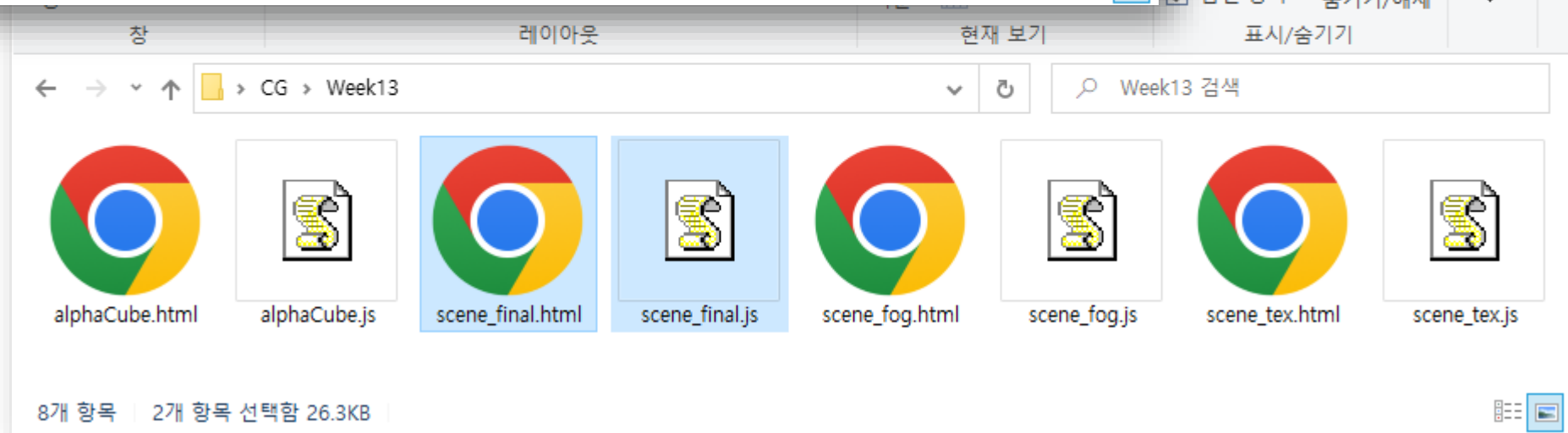
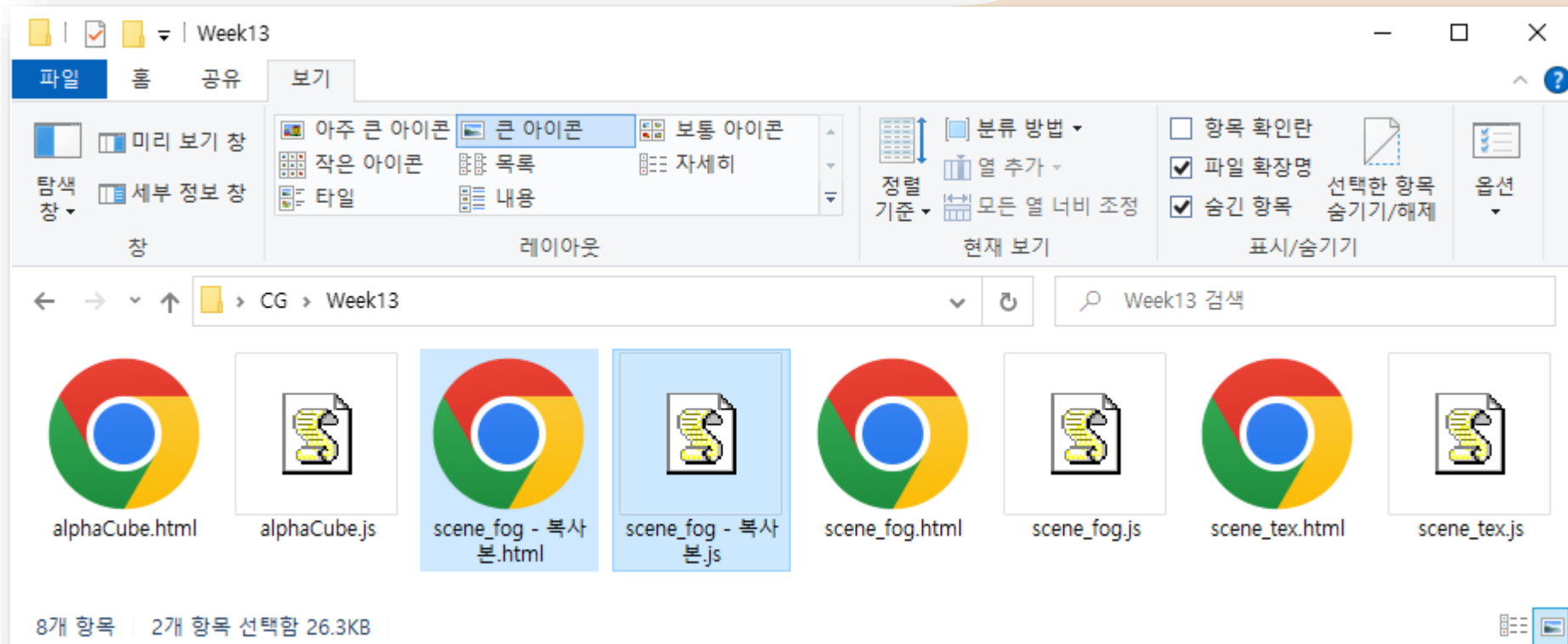


