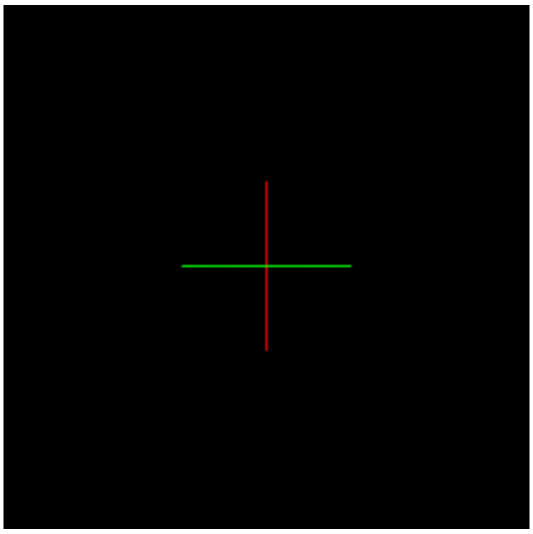
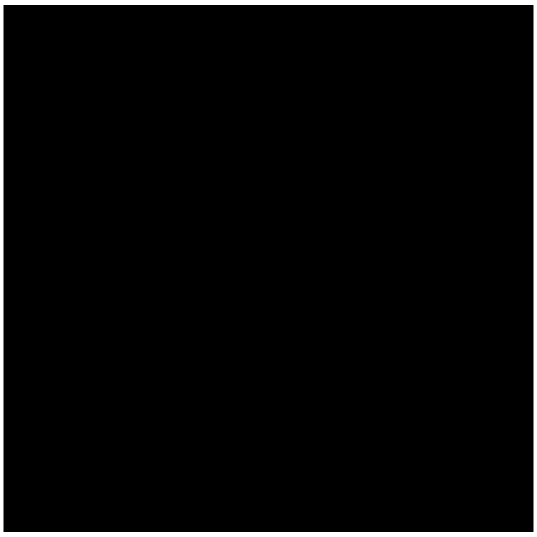
Name: Mustafa Baran Ercan
ID: 2881055520 **Section**: CMPE_360_01

PROJECT 7 REPORT

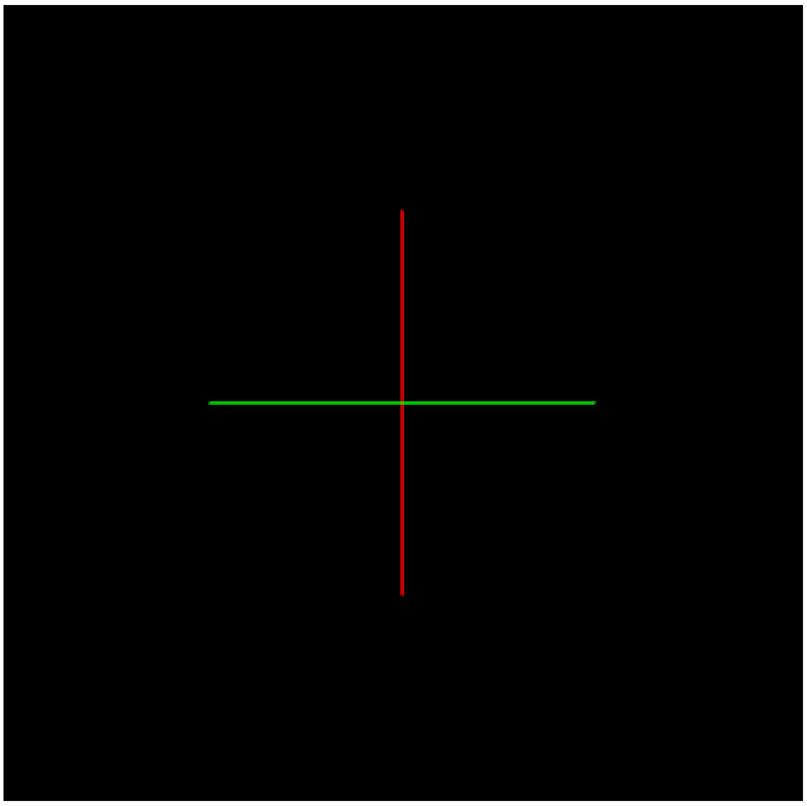
1. lookAt() function is typically used in computer graphics to define the position and orientation of a camera. It creates a transformation matrix that moves the camera to a specific position (eye), looks towards a target location (gaze), and defines the up direction. The translate() function creates a translation matrix that moves an object in 3D space. The difference in the display occurs because of the -15 value in the translate function, which corresponds to the movement of -15 units along the z-axis. Therefore, I see the object a little bit farther away than the lookat() function, which had 10 unit distance from the object on the origin.



