Universidad de las Ciencias Informáticas
Facultad 5

Herramienta de Configuración para Generadores de Entornos 3D basados en Capas.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores: José Dariel León Fleitas.

Alexis Roger Pedrón Borjas.

Tutor: Ing. Yenifer del Valle Guevara.

Co-tutor: Ing. Omar Correa Madrigal.

Ciudad de La Habana, junio 2010.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y autorizamos al Proyecto Juegos-CNeuro de la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

José Dariel León Fleitas

_____
Firma del Autor

Alexis Roger Pedrón Borjas

_____
Firma del Autor

Yenifer del Valle Guevara

_____
Firma del Tutor

DATOS DE CONTACTO

Tutor

Nombre y Apellidos: Yenifer del Valle Guevara
Edad: 28 años
Ciudadanía: Cubana
Institución: Universidad de las Ciencias Informáticas
Título: Ingeniero en Ciencias Informáticas
Categoría Docente: Profesor Adiestrado
E-mail: ydelvalle@uci.cu
Graduado en la Universidad de las Ciencias Informáticas, 3 años de experiencia en proyectos de Realidad Virtual.

Co-tutor

Nombre y Apellidos: Omar Correa Madrigal
Edad: 28 años
Ciudadanía: Cubana
Institución: Universidad de las Ciencias Informáticas
Título: Ingeniero en Ciencias Informáticas
Categoría Docente: Profesor Adiestrado
E-mail: ocorrea@uci.cu
Graduado en la Universidad de las Ciencias Informáticas, 3 años de experiencia en proyectos de Realidad Virtual.

AGRADECIMIENTOS

Alexis:

A mis padres Silvia y Fernando, por brindarme siempre su apoyo en todo momento, por sacrificarse tanto y darme tanto amor, por ser ambos mis guías y mis ejemplos a seguir, realmente nunca podré agradecerles todo lo que han hecho por mí, este logro es más suyo que mío.

A mis abuelos, en especial a mi abuelo Roger, que donde quiera que esté espero que se sienta orgulloso de mí.

A mi tía Sara por su cariño y mi primo Alejandro que es mi hermano pequeño.

A mi familia de la Habana, tía María, Mimi y Pedro Alejandro que desde que llegué me acogieron como uno más de ellos.

A mi novia Rosana, por su amor, comprensión y apoyo en todo momento.

A mi compañero de tesis Jose por ser un buen amigo y soportarme todo este tiempo.

A todos mis amigos, los viejos y los nuevos, los del grupo y los del proyecto, de todos aprendí algo.

Jose:

A mi madre por su esfuerzo sobrenatural y apoyo sin el cual no se pudiera haber logrado todo lo aprendido, sin lugar a dudas ha sido mi guía fundamental y mi inspiración.

A mis hermanos y mi prima por su ayudita en la ubicación…Tú sabes!!!

A la gente de "La Gasolinera", mi familia, aunque no exista sangre de por medio, que junto a mi madre me han seguido paso a paso.

A Rosana por su HASEE, sin la cual no hubiese sido posible la realización de este trabajo.

A Andileidys por toda su comprensión y por esa niña tan linda que me ha dado.

A mi suegra, por haber sido hasta ahora el "padre" de mi niña y por "no" quererme tanto.

A Celeste y Nildo por haberme acogido en su casa como un hijo más.

A todos mis amigos, que en los momentos difíciles me ayudaron a seguir adelante.

A Alexis, que bueno, el sabe, jejejeje.

A todas aquellas personas que de una forma u otra formaron parte de este trabajo y sin los cuales no hubiera sido posible realizarlo, a nuestros compañeros del grupo y del proyecto que nos brindaron sus consejos y experiencias, en especial a Julio Oscar y Argelio por brindarnos su ayuda en todo momento.

Gracias a todos.

DEDICATORIA

…a la memoria de mi abuelo Roger, quien más que un abuelo fue un padre para mí.

…a mi madre y mi padre, que han sido faro y guía de cada uno de mis pasos.

…a toda mi familia y amigos.

Alexis.

… a la memoria de mi padre, quien hubiese querido estuviera presente para ver lo que he logrado.

…a mi madre y mi niña que me han motivado a concluir mí camino.

…a toda aquella persona que de una forma u otra me ha ayudado a realizar este trabajo.

Jose.

RESUMEN

En la actualidad, los Sistemas de Realidad Virtual (SRV) son utilizados en disímiles áreas que van desde la medicina, hasta la industria del entretenimiento como es el caso de los videojuegos. La curiosidad de un jugador eleva la demanda de crear entornos más extensos en los videojuegos, para esto los desarrolladores utilizan varias técnicas de generación de entornos virtuales, una de ellas es la técnica de generación por capas, esta se basa en un sistema de n capas que se encargan de generar los diferentes contenidos del entorno, estas capas son controladas por un generador de entornos que utiliza un fichero que contiene la configuración de estas. Este trabajo propone una aplicación que permita realizar la configuración de las capas y generar el fichero de configuración.

En el desarrollo de la investigación se ha realizado un estudio de diferentes temas como: las técnicas de generación de entornos y contenidos existentes, bibliotecas para el desarrollo de interfaces gráficas de usuarios; así como, un análisis del formato XML para el almacenamiento de datos. A partir de esta investigación se proponen las características técnicas de la solución y se determina RUP como metodología de desarrollo y C++ como lenguaje de programación.

Como resultado se obtuvo una herramienta que permite realizar el proceso de configuración de las capas así como generar el fichero de configuración de estas y que garantiza además, el incremento de la productividad de los desarrolladores del proyecto Juegos-CNeuro de la Facultad 5 de la Universidad en las Ciencias Informáticas.

PALABRAS CLAVES

Videojuegos, entornos virtuales, capas, contenidos, generador de entornos, fichero de configuración.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En el mundo actual la tecnología ha progresado más rápido que nuestra habilidad para imaginar que vamos a hacer con ella. Las computadoras se han convertido en una herramienta de suma importancia, no sólo para el desarrollo de los pueblos, sino también, para el desarrollo de la ciencia y nuevas tecnologías debido a los crecientes avances que en materia de informática se han alcanzado, los que han permitido el surgimiento y desarrollo de nuevos campos entre los que se destaca la Realidad Virtual (RV).

La realidad cotidiana está llena de entornos virtuales: el cine, la televisión, la lectura, son ambientes o actividades que disminuyen la sensación de presencia en el entorno real e incrementan la de encontrarse en otra parte. Sin embargo la RV informática va más allá de la sensación de inmersión en otro mundo ya que nos permite interactuar con el entorno virtual de diferentes maneras.

La RV es simulación por computadora, dinámica y tridimensional, con alto contenido gráfico, acústico y táctil, orientada a la visualización de situaciones y variables complejas, durante la cual el usuario ingresa, a través del uso de sofisticados dispositivos de entrada, a “mundos” que aparentan ser reales, resultando inmerso en ambientes altamente participativos, de origen artificial. [1]

En la actualidad, la RV se aplica en múltiples sistemas y en diferentes ramas. Tal es el caso de los simuladores de vuelos utilizados por el ejército y que son hoy en día una herramienta fundamental para el entrenamiento de los pilotos. Los cirujanos pueden realizar operaciones simuladas para ensayar las técnicas más complicadas, antes de una operación real. Los astronautas tienen la posibilidad de volar sobre la superficie simulada de un planeta desconocido y experimentar la sensación que tendrían si estuvieran allí. Los arquitectos utilizan diferentes herramientas que les permiten crear modelos virtuales de futuros edificios, permitiéndoles experimentar en la etapa de diseño con distintos modelos para luego decidirse por el más correcto. El campo más habitual y conocido de la RV es el del ocio y entretenimiento ya que podemos disfrutar de potentes Videojuegos de diversos tipos en nuestras casas, este constituye, sin duda alguna, uno de los mercados más prominentes en cuanto a rentabilidad y grandes ganancias.

En nuestro país debido al auge que ha alcanzado la informática gracias al esfuerzo de la Revolución poco a poco ha ido creciendo el desarrollo de Sistemas de Realidad Virtual (SVR). La Universidad de las

Ciencias Informáticas (UCI) se ha vinculado activamente en este empeño y para ello cuenta con la Facultad 5 en la que se desarrollan proyectos vinculados al campo de los SRV destacándose los proyectos de desarrollo de Videojuegos como es el caso del Proyecto Juegos-CNeuro.

En el Proyecto Juegos-CNeuro se está desarrollando un videojuego que tiene como nombre Meteorix, el objetivo del mismo consiste en contribuir al desarrollo de habilidades visuales en niños con discapacidades visuales. Para llevar a cabo la realización del juego se utilizan herramientas y bibliotecas que se están desarrollando en el proyecto como es el caso del Generador de Entornos 3D basado en Capas, el cual es una biblioteca que se utiliza para la creación dinámica de ambientes virtuales.

Dicho generador, para su correcto funcionamiento, necesita de un fichero de configuración que se ajuste a sus necesidades, crear este fichero de forma manual resulta engorroso por las siguientes razones:

- Se debe realizar una serie considerable de cálculos para definir la posición de las capas a utilizar por el generador.
- El generador puede usar de una a n capas por lo que sería muy trabajoso realizar la configuración capa por capa.
- Significaría un gasto de tiempo considerable.

Esta situación precisamente es la que da lugar a un ***problema científico*** que se manifiesta a través de la siguiente interrogante: ¿Cómo lograr la generación correcta y eficiente del fichero de configuración para el Generador de Entornos 3D basados en Capas del proyecto Juegos-CNeuro?

El **objetivo general** que se propone este trabajo es elaborar una herramienta que permita generar el fichero de configuración para el Generador de Entornos 3D basados en Capas del proyecto Juegos-CNeuro, teniendo como **objeto de estudio** el análisis de los Sistemas de Configuración de Entornos Virtuales derivándose como **campo de acción** el estudio de los procesos de generación de ficheros de configuración para Entornos Virtuales 3D.

A continuación se mencionan un conjunto de acciones concretas que permitirán satisfacer el objetivo planteado:

- ✓ Estudio de las características de los ficheros de configuración de Entornos 3D para poder definir las características y elementos del fichero a crear.
- ✓ Estudio de los diferentes Generadores de Ficheros existentes para poder definir las funcionalidades y elementos principales que los conforman.
- ✓ Estudio de las características de los ficheros XML para definir un formato de fichero propio.
- ✓ Definir la solución a partir del estudio realizado.
- ✓ Diseño e implementación de la herramienta de configuración.

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

Analítico-Sintético: El uso del mismo nos permitirá realizar un estudio por separado de cada una de las Herramientas Generadoras de Ficheros de Configuración de Entornos 3D para así determinar cuáles son los elementos fundamentales que las caracterizan, así como de los diferentes formatos de los Ficheros de Configuración de Entornos 3D existentes en la actualidad, obteniendo así una gran cantidad de información que junto con la bibliografía nos permitirá un mejor entendimiento del problema.

Histórico-Lógico: Este método está vinculado al conocimiento de las distintas etapas de los objetos en su sucesión cronológica, por tanto, nos permitirá tener una visión de cómo han ido evolucionando las diferentes Herramientas Generadoras de Ficheros en materia de arquitectura y diseño y cuál es su tendencia actual.

Para una mejor comprensión, el documento está dividido en 4 capítulos de la siguiente forma:
Cap. 1 Fundamentación Teórica.
Cap. 2 Soluciones Técnicas.
Cap. 3 Características del Sistema.
Cap. 4 Diseño e Implementación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se analiza en detalle el proceso de Generación de Entornos Virtuales, abordando las diferentes técnicas y algoritmos de generación de contenidos y entornos virtuales, haciendo énfasis en la técnica de Generación de Entornos Virtuales basados en Capas. También se realiza un estudio de las interfaces gráficas de usuario y de las principales bibliotecas de desarrollo de interfaces gráficas de usuario, con el objetivo de determinar la que más se adecue a las necesidades de desarrollo.

Además, se realiza un estudio de los principales motores gráficos libres existentes en la actualidad, resaltándose las potencialidades y características de cada uno, para así poder determinar el que mejor cumpla con los requisitos necesarios para el desarrollo del sistema. Por último, se realiza un estudio de los ficheros XML, analizando sus características y las ventajas que trae el empleo de este como formato para almacenar información.

1.1 Generación de Entornos Virtuales

El avance alcanzado por el hardware y la necesidad de lograr visualizaciones más realistas han permitido obtener a través del gráfico por computadoras resultados realmente impresionantes. Aunque en la actualidad la capacidad de procesamiento de las computadoras permite niveles aceptables de visualización todavía la complejidad de los sistemas sigue colocándolos en crisis, por lo que se mantiene latente la necesidad de construir los Entornos Virtuales de manera eficiente.

La curiosidad de un jugador o de un usuario de cualquier sistema de Realidad Virtual, eleva constantemente la demanda de crear entornos más extensos y con menos restricciones de movimiento. Los desarrolladores a pesar del avance de las tecnologías de diseño, presentan siempre problemas con el tiempo y el esfuerzo para elaborarlos. Por tal motivo la Generación Automática de Entornos ha devenido siempre como campo de apoyo para reducir tales problemas, permitiendo así, grandes niveles de desarrollo.

La Generación de Entornos Virtuales es una técnica que haciendo uso de la computación automatiza la construcción de entornos virtuales apoyada en los gráficos por computadoras. Su aplicación se evidencia

en los Sistemas Geoespaciales 3D, en Videojuegos, en uso militar como simuladores de juegos así como en muchas otras aplicaciones informáticas en la actualidad. [2]

Una de las potencialidades que brinda la generación es la creación de entornos altamente variables tanto de forma como de contenido, lo cual brinda la posibilidad de seleccionar los mejores para aquellos juegos con contenidos predefinidos y crear también juegos versátiles con contenidos variables. Además, brinda la posibilidad de crear entornos infinitos [3] en tiempo real con una gran variedad de contenidos válidos para videojuegos y simuladores.

1.1.1 Generación en Tiempo Real

La Visualización en Tiempo Real (Real Time Rendering) es descrito por Akenine Moeller y Eric Haines en [4] como un proceso en donde el usuario reacciona y actúa ante un coherente cambio de imágenes. El ciclo de interacción entre el usuario y la imagen dibujada es medido en cuadros por segundos (fps), y estos para que sean interpretados por el usuario en tiempo real deben oscilar con una frecuencia por encima de los 30 fps.

La generación de entornos en tiempo real implica cumplir con este principio al ejecutar su proceso, por lo tanto, toda técnica o algoritmo tiene muy en cuenta el factor de eficiencia, asociado en muchos casos a la reutilización de contenidos (optimización de los recursos de memoria) y a la complejidad operacional (optimización de los recursos de procesamiento) de estos. A continuación se explicarán cuales son las técnicas y algoritmos empleados para la generación de entornos en tiempo real.

1.1.2 Técnicas y Algoritmos para la Generación de Contenidos

Los contenidos u objetos que conforman un entorno virtual ocupan un espacio importante en el proceso de generación de entornos virtuales. En varios sistemas los contenidos son construidos en pre-procesamiento por diseñadores de manera manual para luego ser cargados en la aplicación en tiempo real y colocados según una estrategia de generación seleccionada, en otros, los contenidos son construidos de forma procedural en tiempo de ejecución. Ambas formas de generación se ajustan a aplicaciones que recrean entornos reales como sistemas Geoespaciales 3D. Existen varias técnicas y algoritmos para la generación de contenidos, entre los que se encuentran:

- Ruidos (*Noise*): Por un *noise* se entiende una función irregular primitiva usada para romper la monotonía de patrones. Estos han sido aplicados en la generación de texturas y geometrías como paisajes montañosos, nubes y materiales como la madera y mármol.

- Fractal: Un fractal está definido como una forma similar a sí misma a diferentes escalas. Este puede ser creado por sucesivas réplicas de la misma forma siguiendo diferentes reglas. Se utilizan en la generación de árboles y terrenos.

- Mapa de Alturas: Es muy utilizada esencialmente en la creación de terrenos. El mapa es un arreglo bidimensional de valores de altura organizados en una rejilla regular. En cada par (x,y) de la rejilla se almacena el valor de z que corresponde a la altura y coincide con la coloración en escalas de grises del pixel (x,y) de la imagen que se tome como mapa, el cual se acota en el intervalo entre 0, que sería la altura más baja del terreno (color negro) y 255 la más alta (color blanco). Ver Figura 1.

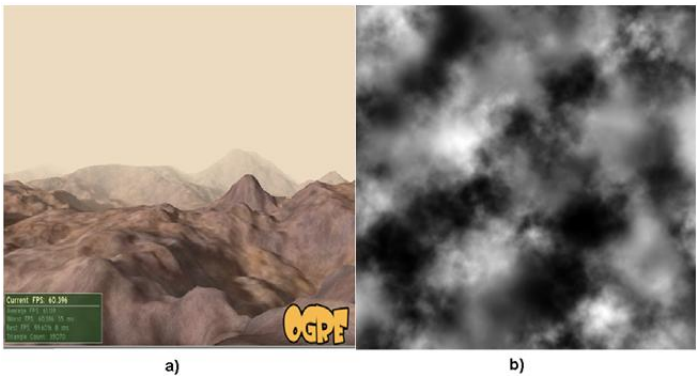


Figura 1: Terreno OGRE(a) generado a partir del mapa de altura (b).

- Gramáticas: Esta técnica hace uso de principios para la especificación de lenguajes, en donde sus componentes pueden ser cualquier tipo de elementos, dígase letras, palabras o geometrías y texturas cuando se habla de generación de contenidos.

1.1.3 Estrategias de Generación de Entornos Virtuales

Las estrategias de generación de entornos virtuales se basan esencialmente en la forma en que los contenidos serán dispuestos en el entorno. Otro elemento esencial en estas estrategias, es la importancia que se le presta a la reutilización de los contenidos en sí, o de las partes (geometrías base y texturas) que conforman a estos, todo con el fin de lograr un nivel de eficiencia alto [3].

