Capítulo 1. Introducción

1.1 Definición del problema

Cuando el famoso juego “Doom” apareció, no solamente nos asombró el grandioso juego, sino que también trajo y popularizo un nuevo modelo de programación de juegos: un “Motor de Juegos”. Este permitía a programadores y diseñadores de juegos, entrar al núcleo del juego para poder modificar y crear nuevos juegos con nuevos modelos, sonidos y escenarios. Para empezar debemos distinguir bien lo que es un “Motor de Juegos” y un “Video Juego”, ya que no son lo mismo. Sería como confundir el procesador de una computadora con toda la computadora. Uno puede quitarle el procesador a una computadora y construir otra computadora con este procesador, y así para muchas computadoras. De la misma forma uno puede tomar el motor de juegos de un video juego y crear otro video juego o crear alguna aplicación en 3D. En un aspecto más técnico podríamos decir que el juego serían todos los modelos, los sonidos, las animaciones, los escenarios y el “Motor de Juegos” incluiría todas las especificaciones tecnológicas que no se ven en el juego como por ejemplo el Renderer, la manipulación de los objetos 3D, los métodos de iluminación, los métodos de sombras, los métodos de reflejos, las matemáticas, los administradores de buffer, texturas y materiales, la administración de los shaders, los dispositivos de entrada, los sonidos, la red, y la inteligencia artificial.

Un “Motor de Juegos” bien diseñado debe ser modular, re-usable, aprovechar al máximo el rendimiento de las tarjetas gráficas y el hardware, y lo suficientemente flexible para poder ser usado en el diseño y desarrollo de otros juegos u otras aplicaciones 3D. Otro aspecto muy importante que debe considerarse al crear un “Motor de Juegos” es saber y conocer bien las características de las computadoras o el tipo de

hardware en el cual va a poder funcionar este, ya que para obtener un buen rendimiento en efectos y gráficos 3D se necesita de hardware especial que no todas las computadoras poseen. En la actualidad los nuevos Video Juegos o aplicaciones 3D necesitan de estas computadoras con hardware gráfico especial para poder jugarse o verse.

Dentro de los mismos Video Juegos existen diferentes tipos de juegos, como por ejemplo juegos en primera persona, juegos de estrategia en tiempo real y juegos de simulación de vehículos. Esto es importante ya que cada tipo de "Motor de Juegos" se enfoca un poco más en algunos aspectos que en otros. Por ejemplo un "Motor" para juegos de combate en primera persona se enfoca más en el detalle visual, en las animaciones, en las texturas, en el número de polígonos en escena y en los métodos Culling, que es el proceso de esconder o remover polígonos que no aparecen en escena. Un ejemplo de estos juegos son el "Doom", "Quake" y "Unreal Tournament".

Para los juegos de estrategia en tiempo real el "Motor de Juegos" se enfoca más en mostrar gran cantidad de objetos al mismo tiempo y en una gran superficie. En este tipo de "Motor" se tiene una vista aérea y se puede reducir el nivel de detalle (LOD-Level of Detail) ya que el jugador nunca se acerca demasiado a los objetos. Un ejemplo de estos juegos son "Age of Empires", "Empire Herat" y "Age of Mythology".

Para los juegos de simulación de vehículos que incluyen la simulación de aviones, helicópteros, tanques, coches de carreras y motos de carrera, el "Motor" debe enfocarse primordialmente en el área de los métodos Culling y en nivel de detalle (LOD), para reducir el número de polígonos a texturizar y pintar en escena, al igual del uso de "Fog" para dar el efecto al jugador de ver objetos aparecer en la distancia y

desaparecer al pasarlos. Algunos ejemplos de estos juegos son “Need For Speed” y “Comanche 4”.

