# COMP 370 assignment #6: WebGL Texture Mapping

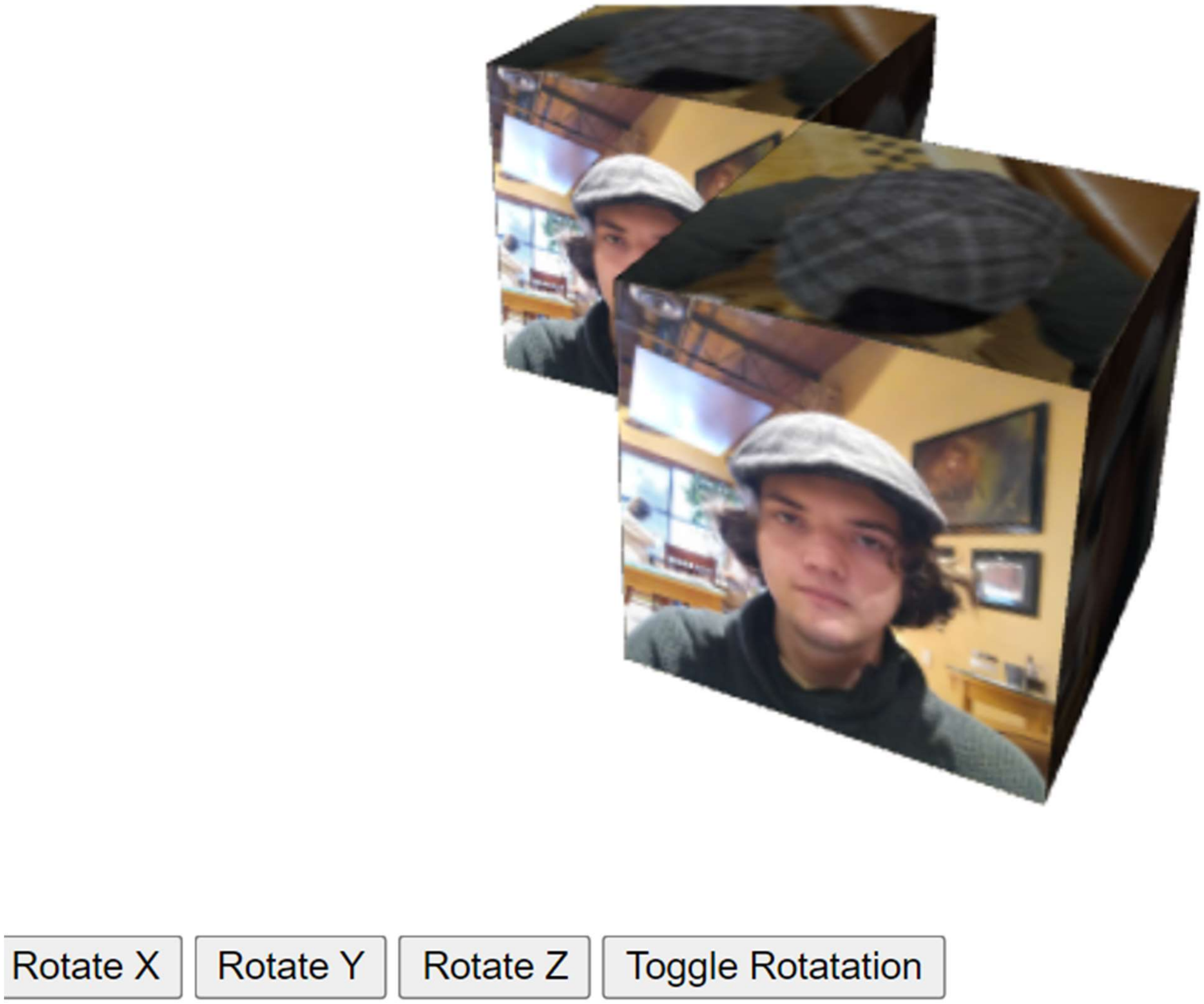
Thomas Williamson

id: 588206

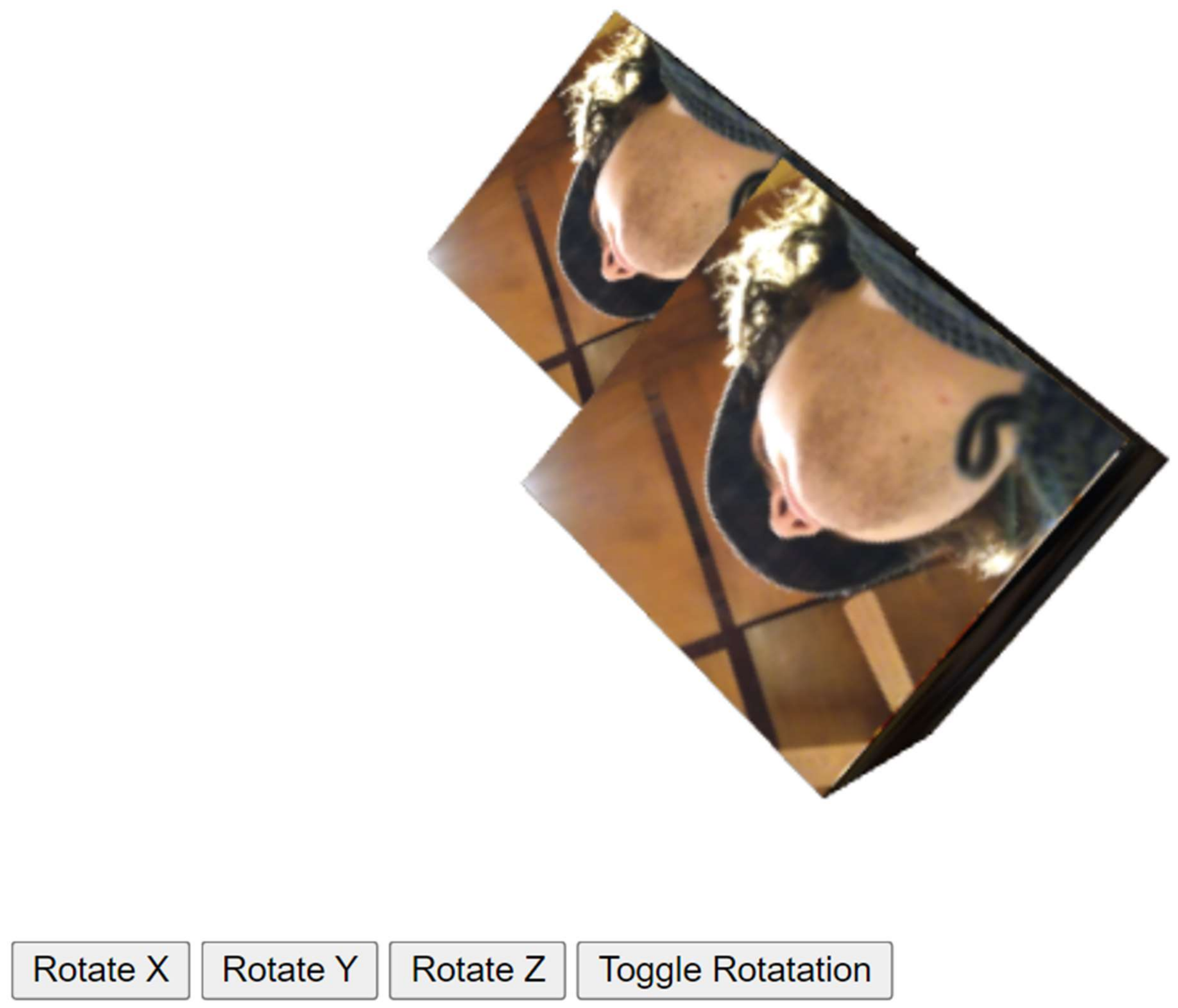
2021/12/01

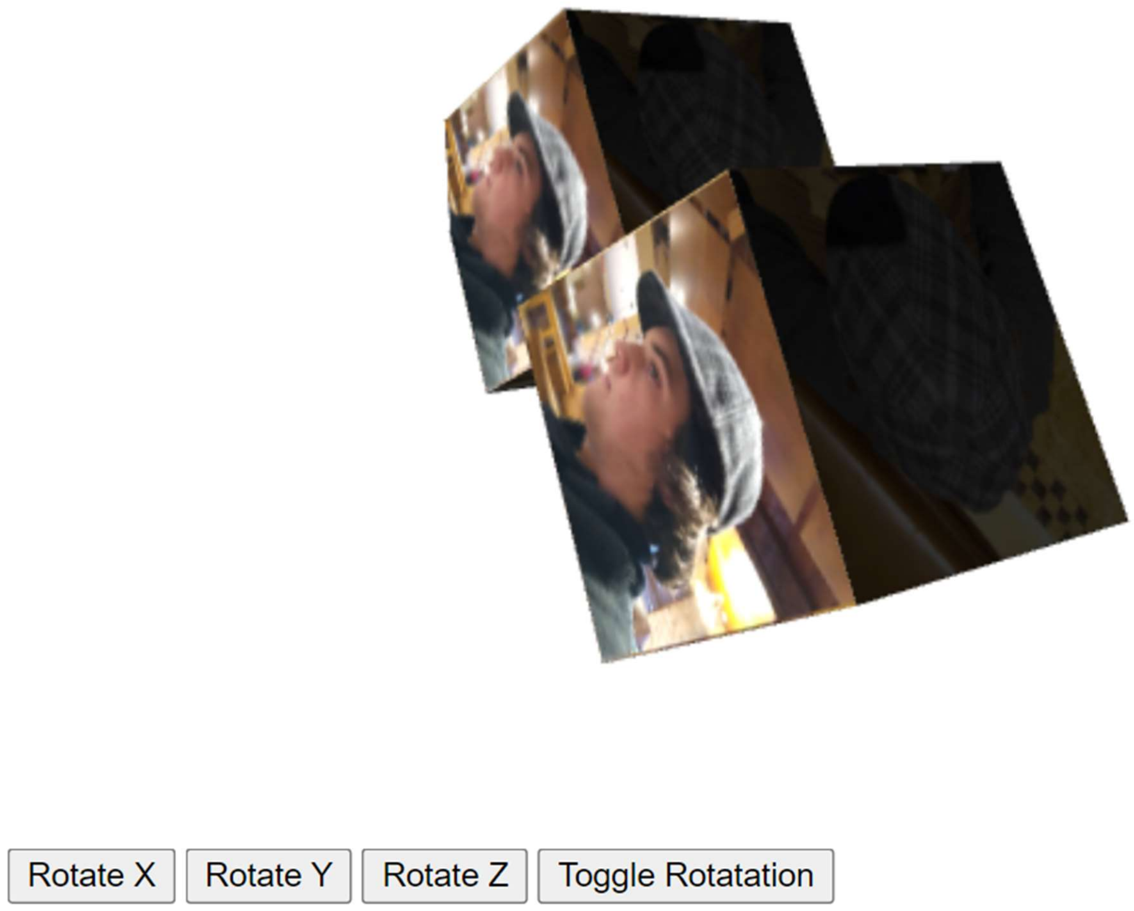
The goal of this project is to make a rotating 3d cube with texture mapping.

