Universidad de las Ciencias Informáticas

Facultad 5 Realidad Virtual

Exportador de Información de Huesos para el Trabajo con Animaciones en un Entorno Virtual

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor

Lorenzo Domínguez García

Tutor

Ing. Alexis Echemendía González

Ciudad de la Habana

Junio, 2008

OPINIÓN DEL TUTOR

Título: Exportador de información de huesos para el trabajo con animaciones en un entorno virtual.

Autor: Lorenzo Domínguez García

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero en Ciencias Informáticas y propongo que se le otorgue al Trabajo de Diploma la calificación de:

_____

Ing. Alexis Echemendía González

Firma _____

Fecha

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo, y autorizo al Proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual de la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autor:

Lorenzo Domínguez García

_____

Tutor:

Ing. Alexis Echemendía González

_____

DATOS DE CONTACTO

Ing. Alexis Echemendía González.

Graduado de Ingeniería Informática en la Universidad de las Ciencias Informáticas. Actualmente es Profesor Instructor de la dicha Universidad.

E-Mail: aechemendia@uci.cu

DEDICATORIA

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

AGRADECIMIENTOS

Mi más sincero agradecimiento a todos aquellos que han contribuido en la realización de este proyecto, a mis amigos, a mi familia que siempre me ha apoyado en los malos momentos que es cuando más se necesita ese apoyo, a mi hermana por darme la fuerza para no dejarme vencer, a mi tutor y amigo que ha sido el segundo miembro de este trabajo, a todas las personas que han creído y que creen en mi.

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 Hardware gráfico le ha permitido a los desarrolladores alcanzar un mayor grado de realismo, la animación de personajes es uno de los elementos fundamentales dentro de la representación de un entorno virtual, es muy frecuente ver modelos complejos (humanos, animales) reproduciendo animaciones. El objetivo de este trabajo consiste en elaborar una funcionalidad para exportar las jerarquías de huesos conjuntamente con la información de los vértices asociados en un formato de ficheros propio.

Para cumplir con los objetivos de esta investigación se hizo un análisis de las diferentes características de los formatos de ficheros de huesos que más se utilizan en la actualidad, así como las principales ventajas y desventajas que presentan, además se hace un estudio de las diferentes vías que existen para crear una funcionalidad que permita exportar un fichero desde el 3d Studio Max.

Como resultado de este proceso se obtuvo una herramienta que exporta un formato de fichero que optimiza el espacio en memoria, disminuye el tiempo de carga facilitando la manipulación de las animaciones de los personajes en las escenas representadas por la STK, cumpliendo así las necesidades existentes en este aspecto del proyecto de Herramientas de Desarrollo para Sistemas de Realidad Virtual.

Palabras Claves

Herramienta, Plugin, Formato, Fichero, Jerarquía, Huesos, Peso, Vértice, Animación, STK.

ÍNDICE

INTRODUCCIÓN

La realidad virtual es la simulación por computadora, dinámica y tridimensional, con alto contenido gráfico, acústico y táctil, orientada a la visualización de situaciones y variables complejas, durante la cual el usuario ingresa, a través del uso de sofisticados dispositivos de entrada, a "mundos" que aparentan ser reales, resultando inmerso en ambientes altamente participativos, de origen artificial.[1]

El auge de la realidad virtual ha estado precedido de un largo tiempo de intensa investigación:

- En 1958 la Philco Corporation desarrolla un sistema basado en un dispositivo visual de casco controlado por los movimientos de la cabeza del usuario.
- En el inicio de los 60, Ivan Sutherland y otros crean el casco visor HMD mediante el cual un usuario podía examinar, moviendo la cabeza, un ambiente gráfico. Simultáneamente Morton Heilig inventa y opera el Sensorama.
- Para 1969, Myron Krueger creó ambientes interactivos que permitían la participación del cuerpo completo, en eventos apoyados por computadoras.
- En 1984, Michael McGreevy y sus colegas de la NASA desarrollan lentes de datos con los que el usuario puede ahora mirar el interior de un mundo gráfico mostrado en computadora.
- 1989: ATARI saca al mercado la primera máquina de galería de vídeo juegos con tecnología 3D. En ese mismo año Autodesk presenta su primer sistema de realidad virtual para PC.
- Para el año 1995 los simuladores de vuelo, desde los más perfectos, como los que utilizaban Thomson-Militaire o Dassault, hasta los videojuegos para microordenadores son en sí aplicaciones de la realidad virtual, cuyo fin es situar a la persona en situaciones comparables a la experiencia real.

En la actualidad, la realidad virtual se plasma en una multiplicidad de sistemas que permiten que el usuario experimente "artificialmente" diferentes aspectos de la vida real.[1]

Características de la realidad Virtual:

- Responde a la metáfora de "mundo" que contiene "objetos" y opera en base a reglas de juego que varían en flexibilidad dependiendo de su compromiso con la Inteligencia Artificial.

- Se expresa en lenguaje gráfico tridimensional.

- Su comportamiento es dinámico y opera en tiempo real.

- Su operación está basada en la incorporación del usuario en el "interior" del medio computarizado.

- Requiere que, en principio haya una "suspensión de la incredulidad" como recurso para lograr la integración del usuario al mundo virtual al que ingresa.

- Posee la capacidad de reaccionar ante el usuario, ofreciéndole, en su modalidad más avanzada, una experiencia inmersiva, interactiva y multisensorial.

Las aplicaciones de la RV son notables, es el caso de 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; y finalmente la simulación, ejemplo de este ultimo son los numerosos tipos de simuladores desde la conducción de autos hasta los simuladores más complejos de aviación.[1]

En estos últimos años en Cuba se ha visto un notable desarrollo de este campo. Liderando la vanguardia empresas como SIMPRO y la Universidad de las Ciencias Informáticas, específicamente la facultad 5 que debido a su perfil es quien desarrolla esta línea de trabajo, hoy día se labora en varios proyectos entre los cuales se pueden mencionar: simuladores de conducción, simuladores quirúrgicos, desarrollo de juegos entre otros proyectos de esta índole.

Aun teniendo en cuenta el gran desarrollo tecnológico sigue siendo un problema

representar los entornos virtuales con el mayor realismo posible en tiempo real, para lograr el grado de realismo deseado los desarrolladores hacen uso de la mayor parte de los recursos de los ordenadores en la representación grafica, sonido, cálculos físicos y matemáticos, por lo que se debe optimizar en otros elementos, tratando de hacerlos más viables y eficientes es el caso de los diferentes ficheros que contienen la información que posterior mente será representada en el entorno virtual.

Actualmente el proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual no cuenta con una herramienta que exporte un fichero, que contenga la información de los huesos y los vértices asociados, y que a su vez disminuya el tiempo de carga y el espacio en memoria, por lo que esto constituye la situación problémica existente. De aquí surge la siguiente interrogante, constituyendo el problema científico a resolver, ¿Como elaborar una herramienta que genere un formato de ficheros que sea propio, flexible y que se ajuste a las necesidades del proyecto Herramientas de Desarrollo para Sistemas de Realidad Virtual de la facultad 5?

Como objeto de estudio se trabaja en el análisis de los procesos de generación de ficheros de huesos para entornos virtuales y el campo de acción se basa en el estudio de los procesos de generación de ficheros contenedores de la información de los huesos para la puesta en escena de las animaciones de personajes en los entornos virtuales.

El objetivo general que propone este proyecto es elaborar un plugin que permita exportar las jerarquías de huesos y los vértices asociados a esta, desde la herramienta Autodesk 3d StudioMax.

Con el fin de dar cumplimiento a los objetivos planteados se hace indispensable realizar varias tareas:

- Investigar las características de las animaciones por huesos en los entornos virtuales.

- Investigar las características de los distintos formatos de ficheros que soportan la información por huesos más utilizados en la actualidad.

- Investigar los algoritmos que se emplean para la generación y lectura de estos ficheros.

- Elaboración de soluciones técnicas que permitan alcanzar los objetivos propuestos.

El desarrollo de este trabajo estará estructurado de la siguiente manera:

En el primer capítulo, Fundamentación Teórica, se realiza un análisis bibliográfico donde se investigan las características de las animaciones, en particular las realizadas mediante sistemas de huesos, así como un estudio de los distintos formatos de ficheros de huesos más usados en la actualidad. También se analizan las distintas vías que existen para elaborar un plugin a partir de la herramienta 3d Studio Max. En el segundo capítulo Soluciones Técnicas, se exponen las características que presentara el sistema como solución a los problemas planteados. En el tercero, Descripción de la solución propuesta, se analiza con mayor profundidad el objeto de estudio, se crea el modelo del dominio, se realiza la captura de requisitos y se definen los modelos de casos de uso del sistema. Finalmente en el cuarto capítulo Diseño e Implementación del sistema, se presentan los diagramas de clases de diseño, de secuencia y los diagramas de componentes.

Finalmente, se ofrece un glosario de abreviaturas y otro de términos para ayudar a la comprensión del lenguaje técnico utilizado en el desarrollo de este trabajo investigativo

Capítulo 1: Fundamentación Teórica.

Introducción

Debido el constante desarrollo de los entornos virtuales la búsqueda de un mayor realismo y eficiencia en las animaciones que en ellos se representa también se ha incrementado, dando lugar a una serie de técnicas con un mismo propósito: lograr una animación más cercana a la realidad. Dentro de esta amplia amalgama podemos encontrar la animación por huesos, una de las más eficientes y que aporta mayor realismo a las animaciones de personajes en tiempo real. En este capítulo se hará un estudio de los principales ficheros de huesos, la estructura que presentan, sus ventajas y desventajas, así como las técnicas que se utilizan para exportarlos.

1.1 Animación en tiempo real

La animación en tiempo real es aquella en la que se van generando los fotogramas a medida que son necesarios, o sea, no se tiene previamente la información de cada fotograma, sólo se tiene la de los fotogramas necesarios para, mediante cálculos, generar aquellos cuadros de los cuales no se tiene la información completa. Los cambios en la escena pueden estar dados por la interacción con elementos externos, por ejemplo cuando se indica a través de algún periférico la dirección del movimiento u otra acción a realizar.[2]

1.2 Animación por esqueleto.

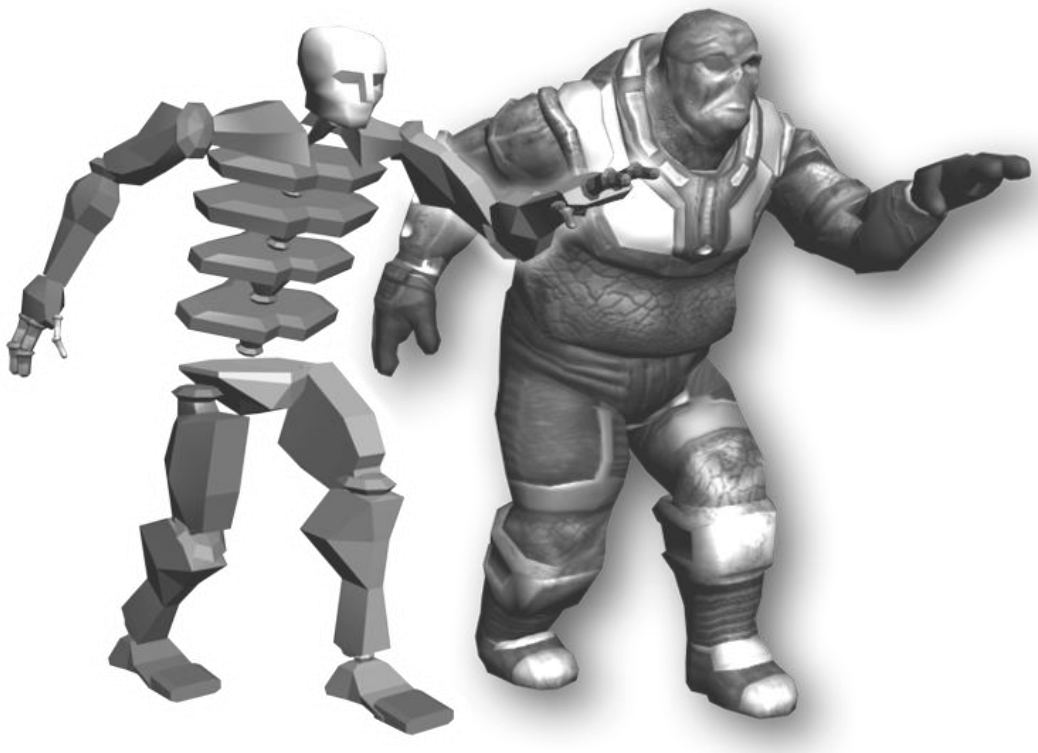


Fig. 1 Ejemplo de animación por esqueleto.

