CG Practice 9

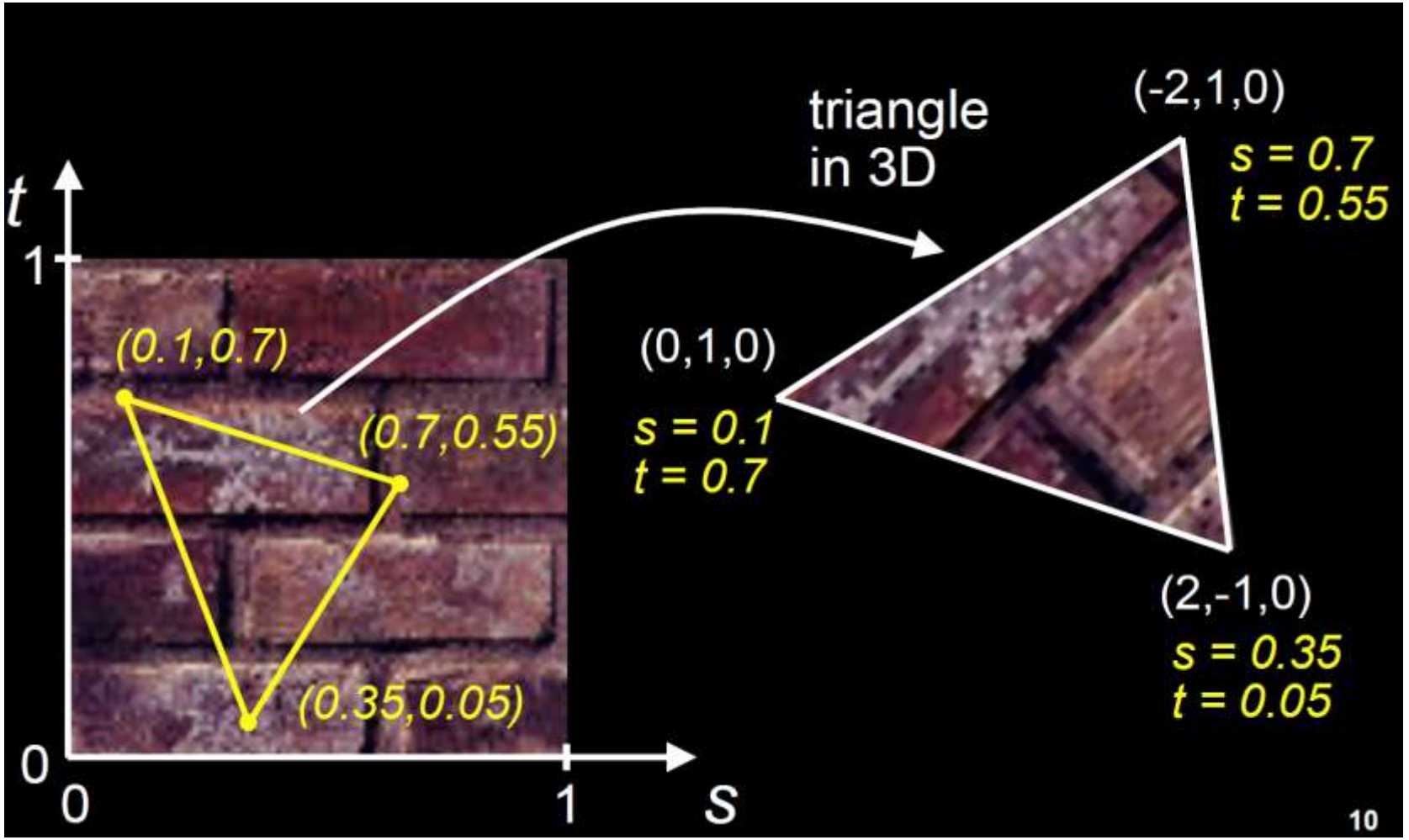
COLLEGE OF COMPUTING

HANYANG ERICA CAMPUS

Q YOUN HONG (홍규연)

Texture Mapping

Texture Mapping



OpenGL에서의 Texture Mapping

- OpenGL에서 Texture 적용하기
  1. Texture 정의하기
     - Texture로 쓸 이미지를 파일에서 읽거나 직접 생성
     - 프로그램 내부의 texture에 할당하기
     - Texturing을 활성화
  2. 각 물체의 vertex들의 texture coordinate을 결정
     - Texture coordinate function을 이용하여 mapping하거나, 외부적으로 읽기
  3. Texture parameter 결정 – wrapping, filtering

Texture Object 사용하기

1. Texture를 texture object에 지정함
2. Texture object를 binding함
3. Texture parameter (wrap mode, filter) 지정
4. Texturing을 활성화
5. 각각의 vertex에 texture coordinate 지정

Vertex Shader

- 각각의 vertex에 대해서 vertex shader는 rasterize할 output texture coordinate를 지정
- Vertex attribute

  = vertex position + (vertex color) + texture coordinate

```
in vec4 vPosition; //vertex position in object coordinates
in vec4 vColor;  //vertex color from application
in vec2 vTexCoord; //texture coordinate from application

out vec4 color; //output color to be interpolated
out vec2 texCoord; //output tex coordinate to be interpolated
```

Fragment Shader

- Texture를 실제로 적용하는 것은 Fragment processing에서 이루어진다. (in fragment shader)
- Texture들은 application에서 sampler 변수로 받음
  - Sampler1D, **Sampler2D**, Sampler3D, SamplerCube
- Sampler는 texture object로부터 texture coordinate에 해당하는 texture color를 반환함

```
in vec4 color;      //color from rasterizer
in vec2 texCoord;   //texure coordinate from rasterizer
uniform sampler2D uTexture; //texture object from application

void main()  {
    gl_FragColor = color * texture2D( uTexture, texCoord );
}
```

Texture Mapping을 위한 c++ 코드


```
GLuint textures;
glGenTextures( 1, &textures );

glActiveTexture( GL_TEXTURE0 );

glBindTexture( GL_TEXTURE_2D, textures );

glTexImage2D( GL_TEXTURE_2D, 0, GL_RGB, TextureSize,
   TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE, image );

glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
    GL_REPEAT );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
     GL_REPEAT );
glTexParameterf( GL_TEXTURE_2D,
      GL_TEXTURE_MAG_FILTER, GL_NEAREST );
glTexParameterf( GL_TEXTURE_2D,
      GL_TEXTURE_MIN_FILTER, GL_NEAREST );
```

Image Loading: stb_image.h

- 이미지 로딩을 위한 오픈 소스 라이브러리 (by Sean Barrett)
- Header-only file: can be found in GitHub

(https://github.com/nothings/stb/blob/master/stb_image.h)

- To use stb_image.h

```
#define STB_IMAGE_IMPLEMENTATION
#include "stb_image.h"
```

Stb_image.h파일을 변환시켜 header file을 source code를 포함한 cpp file로 변환

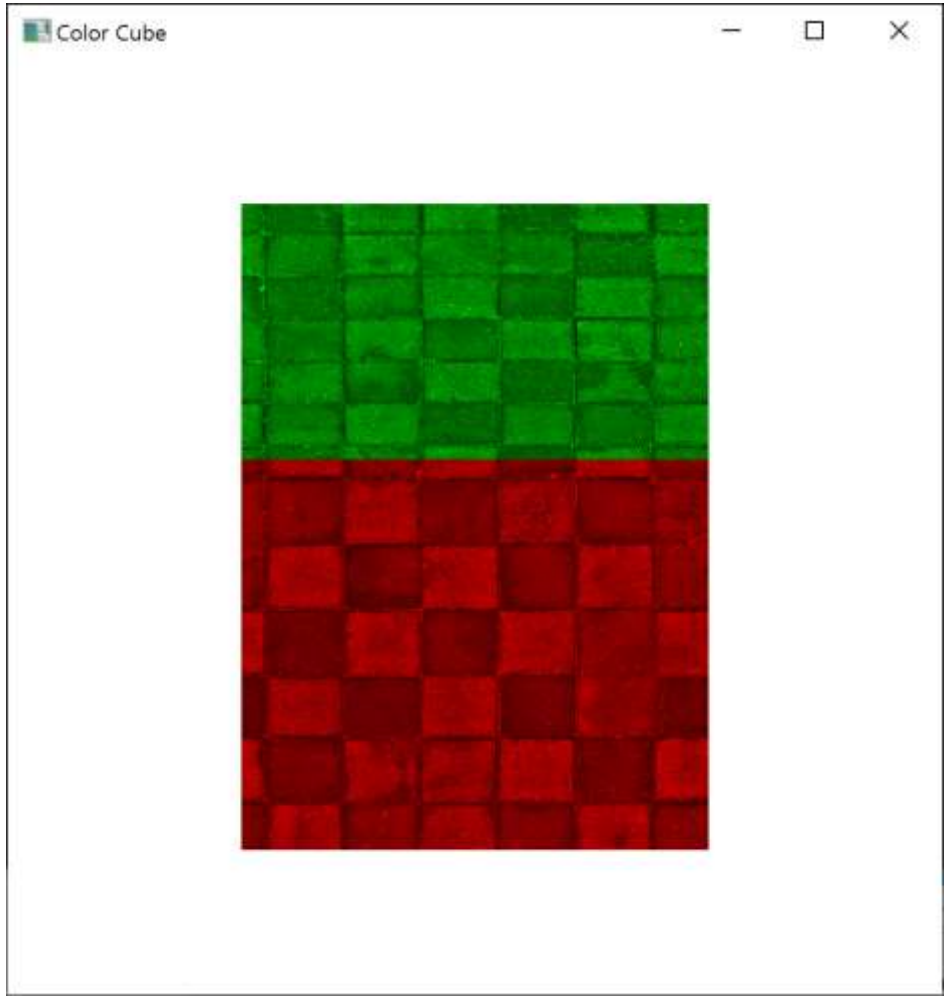
```
int texWidth, texHeight, texChannels;
unsigned char* data = stbi_load("wall.jpg", &texWidth, &texHeight, &texChannels, 0);
```

