

Geometric Objects and Transformations

6TH WEEK, 2021



Modeling a Cube

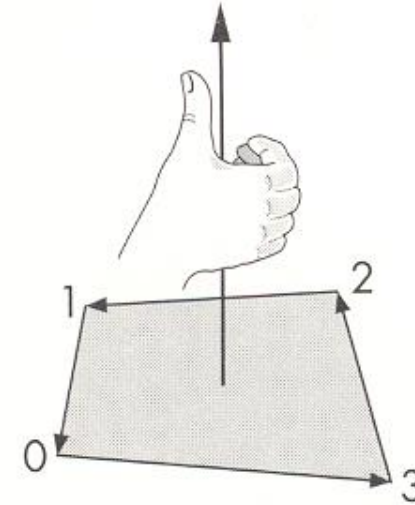
- Surface-based model
 - Outward-pointing face
 - Right-hand rule: counterclockwise order
- Data structure
 - Geometry: location of vertices

```
float vertices[8][3]= { {-1.0, -1.0, -1.0}, {1.0, -1.0, -1.0},  
                        {1.0, 1.0, -1.0}, {-1.0, 1.0, -1.0}, {-1.0, -1.0, 1.0},  
                        {1.0, -1.0, 1.0}, {1.0, 1.0, 1.0}, {-1.0, 1.0, 1.0}};
```

- Topology: connectivity

```
var indices = [ 0, 3, 2, 0, 2, 1 ];  
for ( var i = 0; i < indices.length; ++i )  
    points.push( vertices[indices[i]] );
```

```
gl.drawArrays( gl.TRIANGLES, 0, NumVertices );
```



<> colorCube.html X JS colorCube.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> colorCube.html > html > body > div > button#buttonT

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>학번 이름</title>
5     <script id="vertex-shader" type="x-shader/x-vertex">
6       attribute vec4 vPosition;
7       attribute vec4 vColor;
8       uniform vec3 theta;
9       varying vec4 fColor;
10
11     void main() {
12       // Compute the sines and cosines of theta for each of
13       // the three axes in one computation
14       vec3 angles = radians(theta);
15       vec3 c = cos(angles);
16       vec3 s = sin(angles);
17
18       // Remember: these matrices are column-major
19       mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
20                     0.0, c.x, s.x, 0.0,
21                     0.0, -s.x, c.x, 0.0,
22                     0.0, 0.0, 0.0, 1.0 );
23
24       mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
25                     0.0, 1.0, 0.0, 0.0,
26                     s.y, 0.0, c.y, 0.0,
27                     0.0, 0.0, 0.0, 1.0 );
28
29       mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
30                     -s.z, c.z, 0.0, 0.0,
31                     0.0, 0.0, 1.0, 0.0,
32                     0.0, 0.0, 0.0, 1.0 );
33
34       gl_Position = rz * ry * rx * vPosition;
35       fColor = vColor;
```



<> colorCube.html X JS colorCube.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> colorCube.html > html > body > div > button#buttonT

```
33
34     gl_Position = rz * ry * rx * vPosition;
35     fColor = vColor;
36 }
37 </script>
38
39 <script id="fragment-shader" type="x-shader/x-fragment">
40 precision mediump float;
41 varying vec4 fColor;
42
43 void main() {
44     gl_FragColor = fColor;
45 }
46 </script>
47
48 <script type="text/javascript" src="Common/webgl-utils.js"></script>
49 <script type="text/javascript" src="Common/initShaders.js"></script>
50 <script type="text/javascript" src="Common/MV.js"></script>
51 <script type="text/javascript" src="colorCube.js"></script>
52 </head>
53 <body>
54     <canvas id="gl-canvas" width="512" height="512">
55         Oops... your browser doesn't support the HTML5 canvas element!
56     </canvas>
57     <div>
58         <button id="xButton">Rotate X</button>
59         <button id="yButton">Rotate Y</button>
60         <button id="zButton">Rotate Z</button>
61         <button id="buttonT">Toggle Rotate</button>
62     </div>
63 </body>
64 </html>
```



<> colorCube.html JS colorCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS colorCube.js > quad

```
1  var gl;
2  var points = [];
3  var colors = [];
4
5  var axis = 0;
6  var theta = [0, 0, 0];
7  var thetaLoc;
8
9  var rotation = false;
10
11 window.onload = function init()
12 {
13     var canvas = document.getElementById("gl-canvas");
14
15     gl = WebGLUtils.setupWebGL(canvas);
16     if( !gl ) {
17         alert("WebGL isn't available!");
18     }
19
20     generateColorCube();
21
22     // Configure WebGL
23     gl.viewport(0, 0, canvas.width, canvas.height);
24     gl.clearColor(1.0, 1.0, 1.0, 1.0);
25
26     // Load shaders and initialize attribute buffers
27     var program = initShaders(gl, "vertex-shader", "fragment-shader");
28     gl.useProgram(program);
29
30     // Load the data into the GPU
31     var bufferId = gl.createBuffer();
32     gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
33     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
34
35     // Associate our shader variables with our data buffer
```



```

1  # Import the pandas module
2  import pandas as pd
3
4  # Create a dictionary of data
5  data = {
6      'Year': 2010,
7      'Country': 'USA',
8      'Population': 309,
9      'GDP': 14.7,
10     'GDP_per_capita': 47400,
11     'Life expectancy': 78.4,
12     'Fertility rate': 2.1,
13     'Infant mortality rate': 10.6,
14     'HDI': 0.854
15 }
16
17 # Create a pandas DataFrame
18 df = pd.DataFrame(data)
19
20 # Print the DataFrame
21 print(df)
22
23 # Access the 'Country' column
24 country = df['Country']
25
26 # Print the 'Country' column
27 print(country)
28
29 # Access the 'Population' column
30 population = df['Population']
31
32 # Print the 'Population' column
33 print(population)
34
35 # Access the 'GDP' column
36 gdp = df['GDP']
37
38 # Print the 'GDP' column
39 print(gdp)
40
41 # Access the 'GDP_per_capita' column
42 gdp_per_capita = df['GDP_per_capita']
43
44 # Print the 'GDP_per_capita' column
45 print(gdp_per_capita)
46
47 # Access the 'Life expectancy' column
48 life_expectancy = df['Life expectancy']
49
50 # Print the 'Life expectancy' column
51 print(life_expectancy)
52
53 # Access the 'Fertility rate' column
54 fertility_rate = df['Fertility rate']
55
56 # Print the 'Fertility rate' column
57 print(fertility_rate)
58
59 # Access the 'Infant mortality rate' column
60 infant_mortality_rate = df['Infant mortality rate']
61
62 # Print the 'Infant mortality rate' column
63 print(infant_mortality_rate)
64
65 # Access the 'HDI' column
66 hdi = df['HDI']
67
68 # Print the 'HDI' column
69 print(hdi)

```

<> colorCube.html JS colorCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS colorCube.js > quad

```
69
70 function render() {
71     gl.clear(gl.COLOR_BUFFER_BIT);
72
73     if( rotation ) {
74         theta[axis] += 2.0;
75     }
76     gl.uniform3fv(thetaLoc, theta)
77
78     gl.drawArrays(gl.TRIANGLES, 0, points.length);
79
80     window.requestAnimationFrame(render);
81 }
82
83 function generateColorCube() {
84     quad(1, 0, 3, 2);
85     quad(2, 3, 7, 6);
86     quad(3, 0, 4, 7);
87     quad(4, 5, 6, 7);
88     quad(5, 4, 0, 1);
89     quad(6, 5, 1, 2);
90 }
91
92 function quad(a, b, c, d) {
93     vertexPos = [
94         vec4(-0.5, -0.5, -0.5, 1.0),
95         vec4( 0.5, -0.5, -0.5, 1.0),
96         vec4( 0.5,  0.5, -0.5, 1.0),
97         vec4(-0.5,  0.5, -0.5, 1.0),
98         vec4(-0.5, -0.5,  0.5, 1.0),
99         vec4( 0.5, -0.5,  0.5, 1.0),
100        vec4( 0.5,  0.5,  0.5, 1.0),
101        vec4(-0.5,  0.5,  0.5, 1.0)
102    ];
103
```

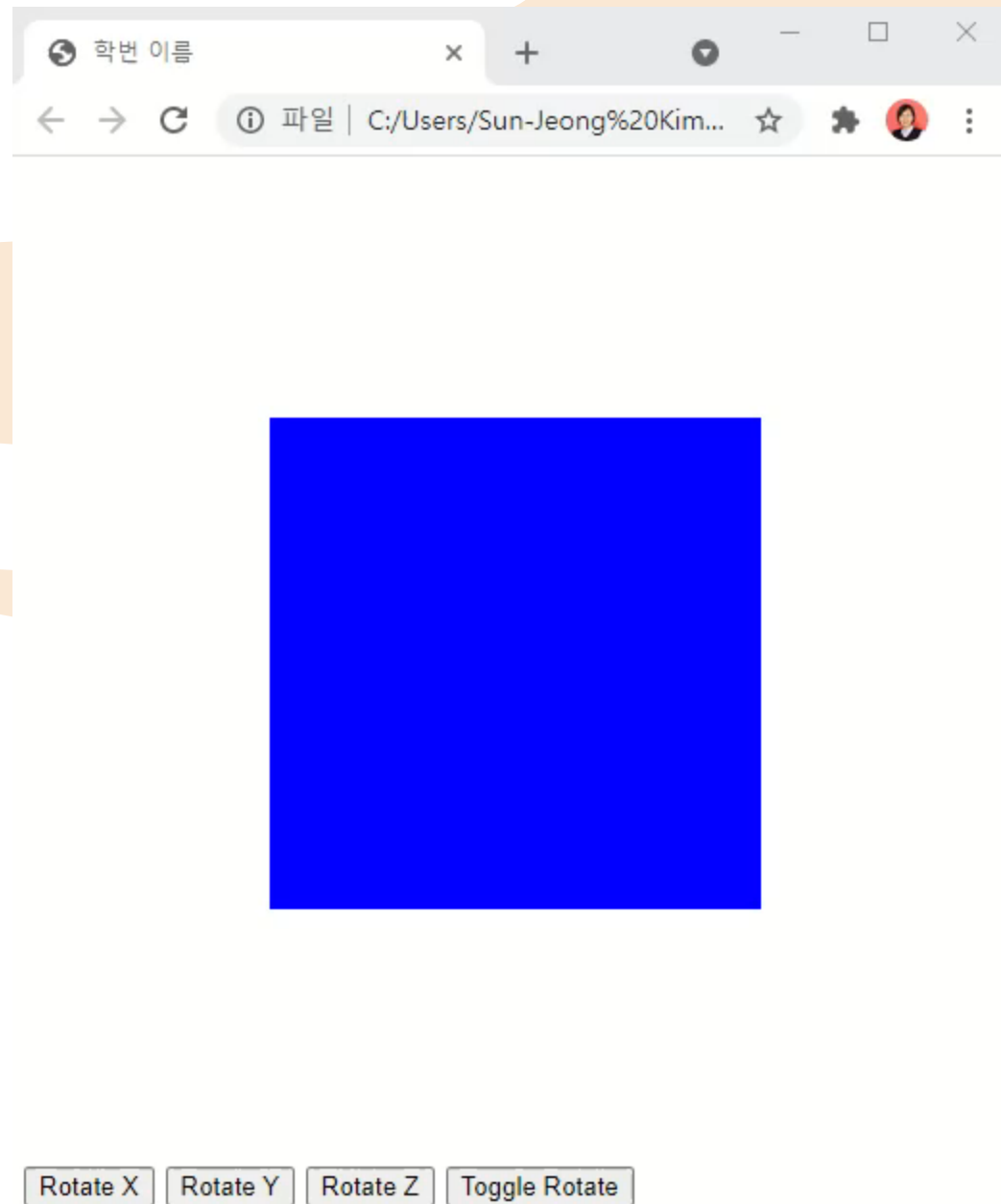


<> colorCube.html JS colorCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS colorCube.js > quad

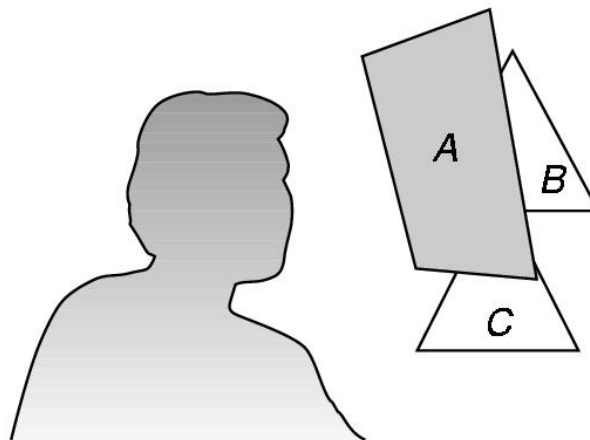
```
93     vertexPos = [  
94         vec4(-0.5, -0.5, -0.5, 1.0),  
95         vec4( 0.5, -0.5, -0.5, 1.0),  
96         vec4( 0.5,  0.5, -0.5, 1.0),  
97         vec4(-0.5,  0.5, -0.5, 1.0),  
98         vec4(-0.5, -0.5,  0.5, 1.0),  
99         vec4( 0.5, -0.5,  0.5, 1.0),  
100        vec4( 0.5,  0.5,  0.5, 1.0),  
101        vec4(-0.5,  0.5,  0.5, 1.0)  
102    ];  
103  
104    vertexColor = [  
105        vec4(0.0, 0.0, 0.0, 1.0),    // black  
106        vec4(1.0, 0.0, 0.0, 1.0),    // red  
107        vec4(1.0, 1.0, 0.0, 1.0),    // yellow  
108        vec4(0.0, 1.0, 0.0, 1.0),    // green  
109        vec4(0.0, 0.0, 1.0, 1.0),    // blue  
110        vec4(1.0, 0.0, 1.0, 1.0),    // magenta  
111        vec4(1.0, 1.0, 1.0, 1.0),    // white  
112        vec4(0.0, 1.0, 1.0, 1.0)     // cyan  
113    ];  
114  
115    // We need to partition the quad into two triangles in order for WebGL  
116    // to be able to render it. In this case, we create two triangles from  
117    // the quad indices.  
118    var index = [ a, b, c, a, c, d ];  
119    for(var i=0; i<index.length; i++) {  
120        points.push(vertexPos[index[i]]);  
121  
122        // for solid colored faces use  
123        colors.push(vertexColor[a]);  
124    }  
125 }  
126
```





Hidden-Surface Removal

- To see only those surfaces in front of other surfaces
- Visible-surface algorithms or hidden-surface-removal algorithm
 - Algorithms for ordering objects
 - OpenGL uses the z-buffer algorithm that saves depth information



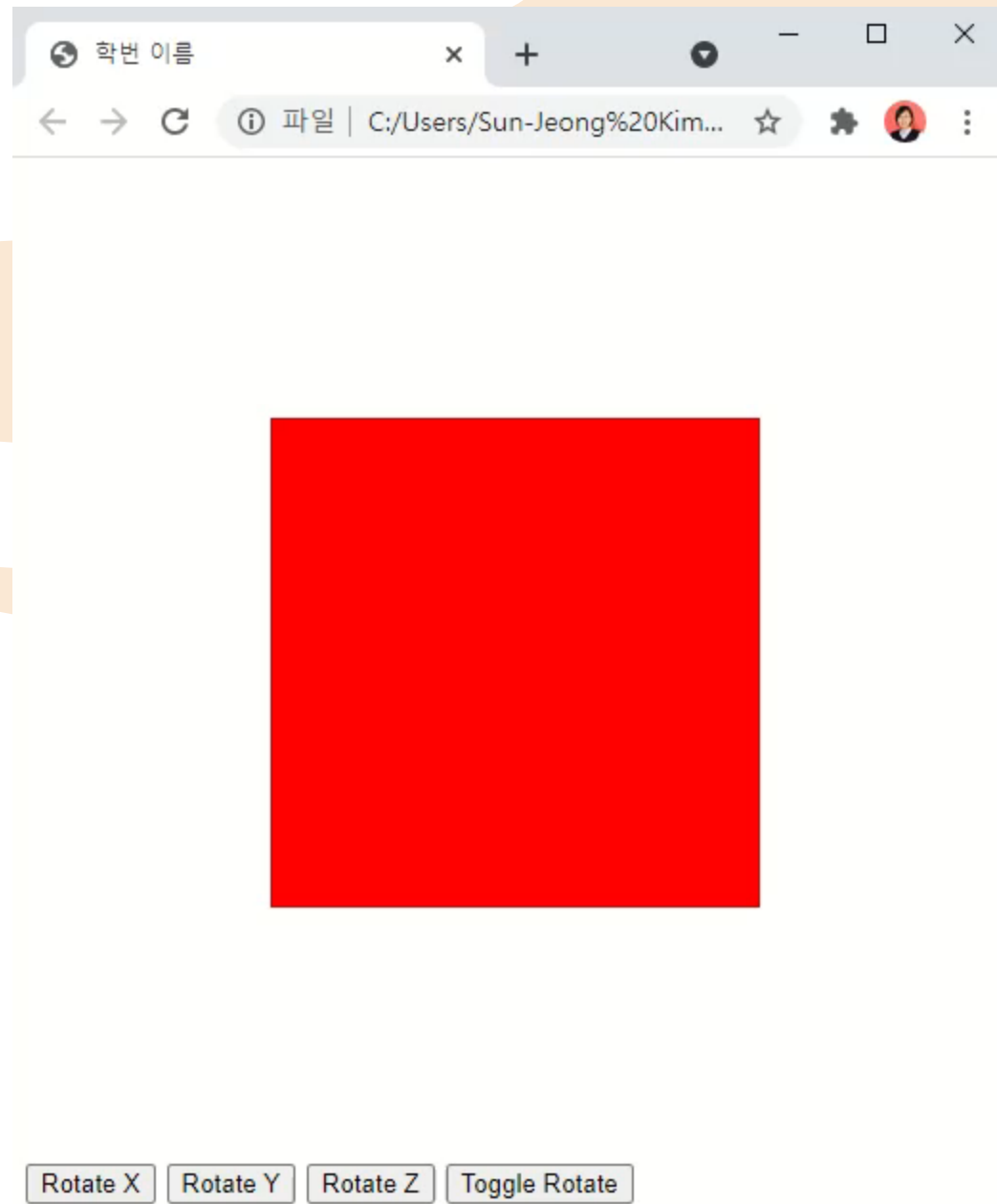
Hidden-Surface Problem

Using the Z-buffer Algorithm

- The algorithm uses an extra buffer, the z-buffer, to store depth information as geometry travels down the pipeline
- Depth buffer is required to available in WebGL
- It must be
 - Enabled
 - `gl.enable(gl.DEPTH_TEST);`
 - Cleared in for each render
 - `gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);`

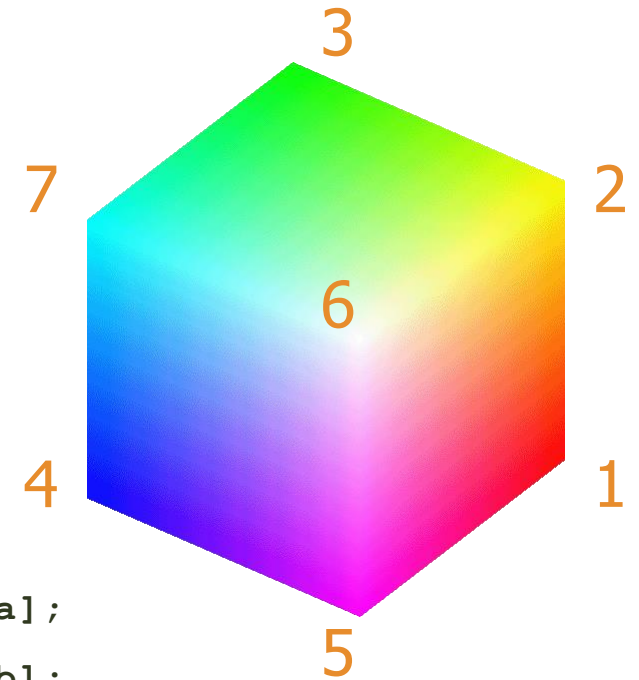
```
21
22 // Configure WebGL
23 gl.viewport(0, 0, canvas.width, canvas.height);
24 gl.clearColor(1.0, 1.0, 1.0, 1.0);
25
26 // Enable hidden-surface removal
27 gl.enable(gl.DEPTH_TEST);
28
29 // Load shaders and initialize attribute buffers
30 var program = initShaders(gl, "vertex-shader", "fragment-shader");
31 gl.useProgram(program);
32
```

