UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Propuesta de herramienta para el desarrollo de Paseos Virtuales.

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

Autores:

Yoan Sánchez Gómez

Genry Tamayo Sánchez

Tutor: Minardo Gollún González López

Co-tutor: Yessy Cedeño González

Junio 2009

DATOS DE CONTACTO

Ing. Minardo Gollún González López

Graduado de Ingeniería Informática en la Universidad de las Ciencias Informáticas. Actualmente es profesor instructor en dicha universidad. Imparte las asignaturas Introducción a la Programación y Programación 1 y es Jefe del Proyecto “Paseo Virtual”, perteneciente al Polo de Realidad Virtual de la Facultad 5.

E-Mail: mgonzalezl@uci.cu

Ing. Yessy Cedeño González

Graduado de Ingeniería Informática en la Universidad de las Ciencias Informáticas. Actualmente es profesor instructor en dicha universidad. Imparte las asignaturas de Ingles I y II respectivamente.

E-Mail: ycgonzalez@uci.cu

AGRADECIMIENTOS

Hoy, cuando ya nuestros días de estudiantes culminan para quedar formados como profesionales en el campo de las ciencias informáticas, quisiéramos hacer llegar un sincero agradecimiento a nuestro Comandante en Jefe Fidel Castro Ruz por crear esta universidad y darnos la oportunidad de ser participes de uno de sus sueños: de ser hombres de futuro. Especialmente quisiéramos agradecer al Ing. Yessy Cedeño González por su disposición a brindarnos parte de su tiempo y conocimiento, y a nuestro tutor el Ing. Minardo Gollun González López por tener paciencia y apoyarnos en todo.

A mi madre, por tener fe en mí, por el apoyo y amor que siempre me ha dado, a mi hermana que ha hecho que mi estancia en la universidad sea más colorida, a mi familia en general. A mis amigos en la UCI, a mis compañeros de aula que han compartido todos los momentos de estos años.

Yoan

A los profesores, amigos, compañeros y todo aquel que colaboró en mi formación como profesional en el campo de las ciencias informáticas. A mis padres por haber confiado en mí y haberme apoyado tanto en estos 5 años de estudio. A mi novia por ser alguien tan importante en mi vida.

Genry

DEDICATORIA

A mis padres, a mí hermana y a toda mi familia.

Yoan

A mis padres, mi hermana y a mi tía Taty.

A mi familia por quererme tanto.

A mi novia Mercedes.

Genry

RESUMEN

La creación y modificación de modelos 3D, de grafos de caminos y montaje de un entorno virtual, son la base para presentar un paseo virtual. Los paseos virtuales permiten la interactividad con los usuarios. Entre las opciones más usuales, se encuentran el posicionamiento directo en ubicaciones concretas del lugar, visitas guiadas, o bien, la aparición de paneles de información. Estas aplicaciones se agrupan en herramientas que permiten el montaje de paseos virtuales, las cuales brindan además funcionalidades de configuración de los escenarios virtuales, de los menús de navegación entre otras. Por ello este trabajo tiene como base diseñar e implementar una herramienta gráfica para el montaje y configuración de los modelos 3D para paseos virtuales, utilizando para ello el framework QT y la biblioteca SceneToolKit.

Para lograr nuestra meta se proponen llevar a cabo las siguientes tareas: analizar las características básicas de las técnicas de montaje de entornos virtuales y formato de ficheros para almacenar la información de formato 3D para la realización de recorridos virtuales; estudiar el funcionamiento de la biblioteca gráfica SceneToolKit para su aplicación en los montajes y configuración de los modelos 3D. También se hará una descripción de las características y dinámica del sistema.

Palabras clave:

Entorno virtual, grafos de caminos, menús de navegación, modelos 3D, paseo virtual, realidad virtual, simuladores.

ABSTRACT

The creation and modification of 3D models, path of grafts and assembly of a virtual environment, is the base in order to introduce a virtual tour. The virtual tour allows the interactivity with the users. Between the most usual options, it encounters the direct position to concrete locations of the place, guided visits, or well, the apparition of information panels. These applications are group in to tools that let the assembly of virtual tour, which additionally offer functions of configuration the virtual environment, of the navigation menu, between other. For that reason this work have like a goal design and implementation of a graphical tool to assembly and configuration of 3D models for virtual tours, utilizing the QT framework and the library SceneToolKit for it.

In order to achieve our goal we are proposed to carry out the next tasks: analyzing the basic characteristics of the assembly techniques of virtual environments and files format in order to archive information of 3D format for the realization of virtual travels; study the fundamental operation of the graphic library SceneToolKit for their application in the assemblies and configuration of the 3D models. We will also become a description of the characteristics and dynamic of the system.

Keywords:

3D models, navigation menu, path of grafts, simulators, virtual environments, virtual reality, virtual tour.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La realidad virtual en la actualidad se ha convertido en un apoyo vital a casi todos los procesos, por ejemplo: en la industria de generación eléctrica; en los experimentos físico-nucleares; la misma www(world wide web); etc., se cuenta con ella porque ahorra recursos y tiempo, porque al simular una de estas actividades, ya sea: el de seguimiento de temperatura de una caldera; la fusión atómica; etc., se puede llegar a los resultados reales en un ambiente virtual sin poner en peligro vidas humanas. Debido a las múltiples ventajas se considera que por sus aportes a ramas tan diversas como la medicina, la educación, las ingenierías, la arquitectura y el ocio, es uno de los medios más importantes que existe para el trabajo en entornos 3D.

Dentro de las aplicaciones que se pueden realizar con los entornos virtuales están los paseos virtuales que son una simulación de la realidad, que puede ser a través de la web o de una aplicación de desktop. Permite a los usuarios experimentar una sensación de inmersión del lugar. Los paseos virtuales permiten la interactividad con los usuarios. Entre las opciones más usuales, se encuentran el posicionamiento directo en ubicaciones concretas del lugar, visitas guiadas, o bien, la aparición de paneles de información.

Las distintas aplicaciones de esta tecnología pueden ser:

- Puesta en valor de enclaves turísticos.
- Presentación en la Web de proyectos urbanísticos de futuro.
- Visitas a parques empresariales.
- Promoción de casas rurales o entornos similares.

En Cuba, existe un incipiente desarrollo de herramientas con el fin de configurar entornos 3D para paseos virtuales, en la UCI se cuenta con el Polo de Realidad Virtual, en la Facultad 5, en el cual existe un proyecto dedicado al desarrollo de un motor gráfico 3D (SceneToolKit) para el uso de este en los otros proyectos productivos. En el Polo también existe el proyecto Paseos Virtuales el cual se encarga del diseño y montaje de los modelos 3D en el entorno del paseo. El principal reto es el montaje del entorno, pues no

se cuenta con una herramienta gráfica que permita el montaje de los modelos, la configuración de menús de navegación y de los principales modelos, así como de definir el modo del paseo (dirigido o libre). Estas tareas se realizan parcialmente en los mismos programas de diseño gráfico como el 3D Max Studio y el Blender pero no suplen todas las necesidades del grupo de desarrollo por estar limitados a las funciones de cámaras y de montajes, además de que el primero es un software propietario que acarrearía la obligación del pago de la licencia para poder comercializar el producto final.

Ante estos inconvenientes se plantea la siguiente interrogante como **problema científico:** ¿Cómo realizar el montaje y configuración de los modelos 3D para paseos virtuales?

Para esto se plantea como **objeto de estudio** las aplicaciones de los motores gráficos, para cargar y configurar los modelos 3D.

Campo de acción: Aplicaciones de los Motores gráficos, para cargar los entornos y configurar los modelos 3D para paseos virtuales.

En el presente trabajo de diploma se propone como **objetivo:** Diseñar una herramienta gráfica para el montaje y configuración de los modelos 3D para paseos virtuales.

Idea a defender: El diseño de la herramienta gráfica permitirá la elaboración de paseos virtuales dinámicos que satisfacen las necesidades de interacción usuario-entorno.

Se plantean entonces un grupo de **tareas** que permitirán cumplir con los objetivos, y que se resumen en las siguientes:

- Analizar las características de las herramientas para el montaje de paseos virtuales que están en el mercado actual para decidir que variante escoger (web/escritorio).
- Analizar las características básicas de las técnicas de montaje de entornos virtuales y formato de ficheros para almacenar la información de formato 3D para la realización de recorridos virtuales.

- Estudiar el funcionamiento de la biblioteca gráfica SceneToolKit 2.4 para su aplicación en los montajes y configuración de los modelos 3D.
- Realizar el análisis y diseño del software a través de la metodología seleccionada para crear la base de la implementación.
- Desarrollar el módulo de grafo de caminos para posibilitar el movimiento dentro del paseo de forma dirigida.

Como resultado de este trabajo se pretende que obtener un prototipo funcional de la Herramienta gráfica VT-STK, para el montaje de entornos 3D para paseos virtuales –con funcionalidades tales como: la configuración de menús de navegación del paseo, de cargar los principales modelos con condiciones especiales para ellos, así como de definir el modo del paseo (dirigido o libre) – utilizando la biblioteca STK (SceneToolKit).

El contenido de este trabajo se encuentra estructurado de la siguiente manera:

En su primer capítulo “Fundamentación Teórica”, se hace un análisis bibliográfico donde se realiza un estudio de algunas herramientas para desarrollar paseos virtuales, se detallan las características principales de la biblioteca SceneToolKit así como del framework multiplataforma QT.

En el capítulo 2, “Soluciones Técnicas”, se exponen las características técnicas que presentará el sistema como solución a los problemas planteados.

En el capítulo 3, “Características del Sistema”, se crea el modelo del dominio, se hace la captura de requisitos y se crean los modelos de casos de uso del sistema.

En el capítulo 4, “Análisis y Diseño del Sistema”, se presentan los diagramas de clases del análisis, los diagramas de clases del diseño, los diagramas de secuencia, se muestran los diagramas de despliegue y de componentes. Finalmente, se ofrece un glosario de términos para ayudar a la comprensión del lenguaje técnico utilizado a lo largo del trabajo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se hará referencia nuestro campo de acción, la visualización en tiempo real y visualización 3D en tiempo real, las principales herramientas utilizadas actualmente para el desarrollo de paseos virtuales, motores gráficos y sus características, herramientas de desarrollo, el framework QT, la biblioteca STK (SceneToolKit), así como alguna de las metodologías a utilizar para el desarrollo de esta labor.

1.1 Visualización.

1.1.1 Visualización en tiempo real.

Implica generar al menos de 20 a 30 imágenes por segundo. Permite la interacción con el usuario. [1]

1.1.2 Visualización 3D en tiempo real.

Permite el desarrollo de nuevos tipos de aplicaciones y ofrece muchas ventajas:
Mayor realismo, mejor visualización, etc.
Campos de aplicación:
• Videojuegos: mercado de gran importancia en el sector del entretenimiento
• Simulación y entrenamiento.
• Educación o divulgación.
• Visualización científica: por ejemplo en la medicina.

Requisitos de una herramienta avanzada para el desarrollo de aplicaciones de visualización 3D en tiempo real.

• Generales: diseño orientado a objetos, documentación, etc.
• Scripting
• Editores de escenas
• Física: detección de colisiones, física de cuerpos rígidos, etc.
• Iluminación avanzada: iluminación por píxel, etc.

• Shaders: por vértice, por píxel, etc.
• Gestión de escenas: niveles de detalle, culling, etc.
• Animación: animación por claves, animación de esqueletos, deformación de mallas.
• Visualización de terreno: CLOD, etc.
• Efectos de post-procesado: bloom, noche, sepia, etc.
• Soporte de Audio y Video.
• Sistema de Red.
• Inteligencia Artificial. [2]

1.2. Herramientas

1.2.1 Adobe® Atmosphere™

Adobe Atmosphere es una herramienta de creación profesional para el ensamblaje y creación de decorados interactivos tridimensionales. Este nuevo tipo de multimedia incrustado ofrece a la Web o al diseñador de documentos la posibilidad de mostrar una gran variedad de contenido interactivo, tales como objetos 3D, sonido, transferencia de audio y vídeo, animaciones de Macromedia® Flash™ (SWF) y comportamientos físicos, todo en el contexto de una representación teatral en directo.

Adobe Atmosphere permite crear ricos mundos 3D que podrán ser visitados On-line por otras personas y éstas podrán interactuar entre ellas en tiempo real.

Una vez creado un mundo 3D o te unas a uno ya creado, podrás moverte con total libertad por el mismo, chateando con todos los visitantes virtuales de esa creación.

Interfaz de Adobe Atmosphere

Este software presenta un ambiente de trabajo similar al del resto de las aplicaciones de Adobe, una característica que agradecen los usuarios de la empresa mencionada.

Área de desarrollo: se muestra sencilla, y bastante limitada por tratarse de una edición beta, es posible generar increíbles entornos 3D, implementando paredes, pisos, y muchos efectos visuales como niebla, animaciones, e interactividad.

Herramienta Adobe Atmosphere

Hasta ahora, las funciones del programa resultan interesantes, aunque todavía están en evolución. Las operaciones más complejas, que se encargan de crear efectos únicos, e interactividad a nuestro mundo 3D, están hechas bajo Java Script, aunque los códigos no los trae consigo, no incluye efectos preestablecidos para programadores inexpertos. Se tiene como perspectiva en un futuro disfrutar de una lista de comandos en la versión final, para facilitar la creación de escenarios complejos.

El modelo virtual puede gozar de bellos paisajes. Uno puede insertar una imagen, y adaptarla a un objeto para que se vea lo más realista posible.

Uno de los principales aspectos del programa, es el llamado Avator, el cual posibilita el desarrollo de chats interactivos comunicando a los usuarios vía texto de una manera extraordinaria. [3]

1.2.2 VRML (*Virtual Reality Modeling Language*)

VRML es un lenguaje para el desarrollo de aplicaciones de realidad virtual inmersiva, en forma de mundos virtuales compuestos de un espacio, normalmente tridimensional, donde los objetos son interactivos. En estos mundos virtuales el usuario podrá adentrarse, eligiendo entre varias perspectivas, e interactuar con los objetos que allí se encuentran. Esta tecnología es cada vez más accesible para el usuario medio, quien puede disponer de mejores equipos multimedia a precios asequibles, o simplemente de un ordenador y un navegador de internet.

Además es independiente de la plataforma donde se ejecute el visualizador, tiene capacidad para trabajar de un modo eficiente con conexiones lentas, y es extensible, es

decir, susceptible de ser ampliado fácilmente, y permite la creación de mundos virtuales que representen lugares geográficos remotos. [4]

1.2.3 Tradky Software

Permite crear y editar ambientes virtuales en 3 dimensiones (ferias, museos, galerías de arte, ciudades virtuales) que pueden ser visitados desde cualquier ordenador con una conexión a internet de diferentes formas, según la potencia y la conexión de cada usuario.
Es la única plataforma capaz de generar tres formas de visitar sus entornos virtuales, en 3D real, Flash y Texto con gráficos. Se monta una sola vez, pero se puede navegar en múltiples formatos.
Está diseñado para ser un centro de información donde los negocios pueden realizarse interactuando visitantes y expositores directamente, gracias a la potente red social integrada con video chat y videoconferencia. Su montaje se realiza online y es intuitivo. [5]

1.2.4 Look Twice:

Captura, Procesamiento y Almacenamiento de imágenes digitales planas, virtuales y en movimiento.- Imágenes en 360° VR Cilíndricas, Cúbicas y Esféricas en formatos QuickTime, Java y Flash - Scouting de Locaciones en 360° VR con agregado de Puntos Cardinales y Puntos GPS - Paseos Virtuales - Interactividad con Google Earth - Fotografía de Productos: Naturaleza e Industria - Diseño de objetos - Procesamiento de imágenes: Edición, corrección, fotomontajes en RAW - Diseño de Software Multimedia a medida: Bases de datos de Fotografía, Video y Casting. Sistemas de Timing para productoras de Cine y TV - Diseños especiales de monturas para equipos de fotografía y vídeo. [6]

1.2.5 PaseoExpress

PaseoExpress es la primera herramienta de desarrollo gráfico diseñado para el manejo de operaciones y su equipo. PaseoExpress permite modelar y generar scripts operacionales gráficamente. Los scripts generados pueden ser controlados directamente por la programación de su aplicación.

Esta aplicación permite:

1. Desarrollo de scripts con pocos clicks.
2. Configuración intuitiva, modo de recuperación, manejo de eventos, administración de errores y registros, ejecución de trazas.

Beneficios incluidos:

Toda la configuración es gráfica.
Tecnología propia de control y captura de errores.
Modo de recuperación automática.
Ejecución remota.
Validación paso por paso.
No es requerida la compilación. [7]

1.2.6 AutoARQ Paisajismo

Es la nueva y evolucionada herramienta para el diseño y la proyección de áreas verdes. Constituye una evolucionada aplicación que incorpora, dentro del entorno de AutoCAD, las funciones necesarias para poder desarrollar un proyecto paisajístico en todas sus fases: desde el plano técnico en 2D, hasta la generación de imágenes reales y paseos virtuales a lo largo del diseño.

Incorpora las herramientas necesarias para proyectar fácilmente planos y maquetas tridimensionales de parques y jardines, plazas, campos de golf, zonas comunes de edificios, etc. Trabajar con herramientas específicas permite ahorrar tiempo en el desarrollo del proyecto y simplifica procesos en los que se invierte la mayor parte del tiempo.

Como herramientas destaca una amplia base de datos con características de plantas, herramientas para generar y editar terrenos, librerías de especies vegetales en 2D y 3D, elementos estructurales (vallas, caminos, escaleras,...), sistema para definir la documentación técnica, tablas de datos exportables a Excel, nuevo sistema de riego, librerías con mobiliario urbano, nuevo módulo de renderizado. [8]

1.2.7 QuickTime VR

QuickTime VR es una tecnología basada en fotografías o imágenes planas y 3D; y su ambiente natural es el terreno de las producciones multimedia en CD-ROM, que permitirá a los desarrolladores utilizarla en un amplio abanico de títulos. Su enorme versatilidad permite acercarse a una percepción de la realidad, más próxima a una visión cotidiana, es decir la forma en que funcionan los sentidos, y sentir como si realmente se estuviera navegando dentro de la propia película.

Una de sus posibilidades más prometedoras es la de utilizar imágenes del mundo real (a partir de fotografías de alta calidad, lo que constituye una ventaja sobre películas basadas en vídeo, además de poder alterar la secuencia del visionado) o imágenes generadas mediante aplicaciones de 3D.

Por otro lado, aunque resulte hasta cierto punto paradójico, el kit de desarrollo para generar películas con un alto nivel de interactividad resulta excesivamente austero y poco intuitivo para un usuario. La interfaz de usuario no existe, tal y como estamos habituados en el entorno Macintosh, y por el momento el software está claramente dirigido a desarrolladores.

QuickTime VR no requiere un hardware adicional, ya que es una extensión de la propia arquitectura QuickTime, aunque deberá tener instalado varios programas, entre ellos el MPW que consiste en un conjunto de herramientas para desarrollar software en C, C++ o lenguaje ensamblador para procesadores 680x0 y PowerPC.

Lo más básico que se puede hacer con QuickTime VR es la generación de una película navegable, permite añadir más interacción a dicha película para que el usuario pueda " coger " con su mano alguno de los elementos que hay sobre la película; saltar de una zona a otra, o simplemente desea integrarla en un interactivo realizado con Director.

El programa nos proporciona una serie de plantillas que ejecutan todas las funciones más comunes para realizar la película VR. Sin embargo, estas sirven para ejecutar órdenes

preestablecidas de antemano (que hacen referencia a un disco que simula un Photo CD y que está incluido en el kit de desarrollo), por lo que, para poder utilizarlos, debemos editar los scripts, determinando los archivos de origen, destino y otros parámetros que afecten al ángulo de visión de la lente que originó los fotogramas, así como también el ángulo de rotación de las fotografías

Para conseguir cada una de estas propiedades, el kit de desarrollo contiene herramientas que le permitirán definir puntos calientes (aquellos puntos que, una vez seleccionados, permiten ejecutar una acción determinada sobre el objeto); saltos no lineales entre diversos puntos de la película; y, por último, rutinas externas para la inclusión de películas VR en otros proyectos multimedia. [9]

1.2.8 Tourweaver 4.00

Tourweaver es un programa especialmente desarrollado para diseñar e implementar entornos panorámicos en Flash. Tourweaver crea visitas virtuales con componentes tales como la ventana emergente, activa plan de punto, Radar y enlace URL. Tourweaver se utiliza para crear visitas virtuales de bienes raíces, hotelería y turismo. [10]

1.2.9 Panoweaver 5.00

El software de edición, montaje y publicación de imágenes panorámicas líder en el mercado, con Panoweaver montará fácilmente panorámicas de alta calidad en tan sólo 3 minutos. Panoweaver HDR también crea panoramas, y las exportaciones de pantalla completa Flash / películas QTVR.

1.2.10 Kit Studio 2008

KitStudio 2008 es quizás el más completo software para la construcción de panoramas y kit para visitas virtuales en el mercado. Integra las últimas técnicas de vistas panorámicas de 360 º de software como: Easypano, Panoweaver 5.00 orgánico y Tourweaver 4.00 Professional. Con este kit profesional de vistas panorámicas para visitas virtuales, se

puede crear vistas 360 º de inmobiliaria, visitas virtuales a hoteles, restaurantes, museos y visitas virtuales guiadas para muchos otros campos. [11]

1.3. Motores Gráficos.

Framework, toolkit, scenegraph o Motor Gráfico:

– Proporcionan funcionalidades de alto nivel: visualización, carga de modelos 3D, etc.

