Universidad de las Ciencias Informáticas

Facultad 5

Editor de Personajes para el sistema de entrenamientos Indicios Aduaneros

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

Autor: Héctor Orestes Romani Goicochea

Tutor: Ing. Jaime González Campistruz

Ciudad de La Habana, junio de 2010

"Estar preparado es importante, saber esperarlo es aún más, pero aprovechar el momento adecuado es la clave de la vida."

Arthur Schnitzler

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 firmo la presente a los ____ días del mes de _____ del año _____.

Autor:

_____

Héctor Orestes Romani Goicochea

Tutor:

_____

Ing. Jaime González Campistruz

Datos de contacto

Nombre y Apellidos: Ing. Jaime González Campistruz

e-mail: jgonzalezc@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la Universidad de Ciencias Informáticas (UCI) en el 2007. Profesor de la UCI, con 3 años de experiencia en su desempeño laboral. Actualmente se desempeña como jefe del proyecto “Indicios Aduaneros”.

Agradecimientos

A mis padres por su apoyo en los momentos difíciles.

A mi hermano, que es lo máximo.

A mis tías Norma y Maicho por la confianza depositada en mí.

A mis hermanas Iosmaylly y Mara, por estar pendiente a mí en todo momento.

A mis primos Albertico y Adrián.

A mi tutor, que más que un tutor ha sido mi amigo.

A Alexey y Lannie.

A mis amigos, Alexis, Jose, Sanjony, Darluin, Yandry, Pavel, Gustavo, Adonis, Darluin, Joel, Hung, Fernando, Yosuan.

A Danyer, Alejandro, Pander, Iriobe y Reinier.

A Tatiana, Lianet, Adys, Arais, por estar ahí en el momento que necesité de su ayuda.

A las "Ladronas" estar siempre alegres.

A Daryl, Claudio, Julio Cesar, Elisa, Mónica, Pepito por los consejos.

A la gente de la "autopista".

En fin, a toda aquella persona que me ayudó en estos cinco y fuertes años.

Dedicatoria

A mi mamá Olga y mi papá Héctor, porque sin su apoyo no habría llegado hasta aquí.

A mis tías Norma y Maicho por la ayuda y el apoyo que siempre me han dado.

A Jaime.

A Lannie y Alexey.

A mis hermanas Iosmaylly y Mara.

A mi hermanito Ibrahim.

A Daryl, Albertico, Adrián y Claudio.

A mis amigos del 91.

Resumen

El desarrollo de la Realidad Virtual (RV) ha traído consigo el nacimiento de una nueva era de la informática gráfica. Con la introducción de estas técnicas en la programación basada en tres dimensiones, se ha logrado un nivel de aceptación considerable en la comunidad mundial, ya que proporcionan una aproximación al entorno real que se quiere simular. En las aplicaciones de RV existe una gran variedad de objetos, es decir, animales, cuerpos físicos, personas. Estos objetos pueden tener características físicas o psicológicas las cuales pueden ser empleadas en sistemas de Inteligencia Artificial (IA). El objetivo de este trabajo es desarrollar una aplicación que permita la asignación de propiedades a objetos, partiendo de las características con que se quiere generar dicho objeto.

En la investigación efectuada como parte del presente trabajo se abordan los conceptos y técnicas más utilizadas en la creación de editores y generadores de objetos, así como ejemplos de los editores más usados en el orbe. En este documento se muestra el proceso completo para editar y generar personajes (objetos) de manera manual y aleatoria, donde luego de ser generados los personajes, se procede a la salva de la configuración para la cual se usó el formato XML. También se definen los pasos para la carga de un fichero XML con los datos de los personajes generados anteriormente.

A continuación se expone la fase de investigación, luego el diseño y la implementación de un conjunto de clases que permiten editar y generar personajes. Como resultado de este proceso se obtuvo una aplicación que posibilita la edición y generación de personajes para el proyecto Indicios Aduaneros residente en el Centro de Desarrollo de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas de Cuba.

Palabras clave

Entornos virtuales, fichero, motores gráficos, personajes, random, realidad Virtual, XML.

Resumen

Abstract

The development of Realidad Virtual (RV) has brought with himself the birth of a new one was of graphical computer science. With the introduction of these techniques in the programming based on three dimensions, a level of considerable acceptance in the world-wide community has been obtained, since they provide an approach to the real surroundings that are wanted to simulate. In the applications of RV a great variety of objects, that is to say, animal exists, physical bodies, people. These objects can have physical or psychological characteristics with which the user is going to interact direct or indirectly.

The objective of this work is to develop an application that allows the allocation of properties to objects, starting off of the characteristics whereupon it is wanted to generate this object. In the conducted investigation as it leaves from the present work approach the concepts and techniques more used in the creation of publishers and generators of objects, as well as examples of the used publishers more in the orb. In this document is the complete process to publish and to generate characters (objects) of random way manual and, where after being generated the characters, it is come to the safe one from the configuration for which format XML was used. Also the passages for the load of file XML with the data of the generated characters are defined previously.

Next the phase of investigation is exposed, soon the design and the implementation of a set of classes that allow to publish and to generate characters. As a result of this process an application was obtained that in center makes possible the edition and generation of characters for the project Customs Indications resident of Development of Informática Industrial (CEDIN) of the University of Computer science Sciences of Cuba.

Key words

Virtual surroundings, graphical file, motors, personages, random, virtual reality, XML.

Índice

Introducción

Las nuevas tecnologías de la informática han alcanzado un gran desarrollo dentro de las que se incluye ramas como el entretenimiento y la simulación. Las cuales han hecho posible el surgimiento de nuevas tecnologías para el procesamiento de imágenes por computadora. Como consecuencia del avance que se tiene en la actualidad los sistemas de RV se han hecho más complejos, alcanzando progresivamente un realismo comparable con los procesos del mundo real. La creación de juegos y simuladores, con los cuales el usuario puede interactuar dependiendo del límite de su imaginación son algunas de las aplicaciones que tienen hoy en día la RV.

Los juegos, a diferencia de los simuladores, se crean con fines de entretenimiento, contienen una parte gráfica que engloba todo lo relacionado con el entorno donde se desarrolla el juego y poseen una parte lógica que cuenta con un conjunto de reglas y metas que de ellas depende la aceptación de los usuarios. Por otra parte, los simuladores son creados con fines de aprendizaje y son utilizados en varias materias como la aviación, la conducción, la navegación, entre otras y al igual que los juegos, constan de parte lógica y una parte gráfica.

Dentro de los juegos y los simuladores uno de los elementos clave para el cumplimiento de los propósitos antes mencionados, es la aparición de los personajes y el papel o rol que desempeñan. Pues estos ayudan a incrementar considerablemente la calidad de interactividad de los entornos virtuales y ayuda a que los usuarios se sientan como parte del sistema y que a través de la inteligencia artificial llevan el propósito de su existencia al máximo.

La Universidad de las Ciencias Informáticas de la República de Cuba no se encuentra excepta de estos avances, debido a que cuenta con un grupo de poryectos residentes en el Centro de Informática Industrial dedicados específicamente al trabajo con la RV.

Entre los proyectos existentes en el centro se encuentra Entrenadores Aduana, el cual existe para desarrollar una serie de productos que darán soporte a la preparación de los recursos humanos de la Aduana General de la República de Cuba; Sistema de Entrenamiento RX, Sistema de Entrenamiento y Sistema de Entrenamiento Indicios Aduaneros de Sondeo nombrados así respectivamente. Este último consiste en un juego simulador del proceso de entrada y salida de pasajeros al país a través del servicio

aeroportuario, donde el usuario, que asume el papel de agente de la aduana, trata de identificar los posibles sospechosos de delitos como el tráfico de drogas o armas, terrorismo, entre otros. Como el proceso que se quiere simular es dentro de una terminal de aeropuerto, debe existir un gran flujo de personajes en constante movimiento, donde cada personaje tiene características y documentos particulares. Entre los documentos del pasajero se encuentran el pasaporte, el boleto del viaje, el boucher y dentro de las características se encuentran las físicas y psicológicas. El usuario, que desempeña el rol de agente de la aduana, se basará en estas características y documentos para determinar si un pasajero es sospechoso o no de algún delito. Cada vez que se inicie una partida nueva, el flujo de pasajeros debe variar, es decir, los personajes deben tener particularidades para establecer una diferencia entre ellos, estas pueden ser tanto psicológicas como documentales, debe existir otra cantidad de pasajeros, para así aumentar el dinamismo de la aplicación.

Para obtener este gran flujo de pasajeros en la terminal se tiene que modelar pasajero por pasajero, lo cual se torna un poco difícil y sobretodo demora el tiempo de desarrollo del software. Hasta el momento, la generación de pasajeros se ha hecho manualmente, por lo que surge la necesidad de crear un editor de personajes que permita generar y editar personajes, así como listas de personajes de manera aleatoria y automática.

Dada la situación existente se propone como **problema científico** a resolver con este trabajo, ¿Cómo lograr asignarle características y propiedades a los personajes de forma automática y aleatoria?

Se enmarca el **objeto de estudio** en los sistemas de edición y generación de objetos para sistemas de realidad virtual y se tendrá como **objetivo general de la investigación** el siguiente: Lograr un sistema que permita editar y generar personajes de forma automática.

Para darle cumplimiento al objetivo general, la investigación se basa en el **campo de acción:** Editores y generadores de objetos para sistemas de realidad virtual.

Dado el objetivo planteado anteriormente se hace necesario seguir las **tareas** que a continuación se exponen:

1. Comparar los métodos de generación y edición de objetos ya existentes para tener una visión de su estado del arte.
2. Seleccionar las herramientas a utilizar para el diseño y la implementación del sistema.
3. Estudiar las herramientas para ofrecer un mejor conocimiento de su funcionamiento.
4. Realizar el Diseño del sistema a implementar.
5. Implementar una primera versión del sistema.

Idea a defender:

Con el sistema editor de objetos se podrá alcanzar una mejor manipulación de grupos de objetos, lo que optimizará el tiempo de caracterización de dichos objetos.

Para darle cumplimiento a las tareas se basará la investigación en los marcos de los métodos científicos de investigación "**teóricos y empíricos**". Dentro de los métodos teóricos se encuentra el **Análisis Histórico Lógico**, el cual permite realizar una serie de estudios basados en la evolución y las tendencias actuales de la Edición de Objetos, conociendo así la trayectoria histórica real de su desarrollo. **Análisis Sintético** es otro de los métodos que se usará para llevar a cabo la investigación, la cual se utilizará para el estudio de los conceptos y técnicas empleados dentro de la Realidad Virtual en la Edición de Objetos, analizando todos los documentos para la extracción de los elementos más importantes sobre el tema en cuestión. **Modelación** es otro de los métodos que se llevará a cabo para la pesquisa, el cual se utilizará para la elaboración del diseño del sistema mediante los modelos y diagramas necesarios. **Observación** se empleará para llevar a punta la investigación el cual permite adquirir información necesaria y puede utilizarse en cualquiera de las fases de la investigación, además ofrece un gran acercamiento a la realidad y permite ver una posible solución del problema desde varios puntos de vista.

Capítulo 1: “Fundamentación Teórica”

Introducción

En la industria de la informática gráfica existe una amplia variedad de editores y generadores de objetos, los cuales se encargan de sintetizar y caracterizar dichos cuerpos para luego ser utilizados por otros programas, ya sean juegos o simuladores. Estas herramientas desempeñan un papel fundamental en el desarrollo de aplicaciones relacionadas al tema de la RV, ya que mediante su utilización permiten editar y generar objetos de acuerdo al entorno virtual que se desee simular.

Hace algunas décadas existían serios problemas de edición y generación de objetos, debido a que las técnicas presentes en esos momentos no llenaban las espectativas de la comunidad mundial. Las grandes compañías de juegos y simuladores editaban los objetos de un determinado entorno, de forma manual, lo cual retrasaba el tiempo de construcción del software. Esto trajo consigo que las empresas líderes en el movimiento de la informática gráfica construyeran sus propios sistemas, lo cual era una ventaja, ya que posibilitaría la generación de objetos de acuerdo a las reglas y restricciones del entorno que se quisiera construir.

Actualmente los sistemas editores y generadores de objetos se utilizan para crear personajes con características físicas y psicológicas particulares que posteriormente se insertarán en aplicaciones relacionadas con el entrenamiento de diversas materias. Ejemplo de ello son los simuladores que se emplean en la industria militar, en los cuales debe existir un nivel de presición alto, ya que un error en la vida real costaría una vida humana.

Estas herramientas presentan características comunes. Brindan la posibilidad de cargar, editar, modificar un objeto o entorno, así como exportar un elemento generado a un formato especificado por el usuario que las use.

El presente capítulo brinda una visión general de los aspectos relacionados con la edición y generación de objetos, así como la conceptualización de los fenómenos asociados al dominio del problema y que son necesarios para obtener un mejor entendimiento de la propuesta de solución. Englobaremos las características de los editores estudiados para el desarrollo de la aplicación, así como la definición de los

principales conceptos referentes al tema y también nos adentraremos en el mundo de los editores y generadores de objetos.

1.1 Técnicas utilizadas en la creación de editores de objetos

1.1.1 Métodos Basados en Matemáticas Formales

Las técnicas matemáticas han sido utilizadas para crear especificaciones de probable exactitud y proporcionan la corrección de programas. Los procedimientos formales son utilizados para garantizar que los programas realicen su función libre de errores. Los propios son adecuados para crear los bloques de construcción (clases) de los Sistemas Orientados al Objeto. (11)

Son métodos basados en la matemática (por algo se llaman formales) en donde su potencial se centra en que la matemática no es ambigua y está libre de errores, por lo que desarrollar software mediante estos métodos, genera programas libres de errores(solo en teoría), el problema es que el costo de expresar un modelo de software en términos lógicos, formales y matemáticos, es mucho más caro que el costo de usar por ejemplo UML, y realizar un buen mantenimiento y desarrollo del software con un buen enfoque de calidad. El alto costo es consecuencia del tiempo necesario para desarrollar el modelo del programa. Además, esto posee una falencia, no cualquier software se puede expresar formalmente.

Los métodos formales tienen la ventaja de comprender mejor el sistema, la comunicación con el cliente mejora, ya que se dispone de una descripción clara y no ambigua de los requisitos de usuario. El sistema se describe de manera más precisa y se asegura matemáticamente que es correcto según las especificaciones. Proporciona una mayor calidad de software según las especificaciones, además de una mayor productividad.

Estos métodos tienen ventajas, y se les asocia una problemática actual que no es más que la falta de madurez en la práctica de los métodos formales es la causa de la imposibilidad de utilizarlos a nivel industrial tal y como se utilizan otros métodos de la Ingeniería del Software. Dentro de las principales causas de este problema se encuentran: El desarrollo de herramientas que apoyen la aplicación de métodos formales es complicado y los programas resultantes son incómodos para los usuarios Los investigadores por lo general no conocen la realidad industrial. Es escasa la colaboración entre la industria

y el mundo académico, que en ocasiones se muestra demasiado dogmático. Se considera que la aplicación de métodos formales encarece los productos y ralentiza su desarrollo.

1.1.1.1 Clasificación de los métodos formales

Se pueden encontrar inmensidad de métodos y técnicas formales con lo que los criterios de clasificación son bastante variados. La clasificación más común se realiza en base al modelo matemático subyacente en cada método, de esta manera, podrían clasificarse en especificaciones basadas en lógica de primer orden y teoría de conjuntos la cual permite especificar el sistema mediante un concepto formal de estados y operaciones sobre estados. Los datos y relaciones/funciones se describen en detalle y sus propiedades se expresan en lógica de primer orden. La semántica de los lenguajes está basada en la teoría de conjuntos. Los métodos de este tipo más conocidos son: Z, VDM y B. También pueden clasificarse en especificaciones algebraicas las cuales proponen una descripción de estructuras de datos estableciendo tipos y operaciones sobre esos tipos.

Para cada tipo se define un conjunto de valores y operaciones sobre dichos valores. Las operaciones de un tipo se definen a través de un conjunto de axiomas o ecuaciones que especifican las restricciones que deben satisfacer las operaciones. Métodos más conocidos: Larch, OBJ, TADs.

1.1.2 Inferencia Lógica

Una inferencia lógica es un software que hace las deducciones a partir de hechos y reglas, utilizando técnicas de Inferencia Lógica, opera con una serie de reglas del área de conocimiento. Selecciona reglas y las encadena en forma efectiva para realizar un razonamiento inferencial. Se puede utilizar encadenamiento hacia delante (razonamiento dirigido por INPUTS) o encadenamiento reverso (razonamiento dirigido por OBJETIVOS) o ambos. Esto capacita a un computador a hacer deducciones complejas sin un programa de aplicación. Es la técnica fundamental utilizada en software de Inteligencia Artificial.

Una inferencia es una evaluación que realiza la mente entre conceptos que, al interactuar, muestran sus propiedades de forma discreta, necesitando utilizar la abstracción para lograr entender las unidades que componen el problema, creando un punto axiomático o circunstancial, que nos permitirá trazar una línea

lógica de causa-efecto, entre los diferentes puntos inferidos en la resolución del problema. Una vez resuelto el problema, nace lo que conocemos como postulado, o una transformada de la original, que al estar enmarcado en un contexto referencial distinto, se obtiene un significado equivalente.

La inferencia es la forma en la que obtenemos conclusiones en base a datos y declaraciones establecidas. Un argumento, por ejemplo es una inferencia, donde las premisas son los datos o expresiones conocidas y de ellas se desprende una conclusión. Una inferencia puede ser: Inductiva, deductiva, transductiva, abductiva.

