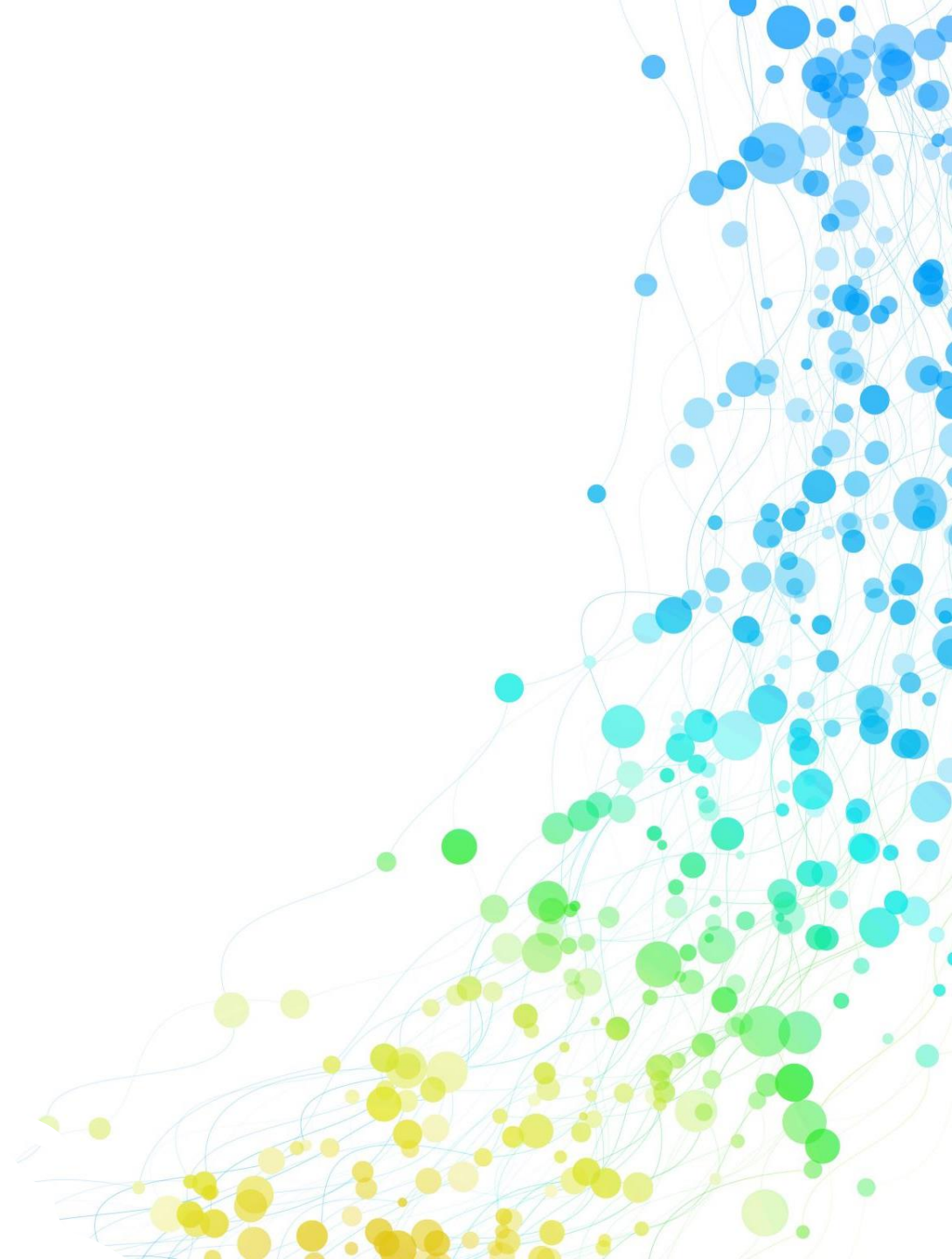
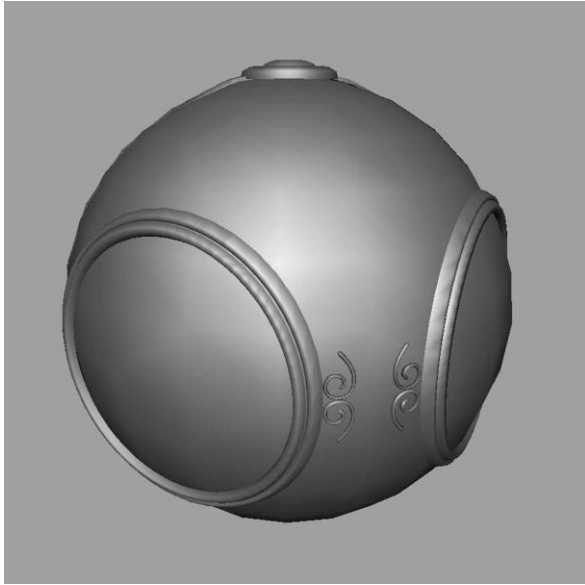


