El mundo de las tarjetas gráficas

Laura Raya González

"La capacidad computacional de todo un centro de computación, al alcance de tus manos"

Las tarjetas gráficas y sus unidades de procesamiento gráfico (GPU), tal y como hoy en día se conocen, componen una pieza fundamental dentro del mundo de los gráficos. Fueron creadas para ofrecer rapidez en sus cálculos, capaces de conseguir una apariencia de realismo para las aplicaciones gráficas como películas de animación, simuladores o videojuegos, liberando de dichas tareas al procesador central (CPU), y, sin duda, van por buen camino.

Aunque estamos acostumbrados a oír hablar del tipo de tarjeta gráfica que tiene nuestro ordenador personal (*ATI, Nvidia, Intel…*), pocos se paran a pensar cómo funcionan y cuál es su diseño a nivel de arquitectura, ignorando que un apasionante mundo se esconde tras de ellas.

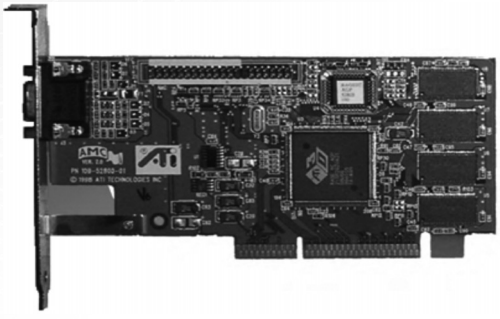

Figura 1. Fotografía de una tarjeta gráfica sin ventilador ni disipador [3].

Una tarjeta gráfica[1] es una tarjeta de expansión para un ordenador, encargada de procesar los datos que vienen de la CPU y transfor-

[1] También llamada tarjeta de vídeo o aceleradora.

marlos en información comprensible y representable en un dispositivo de salida, como un monitor o televisor. Está formada por múltiples componentes; entre ellos se encuentran:

- **La GPU**: como veremos en la siguiente sección es el componente clave de la tarjeta.
- **La memoria de vídeo**: es una memoria rápida que ha evolucionado mucho durante los últimos años. Permite a la tarjeta manejar toda la información visual que le manda la CPU del sistema. Su tamaño varía con el modelo de tarjeta. La memoria empleada en los últimos años está basada en tecnología DDR, destacando DDR2, GDDR3 y GDDR4.
- **RAMDAC:** es un conversor de señal digital a analógica. Se encarga de transformar las señales digitales producidas en el ordenador en una señal analógica que sea interpretable por el monitor.
- **Disipador:** conocido como dispositivo pasivo[2] compuesto de material conductor del calor, que extrae éste de la tarjeta. Su eficiencia va en función de la estructura y la superficie total, por lo que son bastante voluminosos.
- **Ventilador:** corresponde a un dispositivo activo que aleja el calor emanado de la tarjeta al mover el aire cercano. Es menos eficiente que un disipador y produce ruido al tener partes móviles.
- **Alimentación:** hasta hace unos años, la alimentación eléctrica de las tarjetas gráficas no había supuesto un gran problema, sin embargo, la tendencia actual de las nuevas tarjetas es consumir cada vez más energía. Aunque las fuentes de alimentación son cada día más potentes, el cuello de botella se encuentra en el puerto PCIe. Por este motivo, las tarjetas gráficas con un consumo superior al que puede suministrar el PCIe incluyen un conector (PCIe power connector) que permite una conexión directa entre la fuente de alimentación y la tarjeta, sin tener que pasar por la placa base.

Además, no es de extrañar que algunas de las tarjetas de última generación requieran fuente de alimentación propia.

¿Qué es una GPU?

Al igual que nuestro ordenador tiene un cerebro, conocido como CPU[3], que realiza el control de todo el ordenador, la tarjeta gráfica posee su propia unidad especial, llamada GPU[4]. No es más que un procesador dedicado exclusivamente al procesamiento de gráficos tridimensionales para aliviar la carga de trabajo del procesador de propósito general en aplicaciones como los videojuegos o las simulaciones. De esta manera, la CPU se puede dedicar a otro tipo de cálculos (como la inteligencia artificial o los cálculos físicos en el caso de los videojuegos).