– Nos permite manejar una escena 3D y visualizarla. [2]

Framework: entorno de trabajo que proporciona funciones de alto nivel para el desarrollo de aplicaciones.

– XNA.

• Desarrollado por Microsoft.

• Plataformas: Windows, Xbox.

• API: DirectX.

• Lenguaje: C#.

• Gratuito, no software libre.

• Empiezan a aparecer proyectos software libre sobre XNA.

– Ej.: Raven XNA Game Engine.

• Otros proyectos:

– Blade3D (proyecto muy nuevo, licencia indefinida). [2]

Scenegraph

– Que es un scenegraph?

• Estructura de datos jerárquica y reusable.

• Describe los objetos de una escena 3D y sus relaciones.

– Tamaño, posición, orientación, propiedades, materiales.

• Se usan para visualizar escenas virtuales en motores de juegos, CAD, visualización científica, simulación, modelado, etc.

• Se centra en la representación de la escena y su visualización de forma eficiente.

• No es un motor completo, suele ser parte de un de un motor de visualización.

– Utilizados principalmente en aplicaciones científicas, simulación o como base para un motor gráfico. [2]

Algunas herramientas para el manejo de scenegraph.

– Comerciales:

• Performer (SGI).

• Vega (Multigen-Paradigm).

– Gratuitos:

• NvSG.

– Libres:

• OpenSG.

• OpenScenegraph. [2]

Que es un motor gráfico:

– Además de la gestión de escenas, incorpora otras funcionalidades como:

• Animación.

• Scripting.

• Inteligencia Artificial.

• Gestión de Audio y Vídeo.

• Sistema de Red.

• Editores.

• Sistemas físicos.

• Gestión de múltiples scenegraph.

– Se suelen utilizar para el desarrollo de juegos. [2]

1.3.1 Crystal Space.

Desarrollado por Jorrit Tyberghein, basado en renderizado en portales, BSP, Zbuffering y radiosidad. Entre todas sus principales características destacamos la gran portabilidad y la gran escalabilidad proporcionada por el sistema de plugins y, además, bajo licencia LGPL.

Crystal Space es un motor gráfico 3D gratuito de altas prestaciones desarrollado en C++. Soporta: seis grados de libertad, iluminación coloreada, sombreado, espejo, transparencia, superficies reflectantes, objetos animados (basados en frames o en esqueletos), procesado de texturas, sistemas de partículas, niebla volumétrica, transformaciones jerárquicas, etc.

Las plataformas soportadas son UNIX (Linux, y Solaris), DOS, Macintosh, Amiga, Windows, BeOS, NextStep, Rhapsody y ports OpenStep.

Las APIs soportadas son OpenGL para Windows, Linux, Beos, Macintosh y OS/2, Direct3D para Windows y Glide para Linux y Windows.

Renderizado en Sectores y Portales, BSP, Z-Buffer, Sistema de partículas, terrenos, radiosidad, C-buffer y superficies curvas y superficies reflectantes.

Respecto a las luces, soporta Lightmaps, Bumpmapping, Phong y Gouraud, luces dinámicas, multicolores y radiosidad pre-calculada para los Lightmaps.

Los formatos con los que trabaja son 3DS, MD2 (Quake 2), OBJ, POV y ASE. Trabaja las texturas con Texture mapping, Mipmapping, Texturas procedurales, dinámicas y multitextura.

Incorpora el lenguaje script Python y LUA.

Su sistema de visibilidad está basado en una combinación de portales, octrees, árboles BSP y C-buffer.

Incorpora, además, detección de colisiones basado en estructuras jerárquicas de cajas englobantes, jerarquía de objetos, mallas progresivas con LOD, Sprites 2D y 3D, Superficies de Bézier, Motor de simulación de modelaje dinámico (biblioteca física). [12]

1.3.2 OGRE.

Ogre (Object Oriented Graphics Engine) es un motor de gráficos en tres dimensiones multiplataforma. La principal ventaja de Ogre sobre otros Motores Gráficos 3D es que es un proyecto open source bajo licencia LGPL. Esto significa que su uso es gratuito y solo existen unas pocas exigencias para ello.

Flexible y orientado a objetos, funciona en múltiples entornos como Windows, Mac OS X o Linux, aparte de haber sido diseñado con la idea de hacer más sencillo el desarrollo de juegos 3D que exploten al máximo las posibilidades materiales de las tarjetas 3D, y todo esto a través de una interfaz orientada a objetos. Ofrece un sistema de partículas y de gestión de recursos muy potente, además de toda una serie funcionalidades muy interesantes.

Usa un lenguaje de descripción de materiales que permite su gestión de forma independiente al código fuente de la aplicación, brinda soporte para 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, Wings3D, 3D Studio Max, Maya, etc. [12]

1.3.3 Fly 3D.

Desarrollado por Paralelo Computaçao, basado en renderizado por árboles BSP, PVS y portales, entre sus características destacamos la escalabilidad proporcionada por su sistema de plugins.

La plataforma que soporta es Windows. Licencia libre sin coste y acceso a todo el código fuente.

El renderizado se basa en árboles BSP, PVS (Potencial Visibility Set) y Sistema de partículas.

Implementa luces estáticas y dinámicas, sombras dinámicas soft-shadows, mapas de sombras dinámicas, Lightmaps e iluminación por vértice especular y difusa para objetos dinámicos.

Incorpora textura detallada y multitextura.

Otras características son sistema de plugins (DLL's), sistema de detección de colisiones, Exportador / importador de 3Dstudio MAX, LOD para caras curvadas, Quake 3 importador (geometría, texturas y superficies curvas), Inteligencia artificial (A*optimizado) y mallas animadas. [12]

1.3.4 Unreal.

Motor gráfico del juego Unreal desarrollado por la empresa Epic MegaGames basado en una extensión del renderizado en portales conocido como Dynamic Scene Graph Technology (DSG), BSP y radiosidad. Su principal característica es la gran escalabilidad (muy modular).

Se encuentra disponible para las plataformas Linux, Windows, Macintosh, PlayStation 2 y Xbox.

Para desarrollar con este motor se tiene que adquirir una licencia (250.000$ -500.000$) que da acceso a todo el código fuente, a las herramientas y juegos.
El motor, sin embargo, puede utilizarse en términos de desarrollo de juegos sin costo adicional, por lo que para trabajos y desarrollos que hagan uso del motor gráfico sin tratar la parte de render (como son los desarrollos y proyectos de IA e interfaces) el motor es una herramienta posible.

Utiliza el sistema DSG (Dynamic Scene Graph Technology) que es una extensión natural del renderizado en portales, interpolación de mallas, radiosidad, árboles BSP, LOD, superficies curvas y superficies reflectantes.

Incorpora luces multicolores, dinámicas, Lightmaps, Raytracing y enveloped lighting. Soporta nativamente el formato DXF.
En el tema de texturas incorpora Texture mapping, mapas de sombras, mapas de niebla, textura detallada para definir objetos muy detallados, texturas procedurales, texturas en tiempo real de ondas (océanos, lava, etc.), 12 niveles de mipmapping, animación de texturas (animadas), Texturas procedurales, dinámicas y multitextura.
Trabaja con lenguaje script propio nombrado UnrealScript, que consiste en un lenguaje de scripting semi-compilado para acceder a la lógica del juego y usar el potencial del motor gráfico propietario. La idea detrás de este scripting permite trabajar en una interfaz de alto nivel para controlar los objetos en un juego.
Las ventajas de este scripting radican en la velocidad del proceso de desarrollo y en la flexibilidad y capacidad de extensión de un proyecto, teniendo como desventajas la cuestión de la velocidad al tratarse de pre-compilados.

El desarrollo de juegos y herramientas que hacen uso del motor gráfico de Unreal se lleva a cabo con UnrealED, el editor de mapas para el motor grafico de Epic Games que es utilizado en juegos como Unreal, Unreal Tournament, Rune, Deus EX. etc.

Con UnrealED es posible crear gran variedad de mapas y construir con total libertad, por ejemplo paisajes urbanos, el espacio, vehículos, agua, etc., además de implementar efectos como el sonido, música, luces, controlar la gravedad, etc.

De entre otros detalles destacamos: detección de colisiones cilíndrica, Superficies curvas con LOD, Mapas de entorno, Inteligencia Artificial avanzada (path- finding y sistema de navegación), Sistema físico adaptable, Sprites 3D, Sonido digital 3D. [12]

1.3.5 Genesis 3D.

Desarrollado por Eclipse Entertainment basado en renderizado en portales, BSP y radiosidad.

La única plataforma soportada es Windows. Bajo licencia libre y derecho a modificar el código fuente, a cambio de tener que mostrar el logotipo del motor en las aplicaciones que

lo utilicen o bien bajo licencia comercial de coste 10,000$ por título, para que no salga el logotipo de Genesis 3D y código abierto.
Las APIs soportadas son Direct3D y Glide. Renderizado por Portales, árboles BSP, radiosidad y LOD. Incorpora luces multicolores y dinámicas.
Puede trabajar nativamente con el formato de ficheros de animación de 3Dstudio MAX. Texture mapping, textura translúcida, morphing de texturas, texturas animadas y procedurales. Además destacamos Sprites 3D, sombras dinámicas, jerarquía de objetos, detección de colisiones. [12]

1.3.6 Torque (V12)

Motor gráfico utilizado en el juego Tribes 2 de Dinamix basado en renderizado en portales.

Desarrollado para las plataformas, tanto el cliente como el servidor, Windows, Mac OS 9/X y en preparación Linux, con las siguientes APIs: OpenGL para Mac y Linux y OpenGL y DirectX para Windows.

Se basa en Renderizado en sectores y portales, sistema de partículas y terrenos. En el tema de luces encontramos Lightmaps animados, luces por vértice y multipaso.

Soporta Texture mapping y mipmapping, multitexturas, textura detallada, mapas de entorno y texturas animadas. Incorpora también mallas progresivas con LOD (niveles de detalle), Plugins para scripting, detección de colisiones, sombras proyectadas de objetos (recorte contra el entorno), exportadores a 3Dstudio MAX y Milkshape y Plugins 3Dstudio MAX. [12]

1.3.7 Quake 2

Es el motor grafico del Quake 2, desarrollado por John Carmak de Id.Software. Se basa en el renderizado por árboles BSP y radiosidad.

Desarrollado para plataformas Windows, Linux y Macintosh soportando las APIs OpenGL, también existe un port para Direct3D.

Su licencia oscila entorno los 250.000$ que da acceso al código fuente.

Trabaja los renderizados con arboles BSP y radiosidad. Incorpora luces dinámicas, multicolores, Lightmaps. Implementa su lenguaje de script llamado Quake C, así como el uso de DLL's. Trabaja con su formato propio llamado MD2, muy extendido entre diferentes motores gráficos y diferentes editores. [12]

1.3.8 Irrlicht

Irrlicht es un motor 3D multiplataforma de alto rendimiento de código abierto para crear aplicaciones en tiempo real. Sus características principales son: ser fácil de utilizar, extremadamente rápido, extensible y libre de fallos.

El motor es absolutamente flexible, y es posible escribir aplicaciones de diversos usos. Algunos son: usos complejos de la simulación 3D, shooter games en primera y tercera persona, escenas de interior y/o al aire libre, juegos de estrategia en tiempo real, juegos 2D.

Todos los efectos especiales del motor de Irrlicht pueden ser utilizados sin la necesidad de estudiar su documentación por días. El programador puede cambiar e influenciar casi todo en el motor pues es fácil de utilizar.

Estabilidad extrema: La mayoría de las bibliotecas para los usos en tiempo real fallan cuando el usuario hace algo que el programador de la biblioteca no esperó. Esto es diferente en el motor de Irrlicht. Imprime una advertencia y continúa su funcionamiento.

Porciones de importadores incorporados. El motor puede cargar directamente las porciones de formatos del archivo común sin necesidad de utilizar convertidores o exportadores, aumentando tiempo de desarrollo. (3ds, md2, obj, pk3, ms3d, bsp, bmp, tga, jpg, psd)

Independencia de la API: de OpenGL y de DirectX, e incluso un dispositivo nulo. Los dispositivos pueden ser cambiados durante tiempo de pasada.

Ninguna dependencia de otras bibliotecas. [12]

1.4 Herramientas De Desarrollo.

1.4.1 Code::Block.

Es una herramienta de desarrollo libre, es un IDE (entorno de desarrollo integrado) que soporta el lenguaje C++, construido para satisfacer las más altas demandas del usuario. Está diseñado para ser extensible y completamente configurable.

Su arquitectura está basada en plugins, Code::Block puede ser extendido y se le puede agregar cualquier tipo de funcionalidad mediante instalación/codificación de dichos plugins. La compilación y la depuración de funcionalidades ya están añadidas mediante plugins. [13]

1.4.2 Rational Rose Enterprise.

Rational Rose Enterprise es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java™/J2EE™, Visual C++® y Visual Basic®. **Rational Rose**, proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente.

Características adicionales incluidas:

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.

- Soporte Enterprise Java Beans™ 2.0.
- Capacidad de análisis de calidad de código.
- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo. [14]

1.5 Metodologías para desarrollo de software.

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, etc.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño. [15]

1.5.1 RUP.

El proceso unificado de desarrollo (RUP) es una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El

resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. [16]

1.5.2 UML.

Para modelar el análisis y diseño de la biblioteca se escogió el lenguaje UML (Lenguaje Unificado de Modelado), debido a las potencialidades descriptivas que posee.

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. [17]

1.6 Qt: Framework De Desarrollo Multiplataforma.

Qt es un framework de desarrollo multiplataforma con una biblioteca de más de 400 clases, las cuales encapsulan toda la infraestructura necesaria para desarrollar aplicaciones robustas. Posee una utilidad, (QT Designer), para la rápida construcción de capas de interfaz gráfica de usuario con apariencia y aspecto nativos de las plataformas soportadas. Facilita el flujo de trabajo de internacionalización mediante QT Linguist. Está separado en 13 módulos y la API que brinda incluye funciones para conectividad a bases de datos, programación en red, integración con gráficos 3D, desarrollo multi-hilos y lectura/escritura de ficheros XML.

Introduce una alternativa innovadora para comunicación entre objetos, los llamados signals y slots, que reemplazan la antigua e insegura técnica callback. Además provee un modelo de eventos convencional para manipular clics del mouse, teclas presionadas y demás entradas del usuario. [18]

1.7 STK Editor.

STK Editor es una propuesta presentada en junio de 2008 por los ingenieros Yasmany Cubela Medina y Leonardo Nieblas Palau como parte de su tesis de diploma. Dentro de sus objetivos se propusieron: “crear una arquitectura flexible que soporte la adición de plugins” [19]. Esta arquitectura es un prototipo de prueba que será utilizado en el futuro diseño de un entorno de diseño integrado basado en STK y Qt, el cual brindará la opción de agregar extensiones y plugins; para de esa forma empotrar gran variedad de herramientas, tanto de configuración de los distintos aspectos y parámetros de los entornos virtuales de STK, como de uso general para proyectos de investigación académica y actividades docentes. El mayor acierto de la misma es que brinda al usuario una interfaz lo más abstracta posible para implementar el comportamiento de los distintos módulos que interactúan en el núcleo de SceneToolkit, logrando así utilizar las funcionalidades de la Herramienta gráfica de forma más fácil y adicionando soporte para interfaces visuales basadas en Qt. [18]

1.8 Scene Toolkit.

SceneToolKit es un Motor gráfico actualmente en desarrollo por el Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual de la Facultad 5 de la Universidad de Ciencias Informáticas, Cuba. Utiliza las bibliotecas gráficas OpenGL y DirectX sobre las plataformas Windows y Linux. Posee su propio formato de fichero para almacenar la información de los entornos virtuales, “3dx”, aunque se encuentra en desarrollo su sucesor, el “stk”. Posee módulos de comunicación por redes, audio, física, seguimiento de terreno, shaders e interfaz gráfica de usuario. En la actualidad se encuentra en desarrollo un paquete de herramientas visuales que permitirán ahorrar tiempo y esfuerzo a la hora de configurar sus entornos y elementos. Su arquitectura se divide en tres capas: ***Engine*** donde se encuentra el procesamiento importante de carga y manejo de objetos; ***Renderer*** donde se define con qué biblioteca gráfica se van a dibujar los entornos; y una capa ***Application*** donde se hace el manejo de eventos del sistema operativo específico. [18]

Conclusiones.

En este capítulo quedan sentadas las bases para el entendimiento del desarrollo de la aplicación y el ámbito en el que evolucionará, se adentra en el conocimiento de aspectos tales como: la visualización en tiempo real y la visualización 3D en tiempo real; se analizaron algunas de las herramientas que se usan actualmente para el desarrollo de los paseos virtuales; se presentaron de la misma forma los motores gráficos que soportan los modelos y su visualización; se describieron las herramientas y metodologías de desarrollo a utilizar, así como la estructura de la biblioteca Scene ToolKit y del framework QT.

Capitulo 2: Soluciones Técnicas.

Introducción

En este capítulo se hará una descripción de la propuesta a defender, en vista a solucionar la situación actual en el campo de acción. Se proponen soluciones técnicas para el funcionamiento de la herramienta gráfica que permita el montaje y configuración de los modelos 3D para paseos virtuales, y soluciones específicas para lograr la representación visual de los paseos, el almacenamiento de la misma en ficheros y la aplicación de técnicas de visualización.

Para el proceso de realización de recorridos virtuales se toman como base dos elementos fundamentales, los CCD (Creadores de Contenido Digital) de los que depende la calidad gráfica de los modelos 3D y un motor gráfico (engine), que debe brindar una interfaz para soportar los modelos y las texturas, además de proporcionar una estructura de grafo para definir los caminos del paseo, definir las colisiones, así como tener la posibilidad de modificar el aspecto visual de algunos modelos 3D bajo ciertas circunstancias (brillo, contraste, etc.). Importante es el vídeo y las animaciones teniendo en cuenta los formatos que el motor gráfico soporte, estos harán que el conjunto audio visual del paseo virtual sea uniforme y unísono al proceso.

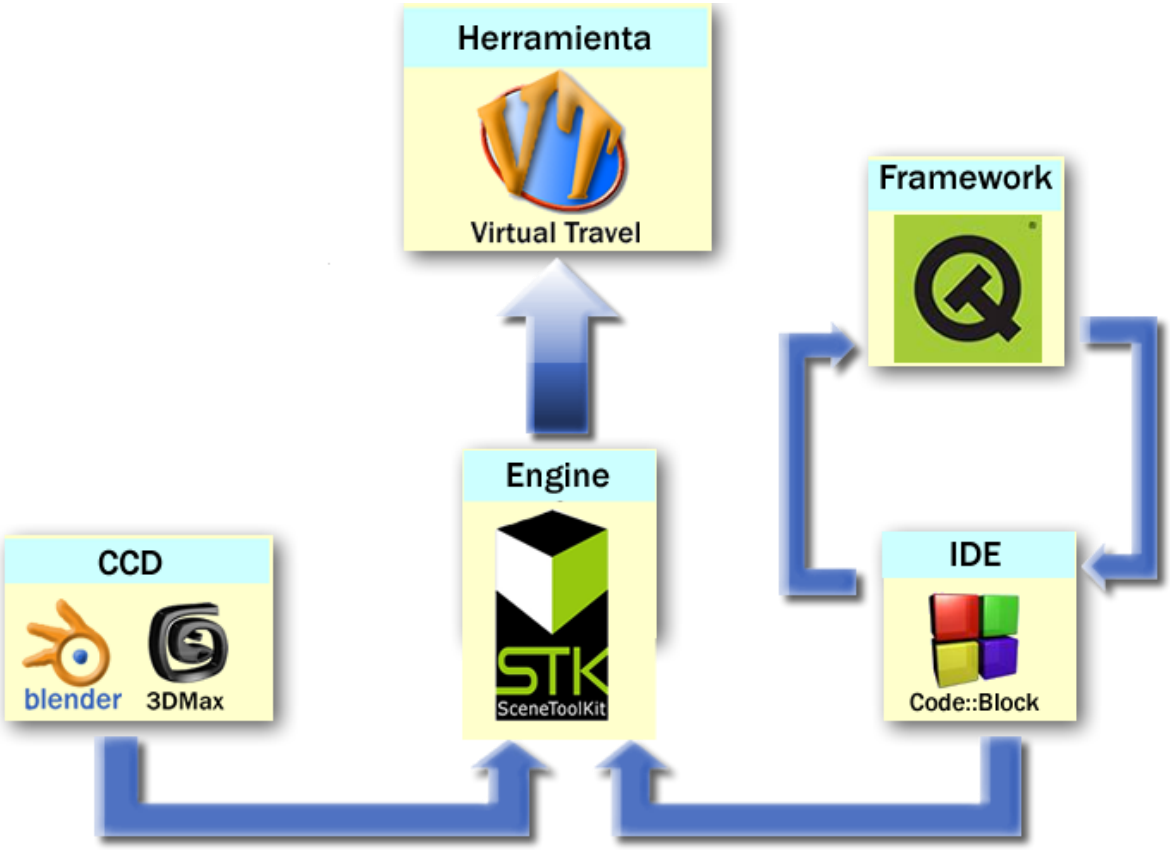


Figura 1. Proceso de edición de un paseo virtual.

