Capítulo 1. Introducción y visión general.

Introducción.

Desde hace poco más de una décadas, los gráficos tridimensionales han cobrado gran popularidad en el área de la informática. Inicialmente debido al gran costo computacional requerido por los procesos necesarios para obtener gráficos realistas, se requería hardware especial costeable únicamente por los gobiernos, universidades y grandes compañías. Con la llegada de las computadoras personales y con el paso del tiempo se suscitó una evolución de sus microprocesadores hasta que estos tuvieron poder suficiente para realizar un procesamiento básico de gráficos 3D. El incremento en la demanda de aplicaciones 3D (en gran parte videojuegos) en el mercado de las computadoras personales, trajo consigo el desarrollo de hardware especial para realizar los cálculos relacionados con los gráficos 3D que anteriormente eran delegados al procesador central. Una menor carga de trabajo para el procesador central se traduce en una mayor disponibilidad de este para realizar otro tipo de cálculos para obtener gráficos cuyo comportamiento sea físicamente realista. Hoy en día las tarjetas aceleradoras de gráficos 3D son un componente que existe prácticamente en cualquier computadora personal en el mercado actual. Gracias a esto, a los lenguajes de programación de alto nivel, a la disponibilidad de herramientas de desarrollo gratuitas y al desarrollo de API’s gráficos tales como OpenGL y DirectX, hoy en día es completamente accesible al programador el desarrollo de aplicaciones 3D.

Definición del problema.

Hemos mencionado que actualmente es posible el desarrollo de aplicaciones 3D en computadoras personales mediante el uso de API's gráficos como OpenGL y DirectX los cuales sirven como puente de comunicación entre el código del programador y el hardware gráfico. A pesar de que este tipo de API's facilitan mucho el desarrollo de aplicaciones 3D, sigue siendo compleja la creación de este tipo de software. Estos API's facilitan el desplegado de la información geométrica al comunicarse con el hardware

gráfico para llevar a cabo los pasos necesarios para transformar esta información en una imagen bidimensional que pueda ser desplegada en pantalla. Incluso constantemente nos proporcionan nuevas funciones que mejoran la apariencia gráfica pero desafortunadamente existen tareas específicas que están fuera del alcance de un API gráfico para lo cual el programador debe de idear, investigar y programar mecanismos para realizar tareas tales como la administración de una escena, manejo de la física de los objetos de la escena y el uso de la inteligencia artificial. Este proceso limita el desarrollo de aplicaciones 3D a un menor grupo de programadores dotados con la experiencia y capacidad suficiente para elaborar y programar los algoritmos requeridos. Incluso para los programadores más experimentados el desarrollo de estos algoritmos consume tiempo valioso en el desarrollo del software.

Objetivos generales.

Se desarrollará un motor de gráficos 3D que se comunique directamente con el API gráfico OpenGL para facilitar a los programadores la elaboración de mecanismos para la visualización de gráficos tridimensionales orientado principalmente a escenas en exteriores. El motor gráfico consistirá en una librería en lenguaje C++ que podrán ser implementada en el código del programador final para incluir recorridos de escenas 3D en su aplicación con un grado de realismo aceptable en un corto lapso de tiempo.

Objetivos específicos.

- Desarrollar mecanismos de fácil implementación para tareas que requieren de un mayor entendimiento sobre conocimientos específicos y de una gran cantidad de código.
- Proporcionar una estructura clara y entendible para el programador final sobre las funciones de la librería a las cuales puede acceder.
- Una vez instalada la librería, garantizar su línea de aprendizaje total en menos de un día.
- Evitarle al programador final la necesidad de utilizar directamente el API OpenGL.
- Brindar una fluidez y rendimiento gráfico aceptable.

Alcances.

- Diseñar una arquitectura básica para el desarrollo de un motor gráfico.
- Utilizar una técnica eficiente para la administración de escenas.
- Manejo de la cámara en tiempo real.
- Implementar la detección de colisiones.
- Utilizar técnicas que mejoren el rendimiento y la calidad visual.
- Importación de modelos y escenas creados en algún paquete de modelado en 3D.
- Uso de código estándar que facilite la portación futura a otras plataformas.

Limitaciones.

- Para verificar la compatibilidad del software se cuenta con un número reducido de equipos de cómputo.
- La cantidad máxima de triángulos a procesar de manera eficiente está limitada por el poder de procesamiento del hardware a utilizar (CPU, GPU y cantidad de memoria).
- Su portación futura a otras plataformas está limitada a aquellas para las cuales exista una implementación de las librerías multiplataforma SDL y GLEW así como también del compilador GCC.

