UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DE UN FORMATO PROPIETARIO DE MODELOS 3D PARA LA ENSEÑANZA

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

LUIS ALBERTO BOBADILLA SOTELO

DIRECTOR DE TESIS
Ing. Genaro Andrés Garrido Lazcano

2010

Agradecimientos

Agradezco ante todo a Dios por regalarme el milagro de la vida, la bendición de mis padres y amigos y la oportunidad de realizarme día con día como persona, hijo y profesionista.

A mi madre, razón y causa de mi desarrollo como persona, por enseñarme mediante su inquebrantable esfuerzo que era posible superar las situaciones más adversas y difíciles, por nunca dejar que me rindiera ante los problemas y por llenar mi vida con su amor. Sin ella no sería la persona que hoy soy. ¡Gracias!

A mi padre, por apoyarme durante todos estos años y mostrarme con el ejemplo la clase de persona que quiero llegar a ser. Gracias por todo el cariño y los buenos deseos que sé, están depositados en mí.

A mi familia por todo el apoyo que recibí de ustedes.

A mis amigos, la familia que yo escogí. Por estar conmigo en las buenas y en las malas, por escucharme en las ocasiones que tuve un problema y por las ocasiones en las que en su complicidad me metí en otros más.

A mi Universidad, mi "Alma Mater", por darme excelentes profesores y estudios que me permitirán alcanzar mis metas en la vida.

A mis sinodales, por el tiempo que invirtieron para que el presente trabajo sea el reflejo de mi esfuerzo y dedicación al estudio en los últimos años.

A todos aquellos no mencionados aquí pero que pusieron su granito de arena en la playa de mi vida para hacer de mí una mejor persona.

ÍNDICE GENERAL

INTRODUCCIÓN V

I. DESCRIPCIÓN (DEFINICIÓN DEL PROBLEMA) V
II. JUSTIFICACIÓN VI
III. MÉTODO VI
IV. DESCRIPCIÓN DEL CONTENIDO VII

CAPITULO I. ANTECEDENTES TEÓRICOS. 1

1.1 GRÁFICAS POR COMPUTADORA 1
1.2 ANTECEDENTES HISTÓRICOS 2
1.3 COMPUTACIÓN GRÁFICA EN LA ACTUALIDAD 4
1.3.1 DISEÑO ASISTIDO POR COMPUTADORA 5
1.3.2 EDUCACIÓN 6
1.3.3 VISUALIZACIÓN 7
1.3.4 ENTRETENIMIENTO 9
1.4 COMPUTACIÓN GRÁFICA EN EL FUTURO 10
1.4.1 ESTEREOSCOPÍA 11
1.4.1.1 IMÁGENES 14
1.4.1.2 LENTES 18
1.4.1.2.1 ANAGLYPH 18
1.4.1.2.2 POLARIZADOS 19
1.4.1.2.3 SHUTTLER 20
1.4.2 REALIDAD EXTENDIDA 21

CAPITULO II. ANÁLISIS DE HERRAMIENTAS DE SOFTWARE EXISTENTES. 23

2.1 PROGRAMAS COMERCIALES 23
2.1.1 AUTODESK MAYA 23
2.1.2 AUTODESK 3D STUDIO MAX 26
2.2 PROGRAMAS DE SOFTWARE LIBRE 28
2.2.1 BLENDER 28

CAPÍTULO III. ANÁLISIS DE FORMATOS EXISTENTES 31

3.1 FORMATO OBJ 32
3.1.1 MESH 35
3.1.2 MATERIALES 36
3.2 FORMATO 3DS 40
3.2.1 INFORMACIÓN MÁS DETALLADA DEL CONTENIDO DE UN ARCHIVO 3DS 43

CAPÍTULO IV. EL FORMATO PROPIETARIO BTO 53

4.1 DESCRIPCIÓN 53
4.2 CONSIDERACIONES DEL DISEÑO 62
4.3 CONSIDERACIONES BÁSICAS DEL FORMATO 62
4.4 VENTAJAS SOBRE OTROS FORMATOS 64
4.4.1 SOBRE OBJ 64

4.4.2 Sobre 3DS 64
4.5 Comparativa de formatos 65
4.6 Código BTO y Visor BTO 73
4.6.1 Métodos de Carga del archivo 74
4.6.2 Métodos de Dibujo 74
4.6.3 Algoritmos para detección de colisiones 75
4.6.3.1 Esfera 76
4.6.3.2 Cilindro 77
4.6.3.3 Caja orientada a los ejes 77
4.6.3.4 Rayo 78
4.6.3.5 Algoritmos para detección de colisiones aplicados 78
4.6.3.6 Campo de acción 79
4.6.3.7 Caja Orientada a los Ejes 81
4.6.3.8 Esfera de colisión. 82
4.6.3.9 Grupos colisionables 83

CAPÍTULO 5. CAMPOS DE APLICACIÓN 85

5.1 Campos aplicables 85
5.2 Uso como soporte a otro software. 85
5.3 Visualización - Visor y editor de mundos 86
5.3.1 Enseñanza – Mano Amiga 96
5.3.2 Realidad Aumentada – Augmented Reality Demo 97
5.3.3 Recorridos Virtuales – Visor BTO 98
5.3.4 Videojuegos 99

CONCLUSIONES 101

BIBLIOGRAFÍA 105

1. **Datos de la alumno**
   Bobadilla
   Sotelo
   Luis Alberto
   55507095
   Universidad Nacional Autónoma de México
   Facultad de Ingeniería
   Ingeniero en Computación
   09903656-8

2. **Presidente**
   Dr. Boris Escalante Ramírez

3. **Vocal**
   Ing. Genaro Andrés Garrido Lazcano

4. **Secretario**
   M.C. Gabriel Mauricio Álvarez Medina

5. **Primer Suplente**
   Ing. Tanya Itzel Arteaga Ricci

6. **Segundo Suplente**
   Ing. Luis Sergio Valencia Castro

7. **Tesis**
   Diseño De Un Formato Propietario De Modelos 3D Para La Enseñanza.
   105 p
   2010

INTRODUCCIÓN

I. Descripción (Definición del problema)

En la actualidad la mayoría de los programas gráficos cuentan con diferentes formatos tridimensionales para la presentación de sus modelos al usuario final.

Sin embargo la aplicación de dichos modelos está supeditada a ese programa en específico, lo que en la mayoría de los casos provoca que el conocimiento de la información que proporciona el formato en que se encuentra dicho modelo este cerrado al público general y sea inaplicable en el desarrollo de software propio.

Haciendo de lado la falta de conocimiento de la información contenida en estos formatos, se presenta también el problema de la “sobre-información”. Estos formatos están orientados a su uso en aplicaciones comerciales específicas, lo que hace que contengan información innecesaria para el usuario.

Existen formatos abiertos al público pero presentan limitaciones al momento de aplicarlos en un programa propio y en muchos casos se requiere de archivos adicionales para hacer lo que un formato comercial hace por sí mismo.

La elaboración de un formato propietario que pudiera englobar las funciones de un formato abierto y las de un formato comercial, evitando la “sobre-información” que estos presentan, ayudaría en el desarrollo de aplicaciones propias al simplificar el

proceso de la presentación en pantalla al programador. Así mismo provocaría la reducción en tamaño y cantidad de los archivos de modelos en dichas aplicaciones.

II. Justificación

El presente estudio representa una respuesta a las necesidades específicas que el programador gráfico requiere. Es con el desarrollo de este formato que se espera minimizar el tiempo de creación de aplicaciones y programas, y así mismo simplificar el tiempo en el que el programador carga, dibuja y presenta modelos en pantalla.

La utilidad de este formato radica en la enseñanza al público inexperto, por lo que puede ser utilizado con fines pedagógicos. Así mismo se pretende desarrollar código que pueda ser utilizado para desarrollar aplicaciones propietarias de diferentes áreas de la ingeniería sin la necesidad de que el usuario se preocupe por el desconocimiento de cómo desplegar gráficos en pantalla.

III. Método

Mediante el estudio de diferentes formatos existentes en la actualidad, se planea el proponer y desarrollar un formato de archivos tridimensionales propio. Mismo que incluirá las características de un formato cerrado al público pero la facilidad de uso que proporciona un formato abierto.

Este formato proporcionará claramente la información que el usuario requiere para elaborar su programa sin meterse en la complicación de comprender un formato cerrado.

IV. Descripción del contenido

Capítulo I. Este capítulo trata de la historia de las gráficas por computadora, del porque y el cómo es que las gráficas por computadora llegaron a ser parte de nuestro día a día, seamos expertos o neófitos. Así mismo analiza diferentes métodos de proyección de gráficas en el mundo actual.

Capítulo II. En este capítulo se discuten algunas de las herramientas comerciales y gratuitas que son empleadas en la actualidad para producir gráficas que son empleadas en programas o aplicaciones con el fin de llevar las gráficas por computadora al usuario final.

Capítulo III. En este capítulo se analizaran algunos de los formatos existentes para el manejo de graficas por computadora, formatos destinados al uso de modelos dentro de diferentes aplicaciones, como lo son el diseño o la animación.

Capítulo IV. Este capítulo explica las consideraciones que se tomaron en cuenta para el desarrollo del formato propietario, así como las ventajas que tiene sobre otros formatos existentes en la actualidad.

Capítulo V. En este capítulo se verán aplicaciones prácticas que se le pueden dar al formato BTO. Así mismo se ejemplifican diferentes funciones y roles que el formato puede tomar.

CAPITULO I. ANTECEDENTES TEÓRICOS.

1.1 Gráficas por computadora

Para hablar de computación gráfica es necesario entender el concepto de gráficas, que es la representación visual de datos. Incluye la visualización de puntos, líneas y superficies pero no está limitado a ello. Estos elementos en su conjunto conforman imágenes que representan visualmente un concepto o un conjunto de valores.

En la actualidad todas estas gráficas que son procesadas por equipo electrónico, son conocidas como gráficas por computadora, y pueden ser obtenidas mediante el modelado manual de gráficos, por medio de escáneres tridimensionales o mediante software especializado.

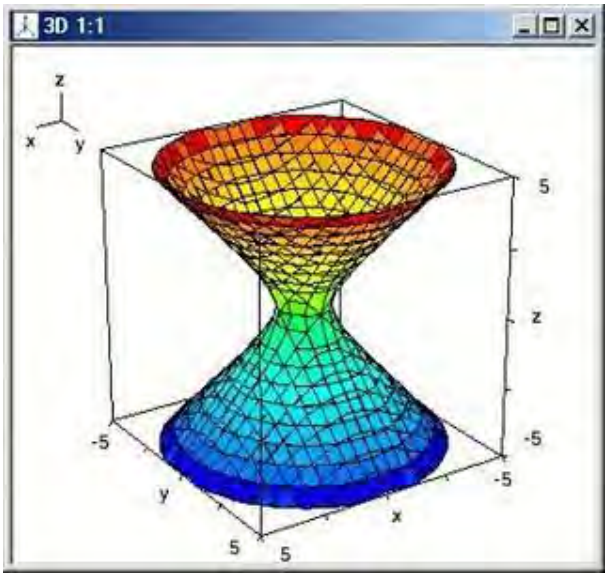


Figura 1.1 - Derive 5 tomada de http://artur-racp.blogspot.com/ publicada en 2009/AGO/26

Esta representación visual tiene aplicaciones en todas las áreas del conocimiento, ya sea por medio de diagramas de barras, imágenes con datos estadísticos, modelos de objetos reales en un espacio bidimensional, o incluso en medios tridimensionales.

Las gráficas ayudan a visualizar información de una manera sencilla y pueden resumir grandes cantidades de datos en una sola imagen.

1.2 Antecedentes históricos

A partir de la cuarta generación de computadoras se hizo posible el interactuar con ambientes gráficos, antes de esto los datos obtenidos mediante el uso de las computadoras solo eran apreciados en medios físicos. El usuario ingresaba datos a la computadora, pero no tenía medios para ver el proceso antes de ser terminado de procesar, y estos procesos tomaban demasiado tiempo en ejecutarse.

Para facilidad del usuario y gracias al desarrollo de nuevas tecnologías, fue posible incluir en las computadoras una interfaz de base texto. Esto permitió a los usuarios aproximar su lenguaje (de alto nivel) al de las máquinas (de bajo nivel).

La introducción de las GUI´s (del inglés Graphical User Interface, o Interfaz Gráfica de Usuario) permitió que el usuario común utilizara las computadoras sin requerir una capacitación exhaustiva previa a su uso. Simplificando enormemente la interacción Hombre-Máquina.

Las interfaces gráficas, surgen por la necesidad de hacer las computadoras más accesibles al usuario común. Pues anterior a esto el uso de las computadoras requería conocimientos de programación y de la interfaz de línea de comandos (de base texto).

Los primeros desarrollos de GUI's se hicieron en el Stanford Research Institute (SRI) bajo el liderazgo de Douglas Engelbart en 1968, quien soñaba con hacer que las computadoras estuvieran al alcance de todos y no solamente de expertos, él creía que las computadoras serían el medio para expandir el potencial humano. Es de esta investigación donde nacen conceptos conocidos hasta el día de hoy, conceptos como ventana, casillas de verificación, botones de radio, menús y punteros de mouse, mismos que fueron ampliados y comercializadas por la compañía Xerox en 1973 con el desarrollo de la Xerox Star. La primera computadora en integrarlos de manera comercial.

Figura 1.2 - GUI de Xerox Star

Para 1984 la compañía Apple utilizando los conceptos de GUI desarrollados por el SRI y mejorados por Xerox desarrolla la Apple Lisa con el lema: ”Pretendimos hacer una computadora tan simple de manejar como una tostadora”. Es a partir de este momento que las puertas del mundo de la computación fueron abiertas al usuario común sin requerir una amplia capacitación previa para el uso de las computadoras.

Para 1985 aparece Windows 1.0 por parte de la compañía Microsoft, con una GUI prácticamente idéntica a la de Apple en su Mac OS, por lo que la compañía Apple interpuso demandas contra Microsoft, sin embargo ambas compañías copiaron los conceptos básicos de Xerox.

En las computadoras de orientación gráfica, casi todas las aplicaciones son WYSIWYG (What You See Is What You Get, lo que se ve en pantalla es lo que se obtiene en papel impreso), sin embargo esto no siempre es necesario o en muchos casos deseable. El diseño asistido por computadora, aplicado a animación por ejemplo, es una muestra de que no siempre es necesario tener una impresión en papel de lo que tenemos en pantalla.

1.3 Computación Gráfica en la actualidad

Actualmente es muy difícil concebir el uso de la computadora sin gráficas. Usualmente pensar en una computadora implica negocios, educación, comunicaciones, entretenimiento y muchos otros conceptos que, sin las gráficas, tomaría mucho tiempo, si es que fuera del todo posible, entender.

Estamos acostumbrados a juzgar una película, por ejemplo, por los efectos visuales que esta conlleva, o ¿Qué junta de negocios estaría completa sin una gráfica que refleje avances obtenidos? Las gráficas por computadora actualmente son parte del día a día.

La computación gráfica en la actualidad tiene aplicaciones en numerosas ramas del saber, entre ellas, las más destacadas:

Ingeniería: Diseño, Análisis, Dibujo, Control.

Otras áreas del saber: Arte, Ciencias, Educación.

1.3.1 Diseño asistido por computadora

Es el uso de la computadora como soporte de diseño, particularmente en el dibujo técnico e ingenieril, como parte de un producto o edificaciones.

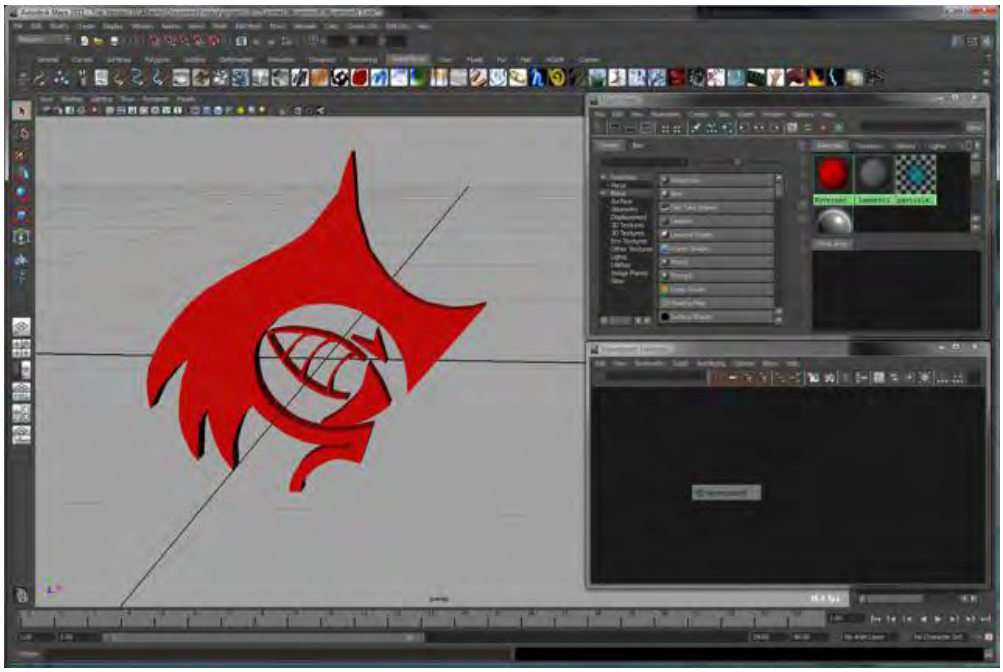


Figura 1.3 - Autodesk Maya 2011

Dentro del amplio mundo del Diseño asistido por computadora (CAD, por sus siglas en inglés - Computer Asisted Design). Encontramos los siguientes subcampos:

MCAD - Mechanical Computer Aided Design, es el CAD especializado en las partes mecánicas.

CAE - Computer Aided Engineering. Es el uso de información tecnológica que ayuda a ingenieros en tareas como el análisis, simulación, diseño, manufactura, planeación y reparación asistido por computadora.

CAM - Computer Aided Manufacturing. Se refiere al uso de herramientas de software que asisten a ingenieros y mecánicos en la manufactura y elaboración de prototipos de componentes. CAM es una herramienta programacional que hace posible la elaboración de diseños por medio del CAD.

GIS - Geographic Information System. Captura, almacena, analiza, maneja y presenta datos que se refieren a una locación. En un sentido más estricto el termino describe cualquier sistema de información que integra, almacena, edita analiza, comparte y muestra información geográfica.

AEC - Architec, Engineer, and Constructing Software.

CAD/CAM - Es el software que conjuga el Diseño y la Manufactura asistidas por computadora.

1.3.2 Educación

Sin duda la visualización de un concepto ayuda a su fácil entendimiento. Plasmar una idea en un medio bidimensional no es una tarea fácil, el explicarle a un alumno (aun de

niveles superiores de educación) conceptos geométricos tridimensionales por ejemplo, representa un reto si no se cuentan con las herramientas adecuadas.

Las gráficas por computadora permiten abreviar en una sola imagen cientos de datos; más aún una gráfica animada permite al estudiante comprender lo que pasa en un ambiente tridimensional aunque el medio de presentación sea plano.

Figura 1.4 - www.nz.wacom-asia.com/education

En el ambiente educativo, las computadoras y las gráficas por computadora no solo ayudan a la enseñanza, sino también en el aspecto económico. En la enseñanza de la arquitectura por ejemplo, se pueden elaborar levantamientos o maquetas en programas como AutoCAD, economizando el método de entrega de maquetas y en algunos casos haciéndolas obsoletas.

1.3.3 Visualización

Retomando un poco el concepto descrito anteriormente, hay situaciones en las que no es posible la visualización de un objeto físicamente, ya sea por motivos económicos, físicos o inviables.

El estudio en la medicina de las partes del cuerpo humano es vital, pero no siempre tienen las universidades y escuelas medios para conseguir, órganos frescos. Programas de educación como “A.D.A.M.” hacen posible que el alumno tenga un acercamiento a ellos sin tenerlos frente a frente físicamente.

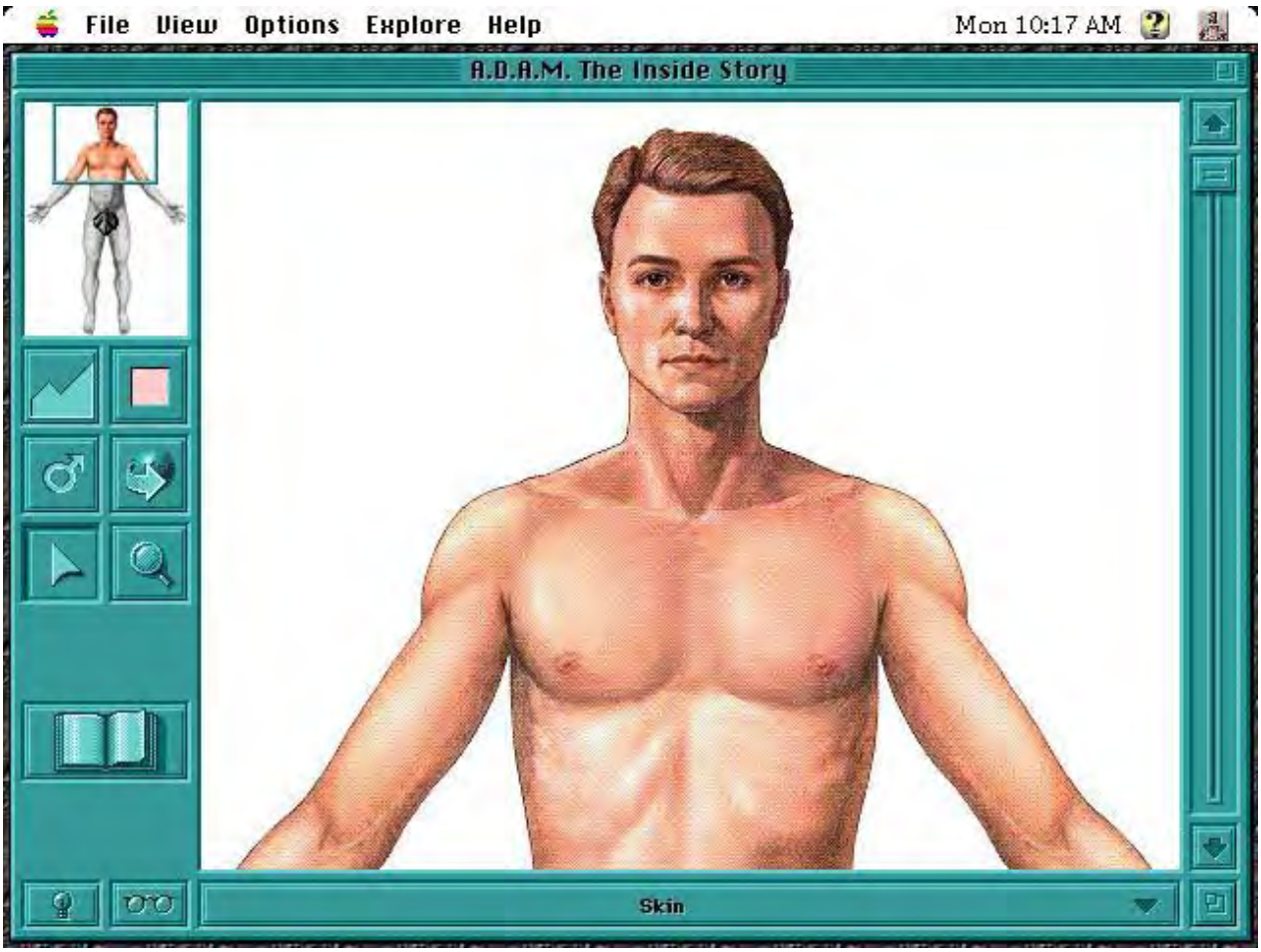


Figura 1.5 - Programa A.D.A.M.

En el ámbito profesional recrear visualizaciones de prototipos es vital, no se pueden crear prototipos cada vez que se hace un cambio en el producto, resultaría demasiado costoso. Esto resulta especialmente cierto en la planeación del producto. Y en el caso específico de la arquitectura, inviable. Si bien es cierto que se pueden hacer representaciones y levantamientos en papel como se ha hecho a través del tiempo, también es cierto que la representación en computadora acelera el proceso drásticamente.