1.1.3.1 Estrategia de Frustum Filling

Frustum Filling (Rellenado del Volumen de Visión) es una de las estrategias más acertadas en cuanto a eficiencia, su principio es rellenar constantemente el Frustum con contenidos, los que son eliminados cuando producto del desplazamiento o por cambios de la orientación salen de este. Fue propuesta por S.Greuter y N.Steward y haciendo uso de ella se pueden generar entornos espaciales, ciudades y paisajes de forma infinita.

1.1.3.2 Estrategia Page Manager

Esta estrategia *Page Manager* (Gestor de Página) se basa en las operaciones de rotación de matrices y permite que un gestor trabaje sobre los diferentes bloques de contenidos asociados a una página, ya sea agregándolos o eliminándolos según el grado de avance del usuario. La figura 2 muestra como procede la estrategia a medida que la cámara avanza de un bloque a otro. Cada desplazamiento entre bloques identifica un cambio de página. Se asume que cada bloque es lo suficientemente grande para contener a la cámara lo cual puede traer problemas si se quiere aplicar la estrategia en bloques de menor tamaño.

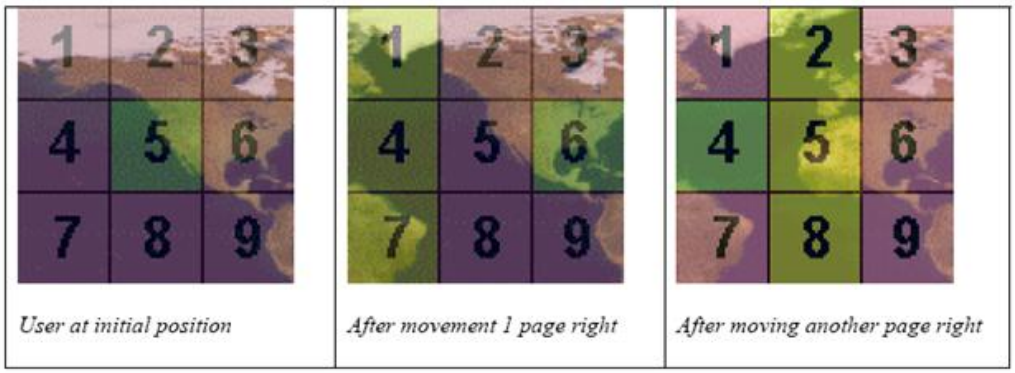


Figura 2: Progreso de la estrategia en dos avances de página a la derecha. El color verde representa el bloque en donde está la cámara y los amarillos los bloques activos, hacia donde se puede desplazar.

1.1.3.3 Estrategia Basada en Gramáticas

Esta estrategia usa como base la generación de gramáticas, las cuales pueden ser utilizadas para generar contenidos y también para definir las diferentes características topológicas de un tipo de entorno, sustentándose en la disposición de los contenidos según ciertas reglas. Las gramáticas libres de contextos [5] son las candidatas potenciales para esta estrategia. Para una mayor comprensión de esta estrategia proponemos el siguiente ejemplo:

Se desea generar un entorno de interiores para videojuegos (Ver figura 3) donde los contenidos de la lógica del videojuego (oponentes, vidas, bonos) son colocados de manera aleatoria. Para llevar a cabo este proceso se necesita primeramente generar el entorno de interiores y luego colocar los contenidos. El primer paso puede ser resultado de usar una gramática de grafos [5] en la cual se establece las interrelaciones posibles entres las diferentes habitaciones, en este caso llamadas Secciones (Ver figura. 4 inciso a). Para el segundo, se crean gramáticas de contenidos cuyo principio seria construir aleatoriamente cierta cantidad de estos elementos. Vale la pena analizar que el ejemplo no incluye las interrelaciones que pueden existir entre los contenidos de la lógica de un juego lo cual puede dar como resultado gramáticas un poco más complejas que permite generar niveles más variables y complejos (Ver figura. 4 inciso b).

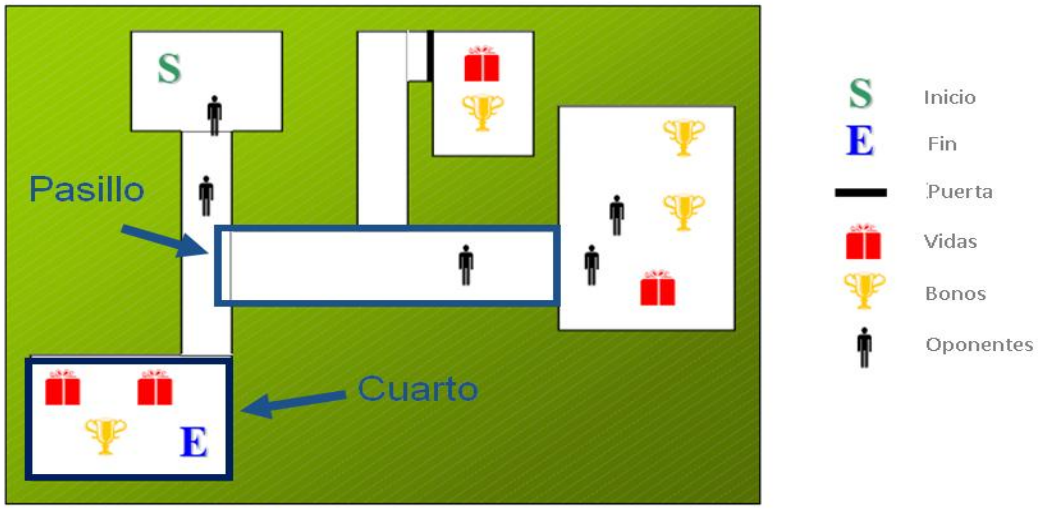


Figura 3: Prototipo de Entorno de Interiores para un Videojuegos.

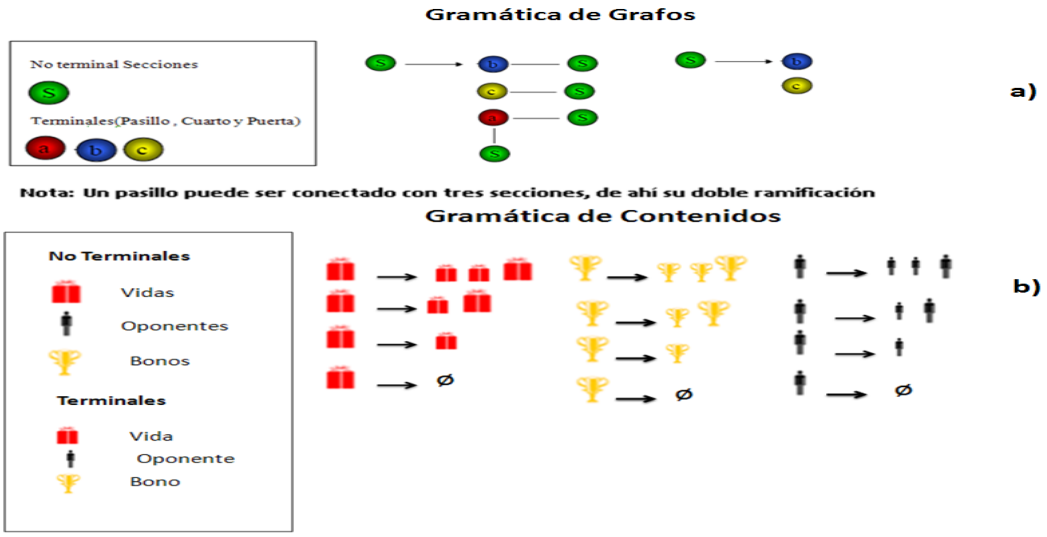


Figura 4: Gramáticas libres de Contexto. a) Gramática de Grafos para generar el entorno de interior y b) Gramática de Contenidos para generar los elementos de la lógica de un videojuego.

1.1.4 Generación de Entornos Virtuales basada en Capas

La generación por capas de entornos infinitos es una técnica que permite recrear entornos infinitos, de forma independiente, de fácil control y con mucha reutilización. Antes de explicar esta técnica se hace necesario conocer una serie de elementos sin los cuales no es posible comprender su funcionamiento. Estos son:

Contenido: Por contenido se entiende todas aquellas geometrías que conforman un entorno virtual.

Estrategia de Generación: Se le denomina a las técnicas y algoritmos de generación de contenidos y de entornos virtuales.

Capa de Generación: Elemento encargado de efectuar el proceso de generación de contenidos en una porción del entorno basado en una estrategia definida y tomando en consideración el espacio que controla según su Topología.

Topología de una Capa: Encierra la forma que esta tiene y delimita el espacio que controla, puede ser 2D o 3D. Las topologías pueden ser capas 2D con forma de cualquier primitiva (Rectángulo, Circunferencia, Triangulo). De igual forma las 3D (Cubo, Esfera).

Cámara: Representa al usuario en el sistema y es un punto esencial para la generación en tiempo real.

Memoria a corto plazo: Le provee al generador un espacio de almacenamiento de información que se borra cada cierto periodo de tiempo y recoge aquellos contenidos que son significativos para la aplicación, ejemplo los elementos de la lógica de un videojuego.

Generador de Entornos: Elemento encargado de orquestar todas las capas dentro de un entorno con vista a lograr la generación del mismo.

La generación por capas de entornos infinitos es una técnica que se basa en un sistema de n capas, que pueden estar asociadas a un sistema de coordenadas local o global de la escena según sea necesario. Estas son completamente independientes, pueden generar contenidos de forma aleatoria o mediante una estrategia definida (Ver figura 5).

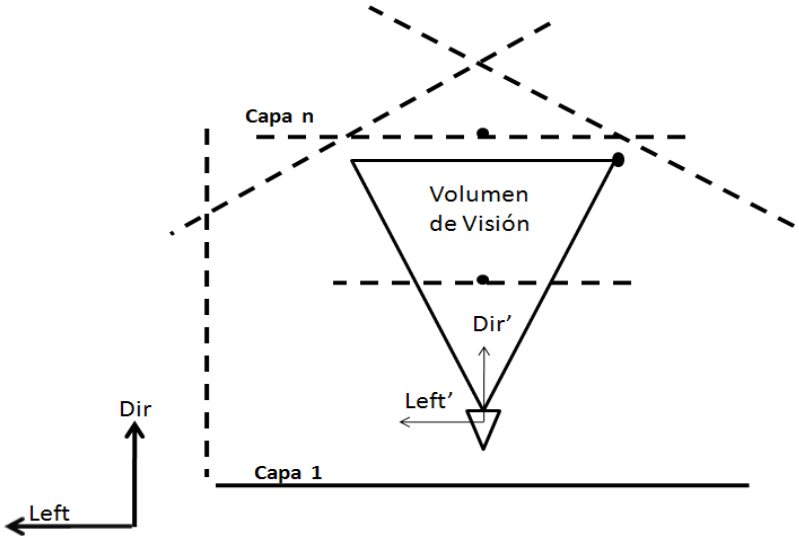


Fig. 5. Representación gráfica del generador.

Como se puede observar las capas pueden orientarse respecto a cualquier sistema de referencia e incluso trasladarse espacialmente. El sistema de generación siempre se traslada relativamente respecto a la cámara lo que permite lograr una estabilidad en el número de capas a controlar, y generar contenidos en posiciones ventajosas para su visibilidad.

Todas las capas son controladas por el Generador de Entornos según la estrategia de generación establecida. La reutilización de contenidos es su principal ventaja, permite reubicar contenidos, la mayor cantidad posible, variando solamente algunas propiedades geométricas de los modelos.

Las capas más pequeñas, que serian capas que generan el contenido más cercano a visualizarse podrían ubicarse más cercanas al cono de visualización de la cámara, otras capas, de mayor dimensión pudieran estar ubicadas más lejanas a la cámara, y estas generarían contenidos no tan frecuente como las capas más activas, pero lo más importante, es que el generador no está sujeto a ninguna restricción de posicionamiento de capas ni de tamaño, para cada aplicación es necesaria una configuración diferente de las capas para su correcto funcionamiento y uso de todas sus prestaciones.

El ciclo de generación se ejecuta en todos en los frames. El orden del proceso es:

1. Se actualizan todas las Capas relativamente a la cámara según la Estrategia de Generación definida.
2. Se realiza un chequeo de colisiones entre capas. En los casos de colisión se corrigen si la estrategia no la concibe como parte de ella.
3. Se ordena a cada capa generar los contenidos y posicionarlos, o posicionarlos solamente según su topología y estrategias definidas. En este punto el generador selecciona si en el espacio de generación se deben posicionar nuevos contenidos o colocar algunos almacenados en la Memoria a corto plazo.
4. Se actualiza el tiempo de vida de la memoria. En caso de que este haya llegado a su fin se limpia la memoria y se condiciona para almacenar nuevos elementos.

1.2 Interfaces Gráficas de Usuarios

La Interfaz Gráfica de Usuario, también conocida como GUI (*Graphic User Interface*), es la parte de un programa o sistema que interactúa con el usuario. La interfaz abarca todos los botones, iconos y ventanas que representan funciones, acciones o información.

La idea fundamental en el concepto de interfaz es el de mediación, entre hombre y máquina. La interfaz es lo que "media", lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano y una máquina como la computadora.

Resumiendo podemos decir que, una interfaz de software es la parte de una aplicación que el usuario ve y con la cual interactúa. Está relacionada con la estructura, la arquitectura, y el código que hace el trabajo del software, pero no se confunde con ellos. La interfaz incluye las pantallas, ventanas, controles, menús, la ayuda en línea, la documentación y el entrenamiento. Cualquier cosa que el usuario ve y con lo cual interactúa es parte de la interfaz.

Desde el punto de ingeniería de software, la interfaz de usuario juega un papel preponderante en el desarrollo y puesta en marcha de todo sistema. Es la carta de presentación del mismo y en ocasiones resulta determinante para la aceptación o rechazo de todo un proyecto. En promedio, se estima que del 35% al 45% de los gastos destinados a un proyecto son direccionados al diseño de la interfaz [6].

Actualmente las GUI dejan mucho que desear en términos de diseño y comunicación, contamos con las herramientas y los medios para desarrollarlas sin embargo no los aprovechamos y nos hemos acostumbrado a los malos diseños. Teniendo en cuenta el lugar importante que ocupan las interfaces en todos los sistemas, debido a que es el principal medio de comunicación con el exterior, se ha planteado como meta el desarrollo de interfaces inteligentes, fáciles de aprender y usar, que permitan a los usuarios hacer su trabajo de una manera cómoda y rápida en la manera que hace más sentido para ellos, en vez de tener que ajustarse al software.

El desarrollo de interfaces de usuarios siempre ha sido y sigue siendo un tema clave, que puede incluso llegar a tener un impacto mayor que otros factores a la hora de definir el éxito de un producto en el

mercado. Interfaces innovadoras, interactivas y creativas, auguran un gran éxito para el producto, un ejemplo de esto ha sido la interfaz táctil de iPod (Apple).

Buscar información sobre lo que realmente determina una buena interfaz puede ser algo muy complicado. Sin embargo, todos los desarrolladores y conocedores del tema están de acuerdo en que una interfaz sencilla, intuitiva y portable catapulta el éxito de un programa; sin embargo, una interfaz compleja y con un alto grado de aprendizaje influye muy negativamente sobre el software, pudiendo hacer incluso que fracase a pesar de ser enormemente eficiente y eficaz.

1.2.1 Características humanas del diseño de interfaz

En el momento de diseñar interfaces de usuario deben tenerse en cuenta las habilidades cognitivas y de percepción de las personas, y adaptar el programa a ellas. Así, una de las cosas más importantes que una interfaz puede hacer es reducir la dependencia de las personas de su propia memoria, no forzándoles a recordar cosas innecesariamente o a repetir operaciones ya realizadas (por ejemplo, introducir un mismo dato repetidas veces).

Las personas tienen unas habilidades distintas de las de la máquina, y ésta debe utilizar las suyas para soslayar las de aquella, para esto se debe de tener en cuenta los siguientes aspectos [7]:

- *Velocidad de Aprendizaje:* Se pretende que la persona aprenda a usar el sistema lo más pronto posible.
- *Velocidad de Respuesta:* El tiempo necesario que utiliza un usuario para realizar una operación en el sistema.
- *Tasa de errores:* Porcentaje de errores que comete el usuario.
- *Retención:* Cuánto recuerda el usuario sobre el uso del sistema en un período de tiempo.
- *Satisfacción:* Se refiere a que el usuario esté a gusto o no con el sistema.

Además de éstos existen otros a considerar como es el caso de las características físicas de las personas, el ambiente o lugar donde va a ser usado el sistema y la cultura de los usuarios, este es un factor importante si el sistema va a lanzarse al mercado internacional.

1.2.2 Pasos para el diseño de una interfaz de usuario

En el proceso de diseño de una interfaz de usuario se pueden distinguir cuatro fases o pasos fundamentales:

Reunir y analizar la información del usuario:

Es decir, definir qué tipo de usuario va a utilizar el programa, qué tareas va a realizar y cómo las va a realizar, qué exigen los usuarios del programa, en qué entorno se desenvuelven los usuarios (físico, social, cultural).

Diseñar la interfaz de usuario:

Es importante dedicar tiempo y recursos a esta fase, antes de entrar en la codificación. Aquí se definen los objetivos de usabilidad del programa, las tareas del usuario, los objetos y acciones de la interfaz, los iconos, vistas y representaciones visuales de los objetos, los menús de los objetos y ventanas. Todos los elementos visuales se pueden hacer primero a mano y luego refinar con las herramientas adecuadas.

Construir la interfaz de usuario:

Es importante realizar un prototipo previo, una primera versión del programa que se realice rápidamente y permita visualizar el producto para poderlo probar antes de codificarlo definitivamente.

Validar la interfaz de usuario:

Se deben realizar pruebas de usabilidad del producto, y de ser posible, con los propios usuarios finales del mismo.

Es importante, en resumen, realizar un diseño que parta del usuario, y no del sistema.

