UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

Formato de Fichero para creación de escenas virtuales para la STK

Trabajo para optar por el Título de Ingeniero en Ciencias Informáticas

Autor: Jorge Reinaldo Rodríguez

Tutor: MSc. Yanoski Camacho Román
Co-Tutores: Ing. Laura Rodríguez García
Ing. Alfredo Quintana López

La Habana
2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas; para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de junio del año 2012.

Jorge Reinaldo Rodríguez
Autor

MSc. Yanoski Camacho Román
Tutor

Ing. Laura Rodríguez García
Co- Tutor

Ing. Alfredo Quintana López
Co-Tutor

A mi madre, por estar siempre a mi lado y mi abuela Nene.

A mi papá que no pudo verme.

A mis tíos, mis hermanos y mi otro papi.

A toda la familia y amigos que siempre confiaron en mí.

Resumen

Con el surgimiento de la realidad virtual, se han creado herramientas que permiten crear y visualizar escenas virtuales para el desarrollo de videojuegos. Para lograr mayor rendimiento y flexibilidad, estas herramientas poseen opciones que permiten crear escenas virtuales mediante la carga de ficheros con información de un objeto tridimensional o de un mundo virtual complejo. Cada una de estas herramientas posee formatos de ficheros propios de acuerdo a sus características, logrando mejorar el redimiendo durante la carga y reduciendo el tiempo en la creación de escenas virtuales.

En la Universidad de las Ciencias Informáticas (UCI), se cuenta con una herramienta propia denominada SceneToolKit (STK) la cual permite el desarrollo de Sistemas de Realidad Virtual (SRV). La versión actual de esta herramienta (v 10.07) posee un conjunto de características que permiten a los usuarios flexibilidad y rendimiento en la creación de escenas virtuales; debido a que el formato de fichero con el que trabaja no está estructurado de forma tal que permita aprovechar las funcionalidades de la herramienta, la creación de escenas virtuales se realiza mediante un complejo proceso. El objetivo de este trabajo es crear un formato de fichero propio que permita usar eficientemente las características de la versión actual de la STK.

Como resultado se obtuvo un formato de fichero estructurado de forma tal que se crean las estructuras de datos necesarias en la versión actual de la STK además de la incorporación al 3ds Max de una funcionalidad que permite exportar objetos tridimensionales o escenas virtuales hacia el fichero antes mencionado.

Palabras claves:
Estructurado, Exportar, Fichero, Formato, Motores ó Herramientas gráficas

Índice

INTRODUCCIÓN. 1

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA. 5

1.1 Conceptos básicos 6
1.2 Scene ToolKit (STK) versión 10.07 7
1.2.1 Estructura de los nodos STK (v07.07) 8
1.2.2 Estructura de los nodos STK (v10.07) 8
1.3 Autodesk 3ds Max 9
1.4 MaxScript y SDK (C++) 9
1.5 MaxScript o SDK? 10
1.6 Biblioteca 3DXI (IGame.h) 10
1.7 Ficheros para la creación de escenas virtuales 11
1.8 Fichero con formato ASCII 12
1.8.1 STL (Standard Tessellation Language) 12
1.8.2 ASE (ASCII SceneExporter) 13
1.8.3 DIRECTX 14
1.9 Fichero con formato binario 16
1.9.1 STL (Standard Tessellation Language) 16
1.9.2 MD2 17
1.9.3 MD3 18
1.9.4 3DS 19
1.10 API gráfica OpenGL 20
1.11 API gráfica Direct3D 21

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA 23

2.1 Características del fichero propuesto 24
2.2 Bloque Cabecera 25
2.3 Bloque RenderObjects 26
2.3.1 Sub-Bloque Luz 27
2.3.2 Sub-Bloque Cámara 28
2.3.3 Sub-Bloque Material 28
2.3.4 Sub-Bloque Mesh 30
2.3.5 Sub-Bloque Spline 30
2.4 Bloque Nodos 31
2.5 Bloque TextureMap 32
2.6 Aspectos generales 32
2.7 Reglas del negocio 33
2.8 Modelo de dominio 34
2.9 Glosario de términos del modelo de dominio 34

2.10 Captura de requisitos 34
2.10.1 Requisitos funcionales *35*
2.10.2 Requisitos no funcionales *35*
2.11 Modelo caso uso del sistema 36
2.11.1 Actor del sistema *36*
2.11.2 Casos de uso del sistema *37*
2.11.3 Diagrama caso de uso del sistema *37*
2.11.4 Descripción de los CU en formato expandido *38*

CAPÍTULO 3 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA 43

3.1 Diagrama de clases del diseño 44
3.2 Descripción de las clases del diseño 49
3.3 Diagramas de secuencia 61
3.4 Implementación del sistema 61

CONCLUSIONES 64

RECOMENDACIONES 65

REFERENCIAS BIBLIOGRÁFICAS 66

GLOSARIO DE TÉRMINOS. 68

ANEXOS 1 70

ANEXOS 2 71

Índice de figuras

Fig. 1 Estructura nodo STK (v10.07) 8
Fig. 2 Estructura del fichero DIRECTX 15
Fig. 3 Estructura del fichero MD2 17
Fig. 4 Estructura del fichero MD3 18
Fig. 5 Estructura del fichero 3DS 19
Fig. 6 Estructura del fichero .STK 25
Fig. 7 Estructura bloque RenderObjects 27
Fig. 8 Modelo de Dominio 34
Fig. 9 Diagrama CU del sistema 37
Fig. 10 Diagrama de paquetes del diseño 44
Fig. 11 Diagrama de clases del paquete 3ds Max SDK 45
Fig. 12 Diagrama de clases del paquete Biblioteca 3DXI 46
Fig. 13 Diagrama de clases del paquete STK_Exporter 47
Fig. 14 Diagrama de clases del paquete Formas 48
Fig. 15 Diagrama de clases del paquete Atributos 49
Fig. 22 Diagrama de despliegue 61
Fig. 23 Diagrama de compontes 62

Índice de tablas

Tabla 1 Descripción del actor del sistema 36
Tabla 2 Expansión CU "Exportar fichero" 39
Tabla 3 Expansión CU "Crear RenderObjects" 40
Tabla 4 Expansión CU "Crear STK_Nodos" 41
Tabla 5 Descripción de la clase SceneExport 51
Tabla 6 Descripción de la clase STK_Exporter 53
Tabla 7 Descripción de la clase RenderObject 54
Tabla 8 Descripcion de la clase STK_TextureMap 54
Tabla 9 Descripcion de la clase STK_Formas 55
Tabla 10 Descripcion de la clase STK_Spline 55
Tabla 11 Descripción de la clase STK_Mesh 56
Tabla 12 Descripción de la clase STK_Atributos 56
Tabla 13 Descripción de la clase STK_Material 57
Tabla 14 Descripción de la clase STK_Luz 58
Tabla 15 Descripcion de la clase STK_Camara 59
Tabla 16 Descripcion de la clase STK_Nodo 59
Tabla 17 Descripción de la clases STK_Arbol 60
Tabla 18 Descripción de la clases STKGroup 61

Introducción.

En la Universidad de las Ciencias Informáticas (UCI) se desarrollan Sistemas de Realidad Virtual (SRV); para ello se ha creado el departamento de Visualización y Realidad Virtual que pertenece al Centro de Informática Industrial (CEDIN). En este departamento existe una herramienta gráfica propia denominada SceneToolKit (STK) que facilita el desarrollo de SRV.

El proceso de creación de una escena virtual con la STK puede ser de dos maneras: manual o mediante la carga de un fichero con información de un objeto tridimensional o de una escena. Crear una escena manualmente es un proceso engorroso, debido a que se deben crear y posicionar los objetos en la escena uno por uno, generándose un código extremadamente largo y complejo; además, no se obtienen los mismos resultados que con una herramienta de modelado y animación 3d, debido a que estas últimas ofrecen facilidades para este tipo de trabajo. Por este motivo la mayoría de los motores o herramientas gráficas, brindan funcionalidades para la carga de ficheros con información exportada de las herramientas de modelado y animación.

El proceso de creación de una escena virtual con la STK mediante la carga de un fichero consta de cuatro fases:

1. Diseñar la escena en la herramienta de modelado y animación 3ds Max.
2. Exportar la escena al fichero 3DX.
3. Cargar el fichero que contiene la escena en la STK.
4. Visualizar la escena cargada.

El fichero que usa la STK para la creación de escenas virtuales se denomina 3DX. Este fichero tiene tres deficiencias, la primera es la repetición de vértices, la cual trae consigo que al cargar el fichero sea necesario hacer búsquedas de los vértices que se repiten para eliminarlos, esto causa que la fase tres en el proceso de creación de escenas virtuales mediante la carga de un fichero en la STK sea compleja. Una

segunda deficiencia es la falta de información necesaria para la creación de escenas virtuales complejas y realistas, esto es debido a que no soporta multitexturas, cámaras y luces. Además la información no está estructurada de forma tal que durante la carga se creen las estructuras de datos existentes en la versión 10.07 de la STK; necesarias para optimizar la visualización de las escenas virtuales.

Como se puede observar las deficiencias en el proceso de creación de escenas virtuales en la STK mediante la carga de un fichero, derivan de las limitaciones que presenta el fichero 3DX. Para dar solución a esta situación se ha pensado modificar la estructura del fichero 3DX pero no se ha podido debido a que no se cuenta con el código fuente ni documentación relacionada con el fichero.

Un intento para mejorar el proceso de creación de escenas virtuales en la STK mediante la carga de un fichero fue un trabajo realizado en el 2007 por los autores Ing. Wendy García López e Ing. Alexis Echemendía González denominado “Sistema de generación de ficheros para entornos virtuales”, el cual presenta una solución para el desarrollo de una funcionalidad que permita exportar escenas desde la herramienta 3ds Max, para ser cargadas en la STK versión 07.07. La solución propuesta por ese trabajo de diploma no es aplicable a la nueva versión de la STK (v 10.07) debido a que la estructura de los nodos del grafo de escena es diferente; en el capítulo uno se aborda las diferencias de la estructura de los nodos de ambas versiones de la STK detalladamente. Tampoco permite crear escenas virtuales complejas y realistas ya que solo presenta información de geometrías, *splines* y materiales que no permiten multitexturas.

Debido a los problemas que presentan las dos soluciones anteriores surge la siguiente pregunta: ¿cómo mejorar el proceso de creación de escenas virtuales en la STK? La cual sería el **problema científico**.

Para dar solución a este problema es necesario desarrollar un formato de fichero que permita eliminar las deficiencias del fichero 3DX y crear las estructuras de datos en la versión 10.07 de la STK. Por tal motivo el presente trabajo de diploma tiene como **objetivo** desarrollar un formato de fichero que permita mejorar el proceso de creación de escenas virtuales en la STK.

El **objeto de estudio** del presente trabajo es el proceso de creación de escenas virtuales y como **campo de acción** el proceso de creación de escenas virtuales mediante la carga de un fichero.

Las tareas que se mencionan a continuación son la guía para lograr el objetivo de este trabajo:

1. Diseño de un formato de fichero que contenga todos los elementos necesarios en las escenas de la STK: multitexturas, luces y cámaras; que corrija la repetición de vértices y que la información esté estructurada acorde a las estructuras de datos de la versión 10.07 de la STK.
   1.1. Descripción de los elementos necesarios en las escenas de la STK.
   1.2. Descripción de los nodos del grafo de escena en las dos versiones de la STK.
   1.3. Caracterización de algunos formatos de ficheros para creación de entornos virtuales.
   1.4. Proponer un formato de fichero teniendo en cuenta las ventajas y desventajas de los estudiados.
2. Incorporación al 3ds Max de una opción para exportar la información de los objetos diseñados hacia el fichero desarrollado.
   2.1. Caracterización de herramientas para el desarrollo de funcionalidades para exportar escenas desde el 3ds Max.
   2.2. Descripción de la solución para exportar escenas desde el 3ds Max.
3. Implementación de la carga del fichero en la STK.

Resultado esperado:

Con el desarrollo del nuevo formato de fichero se espera resolver las tres deficiencias planteadas exportando un fichero que responda mejor a la estructura del grafo de escena y *render* genérico de la STK, lo que facilitará el proceso de creación de escenas virtuales.

A continuación se hace una breve descripción de los capítulos abordados en este trabajo.

Capítulo 1 "Fundamentación teórica". A lo largo del capítulo se caracteriza la STK, a la cual va dirigida el desarrollo del presente trabajo de diploma. Además se caracterizan las distintas formas de desarrollar funcionalidades para 3ds Max. Los formatos de ficheros para la creación de entornos virtuales más usados, así como sus ventajas y desventajas son caracterizados también. Se describen brevemente las APIs gráficas OpenGL y Direct3D.

Capítulo 2 “Solución y descripción técnica”. En este capítulo se describen las características del formato de fichero propuesto dando solución a los problemas planteados. Además se especifica el modelo del dominio, la captura de requisitos y el modelo de casos de usos.

Capítulo 3 “Diseño e implementación del sistema”. En este capítulo se describen las clases del diseño arquitectónicamente significativas y se presentan los diagramas necesarios para la implementación del sistema.

Capítulo 1 Fundamentación teórica.

En el presente capítulo se describe brevemente la herramienta gráfica SceneToolKit, así como la estructura de los nodos en las versiones 07.07 y 10.07(a esta última versión está dirigido el desarrollo de este trabajo). Además se caracteriza el software Autodesk 3ds Max y las herramientas que presenta para el desarrollo de nuevas funcionalidades. También se describen distintos formatos de ficheros para la creación de escenas virtuales, teniendo en cuenta su estructura, formato en que se salvan, ventajas y desventajas. Por último se describen dos APIs gráficas, OpenGL y Direct3D.

1.1 Conceptos básicos

A continuación se exponen definiciones necesarias para comprender con mayor facilidad el presente trabajo de diploma.

Extensión: (del inglés *Plugin*) Aplicación que se relaciona con otra para aportarle nueva funcionalidad y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúa por medio de la Interfaz Programada de la Aplicación. (1)

Escena virtual: (Entorno virtual) Es un mundo creado virtualmente que puede contener desde un único objeto 3d hasta miles de ellos. Por lo general las escenas virtuales son creadas en herramientas de diseño 3d como 3ds Max.

