CG Practice 8

COLLEGE OF COMPUTING

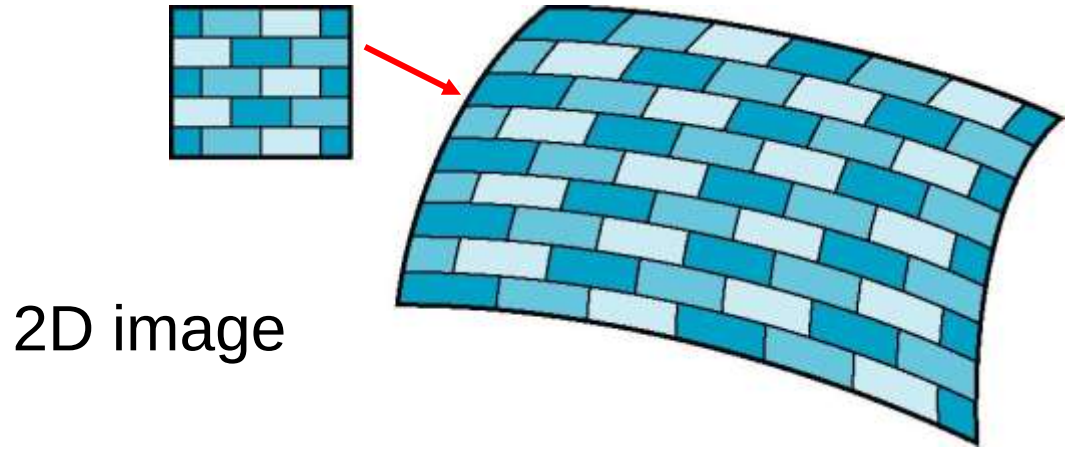
HANYANG ERICA CAMPUS

Q YOUN HONG (홍규연)

Texture Mapping

Texture Mapping

- 3D surface위의 한 점 $P = (x, y, z)$를 2D rectangular image 상의 texture coordinate (s, t)로 매핑



2D image

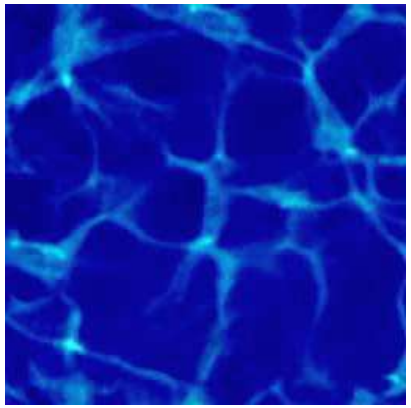
3D surface

Texture

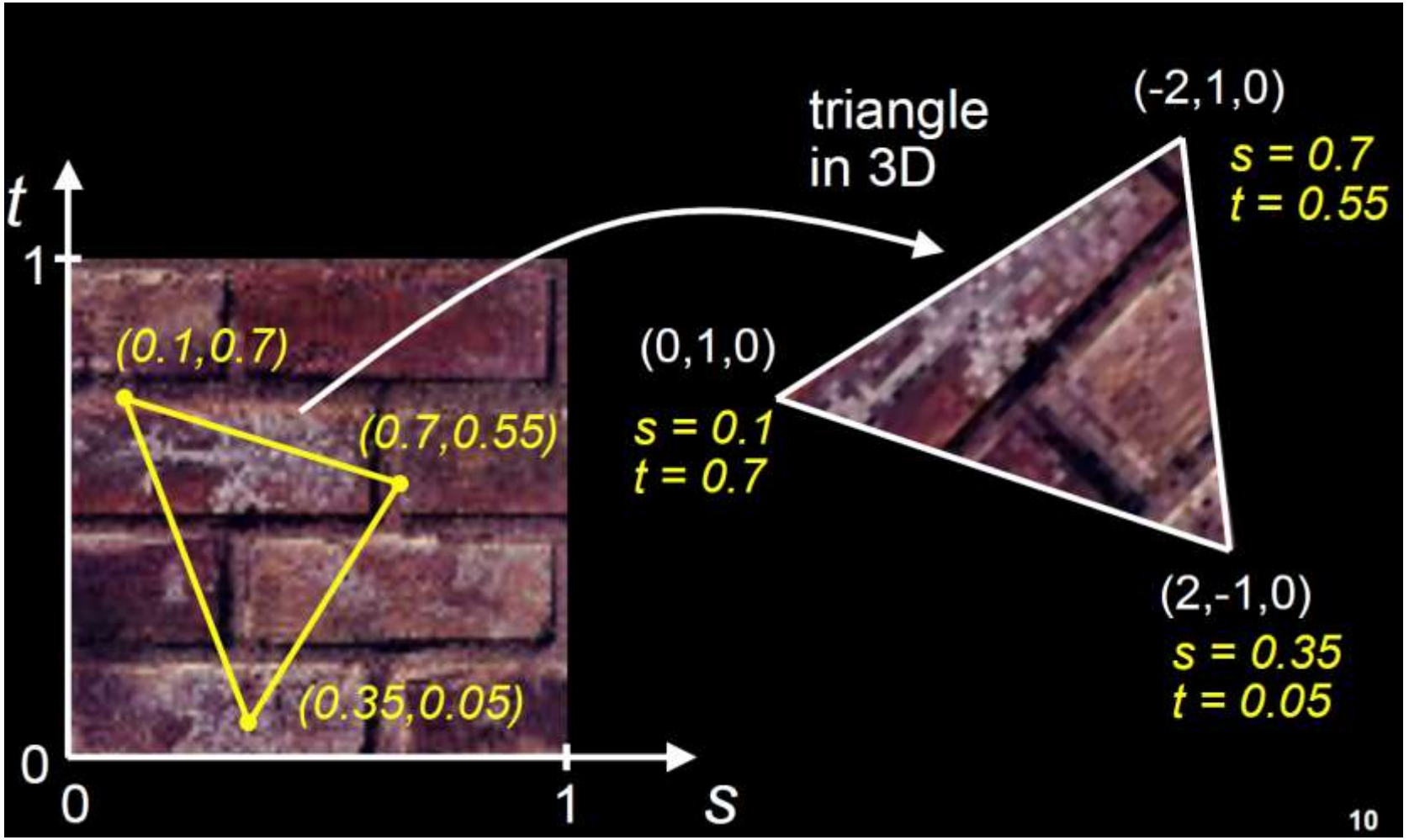
- Texture는 bitmap image임
  - 이미지 라이브러리를 이용해서 메모리로 로드하거나,
  - 직접 프로그램 안에서 만들 수 있음
- 2D array:

```
unsigned char texture[height][width][4]
unsigned char texture[4*height*width]
```