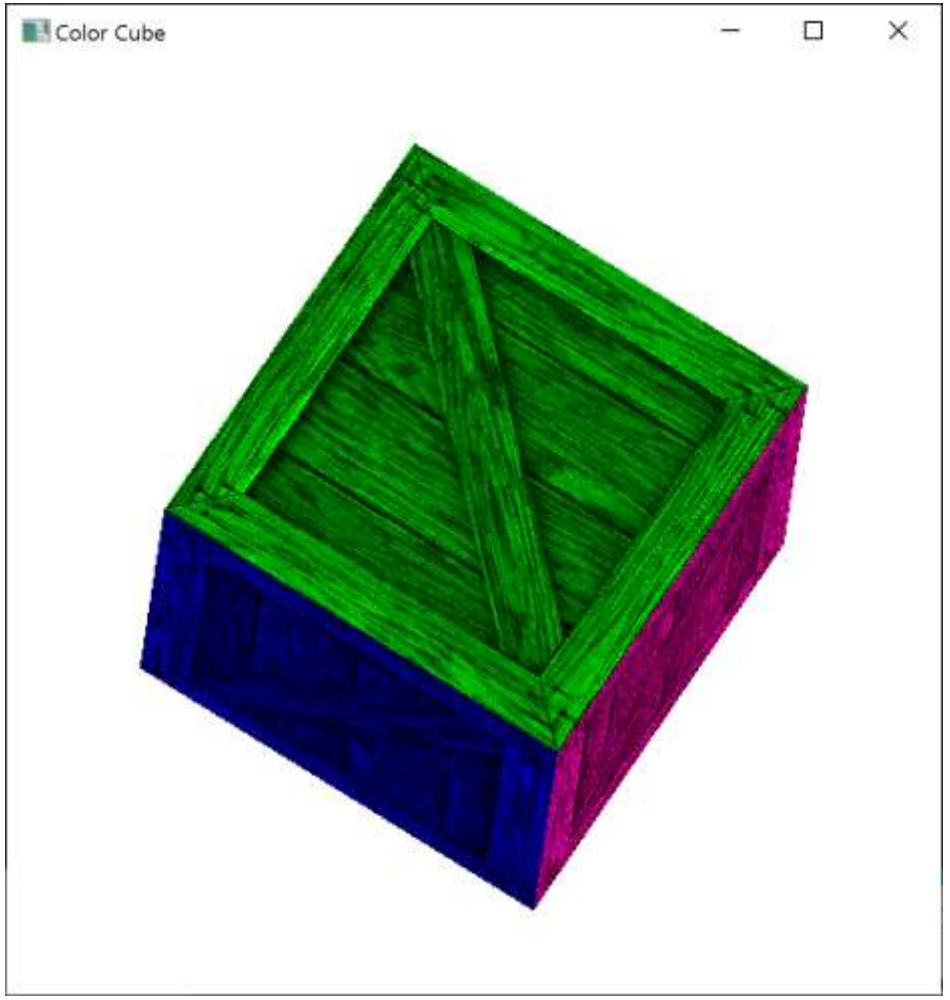
Wall.jpg 파일을 로딩하여 unsigned byte array에 저장

Loading Texture Image Code

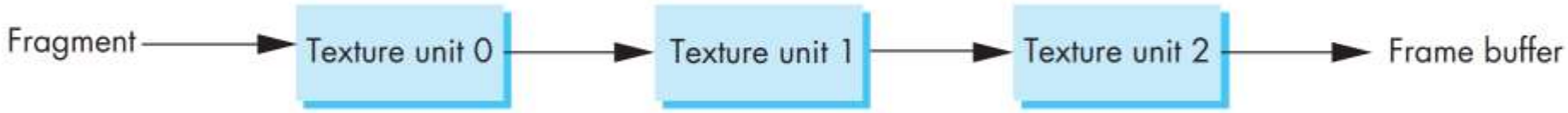

```
int texWidth, texHeight, texChannels;
unsigned char* data = stbi_load("wall.jpg", &texWidth,
&texHeight, &texChannels, 0);

if (data) {
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, texWidth, texHeight, 0,
GL_RGB, GL_UNSIGNED_BYTE, data);
}
else {
  std::cout << "Fail to load wall.jpg\n";
}
stbi_image_free(data);
```

Execution Result



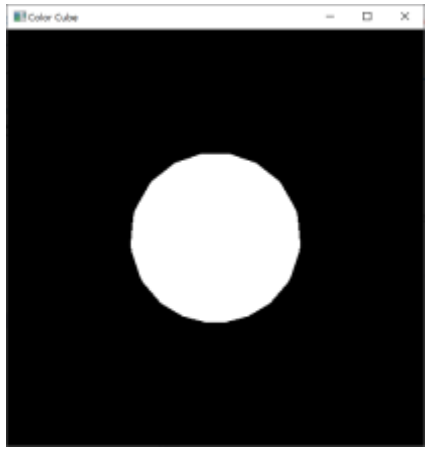


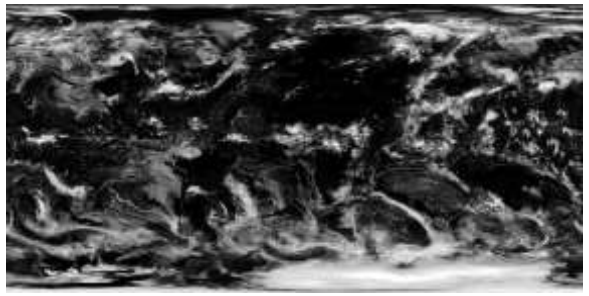
Multi-texturing

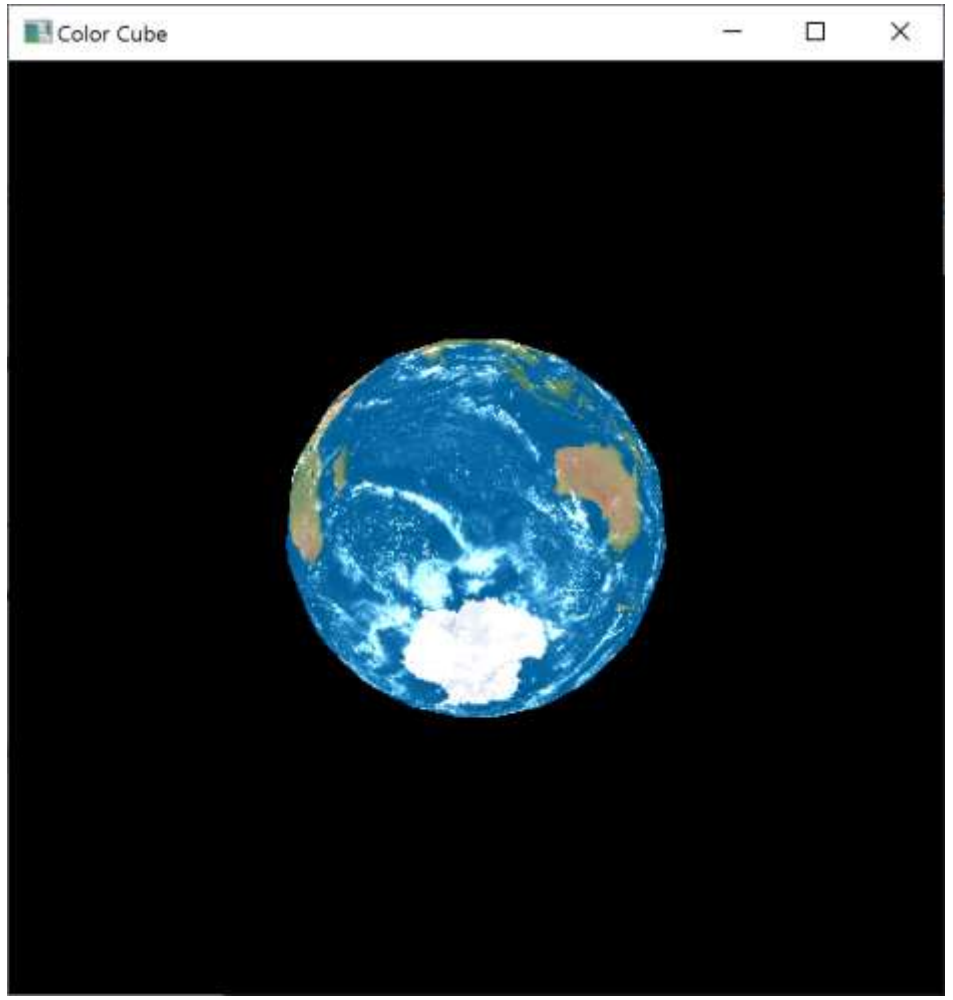



```
glActiveTexture(GL_TEXTURE0); /* unit 0 */
glBindTexture(GL_TEXTURE_2D, object0);
glActiveTexture(GL_TEXTURE1); /* unit 1*/
glBindTexture(GL_TEXTURE_2D, object1);
```