Inductiva

En general una inferencia inductiva es la que se desprende de una o varias observaciones y en general no podemos estar seguros de que será verdadero lo que concluimos. Resumiendo, la inferencia inductiva es la ley general que se obtiene de la observación de uno o más casos y no se puede asegurar que la conclusión sea verdadera en general. (7)

Deductiva

Cuando se conoce una ley general y se aplica a un caso particular, por ejemplo se sabe que siempre que llueve hay nubes, ultimamos que el día de hoy que está lloviendo hay nubes. También se conoce como inferencia deductiva cuando tenemos un caso que analiza todos los posibles resultados y de acuerdo con las premisas sólo hay una posible situación, en este caso decimos que la situación única es la conclusión. Es este caso estamos seguros de que si las premisas son verdaderas entonces la conclusión también lo es. (7)

En este caso se encuentran **MPP: Modus Ponendo Ponens** y **MTT: Modus Tollendo Tollens** que de acuerdo a la tabla de verdad de la condicional son dos formas de establecer una inferencia válida. La inferencia deductiva es la única aceptada como válida en matemáticas y computación para hacer comprobaciones y sacar conclusiones. El tema se discute en forma detallada más delante en Inferencia Deductiva con una condicional.

1.1.3 Bibliotecas de Clases

Una biblioteca de clases contiene implementaciones reusables de tipo objeto. Su objetivo consiste en conseguir el máximo grado de reusabilidad en el desarrollo de software. El software con librería de clases debe ayudar al programador a encontrar, adaptar y reutilizar las clases que necesita para la aplicación que él desarrolle.

Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables. Ejecutables y bibliotecas hacen referencias (llamadas enlaces) entre sí a través de un proceso conocido como enlace, que por lo general es realizado por un software denominado enlazador.

La mayoría de los sistemas operativos modernos proporcionan bibliotecas que implementan la mayor parte de los servicios del sistema. De esta manera, estos servicios se convierten en una "*materia prima*" que cualquier aplicación moderna espera que el sistema operativo ofrezca. Como tal, la mayor parte del código utilizado por las aplicaciones modernas se ofrece en estas bibliotecas.

Bibliotecas Estáticas

Históricamente, las bibliotecas sólo podían ser estáticas. Una biblioteca estática, también conocido como *archivo*, consiste en un conjunto de rutinas que se copian en una aplicación por el compilador o el enlazador, produciendo archivos con código objeto y un fichero ejecutable independiente. Este proceso, y el archivo ejecutable, se conocen como una construcción estática de la aplicación objetivo. La dirección real, las referencias para saltos y otras llamadas a rutinas se almacenan en una dirección relativa o simbólica, que no puede resolverse hasta que todo el código y las bibliotecas sean establecidos a direcciones estáticas finales.

El enlazador resuelve todas las direcciones no resueltas convirtiéndolas en direcciones fijas o que pueden volver a localizarse (desde una base común) cargando todo el código y las bibliotecas en posiciones de memoria en tiempo de ejecución. Este proceso de enlazado puede durar incluso más tiempo que el

proceso de compilación, y debe ser realizado cada vez que alguno de los módulos es recompilado. La mayoría de los lenguajes compilados tienen biblioteca estándar (por ejemplo, la biblioteca estándar de C), pero los programadores también pueden crear sus propias bibliotecas personalizadas. Los compiladores comerciales proporcionan tanto las bibliotecas estándar como las personalizadas.

Un enlazador puede trabajar sobre tipos específicos de ficheros de objeto, y por lo tanto requiere tipos específicos (compatibles) de bibliotecas. Los ficheros objeto recopilados en una biblioteca pueden distribuirse y utilizarse fácilmente. Un cliente, ya sea un programa o una biblioteca de subrutinas, accede a una biblioteca objeto haciendo referencia a su nombre. El proceso de enlazado resuelve las referencias buscando en las bibliotecas del orden dado. Por lo general, no se considera un error si un nombre puede encontrarse varias veces en un determinado conjunto de las bibliotecas.

Bibliotecas Dinámicas

Enlace dinámico significa que las subrutinas de una biblioteca son cargadas en un programa en tiempo de ejecución, en lugar de ser enlazadas en tiempo de compilación, y se mantienen como archivos independientes separados del fichero ejecutable del programa principal. El enlazador realiza una mínima cantidad de trabajo en tiempo de compilación, registra que rutinas de la biblioteca necesita el programa y el índice de nombres o números de las rutinas en la biblioteca. La mayor parte de la labor de enlazado se realiza en el momento en que la aplicación se carga (tiempo de carga o *loadtime*) o durante la ejecución (tiempo de ejecución o *runtime*). El necesario código enlazado, llamado por el cargador, es de hecho parte del sistema operativo subyacente. En el momento adecuado el cargador localiza las bibliotecas en el disco y añade los datos relevantes de éstas en el espacio de memoria del proceso.

Algunos sistemas operativos sólo pueden enlazar una biblioteca en tiempo de carga, antes de que el proceso comience su ejecución, otros son capaces de esperar hasta después de que el proceso haya empezado a ejecutarse y enlazar la biblioteca sólo cuando efectivamente se hace referencia a ella (es decir, en tiempo de ejecución). Esto último se denomina "retraso de carga". En cualquier caso, esa biblioteca es una biblioteca enlazada dinámicamente.

1.2 Motores gráficos

A continuación se hará una revisión de los motores gráficos más empleados y que están presentes en el mercado, algunas de sus características y en qué basan su dibujado.

1.2.1 Ogre (Object-Oriented Graphics Rendering Engine)

Está diseñado desde un comienzo con la idea de la orientación a objetos, por lo que su interfaz es clara, intuitiva y fácil de emplear. Es un proyecto open source bajo licencia LGPL por lo tanto es gratis su uso. No posee nada que envidiarle a los engines más nuevos disponibles en el mercado. Entre sus características se encuentra el soporte para VS 3.0 y exportadores para las herramientas de modelación tridimensional más utilizadas en las empresas desarrolladoras de videojuegos. Esto significa que con Ogre se logra crear juegos o cualquier prototipo de aplicación que trabaje con gráficos tridimensionales que no tengan nada que envidiarle a lo más moderno de la industria de los videojuegos.

Este motor no trae soporte nativo para sonido ni física, lo que no es un problema, ya que para aplicar diferentes características, como sonido, networking, colisión, leyes físicas, entre otras, necesitarás integrar a OGRE con otras librerías. Está orientado al crecimiento de la escena, y diseñado con el objetivo principal de permitir a los programadores producir aplicaciones utilizando gráfico en 3D acelerados por hardware**.** Brinda como beneficio fundamental su diseño coherente y la documentación especificada y consistente que viene con el motor**.**

1.2.2 Quake 2

Es el motor gráfico del videojuego Quake 2, desarrollado por John Carmak de Id Software. Se basa en el renderizado por árboles BSP y radiosidad. Soporta plataformas Windows, GNU/Linux y Macintosh. Su Licencia oscila entorno a los USD 250 000 que da acceso al código fuente.

1.2.3 Torque (V12)

Motor gráfico utilizado en el juego Tribes 2 de Dinamix basado en renderizado en portales. Desarrollado para las plataformas, tanto el cliente como el servidor, Windows, Mac OS 9/X y en preparación GNU/Linux. Es necesaria licencia (USD 100) que da acceso al código fuente. (12)

1.2.4 Maxwell Render

Este motor de render fue uno de los primeros basados en luz física su lanzamiento fue en diciembre de 2004, la ventana que posee es que los resultados son simplemente reales ya que es un motor de Render Físico, la desventaja es el tiempo, liberado de la computadora que se tenga, tarda bastante para conseguir una sola imagen.

Figura 1 Ejemplo de luces con Maxwell.

Es un motor de render que trabaja con luz físicamente realista, sus algoritmos y ecuaciones imitan todas las conductas de la luz.

1.3 Bibliotecas para desarrollar interfaces gráficas

1.3.1 Cegui

El diseño de CEGUI es agudamente configurable. El sistema en sí mismo no carga directamente ficheros. CEGUI interconecta con éstos con código determinado por el usuario, no obstante el paquete del código de fuente de CEGUI viene con un número de módulos para usar ciertos componentes y bibliotecas. Esta libertad permite que el usuario utilice CEGUI en cualquier clase de sistema de gerencia de recurso o de ambiente de funcionamiento. (13)

La representación es lograda por un módulo back-end. El código de fuente de CEGUI viene con módulos para Direct3D, OpenGL, Motor del OGRE 3D, y Motor de Irrlicht. Otros módulos se pueden escribir para los motores de encargo. La carga del archivo y la manipulación de recursos se operan con APIs en código del usuario. El usuario puede definir en CEGUI cómo se cargan los archivos, cómo se asigna la memoria,

y las tareas básicas otras. Esto permite que la biblioteca sea utilizada virtualmente en cualquier ambiente de la codificación.

El sistema textual de la disposición de CEGUI es **Unicode**, aunque su motor de la disposición no puede manejar los sistemas de la escritura que divergen grandemente de alfabeto latino de izquierda a derecha.Viene con un sistema razonable de widgets, comparable a los de la caja de herramientas media del widget.

1.3.2 Qt

Qt es un framework de desarrollo de aplicaciones multiplataforma. Se presenta acompañado de un conjunto de herramientas para brindar facilidad de uso. Muestra sus porpios tipos y estructuras de datos. El funcionamiento de una aplicación efectuada con Qt está considerado en función de funciones y señales (*Signals y Slots*). Está separado por módulos, los cuales son:

QtCore: contiene el núcleo no gráfico de Qt.

QtGui: la colección básica de componentes gráficos.

QtNetwork: clases para escribir clientes y servidores TCP/IP.

QtOpenGL: para facilitar el uso de OpenGL.

QtScript: Expone las aplicaciones a scripting con un lenguaje ECMAscript.

QtScriptTools: un depurador de QtScript.

QtSQL: integración de bases de datos.

QtWebKit: el popular motor web, con Qt.

QtXml: soporte básico de Xml.

QtXmlPatterns: un motor de XQuery 1.0 y XPath 2.0 y parcialmente Xslt.

Phonon: El framework multimedia.

Qt3Support: Compatibilidad con Qt3.

Otros: QtSVG**,** QtDessigner, QtUiTools, QtHelp, QtAssistant, QtTest, QtDBus (solo Unix), y a partir de Qt4.6 QtOpenVG y QtMultimedia.

1.3.3 GTK+

GTK+ o *The Gimp Toolkit* es un conjunto de bibliotecas multiplataforma para el desarrollo de interfaces gráficas. El API GTK+ está escrito en C, pero tiene enlaces a muchos otros lenguajes como C++, Python, C#, entre otros. GTK+ está licenciado bajo la licencia GNU LGPL 2.1, la cual permite el desarrollo de software propietario o libre con GTK+ sin ningún tipo de licencia de honorarios o regalías. Posee un selector de archivos que ofrece una copia de seguridad a ficheros la cual se oculta de manera predeterminada, recuerda el estado de clasificación de la lista de archivos, muestra los tamaños de los archivos de forma predeterminada. Admite la autenticación contra servidores. El selector de archivos puede mostrar diferentes iconos para directorios de usuario. Proporciona API asíncrona y cancelables para resolver nombres de host, la búsqueda inversa de direcciones IP. Los entornos de escritorio no son necesarios para ejecutar los programas GTK+. Si las bibliotecas que requiere el programa están instaladas, un programa GTK+ puede ser ejecutado por encima de otros entornos. (14)

1.4 Formatos utilizados en los videos juegos

Para la composición de un videojuego es preciso el uso de varios formatos aplicables a los objetos que prevalecen en el entorno de la aplicación. A continuación se exponen diversos de estos formatos.

1.4.1 Mesh

Es un archivo informático que contiene información de cualquier tipo, codificada en forma binaria para el propósito de almacenamiento y procesamiento en ordenadores. Ejemplo de ellos son los archivos informáticos que almacenan texto formateado o fotografías, así como los archivos ejecutables que contienen programas.

Muchos formatos binarios contienen partes que pueden ser interpretados como texto. Un archivo binario que solo contiene información de tipo textual sin información sobre el formato del mismo se dice que es un archivo de texto plano. Usualmente se contraponen los términos 'archivo binario' y 'archivo de texto' de forma que los primeros no contienen solamente texto.

Habitualmente se piensa en los archivos binarios como una secuencia de bytes que es lo que implica que dígitos binarios (bits) se agrupen de ocho en ocho comúnmente. Los archivos binarios contienen bytes suelen ser interpretados como alguna cosa que no sean caracteres de texto. Un ejemplo típico son los programas de ordenador compilados; de hecho, las aplicaciones o programas compilados son conocidos como **binarios**, especialmente entre los programadores. Pero un archivo binario puede almacenar imágenes, sonidos, versiones comprimidas de otros archivos. En pocas palabras, cualquier tipo de información.

Algunos archivos binarios tienen una cabecera, esta cabecera es un bloque de metadatos que un programa informático usará para interpretar correctamente la información contenida. Por ejemplo, un archivo GIF puede consistir en múltiples imágenes y la cabecera se usa para identificar y describir cada bloque de datos de cada imagen. Si el archivo binario no tiene cabecera se dice que es un **archivo binario plano**.

1.4.2 Skeleton

Skeletal Animation es una de las técnicas utilizadas principalmente para el movimiento en seres vertebrados. El personaje se divide en 2 partes: la piel, que es la representación superficial de personaje y el esqueleto, el cual es el conjunto de "huesos", los cuales son el punto de referencia en el proceso de animación. El formato skeleton está basado en esta técnica y es utilizado por muchos usuarios de la comunidad mundial.

1.4.3 Obj

OBJ es un formato de archivo con características geométricas, el cual ha tenido un alto nivel de aceptación. Es un formato de datos simple que representa la geometría 3D, que permite saber la posición de cada vértice, la posición de cada coordenada de textura de vértices, normales, y las caras que hacen

de cada polígono definido como una lista de vértices, y vértices textura. Los vértices se almacenan en un orden a la izquierda de forma predeterminada, por lo que la declaración explícita de las normales innecesarios.

1.5 Ficheros de almacenamiento de datos

Para seleccionar el fichero para almacenar información de los objetos se realizó un estudio sobre algunos formatos de fichero para almacenar información, entre ellos se encuentran:

Por ejemplo en LaTeX existen “marcas” que permiten estructurar un documento, ejemplo de ello es identificando el nombre del título del documento, sin embargo no existe forma de obligar a los autores de documentos a que usen estas marcas y algunos de ellos pueden introducir el título de forma que aparezca visualmente igual. Esto conlleva a problemas cuando queremos extraer de forma automática el título de varios documentos.

Otro ejemplo es txt, el cual es formato muy utilizado en el mundo, pero tiene la desventaja de que solo se puede hacer lectura y escritura de los datos contenidos en su interior. Hacer una búsqueda de un elemento costaría un tiempo considerable.

Debido a las deficiencias que tienen otros formatos, se selecciona el formato de fichero XML, ya que a diferencia de otros sistemas usados para crear documentos tiene la ventaja de poder ser más exigente con la organización del documento, lo cual resulta más estructurado. Además, permite hacer una búsqueda de un determinado elemento haciendo referencia al nombre de una marca. Ofrece portabilidad y utilización de la información a través de las distintas plataformas (permite independizar aplicaciones de datos). Es un formato legible por personas y computadores. 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 Conceptos

Para entender todo lo relacionado con la edición de objetos es necesario conocer algunos conceptos que a continuación se exponen.

1.6.1 Randomizar

Algunas de las definicione que se le atribuyen a este concepto son: azar, algo aleatorio. En informática, se puede aplicar a muchos campos, como la programación, donde se suelen utilizar con cierta frecuencia, instrucciones u órdenes para la obtención de valores aleatorios. Realmente, aún no es posible encontrar valores totalmente aleatorios, así que se suele utilizar también el término pseudoaleatorio, para indicar que un valor no es al azar del todo. Se puede aplicar también RANDOM a distintas formas de acceder a la información, sea en memoria, o en algún medio de almacenamiento como los discos. En este caso se consigue hablar de acceso aleatorio (RANDOM) o acceso secuencial (uno detrás de otro, como las casettes de música).

1.6.2 Lógica Difusa

La lógica disfusa es una materia aplicable a temas relacionados con la IA. En términos lógicos una proposición en la lógica clásica sólo admite dos valores posibles: verdadero o falso. No obstante, en la vida real las observaciones no son tan claras como para poder afirmar que una proposición es verdadera o falsa, sino que puede haber diferentes puntos de vista que sean relativos al observador. La lógica difusa es una alternativa que permite cuantificar esta incertidumbre, da la opción de asignar cierta probabilidad a cada uno de los posibles valores y, por lo tanto, añadir la relatividad del observador. La idea original que se halla detrás de la lógica difusa es la de imitar el funcionamiento del razonamiento humano, el cual normalmente no trabaja con valores exactos sino con valores relativos. Un valor en la lógica difusa puede encontrarse entre verdadero y falso, pudiendo tener valores de mucho, un poco. Esta conversión de un valor cuantitativo a un valor cualitativo se realiza mediante lo que se conoce por un subconjunto difuso (*fuzzy set*), y representa la probabilidad de que alguien asigne un cierto valor cualitativo a uno cuantitativo.

1.6.3 Generadores de Objetos

Un generador de objetos es una función de plantilla cuyo único propósito es construir un nuevo objeto de sus argumentos. Piense en ello como una especie de constructor genérico. Un generador de objetos puede ser más útil que un constructor de fricción cuando el tipo exacto que se genera es difícil o imposible de expresar y el resultado del generador se puede pasar directamente a una función en lugar de almacenarse en una variable.

1.7 Tipos de generadores de objetos

1.7.1 GAMA-Generador Automático de Modelos Animados.

GAMA (Generador Automático de Modelos Animados) es una herramienta que facilita la rápida generación de modelos 3D animados, preparados para un uso inmediato en motores gráficos de última generación. Los modelos, que serán de complexión humana y bípedos, contarán con las animaciones y texturas necesarias que permitan de forma suficiente la diferenciación entre varios personajes generados con esta misma aplicación. Mediante una interfaz clara e intuitiva, el usuario tiene la posibilidad de configurar un personaje a su medida en pocos minutos, y prepararlo para su posterior exportación a un motor gráfico. De esta forma, GAMA se bautiza como una herramienta considerablemente útil a la hora de generar proxies de videojuegos y aplicaciones de entorno 3D, dada la posibilidad de generar en poco tiempo múltiples y variados personajes que cubren cada uno de los recursos gráficos de los que es dependiente una determinada aplicación informática de carácter audiovisual.(9)

La idea principal que envuelve a esta aplicación es que mediante un interfaz clara e intuitiva, un usuario sin conocimiento del diseño gráfico y en particular, el diseño gráfico 3D orientado a videojuegos, tenga la posibilidad de configurar un personaje a su medida en pocos minutos listo para ser exportado a un motor gráfico. De esta forma, GAMA se convierte en una herramienta extremadamente útil a la hora de generar proxies de videojuegos, dada la posibilidad de generar en poco tiempo múltiples y variados personajes que cubran cada uno de los recursos gráficos de los que es dependiente una determinada aplicación informática de carácter audiovisual.(9)

GAMA es una aplicación desarrollada en forma de plug-in, que corre bajo el software propietario Maya 6.0, de Autodesk Alias.

