Universidad de las Ciencias Informáticas

Facultad 5

“Escenarios Virtuales”

Editor de personajes 3D

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

Tomás Obaya Espinosa y Julio Álvarez Aguilera
Autores

Ing. Alexis Echemendía González
Tutor

Ciudad de La Habana, Junio 2010

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

DECLARACION DE AUTORIA

Declaramos que somos los únicos autores de este trabajo y autorizamos al Proyecto Escenarios Virtuales de la Línea de Diseño de la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Julio Héctor Álvarez Aguilera
Autor

Tomás Javier Obaya Espinosa
Autor

_____

Ing. Alexis Echemendía González
Tutor

DATOS DE CONTACTO

Nombre y Apellidos: Ing. Alexis Echemendía González.

Edad: 25.

E-Mail: aechemendia@uci.cu.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Titulo: Ingeniero Informático.

Categoría Docente: Profesor Instructor.

Ocupación Actual: Jefe de Línea de *Imágenes y Videos basados en Rendering*.

A la Universidad de las Ciencias Informáticas por forjarnos como profesionales durante estos cinco provechosos años, de conocimientos y experiencias inolvidables y que sin ella no hubiese sido posible la realización de este sueño.

A nuestros profesores de la UCI que con su dedicación y esfuerzo nos han trasmitido valiosas enseñanzas, logrando con ello despertar un gran interés por nuestra carrera durante estos 5 años.

A todos los profesores y personas que por cuestiones de la vida ya no se encuentran entre nosotros pero que influyeron en nuestra formación y educación por más de 15 años de estudio.

A nuestro tutor Alexis por sus orientaciones, paciencia y preocupación. Además por ayudarnos en la construcción y elaboración de la tesis.

A nuestras familias que siempre nos han brindado su confianza, por apoyarnos en todo momento y confiar ciegamente en nosotros.

A nuestros compañeros y amigos que nos ayudaron y nos dieron aliento constante.

A todos aquellos que de una forma u otra estuvieron presentes en la realización de este trabajo.

A todas las personas que han creído y que creen en nosotros.

¡Gracias a Todos!

Agradecimientos

A mi mamá por su forma especial de quererme, por estar siempre pendiente de lo que necesitara y por darme fuerzas para levantarme en los momentos más difíciles.

A mi papá por su preocupación constante, por ser mi brazo derecho y por ser un guía para mí en todo momento.

A mis abuelos especialmente a mis abueditos Xiomara y Héctor por ser mis otros padres, por esperar siempre lo mejor de mí, por apoyarme en todas mis decisiones y por guiarme siempre por el camino correcto.

A mi tía Carmen y a mi prima Lisandra, por su constancia, por su sabiduría, su ejemplo por su ayuda incondicional. Gracias por elevar el concepto de familia a su más alta expresión.

A mi familia de la UCI: Tomy, Alvaro, Fonse, Leyner, Chino, Ernesto, Gualby, yOYck, Hugo, Bencomo y Dairon.

A todos mis amigos, en especial a Tico, Nelson, Elisa, Lilisbeth, Imi, Elesky, Maide, Yuya, Miguel, Luisi, David, Nesti, Yordani, Fernand, Anita, Ari, Ity, Arleis, Eduardito, Nayi, Aleyda, Claudia, Maray, Tala, Yosuan, Freddy, Naty, Liannet y muchos más que de una manera u otra permitieron que este sueño se haya materializado.

A Diana, Ismary, Yami, May, Loyda y Betty por su aporte y colaboración en la realización de este trabajo.

A mis compañeros de grupo por su preocupación y apoyo en estos años.

A mi Dios, Señor y Salvador Jesús el Cristo y a mi señora La Virgen de la Caridad del Cobre que han sido los principales causantes de que hasta aquí haya llegado.

Julio Héctor

Agradecimientos

Quiero agradecer a mi máma por su infinita paciencia, por la confianza depositada en mí y por su manera tan especial de quererme…

A mi difunto padre porque sé que estaría orgulloso de mí y de ver en lo que me he convertido…

A mi hermano por siempre estar sobre mí como un látigo…

A mi abuela Nini, ojalá estuvieras aquí con nosotros…

A mi tía linda Tere y a Wicho, a Alejandro, a Jose, a mis primos, los quiero mucho…

A la otra familia: Horty, Nury, Freddy, Blas, Isa…

A mis amigos de la UCI: el Fonse, Alvaro, Julito, Pepe, Chino, Walby, Leyner, Luis Palma, Yendris,no alcanzan las páginas…
A mis amigos del barrio: Adrián,Reinaldo, Jorge Luis, Manolín, Favián, son muchos…

Que me perdonen los que faltan en la lista, para ustedes también son estos agradecimientos…

Y a todos los que de una forma u otra contribuyeron directa e indirectamente a la culminación de este enorme derroche de esfuerzo y sacrificio, a todos, de veras, a todos GRACIAS!!!

Tomy

A mi hermosa familia, en especial a mis padres y a mis abuelos queridos que no solo se merecen esto, sino mucho más, a ellos les doy gracias por educarme, por amarme, por adorarme y acompañarme en cada segundo, en cada minuto, en cada hora del día, sin más su hijo que los quiere con todo su corazón.

Julio Héctor

Dedico el fruto de este inmenso y provechoso esfuerzo a mi querida madre, la luz de mi vida, a mi adorado hermano, mi segundo educador, a mi hermosa abuela, Nini, y a mi difunto padre…

Tomy

RESUMEN

El creciente desarrollo de las tecnologías se ha revertido directamente sobre la evolución de la Realidad Virtual, esta significativa mejora, principalmente en el campo del Diseño 3D le ha permitido a los desarrolladores alcanzar un mayor grado de realismo. El modelado de personajes (humanos, animales, entre otros) es uno de los elementos fundamentales en el desarrollo de estos avances tecnológicos. En la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) se encuentra la Línea de Diseño (Imágenes y videos basados en rendering) y dentro de ésta el proyecto productivo “Escenarios 3D”, donde no existe una herramienta propietaria y fácil de usar que edite estos modelos 3D. Por tanto, surge la idea de implementar una aplicación gráfica que permita la realización del proceso de edición de estos modelos 3D.

Para el desarrollo de esta herramienta se utiliza RUP como metodología de desarrollo de software, UML como lenguaje de modelado y Visual Paradigm como herramienta case. Además de C++ como lenguaje de programación para la implementación y Microsoft Visual Studio Express Edition como entorno de desarrollo integrado para la programación de los casos de usos. Finalmente, se usa Ogre 3D como motor gráfico y Qt como biblioteca de interfaz gráfica.

Luego de analizar las características de los diferentes editores de personajes 3D más utilizados en la actualidad y los variados formatos de ficheros que más se emplean para almacenar la información referente a estos tipos de modelos, se obtuvo una herramienta capaz de visualizar modelos 3D con posibilidades de edición, además de poder variar el tamaño de dichos modelos, cambiar sus texturas y una propuesta para cambiar los grupos de vértices que se toman como partes editables de un caracter.

Palabras Claves: Herramienta, Realidad Virtual, Personaje, Formato, Fichero, Vértice, Diseño 3D, Modelo.

TABLA DE CONTENIDO

Índice de Figuras

INTRODUCCIÓN

El desarrollo de las tecnologías de la informática y las comunicaciones, gracias al afán de los seres humanos por enriquecer sus conocimientos y mejorar su calidad de vida, ha devenido en un imparable proceso del cual se forma parte directa e indirectamente. Dentro del plano de la informática ha surgido una rama muy importante y aplicable en amplios sectores del quehacer diario del hombre: se trata del campo de la Realidad Virtual (RV), utilizada en varios ámbitos, dentro de los cuales se encuentra la simulación de disímiles procesos y situaciones, que pueden ir desde la recreación virtual en el diseño de un inmenso rascacielos hasta la fabricación de un automóvil. Las aplicaciones de la RV se han convertido en sistemas muy comunes y populares en nuestro tiempo; estos sistemas permiten representar situaciones de nuestro mundo con gran realismo por los niveles de detalles, de inmersión e interactividad que se han obtenido. Los aportes de la RV se hacen muy notables en la educación con el empleo de las multimedias; en las investigaciones científicas con la visualización de fenómenos complejos; en la industria del entretenimiento, donde resaltan el cine y los videos-juegos, también en la simulación, donde se conocen en la actualidad numerosos tipos de simuladores, desde la conducción de autos hasta los simuladores más complejos de aviación **[1]**. Ahorrándose de esta manera cuantiosas sumas de recursos monetarios y humanos.

Tanto en juegos como en simuladores se hace muy necesaria la presencia de personajes; fundamentalmente en los entornos urbanos y juegos de deporte o combate, donde presentan un comportamiento que los distingue unos de otros y está definida su interactividad. Los personajes poseen gran importancia porque pueden realizar acciones que para un humano resultarían muy difíciles o imposibles. De esta forma aumentan los aportes de las aplicaciones de la RV, contribuyendo a incrementar los beneficios económicos y a reducir el riesgo para el hombre en actividades tales como entrenamientos, y demás.

Cuba no está exenta de estos avances tecnológicos y a pesar de las carencias económicas y del injusto bloqueo por parte de los Estados Unidos de América, no se ha mantenido al margen de esta situación y ha adoptado una postura de acción y colaboración conjunta con países hermanos del ALBA dentro de sus posibilidades.

La Universidad de las Ciencias Informáticas (UCI) constituye un elemento fundamental para apoyar dicha postura nacional y por lo tanto dentro de las especificidades de la informática que se estudian en cada una de las facultades de la universidad, la Facultad 5 cuenta con diferentes Líneas y específicamente dentro de éstas con una que es la Línea de Diseño (Imágenes y videos basados en rendering), en la cual se encuentra el proyecto “Escenarios 3D” que se encarga de crear entornos 3D, personajes, texturas y animaciones. En estos momentos dicho proyecto no cuenta con un software propio capaz de editar personajes 3D de una forma fácil y rápida, por lo que existe la necesidad de poseer una herramienta, la cual sea desarrollada en la universidad, facilitando y agilizando el trabajo a los diseñadores de modelos 3D, en aras de una mejor edición de estos modelos, además de que pueda conocerse el código fuente de dicha herramienta y pueda llegar a convertirse en un módulo de una futura aplicación de algún software de diseño 3D.

Dada esta **situación problémica** y ante la escasez de dicho recurso surge el presente trabajo de diploma para dar respuesta a esta problemática. Por tanto se plantea la siguiente interrogante como **problema científico:** ¿Cómo agilizar el trabajo de los diseñadores en la edición de personajes 3D?

Por consiguiente se propone como **objeto de estudio** las herramientas que se utilizan para la edición de personajes 3D.

El **campo de acción** se enmarca en el proceso de desarrollo de un Editor de Personajes 3D, dentro del cual se tiene como **objetivo general** implementar un Editor de personajes 3D.

Se trazan entonces un grupo de **tareas de la investigación** que permitirán cumplir con los objetivos, resumiéndose en las siguientes:

1. Caracterización de las herramientas de diseño 3D más utilizadas en la actualidad.
2. Selección del formato de fichero 3D más apropiado para la herramienta.
3. Revisión de la documentación existente sobre las características de los diferentes editores de personajes 3D en la actualidad.
4. Selección del motor gráfico más apropiado para la visualización de elementos en un Editor de personajes 3D.
5. Elaboración del análisis y diseño de una herramienta de edición de personajes 3D.

Dentro de los **métodos científicos** utilizados se encuentran los siguientes:

Teóricos:

Analítico sintético: Se analizan las teorías, documentos, entre otros, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Inductivo deductivo: Formas de razonamiento que permiten llegar a un grupo de conocimientos generalizadores, tanto desde el análisis de lo particular hasta lo general, es decir, desde el análisis de los elementos generalizadores a uno de menor nivel de generalización.

Empíricos:

Observación: Registro visual de lo que ocurre en una situación real, en un fenómeno determinado, clasificando y consignando los hechos y acontecimientos pertinentes de acuerdo con algún esquema previsto.

Organización del documento

El presente trabajo está estructurado de la siguiente manera: Resumen, Introducción, tres capítulos de Contenido, Conclusiones, Recomendaciones, Bibliografía Referenciada y Consultada, un Glosario de Términos, así como un Índice de Tablas y de Figuras.

Capítulo 1_ Fundamentación Teórica: Se hace un análisis bibliográfico sobre algunas herramientas para desarrollar editores de personajes 3D, así como la justificación de las herramientas y metodologías utilizadas.

Capítulo 2_ Características del Sistema: Muestra los principales rasgos de la herramienta que se propone desarrollar, la cual como lo indica su nombre presenta las características que tendrá el nuevo sistema. También se procede al levantamiento de requisitos del sistema, haciendo un análisis más exhaustivo del objeto de estudio que se plantea, además de que se definen los casos de uso del sistema y la descripción de cada uno de ellos.

Capítulo 3_ Diseño e Implementación del Sistema: Se realiza el diseño y la implementación del sistema, quedando reflejado en diagramas de clases del diseño, así como en los diagramas secuencia de

los casos de usos recogidos en el capítulo anterior. Se muestran además los diagramas de despliegue y de componentes. También se exponen los resultados obtenidos con la puesta en marcha y uso de la herramienta.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

Hoy en día los gráficos por computadora constituyen el elemento clave en la visualización de todo tipo de contenidos: contenidos tradicionalmente 2D (planos, dibujos, publicaciones, presentaciones) y muy especialmente contenidos 3D (modelos de piezas industriales, datos científicos, efectos especiales, así como representaciones realistas de entornos virtuales).

En este capítulo se hará referencia a las características de los ficheros 3D más usados, analizando las ventajas y desventajas que ofrecen, además del análisis detallado de las diferentes herramientas de diseño para exportar el formato de fichero 3D más apropiado, también se hará un estudio profundo de los diferentes editores de personajes 3D existentes en la actualidad, se hace referencia a las diferentes características que presentan los disímiles motores de render gráficos y las principales bibliotecas gráficas, las cuales son empleadas para la visualización de los diferentes archivos 3D. También se abordan las metodologías, las herramientas de desarrollo y los lenguajes informáticos más utilizados para la edición 3D.

1.1 Herramientas de diseño utilizadas para la creación de personajes 3D.

Para la modelación y animación de elementos virtuales se utilizan un gran número de herramientas de diseños 3D. Estas herramientas informáticas constituyen un componente fundamental para la creación de los personajes en 3D.

Entre las herramientas de diseño más conocidas tenemos a:

- Autodesk 3Ds Max
- Maya
- Blender

1.1.1 Autodesk 3Ds Max

Autodesk 3Ds Max (anteriormente 3D Studio Max) es un programa de creación de gráficos y animación 3D desarrollado por Autodesk (anteriormente Discreet). Fue desarrollado originalmente por Kinetix como sucesor para sistemas operativos Win32 del 3D Studio creado para DOS. Más tarde esta compañía fue fusionada con la última adquisición de Autodesk, Discreet Logic. 3Ds Max es uno de los programas de animación 3D más utilizados. Dispone de una sólida capacidad de edición, una omnipresente arquitectura de plugins y una larga tradición en plataformas Microsoft Windows. Es utilizado en mayor medida por los desarrolladores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura **[2]**. Es uno de los programas más sencillos para iniciarse en el mundo de la animación 3D. Su arquitectura abierta, su baja curva de aprendizaje y sus potentísimas herramientas lo convierten en uno de los programas líderes del diseño y la animación 3D en infinidad de ámbitos, como: arquitectura, publicidad, televisión, video, cine, artes escénicas, y desarrollo de juegos. El enorme potencial y las infinitas prestaciones de 3Ds Max lo convierten en un aliado bastante eficaz, para la realización de todo tipo de diseños y animaciones tridimensionales, ya que permite al usuario una fácil visualización y una sencilla representación de los modelos y animaciones; utilizando al mismo tiempo cámaras, vistas o visores. Permite además exportar y salvar a varios formatos, incluso diferentes de los que trae por defecto la herramienta.

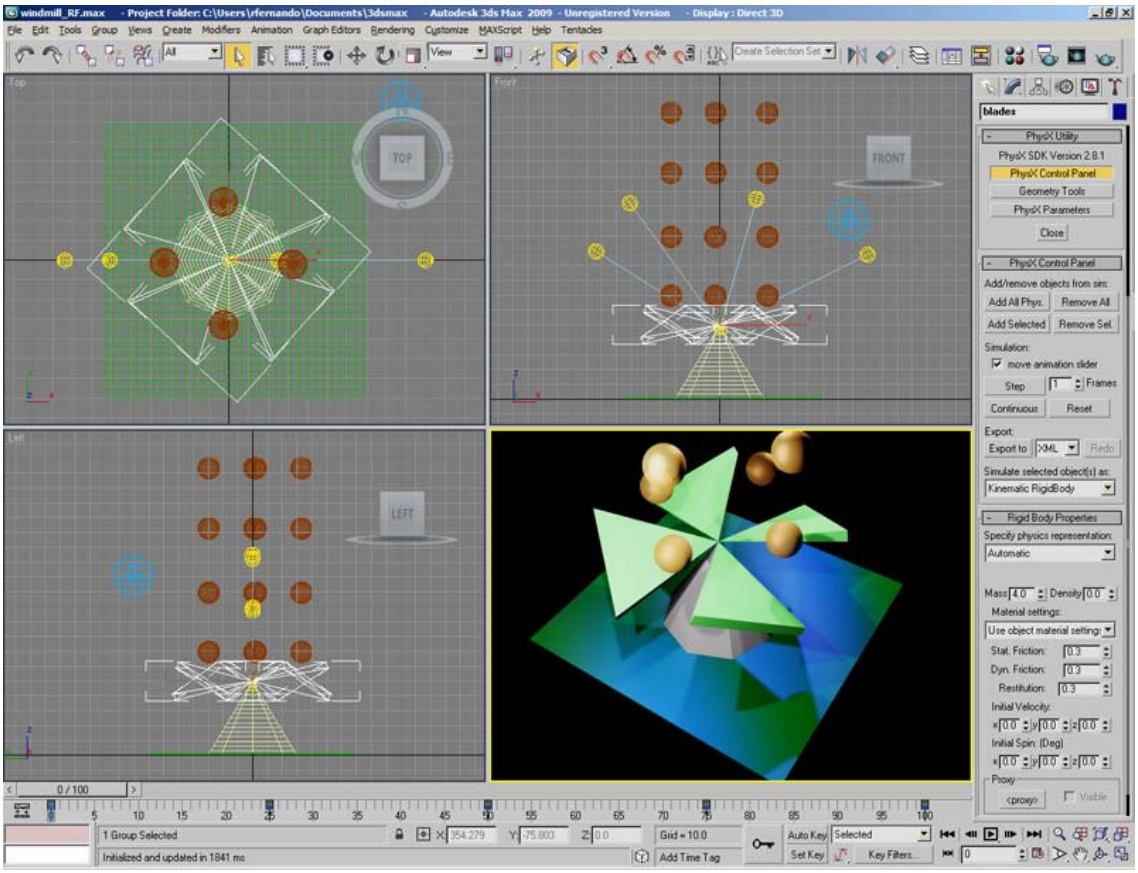


Figura 1: *Herramienta de diseño 3D – Autodesk 3Ds Max.*

1.1.2 Blender

Blender es un programa informático multiplataforma dedicado especialmente al modelado, animación y creación de gráficos tridimensionales **[3]**. El programa fue inicialmente distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, Mac OS X, GNU/Linux, Solaris, FreeBSD e IRIX. Tiene una muy peculiar interfaz gráfica de usuario que se critica como poco intuitiva, pues no se basa en el sistema clásico de ventanas; pero tiene a su vez ventajas importantes sobre éstas, como la configuración personalizada de la distribución de los menús y vistas de cámara. Originalmente, el programa fue desarrollado como una aplicación propia por el estudio de animación holandés NeoGeo; el

principal autor, Ton Roosendaal, fundó la empresa "Not a Number Technologies" (NaN) en junio de 1998 para desarrollar y distribuir el programa. La compañía llegó a la bancarrota en el 2002 y los acreedores acordaron ofrecer Blender como un producto de código abierto y gratuito bajo los términos de la GNU GPL a cambio de 100.000 €. El 18 de julio de 2003, Roosendaal creó sin ánimo de lucro la Fundación Blender para recoger donaciones; el 7 de septiembre se anuncia la recaudación como exitosa (participaron también ex empleados de NaN) y el código fuente se hizo público el 13 de octubre. A pesar de ser una herramienta relativamente nueva, ha gozado de la aceptación de muchos animadores independientes. En la industria de la generación de gráficos avanza como un proyecto prometedor. Existen proyectos actuales que han empezado a usarlo profesionalmente.

