단국대학교 멀티미디어공학 그래픽스 프로그래밍 기말고사 (2009년 봄학기) 2009년 6월 16일

학과 _____ 학번 _____ 이름 _____

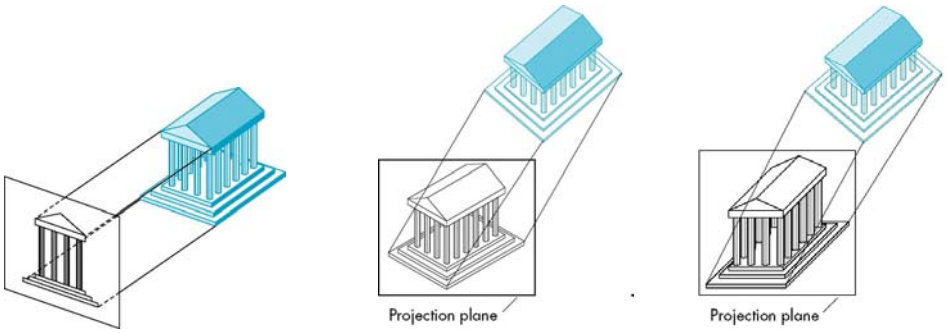
기말고사

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.**
- **성적공고시 중간고사때 제출한 암호를 사용할 것임.**

1. 다음 문제에 답하시오. (50점)

1) 직교투영 (orthographic projection), 축측투영 (axonometric projection), 경사투영 (oblique projection)을 간단히 설명하고, 각 투영 방식의 차이점을 나타내라.



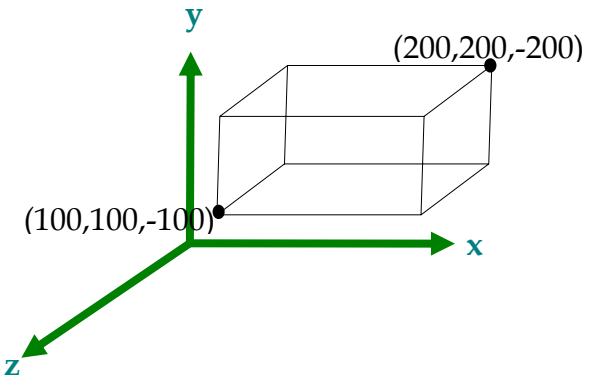
직교투영: 투영선(projector)은 투영면(projection plane)에 수직이다.
축측투영: 투영선은 투영면에 수직이지만, 투영면은 객체에 대해 어떠한 방향에도 존재할 수 있다.
경사투영: 투영선은 투영면과 임의의 각을 가질 수 있다.

2) 다음은 카메라 (Camera) 클래스의 일부를 보여주고 있다. apply() 함수는 뷰행렬 (View Matrix)을 생성하는 함수이다. 빈 칸을 채우시오.

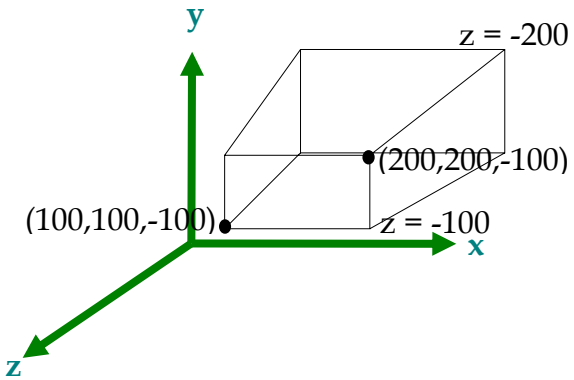
```
camera::camera(…)
{

..//중간생략

  position_= vector3(0.0, 0.0, 0.0);
  right_   = vector3(1.0, 0.0, 0.0);
  up_      = vector3(0.0, 1.0, 0.0);
  look_    = vector3(0.0, 0.0, 1.0);
}
```

```
void camera::apply()
{
  // 중간생략
  glLoadIdentity();

  // Keep camera's axes orthogonal to eachother
  look_.normalize();
  up_ = __vector3::crossProduct(up_, look_);_____________________________
  up_.normalize();
  right_ = __ vector3::crossProduct(look_, right_)_____________________________
  right_.normalize();

  // Build the view matrix:
  V[0] = right_.vec[0]; V[1] = up_.vec[0]; V[2] = look_.vec[0];  V[3] = 0.0f;
  V[4] = _ right_.vec[1];_ V[5] = _ up_.vec[1];_ V[6] = _ look_.vec[1];_ V[7] = 0.0f;
  V[8] = _ right_.vec[2];_  V[9] = _ up_.vec[2];_ V[10] = __ look_.vec[2];_ V[11] = 0.0f;
  V[12] = _ -vector3::dotProduct(right_, pos_);_ V[13] = __ -vector3::dotProduct(up_, pos_)_
V[14] = _ -vector3::dotProduct(look_, pos_);_ V[15] = 1.0f;

  // Multiply the view matrix:
  glMultMatrixf(V);
}
```

