Assignment #4 COMP 3820 – Computer Graphics and Visualization
Shane Steiner
T00622768
3/3/2021

Problem description:
Create a tetrahedron and apply affine transformations to it

```
<!-- File name: assignment4.html-->
<!-- Programmer name: Shane Steiner -->
<!-- Description: sets up shaders and buttons -->
<!-- Creation Date: 3/3/2021 -->

<!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 uTranslation;
uniform vec3 uScale;
uniform mat4 uAffine1;

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,
            uTranslation.x,  uTranslation.y,  uTranslation.z, 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,
            0.0,  0.0, 1.0, 0.0,
            0.0,  0.0, 0.0, 1.0);

    mat4 scale = mat4(
                    uScale.x, 0.0, 0.0, 0.0,
                    0.0, uScale.y, 0.0, 0.0,
                    0.0, 0.0, uScale.z, 0.0,
                    0.0, 0.0, 0.0, 1.0);

    vColor = aColor;
    gl_Position = rz * ry * rx * scale * uAffine1 * aPosition;
    gl_Position.z = -gl_Position.z;
}
</script>

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

precision mediump float;

in vec4 vColor;
out vec4 fColor;

void
main()
{
    fColor = vColor;
}
</script>

<script type="text/javascript"
src="/Assignment4/Common/initShaders.js"></script>
```

