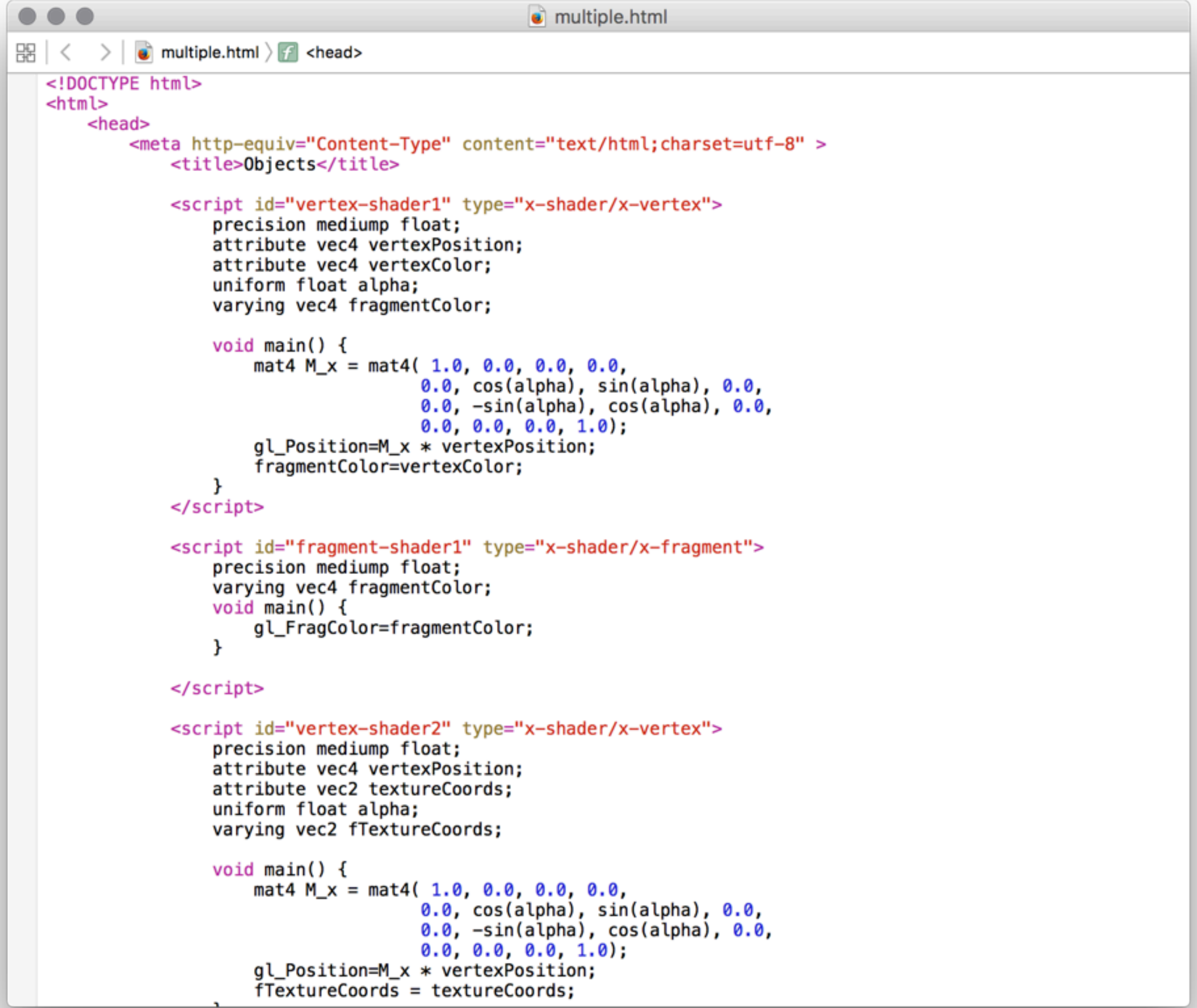
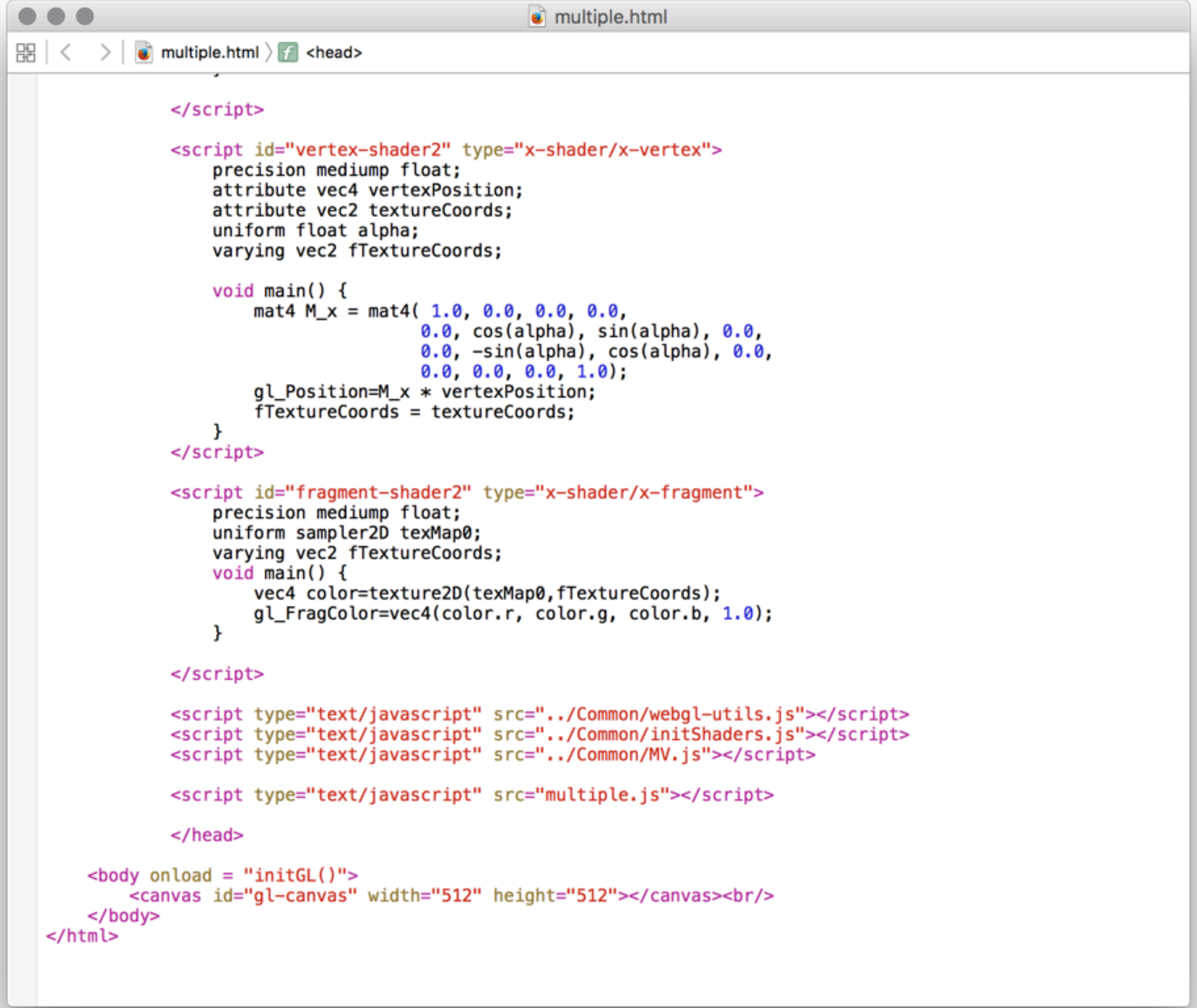
The following code renders a rotating cube and tetrahedron. The cube is multi-colored and the tetrahedron is textured with a random texture.

HTML File



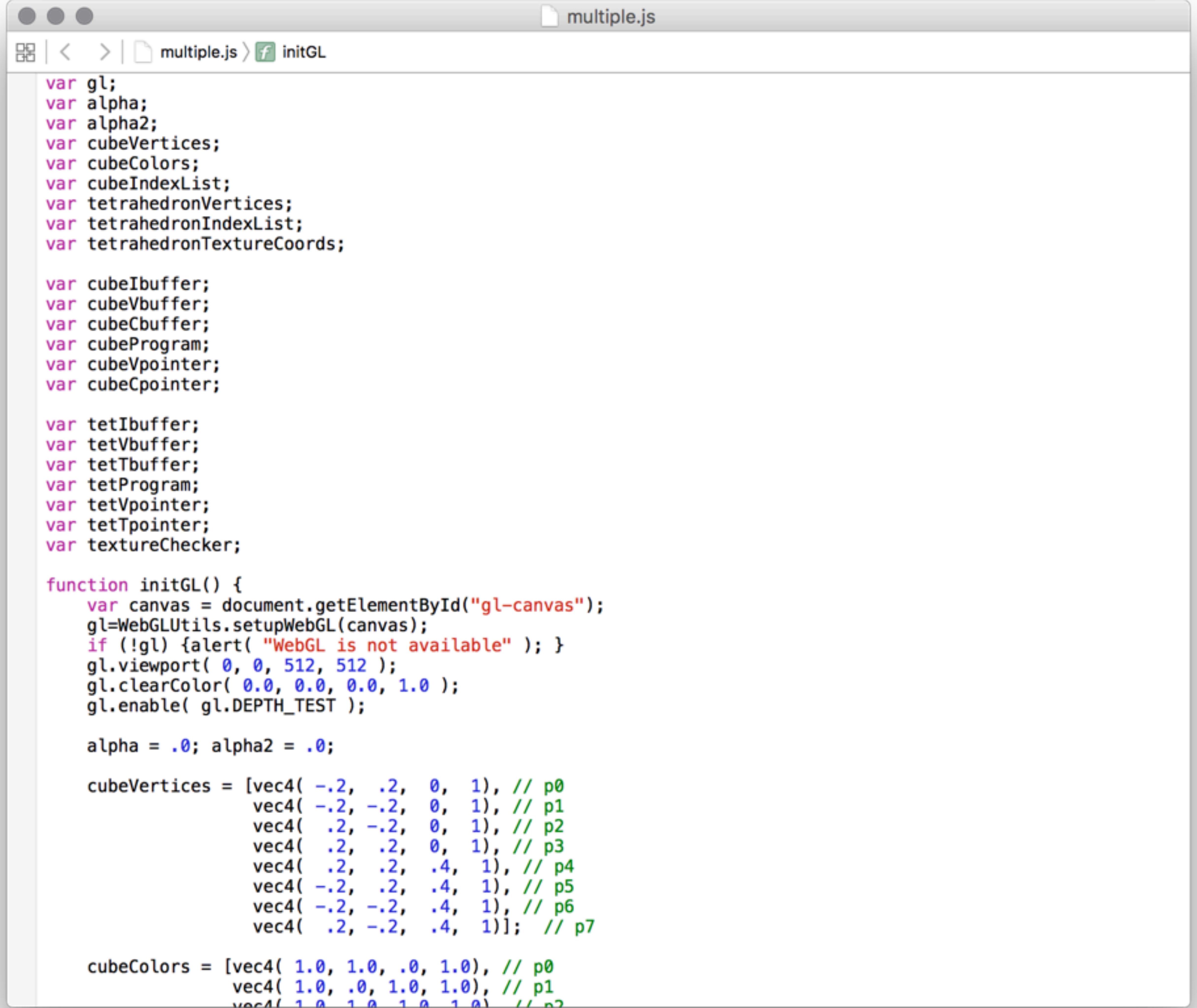
```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;charset=utf-8" >
            <title>Objects</title>

            <script id="vertex-shader1" type="x-shader/x-vertex">
                precision mediump float;
                attribute vec4 vertexPosition;
                attribute vec4 vertexColor;
                uniform float alpha;
                varying vec4 fragmentColor;

                void main() {
                    mat4 M_x = mat4( 1.0, 0.0, 0.0, 0.0,
                                    0.0, cos(alpha), sin(alpha), 0.0,
                                    0.0, -sin(alpha), cos(alpha), 0.0,
                                    0.0, 0.0, 0.0, 1.0);
                    gl_Position=M_x * vertexPosition;
                    fragmentColor=vertexColor;
                }
            </script>

            <script id="fragment-shader1" type="x-shader/x-fragment">
                precision mediump float;
                varying vec4 fragmentColor;
                void main() {
                    gl_FragColor=fragmentColor;
                }

            </script>

            <script id="vertex-shader2" type="x-shader/x-vertex">
                precision mediump float;
                attribute vec4 vertexPosition;
                attribute vec2 textureCoords;
                uniform float alpha;
                varying vec2 fTextureCoords;

                void main() {
