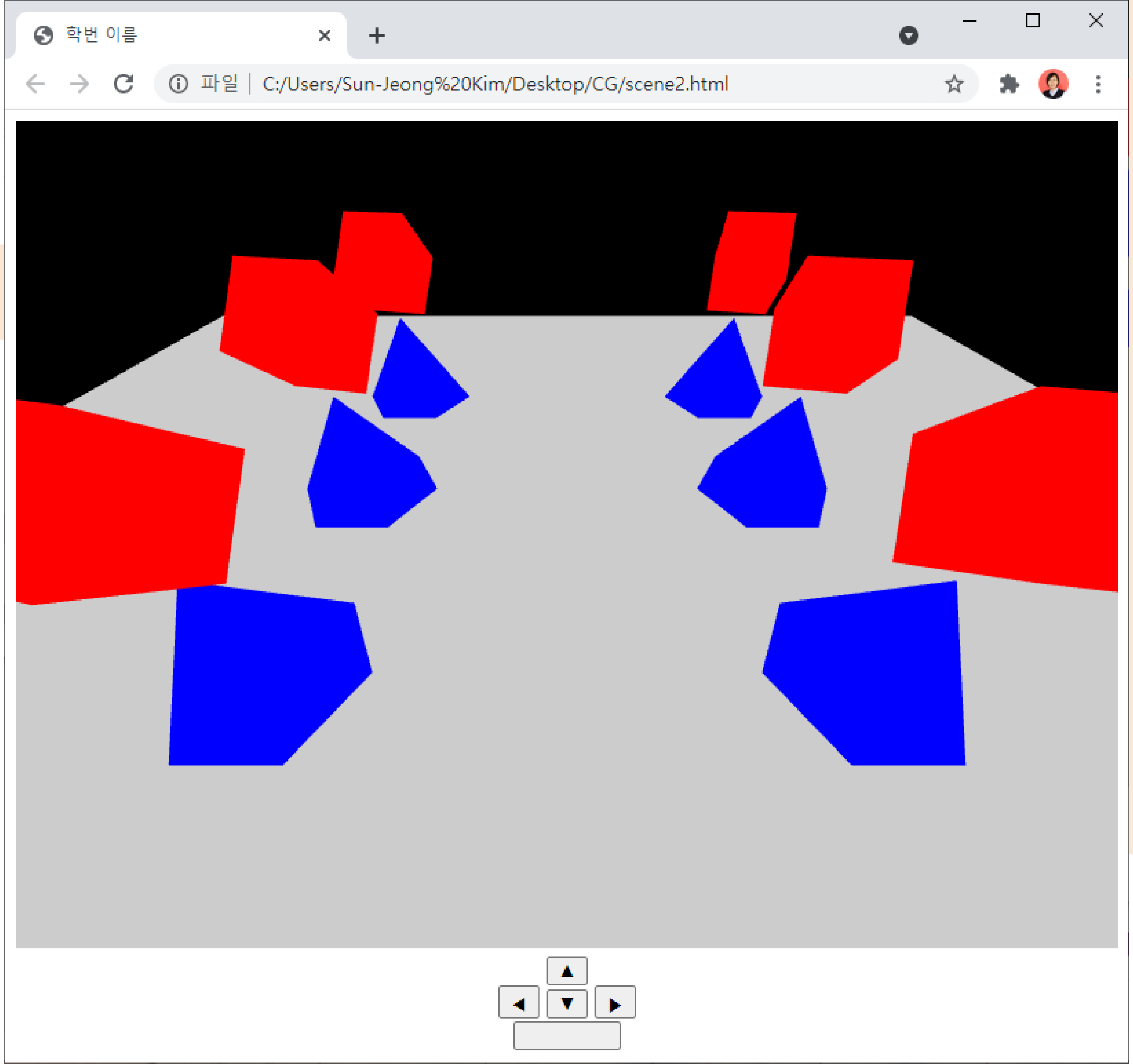
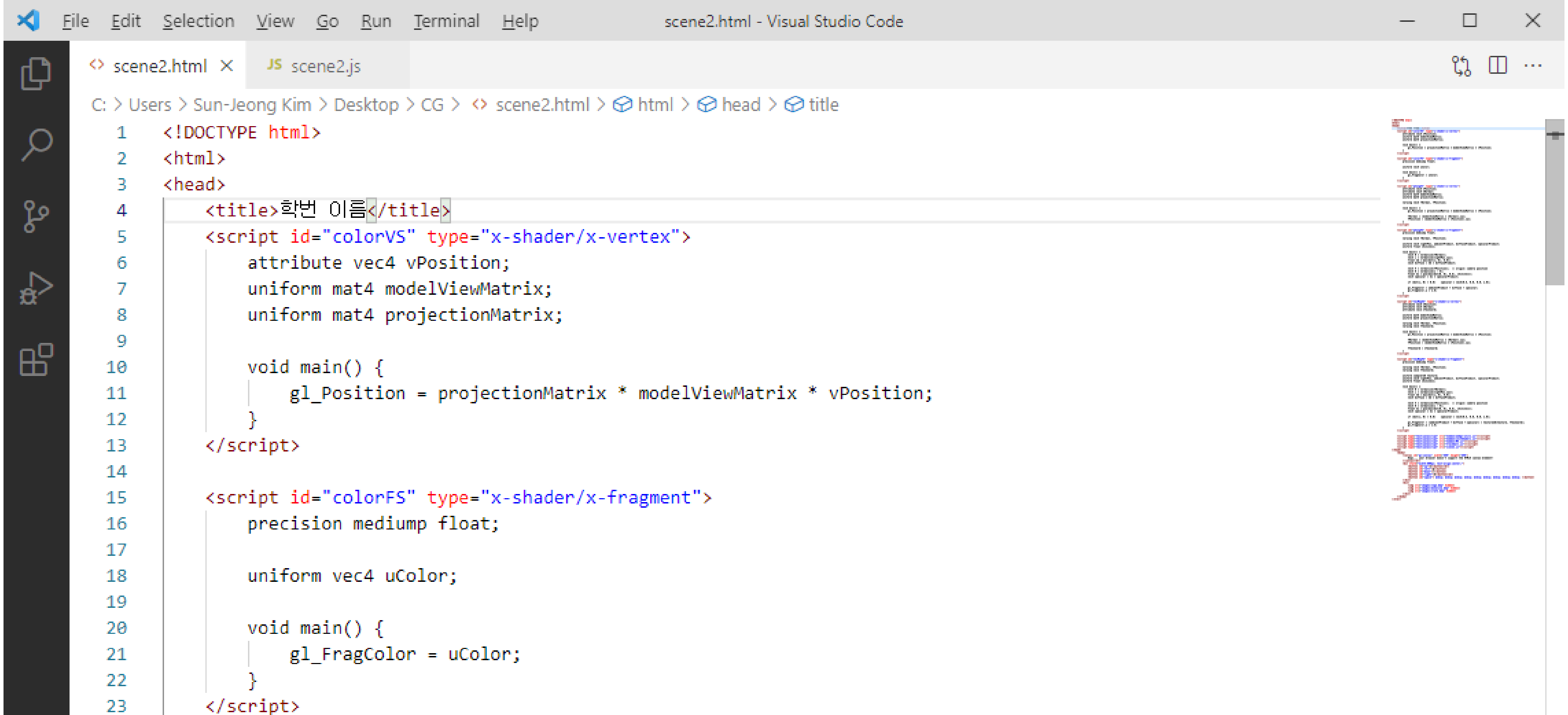
Building a Scene with Texture Mapping

14TH WEEK, 2021

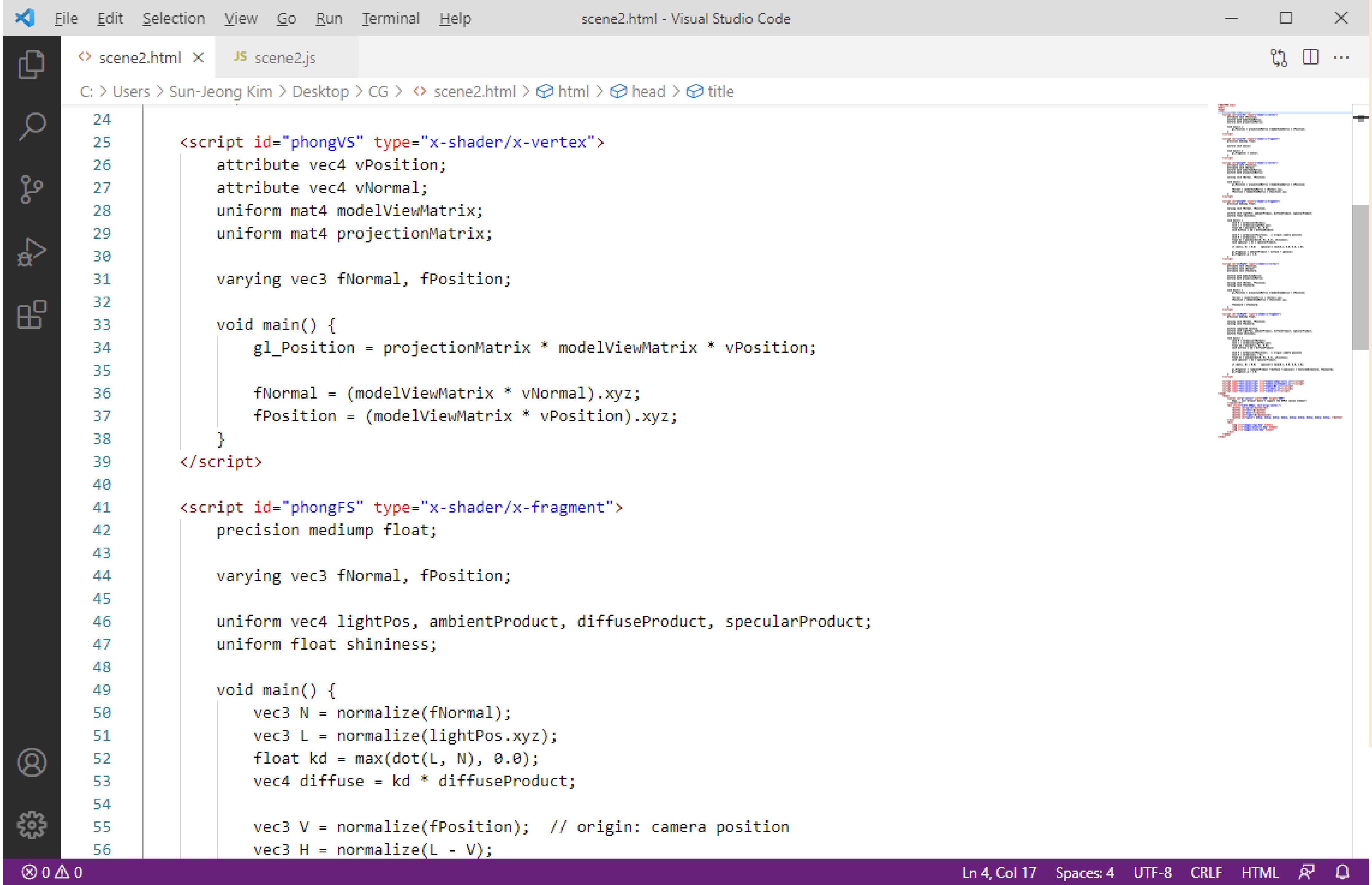

학번 이름
파일 | C:/Users/Sun-Jeong%20Kim/Desktop/CG/scene2.html
▲
◀ ▼ ▶

Multiple Shaders



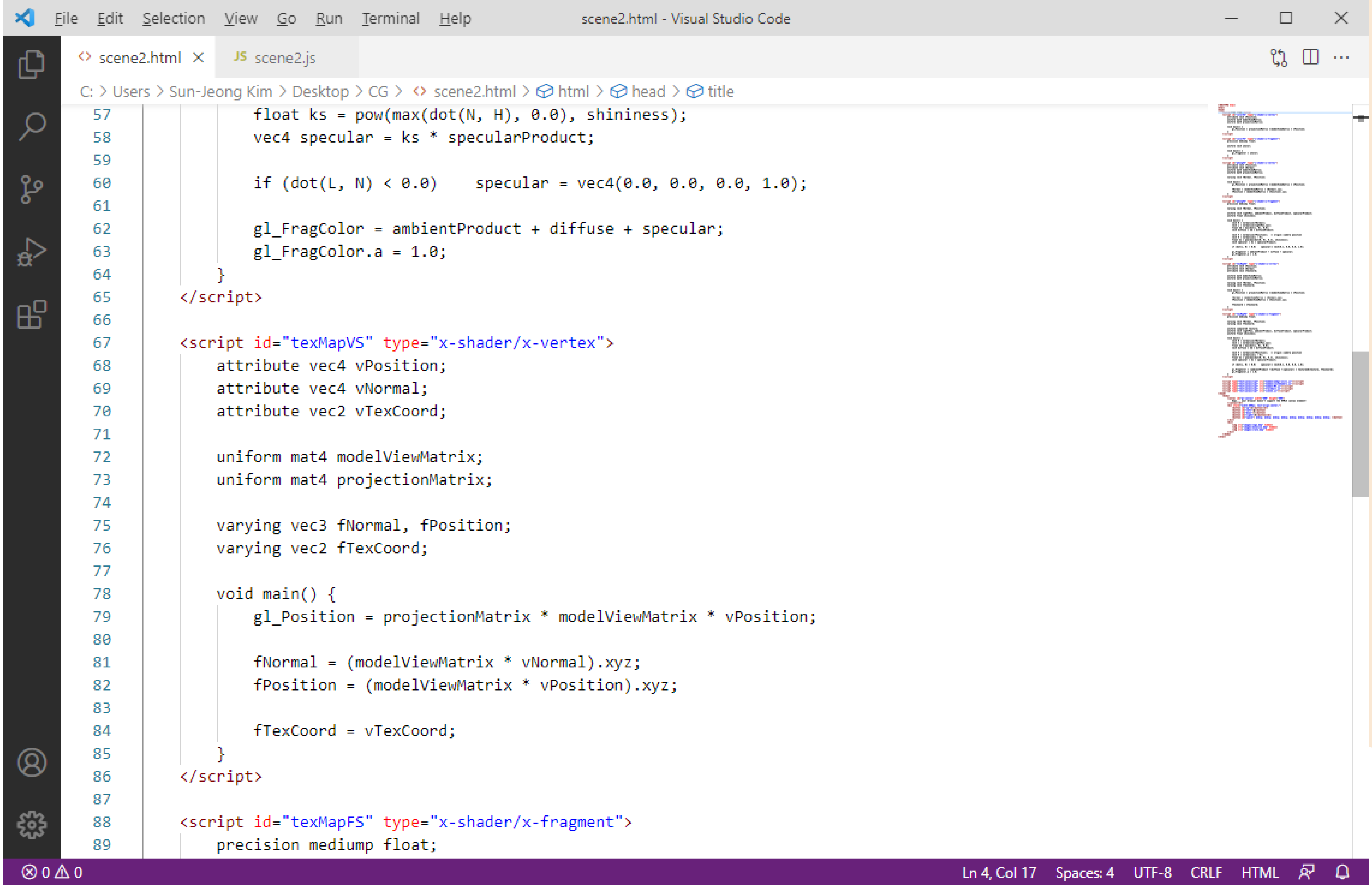
File Edit Selection View Go Run Terminal Help
scene2.html - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.html > html > head > title

```
<!DOCTYPE html>
<html>
<head>
    <title>학번 이름</title>
    <script id="colorVS" type="x-shader/x-vertex">
        attribute vec4 vPosition;
        uniform mat4 modelViewMatrix;
        uniform mat4 projectionMatrix;

        void main() {
            gl_Position = projectionMatrix * modelViewMatrix * vPosition;
        }
    </script>

    <script id="colorFS" type="x-shader/x-fragment">
        precision mediump float;

        uniform vec4 uColor;

        void main() {
            gl_FragColor = uColor;
        }
    </script>
```

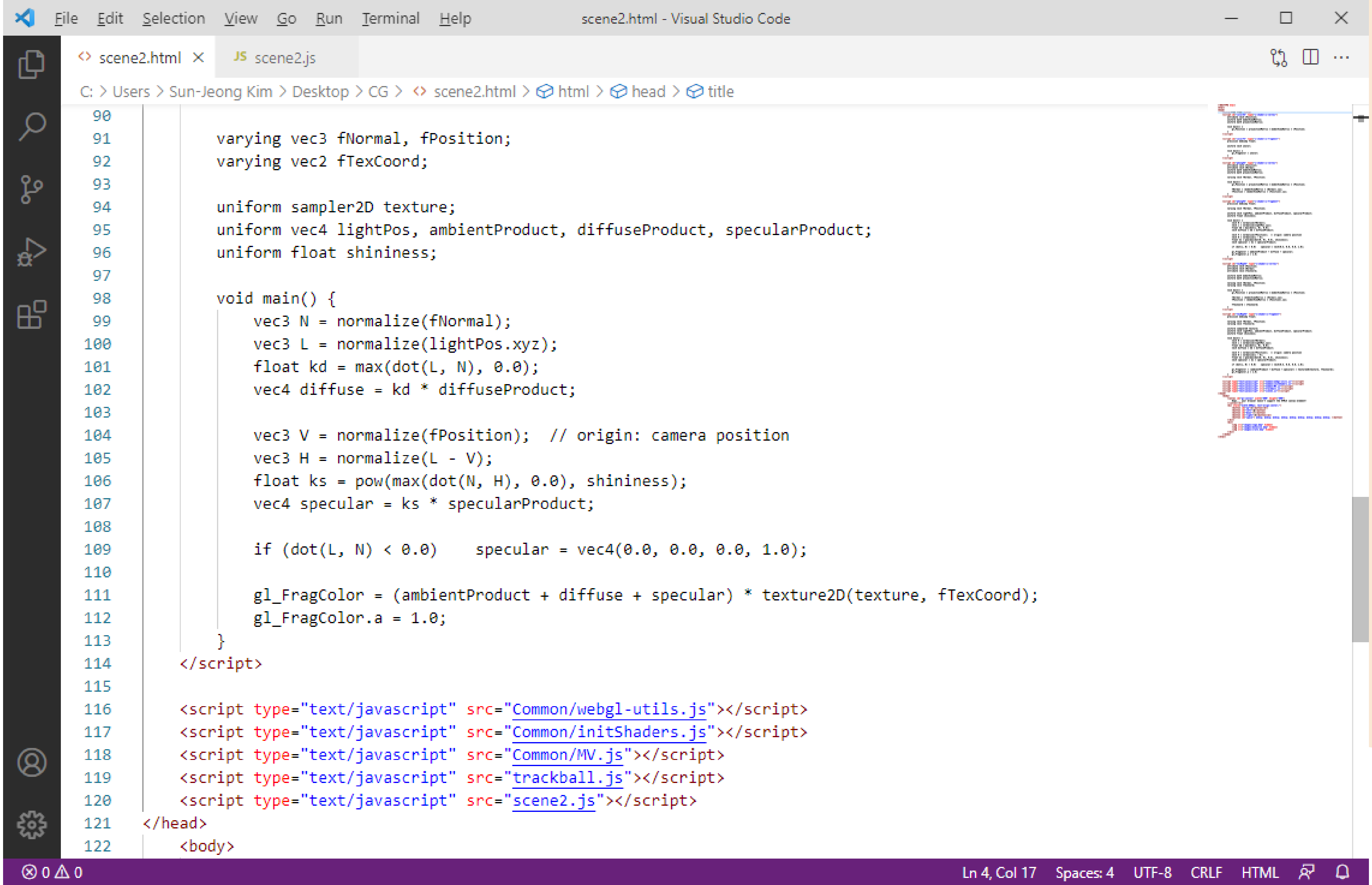


File Edit Selection View Go Run Terminal Help
scene2.html - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.html > html > head > title
⊗ 0 ⚠ 0
Ln 4, Col 17
Spaces: 4
UTF-8
CRLF
HTML

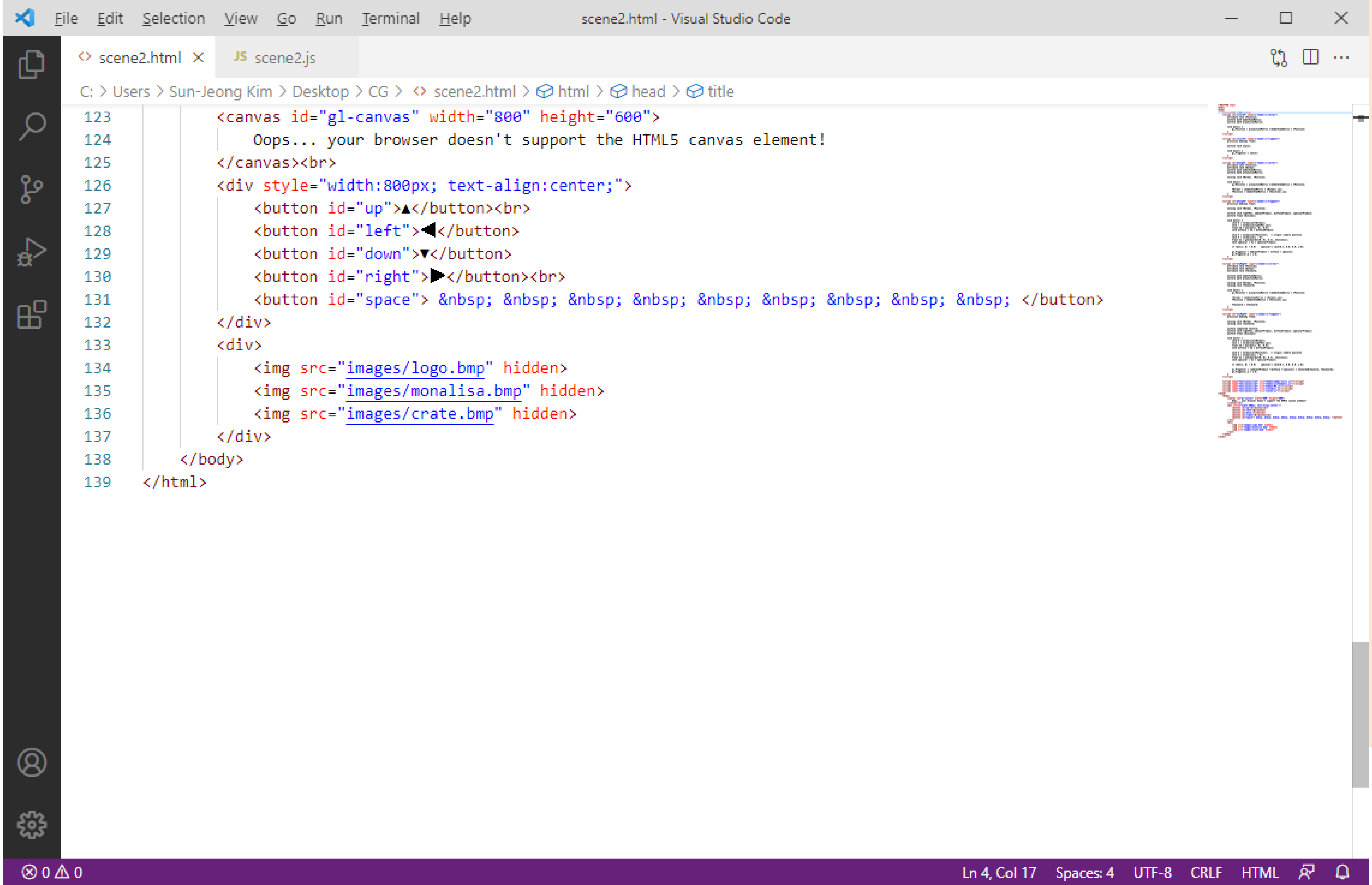
```
    <script id="phongVS" type="x-shader/x-vertex">
        attribute vec4 vPosition;
        attribute vec4 vNormal;
        uniform mat4 modelViewMatrix;
        uniform mat4 projectionMatrix;

        varying vec3 fNormal, fPosition;

        void main() {
            gl_Position = projectionMatrix * modelViewMatrix * vPosition;

            fNormal = (modelViewMatrix * vNormal).xyz;
            fPosition = (modelViewMatrix * vPosition).xyz;
        }
    </script>

    <script id="phongFS" type="x-shader/x-fragment">
        precision mediump float;

        varying vec3 fNormal, fPosition;

        uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
        uniform float shininess;

        void main() {
            vec3 N = normalize(fNormal);
            vec3 L = normalize(lightPos.xyz);
            float kd = max(dot(L, N), 0.0);
            vec4 diffuse = kd * diffuseProduct;

            vec3 V = normalize(fPosition);  // origin: camera position
            vec3 H = normalize(L - V);
```


File
Edit
Selection
View
Go
Run
Terminal
Help
scene2.html - Visual Studio Code
scene2.html
JS scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.html > html > head > title
⊗ 0 ⚠ 0
Ln 4, Col 17
Spaces: 4
UTF-8
CRLF
HTML