Hace unos cuantos años los desarrolladores de video juegos necesitaban muchas líneas de código sobre todo en un lenguaje de bajo nivel como lo es “ensamblador” para crear gráficos sorprendentes con una mayor eficiencia en las tarjetas de video aceleradoras y en el hardware. Esto ocasionaba que no todos los programadores pudieran desarrollar un “Motor de Juegos” y también los limitaba a obtener aplicaciones 3D sin poder explotar al máximo el hardware y los recursos de una computadora dando como resultado muy pocos efectos visuales. Debido a este problema fue que la compañía productora de tarjetas de video aceleradoras, NVIDIA, con la colaboración de Microsoft, crearon un nuevo lenguaje de programación de alto nivel para gráficos llamado High Level Shading Language (HLSL). Con la creación de este nuevo lenguaje de alto nivel de programación NVIDIA y Microsoft pretenden ayudar a los programadores de gráficos a reducir el tiempo de desarrollo de complejas funciones y métodos gráficos al igual que la reducción de líneas de código con comandos simples. También se pretende optimizar el procesamiento de este nuevo lenguaje a través de la capa del API de DirectX y el GPU(Unidad de Procesamiento de Gráficos) o VPU(Unidad de Procesamiento Visual). Otro aspecto importante de este nuevo lenguaje HLSL es que incluye un compilador, que es quien se encargará de compilar los Shaders para aprovechar las características de los GPUs o VPUs, y así también poder cambiar solamente de versión de compilador ante nuevas tecnologías sin necesidad de reescribir las aplicaciones, y así poder mover los juegos a nuevas tecnologías de hardware sin problemas.

Con la aparición de más novedosos video juegos con grandes efectos y con mayores requerimientos de hardware fue que se empezó a pensar en la distribución del trabajo entre el GPU y el CPU para una mejor optimización en los recursos de hardware. No solamente se necesitaba tener trabajando al 100% el puro GPU o el puro CPU, sino que se podía distribuir el trabajo entre estas dos unidades de procesamiento y así pudieran trabajar en forma paralela para un mejor rendimiento en los videojuegos o aplicaciones 3D. En la actualidad esta apareciendo una nueva tecnología en cuanto a GPU's o VPU's en donde se puede tener en una computadora dos de estos trabajando en paralelo para un máximo desempeño de hardware y la posibilidad de más y mejores efectos visuales. Para la compañía NVIDIA esta tecnología se conoce como Scalable Link Interface (SLI) y para la compañía ATI se conoce como CrossFire. Existen muy pocas computadoras con esta tecnología ya que apenas se esta introduciendo en el mercado y por lo tanto es muy poco accesible económicamente a los aficionados de los video juegos, pero en un futuro se piensa adoptar esta nueva tecnología para un máximo desempeño gráfico en las aplicaciones 3D.

Para la optimización del CPU se han creado varios conjuntos de instrucciones basadas en el lenguaje ensamblador que ayudan a la optimización de operaciones. Debido a la competencia entre Intel y AMD se han desarrollado varios de estos conjuntos de instrucciones y se han ido mejorando desde su aparición, por ejemplo MMX, 3DNow!, SSE, SSE2 entre otros nuevos.

Gracias a todas estas innovaciones tecnológicas y a los nuevos lenguajes de programación, un motor de juegos se encarga de la parte técnica como la administración y manejo de objetos 3D, sonidos, dispositivos de entrada, comunicación con la red,

inteligencia artificial, entre otras funciones técnicas permitiendo a los desarrolladores y programadores de aplicaciones 3D y video juegos enfocarse en una programación a mas alto nivel como el diseño de modelos, escenarios, efectos visuales, edición de sonidos, y creación de historias. Los motores de juegos permiten una mayor rapidez y facilidad en el desarrollo de aplicaciones 3D y videojuegos dando como resultado un ahorro en los gastos del desarrollo de este tipo de proyectos a las empresas dedicadas a este tipo de productos.

1.2 Objetivo General

El objetivo de esta tesis es desarrollar el núcleo de un “Motor de Juegos” que sea útil para los desarrolladores de videojuegos o aplicaciones 3D. Para este Motor de Juegos se implementarán algunas funciones principales, enfocándose primordialmente en las características técnicas principales para que sea modular, re-usable y aproveche lo mejor posible los recursos de hardware de una computadora.