Software utilizado.

- MinGW : Compilador de C++ estándar.
- SDL : Librería para C++ utilizado para el manejo de eventos de los dispositivos de entrada y las propiedades de video.
- GLEW: Librería para C++ utilizada para facilitar la activación de las extensiones de OpenGL.
- CodeBlocks : Ambiente de desarrollo para programar en lenguaje C++.
- Milkshape 3D : Paquete de modelado en 3D.
- Biturn : Conversor de archivos 3D.

- PolyTrans : Conversor de archivos 3D.
- PaintShop Pro : Editor de bitmaps.
- T.ED : Editor de escenarios en 3D.
- Sistema operative Windows XP Second Edition.

Hardware utilizado.

- Procesador Pentium 4 HT a 3GHz.
- 1 GB de memoria RAM.
- Tarjeta de video ATI Radeon X600 con 128MB de memoria RAM.

Trabajo relacionado.

Actualmente existen librerías para el lenguaje de programación C++ cuyos objetivos y características son similares a los de este proyecto y que en general se enfocan solo a representar y recorrer escenas 3D. A continuación presentaremos algunas de estas herramientas.

Expresión 3D.

Sitio web: http://www.kotterink.com/expression/

El desarrollador la define como un conjunto de herramientas para la visualización de objetos 3D.

- Utiliza OpenGL como API 3D.
- Desarrollado para las plataformas Windows(utilizando el API Win32) y Linux(utilizando el API X11).
- Es de libre distribución.
- Puede importar los siguientes formatos de archivos 3D: 3DS.
- Puede importar los siguientes formatos de escenas 3D: WMF.

- Puede importar los siguientes formatos de mapas de bits: JPG.
- Técnicas utilizadas para la administración y/o optimización de la escena: Asigna un objeto envolvente a cada modelo 3D para evaluar su visibilidad dentro del "view frustum" ("view frustum culling").
- Implementa técnicas de detección de colisiones.
- Características adicionales: Puede desplegar un "sky box" dado, incluye su propio editor de escenas, implementa un sistema de partículas, utiliza quaterniones para la rotación de la cámara y de modelos.

libQGLViewer.

Sitio web: http://artis.imag.fr/Members/Gilles.Debunne/QGLViewer/index.html

El desarrollador la define como una librería que provee funcionalidad básica para visualización 3D.

Movimiento de cámara, selección de objetos, stereo display (lentes 3D).

- Utiliza OpenGL como API 3D.
- Utiliza las librerías Qt (desarrolladas por Trolltech), para asegurar su portación a las plataformas Windows, Mac y Linux.
- Es de libre distribución pero las librerías Qt son comerciales y se requieren para su compilación.
- No implementa mecanismos para importar archivos de modelos y escenas 3D (debe programarse o acudir a una librería externa).
- No implementa mecanismos para importar archivos con mapas de bits (debe programarse o acudir a una librería externa).
- Técnicas utilizadas para la administración y/o optimización de la escena: Asigna un objeto envolvente a cada modelo 3D para evaluar su visibilidad dentro del "view frustum" ("view frustum culling"). Implementa técnicas de detección de colisiones.
- Características adicionales: Animación de modelos 3d mediante "*keyframes"*, selección de objetos 3D para lanzar algún evento, soporte para utilizar lentes stereo, utiliza quaterniones para la rotación de la cámara y de modelos.

IvF++.

Sitio web: http://ivfplusplus.sourceforge.net/

El desarrollador la define como una librería cuyo objeto es facilitar el uso de OpenGL para desarrollar aplicaciones interactivas.

- Utiliza OpenGL como API 3D.
- Utiliza las librerías FLTK (Fast Light Toolkit), para asegurar su portación a las plataformas Windows, Mac y Linux.
- Es de libre distribución.
- Puede importar los siguientes formatos de archivos 3D: AC3D, DFX.
- No importa archivos con información de escenas 3D.
- Puede importar los siguientes formatos de mapas de bits: JPG, PNG.
- Técnicas utilizadas para la administración y/o optimización de la escena: Asigna un objeto envolvente a cada modelo 3D para evaluar su visibilidad dentro del "view frustum" ("view frustum culling"). Utiliza una técnica para controlar el nivel de detalle (LOD) de los objetos 3D.
- No implementa técnicas de detección de colisiones.
- Características adicionales: Animación de modelos 3d mediante "*keyframes"*, selección de objetos 3D para lanzar algún evento, utiliza quaterniones para la rotación de la cámara y de modelos.

SGL.