2.1 El Sistema.

Para la edición de los modelos 3D los usuarios se apoyan en los CCD que son aplicaciones especializadas, entre ellas se cuenta con: 3D Max Studio; Blender; etc. Estas herramientas tienen como salida los modelos 3D y las texturas a cargar en nuestra herramienta y estas se exportarán como 3DX.

Presenta elementos a considerarse también como entrada, estos son: los vídeos y las animaciones que están en concordancia al ambiente que se diseñe. Los vídeos deberán ser compatibles con el formato AVI, teniendo como Stream de vídeo XviD y las animaciones se exportarán en formato ANX.

Teniéndose como entrada los modelos 3D, los vídeos y las animaciones, el motor gráfico entraría a ser parte fundamental dentro de la herramienta, pues permitiría la carga y

exportación de cada uno de estos elementos; la carga de grafos de caminos; la configuración de la iluminación de los modelos 3D; definir las colisiones.

La herramienta se encargaría además de: configurar el menú de navegación, que está fuertemente relacionado con el grafo de caminos para los recorridos, definir si el paseo será dirigido o libre.

Todo esto daría como salida una aplicación visual de un recorrido virtual determinado. Este proceso se puede apreciar en la siguiente figura:

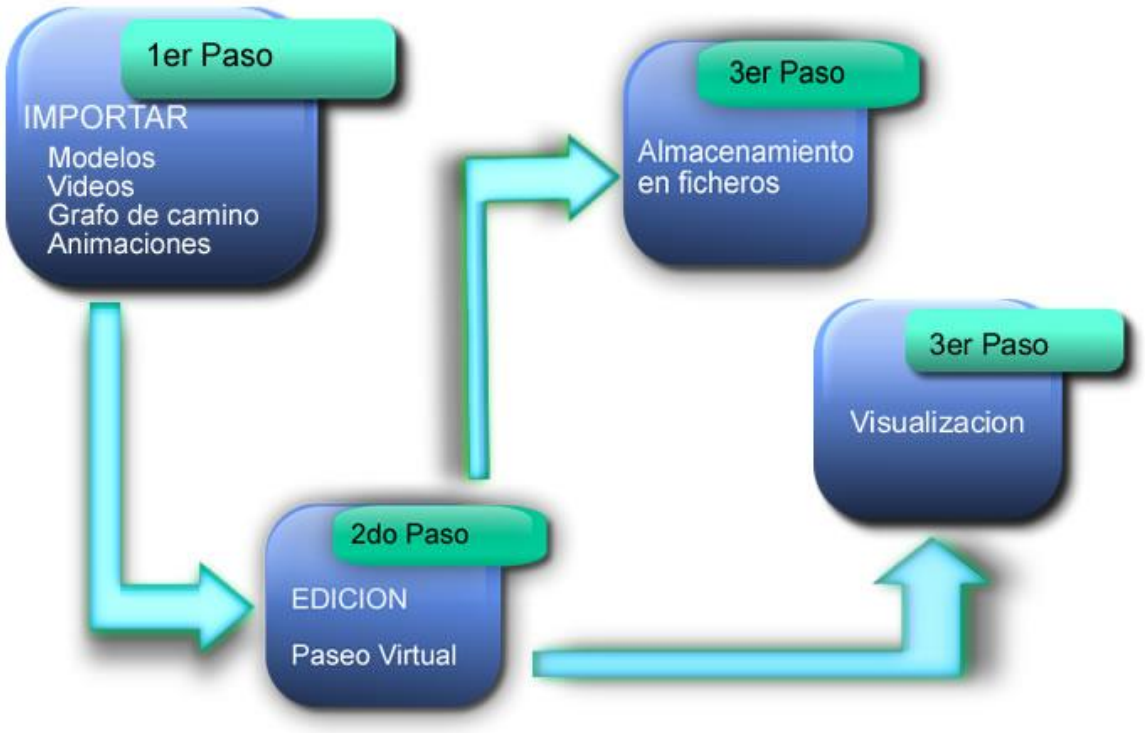


Figura 2. Dinámica de la herramienta de edición de paseos virtuales.

2.2 Grafos de Caminos

Un grafo es un objeto matemático que se utiliza para representar circuitos, redes, caminos, entre otras. Los grafos son muy utilizados en computación, ya que permiten resolver problemas muy complejos. [20]

Como ejemplo pudiésemos tomar nuestra universidad, con carreteras que unen lugares significativos a los que presentan acceso. Cada lugar significativo (Plaza Wifredo Lam, Docentes, Rectorado) tiene varias vías de acceso o carreteras que llegan a ellos, por lo que para trasladarse de un lugar a otro se podrán tomar diversos caminos. Cada carretera tendrá un coste asociado que puede ser representado mediante la longitud de la misma. Mediante la representación por grafos (Ver Figura 3) se puede elegir el camino más corto que conecta dos lugares, determinar si es posible llegar de un lugar a otro, si desde cualquier lugar existe un camino que llegue a cualquier otro, entre otras aplicaciones.

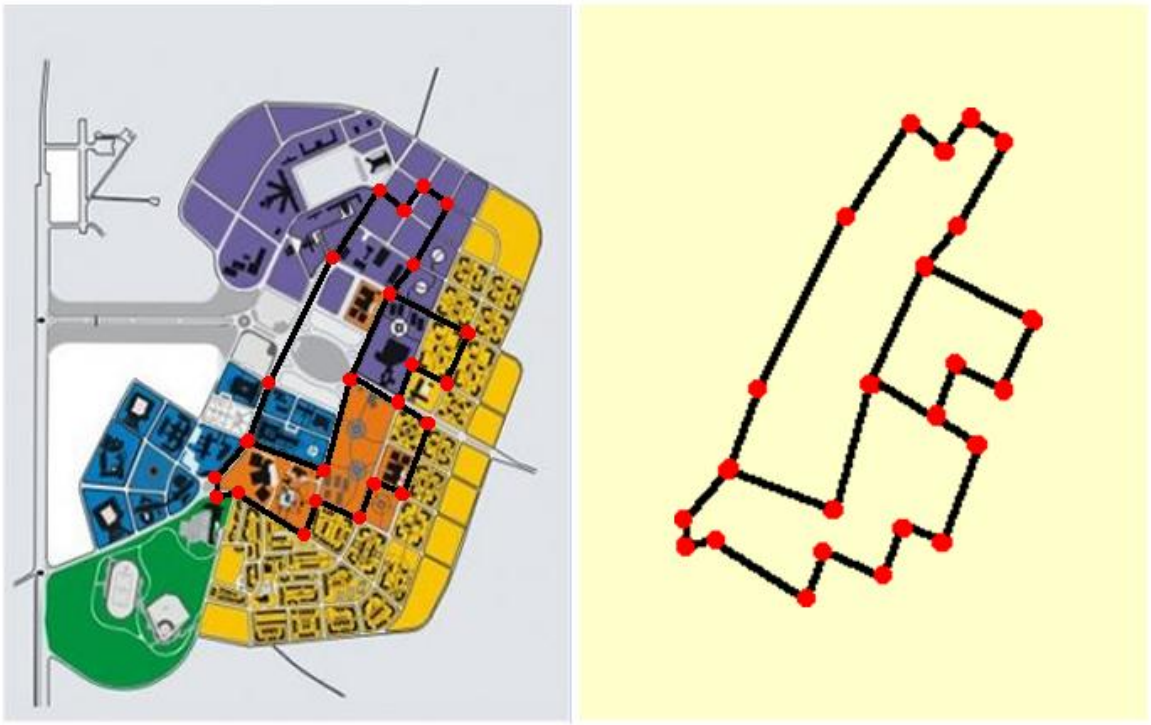


Figura 3. Grafo de caminos.

2.3 Estructura del fichero.

Siguiendo estándares de formatos de ficheros 3D usados en el campo de la Realidad Virtual como son los ficheros 3DS o MD2, el fichero 3dx presenta una estructura dada en bloques de datos, en la siguiente figura se muestra de forma clara esta estructura.

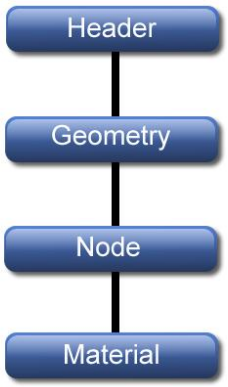


Figura 4. Estructura del fichero.

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

2.3.1 Características del bloque Header.

Header es el bloque que representa el encabezado del fichero 3d, brindando informaciones generales tales como la versión del formato, el nombre de la escena, el tamaño del fichero, entre otros. A continuación se muestra la estructura del encabezamiento (Header) para un mejor entendimiento.

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

2.3.2 Características del bloque Geometry

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

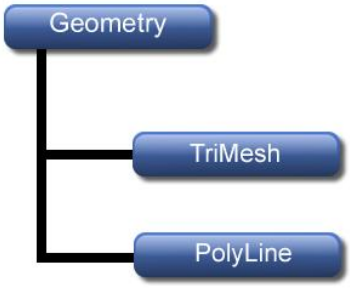


Figura 5. Estructura del bloque geometría.

Existen dos tipos de geometrías a exportar, una malla (TriMesh) y una polilínea (Polyline). Los parámetros que contienen el bloque geometría son comunes para los dos tipos de geometría:

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

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

2.3.2.1 Características del subbloque TriMesh.

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

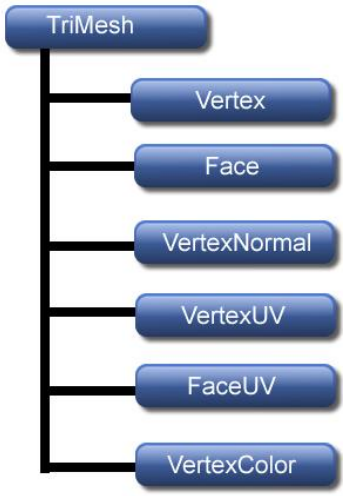


Figura 6. Estructura del subbloque TriMesh.

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

- unsined int t_TriQuantity: cantidad de triángulos que conforman la malla.
- sTriMesh t_TriList[t_TriQuantity]: lista de estructuras triángulos, cada estructura conformada por tres índices de los vértices correspondiente al triángulo.
- sVertex 3*t_VertexNormalList[t_TriQuantity]: lista de estructuras de normales de vértices dada por las caras.
- bool t_ListExist: byte de información que se usa para saber qué atributos contiene la malla; un bit para cada existencia del atributo, los atributos son los que siguen: t_VertexColorList, t_TriColorList, t_UVList0, t_UVList1, t_UVList2 y t_UVList3.

- unsined int t_VertexTQuantity0: cantidad de vértices con textura en el canal de mapeo1.
- unsined int t_VertexTQuantity1: cantidad de vértices con textura en el canal de mapeo 2.
- unsined int t_VertexTQuantity2: cantidad de vértices con textura en el canal de mapeo 3.
- unsined int t_VertexTQuantity3: cantidad de vértices con textura en el canal de mapeo 4.
- unsined int t_VertexCQuantity: cantidad de vértices con color.

- sVertex t_VertexUVList0[t_VertexQuantity]: lista de coordenadas de textura referente a los valores u y v para el mapa difuso.
- sTriMesh t_TriUVList0[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- sVertex t_VertexUVList1[t_VertexTQuantity]: memoria reservada a futuro uso.
- sTriMesh t_TriUVList1[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- sVertex t_VertexUVList2[t_VertexTQuantity]: lista de coordenadas de textura para el mapa de luces.
- sTriMesh t_TriUVList2[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- sVertex t_VertexUVList3[t_VertexTQuantity]: memoria reservada para futuro uso.
- sTriMesh t_TriUVList3[t_TriQuantity]: lista de estructuras de texturas de caras, cada una compuesta por los índices de los vértices con texturas correspondientes al canal.
- sVertexC t_VertexColorList[t_VertexCQuantity]: lista de colores de vértices, dados en la combinación de los tres colores que usa OpenGL para pintar r(rojo), g(verde), b(azul).
- sTriMesh t_TriColorList[t_TriCQuantity]: lista de estructuras colores triángulos, cada estructura está conformada por tres índices de los vértices con color correspondientes al triángulo.

2.3.2.2 Características del subbloque Polyline.

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

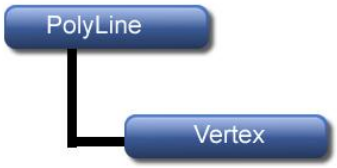


Figura 7. Estructura del subbloque Polyline.

Los parámetros que contienen este subbloque son los que siguen:

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

2.3.3 Características del bloque Node.

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

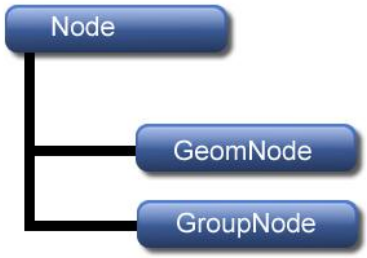


Figura 8. Estructura del bloque Node.

El bloque nodo trata a cada objeto en la escena como un todo, analiza sólo las características generales y propiedades desde el punto de vista de cómo se van a representar en la escena, datos tales como: el grado visibilidad, la matriz de

transformación, entre otros. Está dividido en dos subbloques, lo cual presenta gran ventaja, a continuación se analizan las mismas en cada subbloque.

- GeomNode: optimiza el tamaño del fichero ya que en caso de que en la escena existan objetos que sean instancias o copias idénticas de otro objeto, sólo se exporta en el fichero un único objeto y un nodo por cada instancia, los cuales contienen la matriz de transformación del objeto que representa dónde están ubicados en la escena. Un ejemplo típico es cuándo se exporta un entorno donde hay mucha vegetación, generalmente se crean instancias de un mismo tipo de árbol o arbusto por todo el terreno, por lo que se exportaría un solo árbol y los nodos que son instancias del mismo.
- GroupNode: optimiza el dibujado de la escena y el trabajo con la misma una vez que se cargó el fichero.

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

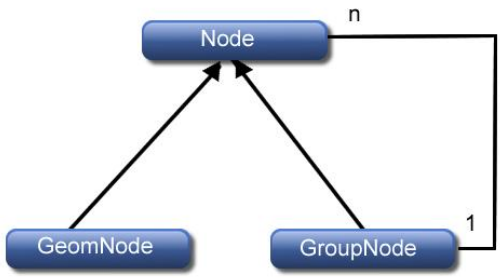


Figura 9. Estructura del subbloque GroupNode.

Ambos tipos de nodos contiene los mismos parámetros, sólo que varían de acuerdo al tipo. Los parámetros son los siguientes.

- int nNode_ChildrenQuantity: cantidad de hijos de un nodo, para el caso de un nodo geométrico es 0.
- bool nNode_Children_Brother: un byte que se usa para saber si es hermano o hijo del nodo anterior, sí el primer bit está activado es hermano, sino es hijo.
- int nNode_Index: índice de la geometría que está contenida en el nodo geométrico, en caso de ser un nodo grupo su índice es -1.

- int nNode_MaterialIndex: índice del material que usa el nodo. En el caso particular de que no use ningún material toma valor -1.
- float nNode_TMatriz: matriz de transformación que exporta el 3d Studio Max.
- bool nNode_BackFaceCull: en caso de que sea 1 los triángulos mirados por detrás son invisibles, en caso que sea 0 visibles.
- float nNode_Visibility: nivel de visibilidad, está dado entre cero y uno.
- int nNode_UserPropertiesSize: tamaño que ocupa un comentario acerca del nodo dado por el usuario.
- char nNode_UserProperties[ng_UserPropertiesSize]: definición dada por el usuario al nodo.

¿Cómo se exporta la lista de hijos?

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

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

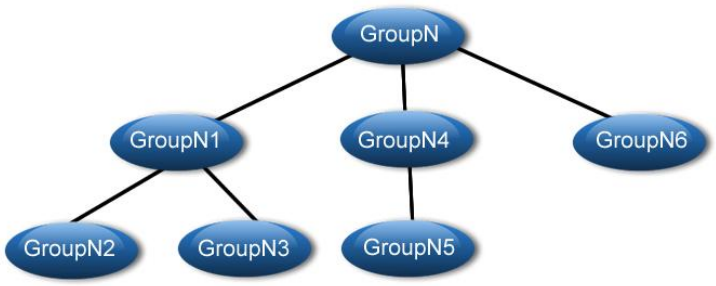


Figura 10. Estructura del árbol de nodos.

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

2.3.4 Características del bloque Material.

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

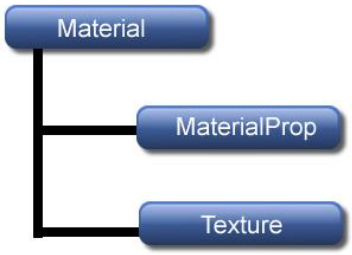


Figura 11. Estructura del bloque Material.

2.3.4.1 Estructura del subbloque MaterialProp.

El subbloque MaterialProp contiene todas las propiedades de los materiales en una escena.

Los parámetros que contienen son los que se exponen a continuación.

- char mp_Name: nombre del material.
- bool mp_TwoSide: en caso de ser 1 está activado el modo de dos lados, en caso de ser 0 lo contrario.
- int mp_AmbientColor: color ambiental dado en rgb.
- int mp_DiffuseColor: color difuso dado en rgb.
- int mp_SpecularColor: color especular dado en rgb.
- float mp_Opacity: nivel de opacidad.
- int mp_IndexAmbient: índice de la textura que se le pasó por el canal ambiente, en caso de no tener asignada ninguna toma valor -1.
- int mp_IndexDiffuso: índice de la textura que se le pasó por el canal difuso, en caso de no tener asignada ninguna toma valor -1.
- int mp_IndexOpacity: índice de la textura que se le pasó por el canal de opacidad, en caso de no tener asignada ninguna toma valor -1.

- int mp_IndexSpecular: índice de la textura que se le pasó por el canal especular, en caso de no tener asignada ninguna toma valor -1.
- int mp_IndexMaterial1: reservado.
- int mp_IndexLighMap: índice de la textura que se le pasó por el canal de mapa de luces, en caso de no tener asignada ninguna toma valor -1.
- int mp_IndexMaterial2: reservado.

2.3.4.2 Estructura del subbloque Textura.

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

- char te_Name[m_TextureQuantity]: lista de caminos y nombres de texturas. [21]

2.4 Interfaz gráfica de usuario.

Para el desarrollo de la interfaz gráfica de usuario se hará uso del framework de STK Editor ya que fue un requisito exigido por el cliente. Entre las características sobresalientes del mismo están la fácil integración de QT y STK, permitiendo la creación de interfaces visuales en conjunto con el motor gráfico, logrando así utilizar las funcionalidades de la Herramienta gráfica de forma más fácil. Además brinda una serie de funcionales que permite al programador enfocarse en su solución específica, sin tener que preocuparse por modelar casos de uso generales como cargar mundo, maximizar/minimizar cámaras, activar el modo malla, cambiar de cámaras, entre otros.

2.5 Consideraciones generales.

El diseño e implementación se corresponderá con la filosofía de Programación Orientada a Objetos. Todo el trabajo de la herramienta se hará a través de funciones programadas puramente en el lenguaje de programación C++ Standard.

Conclusiones

Podemos apreciar que con la utilización de la biblioteca de clases QT y del motor gráfico SceneToolkit en conjunto con el desarrollo de la aplicación con el lenguaje C++, todos multiplataforma, crearemos un Software adaptable a las necesidades según la conveniencia del cliente. Además se destacan las posibilidades de: crear los caminos del entorno de manera visual e intuitiva; vincular los modelos 3D, los vídeos, las animaciones y el sonido de forma visual; configurar el menú de navegación; aplicar técnicas de planificación de movimientos desde un punto de vista estático y visualizar los resultados.

El presente capítulo deja sentadas las bases técnicas sobre las cuales será implementada la herramienta visual para el montaje de paseos virtuales, para dar solución al objetivo propuesto.

Capitulo 3: Características del Sistema.

Introducción

En este capítulo se enmarca en la conceptualización de la solución propuesta y se comienza a tener una visión más práctica del sistema a desarrollar. Se definen las reglas del negocio y el modelo de dominio. Se obtienen los requisitos funcionales (capacidades o funciones que el sistema debe cumplir) y los no funcionales (propiedades o cualidades que el producto debe tener). Además se describen los procesos que responden a las funcionalidades definidas en los requerimientos funcionales.

3.1 Reglas del Negocio.

Las reglas de negocio son una serie de restricciones de la organización a la hora de realizar una determinada actividad, e incluyen las restricciones asociadas a las informaciones (restricciones de integridad) y a las actividades, por lo que regulan algún aspecto del negocio, por esta razón es de gran importancia identificarlas dentro del negocio, evaluar si son relevantes dentro del campo de acción que se está modelando e implementarlas en la propuesta de solución.

1. Los formatos de los modelos 3D que se cargan deben ser 3DX.
2. El formato de los vídeos que se cargan debe ser AVI con stream de vídeo XviD.
3. Las animaciones que ser cargan deben tener el formato ANX.
4. Configuración del menú de navegación de forma visual.
5. Configuración de los principales modelos 3D, su vínculo con los vídeos y las animaciones de forma visual.
6. Visualización en tiempo real del paseo virtual.
7. Se exportará el archivo con el formato 3DX.

3.2 Modelo del Domino.

El modelo de dominio representa un acercamiento a la solución propuesta, donde se modelan los principales conceptos implicados en el desarrollo de la solución, así como las relaciones existentes entre ellos.

A continuación se representa el modelo del dominio referente a la herramienta propuesta.

