단국대학교 멀티미디어공학 그래픽스 프로그래밍 기말고사 (2011년 봄학기) 2011년 6월 14일

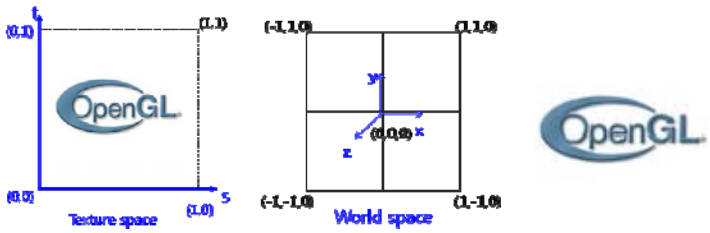
학과 _____ 학번 _____ 이름 _____

기말고사

담당교수: 단국대학교 멀티미디어공학전공 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 OpenGL 환경에서 텍스쳐 매핑과 블렌딩에 관한 문제이다. 아래의 질문에 답하시오. (45점)



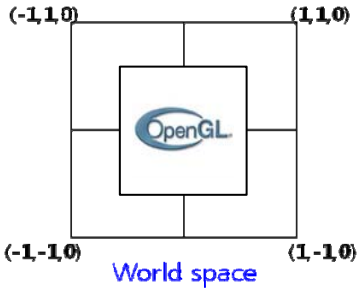
1) 다음은 Quad 메쉬를 그리는 OpenGL 프로그램의 일부이다. 위의 그림 결과와 같이 나타나게 하기 위하여 빈 칸의 텍스쳐 좌표를 채우시오. (10점)

```
void drawTextureQuad()
{
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, textureID);
glBegin(GL_QUADS);
glNormal3f(0, 0, 1);
    glTexCoord2f(0.0, 0.0);
    glVertex3f(-1, -1,  0);
    glTexCoord2f(0.5, 0.0);
    glVertex3f( 0, -1,  0);
    glTexCoord2f(0.5, 0.5);
    glVertex3f( 0,  0,  0);
    glTexCoord2f(0.0, 0.5);
    glVertex3f(-1,  0,  0);
    glTexCoord2f(__0.0__, __0.5__);
    glVertex3f(-1,  0,  0);
    glTexCoord2f(__0.5__, __0.5__);
    glVertex3f( 0,  0,  0);
    glTexCoord2f(__0.5__, __1.0__);
    glVertex3f( 0,  1,  0);
    glTexCoord2f(__0.0__, __1.0__);
    glVertex3f(-1,  1,  0);
    glTexCoord2f(__0.5__, __0.0__);
    glVertex3f( 0, -1,  0);
    glTexCoord2f(__1.0__, __0.0__);
    glVertex3f( 1, -1,  0);
    glTexCoord2f(__1.0__, __0.5__);
    glVertex3f( 1,  0,  0);
```

```
        glTexCoord2f(__0.5__, __0.5__);
        glVertex3f( 0,   0,   0);
        glTexCoord2f(__0.5__, __0.5__);
        glVertex3f( 0,   0,   0);
        glTexCoord2f(__1.0__, __0.5__);
        glVertex3f( 1,   0,   0);
        glTexCoord2f(__1.0__, __1.0__);
        glVertex3f( 1,   1,   0);
        glTexCoord2f(__0.5__, __1.0__);
        glVertex3f( 0,   1,   0);
glEnd();
glDisable(GL_TEXTURE_2D);
}
```

2) 1)번 문제의 Quad 메쉬 OpenGL 프로그램에 텍스쳐 랩 (wrap) 모드는 GL_CLAMP을 사용하고 모두 동일한 렌더링 인자를 사용하는 상황에서, 아래의 OpenGL 코드에서 보이는 텍스쳐 행렬 스택만 조작하여 삽입하였을 때 나타나는 결과 화면을 아래 Quad 메쉬 위에 그림으로 표시하고 설명하라. (10점) (힌트: 정점은 그대로인 상태에서 텍스쳐 좌표계만 변환)

```
glMatrixMode(GL_TEXTURE);
glTranslatef(0.5, 0.5, 0);
glScalef(2, 2, 1);
glTranslatef(-0.5, -0.5, 0);
drawTextureQuad();
```



사각형 메쉬에 적용된 텍스쳐 이미지의 크기가 Pivot (0.5, 0.5)를 중심으로 1/2로 줄어서 나타난다.

