The Graphics Pipeline

Matt Gill

Graphics CS147

Rendering images is a very complicated process that needs specialized hardware and optimized processes to be the most effective. The main process used for rendering is the graphics pipeline. The pipeline can be boiled down into 4 basic steps. Vertices are passed through a vertex processor, then to the clipper and primitive assembler, then through the rasterizer, and then through the fragment processor. These steps can be broken down into more smaller steps, but in general these four steps are what every graphics rendering program will have to perform. The vertex processor is responsible for taking a set of vertices and assigning attributes to them. Essentially in this step you are creating the basic scene. Think of a room with a camera and nothing else, then you start adding things. During this stage it is important to finalize the position of every vertex, so each vertex goes through this step. Next, we have to decide what actually needs to be rendered. For instance, you would not need to render the back side of an object that you can’t see. This is what the clipper is used for, it decides what is necessary to be rendered. Now that we have a scene, we can break it down a displayable object. The problem is, we have to display it on a grid of pixels. To do this we use a rasterizer which decides which pixels to use to draw shapes on a screen. With the scene now transformed into a grid of pixels the fragment shader goes through and determines the color of each pixel based on the scenes lighting and other objects. This process will inevitably leave artifacts and imperfections behind. The fragment processor then goes through each pixel and fixes issues with rasterization and other steps of the process. Throughout this whole process it is important to remember that some hardware is better at some steps of the process. For rendering, the GPU is almost always faster than the CPU, but they must still work together for maximum efficiency. There are some things the CPU would take a very long to execute that the GPU would have no problem with, and so it is important to know which steps should be executed by what hardware. That is one reason that having a pipeline is so important. Is allows for the execution to be done in the correct order and send information to the right hardware as it is needed. It is also important to be aware of bottlenecks in your hardware. For instances if the GPU is running much faster than the CPU the GPU will have to slow down and wait for the CPU to catch up. This is especially true in real time rendering applications such as gaming. As stated before, there is much more to this process than even this list gets into, but it is the basic graphics pipeline and the most versatile.

Works Cited

<https://developer.nvidia.com/gpugems/gpugems/part-v-performance-and-practicalities/chapter-28-graphics-pipeline-performance>

<https://www.geeksforgeeks.org/opengl-rendering-pipeline-overview/>