1.2.3 Bibliotecas de desarrollo de interfaces gráficas de usuarios

1.2.3.1 Glade

Glade es una herramienta de desarrollo visual de interfaces gráficas para GTK+ y GNOME, bajo licencia GNU GPL. Es independiente del lenguaje de programación y predeterminadamente no genera código fuente sino un archivo XML [8].

GladeXML es un formato XML que Glade usa para almacenar los elementos de las interfaces diseñadas. Estos archivos pueden emplearse para construirla en tiempo de ejecución mediante la biblioteca libglade. Algunas versiones de Glade permitían generar automáticamente el código que generaría las interfaces; pero fue desaconsejado y descontinuado.

1.2.3.2 GTK+

GTK+ fue desarrolla inicialmente como un conjunto de herramientas para el programa de manejo de imágenes GIMP. GTK+ es una biblioteca construida sobre GDK (el conjunto de herramientas de dibujo GIMP), que a su vez utiliza las funciones de Xlib. Contiene los objetos y funciones que permiten crear una interfaz gráfica de usuario. Maneja *widgets* como ventanas, botones, menús, etiquetas, deslizadores, y pestañas [9].

Actualmente su uso se ha expandido, es muy usado en el desarrollo de muchos programas, principalmente en los programas para sistemas GNU/Linux. GTK+ es un API orientado a objetos, escrito completamente en C, soporta la idea de clases y funciones de respuesta (es decir punteros a funciones).

GTK+ se basa en varias bibliotecas del equipo de GTK+ y de GNOME entre las que se encuentran:

- GLib: Biblioteca de bajo nivel, es la estructura básica de GTK+ y GNOME.
- GTK: Biblioteca que contiene los objetos y funciones para crear la interfaz.
- GDK: Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.

1.2.3.3 Qt

Es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. Utiliza C++ de forma nativa, aunque además permite programar en otros lenguajes. Corre sobre las plataformas más usadas mundialmente. Además de posibilitar el desarrollo de interfaces gráficas de usuario, incluye otras series de características como son el acceso a base de datos SQL, parser de XML, y una API unificada multiplataforma para el manejo de ficheros [10].

Cuenta con un entorno de desarrollo integrado (IDE de sus siglas en inglés) propio llamado Qt Creator, este se adapta a las necesidades de los desarrolladores de Qt y en él se juntan tanto las bibliotecas Qt como últimas herramientas de desarrollo adicionales como parte del SDK de Qt entre las que se encuentra el *Qt Designer*, una herramienta de Qt para el diseño y la creación de interfaces gráficas de usuario (GUI), por lo que ofrece todo lo necesario para empezar con el desarrollo de Qt en una sola instalación multiplataforma [11]. Qt puede integrarse además con otros IDE's como el Visual Studio para los sistemas Windows y Eclipse en Linux.

A partir de la versión 4.5.0 la licencia de Qt es LGPL, lo que ha aumentado aún más el uso de esta biblioteca, ya que anterior a esta versión la licencia para los sistemas operativos Windows era comercial. La última versión hasta la fecha de Qt puesta a disposición por QtSoftware – la compañía que desarrolla Qt – es la 4.6.1 y para el Qt Creator la versión más actual es la 1.3.1. Desde su nacimiento, la popularidad de Qt ha crecido sin cesar y sigue creciendo hasta nuestros días. Este éxito es un reflejo tanto de la calidad de Qt y de lo agradable que es usar. En la última década, Qt ha pasado de ser un producto utilizado por unos pocos a uno que es utilizado diariamente por miles de clientes y decenas de miles de desarrolladores de código abierto en todo el mundo [12].

1.3 Motores gráficos

1.3.1 OSG

OpenSceneGraph (OSG) [13] es un motor gráfico 3D de código abierto, de alto rendimiento y multiplataforma. Esta herramienta está basada en el concepto de grafos de escena, provee una plataforma orientada a objetos que utiliza OpenGL como API gráfica. OSG está disponible tanto para uso

comercial como no comercial. Está escrito completamente en C++ estándar y OpenGL, hace un uso extensivo de la STL y de los patrones de diseño. Cuenta con una buena documentación.

Aunque la librería está actualmente en desarrollo ya está siendo utilizada por distintos usuarios tanto en el ámbito de la investigación como a nivel comercial. Incluso existen otros desarrollos de código abierto que dan soporte y extienden las capacidades de OpenSceneGraph como, por ejemplo, Open Producer [14] y VR Jugger [15]. En la actualidad es utilizado para el desarrollo de aplicaciones gráficas de alto rendimiento tales como, simuladores de vuelo, juegos, aplicaciones de realidad virtual y visualización científica.

Características

Los principales puntos fuertes de OpenSceneGraph son su rendimiento, portabilidad y las ganancias de productividad asociados al uso de un grafo de escena a continuación mostramos sus características con más detalle:

Rendimiento: Soporta view-frustum culling, occlusion culling, small feature culling, LOD, vertex arrays, vertex buffer objects, OpenGL Shader Language y display lists. Todo esto hace de OSG una de los motores gráficos de más alto rendimiento disponible [16].

Productividad: El núcleo de OSG encapsula la mayoría de las funcionalidades de OpenGL incluyendo las últimas extensiones de este, provee optimización de la visualización, y un conjunto de bibliotecas adicionales las cuales hacen posible desarrollar aplicaciones gráficas de alto rendimiento muy rápidamente. Combinando las experiencias de otros motores gráficos con modernos métodos de ingeniería de software como patrones de diseño, se ha logrado el diseño de una biblioteca robusta y extensible [16].

Portabilidad: El núcleo de OSG ha sido diseño para tener una dependencia mínima de cualquier plataforma, requiriendo poco más que Standard C++ y OpenGL. Esto ha permitido que OSG fuese rápidamente portado a un amplio rango de plataformas. Originalmente fue desarrollado en IRIX, luego portado a Linux, Windows, FreeBSD, Mac OSX, Solaris, HP-UX, AIX y hasta PlayStation2 [16].

Soporte multi-lenguaje: OSG está disponible además en los lenguajes Java, y Pyhton.

Extensiones Soportadas: Para la lectura y escritura de archivos la biblioteca (osgDB) proporciona una amplia variedad de formatos por medio de un mecanismo extensible de plugins dinámicos. Actualmente incluye 55 plugins para cargar formatos 3D y formatos de imágenes [16].

Algunos de los formatos de modelos 3D soportados son COLLADA, 3D Studio MAX (.3ds), Peformer (.pfb), Quake Character Models (.md2), Direct X (.x), VRML 1.0 (.wrl), AC3D (.ac) y el formato nativo .osg. Los formatos de imágenes soportados son rgb, .gif, .jpg, .png, .tiff, .pic, .bmp, .dds, .tga y .quicktime. Además, brinda soporte para texto por medio de los plugins .freetype y .txf.

1.3.2 Crystal Space

Crystal Space [17] es un framework para el desarrollo de aplicaciones 3D, escrito en C++ por Jorrit Tyberghein usando un diseño orientado a objetos. Fue fundado el 26 de agosto del 1997. Crystal Space se usa típicamente como motor de juego pero el framework es más general y puede ser usado para cualquier tipo de visualización 3D. Crystal Space es muy portable y se ejecuta en Microsoft Windows, Linux, UNIX, y Mac OS X.

Soporta 6 verdaderos grados de libertad, iluminación con colores, espejos, trasparencia alfa, superficies reflectantes, sprites 3D, texturas procesuales, radiosidad, sistemas de partículas, halos, niebla volumétrica, scripting mediante Python u otros lenguajes, soporte de pantalla de 8, 16 y 32 bits de profundidad de color. Las Apis soportadas son OpenGL para Windows, Linux, Macintosh y OS/2, Direct3D para Windows y Glide para Linux y Windows.

Crystal Space está licenciado bajo la LGPL, que permite su copia, modificación y redistribución de las modificaciones (con la restricción de que cualquier derivado debe llevar una licencia del mismo tipo). Es software libre.

Características

A continuación se citan las características más importantes de Crystal Space:

Portabilidad: Plataforma altamente independiente con buena plataforma de abstracción. Oficialmente se ejecuta en Linux, OSX y Windows utilizando una amplia gama de compiladores e IDE's incluyendo gcc,

MinGW y Visual C++ 8 y 9. Incluye varios módulos que requieren muy pocas e incluso ninguna dependencia externa. Escrito en C++, utiliza muy pocas extensiones non-estándar [18].

Modularidad: Modularizado utilizando un marco componente de peso ligero, SCF, basado en el modelo COM, los componentes pueden ser fácilmente cargados. Además, permite vinculaciones con otros lenguajes de programación como Java, Python y Perl [18].

Renderizado y Luces: Soporta sistemas de partículas, terrenos, radiosidad, C-buffer y superficies curvas y superficies reflectantes. Soporta hardwares acelerados en tarjetas y plataformas de los más conocidos vendedores. Respecto a las luces, soporta Lightmaps, Bumpmapping, luces dinámicas, multicolores y radiosidad precalculada para los Lightmaps [18].

Shaders: Soporta shaders definidos en XML Markup y shaders implementados utilizando Cg. Los shaders más complicados pueden ser construidos utilizando una sintaxis compuesta por módulos más sencillos que se consiguen con una combinación en carga.

Formatos que soporta: Los formatos con los que trabaja son 3DS, MD2 (Quake 2), OBJ, POV y ASE.

1.3.3 OGRE

OGRE (*Object-Oriented Graphics Rendering Engine*) [19] es un motor de renderizado 3D escrito en el lenguaje de programación C++, orientado a objetos, el cual está diseñado para hacer más fácil e intuitivo el trabajo de los desarrolladores de aplicaciones gráficas aceleradas por hardware.

Sus bibliotecas evitan la dificultad de la utilización de capas inferiores de librerías gráficas como OpenGL y Direct3D, y además, proveen una interfaz basada en objetos del mundo y otras clases de alto nivel. Usa un lenguaje de descripción de materiales que permite su gestión de forma independiente al código fuente de la aplicación, brinda soporte para vertex y shaders, HLSL (DirectX), GLSL (OpenGL) y Cg (DirectX/OpenGL). Brinda soporte para texturas PNG, JPEG, BMP, DDS y DXT/S3TC, así como texturas

variables en tiempo real. En cuanto al modelado, permite la edición externa desde Blender y 3D Studio Max.

OGRE no es un motor de juego propiamente dicho – aunque ha sido usado para el desarrollo de juegos – sino que es un engine de gráficos 3D general, por tanto, para aplicar características como sonido, inteligencia artificial, colisión, leyes físicas, etc., se necesita integrar a OGRE con otras librerías. Esto se debe a que no todo el que utilice un motor gráfico lo hará con el objetivo de desarrollar un juego y además los requisitos necesarios para desarrollar un juego u otro, pueden variar considerablemente por lo que incluir todas estas características al unísono conllevaría a que se tuviesen incluidas muchas funcionalidades que no se fueran a usar, siendo esto un mal diseño. De esta forma, posibilita que sea usado para el desarrollo de disímiles tipo de aplicaciones 3D (juegos, simuladores, aplicaciones de negocio, etc.), dándole al usuario la posibilidad de agregar el resto de las bibliotecas que necesite para el desarrollo de su aplicación.

Es software libre, bajo la licencia LGPL (Lesser General Public License) y con una comunidad muy activa por lo que cuenta con una muy buena documentación online. Ha sido utilizado en algunos videojuegos comerciales, como por ejemplo Ankh y Earth Eternal.

Características

Generales

- Interfaz orienta a objeto fácil de usar, diseñada para minimizar el esfuerzo requerido para visualizar escenas 3D. y para ser independiente de la implementación 3D (Direct3D o OpenGL). [20]
- Arquitectura de plugins que permite extender las funcionalidades del motor gráfico. [20]
- Buena y precisa documentación contando además con una activa comunidad en Internet.
- Disponible para Linux, Mac OSX y todas las versiones de Windows. [20]

Gestión de Escena

- BSP, Octrees, Occlusion Culling, nivel de detalle (LOD).

- Altamente personalizable.
- Grafo de escena jerárquico.

Renderizado

- Soporta multi-textura y multipass blending.
- Los objetos transparentes son gestionados automáticamente.
- Soporte para múltiple técnicas de materiales.
- Sistema de fuentes con fuentes TrueType y texturas pre-creadas.

Iluminación

- Per-vertex, Per-pixel, Light mapping.

Sombras

- Shadow mapping, shadow volume.
- Soporte de técnicas: modulative stencil, additive stencil, modulative projective.
- Texturas-sombras que se desvanecen a larga distancia.

Mallas

- Cargado de malla, Skinning, Progressive.
- Exportadores para herramientas de modelado incluidas 3D Studio Max, Maya, Blender y Wings3D.

Texturizado

- Básico, Multi-texturing, Bumpmapping, Mipmapping, Volumetric, Projected.
- Soporta PNG, JPEG, TGA, BMP y DDS.

Shaders

- Soporta vertex shaders y pixel shaders de bajo nivel escritos en ensamblador, y programas de alto nivel escritos en Cg, DirectX9 HLSL, y GLSL.

1.4 XML como formato para el almacenamiento de datos

XML (*Extensible Markup Language*) [21] significa lenguaje de marcas generalizado. Es un lenguaje usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre, creado por el Consorcio World Wide Web, W3C, en colaboración con un panel que incluye representantes de las principales compañías productoras de software.

Fue propuesto en 1996, y la primera especificación apareció en 1998. Desde entonces su uso ha tenido un crecimiento acelerado, que se espera que continúe durante los próximos años. Su principal novedad consiste en compartir los datos con los que trabajan todos los niveles, por todas las aplicaciones y soportes. Permitiendo compartir la información de una manera segura, fiable y fácil.

XML es extensible pues ofrece al usuario la posibilidad de definir sus propios elementos: el usuario puede definir marcas o etiquetas de su gusto, y estructurar el documento en función de dichas etiquetas. XML es una muy buena opción para escribir ficheros de configuración de programas. La existencia de librerías optimizadas para extraer información y modificar documentos XML facilita la tarea del programador. Un ejemplo típico son los ficheros de configuración usados en Glade [22].

En lo que respecta al uso y aplicaciones de XML, hay que decir que está presente en muchas áreas de la informática actual. Se utiliza mucho en el mundo de las aplicaciones web, también para marcar documentos de carácter variado como bibliotecas digitales, en la representación y transferencia de información del comercio electrónico y también está adquiriendo importancia en el mundo de las bases de datos, no sólo como soporte para la transferencia de datos sino como formato de almacenamiento.

1.4.1 Características

Entre las características de XML que han logrado que se convierta en un formato universal, se encuentran las siguientes:

- Es una arquitectura más abierta y extensible. No se necesita versiones para que pueda funcionar en futuros navegadores.

- Los documentos tienen una estructura que los hace legibles e inteligibles no sólo para los ordenadores, sino también para los humanos.
- Mayor consistencia, homogeneidad y amplitud de los identificadores descriptivos del documento con XML.
- Integración de los datos de las fuentes más dispares. Se puede hacer el intercambio de documentos entre las aplicaciones de la propia PC como en una red local o extensa.
- Datos compuestos de múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje permite agrupar una amplia variedad de aplicaciones, desde páginas web hasta bases de datos.
- Proporciona una representación estructural de los datos que ha probado ser ampliamente implementable y fácil de distribuir.

1.4.2 Ventajas

La principal ventaja de XML es que es un estándar con independencia del tipo de implementación seleccionado. Esto significa que podemos usar herramientas de distintos proveedores para estructurar datos con la especificación XML, almacenarlos en una base de datos, realizar búsquedas o ejecutar cualquier proceso. En conclusión, nuestros datos serán accesibles y procesables por todas las herramientas que siguen el estándar XML con independencia de su plataforma y su fabricante. Además, presenta otras ventajas como:

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- Permite una mejor organización de la información: Comparado con otros sistemas usados para crear documentos, el XML tiene la ventaja de poder ser más exigente en cuanto a la organización del documento, lo cual resulta en documentos mejor estructurados.

- La información será más accesible y reutilizable: Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones.

Conclusiones del capítulo

A lo largo de todo el capítulo se evidenciaron las diferentes técnicas de generación de entornos existentes, haciendo énfasis en la técnica de generación por capas, también se abordó todo lo relativo a las interfaces gráficas de usuario, presentándose los requisitos a tener en cuenta para desarrollar una GUI y se estudiaron las bibliotecas que permiten el desarrollo de interfaces gráficas de usuario para seleccionar la biblioteca a utilizar para el desarrollo del sistema. Además, se realizó un análisis de los motores gráficos libres de más prestaciones y un estudio del formato XML, mostrándose las ventajas que trae el uso de este. Todo esto encaminado a obtener los elementos necesarios para dar solución al objetivo de esta investigación.

CAPÍTULO 2: SOLUCIONES TÉCNICAS

Introducción

En este capítulo se expone la propuesta del sistema a obtener para lograr el objetivo de este trabajo. Además, se analiza la estructura del fichero donde se guardarán los datos persistentes y se hará una breve reseña de las herramientas a utilizar en el desarrollo de la aplicación.

2.1 Objeto de automatización

Con la realización de esta investigación se pretende automatizar el proceso de generación del Fichero de Configuración de las capas a utilizar por el Generador de Entornos Virtuales basado en Capas, mediante la creación de una herramienta que permita configurar las capas a utilizar por el generador, es decir, agregar capas el entorno, editar sus propiedades (posición de la capa en la escena, dirección, contenidos asociados, etc.), y que además permita, luego de realizado todo este proceso, generar el Fichero de Configuración correspondiente, logrando así un ahorro de tiempo para los desarrolladores.

2.2 Información que se maneja

La información que se maneja es la referente a las propiedades de las capas y los contenidos asociados a estas y a las escenas 3D.

2.3 Propuesta del sistema

Se propone una aplicación de escritorio que permita al usuario configurar las capas y contenidos a utilizar en una escena 3D y generar el Fichero de Configuración que va a ser usado por el Generador de Entornos Virtuales basado en Capas. Las principales propiedades de las capas que se podrán configurar son las siguientes:

- Vectores de orientación de la capa (*Dir, Left, Up*): Estos vectores, como su nombre lo indica, son los que permitirán orientar la capa hacia un eje de coordenadas determinado por el usuario.
- Posición de la capa (*Center*): Vector el cual nos indica la posición en que se encuentra ubicada la capa en la escena.

