Universidad de las Ciencias Informáticas

Facultad 5

Editor de calles 3D

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor(es): Sara Iris Díaz Terán.

Darluin Domínguez Cuní.

Tutor: Ing. Alexis Echemendía González.

Ciudad de la Habana, junio 2010

"Año del 50 Aniversario del Triunfo de la Revolución"

Declaración de Autoría.

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Sara Iris Díaz Terán	Darluin Domínguez Cuní
Autor	Autor

_____

Ing. Alexis Echemendía González

Tutor

Datos de Contacto.

Nombre y Apellidos: Ing. Alexis Echemendía González.

Edad: 25.

Ciudadanía: Cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informática.

Categoría Docente: Profesor.

Ocupación actual: Jefe de líneas de imágenes y videos basados en rendering.

E-mail: aechemendia@uci.cu

Agradecimientos.

A la persona que le debo mi vida y mi corazón, por todo su tiempo de apoyo, de comprensión, por ser una verdadera amiga, incondicional, por entregar su amor sin límites, por sus cuidados, a mi madre querida le doy las gracias por confiar en mí siempre, sin duda alguna, por estrechar sus brazos cuando más lo necesito, por estar presente cuando más falta me hace, por escucharme y darme la confianza de que yo puedo decidir, por todo eso y mucho más gracias mamita.

A mi papá, por ser mi inspiración y mi modelo a seguir, por cuidarme y quererme como lo hizo, por enseñarme y educarme, por darme la oportunidad de vivir los primeros 16 años de mi vida más felices por estar él, por su dedicación hacia mí y mi hermano, gracias papi.

A mi hermano querido por quererme como lo hace, por confiar en mí, por cuidarme.

A mi tutor por preocuparse y ayudarme, por ser más que tutor, amigo.

A Tomy, por soportar mis melancolías, por aconsejarme, por estar a mi lado cuando más falta me hace el apoyo de una persona, por darme la oportunidad y demostrarme que en él puedo confiar, por brindarme su ayuda cuando de veras me hace falta, gracias.

A las personas que de una forma u otra me han brindado su ayuda, Karen, Hectico, Leydi, Arais, Lianet.

A las muchachitas del apto, Yailin, Yosly, Adys, Rossemary, Yami, gracias.

Al piquete de Marliuby, Yamila, Yailet.

Sara Iris Díaz Terán.

Agradezco principalmente a mis padres que me dieron la vida, fuerzas, educación y confianza: A mi madre quien con paciencia y firmeza siempre me colocó en el camino correcto, mi padre aunque un poco lejos siempre me apoyó en todo y asignó su aporte técnico. A Miguel Ángel que aunque no llevo su sangre ha luchado a mi lado desde los 11 años.

A mi abuela materna quien siendo mi segunda madre siempre estuvo cuando la necesité; a mi abuelo materno que aunque no se encuentra en este mundo siempre está a mi lado en cada paso que doy.

A mi familia paterna, mis abuelos, tíos, hermanos y primos.

A mi novia por la paciencia que ha tenido conmigo y la exigencia mostrada (igualita a mi madre) para la realización de este trabajo.

A nuestro tutor por su paciencia y por haber encontrado la solución en el momento preciso.

A mi compañera de tesis quien ha tenido la mayor de las paciencias conmigo (excesivamente paciente diría yo) siempre dedicada y preocupada por salir adelante.

Agradecimientos especialísimos a personas que sin ellos no hubiese estado hoy aquí presentándoles este trabajo, es decir, a mis amigos y bautizado por mis cotutores Lannie, Alexis, Hardy, Tomás, Héctor, Sanjony y Marino.

A todo aquel con el que compartí un grupo docente especialmente a los que dicen llamarse del "Cartel", así como a todo mi equipo del doble, a las "Ladronas" lideradas por la sicóloga más loca del mundo siempre mostraron una alta preocupación.

A mi gente de la Lenin que nunca olvido y con los cuales aún comparto.

A mis suegros por siempre estar preocupados.

A mi equipo de baloncesto que ha sido el mejor que he tenido, así como los que hoy no pueden estar aquí pero siempre han formado parte de mí, Naivy, Randi, Bazán, Ailyn, Yoana, Karen, Osmany y Pinillo.

Se que me quedan algunos pero no por eso son menos para mí, en fin muchas gracias a todos ellos.

Darluin Domínguez Cuní.

Dedicatoria.

A mi queridísimo papá, que aunque no esté hoy conmigo, sé que me está observando.

A mi mamá, por apoyarme en todo momento y alentarme.

A mi hermano, por confiar en mí.

Sara Iris Díaz Terán.

Dedico este trabajo especialmente a mi familia y a todos mis seres queridos que alguna vez confiaron en mí, ya sea en este mundo o en el otro.

Darluin Domínguez Cuní.

Resumen.

El desarrollo de las tecnologías está girando muy directamente sobre el crecimiento de la realidad virtual, esta provee considerables mejoras al ámbito del diseño 3D, permitiéndole a desarrolladores un grado mayor de realismo. Muy relacionados a este campo de diseño 3D se encuentran los videojuegos, la edición de calles es uno de los elementos fundamentales en la creación de videojuegos o aplicaciones 3D y se logran gracias a los editores de calles, que representan la herramienta que utilizan los diseñadores para editar nuevos sistemas de viales a los videojuegos.

En la facultad 5 de la Universidad de Ciencias Informáticas (UCI), se encuentra la línea de producción de diseño, donde no existe una herramienta propia que edite calles 3D, por lo que se determinó que era necesario el desarrollo de una aplicación que permita la realización del proceso de edición de calles 3D. El presente trabajo propone y lleva a cabo el desarrollo de una aplicación que permite cargar calles en formato .mesh que son exportadas por el software 3DMax, herramienta utilizada actualmente por la línea para editar las calles, las cuáles son utilizadas por el editor para crear un circuito de carreteras, permitiendo que el proceso de edición sea fácil para los diseñadores y lograr que el producto esté listo en menos tiempo.

Para alcanzar una aplicación óptima se realizó un análisis de formatos de ficheros 3D y la posterior selección de uno, así como de los motores gráficos más usados en la actualidad para seleccionar de estos el adecuado, ya que para la implementación de la herramienta es necesario un producto especializado en la visibilidad y la gestión de mallas 3D.

Como resultado se obtuvo una herramienta que permite realizar el proceso de edición de calles 3D, así como generar un fichero de configuración de estas, garantiza además un proceso rápido y fácil de implementar por los diseñadores de la línea de producción de diseño de la facultad 5 de la Universidad de las Ciencias Informáticas.

Palabra clave: realidad virtual, editores de calles, herramienta, calles 3D, mesh , ficheros, motores gráficos, mallas 3D.

Índice.

Introducción.

Con el avance de las tecnologías y las comunicaciones, gracias al interés y ansia del ser humano de crecer culturalmente y crear un modo de vida mejor para todos, han surgido nuevos conceptos y definiciones, que actualmente están relacionados y son utilizados muy directa o indirectamente por la comunidad que está inmersa en este crecimiento tecnológico. En el plano de la informática, ha nacido la rama de la realidad virtual, fenómeno que se desarrolla en un espacio ficticio y que se percibe a través de los sentidos. La realidad virtual es un sistema interactivo que crea una ilusión de realidad a través de un mundo tridimensional artificial.

Actualmente la realidad virtual se ha ganado un lugar en diferentes esferas de la vida, cuenta con diversas aplicaciones como son en la medicina, en la educación y con fines de entretenimiento, este es el campo más habitual y conocido por todos, por medio de este se puede disfrutar de videojuegos 3D en casas, ordenadores y videoconsolas. Desde que los videojuegos se han incluido a la vida como el entretenimiento fundamental para la comunidad, estos han experimentado un considerable desarrollo, es un aspecto muy importante para los mismos la confección de calles o los sistemas de viales que conforman una ciudad, sin importar el tipo de videojuego que sea, pues necesitan de un entorno o mundo donde los personajes u objetos tengan la oportunidad de moverse, de interactuar entre ellos, es decir donde se desarrolle el juego.

En la industria del software existe gran variedad de editores que se encargan de crear entornos para videojuegos o aplicaciones 3D, estas tienen características similares, permiten cargar y editar calles entre otras funcionalidades, por más que se necesiten herramientas que cumplan con estas características, existe el problema que son generalmente accedidas por empresas que son quienes cuentan con el presupuesto para obtener una de estas y muchas veces no se cuenta con el personal calificado para trabajar con las mismas, se puede citar como ejemplo de estas herramientas al 3D Max, otro software de diseño 3D que necesitan muchas empresas o la comunidad en general, por sus características de modelado, pero es una herramienta costosa, esto ha traído como consecuencia que las empresas decidan crear sus propias aplicaciones para satisfacer las necesidades que existen.

Cuba centrando su esfuerzo en el desarrollo de la industria del software, y a pesar de la situación económica con que cuenta, ha comenzado a dar sus primeros pasos en este campo. Con el objetivo de

incluir a Cuba en el mercado del software, se tiene en cuenta como un espacio en la misma el desarrollo de videojuegos.

La Universidad de Ciencias Informáticas (UCI), cumple un papel importante en esta industria, donde se aplican novedosos sistemas y concepciones de enseñanza, para la formación y superación de profesionales de la informática, la producción de software y servicios asociados para la industria nacional y la exportación, está organizada por diez facultades, las cuales tienen perfiles determinados. La facultad 5 está compuesta por Líneas de Producción, las cuales se dedican a la representación de objetos a través de la realidad virtual, esta provee un conjunto básico de primitivas para el modelaje geométrico tridimensional y tiene la capacidad de dar comportamiento a los objetos y asignar animaciones.

En la Línea de Producción de Diseño de la facultad 5 se lleva a cabo el desarrollo de entornos virtuales para cualquier tipo de videojuego o aplicación 3D. Uno de los aspectos para desarrollar los videojuegos son los entornos que están formados por artefactos y uno de ellos son los sistemas de viales. La elaboración de sistemas de viales es un proceso difícil y engorroso debido a la carencia de una herramienta que se dedique específicamente a este tipo de trabajo y que se ajuste a las necesidades de la Línea. Debido a esto se necesita una herramienta con una interfaz amigable que permita editar sistemas de viales 3D en menos tiempo.

A partir de dichos problemas surge como **problema científico**:

¿Cómo agilizar el proceso de elaboración de un entorno de calles 3D?

Por lo que se define como **objeto de estudio:** herramientas de desarrollo para la edición de calles 3D y donde el **campo de acción** lo constituye: proceso de edición de calles 3D.

Para darle solución al problema existente surge como **objetivo general:** elaborar una herramienta para la edición de calles 3D.

Se hace necesario realizar las siguientes **tareas** para darle cumplimiento al trabajo:

- Caracterizar los editores 3D más usados en la actualidad para confeccionar las características de la nueva aplicación.

- Caracterizar formatos de ficheros 3D para seleccionar el más adecuado.
- Caracterizar algunos de los motores gráficos existentes en la actualidad y librerías gráficas para la posible selección de alguno de ellos.
- Determinar requisitos funcionales y no funcionales para obtener una visión del sistema a desarrollar.
- Definir casos de uso con sus respectivas descripciones para el posterior diseño.
- Caracterizar patrones de diseño para seleccionar el más adecuado.
- Desarrollar una herramienta para la edición de calles 3D.

En el proceso de desarrollo de la investigación científica se emplearon **Métodos Teóricos**:

- **Análisis Histórico-Lógico** porque permite constatar teóricamente cómo ha evolucionado y desarrollado el proceso de edición de calles 3D.
- **Analítico sintético:** Permite hacer un estudio de la documentación y la teoría referente al objeto de estudio.
- **Genético:** Permite determinar el campo de acción del objeto a estudiar en esta investigación.
- **Modelación analógica:** Permite crear una propuesta de cómo realizar esta herramienta.

Organización del documento:

El presente trabajo está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, bibliografía referenciada, un glosario de términos y siglas, así como un índice de figuras.

Capítulo 1_Fundamentación Teórica: se realiza un análisis de las características de algunas herramientas que se utilizan para la edición de calles 3D, así como la justificación de las herramientas y metodologías usadas.

Capítulo 2_Solución Propuesta: muestra los principales rasgos de la herramienta que se propone desarrollar. Se procede al levantamiento de requisitos del sistema, haciendo un profundo análisis del objeto de estudio planteado, también se definen los casos de uso del sistema y la descripción de cada uno de ellos.

Capitulo 3_Diseño e implementación del sistema: se procede a la implementación del sistema basado en los resultados del análisis y el diseño elaborado previamente y se exponen los resultados obtenidos con la puesta en marcha del uso de la herramienta.

Capítulo 1: Fundamentación teórica.

Introducción.

En la industria de la informática gráfica existe una amplia variedad de editores, los cuales se encargan de crear y modelar objetos, cuerpos, entornos virtuales, para luego ser utilizados por otros programas, ya sean videojuegos o aplicaciones. Estas herramientas cumplen un papel importante en el desarrollo de aplicaciones que están relacionadas al tema de la realidad virtual.

En el presente capítulo se brindan algunas características de algunos de los editores que existen actualmente, para obtener una visión para el desarrollo de la aplicación, así como las características de formatos de ficheros 3D y motores gráficos, y se tendrá en cuenta la selección de uno de los ficheros y motores para utilizar en la posterior implementación.

1.1 Entorno virtual.

Un entorno virtual no es más que un mundo representado por medio de la realidad virtual, es decir la imitación de un mundo real mediante la creación o construcción de calles, edificios residenciales, hoteles, bancos, museos, hospitales, árboles, todos estos objetos que en conjunto conforman una ciudad, donde se interactúa mediante un personaje, teniendo reacciones físicas con el entorno, por ejemplo el espacio donde se desarrollaría un videojuego. Estos objetos son generados mediante editores de los cuales se abordará más adelante.

Figura 1: Entorno virtual.

1.2 Editores.

Los editores son herramientas que se utilizan para modelar, crear, diseñar objetos, ya sean personajes, calles, edificios, entre otros, incluso crear animaciones de los objetos que en conjunto conforman un entorno, para ser utilizados posteriormente por aplicaciones o videojuegos 3D.

1.2.1 Topo Cal.

Topo Cal es un potente programa CAD (Diseño asistido por ordenador) de cálculo topográfico, que tiene como función principal la creación de modelos digitales del terreno. Lee DXF y ficheros de puntos ASCII. Triangula, obtiene las curvas de nivel y hace una visión en 3D rotando la vista, exporta el MTD (Modelo Digital del Terreno) a DXF o 3dcara para renderizarlo con AutoCAD.

Permite creación y unión de polilíneas, scroll automático en los dibujos, el control total de las capas, y contiene gran diversidad de opciones gráficas.

Cambios recientes:

- Más potencia y rapidez en el cálculo del MDT y curvado con la optimización de rutinas internas.
- Puede arrancar todas las versiones de AutoCAD e integrarse con ellas a la vez como si fuera una aplicación vertical del mismo.(2)

1.2.2 Google SketshUp.

Google SketshUp es un programa para la creación, edición, y compartimientos de modelos 3D.

SketshUp es un conjunto de potentes herramientas para priorizar ante cualquier cosa la facilidad de uso y aprendizaje. Es ideal para poner “en papel” o boceto, una idea.

Todos los modelos SketshUp están formados por dos elementos: aristas y caras. Las aristas son líneas rectas y las caras son formas bidimensionales que se crean cuando varias aristas forman un bucle plano. Esta herramienta es perfecta para trabajar con rapidez y soltura en 3D. Si dispone de la versión profesional puede exportar las figuras geométricas que se hayan creado a otros programas como AutoCAD o 3D Max.

Pegando las distintas figuras geométricas del modelo para formar grupos, pueden crearse sub-objetos fáciles de mover, copiar y ocultar. Cuenta con un poderoso motor de sombras en tiempo real que permite realizar estudios de sombras precisos en los modelos. Las herramientas de la caja de arena de SketshUp permiten crear, optimizar y alterar el terreno tridimensional. Puede generar paisajes a partir de una serie de líneas de contorno importadas, añadir arcenes y valles. Puede importar archivos 3DS directamente a los modelos de SketshUp. Para facilitar la tarea de convertir planos, secciones, elevaciones e incluso todo el modelo, Google SketshUp Pro permite importar y exportar a los formatos DXF y DWG, además SketshUp Pro exporta una serie de formatos 3D como son: 3DS, OBJ, XSI, entre otros. La última versión de Google Sketchup incluye motores de animación, más de 100 materiales y otras texturas.

