Capítulo 2. Marco Teórico

2.1 Motor de Juegos

Existen muchas discusiones y mucha polémica acerca del verdadero significado de este término, sin embargo la mayoría de los desarrolladores de video juegos y especialistas en el área concuerdan en ciertos aspectos que son fundamentales en un Motor de Juegos y que se puede llegar a decir que es un significado general. Cuando un Motor relacionado al desarrollo de video juegos sobre pasa las expectativas de un Motor Gráfico e incluye el manejo ajeno a componentes relacionados al rendereo, es considerado un Motor de Juegos. (Stephens, 2001)

Otra definición en la cual varios expertos en el tema concuerdan es que un Motor de Juegos es el componente de software central de un video juego que maneja todos los elementos fundamentales para el desarrollo de este, como lo son el rendereo de gráficos, tareas con el audio, con el input, con matemáticas 3D, red e Inteligencia Artificial, entre otros.(Eberly, 2000)

Este término nació a mediados de los 90's y es relacionado especialmente con los video juegos de tipo Combate en Primera Persona, cuyas siglas en inglés son FPS (First-Person Shooter). Los precursores de este nuevo modelo de programación de juegos y aplicaciones 3D fue la compañía productora de video juegos idSoftware, con los video juegos Quake y Doom para PC. (Simpson, 2005)

Figura 2 Vista del Juego Doom 1

Figura 3 Vista del Juego Quake 1

Con este nuevo modelo de programación, se permitía a programadores, artistas o fans, poder entrar al núcleo del video juego y poder crear nuevos personajes, gráficos, armas, escenarios y niveles con cierta facilidad y después verlos dentro del mismo video juego.

Las siguientes generaciones de video juegos, en especial los de Combate en Primera Persona, incluían un Motor de Juegos y el video juego por separado. Así se cumplía el objetivo de que el desarrollo de video juegos posteriores fueran mucho más rápidos y fáciles con la reutilización del Motor, esencial para la competencia en la industria de los juegos. En la actualidad, gracias a los Motores de Juegos, para el desarrollo de un video juego no solamente se necesitan programadores, sino se ha dividido el personal en más áreas, como lo son artistas, diseñadores, guionistas, escritores, efectos especiales, rendereo entre muchos otros.

También cabe destacar que hay diferentes tipos de Motores de Juegos, no solamente los de Combate en Primera Persona, sino también existen los enfocados en Estrategia en Tiempo Real y los de Simulación de Vehículos. Cada tipo tiene sus

diferentes características, y cada uno está especializado en su área y en los efectos que más utilice. Por ejemplo los de Combate en Primera Persona desarrollan más la parte del nivel de detalle visual, la vista y efectos de las texturas, las animaciones, el número de polígonos en escena, el sonido 3D y modo de red para múltiples jugadores.

Para los juegos de Estrategia en Tiempo Real consideran más la parte de poder mostrar gran cantidad de objetos al mismo tiempo en una gran superficie en donde se puede sacrificar el nivel de detalle visual, y no se necesita forzosamente de un sonido 3D, ni de un modo de red para múltiples jugadores.

En cambio, para los Motores enfocados en juegos de Simulación de Vehículos consideran más la parte del nivel de detalle visual, la administración de polígonos en escena, efectos de luz y resplandores, efectos "Fog" para hacer más real el ver objetos aparecer en la distancia, el sonido 3D, y el permitir utilizar dispositivos de entrada con modos Feedback como lo son los volantes o palancas.

Debido al gran avance en efectos gráficos en los videojuegos, se requiere de un hardware especial para que tanto estos Motores junto con los Video Juegos puedan ser instalados y ejecutados. Al comprar un video juego o una aplicación 3D es importante fijarse en los requerimientos, cuanto espacio en disco duro se necesita, cuanto de memoria RAM, que tipo de procesador y a que velocidad, que Sistema Operativo, que API gráfico, y que generación o tipo de tarjeta gráfica con cuanto de memoria en ella lo soportan. Muchas veces uno no se fija en los requerimientos hasta cuando uno instala el video juego y se da cuenta que su computadora no es capaz de soportarlo. Esta parte de los requerimientos de hardware son muy importantes en la industria de los video juegos, ya que se debe tomar en cuenta que hardware esta al alcance de los clientes para no

limitarlo y así poderlo vender. No tendría caso hacer un video juego con efectos visuales muy avanzados pero que sólo fuera soportado por la última generación de tarjetas gráficas ya que este fracasaría y no se vendería mas que a las pocas personas que tuvieran este hardware que además es caro.

