COMP 370 assignment #4: Affine Transformations in a WebGL Application

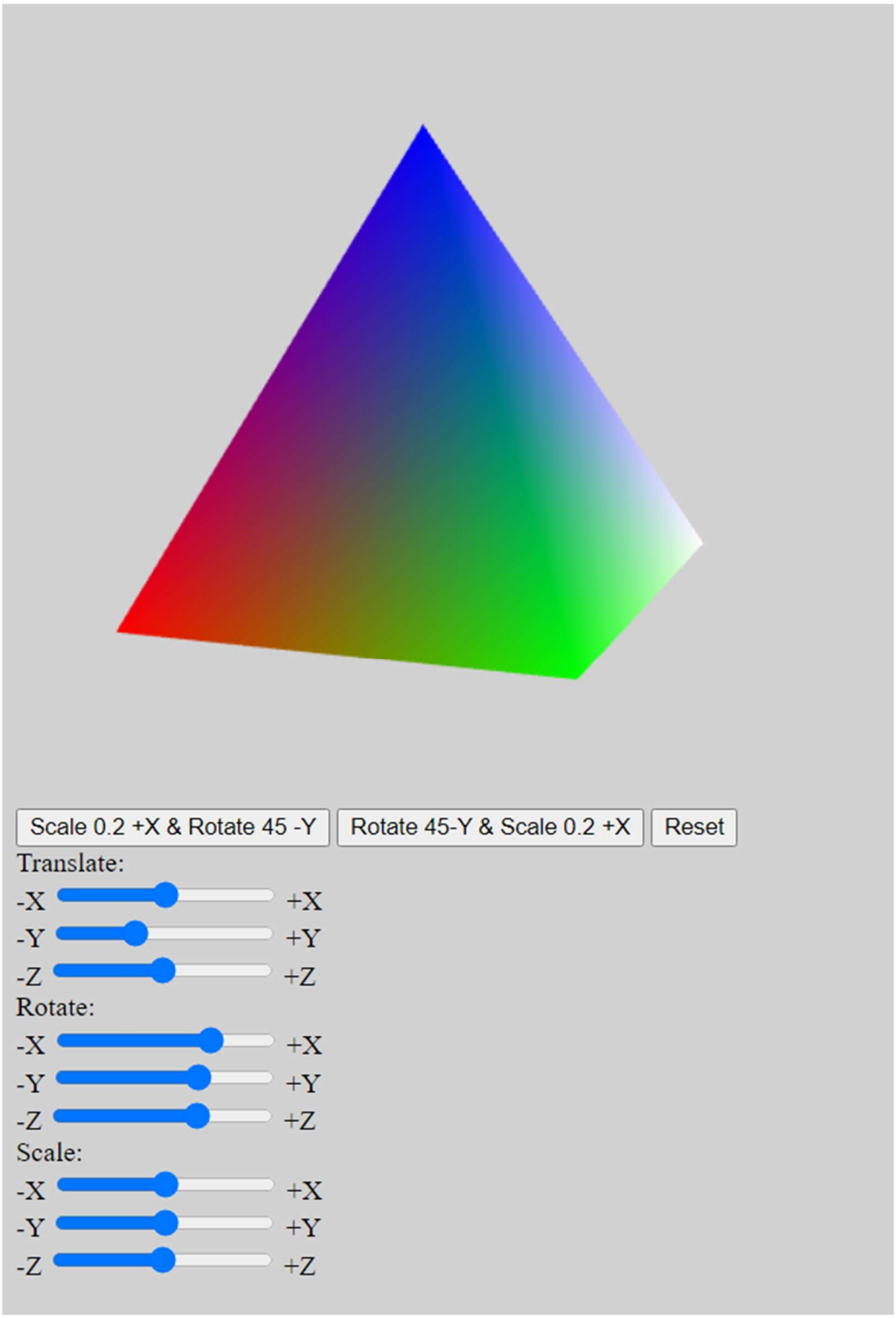
Thomas Williamson

id: 588206

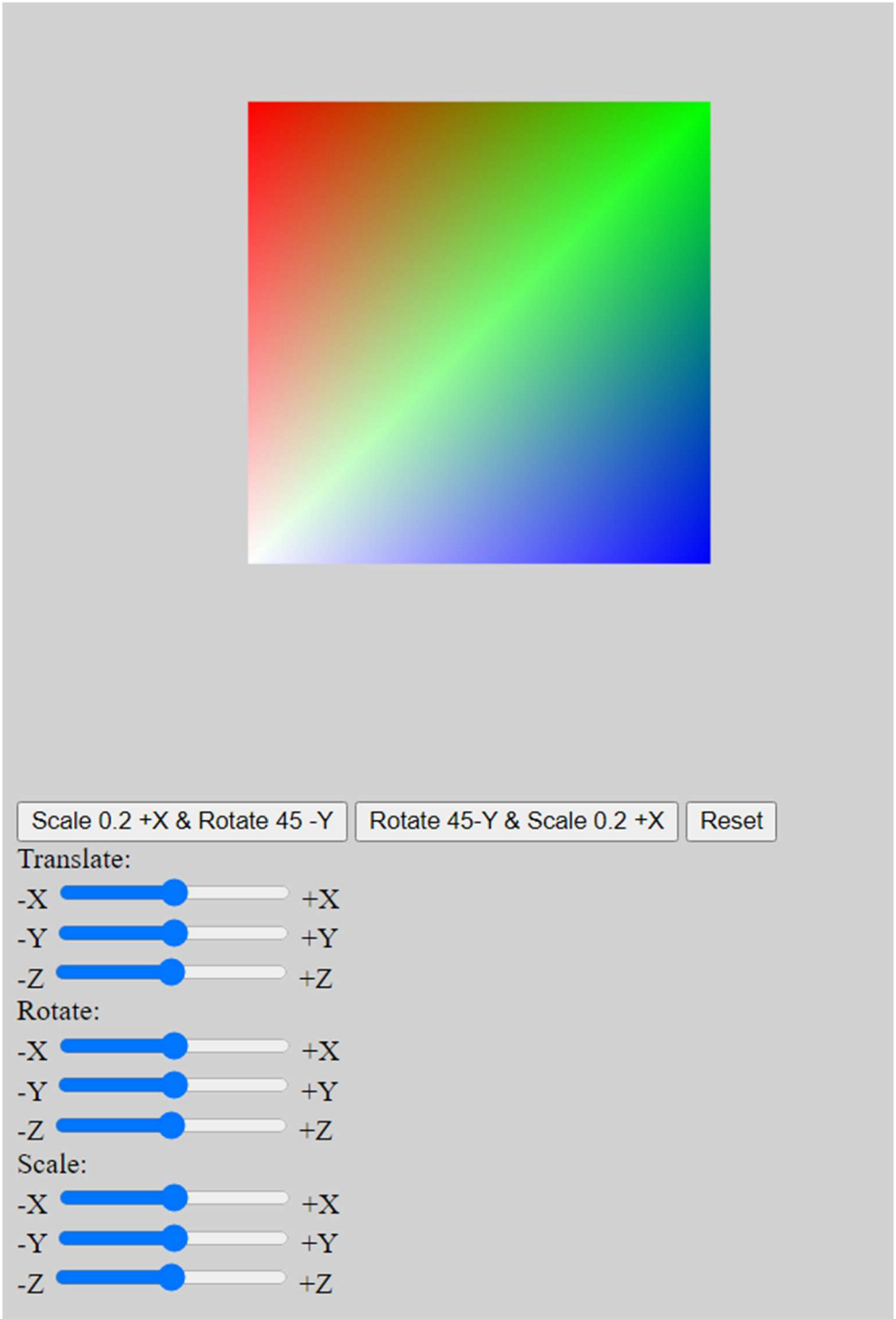
2021/11/16

The goal of this project is to render a tetrahedron and implement sliders to transform, rotate and scale the s3d shape on along the x, y, and z axis. Also implement 2 test case buttons and a reset button.