Multi-texturing



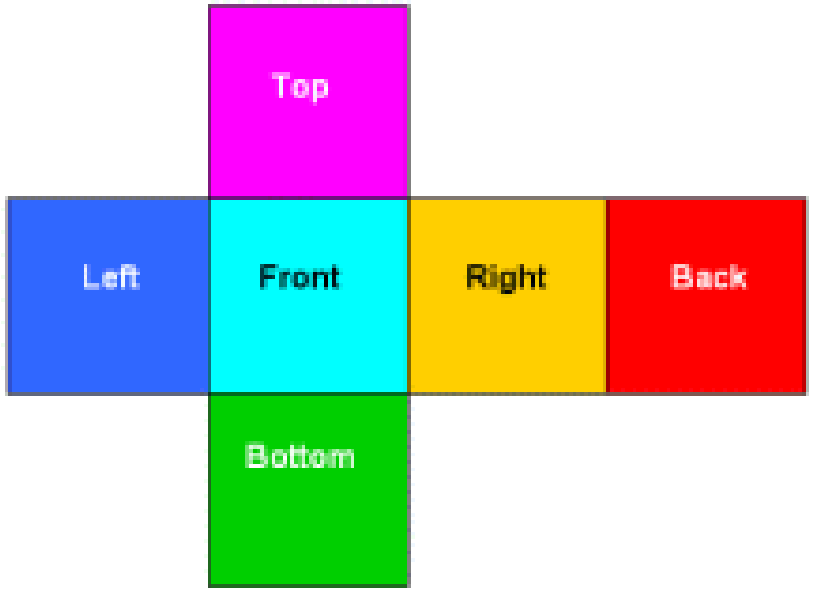


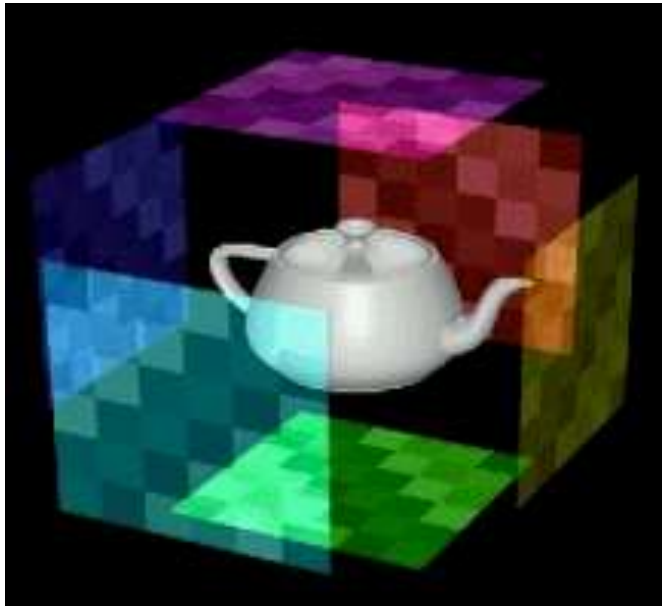


CubeMaps (Box Mapping)

CubeMap

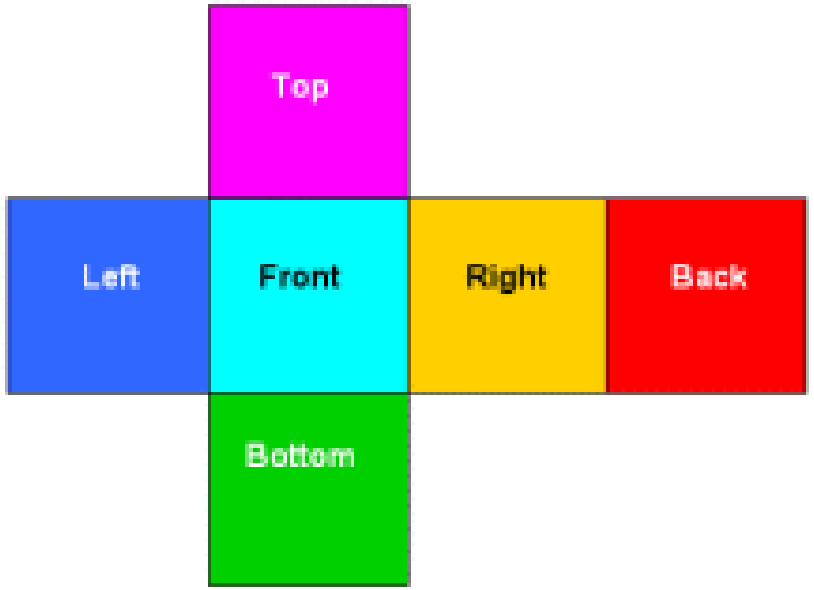
- CubeMap: 2D texture 6개를 포함하는 texture
  - 각각의 2D texture는 박스 (cube)의 한 면임

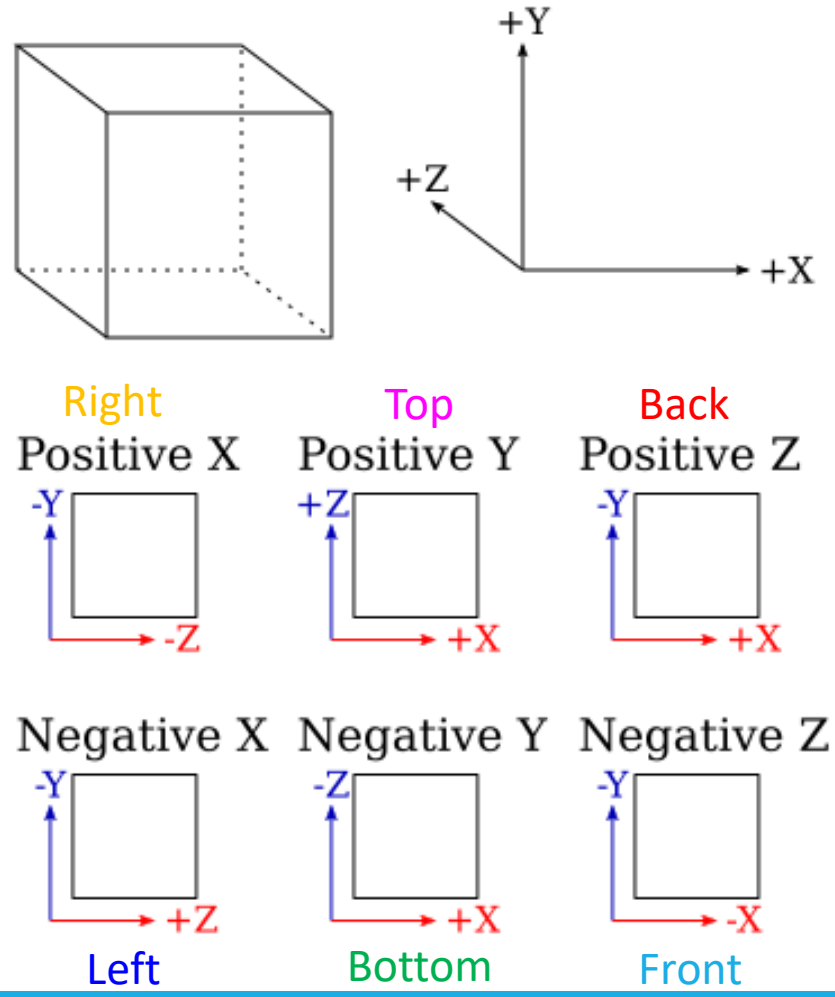




CubeMap (OpenGL convention)

- CubeMap: 2D texture 6개를 포함하는 texture
  - 각각의 2D texture는 박스 (cube)의 한 면임





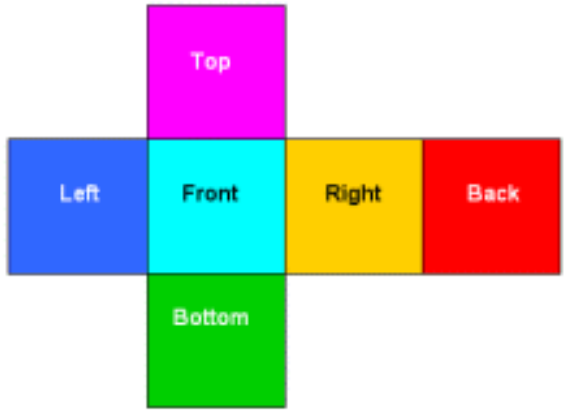
CubeMap in OpenGL

- Cubemap texture 생성:

```
glGenTextures( 1, &texture );
glBindTexture( GL_TEXTURE_CUBE_MAP, texture );
```

- Cubemap의 6면에 2D texture image 로딩하기

```
for (int i = 0; i < 6; i++)
  glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB, TextureSize,
               TextureSize, 0, GL_RGB, GL_UNSIGNED_BYTE, image[i]);
```



Cubemap Face	Layer No.	Orientation
GL_TEXTURE_CUBE_MAP_POSITIVE_X	0	Right
GL_TEXTURE_CUBE_MAP_NEGATIVE_X	1	Left
GL_TEXTURE_CUBE_MAP_POSITIVE_Y	2	Top
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y	3	Bottom
GL_TEXTURE_CUBE_MAP_POSITIVE_Z	4	Back
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z	5	Front

