Universidad de El Salvador
Facultad Multidisciplinaria Oriental
Dept. de Ingeniería y Arquitectura

Asignatura:

Algoritmos Graficos

Docente:

Ing. Ludwin Hernandez

Integrantes del grupo:

Luis Fernando Hernandez Castillo

Maria Yanci Martinez Garcia

Duglas Enrique Díaz Barahona

Año: 2017

Desarrollo de un Videojuego de Naves Espaciales 3D versión 0.0.1 con OpenGL y C++

INDICE

INTRODUCCIÓN

El uso de los entornos gráficos para la realización de animaciones, aplicaciones o juegos se ha hecho mucho más habitual en estos tiempos, con el paso de los años nuevos motores de desarrollo han sacado a relucir todo su potencial, con juegos y animaciones más realistas, haciendo posible de un ambiente mas llamativo que las herramientas antes utilizadas, anteriormente con el uso de OpenGL hacer estas actividades era mucho más lento y menos factible, pero gracias a muchas actualizaciones y desarrolladores se ha hecho la inclusión de muchas librerías y funcionalidades que lo han convertido en una de las mejores herramientas para comenzar a realizar estas actividades desde cero sin uso de renderizado.

Es por eso que vamos a utilizar estas herramientas que OpenGL nos proporciona para el desarrollo del video juego de naves espaciales 3D versión 0.0.1, tomando en cuenta el uso de sus librerías, las cuales serán seleccionadas de acuerdo a las necesidades que se tengan en el transcurso de la realización de este, así como se toma en cuenta que este juego tendrá sus respectivas limitaciones las cuales serán detalladas mas adelante en este documento, cabe mencionar que el lenguaje de programación sera C++ con el que se pretende desarrollar toda la lógica, para la efectiva realización de los procesos que el juego conlleva.

1. EL PROBLEMA.

1.1 TITULO DESCRIPTIVO DEL PROYECTO

Desarrollo de un Juego de Naves espaciales 3D, Versión 0.0.1

1.2 SITUACIÓN PROBLEMÁTICA

Con el uso de OpenGL y el lenguaje C++ se pretende la realización de un juego tridimensional de una nave espacial que va disparando a platillos voladores que aparecen en el escenario teniendo en cuenta que debe esquivarlos para evitar una coalición entre la nave y el enemigo, haciendo uso del teclado para interactuar con ella.

Con lo que se pretende hacer uso de las librerías básicas de OpenGL para lograr que el juego sea mas agradable a la vista del usuario, además hacer uso de texturas, materiales e iluminación.

1.3 PLANTEAMIENTO DEL PROBLEMA.

Con OpenGL se puede desarrollar gráficos para la elaboración de un videojuego ya sea en 2D o 3D y con los conocimientos aprendidos en el curso de la asignatura de algoritmos gráficos se plantea la realización del siguiente proyecto que consiste en la elaboración de un videojuego de una nave espacial que pretende esquivar y disparar a sus enemigos representados por platillos voladores con el fin de destruirlos.

1.4 ENUNCIADO DEL PROBLEMA.

¿Sera posible el desarrollo de un videojuego de naves espaciales usando OpenGL y C++?

1.5 JUSTIFICACIÓN

El presente proyecto de investigación permitirá ampliar nuestros conocimientos en el desarrollo e implementación de gráficos 2D y 3D, con las librerías que OpenGL nos facilita, utilizando los algoritmos y técnicas aprendidos en el aula. Lo que se pretende es construir una aplicación gráfica animada en 3D y el desarrollo de este proyecto nos proporcionara mayor conocimiento de esta herramienta tan esencial para la realización de este videojuego, ademas aportando al usuario final un medio de entretenimieto, asi como tambien servira como una herramienta que fortalece el aprendizaje en sus usuarios desarrollando su logica y una mejor velocidad en sus reflejos.