1.3.4 Entretenimiento

Pensar en el entretenimiento actual sin las gráficas por computadora, es realmente pensar en un retroceso. Las gráficas tienen décadas en la industria del entretenimiento, ya sea en medios pasivos (televisión, cine, etc.) como en activos (videojuegos, recorridos virtuales, etc.).

Sin duda las gráficas juegan un papel vital en la industria del entretenimiento, no es de sorprendernos, que la industria gaste millones de dólares en equipos de vanguardia capaces de generar estas gráficas de manera rápida y precisa.

Respecto a la industria del cine, prácticamente ninguna película en la actualidad se realiza sin la ayuda de computadoras, pues estas están involucradas desde la filmación hasta el proceso de edición, minimizando costos y maximizando ganancias.

Figura 1.6 - 20th Century Fox, Avatar 2009

1.4 Computación Gráfica en el futuro

En un futuro no muy lejano, las gráficas por computadora estarán presentes en prácticamente todos los campos del saber humano. No es de sorprender que más y más a menudo las interfaces gráficas formen parte de los aparatos electrodomésticos, hemos visto ya refrigeradores con pantallas y controles sensibles al tacto. Con la actual tendencia en GUI´s, las interfaces “touchscreen”, el usuario común puede manipular aparatos que en otros tiempos hubieran requerido conocimientos informáticos de mayor nivel, o que son drásticamente simplificados de su uso previo de botones (como las cámaras digitales, teléfonos celulares y videojuegos por ejemplo) a pocos toques con el dedo.

Pero la actual tendencia nos hace pensar en más que GUI´s, la velocidad con que avanza la tecnología y se abaratan los componentes electrónicos (que años atrás eran

incosteables para aparatos comunes) están marcando una clara diferencia en lo que los desarrolladores pueden hacer.

Lo que años atrás era el sueño de Douglas Engelbart, hoy es una realidad.

La industria se enfoca cada día más en la proyección de imágenes tridimensionales a través de medios bidimensionales. En un futuro no muy lejano, las gráficas planas quedarán en un segundo plano, reducidas en mayor medida a GUI´s, el nuevo entretenimiento vendrá en formatos tridimensionales; televisiones, videojuegos, cámaras fotográficas y demás productos comerciales de entretenimiento verán añadida una nueva dimensión espacial.

Es por esto que todas las personas involucradas en las gráficas por computadora deben estar preparadas al cambio. Esto no quiere decir que las gráficas bidimensionales desaparecerán, al contrario serán la base de muchos productos, sin embargo, la industria del entretenimiento será la más afectada por este cambio, lo que conllevará a que la mayor parte de las gráficas por computadora se enfoquen al ambiente 3D.

1.4.1 Estereoscopía

La estereoscopia es la técnica que permite crear la ilusión de profundidad de una imagen bidimensional, es un medio sintético para reproducir lo que nuestros ojos hacen en realidad por sí mismos.

La manera que percibimos las imágenes tridimensionales del mundo, es por medios que solo perciben información bidimensional. La visión monocular (utilizando solamente un ojo) presenta al cerebro humano información en 2 dimensiones, pero es por medio de experiencia e información intuitiva que logramos entender las imágenes percibidas por un solo ojo con información de profundidad.

Entre los métodos intuitivos de los que dispone el cerebro para obtener información tridimensional, se encuentran la distribución de luces y sombras, esto genera contrastes que aportan la información de relieve y volumen, un circulo puede convertirse para el cerebro en una esfera con los cambios correctos de sombras; la superposición de imágenes permite al cerebro identificar cual es el objeto más alejado, si uno de ellos se interpone a otro (lo tapa en alguna parte); la perspectiva ayuda al cerebro a identificar el objeto más alejado si, de dos objetos que presumen tener el mismo tamaño uno de ellos aparenta ser más pequeño; todo esta información, en conjunto con el movimiento de los objetos o el observador ayudan al cerebro a entender la distancia relativa entre ellos, jugando un papel fundamental en la transformación de imágenes planas a volumétricas.

Es por esto que explotando la capacidad del cerebro para obtener información tridimensional a partir de luces, sombras, superposición y perspectiva, se pueden crear imágenes bidimensionales por computadora que el cerebro perciba como tridimensionales. Algoritmos de “headtracking” (localización de la cabeza del espectador mediante cámaras digitales y otros medios) permiten a una computadora acomodar una escena en pantalla de tal suerte que el espectador determine

profundidad. Desafortunadamente este tipo de imágenes solo prestan información para un solo espectador, dado que la imagen no se puede acomodar para todos los usuarios simultáneamente.

Por ello es que con el uso de ambos ojos, el cerebro utiliza los dos canales de información bidimensional visual disponibles para mezclar la información obtenida y producir una nueva y más completa información de profundidad, creando la sensación de volumen y distancia.

Sir Charles Wheatstone en 1840 fue el primero en explotar este conocimiento produciendo imágenes que hoy conocemos como estereogramas. Su técnica consiste en presentar imágenes distintas a cada ojo, lo que conlleva a que el cerebro trate de obtener información tridimensional de ellas (como lo hace de forma natural). Esta técnica funciona para más de un espectador simultaneo, pues en realidad no importa su posición respecto al medio que transmite la imagen, siempre la percibirá como tridimensional, esto solo está limitado a que el espectador perciba la escena en un ángulo en el que el cerebro logre identificarse con la posición de la cámara que obtuvo la imagen. Por lo mismo no está limitado a su proyección por medios electrónicos, los estereogramas pueden ser apreciados desde medios impresos, sin requerir la ayuda de medios especiales como lentes de tercera dimensión. Sin embargo el no utilizarlos causa estrés en los ojos y es por ello que los medios tridimensionales actuales, basados en estereogramas utilizan lentes, reduciendo el cansancio en el ojo humano.

1.4.1.1 Imágenes

Las imágenes en 3D son conocidas como estereogramas, y podemos apreciarlas en tres dimensiones gracias al principio de estereoscopía, sin embargo no hay solamente una manera de llevar información a cada ojo.

Entre las diferentes técnicas para hacerlo correctamente se encuentran:

La técnica de separación física, consiste en físicamente separar mediante un objeto, lo que cada ojo puede ver. Este principio es aplicado en los dispositivos conocidos Viewmasters, ellos presentan un disco con imágenes en película positiva preparadas en torno a un disco de tal manera que el visor presente dos imágenes de un mismo fotograma pero con diferente ángulo de visión. La desventaja de esta técnica es que los ojos deben estar físicamente cubiertos para no ver nada más, y por lo mismo es la única técnica de estereogramas que solo puede ser apreciada por un usuario a la vez en su aplicación tradicional. Sin embargo también presenta la imagen con mejores colores de todas las técnicas de estereogramas. Así mismo la aplicación de la técnica de separación física ha tomado un nuevo giro en los últimos meses al aplicar la técnica de separación física por barrera, esta consiste en hacer que el proyector directamente envíe la información correcta a cada ojo del espectador mediante líneas de pixeles distribuidos en canaletas verticales, una para el ojo izquierdo y otra para el derecho, esto permite que más de un usuario pueda disfrutar de la experiencia tridimensional mientras todos los espectadores se encuentren en lo que se denomina “El área de visibilidad dulce”, como ejemplo de esto tenemos el nuevo Nintendo 3DS y nuevas televisiones aun en desarrollo. El problema con esta variación de la técnica es que no

permite que el usuario incline su cabeza en más de un determinado número de grados, pues a mayor sea ángulo de inclinación mayor es la perdida de efecto de 3D.

Figura 1.7 y 1.8 - Viewmaster con discos. Y Nindendo 3DS http://g4tv.com

La técnica de los ojos cruzados, consiste en colocar las imágenes con sus dos perspectivas una junto a la otra y hacer que el usuario entrecruce sus ojos haciendo coincidir una imagen sobre la otra. Esta técnica, sin contar la técnica de separación física, es la que presenta la imagen con los colores más claramente definidos de todas las técnicas de estereogramas. Sin embargo es también la que más estrés presenta en el ojo humano y ha probado ser una técnica difícil de utilizar para el usuario común, pues en muchos casos no se logra conseguir el ángulo de cruce correcto de las imágenes. A continuación se presenta un estereograma que utiliza la técnica del ojo cruzado, para poder verse en tres dimensiones es necesario que el lector cruce sus ojos hasta hacer coincidir los dos puntos sobre la imagen, se recomienda no se entrecrucen los ojos por mucho tiempo, pues puede causar dolor ocular.

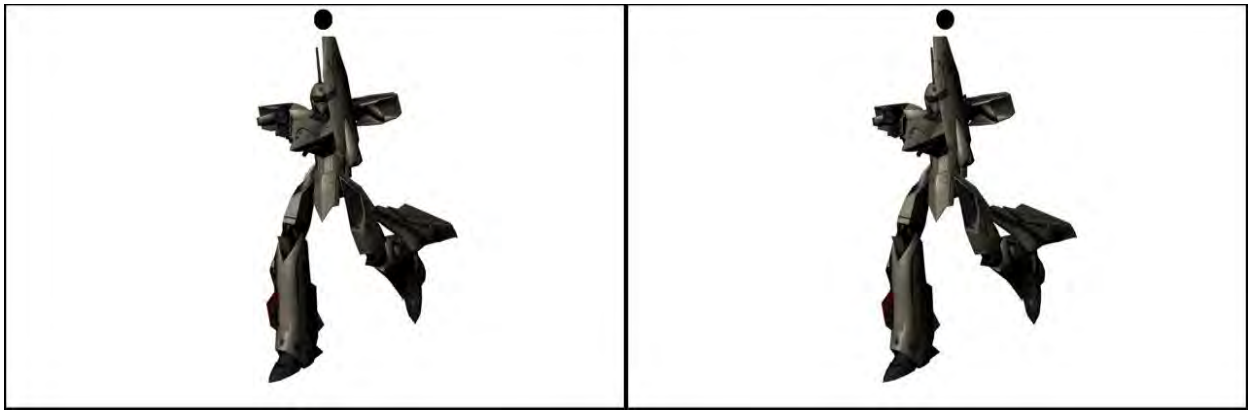


Figura 1.9 – Imagen estereoscópica de ojos cruzados.

Técnica de separación por color, esta técnica consiste en separar la imagen mediante los espectros de luz y la imagen recibe el nombre de estereograma anaglyph (o también traducido como anáglifo). Consiste en filtrar la imagen mediante colores, al momento de producir la imagen, en la que va pensada para el ojo izquierdo se filtra en un color y la del derecho se filtra en otro, el usuario debe entonces utilizar lentes con el mismo tipo de filtros de color con el fin de reproducir la imagen. Esta es la técnica más económica (que no causa dolor en los ojos) desarrollada, sin embargo es también la técnica que presenta mayor distorsión de colores (debido a los filtros). Esta técnica es ilimitada en el número de espectadores y puede ser apreciada en cualquier medio impreso. Otra ventaja de esta técnica es que físicamente ocupa la mitad del espacio de dibujo de las de separación física. A continuación se presenta una imagen anaglyph con separación de canales rojo y cyan (solo apreciable con lentes anaglyph rojo cyan).

Figura 1.10 – Imagen estereoscópica en anaglyph.

Técnica de separación por polarización, esta técnica funciona de manera similar a la de separación de color. La imagen se produce con filtros y el espectador la ve a través de filtros, esta técnica cuenta con la propiedad de que la imagen no pierde color (pero

en la mayoría de los casos pierde brillo, aunque esto puede no presentar un gran problema). La polarización puede ser a través de líneas horizontales y verticales o mediante filtros de círculos en el sentido de las manecillas del reloj y contra sentido.

Técnica de separación por tiempos, esta es sin duda la técnica actual que presenta mejores resultados. La técnica en sí misma no presenta ninguna distorsión de color o brillo, desafortunadamente aun los mejores lentes “shuttler” (requeridos para esta técnica) presentan perdida de brillo (como se verá más adelante). Esta técnica consiste en tapar un ojo y mostrar la imagen del ojo contrario e intercambiar la imagen y el ojo sucesivamente. La desventaja de esta técnica es que no es posible presentarla en un medio impreso, por lo que solamente es aplicable a imágenes en movimiento mediante proyectores. Además que el equipo para reproducirlo debe de ser especial y actualmente es algo caro. Pero los resultados obtenidos son realmente superiores a los de las otras técnicas pues presenta las ventajas de las técnicas de separación física y las de separación por polarización, sin sus desventajas.

1.4.1.2 Lentes

A continuación se explicará el uso de los lentes requeridos por las técnicas anteriormente descritas.

1.4.1.2.1 Anaglyph

Los lentes Anaglyph, son requeridos por la técnica de separación de color. El tipo de lente más común es el Rojo-Cyan, pero últimamente se ha optado por utilizar un tipo de lente con filtros en una tonalidad especial de Verde y Magenta, que presentan menor distorsión de colores para la piel humana, lo que permite no perder tanto la sensación

de realismo de la escena. Son los lentes estereoscópicos más económicos que existen en la actualidad.

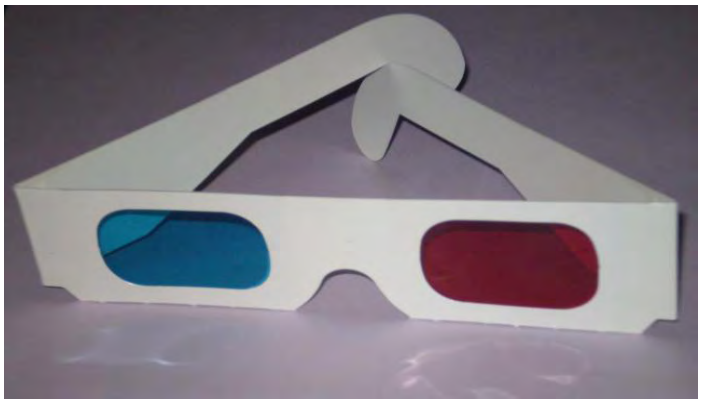


Figura 1.11.1 - Lentes estereoscópicos anaglyph Rojo-cyan

1.4.1.2.2 Polarizados

Como se describió anteriormente los lentes polarizados no presentan pérdida de color, pero sí de brillo, dado que el proceso de polarización hace a estos lentes tener un tono obscuro. Pueden ser de tipo polarizado lineal o circular, sin embargo, ambas polarizaciones tienen la desventaja de que el usuario tiene que verla perfectamente vertical, si el usuario inclina la cabeza la polarización deja de funcionar. La polarización por filtros circulares presenta este problema en menos medida, pero aun así es persistente.

Figura 1.11.2 - Lentes polarizados

1.4.1.2.3 Shuttler

Este es el tipo de lentes más caro del mercado actual, pero es también el que refleja los mejores resultados. Consiste en dos placas de vidrios lcd y un sensor infrarojo, que determina mediante la recepción de señales de un emisor sincronizado con el monitor o retroproyector, que imagen se está mostrando para saber que ojo debe de tapar, envía una corriente al vidrio de lcd (del inglés liquid cristal display, o pantalla de cristal líquido) que obscurece el vidrio una fracción de segundo, suficiente para tapar la imagen que no debe de ver el ojo y este proceso se repite indefinidamente para lograr la sensación de profundidad. Al trabajar con vidrios lcd se tiende a obscurecer la imagen, pero es la opción más viable en términos de velocidad requerida para hacer un efecto realista. El precio de los lentes es un factor determinante para que no sean tan sencillos de adquirir, en el caso específico de los lentes nVidia 3D visión mostrados en la siguiente imagen su precio es de 200 dólares americanos y requieren de una tarjeta gráfica específica y un monitor que funcione a 120Hz por segundo (60 imágenes para cada ojo en un segundo).

Figura 1.11.3 - *Lentes 3dVision de la compañía nVidia.*

1.4.2 Realidad Extendida

Otra de las aplicaciones que tendrá más fuerza para la computación gráfica es la realidad extendida, que consiste en expandir la información que el usuario puede tener de un medio visual en un ambiente real. Un caso concreto de realidad extendida son los displays de los aviones que muestran no solo la imagen real de lo que está viendo la cámara, sino también la triangulación de la pista de aterrizaje y el nivel del horizonte. Este tipo de tecnología ya es actualmente accesible al público general, y en algunos casos se ha aplicado a la industria del videojuego, como en el caso del juego Eye of Judgment para la consola Playstation 3, donde el jugador coloca una carta de juego frente a una cámara y la consola lee la información contenida en códigos de barras de la carta, proyectando en la televisión la imagen del mundo real y superpuesto un modelo tridimensional que representa a un personaje de esa carta.

Figura 1.12 - Eye of Judgment, Sony Computer Entretainment, Playstation 3

CAPITULO II. ANÁLISIS DE HERRAMIENTAS DE SOFTWARE EXISTENTES.

Actualmente en el mundo del desarrollo de gráficas por computadora existen diferentes empresas dispuestas a cubrir las diversas necesidades del usuario. Y existe tanto software libre como propietario.

Dentro del giro del software propietario encontramos a Autodesk una compañía que poco a poco se ha convertido en la empresa líder de desarrollo de gráficas por computadora, gracias al desarrollo de software y a la fusión con otras empresas del mercado. Actualmente Autodesk tiene en su repertorio, programas muy populares de diseño y los importantes de la industria como son AutoCad, Maya, Inventor y 3D Studio Max, entre otros.

Dentro del software libre es sin duda el programa Blender, de Blender Foundation, el que se corona como el mejor software de diseño y animación.

2.1 Programas comerciales

2.1.1 Autodesk Maya

Maya es quizás el software más utilizado en la industria del entretenimiento. Es utilizado, en combinación con el programa RenderMan por la afamada compañía Pixar, creadora de exitosos proyectos como “Toy Story”, “Finding Nemo”, “Monsters Inc” o “The Increadibles”.

Es un software de creación de gráficos en 3D, efectos visuales y animación. Surgió del desarrollo de Power Animator y de la fusión de Alias y Wavefront, dos empresas canadienses dedicadas a los gráficos generados por computadora. Más tarde Silicon, absorbió a Alias Wavefront, y finalmente terminó en las manos de Autodesk.

Dentro de las características principales que podemos encontrar en Maya tenemos:

- Potentes herramientas para construir objetos utilizando modelado por polígonos, NURBS o subdivisión.
- Simulación de efectos de partículas. Simulador de efecto de fluidos.
- Kinemática inversa
- Simulación de ropa
- PaintEffects, herramientas de pintado de 2D dentro de un entorno de renderizado 3D
- Simulador de pelaje.
- Simulador de cabello, efecto fotorealístico de cabello humano implementando curvas y PaintEffects.
- Efectos físicos, colisiones, deformación, presión, gravedad, fuerza del aire, etc.
- Renderizador nativo de “Mental Ray”
- Interfaz completamente configurable.

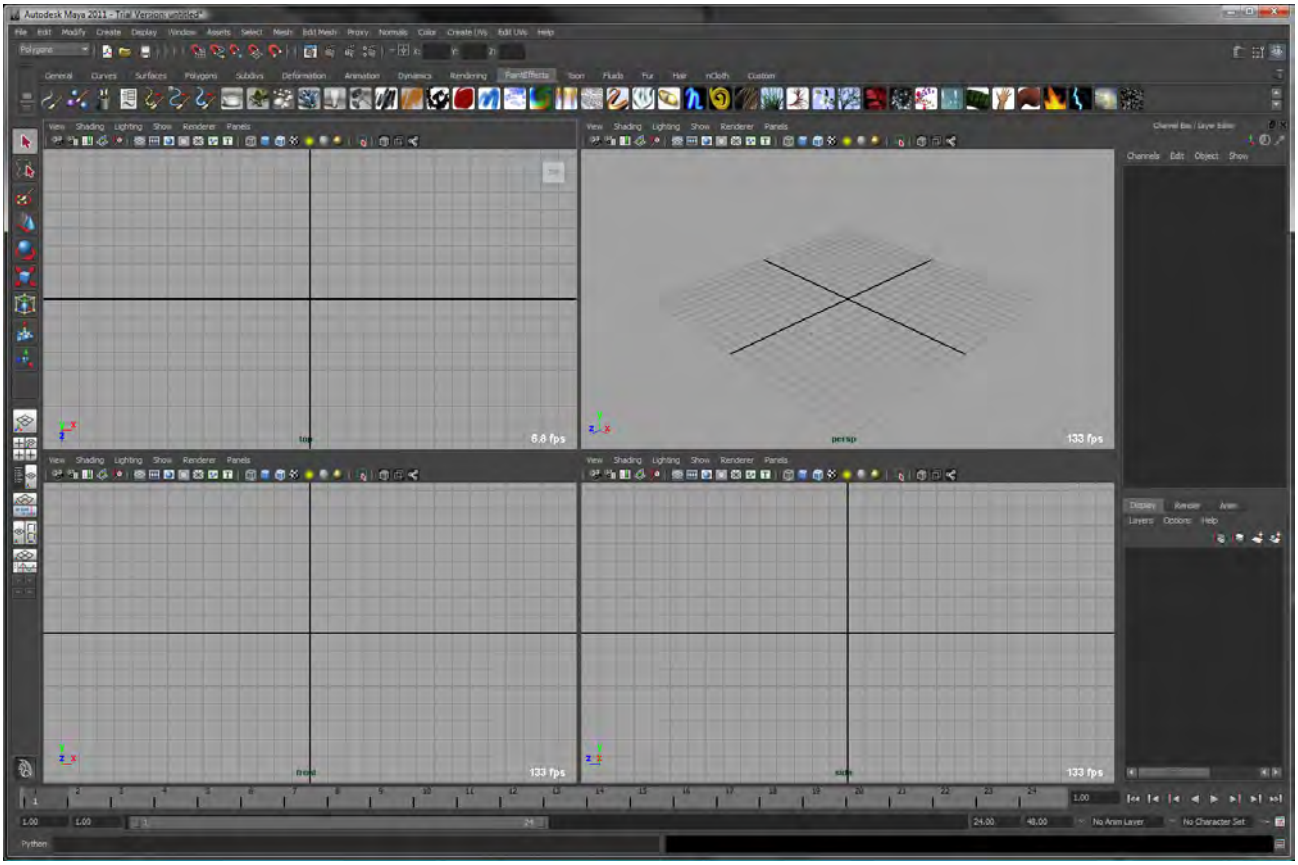


Figura 2.1 - Autodesk Maya 2011

2.1.2 Autodesk 3D Studio Max

3D Studio Max es una aplicación basada en Windows que permite crear tanto modelos como animaciones en tres dimensiones a partir de una serie de puertos de vistas.

Desarrollado por Autodesk Media & Entretainment (conocido también como Discreet y Kinetix) antes de la última adquisición de Autodesk, Discreet Logic. La utilización de 3D Studio Max permite al usuario la fácil visualización y representación de los modelos así como su exportación a otros formatos distintos del que utiliza el propio programa. Su arquitectura abierta y baja curva de aprendizaje lo convierte en el programa de preferencia de muchos usuarios en una infinidad de ámbitos: Arquitectura, Publicidad, Televisión y Video, Cine, Artes Escénicas, Desarrollo de juegos, Ingeniería, Desarrollo multimedia, aplicaciones científicas y generación de gráficos para internet.

3D Studio Max es uno de los programas de animación 3D más utilizados. Posee capacidades de edición muy completas y un sistema basado en plugins que permite el desarrollo de nuevas funciones apropiadas para cada usuario. Es utilizado bastamente por desarrolladores de juegos, películas y efectos especiales.

Ofrece también ciertas características interesantes, como un sistema de sombras avanzadas, simulaciones dinámicas y de partículas, radiosidad, creación de “Normal maps”, renderizado, iluminación global y un sistema intuitivo y personalizado de interfaz

de usuario, además del antes mencionado sistema de plugins que permite a terceros desarrollar software encaminado a satisfacer necesidades específicas.

Las versiones iniciales del programa requerían un dispositivo especial para la prevención de la piratería (llamado Dongle) el cual debía ser conectado al puerto paralelo mientras el programa se encontraba en ejecución. Pero actualmente se ha implementado un mejor control de copias y esta limitante desapareció.

