단국대학교 응용컴퓨터공학 컴퓨터 그래픽스 기말고사 (2017년 가을학기) 2017년 12월 13일 수요일

학과 _____ 학번 _____ 이름 _____

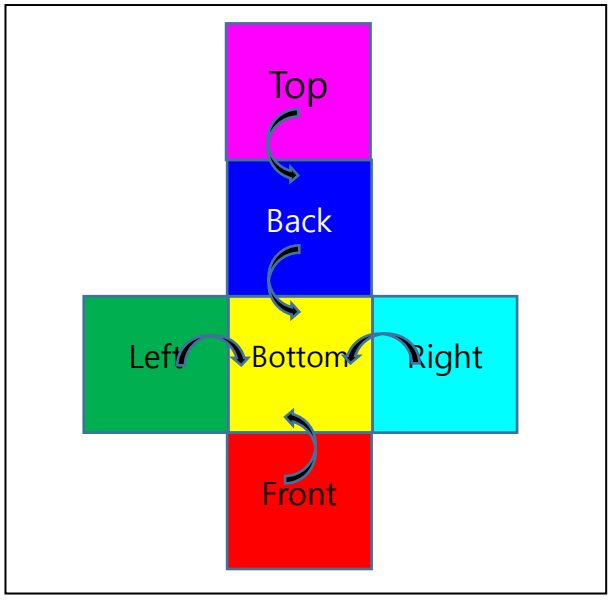
기말고사

담당교수: 박경신

- **답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.**

1. 다음은 SimpleCube3D의 코드 일부를 보여주고 있다. SimpleCube3D의 계층적 구조를 그려라 (각 face에 어떤 tranformation matrix가 적용되는 지 회전방향을 명확히 표시해 줄 것). (10점)

```
void SimpleCube3d::draw(Program* p, glm::mat4& projection, glm::mat4& view, glm::mat4& model) {
        p->useProgram();
        p->setUniform("gProjection", projection);
        p->setUniform("gView", view);
        glm::mat4 m = model * faceTransform[0];
        p->setUniform("gModel", m);
        face[0].draw(); // (1)bottom
        m = model * faceTransform[0] * faceTransform[1];
        p->setUniform("gModel", m);
        face[1].draw(); // (2)left
        m = model * faceTransform[0] * faceTransform[2];
        p->setUniform("gModel", m);
        face[2].draw(); // (3)right
        m = model * faceTransform[0] * faceTransform[3];
        p->setUniform("gModel", m);
        face[3].draw(); // (4)front
        m = model * faceTransform[0] * faceTransform[4];
        p->setUniform("gModel", m);
        face[4].draw(); // (5)back
        m = model * faceTransform[0] * faceTransform[4] * faceTransform[4];
        p->setUniform("gModel", m);
        face[5].draw(); // (6)top
}
void SimpleCube3d::updateTransform() {
        faceTransform[0] = glm::translate(glm::mat4(1.0f), glm::vec3(0, -1, 0));
        faceTransform[1] = glm::translate(glm::mat4(1.0f), glm::vec3(-1, 0, 0)) *
```



```
                glm::rotate(glm::mat4(1.0f), glm::radians(-angle), glm::vec3(0, 0, 1)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(1, 0, 0)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(-2, 0, 0));
        faceTransform[2] = glm::translate(glm::mat4(1.0f), glm::vec3(1, 0, 0)) *
                glm::rotate(glm::mat4(1.0f), glm::radians(angle), glm::vec3(0, 0, 1)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(-1, 0, 0)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(2, 0, 0));
        faceTransform[3] = glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, 1)) *
                glm::rotate(glm::mat4(1.0f), glm::radians(-angle), glm::vec3(1, 0, 0)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, -1)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, 2));
        faceTransform[4] = glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, -1)) *
                glm::rotate(glm::mat4(1.0f), glm::radians(angle), glm::vec3(1, 0, 0)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, 1)) *
                glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, -2));
}
```

2. SimpleCube3D의 draw 함수에서 사용하는 gModel, gView, gProjection가 무엇인지 자세히 설명하라. (5점)

model 변환은 model space에서 구성한 3차원 물체의 정점 좌표를 world space로 변환해주는 것이다.

view 변환은 world space에 배치된 3차원 물체의 정점 좌표를 카메라의 관점인 view space으로 바꿔준다.

projection 변환은 view space에 있는 3차원 물체의 정점 좌표를 정규 장치 좌표계 (normalized device coordinate)로 바꿔준다. 정규 장치 좌표계에서 투영면에 투영시킨다.