1.6 DELIMITACIONES.

Este proyecto de investigación es realizado por los estudiantes de la carrera de Ingeniería de Sistemas de la Facultad Multidisciplinaria Oriental de la Ciudad de San Miguel, para ser presentado en la materia de algoritmos gráficos y mostrar los conocimientos adquiridos en el uso de OpenGL durante el fin de ciclo, en el proyecto no se pretende hacer un uso de librerías y herramientas que no fueron proyectadas en las horas de clases.

El videojuego esta delimitado de la siguiente manera:

El videojuego en cuestion sera desarrollado con OpenGL y C++, el cual sera visualizado en un entorno representado por una ciudad invadida por naves espaciales, donde los elementos con los que consta el videojuego son: platillos voladores, una nave espacial y su entorno se tienen edificios arboles, un fondo con texturas nocturnas y el suelo con textura de concreto y pasto.

Las animaciones e interacciones con las que constan los elementos son:

Simulan rotacion para parecer mas reales mientras se mueven sobre la pantalla.

Platillos voladores (enemigos): Se mueven por la pantalla de la siguiente manera entorno al eje x con una limitante de 100 puntos regresa al inicio y asi continua repitiendose los movimientos mientras esta no sea impactada por un disparo o se impacte con la nave del jugador.

Nave principal: Esta tiene las opciones de desplazarse con las sigientes teclas sea mayusculas o minusculas, la tecla “W” para despazarse hacia arriba, la tecla “S” para desplazarse hacia abajo, la tecla “A” para desplazarse hacia la izquierda, la tecla ”D” se desplaza hacia la derecha,la tecla “E” dispara, la tecla “X” acercamiento, la tecla “Z” para alejar, y la tecla “ barra espaciadora” inicia y pausa el juego, todas estas acciones seran controladas por el usuario para moverse a su conveniencia, la nave principal colisiona cuando una nave enemiga choca contra esta, ataca a la naves enemigas lanzandole rayos para derrivarlas.

Todo sera específicamente controlado por teclado normal y solo sera una demostración, ya que no pasara de ningún nivel se basara prácticamente de una área limitada que tiene 100 puntos en el eje X y en el eje Y 30 puntos solo se implementara las herramientas y recursos que se impartieron en clases de algoritmos gráficos.

1.6.1 LUGAR O ESPACIO

Universidad de El Salvador Facultad Multidisciplinaria Oriental, San Miguel.

1.6.2 TIEMPO

El tiempo que se invirtió para el desarrollo del presente proyecto fue de:

230 horas a lo largo de un mes y medio.

1.6.3 TEORÍAS

Como es de saber y como se ha venido mencionando a lo largo de este documento, nuestras herramientas de trabajo han sido básicamente OpenGL, así que a continuación se mencionan alguna de las librerías que nos han sido muy útiles en este trabajo.

Librería gl: Proporciona todo lo necesario para acceder a las funciones de dibujado de OpenGL.

La librería GLU (OpenGL Utility Library): Una librería de utilidades que proporciona acceso rápido a algunas de las funciones más comunes de OpenGL, a través de la ejecución de comandos de más bajo nivel, pertenecientes a la librería OpenGL propiamente dicha.

GLX (OpenGL Extensión to the X Window System): Proporciona un acceso a OpenGL para poder interactuar con un sistema de ventanas X Window, y está incluido en la propia implementación de OpenGL (su equivalente en Windows es la librería WGL, externa a la implementación de OpenGL).

Libreria SOIL (Simple OpenGL Image Library): Es una pequeña librería para cargar imágenes en 6 tipos de formatos.

Es un cargador de imágenes multiplataforma de dominio público que es extremadamente pequeño. Puede leer una pequeña variedad de formatos útiles (BMP, JPEG, PNG, varios archivos DDS, etc).

Tiene funciones para cargar imágenes en texturas OpenGL. De hecho, sólo permite esto (que es por qué es "simple").
Es software de dominio público.

1.7 OBJETIVOS DEL PROYECTO

1.7.1 OBJETIVO GENERAL.

Desarrollar el videojuego de naves espaciales con el uso de OpenGL y C++.