Fichero con información gráfica: Es un contenedor de información de un objeto 3d o una escena virtual que puede ser manipulada de forma unitaria, por lo general estos ficheros se salvan de forma binaria o ASCII (Texto plano).

Grafo de escena: es un grafo dirigido acíclico de nodos que contiene los datos que definen un escenario virtual y controlan su proceso de dibujado. En la actualidad existen varias librerías gráficas de alto nivel, y cada uno de sus grafos de escena presenta sus propias particularidades. (2)

RenderObjects: (RO) Es la definición que agrupa todos los objetos que conforman una escena virtual en la STK.

Renderización: (del inglés *Render*) Es el proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en Software de diseño en 3D. (3)

Videojuego: Según la Real Academia Española es un dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador.

1.2 Scene ToolKit (STK) versión 10.07

La SceneToolKit es una herramienta que abstrae al desarrollador del trabajo con API de bajo nivel como son OpenGL y Direct3D; de las cuales se habla en los epígrafes 1.9 y 1.10 respectivamente, facilitando de esta manera la representación de entornos virtuales y simulaciones. Cuenta con un motor de *render* y técnicas que permiten un incremento considerable del rendimiento y flexibilidad. (4)

Posee un módulo denominado “Sistema de *render* genérico” que está integrado por varios componentes que presentan un comportamiento de máquina de estado orientada a objetos. Las categorías de los componentes del módulo se exponen a continuación:

- Formas (*Shapes*): Representan los objetos geométricos del entorno visual. Ejemplos típicos pueden ser geometrías 2D y 3D así como superficies y curvas.
- Atributos (*Attributes*): Describen o modifican la apariencia de las formas en el entorno visual, en otras palabras, pueden ser cualquier tipo de atributos gráficos como: color, textura, material, transformaciones.
- Manipuladores (*Handlers*): Describen los algoritmos que interpretan los atributos y las formas. Un *handler* se encarga de interpretar una clase atributo o una clase forma en particular.
- Técnicas (*Techniques*): Describe una estrategia de dibujado para procesar una secuencia de componentes. La técnica es responsable de buscar el *handler* apropiado para un determinado *shape* o *attribute*.
- Motor (*Engine*): Controla el contexto de *render*. Mantiene una técnica activa, a la cual le es delegada la evaluación de los formas (*shapes*) y atributos(*attributes*)
- Contexto de *render* (*RenderingContext*): Mantiene una colección de atributos y *handlers* almacenados en tablas. Está compuesto por tres tablas. (4)

A continuación se hace una comparación de la estructura de los nodos en las versiones 07.07 y 10.07 de la STK, necesaria para poder comprender por qué la solución propuesta por el trabajo de diploma

"Sistema de generación de ficheros para entornos virtuales", abordado en la Introducción del presente trabajo de diploma no es aplicable a la versión actual de la STK.

1.2.1 Estructura de los nodos STK (v07.07)

En la versión 07.07 de la STK todos los objetos de la escena se creaban con un nodo propio, para una geometría se creaba un nodo llamado GeometryNode, para los materiales se creaba uno llamado MaterialNode e incluso había un nodo específico para la creación de autos.

1.2.2 Estructura de los nodos STK (v10.07)

En la versión actual de la STK que es a la que va dirigida el desarrollo del presente trabajo de diploma, se ha modificado la estructura de los nodos para dar una mayor flexibilidad y rapidez en la visualización de las escenas virtuales.

Cada nodo contiene una lista de RenderObjects, dentro de los cuales se encuentran los atributos, técnicas especializadas, manipuladores especializados y formas, además contiene una lista con sus hijos. A continuación se hace una descripción de las formas y los atributos, ya que estos son los dos tipos de RO que se pueden exportar desde una escena de una herramienta de diseño y animación. Las formas son todos aquellos objetos que pueden ser dibujados, como las geometrías y *spline*s mientras que los atributos son todos los elementos que completan una escena virtual, ejemplo de estos últimos son los materiales, las cámaras y las luces. En la siguiente figura se muestra la estructura de los nodos.

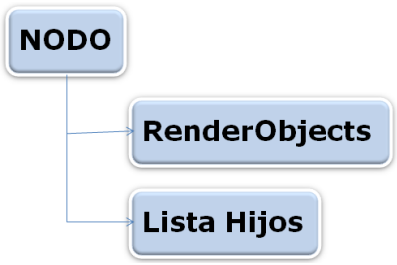

Fig. 1 Estructura nodo STK (v10.07)

1.3 Autodesk 3ds Max

El 3ds Max es uno de los programas más usados para la creación y animación de entornos virtuales. Fue desarrollado por *Autodesk Media & Entertainment*. 3ds Max dispone de una sólida capacidad de edición, una desarrollada arquitectura de *plugins* y una larga tradición en plataformas de Microsoft Windows. Además tiene una optimización integrada con gráficos de alto rendimiento y plataformas específicas. Brinda dos lenguajes para el desarrollo de *plugins, MaxScript y SDK(C++),* estos *plugins* permiten crear, modificar, exportar e importar todos los objetos en una escena haciendo el trabajo más fácil para los usuarios de este programa. Una descripción de estos dos lenguajes se aborda en los epígrafes 1.4 y 1.5 respectivamente. (5)

Proporciona modelos potentes en 3D, animación, *renders* y herramientas de composición que permiten a los artistas y diseñadores estar a un nivel superior en la producción. Ofrece experiencias diferenciadas y conjuntos de herramientas especializadas para los desarrolladores de juegos, artistas de efectos visuales y diseñadores gráficos. (5)

1.4 MaxScript y SDK (C++)

MaxScript:

Es el lenguaje de secuencias de comandos integrado a Autodesk, proporcionando a los usuarios de este producto la mayoría de los aspectos del uso del programa, tales como modelado, animación, materiales, *render*. Permite el control del programa de forma interactiva a través de la ventana de escucha de la línea de comandos, además permite ampliar o reemplazar la interfaz de usuario de los objetos, modificadores, materiales, texturas, efectos de *render*, y efectos atmosféricos. Posibilita desarrollar herramientas para exportar e importar escenas usando archivos ASCII o binarios y escribir controladores de procedimientos que pueden acceder a todo el estado de la escena. (6)

MaxScript es ideal para desarrollar *plugins* de simple complejidad y para escribir pruebas de pequeños fragmentos de funcionalidades. Los *plugins* que se crean con MaxScript corren más lento en comparación con los creados en SDK(C++). (6)

SDK (C++):

Software Development Kit avanzado para el desarrollo personalizado de *plugins* y herramientas. Se compone de un amplio conjunto de clases abstractas en C++ , gran variedad de bibliotecas y el desarrollo de *plugins* es mediante el IDE de desarrollo Visual Studio. Es el preferido para el desarrollo de *plugins* porque presenta bibliotecas y clases que permiten acceder con una mayor facilidad a los nodos de las escenas, además es más efectivo cuando se busca rendimiento. Los *plugins* que se desarrollan con versiones anteriores de SDK deben ser recompilado con la versión apropiada de SDK. (7)

1.5 MaxScript o SDK?

Para el desarrollo del formato de fichero propuesto en el Capítulo dos, es ideal el SDK, ya que permite el uso de bibliotecas posibilitando desarrollar *plugins* que presenten alto nivel de complejidad y necesiten alto rendimiento. Los *plugins* se desarrollan mediante el IDE de desarrollo Visual Studio usando el lenguaje C++; que es el usado por el equipo de desarrollo de la STK, mientras que MaxScript es para *plugins* que no necesiten gran rendimiento y escribir pequeños fragmentos de funcionalidades, además usa un lenguaje de secuencia de comando desarrollado por Autodesk provocando que se deba hacer un estudio previo de este lenguaje.

1.6 Biblioteca 3DXI (IGame.h)

3ds Max *Data Exchange Interface* (3DXI); antes llamada IGame, es una biblioteca que usa SDK(C++) de 3ds Max para exportar escenas. Fue creada en respuesta a la necesidad de llevar a un nivel más simple la forma de extraer los datos de una escena de 3ds Max. Es la más usada por grupos de desarrollo de

plugins para exportar escenas virtuales desde 3ds Max. Presenta clases dirigidas a los nodos y sus atributos, con métodos que permiten mayor flexibilidad y facilidad para acceder a ellos. (7)

1.7 Ficheros para la creación de escenas virtuales

Debe ser un contenedor de información, ya sea de un único objeto o de un mundo virtual complejo. Por lo general estos se guardan de forma binaria o ASCII. Debe contar con todos los atributos que permitan conocer la estructura de la malla, como son los vértices y caras, en el caso más básico. Suelen tener un conjunto de características que brindan un mayor nivel de detalle a las escenas virtuales, entre las cuales podemos mencionar las siguientes:

- Materiales
- Vértices
- Caras
- Vértices con textura
- Caras con textura
- Colores de vértices
- Normales de las caras
- Normales de los vértices
- Estados de *render*
- Cámaras
- Luces

Otros aspectos importantes a tener en cuenta son el formato de almacenamiento y la organización de los atributos en el fichero, estos dos aspectos influyen en la complejidad y el tiempo de carga. El formato del fichero es muy importante ya que de esto depende el tamaño del mismo; un fichero que su información sea escrita en binario será mucho más pequeño que los escritos en ASCII. Por otro lado si se tiene un fichero que su información no está bien estructurada podría afectar negativamente en el tiempo de carga del mismo. (8)

A continuación se mencionan características de los dos tipos de ficheros más usados, así como sus ventajas y desventajas.

1.8 Fichero con formato ASCII

1.8.1 STL (*Standard Tessellation Language*)

El formato STL es nativo de *stereolithography* CAD creado por 3d Systemen Valencia, California, USA. El formato puede encontrarse en datos ASCII o binarios, aunque es más común en formato binario debido a que el tamaño de los archivos es más pequeño. Es uno de los formatos más básicos en cuanto a estructura, muy fácil de cargar y con poca cantidad de atributos, sólo tomando en cuenta, la geometría del objeto (caras y vértices). (9)

Estructura del Fichero STL

***solid*"nombre":** nombre del objeto.
***facet normal* floatfloatfloat:** normales de la cara.
***outerloop*:** inicialización de los vértices de la cara.
***vertex*floatfloatfloat:** coordenadas del vértice en "x, y, z".
***vertex*float floatfloat**
***vertex*float floatfloat**
***endloop*:** fin de la inicialización de los vértices.
***endfacet*:** fin de las declaraciones de cara.
***endsolid*"nombre":** nombre del objeto, dirección o cualquier otro comentario. (9)

Ventajas:
Presenta una estructura muy sencilla la cual facilita cargar el fichero. (9)

Desventajas:

Sólo analiza el físico de la malla, conteniendo las características básicas para definirlas: caras y vértices. En un fichero sólo está contenido un único objeto, en caso específico de exportar más de un objeto en un mismo fichero, se toman todos los objetos como una sola geometría, quitando la posibilidad de trabajar con geometrías por separado. (9)

1.8.2 ASE (ASCII SceneExporter)

Es el usado por el programa 3D Studio Max para exportar las escenas a un formato más "legible" que el archivo .max ya que no se han hecho públicas las especificaciones del mismo. Los datos están almacenados dentro del fichero usando varias banderas que nos indican la propiedad que estamos leyendo.

Vértice y cara: (10)

MESH_NUMVERTEX
Esta palabra clave es seguida por un número entero, esto es el número de vértices en la malla.
MESH_NUMFACES
Esta palabra clave es seguida por un número entero, éste es el número de caras en la malla.
MESH_VERTEX
Palabra clave seguida de un número entero y luego tres *floats*. El entero es el índice de vértice, si hay 120 vértices en la malla los índices irán de 0 a 119. Este índice se utiliza más adelante para la creación de la cara.
MESH_FACE
Palabra clave seguida por cuatro números enteros. El primer número entero es el índice de la cara seguida de un punto y coma. Si hay 250 caras en la malla a continuación, los índices de la cara se van de 0 a 249. Después del índice de la cara habrá una serie de esta manera: A: 0 B: 5 C: 4. Esta serie representa los vértices que conforman la cara.

Normal Data: (10)

MESH_NUMVERTEX

Aunque esta palabra no tiene nada que ver con las normales es útil. Si hay datos normales en el archivo el número de normales de vértice que están contenidos en el archivo es igual al número entero después de esta palabra clave.

MESH_VERTEXNORMAL

Esta palabra clave es seguida por un entero y tres *floats*. El valor entero es el índice, el rango de este índice será de 0 a MEXH_NUMVERTEX – 1.

MESH_FACENORMAL

Esta palabra clave es seguida por un entero y tres *floats*. El valor entero es el índice, el rango de este índice será de 0 a MESH_NUMFACES – 1.

Texture Data: (10)

MESH_NUMTVERTEX

Esta palabra clave es seguida por un número entero, este es el número de vértices de textura en la malla.

MESH_NUMTVFACES

Esta palabra clave es seguida por un número entero, éste es el número de caras de textura en la malla.

MESH_TVERT

Esta palabra clave es seguida por un entero y tres *floats*. El valor entero es el índice, el rango de este índice será de 0 a MEXH_NUMTVERTEX - 1.

MESH_TFACE

Palabra clave seguida por cuatro números enteros. El primer número entero es el índice de la cara con textura. Este valor del índice estará en el intervalo de 0 a MESH_NUMTFACES. Después del índice de la cara habrá tres enteros, estos representan los vértices que componen a la textura de la cara. Cada entero se corresponde con un vértice de texvertex.