Características:

- ✓ Multiplataforma, libre, gratuito y con un tamaño de origen realmente pequeño comparado con otros paquetes de 3D, dependiendo del sistema operativo en el que se ejecuta.
- ✓ Capacidad para una gran variedad de primitivas geométricas, incluyendo curvas, mallas poligonales, vacíos.
- ✓ Junto a las herramientas de animación se incluyen cinemática inversa, deformaciones por armadura o cuadrícula, vértices de carga y partículas estáticas y dinámicas.
- ✓ Edición de audio y sincronización de video.
- ✓ Características interactivas para juegos como detección de colisiones, recreaciones dinámicas y lógica.
- ✓ Posibilidades de renderizado interno versátil e integración externa con potentes trazadores de rayos o "raytracer" libres como kerkythea, YafRay o Yafrid.
- ✓ Lenguaje Python para automatizar o controlar varias tareas.
- ✓ Blender acepta formatos gráficos como TGA, JPG, Iris, SGI, o TIFF. También puede leer ficheros Inventor.
- ✓ Motor de juegos 3D integrado, con un sistema de ladrillos lógicos. Para más control se usa programación en lenguaje Python.

- ✓ Simulaciones dinámicas para softbodies, partículas y fluidos.
- ✓ Modificadores apilables, para la aplicación de transformación no destructiva sobre mallas.
- ✓ Sistema de partículas estáticas para simular cabellos y pelajes, al que se han agregado nuevas propiedades entre las opciones de shaders para lograr texturas realistas.

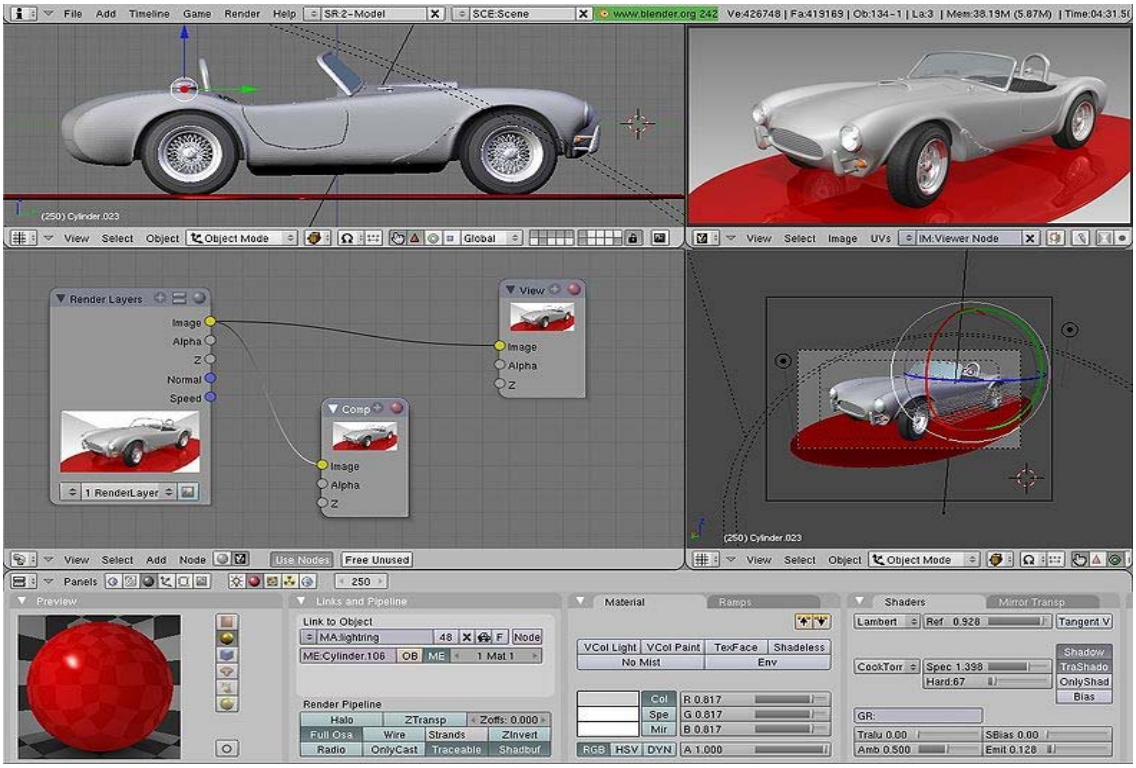


Figura 2: *Herramienta de diseño 3D – Blender.*

1.2 Características de los diferentes Editores de personajes 3D.

En el mundo existen diversos editores de personajes 3D, muchos de éstos satisfacen diversas necesidades de los usuarios, por ejemplo:

1.2.1 Estática 3D Editor

Diseño de objetos, escenas, personajes y animaciones en 3D para VB Editor. Modelador animador 3D que permite diseñar objetos, personajes y escenas en 3D que se pueden utilizar dentro de VB en un Form o Picture. Los objetos se pueden mover en tiempo real. Es útil para hacer juegos en Visual Basic o complementar programas y juegos con escenas y objetos 3D **[4]**. Utiliza texturas de 24 bits en alta densidad. Se pueden aplicar texturas de alta resolución, con transparencia o máscara.

Figura 3: *Editores de personajes 3D – Estática 3D Editor.*

1.2.2 MakeHuman

MakeHuman es una fuente abierta (por lo que es completamente gratuito), innovadora y profesional del software para el modelado de 3D. Incluye una nueva interfaz gráfica muy intuitiva y una malla de alta calidad, optimizados para trabajar en modo de subdivisión de la superficie (por ejemplo, Zbrush). Usando MakeHuman, un personaje puede ser modelado en menos de 2 minutos; MakeHuman es liberado bajo una licencia Open Source (GPL3.0) y está disponible para Windows, Mac OS X y Linux.

La versión 0.9.1 de MakeHuman se publicó en diciembre del 2007. Los trabajos de desarrollo se centran actualmente en la versión 1.0, basado en una nueva interfaz gráfica de usuario y una malla de cuarta generación. MakeHuman incorpora una serie de plugins para exportar un modelo. Esta versión también incorpora cambios considerables en la base de código, que se utiliza una pequeña y eficiente aplicación principal escrita en C, con la mayoría de las funcionalidades para el usuario. Dicha aplicación está ejecutada en Python. Debido a que Python es un lenguaje de scripts, ésto significa que una amplia gama de scripts, plugins y empresas de servicios públicos se pueden añadir sin necesidad de reconstruir la aplicación.

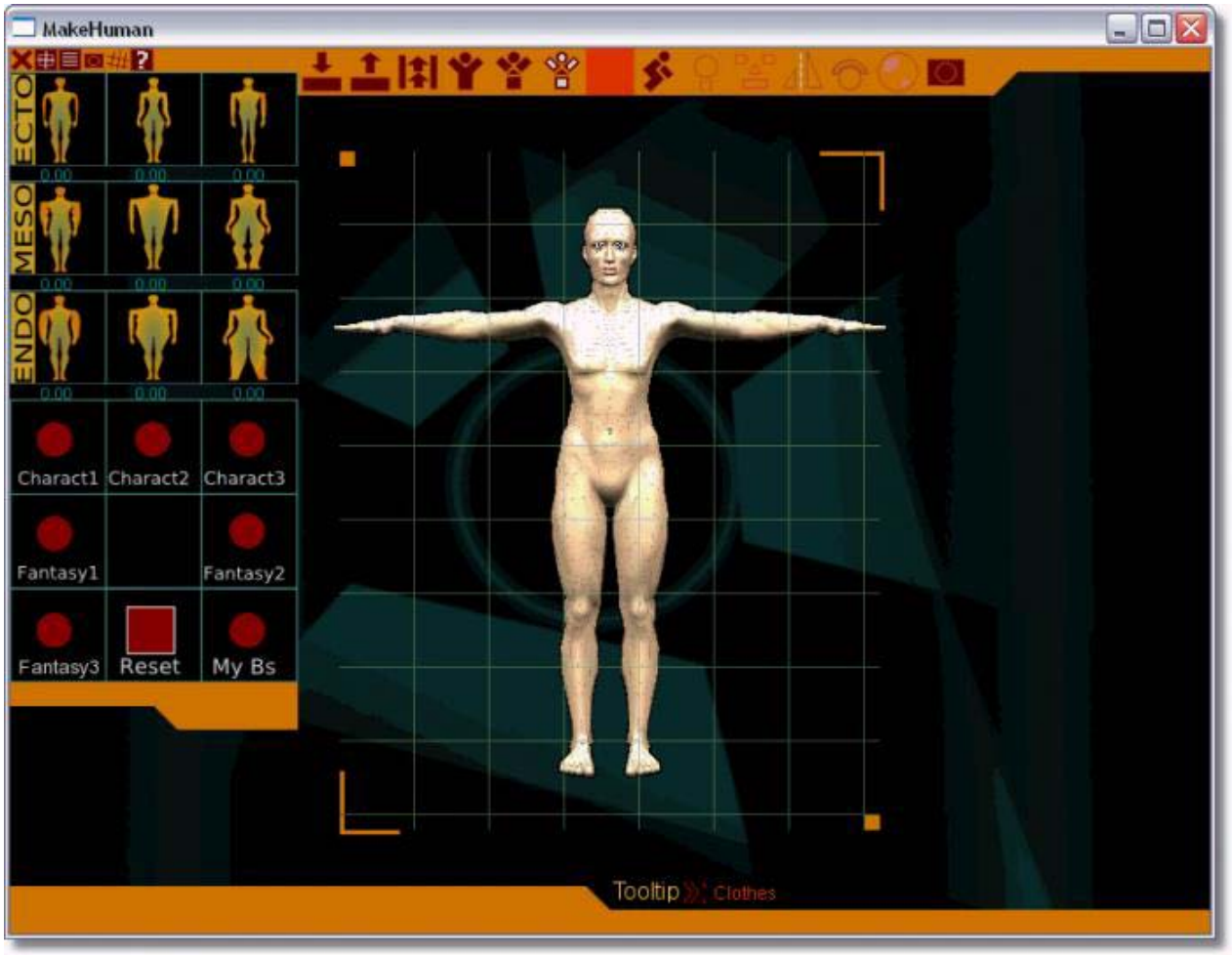


Figura 4: *Editores de personajes 3D – MakeHuman.*

Todas las cifras de MakeHuman se basan en una malla de luz única y profesional, altamente optimizada **[5]**. La modelación de la malla se realiza por deformación de ella misma en lugar de alterar su topología. La malla ha sido modificada a través de una serie de iteraciones para mejorar la estructura de manera que las deformaciones pueden aplicarse de forma realista, mientras que el mantenimiento de un bajo número de polígonos, reduce al mínimo los gastos generales de procesamiento. La malla apoya la subdivisión para permitir una mayor densidad. Un número considerable de los objetivos de la deformación de malla, han sido creados por los artistas para ofrecerle un gran número de puntos de partida para un modelo determinado étnico, de género, la edad y las cifras de masa corporal de su propio diseño.

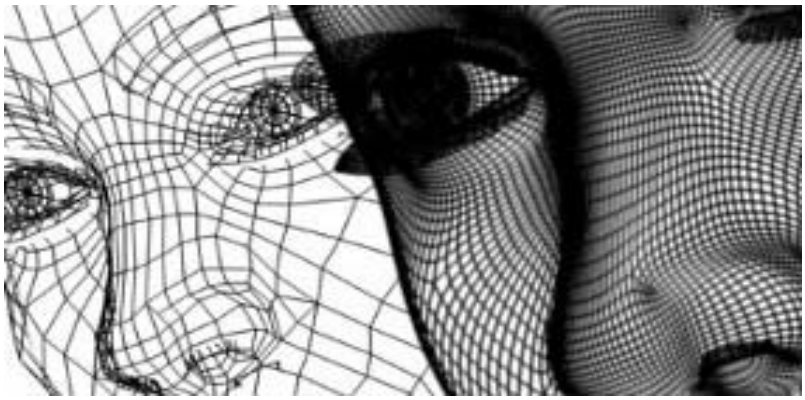


Figura 5: *Editores de personajes 3D – Malla de MakeHuman.*

1.2.3 POSER

POSER es una solución para crear arte y animación en 3D utilizando caracteres. Incluye una interfaz de usuario mejorada para maximizar su espacio de trabajo, mientras que proporciona un mejor flujo de trabajo, una nueva búsqueda habilitada para que pueda encontrar, organizar y usar su contenido más fácil, así como una herramienta parámetro dependiente que le permite enseñar a los objetos en la escena e interactuar unos con otros [6].

Figura 6: *Editores de personajes 3D – Poser.*

Funcionalidades del sistema:

- Generar nuevos personajes a partir de sus fotografías faciales.
- Añadir el pelo y la ropa.
- Generar automáticamente para caminar o correr las animaciones y los personajes.
- Importar archivos de captura de movimiento para el realismo más animado.

Para utilizar correctamente la información de cada personaje 3D creado con los editores y herramientas descritos anteriormente, fue necesario la investigación de los diferentes formatos de ficheros 3D que son exportados por los software de diseño 3D, con el propósito de identificar cuál es el más apropiado y necesario para poder ser importado con la aplicación.

1.3 Características que debe tener un formato de fichero gráfico 3D.

Un fichero gráfico 3D debe ser un contenedor de información de una escena, ya sea de un único objeto o de un mundo virtual complejo, que debe contar con todos los atributos que permitan conocer la estructura de la malla, como son los vértices y caras, en el caso más básico, suelen tener un conjunto de características que brindan un mayor nivel de detalle a las escenas, entre los cuales se pueden mencionar los materiales, los colores de vértices, las normales de las caras, los estados de render, entre otros. Otro aspecto importante es el formato de almacenamiento y la organización de los atributos en el fichero, este último influye en la complejidad, tiempo de carga y facilidad de entendimiento del mismo.

Algunos de los formatos 3D más difundidos en la comunidad de Realidad Virtual para almacenar estas informaciones son:

- 3DS, OBJ y MESH

1.3.1 3DS

El formato 3DS es el nativo de la aplicación Autodesk 3Ds Max de modelado y animación. Aunque superado en muchos aspectos, el formato 3DS todavía se utiliza bastante y ofrece una manera directa de exportar modelos de SketchUp sencillos a una amplia gama de programas para el diseño de modelos 3D, dado que el formato 3DS respeta la asignación de materiales y texturas y la posición de la cámara.

Estructura:

Todos los ficheros están conformados por bloques y casi todos presentan una cabecera que indica donde localizar estos bloques y en qué orden se encuentran, pero el fichero 3Ds no muestra esta característica, el formato 3Ds puede tener estos bloques en cualquier orden y no presenta una cabecera que especifique donde encontrarlos **[7]**. El fichero 3Ds no sólo está dividido en pequeños bloques, sino que cada bloque está dividido en sub-bloques y lo mejor de todo es que no tiene que existir un orden en el fichero, por ello es que a simple vista es muy difícil entender su estructura.

Ventajas

- Es un formato que optimiza muy bien el tamaño de los ficheros.

- Exporta escenas completas, es decir que un mismo fichero puede contener varias mallas.

Desventajas

- La estructura que muestra es compleja y difícil de cargar.
- El formato de archivo 3DS no permite el uso de capas.
- 3DS no puede guardar nombres de archivos que sobrepasen el límite de caracteres 8.3 de DOS.
- El formato 3DS no permite almacenar una cámara ortográfica.
- Todas las mallas deben ser triángulos.
- El número de vértices y polígonos por mallas se limitan a 65536.
- Objeto, la luz y los nombres de la cámara se limitan a 10 caracteres.
- Los nombres de los materiales se limitan a 16 caracteres.

1.3.2 Wavefront OBJ

El formato de archivo obj es un formato de archivo de objetos 3D estándar creado para el uso con Wavefront's Advanced Visualizar. Es un formato de datos simple que representa geometría 3D, la posición de cada vértice, la posición de cada coordenada de textura de vértice, las normales y las caras que hacen que cada polígono sea definido como una lista de vértices **[8]**.

Es genial para objetos estáticos. Contiene geometrías poligonales que presentan, puntos, aristas y caras, para definir un objeto o geometrías de objetos de forma libre que contienen curvas y superficies, aunque en lo que más se emplea es en objetos poligonales. No necesita ningún tipo de encabezado, aunque es usual poner un comentario al inicio del fichero. Los archivos .obj tienen la capacidad de referenciar vértices, normales; ya sea por su absoluta posición de la lista o relativamente usando índices negativos y conteos regresivos.

Ventajas

- Fácil manejo a la hora de cargar el fichero, ya que presenta una estructura sencilla permitiendo guardar toda la información en listas de datos.

- Un fichero puede contener muchos objetos y tratar a cada uno por separado.

Desventajas

- Su uso se centra más en el trabajo con objetos estáticos, ya que no exporta características importantes tales como la matriz de transformación, el bounding box, los colores de vértice, entre otros.

1.3.3 MESH

Es un archivo informático que contiene información de cualquier tipo, codificada en forma binaria para el propósito de almacenamiento y procesamiento en ordenadores. Los archivos .mesh son el formato de archivo que usa OGRE (el motor de gráficos) para representar cualquier objeto 3D en pantalla.

¿Qué es un mesh?

Un mesh es una colección de triángulos que forman las caras de una superficie con una textura (dibujo) aplicado sobre dichas caras. Cualquier elemento puede ser representado por un archivo .mesh, aunque algunos deberían representarse con otros métodos. Se usan Meshes en Rigs of Rods para simular objetos estáticos (carreteras, edificios, señales de tráfico, entre otros) como los modelos 3D que se pegan al chasis de un vehículo, como los tanques de gasolina y demás.

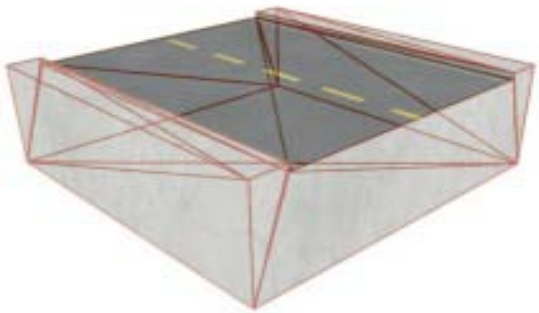


Figura 7: *Modelo 3D – archivo .mesh.*

Muchos formatos binarios contienen partes que pueden ser interpretados como texto. Un archivo binario que sólo contiene información de tipo textual sin información sobre el formato del mismo, se dice que es un archivo de texto plano. Habitualmente 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, lo cual implica que los dígitos binarios (bits) se agrupen de ocho en ocho comúnmente. Los archivos binarios que 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, versión comprimida 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.

Ventajas:

- Puede almacenar cualquier tipo de información.

Desventajas:

- La información que almacena es codificada en forma binaria, para el propósito de almacenamiento y procesamiento en ordenadores.

1.4 Características de los diferentes Motores Gráficos.

Un Motor Gráfico (Graphic Engine, en inglés) es el componente principal de software de un videojuego o de otra aplicación interactiva, que se ejecute en tiempo real. Su uso simplifica el desarrollo de la aplicación y a menudo permite que el juego pueda correr en múltiples plataformas, tales como Consolas de videojuegos y Sistemas Operativos de Mac OS, GNU/Linux y Microsoft Windows. La abstracción del

hardware es una de las principales ventajas que posee el Motor. Un Motor Gráfico ofrece un conjunto de herramientas de desarrollo, además de componentes de software reutilizables, agrupados en subsistemas que presentan alta cohesión en relación a sus comportamientos **[10]**, los subsistemas más comunes son:

- Procesamiento de Entradas.
- Gráficos.
- Animación.
- Audio.
- Comportamiento e Inteligencia Artificial.
- Conectividad / Red.

