ALGORITMOS GRÁFICOS
2017

Desarrollo de un mini juego

llamado Motor Racing en OpenGL, C++, SOIL y SDL

Diego Herrera
Gerson Guerrero
Pedro Andrade
Jehudí Cruz

Desarrollo de un mini juego llamado Motor Racing en OpenGL, C++, SOIL y SDL

Presentado por:

Diego Armando Herrera
Gerson Alexander Sandoval
Pedro José Andrade
Mario Jehudí Cruz

Responsable:

Ing. Ludwin Alduvi Hernández

Asignatura:

Algoritmos Gráficos

Contenido

Introducción

En los tiempos en los que vivimos actualmente, la informática y la tecnología en general se encuentran en una etapa muy desarrollada, en la cual hemos sido testigos de grandes avances tecnológicos y hemos visto evolucionar muchas de las tecnologías que se han usado desde hace muchos años. Entre tantas de las cosas que hemos visto evolucionar, se encuentran los gráficos asistidos por computadora, los cuales en la actualidad se encuentran por todas partes.

Hoy en día no hay ningún lugar en el que no se encuentren presentes los gráficos asistidos por computadora, desde anuncios comerciales, juegos, películas, series de televisión, revistas, libros, etc. Se han convertido en parte fundamental del desarrollo de la vida misma, ya que facilitan en gran manera las actividades que se realizan en diversos campos como la arquitectura, la física, la química, la aviación, la economía, es decir, prácticamente están en todas partes y sin ellos no sería posible que la sociedad avance al ritmo que actualmente lleva, ya que facilita el sector de la construcción permitiendo que se hagan modelos y maquetas de edificaciones que aun ni se construyen, permite simular actividades como vuelos en la aviación, simular acontecimientos físicos o químicos y en fin, nos facilitan el desarrollo de la sociedad misma.

Por todo lo anterior mencionado es de vital importancia conocer los orígenes de la computación gráfica, ya que no todo era tal y como lo conocemos hoy, no existían todos esos grandes software de diseño, renders y motores gráficos que permiten hoy en día hacer maravillas con ellos; todas esas herramientas de las que hoy se disponen tienen sus orígenes, tienen su génesis que se encuentra en las primitivas, es decir esas figuras elementales como el punto, la línea, el circulo, el cuadrado y el triángulo. Por ello, en el presente proyecto que pretendemos desarrollar, se harán uso de todos aquellos elementos básicos para llevarlo a cabo, auxiliando-nos del padre de todos los softwares gráficos que hoy conocemos, nos referimos a OpenGL[1], de la mano del lenguaje de programación C++[2] y las librerías disponibles para poder desarrollar el presente proyecto.

[1] es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

[2] C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup.

1. El Problema

1.1 Título Descriptivo del Proyecto

El presente proyecto ha sido denominado:

"Desarrollo de un mini juego llamado Motor Racing en OpenGL, C++, SOIL y SDL."

Se pretende desarrollar un mini juego basado en OpenGL y controlado en el lado de la programación por C++. El mini juego tratará de un automóvil que el usuario podrá controlar mediante las teclas de desplazamiento del computador.

Motor Racing describe, evidentemente que se trata de un juego de autos, por tal razón se necesitará la librería SOIL[3] para poder añadir texturas e imágenes, así mismo la librería SDL[4] para poder insertar sonidos.

[3] Es una pequeña librería desarrollada en el lenguaje de programación C para facilitar la carga de imágenes en OpenGL
[4] es un conjunto de bibliotecas desarrolladas en el lenguaje de programación C que proporcionan funciones básicas para realizar operaciones de dibujo en dos dimensiones, gestión de efectos de sonido y música, además de carga y gestión de imágenes.

1.2 Situación Problemática

El problema con el que nos encontramos radica en que en la actualidad la mayoría de desarrolladores utiliza solamente software existente y de uso inmediato para poder desarrollar gráficos por computadora, a costa de ignorar y desconocer todos los elementos que engloba la computación gráfica, así como sus principios más elementales.

Esto causa que muchos desarrolladores de gráficos por computadora sean incapaces de comprender y entender todo lo que realmente rodea a los gráficos por computadora, y por consiguiente haciéndolos muy limitados en todos los aspectos técnicos más básicos y en el uso de las librerías disponibles para desarrollar gráficos a nivel muy bajo, pero con mucha potencia, llevando prácticamente el control de todo lo que sucede en su proyecto.