Las principales compañías creadoras de GPUs son ATI (recientemente comprada por AMD) y Nvidia, por lo que los ejemplos expuestos a lo largo del trabajo serán en su mayoría de ambas. Aunque también existen modalidades de Intel, como, por ejemplo, algunas de las integradas en placa.

Mientras que una CPU se basa en la archiconocida arquitectura Von Neumann, una GPU tiene un diseño totalmente diferente conocido como *arquitectura streaming*. El diseño de las CPU hace que, para reducir el número de accesos a la memoria principal (debido a que es muy lento), el área del chip esté cubierta principalmente por una gran jerarquía de memoria. El objetivo de dicha jerarquía es conseguir una memoria de gran velocidad, pero provoca la limitación en el uso de un mayor número de transistores como unidades aritmético lógicas (ALU) y unidades de coma flotante (FPU) [1].

Figura 2. Fotografía de una GPU de la compañía ATI (izquierda) y de Nvidia (derecha).

Sin embargo, en la mayoría de GPUs tenemos un modelo circulante que hace que su arquitectura reduzca considerablemente el gasto energético necesario para que unas etapas y otras accedan a la información y, como consecuencia, el porcentaje dedicado a la computación aritmética aumenta.

---

[2] Se denomina pasivo porque no tiene partes móviles y, por tanto, es silencioso.
[3] Central Processing Unit.
[4] Graphics Processing Unit.

Hoy en día, la complejidad de una GPU es tal que puede rivalizar sin complejos con la de un procesador genérico. De hecho, se dice que la GPU crece en el volumen de cálculo que pueden procesar, a un ritmo tres veces más rápido que los procesadores generales. Basta decir que el chip G80 de Nvidia, utilizado en la familia de tarjetas gráficas de la serie 8, integra 681 millones de transistores.

Esta evolución es debida a que es un hardware gráfico de bajo coste, ya que, el crecimiento de la industria de los videojuegos ejerce una presión muy fuerte en el mercado, con muchos usuarios muy exigentes pero pequeños bolsillos.

De esta manera, se han conseguido altas prestaciones y que, desde hace unos años, puedan ser programadas con lenguajes de medio nivel al igual que las CPUs.

El procesador gráfico por dentro

Una GPU está altamente segmentada y posee gran cantidad de unidades funcionales. Gracias a esta segmentación se consigue una alta capacidad de procesamiento al aplicar ingeniosamente el paralelismo, que es la tendencia actual en arquitectura de procesadores para ordenadores personales. Las CPUs modernas incorporan varios núcleos de procesamiento; por ejemplo, el *Intel Core Duo* o el *Intel Quad*. Por otro lado, una GPU corriente (*GeForce 7800 GTX* o *GeForce 8800 GTX*), poseen 32 y 128 procesadores, respectivamente. Es importante recalcar que el concepto de núcleo del *Intel Core Duo*, por ejemplo, y el de una tarjeta, es sustancialmente diferente. Mientras que los núcleos de la CPU pueden trabajar de una manera independiente, los núcleos de una GPU dependen unos de otros. Sin embargo, queda patente la existencia del paralelismo en las arquitecturas actuales, característica esencial del procesador gráfico.

Estos procesadores se pueden dividir principalmente en dos: aquellos especializados en procesar vértices y los que procesan píxeles. Se establece el vértice y el píxel como los principales componentes que maneja la GPU y, por lo tanto, conceptos elementales con los que trabaja el cauce gráfico.

Podemos resumir de forma abstracta y breve el **cauce gráfico** y su relación con la arquitectura de una GPU. Las operaciones de renderizado[5] se aplican siempre en un mismo orden, comienzan por las que se encuentran relacionadas con el atributo de posición del vértice y finalizan con las que son más cercanas al atributo de color del píxel [1].

Figura 3. Esquema del cauce gráfico en la generación de imágenes sobre una GPU.

