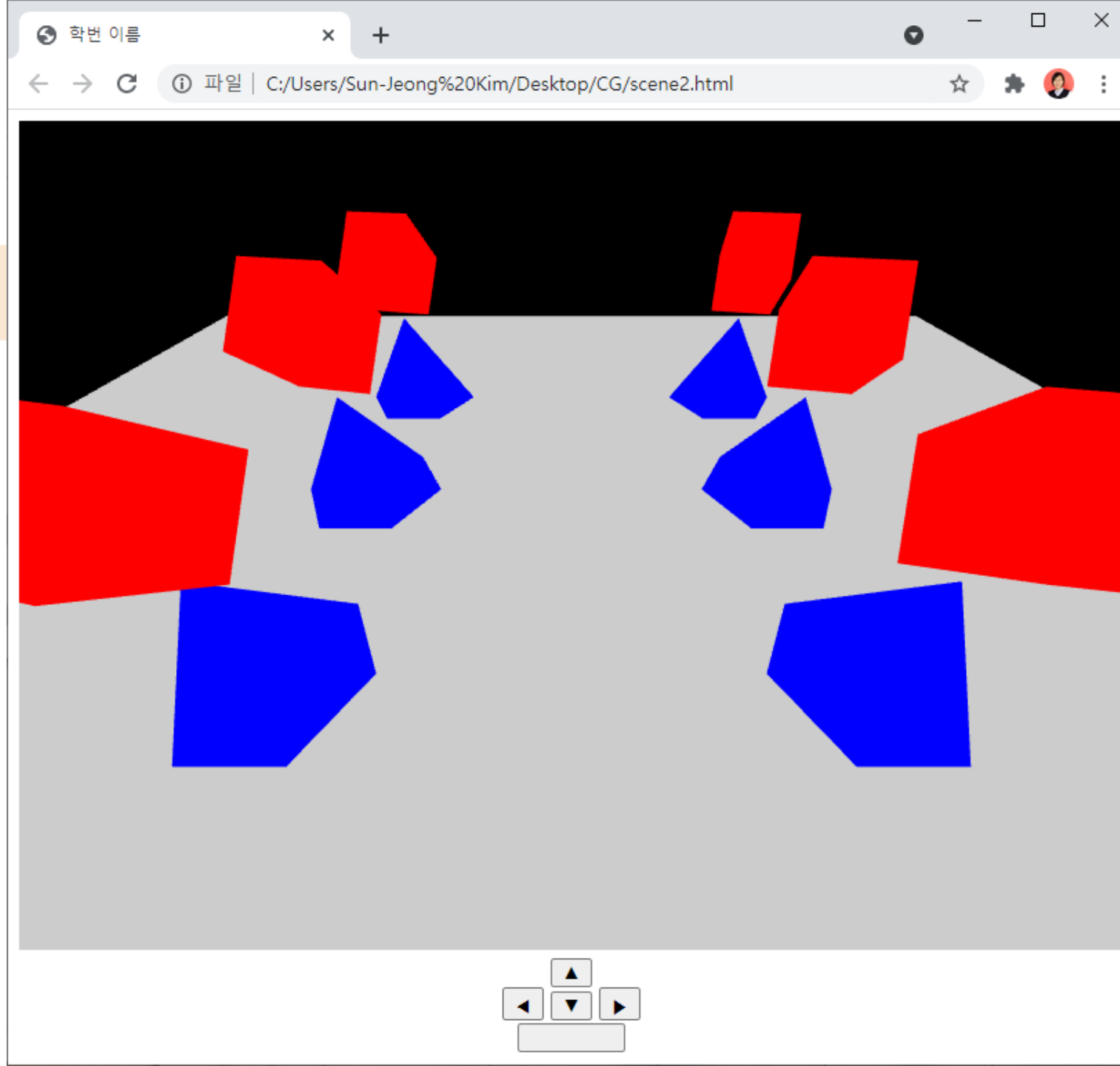


Building a Scene with Texture Mapping

14TH WEEK, 2021





Multiple Shaders



The screenshot shows the Visual Studio Code editor interface. The top bar indicates the active file is 'scene2.html - Visual Studio Code'. The editor has two tabs open: 'scene2.html' and 'scene2.js'. The 'scene2.html' tab is active, showing an HTML document with two embedded shaders. The breadcrumb navigation at the top of the editor area shows the path: 'C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.html > html > head > title'. The HTML code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>학번 이름</title>
5   <script id="colorVS" type="x-shader/x-vertex">
6     attribute vec4 vPosition;
7     uniform mat4 modelViewMatrix;
8     uniform mat4 projectionMatrix;
9
10    void main() {
11      gl_Position = projectionMatrix * modelViewMatrix * vPosition;
12    }
13  </script>
14
15  <script id="colorFS" type="x-shader/x-fragment">
16    precision mediump float;
17
18    uniform vec4 uColor;
19
20    void main() {
21      gl_FragColor = uColor;
22    }
23  </script>
```

On the right side of the editor, a small preview window displays the rendered output of the shaders. It shows a solid blue square, which is the result of the fragment shader's uniform color 'uColor'.

scene2.html x JS scene2.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene2.html > html > head > title

```
24
25 <script id="phongVS" type="x-shader/x-vertex">
26     attribute vec4 vPosition;
27     attribute vec4 vNormal;
28     uniform mat4 modelViewMatrix;
29     uniform mat4 projectionMatrix;
30
31     varying vec3 fNormal, fPosition;
32
33     void main() {
34         gl_Position = projectionMatrix * modelViewMatrix * vPosition;
35
36         fNormal = (modelViewMatrix * vNormal).xyz;
37         fPosition = (modelViewMatrix * vPosition).xyz;
38     }
39 </script>
40
41 <script id="phongFS" type="x-shader/x-fragment">
42     precision mediump float;
43
44     varying vec3 fNormal, fPosition;
45
46     uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
47     uniform float shininess;
48
49     void main() {
50         vec3 N = normalize(fNormal);
51         vec3 L = normalize(lightPos.xyz);
52         float kd = max(dot(L, N), 0.0);
53         vec4 diffuse = kd * diffuseProduct;
54
55         vec3 V = normalize(fPosition); // origin: camera position
56         vec3 H = normalize(L - V);
```



scene2.html X JS scene2.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene2.html > html > head > title

```
57     float ks = pow(max(dot(N, H), 0.0), shininess);
58     vec4 specular = ks * specularProduct;
59
60     if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
61
62     gl_FragColor = ambientProduct + diffuse + specular;
63     gl_FragColor.a = 1.0;
64 }
65 </script>
66
67 <script id="texMapVS" type="x-shader/x-vertex">
68     attribute vec4 vPosition;
69     attribute vec4 vNormal;
70     attribute vec2 vTexCoord;
71
72     uniform mat4 modelViewMatrix;
73     uniform mat4 projectionMatrix;
74
75     varying vec3 fNormal, fPosition;
76     varying vec2 fTexCoord;
77
78     void main() {
79         gl_Position = projectionMatrix * modelViewMatrix * vPosition;
80
81         fNormal = (modelViewMatrix * vNormal).xyz;
82         fPosition = (modelViewMatrix * vPosition).xyz;
83
84         fTexCoord = vTexCoord;
85     }
86 </script>
87
88 <script id="texMapFS" type="x-shader/x-fragment">
89     precision mediump float;
```



scene2.html × JS scene2.js



C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene2.html > html > head > title

```
90
91     varying vec3 fNormal, fPosition;
92     varying vec2 fTexCoord;
93
94     uniform sampler2D texture;
95     uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
96     uniform float shininess;
97
98     void main() {
99         vec3 N = normalize(fNormal);
100         vec3 L = normalize(lightPos.xyz);
101         float kd = max(dot(L, N), 0.0);
102         vec4 diffuse = kd * diffuseProduct;
103
104         vec3 V = normalize(fPosition); // origin: camera position
105         vec3 H = normalize(L - V);
106         float ks = pow(max(dot(N, H), 0.0), shininess);
107         vec4 specular = ks * specularProduct;
108
109         if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
110
111         gl_FragColor = (ambientProduct + diffuse + specular) * texture2D(texture, fTexCoord);
112         gl_FragColor.a = 1.0;
113     }
114 </script>
115
116 <script type="text/javascript" src="Common/webgl-utils.js"></script>
117 <script type="text/javascript" src="Common/initShaders.js"></script>
118 <script type="text/javascript" src="Common/MV.js"></script>
119 <script type="text/javascript" src="trackball.js"></script>
120 <script type="text/javascript" src="scene2.js"></script>
121 </head>
122 <body>
```



scene2.html x JS scene2.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene2.html > html > head > title

```
123 <canvas id="gl-canvas" width="800" height="600">
124 |   Oops... your browser doesn't support the HTML5 canvas element!
125 </canvas><br>
126 <div style="width:800px; text-align:center;">
127 |   <button id="up">▲</button><br>
128 |   <button id="left">◀</button>
129 |   <button id="down">▼</button>
130 |   <button id="right">▶</button><br>
131 |   <button id="space"> &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; </button>
132 </div>
133 <div>
134 |   
135 |   
136 |   
137 </div>
138 </body>
139 </html>
```



scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > ...

```
1  var gl;
2  var points = [];
3  var normals = [];
4  var texCoords = [];
5
6  var program0, program1, program2;    // [program1] Phong shading, [program2] Texture Mapping
7  var modelViewMatrixLoc0, modelViewMatrixLoc1, modelViewMatrixLoc2;
8
9  var eye = vec3(0, 3, 3);
10 var at = vec3(0, 0, 0);
11 const up = vec3(0, 1, 0);
12 var cameraVec = vec3(0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)
13
14 var theta = 0;
15 var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
16 var vertCubeStart, vertCubeEnd, vertHexaStart, vertHexaEnd, vertGroundStart, vertGroundEnd;
17
18 window.onload = function init()
19 {
20     var canvas = document.getElementById("gl-canvas");
21
22     gl = WebGLUtils.setupWebGL(canvas);
23     if( !gl ) {
24         alert("WebGL isn't available!");
25     }
26
27     generateTexCube();
28     generateHexaPyramid();
29     generateTexGround(10);
30
31     // virtual trackball
32     var trball = trackball(canvas.width, canvas.height);
33     var mouseDown = false;
```


scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
34
35   canvas.addEventListener("mousedown", function (event) {
36       trball.start(event.clientX, event.clientY);
37
38       mouseDown = true;
39   });
40
41   canvas.addEventListener("mouseup", function (event) {
42       mouseDown = false;
43   });
44
45   canvas.addEventListener("mousemove", function (event) {
46       if (mouseDown) {
47           trball.end(event.clientX, event.clientY);
48
49           trballMatrix = mat4(trball.rotationMatrix);
50       }
51   });
52
53   // Configure WebGL
54   gl.viewport(0, 0, canvas.width, canvas.height);
55   gl.clearColor(0.0, 0.0, 0.0, 1.0);
56
57   // Enable hidden-surface removal
58   gl.enable(gl.DEPTH_TEST);
59
60   // Load shaders and initialize attribute buffers
61   program0 = initShaders(gl, "colorVS", "colorFS");
62   gl.useProgram(program0);
63
64   // Load the data into the GPU
65   var bufferId = gl.createBuffer();
66   gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
```



```

67     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
68
69     // Associate our shader variables with our data buffer
70     var vPosition = gl.getAttribLocation(program0, "vPosition");
71     gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
72     gl.enableVertexAttribArray(vPosition);
73
74     var viewMatrix = lookAt(eye, at, up);
75     modelViewMatrixLoc0 = gl.getUniformLocation(program0, "modelViewMatrix");
76     gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(viewMatrix));
77     /*
78     // 3D orthographic viewing
79     var viewLength = 2.0;
80     if (canvas.width > canvas.height) {
81         var aspect = viewLength * canvas.width / canvas.height;
82         projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
83     }
84     else {
85         var aspect = viewLength * canvas.height / canvas.width;
86         projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
87     }
88     */
89     // 3D perspective viewing
90     var aspect = canvas.width / canvas.height;
91     projectionMatrix = perspective(90, aspect, 0.1, 1000);
92     var projectionMatrixLoc = gl.getUniformLocation(program0, "projectionMatrix");
93     gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
94     /*
95     //////////////////////////////////////
96     // program1 : Phong Shading
97
98     program1 = initShaders(gl, "phongVS", "phongFS");
99     gl.useProgram(program1);