Con la creación del videojuego que proponemos, denominado **"Desarrollo de un mini juego llamado Motor Racing en OpenGL, C++, SOIL y SDL"** pretendemos que nosotros como estudiantes de Ingeniería de Sistemas Informáticos conozcamos todos los aspectos más básicos y a bajo nivel en el desarrollo de gráficos por computadora, y no nos quedemos solo como usuarios finales de software de desarrollo de gráficos, sino que podamos aprender y comprender lo mayormente posible en cuanto a los aspectos más fundamentales del desarrollo de gráficos por computador.

1.3 Planteamiento del Problema

En la informática, específicamente hablando del campo de la creación de gráficos, existe una herramienta sumamente potente, la cual es usada en todo lo referido a gráficos, desde software de terceros, juegos, renders etc. Esta poderosa herramienta es una librería de gráficos, que lleva por nombre OpenGL.

Es esta poderosa herramienta precisamente la que se pretende utilizar en el desarrollo del proyecto que denominados **"Desarrollo de un mini juego llamado Motor Racing en OpenGL, C++, SOIL y SDL"** el cual como su nombre lo describe, hará uso de más herramientas en combinación con OpenGL. Es aquí donde radica el problema del proyecto, ya que, para desarrollar un videojuego, éste requiere de texturas, imágenes, sonidos y manejo de todo ello conjuntamente, y que, solamente haciendo uso de OpenGL como tal, nos sería imposible.

Aquí es donde entra en juego la librería SOIL, la cual nos ayudará a la hora de importar y manejar imágenes y usarlas como texturas, la cual deberemos estudiar detenidamente para poder hacer un uso correcto de ella. Al mismo tiempo como ya mencionamos se requiere de sonidos y además fuentes de texto para desarrollar el videojuego, para ello recurriremos a la librería SDL para poder importar y manejar ambos aspectos dentro del juego.

Ahora la interrogante es, ¿Cómo poder manejar todo eso junto? Pues sabemos que se requieren conocimientos de programación para poder "enlazar" todas esas librerías y hacer uso conjunto de ellas, y es aquí donde proponemos usar C++ como lenguaje de desarrollo por su potencia, flexibilidad y vasta documentación y compatibilidad con todas las librerías mencionadas anteriormente.

1.4 Enunciado del Problema

¿Se puede elaborar un mini juego completamente funcional sin la necesidad de utilizar software privativo o renders profesionales?

1.5 Justificación

Con la finalidad de aplicar los conocimientos que hemos adquirido en la materia de Algoritmos Gráficos, pretendemos crear un mini-juego interactivo. Con el desarrollo del juego queremos ayudar a que las personas que lo usen puedan mejorar sus reflejos, los usuarios pondrán a prueba su capacidad de competencia ya que el juego tendrá diferentes niveles y escenarios. El mini juego será desarrollado en OpenGL, C++, SOIL y SDL. Esto nos permitirá crear todo lo necesario para nuestro juego desde cero lo cual nos permitirá crear los modelos como mejor nos parezca y que se adapte a la necesidad de lo que estamos haciendo.

1.6 Delimitaciones

1.6.1 Lugar, Espacio o Tiempo

El juego será desarrollado en la Universidad de El Salvador, Facultad Multidisciplinaria Oriental.

1.6.2 Tiempo

El tiempo total para el desarrollo del juego será aproximadamente cuatro semanas.

1.6.3 Delimitación Teórica

Se pretende crear un juego con dos distintos escenarios de los cuales cada uno de ellos tendrá diferentes niveles para que usuario pueda ir superándolos, el juego será controlado por teclas siendo estas izquierda y derecha que serían las direcciones en las cuales el usuario podrá mover el auto, durante el juego aparecerán distintos objetos que deberán ser esquivados, el juego será en 3D. El juego será desarrollado en OpenGL, C++, SOIL y SDL, haciendo uso exclusivo de herramientas de software libre.

1.7 Objetivos del Proyecto

1.7.1 Objetivo General

- Desarrollar el mini juego denominado “Motor Racing en OpenGL”.