Environment Maps

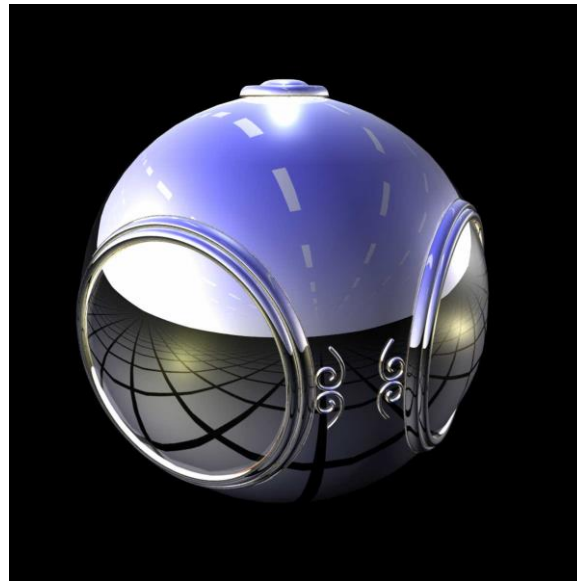
13TH WEEK, 2021



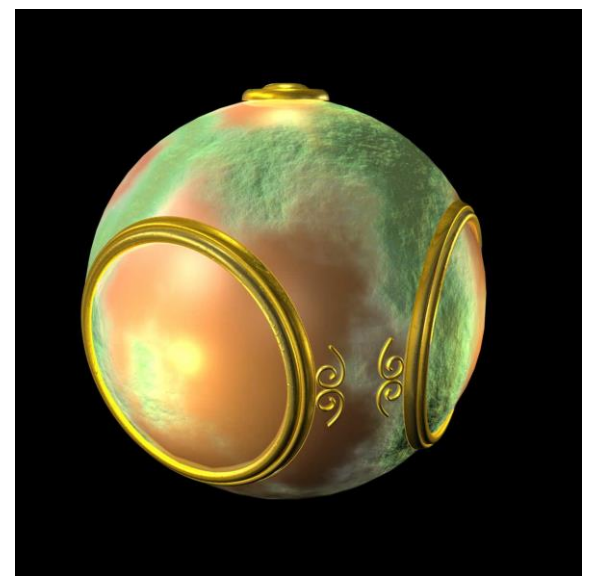
Mapping Variations



Gouraud shading



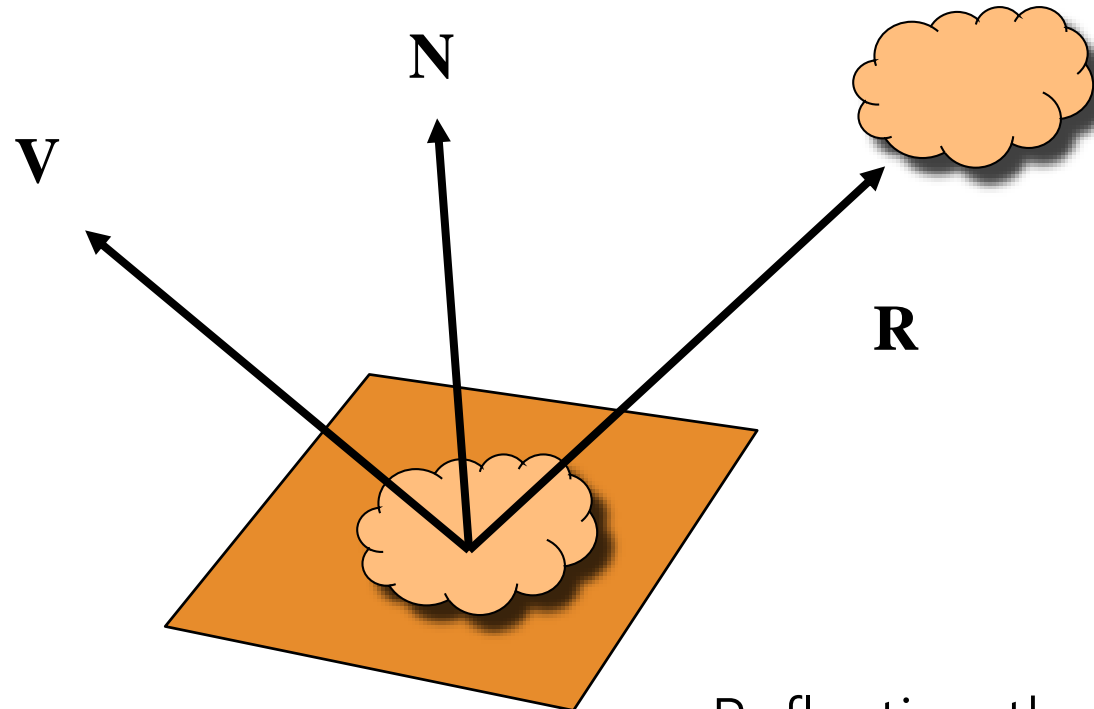
Environment mapping



Bump mapping

Environment Mapping

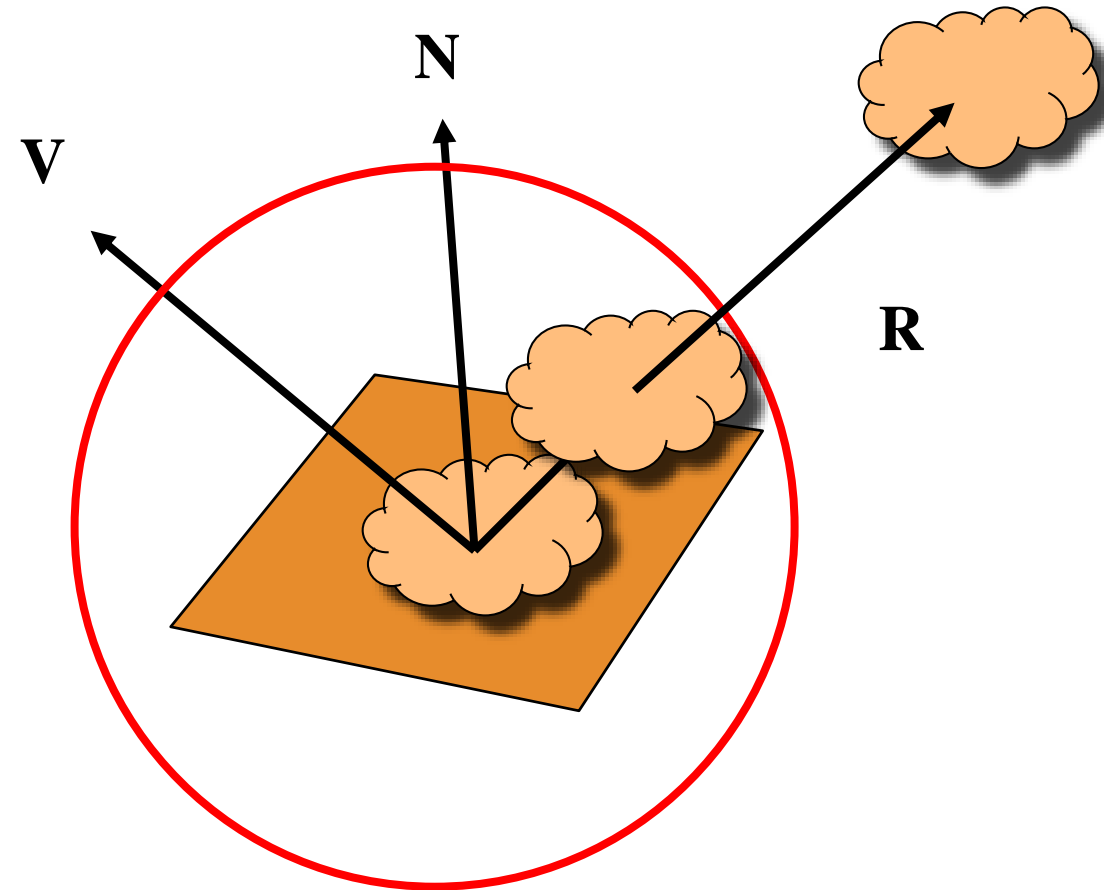
- Environmental (reflection) mapping is way to create the appearance of highly reflective surfaces without ray tracing which requires global calculations