1.7.2 Generador de Caracteres Morpho 3D.

Morpho 3D es un generador para producciones gráficas en tiempo real de escenas 3D y con elementos de todo tipo: 2D, 3D, datos, imágenes, texturas, videos, clips, audios, para operatividad manual o automática. Morpho 3D está basado en la arquitectura de alto nivel que asegura al usuario una libertad creativa superior y una calidad de salida incomparable. Morpho 3D está diseñado para un fácil manejo sin

comprometer sus funcionalidades. El usuario encontrará funciones implementadas como soporte de modelos de texto 3D, escalado de ventana de video, tickers basados en RSS y subtítulos. (10)

Morpho 3D permite al usuario aplicar los clips en formatos como AVI, MPEG, QuickTime ™, Windows Media para cualquier tipo de 2D o 3D del objeto, que no sólo ofrece la libertad creativa necesaria, sino que también hace que sea muy fácil trabajar con una variedad de contenido.

Entre las principales características, destaca interfaz de usuario personalizable. Una potente interfaz que combina de forma visual y guiada, todas las herramientas de diseño, gestión y producción de gráficos 2D y 3D; tablas y plantillas de datos, textos Unicode creados de forma automática en conexión con múltiples bases de datos de diferentes tipos y en tiempo real. Ilimitadas creaciones de páginas inteligentes. Timers, tickers, crop tickers, rolls, crawls. Importaciones con los principales estándares gráficos y broadcast. Archivos PhotoshopTM con capas, imágenes, objetos 3D Collada, PowerPoint, imágenes desde el propio navegador de Internet, Avi, Mpeg2, QuickTimeTM, Windows Media, Audio embebido, SD y HD en todos los formatos. (10)

Control de iluminación de escenas, así como de posicionamiento, rotación y escalado, y basados en programas como PowerPoint TM, y unidos a barras de contenidos dinámicas donde el usuario puede crear páginas atractivas de forma extremadamente rápida y sencilla; Control de Playout en manual o en automático.

Morpho 3D está disponible en una variedad de idiomas como Inglés, francés, portugués, español, ruso, alemán, japonés y chino. Morpho 3D se basa en un 100% **Unicode** compatible con la arquitectura y por lo tanto le permite trabajar en casi todos los idiomas escritos y algunos dialectos muertos.

El **Estándar Unicode** es un estándar de codificación de caracteres esbozado para facilitar el procedimiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas además de pasajes clásicos de lenguas muertas. El vocablo Unicode procede de los tres objetivos perseguidos: universalidad, uniformidad y unicidad.

Unicode detalla un nombre e identificador numérico único para cada carácter o símbolo, el *code point* o *punto de código*, también de otras pesquisas necesarias para su usanza correcta: direccionalidad,

capitalización y otros atributos. Unicode trata las escrituras alfabéticas, ideográficas y símbolos de forma equivalente, lo que figura que se logren combinar en un mismo texto sin la introducción de marcas o caracteres de control.

1.7.3 Generador de terrenos 3D GeoControl2

El revolucionario concepto de GeoControl2 abre completamente nuevas dimensiones de la creación de paisajes virtuales. La combinación única de las más artísticas herramientas y algoritmos semi-científicos permite crear paisajes realistas e impresionantes, sobre la base de diseños conceptuales o fantásticos. El componente interno de GeoControl2 "isolíneas", brinda por primera vez la oportunidad creativa para el control del perfil del paisaje. Las intervenciones no son destructivas, y consiente trabajar en capas siempre y también, posteriormente, reajustar, apagar o cambiar de isolíneas o una capa. (2)

Los filtros de "forma" dan al paisaje una estructura típica, como colinas redondeadas o montañas rígidas. Los semi-científicos algoritmos de erosión convierten estas líneas en realistas y fantásticos paisajes de manera creíble. Además, las corrientes, ríos o incluso redes de río se pueden añadir al terreno. El algoritmo es único, rápido y satisface incluso requerimientos científicos. (2)

Pero los paisajes no sólo son definidos por la naturaleza. El factor humano también es un elemento muy importante. La herramienta "vector", "simula" esta parte de un paisaje. Puede usarlo para integrar carreteras, zonas urbanas o incluso ríos controlados.

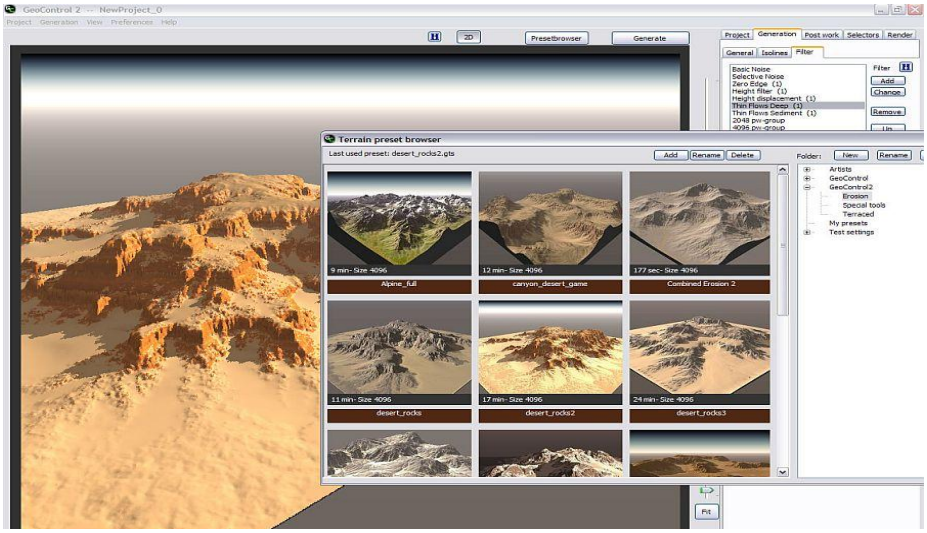

Figura 2 Editor de terrenos GeoControl2.

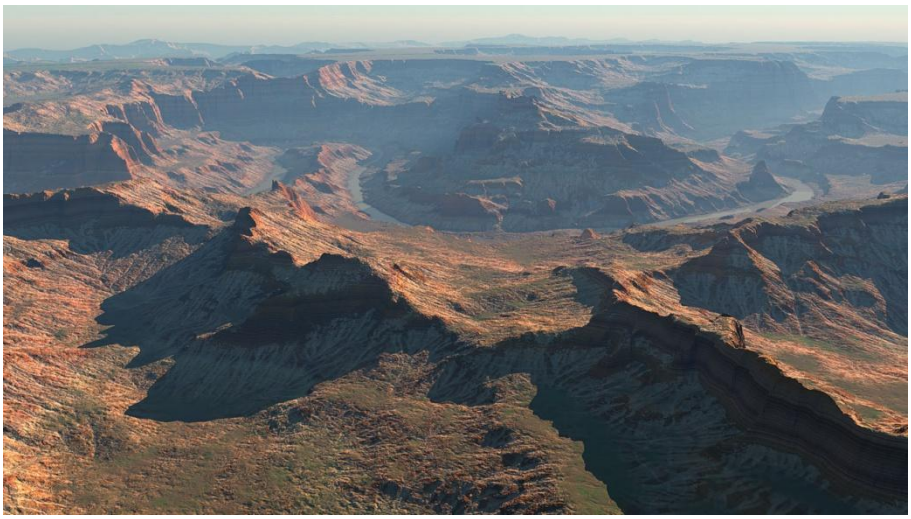

Figura 3 Paisaje generado por GeoControl2.

Figura 4 Paisaje generado por GeoControl2.

Figura 5 Paisaje generado por GeoControl2.

1.8 Editores de Objetos

Un editor de objetos es una herramienta que permite editar las propiedades de un objeto seleccionado. Propiedades comunes incluyen anchura mínima y la altura, color, texto, entre otras.

1.8.1 Tipos de editores de objetos

1.8.1.1 Misit Model 3D

Modelador OpenGL 3D que usa triángulos a modo de estructura básica. Misfit Model 3D está pensado para producir eficazmente y con tiempos de ejecución relativamente cortos, por lo que su interfaz reúne aptitudes de funcionalidad y claridad.

Misfit Model 3D ofrece todas las opciones y características básicas de un buen modelador 3D de este calibre, es decir, para poder ejercer manipulaciones precisas como definir, ubicar, agrupar, aplicar texturas, e incluso operaciones relativas a las animaciones con esqueletos y aplicación de texturas sencillas, a lo que se añade la opción de deshacer multinivel o las capacidades de scripting y procesamiento por lotes mediante la línea de comando, y un sistema de plug-ins. (5)

Precisamente, la presencia de una arquitectura orientada a plug-ins permite añadir filtros de imágenes y nuevas clases de modelos con los que se podrá trabajar, así como la creación de nuevas herramientas y objetos donde se admita ampliar las capacidades de Misfit Model 3D.

Hay muchas formas de crear modelos tridimensionales. Una de las más difundidas se basa en el uso de polígonos con caras triangulares.

Misfit Model 3D es un programa de modelado basado en OpenGL con el que podemos construir modelos a partir de formas básicas como esferas, cubos o toroides. Estos elementos primitivos se modifican mediante maniobras de extrusión, subdivisión, aplanamiento, entre otras. Las texturas se aplican a las secciones que se elijan de un objeto, y pueden generarse animaciones. Sin embargo, hay que tener presente que Misfit Model 3D no es un programa de renderizado, con lo que la generación de imágenes no es su propósito principal. (5)

Se entiende por **modelado** la creación de una estructura o imagen de un objeto real o a partir de un modelo ya existente. El modelado se refiere generalmente a la creación manual de una imagen tridimensional (modelo) del objeto real, por ejemplo en arcilla, madera u otros materiales.

1.8.1.2 Editor de modelos 3D mjbWorld

El mjbWorld es uno de los editores estándar basado en 3D, lo cual gráficamente puede editar y permite a los usuarios ver e interactuarle otros formatos al 3D. Está también dirigido a proveer una plataforma que cede a usarse para experimentar con física, animación, y demás. Las versiones para Java, C++ y C# es incluida**.**

Este editor, permite a usuarios que no son expertos en el diseño crear, revisar modelos 3D, modificar modelos existentes, exhibir y hacer reconocimientos de mundos 3D. Tiene la característica de crear un modelo propio de un hombre o una mujer, los cuales van a ser de apariencia promedio a fin de que puedan ser confeccionados a la medida para juegos diferentes, avatares y otras aplicaciones como simuladores. Además de poder generar un modelo, el software permite dar soporte a texturas, simplificación de mallas. (1)

A la hora de crear un modelo del cuerpo humano, se le puede añadir características físicas como delgadez, obesidad, músculos. También se pueden asignar características psicológicas como tristeza, felicidad, enojo. Tiene además modelos y texturas predefinidos para modelar personajes de combate, texturas para trajes, para ropas informales. Simula el comportamiento de cuerpos sólidos.

Ventajas:

Es una aplicación a distinción de un **applet**[1]. Es instalado y corrido en el sistema de usuarios, no a través de un navegador. Es optimizado para editar a distinción de un espectador. Aunque puede ser usado como

[1] Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. Debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por los mismos.

un espectador **VRML** [2] aplicaciones más simples pueden preferir una solución más ligera. No tiene límites para enumerar formas soportadas, ni para la complejidad de las formas (número de triángulos del modelo), ni para la resolución.

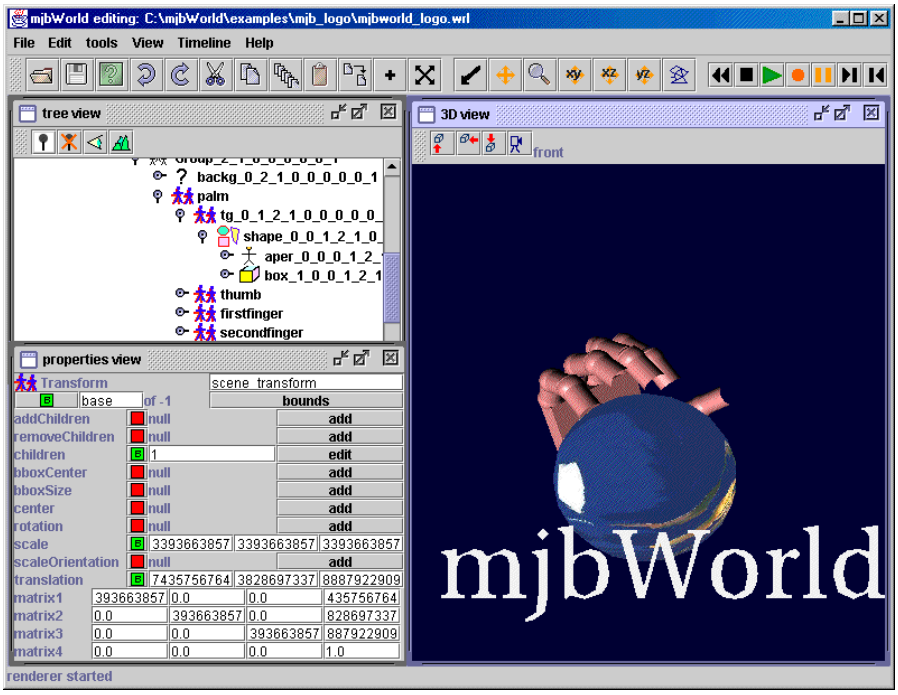

Figura 6 Editor de objetos mjbWorld.

1.8.1.3 Editor de objetos del motor de juegos Unreal

Unreal Engine es un motor de juego de PC y consolas creados por la compañía Epic Games. Implementado inicialmente en el shooter en primera persona *Unreal* en 1998, siendo la base de muchos juegos desde entonces. También se ha utilizado en otros géneros como el rol y juegos de perspectiva en tercera persona. Está escrito en C++, siendo compatible con varias plataformas como PC (Microsoft Windows, GNU/Linux), Apple Macintosh (Mac OS, Mac OS X) y la mayoría de consolas (Dreamcast,

[2] *Virtual Reality Modeling Language*. "Lenguaje para Modelado de Realidad Virtual") .Formato de archivo normalizado que tiene como objetivo la representación de escenas u objetos interactivos tridimensionales; diseñado particularmente para su empleo en la web.

Gamecube, Wii, Xbox, Xbox 360, PlayStation 2, PlayStation 3). Unreal Engine también ofrece varias herramientas adicionales de gran ayuda para diseñadores y artistas. (15)

Dentro de estas herramientas adicionales, Unreal presenta un editor de entornos el cual facilita la creación de juegos, el cual hace más fácil e intuitiva la interacción con el usuario. Esta herramienta posee una interfaz amigable y ofrece un amplio lote de funcionalidades todas desarrolladas en puro C++. Contiene además objetos, entornos, sonidos predefinidos, los cuales contienen propiedades únicas y un script editable donde se pueden actualizar dichas características, es decir, estos objetos se pueden modificar de la manera que el usuario desee. También posee un flujo de eventos a los cuales se puede someter cierto entorno u objeto respectivamente. Brinda la posibilidad de crear objetos propios del usuario, así dar soporte a texturas y animaciones a dichos objetos.

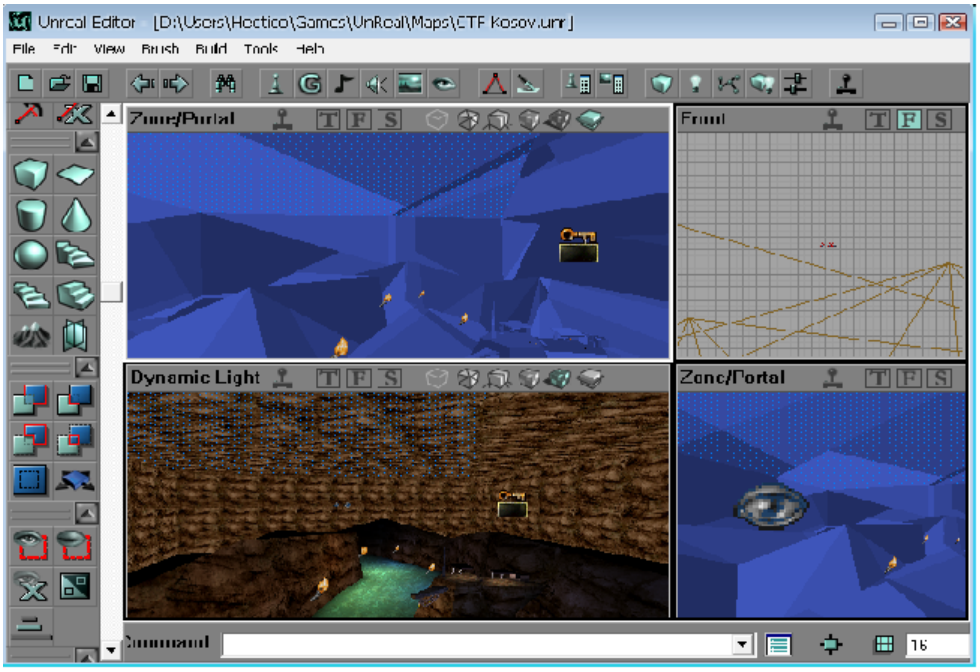

Figura 7 Editor de objetos del motor de juegos Unreal.

A pesar de la existencia de estos editores y del alcance que tienen producto de sus características, ninguno brinda solución al problema científico planteado debido a que muchas de estas herramientas pertenecen a empresas propietarias sumado a que son creadas con un objetivo específico para satisfacer una necesidad única. Además no se ajustan a los requerimientos del sistema de entrenamientos “Indicios Aduaneros”.