La CPU coge una lista precalculada con los vértices de todos los polígonos de la escena y la envía a la GPU. Así, inicialmente, a la GPU le llega la información de la CPU en forma de vértices, lo que evita enviar toda la información de la imagen, por lo que el tráfico en el bus (PCI o AGP) es mucho menor, en la mayoría de las veces, enviando únicamente los vértices, que si tuviéramos que enviar todas las características de la escena.

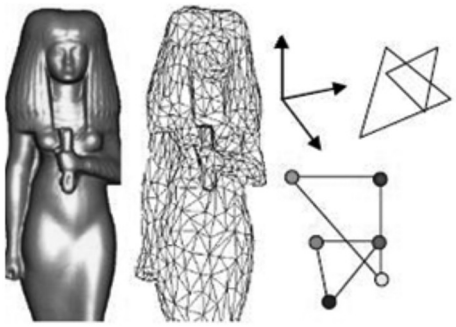

Figura 4. Imagen de la diosa ISIS formada por triangulación, para pasarle los vértices a la GPU.

Procesador de vértices

Después de una etapa de ordenación e instanciación de los vértices, el tratamiento que reciben los vértices se realiza en el **procesador de vértices**. Un **vértice** es la esquina de una primitiva (generalmente, un triángulo) donde se unen dos bordes que lo delimitan. Un **triángulo** tiene tres vértices y los objetos que conocemos hoy en día pueden formarse por miles de triángulos *(véase la figura 4)*. Así, con sólo definir tres vértices, se pueden generar todos los **píxeles** que constituyen, por ejemplo, el área de un triángulo, y obtener así las diversas primitivas de las *3D API* gráficas para dar forma a los objetos de la escena.

Un vértice lleva información sobre sus coordenadas, color, peso, textura, tamaño..., y con ellos se realizan transformaciones, como la traslación o la

[5] En términos de visualizaciones en ordenador, más específicamente en 3D, la "renderización" es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena 3D.

rotación de las figuras. El procesador cambia la posición de los vértices, lo que afectará a la posición en donde será dibujado el objeto al final. Una GPU puede tener varios procesadores de vértices; por ejemplo, la *Nvidia Geforce 7800GTX* tiene 8 y la *Nvidia GeForce 7950* contiene hasta 16 procesadores de vértices. Cuando en la caja de su tarjeta gráfica ve que le venden un número de *vertex shader* determinado, se refiere a este tipo de procesadores. Cuantos más procesadores disponga de este tipo, mayor trabajo en paralelo puede realizar su tarjeta gráfica, por lo que conseguirá un mayor rendimiento [3].

Esta etapa del cauce es programable desde los años 2000 y 2001, conocidos como **vertex shaders** los programas con dicho objetivo. Es decir, se trata de hardware programable[6]. Antes de disfrutar de las ventajas del hardware gráfico programable, en un código que usara alguna librería gráfica para el renderizado, había unas pautas muy fijas. Los datos seguían siempre el mismo camino, pasaban por los mismos sitios, se transformaban de la misma manera y los resultados que se obtenían, por lo tanto, tenían poca variedad (a pesar de que la API ofrece muchas posibilidades de configuración).

Con la introducción de los *vertex shaders*, se modifica el procesador de vértices de la tarjeta gráfica, de tal manera que el programa se convierte en un microcódigo que configura una serie de etapas. De esta manera, se puede utilizar la GPU para operaciones como *morphing*, *waves* y otras transformaciones procedurales a nivel de vértices. Es decir, se pueden conseguir los gráficos que hoy en día estamos acostumbrados a ver en películas y videojuegos.

Rasterizado

Tras la manipulación de los vértices, se define la parte de ellos que se va a visualizar (técnica llamada **clipping**[7]**)** y se eliminan los triángulos cuya cara no está orientada hacia la cámara (técnica denominada **culling**). Así, no se trabajará con aquellos objetos o partes de un objeto que en la escena final no vayan a ser visibles. Esto sucede en la etapa de **rasterización**, donde cada polígono es convertido a una serie de fragmentos del tamaño de un pixel. Este paso del cauce se encarga de convertir cada punto, línea o polígono 3D en una matriz 2D de puntos, donde se guarda información acerca del color, profundidad…, realizando una interpolación de cada uno de los puntos que lo conforman.