Entre sus opciones se encuentran:

- Cubo de pintura: que permite pintar el modelo con materiales como colores y textura.
- Escenas: permite guardar las vistas precisas del modelo para volver a ella después.(3)

La figura que se muestra a continuación es de un diseño gráfico desarrollado por la herramienta google sketshup:

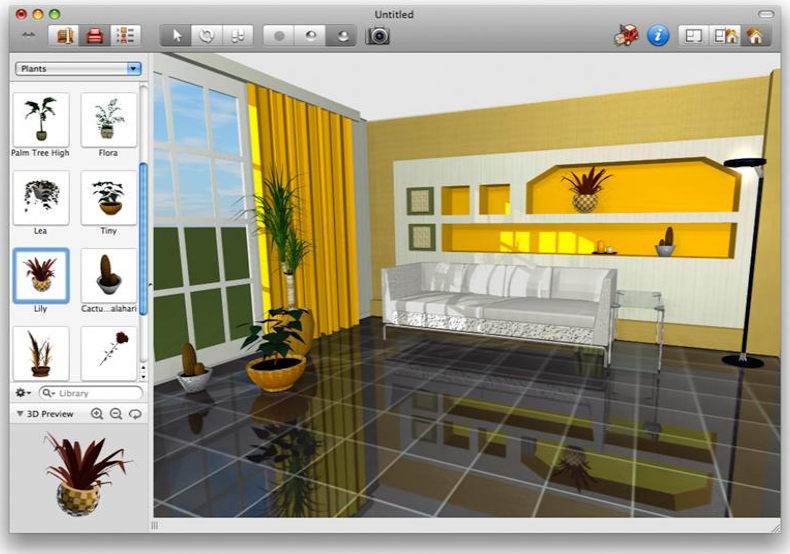


Figura 2: Google Sketshup.

1.2.3 Editores de calles 3D.

Los editores de calles 3D son las herramientas que se utilizan para diseñar sistemas de carreteras que pueden ser utilizados por videojuegos o aplicaciones 3D. En algunos casos los creadores construyen los videojuegos con sus propios editores, pero no siempre es así, a veces los editores son una parte separada del videojuego, y es cuando surge la necesidad de crear uno.

Entre los editores de calles más usados se encuentran:

- AutoCAD.
- Istram Ispol.

1.2.3.1 AutoCAD.

Es un programa de diseño, que se encuentra en el campo de la herramienta de Diseño Asistido por Ordenador (CAD). Gracias a su gran variedad, AutoCAD se ha convertido en un estándar general. Permite el trabajo en equipo y minimiza cualquier posibilidad de errores. Se destacan opciones para la edición de dibujos en 3D, como son sombreados, creación de ambientes, imágenes de fondo, entre otros. Permite dibujar de manera rápida, ágil y sencilla, sin las desventajas que tiene el dibujo cuando se hace a mano. (12) Presenta un conjunto de elementos para la creación de un dibujo:

Archivo: este menú permite seleccionar entre diferentes opciones: nuevo para la creación de un dibujo o abrir para trabajar sobre un archivo existente.

Las barras de herramientas básicas que deben estar activas son: dibujo, modificar y acotar; y en el menú principal las barras de capas y propiedades, que permiten definir color, tipos de líneas y calibres. Luego se trabaja con las opciones de dibujo, entre las cuales que se encuentran: línea, círculo, arco, rectángulo, polígono, entre otros. La selección del comando se realiza con el botón izquierdo del mouse, así como también la señalización de puntos en el área gráfica o la selección de entidades. Con el botón del lado derecho del mouse se confirma un dato o se cierra el comando, equivale a enter, también se utiliza para desplegar menús.

Posteriormente se trabaja con las herramientas de la barra modificar, que permite realizar cambios en las entidades ya dibujadas como: borrar, recortar, alargar, copiar, mover o rotar.

La última etapa del dibujo es el acotado, del cual hay que establecer sus características por el menú formato en la opción estilo de acotad, y la ubicación de los textos de notas, especificaciones del dibujo, identificación del gráfico y rótulos con el comando texto de la barra de herramientas de dibujo. (16)

AutoCAD presenta varias versiones y las mismas se crearon con un objetivo específico y siempre con la idea de mejorar el trabajo con esta herramienta, por ejemplo: AutoCAD 2006 permitió un mejor trabajo y más rápido, AutoCAD 2007 se centró en mejorar la capacidad de los diseñadores en la creación, navegación y edición de un diseño, AutoCAD 2008 se enfocó en aumentar la capacidad de los diseñadores para documentar los diseños.

Se ha llegado a la conclusión de que la mejor característica de AutoCAD es la versatilidad que le acompaña, pero a la vez es su mayor defecto, por ser tan versátil depende de plug-ins o aplicaciones específicas para cada especialidad.

Otra desventaja que le acompaña es el alto precio de este tipo de software, a continuación se muestra una tabla con los precios de las versiones más recientes de AutoCAD (12):

Tabla 1: Precios AutoCAD.

	Software	**Precio original**
	Autodesk AutoCAD 2008 (32 bit)	$3995.90
	Autodesk AutoCAD 2009	$3995.90
	Autodesk AutoCAD 2010	$3995.90

	Autodesk AutoCAD 2011	$3995.00

Debido al alto precio que tiene este software prácticamente quienes acceden a estos productos son las empresas. AutoCAD no es multiplataforma, por lo que en el 2002 la empresa Ribbonsoft creó QCad, el AutoCAD de Linux, es multiplataforma, la funcionalidad es la misma, está licenciado bajo GPL, aunque tiene una interfaz más fácil de entender. Pero esta herramienta también tiene sus desventajas, solamente puede editar archivos DXF, esto implica que hay que combinarlo con TrueConvert para trabajar con archivos producidos por AutoCAD. Está diseñado solamente para 2D, en el caso de 3D lo que tiene es una proyección isométrica llamada pseudo 3D. (4)

A continuación se muestra una imagen del programa de diseño AutoCAD, con una calle que ha sido diseñada por dicha herramienta:

Figura 3: Herramienta de diseño AutoCAD.

1.2.3.2 Istram Ispol.

A diferencia de otros programas, el entorno de trabajo de Istram Ispol está específicamente diseñado para permitir mecanizar los datos geométricos de los elementos del proyecto, obteniendo de esta manera resultados gráficos e informes de manera inmediata.

El entorno de trabajo permite la edición de la cartografía digital, proporcionándole al usuario las herramientas típicas de un editor CAD y otras especificaciones, además trabaja 100% con entidades 3D. La aplicación permite la importación y exportación de datos de diversa procedencia, incluyendo estaciones de topografía, dispositivos GPS y formatos digitales de las aplicaciones más extendidas, como son: *.cej (ejes de planta), * .vol (alzado y plataforma de Ispol), este último se compone de una serie de ellos que pueden utilizarse como composición del total: *.ssl (de suelo seleccionado/sobre excavación), * .stp (sección tipo de desmonte/terraplén), entre otros. Ispol permite diseñar desde la carretera más sencilla hasta la autovía más compleja.

En la siguiente figura se muestra una calle editada por la herramienta Istram Ispol:

Figura 4 Muestra de calles 3D.

La aplicación ofrece una variada lista de herramientas 3D que son aplicables al diseño de obra superficial y extracción de minerales. Los clientes diseñan y controlan canteras, escombreras y vertederos, y

embalses de cualquier tamaño. Permite trabajar con cartografía basada en curvas de nivel, rejillas 3D o con modelos digitales del terreno, los cuales pueden ser generados de manera automática. Algunas de éstas permiten además realizar tareas de análisis espacial, detectando y calculando intersecciones entre superficies, generando llanuras de inundación o calculando cuencas visuales. Es una herramienta compleja, exige un largo período de aprendizaje y es cara, su precio es de casi 15 000 euros.(6)

1.3 Características del 3D Max.

3D Max es un programa que permite crear desde cero imágenes estáticas o animaciones de objetos tridimensionales (1), es uno de los programas de animación 3D más utilizados. Dispone de una sólida capacidad de edición, una omnipresente arquitectura de plug-ins y una larga tradición en plataformas Microsoft Windows. 3D Max es utilizado en generalmente por desarrolladores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura.(11)

En 3D Max se encuentran las herramientas de modelado para crear las estructuras de objetos en tres dimensiones, y posteriormente aplicarles materiales, iluminación y animaciones para componer la escena, por esto se considera que el programa desata la creatividad de los diseñadores.

Uno de los procesos que caracteriza a esta herramienta es el modelado y es quizás el más costoso de la creación de imágenes y a la que el programa le dedica mayor cantidad de recursos. Para crear objetos se emplean formas simples ya definidas, como cajas, cilindros o esferas o se le permite al usuario la creación de formas con herramientas de modelado o incluso el modelado a mano. Las formas se manipulan usualmente como objetos de rejilla, muy parecidos a una estructura de alambre, esto determina sus aspectos.

Luego de la representación de objetos de rejilla de la escena 3D, se procede a cubrir los objetos con superficies que imitan distintos materiales, puede ser: madera, metal, mármol, entre otros. Lo que les da realismo a las imágenes no son solo las superficies que recubren los objetos, sino el efecto de volumen que se consigue mediante la iluminación. El programa permite situar puntos de luz sobre los objetos como si tuvieran superficies reales, definiendo brillo, sombra, transparencia y reflejos.

Para obtener la imagen final es necesario realizar una serie de cálculos más precisos de las superficies, las luces y las sombras. A este proceso se le denomina representación o renderización.

A continuación se muestra los precios que existen para las versiones del 3D Max (13):

Tabla 2: Precios 3D Max.

	Software	Precio original
	Autodesk 3Ds Max 2008	$3295.90
	Autodesk 3Ds Max 2009	$3495.90
	Autodesk 3Ds Max 2010	$3495.90
	Autodesk 3ds Max 2011	$3495.00

Este programa exige un ordenador potente, aunque puede ejecutarse en máquinas inferiores, pero con un menor rendimiento.(1)

Modelado no es más que la creación de una imagen o estructura de un objeto real a partir de un modelo que ya existe. Modelado se refiere generalmente a la creación manual de una imagen tridimensional (modelo) del objeto real, por ejemplo: en arcilla, madera, entre otros.

1.4 Formatos de ficheros 3D.

Los formatos gráficos no son más que archivos en los que se guarda la información que conforma un objeto 3D. Cada formato es independiente. La mayoría de estos posee una cabecera que le indica al programa que solicite las características de las imágenes que almacenan, por ejemplo: el color, tipo, entre otras. Cada formato tiene una organización propia de su estructura.

Algunos de los formatos gráficos más utilizados en el almacenamiento de información son:

- 3DS
- OBJ
- DWF
- MESH

1.4.1 3DS.

El formato 3DS es nativo de la aplicación original 3D Studio de modelado y animación. Es un formato binario y está estructurado por bloques y sub-bloques de información, esta estructura permite el acceso de forma rápida a los bloques de interés, sin necesidad de leer los bloques restantes.

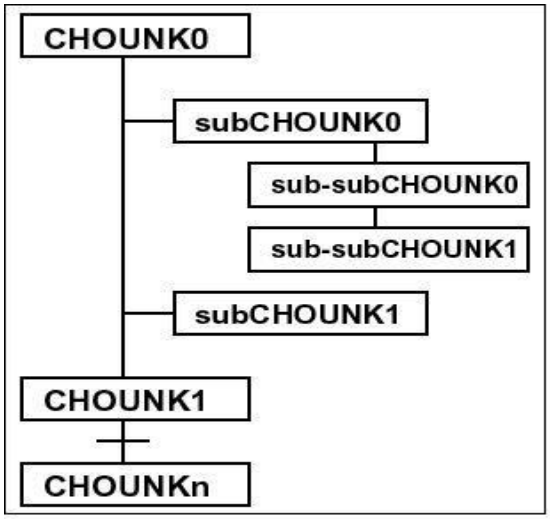


Figura 5: Formato 3DS.

Ventajas:

El formato 3DS contiene información adicional que indica las unidades originales utilizando un factor de escala. Esta información permite a una aplicación capaz de leer 3DS ajustar automáticamente el tamaño de los modelos 3DS al tamaño real.

Desventajas:

El formato 3DS no puede guardar nombres de archivos que excedan el límite de caracteres 8.3. Esto puede ser una limitación para aquellos que utilicen sistemas operativos modernos para gestionar proyectos que sean complejos o bibliotecas de archivos.

No permite almacenar una cámara ortográfica. Este formato simula a través de una cámara de perspectiva con un campo visual muy pequeño o una longitud de lente grande. El punto de visión se alejará todo lo que pueda para producir la misma anchura y altura del plano de proyección.(7)

La siguiente figura es una imagen que ha sido exportada por el formato 3DS:

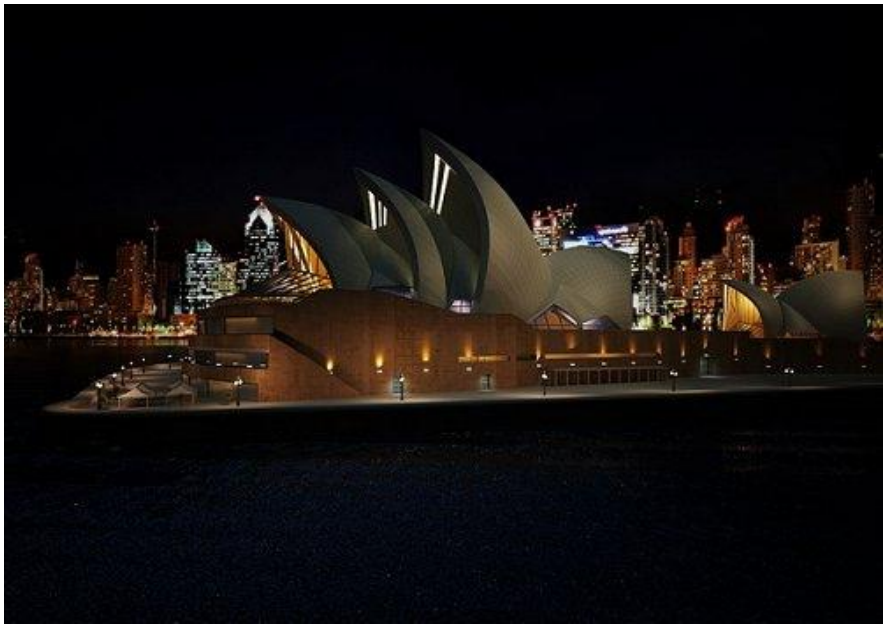


Figura 6: Modelo en formato 3DS.

1.4.2 OBJ.

Este es un formato de archivos de objetos 3D estándar. Es genial para objetos estáticos. Contiene geometrías poligonales que presentan puntos, caras y aristas, para definir un objeto o geometría de objetos de una forma libre que contengan superficies y curvas, aunque es más utilizado en objetos poligonales. No necesita encabezado, aunque es usual poner un comentario al inicio del fichero.

A continuación se muestra la estructura del formato .obj:

```
# Cuadrado de 2 x 2

  mtllib master.mtl

  v 0.000000 2.000000 0.000000

  v 0.000000 0.000000 0.000000

  v 2.000000 0.000000 0.000000

  v 2.000000 2.000000 0.000000

  vt 0.000000 1.000000 0.000000

  vt 0.000000 0.000000 0.000000

  vt 1.000000 0.000000 0.000000

  vt 1.000000 1.000000 0.000000

  # 4 vértices

  usemtl wood

  f 1/1 2/2 3/3 4/4
```

Ventajas:

Fácil manejo a la hora de cargar el fichero, pues contiene una estructura sencilla que permite guardar toda la información cualquier cantidad de listas de datos.

Desventajas:

El uso de este formato se centra en el trabajo con objetos estáticos, ya que no exporta características importantes como la matriz de transformación, los colores de vértice, entre otros.

1.4.3 Mesh.

Es un archivo informático que contiene información de cualquier tipo, es el formato de archivo que usa Ogre 3D (motor gráfico) para representar cualquier objeto 3D. A este formato binario lo define una única malla. Con el uso de una colección de polígonos crea una representación digital de modelos geométricos para formar una superficie con una textura (dibujo) en cada objeto.

