텍스처 매핑

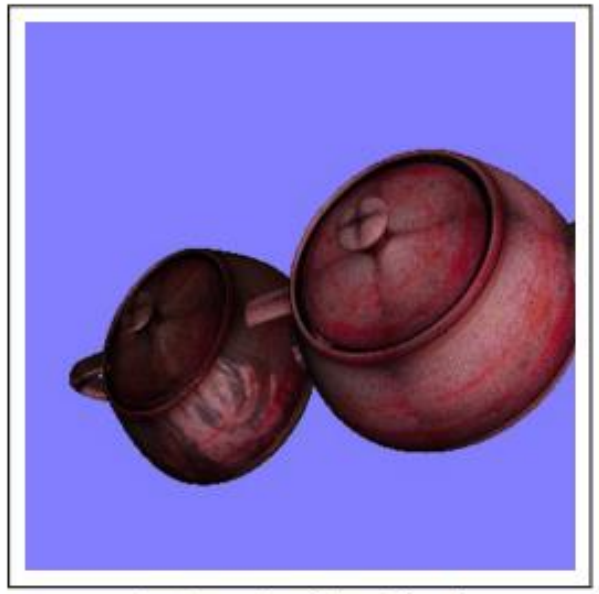
동명대학교 게임공학과
강영민

텍스처 매핑

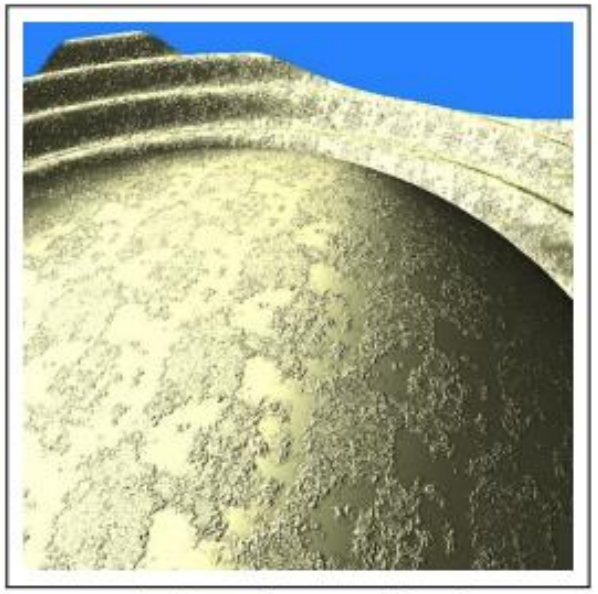
텍스처 매핑은 컴퓨터 그래픽스에서 어떤 물체의 표면에 추가적인 재질감이나, 색상 등을 입히는 작업이다. 자주 사용되는 매핑 몇 가지는 다음과 같다.

- 컬러 매핑: 이미지의 컬러를 표면에 입히는 텍스처 매핑
- 범프 매핑: 이미지를 활용하여 표면에 굴곡이 있는 것처럼 느끼게 만드는 매핑
- 환경매핑: 주변의 환경이 객체에 반사되는 형태로 입혀지는 텍스처 매핑

