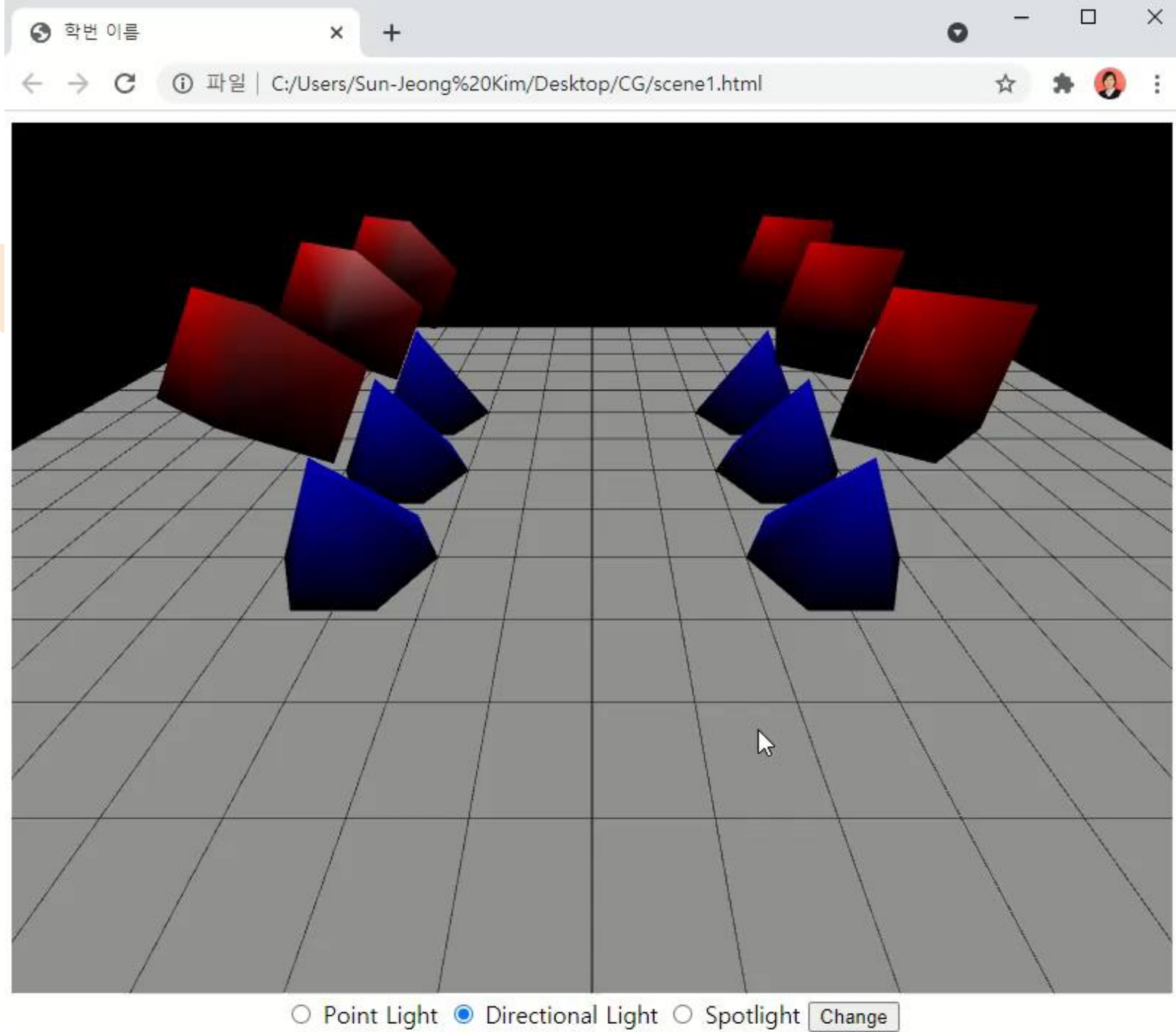


Building a Scene with Light

11TH WEEK, 2021







scene1.html × JS scene1.js



C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene1.html > html > head > script#vertex-shader

```
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 vNormal;
8          uniform mat4 modelViewMatrix;
9          uniform mat4 projectionMatrix;
10         varying vec4 fColor;
11
12         uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
13         uniform vec3 kAtten, spotDir;
14         uniform float shininess, spotExp;
15         uniform int lighting; // 0: point, 1: directional, 2: spotlight
16
17         void main() {
18             gl_Position = projectionMatrix * modelViewMatrix * vPosition;
19
20             vec3 N = normalize((modelViewMatrix * vNormal).xyz);
21             if (lighting == 0) { // point light
22                 vec3 pos = (modelViewMatrix * vPosition).xyz;
23                 vec3 light = lightPos.xyz - pos;
24                 float d = length(light);
25                 float atten = 1.0 / (kAtten[0] + kAtten[1]*d + kAtten[2]*d*d);
26                 vec3 L = normalize(light);
27                 float kd = max(dot(L, N), 0.0);
28                 vec4 diffuse = kd * diffuseProduct;
29
30                 vec3 V = normalize(-pos);
31                 vec3 H = normalize(L + V);
32                 float ks = pow(max(dot(N, H), 0.0), shininess);
33                 vec4 specular = ks * specularProduct;
```