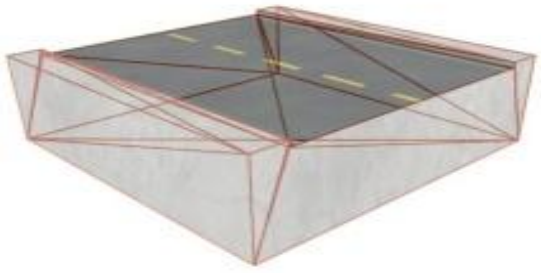


Figura 7: Archivo .mesh.

Ventaja:

Puede almacenar cualquier tipo de objeto.

Desventaja:

Codifica la información en forma binaria con el objetivo de almacenamiento y procesamiento en computadoras.

1.5 Ficheros de almacenamiento de datos.

Para seleccionar el fichero para almacenar la información de los modelos se realizó un estudio sobre algunos formatos de fichero para el almacenamiento de información, los más usados son:

El fichero de texto (txt), que es un formato muy utilizado en el mundo, pero tiene la desventaja de que solo se puede hacer lectura y escritura de los datos que contiene en su interior. Realizar una búsqueda de un elemento costaría un tiempo considerable.

Otro conocido es el formato de fichero XML, este a diferencia de otros sistemas que se usan para crear documentos tiene la ventaja de poder ser más exigente con la organización del documento, por lo que es más estructurado. Además permite hacer una búsqueda de un elemento determinado haciendo referencia al nombre de una marca. Ofrece portabilidad y utilización de la información a través de las distintas plataformas. Es un formato legible para personas y ordenadores. La especificación de documentos XML es simple, rápida, precisa y concisa. Este metalenguaje proporciona una forma de aplicar etiquetas para describir las partes que componen un documento, permitiendo además el intercambio de documentos entre diferentes plataformas.

1.6 Motores gráficos.

Un motor gráfico, es el componente de software principal de un videojuego o de otra aplicación que se ejecute en tiempo real. Su uso simplifica el desarrollo de la aplicación y permite que el juego pueda correr en varias plataformas. Es capaz de ahorrar tiempo y dinero a través del fomento de la reutilización enfocada a diferentes aspectos.

Ofrecen un conjunto de herramientas de desarrollo, además de componentes de software reutilizables, proporcionan funciones de renderizado 2D, 3D y de sprites. Suelen apoyarse en librerías/APIs gráficas como son OpenGL o DirectX. Se encargan de la visibilidad (buffers), el mapeado, la gestión de mallas 3D, entre otras cosas.

Los motores gráficos son productos altamente complejos y especializados, capaces de ahorrar tiempo y dinero a través del fomento de la reutilización enfocada a diferentes aspectos del videojuego. Algunos de los motores gráficos open source más conocidos son:

- Ogre 3D
- Crystal Space
- Irrlicht
- Reality Factory
- RealmForge GDK

De estos motores gráficos unos de los que más se destacan son Ogre 3D y Crystal Space:

1.6.1 Ogre 3D.

Ogre (Object Oriented Graphics Engine) es un motor de gráficos tridimensional multiplataforma. La principal ventaja de Ogre sobre otros engines 3D es que es un proyecto open source bajo licencia LGPL (Lesser General Public License), esto quiere decir que su uso es gratuito y apenas existen exigencias para su uso.

Figura 8: Ogre 3D

Fue diseñado bajo la filosofía de orientación a objetos, por lo que presenta una interfaz clara, intuitiva y fácil de usar, por lo que en Ogre se pueden hacer juegos o cualquier tipo de aplicación como simulación, corporativas o de investigación que requieran gráficos tridimensionales y que tengan poco que envidiar a los realizados por la mayoría de los motores del mercado.

Ogre utiliza una jerarquía de clases flexibles que permiten diseñar complementos para especializar la organización de la escena, esto permite que se pueda realizar la escena a su gusto. Es independiente de la API gráfica, se puede utilizar OpenGL o DirectX. (8)

La figura que se muestra a continuación demuestra el trabajo realizado con el motor gráfico Ogre 3D:

Figura 9: Ogre 3D.

1.6.2 Crystal Space.

Crystal Space es un motor gráfico 3D gratuito. Soporta iluminación coloreada, espejo, transparencia, superficies reflectantes, objetos animados, procesado de textura, sistemas de partículas, niebla volumétrica, transformaciones jerárquicas, soporte de pantalla de 8, 16 y 32 bits de profundidad de color, utiliza la librería OpenGl, es multiplataforma. Crystal Space está licenciado bajo la LGPL, que permite su copia, modificación y redistribución de las modificaciones.

Figura 10: Crystal Space.

Características generales:

- Es un motor genérico, comprensible y competente.
- Contiene módulos para gráficos 2D y 3D, sonido, detección de colisión.

A continuación se muestra una imagen en la que se utilizó Crystal Space como motor de render:

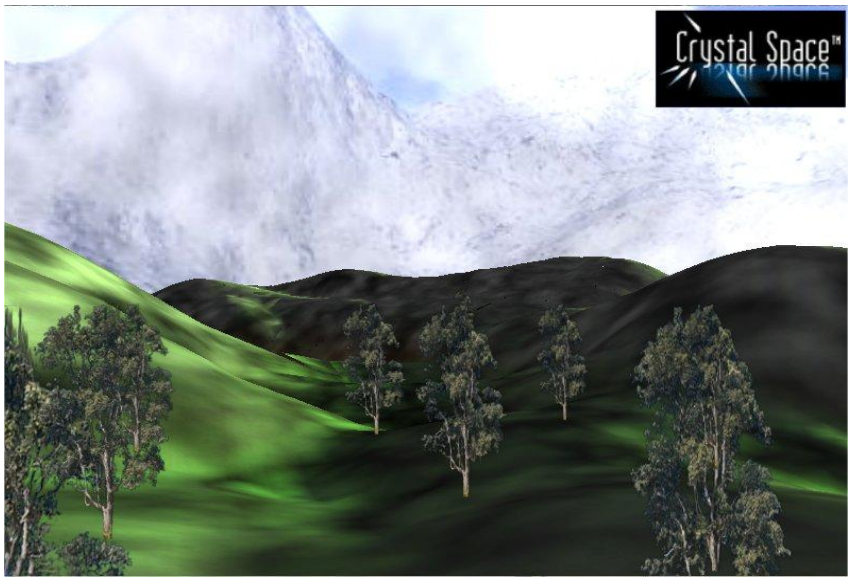


Figura 11: Crystal Space

1.7 Bibliotecas gráficas.

Una biblioteca gráfica es un software que genera imágenes en base a modelos matemáticos y patrones de iluminación, texturas, entre otros. Tiene como objetivo fundamental la independencia del hardware (tanto dispositivos de entrada y de salida) y de la aplicación (es accedida a través una interfaz única (al menos para cada lenguaje de programación)).

En el grupo de las bibliotecas gráficas pueden encontrarse estos ejemplos:

- OpenGL
- DirectX
- GKS
- PHIGS

Las bibliotecas gráficas más usadas son OpenGL y DirectX, que por ser el tema muy extenso solo se abordarán estas:

1.7.1 OpenGL

OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fundamentalmente OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener. Partiendo de ella, los fabricantes de hardware crean implementaciones, que son bibliotecas de funciones que se ajustan a los requisitos de la especificación (14). La interfaz consiste en un grupo de funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples.

Es usada ampliamente en realidad virtual, representación científica, visualización de información, simulación de cualquier tipo y en el desarrollo de videojuegos. Tiene dos propósitos fundamentales: ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme y ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL.

Características que OpenGL implementa:

- Primitivas geométricas: puntos, líneas, polígonos, imágenes.
- Codificación de color en modos RGBA (rojo-verde-azul-alfa) o de color indexado.
- Visualización y modelado que permite disponer objetos en una escena tridimensional, mover la cámara por el espacio y seleccionar la posición ventajosa deseada para visualizar la escena de composición, entre otras.

1.7.2 DirectX:

DirectX es un conjunto de interfaces de programación de aplicaciones (API) para el manejo de las tareas relacionadas con multimedia, especialmente en la programación de juegos y video, en las plataformas de Microsoft Windows.

DirectX consta de las siguientes APIs:

Direct3D: se utiliza en el procesado y la programación de gráficos en tres dimensiones (una de las características más usadas de DirectX).

Direct Graphics: para dibujar imágenes en dos dimensiones (planas), y para la representación de imágenes en tres dimensiones.

DirectInput: se utiliza para procesar datos del teclado, Mouse, joystick y otros controles para juegos.

DirectSound: para la reproducción y grabación de sonidos de ondas. Entre otras.

Consideraciones parciales.

De forma general en este capítulo se realizó un análisis de los formatos más usados en el mundo, dando a conocer las características, ventajas y desventajas de cada uno de estos, además abordamos el funcionamiento de diferentes editores de calles 3D, así como las herramientas necesarias para el desarrollo de estas aplicaciones.

Capítulo 2: Solución Propuesta.

Introducción.

En este capítulo se da a conocer una propuesta a desarrollar con el objetivo de cubrir los problemas analizados anteriormente y correspondiente al estudio realizado se determinará la vía más adecuada para lograr la solución planteada.

A continuación de los conceptos estudiados en el capítulo anterior se propone una solución a la problemática, desarrollar una aplicación que trabaje solamente el diseño de calles 3D para facilitarle la tarea a quien desee realizar esta actividad, para asegurar un mejor funcionamiento de la aplicación se determinan las herramientas y lenguaje de modelado que beneficien el desarrollo de la misma.

2.1 Modelo de Dominio.

El modelo de dominio es una representación visual de los objetos que están relacionados con el proyecto, es decir un diagrama con los objetos que existen en el proyecto y las relaciones entre ellos. Debido a que no se logra obtener un proceso del negocio con fronteras bien establecidas, se propone confeccionar un modelo de dominio que representan los objetos que participan en la aplicación y las relaciones entre ellos.

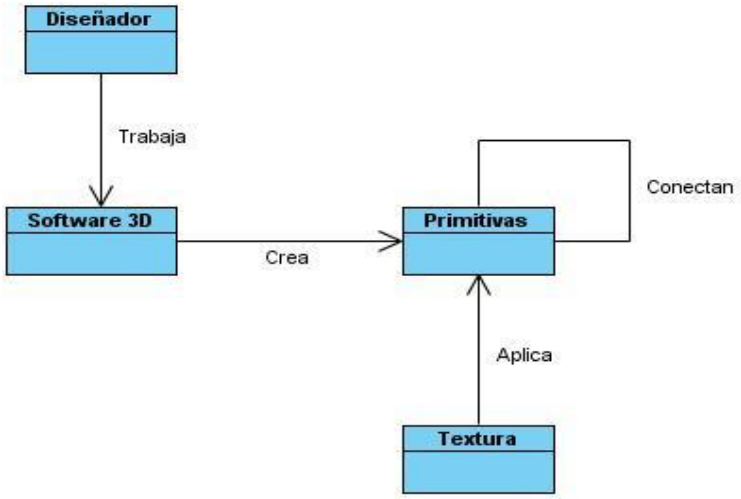


Figura 12: Modelo de dominio.

2.2 Descripción de las clases del dominio.

Diseñador: el diseñador es la persona que trabaja directamente con el software 3D.

Software 3D: se refiere a la herramienta que permite modelar o crear primitivas, a las cuales se le asignan propiedades como la textura, y las cuales se conectarán entre ellas para construir un sistema de calles.

Primitivas: es un objeto que ya está definido en el software, pueden ser una caja, cubo esfera, entre otros.

Textura: característica que se la asigna a un objeto tridimensional, en este caso una imagen.

2.3 Herramientas y metodologías para el desarrollo.

En la realización de este trabajo se investigaron las herramientas y metodologías posibles a utilizar. Por consiguiente se da a conocer la metodología utilizada y las herramientas seleccionadas con cada una de sus características.

Las metodologías utilizadas en el desarrollo de vida de un software garantizan el conocimiento del camino a seguir desde el comienzo de la implementación y antes, por lo que de esta forma se asegura un producto final con calidad. Con este objetivo se escogió la Metodología Racional Unified Process (RUP).

RUP es un proceso de desarrollo de software y que junto al Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada en el análisis, implementación y documentación de sistemas orientados a objetos. Es iterativo e incremental, está centrado en la arquitectura y guiado por casos de uso.

El RUP divide el desarrollo del software en 4 fases: inicio, elaboración, construcción y transición, entre las 4 fases se ponen de manifiesto 9 flujos de trabajo: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Instalación, Administración del proyecto, Administración de configuración y cambios, y Ambiente, donde los primeros 6 de ellos son flujos de trabajo de ingeniería y los últimos 3 de apoyo. El RUP unifica al equipo de desarrollo de software y lo mantiene enfocado en producir software operativo a tiempo, con las características y calidad requeridas.

Herramientas: La herramienta utilizada fue Microsoft Visual C++ Express Edition, para la documentación de esta el Visual Paradigm.

2.3.1 Microsoft Visual C++ Express Edition.

Visual C++ Express Edition es el entorno de desarrollo integrado (IDE) más adecuado, pues es soportado por sistemas operativos Windows creado por Microsoft. Es completamente gratuito, permite la creación de aplicaciones, sitios y aplicaciones web en C++ de la forma más sencilla. Presenta una potente y ordenada interfaz. Cuenta con una amplia colección de librerías, un completo sistema de conexión y utilización de bases de datos.

2.3.2 Visual Paradigm.

El Visual Paradigm es la herramienta que asegura la optimización del producto, debido a que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML le permite una rápida construcción de aplicaciones con calidad, mejores y con un mejor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además es una herramienta multiplataforma.

2.3.3 Qt.

Se consideró a Qt como la plataforma más adecuada para el desarrollo, ya que es una amplia plataforma de desarrollo que contiene clases, librerías y herramientas para la construcción de aplicaciones de interfaz gráfica en C++ que pueden operar en varias plataformas. Con Qt pueden desarrollarse ricas aplicaciones gráficas y que operen mejor que las nativas, ya que incluye nuevas tecnologías como OpenGL, XML, Bases de datos, y mucho más. Dispone de un gran número de herramientas que le facilitan la creación de botones, formularios y ventanas de diálogo con el uso del Mouse.

Le caracterizan tres grandes ventajas:

- Qt es totalmente gratuito para aplicaciones de código abierto.
- Las herramientas, librerías y clases están disponibles para casi todas las plataformas Unix y sus derivados (Linux, MacOS X, entre otras), así como para la familia de Windows.

- Qt tiene una extensa librería con clases y herramientas para la creación de ricas aplicaciones. Estas librerías son muy fáciles de usar y tienen una gran herencia de programación orientada a objetos.

2.3.4 Ogre 3D.

Se consideró el uso de Ogre 3D ya que muchos de otros motores aunque son impresionantes técnicamente, la falta de cohesión y del adecuado diseño en la documentación les impide un uso eficaz. Además la mayor parte de los motores están diseñados para un determinado estilo de juego. Ogre es diferente, prioriza el diseño frente a la acumulación de características. Las características se han considerado a fondo y son incluidas en el diseño general de la forma más elegante posible y siempre documentada. De esta forma se consigue que los elementos del motor formen siempre parte de un conjunto coherente.

Lenguaje: para desarrollar la herramienta se seleccionaron: C++ y UML. El C++ se escogió para realizar la implementación de la aplicación y el UML para modelar los artefactos de software en todo el proceso de desarrollo del mismo.

2.3.5 XML.

Se seleccionó para realizar la exportación del entorno, el fichero XML, debido a que Ogre no brinda la posibilidad de salvar varios objetos como un solo objeto, debido a esto es necesario el uso de un fichero para salvar la información de la posición de los objetos que se encuentran en la escena. XML a diferencia de otros ficheros que se utilizan para crear documentos es muy estructurado en cuanto a la organización del documento, es un formato legible para personas y ordenadores, la especificación de documentos XML es simple, rápida y precisa, además ofrece portabilidad y utilización de la información a través de distintas plataformas.

2.3.6 DirectX.

DirectX fue la selección más apropiada entre las bibliotecas gráficas estudiadas, ya que brinda la posibilidad de facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y video en la plataforma Microsoft Windows. Contiene una colección de API creadas para el procesado y programación de gráficos en tercera dimensión, para procesar datos del teclado, mouse, joystick y otros controles para juegos, para dibujar imágenes en dos dimensiones entre otras funciones.

