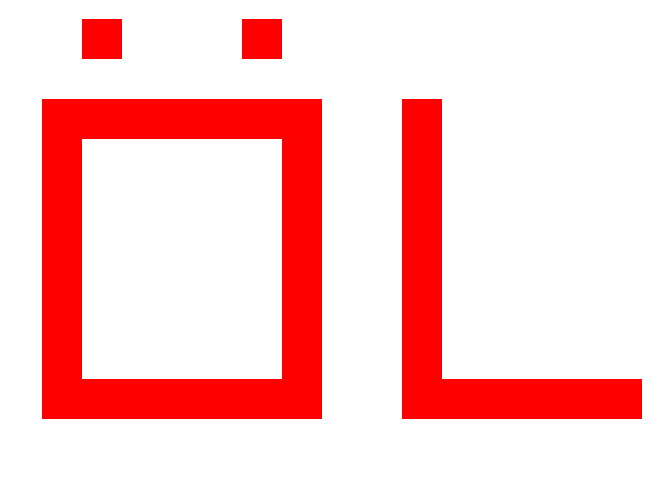
COM3037 Project Report

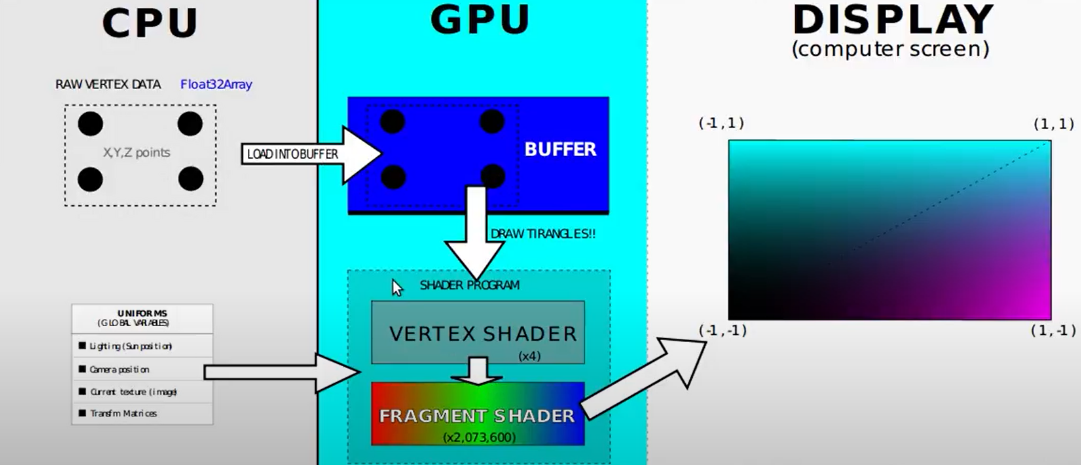
by Ömer Faruk Uysal

For the project, the given task is to display the first letters of our first name and last name using geometric WebGL primitives. To achive this task, I will display the letter ‘Ö’ and the letter ‘L’, which come from my name, Ömer Faruk Uysal. The resulting visualization will look like the following figure.



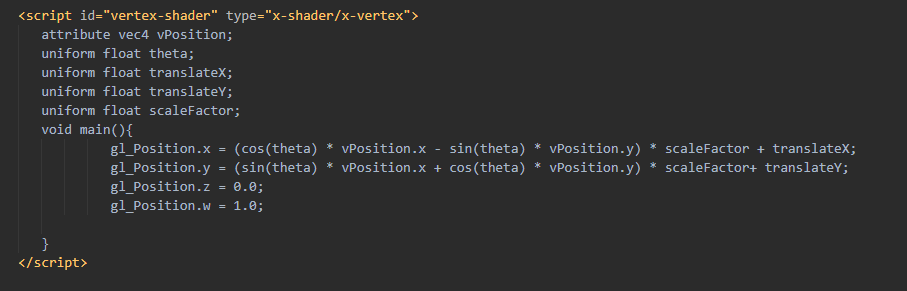
Figure

To display something like this on the screen, we first need to look at some basics of computer graphics. For example, at the beginning, we have to define our desired output with raw vertex data. After that, we need to load our raw data into the GPU buffer. Next, we will create a shader program with the help of WebGL functions, and this program will convert our raw data into a visualized graphics and display it on the screen. Figure 2, shows these steps.



Figure

To complete this task, one HTML and one JavaScript file is created. In the HTML file, the most important things are the canvas decleration and scripts declerations for the vertex-fragment shaders. Canvas is the place where our graphics will be displayed. And in the vertex shader, we define a function to maintain our graphics according to some paramters. That function is given in the Figure 3.



