Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título: Implementación del módulo de persistencia y visualización para una herramienta de creación de Centros Expositivos Virtuales

Autor: Andy Torres Utra

Tutor: Ing. Minardo Gollún González López

Cotutor: MsC. Lidiexy Alonso Hernández

Ciudad de la Habana

2011

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizamos al tutor Minardo Gollún González López y al Centro de Informática Industrial (CEDIN), de 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 _____ del año _____.

Andy Torres Utra **Minardo G. González López**

_____ _____

Firma del Autor Firma del Tutor

Agradecimientos:

A Enier, mi equipo de trabajo del proyecto Paseos Virtuales.

A mi tutor, por la ayuda brindada durante el desarrollo de la tesis.

A mi tía Norma, Benigno, Gaby y Patricia, que fueron mi familia más cercana durante estos 5 años.

A todos mis compañeros: Manuel, Evert, Reinier, Fumero, Nidelso, Rufino, Tony, y otros tantos, por estos magníficos 5 años de carrera.

A todos aquellos que contribuyeron, de una forma u otra, a mi formación como ingeniero y como persona.

Dedicatoria

A mis cuatro padres, María, Rogelio, Edmundo y Yetny, algunos más lejanos que otros, pero que siempre han estado ahí.
A toda mi familia.

Datos de contacto:

Tutor:

Nombre y Apellidos: Minardo Gollún González López

Edad: 28 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Instructor

E-mail: mgonzalezl@uci.cu

Graduado de la UCI, con cinco años de experiencia en el tema de Realidad virtual, y jefe del proyecto Paseos Virtuales en el Centro de Desarrollo de Informática Industrial.

Resumen:

La creación de un módulo de edición de escena 3D para una herramienta de creación de Centros Expositivos Virtuales supone un avance para el proyecto Paseos Virtuales del CEDIN[1], debido a la optimización que le brinda al proceso de crear un Paseo Virtual[2]. Sin embargo, para asegurar la completa funcionalidad de dicha herramienta, esta necesita un módulo de persistencia y visualización para asegurar que los cambios hechos a un escenario no se pierdan, a la vez que permitirá visualizar el Paseo una vez concluida la etapa de edición.

A través de la realización de este Trabajo de Diploma, se desarrolló exitosamente un formato de fichero propio, basado en la información que se necesita guardar de un Paseo Virtual, y una aplicación de visualización sencilla e intuitiva, que cumple con los requisitos funcionales para presentar dicho Paseo. La integración de este módulo con el módulo de edición contribuirá a la robustez de la herramienta, haciendo la producción de Paseos Virtuales una tarea sencilla y potenciando a la vez la difusión de este tipo de productos en nuestro país.

[1] Centro de Informática Industrial, perteneciente a la Universidad de las Ciencias Informáticas (UCI).

[2] Aplicación que simula un recorrido por un entorno virtual.

Índice:

Índice de Figuras y Tablas:

Introducción

Los Entornos Virtuales son una representación de la realidad, hechos con la ayuda de computadoras. Son escenarios, reales o ficticios, que sitúan al visitante en un espacio determinado. Inicialmente surgieron para aplicarlos al campo de los videojuegos, pero a medida que se fueron desarrollando se encontraron aplicaciones que iban más allá de la mera simulación de un entorno, en otros campos como la medicina, la educación, los simuladores de vuelo, entre otros.

Es entonces cuando surge la idea de simular lugares específicos, de importancia histórica, cultural o científica, con la finalidad de que una persona que se encontrase en cualquier otro lugar del globo terráqueo pudiese, al menos virtualmente, "visitar" dicho sitio sin tener que desplazarse físicamente hasta allí; conjuntamente con la posibilidad de interactuar con los objetos del entorno, lo que aumenta considerablemente la sensación de realidad del visitante. A este tipo de simulaciones se les llamó Paseos Virtuales.

Actualmente, prestigiosos centros a nivel mundial hacen uso de los Paseos Virtuales para dar a conocer sus instalaciones, o para retroceder cientos o miles de años en el pasado y presentar eventos históricos de importancia que en este tiempo no se pudiesen presenciar. Tales son los casos del Smithsonian National Museum of Natural History, de los E.E.U.U, que nos permite recorrer sus salas a través de la web; la enciclopedia Microsoft Encarta, que ofrece un recorrido por la Atenas del año 432 A.C.; e incluso el juego Assasin's Creed II, en el cual el jugador se mueve por la Venecia de 1476, una Venecia detallada y realista, con sus canales típicos y habitantes de la época, recreando lugares como la basílica de San Marcos o el Puente de Rialto.

En Cuba, es muy poca la experiencia que se tiene en lo que a visitas virtuales se refiere. El mayor exponente de ello es la Universidad de las Ciencias Informáticas (UCI), y en ella, particularmente en el CEDIN[3], en la línea de Imágenes y Videos Basados en Render, se está

[3] Centro de Informática Industrial, perteneciente a la Universidad de las Ciencias Informáticas (UCI).

desarrollando una herramienta para la creación de Centros Expositivos Virtuales[4] que permitirá a un usuario, sin tener conocimientos previos de programación, diseñar un Paseo Virtual, pudiendo incorporarle a este modelos diseñados en herramientas de diseño 3D, y editar el escenario a su gusto. Dicha herramienta necesita un módulo de persistencia y visualización de la información, o sea, necesita un módulo que permita guardar los cambios efectuados a un escenario cualquiera y, en cualquier momento exportar ese escenario, pudiendo visualizarlo luego como un Paseo Virtual.

Es por eso que se plantea como **problema investigativo a resolver** ¿Cómo desarrollar un módulo para una herramienta de creación de Centros Expositivos Virtuales que permita: guardar los cambios efectuados en el módulo de edición, salvarlos en un formato de escena propio y visualizarlos posteriormente?

Para la solución del problema se plantea como **objeto de estudio** "las técnicas para conservar información digital" y como **campo de acción** "los formatos de escenas 3D y las técnicas para la creación de ficheros personalizados", teniendo como **objetivo general** "Implementar el módulo de persistencia para una herramienta de creación de Centros Virtuales Expositivos"; esperándose como posible resultado la obtención del módulo de persistencia para una herramienta de creación de Centros Virtuales Expositivos.

Para el cumplimiento del objetivo planteado anteriormente se trazan las siguientes **tareas a desarrollar**:

- Resumir el estado del arte relacionado con las técnicas de programación.
- Caracterizar posibles metodologías y lenguajes a utilizar para este tipo de desarrollo.
- Caracterizar los formatos de escenas 3D.
- Caracterizar los Paseos Virtuales.
- Resumir el Análisis y Diseño del módulo a Implementar
- Identificar las herramientas y lenguajes para el desarrollo del módulo.
- Diseñar y graficar el Diagrama de Componentes.

[4] Paseos Virtuales destinados a la exposición de objetos.

- Implementar los Casos de Usos del Módulo.
- Diseñar describir y aplicar pruebas de Caja Negra.

Con la realización de este Trabajo de Diploma se dotará a la herramienta de creación de Exposiciones Virtuales con un módulo de persistencia que permitirá conservar los Paseos Virtuales en un formato de escena propio, además de brindar una aplicación que permitirá visualizarlos como Paseo Virtual propiamente dicho, sin las opciones de edición, en la forma en que será presentado al usuario final.

En el progreso de la investigación científica se emplearon los siguientes métodos:

Métodos del nivel teórico: posibilitaron descubrir, analizar y sistematizar los resultados obtenidos, para llegar a conclusiones confiables que permitan resolver el problema. En tal sentido se usaron:

- El Analítico – sintético se utilizó al analizar toda la información relacionada con el tema de tesis, ya que permiten la extracción de los elementos más importantes de cada documento analizado. La inducción-deducción se utilizó durante toda la investigación, para llegar a conclusiones y hacer generalizaciones.
- El histórico-lógico se utilizó al estructurar la trayectoria del objeto en el transcurso de su historia y en orden cronológico. La Modelación Analógica se utilizó para la elaboración tanto de proceso actual, como propuestas de solución.

Métodos del nivel empírico: permitieron descubrir y acumular un conjunto de datos, que sirven de base para dar respuesta a las preguntas científicas. Para ello se utilizaron:

- La observación del tamaño resultante en los ficheros de las escenas, exportados desde las herramientas de diseño y la calidad de dichas escenas en tiempo real.

Estructura del trabajo.

El Trabajo de Diploma está estructurado en 3 capítulos:

Capítulo 1. Fundamentación teórica.

Incluye un estado del arte del tema tratado a nivel internacional, nacional y de la Universidad, de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad o en las que se apoya para la solución del problema que se enfrenta, sobre los que es necesario profundizar.

Capítulo 2. Resumen del Análisis y Diseño del módulo a Implementar.

Incluye las características del sistema, además de un resumen de los flujos de trabajo de Análisis y Diseño del mismo.

Capítulo 3. Implementación y pruebas.

Incluye los diagramas de despliegue y de componentes del flujo de trabajo de Implementación, y, del flujo de trabajo de Pruebas, los modelos de prueba, con la descripción de los casos de prueba de integración

Bibliografía consultada.

Capítulo 1. Fundamentación teórica.

Introducción al capítulo 1

Los editores de escenarios en tres dimensiones se han ido popularizando a través de los años debido al avance de los videojuegos, principalmente, dada la libertad que le brindan al usuario de crear o modificar un escenario del juego, y adaptarlo a su gusto. Para que la información de los modelos que componen el escenario persista en el tiempo, o sea, no se pierda una vez cerrado el programa, es necesario almacenarla en ficheros gráficos[5], a la vez que para visualizarlos, es necesaria la presencia de un motor gráfico[6](o engine) que se adapte a las características del fichero en cuestión. En este capítulo se hará un estudio de los ficheros gráficos 3D más usados, los motores gráficos más actuales y las herramientas de diseño y programación más aptas para crear aplicaciones de visualización 3D; analizando de ellos las características, ventajas y desventajas que ofrecen.

1.1 Características de los ficheros 3D.

Un fichero 3D es un contenedor de información, generalmente de un escenario en tres dimensiones, ordenada mediante un formato de almacenamiento. Debe contener la posición de los objetos en la escena, materiales, texturas, orientación de las normales de las caras, posición y orientación de las luces, estados de render[7], así como todos los restantes datos que conforman un escenario 3D. Dependiendo del formato de almacenamiento que se utilice, esto influirá en la complejidad del fichero, facilidad de entendimiento del mismo y tiempo de carga (**Echemendía González** and **García López** 2007).

[5] Archivo digital para almacenamiento gráfico.

[6] Serie de rutinas de programación que permiten el diseño, la creación y la representación de un videojuego.

[7] Creación y representación como imagen 2D, de los modelos gráficos en una escena.

1.2 Características de los formatos de almacenamiento 3D. Ficheros que los utilizan.

Un formato de almacenamiento 3D es un conjunto de reglas (algoritmo) que define la manera correcta de almacenar datos en memoria. En la actualidad se utilizan diversos formatos para el almacenamiento de la información en ficheros, siendo los más comunes el binario y el ASCII.

1.2.1 Formato binario.

Los ficheros binarios contienen típicamente una secuencia de bytes, o grupos ordenados de 8 bits. A la hora de crear un formato de archivo personalizado para un programa, el programador organiza estos bytes en la forma que almacene la información necesaria para la aplicación. El formato binario puede incluir múltiples tipos de datos en el mismo archivo, tales como imagen, video, audio, y las características de un escenario virtual (**Department of Computer Sciences** 2003). Son muy utilizados en juegos, por la rapidez de carga e interpretación del archivo.

Ficheros que emplean formato binario para su composición:

- STL (Standard Tessellation Language).
- Autodesk 3DS.
- Blender .BLEND.

1.2.2 Formato ASCII.

Un fichero ASCII es un fichero binario que está formado por caracteres ASCII. ASCII es el acrónimo inglés de American Standard Code for Information Interchange (Código Estadounidense Estándar para el Intercambio de Información).

Los ficheros ASCII, o de texto, son más restrictivos que los binarios porque sólo contienen datos textuales. Sin embargo, a diferencia de los binarios, son menos sensibles a corromperse. Mientras que un pequeño error en un archivo binario puede hacerlo ilegible, el mismo error en un archivo ASCII puede simplemente mostrarse luego de que el archivo haya sido abierto. Como los ficheros ASCII usan un formato estándar, múltiples programas son capaces de leer y editar estos ficheros (**FileInfo.com** 2010).

Ficheros que emplean formato ASCII para su composición:

- Wavefront OBJ.
- STL (Standard Tessellation Language).
- ASE (ASCII Scene Exporter).
- DXF (Drawing Exchange Format File).
- Ogre DotScene.

1.3 Formatos de ficheros 3D más utilizados en formato binario. Características.

1.3.1 STL (Standard Tessellation Language).

El formato STL es nativo de Stereolithography CAD, creado por 3d System en Valencia, California, Estados Unidos. Debido a que los ficheros STL ASCII pueden volverse muy grandes, existe una versión binaria de STL. El fichero STL binario tiene una cabecera (header) de 80 caracteres (que es generalmente ignorada, pero que no debe comenzar con 'solid' porque eso llevaría a la mayoría de los softwares a creer que es un fichero STL ASCII). Siguiendo al header hay un entero sin signo que indica cuántas caras triangulares hay en fichero. Luego, las descripciones de cada triángulo. El fichero termina cuando terminan los triángulos.

Cada triángulo se describe mediante 12 números de punto flotante de 32 bits (float): tres para la dirección de la normal de la cara, y tres para las coordenadas de cada vértice (como en la versión ASCII de STL). Seguidamente, un entero sin signo de 2 bytes (casi siempre 0), que indica que ya se terminó de describir la cara.

Estructura del fichero en formato binario:

`string` `Comentario`: un comentario al inicio de fichero.

`unsigned long int`: número de caras.

`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.

1.3.2 Autodesk 3DS.

Es un formato de archivo creado por Autodesk para el 3D Studio Max. Originalmente se creó como formato de archivo nativo para 3D Studio DOS (versiones 1-4). Debido a su fiabilidad y solidez, se ha convertido en un estándar industrial para el intercambio de modelos 3D entre programas de diseño 3D. Fue reemplazado en versiones posteriores por el formato .MAX, también de Autodesk.

Su organización está basada en bloques, o CHUNKS, en el que cada sección está contenida en un bloque que contiene un identificador de CHUNK, la longitud de los datos, y los datos propiamente dichos. Tiene una estructura jerárquica, similar a un árbol DOM xml. El fichero 3ds no sólo está dividido en pequeños bloques, sino que cada bloque está dividido en sub-bloques y lo mejor de todo es que no tiene que existir un orden en el fichero, por ello es que a simple vista es muy difícil entender su estructura (**Autodesk.com** 2007).

```
0x4D4D // Main Chunk
├─ 0x3D3D // 3D Editor Chunk
│  ├─ 0x4000 // Object Block
│  │  ├─ 0x4100 // Triangular Mesh
│  │  │  ├─ 0x4110 // Vertices List
│  │  │  ├─ 0x4120 // Faces Description
│  │  │  │  └─ 0x4130 // Faces Material
│  │  │  ├─ 0x4140 // Mapping Coordinates List
│  │  │  │  └─ 0x4150 // Smoothing Group List
│  │  │  └─ 0x4160 // Local Coordinates System
│  │  ├─ 0x4600 // Light
│  │  │  └─ 0x4610 // Spotlight
│  │  └─ 0x4700 // Camera
│  └─ 0xAFFF // Material Block
│     ├─ 0xA000 // Material Name
│     ├─ 0xA010 // Ambient Color
│     ├─ 0xA020 // Diffuse Color
│     ├─ 0xA030 // Specular Color
│     ├─ 0xA200 // Texture Map 1
│     ├─ 0xA230 // Bump Map
│     └─ 0xA220 // Reflection Map
│        │  /* Sub Chunks For Each Map */
│        ├─ 0xA300 // Mapping Filename
│        └─ 0xA351 // Mapping Parameters
└─ 0xB000 // Keyframer Chunk
   ├─ 0xB002 // Mesh Information Block
   ├─ 0xB007 // Spot Light Information Block
   └─ 0xB008 // Frames (Start and End)
      ├─ 0xB010 // Object Name
      ├─ 0xB013 // Object Pivot Point
      ├─ 0xB020 // Position Track
      ├─ 0xB021 // Rotation Track
      ├─ 0xB022 // Scale Track
      └─ 0xB030 // Hierarchy Position
```

Fig. 1 Estructura de un fichero 3DS. IDs de CHUNKS más comunes

1.3.3 Blender Foundation BLEND.

Creado como formato de archivo nativo de Blender, está diseñado con alto nivel de abstracción; orientado entre otras cosas para permitir que proyectos de animaciones complejas puedan ser creados dentro de un archivo, la reutilización eficiente de datos, y el uso de plantillas, o backdrops. Esto devino en una “base de datos” no estándar, diferente al grafo de escena

tradicional, pero basado en la creación de “mundos de datos” 3D donde se pueden construir tantos grafos de escena como se necesiten (**Blender.org** 2010).