                    mat4 M_x = mat4( 1.0, 0.0, 0.0, 0.0,
                                    0.0, cos(alpha), sin(alpha), 0.0,
                                    0.0, -sin(alpha), cos(alpha), 0.0,
                                    0.0, 0.0, 0.0, 1.0);
                    gl_Position=M_x * vertexPosition;
                    fTextureCoords = textureCoords;
```


multiple.html

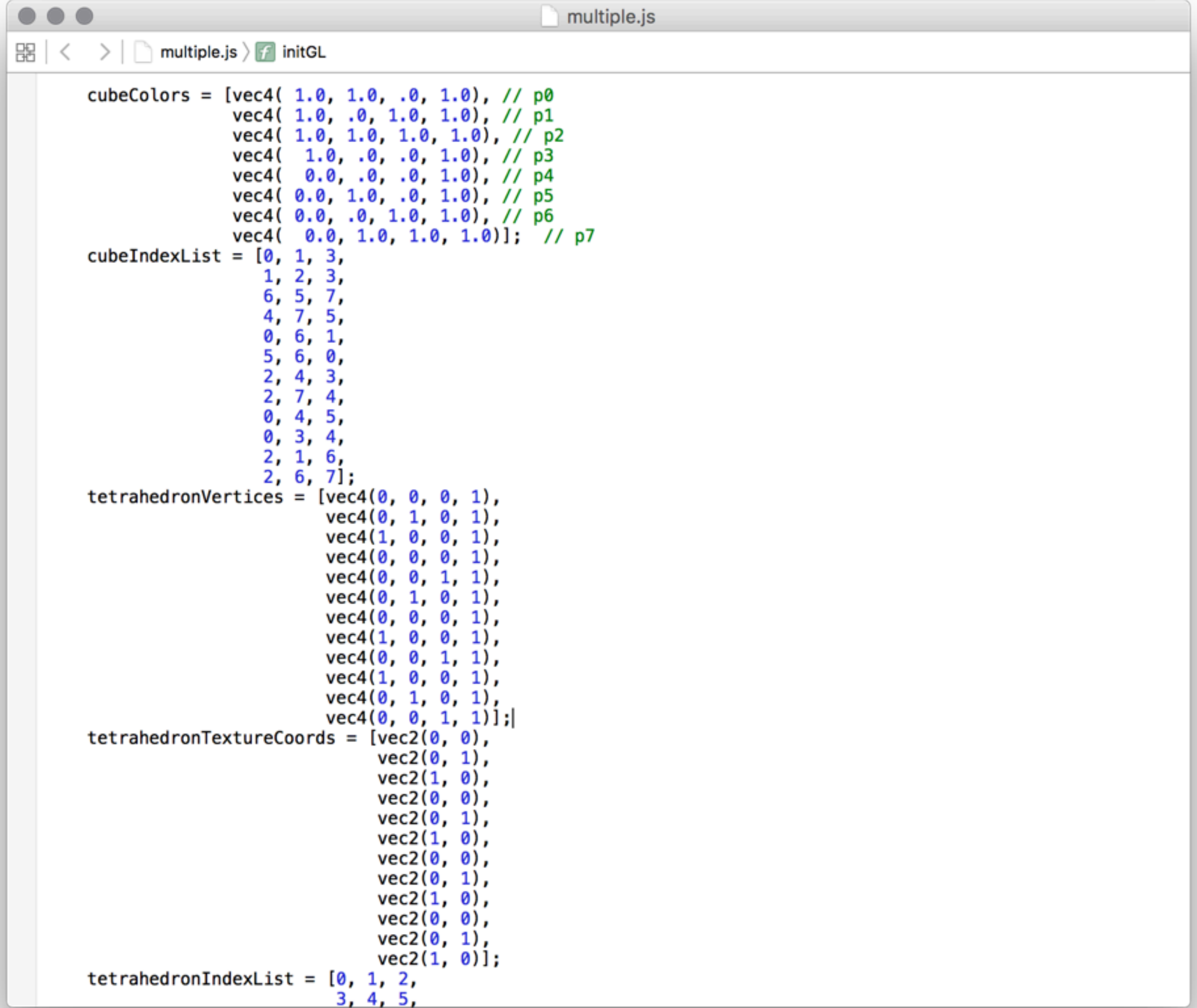
multiple.html 〉 <head>

```
        </script>

        <script id="vertex-shader2" type="x-shader/x-vertex">
            precision mediump float;
            attribute vec4 vertexPosition;
            attribute vec2 textureCoords;
            uniform float alpha;
            varying vec2 fTextureCoords;

            void main() {
                mat4 M_x = mat4( 1.0, 0.0, 0.0, 0.0,
                                0.0, cos(alpha), sin(alpha), 0.0,
                                0.0, -sin(alpha), cos(alpha), 0.0,
                                0.0, 0.0, 0.0, 1.0);
                gl_Position=M_x * vertexPosition;
                fTextureCoords = textureCoords;
            }
        </script>

        <script id="fragment-shader2" type="x-shader/x-fragment">
            precision mediump float;
            uniform sampler2D texMap0;
            varying vec2 fTextureCoords;
            void main() {
                vec4 color=texture2D(texMap0,fTextureCoords);
                gl_FragColor=vec4(color.r, color.g, color.b, 1.0);
            }

        </script>

        <script type="text/javascript" src="../Common/webgl-utils.js"></script>
        <script type="text/javascript" src="../Common/initShaders.js"></script>
        <script type="text/javascript" src="../Common/MV.js"></script>