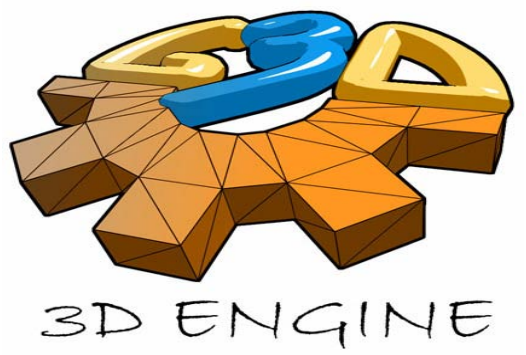
Cada subsistema es un componente por derecho propio, un componente con una estructura particular e independiente, formado por otros subsistemas más especializados.

Los Motores Gráficos son productos altamente complejos y especializados, capaces de ahorrar tiempo y dinero a través del fomento de la reutilización enfocada a diferentes aspectos del videojuego. Tener un Motor Gráfico ayuda a ser productivo, a crear juegos sofisticados con menos esfuerzo y a recuperar y aplicar las mejores prácticas que los expertos de la industria de producción de videojuegos han acumulado a lo largo de los años.

Algunos de los Motores Gráficos más conocidos son:

- Crystal Space.
- Ogre3D.
- G3D Engine.
- The Nebula Device 2.
- Blender Game Engine.

1.4.1 G3D Engine (Graphics Three Dimensional)



Graphics Three Dimensional (G3D) es un Engine Open Source, es muy usado en juegos comerciales y aplicaciones para simuladores militares. G3D soporta rendering en tiempo real, off-line rendering. Proporciona un conjunto de rutinas y estructuras comunes que son necesitadas en todas las aplicaciones gráficas. Garantiza el uso de librerías de bajo nivel como OpenGL y los sockets para la red, se pueden usar sin restricciones de las funcionalidades y de rendimiento. Presenta una arquitectura robusta y optimizada, permitiendo integrar otras librerías de forma sencilla y rápida; convirtiéndose en una excelente herramienta para el desarrollo de videojuegos y simuladores **[11]**.

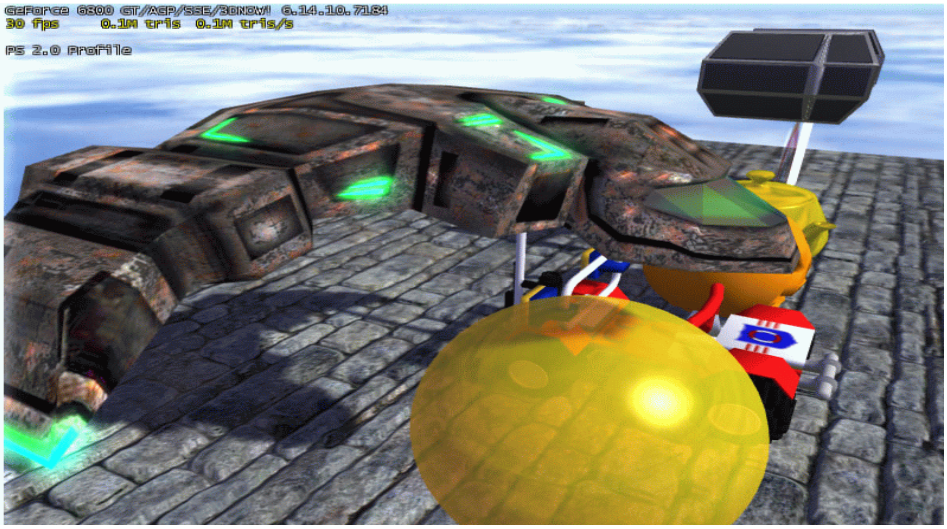


Figura 8: *Motores Gráficos – G3D Engine.*

Características Generales de G3D Engine:

Autor	Morgan McGuire
API Gráfica	OpenGL, DirectX
Lenguaje de Programación	C/C++, XCode, y GCC compatible
Documentación	Sí
Sistema Operativo Soportado	Windows, OS X, Linux, and FreeBSD
Estado	Versión 7.0
Características	
Características Generales	• Diseño Orientado a Objetos
Física	• Detección de Colisiones
Iluminación	• Volumétrica
Sombra	• *Shadow Mapping, Projected Planar, Shadow Volume*
Texturizado	• Básico • Soporta imágenes JPG, PNG, BMP, PPM, PCX, TGA, DDS, e ICO
Shaders	• *Vertex*, *Pixel*, GLSL, Cg, y ASM con chequeo de errores y extensiones
Administrador de Escenas	• BSP, LOD
Animación	• Animación por *Keyframe*

Mallas	• Cargado de Mallas. Soporta 3DS, IFS, MD2, BSP, y modelos personalizables.
Efectos Especiales	• Mapeo del entorno, destellos, *Billboarding*, Sistema de Partículas, Cielo, Agua, Fuego, Explosiones, Neblina, Espejos.
Terreno	• *Rendering*.
Sistema de Red	• Cliente-Servidor, Red basada en TCP y UDP.
Inteligencia Artificial	• *Scripted*.
Rendering	• Funciones Fijas, Fuentes.
GUI	• GUI parametrizable, con Botones, Lists, Edit, Scrollbars.
Tools	• Visualizador de Modelos, GPU benchmark, Herramientas de depuración en tiempo real. • Administrador automático de memoria opcional. • Matriz n x m optimizada. • Configuración de ficheros de lectura/escritura. • Ficheros Javas y clases de red.

Tabla 1: *Características de G3D.*

Figura 9: *Motores Gráficos – G3D Engine.*

1.4.2 Ogre3D (Object Oriented Graphics Engine)

OGRE 3D (acrónimo del inglés Object-Oriented Graphics Rendering Engine) es un motor de renderizado 3D orientado a escenas, escrito en el lenguaje de programación C++. Sus bibliotecas evitan la dificultad de la utilización de capas inferiores de librerías gráficas como OpenGL y Direct3D, además, proveen una interfaz basada en objetos del mundo y otras clases de alto nivel. El motor es software libre, licenciado bajo LGPL y con una comunidad muy activa. Es un flexible motor escrito en C++, orientado al crecimiento de la escena y diseñado con el objetivo principal de permitir a los programadores producir aplicaciones

utilizando gráficos en 3D, acelerados por hardware. Este motor fue diseñado principalmente para proveer a los programadores con una solución de primera para gráficos. Su belleza recae en que puede usarse para todo lo que el creador lo requiera: juegos, simulaciones, aplicaciones de negocios, entre muchas más **[12]**. Además, los requerimientos del motor pueden variar enormemente, incluso dentro de la industria de juegos. Su beneficio principal es su diseño coherente y la documentación detallada y consistente que viene con el motor.

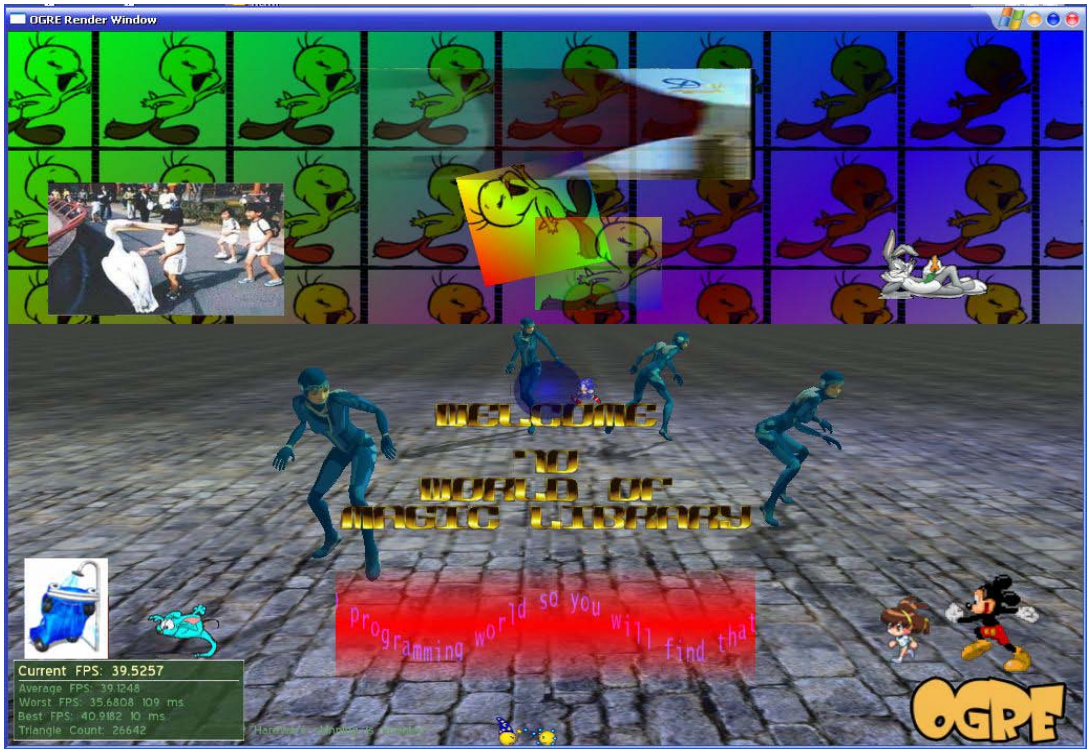


Figura 10: *Motores Gráficos – Ogre3D.*

Características Generales de OGRE:

Autor	Steve Streeting
API Gráfica	OpenGL, DirectX
Lenguaje de Programación	C/C++
Documentación	Sí
Sistemas Operativos	Windows, Linux, Mac OS
Estado	Productiva/Estable
Características	
Características Generales	• Diseño Orientado a Objeto, Arquitectura de *Plugin*, Sistema de Salva/Carga. • Simple, interfaz orientada a objeto fácil de usar, diseñada para minimizar los esfuerzos necesarios para renderear escenas 3D, y para ser independiente de la implementación 3D (e.g. Direct3D/OpenGL). • Arquitectura flexible de plugin que permite al motor gráfico ser extendido sin realizar una recompilación. • Diseño de una documentación completa y clara de todas las clases del motor gráfico. • Soporta ZIP/PK3 para compactar.
Scripting	• Lenguaje *Scripted*.
Física	• Básica, Detección de Colisiones, Cuerpos Rígidos. • Incluye acoples para sistemas físicos como (*Open Dynamics Engine*, *Novodex* y *Tokamak*).
Iluminación	• *Per-vertex*, *Per-pixel*, *Lightmapping*. • Puede tener un número ilimitado de luces en la escena.
Sombra	• *Shadow* Mapping, *Shadow Volume*.
Texturizado	• Básico, *Multi-Texturing*, *Bumpmapping*, *Mipmapping*, *Volumetric*, *Projected*. • Soporta imágenes PNG, JPEG, TGA, BMP y DDS.
Shaders	• *Vertex*, *Pixel*, *High Level*: • Soporta *vertex* y *shaders*, ambos programas de bajo nivel escrito en ensamblador, y programas de alto nivel escrito en Cg o DirectX9 HLSL, proveyendo soporte automático para muchos parámetros constantes. • Soporta GLSL.
Administrador de Escenas	• General, BSP, *Octrees*, LOD.
Animación	• Cinemática Inversa, Animación Esquelética, Mezcla de Animación.
Mallas	• Carga de Malla, *Skinning*, *Progressive*. • *Skinning* por aceleración gráfica. • Exporta a partir de muchas herramientas de modelación que incluye Milkshape3D, 3D Studio Max, Maya, Blender y Wings3D.
Superficies & Curvas	• *Splines*.
Efectos Especiales	• *Billboarding*, Sistema de Partículas, Cielo, Agua, Neblina.
Rendering	• Funciones fijas, Fuentes, GUI. • Material LOD. • Soporte para múltiples técnicas de materiales. • Los objetos transparentes son automáticamente administrados. • Sistema de Fuentes: Fuentes *TrueType* y texturas precreadas. • Sistema GUI 2D con Botones, *Lists*, *Edit Boxes*, *Scrollbars*.

Tabla 2: *Características de Ogre3D.*

Figura 11: *Motores Gráficos – Ogre3D.*

1.5 Características de las diferentes Bibliotecas Gráficas

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

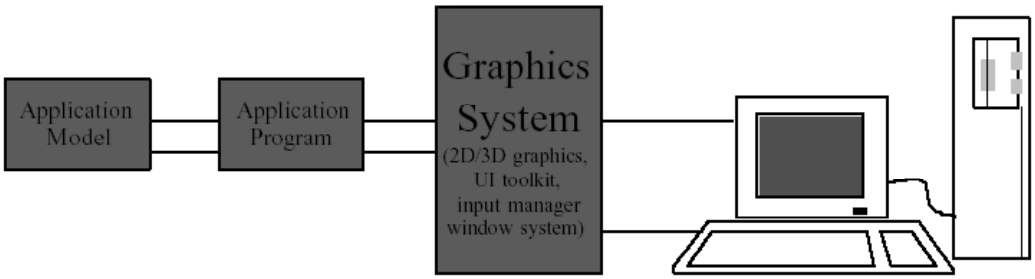


Figura 12: *Definición Librerías Gráficas.*

Entre las Librerías Gráficas más conocidas se encuentran**:**

- OpenGL
- DirectX
- GKS
- PHGS
- PEX

1.5.1 OpenGL.

OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma, para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en un grupo de funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples. Fue desarrollada originalmente por Silicon Graphics Inc. (SGI) en 1992 y se utilizan ampliamente en realidad virtual, representación científica, visualización de información y simulación de todo tipo, en el desarrollo de videojuegos, donde compite con Direct3D en plataformas Microsoft Windows. A partir de OpenGL, los fabricantes de hardware crean implementaciones, que son bibliotecas de funciones que se ajustan a los requisitos de la especificación, utilizando aceleración hardware cuando es posible.

OpenGL tiene dos propósitos esenciales:

- Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.
- Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

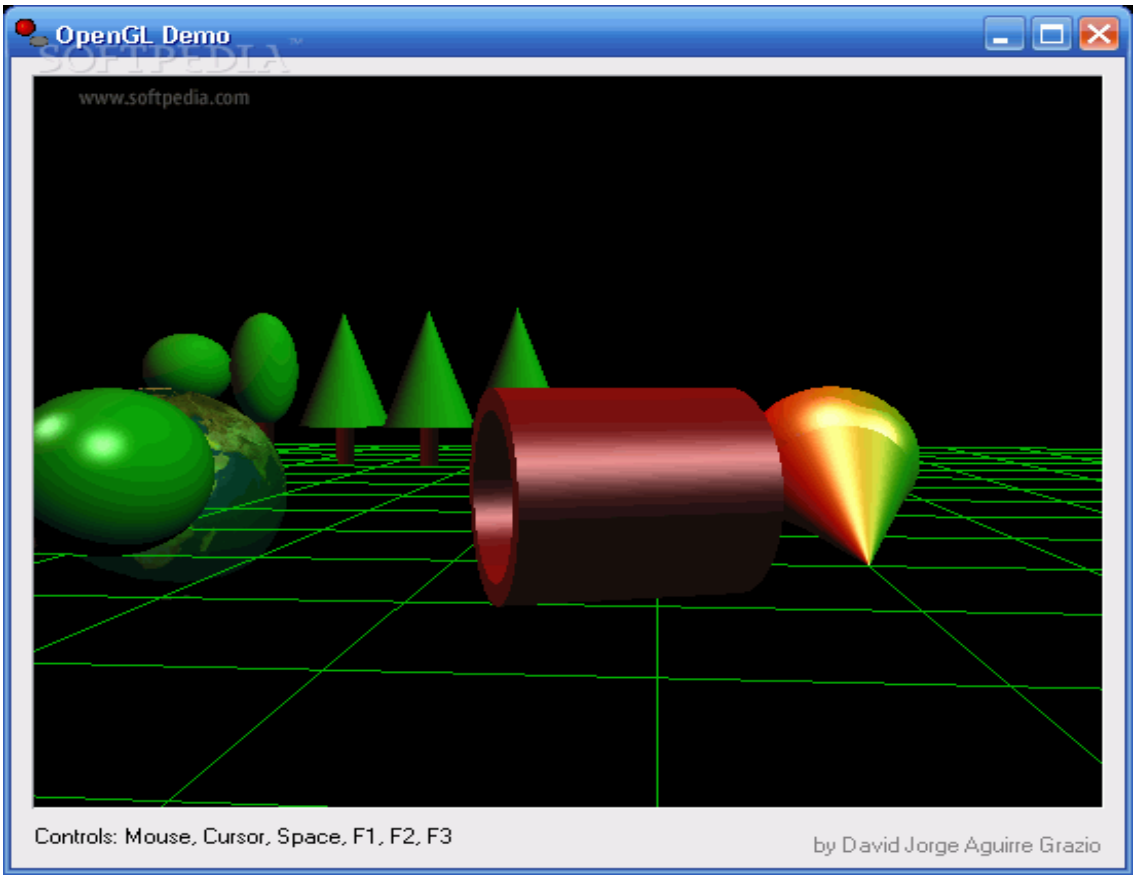


Figura 13: *Librerías Gráficas – OpenGL.*

Se han programado varias bibliotecas externas que añaden características no disponibles en el propio OpenGL. Algunas de ellas son:

- GLU: Ofrece funciones de dibujo de alto nivel basadas en primitivas de OpenGL. Las funciones de GLU se reconocen fácilmente pues todas empiezan con el prefijo glu.
- GLUT: API multiplataforma que facilita una rudimentaria funcionalidad para el manejo de ventanas e interacción por medio de teclado y ratón.
- GLUI: Interfaz de usuario basada en GLUT; proporciona elementos de control tales como botones, cajas de selección y spinners. Es independiente del sistema operativo, sustentándose en GLUT para manejar los elementos dependientes del sistema.

1.5.2 DirectX.

DirectX es una colección de API creadas y recreadas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y video en la plataforma Microsoft Windows. A pesar de ser desarrollado exclusivamente para la plataforma Windows, una implementación open source de su API se encuentra en progreso para sistemas Unix (en particular Linux) y X Windows System por el proyecto WineHQ, del cual existe fork propietario, Cedega, desarrollada por la empresa de software Transgaming y orientada a la ejecución de juegos desarrollados para Windows bajo sistemas Unix.

DirectX consta de los siguientes APIs:

- Direct3D: Utilizado para el procesado y la programación de gráficos en tres dimensiones (una de las características más usadas de DirectX).
- Direct Graphics: Para dibujar imágenes en dos dimensiones (planas) y para representación de imágenes en tres dimensiones.
- DirectInput: Utilizado para procesar datos del teclado, mouse, joystick y otros controles para juegos.
- DirectPlay: Para comunicaciones en red.
- DirectSound: Para la reproducción y grabación de sonidos de ondas.
- DirectMusic: Para la reproducción de pistas musicales compuestas con DirectMusic Producer.
- DirectShow: Para reproducir audio y video con transparencia de red.

- DirectSetup: Para la instalación de componentes DirectX.

Figura 14: *Librerías Gráficas – DirectX.*

1.6 Metodologías y Herramientas de Desarrollo

1.6.1 Metodología:

El desarrollo de software no es, sin dudas, una tarea fácil. Como resultado de este problema ha surgido una alternativa desde hace mucho: la metodología. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software, con el fin de hacerlo más predecible y eficiente. Éstas se basan en una combinación de los modelos de procesos genéricos (cascada, evolutivo, incremental, entre otras). Adicionalmente, una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, y demás. Habitualmente se utiliza el término “método”

para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño.

1.6.1.1 Rational Unified Process

Rational Unified Process (RUP, siglas en inglés) es un proceso de desarrollo de software y junto con el lenguaje unificado de modelado, UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Provee un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad, que satisfaga la necesidad del usuario final dentro de un tiempo y presupuesto previsible. El RUP mejora la productividad del equipo, ya que permite que cada miembro del grupo, sin importar su responsabilidad específica, acceda a la misma base de datos de conocimiento. Permitiendo que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar software. Contienen tres características fundamentales que lo hacen una metodología robusta y poderosa: es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Cuando se caracteriza RUP como dirigido por casos de uso se refiere a que los casos de uso dirigen todo el proceso del desarrollo del software, ya que éstos reflejan lo que los clientes necesitan y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo del sistema.