Los bloques de datos en Blender están almacenados en un árbol principal, y son como los bloques de construcción del usuario: pueden ser copiados, modificados y vinculados unos con otros. Por ejemplo, el tipo de bloque “Object (objeto)” hace aparecer en el escenario 3D un bloque “Mesh (malla)”. El Objeto determina la ubicación exacta, rotación, y tamaño de la malla. El Mesh sólo almacena la ubicación de los vértices y las caras. También, más de un Object puede tener un vínculo con el mismo Mesh (por ejemplo, para crear una instancia). Otros tipos de bloques, como el ‘Material’, también pueden vincularse con Meshes (**Blender.org** 2010).

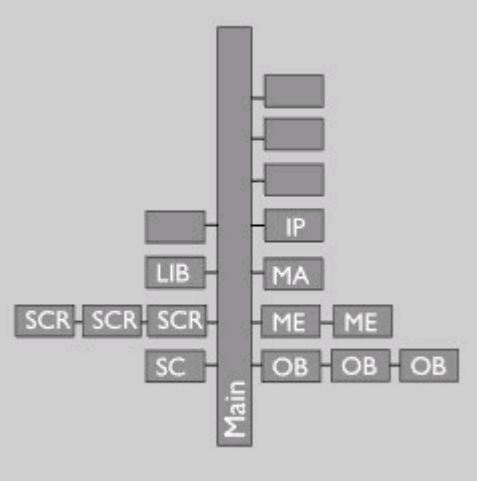

Fig. 2 Árbol de datos

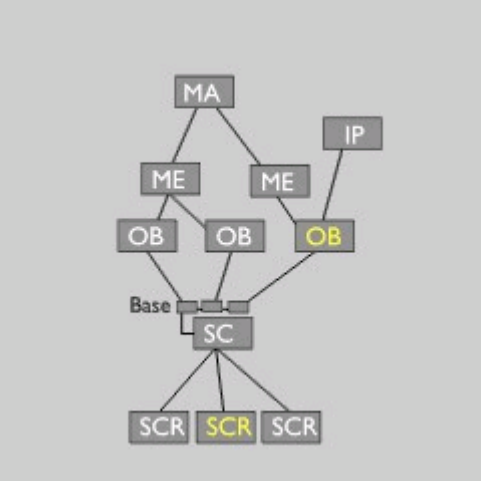

Fig. 3 Grafo de escena

Todos los bloques que están en el árbol principal (Main tree) son llamados Bloques de Librerías (Library Blocks, o Libdata). Dichos bloques comienzan con el ID struct como sigue:

```
<ccode>
typedef struct ID {
void *next, *prev; //para insertar en listas
```

struct ID *newid; //datos temporales para encontrar links nuevos

struct Library *lib; //puntero a librería opcional

char name[24]; //nombre único, comienza con un identificador de 2 bytes

short us; //cantidad de usuarios en el bloque

short flag; //banderas de bit para indicar tipos especiales

} ID;

</ccode>(**Blender.org** 2010)

1.4 Formatos de ficheros 3D más utilizados en formato ASCII. Características.

1.4.1 Wavefront OBJ.

El formato de archivo .obj es un formato de archivo de objetos 3D estándar creado para el uso con Wavefront's Advanced Visualizer (**Echemendía González** and **García López** 2007).

Es recomendable para objetos estáticos. Contiene geometrías poligonales que presentan puntos, aristas y caras, para definir un objeto o geometrías de objetos de forma libre que contienen curvas y superficies, aunque en lo que más se usa es en objetos poligonales. No necesita ningún tipo de encabezado, aunque es usual poner un comentario al inicio del fichero (**Echemendía González** and **García López** 2007).

Estructura del fichero

Se analizará el fichero para objetos poligonales que son los más usados en los entornos virtuales.

Un comentario con la versión o el autor del modelo.

v float float float: coordenada del vértice.

vn float float float: valor de la normal en el punto.

…

vt u v w: valor de las coordenadas de textura.

…

f int/int/int int/int/int int/int/int : información de las caras.

…

Los tres puntos suspensivos (...) significan que se repite el parámetro de acuerdo a la cantidad de atributos del tipo padre que exista. Lo que aparece en negrita y cursiva son las palabras reservadas que utiliza.

Especificaciones de las caras

La información de la cara está dada por:

f int/int/int int/int/int int/int/int

El primer int es para el número del vértice, el segundo es para el número de la textura y el tercero para el número de la normal.

Si no presenta normales de vértices y sí coordenadas de mapeo, sería de la siguiente forma.

f int/int

Si no presenta coordenadas de mapeo, entonces la cara se representa de la siguiente manera.

f int/ /int

1.4.2 STL (Standard Tessellation Language).

El formato puede encontrarse en datos ASCII o binarios, aunque es más común en formato binario debido a que el tamaño de los archivos son más pequeños. 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) (**Echemendía González** and **García López** 2007).

```
solid name          //nombre del objeto. Este continúa con un número de triángulos, cada uno
                    //estructurado como sigue
 facet normal ni nj nk      //dirección del vector normal de la cara
    outer loop              //inicialización de los vértices de la cara
      vertex v1x v1y v1z    //coordenadas z, y, z del vértice
   vertex v2x v2y v2z
```

```
  vertex v3x v3y v3z
 endloop  //fin de los vértices de la cara
endfacet   //fin de la cara
```

1.4.3 ASE (ASCII Scene Exporter).

Es el preferido por todos los que se inician en la programación de espacios virtuales, para la carga de objetos estáticos. El formato es basado en un identificador que siempre comienza por el carácter *, ejemplo: *Nombre_del_atributo.

El fichero está organizado en forma de jerarquía de clases, las cuales comienzan por un identificador y luego utilizan los caracteres **{** y **}** para encerrar todo el contenido respecto a este identificador. A continuación se muestra la estructura (**UnrealWiki** 2006).

```
*3DSMAX_ASCIIEXPORT          // versión del fichero.
*COMMENT                     // "un comentario"
*SCENE {}
 *MATERIAL_LIST
{
*MATERIAL_COUNT          // cantidad de materiales.
*MATERIAL 0 {}
}
*GEOMOBJECT
{
*NODE_NAME               //"nombre del objeto"
*NODE_TM {}
*MESH {}
*MESH_FACE_LIST {}
*MESH_NUMTVERTEX       // cantidad de vértices.
 *MESH_TVERTLIST {}
```

```
*MESH_NUMTVFACES        // cantidad de caras con textura.
*MESH_TFACELIST {} *MESH_NUMCVERTEX      // cantidad de vértices con colores.
*MESH_CVERTLIST {}
 *MESH_NUMCVFACES      // cantidad de caras con colores de vértices.
*MESH_CFACELIST {}
*MESH_NORMALS {}
}
*PROP_MOTIONBLUR 0
 *PROP_CASTSHADOW 1
 *PROP_RECVSHADOW 1
 *MATERIAL_REF 0
```

Especificaciones:

***SCENE {}**: contiene todo respecto a una escena, cuadro en que comienza la animación, cuadro en que termina, velocidad de la animación, color del ambiente, etcétera.

***MATERIAL_LIST {}**: contiene información de todos los materiales usados por la malla.

***GEOMOBJECT {}**: contiene información de la geometría del objeto.

***LIGHTOBJECT** y ***CAMERAOBJECT**: contiene información acerca de las luces y las cámaras en la escena (**UnrealWiki** 2006).

1.4.4 DXF (Drawing Exchange Format File).

Formato de datos desarrollado por Autodesk y usado por imágenes vectoriales de herramientas de diseño asistido por computadoras (CAD, por sus siglas en inglés). Fue creado como un formato universal de forma que los documentos de AutoCAD pudiesen ser abiertos más fácilmente por otros programas (**Autodesk.com** 2010).

La organización general de un fichero DXF es la siguiente:

Sección **HEADER**: información general acerca del dibujo. Consiste en número de versión de base de datos de AutoCAD y un número de variables de sistema. Cada parámetro contiene un nombre de variable y su valor asociado.

Sección **CLASSES**: contiene información para las clases definidas por la aplicación, cuyas instancias aparecen en las secciones **BLOCKS**, **ENTITIES**, y **OBJECTS** de la base de datos. Una definición de clase está permanentemente fija en la jerarquía de clases.

Sección **TABLES**: contiene definiciones para las tablas de símbolos **APPID** (application identification table) **BLOCK_RECORD** (block reference table) **DIMSTYLE** (dimension style table) **LAYER** (layer table) **LTYPE** (linetype table) **STYLE** (text style table) **UCS** (User Coordinate System table) **VIEW** (view table) **VPORT** (viewport configuration table).

Sección **BLOCKS**: contiene definiciones de bloques y entidades que conforman cada referencia de bloque en el dibujo.

Sección **ENTITIES**: contiene los objetos gráficos de la escena, incluyendo las referencias de bloque (entidades insertadas).

Sección **OBJECTS**: aquí están los objetos no gráficos de la escena. Todo lo que no sea entidades o tablas está almacenado aquí. Por ejemplo, en esta sección están los diccionarios que contienen estilos de mlines y los grupos (**Autodesk.com** 2010).

Ogre DotScene, DotMesh, y DotMaterial.

1.4.5 OGRE DotScene.

También conocido como .scene es un formato de fichero XML estandarizado. Está hecho para usarse en la inicialización de una escena en Ogre. Es muy útil para cualquier tipo de aplicaciones o juegos. La estructura está basada en nodos, cada uno de los cuales representan los diferentes elementos que pueden estar presentes en la escena. No contiene ningún dato de las mallas, texturas, etc. (**Ogre3D.org** 2009).

Un ejemplo simple de un fichero .scene:

```
<scene formatVersion="">
    <nodes>
        <node name="Robot" id="3">
            <position x="10.0" y="5" z="10.5" />
            <scale x="1" y="1" z="1" />
            <entity name="Robot" meshFile="robot.mesh" static="false" />
        </node>
        <node name="Omni01" id="5">
            <position x="-23" y="49" z="18" />
            <rotation qx="0" qy="0" qz="0" qw="1" />
            <scale x="1" y="1" z="1" />
            <light    name="Omni01"    type="point"    intensity="0.01"
contrast="0">
                <colourDiffuse r="0.4" g="0.4" b="0.5" />
                <colourSpecular r="0.5" g="0.5" b="0.5" />
            </light>
        </node>
    </nodes>
</scene>
```

Los datos de las mallas, texturas, etc. están contenidos en otro tipo de ficheros, de los que el .scene se auxilia para guardar la información de los objetos. Estos son el Dotmesh(.mesh), y el Dotmaterial(.material). En el .mesh se almacena la información física de las mallas, dígase vértices, posición, dirección de las normales, etc., y en el .material se encapsulan las coordenadas de texturas de los objetos presentes en la escena (**Ogre3D.org** 2009). Es importante señalar que aunque el .material es un fichero ASCII, el .mesh es un fichero de tipo binario.

1.5 Herramientas de diseño 3D utilizadas para el diseño de escenarios virtuales.

Para la modelación de escenarios virtuales se utilizan varias herramientas de diseño 3D, dependiendo del fin que se busca lograr. Por ejemplo, en el mercado de los videojuegos es muy popular el 3ds Max Studio, por sus herramientas para ello. Asimismo, el Maya es de la preferencia de los animadores y productores de cine en general. El Blender, como herramienta basada en software libre, está ganando más adeptos cada día, y se perfila como una herramienta con un futuro prometedor. A continuación se detallan las características fundamentales de dichas herramientas, por ser las más usadas en Centro de Informática Industrial (CEDIN), de la Universidad de las Ciencias Informáticas.

1.5.1 Autodesk 3D Studio Max.

Las herramientas de modelado, animación, rendering, y composición del 3ds Max (como también se le conoce) están diseñadas para desarrolladores de juegos, artistas de efectos visuales, y diseñadores gráficos que trabajen con juegos, aunque el software es también usado por productores de televisión y cine. Estas son sus características principales:

Modelado 3D:

- Tiene uno de los paneles de herramientas más ricos en contenido que hay actualmente en la industria:
- Creación eficiente de objetos orgánicos y paramétricos con características de modelado de polígono, splines (líneas), y basado en NURBS[8].

[8] Acrónimo inglés de la expresión *Non Uniform Rational B-splines.* Es un modelo matemático muy utilizado en la computación gráfica para generar y representar curvas y superficies.

- Más de 100 herramientas de diseño de modelado poligonal y libre presentes en el set de herramientas Graphite.
- Control preciso del número de caras y vértices en los objetos con tecnología ProOptimizer y reducción de complejidad de una selección de hasta un 75% sin pérdida de detalle (**Autodesk.com** 2011).

3ds Max SDK:

- El SDK (Software Developer Kit) puede ser usado para extender e implementar prácticamente cada aspecto de la aplicación 3ds Max, incluyendo la geometría de la escena, controladores de animación, efectos de cámara y atmosféricos, crear nuevos componentes de escena, controlar el comportamiento de componentes existentes, y exportar los datos de la escena a un formato personalizado. Los desarrolladores pueden explotar un nuevo cargador administrado de plug-ins .NET, haciendo más fácil desarrollar plug-ins en C# u otros lenguajes .NET (**Autodesk.com** 2011).

Renderizador de Hardware Quicksilver:

- Un innovador, nuevo renderizador basado en hardware que puede producir imágenes de alta calidad a altas velocidades. Este nuevo motor de render multi-hilo presente en 3ds Max 2011 usa tanto la unidad de procesamiento de la computadora (CPU, como la unidad de procesamiento de gráficos(GPU), y soporta elementos de render alpha y z-buffer; profundidad de campo, motion blur(desenfoque de movimiento), reflexiones dinámicas, oclusión de área, fotométrica y ambiental, efectos de iluminación indirecta junto a mapas de sombras de precisión adaptativa; y la capacidad de producir imágenes a altas resoluciones (**Autodesk.com** 2011).

1.5.2 Autodesk Maya.

Maya, aunque es desarrollado también por Autodesk, está más orientado a la producción audiovisual. La diferencia comienza en la interfaz visual, que está optimizada para esos fines, y continúa en las diferentes herramientas que, aunque comparten el mismo productor, tienen

equipos de desarrollo que optimizan más en cada versión las características del programa. A continuación se detallan algunas de estas características:

Interfaz de usuario mejorada:

- Este software de animación está disponible en versiones para Mac OS X, Windows de 32 y 64 bits, y Linux de 64 bits, con una interfaz que ofrece elementos anclables y editores más flexibles. Además, un nuevo examinador de nodos dentro del Hypershade que muestra los nodos por categoría en una vista de árbol con búsqueda y que provee más fácil acceso a los nodos usados frecuentemente (**Autodesk.com** 2011).