2.2 DirectX

DirectX surgió en septiembre de 1995 con la aparición del Sistema Operativo Microsoft Windows 95. La creación de este API fue principalmente debido a que los programadores de video juegos veían al sistema operativo anterior, DOS, una mejor plataforma para el desarrollo de video juegos, lo que significaba un probable fracaso en Windows 95. Fue por eso que el equipo integrado por Craig Eister, Alex St. John y Eric Engstron fueran asignados a una misión obteniendo como resultado lo que ahora conocemos como DirectX.

Esta librería de controladores, DirectX, fue desde un principio pensada especialmente para mejorar los gráficos y los sonidos en la plataforma de Microsoft Windows. Estos controladores en la actualidad son indispensables para la mayoría de los video juegos y se recomienda tenerlos instalados si alguna computadora bajo la plataforma de Windows piensa ser utilizada para actividad multimedia. (Softonic, 2004)

En la actualidad DirectX es una colección de API's cuyos componentes principales son:

- Direct3D: Para dibujar gráficos en 3D y permite optimizar su renderización tomando en cuenta las ventajas del hardware que existe en la computadora, y si no se encuentra hardware lo hace a nivel software.
- DirectDraw: Para manipular la memoria de video y aplicar técnicas para la copia de bloques de esta memoria, mezclar mapa de bits, intercambio de páginas entre otras. (González, 2000)
- DirectInput: Para el proceso de datos de dispositivos de entrada como el teclado, Mouse y Joysticks.

- DirectPlay: Para la comunicación por red encargándose de las características multijugador en los video juegos.
- DirectSound: Para tocar y grabar sonidos proporcionando tecnologías de mixing, sonido estéreo y 3D tomando en cuenta las capacidades del hardware que posea la computadora.
- DirectMusic: Para músicos profesionales.
- DirectSetup: Para la instalación de componentes de DirectX o automatización de la instalación de controladores de DirectX.

Como podemos ver, esta colección de API's es para la plataforma de Microsoft Windows, es gratuita y la mayoría de los juegos de PC que lo utilizan lo traen consigo. La última versión de este API es la 9.0c, que también soporta Shader Model 3.0, y la versión 10 aún es de prueba para el nuevo Sistema Operativo Microsoft Windows Vista, el cual soportara Shader Model 4.

DirectX también ha sido tomado en cuenta para consolas de video juegos, ya que gracias a la colaboración entre Microsoft y NVIDIA, se pudo crear la famosa consola Xbox (Abreviación de DirectXbox), que utiliza un API parecido a DirectX versión 8.1.

2.3 GPU’s y VPU’s

El primer GPU fue diseñado por una de las más famosas compañías productoras de hardware para gráficos, NVIDIA. La primera tarjeta aceleradora de gráficos en el mercado que poseía un GPU fue la GeForce 256. Esta Unidad de Procesamiento de Gráficos (GPU), es muy parecido al CPU de una computadora, solamente que dedicada al procesamiento de gráficos para un mejor desempeño y rendimiento en aplicaciones como los video juegos. Esta primera tarjeta aceleradora tenía la capacidad de calcular miles de millones de operaciones por segundo, procesar 10 millones de polígonos por segundo y contaba con 22 millones de transistores. En la actualidad, las GPU’s tienen más y mejores capacidades, y pueden incluso superar la velocidad de algunos CPU’s antiguos, ya que han llegado a tener una velocidad de entre 500 y 600 MHz.

Después de la llegada del API DirectX versión 8.0, los GPU’s agregaron la tecnología de los Shaders, en donde cada píxel y vertex podían ser programados con pequeños archivos. La primera tarjeta en el mercado con estas capacidades fue la GeForce3. Para todo esto, se necesitaban muchas operaciones con matrices y vectores en el GPU, por lo que ingenieros y científicos han estado estudiando la manera de poder combatir esta situación para mejorar el desempeño de las tarjetas gráficas. Una de estas soluciones propuestas ha sido la utilización de GPU’s de forma paralela, como la tecnología SLI de NVIDIA y CrossFire de ATI. Otra alternativa es la de aprovechar al máximo las capacidades de una computadora y utilizar tanto el CPU y el GPU distribuyendo el trabajo entre ellos sabiendo que tareas asignar a cada uno. Esta última alternativa es la más utilizada, ya que es más económica y esta al alcance de la mayoría de los clientes consumidores de video juegos y aplicaciones 3D, aparte de que las tecnologías SLI y Crossfire apenas están empezando.