Figure

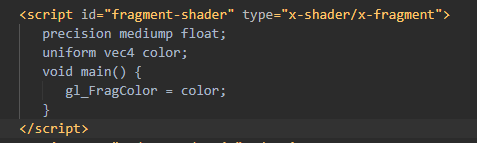
‘attribute’ and ‘uniform’ keywords indicates a global variable in WebGL. ‘vPosition’ is a vector of for element to indicate the position of our 2D graphic. Since we have 2D graphics the 3rd and 4th elements will be fixed in our program but the 1st and 2nd elemets will indicates the x and y coordinates of our graphics. The next variable ‘theta’ is used for rotation purpose. The rotated position of the graphic will be calculated with the formula given in the Figure 4.



Figure

The varible ‘scaleFactor’ is used to scale the graphic in the desired size. To achive that, calculated x and y coordinates are multiplied with scaling factor. And, as a last step ‘transşateX’ and ‘translateY’ variables are used to update the position of the graphic as desired. So in that purpose, these variables are added to the x and y coordinates of the graphic.

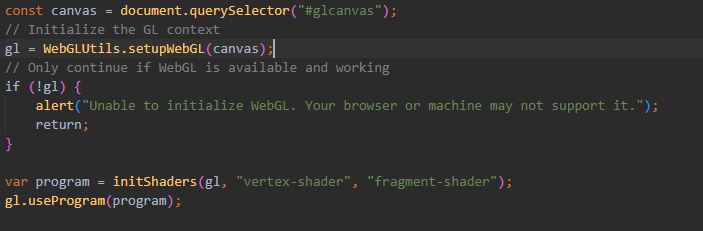
For the coloring purpose, fragment shader is constucted with the following code snippet given in the Figure 5. A vector variable with 4 elements is used in this function.



Figure

HTML file also contains buttons, paragraphs, heading and some CSS definitions but these are not explained in the report since they are not main concern of this task.

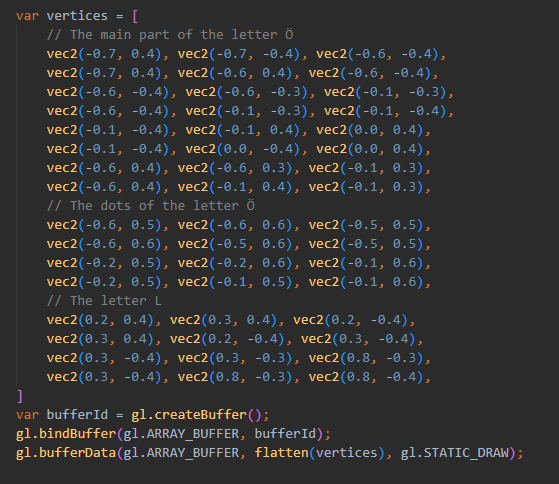
In the JavaScript file, first we need to setup the WebGL enviroment we work on. To achieve that we exploit the code snippet given in the Figure 5.



Figure

The ‘setupWebGL’ function loads the WebGL content properly and the ‘initShaders’ function initializes the shaders. Note that, the parameters given in that function is the ids of script tags defined in our HTML file. Lastly, ‘useProgram’ function makes the program ready to use.

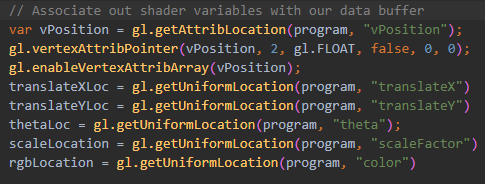
The following code snippet given in the Figure 6, creates a buffer and binds it to our raw vertex data as I mentioned at the beginning of report.



Figure

For defining our raw vertex data, I defined a variable named as ‘vertices’. This varible is a list of vectors of two elements which constructs the points of the triangles to visualize our graphic. For the letter ‘Ö’, I used 12 triangles, 4 of them for the dots of the letter. And for the letter ‘L’, I used 4 triangles.

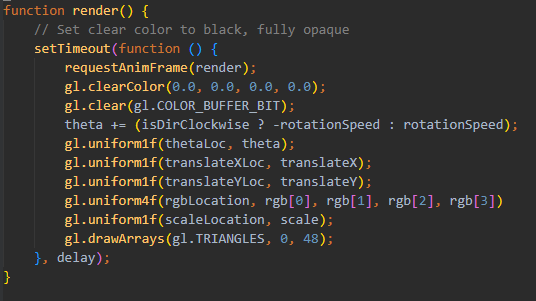
The following code snippet given in Figure 7, associates the shader variables decleared in HTML file with our variables in JavaScript file.



Figure

Notice that given parameters to the functions, are variable names we decleared in the shader script tags.

Now, we need to render our graphic according to the parameters we defined. Following code snippet helps us to do that.



