

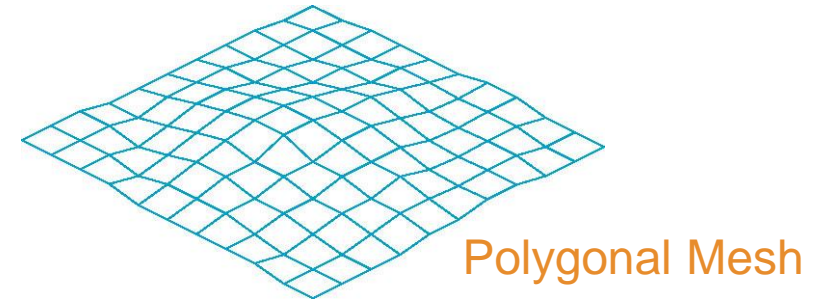
# Shading

11<sup>TH</sup> WEEK, 2021



# Polygonal Shading

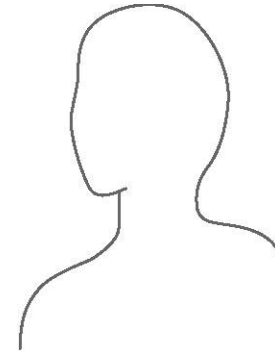
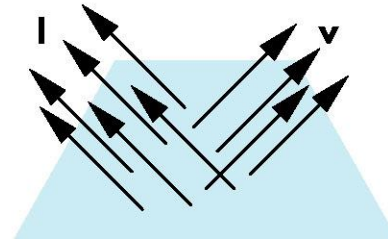
- Shading calculations are done for each vertex
  - Vertex colors become vertex shades



- 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

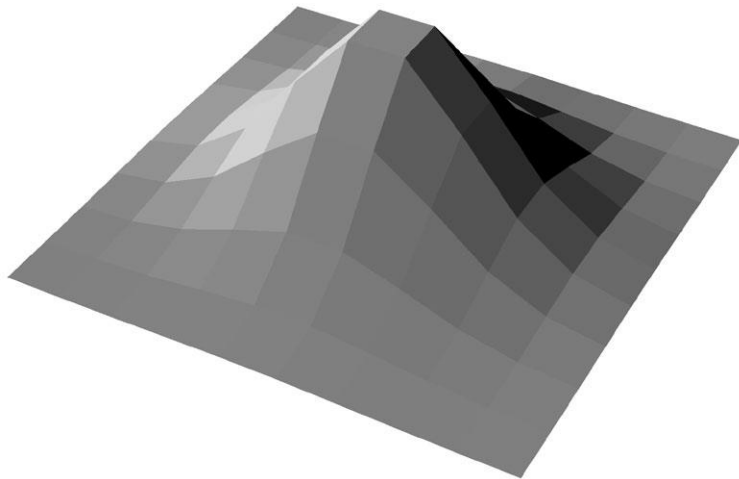


- One shading calculation for each polygon
- Consider model of sphere

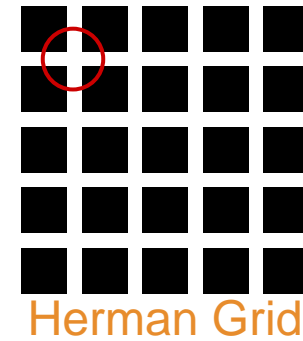


# Flat Shading

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



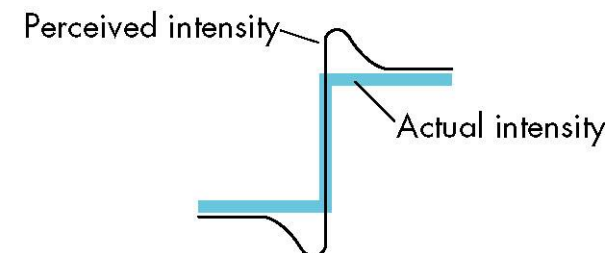
Flat shading of polygonal mesh



Herman Grid



Step chart

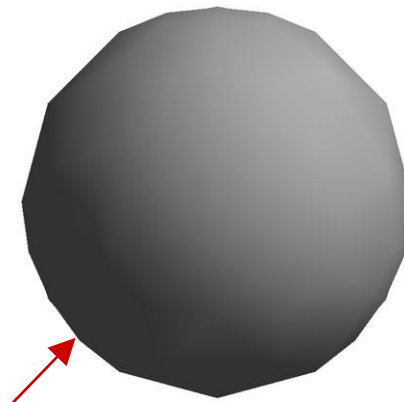
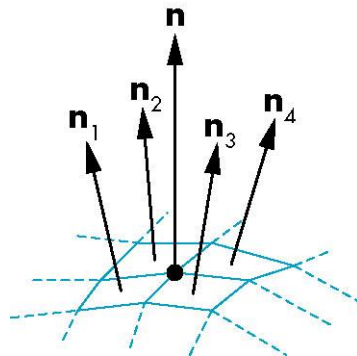


Perceived and actual intensities at an edge

# 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

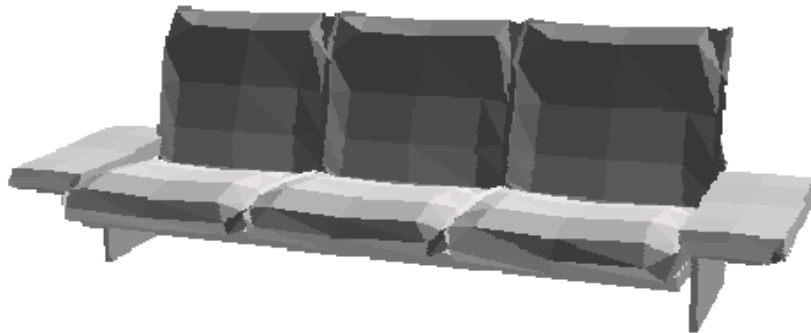
$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$