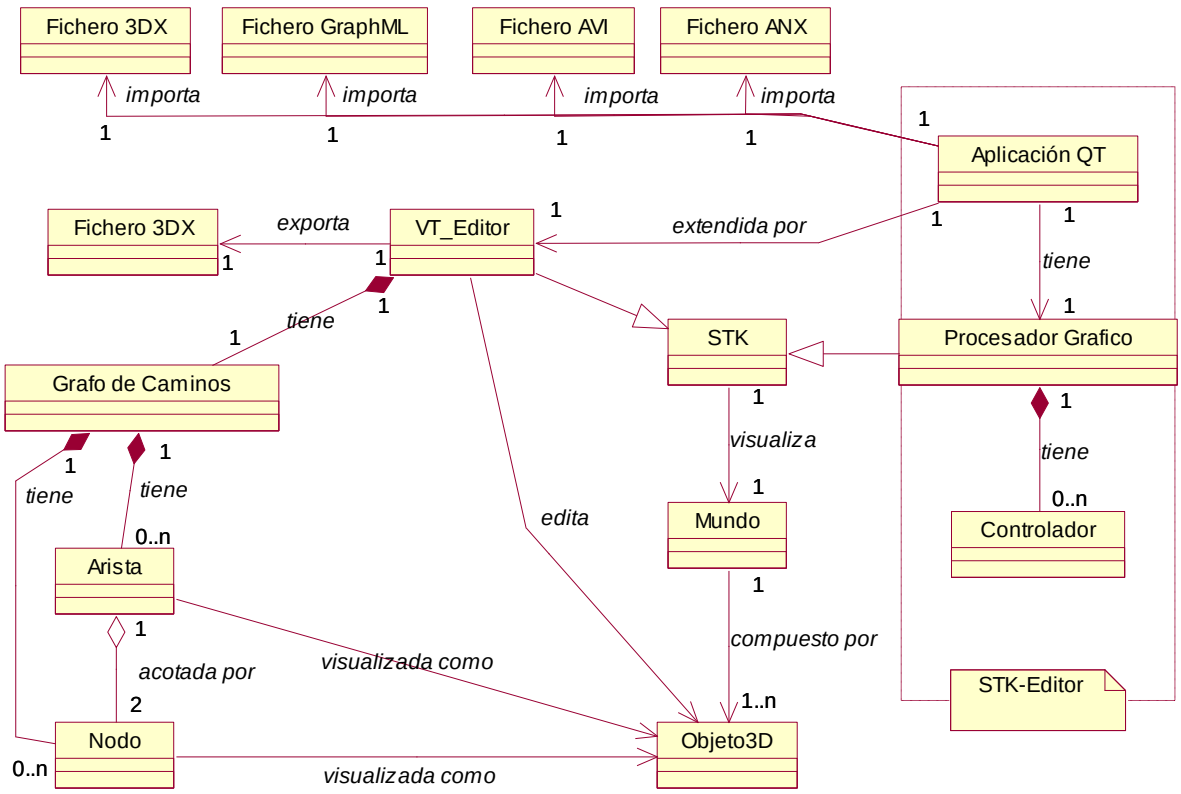


Figura 12. Modelo del dominio.

3.3 Glosario de términos del negocio.

VT_Editor: responsable de la gestión y edición de los elementos del **Mundo**.

Grafo de caminos: es una estructura integrada por un conjunto de caminos, así como los arcos y nodos relacionados con los mismos.

Nodo: estructuras que sirven de guías a la hora de crear caminos en el entorno 3D, son vértices o posiciones específicas del entorno que describen la secuencia de los caminos.
Arista: son arcos o uniones creados entre un par de vértices o nodos, las cuales tendrán una dirección indicando la navegabilidad entre los nodos. Poseen un conjunto de vértices de control.
Fichero 3DX: fichero con la información referente a un mundo 3D.
Fichero GraphML: fichero con la información referente al grafo de caminos.
Fichero AVI: fichero con la información referente al vídeo.
Fichero ANX: fichero con la información referente a las animaciones.
Mundo: mundo 3D, entorno virtual, contiene la información gráfica de varios objetos 3D.
Objeto 3D: representación gráfica tridimensional computarizada de un objeto del mundo real.
Procesador gráfico: es la instancia de STK que servirá de base a la arquitectura de STK Editor, especializa a su ancestro mediante la incorporación de los controladores.
Aplicación Qt: interfaz gráfica de usuario basada en Qt que brinda todas las funcionalidades de STK de forma visual e intuitiva. Define interfaces para que se le puedan acoplar extensiones.
STK: motor gráfico para la visualización de objetos 3D.
Controlador: fragmento de funcionalidad de STK implementado por el usuario (programador de extensiones) para lograr un comportamiento personalizado de las funcionalidades del motor gráfico.

3.4 Captura de requisitos.

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, la cual tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. [22]

3.4.1 Requisitos funcionales.

Son capacidades o condiciones que el sistema debe cumplir y se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. [22]

RF1. Importar los modelos 3D.

1.1. Cargar archivo con formato .3DX.

RF2. Importar los vídeos.

2.1. Cargar archivo con formato .AVI.

RF3. Importar las animaciones.

3.1. Cargar archivo con formato .ANX.

RF4. Importar Grafo de Camino

4.1. Cargar archivo con formato .GraphML.

RF5. Dibujar el mundo 3D.

5.1. Crear escena con todos los objetos que se definan en el mundo virtual cargado.

RF6. Gestionar caminos.

6.1. Gestionar seguimiento de la cámara por el grafo de camino.

6.2. Crear colisiones.

RF7. Dibujar menú.

7.1. Crear menú de navegación.

RF8. Editar los modelos 3D.

8.1. Modificar color del modelo.

8.2. Adicionar un modelo a lista de modelos significativos.

8.3. Eliminar un modelo de la lista de modelos significativos.

RF9. Vincular videos, animaciones y modelos.

9.1. Crear vínculo seleccionando con el clic del mouse los modelos 3D y los vídeos y/o animaciones correspondientes a estos.
9.2. Eliminar vínculo entre los modelos 3D y los vídeos y/o animaciones correspondientes a estos.

RF10. Exportar mundo.
10.1. Guardar archivo en formato .3DX.

3.4.2 Requisitos no funcionales.

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

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. [22]

Usabilidad

- Cualquier usuario que quiera editar un paseo virtual sobre un entorno 3D.

Portabilidad

- Debe ser portable a cualquier Sistema Operativo Windows, familia de Unix.

Requerimientos de Software

- Sistema Operativo Windows XP o superior.

Requerimientos de Hardware

- . Memoria RAM de 256 MB o superior.

- Tarjeta de Vídeo 128 MB o superior.

Restricciones en el Diseño e Implementación

- Lenguaje de programación C++ bajo el paradigma de programación Orientado a Objeto.

Diseño e Implementación

- Se debe desarrollar utilizando el Entorno de Desarrollo Integrado (del inglés Integrated Development Environment (IDE)) Code::Block.
- Se debe desarrollar utilizando la biblioteca gráfica STK.

3.5 Modelo de casos de uso del sistema.

En esta sección se muestran los casos de uso del sistema divididos de acuerdo a etapas importantes en cuanto al proceso de exportación del fichero, se concibe los casos de uso del sistema así como los actores que van a interactuar con cada uno de ellos. Además se seleccionan los casos de uso correspondientes al ciclo de desarrollo para hacerles sus especificaciones textuales en formato expandido.

3.5.1 Actores del sistema.

Los actores representan entidades que interactúan con el sistema, un actor del sistema es aquel que se beneficia de con los resultados de las funcionalidades del mismo. [22]

Actores	Justificación
Diseñador	Es el que se beneficia con las funcionalidades que brinda el sistema, en este caso exportar un fichero en formato STK.

Tabla 1: Descripción de actor del sistema.

3.5.2 Casos de Uso del sistema.

1. Cargar Contenido Digital.
2. Diseñar menú de navegación.
3. Gestionar Caminos.
4. Gestionar Modelos Significativos.
5. Gestionar vínculos de contenido digital.
6. Gestionar entorno.
7. Exportar mundo.

3.5.3 Paquetes de Casos de Uso del sistema.

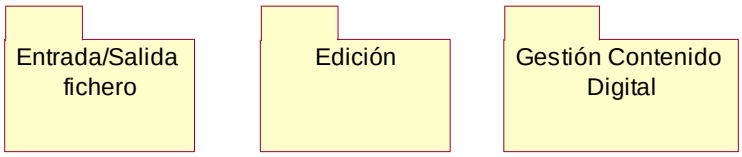


Figura 13. Paquetes de Casos de Uso del sistema.

En la figura anterior quedan representados los 3 paquetes generales del sistema. Entrada/Salida fichero se encarga de leer y escribir la información de forma persistente, Edición calcula y consulta caminos mínimos, diseño del menú y registra cambios en los modelos 3D, mientras Gestión de Contenido Digital crea, modifica y elimina los vínculos entre el contenido digital, así como crea o elimina un modelo significativo.

3.5.3 Diagramas de Casos de Uso del sistema.

El diagrama de casos de uso del sistema describe la funcionalidad propuesta del nuevo sistema, basándose en la captura de los requisitos funcionales, y muestra las relaciones entre los casos de uso que representan las funcionalidades o principales procesos del sistema y los actores que interactúan con el mismo.

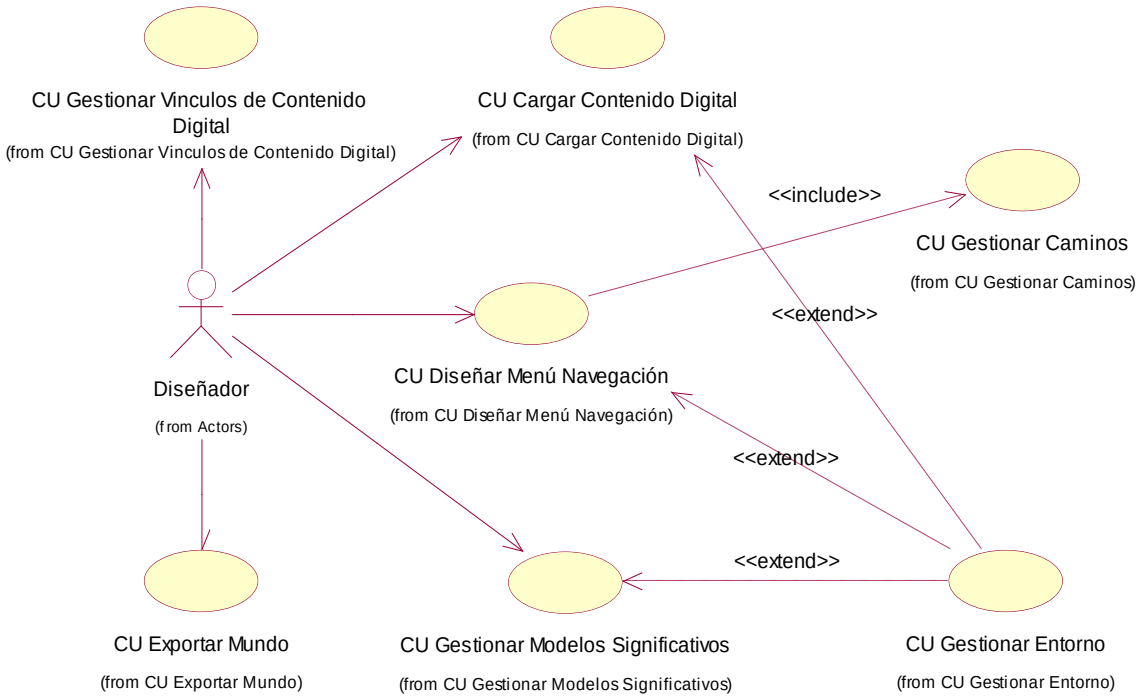
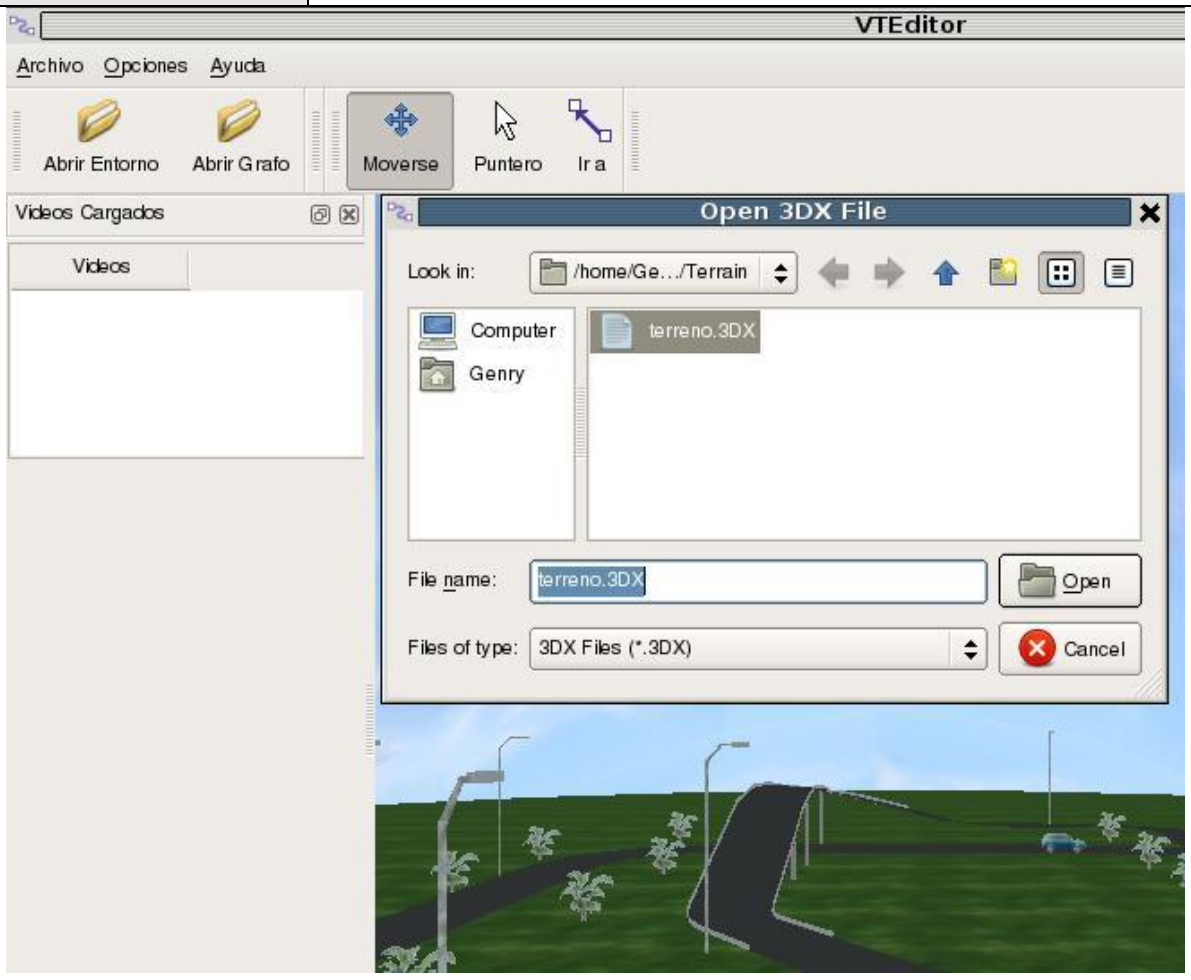


Figura 14. Diagrama de caso de uso del sistema.

El diseñador inicia el CU “Cargar Contenido Digital” en la cual se carga en la herramienta los modelos 3d, los vídeos, las animaciones y el grafo de camino, y a su vez se crea como subproceso el CU “Gestionar Entorno” que se encarga de cargar, actualizar y visualizar en el espacio de trabajo los objetos del mundo, a este CU también se utiliza como subproceso de dos CU: “Diseñar Menú Navegación” y “Gestionar Modelos Significativos”, el primero de estos tiene la responsabilidad de crear el menú de navegación del paseo virtual; además este incluye un CU “Gestionar Caminos” que crea las colisiones y prepara la cámara para llegar a los lugares significativos, el segundo se encarga de que queden registrados en un listado los modelos significativos. El diseñador también puede iniciar el CU “Gestionar Vínculos de Contenido Digital” donde se registran en los listados correspondientes el vínculo entre los modelos significativos y los vídeos/animaciones marcados con un id y el CU ”Exportar Mundo” que se encarga de exportar toda la información de la escena.

3.5.4 Descripción de los Casos de Uso del sistema.

Caso de Uso	
CU-1	**Cargar Contenido Digital.**
Actores	Diseñador
Resumen	El CU se inicia cuando el Diseñador solicita importar contenido digital. Solicita la opción de importar contenido digital. Selecciona el camino donde se almacena el archivo como modelos 3D en formato .3dx, archivos de vídeo en formato .avi, animaciones en formato .anx o el archivo del grafo de caminos con formato .graphml.
Referencias	RF1, RF2, RF3, RF4.
Precondiciones	-



Flujo Normal de Eventos	
Sección Cargar Modelos 3D.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario se dirige al menú principal y selecciona *Archivo*.	1.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Cargar*.
2. El usuario escoge dentro del submenú *Cargar* la opción *Cargar Modelos 3d*.	2.1. El sistema muestra un cuadro de diálogo pre-configurado que permite especificar la ruta de donde se va a cargar el archivo, o cancelar la operación.
3. El usuario especifica la ruta de donde desea cargar el archivo y pulsa abrir.	3.1. El sistema cierra el cuadro de diálogo. 3.2. El sistema verifica que la ruta sea válida. 3.3. El sistema carga el archivo de la ruta especificada y construye el mundo. Ver CU-6
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
3.1 El usuario decide cancelar la operación de cargar el fichero.	
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	3.2.1. Si la ruta no es válida el sistema muestra un mensaje de error.
Sección Cargar Vídeo.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario se dirige al menú principal y selecciona *Archivo*.	1.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Cargar*.
2. El usuario escoge dentro del submenú *Cargar* la opción *Cargar Vídeo*.	2.1. El sistema muestra un cuadro de diálogo pre-configurado que permite especificar la ruta de donde se va a cargar

	el archivo, o cancelar la operación.
3. El usuario especifica la ruta de donde desea cargar el archivo y pulsa abrir.	3.1. El sistema cierra el cuadro de diálogo. 3.2. El sistema verifica que la ruta sea válida. 3.3. El sistema carga el archivo de la ruta especificada y lo carga en un listado de Videos Cargados.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
3.1. El usuario decide cancelar la operación de cargar el fichero.	
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	3.2.1. Si la ruta no es válida el sistema muestra un mensaje de error.
Sección Cargar Animación.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario se dirige al menú principal y selecciona *Archivo*.	1.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Cargar*.
2. El usuario escoge dentro del submenú *Cargar* la opción *Cargar Animación*.	2.1. El sistema muestra un cuadro de diálogo pre-configurado que permite especificar la ruta de donde se va a cargar el archivo, o cancelar la operación.
3. El usuario especifica la ruta de donde desea cargar el archivo y pulsa abrir.	3.1. El sistema cierra el cuadro de diálogo. 3.2. El sistema verifica que la ruta sea válida. 3.3. El sistema carga el archivo de la ruta especificada y lo carga en un listado de Animaciones Cargadas.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**

3.1. El usuario decide cancelar la operación de cargar el fichero.	
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	3.2.1. Si la ruta no es válida el sistema muestra un mensaje de error.
Sección Cargar Grafo de Caminos.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario se dirige al menú principal y selecciona *Archivo*.	1.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Cargar*.
2. El usuario escoge dentro del submenú *Cargar* la opción *Cargar Grafo de Caminos*.	2.1. El sistema muestra un cuadro de diálogo pre-configurado que permite especificar la ruta de donde se va a cargar el archivo, o cancelar la operación.
3. El usuario especifica la ruta de donde desea cargar el archivo y pulsa abrir.	3.1. El sistema cierra el cuadro de diálogo. 3.2. El sistema verifica que la ruta sea válida. 3.3. El sistema carga el archivo de la ruta especificada y lo carga en el Mundo. Ver CU-6
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
3.1. El usuario decide cancelar la operación de cargar el fichero.	
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	3.2.1. Si la ruta no es válida el sistema muestra un mensaje de error.
Puntos de Extensión	CU-6
Poscondiciones	Al cargarse los modelos 3d, los videos, las animaciones y el grafo de caminos, se crea una lista con los videos para asignarse a los

	modelos, una lista con las animaciones que se van a poner en el mundo, un grafo de caminos por donde se va a mover la cámara del mundo y todos los modelos 3d incluyendo el mundo cargados en la escena.
Prioridad	Crítica.

Tabla 2: Descripción del CU Cargar Contenido Digital.

Caso de Uso	
CU-2	**Diseñar menú de navegación.**
Actores	Diseñador.
Resumen	El CU se inicia cuando el Diseñador selecciona en el menú de la aplicación Crear Menú de navegación.
Referencias	RF7
Precondiciones	Haber cargado el Mundo y el grafo de caminos con los nodos nombrados según su importancia por ubicación en el Mundo.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario se dirige al menú principal y selecciona *Opciones*.	1.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Crear menú de navegación.*
2. El usuario escoge el submenú *Crear menú de navegación.*	2.1. El sistema muestra una forma donde se listan los lugares más significativos que serán de punto de referencia para el paseo. Estos coinciden también con el nombre de los nodos significativos del grafo de caminos. 2.2. Se seleccionará si el menú será visible o no en una primera instancia. 2.3. Se seleccionará si el paseo va a ser libre o dirigido.
3. El usuario acepta la configuración del menú de navegación.	3.1. El sistema crea el menú de navegación con la siguiente estructura:

	quedará en la esquina superior izquierda del mundo; tendrá un listado numerado con los lugares significativos que al marcar el número del lugar se moverá hacia él, y a modo de información también mostrará opciones predefinidas por el sistema como son: - Aumentar o disminuir velocidad de movimiento con las teclas "+" y "-". - Mostrar u ocultar el menú con la letra "M". - Cambiar modo de Paseo: Libre o Dirigido con la letra "O".
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	2.1.1 El sistema muestra un mensaje de error si no se han cumplido las precondiciones.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
3.1 El usuario cancela la configuración del menú de navegación.	
Puntos de Extensión	-
Poscondiciones	Se crea el menú de navegación.