Modelado 3D:

- Ofrece un set poderoso para el modelado 3D, incluyendo NURBS, Superficies de Subdivisión, y un conjunto de herramientas para polígonos (**Autodesk.com** 2011).

Animación general:

- Maya entrega una amplia gama de herramientas especializadas para la animación por keyframes, procedural, y a base de scripts, incluyendo:
- Un sistema de capas altamente controlable, no destructivo, que funciona con cualquier atributo del Editor de Animaciones No Lineales Trax, para mezcla, combinación, y edición no destructiva de poses y clips animados.
- Una herramienta Establecer Llave Conducida que permite hacer fotogramas clave relaciones complejas entre parámetros animados.
- Editores Graph y Dopesheet que proveen funciones precisas de curvas para controlar cómo los atributos animados cambian en el tiempo.
- Un conjunto de deformadores para modelado estático o animación (**Autodesk.com** 2011).

Herramientas de administración de datos y escenas:

- A medida que la complejidad de las escenas se incrementa, Maya provee herramientas y flujos de trabajo para administrar más eficientemente grandes conjuntos de datos.

- Un grafo de dependencia permite al artista ver y editar las relaciones entre los nodos. Mediante las referencias de valores y archivos el artista puede también segmentar escenas para mejor rendimiento y para administrar flujos de trabajo colaborativos e iterativos.
- El historial de construcción editable y animable permite hacer modificaciones amplias de datos modelados sin necesidad de reconstruirlos.
- A través del Render Proxy, que está dentro del software de rendering mental ray, los elementos de la escena pueden ser reemplazados por una malla simple, de baja resolución: los datos pre-traducidos son cargados sólo cuando se requieran para renderizar (**Autodesk.com** 2011).

1.5.3 Blender.

Blender es un proyecto de la Blender Foundation, alternativa de software libre que va ganando adeptos a medida que mejora y se le añaden nuevas herramientas. A pesar de ser un software relativamente joven, se perfila como una buena alternativa, avalado por la libre distribución de sus licencias. Tiene características que lo sitúan en un lugar destacado, sobre todo por el esfuerzo gratuito de sus desarrolladores:

Interfaz:

- Disposición de ventanas flexible y completamente configurable con tantas configuraciones de pantalla como el usuario prefiera.
- Capacidad de deshacer en todos los niveles.
- Cualquier espacio de ventana puede ser cambiado a cualquier tipo de ventana (editor de curvas, NLA, vista 3D, etc.).
- Editor de texto interno para anotaciones y edición de scripts de Python.
- Interfaz gráfica para scripts de Python.
- Interfaz consistente a través de todas las plataformas (**Blender.org** 2011).

UV Unwrapping:

- Métodos de unwrapping conformales y basados en ángulos.
- Edición de degradado interactivo de mapas UV para transformaciones suaves.
- Unwrapping basado en costuras (seams).
- Proyecciones de vista cúbica, cilíndrica y esférica.
- Subdivisión Catmull-Clark para menor distorsión de los UVs.
- Capas UV multiples (**Blender.org** 2011).

Creación de juegos 3D en tiempo real:

- Editor gráfico lógico para definir comportamiento interactivo win tener que programar.
- Detección de colisiones y simulación de dinámicas soportado para Bullet. Bullet es una librería para detección de colisiones y dinámicas de código abierto desarrollada para Play Station 3.
- API para scripting de Python con control e inteligencia artificial, lógica de juego avanzada completamente definida.
- Soporta todos los modos de iluminación de OpenGLTM, incluyendo transparencias, y texturas animadas y mapeadas por reflexión.
- Soporte para multimateriales, multitexturas y modos de fusión de texturas, iluminación per-pixel, dinámica, modos de mapeo, fusión de texturas vertexPaint GLSL, toon shading, materiales animados, mapeo Normal y Parallax.
- Audio, usando el toolkit SDL (**Blender.org** 2011).

Modelado:

- Superficies de subdivisión Catmull-Clark con isolíneas óptimas.
- Capacidad de modelado multirresolución con pinceles procedurales de mapas 2D y 3D(Paint, Smooth, Pinch, Inflate, Grab), con soporte para simetrías.
- Modificadores de deformación: Lattice, Curve, Armature y Displace.

- Modificador Mirror con clipping para los vértices del centro y eliminación automática de caras interiores.
- Acceso a scripting Python para personalizar las herramientas (**Blender.org** 2011).

1.6 Motores gráficos.

El motor gráfico (Graphic Engine, en inglés), es el componente de software principal de un videojuego o de otra aplicación interactiva que se ejecute en tiempo real. Su uso simplifica el desarrollo de la aplicación y a menudo permite que el juego pueda correr en múltiples plataformas, tales como Consolas de videojuegos y Sistemas Operativos de Mac OS, GNU/Linux y Microsoft Windows. La abstracción del hardware es una de las principales ventajas que posee el Motor. Un Motor Gráfico ofrece un conjunto de herramientas de desarrollo, además de componentes de software reutilizables, agrupados en subsistemas que presentan alta cohesión en relación a sus comportamientos, los subsistemas más comunes son: procesamiento de entradas, gráficos, animación, audio, comportamiento e inteligencia artificial, y conectividad / red, entre otros.

Se toman como objeto de análisis G3D, y Ogre por su popularidad y funcionalidad, adquiridas no sólo en nuestra universidad, sino, también a nivel mundial. También se hace referencia al SceneToolKit, por ser una herramienta nacional, desarrollada y mejorada en nuestro centro.

1.6.1 G3D.

G3D (GraphicsThree Dimensional Engine) es un motor gráfico escrito en C++, que ha sido utilizado en una amplia gama de aplicaciones, entre ellas los videojuegos. Es importante destacar que las APIs gráficas de bajo nivel como OpenGL y Direct3D son muy sencillas de utilizar, no significando esto último una pérdida o limitación en el rendimiento del producto final.(SourceForge.net 2010)

G3D es compatible con varios sistemas operativos como Windows, Linux y OS X. Además de su diseño orientado a objetos y el soporte de varias extensiones de imágenes como JPG, TGA y

PNG; G3D integra dentro de sí el trabajo con shaders y diversas técnicas de dibujado como Shadow Mappingy Shadow Volumes que aportan sin duda un toque de realismo al entorno virtual (**SourceForge.net** 2010).

Principales características:

- Soporte para modelos 3DS, IFS, MD2, BSP, PLY2, OFF.
- Soporte para imágenes JPG, PNG, BMP, PPM, PCX, TGA, DDS, e ICO.
- MP4, MPG, MOV, AVI, DV, QT, WMV, video .
- Themed GUI and fontrendering.
- Administración de memoria automática de forma opcional.
- Red basada en los protocolos TCP y UDP.
- Optimización de cálculos con matrices.
- Compatible con Visual C++, XCodeygcc.
- Amplia documentación que incluye también programas de ejemplo.

1.6.2 OGRE (Object Oriented Graphics Engine).

OGRE es un motor gráfico de código abierto, multiplataforma escrito en C++. Cuenta con una gran comunidad internacional y una amplia documentación; además de permitir el acople de nuevos plugins incrementando su flexibilidad (**Ogre3D.org** 2009).

A continuación se expondrán alguna de sus principales características.

Texturas:

- TextureMapping. Cálculo automático de Mip-Maps.
- Texturas multinivel en un solo paso.
- Texturas animadas.
- Cálculo procedural de coordenadas de textura.

Iluminación:

- Luces puntuales, direccionales y de foco.
- Nieblas.
- Luces dinámicas, de colores con sombras suaves.

- Radiosidad precalculada sobre los ligthmaps.

Modelado:

- Mallas poligonales con soporte de LODs discretos.
- Superficies de Bezier.
- Grafo jerárquico de escena.
- Edición externa mediante 3D Studio MAX.
- Modelado de terreno.
- Importación de niveles de Quake3.
- Terrains.

Estructura:

La estructura de Ogre, como se puede ver en la figura 4 da al desarrollador un fácil manejo de los plugins, además de ser una estructura bien modulada.

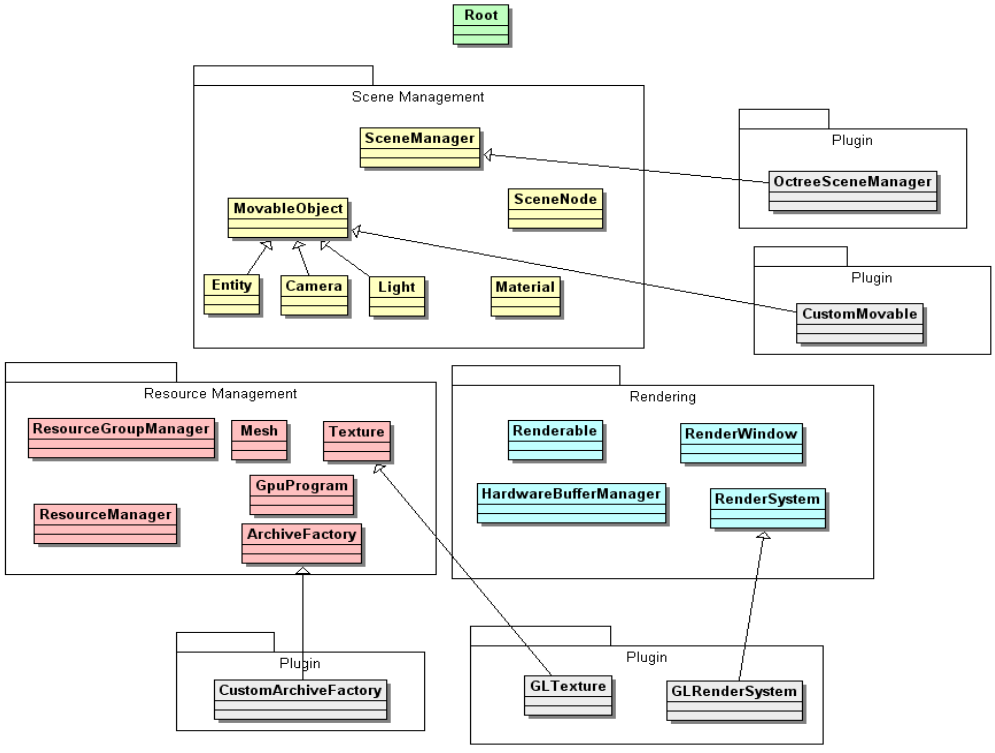

Fig. 4 Estructura de Ogre (Ogre3D.org 2009).

1.6.3 SceneToolKit (STK).

La SceneToolkit[9] es una herramienta que abstrae al desarrollador del trabajo con APIs de bajo nivel como son OpenGL y DirectX, facilitando de esta manera la representación de entornos virtuales y simulaciones. Hasta su última versión publicada, dicha herramienta no contaba con un motor de render que tuviera una estructura bien definida además de carecer de las técnicas más recientes que permiten un incremento considerable del rendimiento (**Quintana López** and **Alonso Monteagudo** 2010).

El sistema de render actual basa el dibujado de todas sus geometrías en vertexarrays, lo cual representa un paso de avance con respecto al modo inmediato de render (inmediatemode); además el procesamiento innecesario de cambios en los estados de render se traduce en un decremento en cuanto a la eficiencia del sistema. Otra deficiencia lo constituye la no posibilidad de integración de shaders dentro del sistema; que si bien estos se han tratado de incluir en versiones no estables, no han sido establecidos de forma definitiva (**Quintana López** and **Alonso Monteagudo** 2010) .

Por otra parte la SceneToolKit brinda soporte para las APIsOpenGL y DirectX, además de ser compatible con múltiples sistemas operativos (Windows y Linux) facilitando en cada uno de ellos la interacción con ventanas y periféricos (mouse, teclado, joystick). Su arquitectura dividida por módulos permite la integración y/o acoplamiento de nuevos módulos sin necesidad de una restructuración a gran escala del sistema completo (**Quintana López** and **Alonso Monteagudo** 2010).

[9] Toda la información referente a la SceneToolKit es tomada de documentos del expediente de proyecto.

Clases más significativas.

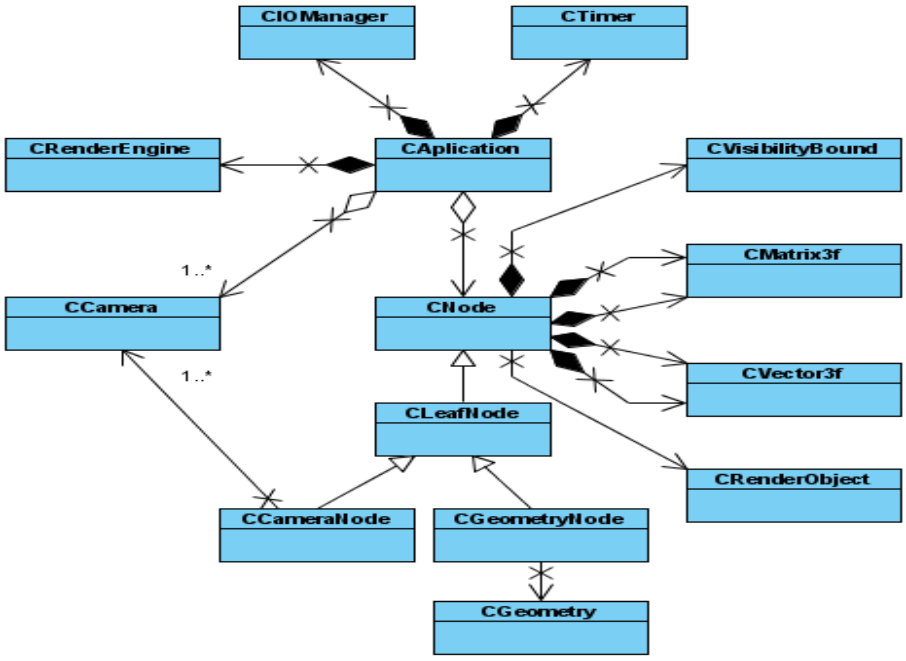

Fig. 5 Principales clases del SceneToolKit.

1.7 Entornos de Desarrollo Integrado.

Un entorno de desarrollo integrado (en inglés Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones (**Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo** 2009).

1.7.1 Microsoft Visual Studio.

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones Web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones

móviles. Visual Basic, Visual C++, Visual C# y Visual J# utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y Servicios Web XML (**Microsoft**).

Acelera de manera significativa la producción de software y su documentación está entre las mejores. La interfaz es altamente amigable con el usuario, permitiendo que el tiempo en implementar una solución o aplicación determinada sea mucho menor. Los ejecutables desarrollados por esta herramienta son generalmente de menor tamaño, lo que hace que ocupe un lugar cimero en la producción de software al lograr aplicaciones óptimas y de poco volumen.

1.7.2 Qt Creator.

Qt Creator es un entorno de desarrollo multiplataforma de código abierto muy completo, hecho para desarrollar aplicaciones en C++ de manera sencilla y rápida. Como su nombre lo indica, está basado en la librería Qt y cuenta con las siguientes características:

- Editor de código C++ y Javascript.
- Diseñador de interfaz de usuario integrado.
- Herramientas de administración y construcción de proyectos.
- Debuggers DGB y CDB.
- Soporte para control de versiones.
- Simulador para interfaces de dispositivos móviles.
- Soporte para destinos móviles y de escritorio (**Nokia** 2011).

Qt Creator es distribuido bajo tres tipos de licencias: Qt Commercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y está disponible para las plataformas: Linux, Mac OSX; Windows, Windows CE, Symbian y Maemo.