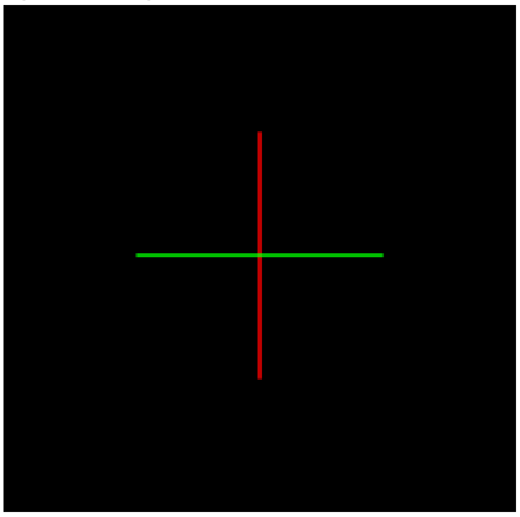
2. The display disappears since I am neither arranging the position or the direction of the camera, nor translating the axis. They are still there, I am just not seeing them.



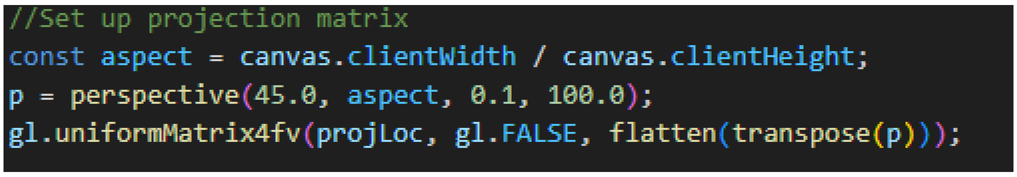
3. Restore the lookAt() call.
4. Take a look at perspective():
    a. The second parameter “1.0” in the perspective() function represents the aspect ratio. Aspect ratio refers to the ratio of the width to the height of the viewing area. When the aspect ratio is 1.0 and I changed the dimensions to 400x400, I still get a regular image because of the ratio is 1.0.



    b. I get an image with the same ratio (1.0), but just smaller than the previous.



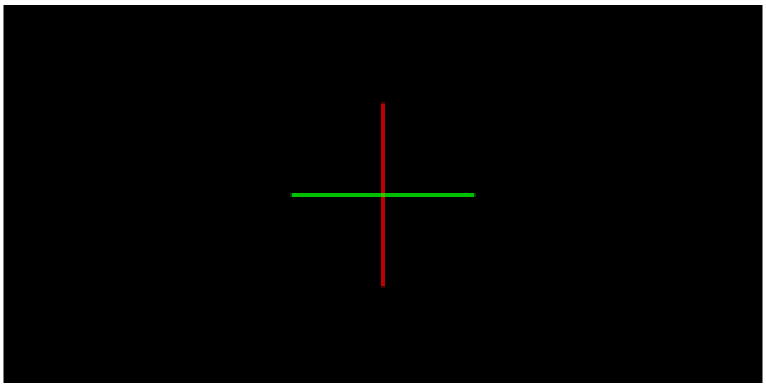
    c. I modified the aspect ratio to the value of “canvas.clientWidth / canvas.clientHeight”. With those values I obtain what size our canvas is being displayed on the screen. Therefore, I always get the correct aspect ratio. Here are two examples that I tested.