1.7.2 OBJETIVOS ESPECIFICOS

- Realizar una investigacion acerca de los componentes necesarios en el desarrollo de un videojuego.

- Desarrollar el videojuego mediante el uso de algunos algoritmos ya conocidos, dandole un entorno aceptable al publico.

- Llevar a cabo la implementacion del videojuego para su uso final.

- Ejecutar las pruebas pertinentes para comprobar que todo funciona como se espera.

2. FUNDAMENTACIÓN TEÓRICA

A continuación se presentan las bases teóricas asi como conceptos de gran importancia en las cuales se apoya la implementacion del proyecto de investigación.

GRAFICACION POR COMPUTADORA

La graficación por computadora es la creación, almacenamiento, manipulación y despliegue de imágenes con la asistencia de una computadora. La computación gráfica o gráficos por ordenador es el campo de la informática visual, donde se utilizan computadoras tanto para generar imágenes visuales sintéticamente como integrar o cambiar la información visual y espacial probada del mundo real. El primer mayor avance en la gráfica realizada por computadora era el desarrollo de Sketchpad en 1962 por Iván Sutherland.

OpenGL

Open Graphics **L**ibrary es una especificación estándar que define una API multi-lenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos. Fue desarrollada originalmente por Silicon Graphics Inc. (**SGI**) en 19922 y se usa ampliamente en CAD[1], realidad virtual, representación científica, visualización de información y simulación de vuelo. También se usa en desarrollo de videojuegos, donde compite con Direct3D[2] en plataformas Microsoft Windows.

[1]CADD (computer-aided design and drafting), que significan «bosquejo y diseño asistido por computadora».

[2] **Direct3D** es parte de **DirectX**(conjunto de bibliotecas para multimedia), propiedad de Microsoft.

Fundamentalmente OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener.

Partiendo de ella, los fabricantes de hardware crean implementaciones, que son bibliotecas de funciones que se ajustan a los requisitos de la especificación, utilizando aceleración hardware cuando es posible. Dichas implementaciones deben superar unos tests de conformidad para que sus fabricantes puedan calificar su implementación como conforme a OpenGL y para poder usar el logotipo oficial de OpenGL.

Hay implementaciones eficientes de OpenGL para Mac OS, Microsoft Windows, GNU/Linux, varias plataformas Unix y PlayStation 3. Existen también varias implementaciones en software que permiten ejecutar aplicaciones que dependen de OpenGL sin soporte de aceleración hardware. Es destacable la biblioteca de software libre / código abierto Mesa 3D, una API[1] de gráficos sin aceleración hardware y completamente compatible con OpenGL. Sin embargo, para evitar los costes de la licencia requerida para ser denominada formalmente como una implementación de OpenGL, afirma ser simplemente una API muy similar.

[1]**API:** La interfaz de programación de aplicaciones, abreviada como API del inglés: Application Programming Interface, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

PRIMITIVAS DE DIBUJO :

En OpenGL solo se pueden dibujar primitivas muy simples, tales como puntos lineas, cuadrados, triangulos y polygonos, a partir de estas primitivas es posible construir primitivas mas complejas como arcos y circulos aproximandolos por poligonos.
Toda primitiva de dibujo se construye con un par:
glBegin(tipo_de_primitiva); glVertex2f(); ... glEnd();
Donde tipo_de_primitiva puede ser los siguientes:
GL_POINTS: Cada vertice es un punto
GL_LINES: Cada par de vertices sucesivos es una linea
GL_LINE_STRIP: lineas conectadas.
GL_LINE_LOOP: lineas conectadas donde el ultimo y el primer vertice indican una linea cerrando el poligono.
GL_POLYGON: poligono (relleno o no) donde los vertices sucesivos componiendolo se dan el sentido contrario de las manecillas del reloj.
GL_QUADS: cuadrilateros separados, cada 4 vertices hacen un quad.
GL_QUAD_STRIP: tira de cuadrados unidos, cada par de vertices sucesivos forman un cuadrado con el par anterior.
GL_TRIANGLES: Triangulos separados, cada 3 vertices hacen un triangulo.
GL_TRIANGLE_STRIP: tira de triangulos unidos (similara quad_strip).
GL_TRIANGLE_FAN: Grupo de triangulos con un unico vertice comun a todo.