```

1. **Introduction**
 2. **Background**
 3. **Methodology**
 4. **Results**
 5. **Discussion**
 6. **Conclusion**
 7. **References**
 8. **Appendix**
 9. **Figure 1**
 10. **Figure 2**
 11. **Figure 3**
 12. **Figure 4**
 13. **Figure 5**
 14. **Figure 6**
 15. **Figure 7**
 16. **Figure 8**
 17. **Figure 9**
 18. **Figure 10**
 19. **Figure 11**
 20. **Figure 12**
 21. **Figure 13**
 22. **Figure 14**
 23. **Figure 15**
 24. **Figure 16**
 25. **Figure 17**
 26. **Figure 18**
 27. **Figure 19**
 28. **Figure 20**
 29. **Figure 21**
 30. **Figure 22**
 31. **Figure 23**
 32. **Figure 24**
 33. **Figure 25**
 34. **Figure 26**
 35. **Figure 27**
 36. **Figure 28**
 37. **Figure 29**
 38. **Figure 30**
 39. **Figure 31**
 40. **Figure 32**
 41. **Figure 33**
 42. **Figure 34**
 43. **Figure 35**
 44. **Figure 36**
 45. **Figure 37**
 46. **Figure 38**
 47. **Figure 39**
 48. **Figure 40**
 49. **Figure 41**
 50. **Figure 42**
 51. **Figure 43**
 52. **Figure 44**
 53. **Figure 45**
 54. **Figure 46**
 55. **Figure 47**
 56. **Figure 48**
 57. **Figure 49**
 58. **Figure 50**
 59. **Figure 51**
 60. **Figure 52**
 61. **Figure 53**
 62. **Figure 54**
 63. **Figure 55**
 64. **Figure 56**
 65. **Figure 57**
 66. **Figure 58**
 67. **Figure 59**
 68. **Figure 60**
 69. **Figure 61**
 70. **Figure 62**
 71. **Figure 63**
 72. **Figure 64**
 73. **Figure 65**
 74. **Figure 66**
 75. **Figure 67**
 76. **Figure 68**
 77. **Figure 69**
 78. **Figure 70**
 79. **Figure 71**
 80. **Figure 72**
 81. **Figure 73**
 82. **Figure 74**
 83. **Figure 75**
 84. **Figure 76**
 85. **Figure 77**
 86. **Figure 78**
 87. **Figure 79**
 88. **Figure 80**
 89. **Figure 81**
 90. **Figure 82**
 91. **Figure 83**
 92. **Figure 84**
 93. **Figure 85**
 94. **Figure 86**
 95. **Figure 87**
 96. **Figure 88**
 97. **Figure 89**
 98. **Figure 90**
 99. **Figure 91**
 100. **Figure 92**
 101. **Figure 93**
 102. **Figure 94**
 103. **Figure 95**
 104. **Figure 96**
 105. **Figure 97**
 106. **Figure 98**
 107. **Figure 99**
 108. **Figure 100**
 109. **Figure 101**
 110. **Figure 102**
 111. **Figure 103**
 112. **Figure 104**
 113. **Figure 105**
 114. **Figure 106**
 115. **Figure 107**
 116. **Figure 108**
 117. **Figure 109**
 118. **Figure 110**
 119. **Figure 111**
 120. **Figure 112**
 121. **Figure 113**
 122. **Figure 114**
 123. **Figure 115**
 124. **Figure 116**
 125. **Figure 117**
 126. **Figure 118**
 127. **Figure 119**
 128. **Figure 120**
 129. **Figure 121**
 130. **Figure 122**
 131. **Figure 123**
 132. **Figure 124**
 133. **Figure 125**
 134. **Figure 126**
 135. **Figure 127**
 136. **Figure 128**
 137. **Figure 129**
 138. **Figure 130**
 139. **Figure 131**
 140. **Figure 132**
 141. **Figure 133**
 142. **Figure 134**
 143. **Figure 135**
 144. **Figure 136**
 145. **Figure 137**
 146. **Figure 138**
 147. **Figure 139**
 148. **Figure 140**
 149. **Figure 141**
 150. **Figure 142**
 151. **Figure 143**
 152. **Figure 144**
 153. **Figure 145**
 154. **Figure 146**
 155. **Figure 147**
 156. **Figure 148**
 157. **Figure 149**
 158. **Figure 150**
 159. **Figure 151**
 160. **Figure 152**
 161. **Figure 153**
 162. **Figure 154**
 163. **Figure 155**
 164. **Figure 156**
 165. **Figure 157**
 166. **Figure 158**
 167. **Figure 159**
 168. **Figure 160**
 169. **Figure 161**
 170. **Figure 162**
 171. **Figure 163**
 172. **Figure 164**
 173. **Figure 165**
 174. **Figure 166**
 175. **Figure 167**
 176. **Figure 168**
 177. **Figure 169**
 178. **Figure 170**
 179. **Figure 171**
 180. **Figure 172**
 181. **Figure 173**
 182. **Figure 174**
 183. **Figure 175**
 184. **Figure 176**
 185. **Figure 177**
 186. **Figure 178**
 187. **Figure 179**
 188. **Figure 180**
 189. **Figure 181**
 190. **Figure 182**
 191. **Figure 183**
 192. **Figure 184**
 193. **Figure 185**
 194. **Figure 186**
 195. **Figure 187**
 196. **Figure 188**
 197. **Figure 189**
 198. **Figure 190**
 199. **Figure 191**
 200. **Figure 192**
 201. **Figure 193**
 202. **Figure 194**
 203. **Figure 195**
 204. **Figure 196**
 205. **Figure 197**
 206. **Figure 198**
 207. **Figure 199**
 208. **Figure 200**
 209. **Figure 201**
 210. **Figure 202**
 211. **Figure 203**
 212. **Figure 204**
 213. **Figure 205**
 214. **Figure 206**
 215. **Figure 207**
 216. **Figure 208**
 217. **Figure 209**

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
100
101 // Load the data into the GPU
102 bufferId = gl.createBuffer();
103 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
104 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
105
106 // Associate our shader variables with our data buffer
107 vPosition = gl.getAttribLocation(program1, "vPosition");
108 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
109 gl.enableVertexAttribArray(vPosition);
110
111 // Create a buffer object, initialize it, and associate it with
112 // the associated attribute variable in our vertex shader
113 var nBufferId = gl.createBuffer();
114 gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
115 gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
116
117 var vNormal = gl.getAttribLocation(program1, "vNormal");
118 gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
119 gl.enableVertexAttribArray(vNormal);
120
121 modelViewMatrixLoc1 = gl.getUniformLocation(program1, "modelViewMatrix");
122 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(viewMatrix));
123
124 // 3D perspective viewing
125 projectionMatrixLoc = gl.getUniformLocation(program1, "projectionMatrix");
126 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
127
128 setLighting(program1);
129
130 ///////////////////////////////////////////////////
131 // program2 : Texture Mapping
132
```

```
100 // Load the data into the GPU
101 bufferId = gl.createBuffer();
102 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
103 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
104
105 // Associate our shader variables with our data buffer
106 vPosition = gl.getAttribLocation(program1, "vPosition");
107 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
108 gl.enableVertexAttribArray(vPosition);
109
110 // Create a buffer object, initialize it, and associate it with
111 // the associated attribute variable in our vertex shader
112 var nBufferId = gl.createBuffer();
113 gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
114 gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
115
116 var vNormal = gl.getAttribLocation(program1, "vNormal");
117 gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
118 gl.enableVertexAttribArray(vNormal);
119
120 modelViewMatrixLoc1 = gl.getUniformLocation(program1, "modelViewMatrix");
121 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(viewMatrix));
122
123 // 3D perspective viewing
124 projectionMatrixLoc = gl.getUniformLocation(program1, "projectionMatrix");
125 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
126
127 setLighting(program1);
128
129 ///////////////////////////////////////////////////
130 // program2 : Texture Mapping
131
132
```

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
133 program2 = initShaders(gl, "texMapVS", "texMapFS");
134 gl.useProgram(program2);
135
136 // Load the data into the GPU
137 bufferId = gl.createBuffer();
138 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
139 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
140
141 // Associate our shader variables with our data buffer
142 vPosition = gl.getAttribLocation(program2, "vPosition");
143 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
144 gl.enableVertexAttribArray(vPosition);
145
146 // Create a buffer object, initialize it, and associate it with
147 // the associated attribute variable in our vertex shader
148 nBufferId = gl.createBuffer();
149 gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
150 gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
151
152 vNormal = gl.getAttribLocation(program2, "vNormal");
153 gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
154 gl.enableVertexAttribArray(vNormal);
155
156 var tBufferId = gl.createBuffer();
157 gl.bindBuffer(gl.ARRAY_BUFFER, tBufferId);
158 gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoords), gl.STATIC_DRAW);
159
160 var vTexCoord = gl.getAttribLocation(program2, "vTexCoord");
161 gl.vertexAttribPointer(vTexCoord, 2, gl.FLOAT, false, 0, 0);
162 gl.enableVertexAttribArray(vTexCoord);
163
164 modelViewMatrixLoc2 = gl.getUniformLocation(program2, "modelViewMatrix");
165 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(viewMatrix));
```

```
133 program2 = initShaders(gl, "texMapVS", "texMapFS");
134 gl.useProgram(program2);
135
136 // Load the data into the GPU
137 bufferId = gl.createBuffer();
138 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
139 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
140
141 // Associate our shader variables with our data buffer
142 vPosition = gl.getAttribLocation(program2, "vPosition");
143 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
144 gl.enableVertexAttribArray(vPosition);
145
146 // Create a buffer object, initialize it, and associate it with
147 // the associated attribute variable in our vertex shader
148 nBufferId = gl.createBuffer();
149 gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
150 gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
151
152 vNormal = gl.getAttribLocation(program2, "vNormal");
153 gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
154 gl.enableVertexAttribArray(vNormal);
155
156 var tBufferId = gl.createBuffer();
157 gl.bindBuffer(gl.ARRAY_BUFFER, tBufferId);
158 gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoords), gl.STATIC_DRAW);
159
160 var vTexCoord = gl.getAttribLocation(program2, "vTexCoord");
161 gl.vertexAttribPointer(vTexCoord, 2, gl.FLOAT, false, 0, 0);
162 gl.enableVertexAttribArray(vTexCoord);
163
164 modelViewMatrixLoc2 = gl.getUniformLocation(program2, "modelViewMatrix");
165 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(viewMatrix));
```

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
166
167 // 3D perspective viewing
168 projectionMatrixLoc = gl.getUniformLocation(program2, "projectionMatrix");
169 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
170
171 setLighting(program2);
172 setTexture();
173 */
174 // Event listeners for buttons
175 var sinTheta = Math.sin(0.1);
176 var cosTheta = Math.cos(0.1);
177 document.getElementById("left").onclick = function () {
178     var newVecX = cosTheta*cameraVec[0] + sinTheta*cameraVec[2];
179     var newVecZ = -sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
180     cameraVec[0] = newVecX;
181     cameraVec[2] = newVecZ;
182 };
183 document.getElementById("right").onclick = function () {
184     var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
185     var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
186     cameraVec[0] = newVecX;
187     cameraVec[2] = newVecZ;
188 };
189 document.getElementById("up").onclick = function () {
190     var newPosX = eye[0] + 0.5 * cameraVec[0];
191     var newPosZ = eye[2] + 0.5 * cameraVec[2];
192     if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 ) {
193         eye[0] = newPosX;
194         eye[2] = newPosZ;
195     }
196 };
197 document.getElementById("down").onclick = function () {
198     var newPosX = eye[0] - 0.5 * cameraVec[0];
```


The image shows a LaTeX Beamer presentation slide. At the top left, there are navigation icons for a presentation (back, forward, search, etc.). The slide title is "Theorem 1.1". The main content is a complex mathematical theorem statement involving various symbols like \mathcal{A} , \mathcal{B} , \mathcal{C} , \mathcal{D} , \mathcal{E} , \mathcal{F} , \mathcal{G} , \mathcal{H} , \mathcal{I} , \mathcal{J} , \mathcal{K} , \mathcal{L} , \mathcal{M} , \mathcal{N} , \mathcal{O} , \mathcal{P} , \mathcal{Q} , \mathcal{R} , \mathcal{S} , \mathcal{T} , \mathcal{U} , \mathcal{V} , \mathcal{W} , \mathcal{X} , \mathcal{Y} , \mathcal{Z} . The theorem is followed by a proof sketch and a list of references. The slide is part of a presentation, as indicated by the navigation icons in the top left corner.

```

232     gl.uniform1f(gl.getUniformLocation(program, "shininess"), 100.0);
233 }
234
235 function setTexture() {
236     var image = new Image();
237     image.src = "images/logo.bmp";
238
239     var texture0 = gl.createTexture();
240     gl.activeTexture(gl.TEXTURE0);
241     gl.bindTexture(gl.TEXTURE_2D, texture0);
242
243     gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image);
244     gl.generateMipmap(gl.TEXTURE_2D);
245     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
246     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR_MIPMAP_LINEAR);
247
248     var image1 = new Image();
249     image1.src = "images/crate.bmp";
250
251     var texture1 = gl.createTexture();
252     gl.activeTexture(gl.TEXTURE1);
253     gl.bindTexture(gl.TEXTURE_2D, texture1);
254
255     gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image1);
256     gl.generateMipmap(gl.TEXTURE_2D);
257     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
258     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR_MIPMAP_LINEAR);
259 }
260
261 function render() {
262     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
263
264     theta += 2.0;

```

The image displays a vertical strip of 100 numbered pages from a book. The pages are arranged in a single column, with page numbers 1 through 100 visible on the left side. The content of the pages varies significantly, including:

- Textual content:** Various paragraphs of text in different fonts and sizes, some appearing to be in a non-Latin script (possibly Cyrillic or Greek).
- Tables:** Several tables with multiple columns and rows of data, some with headers and footers.
- Diagrams:** A few diagrams, including what appears to be a flowchart or a diagram of a structure.
- Figures:** A few figures, including a bar chart and a line graph.
- Footnotes:** Some pages contain footnotes or references at the bottom.

The overall appearance is that of a dense, multi-page document, possibly a technical manual or a scientific paper.

scene2.html

JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > render

```
265
266   at[0] = eye[0] + cameraVec[0];
267   at[1] = eye[1] + cameraVec[1];
268   at[2] = eye[2] + cameraVec[2];
269   var viewMatrix = lookAt(eye, at, up);
270
271   var colorLoc = gl.getUniformLocation(program0, "uColor");
272
273   // draw the ground
274   gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
275
276   modelViewMatrix = mult(viewMatrix, trballMatrix);
277   gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
278   gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);
279
280   for (var z=-5; z<5; z+=3) {
281     // draw a cube
282     gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
283
284     var rMatrix = mult(rotateY(theta), rotateZ(45));
285     var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
286     modelMatrix = mult(trballMatrix, modelMatrix);
287     modelViewMatrix = mult(viewMatrix, modelMatrix);
288     gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
289     gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
290
291     modelMatrix = mult(translate(3, 1.3, z), rMatrix);
292     modelMatrix = mult(trballMatrix, modelMatrix);
293     modelViewMatrix = mult(viewMatrix, modelMatrix);
294     gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
295     gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
296
297     // draw a hexa-pyramid
```