Existe un gran dilema en cuanto a cuando un Motor Gráfico se considera un Motor de Juegos o que es lo que se considera realmente un Motor de Juegos, pero varios autores y expertos en el tema coinciden en que un Motor Gráfico pasa a ser Motor de Juegos cuando no solamente administra objetos y modelos 3D, sino que también administra otro tipo de dispositivos diferente al render, como dispositivos de entrada, de audio, de red o cuando se agregan diferentes módulos como el de Inteligencia Artificial. Este proyecto a desarrollar se considera Motor de Juegos ya que va a tener un sub-Motor Gráfico, un sub-Motor Input, y un sub-Motor de Audio.

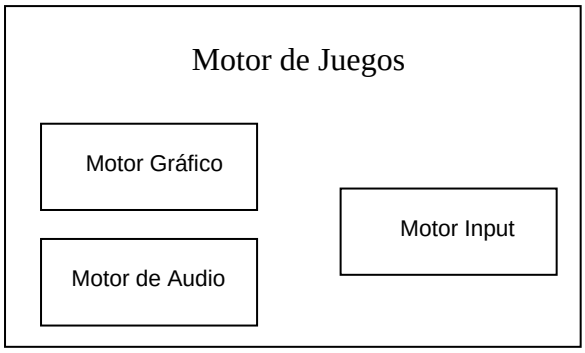

Figura 1 Motor de Juegos con los tres sub-Motores.

1.3 Objetivos Específicos

En esta tesis se va a desarrollar el núcleo de un Motor de Juegos en donde se implementarán algunas funciones principales. Este Motor estará enfocado a juegos de combate en primera persona que sea modular, re-usable y lo suficientemente flexible para que se pueda utilizar en el diseño y creación de diferentes juegos o aplicaciones 3D de este tipo. Aprovechará los recursos de hardware lo mejor posible distribuyendo el trabajo al CPU y al GPU. Se trabajará el CPU con un conjunto de instrucciones de bajo nivel llamado Streaming SIMD Extensión (SSE) para operaciones de matrices más rápidas y un mejor aprovechamiento del CPU, mientras que para el GPU se utilizarán Shaders para su mejor desempeño.

Este “Motor de Juegos” estará formado por un sub-Motor 3D, un sub-Motor de Audio, y un sub-Motor Input. Dentro del sub-Motor 3D se encontrara el Render, que es la parte más importante de los motores de juegos ya que este es el que hace que las figuras, materiales, texturas, superficies, etc., se desplieguen en la pantalla. Mientras mejor sea el Render, mejor será catalogado el Motor de Juegos, es una parte muy importante del motor. El sub-Motor de Input soportara la entrada del Teclado, el Mouse, y un Joystick o un Joypad. Mientras que el sub-Motor de Audio podrá abrir diferentes formatos de audio. Se podrán manipular los vértices y los píxeles de los modelos 3D a partir de archivos shaders en un lenguaje de alto nivel como lo es HLSL.

El Motor 3D estará conformado por las siguientes partes:

- Un administrador de materiales y texturas: Una clase que administre tanto los materiales como las texturas de cada modelo 3D para que no se carguen y

guarden materiales o texturas iguales y no alteren el rendimiento del motor ni la memoria.

- Un administrador de buffer para Renderear: Encargado de renderear los vértices y presentar en la pantalla las imágenes o modelos 3D.

- Un administrador de movimientos de Cámara: Encargado de administrar los movimientos y tiempos de la cámara, tanto en Primera Persona como Libre.

- Una librería encargada de las matemáticas 3D: Encargada de las matemáticas relacionadas a los gráficos en 3D en donde se utilizará el conjunto de instrucciones SSE para una mayor rapidez en las operaciones de matrices.

- Abrir y cargar Vertex y Píxel Shaders: Se podrán crear grandes efectos visuales con la ayuda de los shaders en un lenguaje de alto nivel como lo es HLSL y se podrán modificar estos shaders sin necesidad de re-compilar toda la aplicación.