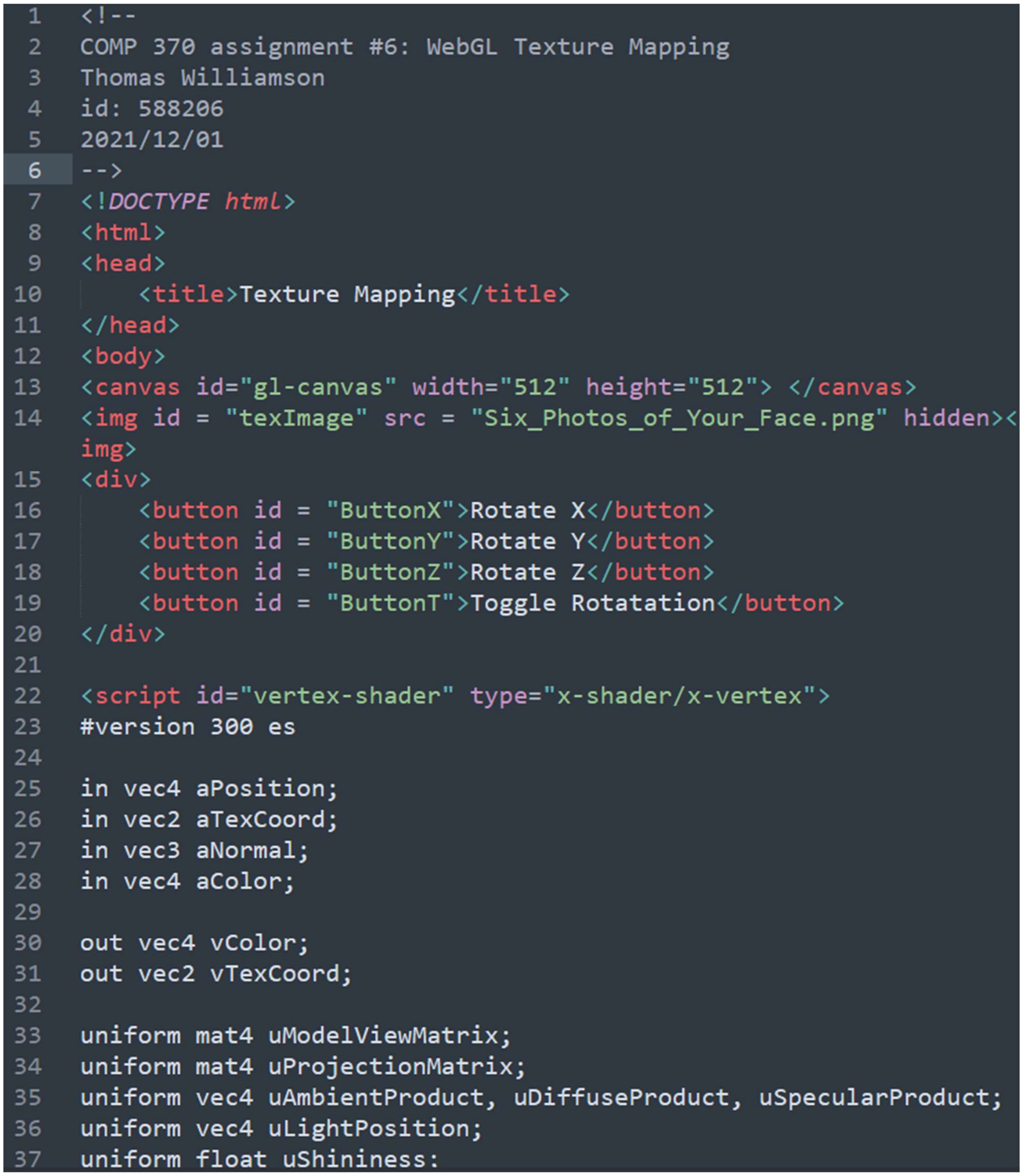
Startup

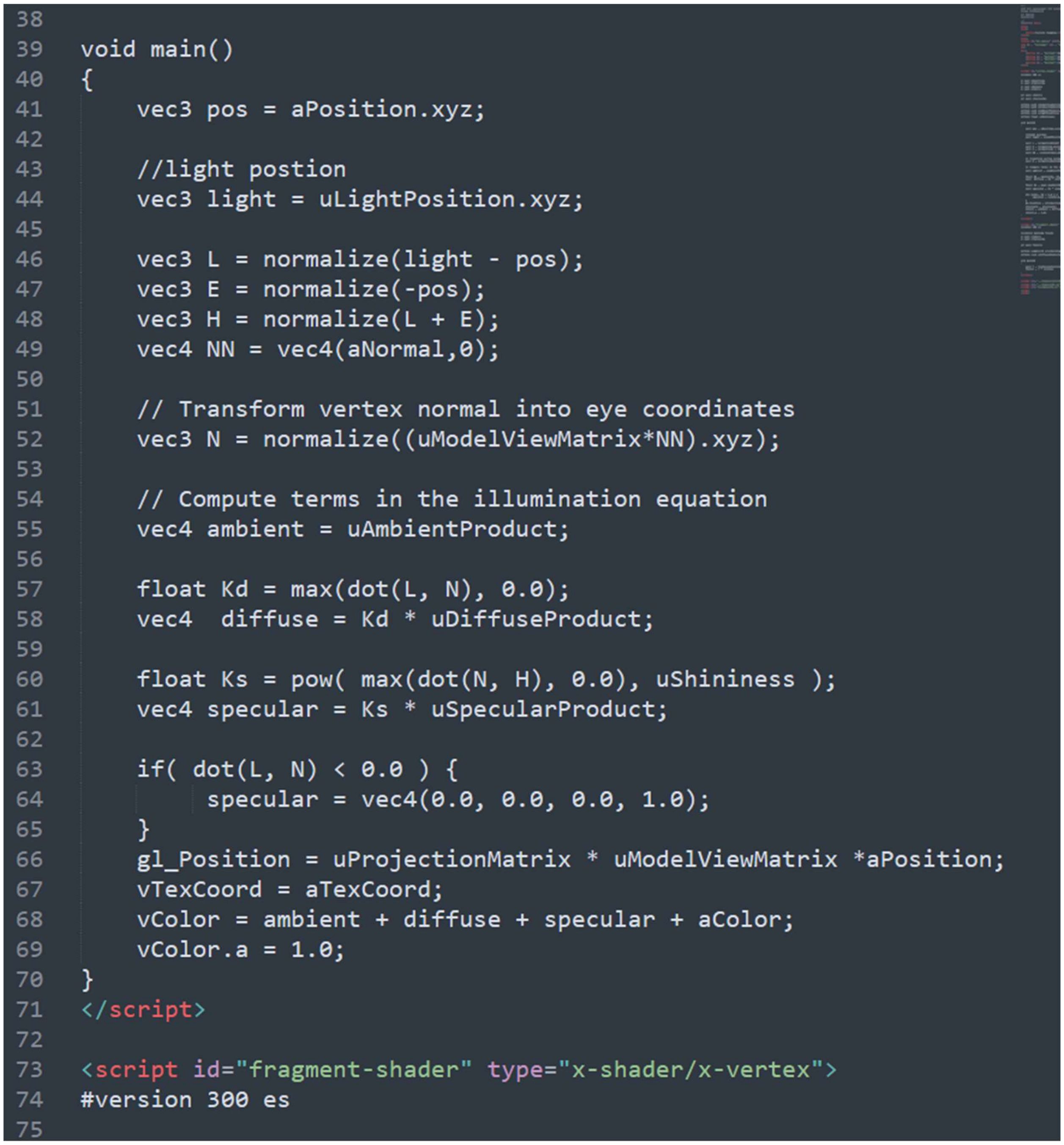

Rotate X
Rotate Y
Rotate Z
Toggle Rotatation

Rotate X
Rotate Y
Rotate Z
Toggle Rotatation

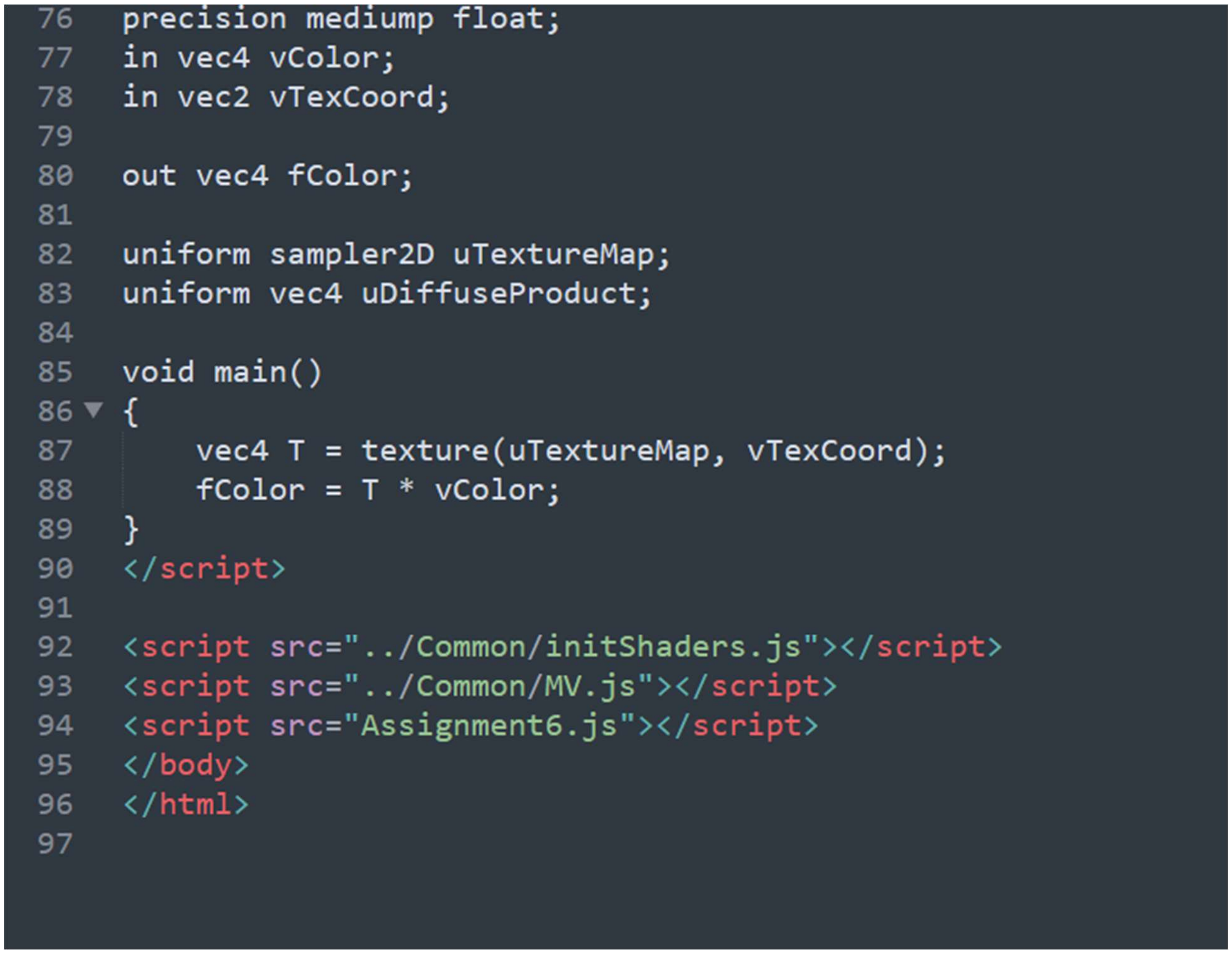

Rotate X
Rotate Y
Rotate Z
Toggle Rotatation


Rotate X
Rotate Y
Rotate Z
Toggle Rotatation

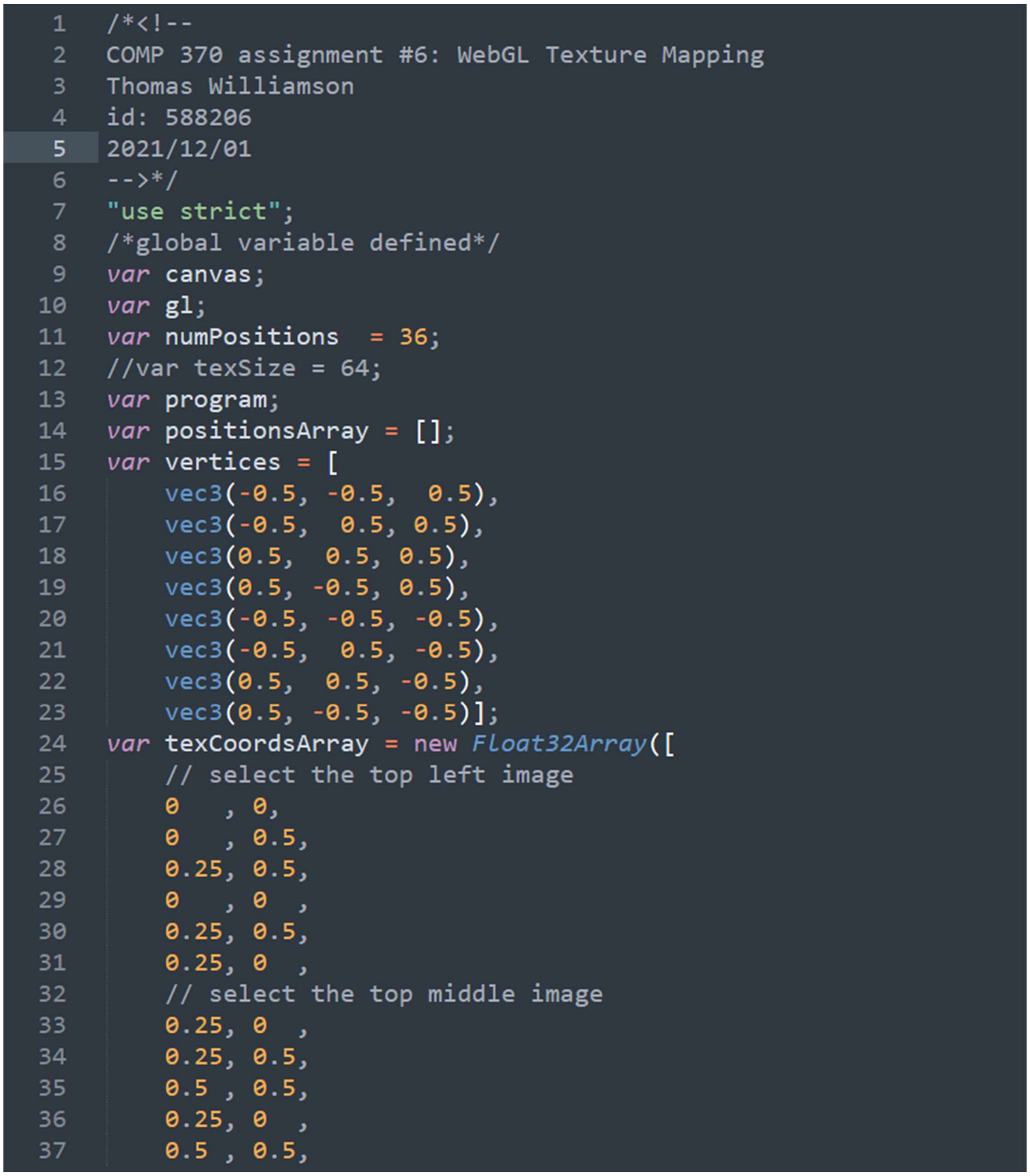

```
<!--
COMP 370 assignment #6: WebGL Texture Mapping
Thomas Williamson
id: 588206
2021/12/01
-->
<!DOCTYPE html>
<html>
<head>
    <title>Texture Mapping</title>
</head>
<body>
<canvas id="gl-canvas" width="512" height="512"> </canvas>
<img id = "texImage" src = "Six_Photos_of_Your_Face.png" hidden></img>
<div>
    <button id = "ButtonX">Rotate X</button>