```
function lookAt(eye, at, up) {
    var x = at[0] - eye[0], y = at[1] - eye[1], z = at[2] - eye[2];
    var len = Math.sqrt(x*x + y*y + z*z);
    x /= len; y /= len; z /= len;
    var upx = up[0], upy = up[1], upz = up[2];
    var crossx = y*upz - z*upy, crossy = z*upx - x*upz, crossz = x*upy - y*upx;
    var len2 = crossx*crossx + crossy*crossy + crossz*crossz;
    var rad = len2 > 0.0001 ? 1/Math.sqrt(len2) : 0;
    crossx *= rad; crossy *= rad; crossz *= rad;
    var tr00 = crossx, tr01 = crossy, tr02 = crossz, tr10 = -crossz, tr11 = crossx, tr12 = crossy, tr20 = crossy, tr21 = -crossx, tr22 = crossz;
    var tr30 = eye[0], tr31 = eye[1], tr32 = eye[2];
    return [tr00, tr01, tr02, tr10, tr11, tr12, tr20, tr21, tr22, tr30, tr31, tr32];
}

function mult(m1, m2) {
    var m10 = m1[0], m11 = m1[1], m12 = m1[2], m13 = m1[3], m14 = m1[4], m15 = m1[5], m16 = m1[6], m17 = m1[7], m18 = m1[8], m19 = m1[9], m110 = m1[11], m111 = m1[12], m112 = m1[13];
    var m20 = m2[0], m21 = m2[1], m22 = m2[2], m23 = m2[3], m24 = m2[4], m25 = m2[5], m26 = m2[6], m27 = m2[7], m28 = m2[8], m29 = m2[9], m210 = m2[11], m211 = m2[12], m212 = m2[13];
    var m0 = m10*m20 + m11*m21 + m12*m22 + m13*m23 + m14*m24 + m15*m25 + m16*m26 + m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m1 = m10*m21 + m11*m22 + m12*m23 + m13*m24 + m14*m25 + m15*m26 + m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m2 = m10*m22 + m11*m23 + m12*m24 + m13*m25 + m14*m26 + m15*m27 + m16*m28 + m17*m29 + m18*m210 + m19*m211 + m110*m212;
    var m3 = m10*m23 + m11*m24 + m12*m25 + m13*m26 + m14*m27 + m15*m28 + m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m4 = m10*m24 + m11*m25 + m12*m26 + m13*m27 + m14*m28 + m15*m29 + m16*m210 + m17*m211 + m18*m212;
    var m5 = m10*m25 + m11*m26 + m12*m27 + m13*m28 + m14*m29 + m15*m210 + m16*m211 + m17*m212;
    var m6 = m10*m26 + m11*m27 + m12*m28 + m13*m29 + m14*m210 + m15*m211 + m16*m212;
    var m7 = m10*m27 + m11*m28 + m12*m29 + m13*m210 + m14*m211 + m15*m212;
    var m8 = m10*m28 + m11*m29 + m12*m210 + m13*m211 + m14*m212;
    var m9 = m10*m29 + m11*m210 + m12*m211 + m13*m212;
    var m10 = m11*m20 + m12*m21 + m13*m22 + m14*m23 + m15*m24 + m16*m25 + m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m11 = m11*m21 + m12*m22 + m13*m23 + m14*m24 + m15*m25 + m16*m26 + m17*m27 + m18*m28 + m19*m210 + m110*m211 + m111*m212;
    var m12 = m11*m22 + m12*m23 + m13*m24 + m14*m25 + m15*m26 + m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m13 = m11*m23 + m12*m24 + m13*m25 + m14*m26 + m15*m27 + m16*m28 + m17*m29 + m18*m210 + m19*m210 + m110*m211 + m111*m212;
    var m14 = m11*m24 + m12*m25 + m13*m26 + m14*m27 + m15*m28 + m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m15 = m11*m25 + m12*m26 + m13*m27 + m14*m28 + m15*m29 + m16*m210 + m17*m211 + m18*m212;
    var m16 = m11*m26 + m12*m27 + m13*m28 + m14*m29 + m15*m210 + m16*m211 + m17*m212;
    var m17 = m11*m27 + m12*m28 + m13*m29 + m14*m210 + m15*m211 + m16*m212;
    var m18 = m11*m28 + m12*m29 + m13*m210 + m14*m211 + m15*m212;
    var m19 = m11*m29 + m12*m210 + m13*m211 + m14*m212;
    var m20 = m12*m20 + m13*m21 + m14*m22 + m15*m23 + m16*m24 + m17*m25 + m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m21 = m12*m21 + m13*m22 + m14*m23 + m15*m24 + m16*m25 + m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m22 = m12*m22 + m13*m23 + m14*m24 + m15*m25 + m16*m26 + m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m23 = m12*m23 + m13*m24 + m14*m25 + m15*m26 + m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m24 = m12*m24 + m13*m25 + m14*m26 + m15*m27 + m16*m28 + m17*m29 + m18*m210 + m19*m210 + m110*m211 + m111*m212;
    var m25 = m12*m25 + m13*m26 + m14*m27 + m15*m28 + m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m26 = m12*m26 + m13*m27 + m14*m28 + m15*m29 + m16*m210 + m17*m211 + m18*m212;
    var m27 = m12*m27 + m13*m28 + m14*m29 + m15*m210 + m16*m211 + m17*m212;
    var m28 = m12*m28 + m13*m29 + m14*m210 + m15*m211 + m16*m212;
    var m29 = m12*m29 + m13*m210 + m14*m211 + m15*m212;
    var m30 = m13*m20 + m14*m21 + m15*m22 + m16*m23 + m17*m24 + m18*m25 + m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m31 = m13*m21 + m14*m22 + m15*m23 + m16*m24 + m17*m25 + m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m32 = m13*m22 + m14*m23 + m15*m24 + m16*m25 + m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m33 = m13*m23 + m14*m24 + m15*m25 + m16*m26 + m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m34 = m13*m24 + m14*m25 + m15*m26 + m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m35 = m13*m25 + m14*m26 + m15*m27 + m16*m28 + m17*m29 + m18*m210 + m19*m210 + m110*m211 + m111*m212;
    var m36 = m13*m26 + m14*m27 + m15*m28 + m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m37 = m13*m27 + m14*m28 + m15*m29 + m16*m210 + m17*m211 + m18*m212;
    var m38 = m13*m28 + m14*m29 + m15*m210 + m16*m211 + m17*m212;
    var m39 = m13*m29 + m14*m210 + m15*m211 + m16*m212;
    var m40 = m14*m20 + m15*m21 + m16*m22 + m17*m23 + m18*m24 + m19*m25 + m110*m210 + m111*m211 + m112*m212;
    var m41 = m14*m21 + m15*m22 + m16*m23 + m17*m24 + m18*m25 + m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m42 = m14*m22 + m15*m23 + m16*m24 + m17*m25 + m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m43 = m14*m23 + m15*m24 + m16*m25 + m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m44 = m14*m24 + m15*m25 + m16*m26 + m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m45 = m14*m25 + m15*m26 + m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m46 = m14*m26 + m15*m27 + m16*m28 + m17*m29 + m18*m210 + m19*m210 + m110*m211 + m111*m212;
    var m47 = m14*m27 + m15*m28 + m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m48 = m14*m28 + m15*m29 + m16*m210 + m17*m211 + m18*m212;
    var m49 = m14*m29 + m15*m210 + m16*m211 + m17*m212;
    var m50 = m15*m20 + m16*m21 + m17*m22 + m18*m23 + m19*m24 + m110*m210 + m111*m211 + m112*m212;
    var m51 = m15*m21 + m16*m22 + m17*m23 + m18*m24 + m19*m25 + m110*m210 + m111*m211 + m112*m212;
    var m52 = m15*m22 + m16*m23 + m17*m24 + m18*m25 + m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m53 = m15*m23 + m16*m24 + m17*m25 + m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m54 = m15*m24 + m16*m25 + m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m55 = m15*m25 + m16*m26 + m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m56 = m15*m26 + m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m57 = m15*m27 + m16*m28 + m17*m29 + m18*m210 + m19*m210 + m110*m211 + m111*m212;
    var m58 = m15*m28 + m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m59 = m15*m29 + m16*m210 + m17*m211 + m18*m212;
    var m60 = m16*m20 + m17*m21 + m18*m22 + m19*m23 + m110*m210 + m111*m211 + m112*m212;
    var m61 = m16*m21 + m17*m22 + m18*m23 + m19*m24 + m110*m210 + m111*m211 + m112*m212;
    var m62 = m16*m22 + m17*m23 + m18*m24 + m19*m25 + m110*m210 + m111*m211 + m112*m212;
    var m63 = m16*m23 + m17*m24 + m18*m25 + m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m64 = m16*m24 + m17*m25 + m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m65 = m16*m25 + m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m66 = m16*m26 + m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m67 = m16*m27 + m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m68 = m16*m28 + m17*m29 + m18*m210 + m19*m210 + m110*m211 + m111*m212;
    var m69 = m16*m29 + m17*m210 + m18*m211 + m19*m212;
    var m70 = m17*m20 + m18*m21 + m19*m22 + m110*m210 + m111*m211 + m112*m212;
    var m71 = m17*m21 + m18*m22 + m19*m23 + m110*m210 + m111*m211 + m112*m212;
    var m72 = m17*m22 + m18*m23 + m19*m24 + m110*m210 + m111*m211 + m112*m212;
    var m73 = m17*m23 + m18*m24 + m19*m25 + m110*m210 + m111*m211 + m112*m212;
    var m74 = m17*m24 + m18*m25 + m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m75 = m17*m25 + m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m76 = m17*m26 + m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m77 = m17*m27 + m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m78 = m17*m28 + m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m79 = m17*m29 + m18*m210 + m19*m211 + m110*m211 + m111*m212;
    var m80 = m18*m20 + m19*m21 + m110*m210 + m111*m211 + m112*m212;
    var m81 = m18*m21 + m19*m22 + m110*m210 + m111*m211 + m112*m212;
    var m82 = m18*m22 + m19*m23 + m110*m210 + m111*m211 + m112*m212;
    var m83 = m18*m23 + m19*m24 + m110*m210 + m111*m211 + m112*m212;
    var m84 = m18*m24 + m19*m25 + m110*m210 + m111*m211 + m112*m212;
    var m85 = m18*m25 + m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m86 = m18*m26 + m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m87 = m18*m27 + m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m88 = m18*m28 + m19*m29 + m110*m210 + m111*m211 + m112*m212;
    var m89 = m18*m29 + m19*m210 + m110*m211 + m111*m212;
    var m90 = m19*m20 + m110*m210 + m111*m211 + m112*m212;
    var m91 = m19*m21 + m110*m210 + m111*m211 + m112*m212;
    var m92 = m19*m22 + m110*m210 + m111*m211 + m112*m212;
    var m93 = m19*m23 + m110*m210 + m111*m211 + m112*m212;
    var m94 = m19*m24 + m110*m210 + m111*m211 + m112*m212;
    var m95 = m19*m25 + m110*m210 + m111*m211 + m112*m212;
    var m96 = m19*m26 + m110*m210 + m111*m211 + m112*m212;
    var m97 = m19*m27 + m110*m210 + m111*m211 + m112*m212;
    var m98 = m19*m28 + m110*m210 + m111*m211 + m112*m212;
    var m99 = m19*m29 + m110*m210 + m111*m211 + m112*m212;
    return [m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15, m16, m17, m18, m19, m20, m21, m22, m23, m24, m25, m26, m27, m28, m29, m30, m31, m32, m33, m34, m35, m36, m37, m38, m39, m40, m41, m42, m43, m44, m45, m46, m47, m48, m49, m50, m51, m52, m53, m54, m55, m56, m57, m58, m59, m60, m61, m62, m63, m64, m65, m66, m67, m68, m69, m70, m71, m72, m73, m74, m75, m76, m77, m78, m79, m80, m81, m82, m83, m84, m85, m86, m87, m88, m89, m90, m91, m92, m93, m94, m95, m96, m97, m98, m99];
}
```

```

298     gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 1.0); // blue
299
300     modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
301     modelMatrix = mult(trballMatrix, modelMatrix);
302     modelViewMatrix = mult(viewMatrix, modelMatrix);
303     gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
304     gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
305
306     modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
307     modelMatrix = mult(trballMatrix, modelMatrix);
308     modelViewMatrix = mult(viewMatrix, modelMatrix);
309     gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
310     gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
311 }
312
313 requestAnimationFrame(render);
314 }
315
316 function generateTexCube() {
317     vertCubeStart = points.length;
318     vertCubeEnd = 0;
319     texQuad(1, 0, 3, 2);
320     texQuad(2, 3, 7, 6);
321     texQuad(3, 0, 4, 7);
322     texQuad(4, 5, 6, 7);
323     texQuad(5, 4, 0, 1);
324     texQuad(6, 5, 1, 2);
325 }
326
327 function texQuad(a, b, c, d) {
328     const vertexPos = [
329         vec4(-0.5, -0.5, -0.5, 1.0),
330         vec4( 0.5, -0.5, -0.5, 1.0),

```

Figure 1: Schematic representation of the experimental design.

The figure illustrates the experimental design, showing the flow of participants through different phases and conditions. The design is divided into two main sections: **Pre-Test** and **Main Experiment**.

Pre-Test Section:

- Pre-Test Phase:** 100 participants are initially assigned. They are divided into two groups of 50. Each group is further divided into two subgroups of 25. The subgroups are then divided into two conditions: **Pre-Test** and **Post-Test**. The **Pre-Test** condition is further divided into **Pre-Test** and **Post-Test** conditions. The **Post-Test** condition is further divided into **Pre-Test** and **Post-Test** conditions.

Main Experiment Section:

- Main Experiment Phase:** 100 participants are initially assigned. They are divided into two groups of 50. Each group is further divided into two subgroups of 25. The subgroups are then divided into two conditions: **Main Experiment** and **Post-Test**. The **Main Experiment** condition is further divided into **Main Experiment** and **Post-Test** conditions. The **Post-Test** condition is further divided into **Main Experiment** and **Post-Test** conditions.

The flowchart shows the progression of participants through these phases and conditions, with the final number of participants in each condition being 100.

 texQuad