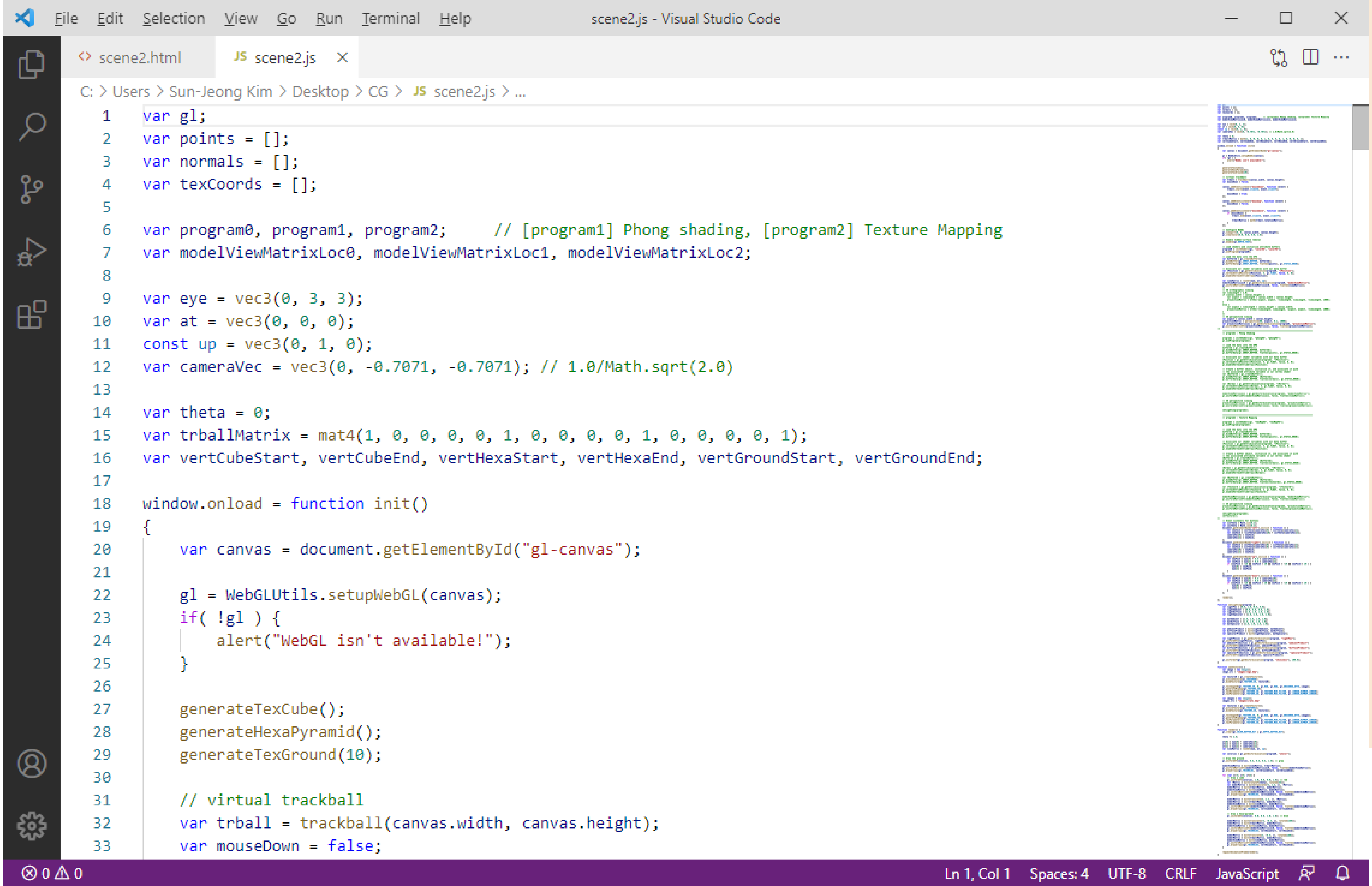
```
            float ks = pow(max(dot(N, H), 0.0), shininess);
            vec4 specular = ks * specularProduct;

            if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);

            gl_FragColor = ambientProduct + diffuse + specular;
            gl_FragColor.a = 1.0;
        }
    </script>

    <script id="texMapVS" type="x-shader/x-vertex">
        attribute vec4 vPosition;
        attribute vec4 vNormal;
        attribute vec2 vTexCoord;

        uniform mat4 modelViewMatrix;
        uniform mat4 projectionMatrix;

        varying vec3 fNormal, fPosition;
        varying vec2 fTexCoord;

        void main() {
            gl_Position = projectionMatrix * modelViewMatrix * vPosition;

            fNormal = (modelViewMatrix * vNormal).xyz;
            fPosition = (modelViewMatrix * vPosition).xyz;

            fTexCoord = vTexCoord;
        }
    </script>

    <script id="texMapFS" type="x-shader/x-fragment">
        precision mediump float;
```


File
Edit
Selection
View
Go
Run
Terminal
Help
scene2.html - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.html > html > head > title
0 0
Ln 4, Col 17
Spaces: 4
UTF-8
CRLF
HTML

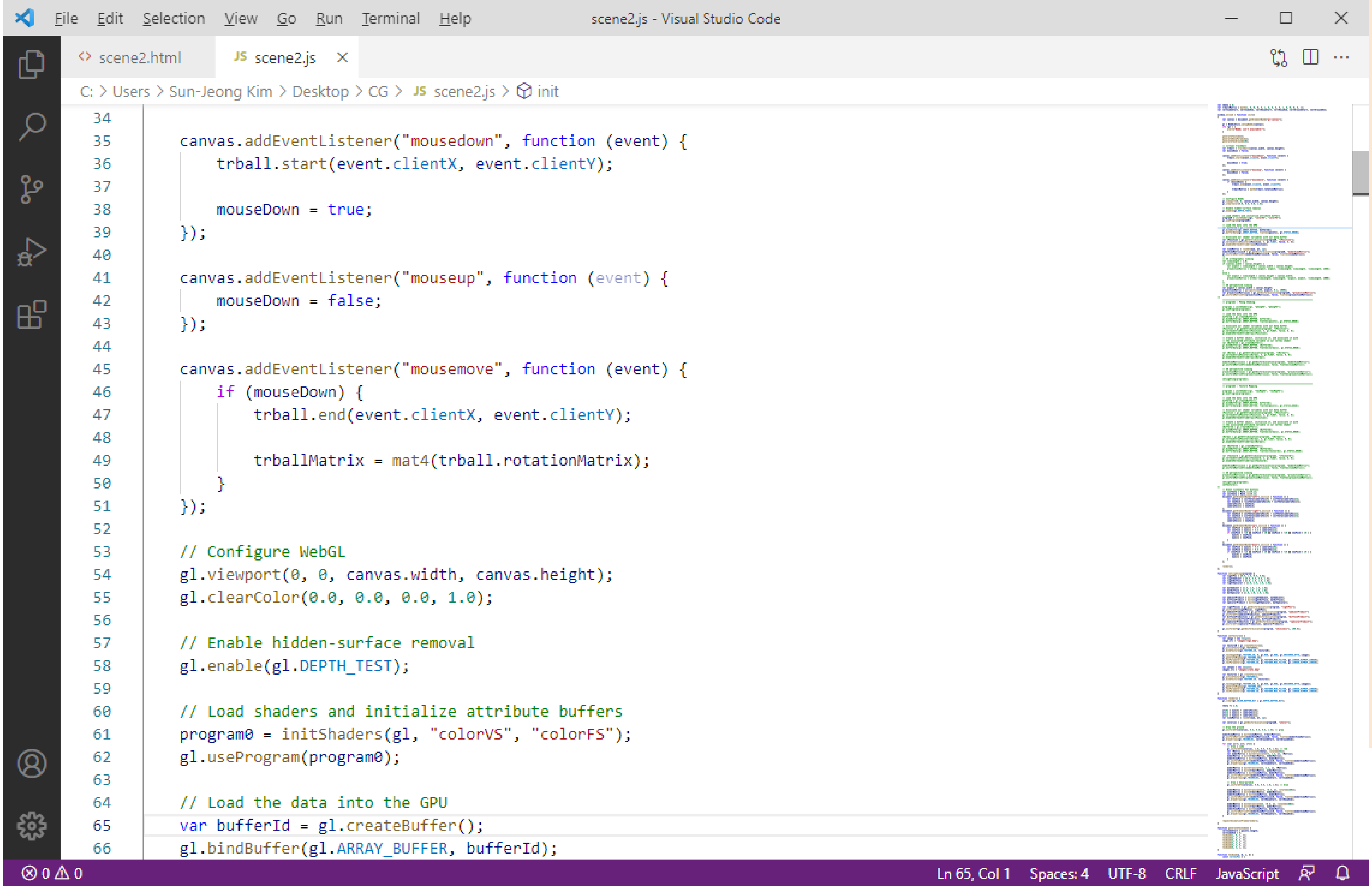
```

        varying vec3 fNormal, fPosition;
        varying vec2 fTexCoord;

        uniform sampler2D texture;
        uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
        uniform float shininess;

        void main() {
            vec3 N = normalize(fNormal);
            vec3 L = normalize(lightPos.xyz);
            float kd = max(dot(L, N), 0.0);
            vec4 diffuse = kd * diffuseProduct;

            vec3 V = normalize(fPosition);  // origin: camera position
            vec3 H = normalize(L - V);
            float ks = pow(max(dot(N, H), 0.0), shininess);
            vec4 specular = ks * specularProduct;

            if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);

            gl_FragColor = (ambientProduct + diffuse + specular) * texture2D(texture, fTexCoord);
            gl_FragColor.a = 1.0;
        }
    </script>

    <script type="text/javascript" src="Common/webgl-utils.js"></script>
    <script type="text/javascript" src="Common/initShaders.js"></script>
    <script type="text/javascript" src="Common/MV.js"></script>
    <script type="text/javascript" src="trackball.js"></script>
    <script type="text/javascript" src="scene2.js"></script>
</head>
    <body>
```


File
Edit
Selection
View
Go
Run
Terminal
Help
scene2.html - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.html > html > head > title
0 0
Ln 4, Col 17
Spaces: 4
UTF-8
CRLF
HTML

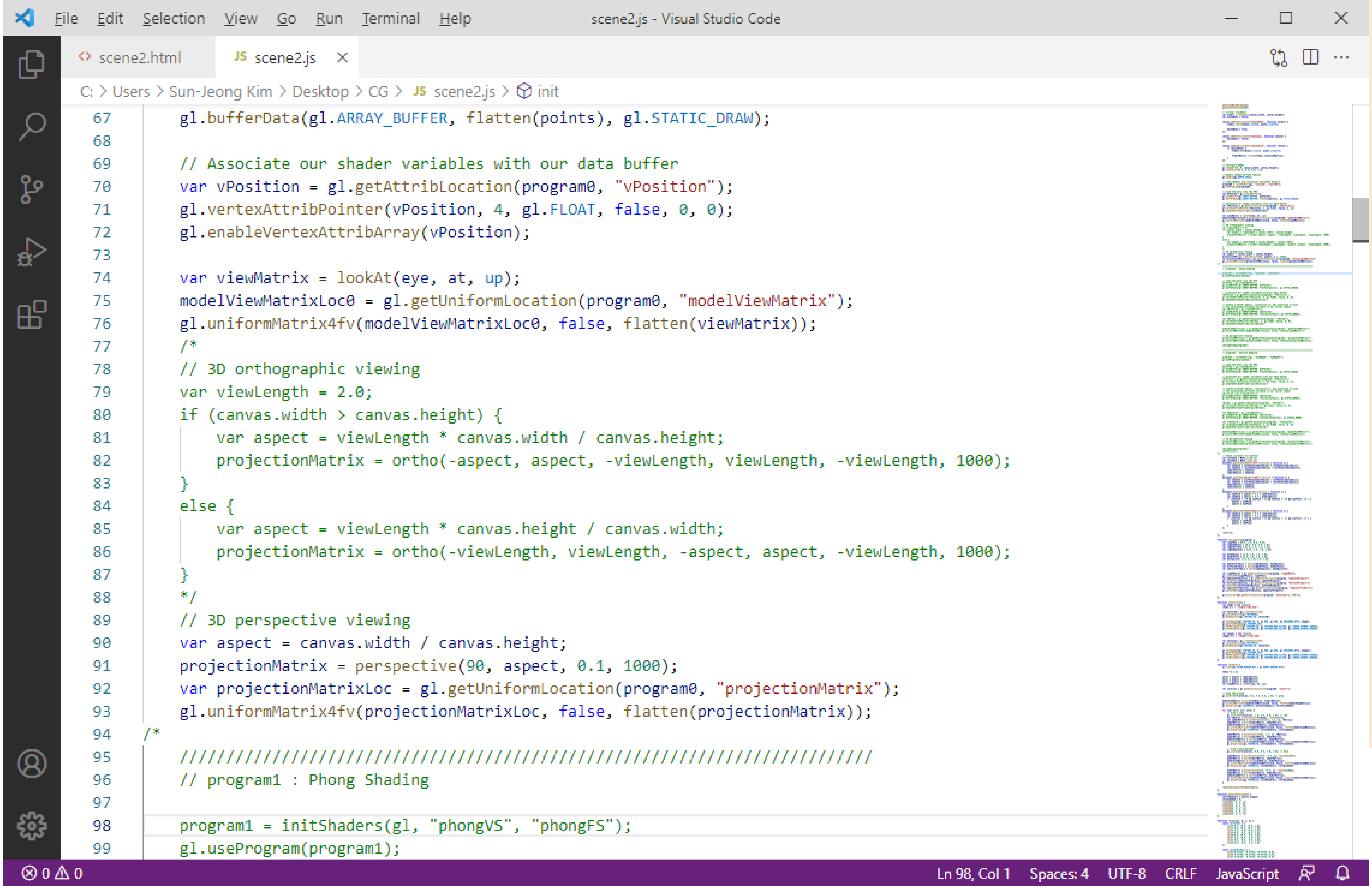
```html
        <canvas id="gl-canvas" width="800" height="600">
            Oops... your browser doesn't support the HTML5 canvas element!
        </canvas><br>
        <div style="width:800px; text-align:center;">
            <button id="up">▲</button><br>
            <button id="left">◀</button>
            <button id="down">▼</button>
            <button id="right">▶</button><br>
            <button id="space"> &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; </button>
        </div>
        <div>
            <img src="images/logo.bmp" hidden>
            <img src="images/monalisa.bmp" hidden>
            <img src="images/crate.bmp" hidden>
        </div>
    </body>
</html>
```


File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > ...
⊗ 0 ⚠ 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript

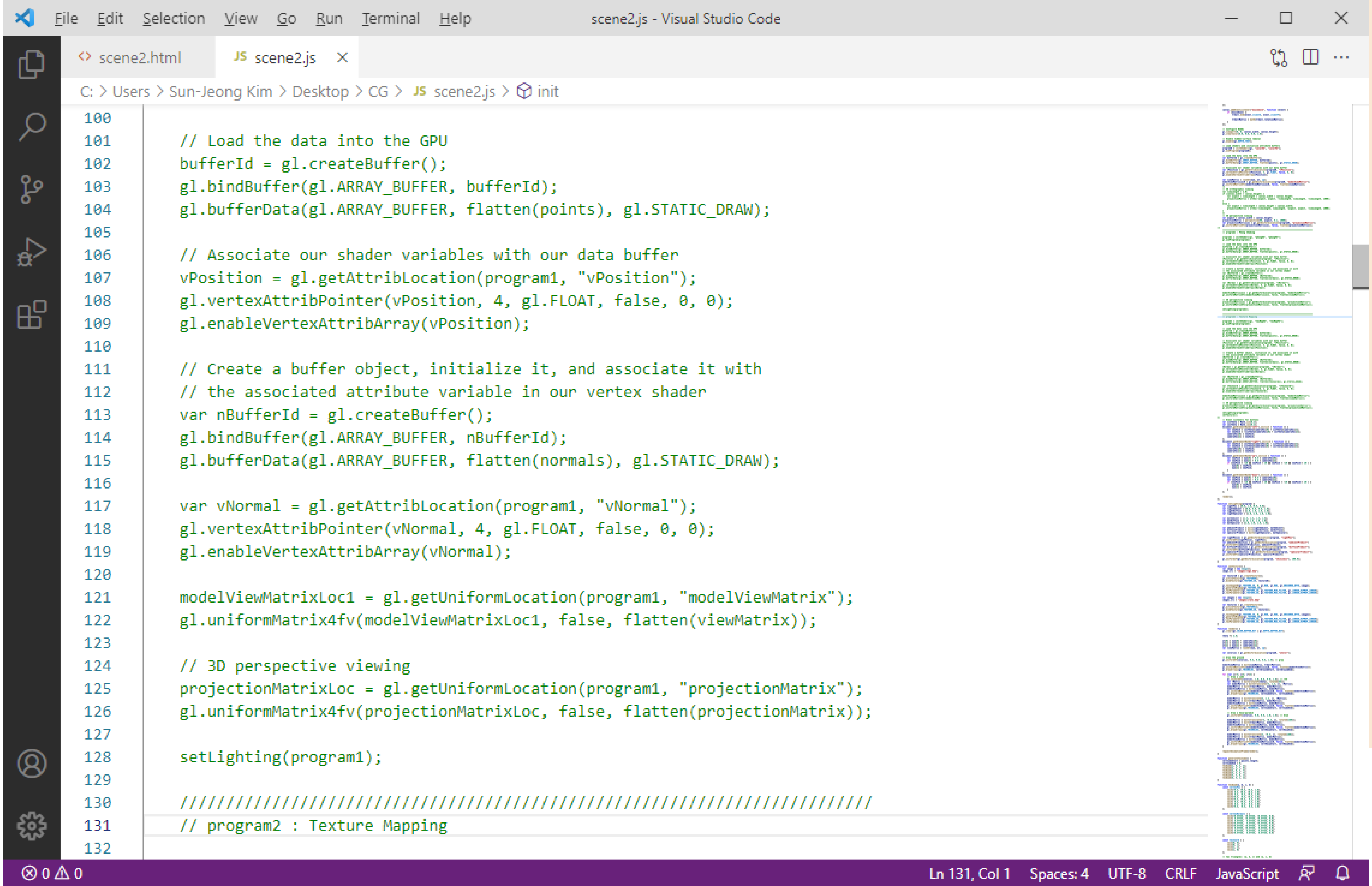
```javascript
var gl;
var points = [];
var normals = [];
var texCoords = [];

var program0, program1, program2;     // [program1] Phong shading, [program2] Texture Mapping
var modelViewMatrixLoc0, modelViewMatrixLoc1, modelViewMatrixLoc2;

var eye = vec3(0, 3, 3);
var at = vec3(0, 0, 0);
const up = vec3(0, 1, 0);
var cameraVec = vec3(0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)

var theta = 0;
var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
var vertCubeStart, vertCubeEnd, vertHexaStart, vertHexaEnd, vertGroundStart, vertGroundEnd;

window.onload = function init()
{
    var canvas = document.getElementById("gl-canvas");

    gl = WebGLUtils.setupWebGL(canvas);
    if( !gl ) {
        alert("WebGL isn't available!");
    }

    generateTexCube();
    generateHexaPyramid();
    generateTexGround(10);

    // virtual trackball
    var trball = trackball(canvas.width, canvas.height);
    var mouseDown = false;
```


File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > init
0 0
Ln 65, Col 1 Spaces: 4 UTF-8 CRLF JavaScript

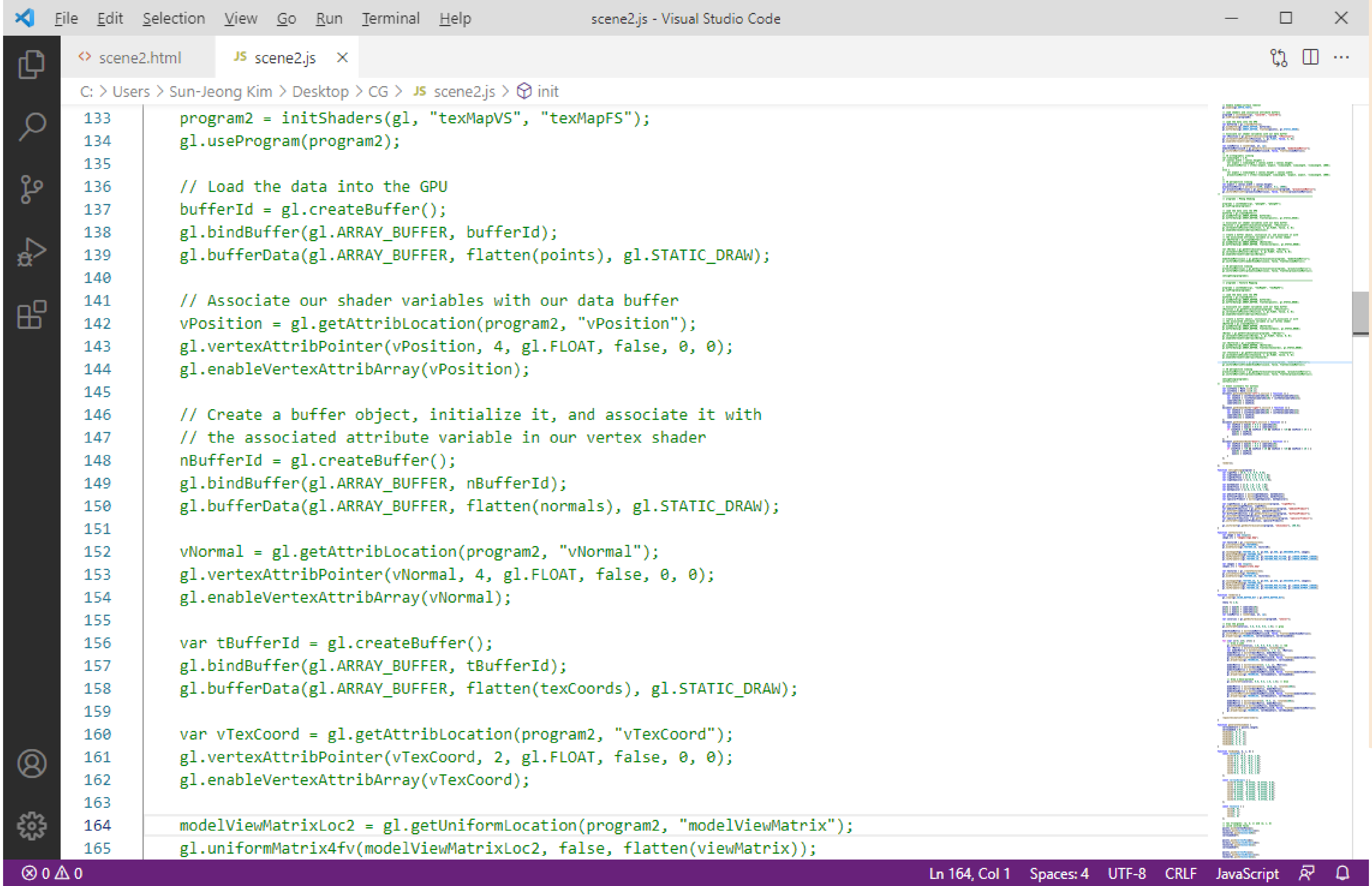
```javascript

    canvas.addEventListener("mousedown", function (event) {
        trball.start(event.clientX, event.clientY);

        mouseDown = true;
    });

    canvas.addEventListener("mouseup", function (event) {
        mouseDown = false;
    });

    canvas.addEventListener("mousemove", function (event) {
        if (mouseDown) {
            trball.end(event.clientX, event.clientY);

            trballMatrix = mat4(trball.rotationMatrix);
        }
    });

    // Configure WebGL
    gl.viewport(0, 0, canvas.width, canvas.height);
    gl.clearColor(0.0, 0.0, 0.0, 1.0);

    // Enable hidden-surface removal
    gl.enable(gl.DEPTH_TEST);

    // Load shaders and initialize attribute buffers
    program0 = initShaders(gl, "colorVS", "colorFS");
    gl.useProgram(program0);

    // Load the data into the GPU
    var bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
```



