

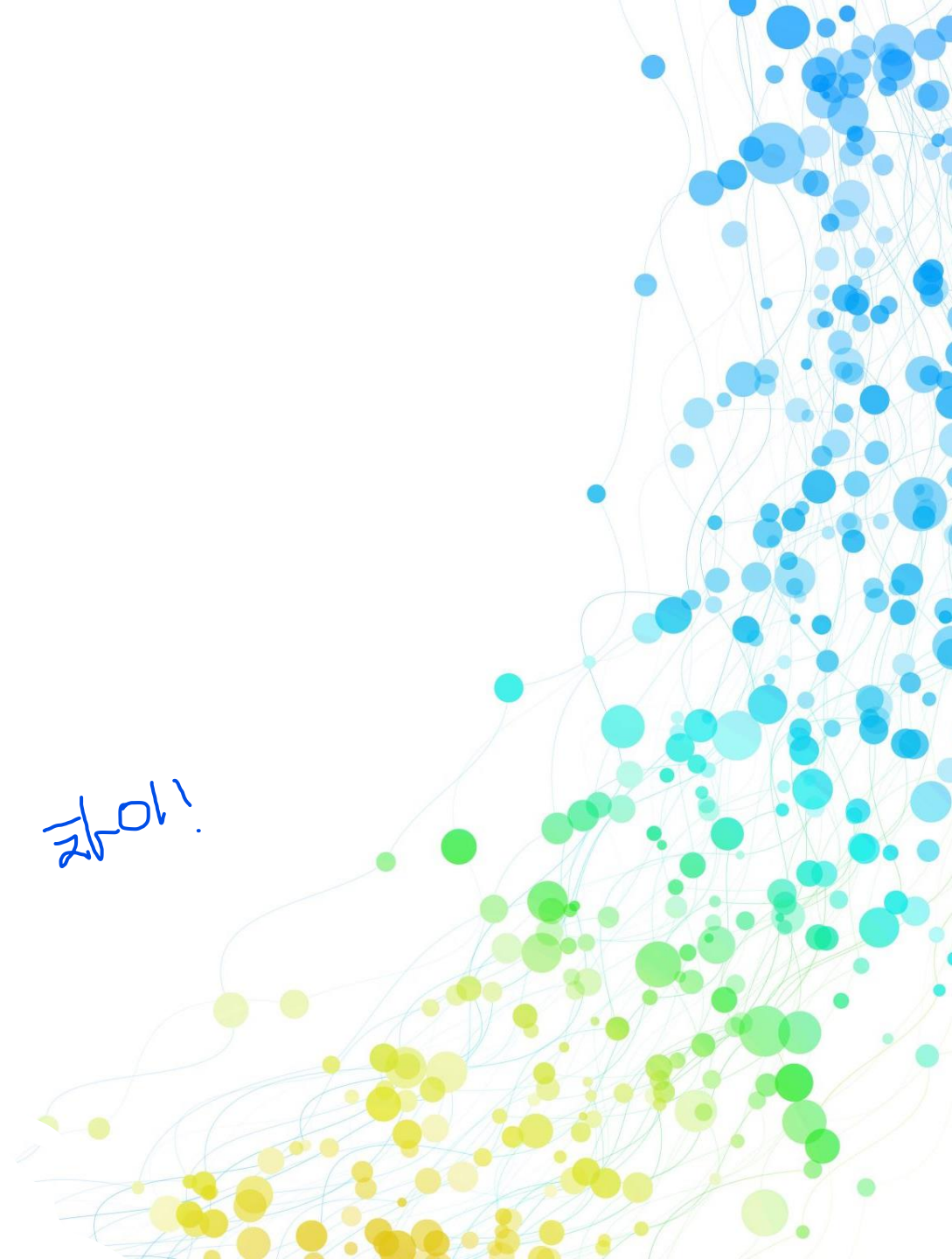
Environment Maps

13TH WEEK, 2022

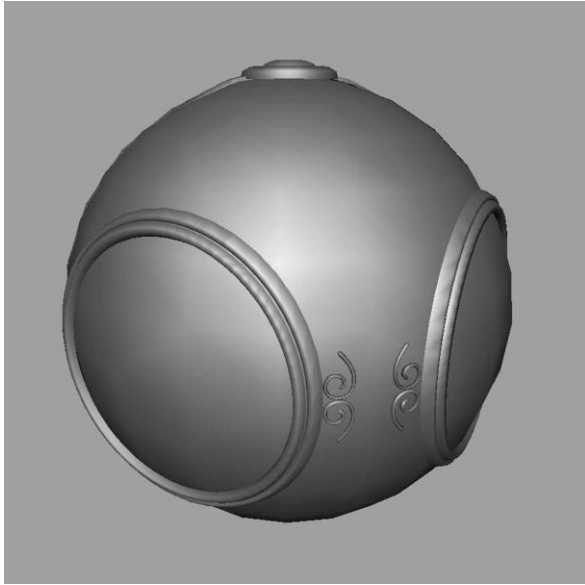
환경

환경

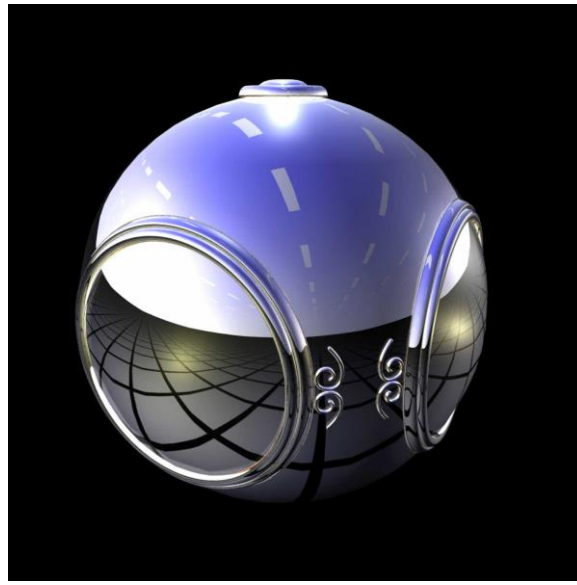
라이!