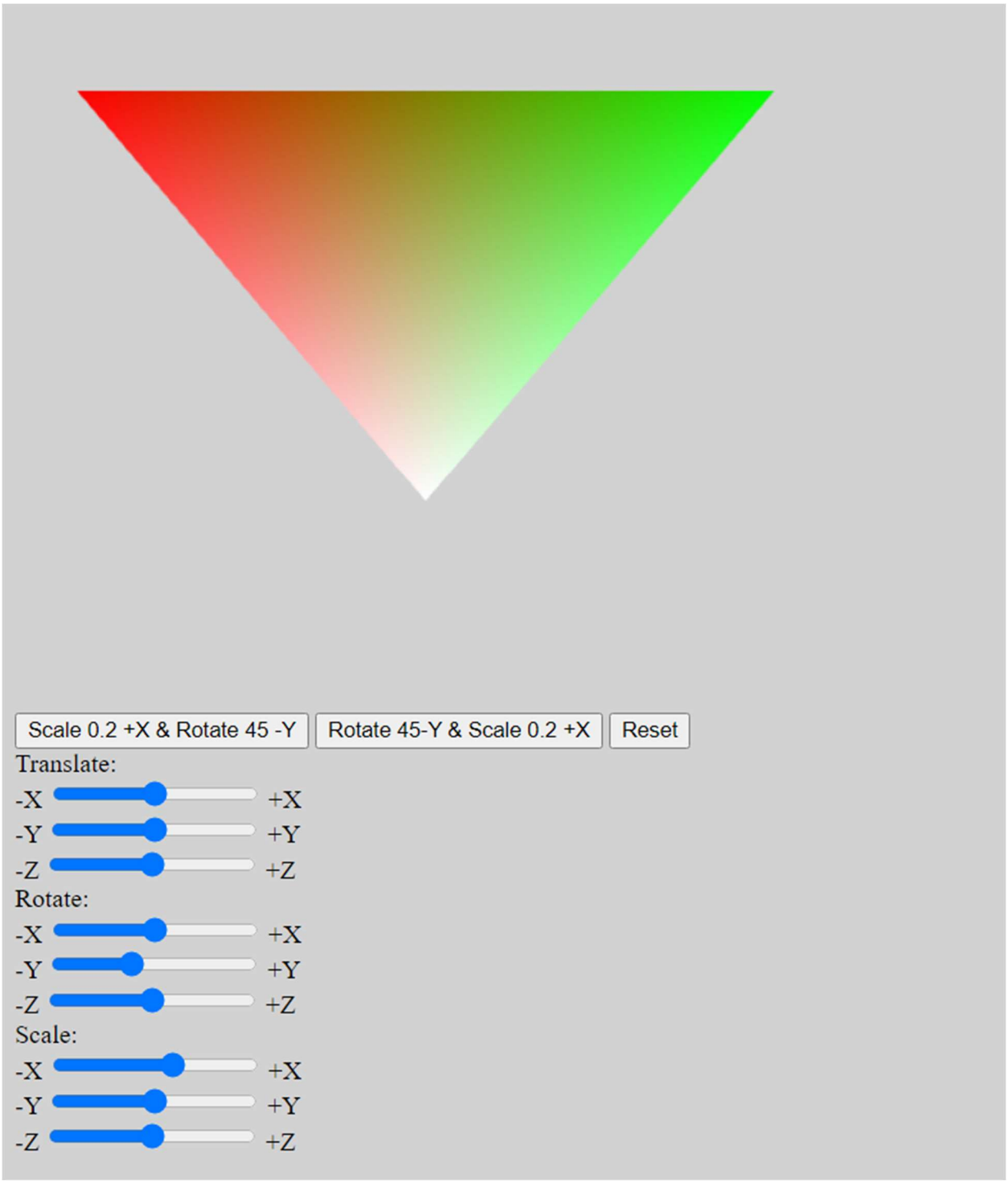
Startup



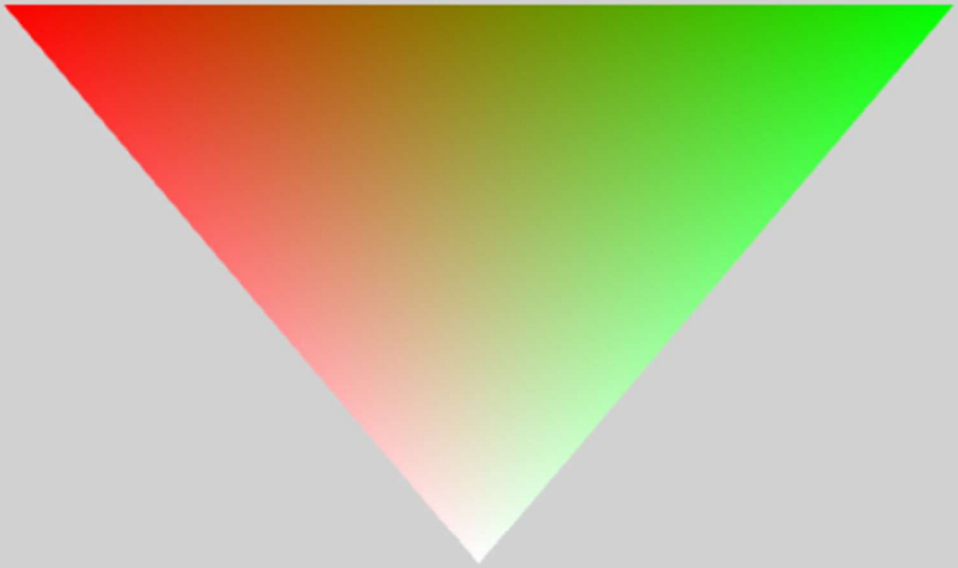
Reset



Test case 1


Scale 0.2 +X & Rotate 45 -Y
Rotate 45-Y & Scale 0.2 +X
Reset
Translate:
-X +X
-Y +Y
-Z +Z
Rotate:
-X +X
-Y +Y
-Z +Z
Scale:
-X +X
-Y +Y
-Z +Z

Test case 2