텍스처 매핑의 결과 예시



(a) 컬러 매핑



(b) 범프 매핑

(c) 환경 매핑

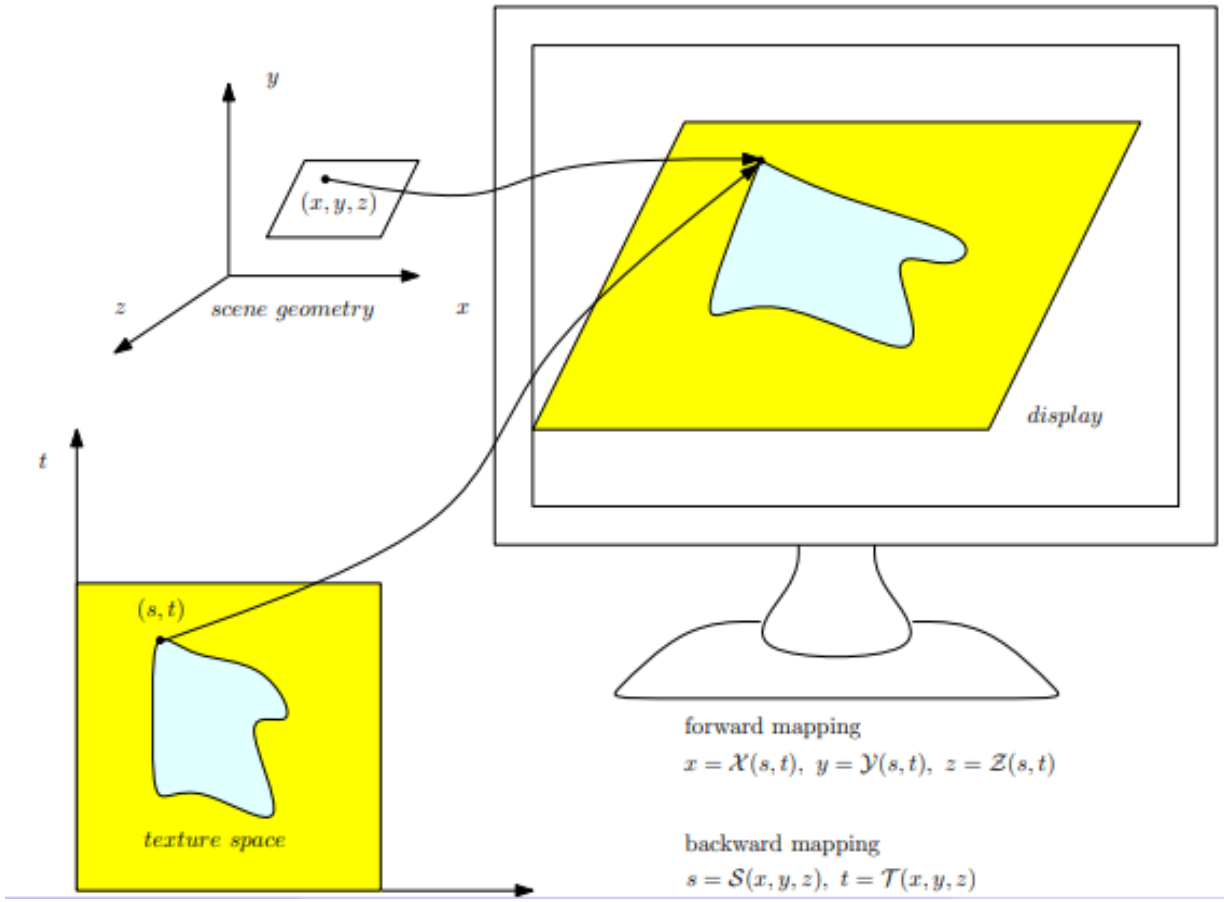
텍스처 매핑의 방법 – 전방(forward) 매핑

- 텍스처 매핑의 아이디어는 매우 간단
- 텍스처 공간에서의 좌표 (s,t)를 텍스처 좌표, 물체 표면의 좌표 (x,y,z)를 전역공간 좌표라 함
- 장면을 표현하는 기하공간의 좌표 (x,y,z)는 실수 원소를 가지는 데에 반해 텍스처 공간의 좌표 (s,t)는 정수 원소
- 하나의 정수 좌표에 의해 얻어지는 텍스처 이미지 상의 한 점을 텍셀(texel)
- 화면에 그려질 때 화면 좌표계의 한 점을 화소 혹은 픽셀(pixel)

매핑이라는 것은 텍스처가 표현하는 정보를 장면을 표현하는 기하 공간의 물체로 옮기면 되므로, 이것을 이런 매핑 함수로 구현할 수 있다.

$$x = \mathcal{X}(s,t),\ y = \mathcal{Y}(s,t),\ z = \mathcal{Z}(s,t)$$

텍스처 매핑의 방법 – 후방(backward) 매핑



- 전방 매핑: 텍스처 이미지는 정수개의 픽셀로 구성된 텍셀을 객체에 옮겨 놓게 되면 빈 공간이 생길 수 있음
- 실제 텍스처 매핑은 역방향 매핑(backward mapping)을 수행
- $s = \mathcal{S}(x,y,z),\ t = \mathcal{T}(x,y,z)$

OpenGL 텍스처 매핑

- OpenGL에서 텍스처를 적용하는 과정은 다음 3단계
  - 텍스처 지정 – 이미지 읽기 혹은 생성 후 텍스처로 지정하고 사용 가능하게 설정
  - 텍스처 좌표 지정 – 매핑 함수는 응용 프로그램의 몫
  - 텍스처 파라미터 설정 – 래핑, 필터링 방법 설정