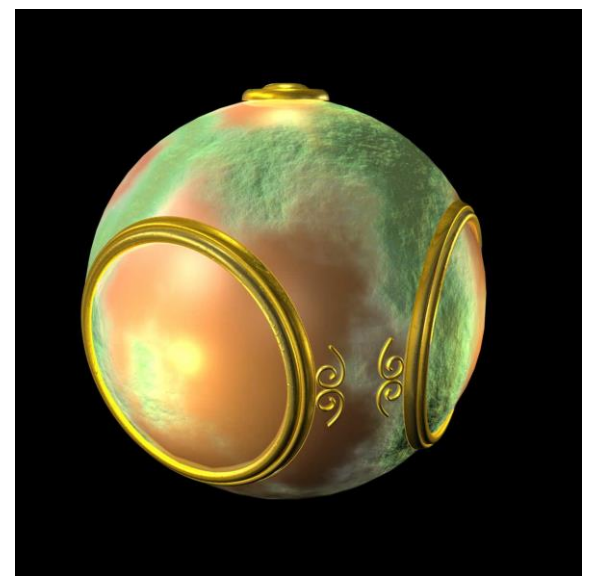
Mapping Variations



Gouraud shading



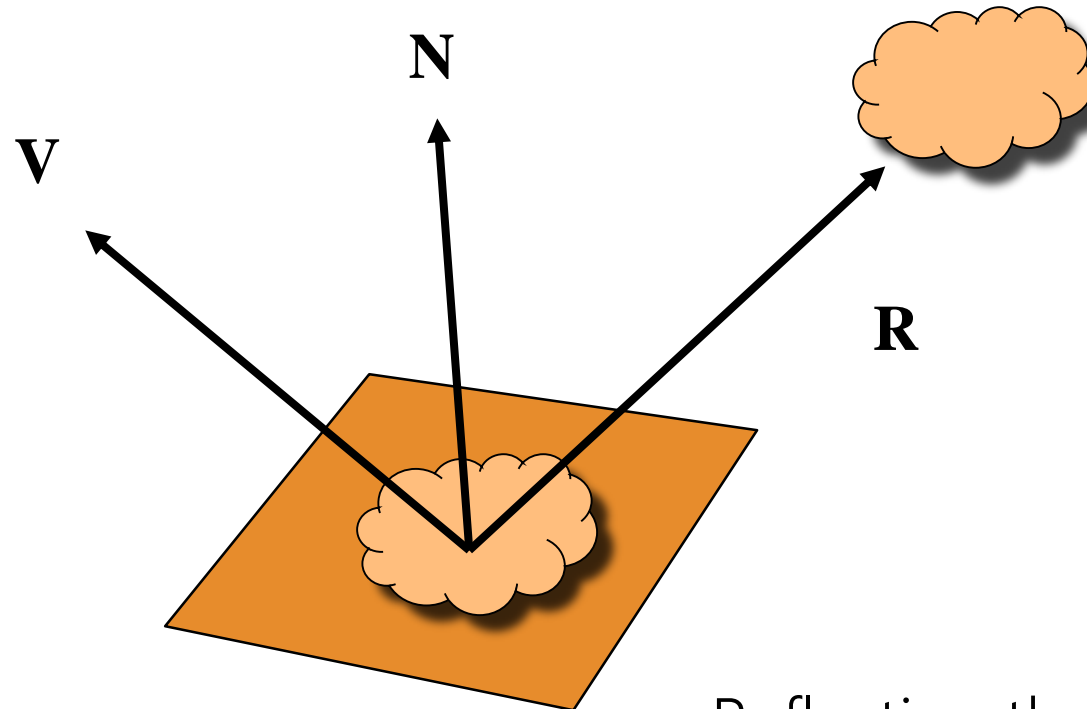
Environment mapping



Bump mapping

Environment Mapping

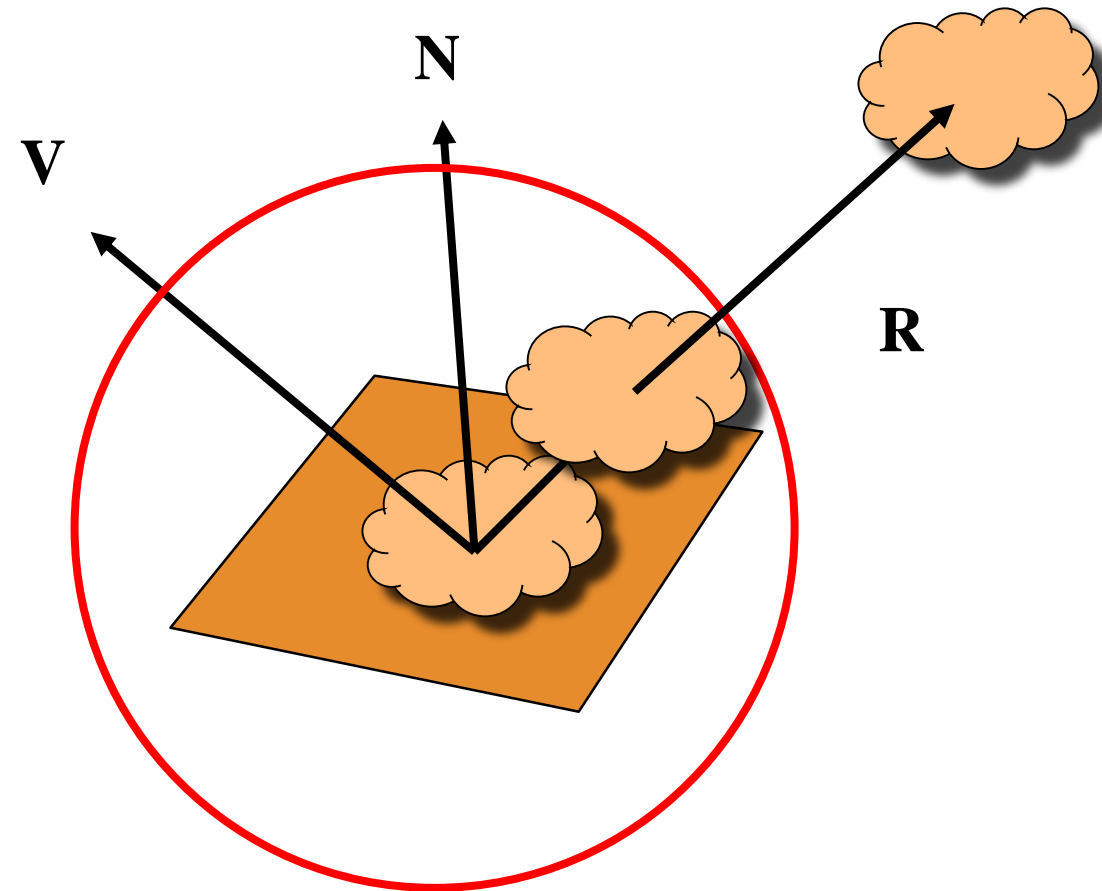
- Environmental (reflection) mapping is way to create the appearance of highly reflective surfaces without ray tracing which requires global calculations



