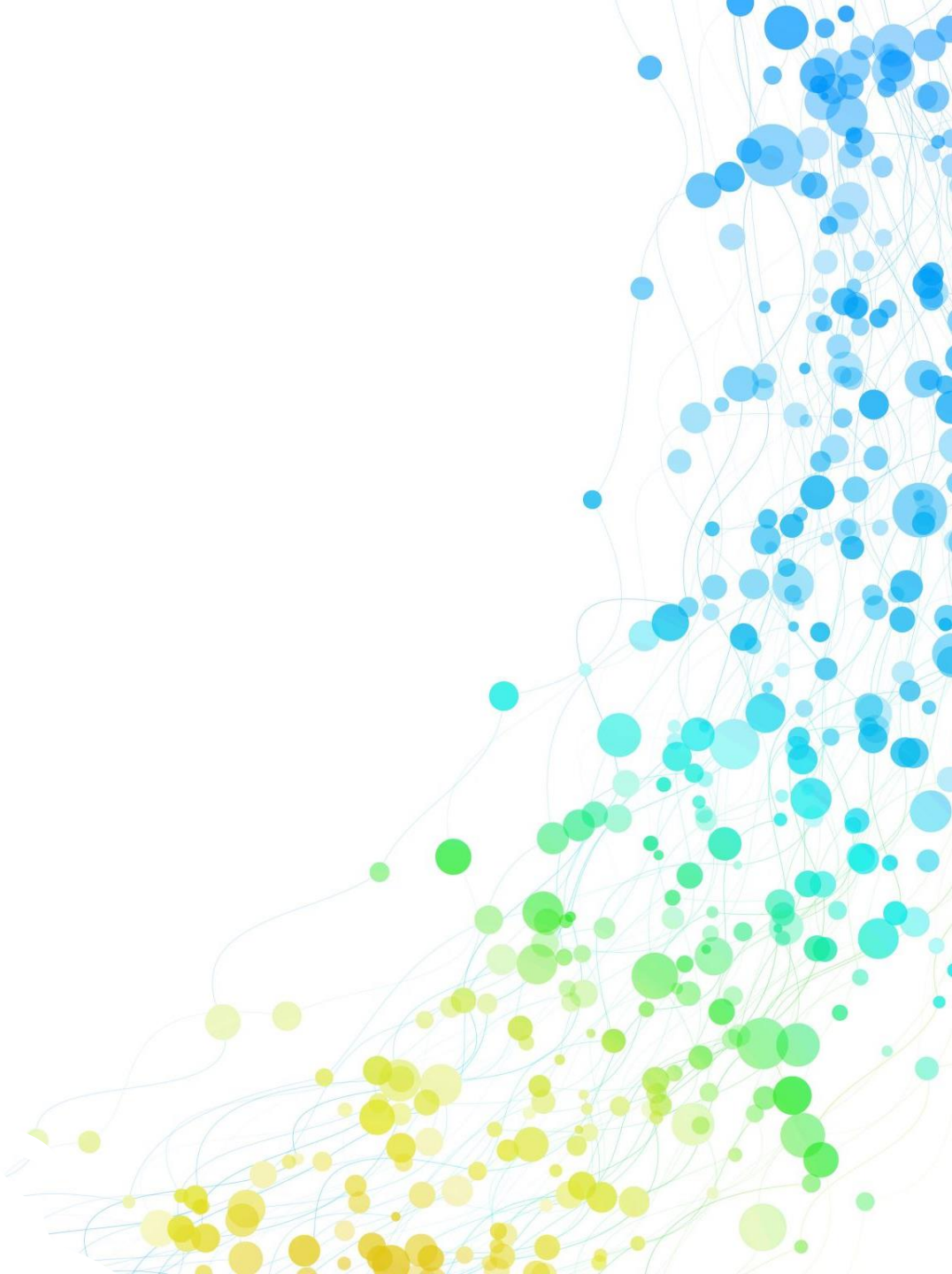
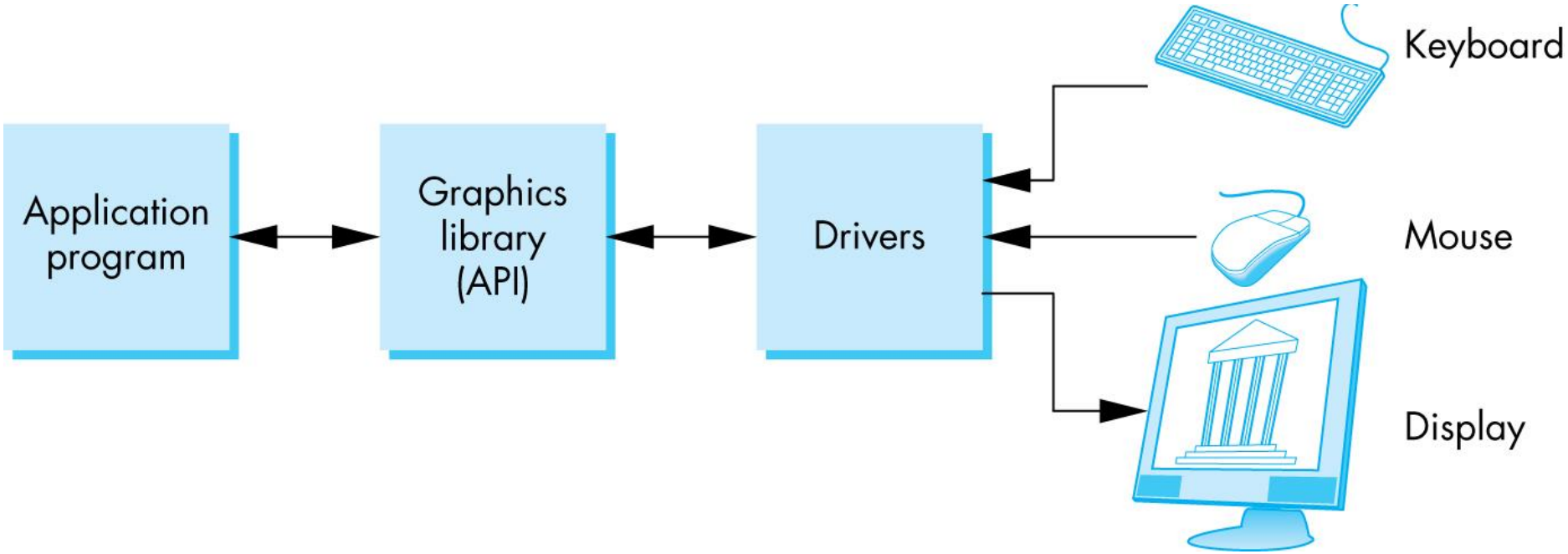
Graphics Programming

3RD WEEK, 2021



The Programmer's Interface

- Programmer sees the graphics system through a software interface
  - Application Programmer Interface (API)



API Contents

- Functions that specify what we need to form an image
  - Objects
  - Viewer
  - Light Source(s)
  - Materials

- Other information
  - Input from devices such as mouse and keyboard
  - Capabilities of system

Object Specification

- Most APIs support a limited set of primitives including
  - Points
  - Line segments
  - Polygons
  - Some curves and surfaces
    - Quadrics
    - Parametric polynomials

- All are defined through locations in space or vertices

Example (GPU based)

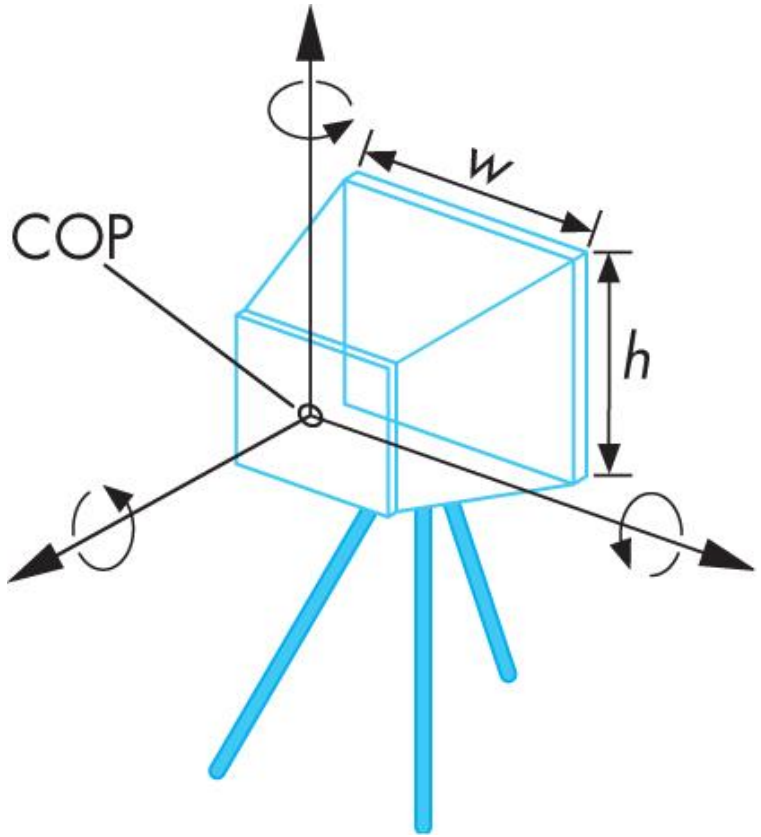
- Put geometric data in an array

```
vec3 points[3];
points[0] = vec3(0.0f, 0.0f, 0.0f);
points[1] = vec3(0.0f, 1.0f, 0.0f);
points[2] = vec3(0.0f, 0.0f, 1.0f);
```

- Send array to GPU
- Tell GPU to render as triangle

Camera Specification

- Six degrees of freedom
  - Position of center of lens
  - Orientation
- Lens
- Film size
- Orientation of film plane



Lights and Materials

- Types of lights
  - Point vs. directional light sources
  - Spotlights
  - Near and far sources
  - Color properties

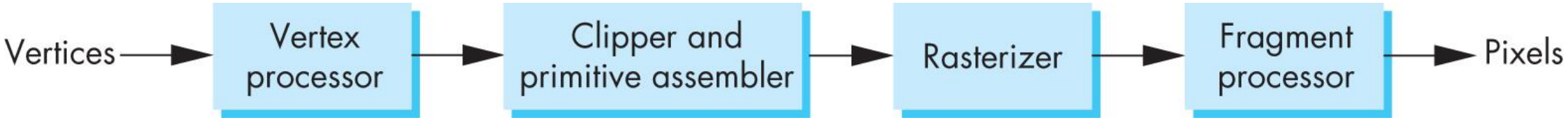
- Material properties
  - Absorption: color properties
  - Scattering: diffuse and specular components

OpenGL

- A platform-independent API that was
  - Easy to use
  - Close enough to the hardware to get excellent performance
  - Focus on rendering
  - Omitted windowing and input to avoid window system dependencies

Modern OpenGL

- Performance is achieved by using GPU rather than CPU
- Control GPU through programs called shaders
- Application's job is to send data to GPU
- GPU does all rendering



OpenGL 3.1

- Totally shader-based
  - No default shaders
  - Each application must provide both a vertex and a fragment shader
- No immediate mode
- Few state variables
- Most 2.5 functions deprecated
- Backward compatibility not required

Retained Mode Graphics

- Put all vertex and attribute data in array
- Send array to GPU to be rendered immediately
- Almost OK but problem is we would have to send array over each time we need another render of it
- Better to send array over and store on GPU for multiple renderings