```

331         vec4( 0.5, 0.5, -0.5, 1.0),
332         vec4(-0.5, 0.5, -0.5, 1.0),
333         vec4(-0.5, -0.5, 0.5, 1.0),
334         vec4( 0.5, -0.5, 0.5, 1.0),
335         vec4( 0.5, 0.5, 0.5, 1.0),
336         vec4(-0.5, 0.5, 0.5, 1.0)
337     ];
338
339     const vertexNormals = [
340         vec4(-0.57735, -0.57735, -0.57735, 0.0),
341         vec4( 0.57735, -0.57735, -0.57735, 0.0),
342         vec4( 0.57735, 0.57735, -0.57735, 0.0),
343         vec4(-0.57735, 0.57735, -0.57735, 0.0),
344         vec4(-0.57735, -0.57735, 0.57735, 0.0),
345         vec4( 0.57735, -0.57735, 0.57735, 0.0),
346         vec4( 0.57735, 0.57735, 0.57735, 0.0),
347         vec4(-0.57735, 0.57735, 0.57735, 0.0)
348     ];
349
350     const texCoord = [
351         vec2(0, 0),
352         vec2(0, 1),
353         vec2(1, 1),
354         vec2(1, 0)
355     ];
356
357     // two triangles: (a, b, c) and (a, c, d)
358     // solid colored faces
359     points.push(vertexPos[a]);
360     normals.push(vertexNormals[a]);
361     texCoords.push(texCoord[0]);
362     vertCubeEnd++;
363

```

[illegible]

```

364     points.push(vertexPos[b]);
365     normals.push(vertexNormals[b]);
366     texCoords.push(texCoord[1]);
367     vertCubeEnd++;
368
369     points.push(vertexPos[c]);
370     normals.push(vertexNormals[c]);
371     texCoords.push(texCoord[2]);
372     vertCubeEnd++;
373
374     points.push(vertexPos[a]);
375     normals.push(vertexNormals[a]);
376     texCoords.push(texCoord[0]);
377     vertCubeEnd++;
378
379     points.push(vertexPos[c]);
380     normals.push(vertexNormals[c]);
381     texCoords.push(texCoord[2]);
382     vertCubeEnd++;
383
384     points.push(vertexPos[d]);
385     normals.push(vertexNormals[d]);
386     texCoords.push(texCoord[3]);
387     vertCubeEnd++;
388 }
389
390 function generateTexGround(scale) {
391     vertGroundStart = points.length;
392     vertGroundEnd = 0;
393     for(var x=-scale; x<scale; x++) {
394         for(var z=-scale; z<scale; z++) {
395             // two triangles
396             points.push(vec4(x, -1.0, z, 1.0));

```

A. Experimental design

Experimental design

Pre-test

Pre-test 1

Pre-test 1a

Pre-test 1b

Pre-test 2

Pre-test 2a

Pre-test 2b

Main test

Main test 1

Main test 1a

Main test 1b

Main test 2

Main test 2a

Main test 2b

B. Data analysis

Data analysis

Pre-test

Pre-test 1

Pre-test 1a

Pre-test 1b

Pre-test 2

Pre-test 2a

Pre-test 2b

Main test

Main test 1

Main test 1a

Main test 1b

Main test 2

Main test 2a

Main test 2b

C. Results

Results

Pre-test

Pre-test 1

Pre-test 1a

Pre-test 1b

Pre-test 2

Pre-test 2a

Pre-test 2b

Main test

Main test 1

Main test 1a

Main test 1b

Main test 2

Main test 2a

Main test 2b

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > ...

a

b

c

d

e

f

g

h

i

j

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > generateHexaPyramid

```
430 const vertexPos = [  
431     vec4(0.0, 0.5, 0.0, 1.0),  
432     vec4(1.0, 0.5, 0.0, 1.0),  
433     vec4(0.5, 0.5, -0.866, 1.0),  
434     vec4(-0.5, 0.5, -0.866, 1.0),  
435     vec4(-1.0, 0.5, 0.0, 1.0),  
436     vec4(-0.5, 0.5, 0.866, 1.0),  
437     vec4(0.5, 0.5, 0.866, 1.0),  
438     vec4(0.0, -1.0, 0.0, 1.0)  
439 ];  
440  
441 const vertexNormal = [  
442     vec4(0.0, 1.0, 0.0, 0.0),  
443     vec4(1.0, 0.0, 0.0, 0.0),  
444     vec4(0.5, 0.0, -0.866, 0.0),  
445     vec4(-0.5, 0.0, -0.866, 0.0),  
446     vec4(-1.0, 0.0, 0.0, 0.0),  
447     vec4(-0.5, 0.0, 0.866, 0.0),  
448     vec4(0.5, 0.0, 0.866, 0.0),  
449     vec4(0.0, -1.0, 0.0, 0.0)  
450 ];  
451  
452 vertHexaStart = points.length;  
453 vertHexaEnd = 0;  
454 for (var i=1; i<6; i++) {  
455     points.push(vertexPos[i]);  
456     normals.push(vertexNormal[i]);  
457     vertHexaEnd++;  
458  
459     points.push(vertexPos[i]);  
460     normals.push(vertexNormal[i]);  
461     vertHexaEnd++;  
462
```



scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > generateHexaPyramid

```
463     points.push(vertexPos[i+1]);
464     normals.push(vertexNormal[0]);
465     vertHexaEnd++;
466
467     points.push(vertexPos[7]);
468     normals.push(vertexNormal[7]);
469     vertHexaEnd++;
470
471     points.push(vertexPos[i+1]);
472     normals.push(vertexNormal[i+1]);
473     vertHexaEnd++;
474
475     points.push(vertexPos[i]);
476     normals.push(vertexNormal[i]);
477     vertHexaEnd++;
478 }
479 points.push(vertexPos[0]);
480 normals.push(vertexNormal[0]);
481 vertHexaEnd++;
482
483 points.push(vertexPos[6]);
484 normals.push(vertexNormal[0]);
485 vertHexaEnd++;
486
487 points.push(vertexPos[1]);
488 normals.push(vertexNormal[0]);
489 vertHexaEnd++;
490
491 points.push(vertexPos[7]);
492 normals.push(vertexNormal[7]);
493 vertHexaEnd++;
494
495 points.push(vertexPos[1]);
```





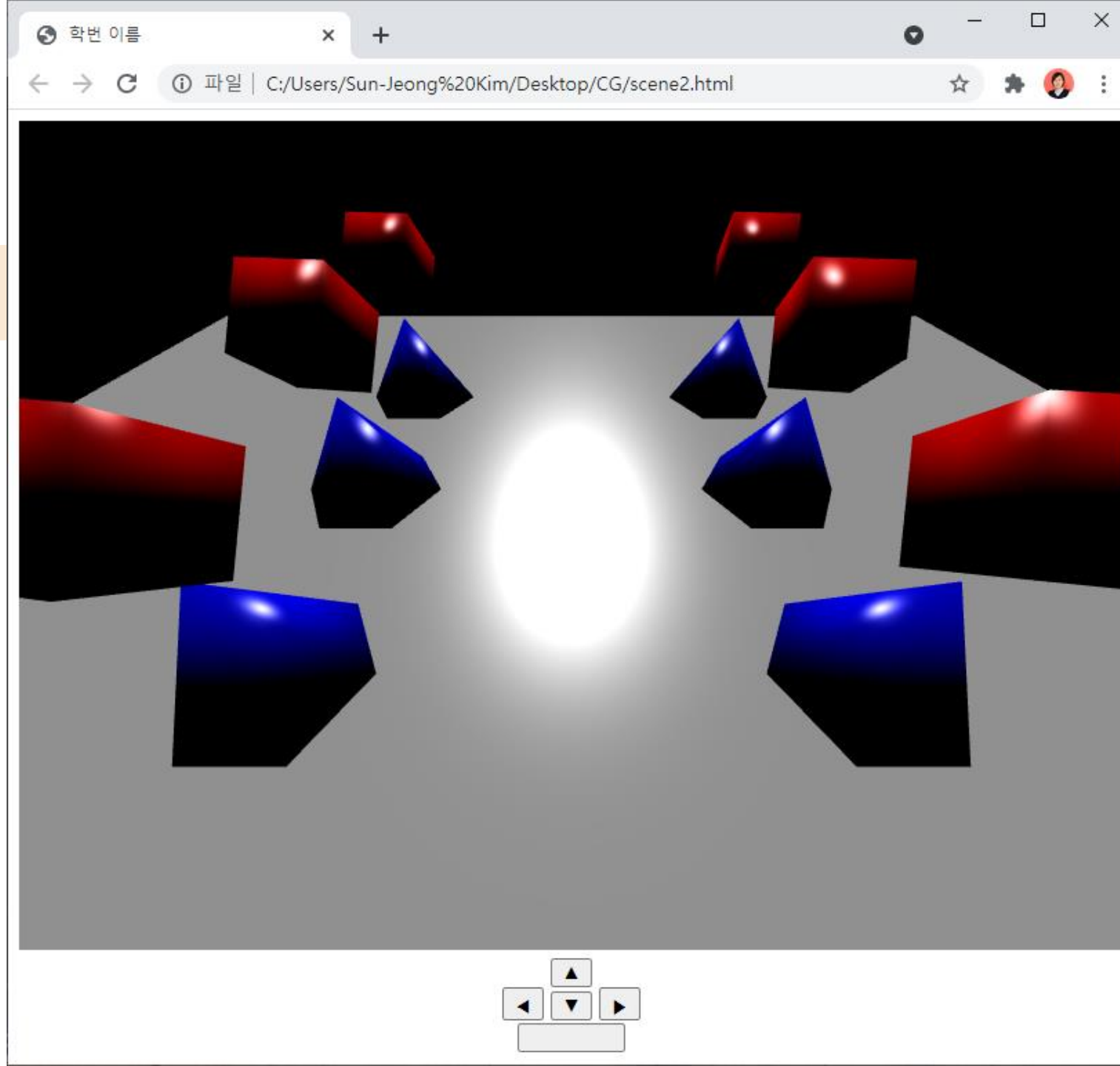
scene2.html JS scene2.js ×



C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > ...