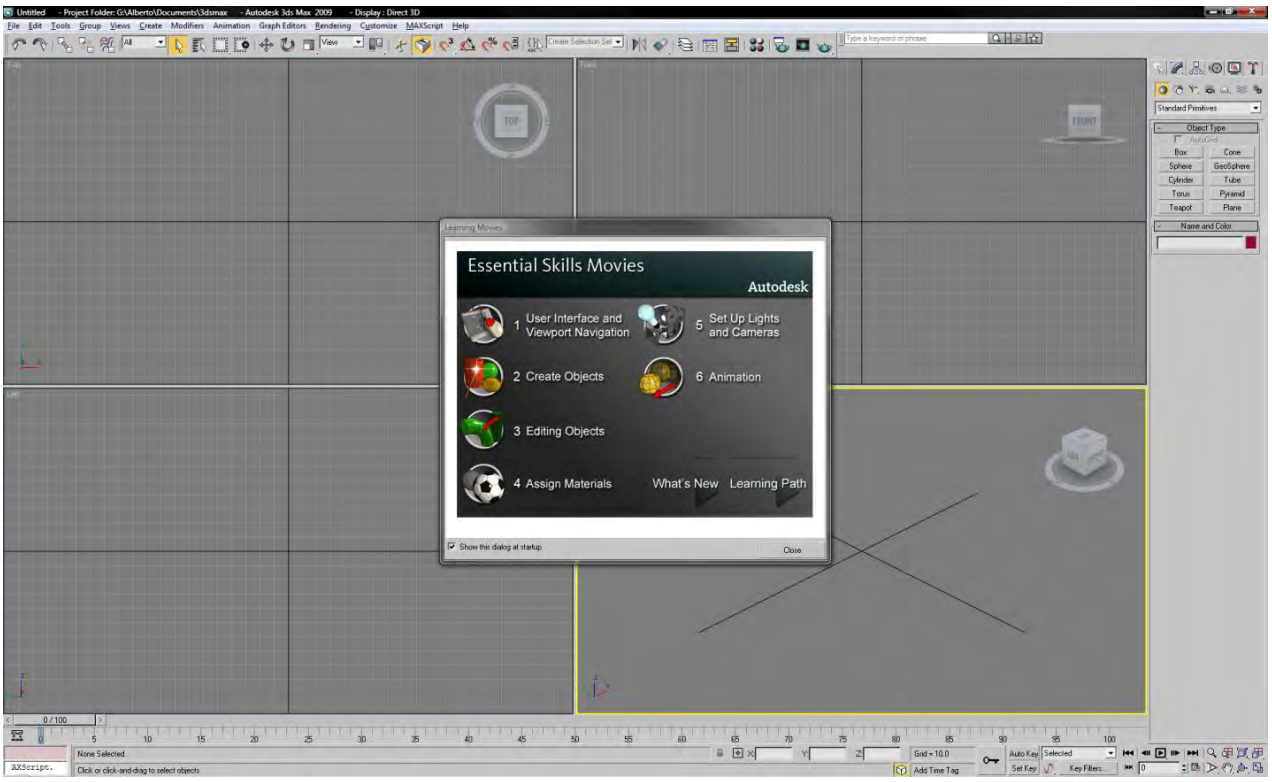


Figura 2.2 - Autodesk 3D Studio Max 2009

2.2 Programas de software libre

2.2.1 Blender

Fue desarrollado como una aplicación propia por el estudio de animación holandés NeoGeo; el principal autor, Ton Roosendaal, Fundó la empresa Nor a Number Technologies (NaN) en junio de 1998 para desarrollar y distribuir el programa, pero la compañía llegó a la bancarrota en 2002 y los acreedores acordaron ofrecer Blender como un producto de código abierto y gratuito bajo los términos de la licencia GNU GPL a cambio de 100 000 euros y en 2003 Roosendaal creó sin fines de lucro la fundación Blender, para recoger donaciones.

Es distribuido de forma gratuita y es compatible con todas las versiones de Microsoft Windows, Linux, Solaris, IRIX y MacOS X. Desde su liberación y gracias a continuas actualizaciones, así como su gran mejora del API de Python, el nuevo diseño de interfaz, soporte para el potente YafRay, actualización del motor de render propio y la amplia comunidad de desarrolladores y artistas gráficos ha hecho que se convierta en la herramienta gráfica libre por excelencia.

Pese a lo que se ha clasificado como una interfaz gráfica poco intuitiva, debido a que no se basa en el clásico sistema de ventanas, es también esta característica una de las ventajas importantes sobre el mismo sistema de ventanas pues la configuración personalizada de la distribución de menús y las vistas de cámara permite al usuario una mayor personalización de su ambiente de trabajo.

Entre las principales características con las que cuenta Blender se pueden encontrar las siguientes:

- Multiplataforma, libre y gratuita, con distribuciones generalmente pequeñas de 5 a 50 megas.
- Capacidad para una gran variedad de primitivas geométricas, incluyendo curvas, mallas poligonales, NURBS y metaballs.
- 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 videojuegos como detección de colisiones, recreaciones dinámicas y lógica.
- Posibilidades de renderizado interno versátil e integración externa con el potente trazador de rayos libre YafRay.
- Lenguaje Python para automatizar o controlar las tareas.
- Blender acepta formatos gráficos como TGA, JPG, Iris, SGI o Tiff. Así como archivos Inventor.
- Motor de juegos 3D integrado con un sistema de ladrillos lógicos.
- Simulaciones dinámicas para softbodies, partículas y fluidos.
- Modificaciones apilables para la aplicación de transformación no destructiva sobre mallas.
- Sistema de partículas estáticas para simular cabellos o pelajes.

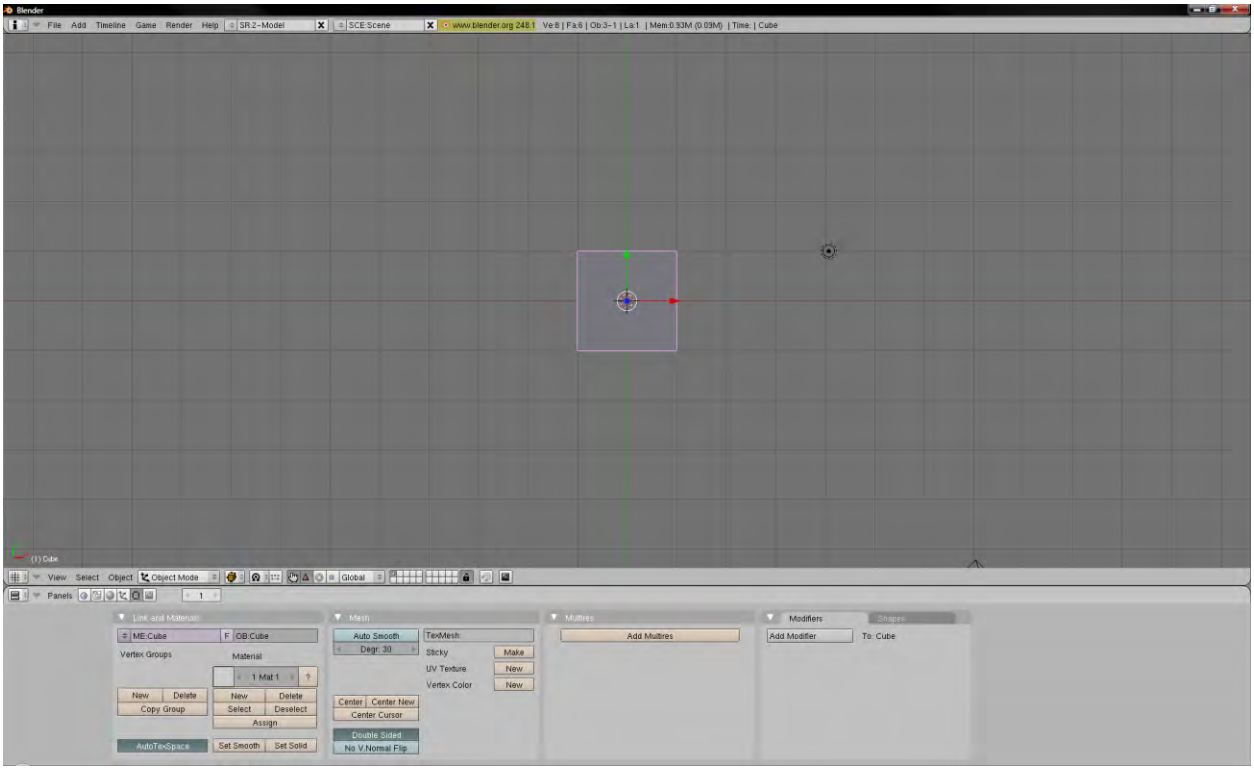


Figura 2.3 - Blender 2.48a

CAPÍTULO III. ANÁLISIS DE FORMATOS EXISTENTES

En la actualidad existe una gran variedad de formatos de modelos en 3D existentes, sin embargo no hay un formato que reúna las características del formato desarrollado en el presente tema. Si bien es cierto que existen formatos abiertos y fáciles de entender como el formato OBJ, este presenta limitaciones respecto a animaciones, necesitando de código adicional a un cargador de modelo para hacer muchas cosas que otros formatos pensados en videojuegos pueden hacer y es cierto que un formato específico para juegos de video (como el MD2 por ejemplo) es complicado de enseñar a un joven desarrollador. Un formato muy popular en los programas en la actualidad es el .3DS sin embargo la estructura de este formato está pensada para ser utilizada en el programa 3D Studio Max y no para que un programador pueda usarlo. Si bien es un formato que reúne muchas características que un desarrollador puede utilizar, también contiene más información de la que el programador necesitaría para desarrollar código, información relacionada a las cámaras, luces etc.

La principal de las ventajas propuestas en esta tesis es que el programador inexperto puede entender el formato con suma facilidad, esto se describirá en la sección correspondiente del presente trabajo. Y es para entender estas ventajas que es necesario explicar las características principales de los formatos más utilizados al momento de desarrollar código.

3.1 Formato OBJ

Este formato fue desarrollado por Wavefront (mejor conocido por sus herramientas gráficas que incluían animación, modelado, y composición de imágenes), mucho antes de su fusión con Alias y era utilizado en la aplicación "Wavefront´s Advanced Visualizer" para almacenar datos de objetos geométricos compuestos de líneas, polígonos y curvas de formas libres y superficies. Así mismo el formato soporta curvas racionales e irracionales, incluyendo las basadas en Bezier, B-spline y ecuaciones de Taylor. Pero por lo general se utiliza únicamente la información poligonal en aplicaciones prácticas.

El contenido del archivo esta en formato ASCII, por lo que pueden ser leídas fácilmente y la última versión del formato es la v3.0 con la extensión .obj. También existe la versión binaria del archivo, con terminación .MOD pero su distribución es mínima comparada con la versión ASCII. Además de que la dificultad humana de entender con facilidad líneas binarias la hace poco práctica de entender.

Los archivos OBJ no requieren ninguna clase de cabecera, aunque generalmente comienzan con una línea de comentario del algún tipo, mismas comienzan con el carácter '#', así mismo espacios y líneas en blanco pueden ser agregadas al archivo para facilitar la lectura. Cada línea con información útil comienza con una palabra clave, seguida de información relacionada a dicha palabra, y si es necesario las líneas pueden ser continuadas mediante el carácter lógico de continuación "\", de tal suerte que el archivo puede ser procesado de una sola pasada de principio a fin del archivo,

pero por lo general los cargadores lo hacen en dos pasadas para saber la información exacta a cargar y reservar espacio en memoria antes de cargar línea a línea.

Las siguientes son palabras clave del programa, pueden ser o no incluidas en el archivo OBJ y están acomodadas por tipos de datos:

Datos de Vértices:

"v" Vértices geométricos
"vt" Vértices de texturas
"vn" Vértices normales
"vp" Vértices de espacio paramétrico

Curvas y superficies de forma libre:

"deg" Grados
"bmat" Matriz base
"step" Tamaño del avance
"csType" Tipo de curva o superficie

Elementos:

"p" Punto
"l" Linea
"f" Cara
"curv" Curva
"curv2" Curva en 2D
"surf" Superficie

Declaraciones de cuerpo de Curvas y Superficies de forma libre:

"parm" Valores de parámetro
"trim" Outliner trimming loop
"hole" Inliner trimming loop
"scrv" Curva especial
"sp" Punto especial
"end" Fin de la declaración

Conectividad entre superficies de forma libre:

"con" Conector

Agrupación:

"g" nombre del grupo
"s" grupo de suavizado
"mg" grupo de mezcla
"o" nombre del objeto

Display/Render:

"bevel" Interpolación de bevel
"c_interp" interpolación de color
"d_interp" interpolación de disolución
"Lod" nivel de detalle
"usemtl" nombre del material
"mtllib" librería de materiales
"shador_obj" shadow casting
"trace_obj" Ray tracing
"ctech" Técnica de aproximación a la curva
"stech" Técnica de aproximación a la superficie

3.1.1 Mesh

Los archivos OBJ más comúnmente encontrados solamente contienen información de caras poligonales. Para describir un polígono el archivo primero declara los puntos necesarios para formar una cara con la palabra clave “v” y después describe la cara con la palabra clave “f”. La línea de descripción de la cara contiene la enumeración de los puntos que conforman la cara, en el orden en el que se encontraron en el archivo. El siguiente es un ejemplo de la descripción mínima en un archivo OBJ para describir un triángulo:

Ejemplo 3.1

```
# Simple Wavefront file
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 0.0 0.0
f 1 2 3
```

Describiendo línea a línea el archivo indica en la línea 1 un comentario, que como ya se mencionó no es necesaria pero es común encontrarla, la línea 2 a 4 describen las tres coordenadas de cada vértice del modelo, la línea 5 describe la manera en la que se creara la cara, es decir, hacer un triángulo utilizando los vértices 1, 2 y 3 en ese orden.

También es posible referenciar puntos utilizando índices negativos, esto hace más fácil incluir información adicional al final de archivo, simplemente copiando y pegando los contenidos de un archivo al fin de otro, aunque en la práctica este es el menos común de los archivos OBJ encontrados. El siguiente es el mismo archivo OBJ utilizando índices negativos:

Ejemplo 3.2

```
# Simple Wavefront file
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 0.0 0.0
f -3 -2 -1
```

3.1.2 Materiales

Los archivos OBJ no contienen información de colores para caras, sin embargo pueden referenciar materiales que son guardados en un archivo separado, conocido como librería de materiales. Esta librería puede ser cargada en memoria utilizando la palabra clave "*mtllib*". La librería de materiales contiene definiciones para los valores RGB (Rojo, Verde y Azul) para los colores ambiental, difuso, especular y otros tipos de características como transparencias o nivel de brillo.

Todas las caras posteriormente definidas a la palabra clave "usemtl" utilizaran el material definido en la palabra clave.

Las caras y superficies pueden ser agrupadas mediante la palabra clave "g" que es usada para crear lo que podríamos definir como grupos o "sub-objetos" para hacer más fácil la edición de modelos tridimensionales. Las caras pueden pertenecer a más de un grupo pero esto puede presentar complicaciones para muchos programas, por lo que por lo general solo pertenecen a uno solo. El siguiente es un ejemplo más complicado de un archivo que utiliza los comandos descritos.

Ejemplo 3.3

Cubo con materiales----------

```
# This cube has a different material
# applied to each of its faces.
mtllib master.mtl
v    0.000000  2.000000  2.000000
v    0.000000  0.000000  2.000000
v    2.000000  0.000000  2.000000
v    2.000000  2.000000  2.000000
v    0.000000  2.000000  0.000000
v    0.000000  0.000000  0.000000
v    2.000000  0.000000  0.000000
v    2.000000  2.000000  0.000000
# 8 vertices
g front
usemtl red
f 1 2 3 4
g back
usemtl blue
f 8 7 6 5
g right
usemtl green
f 4 3 7 8
g top
usemtl gold
f 5 1 4 8
g left
usemtl orange
f 5 6 2 1
g bottom
usemtl purple
f 2 6 7 3
# 6 elements
```

Los archivos OBJ pueden contener información adicional relevante a las normales utilizadas para cada vértice de cada cara y la manera en la que una textura puede ser aplicada a ella. Lo que requiere que la palabra clave “f” pueda tener diferentes entradas posterior a ella, ya hemos descrito las caras que no contienen información de vértices de texturas o de normales, pero si quisiéramos incluirlos en la construcción de la cara, posterior a definirlos, deberíamos formarlos especificando si se incluyen o no. La

siguiente es una línea de construcción de cara que incluye vértices de textura pero no de normales:

Ejemplo 3.4

```
f 0/0 1/1 2/2
```

Mientras que la siguiente es una línea de construcción de cara que incluye vértices de normales pero no de textura:

Ejemplo 3.5

```
f 0//0 1//1 2//2
```

Y finalmente la siguiente es una línea que contiene información de ambos:

Ejemplo 3.6

```
f 0/0/0 1/1/1 2/2/2
```

El siguiente archivo es la biblioteca de materiales y un archivo OBJ que contiene un cubo con un solo material y una sola textura aplicada a cada una de sus caras:

Ejemplo 3.7

MiCubo.mtl----------

```
newmtl blinn1SG
illum 4
Kd 0.00 0.00 0.00
Ka 0.00 0.00 0.00
Tf 1.00 1.00 1.00
map_Kd LABS.TGA
Ni 1.00
Ks 0.50 0.50 0.50
```

MiCubo.obj----------

```
# This file uses centimeters as units for non-parametric coordinates.
mtllib miCubo.mtl
g default
v -0.500000 -0.500000 0.500000
```

```
v 0.500000 -0.500000 0.500000
v -0.500000 0.500000 0.500000
v 0.500000 0.500000 0.500000
v -0.500000 0.500000 -0.500000
v 0.500000 0.500000 -0.500000
v -0.500000 -0.500000 -0.500000
v 0.500000 -0.500000 -0.500000
vt 0.000000 0.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 1.000000 0.000000
vt 1.000000 1.000000
vt -0.000000 1.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 1.000000 1.000000
vt 0.000000 1.000000
vt 1.000000 1.000000
vt 0.000000 1.000000
vt 1.000000 1.000000
vt 0.000000 1.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 1.000000 1.000000
vt 0.000000 1.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vt 1.000000 1.000000
vt 0.000000 1.000000
vt 0.000000 0.000000
vt 1.000000 0.000000
vn 0.000000 0.000000 1.000000
vn 0.000000 0.000000 1.000000
vn 0.000000 0.000000 1.000000
vn 0.000000 0.000000 1.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn 0.000000 0.000000 -1.000000
vn 0.000000 0.000000 -1.000000
vn 0.000000 0.000000 -1.000000
vn 0.000000 0.000000 -1.000000
vn 0.000000 -1.000000 0.000000
vn 0.000000 -1.000000 0.000000
vn 0.000000 -1.000000 0.000000
vn 0.000000 -1.000000 0.000000
vn 1.000000 0.000000 0.000000
vn 1.000000 0.000000 0.000000
vn 1.000000 0.000000 0.000000
vn 1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000
g MiCuboG1
usemtl blinn1SG
```

```
f 1/19/1 2/20/2 3/22/3
f 3/22/3 2/20/2 4/21/4
f 3/15/5 4/16/6 5/18/7
f 5/18/7 4/16/6 6/17/8
f 5/5/9 6/6/10 7/24/11
f 7/24/11 6/6/10 8/23/12
f 7/7/13 8/8/14 1/10/15
f 1/10/15 8/8/14 2/9/16
f 2/2/17 8/4/18 4/12/19
f 4/12/19 8/4/18 6/11/20
f 7/1/21 1/3/22 5/14/23
f 5/14/23 1/3/22 3/13/24
----------------------------------------------------------------------------------------------------
```

Como podemos apreciar el leer los archivos .mtl y .obj es muy sencillo una vez que se conocen las palabras clave, una persona a simple vista podría incluso dibujar esta representación geométrica en papel.

3.2 Formato 3DS

El formato 3DS (.3ds) es un formato cerrado, propiedad de Autodesk, esto significa que se puede hacer uso del archivo por cuenta propia, pero ni Autodesk ni alguna empresa subsidiaria de Autodesk puede brindar soporte técnico en el uso de este archivo, mismo que está escrito en binario, por lo que no puede ser leído a simple vista y por lo mismo no contiene comentarios o formato que simplifiquen su lectura.

El archivo 3ds, está formado en base a “chunks”, estos son paquetes de información. Cada chunk contiene un identificador, tamaño y datos. Se presentan de forma continua a lo largo del archivo, cada chunk después del otro. Esto es de suma utilidad, pues si no sabemos qué es lo que hace cada chunk (y aun actualmente no se ha podido descifrar que es lo que hace cada uno de los chunks) podemos brincar la cantidad de datos que el tamaño indica. El tamaño del chunk indica directamente la cantidad de bits

que deben ser leídos en este chunk o visto de otro modo en cuantos bits comienza el siguiente.

La información binaria en el archivo .3ds está escrita de una manera especial, el byte menos significativo se encuentra al principio en un entero. Un chunk puede definirse como:

inicio	fin	Tamaño	Nombre
0	1	2	ID del chunk
2	5	4	Apuntador al siguiente chunk, relativo al lugar donde el chunk se encuentra (o la longitud del chunk)

Tabla 3.1

Los chunks tienen impuesta una jerarquía, definida por su identificador. Los archivos 3ds, contienen un chunk primario con identificador 4D4Dh, y este es siempre el primer chunk del archivo.

Para darnos una referencia a la jerarquía de los chunks, a continuación se presenta un diagrama para mostrar los diferentes identificadores de chunks y su lugar en el archivo. Se les da un nombre para identificar su función dentro del archivo.

Tabla 3.2

```
MAIN3DS  (0x4D4D)
|
+--EDIT3DS  (0x3D3D)
| |
| +--EDIT_MATERIAL (0xAFFF)
| | |
| |  +--MAT_NAME01 (0xA000) (See mli Doc)
| |
| +--EDIT_CONFIG1  (0x0100)
| +--EDIT_CONFIG2  (0x3E3D)
| +--EDIT_VIEW_P1  (0x7012)
```

```
| | |
| | +--TOP          (0x0001)
| | +--BOTTOM       (0x0002)
| | +--LEFT         (0x0003)
| | +--RIGHT        (0x0004)
| | +--FRONT        (0x0005)
| | +--BACK         (0x0006)
| | +--USER         (0x0007)
| | +--CAMERA       (0xFFFF)
| | +--LIGHT        (0x0009)
| | +--DISABLED     (0x0010)
| | +--BOGUS        (0x0011)
| |
| +--EDIT_VIEW_P2  (0x7011)
| | |
| | +--TOP          (0x0001)
| | +--BOTTOM       (0x0002)
| | +--LEFT         (0x0003)
| | +--RIGHT        (0x0004)
| | +--FRONT        (0x0005)
| | +--BACK         (0x0006)
| | +--USER         (0x0007)
| | +--CAMERA       (0xFFFF)
| | +--LIGHT        (0x0009)
| | +--DISABLED     (0x0010)
| | +--BOGUS        (0x0011)
| |
| +--EDIT_VIEW_P3  (0x7020)
| +--EDIT_VIEW1    (0x7001)
| +--EDIT_BACKGR   (0x1200)
| +--EDIT_AMBIENT  (0x2100)
| +--EDIT_OBJECT   (0x4000)
| | |
| | +--OBJ_TRIMESH   (0x4100)
| | | |
| | | +--TRI_VERTEXL         (0x4110)
| | | +--TRI_VERTEXOPTIONS    (0x4111)
| | | +--TRI_MAPPINGCOORS     (0x4140)
| | | +--TRI_MAPPINGSTANDARD  (0x4170)
| | | +--TRI_FACEL1          (0x4120)
| | | | |
| | | | +--TRI_SMOOTH          (0x4150)
| | | | +--TRI_MATERIAL        (0x4130)
| | | |
| | | +--TRI_LOCAL           (0x4160)
| | | +--TRI_VISIBLE         (0x4165)
| | |
| | +--OBJ_LIGHT    (0x4600)
| | | |
| | | +--LIT_OFF             (0x4620)
| | | +--LIT_SPOT            (0x4610)
| | | +--LIT_UNKNWN01        (0x465A)
| | |
| | +--OBJ_CAMERA   (0x4700)
| | | |
| | | +--CAM_UNKNWN01        (0x4710)
| | | +--CAM_UNKNWN02        (0x4720)
| | |
| | +--OBJ_UNKNWN01 (0x4710)
| | +--OBJ_UNKNWN02 (0x4720)
| |
| +--EDIT_UNKNW01  (0x1100)
```

```
        | +--EDIT_UNKNW02  (0x1201)
        | +--EDIT_UNKNW03  (0x1300)
        | +--EDIT_UNKNW04  (0x1400)
        | +--EDIT_UNKNW05  (0x1420)
        | +--EDIT_UNKNW06  (0x1450)
        | +--EDIT_UNKNW07  (0x1500)
        | +--EDIT_UNKNW08  (0x2200)
        | +--EDIT_UNKNW09  (0x2201)
        | +--EDIT_UNKNW10  (0x2210)
        | +--EDIT_UNKNW11  (0x2300)
        | +--EDIT_UNKNW12  (0x2302)
        | +--EDIT_UNKNW13  (0x2000)
        | +--EDIT_UNKNW14  (0xAFFF)
        |
        +--KEYF3DS (0xB000)
          |
          +--KEYF_UNKNWN01 (0xB00A)
          +--..... (0x7001) ( viewport, same as editor )
          +--KEYF_FRAMES   (0xB008)
          +--KEYF_UNKNWN02 (0xB009)
          +--KEYF_OBJDES   (0xB002)
            |
            +--KEYF_OBJHIERARCH  (0xB010)
            +--KEYF_OBJDUMMYNAME (0xB011)
            +--KEYF_OBJUNKNWN01  (0xB013)
            +--KEYF_OBJUNKNWN02  (0xB014)
            +--KEYF_OBJUNKNWN03  (0xB015)
            +--KEYF_OBJPIVOT     (0xB020)
            +--KEYF_OBJUNKNWN04  (0xB021)
            +--KEYF_OBJUNKNWN05  (0xB022)
----------------------
```

3.2.1 Información más detallada del contenido de un archivo 3DS

Main Chunk

El chunk principal de archivo tiene el identificador 4D4Dh es realmente el archivo completo, así que el tamaño de este chunk es el tamaño del archivo menos su cabecera. Contenidos en el tenemos dos chunks importantes, el chunk de 3D (3D3Dh) y el chunk de Keyframes (B000).