- Texture 안에 있는 pixel은 ***texel***이라고 불림
- Texture coordinate: (s,t), where $(s,t) \in [0,1]^2$

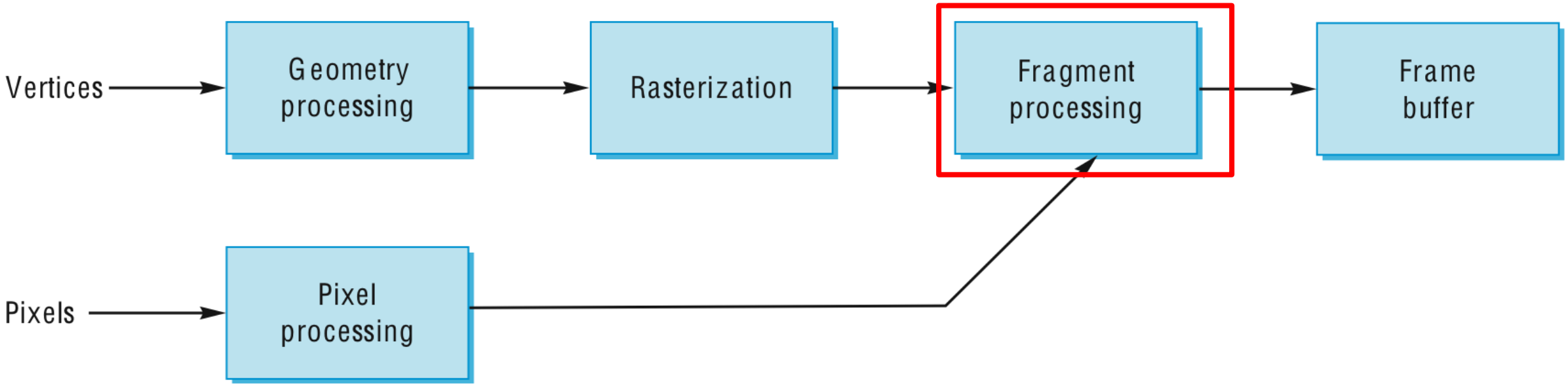


Texture Mapping



Texture Mapping이 일어나는 장소는?

Texture mapping은 fragment processing의 일부로 행해진다



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 Image 결정하기

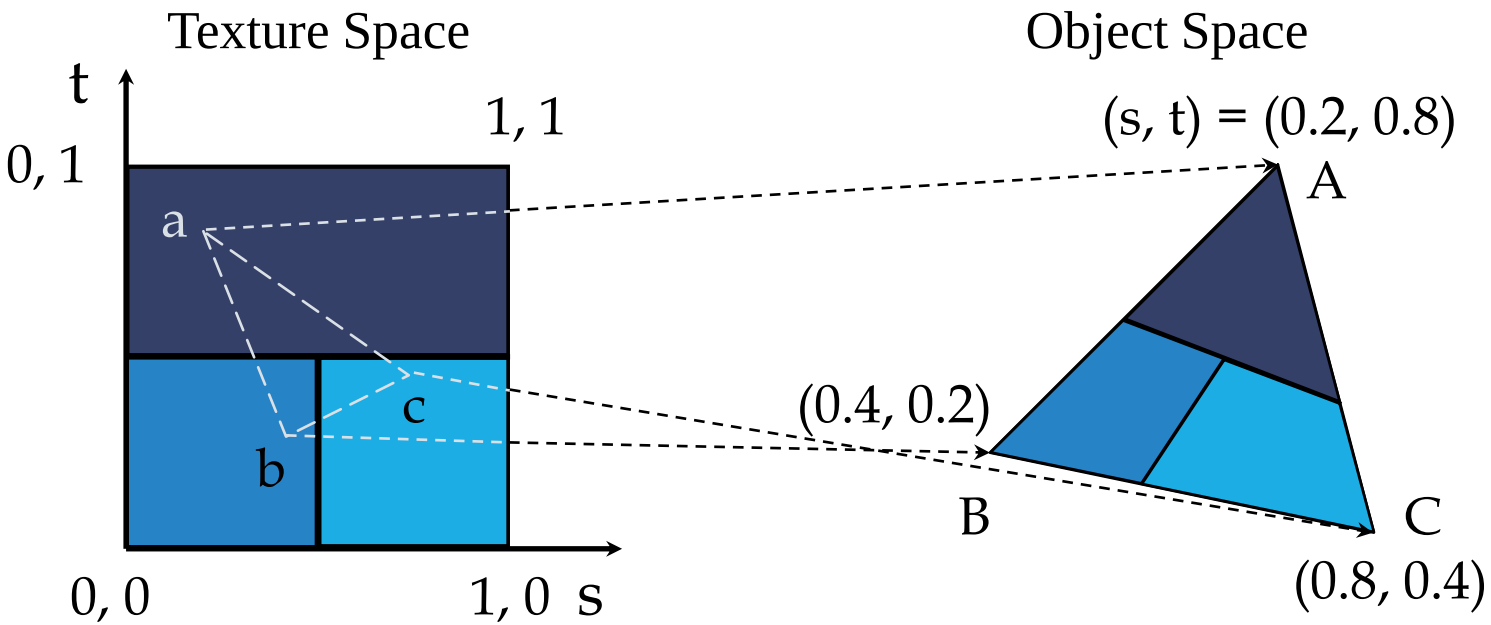
- Texture 이미지는 texel들의 배열로 CPU memory에 저장함

```
GLubyte texels[512][512][3];
```