        <script type="text/javascript" src="multiple.js"></script>

        </head>

    <body onload = "initGL()">
        <canvas id="gl-canvas" width="512" height="512"></canvas><br/>
    </body>
</html>
```

JavaScript File



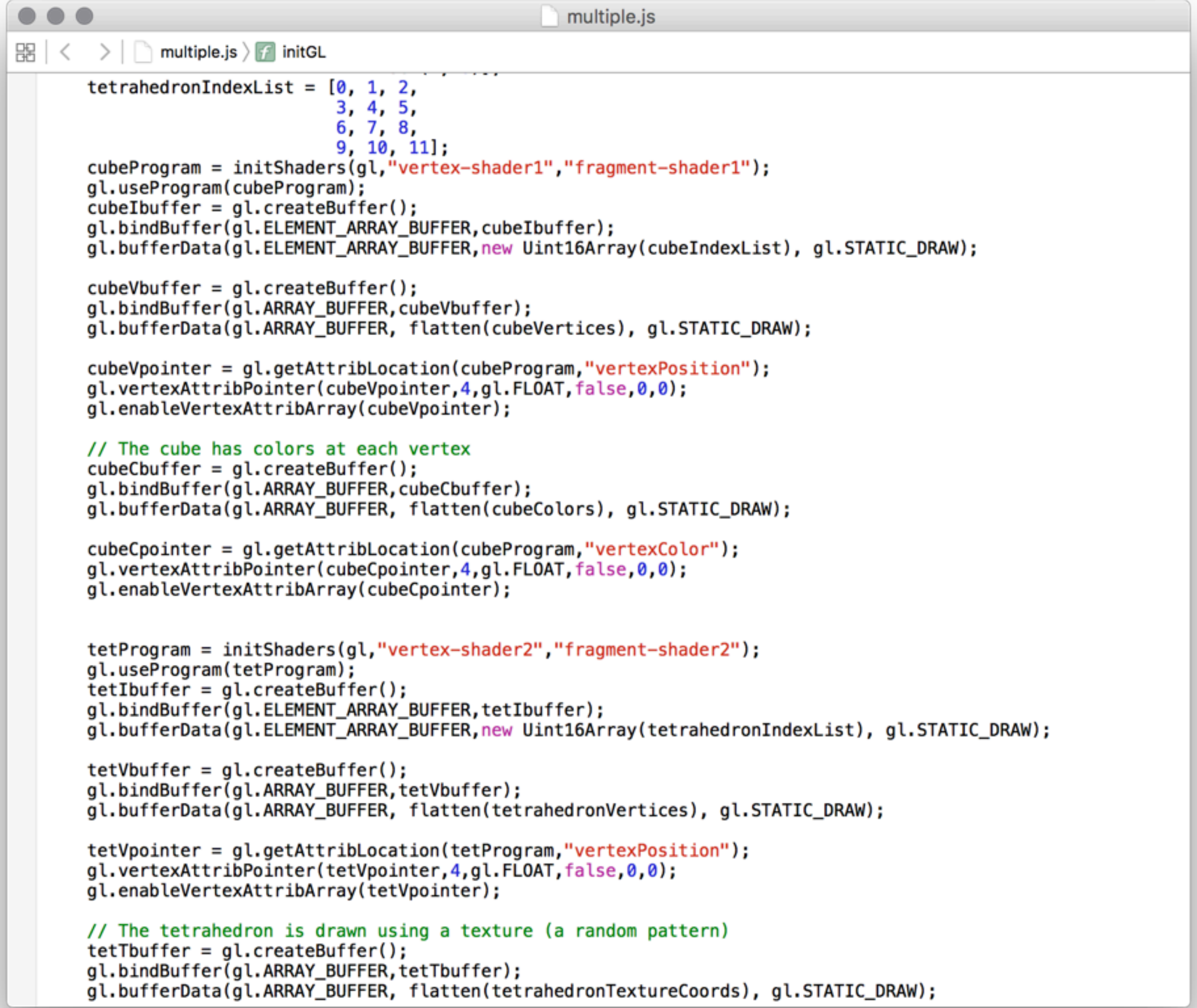
```
var gl;
var alpha;
var alpha2;
var cubeVertices;
var cubeColors;
var cubeIndexList;
var tetrahedronVertices;
var tetrahedronIndexList;
var tetrahedronTextureCoords;

var cubeIbuffer;
var cubeVbuffer;
var cubeCbuffer;
var cubeProgram;
var cubeVpointer;
var cubeCpointer;

var tetIbuffer;
var tetVbuffer;
var tetTbuffer;
var tetProgram;
var tetVpointer;
var tetTpointer;
var textureChecker;

function initGL() {
    var canvas = document.getElementById("gl-canvas");
    gl=WebGLUtils.setupWebGL(canvas);
    if (!gl) {alert( "WebGL is not available" ); }
    gl.viewport( 0, 0, 512, 512 );
    gl.clearColor( 0.0, 0.0, 0.0, 1.0 );
    gl.enable( gl.DEPTH_TEST );

    alpha = .0; alpha2 = .0;

    cubeVertices = [vec4( -.2,  .2,  0,  1), // p0
                    vec4( -.2, -.2,  0,  1), // p1
                    vec4(  .2, -.2,  0,  1), // p2
                    vec4(  .2,  .2,  0,  1), // p3
                    vec4(  .2,  .2,  .4,  1), // p4
                    vec4( -.2,  .2,  .4,  1), // p5
                    vec4( -.2, -.2,  .4,  1), // p6
                    vec4(  .2, -.2,  .4,  1)];  // p7

    cubeColors = [vec4( 1.0, 1.0, .0, 1.0), // p0
                  vec4( 1.0, .0, 1.0, 1.0), // p1
```



multiple.js

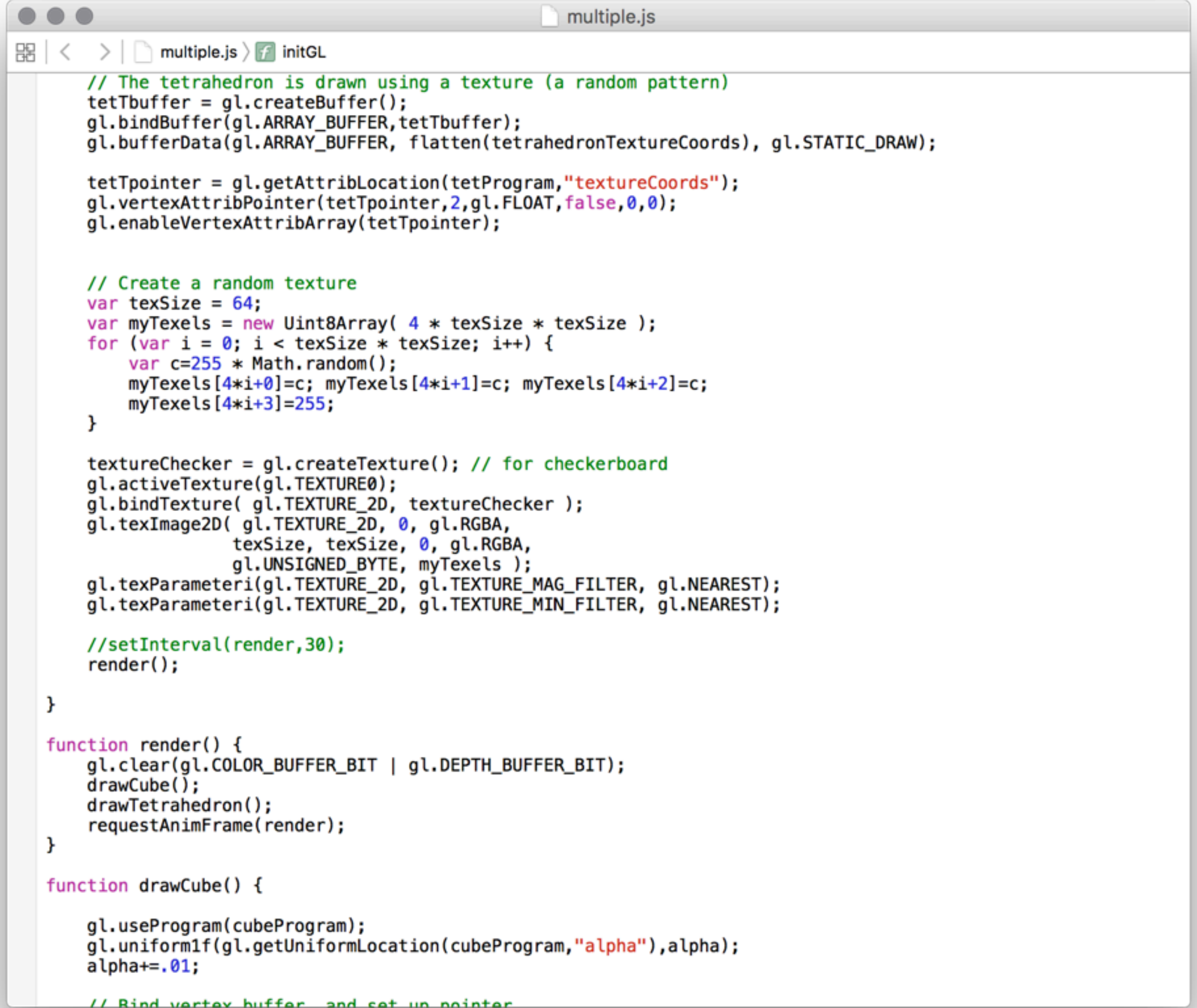
multiple.js 〉 initGL

```
    cubeColors = [vec4( 1.0, 1.0, .0, 1.0), // p0
                  vec4( 1.0, .0, 1.0, 1.0), // p1
