de Clases de Diseño.

El diagrama de clases de diseño representa un modelo del objeto que describe la realización de casos de uso, y sirve como una abstracción del modelo de implementación y su código fuente. Este se usa como la entrada esencial a las actividades en la implementación. De forma general la fig.4.7 constituye el modelo de diseño de la aplicación, excluyendo algunas particularidades de la clase CEditor que se exponen con más claridad en el **Anexo 8** para no hacer engorrosa la representación.

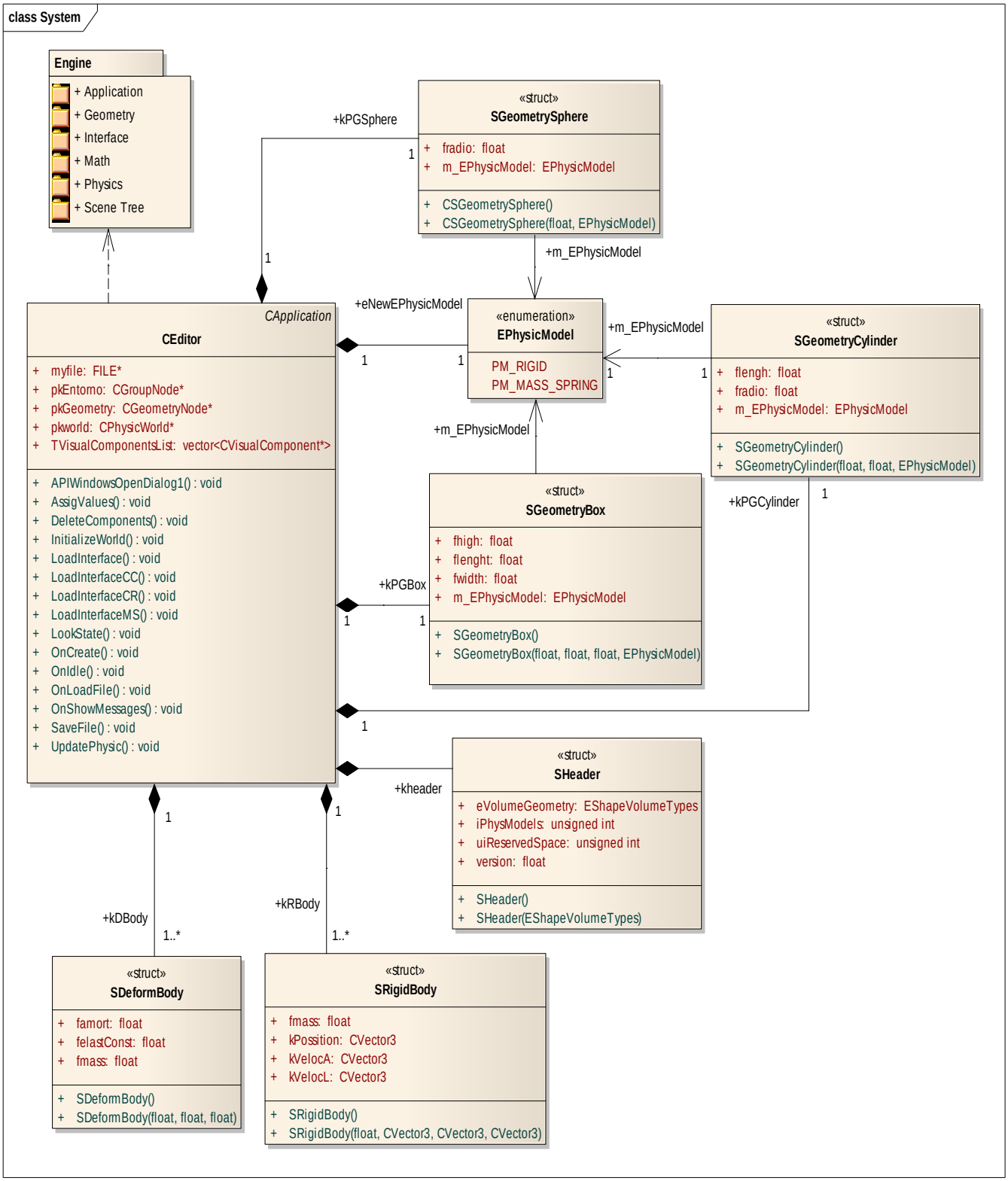


Fig 4. 4: Diagrama de Clases de Diseño General.

4.4 Diagramas de Secuencia.

Luego de la expansión de cada caso de uso destacado en el capítulo anterior, se pueden graficar los diagramas de secuencias que dan una idea más precisa del tiempo de ejecución de cada algoritmo que interviene en los CU.

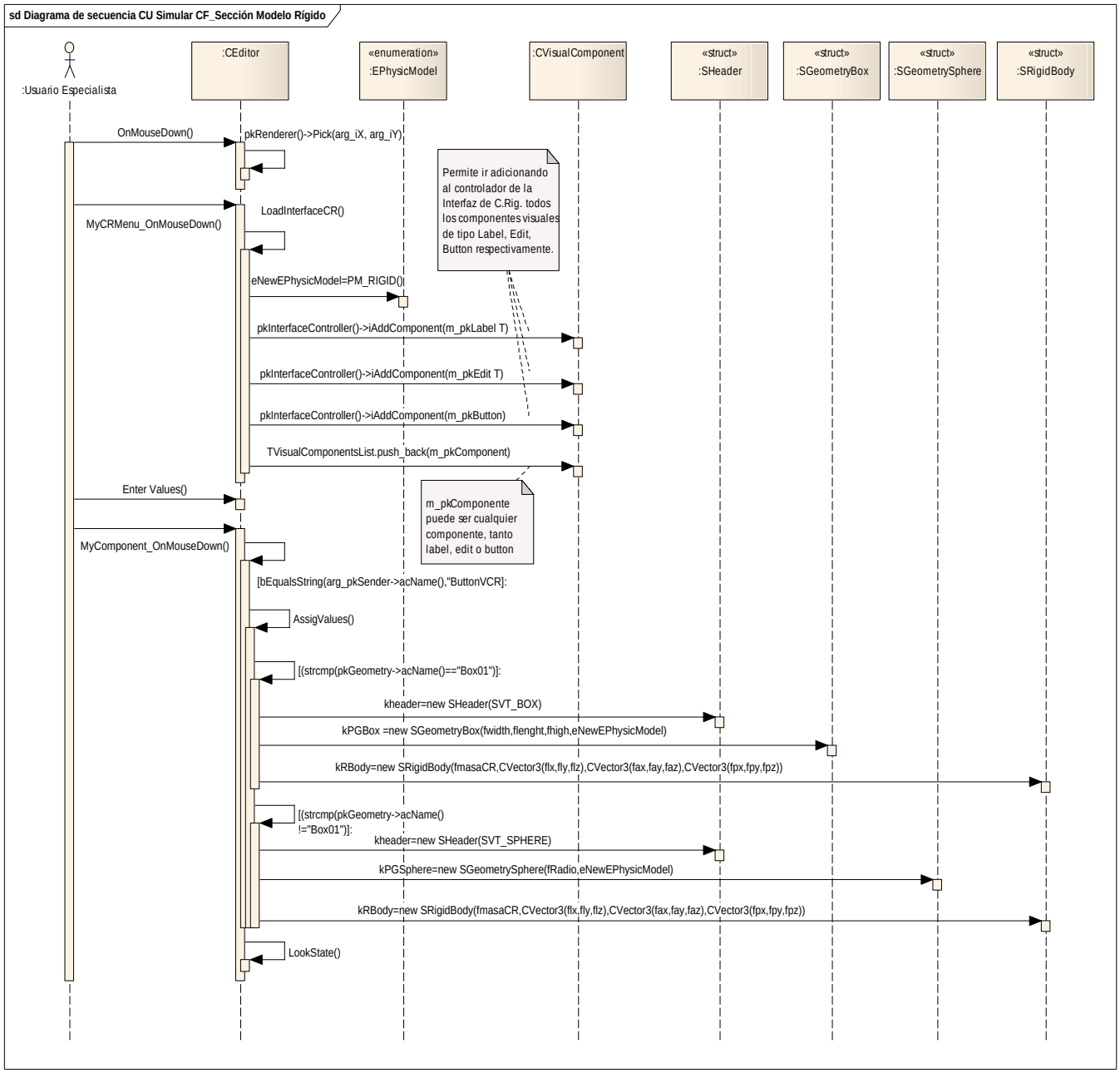


Fig 4. 5: Diagrama de Secuencia CU Simular Comportamiento Físico, Sección Modelo Rígido.