    <button id = "ButtonY">Rotate Y</button>
    <button id = "ButtonZ">Rotate Z</button>
    <button id = "ButtonT">Toggle Rotatation</button>
</div>

<script id="vertex-shader" type="x-shader/x-vertex">
#version 300 es

in vec4 aPosition;
in vec2 aTexCoord;
in vec3 aNormal;
in vec4 aColor;

out vec4 vColor;
out vec2 vTexCoord;

uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
uniform vec4 uAmbientProduct, uDiffuseProduct, uSpecularProduct;
uniform vec4 uLightPosition;
uniform float uShininess;
```



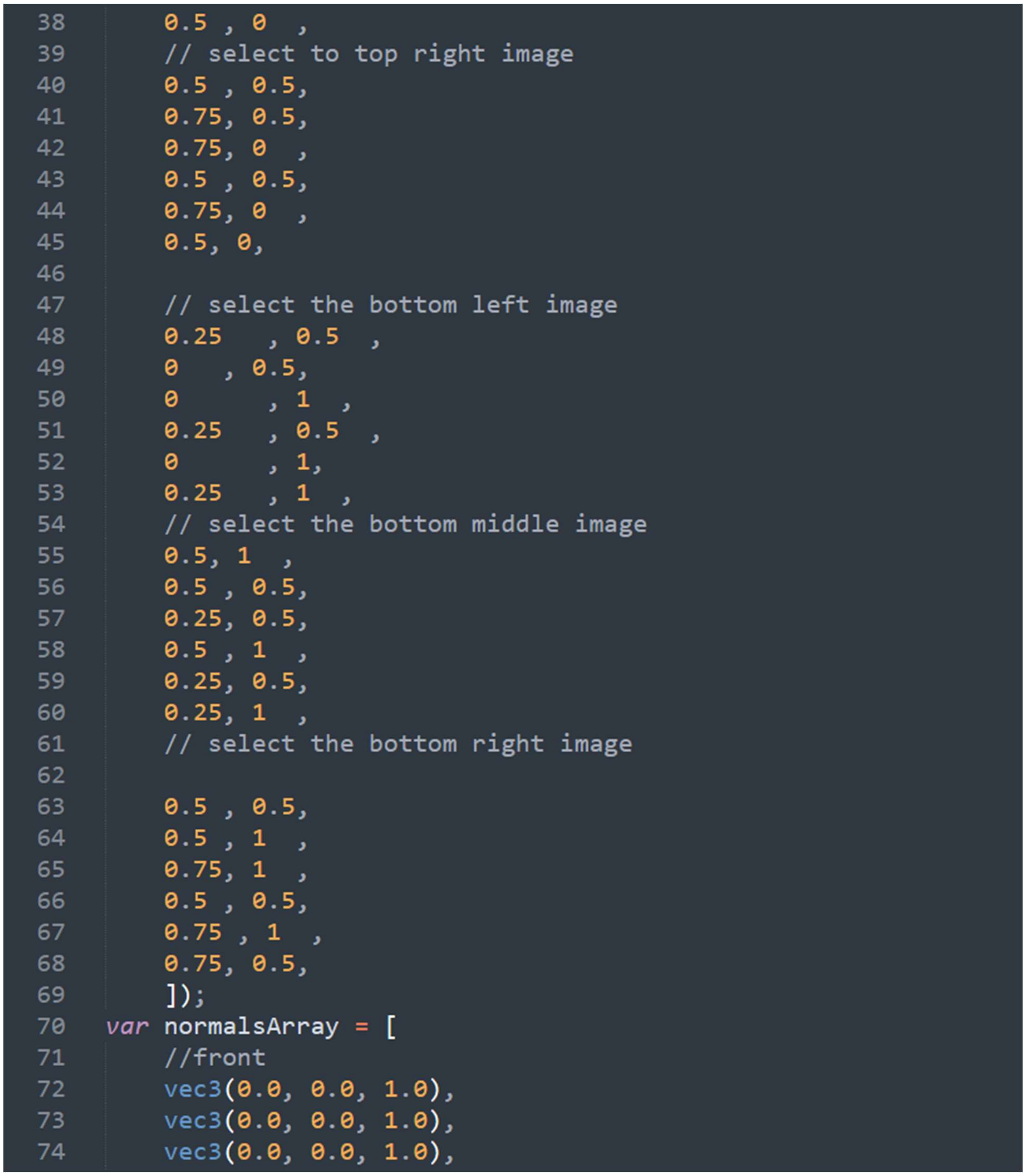
```

void main()
{
    vec3 pos = aPosition.xyz;

    //light postion
    vec3 light = uLightPosition.xyz;

    vec3 L = normalize(light - pos);
    vec3 E = normalize(-pos);
    vec3 H = normalize(L + E);
    vec4 NN = vec4(aNormal,0);

    // Transform vertex normal into eye coordinates
    vec3 N = normalize((uModelViewMatrix*NN).xyz);

    // Compute terms in the illumination equation
    vec4 ambient = uAmbientProduct;

    float Kd = max(dot(L, N), 0.0);
    vec4  diffuse = Kd * uDiffuseProduct;

    float Ks = pow( max(dot(N, H), 0.0), uShininess );
    vec4 specular = Ks * uSpecularProduct;

    if( dot(L, N) < 0.0 ) {
        specular = vec4(0.0, 0.0, 0.0, 1.0);
    }
    gl_Position = uProjectionMatrix * uModelViewMatrix *aPosition;
    vTexCoord = aTexCoord;
    vColor = ambient + diffuse + specular + aColor;
    vColor.a = 1.0;
}
</script>

<script id="fragment-shader" type="x-shader/x-vertex">
#version 300 es

