Universidad de las Ciencias Informáticas

Facultad 5

Título: EDITOR DE CONFIGURACIONES DE ESCENAS PARA LABORATORIOS VIRTUALES

Trabajo de Diploma para optar por el título de:

Ingeniero en Ciencias Informáticas.

Autor: Jandy Miguel Gómez Rodríguez.

Tutor(es): Msc. Orlay García Ducongé

Ing. Yaima Fiallo Valle.

DECLARACIÓN DE AUTORÍA:

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los __ días del mes de __ del año 2012.

Autor:

Nombre del autor: Jandy M Gómez Rodríguez.

Tutores:

Nombre del primer tutor. Orlay García Ducongé.

Nombre del segundo tutor. Yaima Fiallo Valle.

Agradecimientos:

La elaboración de este trabajo ha estado guiada por muchas personas que directa o indirectamente han contribuido a su terminación, a todas esas personas quiero hacerles llegar mis más sinceros agradecimientos. Agradecer a la oportunidad que me brindó la vida de estudiar en esta universidad. De forma especial a mi tutor Msc Orlay García, cuya actitud en todo momento ha sido la de mas que un tutor, un amigo con el que siempre he podido contar en estos 4 años en los que ha sido mi tutor. Al Msc Leoder Alemañy y la Ing Yaima Fiallo dispuestos en todo momento a atender las dudas y propuestas que iban surgiendo. A mis padres, con los cuales he podido contar en todo momento para apoyarme en las buenas y las malas. A mi hermano quien es mi mejor amigo y mi más fuerte aliado en la batalla de la vida. A mi abuela quien siempre me ha brindado su sabiduría, y ha sido fuente de inspiración para lograr los resultados que hoy he obtenido. A mi tía, mi tío y mi primo, quienes son parte muy importantes de mí y siempre me han brindado su apoyo y ayuda incondicional. A mi novia Yisel, por haberme brindado su cariño, comprensión y apoyo en los momentos más difíciles. A todos mis amigos, quienes han estado al tanto de este trabajo en todo momento. A mi familia en general por todo el apoyo y confianza que han depositado en mí.

Dedicatoria:

A mi abuelo Keké y mi abuela Divina
A mis padres. A mi hermano
A toda mi familia

Resumen:

El presente trabajo presenta una herramienta para editar las configuraciones de las escenas creadas por el grupo de diseño encargado de la realización de las escenas y modelos 3D en el proyecto laboratorios virtuales. También pretende el proceso de desarrollo de futuras versiones de laboratorios virtuales. Se propone un sistema que sea capaz de leer los ficheros (XML) de extensión ".scene" que describen las propiedades de los objetos tridimensionales que conforman los entornos de los laboratorios virtuales. Se hace necesario a dichos ficheros poder realizarles modificaciones, así como añadirles nuevos elementos.

Contenido

Introducción:

La realidad virtual puede considerarse como un mundo virtual generado por computadoras o por algún sistema informático. La cual brinda al usuario la sensación de estar en un mundo que no es real, pero que puede interactuar en alguna medida con los objetos que lo rodean en ese mundo, como lo haría en la vida real. La realidad virtual ideal sería aquella que hiciera una reproducción completa de todos los sentidos presentes en la vida real (tacto, gusto, olfato, vista etc.). Pero en la actualidad debido al elevado costo de recrear todos los sentidos, los sistemas se centran en recrear los sentidos auditivos y visuales. Como forma de compensar esta ausencia de sentidos y para aumentar la sensación de realismo, se resalta la interacción profunda con los elementos que componen el mundo virtual.

La realidad virtual, puede ser clasificada como inmersiva o no inmersiva. Dicha clasificación viene dada de acuerdo a la forma en la que se interactúe con el sistema. Los métodos inmersivos requieren dispositivos especiales, como cascos, guantes, sensores de posicionamiento, entre otros. Los cuales hacen que la interacción entre las personas y los mundos virtuales sea realista.

En los métodos no inmersivos, el mundo es representado a través de una ventana de escritorio. Este método no necesita de dispositivos especiales para la interacción con el sistema, permitiendo interactuar en tiempo real con diferentes personas en espacios y ambientes que en la realidad pueden no existir. La interacción se realiza utilizando los medios convencionales como el *mouse* y el teclado.

En nuestra universidad específicamente en el Centro de Informática Industrial (CEDIN), se desarrollan aplicaciones de realidad virtual. Nuestro centro fue contratado para la realización de tres laboratorios virtuales utilizando los métodos no inmersivos.

Durante el desarrollo de los mismos se detectaron un conjunto de problemas. Ocurría que los modelos entregados por el equipo de diseño encargado de la confección de las escenas de los laboratorios virtuales, no tenían la configuración adecuada para ser visualizados en los laboratorios virtuales. Además los modelos carecían de información necesaria para el trabajo dentro de los laboratorios, la cual no era posibles añadir utilizando las herramientas de diseño con las que se trabaja. Estas configuraciones eran modificadas o añadidas de forma manual o a nivel de programación, sobre los ficheros que eran exportados por el grupo de diseño.

Los Laboratorios Virtuales se desarrollaron mediante el empleo del motor gráfico Ogre3D, que cuenta con OgreMax, un *plugin* que permite su acople a las herramientas de diseño para generar ficheros de configuración de extensión ".scene". Las escenas de los laboratorios virtuales se basaron en estos archivos pero fueron incorporadas nuevas características. Este *plugin* incorpora una aplicación que representa visualmente la información contenida en los ficheros de configuración, pero no permite la realización de cambios visualmente o sea las modificaciones se deben realizar a nivel de código. También existe el software Ogitor, que es un editor de escenas, pero utiliza ficheros de extensión ".ogscene" que no es compatible con la arquitectura definida para los laboratorios virtuales.

Por tanto la situación problémica puede resumirse en la necesidad de una herramienta que visualice y manipule los ficheros de configuración .scene, con los que se trabaja en el desarrollo de los laboratorios virtuales.

Teniendo en cuenta lo anterior se formula el siguiente **problema científico:** ¿Cómo editar las configuraciones de las escenas en los laboratorios virtuales a través de su modificación por el equipo de desarrollo de los laboratorios virtuales?

Es objetivo de la investigación desarrollar un sistema que permita a los desarrolladores la edición de configuraciones a las escenas en los laboratorios virtuales.

El cual precisa como objeto de estudio: las características de las escenas en los laboratorios virtuales. Definiendo como campo de acción el proceso de configuración de escenas en los laboratorios virtuales.

Posible resultado: Una herramienta para la configuración de escenas de los laboratorios virtuales.

Para cumplir el objetivo se proponen las siguientes **tareas investigativas**:

1. Elaboración del marco teórico a partir de la determinación de las particularidades de los laboratorios virtuales y sus características.
2. Análisis de la arquitectura de los laboratorios virtuales para poder integrar la herramienta resultante con los mismos.
3. Determinación y selección de herramientas y lenguajes para la implementación del sistema.
4. Diseño del sistema para tener una guía durante el proceso de desarrollo.
5. Implementación del sistema a partir del diseño realizado.
6. Validación del sistema elaborado para saber si cumple con las necesidades del proyecto laboratorios virtuales.

Entre los métodos científicos utilizados se destacan:

Métodos Teóricos:

Analítico-Sintético: Con el uso de este método de investigación se analizarán las informaciones obtenidas, mediante su desglose, para después realizar una síntesis de las mismas y arribar a las principales ideas.

Histórico-Lógico: Mediante este método se analizará la trayectoria y la evolución de las diferentes formas de manipular los objetos en los laboratorios virtuales para seleccionar la más adecuada.

Modelación: Este método se utilizará para crear un prototipo funcional del sistema de interacción natural.

Métodos Empíricos:

Análisis documental: Para seleccionar la información necesaria para la construcción del marco teórico.

Consulta de especialistas: Para recibir los criterios de validación sobre la aplicabilidad y utilidad de lo realizado.

Observación: Para reunir información visual sobre lo que el objeto de estudio hace y cómo se comporta.

Pruebas: Para valorar el desempeño del sistema que se va a elaborar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Las interfaces gráficas (1)

En los tiempos que se viven, resulta imposible imaginar nuestra vida diaria sin la presencia de las computadoras. Esto viene dado a que se han introducido en nuestra vida como unas herramientas potentes de las cuales resulta muy difícil prescindir.

Paralelamente a este desarrollo de las ciencias de la computación, se ha venido dando la tendencia de hacer más fácil la interacción hombre-PC, para lograr que la utilización de los sistemas no sea solo para usuarios avanzados sino para todo tipo de usuarios. Por tanto han ido desapareciendo paulatinamente las consolas con gran cantidad de comandos siendo sustituidas por botones, clics, menús y otros elementos que corresponden con las interfaces gráficas, apoyándose en las ventajas que brindan los periféricos como: *mouse*, teclado, etc.

Las Interfaces Gráficas de Usuario en inglés *Graphics User Interface* (GUI) son las técnicas que más se utilizan hoy en día a la hora de programar, puesto que logran un nivel de aceptación elevado entre los usuarios finales de los productos y dan la posibilidad de hacer más agradable las aplicaciones a la vista de los usuarios, con la posibilidad de hacer más amigable el software.

Los inicios de las interfaces gráficas se remontan a principios de los años 80 del pasado siglo Cuando la empresa Xerox Corporation desarrolló el Xerox PARC y más tarde el Xerox Star 8010, los que fueron los primeros ordenadores personales, que contaban con un rudimentario sistema operativo que utilizaba una interfaz de usuario como la conocemos hoy. También estos ordenadores utilizaban el *mouse* como periférico de entrada. El sistema operativo de este ordenador denominado Alto, nunca se comercializó, sin embargo Xerox, donó parte de sus investigaciones al resto de sus fabricantes, siendo la Apple Lisa la primera computadora con interfaz gráfica en ponerse a la venta en el mercado. Y posteriormente fue generalizado su uso a las demás empresas como Microsoft.

Hasta lo que conocemos hoy en día, donde casi todos los sistemas cuentan con una interfaz de usuario amigable, aunque sin deshacerse de las consolas.

EL gráfico 3D (2)

Paralelamente al desarrollo de las GUI, también han venido ganando en calidad y realismo las aplicaciones visuales. Dando lugar a la aparición de los gráficos en tercera dimensión (3D) generados por computadoras y más tarde a una rama de las ciencias de la computación conocida como Realidad Virtual (RV). Los gráficos en 3D pueden definirse como la representación de los datos geométricos que se almacenan en la computadora, con el propósito de realizar cálculos para crear imágenes. Estas imágenes generadas utilizando técnicas 3D, permiten trabajar con elementos que tributan grandemente a la calidad de la imagen final. Por ejemplo, el tratamiento de la perspectiva, que brinda una verdadera sensación de profundidad a un ambiente de 2D como es una imagen plana. También está el trabajo de la iluminación y su simulación, al proyectarse sobre superficies que generen sombra, así como mejorar otros como la proporción, el color en las imágenes, con el objetivo de mejorar la calidad visual de la imagen final.

Realidad Virtual (3)

Todo este desarrollo en el grafico 3D, ha traído consigo que se comiencen a desarrollar aplicaciones, cuyo objetivo sea el de simular, en un entorno generado por computadoras, un mundo virtual con cierto grado de realismo. Estas aplicaciones se denominan aplicaciones de Realidad Virtual. A pesar de que en la literatura se pueden encontrar muchas definiciones de realidad virtual (en inglés, *virtual reality* VR), seguramente una de las más completas es la que propuso A. Rowell: (1)

“La Realidad Virtual es una simulación interactiva por computador desde el punto de vista del participante, en la cual se sustituye o se aumenta la información sensorial que recibe”.

Uno de los elementos básicos que caracterizan a un sistema de RV, es la simulación interactiva. La misma consiste en una simulación los sentidos, los cuales se recrean en un mundo virtual, que solo existe en la memoria del ordenador. Y que sea interactiva, es lo que la distingue de una animación, principalmente porque en una animación como por ejemplo una película, los espectadores son entes pasivos, porque no pueden alterar el contenido de las imágenes que se proyectan en la pantalla. Esto no se puede hacer, porque las mismas han sido grabadas desde una posición, siguiendo una trayectoria que será inalterable. No sucede así en un sistema de RV, el cual permite al usuario además de interactuar con el medio y sus objetos, moverse por el mundo de manera dinámica, sin movimientos predefinidos.

Dado que el movimiento de un usuario en la escena es imprevisible y los puntos de vista son infinitos. Es necesario tener una representación geométrica del espacio en 3D. De manera tal que la imagen pueda ser calculada atendiendo a la posición que adquiera el observador en la escena. Y además de este modelo geométrico, también deben contar con una visualización realista, apoyada en algoritmos de sincronización de imágenes, que permitan mostrar los elementos a partir de las representaciones de las escenas.

La realidad virtual utiliza la interacción implícita en contraposición a la interacción clásica. Esta última es cuando un usuario quiere realizar una determinada operación como por ejemplo, moverse de una habitación a otra, es necesario que se comunique de forma explícita con el sistema. Para la realización del procedimiento de interacción, el usuario utiliza un esquema de comunicación determinado por la interfaz de la aplicación. Ya sea una basada en comandos o una basada en GUI. En cualquiera de los dos casos, el usuario debe conocer o recordar cuál sería el botón o comando asociado a esa determinada acción que quiere realizar. Y esta interacción se envía por los dispositivos de interacción clásicos como *mouse* y teclados.

A pesar de que las interfaces gráficas en los últimos años han mejorado. No dejan de suponer cierto esfuerzo por parte del usuario, que requiere de cierto tiempo de entrenamiento y familiarización con el sistema. En cambio, la realidad virtual con interacción a través de métodos implícitos, lo que busca es que la comunicación entre el usuario y el sistema, se lleve a cabo por métodos que impliquen la utilización de movimientos naturales. Un ejemplo es el control de la cámara virtual. En un sistema de realidad virtual, con este tipo de interacción, el sistema actualizaría la posición de la misma, en función de los movimientos de la cabeza del usuario. Si el usuario quiere ver la parte del mundo virtual que tiene detrás, no tendría que utilizar ningún comando ni pulsar una tecla solamente tendría que realizar el mismo gesto que haría en el mundo real.

Independientemente del tipo de interacción que se utilice, ya sea implícita o explícita, la presencia de dispositivos periféricos para la entrada de datos en cualquier sistema de RV siempre va a existir. Pero la diferencia fundamental es la percepción que tiene el usuario de estos dispositivos.

Escena (4)

Las aplicaciones de RV, están compuestas principalmente por un grupo de entidades geométricas que en su conjunto se combinan para dar lugar a formas verdaderamente complejas, y al mismo tiempo estas entidades cuentan con propiedades que las definen dentro del espacio tridimensional. La combinación de estos elementos en un punto espacial es lo que en el gráfico 3D se puede definir como escena 3D. Para la obtención de dicha escena, el ordenador debe realizar los cálculos de la generación de imagen (*render*), generación de sonido espacial, calculo de colisiones, etc. Los elementos observados en una escena, no son más que la modelación en tres dimensiones de cada uno de los elementos que la componen, estos elementos son llamados modelos 3D.

El proceso de creación de los modelos tridimensionales o modelado como también se le conoce, parte fundamentalmente de estructuras simples que componen un elemento de mayor complejidad. Por ejemplo, un balón de futbol es una esfera, una lata es un cilindro y un dado es un cubo. Estos son objetos simples basados en formas básicas. Paralelamente estas estructuras básicas están compuestas por elementos todavía más simples que componen cualquier forma en un entorno tridimensional. Estas estructuras llamadas polígonos pueden observarse por ejemplo en un cubo que a su vez está compuesto por 6 caras, de las cuales cada una es un polígono. Pero incluso una forma redondeada está compuesta por polígonos, un ejemplo en la vida real se puede apreciar en un balón de futbol que está compuesto por 12 pentágonos y 20 hexágonos.