3) OpenGL에서 직교투영 (Orthogonal projection) 함수인 glOrtho(100, 200, 100, 200, 100, 200)의 관측공간(View volume)을 그림으로 표시하라.



4) OpenGL에서 원근투영 (Perspective projection) 함수인 glFrustum(100, 200, 100, 200, 100, 200)의 관측공간(View volume)을 그림으로 표시하라.



5) 다음은 OpenGL에서 GL_FLAT과 GL_SMOOTH의 Shading Model 사용하는 경우를 보여주는 간단한 프로그램이다. 출력 결과에 차이를 설명하라.

```
void drawTriangle()
{
    glBegin(GL_TRIANGLES);
    glColor3f(1, 0, 0); // red
    glVertex3f(-1, -1, 4);
    glColor3f(0, 1, 0); // green
    glVertex3f(1, -1, 4);
    glColor3f(0, 0, 1); // blue
    glVertex3f(0, 1, 4);
    glEnd();
}

void drawTriangle2()
{
    glBegin(GL_TRIANGLES);
    glColor3f(0, 0, 1); // blue
    glVertex3f(-1, -1, 4);
    glColor3f(0, 1, 0); // green
    glVertex3f(1, -1, 4);
    glColor3f(1, 0, 0); // red
    glVertex3f(0, 1, 4);
    glEnd();
}

void display()
{
    glClearColor(0.5, 0.7, 1, 0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 1, 0.1, 1000);
    glMatrixMode(GL_MODELVIEW);

    glLoadIdentity();
    glTranslatef(0.0, 0.0, -viewDistance);
    glRotatef(viewRotX, 1.0, 0.0, 0.0);
    glRotatef(viewRotY, 0.0, 1.0, 0.0);

    if (shading == GL_FLAT)
            glShadeModel(GL_FLAT);
    else if (shading == GL_SMOOTH)
            glShadeModel(GL_SMOOTH);

    glPushMatrix();
    glTranslatef(-1.0, 0.0, 0.0);
    drawTriangle();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1.0, 0.0, 0.0);
    drawTriangle2();
    glPopMatrix();
```

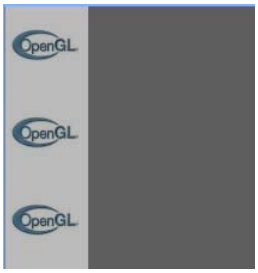
```
    glutSwapBuffers();
}
```

GL_FLAT: Flat shading (플랫세이딩)은 주어진 하나의 다각형 전체를 동일한 색으로 칠하는 방식으로 다각형의 첫 번째 정점의 색으로 다각형 전체에 칠해짐. 즉, 왼쪽 삼각형은 파란색으로, 오른쪽 삼각형은 빨간색으로 칠해짐

GL_SMOOTH: Gouraud shading (그로우세이딩)은 정점의 색을 보간하여 칠하는 방식으로 다각형의 각 정점의 색들 간에 보간된 색으로 칠해짐. 즉, 왼쪽 삼각형은 파란색(위)->초록색(오)->빨간색(왼)->파란색(위)로 변하고, 오른쪽 삼각형은 빨간색(위)->초록색(오)->파란색(왼)->빨간색(위)로 칠해짐