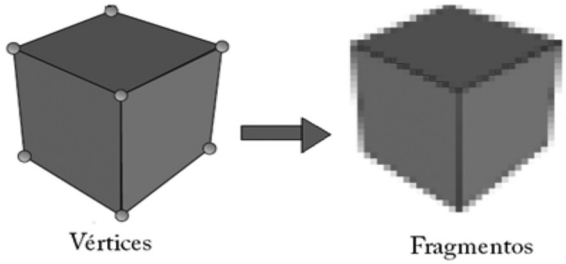

Figura 5. Proceso del rasterizado.

Procesado de fragmentos

El resultado de la etapa de rasterización es lo que le llega al procesador de fragmentos. Un fragmento es el conjunto de datos que se produce tras la rasterización de las primitivas (triángulos, puntos, cuadrados, etc.), como, por ejemplo, el color, la profundidad (Z), el dato del *stencil buffer*, etc.

En este hardware se realizan las transformaciones referentes a los fragmentos, tales como la aplicación de texturas o transformaciones sobre los fragmentos que, finalmente, constituyen el píxel a visualizar.

Esta etapa también es programable por medio de **fragment shaders**[8] que se convierten en el microcódigo correspondiente. Si bien es interesante poder manipular los vértices en la GPU, como se comentó anteriormente, lo que los programadores también querían era poder manipular los fragmentos (píxeles) a su antojo.

Las GPU actuales también cuentan con varias unidades, disponiendo más que de procesadores de vértices (la mayoría de los casos). Esto se debe a que en el cauce aumenta el volumen de datos a medida que lo recorremos (y recordemos que los procesadores de fragmentos se encuentran al final del mismo) y necesitamos más unidades en este punto para que haya un equilibrio y no se produzca un cuello de botella[9]. Por ejemplo, la Nvidia 7800GTX tiene 24 y la Nvidia 7950 posee 48 procesadores de píxeles.

---

[6] Es más correcto llamarlo configurable, en vez de programable.
[7] Consiste en la eliminación de cualquier polígono fuera del punto de vista de la cámara.
[8] También lo encontrará en otras referencias como pixel shader.
[9] Se llama cuello de botella cuando tenemos una fase de una cadena de producción más lenta que otras que ralentiza el proceso de producción global.

Finalmente, están las etapas de **ROP** y de **blending** que van componiendo la escena. También se pasan múltiples tests para eliminar aquellas cosas que queden ocultas por otros objetos, que no queden dentro del punto de vista de la cámara…

Si resumimos todo el proceso en un simple esquema, éste quedaría como en la figura 6:

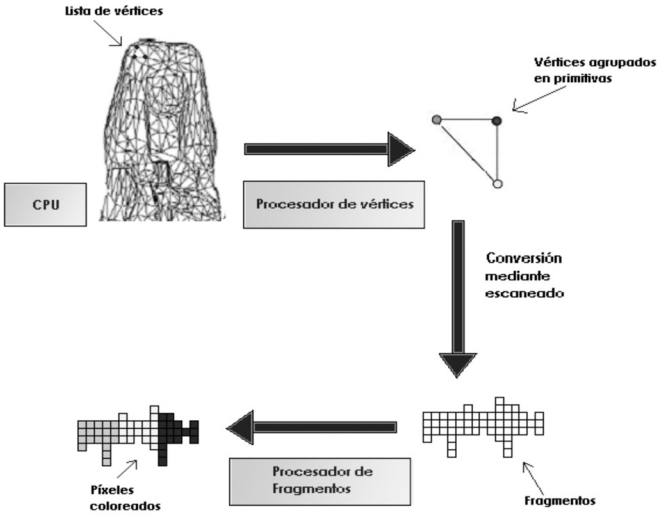

Figura 6. Resumen cauce gráfico en la GPU.

Programación en la GPU

Las GPUs han evolucionado tanto que se pueden programar[10] sobre ellas ciertos efectos, aumentando el rendimiento de nuestro proceso. Se llamarán *shaders* (también conocidos como sombreadores) a los programas que permiten crear código ejecutado en la GPU, pero hay que tener en cuenta que no todas las etapas del pipeline[11] son programables. Existen *shaders* para vértices (que se ejecutarán en el procesador de vértices anteriormente estudiado) y *shaders* para fragmentos (que lo harán en el procesador de fragmentos).