3) 1) 2)번 문제의 Quad 메쉬를 그리는 OpenGL 프로그램에서, 만약 조명 (Lighting)이 작동되지 않는다면, 무엇이 문제인지 해결 방법을 자세히 설명하라. (10점) (힌트: 텍스쳐 매핑을 위한 환경 변수 설정)

OpenGL에서 default texture environment는 GL_MODULATE, 즉 조명 (lighting)과 텍스쳐 매핑 (texture mapping) 색이 혼합이 되는 것이 기본값이다.

그런데, 만약 코드에서 glTexEnv (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL); /* or GL_REPLACE */로 되어 있다면 텍스쳐 매핑의 색만을 이용하게 되므로 조명(Lighting)이 적용되지 않는다. 따라서 GL_DECAL을 GL_MODULATE로 바꾸거나, 또는 glTexEnv 부분을 지워버리면 (기본값이 GL_MODULATE가 사용되므로) 조명이 작동되기 시작한다.

4) 다음은 OpenGL Extension에서 제공하는 Point Sprites에 대해 자세히 설명하라. (5점)

포인트 스프라이트란 하드웨이 기반의 빌보드 기법으로, 사용 예로써 파티클 시스템 (Particle System)에서 다수의 입자(Particle)을 텍스쳐 사격형 메쉬 (Quad) 대신 점(Point)를 사용하여 지정하는 기법이다. GL_ARB_point_parameters를 하드웨어가 지원하는지 확인한 후, 포인트 스프라이트로 texture coordinate을 지정하고 enable하기 위해, 환경변수로 포인트스프라이트를 활성화한다.
glTexEnvf(GL_POINT_SPRITE_ARB, GL_COORD_REPLACE_ARB, GL_TRUE);
glEnable(GL_POINT_SPRITE_ARB);
그리고 draw 함수 내부에서 포인트스프라이트를 활성화하고 점(Point)로 그림을 그린다.
glEnable(GL_POINT_SPRITE_ARB);
glBegin(GL_POINTS); .... glEnd();
glDisable(GL_POINT_SPRITE_ARB);

5) 다음은 블렌딩 방식에 따른 SourceFactor와 DestinationFactor값을 빈 칸에 적어라. (5점)

블렌딩 공식: C = SourceFactor*C_s + DestinationFactor*C_d
알파 블렌딩 (alpha blending): C = (Sa, Sa, Sa, Sa) * C_s + (1-Sa, 1-Sa, 1-Sa, 1-Sa) * C_d
SourceFactor는 Source의 alpha값인 (Sa, Sa, Sa, Sa)
DestFactor는 Source의 inverse alpha값인 (1-Sa, 1-Sa, 1-Sa, 1-Sa)

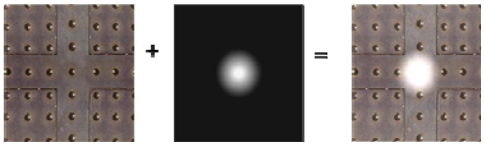
블렌딩 공식에서 Cs와 Cd는 무엇인가?
Cs은 현재 그리고자 하는 색이고, Cd는 프레임버퍼에 있는 색

덧셈 블렌딩 (addition blending) 의 경우
SourceFactor는 ___(1, 1, 1, 1)_____
DestinationFactor는 __(1, 1, 1, 1)_____
C = __Cs + Cd_____

곱셈 블렌딩 (multiply blending)의 경우,
SourceFactor는 __(0, 0, 0, 0)_____
DestinationFactor는 __Cs_____
C = ___Cs * Cd_____

블렌딩을 설정한 채 불투명한 것을 그리고자 할 (no blending) 경우
SourceBlendFactor는 __(0, 0, 0, 0)_____
DestBlendFactor는 __(1, 1, 1, 1)_____
OutputPixel = ___Cd_____

6) 블렌딩을 이용한 라이트 매핑(light mapping)은 어떻게 만드는지 자세히 설명하라. (5점)