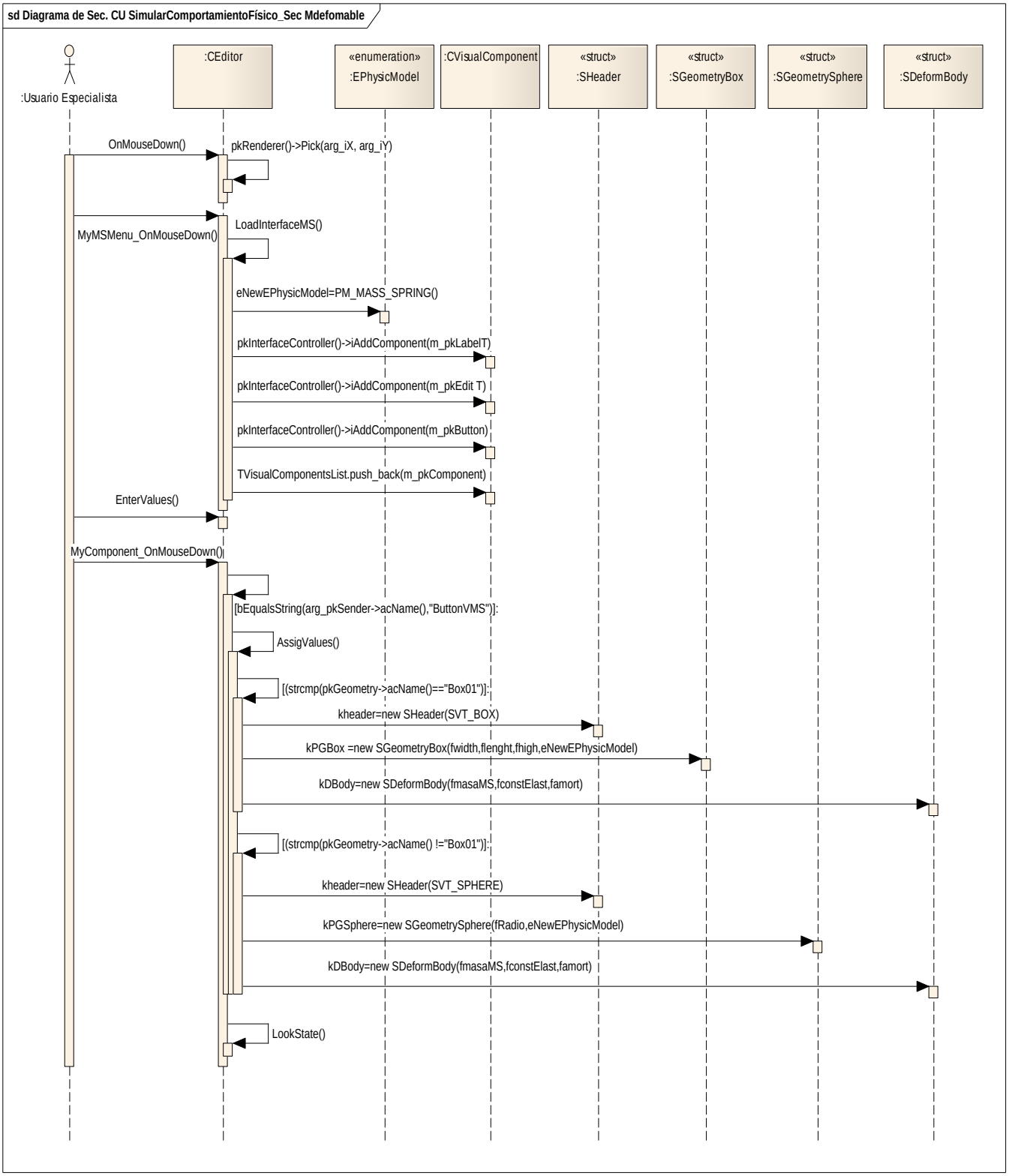


Fig 4. 6: Diagrama de Secuancia CU Simular Comportamiento Físico, Sección Modelo Deformable.

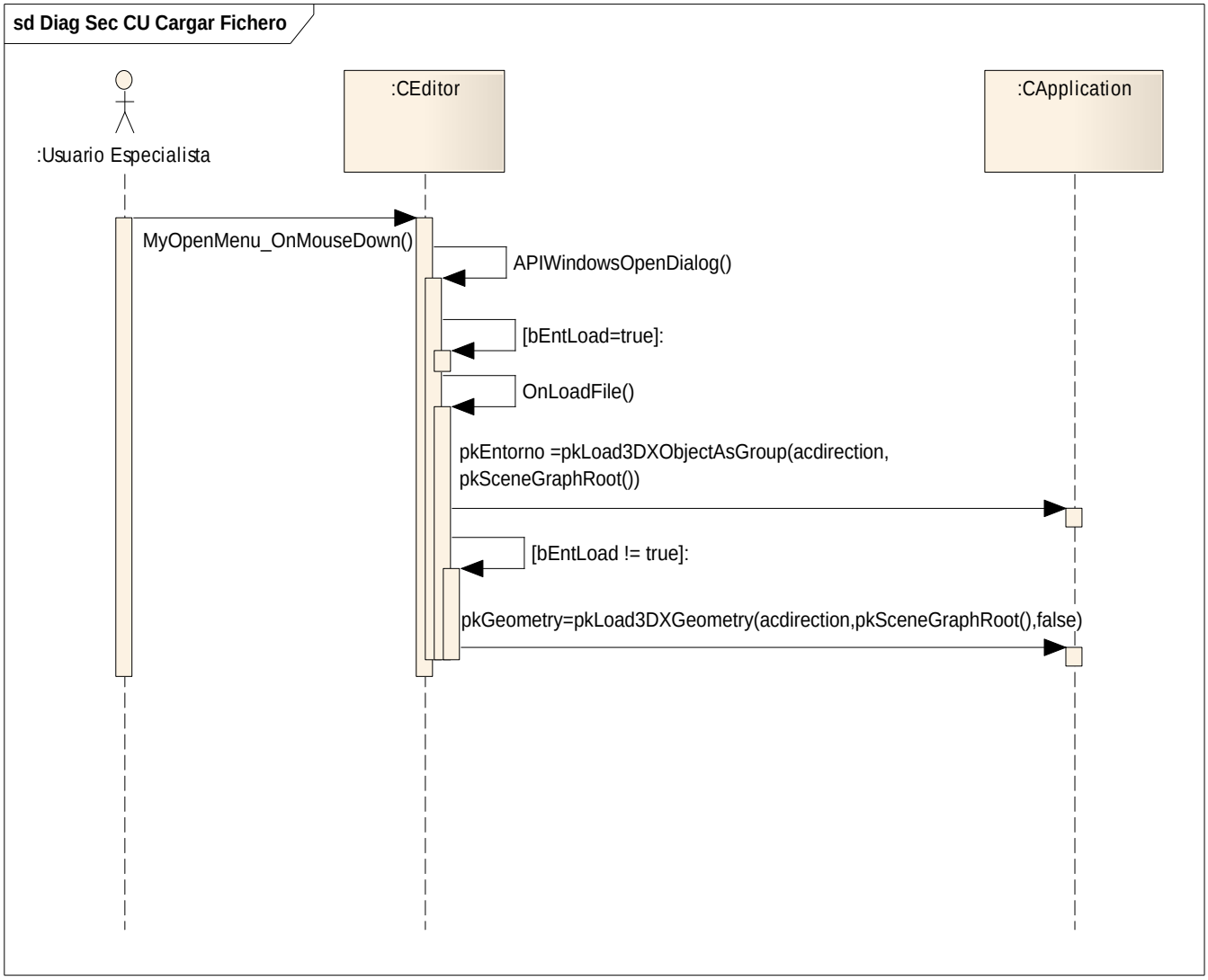


Fig 4. 7: Diagrama de Secuencia CU Cargar Fichero.

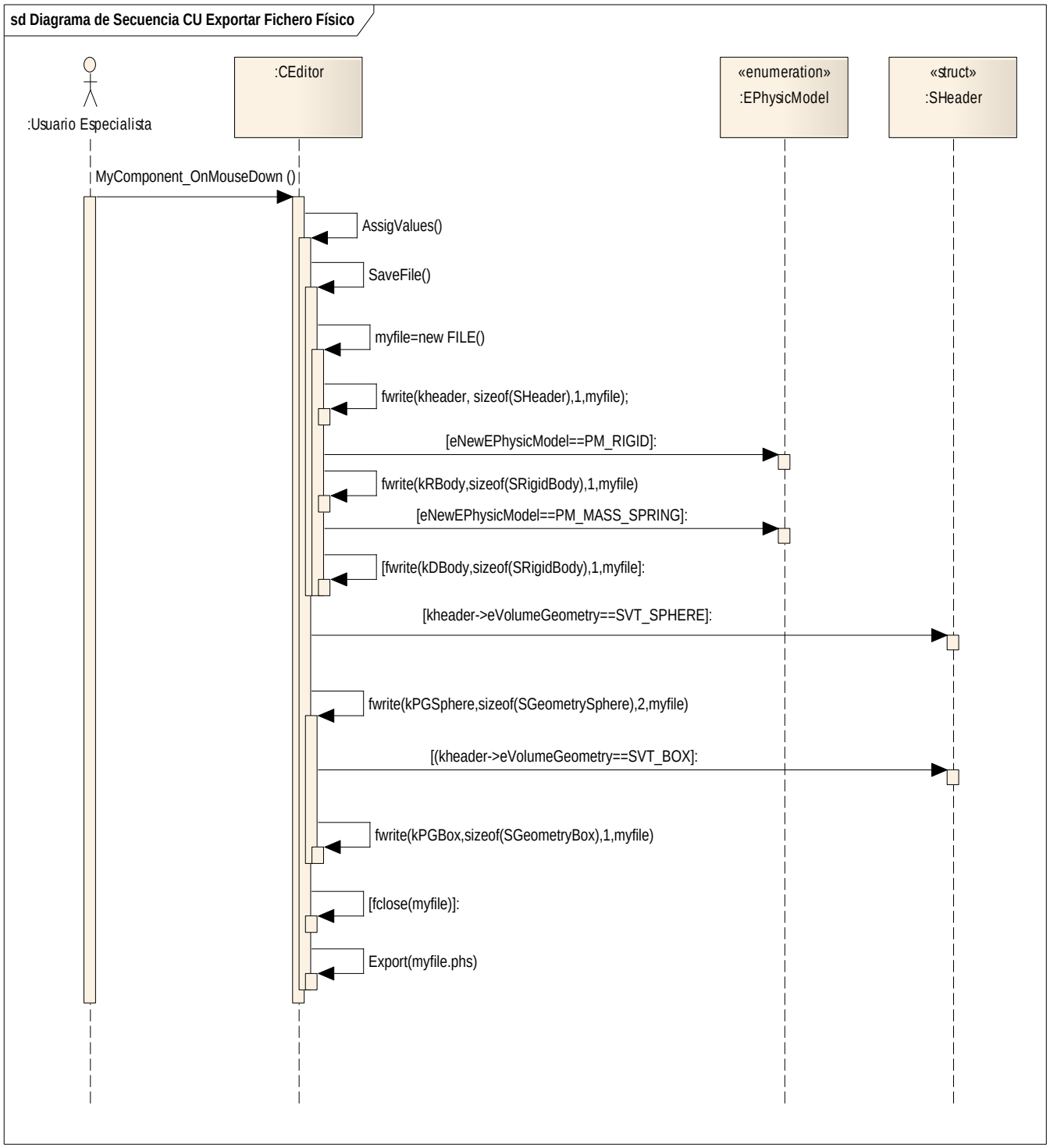


Fig 4. 8: Diagrama de Secuencia CU Exportar Fichero.

4.5 Diagrama de Despliegue.

Un diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo, muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Están formados por instancias de los componentes software que representan manifestaciones del código en tiempo de realización. El diagrama de despliegue para este sistema resulta muy simple, incluye sólo la representación gráfica de una PC, por tal motivo se decide que no es necesario hacer la representación del mismo.

4.6 Diagrama de Componentes.

Un componente no es más que el empaquetamiento físico de los elementos del modelo de diseño, luego el diagrama de componentes muestra la organización y las dependencias entre un conjunto interconectado de estas unidades, además de cubrir la vista estática ejecutable de la implementación de la aplicación. A continuación se muestra el diagrama correspondiente a este sistema.