VPU significa Unidad de Procesamiento Visual, cuya funcionalidad principal también es el procesamiento de gráficos al igual que un GPU. Podríamos decir que son lo mismo, solo que por causas de competitividad entre empresas, para NVIDA es GPU, y para ATI es VPU.

Figura 4 GPU de tarjeta gráfica NVIDIA GeForce 6600 GT

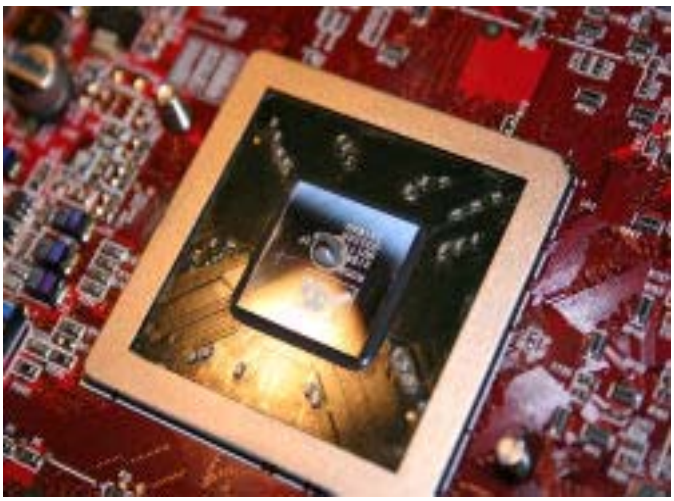

Figura 5 VPU de tarjeta gráfica ATI Radeon 9800 PRO

2.4 High Level Shading Language (HLSL)

Este lenguaje shader, High Level Shader Language, fue desarrollado por Microsoft para usarse con Direct3D. Para poder entender un poco más sobre este lenguaje shader, debemos empezar por saber que es un Shader.

Un Shader es un programa usado en gráficos 3D por computadora, en donde se determinan las propiedades finales de una superficie de un objeto o imagen, como lo son la luz, las texturas, los reflejos, las sombras, entre otros efectos. Fueron introducidos por primera vez en RenderMan de PIXAR, y después de esto fueron tomando mucho más fuerza en el campo de los gráficos 3D, ya que brindaban gran flexibilidad y eficiencia a los programadores acelerando el tiempo de desarrollo y proporcionando grandiosos efectos a modelos 3D. Hay dos tipos de Shaders, los Vertex Shaders y los Píxel Shaders.

Los Vertex Shader manipulan los vértices y los datos e información sobre estos, para operaciones matemáticas, la orientación en el espacio, las coordenadas de las texturas y diferencias en el color. Cabe aclarar que los Vertex Shaders no pueden crear ni destruir vértices, solo los pueden manipular. Estos son usados para crear efectos visuales especiales a objetos 3D manipulando sus vértices como la animación de personajes utilizando “bones” y la deformación de superficies. (NVIDIA)

Los Pixel Shaders, al igual que los Vertex Shaders, son pequeños programas que son procesados y ejecutados en los GPU’s o VPU’s. Pero estos manipulan y procesan todo lo relacionado con los píxeles, creando también grandes efectos como interacción con agua, normal maps, fog, permitiendo la alteración de la luz y de la superficie,

creando efectos visuales muy reales. Para este tipo de Shaders, se necesita información que manejan los Vertex Shaders como por ejemplo la orientación en el espacio y los vectores de la vista. (NVIDIA)

Los Shaders fueron incorporados por primera vez en DirectX en la versión 8.0, y eran programas cortitos y en un lenguaje ensamblador. Fue por esto que en la versión 9.0 se agregó High Level Shading Language, que tenía una sintaxis parecida a C y era más fácil para la mayoría de los programadores. Este lenguaje de alto nivel de Shaders se puede implementar desde otros lenguajes de alto nivel como C++ y C# para un desarrollo más rápido, grande, y con Shaders más complejos. En el siguiente diagrama se puede observar como es que trabajaban las diferentes versiones de Shaders en DirectX, en el de la izquierda con ensamblador y en la derecha con HLSL. El funcionamiento es igual, solo cambia la escritura de los programas Shaders, uno en ensamblador y otro en una sintaxis parecida a C, y el compilador para cada uno. Esto ofrece una ventaja ya que para futuras versiones o cambios o aumentos en la funcionalidad de los shaders, solo se tendrá que actualizar el compilador y no los programas, ya que las nuevas versiones deben soportar los Shaders de antiguas versiones.