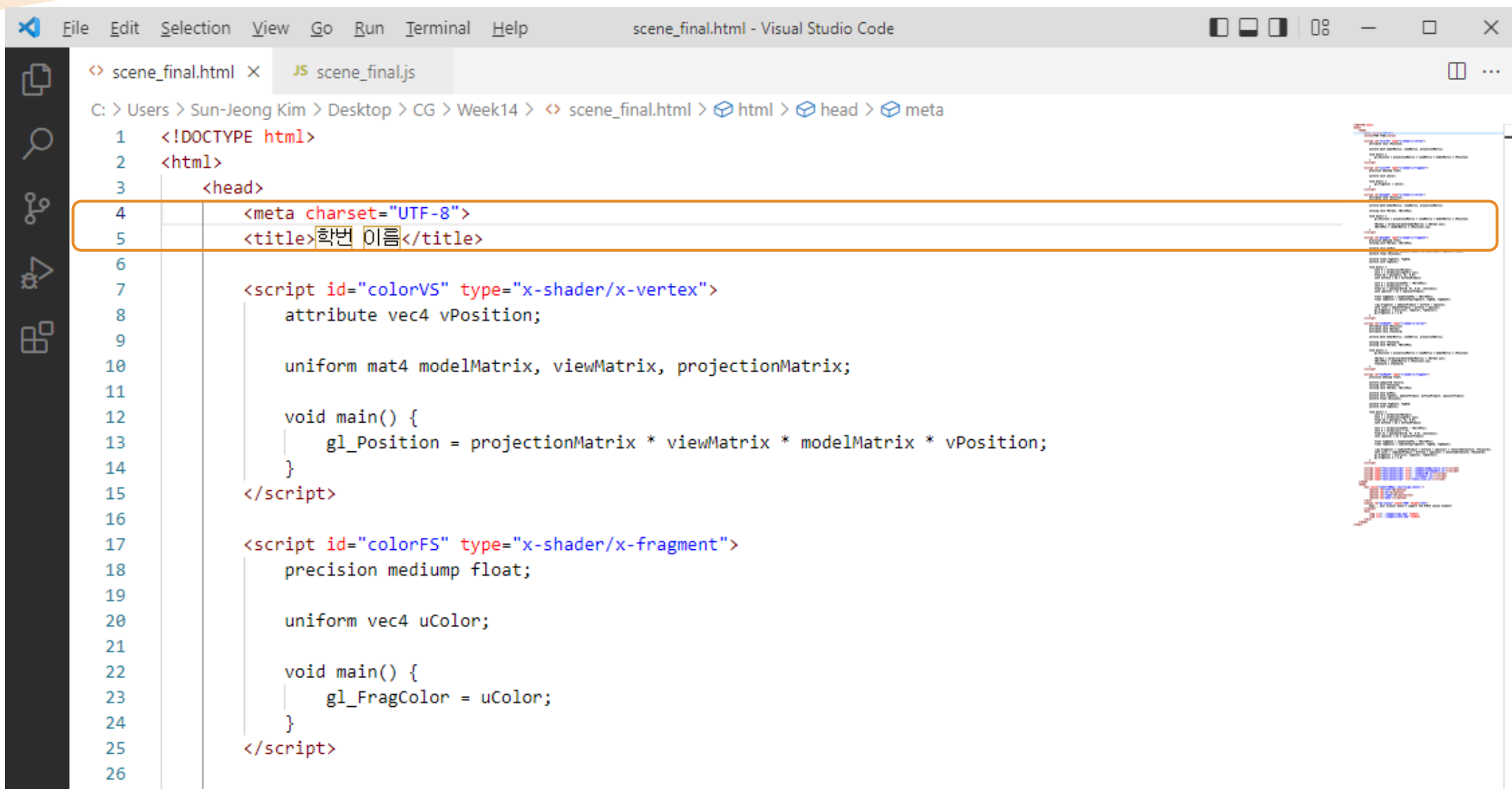
Building a Scene – Final Version

15TH WEEK, 2022





Fog Start/End/Color



```
File Edit Selection View Go Run Terminal Help scene_final.html - Visual Studio Code
scene_final.html x JS scene_final.js
C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > <> scene_final.html > html > head > meta
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>학번 이름</title>
6
7 <script id="colorVS" type="x-shader/x-vertex">
8     attribute vec4 vPosition;
9
10     uniform mat4 modelMatrix, viewMatrix, projectionMatrix;
11
12     void main() {
13         gl_Position = projectionMatrix * viewMatrix * modelMatrix * vPosition;
14     }
15 </script>
16
17 <script id="colorFS" type="x-shader/x-fragment">
18     precision mediump float;
19
20     uniform vec4 uColor;
21
22     void main() {
23         gl_FragColor = uColor;
24     }
25 </script>
26
```

scene_final.html x JS scene_final.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > <> scene_final.html > html > head > meta

```

42
43     <script id="phongFS" type="x-shader/x-fragment">
44         precision mediump float;
45         varying vec3 fNormal, fWorldPos;
46
47         uniform vec3 eyePos;
48         uniform vec4 lightSrc, ambientProduct, diffuseProduct, specularProduct;
49         uniform float shininess;
50
51         uniform float fogStart, fogEnd;
52         uniform vec4 fogColor;
53
54         void main() {
55             vec3 N = normalize(fNormal);
56             vec3 L = normalize(lightSrc.xyz);
57             float kd = max(dot(L, N), 0.0);
58             vec4 diffuse = kd * diffuseProduct;
59
60             vec3 V = normalize(eyePos - fWorldPos);
61             vec3 H = normalize(L + V);
62             float ks = pow(max(dot(N, H), 0.0), shininess);
63             vec4 specular = ks * specularProduct;
64
65             float fogDepth = length(eyePos - fWorldPos);
66             float fogFactor = smoothstep(fogStart, fogEnd, fogDepth);
67
68             //gl_FragColor = ambientProduct + diffuse + specular;
69             vec4 color = ambientProduct + diffuse + specular;
70             gl_FragColor = mix(color, fogColor, fogFactor);
71             gl_FragColor.a = 1.0;
72         }
73     </script>
74
75     <script id="texMapVS" type="x-shader/x-vertex">
76         attribute vec4 vPosition;

