Investigación Shaders

Damian Eduardo Burboa Almada 189157

Ciudad Obregon, Sonora

Shaders. Sin entrar en terrenos de difícil comprensión, los shaders son pequeños programas que se encargan del procesamiento de vértices (Vertex shaders) y de pixeles (Píxel shaders). La principal ventaja es que, como su naturaleza lo indica, pueden ser programados por el desarrollador, otorgando una flexibilidad que hasta antes de la aparición de los shaders era poco más que impensada. Son utilizados para realizar transformaciones y crear efectos especiales, como por ejemplo iluminación, fuego o niebla. Los shaders pueden programarse tanto en el lenguaje ensamblador nativo de la tarjeta gráfica (opción obsoleta) o en un lenguaje de alto nivel, como son el HLSL, GLSL o el Cg. Desde que comenzó la “revolución 3D” en el ámbito de los juegos de computadora, allá por mediados de la década de los 90’, la tendencia de la tecnología aplicada a este rubro ha sido trasladar el trabajo de procesamiento de gráficos tridimensionales, desde la CPU hacia la tarjeta de video.

En primer lugar, fue el filtro de las texturas, para lo cual se crearon chips especialmente dedicados para realizar esta tarea. Así nacieron las famosas placas aceleradoras 3D, que incorporaban dichos chips y una cantidad de memoria propia en la misma tarjeta. Luego, con la salida del GeForce 256 de NVIDIA, el procesador gráfico pasó a encargarse de lo que, hasta ese momento, realizaba la CPU. Estamos hablando de la función de Transformación e Iluminación (Transform & Lighting), utilizada para llevar a cabo los cálculos de geometría y de iluminación general de una escena en 3D. Hubo una versión mejorada de este motor, a la que se llamó de Segunda Generación. Ésta vino incluida a partir de la GeForce 2 y la gama Radeon de ATI, avanzando un poco más en cuanto a materia gráfica.

El gran cambio se dio a partir de la incorporación de los Píxel shaders y Vertex shaders. Esto permitió a los programadores una mayor libertad a la hora de diseñar gráficos en tres dimensiones, ya que puede tratarse a cada píxel y cada vértice por separado. De esta manera, los efectos especiales y de iluminación puede crearse mucho más detalladamente, sucediendo lo mismo con la geometría de los objetos. Un vertex shader es una función que recibe como parámetro un vértice. Sólo trabaja con un vértice a la vez, y no puede eliminarlo, sólo transformarlo. Para ello, modifica propiedades del mismo para que repercutan en la geometría del objeto al que pertenece. Con ésto se puede lograr ciertos efectos específicos, como los que tienen que ver con la deformación en tiempo real de un elemento; por ejemplo, el movimiento de una ola. Donde toma una gran importancia es en el tratamiento de las superficies curvas, y su avance se vio reflejado en los videojuegos más avanzados de la actualidad. Particularmente, en el diseño de los personajes y sus expresiones corporales. Por otro lado, un Pixel Shader no interviene en el proceso de la definición del "esqueleto" de la escena (conjunto de vértice, wireframe), sino que forma parte de la segunda etapa: la rasterización o render (paso a 2D del universo 3D). Allí es donde se aplican las texturas y se tratan los píxeles que forman parte de ellas. Básicamente, un Pixel Shader especifica el color de un píxel.

Este tratamiento individual de los píxeles permite que se realicen cálculos principalmente relacionados con la iluminación en tiempo real, con la posibilidad de iluminar cada pixel por separado. Así es como se lograron crear los fabulosos efectos de este estilo que se pueden apreciar en videojuegos como Doom 3, Far Cry y Half Life 2, por mencionar sólo los más conocidos. La particularidad de los píxel shaders es que, a diferencia de los vertex shaders, requieren de un soporte de hardware compatible. En otras palabras, un juego programado para hacer uso de píxel shaders requiere si o si de una tarjeta de video con capacidad para manipularlos.

Hay distintos lenguajes de programación de alto nivel para ello. De alto nivel significa que tú escribes algo sencillo y el lenguaje, compilador en realidad, ya se encarga de pelearse con el hardware para que haga lo que tú quieres. En este caso las tarjetas gráficas poseen montones de núcleos (pipelines) que ejecutan las tareas que les indicas. Cada uno de esos núcleos ejecutará un shader, pudiendo ser de distinto tipo, como la iluminación, el sombreado, el reflejo en el agua, la niebla que clarea un píxel más cuanto más lejos esté de la cámara, etc.