2.4 Propuesta.

Se realizó una comparación entre los editores estudiados anteriormente y la aplicación tomando en cuenta las características de cada uno de ellos:

Características	Editor 3D	AutoCAD	Istram Ispol
Objetos prediseñados.	Contiene calles prediseñadas.	No.	No .
De la universidad.	Propia de la universidad.	No.	No.
Conocimiento del código fuente.	Si.	No.	No.
Costo.	Gratis.	Versiones (2008-2010) : $3995.90. Versión 2011: $3995.00	15 000 euros.
Tiempo que toma el diseño.	Rápido.	Rápido.	Prolongado.
Facilidad en el diseño.	Sencillo.	Sencillo.	Complejo.
Peso de la herramienta.	Ocupa poco espacio.	Requiere espacio.	Requiere espacio.

Figura 13: Comparación entre editores.

Como se puede apreciar en la comparación anterior se demuestra que estos editores presentan desventajas que impiden acceso total y utilización de los mismos, por estas razones es necesario

desarrollar una nueva herramienta, a la que se le definen funcionalidades similares a las de estos editores como: un menú en el que se puede seleccionar la opción Open para la inserción de un nuevo modelo de calle 3D, la aplicación no presenta las opciones de dibujo del AutoCAD: las líneas, círculos, polígonos, entre otros que es necesario aplicarles propiedades y modificaciones, a diferencia contiene una biblioteca con modelos prediseñados, que agiliza el proceso de edición del entorno, ya que no es necesario realizar el modelado de los objetos, pues es otro software el que se encarga de este trabajo, presenta opciones para eliminar, insertar, mover, rotar las calles de la escena, mediante la manipulación del mouse y el teclado.

Para la edición de modelos 3D, incluidas calles, se utilizan herramientas especializadas de diseño 3D, como son el 3D Max y Maya. Estas herramientas tienen como salida los modelos de calles 3D, que se exportan en formato .mesh y las texturas que se van a cargar en nuestra aplicación.

Al tener como entrada los modelos de calles 3D, anteriormente diseñados con el 3D Max, el motor gráfico (Ogre 3D) es fundamental en la herramienta, ya que permite la carga y almacenamiento en una biblioteca de los modelos 3D, el editor va a estar estructurado por una escena donde el usuario armará el entorno, el mismo tiene la posibilidad de seleccionar las calles de la biblioteca y una vez añadidas en la escena la aplicación le brinda la opción de modificar la rotación de las mismas o de cambiar la posición en la escena para acomodar y armar un entorno de viales lógico, posteriormente a la generación del nuevo modelo por la aplicación, se exportará con el fichero XML.

Para un mejor entendimiento del proceso que llevará a cabo el sistema se esquematizó el siguiente flujo del sistema:

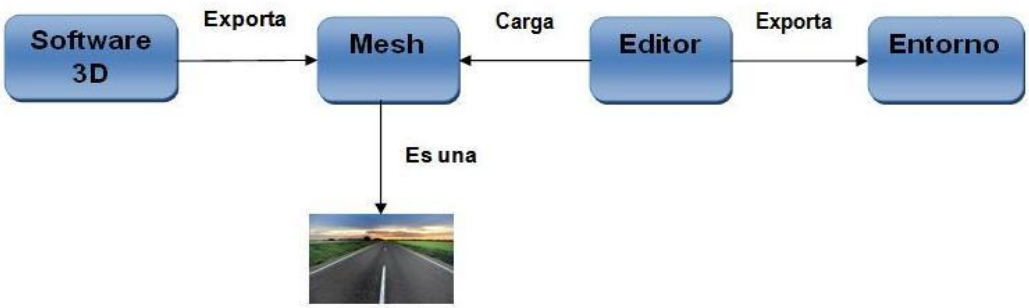


Figura 14: Flujo del sistema.

2.4.1 Restricciones del sistema.

Con el propósito de definir algunas condiciones para satisfacer las necesidades de funcionamiento del editor de calles se establecen las siguientes restricciones:

1. Los modelos 3D van a cargarse en formato .mesh.
2. Los formatos de textura soportados debes ser .jpg, raw, png.
3. Los elementos de calles que se exporten deben estar en el eje de coordenadas <0,0,0>.
4. Las unidades de medidas de los objetos a cargar deben estar en centímetros.
5. Las medidas de una calle estarán dadas por las siguientes medidas: 700 centímetros de ancho y 700 de largo.

2.4.2 Ventajas de la aplicación.

Con esta solución se establecen las bases para un editor más completo, que permita cargar objetos (calles) de cualquier tipo, que contenga calles predeterminadas, permitiendo un trabajo rápido, ágil y sencillo, y genere un entorno que será utilizado posteriormente por juegos, simuladores o aplicaciones 3D.

Con el estudio realizado en este trabajo quedó demostrada la existencia de herramientas de diseño, aunque el uso de estas es reducido debido a las desventajas que traen consigo, muchas de ellas son difíciles de adquirir por el alto precio que tienen y requieren de experiencia para su uso, por lo que la aplicación propuesta brinda las siguientes ventajas:

- Uso en la universidad.
- Se tiene completo conocimiento del código fuente.
- Presenta una interfaz clara y fácil de usar.
- Ocupa poco espacio.
- No hay que pagar para obtener la aplicación, ya que es gratis.
- Minimiza el tiempo de trabajo con la misma.

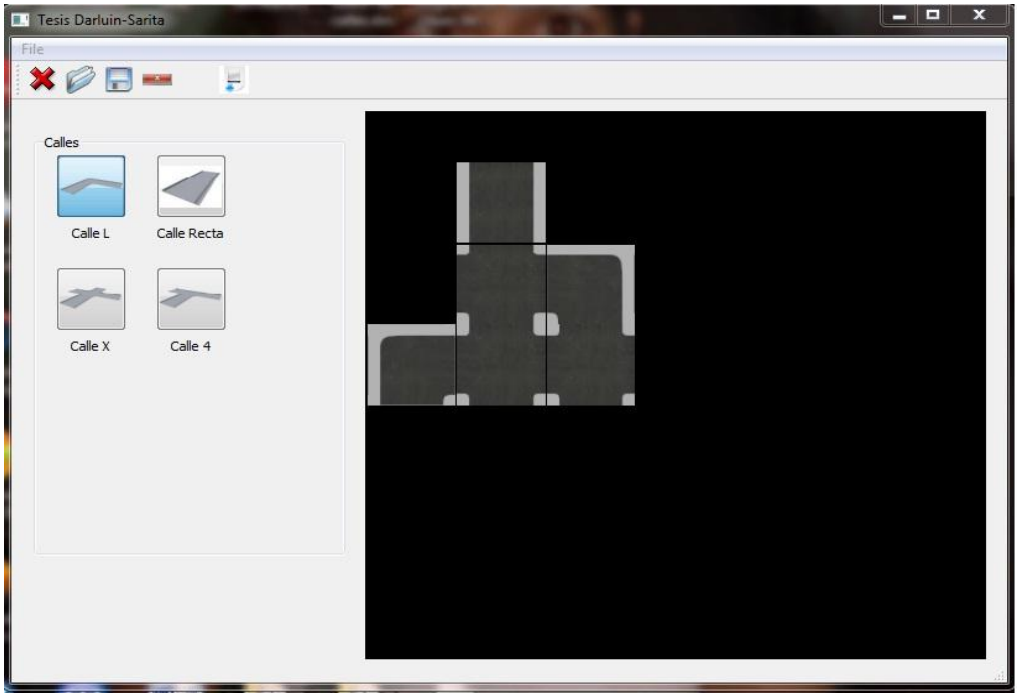


Figura 15: Prototipo de interfaz.

2.4.3 Necesidades del cliente.

Entre las necesidades que presenta el cliente se encuentran:

- Cargar calles: permite cargar un fichero del motor gráfico empleado.
- Crear un entorno virtual: se crea un sistema de viales mediante la relación o unión lógica de las calles.
- Exportar entorno: permite salvar el entorno con el formato XML.
- Notificar la exportación: un mensaje que se envía informando que se exportó correctamente el entorno.

Para utilizar correctamente la información de cada calle 3D, que debe ser cargada por nuestra aplicación, fue necesario llevar a cabo una investigación acerca de los diferentes formatos de ficheros 3D que son exportados por las herramientas de diseño 3D, como las que se describen en este trabajo, además se le realizó un estudio al formato XML, con el objetivo de utilizarlo en la exportación de nuestra aplicación.

2.5 Captura de requisitos.

A continuación se definen las funcionalidades del sistema, de esta manera se comienza con la captura de requisitos.

El sistema debe permitir:

RF1: Gestionar calles.

RF1.1: Insertar una calle en la escena.

RF1.2: Eliminar una calle de la escena.

RF1.3: Modificar la rotación de una calle.

RF2: Importar calles.

RF2.1: Insertar una calle en la biblioteca.

RF3: Exportar entorno de calles.

2.5.1 Requisitos no funcionales.

Las propiedades que debe presentar la aplicación para darle cumplimiento a los requisitos funcionales se mencionan a continuación:

Usabilidad: el sistema podrá ser usado por cualquier persona, no necesariamente debe tener experiencia para utilizar la herramienta.

Software: el sistema debe funcionar sobre sistemas operativos Windows versión XP o superior.

Requerimientos de Hardware: microprocesador Intel Pentium IV o superior, memoria RAM de 256 MB o superior, tarjeta de Vídeo 128 MB o superior.

Restricciones en el diseño e implementación: debe ser implementado con el lenguaje de programación C++. Microsoft Visual C++ Express Edition. Qt y el motor gráfico Ogre 3D.

Apariencia o interfaz externa: la aplicación presentará una interfaz atractiva para el usuario e interactiva, fácil de usar y cómoda con botones de acceso, además del menú que debe tener para seleccionar la acción que el usuario desee ejecutar.

2.6 Modelo del sistema.

2.6.1 Actores del sistema.

Los actores del sistema son roles que un usuario o usuarios llevan a cabo en algún momento, pueden ser otros sistemas con los que interactúa el sistema, estos estimulan al sistema con eventos de entrada o la respuesta de algún resultado que se produzca.

En este caso particular quien constituye el actor del sistema será el diseñador de calles, ya que hará uso del sistema, se muestra a continuación:

Tabla 3: Justificación de actores.

Actor	Descripción
Diseñador	El diseñador es el encargado de insertar una calle en la biblioteca o eliminarla si desea, así como armar el entorno, seleccionando de la biblioteca de calles algunos modelos e insertarlos en la escena, una vez ahí puede eliminar una calle o modificar la rotación de una de ellas y finalmente exportar el entorno.

2.6.3 Diagrama de Casos de Uso.

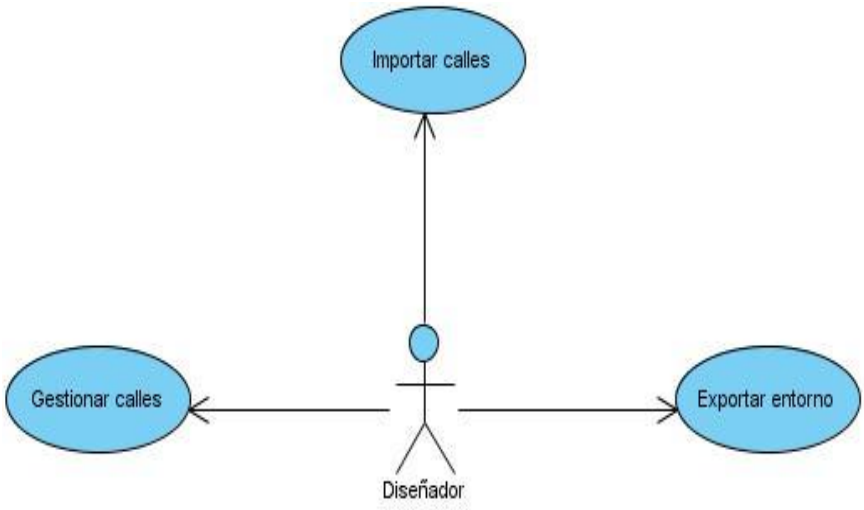


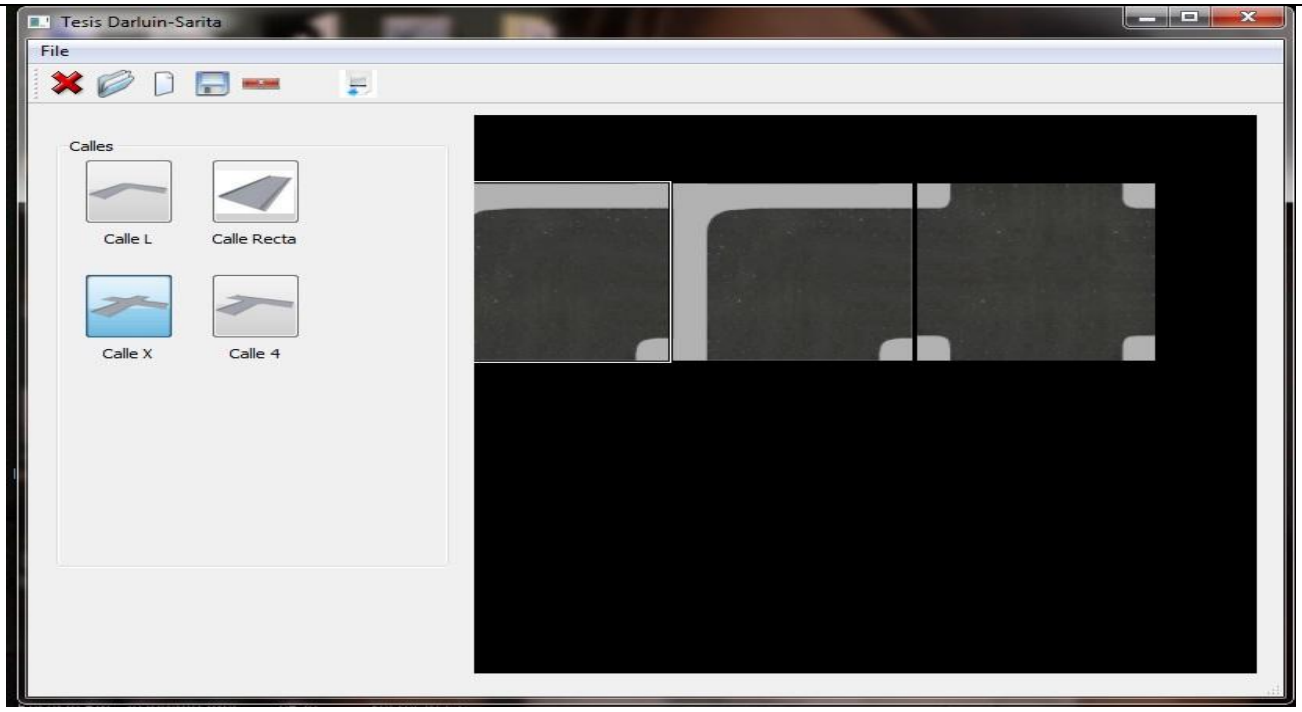
Figura 16: Diagrama de Casos de uso del sistema.

2.6.4 Especificaciones de los casos de uso del sistema.

Tabla 4: Especificación del CU Gestionar calles.