```



scene_final.html x JS scene_final.js

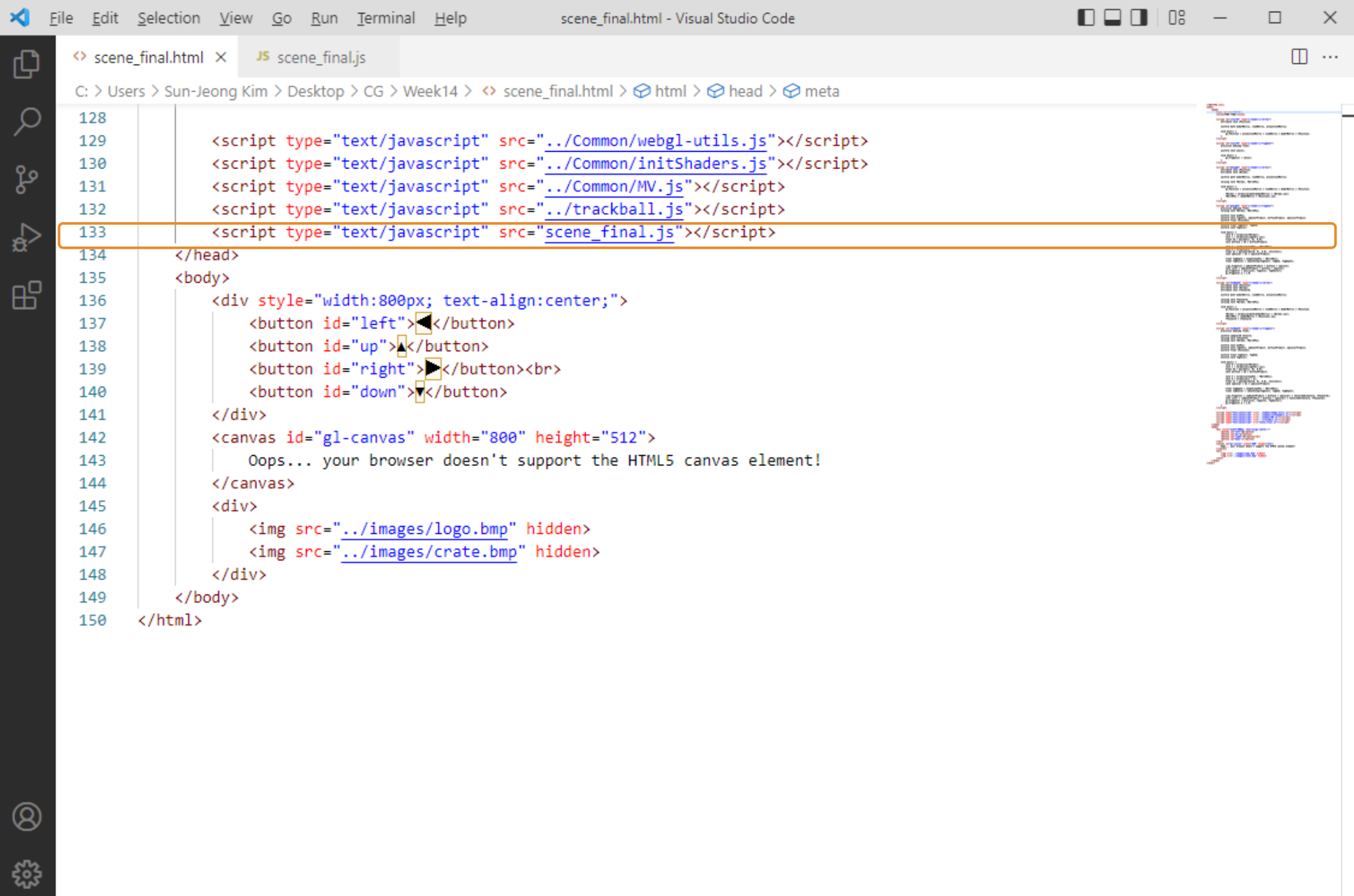
C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > <> scene_final.html > html > head > meta