Centrado en la arquitectura se refiere a que se incluyen los aspectos estáticos y dinámicos más significativos del sistema. Además recoge una serie de factores como la plataforma en la cual funciona el software, los bloques de construcción reutilizables que se tienen, consideraciones de implementación, sistemas heredados y requisitos no funcionales.

Es iterativo e incremental porque el proyecto o el desarrollo de una aplicación se pueden dividir en partes y desarrollarlas de manera iterativa, incrementándose a medida que se integran unas con otras hasta llegar a formar la tarea final. Las iteraciones hacen referencia a pasos en el flujo de trabajo y los incrementos, el crecimiento del producto.

Las actividades de RUP se centran en crear y mantener modelos utilizando UML. Como no existe un único proceso que sea apropiado para todos los desarrollos, RUP es un proceso configurable. Se adapta tanto a

grupos pequeños de desarrollo como a grandes organizaciones. Basándose en lo que se consideran las mejores prácticas de desarrollo de software, este resulta apropiado para ser aplicado en una amplia gama de proyectos y organizaciones.

1.6.1.2 Extreme Programming (XP)

La metodología Extreme Programming (XP) o Programación Extrema es una variante de las metodologías ágiles con más aceptación en la comunidad internacional de desarrollo. Los fundamentos de la misma, según su creador, son: mejorar la comunicación, buscar la simplicidad, buscar retroalimentación y que siempre hay que proceder con valentía. Una de las herramientas más importantes en la metodología XP es el desarrollo orientado a pruebas, que utiliza las pruebas unitarias como eje de todo desarrollo. Las interacciones suelen ser muy cortas y se promueve a los programadores a buscar soluciones y experiencia con ellas, programar sin miedo a descomponer el sistema.

1.6.2 Herramientas de Desarrollo:

Las herramientas de desarrollo son aquellos programas o aplicaciones que tienen cierta importancia en el desarrollo de un programa (programación). Pueden ser de importancia vital (como un ensamblador, un compilador o un editor) o de importancia secundaria, como una IDE (Interfaz de Desarrollo Estructurada), que simplifican la labor y ahorran mucho tiempo. Permiten organizar el código, colorean la sintaxis del lenguaje, ayudan a depurar, etc.

1.6.2.1 Visual Paradigm:

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es muy fácil de usar y soporta la última notación UML 2.1, ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI,

generador de informes, editor de figuras, integración con MS Visio, plug-in, integración IDE con Visual Studio, IntelliJ IDEA, Eclipse, NetBeans y otros **[13]**.

Características:

- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversión (nueva característica).
- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).
- Ingeniería de ida y vuelta
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso – Entorno, todo en uno, para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso
- Diagramas de flujo de datos.
- Soporte ORM - Generación de objetos Java desde la base de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio.

- Editor de figuras.
- Plataforma Java (Windows/Linux/Mac OS X).

1.6.2.2 Rational Rose:

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML. Esta herramienta propone la utilización de cuatro tipos de modelos para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Es una de las más poderosas herramientas de modelado visual para el análisis y diseño de sistemas basados en objetos. Rational Rose Enterprise es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java™/J2EE™, Visual C++® y Visual Basic®. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado (Unified Modeling Language, UML siglas en inglés) para el equipo que facilita la creación de software de calidad más rápidamente.

Características:

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Característica de control por separado de componentes, modelo que permite una administración más granular y el uso de modelos.
- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo-código configurables.
- Soporte Enterprise Java Beans™ 2.0.
- Capacidad de análisis de calidad de código.

- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo **[14]**.

1.6.2.3 Microsoft Visual Studio TeamSystem 2008

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002).

Windows Visual Studio TeamSystem 2008 tiene la posibilidad de implementar aplicaciones para soluciones integrales que aprovechen de manera óptima la ventaja de cada lenguaje. Acelera de manera significativa la producción de software. La interfaz es altamente amigable con el usuario, permitiendo que el tiempo en implementar una solución o aplicación determinada sea mucho menor. Los ejecutables desarrollados por esta herramienta son generalmente de menor tamaño, lo que hace que ocupe un lugar cimero en la producción de software al lograr aplicaciones óptimas y de poco volumen. Ofrece una amplia gama de herramientas para todas las fases del desarrollo de software, incluidos la creación, la prueba, la implementación, la integración y la administración. Además, permite a los desarrolladores comunicarse mediante PCs, servidores, la Web y dispositivos móviles.

Microsoft Visual Studio TeamSystem 2008 ofrece herramientas de desarrollo de aplicaciones, como:

- Sistema de proyectos para administrar los datos requeridos para el desarrollo de aplicaciones de su proyecto.
- Herramientas de edición de código para escribir y modificar texto y código.
- Herramientas de refactorización y depuración para mejorar el código e identificar y resolver errores lógicos.
- Herramientas de generación de informes para ayudarlo a llevar registros y para tareas de administración.
- Herramientas de plataformas para la escritura de aplicaciones para tecnologías de Office, Windows CE, .NET y Windows.
- Herramientas avanzadas para diseñar, implementar, analizar, probar y evaluar el progreso del desarrollo de aplicaciones.
- Opciones de lenguajes que incluyen JScript 8.0, Visual Basic 2008, Visual C# 2008 y Visual C++ 2008.

1.6.2.4 Microsoft Visual Studio Express Edition 2008:

Microsoft Visual Studio Express Edition es un programa de desarrollo en un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows, distribuido por Microsoft Corporation. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Es de carácter gratuito y es proporcionado por la compañía Microsoft Corporation, orientándose el producto a principiantes, estudiantes y entusiastas en la programación web y de aplicaciones, ofreciéndose dicha aplicación a partir de la versión 2005 de Microsoft Visual Studio **[15]**.

Características:

Visual Studio Express permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002, se incorporan las versiones Framework 3.5 y Framework 4.0 para las ediciones 2005 ,2008, 2010). Así se

pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Cabe destacar que estas ediciones son iguales al entorno de desarrollo comercial de Visual Studio Professional, pero sin características avanzadas. Las ediciones que hay dentro de cada suite son:

- ✓ Visual Basic Express Edition.
- ✓ Visual C# Express Edition.
- ✓ Visual C++ Express Edition.
- ✓ Visual J# Express Edition (Desaparecido en Visual Studio express 2008).
- ✓ Visual Web Developer Express Edition:

Se emplea en la programación con lenguaje ASP.NET. Está orientado a la programación y diseño web, incluyendo un editor visual WYSIWYG y otro HTML, con autocompletado de código (IntelliSense), coloración de sintaxis y validación. Aparte de ASP. NET también soporta Visual Basic .NET y C Sharp (C#). Contiene un servidor web local para realizar pruebas en ASP.NET, un depurador para ubicar errores en el código fuente y una herramienta de publicación en línea de sitios creados.

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de MS SQL Server llamada SQL Server Express Edition, cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente utiliza un procesador y 1 Gb de RAM, y no cuenta con el Agente de SQL Server.

En el pasado se incluyeron los siguientes productos, actualmente desaparecidos en versiones como Visual Studio express 2005 y 2008:

- ✓ Visual InterDev.
- ✓ Visual J++.
- ✓ Visual FoxPro.
- ✓ Visual SourceSafe.

1.7 Lenguajes Informáticos

Un lenguaje informático es un lenguaje usado por, o asociado con, ordenadores.

Los lenguajes informáticos pueden clasificarse en:

- ✓ Lenguajes de Programación
- ✓ Lenguajes de Modelado

1.7.1 Lenguajes de Programación: C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C, con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Entre sus principales características está el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica, además de ser un lenguaje de alto nivel que está considerado como un lenguaje potente al poder trabajar tanto en bajo como en alto nivel. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores) y de poder crear nuevos tipos que se comporten como tipos fundamentales. Además presenta una biblioteca estándar muy poderosa.

1.7.2 Lenguajes de Modelado: UML

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, así como aspectos concretos: expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables **[16]**. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en el desarrollo de software, entregando gran variedad de

formas para dar soporte a una metodología de desarrollo de software (tal como RUP). En otras palabras, es el lenguaje en el que está descrito el modelo.

1.8 Biblioteca de desarrollo de Interfaz Gráfica de Usuario: Qt

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Dicha biblioteca es utilizada principalmente en KDE, Google Earth, Skype, Qt Extended, Adobe Photoshop Album, VirtualBox y Opie. Es producido por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el 17 de junio de 2008.3

Qt es utilizada en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros. Utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings.

Funciona en todas las principales plataformas y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

Distribuida bajo los términos de GNU Lesser General Public License (y otras), Qt es software libre y de código abierto.

Qt se encuentra disponible para sistemas tipo Unix con el servidor gráfico X Windows System (Linux, BSDs, Unix), para Apple Mac OS X, para sistemas Microsoft Windows, para Linux empotrado (en inglés Embedded Linux, para sistemas integrados como PDA, Smartphone, entre otros más) y para dispositivos que utilizan Windows CE7

Qt Software anunció el 20 de octubre de 2008 una versión de Qt para la plataforma S60.8 9

Adicionalmente también está disponible QSA (Qt Scripts for Applications), que, basándose en ECMAScript/Java Script, permite introducir y crear scripts en las aplicaciones creadas con Qt.

Hay tres ediciones de Qt disponibles en cada una de estas plataformas, llamadas:

GUI Framework – edición con nivel reducido de GUI, orientado a redes y bases de datos **[17]**.

Full Framework – edición completa comercial

Open Source – edición completa Open Source

Conclusiones parciales

En el transcurso de este capítulo han quedado sentadas las bases para el entendimiento del tema. Para lograr lo antes expuesto se hizo referencia a las características de los ficheros 3D más usados, analizando las ventajas y desventajas que ofrecen cada uno de ellos, además del análisis detallado de las diferentes herramientas de diseño 3D. También se realizó un estudio profundo de los diferentes editores de personajes 3D que existen hoy en el mundo de la Realidad Virtual, así como las diferentes características que presentan los diferentes motores de render gráficos y las principales librerías gráficas existentes en la actualidad. Finalmente se aborda sobre las metodologías, las herramientas de desarrollo y los lenguajes informáticos más utilizados para la edición 3D.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

Introducción

En este capítulo se realiza una descripción de la propuesta a defender, en vista a solucionar la situación actual en el campo de acción. Se proponen soluciones técnicas para el funcionamiento de la herramienta gráfica que permita la edición de los modelos 3D. Se hará una descripción a nivel conceptual de la solución propuesta en el capítulo anterior. Se definen las reglas a cumplir para asegurar el correcto funcionamiento del sistema, nombradas como Reglas del Negocio. Se tratarán los conceptos más importantes del área de interés mediante un Modelo de Dominio. Además, serán descritas las capacidades que deberá cumplir el sistema en forma de Requisitos Funcionales y No Funcionales y por último, se hará un análisis profundo de los Casos de Uso del sistema.

2.1 El Sistema

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

Para poder editar las partes especificas del modelo 3D se hará uso de un fichero que contenga la información de dicho modelo 3D (archivo .mesh). Teniendo como entrada el modelo 3D seleccionado, previamente diseñado con la herramienta 3D Max Studio, el motor gráfico entraría a ser parte fundamental dentro de la herramienta (Ogre 3D), pues permitiría la carga y exportación de dicho modelo 3D. La herramienta se encargaría de: editar dicho modelo 3D (Personaje), dando la posibilidad de modificar disímiles características del modelo seleccionado y salvaguardar dichos cambios para su uso futuro.

Todo esto daría como salida una aplicación visual, en la cual quedaría modificado dicho modelo 3D (Personaje).

Este proceso se puede apreciar en la siguiente figura:

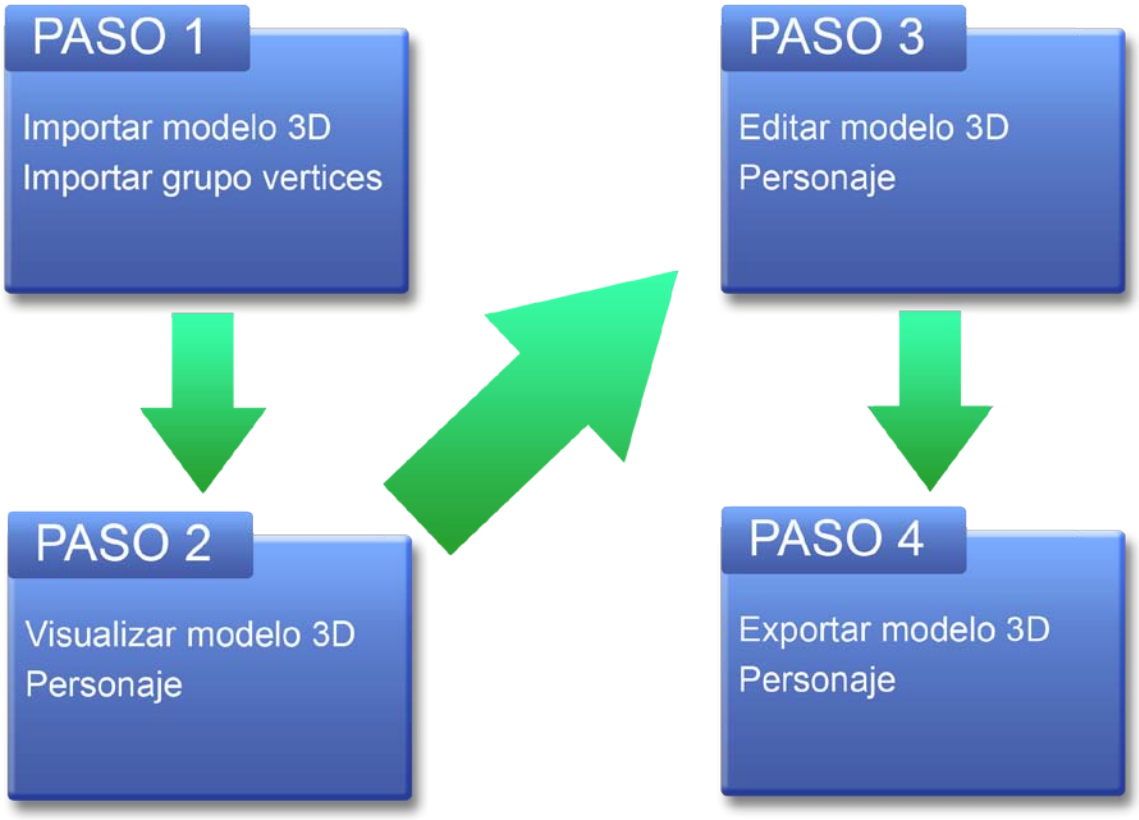


Figura 15: *Dinámica de un Editor de Personajes 3D.*

2.2 Propuesta de la Estructura del fichero: Grupo de Vértices

El fichero estará organizado de esta manera, primero el encabezado (Header) y después el grupo de vértices (VertexGroup). Está organizado en forma de bloques, este tipo de estructura permite que a la hora de leer el fichero se cargue un bloque de datos por completo, el cabezal del disco duro sólo tendrá que moverse en una sola dirección una vez que se está leyendo un bloque, logrando mayor velocidad de lectura cuando se está cargando el fichero. Por último, el bloque Grupo de Vértices (VertexGroup) estará repitiéndose tantas veces como grupo de vértices tenga el modelo 3D.

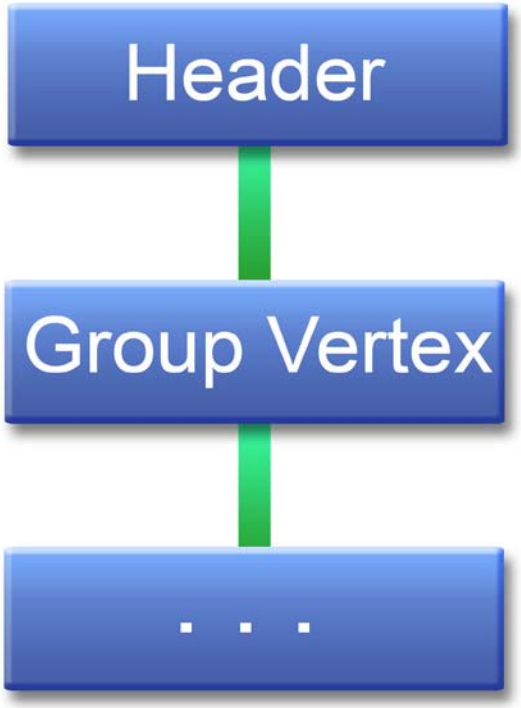


Figura 16: *Propuesta de Estructura de un Fichero Grupo de Vértice.*

2.2.1 Características del bloque Header

Header es el bloque que representa el encabezado del fichero 3D, brindando informaciones como el nombre del fichero y la cantidad de grupos de vértices que posee el personaje. A continuación se muestra la estructura del encabezamiento (**Header**) para un mejor entendimiento:

- # **nombre_fichero:** nombre del fichero.
- **int cant_grupos _vértices:** la cantidad de grupos de vértices del personaje.

¿El por qué incluir un encabezado en el formato de fichero?

El encabezado es un contenedor de informaciones generales que describen aspectos básicos del fichero, dando la posibilidad de conocer algunos elementos básicos del formato en el que se está trabajando.

2.2.2 Características del bloque VertexGroup

VertexGroup contiene toda la información respecto a los grupos de vértices presentes en el personaje. Brinda informaciones como el nombre del grupo de vértice, la cantidad de vértices de cada grupo de vértices, el índice del vértice que tiene el personaje y el peso de cada vértice. El formato de almancenamiento del fichero es en binario por brindar mayor compresión de datos y confidencialidad en la información. A continuación se muestra la estructura del **VertexGroup** para un mejor entendimiento:

- **char [32] nombre_grupo_vertice:** nombre del grupo de vértice.
- **int cant_vertices:** cantidad de vértices que tiene un grupo de vértices.
- **int index_vertice:** índice de un vértice en el personaje.
- **float peso:** peso asociado de un vértice.

2.3 Forma de Salvado

El fichero se exportará en formato .mesh, ya que es el formato de archivo que utiliza Ogre 3D y está codificado en forma binaria. Los formatos de ficheros de este tipo son de menor tamaño que los que están en forma de texto plano, disminuyendo el tiempo de salvado y de carga del mismo. Además, esta forma de salvado brinda la posibilidad de tener confidencialidad en la información contenida en el fichero.

2.4 Motor Gráfico:

2.4.1 Ogre 3D 1.6

Se decide utilizar Ogre 3D como motor gráfico debido a que es orientado a escenas, escrito en el lenguaje de programación C++, sus bibliotecas evitan la dificultad de la utilización de capas inferiores de librerías gráficas como OpenGL y Direct3D y además, proveen una interfaz basada en objetos del mundo y otras

clases de alto nivel. El motor es software libre, licenciado bajo LGPL, con una comunidad muy activa y vasta documentación en Internet.

2.5 Herramientas de Desarrollo:

2.5.1 Visual Paradigm Suite 3.4

La herramienta CASE que se utilizará es Visual Paradigm Suite 3.4, ya 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. Además, dispone de una versión libre para la comunidad. Es muy fácil de usar y soporta la última notación UML; por lo que el lenguaje de modelado ideal a utilizar es el UML (valga la redundancia).

2.5.2 Microsoft Visual Studio Express Edition 2008:

Se trabajará con Microsoft Visual Studio Express Edition 2008, ya que soporta el lenguaje Visual C++, es de carácter gratuito y es orientado a principiantes, estudiantes y entusiastas en la programación de aplicaciones.

2.6 Metodología de desarrollo de software:

Se escogió la metodología RUP; ya que constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es robusta y dirigida por casos de uso, centrada en la arquitectura, iterativa e incremental. Permite efectuar cambios de última hora.

2.7 Biblioteca de desarrollo de Interfaz Gráfica de Usuario:

Se seleccionó QT como biblioteca de interfaz gráfica de usuario, ya que utiliza el lenguaje de programación C++ de forma nativa, es completamente gratuita para aplicaciones de código abierto, con Qt se pueden desarrollar ricas aplicaciones gráficas. Además Qt dispone de una amplia gama de herramientas que facilitan la creación de formularios, botones y ventanas de diálogo con el uso del ratón,

las aplicaciones creadas con la misma son muy elegantes, se ven y se operan mejor que las aplicaciones nativas.