Shaders por hardware
Este tipo de shaders se divide en tres tipos:

- Vertex shader: Sirve para aplicar transformaciones a un vértice en la malla de un objeto. Lo más básico es que muevas el objeto y entonces los vértices de ese objeto se muevan también, pudiendo rotarse. También sirven para transformar los objetos en otro, haciendo que el pelo de Goku se ponga de punta cuando se transforme en SSJ por ejemplo. Son los primeros en ejecutarse.
- Geometry shader: Son de los más modernos, requiriendo al menos DX10, por ejemplo. Su función es recibir como entrada los vértices y crear las líneas que los enlaza, pudiendo recibir un octógono y convertirlo en un círculo creando más puntos intermedios, aunque para eso podría haber recibido un círculo directamente, pero bueno.
- Fragment shader: Recibe cada uno de los fragmentos generados en el paso anterior e interpola (hace una media) entre los vértices para calcular los colores y sombras que mencionaba antes.

La versión del Shader Model es importante en las tarjetas, porque indica qué tipo de órdenes se le pueden dar a la tarjeta. Así, en la versión 1.0 le ordenas sentarse al perro y en la 2.0 le pides que se tumbe. Con cada nueva versión aparecen nuevas características que facilitan la aceleración por hardware de esas funciones.

Mostrar un juego moderno donde tenemos un mundo tridimensional no es algo sencillo computacionalmente hablando. Piensa que el equipo tiene que permitirte moverte e interactuar con los objetos de un universo convirtiéndolo en algo realmente complejo.

Al principio, los primeros juegos tridimensionales necesitaban sólo del procesador para funcionar, pero es sólo con la llegada de las tarjetas gráficas modernas cuando podemos realmente sumergirnos en ellos ya que estas se encargan de gran parte del trabajo que requiere gestionar este universo. El funcionamiento a la hora de crear estos mundos por parte de estos dispositivos es muy parecido, da igual de que marca o tarjeta estemos hablando. Simplificando mucho: **Se seleccionan los objetos.** Los transformamos para colocarlos en el lugar y en la posición relativa al jugador. Puedes imaginarte la cantidad de cálculos que lleva sólo este paso.

Se convierten a dos dimensiones. El monitor es plano, luego necesitamos pasar de tres a dos dimensiones. De esta forma sabremos donde va cada objeto y si alguno no va a aparecer en pantalla. Estos últimos se descartan- **Se ordenan mostrando los que estén más cerca.** El solapamiento entre objetos se calcula a nivel de pixeles lo cual hace que sea lo más preciso posible. Una vez que sabemos cuál es el punto de un objeto más cercano al observador ya podemos trabajar sobre él. Normalmente los objetos se definen como mallas de triángulos lo cual hace más sencillo trabajar con ellos.

Los Pixel Shaders entran en juego en este momento. Su tarea es calcular entre otras cosas el color y sombras de un determinado pixel. Algunos efectos típicos que se realizan gracias a este elemento son: **Bump mapping.** Son texturas con relieves. Gracias al sombreado puedes hacer que una roca, por ejemplo, no parezca totalmente plana. De esta forma se simulan efectos muy complejos. La teselación, que aparece como una técnica relacionada con DirectX 11, realiza algo parecido pero de una forma mucho más avanzada. **Sombras.** El Pixel Shader se encarga de comprobar cuando cantidad de luz incide sobre el objeto que estamos calculando. De esta forma se crean las sombras. Esto se puede complicar todo lo que se quiera ya que existen objetos que pueden proyectar sombras más o menos precisas o incluso de colores. **Luces.** No todas las luces son iguales. No es lo mismo el sol, o una bombilla, por poner dos ejemplos. Un determinado objeto puede recibir a la vez varios tipos de estas y hay que calcular como incide sobre el pixel que estamos calculando. **Niebla.** Un efecto muy usado es la niebla. El Pixel Shader también se encarga de calcular el nivel de niebla que ves en cada pixel de la pantalla. Está relacionado con la distancia de cada punto al jugador. **Transparencia.** No todos los objetos son opacos y a veces es necesario saber que hay detrás del objeto para combinar el color de las diferentes texturas.

Espejo. Crear un espejo no es sencillo ya que requiere de muchos cálculos, hay que saber que imágenes se proyectan sobre él. El pixel Shader también se encarga de llevar

a cabo este efecto. Puede realizar ciertos efectos de post procesado. Se llaman así porque no se trabaja con los objetos en 3D sino ya en 2D. En este caso trabaja con una imagen bidimensional ya formada: **Emborronamiento.** Por ejemplo para hacer ciertas partes de la escena menos visible. Lo puedes ver en ciertos juegos cuando simulan que estas borracho o drogado. **Creación de halos.** Para luces o proyectiles que emiten luz. **Detección de bordes.** Para crear efecto de dibujos animados.