Other Versions

- OpenGL ES
  - Embedded systems
  - Version 1.0 simplified OpenGL 2.1
  - Version 2.0 simplified OpenGL 3.1
    - Shader-based
- WebGL
  - Javascript implementation of ES 2.0
  - Supported on newer browsers
- OpenGL 4.1 and 4.2
  - Add geometry shaders and tessellator

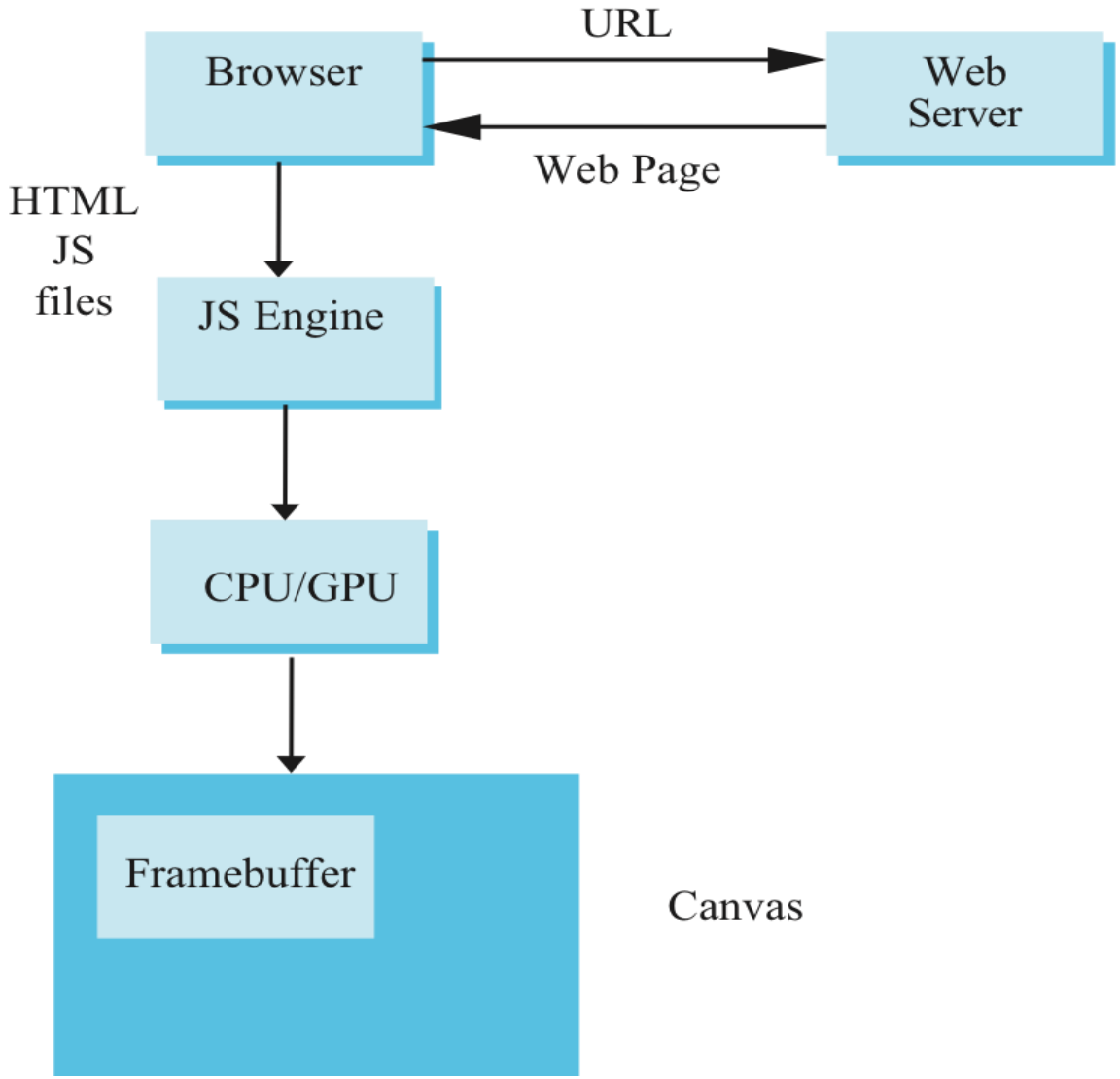
GLSL

- OpenGL Shading Language
- C-like with
  - Matrix and vector types (2, 3, 4 dimensional)
  - Overloaded operators
  - C++ like constructors
- Similar to Nvidia's Cg and Microsoft HLSL
- Code sent to shaders as source code
- New OpenGL functions to compile, link and get information to shaders

WebGL and GLSL

- WebGL requires shaders and is based less on a state machine model than a data flow model
- Most state variables, attributes and related pre 3.1 OpenGL functions have been deprecated
- Action happens in shaders
- Job of application is to get data to GPU

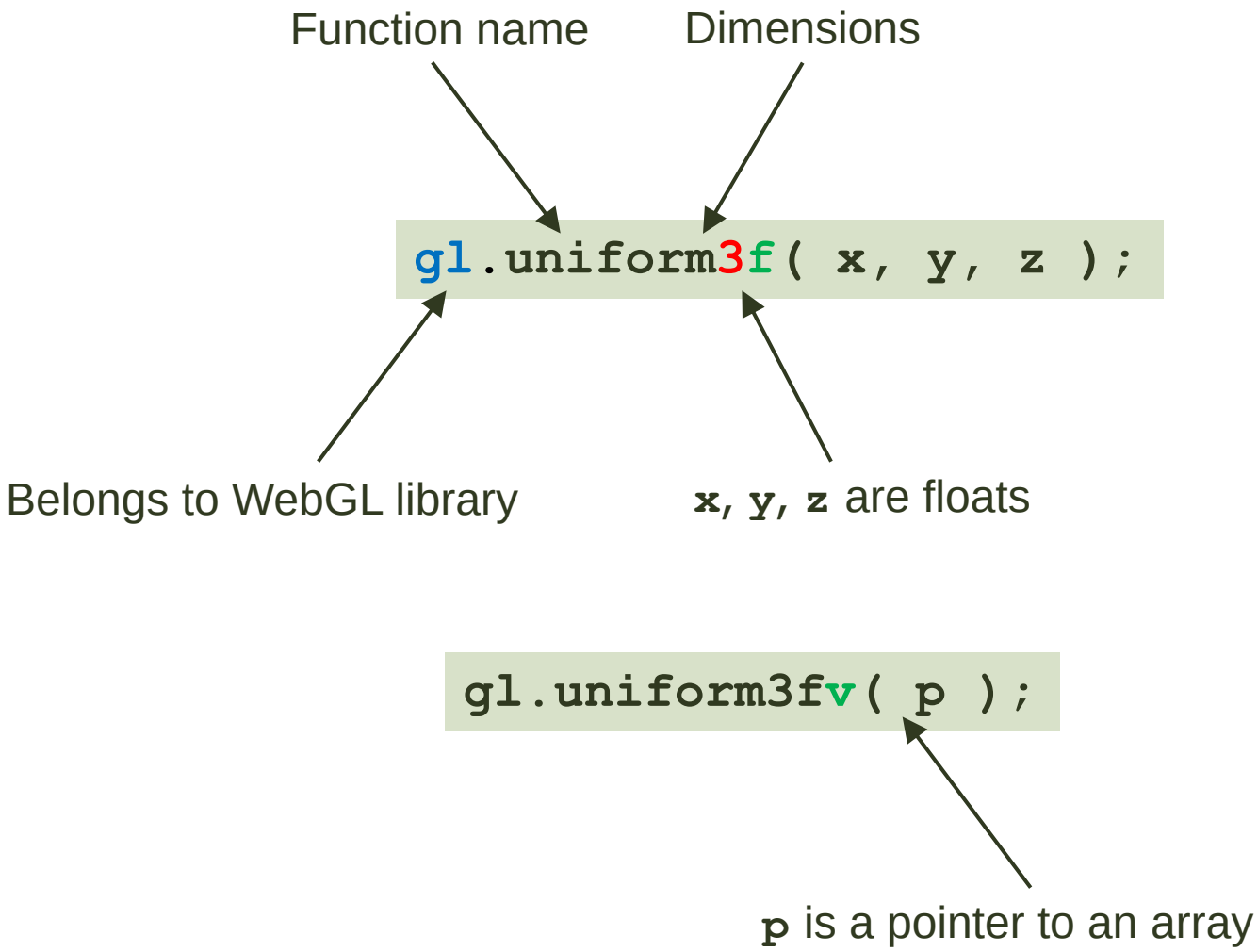
Execution in Browser



Event Loop

- Remember that the sample program specifies a render function which is an event listener or callback function
  - Every program should have a render callback
  - For a static application we need only execute the render function once
  - In a dynamic application, the render function can call itself recursively but each redrawing of the display must be triggered by an event

WebGL Function Format

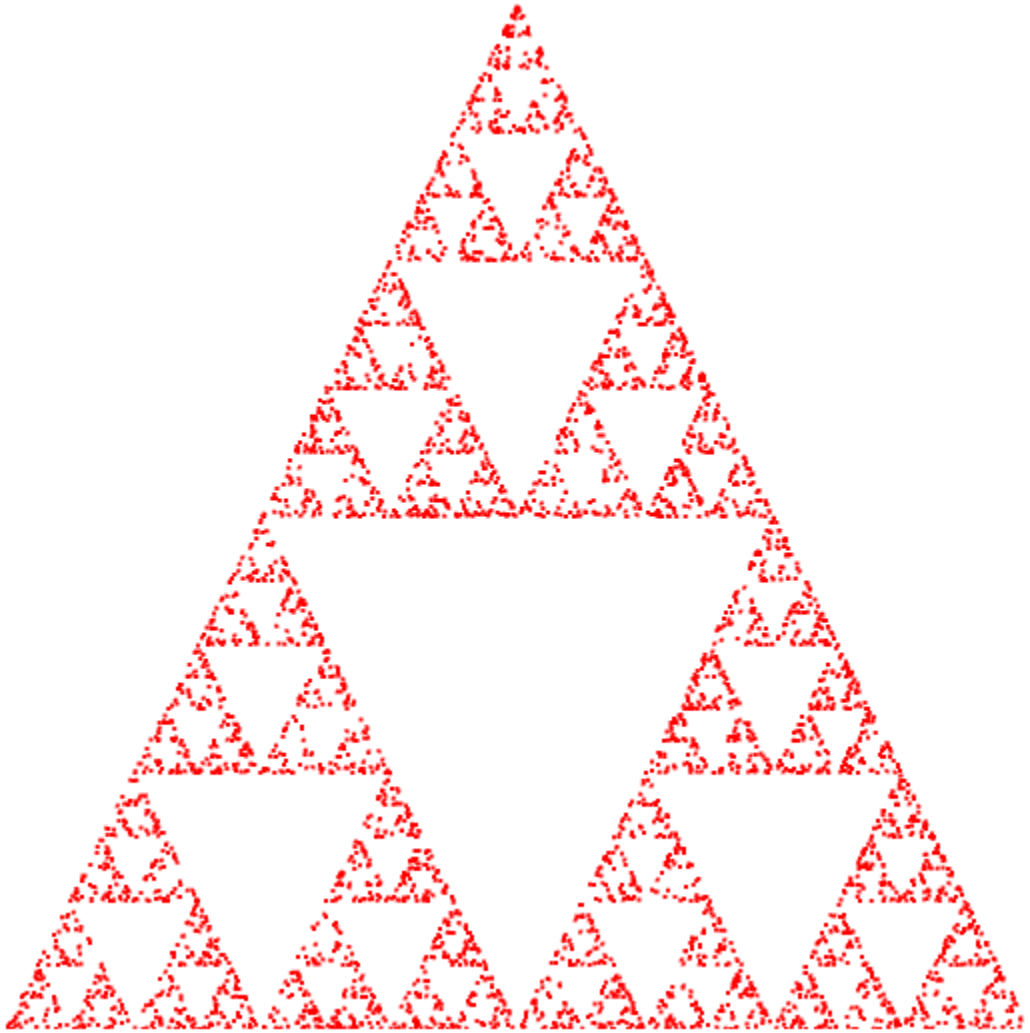


WebGL Constants

- Most constants are defined in the canvas object
  - In desktop OpenGL, they were in #include files such as **`gl.h`**
- Examples
  - desktop OpenGL
    - **`glEnable(GL_DEPTH_TEST);`**
  - WebGL
    - **`gl.enable(gl.DEPTH_TEST);`**
  - **`gl.clear(gl.COLOR_BUFFER_BIT);`**

