Capítulo 2. Conceptos generales.

Definición de motor gráfico.

El término “motor gráfico” (graphic engine) es ampliamente utilizado en todo el mundo principalmente por programadores de video juegos. Trataremos a continuación de clarificar su significado. Un motor es una maquina utilizada para propulsar un vehículo. Refiriéndonos al área de la programación, un motor es parte de un proyecto de software que se encarga de propulsar ciertas funcionalidades en el software. Para tener un mejor entendimiento de esto podemos poner el ejemplo de un automóvil en el cual si giramos la llave entonces el motor encenderá. Posteriormente pisamos el acelerador y el motor transmitirá energía cinética a la transmisión lo cual hará que el vehiculo se ponga en movimiento. El conductor no necesita saber exactamente que es lo que esta pasando en el interior del motor, solo le interesa manejar el automóvil. El mismo concepto se aplica para un motor gráfico. Tenemos un determinado método que podríamos llamarle init_Engine() el cual inicializara el adaptador gráfico para recibir ordenes. En el momento que le mandamos información sobre un modelo en 3D, el motor debe desplegarlo en pantalla de la forma en que se le indique. El paso anterior equivaldría a pisar el acelerador del automóvil.

El trabajo de un motor gráfico es el de realizar todas las tareas de bajo nivel tales como comunicarse con el adaptador gráfico, administrar la escena, transformar la geometría del mundo 3D y lidiar con diversos procesos matemáticos que afectan su comportamiento. Todo este trabajo es necesario pero es algo con lo cual el programador inexperto no desea lidiar al desarrollar una aplicación 3D y es entonces cuando se recurre a un motor gráfico en el cual ya se encuentran implementadas las funcionalidades requeridas.

Definición de API.

El término API significa Interfaz para la Programación de Aplicaciones (Application Programming Interfaz), y se refiere a una librería que implementa cierta funcionalidad mediante una interfaz con la cual el programador pueda acceder a dicha funcionalidad [Zerbst, 2004] . Internamente, un motor gráfico hace uso de un API's gráfico para acceder a la funcionalidad que nos proporciona el hardware gráfico. En la actualidad son dos los API's gráficos que predominan en la industria: Direct3D y OpenGL.

Direct3D vs OpenGL.

En los días en que recién comenzó el uso de Windows 95, la mayor parte de los juegos para la arquitectura de Intel eran desarrollados para funcionar bajo el sistema operativo MS-DOS. Microsoft deseaba animar a los desarrolladores a cambiarse a la plataforma de Windows 95 y hacerlo un sistema operativo más popular. Sin embargo, Windows 95 no proveía la funcionalidad necesaria para ser considerada como una buena plataforma de desarrollo de video juegos y fue así como surgió el API DirectX. No fue sino hasta su versión 3 cuando se obtuvo un API que fuese estable y fue hasta su version 5 cuando se obtuvo una API que fuese considerado útil. Para su versión 7 los desarrolladores de videojuegos estaban satisfechos con DirectX el cual se consideraba relativamente fácil de implementar. Hasta ese momento para las funciones gráficas DirectX se dividía en dos componentes: DirectDraw para gráficos 2D y Direct3D para gráficos 3D. Con la llegada de la versión 8 se decidió deprecar la interfaz para DirectDraw ya que se podía tener la misma funcionalidad usando Direct3D para aplicaciones 2D pero con un mayor rendimiento mediante el uso del hardware 3D. [Promit, 2002]

OpenGL fue originalmente desarrollado por Silicon Graphics como descendiente de un API conocido como Iris GL para su sistema operativo basado en UNIX. Fue creado con la idea de establecerse como un estándar por lo cual actualmente esta disponible para un gran número de plataformas. OpenGL es revisado por un comité conocido como

Architecural Review Board (ARB) compuesto por representantes de las compañías más importantes de la industria de los gráficos por computadora. OpenGL a sido usado por años en diversos sectores tecnológicos. Fue desarrollado con una clara visión del futuro y se ha mantenido estable y consistente. [Promit, 2002]