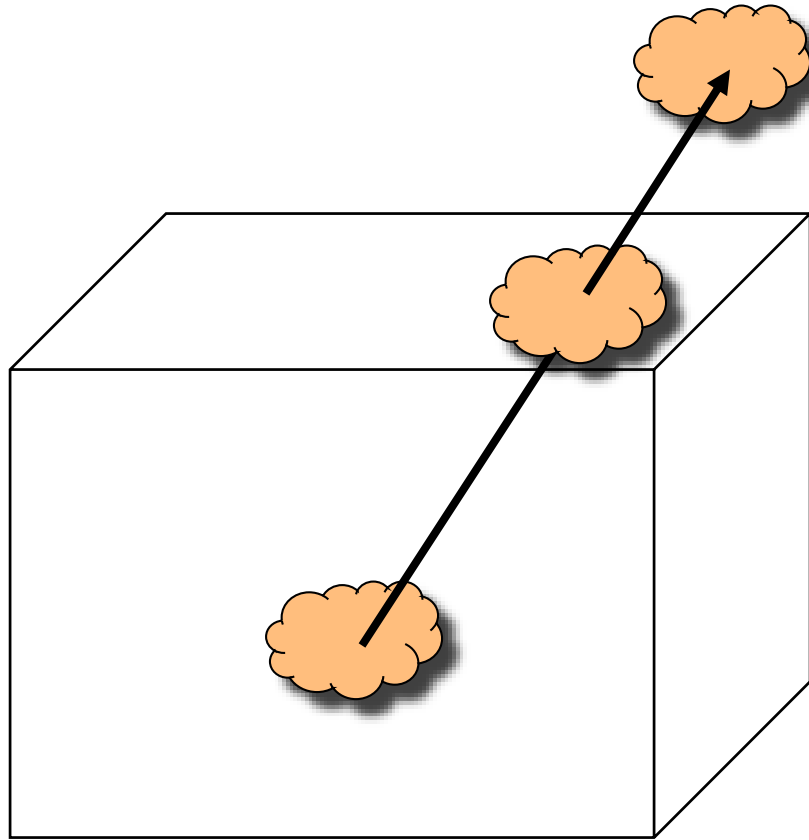
Reflecting the Environment

Hemisphere Map as a Texture

- If we map all objects to hemisphere, we cannot tell if they are on the sphere or anywhere else along the reflector
- Use the map on the sphere as a texture that can be mapped onto the object
- Can use other surfaces as the intermediate
 - Cube maps
 - Cylinder maps