텍스처 설정

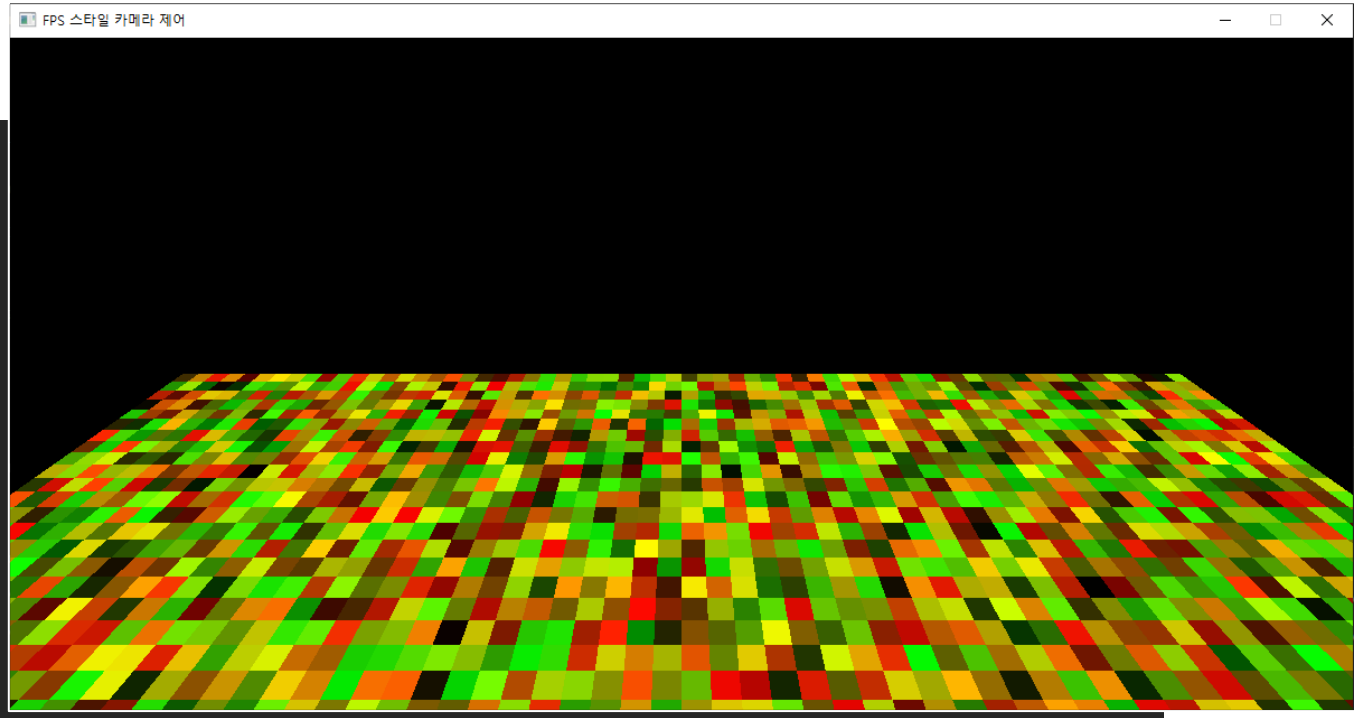
- glTexImage2D – 2차원 텍스처(GL_TEXTURE_2D)를 생성하는 것
  - 크기가 TEXSIZE x TEXSIZE
  - 각각의 원소가 GL_FLOAT로 지정했음 – GL_UNSIGNED_BYTE도 가능
  - 이미지의 각 원소는 GL_RGB – GL_RGBA, GL_LUMINANCE 등도 가능
- glTexParameterf – 텍스처 파라미터 설정
  - 텍스처 좌표가 [0,1] 범위를 넘을 때 어떻게 할 것이지
    - GL_WRAP, GL_CLAMP
  - MAG/MIN 필터는 텍스처가 원래의 크기보다
    - 더 큰 공간으로 가거나 – mag filter
    - 더 작은 공간으로 갈 때 – min filter
- glEnable(GL_TEXTURE_2D)을 통해 설정된 텍스처가 동작함

텍스처 이미지 만들어 설정하기

```
TEXSIZE = 64
def createTexture() :
    return np.random.rand( TEXSIZE, TEXSIZE, 3 )

def SetupTexture(texImage):
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, TEXSIZE, TEXSIZE, 0, GL_RGB, GL_FLOAT, texImage)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
    glEnable(GL_TEXTURE_2D)
```