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 esto se pueden 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.

Un **fragment shader** no interviene en el proceso de la definición del "esqueleto" de la escena. Aplica las texturas y se tratan los píxeles que forman parte de ellas. Básicamente, un *fragment shader* especifica el color de un píxel. Este tratamiento individual de los píxeles permite que se realicen cálculos principalmente relacionados con la textura del elemento y en tiempo real.

Existen diferentes lenguajes para poder programar una GPU. Desde el más bajo nivel, como puede ser el ensamblador, como distintos lenguajes de alto nivel como son Cg de Nvidia, GLSL, HLSL (High Level Shading Language) de Microsoft, OpenGL Shader, CUDA de Nvidia…

CG

Su nombre viene de "C for Graphics" y fue desarrollado por Nvidia. *Cg* está definido sobre lenguaje *C*, por lo que su sintaxis resulta muy parecida, a pesar de ser un lenguaje de propósito específico. Sin embargo, la manera de programar es completamente diferente a lo que un programador de CPU estaría acostumbrado. Esto es debido a que se está programando sobre una arquitectura completamente diferente (como vimos en los apartados anteriores) y es necesario su conocimiento para poder empezar a programar sobre ella [5].

Cg ofrece la posibilidad de programar sombras, geometrías, animaciones… sobre el hardware gráfico, por lo que es un *shading language*. La manera de programar en *Cg* es muy parecida a la estructura de una GPU que se vio anteriormente. Se trabaja con vértices y con fragmentos, creando *vertex shaders* y *fragment shaders*.

Además del lenguaje, se debe utilizar una **API** (*Application Programing Interface*) gráfica para comunicar el procesador central con el gráfico. Hay dos APIs principales: por un lado **OpenGL** (*Open Graphics Language*), que es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fue creada por *Silicon Graphics* a principios de los años 1990 y es gratuita, libre. Se utiliza principalmente en aplicaciones de CAD, realidad virtual o simulación de vuelo.

Su principal competidor es **DirectX**, creado por Microsoft, que sólo funciona para Windows. Es utili-

[10] Los lenguajes de programación como C, Java, Pascal, C++ son lenguajes para la CPU.
[11] Otra forma de llamar al cauce gráfico.

zado por la mayoría de los videojuegos comercializados para Windows.

CUDA

CUDA es una implementación tecnológica de Nvidia nacida en noviembre de 2006. Se compone de una arquitectura hardware y de un SDK de alto nivel para compilar los programas que vayan a hacer uso de la misma (todas las GeForce a partir de la serie 8 incluyen la tecnología Nvidia CUDA) trabajando en conjunto. Mediante un grupo de extensiones sencillas del lenguaje permite al programador hacer uso de la tarjeta para funciones de cálculos masivos y paralelos, es decir, para propósitos generales. De esta manera, la GPU puede ser utilizada para cualquier tipo de operaciones matemáticas que involucren procesamiento en paralelo [2].

Contiene una síntesis muy parecida a C para crear programas que se ejecuten en diferentes **threads**. Esto quiere decir que la tecnología CUDA procesará miles de tareas simultáneamente, habilitando así una gran capacidad de flujo de datos. Permite a cientos de núcleos de procesador comunicarse simultáneamente y cooperar para resolver problemas complejos de cómputo.

A diferencia de Cg, no requiere del uso de una API para comunicarse con la CPU.

La nueva arquitectura unificada

Con el nacimiento de CUDA, cambió la manera de diseñar y programar las GPUs. Mientras que, como vimos anteriormente, hasta la serie 8 de Nvidia las tarjetas tenían procesador de vértices y procesador de fragmentos, la arquitectura unificada lo que realizó fue la unión de ambos procesadores. De tal manera, que en las tarjetas nuevas de Nvidia todos sus procesadores pueden comportarse como procesadores de vértices o procesadores de fragmentos. Por ejemplo, la tarjeta 8800GTX de Nvidia cuenta con 128 procesadores unificados, capaces de realizar operaciones sobre vértices, primitivas geométricas o fragmentos.