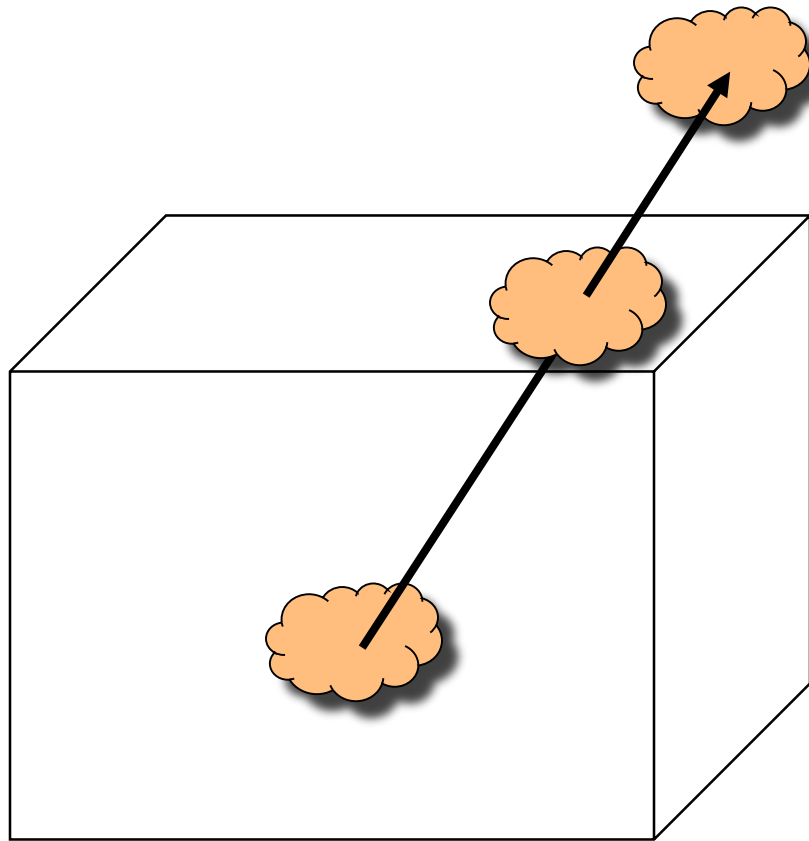
Reflecting the Environment

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

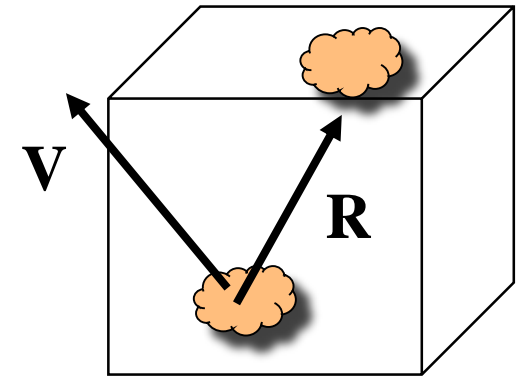


Cube Map



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

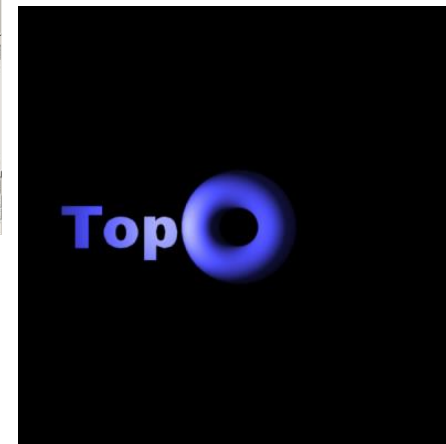
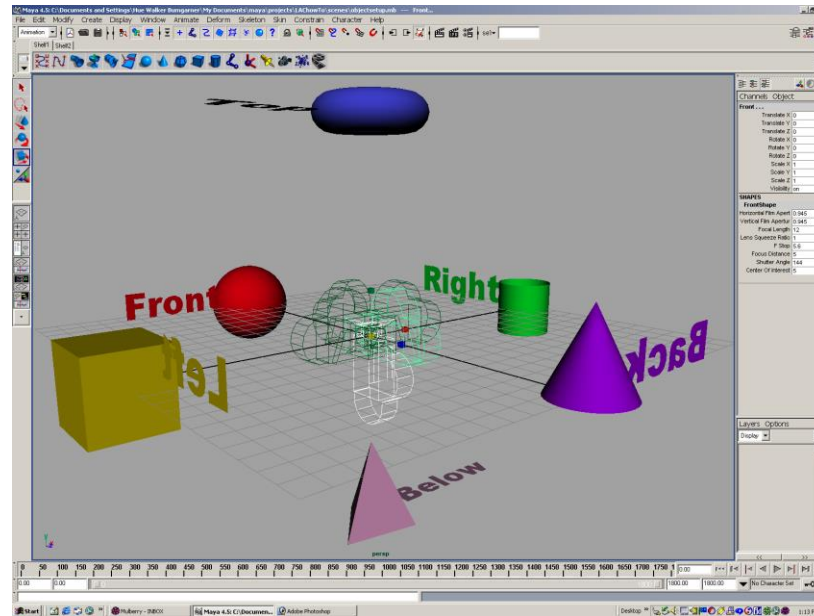
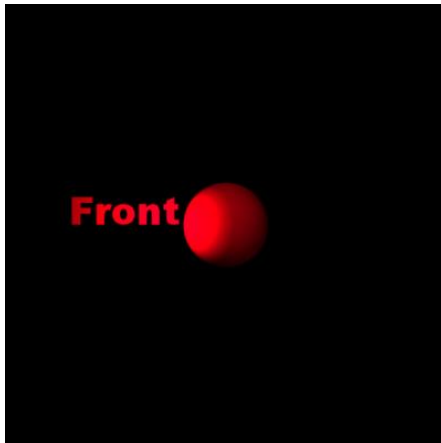
- Environment maps are usually computed in world coordinates which can differ from object coordinates because of the modeling matrix
 - May have to keep track of modeling matrix and pass it to the shaders as a uniform variable
- Can also use reflection map or refraction map for effects such as simulating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

Issues

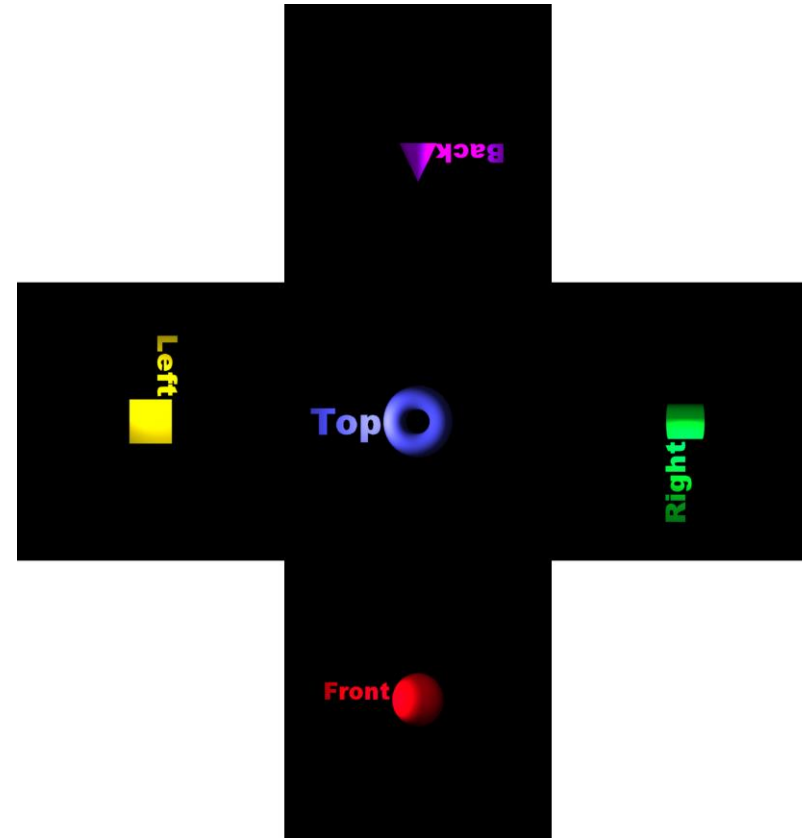
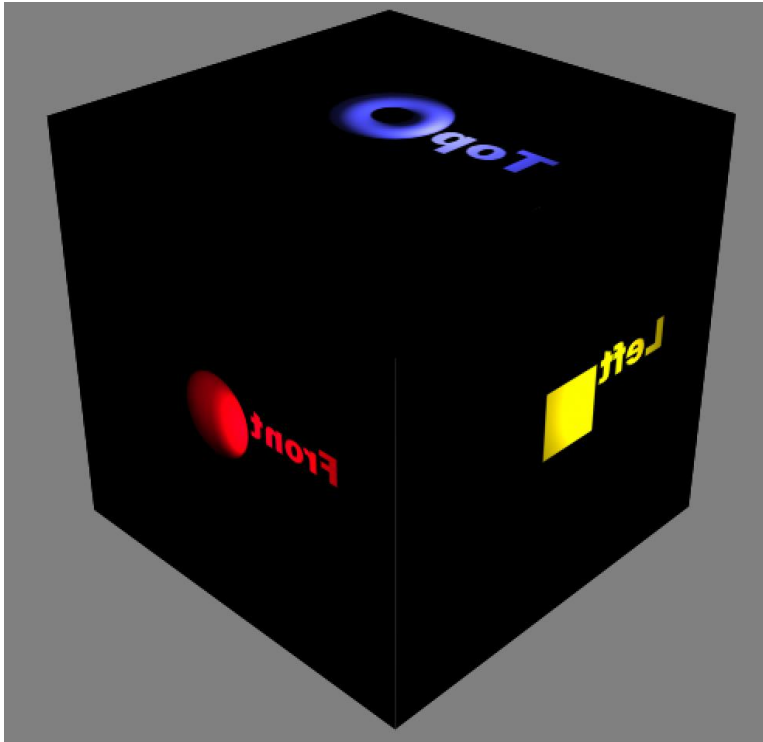
- Must assume environment is very far from object (equivalent to the difference between near and distant lights)
- Object cannot be concave (no self reflections possible)
- No reflections between objects
- Need a reflection map for each object
- Need a new map if viewer moves