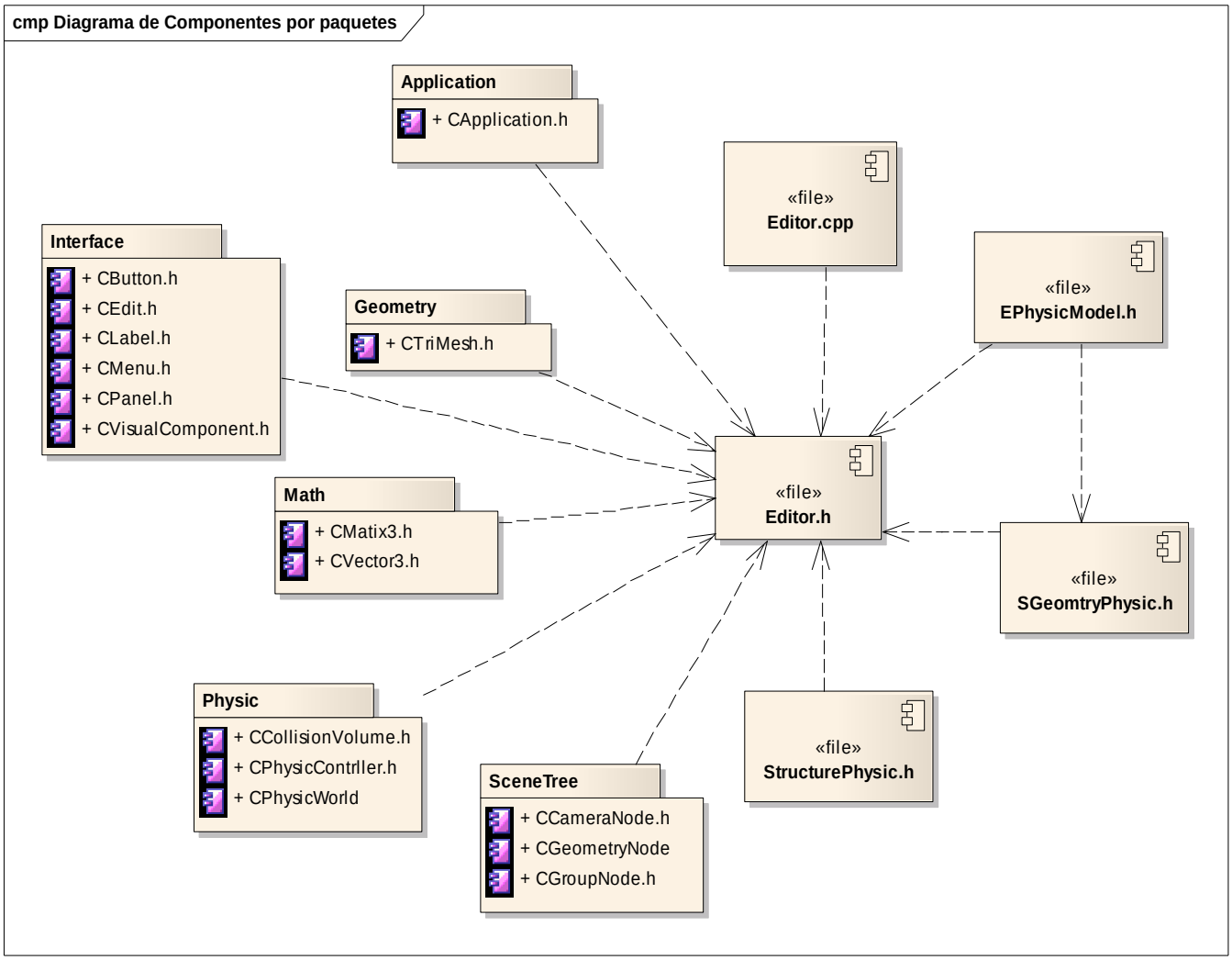


Fig 4. 9: Diagrama de Componentes General.

4.7 Nomenclatura y estándares de codificación.

El código de la herramienta sigue algunos estándares propuestos por el grupo de desarrollo (respetando los estándares de codificación para C++ (uso de espacios y líneas en blanco, etc.). Está programado en inglés, debido a 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.

Enumerados:

Para los enumerados se utiliza el indicador "**E**" en el nombre del tipo,

Ej: enum **E**MyEnum

{

ME_VALUE,

ME_OTHER_VALUE

};

Estructuras:

Se utiliza el indicador "**S**" para indicar que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

Ej: struct **S**MyStruct {…};

Clases:

Se utiliza el indicador "**C**" para indicar que es una clase. Ver más adelante la nomenclatura de las variables miembros.

Ej: class **C**ClassName;

Interfaces:

Se utiliza el indicador "**I**" para indicar que es una interfaz.

Ej: **I**MyInterfaceName

Listas e iteradores STD: vector<>TNameList;

TNameList::iterator TNameListIter;

Declaración de variables:

Los nombres de las variables comienzan con un identificador del tipo de dato al que correspondan. En el caso de que sean variables miembros de una clase, se le antepone el identificador "**m**_", si son globales se les antepone el identificador "**g_**", y en caso de ser argumentos de algún método, se les antepone el prefijo "**arg_**".

Ej:

bool **b**VarName;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name;

char* **ac**Name;

bool **m_b**MemberVarName;

char **g_c**GlobalVarName;

Intancias de tipos creados:

Ej:

EMyEnumerated **e**Name;

SMyStructure **k**Name;

CClassName **k**ObjectName;

CClassName* **pk**Name;

CClassName* **ak**Name;

CClassName* **m_ak**Name;

IMyInterface* **pI**Name;

Constructor y destructor:

Ej: CClassName (bool arg_bVarName, float& arg_fVarName);

~CClassName ();

Funciones:

Ej: bool bFunction1 (…);

int* piFunction2 (…);

CClassName* pkFunction3 (…);

Procedimientos:

Ej: void Procedure4 (…);

Métodos de acceso a miembros:

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

Ej:

int **m**_**i**MyVar;

int **i**MyVar();
void MyVar(int arg_**i**MyVar)

Conclusiones

Con la realización de este trabajo se logró definir una herramienta (Editor de Propiedades Físicas) para Sistemas de Realidad Virtual que permite incorporarle, de manera efectiva, un comportamiento físico a diseños 3D elaborados previamente, a partir de la carga de los cuerpos que estos representan, su edición teniendo en cuenta un modelo físico-matemático con indicadores que le aportarán la dinámica que se pretende alcanzar y finalmente creando un fichero que será el contenedor de las características físicas adicionadas, las cuales le proporcionan una actuación más real en los procesos de simulación que intervenga. Todo lo anterior tiene como ventaja que no es necesario acceder al código fuente previo, durante o luego de la ejecución de un determinado suceso simulado, pues algún cambio específico de algunas de estas propiedades puede realizarse de forma independiente mediante la herramienta.

De esta manera se logra dar cumplimento a los objetivos planteados ya que:

- ✓ Se hizo un estudio de varios formatos para el trabajo con ficheros 3D que intervienen en un proceso simulado, identificando los aspectos más significativos de sus estructuras, lo cual contribuyó a la carga eficiente de unos o varios de estos ejemplares que representan los cuerpos a ser editados desde el punto de vista físico, así como un entorno en el cual se desea realizar la simulación de los cambios.
- ✓ Se realizó el estudio y la caracterización de las propiedades de los modelos físicos definidos para lograr la simulación tanto de objetos rígido como blandos. Se seleccionaron aquellas que son constantes en el modelo pero que varían según las características específicas de cada cuerpo, las cuales formaron parte del Editor de Propiedades Físicas, donde se ilustraron diferentes comportamientos según los valores definidos para cada propiedad.
- ✓ Se utilizó la herramienta SceneToolKit desarrollada para el trabajo con aplicaciones de RV, teniendo en cuenta que tiene acoplado un engine físico que permite simular la conducta de sistemas rígidos actualizando el estado de estos cuerpos en el tiempo según los valores del modelo. Por lo que se pudo adicionar las cualidades necesarias para que cada elemento editado que interviniera en la simulación mostrara el mayor grado de realismo posible.
- ✓ Se definió un fichero físico .phs en función del estudio de las principales formas de organización, tal es el caso del .STK propio de la SceneToolKit, con el propósito de almacenar correctamente los datos de la geometría, el modelo físico y otros aspectos que se relacionan.

- ✓ Se verificó que el fichero .phs que se exportó contenía todos los atributos que anteriormente habían sido incorporados durante el paso de edición, con la realización de un DEMO que muestra como se comporta determinado cuerpo que tiene asociado un fichero físico .phs.

Recomendaciones

Se recomienda darle seguimiento a los siguientes aspectos del trabajo:

- ✓ Basados en el diseño planteado anteriormente realizar el acoplamiento del Módulo de Deformación de Objetos para Sistemas de Realidad Virtual de la SceneToolKit para poder probar los cambios experimentados en cuerpos con características deformables de la misma forma que se procede con los cuerpos rígidos en la presente edición.
- ✓ Lograr que se pueda obtener, modificar y editar las dimensiones de la geometría asociada al cuerpo cargado, que trae por defecto del 3DMax Studio.
- ✓ Se sugiere un estudio y desarrollo del trabajo con mallas pues en el mundo se están dando pasos significativos en este sentido.

Referencias Bibliográficas

[1] Realidad Virtual, Disponible en: http://www.monografias.com/trabajos/vr/vr.shtml

[Consultado: 2008]

[2] Wikipedia, La enciclopedia libre. [Consultado en: 2007]. "Formato de almacenamiento"

Disponible en: http://es.wikipedia.org/wiki/Formato_de_almacenamiento

[Consultado en: 2008].

[3] Wikipedia, La enciclopedia libre. "STL (File Format)"

Disponible en: http://en.wikipedia.org/wiki/STL_(file_format) [Consultado en: 2007].

[4] Pipho, Evan. "Focus On 3D Models", Premier Press. USA. 2003 Capítulo 7: The 3ds Models.

[5] Alexis Echemendia Gonzalez, Wendy García López**, "**Sistema de Generación de Ficheros para Entornos Virtuales**",** Universidad de las Ciencias Informáticas, 2007 [Consultado: 2008]

[6] Young-Min Kang, H.-G.C., Real-time Animation of Complex Virtual Cloth with Physical. Plausibility and Numerical Stability.Teleoperators & Virtual Environments [Consultado en: 2007].

[7] Lifshitz., L., *Mecánica: Mecánica del sólido rígido*. Reverté ed. 1991, Barcelona.

[Consultado en: 2007].

[8] Rosario, L.M., Deformación Plástica. Mecánica de sólidos deformables. [Consultado: 2007].