```
73 function render() {
74   gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
75
76   if( rotation ) {
77     theta[axis] += 2.0;
78   }
79   gl.uniform3fv(thetaLoc, theta)
80
81   gl.drawArrays(gl.TRIANGLES, 0, points.length);
82
83   window.requestAnimationFrame(render);
84 }
```



Color Cube (1)

```
float vertex_pos[8][3]= { {-1.0, -1.0, -1.0}, {1.0, -1.0, -1.0},  
                          {1.0, 1.0, -1.0}, {-1.0, 1.0, -1.0}, {-1.0, -1.0, 1.0},  
                          {1.0, -1.0, 1.0}, {1.0, 1.0, 1.0}, {-1.0, 1.0, 1.0}};  
  
float vertex_color[8][3] = { {0.0, 0.0, 0.0}, {1.0, 0.0, 0.0},  
                             {1.0, 1.0, 0.0}, {0.0, 1.0, 0.0}, {0.0, 0.0, 1.0},  
                             {1.0, 0.0, 1.0}, {1.0, 1.0, 1.0}, {0.0, 1.0, 1.0}};  
  
int Index = 0;  
void quad( int a, int b, int c, int d )  
{  
    colors[Index] = vertex_color[a];    points[Index] = vertex_pos[a];  
    Index++;  
    colors[Index] = vertex_color[b];    points[Index] = vertex_pos[b];  
    Index++;  
    colors[Index] = vertex_color[c];    points[Index] = vertex_pos[c];  
    Index++;  
  
    colors[Index] = vertex_color[a];    points[Index] = vertex_pos[a];  
    Index++;  
    colors[Index] = vertex_color[c];    points[Index] = vertex_pos[c];  
    Index++;  
    colors[Index] = vertex_color[b];    points[Index] = vertex_pos[b];  
    Index++;  
}
```



Color Cube (2)

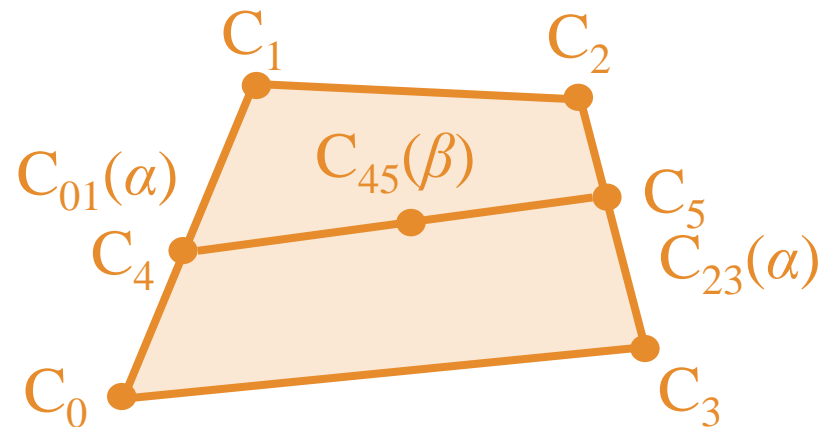
```
void generateColorCube()  
{  
    quad( 1, 0, 3, 2 );  
    quad( 2, 3, 7, 6 );  
    quad( 3, 0, 4, 7 );  
    quad( 4, 5, 6, 7 );  
    quad( 5, 4, 0, 1 );  
    quad( 6, 5, 1, 2 );  
}
```

- Bilinear interpolation

$$C_{01}(\alpha) = (1 - \alpha)C_0 + \alpha C_1$$

$$C_{23}(\alpha) = (1 - \alpha)C_2 + \alpha C_3$$

$$C_{45}(\beta) = (1 - \beta)C_4 + \beta C_5$$



colorCube.html JS colorCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS colorCube.js > quad

```
106
107     vertexColor = [
108         vec4(0.0, 0.0, 0.0, 1.0),    // black
109         vec4(1.0, 0.0, 0.0, 1.0),    // red
110         vec4(1.0, 1.0, 0.0, 1.0),    // yellow
111         vec4(0.0, 1.0, 0.0, 1.0),    // green
112         vec4(0.0, 0.0, 1.0, 1.0),    // blue
113         vec4(1.0, 0.0, 1.0, 1.0),    // magenta
114         vec4(1.0, 1.0, 1.0, 1.0),    // white
115         vec4(0.0, 1.0, 1.0, 1.0)     // cyan
116     ];
117
118     // We need to partition the quad into two triangles in order for WebGL
119     // to be able to render it. In this case, we create two triangles from
120     // the quad indices.
121     var index = [ a, b, c, a, c, d ];
122     for(var i=0; i<index.length; i++) {
123         points.push(vertexPos[index[i]]);
124         colors.push(vertexColor[index[i]]);
125
126         // for solid colored faces use
127         // colors.push(vertexColor[a]);
128     }
129 }
130
```

```
function init() {
    // Create WebGL context
    const canvas = document.getElementById('canvas');
    const gl = canvas.getContext('webgl');

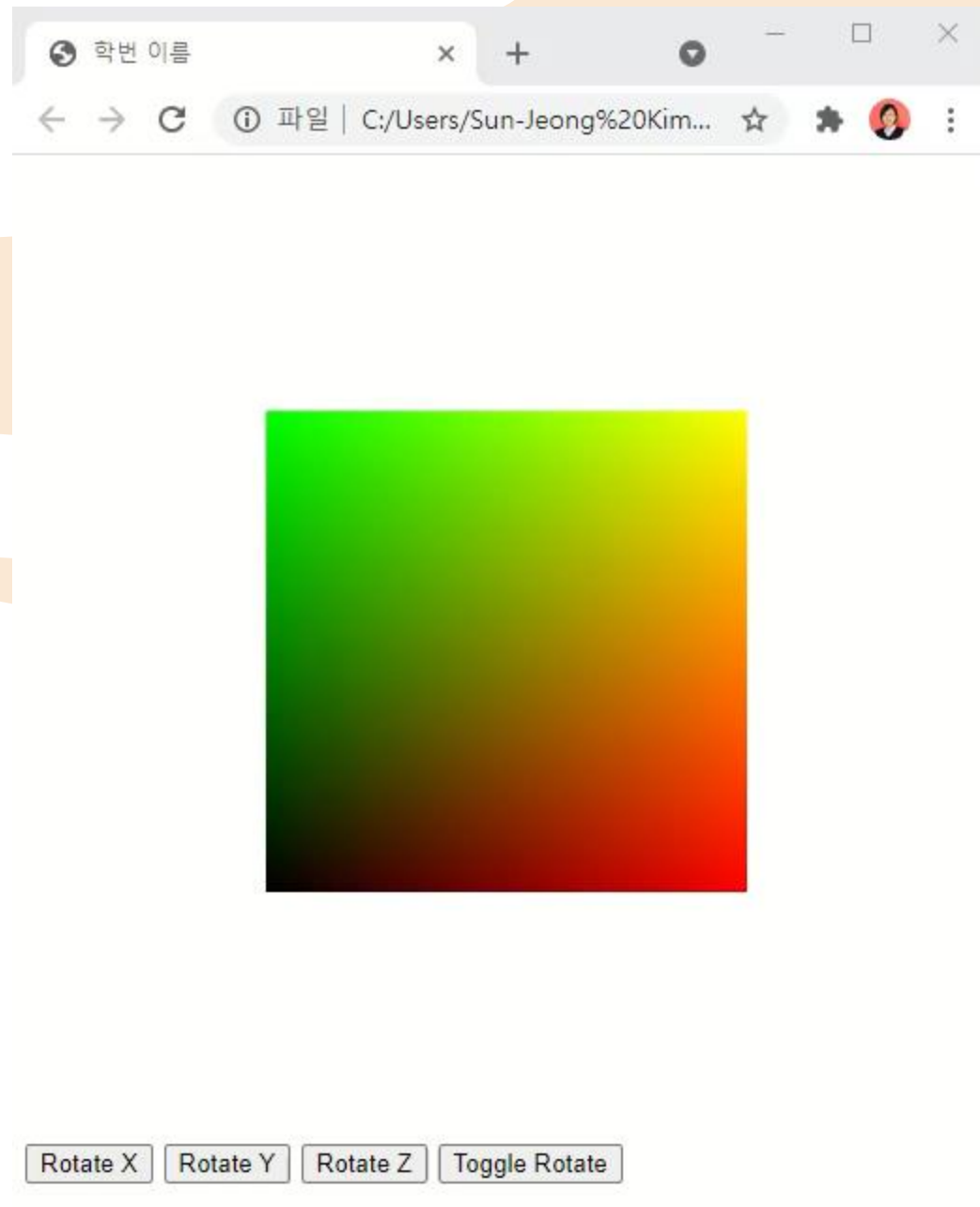
    // Load shaders
    const vertexShaderSource = `
    attribute vec3 position;
    attribute vec4 color;
    varying vec4 vColor;
    void main() {
        vColor = color;
        gl_Position = vec4(position, 1.0);
    }`;
    const fragmentShaderSource = `
    precision mediump float;
    varying vec4 vColor;
    void main() {
        gl_FragColor = vColor;
    }`;

    // Compile shaders
    const vertexShader = compileShader(gl, gl.VERTEX_SHADER, vertexShaderSource);
    const fragmentShader = compileShader(gl, gl.FRAGMENT_SHADER, fragmentShaderSource);

    // Link program
    const program = gl.createProgram();
    gl.attachShader(program, vertexShader);
    gl.attachShader(program, fragmentShader);
    gl.linkProgram(program);

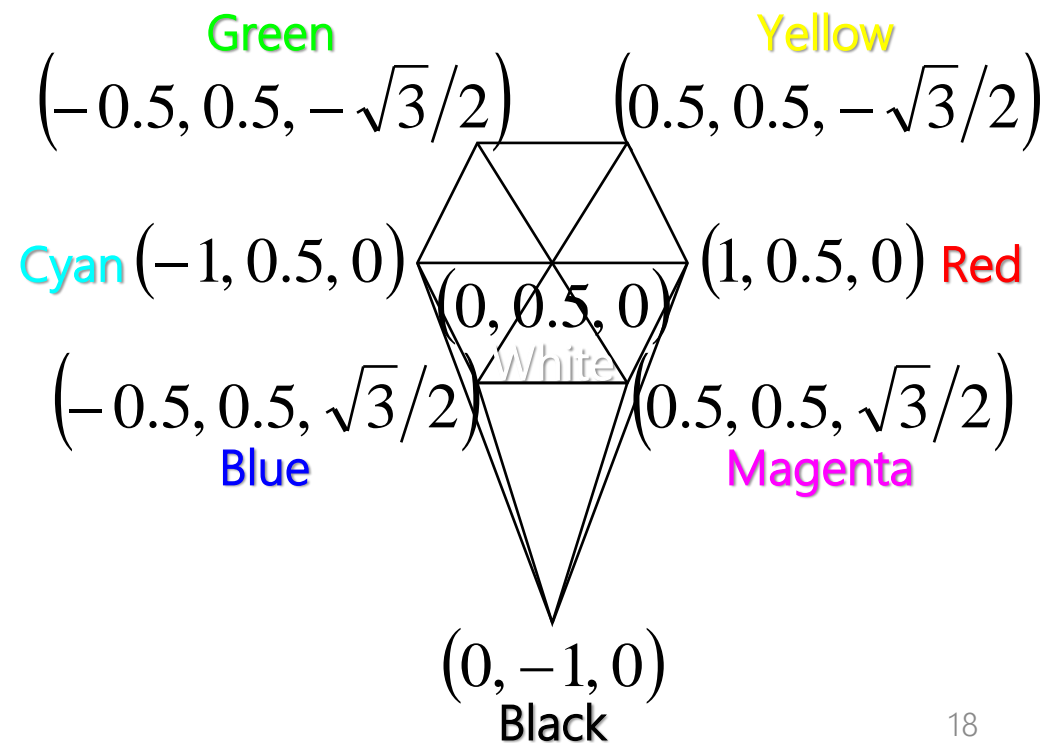
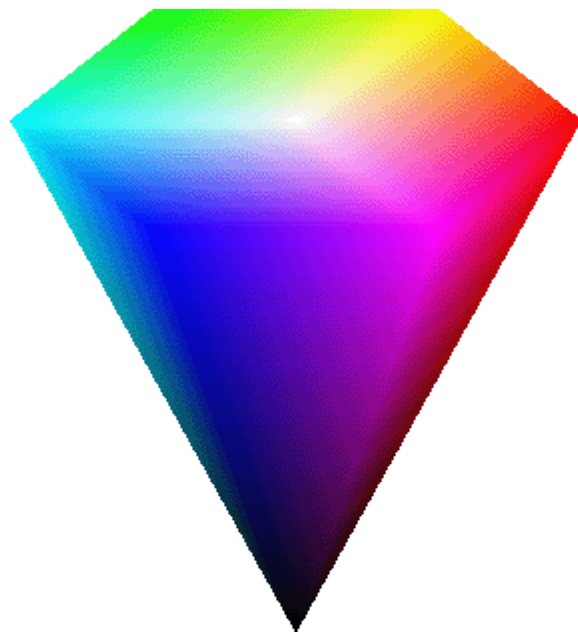
    // Create buffers
    const positionBuffer = gl.createBuffer();
    const colorBuffer = gl.createBuffer();

