단국대학교 멀티미디어공학 그래픽스 프로그래밍 기말고사 (2008년 봄학기) 2008년 6월 19일

학과 _____ 학번 _____ 이름 _____

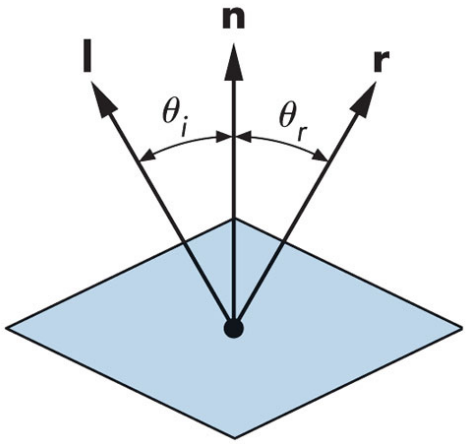
기말고사

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) 표면의 법선 벡터 (normal vector) N과 표면에서 광원으로 향하는 광원 벡터 (light vector) L이 주어졌을 때, 반사 벡터 (reflection vector) R을 유도하라.



2) OpenGL의 glMatrixMode(GLenum mode)는 현재 행렬의 모드를 설정하는 함수이다. mode 인자 3가지를 간단히 설명하시오.

3) 다음 관측함수 glOrtho(left, right, bottom, top, near, far), glFrustum(left, right, bottom, top, near, far), gluPerspective(fovy, aspect, near, far) 함수를 간단히 설명하고 관측공간을 그림으로 표시하라.

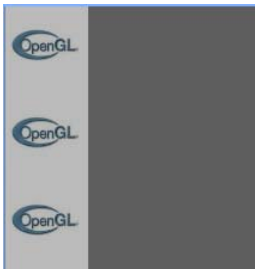
4) OpenGL에서 제공하는 광원 (light source)의 종류 4가지를 간단히 설명하시오.

5) 객체 공간 기법의 은면 제거 알고리즘을 2가지를 간단히 설명하라.

학과 _____ 학번 _____ 이름 _____

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 칸에 알맞은 모드를 채우시오.

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, _____);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, _____);



glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, _____);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, _____);

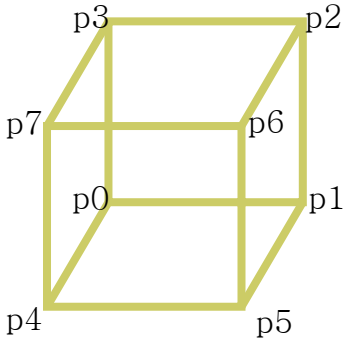
7) 다음은 실습예제로 제공한 텍스쳐 환경변수 OpenGL 프로그램의 일부를 보여주고 있다. OpenGL 함수 glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, mode)는 텍스쳐와 음영간의 상호작용을 지정할 수 있게 한다. GL_MODULATE, GL_DECAL, GL_BLEND, GL_REPLACE 모드를 사용했을 때 텍스쳐 값이 어떻게 나타나는지 빈 칸에 간단히 설명하시오. C_t: 텍스쳐 색, C_f: 프레임버퍼의 색, C_b: 블렌딩 색을 사용하여, 식으로 설명하시오.

```
void draw()
{
// .. 중간 생략
    initLight();

    glEnable(GL_TEXTURE_2D);
    glColor3f(1.0, 1.0, 1.0);
    glBindTexture(GL_TEXTURE_2D, texID);

    // 1. _____
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
    glPushMatrix();
    glTranslatef(-1.1, 1.0, 0.0);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
```

```
        gluSphere(quadric, 1.0, 16, 8);
        glPopMatrix();

        // 2. _____ _____
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
        glPushMatrix();
        glTranslatef(1.1, 1.0, 0.0);
        glRotatef(-90.0, 1.0, 0.0, 0.0);
        gluSphere(quadric, 1.0, 16, 8);
        glPopMatrix();

        // 3. _____
        GLfloat blendcolor[] = {0.0, 1.0, 0.0, 0.5};
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_BLEND);
        glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, blendcolor);
        glPushMatrix();
        glTranslatef(-1.1, -1.0, 0.0);
        glRotatef(-90.0, 1.0, 0.0, 0.0);
        gluSphere(quadric, 1.0, 16, 8);
        glPopMatrix();

        // 4. _____
        glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
        glPushMatrix();
        glTranslatef(1.1, -1.0, 0.0);
        glRotatef(-90.0, 1.0, 0.0, 0.0);
        gluSphere(quadric, 1.0, 16, 8);
        glPopMatrix();

        glBindTexture(GL_TEXTURE_2D, 0);
        glDisable(GL_TEXTURE_2D);
// .. 중간 생략
}
```