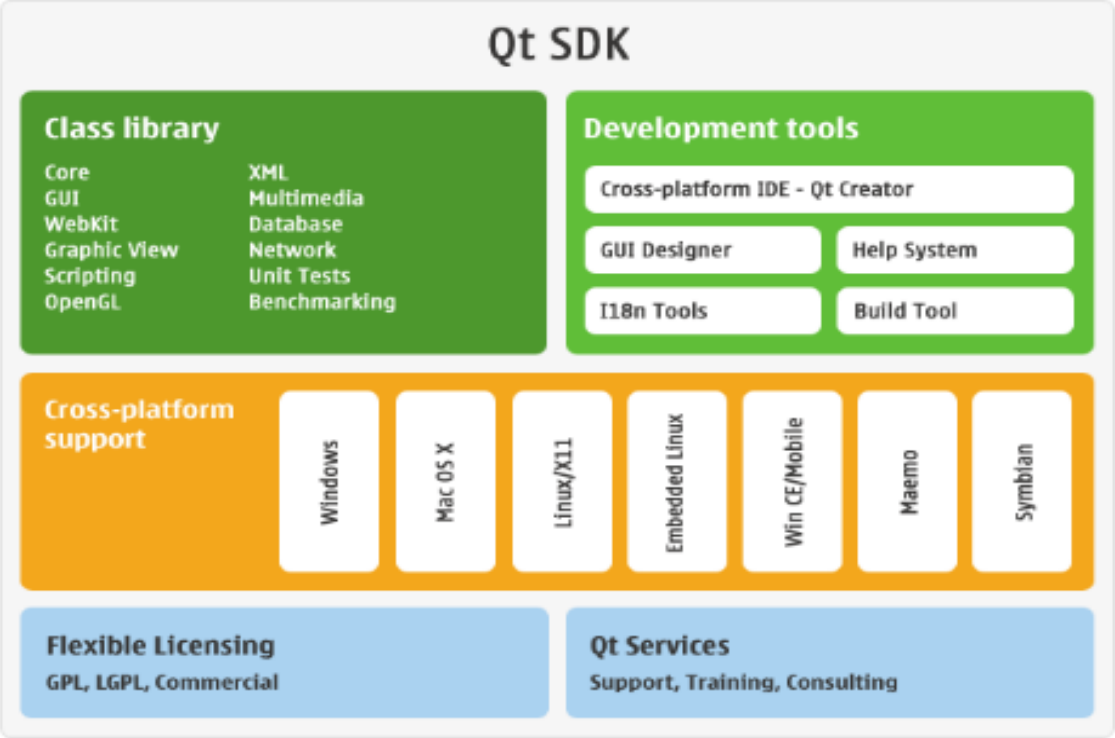

Fig. 6 Características del SDK Qt

1.7.3 Eclipse.

El IDE Eclipse para desarrolladores de C/C++ es de código abierto y corre sobre la Plataforma Eclipse. Provee funcionalidades avanzadas para desarrolladores de C/C++, que incluyen un editor (con realzamiento de sintaxis y completamiento de código), launcher, debugger, un motor de búsquedas y generador de ficheros (**Eclipse.org** 2011). No incluye compilador ni debugger, es necesario instalarlos de forma independiente.

1.8 Lenguajes de modelado, programación y framework.

1.8.1 Lenguaje Unificado de Modelado (UML).

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Aun cuando todavía no es un estándar oficial, está respaldado por el OMG (Object Management

Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional, o RUP, por sus siglas en inglés), pero no especifica en sí mismo qué metodología o proceso usar (**RedIRIS.com** 2009). En nuestro caso, es el lenguaje que se utilizó en el flujo de diseño de la herramienta, por eso su obligatoria utilización.

1.8.2 C++.

C++ es un lenguaje orientado a objetos de probada eficacia para generar aplicaciones de alto rendimiento como las aplicaciones gráficas. Aporta un nivel de productividad muy superior a C, sin comprometer la flexibilidad, el rendimiento o el control. Es uno de los lenguajes de sistemas más conocido en el mundo, altamente compatible, y que soporta las bibliotecas gráficas Glide, OpenGL, DirectX y G3D que son muy utilizadas en la industria de los videojuegos. Al hacer uso del paradigma de la programación orientada a objetos, son considerables las ventajas que este lenguaje ofrece (**Rodríguez Noa** and **Gómez Finalé** 2008) .

1.8.3 Framework Qt

Qt es una aplicación multiplataforma y un framework para interfaces de usuario (UIs, por sus siglas en inglés). A través de ella, se pueden escribir aplicaciones web y desplegarlas para escritorio, sistemas operativos móviles e integrados, sin tener que reescribir el código fuente.

Incluye una librería de clases de C++, portabilidad para sistemas operativos integrados y de escritorio, herramientas para desarrolladores con un IDE multiplataforma, y un alto rendimiento en tiempo real (**Nokia** 2011).

Consideraciones del capítulo.

A través del estudio de este capítulo, se pueden obtener los conocimientos necesarios para comprender todo lo referente a la solución que se da en este trabajo para el problema de la falta de un módulo de persistencia y visualización para una herramienta de desarrollo de Centros Expositivos Virtuales.

Se hizo un estudio de los principales formatos de ficheros 3D, las herramientas utilizadas para el diseño de escenarios virtuales, los motores gráficos utilizados para la visualización de escenarios en 3D, así como las herramientas de modelado y programación necesarias para llevar a cabo una aplicación funcional.

Capítulo 2. Características del sistema.

Introducción al capítulo 2.

En el presente capítulo se hará un resumen de la fase de Análisis y Diseño, cuyos resultados le fueron entregados al autor para la implementación del módulo. Se definirán los requisitos, tanto funcionales como no funcionales, que deberá cumplir el módulo, así como los Casos de Uso correspondientes al diseño del mismo, cuyos diagramas se presentan. Por último, se presentarán los modelos conceptuales, dígase diagramas y descripción de clases, realización de Casos de Uso y prototipo de interfaz.

2.1 Especificación de los requisitos de software.

Los requisitos son una especificación de lo que el sistema debe cumplir para satisfacer al cliente. Estos pueden dividirse en requisitos funcionales y requisitos no funcionales. En los próximos epígrafes se especifican los requisitos tanto funcionales como no funcionales para cada uno de los módulos que conforman la herramienta que se propone.

2.1.1 Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir.

Tabla 1. Requisitos funcionales.

N^{o}	Nombre	Descripción
RF 1	Inicializar Datos.	Lista en el visualizador los nombres de los paseos previamente creados en el módulo Editor que se encuentren en la carpeta de proyectos de la aplicación.
RF 2	Definir modo de paseo.	Define la forma en que se realizará el recorrido por el centro expositivo virtual, el recorrido puede realizarse de forma libre (utilizando los controles del teclado) o dirigido (se muestra el centro virtual a través del recorrido de la cámara).

RF 3	Editar cámara.	Modifica la velocidad y la altura de la cámara mediante los dispositivos de entrada, de esta forma es posible realizar el recorrido por el centro virtual.
RF 3.1	Modificar velocidad.	Disminuye o aumenta la velocidad con que se moverá la cámara en el paseo.
RF 3.2	Modificar altura.	Disminuye o aumenta la altura a la que estará la cámara durante el paseo.
RF 3.3	Pausar paseo.	Disminuye la velocidad de la cámara a valor cero.
RF 4	Mostrar información asociada a un objeto.	Muestra la información asociada a un objeto en una pequeña ventana al hacer clic con el mouse encima del mismo.
RF 5	Resaltar objetos expositivos.	Al pasar el mouse por encima de un objeto el puntero cambia de forma indicando que puede ser seleccionado.
RF 6	Comenzar paseo.	Carga todos los elementos: modelos, luces, recorridos, cámara del paseo seleccionado y lo muestra en pantalla completa.
RF 7	Cerrar paseo.	Cierra la vista previa del paseo virtual y se muestran el resto de las ventanas que conforman el Editor.
RF 8	Mostrar ayuda del paseo.	Muestra información sobre cómo realizar el recorrido por el entorno virtual.

2.1.2 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Para la herramienta a desarrollar se especifican los siguientes requisitos no funcionales:

Requisitos no funcionales.

1. Usabilidad.

RnF 1.1 Utilización del sistema. Los usuarios del sistema serán visitantes interesados en observar un paseo virtual determinado con habilidades en la manipulación del teclado y mouse (ratón) de una computadora.

2. Rendimiento.

RnF 2.1 Velocidad de representación de imágenes. Representar las imágenes o cuadros (frame) a una velocidad mínima de 30 frames por segundo.

3. Portabilidad.

RnF 3.1 Ejecución de la herramienta. La herramienta permitirá ser compilada y ejecutada sobre la plataforma Windows y Linux.

4. Restricciones de diseño.

RnF 4.1 Lenguajes de programación. Se utilizará el lenguaje de programación C++.

RnF 4.2 Herramientas de desarrollo. Para la implantación del sistema se utilizará el IDE QT Creator.

RnF 4.3 Herramientas de diseño gráfico. Se utilizará la herramienta Blender para la modelación de gráficos.

RnF 4.4 Implementación del sistema. Se regirá por la filosofía de Programación Orientada a Objetos y se utilizará el motor gráfico Ogre.

5. Hardware

RnF 5.1 Instalación de la herramienta. Para la instalación del software la computadora destino deberá cumplir con los siguientes requisitos.

- Memoria RAM de 1 Gb o Superior.
- Velocidad de Microprocesador a 3.0 GHz o superior.
- Recursos de Video de 128 Mb o superior.

6. Interfaz.

RnF 6.1 Interfaz de Usuario. Para facilitar el uso la interfaz de la herramienta las opciones de servicios se representarán con un icono identificador y al colocar el ratón encima de alguno de estos icono debe mostrarse un texto que muestre la opción en cuestión con su función descrita en una frase breve para un reconocimiento rápido por el usuario

7. Ayuda y documentación en línea.

RnF 7.1 Manual de ayuda. Contará con un manual de ayuda con información referente a cómo utilizar el teclado y el mouse (ratón) para desplazarse por el paseo virtual y acceder a la información de los objetos.

2.2 Definición de los casos de uso.

Una vez recopilados los requisitos se pueden definir los casos de uso los cuales facilitaran una descripción de cómo el sistema se usará. El caso de uso describe la manera en que los actores interactúan con el sistema.

Definición de los actores.

Un actor representa papeles que las personas (o dispositivos) juegan como impulsores del sistema.

Actores	**Justificación**
Diseñador del Paseo.	Es la persona que interactúa con el módulo Editor de la herramienta para crear o editar un paseo virtual a un centro expositivo.

Listados de casos de uso (CU):

CU-1. Inicializar Datos.
CU-2. Definir modo de paseo.
CU-3. Editar cámara.
CU-4. Mostrar información dun objeto.
CU-5. Resaltar objetos expositivos.
CU-6. Comenzar paseo.
CU-7. Cerrar paseo.
CU-8. Mostrar ayuda del paseo.

2.2.1 Diagrama de casos de uso.

La representación de los diagramas de casos de uso permite especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los actores del sistema.

En las siguientes figuras se muestra la vista de casos de uso para el módulo Visualizador que representan los casos de usos arquitectónicamente significativos. En el Anexo 1 del documento se podrán observar los diagramas de casos de uso representando todos los casos de uso.

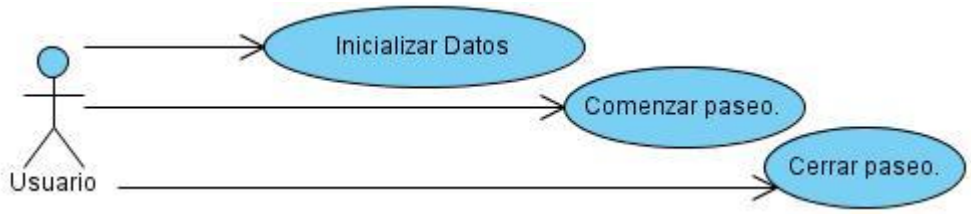

Fig. 7. Diagrama de casos de uso arquitectónicamente significativos del módulo Editor.

2.2.2 Descripción textual de los casos de usos.

Mediante la descripción textual de un caso de uso se describe paso a paso la secuencia de eventos que los actores realizan para completar un proceso a través del sistema. Se especifican las acciones que realizan los actores sobre el sistema y la respuesta que surge como consecuencia de cada evento. Seguidamente se muestran las descripciones textuales de los casos de usos arquitectónicamente significativos. Pueden verse en el Anexo 2 las descripciones textuales de todos los casos de uso definidos para la herramienta.

Tabla 2. Descripción textual del CU Inicializar Datos.

Caso de uso	
CU-1	Inicializar Datos.
Propósito	El objetivo del caso de uso es que el usuario pueda disponer de todos los paseos virtuales que posea el visualizador en su carpeta de proyectos.
Prioridad	Crítico.
Actores: Usuario.	
Resumen: El usuario inicializa el caso de uso cuando ejecuta el Visualizador. El Visualizador lista los paseos previamente creados en el módulo Editor que se encuentren en la carpeta de proyectos de la aplicación para posibilitar la selección del paseo que desee visualizar el usuario.	
Referencias	RF 1.
Precondiciones	El paseo debe estar creado con el módulo Editor.
Poscondiciones	Se muestra en el área de visualización el paseo virtual cargado.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario ejecuta el Visualizador.	1.1 Se abre el sistema, y se listan los paseos que se encuentren en la carpeta de proyectos de la aplicación.

Tabla 3. Descripción textual del CU Comenzar Paseo.

Caso de uso	
CU-6	Comenzar paseo.
Propósito	El objetivo del caso de uso que el usuario pueda visualizar el paseo en pantalla completa.
Prioridad	Crítico.
Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario selecciona de la lista de paseos disponibles el que desea visualizar.	
Referencias	RF 6.
Precondiciones	El paseo debe estar disponible en la lista de paseos del visualizador.
Poscondiciones	Se muestra el paseo en pantalla completa.
Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un paseo virtual de los disponibles en la lista de paseos del visualizador.	1.1 El sistema carga los recursos (luces, cámara, recorridos, modelos) del paseo seleccionado y lo muestra en pantalla completa. Finaliza así el caso de uso.

Tabla 4. Descripción textual del CU Cerrar Paseo.

Caso de uso

CU-7	Cerrar paseo.
Propósito	El objetivo del caso de uso que el usuario pueda visualizar la pantalla inicial del Visualizador y el resto de las opciones.
Prioridad	Crítico.
Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario selecciona la opción cerrar paseo.	
Referencias	RF 7.
Precondiciones	El paseo debe estar cargado en el área de visualización.
Poscondiciones	Se muestra la pantalla inicial del Visualizador.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción cerrar paseo.	1.1 El sistema cierra el paseo virtual que se encuentra en ejecución, visualizando la pantalla inicial y mostrando el resto de las opciones del Visualizador. Finaliza el caso de uso.

2.3 Modelos conceptuales de la herramienta.

El modelo conceptual que se muestra a continuación es una forma representar los conceptos significativos asociados a la propuesta de solución para su mejor comprensión.