Esto se inició porque existen algunas aplicaciones gráficas que utilizan mucha geometría o iluminación, que requieren de un mayor número de procesadores de vértices y apenas algún procesador de fragmentos.

Figura 7. Fotografía de una tarjeta gráfica 8800GTX de Nvidia.

Como con el cauce gráfico clásico[12] había un número definido de procesadores de vértices y otro de fragmento, con mucha probabilidad los procesadores de fragmentos no harían nada, mientras que los de vértices serían insuficientes.

Lo mismo ocurriría en una aplicación que hiciera uso de un gran número de texturas. Los procesadores de fragmentos estarían saturados mientras que los de vértices no harían nada. Por ello, al crear un procesador unificado[13], estos problemas desaparecerían, ya que es posible utilizar un cierto número variable de recursos dependiendo de la carga en cada etapa y en cada momento.

En este tipo de arquitecturas, se puede conseguir un mejor rendimiento. Por otro lado, las FPUs (donde se realizan todos los cálculos necesarios) están mejor diseñadas que en series anteriores, por lo que consiguen mayor precisión en sus cálculos.

Hoy en día, si se va a una tienda de informática y se desea comprar una de las últimas tarjetas de Nvidia, dicho hardware tendrá arquitectura unificada, por lo que los conocimientos aquí explicados pueden resultarle de mucha ayuda a la hora de elegir uno u otro modelo. O si tiene alguna duda, por ejemplo, entre comprar una serie 7 de Nvidia o una serie 8, sea consciente de que la arquitectura entre una y otra es completamente diferente, por lo que, posiblemente, su rendimiento también lo sea.

Capacidad de cómputo

Como indicamos al inicio de este artículo, las GPUs han evolucionado tanto que han conseguido ser una herramienta más potente a lo que a cálculos se refiere. Podemos ver en la figura 8 cómo el núme-

[12] Con este término nos referiremos al diseño de las GPUs que se describieron en secciones anteriores.
[13] Conocido como *scalar stream processor*.

ro de **GFLOPs**[14] es considerablemente más elevado en el caso de la GPU [4].

En los últimos años, muchos países y empresas grandes han invertido miles de millones en crear centros de supercomputación (como el *Marenostrum* en Barcelona) capaces de calcular 42144 GFLOPs en pico. Este tipo de arquitecturas pueden resultar demasiado costosas (no sólo su creación, sino también su mantenimiento), por lo que su número es aún pequeño en todo el mundo.

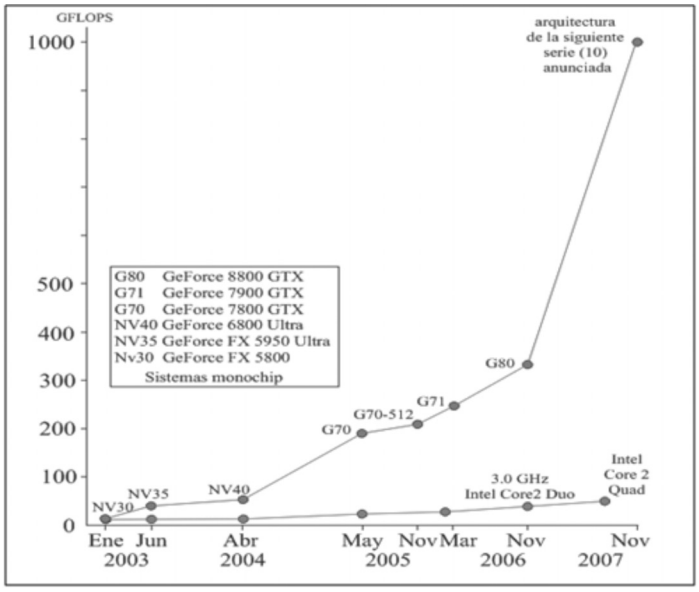

Figura 8. Gráfico que muestra la evolución de las GPUs frente a las CPUs. Es importante recalcar que se está comparando únicamente la capacidad de ejecutar instrucciones en coma flotante [3].