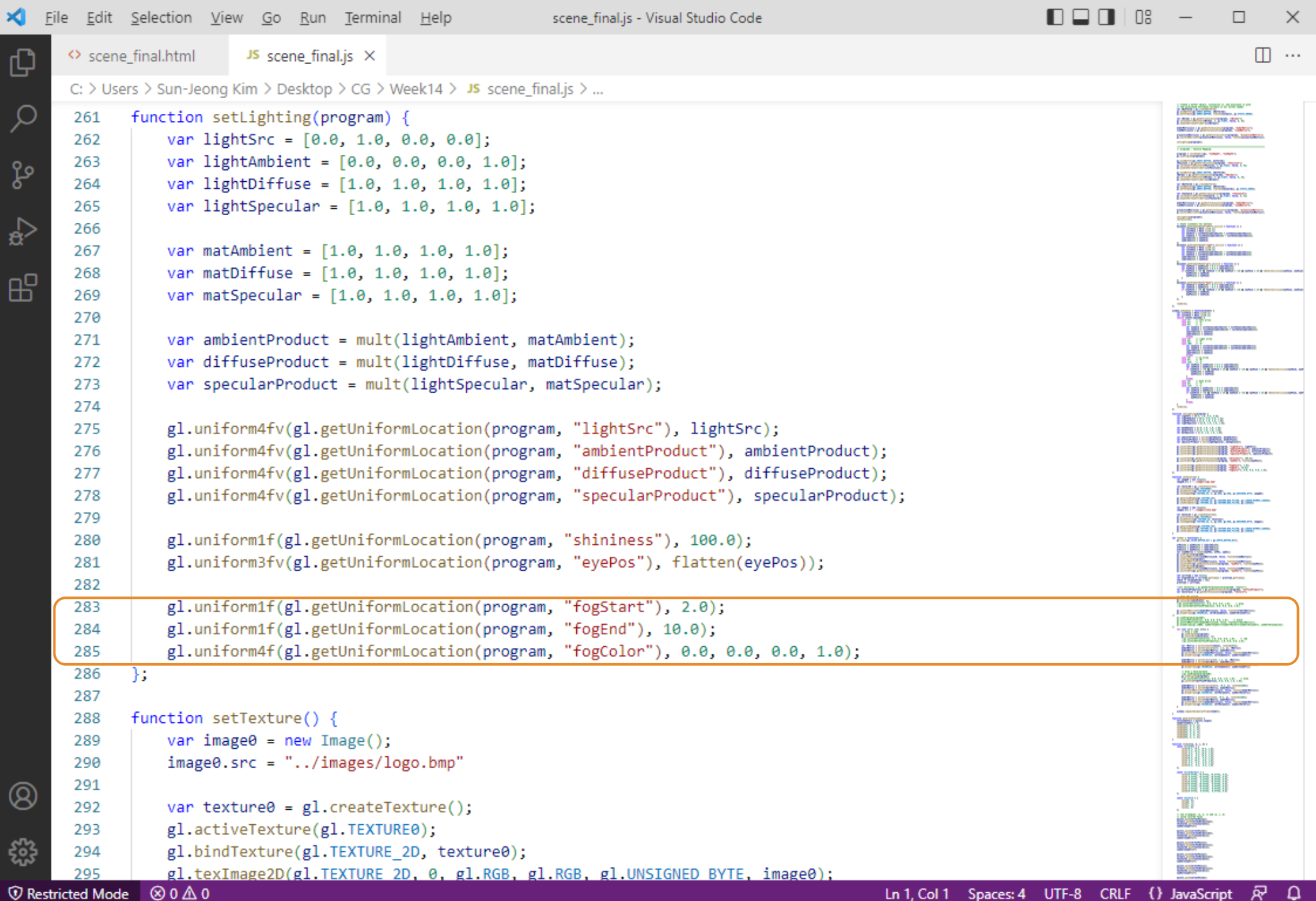
```

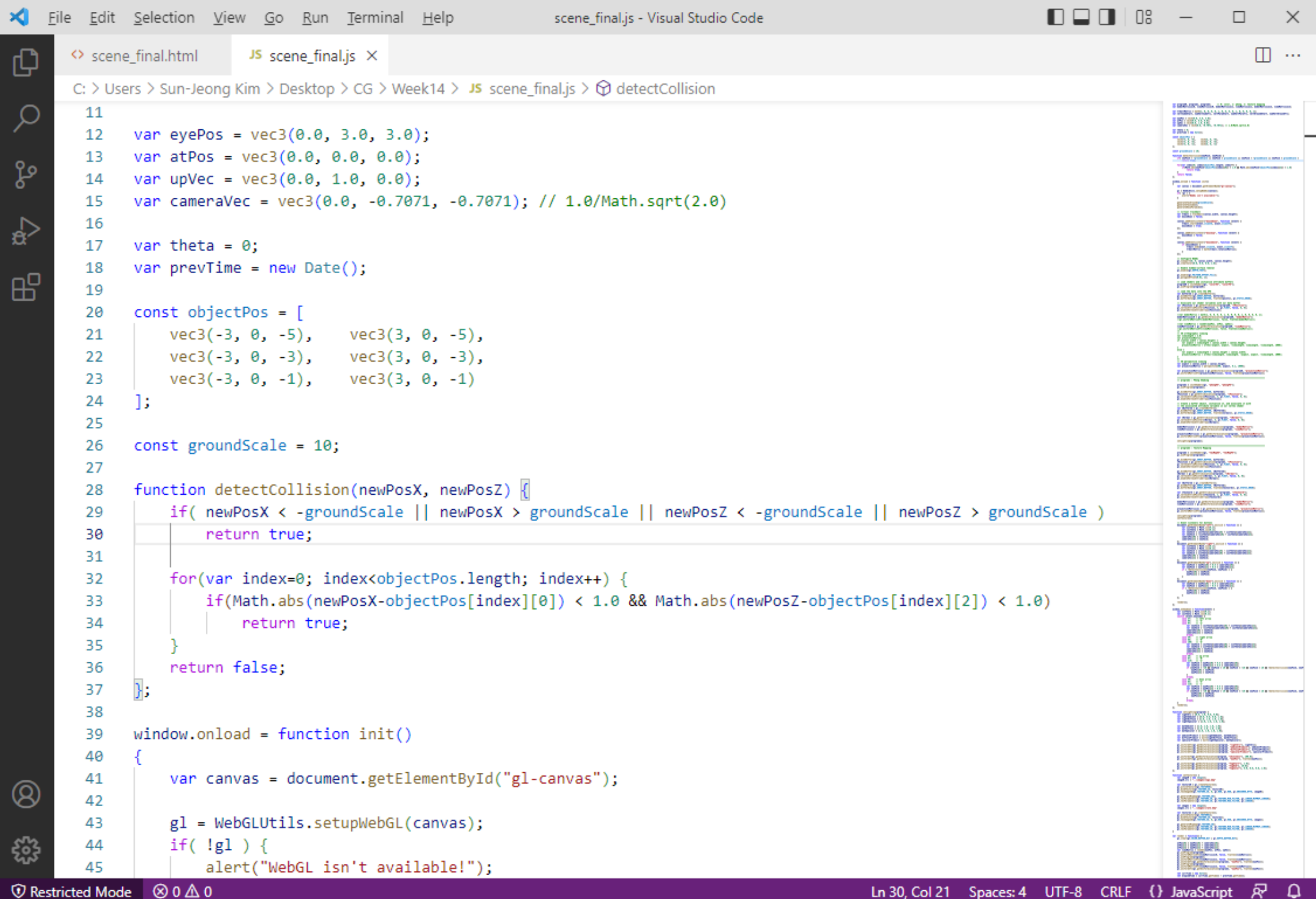
93
94     <script id="texMapFS" type="x-shader/x-fragment">
95         precision mediump float;
96
97         uniform sampler2D texture;
98         varying vec2 fTexCoord;
99         varying vec3 fNormal, fWorldPos;
100
101         uniform vec3 eyePos;
102         uniform vec4 lightSrc, ambientProduct, diffuseProduct, specularProduct;
103         uniform float shininess;
104
105         uniform float fogStart, fogEnd;
106         uniform vec4 fogColor;
107
108         void main() {
109             vec3 N = normalize(fNormal);
110             vec3 L = normalize(lightSrc.xyz);
111             float kd = max(dot(L, N), 0.0);
112             vec4 diffuse = kd * diffuseProduct;
113
114             vec3 V = normalize(eyePos - fWorldPos);
115             vec3 H = normalize(L + V);
116             float ks = pow(max(dot(N, H), 0.0), shininess);
117             vec4 specular = ks * specularProduct;
118
119             float fogDepth = length(eyePos - fWorldPos);
120             float fogFactor = smoothstep(fogStart, fogEnd, fogDepth);
121
122             //gl_FragColor = (ambientProduct + diffuse + specular) * texture2D(texture, fTexCoord);
123             vec4 color = (ambientProduct + diffuse + specular) * texture2D(texture, fTexCoord);
124             gl_FragColor = mix(color, fogColor, fogFactor);
125             gl_FragColor.a = 1.0;
126         }
127     </script>