Además de los modelos 3D, una escena puede contener otros elementos. Por ejemplo, comportamiento físico dentro de ella, la iluminación, sonido ambiente, texturas aplicadas en modelos, etc. Todos estos datos mencionados constituyen informaciones que se encuentran almacenados en ficheros que guardan las escenas. Dichos ficheros son leídos e interpretados y luego el sistema construye y visualiza la escena.

La información no se encuentra almacenada dentro del fichero de manera arbitraria. Sino que utiliza una estructura de manera tal que exista una jerarquía en la escena. Por ejemplo, un auto en el cual la carrocería constituye un modelo y los neumáticos son otros modelos. Resultaría muy complejo aplicar cualquier transformación como, movimiento, a cada uno de los modelos que componen el auto, sin embargo, si se coloca los neumáticos en un nivel más bajo que la carrocería dentro de la distribución de la escena, cualquier transformación que se le aplique a la carrocería, se trasmitiría a los neumáticos. La estructura de dato que más ayudaría para realizar esta operación serían los árboles. Por tanto en el ejemplo anterior quedaría en un nivel superior una estructura llamada carrocería, de las cuales descenderían hacia un nivel inmediato inferior varias estructuras llamadas neumáticos.

Bibliotecas de desarrollo de sistemas de realidad virtual (1)

Desde el punto de vista de la Informática (computación), una biblioteca es un conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de que puedan ser aprovechadas por otros programas. Existen dos tipos de bibliotecas, las bibliotecas estáticas o de enlace y las bibliotecas compartidas o de enlace dinámico. De estas, las primeras hacen enlaces entre ellas, o sea, arreglan las referencias a rutinas en el programa para que apunten a su localización en la biblioteca al momento de compilación, mientras que el segundo tipo de bibliotecas se enlazan en tiempo de ejecución.

La denominación de biblioteca compartida hace énfasis en que, comúnmente, los procesos que la enlazan comparten una única parte de la memoria donde se encuentran las instrucciones de los subprogramas.

Algunos ejemplos de bibliotecas son:

• En Windows las DLL (Dynamic Link Library), las cuales entran dentro de la clasificación de las bibliotecas dinámicas, mientras que las estáticas son las lib.

• En los sistemas Unix y Linux, las bibliotecas dinámicas tienen extensión .so, mientras que las estáticas toman extensión .a.

En el mundo de la realidad virtual podemos encontrar muchas de estas bibliotecas vinculadas a bibliotecas de desarrollo de sistemas de realidad virtual, algunas más avanzadas que otras, pero existen las que son más fáciles de usar. Dentro de estas bibliotecas vinculadas a la realidad virtual, se puede distinguir que la mayoría brindan al usuario ficheros tanto de enlace dinámico, como de enlace estático, ambas con el objetivo de hacer más eficiente el proceso de compilación o ejecución según sea el caso.

Algunas de estas bibliotecas de desarrollo de sistemas de realidad virtual son:

• DirectX.

• Ogre.

• G3D.

• A7

Editores de Escenas (5)

Cuando se observa un fichero de escena, solamente se puede interpretar a simple vista, un grupo de números y letras, que definirían sus características como: que objetos o modelos hay en la escena, las texturas que hay aplicadas, la posición de los objetos en la escena etc. Pero no se cuenta con una idea exacta de los cambios que se realizan. Para resolver esta problemática, se crean los editores de escena. Que son herramientas que al mismo tiempo que interpretan el código de un fichero de escena lo visualizan. También permite modificar las características de las escenas. Todo esto sin perder la organización de la información dentro del fichero. De manera que una vez que se modifica la información, la misma pueda ser guardada ya sea en un nuevo archivo o la modificación del ya existente.

Es necesario destacar que aunque los editores de escenas son capaces de modificar la información de un archivo de escena. Solo pudiera modificar la información de los modelos ya existentes en la escena. Esto resulta importante destacar, porque en el mundo existen softwares muy potentes como: 3D Max, Blender, Maya etc. Que pudieran desde cierto punto de vista, considerarse como editores de escena, ya que son capaces de modificar los elementos de una escena y exportar luego un archivo con las características modificadas. Pero su naturaleza no es la de modificar solamente, porque al mismo tiempo son capaces de crear los propios modelos desde cero. Una vez dicho esto se puede definir que un editor de escena es:

Un software que permite la modificación de las características de los elementos de una escena, sin que brinde la posibilidad de la modificación de las mallas de dichos elementos y la creación de modelos desde cero.

Editores de Escena basados en Ogre (6)

Ogitor SceneBuilder

Este software es un editor de escena, que utiliza el motor de render OGRE para la visualización. Permite la fácil creación de escenas que posteriormente podrán ser cargados con una aplicación de OGRE. La ventaja que brinda esta herramienta es que las escenas, pueden ser utilizadas tanto para aplicaciones de creación rápida como para una aplicación final. El nombre Ogitor según sus creadores viene de las palabras Editor-Ogro. Una de las características novedosas que incorpora Ogitor es su concepto de interfaz de plugins. La cual permite la creación de módulos personalizados y que pueden ser integrados de manera sencilla por parte de los usuarios.

Esta propiedad del Ogitor permite la adición de nuevas características por parte de los usuarios. Que da la posibilidad de seguir enriqueciéndolo en cuanto a funcionalidades sin que pierda la esencia. Algunas de estas características adicionales son por ejemplo bibliotecas físicas, de sonido etc. La personalización de SceneManager de OGRE permite, dependiendo de la aplicación, el control total de los elementos en Ogitor por medio de secuencias de comandos y propiedades personalizadas. La escena cuenta internamente con una estructura de ordenamiento basada en XML y el formato de salida de sus ficheros es .ogescene.

Características.

- Arquitectura basada en plugins
    - Plugins compatibles
        - Hydrax - Over-/underwater simulación océano.
        - SkyX - simulación del cielo, con ciclos de día y noche.
- Multiplataforma
  se ejecuta en Windows, Linux y Mac OSX.

- Gestión de proyectos basados en la escena.
- Interfaz de scripting extensible través de Angelscript.
- Interfaz personalizable
    - Los paneles se pueden organizar y definir su tamaño a gusto del usuario.
    - Guardar y cargar diseños de ventana.
    - Traducciones que existe en muchos idiomas.

OgreMax Exportador Escena

Software desarrollado por Derek Nedelman que se acopla a manera de *plugin* con otros softwares de diseño en 3D, como el Blender. El exportador de escena OgreMax genera un fichero de formato .scene con las mismas características que la escena que se observa en una herramienta de diseño, en cuanto al ordenamiento de las estructuras XML. Además del *plugin*, el software cuenta con dos aplicaciones la información de la escena. Y un visor que permite al usuario un conjunto de funciones de manera tal que pueda explorar la escena. Pero los cambios solo se pueden hacer directamente en el código de la escena o a través del software de edición como 3D Max o Blender.

Características del exportador.

- Elementos de la escena.
    - Nodos
    - Cámaras
    - Luces
    - Entidades
    - Mallas
    - Los datos de usuario personalizadas
    - Pista / buscar objetivos

    - Medio ambiente (caja de cielo, Sky Dome, el plano del cielo, el rango medio ambiente, la niebla, el color del ambiente, la configuración de la sombra)
    - Terreno (con una escena suplente)
    - Sistemas de partículas (con una escena suplente)
    - Modelos (describe una colección de objetos)
- Exportación de malla
    - Compatible con el modificador Morpher
    - Exportación en forma de triángulos, puntos o líneas (aristas visibles)
    - Colores de los vértices
    - Normales
    - Coordenadas de textura
    - Submallas (para múltiples submateriales)
    - XML y formatos binarios
- Exportación de mapa de bits
    - Copiar / convertir mapas de bits en el directorio de salida de escena
- Ventanas personalizadas Escena
    - Objetos de previsualización de la escena y sus animaciones en tiempo real
    - OgreMax cambio material aparece instantáneamente en ventanas
    - Crear como ventanas que quieras, ya sea incrustadas en las ventanas estándar o de libre flotación-por encima de la interfaz de usuario principal
- Otras características
    - La altura del terreno y la generación de mapa de color
    - Cubo de la generación de mapas
    - De archivos de vídeo a la imagen de los archivos de conversión
    - Tipo de ayuda personalizada

Funciones del Visor

- Cargas. Escena y archivos. Malla
- Permite al usuario navegar a través de la escena con cualquier cámara
- Opción de bloquear la cámara de visión a una cámara de escena
- Escenas de muestra

OFusion

Este software, también es un plugin desarrollado por Tecnologías oFusion que se acopla con otros software de diseño 3D. Representa la siguiente generación de herramientas WYSIWG para el desarrollo de contenidos de juegos. WYSIWG es el acrónimo de ***W****hat* ***Y****ou* ***S****ee* ***I****s* ***W****hat* ***Y****ou* ***G****et*, (lo que ves es lo que obtienes). Las principales perspectivas que tiene son en el desarrollo de la próxima generación de OGRE *Powered Games*. Creado principalmente para satisfacer las crecientes demandas de los artistas, con una herramienta más eficiente. Y de esta forma reducir el tiempo de producción. Incluye un conjunto de herramientas para crear y ver el contenido de alta calidad y ser utilizado por el motor de render OGRE.

Otros editores no basados en Ogre

OSGEdit

El desarrollo de OSGEdit comenzó en 2002, apoyado por el *hosting de Source Forge* y el respaldo de la comunidad *OpenSceneGraph* (OSG).

De entre sus características debía permitir montar escenas, hacer pequeños ajustes y ejecutar procesos típicos de un grafo de escena.

Aunque la idea evolucionó en cierto modo, el desarrollo continuó enfocándose en hacer un compositor de escenas, no una herramienta completa de creación de contenidos con modelado, animación etc. Su diseño también ha evolucionado desde la primera versión donde la interfaz era rígida hasta las últimas versiones donde la interfaz se genera dinámicamente. Las contribuciones a esta herramienta vienen dadas gracias a la gran cantidad de personas que tributan a este proyecto.

La generación del GUI está automatizada, ya que esta es la única manera de soportar cualquier característica de OSG en un tiempo razonable. No obstante, esta característica de la interfaz se ve penalizada con un incremento de la inestabilidad del programa

La generación automática de la GUI se realiza utilizando una descripción XML proporcionando meta-datos sobre las clases OSG. Los meta-datos separan los detalles OSG de la interfaz de usuario proporcionando un mecanismo genérico para conocer los atributos de cada clase y sus características la interfaz de usuario se genera a partir de estos meta-datos así como las acciones que actualizan el grafo escena relacionadas con la opción undo/redo del programa.

Características de OSGEdit:

- El grafo escena se representa mediante un árbol donde se pueden seleccionar los nodos para cualquier operación. Este árbol es realmente un grafo, un mismo nodo puede colgar distintas ramas, aunque realmente represente un nodo compartido no una copia.
- Cada tipo de nodo tiene un diálogo de propiedades integrado en la ventana (la mayoría soportadas).
- Se pueden manipular las transformaciones visualmente, hay herramientas para mover, escalar y rotar nodos de transformación visualmente con el ratón.
- Todas las operaciones excepto *new*/*open*/*save* y *zoom* se pueden deshacer y rehacer sin límite.
- Se pueden abrir todos los formatos que soporta OSG así como guardar en todos los formatos que soporta OSG.
- Se pueden unir ficheros añadiéndolos como nodos hijos.
- Se pueden copiar, pegar y cortar nodos.

Valve Hammer Editor

El Valve Hammer Editor era antes conocido con el nombre de *Worldcraft.* Valve Hammer es el editor oficial de la empresa Valve para el juego *Half-Life*. Es un editor gratuito, se puede descargar de cualquier página oficial. Con él se pueden crear mapas para *Half-Life*, *Counter-Strike*, etc. Su funcionamiento es sencillo, tiene una interfaz intuitiva, y su puesta en funcionamiento es relativamente fácil. La versión oficial. Está en inglés, aunque hay una versión no-oficial en español.

Actualmente la versión de este software, de la desarrolladora Valve, es la 4.0. Esta versión si es de "pago", se adquiere cuando se compra una copia original de los siguientes videojuegos: *Half-Life 2*, *Counter-Strike*, *Day of Defeat Source*. Cuando adquirimos uno o varios de estos productos, estamos obligados a registrarlos en la plataforma *Online Steam* de la propia compañía desarrolladora del *Half-Life*, Valve. Una vez registrada nuestra copia de software en dicha plataforma, se nos dará la posibilidad para acceder al *Kit* de desarrollo para el *Source Engine*. La herramienta se llama *Source SDK* (*Software Development Kit*) donde dispondremos de las siguientes herramientas:

- Valve Hammer Editor 4.0
- HL2 Model Viewer
- HL2 Face Poser

Estándar XML (1)

XML *o Extensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y una adaptación del SGML (*Standard Generalized Markup Language*) y permite definir la gramática de lenguajes específicos (de esta misma manera HTML es a su vez un lenguaje definido por SGML). XML no fue concebido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es una tecnología sencilla que posee a su alrededor

otras que hacen su complemento y a su vez la hacen mucho más grande. En la actualidad tiene gran importancia, ya que permite la compatibilidad entre sistemas para compartir información de una manera segura, fiable y fácil.

Algunas de las ventajas que XML reporta son:

• Es extensible, lo cual significa que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
• El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de la aplicación.
• Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

Actualmente existe un grupo de tecnologías que hacen uso de este tipo de formato para el almacenamiento de información, específicamente en el ámbito de la Realidad Virtual. Actualmente podemos encontrar diferentes tipos de ficheros 3D, que hacen uso del formato XML para almacenar información del estado de la geometría, textura, etc. Esto a su vez ha permitido grandes avances en la tecnología web, ya que actualmente podemos encontrar disponibilidades de navegación por entornos en tres dimensiones. Es decir que actualmente el formato XML ha posibilitado la extensión de la realidad virtual hasta esta esfera.

Bibliotecas para el trabajo con XML

Stream XML

Esta biblioteca permite la lectura de datos de un documento XML como una secuencia de símbolos. Esto difiere de otras bibliotecas que proporcionan

controladores para recibir eventos del analizador XML mientras que el *Stream* XML controla el ciclo, sacando símbolos del lector cuando se necesitan. Este enfoque permite construir analizadores recursivos-descendentes, permitiendo que el código de análisis XML se divida en diferentes métodos o clases.

SAX XML

Procesa el documento o información en XML por eventos. De esta forma la información en XML se analiza conforme esta sea presentada (*evento por evento*), efectivamente manipulando cada elemento a un determinado tiempo, sin incurrir en uso excesivo de memoria. Si bien este es un enfoque rápido y sencillo para leer documentos XML, la manipulación es relativamente difícil, porque los datos se almacenan y se desechan de forma serial.

DOM XML

DOM genera un árbol jerárquico en memoria. La información en cada elemento, es considerado un nodo dentro del árbol. Este árbol jerárquico de información, puede ser modificado

Este árbol jerárquico **de información en memoria** permite que a través de algún "*parser"* sea manipulada la información, las ventajas serian las siguientes:

- Puede ser agregado un nodo (Información) en cualquier punto del árbol.
- Puede ser eliminada información de un *nodo* en cualquier punto del árbol.

Lo anterior se ejecuta sin incurrir en las penalidades o limitaciones de manipular un archivo de alguna otra manera.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

Descripción de la solución propuesta.

Como propuesta de solución técnica de este trabajo se propone: el desarrollo de una aplicación con interfaz gráfica de usuario para la configuración de escenas. Estas serán utilizadas en el desarrollo de laboratorios virtuales con base en el motor de render OGRE.