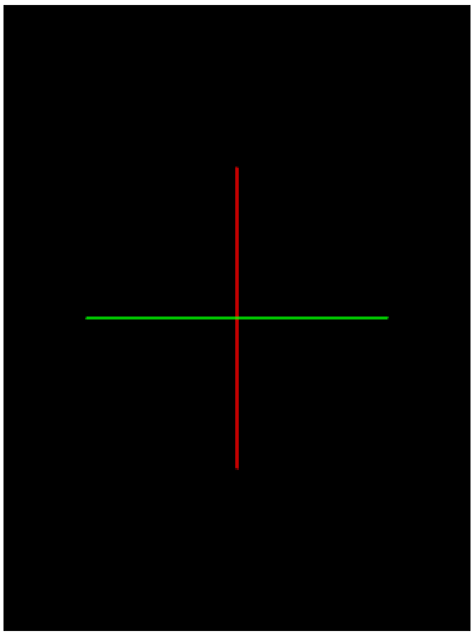
```
//Set up projection matrix
const aspect = canvas.clientWidth / canvas.clientHeight;
p = perspective(45.0, aspect, 0.1, 100.0);
gl.uniformMatrix4fv(projLoc, gl.FALSE, flatten(transpose(p)));
```

d.

"400x200"



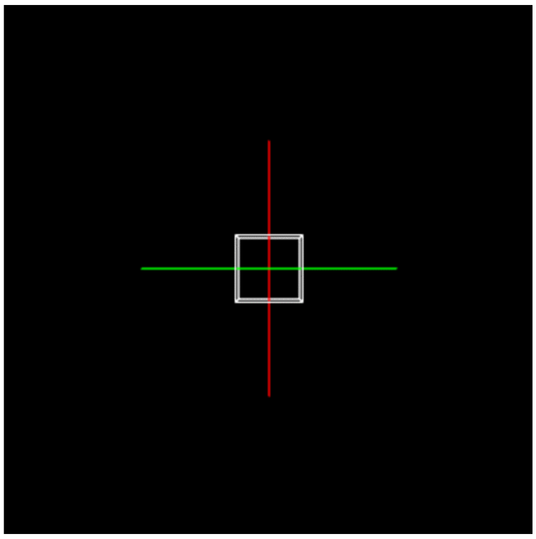
"300x400"



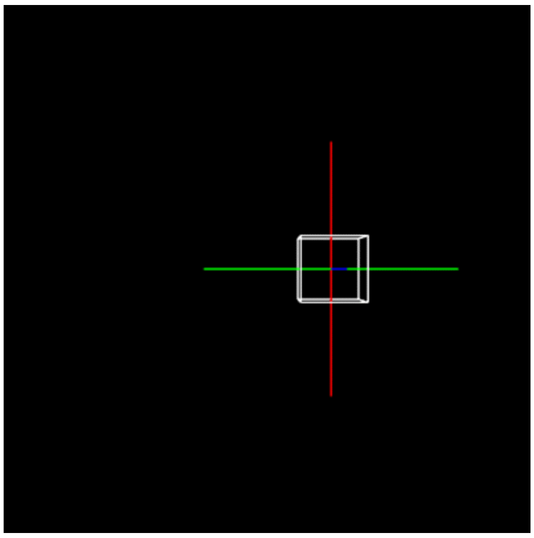
5. I can use this command:

   "gl.drawArrays(shapes.wireCube.type, shapes.wireCube.start, shapes.wireCube.size)"

   It takes three arguments. The first one specifies the type of shape to be drawn. The second one specifies the starting index of the vertex array to be drawn. The third one specifies the number of vertices to be drawn.



6. In order to move the cube to the position of (1, 0 ,0) relative to our axes, I should use a translation matrix. I want to translate the cube by 1 unit in x axis and 0 units in t and z axes. Therefore I can use: “mv = mult(mv, translate(1, 0, 0))”



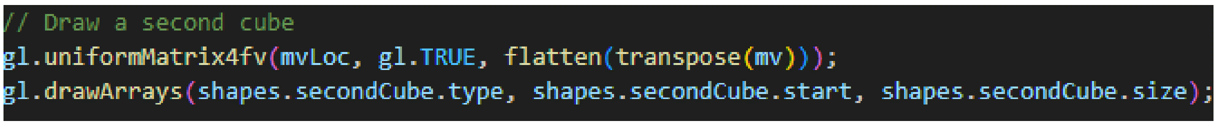
7. To draw a second cube:

   First, I added a new shape called secondCube

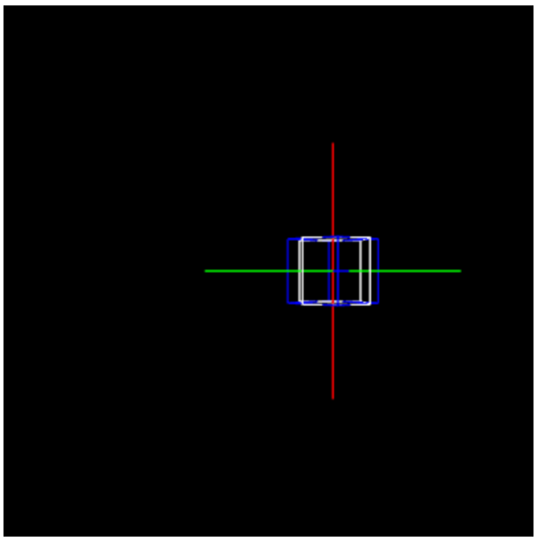
   secondCube: {points:[], colors:[], start:0, size:0, type: 0}

   Then I copied all of the attributes of the first cube, except I made it blue with this line of code: “shapes.secondCube.colors.push(blue)”

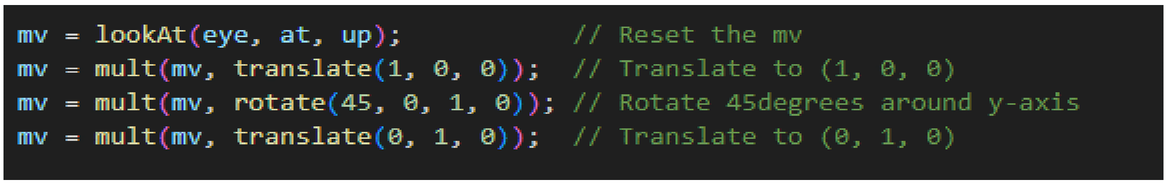
   Then, to draw the second cube correctly, I reset the mv to default lookAt() values again. After that, I rotated the mv with “mv = mult(mv, rotate(45, 0, 1, 0))” and draw the second cube.



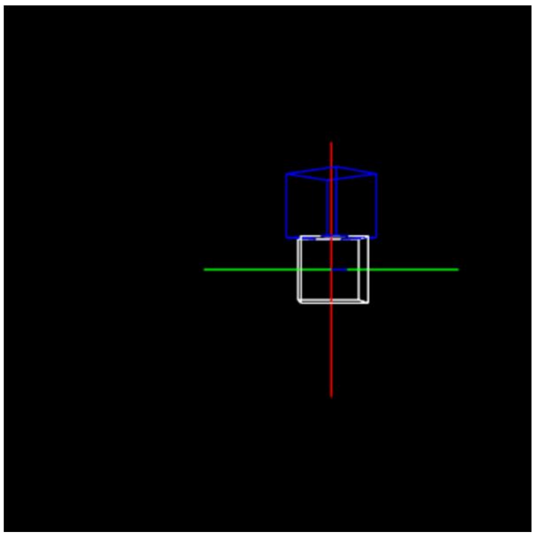
```
// Draw a second cube
gl.uniformMatrix4fv(mvLoc, gl.TRUE, flatten(transpose(mv)));
gl.drawArrays(shapes.secondCube.type, shapes.secondCube.start, shapes.secondCube.size);
```



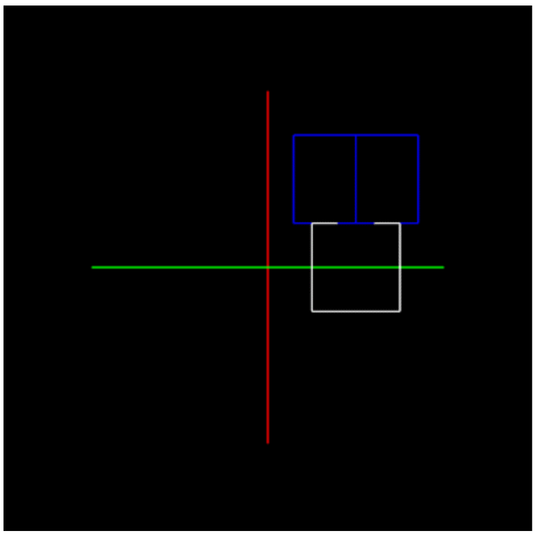
8. To place the rotated cube above the first cube which is centered at (1, 0, 0) relative to axes, first I translated the second cube to the (1, 0, 0) too. Then I rotated the cube, and finally translated it to the above of the first cube.