  - 이미지 파일에서 부르거나 프로그램 내에서 직접 생성

Texture Mapping하기

- 각 vertex의 texture coordinate을 이용해서 interpolation



Texture Parameters

- OpenGL에는 다양한 texture 관련한 parameter들이 존재하여 어떻게 texture가 적용될 것인지 결정
  - Wrapping parameters: texture cooridnate s나 t가 [0,1] 이외의 값이 될 경우의 동작을 정의
  - Filter mode: texel의 값을 결정할 때, point sample 대신 area averaging 등 허용
  - Mipmapping: 다해상도 텍스쳐 지원
  - Environment parameters: texture mapping이 다른 shading과 어떻게 상호작용하는지 결정

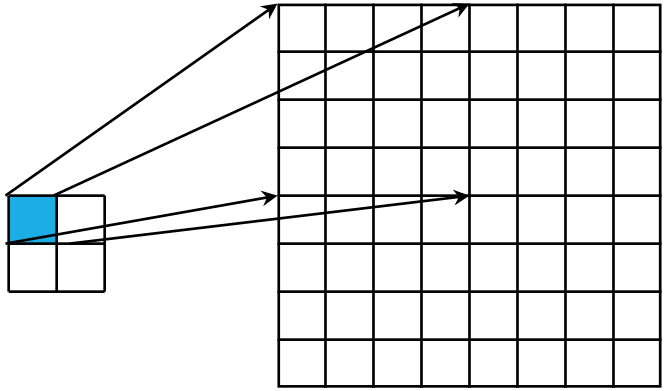
Wrapping Mode

- Clamping: if s,t > 1, use 1, if s,t < 0 use 0
- Wrapping: use s,t, modulo 1

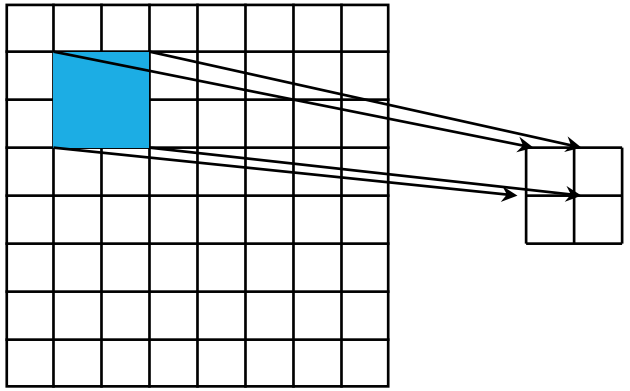
```
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_S, GL_CLAMP )
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_T, GL_REPEAT )
```

Magnification/Minification

- More than one texel can cover a pixel (minification) or more than one pixel can cover a texel (magnification)
- Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values



Magnification



Minification

Magnification/Minification

glTexParameteri(target, type, mode)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MAG_FILTER,
                GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MIN_FILTER,

                GL_LINEAR);
```

=> linear filtering은 edge filtering을 위해서 border를 따라서 1줄 씩 extra texel이 필요함

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

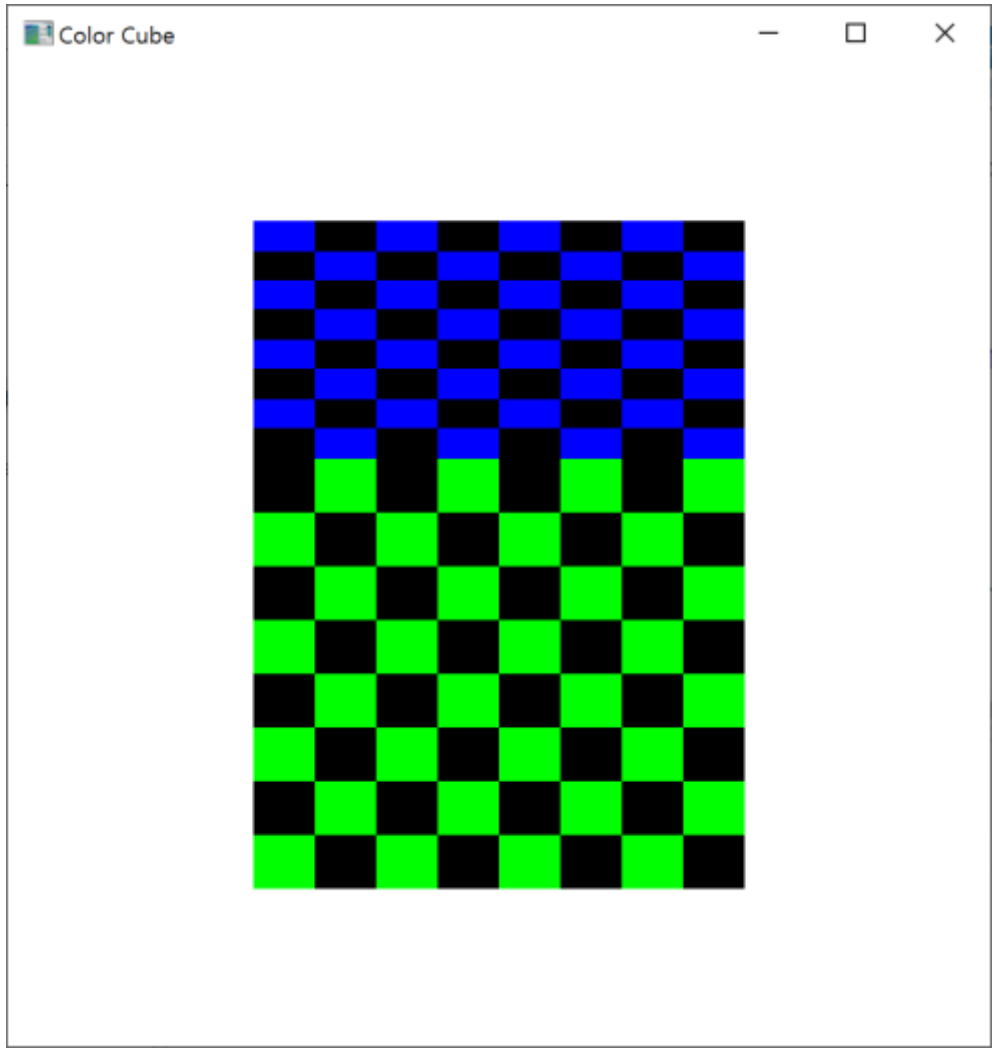
- Texture를 실제로 적용하는 것은 Fragment processing에서 이루어진다. (in fragment shader)
- Texture들은 application에서 sampler 변수로 받음
  - Sampler1D, **Sampler2D**, Sampler3D, SamplerCube
- Sampler는 texture object로부터 texture coordinate에 해당하는 texture color를 반환함