[9] Materia. Propiedades generales de la materia.

Disponible en: http://html.rincondelvago.com/materia.html [Consultado en: 2008]

[10] Propiedades Físicas de los Cuerpos. Disponible en:

http://es.wikipedia.org/wiki/Propiedades_f%C3%ADsicas_de_los_cuerpos [Consultado en: 2008]

[11] Ruiz, D.G., *Motores de Física.* [Consultado: 2088]

[12] Bucciarelli Jr. Louis L. "Engineering Mechanics for Structures".

Disponible en: http://web.mit.edu/emech/dontindex-build/index.html [Consultado en: 2007]

[13] Nealen A, Müller M, Keiser R, Boxerman E and Carlson M "Physically Based Deformable Models in Computer Graphics" EUROGRAPHICS 2005. [Consultado en: 2007].

[14] UCIENCIA, II Conferencia Científica de la Universidad de las Ciencias Informáticas, II Taller de Realidad Virtual, [Consultado en: 2007].

Disponible en: http://www.informaticahabana.com/evento_virtual/files/MUL036.pdf

[15] O'Brien J. F. "Graphical Modeling and Animation of Fracture" Georgia Institute of Technology, USA July 2000. [Consultado en: 2008].

[16] Mendoza C., Sundaraj K., Laugier C. "Issues in Deformable Virtual Objects Simulation with Force Feedback" SHARP Project, Saint Martin, Francia 2002. [Consultado: 2007]

[17] Drägner Ch. "A ChainMail Algorithm for Direct Volumen Deformation in Virtual Endoscopic Simulation". Vienna University of Technology. Mayo 2005. [Consultado: 2007]

[18] Nealen A, Müller M, Keiser R, Boxerman E and Carlson M "Physically Based Deformable Models in Computer Graphics" EUROGRAPHICS 2005. [Consultado: 2007]

[19] 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. [Consultado: 2007]

[20] Székely G et al. "Virtual Reality-Based Simulation of Endoscopic Surgery". Swiss Federal Institute of Technology, Zürich, Switzerland. June 3, 2000. [Consultado: 2008]

[21] Al-khalifah A., Roberts D. "Survey of modeling approaches for medical simulators" Centre for Virtual Environments, the University of Salford, Manchester, UK 2004. [Consultado: 2008]

[22] Carrara Studio2

Disponible en: http://www.macuarium.com/macuarium/actual/especiales/2002_10_23_carrara2.shtml [Consultado: 2008]

[23] 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. [Consultado: 2007]

[24] Prietoni N. "Physically Based Deformable Objects in Computer Graphics." Universidad de Geneva. [Consultado: 2007]

[25] Raissel Ramirez Orozco, Osley Bretau Camejo**, "**Deformación de Objetos para Sistemas de Realidad Virtual**",** Universidad de las Ciencias Informáticas, 2007. [Consultado: 2008]

Bibliografía Consultada

- ✓ Pairó, Joan Trias. Geometría para la Informática y CAD. 2003. Francia: SHARP Project, Saint Martin, 2002.
- ✓ Walter Mora F, Geovanni Figueroa M. Vectores, rectas y planos. Disponible en: http://www.cidse.itcr.ac.cr/cursos-linea/Algebra-Lineal/algebra-vectorial-geova-walter/node1.html.
- ✓ Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. La Habana: Editorial Félix Varela, 2004.
- ✓ Baraff David. Physically Based Modeling Rigid Body Simulation. 2001
- ✓ Ivar Jacobson, Grady Booch, James Rumbaugh. El Proceso Unificado de Desarrollo de Sofware. La Habana: Felix Varela, 2004.
- ✓ Erleben Kenny. Physics Primer Basic Rigid Body Physics. 2002.
- ✓ Ladislav Kavan. Rigid Body Collision Response. 2003
- ✓ LaMothe A, “Tricks of the 3D Game Programming Gurus” Indianapolis, SAMS, 2003, 1601.
- ✓ Terzopoulos D, Waters K “Physically-Based Facial Modeling, Analysis, and Animation” Journal of Visualization and Computer Animation, 1(2):73–80, 1990.
- ✓ Kühnapfel U, “Endoscopy Surgery Training Using Virtual Reality and Deformable Tissue Simulation.” Karlsruhe, Alemania, 2000.
- ✓ Erleben Kenny. Physics Primer Basic Rigid Body Physics. 2002.
- ✓ Liudmila Pupo Peña, Liudmila Reyes Álvarez. “Módulo para simulación de la dinámica de los cuerpos rígidos”, Universidad de las Ciencias Informáticas, 2007 [Consultado: 2008]

Glosario de Abreviaturas

***SDK*:** Software Development Kit (kit de desarrollo de software): es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto.

***RV*:** Realidad Virtual.

***SRV*:** Sistemas de Realidad Virtual.

***UCI*:** Universidad de la Ciencias Informáticas.

***3D*:** Tres dimensiones, en la mayoría de los casos se representan con las letras x, y, z.

3DS: 3D Studio (file format).

ASCII: American Standard Code for Information Interchange.

ASE: ASCII Scene Exporter.

DIRECT X: Es una colección de APIs creadas para facilitar tareas relacionadas con la programación de juegos en la plataforma Microsoft Windows.

***MD3*:** Nombre del formato que da Id Software al juego Quake 3.

OBJ: Nombre del formato creado por Alias Wavefront.

***STL*:** Standard Tessellation Language.

Glosario de Términos

A:

Animación: Simulación de un movimiento creada por la muestra de una serie de imágenes o fotogramas.

*****

B:

Binario*:* En matemática el sistema binario es un sistema de numeración en el que los números se representan utilizando las cifras cero y uno ('0' y '1').

*****

C:

C++: Lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos (POO).

Comportamiento *(behavior):* cambio de atributos o características de los objetos visibles y no visibles del mundo virtual y que cambian el aspecto visual de la escena logrando una animación.

*****

E:

Escena: Cada una de las partes de que consta una obra dramática, una película, o una animación y que representa una determinada situación, con los mismos personajes u objetos.

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.

Engine*:* Motor, traducido al español.

*****

F:

***Fichero**:* Agrupación de información relacionada que puede ser manipulada de forma unitaria por el sistema operativo de un ordenador, tratada como una unidad de almacenamiento en memoria secundaria y organizada de forma estructurada para facilitar la búsqueda de datos individuales.

***Formas**:* Objeto compuesto por líneas y vértices en el espacio.

*****

H:

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

*****

I:

***Instancias de un objeto**:* Es un grupo de objetos que mantienen las mismas características y tienen una estrecha relación, de tal manera que si ocurre un cambio en uno de ellos el resto sufre el mismo cambio.

*****

J:

***Jerarquía de vértices**:* Es una lista ordenadas de vértices teniendo en cuanta algún tipo de prioridad.

*****

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:

Módulo: Es un componente auto controlado de un sistema, el cual posee una interfaz bien definida hacia otros componentes; algo es modular si es construido de manera tal que se facilite su ensamblaje, acomodamiento flexible y reparación de sus componentes.

***Motores físicos*:** Simulan la dinámica de los cuerpos rígidos en los juegos y simuladores actuales.

*****

O:

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

*****

P:

Propiedades físicas: Propiedades de los objetos del mundo real que se simularán con los objetos virtuales a través de la manera de ejecutar sus tareas, por ejemplo, la masa.

*****

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.

Sistema de realidad virtual (SRV): Sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real.

*****

T:

Textura: Imagen que sirve de “piel” a los modelos en un mundo virtual.

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

Timer*:* Temporizador, traducido al español.

*****

V:

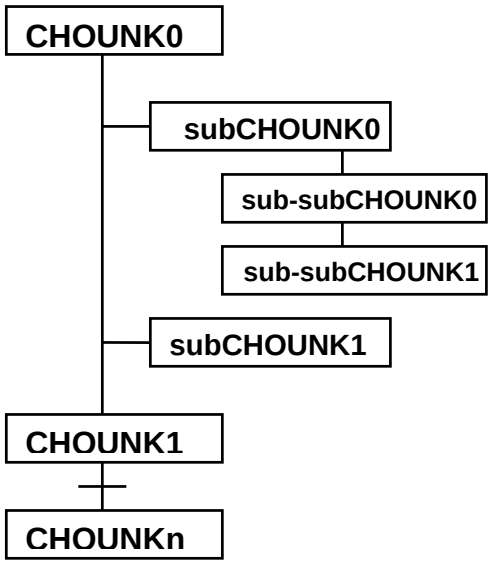
Vector: Cantidad que expresa magnitud y dirección.

Vértice: Es un punto en el espacio dado por tres coordenadas x, y, z.

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

Anexos

Anexo 1: Estructura ampliada del 3DS.



Anexo 1. 1: Estructura ampliada del 3DS.

Anexo 2: Descripción de la estructura de un fichero 3DX.

Los **Tags** contienen el identificador de objeto que puede ser cada uno de los siguientes:

GEOMETRY_TAG = 1000

MATERIALLIST_TAG = 1001

POLYLINE_TAG = 1002

LIGHT_TAG = 1003

CAMERA_TAG = 1004

HELPER_TAG = 1005

XREFOBJECT_TAG = 1006

SELECTIONSETS_TAG = 1007

Encabezamiento:

El encabezamiento tiene la siguiente estructura:

unsigned int FileID (4bytes)

unsigned int reserved1 (4bytes)

unsigned int reserved2 (4bytes)

unsigned int reserved3 (4bytes)

El campo FileID, identifica el fichero el cual debe contener el valor 0x31584433 (hexadecimal), valor que representa los 4 caracteres “3DX1”. Los restantes 3 campos se reservan para un futuro uso.

Estructura de los datos de objetos.

MaterialList

Este contiene una lista con los parámetros de todos los materiales usados por los objetos de la escena (¡solo los usados por al menos un objeto!), y también solo los de clase **StandardMaterial.**

La estructura es la siguiente:

Primero se encuentra la cantidad de materiales.

unsigned int MaterialsCount (4bytes)

…y seguidamente un record tras otro con la información de cada material: Mat0, Mat1, Mat2…

La estructura de cada Mat(n) es la siguiente:

rgb Ambient (3 bytes)

rgb Diffuse (3 bytes)

rgb Specular (3 bytes)

unsigned char Opacity (1 byte)

unsigned char Reflection (1 byte)

bool TwoSided (1 byte)

char DiffuseMap[32] (32 bytes)

char OpacityMap[32] (32 bytes)

char ReflectionMap[32] (32 bytes)

En total son 108 bytes por cada Mat(n).