scene1.html X JS scene1.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene1.html > html > head > script#vertex-shader

```
34
35     if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
36
37     fColor = ambientProduct + atten * (diffuse + specular);
38 }
39 else if (lighting == 1) { // directional light
40     vec3 L = normalize(lightPos.xyz);
41     float kd = max(dot(L, N), 0.0);
42     vec4 diffuse = kd * diffuseProduct;
43
44     vec3 V = normalize((modelViewMatrix * vPosition).xyz); // origin: camera position
45     vec3 H = normalize(L - V);
46     float ks = pow(max(dot(N, H), 0.0), shininess);
47     vec4 specular = ks * specularProduct;
48
49     if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
50
51     fColor = ambientProduct + diffuse + specular;
52 }
53 else { // spotlight
54     vec3 pos = (modelViewMatrix * vPosition).xyz;
55     vec3 light = lightPos.xyz - pos;
56     float d = length(light);
57     float atten = 1.0 / (kAtten[0] + kAtten[1]*d + kAtten[2]*d*d);
58     vec3 L = normalize(light);
59     float kd = max(dot(L, N), 0.0);
60     vec4 diffuse = kd * diffuseProduct;
61
62     vec3 V = normalize(-pos);
63     vec3 H = normalize(L + V);
64     float ks = pow(max(dot(N, H), 0.0), shininess);
65     vec4 specular = ks * specularProduct;
66
```



scene1.html X JS scene1.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene1.html > html > head > script#vertex-shader

```
67         if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
68
69         float spotDot = max(dot(normalize(spotDir), -L), 0.0);
70         if (spotDot > 0.0)
71             atten *= pow(spotDot, spotExp);
72         else
73             atten = 0.0;
74
75         fColor = ambientProduct + atten * (diffuse + specular);
76
77     }
78     fColor.a = 1.0;
79 }
80 </script>
81
82 <script id="fragment-shader" type="x-shader/x-fragment">
83     precision mediump float;
84     varying vec4 fColor;
85
86     void main() {
87         gl_FragColor = fColor;
88     }
89 </script>
90
91 <script type="text/javascript" src="Common/webgl-utils.js"></script>
92 <script type="text/javascript" src="Common/initShaders.js"></script>
93 <script type="text/javascript" src="Common/MV.js"></script>
94 <script type="text/javascript" src="trackball.js"></script>
95 <script type="text/javascript" src="scene1.js"></script>
96 </head>
97 <body>
98     <canvas id="gl-canvas" width="800" height="600">
99         Oops... your browser doesn't support the HTML5 canvas element!
```



scene1.html X JS scene1.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene1.html > html > head > script#vertex-shader

```
90
91     <script type="text/javascript" src="Common/webgl-utils.js"></script>
92     <script type="text/javascript" src="Common/initShaders.js"></script>
93     <script type="text/javascript" src="Common/MV.js"></script>
94     <script type="text/javascript" src="trackball.js"></script>
95     <script type="text/javascript" src="scene1.js"></script>
96 </head>
97 <body>
98     <canvas id="gl-canvas" width="800" height="600">
99         Oops... your browser doesn't support the HTML5 canvas element!
100 </canvas><br>
101     <div style="width: 800px; text-align: center;">
102         <input type="radio" id="point" name="lighting"> Point Light
103         <input type="radio" id="directional" name="lighting" checked> Directional Light
104         <input type="radio" id="spotlight" name="lighting"> Spotlight
105         <button id="change">Change</button>
106     </div>
107 </body>
108 </html>
```



scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
1  var gl;
2  var points = [];
3  var normals = [];
4
5  var theta = 0.0;
6
7  var viewMatrix;
8  var modelViewMatrixLoc, lightingLoc, diffuseProductLoc;
9
10 var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
11
12 var numVertCubeTri, numVertPyraTri, numVertGroundTri, numVertGroundLine;
13
14 var lighting = 1;  // 0: point, 1: directional, 2: spotlight
15
16 window.onload = function init()
17 {
18     var canvas = document.getElementById("gl-canvas");
19
20     gl = WebGLUtils.setupWebGL(canvas);
21     if( !gl ) {
22         alert("WebGL isn't available!");
23     }
24
25     generateCube();
26     generateHexaPyramid();
27     generateGround(10);
28
29     // virtual trackball
30     var trball = trackball(canvas.width, canvas.height);
31     var mouseDown = false;
32
33     canvas.addEventListener("mousedown", function (event) {
```