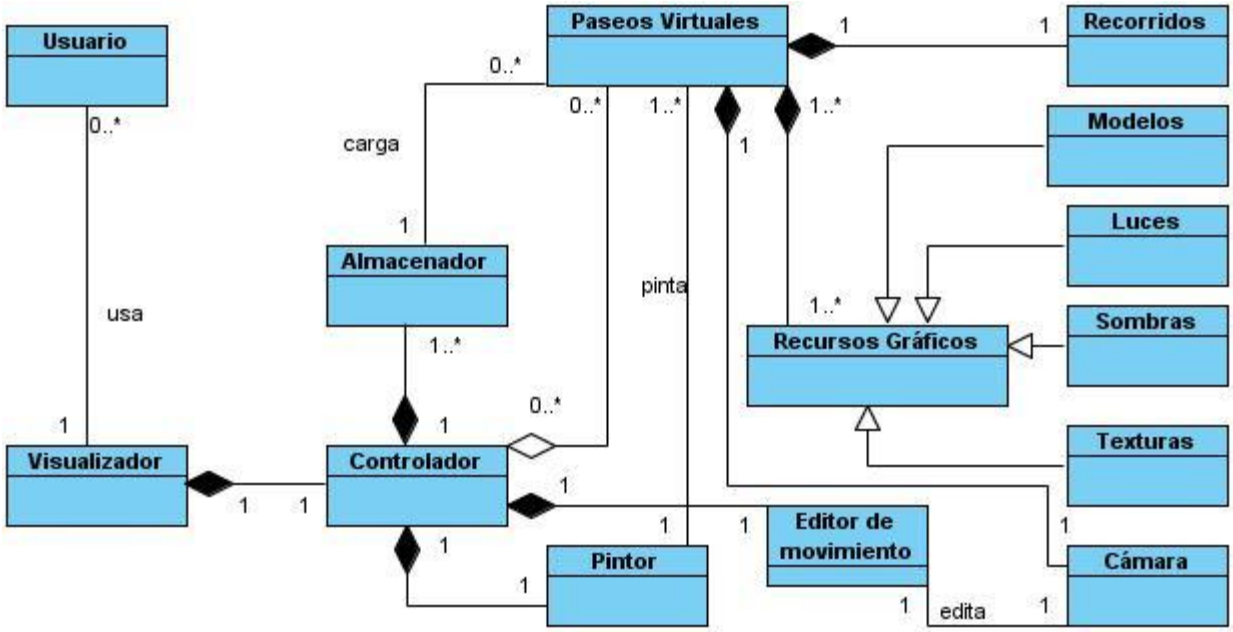

Fig. 8. Modelo conceptual del módulo Visualizador.

2.3.1 Diagramas de clases.

Los diagramas de clases se utilizan para mostrar clases y sus relaciones, para saber las clases que participaran en la realización de un caso de uso y las responsabilidades que deberán cumplir. El diagrama de clases del diseño incluye más detalles que el diagrama de clases del modelo de análisis lo cual es necesario para la adaptación del modelo de diseño al entorno de la implementación. Las clases que intervienen en la realización de los casos de uso de los módulos de la herramienta estarán estructuradas según el patrón tres capas, ya que permite simplificar y comprender el desarrollo del sistema reduciendo la dependencia de forma que las capas más bajas no sean conscientes de ningún detalle de las superiores.

Clases que se encuentran en la capa Presentación.

Clases que se encuentran en la capa Aplicación.

Clases que se encuentran en la capa Acceso a Datos.

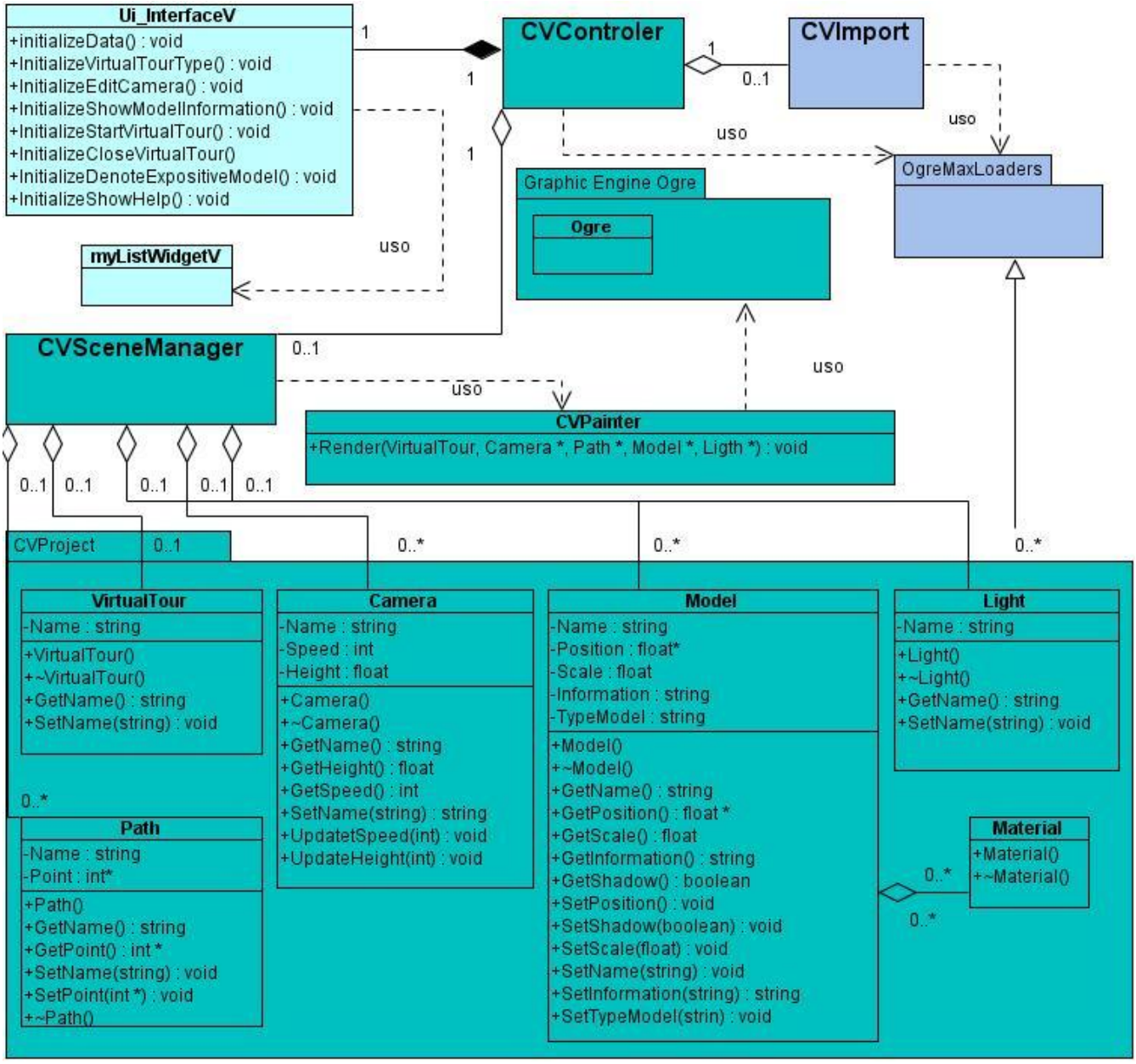

Fig. 9. Diagrama de clases del diseño del módulo Visualizador.

Los diagramas de clases del diseño que se muestran en la Figura 9 representan las clases, sus relaciones y sus responsabilidades.

2.3.2 Descripción de las clases.

A continuación se realiza una breve descripción de las clases arquitectónicamente significativas para el diseño de la herramienta. Clases arquitectónicamente significativas del módulo Visualizador.

“CVControler”. Clase controladora que tiene la responsabilidad de enviar las respuestas a las peticiones que realiza el usuario hacia la clase Interfaz. Contiene todas las funcionalidades que dan respuesta a los casos de uso del sistema.

“CVImport”. Clase controladora que tiene la responsabilidad de administrar los ficheros que almacenan los datos persistentes del sistema.

“CVSceneManager”.Clase controladora que tiene la responsabilidad de administrar los datos de las entidades que maneja el sistema.

2.3.3 Realización de casos de uso.

La realización de los casos de uso se describe durante el análisis y el diseño a través de los diagramas de colaboración y los diagramas de secuencia. Los diagramas de colaboración se utilizan en el modelo de análisis mostrando cómo el control pasa de un objeto a otro a medida que se lleva a cabo el caso de uso, y los mensajes que se envían entre los objetos. El nombre del mensaje indica el motivo del objeto que realiza la llamada en su interacción con el objeto invocado. La realización de un caso de uso en el modelo de diseño describe como se realiza el caso de uso en términos de las clases de diseño correspondientes. Para modelar las interacciones entre los objetos del diseño se utiliza el diagrama de secuencia. En las siguientes figuras se representan los diagramas de secuencia de los casos de uso arquitectónicamente significativos que fueron realizados durante el diseño. Para cada escenario de un caso de uso se tiene un diagrama de secuencia.

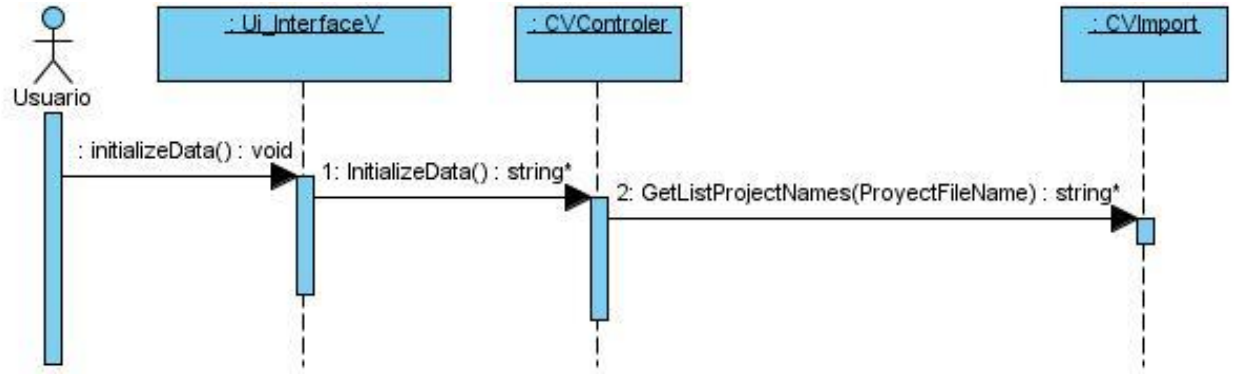

Fig. 10. Diagrama de secuencia CU-1 Inicializar Datos.

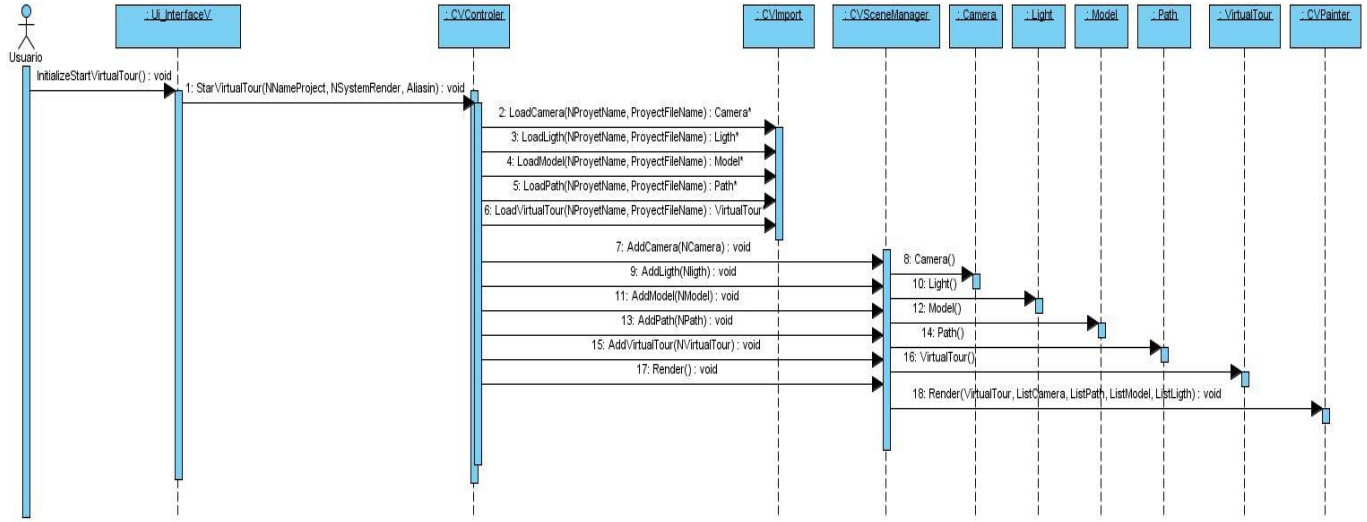

Fig. 11. Diagrama de secuencia CU-6 Comenzar Paseo.

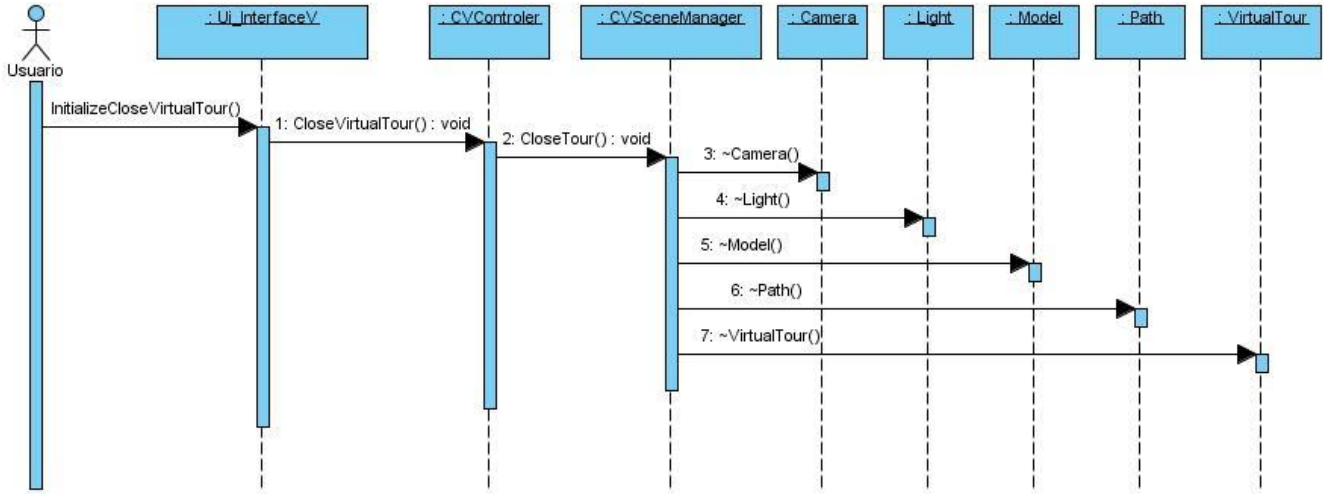

Fig. 12. Diagrama de secuencia CU-7 Cerrar Paseo.

2.3.4 Prototipo de interfaz.

En las Figuras 22 y 23- se muestra la interfaz de usuario del módulo Visualizador, se describen a continuación los componentes de la interfaz.

Figura 22-A: Área de Proyectos, componente que permite listar los nombres de los proyectos que se encuentran en la carpeta del sistema.

Figura 22-B: Área de Ayuda, área de visualización inicial de la ayuda.

Figura 23-A: Área de Edición de Cámara, componente que permite cambiar las opciones de la cámara (velocidad y altura), o cerrar el paseo actual.

Figura 23-B: Área de Recorrido, componente que permite escoger el tipo de recorrido (manual o automático), y la trayectoria a seguir.

Figura 23-C: Área de Render, componente de visualización del Paseo Virtual.

Figura 23-D: Área de Información, área de visualización de la información asociada a un objeto.

A continuación se mencionan los componentes de la propuesta de interfaz de usuario que dan respuesta a los casos de uso arquitectónicamente significativos del módulo Visualizador:

CU-1. Inicializar Datos y CU-6. Comenzar paseo.

La aplicación da respuesta mediante los componentes que se encuentra en el área A (Proyectos), y en el área B (Render). Ver Figura 22-A, 23-B.

CU-2. Definir modo de paseo.

La aplicación da respuesta mediante los componentes que se encuentra en el área B (Recorrido). Ver Figura 23-B.

CU-3. Editar cámara.

La aplicación da respuesta mediante los componentes que se encuentra en el área A (Edición de Cámara). Ver Figura 23-A.

CU-4. Mostrar información asociada a un objeto.

La aplicación da respuesta mediante los componentes que se encuentra en el área D (Información). Ver Figura 23-D.

CU-8. Mostrar ayuda del paseo.

La aplicación da respuesta mediante los componentes que se encuentra en el área B (Ayuda). Ver Figura 22-B.

