UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
CENTRO DE INFORMÁTICA INDUSTRIAL

Interpretación de ficheros 3D para la biblioteca gráfica de visión estereoscópica

Tesis presentada para optar por el Título de Ingeniería en Ciencias Informáticas

Autor: **Osmany Nuñez Hernandez**

Tutor: **MSc. Yoander Cabrera Díaz**

Co-tutor: **MSc. Liudmila Pupo Peña**

Ciudad de la Habana, Mayo 2011

Agradecimientos

Quisiera agradecer a mis padres, que sin su apoyo, amor y comprensión, alcanzar esta meta hubiese sido más difícil, en especial a mi mamá por ser un ejemplo a seguir durante toda mi vida de estudiante. Agradecer a mi hermano por siempre estar ahí brindándome su apoyo en todo momento. A toda mi familia que de una forma u otra han contribuido a mi educación como persona y han incentivado en mí el deseo de ser alguien útil en la vida.

A mis profesores desde que comencé a leer y a escribir porque sin ellos nunca hubiese podido realizar este gran sueño de ser ingeniero en ciencias informáticas, hasta los que hoy me están formando como profesional.

A mis compañeros desde la primaria hasta la universidad, que de cada uno e aprendido a ser una mejor persona, en especial a mis compañeros de estos últimos cinco años de carrera que se han convertido en una gran familia, Yasmany, Luis Guillermo, Leonel, Daniel, Roiley, Fumero, Suleika y a todos aquellos que de una forma u otra ayudaron a ser mi estancia en la universidad mas acogedora.

A mi tutor por apoyarme en todo lo que ha hecho falta para que este trabajo quede con la calidad requerida.

A nuestro comandante en jefe por ser el faro que ilumina nuestro país, gracias a él y a la revolución he podido estudiar esta gran carrera.

Dedicatoria

A mi mamá, a mi papá y a mi hermano

A toda mi familia

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser autor de la presente tesis y se le reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Firma del autor

Resumen.

En este trabajo se desarrolla un módulo de interpretación de ficheros 3D sobre una biblioteca de visión estereoscópica y realidad virtual. La misma tiene distintos campos de aplicación, lo que hace necesario la implementación de esta funcionalidad. En primer lugar y como característica singular y más importante, la biblioteca es utilizada en aplicaciones de evaluación y entrenamiento de funciones visuales como la agudeza y la visión binocular.

El presente trabajo consiste en elaborar la funcionalidad de importar modelos 3D en un formato de archivo seleccionado, para contribuir a mejorar la calidad visual y minimización del tiempo de desarrollo en la construcción de escenas complejas que demanden alto grado de realismo virtual.

En aras de cumplir con los objetivos en esta investigación se hará un análisis de las diferentes características de los archivos gráficos 3D más usados en la actualidad en motores gráficos. Se establecerán comparaciones entre los modelos, atendiendo a parámetros de interés para la obtención de una solución que satisfaga las necesidades de cuya biblioteca. Sin descartar que los aportes realizados por el autor a la herramienta contribuyen a que la misma posibilite una amplia utilización en diversas ramas de la ciencia, la docencia y la investigación.

Palabras clave

Biblioteca, Ficheros 3D, Módulo, Motores gráficos, Realidad virtual

Índice general

Índice de figuras

Índice de tablas

Introducción

En la actualidad el mundo moderno presenta un nuevo fenómeno causado por las consecuencias de la última revolución industrial, devenido por el desarrollo de las telecomunicaciones y la informática, la realidad virtual (RV) es uno de los paradigmas del amplio mundo de la informatización. La necesidad de crear sistemas encaminados a garantizar el funcionamiento de la economía cubana principalmente en el campo de la salud, constituye un reto.

El término realidad virtual se asocia a todo lo relacionado con imágenes en tres dimensiones generadas por computadora y con la interacción de los usuarios en un ambiente gráfico. Ello supone la existencia de un complejo sistema electrónico para proyectar espacios visuales en 3D y para enviar y recibir señales con información sobre la actuación del usuario, quien, con un sistema de este tipo, puede sentir que se encuentra inmerso en un mundo virtual. [José R. Hilera, 1999]

Una biblioteca gráfica es la parte de un programa que controla, gestiona y actualiza los gráficos 3D en tiempo real, facilitando el desarrollo de juegos y aplicaciones sin necesidad de enfrentarse directamente al lenguaje de programación utilizado en cuestión.[Argente, 2006]

Para el desarrollo de aplicaciones de RV en la actualidad existen muchas bibliotecas gráficas de alto nivel como son OGRE 3D, OpenSceneGraph(OSG) y CrystalSpace, todas ellas hacen uso de modelos 3D para minimizar el tiempo de desarrollo en la construcción de sistemas.[OGR, 2010]

El uso de modelos 3D cada vez más se hace necesario en aplicaciones que visualizan escenas complejas con un alto grado de realismo virtual, visibles en campos como la educación, salud, deporte, cine, entre otros.

Con la previa y actual creación de la biblioteca gráfica de visión estereoscópica(del inglés Graphics Library for Stereoscopy Vision) (GLSVe) en el Centro de Informática Industrial (CEDIN), específicamente en la facultad 5, de la Universidad de las Ciencias Informáticas (UCI) de Cuba en colaboración con la universidad de Oviedo de España. La misma es utilizada para la creación de aplicaciones de visión estereoscópica. Actualmente esta biblioteca es deficiente en la construcción de escenas complejas debido a que solo cuenta con modelos geométricos básicos, lo cual constituye tardanza en el tiempo de desarrollo de aplicaciones que demandan alto grado de realismo virtual y la calidad visual en la escena.

Partiendo de la problemática planteada surge el siguiente **problema científico** ¿Cómo incorporar modelos 3D a la GLSVe?.

El **objeto de estudio** de la investigación para dar solución al problema planteado, se centra en la interpretación de ficheros 3D para sistemas de realidad virtual. El **campo de acción** se enfoca en la interpretación de ficheros 3D en la GLSVe.

Con el propósito de brindarle solución al problema, se plantea como **objetivo general**, desarrollar un módulo en la GLSVe que permita interpretar ficheros 3D para aumentar la eficiencia en las aplicaciones.

Ideas a defender

Con el desarrollo del módulo de interpretación de ficheros 3D se logrará proveer a la GLSVe de objetos 3D y escenas virtuales creadas en herramientas de diseño. Se minimizará el tiempo de desarrollo de aplicaciones utilizando dicha biblioteca y se logrará un mayor realismo en escenas virtuales.

Estructura del documento

El presente documento se encuentra estructurado en Introducción, Capítulos, Conclusiones y Recomendaciones. A continuación se hace una breve descripción del contenido de los cuatro capítulos.

Capítulo 1: "Fundamentación teórica", el presente capítulo tiene como objetivo estudiar diferentes formatos de ficheros 3D, analizando sus características, ventajas y desventajas de su uso en el desarrollo de aplicaciones de RV. De cada uno de hará una descripción detallada de su estructura, composición y recursos con los que cuenta.

Capítulo 2: "Solución propuesta", en este capitulo se hará una descripción detallada de la solución propuesta al problema, primeramente se hace una comparación general de los ficheros estudiados y se seleccionan los ficheros a interpretar. Se hace una breve presentación de la biblioteca GLSVe y se describen las funcionalidades incorporadas en ella. Además, se documentan las técnicas de solución empleadas para dar cumplimiento a los objetivos.

Capítulo 3: "Diseño de la solución", en este capítulo se define una visión más detallada y específica del sistema que se va a desarrollar, tomando como base la propuesta de solución planteada en el capítulo anterior. Se creará el modelo del dominio, haciendo una explicación de los conceptos presentes en este durante el desarrollo del glosario de términos. También se determinan las capacidades o funciones que el sistema debe cumplir (requisitos funcionales) y las propiedades o cualidades que el producto debe tener (requisitos no funcionales) y se describen los procesos en forma de casos de uso que dan cumplimiento a los requisitos funcionales.

Capítulo 4: "Análisis de resultados", este capitulo tiene como objetivo analizar los resultados obtenidos luego de la ejecución de pruebas a la aplicación de demostración del módulo.

Capítulo 1

Fundamentación Teórica

1.1. Introducción

El presente capítulo tiene como objetivo estudiar diferentes formatos de ficheros 3D, analizando sus características, ventajas y desventajas de su uso en el desarrollo de aplicaciones de RV. De cada uno de hará una descripción detallada de su estructura, composición y recursos con los que cuenta.

1.2. Los ficheros 3D

Un fichero gráfico 3D es un contenedor de información de una escena, ya sea de un único objeto o de un mundo virtual complejo, que debe contar con todos los atributos que permitan conocer la estructurada de la malla, como son los vértices y caras, vea la figura 1.1, en el caso más básico y además suelen tener un conjunto de características que brindan un mayor nivel de detalle a las escenas, entre los cuales se puede mencionar, los materiales, los colores de vértices y las normales de las caras. [Wendy García López, 2007]

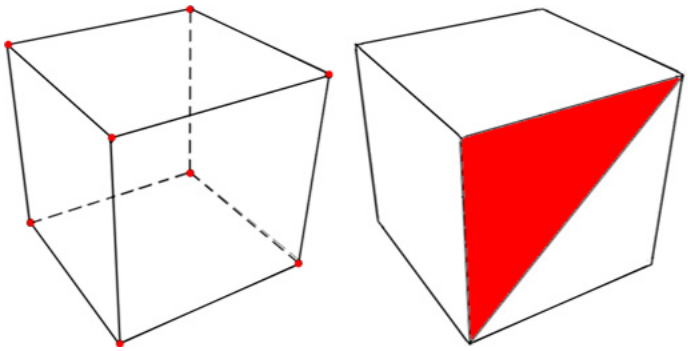

Figura 1.1: Ficheros 3D

Por mucho tiempo a la hora de recrear un entorno virtual uno de los grandes problemas ha sido la forma de generar y cargar archivos gráficos que cumplan con los requisitos necesarios, ya sea para disminuir el tamaño de los archivos o para facilitar la carga y puesta en escena de los mismos. En la actualidad se utilizan diversos formatos para el almacenamiento de la información en ficheros, entre los que más se utilizan se encuentran el ASCII y el Binario.