8) 다음은 입방체 (cube)를 그리는 OpenGL 프로그램의 일부이다. 빈 칸을 채우시오.

```
void drawCube()
{
        float vertex[8][3];
        vertex[0][0] = -1; vertex[0][1] = -1; vertex[0][2] = -1;
        vertex[1][0] = 1; vertex[1][1] = -1; vertex[1][2] = -1;
        vertex[2][0] = 1; vertex[2][1] = 1; vertex[2][2] = -1;
        vertex[3][0] = -1; vertex[3][1] = 1; vertex[3][2] = -1;
        vertex[4][0] = -1; vertex[4][1] = -1; vertex[4][2] = 1;
        vertex[5][0] = 1; vertex[5][1] = -1; vertex[5][2] = 1;
        vertex[6][0] = 1; vertex[6][1] = 1; vertex[6][2] = 1;
        vertex[7][0] = -1; vertex[7][1] = 1; vertex[7][2] = 1;

        float normal[6][3];
        normal[0][0] = 1.0; normal[0][1] = 0.0; normal[0][2] = 0.0;
        normal[1][0] = 0.0; normal[1][1] = 1.0; normal[1][2] = 0.0;
        normal[2][0] = 0.0; normal[2][1] = 0.0; normal[2][2] = 1.0;
        normal[3][0] = -1.0; normal[3][1] = 0.0; normal[3][2] = 0.0;
        normal[4][0] = 0.0; normal[4][1] = -1.0; normal[4][2] = 0.0;
```



```
        normal[5][0] = 0.0; normal[5][1] = 0.0; normal[5][2] = -1.0;

        glBegin( GL_QUADS );
        glNormal3fv( normal[0] ); // right (오른쪽)
        glVertex3fv( vertex[5] );
        glVertex3fv( vertex[1] );
        glVertex3fv( vertex[2] );
        glVertex3fv( vertex[6] );

        glNormal3fv( _____ ); // top (윗쪽)
        glVertex3fv( vertex[6] );
        glVertex3fv( _____ );
        glVertex3fv( _____ );
        glVertex3fv( _____ );

..... // 중간 생략

        glNormal3fv( _____ ); //back (뒤쪽)
        glVertex3fv( vertex[0] );
        glVertex3fv( _____ );
        glVertex3fv( _____ );
        glVertex3fv( _____ );
        glEnd();
    }
```

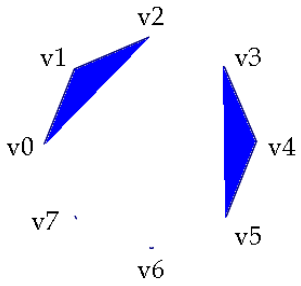
9) 다음은 구체(sphere)를 그리는 OpenGL 프로그램의 일부이다. 빈 칸을 채우시오.

```
void drawSphere(float radius, int stacks, int slices)
{
    float lon, lat, v[3], n[3];
    float lonstep = M_PI/stacks;
    float latstep = M_PI/slices;

    for (lon = 0.0; lon <= 2*M_PI; lon += (lonstep)) {
        glBegin(GL_TRIANGLE_STRIP);
        for (lat = 0.0; lat <= M_PI + latstep; lat += (latstep)) {
            n[0] = _____
            n[1] = _____
            n[2] = _____
            v[0] = radius * cosf(lon)*sinf(lat);
            v[1] = radius * sinf(lon)*sinf(lat);
            v[2] = radius * cosf(lat);
            glNormal3fv(n);
            glVertex3fv(v);

            n[0] = _____
            n[1] = _____
            n[2] = _____
            v[0] = radius * cosf(lon + lonstep)*sinf(lat);
            v[1] = radius * sinf(lon + lonstep)*sinf(lat);
            v[2] = radius * cosf(lat);
            glNormal3fv(n);
            glVertex3fv(v);
        }
        glEnd();
    }
```

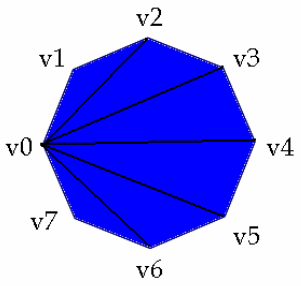
}

10) 다음은 간단한 OpenGL 프로그램의 일부이다. 오른쪽 화면 출력결과를 참고하여 빈 칸을 채우시오.