1.8.3 DIRECTX

El fichero DirectX fue construido para Direct3D y expandido para DirectX 6.0, a partir de ahí ha ido evolucionando debido a su capacidad de expandirse. (11)

Estructura del fichero DIRECTX

El formato está basado completamente a base de plantillas que definen cómo almacenar la información en el fichero. Presenta una estructura jerárquica que permite insertar más plantillas y así expandir el fichero con nuevos bloques de información. El formato de este fichero se presenta en la Fig. 2. (11)

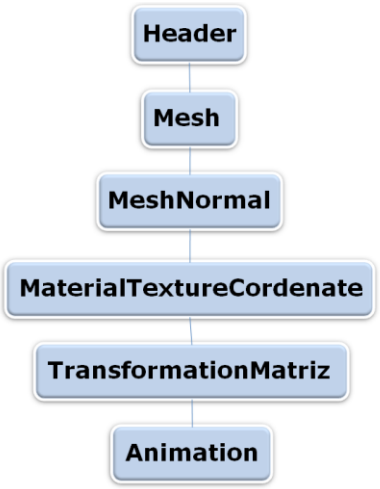

Fig. 2 Estructura del fichero DIRECTX

Ventajas:

Presenta una estructura jerárquica con buena organización y permite ampliar la cantidad de atributos de una escena mediante el uso de plantillas. Optimiza espacio en memoria ya sea en formato ASCII como en binario. (11)

Desventajas:

No cumple con todas las necesidades que se requieren en una herramienta gráfica para la creación de escenas virtuales. (11)

1.9 Fichero con formato binario

1.9.1 STL (Standard Tessellation Language)

En forma binaria usa la representación numérica de puntos enteros y flotantes de IEEE. (12)

Estructura del fichero STL

• **string Comentario**: un comentario al inicio de fichero.
• **unsineg 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.
• **unsined int c**: cuando está en cero indica que terminó la información de la cara.

Lo que está en negrita se repite de acuerdo a la cantidad de caras que tenga el objeto. Los dos últimos bytes son para especificar que ahí termina la información de una cara. (12)

1.9.2 MD2

Es un formato creado por *ID software, inc* para el juego Quake II. Desde el momento que surgió se convirtió en un formato muy popular debido a su facilidad de uso. Los modelos MD2 pueden ser usados para cualquier objeto, tales como: armas, personajes y piezas del terreno. La estructura que presenta es la que se muestra en la Fig. 3. (13)

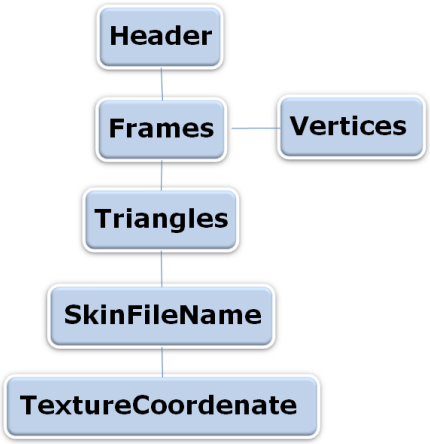

Fig. 3 Estructura del fichero MD2

Ventajas:

Al presentar una estructura en bloques, permite que se pueda guardar de manera sencilla todo el contenido en estructuras de datos. El uso de cabeceras ofrece la ventaja de saber en qué lugar y qué cantidad de un tipo determinado de información contiene el fichero. (13)

Desventajas:

Contiene poca información de una escena, le faltan atributos que pueden ser significativos. (13)

1.9.3 MD3

Es un formato creado por *Id Software, Inc* para el famoso juego Quake *III Arena* y otros derivados como: Retorno al Castillo Wolfenstein, Jedi el Caballero Medieval. Todos comparten las características de la rápida puesta en escena de los elementos 3d, por lo que está presente la alta optimización de recursos. A continuación se muestra la estructura que presenta en la Fig. 4. (14)

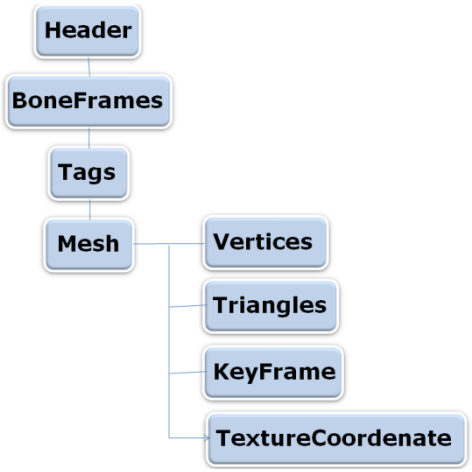

Fig. 4 Estructura del fichero MD3

Ventajas:

Un archivo puede contener más de un objeto. Presenta una estructura en bloques que permite guardar todos los atributos en estructuras de datos de forma organizada, según el orden en que se establecen los bloques. (14)

Desventajas:

Siguen arrastrando la falta de parámetros significativos desde el MD2, entre los que podemos mencionar los colores de vértices y las propiedades de materiales. (14)

1.9.4 3DS

Creado originalmente para 3d StudioMax por Autodesk. Se encuentra en forma de binario y presenta una estructura muy completa. (15)

Estructura del fichero

Todos los 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 esta característica, el formato 3ds puede tener estos bloques en cualquier orden y no presenta una cabecera que especifique donde encontrarlos. Veamos el siguiente ejemplo en la Fig. 5. (15)

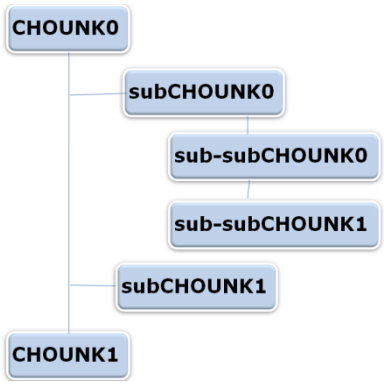

Fig. 5 Estructura del fichero 3DS

Ventajas:

Es un formato que optimiza muy bien el tamaño de los ficheros ya que se encuentra en formato binario. Exporta escenas completas, es decir que un mismo fichero puede contener varias mallas. (15)

Desventajas:

No muestra un orden en los bloques de datos. La estructura que muestra es compleja y difícil de cargar. (15)

1.10 API gráfica OpenGL

Creada por Silicon Graphics en 1992, es el principal ambiente de programación para desarrollar aplicaciones gráficas existentes. Es multiplataforma (habiendo incluso un openGL para móviles). Está diseñado para usarse en una gran variedad de sistemas operativos sin tener que modificarlo o modificándolo muy poco, esto significa una gran ventaja para cualquier desarrollador al abarcar el mayor mercado posible. Además su arquitectura fue pensada para poder evolucionar conforme avanza el tiempo, de modo que pueda adaptarse con facilidad a las novedades que van surgiendo en la industria. Desde que apareció, OpenGL ha sido un ambiente de programación estable y confiable, y quizás la razón por la que es más conocido es porque fue la primera en ofrecer la posibilidad de desarrollar aplicaciones 3D; fue precisamente con OpenGL que se hicieron los primeros juegos 3D como Wolfenstain y Doom, hoy también es muy usado en juegos como Quake y muchos más. (16)

Arquitectura de OpenGL

Es una colección de cientos de funciones que dan acceso a todas las ventajas ofrecidas por el HW gráfico. En el corazón del OpenGL está el ***rendering pipeline***, (tubería de *rendering:* serie de pasos que se siguen para aplicar el *render*, y que son manipulados por OpenGL) (17)

Sobre Windows, OpenGL brinda una alternativa que consiste en usar *Graphics Device Interface* (**GDI**, Interfaces de Dispositivos Gráficos), diseñada para hacer el HW gráfico completamente invisible al programador a través de capas de abstracción (esto aminora la velocidad). Windows define además un conjunto de funciones API Win32 que son usadas como interfaz con OpenGL. (17)

OpenGL, no obstante, ofrece su propio grupo de bibliotecas conocido como *OpenGLUtilityToolKit*(**GLUT**, Herramientas y Utilidades de OpenGL), con soportes disponibles en la mayoría de las plataformas y funcionalidades básicas en el área del trabajo con ventanas. GLUT mantiene la independencia de las plataformas, es decir, se puede mover fácilmente una aplicación basada en GLUT de Windows hacia Unix, con unos pocos cambios. (17)

1.11 API gráfica Direct3D

Direct3D es parte de DirectX, una API desarrollada por Microsoft disponible tanto en los sistemas Windows de 32 y 64 bits, como para sus consolas Xbox y Xbox 360 para la programación de gráficos 3D.
El objetivo de Direct3D es facilitar el manejo y trazado de entidades gráficas elementales, como líneas, polígonos y texturas, en cualquier aplicación que despliegues gráficos en 3D, así como efectuar de forma transparente transformaciones geométricas sobre dichas entidades. Direct3D provee también una interfaz transparente con el hardware de aceleración gráfica. (18)

Arquitectura de Direct3D

Con arquitectura basada en el COM de Microsoft, la mayor ventaja que presenta Direct3D frente al GDI es que Direct3D se comunica directamente con los drivers de pantalla, consiguiendo mejores resultados en la representación de los gráficos por pantalla que OpenGL. (18)

Direct3D está compuesto por el modo retenido y el modo inmediato. El modo inmediato da soporte a todas las primitivas de procesamiento 3D que permiten las tarjetas gráficas. El modo retenido, construido sobre el anterior, presenta una abstracción de nivel superior ofreciendo funcionalidades de gráficos como jerarquías o animaciones. El modo retenido ofrece muy poca libertad a los desarrolladores, siendo el modo inmediato el que más se usa. (18)

El modo inmediato de Direct3D trabaja fundamentalmente con los llamados dispositivos (*devices*). Son los encargados de realizar los *render* de la escena. El dispositivo ofrece un interfaz que permite diferentes opciones de *render*. Por ejemplo un dispositivo mono permite hacer *render* en blanco y negro mientras que un dispositivo RGB permite el uso de colores. (18).

Conclusiones del capítulo

En el transcurso de este capítulo, se formularon algunos conceptos básicos para un mejor entendimiento del tema y del próximo capítulo. Se mostraron las características necesarias a conocer de la STK, específicamente la estructura de los nodos del grafo de escena. Se caracterizaron también las opciones que presenta 3ds Max para el desarrollo de *plugins* y se describió por que el SDK es más útil para desarrollar la solución propuesta en el Capítulo dos. Se realizó un estudio de los ficheros más usados para crear entonos virtuales, lo cual permitió conocer características importantes y necesarias para desarrollar un formato de fichero útil que permita usar eficientemente las características de la nueva versión de la STK.

Capítulo 2 Descripción de la solución propuesta

En este capítulo se exponen las características que presenta el formato de fichero propuesto. A partir de este capítulo se comienza a tener una visión del sistema obteniendo una descripción conceptual de la solución propuesta. Se muestran conceptos, restricciones, requisitos y los casos de uso; necesario para modelar la solución propuesta en el presente capítulo.

2.1 Características del fichero propuesto

Dado que ninguno de los formatos de fichero estudiados en el Capítulo uno presentan la estructura adecuada para crear las estructuras de datos de la versión actual de la STK, es necesario el diseño de un nuevo formato. De los ficheros estudiados se tomaron importantes características de su estructura, así como sus ventajas y desventajas que permitieron el diseño del formato propuesto por el presente trabajo de diploma.

En el Capítulo uno se muestra que la mayoría de los formatos de ficheros existentes utilizan una estructura en forma de bloques de datos, lo que permite que al leer el fichero se cargue un bloque de información completo a la vez, logrando que no sea necesario hacer varias lecturas del fichero y propiciar una adecuada organización de la información; por esa razón se decide que el fichero propuesto presente una estructura en forma de bloques de datos.

También se investigó que los ficheros que son salvados en formato binario ocupan menos espacio en disco y brindan la posibilidad de tener confidencialidad de la información; por esto el fichero propuesto se salva en formato binario, obteniendo estas ventajas.

Debido a que el fichero propuesto por el presente trabajo de diploma es desarrollado para la SceneToolKit, la extensión que se emplea es .STK, que son las abreviaturas por las que se conoce a esta herramienta gráfica.

En la Fig. 6 se muestra la estructura del fichero diseñado.

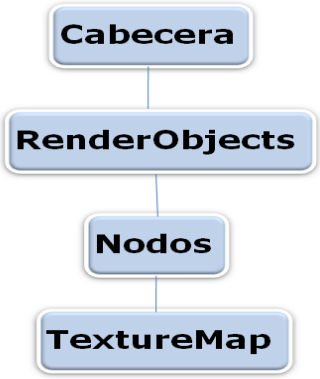

Fig. 6 Estructura del fichero .STK

Notas:

1. El fichero contiene un tipo de dato denominado **Property**, el cual presenta las siguientes características:
   tipo: NULL=-1,int=0, float=1,Point3=2,char=4,Point4=5.
   valor: valor asignado de acuerdo al tipo.
   Un ejemplo de este tipo de dato: Point3 con x = 1, y = 1, z=1;
   3111, el primer número representa el tipo, el segundo “x”, el tercero “y” y el cuarto “z”.

2. En el fichero cada string esta precedido de un entero que representa la cantidad de caracteres que posee el string.
   Ejemplo: 16Material-defaul2
   16 representa la cantidad de caracteres a leer.
   Material-defaul2 es la cadena que se debe obtener al leer del fichero.

2.2 Bloque Cabecera

Es el bloque que representa el encabezado del fichero. En esta parte se encuentra información general que permite conocer a grandes rasgos el contenido del fichero.

A continuación se muestra la estructura del bloque cabecera.

- int id_Fichero (1001207): identificador del fichero, es único y representa un fichero STK.
- int cant_RO: cantidad de RenderObjects exportados.
- int cant_Nodos: cantidad de Nodos del grafo escena.
- int cant_Tex: cantidad de TextureMap usadas por los materiales.

2.3 Bloque RenderObjects

Este bloque contiene la lista de objetos que se exportan. Es ideal para la estructura del fichero, ya que es posible leer todos los objetos exportados, permitiendo almacenar en memoria la información de todos antes de leer el siguiente bloque.

A continuación se muestra la estructura de este bloque.

- int tipo_RO: tipo del RO: 0 – Forma, 1 – Atributo.
- int tipo_FA: tipo de Forma o Atributo:
  Si tipo_RO: 0; tipo_FA: 0 – Mesh, 1 – Spline.
  Si tipo_RO: 1; tipo_FA: 0 – Material, 1 – Cámara, 2 – Luz.
- int id_RO: identificador del RO.
- string nom_RO: Nomebre del RO

Como se puede observar en la Fig. 7, el bloque RenderObjects se divide en cinco sub-bloques que están organizados como se muestra en la figura, permitiendo que se lean todos los objetos de un tipo antes de pasar a leer otro tipo de objeto.