1.7.2 Objetivos Específicos

- Planificar las actividades para identificar claramente la naturaleza y el alcance del mini juego denominado “Motor Racing”.
- Analizar los datos acerca del sistema y su funcionamiento aplicando “Las librerías como SOIL, SDL” para su completo funcionamiento, en general las técnicas de recopilación de datos específica que es lo que el mini juego debe hacer.
- Desarrollar el mini juego denominado “Motor Racing” con las herramientas vistas en el curso tales como “SOIL, SDL, OpenGL” para que el usuario pueda comprender su objetivo.
- Realizar pruebas para obtener información si en dado caso en el mini juego denominado “Motor Racing” se necesita hacerse correcciones.
- Implementar adecuadamente el mini juego “Motor Racing” en OpenGL, SOIL y SDL.

2. Fundamentación Teórica

OpenGL (Open Graphics Library) es una especificación estándar que define una API[5] multilenguaje 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 1992 y se usa ampliamente en CAD[6], realidad virtual, representación científica, visualización de información y simulación de vuelo. También se usa en desarrollo de videojuegos, donde compite con Direct3D[7] en plataformas Microsoft Windows.

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.

OpenGL tiene dos propósitos esenciales:

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

El funcionamiento básico de OpenGL consiste en aceptar primitivas tales como puntos, líneas y polígonos, y convertirlas en píxeles. Este proceso es realizado por una pipeline gráfica conocida como Máquina de estados de OpenGL 8. La mayor parte de los comandos de OpenGL bien emiten primitivas a la pipeline gráfica o bien configuran cómo la pipeline procesa dichas primitivas. Hasta la aparición de la versión 2.0 cada etapa de la

[5] Son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos.
[6] Diseño asistido por computadoras, más conocido por sus siglas inglesas **CAD** (Computer-Aided Design).
[7] Direct3D es parte de DirectX (conjunto de bibliotecas para multimedia), propiedad de Microsoft. Consiste en una API para la programación de gráficos 3D

pipeline ejecutaba una función prefijada, resultando poco configurable. A partir de la versión 2.0 algunas etapas son programables usando un lenguaje de programación llamado GLSL[8].

OpenGL es una API basada en procedimientos de bajo nivel que requiere que el programador dicte los pasos exactos necesarios para renderizar una escena. Esto contrasta con las APIs descriptivas, donde un programador sólo debe describir la escena y puede dejar que la biblioteca controle los detalles para representarla. El diseño de bajo nivel de OpenGL requiere que los programadores conozcan en profundidad la pipeline gráfica, a cambio de darles libertad para implementar algoritmos gráficos novedosos.

GLUT

GLUT Mecanismos (del inglés OpenGL Utility Toolkit) 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. También posee rutinas para el dibujado de diversas primitivas geométricas (tanto sólidas como en modo wireframe) que incluyen cubos, esferas y teteras. También tiene soporte para creación de menús emergentes.

Los dos objetivos de GLUT son para permitir la creación de código más portable entre diferentes sistemas operativos (GLUT es multiplataforma) y hacer OpenGL más simple. Introducirse en la programación con OpenGL utilizando GLUT conlleva normalmente sólo unas pocas líneas de código y hace innecesario el conocimiento de las APIs específicas de cada sistema operativo.

Todas las funciones de GLUT comienzan con el prefijo glut[9] (por ejemplo, glutPostRedisplay[10] indica que la ventana actual necesita ser redibujada).

Algunas decisiones de diseño del GLUT original hace difícil a los programadores realizar las tareas deseadas. Esto llevó a muchos a crear

[8] Es un lenguaje de alto nivel de sombreado con una sintaxis basada en el lenguaje de programación C.

[9] Es una biblioteca de utilidades para programas OpenGL que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo.

[10] Marca la ventana actual como la necesidad de ser vuelto a mostrar.

parches y las extensiones para GLUT que se alejan del canon. Algunas implementaciones libres también incluyen arreglos diversos.

Las limitaciones más destacables de la biblioteca GLUT original son:

- La biblioteca exige a los programadores que llamen a glutMainLoop()[11], una función que nunca termina. Esto hace difícil que los programadores integren GLUT en aplicaciones o bibliotecas en las que deseen poder tener control sobre su propio ciclo de ejecución. Una forma habitual de solucionar esto es introducir una nueva función, que suele llamarse glutCheckLoop()[12], que ejecute una sola iteración del ciclo de ejecución de GLUT. Otra solución bastante común es ejecutar GLUT por separado mediante hilos[13], aunque esto puede ser distinto dependiendo del sistema operativo y puede provocar problemas de sincronización. Por ejemplo la implementación de GLUT para Mac OS X obliga a que el glutMainLoop() se ejecute necesariamente en el hilo principal.