```









Collision Detection


```
25
26 const groundScale = 10;
27
28 function detectCollision(newPosX, newPosZ) {
29     if( newPosX < -groundScale || newPosX > groundScale || newPosZ < -groundScale || newPosZ > groundScale )
30         return true;
31
32     for(var index=0; index<objectPos.length; index++) {
33         if(Math.abs(newPosX-objectPos[index][0]) < 1.0 && Math.abs(newPosZ-objectPos[index][2]) < 1.0)
34             return true;
35     }
36     return false;
37 };
38
39 window.onload = function init()
40 {
41     var canvas = document.getElementById("gl-canvas");
42
43     gl = WebGLUtils.setupWebGL(canvas);
44     if( !gl ) {
45         alert("WebGL isn't available!");
46     }
47
48     generateTexGround(groundScale);
49     generateTexCube();
50     generateHexaPyramid();
51
52     // virtual trackball
53     var trball = trackball(canvas.width, canvas.height);
```

[Downloaded from ascelibrary.org by University of California, San Diego on 06/01/15. Copyright ASCE. For personal use only; all rights reserved.](#)

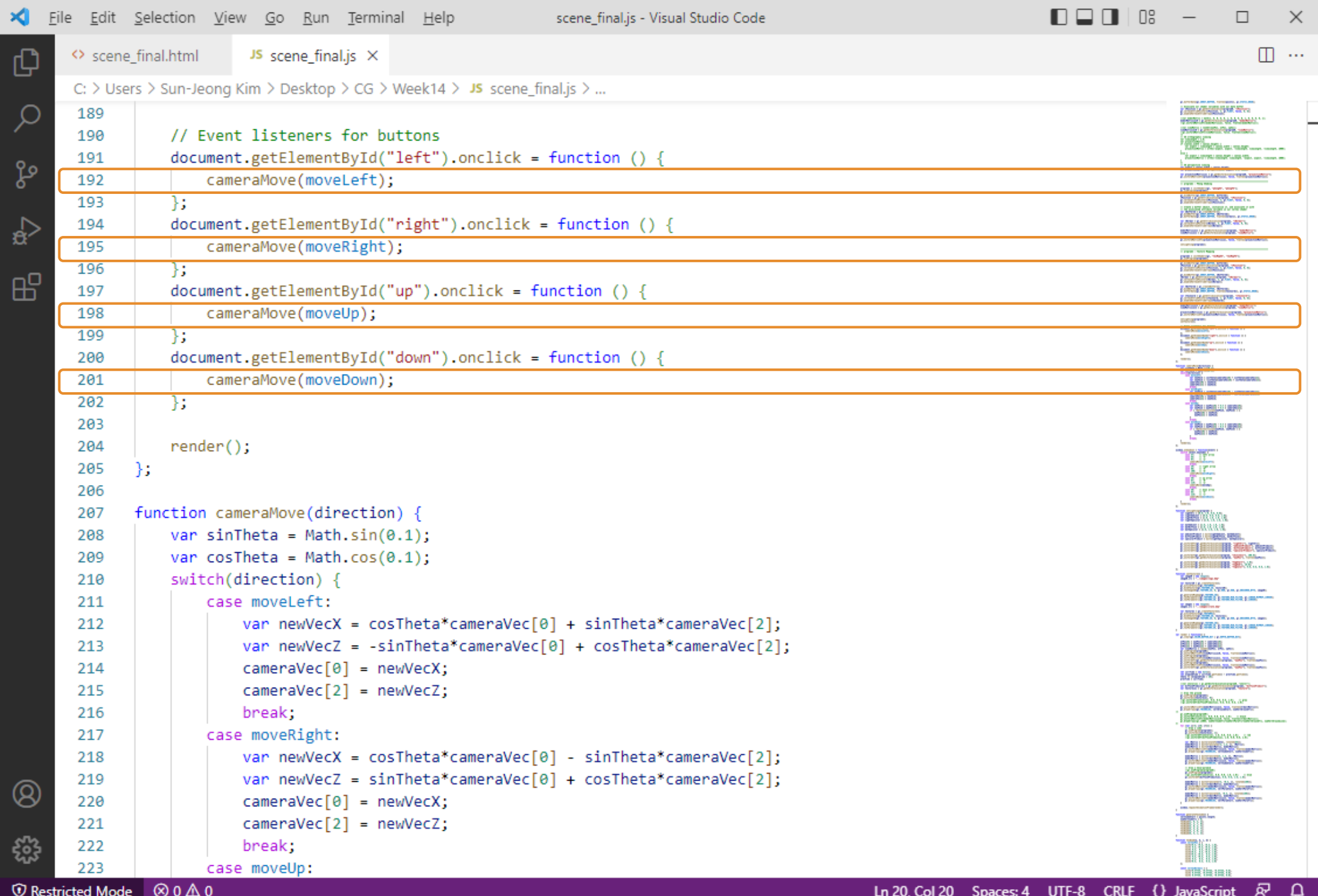
[illegible]

```
File Edit Selection View Go Run Terminal Help scene_final.js - Visual Studio Code
scene_final.html JS scene_final.js X
C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > JS scene_final.js > onkeydown
231 cameraVec[0] = newVecX;
232 cameraVec[2] = newVecZ;
233 break;
234 case 39: // right arrow
235 case 68: // 'D'
236 case 100: // 'd'
237     var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
238     var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
239     cameraVec[0] = newVecX;
240     cameraVec[2] = newVecZ;
241     break;
242 case 38: // up arrow
243 case 87: // 'W'
244 case 119: // 'w'
245     var newPosX = eyePos[0] + 0.5 * cameraVec[0];
246     var newPosZ = eyePos[2] + 0.5 * cameraVec[2];
247     if ( !detectCollision(newPosX, newPosZ) ) {
248         eyePos[0] = newPosX;
249         eyePos[2] = newPosZ;
250     }
251     break;
252 case 40: // down arrow
253 case 83: // 'S'
254 case 115: // 's'
255     var newPosX = eyePos[0] - 0.5 * cameraVec[0];
256     var newPosZ = eyePos[2] - 0.5 * cameraVec[2];
257     if ( !detectCollision(newPosX, newPosZ) ) {
258         eyePos[0] = newPosX;
259         eyePos[2] = newPosZ;
260     }
261     break;
262 }
263 render();
264 };
265
```

Camera Moving



```
File Edit Selection View Go Run Terminal Help scene_final.js - Visual Studio Code