Tabla 3: Descripción del CU Diseñar menú de navegación.

Caso de Uso	
CU-3	**Gestionar Caminos.**
Actores	CU-2
Resumen	El CU se inicia cuando es invocado por el CU Diseñar menú de navegación. Se recorre el grafo de caminos para tener en una lista de aristas referentes a los lugares significativos por las que la

	cámara realizará el paseo.
Referencias	RF6
Precondiciones	Haber cargado el Mundo y el grafo de caminos con los nodos nombrados según su importancia por ubicación en el Mundo. Haber invocado el CU-2.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Brinda una lista de nodos significativos.	1.1. El sistema obtiene en una lista las aristas del grafo de camino con las condiciones óptimas para recorrerlo eficientemente. 1.2. El sistema realiza seguimiento de terreno a lo largo de las aristas para llegar a cualquier lugar.
	2. El sistema crea las colisiones sobre las lista de aristas para el movimiento desde un lugar a otro.

Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	1.1.1 El sistema muestra un mensaje de error si no se han cumplido las precondiciones.

Puntos de Extensión	-
Poscondiciones	Se crea el modo de llegar a cualquier lugar significativo.

Tabla 4: Descripción del CU Gestionar Camino.

Caso de Uso	
CU-4	**Gestionar Modelos Significativos.**
Actores	Diseñador
Resumen	El CU se inicia cuando el Diseñador selecciona un modelo de la escena para configurarlo.

Referencias	RF8
Precondiciones	Haber cargado el Mundo.
Flujo Normal de Eventos	
Sección Adicionar un modelo a lista de modelos significativos.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario oprime el click sobre un objeto de la escena para seleccionarlo.	1.1. El sistema muestra el objeto con un cambio de color que lo diferencia como seleccionado. Ver CU-6
2. El usuario se dirige al menú principal y selecciona *Opciones*.	2.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Configurar Modelo.*
3. El usuario selecciona la opción de *Adicionar Modelo Significativo*.	3.1. El sistema muestra una ventana con el nombre del modelo que esta seleccionado y un CheckBox desmarcado con la opción *Significativo.*
4. El usuario Marca el CheckBox y acepta la configuración.	4.1. El sistema lo adiciona como modelo significativo y lo muestra en un listado de modelos Significativos.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	2.1.1. El sistema muestra un mensaje de error si no se han cumplido las precondiciones.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
4.1 El usuario cierra la ventana *Adicionar Modelo Significativo.*	
Sección Eliminar un modelo de la lista de modelos significativos.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona un modelo de la lista de modelos significativos.	
2. El usuario se dirige al menú principal y	2.1. El sistema muestra varias opciones

selecciona *Opciones.*	entre las que se encuentra el submenú *Configurar Modelo.*
3. El usuario selecciona la opción de *Eliminar Modelo Significativo*.	3.1 El sistema elimina el modelo de la lista y actualiza el estado del modelo correspondiente.
Puntos de Extensión	CU-6
Poscondiciones	Quedan registrados en un listado los modelos significativos.

Tabla 5: Descripción del CU Gestionar Modelos Significativos.

Caso de Uso	
CU-5	**Gestionar Vínculos de Contenido Digital.**
Actores	Diseñador
Resumen	El CU se inicia cuando el Diseñador selecciona un vídeo o una animación para vincularla a un modelo significativo.
Referencias	RF9
Precondiciones	Haber cargado los videos y/o animaciones. Debe existir al menos un modelo en la lista de modelos significativos.
Flujo Normal de Eventos	
Sección Crear vínculos vídeos/animaciones con modelos significativos.	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona un modelo de la lista de modelos significativos. 1.1 El usuario selecciona un vídeo/animación de su respectiva lista.	
2. El usuario se dirige al menú principal y selecciona *Vínculos*.	2.1 El sistema muestra varias opciones entre las que se encuentra el submenú *Crear vínculo.*
3. El usuario selecciona la opción *Crear vínculo.*	3.1. El sistema muestra una ventana de confirmación para llevar a cabo el vínculo.
4. El usuario selecciona la opción aceptar.	4.1. El sistema muestra en las listas de video/animación y modelos significativos

	una actualización con la marca de un id que ayudará a identificar los vídeos/animaciones que ya están asignados a dichos modelos.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	4.1.1. El sistema muestra un mensaje de error si no tiene seleccionado un modelo y un vídeo/animación.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	4.2. El sistema un mensaje de error si el modelo y el vídeo/animación seleccionados ya están vinculados.
****Sección Eliminar vínculos vídeos/animaciones con modelos significativos.**	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona un modelo de la lista de modelos significativos que este referenciado con un id. 1.1 El usuario selecciona un vídeo/animación de su respectiva lista que este referenciado con el mismo id que el modelo.	
2. El usuario se dirige al menú principal y selecciona *Vínculos*.	2.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Eliminar vínculo.*
3. El usuario selecciona la opción *Eliminar vínculo.*	3.1 El sistema muestra una ventana de confirmación para eliminar el vínculo.
4. El usuario selecciona la opción aceptar.	4.1. El sistema elimina el vínculo y actualiza las listas de modelos significativos y vídeo/animación respectivamente.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**

	4.1.1. El sistema muestra un mensaje de error si no tiene seleccionado un modelo y un vídeo/animación.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	4.2. El sistema muestra un mensaje de error si el modelo y el vídeo/animación seleccionados no están vinculados.
Puntos de Extensión	-
Poscondiciones	Quedan registrados en sus listados correspondientes el vínculo entre los modelos significativos y los vídeos/animaciones marcados con un id.

Tabla 6: Descripción del CU Gestionar vínculos de contenido digital.

Caso de Uso	
CU-6	**Gestionar Entorno.**
Actores	CU-1, CU-2,CU-4
Resumen	El CU se inicia cuando es invocado por los CU Cargar Contenido Digital; CU Diseñar Menú de Navegación o Gestionar Modelos Significativos. Se encarga de cargar, modificar y visualizar el mundo.
Referencias	RF10.
Precondiciones	Haber invocado el CU-1. Haber invocado el CU-2. Haber invocado el CU-4.
Flujo Normal de Eventos	
Sección Crear Escena	
Acción del Actor	**Respuesta del Sistema**
1. Al invocar CU-1 “Cargar Contenido Digital” Sección Cargar modelos 3D.	1.1. El sistema visualiza la escena en la aplicación.
Sección Modificar Escena	
Acción del Actor	**Respuesta del Sistema**

1. Al invocar CU-1 “Cargar Contenido Digital” Sección Cargar grafo de caminos.	1.1. El sistema visualiza el grafo de caminos en la aplicación.
2. Al invocar CU-2 “Diseñar menú de navegación ”	2.1. El sistema visualiza un panel en escena.
3. Al invocar CU-4 “Gestionar modelos significativos”	3.1. El sistema visualiza los cambios de color en un modelo.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
Puntos de Extensión	-
Poscondiciones	Se carga, modifica y visualiza el mundo.

Tabla 7: Descripción del CU Gestionar Entorno.

Caso de Uso	
CU-7	**Exportar mundo.**
Actores	Diseñador.
Resumen	El CU se inicia cuando el diseñador selecciona en el menú de la aplicación salvar proyecto.
Referencias	RF10.
Precondiciones	Haber hecho algún cambio en un proyecto cargado anteriormente.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario se dirige al menú general y selecciona *Archivo.*	1.1. El sistema muestra varias opciones entre las que se encuentra el submenú *Salvar*.
2. El usuario selecciona la opción de S*alvar.*	2.1. El sistema muestra un cuadro de diálogo pre-configurado que permite especificar la ruta de donde se va a salvar el archivo, o cancelar la operación.
3. El usuario especifica la ruta donde se	3.1. El sistema verifica que la ruta sea

guardará el proyecto y pulsa aceptar.	válida y guarda el proyecto en la dirección asignada. 3.2. El sistema limpia el espacio de trabajo.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
3.1. El usuario pulsa cancelar.	3.1.1. El sistema cierra el cuadro de diálogo.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	3.1.2. El sistema muestra un error de ruta no válida.
Puntos de Extensión	-
Poscondiciones	Se salva el proyecto y se limpia el espacio de trabajo.

Tabla 8: Descripción del CU Gestionar Entorno.

Conclusiones

En el presente capítulo se definió qué espera el usuario con este sistema. Para ello quedaron establecidos sus requisitos funcionales y no funcionales, se presentó el diagrama de casos de uso del sistema y se describieron los casos de uso que permitirán al usuario obtener los resultados esperados.

CAPITULO 4: ANÁLISIS Y DISEÑO.

Introducción.

La primera parte del capítulo tratará todo lo referente a los diagramas de clases del análisis del sistema propuesto. Su objetivo es brindar una primera visión de las posibles clases del diseño a un alto nivel, pero donde se dejan ver sus relaciones y responsabilidades.

A continuación se presentan los diagramas de clases del diseño por paquetes como resultado del refinamiento de las etapas anteriores. El diagrama de clases es uno de los elementos más significativos dentro de un proyecto de software, ya que brinda una visión bastante completa de todo el sistema, mostrando sus clases, métodos, atributos y las relaciones entre ellos. Se presentan además los diagramas de secuencia de la realización de los casos de uso que intervendrán en el primer ciclo de desarrollo del proyecto.

4.1 Diagrama de paquetes de clases del análisis.

En el siguiente diagrama se muestra la dependencia entre los paquetes del análisis.

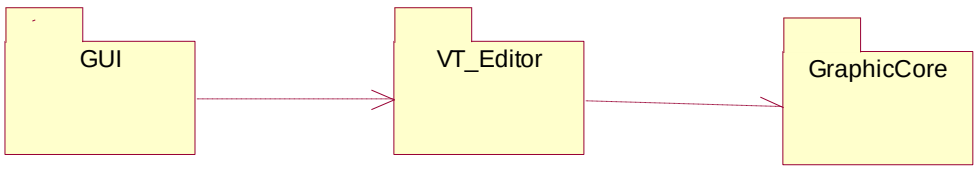


Figura 15. Diagrama de clases de análisis.

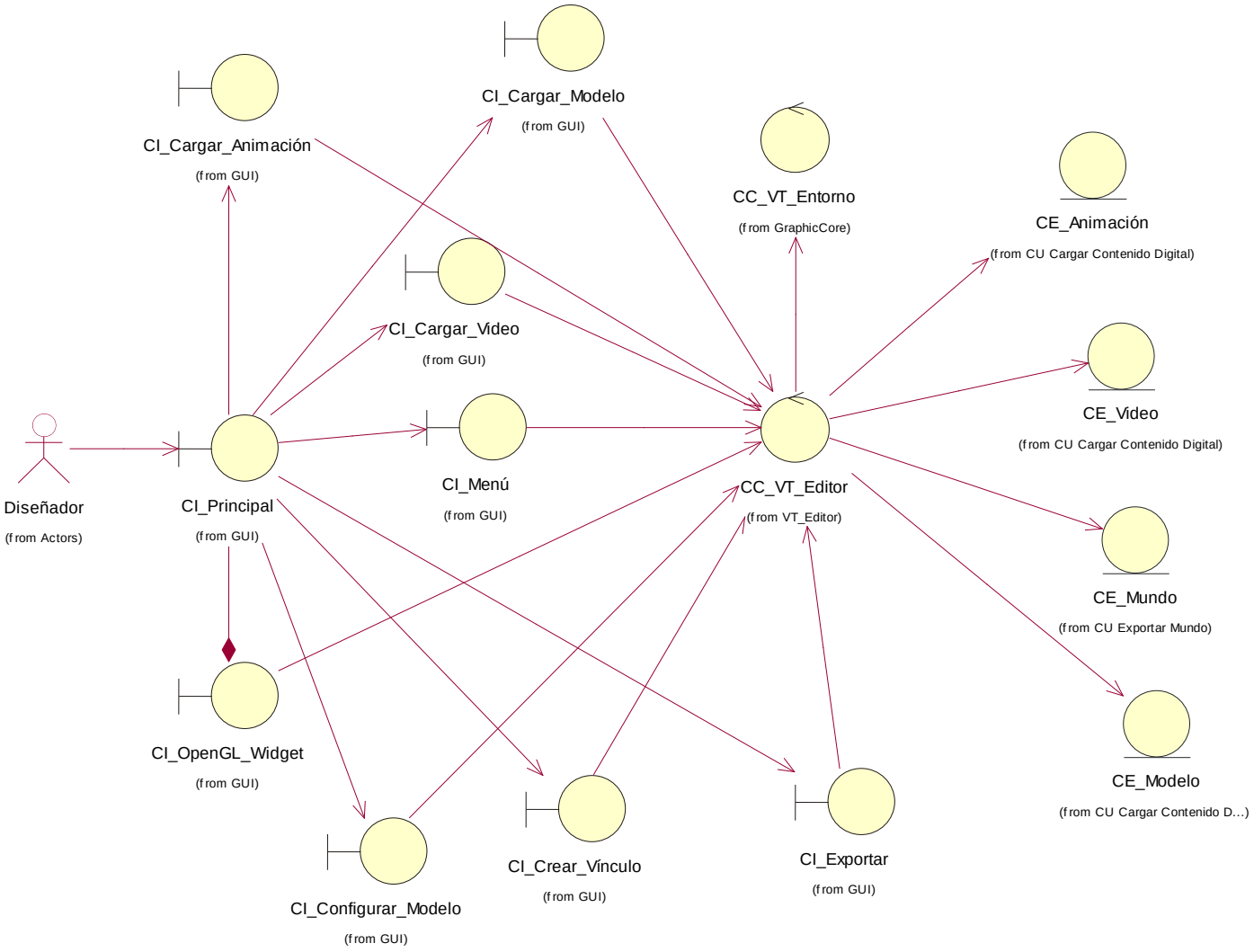


Figura 16. Diagrama de clases de análisis.

Este diagrama muestra las relaciones que existen entre las diferentes clases de análisis de los distintos paquetes. La clase CI_Principal es la ventana principal de la aplicación. La clase CI_Visualizador es un componente visual que será la interfaz donde se dibujarán los gráficos 3D. La clase CC_VT_Editor es la representación de la SceneToolKit. La clase CC_VT_PathGraph es la gestora del grafo de caminos cuando se dibuja la escena. Por

último la clase CC_VT_Entorno es la que se encarga de la carga del entorno en la escena donde se va a realizar el paseo virtual.

4.2 Diagrama de clases del diseño.

El sistema propuesto consta de cuatro paquetes: GUI, VT_Editor, GraphicCore y libSceneToolkitQT. En el paquete GUI se encuentran las clases de manejo de interfaz de usuario, encargadas de la entrada y salida de información de forma visual, así como del manejo de eventos. En el paquete VT_Editor se encuentra el controlador principal de la aplicación y la lógica del negocio. El paquete GraphicCore brinda una interfaz para el trabajo con SceneToolkit, facilitando el trabajo con la misma y abstrayendo al programador de su estructura interna y funcionamiento. Por último, libSceneToolkitQT contiene los binarios de SceneToolkit, utilizados para las operaciones de dibujado 3D y tratamiento de ficheros 3DX.

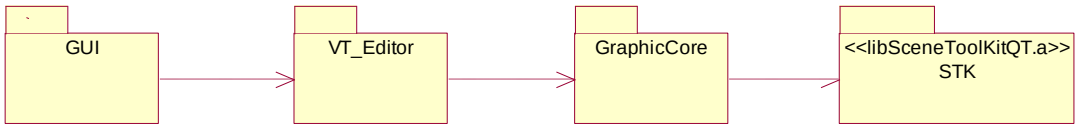


Figura 17. Diagrama de paquetes del diseño.

La figura 17 muestra la dependencia entre los distintos paquetes del diseño. En este punto hay que destacar que GraphicCore depende completamente de SceneToolkit.

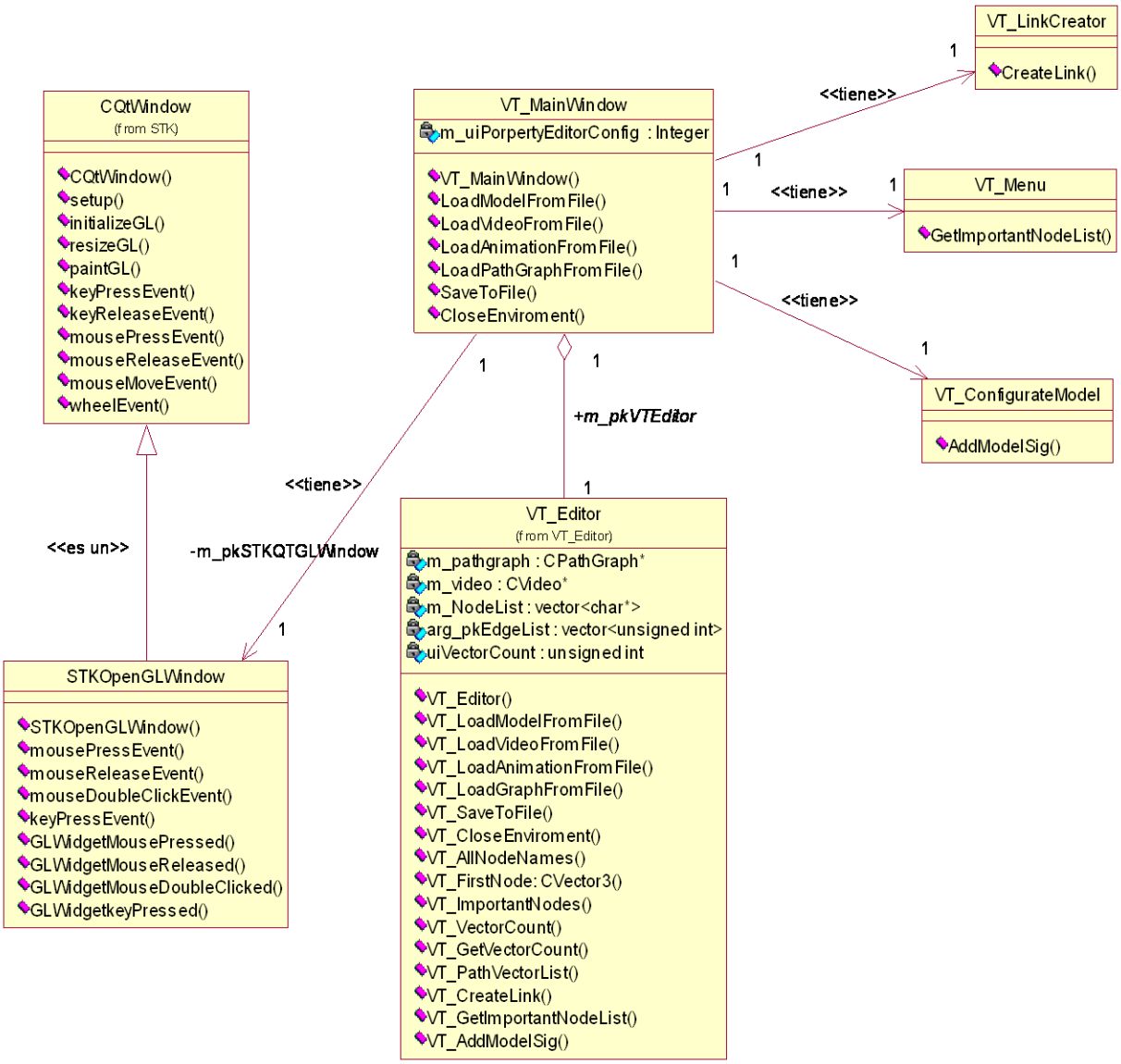


Figura 18. Diagrama de clases del diseño, paquete "GUI".

En este diagrama se muestran todas las clases de interfaz gráfica de usuario y las relaciones entre ellas. Fíjese que VT_MainWindow es la ventana principal. STKOpenGLWindow hereda de CQtWindow, quien se encarga de gestionar los eventos de STK y de dibujar la escena 3D.