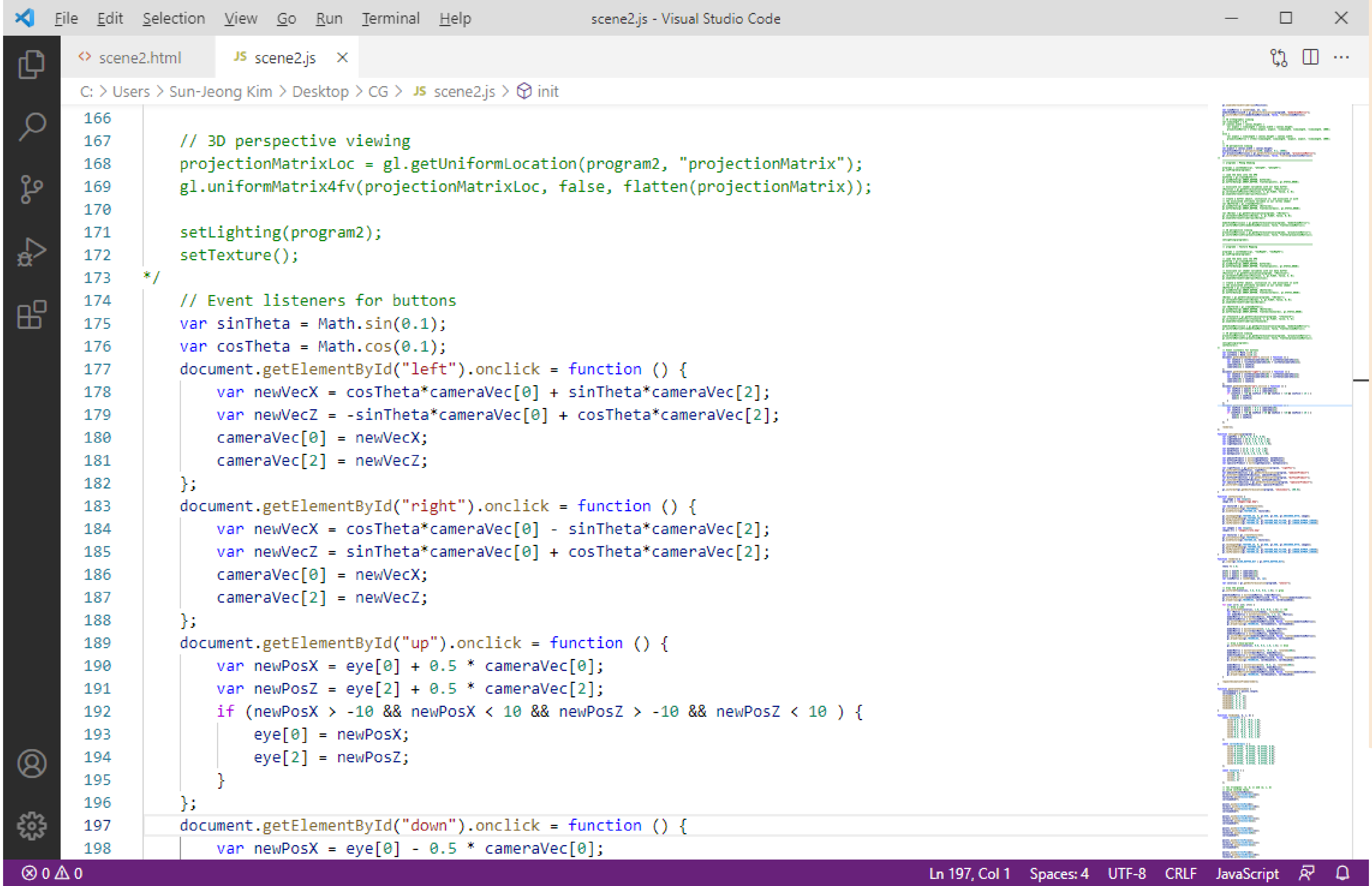
```
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    var vPosition = gl.getAttribLocation(program0, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

    var viewMatrix = lookAt(eye, at, up);
    modelViewMatrixLoc0 = gl.getUniformLocation(program0, "modelViewMatrix");
    gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(viewMatrix));
    /*
    // 3D orthographic viewing
    var viewLength = 2.0;
    if (canvas.width > canvas.height) {
        var aspect = viewLength * canvas.width / canvas.height;
        projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
    }
    else {
        var aspect = viewLength * canvas.height / canvas.width;
        projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
    }
    */
    // 3D perspective viewing
    var aspect = canvas.width / canvas.height;
    projectionMatrix = perspective(90, aspect, 0.1, 1000);
    var projectionMatrixLoc = gl.getUniformLocation(program0, "projectionMatrix");
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
/*
    ///////////////////////////////////////////////////////////////////////////
    // program1 : Phong Shading

    program1 = initShaders(gl, "phongVS", "phongFS");
    gl.useProgram(program1);
```



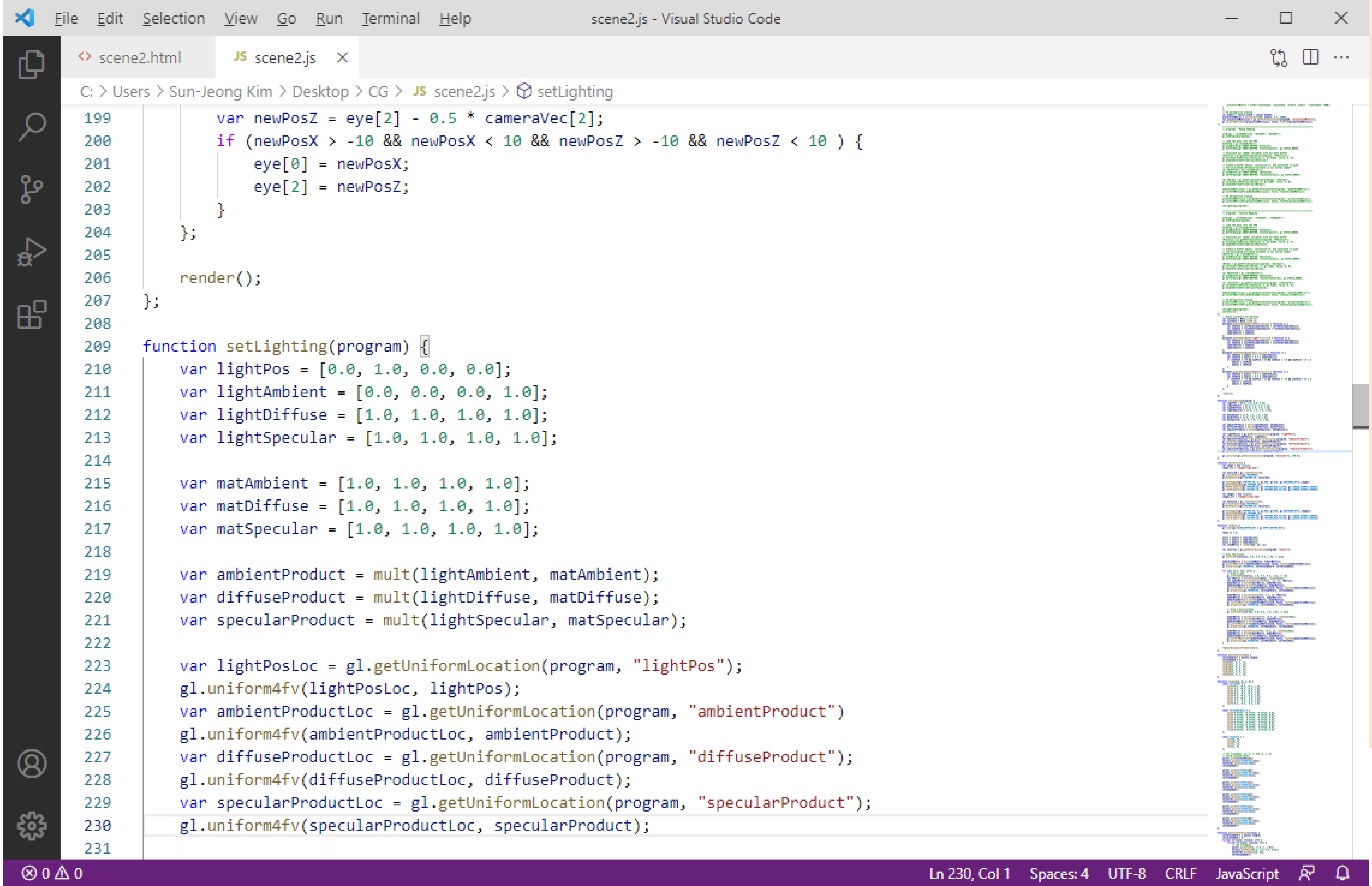
```

    // Load the data into the GPU
    bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    vPosition = gl.getAttribLocation(program1, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

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

    var vNormal = gl.getAttribLocation(program1, "vNormal");
    gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vNormal);

    modelViewMatrixLoc1 = gl.getUniformLocation(program1, "modelViewMatrix");
    gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(viewMatrix));

    // 3D perspective viewing
    projectionMatrixLoc = gl.getUniformLocation(program1, "projectionMatrix");
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));

    setLighting(program1);

    ///////////////////////////////////////////////////////////////////////////
    // program2 : Texture Mapping

```


File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > init
⊗ 0 ⚠ 0
Ln 164, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript

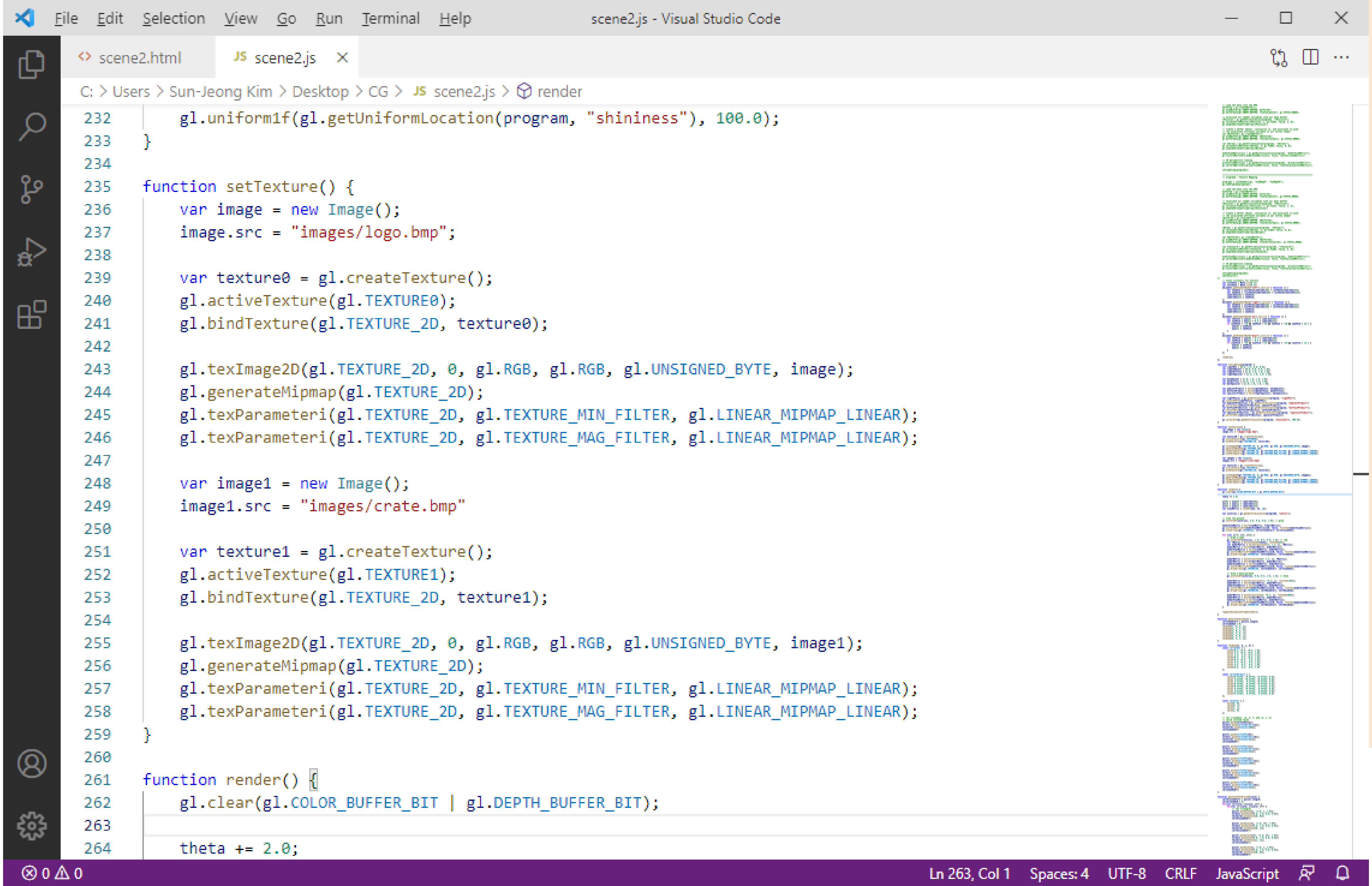
```javascript
    program2 = initShaders(gl, "texMapVS", "texMapFS");
    gl.useProgram(program2);

    // Load the data into the GPU
    bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    vPosition = gl.getAttribLocation(program2, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

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

    vNormal = gl.getAttribLocation(program2, "vNormal");
    gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vNormal);

    var tBufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, tBufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoords), gl.STATIC_DRAW);

    var vTexCoord = gl.getAttribLocation(program2, "vTexCoord");
    gl.vertexAttribPointer(vTexCoord, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vTexCoord);

    modelViewMatrixLoc2 = gl.getUniformLocation(program2, "modelViewMatrix");
    gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(viewMatrix));
```



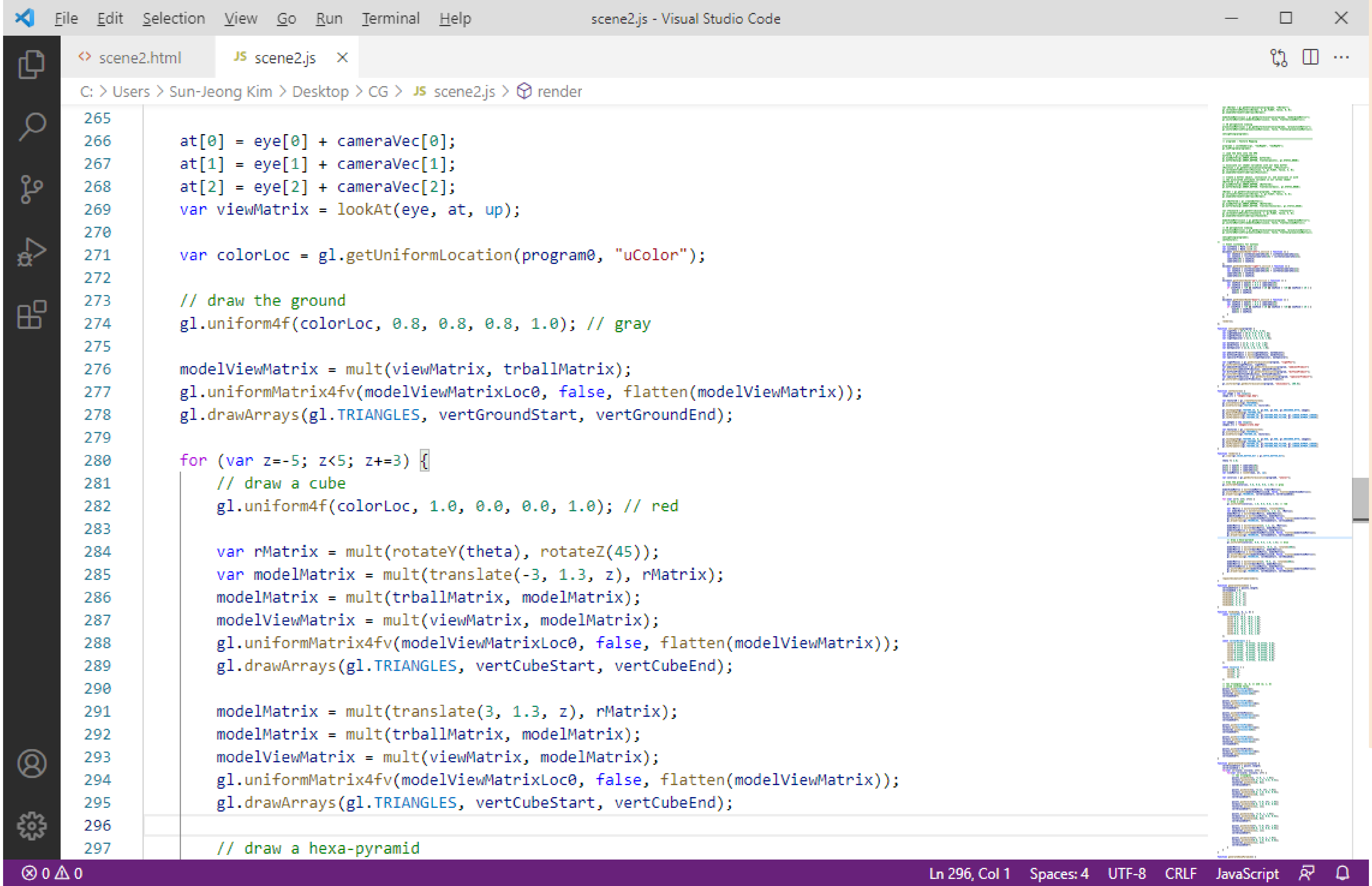
```

    // 3D perspective viewing
    projectionMatrixLoc = gl.getUniformLocation(program2, "projectionMatrix");
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));

    setLighting(program2);
    setTexture();
*/
    // Event listeners for buttons
    var sinTheta = Math.sin(0.1);
    var cosTheta = Math.cos(0.1);
    document.getElementById("left").onclick = function () {
        var newVecX = cosTheta*cameraVec[0] + sinTheta*cameraVec[2];
        var newVecZ = -sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
        cameraVec[0] = newVecX;
        cameraVec[2] = newVecZ;
    };
    document.getElementById("right").onclick = function () {
        var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
        var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
        cameraVec[0] = newVecX;
        cameraVec[2] = newVecZ;
    };
    document.getElementById("up").onclick = function () {
        var newPosX = eye[0] + 0.5 * cameraVec[0];
        var newPosZ = eye[2] + 0.5 * cameraVec[2];
        if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 ) {
            eye[0] = newPosX;
            eye[2] = newPosZ;
        }
    };
    document.getElementById("down").onclick = function () {
        var newPosX = eye[0] - 0.5 * cameraVec[0];
```



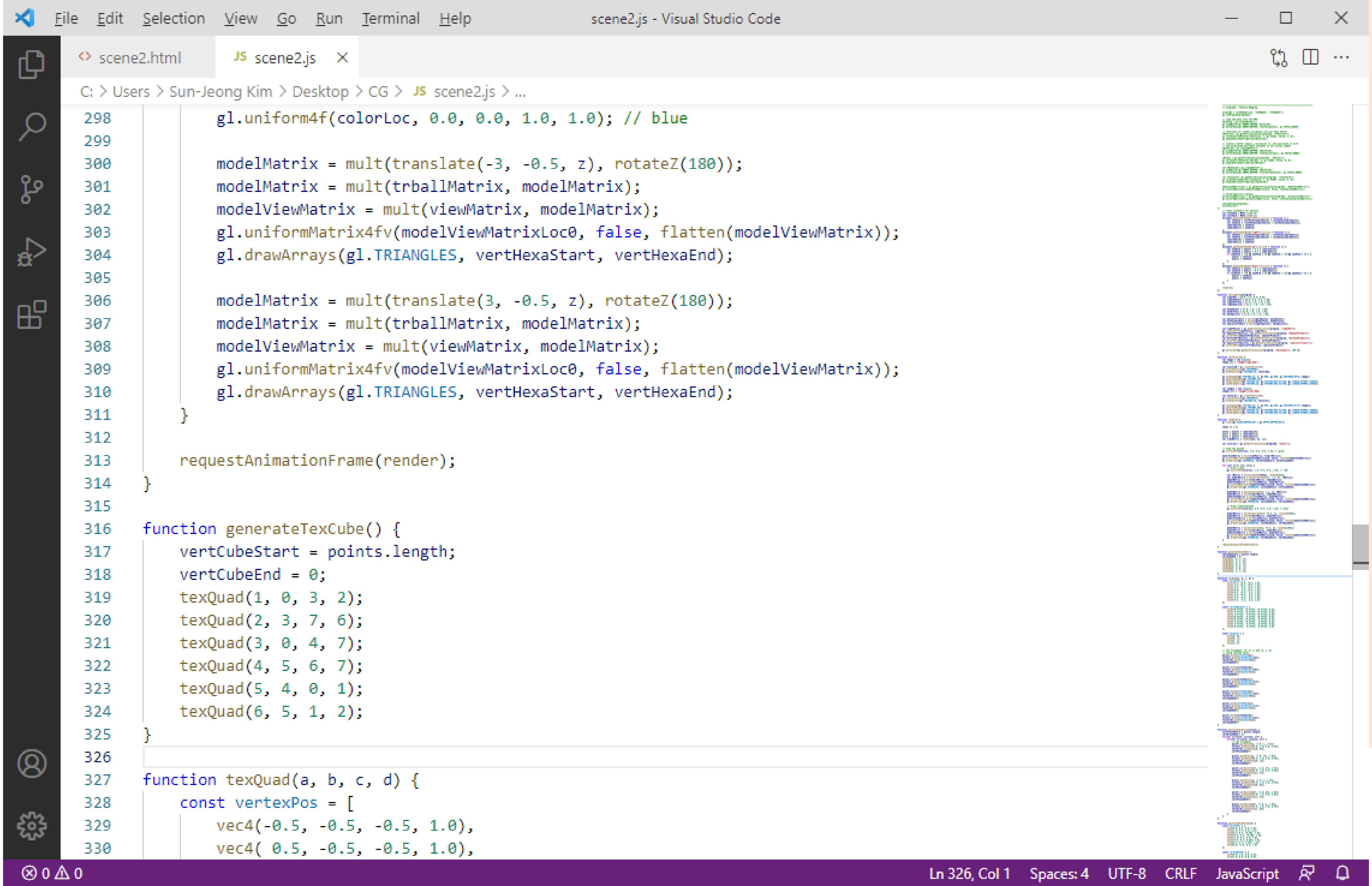
```javascript
        var newPosZ = eye[2] - 0.5 * cameraVec[2];
        if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 ) {
            eye[0] = newPosX;
            eye[2] = newPosZ;
        }
    };

    render();
};

function setLighting(program) {
    var lightPos = [0.0, 1.0, 0.0, 0.0];
    var lightAmbient = [0.0, 0.0, 0.0, 1.0];
    var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
    var lightSpecular = [1.0, 1.0, 1.0, 1.0];

    var matAmbient = [1.0, 1.0, 1.0, 1.0];
    var matDiffuse = [1.0, 1.0, 1.0, 1.0];
    var matSpecular = [1.0, 1.0, 1.0, 1.0];

    var ambientProduct = mult(lightAmbient, matAmbient);
    var diffuseProduct = mult(lightDiffuse, matDiffuse);
    var specularProduct = mult(lightSpecular, matSpecular);

    var lightPosLoc = gl.getUniformLocation(program, "lightPos");
    gl.uniform4fv(lightPosLoc, lightPos);
    var ambientProductLoc = gl.getUniformLocation(program, "ambientProduct")
    gl.uniform4fv(ambientProductLoc, ambientProduct);
    var diffuseProductLoc = gl.getUniformLocation(program, "diffuseProduct");
    gl.uniform4fv(diffuseProductLoc, diffuseProduct);
    var specularProductLoc = gl.getUniformLocation(program, "specularProduct");
    gl.uniform4fv(specularProductLoc, specularProduct);

```



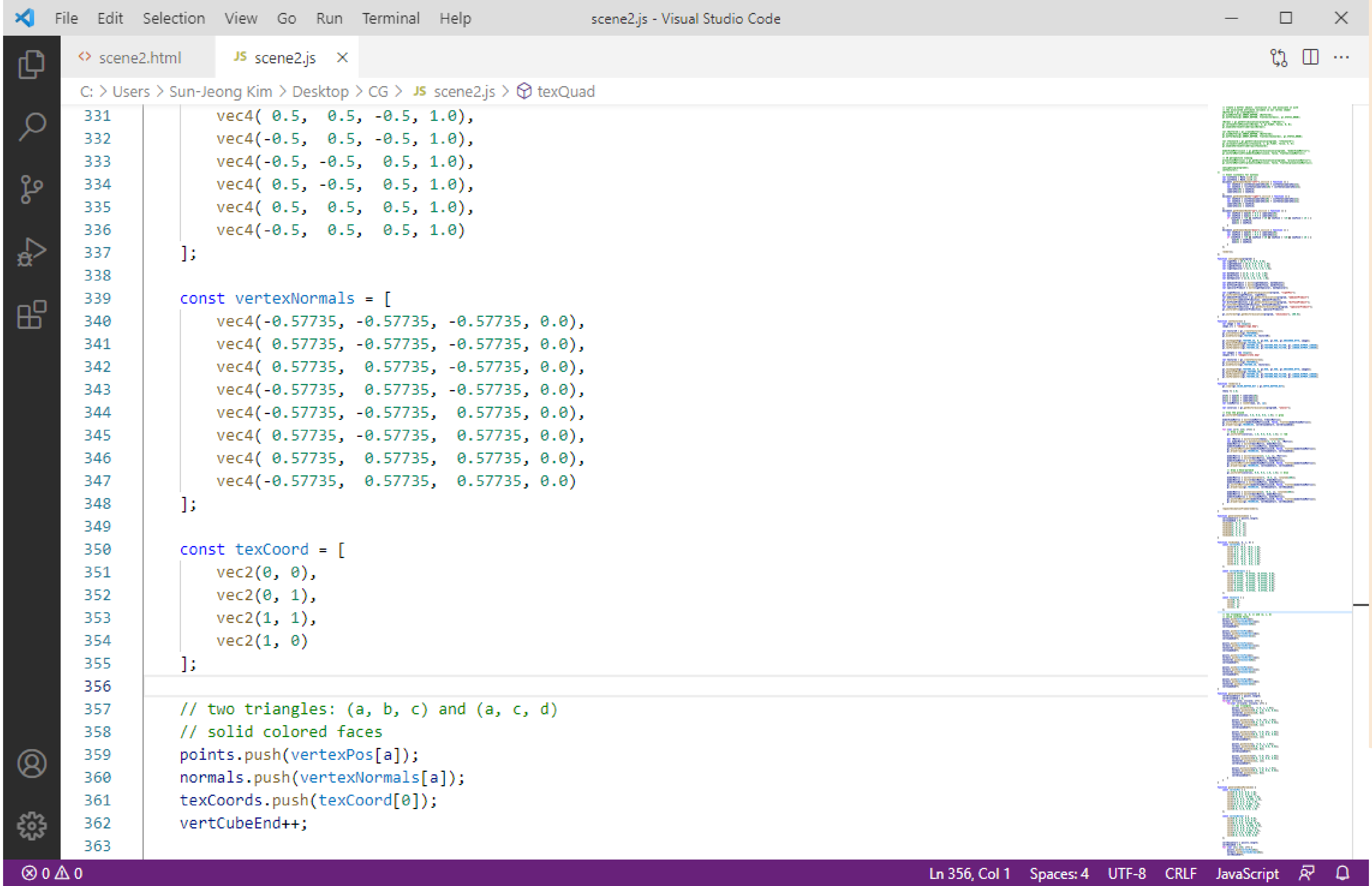
```
    gl.uniform1f(gl.getUniformLocation(program, "shininess"), 100.0);
}

function setTexture() {
    var image = new Image();
    image.src = "images/logo.bmp";

    var texture0 = gl.createTexture();
    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(gl.TEXTURE_2D, texture0);

    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR_MIPMAP_LINEAR);

    var image1 = new Image();
    image1.src = "images/crate.bmp"

    var texture1 = gl.createTexture();
    gl.activeTexture(gl.TEXTURE1);
    gl.bindTexture(gl.TEXTURE_2D, texture1);

    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image1);
    gl.generateMipmap(gl.TEXTURE_2D);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR_MIPMAP_LINEAR);
}

function render() {
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    theta += 2.0;
```