El desarrollo de esta aplicación enmarca un conjunto de requisitos que facilitan en gran medida la configuración de las escenas que serán utilizadas. Es por ello que se proponen los siguientes:

- Brindar la posibilidad de visualizar y seleccionar objetos dentro de escenas 3D.
- Visualizar la estructura jerárquica de la escena.
- Permitir la inserción de nuevas propiedades a partir de plantillas definidas previamente.
- Permitir modificar las características de los objetos dentro de la escena.
- Posibilitar el ocultar los elementos o mostrar los elementos que se encuentren ocultos así como eliminarlos de la escena.
- Buscar dentro de la escena un nodo específico.

XML como formato para el almacenamiento de datos

El almacenamiento de la información de la escena se guarda en un fichero de extensión *.scene* que corresponde al tipo de escena con el que trabajan los Laboratorios Virtuales. Como formato para el almacenamiento de dicha información se escogió el estándar XML. Actualmente este formato es uno de los

más utilizados en el mundo dada las grandes ventajas que presenta. Además es un lenguaje extensible y de fácil entendimiento por parte de terceros.

En el caso de esta aplicación, se necesita que la información esté estructurada por categorías. XML por su naturaleza permite organizar la información en contenedores o categorías, lo que posibilita una amplia compatibilidad entre la manera de organizar la información de esta aplicación, con la forma de guardar la información de XML.

Para exportar el formato, se hará uso de las facilidades que brinda la biblioteca de desarrollo de interfaces visuales Qt, ya que cuenta con un conjunto de clases para el manejo del formato de almacenamiento mencionado.

Formas de salvar y leer

El fichero será exportado con extensión .scene e internamente contendrá los datos almacenados con formato XML, para ello se hará uso de las clases para la manipulación de XML que brinda QT específicamente la *Stream XML* que por su forma de manipular los ficheros, se acopla perfectamente con las tareas que se quieren lograr. De igual forma sucede para leer las escenas.

Método de selección de objetos en escenas 3d

El algoritmo de selección de los objetos en la escena 3D está compuesto por dos fases, en el mismo se utiliza la técnica de *raycast*. La cual lanza un rayo desde un origen, hasta un destino previamente determinado y se van obteniendo todos los objetos con los que va colisionando. Existen dos formas de obtener los objetos, ya sea a nivel de objeto o a nivel de la geometría de los objetos. Para la selección de los objetos, se utilizará la técnica de geometría, que pese a ser más costosa, cumple con las condiciones que se quieren obtener de un nivel de exactitud elevado durante el proceso de selección en la escena.

La segunda fase del algoritmo, utiliza la lista de objetos generada por el *raycast*. Que tiene en la primera posición, el primer elemento con el que colisionó y luego

el segundo, de manera consecutiva hasta el último objeto obtenido. Con dicha lista podemos obtener el elemento y modificar sus características.

Herramientas de desarrollo y lenguaje utilizado

El uso de las herramientas que a continuación se mencionan estará presente durante todas las fases de desarrollo de esta aplicación.

QT Creator.

Qt Creator es un IDE multiplataforma creado por Trolltech para desarrollar aplicaciones en C++. Está basado en la biblioteca Qt y cuenta con las siguientes características principales:

- Editor avanzado para C++.
- Diseñador de formularios (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual.

Actualmente Qt constituye una amplia plataforma de desarrollo que incluye un amplio conjunto de clases, bibliotecas y herramientas para la producción de interfaces gráficas, lo que se combina perfectamente con la herramienta que se pretende desarrollar, cuyo contenido principal es la manipulación de una escena *3D*. QT hace uso del lenguaje C++ como lenguaje de programación y a su vez es capaz de operar en varias plataformas por lo que resulta ideal al cumplir con las políticas de desarrollo de software libre de nuestra universidad.

Esta biblioteca está provista de una amplia gama de herramientas que facilitan la creación de formas, botones y ventanas de diálogo con el uso del mouse, lo que permite la creación de aplicaciones muy elegantes, lo que hace más amigable la

interacción de usuarios, con las aplicaciones desarrolladas. Está provista de tres grandes ventajas ante otras bibliotecas similares y que la rivalizan a su vez:

1. Qt es completamente gratuito para Aplicaciones de Código Abierto.

2. Es una biblioteca que está disponible para casi todas las plataformas, ya sea Sistemas Unix, Linux, MacOS, Solaris, así como para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código, siendo así, que esta se visualizará y actuará mejor que una aplicación nativa.

3. Qt hace uso de una extensa biblioteca de clases y herramientas para la creación de aplicaciones gráficas, estas bibliotecas de clases están bien documentadas, así como de fácil uso y tienen una gran herencia de programación orientada a objetos.

Visual Paradigm

Visual Paradigm es una herramienta UML de tipo profesional, la cual soporta el ciclo completo de desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Este software de modelado UML constituye una gran ayuda en la rápida construcción de aplicaciones de calidad a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generador de código desde diagramas y generar documentación. Esta herramienta CASE (*Computer Aided Software Engineering*) como comúnmente se conoce esta provista de un gran número de tutoriales de modelado UML, demostraciones interactivas de UML, así como proyectos de este tipo. Algunas de sus principales características son:

1. Diagrama de Procesos de Negocio – Proceso, Decisión, Actor de Negocio, Documento Modelado colaborativo con CVS y Subversión (nueva característica).
2. Ingeniería de Ida y Vuelta.
3. Ingeniería Inversa, Código a modelo, código a diagrama;

4. Generación de Código, Modelo a Código, Diagrama a Código.

5. Diagrama de flujo de datos.

7. Generador de informes para elaboración de documentación.

8. Distribución automática de diagramas, Reorganización de las figuras y conectores de los diagramas UML.

Reglas del Negocio.

- El procedimiento de lectura de los ficheros de escenas se hará solamente a archivos con extensión .scene.
- La ruta de los recursos de las escenas (modelos, texturas etc.) deben ser definidas, por medio de la herramienta.

Modelo de dominio.

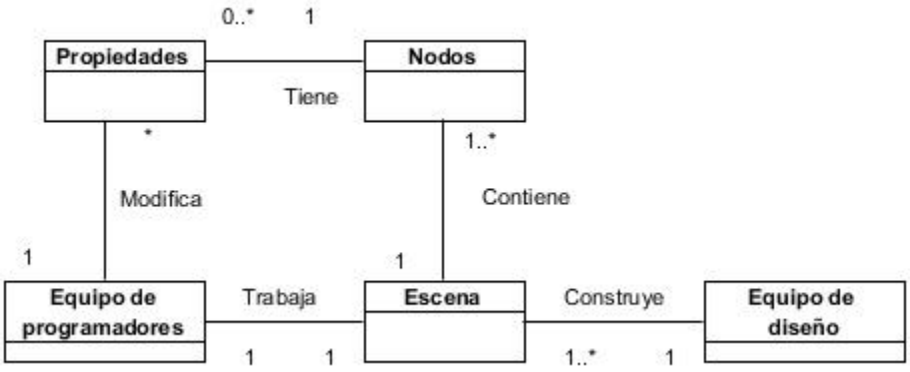


Figura 1: Modelo de dominio.

El modelo de dominio trabaja con los conceptos relacionados a la relación entre los dos equipos de trabajo durante el desarrollo de los laboratorios virtuales. En él se observa la necesidad con la que cuenta el Equipo de desarrollo de modificar las propiedades de las escenas entregadas por el equipo de diseño. Esto no es posible realizar, debido a la inexperiencia de los desarrolladores en herramientas de diseño.

La aplicación final utilizará la misma arquitectura basada en componentes con la que trabajan los laboratorios virtuales. La característica de esta arquitectura es que utiliza componentes ya desarrollados previamente que tributan a la solución final. La ventaja principal es que permiten la reutilización de código para desarrollar otras herramientas de la misma familia.

Captura de requisitos.

Requisitos funcionales:

1. Cargar ficheros *.scene.*
2. Seleccionar los objetos en la escena 3D.
3. Mover la cámara.
4. Rotar la cámara.
5. Insertar elementos a la escena.
6. Editar configuraciones de los nodos.
   6.1. Modificar la posición.
   6.2. Modificar orientación.
   6.3. Escalar los objetos.
   6.4. Ocultar objetos.
   6.5. Mostrar los objetos que estén ocultos.
   6.6. Eliminar objetos.
7. Buscar un objeto específico en la escena.
8. Seleccionar la lista de recursos para los objetos que se van a cargar en la escena.
9. Guardar la escena en un fichero *.scene*.

Requisitos no funcionales:

1. **Requerimientos de Software:**
   1.1. El sistema debe permitir su ejecución en Sistema el operativo Windows
2. **Requerimientos de Hardware:**

2.1. Es necesaria una como mínimo una PC con CPU Intel Pentium IV, 520 Mb de RAM. En cualquier PC con prestaciones superiores.

3. Restricciones en el Diseño y la Implementación:

3.1. Estándar de codificación de los Laboratorios Virtuales.

3.2. Lenguaje de programación C++.

3.3. Biblioteca de clases: Motor de *Rende*r OGRE.

3.4. Arquitectura de los Laboratorios Virtuales.

4. Requerimientos de Usabilidad:

4.1. Permitir una navegación sencilla tanto para usuarios con conocimientos avanzado de informática como para los más inexpertos. Esto se logrará a partir de una estructura de la información correcta, el usuario en todo momento tendrá conocimientos del lugar donde se encuentra dentro del sistema.

5. Interfaz

5.1. **Interfaces de usuario**

5.1.1. Presentar una interfaz gráfica uniforme, incluyendo pantallas, menús y opciones. Los aspectos de: tamaño de las pantallas, color, tipo de letra.

Diagrama de casos de uso

Dada la extensión del diagrama de casos de uso, el mismo no se muestra a continuación, para su análisis dirigirse a la sección anexos. Ver ANEXO1

Descripción de los casos de uso.

Para lograr un mayor entendimiento de las funcionalidades asociadas a cada caso de uso, no es suficiente con la representación gráfica de los diagramas de casos de uso, es por ello que se realiza la expansión de los mismos, donde se muestra más detalladamente sus funcionalidades, logrando así un mejor entendimiento de los mismos.

Caso de uso	Cargar archivo de escena.
Actores	Usuario.

Resumen	El caso de uso inicia cuando el usuario desea cargar una escena, esta es interpretada por el sistema que la visualiza terminando así el caso de uso.
Referencia	RF-1
Precondiciones	Existe un archivo con extensión .scene.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción “Cargar”	2. Muestra la interfaz “FindDialog” para buscar la escena.
3. Especifica el archivo a cargar.	4. Verifica el archivo. 5. Verifica la existencia de un entorno virtual cargado anteriormente. 6. Verifica recursos. 7. Construye el entorno virtual y el árbol de escena. 8. Termina caso de uso.

Flujos alternos.	
“Archivo no válido”	
Acción del actor	**Respuesta del sistema**
	4a. Muestra un mensaje de error “El archivo no es válido”
“Existe entorno virtual cargado”	
Acción del actor	**Respuesta del sistema**
	5a.Descarga el entorno virtual existente 5b.Regresa al paso 6
“Verifica recursos”	
Acción del actor	**Respuesta del sistema**
	6a.Muestra mensaje “Error debe definir una lista de recursos”

Pos condiciones	El entorno virtual queda cargado.
Prioridad	Crítico.

Tabla 1 Caso de Uso, Cargar archivo de escena.

Caso de uso	Seleccionar los objetos en la escena 3D.
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario desea seleccionar un elemento de la escena cargada. El sistema resalta el objeto finalizando el caso de uso.
Referencia	RF-2

Precondiciones	Debe haberse cargado una escena.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario pulsa “clic” izquierdo encima del objeto 3d.	2. Resalta el objeto con una textura blanca y lo despliega en el árbol de escena. 3. Termina así el caso de uso
Pos condiciones	El elemento queda seleccionado.
Prioridad.	Crítico.

Tabla 2. Caso de Uso, seleccionar los objetos en la escena 3D.

Caso de uso	Mover la cámara.
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario desea ubicar la cámara en un lugar específico de la escena 3d. El sistema posiciona la cámara en dicho lugar terminando así el caso de uso
Referencia	RF-3
Precondiciones	Debe haberse cargado una escena.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema.**
1. Se pulsa la tecla de mover la cámara, en la dirección deseada.	2. Verifica la tecla pulsada. 3. Mueve la cámara en la dirección de la tecla correspondiente. 4. Termina el caso de uso
Flujos alternos.	
“Tecla pulsada no es las de mover la cámara”	
Acción del actor	**Respuesta del sistema.**
	2a. No mueve la cámara.

Pos condiciones	La cámara queda ubicada en la última posición en la que se soltó la tecla.
Prioridad.	Crítico.

Tabla 3 ***Caso de Uso, Mover cámara.***

Caso de uso	Rotar cámara.	
Actores	Usuario.	
Resumen	El caso de uso inicia cuando el usuario desea orientar la cámara hacia otra dirección. Y el sistema la rota hacia dicho lugar terminando así el caso de uso.	
Referencia	RF-4	
Precondiciones	Debe haberse cargado una escena.	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. Se presiona “clic” derecho y se arrastra el mouse en la dirección deseada.		2. Se rota la cámara en la dirección deseada por el usuario. 3. Termina el caso de uso.
Pos condiciones	La cámara queda orientada en el sentido de la última posición donde se soltó la tecla.	
Prioridad.	Crítico.	

Tabla 4. Caso de Uso, Rotar cámara.

Caso de uso	Insertar nodos en la escena.
Actores	Usuario.
Resumen	El caso de uso comienza cuando el usuario desea insertar nuevos nodos en la escena. Y el sistema agrega dichos nodos al árbol de escena. Terminando así el caso de uso.
Referencia	RF-5

Precondiciones	Debe haberse cargado una escena.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. Pulsa “clic” izquierdo encima del nodo correspondiente en el cuadro “Árbol de escena”	2. El sistema resalta el nodo como seleccionado.
3. Pulsa “clic” derecho sobre el nodo seleccionado en la ventana árbol de escena.	4. Muestra menú de acciones.
5. Selecciona la opción “insertar”	6. Muestra submenú de nodos a insertar.
7. Selecciona el nodo que desea insertar.	8. El sistema inserta el nodo. 9. Termina caso de uso.
Pos condiciones	Se inserta un nuevo nodo en la escena.
Prioridad.	Crítico.

Tabla 5. ***Caso de Uso, Insertar nodos en la escena.***

Caso de uso	Editar configuraciones de los elementos.
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario desea editar las configuraciones en la escena. El sistema realiza estas configuraciones y termina el caso de uso.
Referencia	RF-6
Precondiciones	Debe haberse cargado una escena.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. Se selecciona un objeto dentro de la escena.	2. Invoca el caso de uso “Seleccionar los objetos en la escena 3D”.
3. Pulsa “clic” derecho sobre el objeto seleccionado dentro de la escena 3d.	4. El sistema muestra menú de acciones a realizar.
5. El usuario selecciona una opción. • “mover” (Ver sección Modifica posición).	

• “rotar” (Ver sección Modificar orientación). • “escalar” (Ver sección Escalar los objetos). • “ocultar” (Ver sección Ocultar objeto). • “mostrar todo” (Ver sección Mostrar los objetos que estén ocultos.) • “eliminar” (Ver sección Eliminar objetos).	
Pos condiciones	EL objeto queda seleccionado.
Prioridad.	Crítico.

Tabla 6. ***Caso de Uso, Editar configuraciones de los elementos.***

Sección	Modificar la posición.	
Resumen	El caso de uso inicia cuando el usuario desea cambiar la posición de un objeto seleccionado. El sistema lo ubica en la posición deseada por el usuario y termina el caso de uso.	
Referencia	RF-6.1	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
		1. El sistema muestra los manipuladores.
2. Se posiciona el mouse sobre el manipulador correspondiente al eje en que se desea el movimiento.		3. El sistema resalta el eje sobre el que se encuentra el mouse.
4. Pulsa “clic” izquierdo y arrastra el mouse en la dirección del eje resaltado.		5. El sistema mueve el objeto en la dirección seleccionada hasta que se suelte el “clic” 6. Termina el caso de uso.
Pos condiciones	El objeto queda ubicado en la última posición.	