scene_final.html JS scene_final.js X
C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > JS scene_final.js > ...
1  var gl;
2  var points = [];
3  var normals = [];
4  var texCoords = [];
5
6  var program0, program1, program2;    // 0: color, 1: phong, 2: texture mapping
7  var modelMatrixLoc0, viewMatrixLoc0, modelMatrixLoc1, viewMatrixLoc1, modelMatrixLoc2, viewMatrixLoc2;
8
9  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
10 var vertCubeStart, numVertCubeTri, vertPyraStart, numVertPyraTri, vertGroundStart, numVertGroundTri;
11
12 var eyePos = vec3(0.0, 3.0, 3.0);
13 var atPos = vec3(0.0, 0.0, 0.0);
14 var upVec = vec3(0.0, 1.0, 0.0);
15 var cameraVec = vec3(0.0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)
16
17 const moveLeft = 0;
18 const moveRight = 1
19 const moveUp = 2;
20 const moveDown = 3;
21
22 var theta = 0;
23 var prevTime = new Date();
24
25 const objectPos = [
26     vec3(-3, 0, -5),    vec3(3, 0, -5),
```



The diagram illustrates a 1000-bit data stream, organized into 100 groups of 10 bits each. Each group is represented by a horizontal bar with a unique color and a label. The labels include group numbers (e.g., 1, 2, 3, ..., 100) and specific bit patterns (e.g., 0000000000, 0000000001, ..., 0000000010). The diagram is divided into sections by vertical lines, with some sections containing multiple groups. The overall structure is a long, narrow rectangle with a grid-like pattern of colored bars and text labels.

[illegible][illegible]

Jumping

```

7  var modelMatrixLoc0, viewMatrixLoc0, modelMatrixLoc1, viewMatrixLoc1, modelMatrixLoc2, viewMatrixLoc2;
8
9  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
10 var vertCubeStart, numVertCubeTri, vertPyraStart, numVertPyraTri, vertGroundStart, numVertGroundTri;
11
12 var eyePos = vec3(0.0, 3.0, 3.0);
13 var atPos = vec3(0.0, 0.0, 0.0);
14 var upVec = vec3(0.0, 1.0, 0.0);
15 var cameraVec = vec3(0.0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)
16
17 const moveLeft = 0;
18 const moveRight = 1
19 const moveUp = 2;
20 const moveDown = 3;
21
22 var theta = 0;
23 var prevTime = new Date();
24
25 var jumping = false;
26
27 const objectPos = [
28     vec3(-3, 0, -5),    vec3(3, 0, -5),
29     vec3(-3, 0, -3),    vec3(3, 0, -3),
30     vec3(-3, 0, -1),    vec3(3, 0, -1)
31 ];
32
33 const groundScale = 10;
34
35 function detectCollision(newPosX, newPosZ) {