```
1 // WebGL
2 // WebGL is a JavaScript API for rendering interactive 2D and 3D
3 // computer graphics in a browser. It uses OpenGL ES 2.0 for
4 // rendering.
5 // WebGL is supported by most modern browsers.
6 // WebGL is a cross-platform API.
7 // WebGL is a low-level API.
8 // WebGL is a stateful API.
9 // WebGL is a stateless API.
10 // WebGL is a stateful API.
11 // WebGL is a stateless API.
12 // WebGL is a stateful API.
13 // WebGL is a stateless API.
14 // WebGL is a stateful API.
15 // WebGL is a stateless API.
16 // WebGL is a stateful API.
17 // WebGL is a stateless API.
18 // WebGL is a stateful API.
19 // WebGL is a stateless API.
20 // WebGL is a stateful API.
21 // WebGL is a stateless API.
22 // WebGL is a stateful API.
23 // WebGL is a stateless API.
24 // WebGL is a stateful API.
25 // WebGL is a stateless API.
26 // WebGL is a stateful API.
27 // WebGL is a stateless API.
28 // WebGL is a stateful API.
29 // WebGL is a stateless API.
30 // WebGL is a stateful API.
31 // WebGL is a stateless API.
32 // WebGL is a stateful API.
33 // WebGL is a stateless API.
34 // WebGL is a stateful API.
35 // WebGL is a stateless API.
36 // WebGL is a stateful API.
37 // WebGL is a stateless API.
38 // WebGL is a stateful API.
39 // WebGL is a stateless API.
40 // WebGL is a stateful API.
41 // WebGL is a stateless API.
42 // WebGL is a stateful API.
43 // WebGL is a stateless API.
44 // WebGL is a stateful API.
45 // WebGL is a stateless API.
46 // WebGL is a stateful API.
47 // WebGL is a stateless API.
48 // WebGL is a stateful API.
49 // WebGL is a stateless API.
50 // WebGL is a stateful API.
51 // WebGL is a stateless API.
52 // WebGL is a stateful API.
53 // WebGL is a stateless API.
54 // WebGL is a stateful API.
55 // WebGL is a stateless API.
56 // WebGL is a stateful API.
57 // WebGL is a stateless API.
58 // WebGL is a stateful API.
59 // WebGL is a stateless API.
60 // WebGL is a stateful API.
61 // WebGL is a stateless API.
62 // WebGL is a stateful API.
63 // WebGL is a stateless API.
64 // WebGL is a stateful API.
65 // WebGL is a stateless API.
66 // WebGL is a stateful API.
67 // WebGL is a stateless API.
68 // WebGL is a stateful API.
69 // WebGL is a stateless API.
70 // WebGL is a stateful API.
71 // WebGL is a stateless API.
72 // WebGL is a stateful API.
73 // WebGL is a stateless API.
74 // WebGL is a stateful API.
75 // WebGL is a stateless API.
76 // WebGL is a stateful API.
77 // WebGL is a stateless API.
78 // WebGL is a stateful API.
79 // WebGL is a stateless API.
80 // WebGL is a stateful API.
81 // WebGL is a stateless API.
82 // WebGL is a stateful API.
83 // WebGL is a stateless API.
84 // WebGL is a stateful API.
85 // WebGL is a stateless API.
86 // WebGL is a stateful API.
87 // WebGL is a stateless API.
88 // WebGL is a stateful API.
89 // WebGL is a stateless API.
90 // WebGL is a stateful API.
91 // WebGL is a stateless API.
92 // WebGL is a stateful API.
93 // WebGL is a stateless API.
94 // WebGL is a stateful API.
95 // WebGL is a stateless API.
96 // WebGL is a stateful API.
97 // WebGL is a stateless API.
98 // WebGL is a stateful API.
99 // WebGL is a stateless API.
100 // WebGL is a stateful API.
```

scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
33 canvas.addEventListener("mousedown", function (event) {
34     trball.start(event.clientX, event.clientY);
35
36     mouseDown = true;
37 });
38
39 canvas.addEventListener("mouseup", function (event) {
40     mouseDown = false;
41 });
42
43 canvas.addEventListener("mousemove", function (event) {
44     if (mouseDown) {
45         trball.end(event.clientX, event.clientY);
46
47         trballMatrix = mat4(trball.rotationMatrix);
48     }
49 });
50
51 // Configure WebGL
52 gl.viewport(0, 0, canvas.width, canvas.height);
53 gl.clearColor(0.0, 0.0, 0.0, 1.0);
54
55 // Enable hidden-surface removal
56 gl.enable(gl.DEPTH_TEST);
57
58 gl.enable(gl.POLYGON_OFFSET_FILL);
59 gl.polygonOffset(0.01, 1);
60
61 // Load shaders and initialize attribute buffers
62 var program = initShaders(gl, "vertex-shader", "fragment-shader");
63 gl.useProgram(program);
64
65 // Load the data into the GPU
```


Figure 1 displays a series of genomic tracks for a region on chromosome 1, specifically focusing on the SLC11A1, SLC11A2, and SLC11A3 genes. The tracks are organized into 18 panels (a-r) and include the following data:

- Panel (a):** RefSeq gene models for SLC11A1, SLC11A2, and SLC11A3.
- Panel (b):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (c):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (d):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (e):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (f):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (g):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (h):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (i):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (j):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (k):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (l):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (m):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (n):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (o):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (p):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (q):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.
- Panel (r):** UCSC Genome Browser tracks for SLC11A1, SLC11A2, and SLC11A3.



