Universidad Católica de Asunción

Facultad de Ciencias y Tecnología

T.A.I. 2

Profesor: Juan de Urraza

Trabajo Práctico

Javier Bordón Giunta

Año 2006

DirectX 10 con énfasis sobre el 3d

La nueva API, que ha sido desarrollada desde cero, tendrá una relación mucho más estrecha con el Sistema Operativo (S.O.) y podremos ver como más de una aplicación, al mismo tiempo, hace uso de las características 3D de la tarjeta (hasta ahora imposible) y será muy raro ver al SO sufrir bajadas en el rendimiento por culpa del driver de la tarjeta de Video. Además, veremos como podremos cambiar al momento el driver de video y podremos resetear el estado de la tarjeta sin que eso afecte al SO, ni haya que reiniciarlo.

Toda esta estabilidad la da la nueva tecnología llamada Windows Display Driver Model (WDDM) que reemplazará al modelo de driver de video en Win XP y de la que hablaremos un poco más tarde.

Este nuevo modelo y la estrecha relación de DirectX 10 con el SO, hace del todo imposible que Windows XP pueda ejecutar DirectX 10 y por tanto será exclusivo de Windows Vista (además del componente de marketing que tiene la exclusividad en Win Vista y la necesidad que crea en el usuario de actualizarse a Vista).
Veremos también, como la compatibilidad hacia atrás de DirectX 10 con DirectX 9 no será directa y se hará a través de un emulador, esta versión se llamará DirectX 9L.

El nuevo pipeline

Si vemos dentro del Pipeline de Direct3D podríamos ver como en un mismo momento, tenemos al Vertex Shader con unidades de ejecución vacías, mientras que el Píxel Shader esta saturado de trabajo o viceversa.

La solución más obvia a este problema es la Unificación de los Shader, es decir, si tenemos a partir de este momento una unidad de ejecución que puede adaptarse a las necesidades puntuales, en vez de tener un estricto Path, es de suponer que el rendimiento final será mayor.

El problema que supone la *genericidad* no es pequeño, es mucho más eficiente hacer unidades de ejecución de propósito específico que de propósito general, además es más barato, es más flexible a subidas en la frecuencia del reloj y supone un reto tecnológico mucho menor, por ello se explica que hasta ahora no se haya afrontado el problema, de hecho, NVIDIA no ha seguido este camino y continúa con el Path fijo (pues no está del todo claro que sea peor), al contrario que ATI que ya tiene desarrollado el primer producto con Shaders unificados (la tarjeta gráfica de la XBOX 360). ATI sacará al mercado las tarjetas con nombre R600, que serán las que tengan Shaders unificados.

Qué hace este nuevo Shader:

El Vertex Shader tradicional, toma un solo vértice a la entrada y debe tener un solo vértice (tratado) a la salida, es imposible para el Vertex Shader crear o destruir triángulos. El Geometry Shader, sin embargo, permite operar sobre primitivas geométricas enteras, esto es líneas, triángulos y puntos, y a su vez permite operar sobre las primitivas vecinas. El Geometry Shader además, puede crear nuevas primitivas, nuevos triángulos, antes de enviarlos por el Pipeline.

Incluso es posible que el Geometry Shader envíe los resultados de nuevo a memoria, permitiendo que los datos vuelvan al inicio del Pipeline sin tener que pasar por la CPU, con ello se podrá acelerar muchísimo los sistemas de partículas como humo o explosiones que normalmente cargan mucho la CPU, siendo de esta forma independiente este efecto de la CPU

Veamos como:

Para hacer que un objeto con una textura de metal refleje el mundo que tiene alrededor, se debe determinar qué es lo que rodea a este objeto y mapearlo encima. Normalmente, esta tarea lleva hasta seis pasadas, pero con el Geometry Shader junto con un Array de Texturas podemos hacer el mismo efecto en una sola pasada, es decir estamos dividiendo por seis el tiempo que tardamos en hacer este efecto tan común (agua, objetos metálicos, cristal… etc).

Este nuevo modelo de Shader (Shader Model 4.0) va a permitir una gran cantidad de efectos que antes no eran posibles, cosas como *Displacement Mapping*, *Stencil Shadow Extrusion*, *Motion Blur* y muchos mas.