```
496     normals.push(vertexNormal[1]);
497     vertHexaEnd++;
498
499     points.push(vertexPos[6]);
500     normals.push(vertexNormal[6]);
501     vertHexaEnd++;
502 }
503
```





scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
94
95 ///////////////////////////////////////////////////
96 // program1 : Phong Shading
97
98 program1 = initShaders(gl, "phongVS", "phongFS");
99 gl.useProgram(program1);
100
101 // Load the data into the GPU
102 bufferId = gl.createBuffer();
103 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
104 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
105
106 // Associate our shader variables with our data buffer
107 vPosition = gl.getAttribLocation(program1, "vPosition");
108 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
109 gl.enableVertexAttribArray(vPosition);
110
111 // Create a buffer object, initialize it, and associate it with
112 // the associated attribute variable in our vertex shader
113 var nBufferId = gl.createBuffer();
114 gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
115 gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
116
117 var vNormal = gl.getAttribLocation(program1, "vNormal");
118 gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
119 gl.enableVertexAttribArray(vNormal);
120
121 modelViewMatrixLoc1 = gl.getUniformLocation(program1, "modelViewMatrix");
122 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(viewMatrix));
123
124 // 3D perspective viewing
125 projectionMatrixLoc = gl.getUniformLocation(program1, "projectionMatrix");
126 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
```

```
1 // WebGL 2.0
2
3 // Canvas
4 const canvas = document.getElementById("canvas");
5 const gl = canvas.getContext("webgl2");
6
7 // Shader
8 const vs = `
9   precision mediump float;
10  attribute vec4 position;
11  attribute vec4 normal;
12  attribute vec4 color;
13  attribute vec4 texCoord;
14  attribute vec4 texCoord2;
15  attribute vec4 texCoord3;
16  attribute vec4 texCoord4;
17  attribute vec4 texCoord5;
18  attribute vec4 texCoord6;
19  attribute vec4 texCoord7;
20  attribute vec4 texCoord8;
21  attribute vec4 texCoord9;
22  attribute vec4 texCoord10;
23  attribute vec4 texCoord11;
24  attribute vec4 texCoord12;
25  attribute vec4 texCoord13;
26  attribute vec4 texCoord14;
27  attribute vec4 texCoord15;
28  attribute vec4 texCoord16;
29  attribute vec4 texCoord17;
30  attribute vec4 texCoord18;
31  attribute vec4 texCoord19;
32  attribute vec4 texCoord20;
33  attribute vec4 texCoord21;
34  attribute vec4 texCoord22;
35  attribute vec4 texCoord23;
36  attribute vec4 texCoord24;
37  attribute vec4 texCoord25;
38  attribute vec4 texCoord26;
39  attribute vec4 texCoord27;
40  attribute vec4 texCoord28;
41  attribute vec4 texCoord29;
42  attribute vec4 texCoord30;
43  attribute vec4 texCoord31;
44  attribute vec4 texCoord32;
45  attribute vec4 texCoord33;
46  attribute vec4 texCoord34;
47  attribute vec4 texCoord35;
48  attribute vec4 texCoord36;
49  attribute vec4 texCoord37;
50  attribute vec4 texCoord38;
51  attribute vec4 texCoord39;
52  attribute vec4 texCoord40;
53  attribute vec4 texCoord41;
54  attribute vec4 texCoord42;
55  attribute vec4 texCoord43;
56  attribute vec4 texCoord44;
57  attribute vec4 texCoord45;
58  attribute vec4 texCoord46;
59  attribute vec4 texCoord47;
60  attribute vec4 texCoord48;
61  attribute vec4 texCoord49;
62  attribute vec4 texCoord50;
63  attribute vec4 texCoord51;
64  attribute vec4 texCoord52;
65  attribute vec4 texCoord53;
66  attribute vec4 texCoord54;
67  attribute vec4 texCoord55;
68  attribute vec4 texCoord56;
69  attribute vec4 texCoord57;
70  attribute vec4 texCoord58;
71  attribute vec4 texCoord59;
72  attribute vec4 texCoord60;
73  attribute vec4 texCoord61;
74  attribute vec4 texCoord62;
75  attribute vec4 texCoord63;
76  attribute vec4 texCoord64;
77  attribute vec4 texCoord65;
78  attribute vec4 texCoord66;
79  attribute vec4 texCoord67;
80  attribute vec4 texCoord68;
81  attribute vec4 texCoord69;
82  attribute vec4 texCoord70;
83  attribute vec4 texCoord71;
84  attribute vec4 texCoord72;
85  attribute vec4 texCoord73;
86  attribute vec4 texCoord74;
87  attribute vec4 texCoord75;
88  attribute vec4 texCoord76;
89  attribute vec4 texCoord77;
90  attribute vec4 texCoord78;
91  attribute vec4 texCoord79;
92  attribute vec4 texCoord80;
93  attribute vec4 texCoord81;
94  attribute vec4 texCoord82;
95  attribute vec4 texCoord83;
96  attribute vec4 texCoord84;
97  attribute vec4 texCoord85;
98  attribute vec4 texCoord86;
99  attribute vec4 texCoord87;
100 attribute vec4 texCoord88;
101 attribute vec4 texCoord89;
102 attribute vec4 texCoord90;
103 attribute vec4 texCoord91;
104 attribute vec4 texCoord92;
105 attribute vec4 texCoord93;
106 attribute vec4 texCoord94;
107 attribute vec4 texCoord95;
108 attribute vec4 texCoord96;
109 attribute vec4 texCoord97;
110 attribute vec4 texCoord98;
111 attribute vec4 texCoord99;
112
113 varying vec4 vColor;
114 varying vec4 vTexCoord;
115 varying vec4 vTexCoord2;
116 varying vec4 vTexCoord3;
117 varying vec4 vTexCoord4;
118 varying vec4 vTexCoord5;
119 varying vec4 vTexCoord6;
120 varying vec4 vTexCoord7;
121 varying vec4 vTexCoord8;
122 varying vec4 vTexCoord9;
123 varying vec4 vTexCoord10;
124 varying vec4 vTexCoord11;
125 varying vec4 vTexCoord12;
126 varying vec4 vTexCoord13;
127 varying vec4 vTexCoord14;
128 varying vec4 vTexCoord15;
129 varying vec4 vTexCoord16;
130 varying vec4 vTexCoord17;
131 varying vec4 vTexCoord18;
132 varying vec4 vTexCoord19;
133 varying vec4 vTexCoord20;
134 varying vec4 vTexCoord21;
135 varying vec4 vTexCoord22;
136 varying vec4 vTexCoord23;
137 varying vec4 vTexCoord24;
138 varying vec4 vTexCoord25;
139 varying vec4 vTexCoord26;
140 varying vec4 vTexCoord27;
141 varying vec4 vTexCoord28;
142 varying vec4 vTexCoord29;
143 varying vec4 vTexCoord30;
144 varying vec4 vTexCoord31;
145 varying vec4 vTexCoord32;
146 varying vec4 vTexCoord33;
147 varying vec4 vTexCoord34;
148 varying vec4 vTexCoord35;
149 varying vec4 vTexCoord36;
150 varying vec4 vTexCoord37;
151 varying vec4 vTexCoord38;
152 varying vec4 vTexCoord39;
153 varying vec4 vTexCoord40;
154 varying vec4 vTexCoord41;
155 varying vec4 vTexCoord42;
156 varying vec4 vTexCoord43;
157 varying vec4 vTexCoord44;
158 varying vec4 vTexCoord45;
159 varying vec4 vTexCoord46;
160 varying vec4 vTexCoord47;
161 varying vec4 vTexCoord48;
162 varying vec4 vTexCoord49;
163 varying vec4 vTexCoord50;
164 varying vec4 vTexCoord51;
165 varying vec4 vTexCoord52;
166 varying vec4 vTexCoord53;
167 varying vec4 vTexCoord54;
168 varying vec4 vTexCoord55;
169 varying vec4 vTexCoord56;
170 varying vec4 vTexCoord57;
171 varying vec4 vTexCoord58;
172 varying vec4 vTexCoord59;
173 varying vec4 vTexCoord60;
174 varying vec4 vTexCoord61;
175 varying vec4 vTexCoord62;
176 varying vec4 vTexCoord63;
177 varying vec4 vTexCoord64;
178 varying vec4 vTexCoord65;
179 varying vec4 vTexCoord66;
180 varying vec4 vTexCoord67;
181 varying vec4 vTexCoord68;
182 varying vec4 vTexCoord69;
183 varying vec4 vTexCoord70;
184 varying vec4 vTexCoord71;
185 varying vec4 vTexCoord72;
186 varying vec4 vTexCoord73;
187 varying vec4 vTexCoord74;
188 varying vec4 vTexCoord75;
189 varying vec4 vTexCoord76;
190 varying vec4 vTexCoord77;
191 varying vec4 vTexCoord78;
192 varying vec4 vTexCoord79;
193 varying vec4 vTexCoord80;
194 varying vec4 vTexCoord81;
195 varying vec4 vTexCoord82;
196 varying vec4 vTexCoord83;
197 varying vec4 vTexCoord84;
198 varying vec4 vTexCoord85;
199 varying vec4 vTexCoord86;
200 varying vec4 vTexCoord87;
201 varying vec4 vTexCoord88;
202 varying vec4 vTexCoord89;
203 varying vec4 vTexCoord90;
204 varying vec4 vTexCoord91;
205 varying vec4 vTexCoord92;
206 varying vec4 vTexCoord93;
207 varying vec4 vTexCoord94;
208 varying vec4 vTexCoord95;
209 varying vec4 vTexCoord96;
210 varying vec4 vTexCoord97;
211 varying vec4 vTexCoord98;
212 varying vec4 vTexCoord99;
213
214 void main() {
215   vColor = color;
216   vTexCoord = texCoord;
217   vTexCoord2 = texCoord2;
218   vTexCoord3 = texCoord3;
219   vTexCoord4 = texCoord4;
220   vTexCoord5 = texCoord5;
221   vTexCoord6 = texCoord6;
222   vTexCoord7 = texCoord7;
223   vTexCoord8 = texCoord8;
224   vTexCoord9 = texCoord9;
225   vTexCoord10 = texCoord10;
226   vTexCoord11 = texCoord11;
227   vTexCoord12 = texCoord12;
228   vTexCoord13 = texCoord13;
229   vTexCoord14 = texCoord14;
230   vTexCoord15 = texCoord15;
231   vTexCoord16 = texCoord16;
232   vTexCoord17 = texCoord17;
233   vTexCoord18 = texCoord18;
234   vTexCoord19 = texCoord19;
235   vTexCoord20 = texCoord20;
236   vTexCoord21 = texCoord21;
237   vTexCoord22 = texCoord22;
238   vTexCoord23 = texCoord23;
239   vTexCoord24 = texCoord24;
240   vTexCoord25 = texCoord25;
241   vTexCoord26 = texCoord26;
242   vTexCoord27 = texCoord27;
243   vTexCoord28 = texCoord28;
244   vTexCoord29 = texCoord29;
245   vTexCoord30 = texCoord30;
246   vTexCoord31 = texCoord31;
247   vTexCoord32 = texCoord32;
248   vTexCoord33 = texCoord33;
249   vTexCoord34 = texCoord34;
250   vTexCoord35 = texCoord35;
251   vTexCoord36 = texCoord36;
252   vTexCoord37 = texCoord37;
253   vTexCoord38 = texCoord38;
254   vTexCoord39 = texCoord39;
255   vTexCoord40 = texCoord40;
256   vTexCoord41 = texCoord41;
257   vTexCoord42 = texCoord42;
258   vTexCoord43 = texCoord43;
259   vTexCoord44 = texCoord44;
260   vTexCoord45 = texCoord45;
261   vTexCoord46 = texCoord46;
262   vTexCoord47 = texCoord47;
263   vTexCoord48 = texCoord48;
264   vTexCoord49 = texCoord49;
265   vTexCoord50 = texCoord50;
266   vTexCoord51 = texCoord51;
267   vTexCoord52 = texCoord52;
268   vTexCoord53 = texCoord53;
269   vTexCoord54 = texCoord54;
270   vTexCoord55 = texCoord55;
271   vTexCoord56 = texCoord56;
272   vTexCoord57 = texCoord57;
273   vTexCoord58 = texCoord58;
274   vTexCoord59 = texCoord59;
275   vTexCoord60 = texCoord60;
276   vTexCoord61 = texCoord61;
277   vTexCoord62 = texCoord62;
278   vTexCoord63 = texCoord63;
279   vTexCoord64 = texCoord64;
280   vTexCoord65 = texCoord65;
281   vTexCoord66 = texCoord66;
282   vTexCoord67 = texCoord67;
283   vTexCoord68 = texCoord68;
284   vTexCoord69 = texCoord69;
285   vTexCoord70 = texCoord70;
286   vTexCoord71 = texCoord71;
287   vTexCoord72 = texCoord72;
288   vTexCoord73 = texCoord73;
289   vTexCoord74 = texCoord74;
290   vTexCoord75 = texCoord75;
291   vTexCoord76 = texCoord76;
292   vTexCoord77 = texCoord77;
293   vTexCoord78 = texCoord78;
294   vTexCoord79 = texCoord79;
295   vTexCoord80 = texCoord80;
296   vTexCoord81 = texCoord81;
297   vTexCoord82 = texCoord82;
298   vTexCoord83 = texCoord83;
299   vTexCoord84 = texCoord84;
300   vTexCoord85 = texCoord85;
301   vTexCoord86 = texCoord86;
302   vTexCoord87 = texCoord87;
303   vTexCoord88 = texCoord88;
304   vTexCoord89 = texCoord89;
305   vTexCoord90 = texCoord90;
306   vTexCoord91 = texCoord91;
307   vTexCoord92 = texCoord92;
308   vTexCoord93 = texCoord93;
309   vTexCoord94 = texCoord94;
310   vTexCoord95 = texCoord95;
311   vTexCoord96 = texCoord96;
312   vTexCoord97 = texCoord97;
313   vTexCoord98 = texCoord98;
314   vTexCoord99 = texCoord99;
315 }
316 `;
317
318 const fs = `
319   precision mediump float;
320   varying vec4 vColor;
321   varying vec4 vTexCoord;
322   varying vec4 vTexCoord2;
323   varying vec4 vTexCoord3;
324   varying vec4 vTexCoord4;
325   varying vec4 vTexCoord5;
326   varying vec4 vTexCoord6;
327   varying vec4 vTexCoord7;
328   varying vec4 vTexCoord8;
329   varying vec4 vTexCoord9;
330   varying vec4 vTexCoord10;
331   varying vec4 vTexCoord11;
332   varying vec4 vTexCoord12;
333   varying vec4 vTexCoord13;
334   varying vec4 vTexCoord14;
335   varying vec4 vTexCoord15;
336   varying vec4 vTexCoord16;
337   varying vec4 vTexCoord17;
338   varying vec4 vTexCoord18;
339   varying vec4 vTexCoord19;
340   varying vec4 vTexCoord20;
341   varying vec4 vTexCoord21;
342   varying vec4 vTexCoord22;
343   varying vec4 vTexCoord23;
344   varying vec4 vTexCoord24;
345   varying vec4 vTexCoord25;
346   varying vec4 vTexCoord26;
347   varying vec4 vTexCoord27;
348   varying vec4 vTexCoord28;
349   varying vec4 vTexCoord29;
350   varying vec4 vTexCoord30;
351   varying vec4 vTexCoord31;
352   varying vec4 vTexCoord32;
353   varying vec4 vTexCoord33;
354   varying vec4 vTexCoord34;
355   varying vec4 vTexCoord35;
356   varying vec4 vTexCoord36;
357   varying vec4 vTexCoord37;
358   varying vec4 vTexCoord38;
359   varying vec4 vTexCoord39;
360   varying vec4 vTexCoord40;
361   varying vec4 vTexCoord41;
362   varying vec4 vTexCoord42;
363   varying vec4 vTexCoord43;
364   varying vec4 vTexCoord44;
365   varying vec4 vTexCoord45;
366   varying vec4 vTexCoord46;
367   varying vec4 vTexCoord47;
368   varying vec4 vTexCoord48;
369   varying vec4 vTexCoord49;
370   varying vec4 vTexCoord50;
371   varying vec4 vTexCoord51;
372   varying vec4 vTexCoord52;
373   varying vec4 vTexCoord53;
374   varying vec4 vTexCoord54;
375   varying vec4 vTexCoord55;
376   varying vec4 vTexCoord56;
377   varying vec4 vTexCoord57;
378   varying vec4 vTexCoord58;
379   varying vec4 vTexCoord59;
380   varying vec4 vTexCoord60;
381   varying vec4 vTexCoord61;
382   varying vec4 vTexCoord62;
383   varying vec4 vTexCoord63;
384   varying vec4 vTexCoord64;
385   varying vec4 vTexCoord65;
386   varying vec4 vTexCoord66;
387   varying vec4 vTexCoord67;
388   varying vec4 vTexCoord68;
389   varying vec4 vTexCoord69;
390   varying vec4 vTexCoord70;
391   varying vec4 vTexCoord71;
392   varying vec4 vTexCoord72;
393   varying vec4 vTexCoord73;
394   varying vec4 vTexCoord74;
395   varying vec4 vTexCoord75;
396   varying vec4 vTexCoord76;
397   varying vec4 vTexCoord77;
398   varying vec4 vTexCoord78;
399   varying vec4 vTexCoord79;
400   varying vec4 vTexCoord80;
401   varying vec4 vTexCoord81;
402   varying vec4 vTexCoord82;
403   varying vec4 vTexCoord83;
404   varying vec4 vTexCoord84;
405   varying vec4 vTexCoord85;
406   varying vec4 vTexCoord86;
407   varying vec4 vTexCoord87;
408   varying vec4 vTexCoord88;
409   varying vec4 vTexCoord89;
410   varying vec4 vTexCoord90;
411   varying vec4 vTexCoord91;
412   varying vec4 vTexCoord92;
413   varying vec4 vTexCoord93;
414   varying vec4 vTexCoord94;
415   varying vec4 vTexCoord95;
416   varying vec4 vTexCoord96;
417   varying vec4 vTexCoord97;
418   varying vec4 vTexCoord98;
419   varying vec4 vTexCoord99;
420
421   void main() {
422     gl_FragColor = vColor;
423   }
424 `;
425
426 // Compile and link the shaders
427 const vsCompiled = gl.compileShader(vs);
428 const fsCompiled = gl.compileShader(fs);
429 const program = gl.createProgram();
430 gl.attachShader(program, vsCompiled);
431 gl.attachShader(program, fsCompiled);
432 gl.linkProgram(program);
433
434 // Use the program
435 gl.useProgram(program);
```