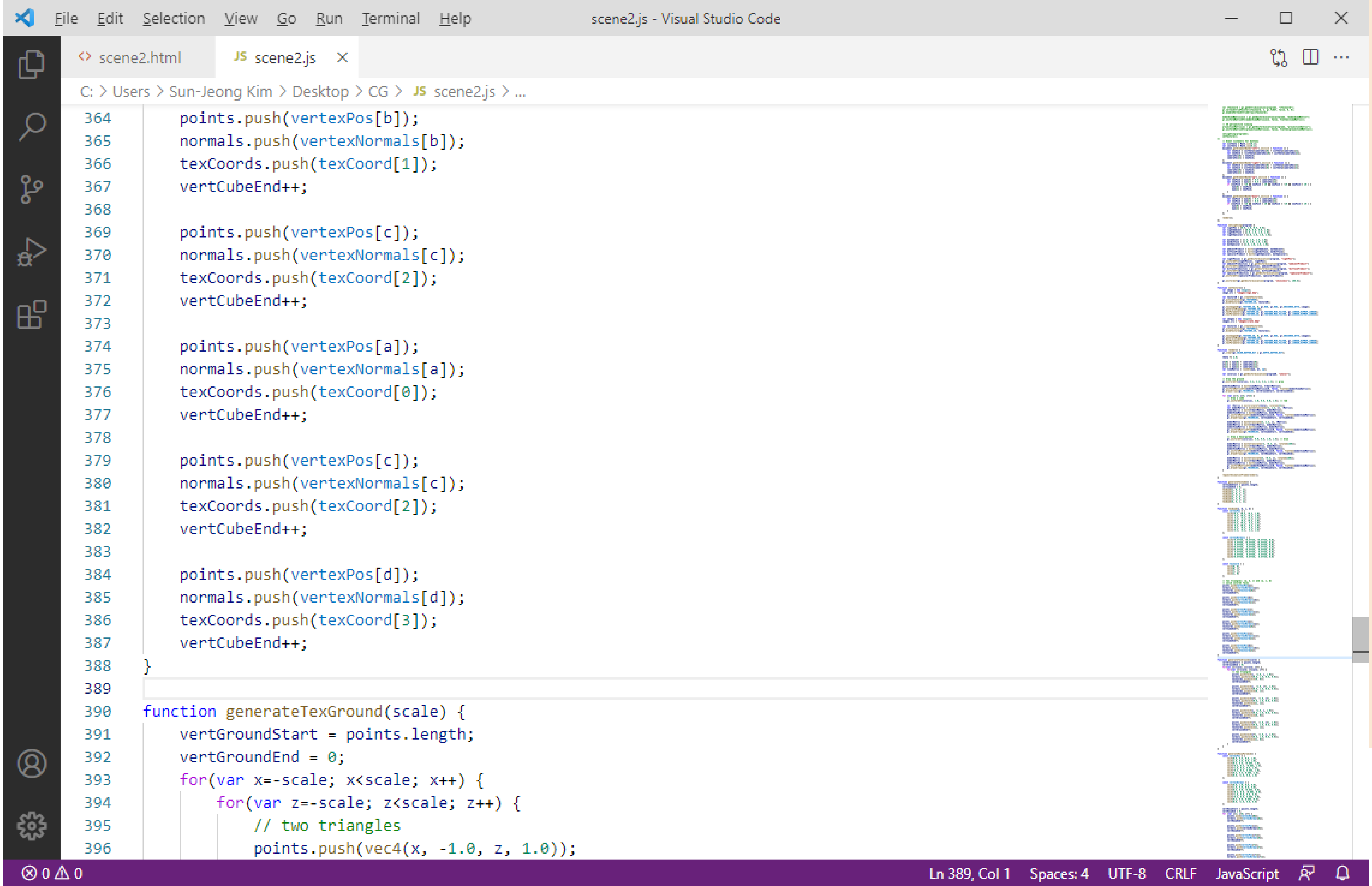
```

    at[0] = eye[0] + cameraVec[0];
    at[1] = eye[1] + cameraVec[1];
    at[2] = eye[2] + cameraVec[2];
    var viewMatrix = lookAt(eye, at, up);

    var colorLoc = gl.getUniformLocation(program0, "uColor");

    // draw the ground
    gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray

    modelViewMatrix = mult(viewMatrix, trballMatrix);
    gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);

    for (var z=-5; z<5; z+=3) {
        // draw a cube
        gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red

        var rMatrix = mult(rotateY(theta), rotateZ(45));
        var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

        modelMatrix = mult(translate(3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

        // draw a hexa-pyramid
```


File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
JS scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > ...
⊗ 0 ⚠ 0
Ln 326, Col 1
Spaces: 4
UTF-8
CRLF
JavaScript

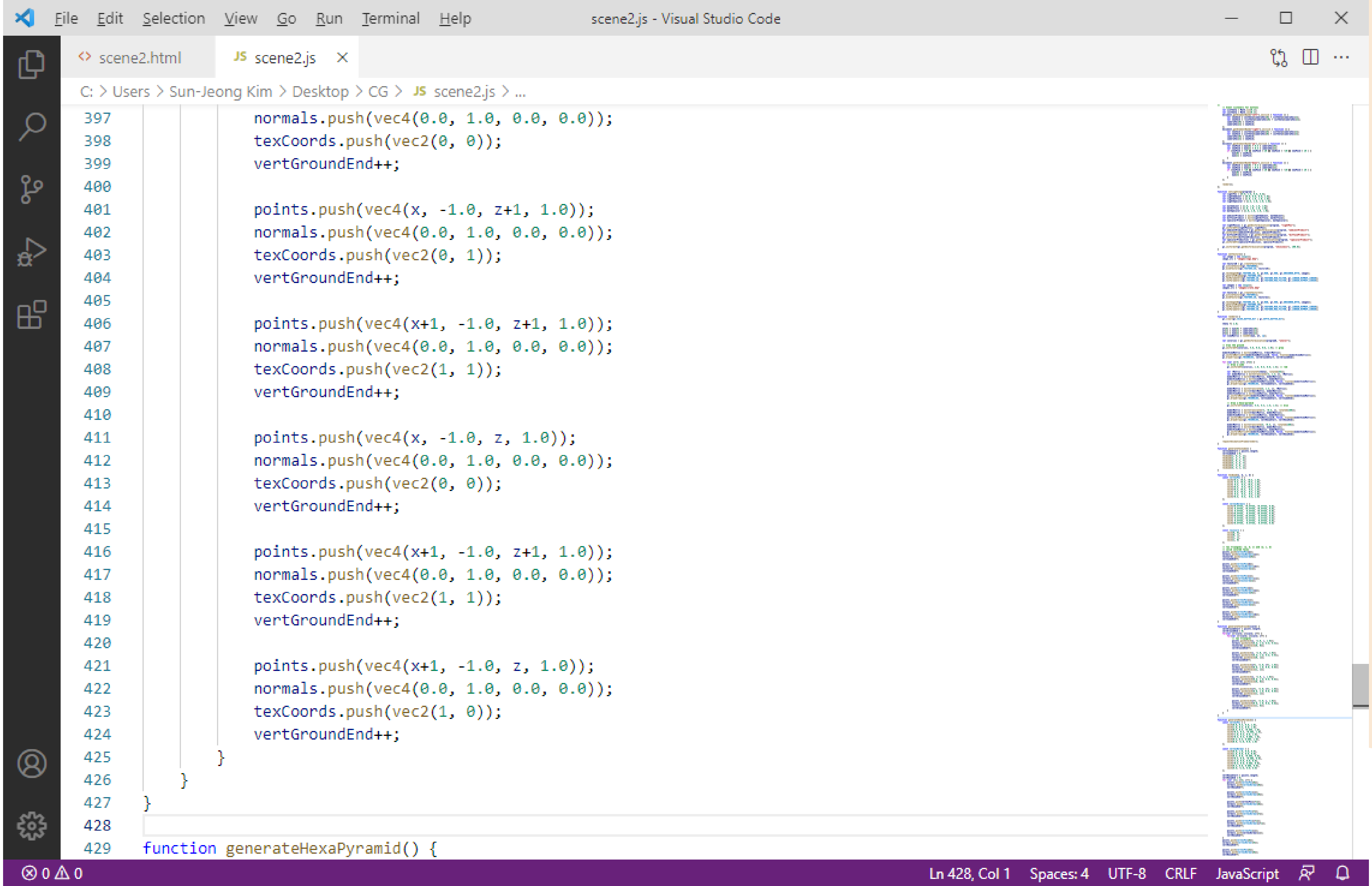
```javascript
        gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 1.0); // blue

        modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
    }

    requestAnimationFrame(render);
}

function generateTexCube() {
    vertCubeStart = points.length;
    vertCubeEnd = 0;
    texQuad(1, 0, 3, 2);
    texQuad(2, 3, 7, 6);
    texQuad(3, 0, 4, 7);
    texQuad(4, 5, 6, 7);
    texQuad(5, 4, 0, 1);
    texQuad(6, 5, 1, 2);
}

function texQuad(a, b, c, d) {
    const vertexPos = [
        vec4(-0.5, -0.5, -0.5, 1.0),
        vec4( 0.5, -0.5, -0.5, 1.0),
```



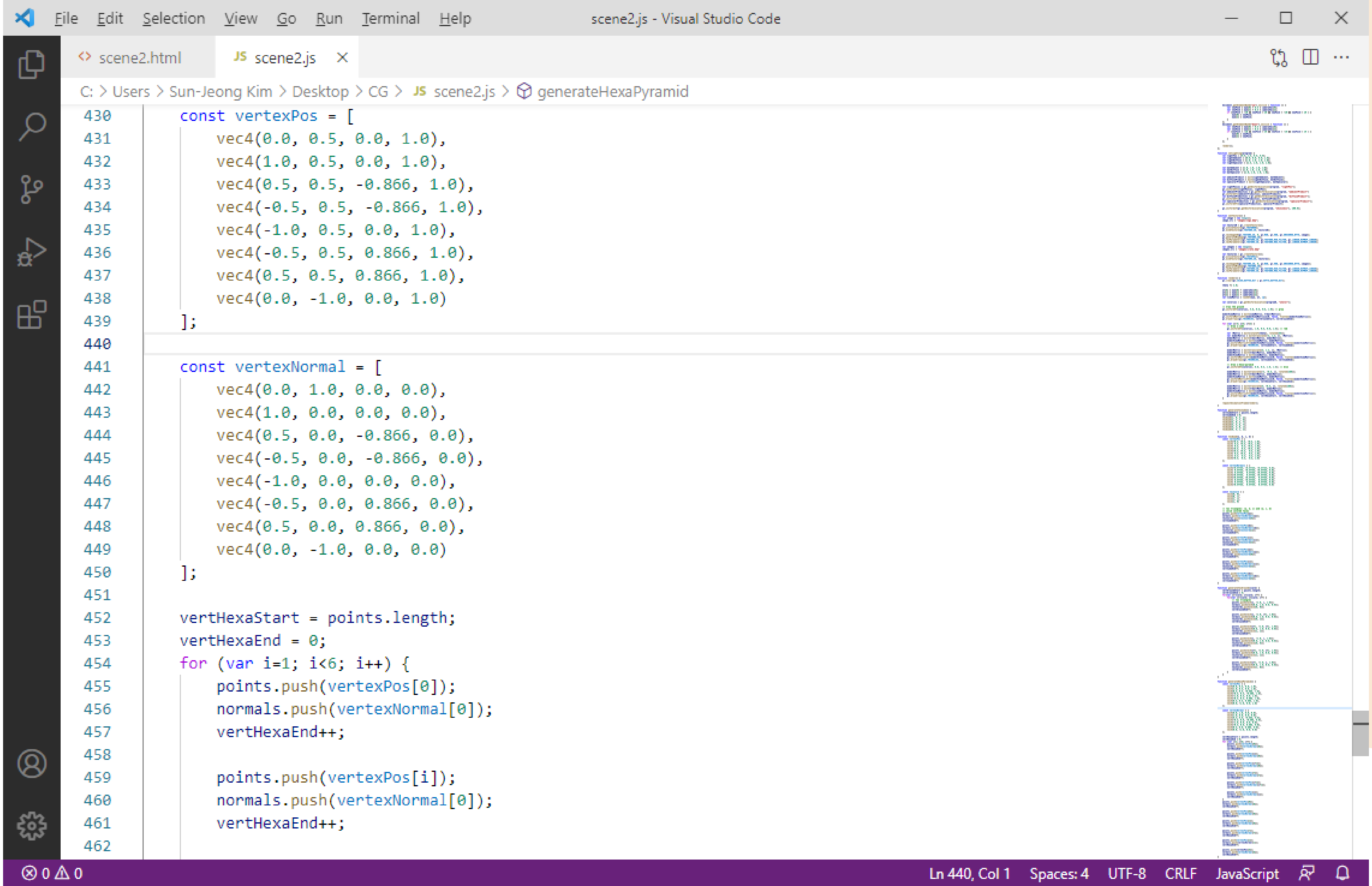
```
        vec4( 0.5,  0.5, -0.5, 1.0),
        vec4(-0.5,  0.5, -0.5, 1.0),
        vec4(-0.5, -0.5,  0.5, 1.0),
        vec4( 0.5, -0.5,  0.5, 1.0),
        vec4( 0.5,  0.5,  0.5, 1.0),
        vec4(-0.5,  0.5,  0.5, 1.0)
    ];

    const vertexNormals = [
        vec4(-0.57735, -0.57735, -0.57735, 0.0),
        vec4( 0.57735, -0.57735, -0.57735, 0.0),
        vec4( 0.57735,  0.57735, -0.57735, 0.0),
        vec4(-0.57735,  0.57735, -0.57735, 0.0),
        vec4(-0.57735, -0.57735,  0.57735, 0.0),
        vec4( 0.57735, -0.57735,  0.57735, 0.0),
        vec4( 0.57735,  0.57735,  0.57735, 0.0),
        vec4(-0.57735,  0.57735,  0.57735, 0.0)
    ];

    const texCoord = [
        vec2(0, 0),
        vec2(0, 1),
        vec2(1, 1),
        vec2(1, 0)
    ];

    // two triangles: (a, b, c) and (a, c, d)
    // solid colored faces
    points.push(vertexPos[a]);
    normals.push(vertexNormals[a]);
    texCoords.push(texCoord[0]);
    vertCubeEnd++;

```



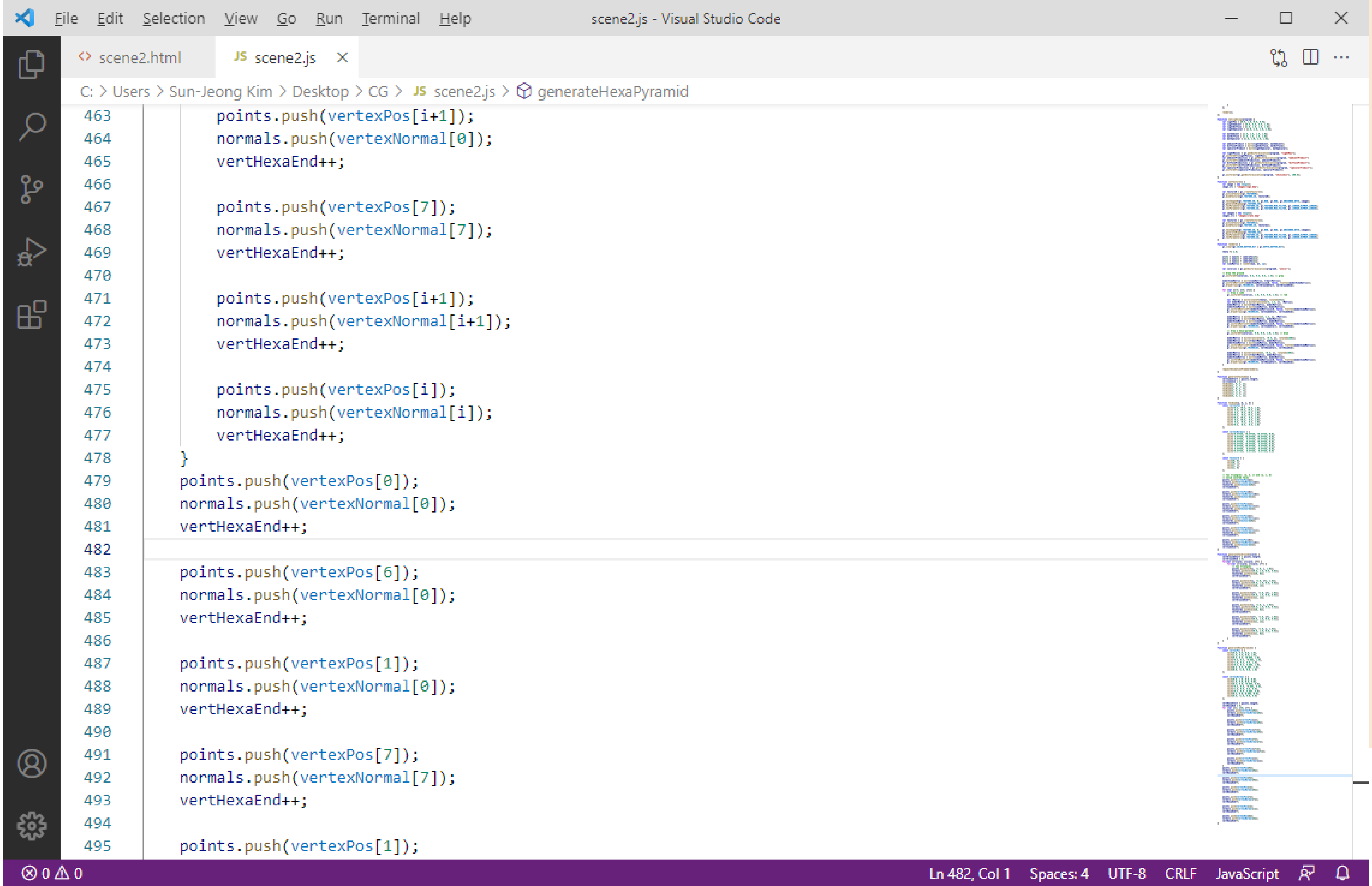
```
    points.push(vertexPos[b]);
    normals.push(vertexNormals[b]);
    texCoords.push(texCoord[1]);
    vertCubeEnd++;

    points.push(vertexPos[c]);
    normals.push(vertexNormals[c]);
    texCoords.push(texCoord[2]);
    vertCubeEnd++;

    points.push(vertexPos[a]);
    normals.push(vertexNormals[a]);
    texCoords.push(texCoord[0]);
    vertCubeEnd++;

    points.push(vertexPos[c]);
    normals.push(vertexNormals[c]);
    texCoords.push(texCoord[2]);
    vertCubeEnd++;

    points.push(vertexPos[d]);
    normals.push(vertexNormals[d]);
    texCoords.push(texCoord[3]);
    vertCubeEnd++;
}

function generateTexGround(scale) {
    vertGroundStart = points.length;
    vertGroundEnd = 0;
    for(var x=-scale; x<scale; x++) {
        for(var z=-scale; z<scale; z++) {
            // two triangles
            points.push(vec4(x, -1.0, z, 1.0));
```



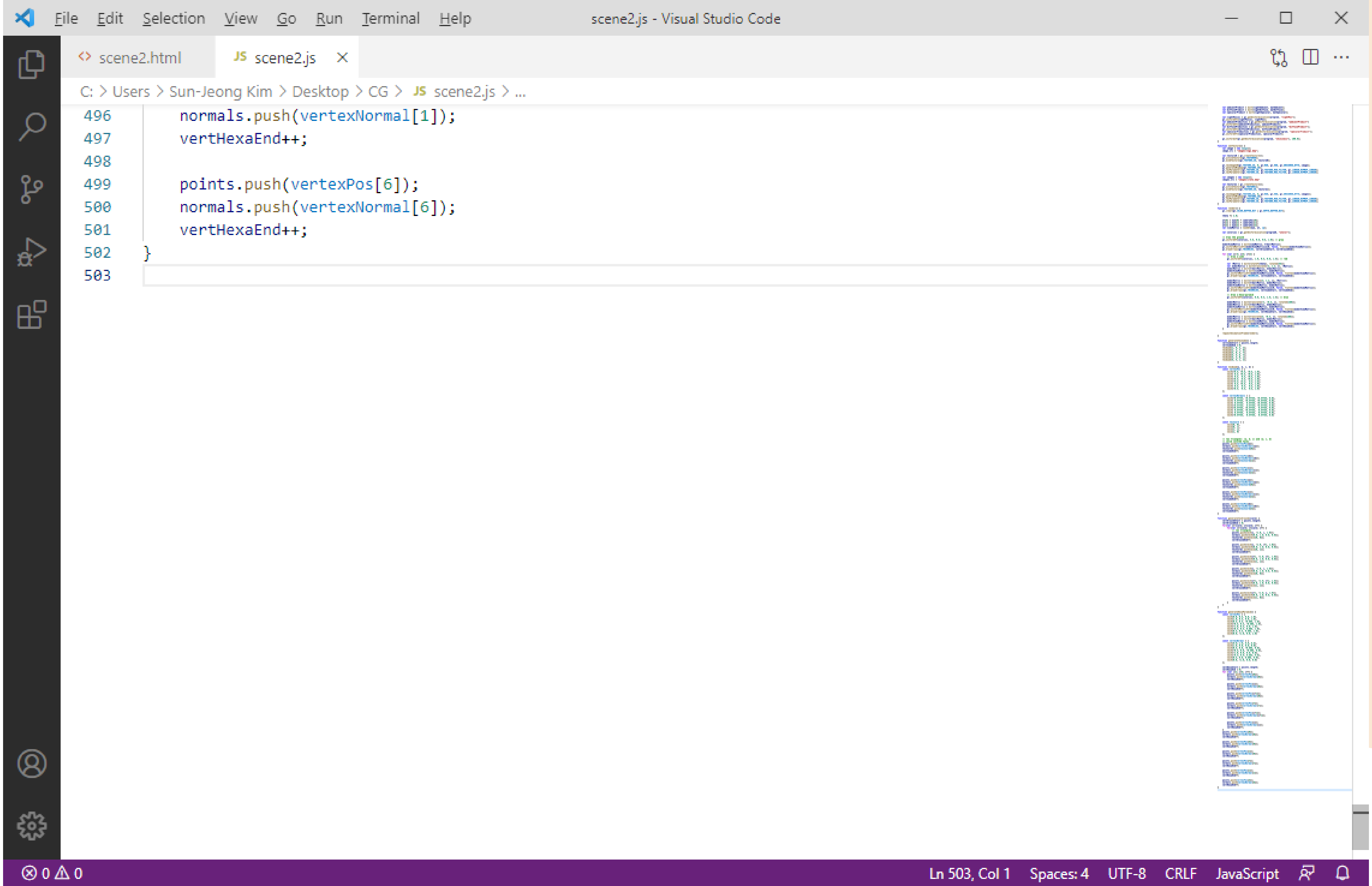
```
            normals.push(vec4(0.0, 1.0, 0.0, 0.0));
            texCoords.push(vec2(0, 0));
            vertGroundEnd++;

            points.push(vec4(x, -1.0, z+1, 1.0));
            normals.push(vec4(0.0, 1.0, 0.0, 0.0));
            texCoords.push(vec2(0, 1));
            vertGroundEnd++;

            points.push(vec4(x+1, -1.0, z+1, 1.0));
            normals.push(vec4(0.0, 1.0, 0.0, 0.0));
            texCoords.push(vec2(1, 1));
            vertGroundEnd++;

            points.push(vec4(x, -1.0, z, 1.0));
            normals.push(vec4(0.0, 1.0, 0.0, 0.0));
            texCoords.push(vec2(0, 0));
            vertGroundEnd++;

            points.push(vec4(x+1, -1.0, z+1, 1.0));
            normals.push(vec4(0.0, 1.0, 0.0, 0.0));
            texCoords.push(vec2(1, 1));
            vertGroundEnd++;

            points.push(vec4(x+1, -1.0, z, 1.0));
            normals.push(vec4(0.0, 1.0, 0.0, 0.0));
            texCoords.push(vec2(1, 0));
            vertGroundEnd++;
        }
    }
}

function generateHexaPyramid() {
```


File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
JS scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > generateHexaPyramid
⊗ 0 ⚠ 0
Ln 440, Col 1 Spaces: 4 UTF-8 CRLF JavaScript