                  vec4( 1.0, 1.0, 1.0, 1.0), // p2
                  vec4(  1.0, .0, .0, 1.0), // p3
                  vec4(  0.0, .0, .0, 1.0), // p4
                  vec4( 0.0, 1.0, .0, 1.0), // p5
                  vec4( 0.0, .0, 1.0, 1.0), // p6
                  vec4(  0.0, 1.0, 1.0, 1.0)];  // p7
    cubeIndexList = [0, 1, 3,
                     1, 2, 3,
                     6, 5, 7,
                     4, 7, 5,
                     0, 6, 1,
                     5, 6, 0,
                     2, 4, 3,
                     2, 7, 4,
                     0, 4, 5,
                     0, 3, 4,
                     2, 1, 6,
                     2, 6, 7];
    tetrahedronVertices = [vec4(0, 0, 0, 1),
                           vec4(0, 1, 0, 1),
                           vec4(1, 0, 0, 1),
                           vec4(0, 0, 0, 1),
                           vec4(0, 0, 1, 1),
                           vec4(0, 1, 0, 1),
                           vec4(0, 0, 0, 1),
                           vec4(1, 0, 0, 1),
                           vec4(0, 0, 1, 1),
                           vec4(1, 0, 0, 1),
                           vec4(0, 1, 0, 1),
                           vec4(0, 0, 1, 1)];
    tetrahedronTextureCoords = [vec2(0, 0),
                                vec2(0, 1),
                                vec2(1, 0),
                                vec2(0, 0),
                                vec2(0, 1),
                                vec2(1, 0),
                                vec2(0, 0),
                                vec2(0, 1),
                                vec2(1, 0),
                                vec2(0, 0),
                                vec2(0, 1),