scene1.html JS scene1.js X



C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
98  */
99  // 3D perspective viewing
100  var aspect = canvas.width / canvas.height;
101  projectionMatrix = perspective(90, aspect, 0.1, 1000);
102  projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
103  gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
104
105  lightingLoc = gl.getUniformLocation(program, "lighting");
106  gl.uniform1i(lightingLoc, lighting);
107
108  // Event listeners for buttons
109  document.getElementById("change").onclick = function () {
110      if (document.getElementById("point").checked)
111          lighting = 0;
112      else if (document.getElementById("directional").checked)
113          lighting = 1;
114      else
115          lighting = 2;
116      gl.uniform1i(lightingLoc, lighting);
117  };
118
119  setLighting(program);
120
121  render();
122
123  };
124
125  function setLighting(program) {
126      var lightPos = [0.0, 1.0, 0.0, 0.0];
127      var lightAmbient = [0.2, 0.2, 0.2, 1.0];
128      var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
129      var lightSpecular = [1.0, 1.0, 1.0, 1.0];
130  }
```



scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
131 var matAmbient = [1.0, 1.0, 1.0, 1.0];
132 var matDiffuse = [1.0, 1.0, 1.0, 1.0];
133 var matSpecular = [1.0, 1.0, 1.0, 1.0];
134
135 var ambientProduct = mult(lightAmbient, matAmbient);
136 var diffuseProduct = mult(lightDiffuse, matDiffuse);
137 var specularProduct = mult(lightSpecular, matSpecular);
138
139 gl.uniform4fv(gl.getUniformLocation(program, "lightPos"), lightPos);
140 gl.uniform4fv(gl.getUniformLocation(program, "ambientProduct"), ambientProduct);
141 diffuseProductLoc = gl.getUniformLocation(program, "diffuseProduct");
142 gl.uniform4fv(diffuseProductLoc, diffuseProduct);
143 gl.uniform4fv(gl.getUniformLocation(program, "specularProduct"), specularProduct);
144
145 gl.uniform3f(gl.getUniformLocation(program, "kAtten"), 1.0, 0.1, 0.1);
146 gl.uniform3f(gl.getUniformLocation(program, "spotDir"), 0.0, -1.0, -1.0);
147 gl.uniform1f(gl.getUniformLocation(program, "spotExp"), 5.0);
148 gl.uniform1f(gl.getUniformLocation(program, "shininess"), 100.0);
149 }
150
151 function render() {
152     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
153
154     theta += 2.0;
155
156     for (var z=-5; z<0; z+=2) {
157         // draw a cube
158         gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0); // red
159
160         var rMatrix = mult(rotateY(theta), rotateZ(45));
161         var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
162         modelMatrix = mult(trballMatrix, modelMatrix);
163         modelViewMatrix = mult(viewMatrix, modelMatrix);
```

scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
163     modelViewMatrix = mult(viewMatrix, modelMatrix);
164     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
165     gl.drawArrays(gl.TRIANGLES, 0, numVertCubeTri);
166
167     modelMatrix = mult(translate(3, 1.3, z), rMatrix);
168     modelMatrix = mult(trballMatrix, modelMatrix);
169     modelViewMatrix = mult(viewMatrix, modelMatrix);
170     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
171     gl.drawArrays(gl.TRIANGLES, 0, numVertCubeTri);
172
173     // draw a hexa-pyramid
174     gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);    // blue
175
176     modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
177     modelMatrix = mult(trballMatrix, modelMatrix);
178     modelViewMatrix = mult(viewMatrix, modelMatrix);
179     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
180     gl.drawArrays(gl.TRIANGLES, numVertCubeTri, numVertPyraTri);
181
182     modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
183     modelMatrix = mult(trballMatrix, modelMatrix);
184     modelViewMatrix = mult(viewMatrix, modelMatrix);
185     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
186     gl.drawArrays(gl.TRIANGLES, numVertCubeTri, numVertPyraTri);
187 }
188
189 // draw the ground
190 gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);    // gray
191
192 modelViewMatrix = mult(viewMatrix, trballMatrix);
193 gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
194 //gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(viewMatrix));
195 gl.drawArrays(gl.TRIANGLES, numVertCubeTri+numVertPyraTri, numVertGroundTri);
```



scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
195     gl.drawArrays(gl.TRIANGLES, numVertCubeTri+numVertPyrTri, numVertGroundTri);
196     gl.drawArrays(gl.LINES, numVertCubeTri+numVertPyrTri+numVertGroundTri, numVertGroundLine);
197
198     requestAnimationFrame(render);
199 }
200
201 function generateCube() {
202     numVertCubeTri = 0;
203     quad(1, 0, 3, 2);
204     quad(2, 3, 7, 6);
205     quad(3, 0, 4, 7);
206     quad(4, 5, 6, 7);
207     quad(5, 4, 0, 1);
208     quad(6, 5, 1, 2);
209 }
210
211 function quad(a, b, c, d) {
212     const vertexPos = [
213         vec4(-0.5, -0.5, -0.5, 1.0),
214         vec4( 0.5, -0.5, -0.5, 1.0),
215         vec4( 0.5,  0.5, -0.5, 1.0),
216         vec4(-0.5,  0.5, -0.5, 1.0),
217         vec4(-0.5, -0.5,  0.5, 1.0),
218         vec4( 0.5, -0.5,  0.5, 1.0),
219         vec4( 0.5,  0.5,  0.5, 1.0),
220         vec4(-0.5,  0.5,  0.5, 1.0)
221     ];
222
223     const vertexNormal = [
224         vec4(-0.57735, -0.57735, -0.57735, 0.0),
225         vec4( 0.57735, -0.57735, -0.57735, 0.0),
226         vec4( 0.57735,  0.57735, -0.57735, 0.0),
227         vec4(-0.57735,  0.57735, -0.57735, 0.0),
```

```
195     gl.drawArrays(gl.TRIANGLES, numVertCubeTri+numVertPyrTri, numVertGroundTri);
196     gl.drawArrays(gl.LINES, numVertCubeTri+numVertPyrTri+numVertGroundTri, numVertGroundLine);
197
198     requestAnimationFrame(render);
199 }
200
201 function generateCube() {
202     numVertCubeTri = 0;
203     quad(1, 0, 3, 2);
204     quad(2, 3, 7, 6);
205     quad(3, 0, 4, 7);
206     quad(4, 5, 6, 7);
207     quad(5, 4, 0, 1);
208     quad(6, 5, 1, 2);
209 }
210
211 function quad(a, b, c, d) {
212     const vertexPos = [
213         vec4(-0.5, -0.5, -0.5, 1.0),
214         vec4( 0.5, -0.5, -0.5, 1.0),
215         vec4( 0.5,  0.5, -0.5, 1.0),
216         vec4(-0.5,  0.5, -0.5, 1.0),
217         vec4(-0.5, -0.5,  0.5, 1.0),
218         vec4( 0.5, -0.5,  0.5, 1.0),
219         vec4( 0.5,  0.5,  0.5, 1.0),
220         vec4(-0.5,  0.5,  0.5, 1.0)
221     ];
222
223     const vertexNormal = [
224         vec4(-0.57735, -0.57735, -0.57735, 0.0),
225         vec4( 0.57735, -0.57735, -0.57735, 0.0),
226         vec4( 0.57735,  0.57735, -0.57735, 0.0),
227         vec4(-0.57735,  0.57735, -0.57735, 0.0),
```