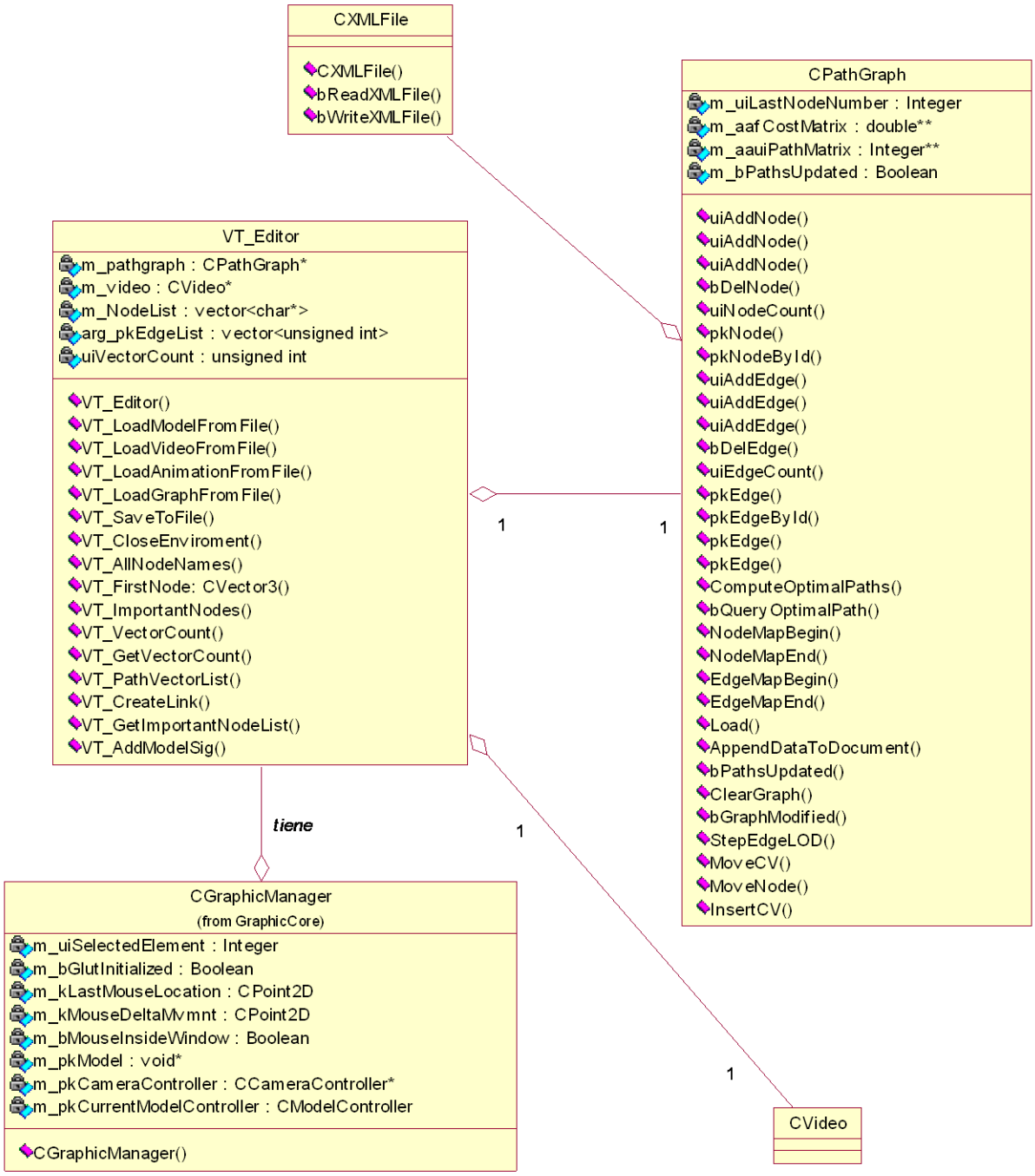


Figura 19. Diagrama de clases del diseño, paquete "VT_Editor".

En este diagrama se aprecia la relación entre las clases CPathGraph y CVideo con VT_Editor, que contiene la lógica del negocio y representa el controlador principal de la aplicación.

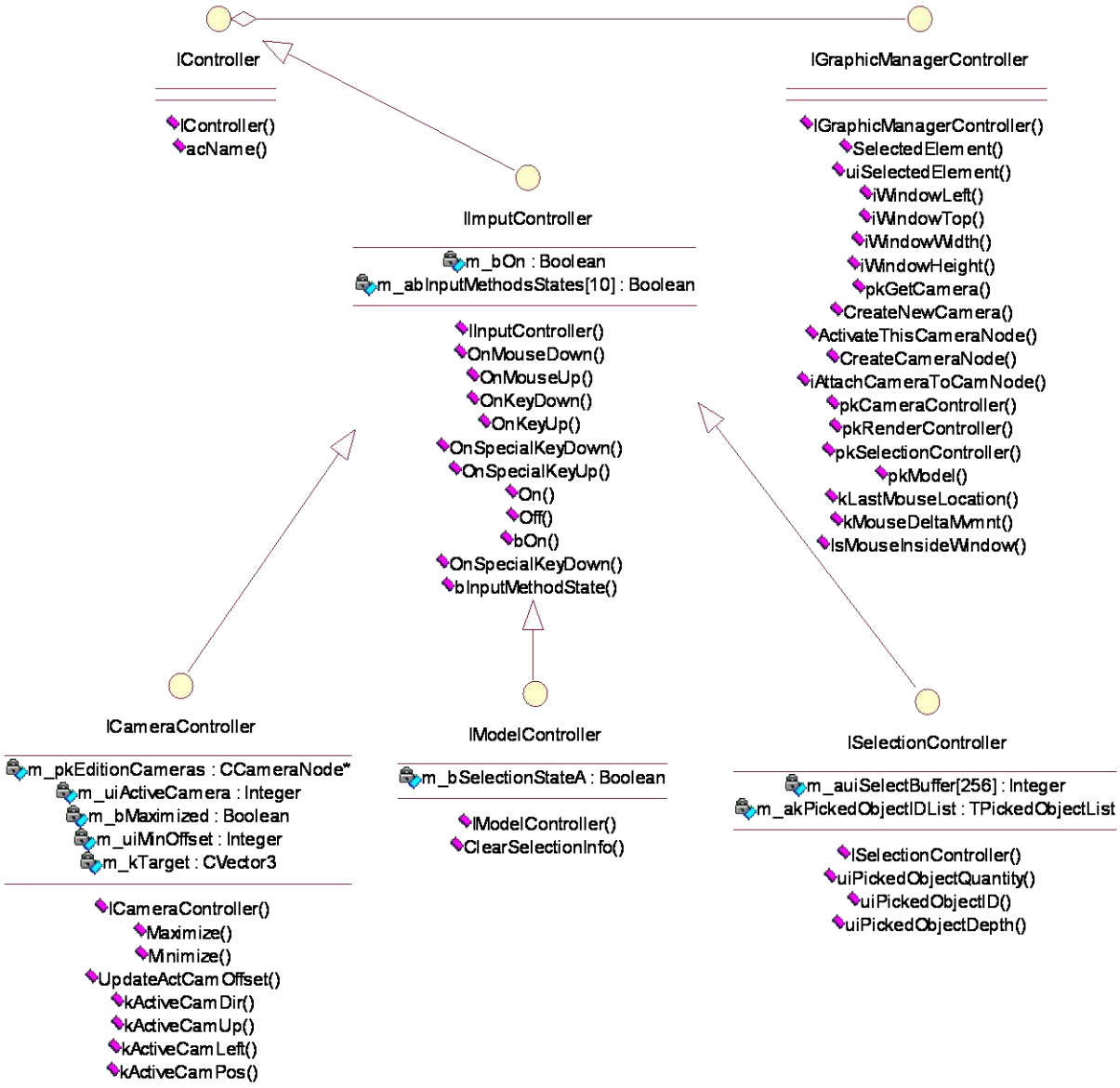


Figura 20. Diagrama de clases del diseño, paquete "GraphicCore", vista "Interfaces".

En esta imagen se puede observar cada uno de los tipos de controladores del framework de STK Editor. IGraphicManagerController es el encargado de brindarles el acceso a las funcionalidades de CGraphicManager. Esta jerarquía de controladores fue tomada del la herramienta GraphBuilder,[17], pues define un aporte al desarrollo del trabajo con el framework.

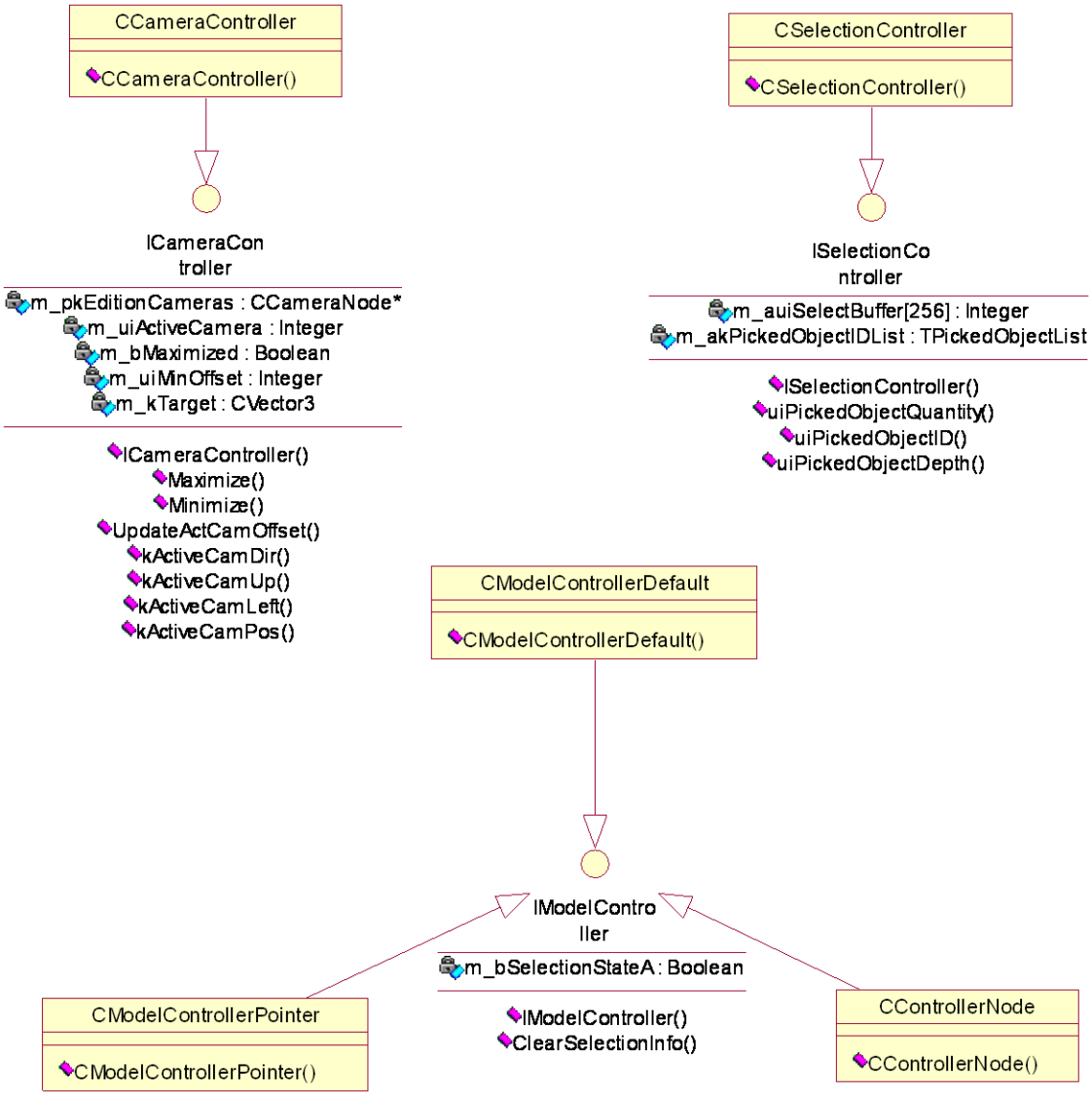


Figura 21. Diagrama de clases del diseño, paquete "GraphicCore", vista "Controllers".

Esta figura especifica cómo el usuario del framework de STK Editor debe heredar las interfaces de los controladores e implementar su propio comportamiento. [17]

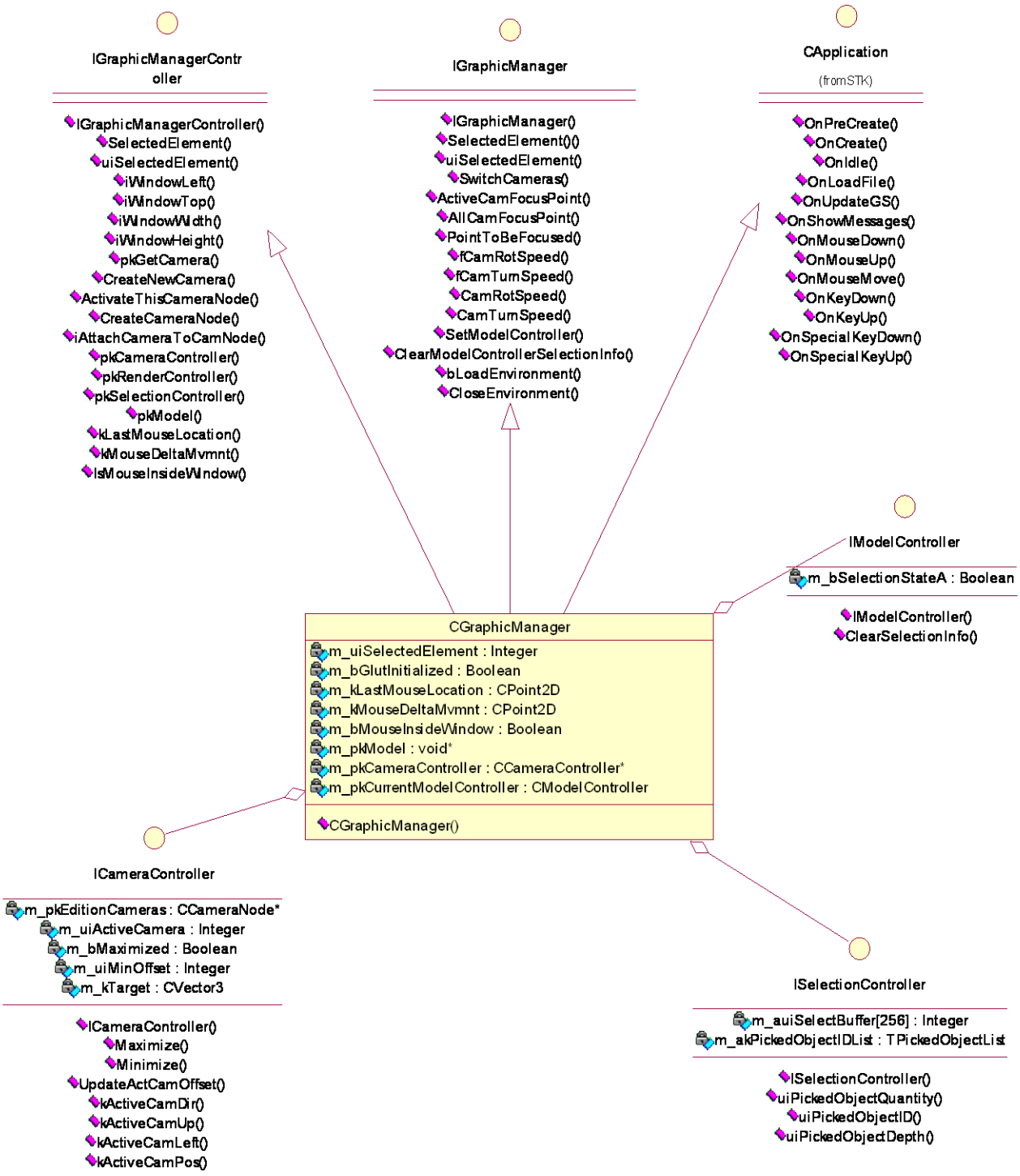


Figura 22. Diagrama de clases del diseño, paquete "GraphicCore", vista "GraphicManager".

Observe en esta imagen cómo CGraphicManager hereda de 3 interfaces; al heredar de CApplication garantiza las funcionalidades de STK; implementa la interfaz IGraphicManager para brindar una interfaz al exterior del paquete (a GraphBuilder) el cual no debe conocer otra información que no sea la que se le brinda; implementa la interfaz IGraphicManagerController para brindar a los controladores acceso a aquella funcionalidad que les es permitida (por ejemplo: un controlador no tiene permitido llamar funciones de manejo de eventos en CGraphicManager, ya que la aplicación podría caer en un ciclo infinito). [17]

4.3 Diagramas de secuencia

A continuación siguen los diagramas de secuencia agrupados por paquetes de casos de uso.

Paquete "Entrada/Salida fichero".

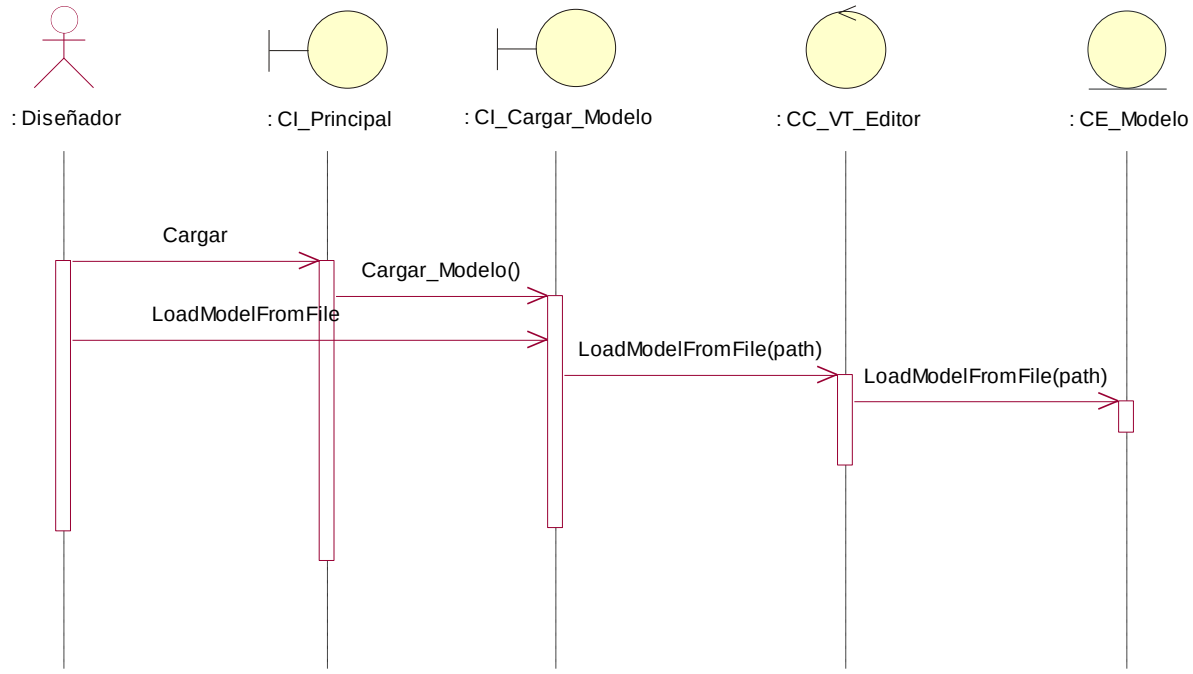


Figura 23. Diagrama de secuencia del caso de uso "Cargar Contenido Digital" sección: "Cargar Modelos 3D".

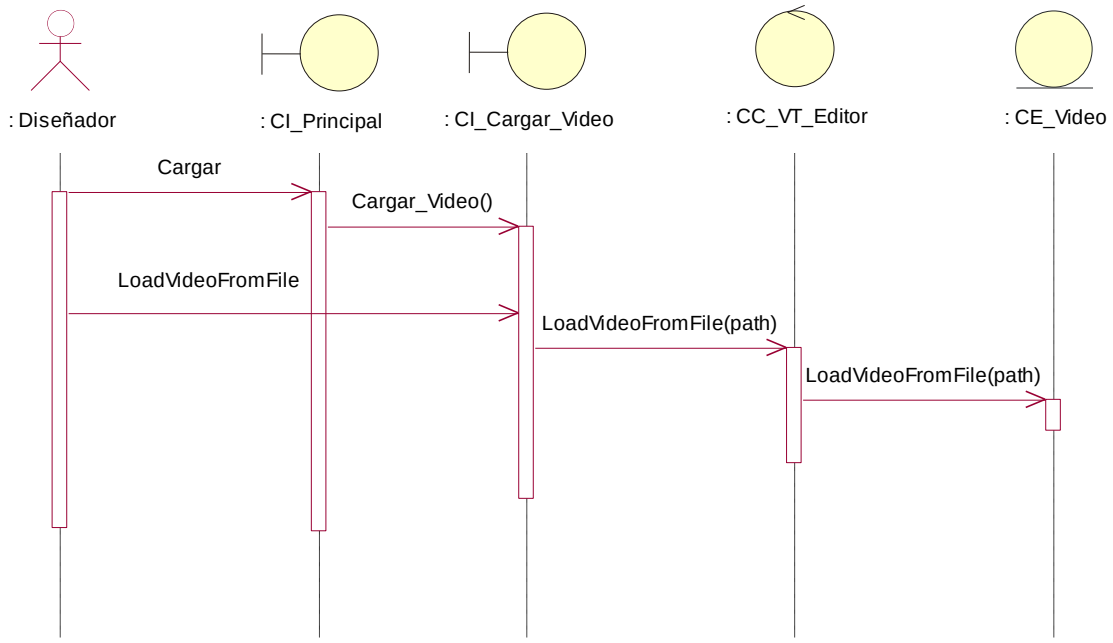


Figura 24. Diagrama de secuencia del caso de uso "Cargar Contenido Digital" sección: "Cargar Vídeo".