Directamente después del chunk principal de 3D se encuentra otro chunk, este puede ser de cualquier tipo permitido dentro del scope del chunk principal (observar la tabla 3.2).

El subChunk de 3D3D

ID	Descripción
0100	Parte de la configuración
1100	(Desconocido)
1200	(Color de fondo)
1201	(Desconocido)
1300	(Desconocido)
1400	(Desconocido)
1420	(Desconocido)
1450	(Desconocido)
1500	(Desconocido)
2100	Bloque de color ambiental
2200	(Desconocido, Puede ser niebla)
2201	(Desconocido, Puede ser niebla)
2210	(Desconocido, Puede ser niebla)
2300	(Desconocido)
3000	(Desconocido)
3D3E	Editor de configuración del bloque principal
4000	Definición de un objeto
AFFF	Comienzo de una lista de materiales

El subChunk AFFF comienza una lista de materiales

A000 – Nombre del material

El subChunk 3D3E – comienza el editor de configuración

7001	Indicador del puerto de vista
7011	Definición del puerto de vista (tipo 2)
7012	Definición del puerto de vista (tipo 1)
7020	Definición del puerto de vista (tipo 3)

Este chunk particularmente contiene lo que parece ser información redundante, su chunk más importante es el 7020, pues él describe los 4 puertos de vistas que están activos en el editor

El subChunk 4000 es el bloque de descripción de objeto, el primer subChunk contenido es el nombre del objeto, contenido en una cadena ASCIIZ, las cadenas ASCIIZ son cadenas de caracteres que terminan en cero.

Un objeto puede ser una cámara, una luz o un mesh

ID	Descripción
4010	(Desconocido)
4012	(Desconocido, puede ser sombra)
4100	Lista de polígonos triangulares
4600	Luz
4700	Cámara

Mapeo, estos chunks son opcionales, pero siempre se encuentran después de una lista de vértices.

SubChunks del subChunk 4100 – Lista de polígonos triangulares

ID	Descripción
4110	Lista de vértices
4111	Opciones de vértices
4120	Lista de caras
4130	Material de la cara
4140	Coordenadas de mapeo
4150	Grupo Smoothing de la cara
4160	Matriz de transición
4165	Visibilidad del objeto

4170 Mapeo estándar

4110 – lista de vértices

Inicio	fin	tamaño	tipo	Nombre
0	1	2	unsigned int	Numero de vértices en el objeto
2	5	4	Float	Valor X
6	9	4	Float	Valor Y
10	13	4	Float	Valor Z

Tabla 3.3

Posterior, los bytes 2 a 13 se repiten conforme al número de vértices en el objeto

El subChunk 4111 no se describirá en el presente documento, pues contiene información referente al manejo de la escena para el programa 3D Studio Max, y en realidad no tiene influencia sobre el cargado del archivo en un programa propio (condicionado a que no se esté programando un manejador de escenas como 3D Studio Max) y en código puede ser completamente ignorado.

El subChunk 4120 contiene la lista de caras del objeto.

Inicio	fin	tamaño	tipo	Nombre
0	1	2	Unsigned int	Total de

				polígonos
2	3	2	Unsigned int	Vértice A
4	5	2	Unsigned int	Vértice B
6	7	2	Unsigned int	Vértice C
8	9	2	Unsigned int	Información de la cara

Tabla 3.4

Los primeros 3 enteros son los tres vértices de la cara (el formato 3ds trabaja con triángulos y no con polígonos).

Ahora bien, el orden de los índices de los vértices cumple el propósito de indicar la normal de la cara. Cumpliendo la regla de la mano derecha. Si los vértices son dados en el orden A B C:

Ejemplo 3.8

```
      C
      ^
      |
A-----B
```

Significara que la normal apunta hacia el lector.

El subChunk 4130 es el indicador del chunk de la cara. Si el objeto tiene unicamente el material por default, no se cuenta con el chunk 4130, de hecho hay un chunk 4130 para cada material presente en el objeto. Cada chunk 4130 comineza con una cadena ASCIIZ referenciando el nombre del material, posterior al cero de la cadena encontramos un entero que dice el número de caras en el objeto a las cuales se le aplica el material, posterior al numero, el identificador de cada cara a la que se aplica.

El subChunk 4140 contiene las coordenadas de mapeo, esto es aplicado para colocar texturas en el objeto. Los primeros dos bytes del chunk son el número de vértices y posterior a ellos por cada dos flotantes es una coordenada.

El subChunk 4160 es el sistema de coordenadas local, los primeros tres bloques de los flotantes, son la definicion de los tres vectores del sistema X Y Z del objeto.

Los SubChunk 4170, 4600, 4610 y 4700 se encuentran descritos en el archivo 3D-Studio File Format de Martin van Velsen y al no ser de relevancia para el cargado de un modelo, no seran descritos en el presente trabajo.

Chunks del manejo de Keyframes

ID	Descripción
B00A	(Desconocido)
7001	(Primera descripcion del chunk)
B008	Frames
B009	(Desconocido)
B002	Inicio de descripcion del objeto

B008 - Descripción de los frames. Este subchunk nos informa cual es el frame inicial y cual es el frame final.

B002 - Inicio del objeto

ID	Descripción.
B010	Nombre y Jerarquia
B011	Objeto Dummy (tambien conocido como grupo vacio)

B013 (Desconocido)
B014 (Desconocido)
B015 (Desconocido)
B020 Pivote
B021 (Desconocido)
B022 (Desconocido)

El subChunk B010 es el descriptor de el nombre y la jerarquia del objeto

Inicio	Fin	Tamaño	Tipo	Nombre
0	¿?	¿?	ASCIIZ	Nombre del objeto
¿?	¿?	2	Unsigned int	Desconocido
¿?	¿?	2	Unsigned int	Desconocido
¿?	¿?	2	Unsigned int	Jerarquia del objeto

Tabla 3.5

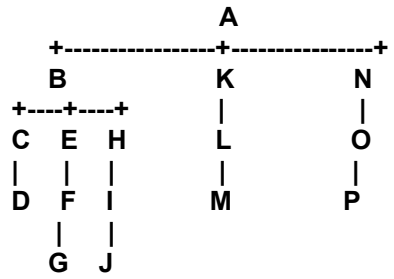
La jerarquia del objeto es un poco complicada, pero su funcionamiento es aproximadamente este, cada objeto en la escena recibe un numero identificando su orden en el árbol de jerarquias, ademas cada objeto esta ordenado en el archivo 3DS de manera como apareceria en el árbol.

La raíz es identificada con el numero -1 (FFFF). Conforme el archivo se esta leyendo se maneja un contador, este incrementa y representa que el objeto es hijo del objeto anterior, pero cuando el parentesco se rompe, el contador regresa al nivel corresponidente.

Ejemplo 3.9

```
object   hierarchy
name

   A     -1
   B      0          This example is taken
   C      1          from 50pman.3ds
   D      2
   E      1          I would really reccomend
   F      4          having a look at one of the
   G      5           examples with the hierarchy
   H      1           numbers to help work it out.
   I      7
   J      8
   K      0
   L     10
   M     11
   N      0
   O     13
   P     14
```



Tomado del archivo 3D-Studio File Format de Martin van Velsen.

El subChunk B020, es el punto pivote del objeto, se desconocen actualmente el significado de los primeros 5 flotantes del chunk, pero posterior a ellos se encuentran los componentes del punto X Y Z.

Inicio	Fin	Tamaño	Tipo	Nombre
0	3	4	Float	Desconocido
4	7	4	Float	Desconocido
8	11	4	Float	Desconocido

12	16	4	Float	Desconocido
16	19	4	Float	Desconocido
20	23	4	Float	Pivote Z
24	27	4	Float	Pivote Y
28	32	4	Float	Pivote X

Tabla 3.6

Como podemos apreciar la lectura de los archivos 3DS no es trivial, y de hecho complica mucho la carga de archivos para el programador inexperto, si se cuenta con un cargador de modelos 3DS el manejo de los objetos se hace de forma sencilla, sin embargo el editar la información contenida del modelo se complica. Asi mismo al presentarse de forma binaria se complica la lectura a ojo de pájaro del archivo, complicando aun mas la enseñanza del manejo del modelo.

Por otro lado provee muchas ventajas sobre el archivo OBJ, como por ejemplo que el tamaño del archivo tiende a ser mucho mas pequeño o que el manejo de animaciones y materiales se puede hacer de manera nativa en el formato sin requerir de archivos adicionales.

CAPÍTULO IV. EL FORMATO PROPIETARIO BTO

En este capítulo se tratará la descripción del formato BTO y de las consideraciones que fueron tomadas en cuenta para su elaboración, el porqué de la estructura que tiene y las ventajas que ofrece sobre otros formatos analizados.

4.1 Descripción

El formato BTO fue pensado desde un principio en las aplicaciones prácticas que un formato de modelos tridimensionales debería de ofrecer, y posteriormente se arregló su estructura en respuesta a la necesidad de adaptarse a un formato de enseñanza práctica. Ofrece claras ventajas sobre formatos como OBJ o 3DS para el manejo de modelos tridimensionales en proyectos no exclusivos del área gráfica, como el manejo de estados y animación tradicional por Keyframes.

El formato BTO basa sus archivos (con terminación .bto) en formato ASCII para su fácil lectura, y está dividido en bloques para su fácil manejo. Permite comentarios dentro del archivo, espacios, tabulaciones y líneas en blanco para facilitar aún más la lectura al usuario. Todos los comentarios deben de comenzar con el carácter “#” o los caracteres “//”. Los espacios y las tabulaciones pueden hacerse entre datos en la misma línea, pero está a consideración de la programación del cargador de archivos el interpretarlos, por lo mismo se recomienda que ellos solo estén contenidos después de los datos, al inicial un comentario o para espaciar.

Todos los archivos BTO comienzan con un identificador de archivo y versión. Así los archivos BTO de la versión más reciente (2.02) comienzan con el identificador ABTO0202, este identificador representa en sus primeros 4 caracteres que se trata de un archivo BTO (ABTO) y a continuación la versión en forma entera (0202).

Posterior a esto se encuentra el bloque de información de contenido, este bloque indica la cantidad de vértices, vértices de texturas, normales, polígonos, materiales, grupos, frames, secuencias, estados y formación del modelo. El orden en el que se agregue esta información, no es importante, siempre y cuando toda sea contenida en él. La utilidad de este bloque consiste en que se puede determinar el número de líneas a leer en los bloques correspondientes y por lo tanto leer el archivo en una sola pasada. Dentro de este mismo bloque la formación del modelo indica el tipo de polígonos contenidos en el modelo, si todos los polígonos del modelo son de tipo 0 el tipo de caras será 0 y esto es igualmente válido para los tipos 1, 2 y 3, estos tipos se describen más adelante, si el modelo está compuesto por diferentes tipos de caras, el tipo de caras será 4, además se indica si el modelo estará o no hecho a base de únicamente triángulos (triangulado) o si los vértices son compartidos por los polígonos (resumido), por último se indica la cantidad de vida que hay en el modelo.

Ejemplo 4.1

Vertices: 8
Vertices de Texturas: 24
Normales: 24
Poligonos: 12
Materiales: 2
Grupos: 1
Frames: 1
Secuencias: 1
Estados: 1
Tipo de Caras: 4
Triangulado: 1

Resumido: 0
Vida: 1

A continuación encontramos el bloque Vértices, este bloque enumera las coordenadas de todos los vértices que componen el polígono, siguiendo un orden secuencial. Este orden es importante, pues de él depende la formación correcta de los polígonos del objeto, siendo el primer vértice declarado el vértice 0 y continuando en orden ascendente directo.

Ejemplo 4.2

-0.500000 -0.500000 0.500000
0.500000 -0.500000 0.500000

En seguida encontramos el bloque Vértices De Textura, este bloque indicara las coordenadas en la que la textura tendrá que ser aplicada al polígono, nuevamente cumplen un orden exacto comenzando con el vértice de textura 0.

Ejemplo 4.3

0.000000 0.000000
1.000000 0.000000
1.000000 0.000000

Posteriormente encontramos el bloque de normales, en él se incluyen todos los vectores normales encontrados en el modelo, el primer vector encontrado es el vector normal 0.

Ejemplo 4.4

0.000000 1.000000 0.000000
0.000000 0.000000 -1.000000

Sigue el bloque de materiales, los archivos BTO contienen en sí mismos, a diferencia de los archivos OBJ, información de materiales en el modelo. Cada línea importante es identificada mediante palabras clave y leída correspondientemente, el orden no es importante a excepción del identificador “Material” que indica un nuevo material en el modelo. Las líneas de materiales pueden tener los siguientes identificadores:

- “Material” indica un nuevo material en el archivo, la sentencia de la línea es:

  Material (Entero, Número de material) (Cadena, Nombre del material)

- “Textura” indica el nombre del archivo de texturas que utiliza el material, puede ser un archivo .tga, .bmp, .png, .gif, .jpg, etc. La sentencia es:

  Textura (Cadena, Nombre del archivo de material, si es NULL, no utiliza archivos de materiales)

- “Illum” indica el modelo de iluminación del material, actualmente no es aplicado, pero está abierto a ser expandido. El modelo de iluminación podrá ofrecer cambios en el material. La sentencia es:

  Illum (Entero, tipo de modelo)

- “Kd” indica el tipo de luz difusa que reflejará el material. La sentencias es:

  Kd (flotante) (flotante) (flotante) (flotante)

- “Ka” indica el tipo de luz ambiental que reflejará el material. La sentencias es:

  Ka (flotante) (flotante) (flotante) (flotante)

- “Ke” indica el tipo de luz de emisión que reflejará el material. La sentencias es:

  Kd (flotante) (flotante) (flotante) (flotante)

- “Ni” indica la cantidad de brillo que tendrá el material. La sentencia es:

  Ni (flotante)

- “Ks” indica el tipo de luz especular que reflejará el material. La sentencias es:

  Kd (flotante) (flotante) (flotante) (flotante)

- “Solido” indica la cantidad de transparencia que tendrá el material. Va de 0 (completamente transparente) a 1 (completamente sólido)

  Solido (flotante)

Ejemplo 4.5

Material 0 blinn1SG
Textura LABS.TGA
Illum 4
Kd 0.000000 0.000000 0.000000 1.000000
Ka 0.000000 0.000000 0.000000 1.000000
Ke 0.500000 0.500000 0.500000 1.000000
Ni 1.00
Ks 0.500000 0.500000 0.500000 1.000000
Solido 1.00

A continuación se encuentra el bloque de polígonos, en él se indica cómo utilizar toda la información anterior para construir los polígonos del modelo. Comienza con 3 líneas que describen el primer polígono (el polígono 0), estas pueden o no aparecer dependiendo de las condiciones dadas.

- La primera línea solo se encuentra en caso de que el tipo de caras del modelo sea 4 pues es necesario indicar que tipo de cara se estará leyendo y en ella se describe el tipo de polígono, los tipos de polígonos dentro de un modelo BTO pueden ser de 4 tipos, pues pueden o no incluir vértices de textura o normales:
    - Serán de tipo 0 si solo contienen información referente a los vértices.
    - Serán de tipo 1 si contienen información referente a vértices, vértices de textura y normales.
    - Serán de tipo 2 si contienen información referente a vértices y a vértices de textura pero no de normales.
    - Será de tipo 3 si contiene información referente a vértices y normales pero no de vértices de textura.
- La siguiente línea indica a cantidad de vértices, si el modelo no está triangulado se incluye una línea que indica la cantidad de vértices que hay en el polígono 0.

- La siguiente línea de descripción indica el material a utilizarse en modelo. Posterior a estas 3 líneas descriptivas se encuentra una sola línea donde se indican la formación del modelo, en el orden v/vt/n, después de esta línea vendrán líneas conforme al número de polígonos haya en el modelo, y se irán intercalando descriptores de caras, como con el polígono 0 según se vayan presentando cambios.

La sintaxis es la siguiente:

Tipo: (Entero, Tipo de polígono) (línea opcional)

Vértices: (Entero, cantidad de vértices) (línea opcional)

Material: (Entero, Identificador de material a aplicar en el polígono)

(Entero, número de vértices en el polígono) (Entero, tipo de polígono) (Entero, material)

(Entero, vértice) / (Entero, opcional, vértice de textura) / (Entero, opcional, normal)…

Ejemplo 4.6	
Tipo: 1	**//Tipo de polígono siguiente**
Vertices: 3	**//Numero de vértices en el polígono**
Material: 0 AmarilloSG	**//Material a aplicar**
0/0/0 1/1/1 2/2/2	**//3 componentes**
Tipo: 3	**//La siguiente cara es del tipo 3**
Vértices: 4	**//y tiene 4 vértices, mismo material**
2//2 1//1 3//3 4//4	**//4 componentes**

Enseguida nos encontramos con el bloque de grupo. Los grupos pueden ser considerados como submodelos, su practicidad consiste en que cada submodelo puede ser editado dentro del contendido del modelo global. Además se cuenta con Jerarquía de grupos, lo cual permite que el comportamiento de un grupo afecte a otro.

El bloque de grupos contiene más información que cualquier otro bloque del formato, pues en él se describe el nombre del grupo; los identificadores de los polígonos

correspondientes al grupo (los polígonos pueden estar en más de un grupo, pero solo pueden ser polígonos consecutivos); el padre jerárquico del grupo; si el grupo es colisionable o no; la cantidad de vida del grupo (el sistema de referencias de colisiones es un sistema basado en juegos de roll (RPG) se cuenta con un contador de vida para cada grupo (HP) que determina la cantidad de colisiones que puede soportar el grupo antes de marcarse como eliminado en una aplicación); el punto de pivote sobre el cual se escalará y rotará el grupo; el punto centroide que indica el centroide calculado a partir de los vértices del grupo; el radio, máximos y mínimos concernientes a las cajas de colisiones y la esfera de colisión por grupo; y la descripción de los Keyframes (fotogramas clave) que tendrá el grupo. Descripción de líneas:

- (Entero, número de grupo) (Cadena, nombre del grupo) (Entero -> Entero, polígono inicial y polígono final del grupo)
- Padre: (Cadena, Nombre del padre del grupo, NULL si no tiene padre)
- Colisionable: (Cadena, Si o No)
- HP: (Entero, cantidad de puntos de vida del modelo)
- Pivote: (flotante, flotante, flotante)
- Centroide: (flotante, flotante, flotante)
- Radio: (flotante)
- XMax: (flotante) Xmin: (flotante)
- YMax: (flotante) Ymin: (flotante)
- ZMax: (flotante) Zmin: (flotante)
- Keyframes: (Entero, número de fotogramas claves y líneas a continuación)
    - ((Entero, número de frame), (flotante, posición X), (flotante, posición Y), (flotante, posición Z), (flotante, ángulo X), (flotante, ángulo Y), (flotante,

ángulo Z), (flotante, tamaño X), (flotante, tamaño Y), (flotante, tamaño Z), (Entero, visibilidad 0 invisible 1 visible))

Ejemplo 4.7

---

0 pCube1 (0 -> 11)
Padre: NULL
Colisionable: Si
HP: 1
Pivote: (0.0000,0.0000,0.0000)
Centroide: (0.0010,0.0010,-0.0010)
Radio: 0.8660
XMax: 0.5010 Xmin: -0.4990
YMax: 0.5010 Ymin: -0.4990
ZMax: 0.4990 Zmin: -0.5010
Keyframes: 1 (0, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.00, 1.00, 1.00, 1)

A continuación se encuentra el grupo Secuencias, este grupo determina el número de animaciones, el inicio y fin de cada animación del modelo. La sintaxis de las líneas es: (Entero, número de secuencia) (Cadena, nombre de la secuencia) (Entero -> Entero, Frame inicial de la secuencia, Frame final de la secuencia).

Ejemplo:

0 Default (0->0)

Sigue el bloque de estados, el sistema de estados está basado en una máquina de estados simple. Para cada estado del modelo se especifica el número de estados a los que el estado actual tiene salidas y que secuencia o animación debe de ejecutarse al salir del estado actual para entrar al estado destino. La sintaxis de las líneas de estados es la siguiente:

- (Cadena, nombre del estado)
- Secuencia: (Entero, animación que se ejecutara mientras el estado este activo)
- Salidas: (Entero, número de estados a los que se tiene salidas)

- E: (Entero, estado al que se puede salir) S: (Entero, secuencia que se anima al salir)

Ejemplo 4.8

Default
Secuencia: 0
Salidas: 0

Por último encontramos el Bloque Campo de acción, este bloque especifica los límites del área en el que el modelo se puede mover en cada fotograma de animación, mediante este campo de acción se pueden calcular colisiones específicas, son el método de preferencia, el fin único de este campo de acción es indicar cuando un modelo se encuentra en el campo de acción de otro, con el fin de revisar colisiones y evitar calcularlas innecesariamente. Sintaxis:

- Centroide: ((Flotante, posición X), (Flotante, posición Y), (Flotante, posición Z)) Radio: (Flotante, radio)
- XMax: (flotante) Xmin: (flotante)
- YMax: (flotante) Ymin: (flotante)
- ZMax: (flotante) Zmin: (flotante)

Ejemplo 4.9

Centroide: (0.000000,0.000000,0.000000) Radio: 0.866025
XMax: 0.500000 Xmin: -0.500000
YMax: 0.500000 Ymin: -0.500000
ZMax: 0.500000 Zmin: -0.500000

La estructura de bloques del archivo BTO permite expandir según las necesidades del usuario la información que se considere pertinente, esta es también una de las grandes ventajas del formato, pues no limita su uso.

4.2 Consideraciones del diseño

Se tomaron diferentes consideraciones al momento de implementar el diseño, pero todas ellas fueron pensadas en la elaboración de un formato sencillo de entender al momento de ser leído y que a la vez ofreciera herramientas que permitieran el desarrollo de funciones prácticas para la carga de modelos. Puntos fuertes fueron tomados de formatos existentes pero aplicados y mejorados en un solo formato. Las características propias del formato nos llevan entonces a formular una serie de consideraciones de diseño que el formato está obligado a mantener.

