# **Graphics Processing Unit (GPU) design concepts**

A graphics processing unit (GPU) is a processor like CPU and TPU for faster graphics processing. Specifically, it designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer to be displayed on a screen.

The parallel structure of a GPU makes it more efficient for algorithms where several components can be executed in parallel such as Machine Learning algorithms/ inference.

In this article, we have explored some of the basic architecture concepts in Graphics Processing Unit (GPU).

**GPU Architecture**

**The Graphics Pipeline**

The input to the GPU is a list of geometric primitives, typically triangles, in a 3-D world coordinate system. Through many steps, those primitives are shaded and mapped onto the screen, where they are assembled to create a final picture. It is instructive to first explain the specific steps in the canonical pipeline before showing how the pipeline has become programmable.

**Vertex Operations:** The input primitives are formed from individual vertices. Each vertex must be transformed into screen space and shaded, typically through computing their interaction with the lights in the scene. Because typical scenes have tens to hundreds of thousands of vertices, and each vertex can be computed independently, this stage is well suited for parallel hardware.

**Primitive Assembly**: The vertices are assembled into triangles, the fundamental hardware-supported primitive in today’s GPUs.

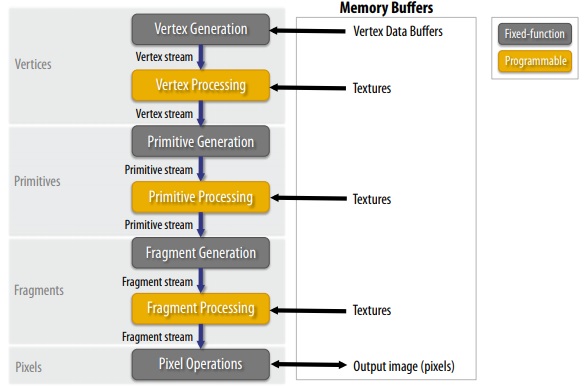
**Rasterization**: Rasterization is the process of determining which screen-space pixel locations are covered by each triangle. Each triangle generates a primitive called a fragment at each screen-space pixel location that it covers. Because many triangles may overlap at any pixel location, each pixel’s color value may be computed from several fragments.

**Fragment Operations:** Using color information from the vertices and possibly fetching additional data from global memory in the form of textures (images that are mapped onto surfaces), each fragment is shaded to determine its final color. Just as in the vertex stage, each fragment can be computed in parallel. This stage is typically the most computationally demanding stage in the graphics pipeline.

**Composition:** Fragments are assembled into a final image with one color per pixel, usually by keeping the closest fragment to the camera for each pixel location

### **Graphics Pipeline**

This image summaries the graphics pipeline in a GPU:

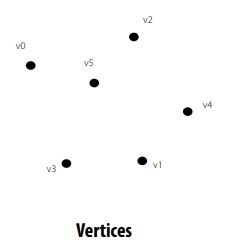


You need to understand the following ideas:

* Vector processing
* Primitive Processing
* Rasterization
* Fragment processing
* Pixel operations

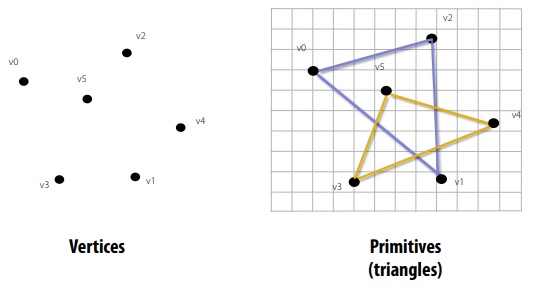
### **Vector processing**

Each vector is transformed in screen space and processed independently.



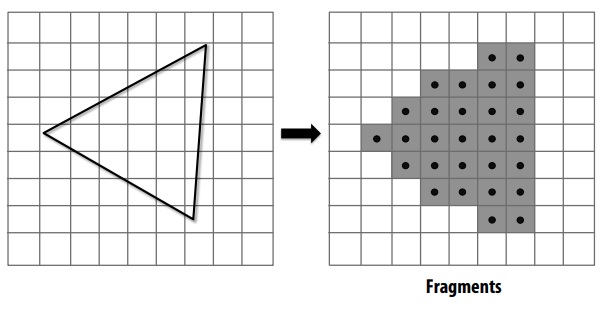
### **Primitive Processing**

The vertices are organized into primitives that are clipped.