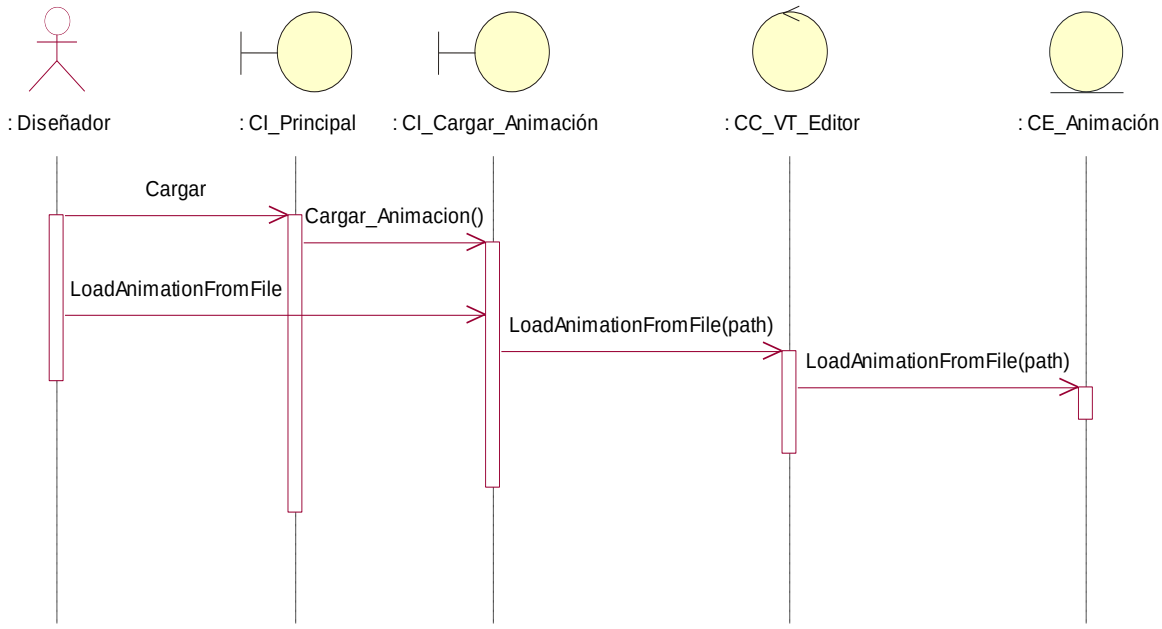


Figura 25. Diagrama de secuencia del caso de uso "Cargar Contenido Digital" sección: "Cargar Animación".

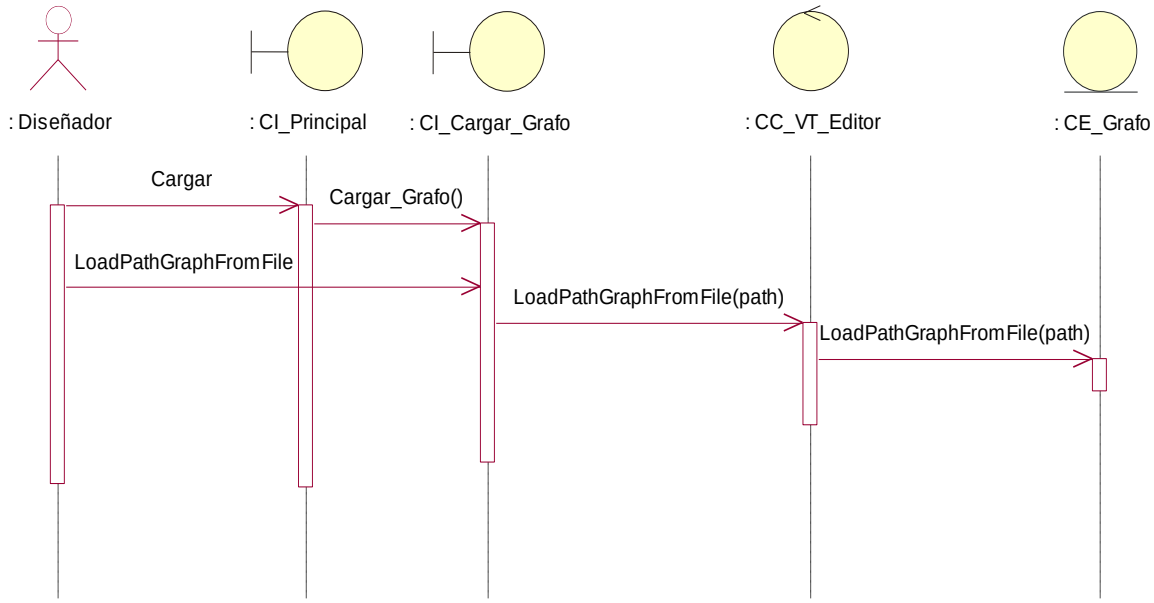


Figura 26. Diagrama de secuencia del caso de uso "Cargar Contenido Digital" sección: "Cargar Grafo de caminos".

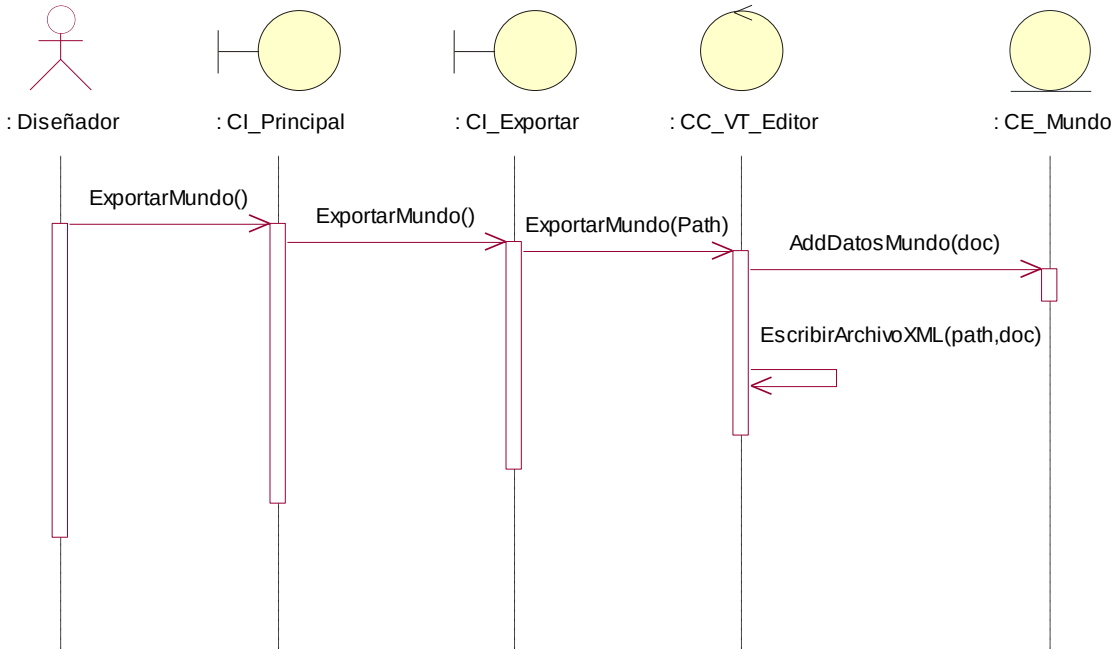


Figura 27. Diagrama de secuencia del caso de uso "Exportar Mundo".

Paquete “Edición”.

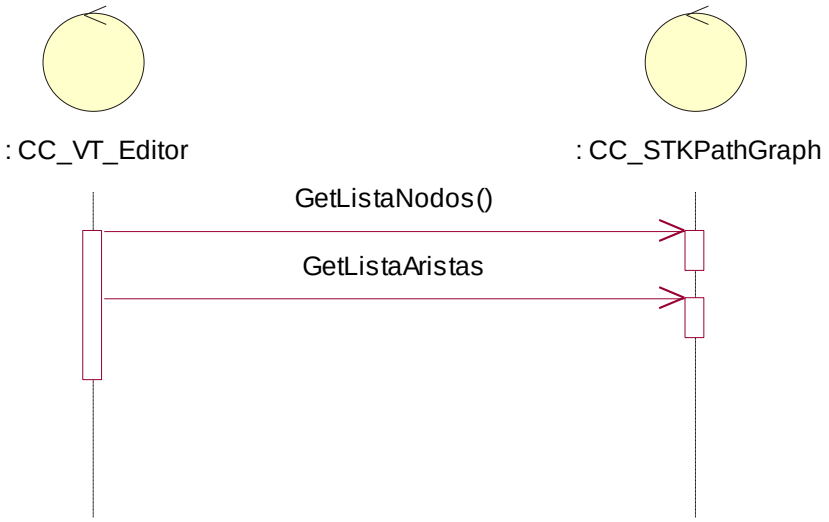


Figura 28. Diagrama de secuencia del caso de uso “Gestionar Caminos”.

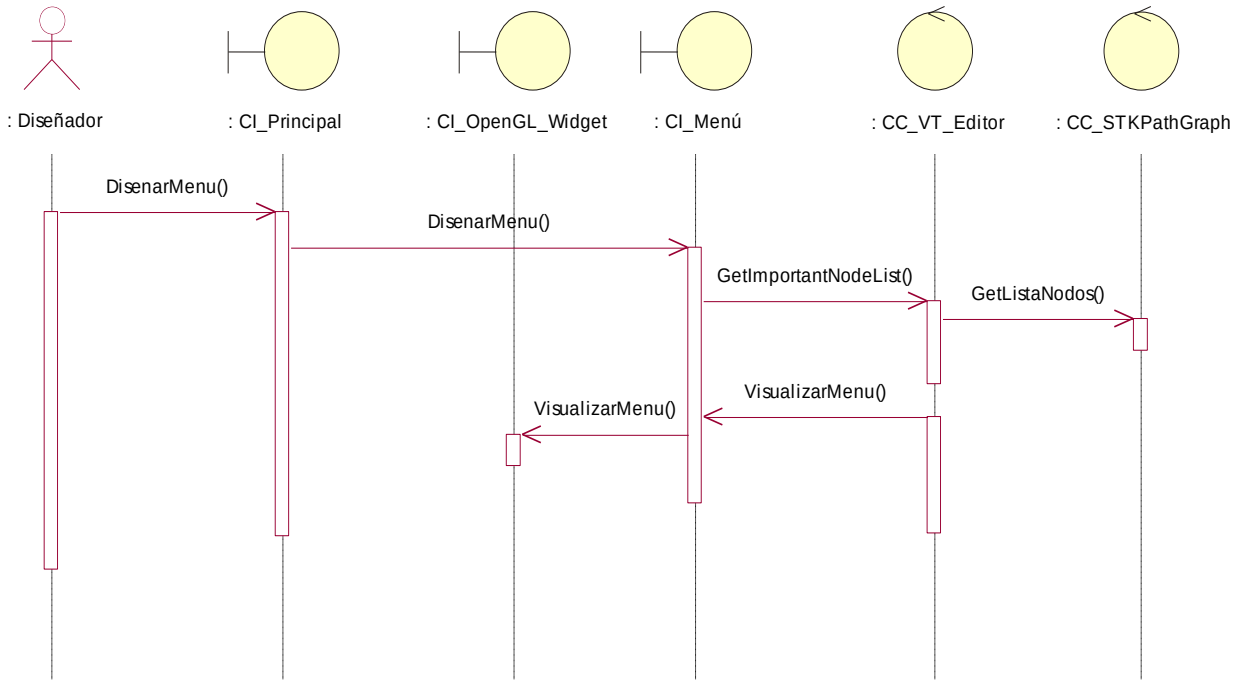


Figura 29. Diagrama de secuencia del caso de uso “Diseñar Menú de Navegación”

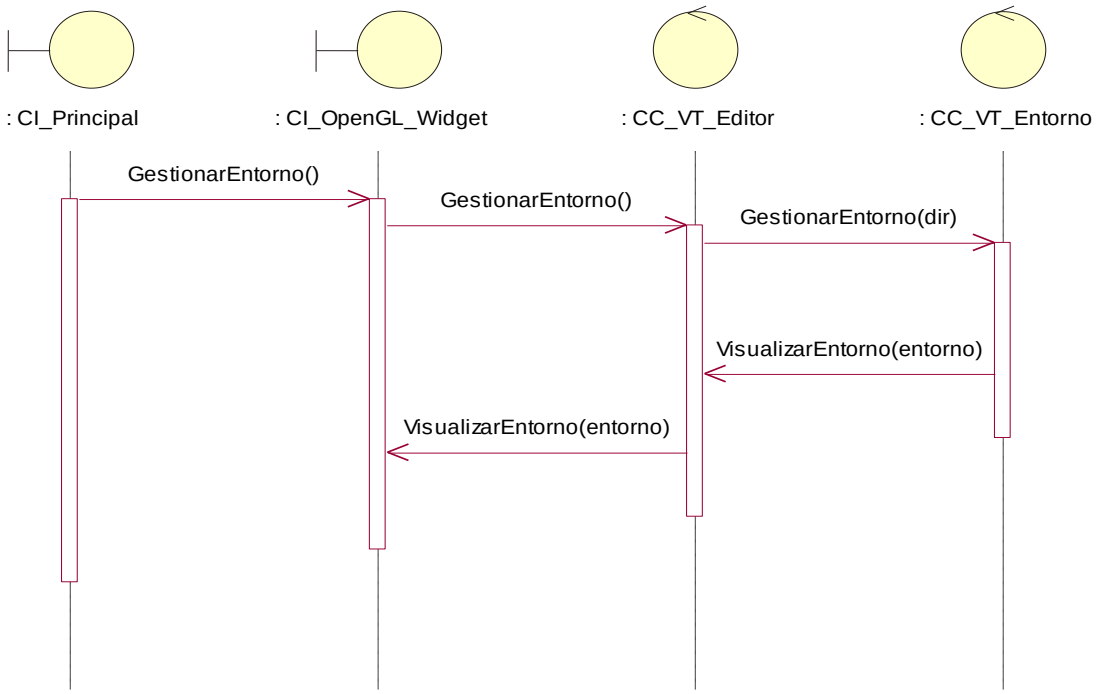


Figura 30. Diagrama de secuencia del caso de uso "Gestionar Entorno"

Paquete “Gestión Contenido Digital”.

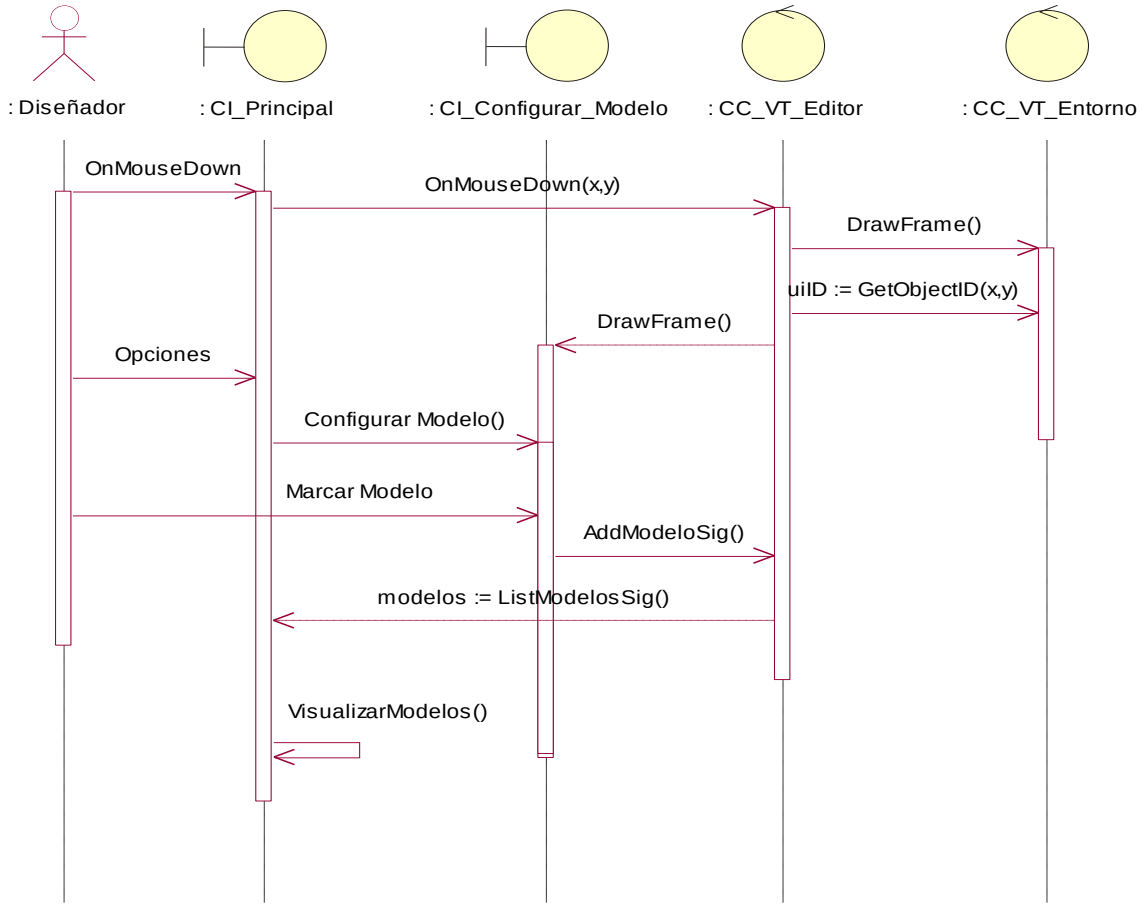


Figura 31. Diagrama de secuencia del caso de uso “Gestionar Modelos Significativos”

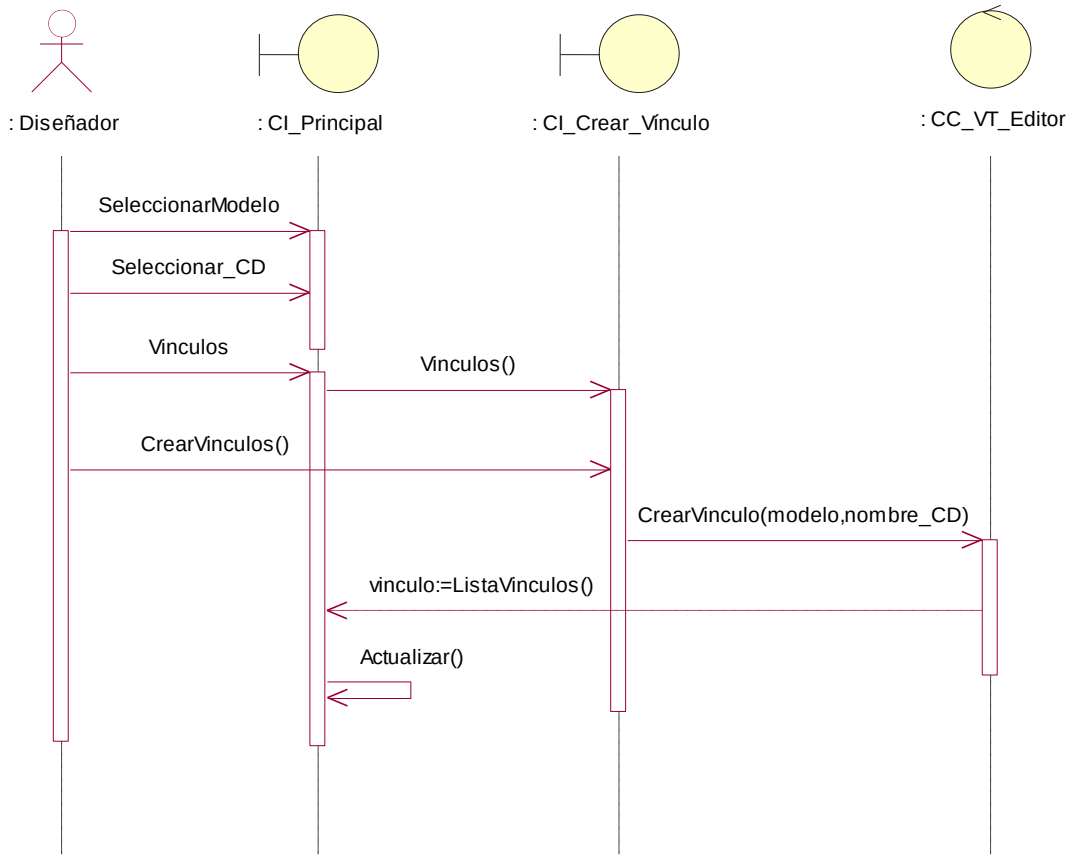


Figura 32. Diagrama de secuencia del caso de uso "Gestionar Vínculo de Contenido Digital".

4.4 Modelo de implementación

En el modelo de implementación se describe cómo los elementos del modelo de diseño que se deben implementar en términos de componentes y cómo se organizan estos de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizado; así como la dependencia entre los componentes. Esto se hace a través del diagrama de paquetes y del diagrama de componentes.

4.4.1 Diagrama de despliegue

El diagrama de despliegue analizado en este documento es muy simple, sólo incluye la representación de una PC, puesto que al ser la aplicación de escritorio todo va a estar en un solo nodo.

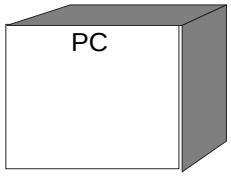


Figura 33. Diagrama de despliegue.

4.4.2 Diagrama de componentes.

A continuación se muestran los diagramas de componentes de los distintos paquetes, estos diagramas muestran la distribución de las clases en ficheros de código fuente. Utilizando los estereotipos de Racional Rose para C++.