- Abrir y cargar archivos de DirectX (.x): Se podrán exportar diferentes modelos 3D desde aplicaciones enfocadas a diseño y modelado como lo es MilkShape y se podrán renderear y manipular por este sub-Motor Gráfico.

El Motor de Audio constará de las siguientes partes:

- Función para abrir sonidos: Se encargará de abrir diferentes formatos de audio.

- Función para manipular sonidos: Se podrán manipular varios y diferentes sonidos cargados para poderlos utilizar durante alguna aplicación.

- Función para reproducir sonidos: Se podrán reproducir diferentes sonidos a la vez y detenerse cuando se les indique ya sea uno por uno o varios a la vez.

El Motor de Input constará de las siguientes partes:

- Un administrador de Joystick: Se encargará de detectar algún joystick o joypad conectado a la computadora, se obtendrán sus capacidades y su funcionamiento como dispositivo de entrada.

- Un administrador de Mouse: Se encargará de detectar y obtener las capacidades del Mouse para poderlo utilizar como un dispositivo de entrada.

- Un administrador de Teclado: Se encargará de detectar y obtener las capacidades del Teclado para poderlo utilizar como un dispositivo de entrada.

1.4 Alcances

Para la implementación de este núcleo de “Motor de Juegos” únicamente se utilizara el API de DirectX. Sólo se realizarán algunas de las partes principales del “Motor” ya que para crear un “Motor de Juegos” completo se necesita un mayor número de desarrolladores, mayor dedicación y un mayor tiempo de desarrollo. Este Motor estará formado por 3 sub-Motores, uno gráfico, uno de audio y uno de input. Se enfocará más a la parte técnica para que este “Motor de Juegos” sea modular, re-utilizable y aproveche lo mejor posible los recursos del hardware como lo es el CPU y GPU. Contará con un nuevo lenguaje de alto nivel (HLSL) para la programación de shaders. Se podrán manipular los vértices y píxeles de las modelos 3D y los meshes con los shaders sin necesidad de que se recompile todo el código tanto de la aplicación como del Motor. Si se desean añadir e implementar nuevas funciones se puede hacer de manera fácil en cualquiera de los sub-Motores. Al hacer modificaciones en los sub-Motores solo se necesitará reemplazar al antiguo DLL por el nuevo en la aplicación que este utilizando el Motor. Esto permitirá que pueda haber nuevas versiones del Motor sin necesidad de cambiar código en los juegos o aplicaciones que lo estén utilizando. Los sub-Motores se podrán utilizar por separado, independientemente el uno del otro.

1.5 Limitaciones

Para que este Motor de Juegos se pueda utilizar se necesitarán computadoras con hardware especial como lo son tarjetas gráficas o tarjetas de video aceleradoras con GPU o VPU que soportan Shaders versión 2_0. Computadoras que tengan procesadores Pentium III , AMD Athlon XP o mas recientes debido a que se utiliza Streaming SIMD Extensions (SSE). Solo se podrá utilizar en plataformas de Microsoft Windows debido a que utiliza DirectX. Se necesitará que la computadora tenga instalado DirectX versión

9.0c. El Sistema Operativo deberá ser Microsoft Windows 98SE o mas reciente debido a que se utiliza SSE.

1.6 Hardware Utilizado

Para el desarrollo de esta tesis se utilizó una computadora ensamblada con las siguientes características:

- Procesador Pentium 4 a 3.2 GHz.
- 1 GB de Memoria RAM.
- Tarjeta gráfica GeForce 7800 GTX con 256M de memoria.
- Tarjeta Sound Blaster Live! 24-bit
- Monitor Samsung 17”
- Bocinas Acteck 5.1 canales
- Game Pad Microsoft SideWinder USB
- Mouse Perfect Choice óptico USB
- Teclado USAP Multimedia PS2

1.7 Software Utilizado

El software que se utilizó fue:

- Microsoft Visual C++ .NET versión 7.1
- Microsoft DirectX 9.0 SDK Update October 2005
- Microsoft DirectX 9.0 High Level Shading Language