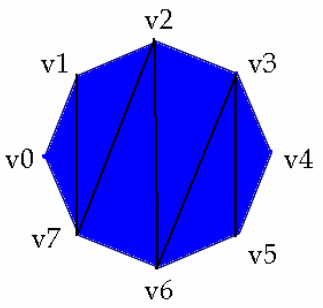
```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(_____);
for (int i = 0; i<8; i++)
    glVertex2fv(v[i]);
glEnd();
```




```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(_____);
for (int i = 0; i<8; i++)
    glVertex2fv(v[i]);
glEnd();
```




```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glBegin(_____);
glVertex2fv(v[0]);
glVertex2fv(v[1]);
glVertex2fv(v[7]);
glVertex2fv(v[2]);
glVertex2fv(v[6]);
glVertex2fv(v[3]);
glVertex2fv(v[5]);
glVertex2fv(v[4]);
glEnd();
```



2. 래스터화 과정에서 선분을 그리는 Bresenham's Line Drawing 알고리즘을 간단히 설명하라. 그리고 DDA (Digital Differential Analyzer) 알고리즘과의 차이점을 비교하라. (10점)

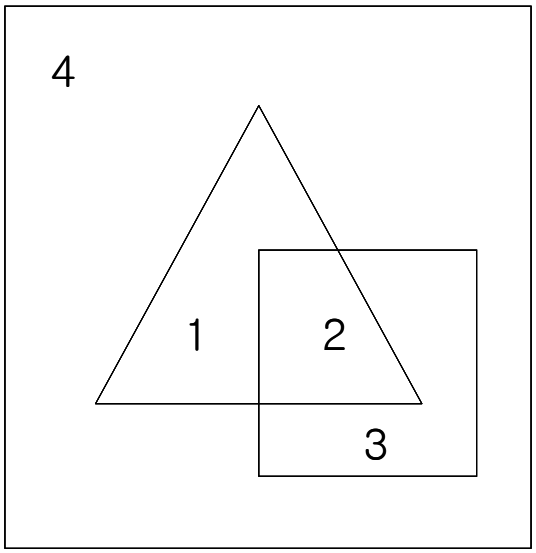
학과 _____ 학번 _____ 이름 _____

3. 다음은 초록색 배경에 빨간색 삼각형과 그 위에 파란색 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. (20점)

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

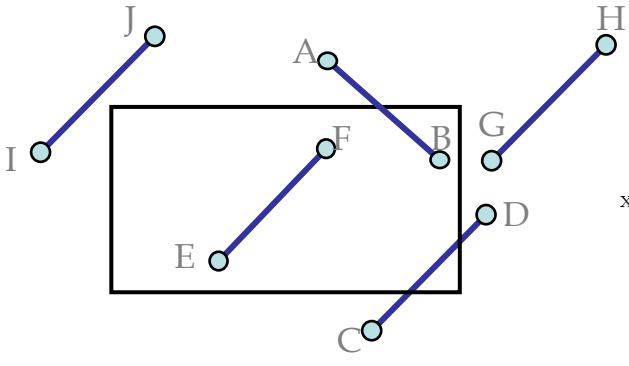
```
void drawObject()
{
    glColor4f(1, 0, 0, 1);
    glBegin(GL_TRIANGLES);
    glVertex3f(0.0, 1.0, -3.0);
    glVertex3f(-1.0, -1.0, -3.0);
    glVertex3f(1.0, -1.0, -3.0);
    glEnd();

    glColor4f(0, 0, 1, 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, 1, 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);
  else if (filter == 4)
       glBlendFunc(GL_SRC_ALPHA, GL_ONE);
  else if (filter == 5)
       glBlendFunc(GL_ZERO, GL_SRC_COLOR);
  else if (filter == 6)
       glBlendFunc(GL_ONE_MINUS_DST_COLOR, GL_ZERO);
  drawObject();
  glDisable(GL_BLEND);
}
```