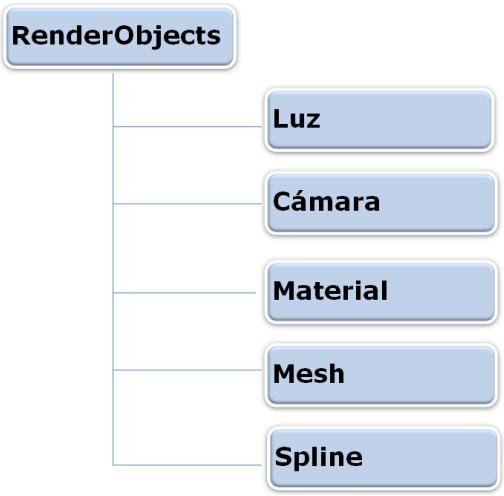

Fig. 7 Estructura bloque RenderObjects

2.3.1 Sub-Bloque Luz

Este sub-bloque permite almacenar información de las luces de una escena, con lo cual se mejora el proceso de creación de escenas virtuales en la STK, permitiendo crear escenas más realistas y complejas. Ninguna de las dos soluciones anteriores abordadas en la Introducción de este trabajo presenta información de luces, por tal motivo este problema se resuelve satisfactoriamente en el formato de fichero propuesto. En la Fig. 5 en Anexos 3 se puede observar dicha información.

A continuación se muestra la estructura del sub-bloque.

- Property c_luz: representa el color de la luz.
- Property m_luz: multiplicador de la luz.
- Property i_At: Inicio de la atenuación de la luz.
- Property f_At: fin de la atenuación de la luz.
- Property off_luz: representa el fall_Of de la luz.
- Property hot_Spot: representa el hot Spot de la luz.
- Property asp_ratio: representa el aspect ratio de la luz.

- Property dec_Start: representa el Decay Start de la luz.
- int tipo_luz: representa el tipo de luz: 0 – Omni, 1 – Spot Target, 2 – Dir Free, 3 – Spot Free, 4 – Dir Target, 5 – Sky.
- int dec_Type: tipo de Decay de la luz: 0 – None, 1 – Inverse, 2 – Inverse Square.
- int shape: tipo del shape usado por la luz: 0 – Rect Ligth, 1 – Circle Ligth.
- int on: representa si la luz esta encendia o apagada: 0 – apagada, 1 – encendida.
- int es_target: Representa si la luz es target o free: 0 – free, 1 – target.
- float [16] m_G: matriz global del objeto.
- float [16] m_L: matriz local del objeto.

2.3.2 Sub-Bloque Cámara

Las soluciones anteriores vistas en la Introducción de este trabajo no presentan información de cámaras, lo cual se resuelve en el formato de fichero propuesto. En este sub-bloque se almacena toda la información de la cámara usada en la escena exportada, permitiendo crear escenas más realistas y complejas en la STK. Esta información se puede observar en la Fig. 6 en Anexos 3.

A continuación se muestra la estructura del sub-bloque.

- int es_Target: representa si la cámara es libre o target: 0 – No, 1 – Si.
- Property t_Dist: representa la distancia del target de la cámara.
- Property fov: representa el FOV de la cámara.
- Property f_Clip: representa el Far Clip de la cámara.
- Property n_Clip: representa el Near Clip de la cámara.
- float [16] m_G: matriz global del objeto.
- float [16] m_L: matriz local del objeto.

2.3.3 Sub-Bloque Material

Este sub-bloque contiene la información de cada uno de los materiales usados en la escena. Presenta dos atributos permitiendo crear materiales con multitexturas; el primero denominado “cant_TexMap” que permite conocer la cantidad de texturas en el material y el segundo denominado “tex_Index” que contiene los índices de la lista de texturas (almacenadas en el bloque TextureMap) usadas por el material. El formato de fichero 3DX visto en la introducción del presente trabajo de diploma no permite la creación de materiales con multitexturas, el formato de fichero propuesto si brinda esta posibilidad mediante la información almacenada en este sub-bloque, permitiendo crear escenas virtuales más complejas y realistas. En la Fig. 7 en Anexos 3 se puede observar dicha información.

A continuación se muestra la estructura del sub-bloque.

- string class_Mat: representa la clase del materia, ejemplo: Standard.
- string nom_Mat: representa el nombre del Material como se ve en el editor de 3ds Max.
- int two_Side:1 si el material es Two Side, 0 caso contrario.
- Property amb_Data: valor ambiental.
- Property dif_Data: valor difuso.
- Property emis_Data: valor Emissive.
- Property gloss_Data: valor Glossenis.
- Property opac_Data: valor de Opacidad.
- Property spec_Data: valor Specular.
- Property emis_A: Emisive Amount.
- Property spec_L: nivel Specular.
- float shif: nivel de brillo.
- Color [r,g,b] c_A: color ambiental.
- Color [r,g,b] c_D: color Difuso.
- Color [r,g,b] c_S: color Specular.
- int cant_TexMap: cantidad de TextureMap usadas por el material
- int [cant_TexMap] tex_Index: indice de la lista de TextureMap usadas por el material.

2.3.4 Sub-Bloque Mesh

La repetición de vértices es un problema que presenta el fichero 3DX como se pudo conocer en la Introducción de este trabajo de diploma. El sub-bloque Mesh resuelve este problema almacenando una lista de vértices que coincide con la cantidad de vértices encontrados en la geometría y con la estructura que realmente se van a usar en la visualización de las escenas en la STK. En la Fig. 8 en Anexos 3 se muestra esta información.

A continuación se muestra la estructura de este sub-bloque.

- string name_Mat: nombre del material usado por el objeto.
- int cant_V: cantidad de vértices de la malla.
- Point3 [cant_V] vert: lista de los vértices de la malla.
- int cant_C: cantidad de caras (triángulos) en la malla.
- int [cant_C *3] caras: representa la lista de caras que componen la malla; una cara es un arreglo con los tres índices de la lista de vértices que componen la cara.
- int cant_TexVertex: cantidad de TexVertex encontrados en la malla.
- Point2 [cant_TexVertex] texVertex: lista de los TexVertex encontrados en la malla.
- float [16] m_G: matriz global del objeto.
- float [16] m_L: matriz local del objeto.

2.3.5 Sub-Bloque Spline

En este fichero también se adiciona otro tipo de objeto que puede ser dibujado, es el caso de los *Splines*. El fichero 3DX no soporta información de este tipo de objeto, lo cual es otra mejora con respecto a ese fichero. En la Fig. 9 en Anexos 3 se muestra la información de este sub-bloque.

A continuación se muestra la estructura del sub-bloque Spline.

- int cant_Knot: cantidad de Knots que tiene la spline.

- Point3 [x,y,z] vect_In: vector de entrada del Knot.
- Point3 [x,y,z] vect_Out: vector de salida del Knot.
- Point3 [x,y,z] p_Knot: posición del Knot.
- int tipo_Knot: tipo del Knot: 0 – Auto, 1 – Corner, 2 – Bezier, 3 – Bezier-Corner.
- float [16] m_G: matriz global del objeto.
- float [16] m_L: matriz local del objeto.

2.4 Bloque Nodos

Este bloque contiene la información de los nodos exportados permitiendo crear la estructura de los nodos del grafo de escena de la versión actual de la STK. El fichero 3DX no permite crear la estructura del grafo de escena y la solución realizada en el año 2007 no crea la estructura del grafo de escena de la versión actual de la STK, lo que hace que el formato de fichero propuesto sea más útil. En la Fig. 10 en Anexo 3 se muestra la información referente a este bloque.

A continuación se muestra la estructura del bloque.

- int id: identificador del nodo.
- string tipo: tipo del nodo. Camara_Luz, Luz, Camara, Material, Trimesh, Spline.
- string nombre: nombre del nodo.
- string nom_padre: nombre del padre
- string nom_Mat: nombre del material usado por el nodo; es “null” si el nodo no es de tipo “Trimesh”
- int cantRO: cantidad de RO que son incluidos en el nodo.
- int [cantRO] ind_RO: índices de los RO asociados al nodo.
- float [16] m_G: matriz global del objeto.
- float [16] m_L: matriz local del objeto.

2.5 Bloque TextureMap

Este bloque contiene la lista de texturas usadas por los materiales en la escena. Es un bloque creado para poder crear de forma más sencilla materiales con multitexturas, permitiendo no repetir información de una misma textura usada por varios materiales. En la Fig. 11 en Anexo 3 se muestra dicha información.

A continuación se describe la estructura del bloque.

- int id_Tex: representa el identificador de la textura.
- string nom_Tex: nombre de la textura.
- string c_Tex: clase de la textura.
- string nom_File: dirección completa donde está la textura.
- int map_Ch: canal de mapeo usado por la textura.
- int slot: slot por el cual fue pasada la textura:1 – Difuso, 2 – Especular, 3 – Glossines, 4 – Nivel Especular, 5 – Self Iluminate, 6 – Opacidad.
- Property u_C: U Clip de la textura.
- Property v_C: V Clip de la textura.
- Property h_C: H Clip de la textura.
- Property w_C: W Clip de la textura.
- Property u_Off: U Off Set de la textura.
- Property v_Off: V Off Set de la textura.
- Property u_T: U Tilling de la textura.
- Property v_T: V Tilling de la textura.
- Property u_A: U Angle de la textura.
- Property v_A: V Angle de la textura.
- Property w_A: W Angle de la textura.

2.6 Aspectos generales

Para el desarrollo de la solución propuesta por el presente trabajo de diploma se seleccionó el SDK de entre las dos opciones que brinda 3ds Max 2011, permitiendo el desarrollo de *plugin* de alto rendimiento y flexibilidad como se expuso en el epígrafe 1.5 del Capítulo uno, además para poder hacer uso de la biblioteca 3DXI, ideal para la extracción de datos de una escena, descrita en el epígrafe 1.6 del Capítulo uno. También se hace uso de la herramienta Visual Studio Express Edition 2008 SP1, debido a que esta versión del Visual Studio tiene instalada la actualización de seguridad de julio del 2008, necesaria para poder utilizar el SDK de 3ds Max 2011. El diseño e implementación se hace siguiendo la programación orientada a objeto usando el lenguaje C++, debido a que el SDK de 3ds Max está desarrollado en este lenguaje.

2.7 Reglas del negocio

A continuación se definen las reglas del negocio orientadas a este modelo de dominio.

- Para poder generar el fichero se debe tener instalado el 3ds Max 2011 o versiones posteriores a la 6.0
- La escena no puede estar vacía para poder generar el fichero.
- Todas los objetos derivados de los *Spline* deben ser convertidas a *Editable Poly* antes de ser exportadas, de lo contrario no se exportan.
- Soporta una sola cámara en la escena, de lo contrario no se exporta el fichero.
- Cada objeto tendrá solamente un material.
- No soporta multimateriales.
- En caso de que no se quiera exportar un objeto, este debe ponerse invisible.

2.8 Modelo de dominio

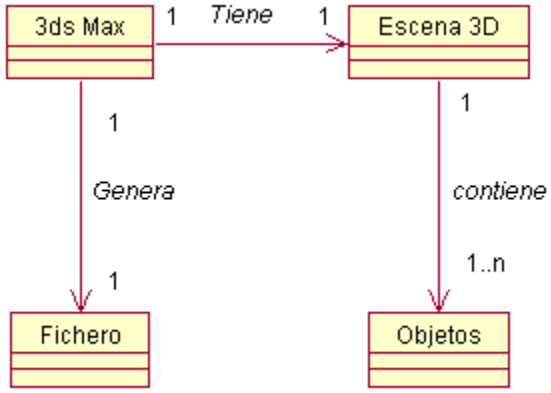

Fig. 8 Modelo de Dominio

2.9 Glosario de términos del modelo de dominio

3ds Max: Herramienta donde se crea la escena 3d.

Escena 3D: Es una escena virtual en la cual se ubican objetos en el espacio 3d.

Objetos: Son los componentes que integran una escena virtual: geometrías, *spline*, cámaras, luces, materiales y texturas.

Fichero: Es un contendor de información referente a una escena 3d.

2.10 Captura de requisitos

A continuación se describen los requisitos funcionales y no funcionales del sistema. Es importante mencionar que los requisitos funcionales y no funcionales expuestos a continuación son con respecto a la aplicación que permite el desarrollo del formato de fichero propuesto por el presente trabajo de diploma.

2.10.1 Requisitos funcionales

1. Almacenar lista de RenderObjects.
2. Almacenar lista de STK_Nodos.
3. Almacenar lista de STK_TextureMap.
4. Crear RenderObjects.
5. Crear STK_Mallas.
6. Crear STK_Spline.
7. Crear STK_Luz.
8. Crear STK_Camara.
9. Crear STK_Material.
10. Crear STK_TextureMap.
11. Crear STK_Nodo.
12. Crear grafo de escena.
13. Exportar cabecera del fichero.
14. Exportar lista de RenderObjects.
15. Exportar lista de STK_Nodos.
16. Exportar lista de STK_TextureMap.
17. Exportar STK_Mesh.
18. Exportar STK_Spline.
19. Exportar STK_Luz.
20. Exportar STK_Camara.
21. Exportar STK_Material.
22. Exportar STK_textureMap.
23. Crear fichero .STK.

2.10.2 Requisitos no funcionales

Usabilidad:
Cualquier usuario de 3ds Max.

Soporte:
Compatible con versiones anteriores a 3ds Max 2011 y posterior a la 6.0 que corren sobre plataforma Microsoft Windows (recomendado Windows 7).

Software:
3ds Max 2011 o versiones posteriores a la 6.0. DirectX 9.0c o superior. Cualquier versión del Sistema Operativo Microsoft Windows, (Windows 7 recomendado). Versión de Visual Studio SP1 que tenga instalada la actualización de seguridad de julio del 2008; necesario para poder usar el SDK de 3ds Max 2011.