CubeMap in OpenGL

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

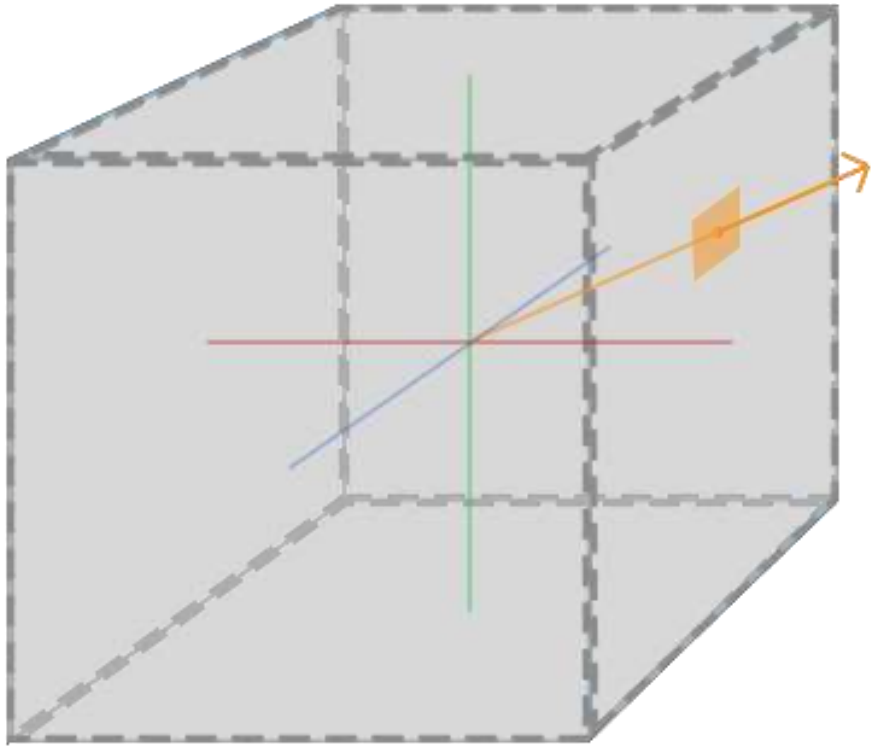
- Texture parameter 설정:
  Texture2D와 비슷하기 Texture parameter 설정 가능

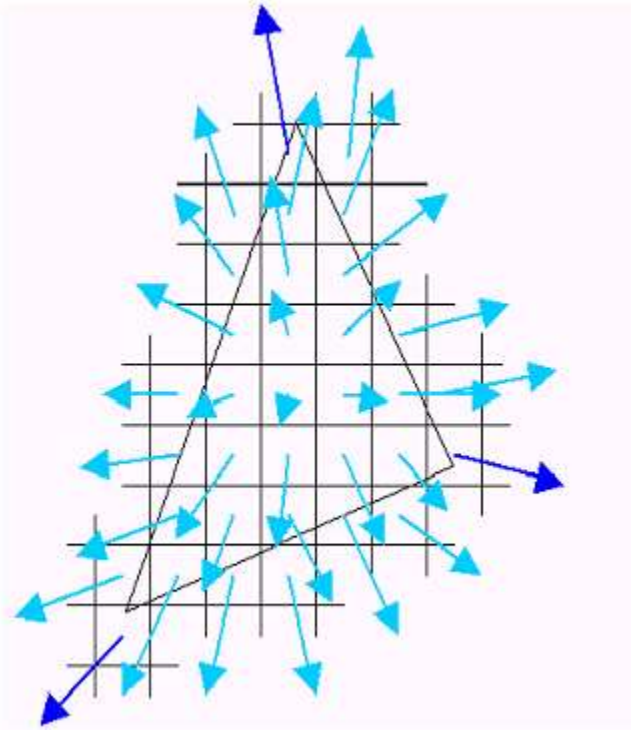
```
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
```

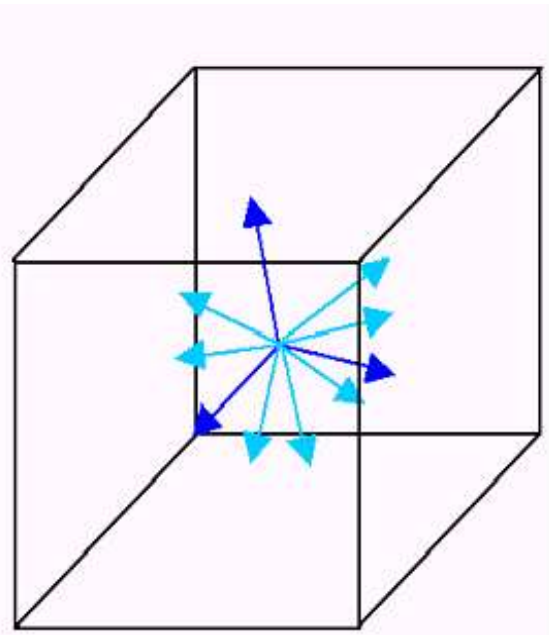
- Texture 사용 전, glActiveTexture()로 texture enabling하기

CubeMap in OpenGL

- 각 점의 texture coordinate: 그 점에서의 법선 방향(vec3)으로 향하는 ray가 unit cube와 만날 때 그 ray의 direction







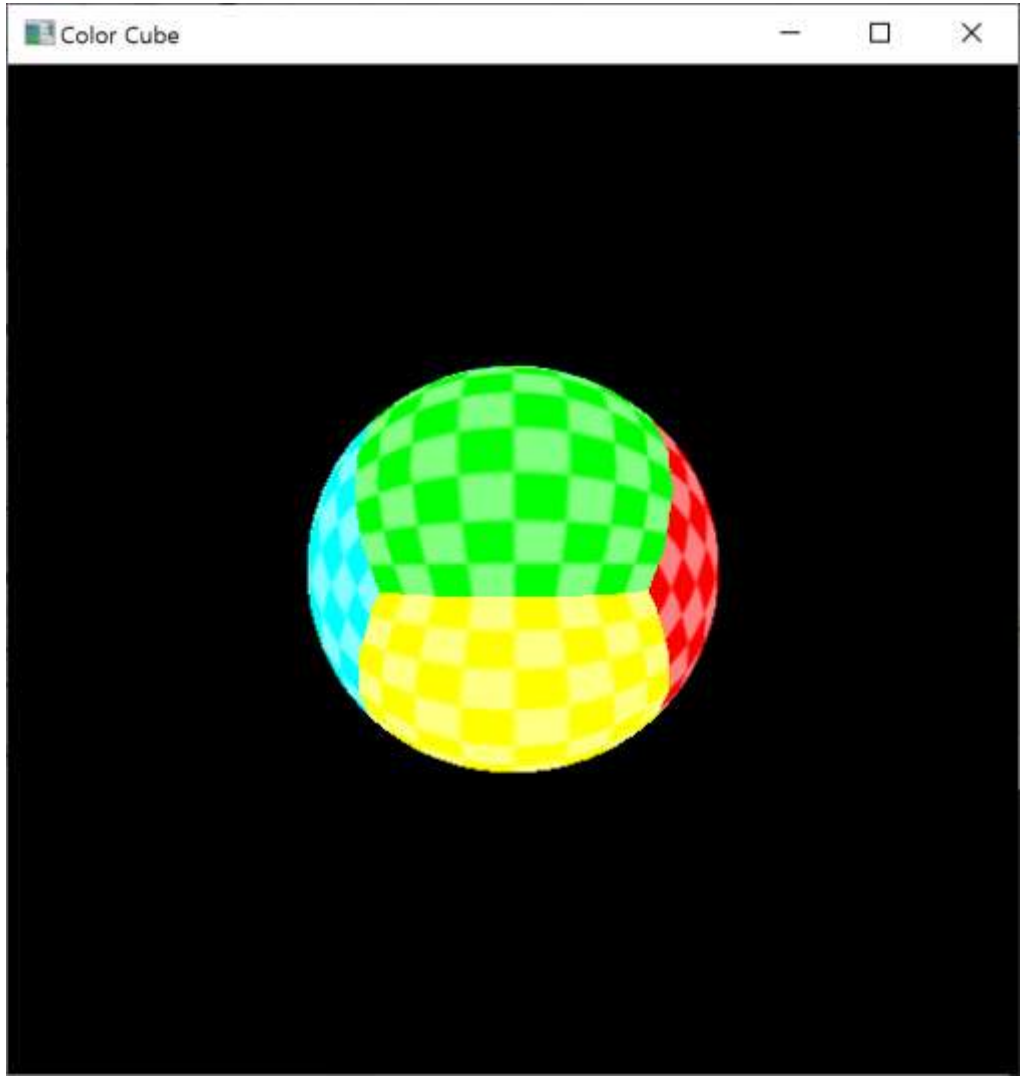
CubeMap in OpenGL


```
#version 330

layout (location = 0) in  vec4 vPosition;
layout (location = 1) in  vec3 vNormal;
layout (location = 2) in  vec2 vTexCoord;

out vec4 color;
out vec3 texCoord;

uniform mat4 ModelView;
uniform mat4 Projection;

void main()
{
    color       = vec4(1.0, 1.0, 1.0, 1.0);
    texCoord    = vNormal;
    gl_Position = Projection * ModelView *
vPosition;
}
```

```
#version 330

in  vec4 color;
in  vec3 texCoord;

out vec4 fColor;

uniform samplerCube cubeTex;

void main()
{
    fColor = color * texture( cubeTex,
texCoord );
}
```

Execution Result

Task: Let’s render this scene!