- Cámara: Esta propiedad nos permitirá definir la distancia que existirá entre la capa y la cámara y además si la capa sigue a la cámara o no.

- Topología: Esta propiedad nos permite modificar la topología actual de la Capa (*Plane, Cube o Point*) así como definir, de acuerdo con la topología seleccionada, el Ancho (*Width*), Alto (*Height*) y la Profundidad (*Depth*) de la capa.

- Estrategia: Permite seleccionar la estrategia que se va a utilizar para generar el contenido de la capa (*RandomOriented, RandomNonOriented*).

La aplicación además deberá cumplir con las siguientes restricciones:

- Las capas pueden tener o no contenidos asociados, en caso de tener, estos deben ser de un solo tipo de contenido.
- La escena puede tener o no contenidos compartidos, en caso de tener, estos pueden ser de ambos tipos de contenidos.
- Los formatos de los contenidos a cargar serán: .png .mesh .jpg .bmp .3dx .3ds.
- El fichero de configuración que se genera debe cumplir con las especificaciones y estructura que se proponen en el siguiente epígrafe.

Para la creación de la interfaz gráfica de usuario el sistema hará uso de la biblioteca Qt, en su versión 4.6.1. La decisión de escoger Qt estuvo basada fundamentalmente es que es una biblioteca multiplataforma, la cual tiene una gran popularidad y aceptación en la actualidad y que cuenta con una excelente documentación por lo que se hace muy fácil aprender a trabajar con ella. Además, distribuida bajo los términos de GNU Lesser General Public License (LGPL), Qt es software libre y de código abierto.

Se usará el motor gráfico OGRE en su versión 1.6 para la representación de las escenas que se van a configurar en la aplicación. Esta decisión estuvo basada en que OGRE es un motor gráfico de gran rendimiento, y está diseñado para aprovechar al máximo las potencialidades del hardware moderno. Cuenta con una documentación buena, aún cuando no se podría calificar de excelente, sin embargo, cuenta con una comunidad muy activa en Internet que logra suplir la falta de documentación detallada que pueda existir sobre algún tema en particular.

El Fichero de Configuración generado por la Herramienta estará en formato XML, se decidió este formato ya que es un formato robusto y flexible que ha ganado gran popularidad en los últimos años debido a su estándar abierto y libre y que permite una mejor estructura y organización de los datos. Para la manipulación de los datos de este fichero usamos el parser TinyXML, este es un simple y pequeño parser, escrito en lenguaje C++, que puede ser integrado fácilmente con otros programas, cuenta con una buena documentación en Internet y es libre.

2.4 Modelo de persistencia

2.4.1 Formato del fichero de configuración de las capas

Luego de haber configurado las capas se procede a generar el Fichero de Configuración llamado ConfigLayers, para ello se utilizará el formato XML. La extensión del fichero donde se almacenará la configuración de las capas será **.xml**.

2.4.2 Estructura general del fichero de configuración

El primer elemento que contiene el fichero es el encabezado XML: <?xml version="1.0"?>
La principal etiqueta del fichero se denominará *ConfigLayers*, esta tendrá anidada toda la información referida a la configuración de las capas así como demás elementos de la escena, como es el caso de los contenidos compartidos. Le sigue la etiqueta *Layers*, la cual contiene a todas las capas existentes en la escena y cuenta con el atributo *CantLayer* que nos indica la cantidad de capas con que cuenta el fichero. Junto a esta encontramos la etiqueta *SharedContents* la que contiene todos los contenidos compartidos que van a ser utilizados en la escena. De manera general la estructura del fichero de configuración sería la siguiente:

```
<?xml version="1.0"?>
< ConfigLayers >
   < Layers CantLayer="n" >
     < Layer id="1" >
     < Layer id="2" >
     < Layer id="1" >
   </Layers >
   < SharedContents >
```

```
      < Contents Type="GROUP" Cant="0" >
      < Contents Type="ENVIROMENT" Cant="0" >
   < /SharedContents >
< /ConfigLayers >
```

2.4.2.1 Etiqueta Layer

Por cada capa se definirá una etiqueta *Layer*. Esta etiqueta tendrá anidada toda la información referente a las propiedades de la capa, entre los que encontramos los vectores de dirección y posición de la capa, así como su topología y contenidos asociados entre otros. En el caso de los contenidos asociados a la capa estos serán ubicados en la etiqueta *Contents* especificándose el tipo de contenido y la cantidad, una capa tendrá contenidos grupos (GROUP) o contenidos ambientes (ENVIROMENT) pero no ambos. Una capa puede tener o no contenidos, en el caso de tenerlos puede contener uno o varios. Para una mayor comprensión de la estructura de la etiqueta *Layer* le mostramos el siguiente ejemplo:

```
<Layer>
 <Id>Capa1</Id>
 <Dir>
   <X>1</X>
   <Y>0</Y>
   <Z>0</Z>
 </Dir>
 <Left>
 <Up>
 <Center>
   <X>10</X>
   <Y>20</Y>
   <Z>0</Z>
 </Center>
 <CameraDistance>0.00</CameraDistance>
 <FollowCamera>0</FollowCamera>
 <Topology Type="Cube">
   <Depth>0.00</Depth>
   <Width>0.00</Width>
   <Height>0.00</Height>
  </Topology>
  <Strategy>
```

```
<Name>Random</Name>
      </Strategy>
      <Contents Type="GROUP" Cant="1">
        <Group Id="Grupo1">
          <Content Type="BASE">
          <Content Type="ENVIROMENT" Cant="1">
        </Group>
      </Contents>
    </Layer>
```

Los contenidos grupos estarán formados por un contenido base, el cual es obligatorio, y uno o más contenidos ambientes asociados a esta base, también se acepta un contenido grupo formado por un contenido base y ningún contenido ambiente asociado a este.

A continuación se muestra un ejemplo:

```
<Group Id="Grupo1">
   <Content Type="BASE">
     <Element Id="winter">Media/Winter.jpg</Element>
   </Content>
   <Content Type="ENVIROMENT" Cant="1">
     <Element Id="water">Media/Water lilies.jpg</Element>
   </Content>
</Group>
```

En el caso de los contenidos ambientes estos se ubican en la etiqueta *Contents* como antes se mencionaba. A continuación se muestra un ejemplo:

```
<Contents Type="ENVIROMENT" Cant="2">
   <Element Id="hills">Media/Blue hills.jpg</Element>
   <Element Id="flores">Media/Water lilies.jpg</Element>
 </Contents>
```

2.4.2.2 Etiqueta SharedContents

La etiqueta *SharedContents*, como su nombre lo indica, es la encargada de contener los contenidos compartidos que utilizará el generador. Los contenidos compartidos son aquellos que no se asocian a una

capa en específico sino que pueden ser utilizados por cualquier capa que los necesite, el generador es el encargado de asignarle estos contenidos a la capa que los solicite. La etiqueta *SharedContents* por tanto anidará estos contenidos los cuales, al igual que los contenidos asociados a las capas, pueden ser de dos tipos: contenidos grupos (GROUP) o contenidos ambientes (ENVIROMENT).Para una mayor comprensión de la estructura de la etiqueta le mostramos el siguiente ejemplo:

```
<SharedContents>
  <Contents Type="GROUP" Cant="1">
    <Group Id="Grupo2">
      <Content Type="BASE">
        <Element Id="hills">Media/Blue hills.jpg</Element>
      </Content>
      <Content Type="ENVIROMENT" Cant="1">
        <Element Id="sunset">Media/Sunset.jpg</Element>
      </Content>
    </Group>
  </Contents>
  <Contents Type="ENVIROMENT" Cant="3">
    <Element Id="winter">Media/Winter.jpg</Element>
    <Element Id="liles">Media/Water lilies.jpg</Element>
    <Element Id="sunset">Media/Sunset.jpg</Element>
  </Contents>
</SharedContents>
```

2.4.3 Formato del fichero para salvar un proyecto

Para que el usuario tenga la posibilidad de guardar el estado del proyecto en el que esté trabajando - y de esa forma poder continuar la configuración de las capas de la escena en otro momento – se hace necesario guardar esta información persistentemente. Para esto se crea un fichero en el formato XML cuya extensión será **.lcpro**, que es el acrónimo de Layers Configuration Project.

2.4.4 Estructura del fichero para salvar un proyecto

La principal etiqueta del fichero se denominará *LayersConfigProject*, esta tendrá anidada toda la información del estado del proyecto que incluye el nombre del proyecto que se guardará en la etiqueta *ProjectName*, la dirección en donde se encuentra el fichero, las propiedades de las capas y los

contenidos. Para almacenar la información de las capas y contenidos se usará la misma estructura del Fichero de Configuración explicado anteriormente. Para una mejor comprensión le mostramos a continuación la estructura general de este fichero:

```
<?xml version="1.0”?>
< LayersConfigProject >
  < ProjectName > Escena1.lcpro< /ProjectName >
  < ProjectDir > D:\User\Projects\< / ProjectDir >
  < ConfigLayers >
    < Layers CantLayer="n" >
      < Layer id="1" >
      < Layer id="2" >
      < Layer id="1" >
    </Layers >
    < SharedContents >
      < Contents Type="GROUP" Cant="0" >
      < Contents Type="ENVIROMENT" Cant="0" >
    < /SharedContents >
  < /ConfigLayers >
< /LayersConfigProject >
```

2.5 Herramientas de desarrollo

2.5.1 Microsoft Visual Studio 2008

Microsoft Visual Studio 2008 es un IDE que permite a los desarrolladores crear rápidamente aplicaciones de alta calidad y riqueza. Para ello cuenta con un conjunto de herramientas de desarrollo profesionales, las cuales conforman un sistema altamente productivo. Se utiliza solamente como editor de código, dado su fácil manejo e interfaz, y a su módulo integrado de ayuda y completamiento de código: Visual Assist. Se harán uso de bibliotecas estándares para enfocarlo a un esquema multiplataforma.

2.5.2 Rational Rose

Rational Rose es una herramienta CASE profesional que emplea el UML como lenguaje de modelado, que se ha convertido en la notación estandarizada empleada en Rational Rose para especificar, visualizar

y construir desarrollos de software y sistemas. Lo que permite que todo el equipo de desarrollo pueda comunicarse con un lenguaje y una herramienta. Rational Rose es el líder en el mercado en cuanto a herramientas para el análisis, modelado, diseño y construcción orientada a objetos se refiere. Brinda la posibilidad de visualizar, entender y refinar los requerimientos y arquitectura antes de enfrentar el código, evitando desperdiciar esfuerzo durante el ciclo de desarrollo. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Además, permite chequear la sintaxis UML, mantiene la consistencia de los modelos del sistema del software, genera la documentación automáticamente, posibilita la generación de código a partir de los modelos, entre otras funcionalidades.

2.5.3 Lenguaje de desarrollo

El lenguaje de programación a utilizar será C++, ya que es un lenguaje libre, portable y robusto, el cual está soportado bajo el paradigma de la Programación Orientada a Objeto, además de ser el ideal para el desarrollo de aplicaciones gráficas.

Conclusiones del capítulo

Con el capítulo presentado quedan trazadas las guías que servirán para realizar las siguientes fases del trabajo. Además, se tienen las bases por las cuales se regirá el desarrollo del sistema.

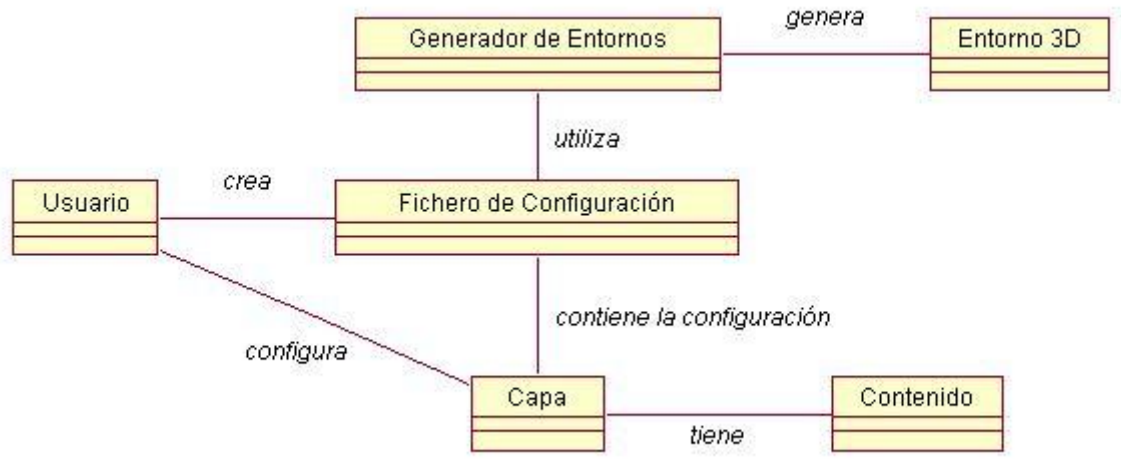
CAPÍTULO 3: CARACTERÍSTICAS DEL SISTEMA

Introducción

En este capítulo se define una visión más detallada y específica del sistema que se va a desarrollar, tomando como base las soluciones técnicas planteadas en el capítulo anterior. Se definen las reglas del negocio y el modelo del dominio. 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). Se describen los procesos – en forma de casos de uso – que dan cumplimiento a los requisitos funcionales.

3.1 Modelo de Dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Es un diagrama con los objetos que existen relacionados con el proyecto y las relaciones que hay entre ellos [23]. Debido a que no se cuenta con una definición total de los procesos de negocio, se plantea confeccionar un modelo de dominio que servirá como referencia a los posibles usuarios para la comprensión de todos los términos utilizados en la elaboración del sistema y como guía para el futuro diseño de este.



3.2 Glosario de Términos del Modelo de Dominio

El glosario de términos del modelo de dominio define los principales vocablos usados en el dominio modelado. A continuación se hace una breve descripción de los principales conceptos manejados en el modelo de dominio para lograr un mejor entendimiento del mismo.

Entorno 3D: Entorno virtual. Contiene la información gráfica de varios objetos 3D.

Contenido: Por contenido se entiende todas aquellas geometrías que conforman un entorno virtual 3D.

Capa: Elemento encargado de efectuar el proceso de generación de contenidos en una porción del entorno 3D.

Fichero de Configuración: Fichero que almacena toda la información referente a la configuración de las capas y contenidos de una escena 3D.

Generador de Entorno: Elemento encargado de organizar y controlar todas las capas dentro de un entorno 3D con vista a lograr la generación del mismo.

Usuario: Es el encargado de configurar las capas y de crear el fichero de configuración.

3.3 Captura de Requisitos

Los requisitos establecen que tiene que hacer exactamente el sistema que se construye [23]. Son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen. Los requisitos se dividen en dos grupos: los Requisitos Funcionales y los Requisitos No Funcionales.

3.3.1 Requisitos Funcionales

Los Requisitos Funcionales (RF) representan la funcionalidad del sistema y se modelan mediante diagramas de Casos de Usos [23]. Un requisito funcional especifica una acción que debe ser capaz de

realizar el sistema. A continuación se enumeran las principales funcionalidades que el sistema debe cumplir:

RF 1. Administrar Proyecto.
- RF 1.1 Crear Proyecto.
- RF 1.2 Cargar Proyecto.
- RF 1.3 Salvar Proyecto.

RF 2. Gestionar Capas.
- RF 2.1 Insertar Capa.
- RF 2.2 Eliminar Capa.
- RF 2.3 Configurar Propiedades de la Capa.

RF 3. Administrar Contenidos Capa.
- RF 3.1 Insertar Contenido Capa.
  - RF 3.1.1 Insertar Contenido Grupo.
  - RF 3.1.2 Insertar Contenido Ambiente.
- RF 3.2 Eliminar Contenido Capa.
- RF 3.3 Mostrar Contenidos Capa

RF 4. Administrar Contenidos Compartidos.
- RF 4.1 Insertar Contenido Compartido.
  - RF 4.1.1 Insertar Contenido Grupo.
  - RF 4.1.2 Insertar Contenido Ambiente.
- RF 4.2 Eliminar Contenido Compartido.
- RF 4.3 Mostrar Contenidos Compartidos.

RF 5. Generar Fichero de Configuración.

3.3.2 Requisitos No Funcionales

Los Requisitos No Funcionales (RNF) representan aquellos atributos o propiedades que debe exhibir el sistema, pero que no son una funcionalidad específica [23]. A continuación se enumeran las cualidades que la aplicación debe presentar para darle cumplimiento a los RF del sistema:

Usabilidad: Los usuarios del sistema deben tener conocimientos mínimos de Informática.

Hardware: Se necesita un ordenador con CPU Intel Pentium IV a 3.00 GHz, 1 Gb de RAM o superior. Mantiene compatibilidad con tarjetas gráficas de la familia NVIDIA (GeForce 9800 GT).

Restricciones en el Diseño e Implementación: Debe ser implementado en el lenguaje C++ estándar, basándose en la filosofía de Programación Orientada a Objetos.

Software: Compatible con la plataforma Windows.

Interfaz de Usuario: La aplicación presenta una interfaz de usuario sencilla, tendrá una barra de menús y una barra de herramientas en el panel superior y un panel de propiedades en el lateral izquierdo que permitirán llevar a cabo todas las funcionalidades.

3.4 Modelo de Casos de Usos del Sistema

El Modelo de Casos de Uso está formado por Actores, Casos de Uso y las relaciones entre ambos; este modelo describe lo que el sistema debe hacer por sus usuarios y bajo qué restricciones. A continuación se identifica los actores del sistema a desarrollar, y partir de los Requisitos Funcionales obtenidos anteriormente se concebirán los Casos de Uso del Sistema y sus respectivas descripciones.