Capítulo 2: “Propuesta de solución”

Introducción

En el presente capítulo se presenta una visión del sistema, para ello se describe la propuesta de solución, representando dicha solución en un modelo de clases del dominio. A la vez se establecen los requisitos funcionales (capacidades o funciones que el sistema debe cumplir) y los no funcionales (propiedades o cualidades que el producto debe tener).

También se describen las capacidades de la aplicación de edición de personajes, descritas en forma de Casos de Uso del Sistema así como las relaciones entre estos. Se incluyen también en este capítulo las reglas del negocio que predominan en este proceso y el modelo del dominio.

2.1 Propuesta de solución

Para satisfacer las necesidades del problema planteado al inicio de esta investigación se propone como solución desarrollar una herramienta que permita editar y generar listas de personajes con mayor eficiencia y rapidez involucrando mayor seguridad en las decisiones. Esta aplicación debe permitir generar una lista de personajes automáticamente, es decir, generación aleatoria de datos. También brindará la posibilidad de generar dicha lista de forma manual. Debe ofrecer la posibilidad de salvar una lista de personajes, donde se guardará la información de la lista completa de personajes en un fichero. Como mismo existirá la opción de guardado de datos, el procedimiento debe permitir cargar una lista de personajes desde un archivo. Para obtener una mayor visualización del personaje seleccionado, el sistema brindará la posibilidad de rotar dicho personaje. Por último, el editor contará con la opción de salir, donde se cerrará automáticamente de la aplicación.

2.2 Biblioteca para generar y cargar el fichero XML

Para la generación del fichero donde se almacenará la configuración de los personajes se utilizará la biblioteca TinyXML, la cual provee funciones estándar como son: carga de un fichero partiendo de su nombre, generación de un documento con formato XML, verificación de la existencia de un archivo con dicho formato, entre otras.

Haciendo uso de este grupo de subrutinas se desarrollará un algoritmo que posibilitará la generación de un fichero con la configuración de todos los personajes generados por el sistema, donde la creación de las etiquetas contenidas en el archivo se hará partiendo de las propiedades con las que se generaron los personajes. El fichero XML tendrá la siguiente estructura:

```
<?xml version="1.0”?>
<Vuelo>
   <Nombre>Air France-2340 </Nombre>
   <A_origen> Italia </A_origen>
   <A_destino> Cuba </A_destino>
   <Fecha_llegada> 6-4-2010 </Fecha_llegada>
   <Hora_llegada> 10:30pm </Hora_llegada>
   <Tipo_viajero> Turístico </Tipo_viajero>
```

```
<Procedencia> Francia </Procedencia>
<Cantidad_pasajeros> 200 </Cantidad_pasajeros>
<Lista_Pasajeros>
   <Pasajero>
      <Mesh_Information>
         <Nombre_Mesh>Character_M01</Nombre_Mesh>
      </Mesh_Information>
      <General>
         <VR>25634634</VR>
         <Profesión>cvjxcvh</Profesión>
         <Motivo_Viaje>xcghxcvh</Motivo_Viaje>
      </General>
      <Documentos>
         <Pasaporte>
            <Nombre>gh</Nombre>
            <Nacionalidad>ghdf</Nacionalidad>
            <N_PNR>gdfgh</N_PNR>
            <Sexo>Masculino</Sexo>
            <Edad>22</Edad>
            <Origen>dgsdg</Origen>
            <Destino>dfhdf</Destino>
            <Desp>gfj</Desp>
            <TD>fgj</TD>
            <Num_Doc>67896</Num_Doc>
            <País>fgj</País>
            <Categoría>fgjgfj</Categoría>
            <Riesgo>fgjgfj</Riesgo>
            <Fecha_Emisión>3/1/2000</Fecha_Emisión>
            <Fecha_Vencimiento>4/1/2000</Fecha_Vencimiento>
            <Fecha_Nacionalización>4/1/2000</Fecha_Nacionalización>
```

```
        <Observaciones>fgj</Observaciones>
        <Visa>fgj</Visa>
    </Pasaporte>
    <Boleto_Viaje>
        <Fecha_Emisión>1/1/2000</Fecha_Emisión>
        <Lugar_Emisión>hgkgh</Lugar_Emisión>
        <Estancia>4</Estancia>
        <Itinerario>jkghk</Itinerario>
        <Forma_Pago>Cheque</Forma_Pago>
    </Boleto_Viaje>
    <Boucher>
        <Hospedaje>gjkgh</Hospedaje>
        <Personas_Acompañantes>4</Personas_Acompañantes>
        <Tiempo>6</Tiempo>
        <Observaciones>fgkfkg</Observaciones>
    </Boucher>
</Documentos>
<Física>
    <Vestimenta>fdgjdfg</Vestimenta>
    <Estatura>0.1</Estatura>
    <Masa>0.08</Masa>
    <Velocidad_Maxima>0.12</Velocidad_Maxima>
</Física>
<Psicológicas>
    <Nivel_Nerviosismo>2</Nivel_Nerviosismo>
    <Nivel_Cansancio>3</Nivel_Cansancio>
    <Nivel_Ansiedad>2</Nivel_Ansiedad>
    <Factor_Nerviosismo>4</Factor_Nerviosismo>
    <Factor_Cansancio>4</Factor_Cansancio>
    <Factor_Ansiedad>2</Factor_Ansiedad>
```

```
                <Factor_Fuma>31</Factor_Fuma>
            </Psicológicas>
            <Equipaje>
                <Ligero>2</Ligero>
                <Pesado>3</Pesado>
            </Equipaje>
        </Pasajero>
    </Lista_Pasajeros>

</Vuelo>
```

También se implementará una función que permitirá cargar una lista de personajes donde inicialmente el algoritmo verificará la existencia del fichero contenedor de la lista a través de su nombre. Luego se procede a la interpretación del contenido del archivo para comparar si la estructura interna del fichero es compatible con la estructura generada anteriormente. Posteriormente se capturará el dato contenido en cada etiqueta y finalmente retornará una lista con todos los personajes cargados.

2.3 Características del editor

Actualmente el procedimiento para generar personajes para el sistema de entrenamientos “Indicios Aduaneros” se hace de forma manual, lo cual es un proceso complejo porque solo se puede generar una cantidad exacta de pasajeros y no hay diversidad en sus propiedades. Además, el algoritmo para cargar los personajes contenidos en el fichero XML consume demasiado tiempo debido a que la configuración de cada pasajero se obtiene partiendo de un personaje a otro. La herramienta que se desea crear posibilitará que la generación de personajes se haga de una manera óptima en términos de tiempo, facilitando así la creación de un personaje o una lista de personajes, donde la asignación de propiedades se podrá hacer manual o automática. En caso de ser automática se le distribuyen las propiedades contables al personaje de manera aleatoria. La carga de personajes se hará partiendo de un algoritmo el cual interpreta el fichero devolviendo una lista de personajes que posteriormente van a ser visualizados.

2.4 Interfaz visual

El sistema ofrecerá una interfaz intuitiva y de fácil manejo, que permitirá editar personajes de forma manual, insertando las propiedades del personaje visualizado en ese momento en el viewport representado en la parte derecha de la aplicación. También brindará la posibilidad de generar una cantidad dada de personajes, lo cual se hará de manera automática, asignando las propiedades contables del personaje de manera aleatoria. Ofrecerá además la opción de salvar la configuración de un personaje o una lista de personajes, así como la carga de una lista de personajes almacenada en un fichero XML. Permitirá también el cierre de la aplicación.

2.5 Herramientas a utilizar

2.5.1 Visual Studio (C++)

Visual Studio será el IDE de desarrollo que se usará para programar esta aplicación, ya que ofrece un entorno de peso para Windows. Además puede ser usado para escribir aplicaciones de consola, en su opción de **C++**. Visual Studio proporciona un modelo extensible para añadir nuevos proyectos y muchas otras aplicaciones de Microsoft que ahora se integran directamente en el IDE. Algunos de los más comunes Visual Assist y Visual Studio Tools para Office, además ofrece la posibilidad de adjuntarle motores gráficos (OGRE). Una de las características más importantes de Visual Studio es la capacidad de paso a través de solicitud de línea en tiempo de ejecución (depuración). Está construido para el desarrollo de aplicaciones, por lo que proporciona métodos intuitivos para la organización de los archivos de código en diferentes proyectos y sus diversas aplicaciones en soluciones.

2.5.2 Ogre

Ogre (Object-Oriented Graphics Rendering Engine)
Para el desarrollo de esta aplicación se utilizará la herramienta gráfica OGRE, la cual es un motor gráfico desarrollado en C++, multiplataforma y libre. Permite el renderizado requerido para aplicaciones como la que se pretende desarrollar. Proporciona una biblioteca de clases basada en *Direct3D* y *OpenGL* que contiene un conjunto de clases que permiten la generación de mundos virtuales. Permite que las escenas se dividan en “páginas” (pedazos de un mapa muy grande se procesan uno a la vez, permitiendo cargar escenas muy grandes). Cada página puede tener sus propios *HeightMaps*. Su renderizado está basado

en Octree, BSP (*Binary Space Partition*), y portales. Provee bibliotecas asociadas, como son CEGUI y OIS, las cuales permiten la visualización de elementos de interfaz y el trabajo con los dispositivos de entrada de datos (mouse, teclado, joystick, entre otras), respectivamente. Además cuenta con un conjunto de funciones para el trabajo con valores aleatorios que serán utilizadas para la asignación random de las propiedades contables de los personajes. La función utilizada para realizar esta tarea es *RangeRandom* la cual permite generar valores en un determinado rango.

2.5.3 Qt 4.6.1

Para la realización de la interfaz principal de este trabajo se utilizará Qt versión 4.6.1, ya que proporciona una API orientada a objetos para crear aplicaciones. Dicha API cuenta con los métodos necesarios para el trabajo con ficheros XML, soporte de redes, una API multiplataforma unificada para la manipulación de archivos y una multitud de otras funciones para el trabajo referente a ficheros, conjuntamente con estructuras de datos tradicionales. Provee un sistema gráfico basado en ventanas. Es una biblioteca multiplataforma para desarrollar interfaces de usuario y también para el desarrollo de programas sin interfaz gráfica como son herramientas de la consola y servidores. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios lenguajes de programación a través de **bindings,** que en el campo de la programación, es una adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquel en el que ha sido escrita.

2.5.4 Visual Paradigm for UML 6.4

Una vez seleccionado el lenguaje y la notación de modelado se procede a elegir la herramienta CASE (Computer Aided Software Engineering) a emplear. Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de un software. Permite la generación de código inverso, así como generar código desde diagramas y documentación. Brinda soporte para UML y cuenta con un IDE de integración para Visual Studio.

Por lo antes expuesto se selecciona Visual Paradigm como herramienta para el modelado del sistema a desarrollar, ya que se adapta a las necesidades y requerimientos del sistema permitiendo con esto su modelación de forma íntegra.

2.6 Bibliotecas a utilizar

2.6.1 TinyXML

Los cambios que se hagan con esta herramienta se guardarán en un fichero XML. Para garantizar esto se harán funciones que permitan la salva en dicho fichero, por lo que se utilizará la biblioteca dinámica TinyXML la cual permite cargar, interpretar, modificar un documento con formato XML, incluso brinda la posibilidad de salvar textos en dicho formato.

2.7 Visión general de la arquitectura

A contuación se muestra una visión general de la arquitectura del sistema, con el objetivo de obtener un mayor entendimiento de su funcionamiento.

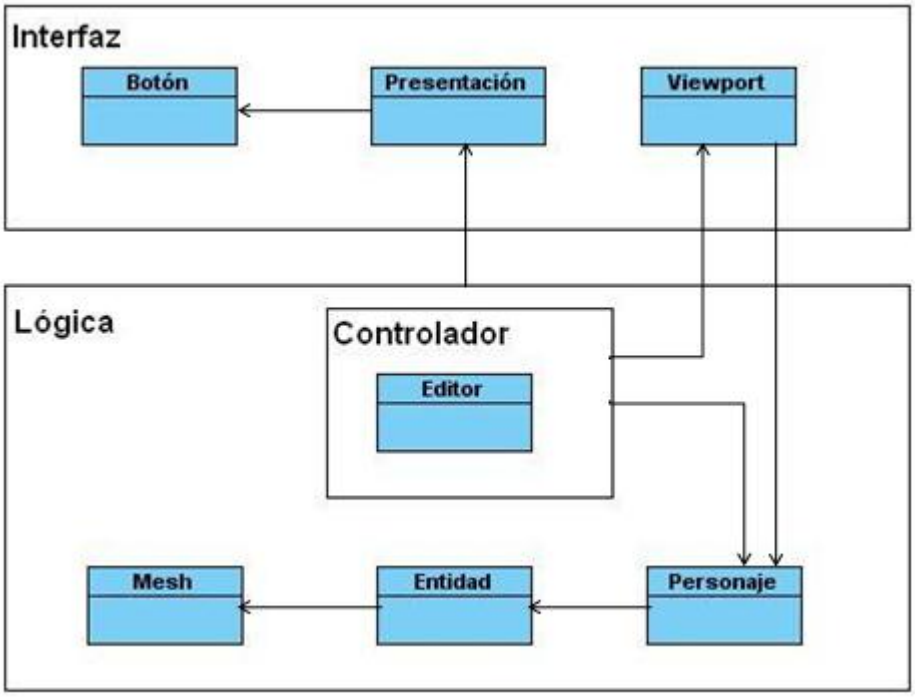

Figura 8 Visión general de la arquitectura.

Capítulo 3: “Características del sistema”

Introducción

En el presente capítulo se hará una caracterización del sistema en términos funcionales, definiendo así, los requisitos funcionales, los casos de uso y sus descripciones. Se describe además, la visión general de la arquitectura, asi como los requisitos no funcionales del sistema.

3.1 Restricciones del sistema

Con el propósito de definir un código que permita satisfacer las necesidades del funcionamiento correcto del editor de personajes, se establecen las siguientes reglas del negocio:

- Los objetos cargados con el editor deben tener el formato mesh, que usa por defecto el motor gráfico Ogre.
- El fichero a cargar debe tener la estructura definida por el editor.
- La dirección donde se guarde el fichero de configuración de los personajes será única.

3.2 Modelo de dominio

El modelo de dominio es una representación visual estática de los objetos en el contexto del proyecto. Es decir, un diagrama con los objetos que existen relacionados con el proyecto y los vínculos que hay entre ellos. Es una representación estática porque no representa la interacción en el tiempo de los objetos, sino que representa una visión “parada” de las clases y sus interacciones.

Debido a que no se cuenta con una definición total de los procesos de negocio, se plantea confeccionar un modelo de dominio en el cual se enmarcan un conjunto de clases que participan en la aplicación y las relaciones que existen entre ellas.

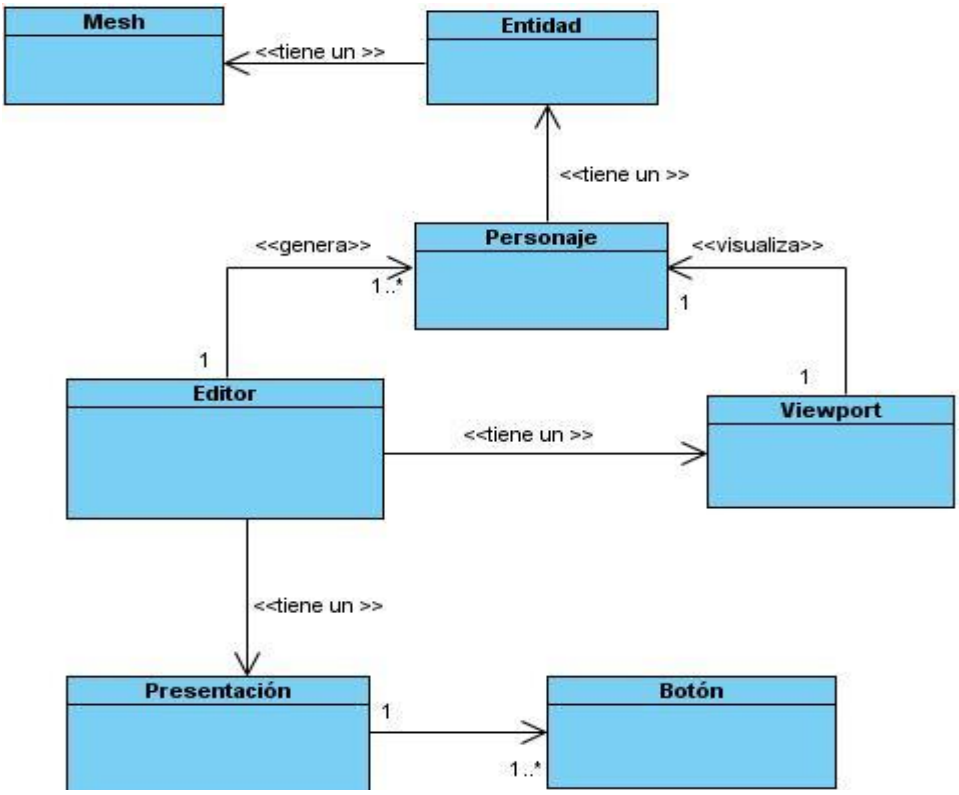

Figura 9 Modelo de dominio.

3.2.1 Conceptos del modelo de dominio

Editor: Herramienta que permite editar personajes, asignándoles propiedades, donde dichos personajes se van a ir almacenando en una lista a medida que se van editando y finalmente se genera un fichero XML donde permanecerá la lista de personajes con todas sus propiedades asignadas anteriormente.

Viewport: Cuadro de visualización, donde se refleja un personaje seleccionado.

Personaje: Es la unión del esqueleto, con la malla del objeto, incluyendo las texturas.

Mesh: Extensión, malla perteneciente al nodo físico de tipo triangular.

Entidad: Nodo físico.

Presentación: Interfaz principal del editor, la cual contiene los vínculos con las principales funciones del mismo.

Botón: Enlaza la presentación con las principales funciones pertenecientes al editor.

3.3 Captura de requisitos

El editor de personajes tendrá que cumplir con una serie de condiciones y capacidades para dar solución al problema planteado. Se define dichas condiciones y funcionalidades en los siguientes requisitos clasificados como funcionales y no funcionales.