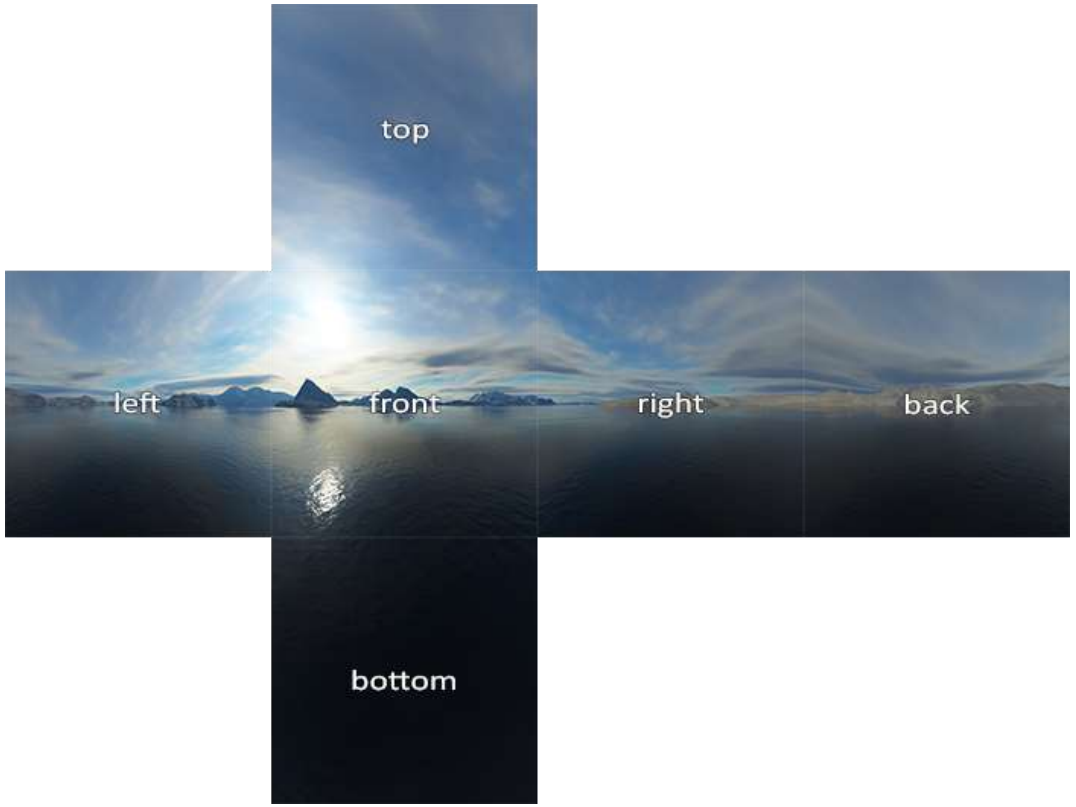
CubeMap Application I: Skybox

Skybox

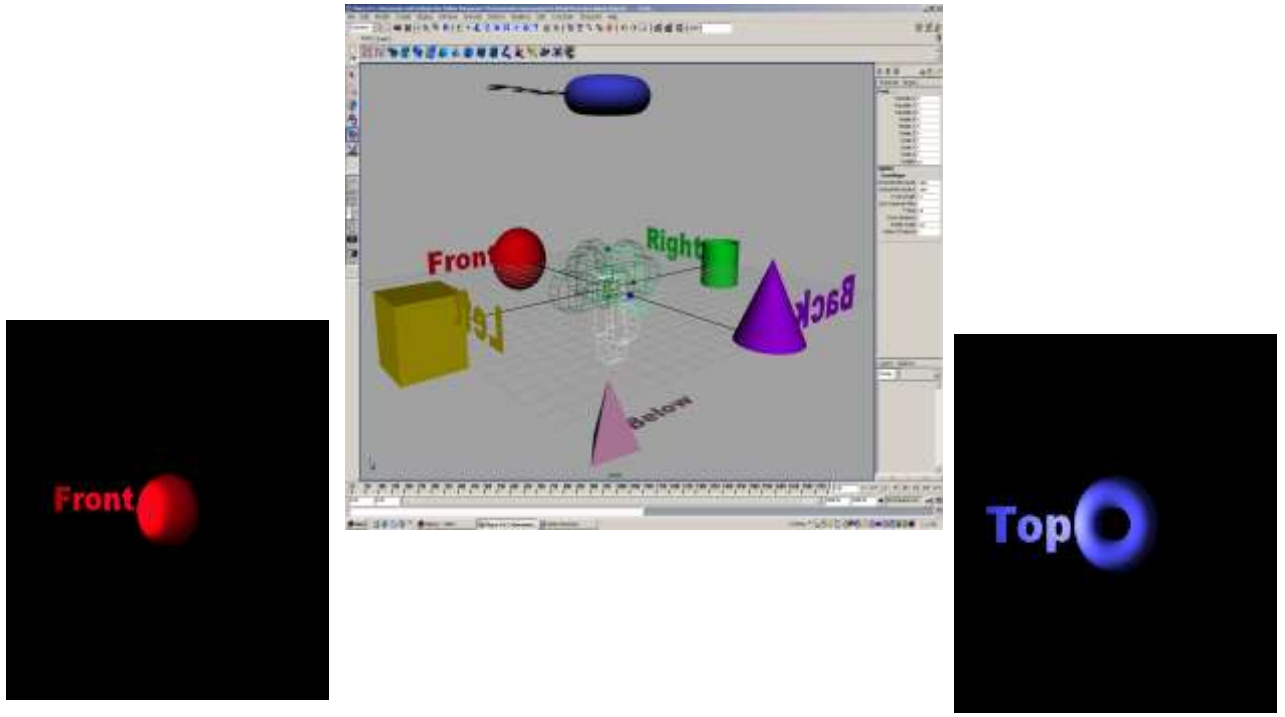
- Skybox: scene을 둘러싼 큰 cube로 video game 등에서 우리가 보는 scene 등이 실제보다 크게 보이는 착각을 일으킴

Scene with a skybox (from Elder Scrolls)

Skybox



Skybox image 예시



Skybox image 찍는 법: 90°씩 회전하며 6 camera로 캡쳐(촬영)

More images in http://www.humus.name/index.php?page=Textures

Skybox

- Add skybox to the scene
  1. Load images and set up a cube map
  2. Display a skybox: map skybox image to the unit cube box

```
std::string skyboxFileNames[6] = {
  "right.jpg", "left.jpg“, "top.jpg",
  "bottom.jpg”, "front.jpg","back.jpg“ };
glGenTextures(1, &skyboxTexture);
glBindTexture(GL_TEXTURE_CUBE_MAP, skyboxTexture);

for (int i = 0; i < 6; i++) {
  int texWidth, texHeight, texChannels;
  std::string fileName = "skybox\\" + skyboxFileNames[i];
  unsigned char* data = stbi_load(fileName.c_str(),
          &texWidth, &texHeight, &texChannels, 0);
if (data) {
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0,
                 GL_RGB, texWidth, texHeight, 0,
                 GL_RGB, GL_UNSIGNED_BYTE, data);
}
else {
    std::cout << "Cannot load " << fileName << "\n";
}
stbi_image_free(data);
```

```
//Make a cube in [-1,1]x[-1,1]x[-1,1]
//   define vertices…
//    setup VAO, VBO, etc. …
```

Skybox

- Skybox의 texture coordinate: cube의 local coordinate

  (가정: cube는 (0,0,0)을 중심으로 놓여져 있을 때)

  =>texture coordinate을 따로 저장할 필요 없음

```
#version 330

layout (location = 0) in  vec4 aPos;
out vec3 texCoord;

uniform mat4 ModelView;
uniform mat4 Projection;

void main()
{
    texCoord    = aPos.xyz;
    gl_Position = Projection * ModelView * aPos;
}
```

```
#version 330

in  vec3 texCoord;
out vec4 fColor;
uniform samplerCube skybox;

void main()
{
    fColor = texture( skybox, texCoord );
}
```

Skybox

- In display()

```
glDepthFunc(GL_LEQUAL);
glUseProgram(skyboxProgram);
ModelView = glGetUniformLocation(skyboxProgram, "ModelView");
Projection = glGetUniformLocation(skyboxProgram, "Projection");

model_view = mat4();
glUniformMatrix4fv(ModelView, 1, GL_TRUE, model_
glUniformMatrix4fv(Projection, 1, GL_TRUE, p);

glBindVertexArray(skyboxVAO);

glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_CUBE_MAP, skyboxTexture

glDrawArrays(GL_TRIANGLES, 0, NumSBVertices);
glBindVertexArray(0);
glDepthFunc(GL_LESS);
```

- Brute force approach: depth writing을 끄고 먼저 skybox를 그리고, object를 그림

glDepthMask(FALSE);
//drawing skybox

```
#version 330

layout (location = 0) in  vec4 aPos;
out vec3 texCoord;

uniform mat4 ModelView;
uniform mat4 Projection;

void main()
{
    texCoord    = aPos.xyz;
    vec4 pos = Projection * ModelView * aPos;
    gl_Position = pos.xyww;
}
```