Casos de uso	Gestionar calles
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el diseñador selecciona una calle de la biblioteca de calles y la inserta en la escena, puede modificar la rotación de una o eliminarla de la escena, el caso de uso finaliza cuando el diseñador termina de construir el entorno de viales.
Precondiciones:	El sistema debe haber habilitado las funcionalidades con las que va a trabajar el diseñador.
Propósito:	El caso de uso se realiza con el propósito de crear un entorno de viales, a medida que el diseñador selecciona las calles de la biblioteca y las inserta en la escena, modifica la rotación de las mismas para obtener calles más lógicas o eliminarla de la

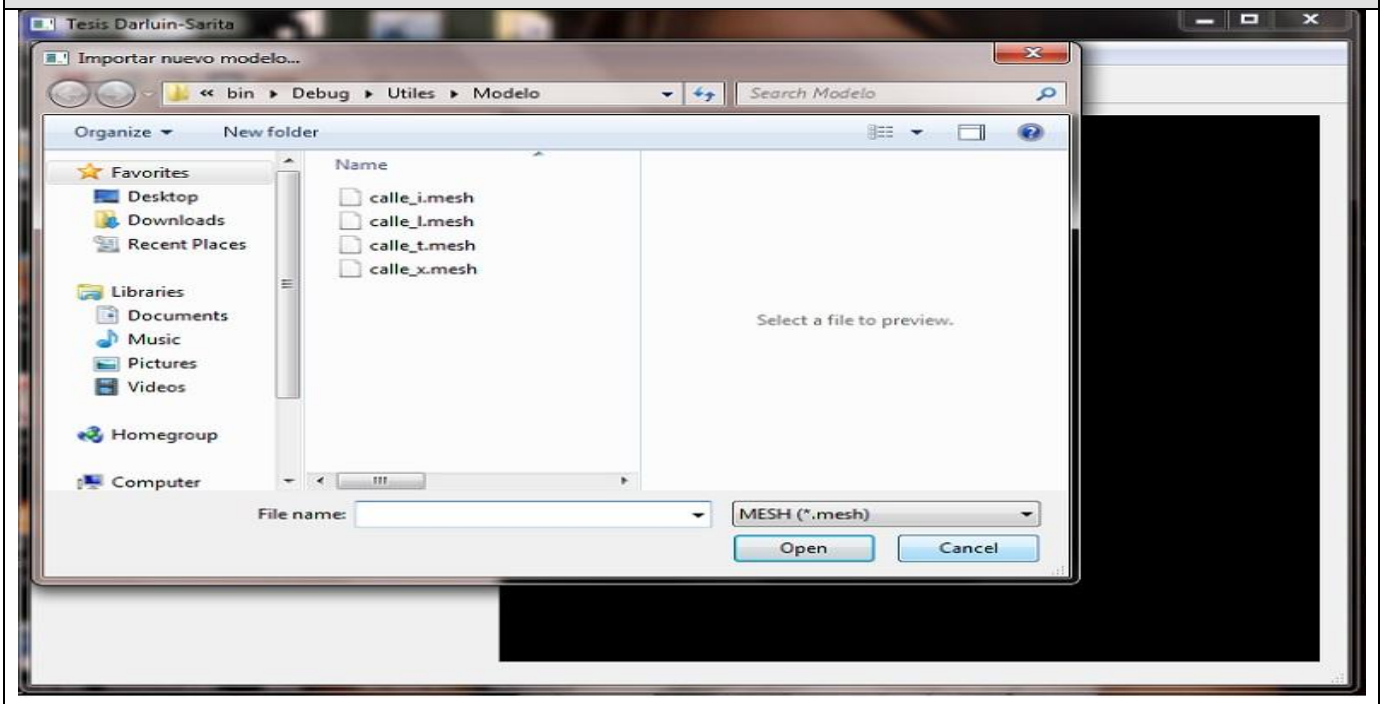
	escena.
Referencias:	RF1
Prioridad:	Crítico
Flujo normal de eventos.	
Sección “Insertar calles en la escena”.	
Acción del actor	**Respuesta del sistema**
1. El diseñador selecciona los modelos de calles de la biblioteca para insertar en la escena.	1.1 Espera que inserte la calle seleccionada en la escena.
2. El diseñador inserta la calle en la escena.	2.1 Guarda en memoria la posición y propiedades de la calle.
	2.2 Visualiza la calle en la escena.
Sección “Eliminar calle de la escena”.	
1. El diseñador selecciona una calle de la escena si desea eliminarla.	1.1 Espera que el diseñador oprima la opción “Eliminar”.
2. El diseñador selecciona la opción “Eliminar”.	2.1 Borra de su memoria las propiedades y posición en la escena de la calle.
	2.2 Elimina la calle de la escena.
Sección “Modificar la rotación de una calle”.	
1. El diseñador selecciona una calle de la escena en caso de querer modificar su rotación.	1.1 Espera que el diseñador ejecute alguna acción.
2. El diseñador presiona las teclas del teclado.	2.1 Modifica la rotación de la calle seleccionada.
	2.2 Guarda en memoria la nueva posición de rotación de la calle.
Prototipo de interfaz	



Pos-condiciones:	Se insertó, se eliminó una calle o se modificó la rotación de esta.

Tabla 5: Especificación del CU Importar calles.

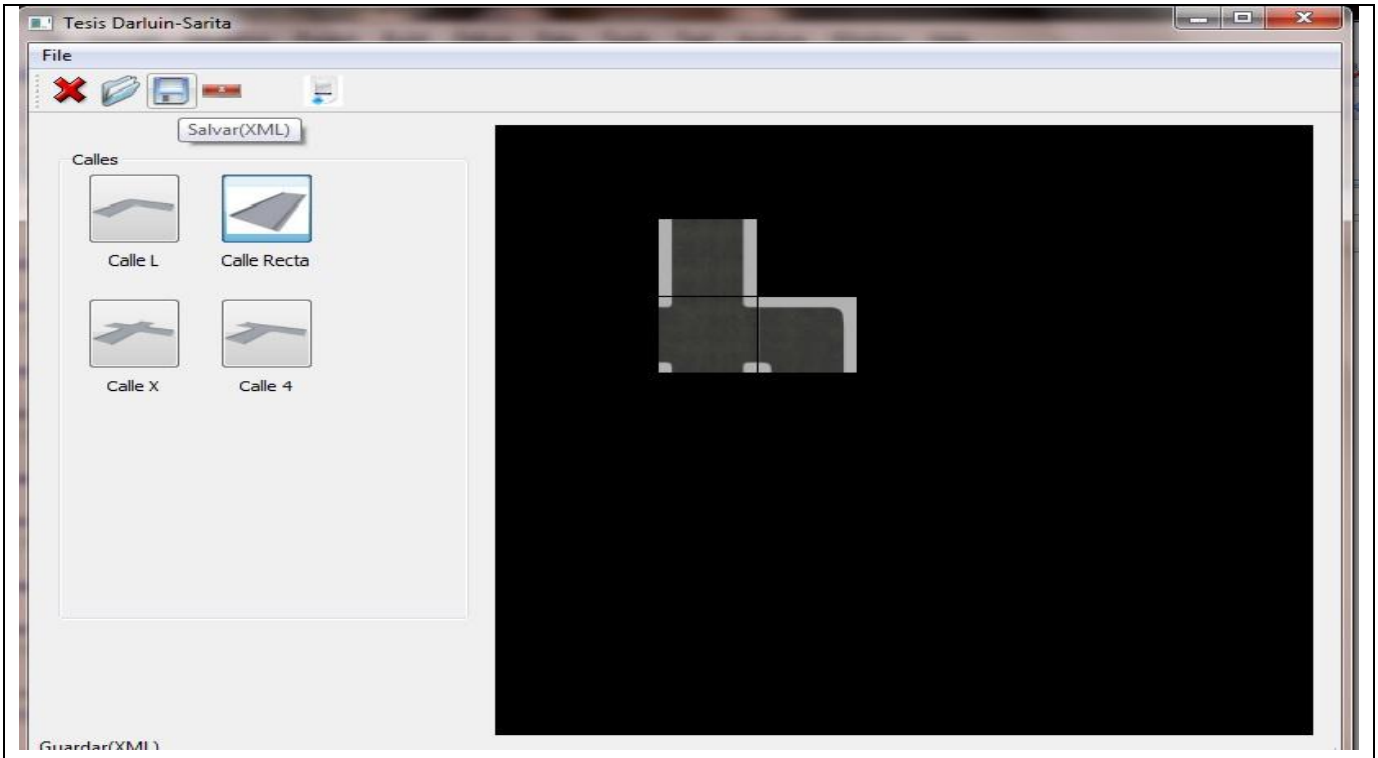
Caso de uso	Importar calles
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el diseñador selecciona la opción “Open”, selecciona en la ventana de búsqueda un modelo de calles, carga el fichero .mesh y finaliza el caso de uso cuando lo adiciona en la biblioteca de calles.
Precondiciones:	El sistema debe haber habilitado las funcionalidades con las que va a trabajar el diseñador.

Propósito:	Cargar una calle y adicionarla en la biblioteca de calles.
Referencias:	RF2
Prioridad:	Crítico
Flujo normal de eventos	
Sección "Insertar calle en la biblioteca".	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción "Open".	1.1 Muestra una ventana para la búsqueda.
2. Selecciona el modelo de la calle.	2.1 Carga el .mesh.
	2.2 Inserta el .mesh en la biblioteca de calles.
	2.3 Visualiza el modelo.
Prototipo de interfaz	
	
Pos-condiciones:	Se cargó una calle, y se adicionó a la biblioteca de

	calles.

Tabla 6: Especificación del CU Exportar entorno.

Caso de uso	**Exportar entorno**
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el diseñador selecciona la opción "Save", selecciona un directorio para salvar el entorno y lo exporta. El caso de uso termina cuando el sistema envía un mensaje.
Precondiciones:	El sistema debe haber habilitado todas las funcionalidades con las que va a trabajar el diseñador.
Propósito:	Salvar el entorno de viales.
Referencias:	RF3
Prioridad:	Crítico
Flujo normal de eventos	
Sección "Exportar entorno"	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción "Save".	1.1 Le muestra los directorios para salvar el entorno.
2. Selecciona un directorio y realiza la exportación.	2.1 Salva el entorno.
	2.2 El sistema envía un mensaje: "El entorno se exportó correctamente".
Prototipo de interfaz	



Pos-condiciones	Se exportó el entorno.

2.7 Diagramas de clases del análisis.

CU Gestionar calles.

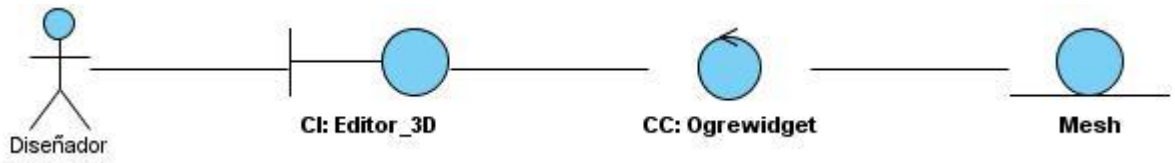


Figura 17: CU Gestionar calles.

CU Importar calles.

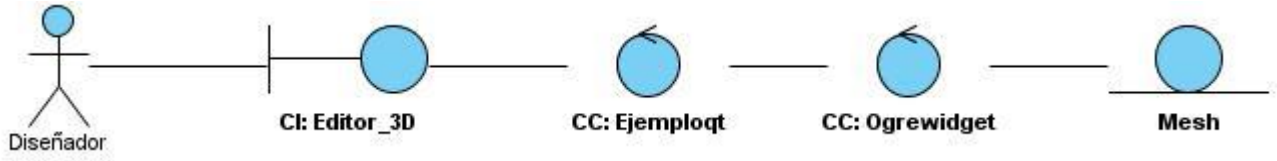


Figura 18: CU Importar calles.

CU Exportar entorno.

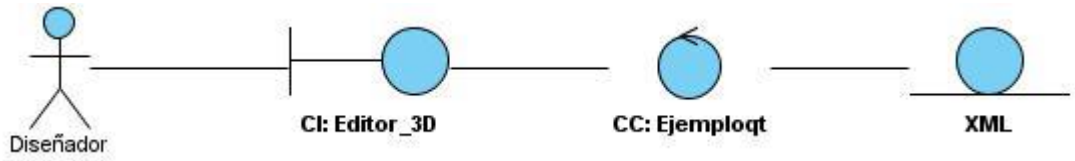


Figura 19: CU Exportar entorno.

2.8 Diagramas de colaboración.

Los diagramas de interacción describen el modo en el que cada operación, detectada en los diagramas de secuencia, lleva a cabo sus responsabilidades y modifica el estado del sistema. En UML los diagramas de interacción pueden representarse mediante los diagramas de colaboración y/o diagramas de secuencia, ambos son representaciones alternas de interacciones. El tipo de diagrama representado en el análisis es el diagrama de colaboración, que muestran cómo los objetos se asocian unos con otros.

2.8.1 CU Importar calles.

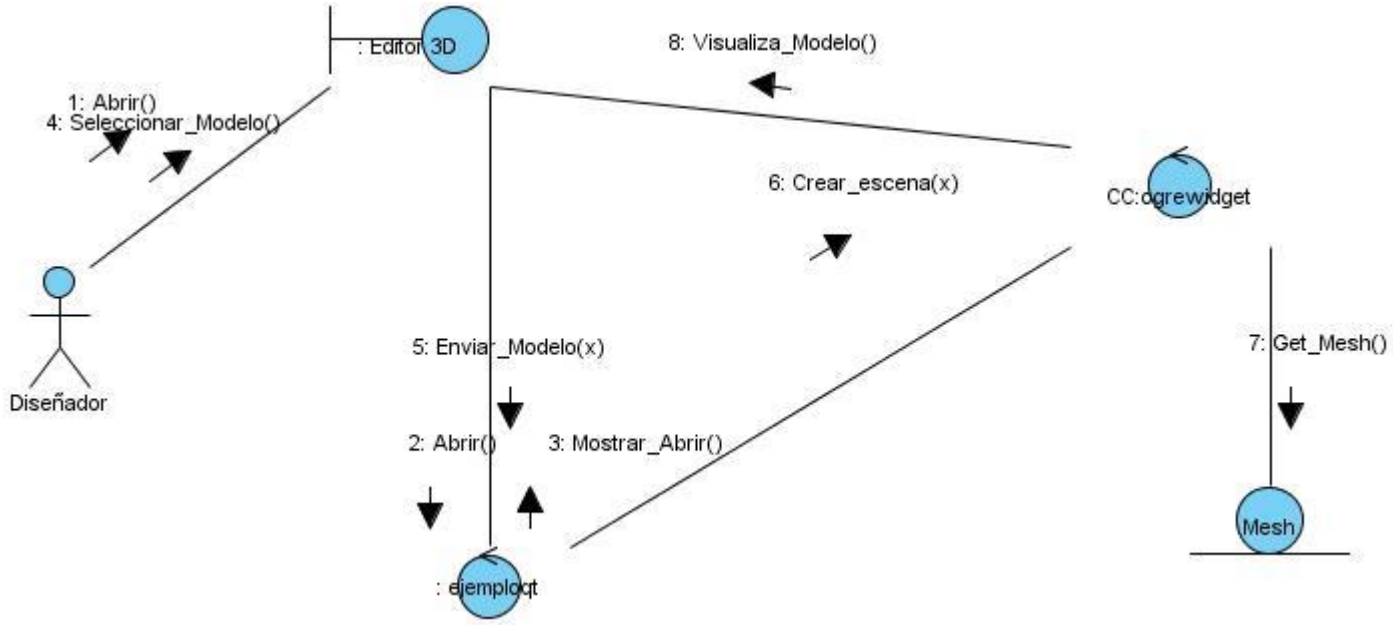


Figura 20: Diagrama de colaboración "Importar calles".

El diseñador selecciona la opción Abrir, y “CC: ejemploqt” le muestra una ventana a este, donde selecciona el modelo, “CI: Editor 3D” le envía el modelo escogido por el diseñador a “CC: ogrewidget” para que se encargue de buscar los datos que necesita del fichero para la visualización del modelo.

2.8.2 CU Exportar entorno.

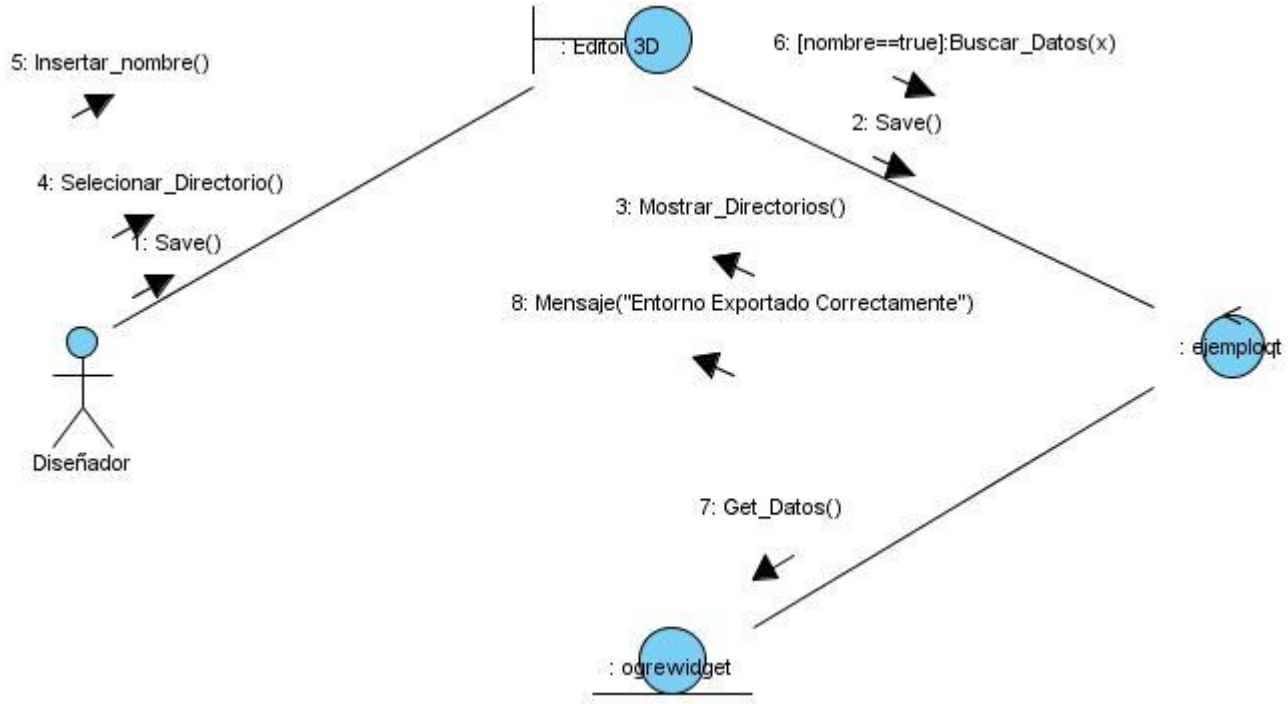


Figura 21: Diagrama de colaboración "Exportar entorno".