텍스처 지정과 사용




```
def initializeGL(self):
    myTexture = createTexture()
    SetupTexture(myTexture)

def paintGL(self):


    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(60, 2, 0.2, 100)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    gluLookAt( self.cam[0], self.cam[1] + 0.6, self.cam[2], # 카메라 위치
            self.target[0], self.target[1], self.target[2], # 카메라가 쳐다보는 지점
            0, 1, 0  # 카메라 위쪽 방향
    )

    drawPlane()
```

이미지 로딩

- Pillow
  - pip install Pillow

```
from PIL import Image

TEXSIZE = 64
def createTexture() :
    return np.random.rand( TEXSIZE, TEXSIZE, 3 )

def load_texture(file_path):
    img = Image.open(file_path)
    img_data = img.tobytes("raw", "RGB", 0, -1)

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, img.width, img.height, 0, GL_RGB, GL_UNSIGNED_BYTE, img_data)
```

사용

```
    def initializeGL(self):
        #myTexture = createTexture()
        #SetupTexture(myTexture)
        load_texture('./Textures/image.jpg')
        glEnable(GL_TEXTURE_2D)
```

복수의 텍스처

- 텍스처 핸들
  - 두 개의 텍스처를 다룰 수 있도록 텍스처를 가리키는 번호 필요
- 각각의 텍스처를 사용할 때마다 원하는 텍스처 지정
  - 텍스처 바인딩

텍스처 준비

```
from PIL import Image

# Load image using PIL
def load_texture(file_path):
    img = Image.open(file_path)
    img_data = img.tobytes("raw", "RGB", 0, -1)

    texture = glGenTextures(1)
    glBindTexture(GL_TEXTURE_2D, texture)

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, img.width, img.height, 0, GL_RGB, GL_UNSIGNED_BYTE, img_data)
    print(img.width, img.height)

    return texture
```