Hoy en día existen muchas herramientas para el diseño gráfico, tales como 3DMax, Malla y Blender, todos con características muy similares en cuanto a modelación, texturizado, animación, etc. Muchas de estas herramientas están bajo licencias de uso pero otras no, como es el caso de Blender, por lo que se toma especial énfasis a la hora de seleccionar los ficheros a tratar.

Todos los ficheros 3D están conformados por bloques y casi todos presentan una cabecera que indica donde localizar estos bloques y en qué orden se encuentran.

El sistema de ficheros de las bibliotecas gráficas es muy variado, muchos de los más utilizados se muestran a continuación:

1.2.1. Collada

Collada (DAE) creado y administrado por el consorcio Grupo Khronos. Define un estándar abierto de esquema XML.[Khronos Group, 2008] Cuenta con una geometría de malla bien detallada con una descripción flexible de los objetos complejos, permite transformaciones jerárquicas (rotación, traslación, matriz de escala entre otros), tiene información completa de materiales, texturas, sombreados, luces, cámaras, animaciones, física(cuerpos rígidos, limitaciones, volúmenes de colisión).

La física fue agregada al estándar de COLLADA para permitir que los creadores puedan definir varias cualidades físicas en escenas visuales. Por ejemplo, se puede definir características tales como fricción, gravedad, etc. En este trabajo no se hace uso de parámetros de la física, solo se especifica para posteriores actualizaciones.

Es un formato de fichero nuevo, está plenamente documentado y especificado, es libre de derechos, altamente extensible y es exportado por herramientas de código abierto. [Group, 2008]

$< COLLADA >$

$< library\_cameras >$

Se definen las características de la(s) cámaras de la escena.

$< library\_effects >$

Se definen efectos adicionales tales como transparencia, colores, brillo entre otros.

$< library\_lights >$

Se definen los parámetros de luces.

$< library\_materials >$

Se definen los materiales utilizados en la escena.

$< library\_geometries >$

Se definen todos los parámetros de la geometría, vértices, normales, coordenadas de textura y caras.

$< library\_visual_scenes >$

Se definen parámetros como rotación, escala o traslación de todos los elementos de la escena.

$< library\_physics_{m}odels >$

Se definen parámetros de física de los elementos de la escena tales como masa, dinámica, entre otros.

$< library\_physics_{s}cenes >$

$< /COLLADA >$

[Khronos Group, 2008]

Este formato puede ser exportado por Blender, 3D Studio Max instalando el plugin ColladaMax y Maya con el plugin ColladaMaya. Muchos de los motores gráficos utilizan este formato para desarrollar sus aplicaciones como es el caso de *irrlicht engine*, *Unreal Engine*, *Torque 3D*, entre otros.

1.2.2. 3DS

El fichero 3D-Studio (3DS) es utilizado por el AutoDesk 3D Studio MAX de modelado, rendering y animación de paquete en el PC. Se encuentra en código binario y presenta una estructura muy completa.

El fichero 3DS esta dividido en bloques y cada bloque está dividido en sub-bloques, para de esta forma organizar los objetos con sus características. No tiene un orden en el fichero, por lo que a simple vista es muy difícil entender su estructura. El formato en su estructura general es como se muestra en la figura 1.2.

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.

3DS es un formato que optimiza muy bien el tamaño de los ficheros ya que se encuentra en código binario. Exporta escenas completas, es decir que un mismo fichero puede contener varias mallas. Como única desventaja tiene, que al no mostrar un orden en los bloques de datos y la estructura, se hace complejo y difícil de interpretar.

Este formato puede ser exportado por: Blender y 3DMax Studio. Mucho de los motores gráficos hacen uso de este formato para representar modelos en escenas virtuales, como: Cristal Space, Fly3D, Genesis3D, Irrlicht.

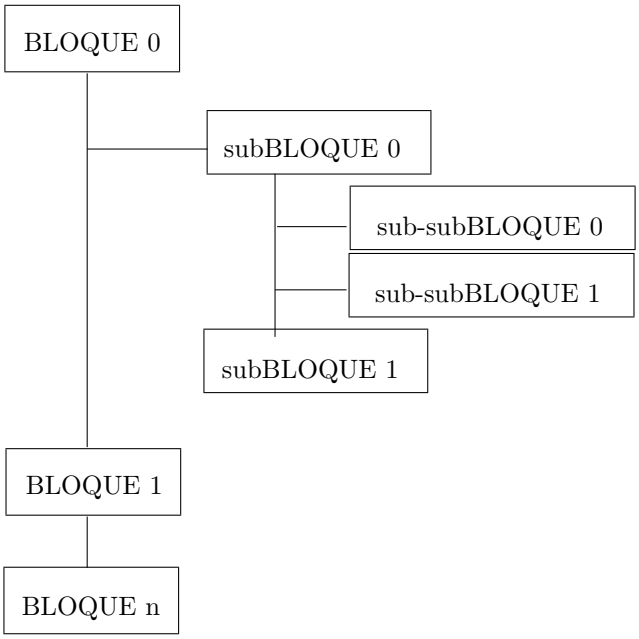

Figura 1.2: Estructura general del 3ds[Evan Pipho, 2003]

1.2.3. OBJ

Es un formato de archivo de texto, lo que significa que se puede editar archivos OBJ en cualquier editor de texto. Presenta una estructura sencilla y bien organizada. El primer carácter de cada línea especifica el tipo de comando.

v xyz Especifica un vértice con sus tres coordenadas. El vértice está implícitamente nombrado por el orden en que se encuentra en el archivo. Por ejemplo, el primer vértice en el archivo se hace referencia como 1, el segundo como 2 y así sucesivamente. Ninguno de los comandos vértice en realidad especifican cualquier geometría, no son más que puntos en el espacio.

Ejemplo: v 0.00 0.00 0.00

vn xyz Especifica un vector normal. Muchas veces éstos no se utilizan, ya que la **f** usará el orden de la **v** se dan órdenes para determinar el lugar normal. Al igual que el comandos **vt** , no quieren decir nada hasta que agrupa con un vértice en un comando **f**.

Ejemplo: vn 0.00 1.00 0.00

vt xyz El comando especifica el vértice de coordenadas de textura UV (y, opcionalmente, W) de mapeo. Estos serán los valores de punto flotante entre 0 y 1, que dice cómo asignar la textura. Realmente no le dicen nada por sí mismos, deberán agruparse con un vértice en un comando **f**.

Ejemplo: vt 1.00 0.00 0.00

g nombre El comando especifica el nombre del grupo una sub-agrupación de objetos. Todos los comandos **f** que siguen se consideran en el mismo grupo.

Ejemplo: g caja

usemtl El comando, utilice material le permite nombrar a un material a utilizar. Todos los comandos **f** siguen la voluntad de utilizar el mismo material, hasta que otro comando **usemtl** se encuentra.

Ejemplo: usemtl 01_Default

f [vértice1, vértice2, vértice3] El comando especifica la cara de un polígono a partir de tres vértices de la lista, para cada vértice, también puede haber un vt asociados, que dice cómo asignar la textura en este punto, y / o una vn, que especifica una normal en este punto. Puede estar compuesto por tres o cuatro componentes según el tipo de cara (triángulo o cuadrilátero).

Ejemplo: f 1/10/1 3/12/3 4/11/4

s nombre de grupo El nombre de grupo se utiliza para hacer grupos separados. Todos los posteriores comandos **f** están en el grupo de suavizado mismo hasta que otro comando s se encuentra.

Ejemplo: s 2

OBJ es un fichero de fácil manejo ya que tiene una estructura sencilla. Un fichero puede almacenas varios objetos y tratar a cada uno por separado. Tiene como desventaja que solo se centra en trabajo con objetos estáticos ya que no exporta matriz de transformación,

bounding box, colores de vértices, entre otros. Este formato puede ser exportado por 3D Max Studio, Maya, Blender, entre otros.

Motores gráficos como Irrlicht y cristal space hacen uso de este formato para el desarrollo de aplicaciones de realidad virtual.

1.2.4. ASE

ASCII Scene Exporter (ASE) tiene una buena organización de los atributos de un objeto, todo manejado en forma de clases. Contiene todos los atributos necesarios para conformar un buen fichero, tales como; cantidad de materiales, colores de vértices, normales, grupos de suavizado, entre otros. En un fichero contiene más de una malla, por lo que se puede exportar en uno solo escenas completas.[ASE, 2010]

El formato es basado en un identificador que siempre comienza por el carácter (*), ejemplo: Nombre_del_atributo

Dentro del bloque (*SCENE*) se encuentran todas los objetos y materiales, y cada uno de estos bloques esta divididos en sub-bloques, lo cual hace una organización jerárquica de los objetos en el fichero. Contiene todo respecto a una escena, cuadro en que comienza la animación, cuadro en que termina, velocidad de la animación, color del ambiente, entre otros.

El bloque de materiales se representa con el identificador (*MATERIAL_LIST*) donde aparecen todos los materiales utilizados en la escena, mostrando de cada uno de ellos un identificador y la dirección física de la imagen.

El bloque (*GEOMOBJECT*) contiene los parámetros de mallas de la escena, este está compuesto por una o más sentencias (*MESH*), las cuales representan cada una a una malla en especifico. Los parámetros de cada malla están representados por listas de vértices, normales, texturas y caras; (*MESH_VERTEX_LIST*), (*MESH_TVERTLIST*), (*MESH_NORMALS*) y (*MESH_TFACELIST*) respectivamente.
La lista de vértices contiene toda la información correspondiente a los vértices de posición del modelo. La lista de normales contiene los vértices normales para cada vértice de posición. La lista de texturas contiene las coordenadas de textura para cada vértice de posición. La lista de caras es la relación entre los vértices de posición, vértices normales y coordenadas de textura, donde cada cara esta formada por tres de cada uno de estos

vértices.

ASE es un formato de fichero con una buena organización de los atributos de los objetos. Tiene la desventaja que normalmente los ficheros son muy grandes. Puede ser exportado por 3D Max Studio. El motor gráfico Cristal Space hace uso de este formato de fichero 3D.