scene1.html JS scene1.js X

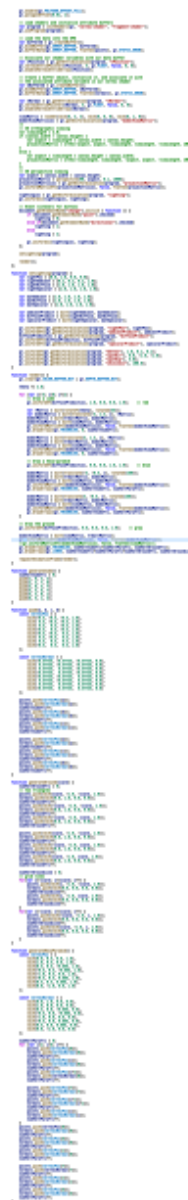
C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
228     vec4(-0.57735, -0.57735, 0.57735, 0.0),
229     vec4( 0.57735, -0.57735, 0.57735, 0.0),
230     vec4( 0.57735, 0.57735, 0.57735, 0.0),
231     vec4(-0.57735, 0.57735, 0.57735, 0.0)
232 ];
233
234 points.push(vertexPos[a]);
235 normals.push(vertexNormal[a]);
236 numVertCubeTri++;
237 points.push(vertexPos[b]);
238 normals.push(vertexNormal[b]);
239 numVertCubeTri++;
240 points.push(vertexPos[c]);
241 normals.push(vertexNormal[c]);
242 numVertCubeTri++;
243
244 points.push(vertexPos[a]);
245 normals.push(vertexNormal[a]);
246 numVertCubeTri++;
247 points.push(vertexPos[c]);
248 normals.push(vertexNormal[c]);
249 numVertCubeTri++;
250 points.push(vertexPos[d]);
251 normals.push(vertexNormal[d]);
252 numVertCubeTri++;
253 }
254
255 function generateGround(scale) {
256     numVertGroundTri = 0;
257     // two triangles
258     points.push(vec4(scale, -1.0, -scale, 1.0));
259     normals.push(vec4(0.0, 1.0, 0.0, 0.0));
260     numVertGroundTri++;
```


scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

```
260     numVertGroundTri++;
261     points.push(vec4(-scale, -1.0, -scale, 1.0));
262     normals.push(vec4(0.0, 1.0, 0.0, 0.0));
263     numVertGroundTri++;
264     points.push(vec4(-scale, -1.0, scale, 1.0));
265     normals.push(vec4(0.0, 1.0, 0.0, 0.0));
266     numVertGroundTri++;
267
268     points.push(vec4(scale, -1.0, -scale, 1.0));
269     normals.push(vec4(0.0, 1.0, 0.0, 0.0));
270     numVertGroundTri++;
271     points.push(vec4(scale, -1.0, scale, 1.0));
272     normals.push(vec4(0.0, 1.0, 0.0, 0.0));
273     numVertGroundTri++;
274     points.push(vec4(scale, -1.0, scale, 1.0));
275     normals.push(vec4(0.0, 1.0, 0.0, 0.0));
276     numVertGroundTri++;
277
278     numVertGroundLine = 0;
279     // grid lines
280     for(var x=-scale; x<=scale; x++) {
281         points.push(vec4(x, -1.0, -scale, 1.0));
282         normals.push(vec4(0.0, 0.0, 0.0, 0.0));
283         numVertGroundLine++;
284         points.push(vec4(x, -1.0, scale, 1.0));
285         normals.push(vec4(0.0, 0.0, 0.0, 0.0));
286         numVertGroundLine++;
287     }
288     for(var z=-scale; z<=scale; z++) {
289         points.push(vec4(-scale, -1.0, z, 1.0));
290         normals.push(vec4(0.0, 0.0, 0.0, 0.0));
291         numVertGroundLine++;
292         points.push(vec4(scale, -1.0, z, 1.0));
```




```

        points.push(vec4(scale, -1.0, z, 1.0));
        normals.push(vec4(0.0, 0.0, 0.0, 0.0));
        numVertGroundLine++;
    }
}

function generateHexaPyramid() {
    const vertexPos = [
        vec4(0.0, 0.5, 0.0, 1.0),
        vec4(1.0, 0.5, 0.0, 1.0),
        vec4(0.5, 0.5, -0.866, 1.0),
        vec4(-0.5, 0.5, -0.866, 1.0),
        vec4(-1.0, 0.5, 0.0, 1.0),
        vec4(-0.5, 0.5, 0.866, 1.0),
        vec4(0.5, 0.5, 0.866, 1.0),
        vec4(0.0, -1.0, 0.0, 1.0)
    ];