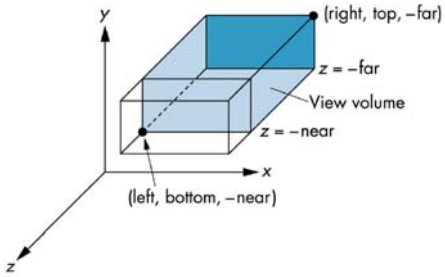


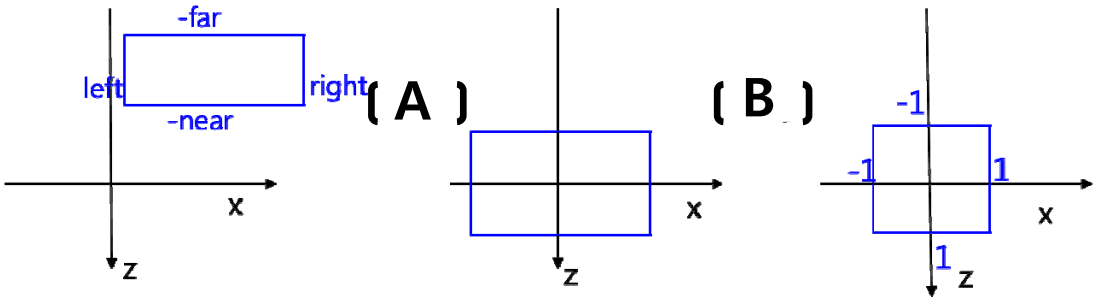
라이트 매핑 (light-mapping): 물체 면의 밝기를 직접 계산하는 대신 여러 개의 텍스쳐를 혼합하여 그 영상을 물체의 면에 입혀서 조명 효과를 주는 것이다.

다중패스 멀티 텍스쳐링 (multi-pass multi-texturing) 방식으로 제작한 라이트 매핑은 동일한 장면(또는 다각형) 자체를 여러 번 렌더링하는 것으로, textureID1 (배경이미지)를 그리고 난 후, 동일한 다각형 자체에 textureID2 (라이트매핑 이미지)를 합성(blending)하여 여러 번 다각형을 그려야 한다.

2. 관측과 관련된 다음 문제에 답하시오. (25점)

1) 다음은 glOrtho(left, right, bottom, top, near, far) 함수가 직육면체 관측공간에서 정규화된 관측 공간으로 변환시키는 직교 투영 행렬 (orthographic projection matrix)을 생성해 내는 과정을 보여주고 있다. 각 단계별로 필요한 (A)와 (B) 변환행렬을 자세히 설명하시오. (5점)



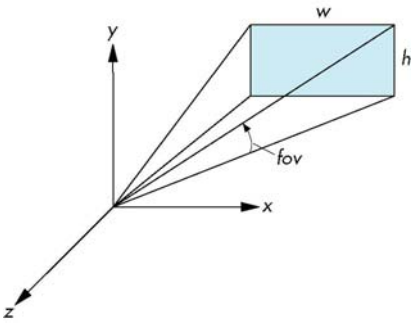


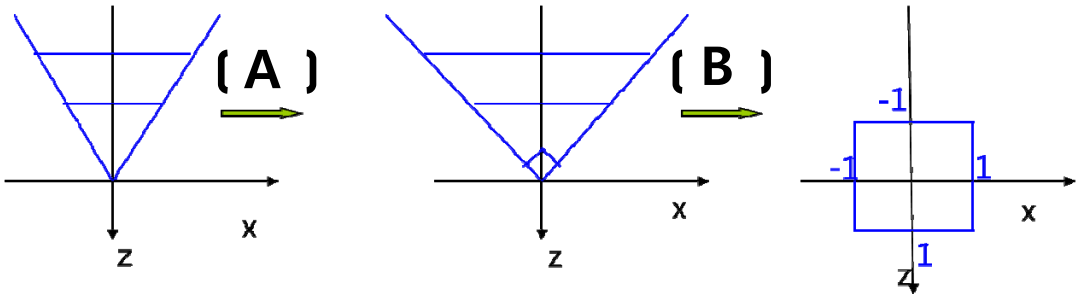
(A) 행렬: 지정된 관측공간의 중심을 정규관측공간의 중심으로 이동시키는 행렬 T을 적용

(B) 행렬: $x = \pm 1, y=\pm 1, z=\pm 1$ 로 바꿔주기 위하여 관측공간의 변의 길이가 2가 되도록 크기 변환 행렬 S 를 적용