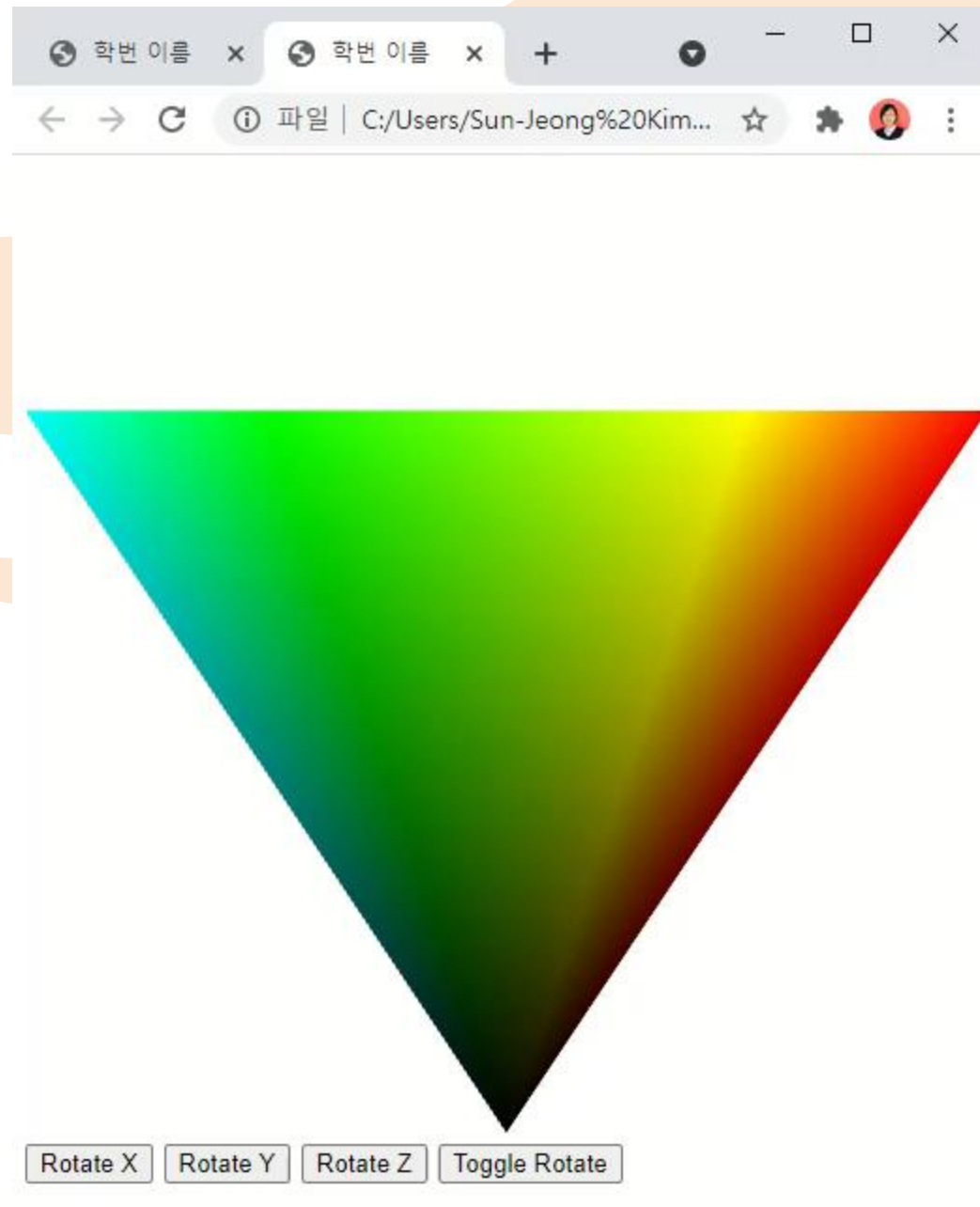
    // Draw
    gl.useProgram(program);
    gl.bufferData(positionBuffer, vertexPos, gl.STATIC_DRAW);
    gl.bufferData(colorBuffer, vertexColor, gl.STATIC_DRAW);
    gl.drawArrays(gl.TRIANGLES, 0, 6);
}
```



연습 문제 (1)

- Hexagonal Pyramid를 그리시오.
 - colorCube.html, js → hexaPyramid.html, js로 복사
 - Vertex 8개
 - Triangle 12개





Line-Preserving Transformations

- Two classes of transformations of importance to CG that preserves lines
 - Affine and projective transformations
- Homogeneous coordinates
 - Vectors: $(x, y, z, \underline{0})$, points: $(x, y, z, w) = (x/w, y/w, z/w, \underline{1})$
 - 4x4 matrices: modeling, viewing, and projection

- OpenGL pipeline



- Model-view: to position objects relative to camera
- Projection: to specify clipping volume and map vertices to a normalized coordinate system

<> transform.html X JS transform.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > body > div

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Transformations</title>
5      <script id="vertex-shader" type="x-shader/x-vertex">
6        attribute vec4 vPosition;
7        attribute vec4 vColor;
8        uniform vec3 theta;
9        varying vec4 fColor;
10
11      void main() {
12        // Compute the sines and cosines of theta for each of
13        // the three axes in one computation
14        vec3 angles = radians(theta);
15        vec3 c = cos(angles);
16        vec3 s = sin(angles);
17
18        // Remember: these matrices are column-major
19        mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
20                       0.0, c.x, s.x, 0.0,
21                       0.0, -s.x, c.x, 0.0,
22                       0.0, 0.0, 0.0, 1.0 );
23
24        mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
25                       0.0, 1.0, 0.0, 0.0,
26                       s.y, 0.0, c.y, 0.0,
27                       0.0, 0.0, 0.0, 1.0 );
28
29        mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
30                       -s.z, c.z, 0.0, 0.0,
31                       0.0, 0.0, 1.0, 0.0,
32                       0.0, 0.0, 0.0, 1.0 );
33
34        gl_Position = rz * ry * rx * vPosition;
35        fColor = vColor;
```



<> transform.html X JS transform.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > body > div > button#buttonT

```
33
34         gl_Position = rz * ry * rx * vPosition;
35         fColor = vColor;
36     }
37 </script>
38
39     <script id="fragment-shader" type="x-shader/x-fragment">
40     precision mediump float;
41     varying vec4 fColor;
42
43     void main() {
44         gl_FragColor = fColor;
45     }
46 </script>
47
48     <script type="text/javascript" src="Common/webgl-utils.js"></script>
49     <script type="text/javascript" src="Common/initShaders.js"></script>
50     <script type="text/javascript" src="Common/MV.js"></script>
51     <script type="text/javascript" src="transform.js"></script>
52 </head>
53 <body>
54     <canvas id="gl-canvas" width="512" height="512">
55         Oops... your browser doesn't support the HTML5 canvas element!
56     </canvas>
57     <div>
58         <button id="xButton">Rotate X</button>
59         <button id="yButton">Rotate Y</button>
60         <button id="zButton">Rotate Z</button>
61         <button id="buttonT">Toggle Rotate</button>
62     </div>
63 </body>
64 </html>
```



<> transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > generateHexaPyramid

```
1  var gl;
2  var points = [];
3  var colors = [];
4
5  var axis = 0;
6  var theta = [0, 0, 0];
7  var thetaLoc;
8
9  var rotation = false;
10
11 window.onload = function init()
12 {
13     var canvas = document.getElementById("gl-canvas");
14
15     gl = WebGLUtils.setupWebGL(canvas);
16     if( !gl ) {
17         alert("WebGL isn't available!");
18     }
19
20     generateColorCube();
21     generateHexaPyramid();
22
23     // Configure WebGL
24     gl.viewport(0, 0, canvas.width, canvas.height);
25     gl.clearColor(0.9, 0.9, 0.9, 1.0);
26
27     // Enable hidden-surface removal
28     gl.enable(gl.DEPTH_TEST);
29
30     // Load shaders and initialize attribute buffers
31     var program = initShaders(gl, "vertex-shader", "fragment-shader");
32     gl.useProgram(program);
33
34     // Load the data into the GPU
35     var bufferId = gl.createBuffer();
```



<> transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > generateHexaPyramid

```
34 // Load the data into the GPU
35 var bufferId = gl.createBuffer();
36 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
37 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
38
39 // Associate our shader variables with our data buffer
40 var vPosition = gl.getAttribLocation(program, "vPosition");
41 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
42 gl.enableVertexAttribArray(vPosition);
43
44 // Create a buffer object, initialize it, and associate it with
45 // the associated attribute variable in our vertex shader
46 var cBufferId = gl.createBuffer();
47 gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
48 gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);
49
50 var vColor = gl.getAttribLocation(program, "vColor");
51 gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);
52 gl.enableVertexAttribArray(vColor);
53
54 thetaLoc = gl.getUniformLocation(program, "theta");
55 //gl.uniform3fv(thetaLoc, theta);
56
57 // Event listeners for buttons
58 document.getElementById("xButton").onclick = function () {
59     axis = 0;
60 };
61 document.getElementById("yButton").onclick = function () {
62     axis = 1;
63 };
64 document.getElementById("zButton").onclick = function () {
65     axis = 2;
66 };
67 document.getElementById("buttonT").onclick = function () {
68     rotation = !rotation;
```

```
function generateHexaPyramid() {
    // Create a new WebGL context
    const canvas = document.getElementById('canvas');
    const gl = canvas.getContext('webgl');

    // Load the vertex and fragment shaders
    const vertexShader = loadShader(gl, 'vertexShader.js');
    const fragmentShader = loadShader(gl, 'fragmentShader.js');

    // Create a program object
    const program = gl.createProgram();
    gl.attachShader(program, vertexShader);
    gl.attachShader(program, fragmentShader);
    gl.linkProgram(program);

    // Create a buffer object, initialize it, and associate it with
    // the associated attribute variable in our vertex shader
    const points = [
        // Bottom hexagon vertices
        -1.0, -1.0, 0.0, 1.0, -1.0, 0.0, 1.0, 1.0, 0.0, -1.0, 1.0, 0.0,
        // Top hexagon vertices
        -1.0, 1.0, 0.0, 1.0, 1.0, 0.0, -1.0, 1.0, 0.0, -1.0, -1.0, 0.0,
        // Center point
        0.0, 0.0, 1.0
    ];
    const colors = [
        // Bottom hexagon colors
        0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
        // Top hexagon colors
        0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
        // Center point color
        0.0, 0.0, 0.0, 0.0
    ];

    // Create a buffer object, initialize it, and associate it with
    // the associated attribute variable in our vertex shader
    const bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Create a buffer object, initialize it, and associate it with
    // the associated attribute variable in our vertex shader
    const cBufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    const vPosition = gl.getAttribLocation(program, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

    const vColor = gl.getAttribLocation(program, "vColor");
    gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vColor);

    const thetaLoc = gl.getUniformLocation(program, "theta");
    //gl.uniform3fv(thetaLoc, theta);

    // Event listeners for buttons
    document.getElementById("xButton").onclick = function () {
        axis = 0;
    };
    document.getElementById("yButton").onclick = function () {
        axis = 1;
    };
    document.getElementById("zButton").onclick = function () {
        axis = 2;
    };
    document.getElementById("buttonT").onclick = function () {
        rotation = !rotation;
```

transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > generateHexaPyramid

```
67     document.getElementById("buttonT").onclick = function () {
68         rotation = !rotation;
69     };
70
71     render();
72 };
73
74 function render() {
75     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
76
77     if( rotation ) {
78         theta[axis] += 2.0;
79     }
80     gl.uniform3fv(thetaLoc, theta)
81
82     gl.drawArrays(gl.TRIANGLES, 0, points.length);
83
84     window.requestAnimationFrame(render);
85 }
86
87 function generateColorCube() {
88     quad(1, 0, 3, 2);
89     quad(2, 3, 7, 6);
90     quad(3, 0, 4, 7);
91     quad(4, 5, 6, 7);
92     quad(5, 4, 0, 1);
93     quad(6, 5, 1, 2);
94 }
95
96 function quad(a, b, c, d) {
97     vertexPos = [
98         vec4(-0.5, -0.5, -0.5, 1.0),
99         vec4( 0.5, -0.5, -0.5, 1.0),
100        vec4( 0.5,  0.5, -0.5, 1.0),
101        vec4(-0.5,  0.5, -0.5, 1.0),
```

```
102    ];
103    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexPos), gl.STATIC_DRAW);
104    vertexBuffer = gl.createBuffer();
105    gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
106    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexPos), gl.STATIC_DRAW);
107    vertexBuffer.bind();
108    gl.vertexAttribPointer(shaderProgram.vertexAttribPointer, 4, gl.FLOAT, false, 4*4, 0);
109    gl.drawArrays(gl.TRIANGLES, 0, 4);
110
111    // Generate the other faces of the cube
112    quad(1, 0, 3, 2);
113    quad(2, 3, 7, 6);
114    quad(3, 0, 4, 7);
115    quad(4, 5, 6, 7);
116    quad(5, 4, 0, 1);
117    quad(6, 5, 1, 2);
118
119    // Draw the cube
120    gl.drawArrays(gl.TRIANGLES, 0, 24);
121
122    // Request the next animation frame
123    window.requestAnimationFrame(render);
124
125    // End of the main loop
126    return;
127
128 // End of the main loop
129 return;
```

```

96 function quad(a, b, c, d) {
97     vertexPos = [
98         vec4(-0.5, -0.5, -0.5, 1.0),
99         vec4( 0.5, -0.5, -0.5, 1.0),
100        vec4( 0.5,  0.5, -0.5, 1.0),
101        vec4(-0.5,  0.5, -0.5, 1.0),
102        vec4(-0.5, -0.5,  0.5, 1.0),
103        vec4( 0.5, -0.5,  0.5, 1.0),
104        vec4( 0.5,  0.5,  0.5, 1.0),
105        vec4(-0.5,  0.5,  0.5, 1.0)
106    ];
107
108    vertexColor = [
109        vec4(0.0, 0.0, 0.0, 1.0), // black
110        vec4(1.0, 0.0, 0.0, 1.0), // red
111        vec4(1.0, 1.0, 0.0, 1.0), // yellow
112        vec4(0.0, 1.0, 0.0, 1.0), // green
113        vec4(0.0, 0.0, 1.0, 1.0), // blue
114        vec4(1.0, 0.0, 1.0, 1.0), // magenta
115        vec4(1.0, 1.0, 1.0, 1.0), // white
116        vec4(0.0, 1.0, 1.0, 1.0)  // cyan
117    ];
118
119    // We need to partition the quad into two triangles in order for WebGL
120    // to be able to render it. In this case, we create two triangles from
121    // the quad indices.
122    var index = [ a, b, c, a, c, d ];
123    for(var i=0; i<index.length; i++) {
124        points.push(vertexPos[index[i]]);
125        colors.push(vertexColor[index[i]]);
126    }
127 }
128
129 function generateHexaPyramid() {
130     vertexPos = [