Se incluye dentro de las diversas técnicas de animación que existen en la actualidad y consiste en deformar la malla de un cuerpo a través de una estructura jerárquica de huesos, a los que se les asocia un grupo de vértices. Es una técnica sumamente utilizada actualmente por los programadores, dada su rapidez para procesar y obtener excelentes resultados, y la posibilidad de incluir una gran cantidad de detalles en la animación de un personaje, desde las arrugas de la piel hasta la definición de los músculos de su cuerpo.[3]

Es usada en muchos juegos y cada vez se hacen más perfectas. Uno de los primeros juegos en usar esta técnica fue el Half-Life donde las criaturas tienen un movimiento más fluido y realista que en juegos anteriores.

Es una técnica de animación por computadora que se usa principalmente para la creación de vertebrados virtuales, en donde un personaje está representado en dos partes: una representación superficial usada para dibujar al personaje (piel) y una construcción jerárquica de huesos usada sólo para la animación: a esto se le llama el esqueleto (Fig2).[3]

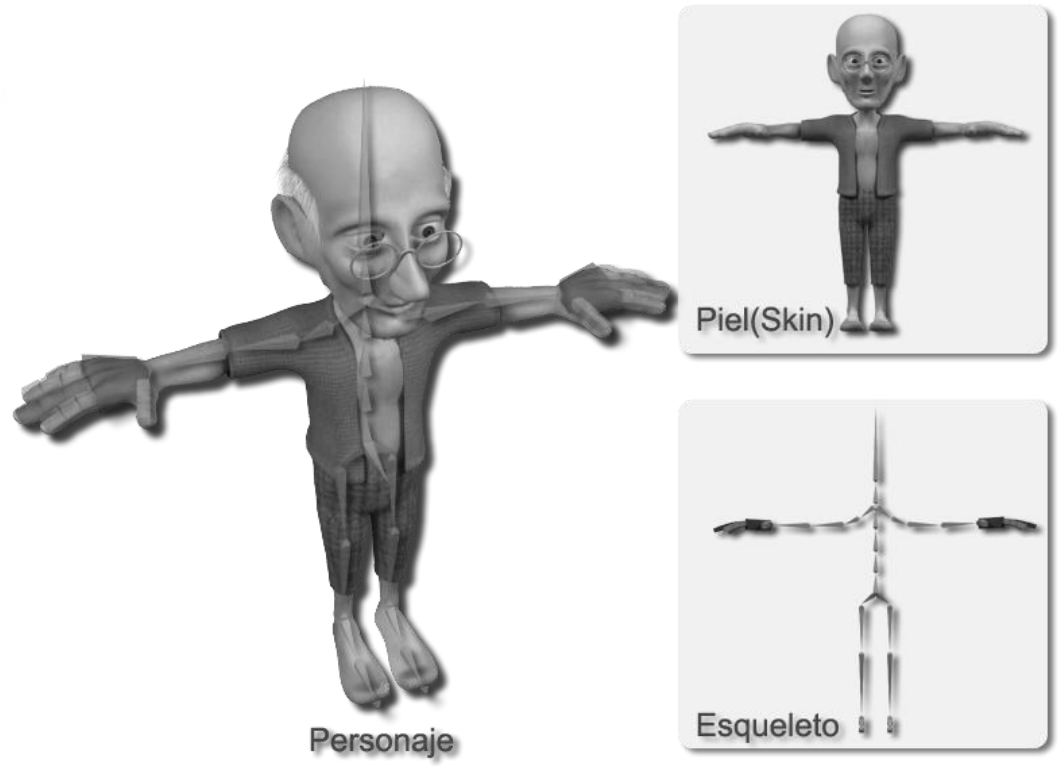


Fig. 2 Relación Esqueleto-Personaje.

Esta técnica se logra con la construcción de una serie de “huesos” los cuales tienen una transformación en tres dimensiones. Esta transformación contiene la posición, escala y orientación además de un hueso opcional padre. De esta manera, los huesos forman una jerarquía. La trasformada completa de un nodo hijo es el producto de su transformada padre y de su propia transformada. Esto se refiere a que al mover el hueso del muslo también moverá el hueso de la pantorrilla, por ejemplo. Conforme el personaje es animado, los huesos cambian su transformación mientras pasa el tiempo, bajo la influencia de algunos controles de animación.[3]

Para una malla poligonal, cada vértice puede tener un peso de mezcla para cada hueso. Para calcular la posición final del vértice, cada transformada de hueso es aplicada a la posición del vértice escalada por su correspondiente peso. Este algoritmo se llama matriz de paleta de piel, porque el conjunto de transformadas de hueso, almacenadas como matrices de trasformadas, forman una paleta para cada vértice de la piel para ser elegidas. [3]

Cada hueso del esqueleto está asociado con alguna porción de la representación visual del personaje. La Malla está asociada con un grupo de vértices; por ejemplo, en un modelo de un ser humano, el hueso del muslo estará asociado con los vértices que hacen el polígono del modelo del muslo. Porciones de la piel del personaje pueden ser asociadas con múltiples huesos, cada uno teniendo un factor de escala llamado peso del vértice, o peso de la mezcla. El movimiento de la piel cercana a las articulaciones: pueden, por lo tanto, ser influenciados por los dos huesos.

La animación esqueletal es útil porque permite a los animadores controlar únicamente aquellas características del modelo que se mueven independientemente. Un personaje no puede mover su parte inferior de la espinilla independientemente de la parte superior. Por el contrario: hace una animación vértice por vértice y sólo se moverá el hueso y los vértices relacionados a éste. La animación esqueletal es una forma estándar de animar personajes u objetos mecánicos por un tiempo prolongado, generalmente más de 100 cuadros. Generalmente es usado en videojuegos y en la industria cinematográfica. También puede ser aplicada a objetos mecánicos y cualquier otro objeto que tenga elementos rígidos y junturas. [3]

1.2.1 Stitching

Con esta técnica los vértices correspondientes a cada hueso son transformados aplicándoles la matriz de transformación del hueso para cada fotograma a mostrar. La deficiencia de esta técnica es que pierde suavidad la malla en las intersecciones de los

huesos, pues al flexionarse una articulación del cuerpo los vértices más cercanos a esta -y que pertenecían a huesos diferentes- se alejan demasiado, lo que crea ángulos abruptos en la zona de la articulación y una sensación de rigidez. [4]

1.2.2 Skinning

Esta técnica es básicamente como la anterior pero soluciona su deficiencia. Los vértices pueden ser influenciados por más de un hueso. Para estos casos cada hueso tiene un valor llamado *weight* (peso), que indica cuánta influencia tiene este hueso sobre el vértice. [4]

1.3 Jerarquía de huesos.

Los esqueletos utilizados para las animaciones poseen una jerarquía determinada que posibilita que la cinemática directa pueda ser utilizada correctamente. Todas las transformaciones que se le apliquen a un elemento de la jerarquía tendrán efecto también en sus hijos. [5]

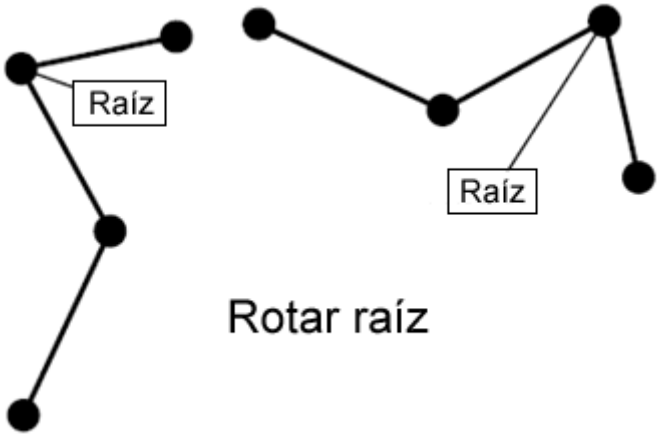


Fig. 3 Jerarquía de elementos.

Para la representación de las figuras humanas, es un estándar utilizar una jerarquía de huesos similar a la del cuerpo humano real. En estas, las caderas son la raíz de la jerarquía y las extremidades los elementos más simples.

1.4 Características que debe tener un formato de fichero de Huesos.

Un fichero de huesos debe ser un contenedor de la información de la jerarquía de huesos y los vértices asociados a esta, debe contar con todos los atributos que permitan conocer la cantidad de huesos que componen el sistema, el peso o afectación que produce un hueso sobre un vértice de la malla del personaje u objeto.

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.

1.5 Características de los formatos de fichero más usados que soportan la animación por Huesos.

1.5.1 FBX

El formato FBX desarrollado por Autodesk, provee un intercambio universal de recursos en 3D. Éste es un formato de creación e intercambio de 3D, independiente de la plataforma y completamente gratuito (pero no libre), admite los principales elementos de datos en 3D, así como elementos en 2D de audio y de medios de video.

FBX es un formato combinado entre binario y código ASCII, lo que le permite soportar, almacenar, convertir y transportar complejas escenas en 3D combinadas con 2D., soporta la animación por esqueleto ya que permite exportar segmentos de esqueleto (raíz, extremidades y nodos de extremidades), Varias tomas en animaciones MOCAP y posiciones enlazadas en una lista de nodos. [3]

1.5.2 DirectX

.X fue creado por Microsoft para DirectX, el mismo posee una arquitectura y un formato de archivo libre de contexto. Es conducido por plantillas y es libre de cualquier uso.

El formato X almacena todo en un método jerárquico. Cada nivel de la jerarquía puede tener cualquier número de “objetos” para mantener datos, pero es recomendable solo tener un número pequeño de “objetos” por cada nivel. El “objeto” principal del modelo de jerarquía geométrica es un cuadro. Para animaciones, el “objeto” principal de la jerarquía es una set de animación. Los ficheros X están compuestos de plantillas, que pueden ser definidas por el usuario. Una plantilla es una "definición de como quiere el usuario que se almacene la información". [3]

Este formato soporta la animación por esqueleto (objetos articulados), en forma de marcos (frames) o referencias, cada frame contiene una FrameTransformMatrix. Las cadenas de cinemática inversa (IK chains) no son parte del formato X, sin embargo la naturaleza del fichero permite incluir nuevas plantillas por lo que se podría incluir fácilmente. [6]

1.5.3 MDL

El formato MDL creado por Valve (Halflife) define la estructura del modelo junto con la animación, bounding box, hit box, materiales y la información de la malla, Halflife puede usar su sistema de huesos para mezclar dos animaciones, por ejemplo: puede usar las piernas para la animación de correr y el torso superior para otras animaciones como apuntar o disparar. [7]

Este fichero contiene referencias a otros ficheros como el SMD que contiene la información del skin y los huesos. Este le dice al engine cuales vértices están asignados a cada hueso.

Una de las principales desventajas de este fichero es que fue creado para utilizarlo con el engine de Halflife esto trae problemas a la hora de crear un personaje nuevo, ya que Valve usa un sistema de huesos extandar (Gordon Skeleton), si se crea un personaje con un sistema de huesos distinto el engine lo escalara además de entrar en conflicto a la hora de cargar las animaciones. [7]

1.5.4 MD4

Formato creado por ID Software inc, para el juego Quake3 anteriormente precedido por MD2, MD3. Lo nuevo que presenta esta versión es que se incluye la animación por esqueleto, estos formatos son muy populares debido a su facilidad de uso.