Sitio web: http://sgl.sourceforge.net

El desarrollador la define cómo una librería multiplataforma que implementar funciones relacionadas con la construcción y representación de escenas 3D.

- Utiliza OpenGL como API 3D.
- Desarrollado par alas plataformas Windows, Linux, Solaris, Iris y Mac OS X. Utiliza el API nativo de cada sistema operativo para su portación.

- Es de libre distribución.
- Puede importar los siguientes formatos de archivos 3D: OBJ, FLT, TXP,VRML.
- No importa archivos con información de escenas 3D.
- Puede importar los siguientes formatos de mapas de bits: BMP, JPG, PNG, TGA, TIF, SGI.
- Técnicas utilizadas para la administración y/o optimización de la escena: Asigna un objeto envolvente a cada modelo 3D para evaluar su visibilidad dentro del "view frustum" ("view frustum culling").Utiliza técnicas para controlar el nivel de detalle (LOD) de los objetos 3D.
- No implementa técnicas de detección de colisiones.
- Características adicionales: Animación de modelos 3d mediante "*keyframes"*, proyección de sombras utilizando "shadow maps", utiliza quaterniones para la rotación de la cámara y de modelos.

A continuación presentaremos las características del software desarrollado.

- Utiliza OpenGL como API 3D.
- Utiliza las librerías SDL(Simple Direct media Layer), para asegurar su portación a las plataformas Windows, Mac y Linux aunque actualmente solo se a comprobado su correcta compilación bajo Windows.
- Es propiedad de la UDLA.
- Puede importar los siguientes formatos de archivos 3D: OBJ y de forma limitada MS3D.
- Puede importar los siguientes formatos de escenas 3D: EPR (Editor T.ED).
- Puede importar los siguientes formatos de mapas de bits: BMP, TGA, PCX.
- Técnicas utilizadas para la administración y/o optimización de la escena: Se utilizan "octrees" para evaluar la visibilidad de cada región de la escena dentro del "view frustum" ("view frustum culling"). Se utiliza un algoritmo para evitar dibujar más de una vez una cara contenida dentro de más de un octante. Se selecciona el modo de dibujo más rápido para el caso particular de cada octante (solo para las caras no compartidas).
- Detección de colisiones utilizando octrees.

- Características adicionales: Puede desplegar un “sky box” dado. Puede importar un terreno generado y seccionado por el editor T.ED. Utiliza quaterniones para la rotación de la cámara y de modelos. Suavizado de la rotación de cámara.

A continuación se presenta un cuadro en donde podemos comparar las características del software desarrollado con las herramientas anteriormente mencionadas:

	Expresión 3D	**libQGL Viewer**	**IvF++**	**SGL**	**P. tesis (Mira3D)**
API 3D	OpenGL	OpenGL	OpenGL	OpenGL	OpenGL
Plataforma	Windows, Linux	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows y teóricamente Linux y MacOS
Licencia	Libre distr.	Libre distr.	Libre distr.	Libre distr.	UDLA
Formatos de modelos 3D aceptados	3DS	---	AC3D, DFX	OBJ, FLT, TXP, VRML	OBJ, MS3D(uso limitado)
Formatos de escenas 3D aceptados	WMF	---	---	---	EPR
Formatos de mapas de bits aceptados	JPG	---	JPG, PNG	BMP, JPG, PNG, TGA, TIF, SGI	BMP, TGA, PCX
Técnicas de admón. y/o optimización de escena	VFC en base a objetos env.	VFC en base a objetos env.	-VFC en base a objetos env. - LOD.	-VFC en base a objetos env. - LOD.	-VFC utilizando octrees: -Alg. caras repetidas. -Draw mode auto ajustable.
Detección de colisiones	SI	SI	NO	NO	SI
Caract. adicionales	-Sky box. -Editor de escenas. -Sistema de partículas. -Quaterniones	-Animación de Keyframes -Selección de objetos 3D. -Lentes stereo -Quaterniones	-Animación de Keyframes -Selección de objetos 3D. -Quaterniones	-Animación de Keyframes -Shadow maps. -Quaterniones	-Sky box. -Importación de terrenos seccionados en varios archivos (T.ED). -Suavizado de la rot.ación de cámara. -Quaterniones

Comparativamente hablando se puede apreciar que existen características adicionales que el software desarrollado no implementa aunque también se han implementado características que las demás herramientas no incorporan. A pesar de esto, llena todas las características básicas. Posiblemente lo más destacado sea el uso de técnicas de administración y/o optimización basadas en “octrees” y que en las herramientas comparadas no son utilizadas. Se piensa que esto podría marcar una diferencia importante en el rendimiento general del software.