```



<> transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > generateHexaPyramid

```
129 function generateHexaPyramid() {
130     vertexPos = [
131         vec4( 0.0,  0.5,  0.0,  1.0),
132         vec4( 1.0,  0.5,  0.0,  1.0),
133         vec4( 0.5,  0.5, -0.866, 1.0),
134         vec4(-0.5,  0.5, -0.866, 1.0),
135         vec4(-1.0,  0.5,  0.0,  1.0),
136         vec4(-0.5,  0.5,  0.866, 1.0),
137         vec4( 0.5,  0.5,  0.866, 1.0),
138         vec4( 0.0, -1.0,  0.0,  1.0)
139     ];
140
141     vertexColor = [
142         vec4(1.0, 1.0, 1.0, 1.0), // white
143         vec4(1.0, 0.0, 0.0, 1.0), // red
144         vec4(1.0, 1.0, 0.0, 1.0), // yellow
145         vec4(0.0, 1.0, 0.0, 1.0), // green
146         vec4(0.0, 1.0, 1.0, 1.0), // cyan
147         vec4(0.0, 0.0, 1.0, 1.0), // blue
148         vec4(1.0, 0.0, 1.0, 1.0), // magenta
149         vec4(0.0, 0.0, 0.0, 1.0) // black
150     ];
151
152     for(var i=1; i<6; i++) {
153         points.push(vertexPos[0]);
154         colors.push(vertexColor[0]);
155         points.push(vertexPos[i]);
156         colors.push(vertexColor[i]);
157         points.push(vertexPos[i+1]);
158         colors.push(vertexColor[i+1]);
159     }
160     points.push(vertexPos[0]);
161     colors.push(vertexColor[0]);
162     points.push(vertexPos[6]);
163     colors.push(vertexColor[6]);
```

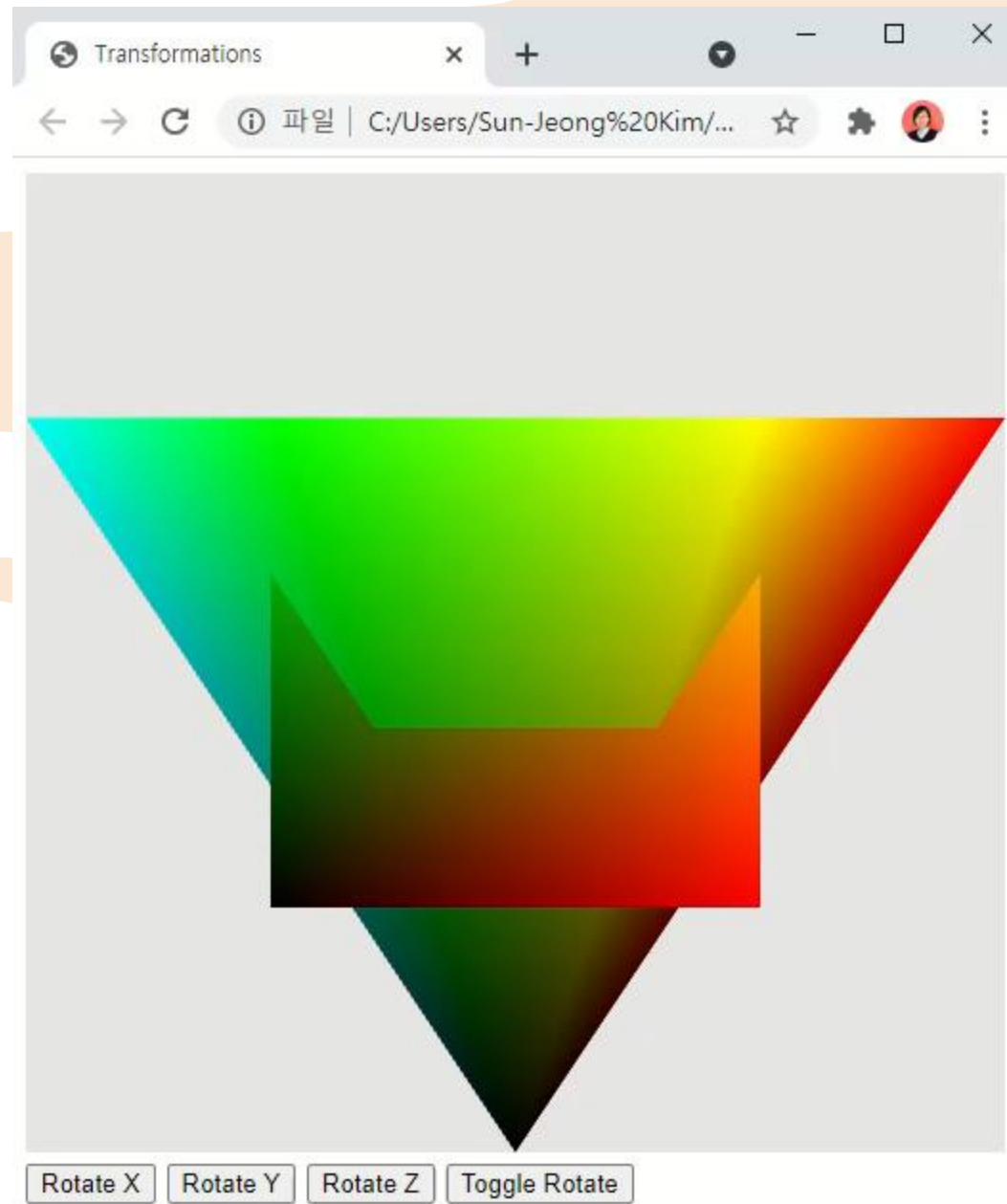
```
1 // HexaPyramid
2 // 6 vertices, 12 edges, 5 faces
3 // 6 vertices, 12 edges, 5 faces
4 // 6 vertices, 12 edges, 5 faces
5 // 6 vertices, 12 edges, 5 faces
6 // 6 vertices, 12 edges, 5 faces
7 // 6 vertices, 12 edges, 5 faces
8 // 6 vertices, 12 edges, 5 faces
9 // 6 vertices, 12 edges, 5 faces
10 // 6 vertices, 12 edges, 5 faces
11 // 6 vertices, 12 edges, 5 faces
12 // 6 vertices, 12 edges, 5 faces
13 // 6 vertices, 12 edges, 5 faces
14 // 6 vertices, 12 edges, 5 faces
15 // 6 vertices, 12 edges, 5 faces
16 // 6 vertices, 12 edges, 5 faces
17 // 6 vertices, 12 edges, 5 faces
18 // 6 vertices, 12 edges, 5 faces
19 // 6 vertices, 12 edges, 5 faces
20 // 6 vertices, 12 edges, 5 faces
21 // 6 vertices, 12 edges, 5 faces
22 // 6 vertices, 12 edges, 5 faces
23 // 6 vertices, 12 edges, 5 faces
24 // 6 vertices, 12 edges, 5 faces
25 // 6 vertices, 12 edges, 5 faces
26 // 6 vertices, 12 edges, 5 faces
27 // 6 vertices, 12 edges, 5 faces
28 // 6 vertices, 12 edges, 5 faces
29 // 6 vertices, 12 edges, 5 faces
30 // 6 vertices, 12 edges, 5 faces
31 // 6 vertices, 12 edges, 5 faces
32 // 6 vertices, 12 edges, 5 faces
33 // 6 vertices, 12 edges, 5 faces
34 // 6 vertices, 12 edges, 5 faces
35 // 6 vertices, 12 edges, 5 faces
36 // 6 vertices, 12 edges, 5 faces
37 // 6 vertices, 12 edges, 5 faces
38 // 6 vertices, 12 edges, 5 faces
39 // 6 vertices, 12 edges, 5 faces
40 // 6 vertices, 12 edges, 5 faces
41 // 6 vertices, 12 edges, 5 faces
42 // 6 vertices, 12 edges, 5 faces
43 // 6 vertices, 12 edges, 5 faces
44 // 6 vertices, 12 edges, 5 faces
45 // 6 vertices, 12 edges, 5 faces
46 // 6 vertices, 12 edges, 5 faces
47 // 6 vertices, 12 edges, 5 faces
48 // 6 vertices, 12 edges, 5 faces
49 // 6 vertices, 12 edges, 5 faces
50 // 6 vertices, 12 edges, 5 faces
51 // 6 vertices, 12 edges, 5 faces
52 // 6 vertices, 12 edges, 5 faces
53 // 6 vertices, 12 edges, 5 faces
54 // 6 vertices, 12 edges, 5 faces
55 // 6 vertices, 12 edges, 5 faces
56 // 6 vertices, 12 edges, 5 faces
57 // 6 vertices, 12 edges, 5 faces
58 // 6 vertices, 12 edges, 5 faces
59 // 6 vertices, 12 edges, 5 faces
60 // 6 vertices, 12 edges, 5 faces
61 // 6 vertices, 12 edges, 5 faces
62 // 6 vertices, 12 edges, 5 faces
63 // 6 vertices, 12 edges, 5 faces
64 // 6 vertices, 12 edges, 5 faces
65 // 6 vertices, 12 edges, 5 faces
66 // 6 vertices, 12 edges, 5 faces
67 // 6 vertices, 12 edges, 5 faces
68 // 6 vertices, 12 edges, 5 faces
69 // 6 vertices, 12 edges, 5 faces
70 // 6 vertices, 12 edges, 5 faces
71 // 6 vertices, 12 edges, 5 faces
72 // 6 vertices, 12 edges, 5 faces
73 // 6 vertices, 12 edges, 5 faces
74 // 6 vertices, 12 edges, 5 faces
75 // 6 vertices, 12 edges, 5 faces
76 // 6 vertices, 12 edges, 5 faces
77 // 6 vertices, 12 edges, 5 faces
78 // 6 vertices, 12 edges, 5 faces
79 // 6 vertices, 12 edges, 5 faces
80 // 6 vertices, 12 edges, 5 faces
81 // 6 vertices, 12 edges, 5 faces
82 // 6 vertices, 12 edges, 5 faces
83 // 6 vertices, 12 edges, 5 faces
84 // 6 vertices, 12 edges, 5 faces
85 // 6 vertices, 12 edges, 5 faces
86 // 6 vertices, 12 edges, 5 faces
87 // 6 vertices, 12 edges, 5 faces
88 // 6 vertices, 12 edges, 5 faces
89 // 6 vertices, 12 edges, 5 faces
90 // 6 vertices, 12 edges, 5 faces
91 // 6 vertices, 12 edges, 5 faces
92 // 6 vertices, 12 edges, 5 faces
93 // 6 vertices, 12 edges, 5 faces
94 // 6 vertices, 12 edges, 5 faces
95 // 6 vertices, 12 edges, 5 faces
96 // 6 vertices, 12 edges, 5 faces
97 // 6 vertices, 12 edges, 5 faces
98 // 6 vertices, 12 edges, 5 faces
99 // 6 vertices, 12 edges, 5 faces
100 // 6 vertices, 12 edges, 5 faces
```

<> transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > generateHexaPyramid

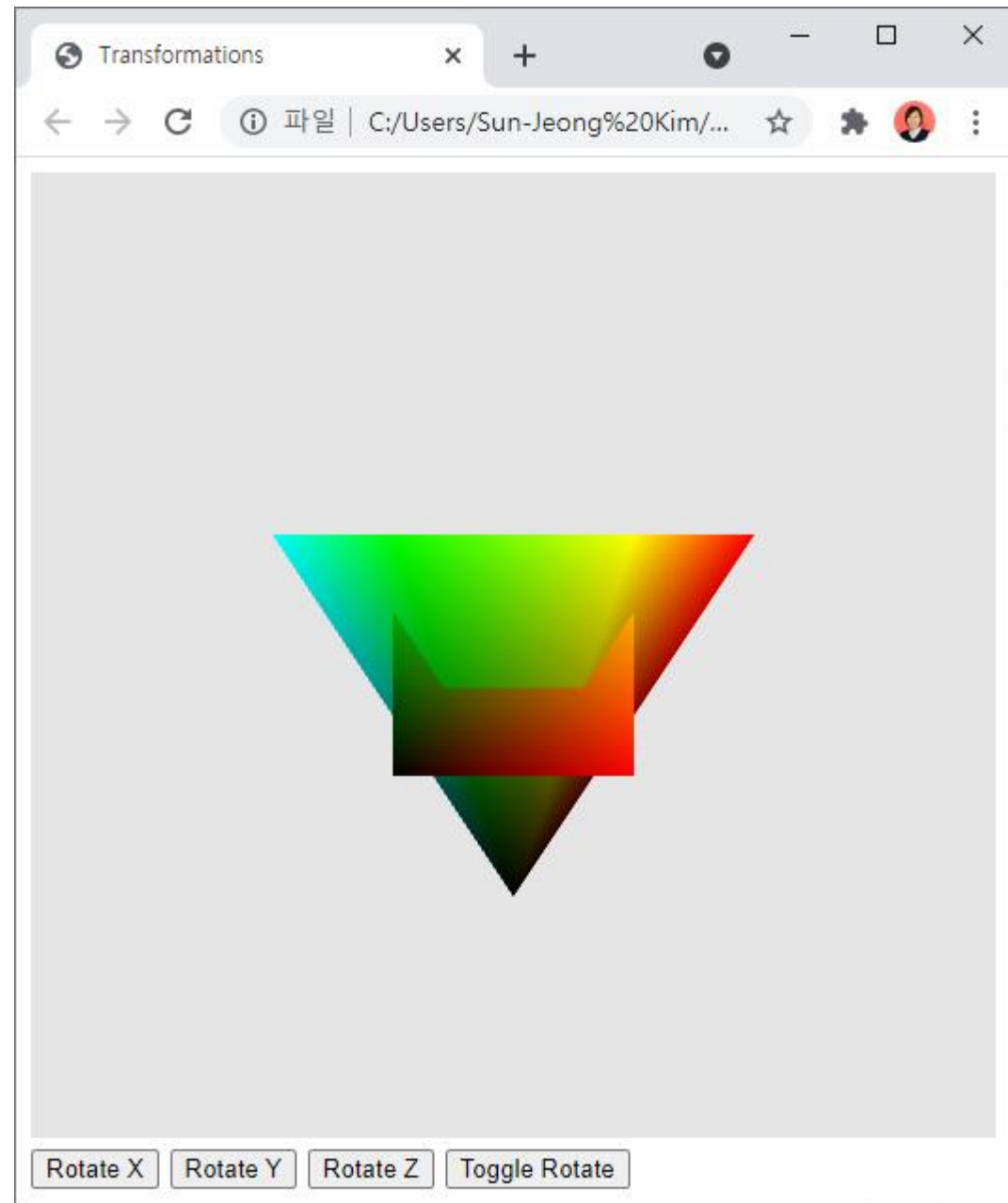
```
151
152     for(var i=1; i<6; i++) {
153         points.push(vertexPos[0]);
154         colors.push(vertexColor[0]);
155         points.push(vertexPos[i]);
156         colors.push(vertexColor[i]);
157         points.push(vertexPos[i+1]);
158         colors.push(vertexColor[i+1]);
159     }
160     points.push(vertexPos[0]);
161     colors.push(vertexColor[0]);
162     points.push(vertexPos[6]);
163     colors.push(vertexColor[6]);
164     points.push(vertexPos[1]);
165     colors.push(vertexColor[1]);
166
167     for(var i=1; i<6; i++) {
168         points.push(vertexPos[7]);
169         colors.push(vertexColor[7]);
170         points.push(vertexPos[i+1]);
171         colors.push(vertexColor[i+1]);
172         points.push(vertexPos[i]);
173         colors.push(vertexColor[i]);
174     }
175     points.push(vertexPos[7]);
176     colors.push(vertexColor[7]);
177     points.push(vertexPos[1]);
178     colors.push(vertexColor[1]);
179     points.push(vertexPos[6]);
180     colors.push(vertexColor[6]);
181
182
```

```
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```



연습 문제 (2)

- 오브젝트의 크기를 반으로 줄이시오.



<> transform.html X JS transform.js

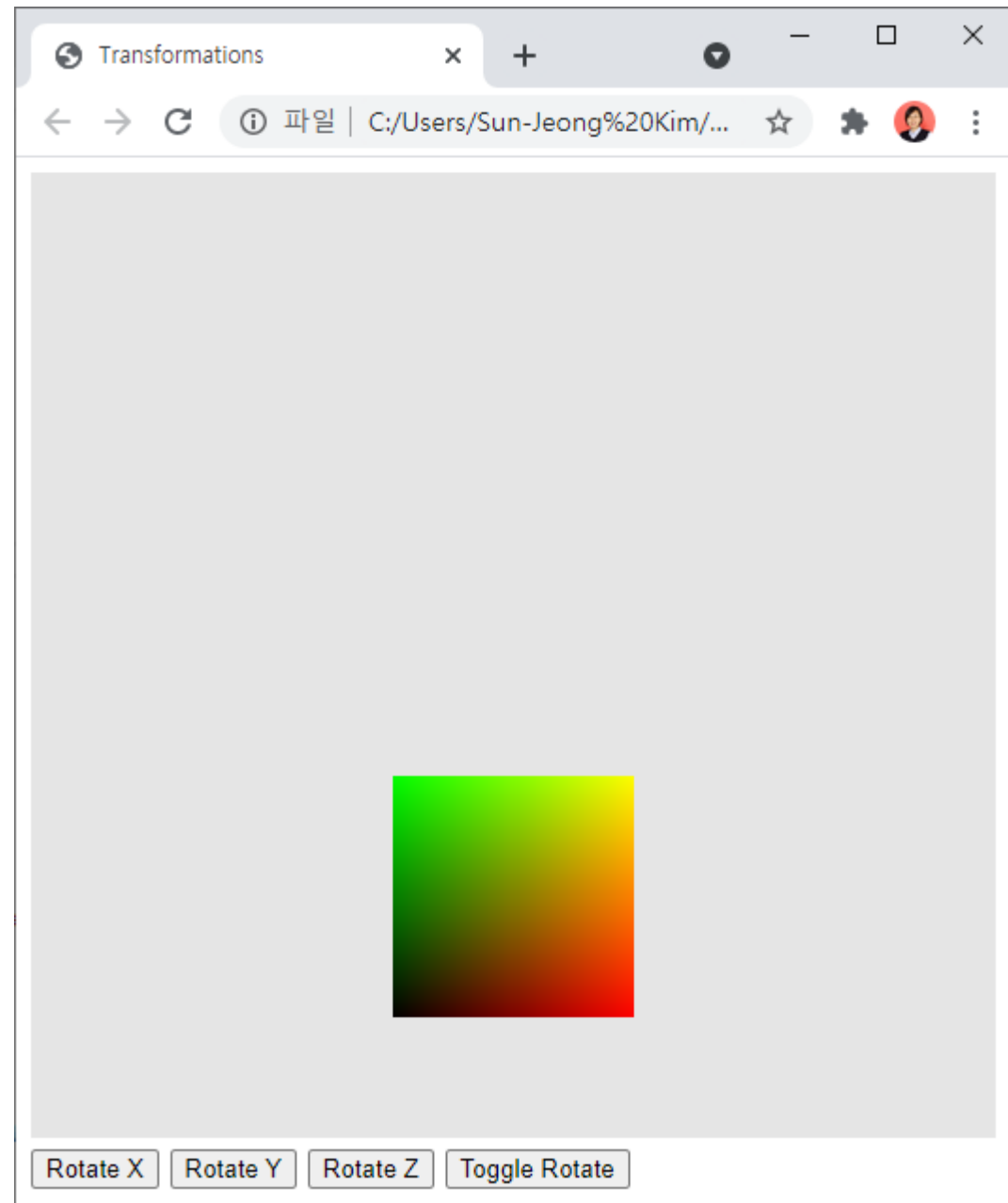
C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader

```
11 void main() {
12     // Compute the sines and cosines of theta for each of
13     // the three axes in one computation
14     vec3 angles = radians(theta);
15     vec3 c = cos(angles);
16     vec3 s = sin(angles);
17
18     // Remember: these matrices are column-major
19     mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
20                   0.0, c.x, s.x, 0.0,
21                   0.0, -s.x, c.x, 0.0,
22                   0.0, 0.0, 0.0, 1.0 );
23
24     mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
25                   0.0, 1.0, 0.0, 0.0,
26                   s.y, 0.0, c.y, 0.0,
27                   0.0, 0.0, 0.0, 1.0 );
28
29     mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
30                   -s.z, c.z, 0.0, 0.0,
31                   0.0, 0.0, 1.0, 0.0,
32                   0.0, 0.0, 0.0, 1.0 );
33
34     mat4 sc = mat4( 0.5, 0, 0, 0,
35                   0, 0.5, 0, 0,
36                   0, 0, 0.5, 0,
37                   0, 0, 0, 1.0);
38
39     gl_Position = rz * ry * rx * sc * vPosition;
40     fColor = vColor;
41 }
42 </script>
43
44 <script id="fragment-shader" type="x-shader/x-fragment">
```



연습 문제 (3)

- Cube를 y축 방향으로 -0.5 만큼 이동시키시오.



<> transform.html X JS transform.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader

```
14   vec3 angles = radians(theta);
15   vec3 c = cos(angles);
16   vec3 s = sin(angles);
17
18   // Remember: these matrices are column-major
19   mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
20                  0.0, c.x, s.x, 0.0,
21                  0.0, -s.x, c.x, 0.0,
22                  0.0, 0.0, 0.0, 1.0 );
23
24   mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
25                  0.0, 1.0, 0.0, 0.0,
26                  s.y, 0.0, c.y, 0.0,
27                  0.0, 0.0, 0.0, 1.0 );
28
29   mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
30                  -s.z, c.z, 0.0, 0.0,
31                  0.0, 0.0, 1.0, 0.0,
32                  0.0, 0.0, 0.0, 1.0 );
33
34   mat4 sc = mat4( 0.5, 0, 0, 0,
35                  0, 0.5, 0, 0,
36                  0, 0, 0.5, 0,
37                  0, 0, 0, 1.0);
38
39   mat4 tr = mat4( 1, 0, 0, 0,
40                  0, 1, 0, 0,
41                  0, 0, 1, 0,
42                  0, -0.5, 0, 1);
43
44   gl_Position = tr * rz * ry * rx * sc * vPosition;
45   fColor = vColor;
46 }
47 </script>
```



transform.html JS transform.js X

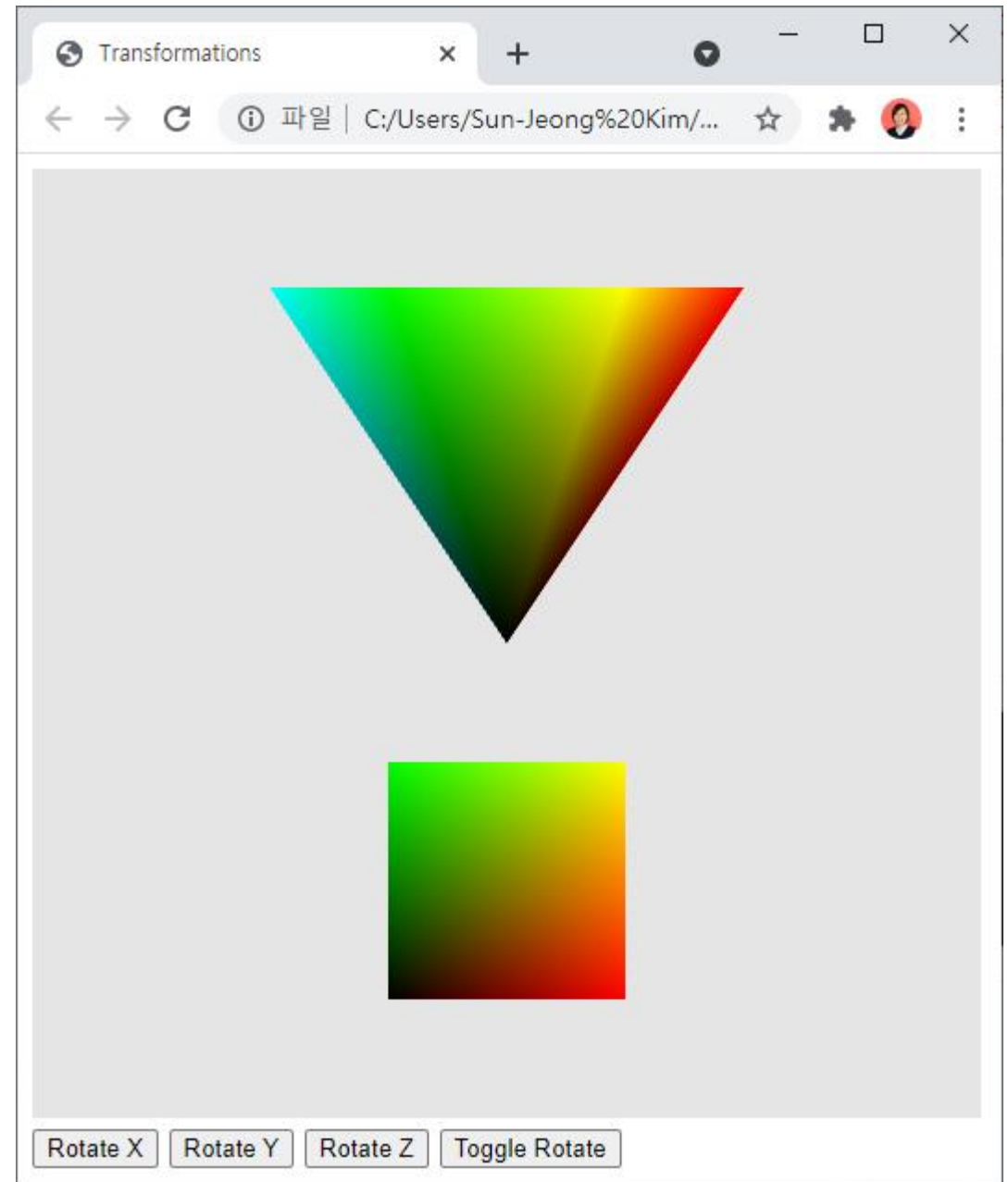
C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > init

```
1  var gl;
2  var points = [];
3  var colors = [];
4
5  var axis = 0;
6  var theta = [0, 0, 0];
7  var thetaLoc;
8
9  var rotation = false;
10
11 window.onload = function init()
12 {
13     var canvas = document.getElementById("gl-canvas");
14
15     gl = WebGLUtils.setupWebGL(canvas);
16     if( !gl ) {
17         alert("WebGL isn't available!");
18     }
19
20     generateColorCube();
21     //generateHexaPyramid();
22
23     // Configure WebGL
24     gl.viewport(0, 0, canvas.width, canvas.height);
25     gl.clearColor(0.9, 0.9, 0.9, 1.0);
26
27     // Enable hidden-surface removal
28     gl.enable(gl.DEPTH_TEST);
29
30     // Load shaders and initialize attribute buffers
31     var program = initShaders(gl, "vertex-shader", "fragment-shader");
32     gl.useProgram(program);
33
34     // Load the data into the GPU
35     var bufferId = gl.createBuffer();
```



연습 문제 (4)

- Hexagonal Pyramid를 y축 방향으로 0.5만큼 이동 시키시오.



<> transform.html X JS transform.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Transformations</title>
5      <script id="vertex-shader" type="x-shader/x-vertex">
6        attribute vec4 vPosition;
7        attribute vec4 vColor;
8        uniform vec3 theta;
9        uniform vec3 dist;
10       varying vec4 fColor;
11
12       void main() {
13         // Compute the sines and cosines of theta for each of
14         // the three axes in one computation
15         vec3 angles = radians(theta);
16         vec3 c = cos(angles);
17         vec3 s = sin(angles);
18
19         // Remember: these matrices are column-major
20         mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
21                        0.0, c.x, s.x, 0.0,
22                        0.0, -s.x, c.x, 0.0,
23                        0.0, 0.0, 0.0, 1.0 );
24
25         mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
26                        0.0, 1.0, 0.0, 0.0,
27                        s.y, 0.0, c.y, 0.0,
28                        0.0, 0.0, 0.0, 1.0 );
29
30         mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
31                        -s.z, c.z, 0.0, 0.0,
32                        0.0, 0.0, 1.0, 0.0,
33                        0.0, 0.0, 0.0, 1.0 );
34
35         mat4 sc = mat4( 0.5, 0.0, 0.0, 0.0,
```



<> transform.html X JS transform.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader

```
15     vec3 angles = radians(theta);
16     vec3 c = cos(angles);
17     vec3 s = sin(angles);
18
19     // Remember: these matrices are column-major
20     mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
21                   0.0, c.x, s.x, 0.0,
22                   0.0, -s.x, c.x, 0.0,
23                   0.0, 0.0, 0.0, 1.0 );
24
25     mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
26                   0.0, 1.0, 0.0, 0.0,
27                   s.y, 0.0, c.y, 0.0,
28                   0.0, 0.0, 0.0, 1.0 );
29
30     mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
31                   -s.z, c.z, 0.0, 0.0,
32                   0.0, 0.0, 1.0, 0.0,
33                   0.0, 0.0, 0.0, 1.0 );
34
35     mat4 sc = mat4( 0.5, 0, 0, 0,
36                   0, 0.5, 0, 0,
37                   0, 0, 0.5, 0,
38                   0, 0, 0, 1.0);
39
40     mat4 tr = mat4( 1, 0, 0, 0,
41                   0, 1, 0, 0,
42                   0, 0, 1, 0,
43                   dist.x, dist.y, dist.z, 1);
44
45     gl_Position = tr * rz * ry * rx * sc * vPosition;
46     fColor = vColor;
47 }
48 </script>
```



<> transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > render

```
1  var gl;
2  var points = [];
3  var colors = [];
4
5  var axis = 0;
6  var theta = [0, 0, 0];
7  var thetaLoc;
8  var dist = [0, 0, 0];
9  var distLoc;
10
11 var rotation = false;
12
13 window.onload = function init()
14 {
15     var canvas = document.getElementById("gl-canvas");
16
17     gl = WebGLUtils.setupWebGL(canvas);
18     if( !gl ) {
19         alert("WebGL isn't available!");
20     }
21
22     generateColorCube();
23     generateHexaPyramid();
24
25     // Configure WebGL
26     gl.viewport(0, 0, canvas.width, canvas.height);
27     gl.clearColor(0.9, 0.9, 0.9, 1.0);
28
29     // Enable hidden-surface removal
30     gl.enable(gl.DEPTH_TEST);
31
32     // Load shaders and initialize attribute buffers
33     var program = initShaders(gl, "vertex-shader", "fragment-shader");
34     gl.useProgram(program);
35
```



transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > render

```
46 // Create a buffer object, initialize it, and associate it with
47 // the associated attribute variable in our vertex shader
48 var cBufferId = gl.createBuffer();
49 gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
50 gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);
51
52 var vColor = gl.getAttribLocation(program, "vColor");
53 gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);
54 gl.enableVertexAttribArray(vColor);
55
56 thetaLoc = gl.getUniformLocation(program, "theta");
57 //gl.uniform3fv(thetaLoc, theta);
58 distLoc = gl.getUniformLocation(program, "dist");
59 //gl.uniform3fv(distLoc, dist);
60
61 // Event listeners for buttons
62 document.getElementById("xButton").onclick = function () {
63     axis = 0;
64 };
65 document.getElementById("yButton").onclick = function () {
66     axis = 1;
67 };
68 document.getElementById("zButton").onclick = function () {
69     axis = 2;
70 };
71 document.getElementById("buttonT").onclick = function () {
72     rotation = !rotation;
73 };
74
75 render();
76 };
77
78 function render() {
79     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
80 }
```