Mantiene en esencia el mismo formato que las versiones anteriores un bloque cabecera (Header), donde contiene informaciones generales, tales como posiciones de un tipo de información en el fichero, cantidad de atributos que contiene un bloque, entre otras.

1.6 Software para el diseño de entornos virtuales.

1.6.1 Características que presenta la herramienta 3d Studio Max.

El 3D Studio Max (algunas veces llamado 3ds Max o simplemente Max) es un programa de creación de gráficos y animación 3D desarrollado por Autodesk Media & Entertainment. Fue desarrollado como sucesor para sistemas operativos Win32 del 3D Studio creado para DOS. Kinetix fue más tarde fusionada con la última adquisición de Autodesk, Discreet Logic. La versión de noviembre del 2006 es 3ds max 9 en inglés y la 8 en español. [8]

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. 3ds Max es utilizado en mayor medida por los productores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura. [8]

Con más de una década de orientación a los juegos, el software 3ds max de Discreet es utilizado por más del 80% de los principales productores y editores de juegos, por sus funciones específicas, sus productivos paquetes de herramientas y su acceso a la mayor comunidad mundial que más activamente comparte ideas, scripts y otras funcionalidades(plugins).

Posee completas herramientas 3d con múltiples enfoques para abordar la producción a su manera. Optimización integrada con gráficos de alto rendimiento y plataformas específicas. SDK avanzado para un desarrollo personalizado. Foros activos de la Comunidad de Discreet para ayuda con información técnica y cientos de scripts y herramientas gratuitas. [8]

Principales funciones

- Herramientas de modelado para polígonos y parches sin precedentes.
- Completas herramientas para animación de personajes con total IK y Skinning.
- Herramientas de color de vértices basadas en capas.
- Sistema de integrado de partículas orientado a eventos.
- Visión esquemática mejorada desarrollada con las principales compañías de juegos para la gestión de escenas complejas Sombreado Dinámico UI, crea componentes UI amigables para los artistas para Sombreado HLSL, basados en parámetros dentro del archivo DirectX.FX.
- Emulación de datos del motor de juego dentro del puerto de visión de 3ds Max.
- Interfaz de exportación de juegos que facilita el envío de datos a motores específicos.
- Incorpora dos caminos para crear nuevas herramientas (plugins) de trabajo que son el MaxScript y el SDK. [8]

Lenguajes que brinda para el desarrollo de la funcionalidad.

- MaxScript
- SDK(c++)

1.6.2 Características del MaxScript

En general los plugin creados por MaxScript corren más lentos en comparación con los escritos en C++, al punto de que sí el rendimiento es la cuestión, usar el SDK será lo más sugerente.

Por otro lado MaxScritp brinda algunos métodos de alto nivel que pueden ser encontrados en el SDK(C++) de max y presenta nuevas potencialidades que no están presentes en el SDK. Sí la funcionalidad es soportada por el MaxScritp, pero no por el SDK, entonces el MaxScritp será la única opción. En particular el 3ds Max brinda potencialidades vía el OLE/ActiveX que hace más fácil codificar en MaxScript que con el SDK.

El MaxScript puede ser conveniente para crear prototipos de plugins, para desarrollar plugin de simple complejidad y para escribir pruebas de pequeños fragmentos de funcionalidades. [9]

- ***Características del SDK(C++)***

El SDK es el preferido cuando el rendimiento es lo más importante; en general cuando el cálculo es el principal propósito del plugin.

Debido al gran impacto del 3ds Max en el mundo de la realidad virtual se ha incorporado una nueva librería al SDK, llamada **IGame.lib**, mediante la cual se puede acceder a todos los atributos de los nodos de una escena con un pequeño esfuerzo en comparación con los métodos existentes anteriormente. [9]

Capítulo 2: Descripción de las Soluciones Propuestas

2.1 Forma de Salvado.

El fichero se exportará en **binario**, pues como se analizó en la fundamentación teórica, 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.

2.2 Estructura y nombre del fichero.

El nuevo formato se llamará .BTK (Bone Tool Kit) que se debe a la herramienta para la cual se está elaboro, el nombre completo de la herramienta es Scene Tool Kit (STK). La organización está dada en bloques de datos como muestra la figura.

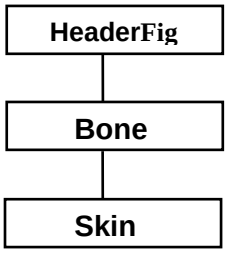


Fig. 4 Estructura del fichero BTK

El formato está estructurado de la siguiente forma, primero la cabecera (Header), y posteriormente el bloque de huesos y finalmente el bloque skin. En la fundamentación teórica se muestra como la mayoría de los formatos ya existentes usan una organización en forma de bloques, este tipo de estructura permite que a la hora de leer el fichero, se cargue un bloque de datos por completo, el cabezal del disco duro sólo tendrá que moverse en una sola dirección una vez que se está leyendo un bloque, logrando mayor velocidad de lectura cuando se está cargando el fichero.

2.2.1 Características del bloque Header.

El bloque **Header** representa el encabezado del fichero, en este se resumen un conjunto de informaciones generales tales como: la versión, nombre, entre otros. A continuación del encabezamiento para un mejor entendimiento.

- float fVersion: Indica la versión actual del fichero.
- unsigned int iBoneCount: Cantidad de huesos que tiene almacenado el fichero.
- unsigned int iReserved1: espacio reservado para posible uso en un futuro, permitiendo insertar cualquier tipo de dato de interés.
- unsigned int iReserved2: espacio reservado para posible uso en un futuro, permitiendo insertar cualquier tipo de dato de interés.
- char acMeshName[32]: nombre de la malla a la que pertenece el sistema de huesos.

¿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 qué versión de formato es con la que se está trabajando, cuántos elementos de un tipo de bloque están en el fichero.

2.2.2 Características del bloque Bone.

Bone contiene toda la información respecto a los huesos, estos están almacenados en forma de árbol jerárquica.

Int iBonneID: Indica el id del hueso.

Int iBonneIDParent: id del hueso padre.

Int iChildQuantity: Cantidad de hijos.

sPoint3d sTMatrix[4]: Matriz de posición del hueso.

2.2.3 Características del bloque Skin.

Está conformado por cSkinStructures donde cada elemento es un vértice que contiene una lista de los huesos que lo afectan.

Int iVertexIndex: índice del vértice.

Int iBonneAffectQuantity: Cantidad de huesos que afectan al vértice.

Bool bVertexType: tipo de vértice (rigid o blended)

Vector<sBoneWeight*> TBoneWeigth: lista de estructuras sBoneWeight

2.4 Aspectos generales.

Con el uso de la librería IGame incluida en el sdk de Max generamos el fichero, como se analizó en la fundamentación teórica, utilizamos la misma por su flexibilidad y rapidez a la hora de crear una nueva funcionalidad para el 3d Studio Max, C++ como lenguaje de programación y el diseño e implementación se hará utilizando la filosofía de programación orientada a objetos.

Conclusiones.

En este capítulo han quedado plasmadas las soluciones técnicas para resolver el problema planteado, el lenguaje de programación que utilizaremos, así como otros elementos que se utilizaran para lograr los objetivos propuestos.

Capítulo 3 Construcción de la Solución Propuesta.

Introducción

En este capítulo se comienza a tener la visión del sistema a desarrollar. Aquí se inicia la concepción práctica del producto a elaborar, se definen un conjunto de reglas, orientadas a los diseñadores gráficos para que exporten la información de los huesos con el nuevo formato de fichero.

3.1 Reglas del Negocio.

Las Reglas del Negocio propuestas son:

1. Los personajes deben estar ubicados en el origen de coordenadas y con los pies sobre el cero del eje vertical Z.

2. Los huesos deben tener asociados al menos un vértice de la malla.

3. Los huesos deben de estar jerarquizados correctamente.

4. Para generar el fichero es necesario tener instalado el 3d Studio Max 2008 o una versión mayor que 3d Studio Max 6.

3.2 Modelo del Dominio

Para el modelamiento del sistema, se seleccionó lo que se conoce como Modelo del Dominio, que viene a ser un modelo conceptual del sistema a desarrollar como se muestra en la siguiente figura:

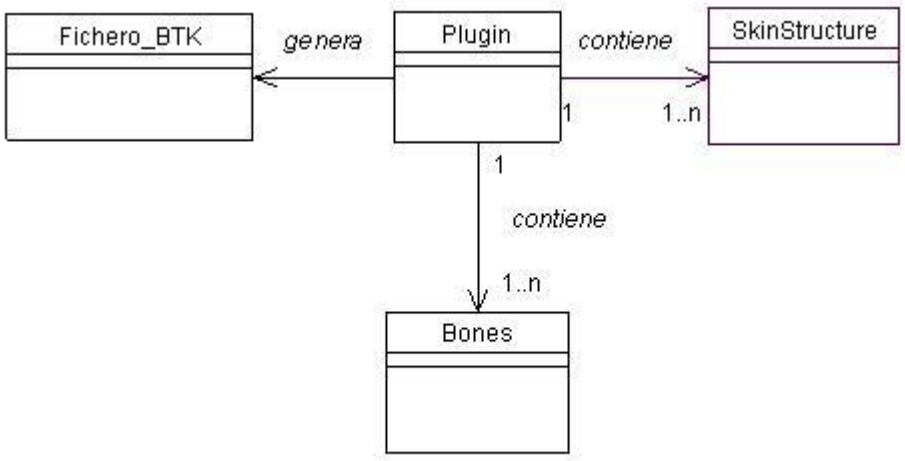


Fig. 5 Modelo de Dominio BTKExporte

3.2.1 Glosario de términos del dominio.

Se describen los conceptos manejados en el modelo de dominio para una mejor comprensión y entendimiento del mismo

Plugin: Es un contenedor de una funcionalidad que permiten a un software realizar tareas adicionales.

Bones: Contiene la lista de huesos almacenados en forma jerárquica.

SkinStructure: Esta estructura contiene una lista de los vértices de la malla donde cada vértice tiene la información de los huesos que los afectan directamente.

Fichero_BTK: Este es el fichero que se genera al exportar desde el 3d StudioMax con el plugin BTKExporter.

3.3 Captura de Requisitos.

En este epígrafe se define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen, que esto no es más que los requerimientos del sistema, tanto los funcionales como los no funcionales.

3.3.1 Requisitos Funcionales.

1. Comprobar escena
   - 1.1 Obtener la cantidad de nodos principales de la escena (TopNode).
   - 1.2 Identificar los nodos principales.
   - 1.3 Comprobar que exista una malla.
   - 1.4 Comprobar que la malla tenga algún modificador skin
   - 1.5 Comprobar que el modificador skin tenga huesos asignados.
   - 1.6 Obtener el nombre de la malla.

2 Almacenar lista de huesos
   - 2.1 Identificar si el nodo es un hueso
   - 2.2 Obtener cantidad de huesos de la jerarquía.
   - 2.3 Obtener id del hueso.
   - 2.4 Obtener id del hueso padre.
   - 2.5 Obtener matriz de transformación del hueso.
   - 2.6 Crear hueso genérico.

3 Almacenar lista de skin
   - 3.1 Identificar si el nodo es de tipo malla
   - 3.2 Obtener el modificador skin de la malla.
   - 3.3 Identificar el tipo de vértice.