En general, Direct3D esta hecho para funcionar solamente bajo la plataforma Windows, incluyendo la versión diseñada para la consola de video juegos Xbox. Organizaciones externas a Microsoft han realizado intentos para portar Direct3D a otras plataformas ajenas a Windows pero solo se ha logrado de forma parcial debido a la gran dependencia que tiene este API con el sistema operativo. Por otro lado ya hemos mencionado que OpenGL esta disponible para un gran número de plataformas tales como UNIX, Linux, Windows, Mac OS y consolas de videojuegos como las fabricadas por Nintendo y Sony. Como podemos ver, en términos de portabilidad Direct3D es una opción más limitada debido a que esta enfocado únicamente a las plataformas Windows [Hsieh, 1997] [Promit, 2002]. Sin embargo si se planea utilizar esta plataforma, Direct3D sería una mejor opción al ofrecer la máxima compatibilidad con éste sistema operativo.

Funcionalmente Direct3D esta diseñado para ser una interfaz con el hardware 3D de modo que las características disponibles dependerán de las características del hardware 3D que se tiene. Por otro lado OpenGL es un sistema que procesa la información y que opcionalmente puede ser acelerado mediante el uso de hardware 3D pero no es necesariamente requerido. [Sharman, 2003] [Promit, 2002]

En términos de facilidad de uso, Direct3D le deja a la aplicación en desarrollo la tarea de manejar los recursos de hardware lo cual causa que se requiera una mayor cantidad de código para ponerse en marcha mientras que OpenGL puede tener implementaciones para la misma tarea en una o pocas llamadas haciendo muchos procedimientos transparentes al usuario aunque con el costo de tener un menor control sobre lo que esta pasando en el hardware. [Sharman, 2003] [Promit, 2002]

Finalmente cabe señalar la diferencia entre el sistema de extensión de características que a adoptado cada uno de estos dos API’s. Por un lado, OpenGL deja que el fabricante de hardware 3D implemente las características adicionales en sus controladores de

hardware. Esto permite que las nuevas características estén disponibles con mayor rapidez aunque inicialmente genera una confusión entre desarrolladores ya que es solo después de un tiempo cuando se estandarizan estas funciones como parte integra del API y no solo como una funcionalidad exclusiva de determinado fabricante. Por otro lado las especificaciones de Direct3D son dadas por una sola organización, que en este caso es Microsoft, por lo cual el API se mantiene más consistente. Esto conlleva a que las actualizaciones del API se lleven a cabo de una forma no muy periódica y que algunas veces se pierdan de vista algunas características importantes recién implementadas en el hardware de algún fabricante. [Sharman, 2003] [Promit, 2002]

Dentro de este proyecto de tesis se a elegido utilizar el API OpenGL pues uno de los objetivos radica precisamente en mantener un código estandarizado para poder facilitar la tarea de portar en un futuro el software a otras plataformas diferentes al sistema operativo Windows.

Librería SDL.

SDL (Simple DirectMedia Layer) es una librería multimedia multiplataforma creada y mantenida por Sam Lantiga y una gran comunidad de programadores diseñada para proveer acceso de bajo nivel al hardware de audio, teclado, mouse, joystick y hardware gráfico (hardware 3D vía OpenGL). Esta librería es usada actualmente en aplicaciones como reproductores de video, emuladores, y videojuegos. SDL soporta actualmente las plataformas Linux, Windows, BeOS, MacOS, FreeBSD, OpenBSD, BSD/OS, Solaris, IRIX y QNX. Existen también versiones no oficiales para Windows CE, AmigaOS, Dreamcast, Atari, NetBSD, AIX, OSF/Tru64, RISC OS y SymbianOS. SDL esta escrito en C pero funciona nativamente en C++ y existen versiones para trabajar en otros lenguajes de programación como Ada, Eiffel, Java, Lua, ML, Pascal, Perl, PHP, Pike, Python y Ruby. [SDL, 2006]

El papel que juega SDL dentro de este proyecto es el de controlar los eventos activados por los dispositivos de entrada (teclado y mouse) así como las propiedades del adaptador de video. Sin SDL, esto puede llegar a ser una tarea desalentadora y tediosa

debido a que cada sistema operativo maneja estas tareas de una forma distinta. Al usar SDL aseguramos que nuestro software pueda funcionar en una amplia variedad de sistemas operativos.