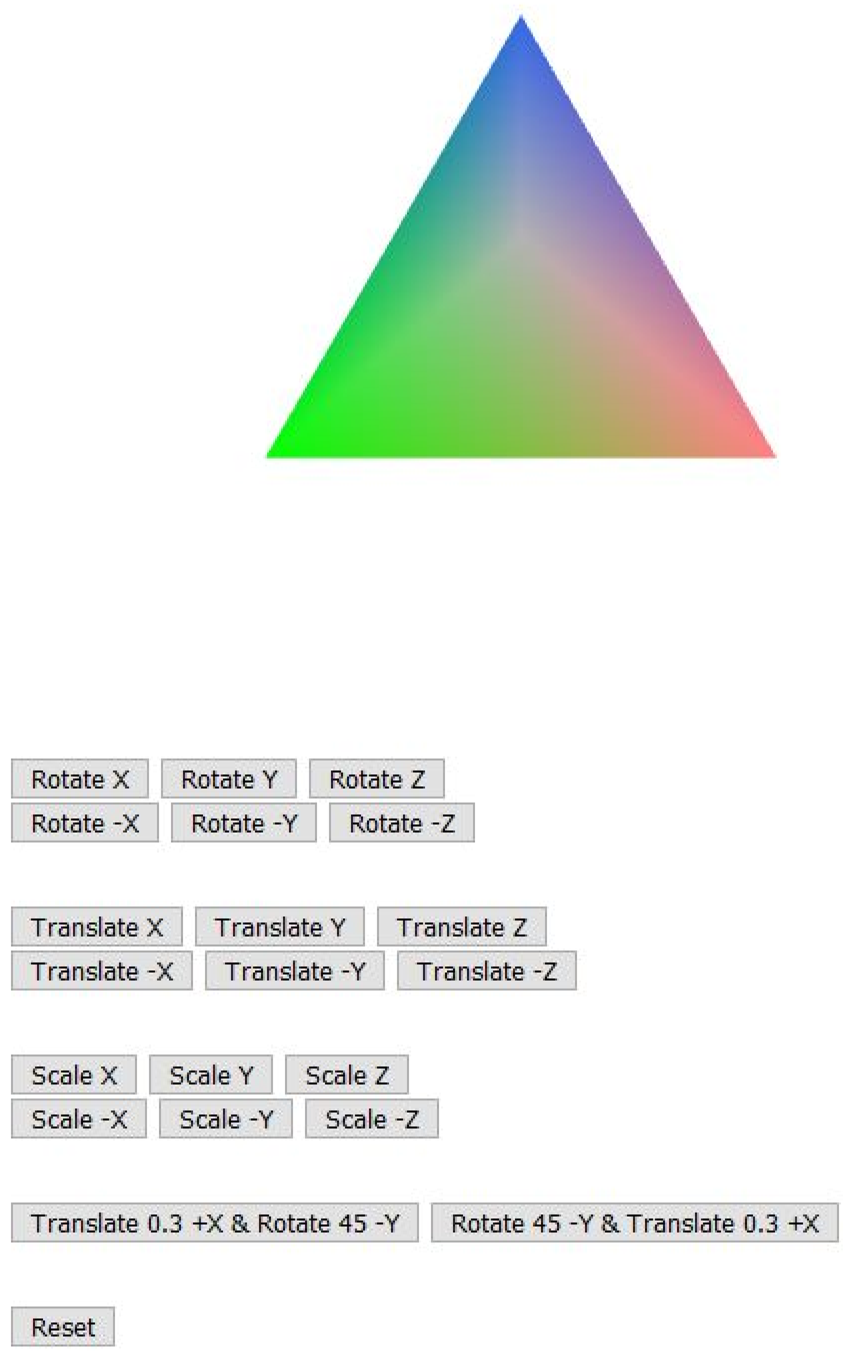
```html
<script type="text/javascript"
src="/Assignment4/Common/MVnew.js"></script>
<script type="text/javascript" src="assignment4.js"></script>
<script type="text/javascript" src="buttonHelper.js"></script>

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

<br/>

<button id= "xButtonRotate">Rotate X</button>
<button id= "yButtonRotate">Rotate Y</button>
<button id= "zButtonRotate">Rotate Z</button>
<div></div>
<button id= "-xButtonRotate">Rotate -X</button>
<button id= "-yButtonRotate">Rotate -Y</button>
<button id= "-zButtonRotate">Rotate -Z</button>

<div style="height: 30px;"></div>

<button id= "xButtonTranslate">Translate X</button>
<button id= "yButtonTranslate">Translate Y</button>
<button id= "zButtonTranslate">Translate Z</button>
<div></div>
<button id= "-xButtonTranslate">Translate -X</button>
<button id= "-yButtonTranslate">Translate -Y</button>
<button id= "-zButtonTranslate">Translate -Z</button>

<div style="height: 30px;"></div>

<button id= "xButtonScale">Scale X</button>
<button id= "yButtonScale">Scale Y</button>
<button id= "zButtonScale">Scale Z</button>
<div></div>
<button id= "-xButtonScale">Scale -X</button>
<button id= "-yButtonScale">Scale -Y</button>
<button id= "-zButtonScale">Scale -Z</button>
```