Cada uno de estos efectos va a mejorar muchísimo la calidad final de cada una de las escenas. Encontrándonos con unas texturas que van a rozar lo real (ver imágenes de Crysis). Además DirectX 10 trae características de morphing, así como transformaciones de personajes en cosas, cambiando de forma mucho más a menudo y con mucho menor coste. Por otro lado, también se espera que la GPU se encargue de parte de la física gracias a DirectX 10, lo que significará una mayor cantidad de objetos interactuando entre ellos. ATI, de hecho, mostró una demo en la que se veía un océano cuya física era tratada enteramente por la GPU

Comparación

Feature	1.1 2001	2.0 2002	3.0 2004†	4.0 2006
instruction slots	128	256	≥512	≥64K
	4+8‡	32+64‡	≥512	
constant regis-	≥96	≥256	≥256	16x4096
ters	8	32	224	
tmp registers	12	12	32	4096
	2	12	32	
input registers	16	16	16	16
	4+2§	8+2§	10	32
render targets	1	4	4	8
samplers	8	16	16	16
textures			4	128
	8	16	16	
2D tex size			2Kx2K	8Kx8K
integer ops				✓
load op				✓
sample offsets				✓
transcendental	✓	✓	✓	✓
ops		✓	✓	
derivative op			✓	✓
flow control		static	stat/dyn	dynamic
			stat/dyn	

Table 1: Shader model feature comparison summary.

Además de todo eso, por hacernos una idea de los cambios, en Shader Model 3.0 en número máximo de instrucciones por cada Shader era de 512, ahora será de **64000,** el número de registros temporales (algo muy importante para los programadores) pasará de 32 a **4096.** Además se pasará de cuatro a tener hasta **ocho objetivos de renderizado** y el **número de texturas** disponibles para un mismo Shader pasa de 16 a **128**, el **tamaño** de textura más grande pasa de 2048x2048 a **8192x8192**. Por último no se soportará en adelante puntos flotantes de 16Bits y serán todos de 32Bits (FP16 a **FP32**).

Cubemap

Se podrá usar también como una herramienta muy poderosa para los “cube mappings”. Un “Cube Mapping” se utilizan para que la textura de un objeto refleje el mundo que hay a su alrededor. Pues bien, el Geometry Shader junto con un array de texturas puede acelerar este efecto.

Procesamiento de Sombras

La siguiente imagen nos muestra coloreada en verde la proyección que hace una figura para generar la sombra, característica que evidentemente cambia según el origen de la luz. En DirectX 9 la sombra es generada en el CPU, reduciendo el rendimiento del sistema en conjunto con cada nuevo personaje con sombra que aparezca en la imagen. En DirectX 10, la sombra puede ser generada y rendeada completamente en la GPU aprovechando su capacidad de hacer cálculos en paralelos con enorme potencia.

Escenas más complejas

En estas imágenes podemos ver ejemplos del efecto que supone toda esta cantidad de herramientas nuevas en DirectX 10, frente a DirectX 9, por ejemplo, las texturas del agua (sistema de partículas, mayor número de triángulos, morphing de triángulos, física avanzada), los árboles (mayor numero de triángulos, luz entre los objetos), las nubes (Texturas volumétricas, luz entre objetos) suponen un salto como no recuerdo otros en cuanto a gráficos se refiere.

WDDM

Básicamente los controladores permitirán más fluidez en el momento de modificar los elementos a renderizar por pantalla y el cambio a otro tipo de elementos o situaciones. Windows Vista tendrá un entorno o escritorio 3D y además potenciará el uso de aplicaciones multitarea real, gracias a los procesadores de más de un núcleo, por lo tanto entendemos que quieren que los cambios de aplicación o pantalla sean lo más rápido posible. También quieren que la tarjeta cuando no encuentre un dato, no se quede esperando mientras recibe respuesta del S.O. sino que proceda con la siguiente tarea y luego cuando el dato esté disponible pueda reemprender el trabajo que estaba haciendo.

