

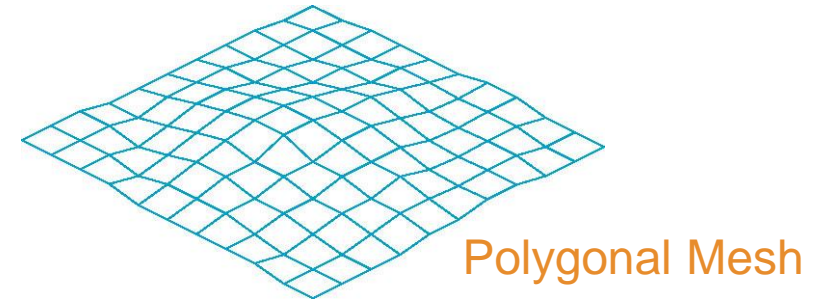
Shading

11TH WEEK, 2022



Polygonal Shading

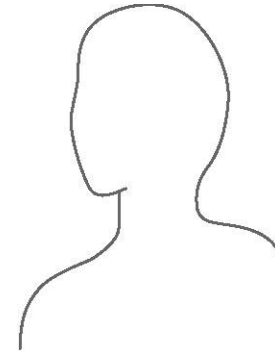
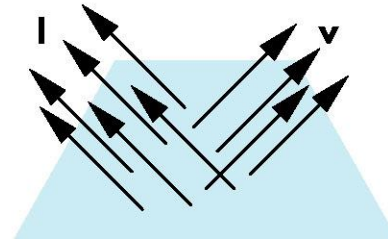
- Shading calculations are done for each vertex
 - Vertex colors become vertex shades



- In OpenGL, vertex shades are interpolated across the polygon by default
 - `glShadeModel(GL_SMOOTH);`
- If we use `glShadeModel(GL_FLAT);` the color at the first vertex will determine the shade of the whole polygon

Flat Shading

- Constant shading – polygons have a single normal
 - Flat polygon
 - n : constant
 - Directional light source
 - l : constant
 - Distant viewer
 - v : constant

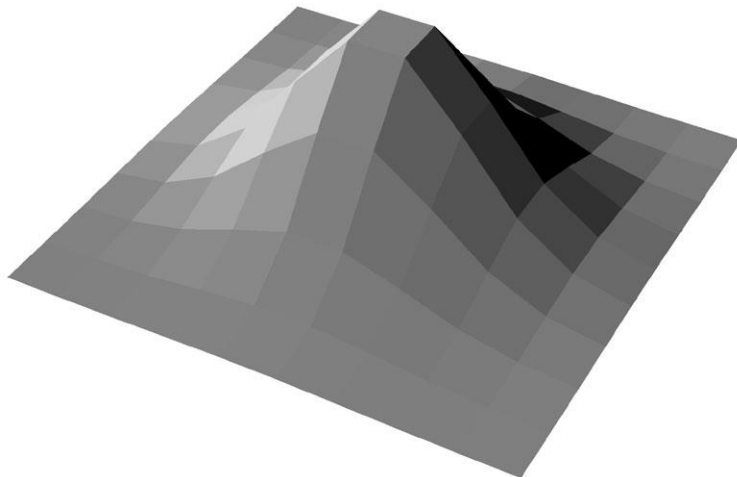
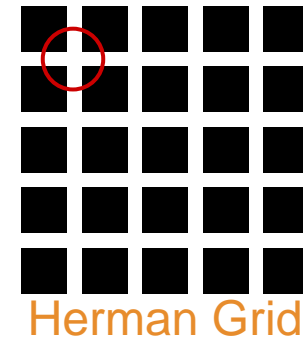


- One shading calculation for each polygon
- Consider model of sphere

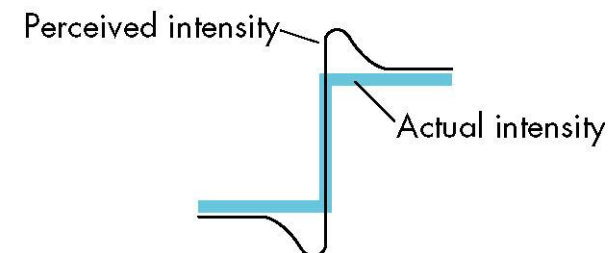


Flat Shading

- Disappointing for a smooth surface
 - Lateral inhibition
 - Human visual system has a remarkable sensitivity
 - Mach bands
 - Perceive the increases in brightness along the edges



Flat shading of polygonal mesh

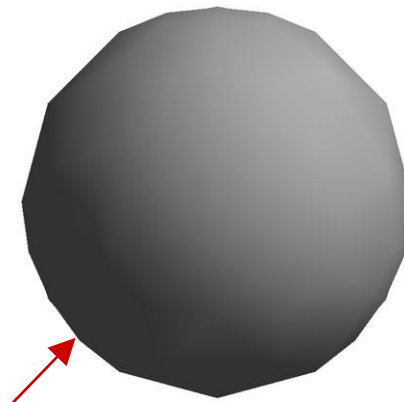
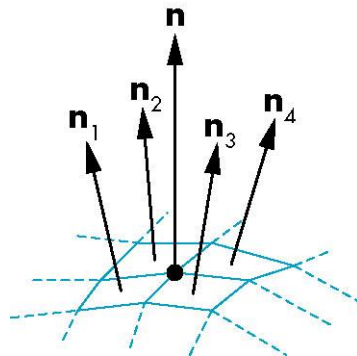


Perceived and actual intensities at an edge

Smooth Shading

- Gouraud shading – we can set a new normal at each vertex
- One shading calculation for each vertex
 - Bilinear interpolation of colors
- Defining vertex normal as the average of the normals around a mesh vertex

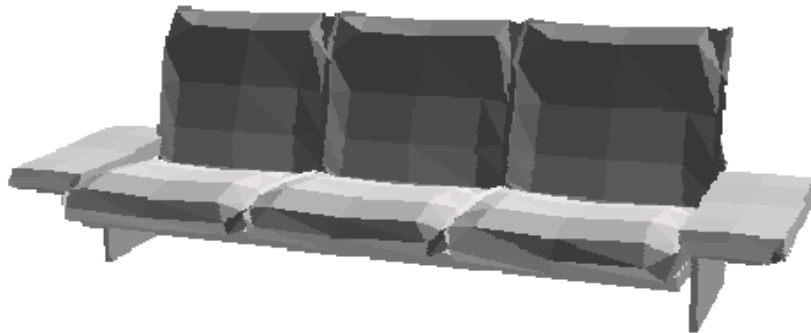
$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$



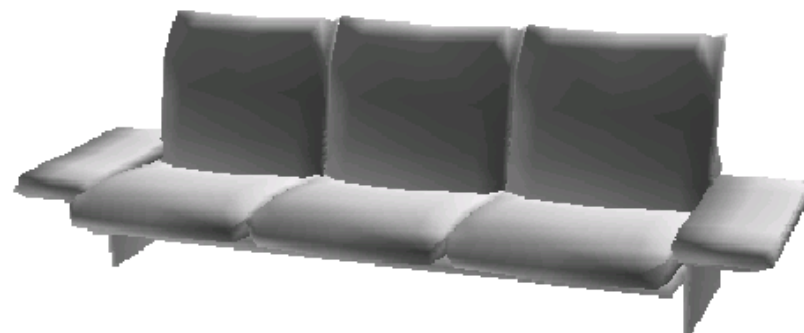
Note silhouette edge

Smooth Shading

- Even the smoothness introduced by Gouraud shading may not prevent the appearance of Mach bands
- If a polygonal mesh is too coarse to capture illumination effects in polygon interiors?



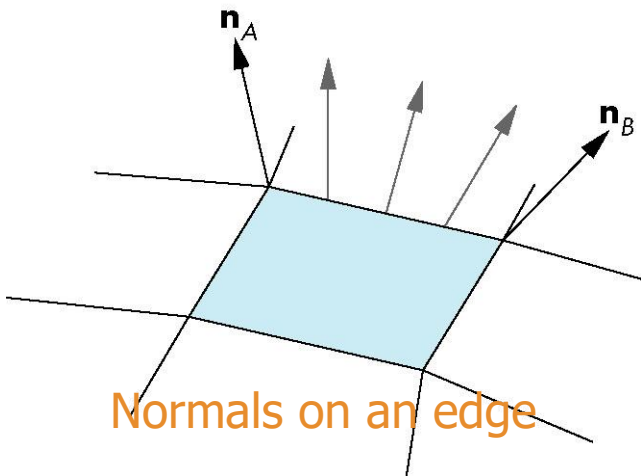
Flat Shading



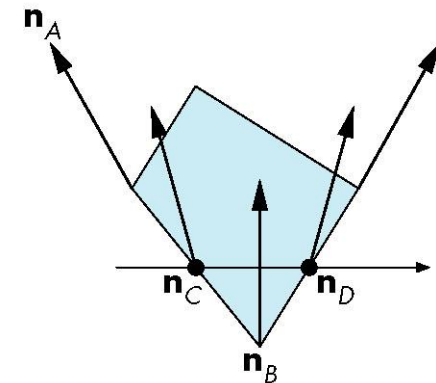
Smooth Shading

Phong Shading

- Interpolating normals across each polygon instead of interpolating vertex intensities
- One shading calculation for each pixel
 - off-line
- Computing vertex normal at each point



$$\mathbf{n}(\alpha) = (1 - \alpha)\mathbf{n}_A + \alpha\mathbf{n}_B$$
$$\mathbf{n}(\alpha, \beta) = (1 - \beta)\mathbf{n}_C + \beta\mathbf{n}_D$$