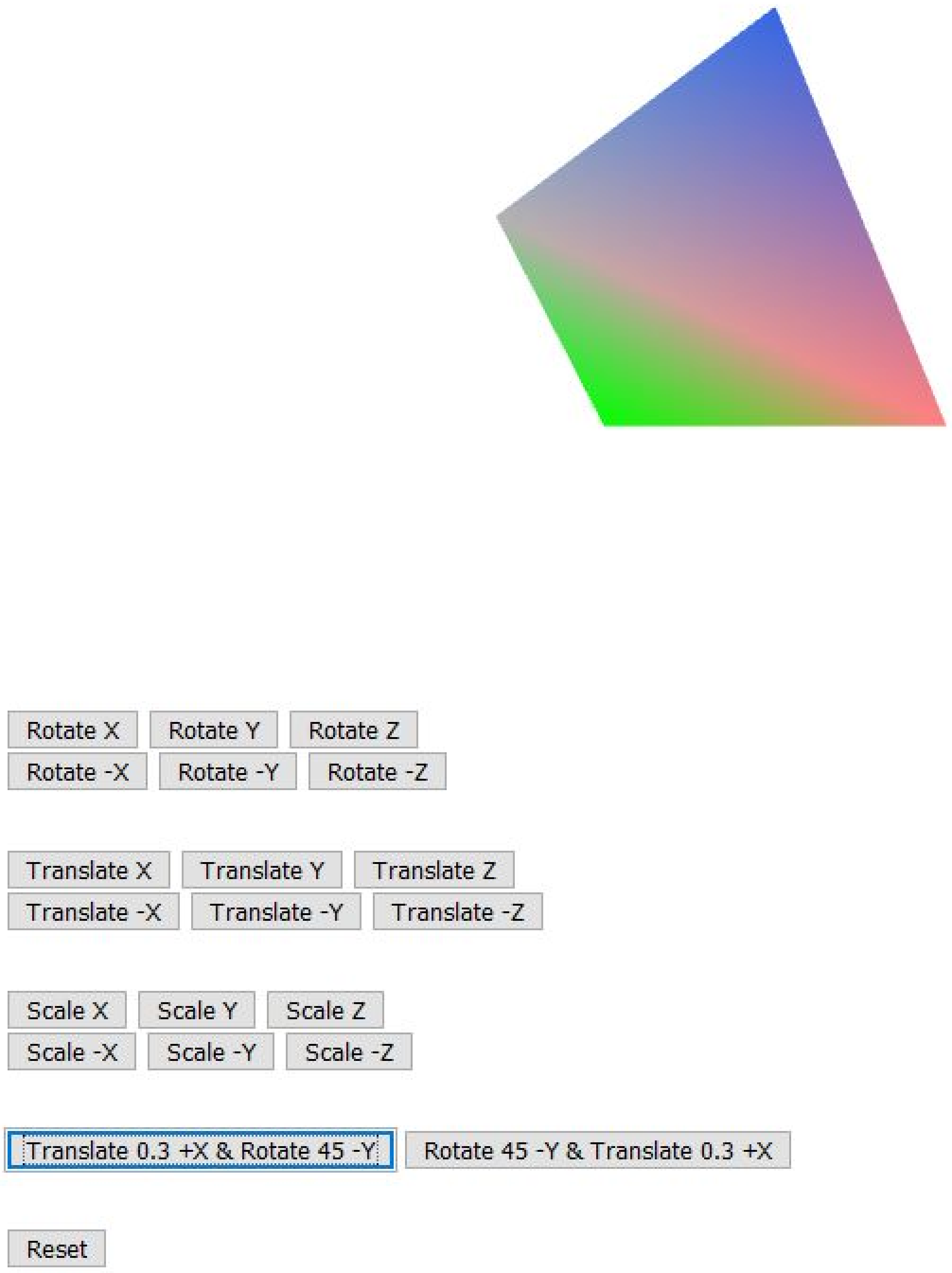
```
<div style="height: 30px;"></div>

<button id="affineButton1">Translate 0.3 +X & Rotate 45 -Y</button>
<button id="affineButton2"> Rotate 45 -Y & Translate 0.3 +X </button>

<div style="height: 30px;"></div>

<button id = "reset">Reset</button>

</body>
</html>

// <!-- File name: assignment4.js-->
// <!-- Programmer name: Shane Steiner -->
// <!-- Description: defines indecies and colors and renders shape -->
// <!-- Creation Date: 3/3/2021 -->

"use strict";

var canvas;
var gl;

var numPositions  = 12;

var xAxis = 0;
var yAxis = 1;
var zAxis = 2;

var axis = 0;
var theta = [0, 0, 0];
var translation = [0, 0, 0];
var scale = [1, 1, 1];
var mat1 = translate(0.3,0,0);
var mat2 = rotateY(-45);
var affine1 = flatten(mat4());

var thetaLoc;
var uTranslation;
var uScale;
var uAffine1;
var centeringconstY = 0.433;
var centeringconstZ = 0.408;

var vertices = [

    vec4(0.0, 0.433-centeringconstY,  0.816-centeringconstZ, 1.0),//0
    vec4(0.5,  0.0-centeringconstY,  0.0-centeringconstZ, 1.0),//1
    vec4(0.0, 0.866-centeringconstY,  0.0-centeringconstZ, 1.0),//2
    vec4(-0.5,  0.0-centeringconstY,  0.0-centeringconstZ, 1.0)//3

];

var vertexColors = [
    vec4(0.7, 0.7, 0.7, 1.0),  // black
    vec4(1.0, 0.5, 0.5, 1.0),  // red
    vec4(0.2, 0.4, 0.9, 1.0),  // yellow
    vec4(0.0, 1.0, 0.0, 1.0),  // green
    // vec4(Math.random(), Math.random(),Math.random(), 1.0),
    // vec4(Math.random(), Math.random(),Math.random(), 1.0),
    // vec4(Math.random(), Math.random(),Math.random(), 1.0),
    // vec4(Math.random(), Math.random(),Math.random(), 1.0),

    // vec4(0.0245019730168341, 0.9200946361333009, 0.06263875857974943,
1),
    // vec4(0.8683096346425287, 0.10982332488784863, 0.4627265576886377,
1),
    // vec4(0.5660531495244645, 0.5364220663020884, 0.6008126141768515, 1)
,
    // vec4(0.4042555652002686, 0.13026390965949775, 0.5537132974235977,
1),

];

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