<> scene2.html

JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
127
128     setLighting(program1);
129  /*
130  //////////////////////////////////////
131  // program2 : Texture Mapping
132
133  program2 = initShaders(gl, "texMapVS", "texMapFS");
134  gl.useProgram(program2);
135
136  // Load the data into the GPU
137  bufferId = gl.createBuffer();
138  gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
139  gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
140
141  // Associate our shader variables with our data buffer
142  vPosition = gl.getAttribLocation(program2, "vPosition");
143  gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
144  gl.enableVertexAttribArray(vPosition);
145
146  // Create a buffer object, initialize it, and associate it with
147  // the associated attribute variable in our vertex shader
148  nBufferId = gl.createBuffer();
149  gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
150  gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
151
152  vNormal = gl.getAttribLocation(program2, "vNormal");
153  gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
154  gl.enableVertexAttribArray(vNormal);
155
156  var tBufferId = gl.createBuffer();
157  gl.bindBuffer(gl.ARRAY_BUFFER, tBufferId);
158  gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoords), gl.STATIC_DRAW);
159
```

```
1 // Shader for texture mapping
2 #version 300 es
3 precision mediump float;
4
5 // Vertex Shader
6 #ifdef VERTEX_SHADER
7 in vec4 vPosition;
8 in vec4 vNormal;
9 in vec2 vTexCoords;
10
11 out vec4 vColor;
12 out vec2 vUV;
13
14 // Uniforms
15 uniform mat4 uModelView;
16 uniform mat4 uProjection;
17 uniform sampler2D uTexture;
18
19 // Constants
20 const vec3 gColor = vec3(1.0, 1.0, 1.0);
21
22 void main()
23 {
24     // Transform position
25     vec4 mvPosition = uModelView * vPosition;
26     vec4 projPosition = uProjection * mvPosition;
27
28     // Transform normal
29     vec3 transformedNormal = mat3(uModelView) * vNormal;
30     float length = length(transformedNormal);
31     transformedNormal = transformedNormal / length;
32
33     // Calculate UV
34     vUV = vTexCoords;
35
36     // Color
37     vColor = gColor;
38 }
39
40 // Fragment Shader
41 #ifdef FRAGMENT_SHADER
42 in vec4 vColor;
43 in vec2 vUV;
44
45 out vec4 fColor;
46
47 // Uniforms
48 uniform sampler2D uTexture;
49
50 void main()
51 {
52     // Sample texture
53     vec4 texColor = texture(uTexture, vUV);
54
55     // Final color
56     fColor = vColor * texColor;
57 }
58
59 // Main function
60 void main()
61 {
62     // Initialize GL
63     glClearColor(0.0, 0.0, 0.0, 1.0);
64     glClear(GL_COLOR_BUFFER_BIT);
65
66     // Load shaders
67     int programId = initShaders(gl, "texMapVS", "texMapFS");
68     gl.useProgram(programId);
69
70     // Load data
71     loadPoints(gl);
72     loadNormals(gl);
73     loadTexCoords(gl);
74
75     // Draw
76     glDrawArrays(GL_TRIANGLES, 0, 36);
77
78     // Swap buffers
79     glSwapBuffers(window);
80 }
81
82 // Load points
83 void loadPoints(GLContext gl)
84 {
85     float points[] = {
86         // Front face
87         -0.5, -0.5, 0.5, 0.5, -0.5, 0.5,
88         // Back face
89         0.5, 0.5, 0.5, 0.5, 0.5, -0.5,
90         // Right face
91         0.5, -0.5, 0.5, 0.5, 0.5, -0.5,
92         // Left face
93         -0.5, -0.5, 0.5, -0.5, 0.5, 0.5,
94         // Bottom face
95         -0.5, 0.5, 0.5, 0.5, 0.5, -0.5,
96         // Top face
97         -0.5, -0.5, 0.5, 0.5, -0.5, 0.5
98     };
99     gl.bufferData(GL_ARRAY_BUFFER, sizeof(points), GL_STATIC_DRAW);
100 }
101
102 // Load normals
103 void loadNormals(GLContext gl)
104 {
105     float normals[] = {
106         // Front face
107         0.0, 0.0, 1.0, 0.0, 0.0, 1.0,
108         // Back face
109         0.0, 0.0, -1.0, 0.0, 0.0, -1.0,
110         // Right face
111         1.0, 0.0, 0.0, 1.0, 0.0, 0.0,
112         // Left face
113         -1.0, 0.0, 0.0, -1.0, 0.0, 0.0,
114         // Bottom face
115         0.0, 1.0, 0.0, 0.0, 1.0, 0.0,
116         // Top face
117         0.0, -1.0, 0.0, 0.0, -1.0, 0.0
118     };
119     gl.bufferData(GL_ARRAY_BUFFER, sizeof(normals), GL_STATIC_DRAW);
120 }
121
122 // Load texCoords
123 void loadTexCoords(GLContext gl)
124 {
125     float texCoords[] = {
126         // Front face
127         0.0, 0.0, 1.0, 1.0, 1.0, 0.0,
128         // Back face
129         1.0, 1.0, 1.0, 1.0, 1.0, 0.0,
130         // Right face
131         1.0, 0.0, 1.0, 1.0, 1.0, 0.0,
132         // Left face
133         0.0, 0.0, 1.0, 1.0, 1.0, 0.0,
134         // Bottom face
135         0.0, 1.0, 1.0, 1.0, 1.0, 0.0,
136         // Top face
137         0.0, 0.0, 1.0, 1.0, 1.0, 0.0
138     };
139     gl.bufferData(GL_ARRAY_BUFFER, sizeof(texCoords), GL_STATIC_DRAW);
140 }
141
142 // Init shaders
143 int initShaders(GLContext gl, const char* vertexShader, const char* fragmentShader)
144 {
145     int shaderId = glCreateShader(GL_VERTEX_SHADER);
146     glShaderSource(shaderId, 1, &vertexShader, NULL);
147     glCompileShader(shaderId);
148
149     shaderId = glCreateShader(GL_FRAGMENT_SHADER);
150     glShaderSource(shaderId, 1, &fragmentShader, NULL);
151     glCompileShader(shaderId);
152
153     int programId = glCreateProgram();
154     glAttachShader(programId, shaderId);
155     glAttachShader(programId, shaderId);
156     glLinkProgram(programId);
157
158     return programId;
159 }
```

```

270 //var colorLoc = gl.getUniformLocation(program0, "uColor");
271
272 var diffuseProductLoc = gl.getUniformLocation(program1, "diffuseProduct");
273
274 // draw the ground
275 //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
276 gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
277
278 modelViewMatrix = mult(viewMatrix, trballMatrix);
279 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
280 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
281 gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);
282
283 for (var z=-5; z<5; z+=3) {
284     // draw a cube
285     //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
286     gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
287
288     var rMatrix = mult(rotateY(theta), rotateZ(45));
289     var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
290     modelMatrix = mult(trballMatrix, modelMatrix);
291     modelViewMatrix = mult(viewMatrix, modelMatrix);
292     //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
293     gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
294     gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
295
296     modelMatrix = mult(translate(3, 1.3, z), rMatrix);
297     modelMatrix = mult(trballMatrix, modelMatrix);
298     modelViewMatrix = mult(viewMatrix, modelMatrix);
299     //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
300     gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
301     gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
302

```

A Study 1

Pre-test (N = 10) → Training (N = 20) → Test (N = 20)

B Study 2

Pre-test (N = 10) → Training (N = 20) → Test (N = 20)

C Study 3

Pre-test (N = 10) → Training (N = 20) → Test (N = 20)

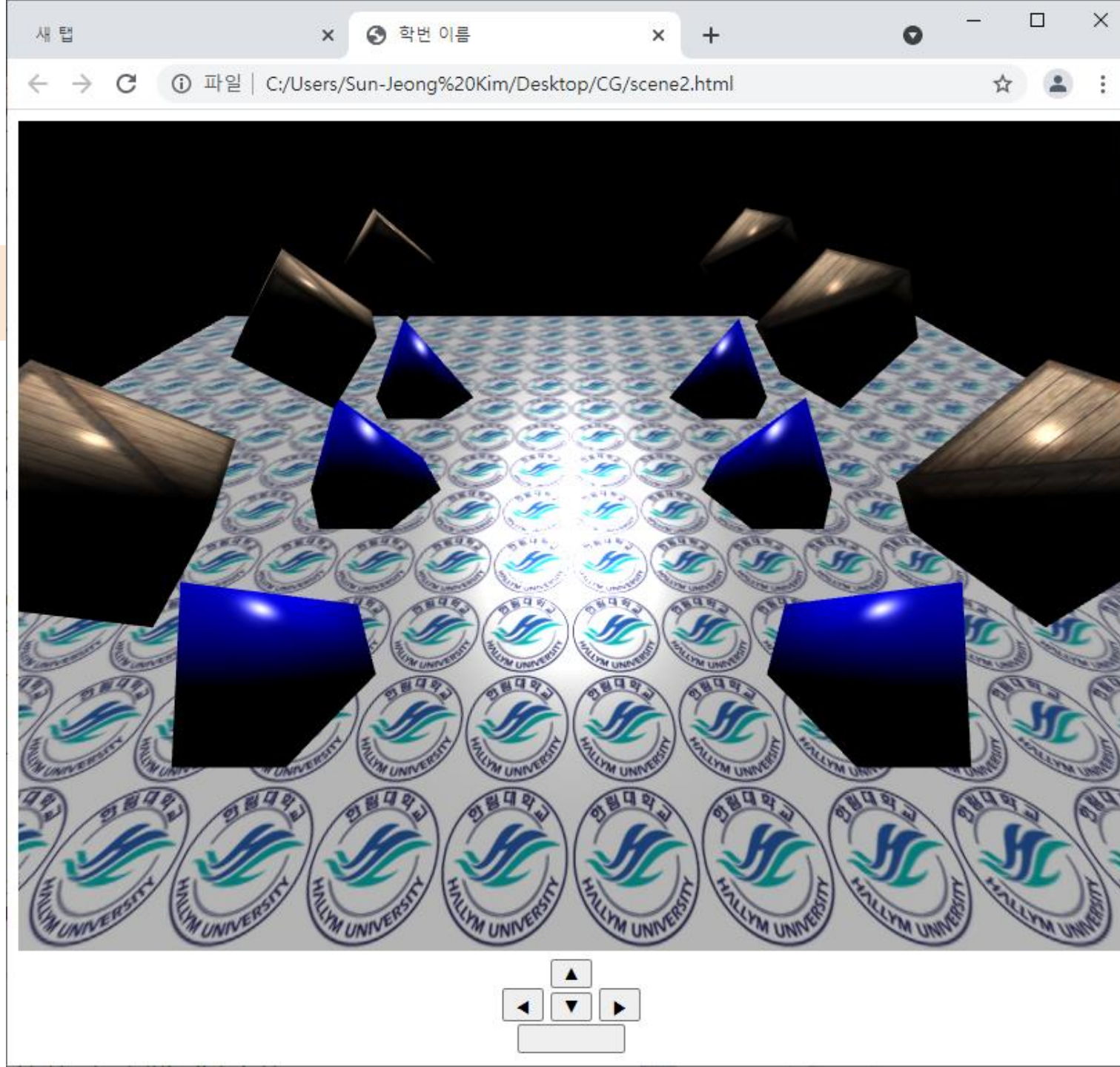
Figure 1. Schematic representation of the experimental design.

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > render

```
303 // draw a hexa-pyramid
304 //gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 1.0); // blue
305 gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);
306
307 modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
308 modelMatrix = mult(trballMatrix, modelMatrix);
309 modelViewMatrix = mult(viewMatrix, modelMatrix);
310 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
311 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
312 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
313
314 modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
315 modelMatrix = mult(trballMatrix, modelMatrix);
316 modelViewMatrix = mult(viewMatrix, modelMatrix);
317 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
318 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
319 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
320
321
322 requestAnimationFrame(render);
323 }
324
325 function generateTexCube() {
326     vertCubeStart = points.length;
327     vertCubeEnd = 0;
328     texQuad(1, 0, 3, 2);
329     texQuad(2, 3, 7, 6);
330     texQuad(3, 0, 4, 7);
331     texQuad(4, 5, 6, 7);
332     texQuad(5, 4, 0, 1);
333     texQuad(6, 5, 1, 2);
334 }
335
```





[illegible]

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init

```
162 gl.enableVertexAttribArray(vTexCoord);
163
164 modelViewMatrixLoc2 = gl.getUniformLocation(program2, "modelViewMatrix");
165 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(viewMatrix));
166
167 // 3D perspective viewing
168 projectionMatrixLoc = gl.getUniformLocation(program2, "projectionMatrix");
169 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
170
171 setLighting(program2);
172 setTexture();
173
174 // Event listeners for buttons
175 var sinTheta = Math.sin(0.1);
176 var cosTheta = Math.cos(0.1);
177 document.getElementById("left").onclick = function () {
178     var newVecX = cosTheta*cameraVec[0] + sinTheta*cameraVec[2];
179     var newVecZ = -sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
180     cameraVec[0] = newVecX;
181     cameraVec[2] = newVecZ;
182 };
183 document.getElementById("right").onclick = function () {
184     var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
185     var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
186     cameraVec[0] = newVecX;
187     cameraVec[2] = newVecZ;
188 };
189 document.getElementById("up").onclick = function () {
190     var newPosX = eye[0] + 0.5 * cameraVec[0];
191     var newPosZ = eye[2] + 0.5 * cameraVec[2];
192     if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 ) {
193         eye[0] = newPosX;
194         eye[2] = newPosZ;
```