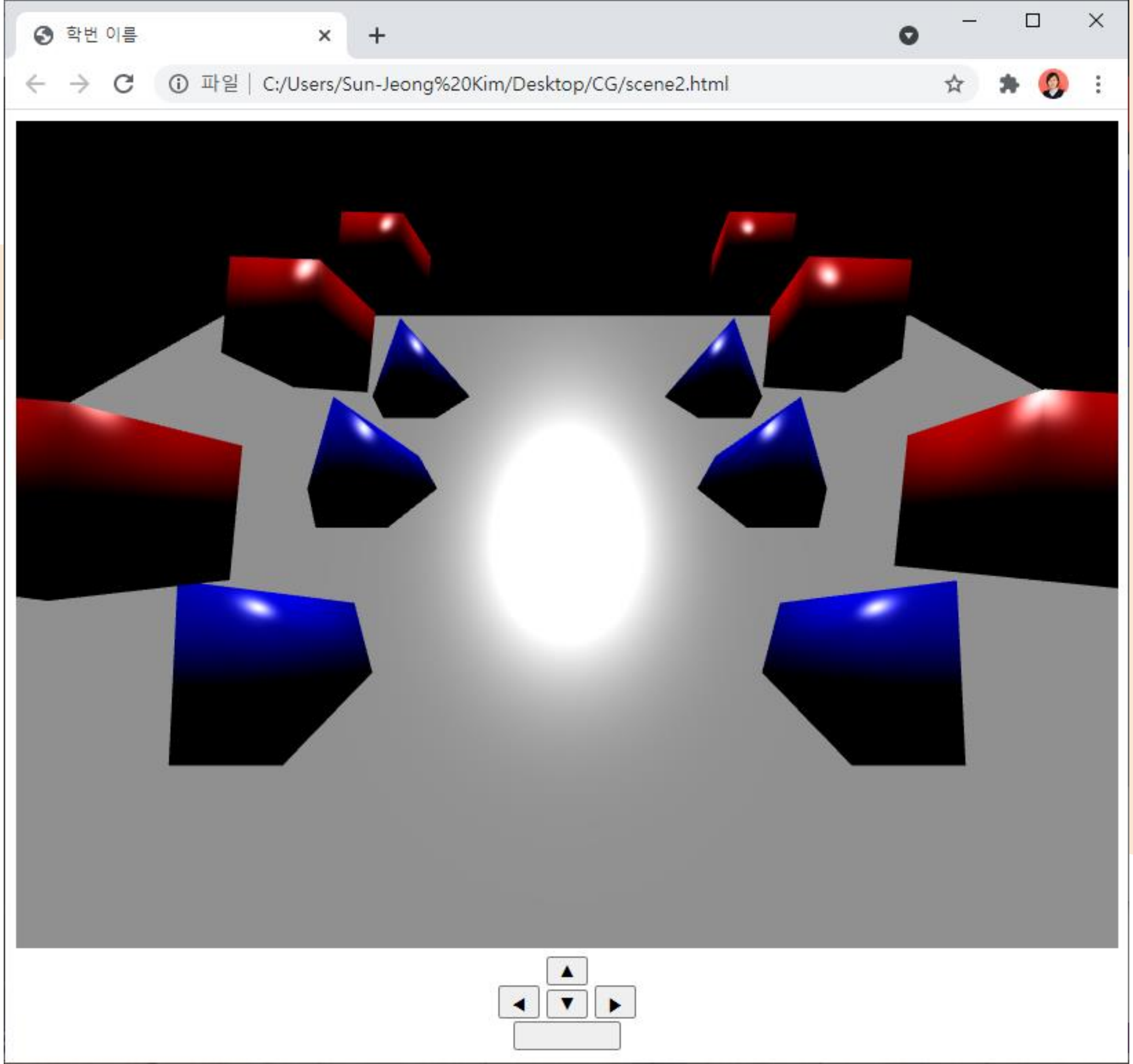
```
    const vertexPos = [
        vec4(0.0, 0.5, 0.0, 1.0),
        vec4(1.0, 0.5, 0.0, 1.0),
        vec4(0.5, 0.5, -0.866, 1.0),
        vec4(-0.5, 0.5, -0.866, 1.0),
        vec4(-1.0, 0.5, 0.0, 1.0),
        vec4(-0.5, 0.5, 0.866, 1.0),
        vec4(0.5, 0.5, 0.866, 1.0),
        vec4(0.0, -1.0,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

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

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

        points.push(vertexPos[i]);
        normals.push(vertexNormal[0]);
        vertHexaEnd++;

```

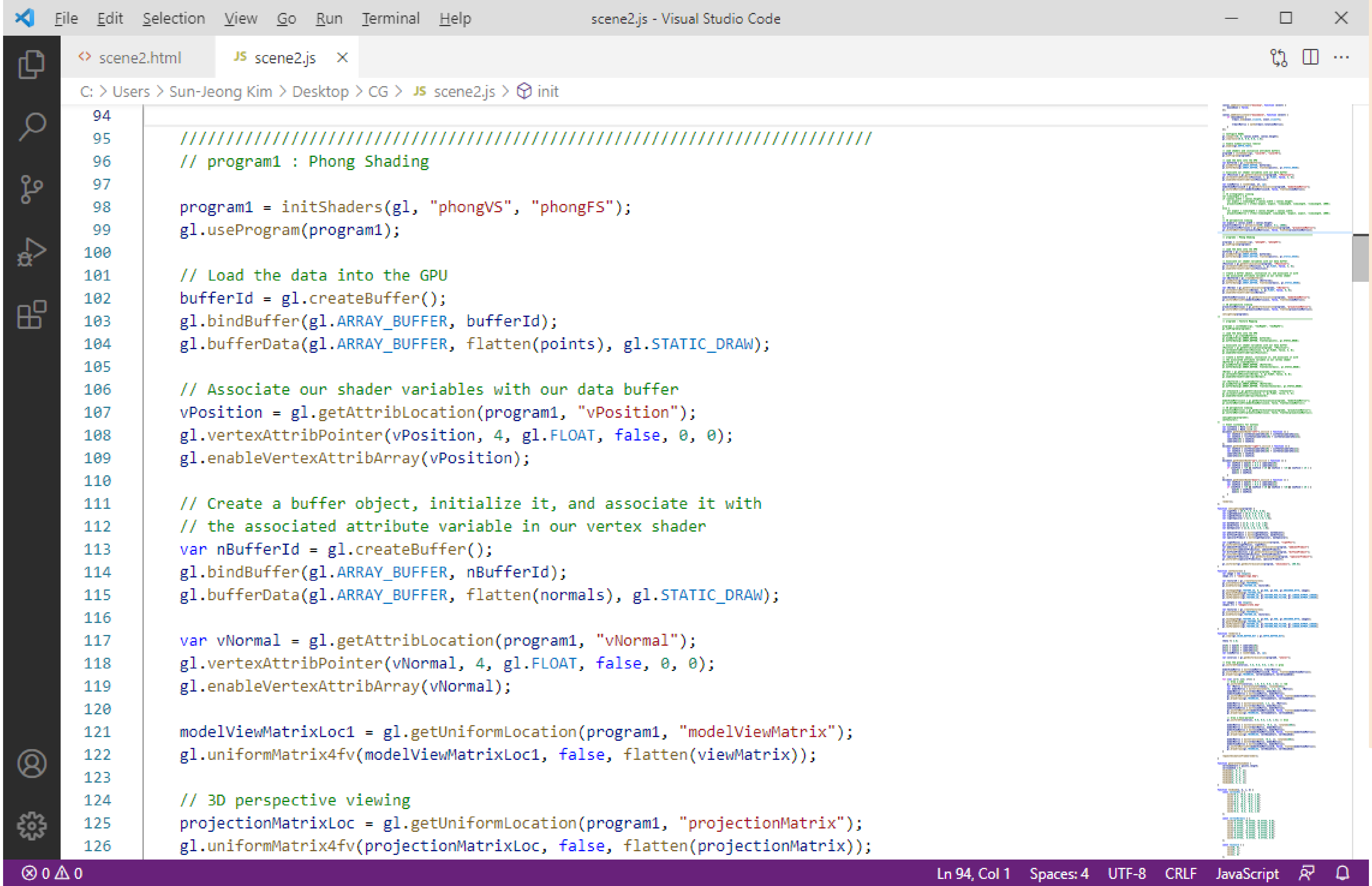


```
        points.push(vertexPos[i+1]);
        normals.push(vertexNormal[0]);
        vertHexaEnd++;

        points.push(vertexPos[7]);
        normals.push(vertexNormal[7]);
        vertHexaEnd++;

        points.push(vertexPos[i+1]);
        normals.push(vertexNormal[i+1]);
        vertHexaEnd++;

        points.push(vertexPos[i]);
        normals.push(vertexNormal[i]);
        vertHexaEnd++;
    }
    points.push(vertexPos[0]);
    normals.push(vertexNormal[0]);
    vertHexaEnd++;

    points.push(vertexPos[6]);
    normals.push(vertexNormal[0]);
    vertHexaEnd++;

    points.push(vertexPos[1]);
    normals.push(vertexNormal[0]);
    vertHexaEnd++;

    points.push(vertexPos[7]);
    normals.push(vertexNormal[7]);
    vertHexaEnd++;

    points.push(vertexPos[1]);
```

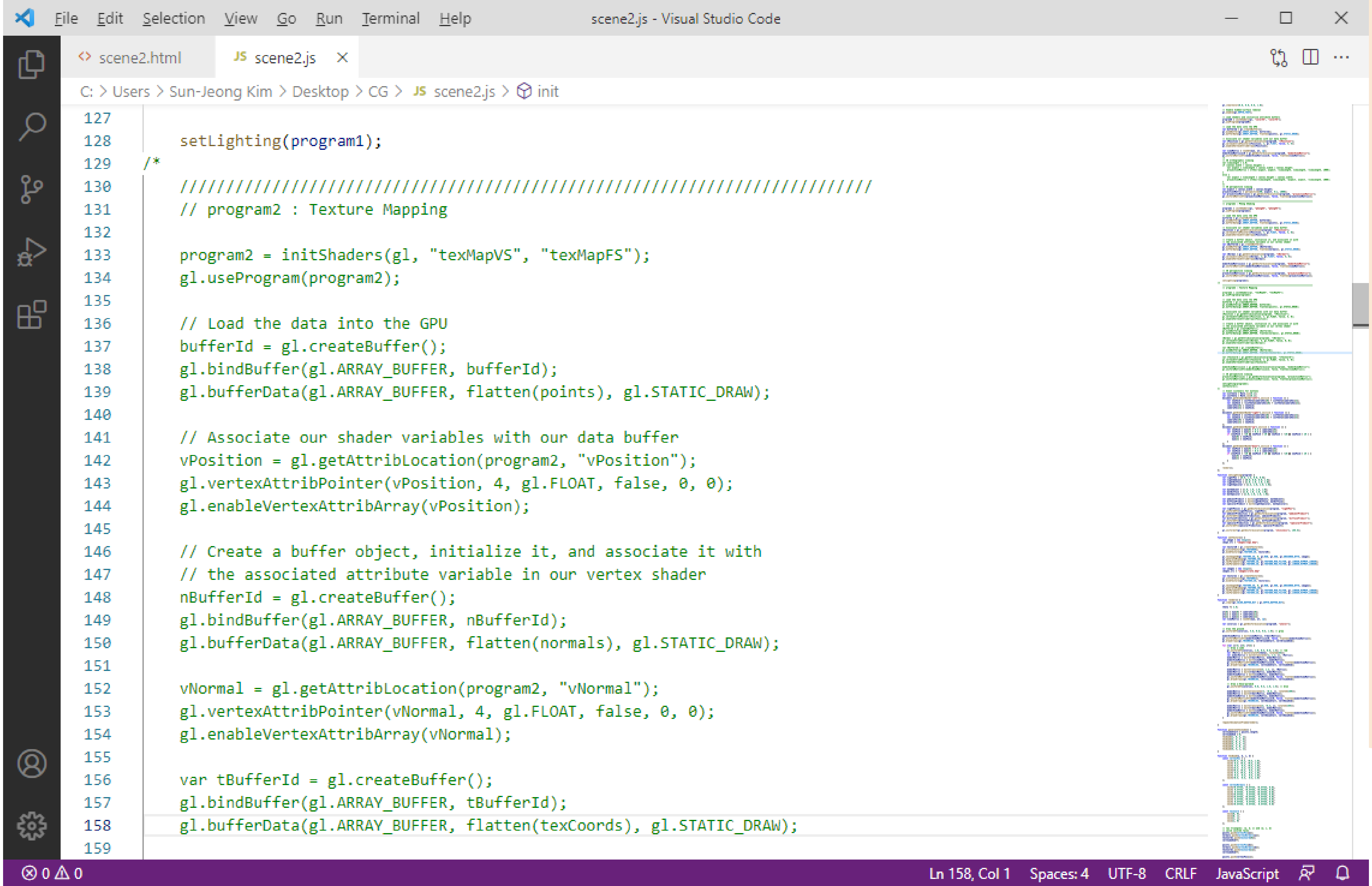


```
        normals.push(vertexNormal[1]);
        vertHexaEnd++;

        points.push(vertexPos[6]);
        normals.push(vertexNormal[6]);
        vertHexaEnd++;
}

```

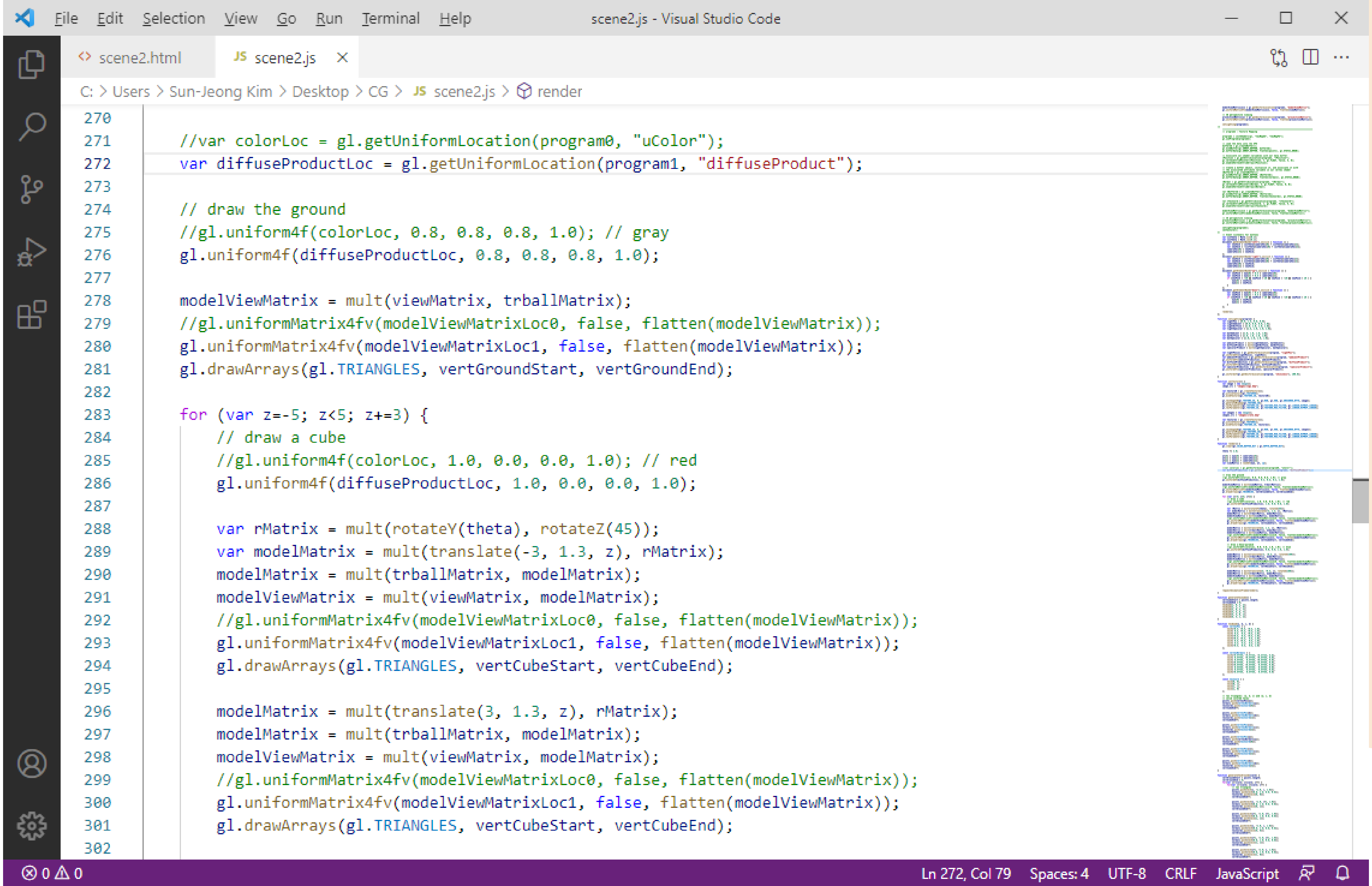

학번 이름
파일 | C:/Users/Sun-Jeong%20Kim/Desktop/CG/scene2.html
▲
◀ ▼ ▶



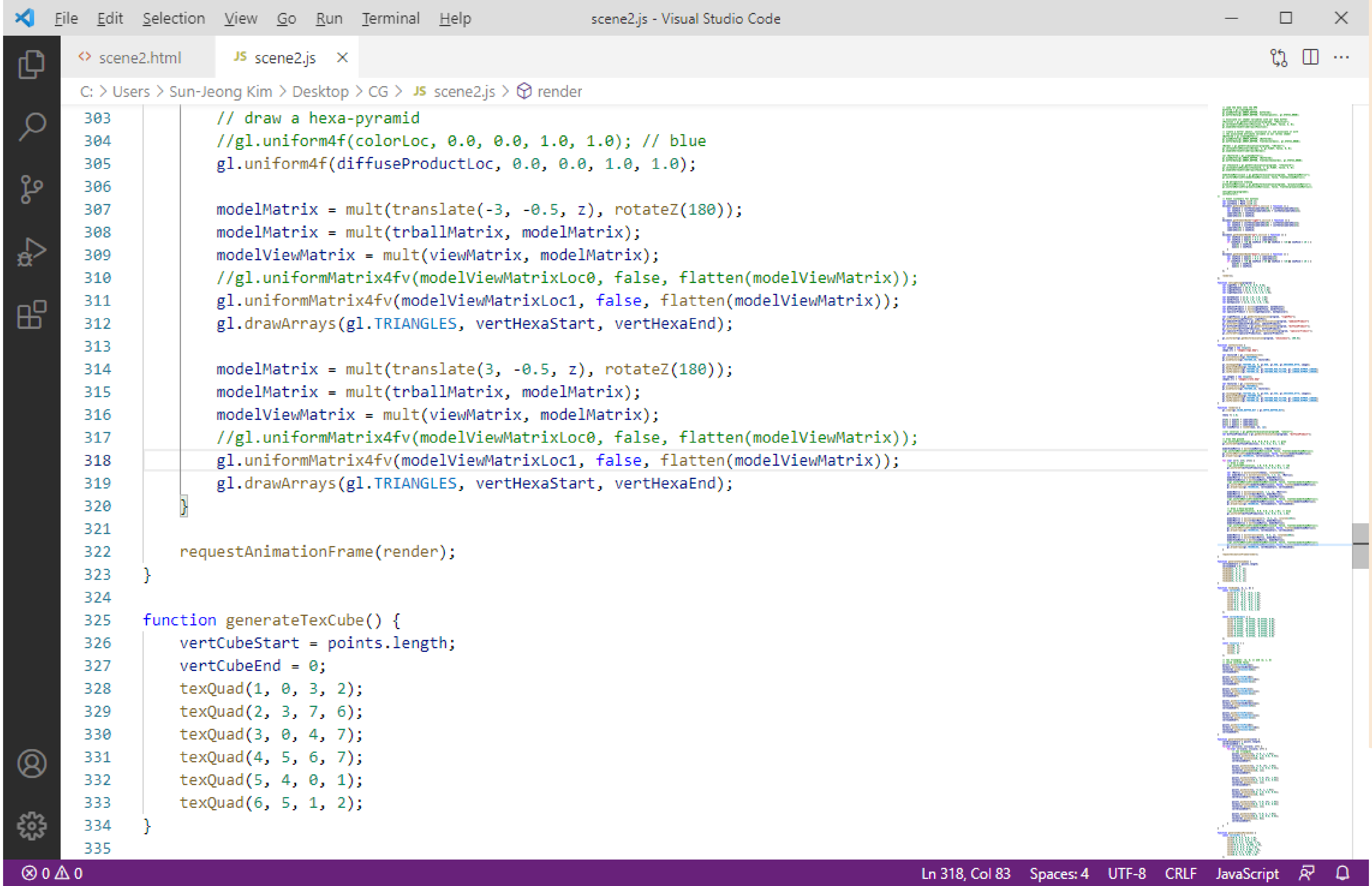
```

    ///////////////////////////////////////////////////////////////////////////
    // program1 : Phong Shading

    program1 = initShaders(gl, "phongVS", "phongFS");
    gl.useProgram(program1);

    // Load the data into the GPU
    bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    vPosition = gl.getAttribLocation(program1, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

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

    var vNormal = gl.getAttribLocation(program1, "vNormal");
    gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vNormal);

    modelViewMatrixLoc1 = gl.getUniformLocation(program1, "modelViewMatrix");
    gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(viewMatrix));

    // 3D perspective viewing
    projectionMatrixLoc = gl.getUniformLocation(program1, "projectionMatrix");
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
```



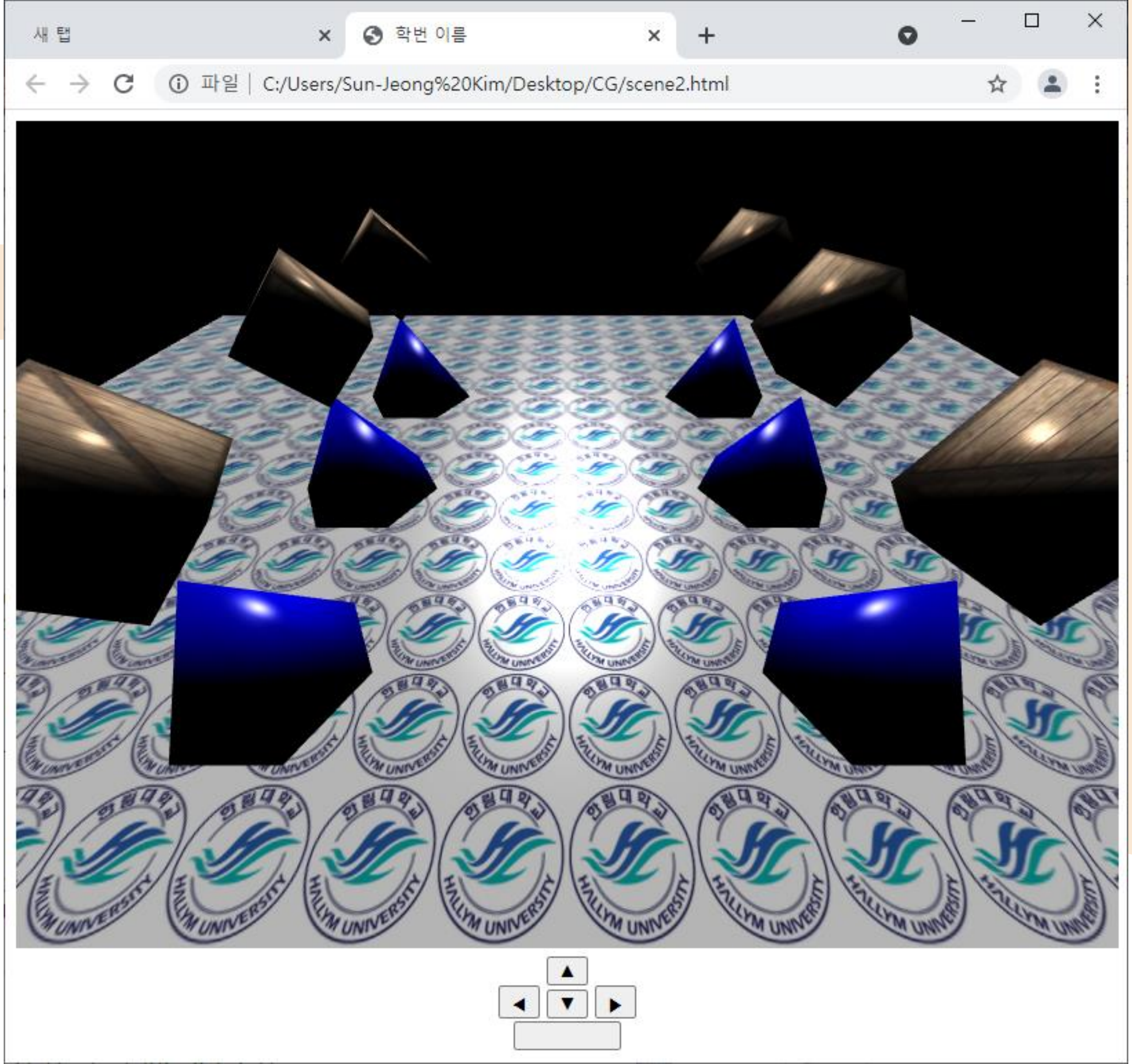
```javascript

    setLighting(program1);
/*
    ///////////////////////////////////////////////////////////////////////////
    // program2 : Texture Mapping

    program2 = initShaders(gl, "texMapVS", "texMapFS");
    gl.useProgram(program2);

    // Load the data into the GPU
    bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    vPosition = gl.getAttribLocation(program2, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

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

    vNormal = gl.getAttribLocation(program2, "vNormal");
    gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vNormal);

    var tBufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, tBufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoords), gl.STATIC_DRAW);