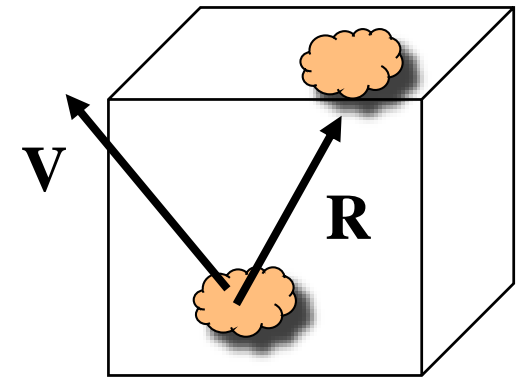


Cube Map



Indexing into Cube Map

- Compute $\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{V})\mathbf{N} - \mathbf{V}$
- Object at origin
- Use largest magnitude component of \mathbf{R} to determine face of cube
- Other two components give texture coordinates



WebGL Implementation

- WebGL supports only cube maps
 - Desktop OpenGL also supports sphere maps
- First must form map
 - Use images from a real camera
 - Form images with WebGL
- Texture map it to object

Cube Maps

- We can form a cube map texture by defining six 2D texture maps that correspond to the sides of a box
- Supported by WebGL through cubemap sampler

```
vec4 texColor = textureCube(mycube, texcoord);
```

- Texture coordinates must be 3D
 - Usually are given by the vertex location so we don't need compute separate tex coords

Environment Maps with Shaders

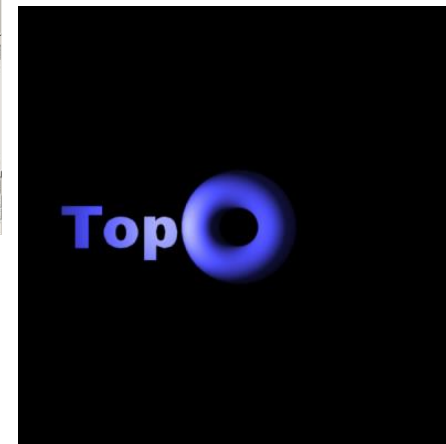
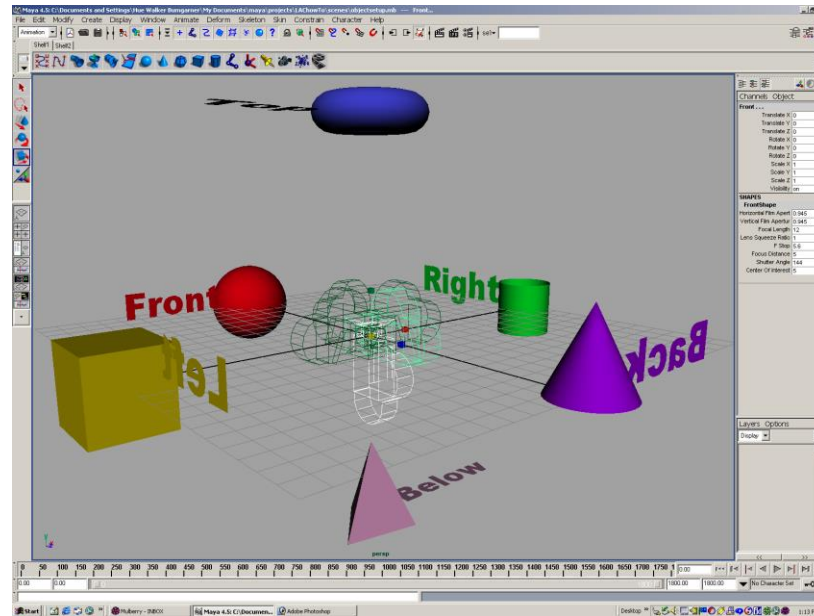
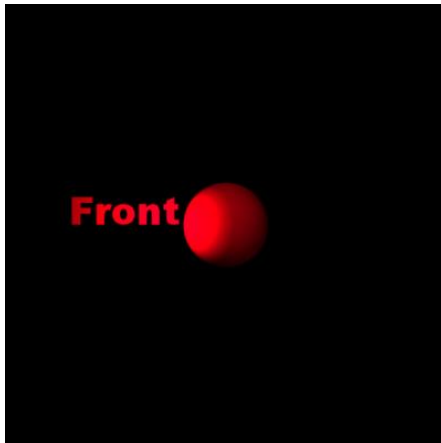
- Environment maps are usually computed in world coordinates which can differ from object coordinates because of the modeling matrix
 - May have to keep track of modeling matrix and pass it to the shaders as a uniform variable
- Can also use reflection map or refraction map for effects such as simulating water