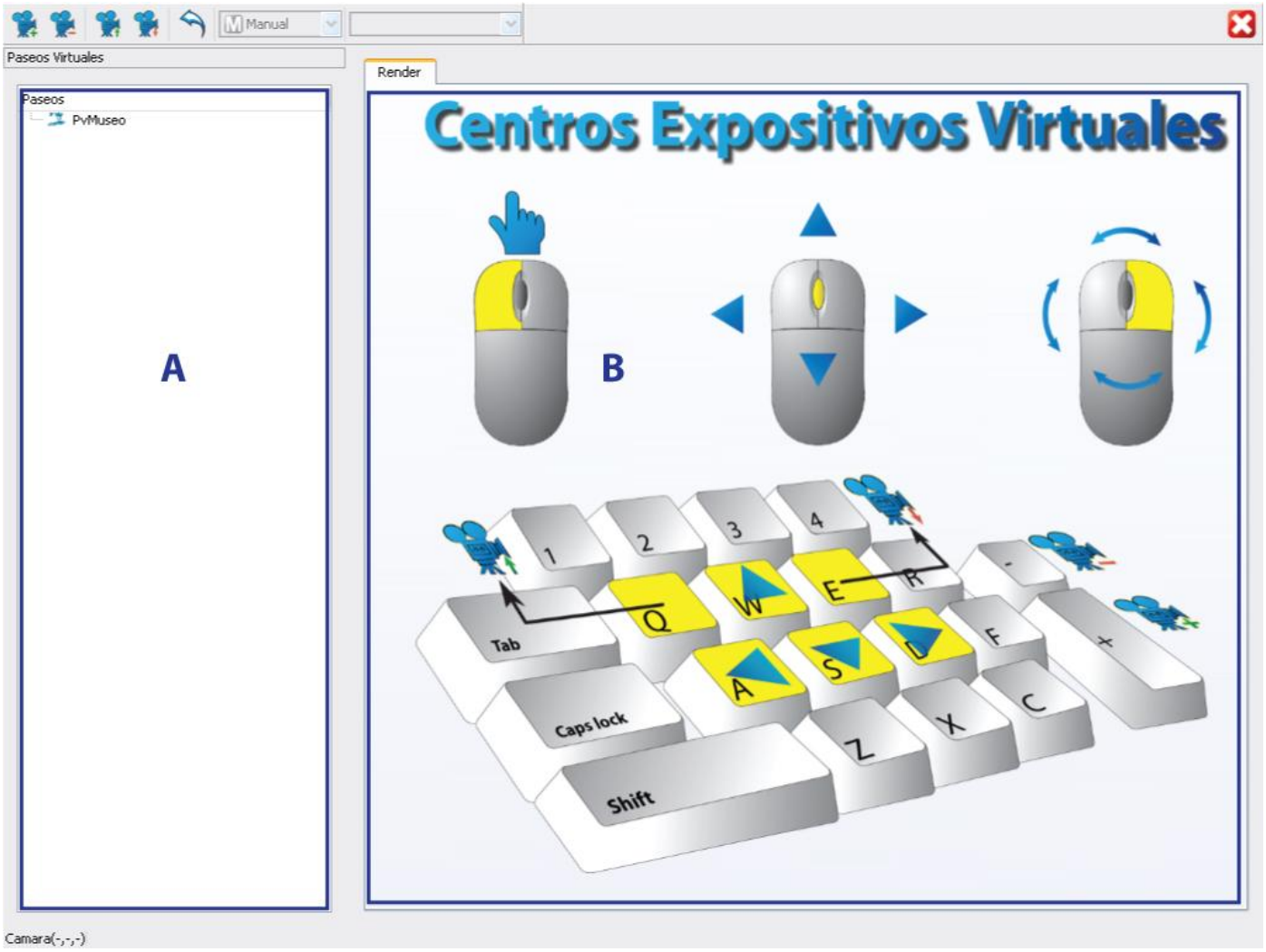

Fig. 13. Interfaz Gráfica del Visualizador.

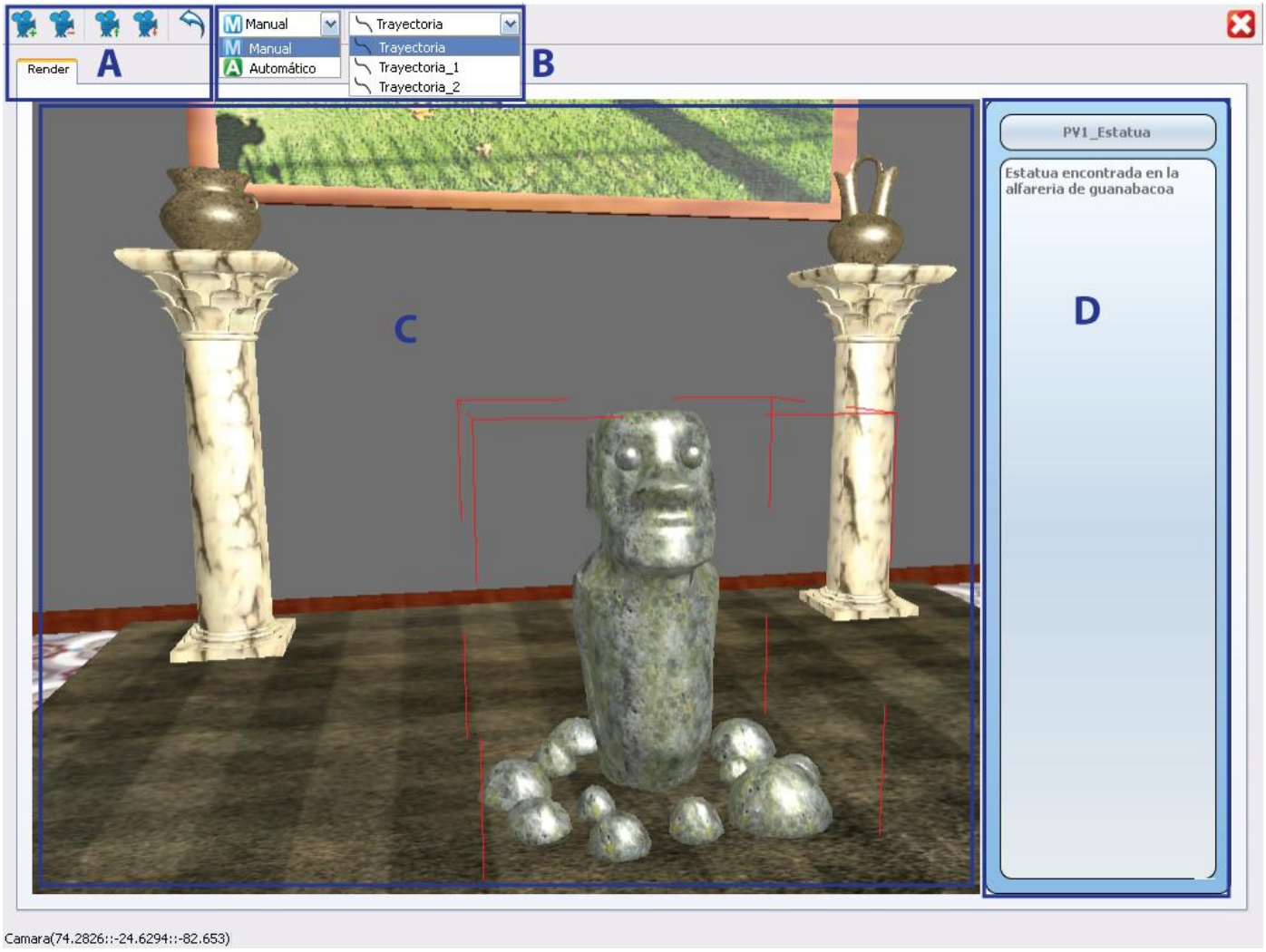

Fig. 14. Interfaz Gráfica del Visualizador.

2.3.5 Modelo de datos

Como solución al problema de la persistencia, se diseñó un modelo de datos, basado en el formato DotScene de Ogre, que guardará los cambios efectuados en el escenario. Además de los datos de la escena, este guardará los tipos de objetos (expositivo, decorativo o delimitador), la información asociada a cada objeto, y un conjunto de puntos que definirán el spline por el cual se recorrerá el Paseo Virtual una vez se cargue en el módulo de visualización. La estructura general del formato es la siguiente:

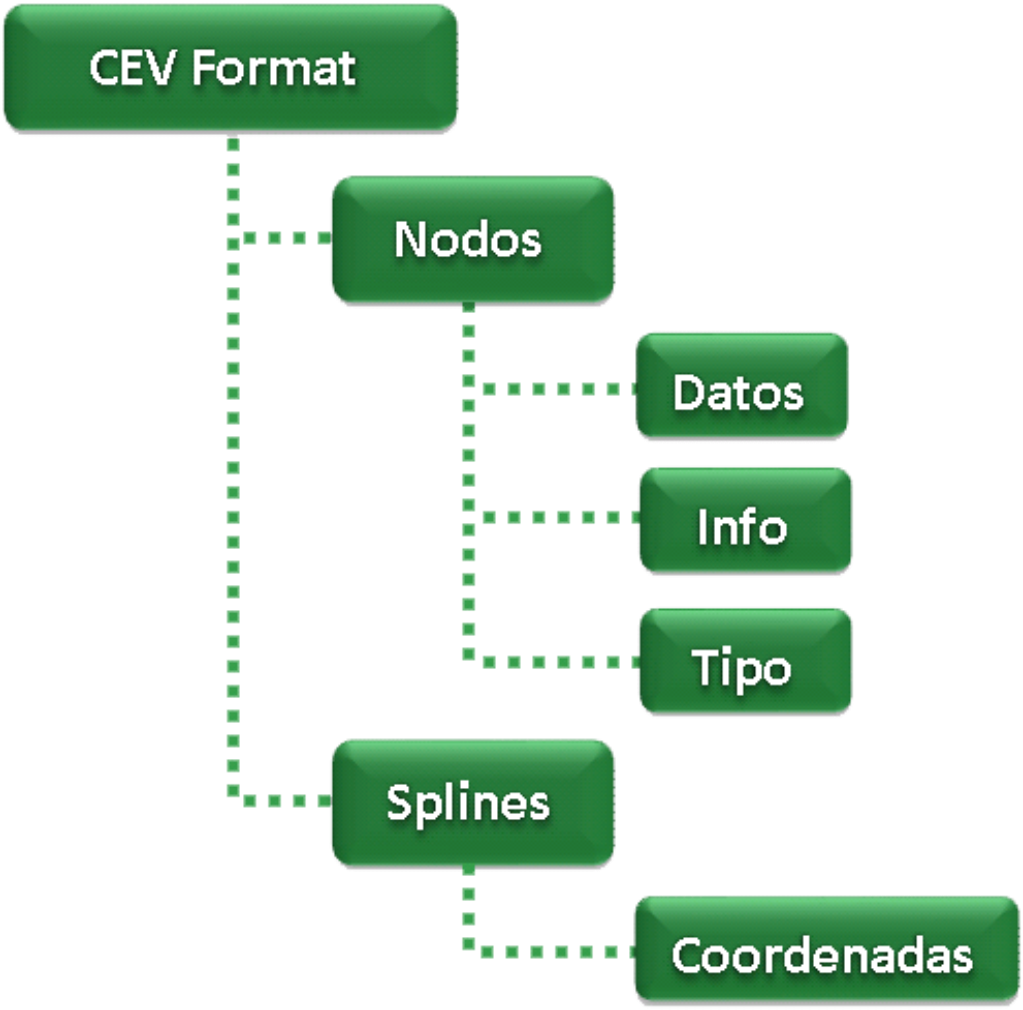

Fig. 15. Estructura del fomato CEV.

Un ejemplo de archivo estándar sería el siguiente:

```
<CEV Format info="Universidad de las Ciencias Informáticas, 2011">
   <nodes>
      <node name="Escultura01">
            //datos del nodo
            <info text="Autor: Rita Longa. Año de creación: 2011"/>
            <obj type="1" />
      </node>
   </nodes>
   <msplines>
      <mspline name="spline1">
      <pto x="0.73394"  y="8.04731e-009" z="0.184101" />
      <pto x="50.73394" y="8.04731e-009" z="50.184101" />
      </mspline>
   </msplines>
</CEV>
```

Donde los nodos representan los objetos de la escena (modelos, luces, etc), son sus datos característicos, dígase posición, rotación, escala, color de la luz si es una luz, y demás. Además de dicha información, se almacena por cada objeto un texto de información, y el tipo de objeto (0 para expositivos, 1 para decorativos y 2 para delimitadores).

La trayectoria de recorrido de la cámara estará definida por una serie de puntos, los cuales se agrupan de acuerdo a sus tres componentes, **x, y, z.**

Consideraciones del capítulo.

Se describió el proceso que actualmente se realiza para la creación de un paseo virtual, para identificar los problemas existentes y precisar los subprocesos que serán objeto de automatización. Se realizó el modelo de dominio para obtener una mejor comprensión de los conceptos y procesos que se manejan dentro del negocio a automatizar. Se presentó la propuesta del sistema a desarrollar como parte de la solución al problema planteado. Se especificaron y describieron los requisitos que la herramienta debe cumplir para obtener su comprensión. Se realizó la definición, representación y descripción textual de los casos de uso del sistema con el objetivo de especificar las funcionalidades del sistema y los actores que interactuarán con el mismo. Se describieron paso a paso la secuencia de eventos que los actores realizaran para inicializar un caso de uso y la respuesta del sistema ante cada acción del actor. Se realizó la matriz de trazabilidad entre los casos de uso y los requisitos para asegurar que se implemente lo que el cliente solicitó. De forma general con la conclusión del presente capítulo se logró describir el sistema que se desea construir lo cual dará paso a una nueva etapa del desarrollo del trabajo para darle cumplimiento a los objetivos trazados.

Capítulo 3. Implementación y Prueba.

Introducción al capítulo 3.

En la fase de Implementación se traduce el diseño de clases, elaborado en la fase de Análisis y Diseño, en componentes físicos, los cuales se convierten en los ficheros correspondientes a la implementación en C++. También se harán las pruebas correspondientes a los Casos de Uso descritos en el flujo de trabajo anterior.

3.1 Lenguajes y herramientas a utilizar.

Para la implementación del módulo se utilizaron las herramientas y lenguajes que más ventajas nos ofrecían para el trabajo con motores gráficos e interfaces visuales, así como para la modelación de la herramienta. Tales fueron:

- Lenguaje de modelado: UML.
- Herramienta de diseño de software: Visual Paradigm para UML.
- Lenguaje de programación: C++.
- Entorno integrado de desarrollo (IDE): QT Creator 4.7.
- Motor Gráfico: Object Oriented Graphics Engine (Ogre). OgreSDK MinGW v1-7-1.

3.2 El modelo de implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describiendo además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros (**Rodríguez Noa** and **Gómez Finalé** 2008).

3.2.1 Diagrama de Componentes.

El diagrama de componentes muestra las relaciones de dependencia entre las partes modulares de un sistema y encapsula la implementación. A continuación se muestra el diagrama de componentes del módulo desarrollado en el presente trabajo.