```



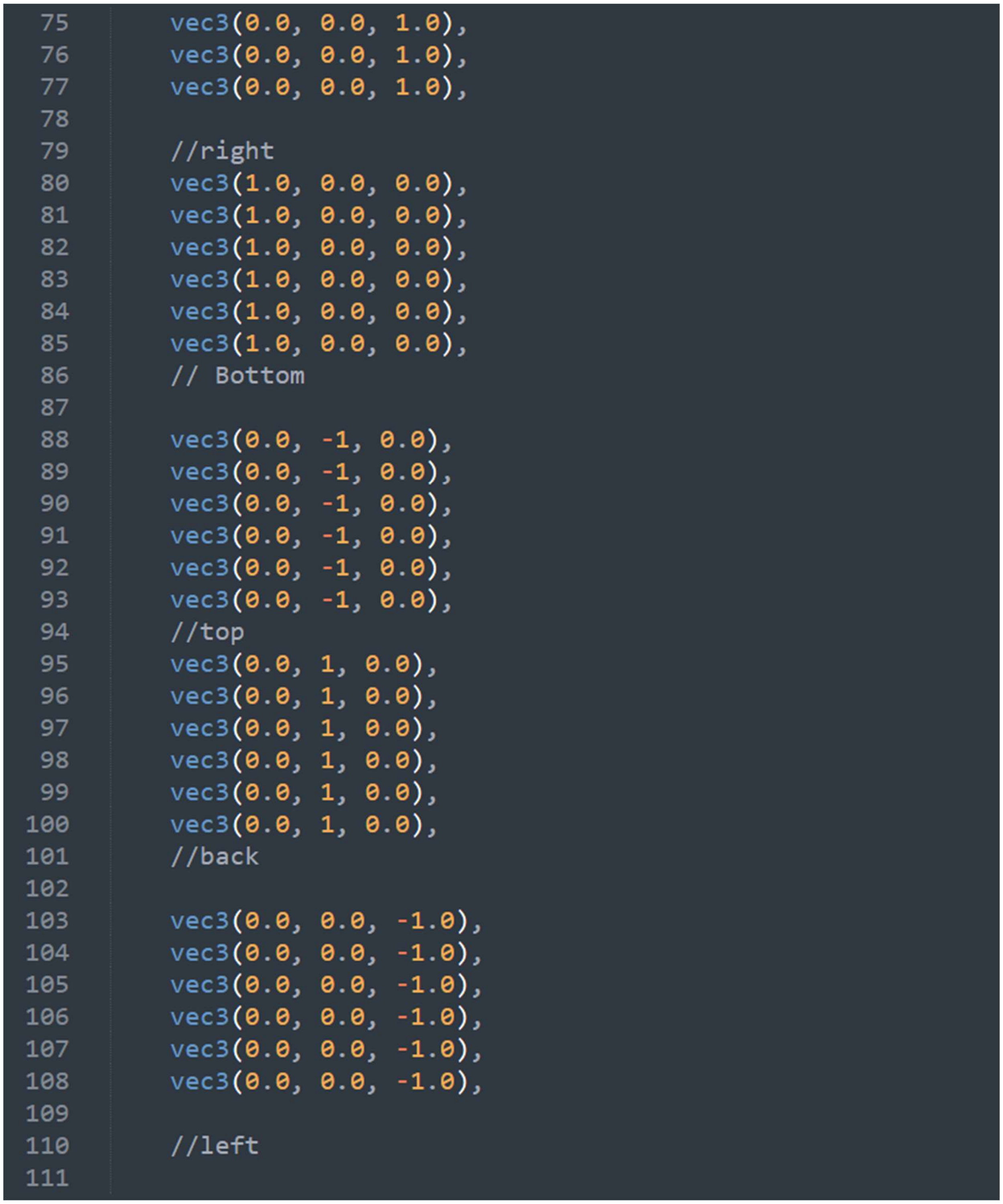
```
precision mediump float;
in vec4 vColor;
in vec2 vTexCoord;

out vec4 fColor;

uniform sampler2D uTextureMap;
uniform vec4 uDiffuseProduct;

void main()
{
    vec4 T = texture(uTextureMap, vTexCoord);
    fColor = T * vColor;
}
</script>

<script src="../Common/initShaders.js"></script>
<script src="../Common/MV.js"></script>
<script src="Assignment6.js"></script>
</body>
</html>

```



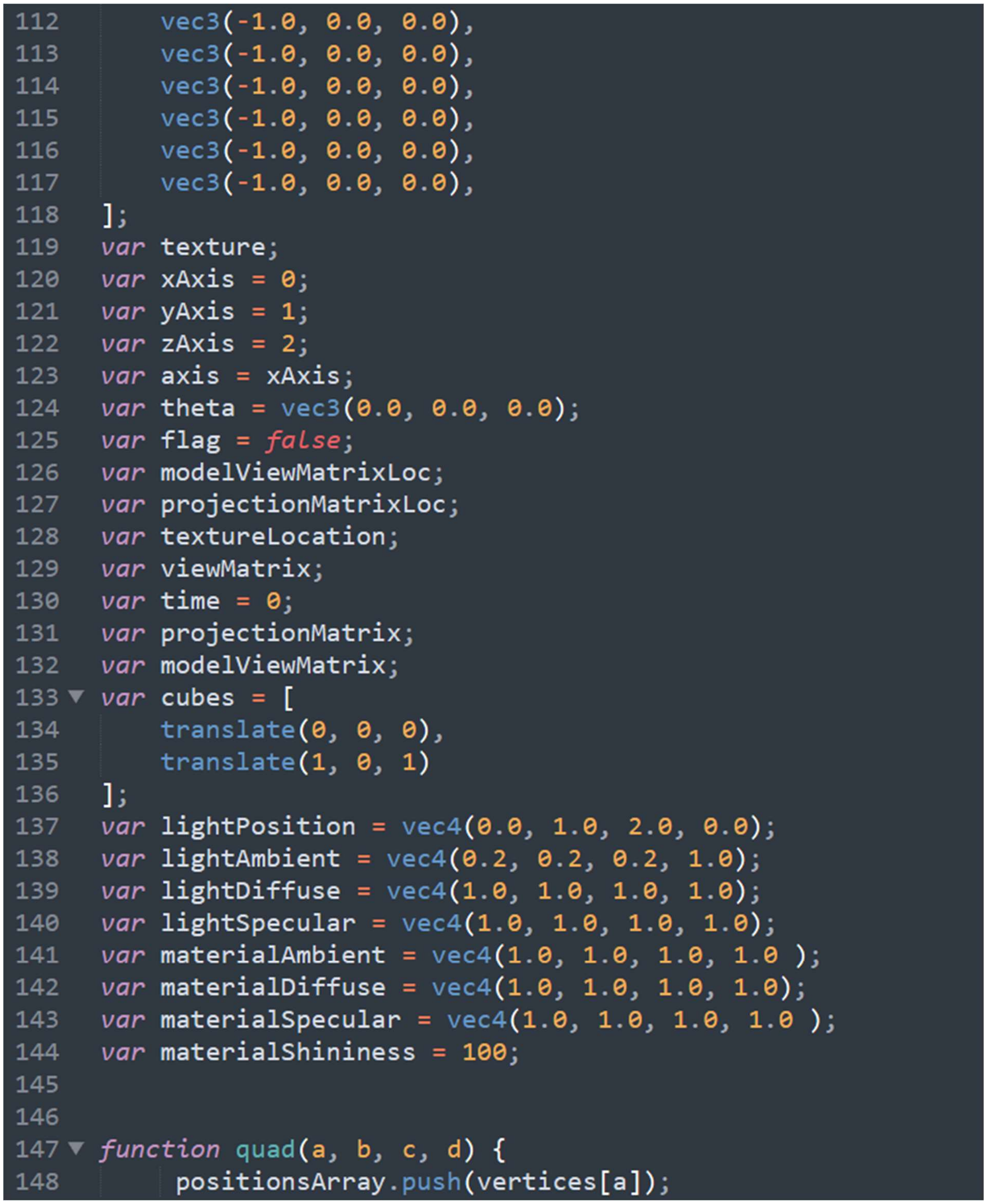
```
/*<!--
COMP 370 assignment #6: WebGL Texture Mapping
Thomas Williamson
id: 588206
2021/12/01
-->*/
"use strict";
/*global variable defined*/
var canvas;
var gl;
var numPositions  = 36;
//var texSize = 64;
var program;
var positionsArray = [];
var vertices = [
    vec3(-0.5, -0.5,  0.5),
    vec3(-0.5,  0.5, 0.5),
    vec3(0.5,  0.5, 0.5),
    vec3(0.5, -0.5, 0.5),
    vec3(-0.5, -0.5, -0.5),
    vec3(-0.5,  0.5, -0.5),
    vec3(0.5,  0.5, -0.5),
    vec3(0.5, -0.5, -0.5)];
var texCoordsArray = new Float32Array([
    // select the top left image
    0   , 0,
    0   , 0.5,
    0.25, 0.5,
    0   , 0  ,
    0.25, 0.5,
    0.25, 0  ,
    // select the top middle image
    0.25, 0  ,
    0.25, 0.5,
    0.5 , 0.5,
    0.25, 0  ,
    0.5 , 0.5,
```