El primer material (Mat0) es el **default material** este siempre estará presente en la primera posición, cada objeto de la escena que en 3DMax no tenga asignado ningún material o que el material no sea de la clase **StandardMaterial** se le será asignado el material 0, el llamado **default material**.

Geometry

Este contiene los vértices y caras de la malla, coordenadas de textura, colores de vértice, etc. Su estructura es la siguiente:

char Name[32] (32 bytes)

Word Handle (4 Bytes) (New) (ID único) **(unsigned)**

char ParentName[32] (32 bytes)

Word ParentHandle (4 bytes) (New) (ID único del padre)

int MaterialID (4 bytes)

bool TransparentMat (1 byte)

float Visibility (4 byte) (entre 0.0 **–** 0.1)

Matriz de transformación

vector3f TM_ROW1 (12 bytes)

vector3f TM_ROW2 (12 bytes)

vector3f TM_ROW3 (12 bytes)

vector3f TM_ROW4 (12 bytes)

Vértices

int NumVertex (4 bytes)

vector3f VertexList [NumVertex] (12*NumVertex Bytes)

Caras

int NumFaces (4 bytes)

vector3i FacesList[NumFaces] (12*NumFaces)

Coordenadas de textura

int NumTVerts (4 bytes)

vector3f TVertList[NumTVerts] (12*NumTVerts) //UVW coords

vector3i TFaceList[NumFaces] (12*NumFaces)

Colores de vértice

int NumCVerts (4 bytes)

rgb CVertsList[NumCVerts] (3*NumCVerts) //vertex colors

vector3i CFaces[NumFaces] (12*NumFaces)

Propiedades de usuario (texto)

int UserPropertiesBuffer_size (4 byte)

char UserPropertiesBuffer[UserPropertiesBuffer_size]

Cuando NumTVerts = 0 es que no hay definidas coordenadas de textura.

Cuando NumCVerts = 0 no hay definidos colores de vértices.

Polyline

Polyline es el line de 3DMAX con una sola curva o sea con un solo spline.

char Name[32] (32 bytes)

char ParentName[32] (32 bytes)

Vértices

int NumVertex (4 bytes)

vector3f VertexList[NumVertex] (12*NumVertex Bytes)

La estructura anterior muestra gran similitud con la que se expone más adelante del fichero .STK que resulta ventajosa, por tener una organización en bloques, bien acoplados y conectados entre sí, y en caso de ser necesario puede agregársele un nuevo miembro sin afectar su buen funcionamiento.

Anexo 3: Estructura del encabezamiento Header del fichero .STK.

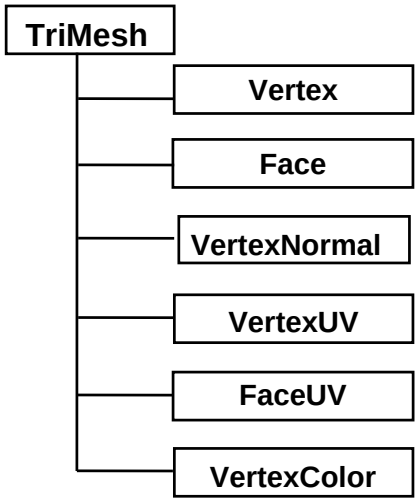
El bloque Header contiene los siguientes parámetros:

- ✓ unsigned int h_iID: identificador que sirve para saber con qué fichero se está trabajando.
- ✓ unsigned int h_iVersion: se usa para guardar la versión del formato, almacenando la misma en 4 bytes.
- ✓ unsigned int h_iHeaderSize: tamaño de la cabecera, lo cual proporciona la posibilidad de saber cuándo se termina de leer la cabecera.
- ✓ unsigned int h_iReserved1: espacio reservado para posible uso en un futuro, permitiendo insertar cualquier tipo de dato de interés.
- ✓ unsigned int h_iReserved2: nuevo espacio reservado para posible uso en un futuro, posee las mismas características que el anterior.
- ✓ unsigned int h_iTriMeshQuantity: cantidad de objetos de tipo TriMesh en la escena (mallas poligonales).
- ✓ unsigned int h_iPolyLineQuantity: cantidad de objetos tipo PolyLinea en la escena (formas).
- ✓ unsigned int h_iNodeQuantity: cantidad de nodos de una escena.
- ✓ unsigned int h_iMaterialQuantity: cantidad de materiales que se usan en la escena.
- ✓ unsigned int h_iTextureQuantity: cantidad de texturas que son usadas por los materiales.
- ✓ char h_acTextureDir: nombre de la carpeta que contiene las texturas. [5]

Anexo 4: Descripción de los subbloques TriMesh y Polyline.

Características del subbloque TriMesh.

TriMesh contiene todo referente a una malla, dígase un objeto poligonal. En la siguiente figura se muestra la estructura que presenta con sus respectivos atributos. [5]



Anexo 4. 1: Estructura ampliada del subbloque TriMesh.

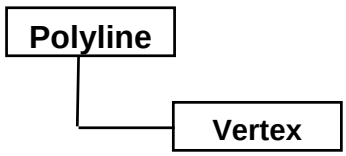
Los parámetros que contienen al subbloque TriMesh son los que siguen a continuación:

- ✓ unsigned int t_TriQuantity: cantidad de triángulos que conforman la malla.
- ✓ sTriMesh t_TriList[t_TriQuantity]: lista de estructuras triángulos, cada estructura conformada por tres índices de los vértices correspondiente al triángulo.
- ✓ sVertex 3*t_VertexNormalList[t_TriQuantity]: lista de estructuras de normales de vértices dada por las caras.
- ✓ bool t_ListExist: byte de información que se usa para saber qué atributos contiene la malla; un bit para cada existencia del atributo, los atributos son los que siguen: t_VertexColorList, t_TriColorList, t_UVList0, t_UVList1, t_UVList2 y t_UVList3.
- ✓ unsigned int t_VertexTQuantity0: cantidad de vértices con textura en el canal de mapeo1.
- ✓ unsigned int t_VertexTQuantity1: cantidad de vértices con textura en el canal de mapeo 2.
- ✓ …en dependencia de la cantidad de canales de mapeo que se tengan.
- ✓ unsigned int t_VertexCQuantity: cantidad de vértices con color.
- ✓ sVertex t_VertexUVList0[t_VertexQuantity]: lista de coordenadas de textura referente a los valores u y v para el mapa difuso.
- ✓ sTriMesh t_TriUVList0[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.

- ✓ sVertex t_VertexUVList1[t_VertexTQuantity]: memoria reservada a futuro uso.
- ✓ sTriMesh t_TriUVList1[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- ✓ sVertex t_VertexUVList2[t_VertexTQuantity]: lista de coordenadas de textura para el mapa de luces.
- ✓ sTriMesh t_TriUVList2[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- ✓ sVertex t_VertexUVList3[t_VertexTQuantity]: memoria reservada para futuro uso.
- ✓ sTriMesh t_TriUVList3[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- ✓ sVertexC t_VertexColorList[t_VertexCQuantity]: lista de colores de vértices, dados en la combinación de los tres colores que usa OpenGL para pintar r(rojo), g(verde), b(azul).
- ✓ sTriMesh t_TriColorList[t_TriCQuantity]: lista de estructuras colores triángulos, cada estructura está conformada por tres índices de los vértices con color correspondientes al triángulo. [5]

Características del subbloque Polyline.

El bloque Polyline contiene todo lo referente a una polilínea en el 3d Studio Max, es decir una línea, conformada por un conjunto de puntos, pudiendo ser abierta o cerrada. En la siguiente figura se muestra la estructura del subbloque Polyline. [5]



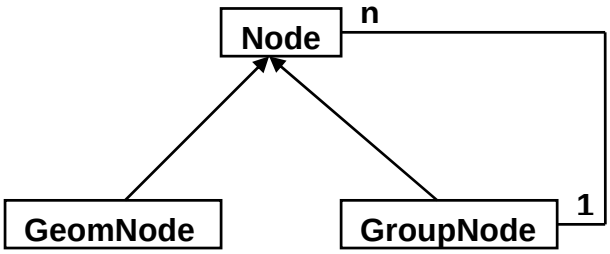
Anexo 4. 2: Estructura del subbloque Polyline.

Los parámetros que contienen este subbloque son los que siguen:

- ✓ bool p_Close: define si es una polilínea cerrada, en caso de no serlo es continua.
- ✓ sVertex p_VertexList [p_VertexQuantity]: arreglo de vértices.

Anexo 5: Representación de los nodos en la escena.

A continuación se muestra una estructura más detallada de la representación de los nodos en la escena.

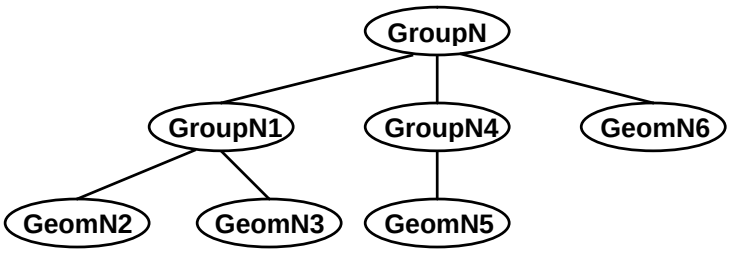


Anexo 5. 1: Estructura del subbloque GroupNode.

¿Cómo se exporta la lista de hijos?

Primeramente se construye el árbol de nodos donde se crea un padre jerárquico que es la cabeza de todos los nodos de la escena, una vez construido se va haciendo un recorrido en preorden y guardando en una lista para luego exportar la misma.

En la próxima figura se muestra un ejemplo de cómo sería la estructura del árbol.



Anexo 5. 2: Estructura del árbol de nodos.

Nota: El orden en que se exportaría está dado por el número que presenta en la figura anterior. [5]

Anexo 6: Características del subbloque MaterialProp.

El subbloque MaterialProp contiene todas las propiedades de los materiales en una escena. Los parámetros que contienen son los que se exponen a continuación.

- ✓ char mp_Name[32]: nombre del material.