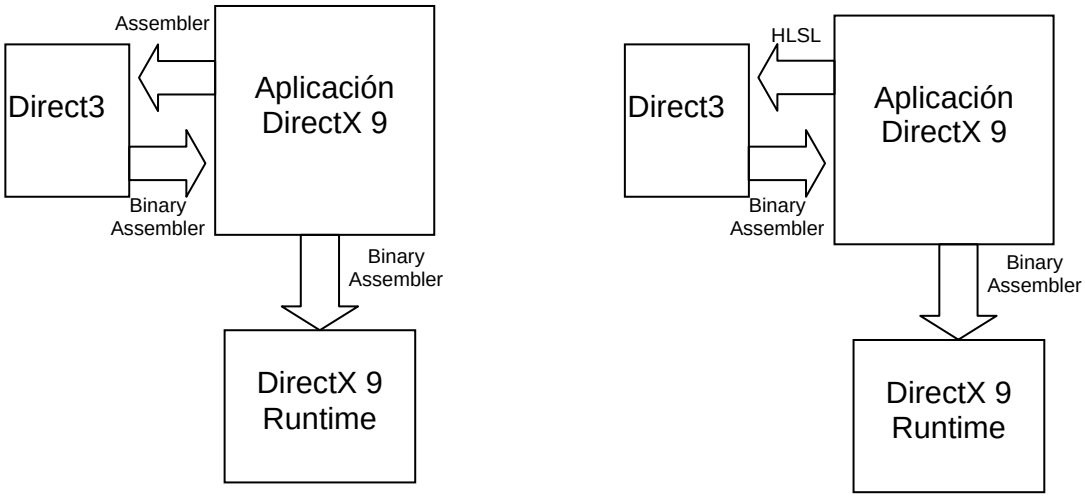

Figura 6 Comunicación entre la aplicación, Direct3D y DirectX 9 Runtime utilizando shaders en HLSL o shaders version 1_x

Al utilizar shaders en un Motor de Juegos no hay que olvidar que también se limita al hardware, ya que no todas las tarjetas gráficas los soportan. En la actualidad podríamos decir que la mayoría de estas tarjetas, ya soportan los shaders. Otro aspecto a considerar es que la versión de shaders para HLSL es a partir del 2_0, por lo que se limita un poco más. Ahora en la industria de tarjetas gráficas podemos encontrar gran variedad de estas, y no nos preocupemos tanto con estas limitaciones, ya que desde la 4 generación de tarjetas gráficas soportan fácilmente los shaders versión 2. En la actualidad estamos en la 7 generación de tarjetas y la última versión de shaders es la 3.

2.5 Streaming SIMD Extensións (SSE)

Streaming SIMD Extensions es un conjunto de instrucciones en forma de lenguaje ensamblador que permite acelerar los cálculos matemáticos en el CPU. SIMD son las iniciales de Single Instruction Multiple Data, que a diferencia del antiguo lenguaje ensamblador, puede procesar con una sola instrucción varios paquetes de datos, ya que los registros son más grandes. Otra versión anterior a SIMD era Single Instruction stream, Single Data Stream (SISD), en ingles y como podemos ver en la siguiente figura la diferencia entre estas dos tecnologías.