                                vec2(1, 0)];
    tetrahedronIndexList = [0, 1, 2,
                            3, 4, 5,
```

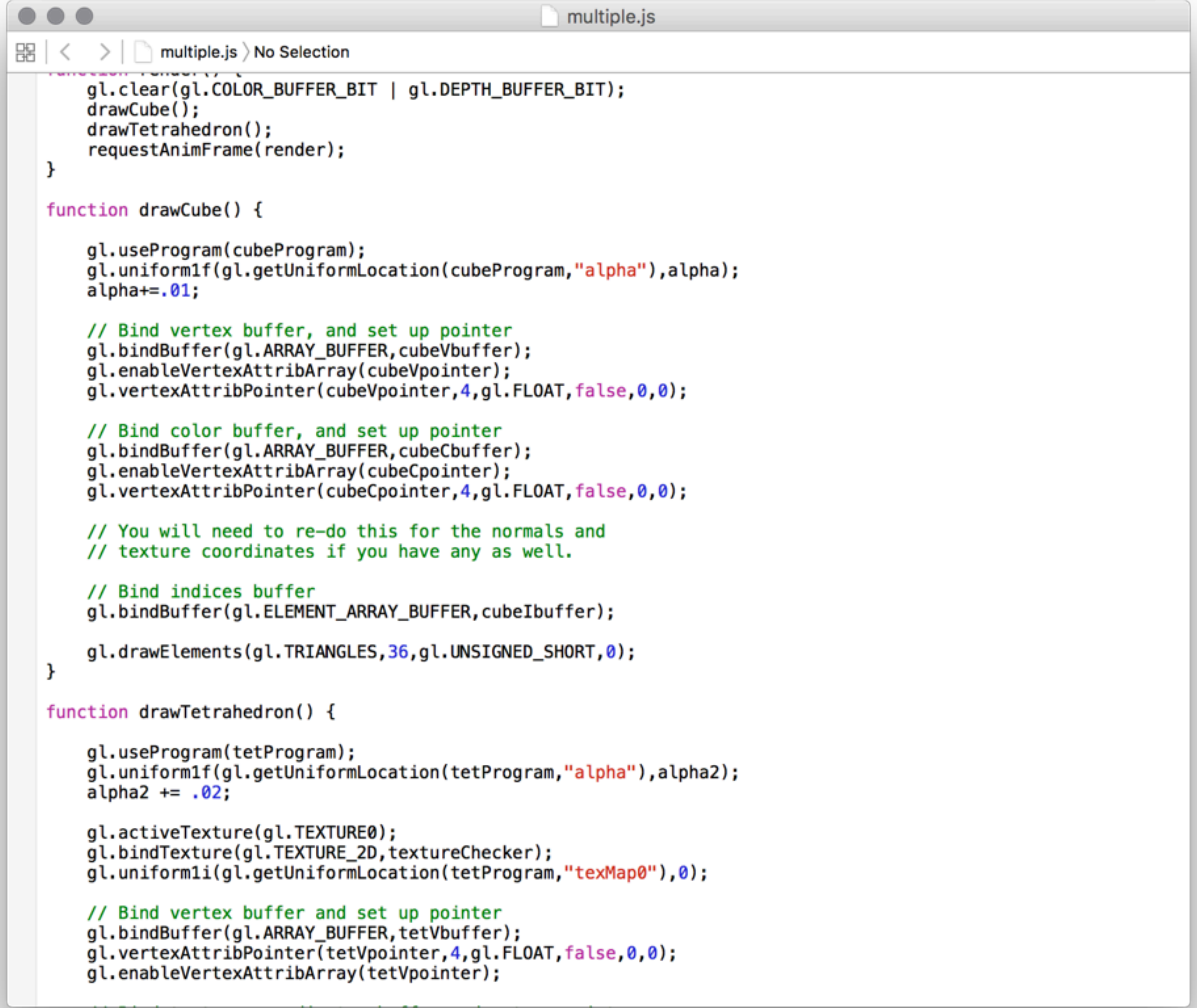

multiple.js

multiple.js 〉 initGL

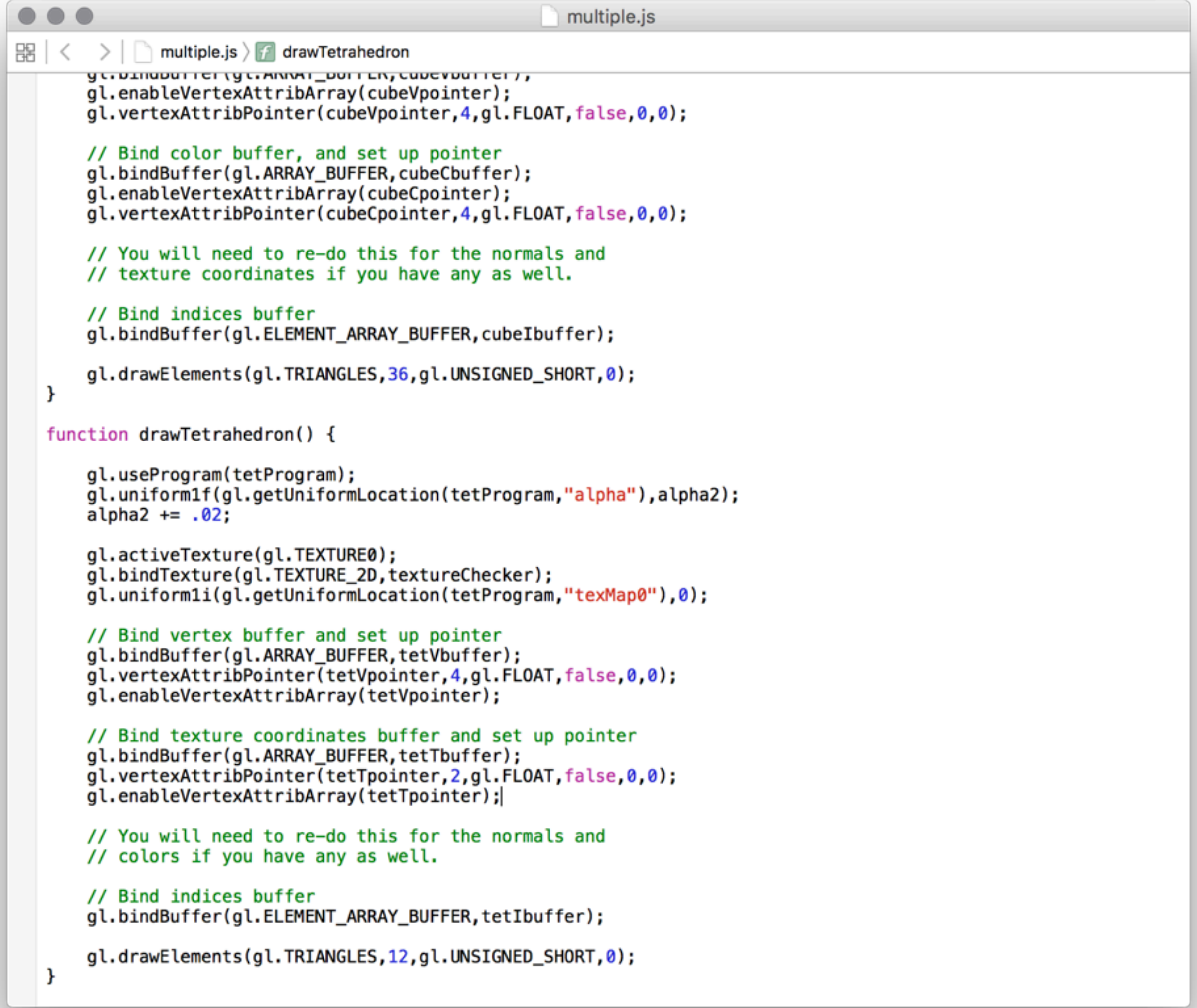
```
    tetrahedronIndexList = [0, 1, 2,
                            3, 4, 5,
                            6, 7, 8,
                            9, 10, 11];
    cubeProgram = initShaders(gl,"vertex-shader1","fragment-shader1");
    gl.useProgram(cubeProgram);
    cubeIbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,cubeIbuffer);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER,new Uint16Array(cubeIndexList), gl.STATIC_DRAW);

    cubeVbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,cubeVbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(cubeVertices), gl.STATIC_DRAW);

    cubeVpointer = gl.getAttribLocation(cubeProgram,"vertexPosition");
    gl.vertexAttribPointer(cubeVpointer,4,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(cubeVpointer);

    // The cube has colors at each vertex
    cubeCbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,cubeCbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(cubeColors), gl.STATIC_DRAW);

    cubeCpointer = gl.getAttribLocation(cubeProgram,"vertexColor");
    gl.vertexAttribPointer(cubeCpointer,4,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(cubeCpointer);


    tetProgram = initShaders(gl,"vertex-shader2","fragment-shader2");
    gl.useProgram(tetProgram);
    tetIbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,tetIbuffer);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER,new Uint16Array(tetrahedronIndexList), gl.STATIC_DRAW);

    tetVbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,tetVbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(tetrahedronVertices), gl.STATIC_DRAW);

    tetVpointer = gl.getAttribLocation(tetProgram,"vertexPosition");
    gl.vertexAttribPointer(tetVpointer,4,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(tetVpointer);

    // The tetrahedron is drawn using a texture (a random pattern)
    tetTbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,tetTbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(tetrahedronTextureCoords), gl.STATIC_DRAW);
```



```
    // The tetrahedron is drawn using a texture (a random pattern)
    tetTbuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER,tetTbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(tetrahedronTextureCoords), gl.STATIC_DRAW);

    tetTpointer = gl.getAttribLocation(tetProgram,"textureCoords");
    gl.vertexAttribPointer(tetTpointer,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(tetTpointer);


    // Create a random texture
    var texSize = 64;
    var myTexels = new Uint8Array( 4 * texSize * texSize );
    for (var i = 0; i < texSize * texSize; i++) {
        var c=255 * Math.random();
        myTexels[4*i+0]=c; myTexels[4*i+1]=c; myTexels[4*i+2]=c;
        myTexels[4*i+3]=255;
    }

    textureChecker = gl.createTexture(); // for checkerboard
    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture( gl.TEXTURE_2D, textureChecker );
    gl.texImage2D( gl.TEXTURE_2D, 0, gl.RGBA,
                   texSize, texSize, 0, gl.RGBA,
                   gl.UNSIGNED_BYTE, myTexels );
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);