Conclusión

La evolución del API no es una simple medida de fuerza bruta, sino que hay diferencias estructurales de por medio. No sólo cambia "cuánto se hace" sino que "qué se hace" y "dónde se hace" también. Esta reubicación y reorganización de la cadena productiva permitirá salvar obstáculos que estaban estrangulando el desarrollo de los juegos e impedían dar el salto al siguiente nivel.
Lo que Directx10 trae al PC es un kit de herramientas que antes no estaban disponibles en Directx9, eso nos dará juegos mas completos en PC que en consolas, por el simple hecho de que es mucho más fácil hacerlo en PC, pues ya ha habido alguien que ha trabajado para ellos, simplificando muchísimo la tarea (Microsoft y su Directx 10).

Investigación.

Talismán y Fahrenheit
Talismán era el nombre de una arquitectura de chips gráficos desarrollada conjuntamente por Microsoft, Silicon Engineering, Samsung Semiconductor, Philips Microelectronics, Cirrus Logia y Fujitsu Microeletronics, en 1996.
La arquitectura de hardware estaba diseñada para encajar con el API de DirectX/Direct3D de Microsoft.
Se suponía que debía ofrecer gráficos 2D y 3D fotorealisticos, aceleración grafica de Windows, alta resolución en la reproducción de archivos mpeg-2, videoconferencia, sonido, etc., todo esto a un costo inferior a US$ 300.
Su objetivo principal era lidiar con las limitaciones de ancho de banda de ese momento y la necesidad de mucha memoria de los chips gráficos.
Samsung fue una de las primeras compañías en lanzar al mercado un chip basado en la arquitectura Talismán.
La primera implementación de esta arquitectura de hardware conocida como "Escalante".
En lugar de renderizar la escena completa en cada frame, se renderizaba solo porciones o capas de imágenes (elementos de escena) para formar la nueva imagen, basándose en la escena anterior.
Utilizaba el concepto de "chunking", que consiste en dividir las imágenes en partes más pequeñas.
El proyecto finalmente murió debido a dos factores principales. Primero, un problema bastante grande era el costo. El costo de algunos componentes para construir el hardware ya llegaba por si solos a alrededor de US$ 300, lo cual incrementaría en gran medida los precios finales.
Segundo, el mercado cambio de manera desfavorable durante el transcurso del proyecto. Mientras Microsoft se tomaba tiempo en desarrollar tecnología que requiera un poco de ancho de banda y poca memoria, el ancho de banda iba incrementando, y los precios de memoria bajaban.
Por estas razones, la mayoría de los usuarios prefería tarjetas graficas de menor costo como las ATI, Orchid, Number Nine, Diamond Multimedia.

Fahrenheit
Fahrenheit fue un esfuerzo de Microsoft y SGI de crear una API de alto nivel para gráficos 3D. El objetivo era crear un estándar que unifique las interfaces de DirectX y OpenGL a la hora de programar.
A través de los mediados de los 90`, SGI estuvo trabajando en una seria de APIs de alto nivel para facilitar la programación con OpenGL. En 1996 surgió OpenGL++, un API en C++ para OpenGL. Luego surgió una versión modificada de API para que pueda ser utilizado con OpenGL y DirectX de Microsoft.
Surgió en 1997, como un "acuerdo de paz" entre Microsoft y SGI para terminar con su "guerra de APIs", en la cual luchaban para dominar las interfaces de programación.
Solo una parte de Fahrenheit (XSG) fue lanzada, y duro muy poco tiempo.
SGI debía proveer la "parte media" del API, utilizada en la mayoría de las aplicaciones, llamada Fahrenheit Scene Graph, así como también una versión modificada para trabajar con modelos mas grandes de las aplicaciones CAD, Fahrenheit Large Model.
Microsoft estaba encargada de desarrollar la parte de más bajo nivel del API, Fahrenheit Low Level, utilizada por la mayoría de los usuarios y programadores que consistía esencialmente en un reemplazo del Direct3D.