Issues

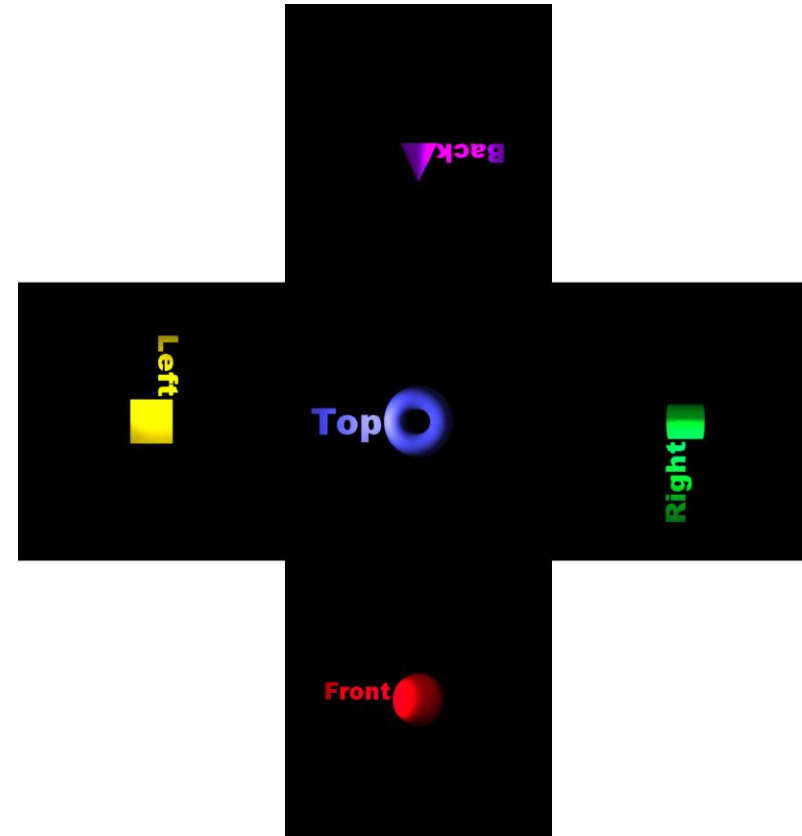
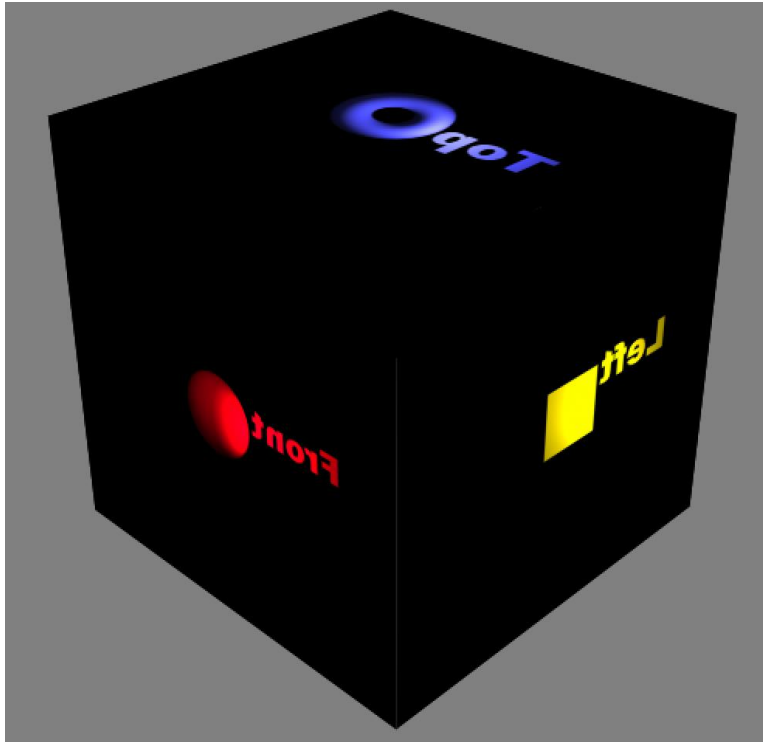
- Must assume environment is very far from object (equivalent to the difference between near and distant lights)
- Object cannot be concave (no self reflections possible)
- No reflections between objects
- Need a reflection map for each object
- Need a new map if viewer moves

Forming Cube Map

- Use six cameras, each with a 90 degree angle of view



vs. Cube Image



Cube Mapping in WebGL

```
gl.textureMap2D(  
    gl.TEXTURE_CUBE_MAP_POSITIVE_X,  
    level, rows, columns, border, gl.RGBA,  
    gl.UNSIGNED_BYTE, image1)
```

- Same for other five images
- Make one texture object out of the six images



envCube.html × JS envCube.js



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

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Cube Mapping</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             uniform mat3 worldMatrix;
11
12             varying vec3 fNormal, fEye;
13
14             void main() {
15                 gl_Position = projectionMatrix * modelViewMatrix * vPosition;
16
17                 fNormal = worldMatrix * vNormal.xyz;
18                 fEye = worldMatrix * vPosition.xyz;
19             }
20         </script>
21
22         <script id="fragment-shader" type="x-shader/x-fragment">
23             precision mediump float;
24
25             uniform samplerCube texMap;
26             varying vec3 fNormal, fEye;
27
28             void main() {
29                 vec3 N = normalize(fNormal);
30                 vec3 V = normalize(fEye);
31                 vec3 R = reflect(-V, N);
32
33                 gl_FragColor = textureCube(texMap, R);
```



envCube.html x JS envCube.js

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

```
33     gl_FragColor = textureCube(texMap, R);
34 }
35 </script>
36
37 <script type="text/javascript" src="Common/webgl-utils.js"></script>
38 <script type="text/javascript" src="Common/initShaders.js"></script>
39 <script type="text/javascript" src="Common/MV.js"></script>
40 <script type="text/javascript" src="trackball.js"></script>
41 <script type="text/javascript" src="envCube.js"></script>
42 </head>
43 <body>
44     <canvas id="gl-canvas" width="512" height="512">
45         Oops... your browser doesn't support the HTML5 canvas element!
46     </canvas>
47     <table>
48         <tr align="center">
49             <td>&nbsp;</td><td>X</td><td>Y</td><td>Z</td>
50         </tr>
51         <tr><td>Positive</td>
52             <td></td>
53             <td></td>
54             <td></td>
55         </tr>
56         <tr><td>Negative</td>
57             <td></td>
58             <td></td>
59             <td></td>
60         </tr>
61     </table>
62 </body>
63 </html>
```



envCube.html JS envCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS envCube.js > setTexture

