Graphics Pipeline

We define the “graphics pipeline” as that a step-by-step process that turns vector data into raster images. In the early days of computer graphics, programmers were not able to manipulate this pipeline at all. Nowadays the GPU allows us to control said pipeline using shaders. Shader programs are used to manipulate graphic data and provide flexibility to the programmer.

The graphics pipeline can be seen as an assembly line for computer graphics. Thanks to this “architecture” the GPU is able to perform billions of calculations per second allowing computers to render more data in a shorter time frame. As for the pipeline, it is divided into the following steps which are mainly executed in the GPU:

* Step 1: The Vertex Shader program establishes a connection with the GPU buffers that contains the data shaders use for model rendering. This data is stored as uniform variables in the shader program.
* Step 2: The shader program is executed on each vertex and it is transformed into normalized device coordinates. This then maps the coordinates in front of a “virtual camera.”
* Step 3: The program clipsaway the data that cannot be seen in front of the “virtual camera.”
* Step 4: The model data from the normalized device coordinates is then mapped in pixels. The vertices collected are formed into triangles.
* Step 5: Geometric primitives are rasterized to evaluate the pixels that are within the raster image’s boundaries.
* Step 6: Fragment shaders are applied on every pixel in the geometric primitives to get a color value for said pixel.
* Final Step: Combines the color values of the pixel assigned by the fragment shader with the one stated by the color value assigned to the draw buffer to finally display the image.

One of the many challenges of computer graphics is the rendering of photorealistic images in a reasonable amount of time. Some of the main obstacles is getting human skin and expressions right. According to NVDIA, textures can cause the GPU’s texture cache to run out of memory causing a bottleneck in the pipeline. This is why we have to be careful when working with graphics and save memory as much as possible. This way the GPU can effectively execute its processes to output images.

Citations:

https://runestone.academy/runestone/books/published/learnwebgl2/01\_the\_big\_picture/3\_3d\_graphics\_pipeline.html

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

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