Una vez que el diseñador selecciona la opción Save, “CC: ejemploqt” le muestra los directorios para que escoja uno de ellos, después de seleccionado el directorio, acepta para salvar el entorno y “CC: ejemploqt” le envía un mensaje para comunicarle que se exportó correctamente el entorno.

2.8.3 CU Gestionar calles.

2.8.3.1 Sesión “Insertar calle en la escena”.

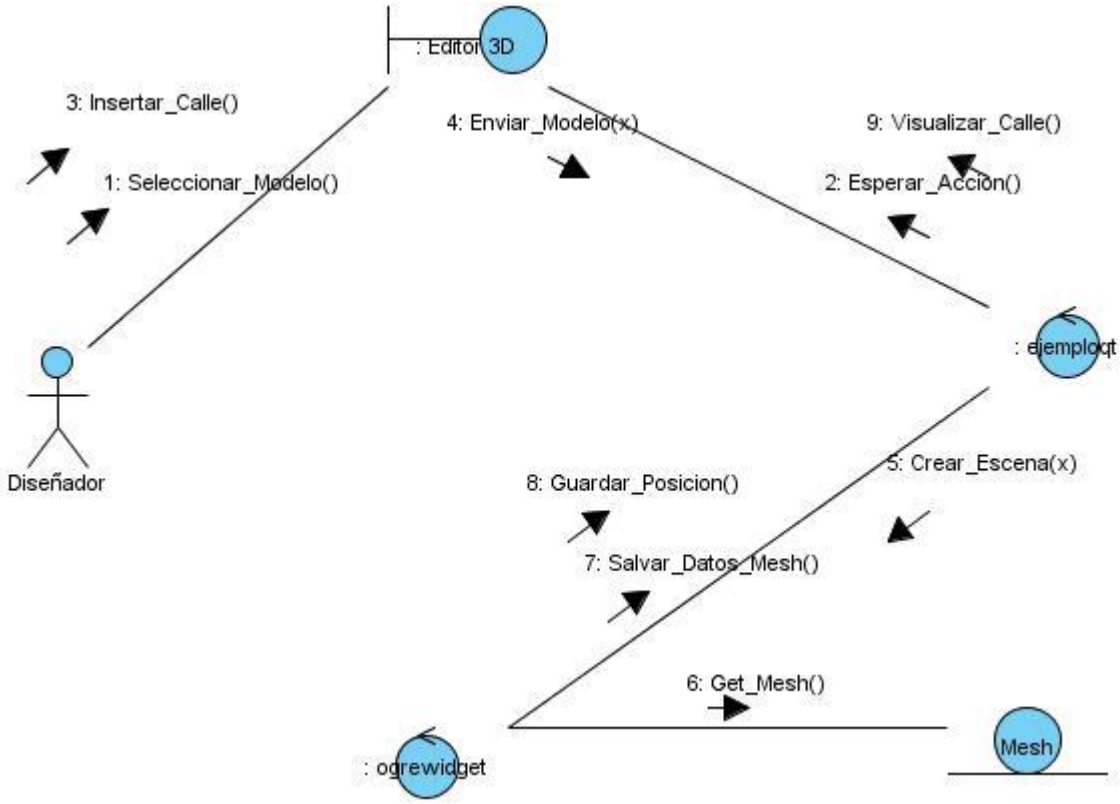


Figura 22: Diagrama de colaboración "Insertar calle en la escena".

Después que el diseñador selecciona la calle de la biblioteca y la inserta en la escena, “CI: Editor 3D” le envía el modelo a “CC: ogrewidget” para que se encargue de buscar información necesaria del .mesh y visualizarlo en la escena.

2.8.3.2 Sesión “Eliminar calle de la escena”.

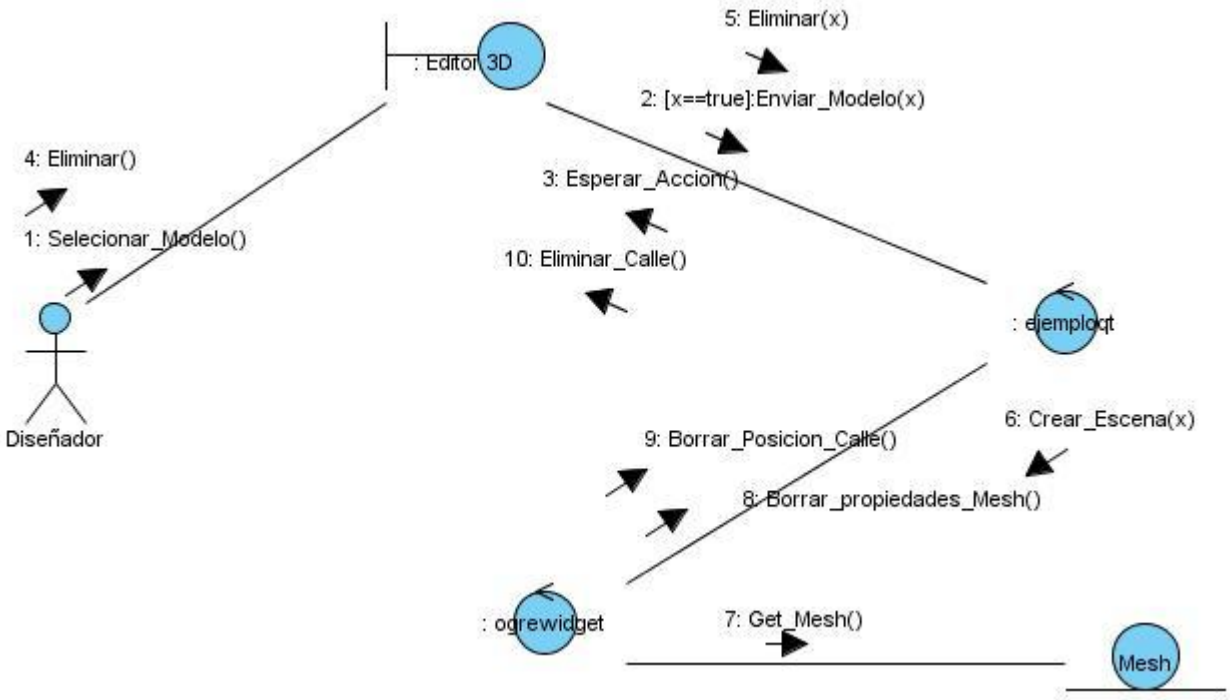


Figura 23: Diagrama de colaboración "Eliminar calle de la escena".

Una vez que el diseñador selecciona una calle de la escena, “CI: Editor 3D” envía el modelo seleccionado a “CC: ejemploqt”, este espera que presione alguna tecla para saber qué acción debe ejecutar, el diseñador selecciona la opción “Eliminar” del menú, “CI: Editor 3D” le envía a “CC: ejemploqt” que acción se va a realizar y finalmente “CC: ogrewidget” toma los datos del .mesh para eliminarlos de la biblioteca y borrar la posición que tiene en la escena.

2.8.3.3 Sesión “Modificar rotación de una calle”.

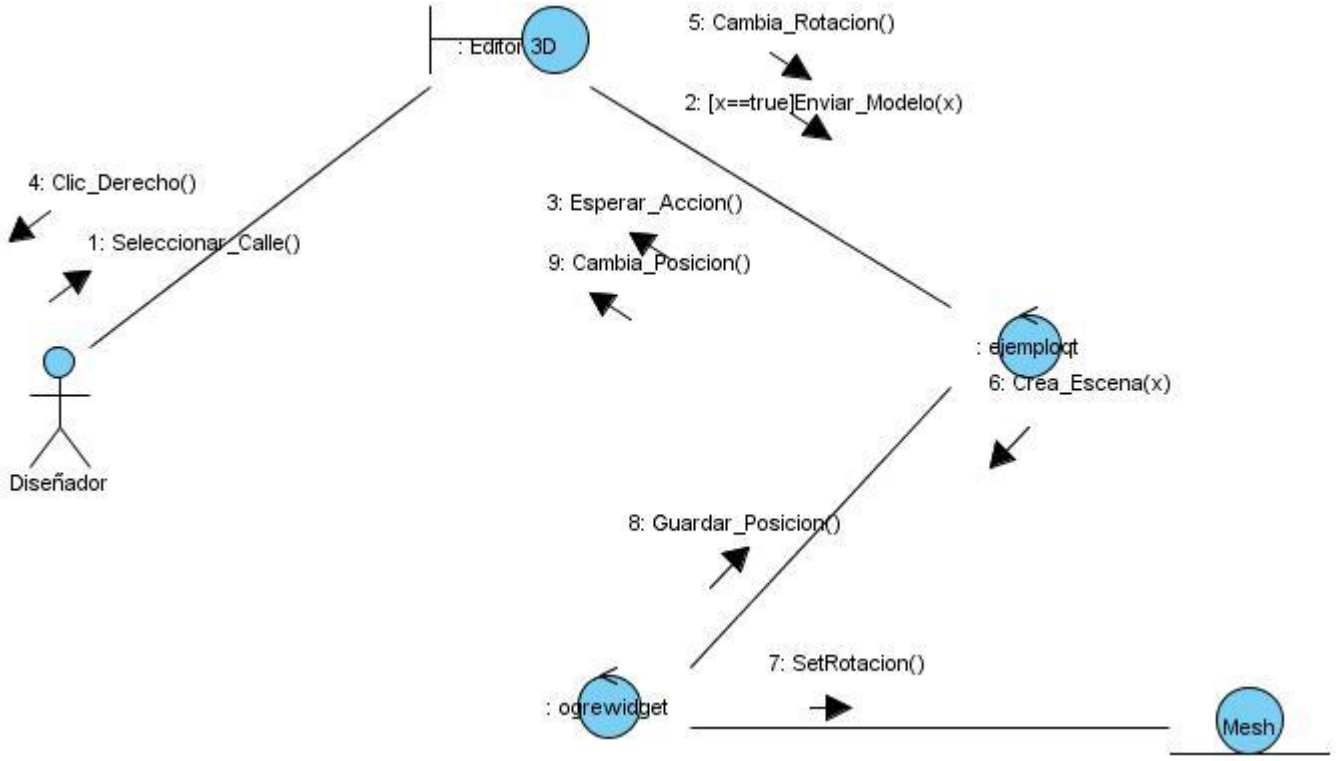


Figura 24: Diagrama de colaboración "Modificar rotación de una calle".

El diseñador selecciona una calle de la escena, “CI: Editor 3D” le envía el modelo a “CC: ejemploqt” y este espera que el diseñador presione una tecla para saber qué acción ejecutar, una vez que el diseñador selecciona la opción, “CI: Editor 3D”, le envía a “CC: ejemploqt” que va a cambiar la rotación de la calle seleccionada y “CC: ogrewidget” se encarga de guardar la posición de la rotación modificada y de cambiarla.

Consideraciones parciales.

En este capítulo quedaron definidas todas las necesidades que deben satisfacerse con el nuevo sistema. Se definieron las herramientas más adecuadas para utilizar en el desarrollo de la aplicación, así como los lenguajes de modelado y programación. Se determinaron los requisitos funcionales, así como las características que debe tener el producto, y se identificaron los casos de uso con sus respectivas especificaciones para permitirle al diseñador obtener los resultados que espera el cliente.

Capítulo 3: Diseño e implementación del sistema.

Introducción.

En el presente capítulo se definen los diagramas de clases y diagramas de secuencias de cada descripción de caso de uso, que describen los aspectos que están familiarizados con el desarrollo de la aplicación.

Teniendo como base los requisitos funcionales y no funcionales que se declararon anteriormente se define la estructura del sistema y además se definen los diagramas de despliegue y componentes.

3.1 Patrones de diseño.

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio (10). Los patrones de diseño describen un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software y entregan una buena solución ya probada. Esto permite diseñar correctamente y en menos tiempo, a construir problemas reutilizables y extensibles y además facilita la documentación, en el presente trabajo se estudiaron diferentes patrones de diseño y se seleccionó solamente uno de ellos para realizar el trabajo.

3.1.1 Singleton.

El patrón Singleton es uno de los patrones más sencillos, tiene la intención de garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella, por lo que si más de un objeto necesita utilizar una instancia de la clase Singleton, esos objetos van a compartir la misma instancia de la clase Singleton. CE_Ogrewidget es la clase que contiene una instancia, creándose una comunicación global hacia esta, por lo las demás clases que existen utilizan la misma instancia de CE_Ogrewidget.

```
    CurrentObject = ogreSceneManager->getRootSceneNode()->createChildSceneNode(Ogre::String(nameCalle)
    CurrentObject->attachObject(ent);

    CurrentObject->setScale(0.1f, 0.1f, 1.0f);
    update();

}
// Permite crear una sola instancia de una clase y con ello un pto de acceso global a ella.
OgreWidget * OgreWidget::Instance()
{
    static OgreWidget ptInstance;
    return &ptInstance;
}
```

Figura 25: Ejemplo Singleton.

3.2 Patrones generales de asignación de responsabilidades (GRASP).

3.2.1 Creador.

El patrón Creador describe que la instancia de un objeto tiene que ser creada por el objeto que tiene la información para ello. La solución de este patrón es asignarle a una clase A la responsabilidad de crear una instancia de la clase B, si: A agrega los objetos B, A contiene los objetos B, A registra las instancias de los objetos B, A utiliza específicamente los objetos B y A contiene los datos de inicialización que serán transmitidos a B. Basándose en la aplicación la clase a la que se le asigna la responsabilidad de crear instancias de un objeto es la clase CE_Ogrewidget, ésta utiliza y tiene conocimiento de métodos, objetos de las clases CE_Interpreter y CE_Ejemploqt.

3.2.2 Experto.

Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad, es decir se le asigna a la clase CE_OgreWidget la responsabilidad de realizar el renderizado de las calles, debido a que contiene las funciones necesarias para el trabajo con el gráfico.