- ✓ bool mp_TwoSide: en caso de ser 1 está activado el modo de dos lados, en caso de ser 0 lo contrario.
- ✓ int mp_AmbientColor[2]: color ambiental dado en rgb.
- ✓ int mp_DiffuseColor[2]: color difuso dado en rgb.
- ✓ int mp_SpecularColor[2]: color especular dado en rgb.
- ✓ float mp_Opacity: nivel de opacidad.
- ✓ int mp_IndexAmbient: índice de la textura que se le pasó por el canal ambiente, en caso de no tener asignada ninguna toma valor -1.
- ✓ …lo mismo sucede con el IndexDiffuso, IndexOpacity, IndexSpecular.
- ✓ int mp_IndexMaterial1: reservado.
- ✓ int mp_IndexLighMap: índice de la textura que se le pasó por el canal de mapa de luces, en caso de no tener asignada ninguna toma valor -1.
- ✓ int mp_IndexMaterial2: reservado. [5]

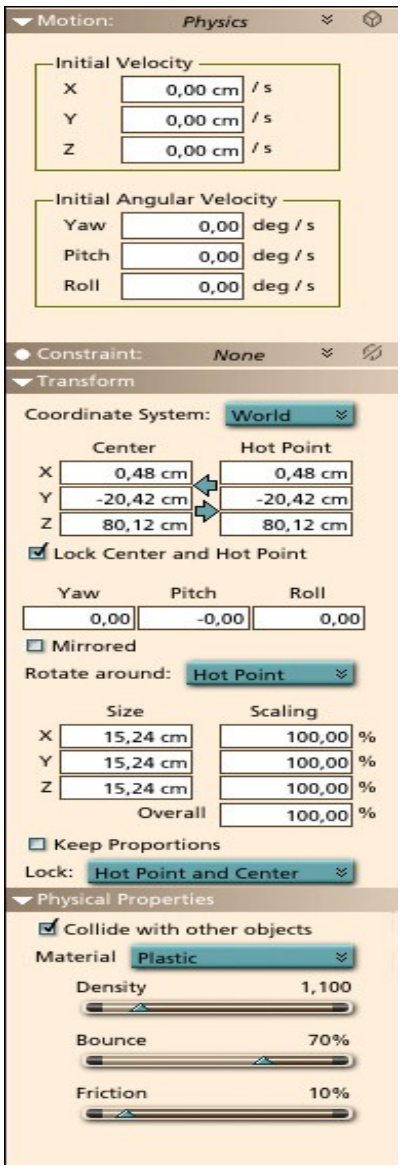
Características del subbloque Textura.

El subbloque Textura contiene información referente a las texturas que se usan en una escena, las texturas en este subbloque no son más que los ficheros que representan una imagen dígase un bmp, un tga u otro. Este subbloque contiene el nombre de la textura con su extensión.

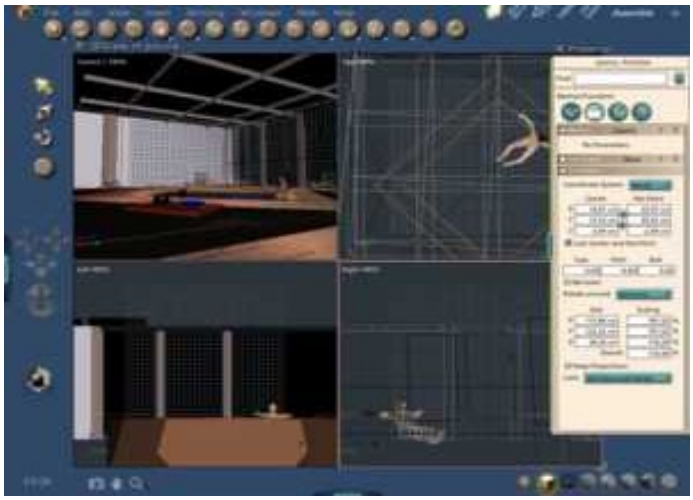
Los parámetros que contiene son los siguientes.

- ✓ char te_Name [32][m_TextureQuantity]: lista de caminos y nombres de texturas. [5]

Anexo7: Ejemplos de Interfaces de algunos Editores de Escenas 3D en el Mundo.

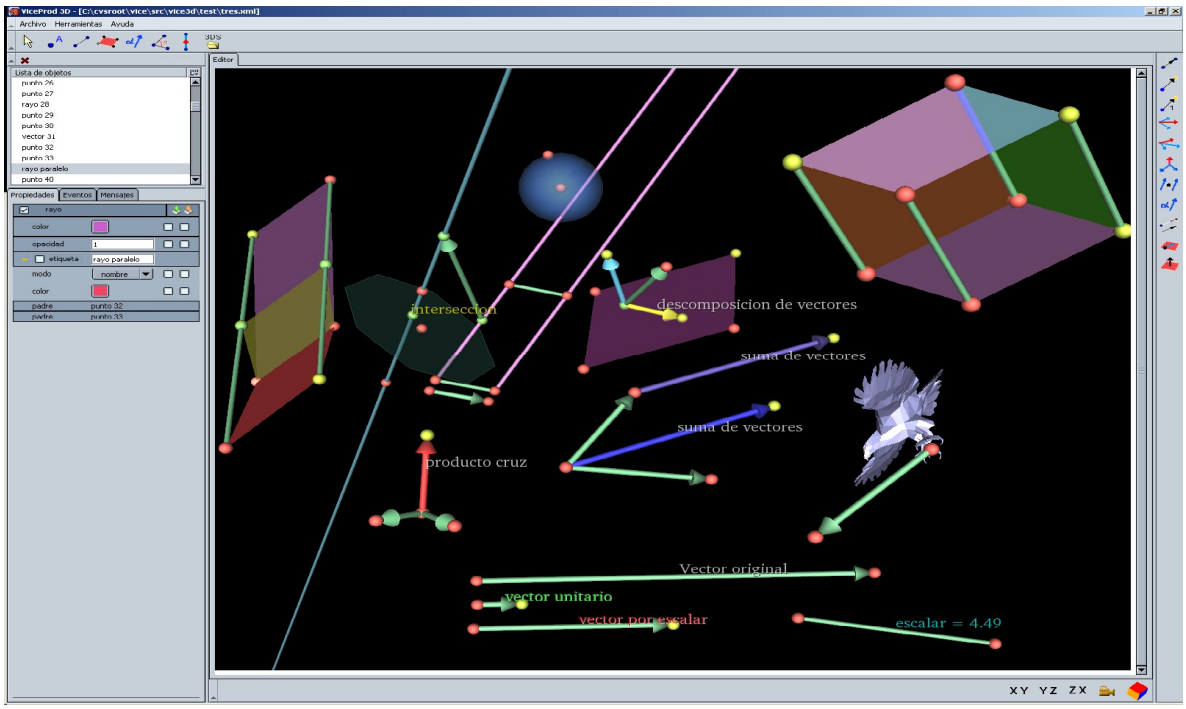


Editando Propiedades Físicas

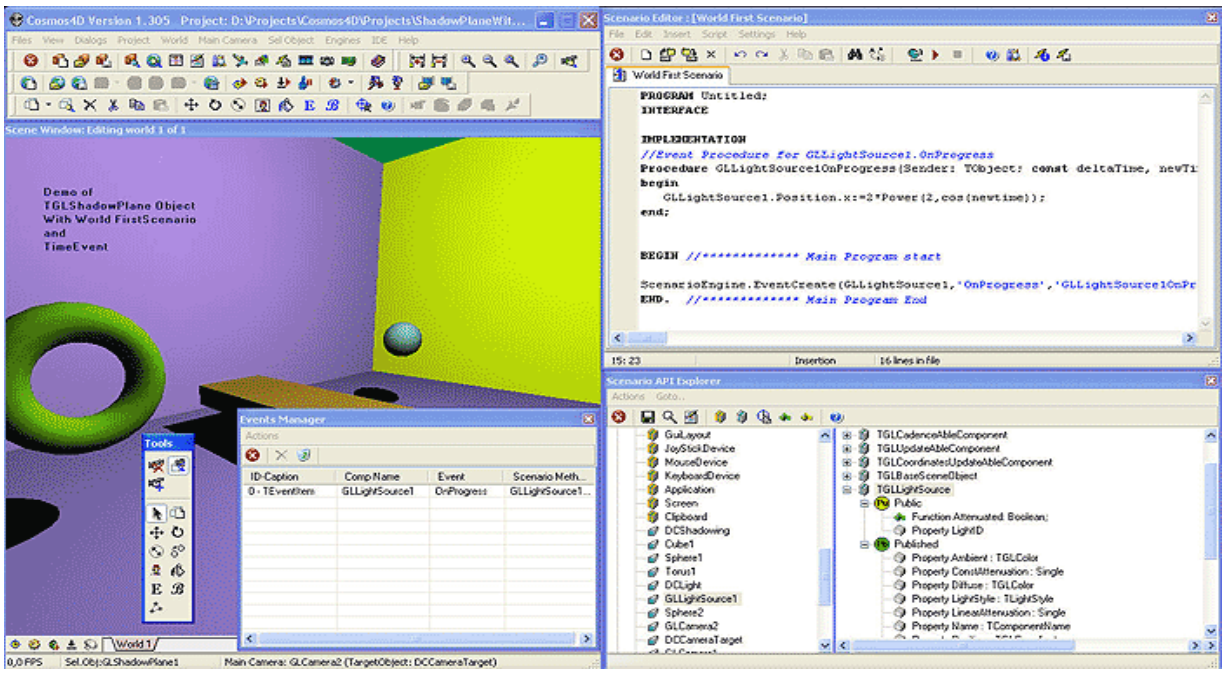


Interfaz Gráfica

Anexo 7. 1: Ejemplos de la interfaz de Carrara Studio 2.



Anexo 7. 2: Interfaz de un Editor de Escenas 3D.



Anexo 7. 3: Interfaz del Editor Cosmo 4D.

Anexo8: Descripción de las Clases del Diseño del Sistema.

Anexo 8. 1: Descripción de la Clase de Diseño CEditor.