3.3.1 Requisitos funcionales

A continuación se enumeran las principales funcionalidades presentes en la aplicación, que son las capacidades con las que el sistema debe cumplir:

RF 1. Iniciar Aplicación.
- RF 1.1. Iniciar Qt.
- RF 1.2. Iniciar OGRE.
- RF 1.3. Iniciar Recursos.
- RF 1.4. Crear escenario de trabajo.

RF 2. Generar Personaje.
- RF 2.1. Asignar datos generales del personaje.
- RF 2.2. Asignar propiedades documentales.
- RF 2.3. Asignar propiedades del equipaje.
- RF 2.4. Asignar propiedades físicas.
- RF 2.5. Asignar propiedades psicológicas.

RF3. Generar Personajes Automáticamente.
- RF 3.1. Escoger valores aleatorios.

RF 4. Rotar Cámara.

RF 5. Salvar Lista de Personajes.

RF 6. Cargar Lista de Personajes.

RF 7. Salir de la Aplicación.

RF 7.1. Destruir Recursos.

3.3.2 Requisitos no funcionales

A continuación se enumeran las cualidades que la aplicación debe presentar:

Legales: Se utilizarán bibliotecas externas con licencias libres, por ejemplo TinyXML 2.5.3, la cual está bajo licencia GNU (GPL y LGPL).

Interfaz: Dentro de los tipos de interfaz se utilizarán.

- **Interfaz de hardware**, a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado y mouse.
- **Interfaz de software**, destinada a entregar información acerca de los procesos de control, a través de lo que el usuario observa habitualmente en la pantalla.

Usabilidad: El sistema debe proporcionar una interfaz sencilla y fácil de entender.

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

Documentación: Se utilizará documentación de código con los estándares establecidos en el proyecto.

Hardware: Compatibilidad con tarjetas gráficas de la familia *NVIDIA* (*Geforce* 3, *Geforce* 4, FX 5200). Se necesita un ordenador con CPU Intel Pentium IV, 1 Gb de HDD, 512 Mb de RAM.

Software: Se utilizará el IDE de desarrollo Microsoft Visual Studio 2008; el motor gráfico OGRE 1.6.0; la biblioteca de desarrollo de interfaces gráficas de usuario QT 4.6.1 y el parser de XML TinyXML 2.5.3.

3.4 Definición de los Casos de Uso del Sistema

3.4.1 Actores del sistema

Tabla 1 Actores del Sistema.

Actor	Descripción
Usuario	Responsable de iniciar la interacción con el sistema, además tiene los privilegios necesarios para generar, guardar y salvar listas de personajes.

3.4.2 Casos de Uso del Sistema

CU1. Iniciar Aplicación.

CU2. Generar Personaje.

CU3. Generar Personajes Automáticamente.

CU5. Rotar Cámara.

CU6. Salvar Lista de Personajes.

CU7. Cargar Lista de Personajes.

CU8. Salir de la Aplicación.

3.4.3 Diagrama de Casos de Uso del Sistema

El diagrama de Casos de Uso del Sistema ofrece una panorámica general que se ha definido para satisfacer los requisitos funcionales del sistema como tal.

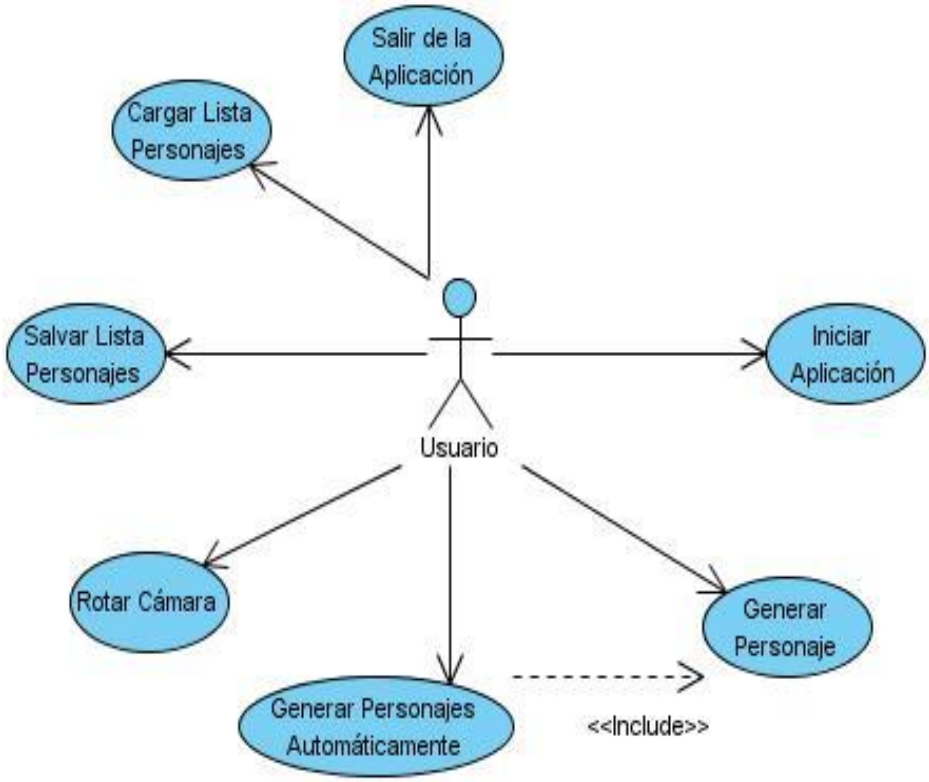

Figura 10 Diagrama de Casos de Uso del Sistema.

3.4.4 Especificación de los Casos de Uso del Sistema

Para lograr un mayor entendimiento de las funcionalidades asociadas a cada caso de uso, no es suficiente la representación gráfica de dichos casos de uso, por lo que a continuación se muestra la descripción de los casos de uso definidos en el sistema.

Tabla 2 Descripción del caso de uso “Iniciar Aplicación”

Caso de Uso:	Iniciar Aplicación
Actores:	Usuario
Propósito:	Permitirle al actor la ejecución del editor.
Resumen:	El caso de uso se inicia cuando el usuario ejecuta la aplicación. Posteriormente se inicializan los recursos necesarios para levantar la interfaz del editor. Finaliza cuando se muestra la interfaz de la aplicación.

Precondiciones:	-
Referencias	RF1
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Realiza acción que da inicio al caso de uso.	
	1.1. Inicia los recursos de la aplicación.
	1.2. Activa la captación de eventos de entrada de datos (Mouse y Teclado).
	1.3. Crea el escenario de trabajo.
(Ver figura 21)	
Poscondiciones	El sistema levanta la aplicación, la cual está lista para su uso.

Tabla 3 Descripción del caso de uso “Generar Personaje”

Caso de Uso:	Generar Personaje
Actores:	Usuario
Propósito:	Permitirle al actor asignar propiedades (Documentos (Pasaporte, Boleto, Boucher), Físicas, Psicológicas, Equipaje) y generar una cantidad deseada de personajes, ya sea manual o automáticamente.
Resumen:	El caso de uso se inicia cuando el usuario pulsa en el menú la opción “Generar Personaje”. Luego el sistema muestra un panel con todas las propiedades que se pueden asignar a un personaje, donde el usuario va a modificar dichas propiedades y al final pulsa el botón “Generar”. Seguidamente el sistema muestra un panel para especificar la cantidad de personajes que se desean generar, donde el usuario pulsa el botón “Generar”. Luego el sistema genera el personaje que editó el usuario manualmente y por último la cantidad de personajes entrada por el usuario, donde después se guarda se almacenan en una estructura de datos. El caso de uso finaliza cuando se almacenan en memoria los personajes

	generados.
Precondiciones:	-
Referencias	RF3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Pulsa en el menú la opción “Editar Personaje”.	
	1.2. Muestra un panel de edición con todas las propiedades que se le pueden adjuntar a un personaje.
2. Asigna las propiedades al personaje seleccionado.	
2.1. Pulsa el botón “Generar” del panel de edición.	
	3. Muestra un panel para especificar la cantidad de personajes a generar.
4. Especifica la cantidad a generar y pulsa el botón “Generar” del panel.	
	5. Genera el personaje editado por el usuario inicialmente y la cantidad entrada por él mismo y luego los almacena en una estructura de datos (lista) en memoria.
(Ver figura 22)	
Poscondiciones	EL sistema guarda en memoria los personajes generados.

Tabla 4 Descripción del caso de uso “Generar Personajes Automáticamente”

Caso de Uso:	Generar Personajes Automáticamente
Actores:	Usuario
Propósito:	Permitirle al actor la generación de personajes de manera automática.

Resumen:	El caso de uso se inicia cuando el usuario pulsa en el menú la opción “Generar Automático”. Posteriormente el sistema muestra un campo para especificar la cantidad de personajes que se desean generar. Luego el usuario entra la cantidad deseada y pulsa el botón “OK”. El sistema genera dicha cantidad de personajes y los almacena en una estructura de datos. El caso de uso finaliza cuando se almacenan en memoria los personajes generados.
Precondiciones:	Se debe haber ejecutado el CU Generar Personaje
Referencias	RF4
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Pulsar el botón “Generar personajes”.	
	1.1 Muestra un campo para la especificación de la cantidad de personajes que se desean generar.
2. Entra la cantidad de personajes que desea que se generen.	
2.1. Pulsa el botón “OK” del cuadro de especificación.	
	3. Genera la cantidad de personajes entrada por el usuario.
	3.1. Almacena la los personajes generados en una estructura de datos (lista) en memoria.
(Ver figura 23)	
Poscondiciones	EL sistema guarda en memoria los personajes generados.

Tabla 5 Descripción del caso de uso “Rotar Cámara”

Caso de Uso:	Rotar Cámara
Actores:	Usuario

Propósito:	Permitirle al actor rotar la cámara para obtener una mayor visualización del personaje visible en ese momento.
Resumen:	El caso de uso se inicia cuando el usuario pulsa en el menú la opción Rotar Personaje o acciona las teclas de rotación establecidas. Posteriormente el sistema toma el personaje seleccionado y lo rota hacia la izquierda en el eje de coordenadas Y o rota el personaje hacia una dirección dependiendo de la tecla de control que se haya pulsado. El caso de uso finaliza cuando el sistema termine la ejecución de la rotación del personaje visualizado en ese momento.
Precondiciones:	Se debe haber pulsado en el menú la opción “Editar Personaje” o “Cargar Personajes”.
Referencias	RF5
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
a) Si pulsa en la aplicación el botón Rotar Personaje ir a sección “Rotar por botón”. b) Si pulsa las teclas de rotación (A, D, S, W o Left, Right, Up, Down) ir a sección “Rotar por teclas”.	
Sección “Rotar por botón”	
Acción del Actor	**Respuesta del Sistema**
1. Pulsa en la aplicación el botón ‘Rotar Personaje”.	
	1.1. Rota el personaje seleccionado, hacia la izquierda con un ángulo de 360 grados en el eje de coordenadas Y.
Sección “Rotar por teclas”	
Acción del Actor	**Respuesta del Sistema**

a) Si pulsa las teclas Up o W ir a sección “Izquierda”. b) Si pulsa las teclas Down o S ir a sección “Derecha”.	
	a.1) 1.1 El sistema rota el personaje 360 grados en dirección izquierda. b.1) 1.1 El sistema rota el personaje 360 grados en dirección derecha.
Poscondiciones	El sistema cambia la posición del personaje dependiendo del tiempo que se haya oprimido las teclas o el botón para rotar.

Tabla 6 Descripción del caso de uso “Salvar Lista Personajes”

Caso de Uso:	Salvar Lista Personajes
Actores:	Usuario
Propósito:	Permitirle al actor guardar la configuración de los personajes generados.
Resumen:	El caso de uso se inicia cuando el usuario pulsa en el menú la opción Salvar. Luego el sistema toma la lista donde se estaban almacenando todos los personajes generados y genera un fichero XML donde se va a guardar la lista de todos los personajes con las propiedades de cada uno de ellos. El caso de uso finaliza cuando se generó el fichero XML.
Precondiciones:	Se debe haber generado al menos 1 personaje.
Referencias	RF6, RF3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Pulsa en el menú la opción Salvar.	
	1.1. Genera un fichero XML donde va a almacenar definitivamente los personajes que se

	habían guardado en la estructura de datos (lista).
Poscondiciones	El sistema genera un fichero con la configuración de los personajes que fueron generados.

Tabla 7 Descripción del caso de uso “Cargar Lista Personajes”

Caso de Uso:	Cargar Lista Personajes
Actores:	Usuario
Propósito:	Permitirle al actor cargar una lista de personajes desde un fichero XML.
Resumen:	El caso de uso se inicia cuando el usuario pulsa en el menú la opción “Cargar Personajes”. Seguidamente el sistema verifica la existencia del fichero y muestra una ubicación donde se encuentran todos los ficheros a cargar. Luego especifica el fichero a cargar y finalmente el sistema carga la lista de personajes. El caso de uso finaliza cuando se visualizan los personajes cargados.
Precondiciones:	El archivo donde se concentran los ficheros XML de configuración no debe estar vacío.
Referencias	RF7
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario pulsa en el menú la opción “Cargar Personajes”.	
	1.1. Muestra un cuadro de especificación de directorios.
2. Especifica la ruta del fichero que desea cargar y pulsa “abrir”.	
	2.1. Verifica que el nombre del fichero seleccionado por el usuario se corresponde con el nombre de un fichero generado por el editor.

	2.2. Verifica que la estructura del fichero sea correcta.
	2.3. Notifica al usuario el estado de la carga (si se cargó correctamente).
	2.4. Muestra el listado de los personajes cargados que estaban guardados en el fichero XML.
Flujos Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si el nombre del fichero seleccionado no concuerda con el nombre definido, el sistema muestra un mensaje anunciando que no reconoce el fichero seleccionado.
Flujos Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	2.2 Si la estructura del fichero seleccionado no se corresponde con la estructura definida por el editor, el sistema muestra un mensaje orientando que estructura del fichero no es compatible con la estructura definida.
(Ver figura 24)	
Poscondiciones	El sistema visualiza los personajes cargados, simultáneamente con las propiedades pertenecientes a cada uno.

Tabla 8 Descripción del caso de uso “Salir de la Aplicación”

Caso de Uso:	Salir de la Aplicación
Actores:	Usuario

Propósito:	Permitirle al actor el cierre de la aplicación.
Resumen:	El caso de uso se inicia cuando el usuario pulsa en el menú la opción Salir. Posteriormente el sistema destruye los recursos y objetos inicializados al inicio de la aplicación y luego cierra el editor.
Precondiciones:	
Referencias	RF8
Prioridad	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Pulsa en el menú la opción Salir.	
	1.2. Destruye todos los recursos relacionados con la aplicación.
	1.3. Cierra automáticamente la aplicación.
Poscondiciones	La aplicación se cierra automáticamente.

Capítulo 4 “Diseño e Implementación”

Introducción

El diseño y la implementación constituyen dos etapas fundamentales dentro del proceso de desarrollo de un software. En el presente capítulo se muestra el diseño de clases del sistema, así como los atributos presentes en ellas, definiendo además las responsabilidades y relaciones en dichas clases. Seguidamente se puntualizan los diagramas de secuencia por cada caso de uso que intervendrán en el primer ciclo de desarrollo del software obteniendo así un mejor entendimiento de la herramienta.

Esta fase del proyecto contiene también la creación del diagrama de componentes que se traduce a ficheros .h y .cpp correspondientes a la implementación del sistema. Se especifica además la nomenclatura conjuntamente con los estándares de codificación.

4.1 Estándares de codificación

Nombre de los ficheros:

Se nombrarán los ficheros .h y .cpp de la siguiente forma:

CE_NombreUnit.h

Tipos de datos se nombrarán mediante el siguiente patrón:

Objetos:

Los objetos se nombrarán con *pt* delante del nombre del objeto.

ptNombreObjeto

Booleanos:

bool blNombre

Indicando *bl* delante del nombre de la variable.

Enteros:

int nNombre

Indicando *n* delante del nombre de la variable.

Reales:

Ogre::Real rNombre

Indicando *r* delante del nombre de la variable.

Listas:

pkNombreLista

Indicando *pk* delante del nombre de la lista.

Iteradores:

std::vector<Ogre::Entity*>::iterator itNombre

Indicando *it* delante del nombre del iterador.

Cadenas:

std::string strNombre

Indicando *str* delante del nombre de la variable.

Argumentos:

void Set_Nivel_Nerviosismo(int argNervios)

Indicando *arg* delante del nombre del argumento.

Clases:

class CE_Nombre

Indicando con *CE_* delante del nombre de la clase.