    const vertexNormal = [
        vec4(0.0, 1.0, 0.0, 0.0),
        vec4(1.0, 0.0, 0.0, 0.0),
        vec4(0.5, 0.0, -0.866, 0.0),
        vec4(-0.5, 0.0, -0.866, 0.0),
        vec4(-1.0, 0.0, 0.0, 0.0),
        vec4(-0.5, 0.0, 0.866, 0.0),
        vec4(0.5, 0.0, 0.866, 0.0),
        vec4(0.0, -1.0, 0.0, 0.0)
    ];

    numVertPyrTri = 0;
    for (var i=1; i<6; i++) {
        points.push(vertexPos[i]);
        normals.push(vertexNormal[i]);
    }
}

```

scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

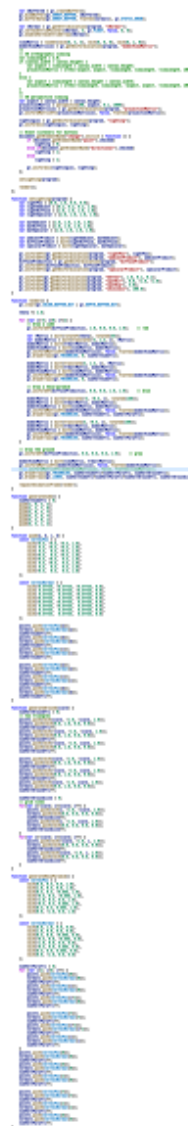
```
324     normals.push(vertexNormal[0]);
325     numVertPyraTri++;
326     points.push(vertexPos[i]);
327     normals.push(vertexNormal[0]);
328     numVertPyraTri++;
329     points.push(vertexPos[i+1]);
330     normals.push(vertexNormal[0]);
331     numVertPyraTri++;
332
333     points.push(vertexPos[7]);
334     normals.push(vertexNormal[7]);
335     numVertPyraTri++;
336     points.push(vertexPos[i+1]);
337     normals.push(vertexNormal[i+1]);
338     numVertPyraTri++;
339     points.push(vertexPos[i]);
340     normals.push(vertexNormal[i]);
341     numVertPyraTri++;
342 }
343 points.push(vertexPos[0]);
344 normals.push(vertexNormal[0]);
345 numVertPyraTri++;