Hardware:
RAM 512 o superior.

2.11 Modelo caso uso del sistema

En esta sección se muestran los casos de usos del sistema divididos de acuerdo a etapas importantes en el proceso de exportación del fichero, así como los actores que van a interactuar con cada uno de ellos.

2.11.1 Actor del sistema

Actor	Justificación
Diseñador	El diseñador se beneficia con las funcionalidades que brinda el sistema permitiéndole exportar ficheros con el formato .STK.

Tabla 1 Descripción del actor del sistema

2.11.2 Casos de uso del sistema

A continuación se exponen los casos de usos propuestos para este sistema.

1. Exportar fichero
2. Crear RenderObjects
3. Crear STK_Nodos

2.11.3 Diagrama caso de uso del sistema

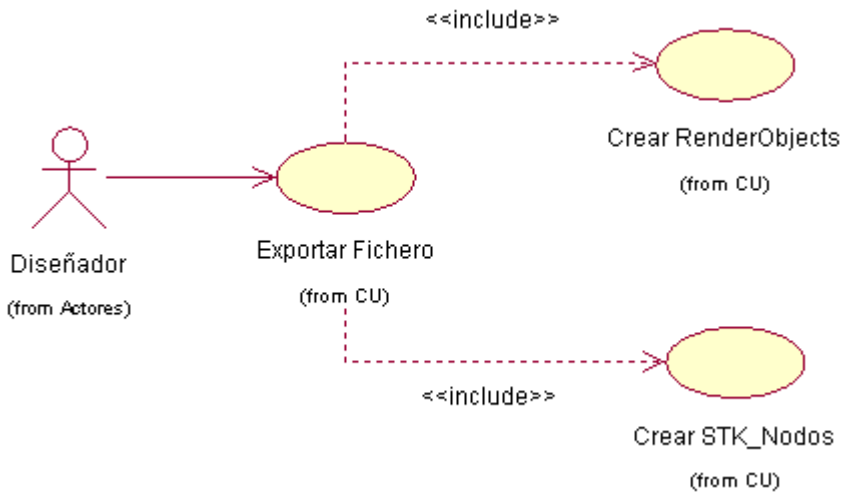

Fig. 9 Diagrama CU del sistema

El diseñador inicia el CU “Exportar fichero” el cual se encarga de almacenar y exportar toda la información de la escena en el fichero .STK. Este CU incluye dos sub-procesos, donde se encuentran los CU “Crear RenderObjects” que se encarga de crear y almacenar toda la información referente a los RenderObjects de la escena, el CU “Crear STK_Nodos” es el encargado de crear y almacenar toda la información de los STK_Nodos del grafo escena.

2.11.4 Descripción de los CU en formato expandido

Para entender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del diagrama de casos de uso, por este motivo se hace la expansión de los mismos.

Caso de uso	
CU-1	**Exportar fichero.**
Propósito	Exportar toda la información de una escena.
Actores Diseñador	
Resumen: El CU inicia cuando el diseñador solicita la opción STK_Exporter (*.STK) en el 3ds Max. Posteriormente selecciona el lugar donde se almacenará el fichero y se procede a almacenar y exportar toda la información de la escena, terminando así el CU.	
Referencias	R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, CU-2<include>, CU-3<include>

Acción del actor	**Respuesta del sistema**
1. Selecciona la opción exportar .STK en el 3ds Max.	2. Se verifica la existencia y visibilidad de nodos en la escena.
	3. Si no existen nodos ir al flujo alternativo “Escena vacía”.
	4. Se crea el fichero .STK.
	5. Se invoca el CU-2 “Crear RenderObjects”.
	6. Se invoca el CU-3 “Crear STK_Nodos”.
	7. Se escribe en el fichero la información del bloque cabecera.
	8. Se escribe en el fichero la información de los RenderObjects de la escena.
	9. Se escribe en el fichero la información de los STK_Nodos del grafo escena.
	10. Se escribe en el fichero la información de las

	STK_TextureMap usadas por los STK_Mteriales.
	11. Se cierra el fichero creado.
	12. Se libera la biblioteca 3DXI.
	13. Se muestra el mensaje “Éxito al exportar”.
Flujo alternativo Escena vacía	
Acción del actor	**Respuesta del sistema**
	3. Se muestra el mensaje “Fallo al exportar”
Puntos de extensión.	

Tabla 2 Expansión CU “Exportar fichero”

Caso de uso	
CU-2	**Crear RenderObjects**
Propósito	Crear y almacenar todos los RO en de la escena.
Actores	
Resumen: Se guarda la información necesaria de todos los objetos de la escena y se crean los RO correspondientes, terminando así el CU.	
Referencias	R1, R3, R4, R5, R6, R7, R8, R9, R10
Acción del actor	**Respuesta del sistema**
	1. Verifica la existencia de material en el nodo.
	2. Si no tiene material ir al flujo alternativo “No Material”.
	3. Se guarda la información del material
	4. Verifica si el material contiene texturas.
	5. Si no contiene texturas ir al flujo alternativo “No texturas”.
	6. Se guarda la información de las texturas.

	7. Se crea la STK_TextureMap.
	8. Se almacena la STK_TextureMap en la lista de Texturas.
	9. Se crea el STK_Material.
	10. Se almacena el material en la lista de RO.
	11. De acuerdo al tipo de objeto que posea el nodo, se crea el RO correspondiente y se almacena en la lista de RO.
Flujo alternativo No Material	
Acción del actor	**Respuesta del sistema**
	3 De acuerdo al tipo de objeto que posea el nodo, se crea el RO correspondiente y se almacena en la lista de RO.
Flujo alternativo No texturas	
Acción del actor	**Respuesta del sistema**
	5 Se crea el STK_Material.
	6 Se almacena el material en la lista de RO.
	7 De acuerdo al tipo de objeto que posea el nodo, se crea el RO correspondiente y se almacena en la lista de RO.
Puntos de extensión.	

Tabla 3 Expansión CU "Crear RenderObjects"

Caso de uso	
CU-3	**Crear STK_Nodos**
Propósito	Crear y almacenar los nodos del grafo escena.
Actores	

Resumen: Se crean los nodos y el grafo escena. Posteriormente estos se almacenan en una lista para ser exportados, terminando así el CU.	
Referencias	R2, R11,R12
Acción del actor	**Respuesta del sistema**
	1. Se crea el nodo raíz.
	2. Se crea el grafo de escena con el nodo raíz.
	3. Se crea el STK_Nodo de acuerdo al RO.
	4. Se adiciona el STK_Nodo al grafo de escena.
	5. Se pasa el grafo de escena para la lista de STK_Nodos.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Puntos de extensión.	

Tabla 4 Expansión CU "Crear STK_Nodos"

Conclusiones del capítulo

A lo largo de este capítulo se describe cada uno de los aspectos del formato de fichero propuesto. Además se han descrito las funcionalidades críticas que deben estar presentes en la solución. Se definieron los requisitos funcionales y no funcionales, agrupando los requisitos funcionales en casos de uso descritos para un mejor entendimiento de los procesos. Se presentaron las aplicaciones con la cual se desarrolla el sistema.

Capítulo 3 Diseño e implementación del sistema

En el presente capítulo se describen los diagramas de clases del diseño y las características de las clases arquitectónicamente significativas. Además se muestran los diagramas necesarios para la implementación de la solución.

3.1 Diagrama de clases del diseño

Debido a la complejidad del diagrama de clases del diseño y para una mejor comprensión del mismo por parte de los lectores se han agrupado las clases en paquetes. Dentro del paquete 3ds Max SDK se encuentran las clases usadas del SDK de 3ds Max para exportar escenas. En el paquete Biblioteca 3DXI, se representan las clases usadas de la biblioteca 3DXI. El paquete STK_Exporter contiene las clases diseñadas para la implementación del sistema. El paquete Formas contiene las relaciones de las clases STK_Mesh y STK_Spline con el sistema mientras que el paquete Atributos contiene las relaciones de las clases CSTK_Material, CST_Camara y CSTK_Luz con el sistema.

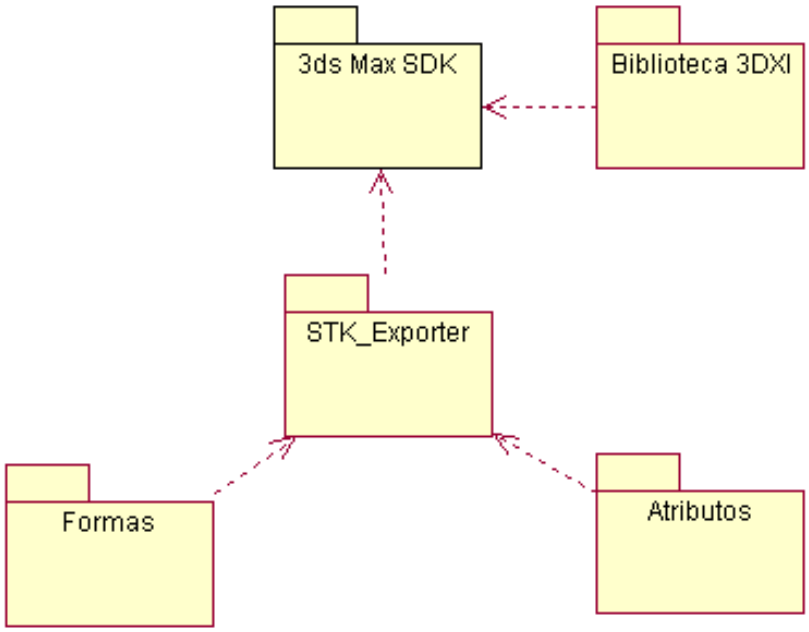

Fig. 10 Diagrama de paquetes del diseño

A continuación se describen los diagramas de clases de acuerdo a los paquetes a las que pertenecen.