```

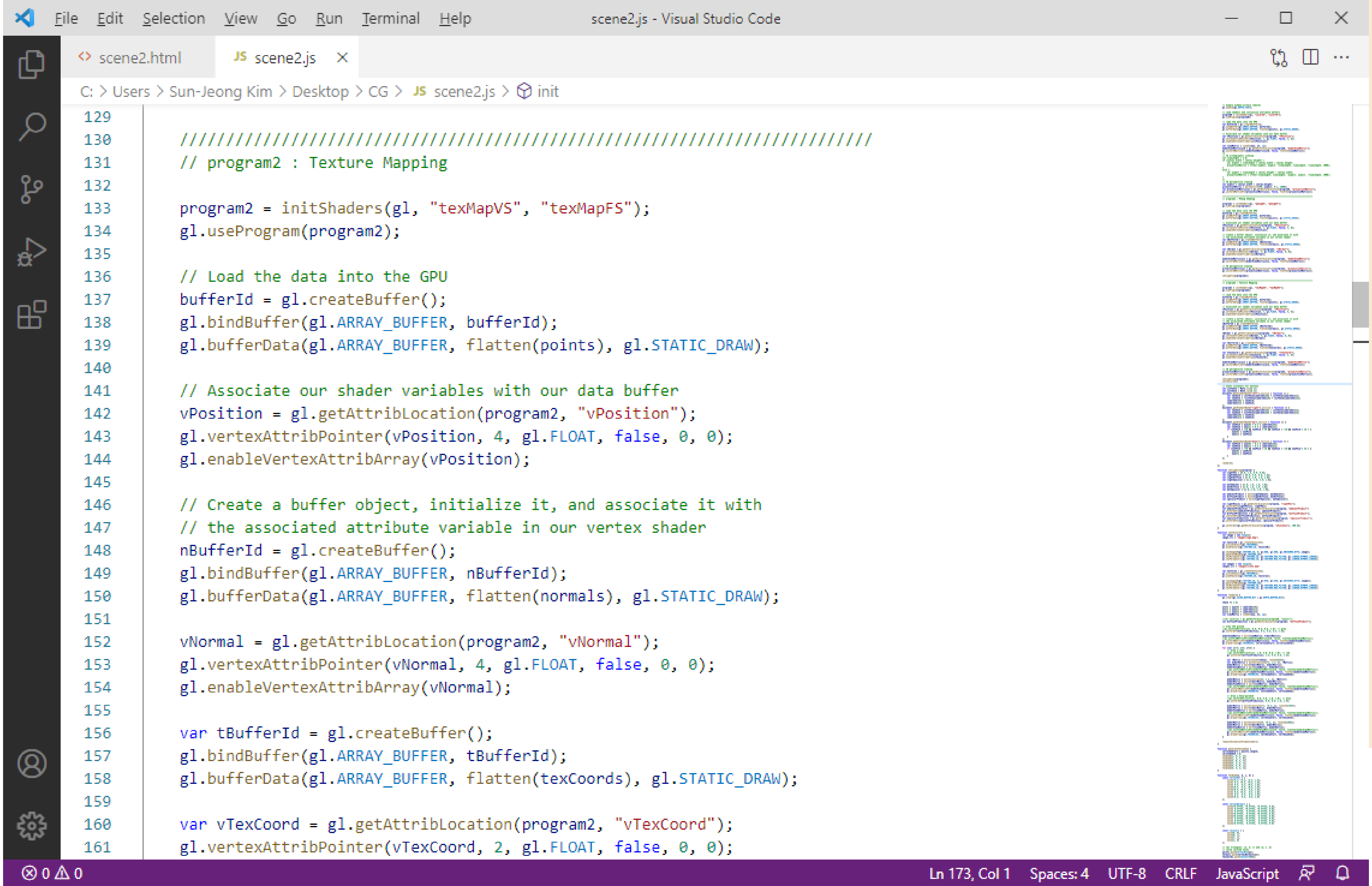

File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > render
0 0
Ln 272, Col 79 Spaces: 4 UTF-8 CRLF JavaScript

```javascript

    //var colorLoc = gl.getUniformLocation(program0, "uColor");
    var diffuseProductLoc = gl.getUniformLocation(program1, "diffuseProduct");

    // draw the ground
    //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
    gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);

    modelViewMatrix = mult(viewMatrix, trballMatrix);
    //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
    gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);

    for (var z=-5; z<5; z+=3) {
        // draw a cube
        //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
        gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);

        var rMatrix = mult(rotateY(theta), rotateZ(45));
        var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

        modelMatrix = mult(translate(3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

```

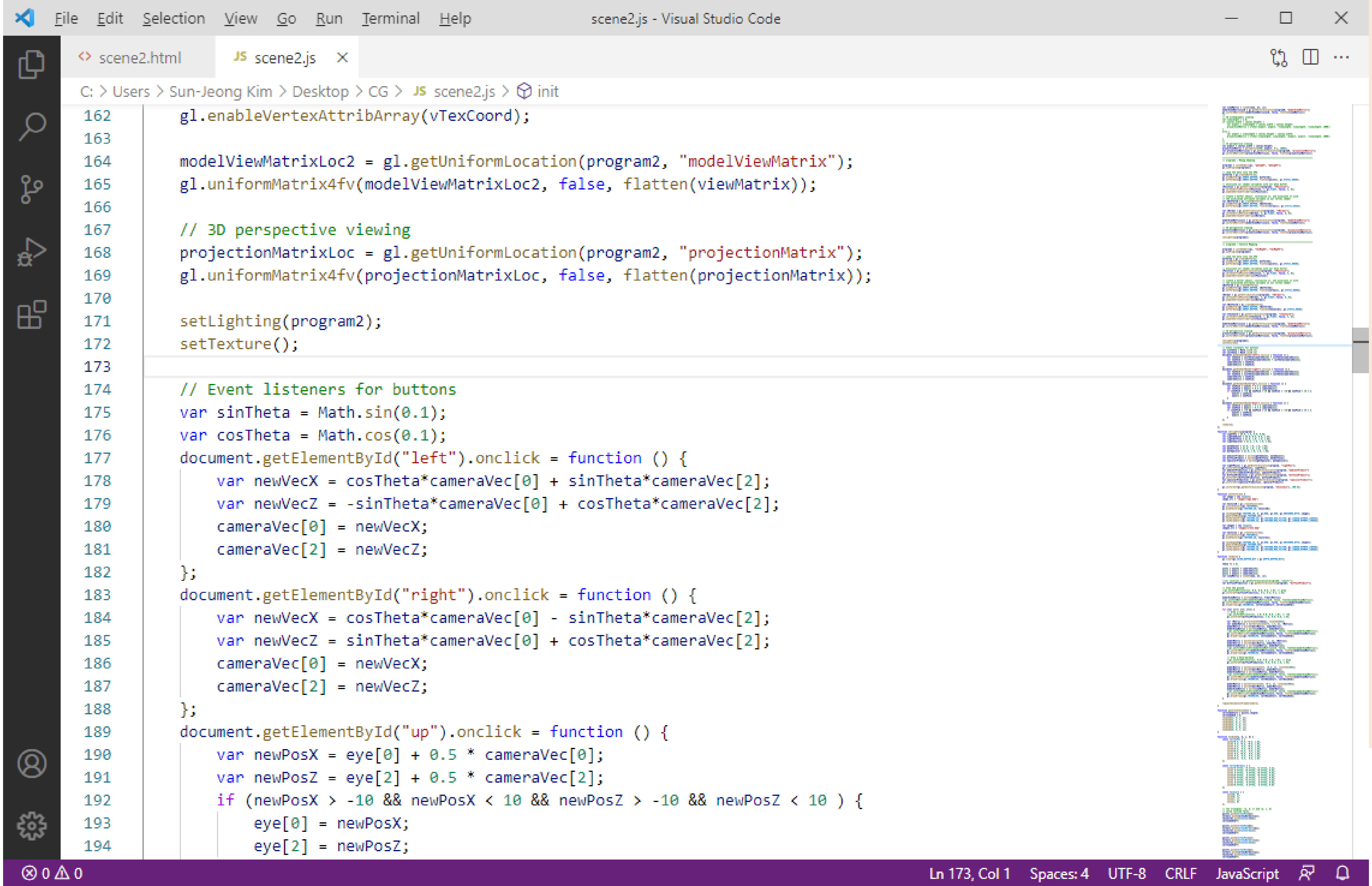


```javascript
        // draw a hexa-pyramid
        //gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 1.0); // blue
        gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);

        modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
    }

    requestAnimationFrame(render);
}

function generateTexCube() {
    vertCubeStart = points.length;
    vertCubeEnd = 0;
    texQuad(1, 0, 3, 2);
    texQuad(2, 3, 7, 6);
    texQuad(3, 0, 4, 7);
    texQuad(4, 5, 6, 7);
    texQuad(5, 4, 0, 1);
    texQuad(6, 5, 1, 2);
}
```

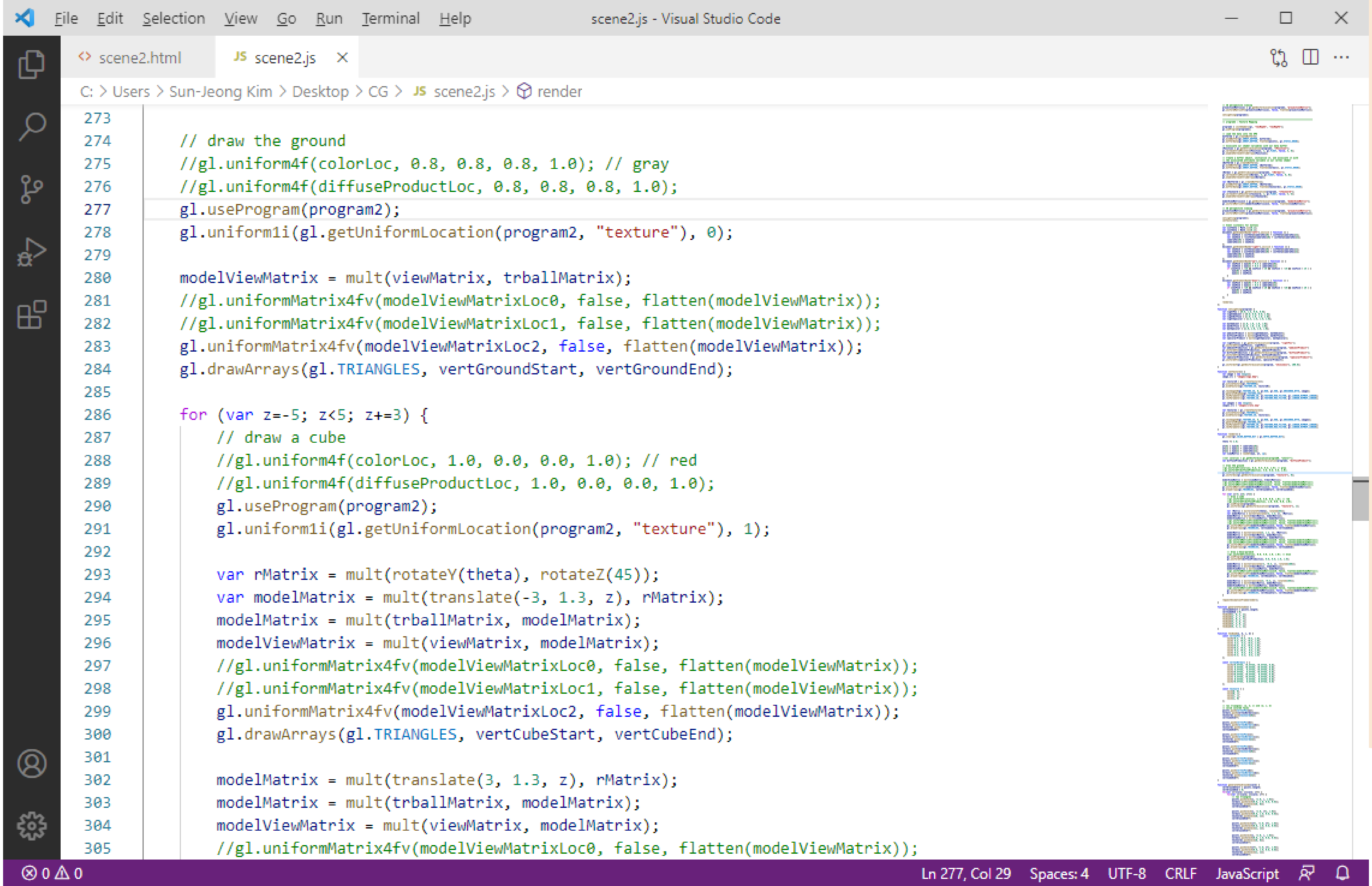

새 탭
학번 이름
파일 | C:/Users/Sun-Jeong%20Kim/Desktop/CG/scene2.html
▲
◀ ▼ ▶



```

    //////////////////////////////////////////////////////////////////////////////
    // program2 : Texture Mapping

    program2 = initShaders(gl, "texMapVS", "texMapFS");
    gl.useProgram(program2);

    // Load the data into the GPU
    bufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

    // Associate our shader variables with our data buffer
    vPosition = gl.getAttribLocation(program2, "vPosition");
    gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

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

    vNormal = gl.getAttribLocation(program2, "vNormal");
    gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vNormal);

    var tBufferId = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, tBufferId);
    gl.bufferData(gl.ARRAY_BUFFER, flatten(texCoords), gl.STATIC_DRAW);

    var vTexCoord = gl.getAttribLocation(program2, "vTexCoord");
    gl.vertexAttribPointer(vTexCoord, 2, gl.FLOAT, false, 0, 0);
```

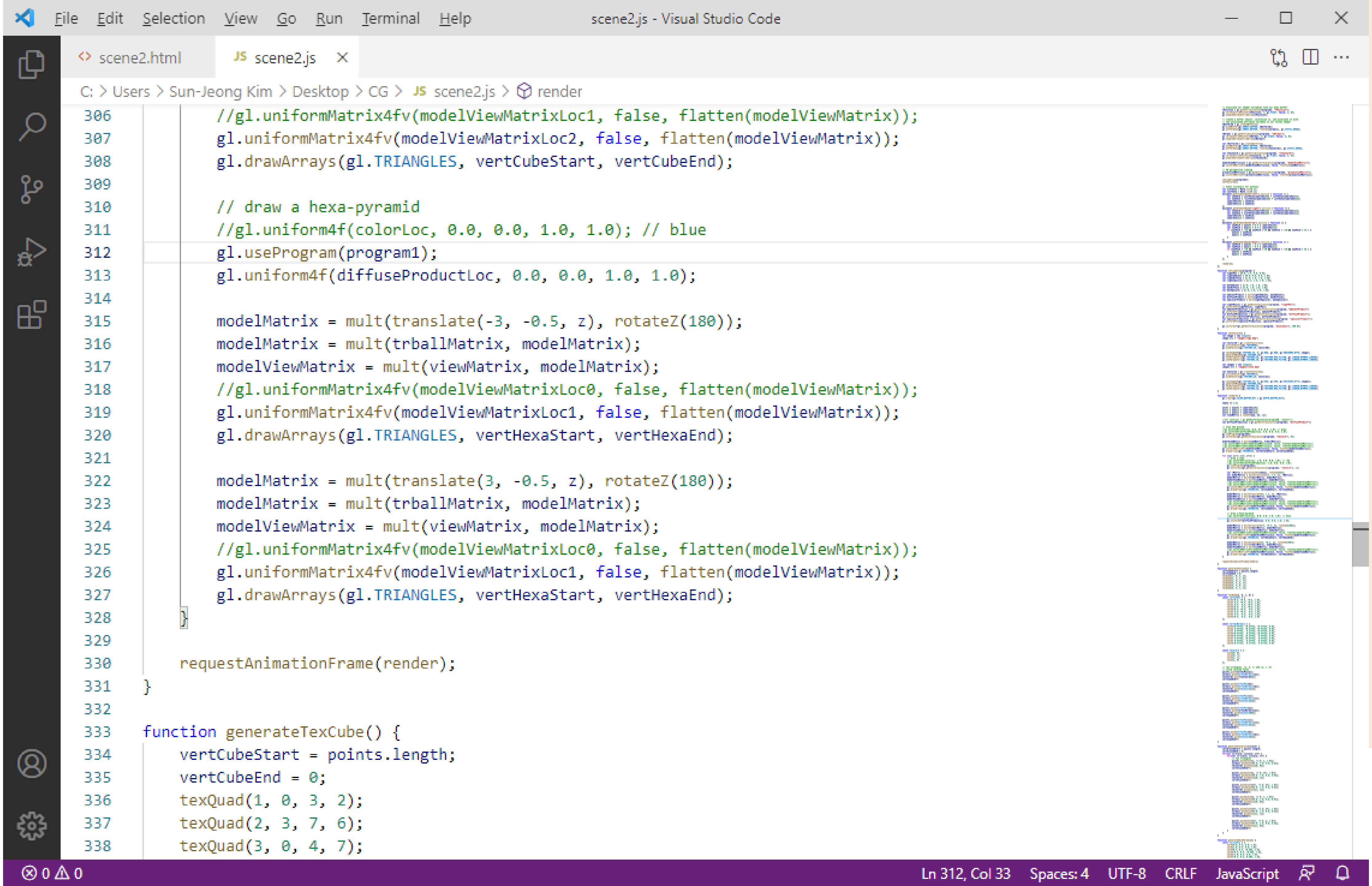


```javascript
    gl.enableVertexAttribArray(vTexCoord);

    modelViewMatrixLoc2 = gl.getUniformLocation(program2, "modelViewMatrix");
    gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(viewMatrix));

    // 3D perspective viewing
    projectionMatrixLoc = gl.getUniformLocation(program2, "projectionMatrix");
    gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));

    setLighting(program2);
    setTexture();

    // Event listeners for buttons
    var sinTheta = Math.sin(0.1);
    var cosTheta = Math.cos(0.1);
    document.getElementById("left").onclick = function () {
        var newVecX = cosTheta*cameraVec[0] + sinTheta*cameraVec[2];
        var newVecZ = -sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
        cameraVec[0] = newVecX;
        cameraVec[2] = newVecZ;
    };
    document.getElementById("right").onclick = function () {
        var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
        var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
        cameraVec[0] = newVecX;
        cameraVec[2] = newVecZ;
    };
    document.getElementById("up").onclick = function () {
        var newPosX = eye[0] + 0.5 * cameraVec[0];
        var newPosZ = eye[2] + 0.5 * cameraVec[2];
        if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 ) {
            eye[0] = newPosX;
            eye[2] = newPosZ;
```



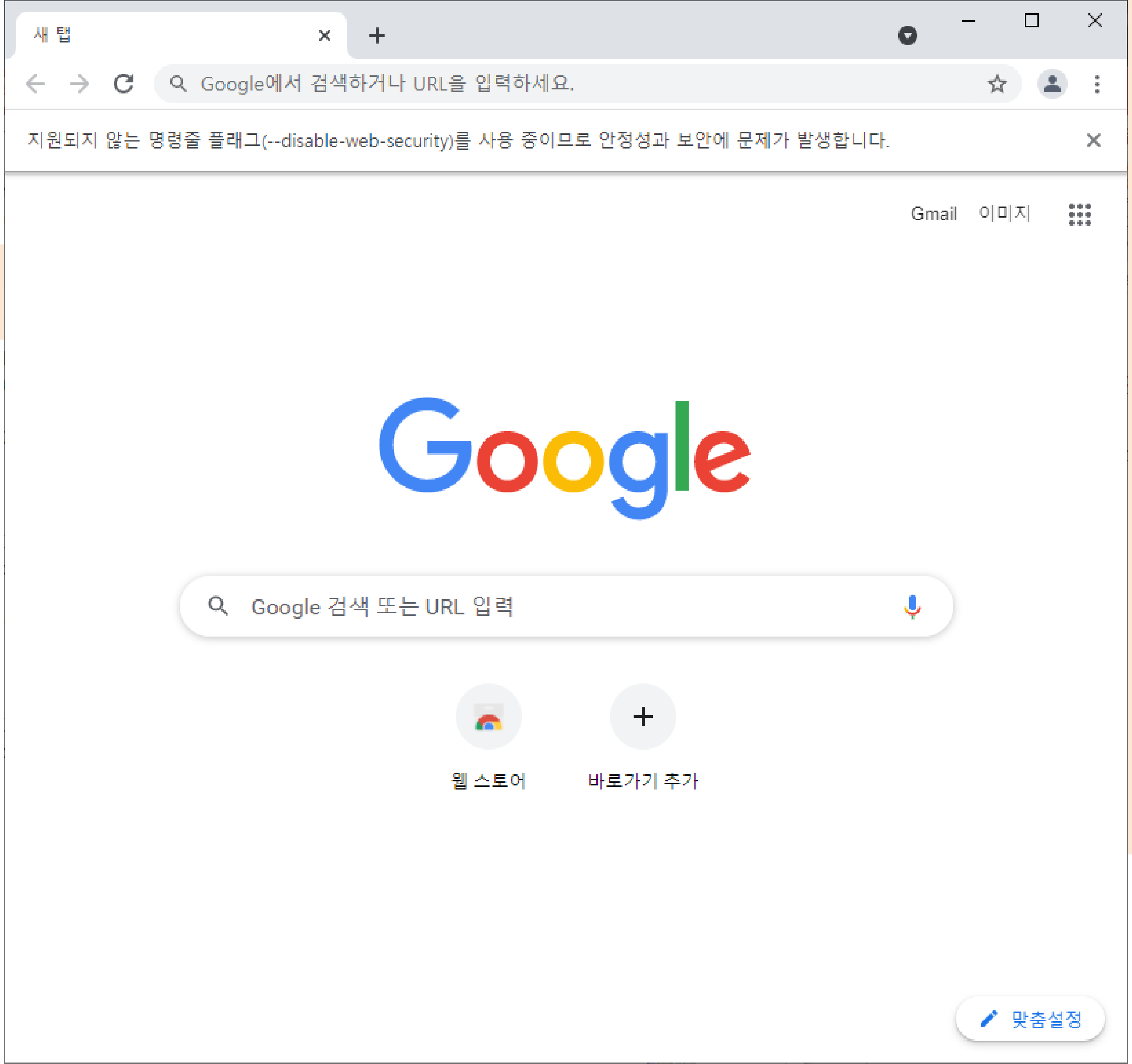
File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > render
Ln 277, Col 29
Spaces: 4
UTF-8
CRLF
JavaScript

```javascript

    // draw the ground
    //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
    //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
    gl.useProgram(program2);
    gl.uniform1i(gl.getUniformLocation(program2, "texture"), 0);

    modelViewMatrix = mult(viewMatrix, trballMatrix);
    //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
    //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
    gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);

    for (var z=-5; z<5; z+=3) {
        // draw a cube
        //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
        //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
        gl.useProgram(program2);
        gl.uniform1i(gl.getUniformLocation(program2, "texture"), 1);

        var rMatrix = mult(rotateY(theta), rotateZ(45));
        var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

        modelMatrix = mult(translate(3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
```



File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > render
0 0
Ln 312, Col 33
Spaces: 4
UTF-8
CRLF
JavaScript

```javascript
        //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

        // draw a hexa-pyramid
        //gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 1.0); // blue
        gl.useProgram(program1);
        gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);

        modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);
    }

    requestAnimationFrame(render);
}

function generateTexCube() {
    vertCubeStart = points.length;
    vertCubeEnd = 0;
    texQuad(1, 0, 3, 2);
    texQuad(2, 3, 7, 6);
    texQuad(3, 0, 4, 7);
```

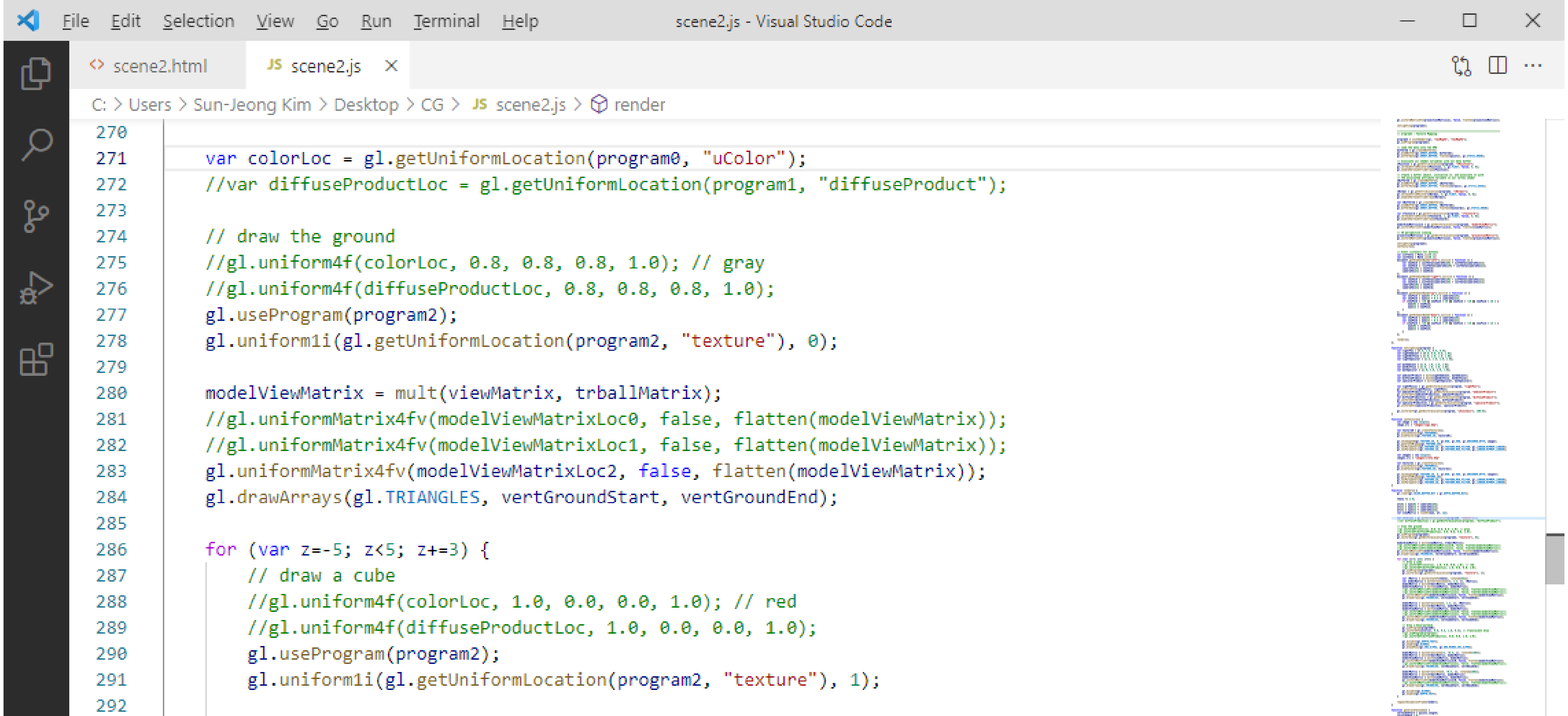


새 탭
Google에서 검색하거나 URL을 입력하세요.