LIBRERIA GLUT

GLUT Mecanismos (del inglés *Open**GL** **U**tility **T**oolkit*) es una biblioteca de utilidades para programas OpenGL que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo. Entre las funciones que ofrece se incluyen declaración y manejo de ventanas y la interacción por medio de teclado y ratón.

A continuación se listan las funciones que se han implementado para el dibujo de primitivas con la librería glut.

glutSolidSphere (GLdouble radius);

glutWireCube (GLdouble size);

glutWireTorus (GLdouble innerRadius, GLdouble outerRadius);

glutWireCone (GLdouble radius, GLdouble heigth);

COLORES

OpenGL puede utilizar dos modos de color: color RGBA y color indexado. Nosotros en el desarrollo de este juego nos vamos a centrar en el color RGBA para los elementos que se tendran en pantalla. Este color recibe este nombre porque se compone de cuatro componentes: Rojo(Red), Verde(Green), Azul(Blue), y canal Alfa, o transoparencia.

IMPLEMENTACION DE MATERIALES:

un material define, para un determinado objeto, que componentes refleja cada tipo de luz.

Por ejemplo, un plastico rojo, emitira toda la componente roja de las lices ambiental y difusa, pero generalmente emitira brillos de color blanco o bajo una luz blanca, por lo que su componente especular sera de color blanco. De este modom la componente de emision sera negra, puesto que un color plastico no emite luz.

En un modelo con iluminacion, el color de un objeto se define por su material, no por el color de sus vertices.

Para implementar un materia, estableceremos los valores de las diversas características del mismo, mediante las funciones del tipo glMaterial*(). Estas funciones reciben tres parámetros:

1.Caras del polígono a las que se aplica el material. Podemos aplicar el material a la cara frontal (GL_FRONT) o a ambas caras del polígono (GL_FRONT_AND_BACK).

2. Caracteristica que definimos, mediante su correspondiente constante.

3. Valor de la caracteristica.

Las diferentes caracteristicas son:

-color del material:define el comportamiento del material para los distintos tipos de luz descritos. El color ambiental, difuso y especular definen los colores del espectro que este material refleja de las luces que recibe. El color emitido es el tipo de luminosidad que posee el material, independientemente de la luz que le afecte.

Se define mediante la llamada:

glMaterialfv(GL_FRONT[_AND_BACK],[GL_AMBIENT|GL_DIFUSSE|

GL_AMBIENT_AND_DIFUSSE|GL_SPECULAR],val_ptr);

“val_ptr” es un puntero a un conjunto de valores RGBA que especifica el color de la componente especifica.

Brillo de los reflejos:

La intensidad del brillo puede definirse a traves de la llamada:
glMaterialf(GL_FRONT[_AND_BACK],GL_SHININESS,val);

Ademas, podemos cambiar alguno de los valores establecidos para los colores de un material mediante la funcion glColorMateria(). Mediante esta funcion, que debemos activas, podemos cambiar alguno de los colores del material.

Esta opcion se activa mediante:

glEnable(GL_COLOR_MATERIAL);

y una vez activada, establecemos la propiedad que queremos cambiar mediante:

glColorMaterial(GL_FRONT[_AND_BACK],[GL_AMBIENT|GL_DIFUSSE|

GL_AMBIENT_AND_DIFUSSE|GL_SPECULAR]);
Y cambiamos el color reflejado de este tipo de luz mediante:

glColor3f(r_val,g_val,b_val);

APLICACION DE LUZ A LA ESCENA:

Una luz aporta iluminación a una escena, según unas determinadas componentes de color.

Se entiende por fuente de luz, como entidad que proporciona luz a una escena, y luz, como aportación de esa fuente a la iluminación de la escena.