3.4 Obtener el índice del vértice.

3.5 Obtener los huesos que afectan el vértice.

3.6 Obtener el peso del hueso sobre el vértice.

3.7 Crear un skin genérico.

4 Escribir la información del bloque cabecera en el fichero.

5 Escribir la información del bloque huesos en el fichero.

6 Escribir la información del bloque skin en el fichero.

7 Crear el fichero btk.

3.3.2 Requisitos no Funcionales.

- **Software:**
- 3d Studio Max versión desde la 6 hasta la 8 que corre en los sistemas operativos, Windows XP Home Edition SP2, Windows 2000 SP4, Windows XP Profesional SP2 (recomendado), además es necesario tener instalado DirectX 9.0c.
- **Hardware:** Tarjeta RAM 128 en adelante.
- **Restricciones de Diseño e implementación**: Debe utilizarse una arquitectura por capas donde exista una capa superior en lenguaje C++ orientado a objetos y otra inferior en lenguaje C++ que utiliza la librería **IGAME** y **SDK** del 3dStudioMax.
- **Legales:** Se regirá por las normas ISO 9000.
- **Usabilidad:** Cualquier usuario operador del 3dStudio Max.
- **Rendimiento:** Como aplicación en tiempo real debe tener alta velocidad de procesamiento de los cálculos, tiempo de respuesta y recuperación del sistema.
- **Soporte:** Compatible con versiones de 3d Studio Max desde la 6 hasta la 2008 que corren sobre sistemas operativos Microsoft Windows NT versión 4.0 en adelante.
- **Portabilidad:** La herramienta puede ser modificada para exportar la información de los huesos de cualquier otro Software de diseño, gracias a la arquitectura que presenta.

3.4 Modelo de Casos de Usos del Sistema.

Las funcionalidades definidas en el epígrafe anterior “Captura de requisitos” se agrupan dando como resultado los casos de uso del sistema, además de estos elementos en este capítulo analizáremos los actores que van a interactuar con cada uno de ellos.

3.4.1 Actor del sistema

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

Tabla 1: Actor Del Sistema.

3.4.2 Casos de Usos del Sistema

Id CUS	**Casos de Usos del Sistema**
CUS1	Exportar Fichero
CUS2	Crear estructura de huesos
CUS3	Crear estructura de Skin

Tabla 2: Identificador por cada Caso Uso del Sistema.

3.4.3 Diagramas de CUS y CA

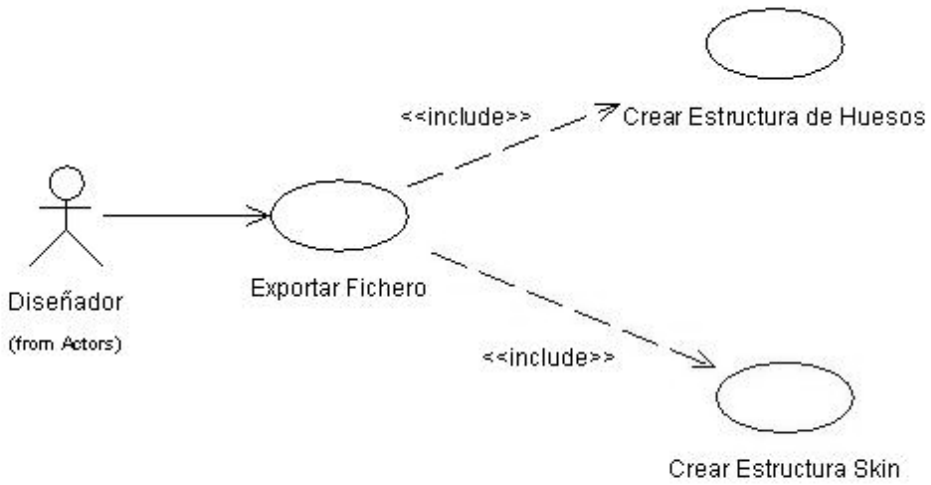


Fig. 6 Diagrama de CUS.

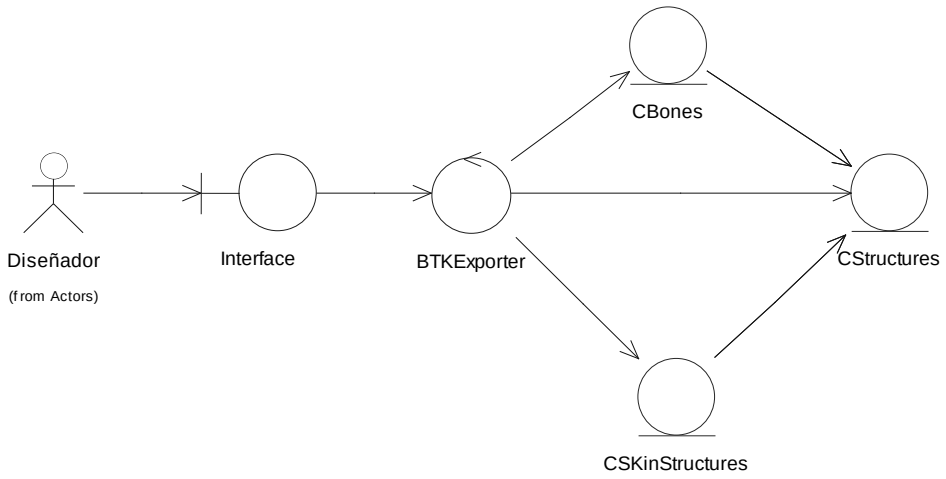


Fig. 7 Diagrama de CA

3.4.4 Especificación de los CUS en formato expandido

Caso de uso:	Exportar Fichero
Actor (es):	Diseñador
Propósito:	Crear fichero BTK
Resumen:	El CUS comienza cuando el actor exporta la estructura de huesos del personaje u objeto y termina al generarse el fichero.
Referencias:	CU 2, CU 3
Pre-condiciones:	

Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
Sección: Exportar Fichero.	
1- Seleccionar opción exportar fichero BTK.	1.1- Muestra una ventana con las opciones de tipo de fichero a exportar.
2- Selecciona las opciones que desee y selecciona la opción a exportar.	2.1 Se comprueba la escena. 2.2 Si no cumples las condiciones ir al curso alternativo “Condiciones no cumplidas”. 2.3 Se invoca al CU-2 “Crear estructura de huesos”.

	2.4 Se invoca al CU-3 “Crear estructura de Skin”. 2.5 Se crea el fichero BTK.
Curso Alternativo de los Eventos	
Sección: Condiciones no cumplidas.	
	1.1.3- Si la escena no cumple con las condiciones se muestra un mensaje.
Post-condiciones:	
Prioridad:	Crítico.

Tabla 3: Descripción del CUS “Exportar Fichero”.

Caso de uso:	Crear estructura de huesos
Actor (es):	CU “Exportar Fichero”
Propósito:	Crear la estructura de huesos del objeto o personaje para almacenarla en el fichero BTK
Resumen:	El CU se inicia cuando es invocado por el CU “Exportar fichero”. Se recorren los huesos de la escena, se hace un proceso de captura de datos y se almacenan en la lista de huesos de forma jerárquica.
Referencias:	Hereda del CUS1
Pre-condiciones:	

Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Invoca al CU-1 “Crear estructura de huesos”.	1.1 Se recorren los nodos de tipo hueso de la escena 1.2 Se capturan todos los atributos necesarios para la construcción del árbol jerárquico. 1.3 Se crea el hueso genérico. 1.4 Se almacena en la lista de huesos.
Curso Alternativo de los Eventos	

Post-condiciones:	
Prioridad:	Crítico.

Tabla 4: Descripción del CUS “Crear estructura de huesos”.

Caso de uso:	Crear estructura Skin
Actor (es):	CU “Exportar Fichero”
Propósito:	Crear la estructura de Skin para almacenarla en el fichero BTK
Resumen:	El CU se inicia cuando es invocado por el CU “Exportar fichero”. Se recorren los vértices de la malla, se comprueba el tipo de vértice, posteriormente se recorre la lista de huesos que afectan dicho vértice se hace un proceso de captura de datos y se almacenan en la lista de huesos de forma jerárquica.
Referencias:	Hereda del CUS1
Pre-condiciones:	

Curso Normal de los Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Invoca al CU-1 “Crear estructura de Skin”.	1.1 Se obtiene el Skin del nodo de tipo malla. 1.2 Se recorre la lista de vértices. 1.3 Se identifica el tipo de vértice. 1.4 Se capturan los atributos del vértice. 1.5 Se recorre la lista de huesos que afectan al vértice. 1.6 Se almacenan los huesos en la lista. 1.7 Se crea un SkinStructure 1.8 Se almacena en la lista de SkinStructure.
Curso Alternativo de los Eventos	

Post-condiciones:	
Prioridad:	Crítico.

Tabla 5: Descripción del CUS “Crear estructura Skin”.

Conclusiones

En este capítulo se definió las reglas del negocio, se le dio una primera visión de lo que se pretende obtener con este módulo. Para ello quedaron establecidos los requisitos funcionales y los no funcionales del mismo, se muestra los diagramas de casos de usos del sistema y la descripción textual de los mismos, lo que le permitirá al futuro usuario obtener los resultados esperados por el cliente.

Capítulo 4: Diseño del Sistema

Introducción

En este capítulo se presentan los diagramas de clases del diseño como resultado del refinamiento de las etapas anteriores. Se muestran los diagramas de clases agrupados por paquetes así como las relaciones entre los mismos,. Se presentan además los diagramas de interacción del diseño, en este caso los diagramas de secuencia de la realización de los casos de usos.

4.1 Diagramas de clases del Diseño.

Por la complejidad del Diagrama de clases de diseño y para un mejor entendimiento se han agrupado las clases en paquetes, el paquete IGame_SDK encapsula las clases que se utilizan de la librería IGame del 3dStudioMax y las clases que utiliza del SDK, encargadas de manejar los nodos de una escena, el paquete BTKExporter contiene las clase del mismo nombre y su relación con las clases del IGame_SDK. En el paquete Bones es donde se encapsula las clases encargadas de almacenar la estructura de huesos, y en el paquete Skin se encapsula las clases que se encargan de almacenar la información de los vértices y los huesos que afectan a los mismos.

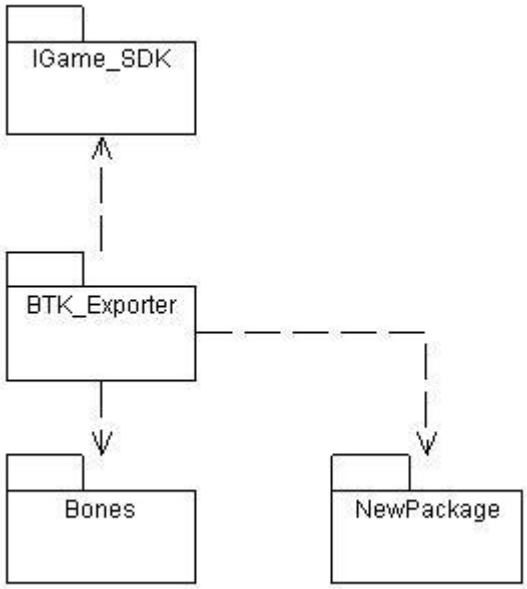


Fig. 8 Diagrama de paquetes del diseño.