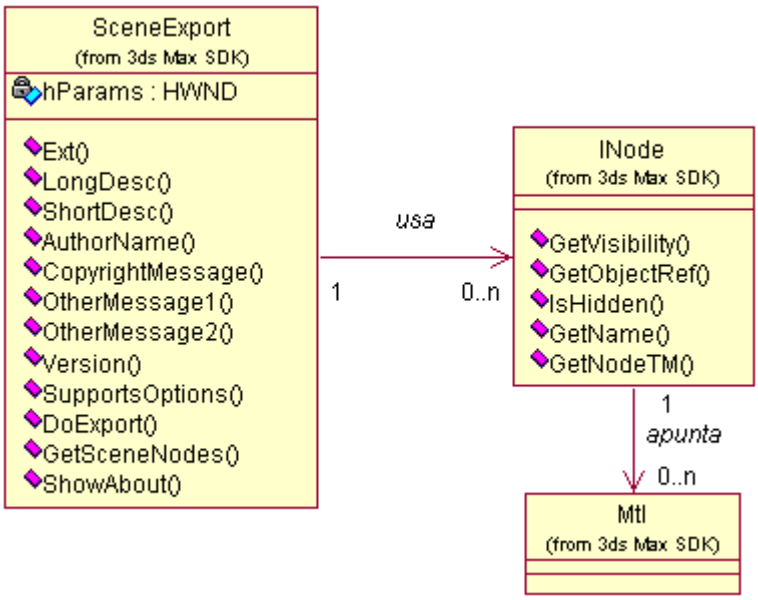

Fig. 11 Diagrama de clases del paquete 3ds Max SDK

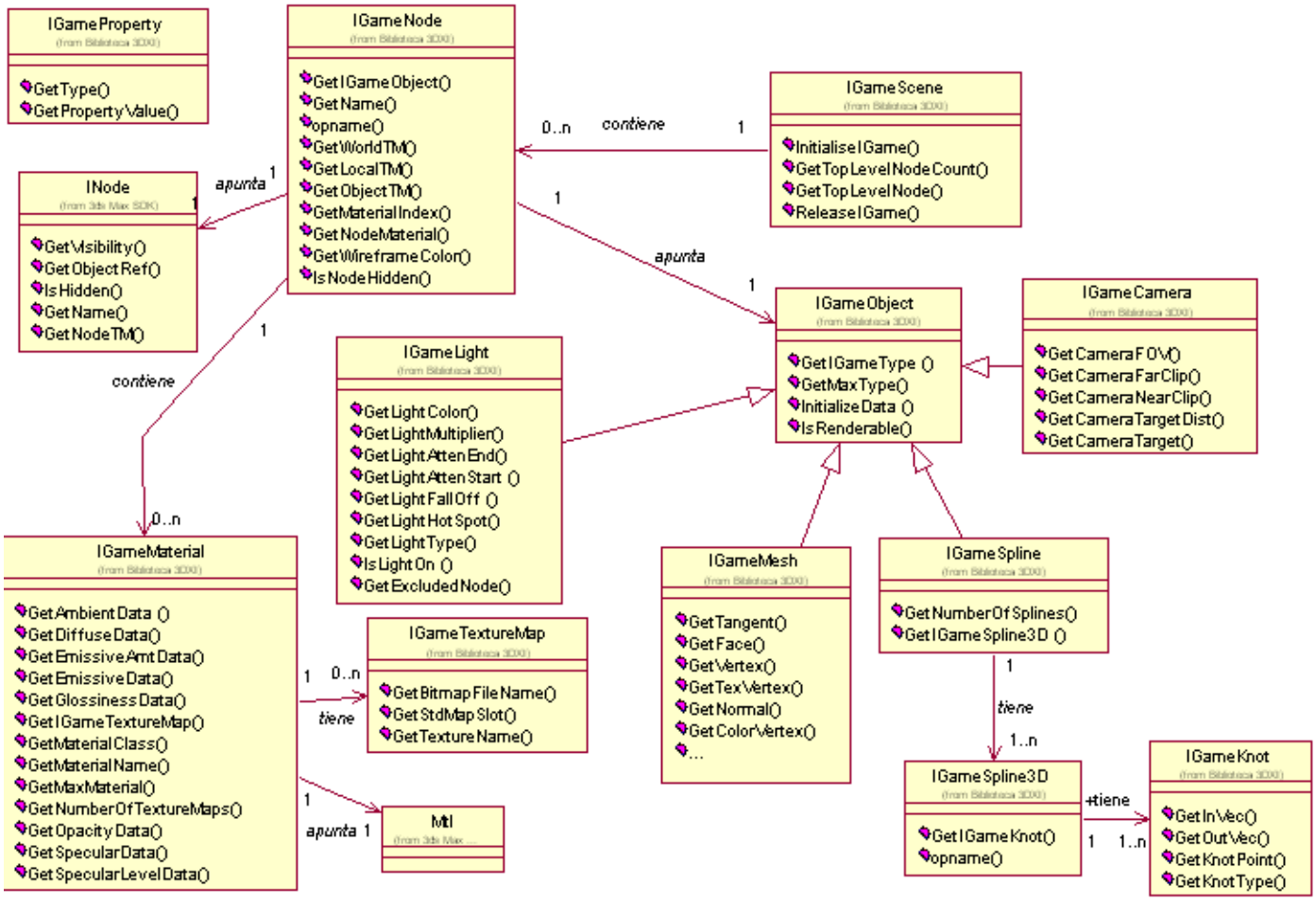

Fig. 12 Diagrama de clases del paquete Biblioteca 3DXI

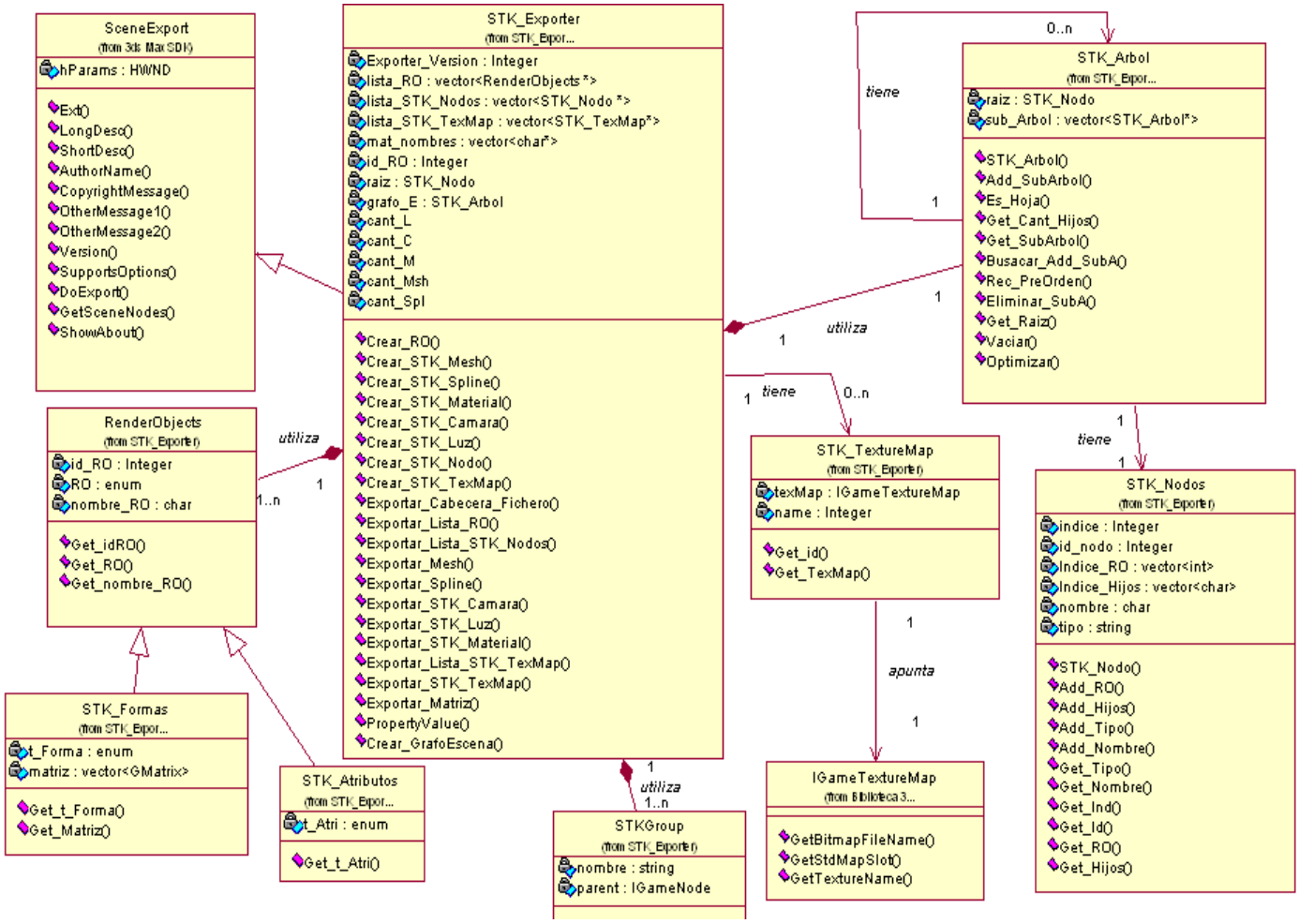

Fig. 13 Diagrama de clases del paquete STK_Exporter

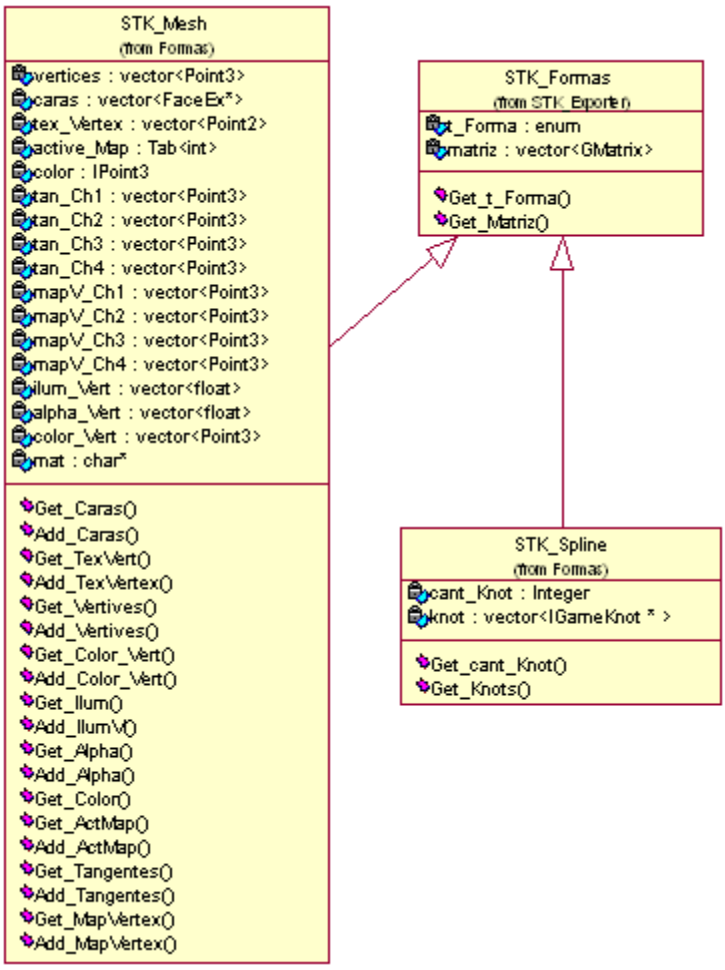

Fig. 14 Diagrama de clases del paquete Formas

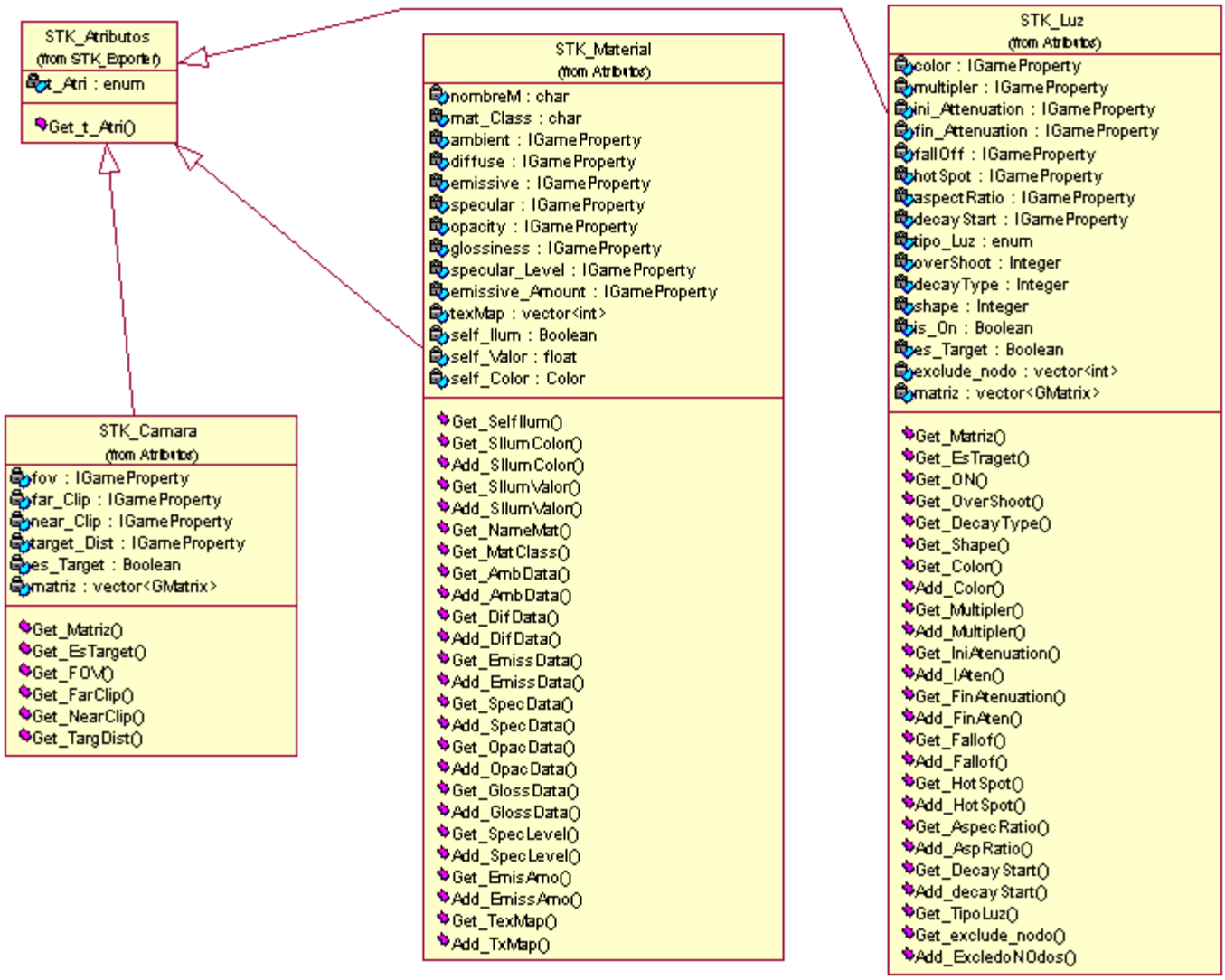

Fig. 15 Diagrama de clases del paquete Atributos

3.2 Descripción de las clases del diseño

A continuación se describen las clases arquitectónicamente significativas. Del paquete 3ds Max SDK solo se describe la clase SceneExport, debido a que de esta son heredados métodos y atributos. Los métodos de acceso a miembros de clases; en este caso se opta por usar “Get_” y “Add_” como métodos de acceso

y de modificación respectivamente, no son especificados ya que su implementación y funcionalidad son sencillas.

Nombre: SceneExporter	
Tipo de clase: Entidad	
Atributo	**Tipo**
hParams	HWND
Para cada responsabilidad:	
Nombre	**ExtCount()**
Descripción	Retorna el número de extensiones soportadas por el *plugin.*
Nombre	**Ext(int n)**
Descripción	Retorna la extensión de acuerdo con el índice n.(en nuestro caso .STK)
Nombre	**LongDesc()**
Descripción	Retorna una descripción amplia del *plugin*.
Nombre	**ShortDesc()**
Descripción	Retorna una corta descripción del *plugin*.
Nombre	**AuthorName()**
Descripción	Retorna el nombre del autor del *plugin*.
Nombre	**CopyrightMessage()**
Descripción	Derechos de autor.
Nombre	**OtherMessage1()**
Descripción	Retorna un mensaje.
Nombre	**OtherMessage2()**
Descripción	Retorna un mensaje.
Nombre	**Version()**
Descripción	Retorna la versión del Exportador.
Nombre	**ShowAbout(HWND hWnd)**
Descripción	Retorna infamación de interés acerca del *plugin.*
Nombre	**SupportsOptions(int ext, DWORD options)**

Descripción	Decide si todas las opciones son soportadas.
Nombre	**GetSceneNodes(INodeTab& i_nodeTab, INode* i_currentNode /*=NULL*/)**
Descripción	Extrae todos los nodos de la escena para inicializar la biblioteca 3DXI.
Nombre	**DoExport()**
Descripción	Es donde ocurre todo el proceso de extracción y almacenamiento de información, es el método principal donde se hacen llamadas a los demás métodos para lograr exportar la información al fichero.

Tabla 5 Descripción de la clase SceneExport

Es importante aclarar que los métodos de la clase SceneExport son heredados por STK_Exporter; la cual se describe a continuación.