2.8 Reglas del Negocio

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

- ✓ Los ficheros de textura que se importarán tienen que ser .jpg, .gif o .png.
- ✓ El formato del modelo 3D que se importará tiene que ser .mesh.
- ✓ Los nombres de los ficheros de textura, del formato de modelo 3D y del material no pueden tener espacios entre caracteres en sus nombres.
- ✓ El modelo 3D puede tener asignado un material o varios materiales (Permite cargar multimateriales).
- ✓ Las unidades de medida del 3D Max tienen que ser centímetros.
- ✓ El nombre del grupo de vértices no puede tener más de 32 caracteres.
- ✓ El modelo 3D modificado se exportará con el formato .mesh.
- ✓ Los modelos 3D tienen que exportarse en el eje de coordenadas <0, 0, 0>.

2.9 Modelo de dominio

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés y enlaza estos objetos unos con otros. La identificación y la asignación de un nombre para estos objetos ayudan a desarrollar un glosario de

términos, que permitirá una mejor comunicación entre todos los que están trabajando en el sistema. Más adelante, los objetos del dominio ayudan a identificar algunas de las clases que se analizan y diseñan en el sistema. El Modelo de Dominio representa un acercamiento a la solución propuesta, donde se modelan los principales conceptos implicados en el desarrollo de la solución, así como las relaciones existentes entre ellos. Se aplica el Modelo de Dominio cuando no es posible encontrar una estructura en los procesos de negocios, es decir, éstos no están bien definidos y no se pueden determinar con claridad sus fronteras ni quienes se benefician de los mismos. El Diagrama de Dominio servirá como guía para el futuro diseño del sistema.

A continuación se representa el Modelo de Dominio referente a la herramienta propuesta.

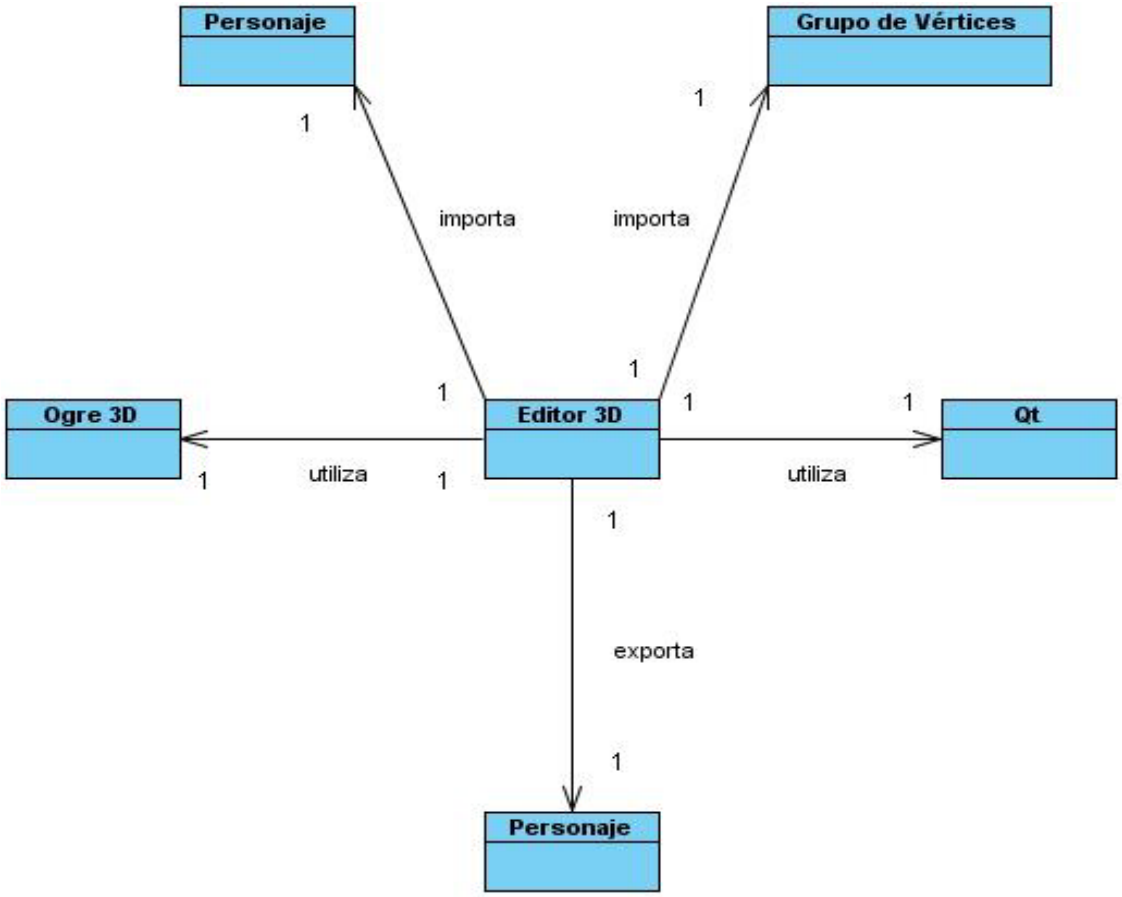


Figura 17: *Modelo de Dominio*

2.9.1 Glosario de términos del Modelo de Dominio:

Editor 3D: Controla todas las funcionalidades de la aplicación.

Personaje: Controla toda la información referente al personaje.

Ogre 3D: Motor Gráfico que visualiza al personaje 3D.

Qt: Biblioteca de desarrollo de Interfaz Gráfica de Usuario para modificar el personaje 3D.

Herramienta Diseño 3D: Crea el personaje 3D y lo exporta como .mesh.

Grupo de Vértices: Elementos específicos a modificar del personaje 3D.

2.10 Captura de Requisitos

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

Los requisitos constituyen un acuerdo que se firma entre el cliente y los desarrolladores del sistema, donde quedan reflejados las cualidades y capacidades que tendrá el mismo.

2.10.1 Requisitos Funcionales

Son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los Requisitos Funcionales de la herramienta a desarrollar se muestran a continuación:

RF1: Iniciar aplicación.

RF1.1 Inicializar OGRE.

RF1.1.1 Inicializar cámara.

RF1.2 Inicializar OIS.

RF1.3 Inicializar QT.

RF1.4 Crear escenario de trabajo (Interfaz).

RF2: Importar modelo 3D.

RF2.1 Chequear que exista el fichero 3D.

RF2.2 Cargar archivo en formato .mesh.

RF2.3 Chequear que el .mesh seleccionado cumple con las especificaciones de dicho formato.

RF2.4 Permita seleccionar el modelo 3D del directorio de modelos.

RF3: Importar fichero Grupo de Vértices.

RF3.1 Permitir identificar el grupo de vértices.

RF3.2 Permitir identificar el peso por cada vértice.

RF3.3 Permitir identificar el índice de un vértice.

RF4: Editar Personaje 3D.

RF4.1 Seleccionar características (grupo de vértices).

RF4.2 Editar características (grupo de vértices).

RF4.3 Permitir modificar el valor de un grupo de vértices (aumentar o disminuir dicho valor).

RF4.4 Permitir aplicar otra textura al personaje.

RF5: Exportar modelo 3D.

RF5.1: Definir el directorio destino del archivo 3D.

RF5.2: Guardar archivo 3D.

RF5.3 Exportar el modelo 3D modificado.

RF5.4 Exportar el fichero grupo de vértices modificado.

RF6: Cerrar la aplicación.

RF6.1 Destruir Recursos.

2.10.2 Requisitos No Funcionales

Los Requisitos No Funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los Requisitos No Funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, ya que si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Los requisitos no funcionales de la herramienta a desarrollar son los que se relacionan a continuación:

Usabilidad:

- Cualquier usuario que desee editar un personaje 3D.

Portabilidad:

- Debe ser portable a cualquier sistema operativo Windows.

Requerimientos de Software:

- Sistema operativo Windows XP o superior.

Requerimientos de Hardware:

- 512 MB RAM o superior.
- Procesador Intel Pentium IV o superior.
- DirectX 9.0c o mayor.
- 40 MB de espacio libre en disco duro.
- 256 MB de video o superior

Restricciones en el Diseño e Implementación:

- Lenguaje de programación C++ bajo el paradigma de Programación Orientada a Objetos.

Diseño e Implementación:

- Se debe desarrollar utilizando el Entorno de Desarrollo Integrado Microsoft Visual Studio Express Edition 2008.
- Se debe desarrollar utilizando la biblioteca de desarrollo de Interfaz Gráfica de Usuario QT.
- Se debe desarrollar utilizando el motor gráfico Ogre 3D 1.6.

2.11 Modelo de casos de uso del sistema

En esta sección se conciben los Casos de Uso del sistema, así como los actores que van a interactuar con cada uno de ellos. Además se seleccionan los Casos de Uso correspondientes al ciclo de desarrollo para realizar sus especificaciones textuales en formato expandido.

Los Casos de Uso son un artefacto clave en el proceso unificado de desarrollo de software, ya que contituyen el depósito principal de los Requisitos Funcionales que gobiernan el diseño, la construcción, las pruebas y muchos otros aspectos de este proceso.

2.11.1 Actores del sistema

Los Actores representan entidades que interactúan con el sistema, un actor del sistema es aquel que se beneficia de los resultados de las funcionalidades del mismo. También pueden ser otros sistemas con los que el sistema interactúa.

En este caso particular quien hará uso del sistema será un diseñador de personajes 3D, el cual constituye el actor del sistema, específicamente será llamado, como se muestra en la tabla 1.

Actor	Descripción
Diseñador	Persona que trabaja e interactúa con la herramienta.

Tabla 3: *Actor del Sistema.*

2.11.2 Casos de Uso del Sistema

Un Caso de Uso del sistema es un documento narrativo que describe la secuencia de un Actor (agente externo) que utiliza un sistema para completar un proceso. A continuación se expondrá el listado de Casos de Uso del sistema, que incluye su Actor, una breve descripción de cada uno, así como una referencia a su Requisito Funcional asociado.

CU 1: Importar modelo 3D

CU1:	Importar modelo 3D.
Actor:	Diseñador
Descripción:	Su propósito es cargar el modelo 3D en formato .mesh para editar el diseño 3D correspondiente.
Referencias:	RF2, RF2.1, RF2.2, RF2.3, RF2.4

Tabla 4: *CU1 Importar modelo 3D.*

CU 2: Importar Fichero Grupo de Vértices.

CU2:	Importar Fichero Grupo de Vértices.
Actor:	Diseñador
Descripción:	Su propósito es cargar el fichero que contiene los grupos de vértices para editar el diseño 3D correspondiente.
Referencias:	RF3, RF3.1, RF3.2, RF3.3

Tabla 5: *CU2 Importar Fichero Grupo de Vértices.*

CU 3: Editar personaje 3D.

CU3:	Editar personaje 3D.
Actor:	Diseñador
Descripción:	Su propósito es editar el modelo 3D correspondiente a partir de las características de dicho personaje (grupo de vértices).
Referencias:	RF4, RF4.1, RF4.2, RF4.3, RF4.4

Tabla 6: *CU3 Editar personaje 3D.*

CU 4: Exportar modelo 3D.

CU4:	Exportar modelo 3D.
Actor:	Diseñador
Descripción:	Su propósito es exportar el modelo 3D en formato .mesh para guardar el diseño 3D correspondiente.
Referencias:	RF5, RF5.1, RF5.2, RF5.3, RF5.4

Tabla 7: *CU4 Exportar modelo 3D.*

2.11.3 Diagrama de Casos de Uso del Sistema

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

A continuación se representa el diagrama de casos de uso del sistema referente a la herramienta propuesta.

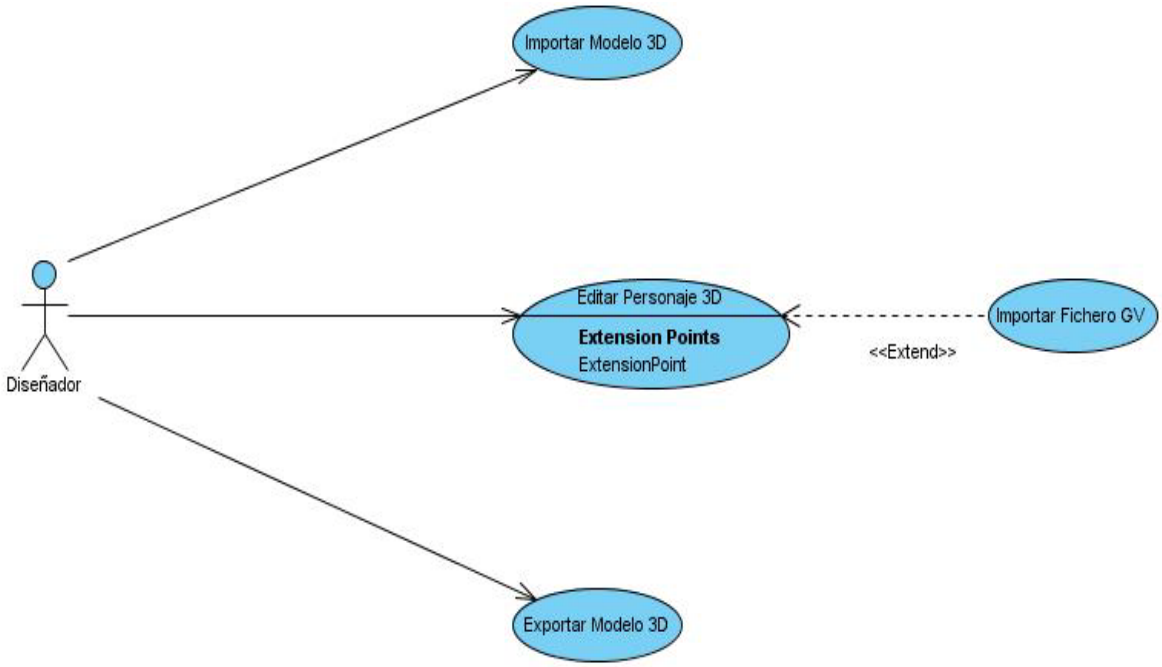


Figura 18: *Diagrama de Casos de uso del Sistema*

2.11.4 Descripción de los Casos de Uso del Sistema

CU 1: Importar modelo 3D.

Caso de Uso:	Importar modelo 3D.	
Actores:	Diseñador	
Resumen:	El caso de uso se inicia cuando el usuario elige en el menú principal la opción Cargar Personaje y especifica un modelo 3D .mesh a cargar. El sistema procede a cargar el modelo 3D seleccionado.	
Precondiciones:	-	
Referencias:	RF2, RF2.1, RF2.2, RF2.3, RF2.4	
Flujo Normal de Eventos		
Sección "Importar modelo 3D"		
Acción del Actor		**Respuesta del Sistema**
1. Selecciona la opción "Importar" del menú principal.		2 Se muestra el cuadro de diálogo Seleccionar Modelo 3D que le permitirá al Diseñador especificar el modelo 3D seleccionado.
3. Selecciona el modelo 3D y acciona el botón OK.		4. Se cierra el cuadro de diálogo y se procede a cargar el fichero .mesh especificado por el Diseñador.
Prioridad:	Crítico	
Postcondiciones:	Se visualizará en la pantalla el modelo 3D predeterminado.	

Tabla 8: *Descripción CU Importar modelo 3D.*

CU 2: Importar Fichero Grupo de Vértices.

Caso de Uso:	Importar Fichero Grupo de Vértices.
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el usuario elige en el menú principal

	la opción Cargar Grupo de Vértices. El sistema procede a cargar todas las características a modificar del modelo 3D seleccionado.
Precondiciones:	Realización del CU Importar modelo 3D.
Referencias:	RF3, RF3.1, RF3.2, RF3.3
Flujo Normal de Eventos	
Sección "Importar modelo 3D"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción "Importar FGV" del menú principal.	2 Se muestran todas las características a modificar del modelo 3D seleccionado (Grupo de Vértices).
Prioridad:	Secundario
Postcondiciones:	Se visualizará en la pantalla el modelo 3D predeterminado.

Tabla 9: *Descripción CU Importar Fichero Grupo de Vértices.*

CU 3: Editar personaje 3D.

Caso de Uso:	Editar personaje 3D.
Actores:	Diseñador
Resumen:	El caso de uso se inicia cuando el Diseñador escoge la opción de Editar Personaje 3D.
Precondiciones:	Realización del CU Importar modelo 3D. Realización del CU Importar Fichero Grupo de Vértices.
Referencias:	RF4, RF4.1, RF4.2, RF4.3, RF4.4
Flujo Normal de Eventos	
Sección "Editar personaje 3D**"**	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción "Editar Personaje 3D".	2. Muestra un menú que contiene los elementos necesarios para modificar las características del personaje 3D (Grupo de Vértices).
Prioridad:	Crítico
Postcondiciones:	Se visualizará en la pantalla el resultado obtenido del proceso de

	edición del personaje 3D.

Tabla 10: *Descripción CU Editar personaje 3D.*

CU 4: Exportar modelo 3D.

Caso de Uso:	Exportar modelo 3D.	
Actores:	Diseñador	
Resumen:	El caso de uso se inicia cuando el usuario elige en el menú principal la opción Salvar Personaje y guarda el modelo 3D (archivo .mesh).	
Precondiciones:	Debe haberse realizado el CU Editar personaje 3D.	
Referencias:	RF5, RF5.1, RF5.2, RF5.3, RF5.4	
Flujo Normal de Eventos		
Sección "Exportar modelo 3D**"**		
Acción del Actor		**Respuesta del Sistema**
1. Selecciona la opción "Exportar" del menú principal.		2. El sistema muestra un cuadro de diálogo que permite especificar la dirección donde se va a salvar el archivo .mesh, el archivo .fgv, el material y las texturas.
3. El usuario especifica la ruta donde se va a guardar el modelo 3D seleccionado (archivo .mesh) y pulsa el botón aceptar.		4. El sistema verifica que la ruta sea válida y guarda el modelo 3D en la dirección asignada.
Prioridad:	Crítico	
Postcondiciones:	Se muestra el espacio de trabajo con el modelo 3D guardado.	

Tabla 11: *Descripción CU Exportar modelo 3D.*

2.12 Prototipo de Interfaz de Usuario

La Interfaz de Usuario, también conocida por sus siglas en inglés como GUI (Graphical User Interface), es la encargada de mediar la interacción del usuario con el software de una forma fácil, obteniéndose

resultados mejores y llegando a una solución más viable. La misma debe ser sencilla para que el usuario logre, con un mínimo de conocimiento, llegar a la solución deseada. A continuación se presenta una posible interfaz de la herramienta propuesta y se da una descripción de sus funcionalidades.

Figura 19: *Prototipo de Interfaz de Usuario.*

Menú Importar

La opción Importar se utiliza para proceder a la carga del archivo .mesh a editar. En el cuadro de texto File Name el diseñador especifica la dirección y el nombre del fichero a cargar. Al presionar el botón Ok se procede a la carga del fichero especificado.

Menú Exportar

La opción Exportar presenta un cuadro de texto en el cual el diseñador de la aplicación especificará la ubicación física y el nombre con el cual guardará el fichero .mesh, el cual guarda la información del personaje editado previamente.

Conclusiones parciales

Se puede apreciar que con la utilización de la biblioteca de clases QT y del Motor Gráfico Ogre 3D, en conjunto con el desarrollo de la aplicación con el lenguaje C++, todos multiplataforma, se creará un software adaptable a las necesidades según la conveniencia del cliente. En el presente capítulo se definió qué espera el usuario con este sistema, para ello quedaron establecidos sus Requisitos Funcionales y No Funcionales, se presentó el Diagrama de Casos de Uso del sistema y se describieron los Casos de Uso que permitirán al usuario obtener los resultados esperados.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

Introducción

En este capítulo se profundizará en el desarrollo del sistema, se encierra el diseño y se da paso a la implementación del mismo. En la primera parte de este capítulo se abordan los temas relacionados con el Diseño del Sistema, identificado con los Diagramas de Clases de Diseño y además se exponen los Diagramas de Secuencia de la realización de los Casos de Uso correspondientes. En la segunda parte se pasa a la Implementación del Sistema y se muestran los Diagramas de Despliegue y de Componentes y, finalmente, se realizará un análisis de los resultados del sistema con respecto al rendimiento y cumplimiento de los objetivos propuestos.

