**Breve introducción**

DirectX es un conjunto de APIs pensadas para manejar tareas multimedia en plataformas Microsoft, especialmente el desarrollo de videojuegos, y la visualización de datos y video.

Para crear aplicaciones que puedan usar las APIs de DirectX se utiliza el DirectX SDK, un conjunto de librerías y cabeceras, en formato binario redistribuible.

**Un poco de historia**

En 1994, ya a las puertas del lanzamiento del sistema operativo Windows 95, los programadores de videojuegos seguían viendo como una mejor opción desarrollar los juegos para sistemas operativos MS-DOS anteriores, ya que lo veían como una plataforma más estable.

Para mayor agravio, durante las navidades de 1994, Windows lanzó el juego del Rey León para ordenador. Por aquel entonces, los sistemas operativos Windows usaban WinG como API para gestionar el entorno gráfico. También por aquellas fechas, COMPAQ lanzó su modelo de ordenador Presario, con el detalle de que no se habían testeado los drivers de gráficos del computador con la API de WinG. El resultado: un sinfín de BSoD y marabunta de clientes enfurecidos.

Además, DOS contaba con acceso a tarjetas de video, tarjetas de sonido, ratones y teclados de forma mucho menos restrictiva de lo que se planteaba en el modelo de Windows 95. A pocos meses de la salida de Windows 95 al mercado, Microsoft tenía que encontrar una salida al problema, que hiciera a los programadores de videojuegos ver Windows 95 como una opción viable para programar su software.

A finales de 1994, un grupo de 3 ingenieros, decidieron realizar una conferencia con algunos de los principales desarrolladores de videojuegos del sector. A partir de esa reunión, sacaron algunas de las principales ideas para el futuro DirectX, y así empezaron a ganarse de nuevo la confianza de las desarrolladoras, en ese punto casi nula. Los ingenieros pretendían tener la API lista para la conferencia de desarrolladores de videojuegos (precursor de la famosa E3), de abril. Tras cuatro meses de trabajo frenético, y a 2 horas de la conferencia de presentación, llegaron los CD con la versión beta del software. Así, nació DirectX.

La primera versión de DirectX se llamó Windows Games SDK (software developement kit), y estaba pensada para poder hacer uso máximo de la tecnología hardware de las tarjetas gráficas en juegos “high end” del momento. Además, DirectX debía servir para crear una serie de directrices estándar que unieran a desarrolladores de software y hardware, para evitar que se repitieran escenarios como los del rey león.

DirectX 1.0 salió en septiembre de 1995, y resultó un éxito. Las siguientes dos versiones de DirectX salieron en menos de un año. DirectX 2.0 salió el e 5 de junio de 1996 sin mayores inconvenientes, pero no pasó lo mismo con DirectX 3.0.

Microsoft, que no es feliz si no es Microsoft, hizo de las suyas. El equipo de DirectX quería importar OpenGL de Windows NT a Windows 95 para tratar el tema de gráficos 3D, pero por temas de políticas internas, se les denegó. Así que los ingenieros de DirectX tuvieron que tirar de un software llamado RenderMorphics para crear Direct3D. De este modo, el 15 de septiembre de 1996 sale al mercado DirectX 3.0.

DirectX 4.0 pasó sin pena ni gloria, y DirectX 5.0, ya para Windows 98, incluyo algunas mejoras sustanciales adaptadas al nuevo sistema operativo, como por ejemplo un panel de control propio. DirectX 6.0, para el mismo S.O. y para Windows SE, incluyó nuevas texturas 3D y una nueva API: DirectMusic, permitiendo a DirectX controlar el audio también Las versiones 7 y 8 incluyeron también mejoras en el apartado de gráficos 3D y de audio, y así llegamos al 2001, y a DirectX 8.1, o lo que es lo mismo, el nacimiento del hermano pequeño del PC, XBOX.

Una anécdota interesante acerca del nacimiento de XBOX. El proyecto inicial de consola nació en el 2000, de la mano de 4 ingenieros en las oficinas de Microsoft, mientras desmontaban portátiles DELL, a partir de los cuales crearon un prototipo de consola, que trabajaba con el, por aquél entonces nuevo, DirectX 8.

En 2001, con la ayuda de NVIDIA, se desarrolla una API específica para XBOX a partir de la versión 8.1 de DirectX, cosa que da pie a la rama de API de DirectX para consolas.

Hasta la fecha, y a partir de la versión 9.0 de DirectX, sobretodo se han centrado en mayormente en la gestión de recursos del computador y, la renderización, de gráficos 2D y 3D.

Formada por las siguientes APIs:

* Direct2D
* Direct3D
* DirectWrite
* DirectCompute
* DirectDraw
* DirectInput
* DirectPlay
* DirectShow
* DirectSound
* DirectSound3D
* DirectMusic
* XAudio2
* DirectX Media
* DXGI
* XACT3

**Tessellation**

Página 7:

Todos los objetos que componen los videojuegos están compuestos unidades llamadas polígonos. La teselación básicamente significa coger un polígono y romperlo en pedazos más pequeños. ¿Cómo resulta útil esto? Bien, permite a los computadores generar mayor detalle en los objetos, usando piezas más pequeñas (por ejemplo, en la generación de curvas). Esta técnica resulta especialmente útil a la hora de crear profundidad y altura.

Con la teselación, los desarrolladores pueden aplicar mapas de desplazamiento (básicamente, son mapas con información sobre la altura que tienen los objetos, a.k.a volumen) a los polígonos existentes, y la teselación calculará y generará el nivel de profundidad adecuado, dando mucho más realismo a los objetos. La teselación también ayuda a crear objetos más suaves y definidos.

Pero, ¿cómo funciona realmente? La teselación funciona con triángulos. Los polígonos mencionados más arriba, se componen de la unión de dos triángulos formando estructuras cuadradas o romboides, llamadas “quads”. La teselación permite duplicar el número de triángulos que conforman estos quads.

Página 8

En DirectX 11, se añaden 3 nuevas fases al proceso de desarrollo del pipeline: hull shader, domain shader, i tessellator stage. Las dos primeras son enteramente programables (codificables) mientras que la última es configurable.

El pipeline opera con los “meshes” u objetos formados por cuadrados comentados anteriormente (los famosos “quads”). Para cada uno de estos cuadrados, se llama al hull shader, usando los vértices del cuadrado como puntos de control. El shader tiene dos funciones básicas:

* La primera, opcional, es convertir los puntos de control de representación básica a otra representación.
* La segunda es computar los factores de teselación, que se pasarán a la fase de teselación (tessellation). Los factores de teselación permitirán dividir la figura en múltiples triángulos en la siguiente fase. El factor de teselación va en un rango de 2 a 64 actualmente, lo que permite dividir cada triángulo de 2 hasta un máximo de 64 veces.

El tessellator es una función fija, pero altamente configurable, que utiliza los factores de teselación para dividir los triángulos. El tessellator no tiene acceso a los puntos de control, así que todas las decisiones de teselación se toman basándose en la configuración de la función y en los factores de teselación enviados en el paso anterior. Cada vértice nuevo obtenido en este paso, se envía al domain shader, junto con la parametrización del propio quad.

El domain shader opera de forma individual para cada uno de los vértices que se han enviado, usando los parámetros del quad enviados junto con él, aunque también tiene acceso a los puntos de control transformados. El domain shader envía toda la información relativa al vértice (posición, textura, coordenadas, etc.) al geometry shader, evaluando la representación de la superficie para cada uno de los vértices. Esto es lo que permite dar volumen a objetos 3D.