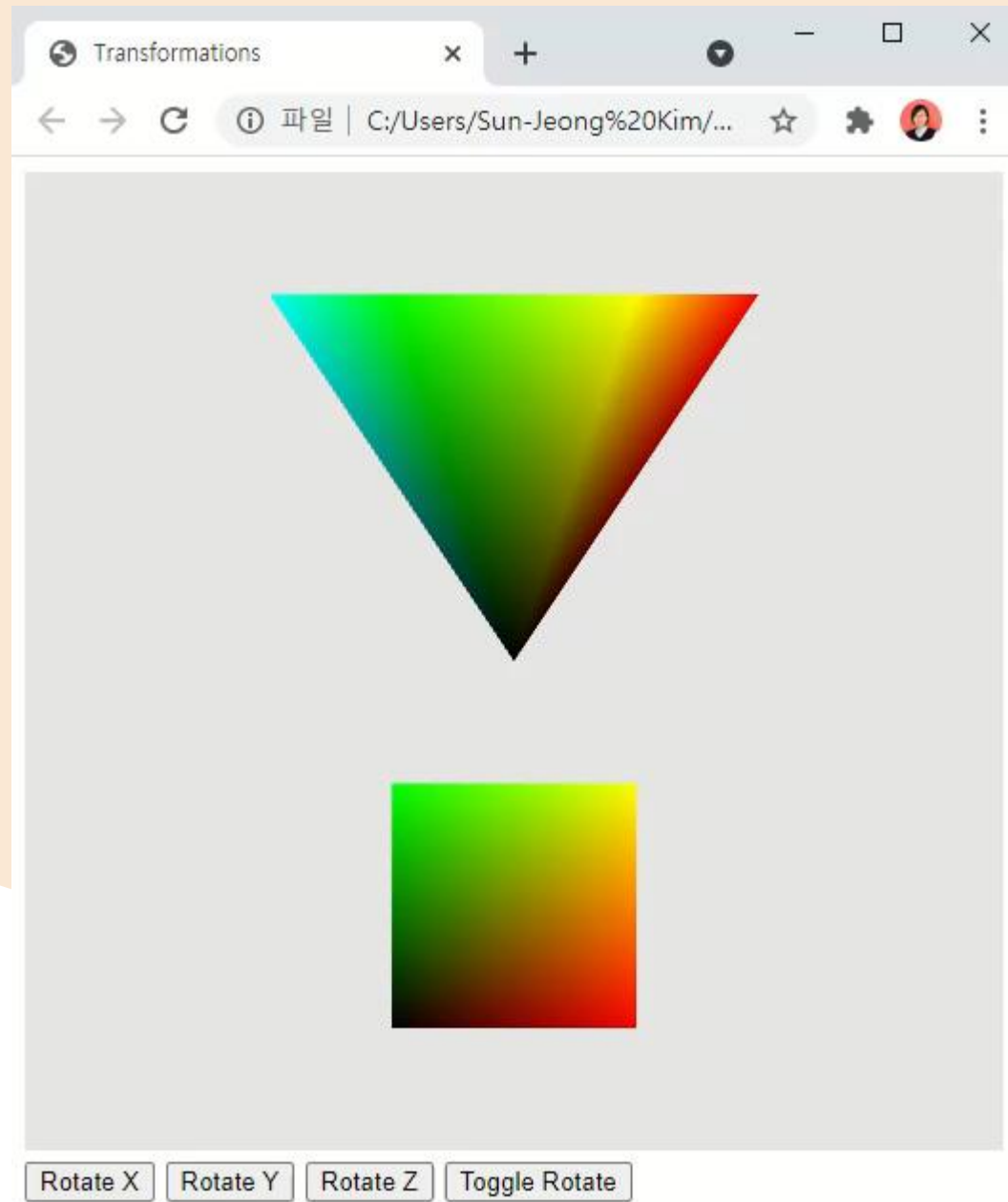
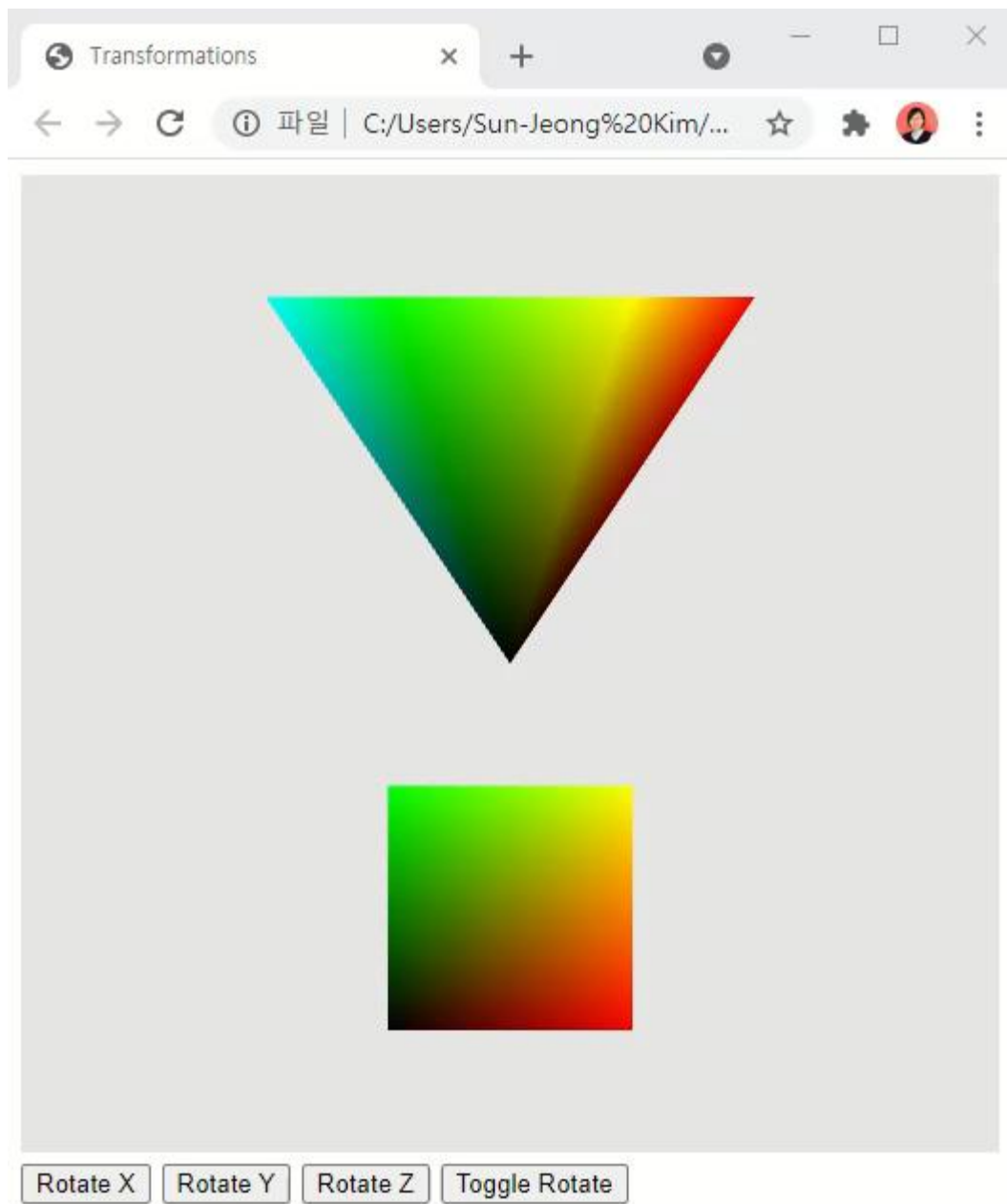


<> transform.html JS transform.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > render

```
78 function render() {
79     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
80
81     if( rotation ) {
82         theta[axis] += 2.0;
83     }
84     gl.uniform3fv(thetaLoc, theta)
85
86     //gl.drawArrays(gl.TRIANGLES, 0, points.length);
87
88     // Draw a color cube (12 triangles * 3 = 36 vertices)
89     dist[1] = -0.5;
90     gl.uniform3fv(distLoc, dist);
91     gl.drawArrays(gl.TRIANGLES, 0, 36);
92
93     // Draw a hexagonal pyramid (12 triangles * 3 = 36 vertices)
94     dist[1] = 0.5;
95     gl.uniform3fv(distLoc, dist);
96     gl.drawArrays(gl.TRIANGLES, 36, 36);
97
98     window.requestAnimationFrame(render);
99 }
100
101 function generateColorCube() {
102     quad(1, 0, 3, 2);
103     quad(2, 3, 7, 6);
104     quad(3, 0, 4, 7);
105     quad(4, 5, 6, 7);
106     quad(5, 4, 0, 1);
107     quad(6, 5, 1, 2);
108 }
109
110 function quad(a, b, c, d) {
111     vertexPos = [
112         vec4(-0.5, -0.5, -0.5, 1.0)
```





<> transform.html X JS transform.js

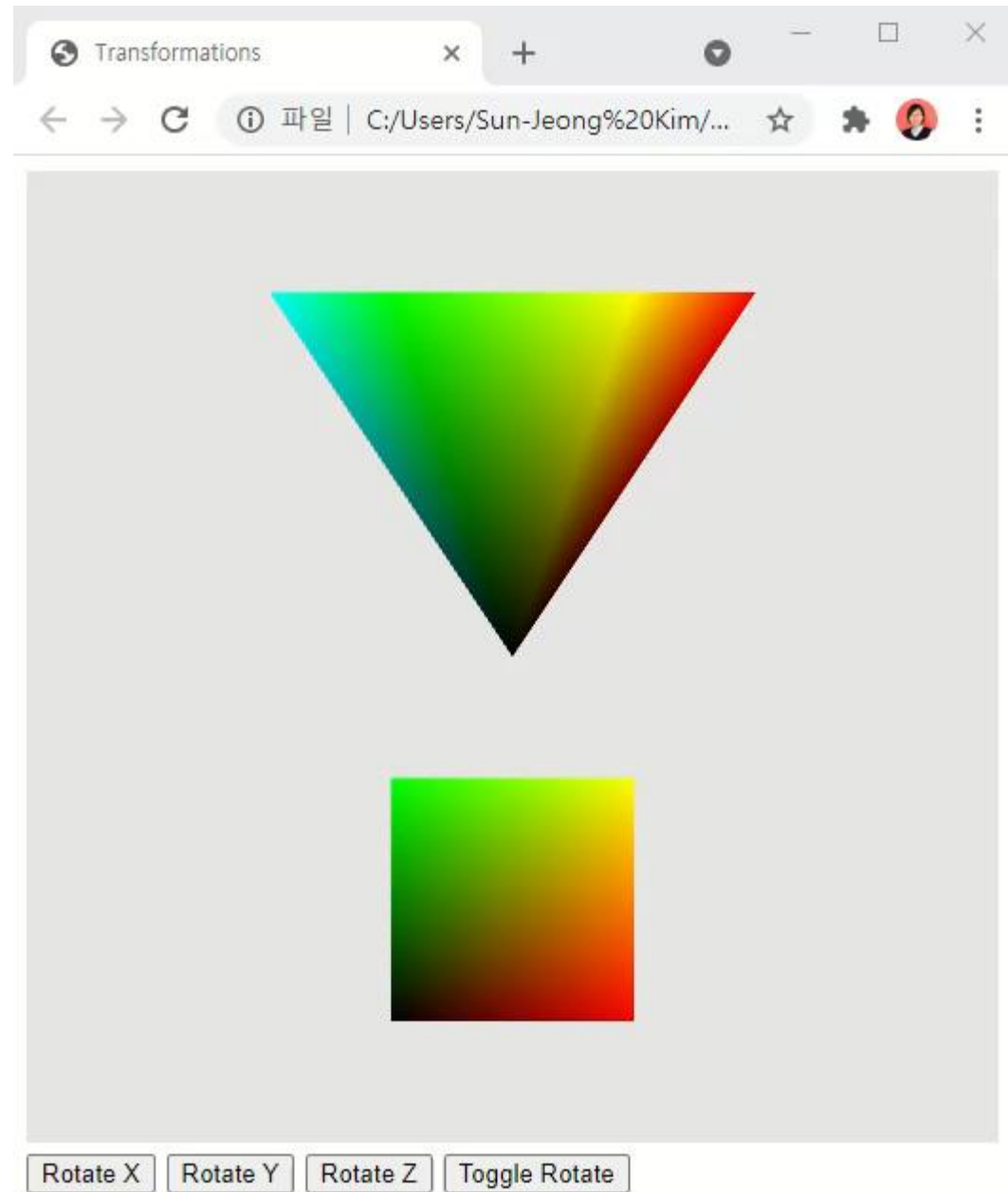
C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader

```
15   vec3 angles = radians(theta);
16   vec3 c = cos(angles);
17   vec3 s = sin(angles);
18
19   // Remember: these matrices are column-major
20   mat4 rx = mat4( 1.0, 0.0, 0.0, 0.0,
21                  0.0, c.x, s.x, 0.0,
22                  0.0, -s.x, c.x, 0.0,
23                  0.0, 0.0, 0.0, 1.0 );
24
25   mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
26                  0.0, 1.0, 0.0, 0.0,
27                  s.y, 0.0, c.y, 0.0,
28                  0.0, 0.0, 0.0, 1.0 );
29
30   mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
31                  -s.z, c.z, 0.0, 0.0,
32                  0.0, 0.0, 1.0, 0.0,
33                  0.0, 0.0, 0.0, 1.0 );
34
35   mat4 sc = mat4( 0.5, 0, 0, 0,
36                  0, 0.5, 0, 0,
37                  0, 0, 0.5, 0,
38                  0, 0, 0, 1.0);
39
40   mat4 tr = mat4( 1, 0, 0, 0,
41                  0, 1, 0, 0,
42                  0, 0, 1, 0,
43                  dist.x, dist.y, dist.z, 1);
44
45   gl_Position = rz * ry * rx * tr * sc * vPosition;
46   fColor = vColor;
47 }
48 </script>
```



연습 문제 (5)

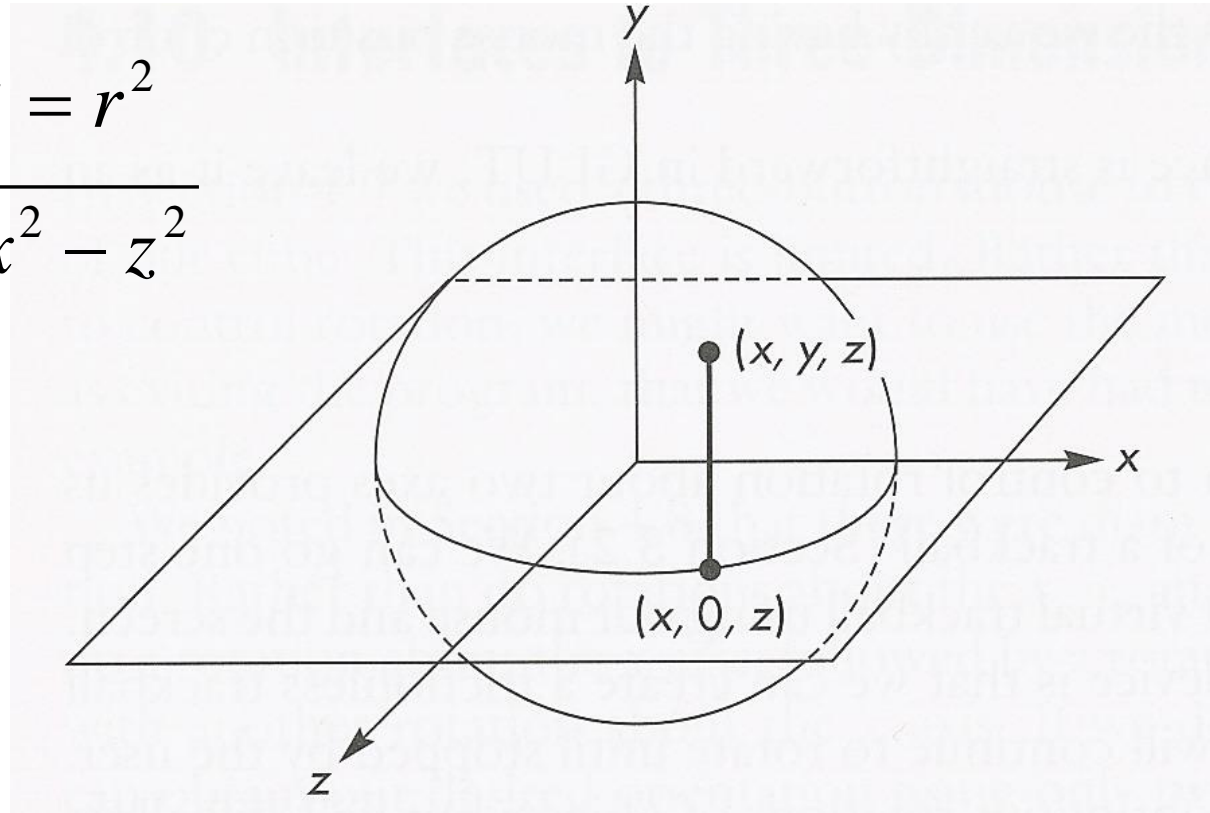
- 두 오브젝트가 서로 반대 방향으로 회전하도록 구현하시오.



