Peyton Somerville

HW1 Report

This program is available to view online with the following link:
https://peyton-somerville.github.io/CompGraphicsHW1/index.html

The first issue I faced was trying to get the shader code working in javascript instead of the html. Moving the vertex and fragment shader over was pretty simple, and just involved making them into a string format. The problem was that the default initShaders() function in WebGL no longer worked when the shaders were moved to the javascript, so I had to do a lot of research to figure out how to make a custom function to initialize the shaders. I wrote a function called initMyShaders() which accomplishes pretty much the same thing as initShaders(), but now works with the shaders in the javascript code.

The next issue I was faced with was figuring out how to have the color change based on a variable. I figured out pretty quickly that the color of the gasket was controlled by the gl_FragColor variable in the fragment shader code. Changing the values in the vec4 would change the color. So I would just need to make gl_FragColor equal to a vec4 color variable, which was picked from an array of vec4s that is in my main javascript code as a global variable called colors[]. The problem with that was the shader code was not able to access variables from the javascript program directly. So I edited the shader code to add a uniform vec4 variable called color, and had gl_FragColor = color. Then I used gl.getUniformLocation() to get the location of the color variable in the shader code, and used that location to set the value to a vec4 color from my colors[] array. This took a bit of research, but it works great.

The last main issue I faced was having the iterations of the loop slow down. If I just did all the iterations in a for loop, then it would go way too fast to even be able to see what is happening. So I needed to find a way to pause for a set amount of time between each iteration. The solution I found involved a built-in function called setInterval(). This function takes two arguments, the first being a function to call, and the second being an interval time in milliseconds. So the setInterval() function will call a function every 2 seconds if you give it 2000 for the second argument. In this function called renderLoop() that setInterval() calls, I first call the render() function to display the current gasket, then I update all the variables to change the color, the size, and the number of points. Then for example every 0.5 seconds, the renderLoop() gets called again from setInterval() which will display a new gasket image and with this done continuously, it becomes an animation.

I added some extra features to control the animation. There is a button which will pause the animation, which calls a built-in function called clearInterval(). Then you can resume the animation with another button, which just calls the setInterval() function again. There are also buttons that will speed up or slow down the animation speed by just changing the interval time for the setInterval() function. Clicking the resume animation button while the animation is already running will mess things up, so I made it so that button will only be visible once the pause button is clicked. Also once the resume button becomes visible, all of the other buttons get hidden because they would also mess up the animation if clicked when they should not be.