```

a) Experimental design

Experimental design

1. Selection of participants (N = 20)

2. Random assignment to two groups (Control and Experimental)

3. Baseline measurement (Pre-test)

4. Intervention (8 weeks)

5. Post-intervention measurement (Post-test)

6. Follow-up measurement (3 months post-intervention)

b) Data analysis

Data analysis

1. Descriptive statistics

2. Inferential statistics (t-test, ANOVA)

3. Effect size calculation

c) Results

Results

1. Descriptive statistics

2. Inferential statistics (t-test, ANOVA)

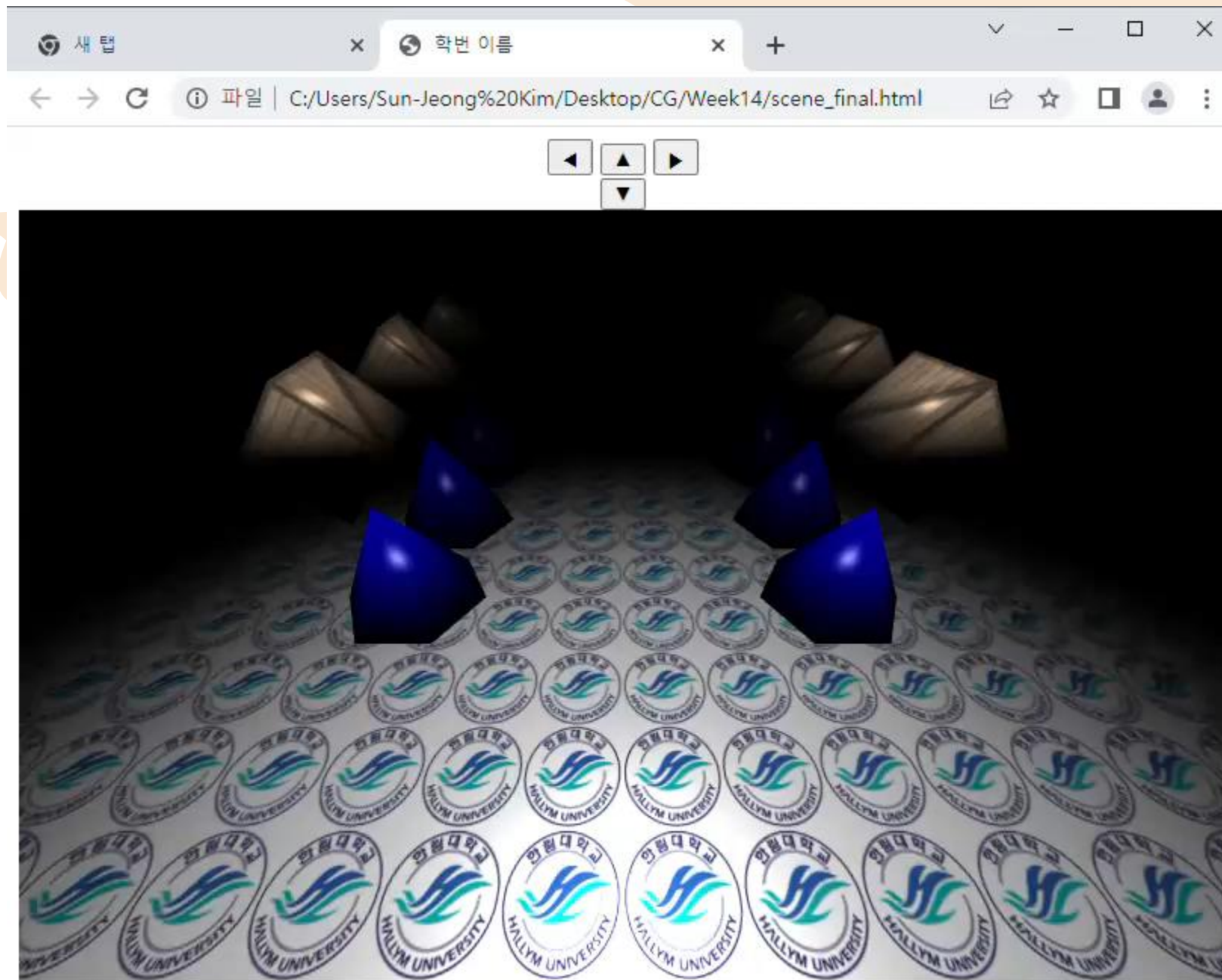
3. Effect size calculation

4. Conclusions

scene_final.html JS scene_final.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > JS scene_final.js > render

```
328
329 var render = function() {
330     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
331
332     let currTime = new Date();
333     let elapsedTime = currTime.getTime() - prevTime.getTime();
334     theta += (elapsedTime / 10);
335     prevTime = currTime;
336
337     if( jumping ) {
338         if( eyePos[1] > 3.0 ) {
339             eyePos[1] -= elapsedTime/1000;
340             if( eyePos[1] < 3.0 ) {
341                 eyePos[1] = 3.0;
342             }
343         }
344         else {
345             jumping = false;
346         }
347     }
348
349     atPos[0] = eyePos[0] + cameraVec[0];
350     atPos[1] = eyePos[1] + cameraVec[1];
351     atPos[2] = eyePos[2] + cameraVec[2];
352     var viewMatrix = lookAt(eyePos, atPos, upVec);
353     gl.useProgram(program0);
354     gl.uniformMatrix4fv(viewMatrixLoc0, false, flatten(viewMatrix));
355     gl.useProgram(program1);
356     gl.uniformMatrix4fv(viewMatrixLoc1, false, flatten(viewMatrix));
357     gl.uniform3fv(gl.getUniformLocation(program1, "eyePos"), flatten(eyePos));
358     gl.useProgram(program2);
359     gl.uniformMatrix4fv(viewMatrixLoc2, false, flatten(viewMatrix));
360     gl.uniform3fv(gl.getUniformLocation(program2, "eyePos"), flatten(eyePos));
361
362     //var uColorLoc = gl.getUniformLocation(program0, "uColor");
```