El proyecto se debería lanzar a fines de 1999 o comienzos de 2000
En 1999, SGI se dio cuenta de que Microsoft no pretendía lanzar aun su parte, Microsoft estaba trabajando oficialmente en el proyecto, pero sin embargo, avanzando muy poco. Además, Microsoft estaba en proceso de invertir mucho dinero en DirectX 7.0. Sin la parte de Microsoft, Fahrenheit no podía funcionar.
En 2000, DirectX 7.0 fue lanzada y se convirtió en la principal interfaz 3D de los juegos en ese momento.
Mas adelante se lanzo Scene Graph, conocido como XSG. Sin embargo, esta fue la única versión del producto y luego desapareció.
Luego SGI se mudo a la plataforma Linux.
Todas las páginas Web relacionadas con Fahrenheit o XSG en los sitios de Microsoft y SGI desaparecieron desde entonces.

Lanzada la segunda versión de la API grafica OpenGL
Incluye compatibilidad con las versiones anteriores entre sus principales novedades.
OpenGL (Open Graphics Library) es un estándar dentro de la industria de la informática de API (Aplication Program Interface) para la creación de imágenes en dos y tres dimensiones. Las grandes ventajas de OpenGL son, entre otras, que ofrece una interfaz de programación unificada para sistemas operativos y plataformas diferentes (Mac OS X, Windows, Linux,….), y que permite abstraer el hardware concreto (siempre y cuando éste ofrezca soporte para OpenGL) a favor de la programación pura y dura.
Antes de la aparición de OpenGL, los programadores de aplicaciones graficas tenían que implementar manualmente el código necesario para que su programa pudiera trabajar con cada plataforma, sistema operativo y hardware diferente, por lo que este conjunto de librerías supone un gran avance en lo que a ahorro de esfuerzo y flexibilidad se refiere.
Su segunda versión es en realidad la séptima versión desde el lanzamiento de la versión 1.0 y presenta como principales novedades las siguientes:

Compatibilidad hacia atrás. Necesaria en casi cualquier desarrollo, significa que los programas escritos para funcionar con las versiones anteriores de la especificación 1.0, 1.1, 1.4,… funcionaran también a la perfección en la nueva versión 2.0
“Shading” programable
Múltiples objetos renderizables simultáneamente

- Microsoft Win32 API
- Sun J2EE APIs
- API for SCSI (Small Computer System Interface) device interfacing
- The Carbon APIs for the Macintosh OS
  Carbon is APPLE Computer`s procedural API for the Macintosh operating system, which permits a good degree of forward or backward Compatibility between source code written to run on the older and now dated MAC OS X, and the newer MAC OS X. It is one of five major APIs available for MAC OS X; in the other Cocoa, Toolbox (for the classic environment), POSIX (for the BSD environment), and Java
  (Environments such as a Perl and Python are considered minor environments because they not generally used for full-fledged application programming)
- Common Object Request Broker Architecture (CORBA)
  CORBA fue definido y esta controlado por el Object Management Group (OMG) que defina las APIs, el protocolo de comunicaciones y los mecanismos

necesarios para permitir la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida.

- Javascript-C de Mozilla Spidermonkey
- Google Web APIs (beta)
- Flickr API documentation
- libSDL

  Simple DirectMedia Layer (SDL) is a cross-platform multimedia library written in C that creates an abstraction over various platforms graphics, sound, and input APIs, allowing a developer to write a computer game or other multimedia application once and run it on many operating systems including GNU/LINUX, Windows and MAC OS X. It manages video, events, digital audio, CD-ROM, sound threads, shared object loading, networking and timers.
- Allegro

  Allegro is a free software / open source software library for video game development, with functions for basic 2D graphics, image manipulation, text output, audio output, midi music, input and timers, as well as additional routines for things like fixed point and floating point matrix arithmetic, Unicode strings, file system access, file manipulation, data files, and (limited, software only) 3D graphics.

  As of version 4.0, programs that use the library work on DOS, Microsoft Windows, BeOS, MAC OS X, and various Unix-like system with (or without) X Window System, abstracting their application programming interfaces (APIs) into one portable interface.
- OpenML

  Open Media Library (OpenML) is a free, cross-platform programming environment designed by the Khronos Group for capturing, transporting, processing, displaying, and synchronizing digital media (2D and 3D graphics, audio and video processing, I/O, and networking)