학과 _____ 학번 _____ 이름 _____

2) 다음은 gluPerspective(fov, aspect, near, far) 함수가 절두체에서 정규화된 관측 공간으로 변환시키는 원근 투영 행렬 (perspective projection matrix)을 유도하는 과정을 보여주고 있다. 각 단계별로 필요한 (A)와 (B) 변환행렬을 자세히 설명하시오. (5점)

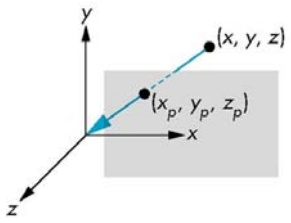




(A) 행렬: 지정된 관측공간을 *x=±z , y=±z, z = near/far* 로 크기변환 행렬 S을 적용

(B) 행렬: *x=±z , y=±z, z = near/far* 관측공간을 x=±1, y=±1, z=±1로 바꿔주는 원근정규화(perspective normalization) 행렬 N을 적용

3) 다음 M_{pers}는 3차원 공간의 한 점 p (x, y, z)를 투영면의 한 점 q (x, y)로 투영하는 원근 투영 행렬(Perspective Projection Matrix)이다. M_{pers} 4x4 행렬을 유도하라. (5점)



$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \mathbf{M_{pers}} \Rightarrow \quad \mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

$\mathbf{M_{pers}} =$ ( )

학과 _____ 학번 _____ 이름 _____

4) 다음 OpenGL 코드에서 밑줄 친 (A) gluLookAt 함수를 제거하고, 화면에 동일하게 그림이 나타나도록 OpenGL 기본 변환함수인 glTranslatef, glRotatef, glScalef 등을 사용하여 코드를 작성하라. (5점)

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // (A)
    glutWireTeapot(2);
    glutSwapBuffers();
}
```

glutLookAt함수는 eye(0, 0, 5), at(0, 0, 0), up(0, 1, 0) 이므로
이를 대체하여 기본변환함수를 사용한다면 glTranslatef(0.0, 0.0, -5.0)를 사용하면 된다.

5) Depth Fighting이 무엇인지 간단히 서술하시오. (5점)

깊이 버퍼에 깊이 값은 한정된 해상도를 갖고 있다. 그런데 깊이 버퍼에서 아주 가까운 깊이 값(depth value)을 가지는 폴리곤의 중첩(overlap)은 "depth-fighting"을 만든다 폴리곤이 그려질 때 부동 소수점 반올림 에러 (floating point round-off errors) 때문에 생기는 현상으로, 폴리곤 임의의 부분이 서로 렌더링 하려는 현상이다.
z-값 같은 면에 multi-ass rendering을 할 경우 Depth Fighting이 생길 수 있는데 glDepthFunc (GL_LEQUAL)을 사용하여 새로운 z-값이 기존의 z-값보다 작거나 같으면 통과시킴으로써 Depth Fighting이 일어나지 않도록 한다 (기본값은 glDepthFunc(GL_LESS)로 새로운 z-값이 기존의 z-값보다 작으면 통과 즉 시야에서 먼 쪽에 있어서 보이지 않는 면은 그리지 않도록 함).

3. 다음은 조명 (lighting)과 쉐이딩 (shading)에 관한 문제이다. 아래의 질문에 답하시오. (30점)

1) 다음은 퐁 조명 모델 (Phong Illumination Model) 공식을 보여주고 있다. 램버트 코사인 법칙 (Lambert's Cosine Law)이 무엇인지 설명하고 이 공식에서 어느 변수인지 서술하라. (5점)

$$I = K_a I_a + \sum_{i=0}^{m-1} \frac{1}{a + bd + cd^2} \left\{ K_d I_d (N \bullet L) + K_s I_s (R \bullet V)^{\alpha} \right\} + E$$

램버트 코사인 법칙에 따르는 빛은 확산 반사 (diffuse reflection)로,
면의 밝기가 입사각 (광원벡터와 법선 벡테의 사이각)의 코사인에 정비례한다.

$$Diffuse\ Reflection \propto \cos\theta$$

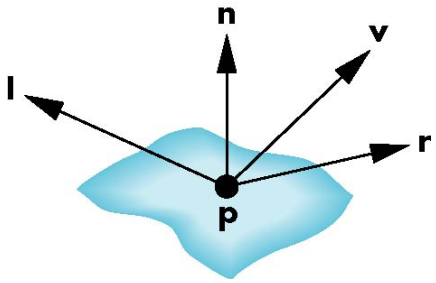
$$Diffuse\ Reflection = K_d I_d \cos\theta = K_d I_d (N \bullet L)$$

2) 이 공식에서 R • V 대신 다른 변수를 사용하여 표현하라. (힌트: OpenGL에서 기본으로 사용하는 조명 모델) (5점)

정반사 (specular reflection)을 Ks Is $(R \bullet V)^n$ 을 사용하는 대신
중간각 H = L + V / | L +V |를 사용하여, Ks Is $(N \bullet H)^{n'}$을 사용한다.

학과 _____ 학번 _____ 이름 _____

3) 카메라의 위치 V (2.0, 2.0, 1.0)이고, 광원의 위치가 (-2.0, 3.0, -1.0)이고, 물체 표면의 정점 좌표 P (0.0, 0.0, 0.0)이고, 표면의 법선 벡터 N (0.0, 1.0, 0.0) 일 때, 정반사 계산에 쓰이는 반사 벡터 R의 값을 구하는 계산과정을 보여라. (5점)



L • N = (-2.0, 3.0, -1.0) dot (0.0, 1.0, 0.0) = 3.0
R = 2 (L • N) N - L
= 2 * 3 * (0.0, 1.0, 0.0) – (-2.0, 3.0, -1.0)
= (0.0, 6.0, 0.0) – (-2.0, 3.0, -1.0)
= (2.0, 3.0, 1.0)

4) OpenGL에서 기본값으로 설정된 세이딩(Shading)이 무엇인지 간단히 설명하라. (5점)

OpenGL에서 기본으로 사용하는 세이딩은 구로우 세이딩 (Gouraud shading)으로, Gouraud shading은 정점의 색으로부터 내부면의 색을 선형보간함. 정점의 법선벡터를 요함. OpenGL에서는 smooth shading이라고 함. 면에 전체적으로 부드러운 음영을 제공함. 그러나 경면광을 감안하지 않음 (실제적인 정점의 법선벡터와 근사적으로 계산된 법선벡터가 완전히 일치하지 않기 때문).

5) 다음 OpenGL 코드에서 LIGHT0와 GL_LIGHT_MODEL_AMBIENT 광원은 무엇인지(색과 위치) 설명하고, 광원과 재질의 ambient, diffuse, specular는 어떻게 사용되는 지 간단히 설명하라. (10점)