4.2 Diagrama de clases de diseño

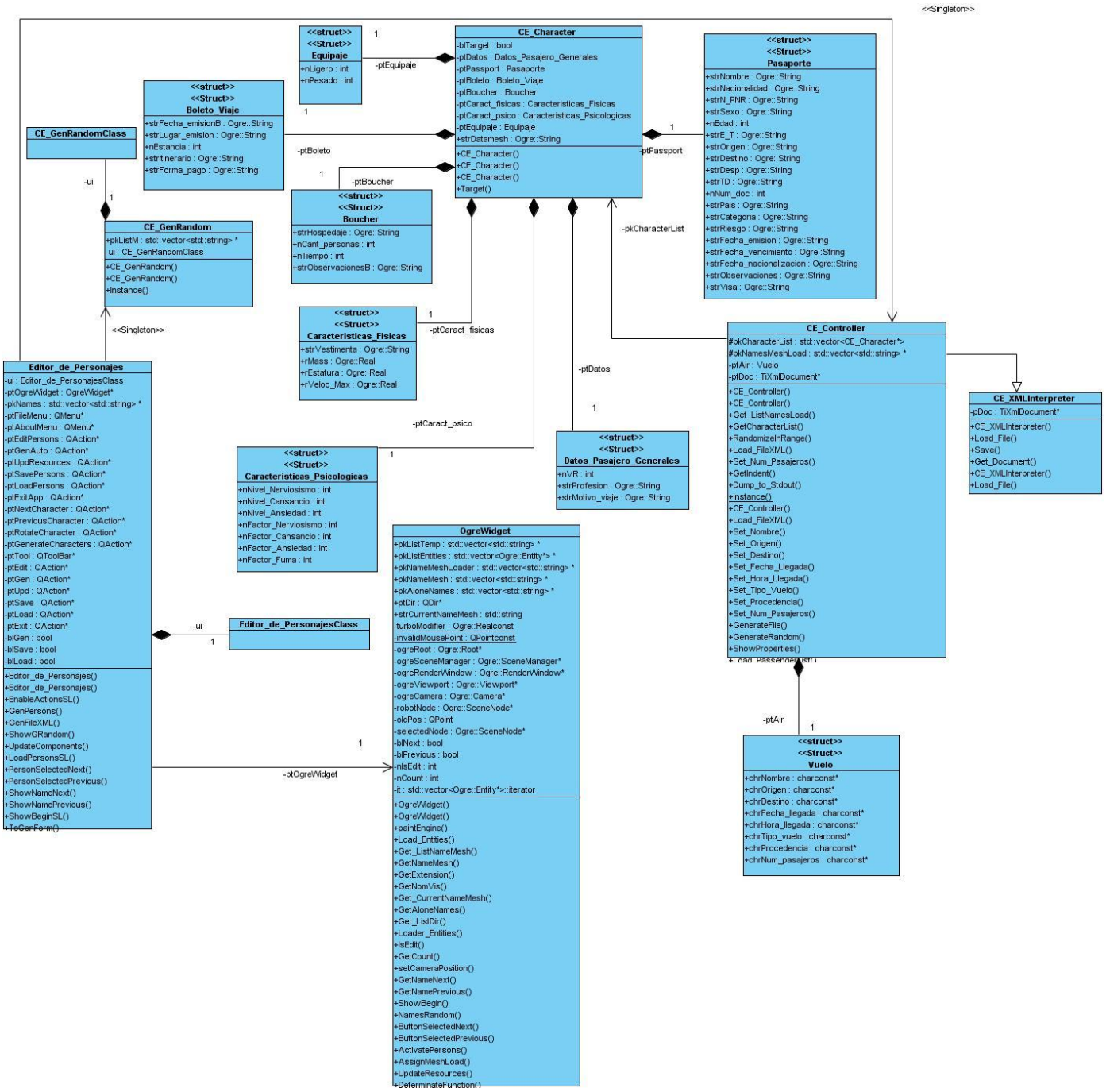

Figura 11 Diagrama de clases del diseño.

4.3 Patrones utilizados

Se emplea el patrón de arquitectura **Modelo-Vista-Controlador**, donde la clase CE_Character representa el modelo, la clase CE_Editor_de_Personajes la vista y la clase CE_Controller el control.

Patrones de diseño

Singleton: La figura 11 muestra las clases correspondientes a la implementación en C++ del sistema.Se emplea el patrón “Singleton” ya que este permite que se puedan hacer llamadas a las clases CE_Controller y CE_GenRandom desde cualquier otra clase, es decir creando un punto de acceso global a ella. Esto se evidencia implementando en las clases anteriormente mencionadas un método que crea una instancia del objeto sólo si todavía no existe alguna.

Controller: Se emplea el patrón controlador el cual sirve como un intermediario entre la clase CE_Editor_de_personajes (interfaz) y CE_Controller (controladora) que contiene las funciones que dan respuesta a la interfaz.

Patrones de asignación de responsabilidades

Experto: Se emplea al asignar a la clase CE_OgreWidget la responsabilidad de realizar el renderizado de los personajes, debido a que contiene las funciones necesarias para el trabajo con el gráfico.

4.4 Descripción de las clases de diseño

Tabla 9 “Descripción de la clase del diseño CE_OgreWidget”

CE_OgreWidget	
Tipo de clase	Interfaz
Atributo	**Tipo**
ogreRoot	Ogre::Root
ogreSceneManager	Ogre::SceneManager

ogreRenderWindow	Ogre::RenderWindow
ogreViewport	Ogre::Viewport
ogreCamera	Ogre::Camera
robotNode	Ogre::SceneNode
oldPos	QPoint
selectedNode	Ogre::SceneNode
blNext	bool
blPrevious	bool
nIsEdit	int
nCount	int
it	std::vector<Ogre::Entity*>::iterator
turboModifier	static const Ogre::Real
invalidMousePoint	static const QPoint
ptDir	QDir
strCurrentNameMesh	std::string
pkListEntities	std::vector<Ogre::Entity*>
pkListTemp	std::vector<std::string>
pkNameMeshLoader	std::vector<std::string>
pkNameMesh	std::vector<std::string>
pkAloneNames	std::vector<std::string>
Para cada responsabilidad	
Nombre:	InitOgreSystem()
Descripción:	Inicializa los objetos de manipulación de Ogre.
Nombre:	SetupNLoadResources()
Descripción	Inicializa los recursos de Ogre.
Nombre	CreateScene()
Descripción:	Crea escenario de trabajo.
Nombre:	Update()
Descripción:	Actualiza el renderizado en el viewport.
Nombre:	keyPressEvent(QKeyEvent * e)

Descripción:	Captura el accionamiento de una tecla.
Nombre:	moveEvent(QMoveEvent * e)
Descripción:	Proporciona el movimiento del mouse.
Nombre:	mouseDoubleClickEvent(QMouseEvent * e)
Descripción:	Captura el accionamiento del doble click izquierdo.
Nombre:	mouseMoveEvent(QMouseEvent * e)
Descripción:	Captura el movimiento del mouse.
Nombre:	mousePressEvent(QMouseEvent * e)
Descripción:	Captura el accionamiento de cualquier tecla del mouse.
Nombre:	mouseReleaseEvent(QMouseEvent * e)
Descripción:	Captura la liberación de cualquier tecla del mouse.
Nombre:	paintEvent(QPaintEvent * e)
Descripción:	Actualiza el renderizado en el Viewport.
Nombre:	resizeEvent(QResizeEvent * e)
Descripción:	Establece un campo de acción del mouse, dentro de un área determinada.
Nombre:	showEvent(QShowEvent * e)
Descripción:	Activa los eventos de Ogre y Qt inicializados anteriormente.
Nombre:	wheelEvent(QWheelEvent * e)
Descripción:	Modifica el control de la cámara.
Nombre:	cameraPositionChanged(const Ogre::Vector3 &pos)
Descripción:	Actualiza la posición de la cámara.
Nombre:	ButtonSelectedNext()
Descripción:	Visualiza el personaje siguiente en la lista de personajes.
Nombre:	ButtonSelectedPrevious()
Descripción:	Visualiza el personaje anterior en la lista de personajes.
Nombre:	setBackgroundColor(QColor c)
Descripción:	Cambia el color al viewport.
Nombre:	setCameraPosition(const Ogre::Vector3 &pos)
Descripción:	Actualiza la posición de la cámara que visualiza los personajes en el viewport.
Nombre:	GetNameNext()

Descripción:	Dado un personaje actual, retorna el nombre del mesh asociado al personaje siguiente.
Nombre:	GetNamePrevious()
Descripción:	Dado un personaje actual, retorna el nombre del mesh asociado al personaje anterior.
Nombre:	ShowBegin()
Descripción:	Retorna el nombre del mesh del personaje que está en el inicio de la lista.
Nombre:	NamesRandom(unsigned in argCant)
Descripción:	Asigna una cantidad dada de nombres de manera aleatoria.
Nombre:	ActivatePersons()
Descripción:	Adjunta las entidades al nodo, para ser visualizados los personajes.
Nombre:	AssignMeshLoad(std::vector<std::string> * argList)
Descripción:	Coloca la los personajes cargados dentro de la lista de personajes.
Nombre:	DestroyOgreSystem()
Descripción:	Destruye todos los recursos necesarios para Ogre.
Nombre:	Load_Entities()
Descripción:	Crea las entidades y las almacena en una lista.
Nombre:	Get_ListNameMesh()
Descripción:	Coloca en una lista todos los archivos presentas en la carpeta models del proyecto.
Nombre:	GetNameMesh(std::string argName)
Descripción:	Retorna el nombre del mesh (sin extensión).
Nombre:	GetExtension(std::string argName)
Descripción:	Retorna la extensión de un archivo.
Descripción:	GetNomVis()
Nombre:	Retorna el nombre del mesh asociado al personaje que esta visible en ese momento.
Descripción:	Get_CurrentNameMesh()
Nombre:	Retorna la variable strCurrentNameMesh.
Descripción:	GetAloneNames()
Descripción:	Retorna la lista de nombres de mesh.

Nombre:	Get_ListDir()
Descripción:	Retorna la lista de archivos.
Nombre:	Loader_Entities()
Descripción:	Retorna la lista de nombres de personajes cargados.
Descripción:	IsEdit()
Nombre:	Selecciona que función se va ejecutar.

Tabla 10 “Descripción de la clase del diseño CE_Controller”

CE_Controller	
Tipo de clase	Controladora
Atributo	**Tipo**
pkCharacterList	std::vector<CE_Character*>
pkNamesMeshLoad	std::vector<std::string>
ptAir	Vuelo
ptDoc	TiXmlDocument
Para cada responsabilidad	
Nombre:	Instance()
Descripción:	Crea una instancia de la clase (Singleton).
Nombre:	Get_ListNamesLoad()
Descripción	Retorna la lista de los nombres de los mesh de los personajes cargados.
Nombre	GetCharacterList()
Descripción:	Retorna la lista de los personajes.
Nombre:	RandomizeInRange(int argLowerBound, int argHigherBound)
Descripción:	Selecciona valores dentro de un rango determinado.
Nombre:	Load_FileXML(const char * nameDoc)
Descripción:	Verifica la existencia del fichero dado un nombre.
Nombre:	Void DataAssignation(Ogre::String argNameMesh,Ogre::String rgA,Ogre::String argB, Ogre::String argC,Ogre::String argD, Ogre::String argE ,Ogre::String argF, Ogre::String argG, Ogre::String argH,Ogre::String argI,Ogre::String argJ,

	Ogre::String argK,Ogre::String argL,Ogre::String argM,Ogre::String argN, Ogre::String argO, Ogre::String argP, Ogre::String argQ,Ogre::String argR, Ogre::String argS,Ogre::String argT, Ogre::String argU, Ogre::String argV, Ogre::String argW, Ogre::String argX,Ogre::String argY, Ogre::String argZ, Ogre::String argAb, Ogre::String argCd, Ogre::String argEf, Ogre::String argGh,Ogre::String argIj, Ogre::String argUn,Ogre::String argDue,Ogre::String argFo,Ogre::String argKl, Ogre::String argMn, Ogre::String argOp,Ogre::String argQr,Ogre::String argLoi,Ogre::String argIuy,Ogre::String argQwe, Ogre::String argSt, Ogre::String argUv);
Descripción:	Asigna los datos a un personaje.
Nombre:	GenerateFile()
Descripción:	Genera un fichero XML con todos los personajes generados.
Nombre:	GenerateRandom(unsigned int cant,std::vector<std::string> * argList)
Descripción:	Genera una cantidad dada de personajes, asignando sus propiedades contables de manera aleatoria.
Nombre:	ShowProperties()
Descripción:	Retorna un personaje dado de la lista que se cargó.
Nombre:	Load_PassengerList()
Descripción:	Carga un fichero XML antes generado.

Tabla 11 “Descripción de la clase del diseño CE_Editor_de_personajes”

CE_Editor_de_personajes	
Tipo de clase	Interfaz
Atributo	**Tipo**
ptOgreWidget	OgreWidget
ui	Editor_de_PersonajesClass
pkNames	std::vector<std::string>
ptFileMenu	QMenu
ptAboutMenu	QMenu
ptEditPersons	QAction
ptGenAuto	QAction

ptSavePersons	QAction
ptLoadPersons	QAction
ptExitApp	QAction
ptNextCharacter	QAction
ptPreviousCharacter	QAction
ptGenerateCharacters	QAction
ptTool	QToolBar
ptEdit	QAction
ptGen	QAction
ptSave	QAction
ptLoad	QAction
ptExit	QAction
blGen	bool
blSave	bool
blLoad	bool
Para cada responsabilidad	
Nombre:	CreateActions()
Descripción:	Crea los botones.
Nombre:	CreateMenu()
Descripción	Crea el menú.
Nombre	CreateToolBar()
Descripción:	Crea la barra de herramientas.
Nombre:	DisableActions()
Descripción:	Desabilita los botones, elementos del menú y de la barra de herramientas.
Nombre:	ChangeListNames()
Descripción:	Captura los nombres de los mesh asociados a los personajes, que se encuentran en la carpeta models.
Nombre:	EditPersons()
Descripción:	Habilita el panel de edición de personajes.

Nombre:	EnableActionsSL()
Descripción:	Hablilita los botones, elementos del menú y de la barra de herramientas.
Nombre:	GenPersons()
Descripción:	Captura los datos entrados por el usuario y verifica que no quede ningún campo vacío.
Nombre:	GenFileXML()
Descripción:	Hace una llamada a la función que genera el fichero XML en la clase CE_Controller.
Nombre:	ShowGRandom()
Descripción:	Muestra un panel de especificación de una cantidad de personajes a generar de manera aleatoria y automática.
Nombre:	UpdateComponents()
Descripción:	Muestra los datos del personaje que ya fue cargado y que en ese momento se está visualizando.
Nombre:	LoadPersonsSL()
Descripción:	Busca el camino donde se encuentra el fichero a cargar, luego hace una llamada a la función Load_PassengerList de la clase CE_Controller y luego otra llamada a la función que activa los personajes en la clase CE_OgreWidget.
Nombre:	PersonSelectedNext()
Descripción:	Hace una llamada a la función que visualiza un personaje siguiente de uno actual.
Nombre:	PersonSelectedPrevious()
Descripción:	Hace una llamada a la función que visualiza un personaje siguiente de uno actual.
Nombre:	ShowNameNext()
Descripción:	Captura el nombre del mesh asociado al personaje siguiente de uno actual.
Nombre:	ShowNamePrevious()
Descripción:	Captura el nombre del mesh asociado al personaje anterior de uno actual.
Nombre:	ShowBeginSL()
Descripción:	Hace una llamada a la función que muestra el nombre del personaje inicial en la clase CE_OgreWidget.
Nombre:	ToGenForm(unsigned int argC)

Descripción:	Hace una llamada a la función que genera personajes de manera aleatoria y automática en la clase CE_Controller.
Nombre:	ActivatePersonsSL()
Descripción:	Hace una llamada a la función que activa los personajes en la clase CE_OgreWidget.

Tabla 12 "Descripción de la clase del diseño CE_GenRandom"

CE_GenRandom	
Tipo de clase	Interfaz
Atributo	**Tipo**
ui	CE_GenRandomClass
Para cada responsabilidad	
Nombre:	AssignCantRandom()
Descripción:	Captura un número de personajes a generar de manera aleatoria y automática.

Tabla 13 "Descripción de la clase del diseño CE_GenRandom"

CE_XMLInterpreter	
Tipo de clase	Controladora
Atributo	**Tipo**
ptDoc	TiXmlDocument
Para cada responsabilidad	
Nombre:	Load_File(const char * name)
Descripción:	Función para cargar el fichero XML.
Nombre:	Save()
Descripción:	Función para generar el fichero XML.
Nombre:	Get_Document()
Descripción:	Retorna el documento generado.

Tabla 14 “Descripción de la clase del diseño CE_Character”

CE_Character	
Tipo de clase	Entidad
Atributo	**Tipo**
ptDatos	Datos_Pasajero_Generales
ptPassport	Pasaporte
ptBoleto	Boleto_Viaje
ptBoucher	Boucher
ptCaract_fisicas	Caracteristicas_Fisicas
ptCaract_psico	Caracteristicas_Psicologicas
ptEquipaje	Equipaje
blTarget	bool
Para cada responsabilidad	
Nombre:	Funciones que retornan todas las variables contenidas en las estructuras pertenecientes al personaje.

4.5 Diagramas de secuencia

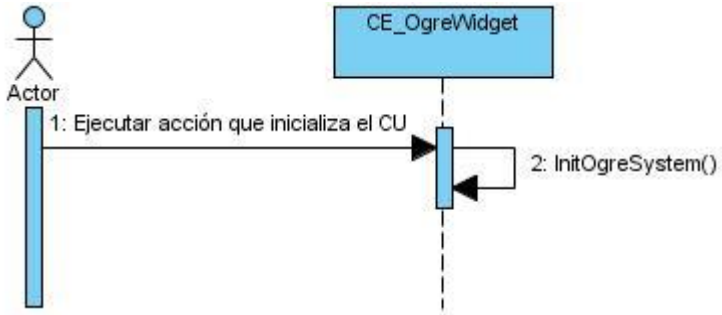

Figura 12 Diagrama de secuencia CU Iniciar la Aplicación.

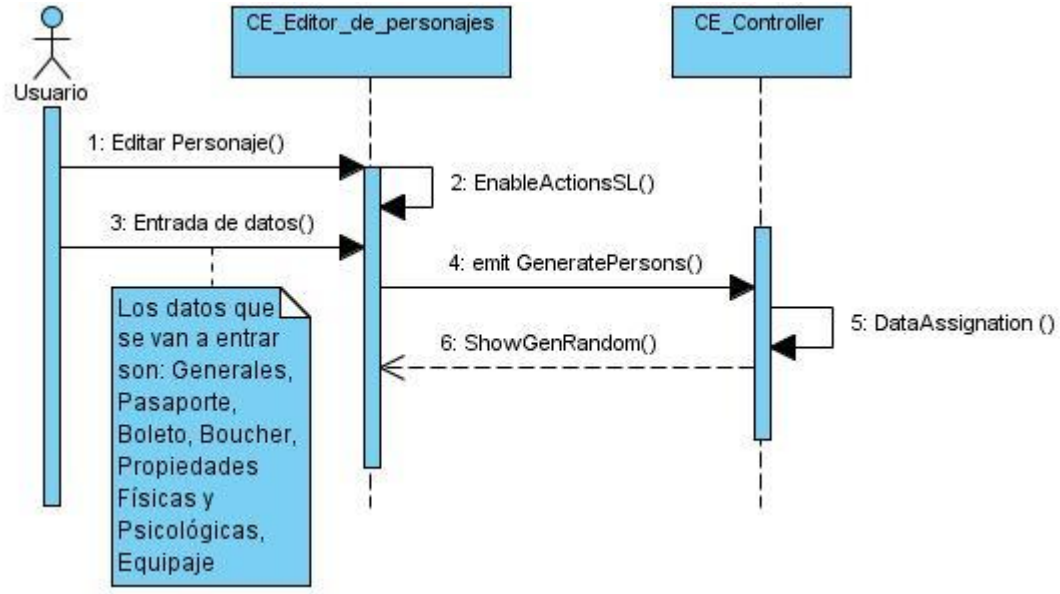

Figura 13 Diagrama de secuencia CU Generar personaje.