3. SimpleCube3D의 gModel, gView, gProjection는 쉐이더(shader) 코드의 uniform 변수로 지정한다. Uniform 변수란 무엇인가? (5점)

쉐이더에서 uniform 변수는 전역 변수이며, primitive 마다 바뀔 수 있고, OpenGL 프로그램에서 쉐이더로 값을 변경하는데 사용함.
Uniform 변수는 vertex shader와 fragment shader 모두에서 사용가능. 쉐어더에서는 상수임.

4. 각각 geometryPositionColor, geometryPositionNormal, geometryPositionNormalTexture 사용한 SimpleCube3D 그리기의 차이점을 설명하라. (5점)

geometryPositionColor는 정점(vertex position)과 색(color)로 된 Quad를 사용하여, 면을 동일한 색으로 그려줌.

geometryPositionNormal는 정점(vertex position)과 법선벡터(normal)로 된 Quad를 사용하여, Lighting 과 Material을 사용하여, 면에 색의 음영이 나타나게 그려줌.

geometryPositionNormalTexture는 정점(vertex position)과 법선벡터(normal)과 텍스쳐좌표(texture coordinate)로 된 Quad를 사용하여, Lighting과 Material과 Texture를 사용하여 면에 색 음영과 텍스쳐가 나타나게 그려줌.

학과 _____ 학번 _____ 이름 _____

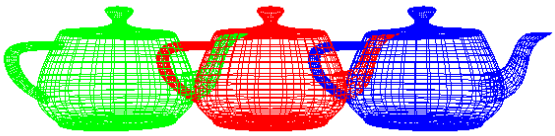
5. 원근 정규화(perspective normalization)를 설명하라. (5점)

원근 정규화는 원근 투영을 직교투영으로 바꾸는 작업이다.

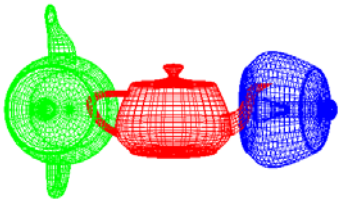
즉, x=±z, y=±z, z=near/far =>x=±1, y=±1, z=±1로 정규화 (normalization).

이 과정에서 원근감이 생성된다.

6. 다음은 같은 물체를 아래의 2가지 방식으로 *90도 Yaw 회전과 90도 Pitch 회전과 90도 Roll 회전을 한 후* 물체의 최종 회전을 보여주고 있다. 그 이유를 설명하라 (즉, 각 회전축이 어딘지, 주전자의 입구와 주전자의 뚜껑이 어디로 향하는지를 정확히 명시할 것) (10점)



yaw(90)



Yawpitch(90)

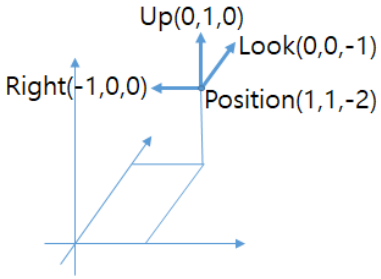
Yawpitchroll(90)

```
// (1) 왼쪽 (초록색)은 World Coordinate System을 중심으로 회전하는 방식
glm::mat4 Ry = glm::rotate(glm::mat4(1), M_PI/2, glm::vec3(0, 1, 0));
glm::mat4 Rx = glm::rotate(glm::mat4(1), M_PI/2, glm::vec3(1, 0, 0));
glm::mat4 Rz = glm::rotate(glm::mat4(1), M_PI/2, glm::vec3(0, 0, 1));
R = Rz * Rx * Ry;
// WCS의 90도 yaw 회전 후 주전자의 입구는 WCS의 z- (주전자 머리는 WCS의 y+)
// WCS의 90도 추가 pitch 회전 후 주전자의 입구는 y+ (주전자 머리는 WCS의 z+)
// WCS의 90도 추가 roll 회전 후 주전자의 입구는 WCS의 x- (주전자 머리는 WCS의 z+)
```

```
// (2) 오른쪽 (파란색)은 Local Coordinate System을 중심으로 회전하는 방식
YPR = glm::yawPitchRoll(M_PI/2, M_PI/2, M_PI/2);
// LCS의 90도 yaw (주전자 머리) 회전 후 주전자의 입구는 WCS의 z- (LCS의 x+축은 WCS의 z-축)
// LCS의 90도 추가 pitch (주전자 입구) 회전 후 주전자의 입구는 z- (주전자 머리는 WCS의 x+)
// LCS의 90도 추가 roll 회전(주전자 옆등) 회전 후 주전자의 입구는 x+ (주전자 머리는 WCS의 z+)
```

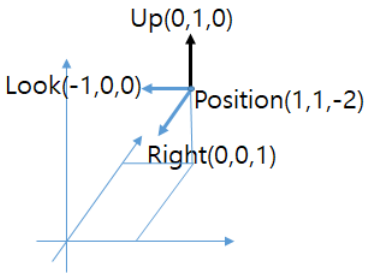
학과 _____ 학번 _____ 이름 _____

7. 다음은 세계 좌표계에서 현재 카메라의 위치와 바라보는 방향을 보여주고 있다. (1) 현재 카메라에서 *90도 Yaw 회전한 후* View1 *(2)* 현재 카메라에서 *90도 Pitch 회전한 후* View2, *(3)* 현재 카메라에서 *90도 Roll 회전한 후*의 카메라 View3 Matrix를 계산하라. 각 회전마다 Right, Up, Look 벡터의 방향을 그림으로 도식화하고 View Matrix를 계산한다. (10점)

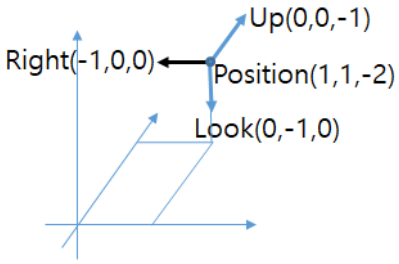


$$View = \begin{pmatrix} r_x & r_y & r_z & -p \bullet r \\ u_x & u_y & u_z & -p \bullet u \\ l_x & l_y & l_z & -p \bullet l \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

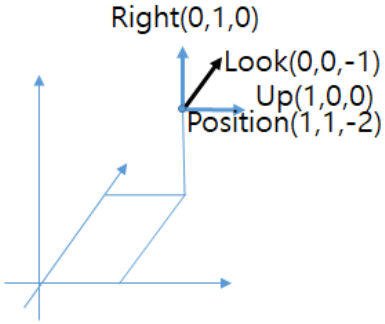
Yaw90 회전 후



Pitch90 회전 후



Roll90 회전 후



$$View1 = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} View2 = \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} View3 = \begin{pmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

8. (정점이 4개인) Quad와 (100개의 Quad로 구성된) Grid를 각각 Gouraud Shading과 Phong Shading를 사용했을 때 차이점을 서술하라. (10점)

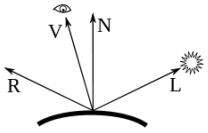
Gouraud shading은 일명 smooth shading라고도 불리며, 폴리곤의 각 정점의 색을 linear interpolation하여 내부를 부드럽게 보간하는 방식이다. (**Per-vertex shading**)
정점에서 lighting연산이 된 Gouraud shading을 Quad에 사용했을 시 정점이 4개이므로 제대로 된 명암 표현이 되지 않는다.
반면 Grid를 사용했을 시 정점이 10000개이므로 Gouraud shading의 Quad와 비교하여 좀 더 부드럽고 자연스러운 명암이 표현이 된다.