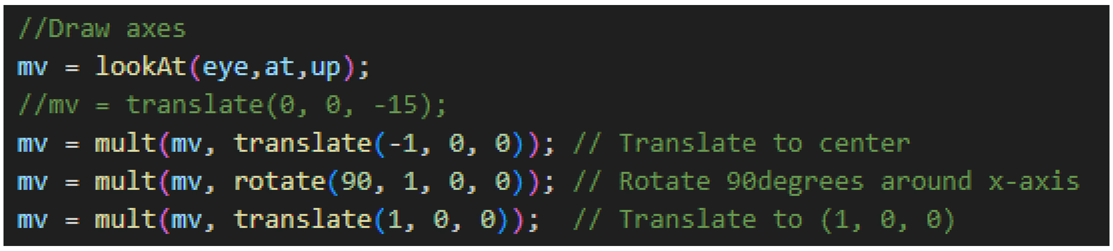
```
mv = lookAt(eye, at, up);             // Reset the mv
mv = mult(mv, translate(1, 0, 0));  // Translate to (1, 0, 0)
mv = mult(mv, rotate(45, 0, 1, 0)); // Rotate 45degrees around y-axis
mv = mult(mv, translate(0, 1, 0));  // Translate to (0, 1, 0)
```



9. To use orthographic projection, I should call the “ortho” function and adjust the values for the left, right, bottom, top, near, and far. For the left, right, bottom and up I selected the value of 3 because I found out that it is the minimum value that displays the objects correctly for my dimension of “400x400”. And for the near and far values, I selected the 1, and 100; those values do not clip any view.



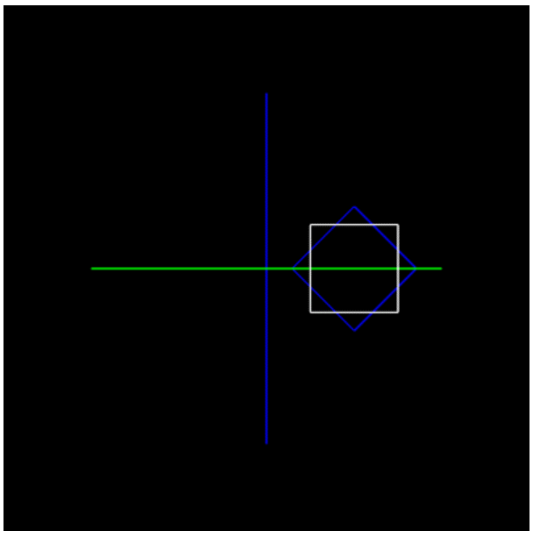
10. In order to do so, first, I need to translate and rotate the axes accordingly:



```
//Draw axes
mv = lookAt(eye,at,up);
//mv = translate(0, 0, -15);
mv = mult(mv, translate(-1, 0, 0)); // Translate to center
mv = mult(mv, rotate(90, 1, 0, 0)); // Rotate 90degrees around x-axis
mv = mult(mv, translate(1, 0, 0));  // Translate to (1, 0, 0)
```

By doing that, first I translate them to center so I can rotate them, and transfer back to their position back.

After doing that for axes, I do that for the second cube too. No need to do it for the first cube since I am creating it relative to the axes.



11. In order to rotate everything, I applied this both to the axes and cubes:

    ```
    mv = mult(mv, rotate(30, 1, 0, 0));
    mv = mult(mv, rotate(30, 0, 1, 0));
    ```

    Here, I first rotate them 30 degrees along the x-axis, then 30 degrees along y- axis. After doing this, I obtained an image where we can see all three of the axes and the objects rotated accordingly. Here is my result: