Anna Wolff

Professor Urness

CS 147

16 February 2021

Assignment 2

The 3D graphics pipeline uses a step-by-step process to render an image, turning the vector graphics into a 2D raster image. To display this image to a screen, a device must have four parts: a central processing unit (CPU), a graphics processing unit (GPU), random access memory (RAM), and graphics memory. The CPU is responsible for performing the general processing of a system. It also runs the operating system that controls how the device functions. The GPU is used to quickly process large amounts of computations. This is generally used for 3D graphics but has recently been used in areas such as speech recognition and neural network training. RAM stores program data and instructions used in the graphics pipeline, and the graphics memory stores data that helps to facilitate 3D graphics such as vector data, raster images, and a screen’s pixels. These components are used in conjunction with each other to create the graphics we see on our screens every day.

While in its early days, the pipeline was immutable, we can now have access to shader programs that allow us to slightly customize the process. These programs can quickly become complex, but they provide the user with more creative freedom by manipulating the graphics data in the pipeline. First, the CPU must use a webGL program to gather the data that the shader program uses and set it up to go through the graphics pipeline in the GPU. It takes in any uniform variables that the shader program needs that won’t get changed throughout the rendering process. Additionally, it will link the vertex shader to the attribute variables that get held in GPU buffers. Once this data has been collected, the vertex shader executes. The vertex shader reorganizes the coordinates of each vertex in the image from their input to the device’s coordinate system. It also calculates the data for various properties of the vertex, such as its color or texture coordinates. Then, the program goes through a process of primitive assembly where it is clipped, transformed for view, and rasterized. These steps help to edit the image to make it show up normally and within the bounds of the screen you are using. Next, a fragment shader collects data from the vertex shader and primitive assembly stages to calculate the color for each pixel between the vertices. Finally, the information is ready to produce the image and can be outputted to the screen through the draw buffer.

Works Cited

Works Cited “1.3 - the 3D Graphics Pipeline — Learn Computer Graphics Using WebGL.” Runestone.academy, 2017, runestone.academy/runestone/books/published/learnwebgl2/01\_the\_big\_picture/3\_3d\_graphics\_pipeline.html. Accessed 16 Feb. 2021.

“WebGL - Graphics Pipeline - Tutorialspoint.” Tutorialspoint.com, 2021, www.tutorialspoint.com/webgl/webgl\_graphics\_pipeline.htm. Accessed 16 Feb. 2021.