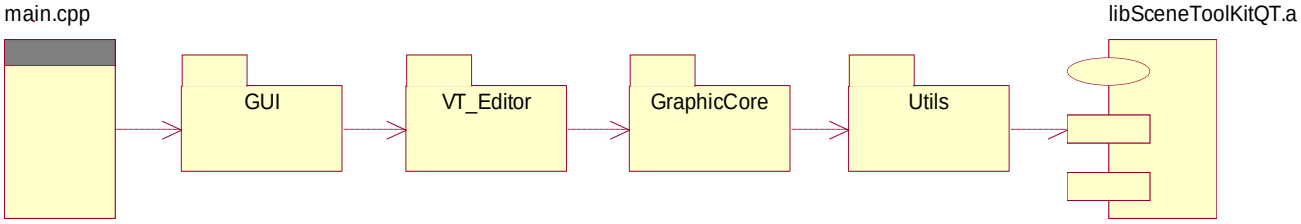


Figura 34. Diagrama de paquetes de componentes.

En esta imagen se refleja la relación entre el componente main.cpp de tipo “Main Program” con los distintos paquetes. El paquete “libSceneToolkit.a” es la librería dinámica de SceneToolkit generada con el compilador GNU G++.

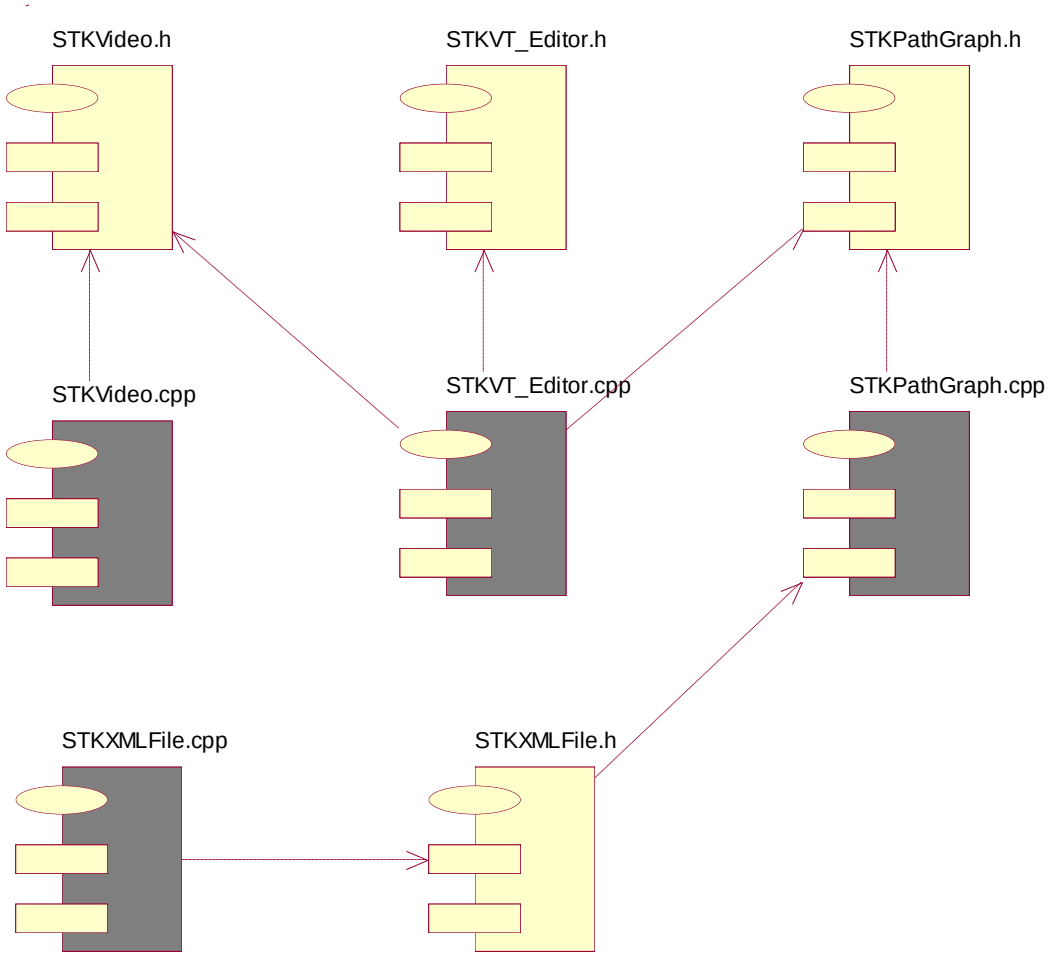


Figura 35. Diagrama de componentes, paquete "VT_Editor".

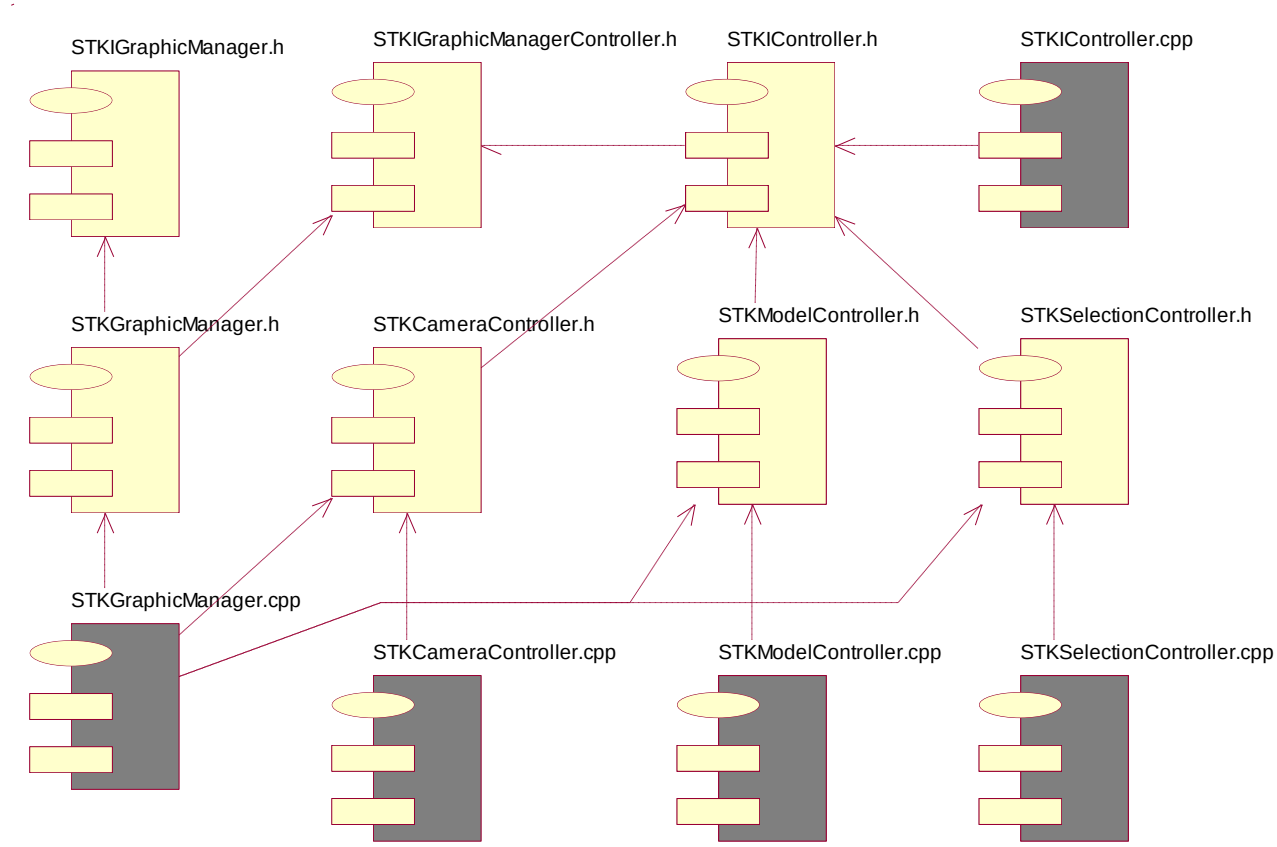


Figura 36. Diagrama de componentes, paquete "GraphicCore".

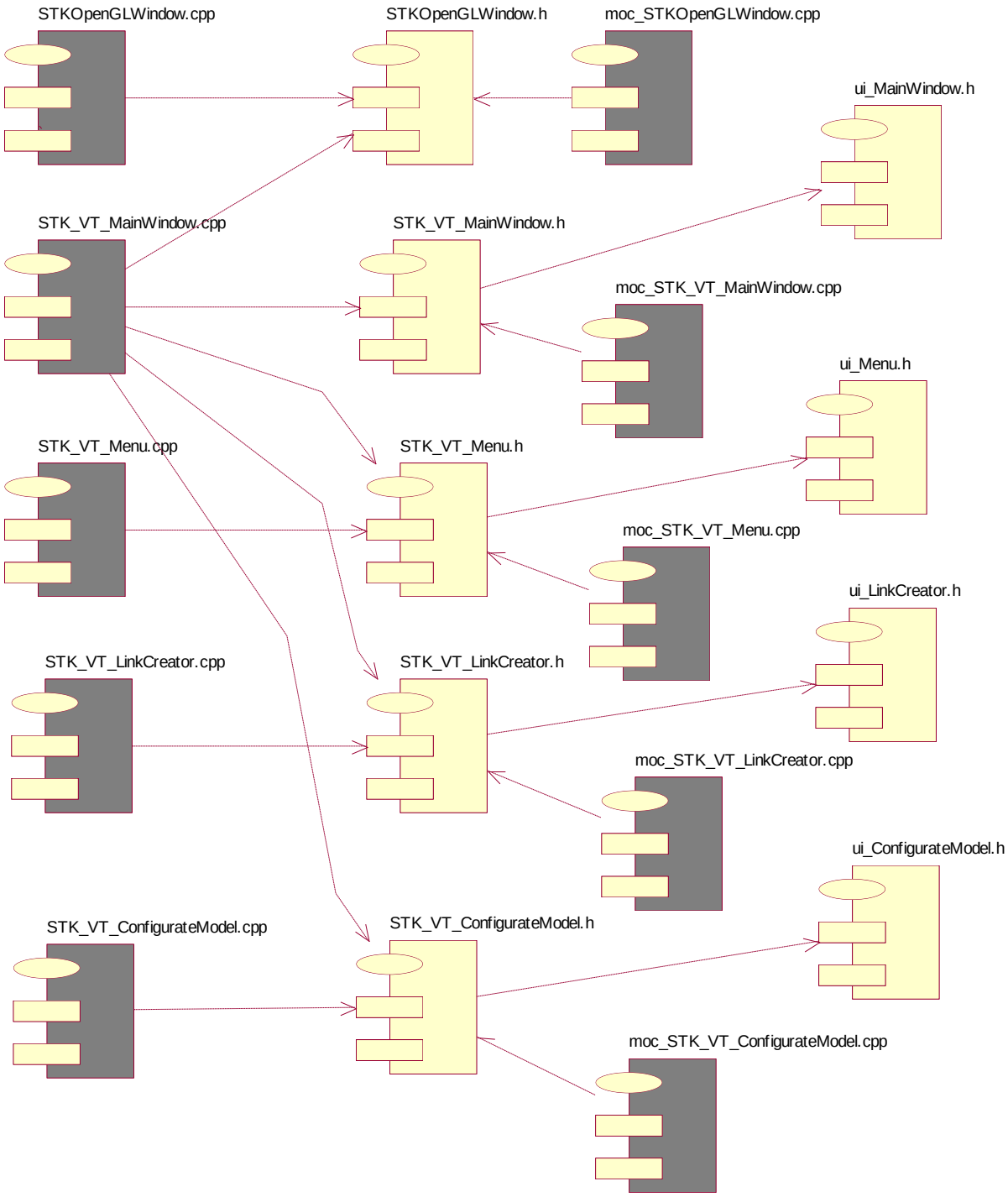


Figura 37. Diagrama de componentes, paquete "GUI".

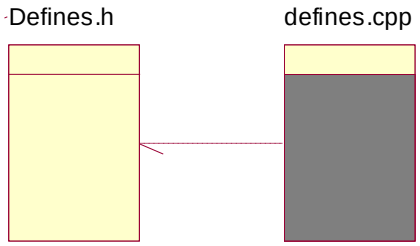


Figura 38. Diagrama de componentes, paquete "Utils".

Conclusiones

En este capítulo se explica cómo se interrelacionan las diferentes clases del sistema y como debería quedar la aplicación después de implementada. A partir de este punto ya se puede comenzar con la implementación del sistema ya que quedan sentadas las bases del análisis y diseño del mismo, por lo que en este momento se encuentra todo listo para pasar a la etapa de programación de los casos de uso. Como algo adicional que nos brinda la herramienta Rational Rose, ya es posible generar el código fuente de los componentes relacionados con los casos de uso.

CONCLUSIONES

Con la implementación del modulo de grafo de caminos que posibilita el movimiento de forma dirigida se hace más cómoda la navegación de los usuarios dentro del Paseo Virtual. La posibilidad de contar con una herramienta para la creación y edición de Paseos Virtuales es vital, ya que facilitaría el trabajo de los diseñadores de dicho proyecto, ahorrando así tiempo en el proceso de entrega de los productos. En este trabajo, en cumplimiento con el objetivo inicialmente planteado, se diseñó un sistema innovador que se adapta perfectamente a las necesidades requeridas por el motor gráfico SceneToolkit y brinda la posibilidad de crear, modificar y editar visualmente un Paseo Virtual.

Para el cumplimiento de los objetivos de este proyecto, en concordancia con las exigencias hechas por el cliente, se requirió primeramente hacer un estudio de varios temas, tales como: motores gráficos y sus características, algunas herramientas que en la actualidad prestan este tipo de servicio, tanto aplicaciones web como de escritorio, formatos de fichero para almacenar información de la escena 3D ya editada y principales características de la biblioteca gráfica SceneToolkit y del framework multiplataforma QT. Luego se realizó la captura de los requisitos funcionales y de los no funcionales, y la agrupación de los primeros en casos de uso del sistema. A continuación se procedió a las etapas de análisis y diseño utilizando los artefactos de RUP, donde surgieron y maduraron las clases, se plantearon los casos de uso y se creó el diagrama de componentes.

La herramienta se diseñó con la idea de que fuera portable a otros sistemas operativos debido a la utilización de funciones de la STL de C++, de la biblioteca de clases QT y del motor gráfico SceneToolkit, todos multiplataforma. Por su flexible arquitectura, el diseño es adaptable a posibles cambios, lo que posibilita que se puedan agregar nuevas funcionalidades en versiones posteriores sin hacer grandes modificaciones, en vistas a confeccionar un producto cada vez más completo y eficaz.

RECOMENDACIONES

Esta herramienta se diseñó con el objetivo de crear de forma visual grafos de caminos en un entorno, la configuración de los modelos en este y visualización de videos y animaciones asociadas a los modelos, utilizando la biblioteca SceneToolkit. Se recomienda continuar profundizando en estas ramas, para proporcionar más dinámica y realismo a las simulaciones en entornos virtuales. Se proponen las siguientes mejoras a la herramienta diseñada como trabajos futuros:

- Desarrollar los restantes módulos de la herramienta.
- Incorporarle un inspector de objetos para mejorar la gestión de modelos en la escena por parte del usuario.
- Fusionar la herramienta diseñada con la herramienta GraphBuilderSTK para un mejor funcionamiento de la misma.
- Incorporarle funcionalidades para hacer simulaciones, teniendo en cuenta el tipo de los caminos.
- Definir atributos que permitan simular comportamientos inteligentes en los nodos del grafo de caminos que haga uso de la información generada por la herramienta desarrollada.

BIBLIOGRAFÍA CITADA

1. **Akenine-Möller, T and Haines, E.** *Real-Time Rendering*, 2nd edition, June 2002.

2. **Seoane, Antonio.** *Herramientas de software libre para el desarrollo de aplicaciones de visualización 3D en tiempo real*. [cited: november 2008]
http://stuff.gpul.org/2007_graficos/doc/2007_JGRAF_03_3Dtreal.pdf

3. **Adobe**. *Adobe Atmosphere software*. [cited: november 2008]
http://www.adobe.com/la/products/atmosphere

4. *Investigación Aplicada de Realidad Virtual Inmersiva.* [cited: november 2008]
http://sig.utpl.edu.ec/sigutpl/Staftpro/realidad/realidad-inmersiva.PDF

5. **Tradky.** *Tradky Software*. [cited: november 2008]
http://www.tradky.com

6. **Look Twice**. *Look Twice Software.* [cited: november 2008]
http://www.looktwice.com.ar

7. **PaseoSoft**. *PaseoExpress.* [cited: november 2008]
http://www.paseosoft.com/products

8. **Paisajismo**. *AutoARQ Paisajismo*. [cited: november 2008]
http://paisajismo.asuni.es

9. **Apple**. *QuickTime VR*. [cited: november 2008]
http://www.apple.com/quicktime/technologies/qtvr

10. **Easypano**. *Tourweaver*. [cited: november 2008]
http://www.easypano.com/es/productsTw.htm

11. **Easypano**. *Panoweaver*. [cited: november 2008]
http://www.easypano.com/es/products.html

12. *Gráficos en computación. Motores Gráficos Ingeniería Informática. FIC UDC.*
[cited: november 2008]
http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Juegos/motores%20graficos/motoresgraficos.htm

13. **The Code::Block Team**. *The open source, cross platform, free C++ IDE*. [cited: november 2008]
http://www.codeblocks.org

14. **Grupo Soluciones Innova.** *Rational Rose Enterprise es el producto más completo de la familia Rational Rose.* [cited: november 2008] http://www.rational.com.ar/herramientas/roseenterprise.html

15. **Ingeniería de Software 1**. *Conferencia 1. Introducción al proceso de desarrollo de software.* Curso 2008-2009 p. 13-14. http://teleformacion.uci.cu

16. **Ingeniería de Software 1.** *Metodología de desarrollo de software.* Materiales complementarios. Curso 2008-2009 p. 2.

17. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* La Habana : Editorial Félix Varela, 2004.

18. **González, Yessy Cedeño and Gil, Ismael Hernández.** *Herramienta de creación de grafos de caminos para la biblioteca "SceneToolkit".* Universidad de las Ciencias Informáticas. La Habana : s.n., 2008. p. 24-25.

19. **Medina, Yasmany Cubela and Palau, Leonardo Nieblas.** *Interfaz Visual para la configuración de Entornos Virtuales desarrollados con la Herramienta SceneToolkit.* Universidad de las Ciencias Infrmáticas. La Habana : s.n., 2008. p. 137.

20. **Marrero, Ismael Viamontes and Garay, Sándor Labrada.** *Editor de pistas de carreras.* Universidad de las Ciencias Infrmáticas. La Habana : s.n., 2008. p. 22.

21. **López, Wendy García and González, Alexis Echemendía.** *Sistema de generación de ficheros para entornos virtuales*. Universidad de las Ciencias Infrmáticas. La Habana : s.n., 2007. p. 38-48.

22. **Ingeniería de Software 1.** *Conferencia 4. Fase de Inicio. Flujo de trabajo de requerimientos*. Curso 2008-2009 p. 2, 5, 14.

BIBLIOGRAFÍA CONSULTADA

1**. Virtual, Proyecto Herramientas de Desarrollo para Sistemas de Realidad.** *SceneToolKit: herramienta básica para desarrollo de sistemas de realidad virtual v2.3.* 2007.

2. **López, Fernando Jiménez y Román, Yanoski Rogelio Camacho.** *Biblioteca gráfica para sistemas de realidad virtual.* 2004.

3. **Fernández., S. Bayarri y M.** *Virtual reality in driving simulation.* 1997.

4. **I. Herman, M.S. Marshall.** *GraphXML - An XML based graph interchange format.* 2000. p. 24. 1386-3681.

5. **Gamma, Erich, y otros.** *Design Patterns. Elements of Reusable Object-oriented Software.* 1998

6. **Booch, Grady.** *Object-oriented analysis and design with applications 2da Edición*. Santa Clara, California : s.n., 1998.

7. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James**. *Proceso Unificado de Desarrllo de Software.* 2000.

GLOSARIO DE TÉRMINOS

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

API: conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Blender: es un programa de creación de gráficos y animaciones 3D desarrollado por Blender Fundation.

Entorno virtual o mundo virtual: se trata de 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.

Framework: estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

MD2: Nombre del formato que da Id Software al juego Quake 2.

Motor Gráfico: componente principal de un video juego u otra aplicación interactiva con gráficos en tiempo real.

Path: camino, curso, ruta, senda, sendero, trayecto, vereda; (informática: trayecto de búsqueda).

Plataforma: Combinación de hardware y software usada para ejecutar aplicaciones.

Realidad Virtual: es un sistema o interfaz informático que genera entornos sintéticos en tiempo real, representación de las cosas a través de medios electrónicos o representaciones de la realidad, una realidad ilusoria, pues se trata de una realidad perceptiva sin soporte objetivo, existe sólo dentro del ordenador.

Render: proceso de generar una imagen desde un modelo. Los medios por los que se puede hacer un renderizado van desde lápiz, pluma, plumones o pastel, hasta medios digitales en dos y tres dimensiones.

Simuladores: es un aparato capaz de reproducir un sistema, los simuladores nos hacen vivir sensaciones que en realidad no están sucediendo.

OpenGL: es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

3D: Tres dimensiones.

3DS*:* 3D Studio (file format).

3D Max Studio: es un programa de creación de gráficos y animaciones 3D desarrollado por Autodesk Media & Entrtaiment (formalmente conocido como Discreet y Kinetix).

XML: Extensible Markup Language, lenguaje de marcas extensible.

Índice de Figuras

Índice de Tablas