346 points.push(vertexPos[6]);
347 normals.push(vertexNormal[0]);
348 numVertPyraTri++;
349 points.push(vertexPos[1]);
350 normals.push(vertexNormal[0]);
351 numVertPyraTri++;
352
353 points.push(vertexPos[7]);
354 normals.push(vertexNormal[7]);
355 numVertPyraTri++;
356 points.push(vertexPos[1]);
```



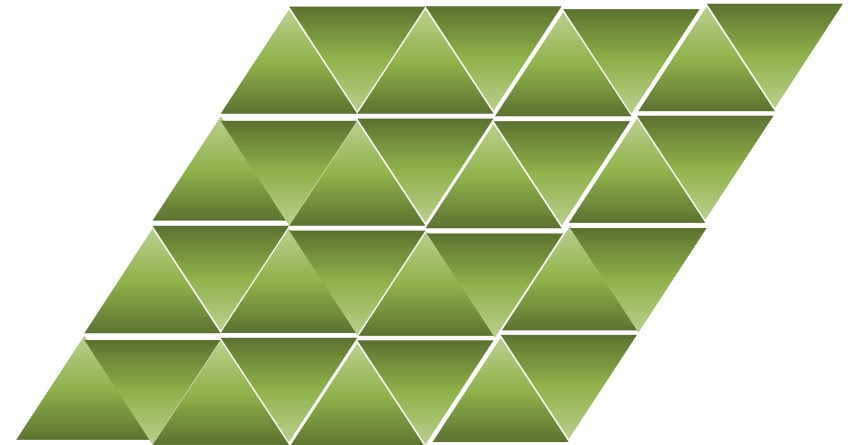
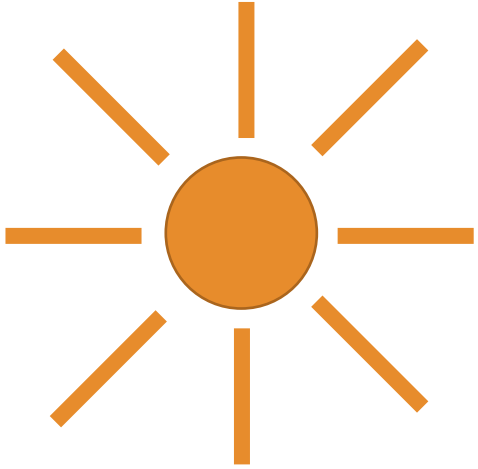
scene1.html JS scene1.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > render

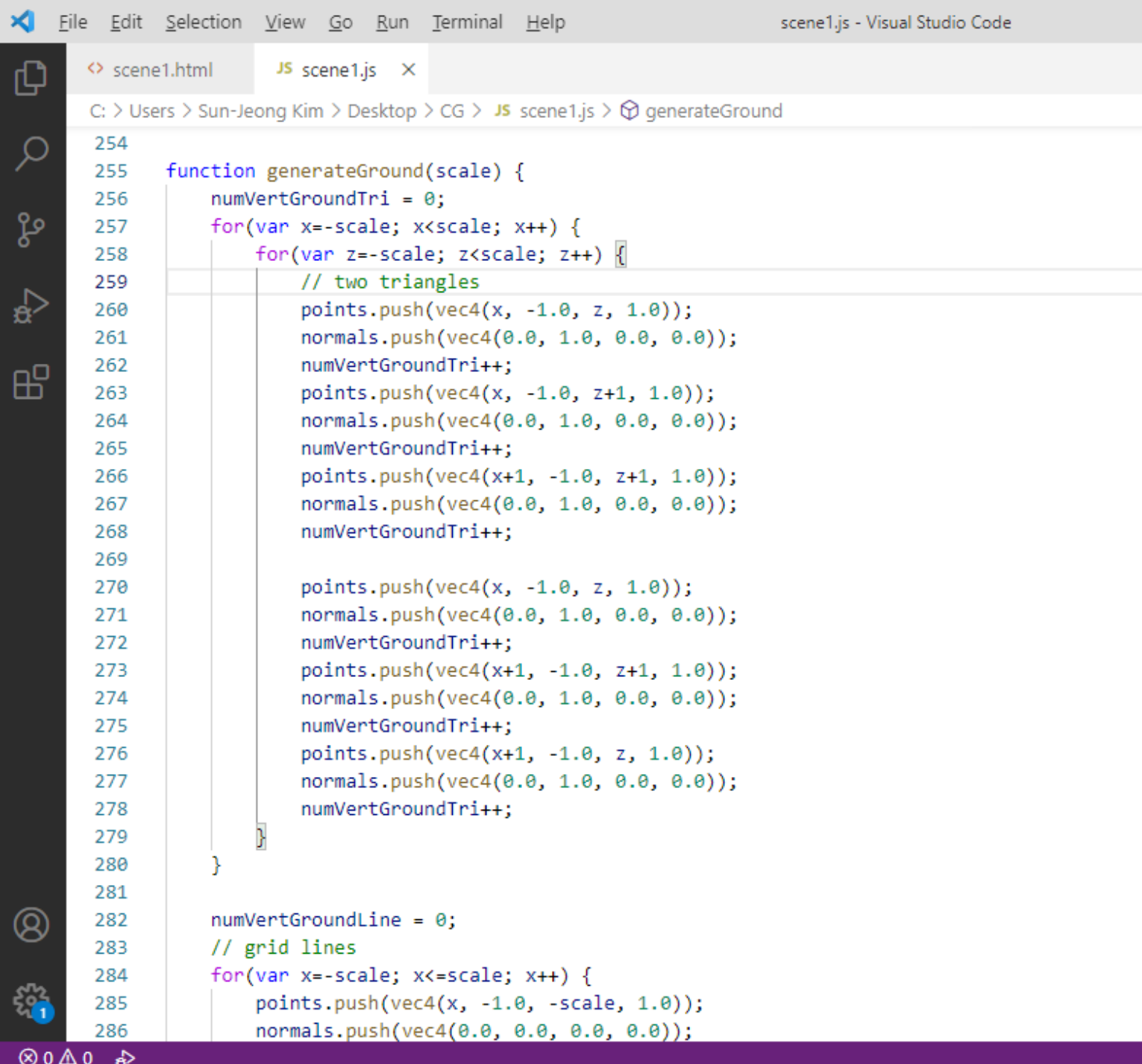
```
343     points.push(vertexPos[0]);
344     normals.push(vertexNormal[0]);
345     numVertPyraTri++;
346     points.push(vertexPos[6]);
347     normals.push(vertexNormal[0]);
348     numVertPyraTri++;
349     points.push(vertexPos[1]);
350     normals.push(vertexNormal[0]);
351     numVertPyraTri++;
352
353     points.push(vertexPos[7]);
354     normals.push(vertexNormal[7]);
355     numVertPyraTri++;
356     points.push(vertexPos[1]);
357     normals.push(vertexNormal[1]);
358     numVertPyraTri++;
359     points.push(vertexPos[6]);
360     normals.push(vertexNormal[6]);
361     numVertPyraTri++;
362 }
363
```



What's the Difference?



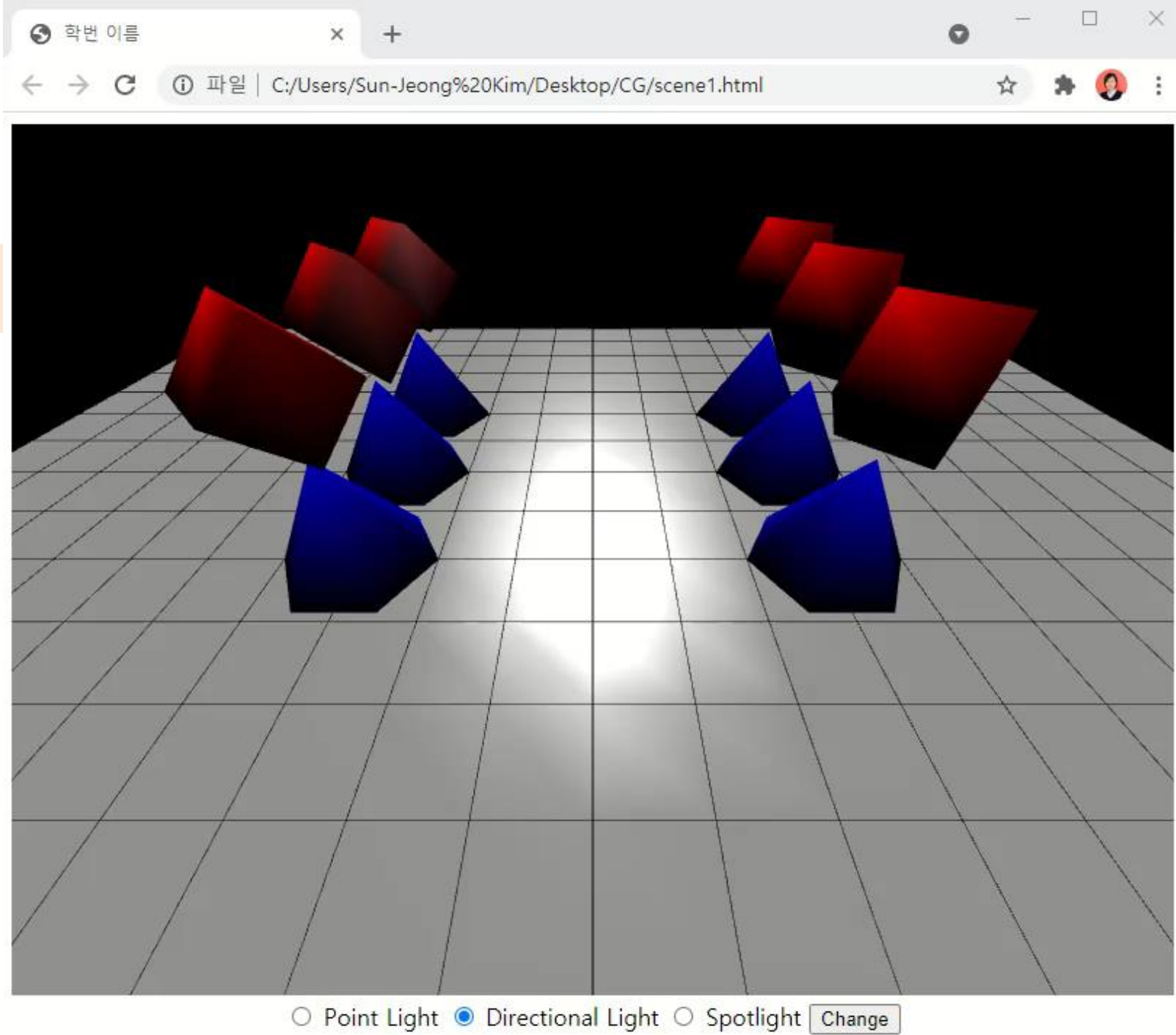
Ground