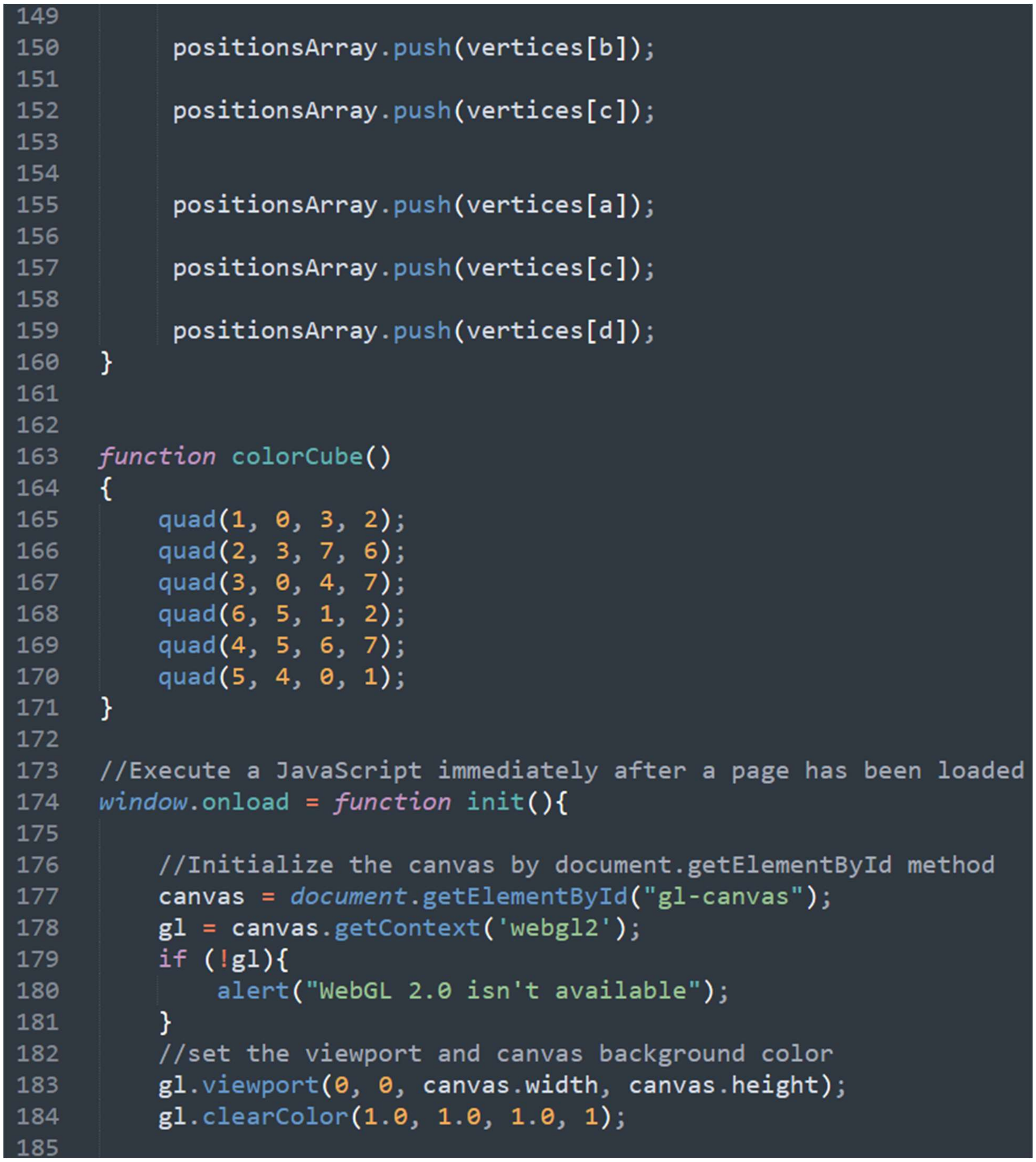
```
    0.5 , 0  ,
    // select to top right image
    0.5 , 0.5,
    0.75, 0.5,
    0.75, 0  ,
    0.5 , 0.5,
    0.75, 0  ,
    0.5, 0,

    // select the bottom left image
    0.25   , 0.5  ,
    0   , 0.5,
    0      , 1  ,
    0.25   , 0.5  ,
    0      , 1,
    0.25   , 1  ,
    // select the bottom middle image
    0.5, 1  ,
    0.5 , 0.5,
    0.25, 0.5,
    0.5 , 1  ,
    0.25, 0.5,
    0.25, 1  ,
    // select the bottom right image

    0.5 , 0.5,
    0.5 , 1  ,
    0.75, 1  ,
    0.5 , 0.5,
    0.75 , 1  ,
    0.75, 0.5,
    ]);
var normalsArray = [
    //front
    vec3(0.0, 0.0, 1.0),
    vec3(0.0, 0.0, 1.0),
    vec3(0.0, 0.0, 1.0),
```