```
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 );
```

Example Result



Example: Texturing on Cube 2

Goal

- Rotating cube에 2D texture 적용하기
- main_texturemapping1.cpp에서 시작,
- texture는 jpg 파일에서 읽음

Loading Texture Image

- 이미지는 texture로 이용될 수 있음
- OpenGL는 이미지 로딩을 지원하지 않음
- 외부 라이브러리를 이용하여 이미지 로딩 (OpenCV, QT…)

⇒ 이미지는 unsigned byte array로 변환하여 저장

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

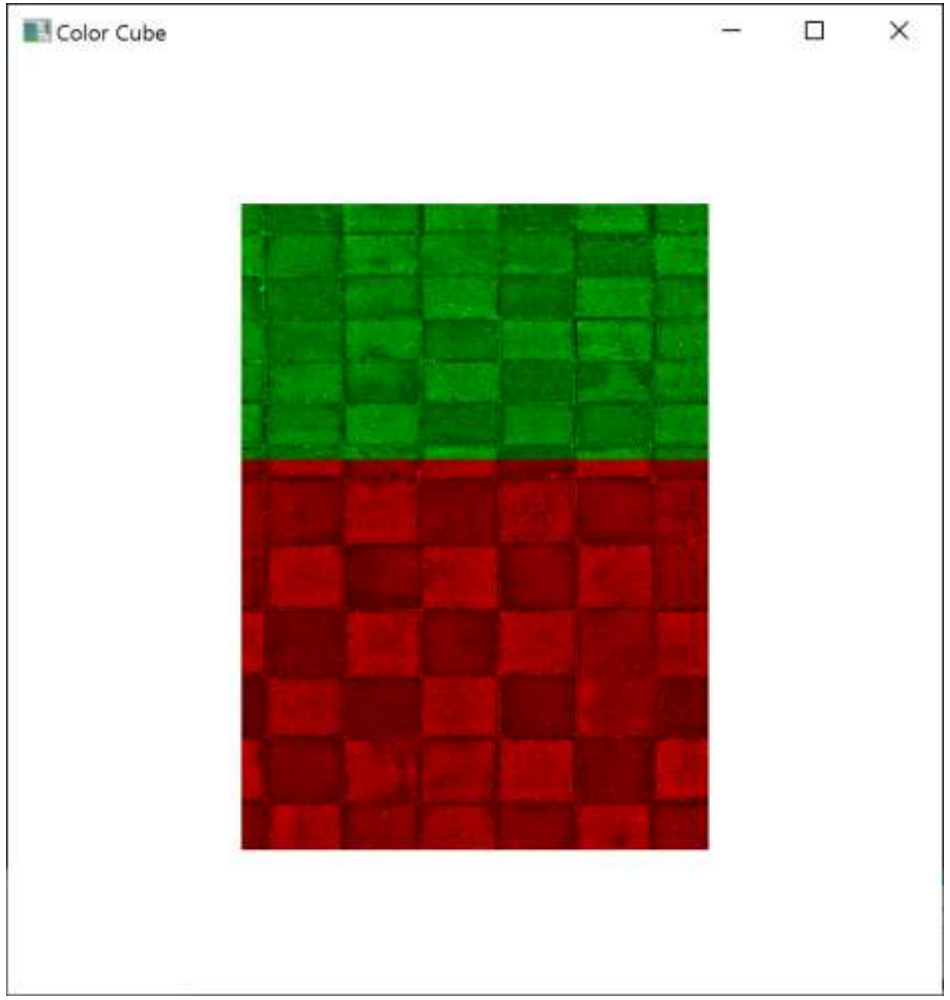
```
int texWidth, texHeight, texChannels;
unsigned char* data = stbi_load("wall.jpg", &texWidth, &texHeight, &texChannels, 0);
```

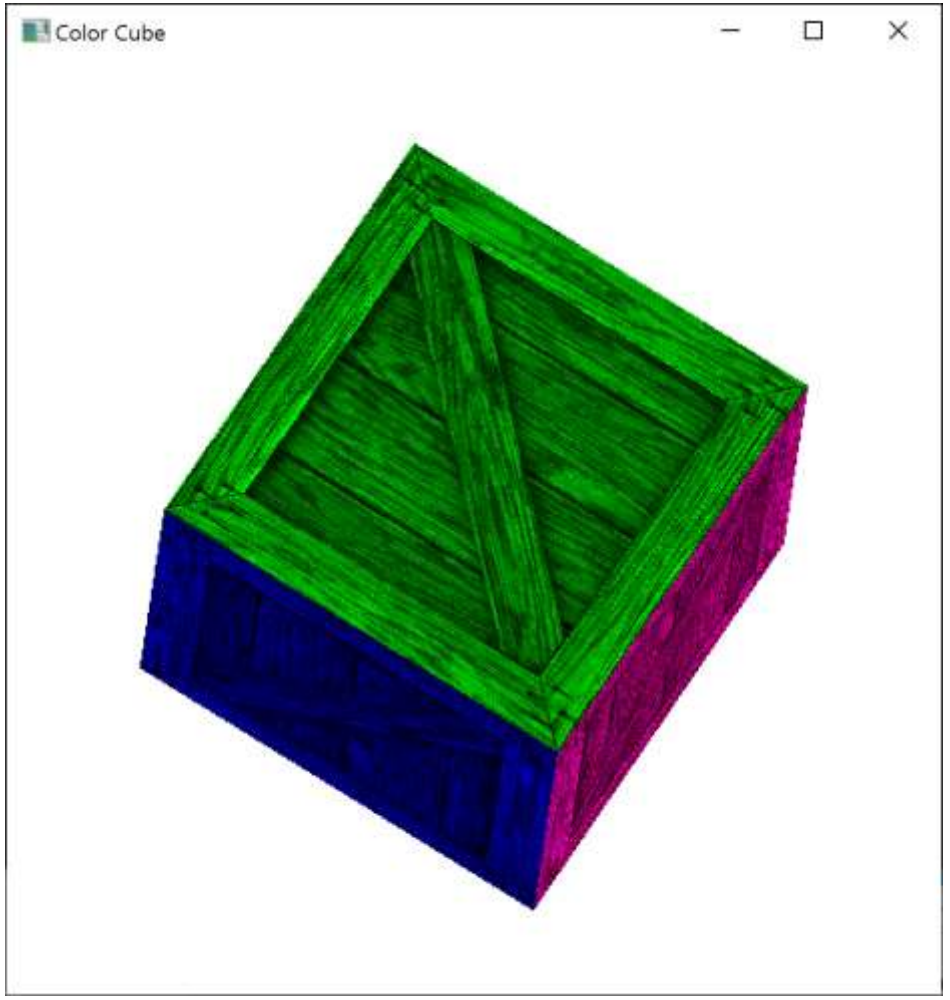