- El hecho de que glutMainLoop() nunca termine también significa que el programa no sale nunca del ciclo de ejecución. Freeglut soluciona este problema introduciendo una nueva función: glutLeaveMainLoop()[14].

- La biblioteca termina el proceso cuando se cierra la ventana, lo que en algunas aplicaciones puede no ser deseable. Por eso muchas implementaciones incluyen una función extra llamada glutWMCloseFunc()[15].

SOIL

Es una pequeña biblioteca C usada principalmente para subir texturas a OpenGL. Se basa en stb_image[16] versión 1.16. Lo he extendido para cargar archivos TGA y DDS, y para realizar funciones comunes necesarias en la carga de texturas OpenGL. SOIL también puede usarse para guardar y cargar

[11] Entra en el bucle de procesamiento de eventos GLUT.

[12] Ejecute una sola iteración del ciclo de ejecución de GLUT.

[13] Es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.

[14] La función glutLeaveMainLoop hace que freeglut detenga su bucle de eventos.

[15] Termina el proceso cuando se cierra la ventana.

[16] Para cargar un archivo PNG de 32 bits

imágenes en una variedad de formatos (útil para cargar mapas de altura, aplicaciones no OpenGL, etc.)

SOIL está destinado a ser utilizado como una biblioteca estática (ya que es pequeño y en el dominio público). Puede utilizar el archivo de biblioteca estática incluido en el archivo zip (libSOIL.a funciona para los compiladores de MinGW[17] y Microsoft ... siéntase libre de cambiarle el nombre a SOIL.lib si lo hace feliz) o compile la biblioteca usted mismo.

Simple DirectMedia Layer(SDL).

Simple DirectMedia Layer (SDL) es un conjunto de bibliotecas desarrolladas en el lenguaje de programación C que proporcionan funciones básicas para realizar operaciones de dibujo en dos dimensiones, gestión de efectos de sonido y música, además de carga y gestión de imágenes. Fueron desarrolladas inicialmente por Sam Lantinga, un desarrollador de videojuegos para la plataforma GNU/Linux.

También proporciona herramientas para el desarrollo de videojuegos y aplicaciones multimedia. Una de sus grandes virtudes es el tratarse de una biblioteca multiplataforma, siendo compatible oficialmente con los sistemas Microsoft Windows, GNU/Linux, Mac OS y QNX, además de otras arquitecturas y sistemas como Sega Dreamcast, GP32, GP2X, etc.

Se han desarrollado una serie de bibliotecas adicionales que complementan las funcionalidades y capacidades de la biblioteca base.

- SDL Mixer: Extiende las capacidades de SDL para la gestión y uso de sonido y música en aplicaciones y juegos. Es compatible con formatos de sonido como Wave, MP3 y OGG, y formatos de música como MOD, S3M, IT y XM.

- SDL Image: Extiende notablemente las capacidades para trabajar con diferentes formatos de imagen. Los formatos compatibles son los siguientes: BMP, JPEG, TIFF, PNG, PNM, PCX, XPM, LBM, GIF, y TGA,

- SDL Net: Proporciona funciones y tipos de dato multiplataforma para programar aplicaciones que trabajen con redes.

- SDL RTF: Posibilita el abrir para leer en aplicaciones SDL archivos de texto usando el formato Rich Text Format RTF.
- SDL TTF: Permite usar tipografías TrueType en aplicaciones SDL.

[17] Es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows.

2.1 Desarrollo

Después de haber detallado teóricamente los conceptos y herramientas que serán necesarias para el desarrollo del mini juego, llegamos a la parte del desarrollo.

A continuación, se hará una breve descripción de los pasos llevados a cabo para el diseño y el desarrollo del mini juego llamado Motor Racing.

El primer paso es tener claras las tecnologías y las herramientas que serán usadas para desarrollar un mini juego, una vez teniendo definidas todas esas herramientas y tecnologías, nos será más fácil comenzar con la puesta en marcha del proyecto y desarrollo del mini juego.