Sin embargo, una de las grandes ventajas de las GPUs es que poseen una gran capacidad computacional frente a otro tipo de arquitecturas con sus FPUs capaces de generar gráficos en tiempo real. Si fuéramos capaces de aprovechar la capacidad de cálculo de un procesador gráfico (una única tarjeta Nvidia 8800GTX llega a 500 GFLOPS de pico) y formar un clúster con varias, podríamos tener un mayor número de GFLOPs a un precio mucho más barato. Si, por ejemplo, creáramos un clúster con nueve tarjetas 8800GTX por, aproximadamente, un precio de 1808 € conseguiríamos alrededor de 4,5 TeraFLOPS.

Comparativa de tarjetas gráficas

Para terminar, vamos a adjuntar un cuadro que compara seis tarjetas actuales (Nvidia y ATI). Las más recientes salieron al mercado en julio del 2008 y cuentan con doble precisión. Esta comparativa ha sido traducida de la página de Tom Hardware. Con esta gráfica, si se desea comprar una de las tarjetas que se exponen a continuación, se podrá hacer una elección más acertada.

Tras la lectura del presente artículo, espero que el entendimiento de la tabla sea mayor y ayude al lector a tomar algunas cifras reales de las tarjetas gráficas de última generación.

GPU	HD 3870 X2	9800 GX2	8800 ULTRA	GTX 260	GTX 280
Frecuencia GPU	825 MHz	600 MHz	612 MHz	576 MHz	602 MHz
Frecuencia ALU	825 MHz	1500 Mhz	1512 MHz	1242 MHz	1296 MHz
Frecuencia memoria	900 MHz	1000 MHz	1080 MHz	999 MHz	1107 MHz
Ancho del bus de memoria	2x256 bits	2x256 bits	384 bits	448 bits	512 bits
Tipo de memoria	GDRR3	GDRR3	GDRR3	GDRR3	GDRR3
Capacidad de memoria	2x512 MB	2x512 MB	768 MB	896 MB	1024 MB
Número de ALUs	640	256	128	192	240
Número de unidades de textura	32	128	32	64	80
Número de ROPs	32	32	24	28	32
Potencia de shaders	1 TFLOPs	1152 GFLOPs	581 GFLOPs	715 GFLOPs	933 GFLOPs
Ancho de banda de la memoria	115,2 GB/s	128 GB/s	103,7 GB/s	111,9 GB/s	141,8 GB/s
Número de transistores	1334 millones	1010 millones	754 millones	1400 millones	1400 millones
Generación	2008	2008	2007	2008	2008
Shader model que soporta	4.1	4.0	4.0	4.0	4.0

[14] En informática, FLOPS es el acrónimo de *Floating point Operations Per Second* (operaciones en coma flotante por segundo). Se usa como una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren de un gran uso de operaciones de coma flotante.

Conclusión

Como se ha podido ver, tras una tarjeta gráfica se esconde un millón de conceptos y funciones claves para la generación de películas animadas, videojuegos, simuladores de vuelo o agencias espaciales. Aunque todo ordenador posee una, en la mayoría de los casos no somos conscientes de la herramienta tan potente que se encuentra pinchada en la placa base de nuestro PC. Resulta un tema fascinante y complicado, que asegura aumentar más su importancia en el mundo de la informática (debido al auge de efectos especiales, películas de animación y videojuegos que está sufriendo la sociedad en los últimos años). Y este breve artículo, no es más que una ínfima introducción a todo lo que esconde.

Referencias destacadas

[1] Manuel Ujaldón Martínez. *Procesadores gráficos para PC.* Ed. Ciencias-3, septiembre 2005.

[2] *Guía de programación de CUDA.* www.nvidia.com/cuda

[3] David Miraut. *Introducción a las tarjetas y Procesadores Gráficos.*

[4] *GPU Gems2. General-Purpose. Computaction on GPUS: A Primer.* Nvidia. 2005.

[5] *The Cg Tutorial - The Definitive Guide To Programmable Real-Time Graphics.* Nvidia. 2003.