Wall.jpg 파일을 로딩하여 unsigned byte array에 저장

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



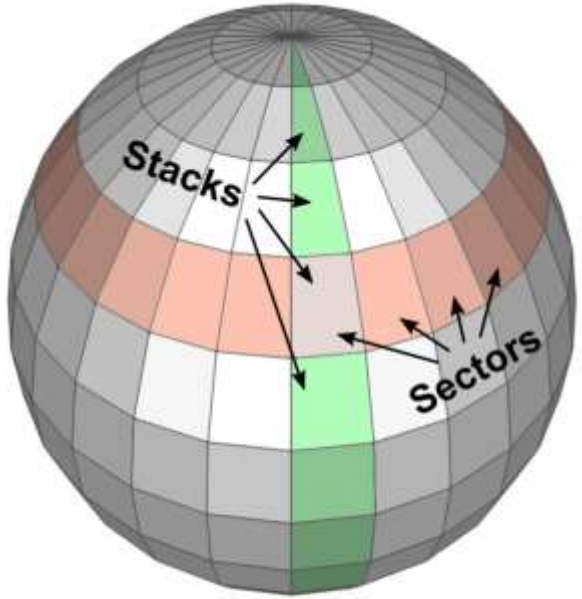


Example: Drawing the Earth

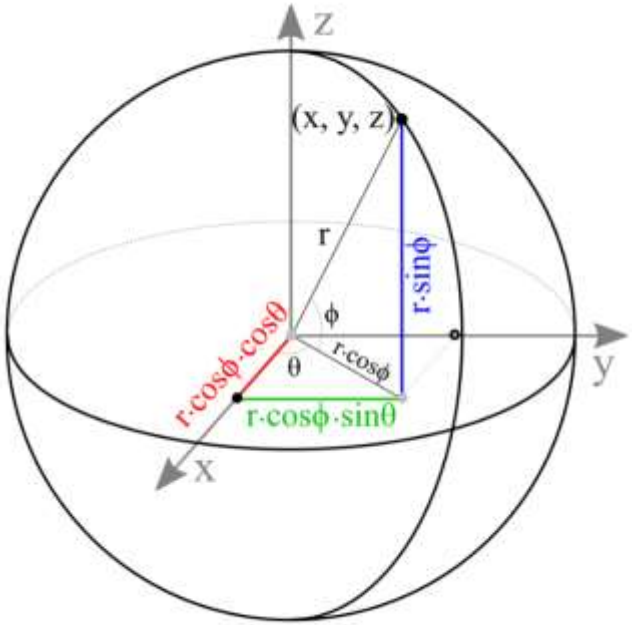
Goal

- Drawing the earth (sphere) with textures
- Multi-texturing
- Rendering with various types of texture maps

Creating a Sphere



Sectors and stacks of a sphere



A point on a sphere using sector and stack angles

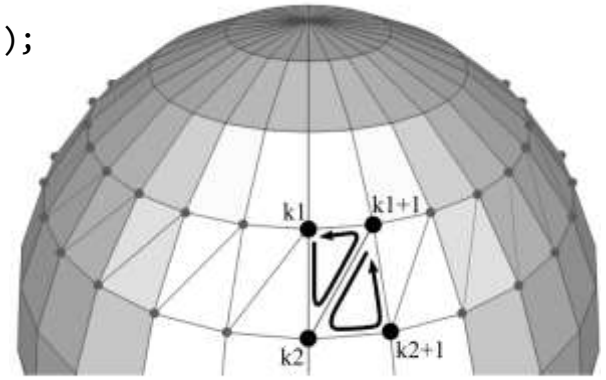
$$x = (r\cos\phi)cos\theta$$
$$y = (r\cos\phi)\sin\theta$$
$$z = rsin\phi$$
where $0 \le \theta \le 2\pi,\ -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}$

Creating a Sphere