```
//gets everything ready to render the shape
window.onload = function init()
{
    canvas = document.getElementById("gl-canvas");

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

    //colorCube();

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

    gl.enable(gl.DEPTH_TEST); //hidden serface removal?

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

    // array element buffer
    //i think this is where the order of verticies is defined for a face
    var iBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, iBuffer);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint8Array(indices),
gl.STATIC_DRAW);


    var cBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(vertexColors), gl.STATIC_DRAW);

    var colorLoc = gl.getAttribLocation( program, "aColor" );
    gl.vertexAttribPointer( colorLoc, 4, gl.FLOAT, false, 0, 0 );
    gl.enableVertexAttribArray( colorLoc );

    var vBuffer = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW);

    var positionLoc = gl.getAttribLocation(program, "aPosition");
    gl.vertexAttribPointer(positionLoc, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(positionLoc);

    thetaLoc = gl.getUniformLocation(program, "uTheta");
    uTranslation = gl.getUniformLocation(program, "uTranslation");
    uScale = gl.getUniformLocation(program, "uScale");
    uAffine1 = gl.getUniformLocation(program, "uAffine1");

    //event listeners for buttons
    initButtons();

    render();
}

//renders the shape
function render()
{
    gl.clear( gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    gl.uniform3fv(thetaLoc, theta);
    gl.uniform3fv(uTranslation, translation);
    gl.uniform3fv(uScale, scale);
    gl.uniformMatrix4fv(uAffine1,false,affine1 );

    gl.drawElements(gl.TRIANGLES, numPositions, gl.UNSIGNED_BYTE, 0);
    requestAnimationFrame(render);
}
```

```
// <!-- File name: buttonHelper.js-->
// <!-- Programmer name: Shane Steiner -->
// <!-- Description: listeners for buttons -->
// <!-- Creation Date: 3/3/2021 -->
```

```
//initlalises button listeners
function initButtons() {
    //rotate
    document.getElementById("xButtonRotate").onclick = function () {
        axis = xAxis;
        theta[axis] += 10.0;
    };
    document.getElementById("yButtonRotate").onclick = function () {
        axis = yAxis;
        theta[axis] += 10.0;
    };
    document.getElementById("zButtonRotate").onclick = function () {
        axis = zAxis;
        theta[axis] += 10.0;
    };
    document.getElementById("-xButtonRotate").onclick = function () {
        axis = xAxis;
        theta[axis] += -10.0;
    };
    document.getElementById("-yButtonRotate").onclick = function () {
        axis = yAxis;
        theta[axis] += -10.0;
    };
    document.getElementById("-zButtonRotate").onclick = function () {
        axis = zAxis;
        theta[axis] += -10.0;
    };


    //Translate
    document.getElementById("xButtonTranslate").onclick = function () {
        axis = xAxis;
        translation[axis] += 0.1;
    };
    document.getElementById("yButtonTranslate").onclick = function () {
        axis = yAxis;
        translation[axis] += 0.1;
    };
    document.getElementById("zButtonTranslate").onclick = function () {
        axis = zAxis;
        translation[axis] += 0.1;
    };
    document.getElementById("-xButtonTranslate").onclick = function () {
        axis = xAxis;
        translation[axis] += -0.1;
    };
    document.getElementById("-yButtonTranslate").onclick = function () {
        axis = yAxis;
        translation[axis] += -0.1;
    };
    document.getElementById("-zButtonTranslate").onclick = function () {
        axis = zAxis;
        translation[axis] += -0.1;
    };


    //Scale
    document.getElementById("xButtonScale").onclick = function () {
        axis = xAxis;
        scale[axis] += 0.1;
    };
    document.getElementById("yButtonScale").onclick = function () {
        axis = yAxis;
        scale[axis] += 0.1;
    };
    document.getElementById("zButtonScale").onclick = function () {
        axis = zAxis;
        scale[axis] += 0.1;
    };
    document.getElementById("-xButtonScale").onclick = function () {
        axis = xAxis;
        scale[axis] += -0.1;
    };
    document.getElementById("-yButtonScale").onclick = function () {
        axis = yAxis;
        scale[axis] += -0.1;
    };
    document.getElementById("-zButtonScale").onclick = function () {
        axis = zAxis;
        scale[axis] += -0.1;
    };

    // affine transformation order buttons
    document.getElementById("affineButton1").onclick = function () {
        affine1 = flatten(mult(mat1 ,mat2));

    };
    document.getElementById("affineButton2").onclick = function () {
         affine1 = flatten(mult(mat2 , mat1));
    };



    //reset
    document.getElementById("reset").onclick = function () {
         axis = 0;
         theta = [0, 0, 0];
         translation = [0, 0, 0];
         scale = [1, 1, 1];
         affine1 = flatten(mat4());
         affine2 = flatten(mat4());
    };
}
```


Rotate X
Rotate Y
Rotate Z
Rotate -X
Rotate -Y
Rotate -Z
Translate X
Translate Y
Translate Z
Translate -X
Translate -Y
Translate -Z
Scale X
Scale Y
Scale Z
Scale -X
Scale -Y
Scale -Z
Translate 0.3 +X & Rotate 45 -Y
Rotate 45 -Y & Translate 0.3 +X
Reset


Rotate X
Rotate Y
Rotate Z
Rotate -X
Rotate -Y
Rotate -Z
Translate X
Translate Y
Translate Z
Translate -X
Translate -Y
Translate -Z
Scale X
Scale Y
Scale Z
Scale -X
Scale -Y
Scale -Z
Translate 0.3 +X & Rotate 45 -Y
Rotate 45 -Y & Translate 0.3 +X
Reset