Rotation with a Virtual Trackball (1)

- Projection of the trackball position to the plane

$$x^2 + y^2 + z^2 = r^2$$
$$\therefore y = \sqrt{r^2 - x^2 - z^2}$$



Rotation with a Virtual Trackball (2)

- Determination of the orientation of a plane

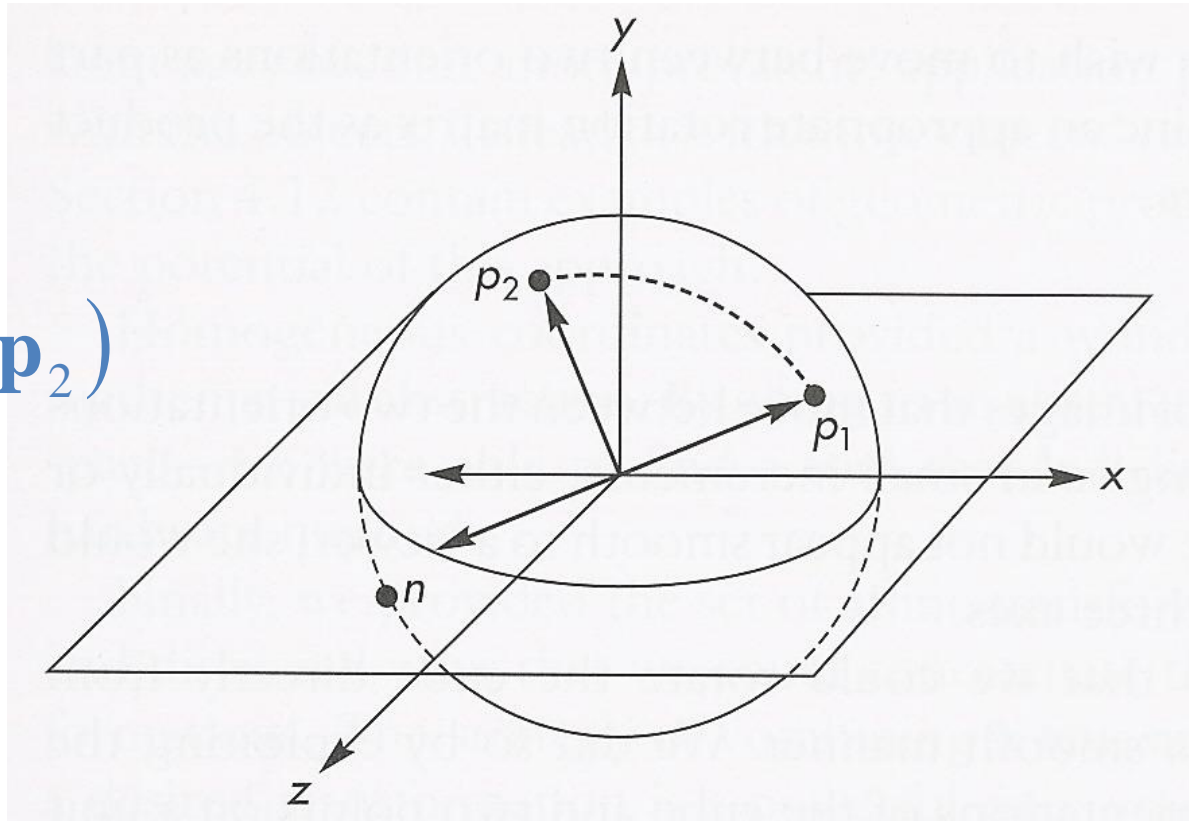
$$\mathbf{n} = \mathbf{p}_1 \times \mathbf{p}_2$$

- Rotation angle

$$\theta = \cos^{-1}(\mathbf{p}_1 \cdot \mathbf{p}_2)$$



Quaternions



Rotations with Quaternions (1)

- Rotation about an arbitrary axis

- Setting up a unit quaternion (**u**: unit vector)

$$s = \cos \frac{\theta}{2}, \quad \mathbf{v} = \mathbf{u} \sin \frac{\theta}{2} = (a, b, c)$$

- Representing any point position **P** in quaternion notation (**p** = (x, y, z))

$$\mathbf{P} = (0, \mathbf{p})$$

- Carrying out with the quaternion operation ($q^{-1} = (s, -\mathbf{v})$)

$$\mathbf{P}' = q\mathbf{P}q^{-1}$$

- Producing the new quaternion

$$\mathbf{P}' = (0, \mathbf{p}')$$

$$\mathbf{p}' = s^2\mathbf{p} + \mathbf{v}(\mathbf{p} \cdot \mathbf{v}) + 2s(\mathbf{v} \times \mathbf{p}) + \mathbf{v} \times (\mathbf{v} \times \mathbf{p})$$

Rotations with Quaternions (2)

- Obtaining the rotation matrix by quaternion multiplication

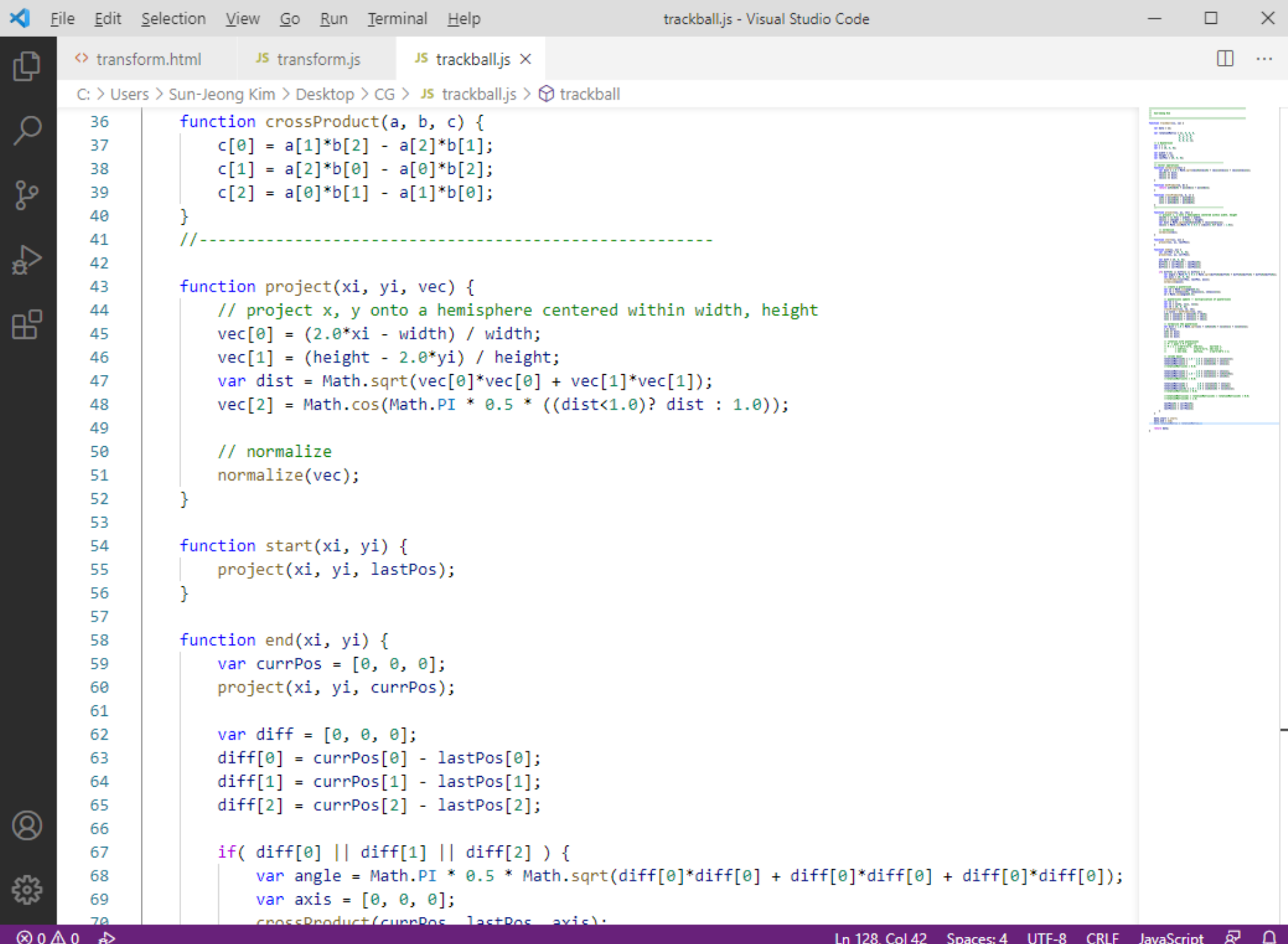
$$\begin{aligned}\mathbf{M}_R(\theta) &= \begin{bmatrix} 1-2b^2-2c^2 & 2ab-2sc & 2ac+2sb \\ 2ab+2sc & 1-2a^2-2c^2 & 2bc-2sa \\ 2ac-2sb & 2bc+2sa & 1-2a^2-2b^2 \end{bmatrix} \\ &= \mathbf{R}_x(-\theta_x)\mathbf{R}_y(-\theta_y)\mathbf{R}_z(\theta)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)\end{aligned}$$

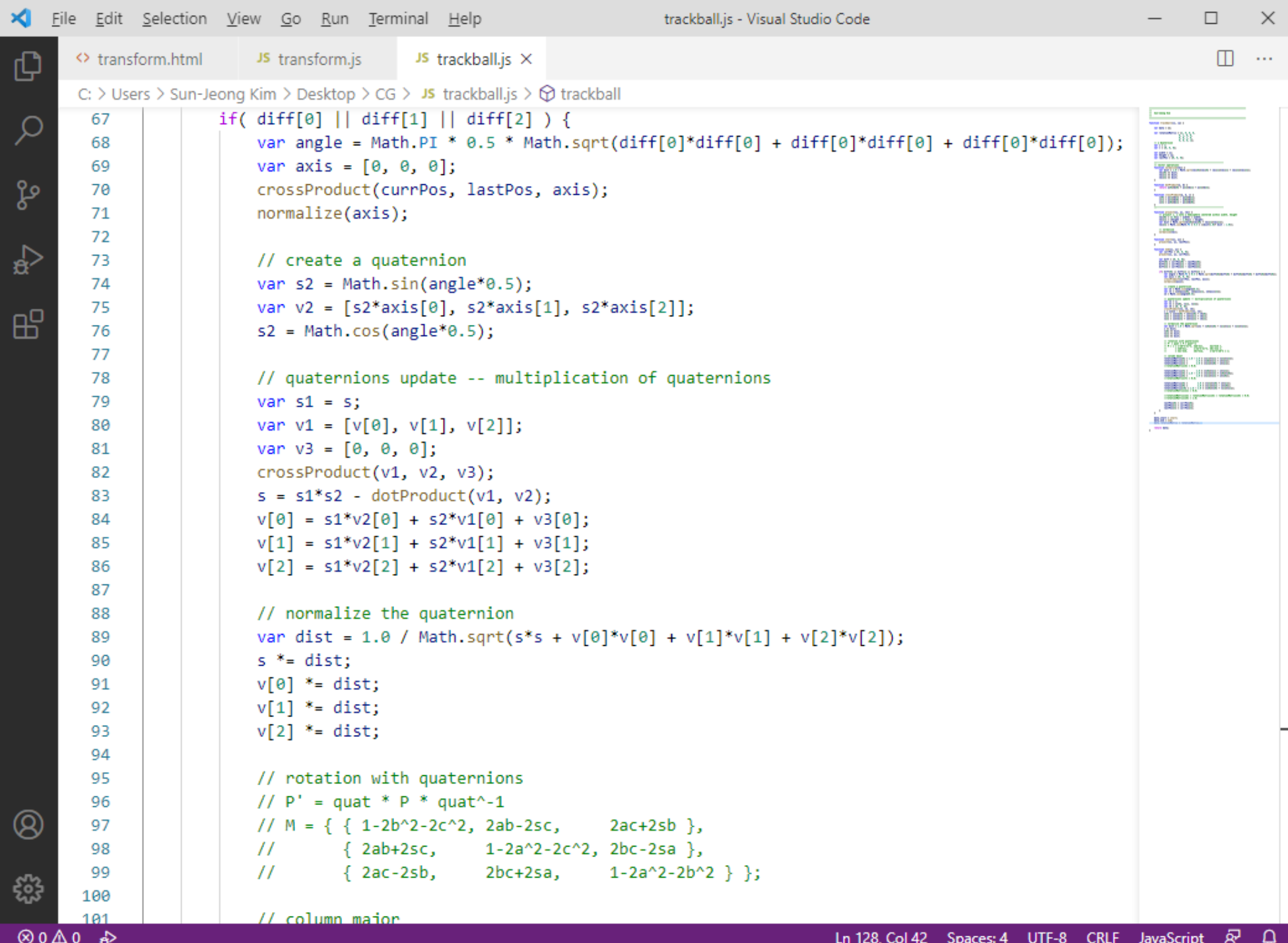
```

1  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2  //
3  //  Sun-Jeong Kim
4  //
5  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
6
7  function trackball(cx, cy) {
8
9      var data = {};
10
11      var rotationMatrix = [1, 0, 0, 0,
12                          |   0, 1, 0, 0,
13                          |   0, 0, 1, 0,
14                          |   0, 0, 0, 1];
15      // a Quaternion
16      var s = 1;
17      var v = [0, 0, 0];
18
19      var width = cx;
20      var height = cy;
21      var lastPos = [0, 0, 0];
22
23      //-----
24      // vector operations
25      function normalize(vec) {
26          var dist = 1.0 / Math.sqrt(vec[0]*vec[0] + vec[1]*vec[1] + vec[2]*vec[2]);
27          vec[0] *= dist;
28          vec[1] *= dist;
29          vec[2] *= dist;
30      }
31
32      function dotProduct(a, b) {
33          return a[0]*b[0] + a[1]*b[1] + a[2]*b[2];
34      }

```







File Edit Selection View Go Run Terminal Helptrackball.js - Visual Studio Code

transform.htmlJS transform.jsJS trackball.js

C: > Users > Sun-Jeong Kim > Desktop > CG > JS trackball.js > trackball

```
100
101 // column major
102 rotationMatrix[0] = 1.0 - 2.0 * (v[1]*v[1] + v[2]*v[2]);
103 rotationMatrix[1] = 2.0 * (v[0]*v[1] + s*v[2]);
104 rotationMatrix[2] = 2.0 * (v[2]*v[0] - s*v[1]);
105 //rotationMatrix[3] = 0.0;
106
107 rotationMatrix[4] = 2.0 * (v[0]*v[1] - s*v[2]);
108 rotationMatrix[5] = 1.0 - 2.0 * (v[2]*v[2] + v[0]*v[0]);
109 rotationMatrix[6] = 2.0 * (v[1]*v[2] + s*v[0]);
110 //rotationMatrix[7] = 0.0;
111
112 rotationMatrix[8] = 2.0 * (v[2]*v[0] + s*v[1]);
113 rotationMatrix[9] = 2.0 * (v[1]*v[2] - s*v[0]);
114 rotationMatrix[10] = 1.0 - 2.0 * (v[0]*v[0] + v[1]*v[1]);
115 //rotationMatrix[11] = 0.0;
116
117 //rotationMatrix[12] = rotationMatrix[13] = rotationMatrix[14] = 0.0;
118 //rotationMatrix[15] = 1.0;
119
120 lastPos[0] = currPos[0];
121 lastPos[1] = currPos[1];
122 lastPos[2] = currPos[2];
123 }
124 }
125
126 data.start = start;
127 data.end = end;
128 data.rotationMatrix = rotationMatrix;
129
130 return data;
131
132
```

trackball.js

Ln 128, Col 42 Spaces: 4 UTF-8 CRLF JavaScript

transform.html X

JS transform.js

JS trackball.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader

1

<!DOCTYPE html>

2

<html>

3

<head>

4

<title>Transformations</title>

5

<script id="vertex-shader" type="x-shader/x-vertex">

6

attribute vec4 vPosition;

7

attribute vec4 vColor;

8

uniform vec3 theta;

9

uniform vec3 dist;

10

uniform mat4 trMatrix;

11

varying vec4 fColor;

12

13

void main() {

14

// Compute the sines and cosines of theta for each of

15

// the three axes in one computation

16

vec3 angles = radians(theta);

17

vec3 c = cos(angles);

18

vec3 s = sin(angles);

19

20

// Remember: these matrices are column-major

21

mat4 rx = mat4(1.0, 0.0, 0.0, 0.0,

22

0.0, c.x, s.x, 0.0,

23

0.0, -s.x, c.x, 0.0,

24

0.0, 0.0, 0.0, 1.0);

25

26

mat4 ry = mat4(c.y, 0.0, -s.y, 0.0,

27

0.0, 1.0, 0.0, 0.0,

28

s.y, 0.0, c.y, 0.0,

29

0.0, 0.0, 0.0, 1.0);

30

31

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

32

-s.z, c.z, 0.0, 0.0,

33

0.0, 0.0, 1.0, 0.0,

34

0.0, 0.0, 0.0, 1.0);