```
File Edit Selection View Go Run Terminal Help
scene1.js - Visual Studio Code

<> scene1.html JS scene1.js X
C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene1.js > generateGround

254
255 function generateGround(scale) {
256     numVertGroundTri = 0;
257     for(var x=-scale; x<scale; x++) {
258         for(var z=-scale; z<scale; z++) {
259             // two triangles
260             points.push(vec4(x, -1.0, z, 1.0));
261             normals.push(vec4(0.0, 1.0, 0.0, 0.0));
262             numVertGroundTri++;
263             points.push(vec4(x, -1.0, z+1, 1.0));
264             normals.push(vec4(0.0, 1.0, 0.0, 0.0));
265             numVertGroundTri++;
266             points.push(vec4(x+1, -1.0, z+1, 1.0));
267             normals.push(vec4(0.0, 1.0, 0.0, 0.0));
268             numVertGroundTri++;
269
270             points.push(vec4(x, -1.0, z, 1.0));
271             normals.push(vec4(0.0, 1.0, 0.0, 0.0));
272             numVertGroundTri++;
273             points.push(vec4(x+1, -1.0, z+1, 1.0));
274             normals.push(vec4(0.0, 1.0, 0.0, 0.0));
275             numVertGroundTri++;
276             points.push(vec4(x+1, -1.0, z, 1.0));
277             normals.push(vec4(0.0, 1.0, 0.0, 0.0));
278             numVertGroundTri++;
279         }
280     }
281
282     numVertGroundLine = 0;
283     // grid lines
284     for(var x=-scale; x<=scale; x++) {
285         points.push(vec4(x, -1.0, -scale, 1.0));
286         normals.push(vec4(0.0, 0.0, 0.0, 0.0));
```

연습 문제 (1)

- Directional 조명을 선택할 때, scene1.js에 있는 `setLighting()` 함수에서,
 - `lightPos`의 값을 변경하면, 결과가 어떻게 바뀌는지 설명하시오.
 - 예) `var lightPos = vec4(1.0, 0.0, 0.0, 0.0);`
 - `lightAmbient`, `lightDiffuse`, `lightSpecular`의 값을 변경하면, 결과가 어떻게 바뀌는지 설명하시오.
 - 예) `var lightAmbient = vec4(1.0, 0.0, 0.0, 1.0);`
`var lightDiffuse = vec4(0.0, 1.0, 0.0, 1.0);`
`var lightSpecular = vec4(0.0, 0.0, 1.0, 1.0);`

연습 문제 (2)

- Point 조명을 선택 후
light.js에 있는 `setLighting()` 함수에서,
 - `lightPos`의 값을 변경하면, 결과가 어떻게 바뀌는지 설명하시오.
 - 예) `var lightPos = vec4(1.0, 0.0, 0.0, 1.0);`
 - `kAtten`의 값을 변경하면, 결과가 어떻게 바뀌는지 설명하시오.
 - 예) `gl.uniform3f(attenLoc, 1.0, 0.0, 0.0);`

연습 문제 (3)

- Spotlight 선택 후
light.js에 있는 setLighting() 함수에서,
 - spotDir의 값을 변경하면, 결과가 어떻게 바뀌는지 설명하시오.
 - 예) `gl.uniform3f(spotDirLoc, 0.0, 0.0, -1.0);`
 - spotExp의 값을 변경하면, 결과가 어떻게 바뀌는지 설명하시오.
 - 예) `gl.uniform1f(spotExpLoc, 10.0);`