Forming Cube Map

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



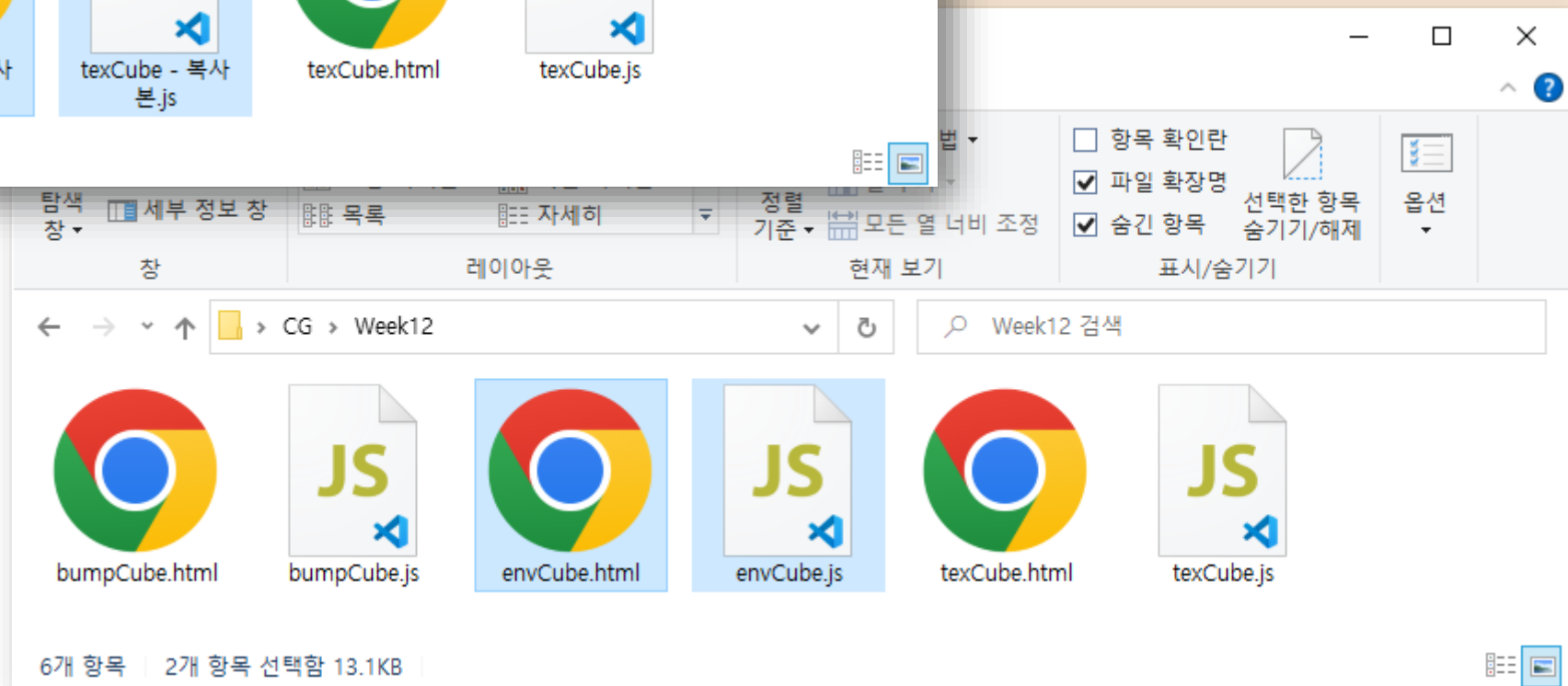
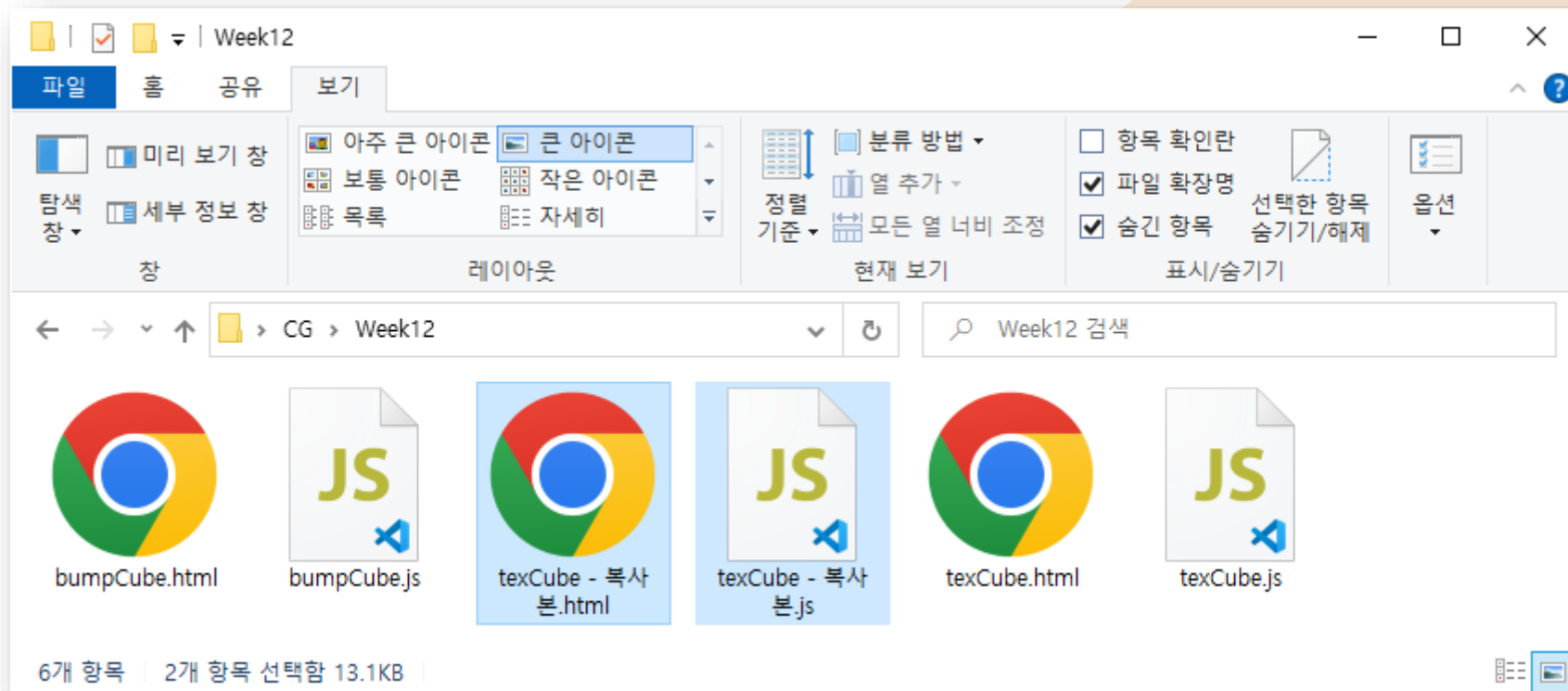
vs. Cube Image