35

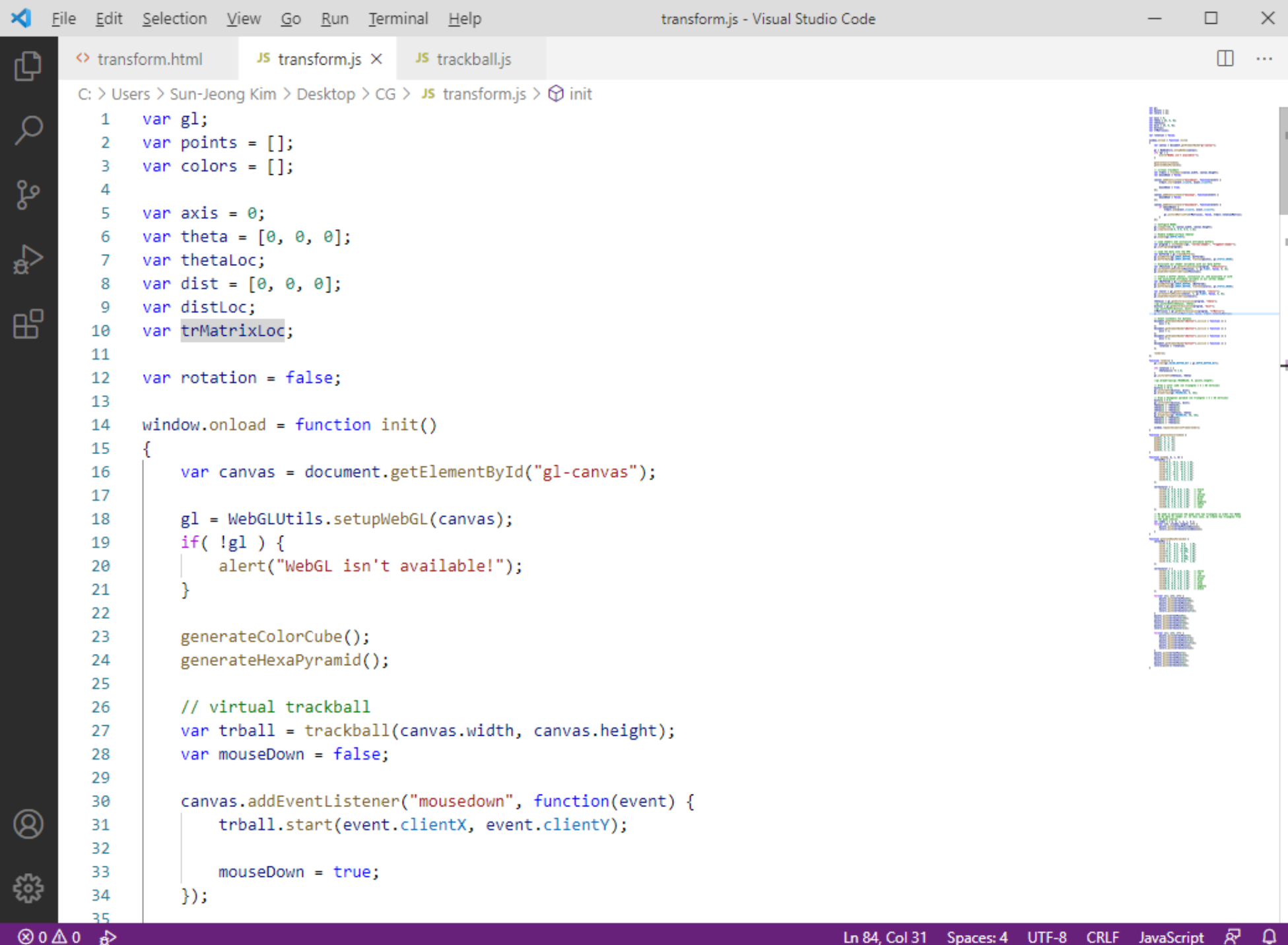



```

<> transform.html X JS transform.js JS trackball.js
C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader
31 mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
32               -s.z, c.z, 0.0, 0.0,
33               0.0, 0.0, 1.0, 0.0,
34               0.0, 0.0, 0.0, 1.0 );
35
36 mat4 sc = mat4( 0.5, 0, 0, 0,
37               0, 0.5, 0, 0,
38               0, 0, 0.5, 0,
39               0, 0, 0, 1.0);
40
41 mat4 tr = mat4( 1, 0, 0, 0,
42               0, 1, 0, 0,
43               0, 0, 1, 0,
44               dist.x, dist.y, dist.z, 1);
45
46 gl_Position = tr * rz * ry * rx * sc * trMatrix * vPosition;
47 fColor = vColor;
48 }
49 </script>
50
51 <script id="fragment-shader" type="x-shader/x-fragment">
52 precision mediump float;
53 varying vec4 fColor;
54
55 void main() {
56     gl_FragColor = fColor;
57 }
58 </script>
59
60 <script type="text/javascript" src="Common/webgl-utils.js"></script>
61 <script type="text/javascript" src="Common/initShaders.js"></script>
62 <script type="text/javascript" src="Common/MV.js"></script>
63 <script type="text/javascript" src="trackball.js"></script>
64 <script type="text/javascript" src="transform.js"></script>
65 </head>

```





```

25
26 // virtual trackball
27 var trball = trackball(canvas.width, canvas.height);
28 var mouseDown = false;
29
30 canvas.addEventListener("mousedown", function(event) {
31     trball.start(event.clientX, event.clientY);
32
33     mouseDown = true;
34 });
35
36 canvas.addEventListener("mouseup", function(event) {
37     mouseDown = false;
38 });
39
40 canvas.addEventListener("mousemove", function(event) {
41     if (mouseDown) {
42         trball.end(event.clientX, event.clientY);
43
44         gl.uniformMatrix4fv(trMatrixLoc, false, trball.rotationMatrix);
45     }
46 });
47
48 // Configure WebGL
49 gl.viewport(0, 0, canvas.width, canvas.height);
50 gl.clearColor(0.9, 0.9, 0.9, 1.0);
51
52 // Enable hidden-surface removal
53 gl.enable(gl.DEPTH_TEST);
54
55 // Load shaders and initialize attribute buffers
56 var program = initShaders(gl, "vertex-shader", "fragment-shader");
57 gl.useProgram(program);
58
59 // Load the data into the GPU

```

[illegible]

transform.js - Visual Studio Code

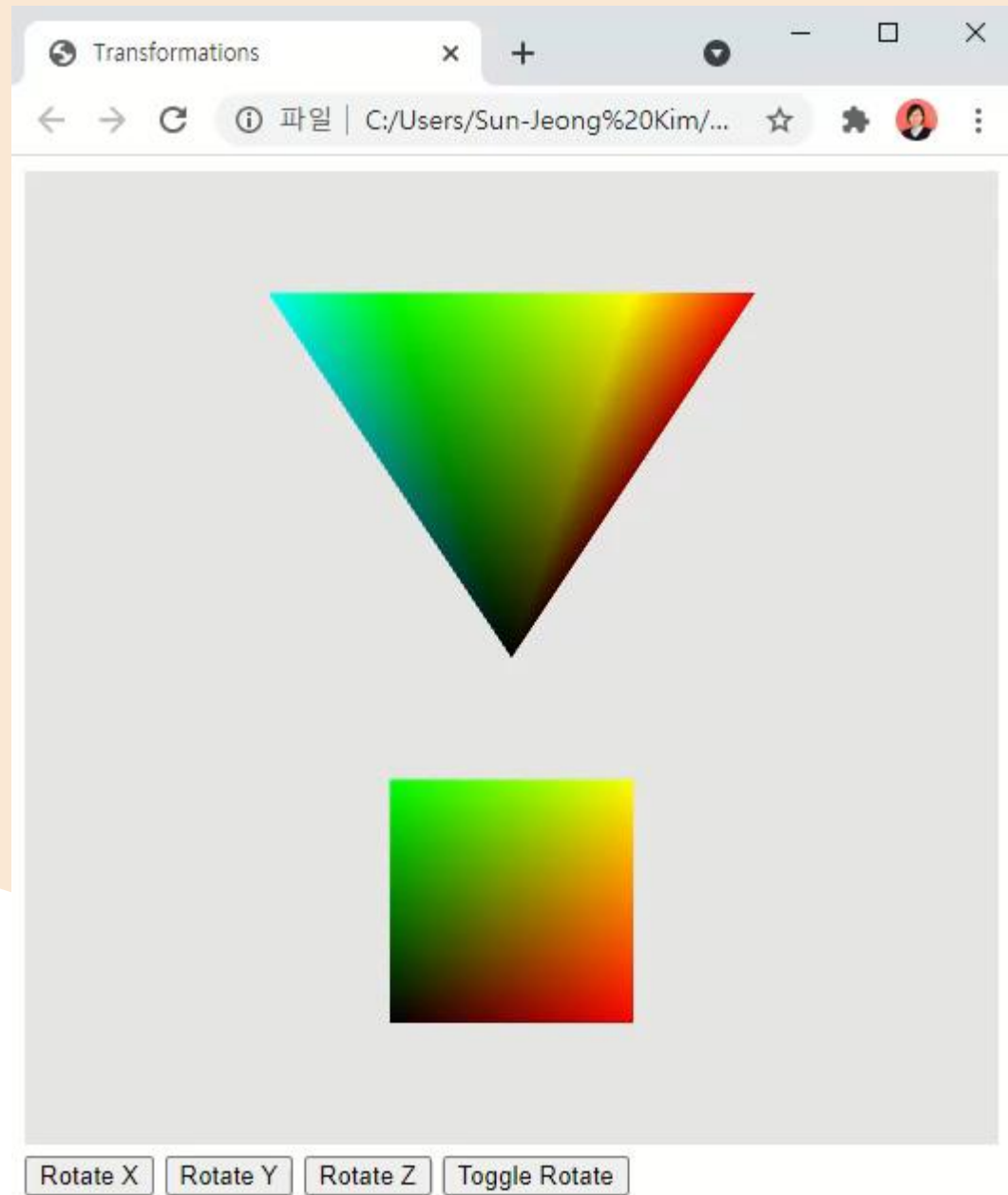
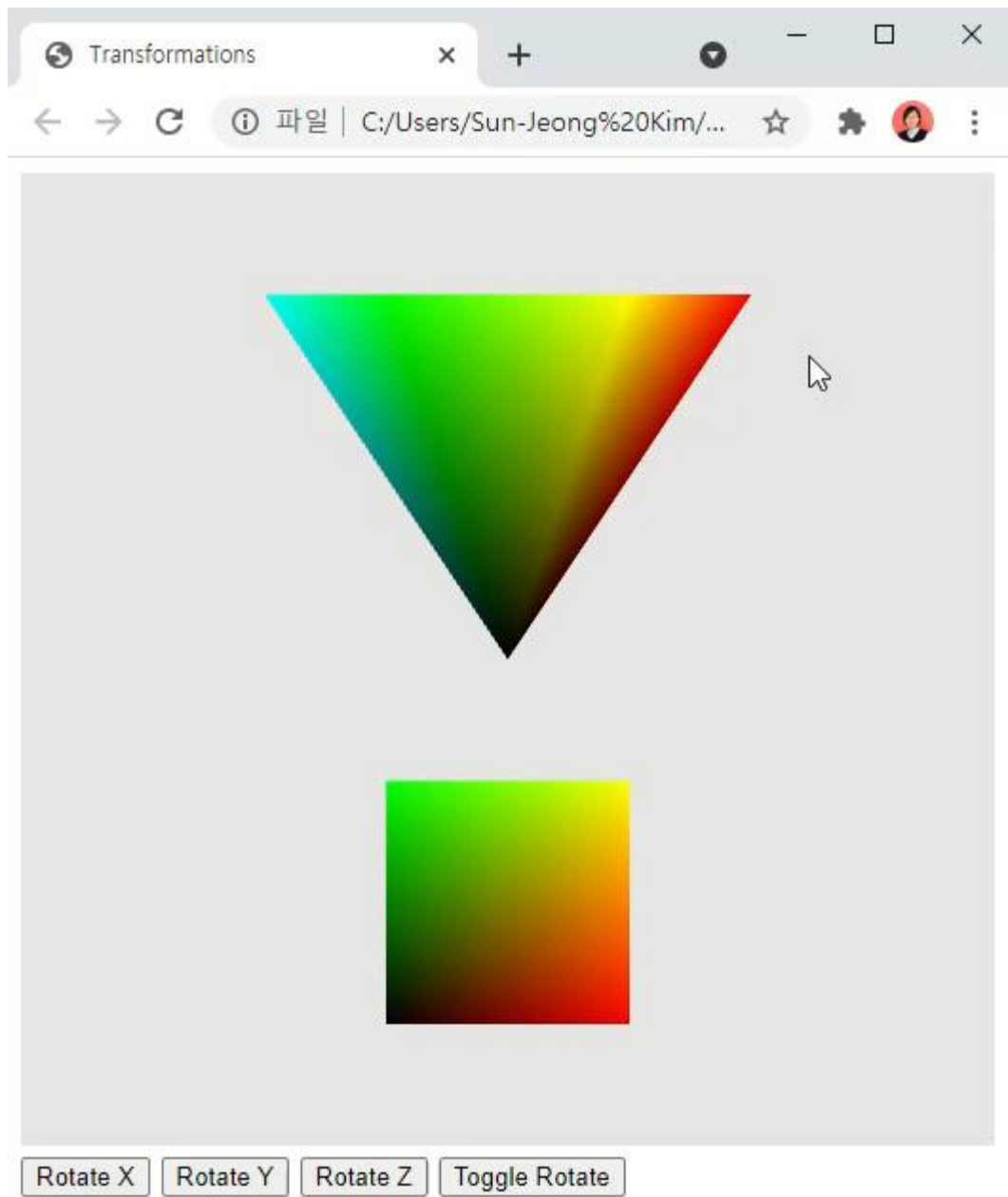
File Edit Selection View Go Run Terminal Help

transform.html JS transform.js X JS trackball.js

C: > Users > Sun-Jeong Kim > Desktop > CG > JS transform.js > init

```
78
79     thetaLoc = gl.getUniformLocation(program, "theta");
80     //gl.uniform3fv(thetaLoc, theta);
81     distLoc = gl.getUniformLocation(program, "dist");
82     //gl.uniform3fv(distLoc, dist);
83     trMatrixLoc = gl.getUniformLocation(program, "trMatrix");
84     gl.uniformMatrix4fv(trMatrixLoc, false, trball.rotationMatrix);
85
86     // Event listeners for buttons
87     document.getElementById("xButton").onclick = function () {
88         axis = 0;
89     };
90     document.getElementById("yButton").onclick = function () {
91         axis = 1;
92     };
93     document.getElementById("zButton").onclick = function () {
94         axis = 2;
95     };
96     document.getElementById("buttonT").onclick = function () {
97         rotation = !rotation;
98     };
99
100     render();
101 };
102
103 function render() {
104     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
105
106     if( rotation ) {
107         theta[axis] += 2.0;
108     }
109     gl.uniform3fv(thetaLoc, theta)
110
111     //gl.drawArrays(gl.TRIANGLES, 0, points.length);
112
```

56



```

<> transform.html X JS transform.js JS trackball.js
C: > Users > Sun-Jeong Kim > Desktop > CG > <> transform.html > html > head > script#vertex-shader
31 mat4 rz = mat4( c.z, s.z, 0.0, 0.0,
32               -s.z, c.z, 0.0, 0.0,
33               0.0, 0.0, 1.0, 0.0,
34               0.0, 0.0, 0.0, 1.0 );
35
36 mat4 sc = mat4( 0.5, 0, 0, 0,
37               0, 0.5, 0, 0,
38               0, 0, 0.5, 0,
39               0, 0, 0, 1.0);
40
41 mat4 tr = mat4( 1, 0, 0, 0,
42               0, 1, 0, 0,
43               0, 0, 1, 0,
44               dist.x, dist.y, dist.z, 1);
45
46 gl_Position = trMatrix * tr * rz * ry * rx * sc * vPosition;
47 fColor = vColor;
48 }
49 </script>
50
51 <script id="fragment-shader" type="x-shader/x-fragment">
52 precision mediump float;
53 varying vec4 fColor;
54
55 void main() {
56     gl_FragColor = fColor;
57 }
58 </script>
59
60 <script type="text/javascript" src="Common/webgl-utils.js"></script>
61 <script type="text/javascript" src="Common/initShaders.js"></script>
62 <script type="text/javascript" src="Common/MV.js"></script>
63 <script type="text/javascript" src="trackball.js"></script>
64 <script type="text/javascript" src="transform.js"></script>
65 </head>

```



연습 문제 (6)

- 4개 오브젝트를 그리시오.
 - 상하 오브젝트들은 서로 반대 방향으로 회전
 - 좌우 오브젝트들도 서로 반대 방향으로 회전