3.4.1 Actor del Sistema

El actor del sistema representa el rol que juega una o varias personas, un equipo o un sistema automatizado, el cual interactúa con el sistema pero no forma parte de él. Un actor del sistema es aquel que se beneficia con los resultados de las funcionalidades del mismo.

Tabla 1. Actor del Sistema.

Actor	Descripción
Usuario	Es el encargado de configurar las capas, asignar los contenidos y generar el fichero de configuración.

3.4.2 Casos de Uso del Sistema

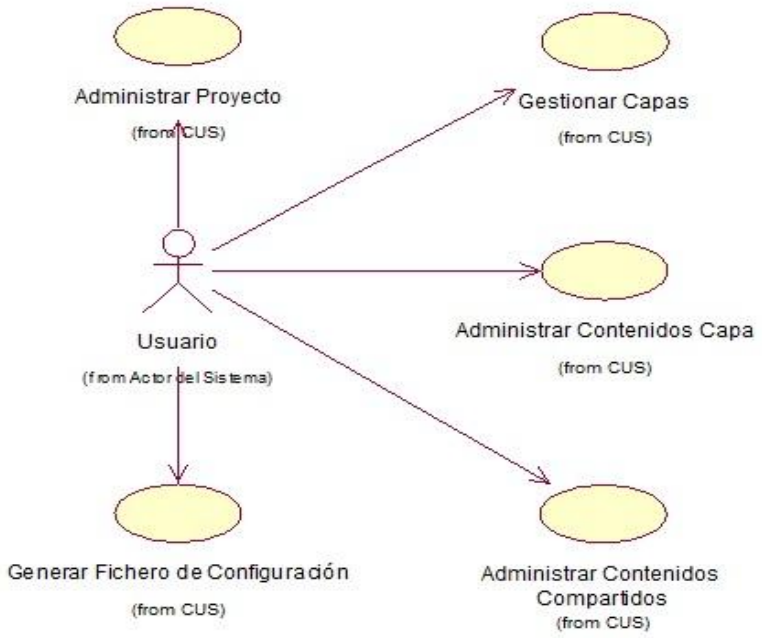
Los Casos de Uso (CU) son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Los CU capturan requisitos potenciales del software. Cada CU proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

A continuación se enumeran los Casos de Uso del sistema:

CU 1. Administrar Proyecto.
CU 2. Gestionar Capas.
CU 3. Administrar Contenidos Capa.
CU 4. Administrar Contenidos Compartidos.
CU 5. Generar Fichero de Configuración.

3.4.3 Diagrama de Casos de Uso del Sistema

Los diagramas de CU especifican la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. Es decir, muestran la relación entre los actores y los casos de uso de un sistema.



3.4.4 Especificación de los Casos de Uso en formato expandido

Para entender la funcionalidad asociada a cada CU no es suficiente con la representación gráfica del diagrama de CU, por esto se hace la expansión de mismo; la cual muestra más detalles de su funcionalidad y permite un mejor entendimiento del mismo.

Tabla 2. Descripción del caso de uso "Administrar Proyecto"

Caso de Uso:	Administrar Proyecto.
Actores:	Usuario.
Resumen:	El CU se inicia cuando el Usuario desea Crear, Cargar o Salvar un proyecto y termina con la ejecución de dicha acción.
Precondiciones:	**-**
Propósito:	Crear, Cargar o Eliminar un proyecto.
Referencias:	RF 1
Prioridad:	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el menú Archivo.	1.1 Brinda una serie de acciones a realizar: a) Si desea crear un proyecto ir a la sección “Crear Proyecto” b) Si desea cargar un proyecto ir a la sección “Cargar Proyecto” c) Si desea salvar un proyecto ir a la sección “Salvar Proyecto”
Sección “Crear Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Verifica si se ha realizado algún cambio en el proyecto actual que no haya sido salvado. 1.2 Crea un nuevo proyecto.
Sección “Cargar Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Verifica si se ha realizado algún cambio en el proyecto actual que no haya sido salvado. 1.2 Muestra un cuadro de diálogo que permite especificar la ruta del proyecto a cargar o cancelar la acción.
2 Especifica la ruta del proyecto y pulsa abrir.	2.1 Verifica que la ruta sea válida. 2.2 Verifica la validez de los datos del proyecto que se desea cargar. 2.3 Carga el proyecto.
Sección “Salvar Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Verifica si el proyecto ha sido salvado con anterioridad. 1.2 Salva el proyecto.
Prototipo de Interfaz	

Ver Anexo 1.	
Flujos Alternos	
Sección "Crear Proyecto"	
Acción del Actor	**Respuesta del Sistema**
	1.1 En caso de haber realizado algún cambio en el proyecto actual muestra un cuadro de diálogo con las opciones guardar o no guardar los cambios realizados.
2. Elige guardar los cambios.	2.1 Salva el proyecto y va al paso 1.2.
3. Elige no guardar los cambios.	3.1 Ir al paso 1.2.
4. Elige cancelar la operación.	4.1 Cierra el cuadro de diálogo.
Sección "Cargar Proyecto"	
Acción del Actor	**Respuesta del Sistema**
	1.1 En caso de haber realizado algún cambio en el proyecto actual muestra un cuadro de diálogo con las opciones guardar o no guardar los cambios realizados.
2. Elige guardar los cambios.	2.1 Salva el proyecto y va al paso 1.2.
3. Elige no guardar los cambios.	3.1 Ir al paso 1.2.
	1.2 Muestra un cuadro de diálogo que permite especificar la ruta del proyecto a cargar o cancelar la acción.
2. Elige cancelar la carga	2.1 Cierra el cuadro de diálogo.
	2.1 Si la ruta no es válida muestra un mensaje de error ("Ruta especificada incorrecta") y pasa al paso 1.2.

	2.2 Si los datos del proyecto cargado no son válidos se muestra un mensaje de error. (“Error al cargar el proyecto”) 2.3 Se crea un proyecto nuevo.
Sección “Salvar Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Si el proyecto no ha sido salvado se muestra un cuadro de diálogo que permite especificar la ruta en donde se va a salvar el proyecto o cancelar la acción.
2. Especifica la ruta a salvar	2.1 Verifica que la ruta sea válida, de no serlo muestra un mensaje de error (“Ruta especificada incorrecta”) y va al paso 1.1. 2.2 Salva el proyecto.
3. Elige cancelar la acción	3.1 Cierra el cuadro de diálogo.
Postcondiciones	Se crea, carga o salva un proyecto.

Tabla 3. Descripción del caso de uso “Gestionar Capas”

Caso de Uso:	Gestionar Capas.
Actores:	Usuario.
Resumen:	El CU se inicia cuando el Usuario Insertar, Eliminar o Configurar la capa y termina con la ejecución de dicha acción.
Precondiciones:	La capa debe estar seleccionada (Secciones “Eliminar Capa” y “Configurar Capa”).
Propósito:	Insertar Eliminar o Configurar una capa.
Referencias:	RF 2.
Prioridad:	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1 Selecciona el menú Capa	1.1 Brinda una serie de acciones a realizar: a) Si desea insertar capas ir a la sección "Insertar Capas". b) Si desea eliminar la capa ir a la sección "Eliminar Capa". c) Si desea configurar capa ir a la sección "Configurar Capa".
Sección "Insertar Capa"	
Acción del Actor	**Respuesta del Sistema**
	1.1 Crea la capa. 1.2 Muestra la capa en el cuadro de visualización.
Sección "Eliminar Capa"	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra un mensaje de alerta ("¿Desea eliminar la capa seleccionada?") con las opciones de eliminar o no la capa. 1.2 Elimina la capa seleccionada. 1.3 Actualiza el cuadro de visualización.
Sección "Configurar Capa"	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra un panel de edición con todas las propiedades de la capa seleccionada.
2 Configura las propiedades de la capa (Dir, Left, Up, Centro, Topología, Estrategia).	2.1 Verifica que los datos de las propiedades de la capa que se desea configurar sean correctos. 2.2 Actualiza la capa. 2.3 Actualiza el cuadro de visualización.
Prototipo de Interfaz	
Ver Anexo 2.	
Flujos Alternos	

Sección “Eliminar Capa”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra un mensaje de alerta (“¿Desea eliminar la capa seleccionada?”) con las opciones de eliminar o no la capa.
2. Selecciona no eliminar la capa.	2.1 Cierra el mensaje de alerta.
Sección “Configurar Capa”	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si los datos de la capa a configurar no son válidos muestra un mensaje de error (“Datos de la capa incorrectos”). 2.2 Ir al paso 1.1.
Postcondiciones	Se ha insertado, eliminado o configurado una capa.

Tabla 4. Descripción del caso de uso “Administrar Contenidos Capa”

Caso de Uso:	Administrar Contenidos Capa.
Actores:	Usuario.
Resumen:	El CU se inicia cuando el Usuario desea Insertar, Eliminar o Mostrar el (los) contenido(s) de una capa y termina con la ejecución de dicha acción.
Precondiciones:	La capa debe estar seleccionada.
Propósito:	Insertar, Eliminar o Mostrar un contenido de una capa.
Referencias	RF 3.
Prioridad	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1 Selecciona la opción Contenidos Capa.	1.1 Brinda una serie de acciones a realizar: a) Si desea insertar un contenido a la capa ir a la sección "Insertar Contenido Capa". b) Si desea eliminar un contenido de la capa ir a la sección "Eliminar Contenido Capa". c) Si desea mostrar los contenidos de la capa ir a la sección "Mostrar Contenidos Capa".
Sección "Insertar Contenido Capa"	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra una ventana con los campos necesarios para insertar un contenido.
2 Elige el tipo de contenido (Grupo o Ambiente) e introduce los datos (DirCont, IdCont) y pulsa insertar.	2.1 Verifica que los datos del contenido a insertar sean correctos. 2.2. Inserta el contenido. 2.3 Actualiza la capa.
Sección "Eliminar Contenido Capa"	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra una ventana con el listado de los contenidos asociados a la capa.
2 Selecciona el contenido a eliminar.	2.1 Elimina el contenido. 2.2 Actualiza la capa. 2.3 Actualiza el listado de contenidos.
Sección "Mostrar Contenidos Capa"	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra una ventana con el listado de los contenidos asociados a la capa.
Prototipo de Interfaz	
Ver Anexo 3.	

Flujos Alternos	
Sección "Insertar Contenido Capa"	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si los datos del contenido a insertar son incorrectos muestra un mensaje de error ("Datos del contenido incorrectos"). 2.2 Ir al paso 1.1.
Postcondiciones	Se ha insertado, eliminado o mostrado un contenido asociado a una capa.

Tabla 5. Descripción del caso de uso "Administrar Contenidos Compartidos"

Caso de Uso:	Administrar Contenidos Compartidos.
Actores:	Usuario.
Resumen:	El CU se inicia cuando el Usuario desea Insertar, Eliminar o Mostrar el (los) contenido(s) compartidos de la escena, y termina con la ejecución de dicha acción.
Precondiciones:	Debe existir al menos una capa.
Propósito:	Insertar, Eliminar o Mostrar los contenidos compartidos.
Referencias	RF 4.
Prioridad	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1 Selecciona la opción Contenidos Compartidos.	1.1 Brinda una serie de acciones a realizar: a) Si desea insertar un contenido compartido ir a la sección “Insertar Contenido Compartido”. b) Si desea eliminar un contenido compartido ir a la sección “Eliminar Contenido Compartido”. c) Si desea mostrar los contenidos compartidos ir a la sección “Mostrar Contenidos Compartidos”.
Sección “Insertar Contenido Compartido”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra una ventana con los campos necesarios para insertar un contenido compartido.
2 Elige el tipo de contenido (Grupo o Ambiente) e introduce los datos (DirCont, IdCont) y pulsa insertar.	2.1 Verifica que los datos del contenido a insertar sean correctos. 2.2. Inserta el contenido. 2.3 Actualiza los contenidos compartidos.
Sección “Eliminar Contenido Compartido”	
Acción del Actor	**Respuesta del Sistema**
	1.1 Muestra una ventana con el listado de los contenidos compartidos.
2 Selecciona el contenido a eliminar.	2.1 Elimina el contenido. 2.2 Actualiza los contenidos compartidos. 2.3 Actualiza listado de contenidos compartidos.
Sección “Mostrar Contenidos Compartidos”	
Acción del Actor	**Respuesta del Sistema**
.	1.1 Muestra una ventana con el listado de los contenidos compartidos.
Prototipo de Interfaz	
Ver Anexo 4.	

Flujos Alternos	
Sección "Insertar Contenido Compartido"	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si los datos del contenido a insertar son incorrectos muestra un mensaje de error ("Datos del contenido incorrectos"). 2.2 Ir al paso 1.1
Postcondiciones	Se ha insertado, eliminado o mostrado un contenido compartido.

Tabla 6. Descripción del caso de uso "Generar Fichero de Configuración"

Caso de Uso:	Generar Fichero de Configuración.
Actores:	Usuario.
Resumen:	El CU se inicia cuando el Usuario selecciona el menú Generar y selecciona la acción Generar Fichero de Configuración y termina el CU con la ejecución de dicha acción.
Precondiciones:	Deben de estar configurados las capas y contenidos de la escena.
Propósito:	Generar el fichero de configuración de las capas.
Referencias	RF 5.
Prioridad	Crítico.
Flujo Normal de Eventos	
Sección ""	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona la opción Generar Fichero de Configuración.	1.1 Muestra un cuadro de diálogo que permite especificar la ruta donde se va a salvar el fichero y la opción de generar o cancelar.

2 Especifica la ruta a salvar y pulsa Generar	2.1 Verifica que la ruta sea válida 2.2 Genera el fichero. 2.3 Muestra un mensaje de confirmación (“Fichero generado correctamente”).
Prototipo de Interfaz Ver Anexo 5.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si la ruta no es válida muestra un mensaje de error (“Ruta especificada incorrecta”) y pasa al paso 1.1
2.1.1 Elige cancelar la acción.	2.2 Cierra el cuadro de dialogo.
Postcondiciones	Se genera el Fichero de Configuración.

Conclusiones del capítulo

En este capítulo se definieron las cualidades y características que espera el usuario que tenga el sistema. Por tanto se dejan sentadas las bases para proceder con el diseño e implementación del sistema.

CAPÍTULO 4: DISEÑO E IMPLEMENTACIÓN

Introducción

En la primera parte del presente capítulo se aborda el diseño del sistema propuesto. Se define la arquitectura del sistema y se muestran los diagramas de clases agrupados por paquetes, además se exponen los diagramas de secuencia de la realización de los casos de uso. A continuación se pasa a la implementación del sistema, definiendo el estándar de código a seguir por los desarrolladores. Además, se elaborará el diagrama de despliegue y de componentes para el sistema.

4.1 Arquitectura utilizada

Se decidió usar una Arquitectura en Capas, debido a que es una de las más usadas ya que se ha demostrado que organizar los elementos de las aplicaciones en capas independientes puede lograr una mayor eficiencia durante el tiempo de desarrollo y luego durante el mantenimiento. Una capa no es más que una agrupación de componentes que tiene una responsabilidad bien definida, estas forman una jerarquía que cumple la siguiente regla: los componentes de una capa solo pueden utilizar componentes de su misma capa o una inferior, nunca de una superior.

La aplicación se divide en tres capas lógicas, la primera es la capa de presentación y se ocupa de la interacción del usuario con el sistema mediante la interfaz gráfica de usuario. La capa intermedia, es la encargada de la lógica de negocio, es básicamente el código al que recurre la capa de presentación para recuperar los datos deseados y la tercera capa contiene los datos necesarios para la aplicación.