3.1 Modelo de Diseño

El Modelo de Diseño es un modelo de objetos que describe la realización de los Casos de Uso y sirve como una abstracción del Modelo de Implementación y de código fuente, ya que es utilizado como una entrada esencial en las actividades de implementación y prueba. Se emplea además para concebir, en adición a la documentación, el diseño del sistema del software. De manera general, es un abarcador y compuesto artefacto que encapsula todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.

3.1.1 Diagramas de Clases del Diseño del Sistema.

Una clase es la descripción de un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones, atributos y significado semántico. Una clase de diseño representa una abstracción de una o más clases en la implementación de un sistema. Exactamente la correspondencia depende del lenguaje de implementación, al igual que su tamaño y sus objetos. Las clases definen objetos, los cuales a su vez implementan los Casos de Uso; si las mismas son buenas o no dependen profundamente del entorno de implementación. De manera general, las clases deben mapearse en un fenómeno en particular: en el lenguaje de implementación y deben estar estructuradas de manera que representen los resultados en un buen código. Aun cuando las peculiaridades del lenguaje de

implementación influyen en el modelo de diseño, en el presente trabajo de diploma se ha tratado de mantener una estructura de clases fácil entendimiento y modificación. A continuación se muestra el Diagrama de Clases de Diseño del sistema propuesto.

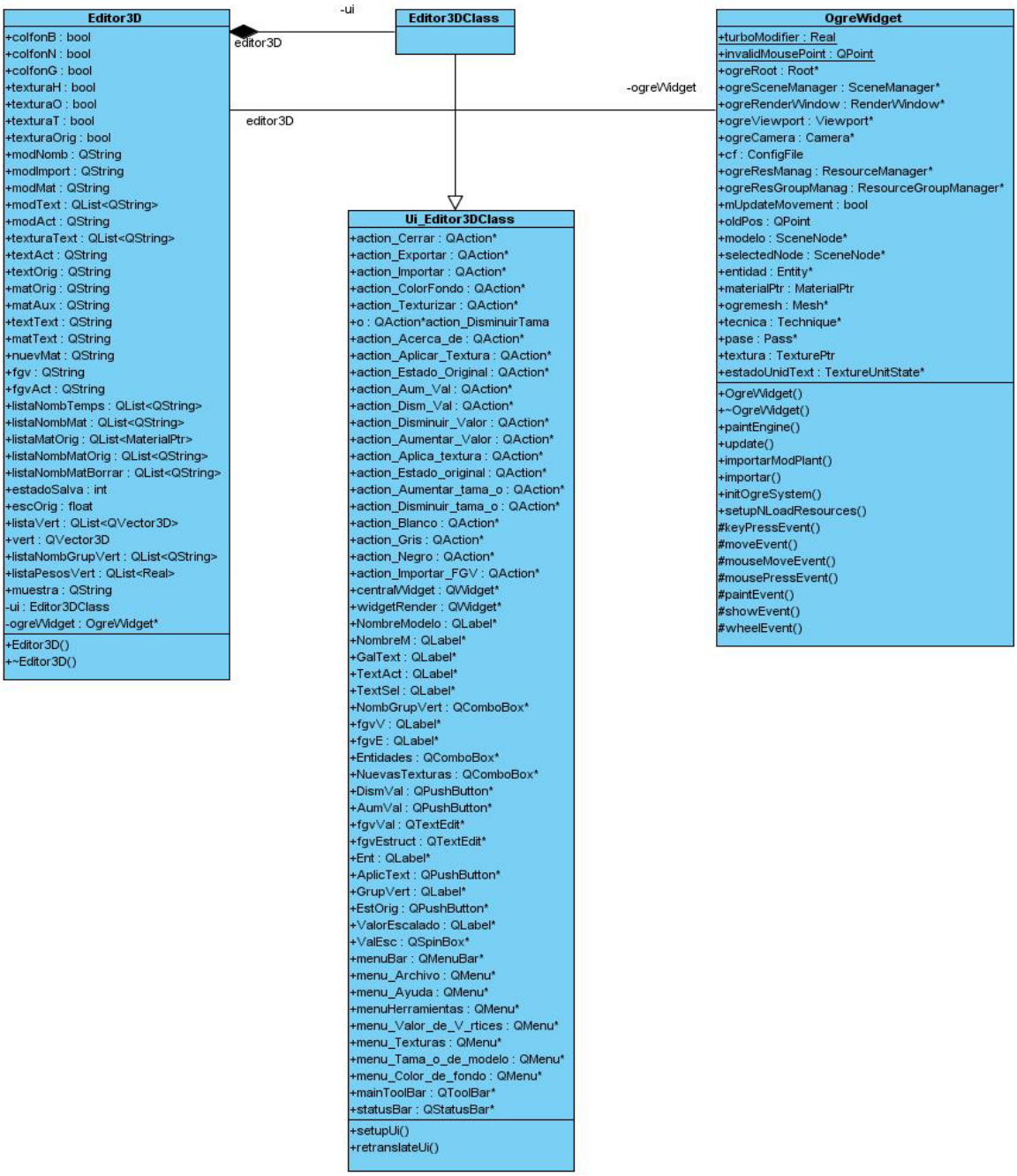


Figura 20: *Diagrama de Clases del Diseño.*

3.1.2 Diagramas de Interacción.

Los Diagramas de Interacción describen el modo en el que cada operación, detectada en los Diagramas de Secuencia, lleva a cabo sus responsabilidades y modifica el estado del sistema. En UML los Diagramas de Interacción pueden representarse a través de los Diagramas de Colaboración y/o de los Diagramas de Secuencia, ambos son representaciones alternas de interacciones. Los Diagramas de Secuencia muestran interacciones entre objetos basadas en el tiempo y los Diagramas de Colaboración muestran cómo los objetos se asocian unos con otros.

El tipo de diagrama seleccionado para construir los Diagramas de Interacción fue el de secuencia, debido a que ellos muestran cómo los objetos se comunican unos con otros en una secuencia de tiempo, así como lo que sucede en cada momento, conteniendo para ello, los objetos con sus ciclos de vida y los mensajes que se envían entre ellos ordenados secuencialmente. Los Diagramas de Secuencia son particularmente importantes para los diseñadores, pues clarifican los roles de los objetos en un flujo, facilitando así una entrada básica para determinar las responsabilidades de las clases y las interfaces. A diferencia de un Diagrama de Colaboración, un Diagrama de Secuencia incluye secuencias cronológicas, pero no incluye relaciones entre objetos. A pesar de que expresan información similar pero mostrada de diferentes maneras, los Diagramas de Secuencia evidencian una secuencia explícita de mensajes y son mejores cuando es importante visualizar el tiempo en que son ordenados los mismos. A continuación se muestran los Diagramas de Secuencia de Diseño del sistema por cada caso de uso:

3.1.2.1 Diagramas de Secuencia de Diseño del CU Importar modelo 3D.

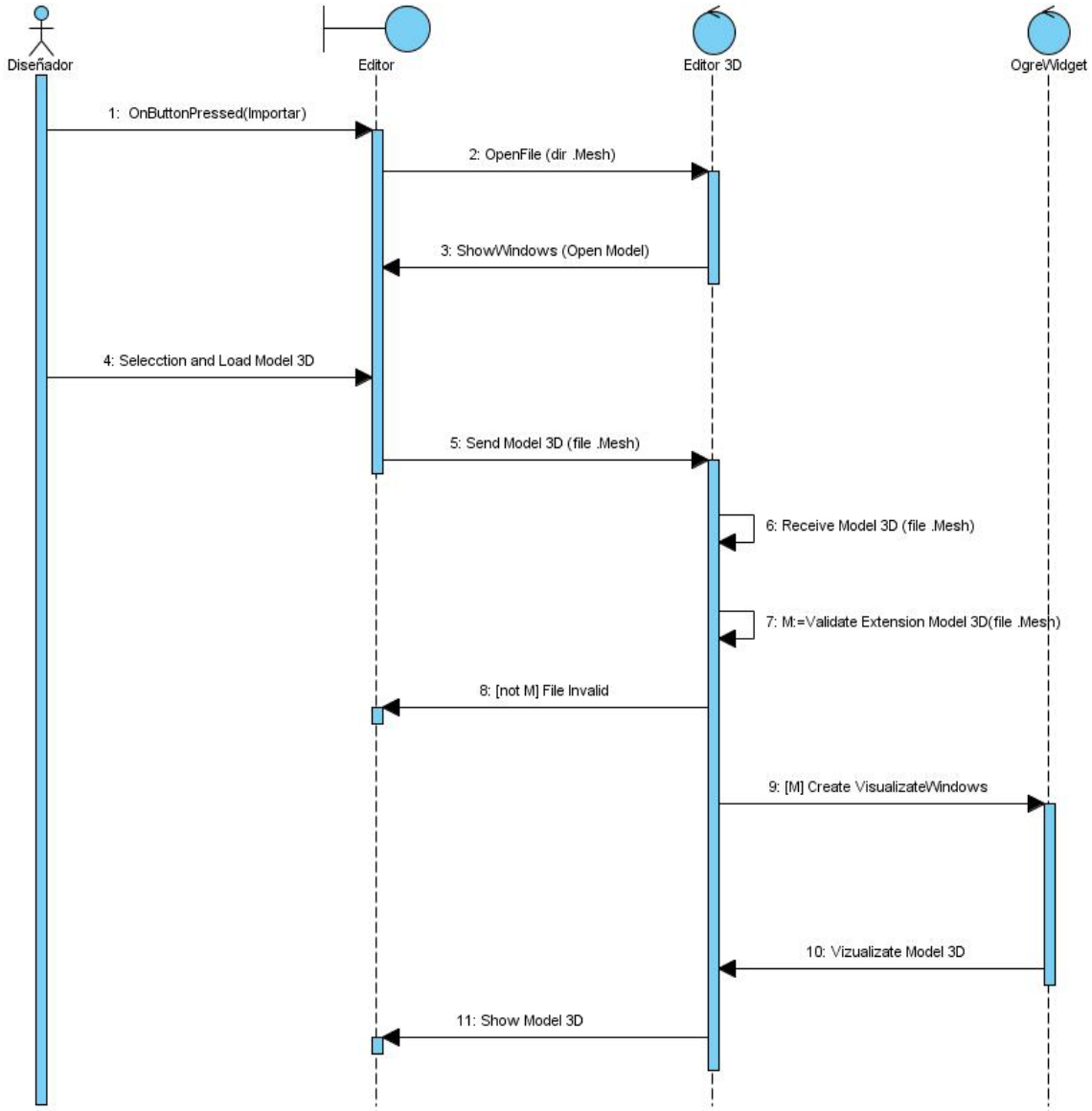


Figura 21: *Diagrama de Secuencia del CU Importar Modelo 3D.*

3.1.2.2 Diagramas de Secuencia de Diseño del CU Importar Fichero Grupo de Vértices.

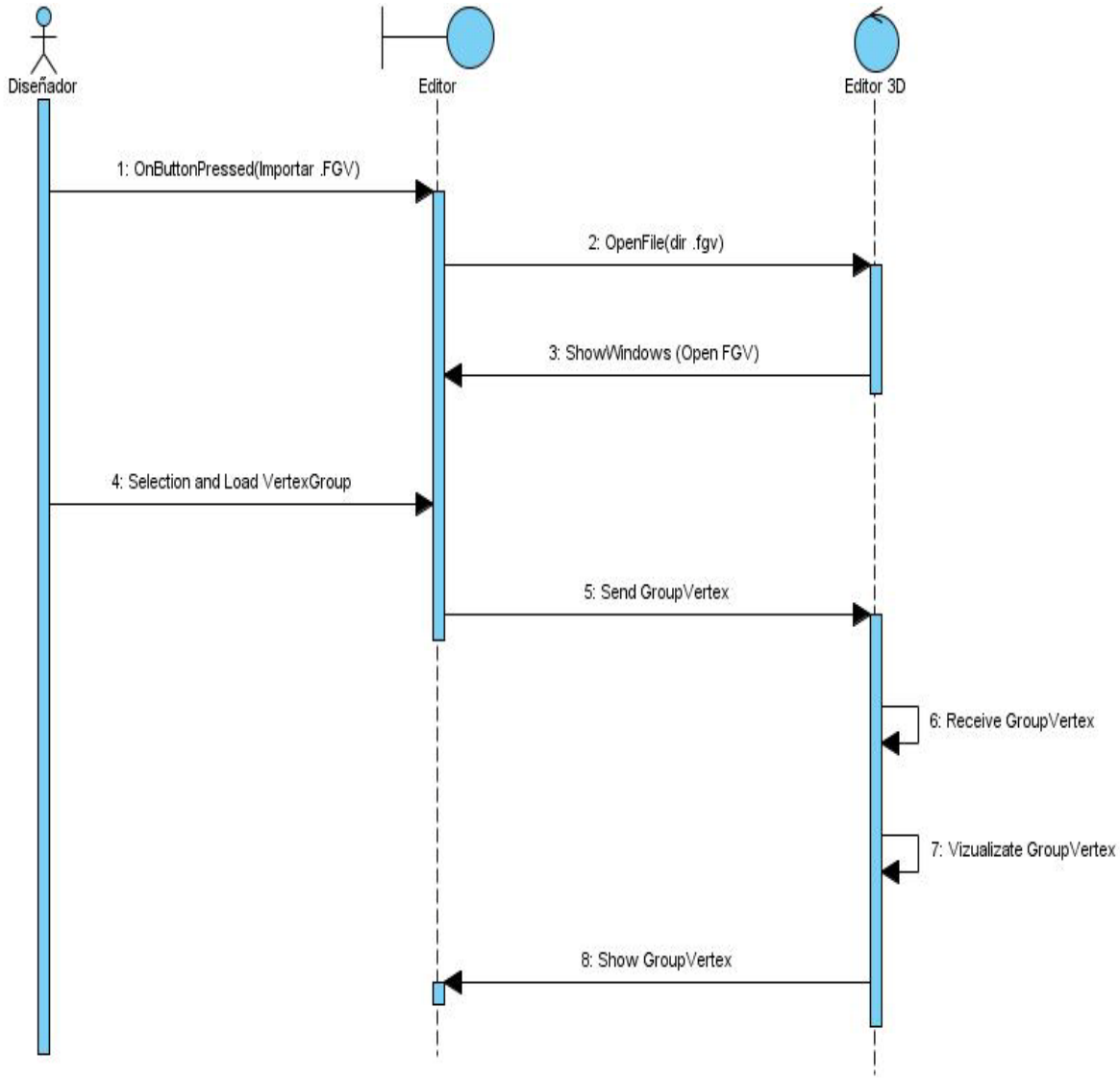


Figura 22: *Diagrama de Secuencia del CU Importar Fichero Grupo de Vértices 3D.*

3.1.2.3 Diagramas de Secuencia de Diseño del CU Editar personaje 3D.

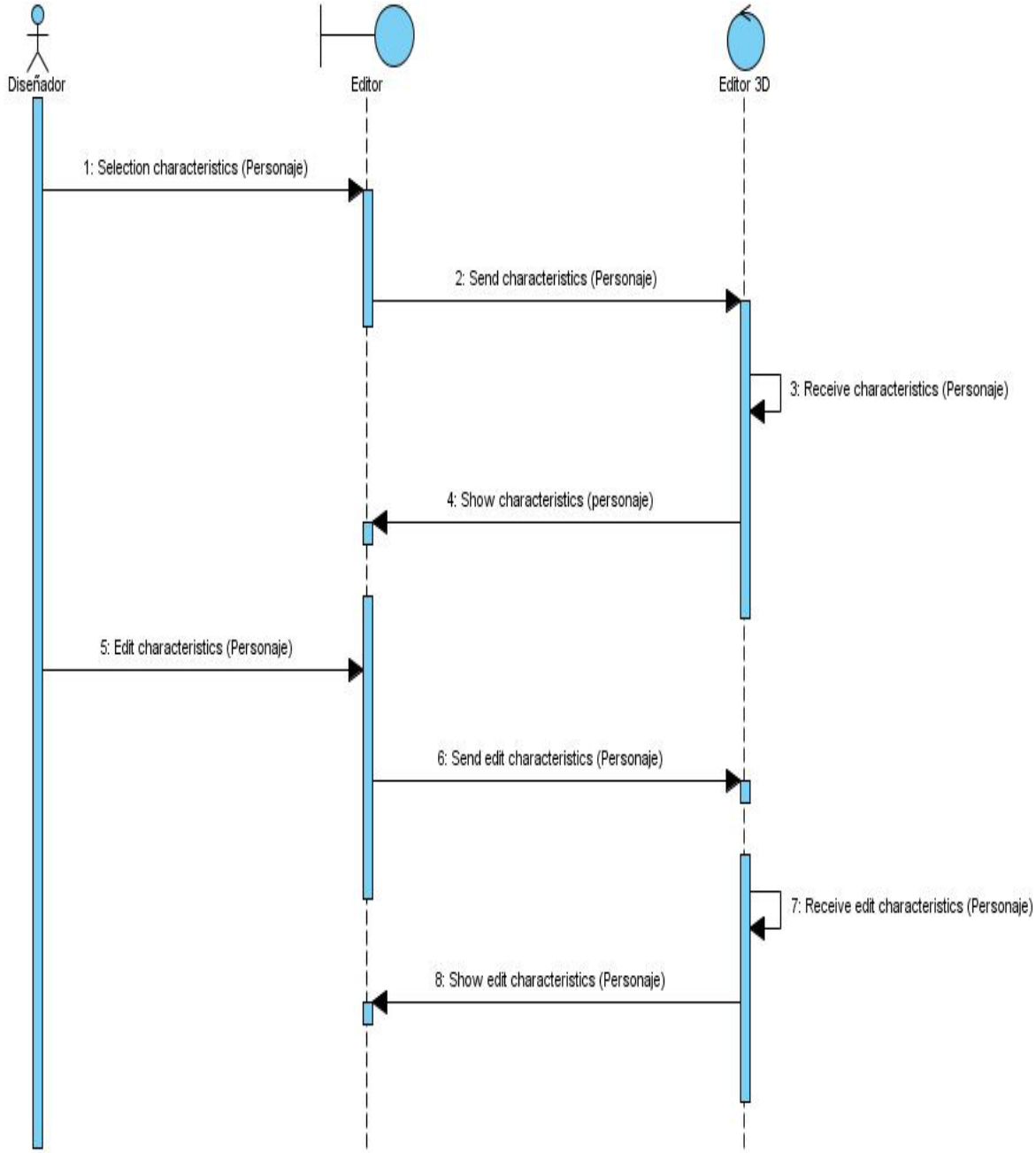


Figura 23: *Diagrama de Secuencia del CU Editar Personaje 3D.*

3.1.2.4 Diagramas de Secuencia de Diseño del CU Exportar modelo 3D.

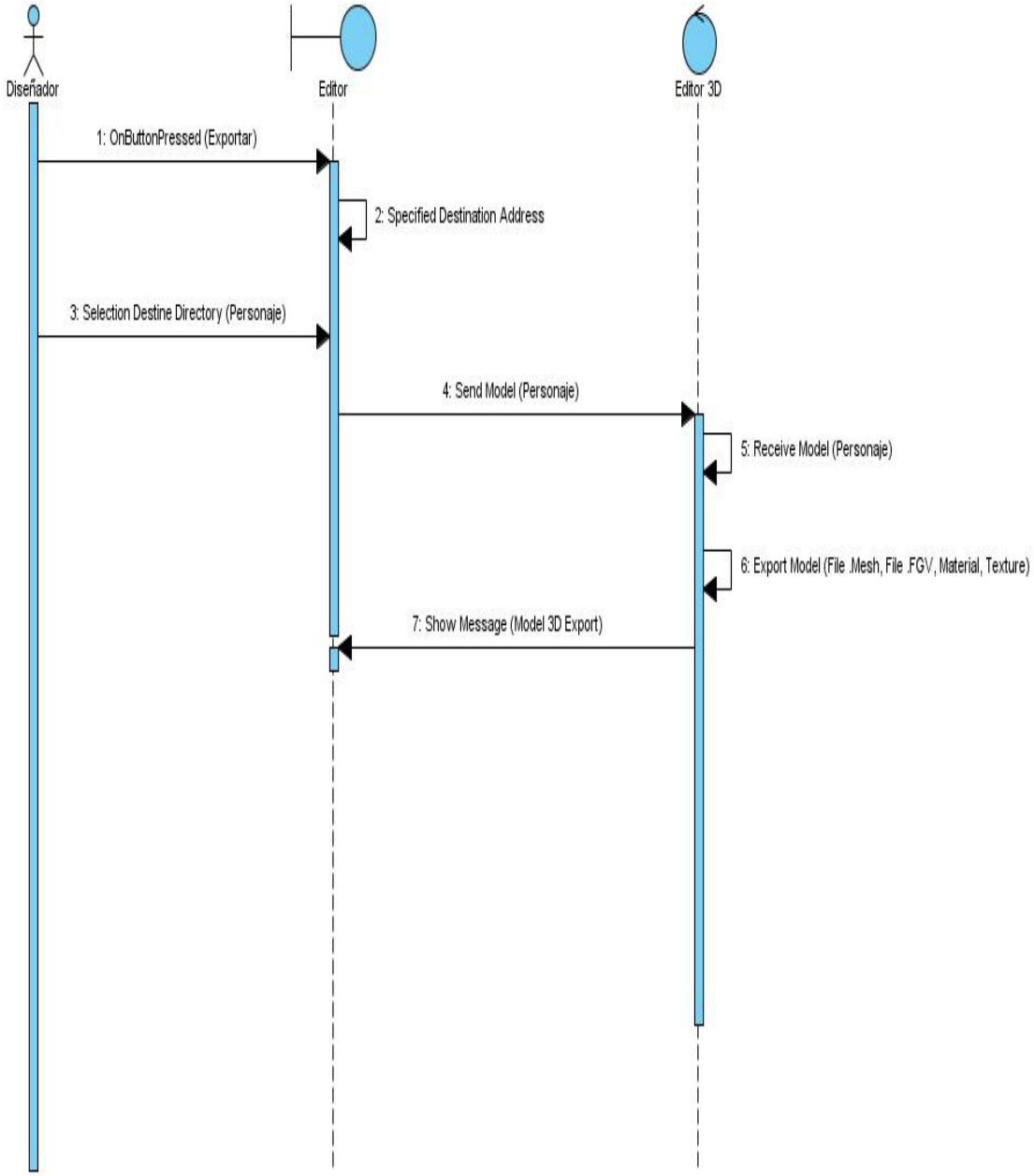


Figura 24: *Diagrama de Secuencia del CU Exportar Modelo 3D.*

3.2 Descripción de las clases de diseño.

En la descripción de las clases se muestran, dan a conocer y quedan plasmados los atributos y métodos que tiene una clase, brindando la información necesaria para lograr un completo entendimiento de la misma. En esta sección se proporciona la descripción de la clase Editor3D, que constituye la principal clase del modelo. Los métodos de acceso a miembros de clases (set y get) no son especificados en las descripciones, puesto que su funcionalidad es sencilla.