1.2.5. STL

Formato del interfaz de estereolitografía (STL), se utiliza para definir los sólidos 3D para ser captado por los sistemas de litografía [1] en 3D. La versión ASCII de este formato es conocido como STLA y tiene una estructura muy simple. No contiene información de textura y puede encontrarse en formato ASCII y binario.

Estructura del archivo

---

solid "nombre": nombre del objeto.
- **facet normal float float float:** normales de la cara.
  - **outer loop**: inicialización de los vértices de la cara.
    - **vertex float float float**: coordenadas del vértice en "x, y, z".
    - **vertex float float float**
    - **vertex float float float**
  - **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.

Los parámetros en negritas se pueden repetir según la cantidad de caras del objeto que represente.[Wendy García López, 2007]

Como se puede apreciar la estructura del formato STL es muy sencilla, solo cuenta con la información de caras y sus vertices correspondientes por lo que en un fichero esta contenido un solo 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. Este fichero puede ser exportado por 3D Max

[1] La litografía es un procedimiento de impresión mediante el cual se reproduce sobre papel la escritura o el dibujo, realizado con tinta especial o lápiz graso, sobre la superficie de una piedra calcárea, de estructura especial, muy compacta y homogénea.

Studio y Blender. Este tipo de formato es soportado por el motor gráfico Irrlich.

1.2.6. DirectX

El DirectX fue construido para Direct3D y expandido para DirectX 6.0, a partir de ese momento ha ido evolucionando debido a su capacidad de expandirse.

Presenta una estructura jerárquica que permite insertar más plantillas y así expandir el fichero con nuevos bloques de información. Véase figura 1.3

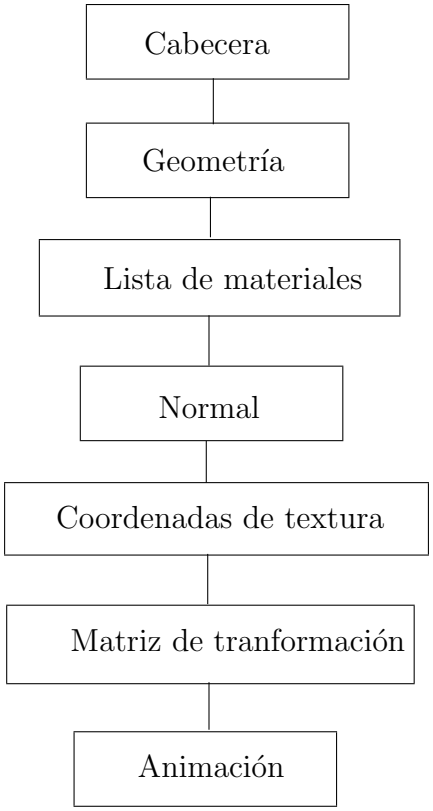

Figura 1.3: Estructura de Direct X [Wendy García López, 2007]

Descripción de los bloques

Cabecera
Contiene información acerca de la versión del exportador, el formato de almacenamiento y el tamaño del fichero en byte.

Geometría
Define cómo está estructurada la malla, brindando atributos como son los vértices y las caras.

Lista de materiales
Define los materiales que se usan en la escena y la relación que existe con los vértices y caras de las geometrías.

Normales
Permite definir los vectores normales de las geometrías.

Coordenadas de Texturas
Están definidos los caminos de las texturas y las coordenadas de mapeo para cada vértice y cara.

Matriz de transformación
Definen la matriz de transformación de cada geometría en el fichero.

Animación
Permite definir los parámetros de animación en caso de que existan en el modelo [Wendy García López, 2007]

El fichero DirectX puede encontrarse tanto en formato ASCII como en binario y ambos optimizan el espacio en memoria. Es exportado por Blender, por 3D Studio y Maya a través de plugins. Motores gráficos como OpenSceneGraph hacen uso de este formato de fichero.

1.2.7. X3D

Este es un estándar que puede ser soportado fácilmente por herramientas de creación, navegadores propietarios, y otras aplicaciones 3D, es extensible, lo que significa que X3D puede emplearse para hacer una pequeña y eficiente animación 3D, o puede usarse para soportar lo ultimo en extensiones streaming o de renderizado. Contiene información de textura. Puede ser utilizado a través de dispositivos de hardware y en una amplia gama de aplicaciones CAD, simulación visual, visualización médica, SIG, entretenimiento, educación, y presentaciones multimedia.

El fichero X3D presenta un clásico formato XML encabezado con la etiqueta (*3DS*), esta subdividido en bloques y sub-bloques, el archivo esta estructurado a partir de las siguientes etiquetas:

(*HEADER*): Representa la cabestra de la escena la cual contiene información de nombre de la escena, autor, entre otros.

(*SCENE*): Contiene todo los parámetros referentes a las mallas, materiales y transformaciones.

(*SHAPE*): Cada etiqueta *shape* representa un objeto dentro de la escena.

(*MATERIAL*): Con esta etiqueta se definen los materiales y texturas utilizados en los objetos dentro de el fichero.

(*COORDINATE*): Describe las coordenadas de textura de los objetos.

(*INDEXED_FACED_SED*): Representa la lista de posiciones de vértices que describen las caras del modelo en cuestión.

Conclusiones

En el presente capítulo se realizo un estudio de algunos de los formatos de ficheros mas usados en la actualidad para el desarrollo de aplicaciones de RV. Se logró entender el funcionamiento básico de las estructuras de los distintos formatos de ficheros analizados y se definieron las principales ventajas y desventajas que presentan cada uno de ellos.

Capítulo 2

Solución Propuesta

2.1. Introducción

En este capitulo se hará una descripción detallada de la solución propuesta al problema, primeramente se hace una comparación general de los ficheros estudiados y se seleccionan los ficheros a interpretar. Se hace una breve presentación de la biblioteca GLSVe y se describen las funcionalidades incorporadas en ella. Además, se documentan las técnicas de solución empleadas para dar cumplimiento a los objetivos.

2.2. Comparación de los ficheros estudiados

En este epígrafe se hace una comparación general de los ficheros expuestos en la fundamentación teórica, de acuerdo a una serie de aspectos importantes para cumplir los objetivos de este trabajo, la comparación se realiza mediante la siguiente tabla 2.1. La x marca los parámetros positivos.

Ficheros/Caracteristicas	3DS	DAE	DirectX	X3D	OBJ	ASE	STL
Información dinámica de textura	x	x	x	x	x	x	
Exportado x herramientas libres	x	x	x	x	x		x
Contenedor de varias mallas	x	x	x	x	x	x	
Información de animación	x	x	x	x			
Parámetros de luces	x	x	x	x		x	
Matriz de transformación	x	x	x	x		x	

Tabla 2.1: Comparación general de ficheros

De acuerdo con la información obtenida en toda la bibliografía consultada, se puede hacer una elección del fichero o de los ficheros a interpretar. Tanto el formato 3DS como Collada(DAE) cuentan con una información muy completa con la única diferencia que 3DS se encuentra en código binario, pero el fichero Collada al presentar una estructura XML se hace mas idóneo para el trabajo en el lenguaje propuesto.

Los formatos DirectX y X3D son pocos usados por las bibliotecas gráficas aunque presentan casi las mismas características que el 3DS y Collada, en el caso de X3D, esta creado para el trabajo en la web y aunque este tiene un gran avance con respecto a los gráficos por computadora todavía no hay evidencias de su uso en bibliotecas gráficas.

Los formatos OBJ, ASE y STL aunque no cuenten con la misma cantidad de información que los demás, no dejan de ser usados, al contrario, muchas de las bibliotecas gráficas mas utilizadas hacen uso de estos formatos.

Como característica importante a la hora de seleccionar los ficheros, se tiene, ser exportado por herramientas libres, en este caso Blender. Otra característica fundamental, que el fichero contenga información de textura y que sea contenedor de varias mallas, lo que quiere decir que en un único fichero se puedan definir diversos objetos separados entre si.

Para cumplir con los objetivos propuestos se decide la interpretación de cuatro ficheros, uno complejo, en este caso Collada y tres ficheros adicionales a modo de enriquecer las opciones de interpretación, en este caso, OBJ, ASE y STL.

2.3. Descripción del problema

En el capítulo anterior se justifica la necesidad de un sistema capas de interpretar ficheros 3D en la biblioteca GLSVe. La biblioteca GLSVe todavía no tenía incorporada funcionalidades para interpretar ningún tipo de fichero 3D, por lo que el desarrollo de aplicaciones con escenarios complejos en la biblioteca se hacia bastante engorroso, lento y con un bajo rendimiento.

2.4. Graphics Library for Stereoscopic Vision (GLSVe)

GLSVe es una biblioteca estructurada en clases que permite representar objetos a distintas profundidades, que se observen por delante del monitor, en este, o detrás, según los paralajes correspondientes, brinda una interfaz fácil e intuitiva para su uso y tiene las siguientes características:

- La visualización de la escena puede ser en modo monoscópico.
- La visualización de la escena puede ser en modo estereoscópico según las técnicas: visión paralela, visión cruzada, anaglifo, polarización y obturación (en los formatos de representación entrelazado horizontal, alternated images y quad buffering).
- Brinda un puntero 3D, útil para la selección y manipulación de los objetos de la escena a distintas profundidades.
- Brinda sonido 2D para motivar al usuario con música de fondo y sonidos de eventos en la realización de actividades.
- Permite sonido 3D para ayudar al usuario moverse en la escena a través del puntero 3D logrando mayor nivel de inmersión y comprensión del ambiente.
- Permite sonido 3D para ambientes de realidad virtual con seguimiento posicional.
- Una vez compilada la biblioteca es posible añadir nuevas primitivas gráficas.

- Permite realizar transformaciones de pan, zoom y parallax.
- El usuario puede gestionar las funciones implementadas para hacer su propio código.

La biblioteca ha sido desarrollada en el lenguaje C#, utilizando el recubrimiento Tao Framework para emplear las herramientas OpenGL, las cuales brindan facilidades para implementar las técnicas estereoscópicas.