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

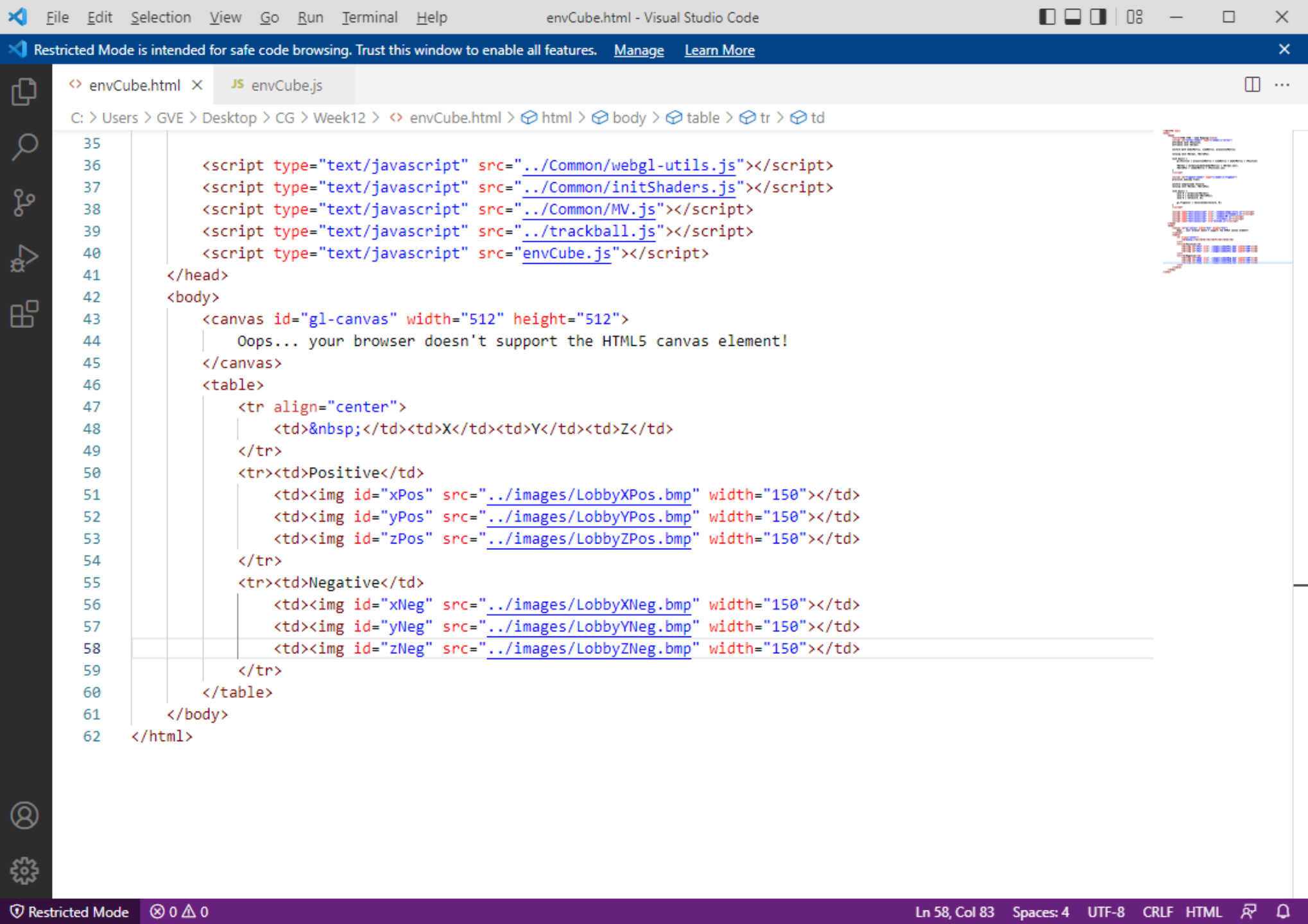


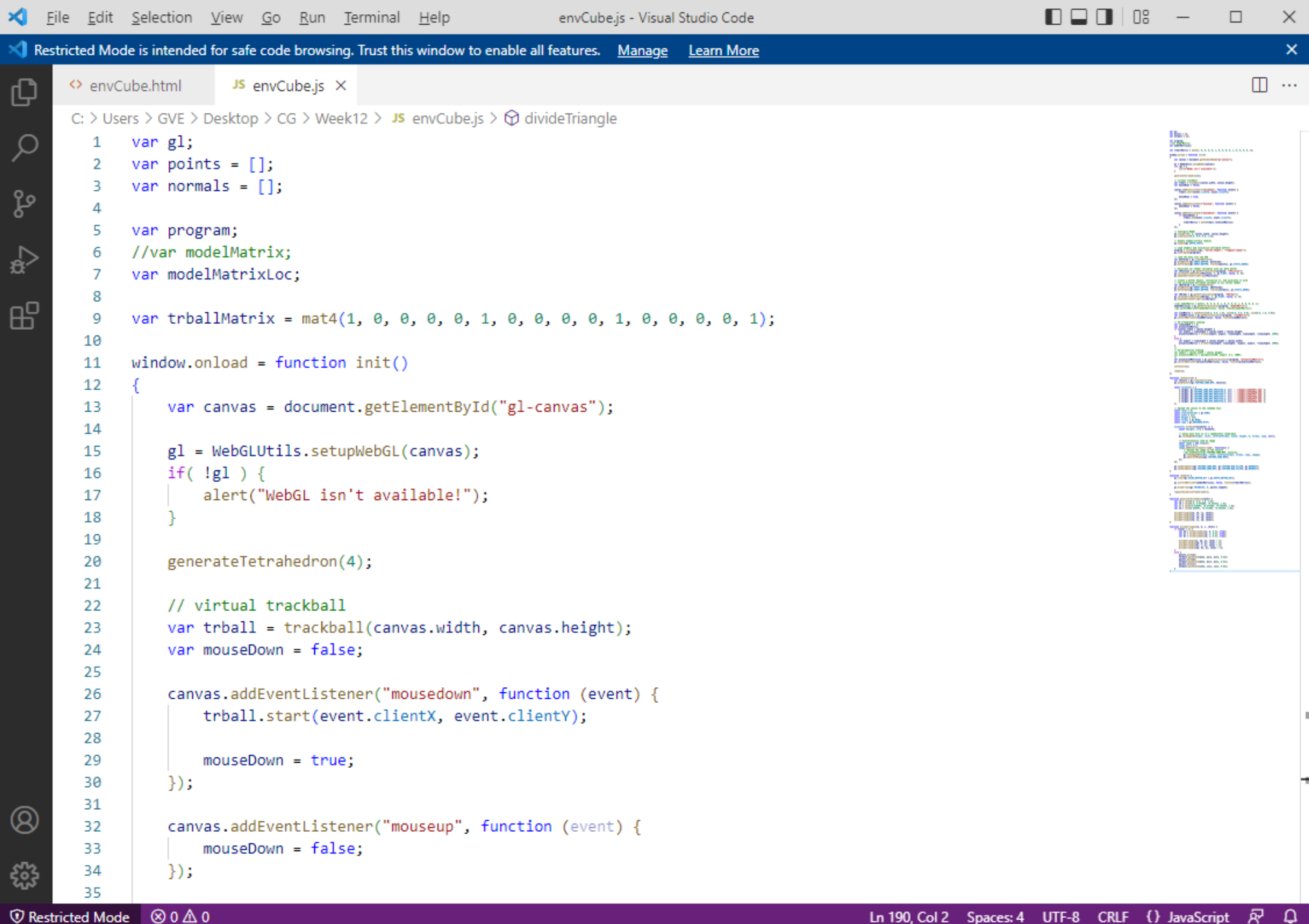
envCube.html x JS envCube.js

C: > Users > GVE > Desktop > CG > Week12 > <> envCube.html > html > body > table > tr > td

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>학번 이름 - Cube Mapping</title>
5          <script id="vertex-shader" type="x-shader/x-vertex">
6              attribute vec4 vPosition;
7              attribute vec4 vNormal;
8
9              uniform mat4 modelMatrix, viewMatrix, projectionMatrix;
10
11              varying vec3 fNormal, fWorldPos;
12
13              void main() {
14                  gl_Position = projectionMatrix * viewMatrix * modelMatrix * vPosition;
15
16                  fNormal = normalize(mat3(modelMatrix) * vNormal.xyz);
17                  fWorldPos = (modelMatrix * vPosition).xyz;
18              }
19          </script>
20
21          <script id="fragment-shader" type="x-shader/x-fragment">
22              precision mediump float;
23
24              uniform samplerCube texture;
25              varying vec3 fNormal, fWorldPos;
26
27              void main() {
28                  vec3 N = normalize(fNormal);
29                  vec3 V = normalize(-fWorldPos);
30                  vec3 R = reflect(V, N);
31
32                  gl_FragColor = textureCube(texture, R);
33              }
34          </script>
35
```







JS envCube.js X



```

1 #!/usr/bin/perl
2
3 use strict;
4 use warnings;
5
6 my $input = "input.txt";
7 my $output = "output.txt";
8
9 # Open input and output files
10 my $in = open(my $fh, "<$input") or die "Could not open $input: $!";
11 my $out = open(my $fo, ">$output") or die "Could not open $output: $!";
12
13 # Read input file line by line
14 while (my $line = <$fh) {
15     # Skip empty lines
16     next if $line =~ /^$/;
17
18     # Process each line
19     my @fields = split /\s+/, $line;
20
21     # Example processing: calculate sum of numbers
22     my $sum = 0;
23     for my $field (@fields) {
24         if ($field =~ /^-?\d+$/) {
25             $sum += $field;
26         }
27     }
28
29     # Write result to output file
30     print $fo "$sum\n";
31 }
32
33 # Close files
34 close $in;
35 close $fo;
36
37 # End of script
38

```

JS envCube.js X


```

1  # Import the necessary libraries
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.model_selection import train_test_split
8  from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
9  from sklearn.svm import SVC
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.linear_model import LogisticRegression
12 from sklearn.neighbors import KNeighborsClassifier
13 from sklearn.tree import DecisionTreeClassifier
14 from sklearn.metrics import roc_auc_score
15 from sklearn.metrics import roc_curve
16 from sklearn.metrics import precision_recall_curve
17 from sklearn.metrics import average_precision_score
18 from sklearn.metrics import f1_score
19 from sklearn.metrics import recall_score
20 from sklearn.metrics import precision_score
21 from sklearn.metrics import log_loss
22 from sklearn.metrics import brier_score_function
23 from sklearn.metrics import jaccard_index_score
24 from sklearn.metrics import hamming_loss
25 from sklearn.metrics import cohen_kappa_score
26 from sklearn.metrics import matthews_correlation_coefficient
27 from sklearn.metrics import confusion_matrix
28 from sklearn.metrics import classification_report
29 from sklearn.metrics import accuracy_score
30 from sklearn.metrics import confusion_matrix
31 from sklearn.metrics import classification_report
32 from sklearn.metrics import accuracy_score
33 from sklearn.metrics import confusion_matrix
34 from sklearn.metrics import classification_report
35 from sklearn.metrics import accuracy_score
36 from sklearn.metrics import confusion_matrix
37 from sklearn.metrics import classification_report
38 from sklearn.metrics import accuracy_score
39 from sklearn.metrics import confusion_matrix
40 from sklearn.metrics import classification_report
41 from sklearn.metrics import accuracy_score
42 from sklearn.metrics import confusion_matrix
43 from sklearn.metrics import classification_report
44 from sklearn.metrics import accuracy_score
45 from sklearn.metrics import confusion_matrix
46 from sklearn.metrics import classification_report
47 from sklearn.metrics import accuracy_score
48 from sklearn.metrics import confusion_matrix
49 from sklearn.metrics import classification_report
50 from sklearn.metrics import accuracy_score
51 from sklearn.metrics import confusion_matrix
52 from sklearn.metrics import classification_report
53 from sklearn.metrics import accuracy_score
54 from sklearn.metrics import confusion_matrix
55 from sklearn.metrics import classification_report
56 from sklearn.metrics import accuracy_score
57 from sklearn.metrics import confusion_matrix
58 from sklearn.metrics import classification_report
59 from sklearn.metrics import accuracy_score
60 from sklearn.metrics import confusion_matrix
61 from sklearn.metrics import classification_report
62 from sklearn.metrics import accuracy_score
63 from sklearn.metrics import confusion_matrix
64 from sklearn.metrics import classification_report
65 from sklearn.metrics import accuracy_score
66 from sklearn.metrics import confusion_matrix
67 from sklearn.metrics import classification_report
68 from sklearn.metrics import accuracy_score
69 from sklearn.metrics import confusion_matrix
70 from sklearn.metrics import classification_report
71 from sklearn.metrics import accuracy_score
72 from sklearn.metrics import confusion_matrix
73 from sklearn.metrics import classification_report
74 from sklearn.metrics import accuracy_score
75 from sklearn.metrics import confusion_matrix
76 from sklearn.metrics import classification_report
77 from sklearn.metrics import accuracy_score
78 from sklearn.metrics import confusion_matrix
79 from sklearn.metrics import classification_report
80 from sklearn.metrics import accuracy_score
81 from sklearn.metrics import confusion_matrix
82 from sklearn.metrics import classification_report
83 from sklearn.metrics import accuracy_score
84 from sklearn.metrics import confusion_matrix
85 from sklearn.metrics import classification_report
86 from sklearn.metrics import accuracy_score
87 from sklearn.metrics import confusion_matrix
88 from sklearn.metrics import classification_report
89 from sklearn.metrics import accuracy_score
90 from sklearn.metrics import confusion_matrix
91 from sklearn.metrics import classification_report
92 from sklearn.metrics import accuracy_score
93 from sklearn.metrics import confusion_matrix
94 from sklearn.metrics import classification_report
95 from sklearn.metrics import accuracy_score
96 from sklearn.metrics import confusion_matrix
97 from sklearn.metrics import classification_report
98 from sklearn.metrics import accuracy_score
99 from sklearn.metrics import confusion_matrix
100 from sklearn.metrics import classification_report

```


envCube.html JS envCube.js X

C: > Users > GVE > Desktop > CG > Week12 > JS envCube.js > divideTriangle