The Sierpinski Gasket (1)

- What is?
  - Interesting shape in area such as fractal geometry
  - Object that can be defined recursively and randomly

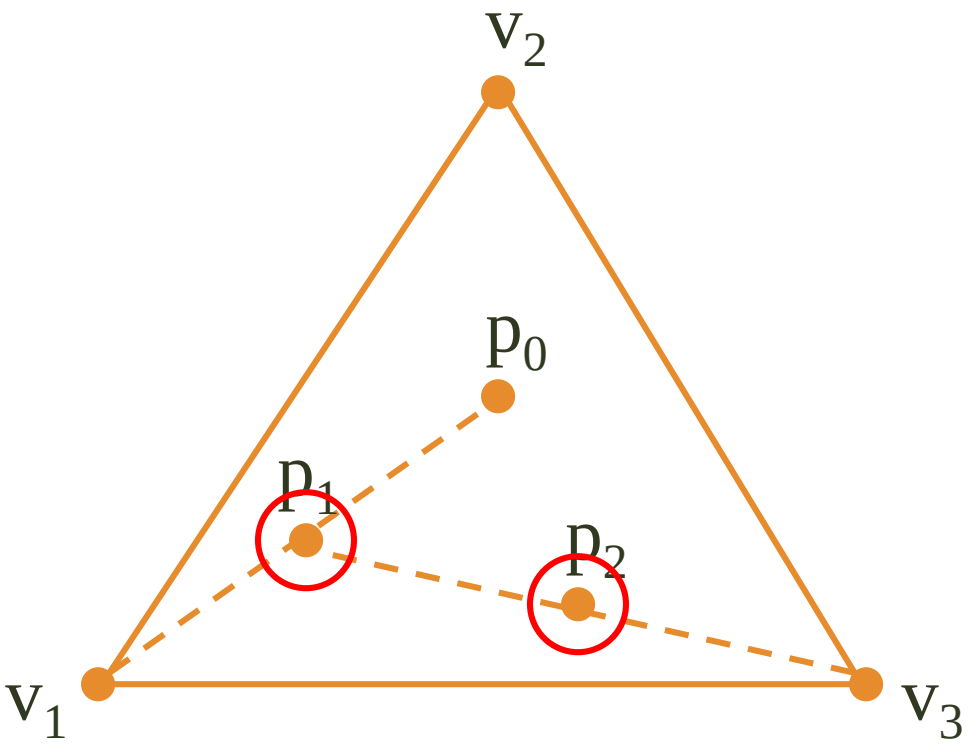


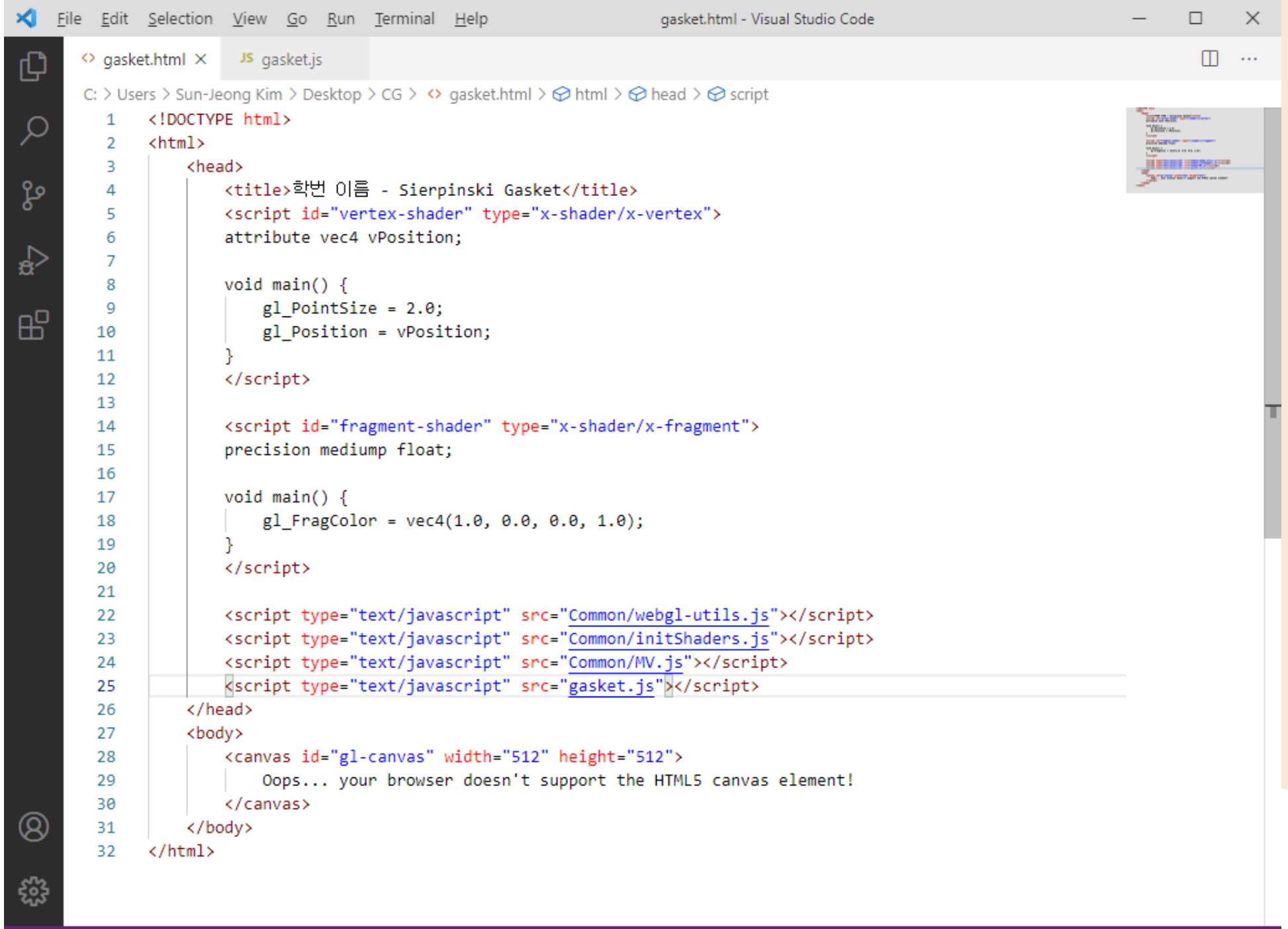
The Sierpinski Gasket (2)

- How to…
  - Start with three vertices in the plane
  1. Pick an initial point at random inside the triangle
  2. Select one of the three vertices at random
  3. Find the point halfway between the initial point and the randomly selected vertex
  4. Display this new point by putting some sort of marker, such as a small circle, at its location
  5. Replace the initial point with this new point
  6. Return to step 2

The Sierpinski Gasket (2)

- How to…




File Edit Selection View Go Run Terminal Help
gasket.html - Visual Studio Code
gasket.html
gasket.js
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket.html > html > head > script
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>학번 이름 - Sierpinski Gasket</title>
5 <script id="vertex-shader" type="x-shader/x-vertex">
6 attribute vec4 vPosition;
7
8 void main() {
9 gl_PointSize = 2.0;
10 gl_Position = vPosition;
11 }
12 </script>
13
14 <script id="fragment-shader" type="x-shader/x-fragment">
15 precision mediump float;
16
17 void main() {
18 gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
19 }
20 </script>
21
22 <script type="text/javascript" src="Common/webgl-utils.js"></script>
23 <script type="text/javascript" src="Common/initShaders.js"></script>
24 <script type="text/javascript" src="Common/MV.js"></script>
25 <script type="text/javascript" src="gasket.js"></script>
26 </head>
27 <body>
28 <canvas id="gl-canvas" width="512" height="512">
29 Oops... your browser doesn't support the HTML5 canvas element!
30 </canvas>
31 </body>
32 </html>
0 0
Ln 25, Col 55 Spaces: 4 UTF-8 CRLF HTML

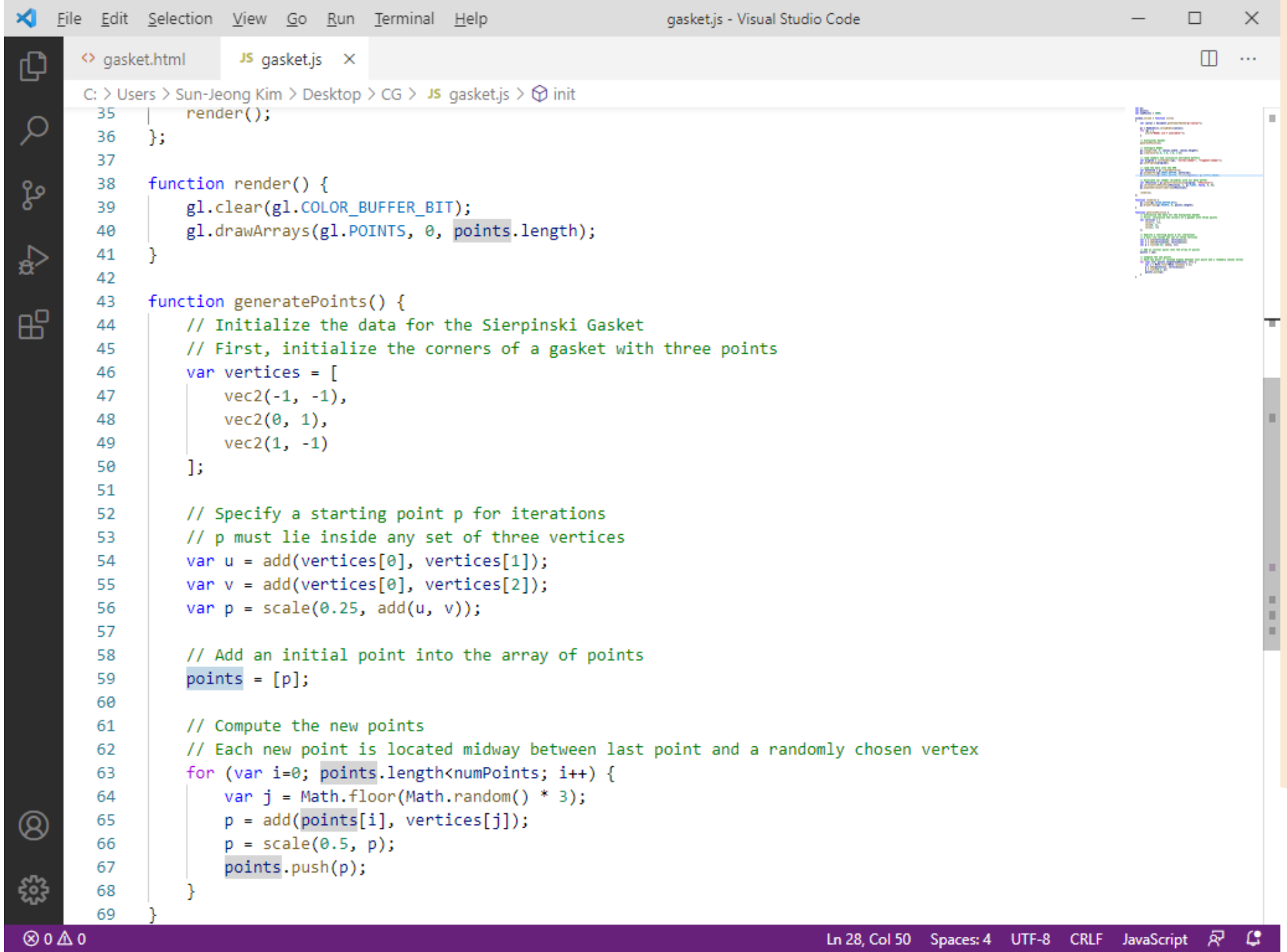
File Edit Selection View Go Run Terminal Help gasket.js - Visual Studio Code