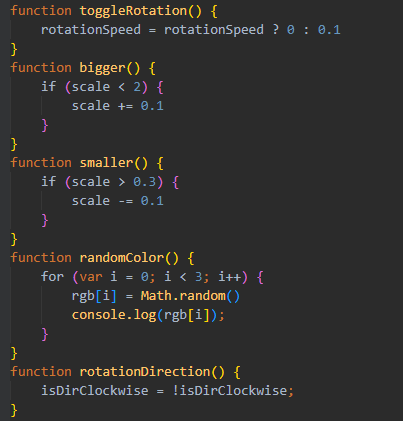
Figure

With this last step, now we are able to display our graphic according to the parameters we defined. However, we have not setup a mechanism to modify our parameters. Now, we need to define functions to modify our parameters. Following functions given in the Figure 9, are defined for the movement of our graphic. According to the movement, functions update the ‘translateX’ or ‘translateY’ variables.



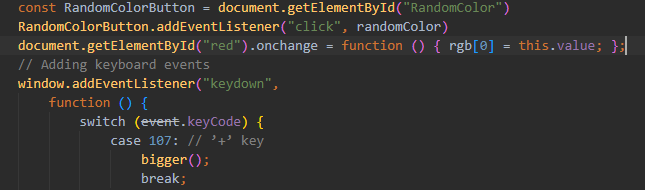
Figure

In the following code snippets, functions are defined for the purposes of scaling, rotating or coloring.



Figure

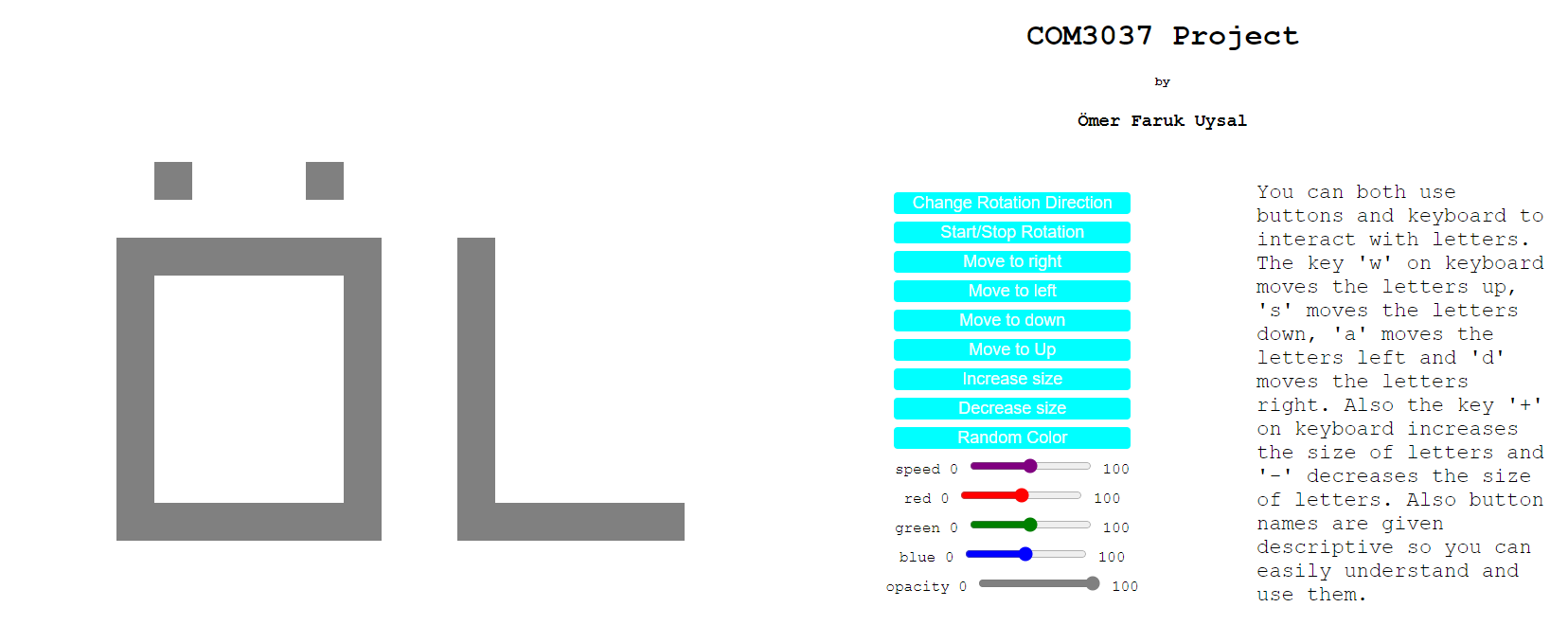
Scaling factor keept in a fixed range and modified according to the desired action. ‘toggleRotation’ function is used to start or stop rotation of graphic. And ‘rotationDirection’ function is used to change the direction of graphic. Also ‘randomColor’ function is defined for coloring the graphic randomly. These functions are linked to the corresponding JavaScript action listeners. In the Figure 12, some example linkings are given.



Figure

A bunch of buttons, sliders and keyboard events are defined and binded to the corresponding actions. With this last step, a 2D graphic of letters are displayed on the screen and maintained with the help of the buttons, sliders or keyboard.

The last view from my computer screen is given in the Figure 13.



19290337

Ömer Faruk Uysal