Phong shading은 (Gouraud shading이 정점의 색을 이용하여 내부를 보간하는 반면) 폴리곤의 각 정점의 normal을 가지고 폴리곤 내부의 각 픽셀마다 normal이 보간되어 specular reflection이 더욱 정확하게 나타나게 하는 방식이다. (**Per-pixel shading**)
pixel에서 lighting연산이 된 Phong shading을 Quad에 사용했을 시 정점의 법선벡터를 보간함으로서 Gouraud shading과 비교하여 제대로 된 명암표현을 (즉 더 자연스러운 경면광 표현) 할 수 있다.
Grid를 사용했을 시 Quad와 비교하여 좀더 나은 경면광 표현을 볼 수 있다. 또한 Gouraud shading된 Grid가 색의 보간으로 인한 뭉게짐 현상이 나타나는 반면 Phong shading의 Grid는 경면광 표현이 제대로 표현된다.

학과 _____ 학번 _____ 이름 _____

9. 다음은 조명 (lighting)에 관한 문제이다. 아래의 질문에 답하시오. (각 5점 총 25점)

$$I = K_a I_a + \sum_{i=0}^{m-1} f_{att}(d)\{K_d I_d (N \bullet L) + K_s I_s (R \bullet V)^n\} + E$$



1) 퐁 직접 조명 모델 (Phong Reflection Model)에서 환경반사 (ambient reflection), 난반사 (diffuse reflection), 정반사 (specular reflection)가 무엇인지, 위의 공식에서 어느 부분인지 설명하라.

K_aI_a : 환경반사 (ambient reflection)는 광원에 직접 노출되지 않는 면에 밝기를 부여함
K_dI_d (N · L): 난반사 (diffuse reflection)는 면의 밝기가 입사각 (광원벡터와 법선 벡터의 사이각)의 코사인에 정비례한다는 램버트 코사인 법칙에 따라서, 물체의 색에 면이 서있는 방향에 따라 차등적 밝기를 부여함
K_sI_s $(V \cdot R)^n$: 퐁 모델은 Reflection Vector, R을 사용하여, shiny surface에서 정반사 (specular reflection) 경면광(물체의 색이 아니라 광원의 색)을 계산해주며, 시점이 정확히 반대방향일 때 보임

2) 정반사(specular reflection)에 사용하는 반사 벡터가 무엇인지 자세히 설명하라.

정반사 (specular reflection)에서 퐁 모델의 경우 Ks Is $(R \bullet V)^n$ 을 사용하는데, 광원을 향하는 벡터 L이 표면의 법선 벡터 N에 정반사된 벡터 R을 모든 정점에 대해 계산해야 한다.

3) L(광원벡터), N(법선벡터), V(시점벡터), R (반사벡터)가 무엇인지 설명하라.

L은 정점에서 광원으로 향하는 벡터 (light vector)
N은 정점에서 법선벡터 (normal vector)
V는 정점의 위치에서 카메라의 위치로 향하는 시점 벡터 (view vector)
R은 정점의 위치에서 광원벡터 (L)가 정점의 법선벡터 (N)에 정반사된 벡터 (reflection vector)

4) 램버트 법칙 (Lambertian Law)을 자세히 설명하라.

램버트 코사인 법칙은 난반사 (diffuse reflection)에서 면의 밝기가 입사각 (즉, 광원 벡터와 법선 벡터의 사이각)의 코사인에 정비례 함을 말하는 것이다. 즉, 면이 서 있는 방향에 따라 차등적 밝기를 제공하여 입체감을 부여할 수 있다.
Fragment shader 코드에서 cosTheta는 법선벡터 (N)과 광원벡터 (L)간의 내적은 두 벡터간의 입사각의 코사인과 비례하며, diffuse reflection과의 곱에서 사용된다.