3.3 Diagrama de clases del diseño.

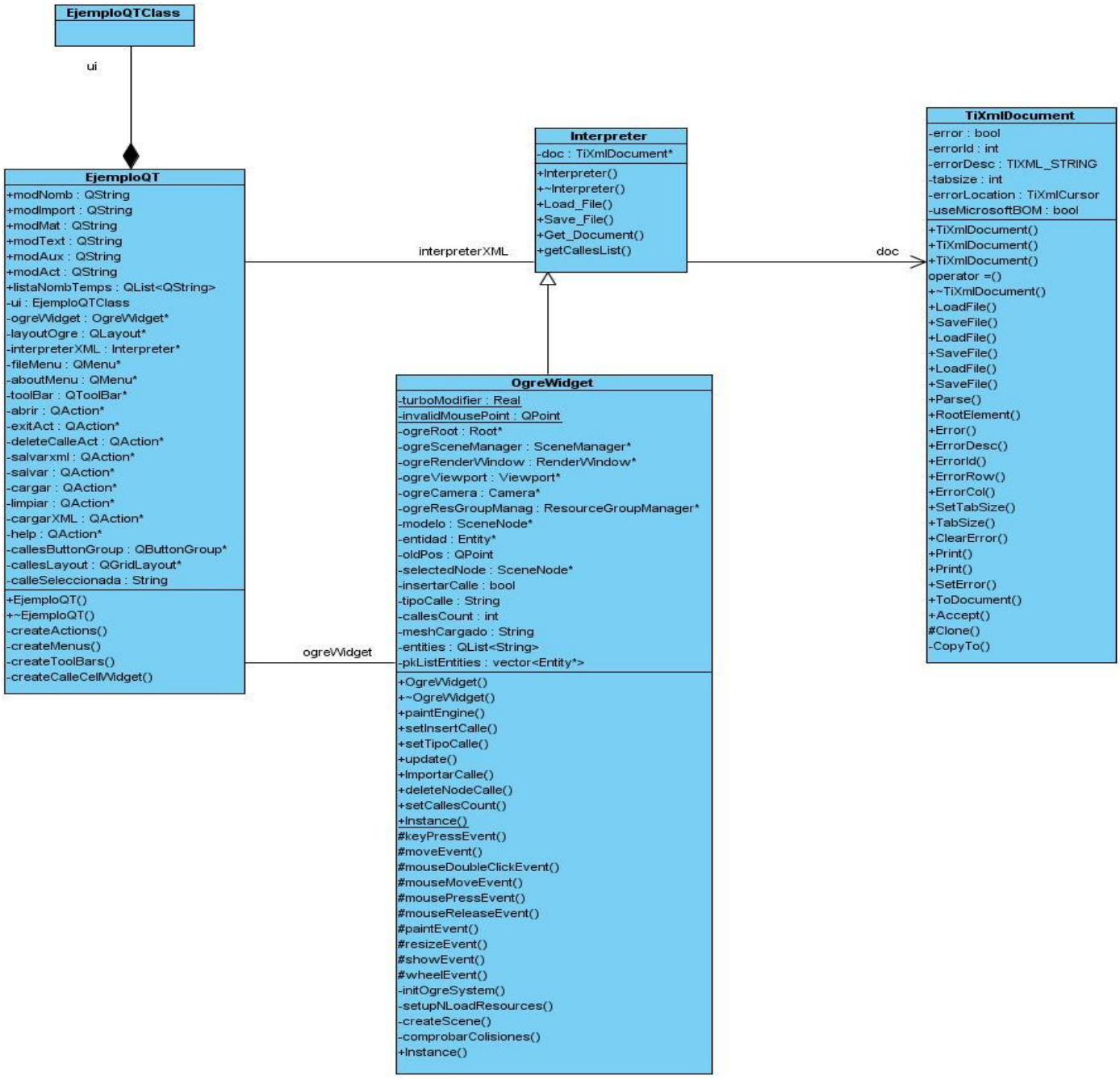


Figura 26: Diagrama de clases de diseño.

3.3.1 Descripción de las clases de diseño.

Tabla 7: Descripciòn de la clase CE_OgreWidget.

Nombre: CE_OgreWidget	
Tipo de clase	Interfaz
Atributo	**Tipo**
ogreRoot	Root *
ogreSceneManager	SceneManager*
ogreRenderWindow	renderWindow*
ogreViewport	Viewport*
ogreCamera	Camera*
ogreResGroupManag	ResourceGroupManager*
modelo	sceneNode*
Entidad	Entity*
oldPos	Qpoint
selectedNode	SceneNode*
insertarCalle	Bool
tipoCalle	String
callesCount	Int
Entities	QList<String>
pkListEntities	Vector<Entity*>
Para cada responsabilidad	
initOgreSystem()	Inicializa los objetos de manipulación de Ogre.
setupNLoadResources()	Inicializa los recursos de Ogre.
createScene()	Crea escenario de trabajo.

update()	Actualiza el renderizdo en el viewport.
keyPressEvent()	Captura el accionamiento de una tecla.
moveEvent()	Proporciona el movimiento del mouse.
mouseDoubleClickEvent()	Captura el accionamiento del doble click izquierdo.
mouseMoveEvent()	Captura el movimiento del mouse.
MousePressEvent()	Captura el accionamiento de cualquier tecla del mouse.
mouseReleaseEvent()	Captura la liberación de cualquier tecla del mouse.
paintEvent()	Actualiza el renderizado en el viewport.
resizeEvent()	Establece un campo de acción del mouse, dentro de un área determinada.
showEvent()	Activa los eventos de Ogre y Qt inicializados anteriormente.
wheelEvent()	Modifica el control de la cámara.
setInsertCalle()	Permite modificar el estado de la calle.
setTipoCalle()	Permite modificar el tipo de calle.
ImportarCalle()	Permite la importación de un objeto.
deleteNodeCalle()	Permite eliminar un nodo de la escena.
setCallesCount()	Permite contar las calles que se insertan.
comprobarColisiones()	Permite las colisiones entre dos objetos.
Instance()	Permite crear una sola instancia de una clase y con esta un punto de acceso global a ella.

Tabla 8: Descripción de la clase CE_EjemploQT.

CE_EjemploQT	
Tipo de clase	Controladora
Atributo	**Tipo**
modNomb	QString
modImport	QString

modMat	QString
modText	QString
modAux	QString
modAct	QString
Iu	EjemploQTClass
ogreWidget	OgreWidget*
layoutOgre	QLayout*
Interpreter XML	Interpreter*
fileMenu	QMenu*
aboutMenu	QMenu*
toolBar	QToolBar*
Abrir	QAction*
exitAct	QAction*
deleteCalleAct	QAction*
Salvarxml	QAction*
Salvar	QAction*
Cargar	QAction*
Limpiar	QAction*
cargarXML	QAction*
callesButtonGroup	QButtonGroup*
calleSeleccionada	String
Para cada responsabilidad	
createActions()	Permite crear botones en la interfaz.
createMenus()	Permite crear un menú en la interfaz.
createToolBars()	Crea la barra de herramientas.

Tabla 9: Descripción de la clase CE_Interpreter.

CE_Interpreter	
Tipo de clase	Controladora
Atributo	**Tipo**
Doc	TiXmlDocument*
Para cada responsabilidad	
LoadFile()	Función para cargar el fichero XML.
SaveFile()	Función para generar el fichero XML.
Get_Document()	Retorna el documento generado.
GetCallesList()	Retorna la lista de calles.

3.4 Diagramas de secuencia.

Los diagramas de secuencia muestran interacciones entre objetos basadas en el tiempo, así como lo que sucede en cada momento, conteniendo para ello, los objetos con sus ciclos de vida y los mensajes que se envían entre ellos ordenados secuencialmente. Los diagramas de secuencia son particularmente importantes para los diseñadores, pues clarifican los roles de los objetos en un flujo, facilitando así una entrada básica para determinar las responsabilidades de una clases y las interfaces.

3.4.1 CU Importar Calle.

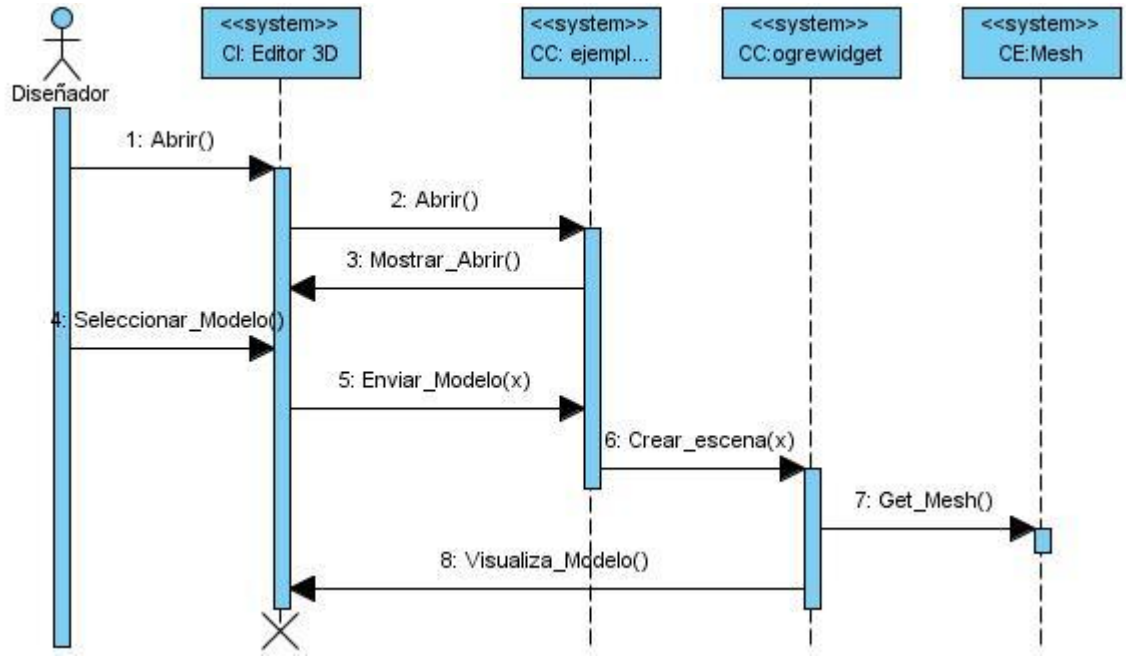


Figura 27: Diagrama de secuencia sesión "Insertar calle en la biblioteca".

Cuando el diseñador selecciona la opción Abrir, “CC: ejemploqt” le muestra una ventana a este, donde selecciona el modelo, “CI: Editor 3D” le envía el modelo escogido a “CC: ogrewidget” para que se encargue de buscar los datos necesarios del fichero para la posterior visualización del modelo.

3.4.2 CU Exportar Entorno.

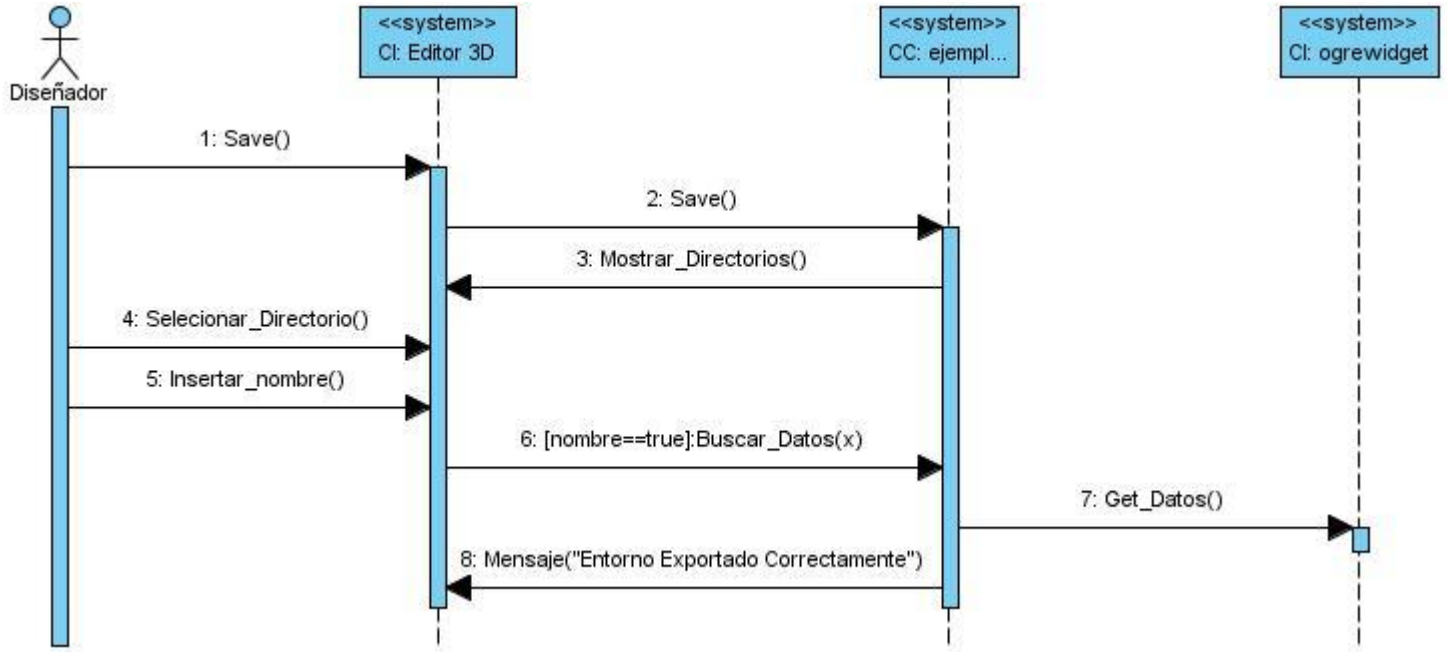


Figura 28: Diagrama de secuencia "Exportar Entorno".

Una vez que el diseñador selecciona la opción Save, “CC: ejemploqt” le muestra los directorios para que escoja uno de ellos, luego de seleccionarlo se salva el entorno y “CC: ejemploqt” le envía un mensaje para comunicarle que se exportó correctamente el entorno.

3.4.3 CU Gestionar calles.

3.4.3.1 Sesión “Insertar calles en la escena”.

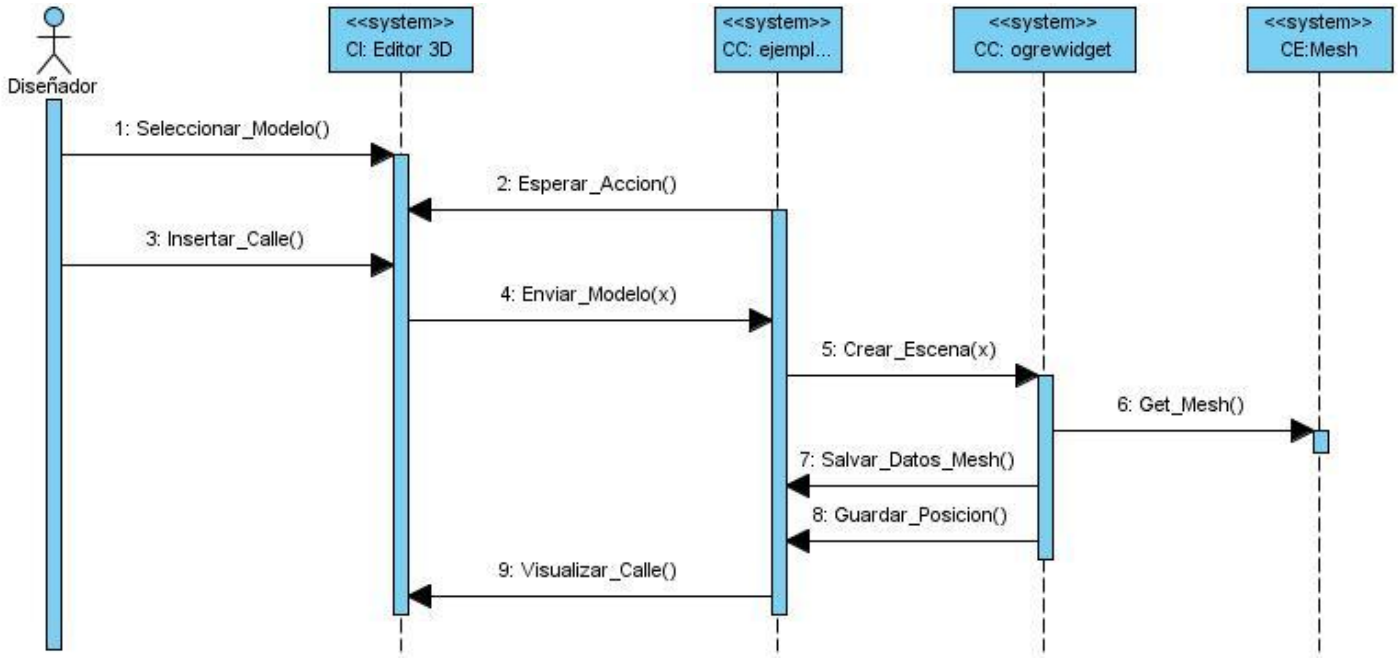


Figura 29: Diagrama de secuencia sesión "Insertar calles en la escena".

Después que el diseñador selecciona la calle de la biblioteca y la inserta en la escena, “CI: Editor 3D” le envía el modelo a “CC: ogrewidget” para que este se encargue de buscar los datos del mesh y poder visualizarlo en la escena.