Nombre: STK_Exporter	
Tipo de clase: Controladora	
Atributo	**Tipo**
Exporter_Version	float
lista_RO	vector<RenderObjects *>
lista_STK_Nodos	vector<STK_Nodo *>
lista_STK_TexMap	vector<STK_TexMap*>
Lista_Grupos	vector<STKGroup*>
mat_nombres	vector<string>
id_RO	int
id_STK_Nodo	int
texMap_Cont	int
raiz	STK_Nodo*
grafo_E	STK_Arbol *
cant_L	int
cant_C	int
cant_M	int

cant_Msh	int
cant_Spl	int
Para cada responsabilidad:	
Nombre	**Crear_RO()**
Descripción	Encargado crear los RO.
Nombre	**Crear_STK_Mesh()**
Descripción	Extrae la información de los objetos con malla y crea los STK_Mesh.
Nombre	**Crear_STK_Spline()**
Descripción	Extrae la información de las Spline y crea las STK_Spline.
Nombre	**Crear_STK_Material()**
Descripción	Extrae la información de los materiales y crea los STK_Material.
Nombre	**Crear_STK_Camara()**
Descripción	Extrae la información de las cámaras y crea las STK_Camara.
Nombre	**Crear_STK_Luz()**
Descripción	Extrae la información de las luces y crea las STK_Luz.
Nombre	**Crear_STK_TexMap()**
Descripción	Crea las STK_TextureMap.
Nombre	**Crear_NodoAtri ()**
Descripción	Crea los STK_Nodos de tipo Atributo.
Nombre	**Crear_NodoFor ()**
Descripción	Crea los STK_Nodos de tipo Forma.
Nombre	**Crear_NodoGrupo ()**
Descripción	Crea los STK_Nodos de tipo Grupo.
Nombre	**Crear_GrafoEscena ()**
Descripción	Crea al grafo escena con los STK_Nodos.
Nombre	**Exportar_Cabecera_Fichero(ofstream * file)**
Descripción	Exporta la cabecera del fichero.
Nombre	**Exportar_Lista_RO(ofstream * file)**
Descripción	Exporta la lista de RO.

Nombre	**Exportar_Lista_STK_Nodos(ofstream * file)**
Descripción	Exporta la lista de STK_Nodos.
Nombre	**Exportar_Lista_STK_TexMap(ofstream * file)**
Descripción	Exporta la lista de STK_TextureMap.
Nombre	**Exportar_Mesh(ofstream * file, STK_Mesh * tri)**
Descripción	Exporta cada STK_Mesh de la lista de RO.
Nombre	**Exportar_Spline(ofstream * file, STK_Spline * spl)**
Descripción	Exporta cada STK_Spline de la lista de RO.
Nombre	**Exportar_STK_Camara(ofstream * file, STK_Camara * cam)**
Descripción	Exporta cada STK_Camara de la lista de RO.
Nombre	**Exportar_STK_Luz(ofstream * file, STK_Luz * luz)**
Descripción	Exporta cada STK_Luz de la lista de RO.
Nombre	**Exportar_STK_Material(ofstream * file, STK_Material* mt)**
Descripción	Exporta cada material de la lista de RO.
Nombre	**Exportar_Matriz(Gmatrix m, ofstream * file)**
Descripción	Exporta las matrices de los objetos.
Nombre	**PropertyValue(IgameProperty * pro, ofstream * f)**
Descripción	Exporta el valor de la estructura **IgameProperty,** este valor puede ser entero, flotante, Point3 o char. Es muy usado en las cámaras, luces y materiales.
Nombre	**Exite_Mat()**
Descripción	Verifica si el material usado por el nodo ya fue creado. Posibilita que no se creen dos materiales idénticos.
Nombre	**Ordenar()**
Descripción	Ordena los RO de la siguiente forma: Luces, Cámaras, Materiales, Mesh, Spline.

Tabla 6 Descripción de la clase STK_Exporter

Nombre: RenderObject	
Tipo de clase: Entidad	
Atributo	**Tipo**

id_RO	int
RO	enum
nombre_RO	string
Para cada responsabilidad:	
Nombre	**RenderObjects()**
Descripción	Construye un objeto de tipo RO.
Nombre	**~RenderObjects()**
Descripción	Destruye un objeto de tipo RO.

Tabla 7 Descripción de la clase RenderObject

Nombre: STK_TextureMap	
Tipo de clase Entidad	
Atributo	**Tipo**
id	int
texMap	IgameTextureMap *
Para cada responsabilidad:	
Nombre	**STK_TexMap()**
Descripción	Crea un objeto de tipo STK_TextureMap.
Nombre	**~STK_TexMap()**
Descripción	Destruye un objeto de tipo STK_TextureMap.

Tabla 8 Descripcion de la clase STK_TextureMap

Nombre: STK_Formas	
Tipo de clase: Entidad	
Atributo	**Tipo**
t_Forma	enum
matriz	vector<Gmatrix>
Para cada responsabilidad:	

Nombre	**STK_Formas()**
Descripción	Construye un objeto de tipo STK_Formas.
Nombre	**~STK_Formas()**
Descripción	Destruye un objeto de tipo STK_Formas.

Tabla 9 Descripcion de la clase STK_Formas

Nombre: STK_Spline	
Tipo de clase: Entidad	
Atributo	**Tipo**
cant_Knot	int
knot	vector<IgameKnot * >
Para cada responsabilidad:	
Nombre	**STK_Spline()**
Descripción	Construye un objeto de tipo STK_Spline.
Nombre	**~STK_Spline()**
Descripción	Destruye un objeto de tipo STK_Spline.

Tabla 10 Descripcion de la clase STK_Spline

Nombre: STK_Mesh	
Tipo de clase Entidad	
Atributo	**Tipo**
mat	string
vertices	vector<Point3>
caras	vector<FaceEx*>
tex_Vertex	vector<Point2>
active_Map	Tab<int>
color	Ipoint3
ilum_Vert	vector<float>
alpha_Vert	vector<float>

color_Vert	vector<Point3>
Para cada responsabilidad:	
Nombre	**STK_Mesh()**
Descripción	Construye un objeto de tipo STK_Mesh.
Nombre	**~STK_Mesh()**
Descripción	Destruye un objeto de tipo STK_Mesh.

Tabla 11 Descripción de la clase STK_Mesh

Nombre: STK_Atributos	
Tipo de clase: Entidad	
Atributo	**Tipo**
t_Atri	enum
Para cada responsabilidad:	
Nombre	**STK_Atributos()**
Descripción	Construye un objeto de tipo STK_Atributos.
Nombre	**~STK_Atributos()**
Descripción	Destruye un objeto de tipo STK_Atributos.

Tabla 12 Descripción de la clase STK_Atributos

Nombre: STK_Material	
Tipo de clase Entidad	
Atributo	**Tipo**
nombreM	string
mat_Class	string
ambient	IgameProperty *
diffuse	IgameProperty *
emissive	IgameProperty *
specular	IgameProperty *
opacity	IgameProperty *

glossiness	IgameProperty *
specular_Level	IgameProperty *
emissive_Amount	IgameProperty *
texMap	vector<int>
c_A	Color
c_S	Color
c_D	Color
shifness	float
two_side	bool
Para cada responsabilidad:	
Nombre	**STK_Material()**
Descripción	Construye un objeto de tipo STK_Material.
Nombre	**~STK_Material()**
Descripción	Destruye un objeto de tipo STK_Material.

Tabla 13 Descripción de la clase STK_Material

Nombre: STK_Luz	
Tipo de clase Entidad	
Atributo	**Tipo**
color	IgameProperty *
multipler	IgameProperty *
ini_Attenuation	IgameProperty *
fin_Attenuation	IgameProperty *
fallOff	IgameProperty *
hotSpot	IgameProperty *
aspectRatio	IgameProperty *
decayStart	IgameProperty *
tipo_Luz	IgameLight::LightType
overShoot	int

decayType	int
shape	int
is_On	bool
es_Target	bool
exclude_nodo	vector<int>
matriz	vector<Gmatrix>
Para cada responsabilidad:	
Nombre	**STK_Luz()**
Descripción	Construye un objeto de tipo STK_Luz.
Nombre	**~STK_Luz()**
Descripción	Destruye un objeto de tipo STK_Luz.

Tabla 14 Descripción de la clase STK_Luz

Nombre: STK_Camara	
Tipo de clase Entidad	
Atributo	**Tipo**
fov	IgameProperty *
far_Clip	IgameProperty *
near_Clip	IgameProperty *
target_Dist	IgameProperty *
es_Target	bool
matriz	vector<Gmatrix>
Para cada responsabilidad:	
Nombre	**STK_Camara()**
Descripción	Construye un objeto de tipo STK_Camara free.
Nombre	**STK_Camara()**
Descripción	Construye un objeto de tipo STK_Camara target.
Nombre	**~STK_Camara()**
Descripción	Destruye un objeto de tipo STK_Camara.

Tabla 15 Descripcion de la clase STK_Camara

Nombre: STK_Nodo	
Tipo de clase Entidad	
Atributo	**Tipo**
indice	int
id_nodo	int
Indice_RO	vector<int>
Indice_Hijos	vector<string>
nombre	string
tipo	string
parent	string
matriz	vector<GMatrix>
Para cada responsabilidad:	
Nombre	**STK_Nodo()**
Descripción	Construye un objeto de tipo STK_Nodo.
Nombre	**~STK_Nodo()**
Descripción	Destruye un objeto de tipo STK_Nodo.

Tabla 16 Descripcion de la clase STK_Nodo

Nombre: STK_Arbol	
Tipo de clase Entidad	
Atributo	**Tipo**
raiz	STK_Nodo*
sub_Arbol	vector<STK_Arbol*>
Para cada responsabilidad:	
Nombre	**STK_Arbol(STK_Nodo* r)**
Descripción	Construye un objeto de tipo STK_Arbol con raíz r;
Nombre	**~STK_Arbol ()**

Descripción	Destruye un objeto de tipo STK_Arbol.
Nombre	**Add_SubArbol(STK_Arbol *a)**
Descripción	Adiciona un árbol (debido a que un nodo es un árbol que es hoja, cuando se habla de adicionar, retornar o eliminar un árbol, es lo mismo que decir que se adiciona, retorna o elimina un nodo) a la lista de subarboles.
Nombre	**Es_Hoja()**
Descripción	Determina si un nodo es hoja o no.
Nombre	**Get_Cant_Hijos()**
Descripción	Retorna la cantidad de hijos que tiene un nodo.
Nombre	**Get_SubArbol(int i)**
Descripción	Retorna el árbol de la lista de subarboles en la posición “i”.
Nombre	**Get_SubArbol(STK_Arbol *arbol,string name)**
Descripción	Retorna verdadero si existe un árbol con nombre “name”.
Nombre	**Busacar_Add_SubA()**
Descripción	Busca un árbol dado un criterio de búsqueda y adiciona un nodo a este árbol.
Nombre	**Rec_PreOrden()**
Descripción	Se hace un recorrido en PreOrden al árbol.
Nombre	**Eliminar_SubA(int p)**
Descripción	Elimina un árbol en la posición i de la lista de subarboles.
Nombre	**Get_Raiz()**
Descripción	Retorna la raíz de un árbol.
Nombre	**Vaciar()**
Descripción	Elimina todos los arboles en la lista de subarboles.
Nombre	**Optimizar(STK_Arbol * a)**
Descripción	Optimiza el árbol de acuerdo a la estructura de los nodos de la STK (v 10.07).

Tabla 17 Descripción de la clases STK_Arbol

Nombre: STKGroup
Tipo de clase Entidad

Atributo		**Tipo**
nombre		string
Nom_Mat		string
parent		IGameNode*
matriz		vector<GMatrix>
Para cada responsabilidad:		
Nombre	**STKGrupo ()**	
Descripción	Construye un objeto de tipo STKGrupo.	
Nombre	**~ STKGrupo ()**	
Descripción	Destruye un objeto de tipo STKGrupo.	

Tabla 18 Descripción de la clases STKGroup

3.3 Diagramas de secuencia

Los diagramas de secuencia se muestran en el Anexo 4.

3.4 Implementación del sistema

Todos los componentes del sistema reflejados en el diagrama de componentes (Fig. 22) serán desplegados en un solo nodo físico, dicho nodo se ve en la Fig. 21.

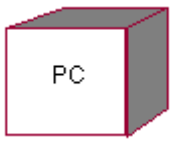

Fig. 16 Diagrama de despliegue

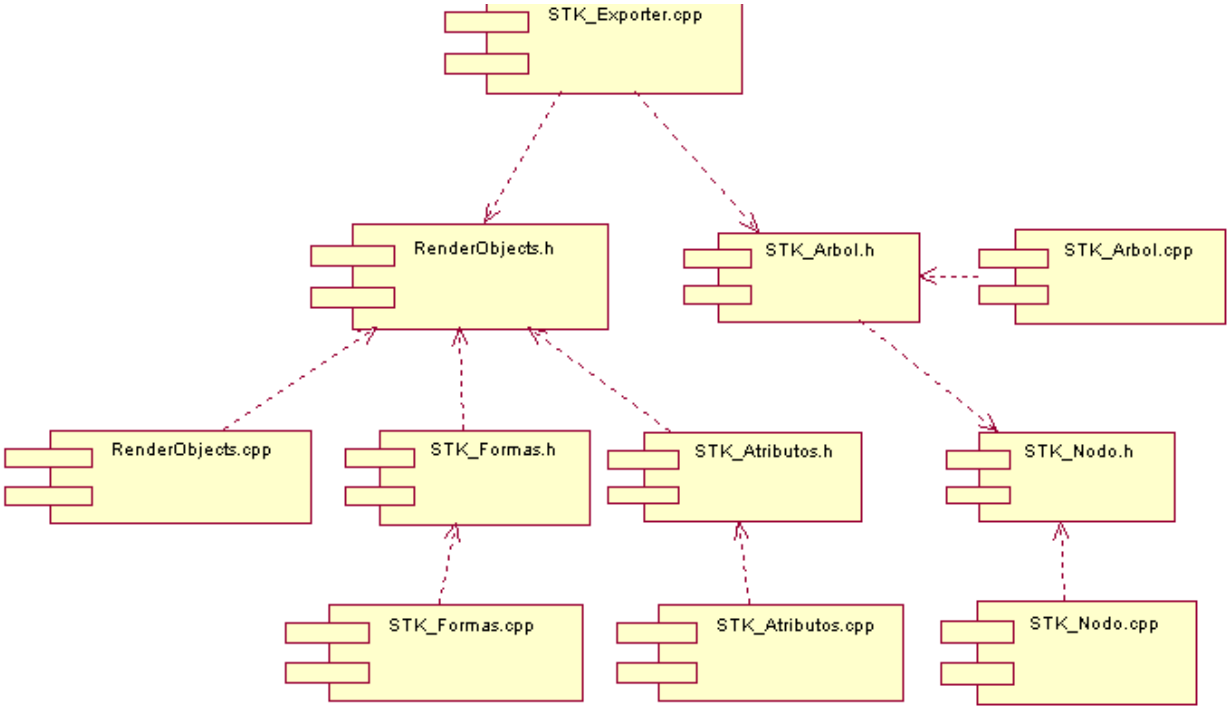

Fig. 17 Diagrama de compontes

Conclusiones del capítulo

A lo largo de este capítulo se mostraron las clases de diseño arquitectónicamente significativas, así como el diseño físico de estas clases organizados por paquetes. A partir de estas clases se obtuvo una aplicación totalmente funcional, que cumple los requisitos funcionales y no funcionales expuestos logrando exportar una escena desde 3ds Max al fichero propuesto en el Capítulo dos.