6) OpenGL 함수 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, mode)는 텍스쳐 좌표값이 (0,1) 범위를 넘어선 값에 대해, mode가 GL_CLAMP는 s,t가 1보다 크면 1을 s,t가 0보다 작으면 1으로 값을 강제 조정하고, GL_REPEAT는 s,t%1을 사용하여 텍스쳐를 반복한다. 화면 출력결과를 참고하여 빈 칸을 채우시오.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, ___GL_CLAMP___);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, ___GL_REPEAT___);
```




```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, ___GL_REPEAT___);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, ___GL_REPEAT___);
```

7) 아래의 코드를 참조하여, OpenGL 환경에서 텍스쳐 확대, 축소, 밉맵 필터링 (Filtering)에 대해서 설명하라.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
gluBuild2DMipmaps(GL_TEXTURE_2D, numComponents == 3 ? GL_RGB : GL_RGBA,
                  imageWidth, imageHeight, numComponents == 3 ? GL_RGB : GL_RGBA,
                  GL_UNSIGNED_BYTE, imagePtr);
```

확대를 위한 필터: 텍셀이 한 pixel보다 클 때 사용되는 필터
축소를 위한 필터: 텍셀이 한 pixel보다 작을 때 사용되는 필터
밉맵핑 필터: 일련의 축소된 크기의 밉맵 택스쳐를 생성하여, 작은 객체에 텍스쳐 맵핑을 할 시 보간 문제를 줄여주는 필터

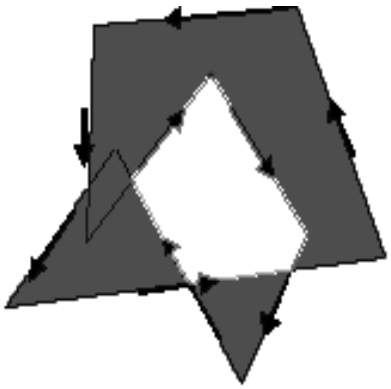
8) 멀티 텍스쳐링 (Multi-texturing) 기법에 대해서 간단히 설명하라. Single-Pass Multi-texturing과 Multi-Pass Multi-texturing 방법의 차이점을 설명하라.

멀티 텍스쳐링: 하나 이상의 텍스쳐를 객체에 적용해서 렌더링 효과를 높이는 것

싱글패스 멀티 텍스쳐링: 하나의 렌더링 패스 안에서 텍스쳐를 여러 개 입히는 것

다중패스 멀티 텍스쳐링: 장면이나 다각형 자체를 여러 번 블렌딩하여 렌더링하는 것으로, 여러 번 다각형을 그려야 하는 단점이 있음

9) 다음 다각형에 홀짝규칙 (even-odd rule)과 접기횟수규칙 (non-zero winding rule)을 사용하여 다각형 채우기 (Polygon filling)를 하라



10) Sutherland-Hodgeman 알고리즘에 대해 간단히 설명하라.

Sutherland-Hodgeman 알고리즘은 다각형을 클리핑하는 알고리즘으로써 클리핑 면에 대한 연속적인 절단 (pipeline clipping of polygons)을 수행하여 클리핑 윈도우 내부만 남기는 방식이다. 절단면으로 처리된 정점을 다음 클리핑 면에 반복 적용하는 방식으로 Concave polygon에는 부적합하다.

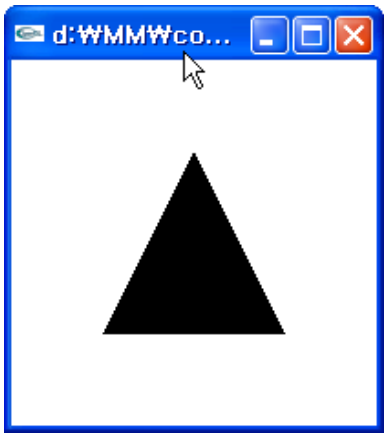
2. 다음에 주어진 삼각형을 그리는 간단한 OpenGL 프로그램을 보고 뷰잉 변환에 관한 문제를 답하라. (10점)

```
#include <GL/glut.h>

// display
void display()
{
  glClearColor(1, 1, 1, 1);
  glClear(GL_COLOR_BUFFER_BIT);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 2.0);
  glMatrixMode(GL_MODELVIEW);

  glColor3f(0.0, 0.0, 0.0);
  glBegin(GL_TRIANGLES);
  glVertex2f(-1.0, -1.0);
  glVertex2f(0.0, 1.0);
  glVertex2f(1.0, -1.0);
  glEnd();

  glFlush();
}

// keyboard
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
  case 0x1B:
  case 'q':
  case 'Q':
          exit(0);
          break;
  }
}
```

