Lucas Swidler

CS147

The graphics pipeline begins with WebGL being run through JavaScript. JavaScript is used to create the code for WebGL projects. The graphics pipeline is the way for the code of WebGL and JavaScript to be turned into 2D or 3D images. The first step in this pipeline is called vertex processing. In this step, each point created in the code gets assigned a position on the screen, which is then stored inside gl\_position in the program. The positions are found methods like drawElements() and drawArray(). In this step as well, vertex shaders are used. These are meant to hold any attributes of the vertices that one may want. This can include color, texture, or any other transformations needed for that vertex. Following the creation of the vertices, primitive assembly must be done. In this step, the vertices are converted into shapes like triangles to make up the image. The next large step in the graphics pipeline is called rasterization. In this step, the raster is created, or all the pixels that will be part of the image. The triangles that were created before are filled with pixels. Included in this stage is clipping, where any extra pixels are removed from the image. For example, for an image depicting a 3D object, the pixels of the backside of the object are removed to appear more realistic. Any pixels that would be outside of the dimensions are also removed in this stage. After rasterization is the final important step, the fragment shader. In this step, any attributes, or specific details, of the vertices are implemented. Common examples of attributes are colors or textures. For pixels besides the vertices, color values are computed using primitive data from the rasterization phase. The final step is the frame buffer, which stores the image and any data associated with it.

To summarize, the vertex processor creates the vertices of the image. In primitive assembly, the vertices are converted into triangles. The rasterizer fills in the triangles with pixels and eliminates any excess, and finally the fragment shader takes any attributes of the vertices and applies them to the image. The frame buffer holds the data of the image and so it can be put onto the screen later.

Sources Used:

<https://www.wisdomjobs.com/e-university/webgl-tutorial-1269/webgl-graphics-pipeline-16662.html>

<https://www.pcmag.com/encyclopedia/term/graphics-pipeline>