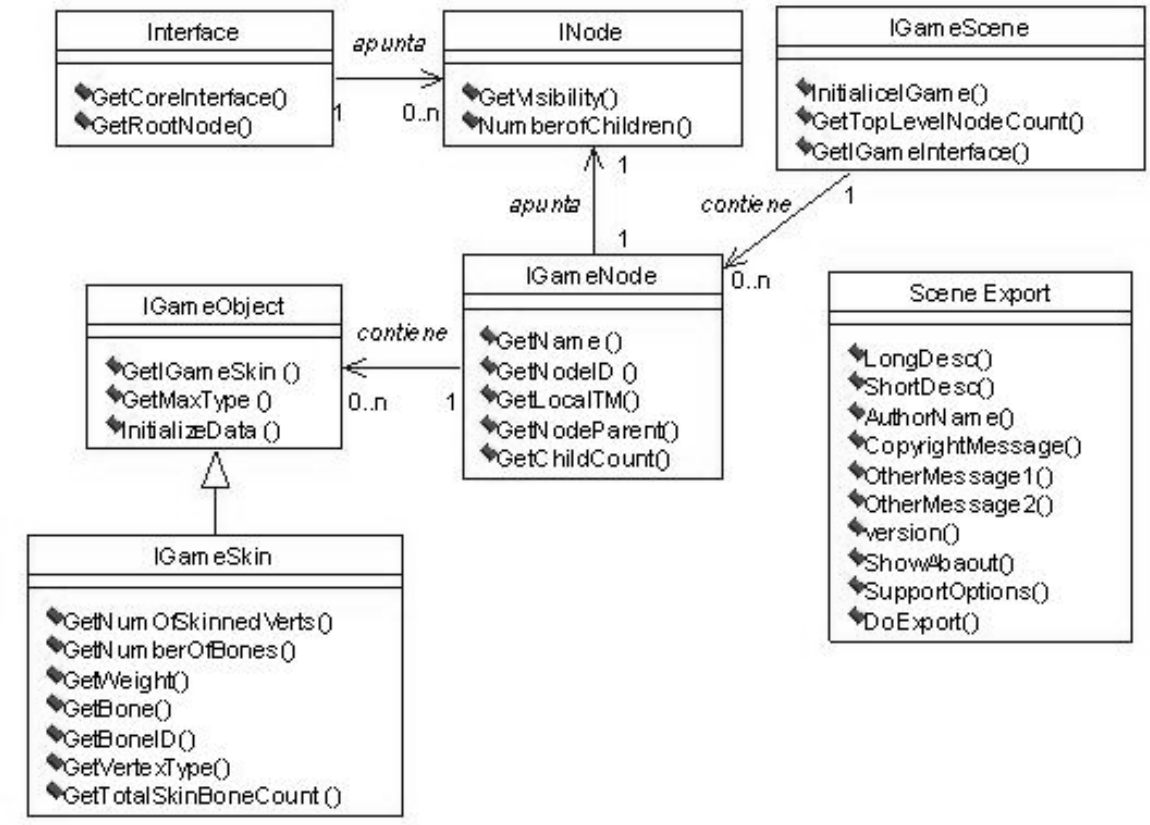
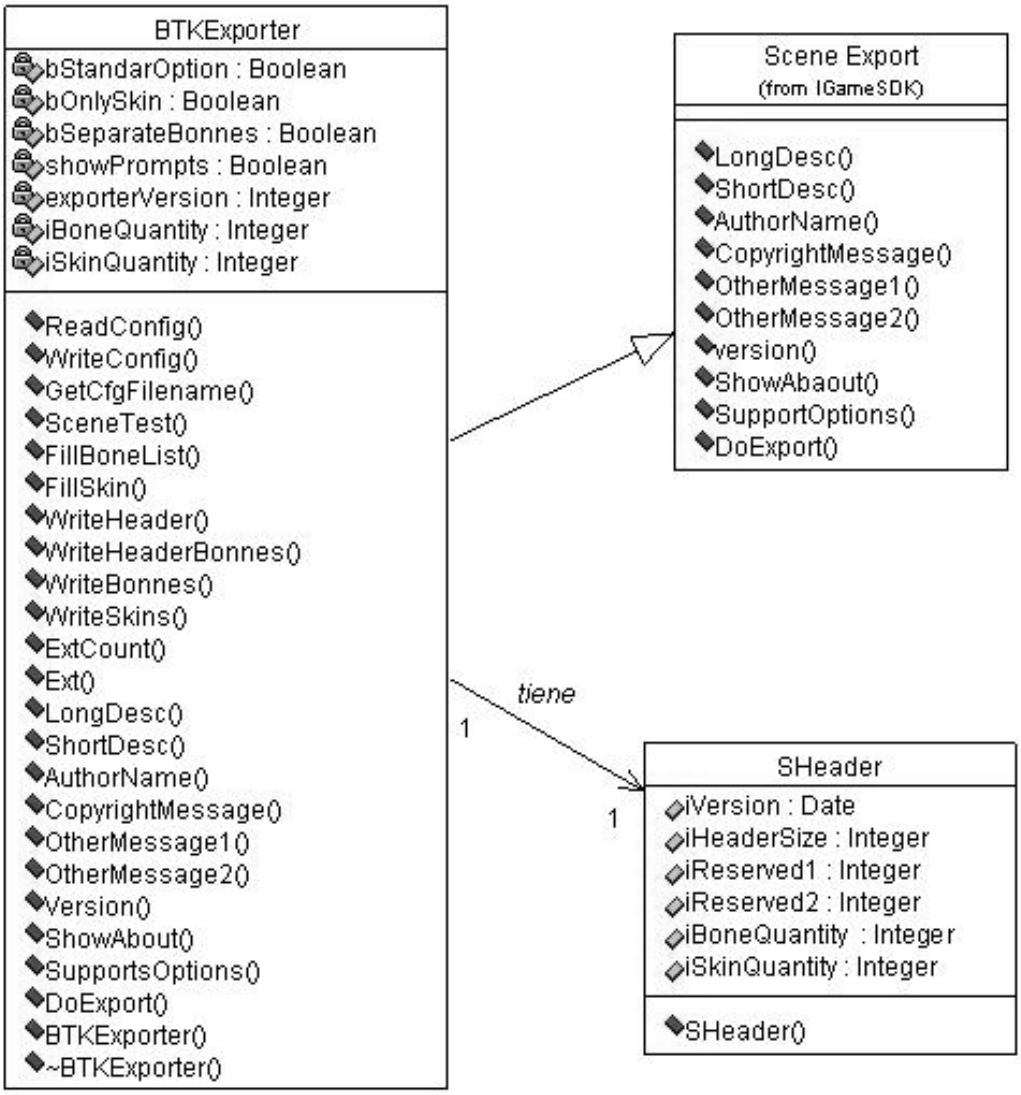
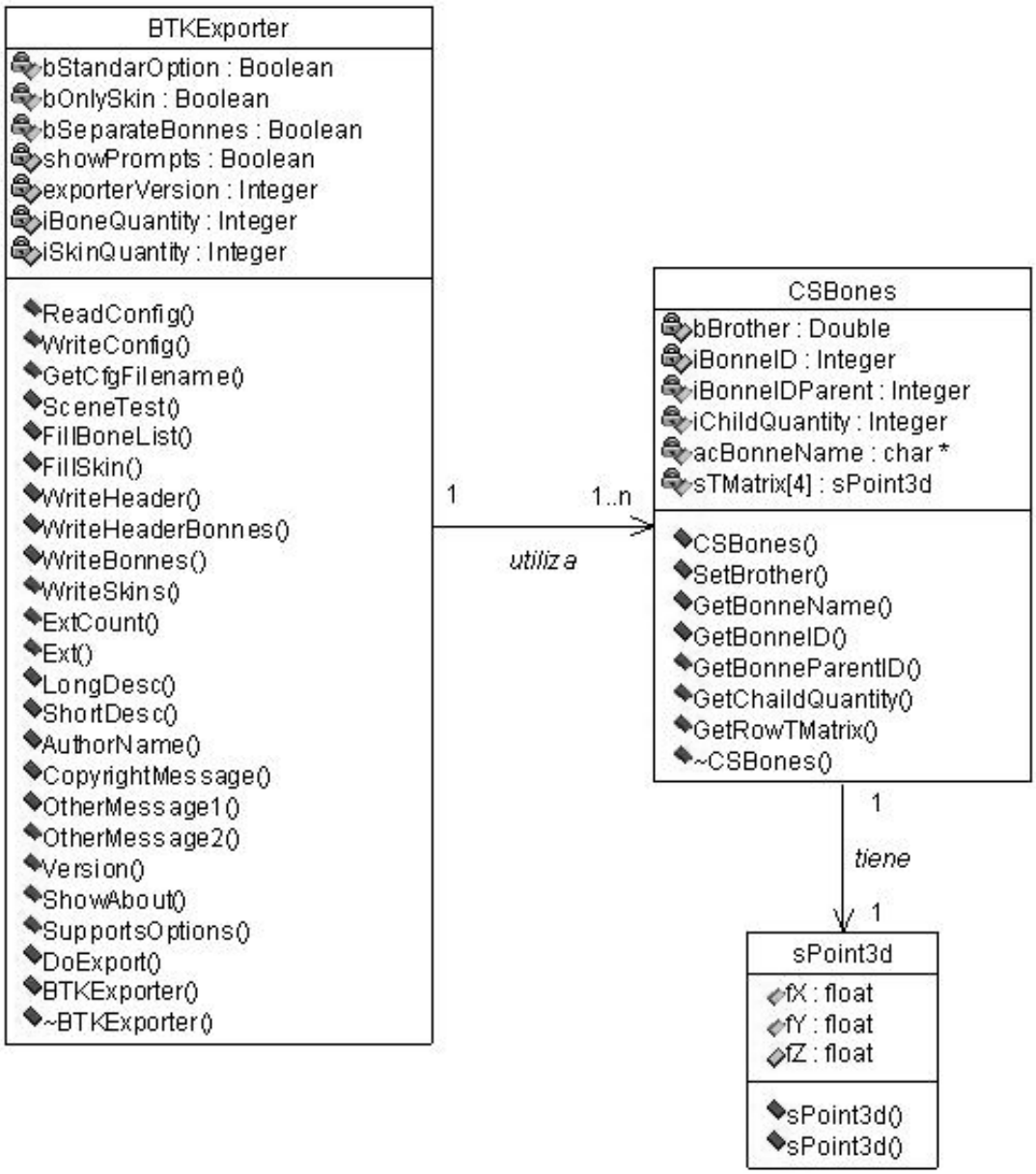


Fig. 9 Diagrama de clases del paquete IGame_SDK


BTKExporter
bStandarOption : Boolean
bOnlySkin : Boolean
bSeparateBonnes : Boolean
showPrompts : Boolean
exporterVersion : Integer
iBoneQuantity : Integer
iSkinQuantity : Integer
ReadConfig()
WriteConfig()
GetCfgFilename()
SceneTest()
FillBoneList()
FillSkin()
WriteHeader()
WriteHeaderBonnes()
WriteBonnes()
WriteSkins()
ExtCount()
Ext()
LongDesc()
ShortDesc()
AuthorName()
CopyrightMessage()
OtherMessage1()
OtherMessage2()
Version()
ShowAbout()
SupportsOptions()
DoExport()
BTKExporter()
~BTKExporter()
Scene Export
(from IGameSDK)
LongDesc()
ShortDesc()
AuthorName()
CopyrightMessage()
OtherMessage1()
OtherMessage2()
version()
ShowAbaout()
SupportOptions()
DoExport()
tiene
1
1
SHeader
iVersion : Date
iHeaderSize : Integer
iReserved1 : Integer
iReserved2 : Integer
iBoneQuantity : Integer
iSkinQuantity : Integer
SHeader()


BTKExporter
bStandarOption : Boolean
bOnlySkin : Boolean
bSeparateBonnes : Boolean
showPrompts : Boolean
exporterVersion : Integer
iBoneQuantity : Integer
iSkinQuantity : Integer
ReadConfig()
WriteConfig()
GetCfgFilename()
SceneTest()
FillBoneList()
FillSkin()
WriteHeader()
WriteHeaderBonnes()
WriteBonnes()
WriteSkins()
ExtCount()
Ext()
LongDesc()
ShortDesc()
AuthorName()
CopyrightMessage()
OtherMessage1()
OtherMessage2()
Version()
ShowAbout()
SupportsOptions()
DoExport()
BTKExporter()
~BTKExporter()
1
1..n
utiliza
CSBones
bBrother : Double
iBonneID : Integer
iBonneIDParent : Integer
iChildQuantity : Integer
acBonneName : char *
sTMatrix[4] : sPoint3d
CSBones()
SetBrother()
GetBonneName()
GetBonneID()
GetBonneParentID()
GetChaildQuantity()
GetRowTMatrix()
~CSBones()
1
tiene
1
sPoint3d
fX : float
fY : float
fZ : float
sPoint3d()
sPoint3d()