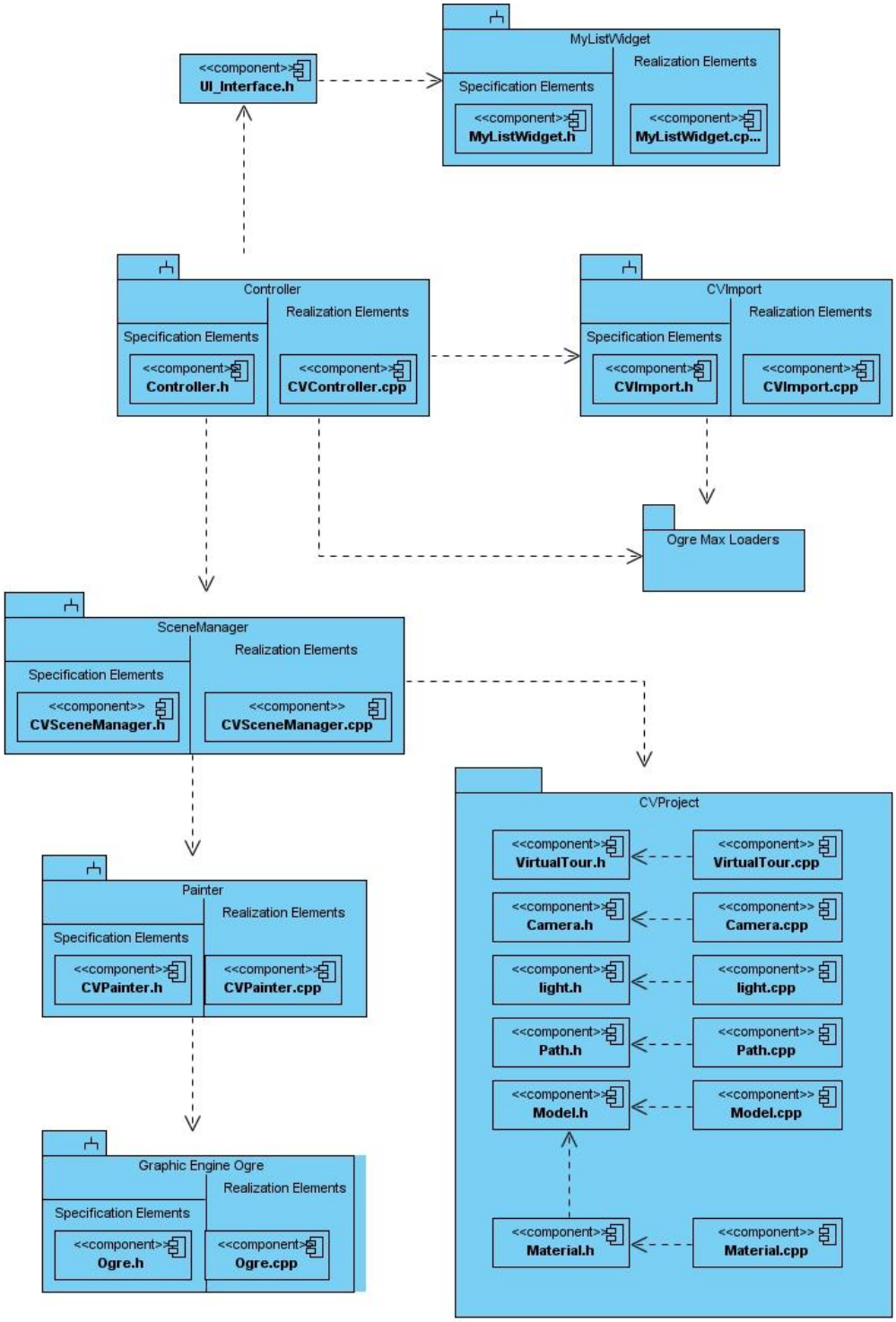

Fig. 16. Diagrama de Componentes.

3.2.2 Diagrama de Despliegue.

El modelo de despliegue es un modelo de objetos que describe la distribución física de un sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. En nuestro caso, al ser desplegado en una misma estación de trabajo, no consideramos necesario mostrar el diagrama.

3.2.3 Implementación del CU Salvar Paseo.

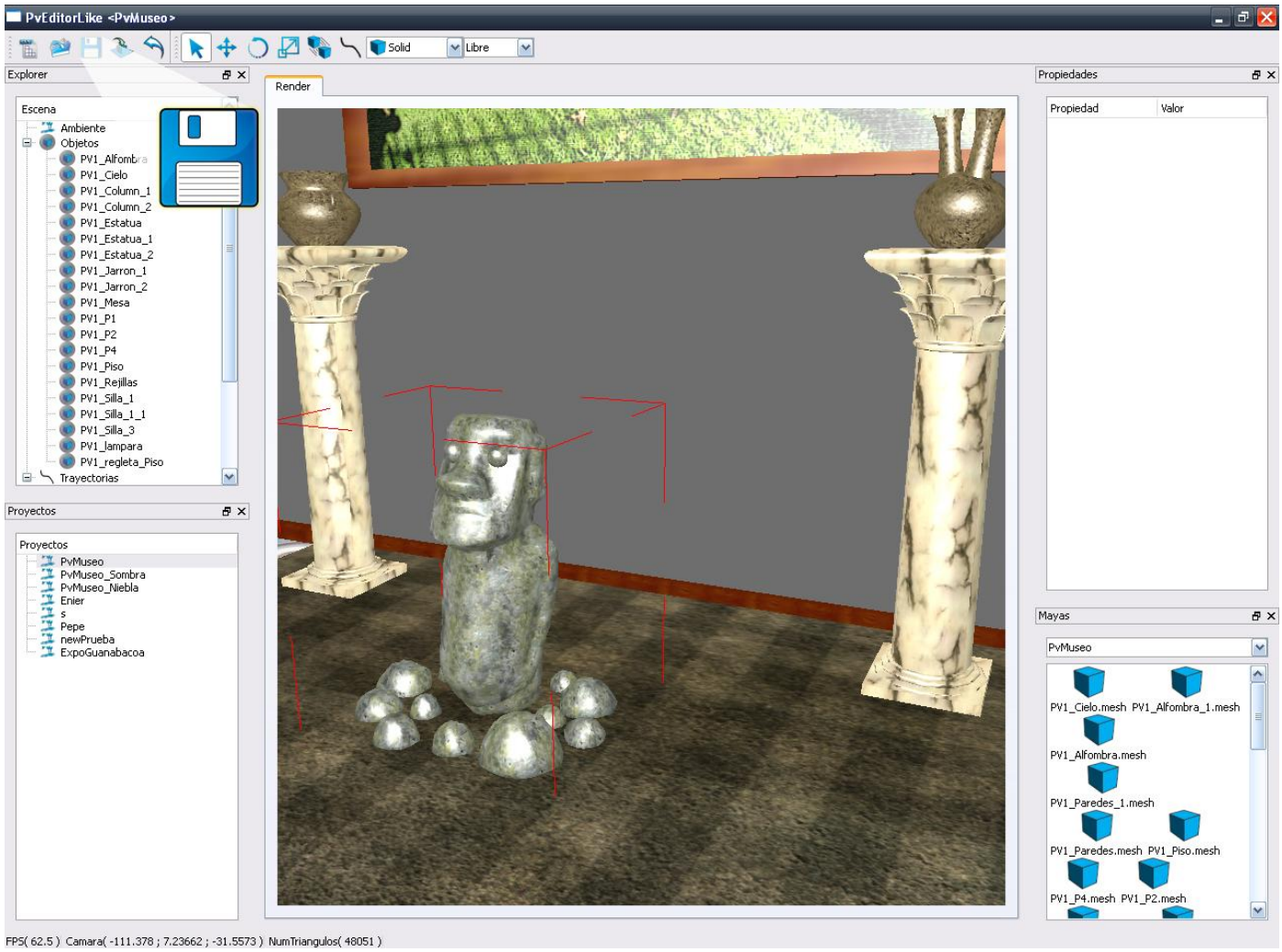

Fig. 17. Implementación del CU Salvar Paseo.

3.2.4 Implementación de los CU Mostrar ayuda e Inicializar recursos.

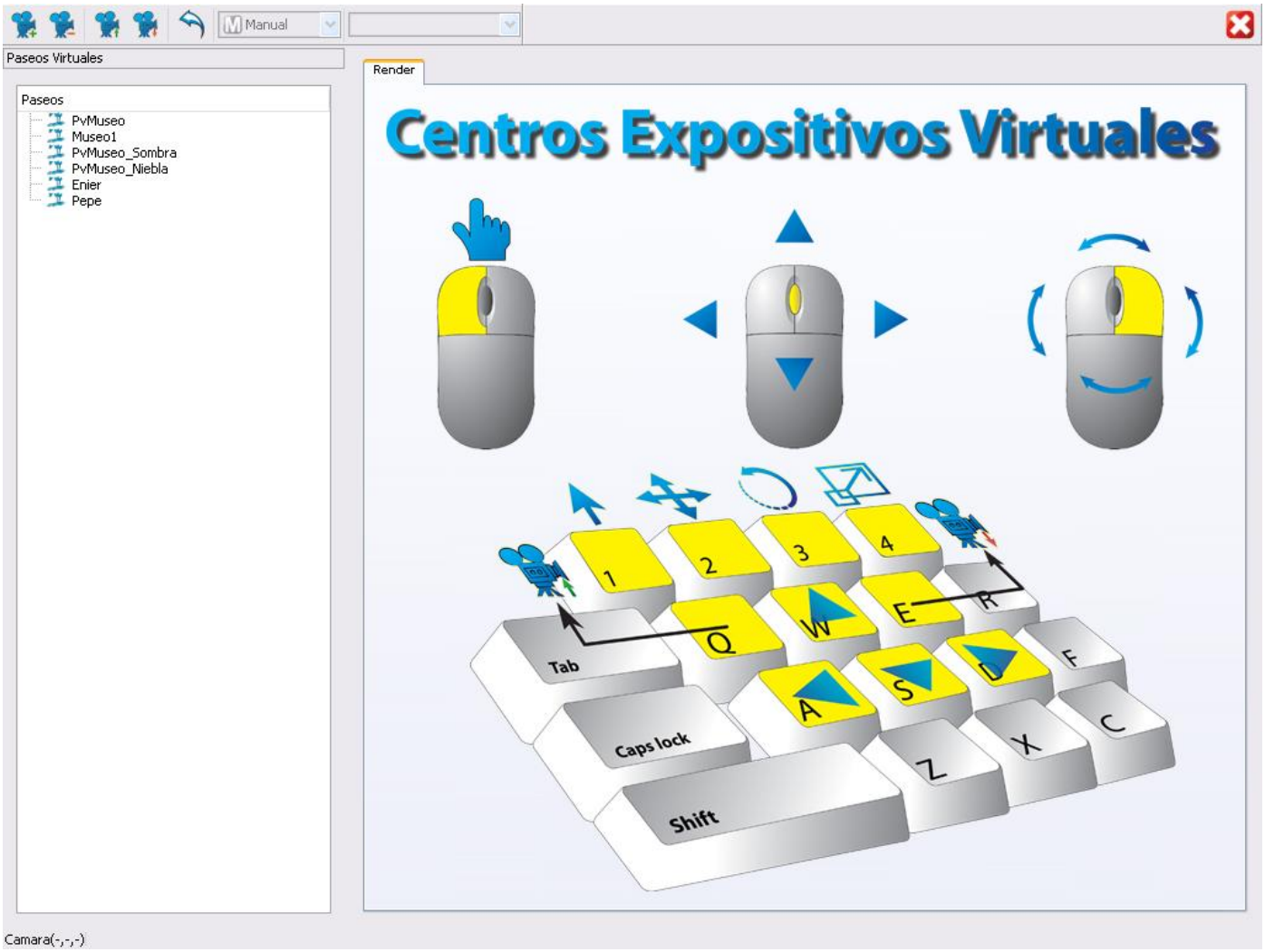

Fig. 18. Implementación de los CU Mostrar ayuda e Inicializar recursos.

3.2.5 Implementación de los CU relacionados con la visualización del Paseo.

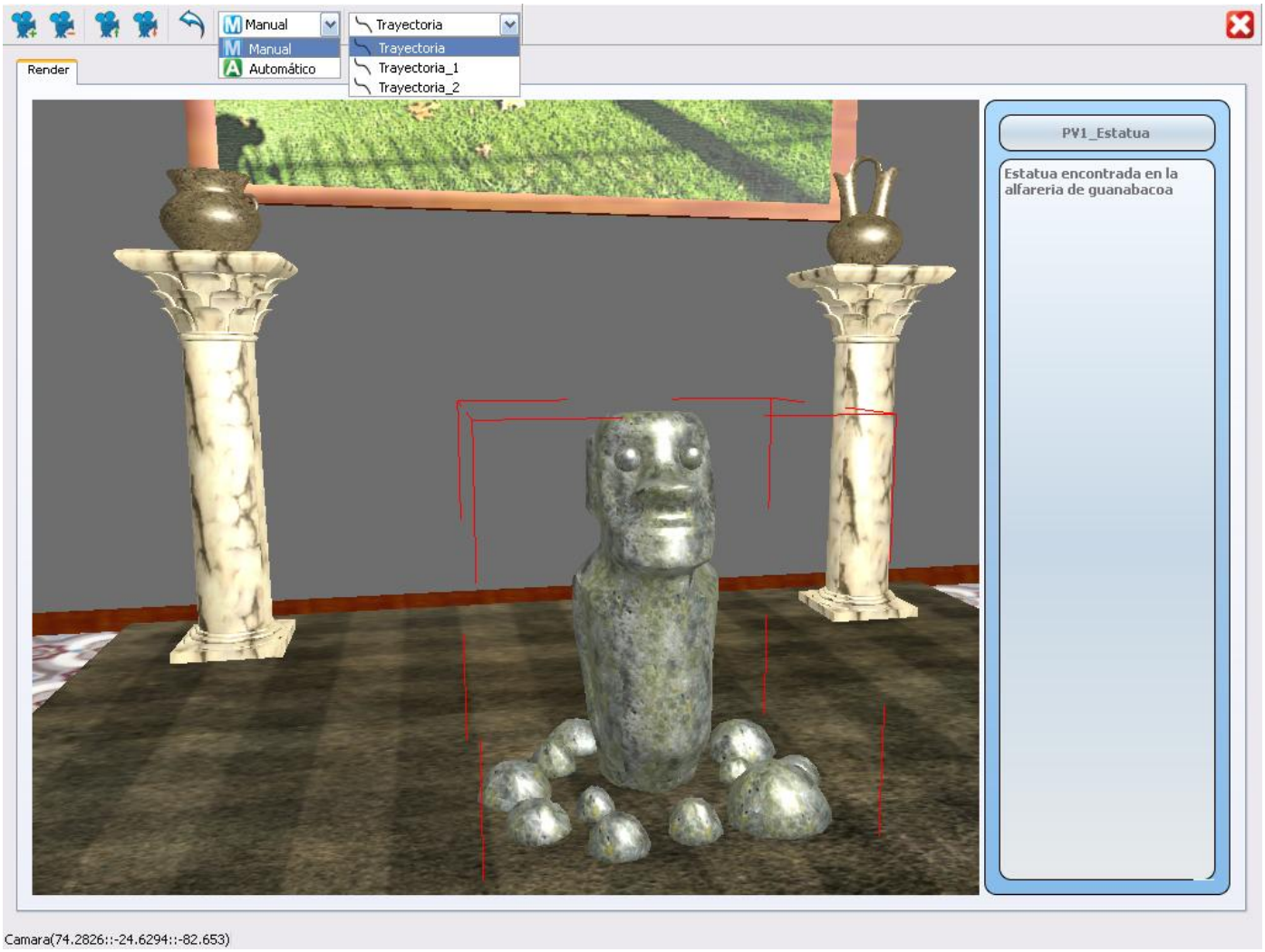

Fig. 19. Implementación de los CU relacionados con la visualización del Paseo.

3.3 Modelo de prueba.

Los modelos de pruebas responden a la implementación de los Casos de Uso más importantes de la aplicación. En este caso, se utilizó para estructurarlos el formato utilizado por los documentos oficiales del Centro de Informática Industrial, al que pertenece el proyecto Paseos Virtuales. El tipo de pruebas realizadas son de caja negra, lo que significa que las pruebas de

caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, y el probador se limita a suministrarle datos como entrada y estudiar la salida.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc). No obstante, pueden ser útiles en cualquier módulo del sistema.

3.3.1 Diseño de caso de prueba. CU Salvar paseo.

Descripción general:

Este Caso de Uso está implementado en el Editor. El diseñador del paseo inicializa el caso de uso cuando da clic en la opción salvar paseo. Se salvan todos los cambios realizados al paseo en la carpeta del proyecto.