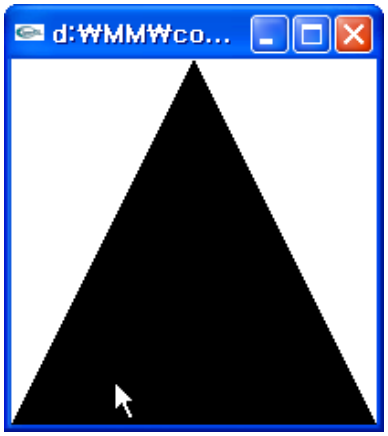
학과 _____ 학번 _____ 이름 _____

```
// reshape
void reshape(int w, int h)
{
  glViewport(0, 0, w, h);
}

int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(200, 200);
  glutInitWindowPosition(0,0);
  glutCreateWindow(argv[0]);
  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);
  glutReshapeFunc(reshape);
  glutMainLoop();
}
```

1) 다음 프로그램을 실행할 경우 200 x 200 크기의 윈도우에 어떤 그림이 그려질 지, 출력결과를 그려라 (정확한 눈금의 값을 명시할 것).



2) 만약 이 코드에서 줄 친 세 문장을 제거하고 프로그램을 실행시키면, 200 x 200 크기의 윈도우에 어떤 그림이 그려질 지, 출력결과를 그려라 (정확한 눈금의 값을 명시할 것).



학과 _____ 학번 _____ 이름 _____

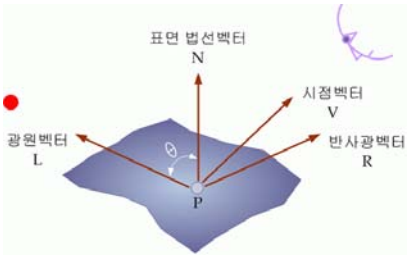
3. 다음은 퐁의 조명 모델 (Phong's Illumination Model) 수식을 기술한 것이다. 아래의 질문에 답하라. (15점)

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

1) 이 공식에서 주변광, 확산광, 경면광, 방출광, 광원과 물체간의 거리에 따른 빛의 감쇠를 의미하는 부분이 어디인지 표시하라.

- 주변광 (Ambient reflection) : __ 광원에 직접 노출되지 않는 면에 밝기를 부여, K_aI_a
- 확산광 (Diffuse reflection) : __ 면이 서있는 방향에 따라 차등적 밝기를 부여함, K_dI_d (N • L)
- 경면광 (Specular reflection) : ___ 반짝이는 표면에서 반사되는 빛을 모델링함, K_sI_s $(R \bullet V)^{\alpha}$___
- 방출광 (Emissive illumination) : ____ 자체가 빛을 발하는 방출조명, E ____
- 광원과 물체간의 거리에 따른 빛의 감쇠(Attenuation) : ____$1/(a + bd + cd^2)$_____

2) 위의 식에서 램버트 법칙 (Lambertian Law)를 설명하라.



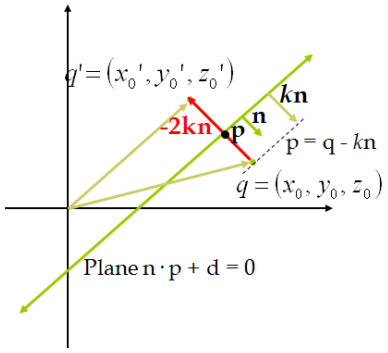
면의 밝기는 입사각(광원 벡터와 면의 법선벡터 사이각)의 코사인에 정비례함: diffuse reflection ∝ cos θ = N • L

3) 위의 식에서 경면광 부분을 바꾼 Blinn 조명 모델 (혹은 Modified Phong Illumination Model)을 설명하라.
경면광에서 중간각(H), 즉 면의 법선벡터(N)와 시점벡터(V)를 이용하여 계산,을 사용함.