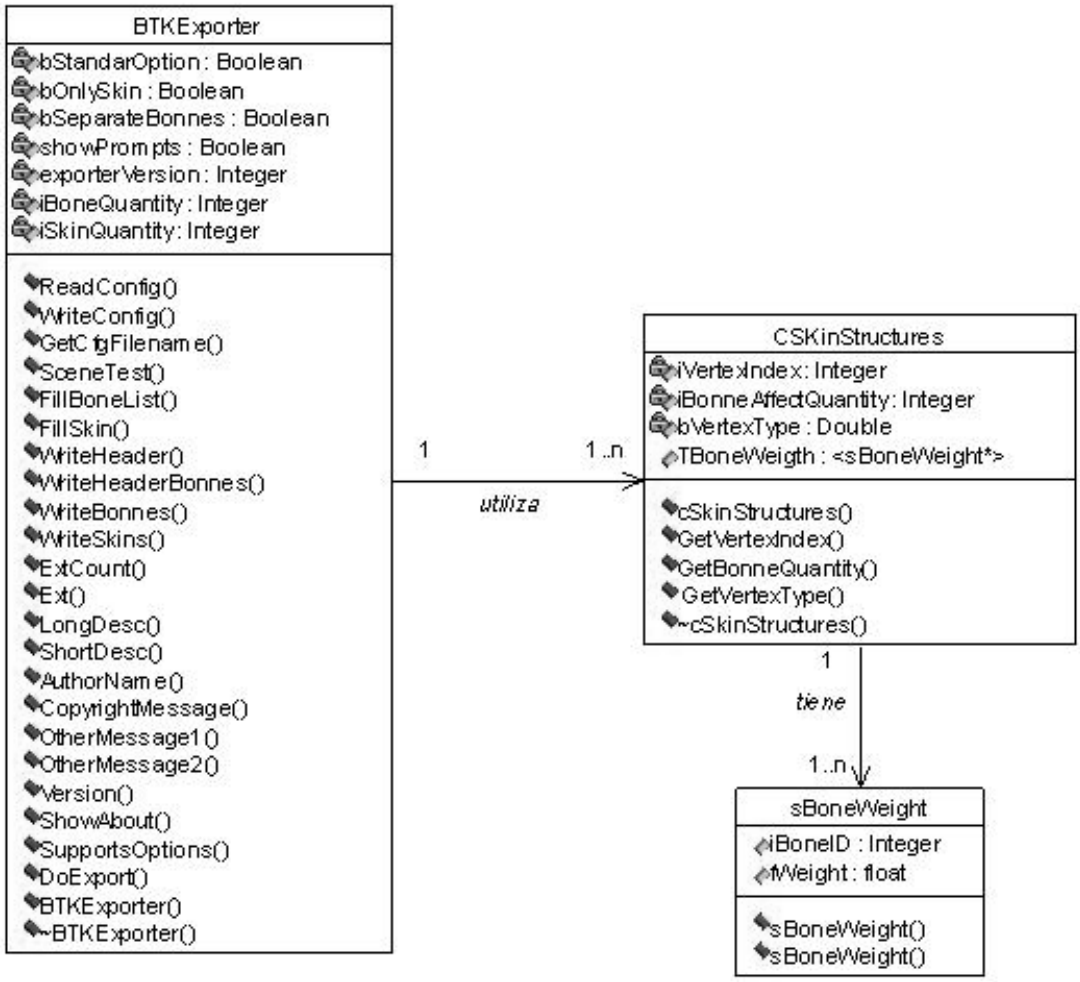


Fig. 10 Diagrama de Clases BTK_Exporter I, II, III.

4.1.1 Descripción de las clases del diseño.

Las clases de diseño que se describen a continuación son las más importantes dentro del diseño de clases del módulo. Los métodos de acceso a miembros de clases (set y get) no son especificados en las descripciones, ni tampoco los destructores, puesto que su funcionalidad es sencilla. Las clases del paquete IGame_SDK no se describen pues no son propias del fichero.

Nombre	BTKExporter
Tipo de clase	Controladora
Atributo	**Tipo**
TBoneList	Vector <CSBones*>
TSkinList	Vector <cSkinStructures*>
Coord_System	int
bStandarOption	bool
bOnlySkin	bool
bSeparateBonnes	bool
showPrompts	bool
exporterVersion	int
acMeshName	char*
iBoneQuantity	int
iSkinQuantity	int
Para cada responsabilidad	
Nombre	ReadConfig()
Descripción	Lee la última configuración que quedó en la aplicación.
Nombre	WriteConfig()
Descripción	Escribe la configuración que dejó el usuario en el fichero de configuración.
Nombre	GetCfgFilename()
Descripción	Carga el archivo de configuración.
Nombre	SceneTest()
Descripción	Comprueba que el personaje u objeto de la escena cumpla con las condiciones necesarias para ser exportado con la herramienta.
Nombre	FillBoneList(IGameNode*, bool)
Descripción	Llena la lista de huesos.
Nombre	FillSkin(IGameNode*)
Descripción	Llena la lista de sKinStructure.
Nombre	WriteHeader(FILE*)
Descripción	Escribe la cabecera del fichero.
Nombre	WriteHeaderBonnes(FILE*)
Descripción	
Nombre	WriteBonnes(FILE*)
Descripción	Retorna m_iHeight.
Nombre	WriteSkins(FILE*)
Descripción	Retorna m_iLength.
Nombre	Ext(int n)
Descripción	Extensión del fichero.
Nombre	ExtCount()
Descripción	Cantidad de extensiones que soporta.
Nombre	LongDesc()
Descripción	Descripción larga del plugins.
Nombre	ShortDesc()

Descripción	Descripción corta del plugins.
Nombre	AuthorName()
Descripción	Nombre del autor o autores.
Nombre	CopyrightMessage()
Descripción	Derechos de autor.
Nombre	OtherMessage1()
Descripción	Mensaje1.
Nombre	OtherMessage2()
Descripción	Mensaje2.
Nombre	Version()
Descripción	Versión del exportador.
Nombre	ShowAbout(HWND hWnd)
Descripción	Muestra una ventana con información de interés sobre la realización del plugins.
Nombre	SupportsOptions
Descripción	Muestra las extensiones que soportan el plugins.
Nombre	DoExport()
Descripción	Se encarga de procesar una escena y hace llamadas a métodos que permiten almacenar información de interés y exportarla en un fichero BTK.
Nombre	CSTKExporter()
Descripción	Construye un objeto de tipo CSTKExporter.
Nombre	~CSTKExporter()
Descripción	Destruye y libera la memoria de un objeto tipo CSTKExporter.

Tabla 6: Descripción de la clase del diseño "BTKExporter"

Nombre	CSBones
Tipo de clase	Entidad
Atributo	**Tipo**
acBonneName	char*
iBonneID	int
iBonneIDParent	int
bBrother	bool
iChildQuantity	int
sTMatrix[4]	sPoint3d
Para cada responsabilidad	
Nombre	CSBones(char* acName, int iID, int iIDParent, bool bBrother, int iChild, sPoint3d sMatrix[4])
Descripción	Construye un objeto de tipo CSBones.

Tabla 7 Descripción de la clase del diseño "CSBones"

	cSkinStructures	
Tipo de clase	Entidad	
Atributo		**Tipo**
TBoneWeight		vector<sBoneWeight*>
iVertexIndex		int
iBonneAffectQuantity		int
bVertexType		bool
Para cada responsabilidad		
Nombre	cSkinStructures(int iAuxVertexIndex, bool bAuxVertexType, int iBonneQuantity)	
Descripción	Construye un objeto cSkinStructures.	

Tabla 8 Descripción de la clase del diseño "cSkinStructures"

Nombre	sPoint3d	
Tipo de clase	Entidad	
Atributo		**Tipo**
fX		float
fY		float
fZ		float
Para cada responsabilidad		
Nombre	sPoint3d(float v_x, float v_y, float v_z)	
Descripción	Construye un objeto sPoint3d.	
Nombre	CCubemap(char*arg_acXPos, char*arg_acXNeg, char*arg_acZPos, char*arg_acZNeg, char*arg_acYPos, char*arg_acYNeg)	
Descripción	sPoint3d()	
Nombre	Constructor vacio por defecto.	

Tabla 9 Descripción de la clase del diseño "sPoint3d"

Nombre	sBoneWeight	
Tipo de clase	Entidad	
Atributo		**Tipo**
iBoneID		int
fWeight		float
Para cada responsabilidad		
Nombre	sBoneWeight(unsigned int iAuxBoneIndex, float fAuxWeigth)	
Descripción	Construye un objeto sBoneWeight	
Nombre	sBoneWeight()	
Descripción	Contructor vacio por defecto.	

Tabla 10 Descripción de la clase del diseño "sBoneWeight"

4.2 Diagramas de Secuencia

Dentro de los diagramas de interacción tomamos el diagrama de secuencia que muestra la interacción entre objetos, ordenadas en secuencia temporal durante un escenario concreto.

.

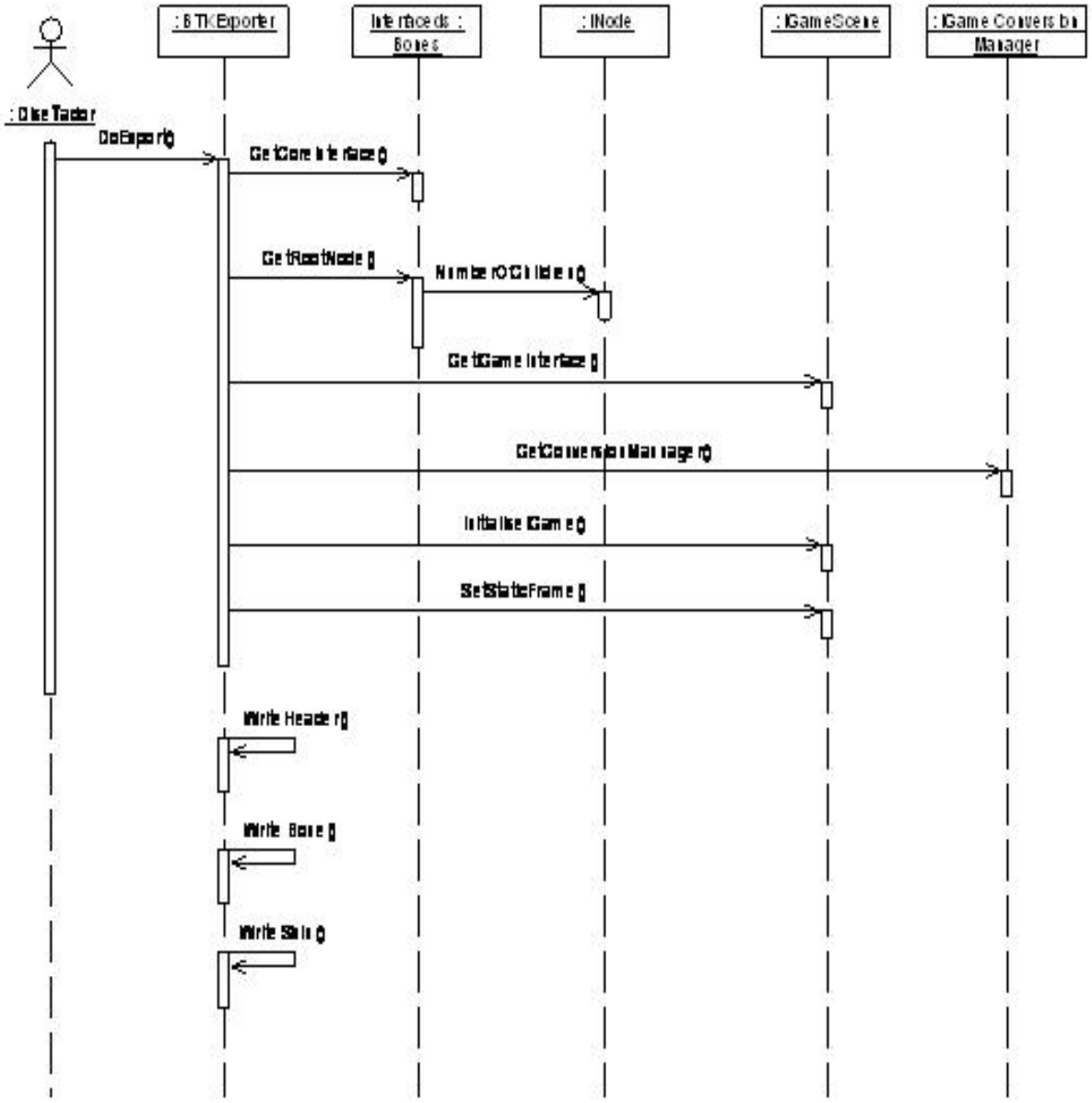


Fig. 11 Diagrama de secuencia: Exportar Fichero.