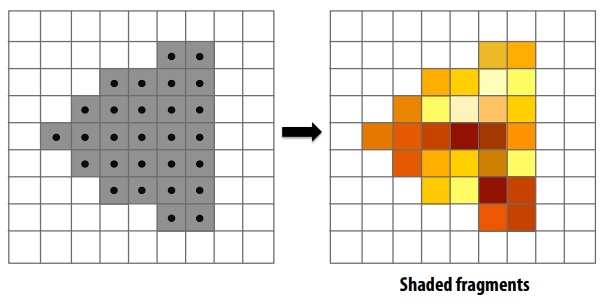
### **Rasterization**

Primitives are rasterized into “pixel fragments”. Each primitive is rasterized independently.



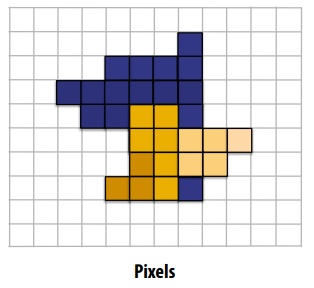
### **Fragment processing**

Fragments are shaded to compute a color at each pixel. Each fragment is processed independently.



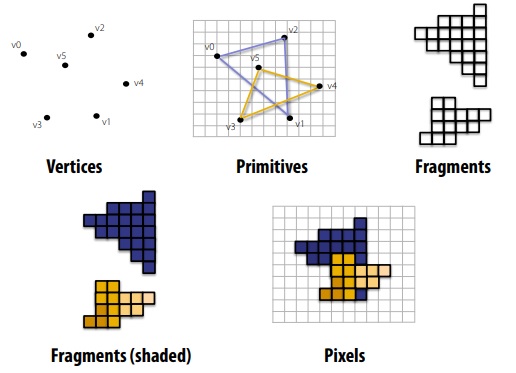
### **Pixel operations**

Fragments are blended into the frame buffer at their pixel locations (z-buffer determines visibility).



There are five pipeline entities:

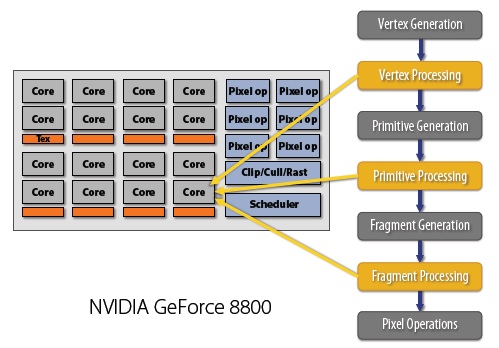
* Vertices
* Primitives
* Fragments
* Shaded fragments
* Pixels



### **Graphics architectures**

In this section, we will take a look at how the Graphics pipeline is implemented and how the independent nature of the operations are maintained.

This image demonstrates the architecture of NVIDIA GeForce 8800 developed in 2006:



The above architecture uses the concept of unified shading which is a significant improvement over the basic CPU architecture and is one of the secrets of GPU's speed.

## Difference between GPU and CPU

The CPU (central processing unit) has often been called the brains of the PC. But increasingly, that brain is being enhanced by another part of the PC – the GPU (graphics processing unit), which is its soul. Architecturally, the CPU is composed of a only few cores with lots of cache memory that can handle a few software threads at a time. In contrast, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously. The ability of a GPU with 100+ cores to process thousands of threads can accelerate some software by 100x over a CPU alone. GPU achieves this acceleration while being more powerful and cost-efficient than a CPU.

## GPU Applications

1. Gaming : PC GPUs were originally invented for 3D gaming on PCs. Using modern GPUs has also enabled game developers to build animated characters that bring the maps to life.
2. Productivity : Microsoft Office 2010 now offers GPU acceleration for some of its graphical elements, like WordArt and PowerPoint transitions.
3. Video Editing : Video editing demands heavy use of system resources even on high end PCs. Consumer applications, like Adobe Premiere Elements 9, are offering features previously available only for professionals. Transitions like page curl, sphere or card flip are all GPU-accelerated in Premiere Elements 9. Effects like refraction and ripple are also accelerated by a GPU. A graphics card with an AMD Radeon GPU will speed up preview and final rendering, making it faster and more fun to create your video.