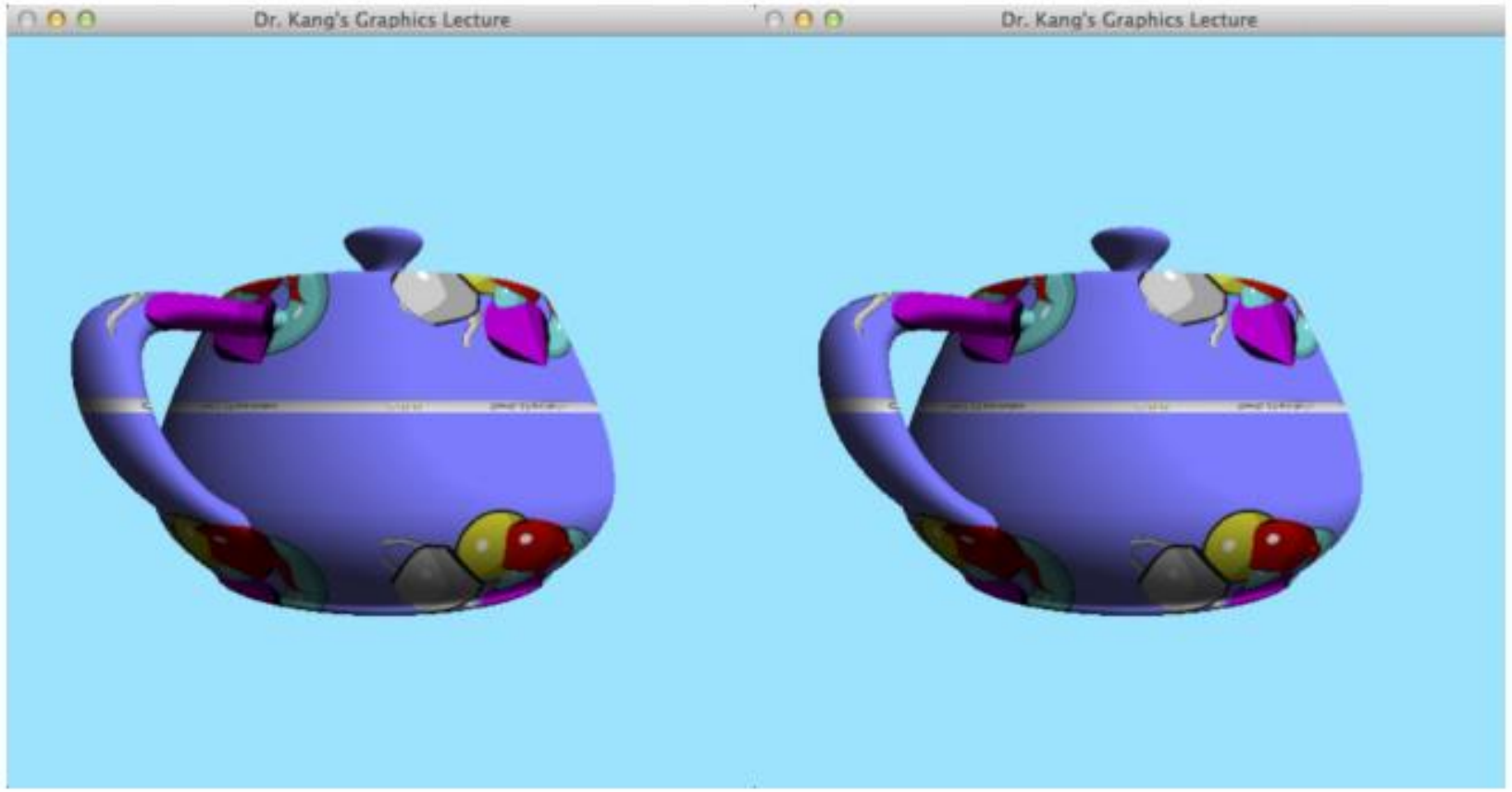
텍스처 좌표 자동 생성

- 텍스처 좌표를 자동으로 생성할 수 있다면, 일일히 텍스처 좌표를 지정하지 않아도 됨
- 환경 매핑과 같은 경우에 이 기법이 유용하게 사용
- OpenGL이 자동으로 각 정점의 텍스처 좌표를 결정하는 방법이 있음
- 이를 위해서는 아래와 같이 자동 생성이 가능하도록 해야 함

```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```

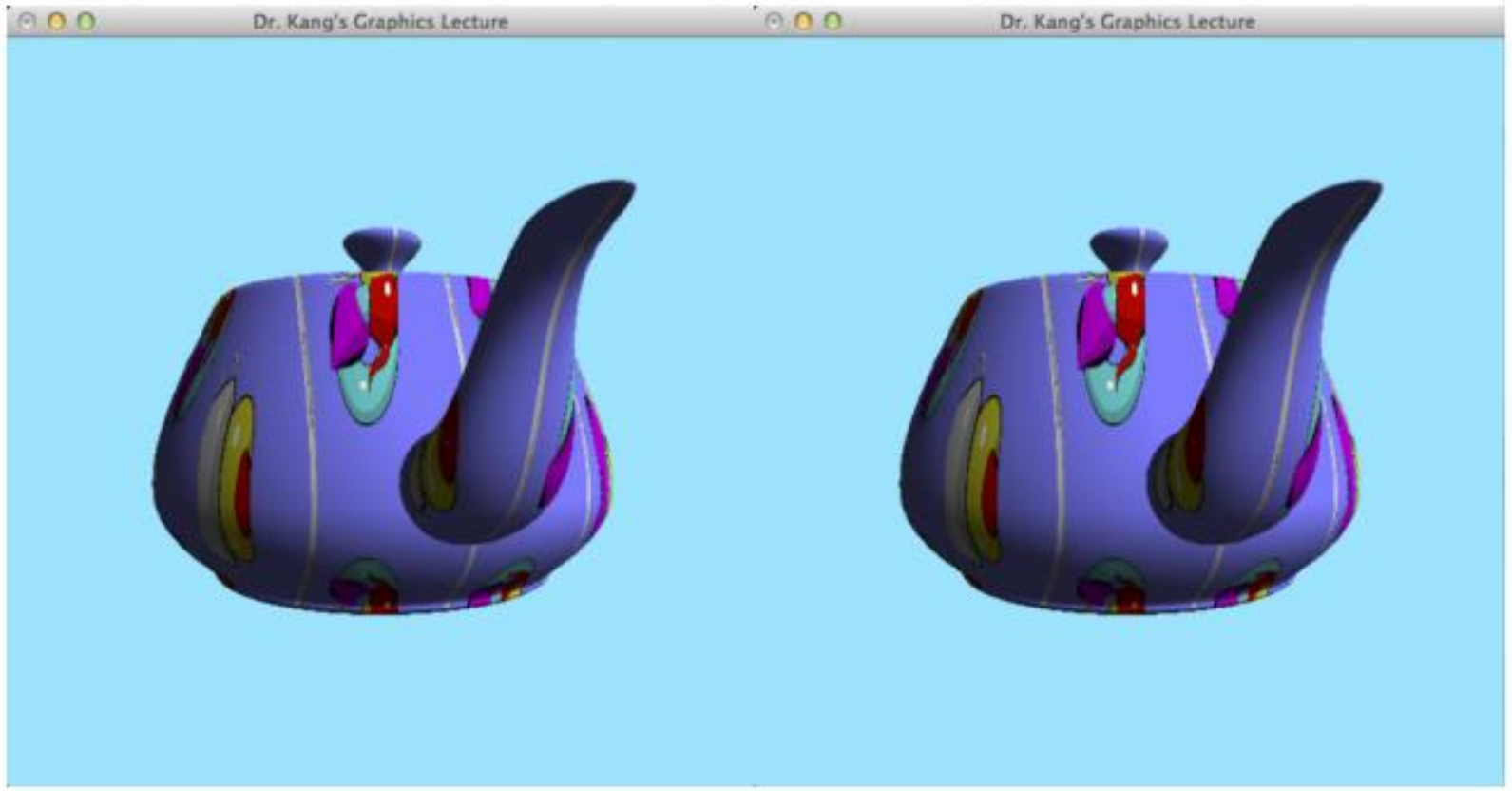
GL_EYE_LINEAR