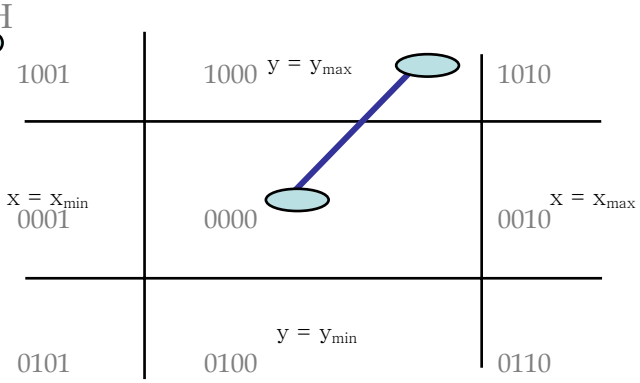
학과 _____ 학번 _____ 이름 _____



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

4. 다음은 Cohen-Sutherland 알고리즘으로 2차원 선분을 클리핑 하려고 한다. 아래 오른쪽 그림은 Cohen-Sutherland 알고리즘의 각 영역에 대한 4-비트 외곽부호 (outcode)를 보여주고 있다. 아래 빈 칸을 채우시오. (10점).





학과 _____ 학번 _____ 이름 _____

선분의 각 끝점의 4 비트 외곽부호 (outcode)를 구하시오.

- A: _____ B: _____
- C: _____ D: _____
- E: _____ F: _____
- G: _____ H: _____
- I: _____ J: _____

선분 클리핑을 위해 4 비트 외곽부호를 이용하여 다음 4 가지 경우로 분류하는 판정 기준을 적으시오.

- 선분의 양 끝점이 클리핑 윈도우 내부에 있는 경우 (accept):

_____

- 선분의 양 끝점이 클리핑 윈도우의 같은 변의 외부에 있는 경우 (reject):

_____

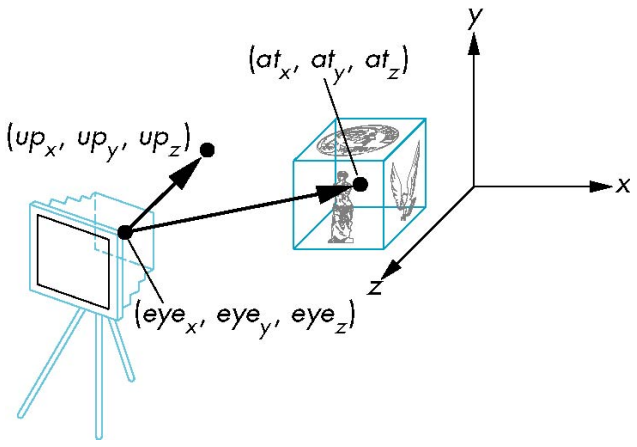
- 선분의 한 끝점은 클리핑 윈도우 내부에 있고, 다른 하나는 외부에 있는 경우 (1 개 교차점을 찾는다 subdivision):

_____

- 선분의 양 끝점이 모두 외부에 있고 선분의 일부가 클리핑 윈도우 내부에 있는 경우 (subdivision):

_____

5. 아래 그림을 참고하여 gluLookAt(1,1,0, 0,0,0, 0,1,0) 함수에서 뷰잉 행렬 (Viewing Matrix) M 을 도출하는 계산과정을 보여라. 아래 빈 칸을 채우시오. (10점)



학과 _____ 학번 _____ 이름 _____

- Eye position (eye): _____
- Look-at position (at): _____
- Up-vector (up): _____
- n (camera frame Z) : _____
- u (camera frame X): _____
- v (camera frame Y): _____
- M (viewing matrix):

$$\mathbf{M} = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

- M (viewing matrix):

$$\mathbf{M} = \begin{pmatrix} 0 & -0.70717 & 0.707107 & 0 \\ 0 & 0.707107 & 0.707107 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1.414214 & 1 \end{pmatrix}$$