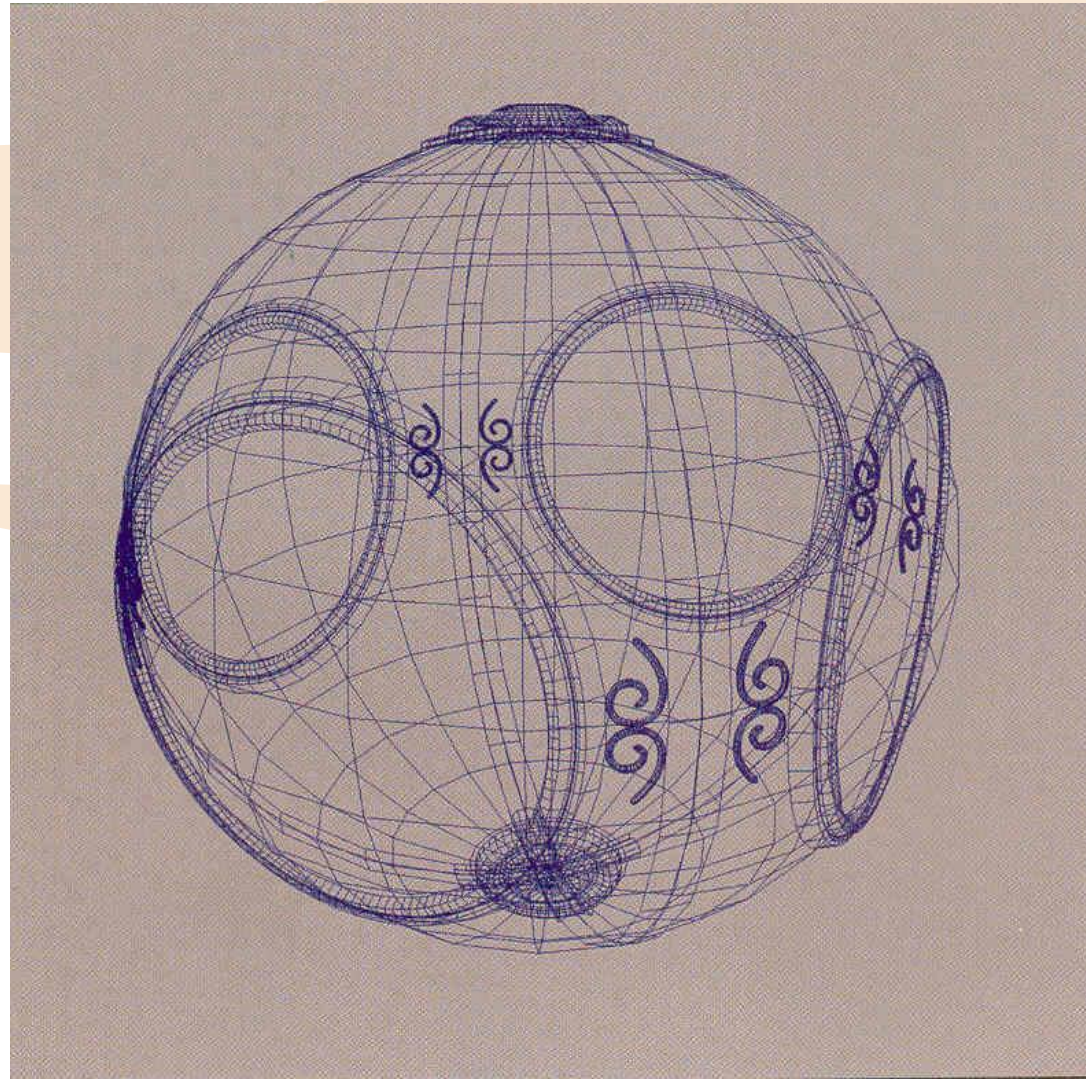
Interpolation of normals

Gouraud vs. Phong Shading

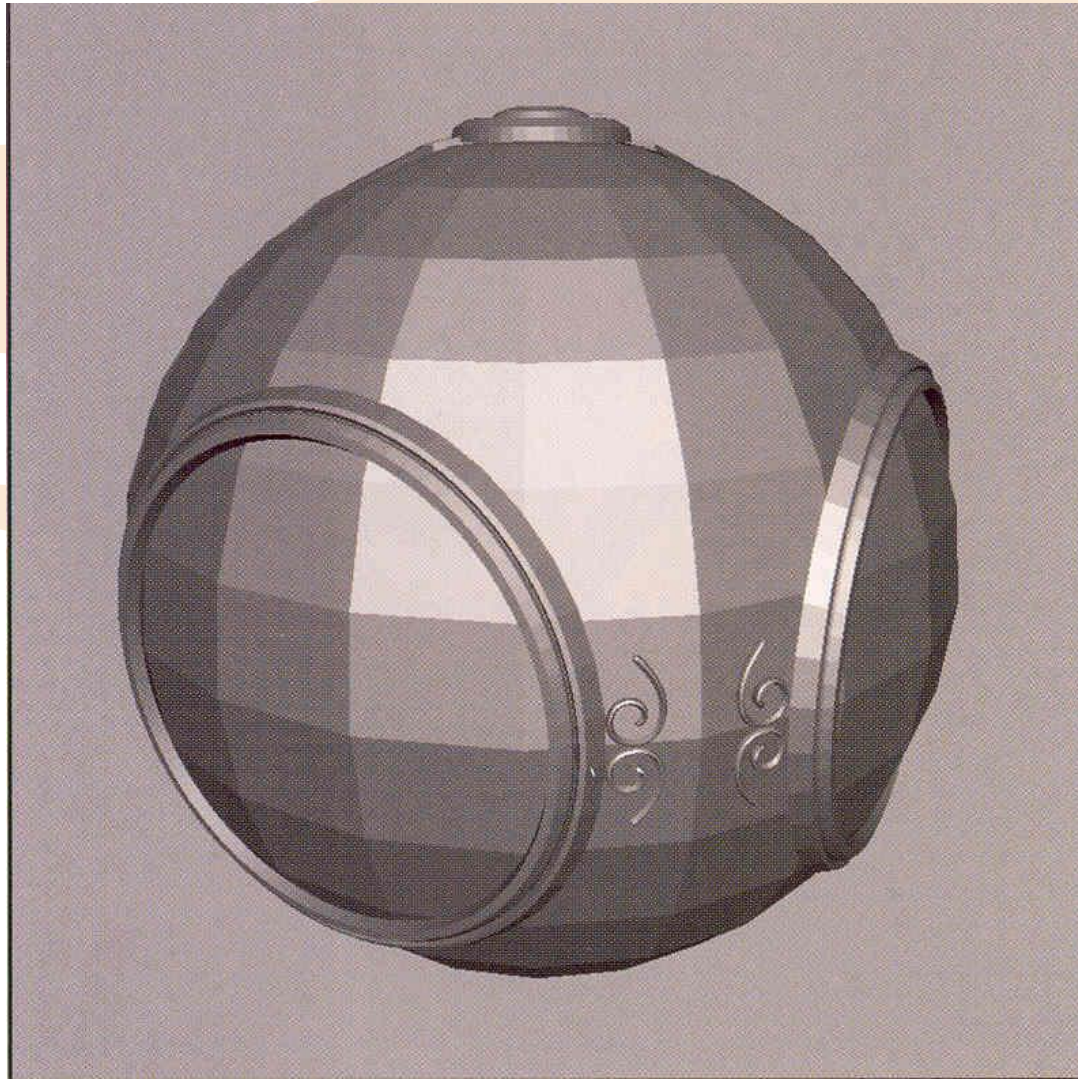
- Gouraud shading
 - Find average normal at each vertex (vertex normals)
 - Apply modified Phong model at each vertex
 - Interpolate vertex shades across each polygon
- Phong shading
 - Find vertex normals
 - Interpolate vertex normals across edges
 - Interpolate edge normals across polygon
 - Apply modified Phong model at each fragment

Comparison

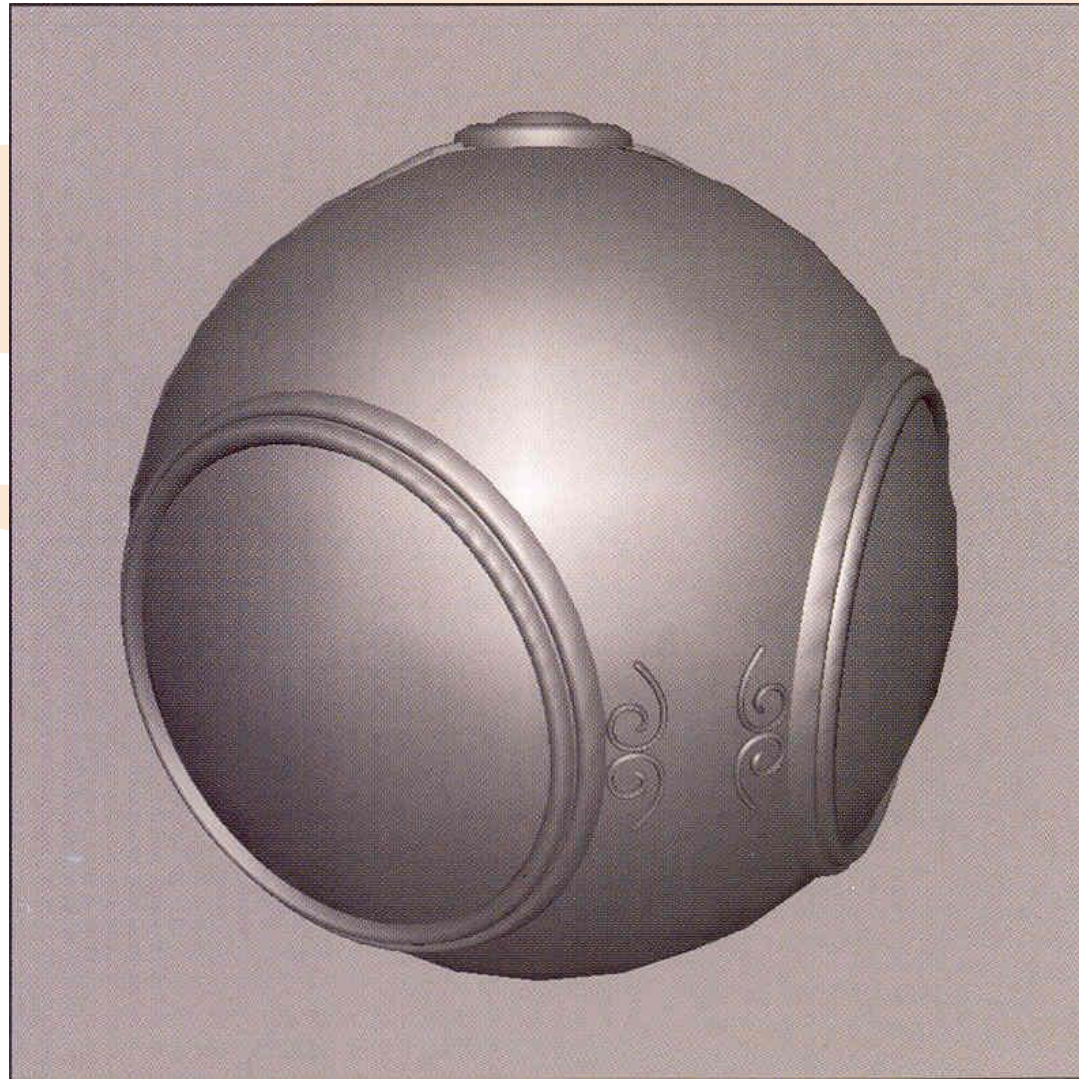
- If polygon mesh approximate surfaces with a high curvatures, Phong shading may look smooth while Gouraud shading may show edges
- Phong shading requires much more work than Gouraud shading
 - Until recently not available in real time systems
 - Now can be done using fragment shaders
- Both need data structures to represent meshes so we can obtain vertex normals