```
1  var gl;
2  var points = [];
3  var normals = [];
4
5  var viewMatrix;
6  var modelViewMatrixLoc, worldMatrixLoc;
7
8  var trballMatrix = mat4(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
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(4);
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     });
```



envCube.html JS envCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS envCube.js > setTexture

```
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(0.9, 0.9, 0.9, 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();
```

envCube.html JS envCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS envCube.js > setTexture

```
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, 0.0, 1.0), vec3(0.0, 0.0, 0.0), vec3(0.0, 1.0, 0.0));
75 modelViewMatrixLoc = gl.getUniformLocation(program, "modelViewMatrix");
76 worldMatrixLoc = gl.getUniformLocation(program, "worldMatrix");
77
78 // 3D orthographic viewing
79 var viewLength = 1.5;
80 var projectionMatrix;
81 if (canvas.width > canvas.height) {
82     var aspect = viewLength * canvas.width / canvas.height;
83     projectionMatrix = ortho(-aspect, aspect, -viewLength, viewLength, -viewLength, 1000);
84 }
85 else {
86     var aspect = viewLength * canvas.height / canvas.width;
87     projectionMatrix = ortho(-viewLength, viewLength, -aspect, aspect, -viewLength, 1000);
88 }
89 /*
90 // 3D perspective viewing
91 var aspect = canvas.width / canvas.height;
92 projectionMatrix = perspective(90, aspect, 0.1, 1000);
93 */
94 var projectionMatrixLoc = gl.getUniformLocation(program, "projectionMatrix");
95 gl.uniformMatrix4fv(projectionMatrixLoc, false, flatten(projectionMatrix));
96
97 setTexture();
98
99 render();
```



envCube.html JS envCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS envCube.js > setTexture

```
100 };
101
102 function render() {
103     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
104
105     modelViewMatrix = mult(viewMatrix, trballMatrix);
106     gl.uniformMatrix4fv(modelViewMatrixLoc, false, flatten(modelViewMatrix));
107
108     var worldMatrix = [
109         vec3(trballMatrix[0][0], trballMatrix[0][1], trballMatrix[0][2]),
110         vec3(trballMatrix[1][0], trballMatrix[1][1], trballMatrix[1][2]),
111         vec3(trballMatrix[2][0], trballMatrix[2][1], trballMatrix[2][2])
112     ];
113     gl.uniformMatrix3fv(worldMatrixLoc, false, flatten(worldMatrix));
114
115     gl.drawArrays(gl.TRIANGLES, 0, points.length);
116
117     window.requestAnimationFrame(render);
118 }
119
120 function setTexture() {
121     var cubeMap = gl.createTexture();
122     gl.bindTexture(gl.TEXTURE_CUBE_MAP, cubeMap);
123
124     const faceInfos = [
125         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_X, url: 'images/LobbyXPos.bmp' },
126         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_X, url: 'images/LobbyXNeg.bmp' },
127         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Y, url: 'images/LobbyYPos.bmp' },
128         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Y, url: 'images/LobbyYNeg.bmp' },
129         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Z, url: 'images/LobbyZPos.bmp' },
130         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Z, url: 'images/LobbyZNeg.bmp' },
131     ];
132     faceInfos.forEach((faceInfo) => {
```



envCube.html

JS envCube.js X

C: > Users > Sun-Jeong Kim > Desktop > CG > JS envCube.js > setTexture