3.4.3.2 Sesión “Eliminar calle de la escena”.

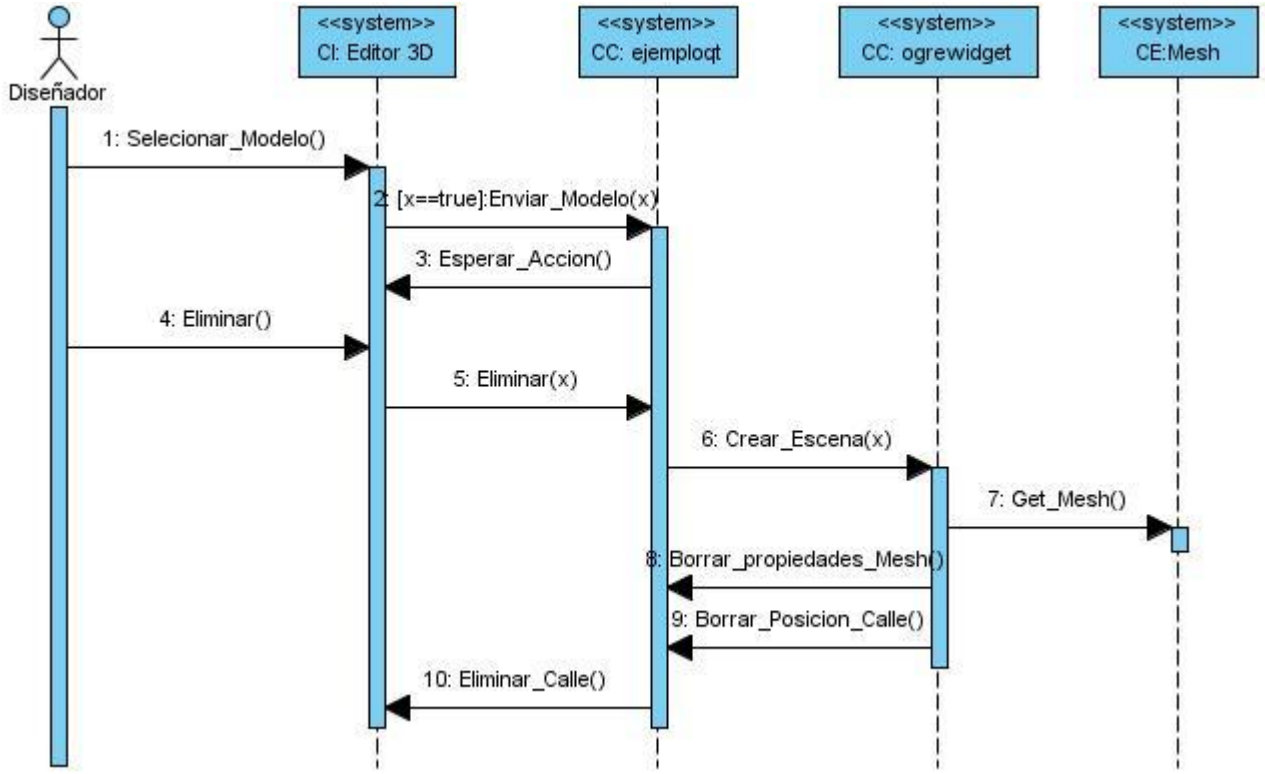


Figura 30: Diagrama de secuencia sesión "Eliminar calle de la escena".

Una vez que el diseñador selecciona una calle de la escena, “CI: Editor 3D” le envía el modelo escogido a “CC: ejemploqt”, este espera que presione alguna tecla para saber qué acción ejecutar, el diseñador presiona el botón “Eliminar” del menú, “CI: Editor 3D” le envía a “CC: ejemploqt” que acción se va a realizar y finalmente “CC: ogrewidget” toma los datos del .mesh para eliminarlos de la biblioteca y borrar la posición que tiene en la escena.

3.4.3.3 Sesión “Modificar rotación de una calle”.

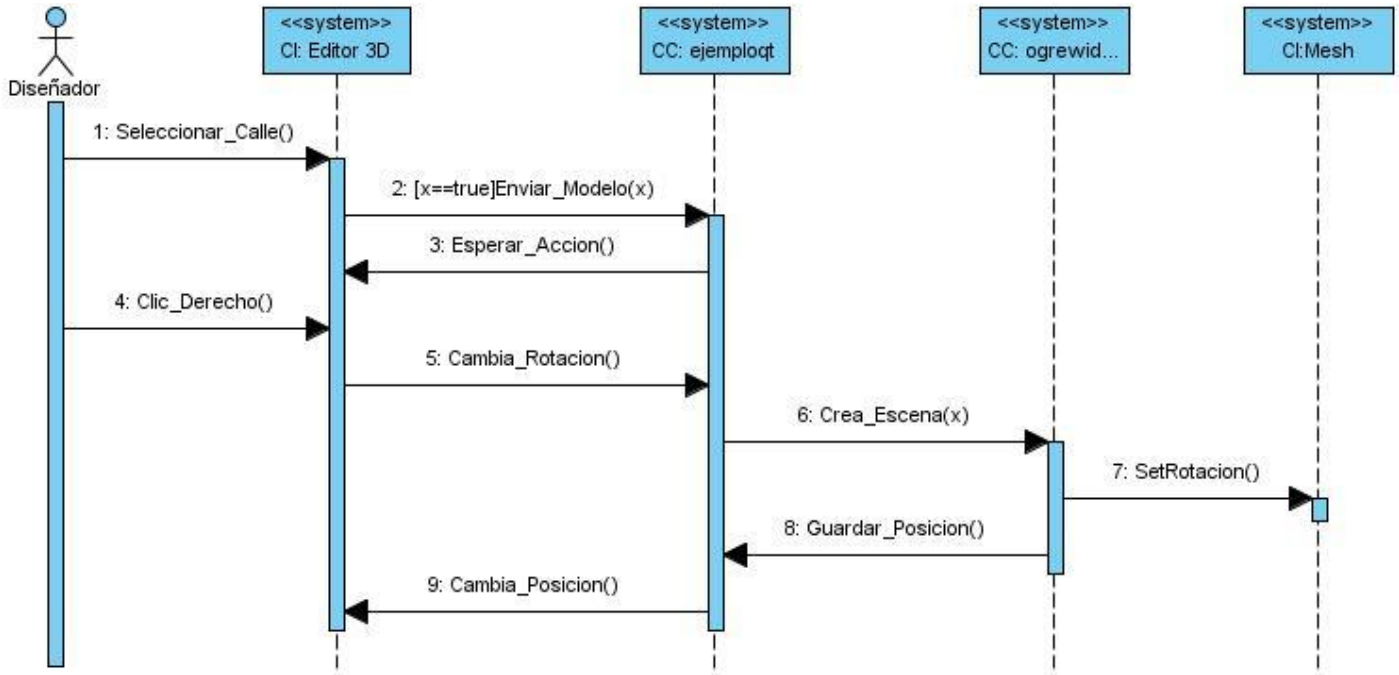


Figura 31: Diagrama de secuencia sesión "Modificar rotación de una calle".

El diseñador selecciona una calle de la escena, “CI: Editor 3D” le envía el modelo a “CC: ejemploqt” y este espera que presione una tecla para saber qué acción ejecutar, una vez que el diseñador presiona la tecla, “CI: Editor 3D”, le envía a “CC: ejemploqt” que va a cambiar la rotación de la calle seleccionada y “CC: ogrewidget” se encarga de guardar la posición de la rotación modificada y de cambiarla.

3.5 Diagrama de despliegue.

El diagrama de despliegue es un grafo de nodos que se relacionan mediante conexiones de comunicación, muestra las relaciones físicas entre los componentes de hardware que forman la topología sobre la que se ejecuta el sistema y la distribución de sus partes. El diagrama de despliegue realizado solamente incluye un nodo físico (PC), el cual no va a tener conexión con ningún otro nodo como serían bases de datos u otra PC puesto que la aplicación no necesita de estos para su correcto funcionamiento, además todas las funcionalidades de la aplicación son realizadas dentro de esta PC.

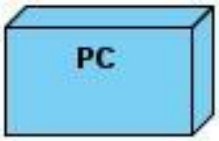


Figura 32: Diagrama de despliegue.

3.6 Diagrama de componentes.

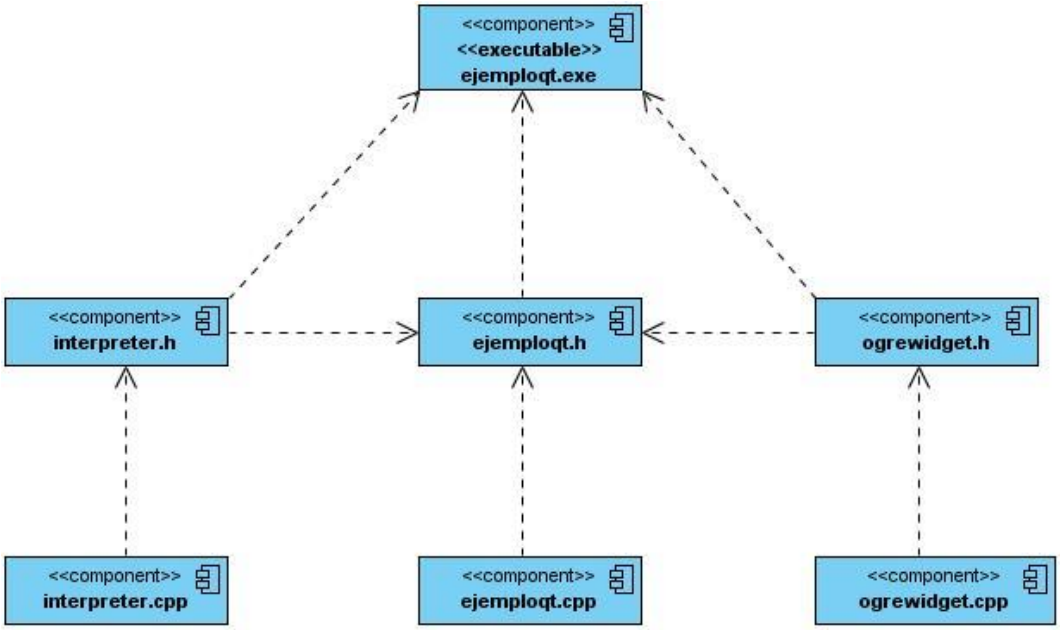


Figura 33: Diagrama de componentes.

Consideraciones parciales.

En el presente capítulo quedaron demostrados los artefactos necesarios para el desarrollo del editor de calles y quedó bien definido el diseño de la aplicación de forma tal que quedó preparado el ambiente para la posterior implementación del sistema.

Conclusiones.

Para resolver el problema de la línea de producción fue necesario darle cumplimiento a las tareas de investigación definidas, es decir realizar un estudio sobre formatos de ficheros 3D y seleccionar uno de ellos para el posterior desarrollo de la aplicación, las ventajas y desventajas que tienen los editores más usados hoy en día, así como de los motores gráficos. Una vez realizado el estudio se pasó a escoger el formato más adecuado al sistema. Después de cumplir con las investigaciones se logró tener una idea de la posible solución al problema. Definiendo las características que tendría la nueva aplicación, sus ventajas, funcionalidades, así como la decisión de las herramientas que se utilizarían en la implementación del sistema.

Posterior a obtener la posible solución se conceptualizó la propuesta llevando a cabo el proceso de la Ingeniería de Software, definiendo algunas restricciones del sistema con el objetivo de definir condiciones para satisfacer con las necesidades, requisitos funcionales, y se identificaron los casos de uso con las descripciones de cada uno, además se diseñaron las clases del sistema y se elaboró el diagrama de componentes que contiene las clases del sistema y el diagrama de despliegue.

Finalmente se cumplieron los objetivos propuestos con el desarrollo de un editor de calles 3D que permite editar un sistema de calles, garantizando con este que el proceso de edición sea más rápido y sencillo, y que demuestra que el método usado por la línea de producción de diseño no es el más adecuado para cubrir las necesidades de dicha línea.

Bibliografía Referenciada.

1. Biblioteca. [Citado el: 2010]"Tesis" http://biblioteca.uci.cu.

2. Arquitectuba.com.ar. *Arquitectuba.com.ar.* [En línea] 2008. http://www.arquitectuba.com.ar/software-gratis/topocal/.

3. Google SetchUp. *Google SetchUp.* [En línea] 2010. [Citado el: 26 de 01 de 2010.] http://sketchup.google.com/intl/es/product/features.html.

4. El rincón del vago. *El rincón del vago.* [Citado el: 25 de 01 de 2010.] http://html.rincondelvago.com/autocad_6.html.

5. Geofumadas2.0. *Geofumadas2.0.* [En línea] 2009. [Citado el: 25 de 01 de 2010.] http://galvarezhn.cartesianos.com/2009/04/25/qcad-alternativa-de-autocad-para-linux-y-mac/.

6. 2. Istram.*Istram.*[En línea]2006. [Citado el: 25 de 01 de 2010.] http://www.istram.net/index.php?opcion=11.

7. Arquitectuba.com.ar. *Arquitectuba.com.ar.* [Citado el: 25 de 01 de 2010] http://www.arquitectuba.com.ar/curso-sketchup-gratis/exportacion-de-modelos-3d-3ds/.

8. *Introducción a Ogre-Por qué OGRE?*

9. Creación de videojuegos. *Creación de videojuegos.*[Citado el: 26 de 01 de 2010.] http://www2.ing.puc.cl/~iic3686/Tutorial_Ogre1/ogre1.htm.

10. *Eva.* [Citado el: 2010] http://eva.uci.cu.

11. *Wikipedia.* [En línea] 2010. [Citado el 2010] http://es.wikipedia.org/wiki/3ds_Max.

12. . Royal Software. *Royal Software.* [En línea] [Citado el 2010] http://royal-oem.com/shop/search/?s=autocad&cpn=www_shirky_com.

13. Royal Software. *Royal Software.* [En línea] [Citado el 2010] http://royal-soft.net/shop/search/?s=3ds%20%25max&cpn=www_nkeconwatch_com_soft_ports.

14. *Bitsbeta.* [En línea] 2010. [Citado el 2010] http://bitsbeta.com/google-sketchup-editor-para-diseno-3d.

15. *Wikipedia.* [En línea] [Citado el 200] http://es.wikipedia.org/wiki/OpenGL

16. Dibujo Técnico. [En línea] http://dibujotecnicolimid.blogspot.com/2008/02/funcion-del-autocad.html.

Bibliografía Consultada

1. Ingeniería de Software 1 . [En línea] http://eva.uci.cu/course/view.php?id=102.

2. Ingeniería de Software 2. [En línea] http://eva.uci.cu/course/view.php?id=259.

3. Visual Paradigm. *Ayuda de Visual Paradigm para UML Enterprise Edition. .*

4. API de OGRE.

5. QT Examples and Demos.

6. **Junker, Gregory.** *Pro OGRE 3D Programming.*

7. **Paradigm, Visual.** *Ayuda de Visual Paradigm para UML Enterprise Edition.*

8. [En línea] http://www.ogre3d.org/tikiwiki/tiki-index.php.

9. Wikipedia. [En línea] http://es.wikipedia.org/wiki/Biblioteca_%28inform%C3%A1tica%29.

Recomendaciones.

- Brindar la posibilidad de insertar en la aplicación otros tipos de objetos prediseñados, sean estos: casas, árboles, entre otros.
- Implementar la funcionalidad de colocar los objetos usando todos los ejes de coordenadas.
- Una ayuda para obtener el entendimiento del funcionamiento de la aplicación.

Glosario de Abreviaturas.

2D: dos dimensiones.

3D: tres dimensiones.

CAD: Diseño Asistido por Ordenador.

BMP: nombre del formato Bit Map.

3DS: formato 3D Studio.

OBJ: formato creado por Alias Wavefront.

DWF: Design Web Format.

STL: Standard Tessellation Language.

ASCII: American Standard Code for Information Interchange.

MTD: Modelo Digital de Terreno.

API: interfaz de programación de aplicaciones.

Glosario de Términos.

3Dmax: software de diseño.

Maya: software de diseño.

Header: cabecera.

Autodesk Inc.: empresa desarrolladora de software.

Mesh: formato 3D.

Polilíneas: secuencia de líneas o segmentos conectados, creados como un objeto único.

Scroll automático: desplazamiento automático.

Buffer: ubicación de la memoria en un ordenador o en un instrumento digital reservada para el almacenamiento temporal de información digital.

Sprites:

Sketchup: software de diseño y modelado.

API: interfaz de programación de aplicaciones.

Engine: motor.

Índice de Figuras.

Índice de Tablas