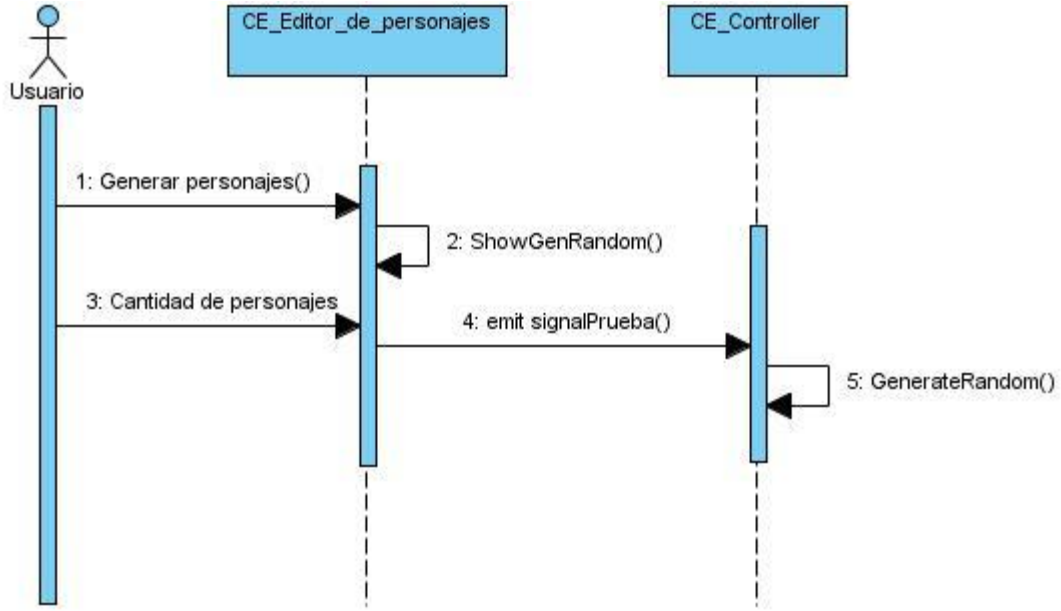

Figura 14 Diagrama de secuencia CU Generar Personajes Automáticamente.

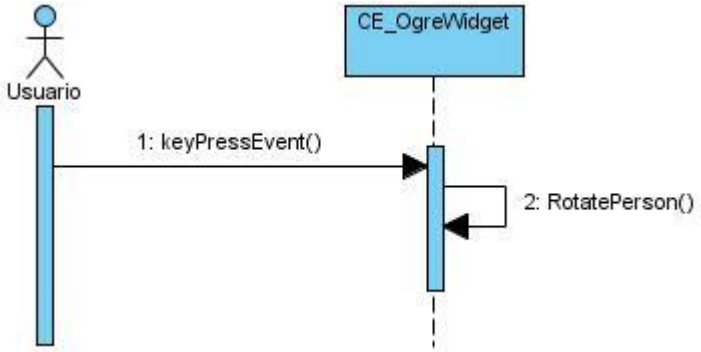

Figura 15 Diagrama de secuencia CU Rotar Personaje.

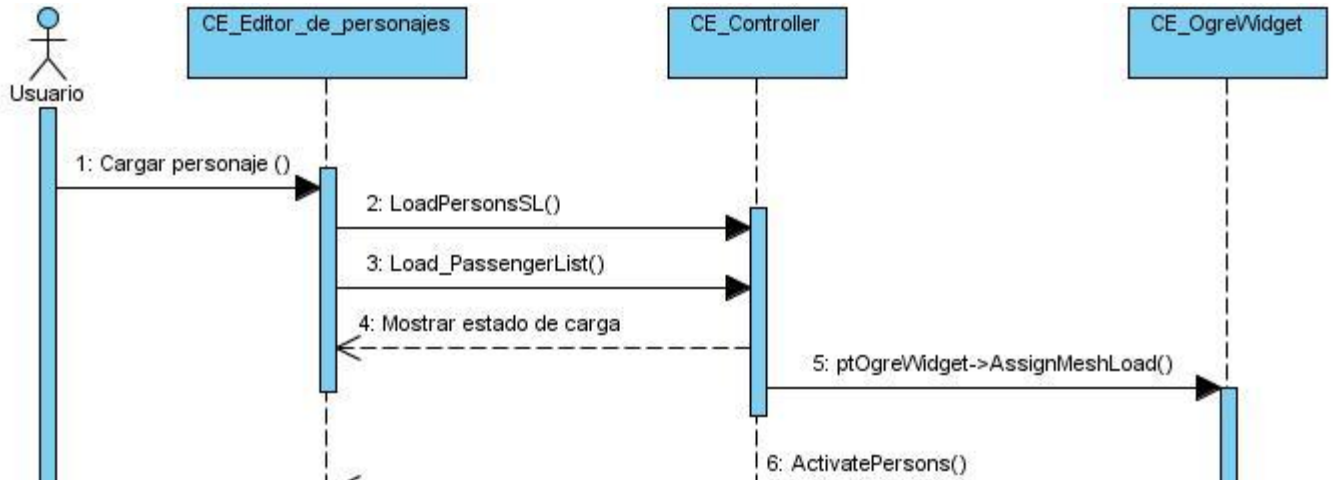

Figura 16 Diagrama de secuencia CU Cargar Lista de Personajes.

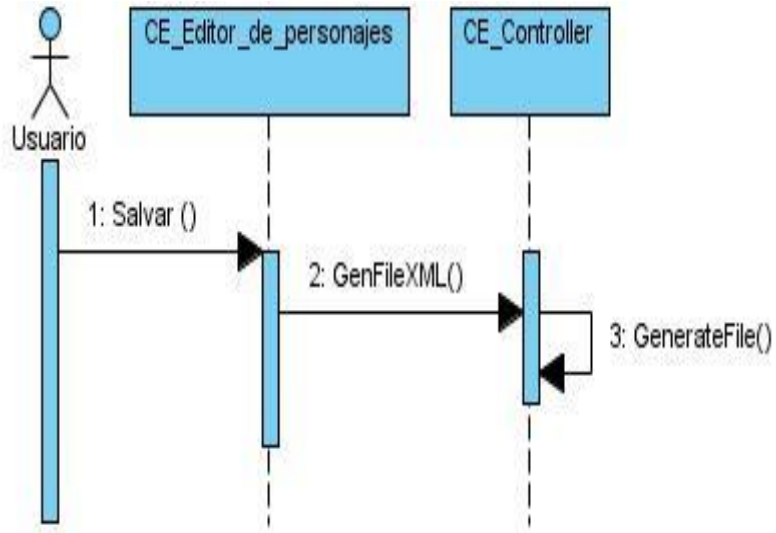

Figura 17 Diagrama de secuencia CU Salvar Lista de Personajes.

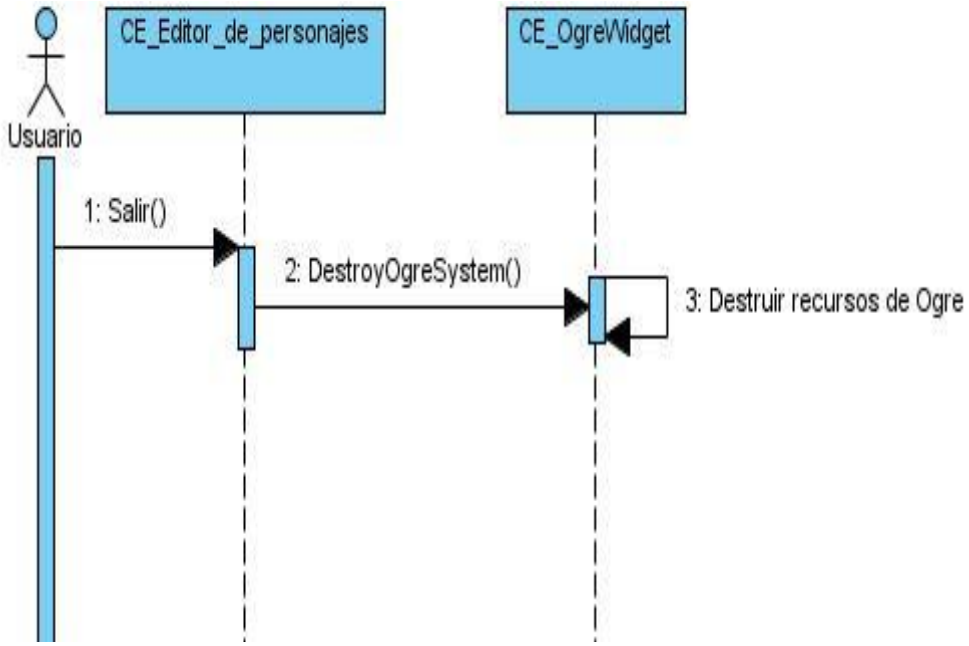

Figura 18 Diagrama de secuencia CU Salir de la Aplicación.

4.6 Diagrama de componentes

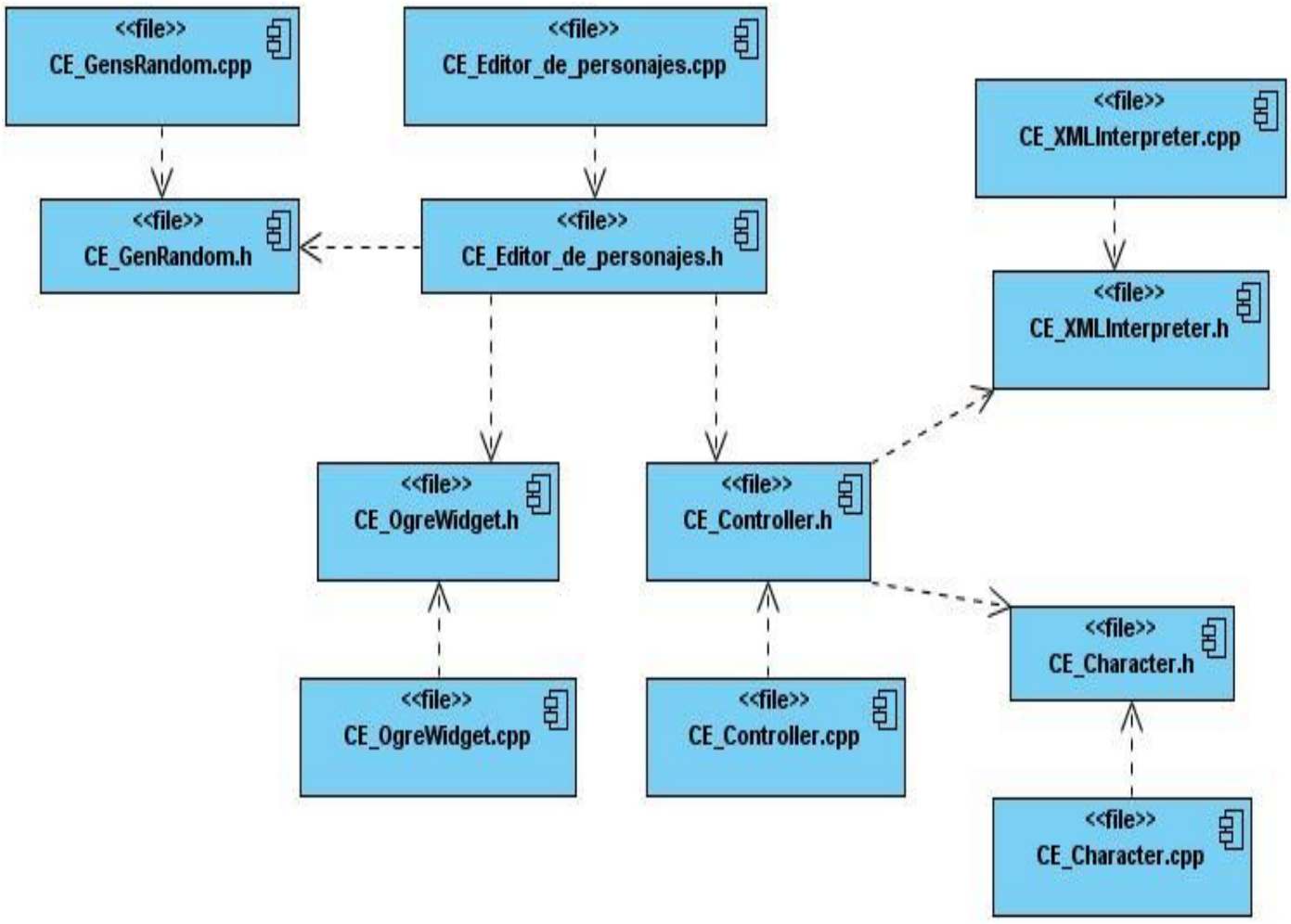

Figura 19 Diagrama de componentes.

4.7 Diagrama de despliegue

Se muestra la distribución física de los nodos que forman el sistema. El producto “Editor de personajes para el sistema de entrenamientos Indicios Aduaneros” solamente necesita un nodo físico (una pc) para desarrollar y ejecutar el software el cual se muestra a continuación.

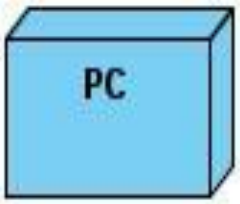

Figura 20 Diagrama de despliegue.

4.8 Validación del módulo

Dentro del código perteneciente al editor de personajes se encuentra un conjunto de clases que se van a integrar posteriormente al módulo GameLogic residente en el código del sistema de entrenamientos “Indicios Aduaneros”. Estas clases son CE_XMLInterpreter, la cual contiene funciones para habilitar la carga y descarga del fichero XML, la clase entidad CE_Character debido a que incluye todos los datos pertenecientes al personaje. Además está la clase CE_Controller de la que solo se va a tomar la función LoadPassengerList (), que permite la carga de un fichero XML con una lista de personajes incluida; la validación de esta función está presente en el editor cuando cargamos un fichero ya generado y se visualiza la lista de personajes que se encuentra dentro del fichero al igual que las propiedades pertenecientes a cada uno de ellos. Esto tiene gran importancia debido a que la función se pudo probar internamente en la aplicación, es decir se logró cargar desde un fichero una lista de personajes.

Conclusiones

Una vez concluida la investigación previa al estudio de la edición de objetos se comprobó en la práctica que el método que se usaba anteriormente para la carga de una lista de personajes no era el adecuado, debido a que presentaba muchas restricciones como la exactitud de los personajes, así como la demora en tiempo de ejecución.

Dando cumplimiento al objetivo de este proyecto y a los requisitos del cliente, se obtuvo el editor de personajes para el sistema de entrenamientos “Indicios Aduaneros”, el cual brinda las funcionalidades básicas de edición y generación de personajes. Con esto se logró optimizar el proceso de asignación de propiedades a personajes de manera aleatoria y automática que anteriormente se hacía de forma manual, permitiendo la unión de la parte lógica (propiedades) con la pieza física (mesh) de los personajes.

Finalmente, se obtuvo un módulo funcional que cumple con los requisitos del cliente y que permite al programador cargar desde un fichero XML una lista de personajes, el cual posteriormente va a ser utilizado por la aplicación final para visualizar los personajes contenidos en el archivo simultáneamente con las propiedades pertenecientes a cada uno de ellos. La veracidad de este módulo se demostró dentro del funcionamiento interno del editor en el momento en que se carga una lista de personajes desde un fichero XML.

Recomendaciones

Durante el desarrollo de este trabajo han surgido algunas ideas para lograr crear una aplición un mayor número de funcionalidades y más fácil de usar, por lo que se recomienda:

- Extensión de la herramienta añadiendo nuevas funcionalidades como la edición de texturas y animaciones.
- Incorporar soporte para más idiomas haciendo uso de Qt Linguist.
- Incorporar un manual de usuario para el fácil entendimiento de la aplicación.

Referencias Bibliográficas

1. mjbWorld.[Online][Cited: 01 15, 2010]

   http://www.euclideanspace.com/prog/mjbWorld/program/index.htm

2. mjbWorld. [Online] [Cited: 01 20, 2010.] http://freshmeat.net/projects/mjbworld

3. GeoControl2. [Online] [Cited: 01 27, 2010.] http://www.geocontrol2.com/e_index.htm

4. GeoControl2. [Online] [Cited: 01 27, 2010.] http://www.cajomi.de/GeoControl/geocontrol_news.htm

5. Misit Model 3D. [Online] [Cited: 01 27, 2010.] http://www.misfitcode.com/misfitmodel3d/

6. Unreal. [Online] [Cited: 01 28, 2010.] http://www.unrealtechnology.com/

7. Inferencia Logica Licenciatura. [Online] [Cited: 02 10, 2010.] http://www.mitecnologico.com/Main/InferenciaLogicaLicenciatura .

8. Inferencia Lógica. [Online] [Cited: 02 10, 2010.] http://www.juntadeandalucia.es/averroes/emilioprados/filosof/Logica/Reglas%20de%20inferencia.htm .

9. GAMA-Generador Automático de Modelos Animados. [Online] 02 12, 10. http://biblioteca.universia.net/ficha.do?id=44390518.

10. Morpho 3D. [Online] 02 12, 10. http://www.produccionprofesional.com/article.php?a=887.

11. Métodos Formales. [Online] 02 14, 10. http://dit.unitn.it/~lopez/publications/FMNotes.pdf.

12. **Catá, Jordi.** *Comparativa de motores gráficos para videojuegos.* 2002.