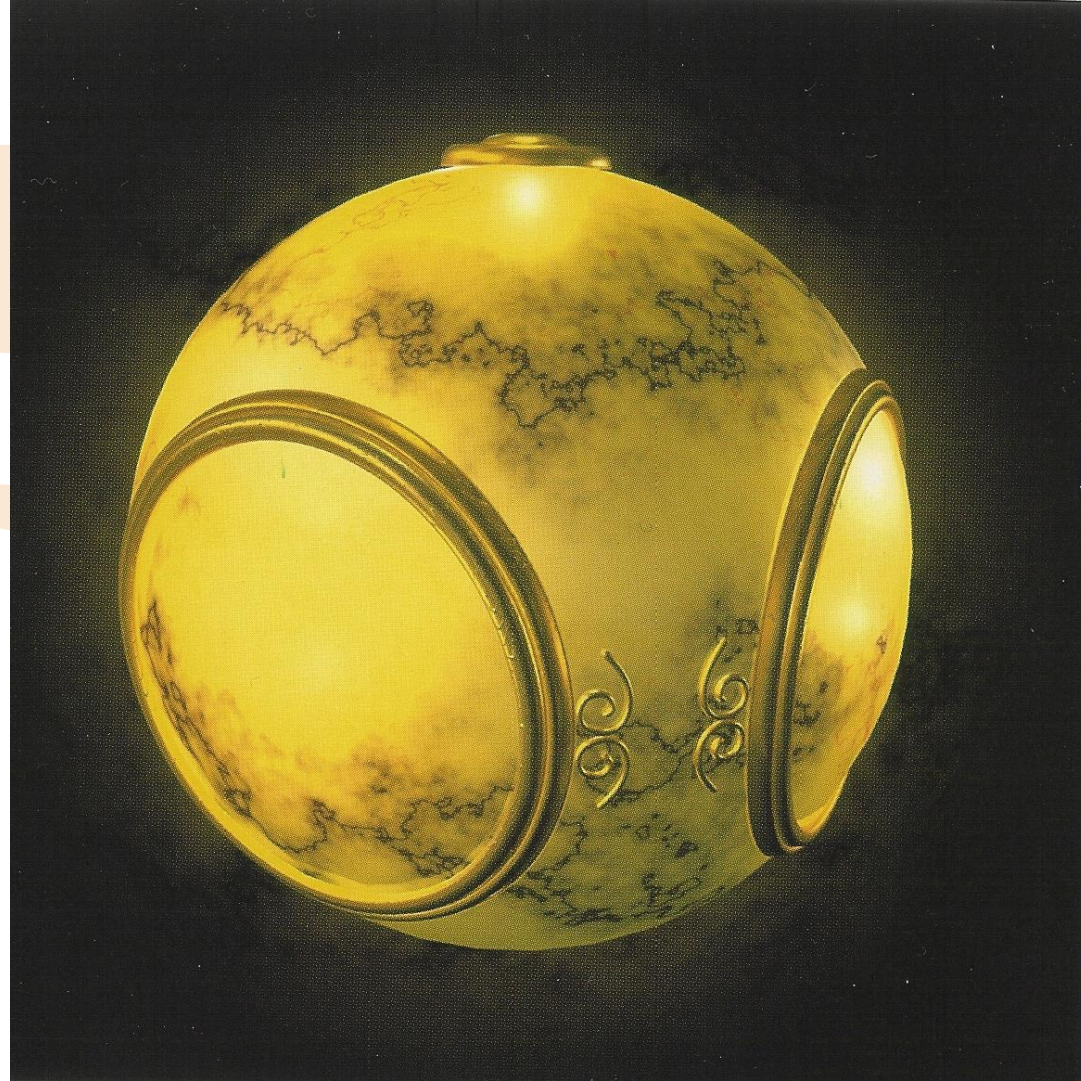
Wireframe



Flat Shading



Gouraud Shading



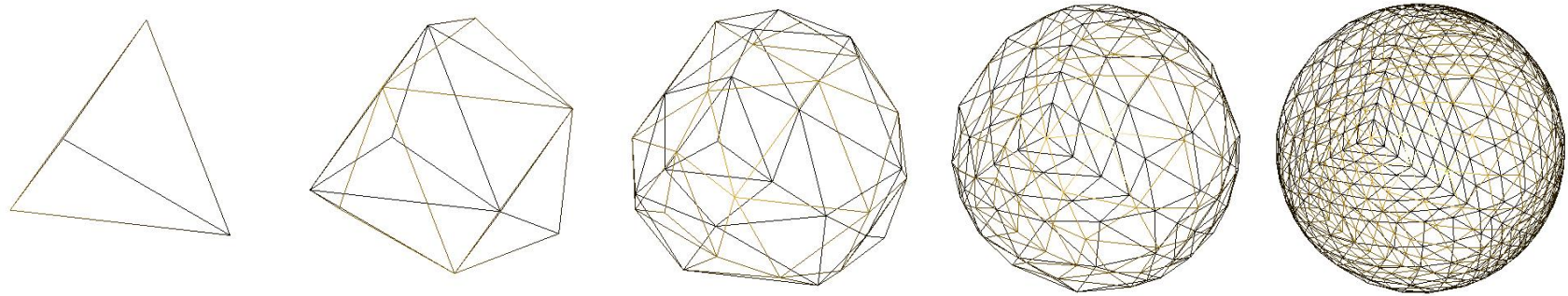
Bump Mapping



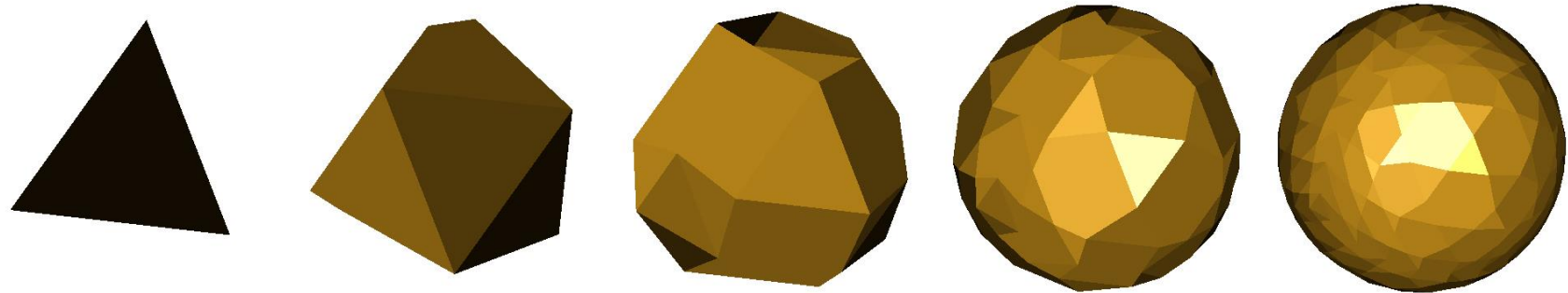
Environmental Mapping

Sphere by Recursive Subdivision

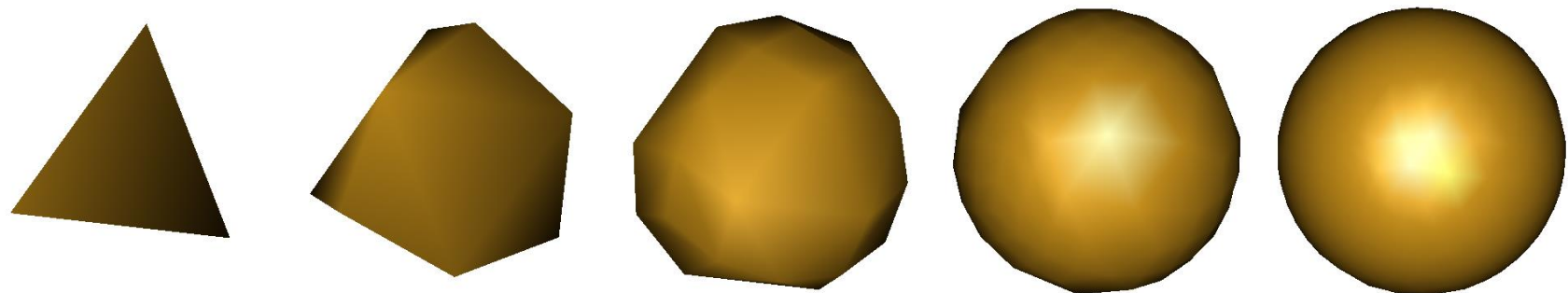
Wireframe

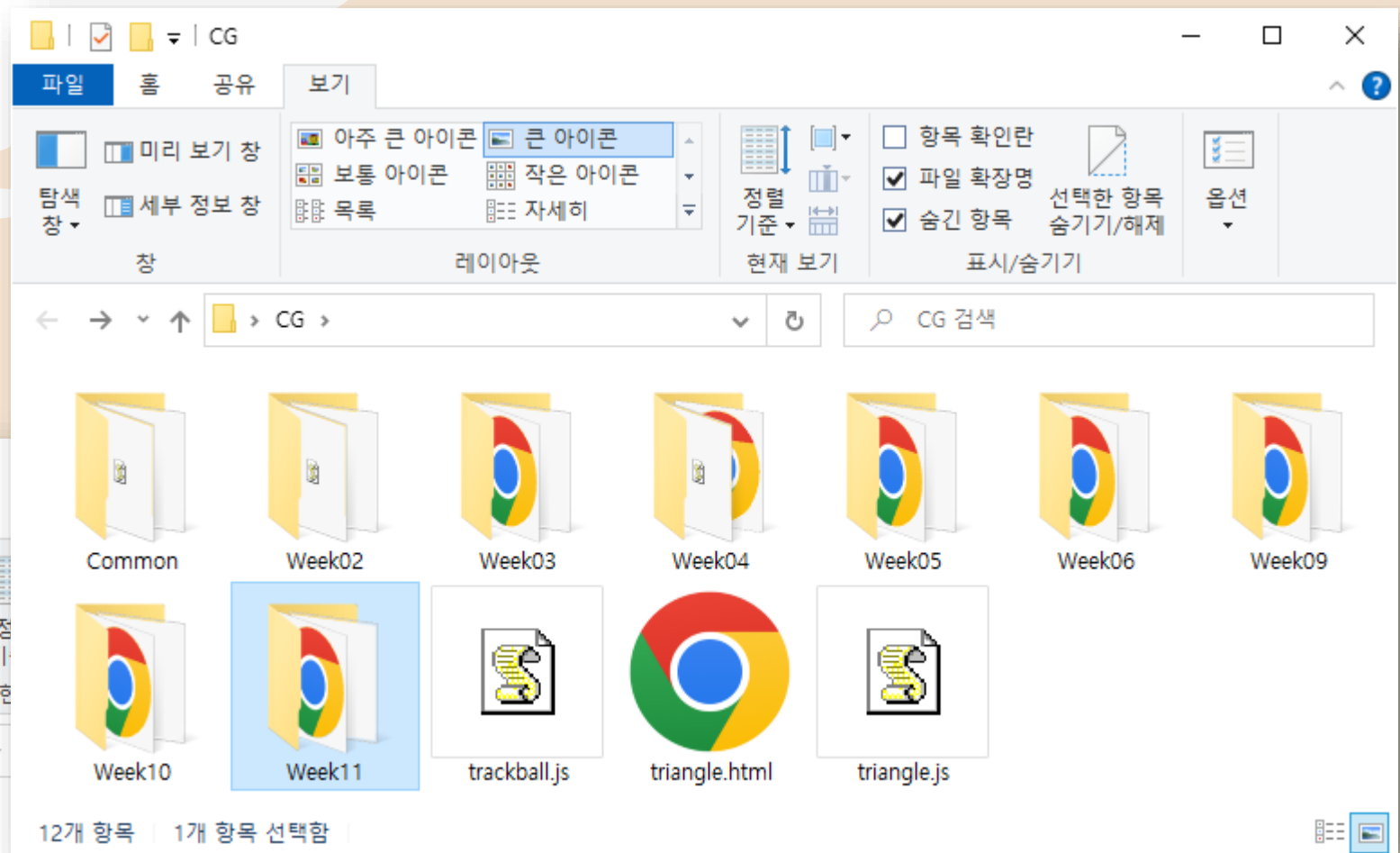
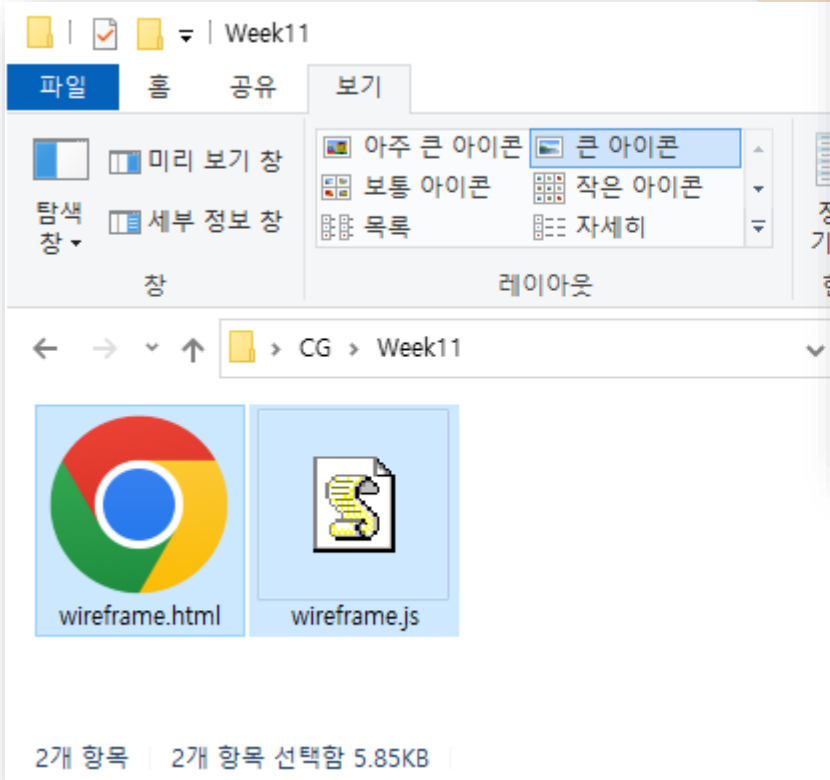


Flat Shading

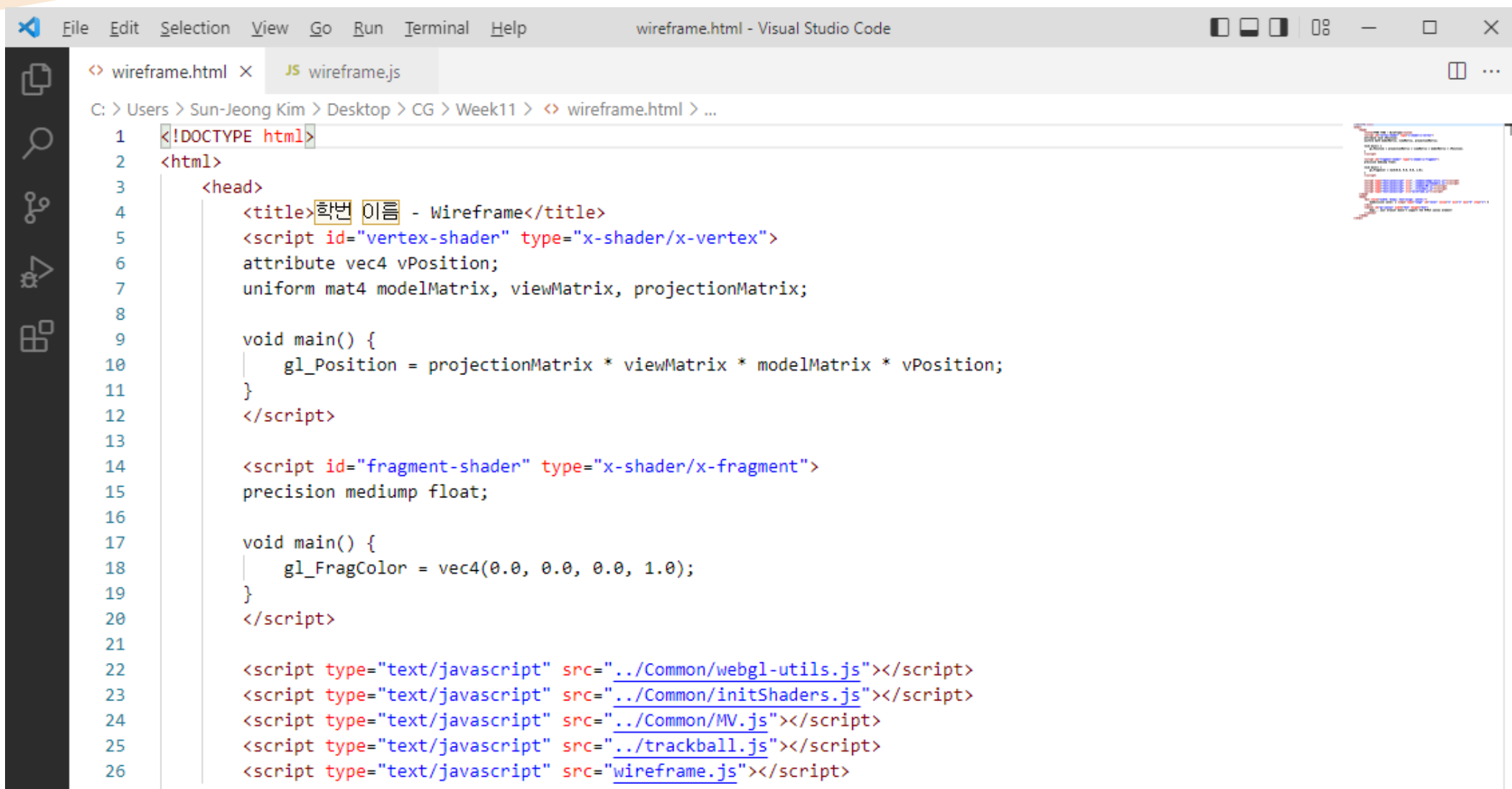


Gouraud Shading





Wireframe



```
<?DOCTYPE html>
<html>
  <head>
    <title>학번 이름 - Wireframe</title>
    <script id="vertex-shader" type="x-shader/x-vertex">
      attribute vec4 vPosition;
      uniform mat4 modelMatrix, viewMatrix, projectionMatrix;

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

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

      void main() {
        gl_FragColor = vec4(0.0, 0.0, 0.0, 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="wireframe.js"></script>
```



<> wireframe.html X JS wireframe.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > <> wireframe.html > ...

```
5      <script id="vertex-shader" type="x-shader/x-vertex">
6      attribute vec4 vPosition;
7      uniform mat4 modelMatrix, viewMatrix, projectionMatrix;
8
9      void main() {
10         gl_Position = projectionMatrix * viewMatrix * modelMatrix * 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(0.0, 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="../trackball.js"></script>
26     <script type="text/javascript" src="wireframe.js"></script>
27 </head>
28 <body>
29     <div style="width: 512px; text-align: center;">
30         Subdivision Level: 1 <input type="range" id="level" value="1" min="1" max="5" step="1"> 5
31     </div>
32     <canvas id="gl-canvas" width="512" height="512">
33         Oops... your browser doesn't support the HTML5 canvas element!
34     </canvas>
35 </body>
36 </html>
```



wireframe.html JS wireframe.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS wireframe.js > ...

```
1  var gl;
2  var points = [];
3
4  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
5
6  //var modelMatrix;
7  var modelMatrixLoc;
8
9  window.onload = function init()
10 {
11     var canvas = document.getElementById("gl-canvas");
12
13     gl = WebGLUtils.setupWebGL(canvas);
14     if( !gl ) {
15         alert("WebGL isn't available!");
16     }
17
18     generateTetrahedron(1);
19
20     // virtual trackball
21     var trball = trackball(canvas.width, canvas.height);
22     var mouseDown = false;
23
24     canvas.addEventListener("mousedown", function (event) {
25         trball.start(event.clientX, event.clientY);
26
27         mouseDown = true;
28     });
29
30     canvas.addEventListener("mouseup", function (event) {
31         mouseDown = false;
32     });
33
34     canvas.addEventListener("mousemove", function (event) {
35         if (mouseDown) {
```



wireframe.html JS wireframe.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS wireframe.js > ...

```
36         trball.end(event.clientX, event.clientY);
37
38         trballMatrix = mat4(trball.rotationMatrix);
39     }
40 });
41
42 // Configure WebGL
43 gl.viewport(0, 0, canvas.width, canvas.height);
44 gl.clearColor(0.9, 0.9, 0.9, 1.0);
45
46 // Enable hidden-surface removal
47 gl.enable(gl.DEPTH_TEST);
48
49 // Load shaders and initialize attribute buffers
50 var program = initShaders(gl, "vertex-shader", "fragment-shader");
51 gl.useProgram(program);
52
53 // Load the data into the GPU
54 var bufferId = gl.createBuffer();
55 gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
56 gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
57
58 // Associate our shader variables with our data buffer
59 var vPosition = gl.getAttribLocation(program, "vPosition");
60 gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
61 gl.enableVertexAttribArray(vPosition);
62
63 //modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
64 modelMatrixLoc = gl.getUniformLocation(program, "modelMatrix");
65 //gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));
66
67 var viewMatrix = lookAt(vec3(0, 0, 1), vec3(0, 0, 0), vec3(0, 1, 0));
68 var viewMatrixLoc = gl.getUniformLocation(program, "viewMatrix");
69 gl.uniformMatrix4fv(viewMatrixLoc, false, flatten(viewMatrix));
70
```



wireframe.html JS wireframe.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS wireframe.js > ...

```
71 // 3D orthographic viewing
72 var viewLength = 1.5;
73 var projectionMatrix;
74 if (canvas.width > canvas.height) {
75     var aspect = viewLength * canvas.width / canvas.height;
76     projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
77 }
78 else {
79     var aspect = viewLength * canvas.height / canvas.width;
80     projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
81 }
82 /*
83 // 3D perspective viewing
84 var aspect = canvas.width / canvas.height;
85 var projectionMatrix = perspective(90, aspect, 0.1, 1000);
86 */
87 var projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
88 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
89
90 // Event listeners for buttons
91 document.getElementById("level").onchange = function(event) {
92     var level = event.target.value;
93
94     points = [];
95     generateTetrahedron(level);
96
97     //gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
98     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
99
100     render();
101 };
102
103 render();
104 };
105
```

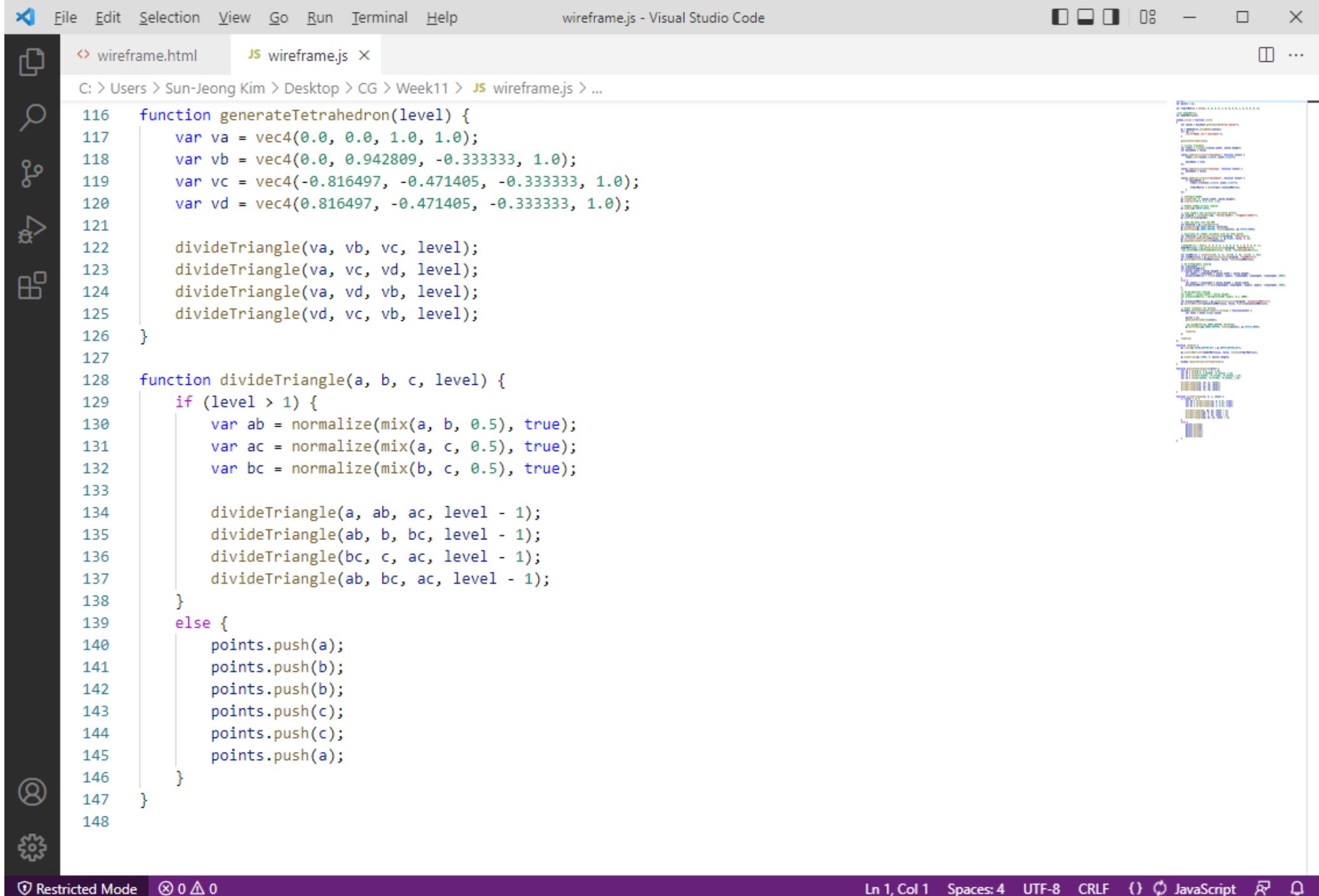


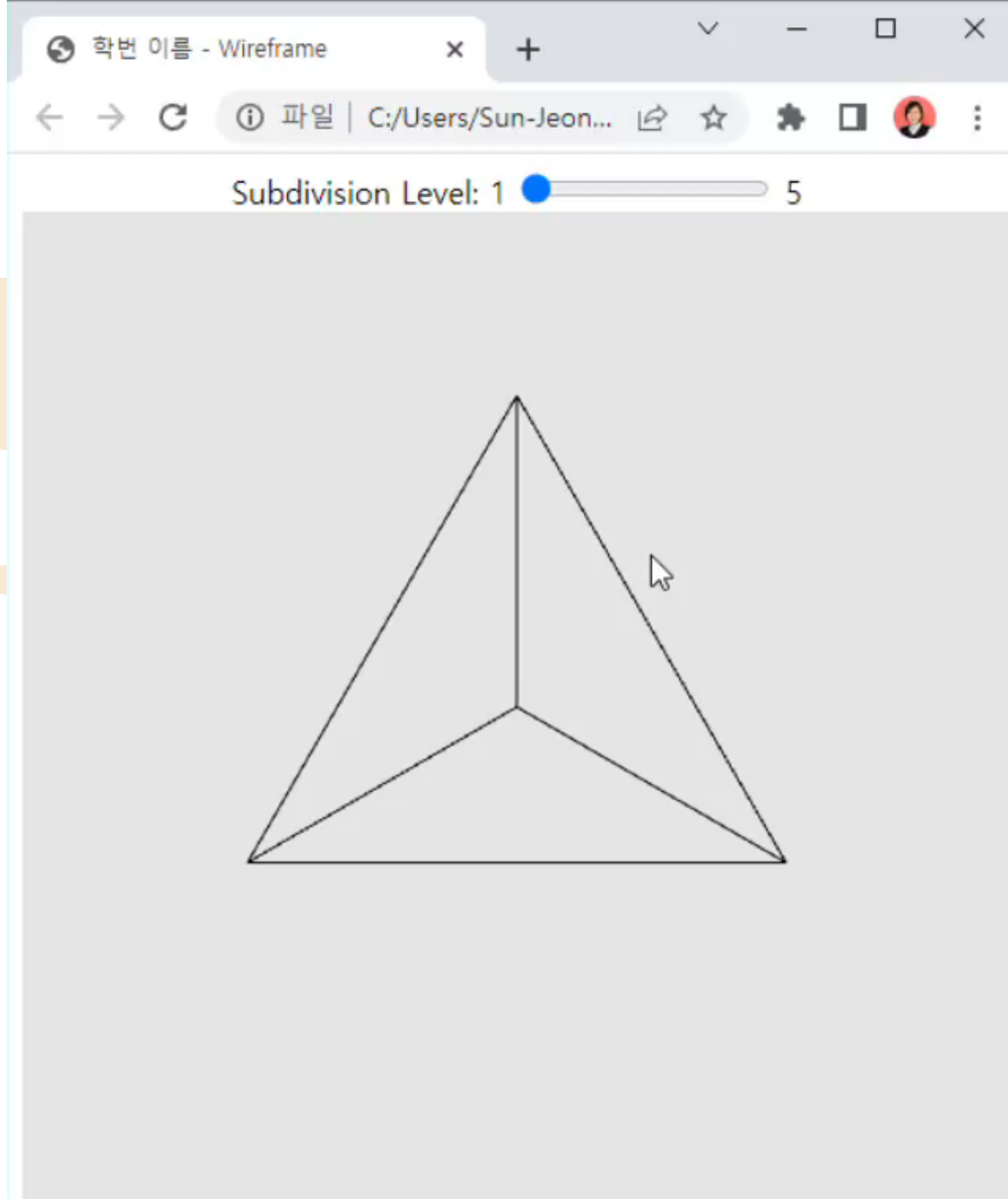
wireframe.html JS wireframe.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS wireframe.js > ...

```
106 function render() {
107     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
108
109     gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(trballMatrix));
110
111     gl.drawArrays(gl.LINES, 0, points.length);
112
113     window.requestAnimationFrame(render);
114 }
115
116 function generateTetrahedron(level) {
117     var va = vec4(0.0, 0.0, 1.0, 1.0);
118     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
119     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
120     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
121
122     divideTriangle(va, vb, vc, level);
123     divideTriangle(va, vc, vd, level);
124     divideTriangle(va, vd, vb, level);
125     divideTriangle(vd, vc, vb, level);
126 }
127
128 function divideTriangle(a, b, c, level) {
129     if (level > 1) {
130         var ab = normalize(mix(a, b, 0.5), true);
131         var ac = normalize(mix(a, c, 0.5), true);
132         var bc = normalize(mix(b, c, 0.5), true);
133
134         divideTriangle(a, ab, ac, level - 1);
135         divideTriangle(ab, b, bc, level - 1);
136         divideTriangle(bc, c, ac, level - 1);
137         divideTriangle(ab, bc, ac, level - 1);
138     }
139     else {
140         points.push(a);
```

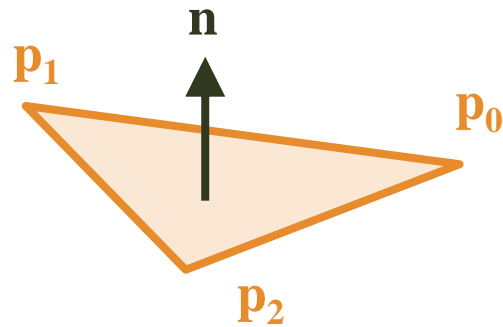






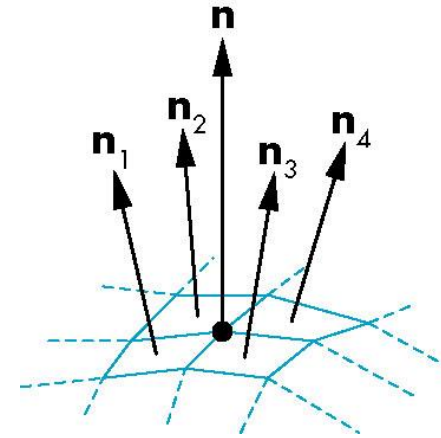
Flat / Gouraud Shading

- Flat shading

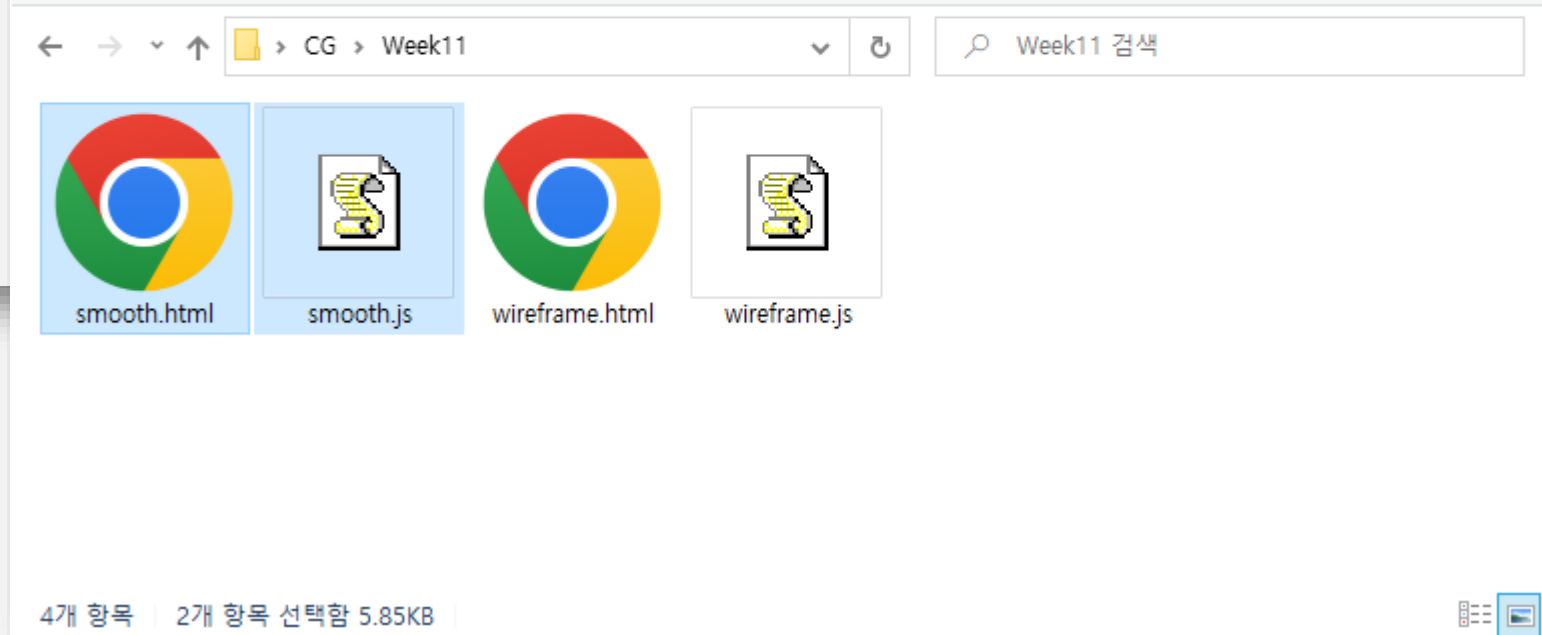
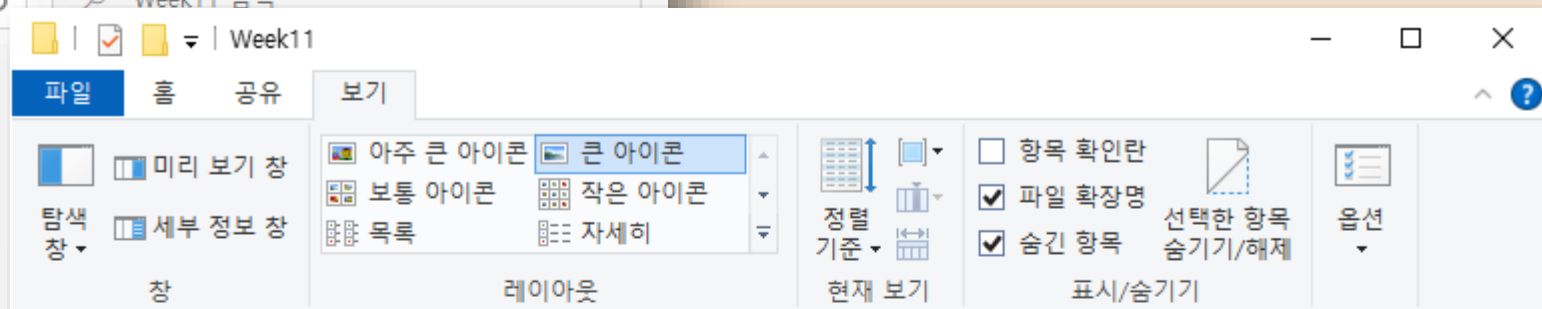
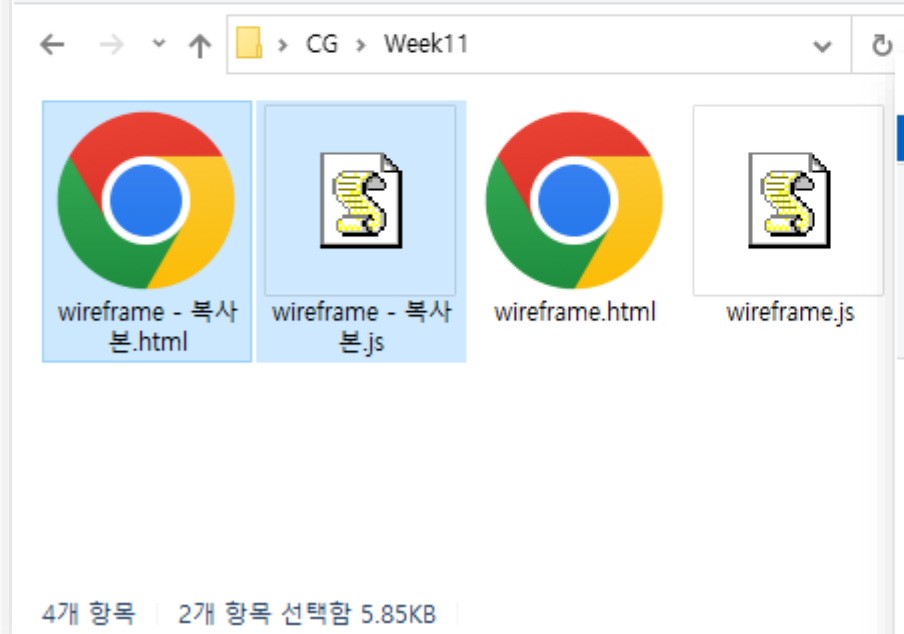
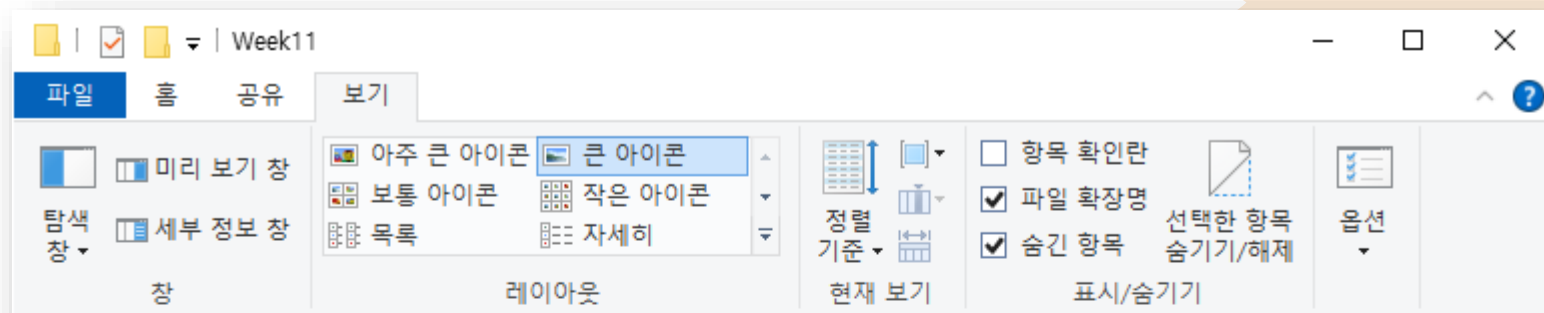


$$\mathbf{n} = (\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0)$$

- Gouraud Shading



$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$



```

<> wireframe.html  JS wireframe.js  <> smooth.html X  JS smooth.js
C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > <> smooth.html > html > body > div > button#change
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>학번 이름 - Flat/Gouraud Shading</title>
5          <script id="vertex-shader" type="x-shader/x-vertex">
6              attribute vec4 vPosition;
7              attribute vec4 vNormal;
8              uniform mat4 modelMatrix, viewMatrix, projectionMatrix;
9              varying vec4 fColor;
10
11             uniform vec3 eyePos;
12             uniform vec4 lightSrc, ambientProduct, diffuseProduct, specularProduct;
13             uniform float shininess;
14
15             void main() {
16                 gl_Position = projectionMatrix * viewMatrix * modelMatrix * vPosition;
17
18                 vec3 N = normalize(mat3(modelMatrix) * vNormal.xyz);
19                 vec3 L = normalize(lightSrc.xyz);
20                 float kd = max(dot(L, N), 0.0);
21                 vec4 diffuse = kd * diffuseProduct;
22
23                 vec3 worldPos = (modelMatrix * vPosition).xyz;
24                 vec3 V = normalize(eyePos - worldPos);
25                 vec3 H = normalize(L + V);
26                 float ks = pow(max(dot(N, H), 0.0), shininess);
27                 vec4 specular = ks * specularProduct;
28
29                 fColor = ambientProduct + diffuse + specular;
30                 fColor.a = 1.0;
31             }
32         </script>
33
34         <script id="fragment-shader" type="x-shader/x-fragment">
35             precision mediump float;

```

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

wireframe.html JS wireframe.js smooth.html X JS smooth.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > smooth.html > html > body > div > button#change

```
33
34     <script id="fragment-shader" type="x-shader/x-fragment">
35     precision mediump float;
36     varying vec4 fColor;
37
38     void main() {
39         gl_FragColor = fColor;
40     }
41     </script>
42
43     <script type="text/javascript" src="../../Common/webgl-utils.js"></script>
44     <script type="text/javascript" src="../../Common/initShaders.js"></script>
45     <script type="text/javascript" src="../../Common/MV.js"></script>
46     <script type="text/javascript" src="../../trackball.js"></script>
47     <script type="text/javascript" src="smooth.js"></script>
48 </head>
49 <body>
50     <div style="width: 512px; text-align: center;">
51         <input type="radio" id="flat" name="shading" checked> Flat Shading
52         <input type="radio" id="smooth" name="shading"> Gouraud Shading
53         <button id="change">Change</button>
54     </div>
55     <div style="width: 512px; text-align: center;">
56         Subdivision Level: 1 <input type="range" id="level" value="1" min="1" max="5" step="1"> 5
57     </div>
58     <canvas id="gl-canvas" width="512" height="512">
59         Oops... your browser doesn't support the HTML5 canvas element!
60     </canvas>
61 </body>
62 </html>
```

Ln 53, Col 39 Spaces: 4 UTF-8 CRLF HTML

Restricted Mode 0 0

```
File Edit Selection View Go Run Terminal Help smooth.js - Visual Studio Code
wireframe.html JS wireframe.js smooth.html JS smooth.js X
C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS smooth.js > divideTriangle
1  var gl;
2  var points = [];
3  var normals = [];
4  var fNormals = []; // for flat shading
5
6  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
7
8  //var modelMatrix;
9  var modelMatrixLoc;
10
11  var flatShading = true;
12
13  window.onload = function init()
14  {
15      var canvas = document.getElementById("gl-canvas");
16
17      gl = WebGLUtils.setupWebGL(canvas);
18      if( !gl ) {
19          alert("WebGL isn't available!");
20      }
21
22      generateTetrahedron(1);
23
24      // virtual trackball
25      var trball = trackball(canvas.width, canvas.height);
26      var mouseDown = false;
27
28      canvas.addEventListener("mousedown", function (event) {
29          trball.start(event.clientX, event.clientY);
30
31          mouseDown = true;
32      });
33
34      canvas.addEventListener("mouseup", function (event) {
35          mouseDown = false;
```

FileEditSelectionViewGoRunTerminalHelp

smooth.js - Visual Studio Code

wireframe.htmlJS wireframe.jssmooth.htmlJS smooth.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS smooth.js > divideTriangle

53// Load shaders and initialize attribute buffers

54var program = initShaders(gl, "vertex-shader", "fragment-shader");

55gl.useProgram(program);

56

57// Load the data into the GPU

58var bufferId = gl.createBuffer();

59gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);

60gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);

61

62// Associate our shader variables with our data buffer

63var vPosition = gl.getAttribLocation(program, "vPosition");

64gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);

65gl.enableVertexAttribArray(vPosition);

66

67// Create a buffer object, initialize it, and associate it with

68// the associated attribute variable in our vertex shader

69var nBufferId = gl.createBuffer();

70gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);

71gl.bufferData(gl.ARRAY_BUFFER, flatten(fNormals), gl.STATIC_DRAW);

72

73var vNormal = gl.getAttribLocation(program, "vNormal");

74gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);

75gl.enableVertexAttribArray(vNormal);

76

77//modelMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

78modelMatrixLoc = gl.getUniformLocation(program, "modelMatrix");

79//gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(modelMatrix));

80

81var viewMatrix = lookAt(vec3(0, 0, 1), vec3(0, 0, 0), vec3(0, 1, 0));

82var viewMatrixLoc = gl.getUniformLocation(program, "viewMatrix");

83gl.uniformMatrix4fv(viewMatrixLoc, false, flatten(viewMatrix));

84

85// 3D orthographic viewing

86var viewLength = 1.5;

87var projectionMatrix;

30

Restricted Mode

0/0

Ln 212, Col 31Spaces: 4UTF-8CRLFJavaScript

```
File Edit Selection View Go Run Terminal Help smooth.js - Visual Studio Code
wireframe.html JS wireframe.js smooth.html JS smooth.js X
C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS smooth.js > divideTriangle

103
104 // Event listeners for buttons
105 document.getElementById("change").onclick = function () {
106     if (document.getElementById("flat").checked)
107         flatShading = true;
108     else
109         flatShading = false;
110
111     gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
112     if (flatShading)
113         gl.bufferData(gl.ARRAY_BUFFER, flatten(fNormals), gl.STATIC_DRAW);
114     else
115         gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
116
117     render();
118 };
119 document.getElementById("level").onchange = function(event) {
120     var level = event.target.value;
121
122     points = [];
123     normals = [];
124     fNormals = [];
125     generateTetrahedron(level);
126
127     gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
128     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
129     gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
130     if (flatShading)
131         gl.bufferData(gl.ARRAY_BUFFER, flatten(fNormals), gl.STATIC_DRAW);
132     else
133         gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
134
135     render();
136 };
137
```


File Edit Selection View Go Run Terminal Help

smooth.js - Visual Studio Code

wireframe.html JS wireframe.js smooth.html JS smooth.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS smooth.js > divideTriangle

```
137
138   setLighting(program);
139
140   render();
141 };
142
143 function setLighting(program) {
144     var lightSrc = [0.0, 0.0, 1.0, 0.0];
145     var lightAmbient = [0.2, 0.2, 0.2, 1.0];
146     var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
147     var lightSpecular = [1.0, 1.0, 1.0, 1.0];
148
149     var matAmbient = [1.0, 0.0, 1.0, 1.0];
150     var matDiffuse = [1.0, 0.8, 0.0, 1.0];
151     var matSpecular = [1.0, 1.0, 1.0, 1.0];
152
153     var ambientProduct = mult(lightAmbient, matAmbient);
154     var diffuseProduct = mult(lightDiffuse, matDiffuse);
155     var specularProduct = mult(lightSpecular, matSpecular);
156
157     gl.uniform4fv(gl.getUniformLocation(program, "lightSrc"), lightSrc);
158     gl.uniform4fv(gl.getUniformLocation(program, "ambientProduct"), ambientProduct);
159     gl.uniform4fv(gl.getUniformLocation(program, "diffuseProduct"), diffuseProduct);
160     gl.uniform4fv(gl.getUniformLocation(program, "specularProduct"), specularProduct);
161     gl.uniform1f(gl.getUniformLocation(program, "shininess"), 100.0);
162     gl.uniform3f(gl.getUniformLocation(program, "eyePos"), 0, 0, 1);
163 }
164
165 function render() {
166     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
167
168     gl.uniformMatrix4fv(modelMatrixLoc, false, flatten(trballMatrix));
169
170     gl.drawArrays(gl.TRIANGLES, 0, points.length);
171
```

Ln 212, Col 31 Spaces: 4 UTF-8 CRLF {} JavaScript

wireframe.html JS wireframe.js smooth.html JS smooth.js X

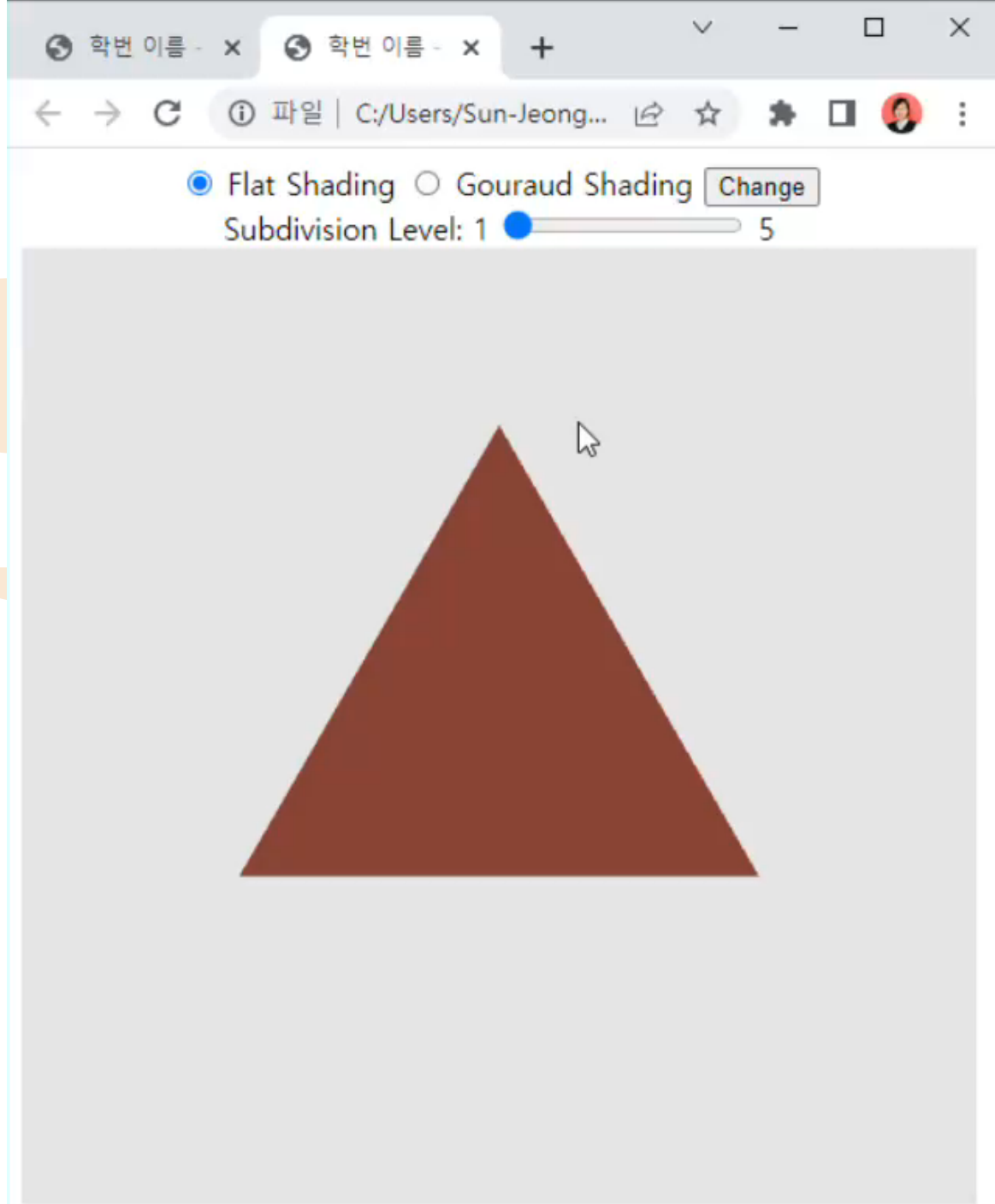
C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS smooth.js > divideTriangle

```

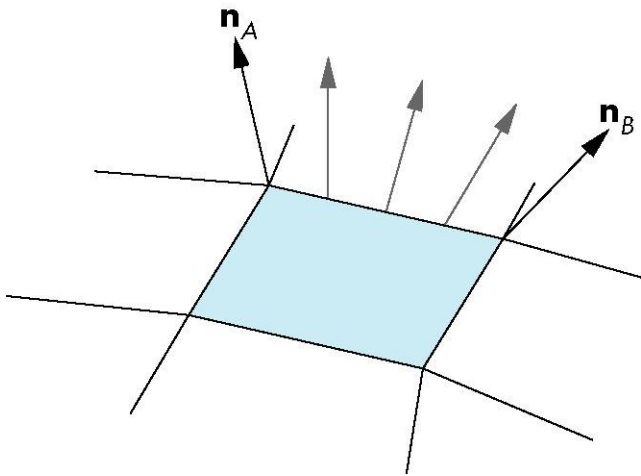
183     divideTriangle(va, vd, vb, level);
184     divideTriangle(vd, vc, vb, level);
185 }
186
187 function divideTriangle(a, b, c, level) {
188     if (level > 1) {
189         var ab = normalize(mix(a, b, 0.5), true);
190         var ac = normalize(mix(a, c, 0.5), true);
191         var bc = normalize(mix(b, c, 0.5), true);
192
193         divideTriangle(a, ab, ac, level - 1);
194         divideTriangle(ab, b, bc, level - 1);
195         divideTriangle(bc, c, ac, level - 1);
196         divideTriangle(ab, bc, ac, level - 1);
197     }
198     else {
199         points.push(a);
200         normals.push(vec4(a[0], a[1], a[2], 0.0));
201         points.push(b);
202         normals.push(vec4(b[0], b[1], b[2], 0.0));
203         points.push(c);
204         normals.push(vec4(c[0], c[1], c[2], 0.0));
205
206         var ab = subtract(b, a);
207         var ac = subtract(c, a);
208         var n = cross(ab, ac);
209         var normal = normalize(vec4(n[0], n[1], n[2], 0.0));
210         fNormals.push(normal);
211         fNormals.push(normal);
212         fNormals.push(normal);
213     }
214 }
215

```



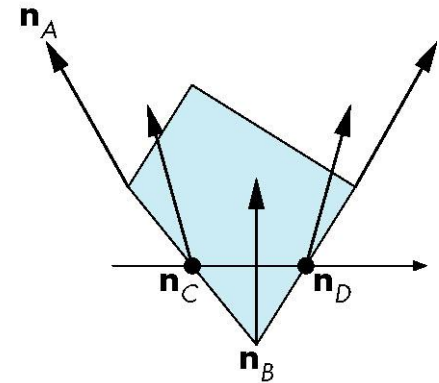


Phong Shading

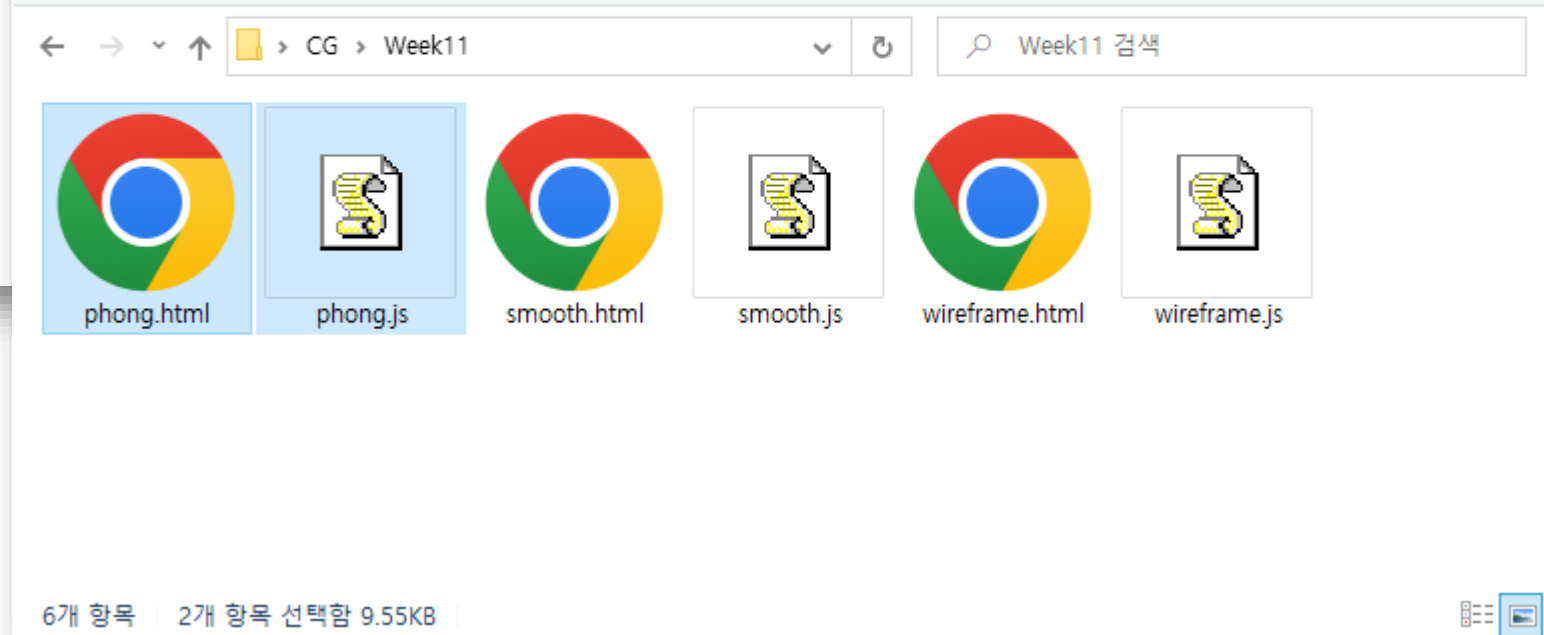
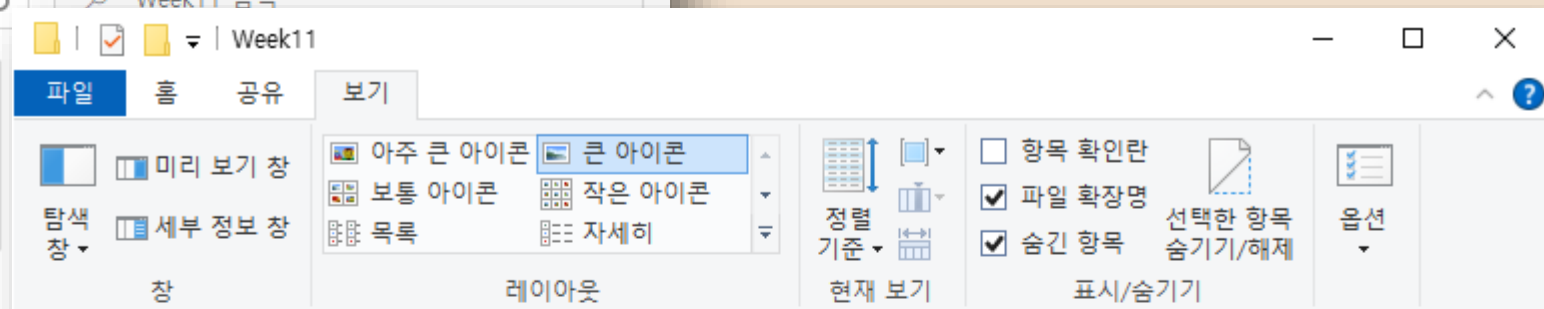
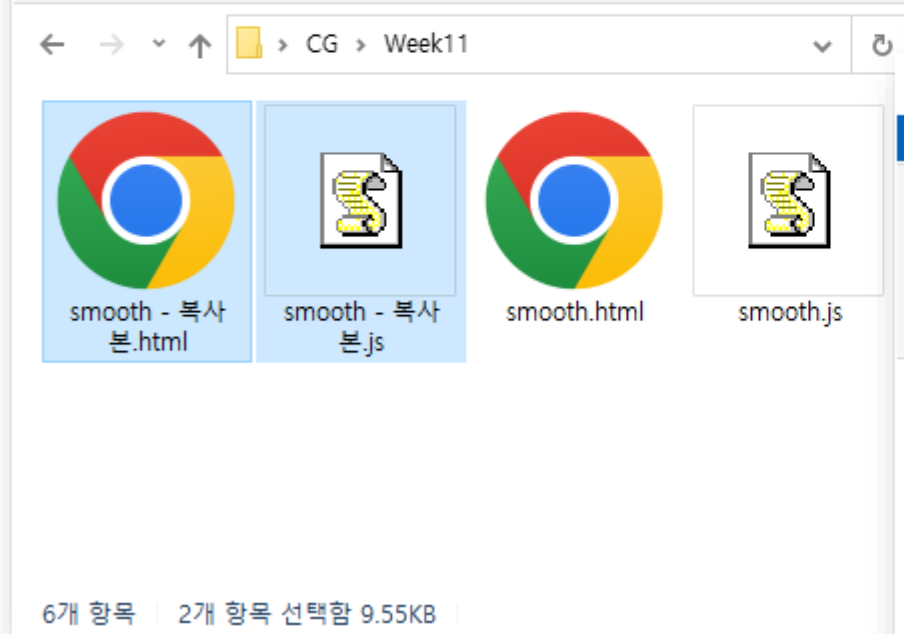
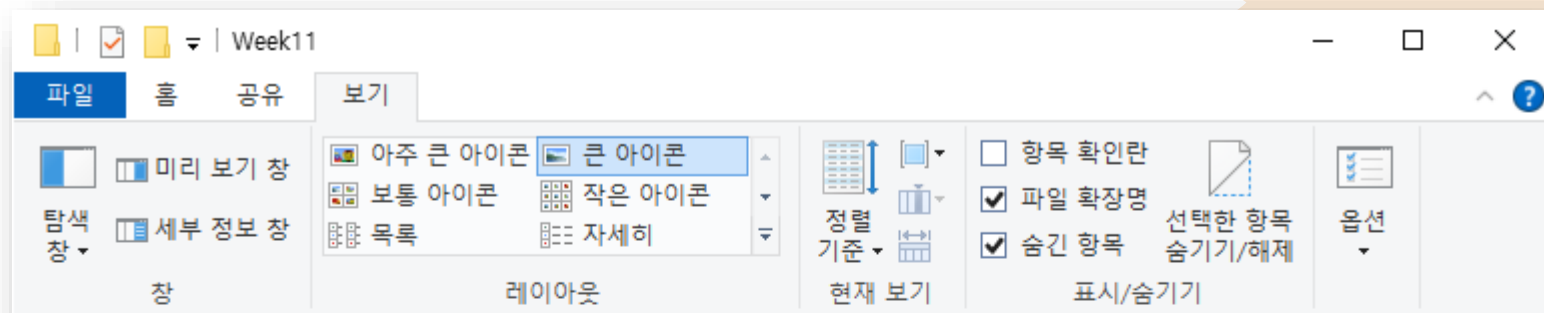


Normals on an edge

$$\mathbf{n}(\alpha) = (1 - \alpha)\mathbf{n}_A + \alpha\mathbf{n}_B$$
$$\mathbf{n}(\alpha, \beta) = (1 - \beta)\mathbf{n}_C + \beta\mathbf{n}_D$$



Interpolation of normals



```

<> wireframe.html JS wireframe.js <> smooth.html JS smooth.js <> phong.html X JS phong.js
C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > <> phong.html > html > head > script
1  <!DOCTYPE html>
2  <html>
3    <head>
4    <title>학번 이름 - Phong Shading</title>
5    <script id="vertex-shader" type="x-shader/x-vertex">
6      attribute vec4 vPosition;
7      attribute vec4 vNormal;
8      uniform mat4 modelMatrix, viewMatrix, projectionMatrix;
9
10     varying vec3 fNormal, fWorldPos;
11
12     void main() {
13       gl_Position = projectionMatrix * viewMatrix * modelMatrix * vPosition;
14
15       fNormal = normalize(mat3(modelMatrix) * vNormal.xyz);
16       fWorldPos = (modelMatrix * vPosition).xyz;
17     }
18   </script>
19
20   <script id="fragment-shader" type="x-shader/x-fragment">
21     precision mediump float;
22     varying vec3 fNormal, fWorldPos;
23
24     uniform vec3 eyePos;
25     uniform vec4 lightSrc, ambientProduct, diffuseProduct, specularProduct;
26     uniform float shininess;
27
28     void main() {
29       vec3 N = normalize(fNormal);
30       vec3 L = normalize(lightSrc.xyz);
31       float kd = max(dot(L, N), 0.0);
32       vec4 diffuse = kd * diffuseProduct;
33
34       vec3 V = normalize(eyePos - fWorldPos);
35       vec3 H = normalize(L + V);

```

File Edit Selection View Go Run Terminal Help

phong.html - Visual Studio Code

wireframe.html JS wireframe.js smooth.html JS smooth.js phong.html X JS phong.js

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > <> phong.html > html > head > script

```
27
28     void main() {
29         vec3 N = normalize(fNormal);
30         vec3 L = normalize(lightSrc.xyz);
31         float kd = max(dot(L, N), 0.0);
32         vec4 diffuse = kd * diffuseProduct;
33
34         vec3 V = normalize(eyePos - fWorldPos);
35         vec3 H = normalize(L + V);
36         float ks = pow(max(dot(N, H), 0.0), shininess);
37         vec4 specular = ks * specularProduct;
38
39         gl_FragColor = ambientProduct + diffuse + specular;
40         gl_FragColor.a = 1.0;
41     }
42     </script>
43
44     <script type="text/javascript" src="../Common/webgl-utils.js"></script>
45     <script type="text/javascript" src="../Common/initShaders.js"></script>
46     <script type="text/javascript" src="../Common/MV.js"></script>
47     <script type="text/javascript" src="../trackball.js"></script>
48     <script type="text/javascript" src="phong.js"></script>
49 </head>
50 <body>
51     <div style="width: 512px; text-align: center;">
52         Subdivision Level: 1 <input type="range" id="level" value="1" min="1" max="5" step="1"> 5
53     </div>
54     <canvas id="gl-canvas" width="512" height="512">
55         Oops... your browser doesn't support the HTML5 canvas element!
56     </canvas>
57 </body>
58 </html>
```

Ln 48, Col 50 Spaces: 4 UTF-8 CRLF HTML

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS phong.js > divideTriangle

```
1  var gl;
2  var points = [];
3  var normals = [];
4
5  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
6
7  //var modelMatrix;
8  var modelMatrixLoc;
9
10 window.onload = function init()
11 {
12     var canvas = document.getElementById("gl-canvas");
13
14     gl = WebGLUtils.setupWebGL(canvas);
15     if( !gl ) {
16         alert("WebGL isn't available!");
17     }
18
19     generateTetrahedron(1);
20
21     // virtual trackball
22     var trball = trackball(canvas.width, canvas.height);
23     var mouseDown = false;
24
25     canvas.addEventListener("mousedown", function (event) {
26         trball.start(event.clientX, event.clientY);
27
28         mouseDown = true;
29     });
30
31     canvas.addEventListener("mouseup", function (event) {
32         mouseDown = false;
33     });
34
35     canvas.addEventListener("mousemove", function (event) {
```



 `divideTriangle`

[illegible]

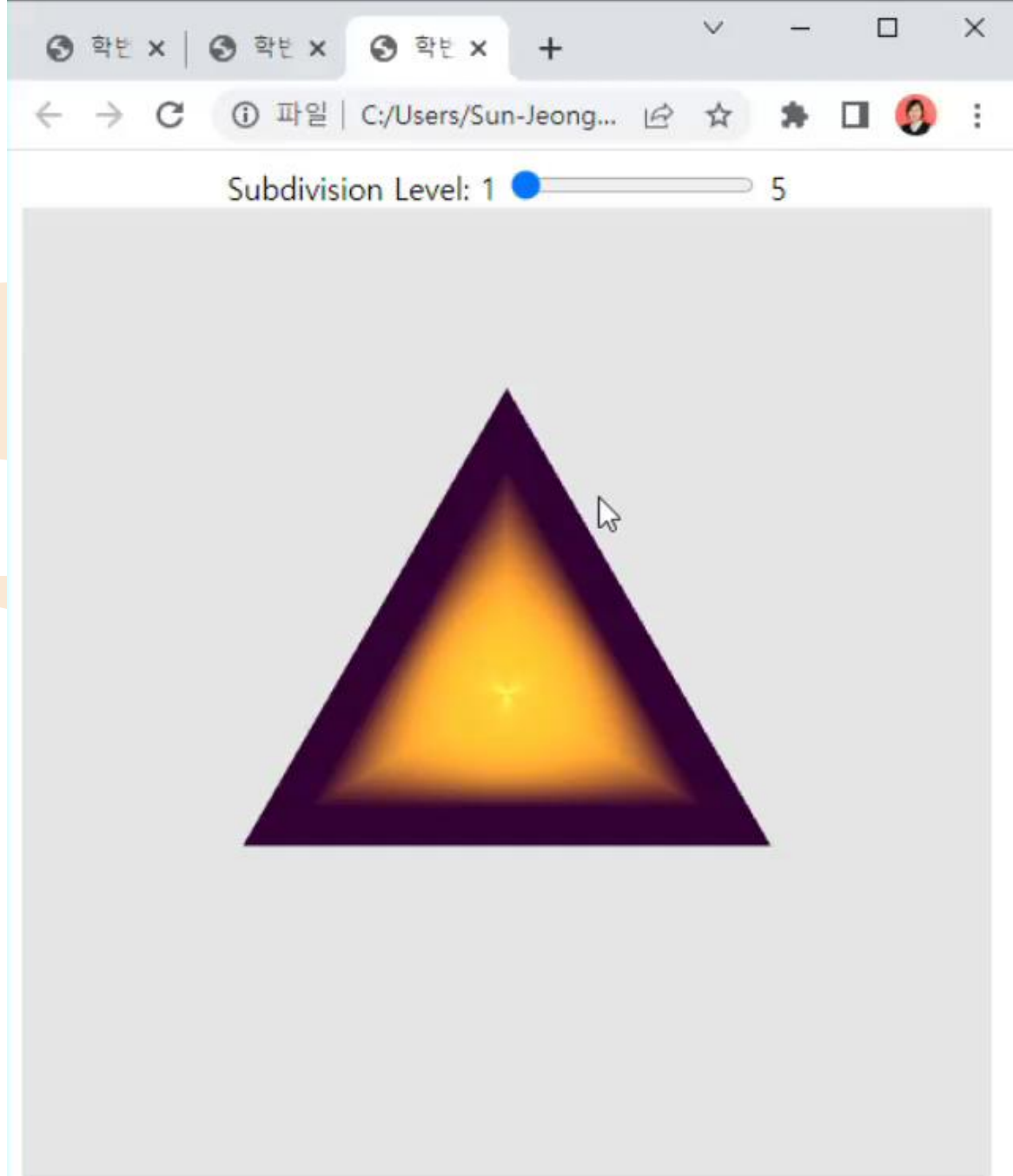
C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS phong.js > divideTriangle

```
100
101 // Event listeners for buttons
102 document.getElementById("level").onchange = function(event) {
103     var level = event.target.value;
104
105     points = [];
106     normals = [];
107     generateTetrahedron(level);
108
109     gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
110     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
111     gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
112     gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
113
114     render();
115 };
116
117 setLighting(program);
118
119 render();
120 };
121
122 function setLighting(program) {
123     var lightSrc = [0.0, 0.0, 1.0, 0.0];
124     var lightAmbient = [0.2, 0.2, 0.2, 1.0];
125     var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
126     var lightSpecular = [1.0, 1.0, 1.0, 1.0];
127
128     var matAmbient = [1.0, 0.0, 1.0, 1.0];
129     var matDiffuse = [1.0, 0.8, 0.0, 1.0];
130     var matSpecular = [1.0, 1.0, 1.0, 1.0];
131
132     var ambientProduct = mult(lightAmbient, matAmbient);
133     var diffuseProduct = mult(lightDiffuse, matDiffuse);
134     var specularProduct = mult(lightSpecular, matSpecular);
```

C: > Users > Sun-Jeong Kim > Desktop > CG > Week11 > JS phong.js > divideTriangle

```
153
154 function generateTetrahedron(level) {
155     var va = vec4(0.0, 0.0, 1.0, 1.0);
156     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
157     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
158     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
159
160     divideTriangle(va, vb, vc, level);
161     divideTriangle(va, vc, vd, level);
162     divideTriangle(va, vd, vb, level);
163     divideTriangle(vd, vc, vb, level);
164 }
165
166 function divideTriangle(a, b, c, level) {
167     if (level > 1) {
168         var ab = normalize(mix(a, b, 0.5), true);
169         var ac = normalize(mix(a, c, 0.5), true);
170         var bc = normalize(mix(b, c, 0.5), true);
171
172         divideTriangle(a, ab, ac, level - 1);
173         divideTriangle(ab, b, bc, level - 1);
174         divideTriangle(bc, c, ac, level - 1);
175         divideTriangle(ab, bc, ac, level - 1);
176     }
177     else {
178         points.push(a);
179         normals.push(vec4(a[0], a[1], a[2], 0.0));
180         points.push(b);
181         normals.push(vec4(b[0], b[1], b[2], 0.0));
182         points.push(c);
183         normals.push(vec4(c[0], c[1], c[2], 0.0));
184     }
185 }
186
```





연습 문제

- “Moving Light Sources” 예제에서 구현했던 scene.html과 scene.js를 Phong Shading으로 변경하시오.