```
104     render();
105 };
106
107 function setTexture() {
108     var texture = gl.createTexture();
109     gl.bindTexture(gl.TEXTURE_CUBE_MAP, texture);
110
111     const faceInfos = [
112         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_X, url: '../images/LobbyXPos.bmp' },
113         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_X, url: '../images/LobbyXNeg.bmp' },
114         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Y, url: '../images/LobbyYPos.bmp' },
115         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Y, url: '../images/LobbyYNeg.bmp' },
116         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Z, url: '../images/LobbyZPos.bmp' },
117         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Z, url: '../images/LobbyZNeg.bmp' },
118     ];
119
120     // Upload the canvas to the cubemap face
121     const level = 0;
122     const internalFormat = gl.RGBA;
123     const width = 512;
124     const height = 512;
125     const format = gl.RGBA;
126     const type = gl.UNSIGNED_BYTE;
127
128     faceInfos.forEach((faceInfo) => {
129         const {target, url} = faceInfo;
130
131         // Setup each face so it's immediately renderable
132         gl.texImage2D(target, level, internalFormat, width, height, 0, format, type, null);
133
134         // Asynchronously load an image
135         const image = new Image();
136         image.src = url;
137         image.addEventListener('load', function() {
138             // Upload the image to the texture
```

```
104     render();
105 };
106
107 function setTexture() {
108     var texture = gl.createTexture();
109     gl.bindTexture(gl.TEXTURE_CUBE_MAP, texture);
110
111     const faceInfos = [
112         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_X, url: '../images/LobbyXPos.bmp' },
113         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_X, url: '../images/LobbyXNeg.bmp' },
114         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Y, url: '../images/LobbyYPos.bmp' },
115         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Y, url: '../images/LobbyYNeg.bmp' },
116         { target: gl.TEXTURE_CUBE_MAP_POSITIVE_Z, url: '../images/LobbyZPos.bmp' },
117         { target: gl.TEXTURE_CUBE_MAP_NEGATIVE_Z, url: '../images/LobbyZNeg.bmp' },
118     ];
119
120     // Upload the canvas to the cubemap face
121     const level = 0;
122     const internalFormat = gl.RGBA;
123     const width = 512;
124     const height = 512;
125     const format = gl.RGBA;
126     const type = gl.UNSIGNED_BYTE;
127
128     faceInfos.forEach((faceInfo) => {
129         const {target, url} = faceInfo;
130
131         // Setup each face so it's immediately renderable
132         gl.texImage2D(target, level, internalFormat, width, height, 0, format, type, null);
133
134         // Asynchronously load an image
135         const image = new Image();
136         image.src = url;
137         image.addEventListener('load', function() {
138             // Upload the image to the texture
```

JS envCube.js X

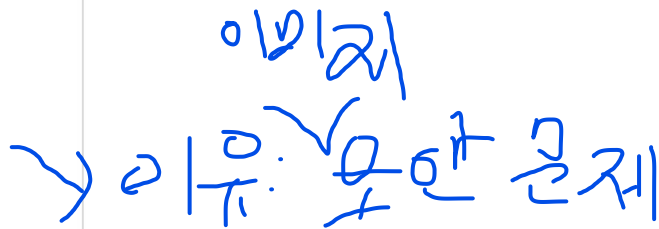
[illegible]

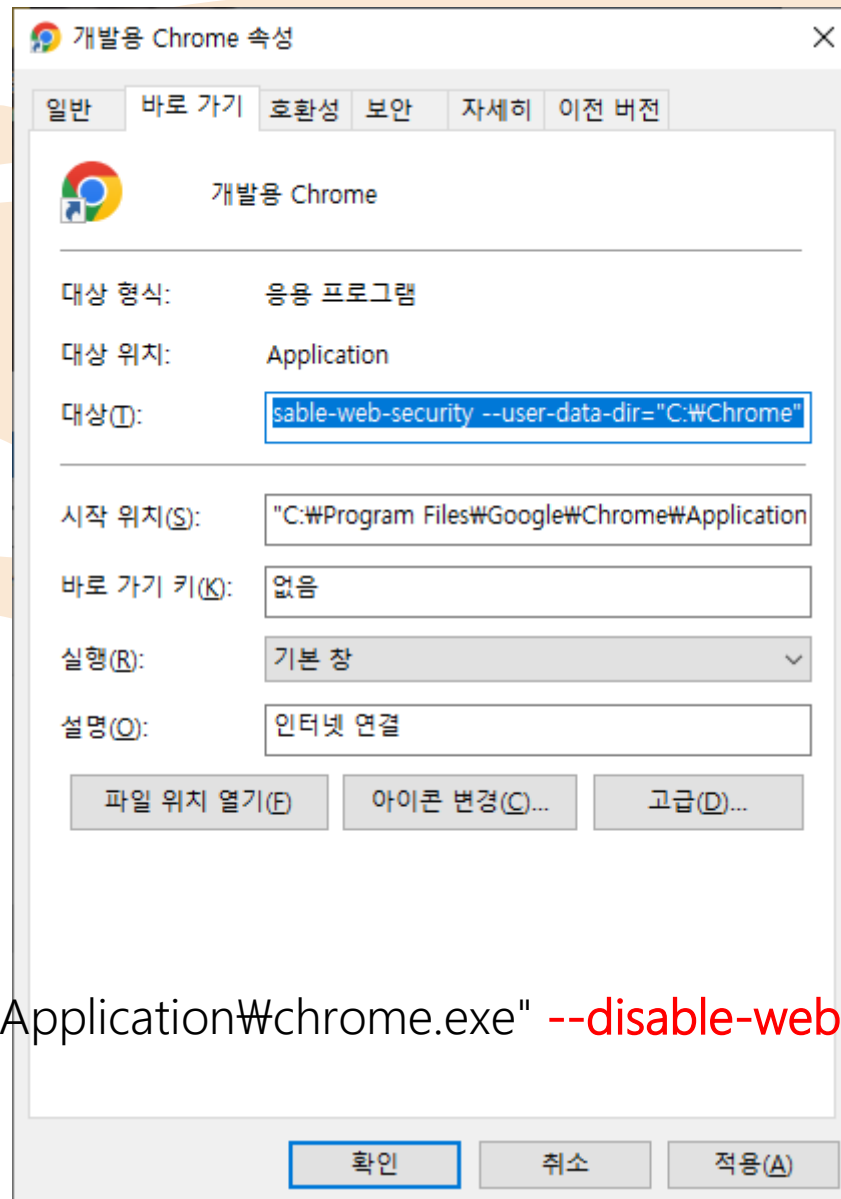
envCube.html JS envCube.js

C: > Users > GVE > Desktop > CG > Week12 > JS envCube.js > divideTriangle

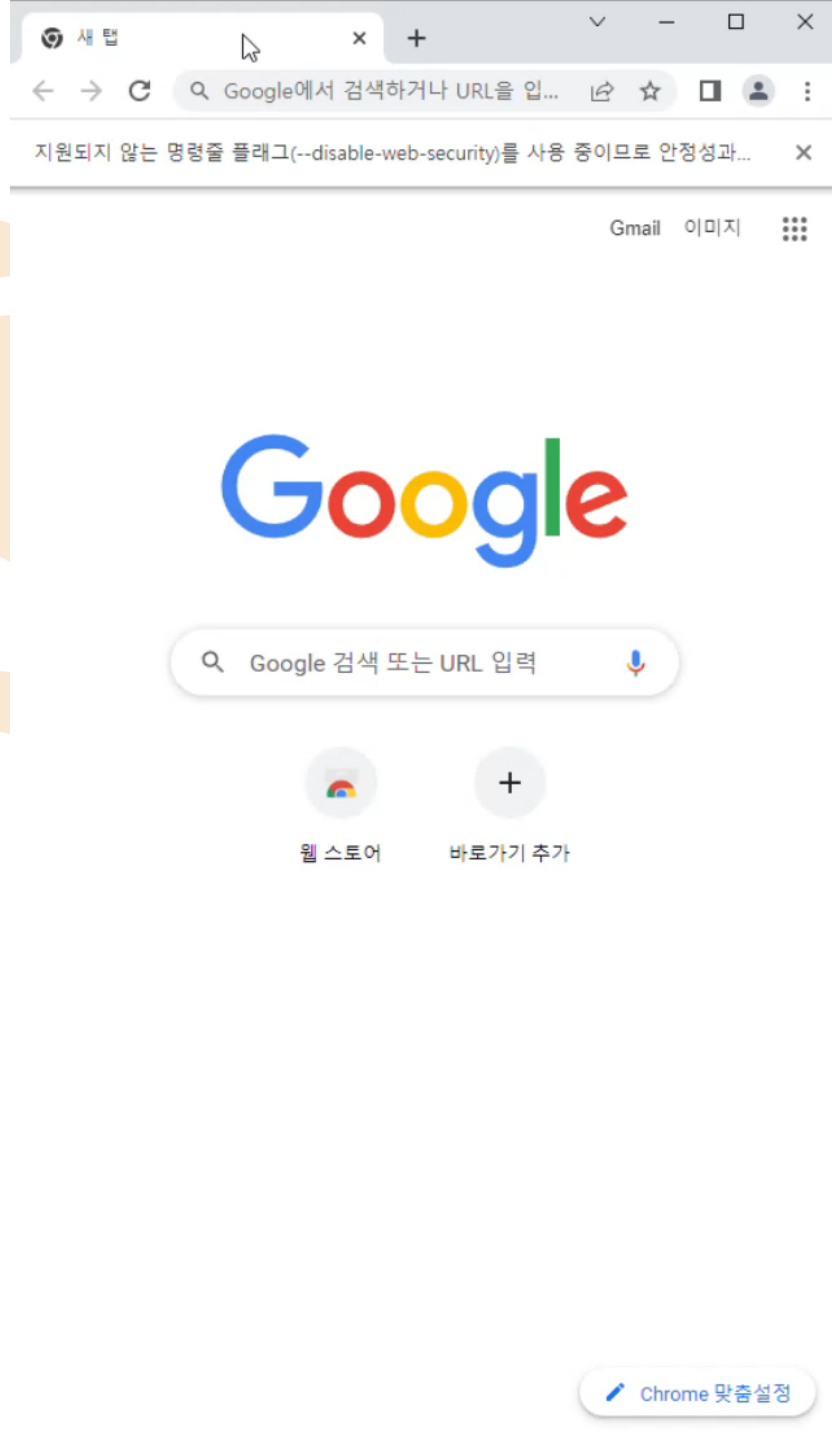
```
158
159 function generateTetrahedron(level) {
160     var va = vec4(0.0, 0.0, 1.0, 1.0);
161     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
162     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
163     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
164
165     divideTriangle(va, vb, vc, level);
166     divideTriangle(va, vc, vd, level);
167     divideTriangle(va, vd, vb, level);
168     divideTriangle(vd, vc, vb, level);
169 }
170
171 function divideTriangle(a, b, c, level) {
172     if (level > 1) {
173         var ab = normalize(mix(a, b, 0.5), true);
174         var ac = normalize(mix(a, c, 0.5), true);
175         var bc = normalize(mix(b, c, 0.5), true);
176
177         divideTriangle(a, ab, ac, level - 1);
178         divideTriangle(ab, b, bc, level - 1);
179         divideTriangle(bc, c, ac, level - 1);
180         divideTriangle(ab, bc, ac, level - 1);
181     }
182     else {
183         points.push(a);
184         normals.push(vec4(a[0], a[1], a[2], 0.0));
185         points.push(b);
186         normals.push(vec4(b[0], b[1], b[2], 0.0));
187         points.push(c);
188         normals.push(vec4(c[0], c[1], c[2], 0.0));
189     }
190 }
191
```