Tabla 7. Sección, Modificar la posición.

Sección	Modificar orientación.
Resumen	El caso de uso inicia cuando el usuario desea cambiar la orientación de un objeto seleccionado. El sistema lo rota en la dirección deseada por el usuario y termina el caso de uso.

Referencia	RF-6.2	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
		1. El sistema muestra los manipuladores.
2. Se posiciona el mouse sobre el manipulador correspondiente al eje en que se desea rotar el objeto.		3. El sistema resalta el eje sobre el que se encuentra el mouse.
4. Pulsa “clic” izquierdo y arrastra el mouse perpendicular a la dirección del eje resaltado.		5. El sistema rota el objeto en la dirección seleccionada hasta que se suelte el “clic” 6. Termina el caso de uso.
Pos condiciones	El objeto queda orientado.	

Tabla 8. Sección, Modificar orientación.

Sección	Escalar los objetos.	
Resumen	El caso de uso inicia cuando el usuario desea cambiar el tamaño de un objeto seleccionado. El sistema lo escala en la dirección deseada por el usuario y termina el caso de uso.	
Referencia	RF-6.3	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
		1. El sistema muestra los manipuladores.
2. Se posiciona el mouse sobre el manipulador correspondiente al eje en que se desea escalar el objeto.		3. El sistema resalta el eje sobre el que se encuentra el mouse.
4. Pulsa “clic” izquierdo y arrastra el mouse en la dirección del eje resaltado.		5. El sistema escala el objeto en la dirección seleccionada hasta que se suelte el “clic” 6. Termina el caso de uso.
Pos condiciones	El objeto queda escalado.	

Tabla 9. Sección, Escalar los objetos.

Sección	Ocultar objeto.
Resumen	El caso de uso inicia cuando el usuario desea ocultar un objeto seleccionado. El sistema lo hace invisible y termina el caso de uso.

Referencia	RF-6.4	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
		1. El sistema oculta el objeto. 2. Termina el caso de uso.
Pos condiciones	El objeto no se muestra en la escena.	

Tabla 10. Sección, Ocultar objeto.

Sección	Mostrar los objetos que estén ocultos.	
Resumen	El caso de uso inicia cuando el usuario desea hacer visible todos los objetos ocultos. El sistema los hace visibles y termina el caso de uso.	
Referencia	RF-6.5	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
		1. El sistema hace visible todos los objetos ocultos. 2. Termina caso de uso.
Pos condiciones	Los objetos se hacen visibles en la escena.	

Tabla 11. Sección, Mostrar los objetos que estén ocultos.

Sección	Eliminar objetos.	
Actores	Usuario.	
Resumen	El caso de uso inicia cuando el usuario desea eliminar un objeto seleccionado. El sistema lo elimina y termina el caso de uso.	
Referencia	RF-6.6	
Precondiciones	Debe haberse seleccionado un objeto.	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
		1. El sistema borra el objeto. 2. Termina el caso de uso.
Pos condiciones	El objeto se elimina de la escena.	

Tabla 12. Sección, Eliminar objetos.

Caso de uso	Buscar un objeto específico en la escena.

Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario desea buscar un objeto seleccionado. El sistema lo muestra y termina el caso de uso.
Referencia	RF-7
Precondiciones	Debe haber una escena cargada.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. Pulsa “clic” izquierdo encima del nodo correspondiente en el cuadro “Árbol de escena”	2. El sistema resalta el nodo como seleccionado.
3. Pulsa “clic” derecho sobre el nodo seleccionado en la ventana árbol de escena.	4. Muestra menú de acciones.
5. Selecciona la opción “Ver objeto”	6. El sistema mueve la cámara hasta la posición del objeto seleccionado y hace visible al objeto. 7. Termina el caso de uso.

Pos condiciones	La cámara se mueve hasta el objeto seleccionado.
Prioridad.	Crítico.

Tabla 13. Caso de Uso, Buscar un objeto específico en la escena.

Caso de uso	Definir lista de recursos.
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario desea definir la lista de recursos correspondiente a la escena que va a cargar.
Referencia	RF-7
Precondiciones	Aplicación corriendo correctamente.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la opción “recursos”.	2. Muestra la interfaz “Recursos” para definir la ruta de la carpeta con los recursos.
3. Selecciona la carpeta con los recursos.	4. Pulsa “Aceptar”. 5. Termina caso de uso.

Pos condiciones	La lista de recursos queda definida.
Prioridad.	Crítico.

Tabla 14. Caso de Uso, Definir lista de recursos.

Caso de uso	Guardar archivo escena.	
Actores	Usuario.	
Resumen	El caso de uso inicia cuando el usuario desea guardar la escena que se encuentra cargada. Y el sistema genera un fichero de extensión .scene y termina el caso de uso.	
Referencia	RF-9	
Precondiciones	Debe haberse cargado una escena.	
Flujo normal de eventos		
Acción del actor		**Respuesta del sistema**
1. Se selecciona la opción “guardar”.		2. Muestra la interfaz “Guardar escena”.
3. Especifica el nombre del archivo a guardar y su ruta.		4. Verifica una escena cargada anteriormente. 5. Genera el fichero de extensión .scene 6. Termina caso de uso.
Flujos alternos.		
“No hay una escena cargada”		
Acción del actor		**Respuesta del sistema**
		4a. Muestra mensaje de error “No existe una escena para guardar”.
Pos condiciones	El archivo queda guardado conservando las propiedades de la escena.	
Prioridad.	Crítico.	

Tabla 15. Caso de Uso, Guardar archivo escena.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SOFTWARE

Diagrama de clases de análisis

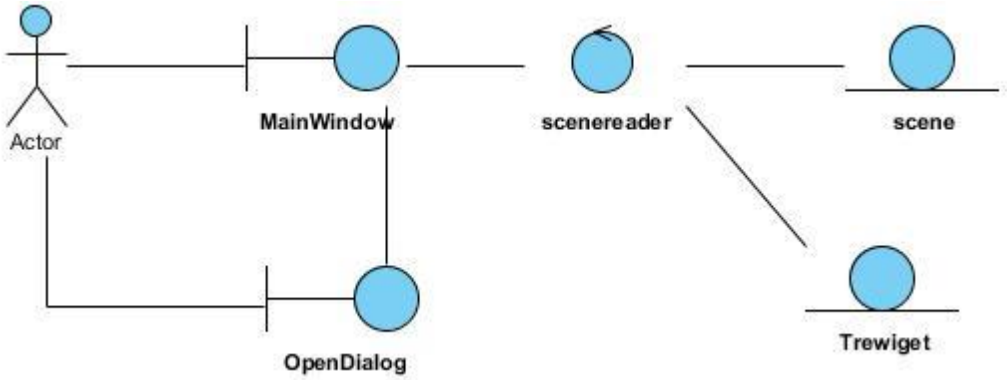


Figura 2 Diagrama de clases de análisis, Cargar escena

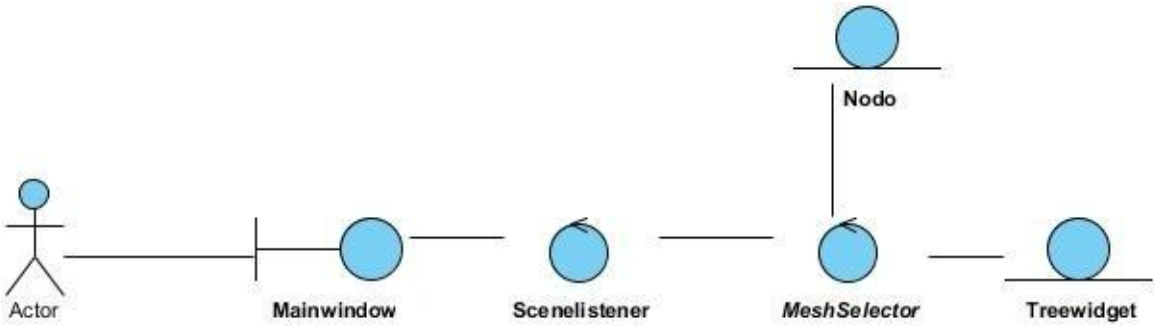


Figura 3 Diagrama de clases de análisis, seleccionar los objetos en la escena 3D.

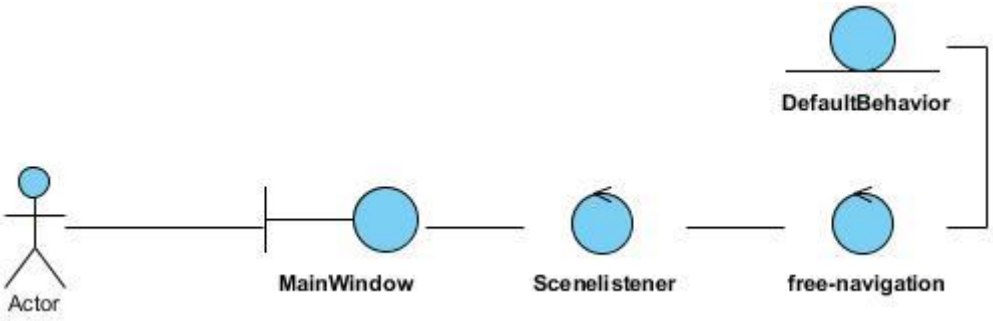


Figura 4 Diagrama de clases de análisis, Mover cámara y Rotar cámara

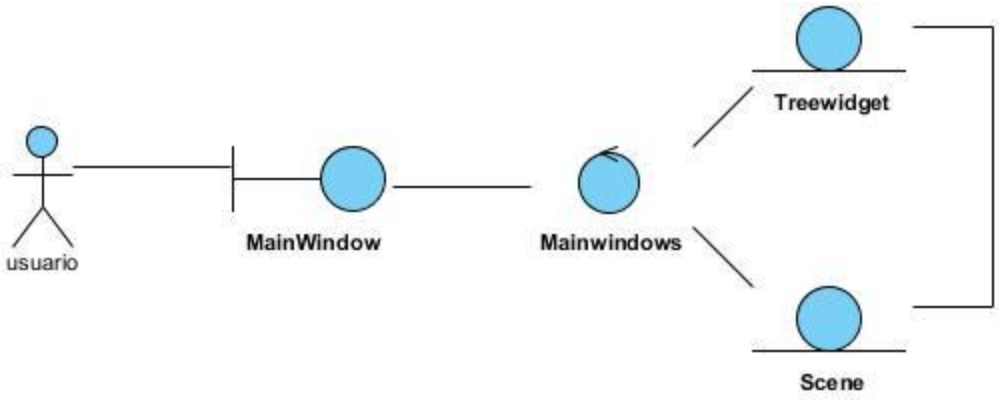


Figura 5. Diagrama de clases de análisis, Insertar nodos a la escena

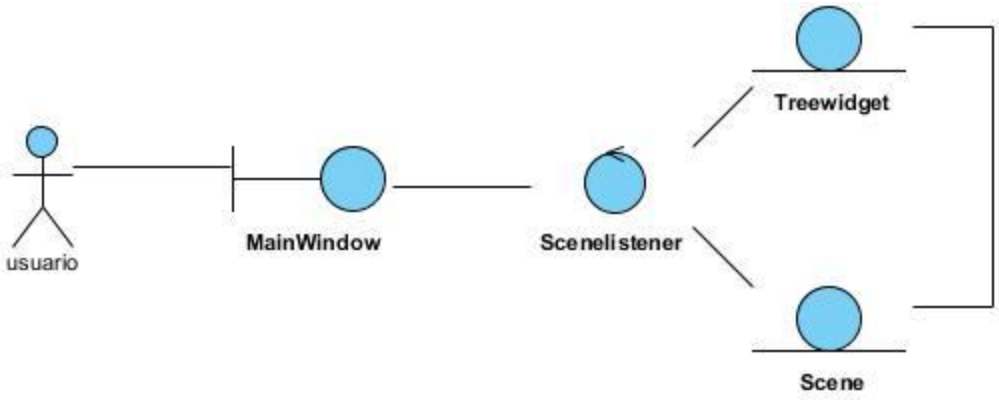


Figura 6. Diagrama de clases de análisis, Editar configuraciones de los elementos.

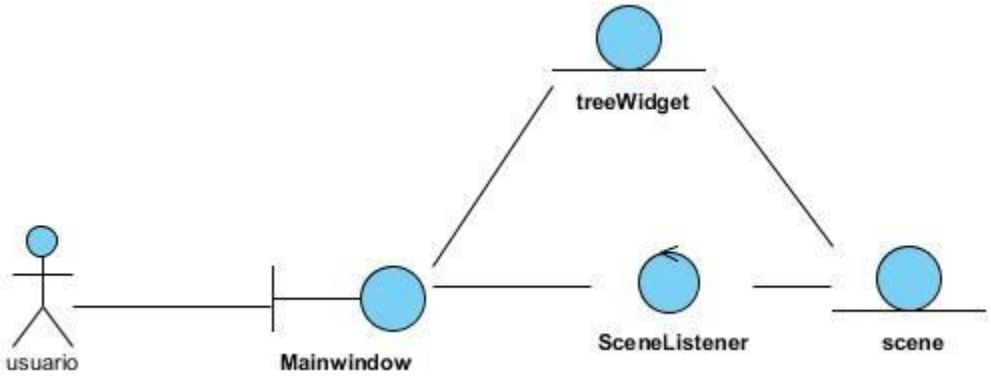


Figura 7. Diagrama de clases de análisis, Buscar un objeto específico en la escena.

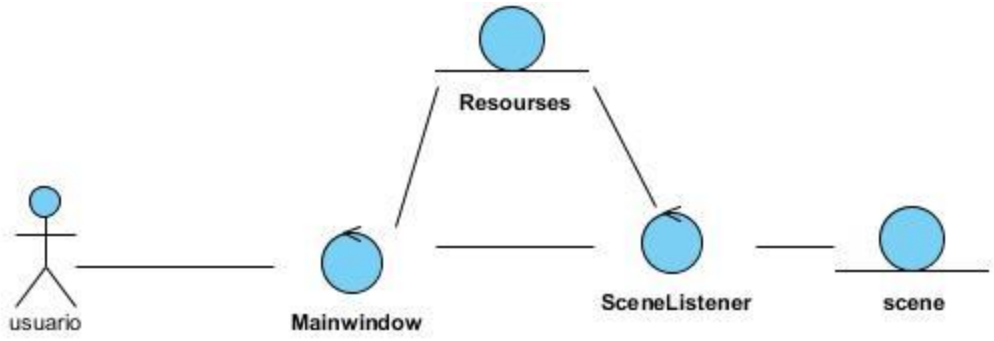


Figura 8. Diagrama de clases de análisis, Definir lista de recursos.

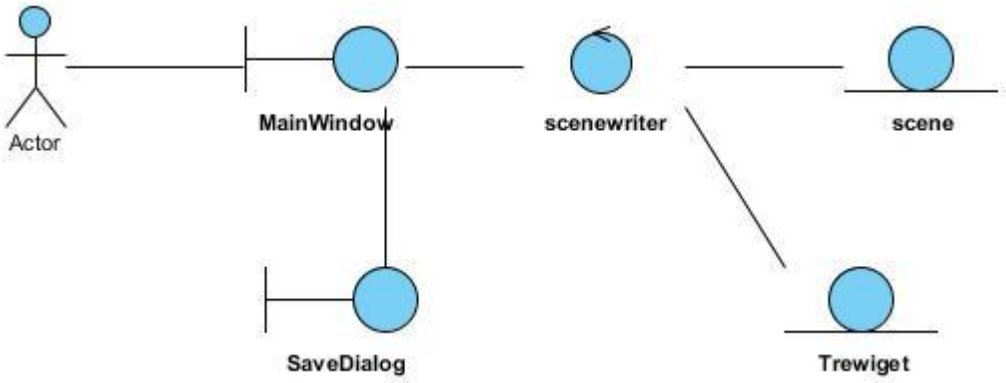


Figura 9. Diagrama de clases de análisis, Guardar archivo de escena.