Una fuente puede emitir tipos diferentes de luz, que son complementarias, y pueden darse a la vez para un cierto tipo de luz.

Los tipos de luz son:

“Emitted” (emitida): es la luz emitida por un objeto. No se ve afectada por ningún otro tipo de luz. Por ejemplo, un fuego emite una determinada luz, y si lo miramos, lo veremos de ese color, independientemente del color de las luces que estén apuntando al fuego.

“Diffuse” (difusa): es la luz que indice sobre un objeto, y proviene de un determinado punto. La intensidad con la que se refleja en la superficie del objeto puede depender del ángulo de incidencia, dirección, etc. Una vez incide sobre un objeto se refleja en todas direcciones.

“Specular” (especular): es la luz que, al incidir sobre un objeto, se ve reflejada con un ángulo similar al de incidencia. Podemos pensar que es la luz que produce los brillos.

“Ambient” (ambiental): podemos considerarlo como los restos de luz que aparecen defido a la reflexión residual de una luz que ya se ha reflejado sobre muchos objetos, y es imposible determinar su procedencia. Es algo así como la iluminación global de una escena.

APLICACION DE LAS TEXTURAS:

Las texturas permiten personalizar aún más el material de un objeto, y nos permiten obtener imágenes mucho más realistas en nuestras escenas. Por ejemplo, si queremos dibujar una pared de ladrillo, tenemos dos opciones: dibujar cada ladrillo, definiendo su material con colores, y dibujar el cemento entre los ladrillos, a base de polígonos, definiendo también su material, o dibujar un único cuadrado con la extensión de la pared, y hacer que su material.

Aplicar las texturas

Para aplicar una textura tenemos que seguir una serie de pasos muy definidos:

1. Creamos la textura.

2. Definimos las condiciones en que se va a aplicar la textura.

3. Habilitar la aplicación de texturas.

4. Dibujar las escenas, proporcionando las coordenadas de textura.

3. ASPECTOS ADMINISTRATIVOS.

3.1 RECURSOS HUMANOS.

Para la elaboración del presente proyecto se debe constar con e personal humano que es parte fundamental para la realización del video juego el cual se clasifica de la siguiente manera:

Diseñadores gráficos:

María Yanci Martínez García

Duglas Enrique Diaz Barahona

Desarrollador:

Luis Fernando Hernández Castillo

3.2 PRESUPUESTO

En el siguiente apartado se dará a conocer el presupuesto del proyecto.

En él se detallaran los gastos de personal generado durante el desarrollo del presente proyecto.

Las horas invertidas al total del proyecto son 230 horas aproximadamente que provienen de los 46 días que se mostraran en la planificación al final realizando una jornada de 5 horas.

Que se descomponen de la siguiente manera:

Actividades	**Días**	**Horas**	**Total de horas**	**Precio por hora en $**	**Precio total por actividad**
Búsqueda y análisis del proyecto a realizar	5	5	25	5	125
Investigación para la realización del video juego	4	5	20	5	100
Desarrollo de la documentación	2	10	10	5	50
Diseño y planeación de la interfaz del video juego	5	5	25	5	125
Elaboración del video juego	30	5	150	5	750
Total	46	30	230	25	1150

3.3 CRONOGRAMA

Actividades diarias realizas en la elaboración del proyecto de video juego 3D versión 0.0.1 con inicio en el mes de mayo en adelante.

Actividades	1° Semana	2° Semana	3° Semana	4° Semana	5° Semana	6° semana	7° Semana
Búsqueda y análisis del proyecto.	X						
Investigación para la realización del videojuego		x					
Planteamiento al docente y aprobación			x				
Desarrollo de la documentación				x			
Entrega del avance de la documentación					x		
Diseño y planeación de la interfaz del video juego			x	x			
Elaboración del video juego			x	x	x	x	
Entrega del proyecto completo							x

4. REFERENCIAS

(Addison-Wesley Publishing Company) *OpenGL Programming Guide*

https://www.khronos.org/opengl/