$$\text{Specular Reflection} = K_s I_s (\cos\psi)^n = K_s I_s (N \bullet H)^{n'}$$
$$where\ 2\psi = \phi$$
$$When\ N \bullet L > 0, H = 1$$
$$When\ N \bullet L \le 0, H = 0$$

4. 다음은 거울 평면 (Plane: ax + by + cz + d = 0)에 대해 점 q=(x_0, y_0, z_0)의 반사된 지점 q′=(x_0', y_0', z_0')을 계산하는 과정을 보여주고 있다. 아래의 질문에 답하라. (15점)

학과 _____ 학번 _____ 이름 _____



1) 평면의 법선벡터 n(a, b, c)가 단위벡터 (unit vector)일 경우, q'을 계산하는 공식을 유도하라.

$$q' = q - 2k\mathrm{n}$$
$$= q - 2\frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}\mathrm{n}$$
$$= q - 2(n \cdot q + d)\mathrm{n} \text{ where } \mathrm{n(a,b,c)} \text{ is unit vector}$$

2) 평면 Plane (0, 0, 1, 2)에 반사된 점 q'의 값을 구하라.

평면(0, 0, 1, 2)에 반사된 점 $q' = (x_0, y_0, -z_0 - 4)$

3) 평면 Plane (a, b, c, d)에 반사하는 q' = R q, 반사행렬 R을 구하라.

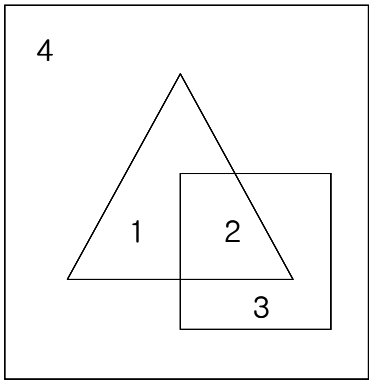
$$\mathbf{R} = \begin{pmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -2bc & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5. 다음 검정색 배경에 파란색 삼각형과 그 위에 빨간색 사각형을 블렌딩하여 그리는 간단한 OpenGL 프로그램의 일부를 보여주고 있다. 아래와 같이 여러 가지 방법으로 블렌딩 함수를 사용했을 때 그림에서 1 (삼각형 부분만), 2 (삼각형과 사각형이 겹치는 부분), 3 (사각형 부분만), 4 (나머지 배경만)의 화면에 출력되는 최종 RGBA 색을 계산하여 표의 빈칸에 넣으시오 (10점)

블렌딩 공식: C = SourceFactor*C_s + DestinationFactor*C_d

```
void drawObject()
```

```
{
    glColor4f(0, 0, 1, 1);
    glBegin(GL_TRIANGLES);
    glVertex3f(0.0, 1.0, -3.0);
    glVertex3f(-1.0, -1.0, -3.0);
    glVertex3f(1.0, -1.0, -3.0);
    glEnd();

    glColor4f(1, 0, 0, 0.5);
    glBegin(GL_QUADS);
    glVertex3f(0.0, -1.0, -2.0);
    glVertex3f(1.0, -1.0, -2.0);
    glVertex3f(1.0, 0.0, -2.0);
    glVertex3f(0.0, 0.0, -2.0);
    glEnd();
}

void draw()
{
  glClearColor(0, 0, 0, 1);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glEnable(GL_BLEND);
  if (filter == 0)
      glBlendFunc(GL_ONE, GL_ZERO);
  else if (filter == 1)
      glBlendFunc(GL_ZERO, GL_ONE);
  else if (filter == 2)
      glBlendFunc(GL_ONE, GL_ONE);
  else if (filter == 3)
      glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  drawObject();
  glDisable(GL_BLEND);
}
```



Blending Func	1	2	3	4
GL_ONE GL_ZERO	(0,0,1,1)	(1,0,0,1)	(1,0,0,1)	(0,0,0,1)
GL_ZERO GL_ONE	(0,0,0,1)	(0,0,0,1)	(0,0,0,1)	(0,0,0,1)
GL_ONE GL_ONE	(0,0,1,1)	(1,0,1,1)	(1,0,0,1)	(0,0,0,1)

GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA	(0,0,0.5,1)	(0.5,0,0.5,0.75)	(0.5,0,0,0.75)	(0,0,0,1)