

Graphics Programming

3RD WEEK, 2021

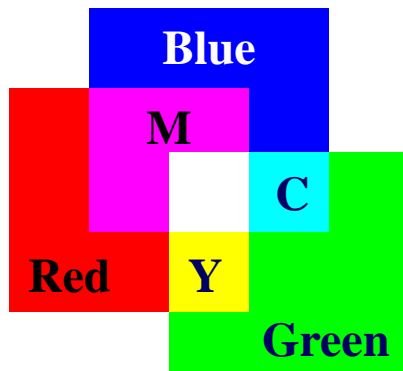


Attributes

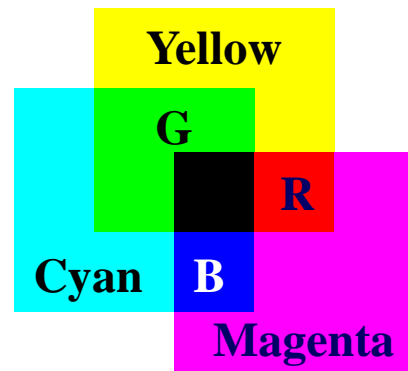
- Properties that determines how to render a geometric primitive (appearance of objects)
 - Color (points, lines, polygons)
 - Size and width (points, lines)
 - Stipple pattern (lines, polygons)
 - Polygon mode
 - Display as filled: solid color or stipple pattern
 - Display edges
 - Display vertices
- Only a few (**gl_PointSize**) are supported by OpenGL functions

Color

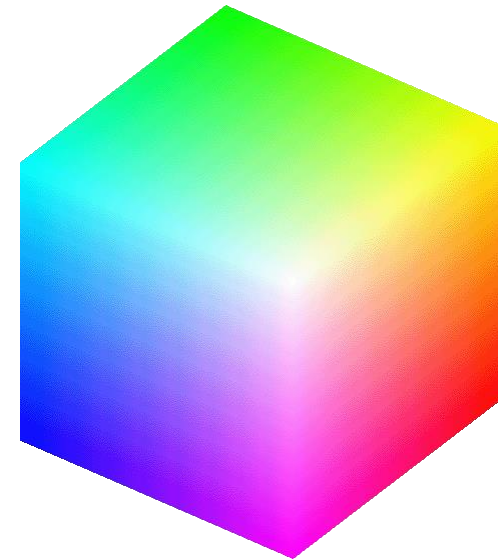
- Three color theory
 - Our brains do not receive the entire color distribution but rather than three values
- Additive color – ex) CRT
- Subtractive color – ex) printing



Additive Color



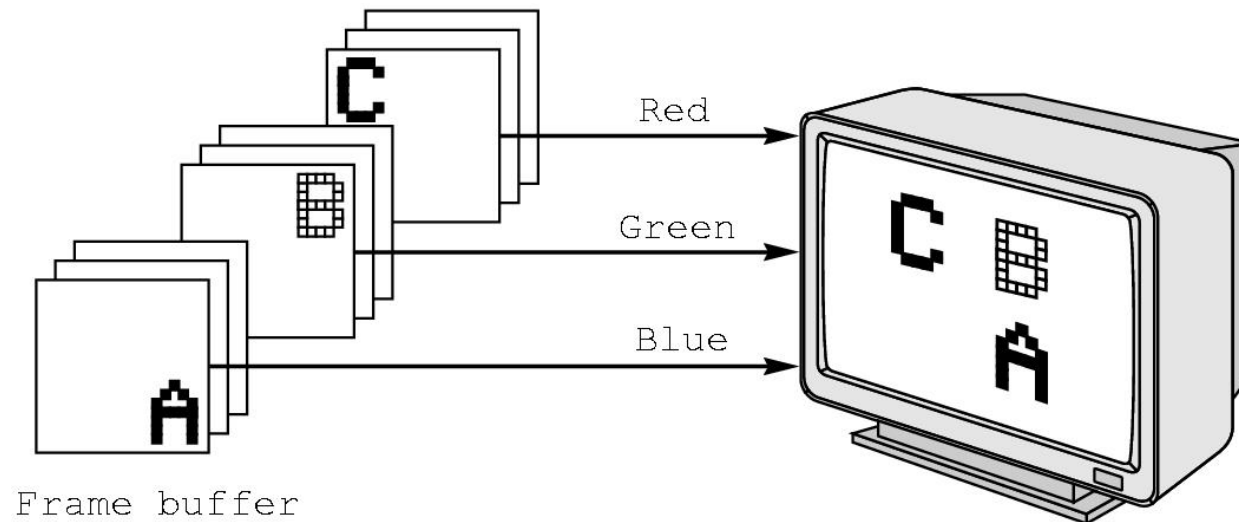
Subtractive Color



Color Solid

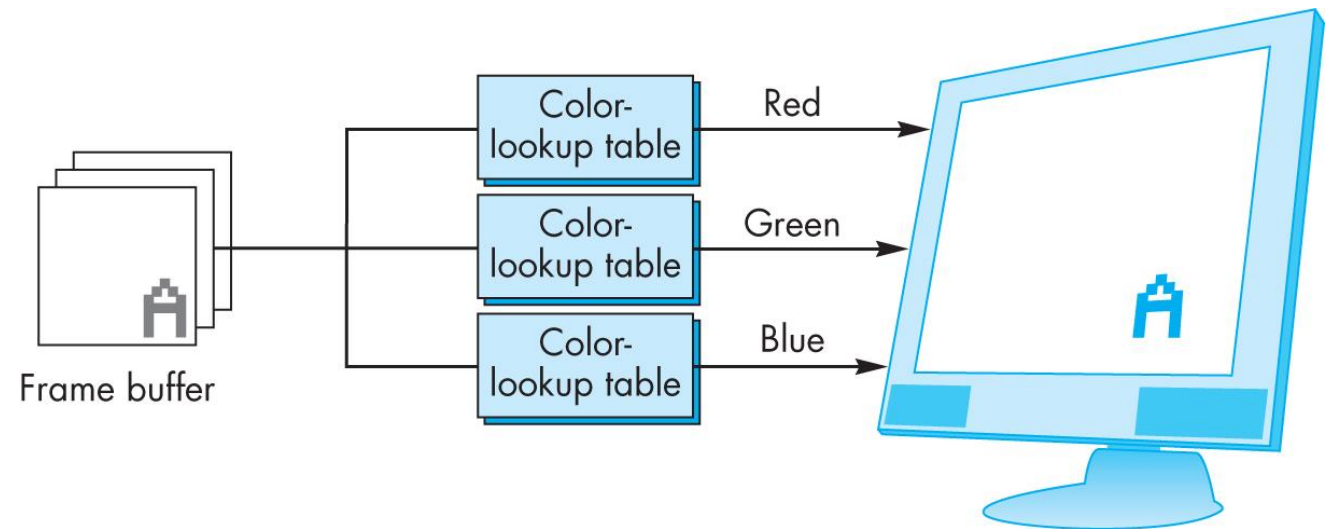
RGB Color

- Each color component is stored separately in the frame buffer
 - Usually 8 bits per component in buffer
 - Color values can range from 0.0 (none) to 1.0 (all) using floats or over the range from 0 to 255 using unsigned bytes



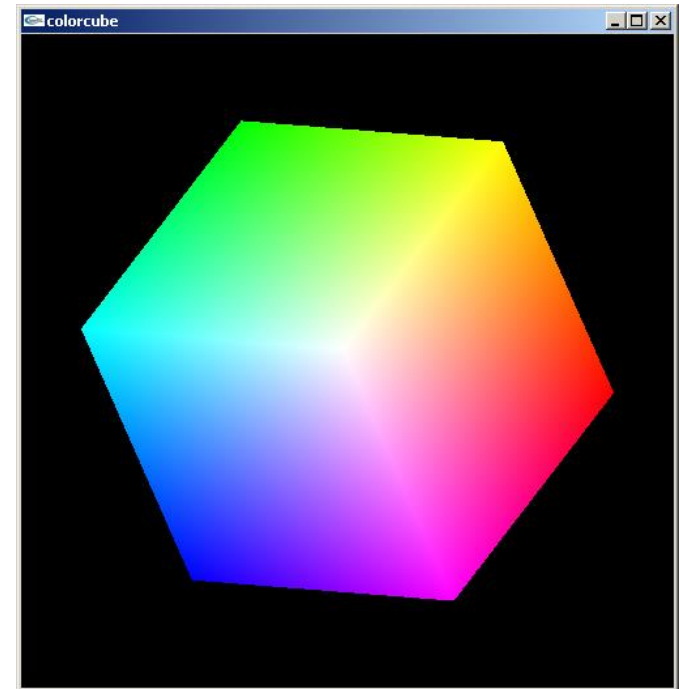
Indexed Color

- Colors are indices into tables of RGB values
- Requires less memory
 - indices usually 8 bits
 - not as important now
 - Memory inexpensive
 - Need more colors for shading



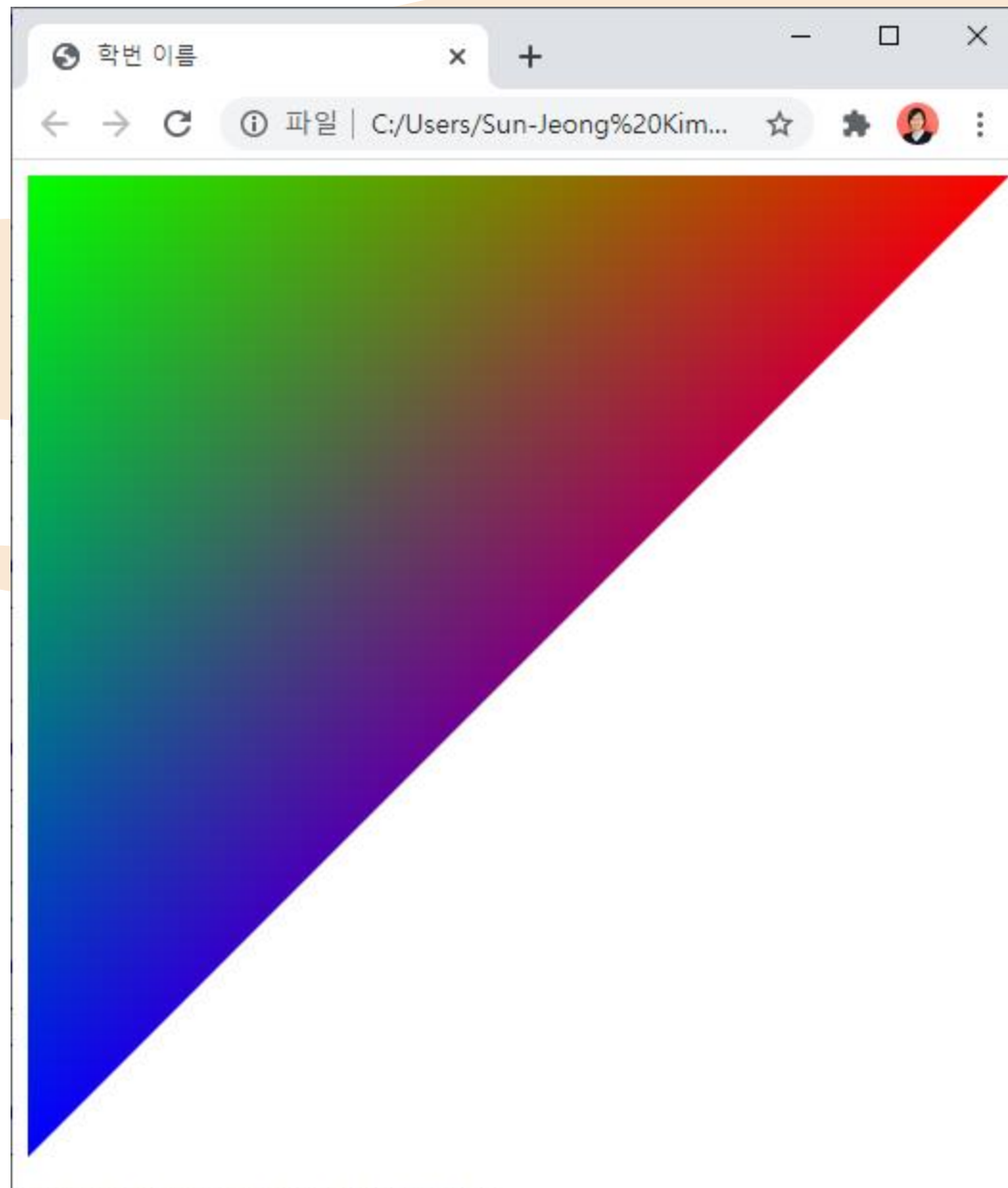
Smooth Color

- Default is smooth shading
 - OpenGL interpolates vertex colors across visible polygons
- Alternative is flat shading
 - Color of first vertex determines fill color
 - Handle in shader



Setting Colors

- Colors are ultimately set in the fragment shader but can be determined in either shader or in the application
- Application color: pass to vertex shader as a uniform variable or as a vertex attribute
- Vertex shader color: pass to fragment shader as varying variable
- Fragment color: can alter via shader code



colorTriangle.html X JS colorTriangle.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> colorTriangle.html > html > head > script

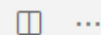
```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>학번 이름</title>
5          <script id="vertex-shader" type="x-shader/x-vertex">
6              attribute vec4 vPosition;
7              attribute vec4 vColor;
8              varying vec4 color;
9
10             void main() {
11                 gl_Position = vPosition;
12                 color = vColor;
13             }
14         </script>
15
16         <script id="fragment-shader" type="x-shader/x-fragment">
17             precision mediump float;
18             varying vec4 color;
19
20             void main() {
21                 gl_FragColor = color;
22             }
23         </script>
24
25         <script type="text/javascript" src="Common/webgl-utils.js"></script>
26         <script type="text/javascript" src="Common/initShaders.js"></script>
27         <script type="text/javascript" src="Common/MV.js"></script>
28         <script type="text/javascript" src="colorTriangle.js"></script>
29     </head>
30     <body>
31         <canvas id="gl-canvas" width="512" height="512">
32             Oops... your browser doesn't support the HTML5 canvas element!
33         </canvas>
34     </body>
35 </html>
```





colorTriangle.html

JS colorTriangle.js X



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

```
1  var gl;
2
3  window.onload = function init()
4  {
5      var canvas = document.getElementById("gl-canvas");
6
7      gl = WebGLUtils.setupWebGL(canvas);
8      if( !gl ) {
9          alert("WebGL isn't available!");
10     }
11
12     var vertices = [
13         vec2(-1, -1),
14         vec2(-1, 1),
15         vec2(1, 1)
16     ];
17
18     var colors = [
19         vec4(0, 0, 1, 1),
20         vec4(0, 1, 0, 1),
21         vec4(1, 0, 0, 1)
22     ]
23
24     // Configure WebGL
25     gl.viewport(0, 0, canvas.width, canvas.height);
26     gl.clearColor(1.0, 1.0, 1.0, 1.0);
27
28     // Load shaders and initialize attribute buffers
29     var program = initShaders(gl, "vertex-shader", "fragment-shader");
30     gl.useProgram(program);
31
32     // Load the data into the GPU
33     var bufferId = gl.createBuffer();
34     gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
35     gl.bufferData(gl.ARRAY_BUFFER, flatten(vertices), gl.STATIC_DRAW);
```

```
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 // ...
```

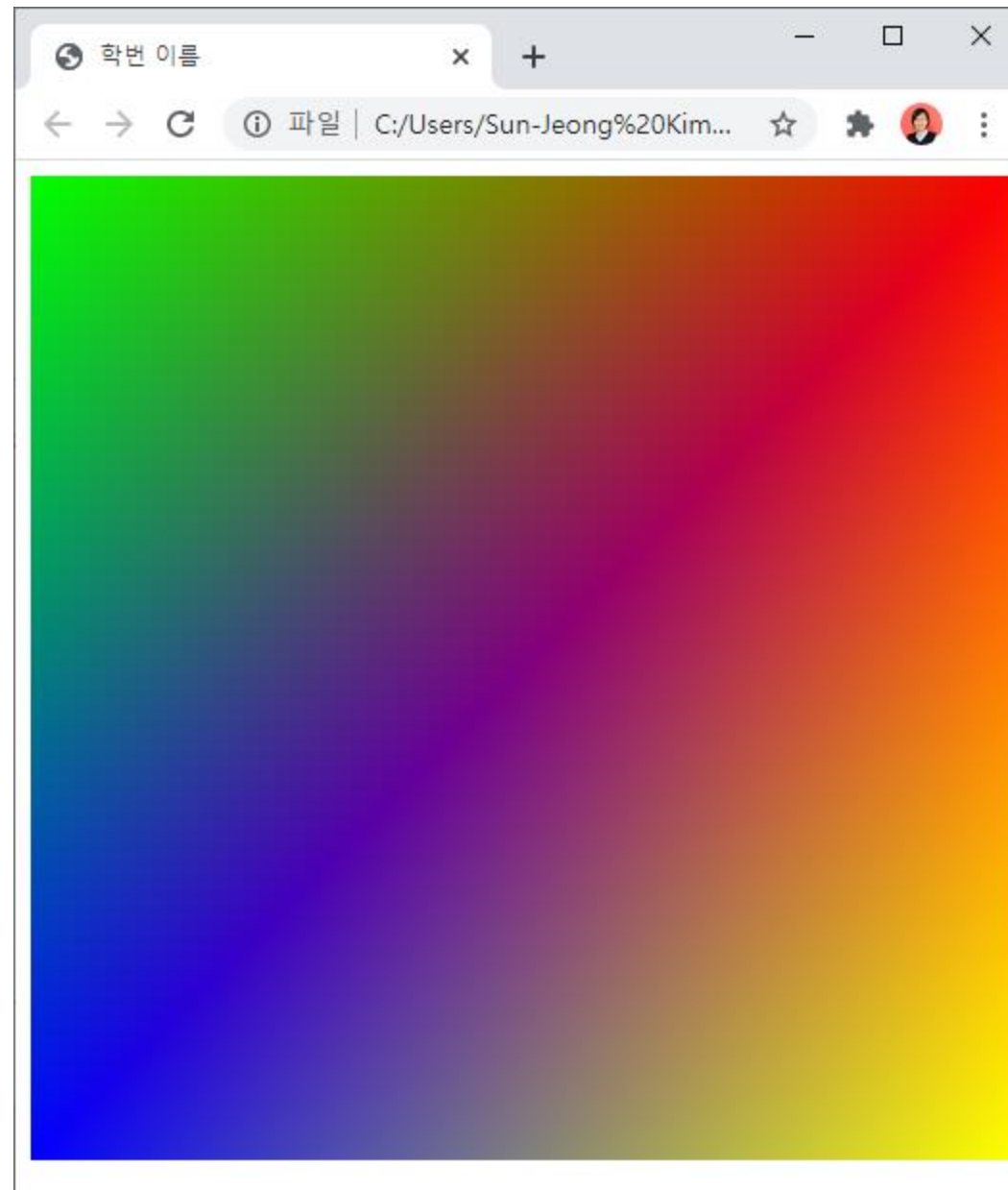
```

10 #
11 # This file contains the definitions of the various
12 # macros used in the code.
13 #
14 # The macros are defined in a way that they can be
15 # used in a portable manner.
16 #
17 # The macros are defined in a way that they can be
18 # used in a portable manner.
19 #
20 # The macros are defined in a way that they can be
21 # used in a portable manner.
22 #
23 # The macros are defined in a way that they can be
24 # used in a portable manner.
25 #
26 # The macros are defined in a way that they can be
27 # used in a portable manner.
28 #
29 # The macros are defined in a way that they can be
30 # used in a portable manner.
31 #
32 # The macros are defined in a way that they can be
33 # used in a portable manner.
34 #
35 # The macros are defined in a way that they can be
36 # used in a portable manner.
37 #
38 # The macros are defined in a way that they can be
39 # used in a portable manner.
40 #
41 # The macros are defined in a way that they can be
42 # used in a portable manner.
43 #
44 # The macros are defined in a way that they can be
45 # used in a portable manner.
46 #
47 # The macros are defined in a way that they can be
48 # used in a portable manner.
49 #
50 # The macros are defined in a way that they can be
51 # used in a portable manner.
52 #
53 # The macros are defined in a way that they can be
54 # used in a portable manner.
55 #
56 # The macros are defined in a way that they can be
57 # used in a portable manner.
58 #
59 # The macros are defined in a way that they can be
60 # used in a portable manner.
61 #
62 # The macros are defined in a way that they can be
63 # used in a portable manner.
64 #
65 # The macros are defined in a way that they can be
66 # used in a portable manner.
67 #
68 # The macros are defined in a way that they can be
69 # used in a portable manner.
70 #
71 # The macros are defined in a way that they can be
72 # used in a portable manner.
73 #
74 # The macros are defined in a way that they can be
75 # used in a portable manner.
76 #
77 # The macros are defined in a way that they can be
78 # used in a portable manner.
79 #
80 # The macros are defined in a way that they can be
81 # used in a portable manner.
82 #
83 # The macros are defined in a way that they can be
84 # used in a portable manner.
85 #
86 # The macros are defined in a way that they can be
87 # used in a portable manner.
88 #
89 # The macros are defined in a way that they can be
90 # used in a portable manner.
91 #
92 # The macros are defined in a way that they can be
93 # used in a portable manner.
94 #
95 # The macros are defined in a way that they can be
96 # used in a portable manner.
97 #
98 # The macros are defined in a way that they can be
99 # used in a portable manner.
100 #

```

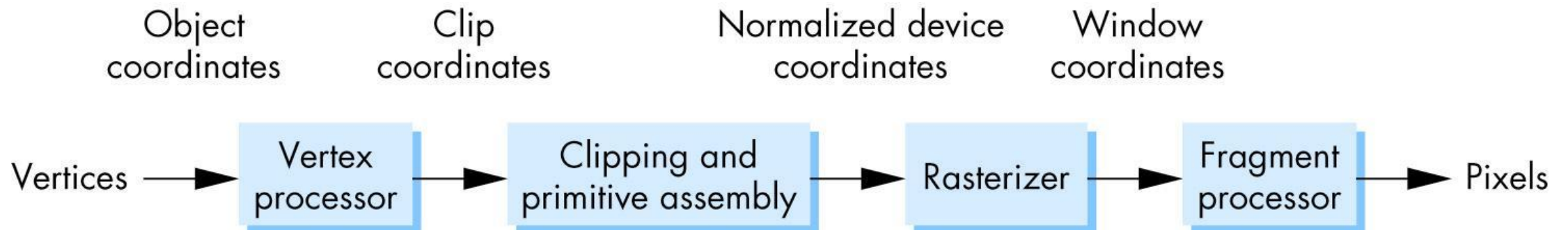
연습 문제 (1)

- 다음과 같이 그리시오.



Programmable Pipelines

- Two components
 - Vertex program (vertex shaders)
 - Fragment program (fragment shaders)
- In the pipeline architecture, the vertex processor and the fragment processor are programmable by application programs called shaders



Vertex Shader Applications

- Moving vertices
 - Morphing
 - Wave motion
 - Fractals
- Lighting
 - More realistic models
 - Cartoon shaders

Fragment Shader Applications (1)

- Per fragment lighting calculations



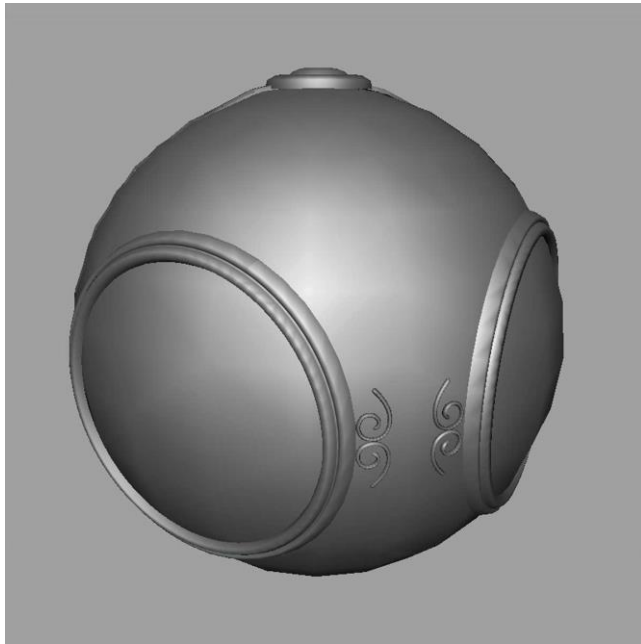
Per vertex lighting



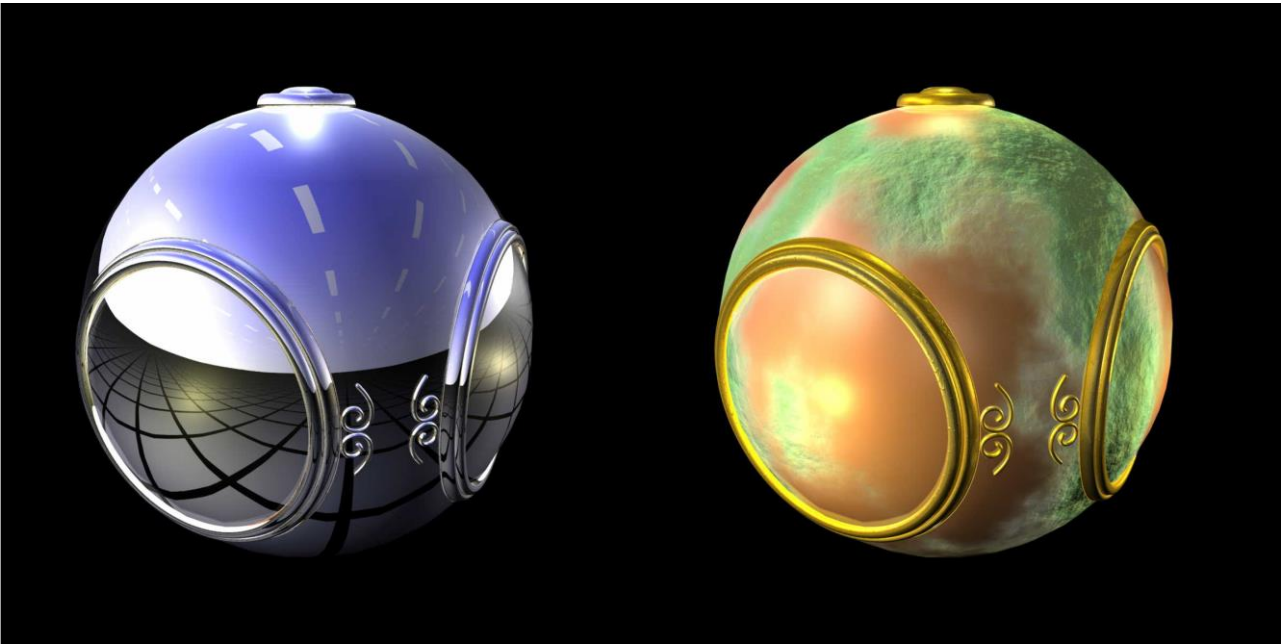
Per fragment lighting

Fragment Shader Applications (2)

- Texture mapping



Smooth shading



Environment
mapping

Bump mapping

Simple Vertex Shader

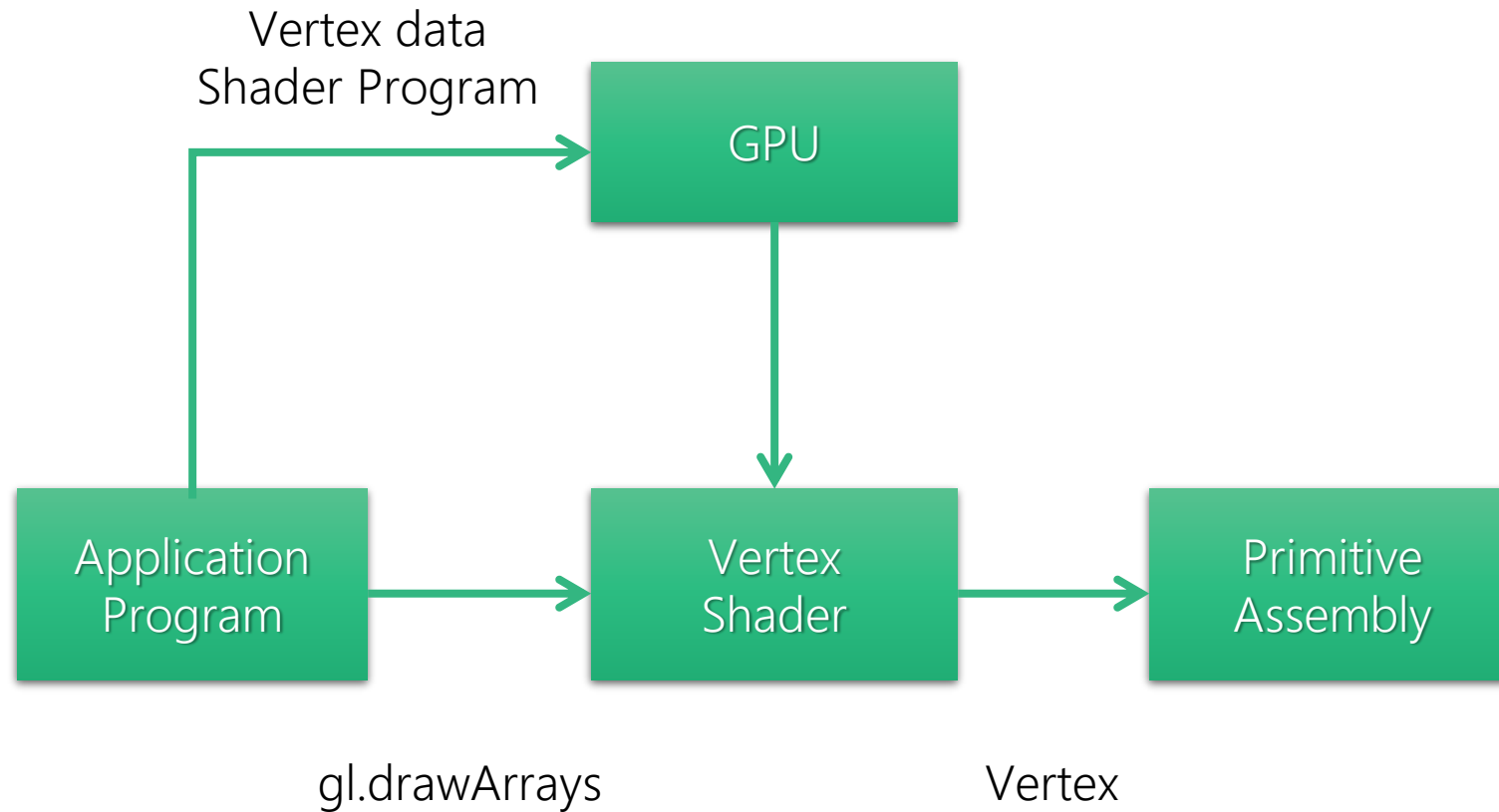
```
attribute vec4 vPosition;  
void main(void)  
{  
    gl_Position = vPosition;  
}
```

input from application

must link to variable in application

built in variable

Execution Model – Vertex Shader



Simple Fragment Shader

```
precision mediump float;
```

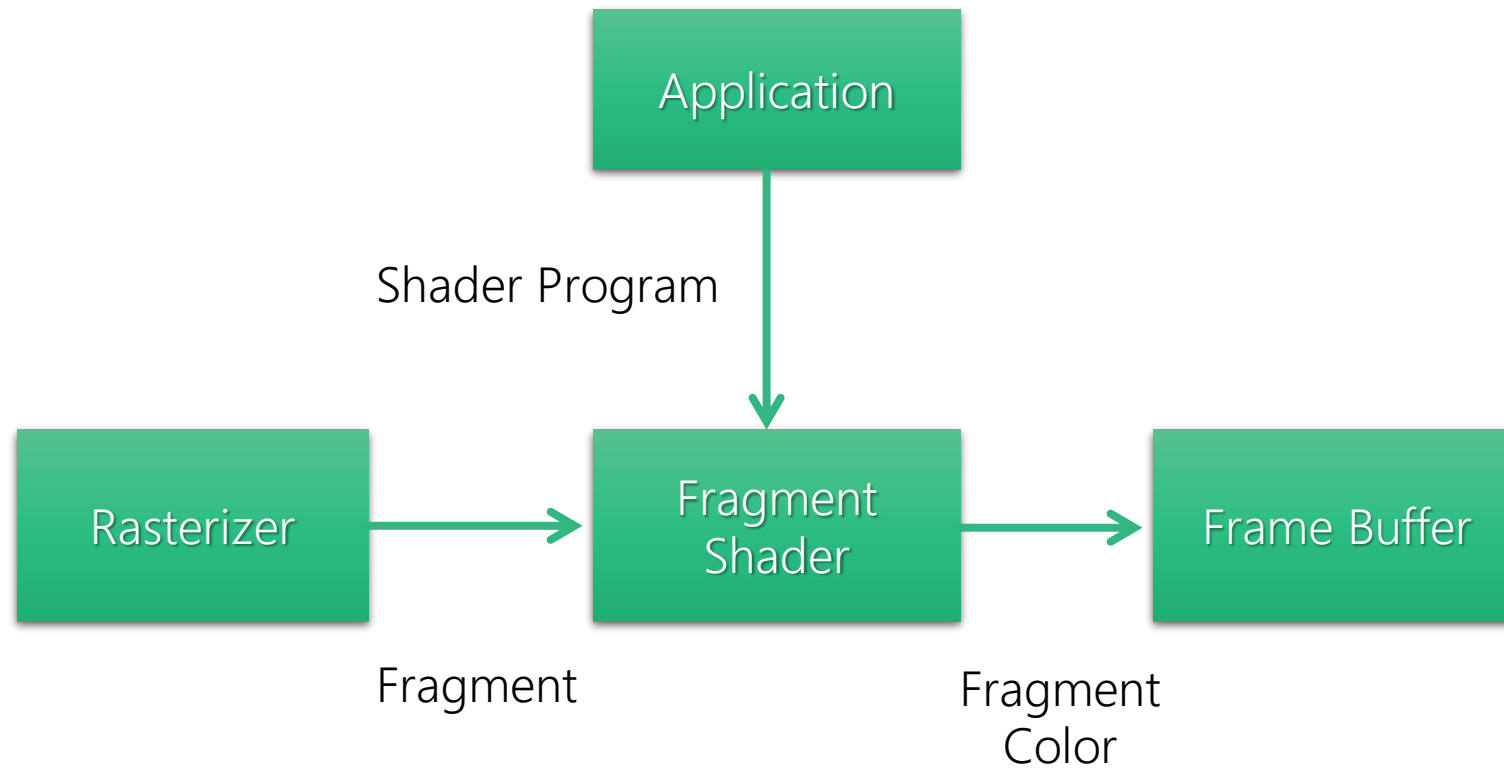
```
void main(void)
```

```
{
```

```
    gl_FragColor = vec4( 1.0, 0.0, 0.0, 1.0 );
```

```
}
```

Execution Model – Fragment Shader



Data Type

- C types: `int`, `float`, `bool`
- Vectors:
 - float `vec2`, `vec3`, `vec4`
 - Also int (`ivec`) and boolean (`bvec`)
- Matrices: `mat2`, `mat3`, `mat4`
 - Stored by columns
 - Standard referencing `m[row][column]`
- C++ style constructors
 - `vec3 a = vec3(1.0, 2.0, 3.0);`
 - `vec2 b = vec2(a);`

Pointers

- There are no pointers in GLSL
- We can use C structs which can be copied back from functions
- Because matrices and vectors are basic types they can be passed into and output from GLSL functions
 - Ex) `mat3 func(mat3 a) ;`
- Variables passed by copying

Qualifiers

- GLSL has many of the same qualifiers such as **const** as C/C++
- Need others due to the nature of the shader architecture
- Variables can change
 - Once per primitive
 - Once per vertex
 - Once per fragment
 - At any time in the application
- Vertex attributes are interpolated by the rasterizer into fragment attributes

Attribute Qualifiers

- Attribute-qualified variables can change at most once per vertex
- There are a few built in variables such as `gl_Position` but most have been deprecated
- User defined (in application program)
 - `attribute float temperature;`
 - `attribute vec3 velocity;`
 - Recent versions of GLSL use `in` and `out` qualifiers to get to and from shaders

Uniform Qualifiers

- Variables that are constant for an entire primitive
- Can be changed in application and sent to shaders
- Cannot be changed in shader
- Used to pass information to shader such as the bounding box of a primitive

Varying Qualifiers

- Variables that are passed from vertex shader to fragment shader
- Automatically interpolated by the rasterizer
- With WebGL, GLSL uses the varying qualifier in both shaders
 - `varying vec4 color;`
- More recent versions of WebGL use out in vertex shader and in in the fragment shader
 - `out vec4 color; // vertex shader`
 - `in vec4 color; // fragment shader`

Example: Vertex Shader

```
attribute vec4 vPosition;  
attribute vec4 vColor;  
varying vec4 fColor;  
void main(void)  
{  
    gl_Position = vPosition;  
    fColor = vColor;  
}
```

Corresponding Fragment Shader

```
precision mediump float;
```

```
varying vec4 fColor;
```

```
void main(void)
```

```
{
```

```
    gl_FragColor = fColor;
```

```
}
```

Sending Colors from Application

```
var cBuffer = gl.createBuffer();  
gl.bindBuffer(gl.ARRAY_BUFFER, cBuffer);  
gl.bufferData(gl.ARRAY_BUFFER, flatten(colors),  
gl.STATIC_DRAW);  
  
var vColor = gl.getAttribLocation(program, "vColor");  
gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);  
gl.enableVertexAttribArray(vColor);
```

Sending a Uniform Variable

```
// in application
```

```
vec4 color = vec4(1.0, 0.0, 0.0, 1.0);
```

```
colorLoc = gl.getUniformLocation(program, "uColor");
```

```
gl.uniform4f(colorLoc, color);
```

```
// in fragment shader (similar in vertex shader)
```

```
uniform vec4 uColor;
```

```
void main() {
```

```
    gl_FragColor = uColor;
```

```
}
```

Operators and Functions

- Standard C functions
 - Trigonometric
 - Arithmetic
 - Normalize, reflect, length
- Overloading of vector and matrix types

```
mat4 a;  
vec4 b, c, d;  
c = b*a; // a column vector stored as a 1d array  
d = a*b; // a row vector stored as a 1d array
```

Swizzling and Selection

- Can refer to array elements by element using [] or selection (.) operator with

- **x, y, z, w**

- **r, g, b, a**

- **s, t, p, q**

- Ex) **a[2], a.b, a.z, a.p** are the same

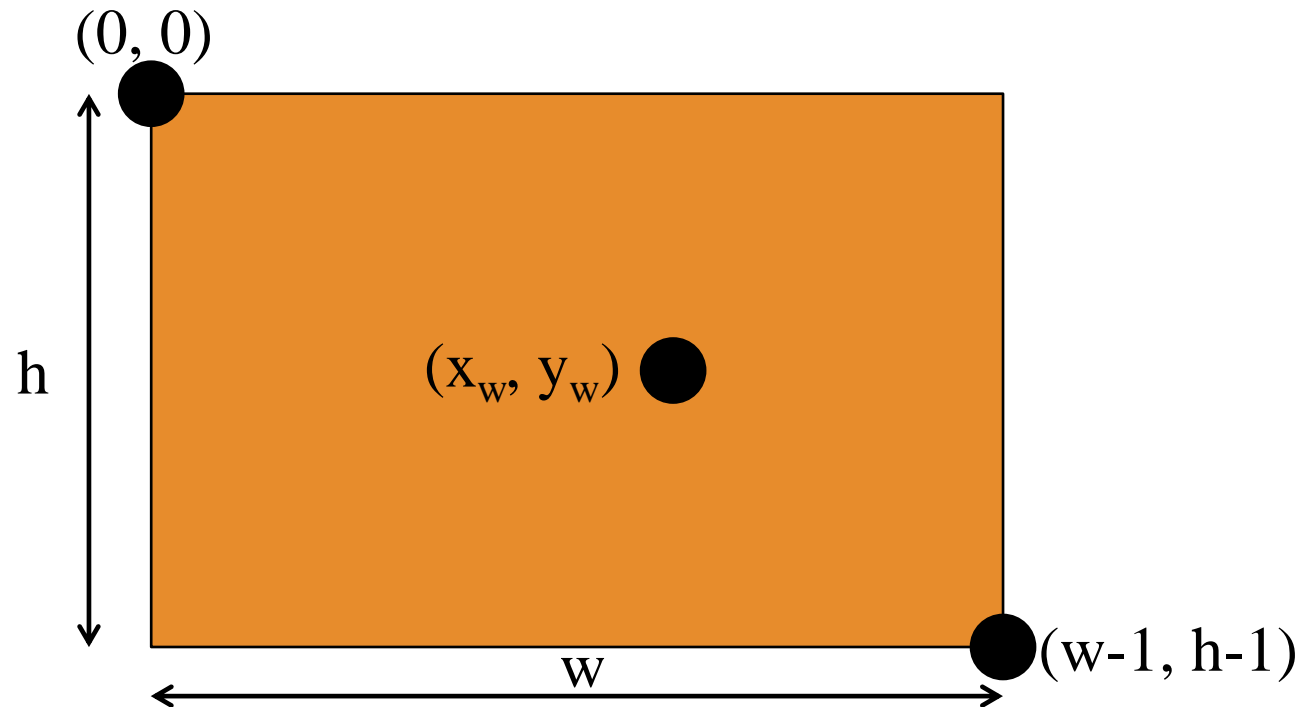
- Swizzling operator lets us manipulate components

```
vec4 a = vec4(1.0, 2.0, 3.0, 4.0);
```

```
a.yz = vec2(1.0, 2.0); // a = (1.0, 1.0, 2.0, 4.0)
```


Position Input

- Returning position from click event
 - Canvas specified in HTML file of size `canvas.width` x `canvas.height`
 - Returned window coordinates are `event.clientX` and `event.clientY`



$$(0, h) \rightarrow (-1, -1)$$

$$(w, 0) \rightarrow (1, 1)$$

$$x = -1 + \frac{2 * x_w}{w}$$

$$y = -1 + \frac{2 * (h - y_w)}{h}$$

drawPoints.html - Visual Studio Code

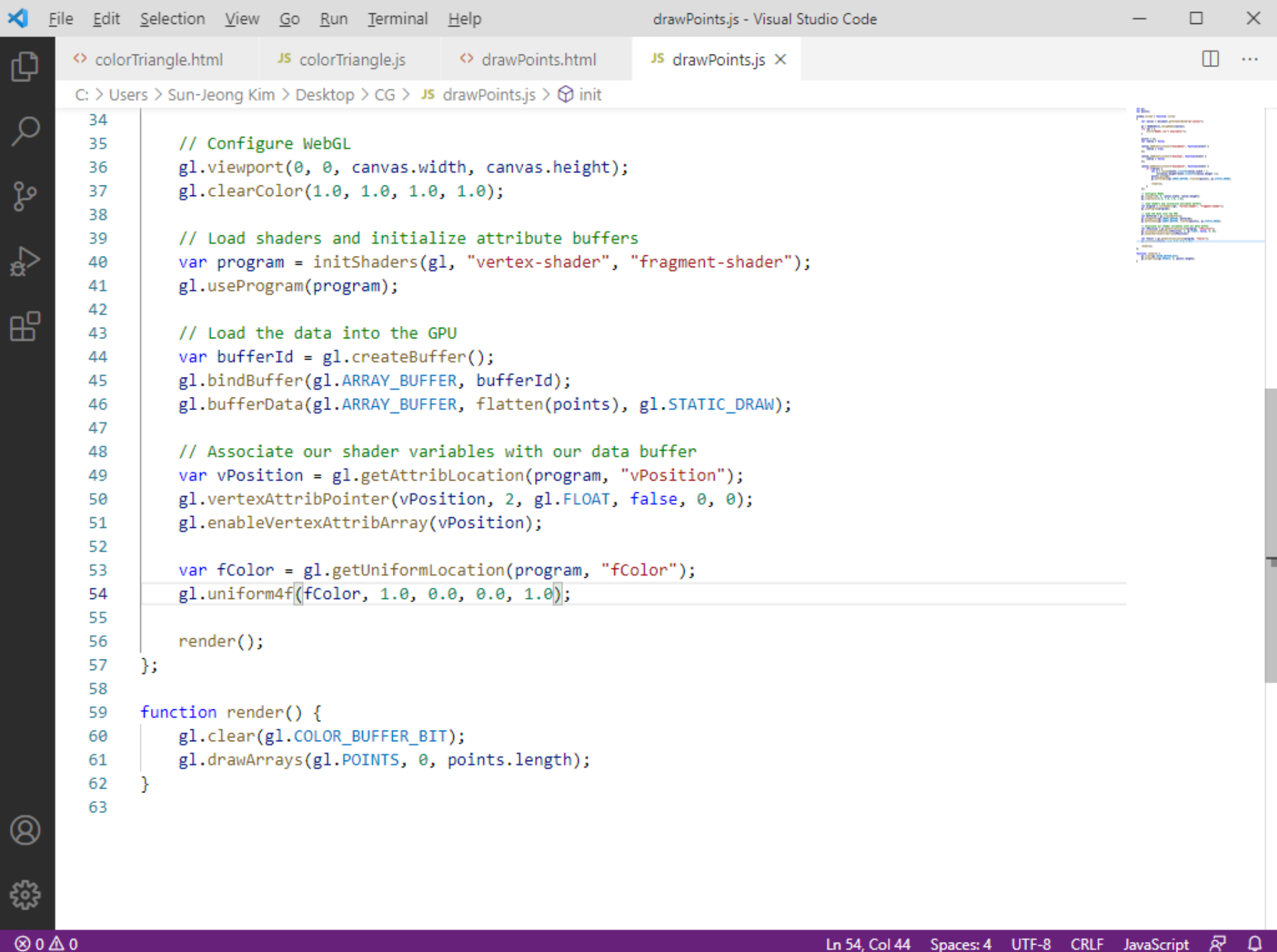
colorTriangle.html JS colorTriangle.js drawPoints.html X JS drawPoints.js

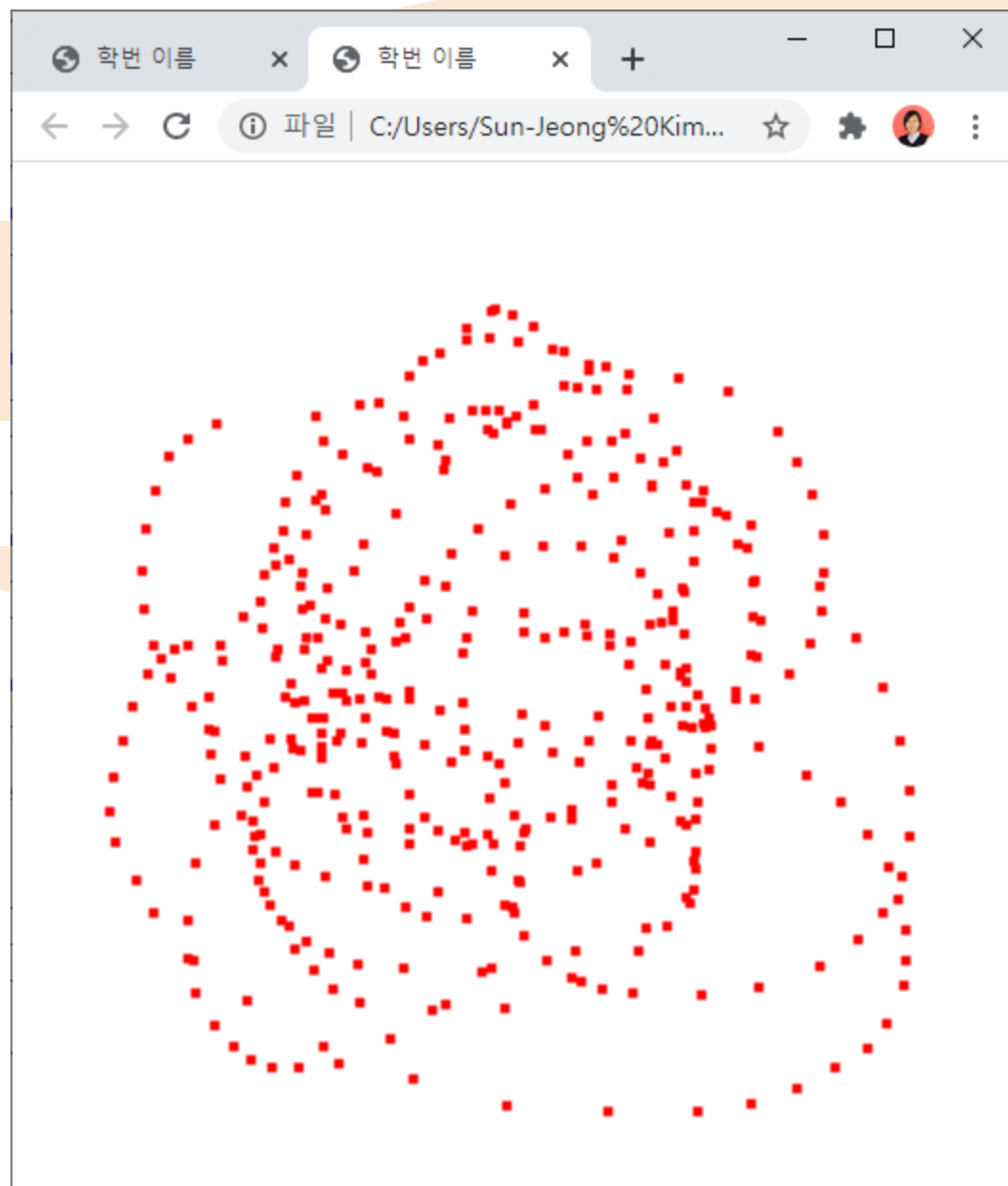
C: > Users > Sun-Jeong Kim > Desktop > CG > > drawPoints.html > html > head > script#fragment-shader

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>학번 이름</title>
5     <script id="vertex-shader" type="x-shader/x-vertex">
6       attribute vec4 vPosition;
7
8       void main() {
9         gl_PointSize = 5.0;
10        gl_Position = vPosition;
11      }
12    </script>
13
14    <script id="fragment-shader" type="x-shader/x-fragment">
15      precision mediump float;
16      uniform vec4 fColor;
17
18      void main() {
19        gl_FragColor = fColor;
20      }
21    </script>
22
23    <script type="text/javascript" src="Common/webgl-utils.js"></script>
24    <script type="text/javascript" src="Common/initShaders.js"></script>
25    <script type="text/javascript" src="Common/MV.js"></script>
26    <script type="text/javascript" src="drawPoints.js"></script>
27  </head>
28  <body>
29    <canvas id="gl-canvas" width="512" height="512">
30      Oops... your browser doesn't support the HTML5 canvas element!
31    </canvas>
32  </body>
33 </html>
```

```
C: > Users > Sun-Jeong Kim > Desktop > CG > JS drawPoints.js > init
1  var gl;
2  var points;
3
4  window.onload = function init()
5  {
6      var canvas = document.getElementById("gl-canvas");
7
8      gl = WebGLUtils.setupWebGL(canvas);
9      if( !gl ) {
10         alert("WebGL isn't available!");
11     }
12
13     points = [];
14     var redraw = false;
15
16     canvas.addEventListener("mousedown", function(event) {
17         redraw = true;
18     });
19
20     canvas.addEventListener("mouseup", function(event) {
21         redraw = false;
22     });
23
24     canvas.addEventListener("mousemove", function(event) {
25         if (redraw) {
26             var p = vec2(2*event.clientX/canvas.width - 1,
27                 2*(canvas.height-event.clientY)/canvas.height -1);
28             points.push(p);
29             gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
30
31             render();
32         }
33     });
34
35     // Configure WebGL
```







연습 문제 (2)

- 웹페이지에 슬라이더를 추가하여, 점의 크기를 입력 받아 그리시오.

```
File Edit Selection View Go Run Terminal Help
drawPoints.html - Visual Studio Code

colorTriangle.html x JS colorTriangle.js x drawPoints.html x JS drawPoints.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> drawPoints.html > html > body > p
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>학번 이름</title>
5     <script id="vertex-shader" type="x-shader/x-vertex">
6       attribute vec4 vPosition;
7       uniform float pointSize;
8
9       void main() {
10        gl_PointSize = pointSize;
11        gl_Position = vPosition;
12      }
13    </script>
14
15    <script id="fragment-shader" type="x-shader/x-fragment">
16      precision mediump float;
17      uniform vec4 fColor;
18
19      void main() {
20        gl_FragColor = fColor;
21      }
22    </script>
23
24    <script type="text/javascript" src="Common/webgl-utils.js"></script>
25    <script type="text/javascript" src="Common/initShaders.js"></script>
26    <script type="text/javascript" src="Common/MV.js"></script>
27    <script type="text/javascript" src="drawPoints.js"></script>
28  </head>
29  <body>
30    <canvas id="gl-canvas" width="512" height="512">
31      Oops... your browser doesn't support the HTML5 canvas element!
32    </canvas>
33    <p>Point Size 1<input type="range" id="pointSize" min="1" max="10" step="1" value="5">10</p>
34  </body>
35 </html>
```

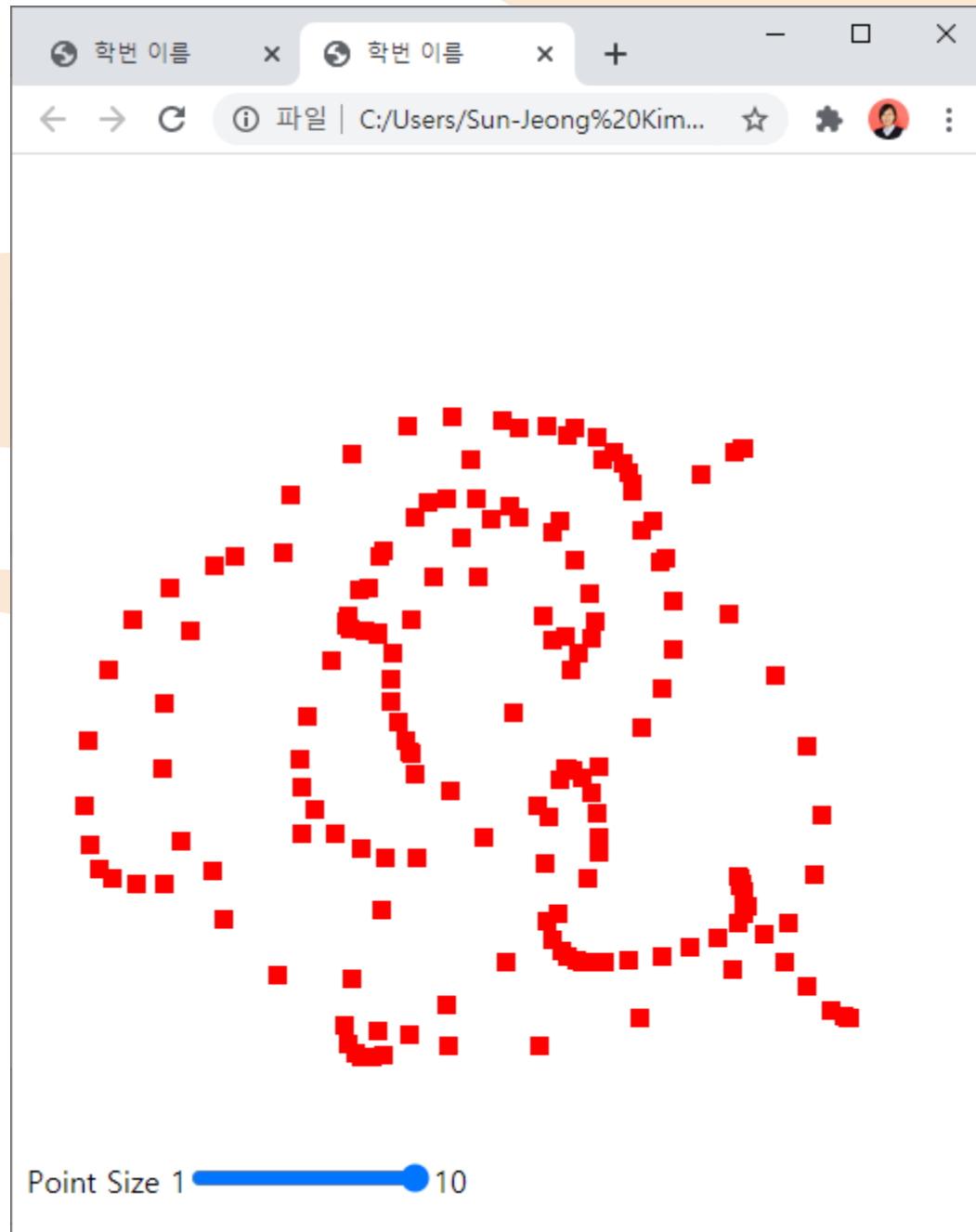


drawPoints.js - Visual Studio Code

colorTriangle.html JS colorTriangle.js drawPoints.html JS drawPoints.js

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

```
42
43 // Load the data into the GPU
44 var bufferId = gl.createBuffer();
45 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
46 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
47
48 // Associate our shader variables with our data buffer
49 var vPosition = gl.getAttribLocation(program, "vPosition");
50 gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
51 gl.enableVertexAttribArray(vPosition);
52
53 var fColor = gl.getUniformLocation(program, "fColor");
54 gl.uniform4f(fColor, 1.0, 0.0, 0.0, 1.0);
55
56 var locPointSize = gl.getUniformLocation(program, "pointSize");
57 gl.uniform1f(locPointSize, 5.0);
58
59 document.getElementById("pointSize").onchange = function () {
60     var size = this.value;
61     gl.uniform1f(locPointSize, size);
62
63     render();
64 }
65
66 render();
67 };
68
69 function render() {
70     gl.clear(gl.COLOR_BUFFER_BIT);
71     gl.drawArrays(gl.POINTS, 0, points.length);
72 }
73
```

연습 문제 (3)

- 점마다 색상을 다르게 채색하시오.

drawPoints.html - Visual Studio Code

colorTriangle.html JS colorTriangle.js drawPoints.html X JS drawPoints.js

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

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>학번 이름</title>
5     <script id="vertex-shader" type="x-shader/x-vertex">
6       attribute vec4 vPosition;
7       attribute vec4 vColor;
8       uniform float pointSize;
9       varying vec4 fColor;
10
11     void main() {
12       gl_PointSize = pointSize;
13       gl_Position = vPosition;
14       fColor = vColor;
15     }
16   </script>
17
18   <script id="fragment-shader" type="x-shader/x-fragment">
19     precision mediump float;
20     varying vec4 fColor;
21
22     void main() {
23       gl_FragColor = fColor;
24     }
25   </script>
26
27   <script type="text/javascript" src="Common/webgl-utils.js"></script>
28   <script type="text/javascript" src="Common/initShaders.js"></script>
29   <script type="text/javascript" src="Common/MV.js"></script>
30   <script type="text/javascript" src="drawPoints.js"></script>
31 </head>
32 <body>
33   <canvas id="gl-canvas" width="512" height="512">
34     Oops... your browser doesn't support the HTML5 canvas element!
35   </canvas>
```

Ln 20, Col 28 Spaces: 4 UTF-8 CRLF HTML

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

1  var gl;
2  var points, colors;
3
4  window.onload = function init()
5  {
6      var canvas = document.getElementById("gl-canvas");
7
8      gl = WebGLUtils.setupWebGL(canvas);
9      if( !gl ) {
10         alert("WebGL isn't available!");
11     }
12
13     points = [];
14     colors = [];
15     var redraw = false;
16
17     canvas.addEventListener("mousedown", function(event) {
18         redraw = true;
19     });
20
21     canvas.addEventListener("mouseup", function(event) {
22         redraw = false;
23     });
24
25     canvas.addEventListener("mousemove", function(event) {
26         if (redraw) {
27             var p = vec2(2*event.clientX/canvas.width - 1,
28                 2*(canvas.height-event.clientY)/canvas.height -1);
29             points.push(p);
30             gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
31             gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
32
33             var c = vec4(Math.random(), Math.random(), Math.random(), 1.0);
34             colors.push(c);
35             gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
```



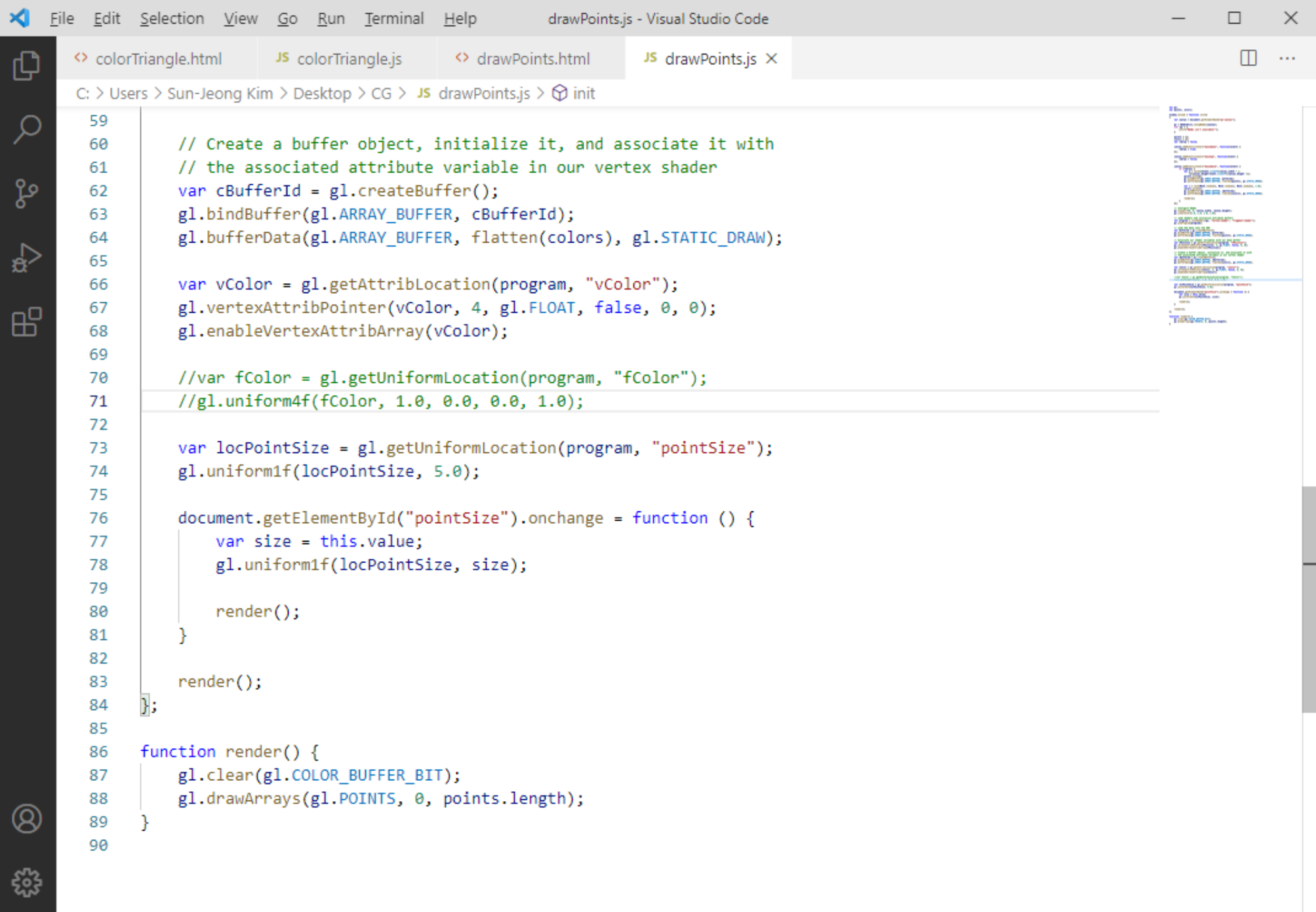
drawPoints.js - Visual Studio Code

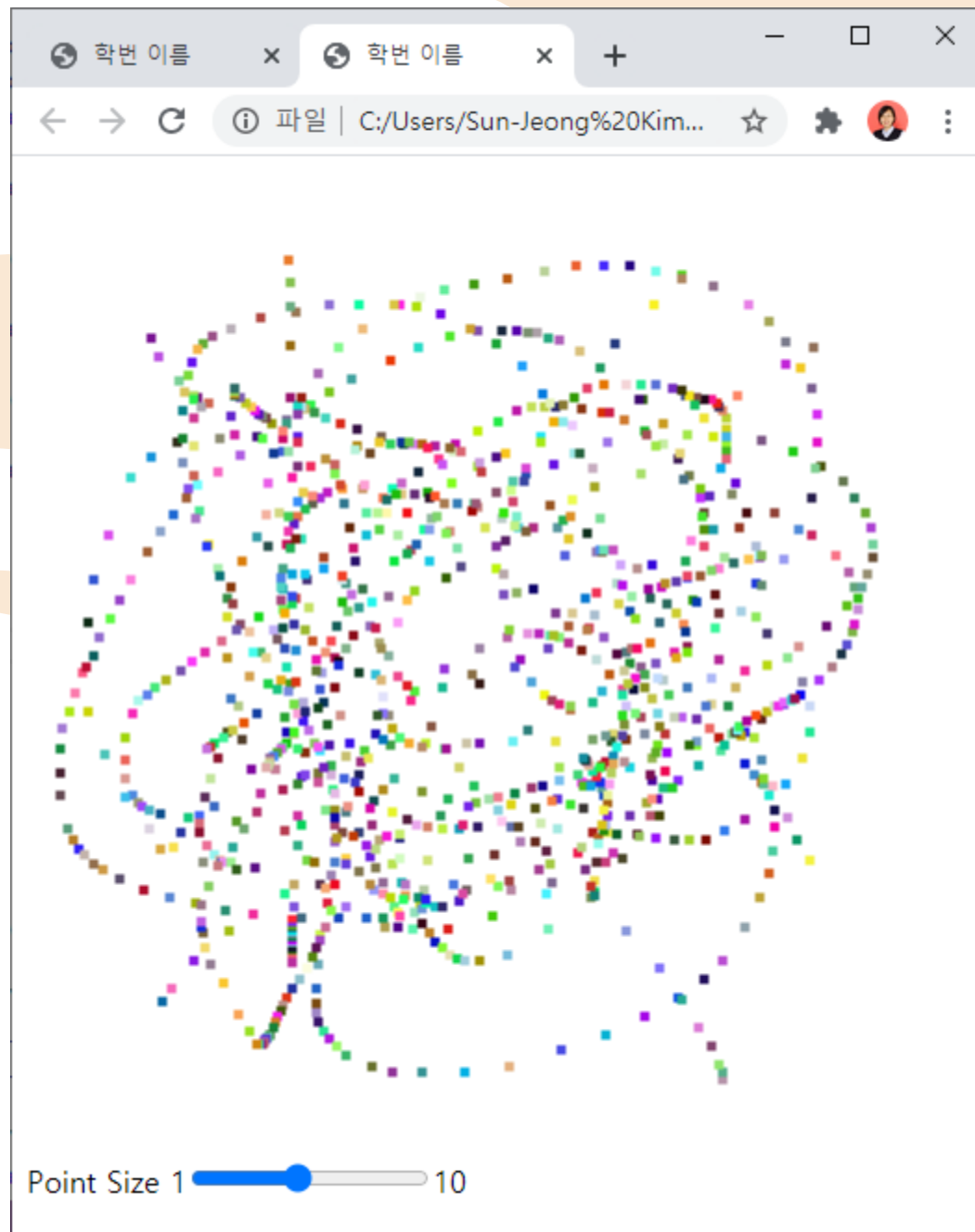
colorTriangle.html JS colorTriangle.js drawPoints.html JS drawPoints.js X

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

```
25 canvas.addEventListener("mousemove", function(event) {
26     if (redraw) {
27         var p = vec2(2*event.clientX/canvas.width - 1,
28             2*(canvas.height-event.clientY)/canvas.height - 1);
29         points.push(p);
30         gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
31         gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
32
33         var c = vec4(Math.random(), Math.random(), Math.random(), 1.0);
34         colors.push(c);
35         gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
36         gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);
37
38         render();
39     }
40 });
41
42 // Configure WebGL
43 gl.viewport(0, 0, canvas.width, canvas.height);
44 gl.clearColor(1.0, 1.0, 1.0, 1.0);
45
46 // Load shaders and initialize attribute buffers
47 var program = initShaders(gl, "vertex-shader", "fragment-shader");
48 gl.useProgram(program);
49
50 // Load the data into the GPU
51 var bufferId = gl.createBuffer();
52 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
53 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
54
55 // Associate our shader variables with our data buffer
56 var vPosition = gl.getAttribLocation(program, "vPosition");
57 gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
58 gl.enableVertexAttribArray(vPosition);
59
```

Ln 71, Col 7 Spaces: 4 UTF-8 CRLF JavaScript





Linking Shader with Application

- Read shaders
- Compile shaders
- Create a program object
- Link everything together
- Link variables in application with variables in shaders
 - Vertex attributes
 - Uniform variables

Program Object

- Container for shaders
 - Can contain multiple shaders
 - Other GLSL functions

```
var program = gl.createProgram();
```

```
/* define shader objects here */
```

```
gl.attachShader( program, vertShdr );  
gl.attachShader( program, fragShdr );  
gl.linkProgram( program );
```

Reading a Shader

- Shaders are added to the program object and compiled
- Usual method of passing a shader is as a null-terminated string using the function `glShaderSource(shdr, text);`
- If the shader is in HTML file, we can get it into application by `getElementById` method
- If the shader is in a file, we can write a reader to convert the file to a string

Adding a Vertex Shader

```
var vertShdr;  
  
var vertElem = document.getElementById(vertexShaderId);  
  
vertShdr = gl.createShader( gl.VERTEX_SHADER );  
  
gl.shaderSource( vertShdr, vertElem.text );  
gl.compileShader( vertShdr );  
  
// after program object created  
gl.attachShader( program, vertShdr );
```

Shader Reader

- Following code may be a security issue with some browsers if you try to run it locally
 - Cross origin request

```
function getShader( gl, shaderName, type ) {  
    var shader = gl.createShader( type );  
    shaderScript = loadFileAJAX( shaderName );  
    if( !shaderScript ) {  
        alert( "Could not find shader source:"  
            + shaderName );  
    }  
}
```

Precision Declaration

- In GLSL for WebGL we must specify desired precision in fragment shaders
 - Artifact inherited from OpenGL ES
 - ES must run on very simple embedded devices that may not support 32-bit floating point
 - All implementations must support mediump
 - No default for float in fragment shader
- Can use preprocessor directives (`#ifdef`) to check if highp supported and, if not, default to mediump

Pass Through Fragment Shader

```
#ifdef GL_FRAGMENT_SHADER_PRECISION_HIGH
precision highp float;
#else
precision mediump float;
#endif
```

```
varying vec4 fColor;
void main(void) {
    gl_FragColor = fColor;
}
```

연습 문제 (4)


- Drag하여 선분들을 그리시오.

drawPoints.html - Visual Studio Code

colorTriangle.html JS colorTriangle.js drawPoints.html X JS drawPoints.js

C: > Users > Sun-Jeong Kim > Desktop > CG > <> drawPoints.html > html > body > div

```
18     <script id="fragment-shader" type="x-shader/x-fragment">
19         precision mediump float;
20         varying vec4 fColor;
21
22         void main() {
23             gl_FragColor = fColor;
24         }
25     </script>
26
27     <script type="text/javascript" src="Common/webgl-utils.js"></script>
28     <script type="text/javascript" src="Common/initShaders.js"></script>
29     <script type="text/javascript" src="Common/MV.js"></script>
30     <script type="text/javascript" src="drawPoints.js"></script>
31 </head>
32 <body>
33     <canvas id="gl-canvas" width="512" height="512">
34         Oops... your browser doesn't support the HTML5 canvas element!
35     </canvas>
36     <p>Point Size 1<input type="range" id="pointSize" min="1" max="10" step="1" value="5">10</p>
37     <div>
38         Choose color:
39         <select id = "colors">
40             <option value = "0">Black</option>
41             <option value = "1">Red</option>
42             <option value = "2">Yellow</option>
43             <option value = "3">Green</option>
44             <option value = "4">Blue</option>
45             <option value = "5">Magenta</option>
46             <option value = "6">Cyan</option>
47         </select>
48     </div>
49 </body>
50 </html>
```



Ln 38, Col 27 Spaces: 4 UTF-8 CRLF HTML


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

1  var gl;
2  var points, colors;
3
4  window.onload = function init()
5  {
6      var canvas = document.getElementById("gl-canvas");
7
8      gl = WebGLUtils.setupWebGL(canvas);
9      if( !gl ) {
10         alert("WebGL isn't available!");
11     }
12
13     points = [];
14     colors = [];
15     var redraw = false;
16     var colorArray = [
17         vec4(0.0, 0.0, 0.0, 1.0), // black
18         vec4(1.0, 0.0, 0.0, 1.0), // red
19         vec4(1.0, 1.0, 0.0, 1.0), // yellow
20         vec4(0.0, 1.0, 0.0, 1.0), // green
21         vec4(0.0, 0.0, 1.0, 1.0), // blue
22         vec4(1.0, 0.0, 1.0, 1.0), // magenta
23         vec4(0.0, 1.0, 1.0, 1.0) // cyan
24     ];
25     var cIndex = 0;
26
27     document.getElementById("colors").onclick = function (event) {
28         cIndex = event.target.value;
29     }
30
31     canvas.addEventListener("mousedown", function(event) {
32         if (!redraw) {
33             var p = vec2(2*event.clientX/canvas.width - 1,
34                 2*(canvas.height-event.clientY)/canvas.height -1);
35             points.push(p);
```




drawPoints.js - Visual Studio Code

File Edit Selection View Go Run Terminal Help

colorTriangle.html JS colorTriangle.js drawPoints.html JS drawPoints.js X

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

```
30
31 canvas.addEventListener("mousedown", function(event) {
32     if (!redraw) {
33         var p = vec2(2*event.clientX/canvas.width - 1,
34             2*(canvas.height-event.clientY)/canvas.height -1);
35         points.push(p);
36         points.push(p);
37         gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
38         gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
39
40         var c = colorArray[cIndex];
41         colors.push(c);
42         colors.push(c);
43         gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
44         gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);
45     }
46     redraw = true;
47 });
48
49 canvas.addEventListener("mouseup", function(event) {
50     redraw = false;
51 });
52
53 canvas.addEventListener("mousemove", function(event) {
54     if (redraw) {
55         var p = vec2(2*event.clientX/canvas.width - 1,
56             2*(canvas.height-event.clientY)/canvas.height -1);
57         points.pop();
58         points.push(p);
59         gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
60         gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
61
62         render();
63     }
64 });
```



Ln 112, Col 27 Spaces: 4 UTF-8 CRLF JavaScript

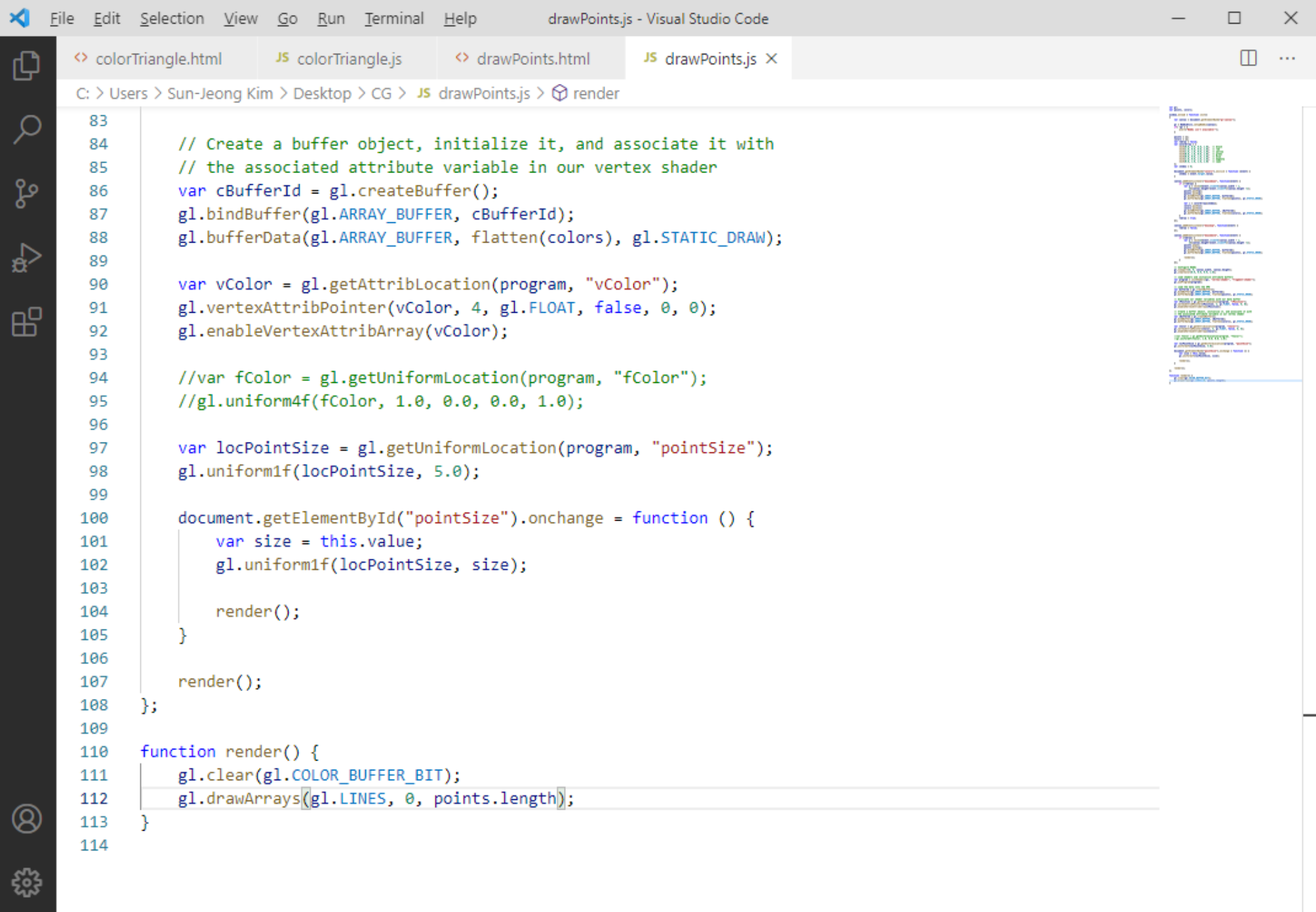
drawPoints.js - Visual Studio Code

colorTriangle.html JS colorTriangle.js drawPoints.html JS drawPoints.js X

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

```
65
66 // Configure WebGL
67 gl.viewport(0, 0, canvas.width, canvas.height);
68 gl.clearColor(0.9, 0.9, 0.9, 1.0);
69
70 // Load shaders and initialize attribute buffers
71 var program = initShaders(gl, "vertex-shader", "fragment-shader");
72 gl.useProgram(program);
73
74 // Load the data into the GPU
75 var bufferId = gl.createBuffer();
76 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
77 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
78
79 // Associate our shader variables with our data buffer
80 var vPosition = gl.getAttribLocation(program, "vPosition");
81 gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0);
82 gl.enableVertexAttribArray(vPosition);
83
84 // Create a buffer object, initialize it, and associate it with
85 // the associated attribute variable in our vertex shader
86 var cBufferId = gl.createBuffer();
87 gl.bindBuffer(gl.ARRAY_BUFFER, cBufferId);
88 gl.bufferData(gl.ARRAY_BUFFER, flatten(colors), gl.STATIC_DRAW);
89
90 var vColor = gl.getAttribLocation(program, "vColor");
91 gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);
92 gl.enableVertexAttribArray(vColor);
93
94 //var fColor = gl.getUniformLocation(program, "fColor");
95 //gl.uniform4f(fColor, 1.0, 0.0, 0.0, 1.0);
96
97 var locPointSize = gl.getUniformLocation(program, "pointSize");
98 gl.uniform1f(locPointSize, 5.0);
99
```

Ln 112, Col 27 Spaces: 4 UTF-8 CRLF JavaScript





연습 문제 (5)

- 세 점을 클릭하여 삼각형들을 그리시오.
- [선택 사항] - 가산점
 - Red / Green / Blue를 슬라이더 바 (Range)로 삼각형의 색상을 입력 받으시오.