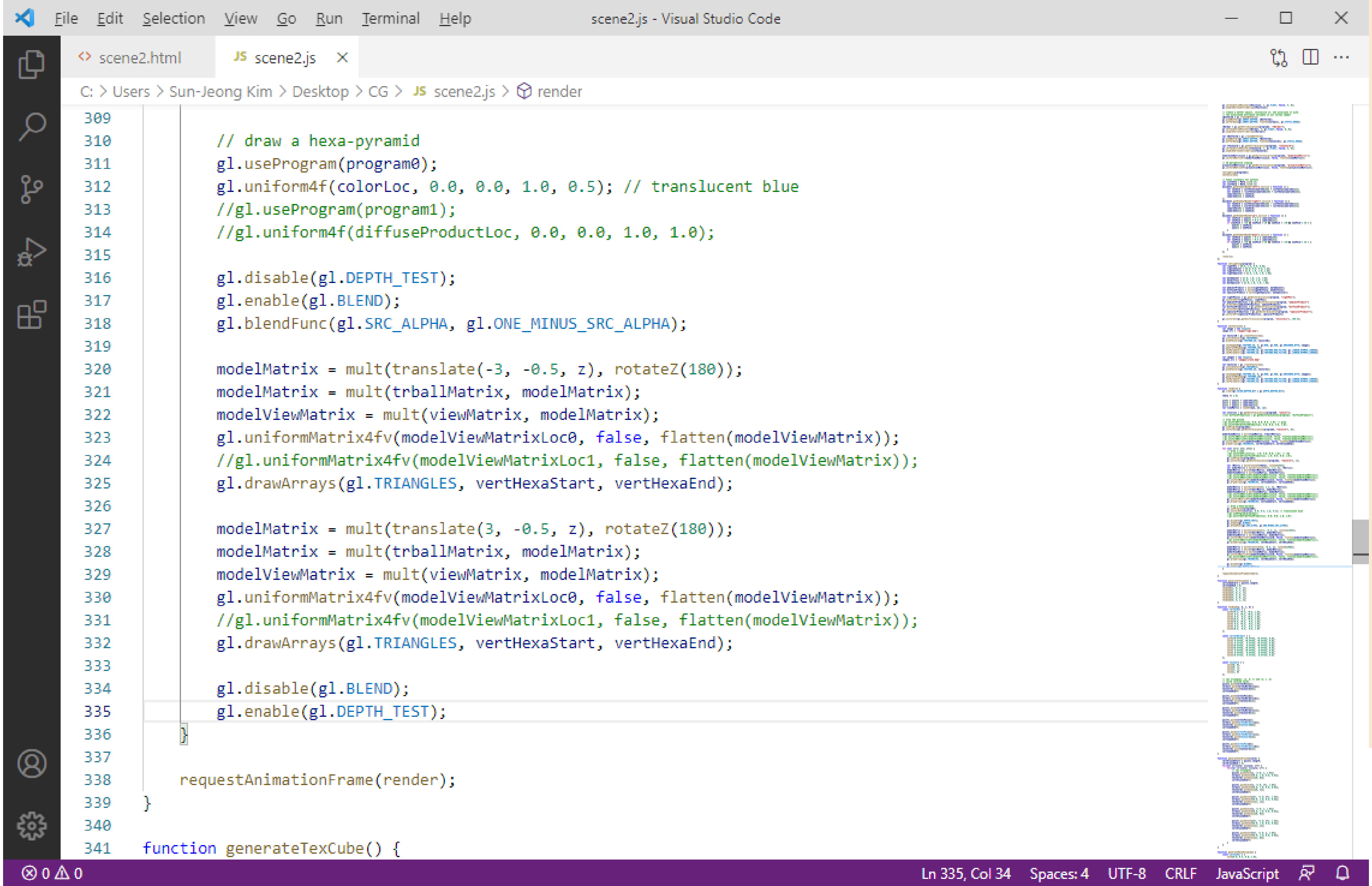
지원되지 않는 명령줄 플래그(--disable-web-security)를 사용 중이므로 안정성과 보안에 문제가 발생합니다.

Gmail
이미지
Google
Google 검색 또는 URL 입력
웹 스토어
바로가기 추가
맞춤설정

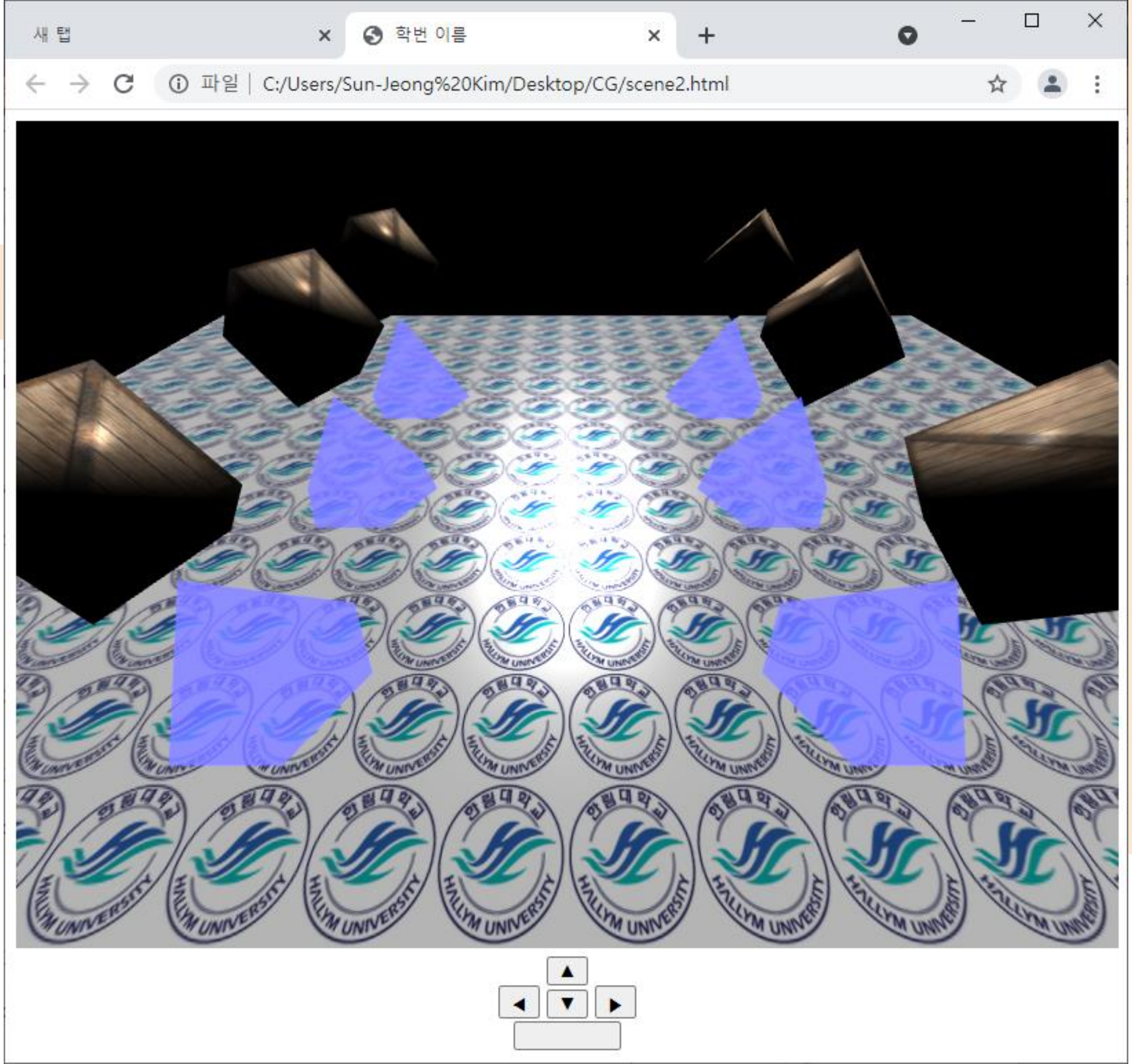
Alpha Blending


File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene2.js > render

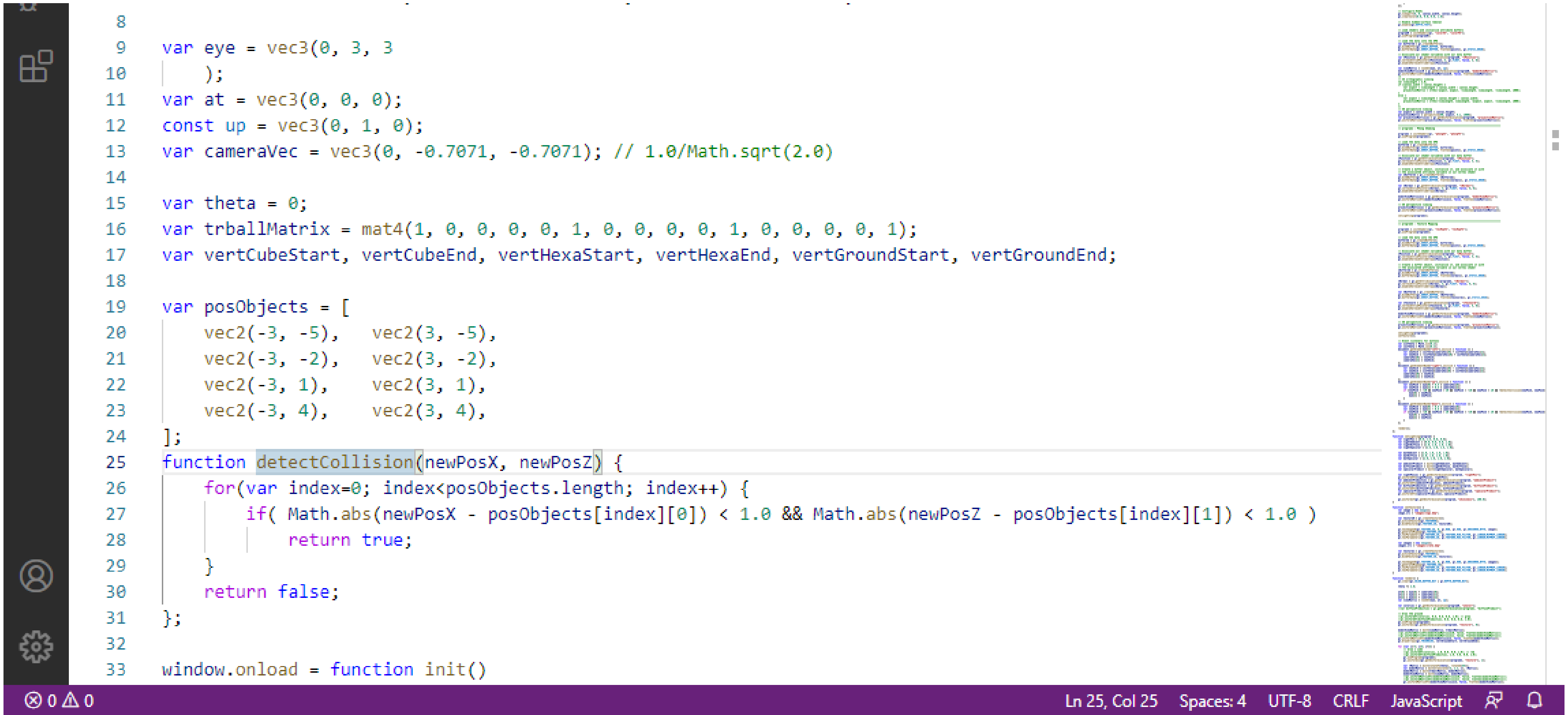
```

    var colorLoc = gl.getUniformLocation(program0, "uColor");
    //var diffuseProductLoc = gl.getUniformLocation(program1, "diffuseProduct");

    // draw the ground
    //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
    //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
    gl.useProgram(program2);
    gl.uniform1i(gl.getUniformLocation(program2, "texture"), 0);

    modelViewMatrix = mult(viewMatrix, trballMatrix);
    //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
    //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
    gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);

    for (var z=-5; z<5; z+=3) {
        // draw a cube
        //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
        //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
        gl.useProgram(program2);
        gl.uniform1i(gl.getUniformLocation(program2, "texture"), 1);

```



```

        // draw a hexa-pyramid
        gl.useProgram(program0);
        gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 0.5); // translucent blue
        //gl.useProgram(program1);
        //gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);

        gl.disable(gl.DEPTH_TEST);
        gl.enable(gl.BLEND);
        gl.blendFunc(gl.SRC_ALPHA, gl.ONE_MINUS_SRC_ALPHA);

        modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        gl.disable(gl.BLEND);
        gl.enable(gl.DEPTH_TEST);
    }

    requestAnimationFrame(render);
}

function generateTexCube() {
```

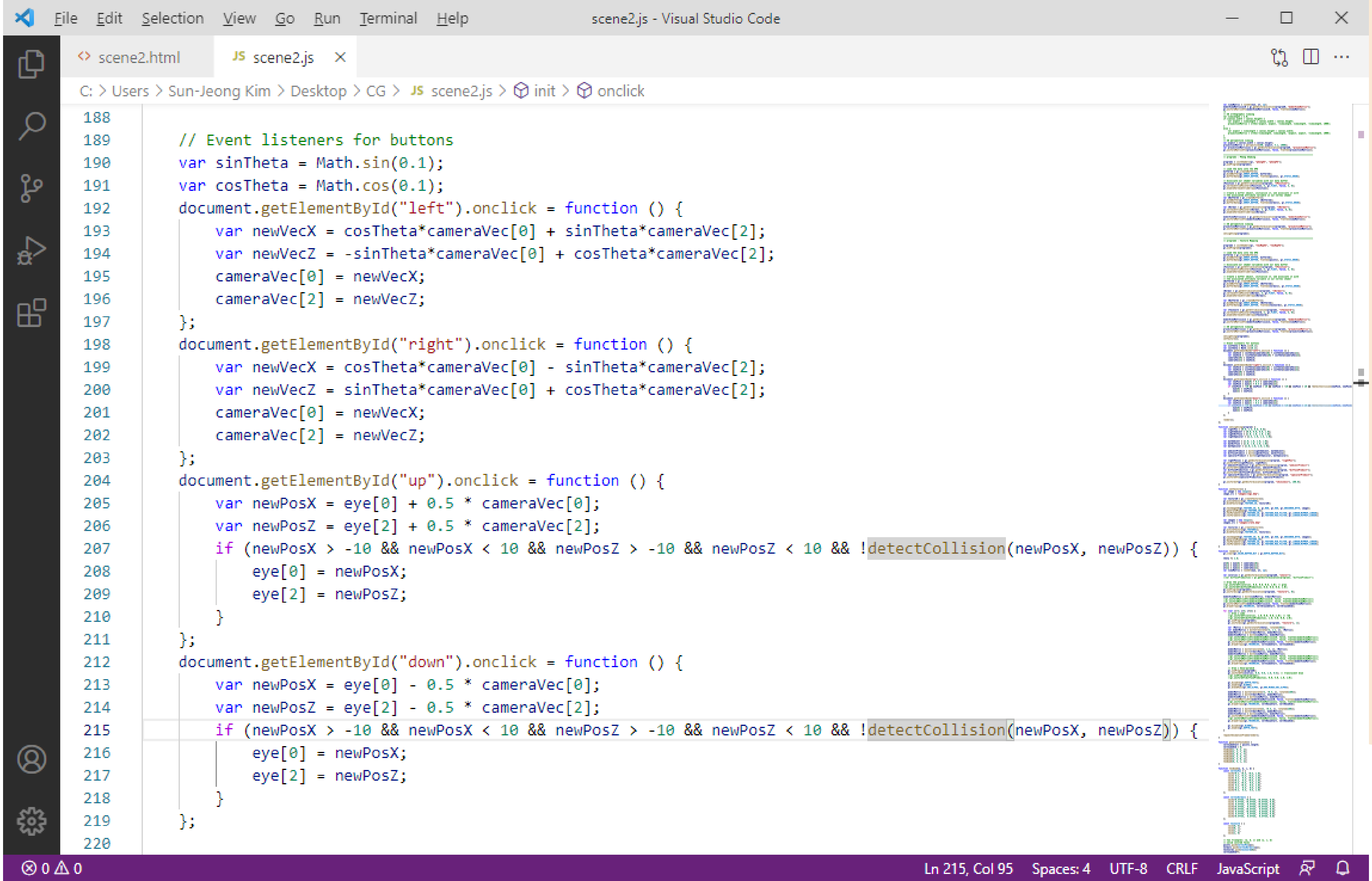

새 탭
학번 이름
파일 | C:/Users/Sun-Jeong%20Kim/Desktop/CG/scene2.html
▲
◀
▼
▶

Collision Detection

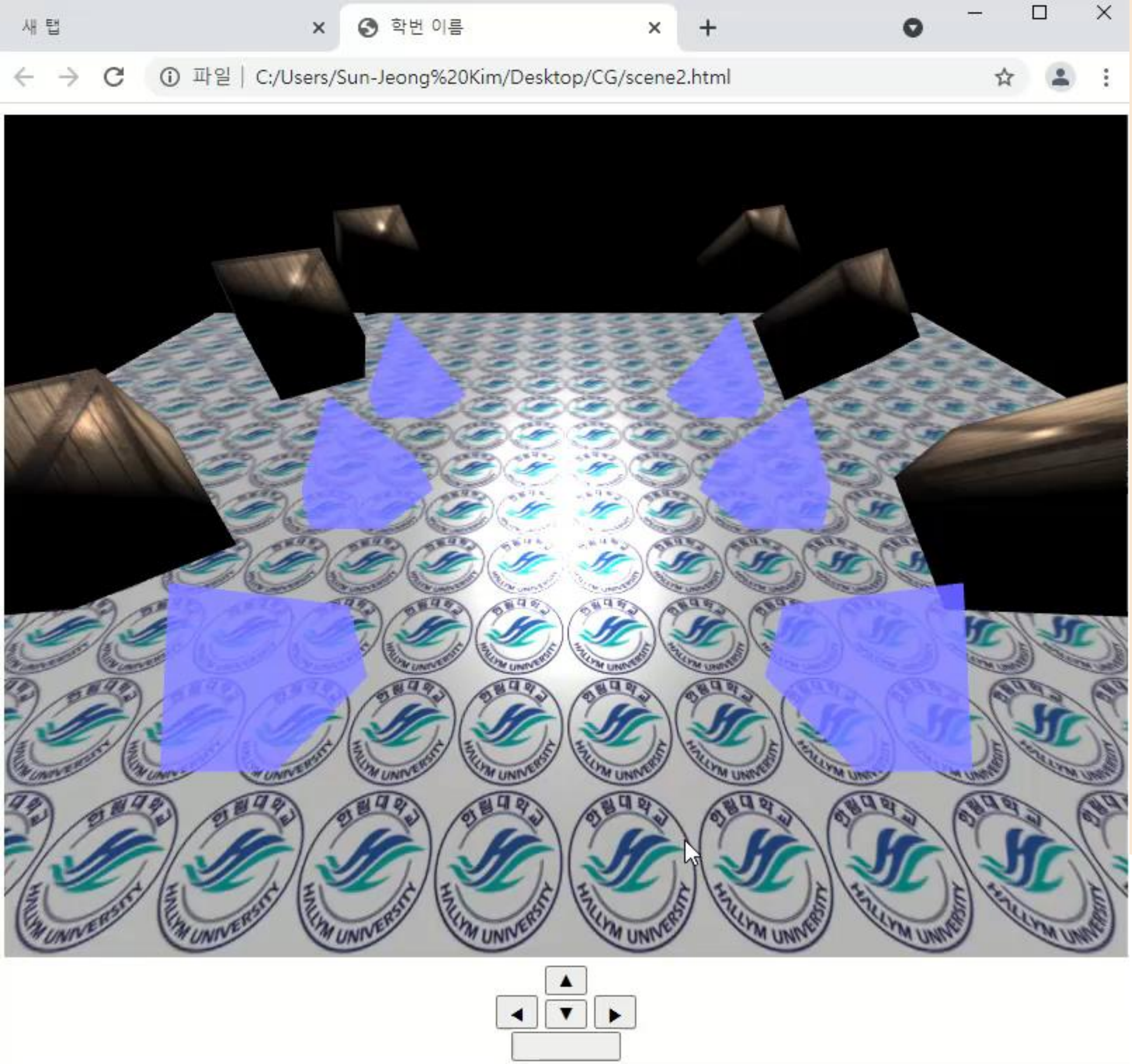


```
var eye = vec3(0, 3, 3
    );
var at = vec3(0, 0, 0);
const up = vec3(0, 1, 0);
var cameraVec = vec3(0, -0.7071, -0.7071); // 1.0/Math.sqrt(2.0)

var theta = 0;
var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
var vertCubeStart, vertCubeEnd, vertHexaStart, vertHexaEnd, vertGroundStart, vertGroundEnd;

var posObjects = [
    vec2(-3, -5),   vec2(3, -5),
    vec2(-3, -2),   vec2(3, -2),
    vec2(-3, 1),    vec2(3, 1),
    vec2(-3, 4),    vec2(3, 4),
];
function detectCollision(newPosX, newPosZ) {
    for(var index=0; index<posObjects.length; index++) {
        if( Math.abs(newPosX - posObjects[index][0]) < 1.0 && Math.abs(newPosZ - posObjects[index][1]) < 1.0 )
            return true;
    }
    return false;
};

window.onload = function init()
```