2.5. Solución Técnica

Para dar cumplimiento al objetivo de este trabajo, interpretar ficheros 3D, se hizo uso de técnicas de programación orientadas a objetos.

2.5.1. Interpretación de formato Collada (DAE)

El fichero Collada esta dividido en bloques y sub-bloques, cada uno de ellos con un propósito bien definido, los bloques que lo conforman son, geometría, luces, materiales, cámaras, física, efectos de escenas, entre otros.

Geometrías

Este bloque contiene toda la información respecto a las geometrías presentes en una escena, tales como vértices de pocisión, normales, de textura y caras. Este puede contener varias mallas independientes.

El bloque comienza con la etiqueta (*library_geometries*) y a partir de aquí se va detallando cada parámetro de la o las mallas contenidas en el fichero. La información de vértices de posición y vértices normales se muestran en una lista de la forma (x y z x y z ....) lo que quieres decir que cada trio conforman un vértice. La lista de coordenadas de textura es similar de la forma (u v u v ...) donde cada par representa la coordenada de textura de cada vector posición. Las caras también están representadas con una lista de la forma (x nx uv y ny uv z nz uv ...) donde cada valor representa una posición en las listas de vértices de posición, normal y de textura antes mencionadas.

Textura

El bloque de texturas define las imagenes a utilizar para el texturizado, puede estar contenido por una o varias imagenes, mostrando de cada una de ellas la dirección en que

se encuentran.

El bloque comienza con la etiqueta (*library_images*) y a continuación se especifican las direcciones físicas de las imagenes utilizadas.

Transformaciones de escena

El bloque transformaciones de escena esta subdividido en nodos los cuales refieren a cada elemento en la escena ya sean mallas, luces o cámaras.

Este comienza con la etiqueta (*library_visual_scenes*), con la etiqueta (*node*) y un identificador del elemento al que se hace referencia se especifican parámetros tales como traslación (*translate*), rotación (*rotate*) y escala (*scale*), cada uno de ellos define un vector el cual modifica al elemento en cuestión.

Cargar DAE

Haciendo uso de bibliotecas adicionales tales como (*System.Xml*). Primeramente creando un objeto de tipo (*XmlTextReader*) y a partir de este se utilizan funciones como:

(*Read*, Leer): Función para crear un ciclo dentro del fichero, se mueve a través de la secuencia de nodos.

(*MoveToElement*, Mover a un elemento): Función que permite moverse nodo a nodo.

(*MoveToAttribute*, mover a un atributo): Función que permite moverse por los atributos que contienen cada nodo en el fichero.

(*ReadString*, leer cadena): Lee el contenido de un nodo de texto o un elemento como cadena.

(*Value*, valor): Función que permite obtener el valor de un atributo.
[Adam Sills, 2010]

A la ves que se va leyendo el archivo se van llenando variables las cuales mas adelante conformaran el modelo 3D. Estas variables están formadas por cinco listas, lista de vértices de posición, de vértices normales, de coordenadas de textura, de caras y una lista para almacenar la información de las texturas utilizadas por el modelo.

El fichero Collada (DAE), exporta los modelos de forma unitaria, lo que significa que las listas de vértices están en base uso, por ello se hace necesario agregar variables como, traslación, rotación y escala. Aplicando técnicas de multiplicación de matrices se logran las transformaciones pertinentes en el modelo para así poder mostrarlo tal y como se diseño.

Pseudocódigo

```
Mientras (fichero.leer)
    Para valor de elemento
        caso (library_images)
            Leer id y dirección
        caso (library_geometries)
            Para valor de elemento
                caso (geometry)
                    Leer id del modelo
                caso Arreglo de posición
                    Leer arreglo de vértices de posición
                caso Arreglo de normales
                    Leer arreglo de vértices normales
                caso Arreglo de vértices de textura
                    Leer arreglo de vértices de textura
                caso (triangles
                    Leer cantidad de triángulos del modelo
                    Leer arreglo de caras
```

2.5.2. Interpretación de formato ASE

El formato ASE como fichero adicional para interpretar, tiene una estructura sencilla, posee todos los parámetros necesarios para cumplir con los objetivos de este trabajo. Al igual que los demás ficheros, ASE está estructurado en bloques, donde solo dos de ellos se utilizaran para representar modelos 3D en escenas de RV. Estos bloques son:

Materiales

Este bloque contiene toda la información necesaria respecto a los materiales y texturas utilizadas por el modelo. El bloque comienza con la sentencia (*MATERIAL_LIST*), a

partir de aquí se enumeran todos los materiales utilizados mostrando de cada uno de ellos todos los parámetros necesarios para ponerlos en escena. A través de la etiqueta (*BITMAP*) se define la dirección física de las imagenes a utilizar.

Geometrías

El bloque geometría comienza con la etiqueta (*GEOMOBJECT*), este contiene todos los parámetros que conforman una malla, como vértices de posición, normales, coordenadas de texturas y caras. La etiqueta (*MESH_VERTEX*) describe los vértices de posición, (*MESH_VERTEXNORMAL*)define las normales, (*MESH_TVERT*) define las coordenadas de textura y (*MESH_FACE*) define las estructura que tendrán las caras del modelo.

Cargar ASE

Haciendo uso de estructuras (*StreamReader*).

Pseudocódigo

```
Mientras (fichero.leer linea)
    Para valor de elemento
        Si linea contiene *BITMAP
                Leer información de textura
        Si linea contiene *MESH_VERTEX
                Leer información de vértices de posición
        Si linea contiene *MESH_FACE
                Leer información de caras
        Si linea contiene *MESH_TVERT
                Leer información de vértices de textura
        Si linea contiene **MESH_TFACE
                Leer información de relación vértices de textura con cada cara
```

2.5.3. Interpretación de formato OBJ

El formato OBJ es uno de los ficheros mas utilizados por las herramientas de diseño para almacenar sus modelos, tiene una estructura muy sencilla pero bastante completa, cumple con los requisitos necesarios para cumplir con los objetivos de este trabajo. El

fichero esta estructurado en bloques, los cuales a continuación se describen:

Textura

Este fichero solo muestra el nombre de otro fichero el cual tendrá la información de textura utilizada, lo define la palabra (*mtllib*) y seguido el nombre del fichero con extensión MTL. Este otro fichero contiene varios parámetros de la textura entre ellos la dirección donde esta se encuentra, encabezado con la palabra (*map_Kd*).

Vértices de posición

Los vértices de posición están descritos en este fichero de forma sencilla, cada uno de ellos lo define una **v** en el inicio de cada linea del fichero, seguido por tres valores los cuales conforman el vector (x, y, z) de cada vértice.

Coordenadas de textura

De la misma manera, las coordenadas de textura están descritas por una **vt** en el inicio de cada linea del fichero y esta seguido de los valores (x, y, z) de cada coordenada de textura del modelo.

Vértices normales

Los vértices normales al igual que los anteriores señalados esta descrito por una **vn** en el inicio de la linea y seguido de sus componentes (x, y, z) correspondientes.

Caras

Las caras están definidas por una **f** comenzando cada linea, le sigue una estructura de la siguiente forma [**v1/vt1/vn1**] [**v2/vt2/vn2**] [**v3/vt3/vn3**], donde cada cara esta compuesta por las componentes de tres vértices de posición y sus correspondientes vértices de textura y normales, cada valor significa una posición en las listas anteriormente mencionadas.

Cargar OBJ

Haciendo uso de estructuras (*StreamReader*).

Pseudocódigo

```
Mientras (fichero.leer linea)
        Si linea comienza mt
                Leer información de textura
```

Si linea comienza *v*
    Leer información de vértices de posición
Si linea comienza *vt*
    Leer información de vértices de textura
Si linea comienza *vn*
    Leer información de vértices normales
Si linea comienza *f*
    Leer información de caras

2.5.4. Interpretación de formato STL

Sin dudas es el formato de ficheros 3D mas sencillo y con menos información de malla. Este fichero no posee información de textura, solo cuenta con vértices de posición y normales.

Bloque principal

El fichero esta estructurado por bloques de polígonos, en este caso triángulos, cada uno de ellos contiene los parámetros necesarios, como vértice normal de cada uno, representado con la palabra **normal** y a continuación el vector (x, y, z) que lo identifica. Los vértices de posición están definidos por la palabra **vertex** y seguido por sus componentes (x, y, z). Cada cara cuenta con tres vértices de posición y un vértice normal.

Cargar STL

Haciendo uso de estructuras (*StreamReader*).

Pseudocódigo

Mientras (fichero.leer linea)
    Si linea comienza *fa*
        Leer información de vértice normal
    Si linea comienza *ve*
        Leer información de vértices de posición

2.5.5. Visualizar el archivo

En la GLSVe se pueden visualizar muchos tipos de objetos, pero todos creados a través de primitivas geométricas (cubos, esferas, puntos, splines), el procedimiento para adicionar cualquiera de estas primitivas basta con hacer referencia a métodos nativos de OpenGL. En el caso de incorporar modelos 3D a escenas en la GLSVe, el proceso lleva unos cuantos pasos, además se hace uso de variables auxiliares, tales como listas para almacenar vértices. Este proceso se detalla a continuación vea figura 2.1.

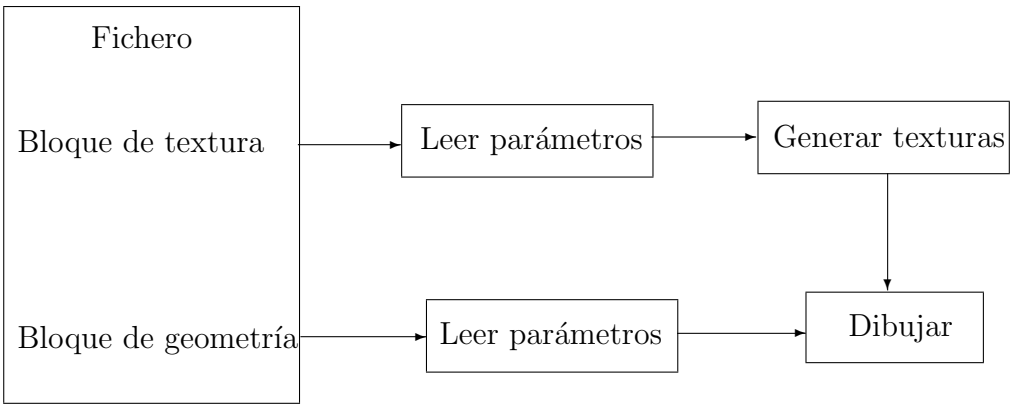

Figura 2.1: Pasos para llegar al dibujado

Cargar y generar texturas

El cargado de texturas se hace mediante (*buffers*) y funciones de OpenGL.
Una vez cargada la imagen en memoria debemos configurar una serie de parámetros que servirán para definir cómo se aplicará la textura sobre el polígono. Normalmente partiremos de una imagen almacenada en algún formato standard (bmp, jpg, gif, png).