Para todo esto las tarjetas gráficas modernas permiten que se les escriba un programa o algoritmo que se encarga de ello. No todos los juegos necesitan por tanto implementar todas las técnicas ni hacerlo de la misma manera. La versión de Pixel Shader indica que instrucciones puede usar el programador. Si un juego necesita, pongamos por ejemplo la versión 4.1, no podrás jugar con él si tu tarjeta sólo es capaz de entender hasta la 3. Por suerte la compatibilidad hacia atrás existe y teniendo la última versión podrás jugar a toda clase de juegos.

En cada versión los fabricantes aprovechan para añadir características más avanzadas. La versatilidad que se consigue es total. Se deja todo en manos del programador, su habilidad, su capacidad y su imaginación. Ten en cuenta que por ejemplo efectos sólo de luces pueden existir cientos. De esta forma, todo este procesado es acelerado por el hardware. Así se hace de manera mucho más rápida, pudiendo crear efectos más realistas, y descargamos al procesador de este trabajo para que así los juegos y el sistema funcionen de manera más fluida.

En el campo de los gráficos por ordenador, un sombreado es un programa informático que se utiliza para hacer sombreado: la producción de niveles adecuados de color dentro de una imagen, o en la era moderna, también para producir efectos especiales o hacer post-procesamiento de vídeo. Una definición en términos de laicos podría ser dada como "un programa que enseña a un ordenador cómo dibujar algo de una manera específica y única". Los shaders calculan los efectos de renderizado en hardware gráfico con un alto grado de flexibilidad. La mayoría de los shaders están codificados para una unidad de procesamiento gráfico, aunque esto no es un requisito estricto. Los lenguajes de sombreado se usan generalmente para programar la canalización de procesamiento de GPU programable, que ha reemplazado en su mayor parte la tubería de función fija que sólo permitió la transformación de geometría común y las funciones de sombreado de píxeles; Con shaders, se pueden usar efectos personalizados. La posición, el matiz, la saturación, el brillo y el contraste de todos los píxeles, vértices o texturas utilizados para construir una imagen final pueden ser alterados sobre la marcha utilizando algoritmos definidos en el sombreador y pueden ser modificados por variables externas o texturas introducidas por El programa que llama al sombreador.

- Jose V. (2014). Shader unificado en Unity. 2020, de Knowledge Sitio web: https://www.adrformacion.com/knowledge/ingenieria-y-proyectos/shader_unificado_en_unity.html
- Wilmer L. (2019). Shader Graph in Unity for Beginners. 2020, de Unity Tutorials Sitio web: https://www.raywenderlich.com/3744978-shader-graph-in-unity-for-beginners
- Will H.. (2017). How to write Shaders in Unity. 2020, de febucci Sitio web: https://www.febucci.com/2019/11/how-to-write-shaders-in-unity/
- Penny Byl. (2019). Shader Development from Scratch for Unity with Cg. 2020, de Udemy Sitio web: https://www.udemy.com/course/unity-shaders/?utm_source=adwords&utm_medium=udemyads&utm_campaign=Unity_v.PROF_la.EN_cc.ROW_ti.8154&utm_content=deal4584&utm_term=_._ag_81264460705_._ad_394028037598_._kw__._de_c_._dm__._pl__._ti_dsa-774930034729_._li_1010165_._pd__._&matchtype=b&gclid=CjwKCAiA1rPyBRAREiwA1UIy8OFHS4aS9eIkXKirXNDIMkwRWUR47jjL3x1OkR75qhvR5vN35lYjHRoClpUQAvD_BwE
- Unknown. (2016). Assets Shader. 2020, de Unity Sitio web: https://docs.unity3d.com/es/530/Manual/class-Shader.htmlUnknown. (2017)
- El Uso y rendimiento de shaders integrados. 2020, de Unity Sitio web: https://docs.unity3d.com/es/2018.4/Manual/shader-Performance.html
- Unknown. (2019). Shader Graph. 2020, de Unity Sitio web: https://unity.com/es/shader-graph
- Rombe F.. (2018). Shaders. 2020, de EcuRed Sitio web: https://www.ecured.cu/Shaders
- Konosoke. (2019). Shaders . 2020, de citriogames Sitio web: http://citrogamers.blogspot.com/2015/09/shaders-esos-que-todos-nombran-y-pocos.html
- Mario J.. (2012). ¿En qué consiste la teselación?. 2020, de aboutespanol Sitio web: https://www.aboutespanol.com/en-que-consiste-la-teselacion-841103

- Carlos V. (2013). Identificar la versión de Píxel Shader / DirectX de tu tarjeta gráfica. 2020, de CCM Sitio web: https://es.ccm.net/faq/12016-identificar-la-version-de-pixel-shader-directx-de-tu-tarjeta-grafica
- Francisco C. (2019). Tutorial sobre tipos de shaders. 2020, de Torre de babel Sitio web: https://fcharte.com/tutoriales/0003_tiposshader/