13. CEGUI. [Online] [Cited: 02 14, 10.] http://www.worldlingo.com/ma/enwiki/es/CEGUI.

14. GTK. [Online] [Cited: 02 14, 10.] http://www.gtk.org/.

Bibliografía Consultada

1. http://www.cornucopia3d.com/purchase.php?item_id=7340

2. http://eprints.ucm.es/8889/

3. http://www.ogre3d.org

4. http://descargar.traducegratis.com/s/direct3d.htm

5. www.worldlingo.com/ma/enwiki/es/Heightmap

6. http://www.masadelante.com/faqs/rss

7. http://www.mastermagazine.info/termino/6771.php

8. http://www.mastermagazine.info/termino/6771.php

9. www.kde.org .

Glosario de Abreviaturas

API: Application Programmer's Interface, (interfaces para programadores de aplicaciones).

CEDIN: Centro de Desarrollo de Informática Industrial.

C++: Lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos (POO). C++ está considerado por muchos como uno de los lenguajes más potentes, debido a que permite trabajar tanto a alto como a bajo nivel.

C#: Es un lenguaje orientado a objetos.

IDE: (Integrated Development Environment) Entorno de desarrollo integrado. Conjunto de programas que se ejecuta a partir de una única interfaz de usuario.

KDE: (K Desktop Environment), uno de los Interfaz Gráfica de Usuario) más importantes para sistemas UNIX.

PC: Son las siglas de Personal Computer (computadora personal).

RSS: RSS son las siglas de RDF Site Summary or Rich Site Summary, un formato XML para sindicar o compartir contenido en la web.

SQL: (Structured Query Language). Es un estándar en el lenguaje de acceso a bases de datos.

UML: (Unifed Modeling Language). Es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo.

XML: XML son las siglas de Extensible Markup Language una especificación/lenguaje de programación desarrollada por el W3C. XML es una versión de SGML, diseñado especialmente para los documentos de la web.

3D: Tres dimensiones, en la mayoría de los casos se representa con las letras x,y,z.

Glosario de Términos

Direct3D: DirectX es un paquete de APIs, aplicaciones que mejoran el rendimiento de juegos y complicadas aplicaciones multimedia, optimizando en gran medida la calidad del sonido y de los gráficos.

Epic Games: Es una empresa desarrolladora de videojuegos establecida en Raleigh.

HeightMaps: Es una imagen de la trama almacena valores, tales como superficie de elevación de datos, para la exhibición dentro de gráficos de computadora 3D.

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

Vector: En programación, es una estructura de almacenamiento que soporta cualquier tipo de datos.

Anexos

Anexo 1 Aplicación obtenida

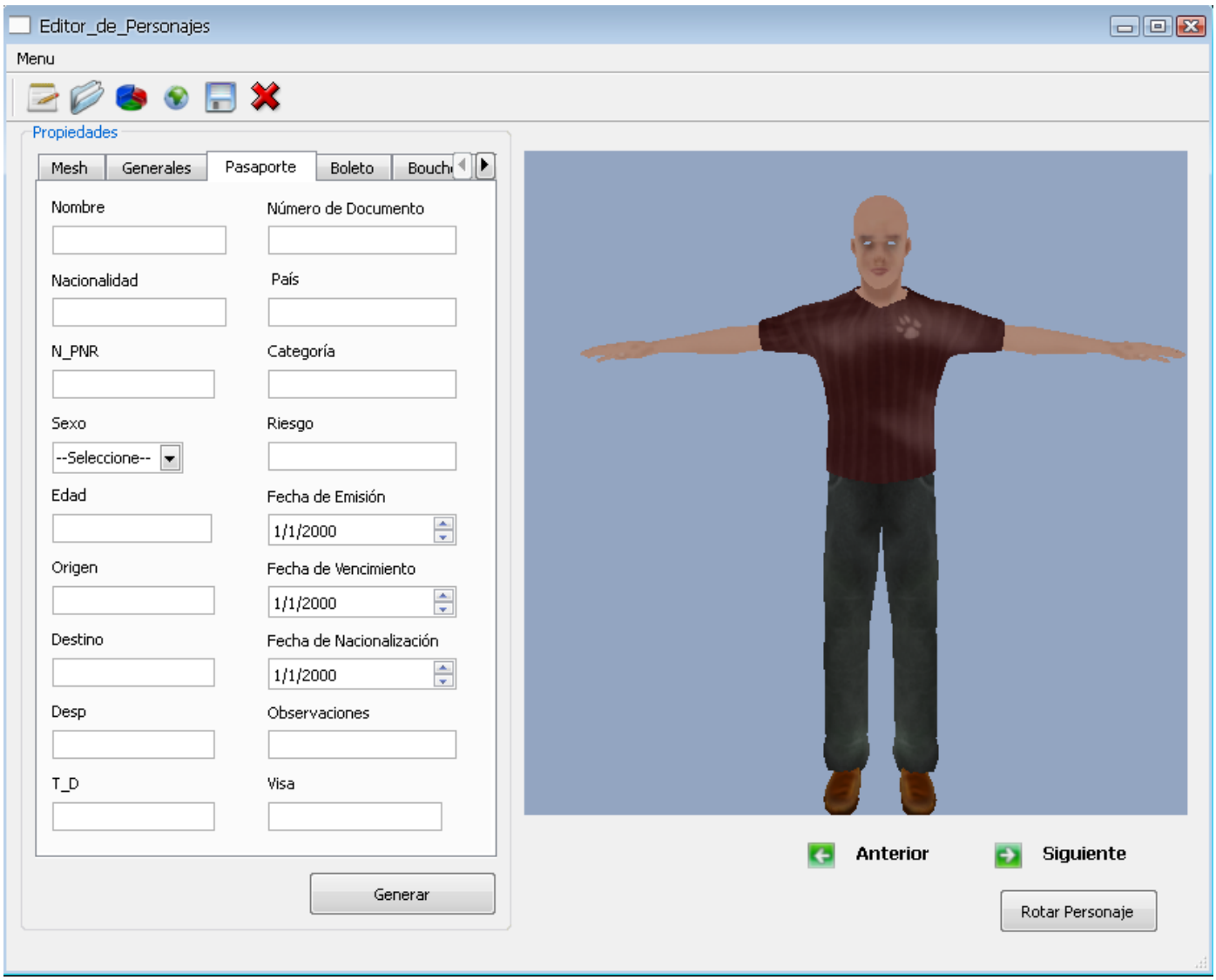

Figura 21 Aplicación obtenida.

Anexo 2 Prototipo no funcional del CU Generar Personaje

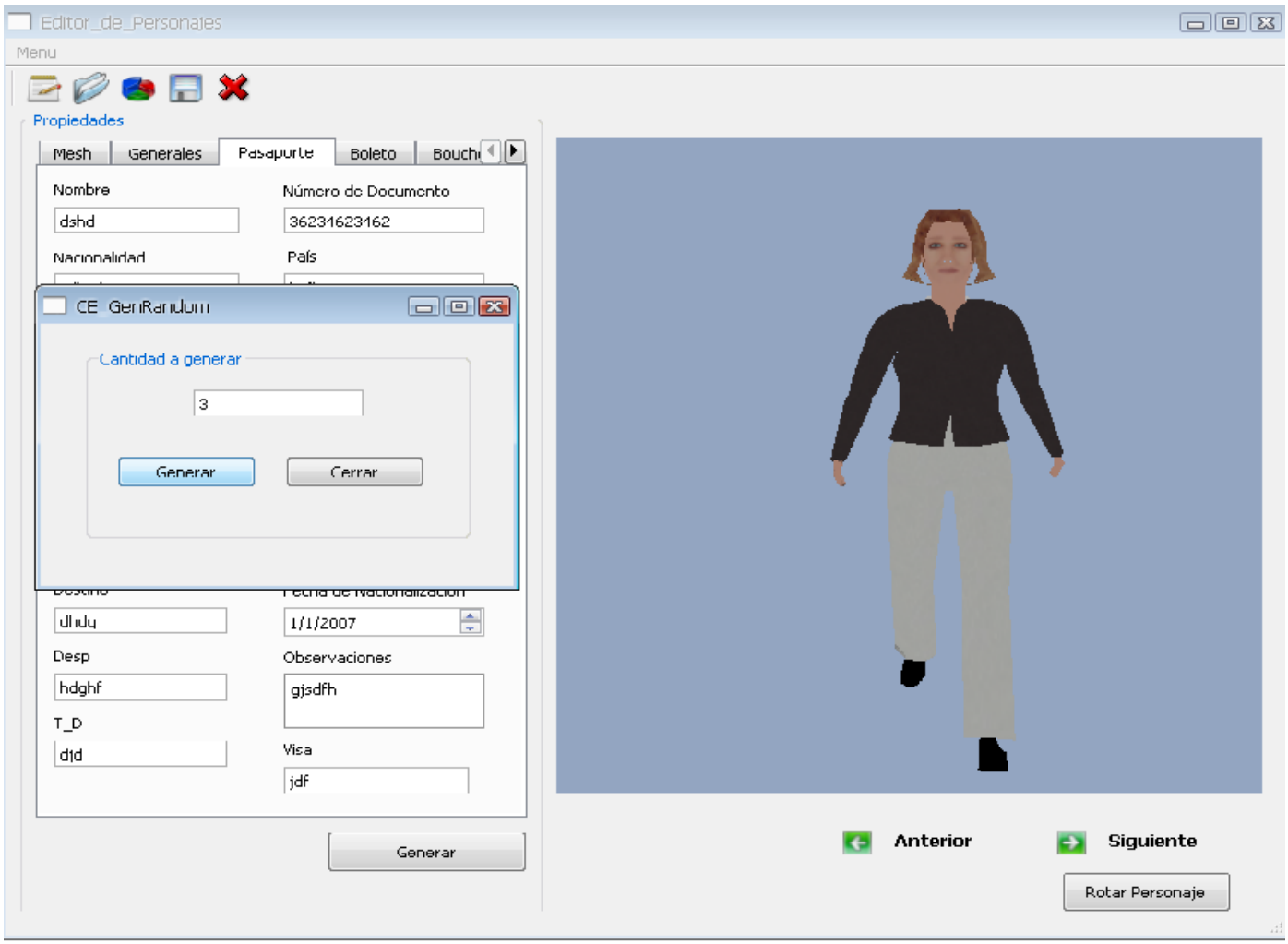

Figura 22 Prototipo no funcional del CU Generar Personaje.

Anexo 3 Prototipo no funcional del CU Generar Personajes Automáticamente

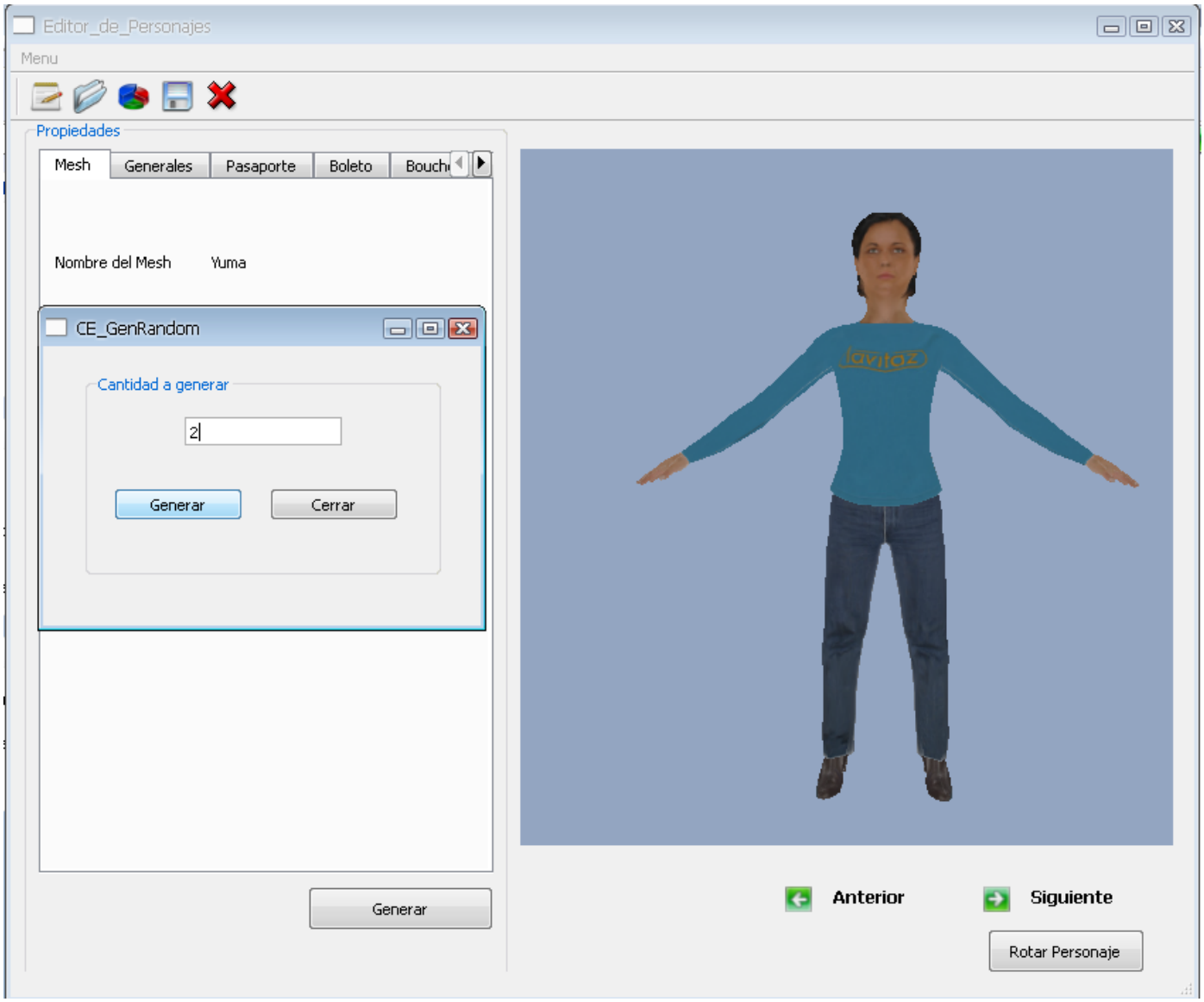

Figura 23 Prototipo no funcional del CU Generar Personajes Automáticamente.

Anexo 4 Prototipo no funcional del CU Cargar Personajes

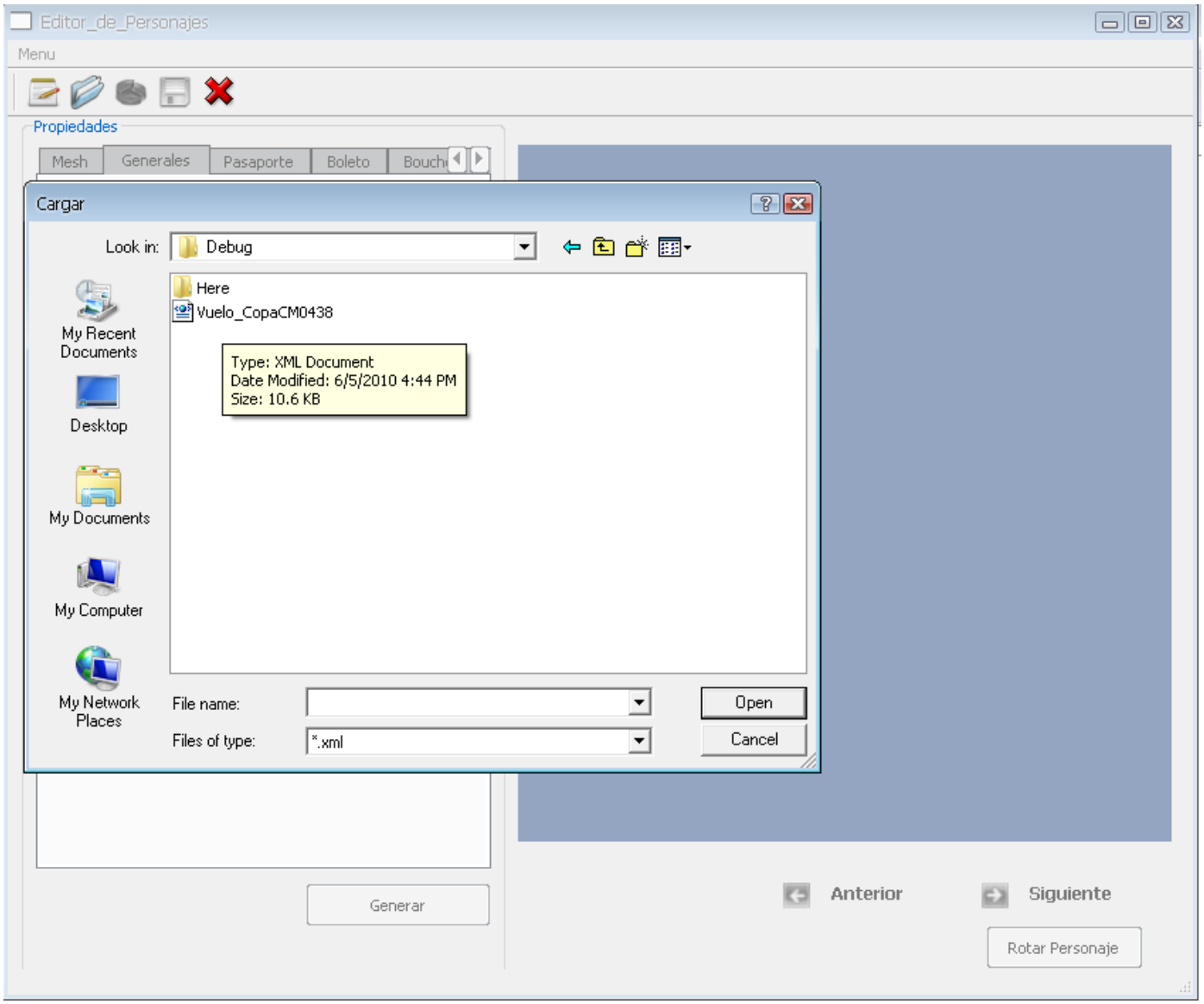

Figura 24 Prototipo no funcional del CU Cargar Personajes.

Índice de figuras

Índice de figuras

Índice de tablas