```
195     }
196 }
197
198 // Draw the scene
199 function drawScene() {
200     gl.clear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
201     gl.enable(GL_DEPTH_TEST);
202
203     // Draw the ground plane
204     drawGroundPlane();
205
206     // Draw the walls
207     drawWalls();
208
209     // Draw the ceiling
210     drawCeiling();
211
212     // Draw the camera frustum
213     drawFrustum();
214
215     // Swap buffers
216     gl.swapBuffers();
217
218     // Request next frame
219     requestAnimationFrame(drawScene);
220 }
221
222 // Initialize the scene
223 function initScene() {
224     // Create WebGL context
225     gl = WebGLRenderingContext.create(canvas);
226
227     // Load shaders
228     loadShaders(gl);
229
230     // Compile shaders
231     compileShaders(gl);
232
233     // Link program
234     linkProgram(gl);
235
236     // Set viewport
237     gl.viewport(0, 0, canvas.width, canvas.height);
238
239     // Set clear color
240     gl.clearColor(0.0, 0.0, 0.0, 1.0);
241
242     // Set clear buffer
243     gl.clearBuffer(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
244
245     // Set culling
246     gl.cullFace(GL_BACK);
247
248     // Set front face
249     gl.frontFace(GL_CCW);
250
251     // Set depth test
252     gl.enable(GL_DEPTH_TEST);
253
254     // Set perspective projection
255     gl.enable(GL_PERSPECTIVE_PROJECTION);
256
257     // Set camera position
258     cameraVec[0] = 0.0;
259     cameraVec[1] = 0.0;
260     cameraVec[2] = 0.0;
261
262     // Set eye position
263     eye[0] = 0.0;
264     eye[1] = 0.0;
265     eye[2] = 0.0;
266
267     // Set look-at position
268     lookAt[0] = 0.0;
269     lookAt[1] = 0.0;
270     lookAt[2] = 0.0;
271
272     // Set view matrix
273     viewMatrix = new Float32Array(16);
274
275     // Set projection matrix
276     projectionMatrix = new Float32Array(16);
277
278     // Set model matrix
279     modelMatrix = new Float32Array(16);
280
281     // Set model-view matrix
282     modelViewMatrix = new Float32Array(16);
283
284     // Set model-view-projection matrix
285     modelViewProjectionMatrix = new Float32Array(16);
286
287     // Set texture
288     texture = new Image();
289     texture.src = "textures/ground.jpg";
290
291     // Set lighting
292     setLighting(gl);
293
294     // Set texture
295     setTexture(gl);
296
297     // Draw the scene
298     drawScene();
299 }
300
301 // Start the scene
302 initScene();
```

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > render

```
273
274 // draw the ground
275 //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
276 //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
277 gl.useProgram(program2);
278 gl.uniform1i(gl.getUniformLocation(program2, "texture"), 0);
279
280 modelViewMatrix = mult(viewMatrix, trballMatrix);
281 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
282 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
283 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
284 gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);
285
286 for (var z=-5; z<5; z+=3) {
287 // draw a cube
288 //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
289 //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
290 gl.useProgram(program2);
291 gl.uniform1i(gl.getUniformLocation(program2, "texture"), 1);
292
293 var rMatrix = mult(rotateY(theta), rotateZ(45));
294 var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
295 modelMatrix = mult(trballMatrix, modelMatrix);
296 modelViewMatrix = mult(viewMatrix, modelMatrix);
297 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
298 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
299 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
300 gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
301
302 modelMatrix = mult(translate(3, 1.3, z), rMatrix);
303 modelMatrix = mult(trballMatrix, modelMatrix);
304 modelViewMatrix = mult(viewMatrix, modelMatrix);
305 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
```

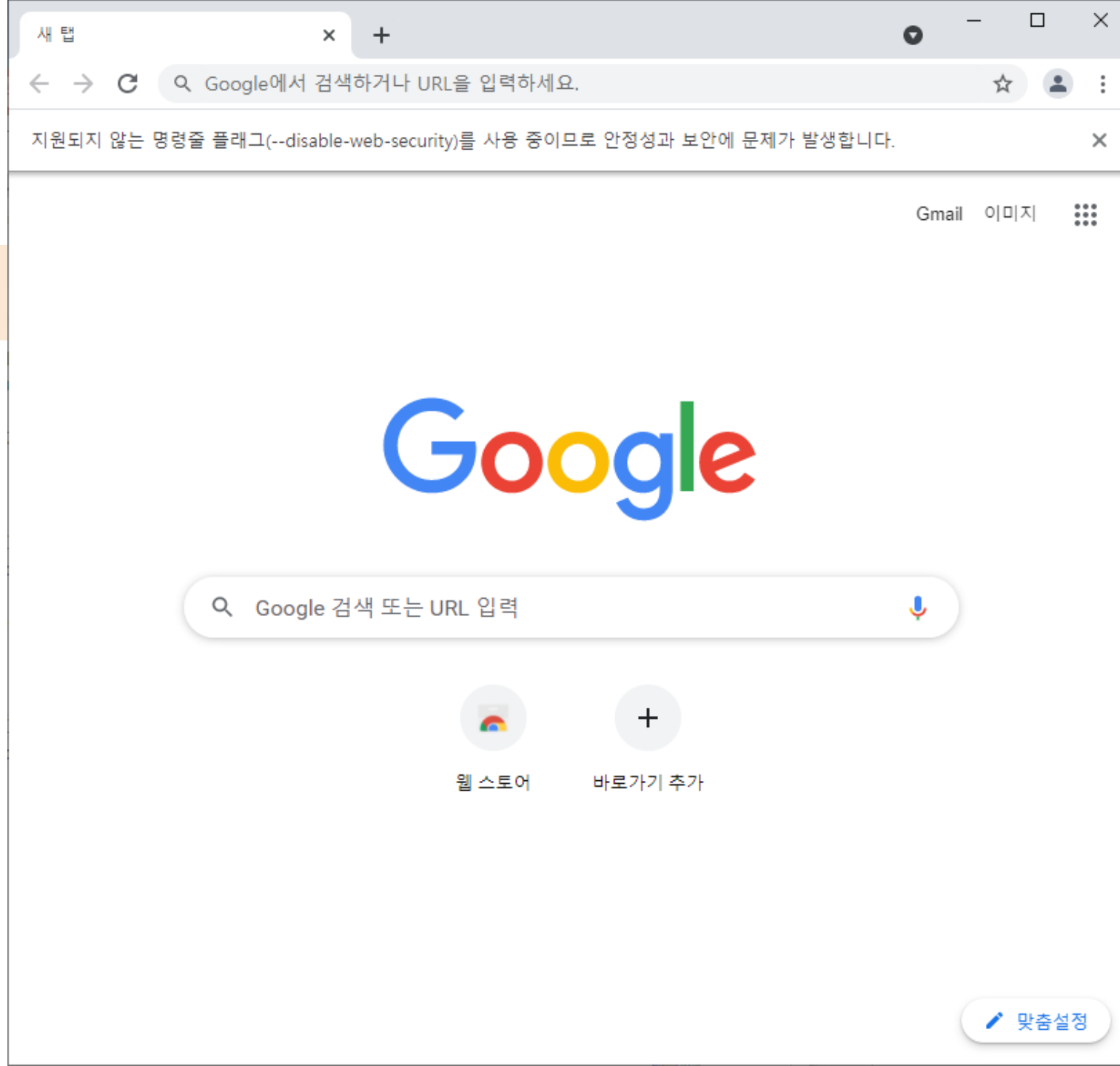


<> scene2.html

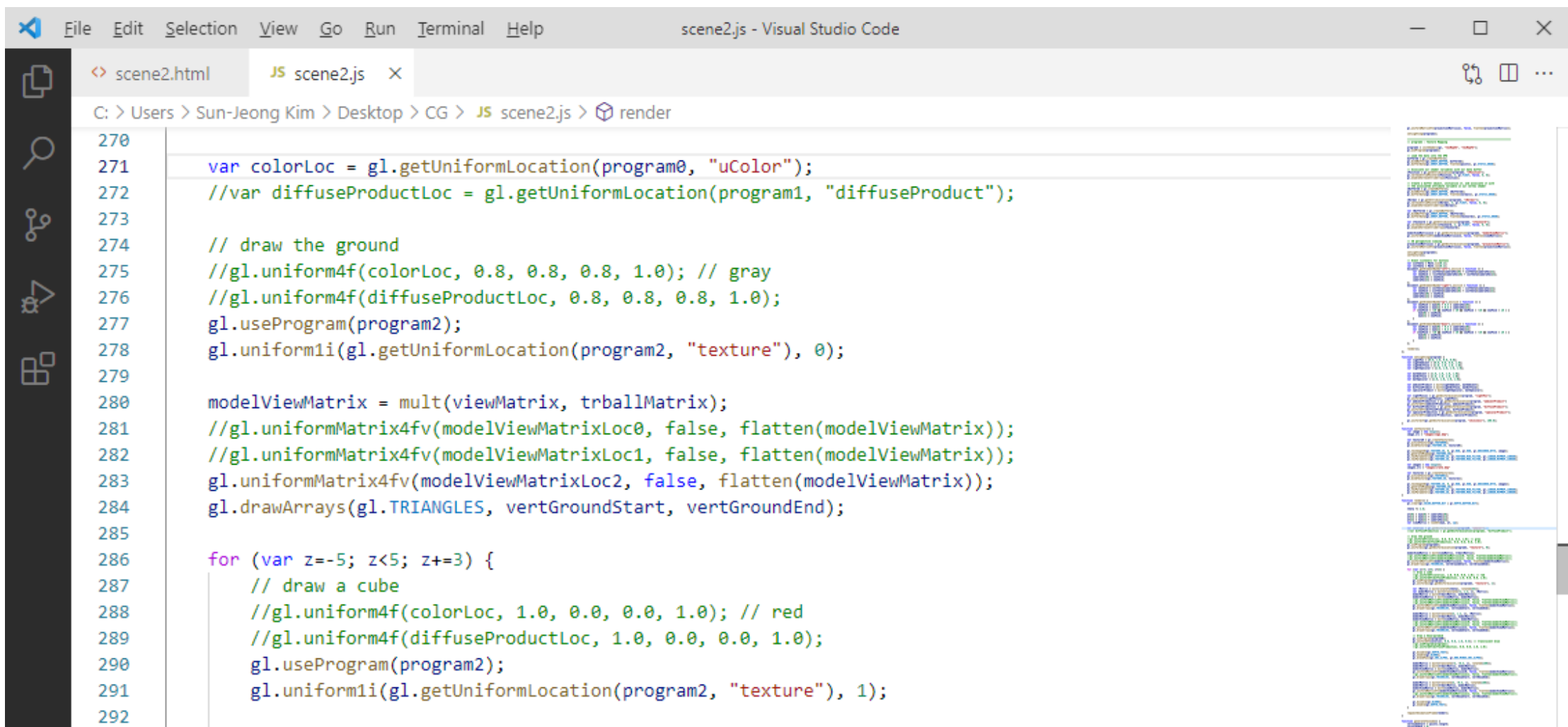
JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > render

```
306 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
307 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
308 gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
309
310 // draw a hexa-pyramid
311 //gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 1.0); // blue
312 gl.useProgram(program1);
313 gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);
314
315 modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
316 modelMatrix = mult(trballMatrix, modelMatrix);
317 modelViewMatrix = mult(viewMatrix, modelMatrix);
318 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
319 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
320 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
321
322 modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
323 modelMatrix = mult(trballMatrix, modelMatrix);
324 modelViewMatrix = mult(viewMatrix, modelMatrix);
325 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
326 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
327 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
328
329
330 requestAnimationFrame(render);
331 }
332
333 function generateTexCube() {
334     vertCubeStart = points.length;
335     vertCubeEnd = 0;
336     texQuad(1, 0, 3, 2);
337     texQuad(2, 3, 7, 6);
338     texQuad(3, 0, 4, 7);
```



Alpha Blending

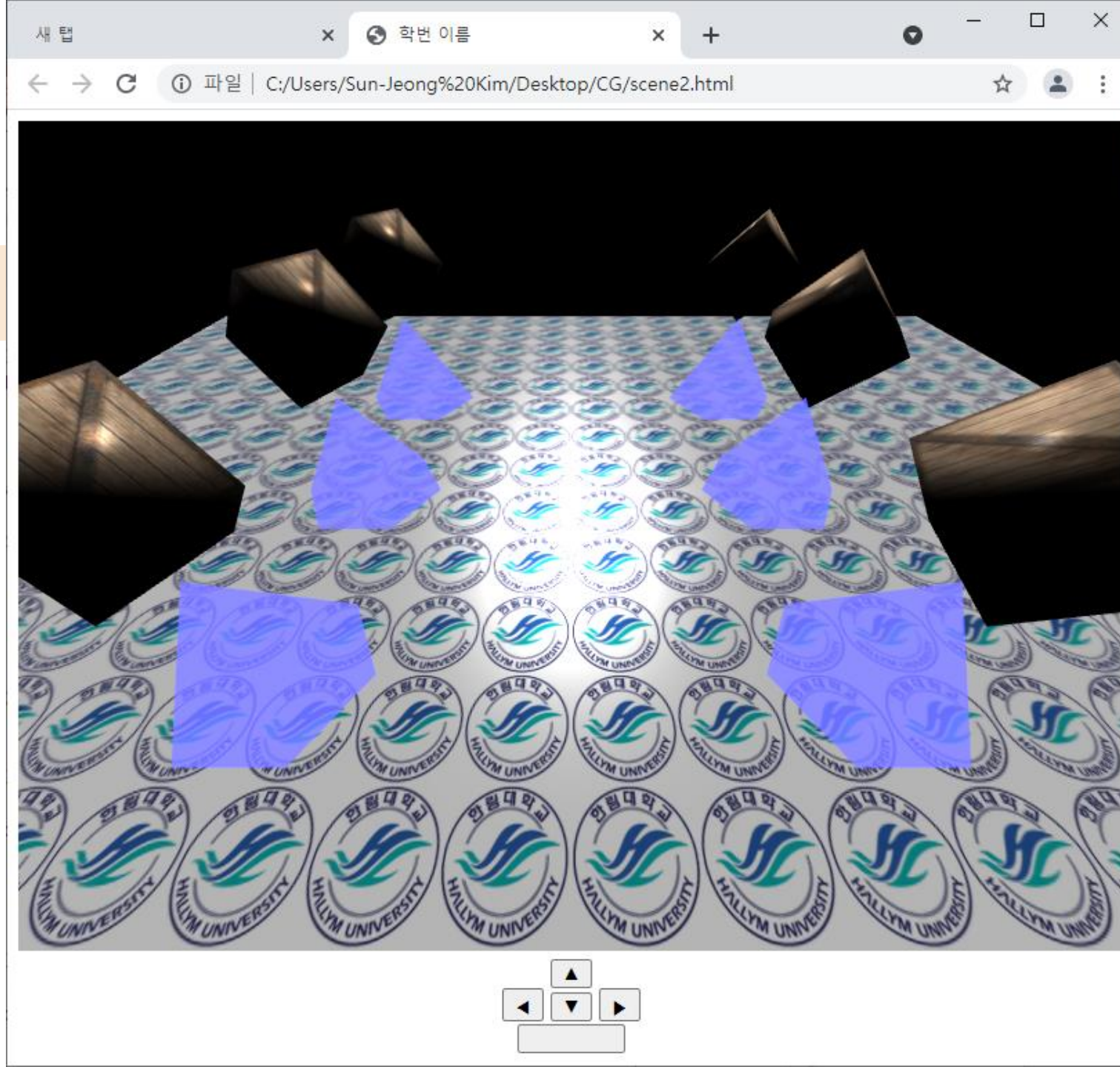