Scale 0.2 +X & Rotate 45 -Y | Rotate 45-Y & Scale 0.2 +X | Reset

Translate:

-X +X

-Y +Y

-Z +Z

Rotate:

-X +X

-Y +Y

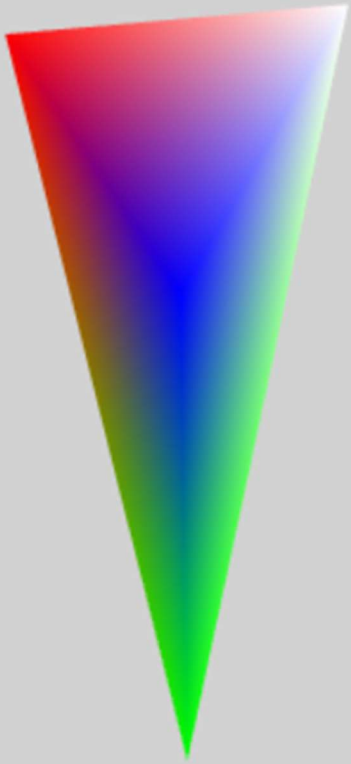
-Z +Z

Scale:

-X +X

-Y +Y

-Z +Z



Scale 0.2 +X & Rotate 45 -Y | Rotate 45-Y & Scale 0.2 +X | Reset

Translate:

-X +X

-Y +Y

-Z +Z

Rotate:

-X +X

-Y +Y

-Z +Z

Scale:

-X +X

-Y +Y

-Z +Z

Code

```
<!--
COMP 370 assignment #4: Affine Transformations in a WebGL Application
Thomas Williamson
id: 588206
2021/11/16
 -->
<!DOCTYPE html>
<html>

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

in  vec4 aPosition;
in  vec4 aColor;
out vec4 vColor;

uniform vec3 uTheta;
uniform vec3 transform;
uniform vec3 scaling;

void main()
{
    // Compute the sines and cosines of theta for each of
    //   the three axes in one computation.
    vec3 angles = radians( uTheta );
    vec3 c = cos( angles );
    vec3 s = sin( angles );

    // Remeber: thse matrices are column-major
    mat4 rx = mat4( 1.0,  0.0,  0.0, 0.0,
            0.0,  c.x,  s.x, 0.0,
            0.0, -s.x,  c.x, 0.0,
            0.0,  0.0,  0.0, 1.0 );

    mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
            0.0, 1.0,  0.0, 0.0,
            s.y, 0.0,  c.y, 0.0,
            0.0, 0.0,  0.0, 1.0 );


    mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
            -s.z,  c.z, 0.0, 0.0,
```

```
43              0.0,  0.0, 1.0, 0.0,
44              0.0,  0.0, 0.0, 1.0 );
45
46      mat4 t = mat4( 1, 0, 0, 0,
47              0, 1, 0, 0,
48              0, 0, 1, 0,
49              transform.x, transform.y, transform.z, 1);
50
51      mat4 scail = mat4( scaling.x, 0, 0, 0,
52              0, scaling.y, 0, 0,
53              0, 0, scaling.z, 0,
54              0, 0, 0, 1);
55
56
57
58
59      vColor = aColor;
60      gl_Position = t * scail *(rz * ry * rx) * aPosition;
61      gl_Position.z = -gl_Position.z;
62  }
63  </script>
64
65  <script id="fragment-shader" type="x-shader/x-fragment">
66  #version 300 es
67
68  precision mediump float;
69
70  in vec4 vColor;
71  out vec4 fColor;
72
73  void
74  main()
75  {
76      fColor = vColor;
77  }
78  </script>
79
80  <script type="text/javascript" src="../Common/initShaders.js"></script>
81  <script type="text/javascript" src="../Common/MVnew.js"></script>
82  <script type="text/javascript" src="assignment4.js"></script>
83
84  <body style="background-color:rgb(210,210,210);">
```

```
<canvas id="gl-canvas" width="512" height="512">
Oops ... your browser doesn't support the HTML5 canvas element
</canvas>

<br/>

<button id= "Test1Button">Scale 0.2 +X & Rotate 45 -Y</button>
<button id= "Test2Button">Rotate 45-Y & Scale 0.2 +X</button>
<button id= "Reset">Reset</button>
<div>
 Translate:<br>
 -X <input id="TranslateX" type="range"
min="-1" max="1" step=".10" value="0" />
+X<br>
 -Y <input id="TranslateY" type="range"
min="-1" max="1" step=".10" value="-0.3" />
+Y <br>
-Z <input id="TranslateZ" type="range"
min="-1" max="1" step=".10" value="0" />
+Z </div>

<div>
Rotate:<br>
 -X <input id="RotateX" type="range"
min="0" max="360" step="1" value="263" />
+X<br>
 -Y <input id="RotateY" type="range"
min="0" max="360" step="1" value="242" />
+Y <br>
-Z <input id="RotateZ" type="range"
min="0" max="360" step="1" value="244" />
+Z </div>

<div>
 Scale:<br>
 -X <input id="ScaleX" type="range"
min="0" max="2" step=".10" value="1" />
+X<br>
 -Y <input id="ScaleY" type="range"
min="0" max="2" step=".10" value="1" />
+Y <br>
-Z <input id="ScaleZ" type="range"
min="0" max="2" step=".10" value="1" />
+Z </div><br>


</body>
</html>