Como segundo paso, se procedió a instalar todas las librerías y herramientas necesarias para poder comenzar el desarrollo. Tales tecnologías fueron, Debian basado en GNU/Linux como sistema operativo, C++ como lenguaje de programación, se instalaron también todas las librerías y herramientas de compilación, en este caso Build-Essentials, Cmake, también la librería gráfica OpenGL, las librerías SOIL Y SDL para manejo de gráficos y multimedia respectivamente, además de un entorno de desarrollo y pruebas, en este caso elegimos Geany.

El tercer paso ya iniciado el desarrollo fue definir e invocar todas las librerías antes mencionadas en nuestro script con extensión cpp mediante la instrucción #include <librería a invocar.h>

También se definieron y nombraron todas las variables y funciones que se iban a necesitar, así como sus tipos de datos.

Dentro de las funciones definidas, tenemos algunas muy específicas propias de las librerías que estamos utilizando como glTranslatef( ), glRotatef( ), SDL_LoadWAV( ), etc. Pero también muchas definidas por nosotros los desarrolladores para tareas espefícicas como una función denominada calle( ) cuyo objetivo es dibujar mediante figuras geométricas definidas por coordenadas una carretera simulada con estos trazados, la cuál es la que recorrerá nuestro carro. También creamos funciones como la que dibuja el vehículo, la que dibuja objetos, paisaje etc.

Como cuarto paso definimos funciones para controlar la cámara y las vistas, además de hacer uso de las librerías para agregar contenido multimedia. Utilizamos SOIL para agregar texturas y SDL para insertar sonido y darle vida a nuestro mini juego.

Como quinto y último paso hicimos testing del mini juego, se probó muchas veces, se detectaron errores, se corrigieron muchos de ellos, se optimizaron las funciones ya creadas, se revisó el código en busca de partes no comentadas debidamente, se organizó el contenido.

3. Aspectos Administrativos

Esta sección aborda las funciones administrativas que se necesitan desarrollar en todos los proyectos. Destaca la importancia de la planeación previa para enfrentar los retos que surgen en el desarrollo del proyecto llamado **"Desarrollo de un mini juego llamado Motor Racing en OpenGL, C++, SOIL y SDL"** y advierte algunas dificultades que se presentan para responder a estos retos.

3.1 Recurso Humano

Se cuenta con cuatro personas los cuales se encargarán de programar el juego denominado "Motor Racing", aplicando los conocimientos adquiridos en el curso de algoritmos gráficos, con estos conocimientos se llevará a cabo el desarrollo del juego por los programadores.

Nombres de los programadores (Recurso humano para el proyecto)	Cargo
Diego Armando Herrera Flores	Analista y programador
Gerson Alexander Sandoval Guerrero	Analista y programador
Pedro José Andrade Serpas	Programador
Mario Jehudí Cruz Ramírez	Programador

3.2 Presupuesto

CONCEPTO	CANTIDAD	PRECIO	TOTAL
Análisis		$100	$100.0
Diseño		$1500	$1500.0
Desarrollo		$3000	$3000.0
Implementación		$1000	$1000.0
Mantenimiento		$400	$400.0
			$6000.0

3.3 Cronograma

Nombre	Fechas		Mayo				Junio			
	Fecha Inicio	Fecha finalización								
Planificación	22/05/17	29/05/17								
Análisis	29/05/17	02/06/17								
Desarrollo	02/06/17	15/06/17								
Realización de pruebas	15/06/17	16/06/17								
Implementación del mini juego	16/06/17	19/06/17								

En esta sección se presenta la organización en fases e iteraciones y el calendario del proyecto, como todo proyecto es sujeto a variaciones lo cual puede ser un aproximado en relación al tiempo.

4. Referencias

Wikipedia, *enciclopedia libre y grupos de enciclopedias para informar a los usuarios:* https://es.wikipedia.org/wiki/MinGW

Wikipedia, *enciclopedia libre y grupos de enciclopedias para informar a los usuarios:* https://es.wikipedia.org/wiki/OpenGL

Wikipedia, *enciclopedia libre y grupos de enciclopedias para informar a los usuarios:* https://es.wikipedia.org/wiki/Simple_DirectMedia_Layer

Lonesock, *pagina oficial sobre la librería SOIL para OpenGL:*

http://lonesock.net/soil.html

Libsdl, *página oficial sobre la librería SDL para OpenGL*

https://www.libsdl.org/