Dibujar

Para el dibujado de los modelos en la escena se utilizaron funciones de OpenGL a través de la biblioteca de recubrimiento Tao Framework. El proceso de dibujado no es mas que representar polígonos (triángulos) utilizando los vértices de posición capturados del fichero, los vértices normales y las coordenadas de textura.[Ope, 2010]

Conclusiones

Durante el desarrollo de este capítulo se hizo una propuesta de solución al problema científico planteado. Después de haber hecho un riguroso análisis en el Capítulo 1 de la teoría que sustenta científicamente el problema esbozado, se establecieron las bases para el desarrollo de la aplicación, exponiendo por qué se hizo la elección de los ficheros a interpretar. Se llegó a la conclusión de interpretar un fichero complejo y tres de menor complejidad. Con esta nueva funcionalidad la GLSVe cuenta con prestaciones suficientes para desarrollar aplicaciones de realidad virtual de forma rápida, cómoda con mejores rendimiento y con mejoras en la calidad visual.

Capítulo 3

Diseño de la solución

Introducción

En este capítulo se define una visión más detallada y específica del sistema que se va a desarrollar, tomando como base la propuesta de solución planteada en el capítulo anterior. Se creará el modelo del dominio, haciendo una explicación de los conceptos presentes en este durante el desarrollo del glosario de términos. También se determinan las capacidades o funciones que el sistema debe cumplir (requisitos funcionales) y las propiedades o cualidades que el producto debe tener (requisitos no funcionales) y se describen los procesos en forma de casos de uso que dan cumplimiento a los requisitos funcionales.

3.1. Modelo de dominio

El modelo del dominio se describe mediante diagramas UML, específicamente con un diagrama de clases conceptual significativo en el dominio del problema.

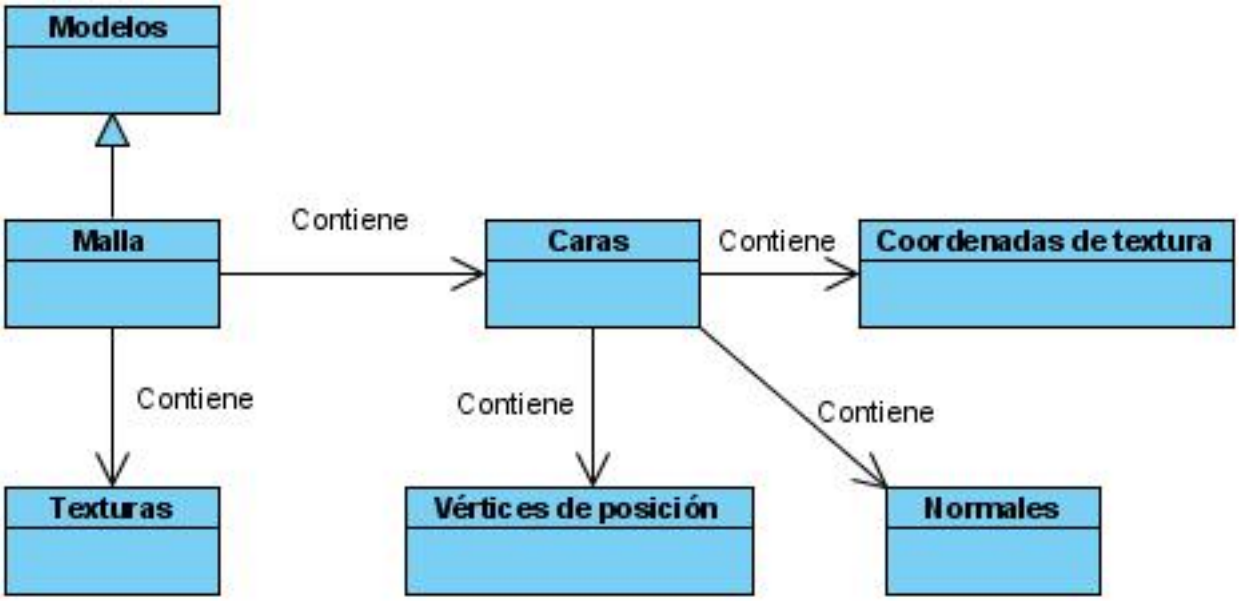

Figura 3.1: Modelo de dominio

3.1.1. Glosario de términos del modelo de dominio

La función del glosario de términos es relacionar todos los conceptos presentes en el modelo de dominio, especificando cada uno de ellos para una mejor comprensión de la solución propuesta.

Modelos: Raíz de muchos tipos de modelos.

Malla: Una malla esta conformada por caras.

Texturas: Representan toda la información de las texturas de una malla.

Caras: Las caras conforman una malla, en este caso representan triángulos;

Vértices de posición: Están conformados por vectores 3D que especifican las posiciones de los vértices de una malla.

Normales: Están conformados por vectores 3D que especifican los vértices normales de una malla.

Coordenadas de textura: Están conformados por vectores 3D que especifican las coordenadas de textura de una malla.

3.2. Especificación de requisitos

Durante la captura de requisitos se exponen las condiciones o capacidades que debe cumplir el sistema, así como también las propiedades o cualidades que debe presentar el mismo para darle solución al problema planteado.

3.2.1. Requerimientos funcionales

- RF1 Cargar el modelo
- RF2 Visualizar modelo
- RF3 Modificar parámetros de los modelos

  Rotar el modelo

  Trasladar el modelo

  Escalar el modelo

3.2.2. Requerimientos no funcionales

Requerimientos de Software.

Sistema Operativo Windows XP.

Requerimientos de Hardware.

Microprocesador Intel Pentium 3 o superior.
Memoria RAM de 256 MB o superior.
Tarjeta de Vídeo 64 MB o superior.

Restricciones de Diseño e Implementación

Se empleará como lenguaje de programación C# y el framework Tao para la implementación y dibujado de los modelos.

3.3. Modelo de caso de uso del sistema

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

3.3.1. Diagrama de caso de uso del sistema

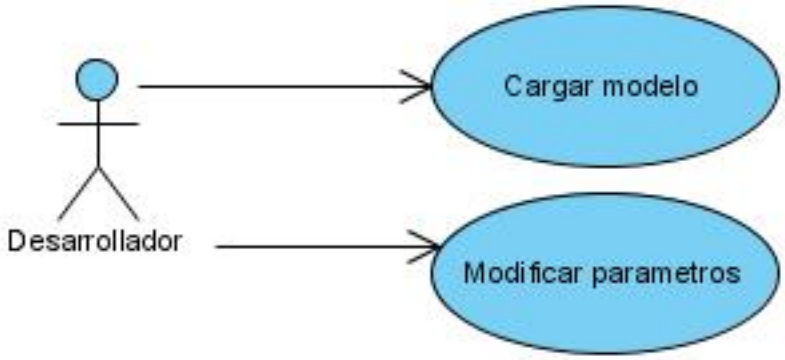

Figura 3.2: Diagrama de caso de uso del sistema

3.3.2. Descripción de los casos de uso del sistema

Caso de uso cargar modelo

Caso de uso	Cargar modelo	
Actores	Usuario	
Resumen	El CU inicia cuando el usuario ejecuta la acción cargar modelo en la aplicación. Inmediatamente el sistema inicializa los recursos necesarios para que cargue la información contenida en el fichero.	
Referencia	RF-1	
Curso normal de los eventos		
Acción de usuario		Respuesta del sistema
1. El usuario ejecuta la acción cargar modelo		1.1 El sistema verifica que el fichero a cargar exista. En caso contrario ver flujo alterno 1.2 El sistema inicializa los recursos para cargar la información contenida en el fichero. 1.3 El sistema inicializa los recursos necesarios para visualizar el modelo
Curso alterno de los eventos		
Flujo alterno 1		
		Si la dirección del fichero introducida es incorrecta o el sistema no encuentra el fichero, el sistema envía un mensaje indicando que el fichero no existe
Precondiciones		El fichero debe estar previamente creado
Poscondiciones		Queda almacenada la información del fichero
Prioridad		Crítico

Tabla 3.1: Especificación de caso de uso cargar modelo

Caso de uso modificar parámetros

Caso de uso	Modificar parámetros
Actores	Usuario
Resumen	El CU inicia cuando el usuario ejecuta la acción modificar parámetros del modelo en la aplicación.
Referencia	RF-3
Curso normal de los eventos	
Sección "Trasladar modelo"	
Acción de usuario	Respuesta del sistema
1. El usuario ejecuta la acción Trasladar	1.1 El sistema verifica que los parámetros de traslación estén correctos. En caso contrario ver flujo alterno. 1.2 El sistema actualiza los valores de posición del modelo.
Sección "Rotar modelo"	
Acción de usuario	Respuesta del sistema
1. El usuario ejecuta la acción Rotar	1.1 El sistema verifica que los parámetros de rotación estén correctos. En caso contrario ver flujo alterno. 1.2 El sistema actualiza los valores de rotación del modelo.
Sección "Escalar modelo"	
Acción de usuario	Respuesta del sistema
1. El usuario ejecuta la acción Escalar	1.1 El sistema verifica que los parámetros de escalado estén correctos. En caso contrario ver flujo alterno. 1.2 El sistema actualiza los valores de rotación del modelo.
Sección "Escalar modelo"	
Acción de usuario	Respuesta del sistema
1. El usuario ejecuta la acción Escalar	1.1 El sistema verifica que los parámetros de escalado estén correctos. En caso contrario ver flujo alterno. 1.2 El sistema actualiza los valores de escala del modelo.
Curso alterno de los eventos	
Flujo alterno 1	
	Si los parámetros introducidos no son correctos el sistema envía un mensaje indicando el tipo de error producido.
Precondiciones	El modelo debe estar previamente creado
Poscondiciones	Queda actualizada la información del modelo en escena
Prioridad	