    //setInterval(render,30);
    render();

}

function render() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    drawCube();
    drawTetrahedron();
    requestAnimFrame(render);
}

function drawCube() {

    gl.useProgram(cubeProgram);
    gl.uniform1f(gl.getUniformLocation(cubeProgram,"alpha"),alpha);
    alpha+=.01;

    // Bind vertex buffer, and set up pointer
```



```
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
    drawCube();
    drawTetrahedron();
    requestAnimFrame(render);
}

function drawCube() {

    gl.useProgram(cubeProgram);
    gl.uniform1f(gl.getUniformLocation(cubeProgram,"alpha"),alpha);
    alpha+=.01;

    // Bind vertex buffer, and set up pointer
    gl.bindBuffer(gl.ARRAY_BUFFER,cubeVbuffer);
    gl.enableVertexAttribArray(cubeVpointer);
    gl.vertexAttribPointer(cubeVpointer,4,gl.FLOAT,false,0,0);

    // Bind color buffer, and set up pointer
    gl.bindBuffer(gl.ARRAY_BUFFER,cubeCbuffer);
    gl.enableVertexAttribArray(cubeCpointer);
    gl.vertexAttribPointer(cubeCpointer,4,gl.FLOAT,false,0,0);

    // You will need to re-do this for the normals and
    // texture coordinates if you have any as well.

    // Bind indices buffer
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,cubeIbuffer);

    gl.drawElements(gl.TRIANGLES,36,gl.UNSIGNED_SHORT,0);
}

function drawTetrahedron() {

    gl.useProgram(tetProgram);
    gl.uniform1f(gl.getUniformLocation(tetProgram,"alpha"),alpha2);
    alpha2 += .02;

    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(gl.TEXTURE_2D,textureChecker);
    gl.uniform1i(gl.getUniformLocation(tetProgram,"texMap0"),0);

    // Bind vertex buffer and set up pointer
    gl.bindBuffer(gl.ARRAY_BUFFER,tetVbuffer);