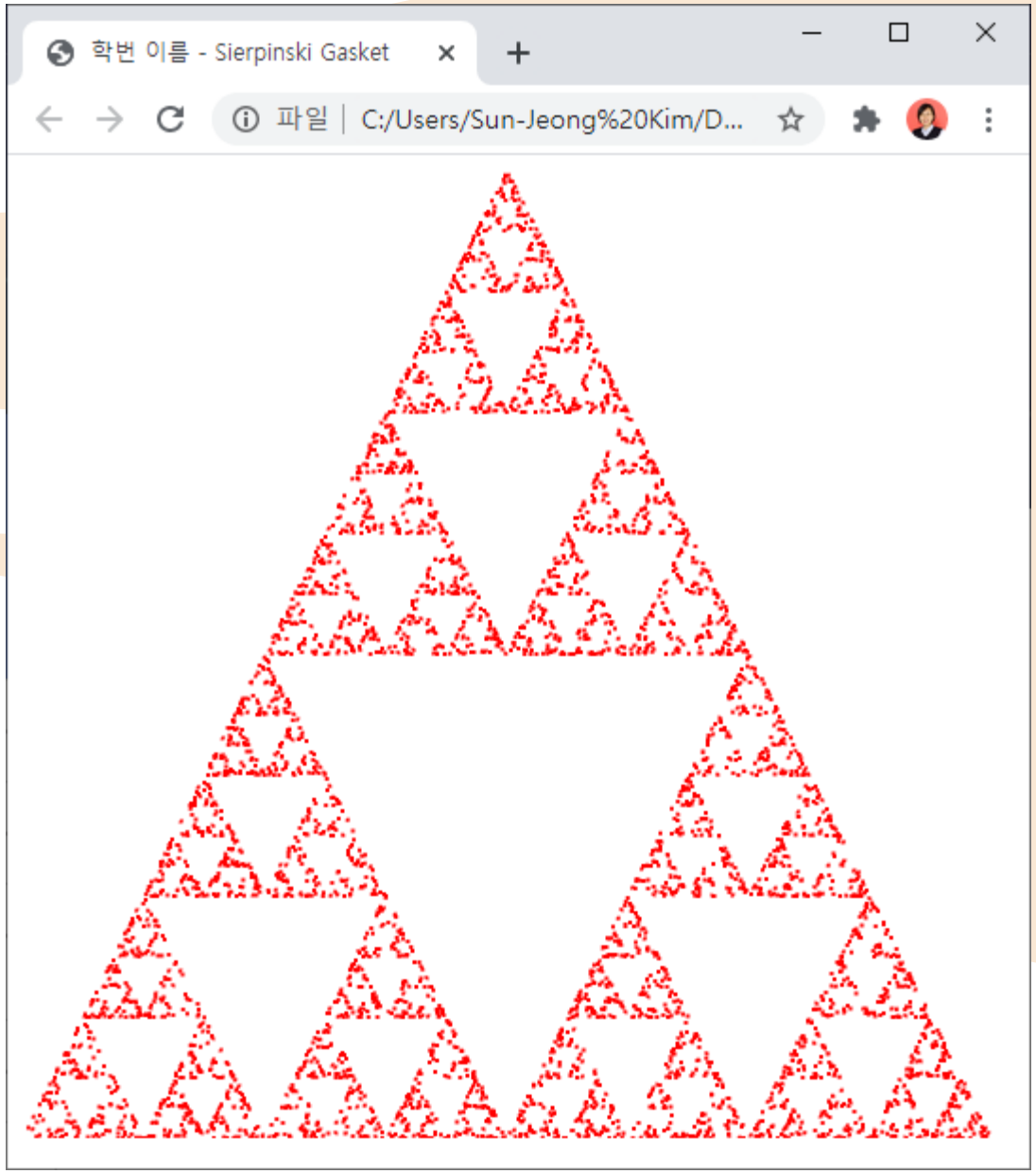
gasket.html gasket.js

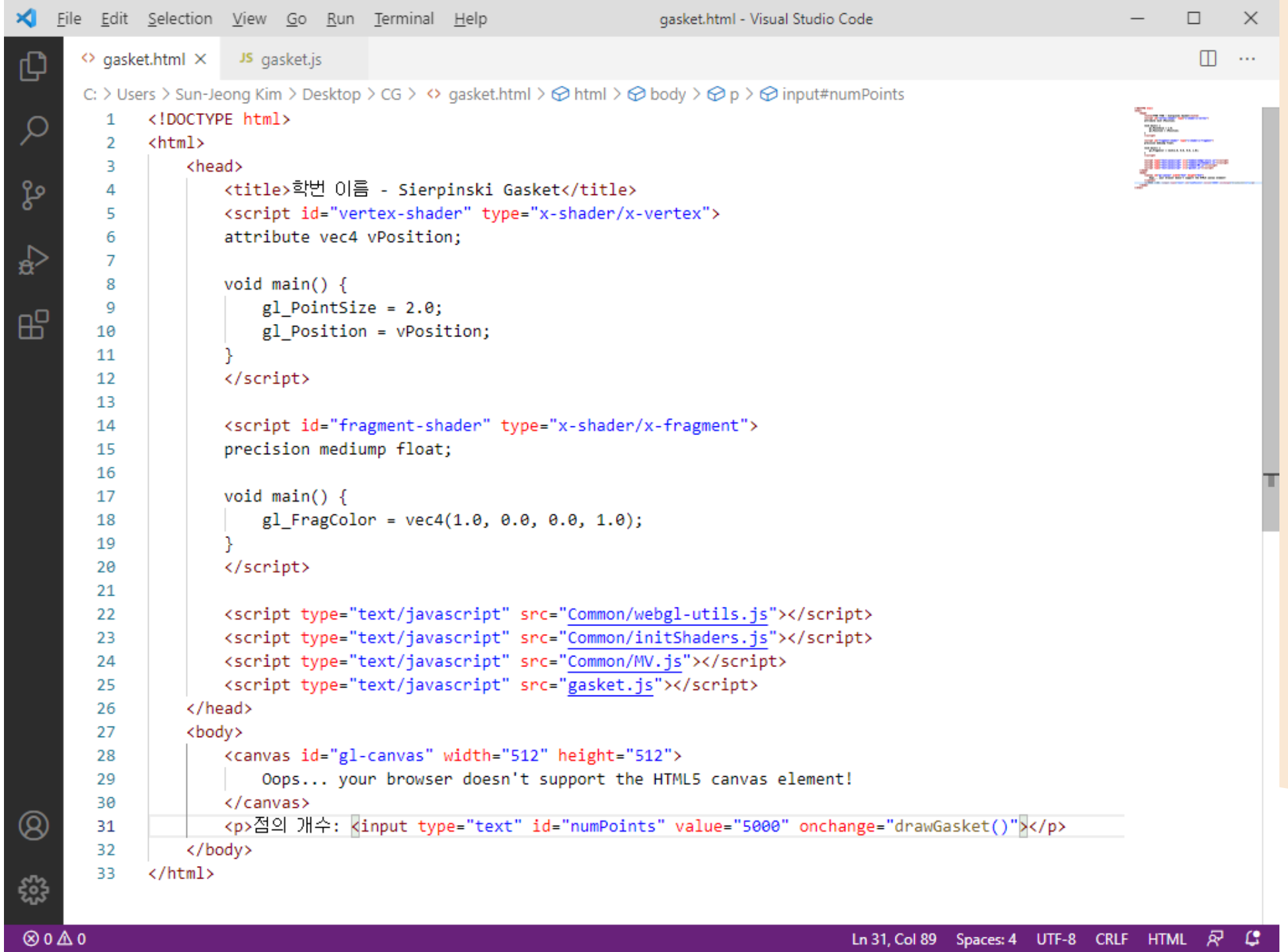
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket.js > init

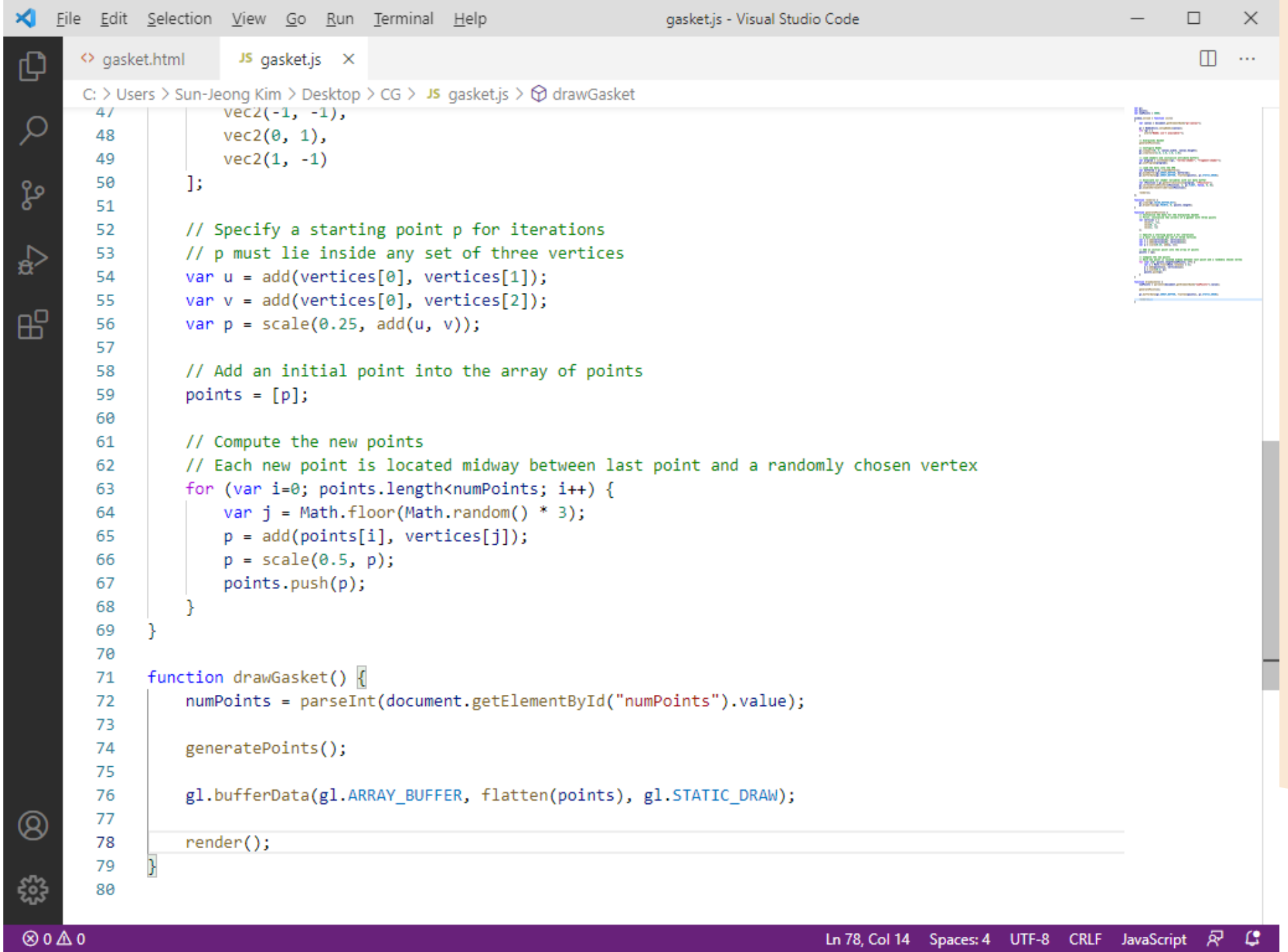
```
1   var gl;
2   var points;
3   var numPoints = 5000;
4
5   window.onload = function init()
6   {
7       var canvas = document.getElementById("gl-canvas");
8
9       gl = WebGLUtils.setupWebGL(canvas);
10      if( !gl ) {
11          alert("WebGL isn't available!");
12      }
13
14      // Sierpinski Gasket
15      generatePoints();
16
17      // Configure WebGL
18      gl.viewport(0, 0, canvas.width, canvas.height);
19      gl.clearColor(1.0, 1.0, 1.0, 1.0);
20
21      // Load shaders and initialize attribute buffers
22      var program = initShaders(gl, "vertex-shader", "fragment-shader");
23      gl.useProgram(program);
24
25      // Load the data into the GPU
26      var bufferId = gl.createBuffer();
27      gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
28      gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
29
30      // Associate our shader variables with our data buffer
31      var vPosition = gl.getAttribLocation(program, "vPosition");
32      gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
33      gl.enableVertexAttribArray(vPosition);
34
35      render();
```