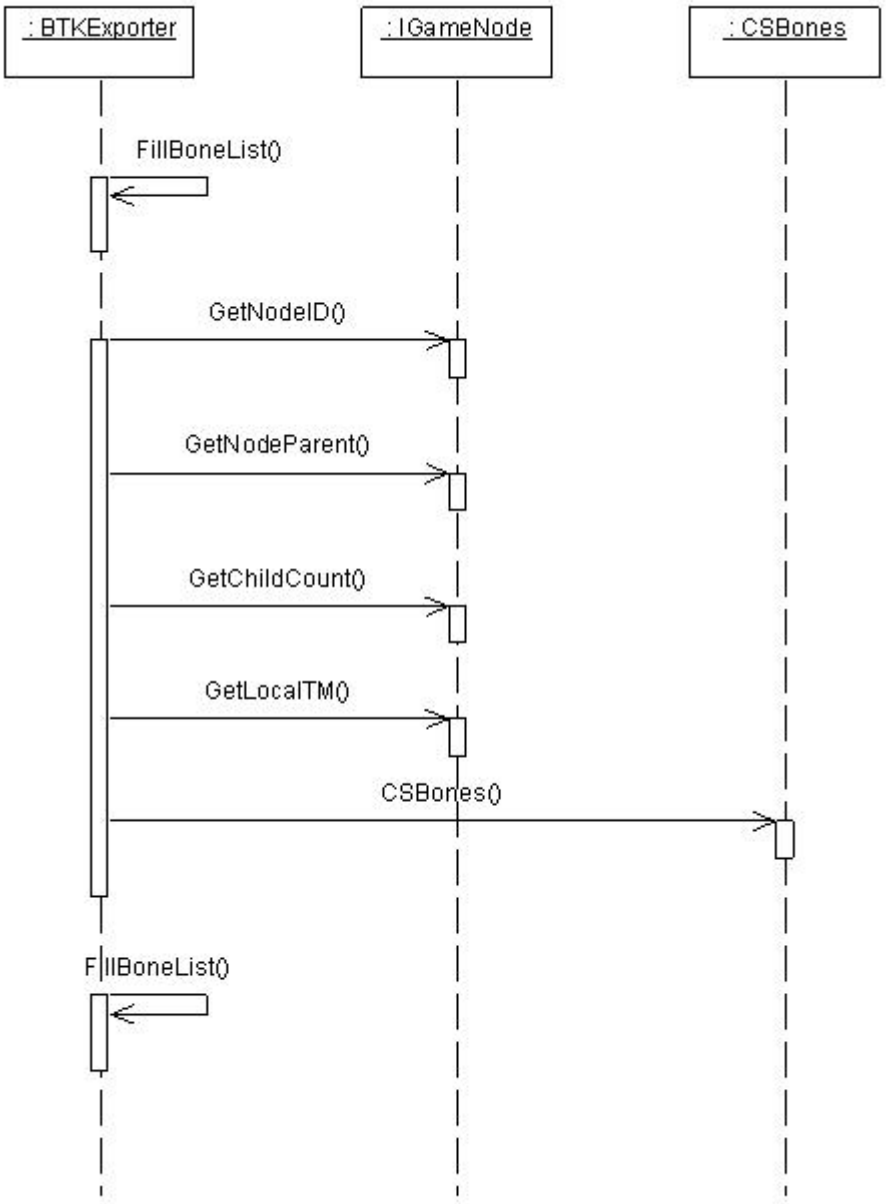


Fig. 12 Diagrama de secuencia: Crear Estructura de Huesos.

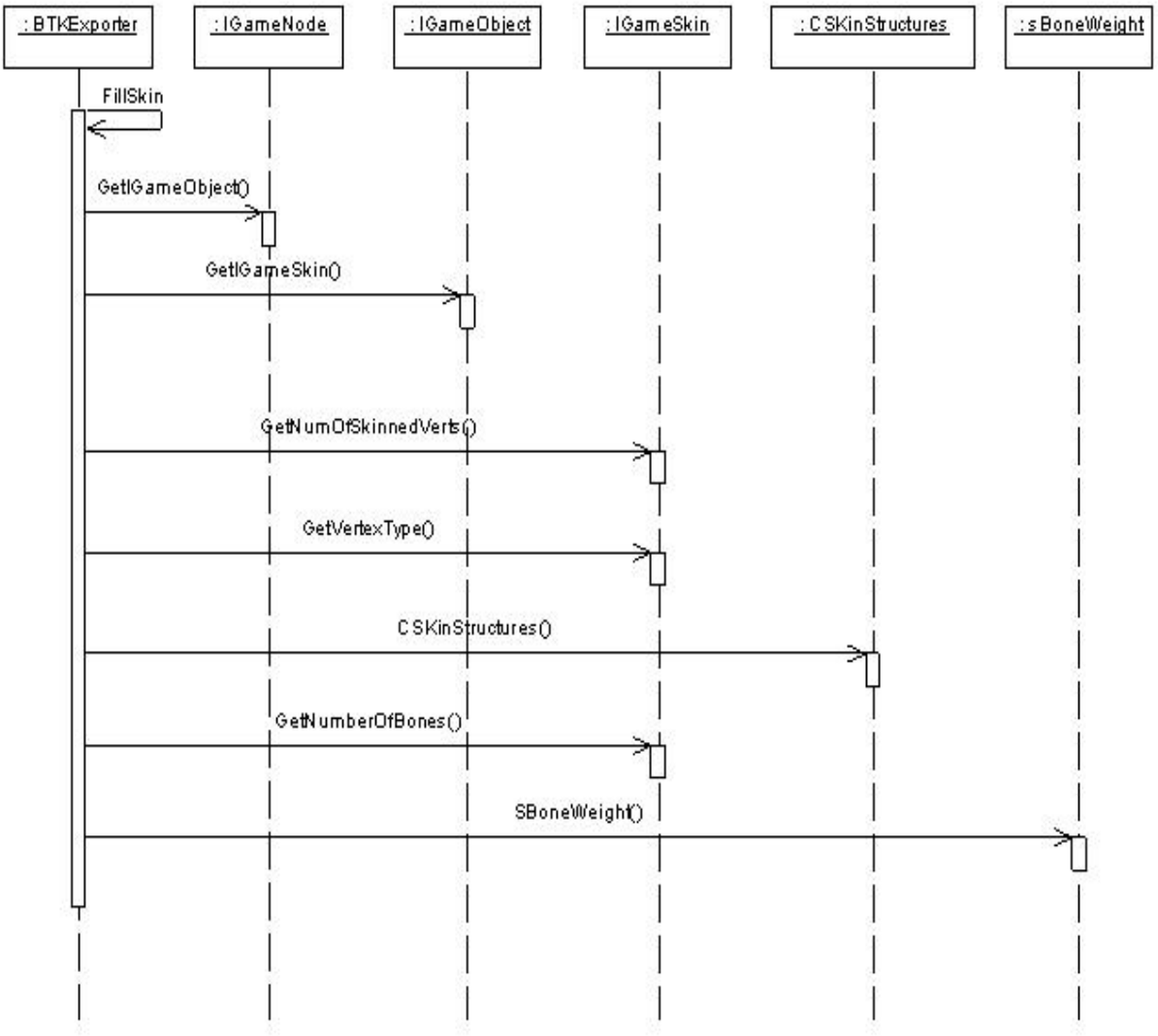


Fig. 13 Diagrama de secuencia: Crear Estructura Skin.

Capítulo 5: Implementación del sistema

Introducción

Esta etapa del proyecto constituye el paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .h y .cpp correspondientes a la implementación en C++.

5.1 Estándares de codificación

Se programará en inglés, debido que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático. Se respetarán los estándares de codificación para C++ (identado, uso de espacios y líneas en blanco, etc.).
El conocimiento de los estándares seguidos para el desarrollo del módulo permitirá un mayor entendimiento del código, y es una exigencia de los autores de la misma que cualquier módulo que se añada debe estar codificado siguiendo estos estándares.

Nombre de los ficheros:
Se nombrarán los ficheros .h y .cpp de la siguiente manera:
STKNameOfUnits.cpp
Se usará **GT** para identificar el nombre de la herramienta (en definición!!)

Constantes:
Las constantes se nombrarán con mayúsculas, utilizándose el "_" para separar las palabras: MY_CONST_ZERO = 0;

Tipos de datos:
Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados: enum **E**MyEnum {**ME**_VALUE, **ME**_OTHER_VALUE};
Indicando con "**E**" que es de tipo enumerado. Nótese que las primeras letras de las constantes de enumerados son las iniciales del nombre del enumerado. Véase otro ejemplo:
enum **E**Node**T**ype {**NT**_GEOMETRYNODE,…};

Estructuras: struct **S**MyStruct {…};
Indicando con "**S**" que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

Clases: class **C**ClassName;
Indicando con "**C**" que es una clase

Declaración de variables:

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepondrá el identificador “” (en minúscula), si son globales se les antepondrá la letra “g”, y en caso de ser argumentos de algún método, se les antepondrá el prefijo “arg_”.

Tipos simples:

bool **b**VarName;
int **i**Name;
unsigned int **ui**Name;
float **f**Name;
char **c**Name;
char* **ac**Name; // arreglo de caracteres
char* **pc**Name; // puntero a un char
char** **aac**Name; // bidimensional
char** **apc**Name; // arreglo de punteros
bool **m_b**MemberVarName; //variable miembro
char **g**cGlobalVarName; //variable global, no se le antepone “”
short **s**Name;

Instancias de tipos creados:

EMyEnumerated **e**Name;
SMyStructure **k**Name;
CClassName **k**ObjectName;
CClassName* **pk**Name; //puntero a objeto
CClassName* **ak**Name; //arreglo de objetos
CClassName* **ak**Name; // variable miembro de clase
IMyInterface* **pI**Name; //puntero interfaces

Métodos

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada. Solamente los constructores y destructores comenzarán con “”.

En el caso de los argumentos se les antepone el prefijo “arg_”

Constructor y destructor:

CClassName (bool **arg**_bVarName, float& **arg**_fVarName);

~**C**ClassName ();

Funciones:

bool bFunction1 (…);

int* **pi**Function2 (…);

CClassName* **pk**Function3 (…);

Procedimientos:

void Procedure4 (…);

Métodos de acceso a miembros

Los métodos de acceso a los miembros de las clases no se nombrarán "Gets" y "Sets", sino como los demás métodos, pero **con el nombre** de la variable a la que se accede y sin "m_":

int **i**MyVar; //variable

Obtención del valor:

```
int iMyVar();
{
   return iMyVar;
}
```

Establecimiento del valor:

```
void MyVar(char* arg_iMyVar)
{
  iMyVar = arg_iMyVar;
}
```

Obtención y establecimiento del valor:

```
int& iMyVar();
{
   return iMyVar;
}
```

5.2 Diagrama de Componentes

Para lograr un mejor entendimiento del diagrama de componentes se dividió en dos vistas, una vista binaria donde se muestra la relación entre los componentes binarios y una vista de código fuente donde se encuentran los componentes .h y .cpp.