Nombre: CEditor	
Tipo de clase: Controladora	
Atributo:	Tipo:
m_pkMenu1	CMenu*
m_pkMenu2	CMenu*
m_pkMenu3	CMenu*
m_pkBackGround	CPanel*
m_pkPanelCR	CPanel*
m_pkPanelMS	CPanel*
m_pkPanelCC	CPanel*
m_pkButtonVCR	CButton*
m_pkButtonECR	CButton*
m_pkButtonVMS	CButton*
m_pkButtonEMS	CButton*
m_pkButtonE	CButton*
m_pkButtonC	CButton*
m_pkButtonB	CButton*
m_pkLabelCR	CLabel*
m_pkLabelM	CLabel*
m_pkLabelVL	CLabel*
m_pkLabelLX	CLabel*
m_pkLabelLY	CLabel*
m_pkLabelLZ	CLabel*
m_pkLabelVA	CLabel*
m_pkLabelAX	CLabel*
m_pkLabelAY	CLabel*
m_pkLabelAZ	CLabel*
m_pkLabelP	CLabel*

m_pkLabelPX	CLabel*
m_pkLabelPY	CLabel*
m_pkLabelPZ	CLabel*
m_pkEditM	CEdit*
m_pkEditLX	CEdit*
m_pkEditLY	CEdit*
m_pkEditLZ	CEdit*
m_pkEditAX	CEdit*
m_pkEditAY	CEdit*
m_pkEditAZ	CEdit*
m_pkEditPX	CEdit*
m_pkEditPY	CEdit*
m_pkEditPZ	CEdit*
m_pkLabelES	CLabel*
m_pkLabelRE	CLabel*
m_pkLabelC	CLabel*
m_pkLabelRC	CLabel*
m_pkLabelLC	CLabel*
m_pkLabelB	CLabel*
m_pkLabelAB	CLabel*
m_pkLabelLB	CLabel*
m_pkLabelHB	CLabel*
m_pkEditRE	CEdit*
m_pkEditRC	CEdit*
m_pkEditLC	CEdit*
m_pkEditAB;	CEdit*
m_pkEditLB;	CEdit*
m_pkEditHB;	CEdit*
TVisualComponentsList	vector<CVisualComponent*>
pkGeometry	CGeometryNode *
m_bAccessed	bool
m_bFull	bool

m_bRead	bool
m_bInterface	bool
m_bNoRead	bool
m_bClean	bool
m_cSpace, m_cSkip	char
acdirection	char *
m_acCubeID	char*
pkworld	CPhysicWorld*
pkCuarto	CGroupNode*
kRBody	SRigidBody *
kheader	SHeader*
eNewEPhysicModel	EPhysicModel
eVolumeGeometry	EShapeVolumeTypes
kPGSphere	SGeometrySphere *
kPGCylinder	SGeometryCylinder*
kPGBox	SGeometryBox*

Para cada responsabilidad:	
Nombre:	OnCreate ()
Descripción:	Es donde se inicializan los valores de la aplicación.
Nombre:	OnLoadFile()
Descripción:	Es donde se cargan los ficheros, y donde se inicializa el *timer* de la aplicación en 0.
Nombre:	OnIdle ()
Descripción:	Es una llamada cíclica correspondiente al ciclo de simulación, que puede ser interrumpido únicamente ante los eventos del Sistema Operativo.
Nombre:	void LoadInterface ()
Descripción:	Se crea la interfaz general del Editor. (paneles, menues, labels, correspondientes a la interfaz general).
Nombre:	void LoadInterfaceCR()
Descripción:	Se crea la interfaz del modelo para cuerpos rígidos.(botones, edits, labels, correspondientes a este modelo).

Nombre:	void LoadInterfaceMS()
Descripción:	Se crea la interfaz del modelo para cuerpos deformables.(botones, edits, labels, correspondientes a este modelo).
Nombre:	void APIWindowsOpenDialog()
Descripción:	Es donde el usuario a través de un cuadro de dialogo selecciona el entorno y los objetos a cargar, almacenándose entonces en variables la dirección de los mismos.
Nombre:	void AssigValues ()
Descripción:	A través de este método se asignan los valores entrados por el usuario a las estructuras definidas.
Nombre:	void LookState()
Descripción:	Este método permite la visualización de los cambios realizados en el comportamiento físico de los objetos.
Nombre:	void DeleteComponents()
Descripción:	Es donde se eliminan los componentes visuales del modelo que ya no se este utilizando.
Nombre:	void SaveFile()
Descripción:	En este método es donde se crea el fichero y donde se guardan todas las propiedades físicas del objeto en el fichero creado.
Nombre:	void UpdatePhysic()
Descripción:	Es donde se chequean las colisiones existentes en la aplicación.
Nombre:	void InitializeWorld()
Descripción	Es donde se inicializan todas las propiedades del mundo físico que va a tener la aplicación y que va a utilizar la ODE.

Anexo 8. 2: Descripción de la Clase de Diseño SGeometrySphere

Nombre: SGeometrySphere	
Tipo de clase: Entidad	
Atributo:	Tipo:
fradio	float
m_EPhysicModel	EPhysicModel
Para cada responsabilidad:	

Nombre:	SGeometrySphere()
Descripción:	Constructor sin parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con geometría esférica.
Nombre:	SGeometrySphere(float arg_fradio, EPhysicModel arg_PhysicModel)
Descripción:	Constructor con parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con geometría esférica.

Anexo 8. 3: Descripción de la Clase de Diseño SGeometryCylinder.

Nombre: SGeometryCylinder	
Tipo de clase: Entidad	
Atributo:	Tipo:
fradio	float
flengh	float
m_EPhysicModel	EPhysicModel
Para cada responsabilidad:	
Nombre:	SGeometryCylinder()
Descripción:	Constructor sin parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con geometría cilíndrica.
Nombre:	SGeometryCylinder(float arg_fradio, float arg_flengh, EPhysicModel arg_PhysicModel)
Descripción:	Constructor con parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con geometría cilíndrica.

Anexo 8. 4: Descripción de la Clase de Diseño SGeometryBox.

Nombre: SGeometryBox	
Tipo de clase: Entidad	
Atributo:	Tipo:
fwidth	float

flenght	float
fhigh	float
m_EPhysicModel	EPhysicModel
Para cada responsabilidad:	
Nombre:	SGeometryBox()
Descripción:	Constructor sin parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con geometría cúbica.
Nombre:	SGeometryBox(float arg_fwidth, float arg_flenght, float arg_fhigh, EPhysicModel arg_PhysicModel)
Descripción:	Constructor con parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con geometría cúbica.

Anexo 8. 5: Descripción de la CLase de Diseño SDeformBody.

Nombre: SDeformBody	
Tipo de clase: Entidad	
Atributo:	Tipo:
fmass	float
felastConst	float
famort	float
Para cada responsabilidad:	
Nombre:	SDeformBody()
Descripción:	Constructor sin parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con modelo deformable.
Nombre:	SDeformBody(float arg_fmass, float arg_felastConst, float arg_famort)
Descripción:	Constructor con parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con modelo deformable.

Anexo 8. 6: Descripción de la Clase de Diseño SRigidBody.

Nombre: SRigidBody	
Tipo de clase: Entidad	
Atributo:	Tipo:
fmass	float
kVelocL	CVector3
kVelocA	CVector3
kPossition	CVector3
Para cada responsabilidad:	
Nombre:	SRigidBody()
Descripción:	Constructor sin parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con modelo rígido.
Nombre:	SRigidBody (float arg_fmass, CVector3 arg_kVelocL, CVector3 arg_kVelocA, CVector3 arg_kPossition).
Descripción:	Constructor con parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan las características de los cuerpos con modelo rígido.

Anexo 8. 7: Descripción de la Clase de Diseño SHeader.

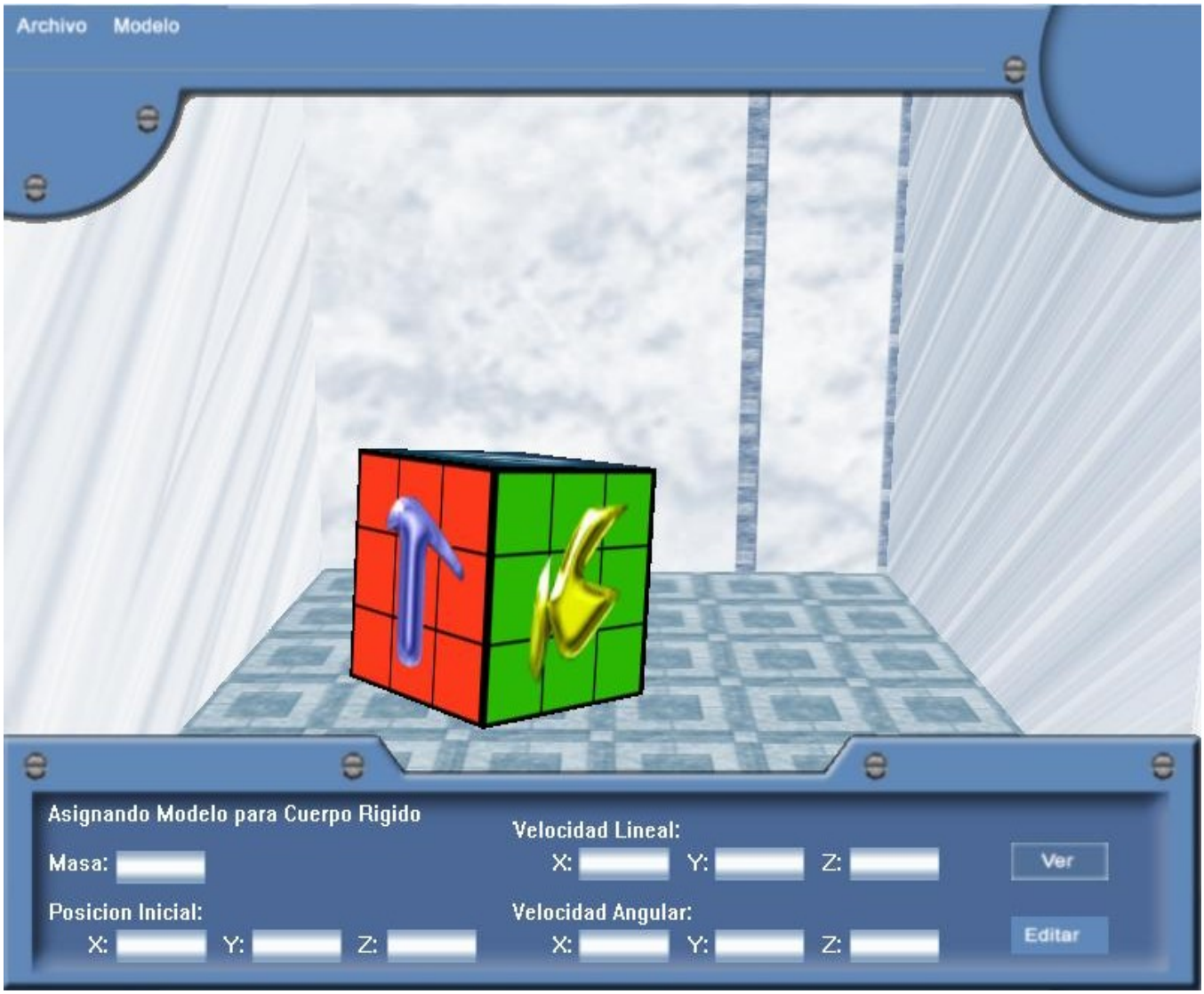
Nombre: SHeader	
Tipo de clase: Entidad	
Atributo:	Tipo:
version	float
eVolumeGeometry	EShapeVolumeTypes
uiReservedSpace	unsigned int
Para cada responsabilidad:	
Nombre:	SHeader()
Descripción:	Constructor sin parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan los datos de la cabecera del fichero físico que se crea.
Nombre:	SHeader(EShapeVolumeTypes arg_eVolumeGeometry)

Descripción:	Constructor con parámetros de la clase se encarga de inicializar el objeto de la misma donde se manipulan los datos de la cabecera del fichero físico que se crea.