Diagramas de interacción

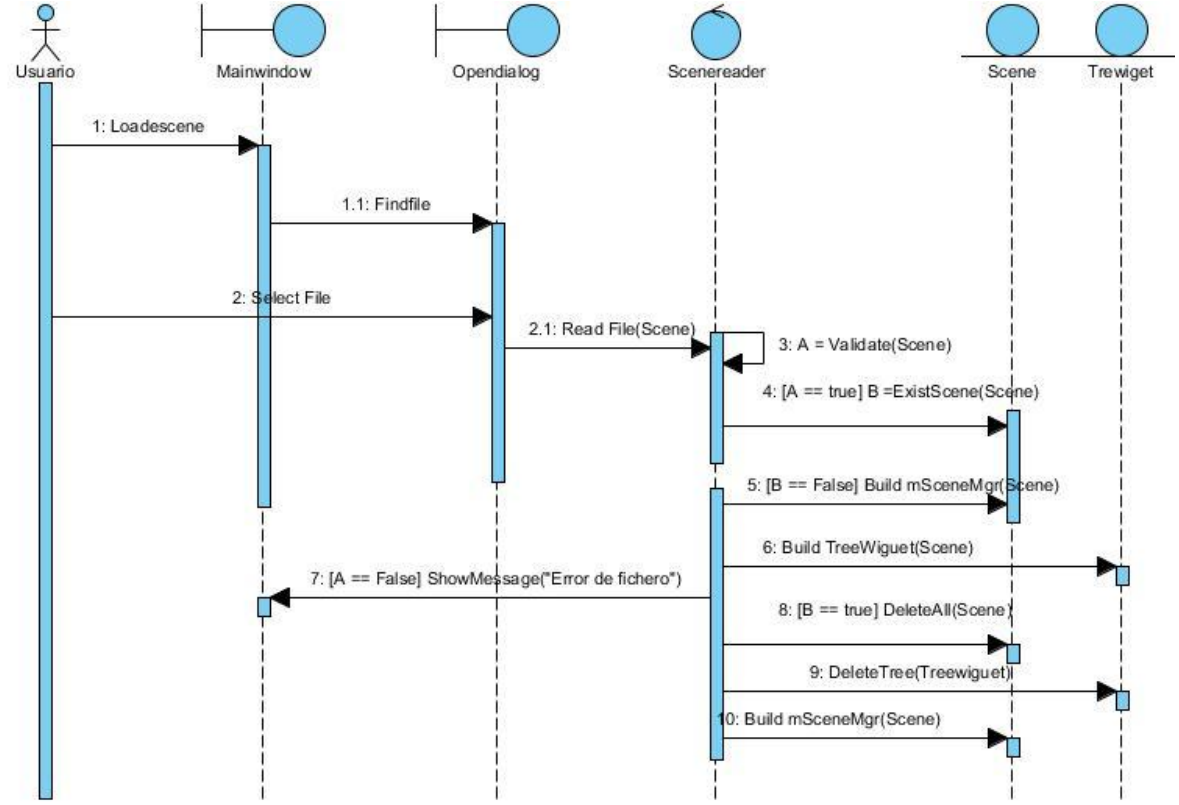


Figura 10. Diagrama de secuencia, Cargar escena

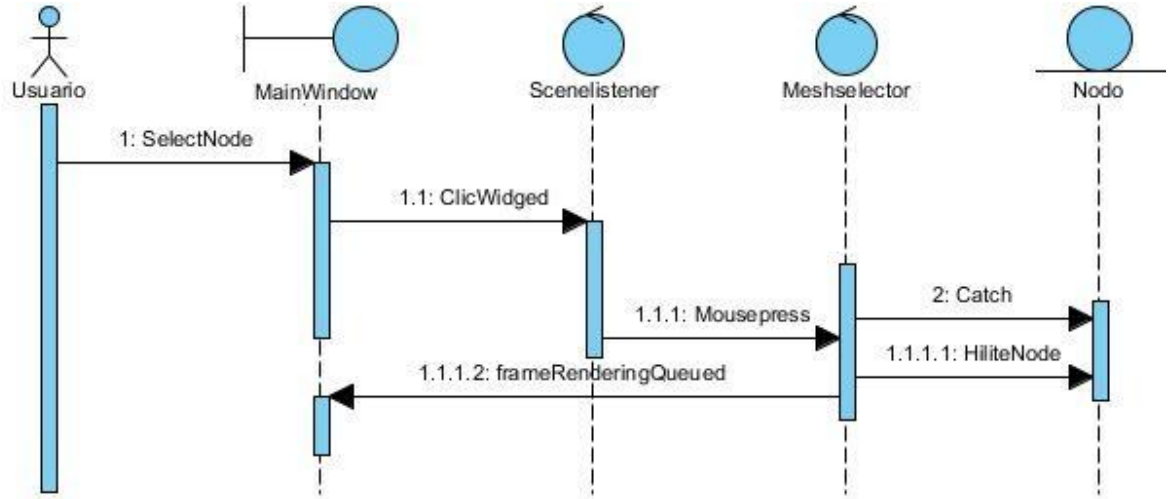


Figura 11. Diagrama de secuencia, Seleccionar los objetos en la escena 3D

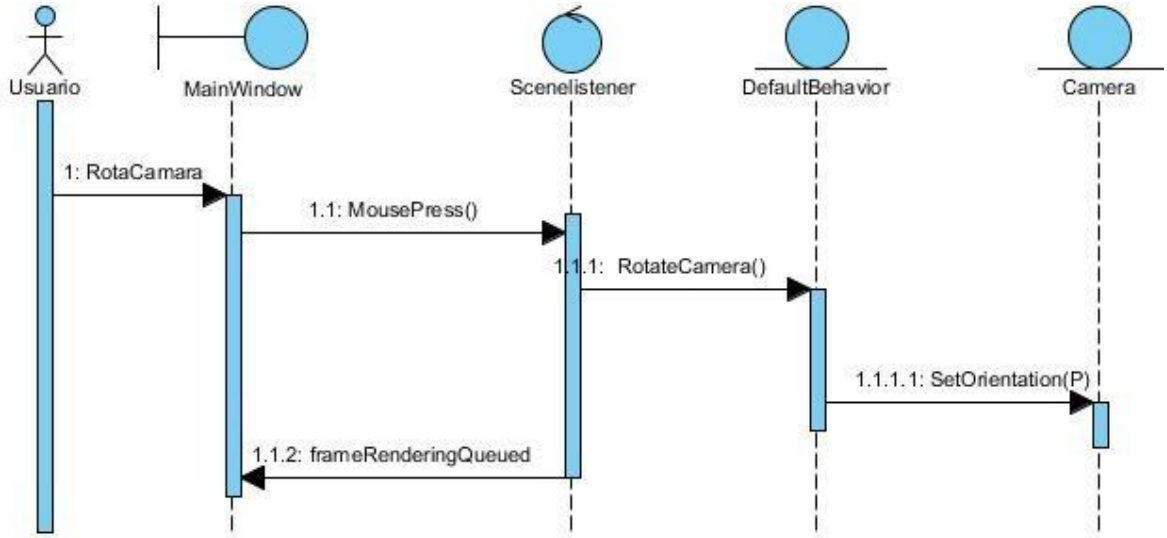


Figura 12. Diagrama de secuencia, Rotar cámara.

En el caso del caso de uso mover cámara el diagrama de secuencia es similar al de rotar. La diferencia viene dada en la modificación que se le aplica a la cámara que en el caso de mover es la "posición" a través del método "SetPosition()".

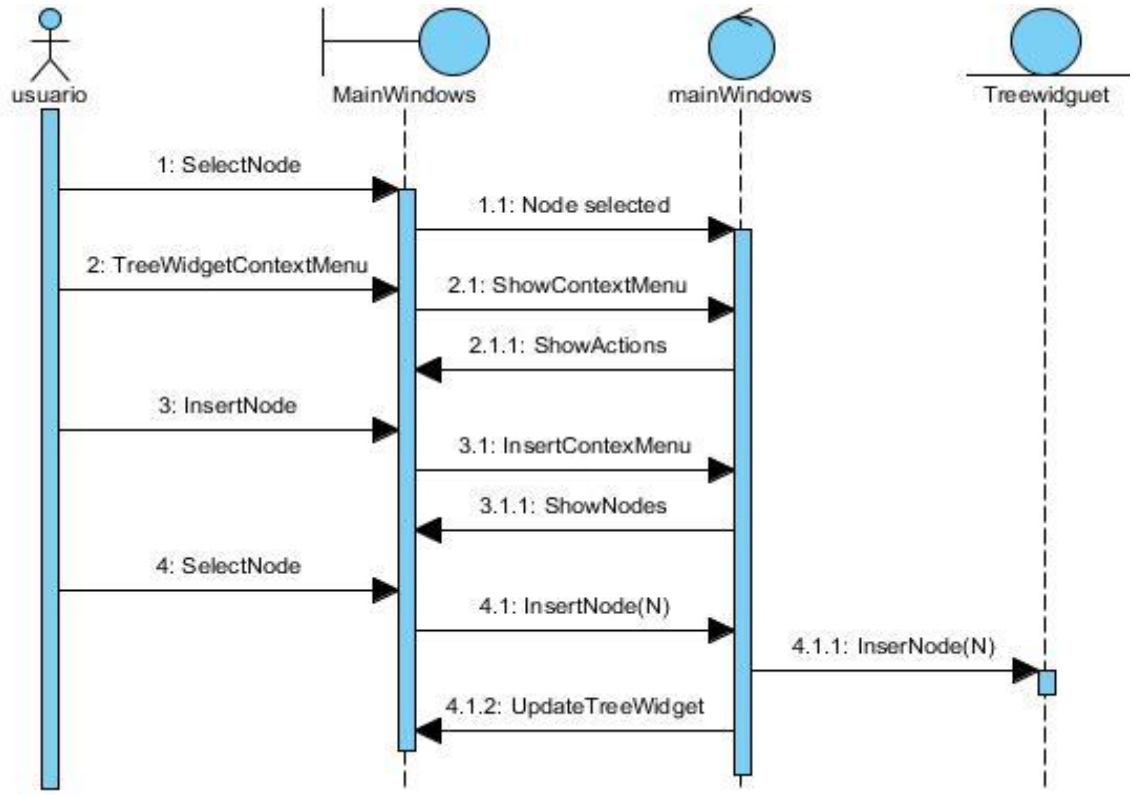


Figura 13. Diagrama de secuencia Insertar nodos en la escena.

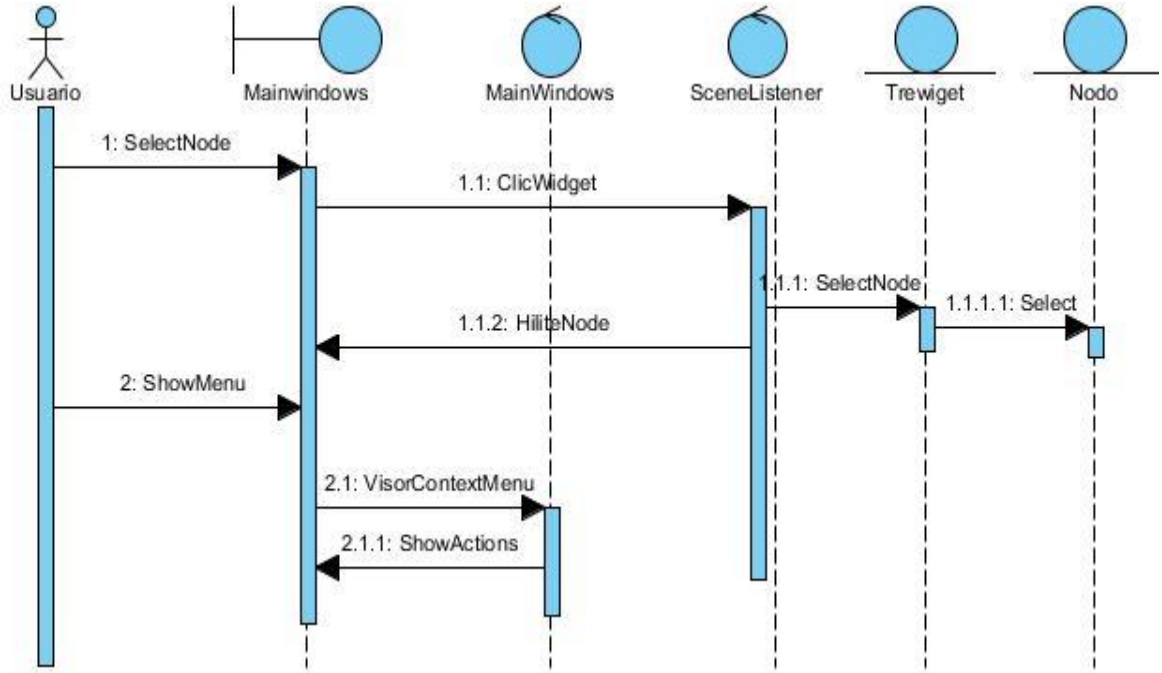


Figura 14. Diagrama de secuencia, Editar configuraciones de los elementos.

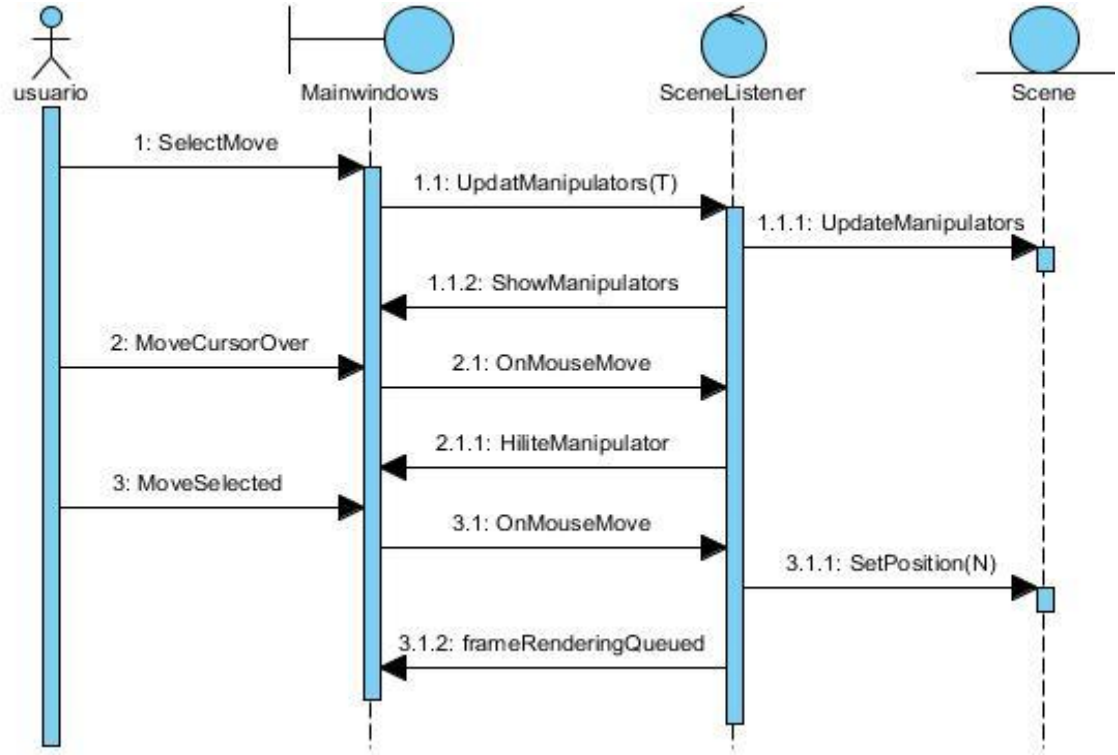


Figura 15. Diagrama de secuencia de la sesión, Modificar posición.