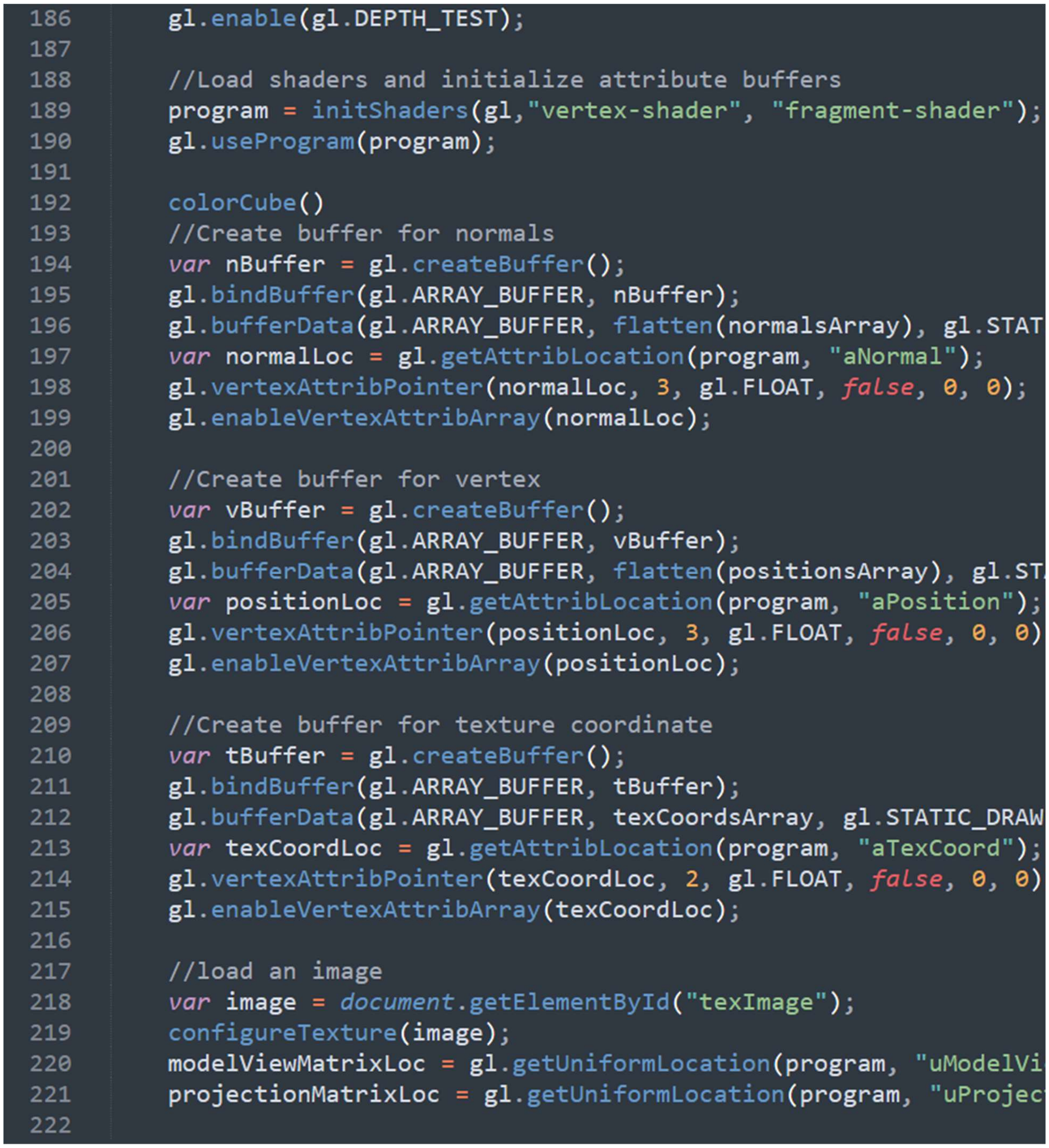
```
    vec3(0.0, 0.0, 1.0),
    vec3(0.0, 0.0, 1.0),
    vec3(0.0, 0.0, 1.0),

    //right
    vec3(1.0, 0.0, 0.0),
    vec3(1.0, 0.0, 0.0),
    vec3(1.0, 0.0, 0.0),
    vec3(1.0, 0.0, 0.0),
    vec3(1.0, 0.0, 0.0),
    vec3(1.0, 0.0, 0.0),
    // Bottom

    vec3(0.0, -1, 0.0),
    vec3(0.0, -1, 0.0),
    vec3(0.0, -1, 0.0),
    vec3(0.0, -1, 0.0),
    vec3(0.0, -1, 0.0),
    vec3(0.0, -1, 0.0),
    //top
    vec3(0.0, 1, 0.0),
    vec3(0.0, 1, 0.0),
    vec3(0.0, 1, 0.0),
    vec3(0.0, 1, 0.0),
    vec3(0.0, 1, 0.0),
    vec3(0.0, 1, 0.0),
    //back

    vec3(0.0, 0.0, -1.0),
    vec3(0.0, 0.0, -1.0),
    vec3(0.0, 0.0, -1.0),
    vec3(0.0, 0.0, -1.0),
    vec3(0.0, 0.0, -1.0),
    vec3(0.0, 0.0, -1.0),

    //left

```



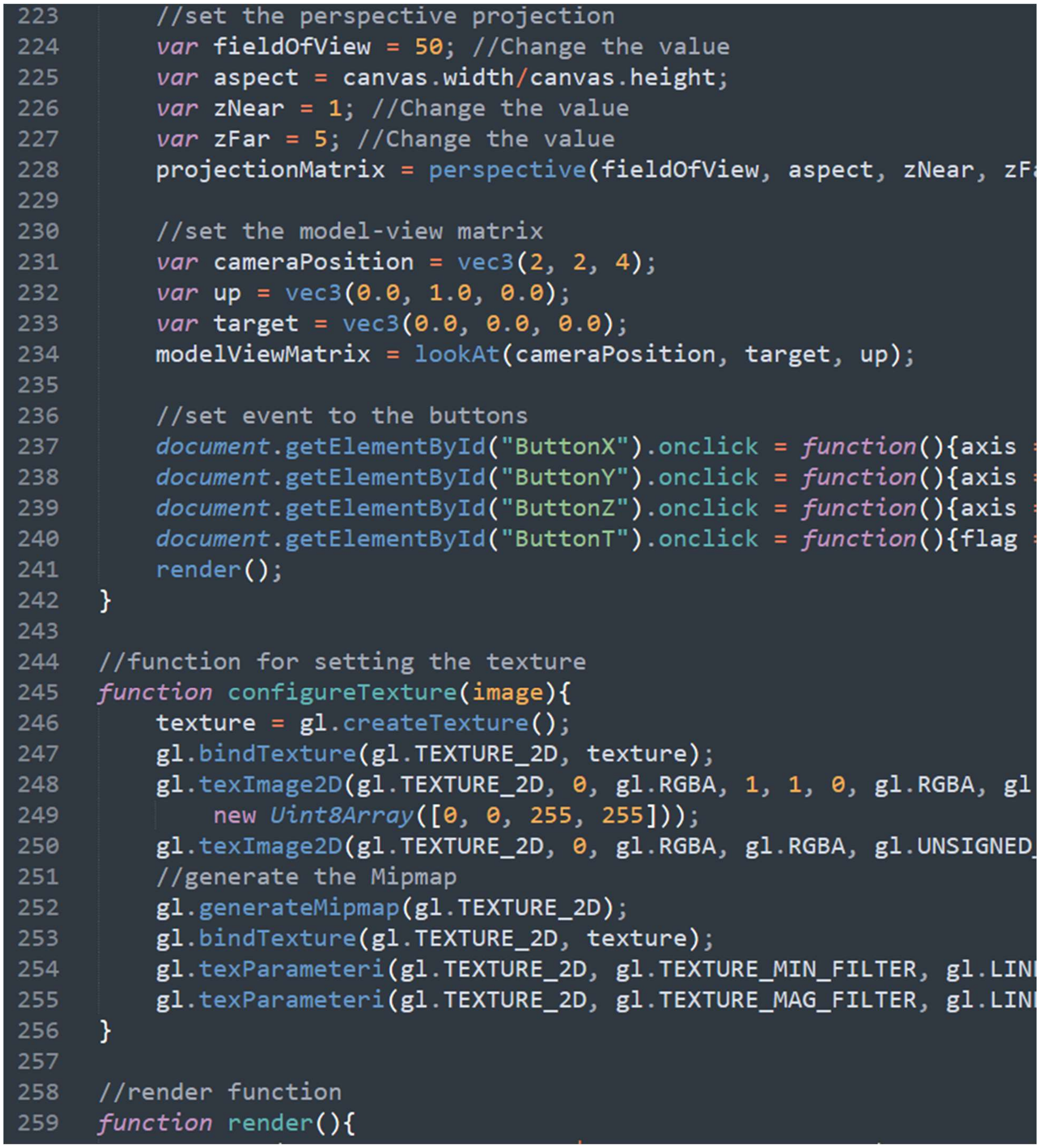
```
    vec3(-1.0, 0.0, 0.0),
    vec3(-1.0, 0.0, 0.0),
    vec3(-1.0, 0.0, 0.0),
    vec3(-1.0, 0.0, 0.0),
    vec3(-1.0, 0.0, 0.0),
    vec3(-1.0, 0.0, 0.0),
];
var texture;
var xAxis = 0;
var yAxis = 1;
var zAxis = 2;
var axis = xAxis;
var theta = vec3(0.0, 0.0, 0.0);
var flag = false;
var modelViewMatrixLoc;
var projectionMatrixLoc;
var textureLocation;
var viewMatrix;
var time = 0;
var projectionMatrix;
var modelViewMatrix;
var cubes = [
    translate(0, 0, 0),
    translate(1, 0, 1)
];
var lightPosition = vec4(0.0, 1.0, 2.0, 0.0);
var lightAmbient = vec4(0.2, 0.2, 0.2, 1.0);
var lightDiffuse = vec4(1.0, 1.0, 1.0, 1.0);
var lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);
var materialAmbient = vec4(1.0, 1.0, 1.0, 1.0 );
var materialDiffuse = vec4(1.0, 1.0, 1.0, 1.0);
var materialSpecular = vec4(1.0, 1.0, 1.0, 1.0 );
var materialShininess = 100;


function quad(a, b, c, d) {
     positionsArray.push(vertices[a]);
```