```
void createSphere(GLfloat radius, int NumSectors, int NumStacks)
{
    GLfloat x, y, z, xy;
    GLfloat stackStep = M_PI / (GLfloat)NumStacks;
    GLfloat sectorStep = 2.0 * M_PI / (GLfloat)NumSectors;

    // compute vertices
    for (int i = 0; i <= NumStacks; i++) {
      vec4 pt;
      vec3 nor;
      vec2 tex;

      float stackAngle = M_PI_2 - (GLfloat)i * stackStep;
      xy = cosf(stackAngle);
      z = sinf(stackAngle);

      for (int j = 0; j <= NumSectors; j++) {
        float sectorAngle = j * sectorStep;
        x = xy * cosf(sectorAngle);
        y = xy * sinf(sectorAngle);

        pt = vec4(radius * x, radius * y, radius * z, 1.0);
        nor = vec3(x, y, z);
        tex = vec2((GLfloat)j / NumSectors, (GLfloat)i / NumStacks);

        points.push_back(pt);
        normals.push_back(nor);
        texCoords.push_back(tex);
      }
    }
```

(출처: http://www.songho.ca/opengl/gl_sphere.html)

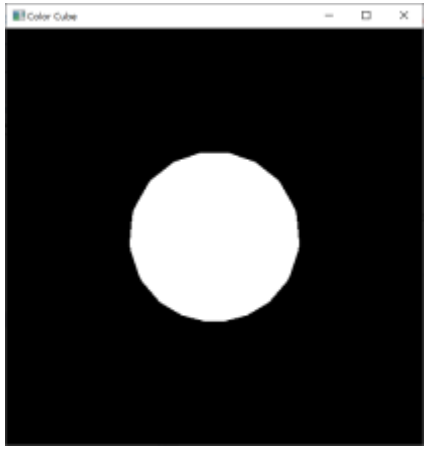
```
//create indices
// k1 - k1 + 1
// |   /   |
// k2 - k2 + 1
for (int i = 0; i < NumStacks; i++) {
  int k1 = i * (NumSectors + 1);
  int k2 = k1 + NumSectors + 1;
  for (int j = 0; j < NumSectors; j++, k1++, k2++) {
    if (i != 0) {
      indices.push_back(k1);
      indices.push_back(k2);
      indices.push_back(k1 + 1);
    }

    if (i != NumStacks - 1) {
      indices.push_back(k1 + 1);
      indices.push_back(k2);
      indices.push_back(k2 + 1);
    }
  }
}
```

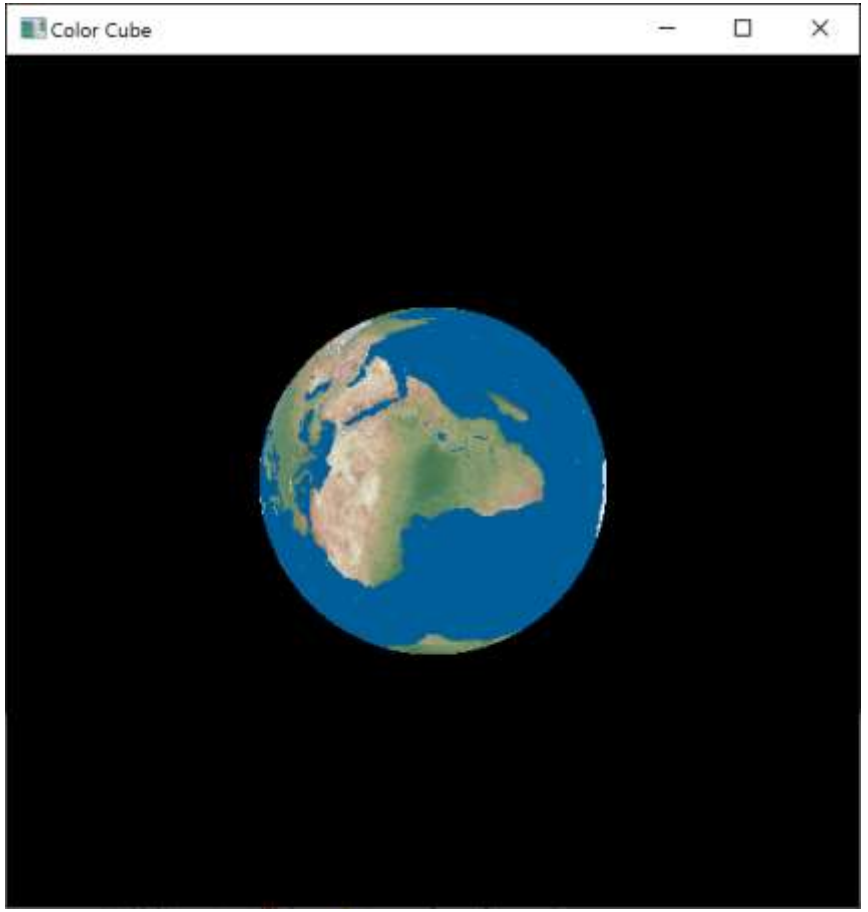


vertex indices to draw triangles of a sphere

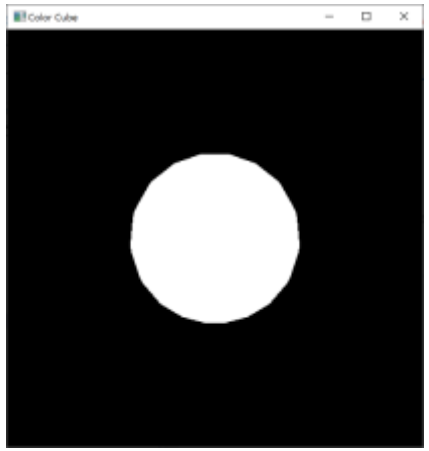
Texturing on a Sphere

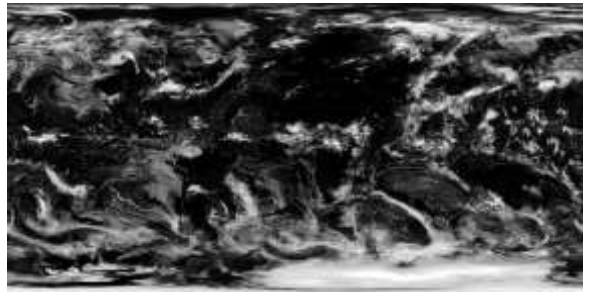
Apply the earth map texture to the sphere



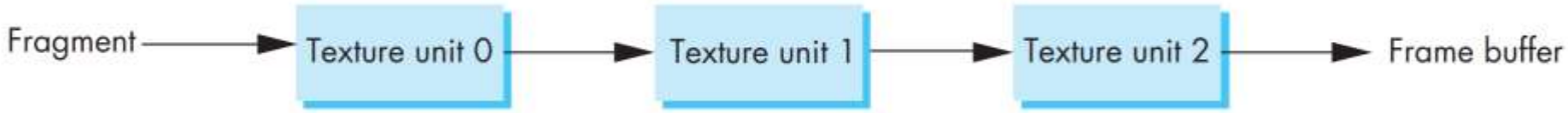


Multi-texturing



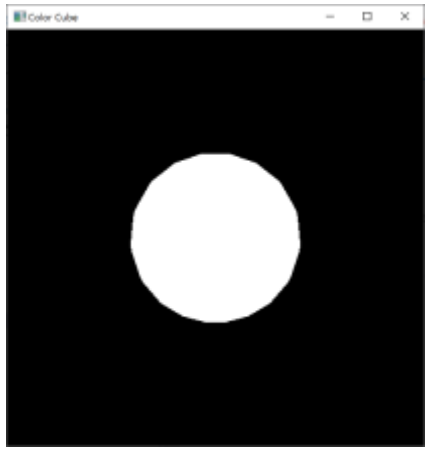


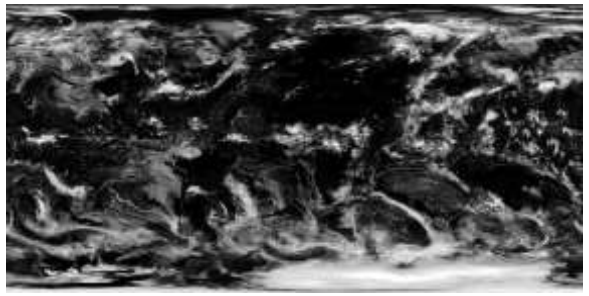
Multi-texturing

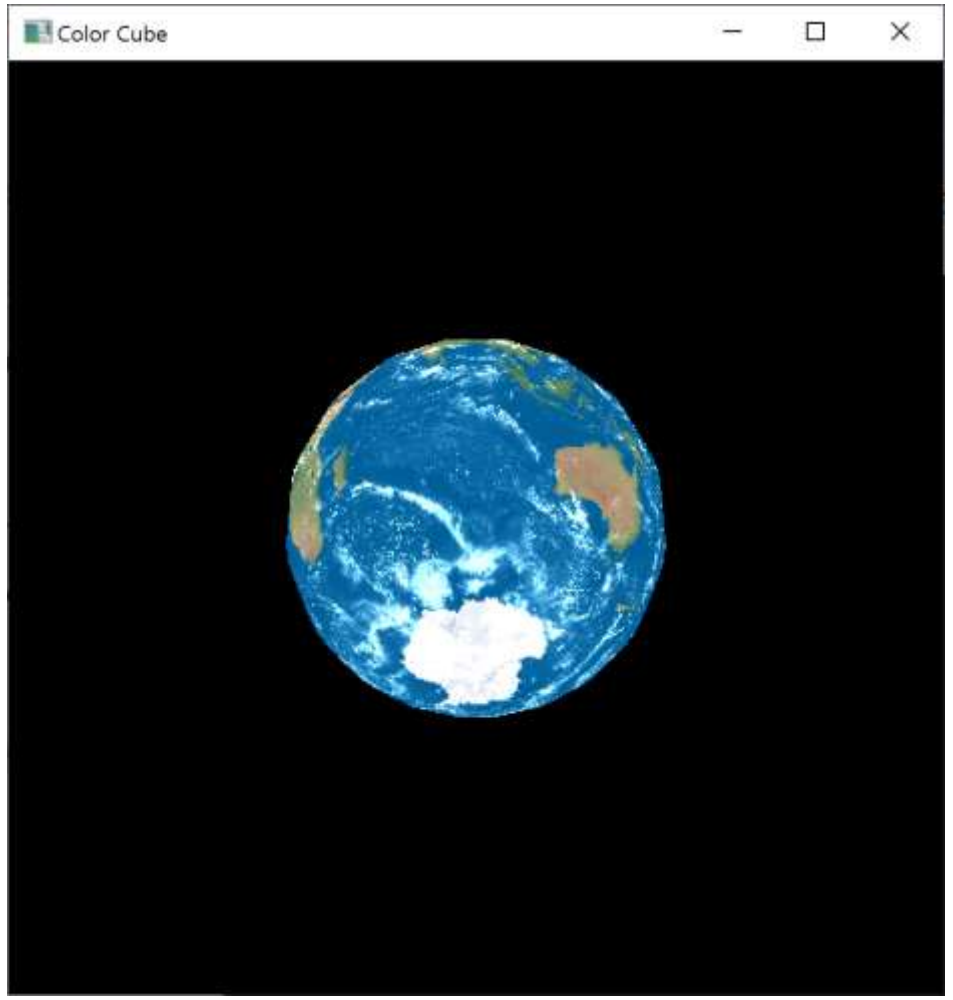