Skybox

만일 camera가 움직인다면 camera의 rotation 부분만 사용

- In display()

```
glDepthFunc(GL_LEQUAL);
glUseProgram(skyboxProgram);
ModelView = glGetUniformLocation(skyboxProgram, "ModelView");
Projection = glGetUniformLocation(skyboxProgram, "Projection");

model_view = mat4();
glUniformMatrix4fv(ModelView, 1, GL_TRUE, model_view);
glUniformMatrix4fv(Projection, 1, GL_TRUE, p);

glBindVertexArray(skyboxVAO);

glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_CUBE_MAP, skyboxTexture);

glDrawArrays(GL_TRIANGLES, 0, NumSBVertices);
glBindVertexArray(0);
glDepthFunc(GL_LESS);
```

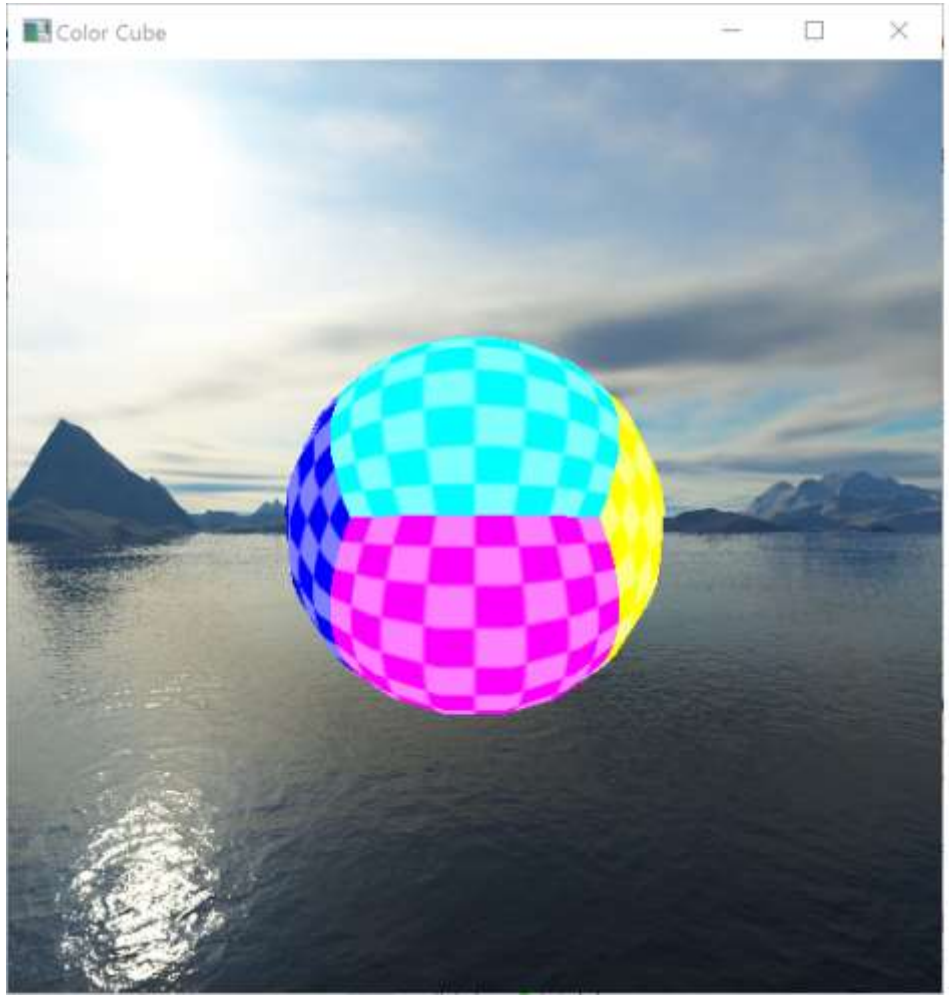
- Brute force approach: depth writing을 끄고 먼저 skybox를 그리고, object를 그림

```
glDepthMask(FALSE);
//drawing skybox
glDepthMask(TRUE);
//drawing objects
```

- Optimized approach: objects들을 먼저 그리고, 남은 공간에만 skybox를 그림

1. Skybox의 depth를 1.0 (maximum)으로 변경
2. glDepthFunc(GL_LEQUAL); Depth Test의 기준 변경

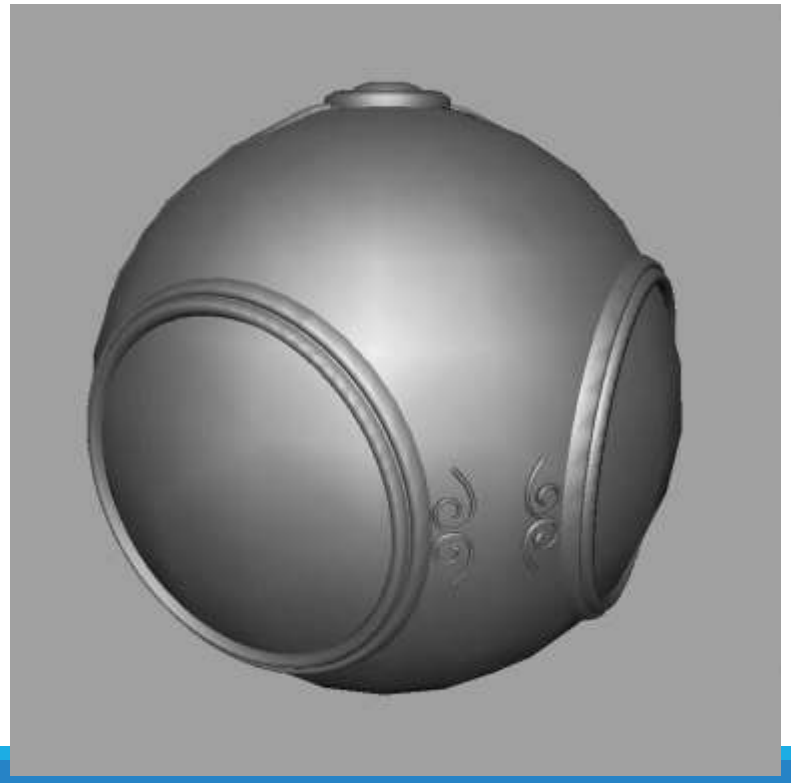
Execution Result



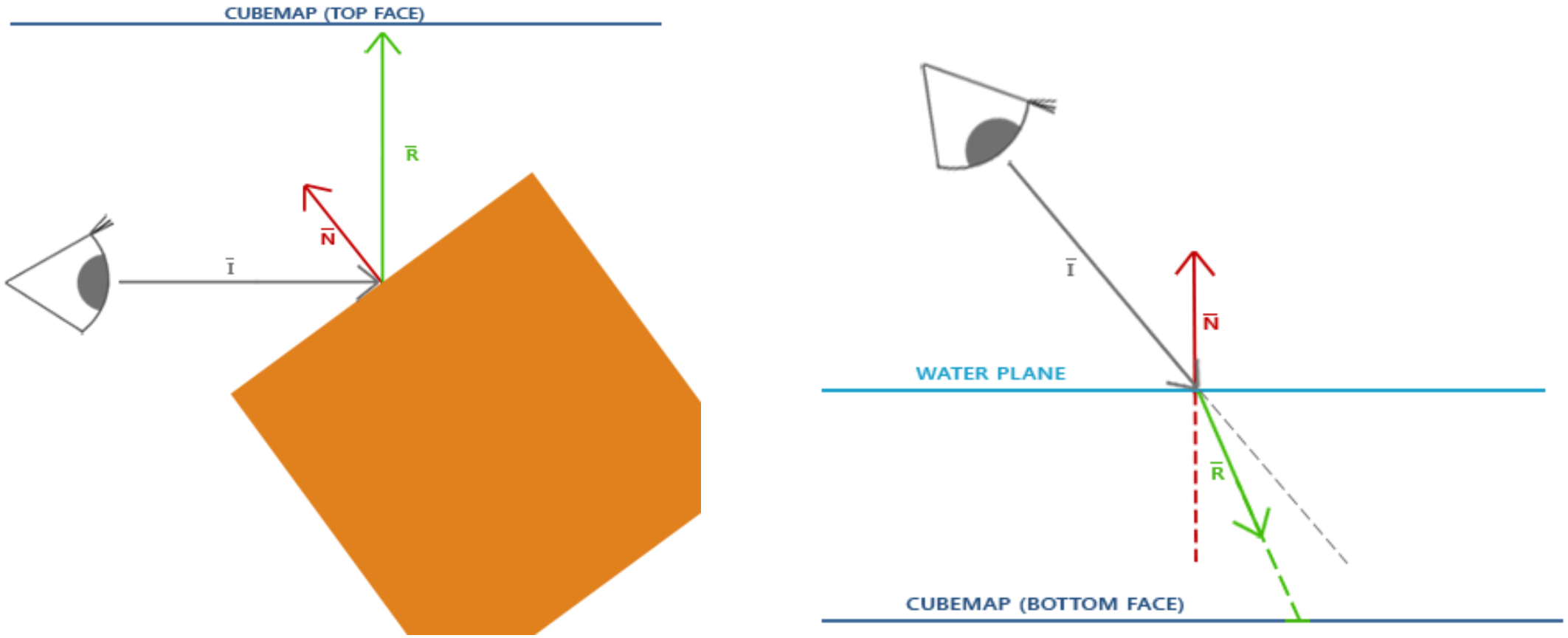
CubeMap Application II: Environment Mapping