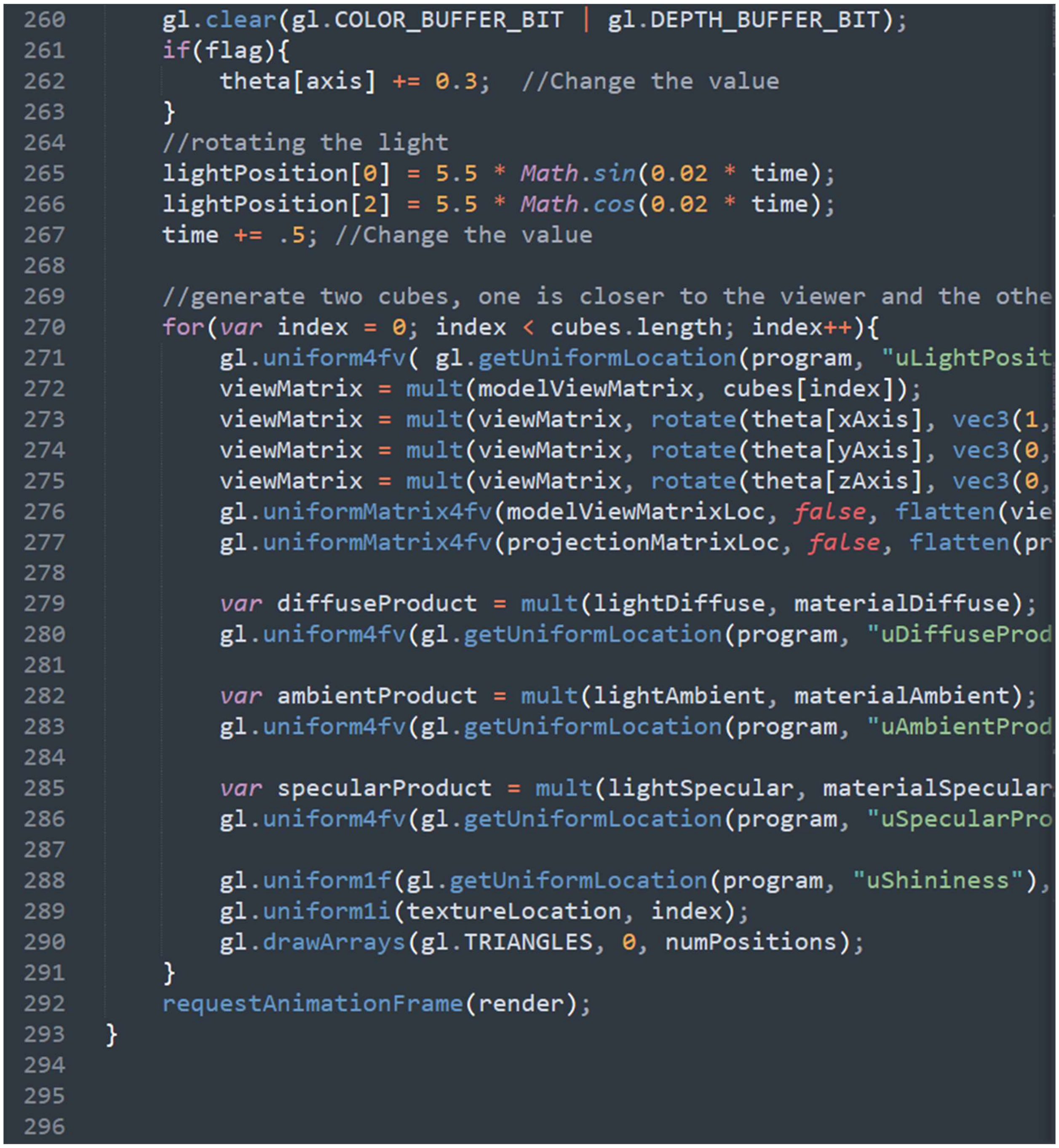
```
        positionsArray.push(vertices[b]);

        positionsArray.push(vertices[c]);


        positionsArray.push(vertices[a]);

        positionsArray.push(vertices[c]);

        positionsArray.push(vertices[d]);
}


function colorCube()
{
    quad(1, 0, 3, 2);
    quad(2, 3, 7, 6);
    quad(3, 0, 4, 7);
    quad(6, 5, 1, 2);
    quad(4, 5, 6, 7);
    quad(5, 4, 0, 1);
}

//Execute a JavaScript immediately after a page has been loaded
window.onload = function init(){

    //Initialize the canvas by document.getElementById method
    canvas = document.getElementById("gl-canvas");
    gl = canvas.getContext('webgl2');
    if (!gl){
        alert("WebGL 2.0 isn't available");
    }
    //set the viewport and canvas background color
    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(1.0, 1.0, 1.0, 1);

```



```
    gl.enable(gl.DEPTH_TEST);

    //Load shaders and initialize attribute buffers
    program = initShaders(gl,"vertex-shader", "fragment-shader");
    gl.useProgram(program);

    colorCube()
    //Create buffer for normals
    var nBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, nBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(normalsArray), gl.STAT
    var normalLoc = gl.getAttribLocation(program, "aNormal");
    gl.vertexAttribPointer(normalLoc, 3, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(normalLoc);

    //Create buffer for vertex
    var vBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(positionsArray), gl.ST
    var positionLoc = gl.getAttribLocation(program, "aPosition");
    gl.vertexAttribPointer(positionLoc, 3, gl.FLOAT, false, 0, 0)
    gl.enableVertexAttribArray(positionLoc);

    //Create buffer for texture coordinate
    var tBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, tBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, texCoordsArray, gl.STATIC_DRAW
    var texCoordLoc = gl.getAttribLocation(program, "aTexCoord");
    gl.vertexAttribPointer(texCoordLoc, 2, gl.FLOAT, false, 0, 0)
    gl.enableVertexAttribArray(texCoordLoc);

    //load an image
    var image = document.getElementById("texImage");
    configureTexture(image);
    modelViewMatrixLoc = gl.getUniformLocation(program, "uModelVi
    projectionMatrixLoc = gl.getUniformLocation(program, "uProjec

```



```
    //set the perspective projection
    var fieldOfView = 50; //Change the value
    var aspect = canvas.width/canvas.height;
    var zNear = 1; //Change the value
    var zFar = 5; //Change the value
    projectionMatrix = perspective(fieldOfView, aspect, zNear, zF

    //set the model-view matrix
    var cameraPosition = vec3(2, 2, 4);
    var up = vec3(0.0, 1.0, 0.0);
    var target = vec3(0.0, 0.0, 0.0);
    modelViewMatrix = lookAt(cameraPosition, target, up);

    //set event to the buttons
    document.getElementById("ButtonX").onclick = function(){axis
    document.getElementById("ButtonY").onclick = function(){axis
    document.getElementById("ButtonZ").onclick = function(){axis
    document.getElementById("ButtonT").onclick = function(){flag
    render();
}

//function for setting the texture
function configureTexture(image){
    texture = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0, gl.RGBA, gl
        new Uint8Array([0, 0, 255, 255]));
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_
    //generate the Mipmap
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LIN
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LIN
}

//render function
function render(){
```



```
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    if(flag){
        theta[axis] += 0.3;  //Change the value
    }
    //rotating the light
    lightPosition[0] = 5.5 * Math.sin(0.02 * time);
    lightPosition[2] = 5.5 * Math.cos(0.02 * time);
    time += .5; //Change the value

    //generate two cubes, one is closer to the viewer and the othe
    for(var index = 0; index < cubes.length; index++){
        gl.uniform4fv( gl.getUniformLocation(program, "uLightPosit
        viewMatrix = mult(modelViewMatrix, cubes[index]);
        viewMatrix = mult(viewMatrix, rotate(theta[xAxis], vec3(1,
        viewMatrix = mult(viewMatrix, rotate(theta[yAxis], vec3(0,
        viewMatrix = mult(viewMatrix, rotate(theta[zAxis], vec3(0,
        gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(vie
        gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(pr

        var diffuseProduct = mult(lightDiffuse, materialDiffuse);
        gl.uniform4fv(gl.getUniformLocation(program, "uDiffuseProd

        var ambientProduct = mult(lightAmbient, materialAmbient);
        gl.uniform4fv(gl.getUniformLocation(program, "uAmbientProd

        var specularProduct = mult(lightSpecular, materialSpecular
        gl.uniform4fv(gl.getUniformLocation(program, "uSpecularPro

        gl.uniform1f(gl.getUniformLocation(program, "uShininess"),
        gl.uniform1i(textureLocation, index);
        gl.drawArrays(gl.TRIANGLES, 0, numPositions);
    }
    requestAnimationFrame(render);
}
```