4.2 Patrones de diseño

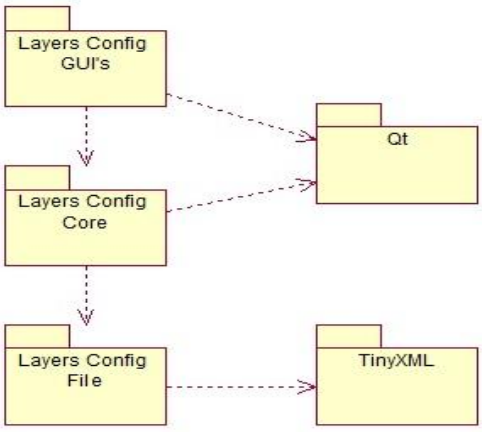
Un patrón describe un problema que ocurre una y otra vez en nuestro entorno y también la solución a dicho problema, de forma que puede reutilizarse continuamente. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Los patrones utilizados en el desarrollo del sistema son los siguientes:

- Experto: Nos indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. [24]
- Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. [24]
- Alta Cohesión: La información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase. [24]

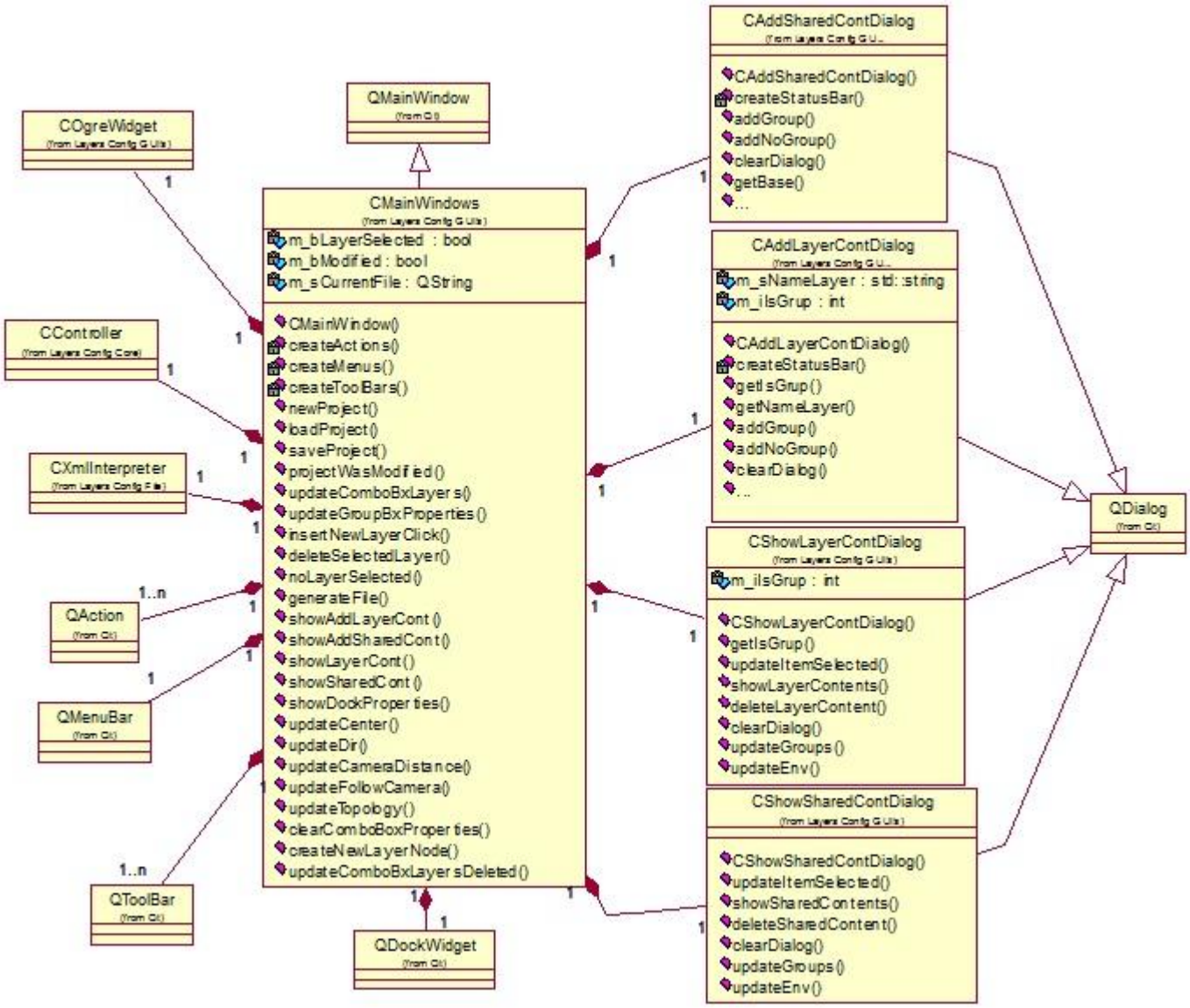
4.3 Diagramas de clases del diseño

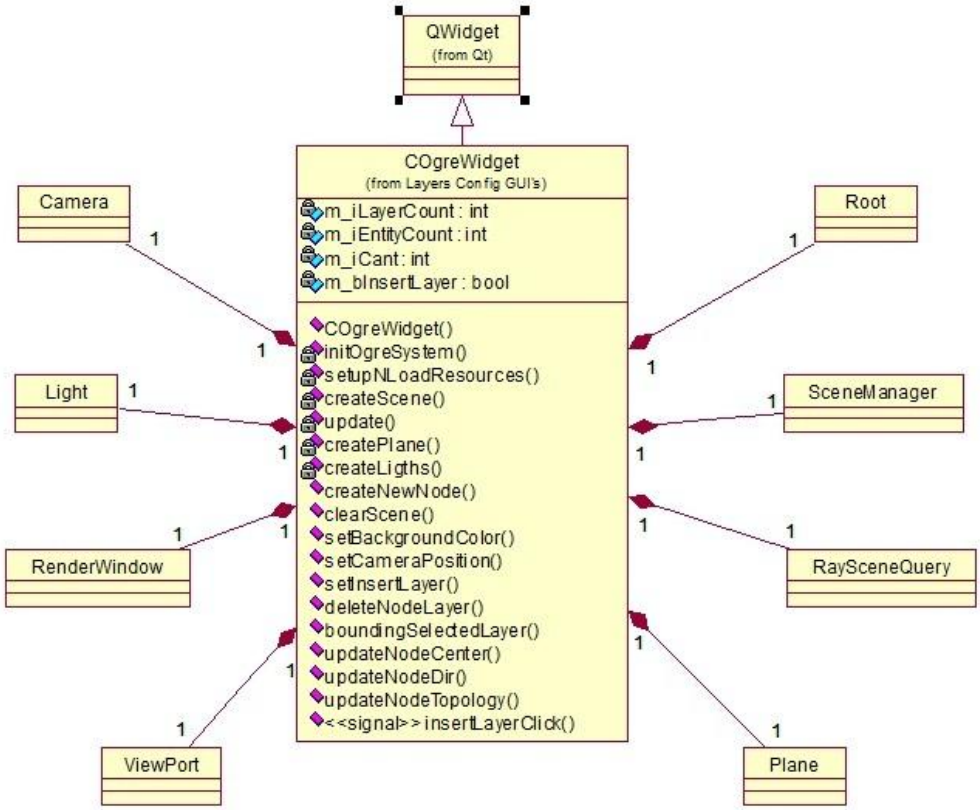
En este epígrafe se representan las clases pertenecientes al diseño del sistema así como sus relaciones, para lograr una mayor comprensión se han agrupado estas clases por paquetes.

4.3.1 Diagrama de paquetes del diseño

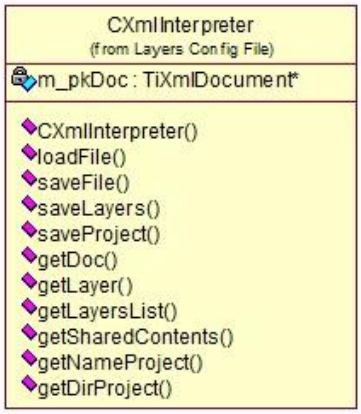


4.3.2 Diagrama de Clases del paquete Layers Config GUI's

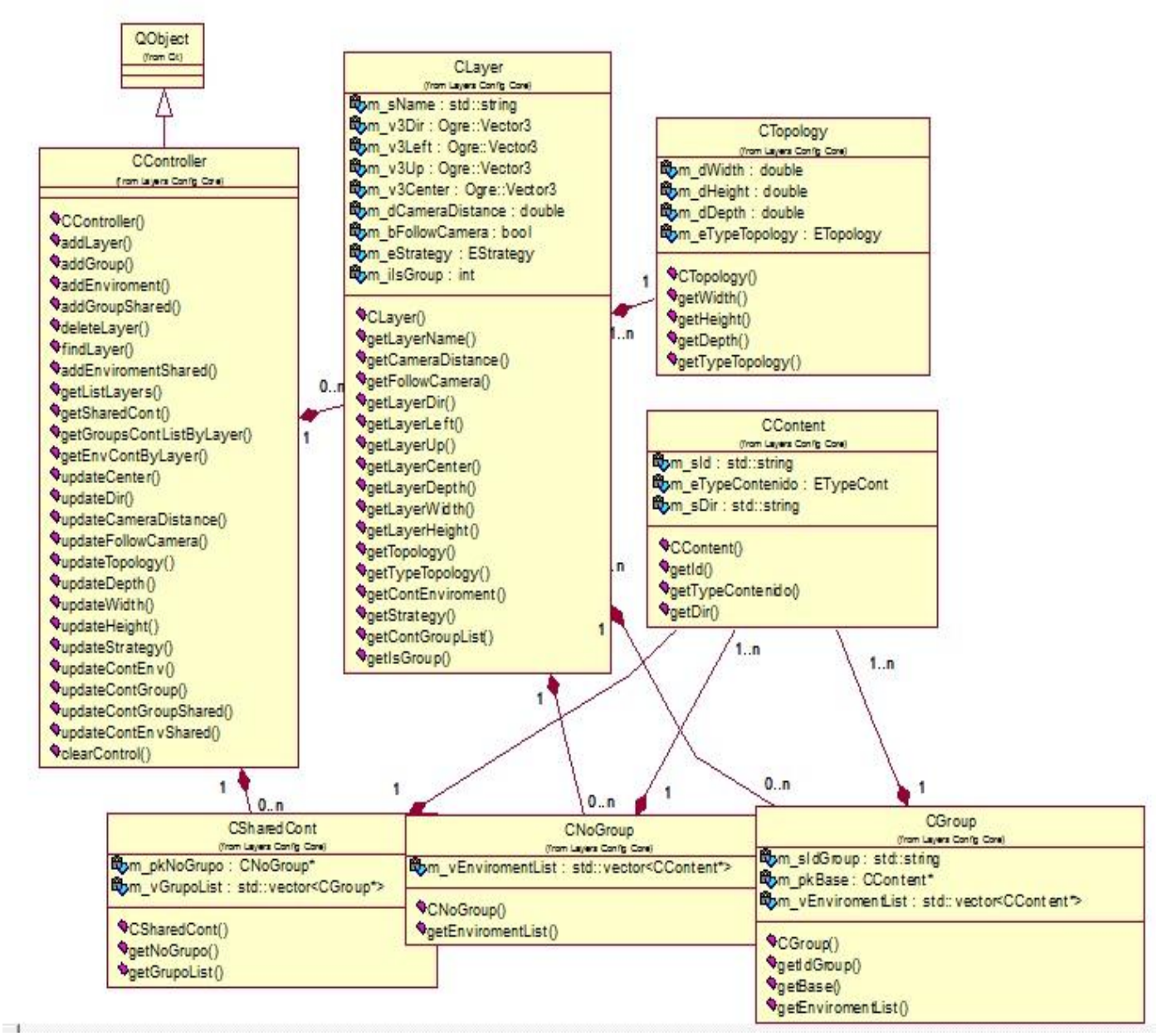




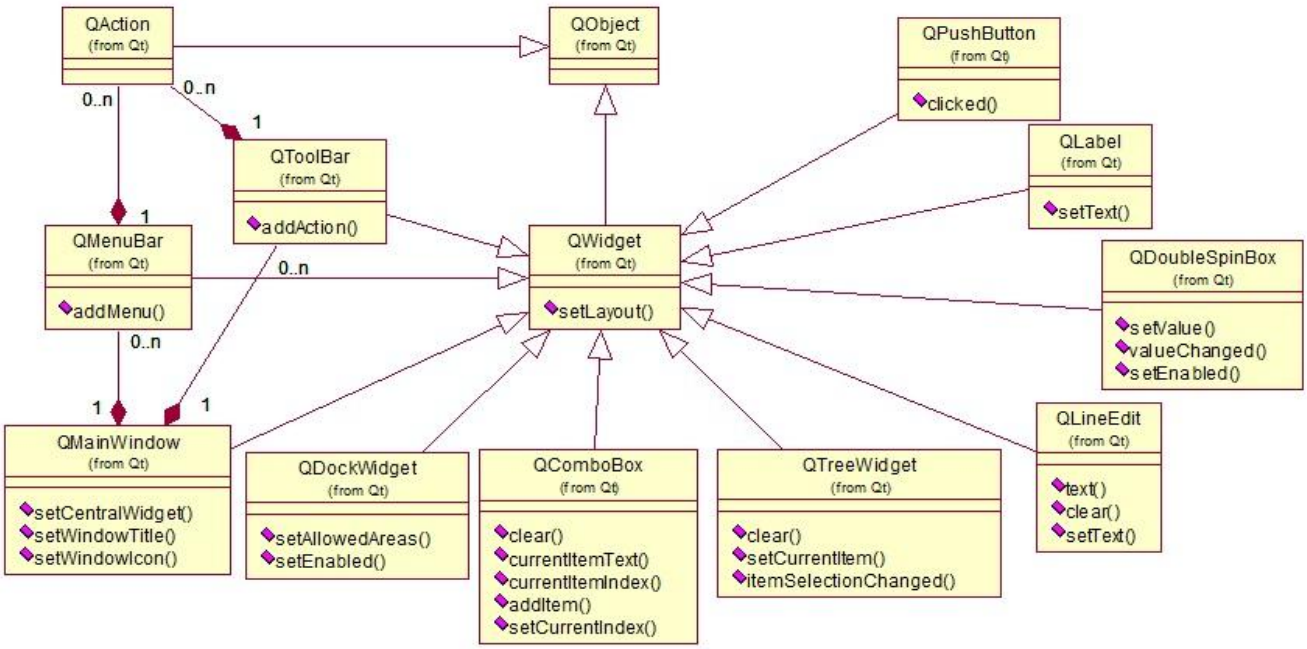
4.3.3 Diagrama de Clases del paquete Layers Config File



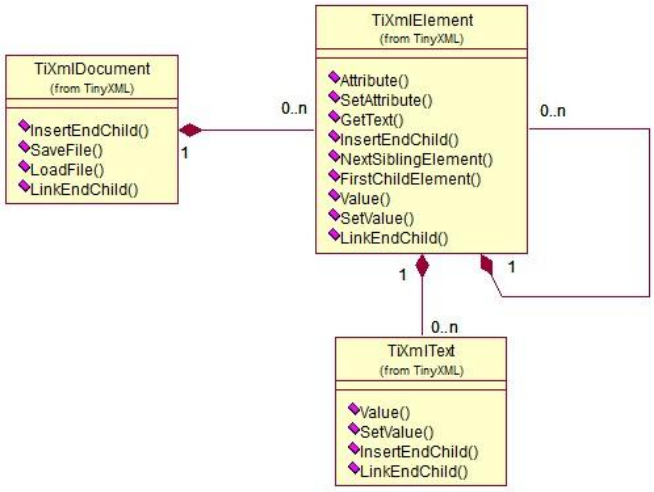
4.3.4 Diagrama de Clases del paquete Config Core



4.3.5 Diagrama de Clases del paquete Qt



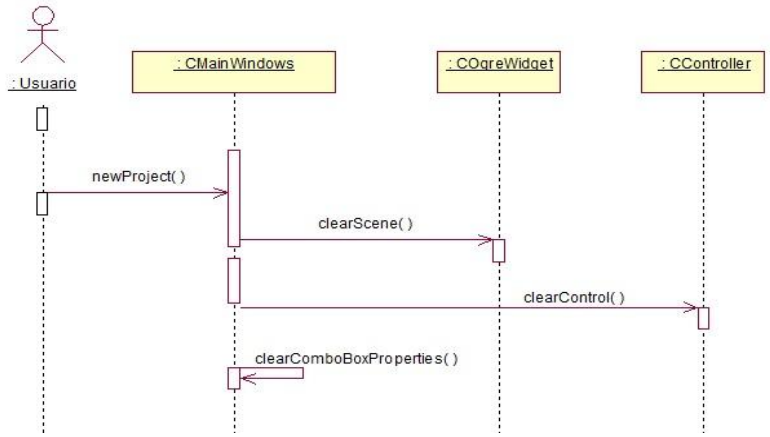
4.3.6 Diagrama de Clases del paquete TinyXML



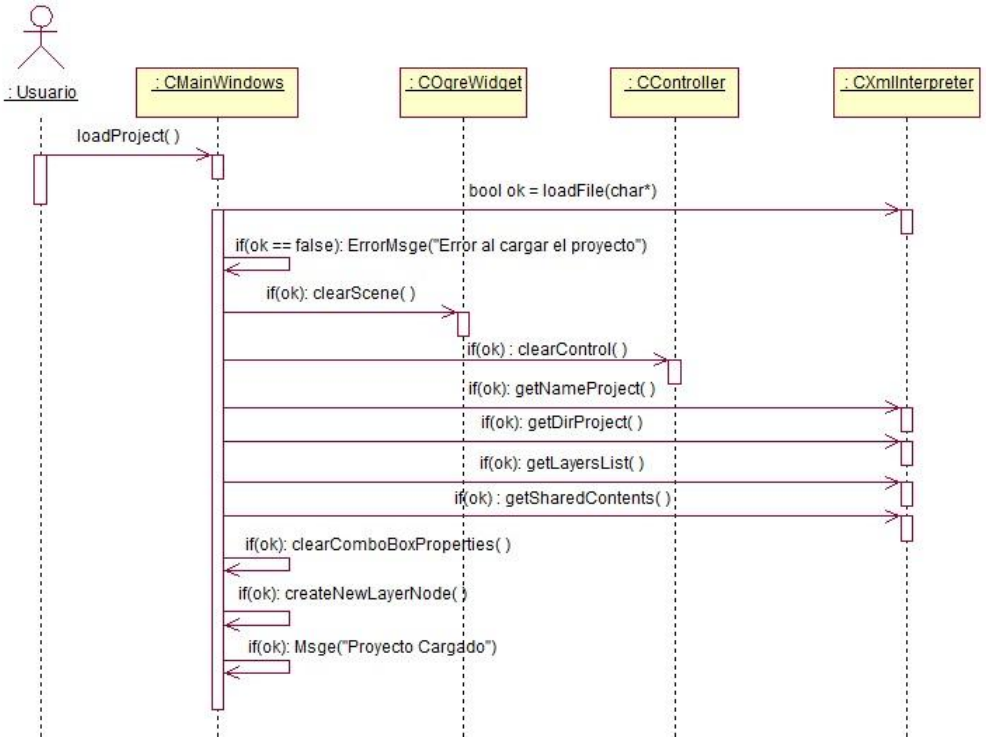
4.4 Diagramas de secuencia

A continuación se muestran los diagramas de secuencia correspondientes a cada uno de los escenarios de los casos de uso.