Environmental Mapping

- Ray tracing없이 물체의 표면에 배경이 반사/굴절되어 나타나는 효과를 구현하는 방법
- Texture mapping을 이용함



Environmental Mapping



Reflection and refraction inside a cubemap

Environmental Mapping

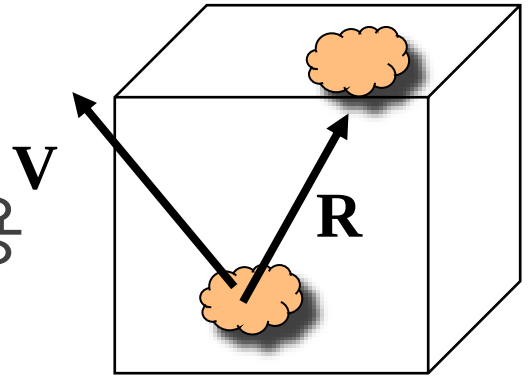
- 물체 (ex. 구)의 각 점의 texture

  => environmental cube map에서 가져옴

  => cube map으로 mapping할 때 다음의 방향 사용

  $R = 2(N \cdot V)N - V$

  (가정: 물체의 중심이 원점에 있음)



- R의 좌표 component중 가장 큰 절대값을 가지는 component
  => mapping되는 cube의 face 결정
- 다른 두 component는 texture coordinate으로 사용

Environmental Mapping in GLSL

- Shaders for sphere

```
#version 330

layout (location = 0) in  vec4 vPosition;
layout (location = 1) in  vec3 vNormal;

out vec3 Normal;
out vec3 Position;

uniform mat4 ModelView;
uniform mat4 Projection;

void main()
{
   Normal = mat3(transpose(inverse(ModelView)))
            * vNormal;
   Position = (ModelView * vPosition).xyz;
   gl_Position = Projection * ModelView *
                 vPosition;
}
```

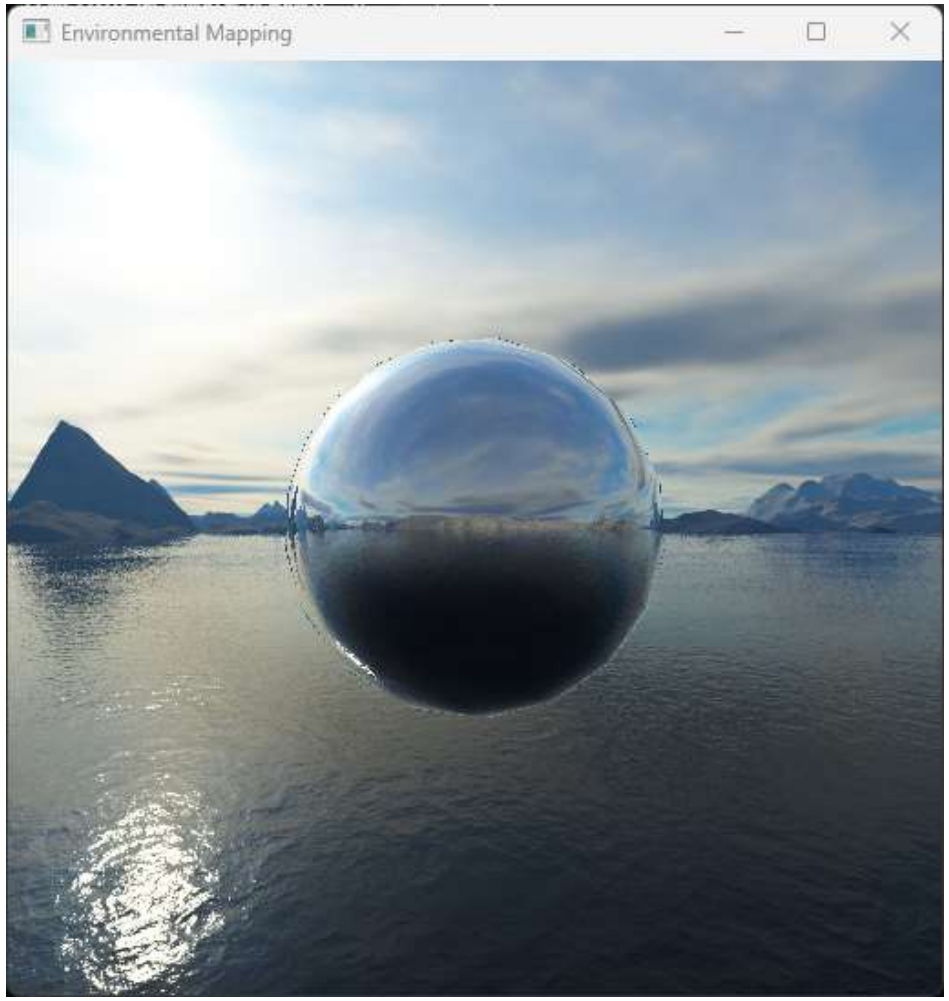
```
#version 330

in  vec3 Normal;
in  vec3 Position;

out vec4 fColor;

uniform vec3 cameraPos;
uniform samplerCube skybox;

void main()
{
    vec3 I = normalize(Position - cameraPos);
    vec3 R = reflect(I, normalize(Normal));
    fColor = vec4(texture(skybox, R).rgb,
1.0);
}
```

Environmental Mapping in GLSL

- In fragment shader:
  - Use function reflect to compute reflected ray
  - I: direction from the eye to the position in world coordinate
  - I = Position – eye_position
  - Normal: normal vector of surface in world coordinate

```
#version 330

in  vec3 Normal;
in  vec3 Position;

out vec4 fColor;

uniform vec3 cameraPos;
uniform samplerCube skybox;

void main()
{
    vec3 I = normalize(Position - cameraPos);
    vec3 R = reflect(I, normalize(Normal));
    fColor = vec4(texture(skybox, R).rgb,
1.0);
}
```

Execution Result



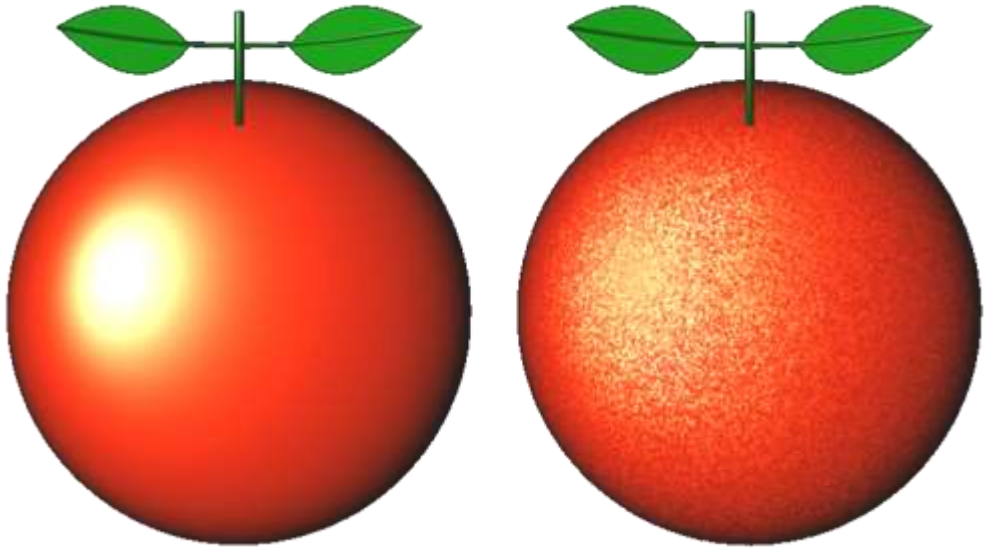
Normal-Bump Mapping

Normal – Bump Mapping

- Shading equation:

$$I = I_a k_a + I_p (k_d \left(\left(N + N(x, y, z) \right) \cdot L \right) + k_s (R \cdot V)^n)$$