Conclusiones

Con la realización del presente trabajo se arribaron a las siguientes conclusiones:

- Para dar solución al problema planteado por el presente trabajo de diploma, se diseñó el fichero binario .STK con un formato estructurado de acuerdo a las características de la versión 10.07 de la SceneToolKit. Para su implementación, se le incorporó al 3ds Max la posibilidad de exportar escenas hacia el fichero de forma rápida y sencilla; y a la SceneToolKit se le adicionaron funciones para la carga de dicho fichero mediante el desarrollo de la clase CSTKInterpreter.

- El fichero .STK resuelve las tres deficiencias señaladas al inicio de esta investigación: se eliminó la repetición de vértices de geometrías. Tiene en cuenta otros objetos como *splines*, cámaras, luces y materiales con multitexturas, que permiten la creación de escenas complejas y realistas. Es posible crear la estructura de los nodos del grafo de escena de la versión actual de la STK, necesario para poder optimizar la visualización de las escenas virtuales.

- El formato de fichero propuesto por el presente trabajo de diploma responde mejor a la estructura actual del grafo de escena y *render* genérico de la STK, ya que la estructura y los bloques de información del mismo lo permiten. Por tal se llegó al resultado esperado permitiendo que se facilite el proceso de creación de escenas virtuales en la STK.

Recomendaciones

Con el desarrollo del presente trabajo, han surgido ideas que pueden ser implementadas en el futuro, para lograr un formato de fichero más útil y una aplicación más efectiva, por tal motivo se recomienda:

- Implementar una interfaz visual la cual permita al diseñador tomar decisiones de lo que desea exportar.

- Darle continuidad a la implementación del *plugin* permitiendo optimizar sus funcionalidades y actualización con respecto a la STK.

- Adicionar información que permita crear multimateriales y tener más de una cámara en la escena.

Referencias Bibliográficas

1. **Chávez Ayala, Dayami.** *Arquitectura de la Plataforma de Transmisión Abierta para Radio y Televisión.* Ciudad de La Habana : s.n., 2010.

2. **EcuRed.** [Online] 2012. http://www.ecured.cu/index.php/Grafo_de_escena.

3. —. [Online] 2012. http://www.ecured.cu/index.php/Renderizaci%C3%B3n.

4. **Quintana López, Alfredo and Alfonso Monteagudo, Yasmany.** *Motor de Render Genérico.* Cuidad de la Habana : Universidad de las Ciencias Informáticas, 2010. p. 110.

5. **Autodesk.** 3ds Max - 3D Modeling, Animation and Rendering Software. [Online] 2011. http://usa.autodesk.com/3ds-max/.

6. —. 3ds Max 9 MAXScript Reference Online Help. [Online] http://images.autodesk.com/adsk/files/maxscript.exe.

7. —. 3ds Max 9 SDK Online Help. [Online] http://images.autodesk.com/adsk/files/3dsmax9sdkhelp.exe.

8. Format. [Online] http://www.freesoftwaremagazine.com/articles/focus_format_history/.

9. The StL Format: Standard Data Format for Fabbers. [Online] 2011. http://www.ennex.com/~fabbers/StL.asp.

10. ASE File Format. [Online] 2012. http://users.polytech.unice.fr/~buffa/cours/synthese_image/DOCS/Tutoriaux/Nate/ase.html.

11. **Engel., Wolfgang F.** *Beginning Direct 3D Game Programming.* USA : Premier Press, 2003, Capitulo 11: Working whith files.

12. STL 2.0 May Replace Old, Limited File Format. [Online] http://www.rapidtoday.com/stl-file-format.html.

13. **Pipho, Evan.** Capítulo 3. *Focus On 3D Models.* USA : Premier Press, 2003, Capítulo 3: Quake ll`s MD2 Models.

14. —. Capítulo 9:. *Focus On 3D Models.* USA : Premier Press, 2003, Capítulo 9: Enter the Quake: Quake lll`s MD3 Format.

15. —. Capítulo 7:. *Focus On 3D Models.* USA : Premier Press, 2003, Capítulo 7: The 3ds Models.

16. API, DirectX y OpenGL. [Online] http://www.mailxmail.com/curso-videojuegos/api-directx-opengl.

17. **ASTLE, Dave and HAWKING, Kevin.** *Beginning OpenGL Game Programming.* USA : Prima Tech Publishing, 2001.

18. **Döllner, Jürge and Hinrichs, Klaus.** *A GenericRenderingSystem.* 2002. Vol. Vol.8.

Diccionarios usados:

- www.softzone.es/glosario/

- www.rae.es/rae.html

- www.glossary.com/

- www.absoluteastronomy.com/

- www.websters-online-dictionary.org

Glosario de términos.

API gráfica: Software para proveer una lógica para el procesamiento de vértices y píxeles.

Escena virtual: (Entorno virtual) Es un mundo creado virtualmente que puede contener desde un único objeto 3d hasta miles de ellos.

Fichero con información gráfica: Es un contenedor de información de un objeto 3d o una escena virtual que puede ser manipulada de forma unitaria, por lo general estos ficheros se salvan de forma binaria o ASCII (Texto plano).

Formato de fichero: Son las caracterizas que definen un fichero como la estructura, la forma de salvado y la extensión.

Herramienta gráfica: (Motor gráfico) Módulo gráfico independiente lo bastante potente para poder tratar toda la información que hay en él, así como su visualización en tiempo real. Este concepto surge por la complejidad gráfica añadida a la hora de tratar con escenarios 3D.

MaxScript: Es el lenguaje de secuencias de comandos integrado a Autodesk para la creación de *plugins*.

Motor de *render*: Es un software especializado destinado a hacerse cargo de los cálculos necesarios para generar una imagen de un modelo o escena 3D.

***Plugin*:** Aplicación que se relaciona con otra para aportarle una nueva funcionalidad y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúa por medio de la Interfaz Programada de la Aplicación.

***Render*:** (Renderización) Es el proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en Software de diseño en 3D.

RenderObject: Es la definición que agrupa todos los objetos que conforman una escena virtual en la STK.

SDK: (3ds Max SDK (C++)) Es el Software Development Kit (SDK) desarrollado por Autodesk para el desarrollo de *plugins* usando Visual Studio.

SceneToolKit: (STK) Formada por *scene*, “escena”, y *toolkit*, "paquete de herramientas". El nombre completo es "Herramientas para Desarrollo de Sistemas de Realidad Virtual", y debe constar del *game engine* y otras herramientas utilitarias.

STK_Camara: Son los objetos de tipo cámara creados para exportar al fichero, se definen así para diferenciarlos de los objetos cámara del 3ds Max.

STK_Material: Son los materiales creados para exportar al fichero, se definen así para diferenciarlos de los materiales del 3ds Max.

STK_Mesh: Son los objetos con malla creados para exportar al fichero, se definen así para diferenciarlos de los objetos con malla del 3ds Max.

STK_Nodo: Son los nodos del grafo de escena creado para exportar al fichero, se definen así para diferenciarlos de los nodos del 3ds Max.

STK_Spline: Son los *Spline* creados para exportar el fichero, se definen así para diferenciarlos de los *Spline* del 3ds Max.

STK_TextureMap: son las texturas con todas sus propiedades usadas por los STK_Materiales de una escena exportada al fichero. Se definen así para diferenciarlas de las TextureMap del 3ds Max.

Videojuego: Según la Real Academia Española es un dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor o de un ordenador.

Anexos 1

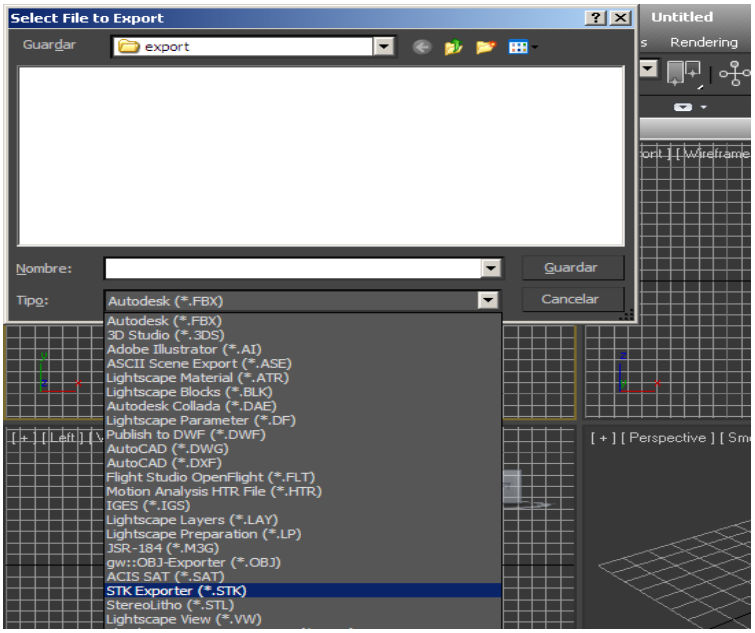

Fig. 1 Opción para exportar al fichero STK en el 3ds Max

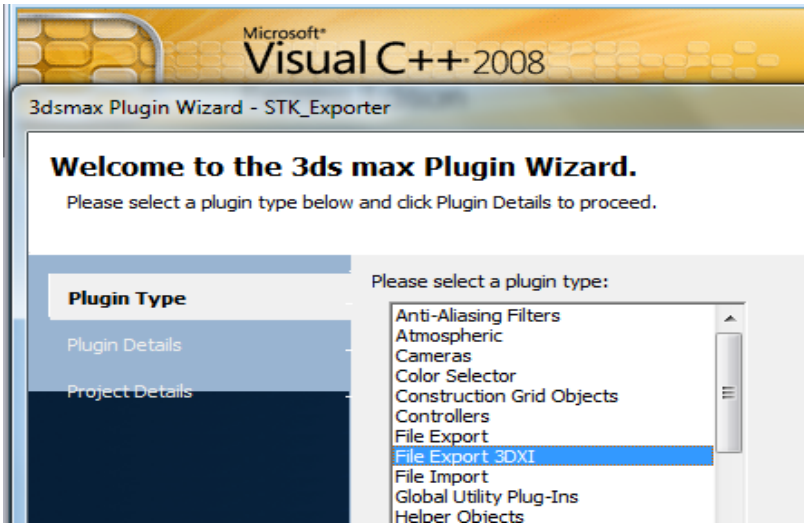

Fig. 2 Asistente para crear proyectos para 3ds Max en Visual Studio

Anexos 2

Se generó un fichero exportando una escena desde el 3ds Max (Fig. 3 Anexo 2) y posteriormente se cargó en un demo usando la STK (Fig. 4 Anexo 2).

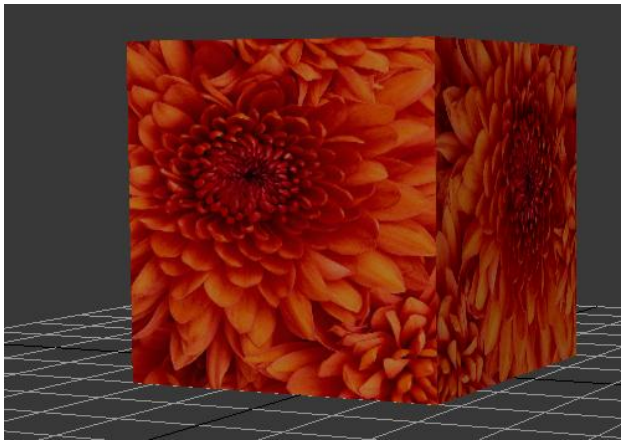

Fig. 3 Escena creada en el 3ds Max

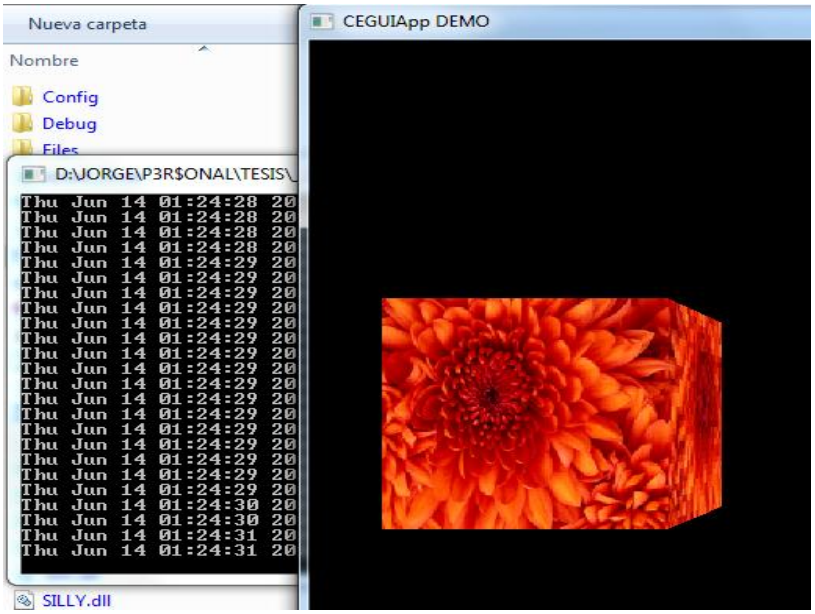

Fig. 4 Escena cargada en el Demo