```

```
/*
COMP 370 assignment #4: Affine Transformations in a WebGL Application
Thomas Williamson
id: 588206
2021/11/16
*/
"use strict";

var canvas;
var gl;

var axis = 0;
var xAxis = 0;
var yAxis =1;
var zAxis = 2;
var theta = [0, 0, 0];
var thetaLoc;
var transf = [0, 0, 0];
var translationLoc;
var scaile = [1, 1, 1];
var scailLoc;
var numElements = 15;


    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 vertexColors = [
        vec4(1.0, 1.0, 1.0, 1.0),  // white
        vec4(1.0, 0.0, 0.0, 1.0),  // red
        vec4(0.0, 1.0, 0.0, 1.0),  // green
        vec4(0.0, 0.0, 1.0, 1.0),  // blue
        vec4(1.0, 1.0, 0.0, 1.0),  // yellow
```

```
        vec4(1.0, 0.0, 1.0, 1.0),  // magenta
        vec4(0.0, 0.0, 0.0, 1.0),  // black
        vec4(0.0, 1.0, 1.0, 1.0)   // cyan
    ];

// indices of the 12 triangles that compise the cube

var indices = [
    0, 1, 2, 255,
    0, 1, 3, 255,
    0, 2, 3, 255,
    3, 2, 1,
];

window.onload = function init()
{
    canvas = document.getElementById("gl-canvas");

    gl = canvas.getContext('webgl2');
    if (!gl) alert("WebGL 2.0 isn't available");


    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(210/255, 210/255, 210/255, 1.0);

    gl.enable(gl.DEPTH_TEST);
    //gl.enable(gl.PRIMITIVE_RESTART_FIXED_INDEX);

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

    // array element buffer

    var iBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, iBuffer);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint8Array(indices), gl.STATIC_DRAW

    // color array atrribute buffer
```

```
83
84      var cBuffer = gl.createBuffer();
85      gl.bindBuffer(gl.ARRAY_BUFFER, cBuffer);
86      gl.bufferData(gl.ARRAY_BUFFER, flatten(vertexColors), gl.STATIC_DRAW);
87
88      var colorLoc = gl.getAttribLocation(program, "aColor");
89      gl.vertexAttribPointer(colorLoc, 4, gl.FLOAT, false, 0, 0);
90      gl.enableVertexAttribArray(colorLoc);
91
92      // vertex array attribute buffer
93
94      var vBuffer = gl.createBuffer();
95      gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
96      gl.bufferData(gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW);
97
98      var positionLoc = gl.getAttribLocation( program, "aPosition");
99      gl.vertexAttribPointer(positionLoc, 3, gl.FLOAT, false, 0, 0);
100     gl.enableVertexAttribArray(positionLoc );
101
102     thetaLoc = gl.getUniformLocation(program, "uTheta");
103     translationLoc = gl.getUniformLocation(program, "transform");
104     scailLoc = gl.getUniformLocation(program, "scaling");
105
106    // console.log(gl.getUniformLocation(program, ""));
107
108     //event listeners for buttons
109     document.getElementById("Test1Button").onclick = function(){
110
111         document.getElementById("RotateY").value = parseFloat(document.getElementB
112         document.getElementById("ScaleX").value = parseFloat(document.getElementBy
113     };
114
115     document.getElementById("Test2Button").onclick = function(){
116         document.getElementById("ScaleX").value = parseFloat(document.getElementBy
117         document.getElementById("RotateY").value = parseFloat(document.getElementB
118     };
119
120     document.getElementById("Reset").onclick =  function(){
121         document.getElementById("ScaleY").value = 1;
122         document.getElementById("ScaleX").value = 1;
123         document.getElementById("ScaleZ").value = 1;
```

```
        document.getElementById("TranslateX").value = 0;
        document.getElementById("TranslateY").value = 0;
        document.getElementById("TranslateZ").value = 0;
        document.getElementById("RotateX").value = 180;
        document.getElementById("RotateY").value = 180;
        document.getElementById("RotateZ").value = 180;
    };

    render();
}

function render()
{
    gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
//    console.log(document.getElementById("RotateX").value);
    theta[xAxis] = document.getElementById("RotateX").value;
    theta[yAxis] = document.getElementById("RotateY").value;
    theta[zAxis] = document.getElementById("RotateZ").value;
    transf[0] = document.getElementById("TranslateX").value;
    transf[1] = document.getElementById("TranslateY").value;
    transf[2] = document.getElementById("TranslateZ").value;
    scaile[0] = document.getElementById("ScaleX").value;
    scaile[1] = document.getElementById("ScaleY").value;
    scaile[2] = document.getElementById("ScaleZ").value;


    gl.uniform3fv(thetaLoc, theta);
    gl.uniform3fv(translationLoc, transf);
    gl.uniform3fv(scailLoc, scaile);


    gl.drawElements(gl.TRIANGLE_FAN, numElements, gl.UNSIGNED_BYTE, 0);
    requestAnimationFrame(render);
}

```