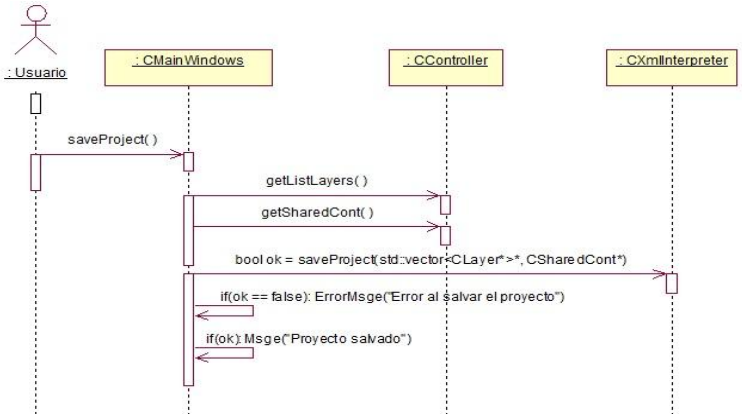
4.4.1 Diagrama de Secuencia: Crear Proyecto



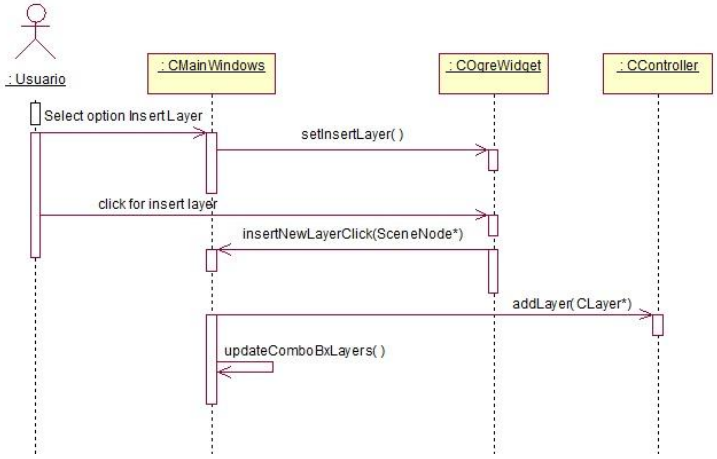
4.4.2 Diagrama de Secuencia: Cargar Proyecto



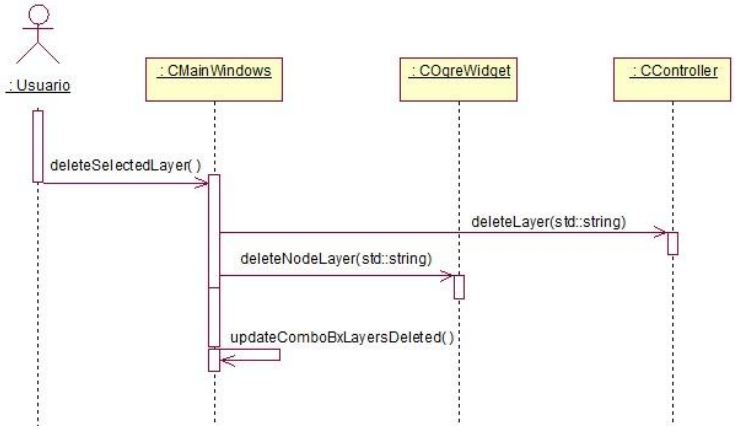
4.4.3 Diagrama de Secuencia: Salvar Proyecto



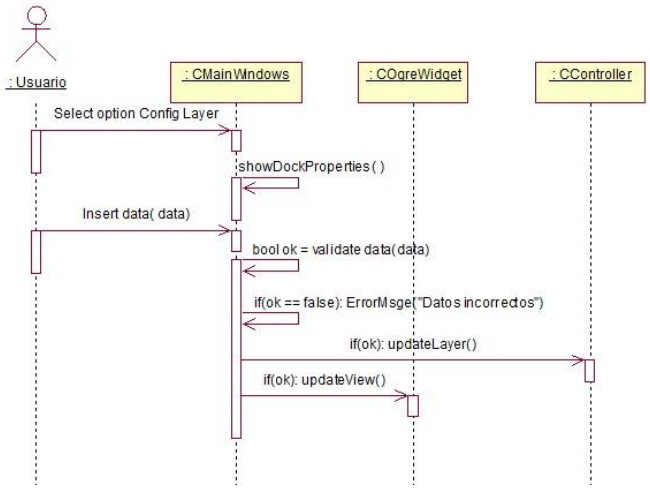
4.4.4 Diagrama de Secuencia: Insertar Capa



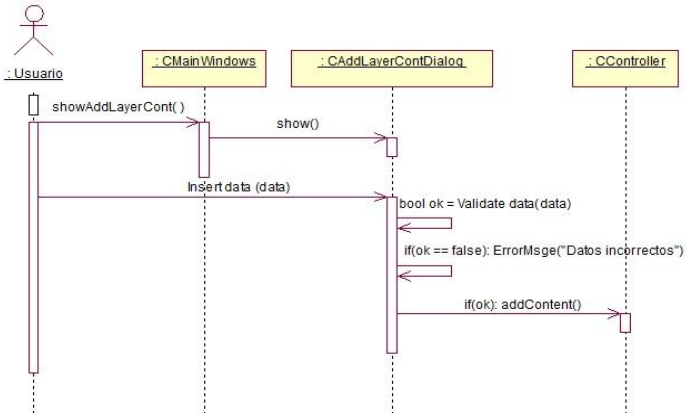
4.4.5 Diagrama de Secuencia: Eliminar Capa



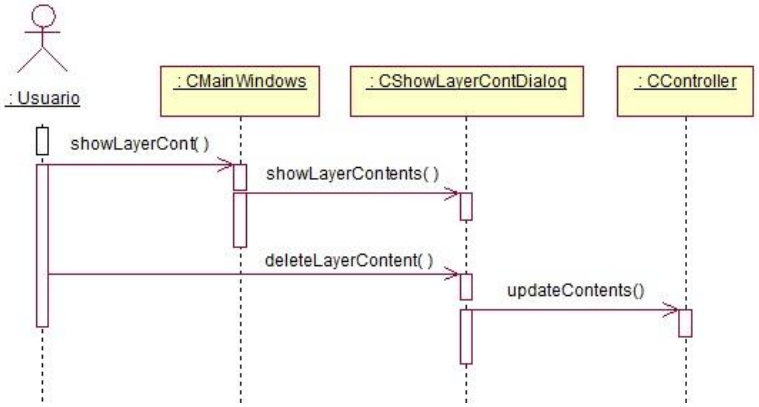
4.4.6 Diagrama de Secuencia: Configurar Capa



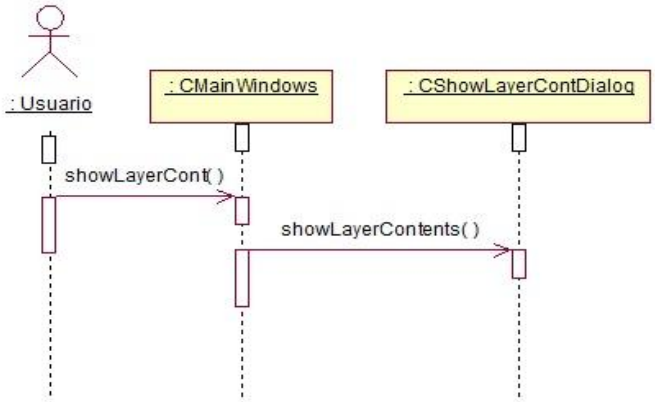
4.4.7 Diagrama de Secuencia: Insertar Contenido Capa



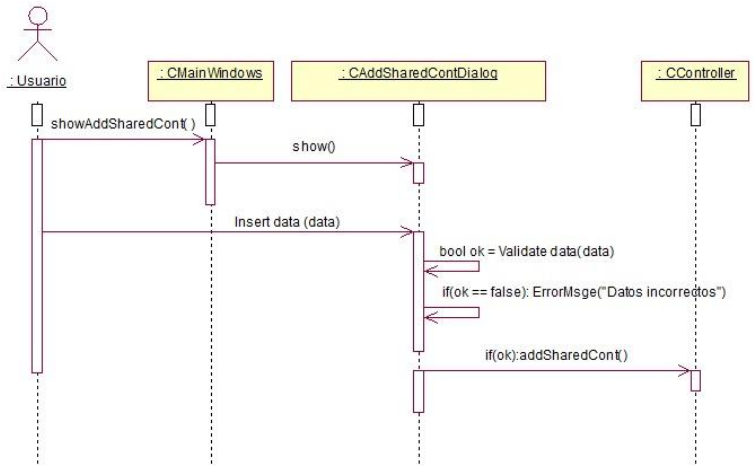
4.4.8 Diagrama de Secuencia: Eliminar Contenido Capa



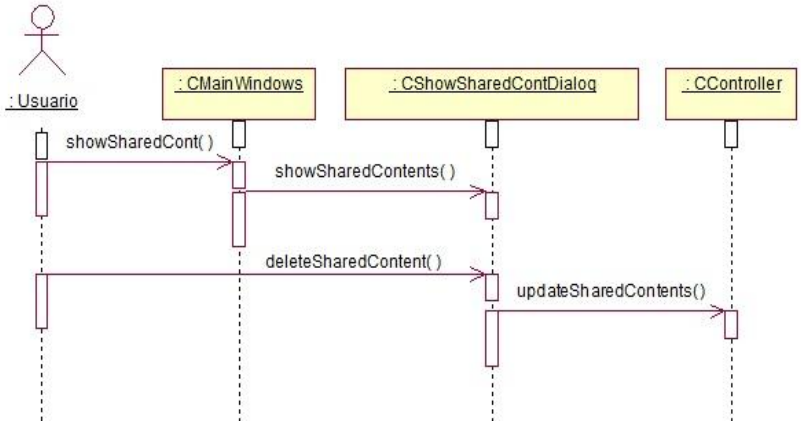
4.4.9 Diagrama de Secuencia: Mostrar Contenido Capa



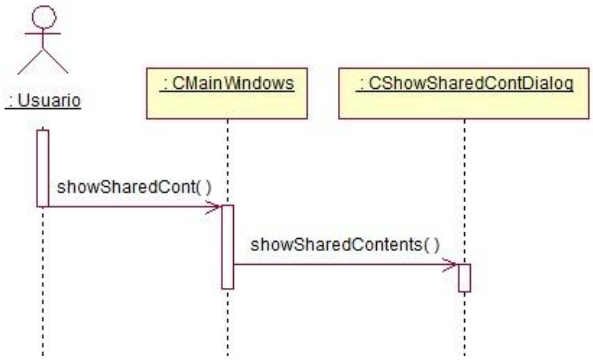
4.4.10 Diagrama de Secuencia: Insertar Contenido Compartido



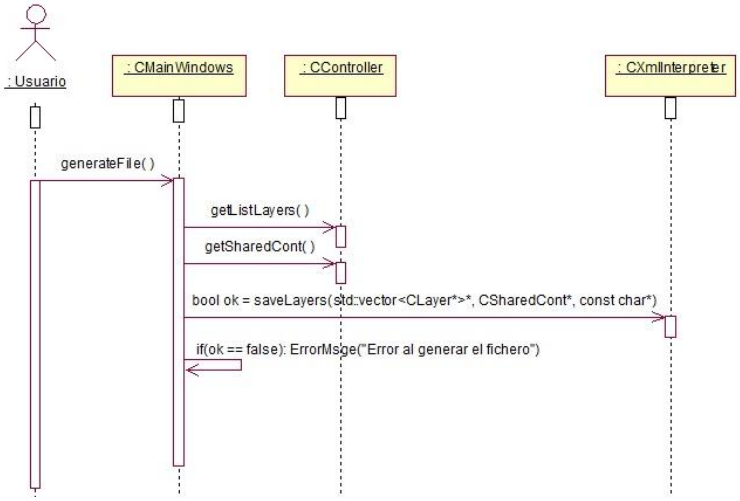
4.4.11 Diagrama de Secuencia: Eliminar Contenido Compartido



4.4.12 Diagrama de Secuencia: Mostar Contenido Compartido



4.4.13 Diagrama de Secuencia: Generar Fichero de Configuración



4.5 Estándares de codificación

El código de la Herramienta sigue algunos estándares propuestos por los autores, respetando los estándares de codificación para C++. El código está escrito en inglés, debido a que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático. El conocimiento de los estándares seguidos para el desarrollo de la misma permitirá un mayor entendimiento del código, y es una exigencia de los autores que cualquier modificación que se realice debe estar codificada siguiendo estos estándares.

Nombre de los ficheros:

Los nombres de los ficheros .h y .cpp se escriben con letra inicial mayúscula, en caso de tener un nombre conformado por más de una palabra estas se escribirán una a continuación de la otra comenzando siempre cada palabra con una letra mayúscula y el resto en minúscula.

Ej. OgreWidget.h

Clases:

Los nombres de las clases se escriben con letra inicial mayúscula, comenzando con la letra "C" que se utiliza para indicar que es una clase, en caso de tener un nombre conformado por más de una palabra estas se escribirán una a continuación de la otra comenzando siempre cada palabra con una letra mayúscula y el resto en minúscula.

Ej. class CLayer

Enumerados:

Para los enumerados se utiliza el indicador "E" en el nombre del tipo, y las constantes se escriben con letra inicial mayúsculas.

Ej. enum EStrategy {RandomOriented, RandomNonOriented};

Tipos de datos de la librería std:

Para las listas de vectores (vector) que nos proporciona la librería estándar de C++, se utiliza el indicador "v", con el sufijo List.

Ej. std::vector<CLayer*> vLayersList;

Declaración de variables:

Los nombres de las variables comienzan con un identificador del tipo de dato al que correspondan y a continuación la palabra o palabras que conformen el nombre de la variable, comenzando cada una con letra mayúscula y el resto con minúsculas. En el caso de que sean variables miembros de una clase, se le antepone el identificador "m_", y en caso de ser argumentos de algún método, se les antepone el prefijo "arg_".

Tipos Simples:

Ej. bool bVarName;

bool m_bVarName; // variable miembro de una clase

Instancias de tipos creados:

Ej. CClassName kObjectName; // objeto de una clase

CClassName* pkName; // puntero a objeto

CClassName* m_pkName; // variable miembro de clase

Métodos:

Los nombres de los métodos comenzarán con letra inicial minúscula, en caso de que el nombre este compuesto por más de una palabra la primera palabra será escrita en minúscula completamente y el resto comenzará con letra mayúscula Los constructores y destructores, como lo exigen los compiladores, llevan el nombre de la clase.

Ej. void deleteLayer(std::string arg_sLayer); //Método

CLayer(); //Constructor ~CLayer(); //Destructor

Métodos de acceso a miembros:

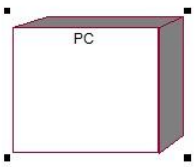
Los métodos de acceso a los miembros de las clases comienzan con el identificador “get” seguido de la palabra o palabras que conformen el nombre de la variable a que hacen referencia.

Ej. bool m_bFollowCamera; // variable

bool& getFollowCamera(); // “Get” y “Set”

4.6 Diagrama de despliegue

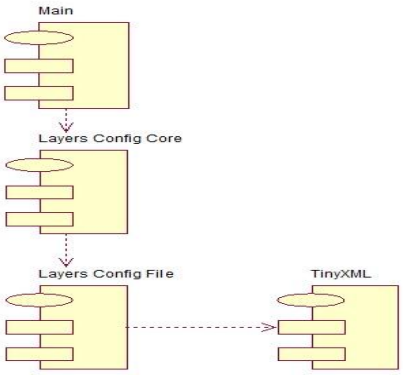
Dado que el sistema es una aplicación desktop no distribuida el diagrama de despliegue se representa con un único nodo.



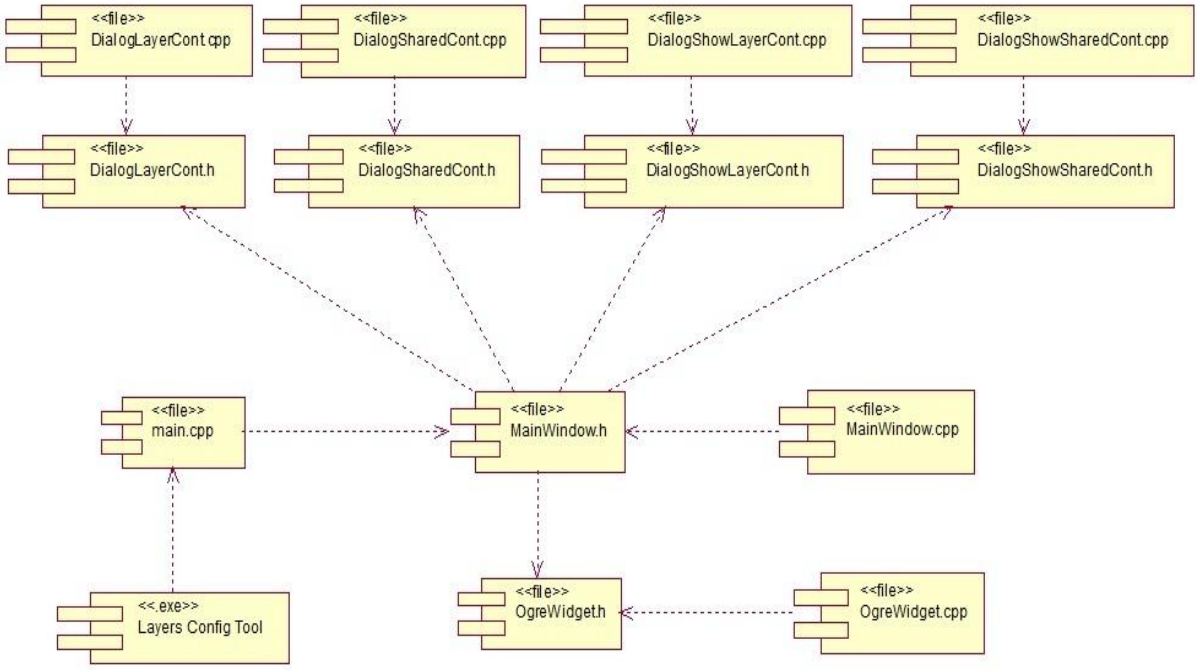
4.7 Diagramas de componentes

El diagrama de componentes describe cómo se implementan los elementos del modelo de diseño en términos de archivos de código fuente y ejecutables. Para lograr un mejor entendimiento, el diagrama de componentes del presente trabajo se ha organizado en paquetes.