En el caso de las secciones “modificar orientación” y “escalar objeto” los diagramas son similares con la diferencia que los métodos invocados son el “SetOrientation()” y “Scale()” respectivamente.

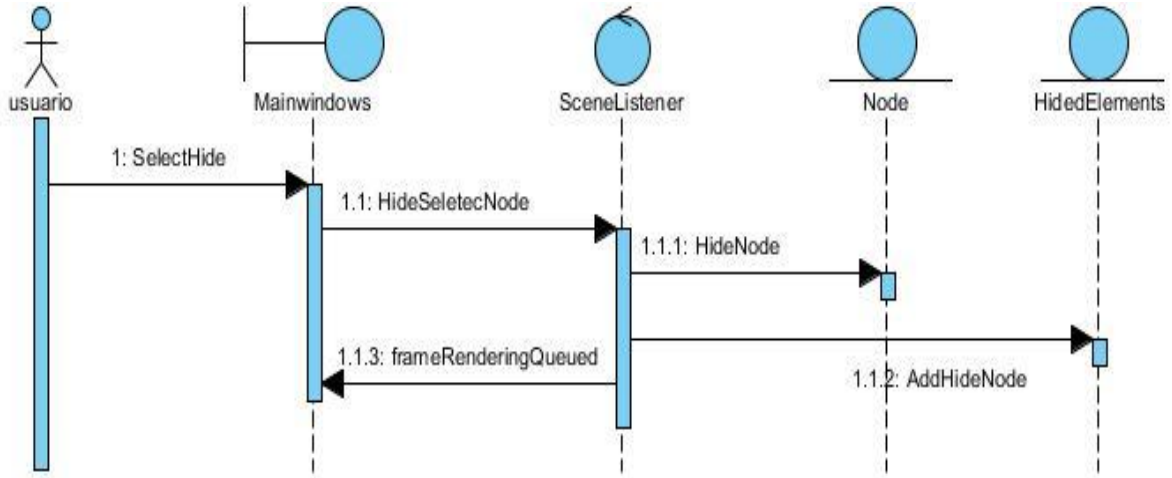


Figura 16. Diagrama de secuencia de la sesión, Ocultar objeto.

Para el caso de las secciones “mostrar objetos que estén ocultos” y “eliminar objeto” los diagramas son similares al de la figura 21 con la diferencia que los métodos invocados para casa sección son “ShowAll()” y “DeleteNode()” respectivamente.

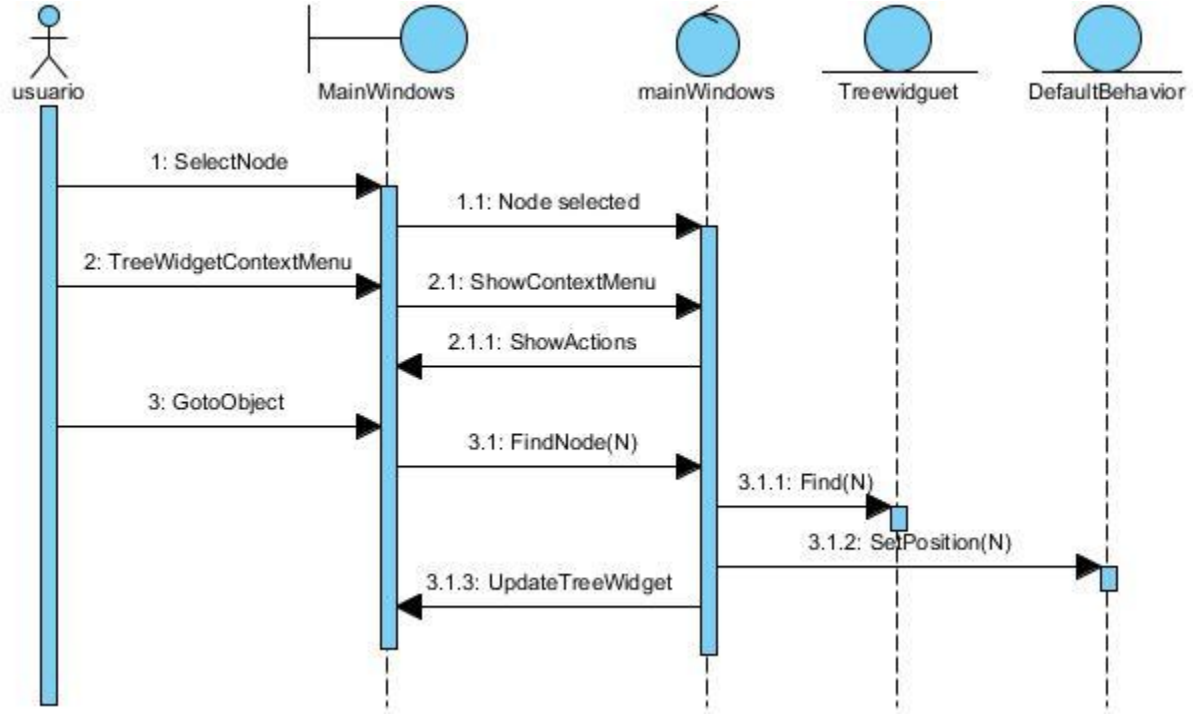


Figura 17. Diagrama de secuencia, Buscar un objeto en la escena 3D.

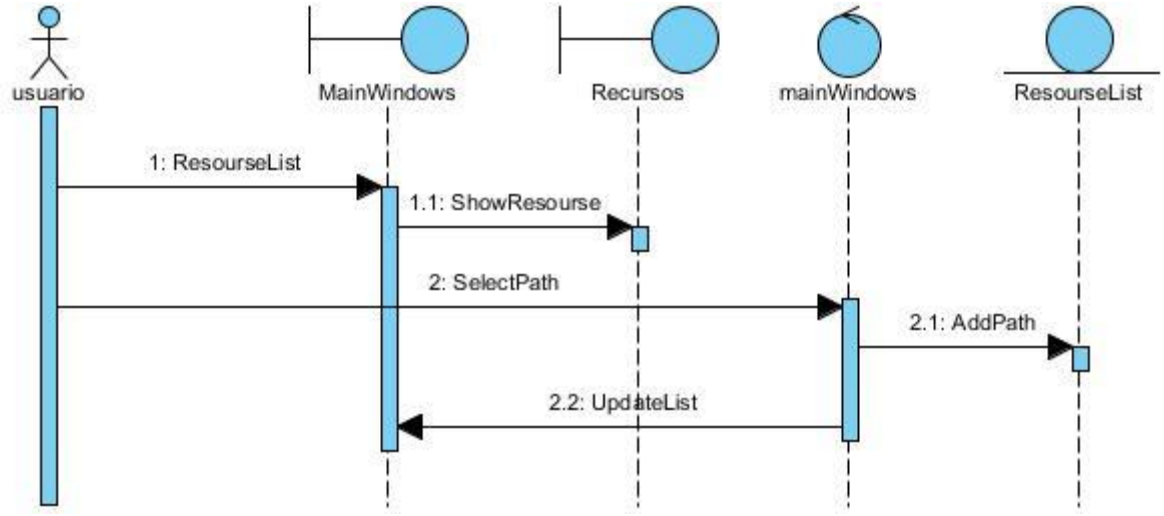


Figura 18. Diagrama de secuencia, Definir lista de recursos

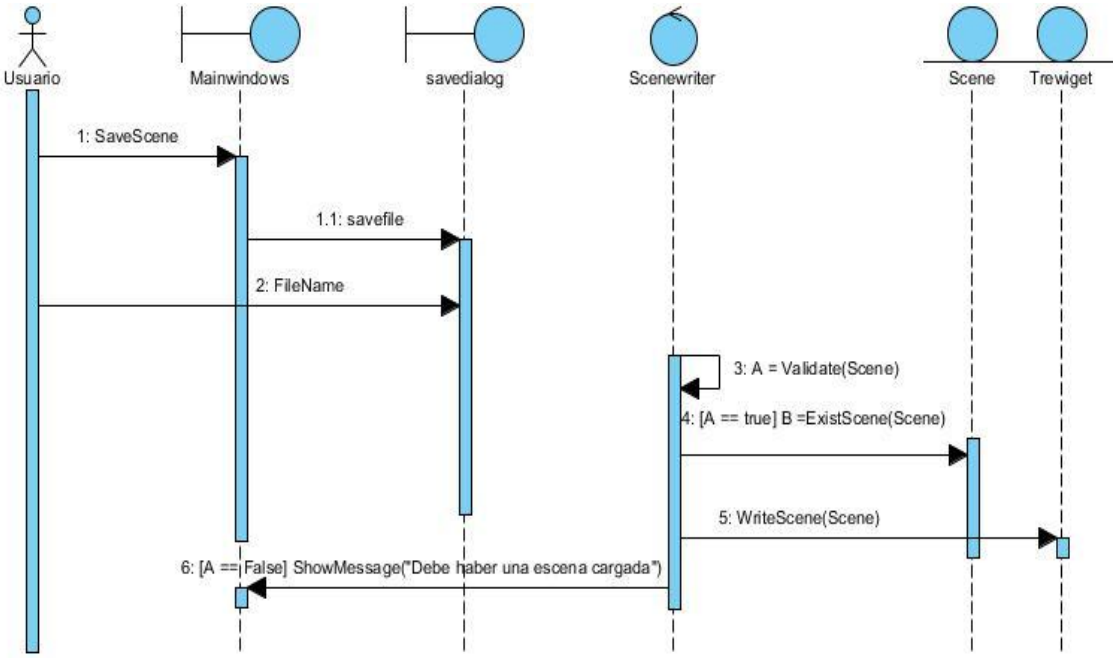


Figura 19. Diagrama de secuencia, Salvar escena.

Diagrama de clases del diseño.

Dada la extensión del diagrama de clases del diseño el mismo no se muestra a continuación. Con el objetivo de facilitar su análisis se ofrece una descripción de las clases y para realizar un análisis más detallado el diagrama se ubica en la sección de anexos. Ver ANEXO2

Descripción de las clases del diseño.

Nombre: SceneListener		
Tipo de clase: Controladora		
Atributo		Tipo
mGuiMgr		GuiManager
mSelMgr		MeshSelector
mScene		Componet_Loadescene
DefaultBehavior		DefaultNavigationBehavior
mSelectedEntity		Ogre::Entity
mSelectedNode		Ogre::SceneNode
Posicioner		GUISystem::Button
Rotater		GUISystem::Button
Scaler		GUISystem::Button
NodeXYZ		Ogre::SceneNode
Mcurrenttool		EditorTool
mCurrentAxis		Axis
-HiliteAxis		Axis
LastMousePos		QPoint
Para cada responsabilidad:		
Nombre	GetEsceneLoader	
Descripción	Retorna el atributo mScene.	
Nombre	ResourseUpdate	
Descripción	Actualiza los recursos de OGRE añadiendo los que se definieron nuevos.	
Nombre	LoadScene	
Descripción	Lee de una dirección el fichero .scene y construye la escena	

	en el visor.
Nombre	UpdateLastImpactPoint
Descripción	Actualiza el punto de impacto del rayo en el momento de mover un objeto.
Nombre	CreateManipulators
Descripción	Construye los manipuladores que modificarán a los objetos en la escena.
Nombre	ShowManipulator
Descripción	Muestra los manipuladores en la escena.
Nombre	HiliteManipulator
Descripción	Resalta el manipulador por el cual pase el cursor.
Nombre	AdjustManipulators
Descripción	Ajusta el tamaño del manipulador de acuerdo a la distancia de la cámara y del objeto seleccionado.
Nombre	DestroyManipulators
Descripción	Destruye los manipuladores creados.
Nombre	UpdateManipulatorsPos
Descripción	Actualiza la posición del manipulador en la escena.
Nombre	MoveNodeAxis
Descripción	Mueve el nodo seleccionado.
Nombre	ScaleNodeAxis
Descripción	Escala el nodo seleccionado.
Nombre	RotateNodeAxis
Descripción	Rota el nodo seleccionado.
Nombre	onKeyPress
Descripción	Captura y procesa todos los eventos producidos al presionar

	una tecla.
Nombre	onKeyRelease
Descripción	Captura y procesa todos los eventos producidos al soltar una tecla.
Nombre	onMousePress
Descripción	Captura y procesa todos los eventos producidos al presionar un botón del mouse.
Nombre	onMouseRelease
Descripción	Captura y procesa todos los eventos producidos al soltar un botón del mouse.
Nombre	onMouseMove
Descripción	Captura y procesa todos los eventos producidos al mover el mouse.
Nombre	FindObject
Descripción	Devuelve el objeto que se desea buscar y en caso de encontrarse invisible lo hace visible.
Nombre	RecursiveFindObject
Descripción	Realiza una búsqueda recursiva en el árbol de escena.
Nombre	frameRenderingQueued
Descripción	Se llama cada vez que se pinta un fotograma de la aplicación.
Nombre	frameStarted
Descripción	Retorna *true* si el fotograma va a pintarse sin error en cualquier otro caso *false*.
Nombre	frameEnded
Descripción	Retorna *true* si el fotograma ha terminado de pintarse sin error en cualquier otro caso *false*.
Nombre	setupContent

Descripción	Carga todo el contenido de la aplicación (escena, cámara, sistema de interface gráfica, etc.).
Nombre	CreateCamera
Descripción	Crea la cámara que mostrará la escena en el visor.
Nombre	CreateViewports
Descripción	Construye la ventana en la cual se visualizará la escena que sea recogida por la cámara.
Nombre	setupTools
Descripción	Construye la herramienta que será utilizada para realizar la técnica de raycast.
Nombre	LoadResourceElements
Descripción	Carga los recursos que serán *renderizados* y mostrados en la escena.
Nombre	buttonHit
Descripción	Este método se ejecuta cada vez que se da un clic en un botón de interface dentro de la escena 3d.
Nombre	CreateGuiSystem
Descripción	Construye los recursos que se utilizan para manipular los elementos de la interface de usuario en la escena 3d.
Nombre	HideSelectedNode
Descripción	Oculta el nodo seleccionado.
Nombre	DeleteSelectedNode
Descripción	Elimina el nodo seleccionado.
Nombre	ShowAllNodes
Descripción	Hace visible todos los nodos que estén ocultos.

Tabla 16: Descripción de la clase de diseño Scenelistener.

Nombre: MainWindows

Tipo de clase: Controladora	
Atributo	Tipo
dockWidget	QDockWidget
sceneWidget	QTreeWidget
menuBar	QMenuBar
readerScene	SceneReader
readerTempl	TemplateReader
writerScene	SceneWriter
Visor	OgreWidget
mSceneListener	SceneListener

Para cada responsabilidad.	
Nombre.	CreateDockwiget
Descripción.	Construye la ventana donde se mostrará el árbol de escena.
Nombre.	CreateSceneWiget
Descripción.	Construye la ventana donde se visualizará la escena 3d en la aplicación.
Nombre.	Createtrewiguet
Descripción.	Construye la estructura de datos que visualizará el árbol de escena.
Nombre.	CreateMenu
Descripción.	Construye los menús con los que contará la aplicación.
Nombre.	ConnectActions
Descripción.	Conecta cada uno de las *action* de la clase, con los métodos correspondientes.
Nombre.	findObjet

Descripción.	Realiza una búsqueda por el treewidget hasta encontrar la etiqueta deseada.
Nombre.	deleteItem
Descripción.	Elimina el nodo seleccionado.
Nombre.	HideNode
Descripción.	Oculta el nodo seleccionado.
Nombre.	ShowAllNodes
Descripción.	Hace visible todos los nodos seleccionados.
Nombre.	ShowMoveManipulator
Descripción.	Muestra los manipuladores correspondientes a la acción de “mover” el nodo seleccionado.
Nombre.	ShowRotateManipulator
Descripción.	Muestra los manipuladores correspondientes a la acción de “rotar” el nodo seleccionado.
Nombre.	ShowScaleManipulator
Descripción.	Muestra los manipuladores correspondientes a la acción de “escalar” el nodo seleccionado.