4.3 Consideraciones básicas del formato

Es importante tomar en cuenta que el formato trabaja en un sistema secuencial de bloques y que el orden de los bloques es importante, pero no limita la forma en la que se lee el archivo, o al uso del formato en cualquier forma. Se propone el archivo BTO.dll para la carga y el manejo de los archivos BTO pero no se limita al uso de dichos archivos, esta DLL permite la carga y conversión de archivos .3DS y .OBJ, asi mismo se invita al programador experto a que una vez conocidas las estructuras de datos contenidos en el formato y tomando en cuenta las consideraciones básicas del formato desarrolle su propio cargador bajo las necesidades específicas que requiera. Las consideraciones básicas son las siguientes:

- Los modelos deben de tener al menos 3 vértices por polígono y no limita el número de vértices que puedan contener, para fines prácticos, la computadora internamente trabaja con triángulos, por lo que de cualquier forma el modelo será convertido a triángulos al momento de dibujarse en pantalla, sin embargo

tomará más tiempo de procesador hacerlo de esta manera que triangular el modelo desde un principio.

- Un modelo que comparte vértices, normales y polígonos por grupo, se limita de muchas maneras, el compartir vértices y normales hace muy difícil trabajar de manera matricial con el archivo, por lo que la implementación de huesos y esqueletos al modelo se elimina. El compartir polígonos en un grupo hace que la animación por Keyframes de grupos se comporte de maneras inesperadas, pues la animación de un polígono respecto a un grupo es afectada también por la del otro, creando deformidades en la maya de polígonos.
- Un modelo puede tener cualquier número de materiales, sin embargo si se planea dibujar el modelo por caras, es necesario que se tenga al menos un material.
- Un modelo puede tener cualquier número de fotogramas, sin embargo es necesario que se cuente con al menos el fotograma 0, pues se asume que la información de construcción de polígonos y grupos está asociada a él.
- Un modelo puede tener cualquier número de secuencias, pero se recomienda al programador que se cree al menos una secuencia para poder manejar la estructura de estados, que valla del cuadro inicial al cuadro final.
- Un modelo puede tener cualquier número de estados, pero se recomienda al programador que al menos se cuente con el estado 0, así se podrá trabajar con la estructura de estados en el manejo de más de un modelo.
- El campo de acción por sí mismo no calcula colisiones, sirve para identificar en qué momento un objeto está en el campo de acción de otro y de esta forma evitar el cálculo de grupo a grupo para todos los objetos de la escena. Si un

grupo está en el campo de acción de otro es porque hay proximidad entre ellos y se puede implementar el cálculo de colisiones que el programador requiera.

4.4 Ventajas sobre otros formatos

El formato BTO ofrece ventajas sobre otros formatos de modelos tridimensionales, como se explica a continuación.

4.4.1 Sobre OBJ

Una de las principales ventajas con las que cuenta el formato BTO sobre el OBJ es que reduce el número de archivos necesarios para lograr el mismo objetivo, un archivo BTO en sí mismo puede contener toda la información de un modelo OBJ con todo y su biblioteca de materiales, además ofrece, animación, secuencias, jerarquías y estados en un solo archivo.

4.4.2 Sobre 3DS

El formato BTO no contiene más información de la necesaria para el pintado del modelo. Además, su forma ASCII ayuda a cualquier persona a leer el archivo y lograr entenderlo a simple vista. Contiene además secuencias y estados que el formato 3DS en sí mismo no contiene, lo que ayuda enormemente a ser aplicado en programas específicos. Los modelos 3DS están siempre triangulados y pese a que esto es una gran ventaja en tiempo de procesador, limita la enseñanza de procesamiento de polígonos a estudiantes de computación gráfica, un modelo en formato BTO no tiene límite en la cantidad de vértices que un polígono puede tener y el código ejemplo, Visor BTO, ofrece una conversión de modelos cuadriculados a triangulares.

4.5 Comparativa de formatos

A continuación se presenta el mismo modelo tridimensional en formatos OBJ, 3DS y BTO, con esto se pretende poder realizar una comparativa en la facilidad de lectura que los formatos ofrecen. El modelo consiste en un cubo cuyas caras están formadas de 2 triángulos respectivamente y cada cara tiene un material diferente.

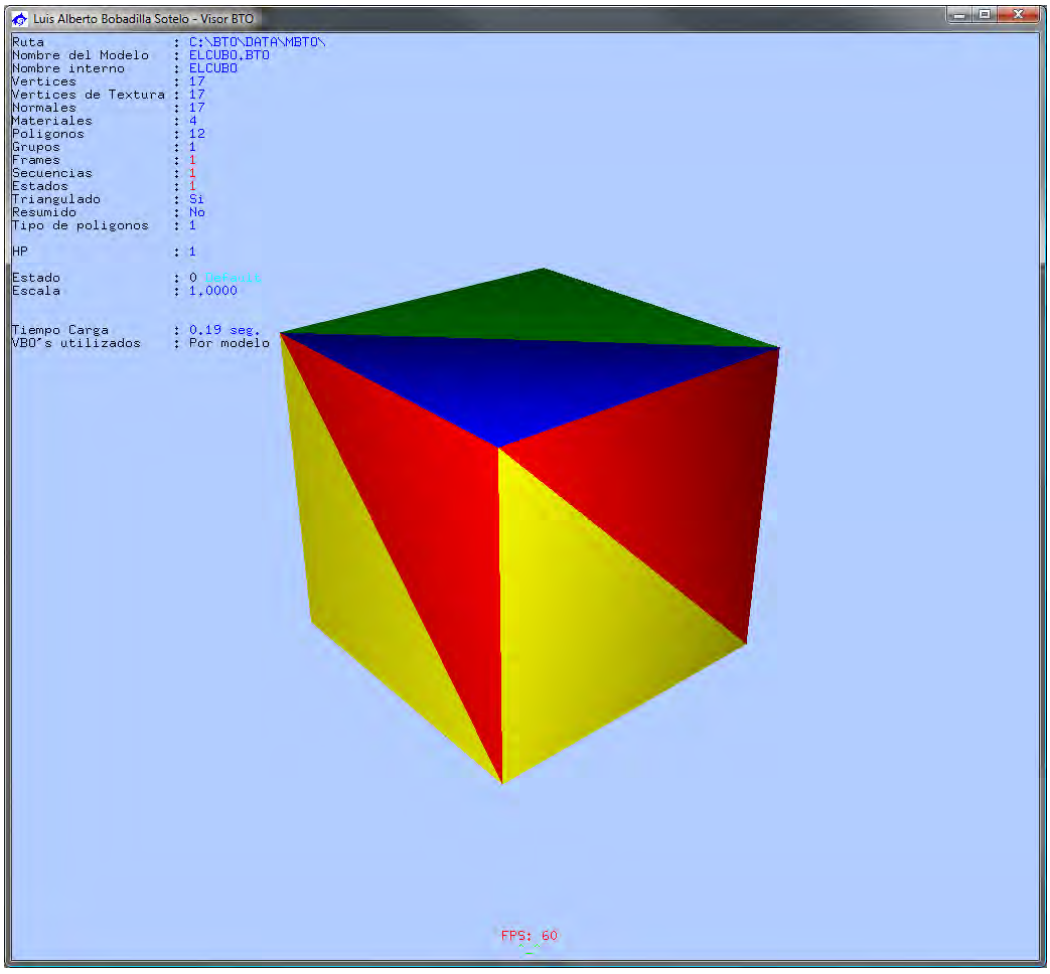


Figura 4.1 - Visor BTO

OBJ

Ejemplo 4.10

Archivo de materiales (.mtl) del modelo obj

```
# ------------------------------------------------------------------
#                  Archivo MTL del modelo El_cubo.obj
# ------------------------------------------------------------------
# Materiales: 4
#Material 0-------------------------------------------------------------
newmtl AmarilloSG
illum 4
Ka 0.50 0.50 0.00
Ks 0.33 0.33 0.00
Kd 1.00 1.00 0.00
Tf 0.80 0.80 1.00
Ni 1.00
#Material 1-------------------------------------------------------------
newmtl RojoSG
illum 4
Ka 0.50 0.00 0.00
Ks 0.33 0.00 0.00
Kd 1.00 0.00 0.00
Tf 0.80 1.00 1.00
Ni 1.00
#Material 2-------------------------------------------------------------
newmtl AzulSG
illum 4
Ka 0.00 0.00 0.50
Ks 0.00 0.00 0.33
Kd 0.00 0.00 1.00
Tf 1.00 1.00 0.80
Ni 1.00
#Material 3-------------------------------------------------------------
newmtl VerdeSG
illum 4
Ka 0.00 0.50 0.00
Ks 0.00 0.33 0.00
Kd 0.00 1.00 0.00
Tf 1.00 0.80 1.00
Ni 1.00
```

```
Archivo .obj
# Poligonos: 12

mtllib El_cubo.mtl
g default
v -0.576196 -0.576196 0.576196
v 0.578505 -0.576196 0.576196
v -0.576196 0.578505 0.576196
v 0.578505 0.578505 0.576196
v -0.576196 0.578505 -0.578505
v 0.578505 0.578505 -0.578505
v -0.576196 -0.576196 -0.578505
v 0.578505 -0.576196 -0.578505
v -0.576196 -0.576196 0.576196
v -0.576196 -0.576196 0.576196
v 0.578505 -0.576196 0.576196
v 0.578505 -0.576196 -0.578505
v 0.578505 -0.576196 -0.578505
v 0.578505 0.578505 -0.578505
v -0.576196 -0.576196 -0.578505
```

```
v -0.576196 0.578505 -0.578505
v -0.576196 0.578505 -0.578505
vt 0.375000 0.000000
vt 0.625000 0.000000
vt 0.375000 0.250000
vt 0.625000 0.250000
vt 0.375000 0.500000
vt 0.625000 0.500000
vt 0.375000 0.750000
vt 0.625000 0.750000
vt 0.375000 1.000000
vt 0.375000 1.000000
vt 0.625000 1.000000
vt 0.875000 0.000000
vt 0.875000 0.000000
vt 0.875000 0.250000
vt 0.125000 0.000000
vt 0.125000 0.250000
vt 0.125000 0.250000
vn -0.894427 0.000000 0.447214
vn 0.447214 0.000000 0.894427
vn -0.408248 0.408248 0.816497
vn 0.666667 0.666667 0.333333
vn 0.000000 0.894427 -0.447214
vn 0.000000 0.447214 -0.894427
vn 0.000000 -0.447214 -0.894427
vn 0.000000 -0.894427 -0.447214
vn 0.000000 -1.000000 0.000000
vn 0.000000 -1.000000 0.000000
vn 0.000000 -1.000000 0.000000
vn 1.000000 0.000000 0.000000
vn 1.000000 0.000000 0.000000
vn 1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000
vn -1.000000 0.000000 0.000000

g pCube1
usemtl AmarilloSG
f 1/1/1 2/2/2 3/3/3
usemtl RojoSG
f 3/3/3 2/2/2 4/4/4
usemtl AzulSG
f 3/3/3 4/4/4 5/5/5
usemtl VerdeSG
f 5/5/5 4/4/4 6/6/6
usemtl RojoSG
f 5/5/5 6/6/6 7/7/7
usemtl AmarilloSG
f 7/7/7 6/6/6 8/8/8
usemtl VerdeSG
f 7/7/7 8/8/8 9/9/9
usemtl AzulSG
f 10/10/10 8/8/8 11/11/11
usemtl AmarilloSG
f 2/2/2 12/12/12 4/4/4
usemtl RojoSG
```

```
f 4/4/4 13/13/13 14/14/14
usemtl AmarilloSG
f 15/15/15 1/1/1 16/16/16
usemtl RojoSG
f 17/17/17 1/1/1 3/3/3
----------------------------------------------------------------------------------------------------
```

3DS

No se incluye la fuente binaria en esta sección pues no es legible, además el conjunto de caracteres ASCII comprende también caracteres no imprimibles. Por lo que la única comparativa que podemos hacer respecto a este formato es que el tamaño en bytes de un archivo 3DS es aproximadamente la 3ra parte de un archivo OBJ.

BTO

```
----------------------------------------------------------------------------------------------------
                                        Ejemplo 4.11
----------------------------------------------------------------------------------------------------
ABTO0202 ------------------------------------------------------------------
#                          Archivo BTO version 2.02
# -----INFO----------------------------------------------------------------
Vertices: 17
Vertices de Texturas: 17
Normales: 17
Poligonos: 12
Materiales: 4
Grupos: 1
Frames: 1
Secuencias: 1
Estados: 1
Tipo de Caras: 1
Triangulado: 1
Resumido: 0
Vida: 1
# ---1 Vertices------------------------------------------------------------
-0.499000 -0.499000 0.499000
0.501000 -0.499000 0.499000
-0.499000 0.501000 0.499000
0.501000 0.501000 0.499000
-0.499000 0.501000 -0.501000
0.501000 0.501000 -0.501000
-0.499000 -0.499000 -0.501000
0.501000 -0.499000 -0.501000
-0.499000 -0.499000 0.499000
-0.499000 -0.499000 0.499000
0.501000 -0.499000 0.499000
0.501000 -0.499000 -0.501000
0.501000 -0.499000 -0.501000
0.501000 0.501000 -0.501000
-0.499000 -0.499000 -0.501000
-0.499000 0.501000 -0.501000
-0.499000 0.501000 -0.501000
# ---2 Vertices de Texturas-----------------------------------------------
0.375000 0.000000
0.625000 0.000000
0.375000 0.250000
0.625000 0.250000
0.375000 0.500000
0.625000 0.500000
0.375000 0.750000
0.625000 0.750000
0.375000 1.000000
0.375000 1.000000
0.625000 1.000000
0.875000 0.000000
0.875000 0.000000
```

```
0.875000 0.250000
0.125000 0.000000
0.125000 0.250000
0.125000 0.250000
# ---3 Normales----------------------------------------------------------
-0.894427 0.000000 0.447214
0.447214 0.000000 0.894427
-0.408248 0.408248 0.816497
0.666667 0.666667 0.333333
0.000000 0.894427 -0.447214
0.000000 0.447214 -0.894427
0.000000 -0.447214 -0.894427
0.000000 -0.894427 -0.447214
0.000000 -1.000000 0.000000
0.000000 -1.000000 0.000000
0.000000 -1.000000 0.000000
1.000000 0.000000 0.000000
1.000000 0.000000 0.000000
1.000000 0.000000 0.000000
-1.000000 0.000000 0.000000
-1.000000 0.000000 0.000000
-1.000000 0.000000 0.000000
# ---4 Materiales--------------------------------------------------------
Material 0 AmarilloSG
Textura NULL
Illum 4
Kd 1.000000 1.000000 0.000000 1.000000
Ka 0.500000 0.500000 0.000000 1.000000
Ke 0.200000 0.200000 0.000000 1.000000
Ni 100.00
Ks 0.333333 0.333333 0.000000 1.000000
Solido 1.00
#-------------------------------------------------------
Material 1 RojoSG
Textura NULL
Illum 4
Kd 1.000000 0.000000 0.000000 1.000000
Ka 0.500000 0.000000 0.000000 1.000000
Ke 0.200000 0.000000 0.000000 1.000000
Ni 100.00
Ks 0.333333 0.000000 0.000000 1.000000
Solido 1.00
#-------------------------------------------------------
Material 2 AzulSG
Textura NULL
Illum 4
Kd 0.000000 0.000000 1.000000 1.000000
Ka 0.000000 0.000000 0.500000 1.000000
Ke 0.000000 0.000000 0.200000 1.000000
Ni 100.00
Ks 0.000000 0.000000 0.333333 1.000000
Solido 1.00
#-------------------------------------------------------
Material 3 VerdeSG
Textura NULL
Illum 4
Kd 0.000000 1.000000 0.000000 1.000000
```

```
Ka 0.000000 0.500000 0.000000 1.000000
Ke 0.000000 0.200000 0.000000 1.000000
Ni 100.00
Ks 0.000000 0.333333 0.000000 1.000000
Solido 1.00
# ---5 Poligonos------------------------------------------------------------
Material: 0 AmarilloSG
0/0/0 1/1/1 2/2/2
Material: 1 RojoSG
2/2/2 1/1/1 3/3/3
Material: 2 AzulSG
2/2/2 3/3/3 4/4/4
Material: 3 VerdeSG
4/4/4 3/3/3 5/5/5
Material: 1 RojoSG
4/4/4 5/5/5 6/6/6
Material: 0 AmarilloSG
6/6/6 5/5/5 7/7/7
Material: 3 VerdeSG
6/6/6 7/7/7 8/8/8
Material: 2 AzulSG
9/9/9 7/7/7 10/10/10
Material: 0 AmarilloSG
1/1/1 11/11/11 3/3/3
Material: 1 RojoSG
3/3/3 12/12/12 13/13/13
Material: 0 AmarilloSG
14/14/14 0/0/0 15/15/15
Material: 1 RojoSG
16/16/16 0/0/0 2/2/2
# ---6 Grupos--------------------------------------------------------------
0 pCube1 (0 -> 11)
Padre: NULL
Colisionable: Si
HP: 1
Pivote: (0.0000,0.0000,0.0000)
Centroide: (0.0010,0.0010,-0.0010)
Radio: 0.8660
XMax: 0.5010 Xmin: -0.4990
YMax: 0.5010 Ymin: -0.4990
ZMax: 0.4990 Zmin: -0.5010
Keyframes: 1
( 0, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.00, 1.00, 1.00, 1)
# ---7 Secuencias-----------------------------------------------------------
0 Default (0->0)
# ---8 Estados-------------------------------------------------------------
# 0-----------------------------------------------------------
Default
Secuencia: 0
Salidas: 0
# ---9 Campo de Accion-----------------------------------------------------
Centroide: (0.001000,0.001000,-0.001000) Radio: 0.866025
XMax: 0.501000 Xmin: -0.499000
YMax: 0.501000 Ymin: -0.499000
ZMax: 0.499000 Zmin: -0.501000
# -------------------------------------------------------------------------
# El principio fundamental de nuestro amor hacia los hombres, es que los
```

```
# débiles y los fracasados no solo deben morir, se les debe ayudar a perecer.
# -------------------------------------------------Friedrich Nietzsche-------
# Archivo generado el: Fri May 21 19:32:05 2010
 ------------------------------------------------------------------------------------------
```

Comentario: Si bien el archivo BTO es más extenso que el archivo OBJ también es de considerarse que ofrece mucha más información que él. Por otro lado la forma ASCII en el que se encuentra el formato siempre ocupara más espacio que la forma binaria en la que está escrita el archivo 3DS.

Es importante recalcar también que la facilidad de lectura que ofrece el formato BTO es mayor a la que cualquiera de los otros dos archivos ofrece.

4.6 Código BTO y Visor BTO

Visor BTO es un programa especialmente diseñado para mostrar y convertir modelos tridimensionales 3DS y OBJ (así como archivos auxiliares) a formato BTO. El programa maneja apuntadores a objetos y la posición, ángulo, tamaño y estado actual de cada modelo puede ser guardado en un archivo WRL que es una lista de modelos, esto permite cargar todos los modelos de una sola vez, al abrir este archivo, haciéndolo ideal para manejar escenas.

Así mismo los archivos de código permiten utilizar BTO's en programas propios, de tal manera que el programador no tiene que preocuparse por entender a profundidad el manejo de modelos tridimensionales o de OpenGL. Una gran utilidad que tienen estos archivos de código, es que permiten portabilidad a otros API's como OGRE.

Lo primero que debe de hacerse en el programa para utilizar modelos es declarar un objeto del tipo Modelos o del tipo Mundos si se pretende utilizar más de un modelo en el programa:

Modelos mModelo;

ó

Mundos mMundo;

Estos tipos son en realidad clases de C++, lo que permite incluir en ellos mismos los métodos para manipularlos. Así mismo la mayoría de los métodos incluyen opciones por defecto, de tal manera que el usuario puede conocer fácilmente que colocar en cada parámetro de la función o en algunos casos simplemente omitir el dar valor alguno.

4.6.1 Métodos de Carga del archivo

Los métodos de carga de archivo existentes en el DLL permiten con facilidad al programador utilizarlos, la descripción de sintaxis y parámetros requeridos están descritos en la documentación. Basta con mencionar que se tiene una implementación de un método de carga para modelos y mundos que permite indicar si se generara una barra de progreso al momento de dibujar en pantalla.

4.6.2 Métodos de Dibujo

Se cuenta con varios métodos de dibujo de la escena según sea el caso de Modelos o Mundos. Particularmente en el caso de la clase Modelos se cuenta con diferentes funciones de dibujo, por si se desea dibujar polígonos, grupos, frames completos, animaciones completas o estados completos, sin embargo se recomienda al programador se utilice el método Dibuja pues el dibuja el modelo en pantalla de forma

automática sin que el programador tenga que preocuparse en seleccionar el tipo de dibujo, se dejan los otros métodos abiertos para que un programador más avanzado decida qué tipo de dibujo le resulta más conveniente. Asimismo se incluye una implementación de dibujo por VBO's lo que acelera drásticamente el dibujo en pantalla, la opción de dibujo por este medio se hace de forma automática en la función Dibuja si el programa detecta que se cuenta con una tarjeta de video compatible.

4.6.3 Algoritmos para detección de colisiones

Detectar cuando un objeto esta colisionando con otro en la computadora no es un asunto tan trivial, se deben de evaluar diferentes aspectos, la manera que daría mejores resultados seria evaluar polígono a polígono si existe una colisión con otro polígono, desafortunadamente, el evaluar las colisiones de esta manera toma demasiado tiempo en procesador, lo que hace que el programa tienda a alentarse a mayor número de polígonos. La solución mejor aplicada es tomar criterios y satisfacer ciertos requisitos, como determinar que hay polígonos que no deben de calculársele colisiones con otro polígono que físicamente no puede tocar, o determinar objetos no colisionables, entre otros.

Es por ello que se tienen ciertos algoritmos básicos para el control de colisiones, mediante los cuales la computadora se ahorra gran cantidad de operaciones.

4.6.3.1 Esfera

Este es uno de los algoritmos más sencillos de implementar, requiere conocer el centroide del objeto estudiado, y conocer el vértice más alejado del mismo, de esta manera se puede calcular el radio de la esfera de colisión, lo que reduce el problema de calcular la colisión a una simple resta en el caso de calcular colisiones esfera-esfera (o lo que es lo mismo una esfera contra otra). Si la distancia que hay entre dos centroides de esferas de colisión es menor a la suma de sus dos radios entonces una esfera está dentro de la otra (hay colisión).

Desafortunadamente el gran problema de este algoritmo es que la esfera es la figura que menos se ajusta a la forma del objeto (a menos de que tenga forma esférica, en cuyo caso es la mejor) generando que se registren colisiones en regiones cercanas al modelo donde no se encuentra ningún polígono.

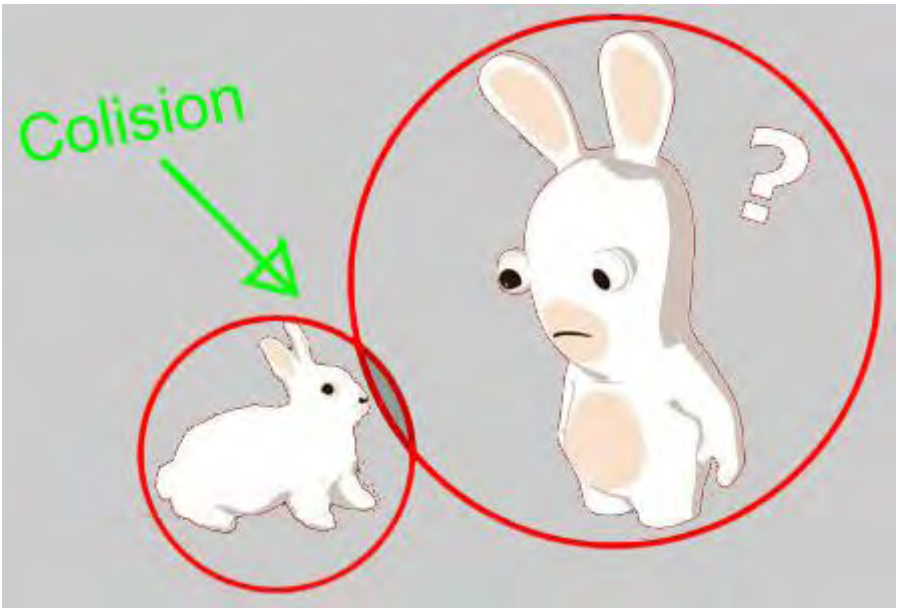


Figura 4.2 – Colisión Esférica.