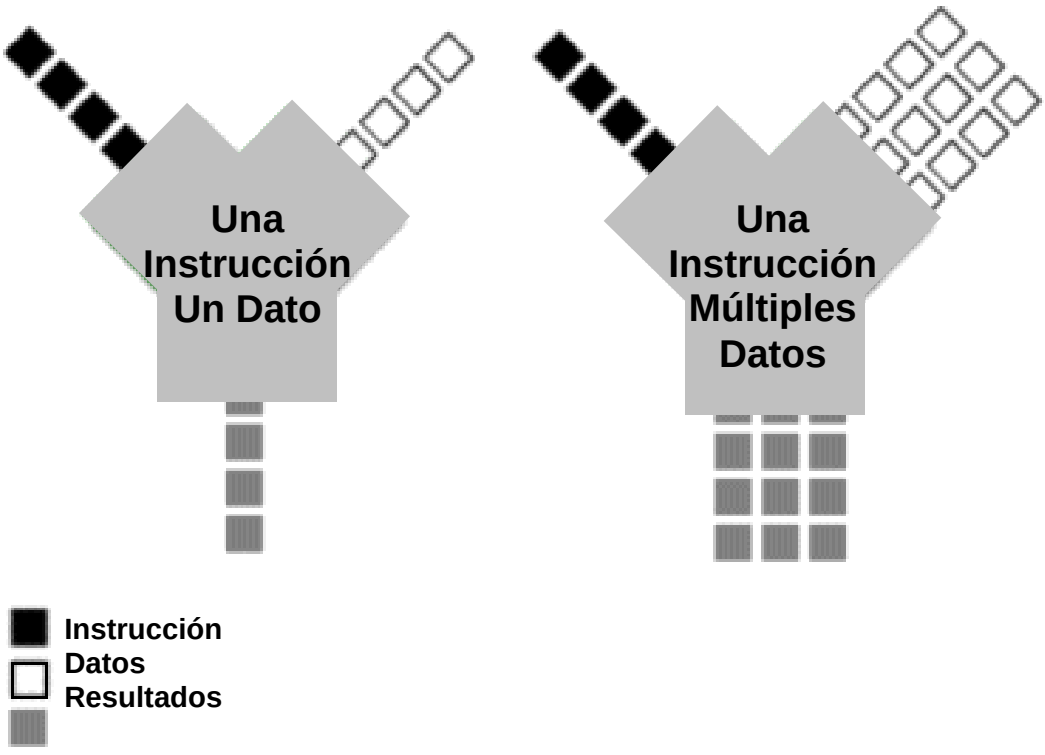

Figura 7 Comparación en el desempeño entre SIMD y SISD

SSE fue introducido en 1999 por Intel en su procesador Pentium III y después este conjunto de instrucciones fue agregado y soportado por los procesadores Athlon

XP de AMD, ya que el desempeño que esta nueva tecnología ofrecía era una gran ventaja para el procesamiento de datos.

Cuando SSE apareció, lo hizo junto con 8 nuevos registros de 128-bits llamados XMMO al XMM7, en donde se podían almacenar en cada registro cuatro paquetes de 32-bits de números de punto flotante de precisión simple cada uno. Con esto los programadores de gráficos 3D por computadora quedaron maravillados, ya que con la ayuda de estos registros se podían almacenar vectores 4D, y SIMD conocía todas las operaciones necesarias para el cálculo de gráficos en 3D. Proporcionando a los programadores todo lo necesario para poder trabajar en gráficos 3D por computadora y explotar los recursos de un procesador para un mejor desempeño y una mayor velocidad.

2.6 Quaterniones

Los quaterniones fueron inventados por Sir William Rowan Hamilton, un matemático irlandés quién a la edad de 9 años ya sabía 13 lenguas y a los 15 ya había estudiado los principios de Newton. Durante toda su vida el se entregó fielmente 22 años al estudio de los quaterniones.

Para describir la rotación de un objeto en el espacio hay tres formas de hacerlo. La primera es usando simples transformaciones de matrices, la segunda es usando ángulos Euler y la tercera es usando quaterniones. En los quaterniones se puede guardar toda la información necesaria sobre la orientación de un objeto reduciendo la necesidad del almacenamiento de memoria en el proceso. De una forma simple podemos decir que un quaternion es un numero cuatro-dimensional, y no son lo mismo que vectores 4D que solo contienen 4 números uni-dimensionales. Para entender un poco mejor los quaterniones debemos mencionar que los números uni-dimensionales son escalares, los bi-dimensionales son los números complejos, los tri-dimensionales aún no los inventan o descubren, y los cuatro-dimensionales son los quaterniones. Los quaterniones tienen 3 partes imaginarias y puede ser representado por un valor escalar y un vector 3D, en donde el valor escalar es el ángulo de rotación aplicado al objeto y el vector representa los ejes en los cuales la rotación tiene lugar. La ventaja es que no se necesitan tres ángulos de rotación para cada eje, sino simplemente se suma rotación tras rotación al quaternion el cual va cambiando su vector de rotación y el valor del ángulo. Esto es importante ya que las matemáticas de quaterniones son más simples que el calcular rotación de matrices y multiplicación de matrices para calcular transformaciones y requieren de menos cálculos, logrando un movimiento de cámara más suave. También

se utilizan los quaterniones en otras áreas, como por ejemplo la animación de esqueletos, cinemática inversa, y físicas 3D.