```
glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
```



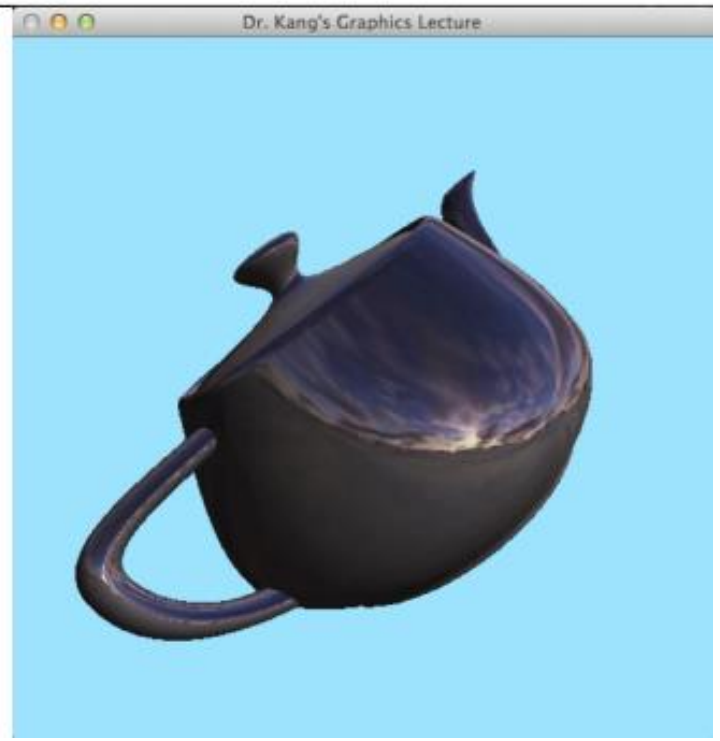
GL_OBJECT_LINEAR

```
glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
```



GL_SPHEREMAP

```
glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
```



메시에 적용하기

```
glBindTexture(GL_TEXTURE_2D, self.tex1)
        glEnable(GL_TEXTURE_GEN_S)
        glEnable(GL_TEXTURE_GEN_T)

        glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR)
        glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR)
        glPushMatrix()
        glRotatef(self.angle, 0, 1, 0)
        self.myLoader.draw_display_list()
        glPopMatrix()

        glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR)
        glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR)
        glPushMatrix()
        glTranslatef(-1.5, 0, 0)
        glRotatef(self.angle, 0, 1, 0)
        self.myLoader.draw_display_list()
        glPopMatrix()

        glBindTexture(GL_TEXTURE_2D, self.tex2)
        glTexGenf(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
        glTexGenf(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
        glPushMatrix()
        glTranslatef(1.5, 0, 0)
        glRotatef(self.angle, 0, 1, 0)
        self.myLoader.draw_display_list()
        glPopMatrix()
```