4.6.3.2 Cilindro

Es un algoritmo similar al de la esfera de colisión, consiste en determinar un radio de colisión en un plano, o lo que es lo mismo, calcular el vértice más alejado del centroide de la figura para todos los cortes en un plano. Posteriormente se calcula la altura del modelo y se tiene un cilindro que ajusta más acorde a ciertos objetos, lo que genera un algoritmo híbrido entre caja orientada a ejes y esfera.

4.6.3.3 Caja orientada a los ejes

Este algoritmo se ajusta más a los objetos que la esfera de colisión, se calcula una serie de coordenadas máximas y mínimas con las que se construye una caja que se ajusta al modelo. De esta forma mediante sumas y restas se calcula si un objeto está dentro de las X, Y y Z del otro objeto, si existen aciertos entonces existe una colisión.

Pese a su fácil implementación y sencillos cálculos, la caja orientada a los ejes presenta un problema, como su nombre lo indica debe estar orientada de la misma manera que la otra caja calculada, o lo que es lo mismo ambos objetos tengan los mismos ejes de referencia, si un objeto gira, aunque sea en un grado sobre cualquiera de sus ejes, las pruebas arrojan resultados falsos.

Figura 4.3 – Colisión por cajas.

4.6.3.4 Rayo

Este algoritmo solo sirve para calcular colisión de una línea con un objeto, la finalidad de este algoritmo es calcular sombras o colisiones de objetos que se desplazan sobre ella. Se tiende una línea de un punto origen a un punto destino y si esta línea traviesa un polígono se dice que hay colisión.

Figura 4.4 – Colisión Rayo - Plano

4.6.3.5 Algoritmos para detección de colisiones aplicados

Visor BTO y BTO.dll utiliza un sistema particular de detección de colisiones, primero se calcula un radio en torno del modelo, este radio es llamado campo de acción. Después si dos campos de acción están en contacto se calculan las colisiones correspondientes, esto hace que solo se tenga que calcular si el campo de acción de todos los objetos de la escena están en contacto con un modelo de interés, si es así se pueden calcular ya

las colisiones entre estos modelos, reduciendo enormemente el cálculo de colisiones en la escena.

La secuencia para determinar colisión de objetos seria entonces:

A) Se calculan colisiones esfera-esfera para todos los modelos tomando sus campos de acción.
B) Si hay colisión con dos campos se evalúa la colisión esfera-esfera esfera-caja o caja-caja con sus cajas orientadas a ejes o sus esferas de colisión para el frame en el que se encuentren.
C) Si hay colisión entre ellas, entonces se procede a calcular colisiones esféricas entre los grupos colisionables mediante la colisión por grupos.

A continuación se describe cada uno de ellos.

4.6.3.6 Campo de acción

El campo de acción es un radio que nace en el centroide de cada modelo, la longitud de este radio está determinada por el vértice del modelo más alejado del centroide, para cada frame de animación. De esta manera el modelo, para cualquier frame dado se encontrará inmerso en una esfera. Gracias a esto solo tenemos que calcular colisiones esfera - esfera para saber si un objeto está en el campo de acción de otro y si es así calcular las colisiones.

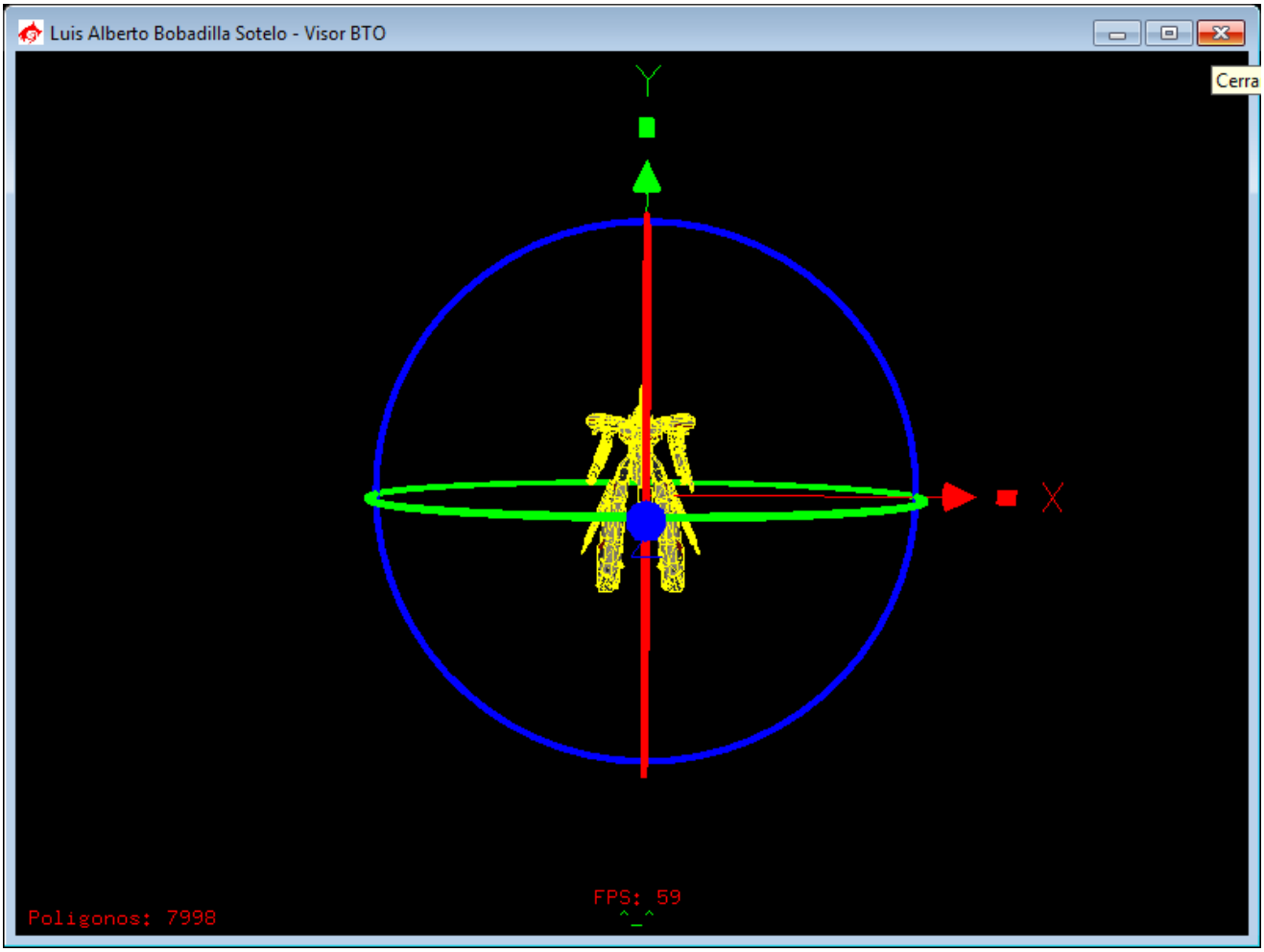


Figura 4.5 – Campo de Acción.

4.6.3.7 Caja Orientada a los Ejes

Las cajas orientadas a los ejes se calculan para cada frame, de tal manera que un modelo tiene una caja orientada para cada frame, se evalúa el frame actual y se puede saber si hay colisión rayo – caja, esfera – caja, cilindro – caja, caja – caja.

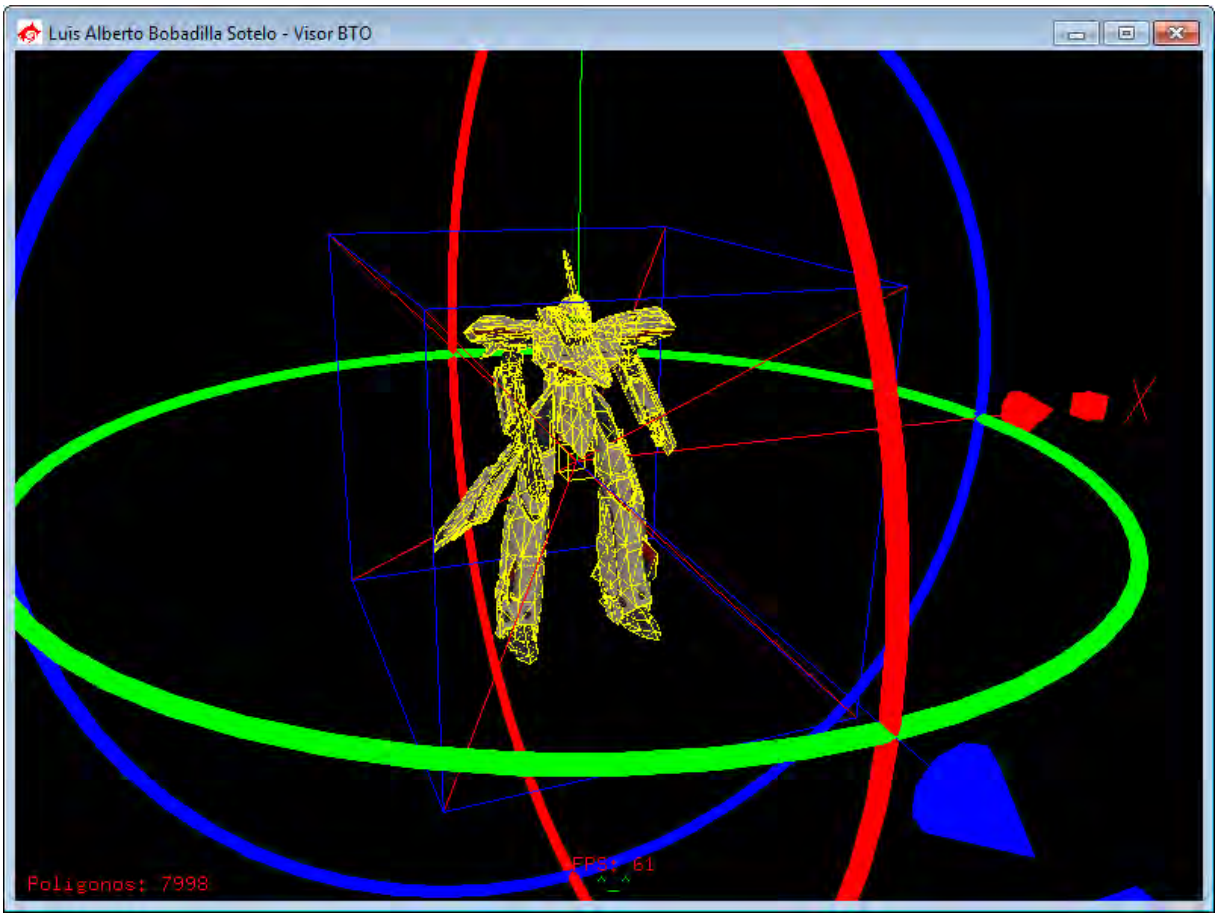


Figura 4.6 – Caja Orientada a los ejes.

4.6.3.8 Esfera de colisión.

Al igual que las cajas orientadas a los ejes, cada frame tiene su propia esfera de colisión, de esta manera, se pueden evaluar colisiones rayo – esfera, esfera – esfera, esfera – caja, cilindro – esfera.

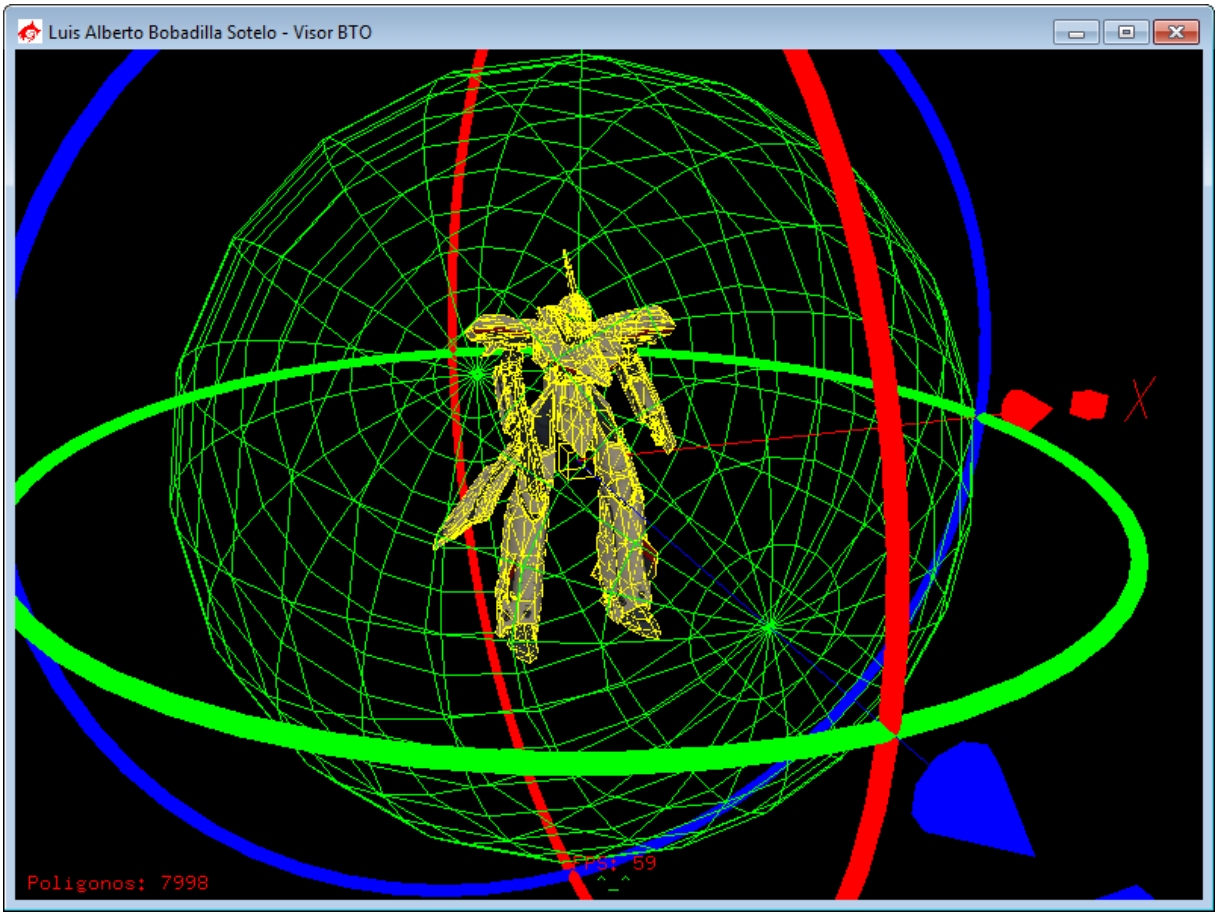


Figura 4.7 – Esfera de Colisión.

4.6.3.9 Grupos colisionables

Cada grupo del modelo tiene un centroide específico y una esfera de colisión especifica. Esta esfera es la misma para cada frame, pero al afectar la posición global del centroide no se tienen que recalcular las posiciones de las esferas.

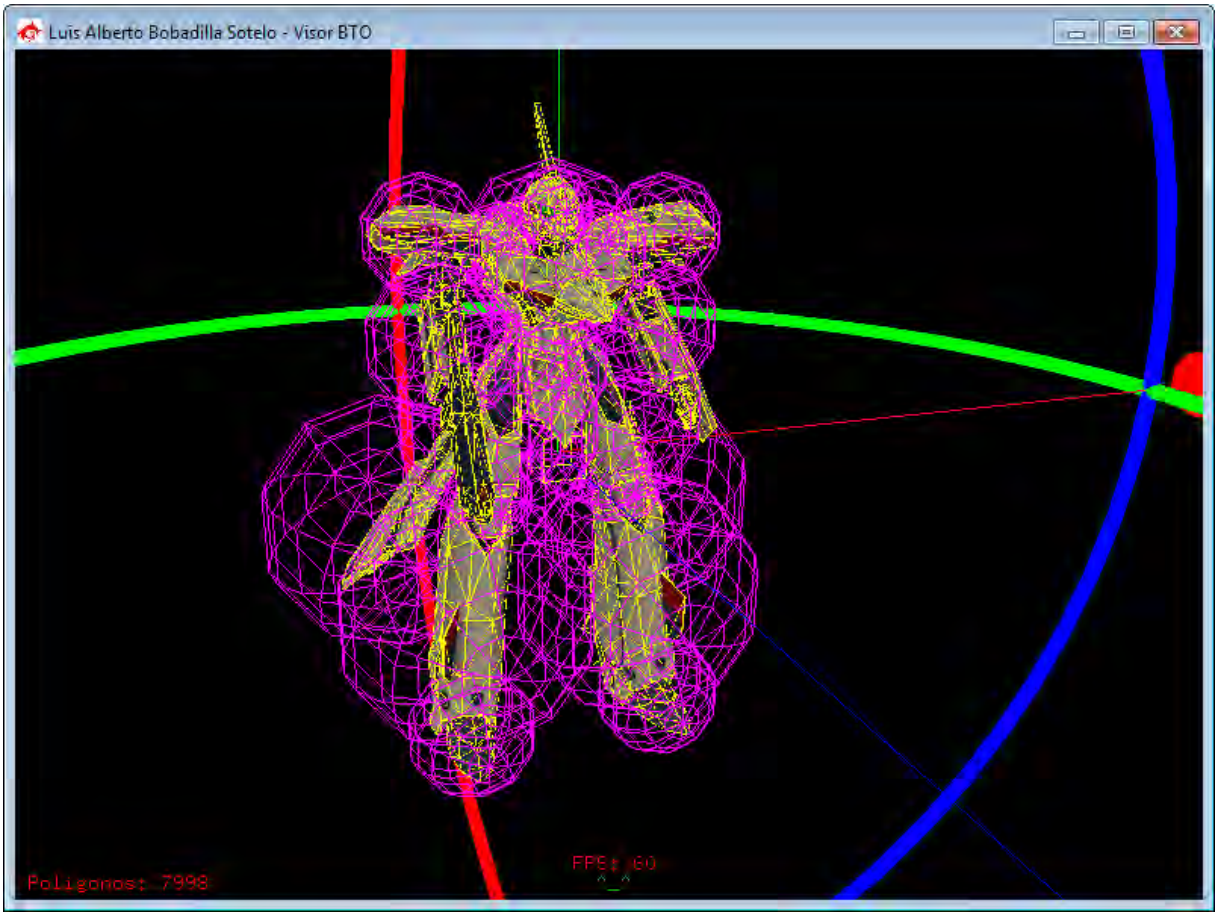


Figura 4.8 – Grupos Colisionables.

CAPÍTULO 5. CAMPOS DE APLICACIÓN

5.1 Campos aplicables

El formato BTO puede ser aplicado en una gran cantidad de campos, desde la enseñanza hasta la industria del entretenimiento, pues a fin de cuentas su aplicación es la misma que la de las gráficas. Sin embargo el código de cargado de BTO ayuda al programador inexperto a no requerir entender los procesos de carga de diferentes formatos, reduciendo su trabajo a programas las rutinas que realmente son de su interés.

5.2 Uso como soporte a otro software.

Se desarrollaron diferentes programas de muestra que utilizan el archivo BTO.dll para demostrar las utilidades y la practicidad con la que se pueden desarrollar nuevos programas basados en la carga de modelos y conversión de modelos a formato BTO.

5.3 Visualización - Visor y editor de mundos

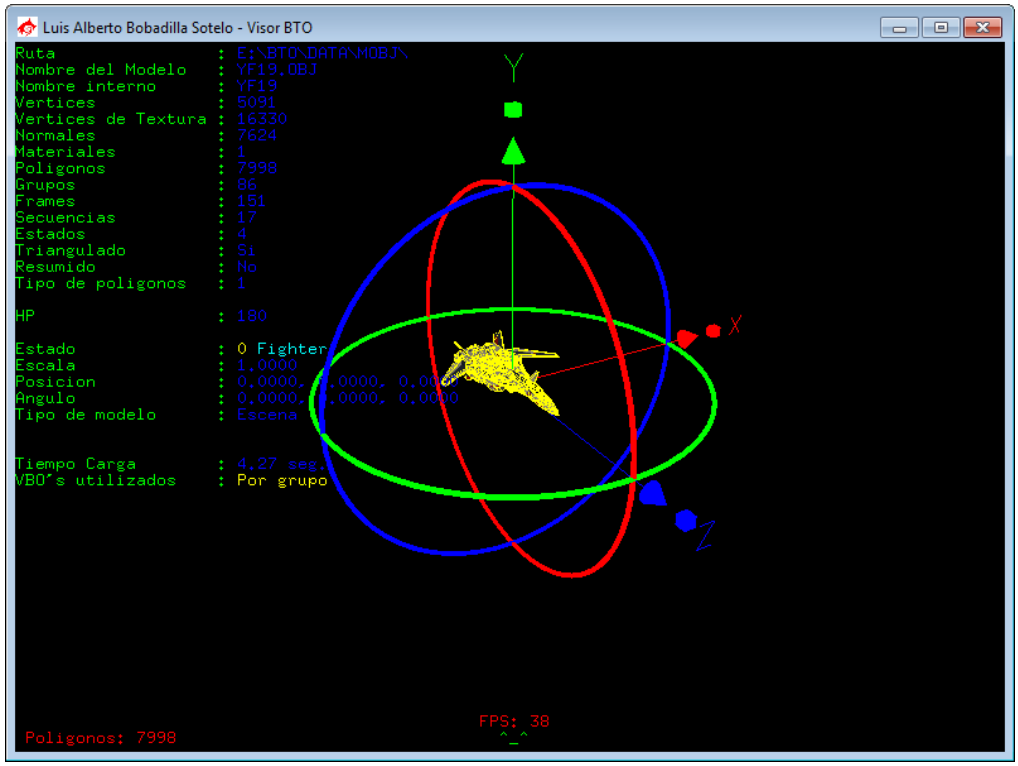


Figura 5.1 – Visor y editor de mundos.

Por medio del dll se realizó un editor de mundos y modelos, en primera instancia es un visor de modelos tridimencionales, que acepta entradas 3DS, OBJ y BTO. Utiliza funciones del dll que permiten al usuario relacionar mediante el “Editor de Registro” de Windows estas extenciones, al igual que las .WRL nativas, para abrir automáticamente con el programa.

En él se permite la carga de múltiples modelos de diferentes formatos y la escritura de un archivo .WRL nativo para la correcta colocacion y carga del los modelos.

La idea del archivo .WRL es cargar los modelos de una manera mas autónoma e independiente de la programación de la aplicación en si misma.

En la carpeta de ejecucion del programa debe de estar contenido el dll, asi como el archivo de configuración del programa, en el caso de que el último no existiera y el programa se ejecute, se crea de manera automática con las opciones por defecto.