```
File Edit Selection View Go Run Terminal Help scene2.js - Visual Studio Code
scene2.html JS scene2.js X
C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > render
270
271 var colorLoc = gl.getUniformLocation(program0, "uColor");
272 //var diffuseProductLoc = gl.getUniformLocation(program1, "diffuseProduct");
273
274 // draw the ground
275 //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
276 //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
277 gl.useProgram(program2);
278 gl.uniform1i(gl.getUniformLocation(program2, "texture"), 0);
279
280 modelViewMatrix = mult(viewMatrix, trballMatrix);
281 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
282 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
283 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
284 gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);
285
286 for (var z=-5; z<5; z+=3) {
287     // draw a cube
288     //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
289     //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
290     gl.useProgram(program2);
291     gl.uniform1i(gl.getUniformLocation(program2, "texture"), 1);
292 }
```

scene2.html JS scene2.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > render

```
309
310 // draw a hexa-pyramid
311 gl.useProgram(program0);
312 gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 0.5); // translucent blue
313 //gl.useProgram(program1);
314 //gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);
315
316 gl.disable(gl.DEPTH_TEST);
317 gl.enable(gl.BLEND);
318 gl.blendFunc(gl.SRC_ALPHA, gl.ONE_MINUS_SRC_ALPHA);
319
320 modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
321 modelMatrix = mult(trballMatrix, modelMatrix);
322 modelViewMatrix = mult(viewMatrix, modelMatrix);
323 gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
324 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
325 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
326
327 modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
328 modelMatrix = mult(trballMatrix, modelMatrix);
329 modelViewMatrix = mult(viewMatrix, modelMatrix);
330 gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
331 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
332 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
333
334 gl.disable(gl.BLEND);
335 gl.enable(gl.DEPTH_TEST);
336
337
338 requestAnimationFrame(render);
339 }
340
341 function generateTexCube() {
```

Collision Detection

```
8
9  var eye = vec3(0, 3, 3
10  |   );
11  var at = vec3(0, 0, 0);
12  const up = vec3(0, 1, 0);
13  var cameraVec = vec3(0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)
14
15  var theta = 0;
16  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
17  var vertCubeStart, vertCubeEnd, vertHexaStart, vertHexaEnd, vertGroundStart, vertGroundEnd;
18
19  var posObjects = [
20  |   vec2(-3, -5),   vec2(3, -5),
21  |   vec2(-3, -2),   vec2(3, -2),
22  |   vec2(-3, 1),    vec2(3, 1),
23  |   vec2(-3, 4),    vec2(3, 4),
24  | ];
25  function detectCollision(newPosX, newPosZ) {
26  |   for(var index=0; index<posObjects.length; index++) {
27  |       if( Math.abs(newPosX - posObjects[index][0]) < 1.0 && Math.abs(newPosZ - posObjects[index][1]) < 1.0 )
28  |           return true;
29  |   }
30  |   return false;
31  | };
32
33  window.onload = function init()
```

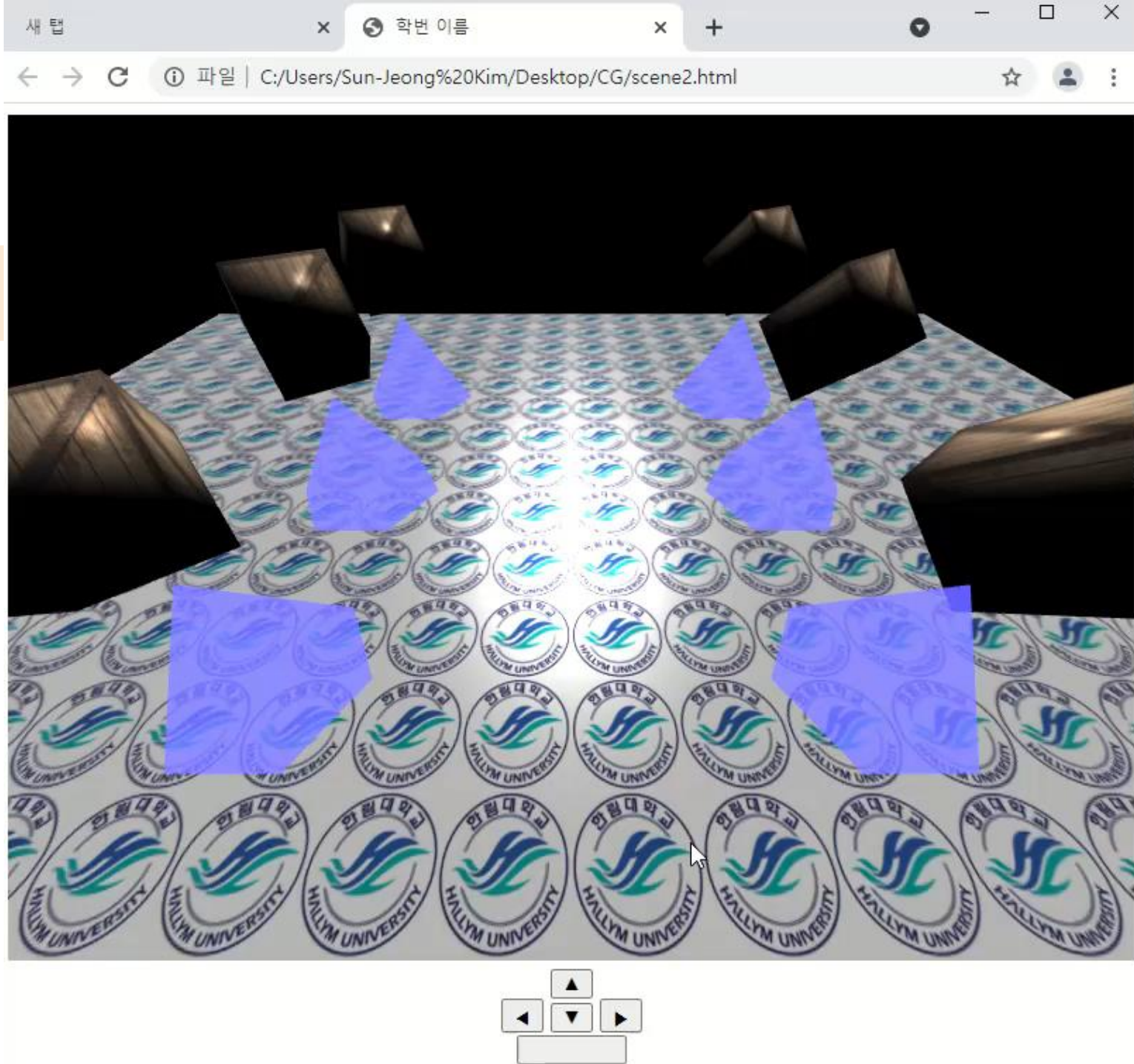
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
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
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2
```

scene2.html JS scene2.js

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init > onclick

```
188
189 // Event listeners for buttons
190 var sinTheta = Math.sin(0.1);
191 var cosTheta = Math.cos(0.1);
192 document.getElementById("left").onclick = function () {
193     var newVecX = cosTheta*cameraVec[0] + sinTheta*cameraVec[2];
194     var newVecZ = -sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
195     cameraVec[0] = newVecX;
196     cameraVec[2] = newVecZ;
197 };
198 document.getElementById("right").onclick = function () {
199     var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
200     var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
201     cameraVec[0] = newVecX;
202     cameraVec[2] = newVecZ;
203 };
204 document.getElementById("up").onclick = function () {
205     var newPosX = eye[0] + 0.5 * cameraVec[0];
206     var newPosZ = eye[2] + 0.5 * cameraVec[2];
207     if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 && !detectCollision(newPosX, newPosZ)) {
208         eye[0] = newPosX;
209         eye[2] = newPosZ;
210     }
211 };
212 document.getElementById("down").onclick = function () {
213     var newPosX = eye[0] - 0.5 * cameraVec[0];
214     var newPosZ = eye[2] - 0.5 * cameraVec[2];
215     if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 && !detectCollision(newPosX, newPosZ)) {
216         eye[0] = newPosX;
217         eye[2] = newPosZ;
218     }
219 };
220
```

연습 문제

- 카메라 위치와 Vertex Position 사이의 거리를 Blending Factor로 하여 Fog 색상을 Blending 하는 shader를 작성하시오.

scene3.html × JS scene3.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene3.html > html > head > script#phongFS

```
40
41 <script id="phongFS" type="x-shader/x-fragment">
42     precision mediump float;
43
44     varying vec3 fNormal, fPosition;
45
46     uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
47     uniform float shininess;
48
49     void main() {
50         vec3 N = normalize(fNormal);
51         vec3 L = normalize(lightPos.xyz);
52         float kd = max(dot(L, N), 0.0);
53         vec4 diffuse = kd * diffuseProduct;
54
55         float fogDepth = -fPosition.z; // the camera looking in the -z direction
56         float fogFactor = smoothstep(2.0, 10.0, fogDepth);
57
58         vec3 V = normalize(fPosition); // origin: camera position
59         vec3 H = normalize(L - V);
60         float ks = pow(max(dot(N, H), 0.0), shininess);
61         vec4 specular = ks * specularProduct;
62
63         if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
64
65         vec4 color = ambientProduct + diffuse + specular;
66         gl_FragColor = mix(color, vec4(0.0, 0.0, 0.0, 1.0), fogFactor);
67         gl_FragColor.a = 1.0;
68     }
69 </script>
70
71 <script id="texMapVS" type="x-shader/x-vertex">
72     attribute vec4 vPosition;
```





scene3.html × JS scene3.js



C: > Users > Sun-Jeong Kim > Desktop > CG > <> scene3.html > html > head > script#texMapFS

```
91
92 <script id="texMapFS" type="x-shader/x-fragment">
93     precision mediump float;
94
95     varying vec3 fNormal, fPosition;
96     varying vec2 fTexCoord;
97
98     uniform sampler2D texture;
99     uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
100     uniform float shininess;
101
102     void main() {
103         vec3 N = normalize(fNormal);
104         vec3 L = normalize(lightPos.xyz);
105         float kd = max(dot(L, N), 0.0);
106         vec4 diffuse = kd * diffuseProduct;
107
108         float fogDepth = -fPosition.z; // the camera looking in the -z direction
109         float fogFactor = smoothstep(2.0, 10.0, fogDepth);
110
111         vec3 V = normalize(fPosition); // origin: camera position
112         vec3 H = normalize(L - V);
113         float ks = pow(max(dot(N, H), 0.0), shininess);
114         vec4 specular = ks * specularProduct;
115
116         if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
117
118         vec4 color = (ambientProduct + diffuse + specular) * texture2D(texture, fTexCoord);
119         gl_FragColor = mix(color, vec4(0.0, 0.0, 0.0, 1.0), fogFactor);
120         gl_FragColor.a = 1.0;
121     }
122 </script>
123
```



scene3.html JS scene3.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene3.js > render > diffuseProductLoc

```
285
286 //var colorLoc = gl.getUniformLocation(program0, "uColor");
287 var diffuseProductLoc = gl.getUniformLocation(program1, "diffuseProduct");
288
289 // draw the ground
290 //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
291 //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
292 gl.useProgram(program2);
293 gl.uniform1i(gl.getUniformLocation(program2, "texture"), 0);
294
295 modelViewMatrix = mult(viewMatrix, trballMatrix);
296 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
297 //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
298 gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
299 gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);
300
301 for (var z=-5; z<5; z+=3) {
302     // draw a cube
303     //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
304     //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
305     gl.useProgram(program2);
306     gl.uniform1i(gl.getUniformLocation(program2, "texture"), 1);
307
308     var rMatrix = mult(rotateY(theta), rotateZ(45));
309     var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
310     modelMatrix = mult(trballMatrix, modelMatrix);
311     modelViewMatrix = mult(viewMatrix, modelMatrix);
312     //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
313     //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
314     gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
315     gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);
316
317     modelMatrix = mult(translate(3, 1.3, z), rMatrix);
```

scene3.html

JS scene3.js

C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene3.js > render

```
324
325 // draw a hexa-pyramid
326 //gl.useProgram(program0);
327 //gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 0.5); // translucent blue
328 gl.useProgram(program1);
329 gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);
330
331 //gl.disable(gl.DEPTH_TEST);
332 //gl.enable(gl.BLEND);
333 //gl.blendFunc(gl.SRC_ALPHA, gl.ONE_MINUS_SRC_ALPHA);
334
335 modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
336 modelMatrix = mult(trballMatrix, modelMatrix);
337 modelViewMatrix = mult(viewMatrix, modelMatrix);
338 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
339 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
340 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
341
342 modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
343 modelMatrix = mult(trballMatrix, modelMatrix);
344 modelViewMatrix = mult(viewMatrix, modelMatrix);
345 //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
346 gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
347 gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
348
349 //gl.disable(gl.BLEND);
350 //gl.enable(gl.DEPTH_TEST);
351 }
352
353 requestAnimationFrame(render);
354 }
355
356 function generateTexCube() {
```