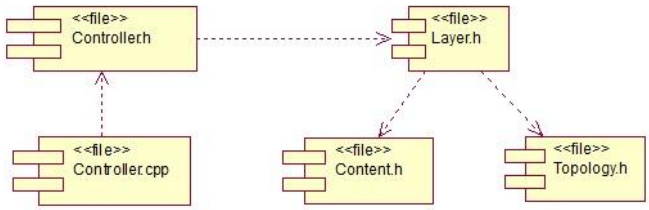
4.7.1 Diagrama de paquetes de componentes



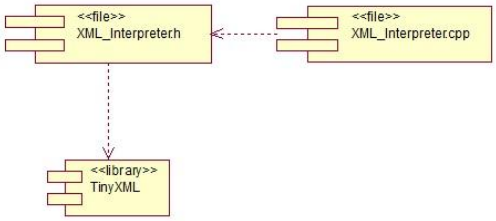
4.7.2 Diagrama de componentes, paquete Main



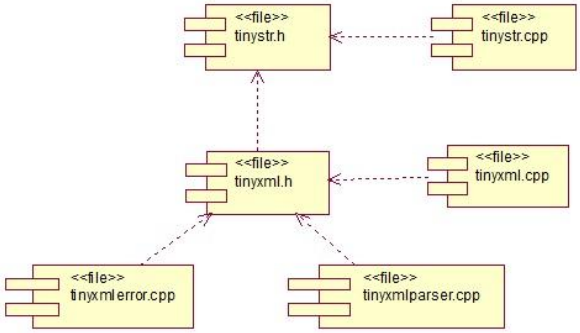
4.7.3 Diagrama de componentes, paquete Layers Config Core



4.7.4 Diagrama de componentes, paquete Layers Config File



4.7.5 Diagrama de componentes, paquete TinyXML



Conclusiones del capítulo

Al culminar el capítulo se tiene concebido detalladamente el diseño del sistema así como el modelo de implementación del mismo. Con esto queda todo listo para pasar a la programación de los casos de uso correspondientes al primer ciclo de desarrollo.

CONCLUSIONES

Se puede concluir al término de la presente investigación, que el objetivo general planteado ha sido cumplido, obteniéndose los siguientes resultados:

- ✓ Se obtuvo una herramienta que brinda la posibilidad de configurar las capas y contenidos de un entorno 3D y genere el fichero de configuración que será utilizado por el generador de entornos 3D basado en capas del proyecto Juegos-CNeuro.

- ✓ La herramienta brinda además otras funcionalidades auxiliares como crear, salvar y cargar un proyecto, con el objetivo de que el trabajo con esta sea lo más sencillo y agradable posible.

- ✓ Se obtuvo además un fichero de configuración, con una estructura propia y flexible, que permitirá almacenar la configuración de las capas y contenidos de una escena 3D.

- ✓ Con el uso de la herramienta desarrollada se garantiza el incremento de la productividad de los desarrolladores, disminuyendo enormemente el tiempo necesario para configurar las capas y generar el fichero de configuración del entorno 3D que se esté desarrollando.

RECOMENDACIONES

Como trabajos futuros se proponen las siguientes mejoras a la herramienta desarrollada:

- ✓ Adicionarle nuevas topologías de capas y estrategias de generación según lo necesiten los generadores.
- ✓ Incorporar soporte para más idiomas haciendo uso de Qt Linguist.
- ✓ Migrar la herramienta en una versión futura hacia la plataforma Linux.
- ✓ Incorporar un manual de usuario para una mayor comprensión del funcionamiento de la aplicación.

BIBLIOGRAFÍA CONSULTADA:

Blanchette, Jasmin y Summerfield, Mark. C++ GUI Programming with Qt 4. Massachusetts : Trolltech AS, 2006. ISBN 0-13-187249-4.

de la Rosa Rodríguez, Argelio. Herramienta de creación y edición de mapas de sonidos para aplicaciones de realidad virtual. Ciudad de La Habana: s.n., 2009.

Pérez Santiesteban, Diana Rosa. Herramienta de Configuración de Niveles para el Paquete de Juegos CNEURO. Ciudad de la Habana: s.n., 2009.

QtSoftware. Qt-4.6-whitepaper. [En línea] http://qt.nokia.com/files/pdf/qt-4.6-whitepaper/view

Ogre3D. Wiki. [En línea] http://www.ogre3d.org/wiki/index.php/Main_Page

TinyXML. Documentation. [En línea] http://www.grinninglizard.com/tinyxmldocs/index.html

REFERENCIAS BIBLIOGÁRFICAS:

[1]. **Galeano G., J.B**. "La realidad virtual". 2010. Disponible en: http://www.monografias.com/trabajos4/realvirtual/realvirtual.shtml

[2]. **Omar Correa Madrigal**, "Generación de Entornos Virtuales en Tiempo Real, una Solución Eficiente y Ajustable para el Tratamiento de Fobias", 2008.

[3]. **Greuter, Stefan y Nigel Stewart**. Beyond the horizon. Computer generated, three-dimensional, infinite virtual worlds without repetition in realtime. 2004.

[4]. **Moeller, Akenine y Eric Haines**. Real-Time Rendering. 2002.

[5]. **Adams, David**. Automatic Generation of Dungeons for Computer Games. 2002. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.1445&rep=rep1&type=pdf

[6]. **Noe Sierra Romero y Sergio V. Chapa Vergara.** Diseño de Interfaces Visuales. Disponible en: http://www.cs.cinvestav.mx/CursoVis/ContenidoVis.html

[7]. **Carlos Aimacaña Toledo**. Interfaz de Usuario. 2000 mitaly_ceat@latinmail.com

[8]. **Glade**. 2010. Disponible en: http://glade.gnome.org

[9]. **The GTK+ Project**. 2010. Disponible en: http://www.gtk.org/index.php

[10]. **QtSoftware**. 2010. Disponible en: http://qt.nokia.com/products

[11]. **QtSoftware**. 2010. Disponible en: http://qt.nokia.com/products/developer-tools/developer-tools

[12]. **Jasmin Blanchette y Mark Summerfield**. C++ GUI Programming with Qt 4. 2006.

[13]. **OpenSceneGraph**. 2010. Disponible en: http://www.openscenegraph.org

[14]. **VR Juggler**. The Programmer's Guide. 2010. Disponible en: http://www.vrjuggler.org

[15]. **Introducing OpenProducer**.2010.Disponible en: http://www.andesengineering.com/Producer

[16]. **OpenSceneGraph**. Introduction. 2010. Disponible en: http://www.openscenegraph.org/projects/osg/wiki/About/Introduction

[17]. **Crystal Space**. 2010. Disponible en: http://www.crystalspace3d.org/main/Main_Page.

[18]. **Crystal Space**. 2010. Disponible en: http://www.crystalspace3d.org/main/Features.

[19]. **Ogre3D**. 2010. Disponible en: http://www.ogre3d.org/about

[20]. **Ogre3D**. Features. 2010. Disponible en: http://www.ogre3d.org/about/features

[21]. **The World Wide Web Consortium (W3C).**2010.Disponible en: http://www.w3.org/standards/xml/core

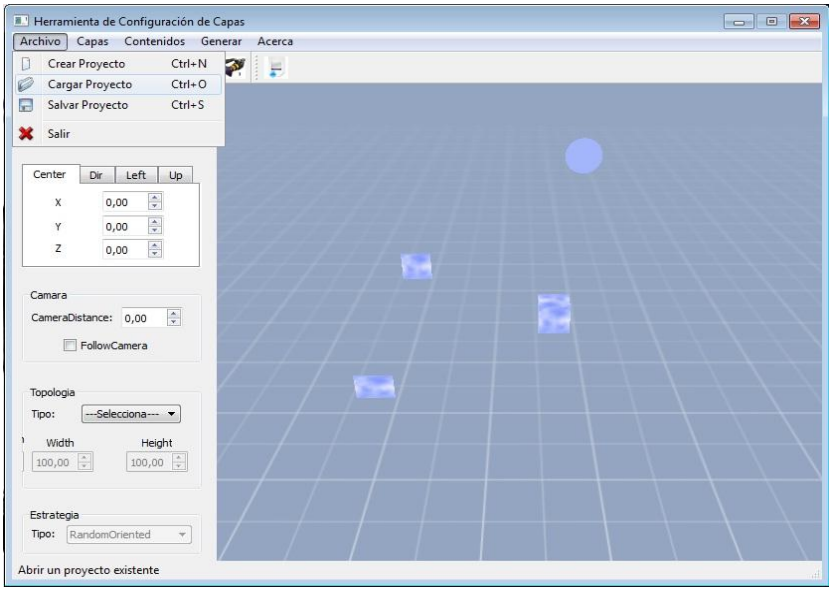
[22]. **Jaime E. Villate**. Introducción al XML. Universidad de Oporto villate@fe.up.pt 2001.

[23]. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James**. El Proceso Unificado de Desarrollo de Software Volumen I. La Habana: Félix Varela, 2004.

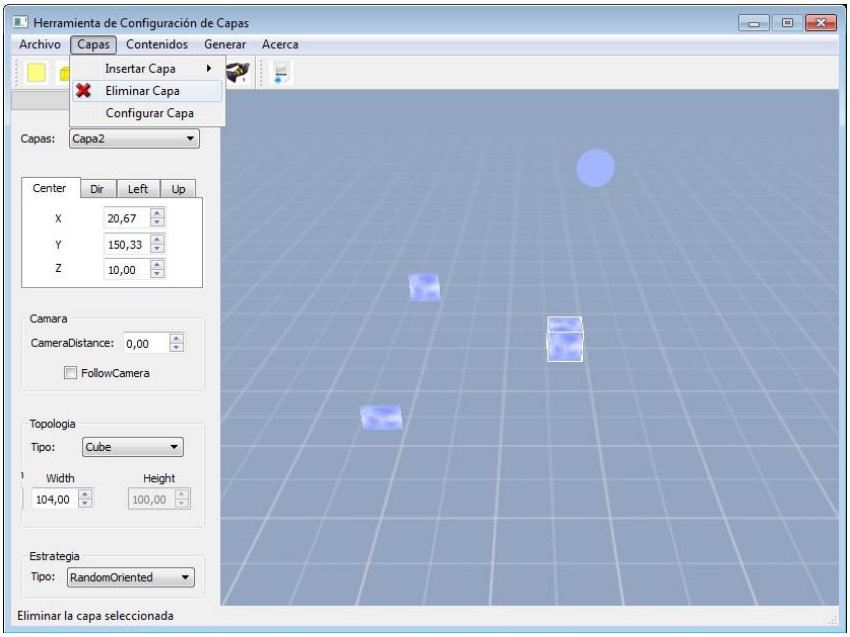
[24]. **Larman, Craig**. UML y Patrones.1999.

ANEXOS

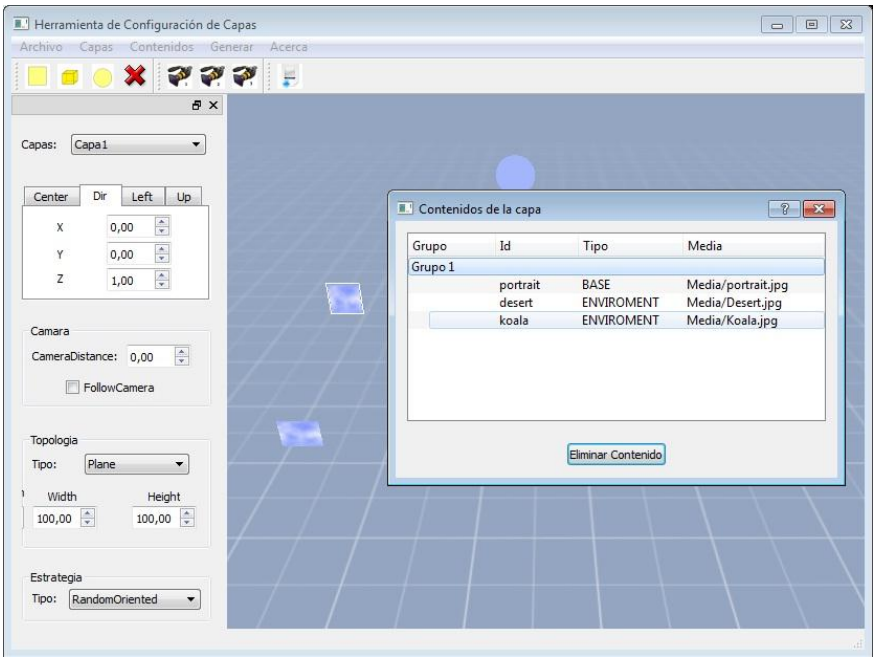
Anexo 1.



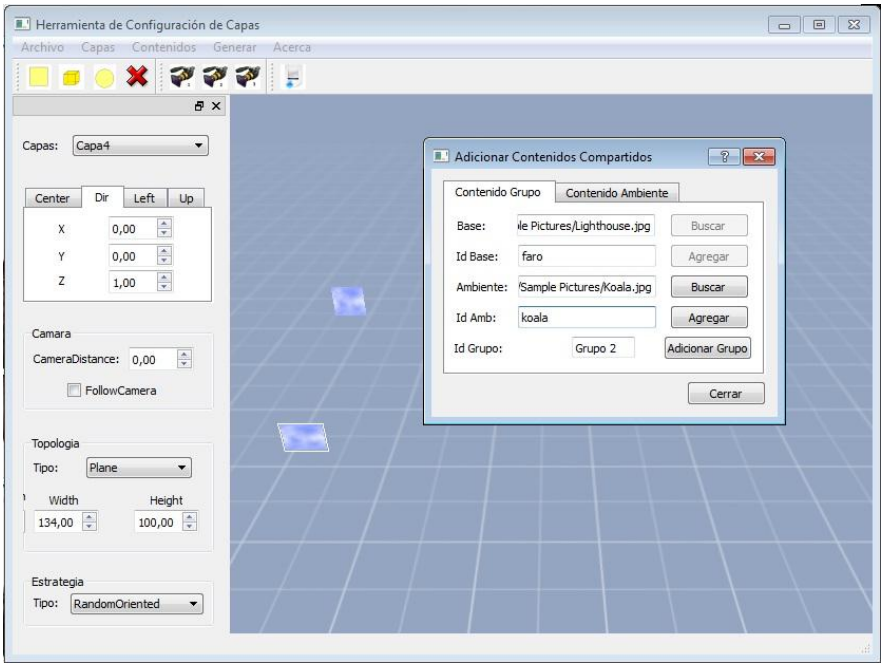
Anexo 2.



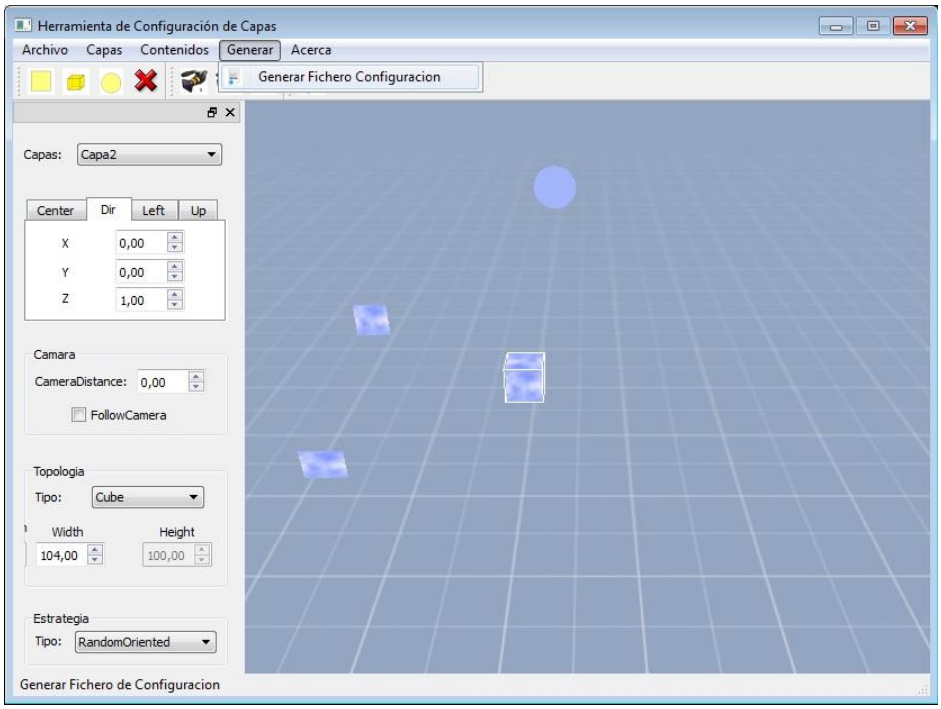
Anexo 3.



Anexo 4.



Anexo 5.


Herramienta de Configuración de Capas
Archivo
Capas
Contenidos
Generar
Acerca
Generar Fichero Configuracion
Capas:
Capa2
Center
Dir
Left
Up
X
0,00
Y
0,00
Z
1,00
Camara
CameraDistance:
0,00
FollowCamera
Topologia
Tipo:
Cube
Width
Height
104,00
100,00
Estrategia
Tipo:
RandomOriented
Generar Fichero de Configuracion

GLOSARIO DE TÉRMINOS:

Entornos virtuales: Es la simulación de mundos o entornos, denominados virtuales, en los que el hombre interacciona con la máquina en entornos artificiales semejantes a la vida real.

Motor Gráfico: Un motor gráfico (*GraphicEngine*) es un término que hace referencia a un conjunto de clases que posibilitan el proceso de render y un conjunto de técnicas para facilitar el manejo de datos.

Parser: Programa o módulo encargado del proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical en un texto.

Realidad virtual: Se define como realidad virtual un simulacro de entornos sintéticos en tiempo real, es la representación de la realidad a través de medios electrónicos, se trata de una realidad ilusoria, que existe sólo dentro de los ordenadores o medios electrónicos. De esta forma se puede definir la realidad virtual como: "La simulación de medios ambientes y de los mecanismos sensoriales del hombre por computadora, de tal manera que se busca proporcionar al usuario la sensación de inmersión y la capacidad de interacción con medios ambientes artificiales".

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

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

Widget: Símbolo gráfico de interface que permite interacción entre el usuario y el ordenador.

1