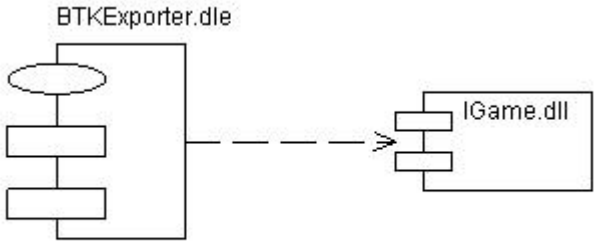


Fig. 14 Diagrama de Componentes de la vista binaria

Fig. 15 Componente binario BTKExporter.dle

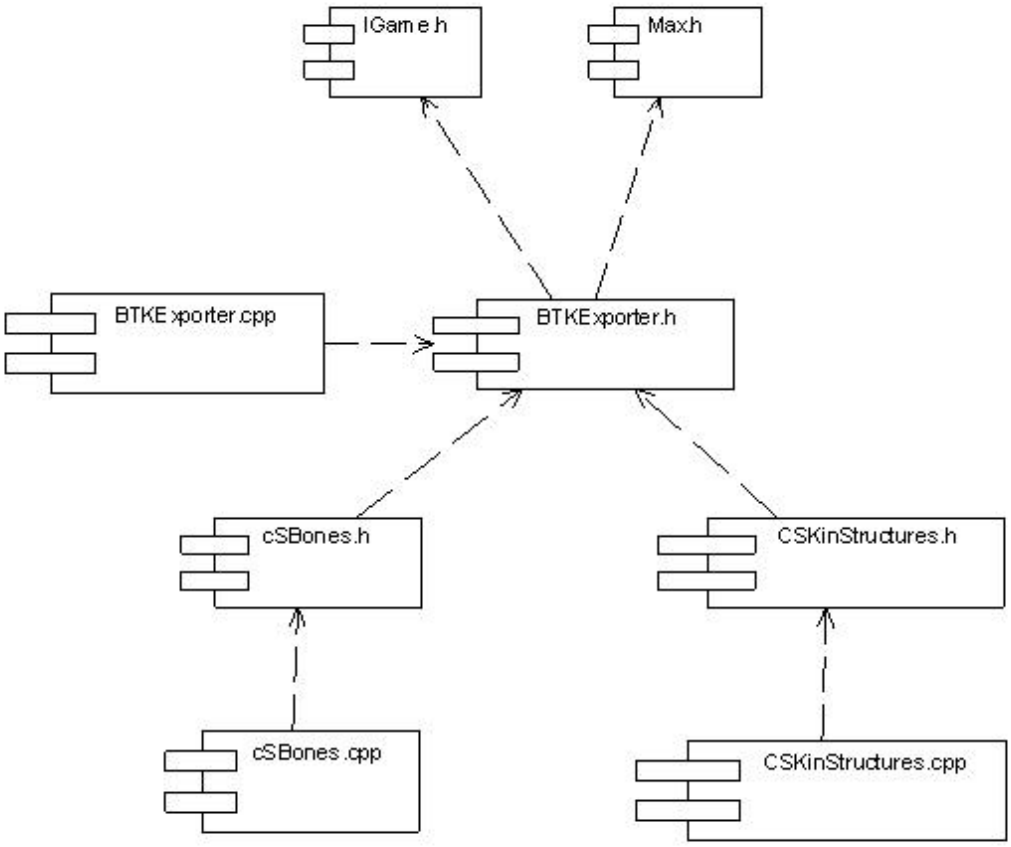


Fig. 16 Diagrama de Componentes de la vista de código de fuente.

5.3 Diagrama de Despliegue

Las relaciones físicas finales entre los componentes de hardware y software del sistema son representados a través de un solo nodo, el cual sería una PC.

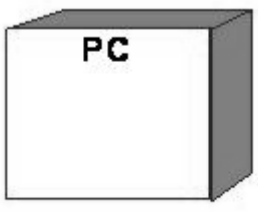


Fig. 17 Diagrama de Despliegue.

Conclusiones

En este momento se encuentra todo preparado para pasar a la etapa de programación de los casos de uso desarrollados en este ciclo. Como posibilidad adicional que brinda la herramienta Rational Rose, ya es posible generar el código fuente de los componentes relacionados con los casos de uso a desarrollar en el primer ciclo.

CONCLUSIONES

Durante la confección de este trabajo se realizó un estudio de las características que presentan las animaciones por huesos en los entornos virtuales, se investigaron los principales formatos de ficheros que soportan la animación por huesos, así como los algoritmos que se emplean en la generación y lectura de los mismos.

Se realizó el proceso de Ingeniería del Software utilizando el Proceso Unificado del Software (RUP) como metodología de desarrollo, posteriormente se hizo la captura de los requisitos funcionales y no funcionales, y la identificación de casos de uso del sistema en conjunto con su descripción en formato expandido, se diseñaron las clases y se creó el diagrama de componentes que contendrá a las clases del sistema.

Como resultado final de este trabajo se logro una herramienta flexible que genera un formato de fichero con toda la información de los huesos y los vértices asociados, que le permitirá a los desarrolladores mejorar la calidad de las animaciones en la herramienta "Scene Toolkit".

RECOMENDACIONES

- Realizar los cambios pertinentes en el modulo de animaciones de la STK para integrar el nuevo fichero BTK.
- Crear una herramienta (plugin) para el 3d Studio Max que permita exportar las animaciones de los huesos.
- Integrar todas las herramientas exportadoras en un mismo plugin.

REFERENCIA BIBLIOGRÁFICA.

1. **M, González Moreira.** Monografías.com. [Online] [Cited: Abril 30, 2008.] http://www.monografias.com/trabajos11/realitual/realitual.shtml .

2. **D, Ruiz and Sansano, A.** http://www.dccia.ua.es. [Online] [Cited: Abril 24, 2008.] http://www.dccia.ua.es/dccia/inf/asignaturas/RG/trabajos/trabajo-david-ruiz.pdf.

3. **S, Javier.** blogspot.com. [Online] [Cited: Marzo 21, 2008.] http://jusafing.blogspot.com/2007/05/animacion-esqueletal-es-una-tecnica-de.html.

4. **M, Putz and K, Hufnagl.** www.cg.tuwien.ac.at. [Online] [Cited: Abril 11, 2008.] http://www.cg.tuwien.ac.at/studentwork/CESCG/CESCG-2002/.

5. **E, Pipho and A, Lamothe.** *3D Model Game Development Series.* Cincinati, Ohio : s.n., 2003.

6. **Paul, Coppens.** gamedev.net. [Online] [Cited: Mayo 10, 2008.] http://www.gamedev.net/reference/programming/features/xfilepc/default.asp.

7. **Scarecrow.** Planetfortress.com. [Online] [Cited: marzo 20, 2008.] http://www.Planetfortress.com/TF2Models .

8. **Autodesk.** autodesk.com. [Online] [Cited: Junio 2, 2008.] http://images.autodesk.com/adsk/files/games_brochure0.pdf .

9. —. autodesk.com. [Online] [Cited: mayo 20, 2008.] http://images.autodesk.com/adsk/files/3dsmax8sdkhelp_chm.zip .

10. **Pipho, Evan.** "Focus On 3D Models". USA : Premier Press, 2003.

11. **Engel, Wolfgang F.** "Beginning Direct 3D Game Programming".

12. **ALBEE, T.** CHARACTER ANIMATION BOOK. 2004.

13. **MULTON, F.** Computer animation of human walking The Journal of Visualization and Computer Animation. FRANCE : s.n., 1999.

14. **MAESTRI, G.** Character Animation 2. USA : New Riders, 1999.

BIBLIOGRAFÍA CONSULTADA

1. Camacho Román, Y. R.; Jiménez López, F. *Biblioteca Gráfica Para Sistemas de Realidad Virtual*. Tesis de pregrado inédita, Instituto Superior Politécnico "JOSÉ ANTONIO ECHEVERRÍA", 2004
2. Alexis Echemendía Gonzalez; Wendy García López. *Sistemas de Generación de ficheros para entornos virtuales*. Tesis de pregrado inédita, Universidad de las Ciencias Informáticas, 2007
3. Karel Pérez Ramírez. *Modulo de animación de personajes para simuladores y juegos*. Tesis de pregrado inédita, Universidad de las Ciencias Informáticas. 2007
4. Autodesk, [Consultado en: 2006] "*Autodesk 3ds Max 8 SDK Help*" Disponible en: http://images.autodesk.com/adsk/files/3dsmax8sdkhelp_chm.zip
5. Pipho, Evan. "*Focus On 3D Models*" Premier Press. USA. 2003
6. Wolfgang F. Engel. "*Beginning Direct 3D Game Programming*" Premier Press.USA.2003

APÉNDICES.

Índice de Figuras y Tablas.

Referencia a Imágenes

Referencia a Tablas

Glosario de Abreviaturas

1D: Una dimensión

2D: Dos dimensiones.

3D: Tres dimensiones.

CPU: Unidad Central de Procesamiento.

CU: Caso de Uso

CUS: Caso de Uso del Sistema.

HW: Hardware

I: Vector Incidente.

RF: Requisito Funcional.

RV: Realidad Virtual

SRV: Sistemas de Realidad Virtual.

ASCII: American Standard Code for Information Interchange.

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

MD4: Nombre del formato que da Id Software al juego Quake 4.

SDK*:* Software Development Kit***.***

SIMPRO: Simuladores Profesionales.

Glosario de Términos

A

Aplicaciones: Cada uno de los programas que, una vez ejecutados, permiten trabajar con el ordenador.
Animación: Simulación de un movimiento creada por la muestra de una serie de imágenes o fotogramas.
Algoritmos: Conjunto finito de pasos o instrucciones que se deben seguir para realizar una determinada tarea.

B

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').

D

DirectX: es una colección de APIs creadas para facilitar tareas relacionadas con la programación de juegos en la plataforma Microsoft Windows.

F

Frame: cada uno de las imágenes que componen una animación.

I

Interpolación: algoritmo matemático que a partir de varios puntos en el espacio, describe una función que contiene a los puntos intermedios

J

Jerarquía de huesos: Es una lista ordenada de huesos teniendo en cuenta algún tipo de prioridad.

M

Malla: Forma de representar un modelo a partir de polígonos. Colección de vértices, aristas y polígonos conectados, de forma que cada arista es compartida como máximo por dos polígonos.

Matriz: Arreglo de elementos.

Módulo: Pieza o conjunto unitario de piezas que se repiten o encajan en una construcción de cualquier tipo.

Matriz de transformación: Matrices definidas para calcular nuevas coordenadas a partir de las ya existentes según una determinada transformación gráfica (rotación, trasladación, escalado y reflexión).

Modelo: Prototipo para la animación.

O

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

P

Poligonal: Del polígono o relativo a él.

Polígono: Figura geométrica plana limitada por segmentos rectos consecutivos no alineados, llamados lados.

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

S

Scene ToolKit: Herramienta de desarrollo de aplicaciones de realidad virtual.

Sistema de Realidad virtual: sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real.

Simulador: Programa computacional basado en cálculos y modelos estadísticos, usados para representar un escenario determinado.

Skin: Elemento externo de un personaje, también hace referencia a la relación de los vértices y los huesos de un personaje.

T

Tarjeta gráfica: Es una tarjeta de circuito impreso encargada de transformar las señales eléctricas que llegan desde el microprocesador en información comprensible y representable por la pantalla del ordenador.

V

Vector: Cantidad que expresa magnitud y dirección.

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

W

Weight: Peso o influencia que ejerce un hueso sobre un vértice.