Tabla 17. Descripción de la clase de diseño MainWindows.

Nombre: scenewriter	
Tipo de clase: entidad	
Atributo	Tipo
XML	QXmlStreamWriter
treeWidget	QTreeWidget
atributeName	Qstring
tagName	Qstring
validTemplate	Bool

Para cada responsabilidad.	
Nombre.	SceneWriter
Descripción.	Construye la estructura que escribirá en el fichero.
Nombre.	writeFile
Descripción.	Devuelve *true* si el proceso de escritura se concluye de forma correcta, en cualquier otro caso retorna *false*.
Nombre.	recursiveWriteItem
Descripción.	Recorre recursivamente el árbol de escena y escribe cada uno de los nodos en el fichero con la estructura de los XML.

Tabla 18. Descripción de la clase de diseño scenewriter.

Nombre: SceneReader	
Tipo de clase: entidad	
Atributo	Tipo
XML	QXmlStreamReader
treeWidget	QTreeWidget
Para cada responsabilidad.	
Nombre.	SceneReader
Descripción.	Construye la estructura que leerá del fichero.
Nombre.	Read
Descripción.	Devuelve verdadero si el proceso de lectura se concluye de forma correcta, en cualquier otro caso retorna falso.
Nombre.	errorString
Descripción.	Retorna el tipo de error durante el proceso de lectura en caso de haberlo.
Nombre.	readScene
Descripción.	Realiza la lectura del primer nodo al cual se le van a adjuntar el resto de los nodos a medida que se continua avanzando en el

	proceso de lectura.
Nombre.	recursiveReadScene
Descripción.	Recorre recursivamente XML y construye cada uno de los nodos y los adjunta al nodo padre del árbol escena.

Tabla 19. Descripción de la clase de diseño scenewriter

Nombre: Teamplatereader	
Tipo de clase: entidad	
Atributo	Tipo
XML	QXmlStreamReader
treeWidget	QTreeWidget
Directory	Qdir
Para cada responsabilidad.	
Nombre.	TemplateReader
Descripción.	Construye la estructura que leerá del fichero los títulos para llenar el menú de adicionar nodos.
Nombre.	Read
Descripción.	Devuelve verdadero si el proceso de lectura se concluye de forma correcta, en cualquier otro caso retorna falso.
Nombre.	errorString
Descripción.	Retorna el tipo de error durante el proceso de lectura en caso de haberlo.
Nombre.	readScene
Descripción.	Realiza la lectura del primer nodo al cual se le van a adjuntar el resto de los nodos a medida que se continua avanzando en el proceso de lectura de las plantillas.
Nombre.	recursiveReadScene

Descripción.	Recorre recursivamente XML de la plantilla y construye cada uno de los nodos y los adjunta al nodo padre del nodo que se adjuntara al nodo de la escena.
Nombre.	InsertItems
Descripción.	Inserta en el menú todas las plantillas que se puedan insertar en el árbol de escena.
Nombre.	readDirectory
Descripción.	Lee de la carpeta donde se encuentran las plantilla y construye una lista con todas la plantillas encontradas.
Nombre.	readForMenu
Descripción.	Agrega al menú el nombre que corresponda a la plantilla de acuerdo al que traiga definido en el campo “menuName”.
Nombre.	readForInser
Descripción.	Lee toda la plantilla y la inserta en el nodo.

Tabla 20. Descripción de la clase de diseño Teamplatereader

CAPITULO 4: IMPLEMENTACION Y PRUEBAS

Estándar de codificación.

El siguiente estándar de código fue el que se definió debido a que es el que se trabaja dentro del marco de los Laboratorios Virtuales.

Identación: una tabulación equivalente a cuatro espacios ejemplo.

```
void MainWindow::clearScene()
{
     treeWidget->clear();
}
```

Declaración:

Variables

Comienzan con la letra m, de la palabra *my* (en español: mi), para simbolizar que son variables privadas de la clase, y todas las palabras que las componen comenzarán con letra inicial mayúscula ejemplo.

```
GUISystem::GUIManager* mGuiMgr;
MeshSelector* mSelMgr;
Component_LoadScene* mScene;
```

Clases

Todas las palabras que las compongan comenzarán con letra inicial mayúscula ejemplo.

```
class SceneReader
```

Funciones

Todas las palabras que contengan se escribirán con letra inicial mayúscula.

```
void CreateNew();

void LoadScene();
```

Condicionales y ciclos: Se utilizarán las llaves solo cuando contengan más de una línea de código.

```
if (temp->text(0) == nodeName)
   valor = temp->text(1);

for (int i = 0; i < Lresourse->count();i++)

{

 Resourselist->addItem(Lresourse->at(i));

 return Resourselist;

}
```

Diagrama de componentes

Para lograr un mayor entendimiento y organización del diagrama de componentes se dividió en, una vista binaria donde se muestra la relación entre los componentes binarios, y una vista de código fuente, la cual está diseñada siguiendo la estructura física de los ficheros de código fuente.

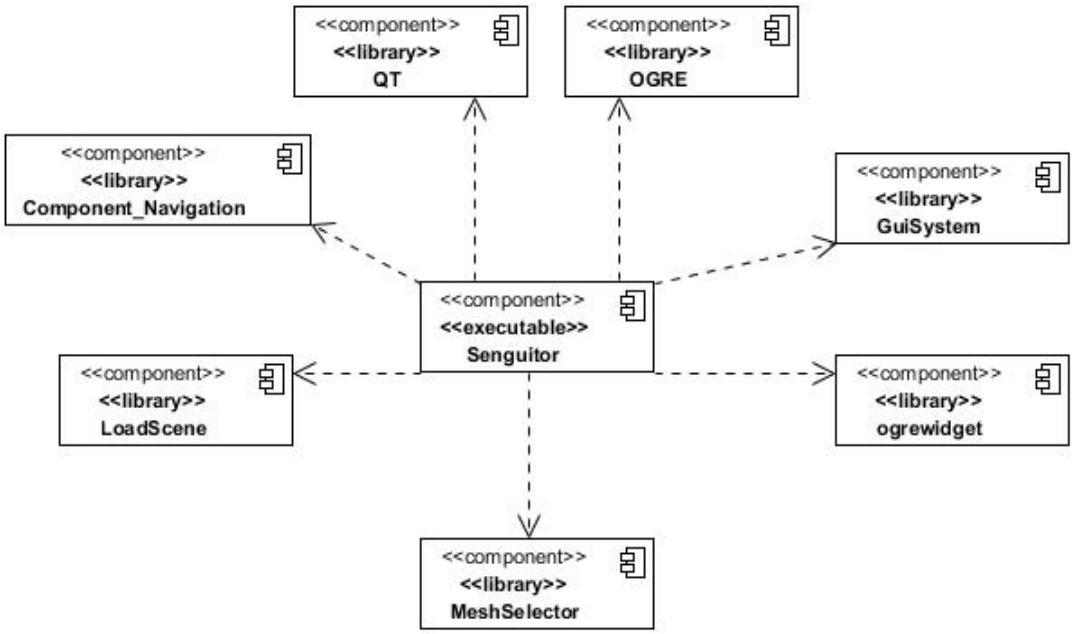


Figura 21. Diagrama de Componentes de la vista binaria.

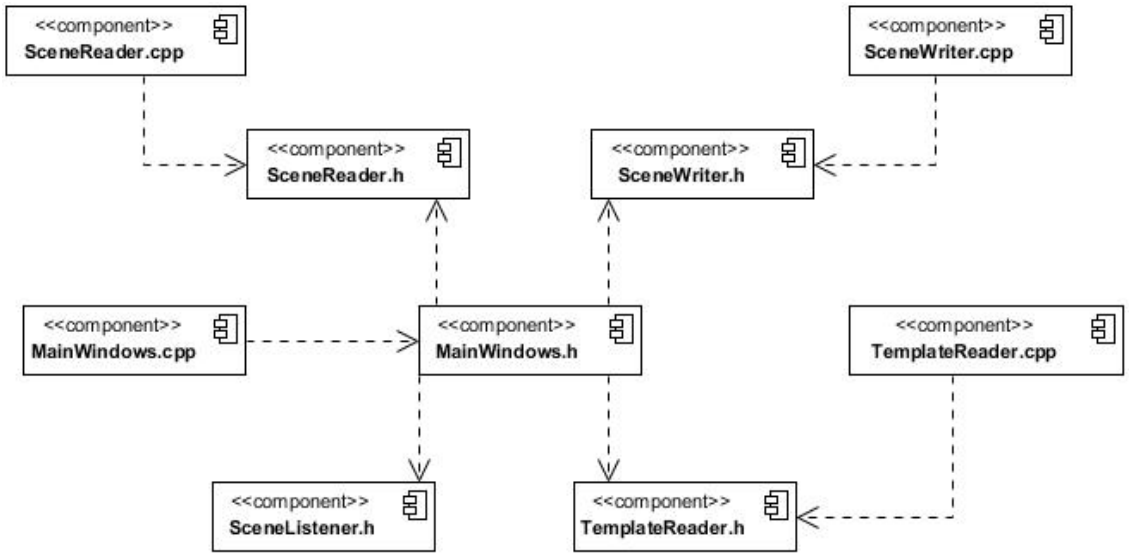


Figura 22. Diagrama de Componentes de la vista de código fuente.

Modelos de pruebas

Caso de Uso: Cargar archivo de escena.

Descripción General: Muestra en el visor la escena 3d que se desea editar.

Condiciones de Ejecución: Aplicación funcionando correctamente.

Secciones a aprobar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Cargar archivo de escena.	1.1 No hay una escena cargada.	Se selecciona la escena que se desea cargar para modificar. Y se muestra en el visor.	Seleccionar acción “Cargar”. Busca la escena que se desea cargar. Se carga el archivo correctamente.

	1.2 Hay una escena cargada.	Muestra un mensaje sobre la acción que se va a realizar.	Seleccionar acción “Cargar”. Muestra mensaje informativo, “Está a punto de Cargar una escena nueva. ¿Desea proseguir y perder los cambios que no se hayan guardados?”.

Tabla 21: Modelo de Prueba del Caso de Uso Cargar archivo de escena.

Caso de Uso: *editar configuraciones de los elementos*.

Descripción General: edita los elementos de la escena modificándolos, ocultándolos, mostrándolos o eliminándolos.

Condiciones de Ejecución: Una escena cargada correctamente.

Secciones a aprobar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. *editar configuraciones de los elementos*	1.1 Hay un objeto seleccionado.	Muestra un menú con las posibles modificaciones a las que se puede someter el objeto.	Clic derecho encima del objeto seleccionado. Seleccione la modificación que se desea aplicar. “Clic” izquierdo sobre el manipulador correspondiente al eje en el que se desea modificar el objeto Arrastrar en dirección del eje.

	1.2 No hay un objeto seleccionado.	Se muestran, en el menú, deshabilitadas las funcionalidades correspondientes a las modificaciones.	“Clic” derecho.

Tabla 22: Modelo de Prueba del Caso de Uso, editar configuraciones de los elementos

Caso de Uso: Seleccionar elementos de la escena.

Descripción General: Selecciona los elementos en la escena.

Condiciones de Ejecución: Una escena cargada correctamente.

Secciones a aprobar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Seleccionar los elementos de la escena.	1.1 No hay un objeto seleccionado.	Selecciona el elemento deseado, resaltando su textura.	Clic izquierdo encima del elemento que se desea seleccionar
	1.2 Hay un elemento seleccionado.	Resalta el nuevo elemento seleccionado retornando el anterior a la normalidad.	Clic izquierdo en el nuevo elemento a seleccionar.

Tabla 23: Modelo de Prueba del Caso de Uso Seleccionar los elementos de la escena.

Caso de Uso: Mover cámara.

Descripción General: Mueve la cámara por toda la escena en la dirección deseada.

Condiciones de Ejecución: Una escena cargada correctamente.

Secciones a aprobar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

1. Mover cámara.	1.1 Una escena cargada correctamente.	Mueve la cámara en la dirección que se desea.	Pulsa la tecla W, S, D, A para ir adelante, atrás, derecha e izquierda respectivamente.
	1.2 No hay una escena cargada.	No se mueve la cámara.	No se realiza ninguna acción independientemente de la tecla que se presione.

Tabla 24: Modelo de Prueba del Caso de Uso Mover cámara

Caso de Uso: Rotar cámara.

Descripción General: Rota la cámara en la dirección deseada.

Condiciones de Ejecución: Una escena cargada correctamente.

Secciones a aprobar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Rotar cámara.	1.1 Una escena cargada correctamente.	Rota la cámara en la dirección que se desea.	"Clic" derecho. Arrastre el mouse en la dirección que se desea rotar la cámara.
	1.2 No hay una escena cargada.	No se mueve la cámara.	No se realiza ninguna acción independientemente de la tecla que se presione.

Tabla 25: Modelo de Prueba del Caso de Uso

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Rotar cámara.	1.1 Una escena cargada correctamente.	Rota la cámara en la dirección que se desea.	"Clic" derecho. Arrastre el mouse en la dirección que se desea rotar la

			cámara.
	1.2 No hay una escena cargada.	No se mueve la cámara.	No se realiza ninguna acción independientemente de la tecla que se presione.

Tabla 26: Modelo de Prueba del Caso de Uso

Caso de Uso: Insertar nodos en la escena.

Descripción General: Inserta nodos, creados previamente, en la escena.

Condiciones de Ejecución: Una escena cargada correctamente.

Secciones a aprobar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1. Insertar nodos en la escena.	1.1 Hay una escena cargada.	Inserta un nodo nuevo en el árbol de escena.	"Clic" derecho sobre un nodo. Seleccione "insertar nodo" Seleccione el nodo que quiere insertar.
	1.2 No hay una escena Cargada	Muestra un mensaje "Debe cargar una escena"	No se realiza ninguna acción
	1.3 No existe ningún nodo a insertar.	No muestra ningún nodo en el submenú de insertar nodo.	No se realiza ninguna acción

Tabla 27: Modelo de Prueba del Caso de Uso Insertar nodos en la escena.

Consulta Realizada.

Como parte del proceso de validación de la herramienta. Se llevó a cabo un experimento con una parte del equipo de desarrollo de los Laboratorios Virtuales. En una muestra de cuatro desarrolladores se le pedía que modificaran la información de una escena sin utilizar la herramienta creada y utilizando la herramienta. En la gráfica que se muestra (Figura 23) que tiene como variable a medir el tiempo en minutos que demora la modificación de una escena. Se aprecia una disminución notable de los tiempos entre uno y otro caso.

.