Nombre:	Editor3D
Tipo de Clase:	Controladora
Atributo	**Tipo**
colfon	bool
textura	bool
modNomb	QString
modImport	QString
modMat	QString
modText	QList<QString>
modAct	QString
texturaText	QList<QString>
textAct	QString
textOrig	QString
matOrig	QString
matAux	QString
textText	QString
matText	QString
nuevMat	QString
fgv	QString
fgvAct	QString
listaTextExport	QList<QString>
listaNombTemps	QList<QString>

listaNombMat	QList<QString>
listaMatOrig	QList<Ogre::MaterialPtr>
listaNombMatOrig	QList<QString>
estadoSalva	int
escOrig	float
listaVert	QList<QVector3D>
vert	QVector3D
listaNombGrupVert	QList<QString>
listaPesosVert	QList<Ogre::Real>
muestra	QString
Responsabilidades:	
Nombre:	**Descripción:**
Editor3D(QWidget *parent = 0, Qt::WFlags flags = 0)	Constructor de la clase
inicApp()	Inicializa la aplicación y a todos los componentes de la misma.
conexiones()	Conecta la interfaz de la aplicación con la clase controladora.
importar()	Carga el modelo 3D seleccionado.
impFGV()	Importa el fichero grupo de vértices del modelo.
exportar()	Guarda todo los cambios del modelo 3D.
colorFondo()	Cambia el color de fondo del componente viewport.
texturizar()	Agrega una nueva textura a la galería.
AplicarTextura()	Aplica la textura seleccionada al modelo 3D.
mostrarTextura()	Visualiza una vista previa de la textura seleccionada.
estadoOriginal()	Devuelve al modelo 3D su estado original.
AumentarTam()	Aumenta la escala del tamaño del modelo.
DisminuirTam()	Disminuye la escala del tamaño del modelo.

escalarNuevMod()	Lee el tamaño del modelo nuevo y aplica este.
aumValFGV()	Aumenta el valor de los vértices del modelo.
dismValFGV()	Disminuye el valor de los vértices del modelo.
EliminarTemps()	Vacía la carpeta Temporales.
Cerrar()	Cerrar la aplicación.
acercaDe()	Muestra una breve reseña sobre la aplicación.
maybeSave()	Muestra un mensaje de alerta para salvar los cambios.
closeEvent(QCloseEvent *event)	Captura el cierre de la aplicación por el icono(x) de la barra de título.

Tabla 12: *Descripción de la clase controladora Editor3D.*

3.3 Patrones de Diseño

Un patrón es la descripción etiquetada de un problema de la solución, de cuándo aplicar la solución y la manera de hacerlo dentro de otros contextos **[18]**. Los Patrones de Diseño describen un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software y entregan una buena solución ya probada. Esto ayuda a diseñar correctamente y en menos tiempo, a construir problemas reutilizables y extensibles y facilita la documentación. En el presente trabajo se estudiaron y aplicaron los siguientes patrones de diseño que pertenecen al grupo de los famosos 23 patrones de Gang of Four (GOF).

3.3.1 Strategy

El patrón strategy (estrategia) está orientado a resolver situaciones en las que se origina el problema base de existir diversas estrategias para abordar un mismo problema. En este sentido, el problema define una interfaz que será implementada de diversas formas **[19]**. Cualquier programa que ofrezca un servicio o función determinada, que pueda ser realizada de varias maneras, es candidato a utilizar este patrón. Puede haber cualquier número de estrategias y cualquiera de ellas podrá ser intercambiada por otra en cualquier momento, incluso en tiempo de ejecución. Este patrón fue implementado en el proceso de carga de los ficheros, en el cuadro de diálogo ¨ Importar ¨, el diseñador pasa la dirección y el nombre del fichero

a cargar y en dependencia de la extensión del mismo, la aplicación decide qué estrategia de carga aplicar. El funcionamiento de este patrón es muy simple y el añadir nuevas estrategias al programa es muy sencillo y apenas implica modificación de código alguna.

3.3.2 Command

Encapsula una petición como un objeto, permitiendo parametrizar las solicitudes de los clientes y soportando, además, la cancelación de operaciones **[20]**. En el caso de la aplicación, se hizo necesario emitir una petición a un objeto sin que éste conociera la operación solicitada o el receptor de la petición. Los botones y los menús de las barras de herramientas llevan a cabo solicitudes en respuesta a las entradas del diseñador, pero éstas no pueden implementar explícitamente las solicitudes en un botón o menú, porque solo las aplicaciones que utilicen la barra de herramientas deberían conocer qué hacer para cada objeto. Además, el diseñador de la misma no tiene forma de saber el receptor o la operación que se llevará a cabo en cada caso. Con la utilización de este patrón se potencia la reutilización. Este patrón fue utilizado para la construcción de la GUI de la aplicación, debido a que el mismo permite independizar la parte de la aplicación que invoca las órdenes de la implementación de los mismos. De esta forma se garantiza que la parte de presentación sea independiente de la parte lógica de la aplicación.

3.3.3 Singleton

El patrón Singleton es uno de los más sencillos, su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella **[20],** por lo que si más de un objeto necesita utilizar una instancia de dicha clase, esos objetos comparten la misma instancia de la clase Singleton. En el caso de la aplicación se necesitó asegurar que una clase tuviera una única instancia y proveer un punto global para acceder a ésta. ¿Cómo asegurar que esa clase tuviera una única instancia y que al mismo tiempo fuera fácilmente accesible? Una variable global hace a un objeto accesible, pero no limita la instanciación de múltiples objetos de un tipo. Una mejor solución fue hacer que la clase misma fuera responsable de garantizar su única instancia, interceptando solicitudes para crear nuevos objetos y facilitando además una vía para acceder a la instancia.

3.4 Tratamiento de Errores

Para el tratamiento de Errores en el presente sistema, se parte de la idea de que una aplicación bien diseñada debe disminuir la posibilidad de cometer errores. En el sistema, el tratamiento de Errores estuvo enfocado principalmente a errores producto de la interacción de los Diseñadores 3D con el sistema. Se trata en todo momento de minimizar la posibilidad de ocurrencia de errores, aprovechando las posibilidades de la Interfaz Gráfica, para lo cual se le proporcionó la opción de elegir o seleccionar la información que se conoce, lo cual facilitará la entrada de datos y la rapidez de la misma. Evidentemente los errores ocurrirán incluso con los diseñadores de más habilidad y experiencia. En el caso de que los datos sean adicionados de forma incorrecta, se realiza una validación de estos mediante funciones que garantizan que sean válidos; en caso que ocurra un error, se presenta una caja de diálogo con el mensaje donde se describe el mismo. Al obtener la confirmación de lectura del mensaje de error, la caja del diálogo desaparece y continúa la ejecución de la aplicación.

3.5 Modelo de Implementación

En el Modelo de Implementación se describe cómo los elementos del Modelo de Diseño se deben implementar en términos de componentes y cómo se organizan éstos de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizado; así como la dependencia entre los componentes. Esto se realiza a través del Diagrama de Despliegue y del Diagrama de Componentes.

3.5.1 Diagrama de Despliegue

El Diagrama de Despliegue es un grafo de nodos unidos por conexiones de comunicación, que muestra las relaciones físicas entre los componentes de hardware que forman la topología sobre la que se ejecuta el sistema y la distribución de sus partes. Éste refleja tanto la distribución del sistema con los nodos físicos (ordenadores) como la correspondencia que tienen los componentes con los nodos. El Diagrama de Despliegue analizado en este documento es muy simple, sólo incluye la representación de una PC, porque la cual es una aplicación de escritorio que solo será usada por el Diseñador y va a estar representado por un solo nodo.

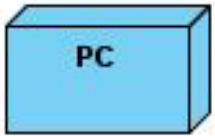


Figura 25: *Diagrama de Despliegue.*

3.5.2 Diagrama de Componentes

Un componente es una parte modular de un sistema, desplegable y reemplazable que típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios y demás). Es un elemento de implementación que representa algo físico, ya sea ficheros o archivos y son creados para poner el código, ya sea código fuente, código binario o código ejecutable. Los Diagramas de Componentes son utilizados para estructurar el Modelo de Implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Es un diagrama que muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Las clases resultantes del diseño se hacen físicas mediante componentes. A continuación se muestra como se agruparon todos los componentes según las clases que éstos contienen.

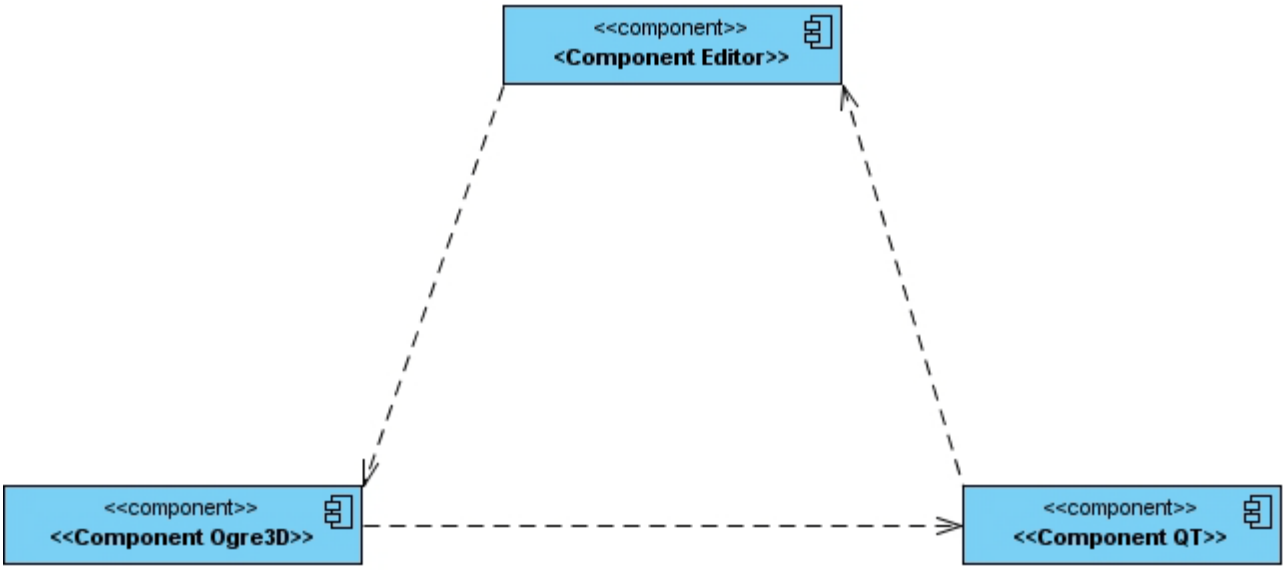


Figura 26: *Diagrama de Componentes.*

3.5.2.1 Diagrama del Componente "Editor"

Figura 27: *Diagrama del Componente "Editor".*

3.5.2.2 Diagrama del Componente "Ogre3D"

Figura 28: *Diagrama del Componente "Ogre3D".*

3.5.2.2 Diagrama del Componente "QT

***Figura 29**: Diagrama del Componente "QT".*

3.6 Estándares de codificación

Los Estándares de Codificación son reglas específicas a una lengua que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Los Estándares de Codificación no destapan problemas existentes, evitan más bien que los errores ocurran. Los bugs frecuentes en programas pueden ser detectados mucho antes o pueden ser incluso evitados totalmente. Durante el desarrollo, los Estándares de Codificación ayudan a los ingenieros a producir un código de alta calidad y a entender y utilizar el código de sus colegas. Pero también realzan considerablemente la capacidad de mantenimiento y rehúso a largo plazo del producto final. Tal práctica del control de bugs en el proceso del desarrollo mejora la calidad mientras que reduce el tiempo de desarrollo, el costo y el esfuerzo.

3.6.1 Reglas de Codificación

Constantes

***Regla*:** Los nombres de constantes se escriben con letras en mayúscula. En caso de estar compuesta por más de una palabra, éstas se separan por un carácter "_".

#define MAXSPEED 1.800f

#define OGREWIDGET_H

Variables

***Regla*:** Las variables se escriben con minúsculas, las variables con nombres compuestos, comienzan con la primera palabra enteramente en minúscula y el resto comenzando con mayúscula.

QVector3D vert

QList<QString> texturaText

Parámetros (argumentos) de métodos

***Regla*:** Cumplen con la misma regla de las variables.

void wheelEvent(QWheelEvent *e)

Clases

***Regla*:** Los nombres de clases comienzan con mayúsculas, con las palabras que la forman en minúsculas y separadas por mayúsculas.

class Editor3D

Métodos miembros de clases

***Regla*:** Los métodos miembros de clases siguen la notación idéntica que para las variables, es decir, comenzando con minúscula y separando las palabras con mayúsculas.

void OgreWidget::keyPressEvent(QKeyEvent *e)

Nombres de enumerados

***Regla*:** El nombre del tipo de enumerador cumple con la regla de las clases y los valores con la misma notación que las constantes.

// No usamos ningún enumerado en la aplicación.

Comentarios

Regla: Se emplean los comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos.

// Esto es un comentario.

3.7 Resultados

Con el desarrollo del Editor de Personajes 3D propuesto se obtuvo como resultado una herramienta que permite que el proceso de edición de personajes, del proyecto Escenario 3D de la Línea de Diseño, se

realice fácilmente, con un mínimo de conocimientos y permitiendo, a su vez, que los Diseñadores de dicho proyecto editen sus propios personajes. A continuación se muestran imágenes de dichos resultados:

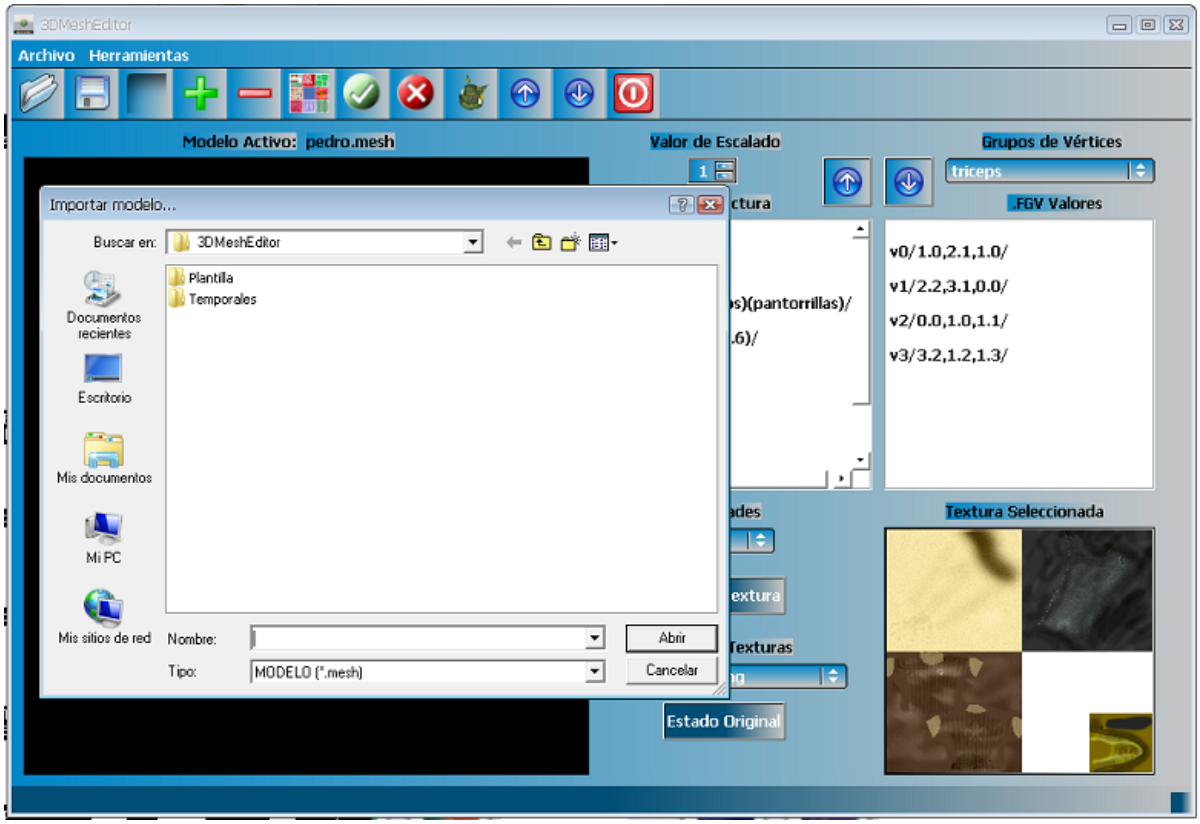


Figura 30: *Importar Modelo 3D.*

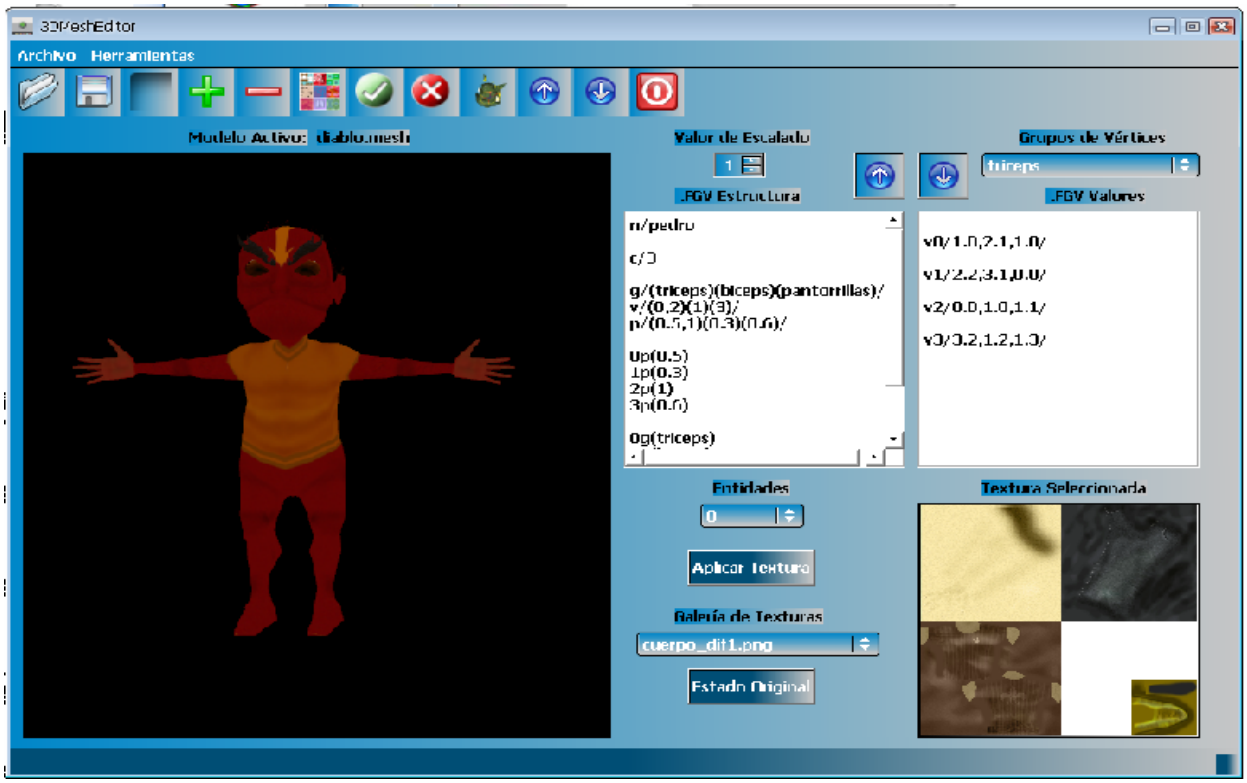


Figura 31: *Nuevo Modelo 3D.*

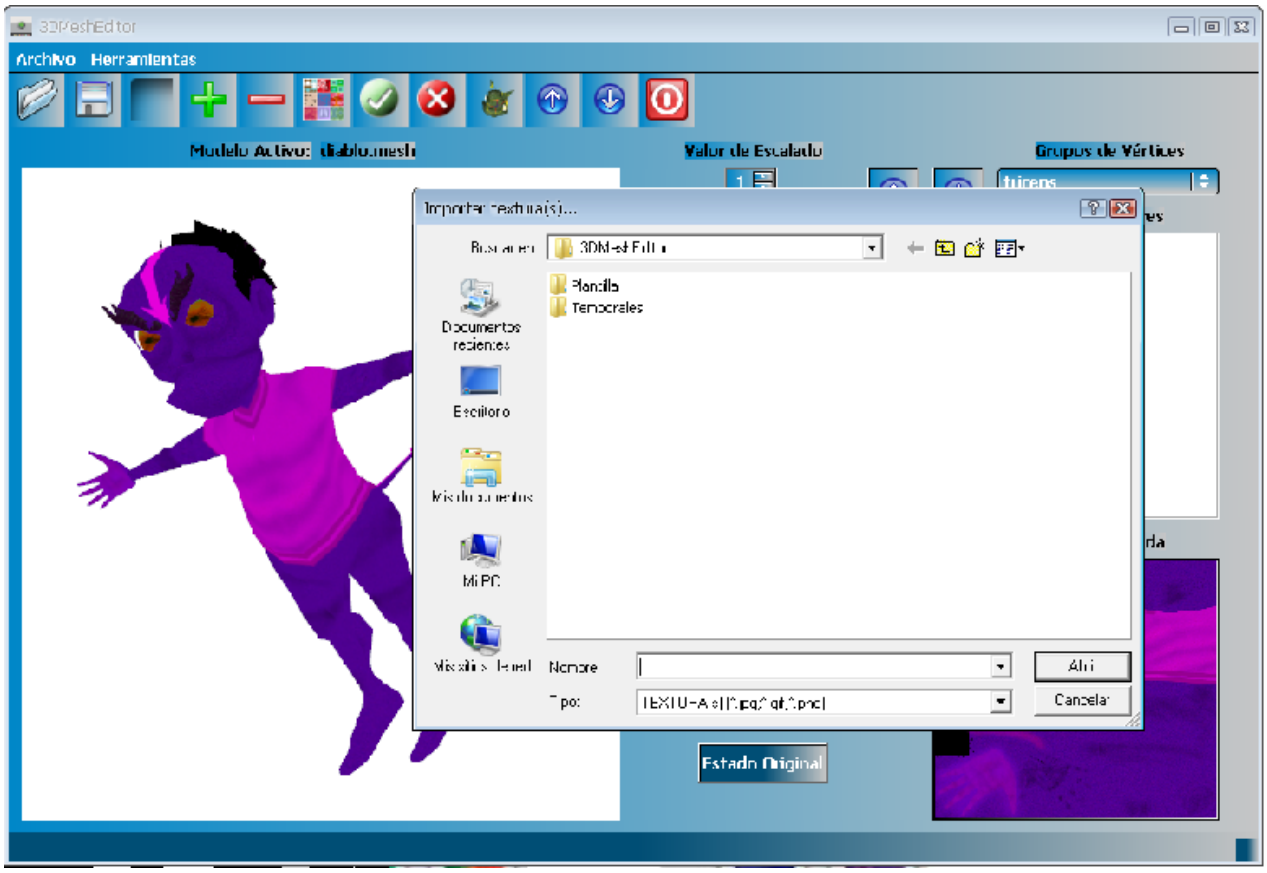


Figura 32: *Importar Textura(s).*

Figura 33: *Aplicar Textura.*

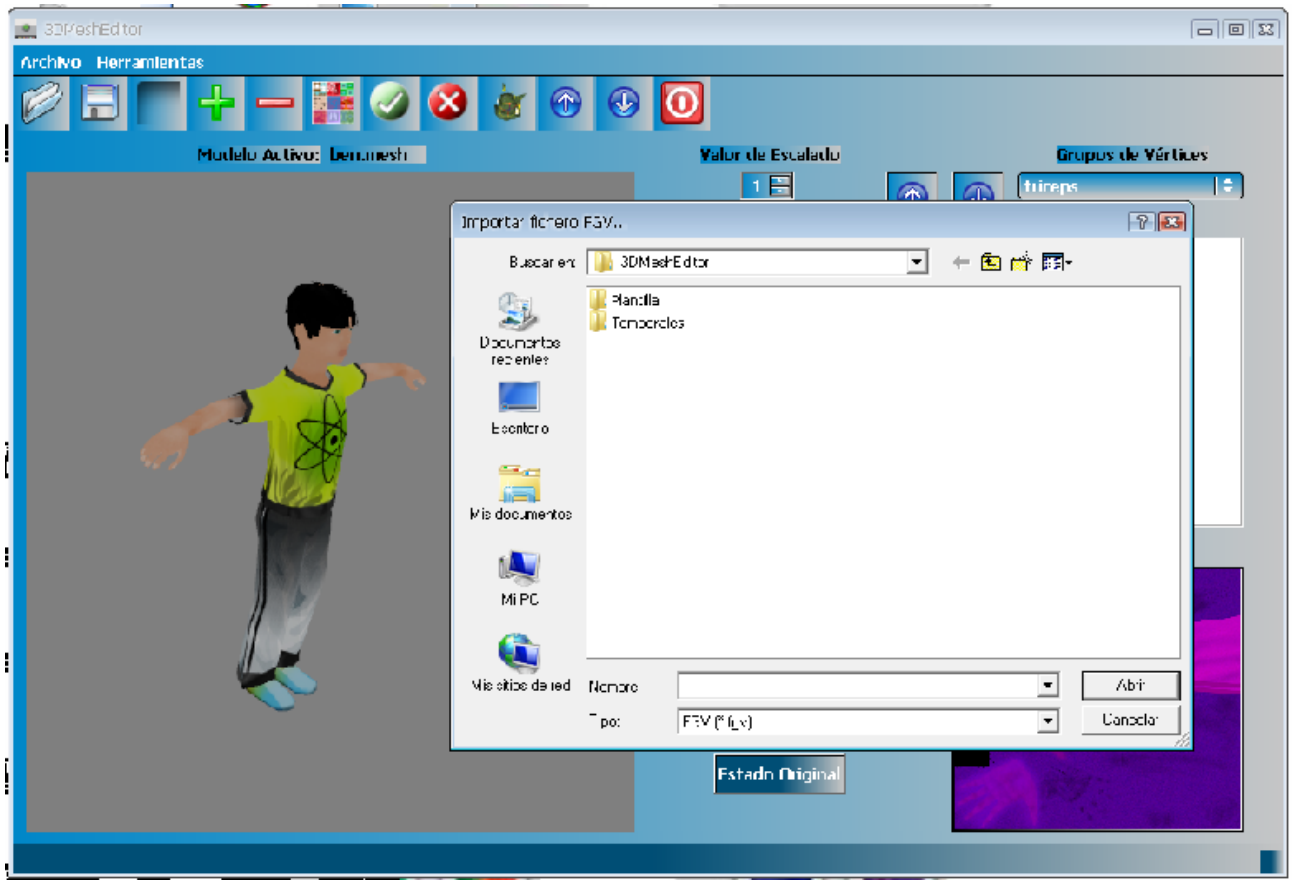


Figura 34: *Importar FGV.*

Conclusiones parciales

A lo largo de este capítulo se mostraron los artefactos necesarios pertenecientes a las etapas de Diseño e Implementación. Después de ser concebido y detallado el Diseño se pasó a detallar cómo se harán físicas las Clases de Diseño, consolidando el proceso de desarrollo para pasar a la programación de los Casos de Uso. Con los Diagramas de Clases y de Componentes se ofrece la posibilidad al programador de generar el código en el lenguaje definido mediante alguna herramienta Case como lo es el “Visual Paradigm”. Finalmente se muestra como quedó la aplicación después de implementada.

CONCLUSIONES

Para el cumplimiento de los objetivos de este proyecto, en concordancia con las necesidades de la Línea de Diseño, fue necesario primeramente realizar un estudio de las técnicas, tecnologías y tendencias actuales en cuanto a las características de los formatos de ficheros 3D, así como las diferentes herramientas de diseño y editores de personajes 3D existentes en la actualidad. En el mismo se analizaron las características de los motores gráficos 3D más usados y sus principales ventajas y desventajas.

Seguidamente se realizó el proceso de Ingeniería del Software utilizando el Proceso Unificado del Software (RUP) como metodología de desarrollo, posteriormente se llevó a cabo la captura de los requisitos funcionales y no funcionales, así como la identificación de casos de uso del sistema en conjunto con su descripción, se comenzaron a elaborar los primeros diseños del modelo, que gradualmente fueron mejorados con el estudio y aplicación de patrones de diseño. Se diseñaron las clases y se elaboró el diagrama de componentes que contendrá las clases del sistema.

Finalmente, se obtuvo una aplicación que cumple con los requisitos planteados, donde el diseñador podrá realizar el proceso de edición de cualquier modelo 3D que se encuentre en formato .mesh. Al mismo tiempo permite variar el tamaño de dichos modelos, cambiar sus texturas y una propuesta para cambiar los grupos de vértices que se toman como partes editables de un carácter, así como exportar dicho modelo 3D.

Es necesario destacar que el proceso de desarrollo del sistema no supuso gastos de recursos, ya que la infraestructura de producción estaba creada. Además, el soporte utilizado para la realización de la aplicación fue totalmente libre y multiplataforma, por lo que no se incurrieron en gastos referentes al pago de licencias y se siguieron las políticas de software libre.

RECOMENDACIONES

Esta herramienta se implementó con el objetivo de editar de forma visual personajes 3D, la carga de los modelos 3D y visualización y edición asociados a los modelos, utilizando la biblioteca gráfica Qt y el motor gráfico Ogre3D. Se recomienda continuar profundizando en estos temas, con el fin de proporcionar más dinámica y realismo a la aplicación. Se proponen las siguientes mejoras a la herramienta como trabajos futuros:

- Desarrollar un plugin para cualquier herramienta de edición 3D (Blender, 3DMax, Maya…) que permita exportar junto con el modelo .mesh su correspondiente fichero de grupos de vértices.
- Fusionar la herramienta con la herramienta Visualizador de Animación no Lineal para personajes 3D, con el propósito de lograr una herramienta más completa y un mejor funcionamiento de la misma.

BIBLIOGRAFÍA

REFERENCIADA

[1]. Gálvez Mozo, A. *Posicionamientos y puestas en pantalla. Un análisis de la producción de sociabilidad en los entornos virtuales*. Barcelona.

[2]. *Autodesk [Consultado en: 2007] "Weapons of Mass Creation"*

Disponible en: http://images.autodesk.com/adsk/files/games_brochure0.pdf

[3]. González Morcillo, Carlos y Vallejo Fernández, David. *Fundamentos de Síntesis de Imagen 3D. "Un Enfoque práctico a Blender". Disponible.*

http://www.esi.uclm.es/www/cglez/fundamentos3D

[4]. archivosPC, [Consultado en: 2010] *"*Estática 3D Editor*" Disponible en:*

http://estatica-3d-editor.archivospc.com

[5]. *Softonic, [Consultado en: 2010] "MakeHuman" Disponible en:*

http://makehuman.softonic.com

[6]. masOportunidades, *[Consultado en: 2010] "Poser" Disponible en:*

http://www.masoportunidades.com.ar/aviso/4809867-poser-8-animacion-3d-personajes- espectacular

[7]. Pipho**, Evan.** *"Focus On 3D Models". Capítulo 7: The 3ds Models. Premier Press. USA. 2003.*

[8]. egModels, [Consultado en: 2010] "Wavefront and Java3D .obj Format" Disponible en:

http://www.eg-models.de/formats/Format_Obj.html

[9]. RoRBook, [Consultado en: 2010] "Meshs/es" Disponible en:

http://wiki.rigsofrods.com/pages/RoRBook/Meshs/es

[10]. **Hawkins, Kevin y Astle, Dave.** *Chapter 20 - Building a Game Engine. OpenGL Game Programming.*

[11]. McGuire, Morgan. *G3D Engine ScreenShots. G3D Engine. [Online] 2007.*

http://g3d- cpp.sourceforge.net/screenshots.html

[12]. Gonzalo Tirado. *Introducción a OGRE: ¿Qué es OGRE? , Características. Enero 2008. Disponible*

[13]. *Free Download Manager, [Consultado en: 2010] "Visual Paradigm for UML" Disponible en:*

http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/

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

[15]. *Microsoft/Express, [Consultado en: 2010] "Visual Studio 2008 Express" Disponible en:*

http://www.microsoft.com/express/

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

[17]. *[Consultado en: 2010 "Qt is a cross-platform application and UI" Disponible en:*

http://qt.nokia.com/

[18]. **Larman, Craig.** *PARTE II FASE DE PLANEACION Y DE ELABORACION. UML y Patrones. Introducción al Análisis y Diseño Orientado a Objeto. Prentice Hall 1999.*

[19]. **Larman, Craig.** *PARTE IV FASE DE DISEÑO. UML y Patrones. Introducción al Análisis y Diseño Orientado a Objeto. Prentice Hall 1999.*

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

CONSULTADA

- **Booch, Grady.** *Object-oriented analysis and design with applications 2da Edición.* Santa Clara, California: s.n., 1998.
- **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *Proceso Unificado de Desarrollo de Software.* 2000.
- *Biblioteca, [Consultado en: 2010] "Tesis" Disponible en: http://Biblioteca.uci.cu*
- *Ogre3D, [Consultado en: 2010] "Motor gráfico Ogre" Disponible en: http://www.Ogre3D.org*

- *The G Files, [Consultado en: 2010] "Motor gráfico Ogre" Disponible en: http://tamudo84.blogspot.com/f2008/01/introduccin-ogre.html*

GLOSARIO DE TÉRMINOS

2D: Dos dimensiones.

3D: Tres dimensiones.

3DS: 3D Studio (file format).

A:

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

API: Una interfaz de programación de aplicaciones o API (del inglés *application programming interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.

*****

B:

Bugs: Error en la escritura de un programa lo que produce un defecto en el sistema.

Binario: En matemática el sistema binario es un sistema de numeración en el que los números se representan utilizando las cifras cero y uno ('0' y '1').

Bounding Box: Recuadro de selección. Para un objeto raster, es el rectángulo más pequeño que encierra completamente todos los puntos pixel que no son completamente transparentes.

*****

D:

Disco Duro: Se llama disco duro o disco rígido al dispositivo encargado de almacenar información de forma permanente en una computadora.

DIRECT X: Es una colección de APIs creadas para facilitar tareas relacionadas con la programación de juegos en la plataforma Microsoft Windows.

*****

E:

Entornos Virtuales: Referido a Mundo virtual. Simulación de mundos o entornos denominados virtuales en los que el hombre interacciona con la máquina en entornos artificiales semejantes a la vida real. Ejemplo de aplicaciones desarrolladas sobre mundos virtuales son los simuladores y los videojuegos.

Editor de Personajes 3D: Herramienta usada para la edición de personajes 3D.

Engine: Componente principal de un videojuego u otra aplicación interactiva con gráficos en tiempo real.

*****

F:

Frame: Cada una de las imágenes que componen una animación.

Framework: Estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos.

Fichero: Bloque lógico de información considerado como una unidad por el usuario. Todos los datos que se almacenan en el ordenador se denominan ficheros. El ordenador puede guardar textos, imágenes, piezas musicales, sonidos y demás. Según su contenido, se almacenan con distintos formatos.

*****

G:

G3D: Graphics Three Dimensional. Engine 3D disponible como código abierto (del inglés Open Source) y libre (del inglés free) con licencia BSD desarrollado en el lenguaje de programación C++. G3D Proporciona una base sólida y altamente optimizada para el desarrollo en general de aplicaciones gráficas y extiende las funcionalidades de OpenGL y sockets incluyendo formatos de modelos 3D y bibliotecas utilitarias.

Gang of Four (Patrones GOF): Es una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular.

*****

H:

Hardware: Son las partes físicas y tangibles de una computadora.

*****

I:

Interfaz de usuario: Es la parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario.

Informática: Es la ciencia que estudia el tratamiento automático y racional de la información.

*****

M:

Modelo: Prototipo para la animación.

Material: Combinación de luces y colores empleadas para definir una apariencia.

Mesh: Término en ingles que significa malla y es un archivo informático utilizado por Ogre 3D.

*****

N:

Nodo: Estructura que representa una posición en un mapa o diagrama.

*****

O:

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

OBJ: Nombre del formato creado por Alias Wavefront.

*****

P:

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

Plugin: Fragmento de funcionalidades que se le agregan a un software.

*****

Q:

Qt: Es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario.

*****

R:

RAM: Memoria de semiconductor en la que se puede tanto leer como escribir información.

Realidad Virtual (RV): Simulación de un medio ambiente real o imaginario que se puede experimentar visualmente en tres dimensiones. La realidad virtual puede además proporcionar una experiencia interactiva de percepción táctil, sonora y de movimiento.

Render ó Renderización: Es el proceso de generar una imagen desde un modelo. La palabra renderización proviene del inglés render. En términos de visualizaciones en ordenador, más específicamente en 3D, la "renderización" es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena 3D.

RUP: Rational Unified Process (Proceso Unificado de Desarrollo). Metodología para el desarrollo de Software.

*****

S:

Shaders: Conjunto de instrucciones gráficas destinadas para el acelerador gráfico que definen el aspecto final de un objeto. Estos determinan materiales, efectos, color, luz, sombra y entre otros.

*****

T:

Textura: Imagen que sirve de “piel” a los modelos 3D.

*****

U:

UML: (Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

*****

V:

Vértice: Es un punto en el espacio dado por tres coordenadas x, y, z.

*****