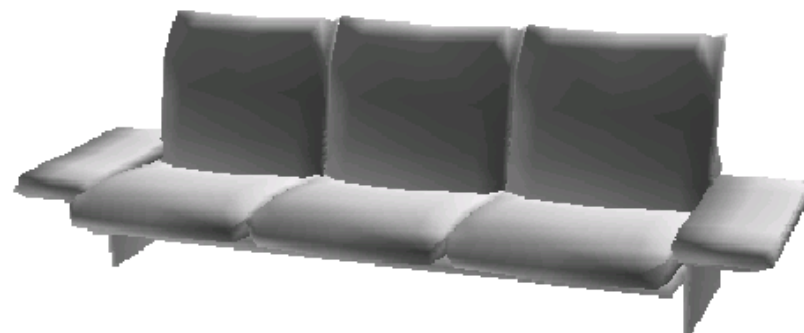
Note silhouette edge

# 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?



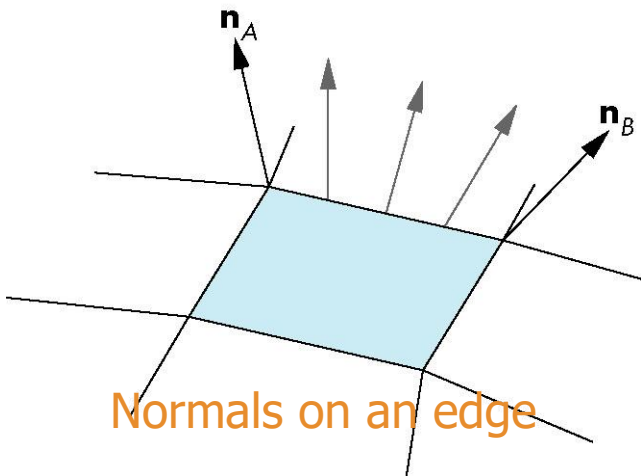
Flat Shading



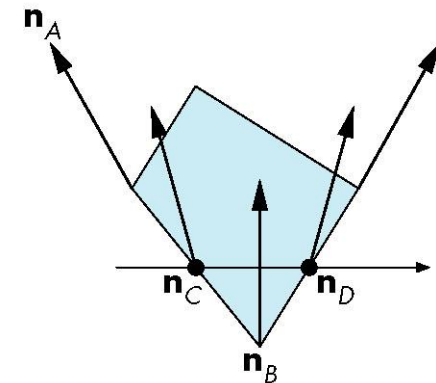
Smooth Shading

# 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

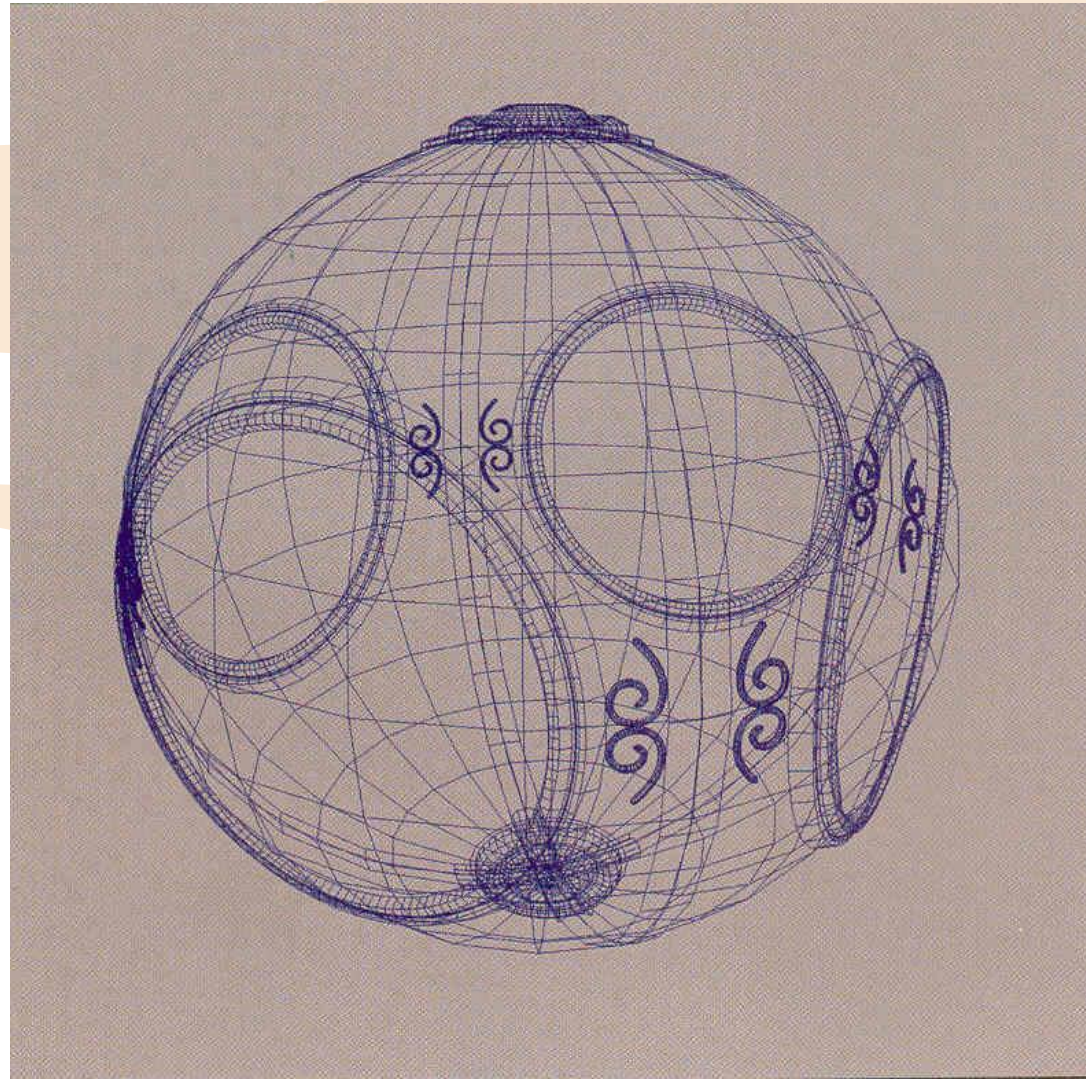
# Gouraud vs. Phong Shading

- 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



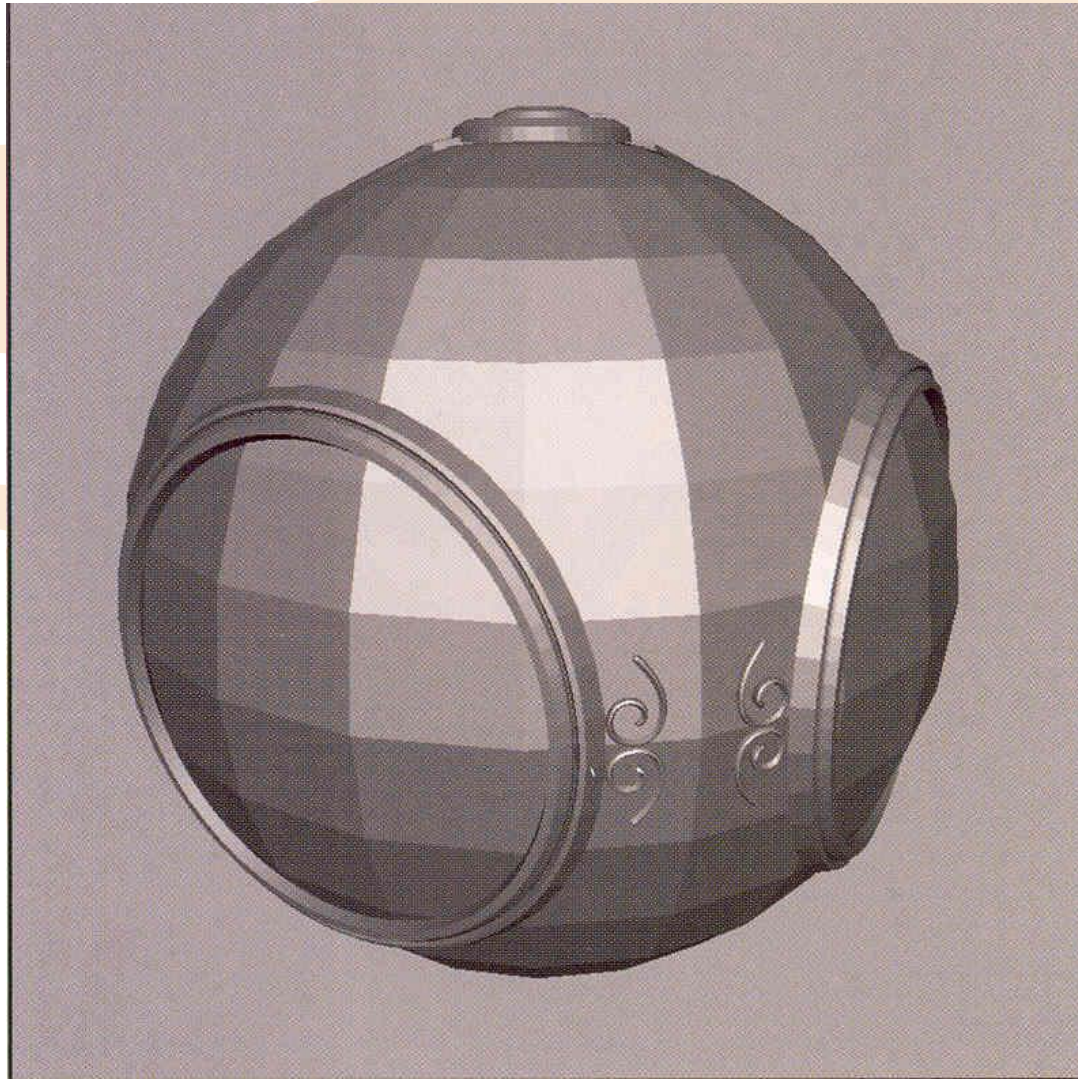
# Comparison

- 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

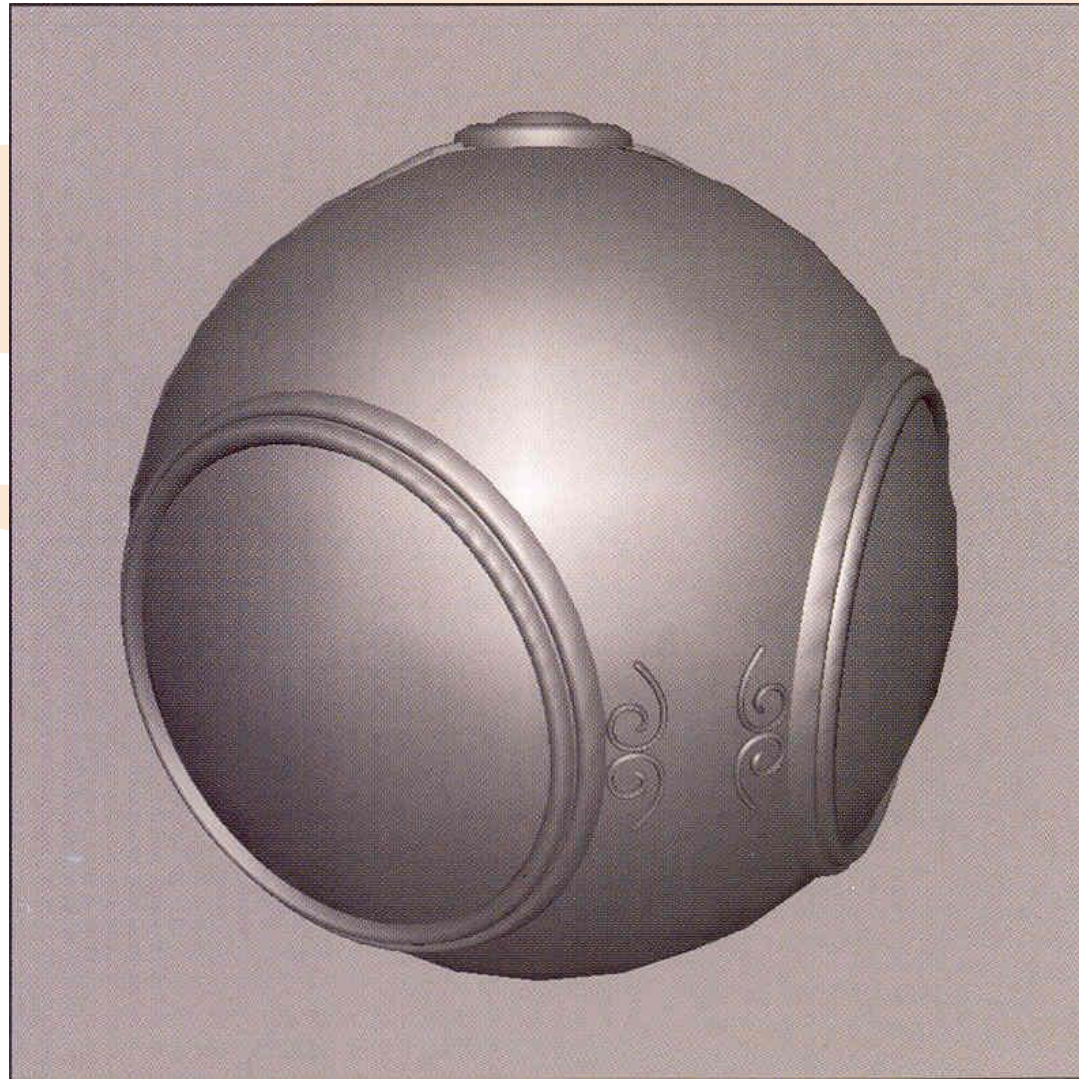


Wireframe



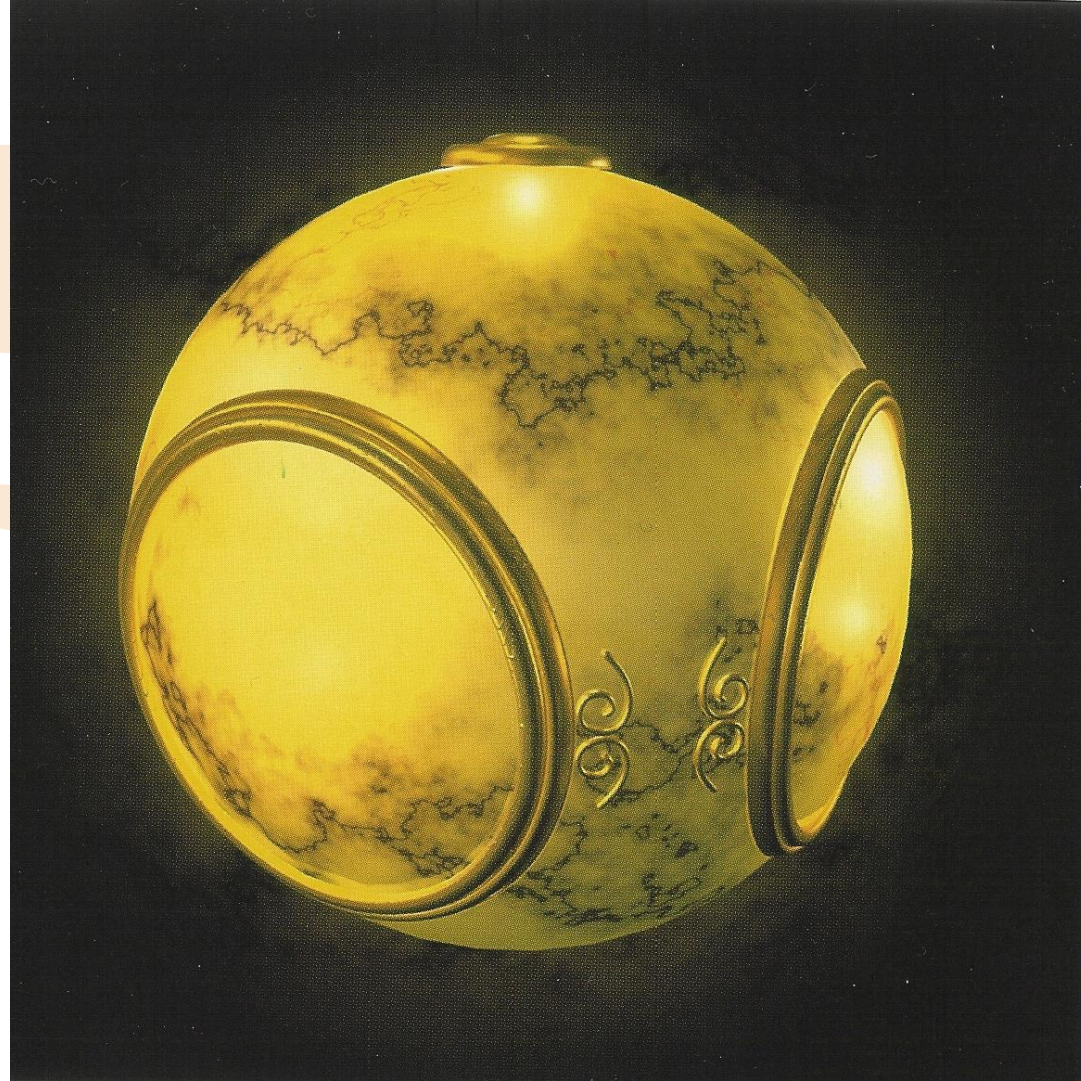


Flat Shading



Gouraud Shading





Bump Mapping

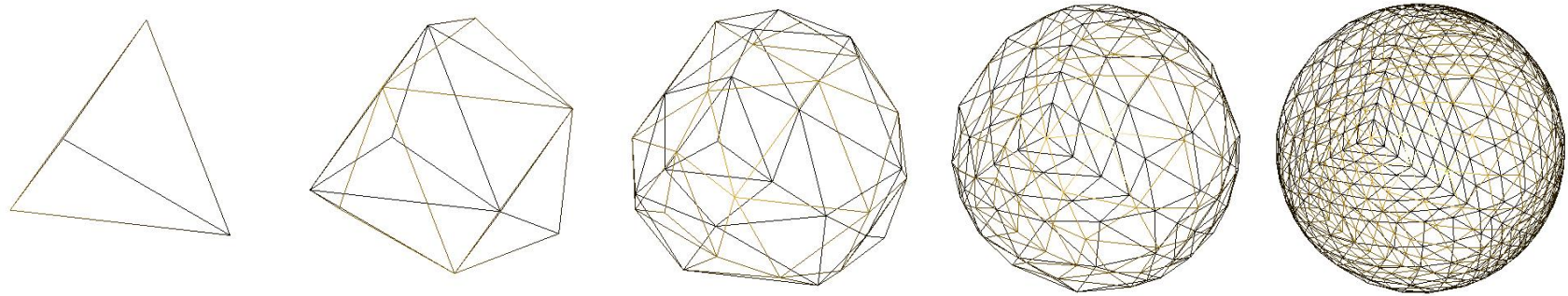


## Environmental Mapping

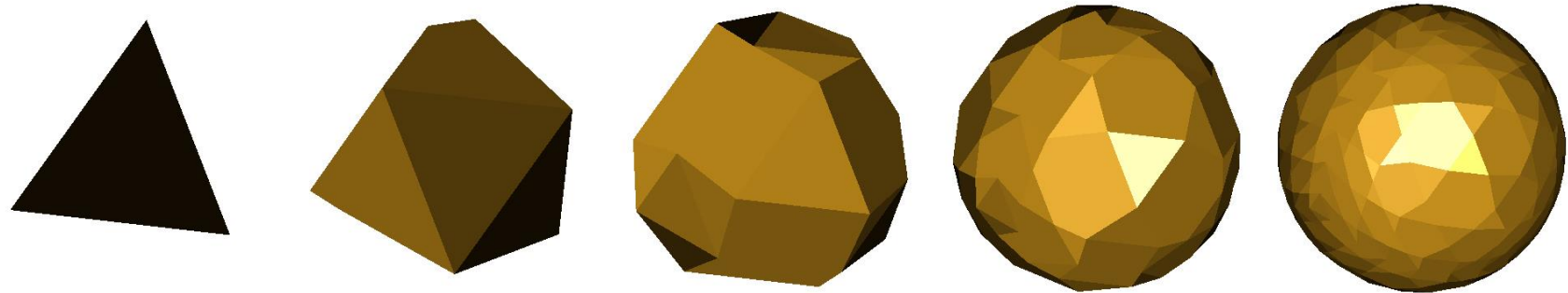


# Sphere by Recursive Subdivision

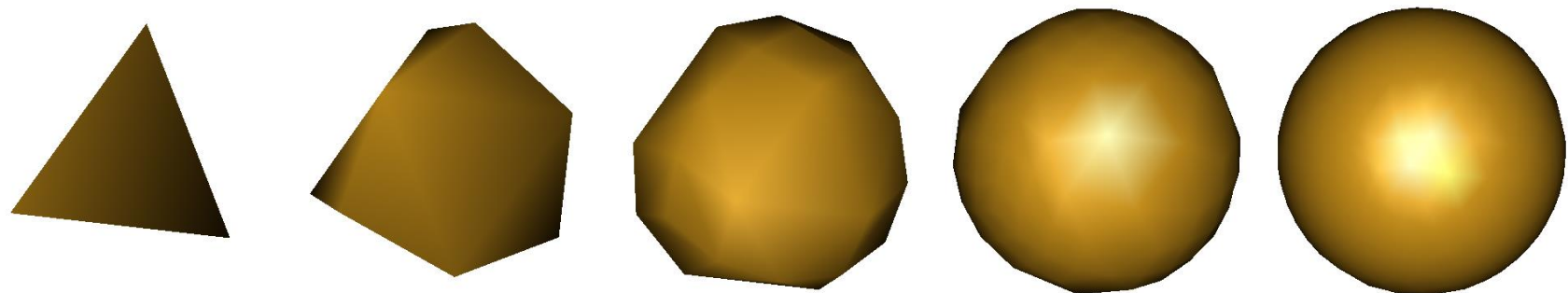
Wireframe



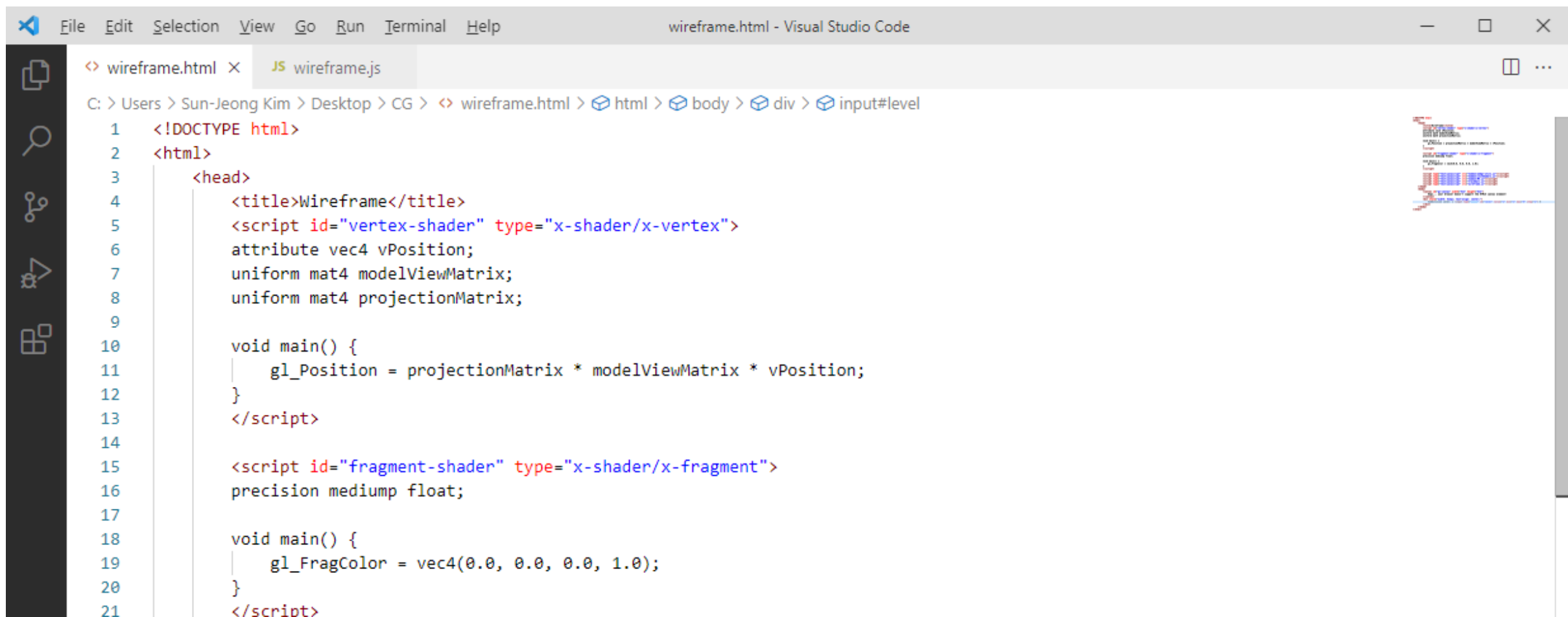
Flat Shading



Gouraud Shading



# Wireframe



The image shows a Visual Studio Code editor window titled "wireframe.html - Visual Studio Code". The editor has two tabs: "wireframe.html" and "JS wireframe.js". The "wireframe.html" tab is active, showing the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Wireframe</title>
5     <script id="vertex-shader" type="x-shader/x-vertex">
6       attribute vec4 vPosition;
7       uniform mat4 modelViewMatrix;
8       uniform mat4 projectionMatrix;
9
10      void main() {
11        |   gl_Position = projectionMatrix * modelViewMatrix * vPosition;
12      }
13    </script>
14
15    <script id="fragment-shader" type="x-shader/x-fragment">
16      precision mediump float;
17
18      void main() {
19        |   gl_FragColor = vec4(0.0, 0.0, 0.0, 1.0);
20      }
21    </script>
```

The code is a simple HTML document that includes two GLSL shaders. The vertex shader (lines 5-13) takes a position vector and transforms it using a model-view and projection matrix. The fragment shader (lines 15-21) sets a solid black color for all fragments. The browser's developer tools are open on the right, showing the "Elements" panel with the HTML structure: `<html> <head> <title>Wireframe</title> <script id="vertex-shader" type="x-shader/x-vertex"> ... </script> <script id="fragment-shader" type="x-shader/x-fragment"> ... </script> </html>`.



&lt; wireframe.html × JS wireframe.js

□ ...

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; &lt;&gt; wireframe.html &gt; html &gt; body &gt; div &gt; input#level



```
5      <script id="vertex-shader" type="x-shader/x-vertex">
6      attribute vec4 vPosition;
7      uniform mat4 modelViewMatrix;
8      uniform mat4 projectionMatrix;
9
10     void main() {
11     |   gl_Position = projectionMatrix * modelViewMatrix * vPosition;
12     }
13     </script>
14
15     <script id="fragment-shader" type="x-shader/x-fragment">
16     precision mediump float;
17
18     void main() {
19     |   gl_FragColor = vec4(0.0, 0.0, 0.0, 1.0);
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="trackball.js"></script>
27     <script type="text/javascript" src="wireframe.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     <div style="width: 512px; text-align: center;">
34     |   Subdivision Level: 1 <input type="range" id="level" value="1" min="1" max="5" step="1"> 5
35     </div>
36   </body>
37 </html>
```

wireframe.html JS wireframe.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS wireframe.js &gt; init

```
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 viewMatrix;
7  var modelViewMatrixLoc;
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 }
```

```
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 viewMatrix;
7  var modelViewMatrixLoc;
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 }
```

&lt; wireframe.html

JS wireframe.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS wireframe.js &gt; init

```
33
34 canvas.addEventListener("mousemove", function (event) {
35     if (mouseDown) {
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(1.0, 1.0, 1.0, 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 viewMatrix = lookAt(vec3(0, 0, 1), vec3(0, 0, 0), vec3(0, 1, 0));
64 modelViewMatrixLoc = gl.getUniformLocation(program, "modelViewMatrix");
65
```

&lt; wireframe.html

JS wireframe.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS wireframe.js &gt; init

```
65
66 // 3D orthographic viewing
67 var viewLength = 1.5;
68 if (canvas.width > canvas.height) {
69     var aspect = viewLength * canvas.width / canvas.height;
70     projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
71 }
72 else {
73     var aspect = viewLength * canvas.height / canvas.width;
74     projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
75 }
76 /*
77 // 3D perspective viewing
78 var aspect = canvas.width / canvas.height;
79 projectionMatrix = perspective(90, aspect, 0.1, 1000);
80 */
81 var projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
82 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
83
84 // Event listeners for buttons
85 document.getElementById("level").onchange = function(event) {
86     var level = event.target.value;
87
88     points = [];
89     generateTetrahedron(level);
90     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
91
92     render();
93 };
94
95 render();
96 };
97
```

&lt; wireframe.html

JS wireframe.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS wireframe.js &gt; init

```
97
98 function render() {
99     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
100
101     modelViewMatrix = mult(viewMatrix, trballMatrix);
102     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
103
104     gl.drawArrays(gl.LINES, 0, points.length);
105
106     window.requestAnimationFrame(render);
107 }
108
109 function generateTetrahedron(level) {
110     var va = vec4(0.0, 0.0, 1.0, 1.0);
111     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
112     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
113     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
114
115     divideTriangle(va, vb, vc, level);
116     divideTriangle(va, vc, vd, level);
117     divideTriangle(va, vd, vb, level);
118     divideTriangle(vd, vc, vb, level);
119 }
120
121 function divideTriangle(a, b, c, level) {
122     if (level > 1) {
123         var ab = normalize(mix(a, b, 0.5), true);
124         var ac = normalize(mix(a, c, 0.5), true);
125         var bc = normalize(mix(b, c, 0.5), true);
126
127         divideTriangle(a, ab, ac, level - 1);
128         divideTriangle(ab, b, bc, level - 1);
129         divideTriangle(bc, c, ac, level - 1);
```

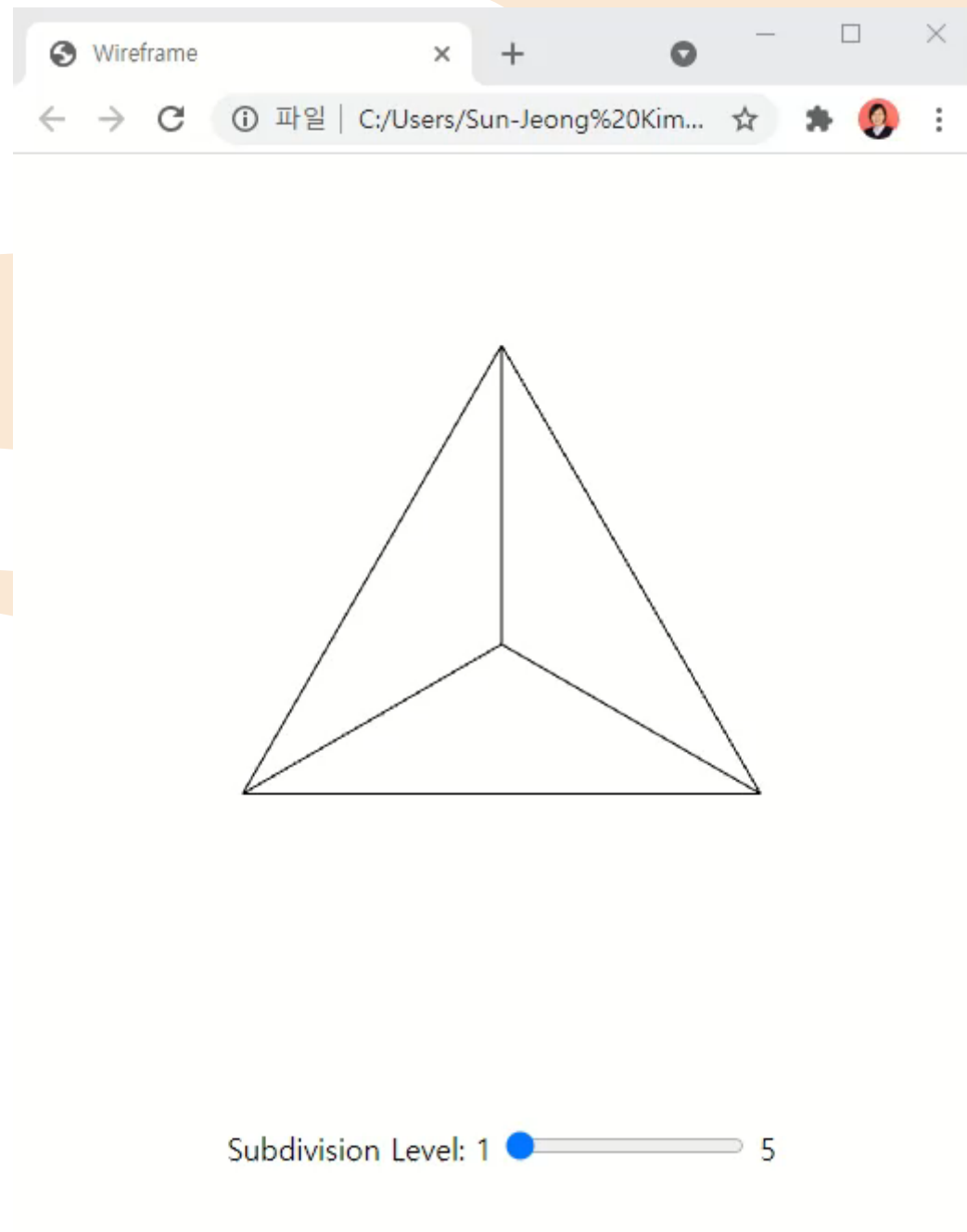
&lt; wireframe.html

JS wireframe.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS wireframe.js &gt; init

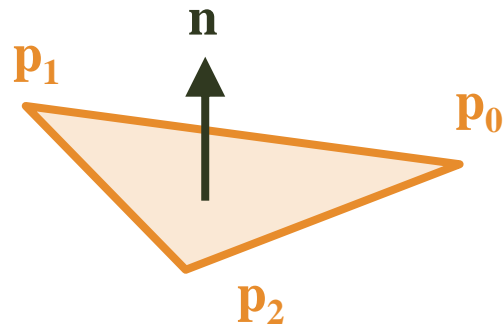
```
109 function generateTetrahedron(level) {
110     var va = vec4(0.0, 0.0, 1.0, 1.0);
111     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
112     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
113     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
114
115     divideTriangle(va, vb, vc, level);
116     divideTriangle(va, vc, vd, level);
117     divideTriangle(va, vd, vb, level);
118     divideTriangle(vd, vc, vb, level);
119 }
120
121 function divideTriangle(a, b, c, level) {
122     if (level > 1) {
123         var ab = normalize(mix(a, b, 0.5), true);
124         var ac = normalize(mix(a, c, 0.5), true);
125         var bc = normalize(mix(b, c, 0.5), true);
126
127         divideTriangle(a, ab, ac, level - 1);
128         divideTriangle(ab, b, bc, level - 1);
129         divideTriangle(bc, c, ac, level - 1);
130         divideTriangle(ab, bc, ac, level - 1);
131     }
132     else {
133         points.push(a);
134         points.push(b);
135         points.push(b);
136         points.push(c);
137         points.push(c);
138         points.push(a);
139     }
140 }
141
```

```
109 function generateTetrahedron(level) {
110     var va = vec4(0.0, 0.0, 1.0, 1.0);
111     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
112     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
113     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
114
115     divideTriangle(va, vb, vc, level);
116     divideTriangle(va, vc, vd, level);
117     divideTriangle(va, vd, vb, level);
118     divideTriangle(vd, vc, vb, level);
119 }
120
121 function divideTriangle(a, b, c, level) {
122     if (level > 1) {
123         var ab = normalize(mix(a, b, 0.5), true);
124         var ac = normalize(mix(a, c, 0.5), true);
125         var bc = normalize(mix(b, c, 0.5), true);
126
127         divideTriangle(a, ab, ac, level - 1);
128         divideTriangle(ab, b, bc, level - 1);
129         divideTriangle(bc, c, ac, level - 1);
130         divideTriangle(ab, bc, ac, level - 1);
131     }
132     else {
133         points.push(a);
134         points.push(b);
135         points.push(b);
136         points.push(c);
137         points.push(c);
138         points.push(a);
139     }
140 }
141
```



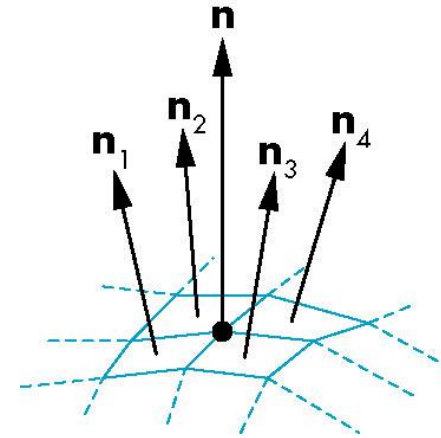
# Flat / Gouraud Shading

- Flat shading



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

- Gouraud Shading



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



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html X JS smooth.js

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; &lt;&gt; smooth.html &gt; html &gt; head &gt; script#vertex-shader

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>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 modelViewMatrix;
9        uniform mat4 projectionMatrix;
10
11        uniform vec4 lightDir;
12        uniform vec4 ambientProduct, diffuseProduct, specularProduct;
13        uniform float shininess;
14        varying vec4 fColor;
15
16        void main() {
17          gl_Position = projectionMatrix * modelViewMatrix * vPosition;
18
19          vec3 N = normalize(modelViewMatrix * vNormal).xyz;
20          vec3 L = normalize(lightDir.xyz);
21          float kd = max(dot(L, N), 0.0);
22          vec4 diffuse = kd * diffuseProduct;
23
24          vec3 V = normalize(modelViewMatrix*vPosition).xyz;    // origin: camera position
25          vec3 H = normalize(L - V).xyz;
26          float ks = pow(max(dot(N, H), 0.0), shininess);
27          vec4 specular = ks * specularProduct;
28
29          if (dot(L, N) < 0.0)    specular = vec4(0.0, 0.0, 0.0, 1.0);
30
31          fColor = ambientProduct + diffuse + specular;
32          fColor.a = 1.0;
33        }
```



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html X JS smooth.js

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; &lt;&gt; smooth.html &gt; html &gt; head &gt; script#vertex-shader

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



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; [⌘] flat

```
1  var gl;
2  var points = [];
3  var normals = [];
4  var fNormals = [];
5
6  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
7
8  var viewMatrix;
9  var modelViewMatrixLoc;
10
11  var flat = 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
```



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; [?] flat

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



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; [?] flat

```
67 // Create a buffer object, initialize it, and associate it with
68 // the associated attribute variable in our vertex shader
69 var nBufferId = gl.createBuffer();
70 gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
71 gl.bufferData(gl.ARRAY_BUFFER, flatten(fNormals), gl.STATIC_DRAW);
72
73 var vNormal = gl.getAttribLocation(program, "vNormal");
74 gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
75 gl.enableVertexAttribArray(vNormal);
76
77 viewMatrix = lookAt(vec3(0, 0, 1), vec3(0, 0, 0), vec3(0, 1, 0));
78 modelViewMatrixLoc = gl.getUniformLocation(program, "modelViewMatrix");
79
80 // 3D orthographic viewing
81 var viewLength = 1.5;
82 if (canvas.width > canvas.height) {
83     var aspect = viewLength * canvas.width / canvas.height;
84     projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
85 }
86 else {
87     var aspect = viewLength * canvas.height / canvas.width;
88     projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
89 }
90 /*
91 // 3D perspective viewing
92 var aspect = canvas.width / canvas.height;
93 projectionMatrix = perspective(90, aspect, 0.1, 1000);
94 */
95 var projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
96 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
97
98 // Event listeners for buttons
99 document.getElementById("change").onclick = function () {
```

&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; [?] flat

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



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; [flat]

```
131
132     setLighting(program);
133
134     render();
135 };
136
137 function render() {
138     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
139
140     modelViewMatrix = mult(viewMatrix, trballMatrix);
141     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
142
143     gl.drawArrays(gl.TRIANGLES, 0, points.length);
144
145     window.requestAnimationFrame(render);
146 }
147
148 function setLighting(program) {
149     var lightDir = [0.0, 0.0, 1.0, 0.0];
150     var lightAmbient = [0.2, 0.2, 0.2, 1.0];
151     var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
152     var lightSpecular = [1.0, 1.0, 1.0, 1.0];
153
154     var matAmbient = [1.0, 0.0, 1.0, 1.0];
155     var matDiffuse = [1.0, 0.8, 0.0, 1.0];
156     var matSpecular = [1.0, 1.0, 1.0, 1.0];
157     var matShininess = 20.0;
158
159     var ambientProduct = mult(lightAmbient, matAmbient);
160     var diffuseProduct = mult(lightDiffuse, matDiffuse);
161     var specularProduct = mult(lightSpecular, matSpecular);
162
163     gl.uniform4fv(gl.getUniformLocation(program, "lightDir"), lightDir);
```

&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; flat

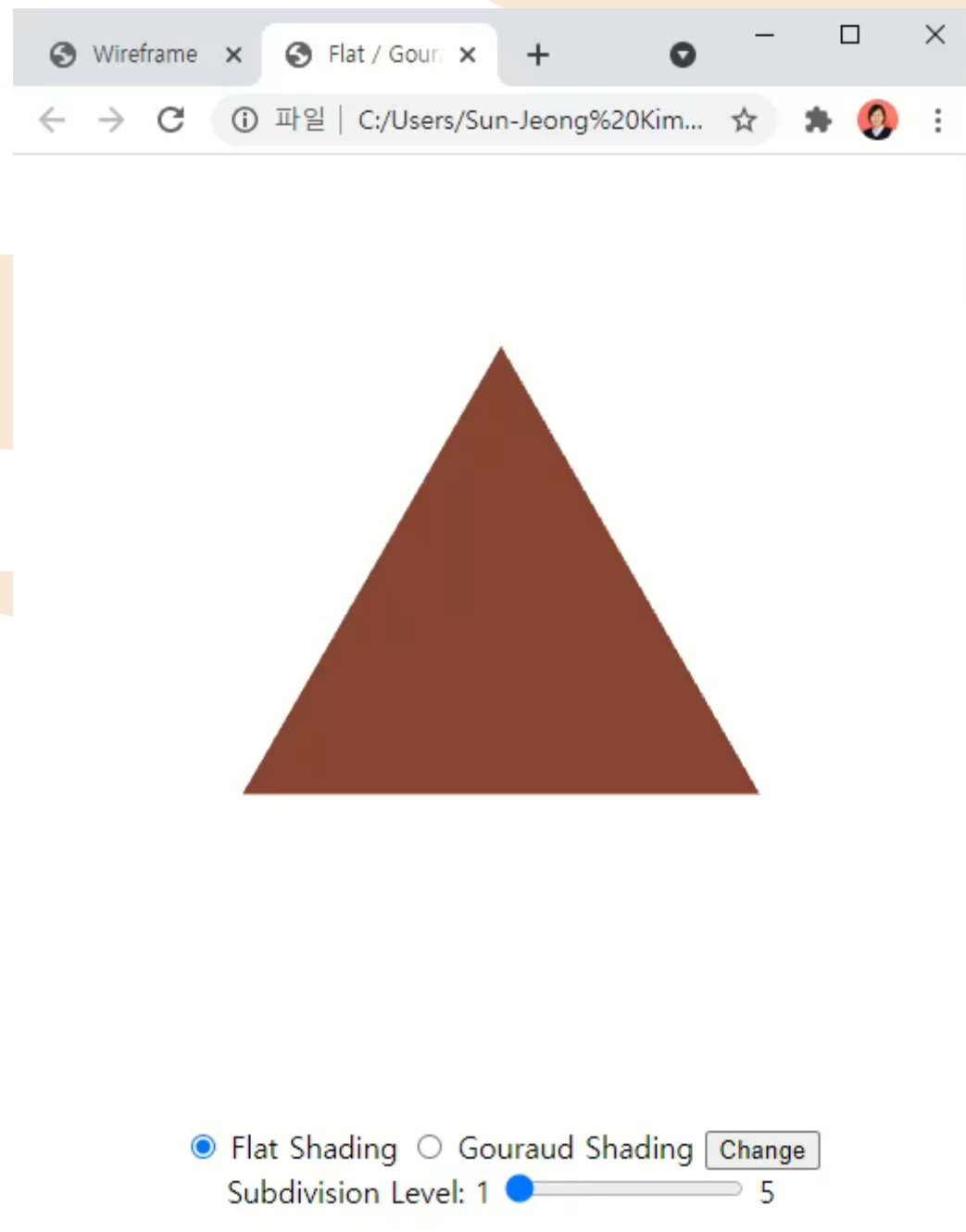
```
163     gl.uniform4fv(gl.getUniformLocation(program, "lightDir"), lightDir);
164     gl.uniform4fv(gl.getUniformLocation(program, "ambientProduct"), ambientProduct);
165     gl.uniform4fv(gl.getUniformLocation(program, "diffuseProduct"), diffuseProduct);
166     gl.uniform4fv(gl.getUniformLocation(program, "specularProduct"), specularProduct);
167     gl.uniform1f(gl.getUniformLocation(program, "shininess"), matShininess);
168 }
169
170 function generateTetrahedron(level) {
171     var va = vec4(0.0, 0.0, 1.0, 1.0);
172     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
173     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
174     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
175
176     divideTriangle(va, vb, vc, level);
177     divideTriangle(va, vc, vd, level);
178     divideTriangle(va, vd, vb, level);
179     divideTriangle(vd, vc, vb, level);
180 }
181
182 function divideTriangle(a, b, c, level) {
183     if (level > 1) {
184         var ab = normalize(mix(a, b, 0.5), true);
185         var ac = normalize(mix(a, c, 0.5), true);
186         var bc = normalize(mix(b, c, 0.5), true);
187
188         divideTriangle(a, ab, ac, level - 1);
189         divideTriangle(ab, b, bc, level - 1);
190         divideTriangle(bc, c, ac, level - 1);
191         divideTriangle(ab, bc, ac, level - 1);
192     }
193     else {
194         points.push(a);
195         normals.push(vec4(a[0], a[1], a[2], 0.0));
```



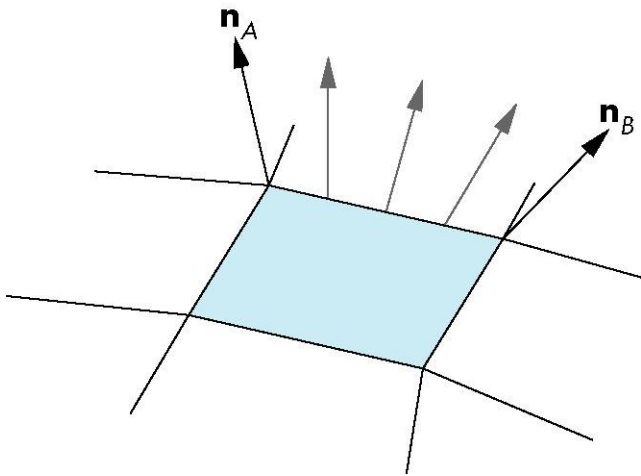
wireframe.html JS wireframe.js smooth.html JS smooth.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS smooth.js &gt; flat

```
181
182 function divideTriangle(a, b, c, level) {
183     if (level > 1) {
184         var ab = normalize(mix(a, b, 0.5), true);
185         var ac = normalize(mix(a, c, 0.5), true);
186         var bc = normalize(mix(b, c, 0.5), true);
187
188         divideTriangle(a, ab, ac, level - 1);
189         divideTriangle(ab, b, bc, level - 1);
190         divideTriangle(bc, c, ac, level - 1);
191         divideTriangle(ab, bc, ac, level - 1);
192     }
193     else {
194         points.push(a);
195         normals.push(vec4(a[0], a[1], a[2], 0.0));
196         points.push(b);
197         normals.push(vec4(b[0], b[1], b[2], 0.0));
198         points.push(c);
199         normals.push(vec4(c[0], c[1], c[2], 0.0));
200
201         var t1 = subtract(b, a);
202         var t2 = subtract(c, a);
203         var n = cross(t1, t2);
204         normal = normalize(vec4(n[0], n[1], n[2], 0.0));
205         fNormals.push(normal);
206         fNormals.push(normal);
207         fNormals.push(normal);
208     }
209 }
```

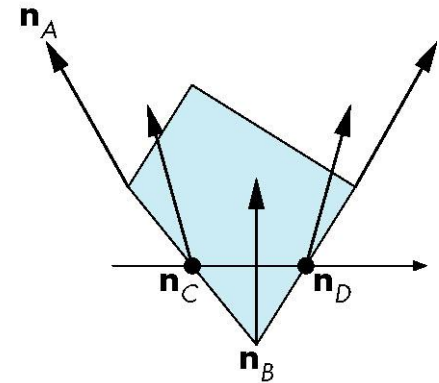


# Phong Shading



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

&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js &lt;&gt; phong.html X JS phong.js

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; &lt;&gt; phong.html &gt; html &gt; head &gt; script#vertex-shader

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Phong Shading</title>
5      <script id="vertex-shader" type="x-shader/x-vertex">
6        attribute vec4 vPosition;
7        attribute vec4 vNormal;
8        uniform mat4 modelViewMatrix;
9        uniform mat4 projectionMatrix;
10
11        uniform vec4 lightDir;
12        varying vec3 fNormal, fLight, fEye;
13
14        void main() {
15          gl_Position = projectionMatrix * modelViewMatrix * vPosition;
16
17          fNormal = normalize(modelViewMatrix * vNormal).xyz;
18          fLight = normalize(lightDir.xyz);
19          fEye = normalize(modelViewMatrix * vPosition).xyz;
20        }
21      </script>
22
23      <script id="fragment-shader" type="x-shader/x-fragment">
24        precision mediump float;
25
26        uniform vec4 ambientProduct, diffuseProduct, specularProduct;
27        uniform float shininess;
28        varying vec3 fNormal, fLight, fEye;
29
30        void main() {
31          vec3 N = normalize(fNormal);
32          vec3 L = normalize(fLight);
33          vec3 V = normalize(fEye);
```





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

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

```

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

```





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

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS phong.js &gt; init &gt; viewLength

```
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 viewMatrix;
8  var modelViewMatrixLoc;
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     });
```



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js &lt;&gt; phong.html JS phong.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS phong.js &gt; init &gt; viewLength

```
34
35   canvas.addEventListener("mousemove", function (event) {
36       if (mousedown) {
37           trball.end(event.clientX, event.clientY);
38
39           trballMatrix = mat4(trball.rotationMatrix);
40       }
41   });
42
43   // Configure WebGL
44   gl.viewport(0, 0, canvas.width, canvas.height);
45   gl.clearColor(1.0, 1.0, 1.0, 1.0);
46
47   // Enable hidden-surface removal
48   gl.enable(gl.DEPTH_TEST);
49
50   // Load shaders and initialize attribute buffers
51   var program = initShaders(gl, "vertex-shader", "fragment-shader");
52   gl.useProgram(program);
53
54   // Load the data into the GPU
55   var bufferId = gl.createBuffer();
56   gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
57   gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
58
59   // Associate our shader variables with our data buffer
60   var vPosition = gl.getAttribLocation(program, "vPosition");
61   gl.vertexAttribPointer(vPosition, 4, gl.FLOAT, false, 0, 0);
62   gl.enableVertexAttribArray(vPosition);
63
64   // Create a buffer object, initialize it, and associate it with
65   // the associated attribute variable in our vertex shader
66   var nBufferId = gl.createBuffer();
```



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js &lt;&gt; phong.html JS phong.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS phong.js &gt; init &gt; viewLength

```
66   var nBufferId = gl.createBuffer();
67   gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
68   gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
69
70   var vNormal = gl.getAttribLocation(program, "vNormal");
71   gl.vertexAttribPointer(vNormal, 4, gl.FLOAT, false, 0, 0);
72   gl.enableVertexAttribArray(vNormal);
73
74   viewMatrix = lookAt(vec3(0, 0, 1), vec3(0, 0, 0), vec3(0, 1, 0));
75   modelViewMatrixLoc = gl.getUniformLocation(program, "modelViewMatrix");
76
77   // 3D orthographic viewing
78   var viewLength = 1.5;
79   if (canvas.width > canvas.height) {
80       var aspect = viewLength * canvas.width / canvas.height;
81       projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
82   }
83   else {
84       var aspect = viewLength * canvas.height / canvas.width;
85       projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
86   }
87   /*
88   // 3D perspective viewing
89   var aspect = canvas.width / canvas.height;
90   projectionMatrix = perspective(90, aspect, 0.1, 1000);
91   */
92   var projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
93   gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
94
95   // Event listeners for buttons
96   document.getElementById("level").onchange = function (event) {
97       var level = event.target.value;
98
```





&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js &lt;&gt; phong.html JS phong.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS phong.js &gt; init &gt; viewLength

```
98
99     points = [];
100     normals = [];
101     generateTetrahedron(level);
102
103     gl.bindBuffer(gl.ARRAY_BUFFER, bufferId);
104     gl.bufferData(gl.ARRAY_BUFFER, flatten(points), gl.STATIC_DRAW);
105     gl.bindBuffer(gl.ARRAY_BUFFER, nBufferId);
106     gl.bufferData(gl.ARRAY_BUFFER, flatten(normals), gl.STATIC_DRAW);
107
108     render();
109 };
110
111 setLighting(program);
112
113 render();
114 };
115
116 function render() {
117     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
118
119     modelViewMatrix = mult(viewMatrix, trballMatrix);
120     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
121
122     gl.drawArrays(gl.TRIANGLES, 0, points.length);
123
124     window.requestAnimationFrame(render);
125 }
126
127 function setLighting(program) {
128     var lightDir = [0.0, 0.0, 1.0, 0.0];
129     var lightAmbient = [0.2, 0.2, 0.2, 1.0];
130     var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
```



&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js &lt;&gt; phong.html JS phong.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS phong.js &gt; init &gt; viewLength

```
126
127 function setLighting(program) {
128     var lightDir = [0.0, 0.0, 1.0, 0.0];
129     var lightAmbient = [0.2, 0.2, 0.2, 1.0];
130     var lightDiffuse = [1.0, 1.0, 1.0, 1.0];
131     var lightSpecular = [1.0, 1.0, 1.0, 1.0];
132
133     var matAmbient = [1.0, 0.0, 1.0, 1.0];
134     var matDiffuse = [1.0, 0.8, 0.0, 1.0];
135     var matSpecular = [1.0, 1.0, 1.0, 1.0];
136     var matShininess = 20.0;
137
138     var ambientProduct = mult(lightAmbient, matAmbient);
139     var diffuseProduct = mult(lightDiffuse, matDiffuse);
140     var specularProduct = mult(lightSpecular, matSpecular);
141
142     gl.uniform4fv(gl.getUniformLocation(program, "lightDir"), lightDir);
143     gl.uniform4fv(gl.getUniformLocation(program, "ambientProduct"), ambientProduct);
144     gl.uniform4fv(gl.getUniformLocation(program, "diffuseProduct"), diffuseProduct);
145     gl.uniform4fv(gl.getUniformLocation(program, "specularProduct"), specularProduct);
146     gl.uniform1f(gl.getUniformLocation(program, "shininess"), matShininess);
147 }
148
149 function generateTetrahedron(level) {
150     var va = vec4(0.0, 0.0, 1.0, 1.0);
151     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
152     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
153     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
154
155     divideTriangle(va, vb, vc, level);
156     divideTriangle(va, vc, vd, level);
157     divideTriangle(va, vd, vb, level);
158     divideTriangle(vd, vc, vb, level);
```

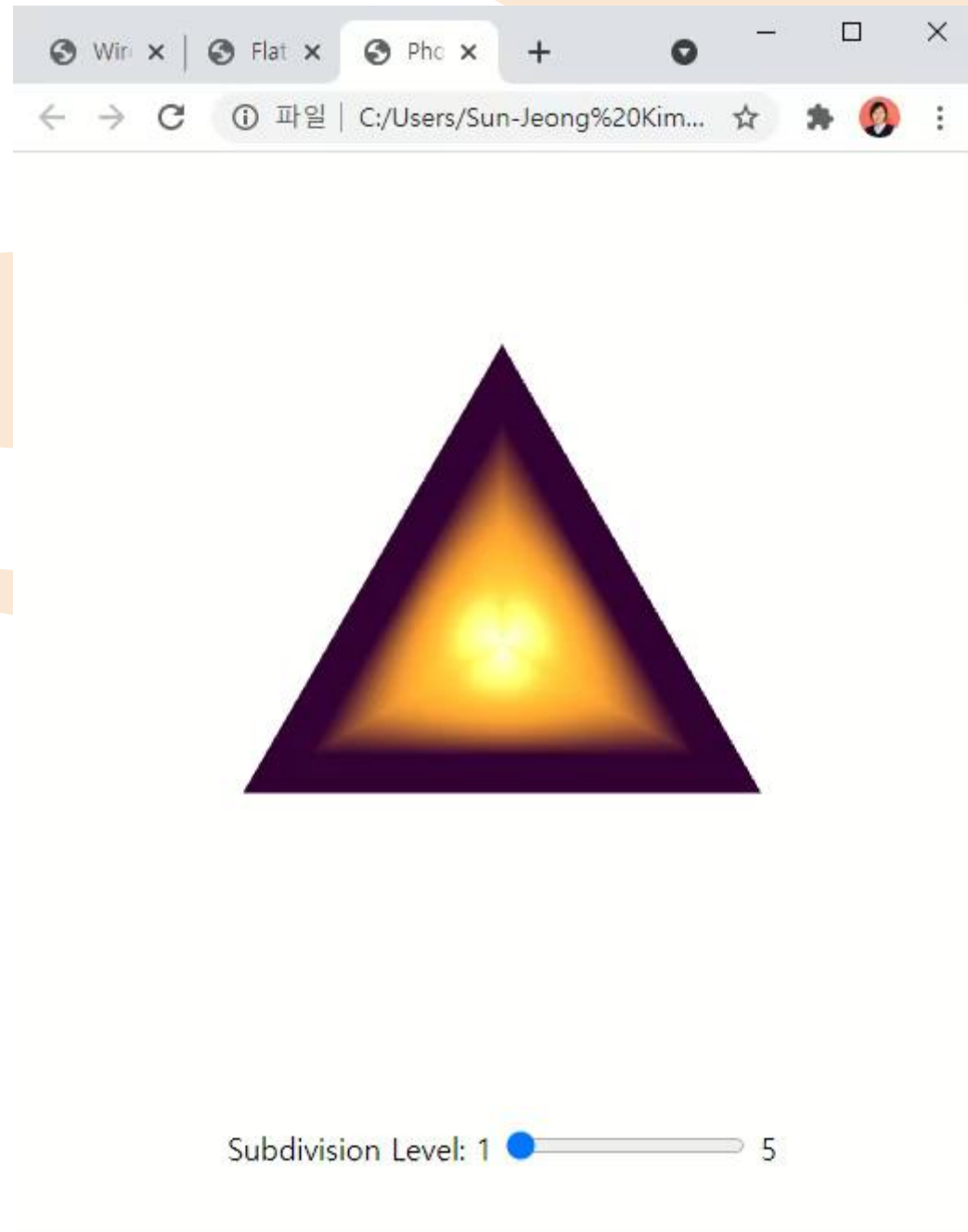


&lt;&gt; wireframe.html JS wireframe.js &lt;&gt; smooth.html JS smooth.js &lt;&gt; phong.html JS phong.js X

C: &gt; Users &gt; Sun-Jeong Kim &gt; Desktop &gt; CG &gt; JS phong.js &gt; init &gt; viewLength

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







수고하셨습니다