I believe that the graphics pipeline are all the components in webGL that truly produce visual graphics. This includes the rasterizer, vertex processor, clipper and primitive assembly, fragment processor, and the shaders. The main part of the pipeline responsible for giving an image color are the shaders, that is the main purpose of the shaders. The rasterizer’s main job is to turn the primitives (produced in the primitive assembler) into fragments for the fragment processor to turn into pixels. The main function of the vertex processor is to carry out coordinate transformations which includes computing the color for each vertex and changing any other attributes of the vertex (section 1.7 of the textbook). The clipper and primitive assembly are responsible for creating faces out of vertices (primitive assembly) and taking apart edges that lie outside of the clipping volume and assembling them into smaller parts. The clipper also skips edges that lie completely outside the clipping volume (notes on clipper and primitive assembly came from the article you provided)! This concludes the basic outline of what I think the graphics pipeline is, I will now go into more detail about the pipeline.

The shaders give every single pixel a color whether it is white, black, or any color in between, the shaders assign a color. The article states that the vertex shader is completely programmed by the user. This includes vertex color, final vertex location, and even the angle at which the vectors will be displayed! The article shows what a full graphics pipeline may look like in OpenGL. It goes as follows: Vertex Shader, Tessellation and Control Shader, Tessellator, Tessellation Evaluation Shader, Geometry Shader, Rasterizer, and lastly the Fragment Shader! All parts in this pipeline are programmable to some degree, with the exceptions of the Tessellator and Rasterizer! Of course, the graphics pipeline we have been looking at is much simpler. That pipeline goes as follows: Vertices, Vertex Processor, Clipper and Primitive Assembler, Rasterizer, Fragment Processor, and then lastly pixels are output! I will now describe how I think the graphics pipeline was used in each of the assignments thus far!

We have used the graphics pipeline on every assignment thus far (except for assignment 5). It was used to produce the triangle with text in it for assignment 1. In assignment two the pipeline was used to output the picture that was comprised of multiple triangles (in my case it was 12 triangles that made a block person). Assignment 3 used the pipeline for outputting the animated object (in my case this was a rotating square). The pipeline was also used in this assignment to control a slider, command a menu, and control a button. In assignment 4 the pipeline was again used to display an object. This time it was a cube that was rotating! The pipeline was also used in this assignment to control buttons for which axis (x, y, or z) that the cube was rotating about. Lastly, in assignment 6 the graphics pipeline was used to display the shaded sphere. It also controlled about 10 sliders and a menu that switched the shading of the sphere between Gouraud shading and Phong shading. That is what I could think of to describe the pipeline.