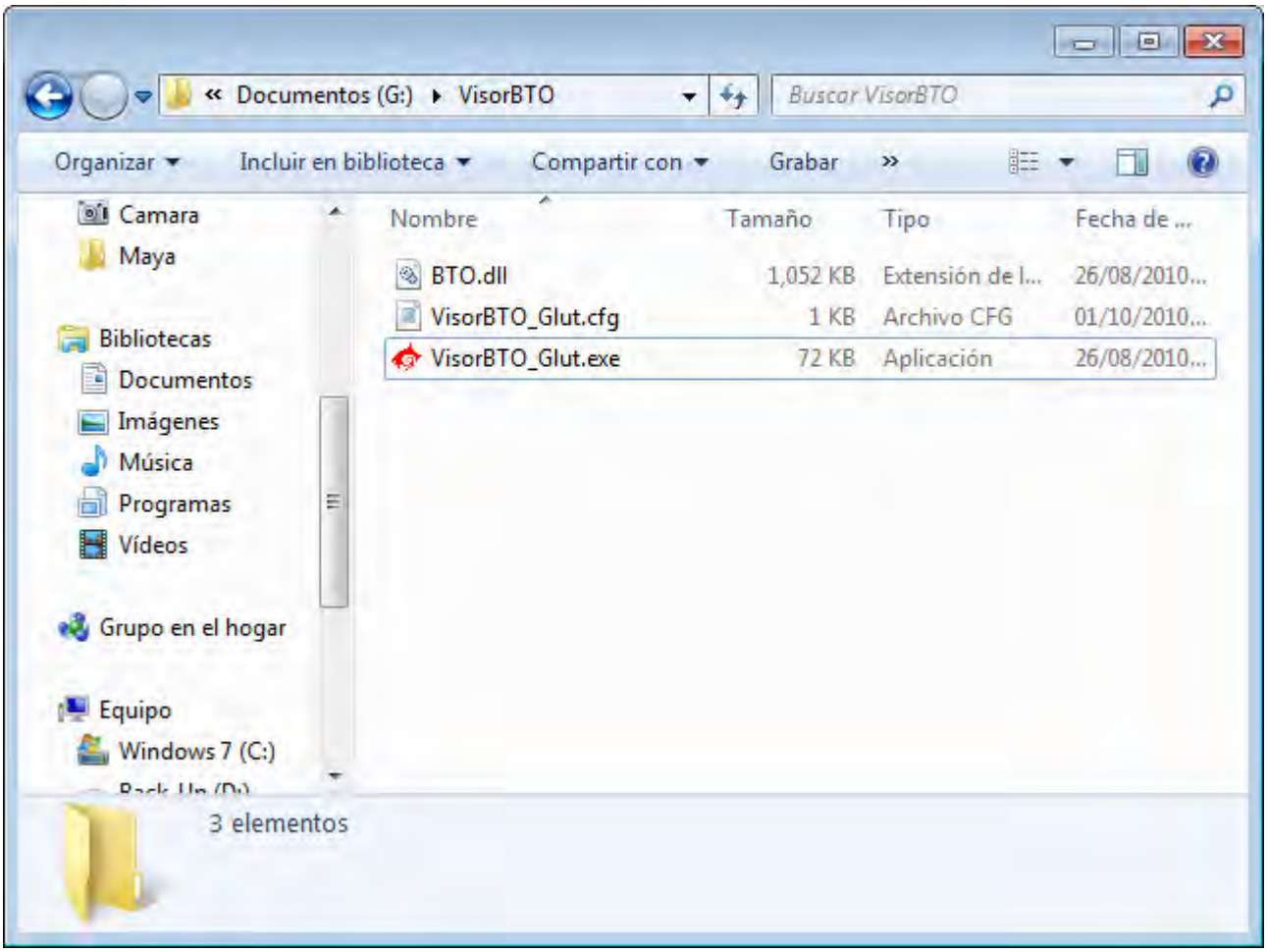


Figura 5.1.1 – Carpeta de ejecución.

El archivo de configuración consiste en una serie de valores que el programa puede tomar para simplificar el trabajo del usuario, en el se incluyen la ultima ruta conocida, asi como la configuración de el tamaño de la ventana en donde se ejecutará el programa, de igual forma se sugieren tamaños predeterminados de ventanas. El archivo de configuración utiliza la misma sintaxis te comentarios de un archivo BTO.

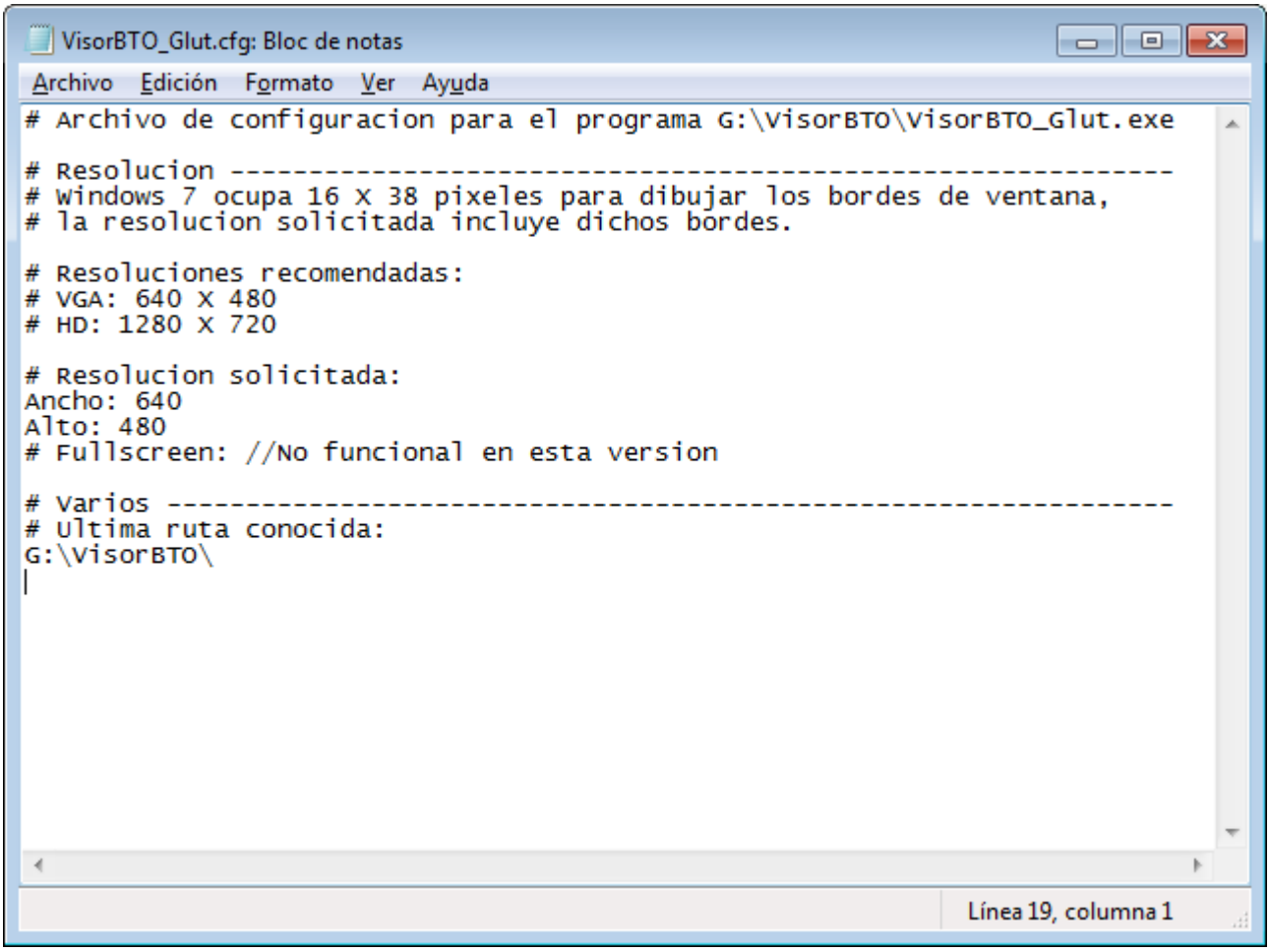


Figura 5.1.2 – Archivo de configuración.

El programa Visor BTO permite el uso del mouse para rotar, transladar y acceder a menús dentro del programa, las opciones del menú principal dependeran de si se tiene o no seleccionado un modelo en el programa.

Las opciones básicas del programa permiten modificar la cantidad de modelos que se encuentren en la escena y la Escena misma.

Los modelos pueden ser agregados dentro del submenú “Archivo” de igual forma se pueden cargar varios modelos de golpe mientras esten contenidos en un archivo de

mundo (WRL), de igual manera se permite descargar todos los modelos del mundo juntos mediante la opcion “Descarga Mundo”.

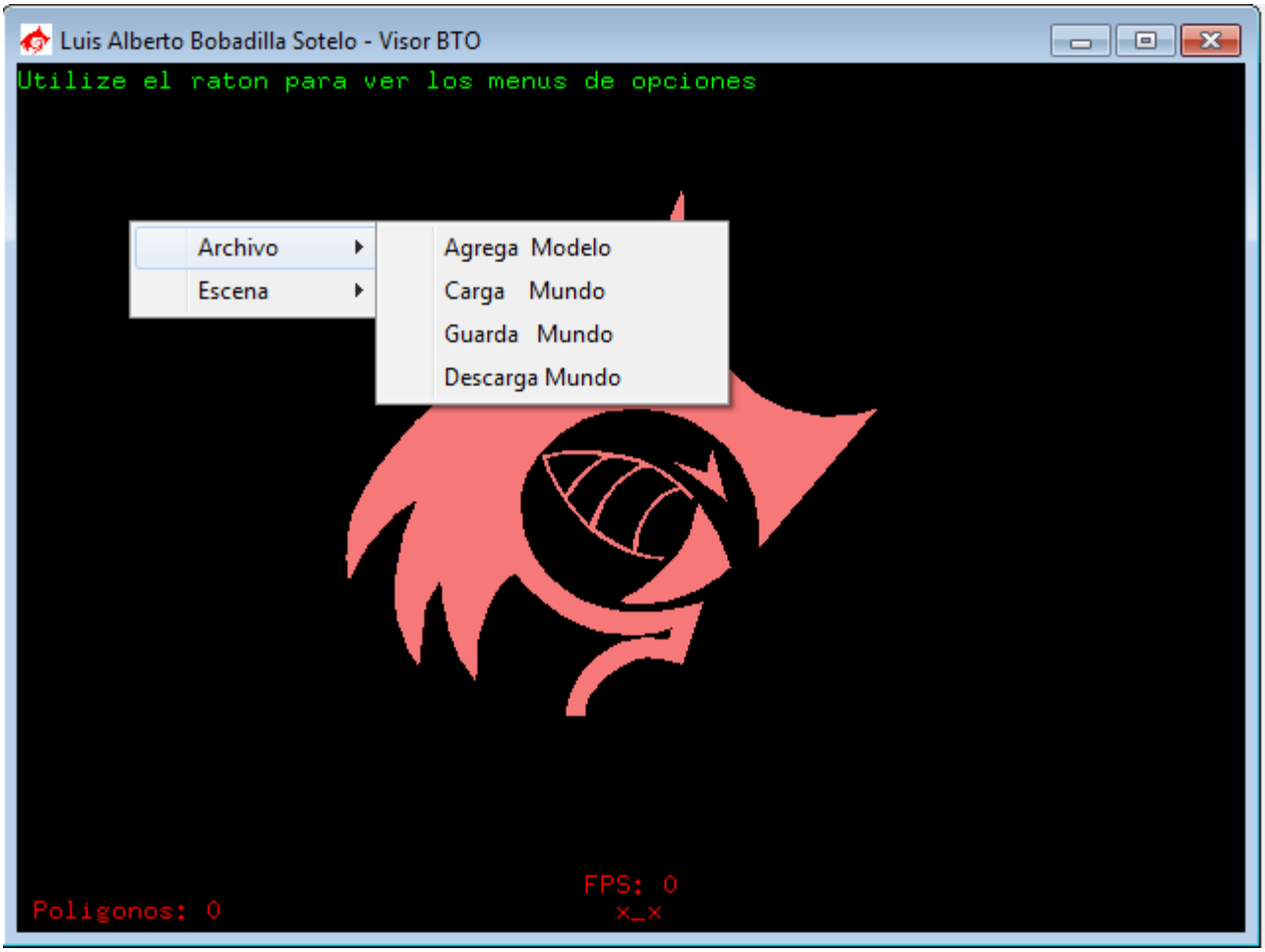


Figura 5.1.3 – Menú Archivo.

La opción “Agrega Modelo” dentro del menú “Archivo” ejecuta una venta de dialogo que permite seleccionar el tipo de archivo a abrir, estos pueden ser bto, .obj o .3ds. Una vez seleccionado el archivo de modelo tridimensional a cargar bastara con dar click en abrir para cargar el modelo especificado.

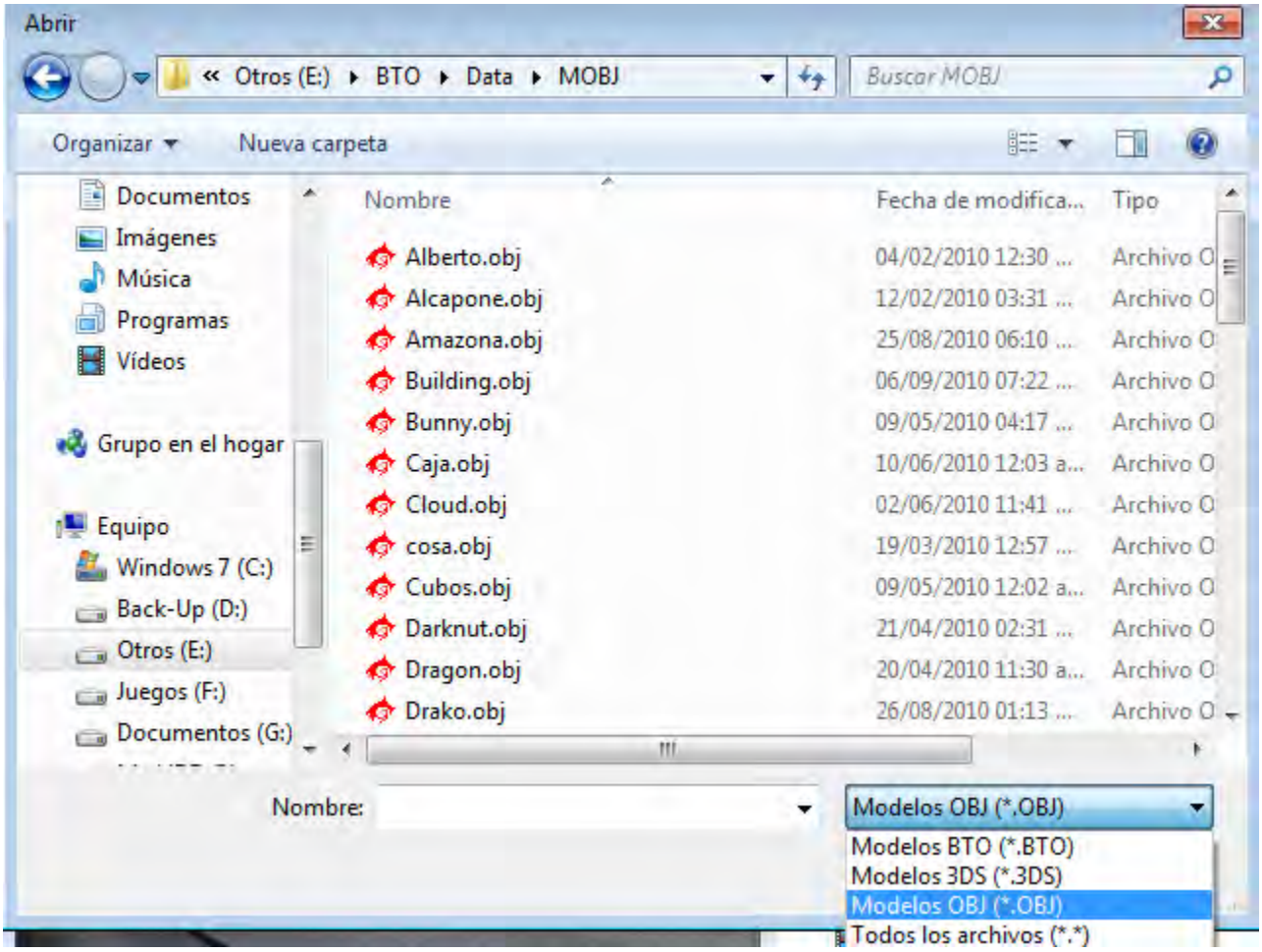


Figura 5.1.4 – Dialogo Abrir.

Una vez que se encuentre abierto al menos un modelo, el espacio de trabajo cambia al de edición. En el si se cuenta exactamente con un modelo, este se toma como el modelo por defecto y muestra la información correspondiente a dicho modelo. Si se cuenta con más de un modelo en memoria, es necesario seleccionar un modelo para mostrar su información. El mostrar la información de un modelo puede ser habilitado o deshabilitado en el menú “Escena” como se verá más adelante.

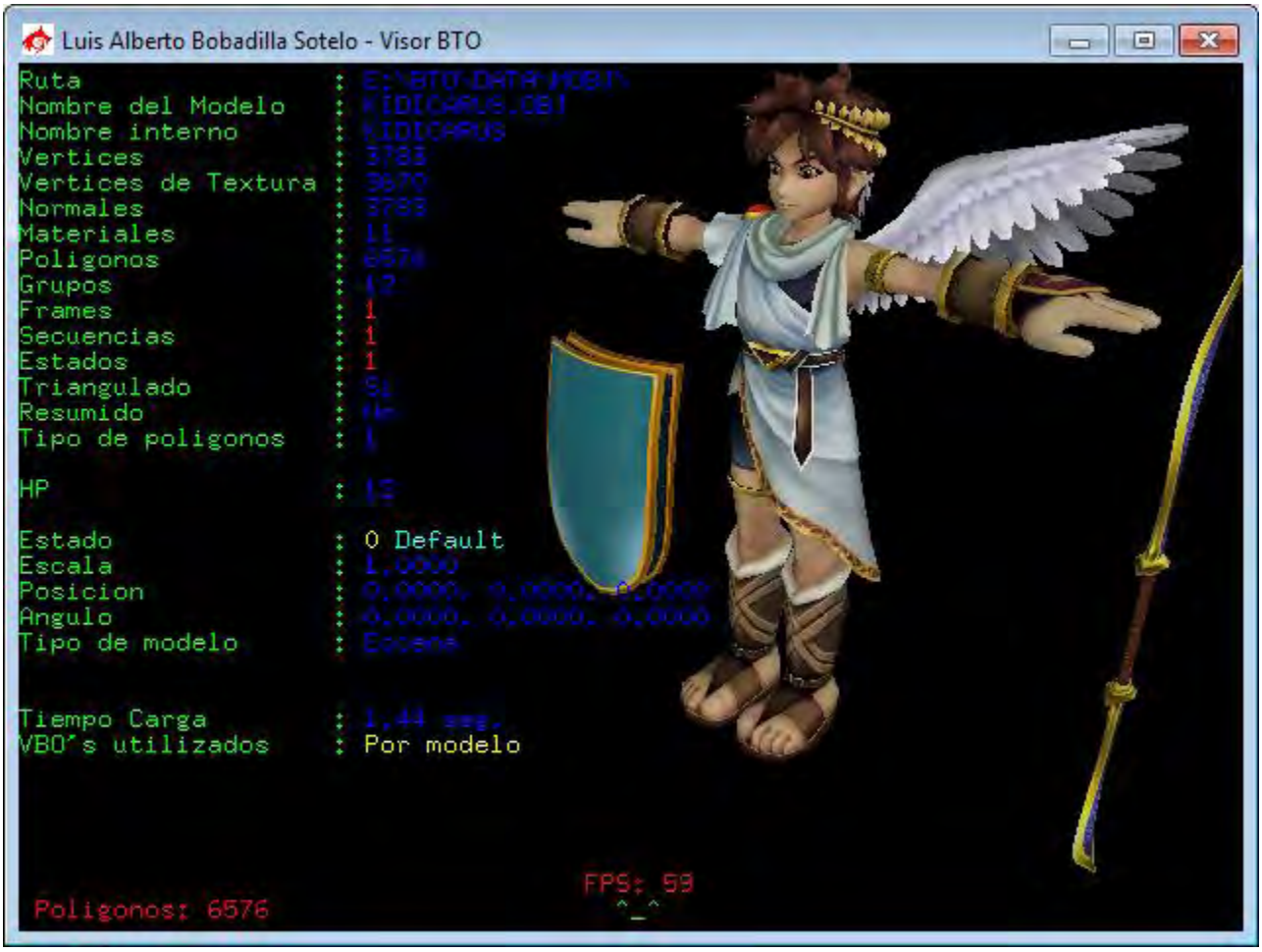


Figura 5.1.5 – Información del modelo.

El menú “Escena” contiene opciones que permiten modificar el comportamiento del espacio de edición, dado que la cámara puede ser movida libremente dentro del “mundo” la opción “Reset” la regresa al origen del mundo, apuntando directamente al origen (0,0,0); la opción “Información” como se comentó con anterioridad permite habilitar o deshabilitar la información referente al modelo seleccionado o el seleccionado por defecto; las opciones “Regla XY” y “Regla YZ” permiten habilitar o deshabilitar reglas de edición en los mencionados planos; la opción “Luces” permite modificar la manera de presentar el modelo, con o sin luces, para apreciar el modelo en un ambiente obscuro; la opción “Fondo” permite cambiar el color de fondo de la aplicación, así como el color de las letras con el fin de lograr un contraste con los

modelos más obscuros; la opción “Rota solo” hace que el modelo seleccionado gire sobre su eje Y con el fin de apreciarlo en su totalidad sin tener que modificar la posición de la cámara; por último, el programa esta optimizado y se ejecuta mediante opciones que utilizan mejor los recursos disponibles en la computadora, es de esta manera que el programa al cargar o modificar un modelo manda la información correspondiente directamente a la tarjeta de video, con el fin de evitar cuellos de botella y minimizar las rutas que los datos tienen que recorrer entre el CPU y el GPU mediante el uso de VBO´s (Vertex Buffered Objects por sus siglas en inglés), dependiendo de la cantidad de información contenida en un modelo y la capacidad del CPU puede ser que no se genere una gran diferencia tener o no activada esta opción, sin embargo para modelos con una gran carga de información es vital contar con la opción VBO´s habilitada, en pruebas realizadas con modelos de más de un millón de polígonos se logró levantar el Frame Rate (la cantidad de pantallas que se dibujan por segundo) de 2 y 4 FPS a 50+ FPS. Si la computadora donde se está corriendo la aplicación no contara con una tarjeta de video o una tarjeta de video compatible con VBO´s el programa logra identificarlo y dibuja los modelos sin compatibilidad de VBO´s sin afectar el funcionamiento general del programa, pero reduciendo drásticamente el desempeño en modelos más grandes.

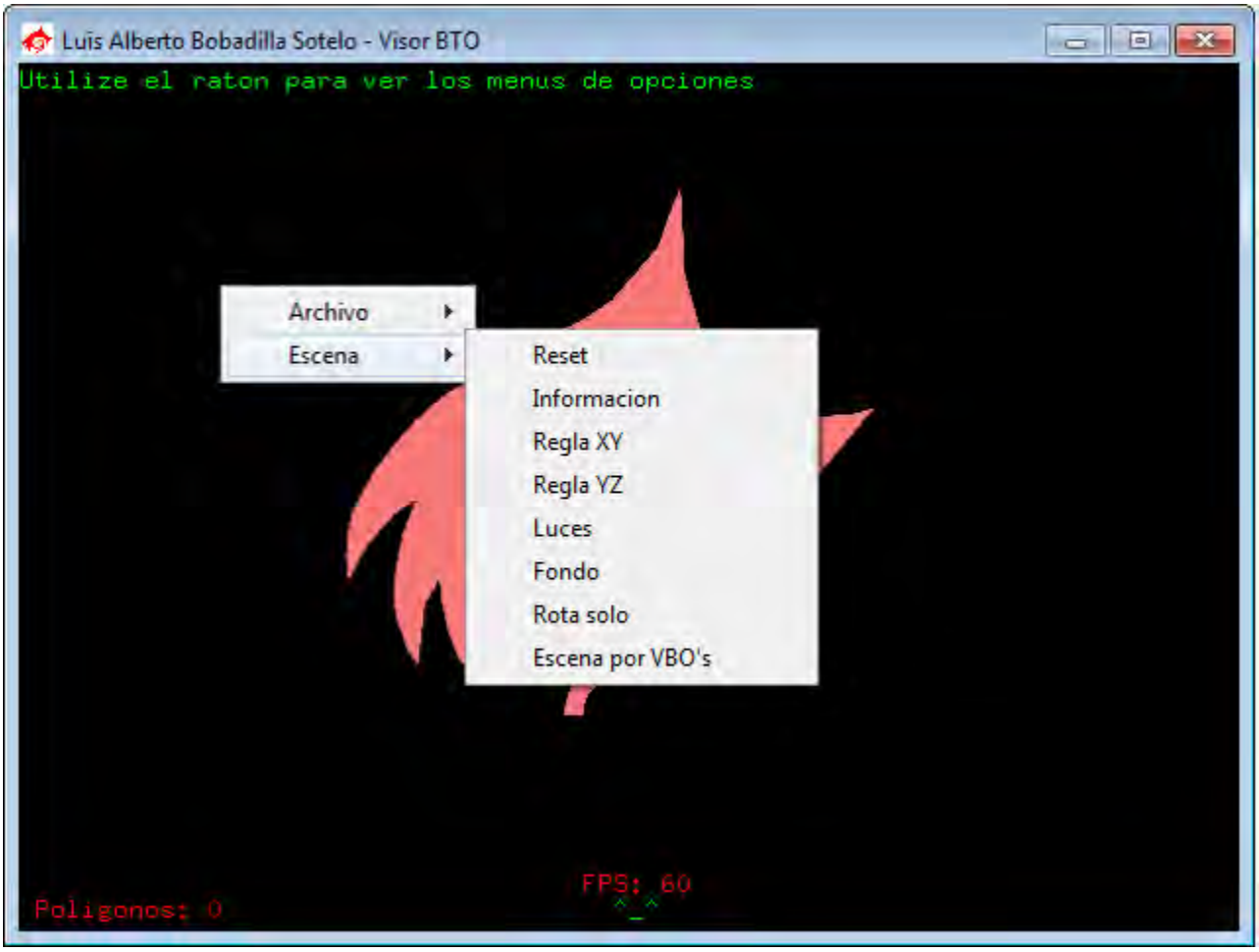


Figura 5.1.5 – Menú Escena.

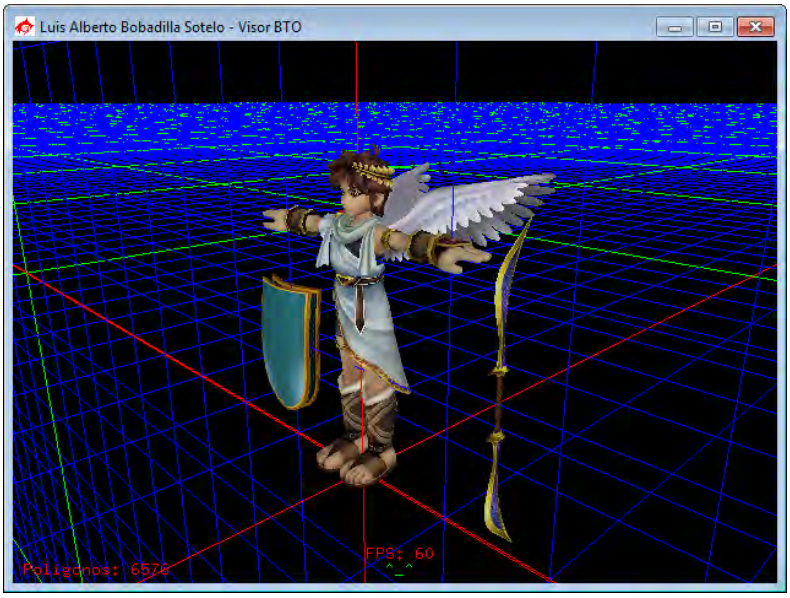


Figura 5.1.5.1 – Espacio de trabajo con reglas activadas.

Si un modelo se encuentra seleccionado el menú principal cambia y da opciones específicas para el modelo seleccionado. Entre ellas se encuentra la opción "Wireframe" que permite visualizar el modelo en modo alambres; las opciones "AABB", "BS" y "Colisionables", permiten visualizar la Caja orientada a los ejes, La esfera de colisión y los grupos colisionables del modelo respectivamente.

Existen también dos submenús con opciones relacionadas agrupadas. El primer grupo es el de funciones, en él se encuentran funciones que permiten modificar las normales (que como ya se explicó en capítulos anteriores ayudan a determinar la reflexión de la luz sobre el modelo), pudiendo ser por cara o por vértice; el segundo grupo es el de tamaño.

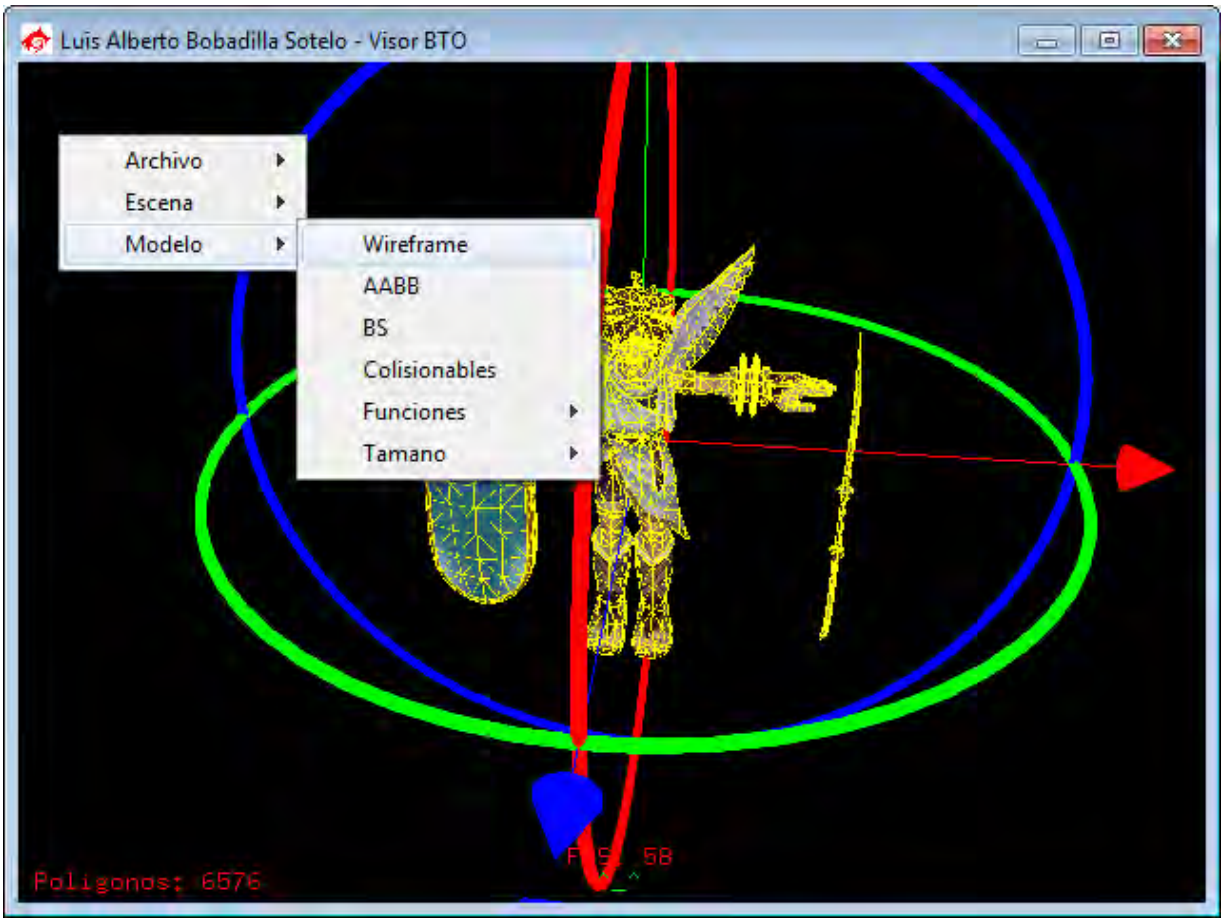


Figura 5.1.6 – Menú Modelo.

El submenú tamaño permite modificar el tamaño que tiene el modelo respecto al mundo, la opción “Original” permite visualizar el modelo con el tamaño con el que fue concebido en el programa de diseño; la opción “Unitario” reduce o amplia el modelo de tal manera que este se encuentre contenido en una caja imaginaria con dimensiones unitarias; la opción “Máximo 100” permite un tamaño libre de modelos hasta llegar a 100 unidades, si el modelo llegara a sobrepasar estas dimensiones se reduce de manera automática para cumplir con esta restricción; las opciones “Tamaño X2”, “Tamaño X5” y “Tamaño X10” modifican el tamaño del modelo en esos múltiplos.

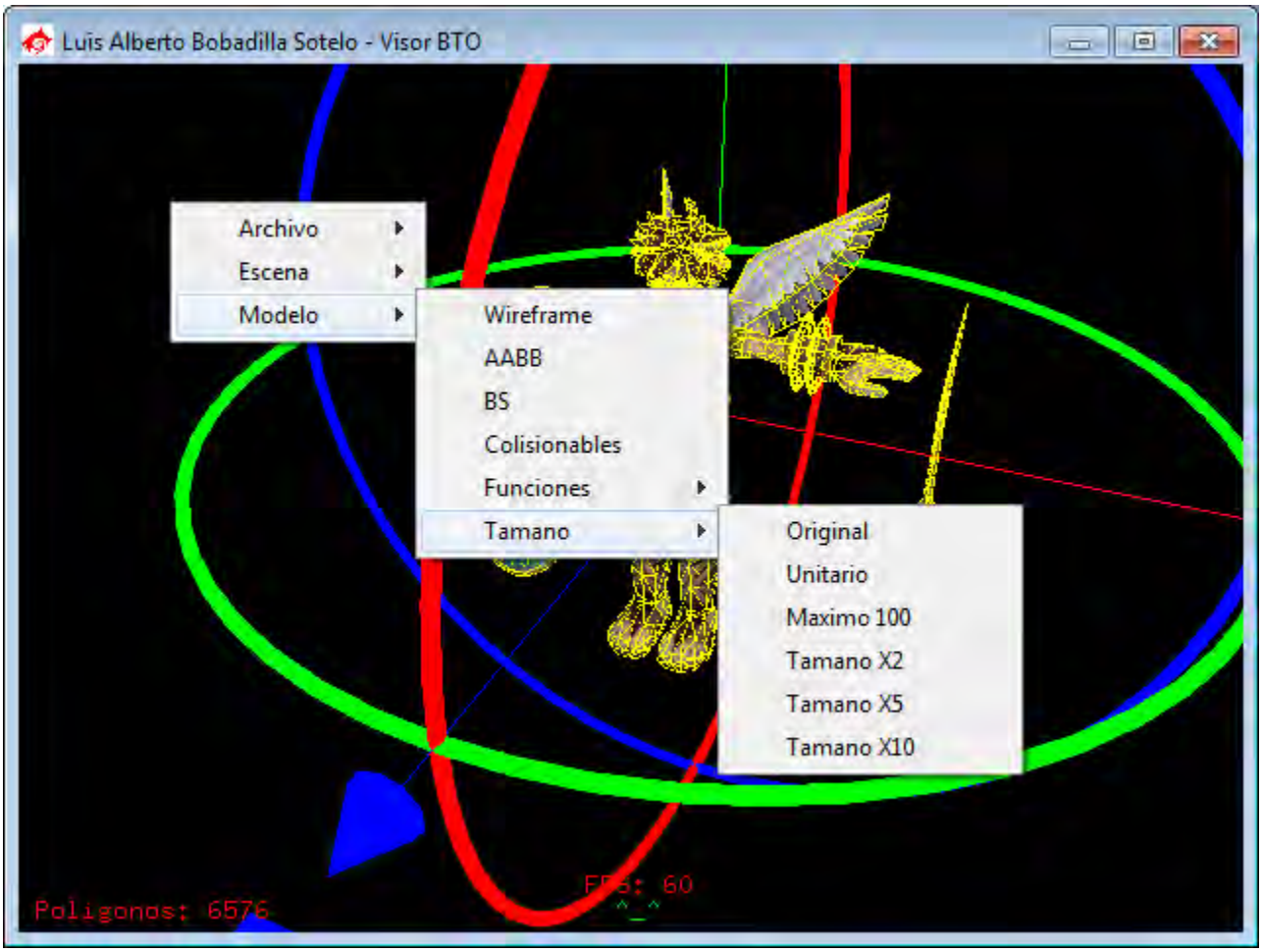


Figura 5.1.5 – Submenú Tamaño.

5.3.1 Enseñanza – Mano Amiga

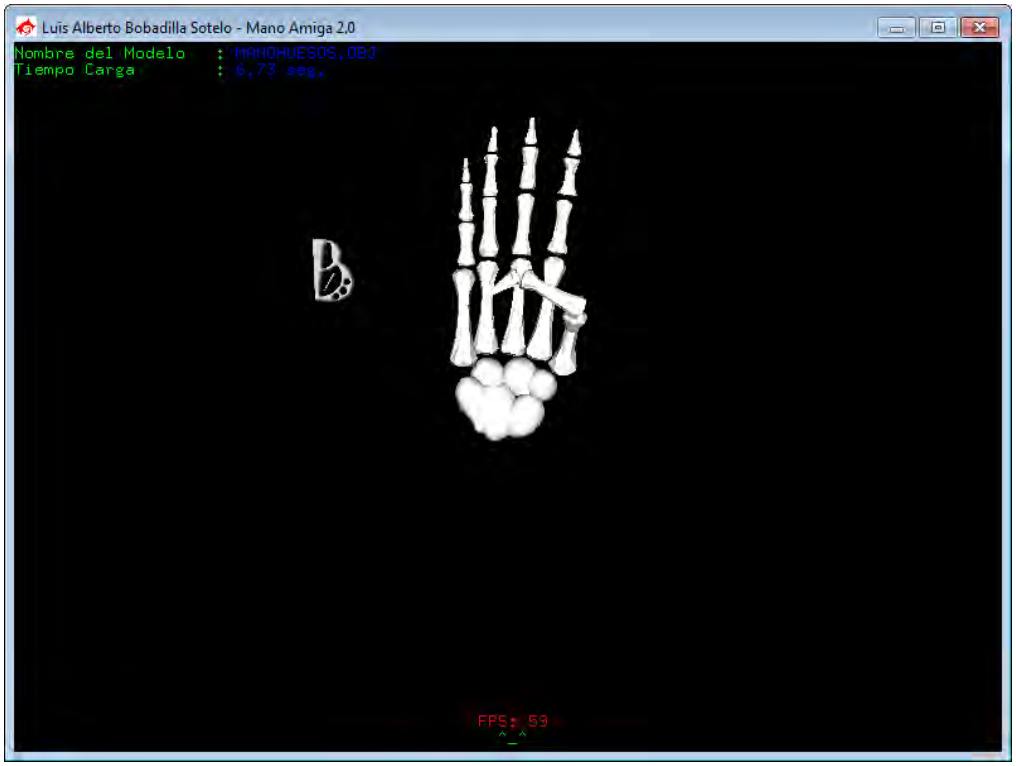


Figura 5.2 – Mano Amiga.

Mano Amiga es una aplicacion que se le da a la dll en la elaboracion de un programa que permita la enseñanza del lenguaje sordomudo a las personas mediante una interfaz grafica.

Consiste en que el usuario presione el teclado y se presente una animación en pantalla de la correcta transición de las posiciones que debe adoptar una mano para conseguir iconisar la tecla presionada en el lenguaje de señas sordomudo mexicano.

El menú permite seleccionar entre diferentes modelos de manos para comodidad del usuario.

5.3.2 Realidad Aumentada – Augmented Reality Demo

Figura 5.3 – Augmented Reality Demo.

Utilizando el API de ARToolKit y el dll para la carga de modelos, se creo el programa Augemented Reality Demo, este consiste en facilitar al usuario una interfase sencilla de utilizar para la correcta visualizacion de modelos tridimencionales superpuestos a imágenes reales obtenidas mediante camaras web y correctamente posicionados mediante marcadores de escena.

Mediante un archivo de configuración el usuario puede facilmente cambiar el marcador que deberá detectarse, el tamaño que este tendra, el modelo que debe mostrarse sobre este marcador, asi como su tamaño y estado.

Permite el uso de animaciones mediantes estados, por lo que también puede ser utilizado en el desarrollo propio de otras herramientas de visualización, como maquetas virtuales por ejemplo.

5.3.3 Recorridos Virtuales – Visor BTO

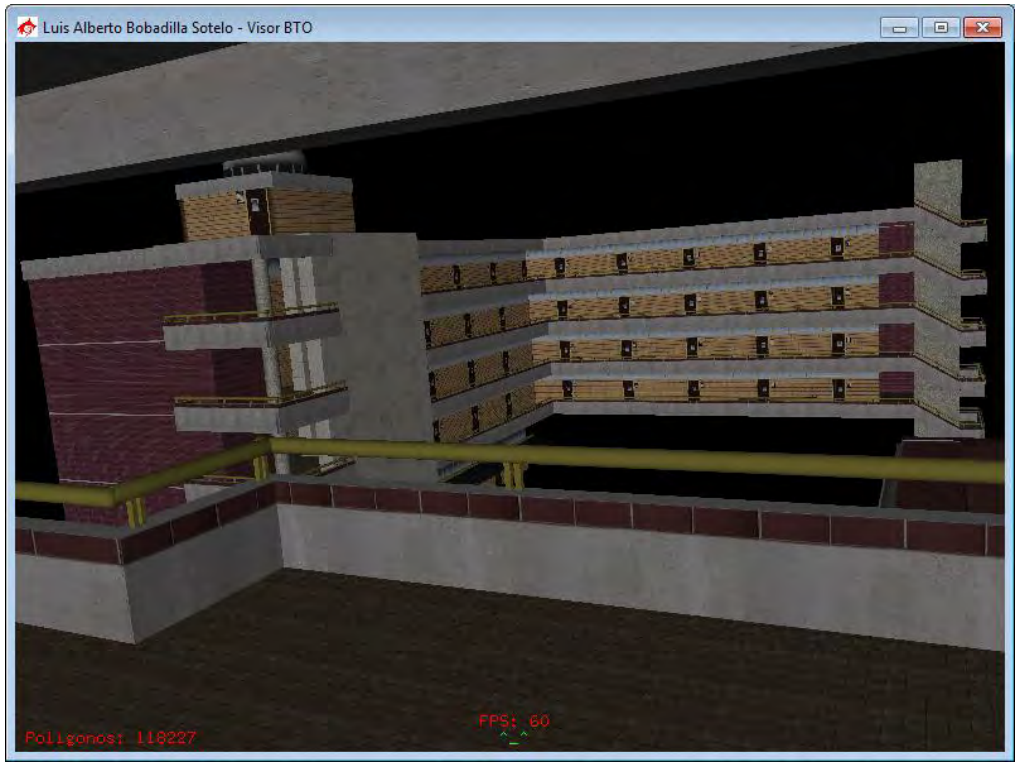


Figura 5.4 – Visor BTO.

El mismo programa Visor BTO puede ser utilizado en diferentes aplicaciones, en este caso concreto el Visor cuenta con un sistema de cámaras que permiten desplazarse en el espacio tridimensional y, mediante modelos acordes al tamaño de visualización (también configurables mediante instrucciones de menú) se puede hacer un recorrido por dichos modelos. De esta manera es posible la visualización y recorrido de maquetas virtuales.

El programador podrá, haciendo uso del sistema de colisiones propio de la dll o programando colisiones propias, hacer la visualización de los modelos de manera más interactiva y de acuerdo a sus necesidades específicas.

5.3.4 Videojuegos

Figura 5.5 – BTO VideoGame DEMO.

El dll puede ser aplicado tambien en otras areas recreativas como es el caso especifico de los juegos de video.

El programa BTO VideoGame DEMO es un ejemplo de un clasico juego tipo “sidescroll shooter” que utiliza un archivo del tipo WRL para cargar modelos en memoria y calcular las colisiones dentro del juego. Asi mismo puede ser utilizado en un sin fín de tipos de juegos de videos, que van desde las plataformas hasta los juegos de roll.

Conclusiones

Como ya se analizó en el capítulo 4, el formato BTO claramente ofrece un buen número de ventajas sobre los formatos 3DS y OBJ, como lo son la inclusión de animaciones en el caso de los archivos OBJ o la máquina de estados nativa en contraposición a ambos formatos. De igual manera ofrece ventajas sobre formatos no analizados en la presente tesis, como lo son modelos realizados en Milkshake o .mesh, que solo almacenan información referente a los vértices, polígonos y materiales de un modelo y no se preocupan por hacerlo interactivo o de fácil aplicación en programas reales. De igual forma permite ser utilizado desde materias básicas de la carrera como la enseñanza de *Estructuras de Datos* hasta casos más específicos para materias avanzadas del módulo de computación gráfica como lo son en el caso *Diseño de Interfaces Multimedia y Realidad Virtual* la enseñanza de carga de modelos tridimensionales o en el caso de *Computación Gráfica Avanzada* colisiones entre modelos y sus implementación en aplicaciones reales.

Por otro lado la lectura sencilla del archivo que contiene un modelo BTO, da la libertad al programador más avanzado a implementar su propio cargador de modelos, apoyándose o no, en las funciones incluidas en la dll, pudiendo ya sea, ignorarla por completo o utilizar las rutinas y funciones que en ella se incluyen para desarrollar nuevas.

Así mismo la implementación del dll ayuda al manejo, carga y edición de modelos reduciendo el tiempo de programación de aplicaciones de una manera drástica de un par de semanas, a un par de días, e incluso se aventura en el campo de aplicaciones

comerciales, siendo que permite la edición de un modelo en tiempo real, sin verse en la necesidad de editarlo en dichas aplicaciones, pudiendo modificar la apariencia del modelo mediante el cambio de texturas, las proporciones en las que se presenta o inclusive, en la modificación de partes del modelo o combinación de varios modelos para formar uno nuevo. Adicional a esto las aplicaciones comerciales no contemplan el manejo de la máquina de estados, lo que da un valor agregado a las funciones incluidas en la dll.

Otra clara ventaja es que el formato no presenta el problema de sobre información que los formatos utilizados en software de diseño presenta, haciéndolos en gran medida más robustos y de fácil distribución.

Para trabajo a futuro, adicional a todas las funciones y rutinas incluidas en la dll y gracias a la estructura modular que el formato posee, puede expandirse e implementarse en los siguientes puntos:

- Un sistema de animación por esqueletos, mediante el cual las animaciones de modelos orgánicos dan un toque más realista a lo presentado en pantalla sin recurrir a arreglos específicos del modelo.

- Un sistema de partículas que permita la integración de efectos visuales de bajo costo pero que aumente el nivel visual de los modelos. Dicho sistema de partículas podría ser editado directamente con funciones de la dll.

- Un sistema de shaders que permita dar efectos prácticos a los modelos tridimensionales presentados.

- Un sistema de mapas de altura embebido en el formato que permita generar terrenos a partir de una imagen. Mediante este sistema se generarán mapas que podrán ser incluidos en un modelo de escenario sin el gasto de información al detallar vértices y caras.

Bibliografía

1. FOLEY. James D.: DAM VAN. Andries: FEINER. StevenK.: HUGHES. John F. Computer Graphics: Principles and Practice in C (2nd Edition) USA Portland, Addison-Wesley Pub Co. 1995.
2. WATT. Alan H. 3D Computer Graphics 3rd edition Wokingham. England, Addison Wesley. 200
3. ANGEL. Edward. Interactive Computer Graphics: A Top-Down Approach with OpenGL 3a. edición, Boston Addison-Wesley. 2002
4. WATT. Alan: POLICARPIO. Fabio. 3D Games Vol. 2: Animation and Advanced Real-Time Rendering USA. Addison-Wesley. 2003
5. AHEAM, Luke.: 3D Game Creation Hingham, Massachussets: Charles River Media 2001
6. Martin van Velsen, 3D-Studio File Format (.3ds), January 1997.

Mesografía

7. http://artur-racp.blogspot.com/ Arturo Chan P. 2010,
8. http://arstechnica.com/old/content/2005/05/gui.ars Jeremy Reimer 2010
9. http://www.mitecnologico.com/Main/ComputacionGrafica Lauro Soto, Ensenada, BC, México 2010,
10. http://www.seccperu.org/files/herramientas3d_cine.pdf David Wong Aitken y Jorge Alvarado Valderrama, 2006
11. http://www.fileformat.info/format/wavefrontobj/egff.htm
12. www.blender.org Blender Foundation,
13. http://www.gamasutra.com/view/feature/3190/advanced_collision_detection_.php