Tabla 3.2: Especificación de caso de uso modificar parámetros

3.4. Diagrama de clases del diseño

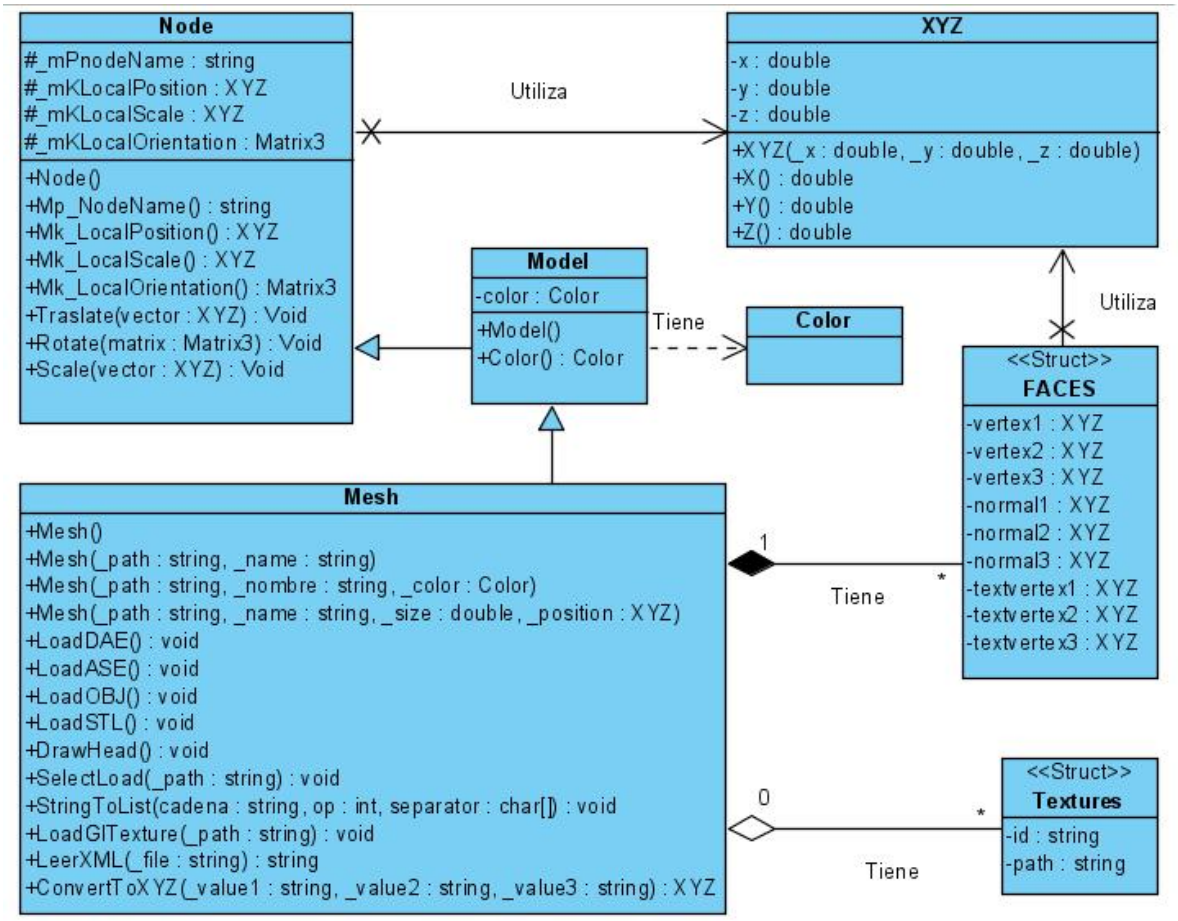

Figura 3.3: Diagrama de clases del diseño

3.5. Descripción de las clases del diseño

Descripción de la clase Mesh

Nombre: Mesh	
Tipo de clase:	
Operaciones	
Nombre	LoadDAE()
Descripción	Interpreta la informacion de un fichero Collada (.dae), contenido en el atributo path.
Nombre	LoadASE();
Descripción	Interpreta la informacion de un fichero (.ase), contenido en el atributo path.
Nombre	LoadOBJ()
Descripción	Interpreta la informacion de un fichero (.obj), contenido en el atributo path.
Nombre	LoadSTL()
Descripción	Interpreta la informacion de un fichero (.stl), contenido en el atributo path.
Nombre	DrawHead()
Descripción	Visualiza la información contenida en la lista de caras
Nombre	SelectLoad(string path)
Descripción	Selecciona el fichero a interpretar según el path pasado por parametro
Nombre	StringToList(string cadena, int operación, char[] separador)
Descripción	Llena listas (vertices de posición, vertices normales, vertices de textura, caras) convirtiendo cadenas de caracteres extraidas del fichero. Lo utiliza el método LoadDAE().
Nombre	LoadGlTexture(string path)
Descripción	Genera la textura contenida en el parametro path
Nombre	ReadMTL(string path)
Descripción	Extrae la dirección de la imagen utilizada como textura en un fichero (.obj)
Nombre	ConvertToXYZ(string v1, string v2, string v3)
Descripción	Método auxliliar que convierte 3 valores de cadena en un vector XYZ

Tabla 3.3: Descripción de la clase Mesh

Descripción de la clase Node

Nombre: Node	
Tipo de clase: Controladora	
Atributos	Tipo
_mPnodeName	String
_mKLocalPosition	XYZ
_mKLocalScale	XYZ
Operaciones	
Nombre	Mp_NodeName()
Descripción	Devuelve el nombre del nodo.
Nombre	Mk_LocalPosition()
Descripción	Devuelve el vector posición del nodo
Nombre	Mk_LocalScale()
Descripción	Devuelve el vector escala del nodo
Nombre	Traslate(XYZ vector)
Descripción	Modifica el vector posición del nodo
Nombre	Rotate(Matrix3 matrix)
Descripción	Modifica la matrix de orientación del nodo
Nombre	Scale(XYZ vector)
Descripción	Modifica el vector escala del nodo

Tabla 3.4: Descripción de la clase Node

Descripción de la clase Model

Nombre: Model	
Tipo de clase: Controladora	
Atributos	Tipo
color	Color
Operaciones	
Nombre	Color()
Descripción	Devuelve el color del modelo.

Tabla 3.5: Descripción de la clase Model

Descripción de la estructura Texture

Nombre: Textures	
Tipo de clase: Auxiliar	
Atributos	Tipo
Id	String
Path	String

Tabla 3.6: Descripción de la estructura Texture

Descripción de la estructura FACE

Nombre: FACES	
Tipo de clase: Auxiliar	
Atributos	Tipo
Vertex1	XYZ
Vertex2	XYZ
Vertex3	XYZ
Normal1	XYZ
Normal2	XYZ
Normal3	XYZ
TextVertex1	XYZ
TextVertex2	XYZ
TextVertex3	XYZ

Tabla 3.7: Descripción de la estructura FACE

Capítulo 4

Análisis de resultados

4.1. Introducción

En el presente capítulo se hace un análisis de todas las posibles situaciones en las cuales se puede encontrar una aplicación. Se analiza el rendimiento de las escenas con los distintos modelos en ella y en escenas completas (varios modelos en la misma escena). También se tiene en cuenta las propiedades de las texturas con respecto a la frecuencia de dibujado.

4.2. Verificación de los resultados

4.2.1. GLSVe antes de la interpretación de ficheros 3D

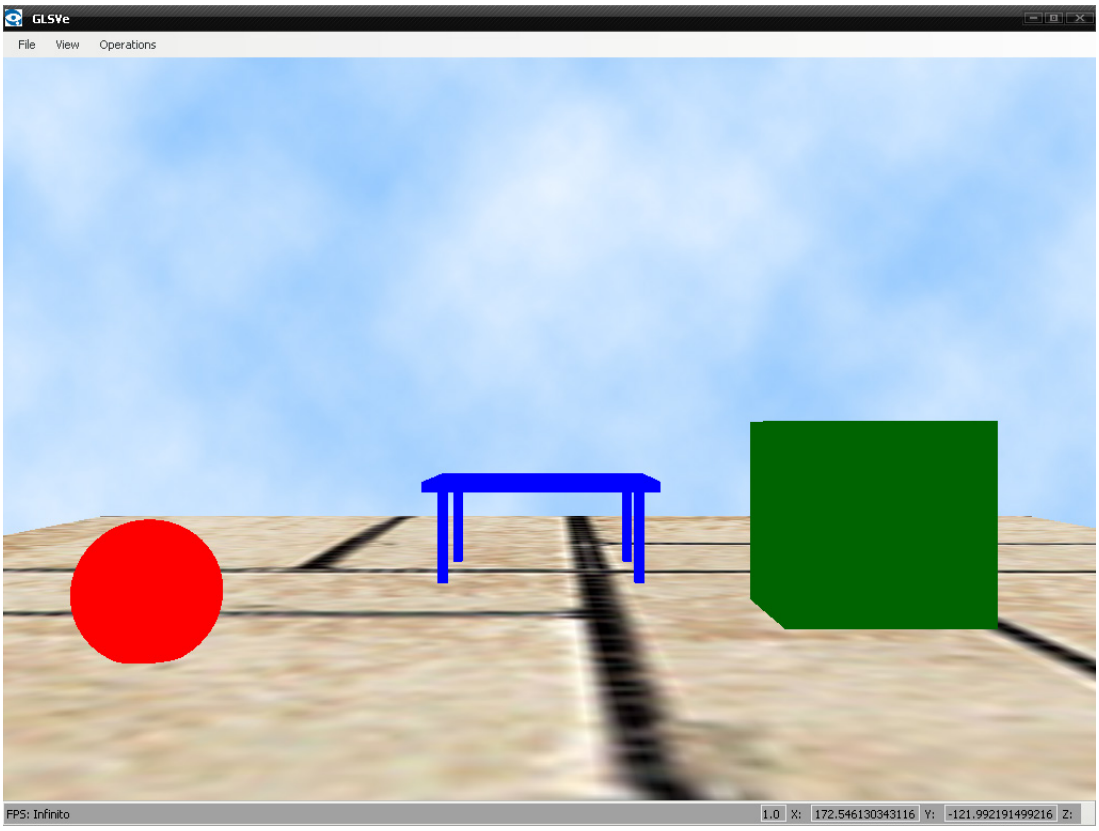

Figura 4.1: Escena de GLSVe antes de interpretar ficheros 3D

Una escena creada con GLSVe antes de la interpretación de ficheros 3D tenia las siguientes características:

Modelos	Primitivas geométricas
Materiales	Color
Modelos con textura	Solo planos
Fps	64
Tiempo de creación de una escena	25 min

Tabla 4.1: Característica de escena en GLSVe antes de la interpretación de ficheros 3D

4.2.2. GLSVe después de la interpretación de ficheros 3D

Figura 4.2: Escena de GLSVe interpretando ficheros 3D

Una escena creada con GLSVe interpretando de ficheros 3D tiene las siguientes características:

Modelos	Importados de herramientas de diseño
Materiales	Color y texturas
Modelos con textura	Todos
Fps	64
Tiempo de creación de una escena	5 min

Tabla 4.2: Característica de escena en GLSVe con interpretación de ficheros 3D

4.3. Pruebas de rendimiento

Las pruebas de rendimiento no son mas que ejecuciones de la aplicación en diferentes situaciones y con diferentes modelos en escena, para medir los *Frame per Second*(Fps), frames por segundo de cada una.

4.3.1. Prueba de interpretación y visualización de un fichero Collada (DAE)

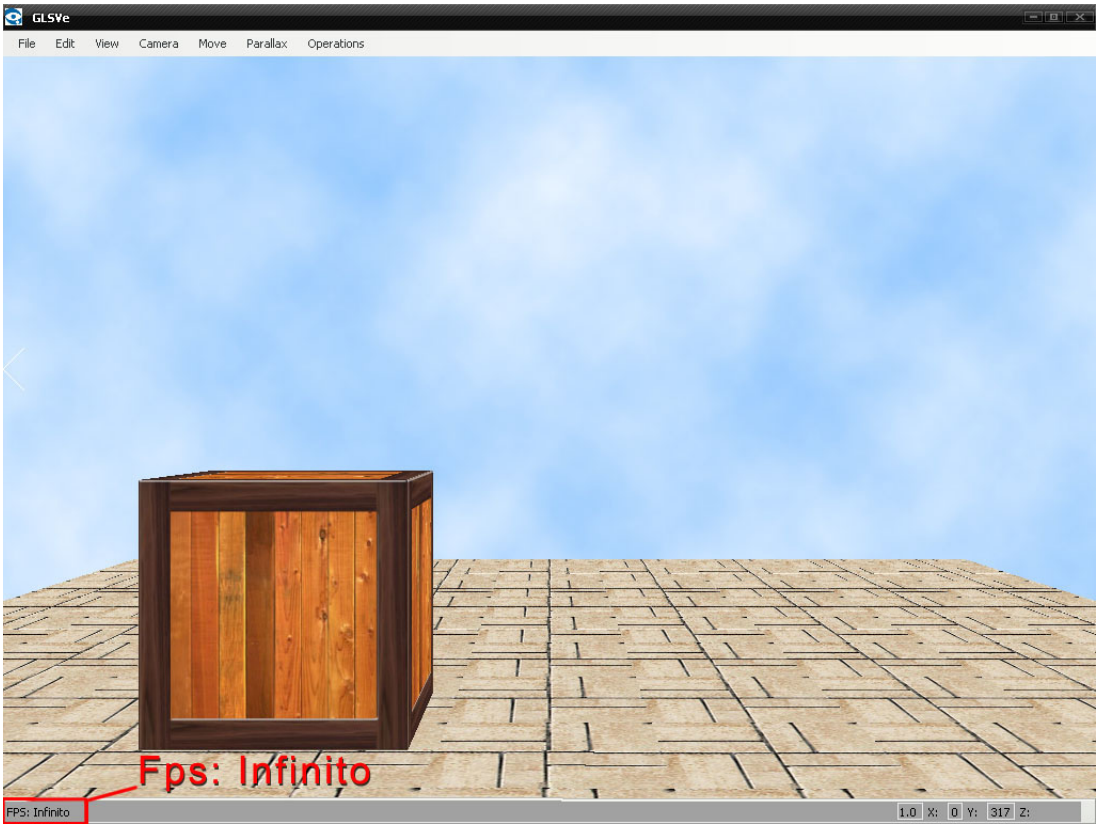

Figura 4.3: Prueba de interpretación del modelo DAE

Tipo de materiales	Textura
Cantidad de triángulos	12
Dimensiones de la textura	256x256
Fps	64

Tabla 4.3: Prueba de interpretación del modelo DAE

4.3.2. Prueba de interpretación y visualización de un fichero OBJ

Figura 4.4: Prueba de interpretación del modelo OBJ

Tipo de materiales	Textura
Cantidad de triángulos	172
Dimensiones de la textura	512x256
Fps	64

Tabla 4.4: Prueba de interpretación del modelo OBJ

4.3.3. Prueba de interpretación y visualización de un fichero ASE

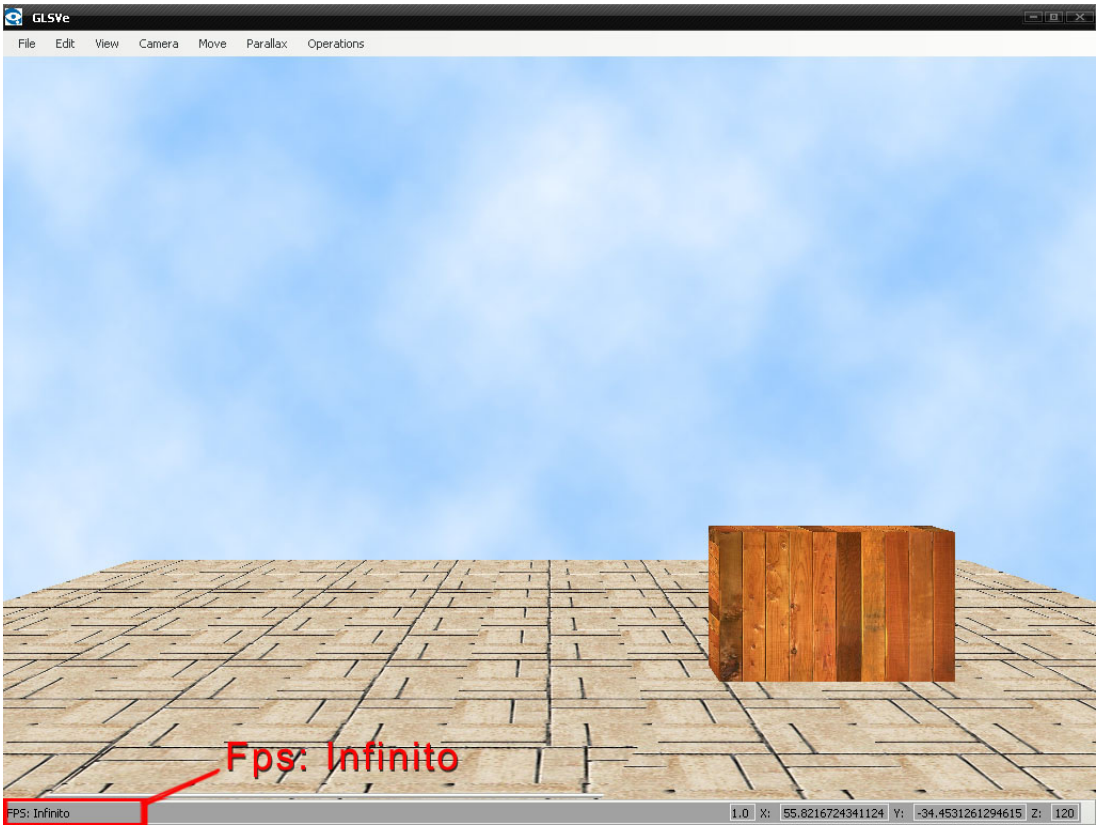

Figura 4.5: Prueba de interpretación del modelo ASE

Tipo de materiales	Textura
Cantidad de triángulos	12
Dimensiones de la textura	512x512
Fps	64