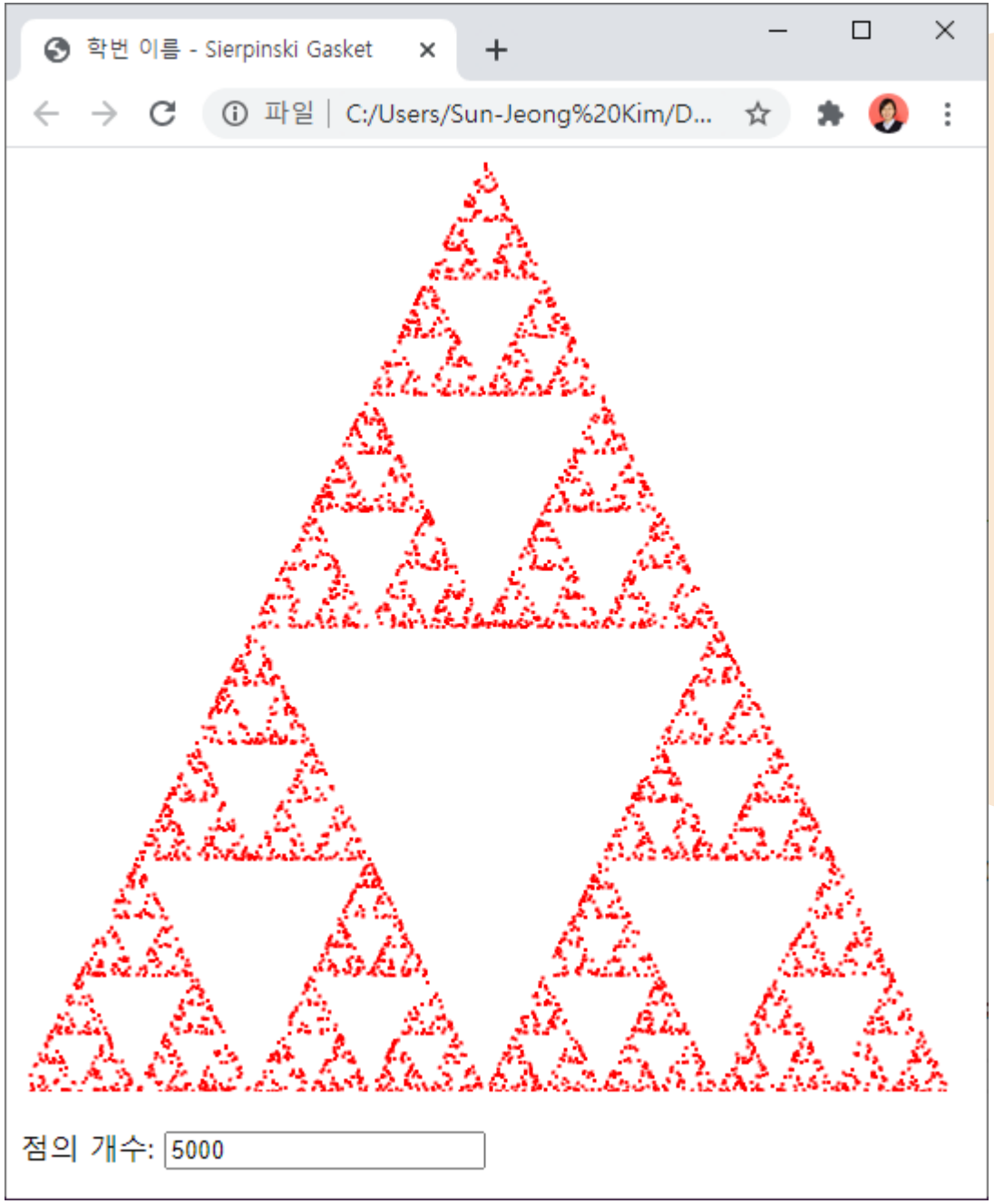
⊗ 0 ⚠ 0 Ln 28, Col 50 Spaces: 4 UTF-8 CRLF JavaScript

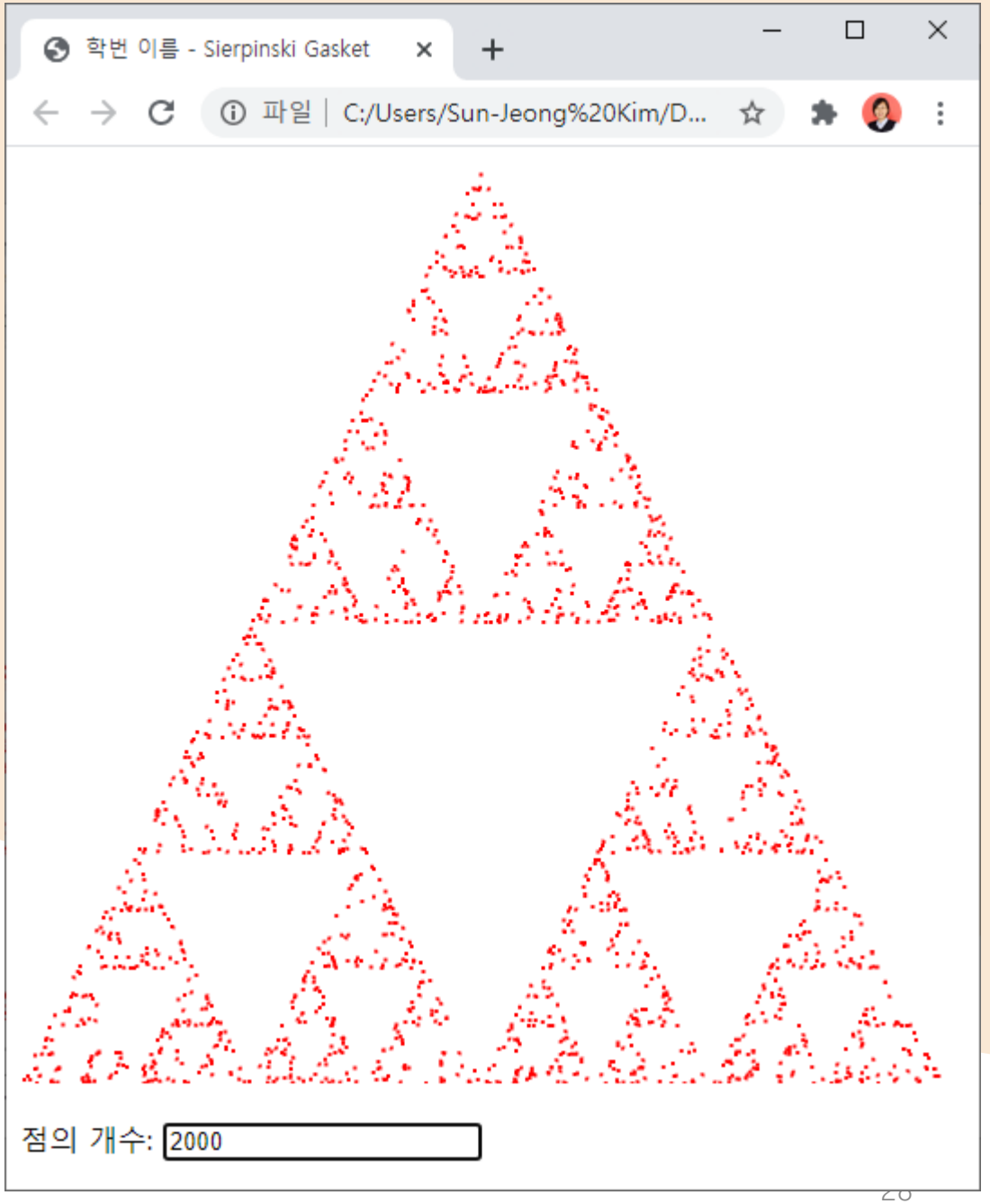

File Edit Selection View Go Run Terminal Help
gasket.js - Visual Studio Code
gasket.html
gasket.js
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket.js > init
render();
};
function render() {
gl.clear(gl.COLOR_BUFFER_BIT);
gl.drawArrays(gl.POINTS, 0, points.length);
}
function generatePoints() {
// Initialize the data for the Sierpinski Gasket
// First, initialize the corners of a gasket with three points
var vertices = [
vec2(-1, -1),
vec2(0, 1),
vec2(1, -1)
];
// Specify a starting point p for iterations
// p must lie inside any set of three vertices
var u = add(vertices[0], vertices[1]);
var v = add(vertices[0], vertices[2]);
var p = scale(0.25, add(u, v));
// Add an initial point into the array of points
points = [p];
// Compute the new points
// Each new point is located midway between last point and a randomly chosen vertex
for (var i=0; points.length<numPoints; i++) {
var j = Math.floor(Math.random() * 3);
p = add(points[i], vertices[j]);
p = scale(0.5, p);
points.push(p);
}
}
0 0
Ln 28, Col 50
Spaces: 4
UTF-8
CRLF
JavaScript


학번 이름 - Sierpinski Gasket
파일 | C:/Users/Sun-Jeong%20Kim/D...