Condiciones de ejecución:

Un proyecto debe estar cargado en el área de edición.

Secciones a probar en el caso de uso:

Tabla 5. Diseño de caso de prueba. CU Salvar paseo.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Salvar paseo existente.	**EC 1.1:** Salvar paseo existente.	El usuario selecciona la opción salvar paseo y el sistema guarda los cambios realizados al paseo en la carpeta de proyecto.	Se guardan las modificaciones al proyecto.

3.3.2 Diseño de caso de prueba. CU Inicializar Datos.

Descripción general:

El usuario inicializa el caso de uso cuando ejecuta el Visualizador. El Visualizador carga los paseos previamente creado en el módulo Editor que se encuentren en la carpeta de recursos de la aplicación y los lista para posibilitar la selección del paseo que desee visualizar el usuario.

Condiciones de ejecución:

El paseo debe estar creado con el módulo Editor.

Secciones a probar en el caso de uso:

Tabla 6. Diseño de caso de prueba. CU Inicializar Datos.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Inicializar Datos	**EC 1.1:** Inicializar Datos	Al ejecutarse el sistema, este carga los proyectos que actualmente tiene en la carpeta de proyectos y los muestra, permitiendo que el usuario abra cada uno de ellos. Se muestran los proyectos actuales.	Ejecutar producto

3.3.3 Diseño de caso de prueba. CU Definir modo de paseo.

Descripción general:

El caso de uso se inicializa cuando el usuario define la forma en que se realizará el recorrido por el centro expositivo virtual, el recorrido puede realizarse de forma libre (utilizando los controles del teclado) o dirigido (se muestra el centro virtual a través del recorrido de la cámara).

Condiciones de ejecución:

El paseo debe estar cargado en el área de visualización.

Secciones a probar en el caso de uso:

Tabla 7. Diseño de caso de prueba. CU Definir modo de paseo

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Definir modo de paseo.	**EC 1.1:** Definir modo de paseo.	Al abrir un proyecto, se le da la posibilidad al usuario de escoger el modo de paseo.	Seleccionar el modo de paseo.

3.3.4 Diseño de caso de prueba. CU Editar cámara.

Descripción general

El caso de uso se inicializa cuando el usuario modificar la velocidad o la altura de la cámara.

Condiciones de ejecución:

El paseo debe estar cargado en el área de visualización.

Secciones a probar en el caso de uso:

Tabla 8. Diseño de caso de prueba. CU Editar cámara.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Editar cámara.	**EC 1.1:** Editar cámara.	El usuario tendrá la posibilidad de escoger la altura a la que quiere la cámara, así como la velocidad a la que esta se moverá.	Modificar altura de la cámara. Modificar velocidad de la cámara.

3.3.5 Diseño de caso de prueba. CU Mostrar información asociada a un objeto.

Descripción general:

El caso de uso se inicializa cuando el usuario selecciona un objeto al hacer clic encima del mismo y se muestra la información asociada al objeto en una pequeña ventana.

Condiciones de ejecución:

El paseo debe estar cargado en el área de visualización.

Secciones a probar en el caso de uso:

Tabla 9. Diseño de caso de prueba. CU Mostrar información asociada a un objeto.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Mostrar información asociada a un objeto.	**EC 1.1:** Mostrar información asociada a un objeto.	Al hacer clic cobre un objeto que posea información, ésta se mostrará sobre la ventana de visualización.	Hacer clic sobre un objeto. Mostrar información.

3.3.6 Diseño de caso de prueba. CU Resaltar objetos expositivos.

Descripción general:

El caso de uso se inicializa cuando el usuario pasa el mouse por encima de un objeto, el puntero cambia de forma indicando que el objeto puede ser seleccionado para ver su información asociada.

Condiciones de ejecución:

El paseo debe estar cargado en el área de visualización.

Secciones a probar en el caso de uso:

Tabla 10. Diseño de caso de prueba. CU Resaltar objetos expositivos.

Nombre de la	Escenarios de la	Descripción de la funcionalidad	Flujo central

sección	sección		
SC 1: Resaltar objetos expositivos.	**EC 1.1:** Resaltar objetos expositivos.	Al pasar el mouse por encima de un objeto que tenga información, el puntero del mouse cambiará.	Posicionar el mouse sobre un objeto. Cambia el puntero del mouse.

3.3.7 Diseño de caso de prueba. CU Comenzar paseo.

Descripción general:

El caso de uso se inicializa cuando el usuario selecciona de la lista de paseos disponibles el que desea visualizar.

Condiciones de ejecución:

El paseo debe estar disponible en la lista de paseos del visualizador.

Secciones a probar en el caso de uso:

Tabla 11. Diseño de caso de prueba. CU Comenzar paseo.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Comenzar paseo.	**EC 1.1:** Comenzar paseo.	Al seleccionar el proyecto que desea abrir, la ventana de visualización del paseo va a pantalla completa, y se ocultan las demás.	Seleccionar proyecto a abrir. Maximizar la ventana de visualización. Ocultar las demás ventanas.

3.3.8 Diseño de caso de prueba. CU Cerrar paseo.

Descripción general:

El caso de uso se inicializa cuando el usuario selecciona la opción cerrar paseo.

Condiciones de ejecución:

El paseo debe estar cargado en el área de visualización.

Secciones a probar en el caso de uso:

Tabla 12. Diseño de caso de prueba. CU Cerrar paseo.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Cerrar paseo.	**EC 1.1:** Cerrar paseo.	Al seleccionar la opción cerrar paseo, se cierra la ventana de visualización y vuelve al estado inicial del visualizador, mostrando los proyectos que están listos para ser visualizados.	Seleccionar opción cerrar. Cerrar paseo. Volver visualizador al estado inicial.

3.3.9 Diseño de caso de prueba. CU Mostrar ayuda del paseo.

Descripción general:

El caso de uso se inicializa cuando el usuario ejecuta la aplicación. En la pantalla principal se muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual.

Condiciones de ejecución:

El paseo debe estar cargado en el área de visualización.

Secciones a probar en el caso de uso:

Tabla 13. Diseño de caso de prueba. CU Mostrar ayuda del paseo.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Mostrar ayuda del paseo.	**EC 1.1:** Mostrar ayuda del paseo.	Una vez que el usuario ejecuta la aplicación, en la pantalla principal se muestra una imagen que lo instruye sobre cómo realizar el recorrido.	Se muestra la ayuda del paseo.

Consideraciones del capítulo.

En este capítulo se describió cómo los elementos entregados como resultado de la fase de Análisis y Diseño son implementados. Se describió además cómo se organizan los componentes de acuerdo con los mecanismos disponibles en el entorno de implementación y en el lenguaje de programación utilizado y cómo dependen los componentes unos de otros.

Se muestran también los casos de pruebas diseñados para realizar el flujo de Pruebas, asegurando que las deficiencias puedan ser solucionadas rápidamente por el desarrollador.

Conclusiones.

Terminado el trabajo se puede concluir que se logró desarrollar un módulo de persistencia y visualización para una herramienta de creación de Centros Expositivos Virtuales, se resumió el estado del arte relacionado con las técnicas de programación, se caracterizaron e identificaron las posibles metodologías y lenguajes a utilizar, los formatos de escenas 3D y los Paseos Virtuales, se resumieron los resultados del Análisis y Diseño recibidos para el desarrollo del módulo, se diseñó y graficó el Diagrama de Componentes, se implementaron los Casos de Usos del módulo, y se diseñaron, describieron y aplicaron las pruebas de Caja Negra.

Todos estos resultados obtenidos demuestran que el objetivo de la investigación se alcanzó satisfactoriamente ya que de manera general se obtuvo un módulo de persistencia y visualización para una herramienta de creación de Centros Expositivos Virtuales; que permite salvar los cambios efectuados en los Paseos Virtuales y visualizar estos Paseos posteriormente mediante una aplicación.

Recomendaciones.

Recomendamos como parte del soporte del módulo:

- Implementar un módulo de detección de colisiones para la aplicación de visualización.
- Investigar nuevos mecanismos de visualización de la información de los objetos del entorno.
- Investigar nuevos mecanismos de detección de recursos.

Referencias Bibliográficas.

Autodesk.com (2007). Autodesk 3ds Max 8 SDK Help.

Autodesk.com. (2010). "Specifications of .DXF file." from http://www.autodesk.com/techpubs/autocad/dxf/.

Autodesk.com. (2011, Ene 2011). "Autodesk 3ds Max Features." from http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13567426.

Autodesk.com. (2011, Ene 2011). "Autodesk Maya Features." from http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13583239.

Blender.org. (2010). "Architecture of .BLEND file." from http://www.blender.org/development/architecture/.

Blender.org. (2011, Jan 2011). "Blender Features." from http://www.blender.org/features-gallery/features/.

Department of Computer Sciences, U. o. M. (2003). "Binary vs. ASCII." from http://www.cs.umd.edu/class/spring2003/cmsc311/Notes/BitOp/asciiBin.html.

Eclipse.org. (2011). "Eclipse IDE for C/C++ Developers." from http://www.eclipse.org/downloads/moreinfo/c.php.

Echemendía González, A. and **W. García López** (2007). Sistema de Generación de Ficheros para Entornos Virtuales., Universidad de las Ciencias Informáticas.

Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo, E. U. I. T. I. O. (2009). Entornos de Desarrollo Integrado.

FileInfo.com. (2010). "What is the difference between binary and text files." from http://www.fileinfo.com/help/binary-vs-text-files.html.

Microsoft. "Introducción a Visual Studio." from http://msdn.microsoft.com/es-es/library/fx6bk1f4(v=VS.80).aspx.

Nokia. (2011). "Qt framework, Qt Creator IDE and tools." from http://qt.nokia.com/products/developer-tools/.

Ogre3D.org. (2009, Dic 22, 2010). "The Core Objects." from http://www.ogre3d.org/docs/manual/manual_4.html#SEC4.

Ogre3D.org. (2009). "Ogre3D Homepage." from http://www.ogre3d.org/.

Quintana López, A. and **Y. Alonso Monteagudo** (2010). Motor de Render Genérico, Universidad de las Ciencias Informáticas.

RedIRIS.com. (2009). "Modelado UML." from https://forja.rediris.es/docman/view.php/282/444/uml20.pdf.

Rodríguez Noa, C. A. and **Y. Gómez Finalé** (2008). Arquitectura de red para la confección de videojuegos multijugadores.

SourceForge.net. (2010). "G3D Project Home ", from http://g3d.sourceforge.net/.

UnrealWiki (2006) "ASE File Format."

Anexos

Anexo 1. Diagrama de casos de uso.

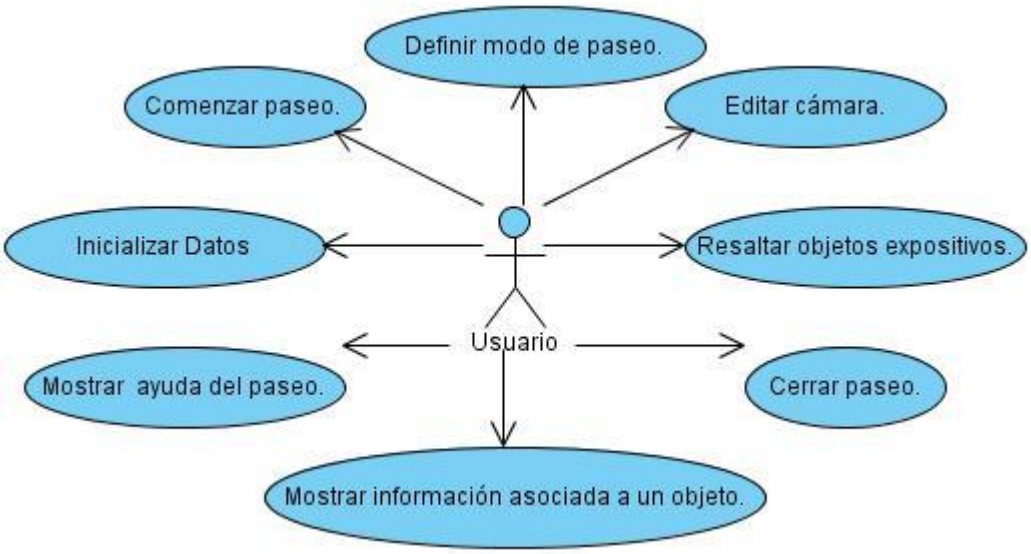

Fig. 20. Diagrama de casos de uso del módulo Visualizador.

Anexo 2. Descripción textual de los casos de uso.

Casos de uso expandidos para el módulo Visualizador.

Caso de uso	
CU-2	Definir modo de paseo.
Propósito	El objetivo del caso de uso es que el usuario realice el paseo por el centro expositivo virtual del modo seleccionado.
Prioridad	Secundario.
Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario define la forma en que se realizará el recorrido por el centro expositivo virtual, el recorrido puede realizarse de forma libre (utilizando los controles del teclado) o dirigido (se muestra el centro virtual a través del recorrido de la cámara).	

Referencias	RF 2.
Precondiciones	El paseo debe estar cargado en el área de visualización.
Poscondiciones	Se define el modo en que se realizará el recorrido por el centro expositivo virtual.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el tipo de recorrido a realizar.	1.1 El sistema muestra el centro expositivo virtual con el tipo de recorrido seleccionado.

Caso de uso	
CU-3	Editar cámara.
Propósito	El objetivo del caso de uso es que el usuario pueda modificar la velocidad y la altura de la cámara.
Prioridad	Secundario.
Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario modificar la velocidad o la altura de la cámara.	
Referencias	RF 3, RF 3.1, RF 3.2, RF 3.3.
Precondiciones	El paseo debe estar cargado en el área de visualización.
Poscondiciones	Aumenta o disminuye la velocidad de la cámara o la altura.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario aumenta o disminuye la velocidad o la altura de la cámara.	1.1 El sistema aumenta o disminuye la velocidad o la altura de la cámara.

Caso de uso	
CU-4	Mostrar información asociada a un objeto.
Propósito	El objetivo del caso de uso es que el usuario pueda observar la información asociada a un objeto.
Prioridad	Secundario.

Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario selecciona un objeto al hacer clic encima del mismo y se muestra la información asociada al objeto en una pequeña ventana.	
Referencias	RF 4.
Precondiciones	El paseo debe estar cargado en el área de visualización.
Poscondiciones	Se muestra la información asociada a un objeto seleccionado.
Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un objeto.	1.1 El sistema muestra la información asociada al objeto en una pequeña ventana.

Caso de uso	
CU-5	Resaltar objetos expositivos.
Propósito	El objetivo del caso de uso es resaltar los objetos expositivos que se encuentran en el centro expositivo virtual.
Prioridad	Secundario.
Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario pasa el mouse por encima de un objeto, el puntero cambia de forma indicando que el objeto puede ser seleccionado para ver su información asociada.	
Referencias	RF 5.
Precondiciones	El paseo debe estar cargado en el área de visualización.
Poscondiciones	Resalta el objeto al pasar el mouse por encima del mismo.
Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario mueve el mouse por encima de uno de los objetos expositivos.	1.1 El sistema cambia el cursor señalando que el objeto puede ser seleccionado. Finaliza el caso de uso.

Caso de uso	
CU-8	Mostrar ayuda del paseo.
Propósito	El objetivo del caso de uso es mostrar una ayuda al usuario con información sobre cómo realizar el recorrido por el entorno virtual.
Prioridad	Secundario.
Actores: Usuario.	
Resumen: El caso de uso se inicializa cuando el usuario selecciona la opción mostrar ayuda. Luego de haber seleccionado la opción se muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual.	
Referencias	RF 8.
Precondiciones	El paseo debe estar cargado en el área de visualización.
Poscondiciones	Se muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario elige la opción mostrar ayuda.	1.1 El sistema muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual. Finaliza el caso de uso.