```
glActiveTexture(GL_TEXTURE0); /* unit 0 */
glBindTexture(GL_TEXTURE_2D, object0);
glActiveTexture(GL_TEXTURE1); /* unit 1*/
glBindTexture(GL_TEXTURE_2D, object1);
```

Multi-texturing

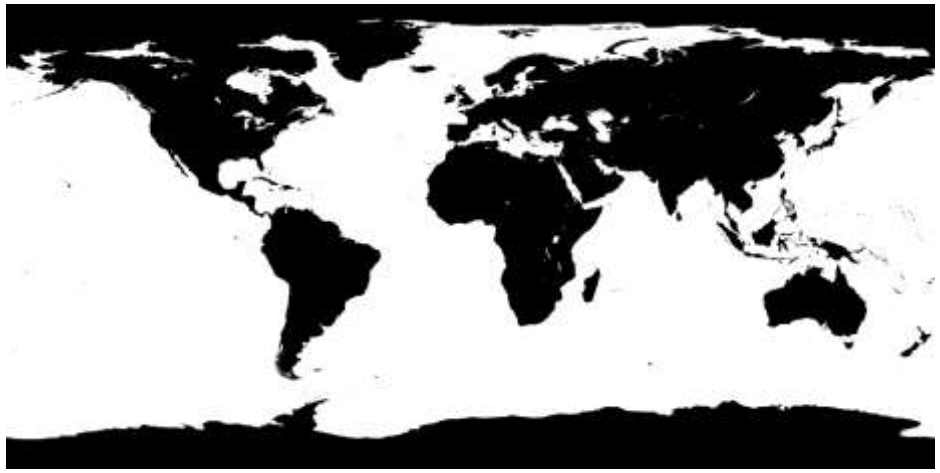






Multi-texturing: Specular Map

- Specify the specular reflectance with the additional texture map

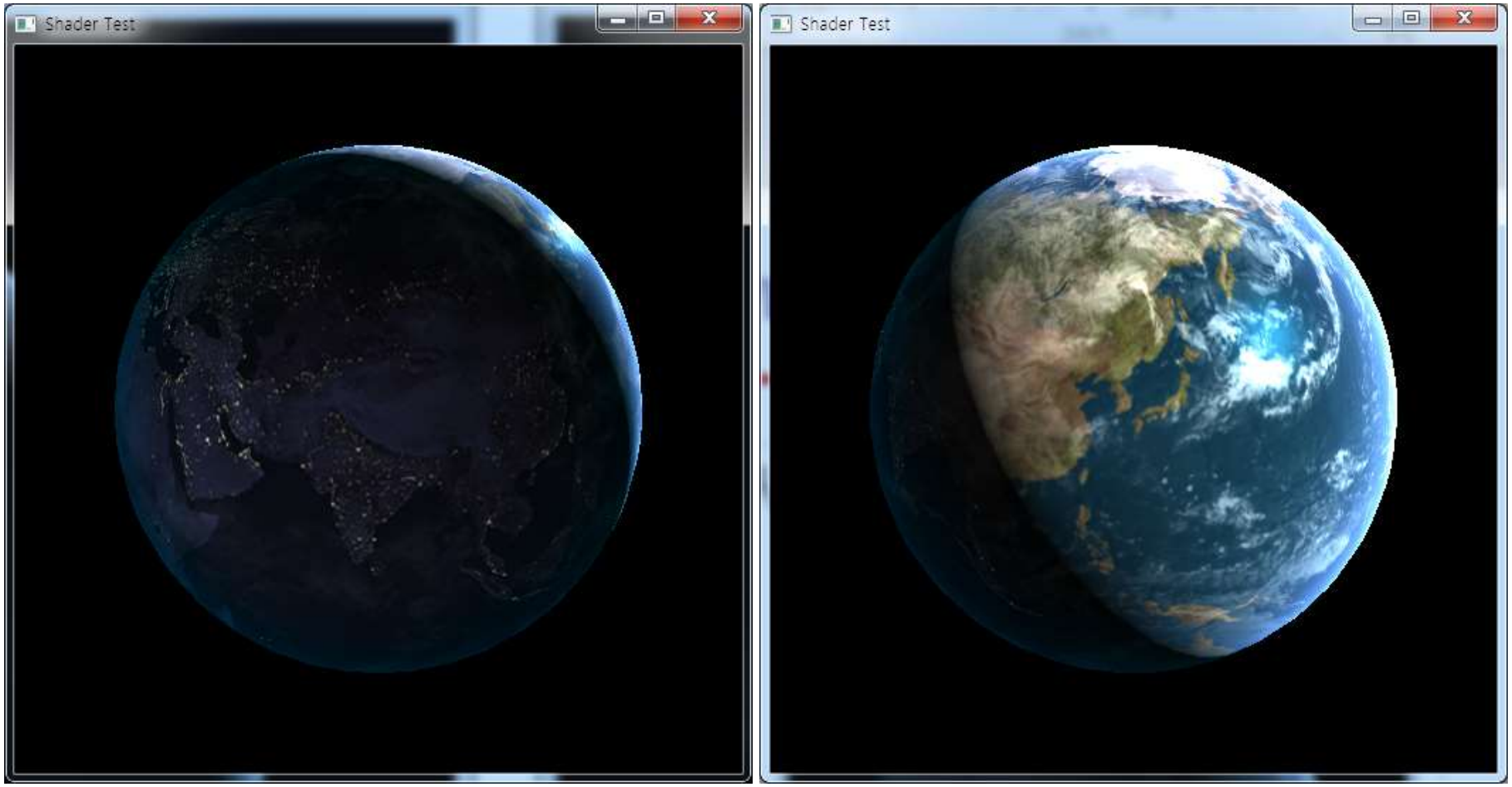


Multi-texturing

Programming Assignment #3

Assignment #3

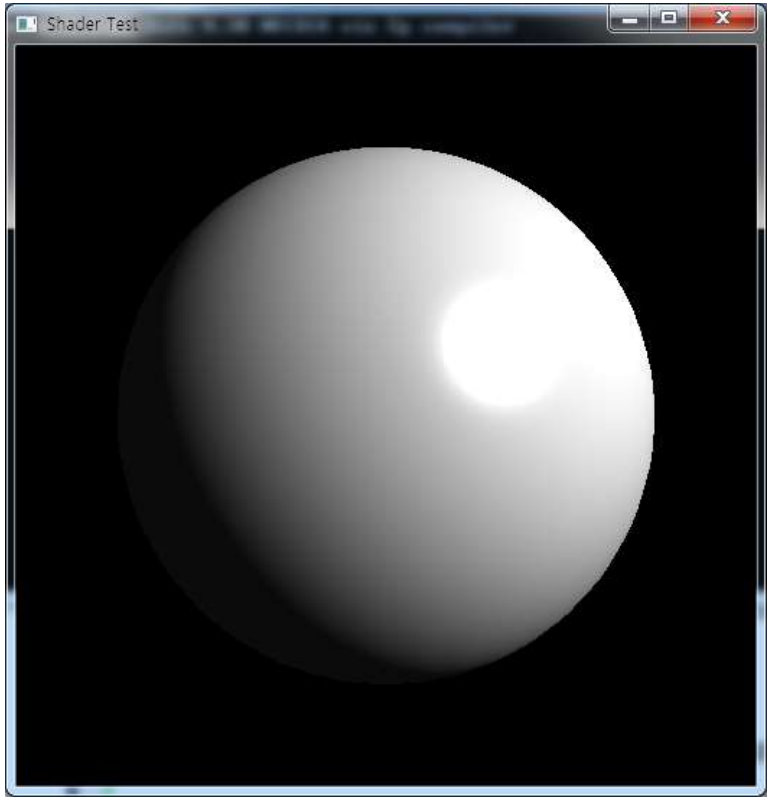
- 좀더 사실적인 지구를 그려보자



Assignment #3

- 구에 lighting 적용하기
- 3개 이상의 텍스쳐를 동시에 사용하여 렌더링
  - 기본 텍스쳐
  - Cloud
  - Reflectance
  - Nighttime
  - Terrain
  - …

참고: http://www.shadedrelief.com/natural3/



Assignment #3

제출물: 하나의 zip파일로 압축

- Main.cpp (소스 파일은 main.cpp로, header들은 필요한 만큼)
- Readme.txt: 구현상의 특이점, 프로그램 사용법, 버그 사항
  - 자신이 어떤 텍스쳐를 이용했는지 설명할 것
- 프로그램 스크린샷
- 제출일: 11월 30일 (목) 23시 59분