```
158
159 function generateTetrahedron(level) {
160     var va = vec4(0.0, 0.0, 1.0, 1.0);
161     var vb = vec4(0.0, 0.942809, -0.333333, 1.0);
162     var vc = vec4(-0.816497, -0.471405, -0.333333, 1.0);
163     var vd = vec4(0.816497, -0.471405, -0.333333, 1.0);
164
165     divideTriangle(va, vb, vc, level);
166     divideTriangle(va, vc, vd, level);
167     divideTriangle(va, vd, vb, level);
168     divideTriangle(vd, vc, vb, level);
169 }
170
171 function divideTriangle(a, b, c, level) {
172     if (level > 1) {
173         var ab = normalize(mix(a, b, 0.5), true);
174         var ac = normalize(mix(a, c, 0.5), true);
175         var bc = normalize(mix(b, c, 0.5), true);
176
177         divideTriangle(a, ab, ac, level - 1);
178         divideTriangle(ab, b, bc, level - 1);
179         divideTriangle(bc, c, ac, level - 1);
180         divideTriangle(ab, bc, ac, level - 1);
181     }
182     else {
183         points.push(a);
184         normals.push(vec4(a[0], a[1], a[2], 0.0));
185         points.push(b);
186         normals.push(vec4(b[0], b[1], b[2], 0.0));
187         points.push(c);
188         normals.push(vec4(c[0], c[1], c[2], 0.0));
189     }
190 }
191
```



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



envCube.html - Visual Studio Code

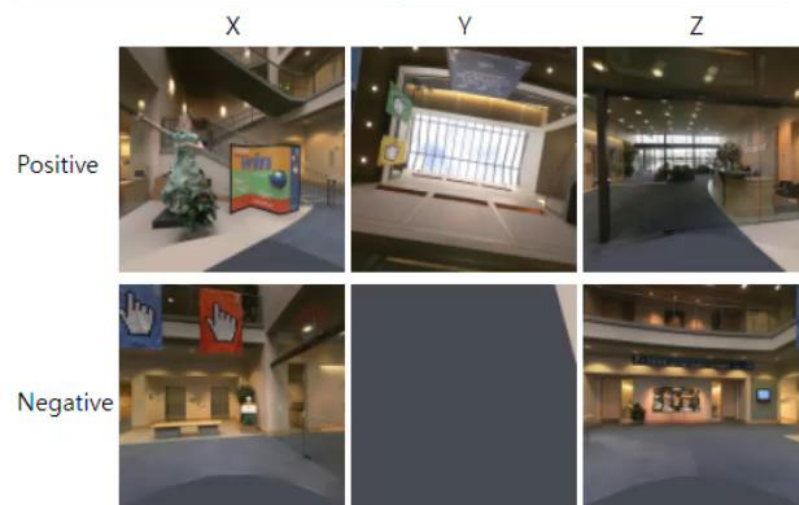
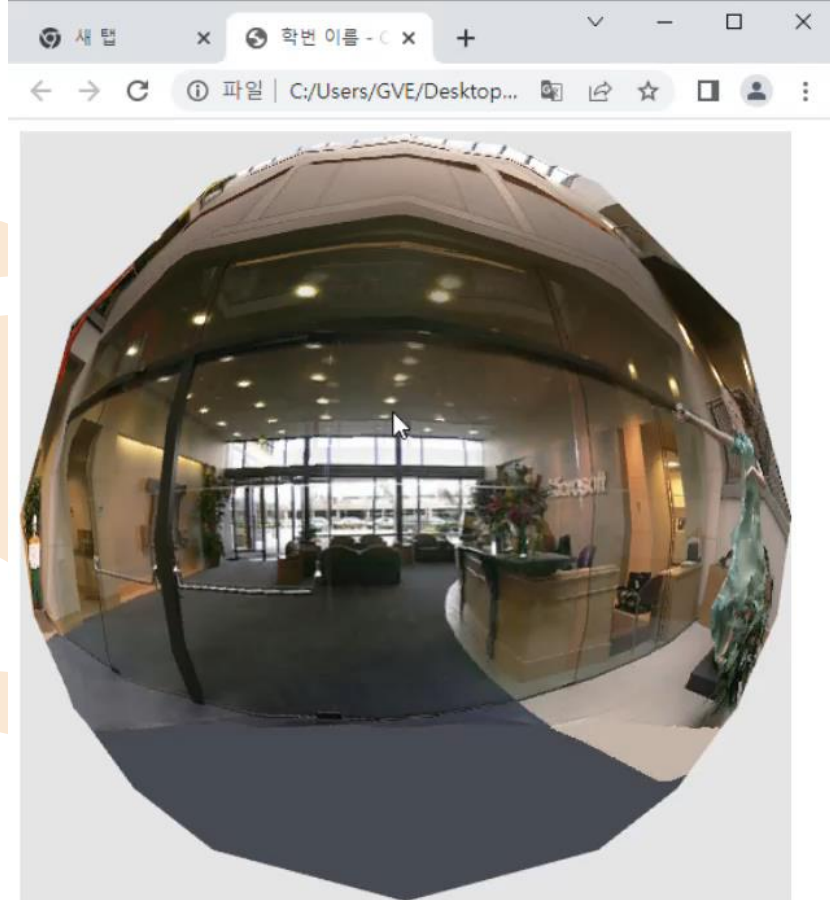
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

envCube.html x JS envCube.js

C: > Users > GVE > Desktop > CG > Week12 > <> envCube.html > html > head > script#vertex-shader

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>학번 이름 - Cube Mapping</title>
5      <script id="vertex-shader" type="x-shader/x-vertex">
6        attribute vec4 vPosition;
7        attribute vec4 vNormal;
8
9        uniform mat4 modelMatrix, viewMatrix, projectionMatrix;
10
11        varying vec3 fNormal, fWorldPos;
12
13        void main() {
14          gl_Position = projectionMatrix * viewMatrix * modelMatrix * vPosition;
15
16          //fNormal = normalize(mat3(modelMatrix) * vNormal.xyz);
17          //fWorldPos = (modelMatrix * vPosition).xyz;
18          fNormal = vNormal.xyz;
19          fWorldPos = vPosition.xyz;
20        }
21      </script>
22
23      <script id="fragment-shader" type="x-shader/x-fragment">
24        precision mediump float;
25
26        uniform samplerCube texture;
27        varying vec3 fNormal, fWorldPos;
28
29        void main() {
30          vec3 N = normalize(fNormal);
31          vec3 V = normalize(-fWorldPos);
32          vec3 R = reflect(V, N);
33
34          gl_FragColor = textureCube(texture, R);
35        }
23
```

