Adam Katz

2/16/2020

Computer Graphics

WebGL and the graphics pipeline.

The graphics pipeline is a step by step process for generating images on a computer. While each step of the pipeline has its own role and purpose, parts can be executed simultaneously in order to streamline efficiency. There are seven steps in the graphics pipeline. Certain sources have varying number of steps however they can be boiled down to four major stages: The vertex shader, primitive assembly, rasterization and fragment processing.

The first step in the graphics pipeline is the vertex shader. The vertex shader is responsible for passing vertices data from the CPU to the GPU. The data consists of the vertices color, texture, and coordinates. This is essentially building the skeleton of the image being rendered, in which each point is processed by the GPU. The second step is the is a primitive assembly of the shapes. During primitive Assembly, the GPU Connecticut’s the vertices form the Vertex Shader into rudimentary triangles, then passes them on to the third step.

The third Step is rasterization. In this step the primitive image is filled in with the remaining pixel information. Rasterization also has two sub steps. The first is Culling, which is the process of discarding any part of the image being rendered. For example in a 3D animation, parts of an object being rendered that are not in direct view do not need to be rendered for that frame. The other subset is Clipping, which refers to discarding the pixels that are outside of the visible frame. That way nothing more than the pixels on the screen are being rendered.

After rasterization, the fourth step is the is Fragment Processing. In this step, the Fragment Shader receives all the information from the past steps and calculated the color value of each pixel in a frame. Once this step is complete frame is produced.

As I stated before, while each of these steps do, they’re own process, they run overlapping to improve efficiency. During an animated film, or video game, multiple frames need to be rendered to keep an acceptable frame per second rate. When one frame is being rasterized, the next maybe going through the vertex shader.