Moving Obstacles

```
7 var modelMatrixLoc0, viewMatrixLoc0, modelMatrixLoc1, viewMatrixLoc1, modelMatrixLoc2, viewMatrixLoc2;
8
9 var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
10 var vertCubeStart, numVertCubeTri, vertPyraStart, numVertPyraTri, vertGroundStart, numVertGroundTri;
11
12 var eyePos = vec3(0.0, 3.0, 3.0);
13 var atPos = vec3(0.0, 0.0, 0.0);
14 var upVec = vec3(0.0, 1.0, 0.0);
15 var cameraVec = vec3(0.0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)
16
17 const moveLeft = 0;
18 const moveRight = 1;
19 const moveUp = 2;
20 const moveDown = 3;
21
22 var theta = 0;
23 var prevTime = new Date();
24
25 var jumping = false;
26
27 var objectPos = [];
28 var prevXPos = 3;
29 var objectDirection = true;
30
31 const groundScale = 10;
32
33 function detectCollision(newPosX, newPosZ) {
34     if( newPosX < -groundScale || newPosX > groundScale || newPosZ < -groundScale || newPosZ > groundScale )
35         return true;
```

```
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 // ...
```


scene_final.html JS scene_final.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > JS scene_final.js > render

```
363
364 // draw the ground
365 gl.useProgram(program2);
366 gl.uniform1i(textureLoc, 0);
367 //gl.uniform4f(uColorLoc, 0.8, 0.8, 0.8, 1.0); // gray
368 //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
369
370 gl.uniformMatrix4fv(modelMatrixLoc2, false, flatten(trballMatrix));
371 gl.drawArrays(gl.TRIANGLES, vertGroundStart, numVertGroundTri);
372
373 /*
374 gl.useProgram(program0);
375 gl.uniform4f(uColorLoc, 0.0, 0.0, 0.0, 1.0); // black
376 gl.uniformMatrix4fv(modelMatrixLoc0, false, flatten(trballMatrix));
377 gl.drawArrays(gl.LINES, numVertCubeTri+numVertPyraTri+numVertGroundTri, numVertGroundLine);
378 */
379
380 objectPos = [];
381 var xPos;
382 if( objectDirection )
383 |   xPos = prevXPos + elapsedTime/1000;
384 else
385 |   xPos = prevXPos - elapsedTime/1000;
386 if( xPos > (groundScale - 1) )
387 |   objectDirection = false;
388 else if( xPos < 1 )
389 |   objectDirection = true;
390 prevXPos = xPos;
391
392 for (var z=-7; z<0; z+=3) {
393 |   // draw a cube
394 |   gl.useProgram(program2);
395 |   gl.uniform1i(textureLoc, 1);
396 |   //gl.uniform4f(uColorLoc, 1.0, 0.0, 0.0, 1.0); // red
397 |   //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
398
399 |   var rMatrix = mult(rotateY(theta), rotateZ(45));
```


File Edit Selection View Go Run Terminal Help

scene_final.js - Visual Studio Code

scene_final.html JS scene_final.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week14 > JS scene_final.js > render

```
397     var rMatrix = mult(rotateY(theta), rotateZ(45));
398     modelMatrix = mult(translate(-xPos, 1.3, z), rMatrix);
399     modelMatrix = mult(trballMatrix, modelMatrix);
400     gl.uniformMatrix4fv(modelMatrixLoc2, false, flatten(modelMatrix));
401     gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
402
403     objectPos.push(vec3(-xPos, 0, z));
404
405     modelMatrix = mult(translate(xPos, 1.3, z), rMatrix);
406     modelMatrix = mult(trballMatrix, modelMatrix);
407     gl.uniformMatrix4fv(modelMatrixLoc2, false, flatten(modelMatrix));
408     gl.drawArrays(gl.TRIANGLES, vertCubeStart, numVertCubeTri);
409
410     objectPos.push(vec3(xPos, 0, z));
411
412     // draw a hexa-pyramid
413     //gl.useProgram(program0);
414     gl.useProgram(program1);
415     //gl.uniform4f(uColorLoc, 0.0, 0.0, 1.0, 1.0);    // blue
416     gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);
417
418     modelMatrix = mult(translate(-xPos, -0.5, z), rotateZ(180));
419     modelMatrix = mult(trballMatrix, modelMatrix);
420     gl.uniformMatrix4fv(modelMatrixLoc1, false, flatten(modelMatrix));
421     gl.drawArrays(gl.TRIANGLES, vertPyraStart, numVertPyraTri);
422
423     modelMatrix = mult(translate(xPos, -0.5, z), rotateZ(180));
424     modelMatrix = mult(trballMatrix, modelMatrix);
425     gl.uniformMatrix4fv(modelMatrixLoc1, false, flatten(modelMatrix));
426     gl.drawArrays(gl.TRIANGLES, vertPyraStart, numVertPyraTri);
427
428
429     window.requestAnimationFrame(render);
430 }
431
```

Ln 410, Col 42 Spaces: 4 UTF-8 CRLF JavaScript