Tabla 4.5: Prueba de interpretación del modelo ASE

4.3.4. Prueba de interpretación y visualización de un fichero STL

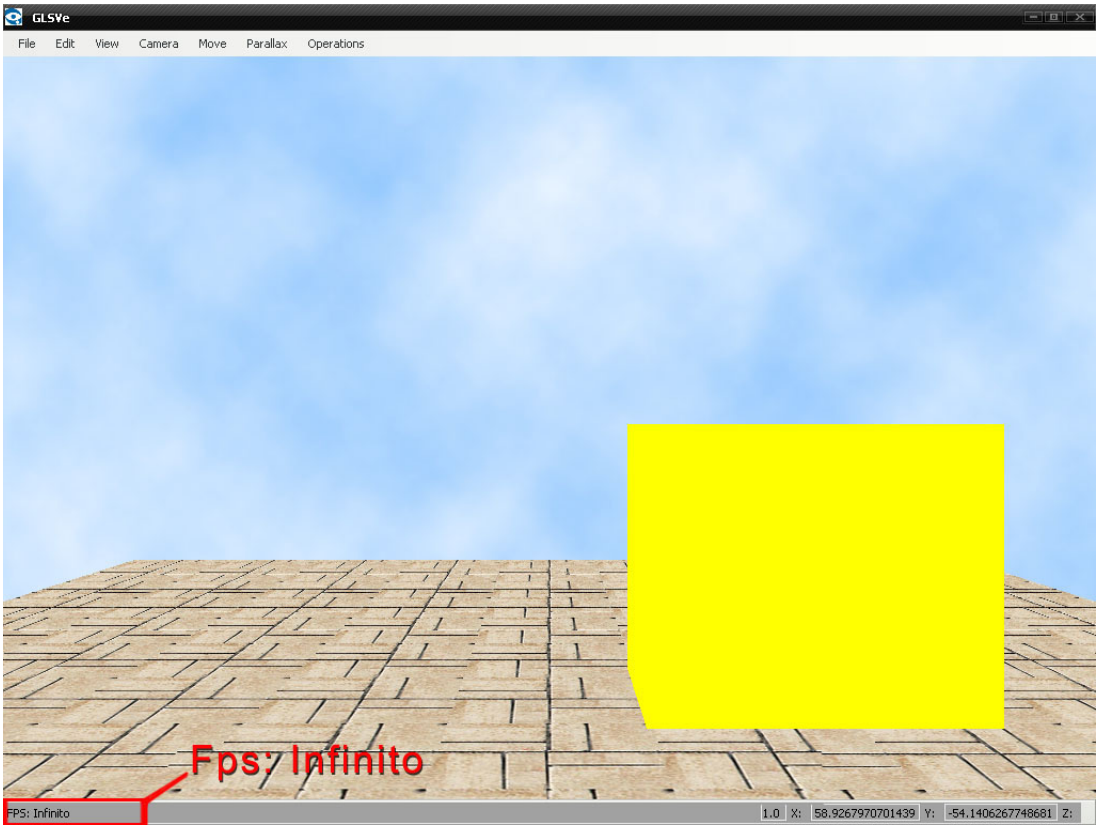

Figura 4.6: Prueba de interpretación del modelo STL

Tipo de material	Color amarillo
Cantidad de triángulos	12
Fps	64

Tabla 4.6: Prueba de interpretación del modelo STL

4.3.5. Prueba de interpretación y visualización de un una escena con varios fichero de distintos tipos

Figura 4.7: Prueba de interpretación de escena completa

Tipo de materiales	Textura
Cantidad de triángulos	3475
Dimensiones de la textura	Todas potencia de 2
Fps	64

Tabla 4.7: Prueba de interpretación de escenas completas

4.3.6. Prueba de rendimiento basado en las dimensiones de textura

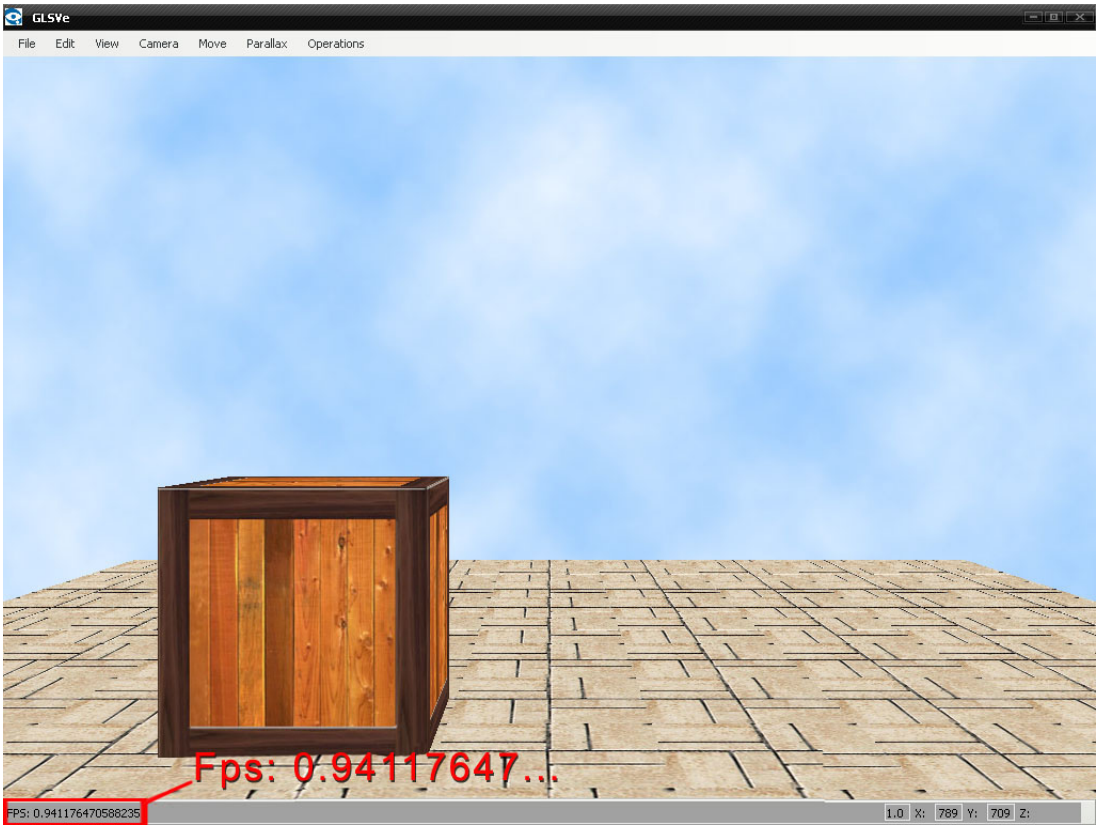

Figura 4.8: Prueba de rendimiento de textura

Tipo de materiales	Textura
Cantidad de triángulos	12
Dimensiones de la textura	280x300
Fps	1

Tabla 4.8: Prueba de rendimiento de textura

Conclusiones de los resultados obtenidos

Después de la ejecución de las pruebas antes expuestas, se llega a la conclusión que para un obtener un rendimiento óptimo en una escena, las texturas utilizadas en los modelos deben tener dimensiones potencias de 2, ejemplo: 128x128, 256x512, 64x128, etc. Se pudo demostrar que el parámetro cantidad de polígonos, no presenta una influencia significativa en el rendimiento de la escena.

Conclusiones

En el transcurso de la investigación realizada se han llegado a las siguientes conclusiones:

- La biblioteca GLSVe con el módulo de interpretación de ficheros 3D, optimizó considerablemente el tiempo de desarrollo de aplicaciones.
- La GLSVe ahora cuenta con mas prestaciones y opciones de visualización.
- Con el cumplimiento en GLSVe de las funcionalidades necesarias, se logró proveer en las actividades desarrolladas, mayor realismo y credibilidad.

Recomendaciones

- Se recomienda estudiar otros tipos de ficheros con otras características para continuar con el enriquecimiento de este módulo.
- Para futuras actualizaciones de la GLSVe hacer énfasis en el tratamiento de ficheros 3D, hacer captura de otras características brindadas por estos ficheros tales como, animación, colisiones, luces, física de cuerpos, entre otros. Ya que para aplicaciones de RV tener bien definido los modelos 3D es fundamental para formar aplicaciones de buena calidad y funcionalidad.
- Desarrollar un editor de escenas, que en conjunto con este módulo puedan editar los escenarios con mayor comodidad y precisión.

Referencias bibliográficas

[ASE, 2010] (2010). "ase file format". http://wiki.beyondunreal.com/wiki/ASE_File_Format.

[OGR, 2010] (2010). Ogre. http://www.ogre3d.org/.

[Ope, 2010] (2010). Open graphics library.

[Adam Sills, 2010] Adam Sills, Mesbah Ahmed, D. F. B. (2010). *XML.NET Developer's Guide*. www.ebook3000.com.

[Argente, 2006] Argente, R. T. (2006). Motores gráficos. *http://informatica.uv.es/iiguia/IG/motores_graf*.

[Evan Pipho, 2003] Evan Pipho, A. L. (2003). *Focus On 3D Models*. Stacy L. Hiquet.

[Group, 2008] Group, K. (2008). Collada overview. *www.khronos.org/collada*.

[José R. Hilera, 1999] José R. Hilera, Salvador Otón, J. M. (1999). Aplicación de la realidad virtual en la enseñanza a través de internet. *http://www.ucm.es/info/multidoc/multidoc/revista/num8/hilera-oton.html*.

[Khronos Group, 2008] Khronos Group, I. (2008). *COLLADA – Digital Asset Schema Release*. http://www.collada.org/2008.

[Wendy García López, 2007] Wendy García López, A. E. G. (2007). Sistema de generaciÒn de ficheros para entornos virtuales. Master's thesis, UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.

Glosario de términos

Biblioteca: desde el punto de vista informático es una colección o conjunto de subprogramas usados para desarrollar un software.

Malla: Forma de representar un modelo a partir de polígonos. Colección de vértices, aristas y polígonos conectados, de forma que cada arista es compartida como máximo por dos polígonos.

Mapa de texturas: Correspondencia entre vértices de una textura y los vértices del modelo.

Material: Combinación de luces y colores usados para definir una apariencia.

Matrices de transformación: Matrices definidas para calcular nuevas coordenadas a partir de las ya existentes según una determinada transformación gráfica (rotación, traslación, escalado y reflexión).

Motor gráfico: (graphic engine en inglés) 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.

Normal: De manera básica es la dirección en la que mira cada cara de un objeto.

Realidad Virtual: representación de escenas u objetos producidos por un sistema informático, dando la sensación de su existencia real.

Renderizado: Proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño en

3D.

Tao Framework : Interfaz de programación de C# para acceso a bibliotecas escritas en C++.

Visión estereoscópica: Visión binocular.

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

Términos en inglés

Buffer: Término del inglés, que define una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

Streaming: Término del inglés, que se refiere a un proceso continuo.

Acrónimos

3D: Tres dimensiones.

3DS: 3D Studio (file format).

ASE: ASCII Scene Exporter.

ASCII: American Standard Code for Information Interchange.

DAE: Digital Asset Exchange.

Fps: Frames per Second, (Frames por segundos).

GLSVe: Graphics Library for Stereoscopic Vision engine.

OBJ: Nombre del formato creado por Alias Wavefront.

RV: Realidad virtual.

SRV: Sistema de realidad virtual.

STL: Standard Tessellation Language.

UCI: Universidad de las Ciencias Informáticas.

XML: Lenguaje de marcas extensible (XML, del inglés extensible Markup Language).