5) Point Light (점광원), Directional Light (방향성광원), Spot Light (점적광원)을 설명하라. 아래의 코드에서 Point Light (점광원), Directional Light (방향성광원), Spot Light (점적광원)을 지정하는데 필요한 변수를 찾아 설명하라.

```
class Light
{
public:
        //중간생략..
        glm::vec4 Position; // position
        glm::vec4 Ambient; // ambient light color
        glm::vec4 Diffuse; // diffuse light color
        glm::vec4 Specular; // specular light color
```

```
        float SpecularShininess; // specular shininess
        float ConstantAttenuation; // light attenuation
        float LinearAttenuation;
        float QuadraticAttenuation;
        float SpotCutoff;      // if (SpotCutoff < 180) then spotlight
        float SpotExponent;
        glm::vec3 SpotDirection;
};
```

점광원 – 광원의 점 (즉, position의 w=1)을 지정하며, 광원의 색 (즉, ambient, diffuse, specular)을 정의하고, attenuation 0, 1, 2를 사용하여 거리에 따라 빛의 세기가 약해지는 정도를 정의한다.

방향성광원 – 광원의 방향 (즉, position의 w=0)을 지정하며, 광원의 색 (즉, ambient, diffuse, specular)을 정의하여 사용한다.

점적광원 – 광원의 점 (즉, position의 w=1)과 방향(direction)을 지정하며, 광원의 색 (즉, ambient, diffuse, specular)를 정의하고, attenuation 0, 1, 2를 사용하여 거리에 따라 빛의 세기가 약해지는 정도와 theta와 phi를 사용하여 원뿔의 안쪽과 바깥쪽 각도를 정의할 수 있다.

10. 다음 OpenGL 텍스쳐 매핑(Texture Mapping)과 블렌딩(Blending) 질문에 답하라. (각 5점 총 15점)

1) Multi-pass Multi-texturing의 원리를 설명하라. Single-pass Multi-texturing과의 차이점은 무엇인가?

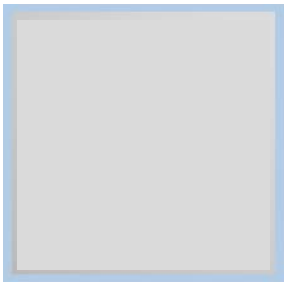
다중패스 멀티 텍스쳐링 (multi-pass multi-texturing) 방식은 동일한 다각형 자체를 여러 번 렌더링 하는 것으로, 배경이미지를 먼저 그리고 난 후, 그 위에 라이트매핑 이미지를 합성(blending)하여 여러 번 다각형을 그리는 것이다.

싱글패스 멀티 텍스쳐링 (single-pass multi-texturing) 방식은 한 다각형에 여러 개 텍스쳐 좌표를 지정하여, 배경이미지와 라이트매핑 이미지를 동시에 그리는 것이다.

2) Quad의 p1(-1,-1,0), p2(1,-1,0), p3(1,1,0), p4(-1,1,0)에 텍스쳐 좌표(texture coordinates)를 t1(-1,-1), t2(0,-1), t3(0,0), t4(-1,0)로 지정했을 때, texture wrapping 방식 (1)GL_REPEAT, (2) GL_MIRRORED_REPEAT, (3)GL_CLAMP_TO_EDGE 따른 출력 결과를 그려라. (opengl.jpg 사용)

(1) GL_REPEAT

(2) GL_MIRRORED_REPEAT



(3) GL_CLAMP_TO_EDGE

학과 _____ 학번 _____ 이름 _____

3) ***glBlendFunc(...);*는 블렌딩 필터를 지정하는 함수이다. 아래의 표를 채워라.**

glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);	***As*Cs + (1-As)*Cd***	*Alpha Blending*
glBlendFunc(GL_ONE, GL_ONE);	***1*Cs + 1*Cd = Cs + Cd***	*Additive Blending*
glBlendFunc(GL_ZERO, GL_SRC_COLOR)	***0*Cs + Cs*Cd = Cs*Cd***	*Modular Blending*
glBlendFunc(GL_ONE, GL_ZERO);	***1*Cs + 0*Cd = Cs***	*No blending*
glBlendFunc(GL_ZERO, GL_ONE);	***0*Cs + 1*Cd = Cd***	*Draw background only*
glBlendFunc(GL_SRC_ALPHA, GL_ONE);	***As*Cs + Cd***	*Brighten the scene*
glBlendFunc(GL_ZERO, GL_SRC_ALPHA);	***0*Cs + As*Cd = As*Cd***	*Darken the scene*

- 끝 -