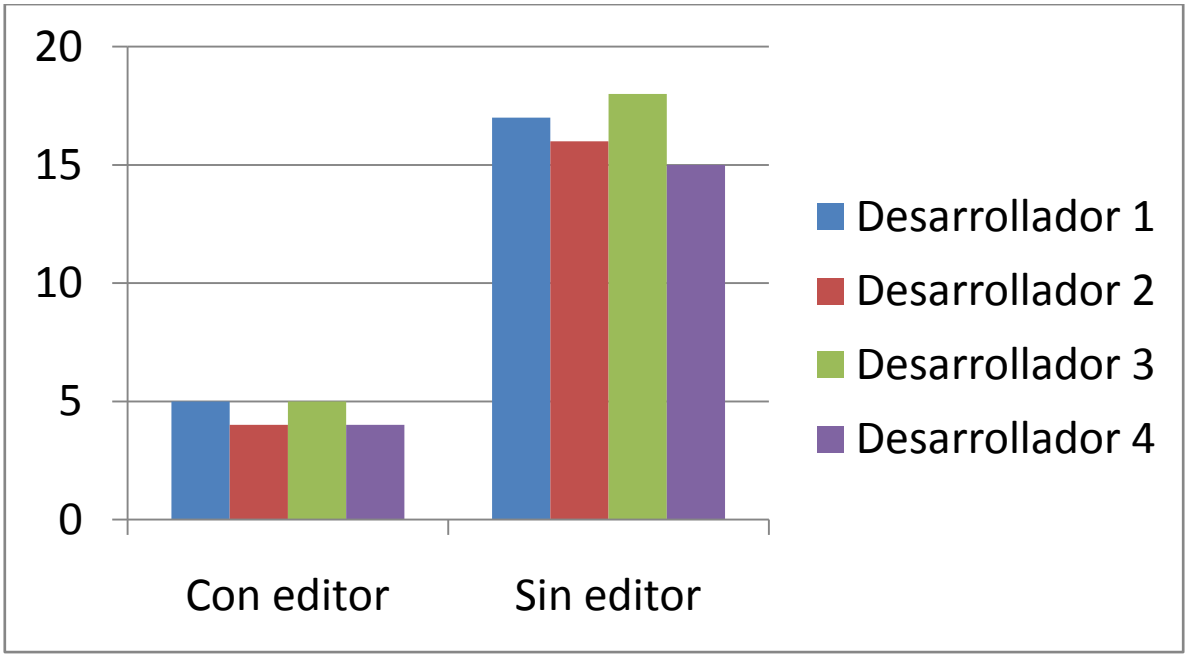


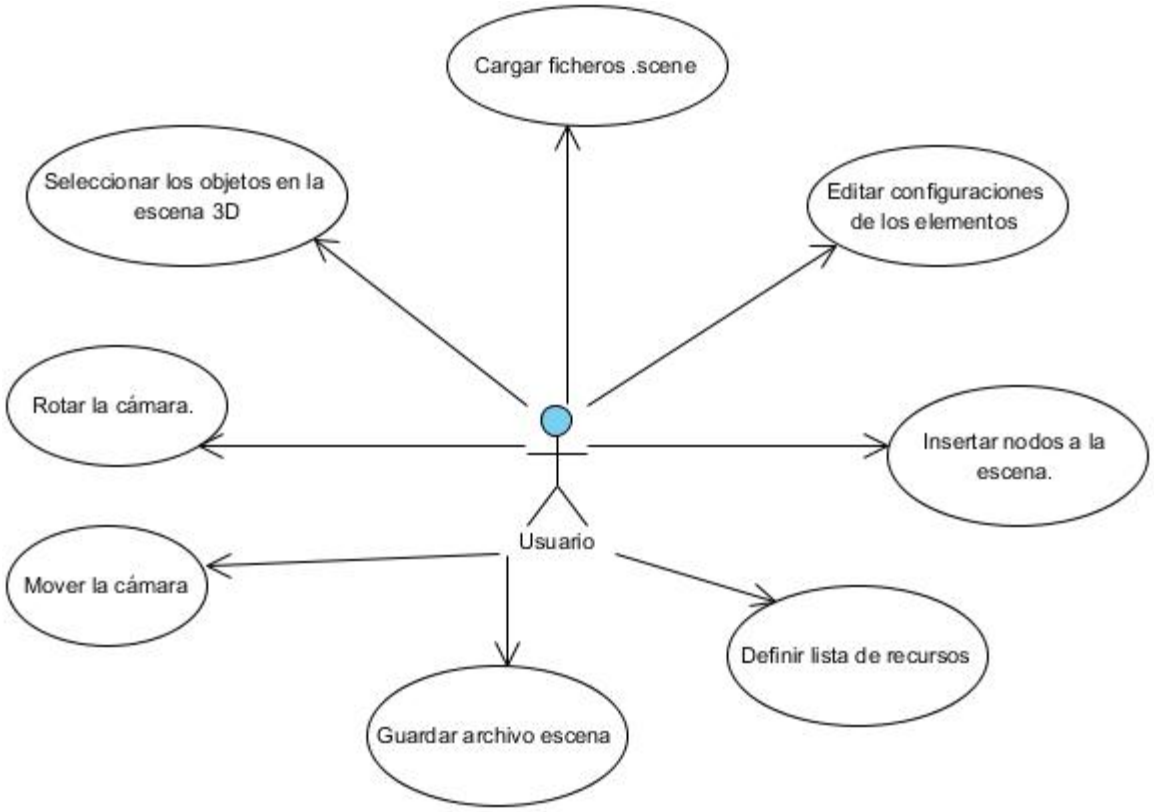
Figura 23. Encuesta realizada a los desarrolladores.

CONCLUSIONES:

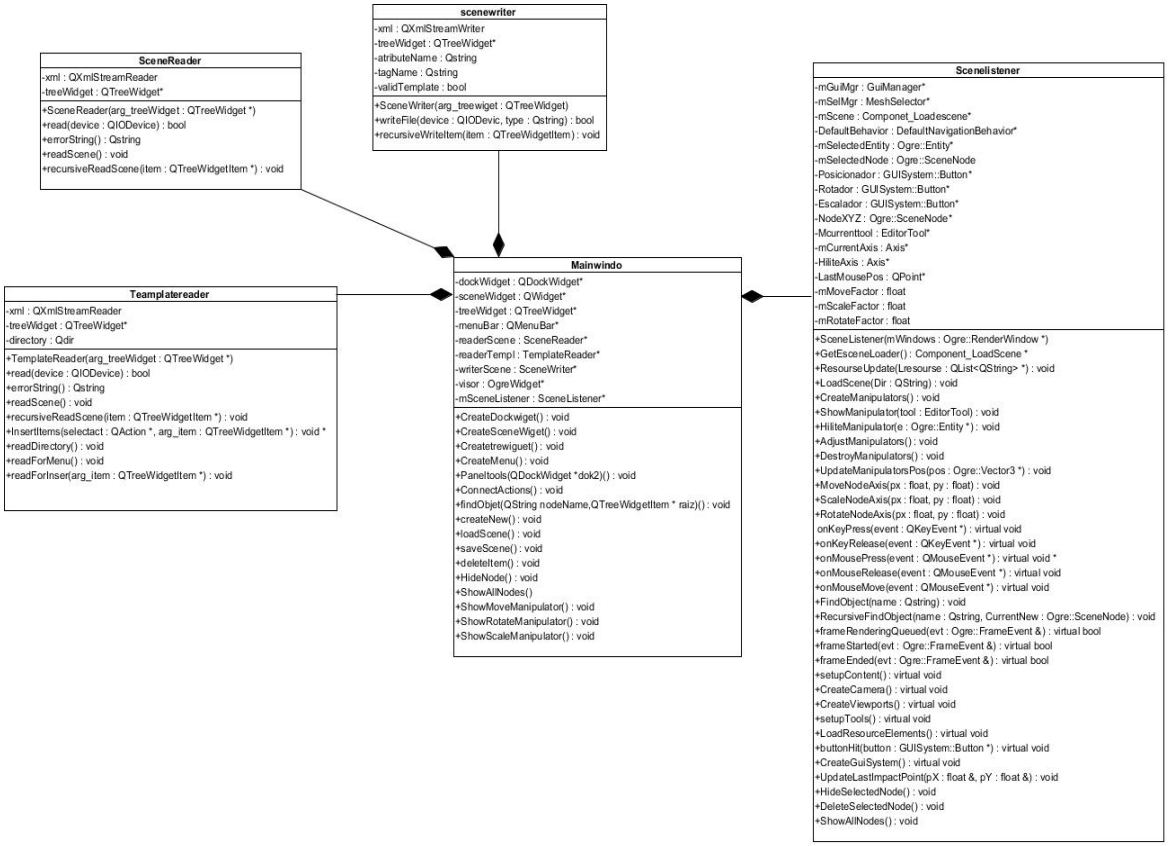
Para lograr el cumplimiento del objetivo de esta investigación, en concordancia con las necesidades del proyecto para el cual fue concebido. Se desarrolló un sistema que permite a los desarrolladores la edición de configuraciones a las escenas en los laboratorios virtuales. La cual cumple con todos los requisitos que se propusieron, permitiéndole al usuario una fácil edición de las configuraciones de los entornos virtuales y la posibilidad de salvar dicha configuración en un fichero de escena compatible con las escenas que emplean los Laboratorios Virtuales.

Anexos.

ANEXO1.



ANEXO2.


scenewriter
-xml : QXmlStreamWriter
-treeWidget : QTreeWidget*
-atributeName : Qstring
-tagName : Qstring
-validTemplate : bool
+SceneWriter(arg_treewiget : QTreeWidget)
+writeFile(device : QIODevic, type : Qstring) : bool
+recursiveWriteItem(item : QTreeWidgetItem) : void
SceneReader
-xml : QXmlStreamReader
-treeWidget : QTreeWidget*
+SceneReader(arg_treeWidget : QTreeWidget *)
+read(device : QIODevice) : bool
+errorString() : Qstring
+readScene() : void
+recursiveReadScene(item : QTreeWidgetItem *) : void
Scenelistener
-mGuiMgr : GuiManager*
-mSelMgr : MeshSelector*
-mScene : Componet_Loadescene*
-DefaultBehavior : DefaultNavigationBehavior*
-mSelectedEntity : Ogre::Entity*
-mSelectedNode : Ogre::SceneNode
-Posicionador : GUISystem::Button*
-Rotador : GUISystem::Button*
-Escalador : GUISystem::Button*
-NodeXYZ : Ogre::SceneNode*
-Mcurrenttool : EditorTool*
-mCurrentAxis : Axis*
-HiliteAxis : Axis*
-LastMousePos : QPoint*
-mMoveFactor : float
-mScaleFactor : float
-mRotateFactor : float
+SceneListener(mWindows : Ogre::RenderWindow *)
+GetEsceneLoader() : Component_LoadScene *
+ResourseUpdate(Lresourse : QList<QString> *) : void
+LoadScene(Dir : QString) : void
+CreateManipulators() : void
+ShowManipulator(tool : EditorTool) : void
+HiliteManipulator(e : Ogre::Entity *) : void
+AdjustManipulators() : void
+DestroyManipulators() : void
+UpdateManipulatorsPos(pos : Ogre::Vector3 *) : void
+MoveNodeAxis(px : float, py : float) : void
+ScaleNodeAxis(px : float, py : float) : void
+RotateNodeAxis(px : float, py : float) : void
onKeyPress(event : QKeyEvent *) : virtual void
+onKeyRelease(event : QKeyEvent *) : virtual void
+onMousePress(event : QMouseEvent *) : virtual void *
+onMouseRelease(event : QMouseEvent *) : virtual void
+onMouseMove(event : QMouseEvent *) : virtual void
+FindObject(name : Qstring) : void
+RecursiveFindObject(name : Qstring, CurrentNew : Ogre::SceneNode) : void
+frameRenderingQueued(evt : Ogre::FrameEvent &) : virtual bool
+frameStarted(evt : Ogre::FrameEvent &) : virtual bool
+frameEnded(evt : Ogre::FrameEvent &) : virtual bool
+setupContent() : virtual void
+CreateCamera() : virtual void
+CreateViewports() : virtual void
+setupTools() : virtual void
+LoadResourceElements() : virtual void
+buttonHit(button : GUISystem::Button *) : virtual void
+CreateGuiSystem() : virtual void
+UpdateLastImpactPoint(pX : float &, pY : float &) : void
+HideSelectedNode() : void
+DeleteSelectedNode() : void
+ShowAllNodes() : void
Mainwindo
-dockWidget : QDockWidget*
-sceneWidget : QWidget*
-treeWidget : QTreeWidget*
-menuBar : QMenuBar*
-readerScene : SceneReader*
-readerTempl : TemplateReader*
-writerScene : SceneWriter*
-visor : OgreWidget*
-mSceneListener : SceneListener*
+CreateDockwiget() : void
+CreateSceneWiget() : void
+Createtrewiguet() : void
+CreateMenu() : void
+Paneltools(QDockWidget *dok2)() : void
+ConnectActions() : void
+findObjet(QString nodeName,QTreeWidgetItem * raiz)() : void
+createNew() : void
+loadScene() : void
+saveScene() : void
+deleteItem() : void
+HideNode() : void
+ShowAllNodes()
+ShowMoveManipulator() : void
+ShowRotateManipulator() : void
+ShowScaleManipulator() : void
Teamplatereader
-xml : QXmlStreamReader
-treeWidget : QTreeWidget*
-directory : Qdir
+TemplateReader(arg_treeWidget : QTreeWidget *)
+read(device : QIODevice) : bool
+errorString() : Qstring
+readScene() : void
+recursiveReadScene(item : QTreeWidgetItem *) : void
+InsertItems(selectact : QAction *, arg_item : QTreeWidgetItem *) : void *
+readDirectory() : void
+readForMenu() : void
+readForInser(arg_item : QTreeWidgetItem *) : void

RECOMENDACIONES:

Con el objetivo de mejorar el sistema elaborado, se propone:

1. Agregarle a la herramienta la funcionalidad de trabajar al mismo tiempo con varias escenas de manera tal que permita mezclar varias escenas.

2. Brindar la posibilidad de cargar el recurso que falta durante el proceso de carga de escena.

3. extender la herramienta de forma tal que no se ejecute como una herramienta independiente sino acoplada a la arquitectura de los laboratorios virtuales como una ventana emergente en tiempo de construcción.

Bibliografía.

1. **Medina, Leonardo Antonio Nieblas Palau y Yasmany Cubela.** *Interfaz Visual para la configuración de Entornos Virtuales desarrollados con la Herramienta Scene Tool Kit.* la Habana : s.n., 2008.

2. WorldLingo. [En línea] http://www.multilingualarchive.com/ma/enwiki/es/3D_computer_graphics. .

3. **Barcelona, DLSI Facultad informatica de.** FIB. [En línea] www.lsi.upc.edu/~virtual/SGI/guions/ArquitecturaRV.pdf.

4. **Microsoft.** msdn. [En línea] 11 de 2007. http://msdn.microsoft.com/es-es/library/ms747437(v=vs.90).aspx.

5. **Rodilla, Miguel Martínez.** *Desarrollo de un sistema de edición de escenas basado en una librería gráfica de alto nivel: OpenSceneGraph.* Valencia : s.n., 2006.

6. **Ogre3d.** ogre3d.org. [En línea] [Citado el: 8 de 05 de 2012.] http://www.ogre3d.org/tikiwiki/OGRE+Exporters.

7. **Miguel Lozano, Carlos calderón.** *Entornos virtuales 3D clásicos e inteligentes: hacia un.* 2005.

8. **Ogre, Comunidad.** ogrees.wikispaces.com. [En línea] http://ogrees.wikispaces.com/.

9. **Moen, Jacob.** Support and community documentation for Ogre3D. [En línea] http://www.ogre3d.org/tikiwiki/SdkTrays..

10. **Fernández, Marcos.** *Informática Gráfica.* Valencia : s.n.

11. **Jackie Neider, Tom Davis, Mason Wood.** *OpenGL Programming Guide, The official guide.* s.l. : Addison-Wesley Publishing Company.

GLOSARIO DE TÉRMINOS:

A

Algoritmo: Es un conjunto pre-escrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

C

Comando: Es una instrucción u orden que el usuario proporciona a un sistema informático, desde la línea de comandos (como una *shell*) o desde una llamada de programación.

I

Interfaz: Es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Intuición: Es aquel conocimiento que es directo e inmediato, sin intervención de la deducción o del razonamiento, siendo habitualmente considerado como evidente.

M

Mundo Virtual: Es un tipo de comunidad virtual en la que simula un mundo o entorno artificial inspirado o no en la realidad, en el cual los usuarios pueden interactuar entre sí a través de personajes o avatares, y usar objetos o bienes virtuales.

Multiplataforma: Se les llama de esa forma a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

R

Render: Se referirse al proceso de generar una imagen a partir de un modelo 3D.