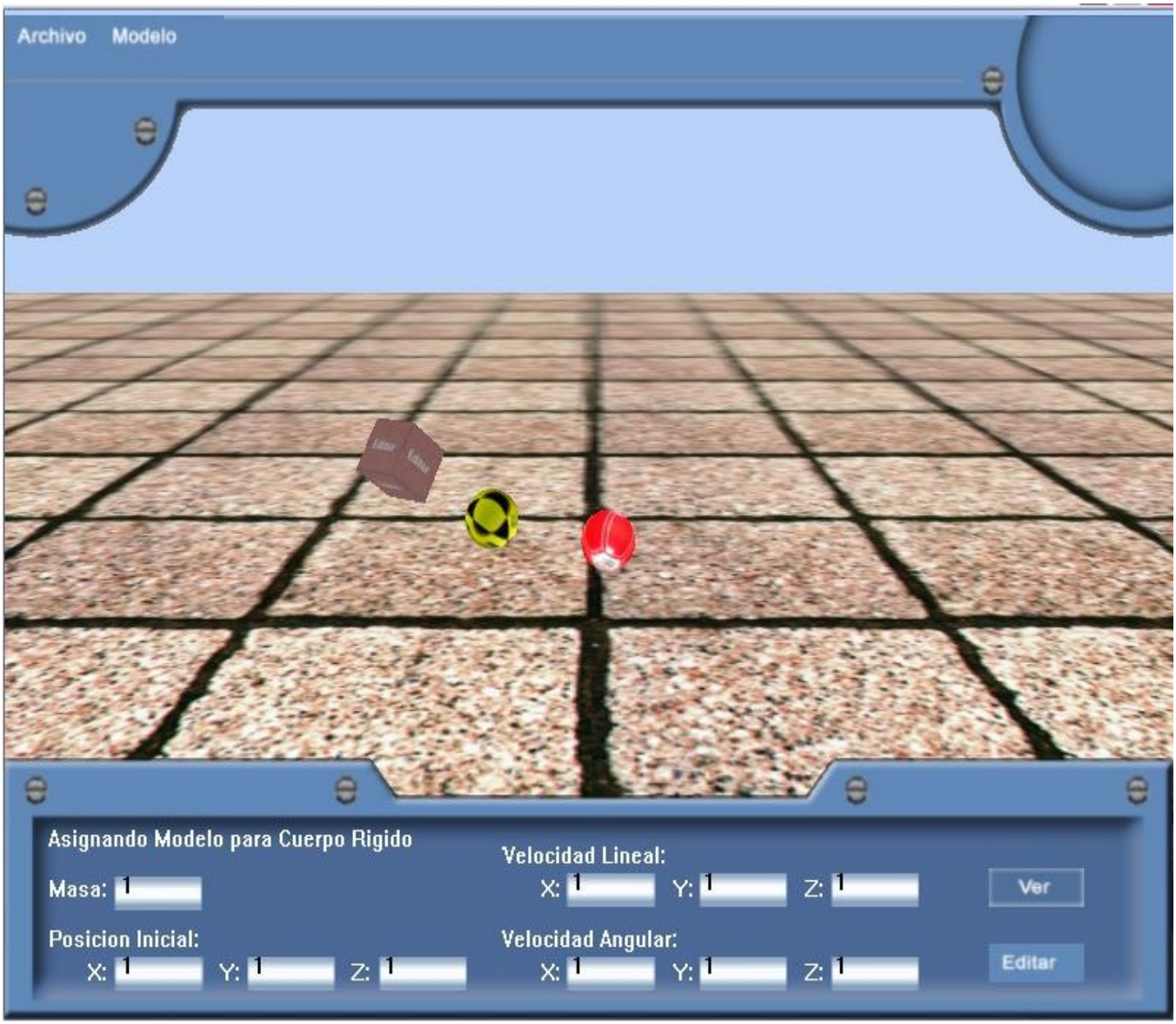
Anexo 8. 8: Descripción de la Clase de Diseño EPhysicModel

Nombre: EPhysicModel	
Tipo de clase: Entidad	
Atributo:	Tipo:
PM_RIGID,	enumerativos
PM_MASS_SPRING	enumerativos
Para cada responsabilidad:	
Descripción:	Contiene dos atributos enumerativos que definen las estructuras de los modelos físicos de los cuerpos rígidos y deformables.

Anexo9: Imágenes de la Interfaz del **Editor de Propiedades Físicas** desarrollado.



Anexo 9. 1: Interfaz del Editor con un cuerpo en la escena.



Anexo 9. 2: Interfaz del Editor asignando los valores de un Modelo Físico y Mostrando el Comportamiento.

Índice de Figuras

FIG 1. 1: ESTRUCTURA DEL 3DS. 10
FIG 1. 2: SISTEMA DE COORDENADAS 3DSMAX. 11
FIG 1. 3: ESTRUCTURA DE UN FICHERO 3DX. 12
FIG 1. 4: ESTRUCTURA DEL FICHERO STK. 13
FIG 1. 5: ESTRUCTURA DEL BLOQUE GEOMETRY. 14
FIG 1. 6: ESTRUCTURA DEL BLOQUE NODE 15
FIG 1. 7: ESTRUCTURA DEL BLOQUE MATERIAL 15
FIG 1. 8: SISTEMA MASA-RESORTE 23
FIG 1. 9: DISCRETIZACIÓN DE UN DOMINIO GLOBAL. 25
FIG 1. 10: REPRESENTACIÓN DE UN SISTEMA POR DOS MALLADOS DIFERENTES HACIENDO USO DE FEM. 26
FIG 1. 11: NOTACIÓN DE BEM Y SUS CONDICIONES DE FRONTERA. 28
FIG 2. 1: ESTRUCTURA DEL FICHERO FÍSICO .PHS. 37
FIG 3. 1: MODELO DE DOMINIO DE LA APLICACIÓN. 43
FIG 3. 2: EMPAQUETAMIENTO DEL LOS CASOS USOS Y ACTORES DEL SISTEMA. 47
FIG 3. 3: DIAGRAMA DE CU DEL SISTEMA. 49
FIG 4. 1: DISEÑO LÓGICO DEL SISTEMA. 57
FIG 4. 2 CLASES DEL PAQUETE APPLICATION Y LOGIC. 58
FIG 4. 3: DIAGRAMA DE PAQUETES GENERAL DEL SISTEMA. 59
FIG 4. 4: DIAGRAMA DE CLASES DE DISEÑO GENERAL. 60
FIG 4. 5: DIAGRAMA DE SECUENCIA CU SIMULAR COMPORTAMIENTO FISICO, SECCIÓN MODELO RIGIDO. 61
FIG 4. 6: DIAGRAMA DE SECUANCIA CU SIMULAR COMPORTAMIENTO FISICO, SECCIÓN MODELO DEFORMABLE. 62
FIG 4. 7: DIAGRAMA DE SECUENCIA CU CARGAR FICHERO. 63
FIG 4. 8: DIAGRAMA DE SECUENCIA CU EXPORTAR FICHERO. 64
FIG 4. 9: DIAGRAMA DE COMPONENTES GENERAL. 66
ANEXO 1. 1: ESTRUCTURA AMPLIADA DEL 3DS. 81
ANEXO 4. 1: ESTRUCTURA AMPLIADA DEL SUBBLOQUE TRIMESH. 86
ANEXO 4. 2: ESTRUCTURA DEL SUBBLOQUE POLYLINE. 87
ANEXO 5. 1: ESTRUCTURA DEL SUBBLOQUE GROUPNODE. 88
ANEXO 5. 2: ESTRUCTURA DEL ÁRBOL DE NODOS. 88
ANEXO 9. 1: INTERFAZ DEL EDITOR CON UN CUERPO EN LA ESCENA. 100
ANEXO 9. 2: INTERFAZ DEL EDITOR ASIGNANDO LOS VALORES DE UN MODELO FISICO Y MOSTRANDO EL COMPORTAMIENTO. 101

Índice de Tablas

TABLA 3. 1: ACTOR DEL SISTEMA. 47
TABLA 3. 2: CU1 CARGAR FICHERO. 47
TABLA 3. 3: CU2 SIMULARCOMPORTAMIENTO FISICO. 48
TABLA 3. 4: CU3 EXPORTAR FICHERO FISICO. 48
TABLA 3. 5: DESCRIPCIÓN CU CARGAR FICHERO. 49
TABLA 3. 6: DESCRIPCIÓN CU SIMULAR COMPORTAMIENTO FISICO. 51
TABLA 3. 7: DESCRIPCIÓN DE LA SECCIÓN MODELO RÍGIDO DEL CU SIMULAR COMPORTAMIENTO FISICO. 52
TABLA 3. 8: DESCRIPCIÓN DE LA SECCIÓN MODELO DEFORMABLE DEL CU SIMULAR COMPORTAMIENTO FÍSICO. 53
TABLA 3. 9: DESCRIPCIÓN DEL CU EXPORTAR FICHERO. 54
ANEXO 8. 1: DESCRIPCIÓN DE LA CLASE DE DISEÑO CEDITOR. 92
ANEXO 8. 2: DESCRIPCIÓN DE LA CLASE DE DISEÑO SGEOMETRYSPHERE 95
ANEXO 8. 3: DESCRIPCIÓN DE LA CLASE DE DISEÑO SGEOMETRYCYLINDER. 96
ANEXO 8. 4: DESCRIPCIÓN DE LA CLASE DE DISEÑO SGEOMETRYBOX. 96
ANEXO 8. 5: DESCRIPCIÓN DE LA CLASE DE DISEÑO SDEFORMBODY. 97
ANEXO 8. 6: DESCRIPCIÓN DE LA CLASE DE DISEÑO SRIGIDBODY. 98
ANEXO 8. 7: DESCRIPCIÓN DE LA CLASE DE DISEÑO SHEADER. 98
ANEXO 8. 8: DESCRIPCIÓN DE LA CLASE DE DISEÑO EPHYSICMODEL 99