File Edit Selection View Go Run Terminal Help
gasket.html - Visual Studio Code
gasket.html
gasket.js
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket.html > html > body > p > input#numPoints
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>학번 이름 - Sierpinski Gasket</title>
5 <script id="vertex-shader" type="x-shader/x-vertex">
6 attribute vec4 vPosition;
7
8 void main() {
9 gl_PointSize = 2.0;
10 gl_Position = vPosition;
11 }
12 </script>
13
14 <script id="fragment-shader" type="x-shader/x-fragment">
15 precision mediump float;
16
17 void main() {
18 gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
19 }
20 </script>
21
22 <script type="text/javascript" src="Common/webgl-utils.js"></script>
23 <script type="text/javascript" src="Common/initShaders.js"></script>
24 <script type="text/javascript" src="Common/MV.js"></script>
25 <script type="text/javascript" src="gasket.js"></script>
26 </head>
27 <body>
28 <canvas id="gl-canvas" width="512" height="512">
29 Oops... your browser doesn't support the HTML5 canvas element!
30 </canvas>
31 점의 개수: <input type="text" id="numPoints" value="5000" onchange="drawGasket()">
32 </body>
33 </html>
0 0
Ln 31, Col 89 Spaces: 4 UTF-8 CRLF HTML

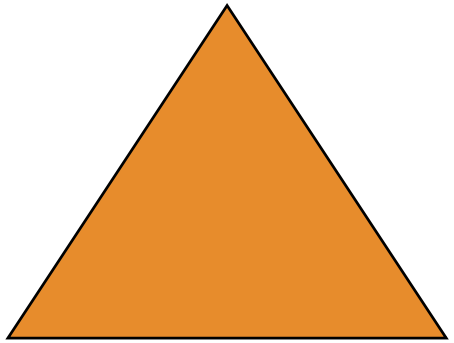

File Edit Selection View Go Run Terminal Help
gasket.js - Visual Studio Code
gasket.html
gasket.js
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket.js > drawGasket
47 vec2(-1, -1),
48 vec2(0, 1),
49 vec2(1, -1)
50];
51
52 // Specify a starting point p for iterations
53 // p must lie inside any set of three vertices
54 var u = add(vertices[0], vertices[1]);
55 var v = add(vertices[0], vertices[2]);
56 var p = scale(0.25, add(u, v));
57
58 // Add an initial point into the array of points
59 points = [p];
60
61 // Compute the new points
62 // Each new point is located midway between last point and a randomly chosen vertex
63 for (var i=0; points.length<numPoints; i++) {
64 var j = Math.floor(Math.random() * 3);
65 p = add(points[i], vertices[j]);
66 p = scale(0.5, p);
67 points.push(p);
68 }
69 }
70
71 function drawGasket() {
72 numPoints = parseInt(document.getElementById("numPoints").value);
73
74 generatePoints();
75
76 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
77
78 render();
79 }
80
0 0
Ln 78, Col 14 Spaces: 4 UTF-8 CRLF JavaScript


학번 이름 - Sierpinski Gasket
파일 | C:/Users/Sun-Jeong%20Kim/D...
점의 개수: 5000

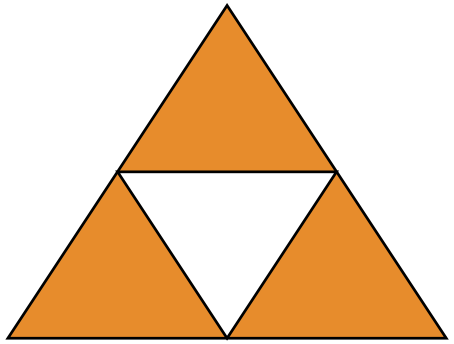

학번 이름 - Sierpinski Gasket
파일 | C:/Users/Sun-Jeong%20Kim/D...
점의 개수: 2000

Polygons and Recursion

- Start with a triangle



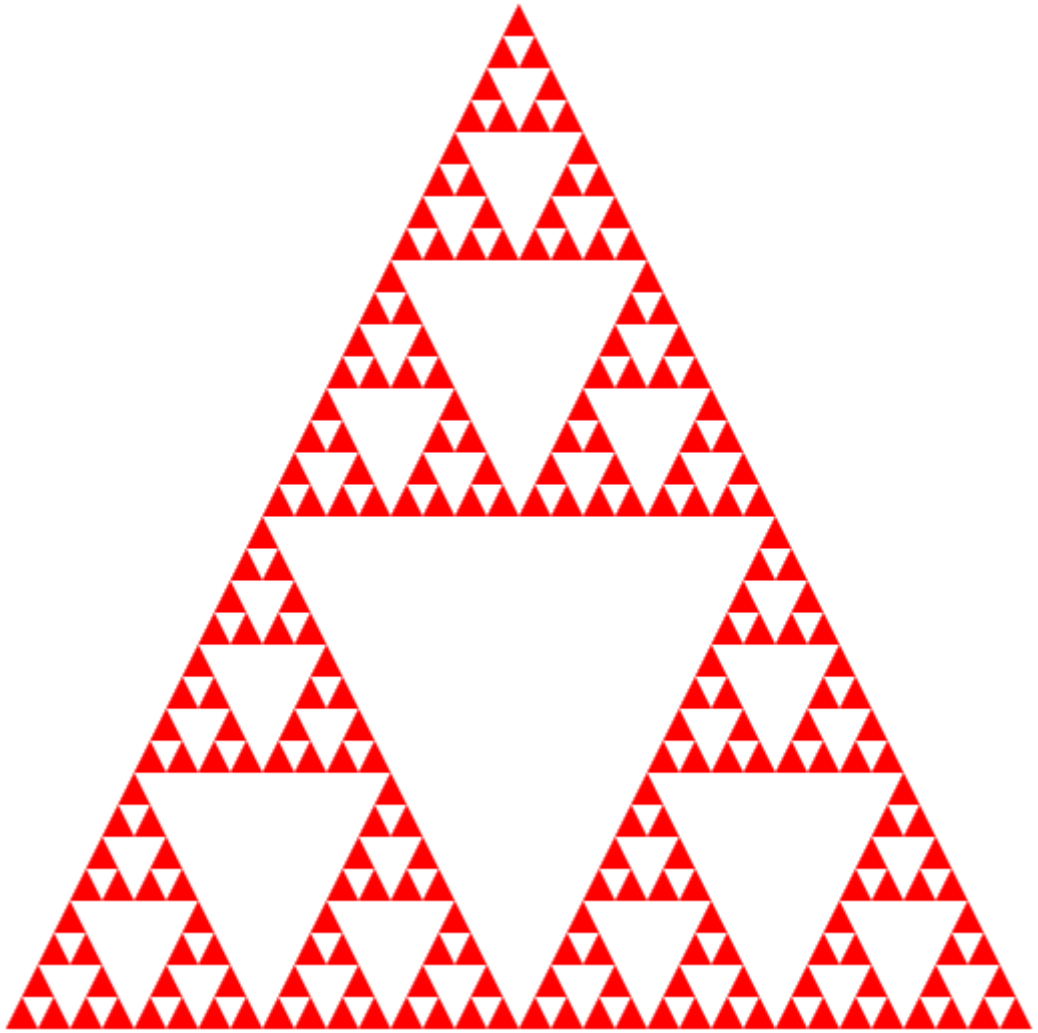
- Connect bisectors of sides and remove central triangle



- Repeat

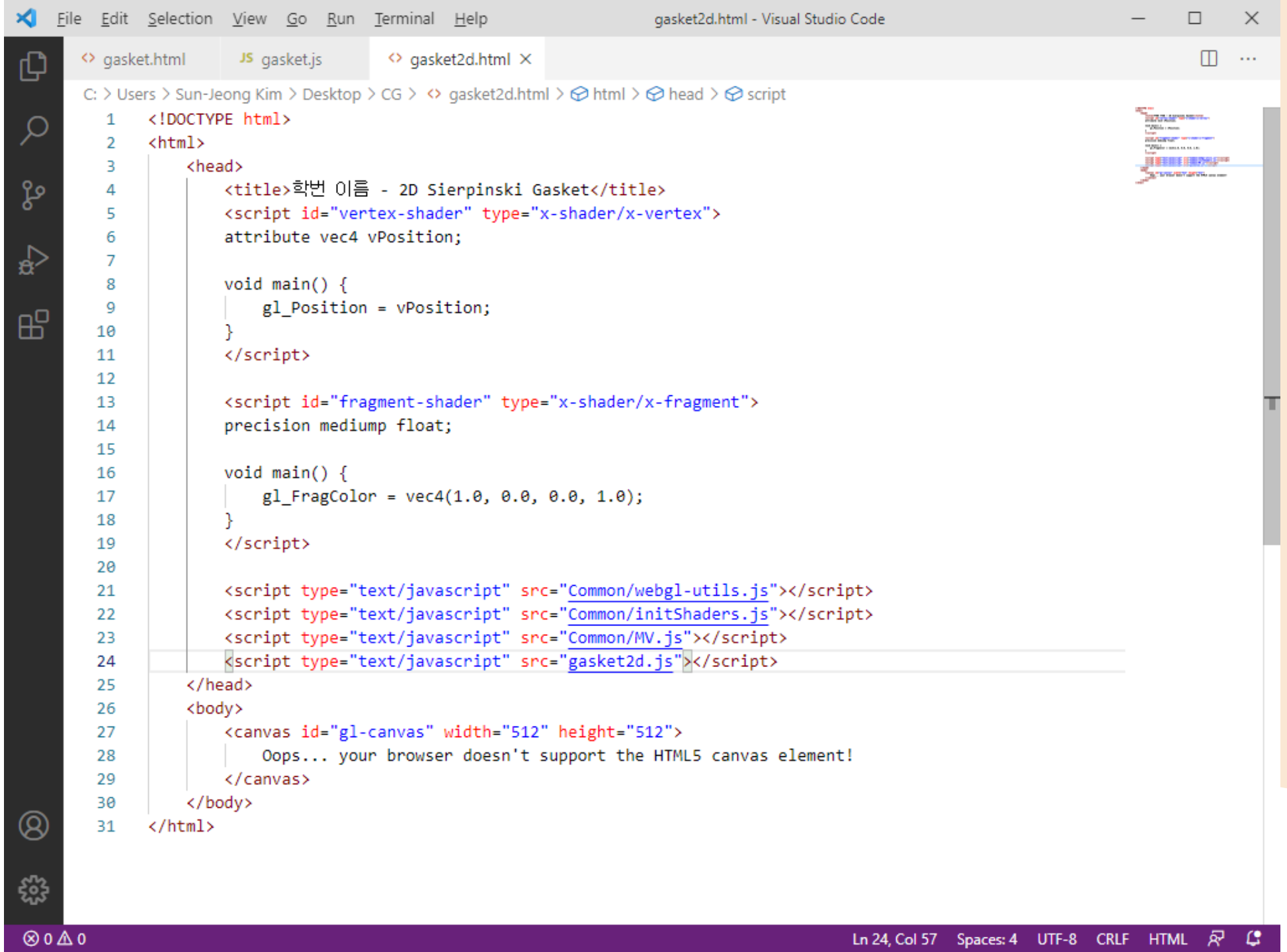
Example

- Five subdivisions



Gasket Program

- HTML file
  - Same as in other examples
  - Pass through vertex shader
  - Fragment shader sets color
  - Read in JS file
- JS file
  - Initialize WebGL
  - Render triangles
  - Draw one triangle
  - Subdivide a triangle


File Edit Selection View Go Run Terminal Help
gasket2d.html - Visual Studio Code
gasket.html
gasket.js
gasket2d.html
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket2d.html > html > head > script
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>학번 이름 - 2D Sierpinski Gasket</title>
5 <script id="vertex-shader" type="x-shader/x-vertex">
6 attribute vec4 vPosition;
7
8 void main() {
9 gl_Position = vPosition;
10 }
11 </script>
12
13 <script id="fragment-shader" type="x-shader/x-fragment">
14 precision mediump float;
15
16 void main() {
17 gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
18 }
19 </script>
20
21 <script type="text/javascript" src="Common/webgl-utils.js"></script>
22 <script type="text/javascript" src="Common/initShaders.js"></script>
23 <script type="text/javascript" src="Common/MV.js"></script>
24 <script type="text/javascript" src="gasket2d.js"></script>
25 </head>
26 <body>
27 <canvas id="gl-canvas" width="512" height="512">
28 Oops... your browser doesn't support the HTML5 canvas element!
29 </canvas>
30 </body>
31 </html>
0 0
Ln 24, Col 57 Spaces: 4 UTF-8 CRLF HTML

File Edit Selection View Go Run Terminal Help gasket2d.js - Visual Studio Code

gasket.html | gasket.js | gasket2d.html | gasket2d.js

C: > Users > Sun-Jeong Kim > Desktop > CG > gasket2d.js > divideTriangle

```
var gl;
var points;
var numTimes = 5;

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

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

    // 2D Sierpinski Gasket
    generateTriangles();

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

    // Load shaders and initialize attribute buffers
    var program = initShaders(gl, "vertex-shader", "fragment-shader");
    gl.useProgram(program);

    // 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(program, "vPosition");
    gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
    gl.enableVertexAttribArray(vPosition);

    render();
```

⊗ 0 ⚠ 0 Ln 73, Col 42 Spaces: 4 UTF-8 CRLF JavaScript

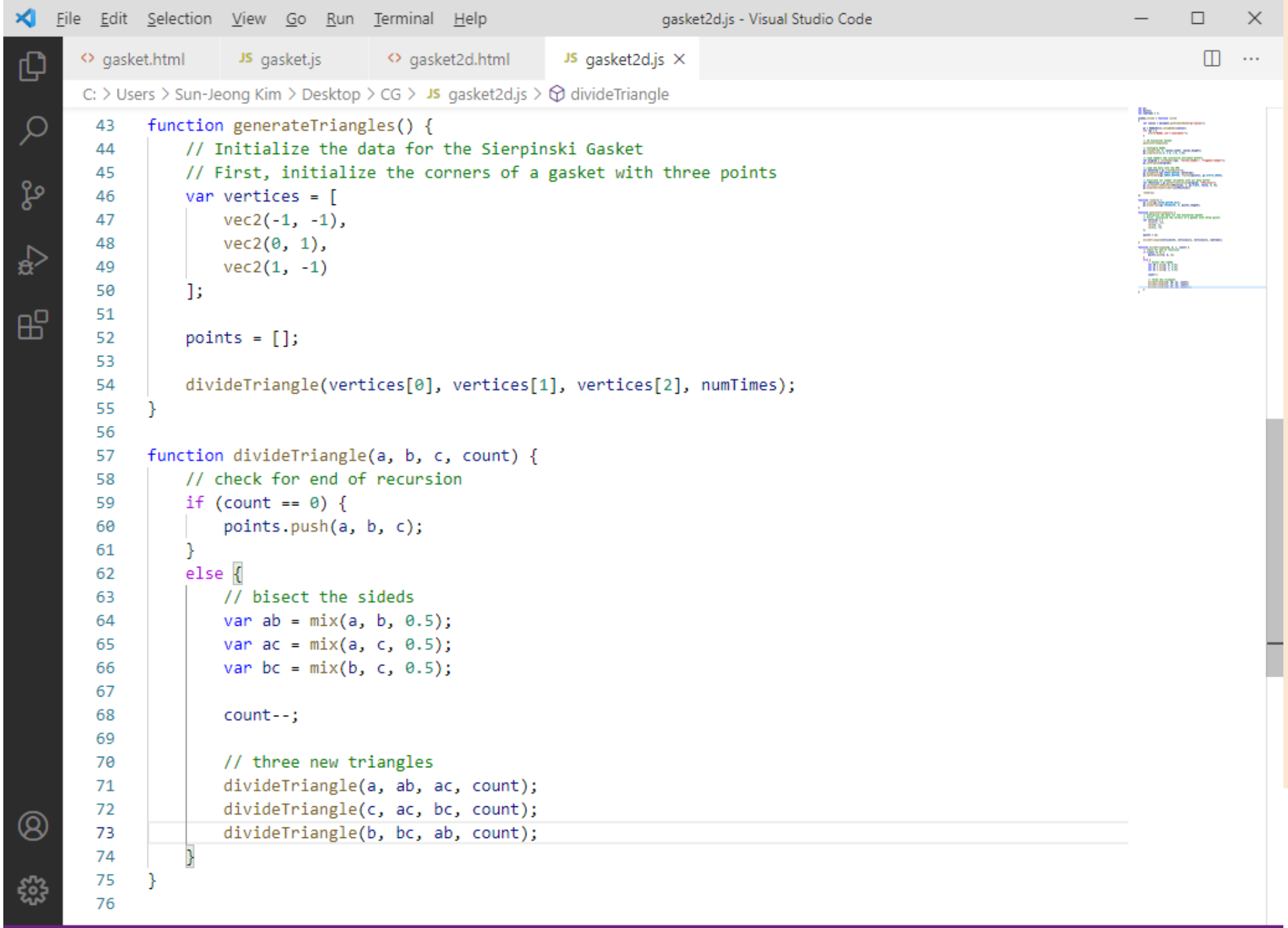
File Edit Selection View Go Run Terminal Help gasket2d.js - Visual Studio Code

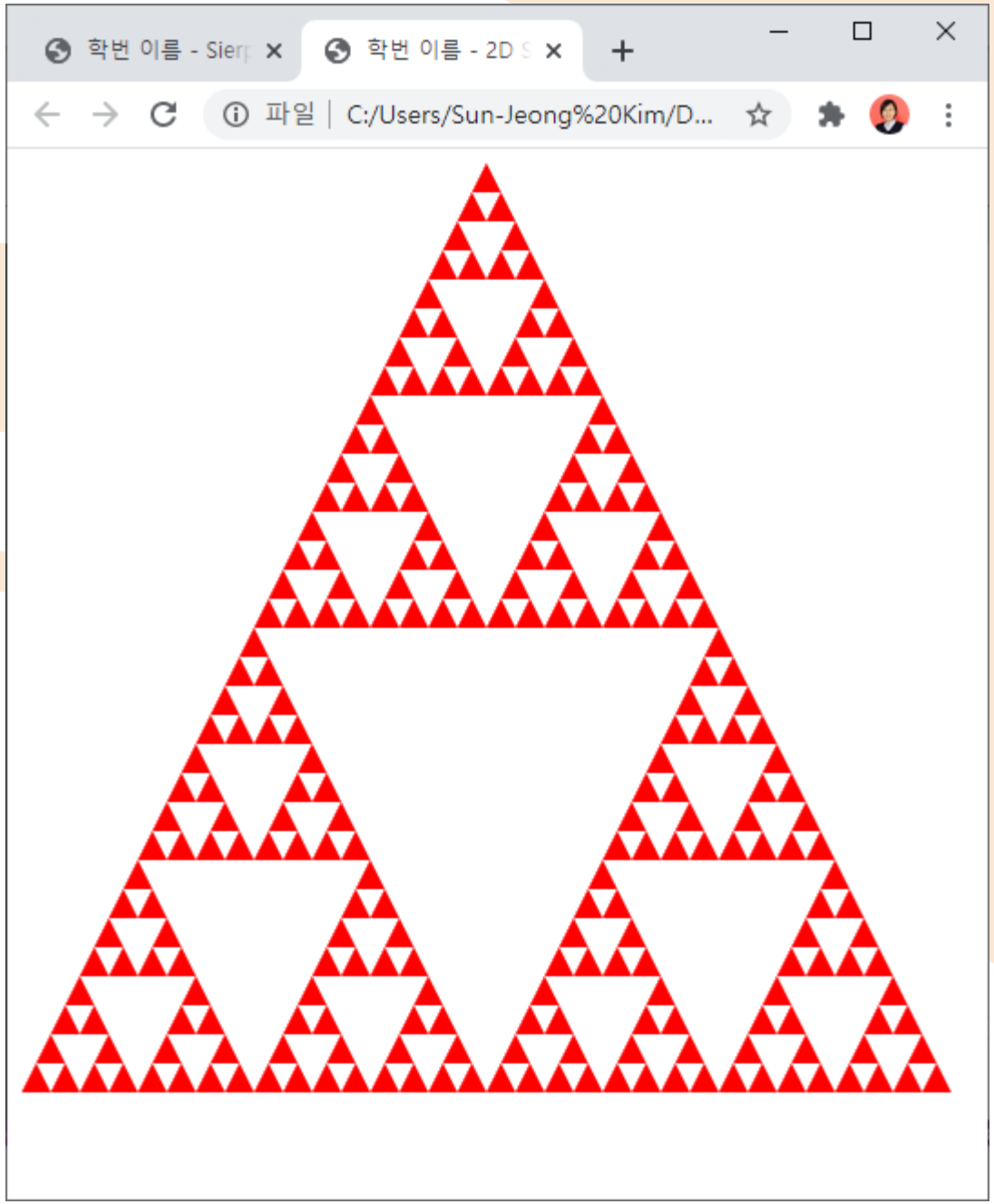
gasket.html | gasket.js | gasket2d.html | gasket2d.js

C: > Users > Sun-Jeong Kim > Desktop > CG > gasket2d.js > divideTriangle

```
35      render();
36  };
37
38  function render() {
39      gl.clear(gl.COLOR_BUFFER_BIT);
40      gl.drawArrays(gl.TRIANGLES, 0, points.length);
41  }
42
43  function generateTriangles() {
44      // Initialize the data for the Sierpinski Gasket
45      // First, initialize the corners of a gasket with three points
46      var vertices = [
47          vec2(-1, -1),
48          vec2(0, 1),
49          vec2(1, -1)
50      ];
51
52      points = [];
53
54      divideTriangle(vertices[0], vertices[1], vertices[2], numTimes);
55  }
56
57  function divideTriangle(a, b, c, count) {
58      // check for end of recursion
59      if (count == 0) {
60          points.push(a, b, c);
61      }
62      else {
63          // bisect the sideds
64          var ab = mix(a, b, 0.5);
65          var ac = mix(a, c, 0.5);
66          var bc = mix(b, c, 0.5);
67
68          count--;
69
```

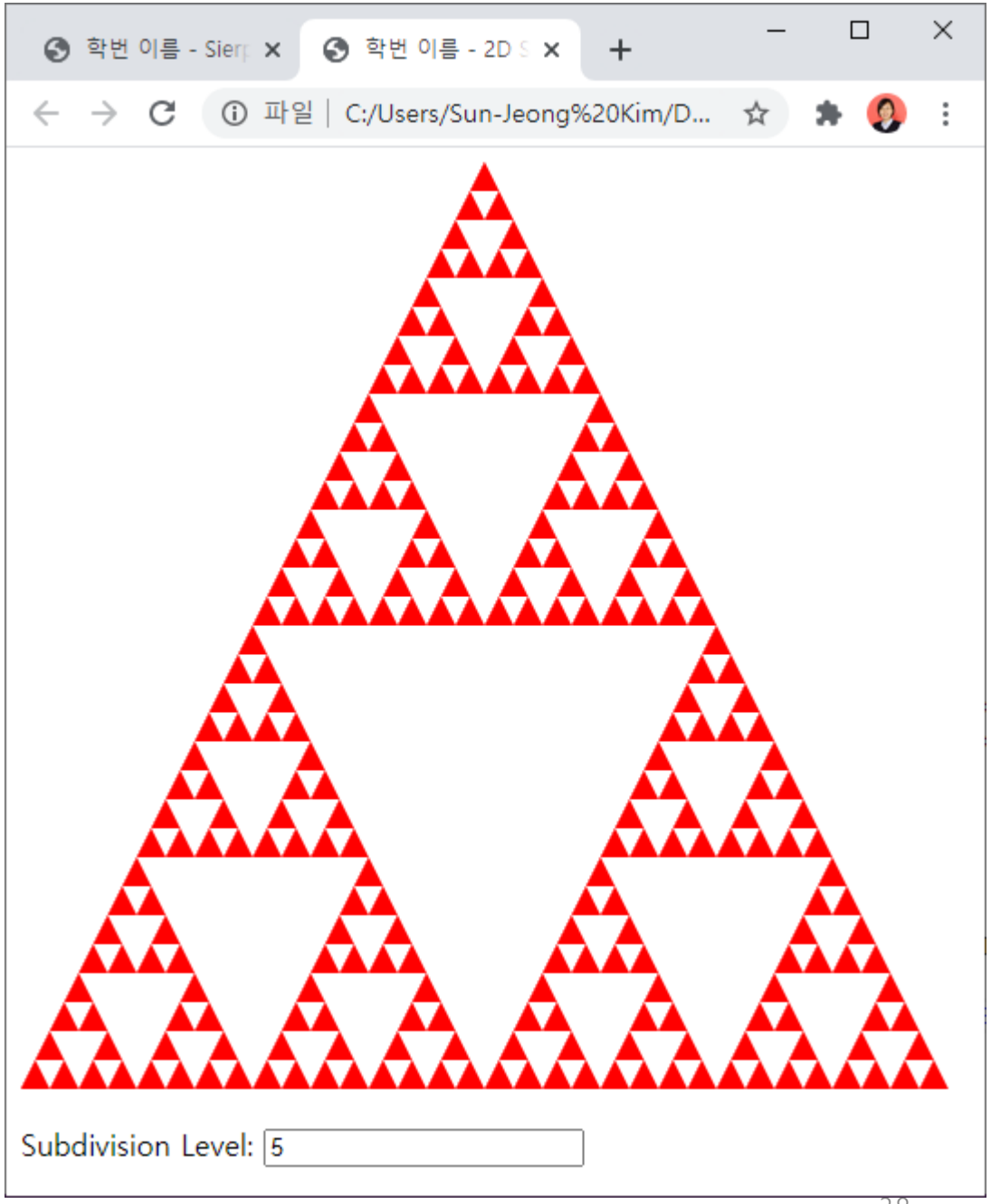
⊗ 0 ⚠ 0 Ln 73, Col 42 Spaces: 4 UTF-8 CRLF JavaScript

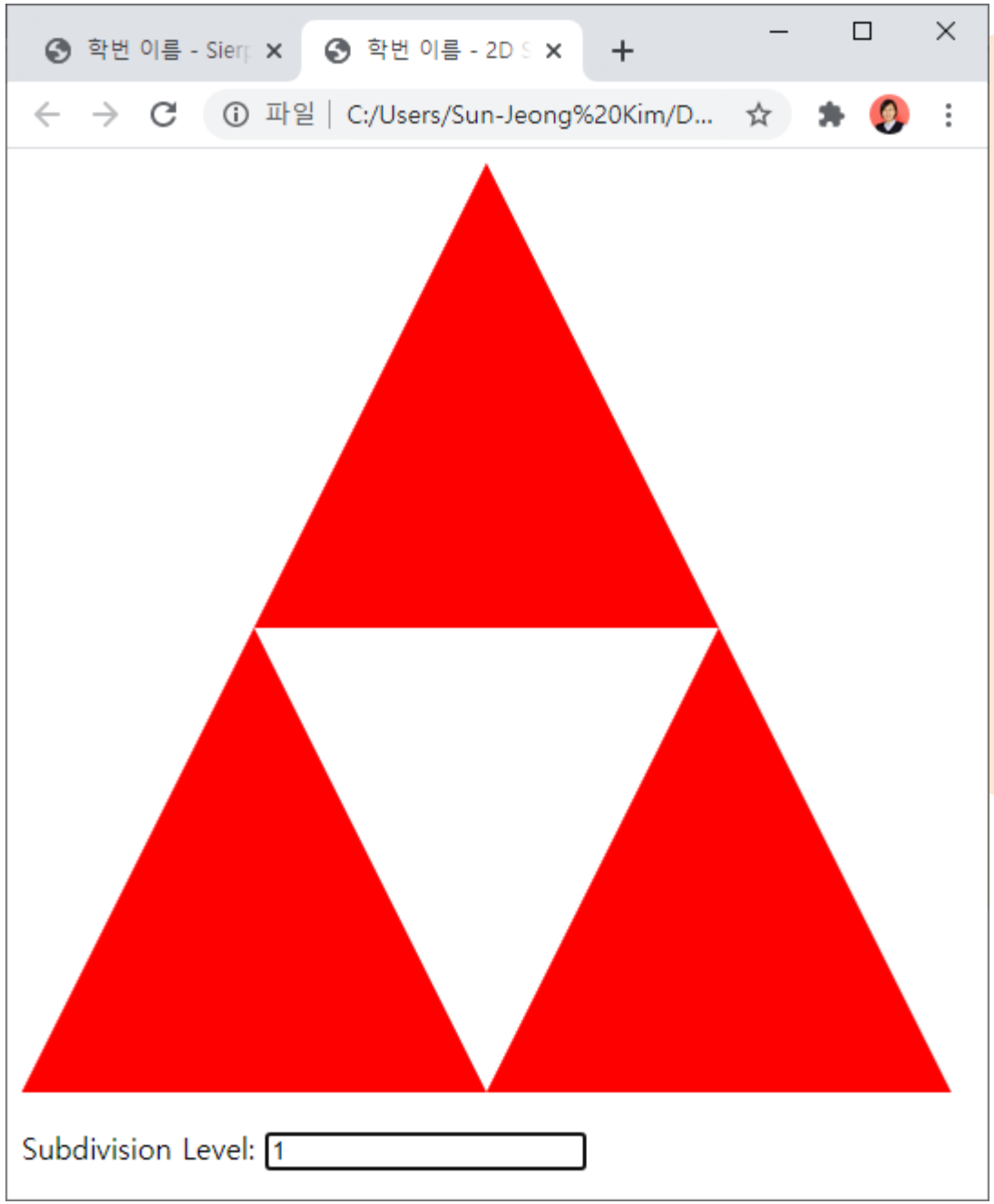

File Edit Selection View Go Run Terminal Help
gasket2d.js - Visual Studio Code
gasket.html
gasket.js
gasket2d.html
gasket2d.js
C: > Users > Sun-Jeong Kim > Desktop > CG > gasket2d.js > divideTriangle
function generateTriangles() {
// Initialize the data for the Sierpinski Gasket
// First, initialize the corners of a gasket with three points
var vertices = [
vec2(-1, -1),
vec2(0, 1),
vec2(1, -1)
];
points = [];
divideTriangle(vertices[0], vertices[1], vertices[2], numTimes);
}
function divideTriangle(a, b, c, count) {
// check for end of recursion
if (count == 0) {
points.push(a, b, c);
}
else {
// bisect the sideds
var ab = mix(a, b, 0.5);
var ac = mix(a, c, 0.5);
var bc = mix(b, c, 0.5);
count--;
// three new triangles
divideTriangle(a, ab, ac, count);
divideTriangle(c, ac, bc, count);
divideTriangle(b, bc, ab, count);
}
}
0 0
Ln 73, Col 42
Spaces: 4
UTF-8
CRLF
JavaScript

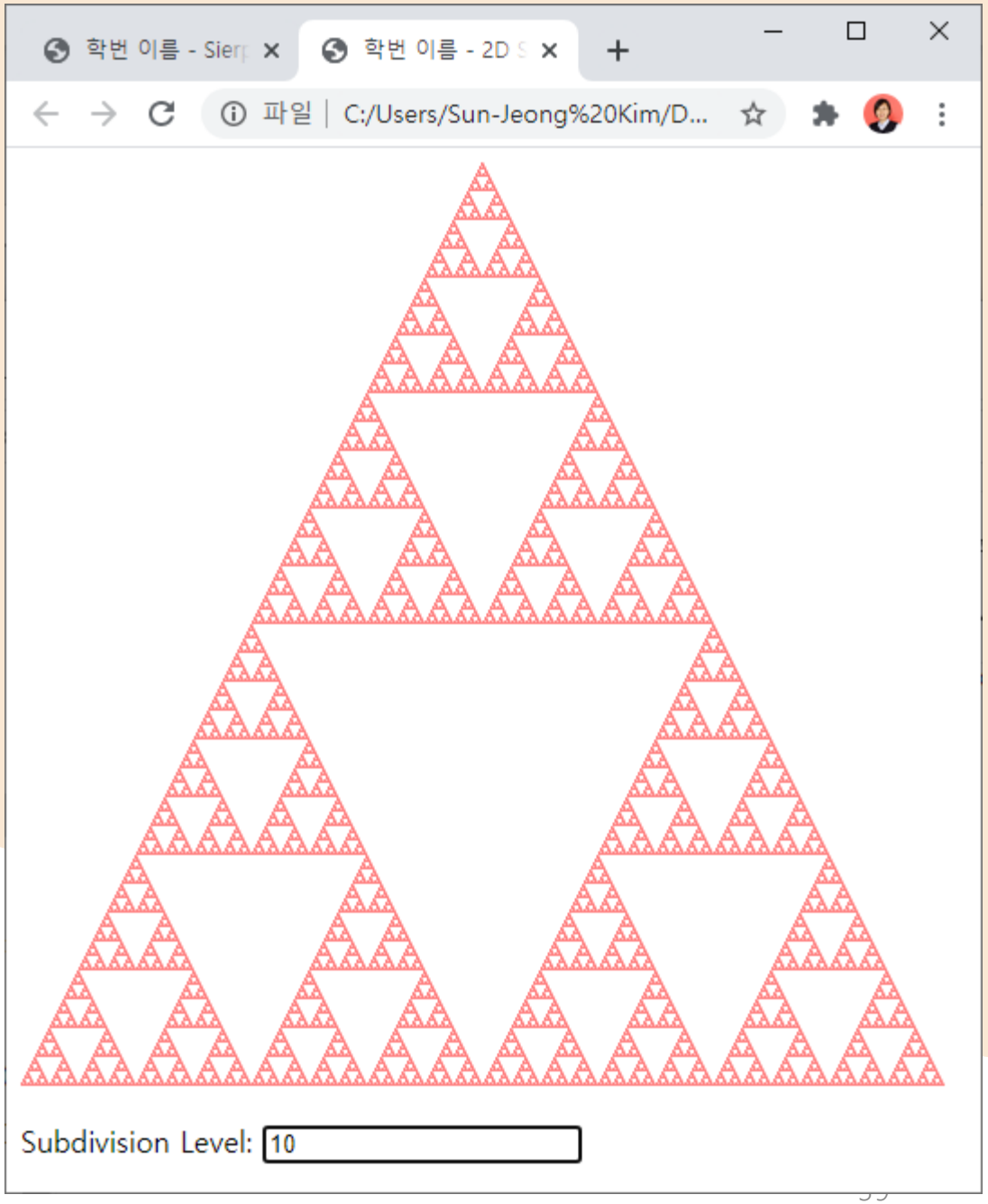

학번 이름 - Sier
학번 이름 - 2D
파일 | C:/Users/Sun-Jeong%20Kim/D...

연습 문제 (1)

- 숫자를 입력 받아 Subdivision을 수행하시오.




학번 이름 - Sier
학번 이름 - 2D
파일 | C:/Users/Sun-Jeong%20Kim/D...
Subdivision Level: 1


학번 이름 - Sier
학번 이름 - 2D
파일 | C:/Users/Sun-Jeong%20Kim/D...
Subdivision Level: 10

수고하셨습니다