⊗ 0 ⚠ 0   Ln 25, Col 25   Spaces: 4   UTF-8   CRLF   JavaScript

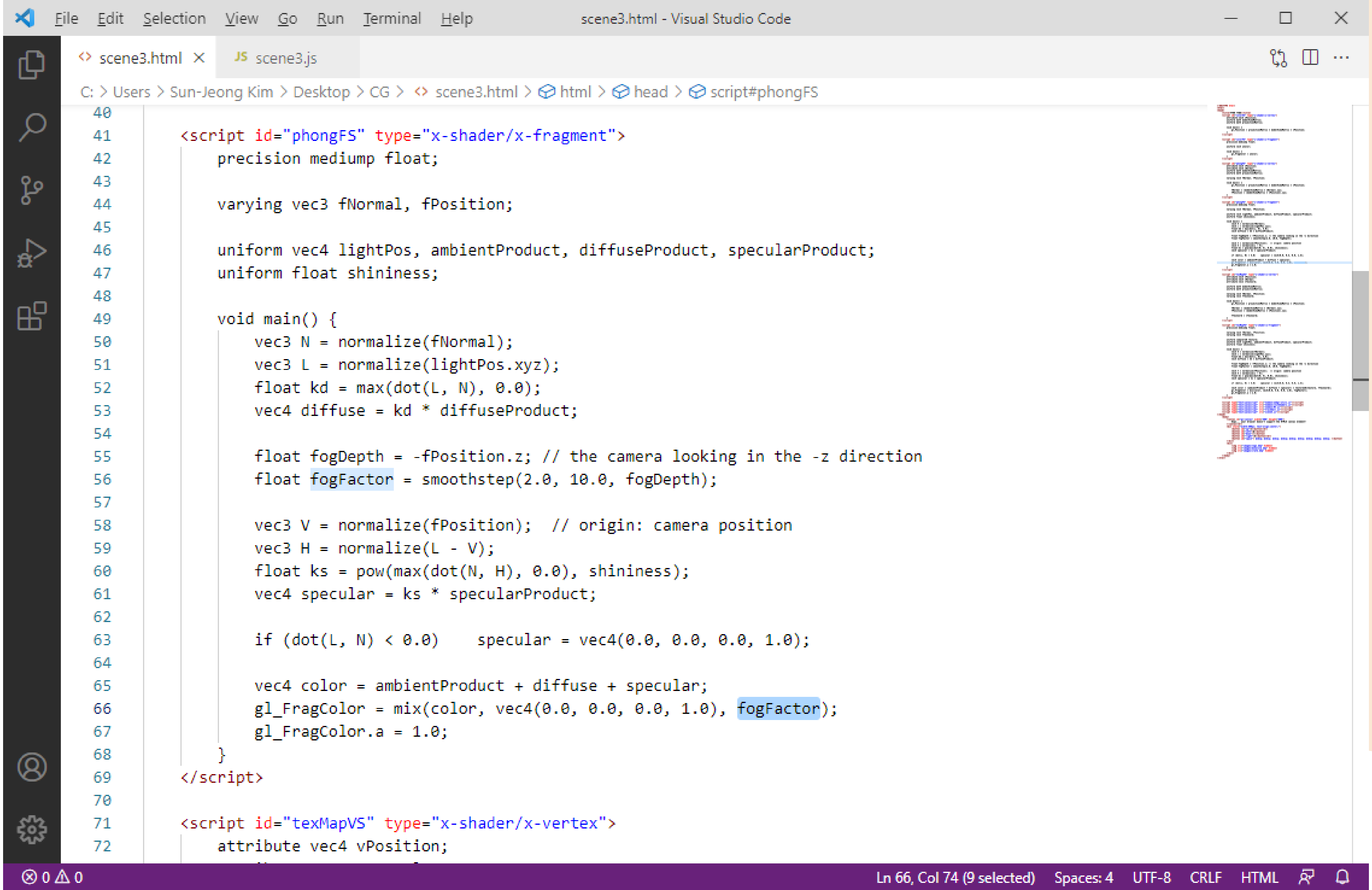

File Edit Selection View Go Run Terminal Help
scene2.js - Visual Studio Code
scene2.html
JS scene2.js
C: > Users > Sun-Jeong Kim > Desktop > CG > JS scene2.js > init > onclick
⊗ 0 ⚠ 0
Ln 215, Col 95
Spaces: 4
UTF-8
CRLF
JavaScript

```

    // Event listeners for buttons
    var sinTheta = Math.sin(0.1);
    var cosTheta = Math.cos(0.1);
    document.getElementById("left").onclick = function () {
        var newVecX = cosTheta*cameraVec[0] + sinTheta*cameraVec[2];
        var newVecZ = -sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
        cameraVec[0] = newVecX;
        cameraVec[2] = newVecZ;
    };
    document.getElementById("right").onclick = function () {
        var newVecX = cosTheta*cameraVec[0] - sinTheta*cameraVec[2];
        var newVecZ = sinTheta*cameraVec[0] + cosTheta*cameraVec[2];
        cameraVec[0] = newVecX;
        cameraVec[2] = newVecZ;
    };
    document.getElementById("up").onclick = function () {
        var newPosX = eye[0] + 0.5 * cameraVec[0];
        var newPosZ = eye[2] + 0.5 * cameraVec[2];
        if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 && !detectCollision(newPosX, newPosZ)) {
            eye[0] = newPosX;
            eye[2] = newPosZ;
        }
    };
    document.getElementById("down").onclick = function () {
        var newPosX = eye[0] - 0.5 * cameraVec[0];
        var newPosZ = eye[2] - 0.5 * cameraVec[2];
        if (newPosX > -10 && newPosX < 10 && newPosZ > -10 && newPosZ < 10 && !detectCollision(newPosX, newPosZ)) {
            eye[0] = newPosX;
            eye[2] = newPosZ;
        }
    };

```

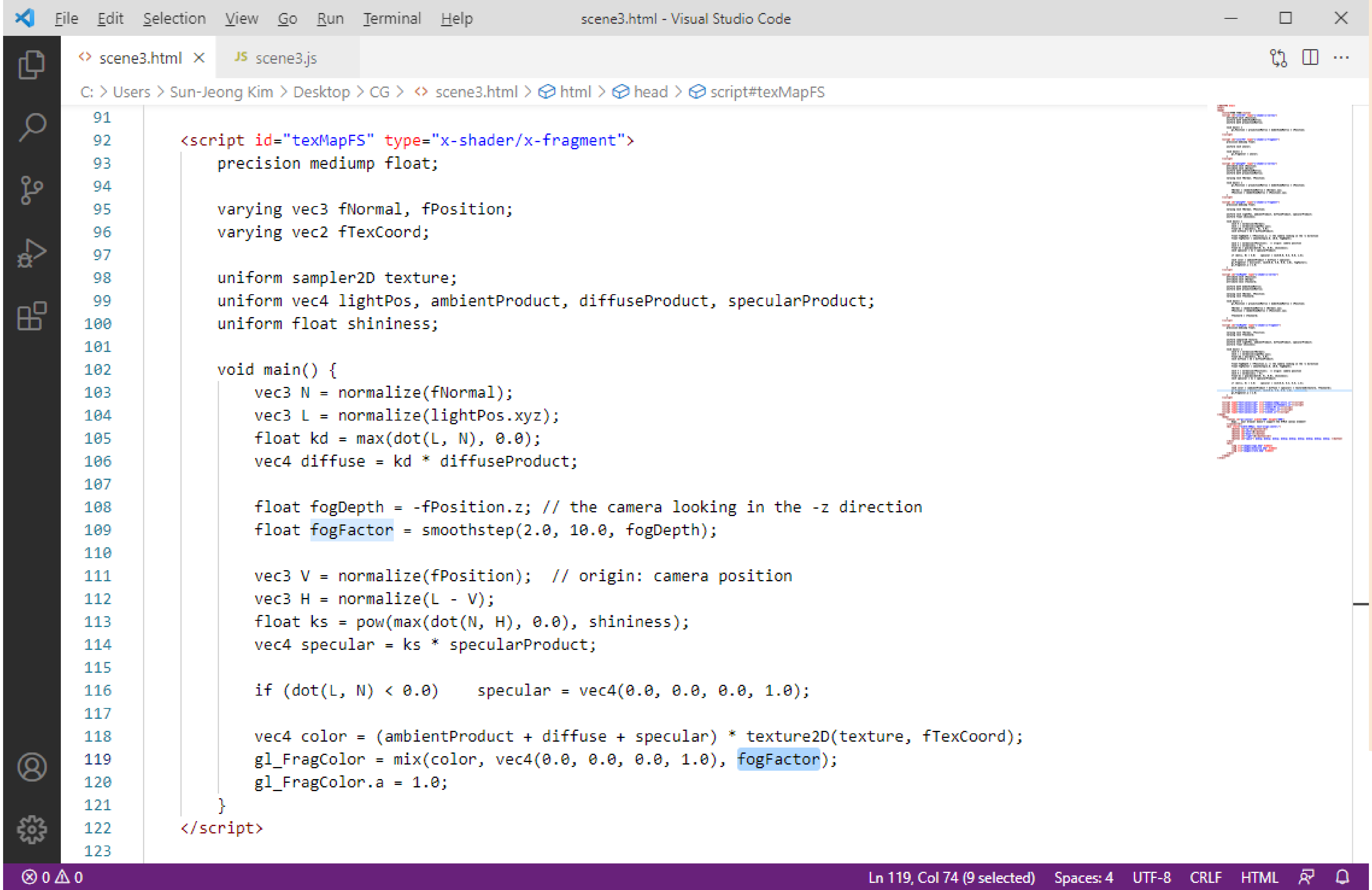

새 탭
학번 이름
파일
C:/Users/Sun-Jeong%20Kim/Desktop/CG/scene2.html
▲
◀ ▼ ▶

연습 문제

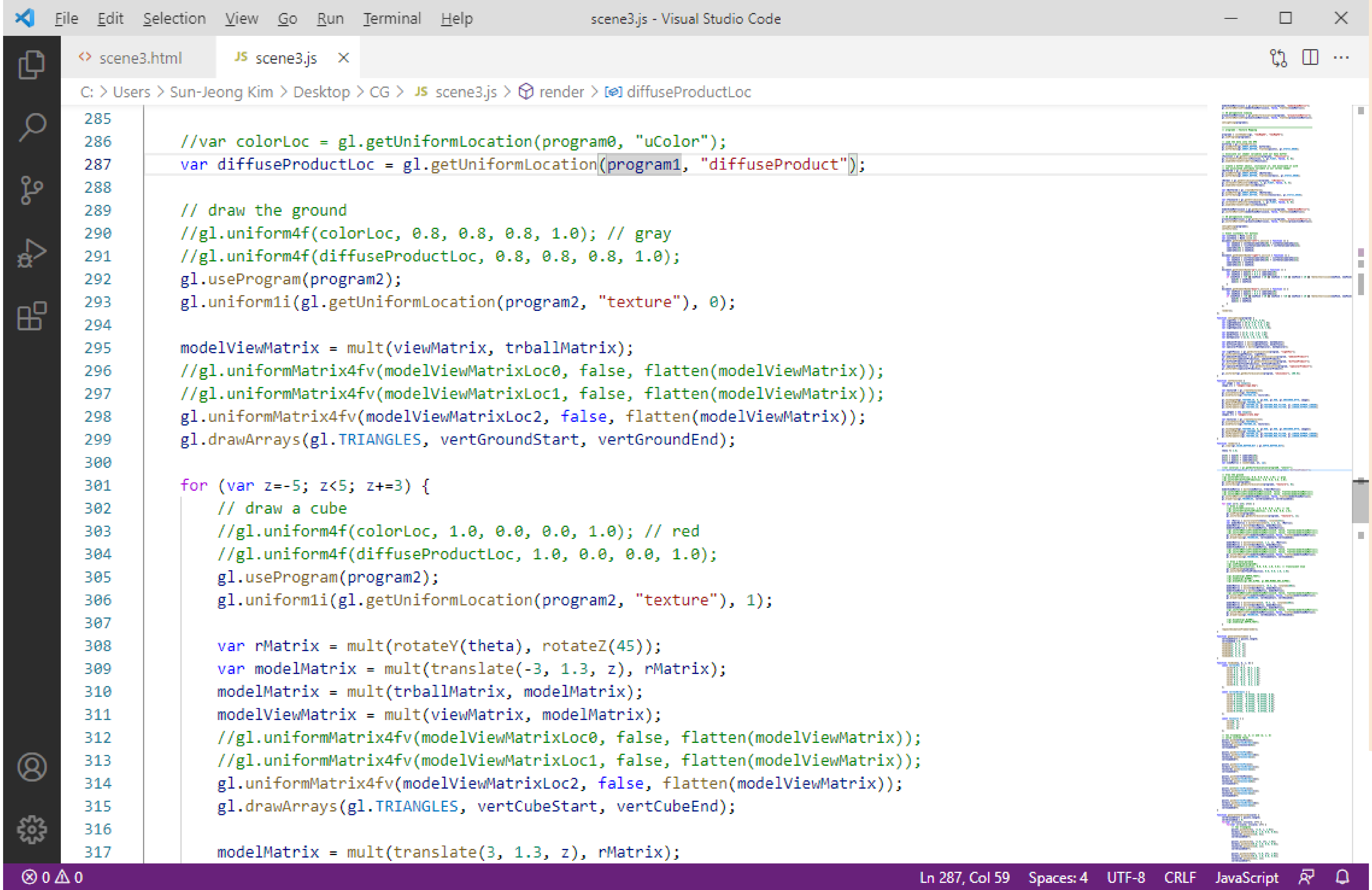
- 카메라 위치와 Vertex Position 사이의 거리를 Blending Factor로 하여 Fog 색상을 Blending 하는 shader를 작성하시오.


File
Edit
Selection
View
Go
Run
Terminal
Help
scene3.html - Visual Studio Code
scene3.html
scene3.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene3.html > html > head > script#phongFS
⊗ 0 ⚠ 0
Ln 66, Col 74 (9 selected)
Spaces: 4
UTF-8
CRLF
HTML

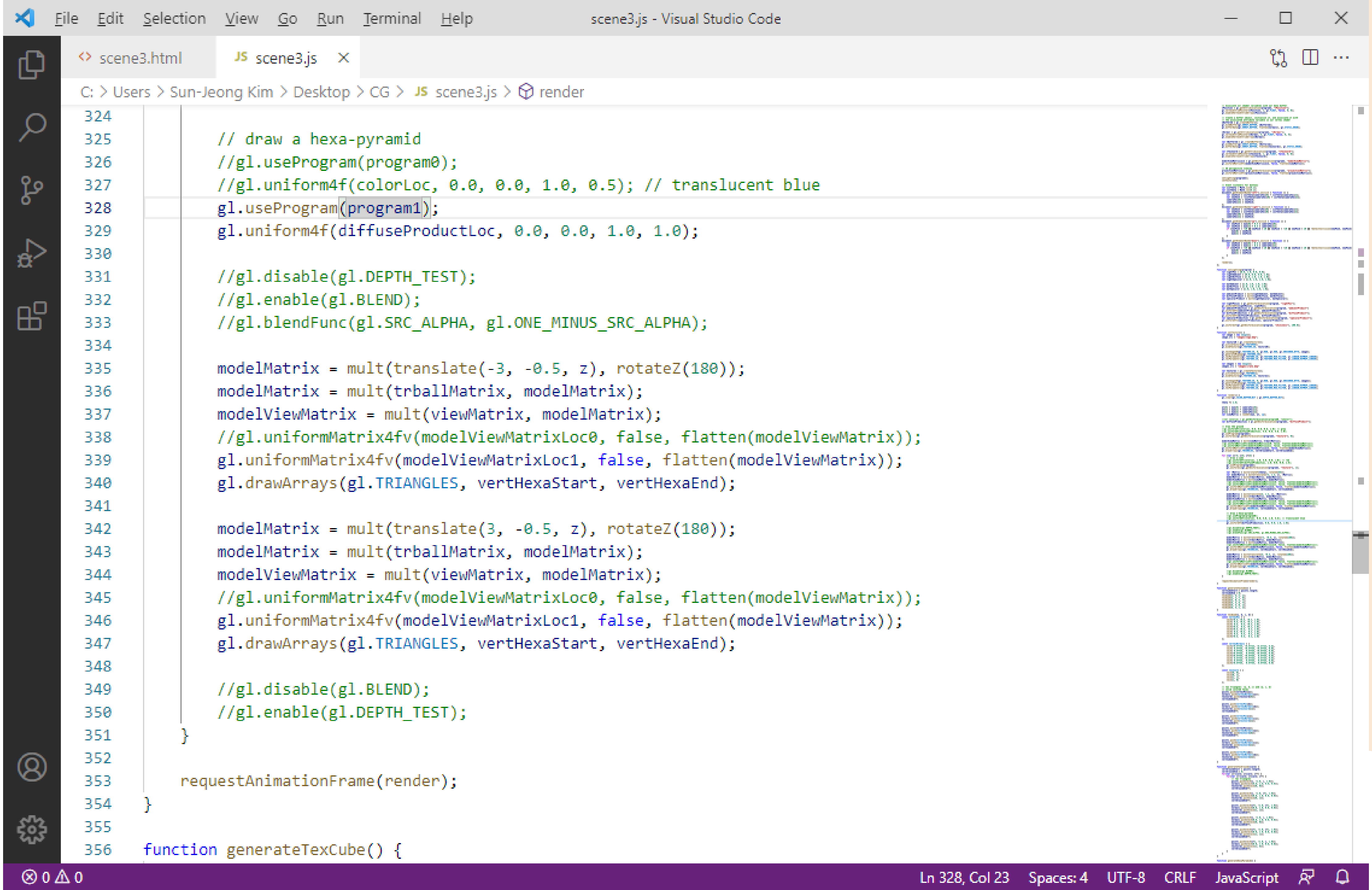
```

    <script id="phongFS" type="x-shader/x-fragment">
        precision mediump float;

        varying vec3 fNormal, fPosition;

        uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
        uniform float shininess;

        void main() {
            vec3 N = normalize(fNormal);
            vec3 L = normalize(lightPos.xyz);
            float kd = max(dot(L, N), 0.0);
            vec4 diffuse = kd * diffuseProduct;

            float fogDepth = -fPosition.z; // the camera looking in the -z direction
            float fogFactor = smoothstep(2.0, 10.0, fogDepth);

            vec3 V = normalize(fPosition);  // origin: camera position
            vec3 H = normalize(L - V);
            float ks = pow(max(dot(N, H), 0.0), shininess);
            vec4 specular = ks * specularProduct;

            if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);

            vec4 color = ambientProduct + diffuse + specular;
            gl_FragColor = mix(color, vec4(0.0, 0.0, 0.0, 1.0), fogFactor);
            gl_FragColor.a = 1.0;
        }
    </script>

    <script id="texMapVS" type="x-shader/x-vertex">
        attribute vec4 vPosition;
```


File
Edit
Selection
View
Go
Run
Terminal
Help
scene3.html - Visual Studio Code
scene3.html
scene3.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene3.html > html > head > script#texMapFS
ⓧ 0 ⚠ 0
Ln 119, Col 74 (9 selected)
Spaces: 4
UTF-8
CRLF
HTML

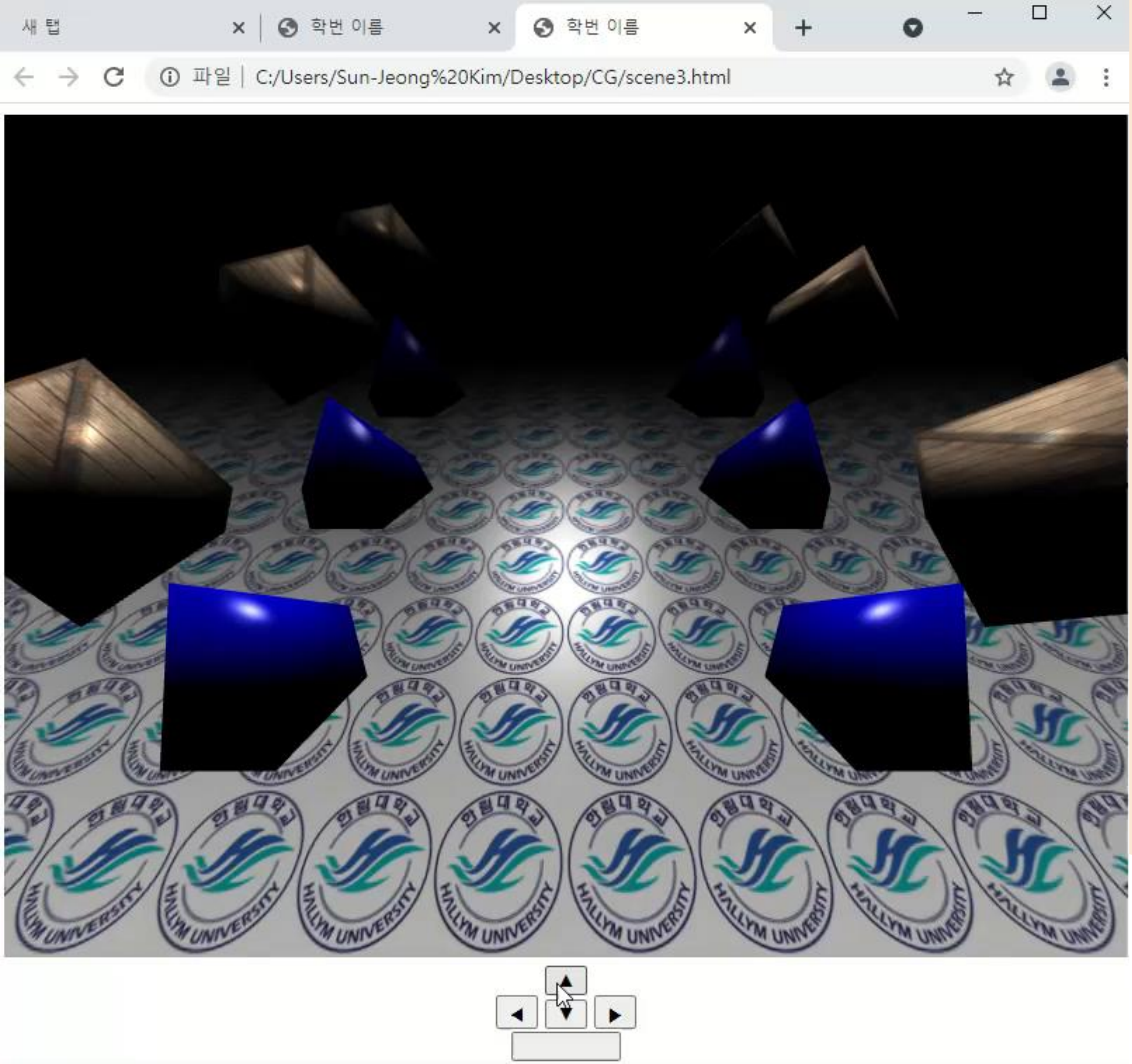
```
    <script id="texMapFS" type="x-shader/x-fragment">
        precision mediump float;

        varying vec3 fNormal, fPosition;
        varying vec2 fTexCoord;

        uniform sampler2D texture;
        uniform vec4 lightPos, ambientProduct, diffuseProduct, specularProduct;
        uniform float shininess;

        void main() {
            vec3 N = normalize(fNormal);
            vec3 L = normalize(lightPos.xyz);
            float kd = max(dot(L, N), 0.0);
            vec4 diffuse = kd * diffuseProduct;

            float fogDepth = -fPosition.z; // the camera looking in the -z direction
            float fogFactor = smoothstep(2.0, 10.0, fogDepth);

            vec3 V = normalize(fPosition);  // origin: camera position
            vec3 H = normalize(L - V);
            float ks = pow(max(dot(N, H), 0.0), shininess);
            vec4 specular = ks * specularProduct;

            if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);

            vec4 color = (ambientProduct + diffuse + specular) * texture2D(texture, fTexCoord);
            gl_FragColor = mix(color, vec4(0.0, 0.0, 0.0, 1.0), fogFactor);
            gl_FragColor.a = 1.0;
        }
    </script>
```


File
Edit
Selection
View
Go
Run
Terminal
Help
scene3.js - Visual Studio Code
scene3.html
scene3.js
C: > Users > Sun-Jeong Kim > Desktop > CG > scene3.js > render > diffuseProductLoc
⊗ 0 ⚠ 0
Ln 287, Col 59
Spaces: 4
UTF-8
CRLF
JavaScript

```

    //var colorLoc = gl.getUniformLocation(program0, "uColor");
    var diffuseProductLoc = gl.getUniformLocation(program1, "diffuseProduct");

    // draw the ground
    //gl.uniform4f(colorLoc, 0.8, 0.8, 0.8, 1.0); // gray
    //gl.uniform4f(diffuseProductLoc, 0.8, 0.8, 0.8, 1.0);
    gl.useProgram(program2);
    gl.uniform1i(gl.getUniformLocation(program2, "texture"), 0);

    modelViewMatrix = mult(viewMatrix, trballMatrix);
    //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
    //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
    gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
    gl.drawArrays(gl.TRIANGLES, vertGroundStart, vertGroundEnd);

    for (var z=-5; z<5; z+=3) {
        // draw a cube
        //gl.uniform4f(colorLoc, 1.0, 0.0, 0.0, 1.0); // red
        //gl.uniform4f(diffuseProductLoc, 1.0, 0.0, 0.0, 1.0);
        gl.useProgram(program2);
        gl.uniform1i(gl.getUniformLocation(program2, "texture"), 1);

        var rMatrix = mult(rotateY(theta), rotateZ(45));
        var modelMatrix = mult(translate(-3, 1.3, z), rMatrix);
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        //gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc2, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertCubeStart, vertCubeEnd);

        modelMatrix = mult(translate(3, 1.3, z), rMatrix);
```



```javascript

        // draw a hexa-pyramid
        //gl.useProgram(program0);
        //gl.uniform4f(colorLoc, 0.0, 0.0, 1.0, 0.5); // translucent blue
        gl.useProgram(program1);
        gl.uniform4f(diffuseProductLoc, 0.0, 0.0, 1.0, 1.0);

        //gl.disable(gl.DEPTH_TEST);
        //gl.enable(gl.BLEND);
        //gl.blendFunc(gl.SRC_ALPHA, gl.ONE_MINUS_SRC_ALPHA);

        modelMatrix = mult(translate(-3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        modelMatrix = mult(translate(3, -0.5, z), rotateZ(180));
        modelMatrix = mult(trballMatrix, modelMatrix);
        modelViewMatrix = mult(viewMatrix, modelMatrix);
        //gl.uniformMatrix4fv(modelViewMatrixLoc0, false, flatten(modelViewMatrix));
        gl.uniformMatrix4fv(modelViewMatrixLoc1, false, flatten(modelViewMatrix));
        gl.drawArrays(gl.TRIANGLES, vertHexaStart, vertHexaEnd);

        //gl.disable(gl.BLEND);
        //gl.enable(gl.DEPTH_TEST);
    }

    requestAnimationFrame(render);
}

function generateTexCube() {
```


새 탭
학번 이름
학번 이름
파일
C:/Users/Sun-Jeong%20Kim/Desktop/CG/scene3.html
HALLYM UNIVERSITY
한림대학교