```
void init(void)
{
    GLfloat ambient[] = {0.0, 0.0, 0.0, 1.0};
    GLfloat diffuse[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat specular[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat position[] = {0.0, 3.0, 3.0, 0.0};
    GLfloat lmodel_ambient[] = {0.2, 0.2, 0.2, 1.0};
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
}

void renderTeapot(GLfloat x, GLfloat y,
      GLfloat ambr, GLfloat ambg, GLfloat ambb,
      GLfloat difr, GLfloat difg, GLfloat difb,
      GLfloat specr, GLfloat specg, GLfloat specb, GLfloat shine)
```

```
{
    GLfloat mat[4];
    glPushMatrix();
    glTranslatef(x, y, 0.0);
    mat[0] = ambr; mat[1] = ambg; mat[2] = ambb; mat[3] = 1.0;
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat);
    mat[0] = difr; mat[1] = difg; mat[2] = difb;
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat);
    mat[0] = specr; mat[1] = specg; mat[2] = specb;
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat);
    glMaterialf(GL_FRONT, GL_SHININESS, shine * 128.0);
    glCallList(teapotList);
    glPopMatrix();
}
```

Light0는 난반사와 정반사광(diffuise & specular)으로 백색(1, 1, 1) (0, 3, 3, 0) 벡터로 된 방향성 광원 (Directional Light)을 보여주고 있다.

GL_LIGHT_MODEL_AMBIENT는 전체장면에 전역적인 주변광(ambient)을 주는 것으로 어두운 회색(0.2, 0.2, 0.2. 1.0)을 주었다.

광원의 색상을 ambient, diffuse, specular로 지정한 것과 물체의 재질의 색상을 ambient, diffuse, specular로 지정한 것으로, 조명모델공식에서 이 둘의 상호작용 값이 적용된다.

Ambient는 광원이 닿지 않는 곳에서 주변광을 넣어주기 위한 것
Diffuse는 물체의 고유한 색이 나타나도록 하는 것
Specular는 물체에 빛이 강하게 닿았을 때 광원의 색이 나타나게 하는 것