```
132     faceInfos.forEach((faceInfo) => {
133         const {target, url} = faceInfo;
134
135         // Upload the canvas to the cubemap face.
136         const level = 0;
137         const internalFormat = gl.RGBA;
138         const width = 512;
139         const height = 512;
140         const format = gl.RGBA;
141         const type = gl.UNSIGNED_BYTE;
142
143         // setup each face so it's immediately renderable
144         gl.texImage2D(target, level, internalFormat, width, height, 0, format, type, null);
145
146         // Asynchronously load an image
147         const image = new Image();
148         image.src = url;
149         image.addEventListener('load', function() {
150             // Now that the image has loaded upload it to the texture.
151             gl.bindTexture(gl.TEXTURE_CUBE_MAP, cubeMap);
152             gl.texImage2D(target, level, internalFormat, format, type, image);
153             gl.generateMipmap(gl.TEXTURE_CUBE_MAP);
154         });
155     });
156
157     gl.texParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
158     gl.texParameteri(gl.TEXTURE_CUBE_MAP, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
159 }
160
161 function generateTetrahedron(level) {
162     var va = vec4(0.0, 0.0, 1.0, 1.0);
163     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
164     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
```



envCube.html JS envCube.js X


C: > Users > Sun-Jeong Kim > Desktop > CG > JS envCube.js > setTexture

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




Cube Mapping

← → ↻ ⓘ 파일 | C:/Users/Sun-Jeong%20Kim/Desktop/CG/envCube.html




X Y Z

Positive



Negative



Elements Console Sources Network >> 6

top Filter Default levels No Issues

Uncaught DOMException: Failed to execute 'texImage2D' on 'WebGLRenderingContext': The image element contains cross-origin data, and may not be loaded.