Restricted Mode 0 0 Tab Moves Focus Ln 19, Col 39 Spaces: 4 UTF-8 CRLF HTML

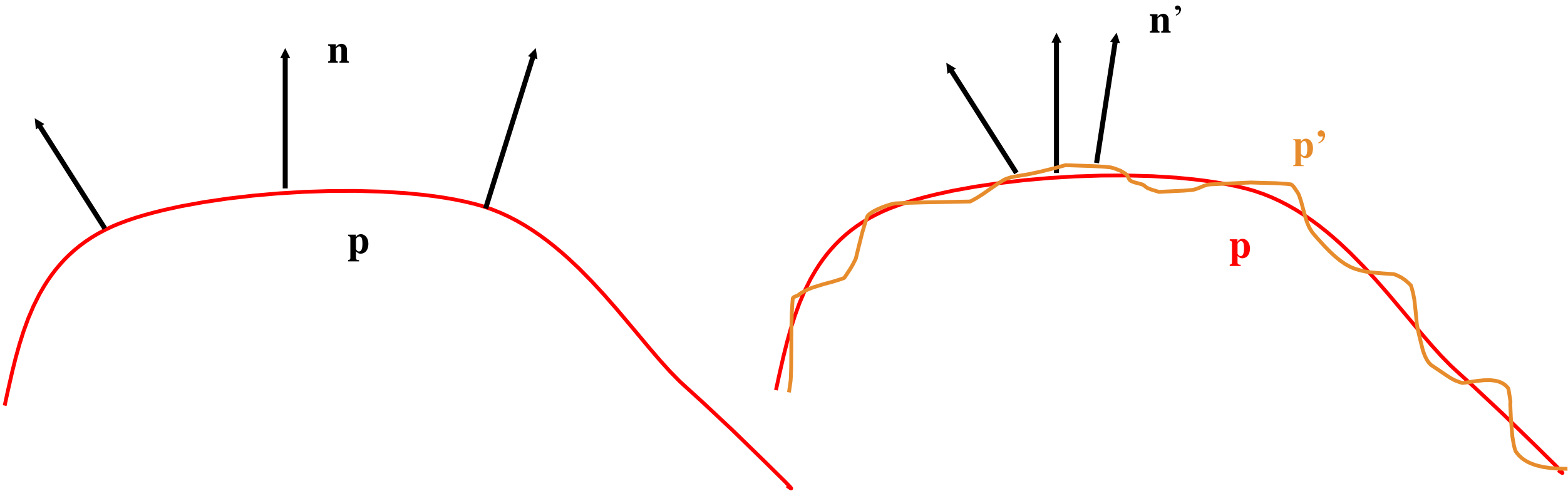


Bump Maps

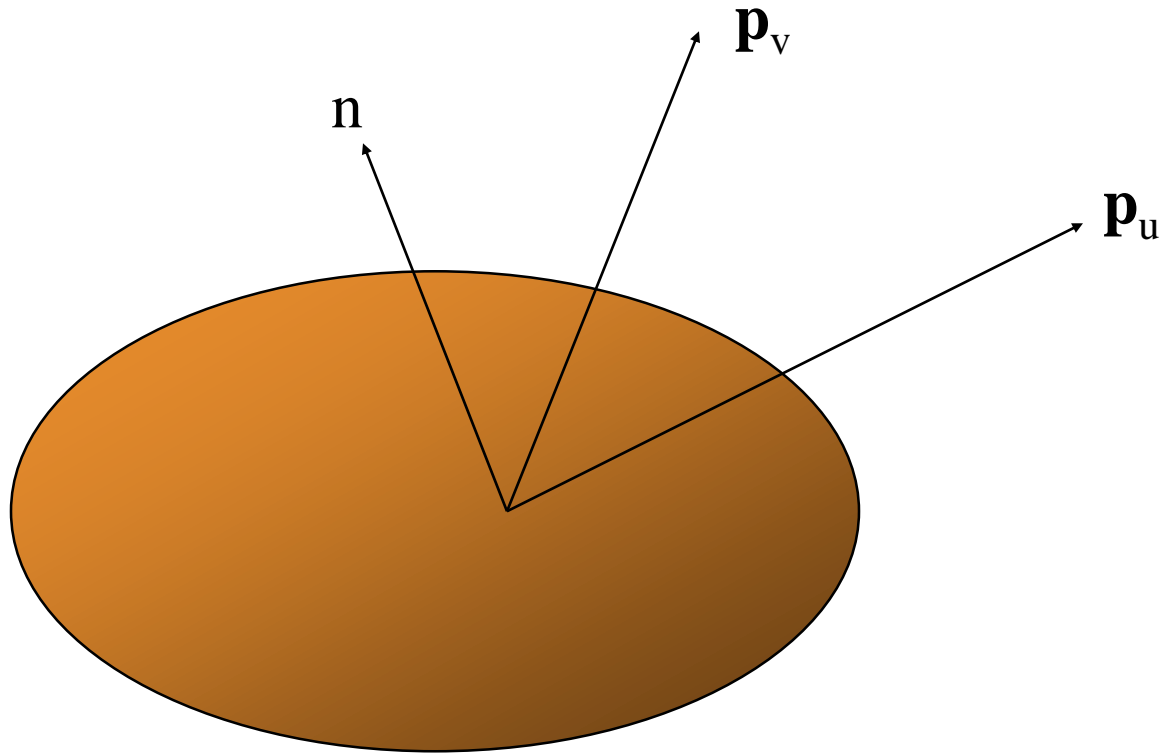
- Consider modeling an orange
- Texture map a photo of an orange onto a surface
 - Captures dimples
 - Will not be correct if we move viewer or light
 - We have shades of dimples rather than their correct orientation
- Ideally we need to perturb normal across surface of object and compute a new color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

Bump Mapping (Blinn)

- Consider a smooth surface



Tangent Plane



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

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

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

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

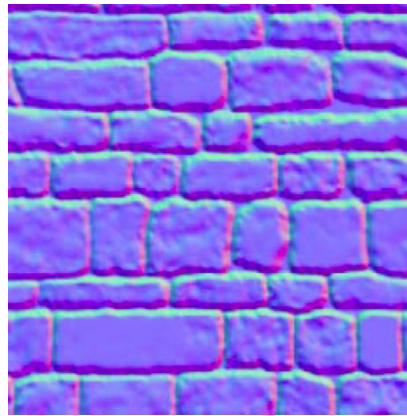
Displacement Function

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

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



Base Texture



Bump Map
(Tangent Space)



Bump Mapping