- 위의 식에서 R 또한 (perturbed) normal을 포함

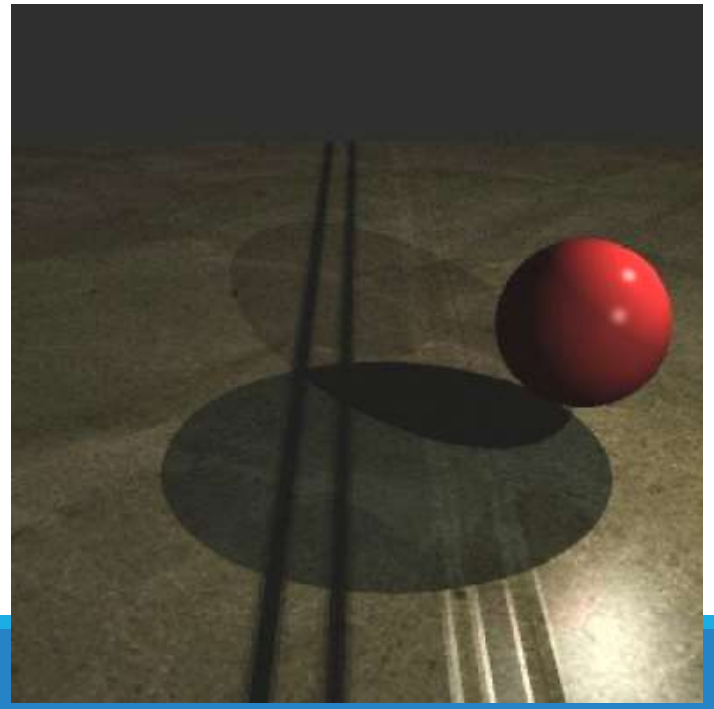


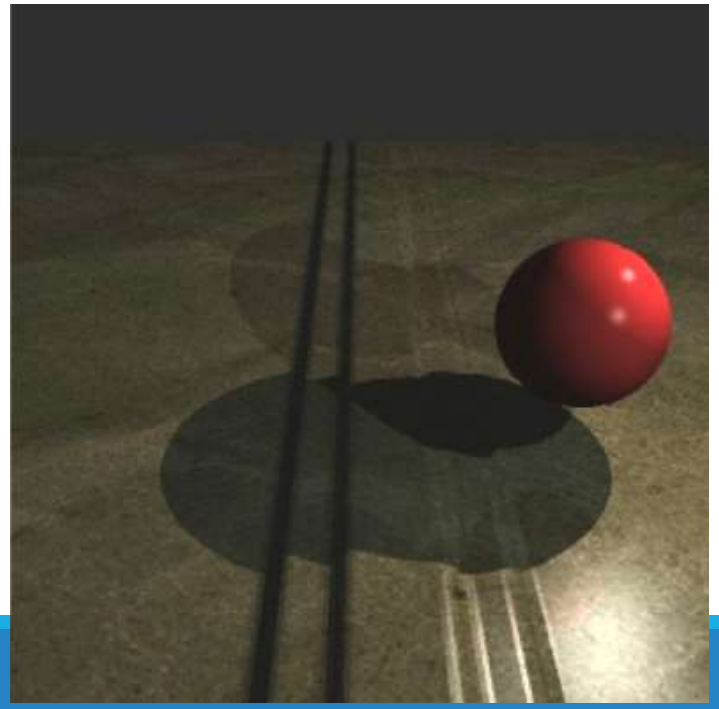
Bump/Displacement Mapping

- Normal/Bump map은 실제 surface의 형태를 변화시키지 않는 shading trick

⇒실루엣을 보면 여전히 부드럽게 보임

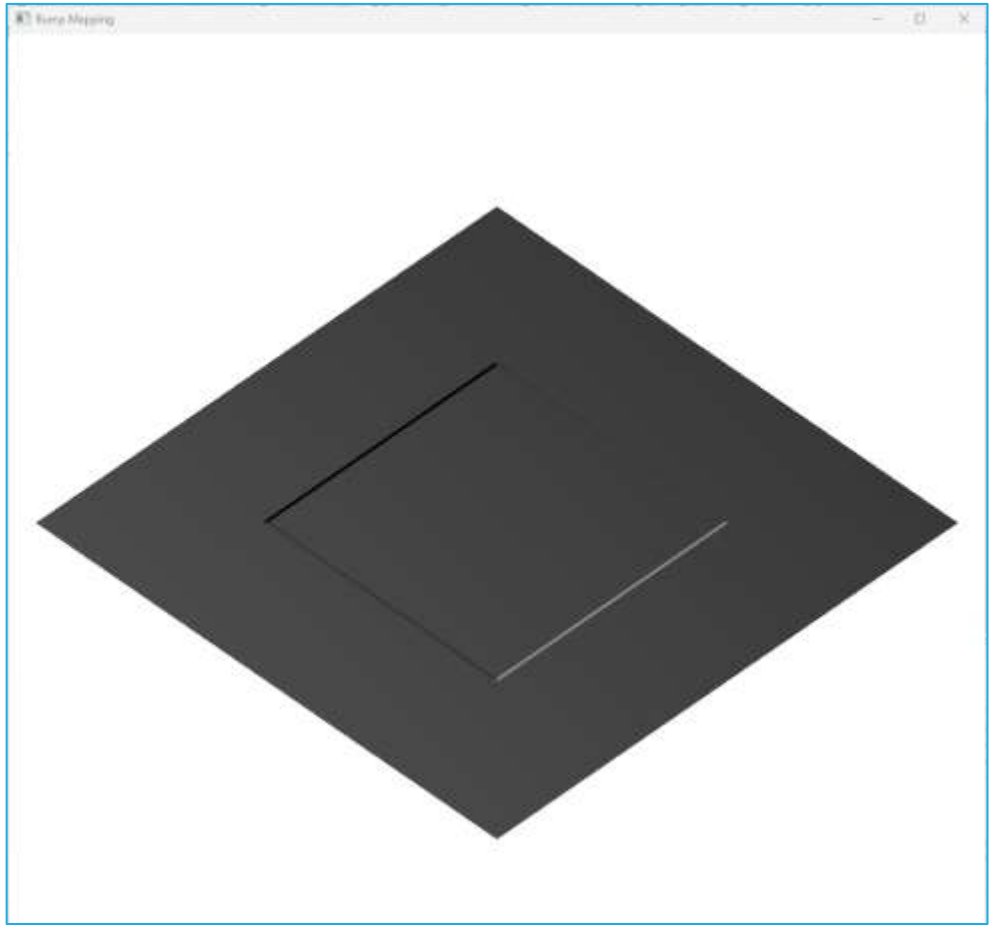
- Displacement mapping: normal 방향의 height map 저장하고 실제로 surface geometry에 영향을 줌





main_texturemapping8.cpp

- Draw a single square plane y = 0
- A light source above the plane rotates in the plane y = 10.0
- Plane has displacement of a small square in the center of the original square



main_texturemapping8.cpp

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

```
void mesh()
{
    point4 vertices[4] = {
       point4(0.0, 0.0, 0.0, 1.0),
       point4(1.0, 0.0, 0.0, 1.0),
       point4(1.0, 0.0, 1.0, 1.0),
       point4(0.0, 0.0, 1.0, 1.0) };

    points[0] = vertices[0];
    tex_coord[0] = vec2(0.0, 0.0);
    points[1] = vertices[1];
    tex_coord[1] = vec2(1.0, 0.0);
    points[2] = vertices[2];
    tex_coord[2] = vec2(1.0, 1.0);
    points[3] = vertices[2];
    tex_coord[3] = vec2(1.0, 1.0);
    points[4] = vertices[3];
    tex_coord[4] = vec2(0.0, 1.0);
    points[5] = vertices[0];
    tex_coord[5] = vec2(0.0, 0.0);
}
```

평면은 2 삼각형으로만 그림

```
int i, j, k;
float d;
float data[N + 1][N + 1]; // N = 256
for (i = 0; i < N + 1; i++)
   for (j = 0; j < N + 1; j++) data[i][j] = 0.0;
for (i = N / 4; i < 3 * N / 4; i++)
   for (j = N / 4; j < 3 * N / 4; j++) data[i][j] = 1.0;

for (i = 0; i < N; i++) for (j = 0; j < N; j++) {
   normals[i][j][0] = data[i][j] - data[i + 1][j];
   normals[i][j][2] = data[i][j] - data[i][j + 1];
   normals[i][j][1] = 1.0;
}

for (i = 0; i < N; i++) for (j = 0; j < N; j++) {
   d = 0.0;
   for (k = 0; k < 3; k++)
      d += normals[i][j][k] * normals[i][j][k];
   d = sqrt(d);
   for (k = 0; k < 3; k++)
      normals[i][j][k] = 0.5 * normals[i][j][k] / d + 0.5;
 }
```

평면의 normal map을 256x256의 texture로 저장

main_texturemapping8.cpp

- Vertex shader for normal mapping

```
#version 330
layout (location = 0) in vec4 vPosition;
layout (location = 1) in vec2 texcoord;

out vec3 L; /* light vector in texture-space coordinates
*/
out vec3 V; /* view vector in texture-space coordinates
*/

uniform vec4 Normal;
uniform vec4 LightPosition;
uniform mat4 ModelView;
uniform mat4 Projection;
uniform mat4 NormalMatrix;
uniform vec3 objTangent; /* tangent vector in object
coordinates */

out vec2 st;
```

```
void main(){
  mat3 NM3 = mat3(NormalMatrix);
  gl_Position = Projection*ModelView*vPosition;
  st = texcoord;
  vec3 eyePosition = vec3(ModelView*vPosition);
  vec3 eyeLightPos = (ModelView*LightPosition).xyz;
  /* normal, tangent and binormal in eye coordinates */
  vec3 N = normalize(NM3*Normal.xyz);
  vec3 T  = normalize(NM3*objTangent);
  vec3 B = cross(N, T);
  /* light vector in texture space */
  L.x = dot(T, eyeLightPos-eyePosition);
  L.y = dot(B, eyeLightPos-eyePosition);
  L.z = dot(N, eyeLightPos-eyePosition);
  L = normalize(L);
  /* view vector in texture space */
  V.x = dot(T, -eyePosition);
  V.y = dot(B, -eyePosition);
  V.z = dot(N, -eyePosition);
  V = normalize(V);
}
```

main_texturemapping8.cpp

- Fragment shader for normal mapping

```
#version 330
in vec3 L;
in vec3 V;
in vec2 st;

uniform sampler2D texMap;
uniform vec4 DiffuseProduct;

out vec4 fColor;

void main(){
   vec4 N = texture2D(texMap, st);
   vec3 NN =  normalize(2.0*N.xyz-1.0);
   vec3 LL = normalize(L);
   float Kd = max(dot(NN.xyz, LL), 0.0);
   fColor = Kd*DiffuseProduct;
   fColor = vec4(fColor.rgb, 1.0);
}
```

Exercise) Environmental Mapping

- Exercise) Teapot_normal.obj에 있는 teapo에 environmental mapping 적용시키기
  - main_shadingbunny.cpp와 main_texturemapping7.cpp 참조하기