at Image.<anonymous> (file:///C:/Users/Sun-Jeong%20Kim/Desktop/CG/envCube.js:152:16)

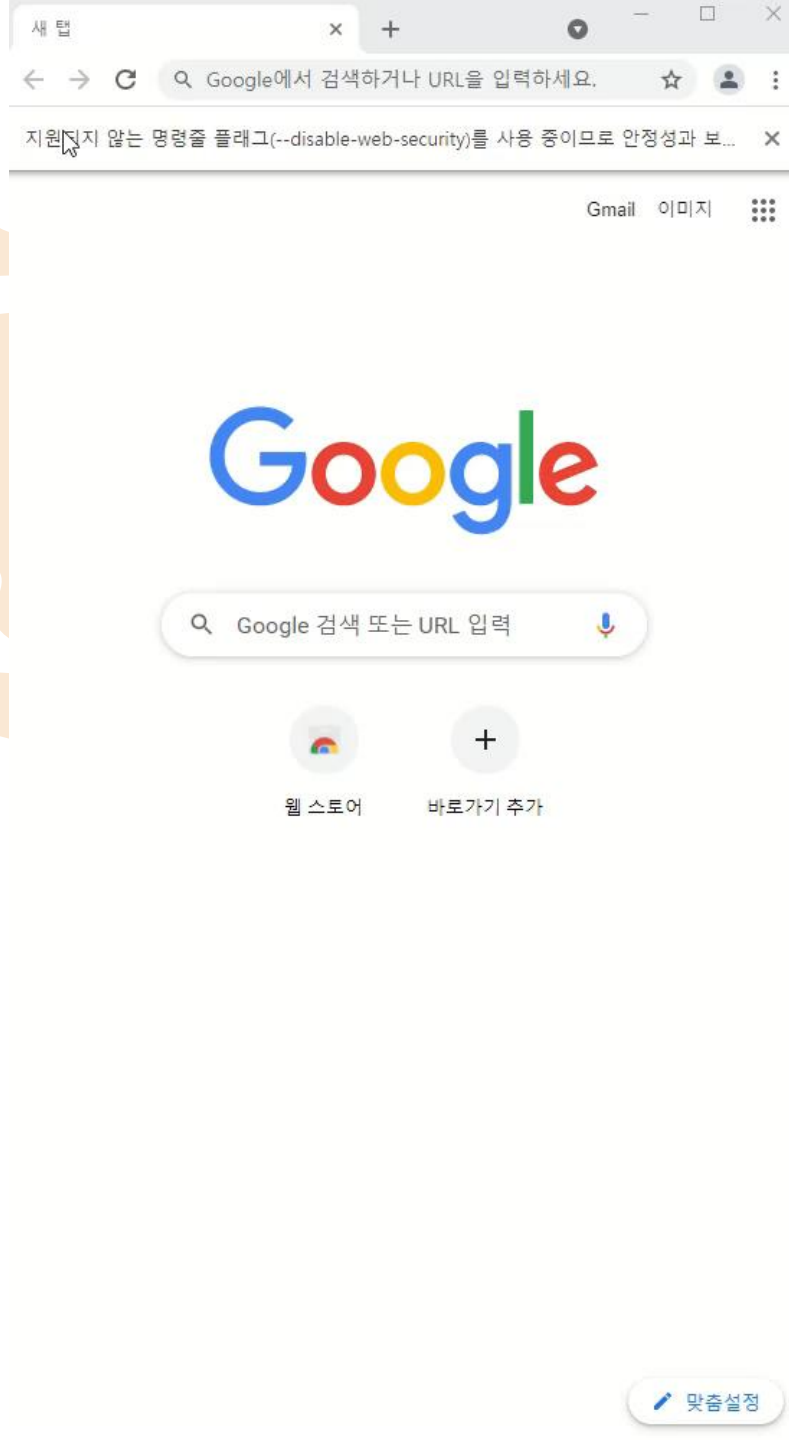
Console What's New

Highlights from the Chrome 90 update

- [New CSS Flexbox debugging tools](#)
Debug and inspect CSS Flexbox with the new CSS Flexbox debugging tools.
- [New Core Web Vitals overlay](#)
Visualize page performance with the new Core Web Vitals overlay.
- [Report Trusted Web Activity issues](#)
Debug Trusted Web Activity issues in the Issues panel.
- [New Trust Token pane](#)
New Trust Token pane in the Application Panel.
- [Emulate the CSS color-gamut media feature](#)
Emulate color-gamut to test different color standards.



"C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe" --disable-web-security --user-data-dir="C:\\chrome"

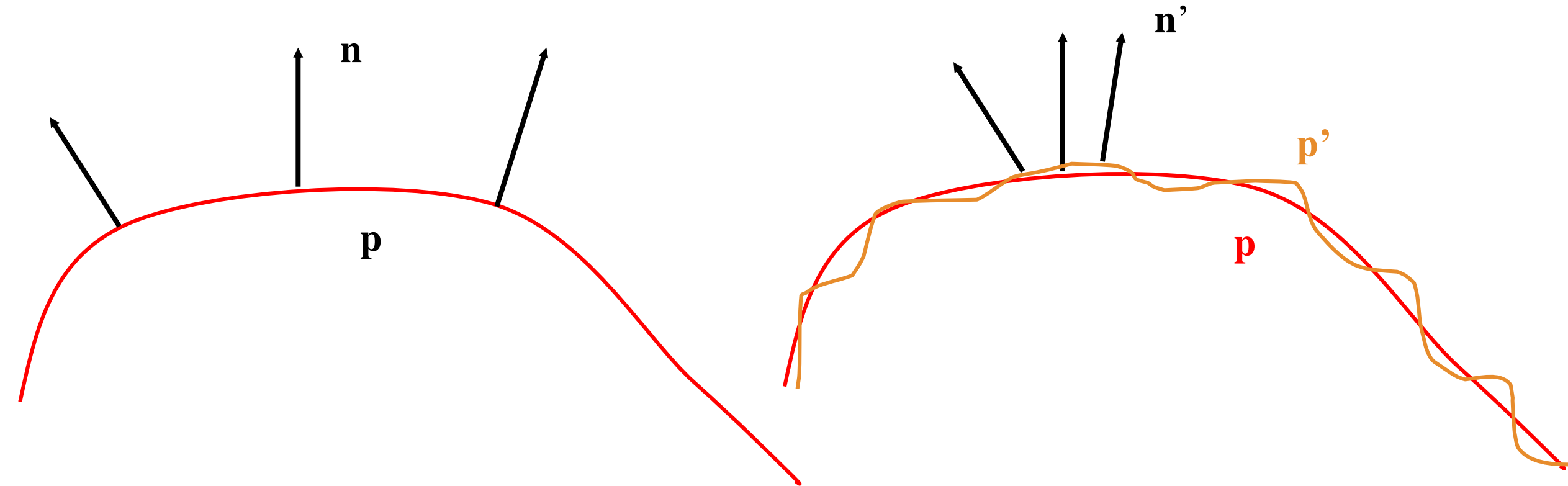


Bump Maps

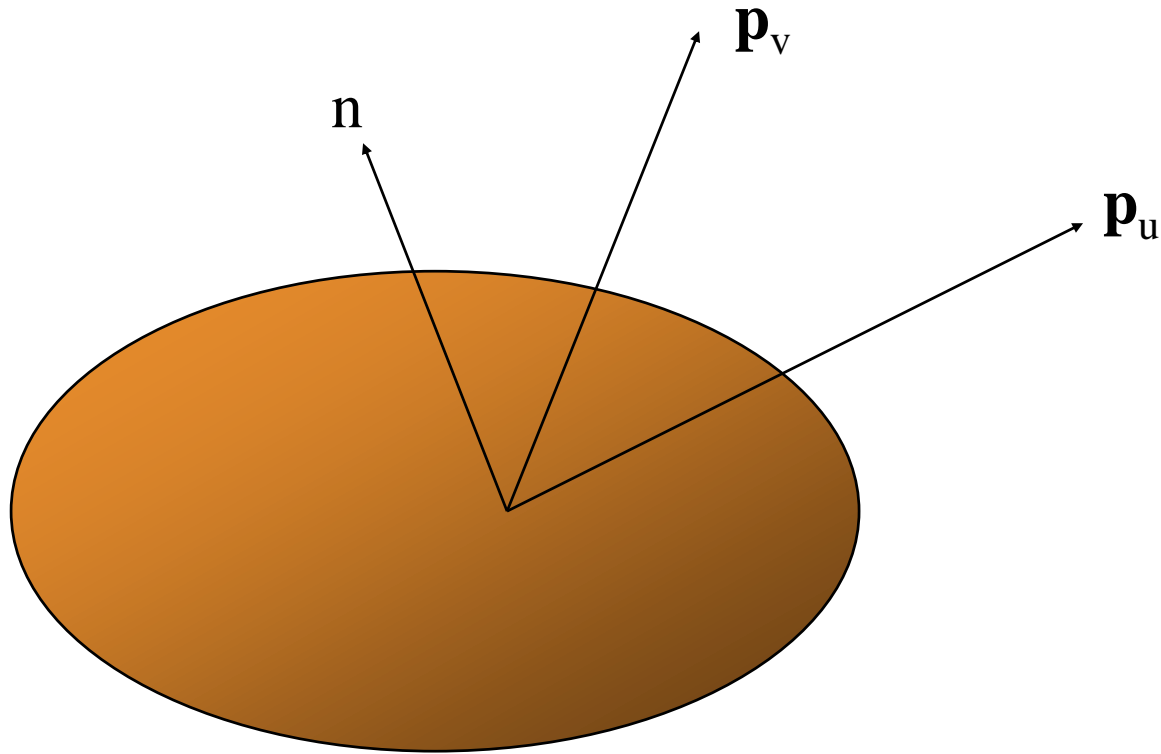
- Consider modeling an orange
- Texture map a photo of an orange onto a surface
 - Captures dimples
 - Will not be correct if we move viewer or light
 - We have shades of dimples rather than their correct orientation
- Ideally we need to perturb normal across surface of object and compute a new color at each interior point

Bump Mapping (Blinn)

- Consider a smooth surface



Tangent Plane



$$\mathbf{p}(u,v) = [x(u,v), y(u,v), z(u,v)]^T$$

$$\mathbf{p}_u = [\partial x / \partial u, \partial y / \partial u, \partial z / \partial u]^T$$

$$\mathbf{p}_v = [\partial x / \partial v, \partial y / \partial v, \partial z / \partial v]^T$$

$$\mathbf{n} = (\mathbf{p}_u \times \mathbf{p}_v) / |\mathbf{p}_u \times \mathbf{p}_v|$$

Displacement Function

$$\mathbf{p}' = \mathbf{p} + d(u,v) \mathbf{n}$$

- $d(u,v)$ is the bump or displacement function
- $|d(u,v)| \ll 1$



Base Texture

+



Bump Map
(Tangent Space)

=



Bump Mapping

수고하셨습니다