    gl.vertexAttribPointer(tetVpointer,4,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(tetVpointer);
```



```
    gl.enableVertexAttribArray(cubeVpointer);
    gl.vertexAttribPointer(cubeVpointer,4,gl.FLOAT,false,0,0);

    // Bind color buffer, and set up pointer
    gl.bindBuffer(gl.ARRAY_BUFFER,cubeCbuffer);
    gl.enableVertexAttribArray(cubeCpointer);
    gl.vertexAttribPointer(cubeCpointer,4,gl.FLOAT,false,0,0);

    // You will need to re-do this for the normals and
    // texture coordinates if you have any as well.

    // Bind indices buffer
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,cubeIbuffer);

    gl.drawElements(gl.TRIANGLES,36,gl.UNSIGNED_SHORT,0);
}

function drawTetrahedron() {

    gl.useProgram(tetProgram);
    gl.uniform1f(gl.getUniformLocation(tetProgram,"alpha"),alpha2);
    alpha2 += .02;

    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(gl.TEXTURE_2D,textureChecker);
    gl.uniform1i(gl.getUniformLocation(tetProgram,"texMap0"),0);

    // Bind vertex buffer and set up pointer
    gl.bindBuffer(gl.ARRAY_BUFFER,tetVbuffer);
    gl.vertexAttribPointer(tetVpointer,4,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(tetVpointer);

    // Bind texture coordinates buffer and set up pointer
    gl.bindBuffer(gl.ARRAY_BUFFER,tetTbuffer);
    gl.vertexAttribPointer(tetTpointer,2,gl.FLOAT,false,0,0);
    gl.enableVertexAttribArray(tetTpointer);

    // You will need to re-do this for the normals and
    // colors if you have any as well.

    // Bind indices buffer
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,tetIbuffer);

    gl.drawElements(gl.TRIANGLES,12,gl.UNSIGNED_SHORT,0);
}
```