단국대학교 멀티미디어공학 그래픽스 프로그래밍 기말고사 (2007년 봄학기) 2007년 6월 15일

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. 맞으면 true, 틀리면 false를 적으시오. (20점)

1) 은면제거 알고리즘 중 페인터 알고리즘 (painter's algorithm)은 이미지공간 (image-space) 기법이기 보다는 객체공간 (object-space) 기법이다. _____T___
2) DDA (Digital Differential Analyzer) 알고리즘은 직선을 픽셀로 변환하는 알고리즘이며 부동소수곱셈 연산을 필요로 한다. ____F___
3) 임의의 복잡한 다각형 (polygon)을 단순한 다각형 (예를 들어, 삼각형)으로 나누는 작업을 분할 (tessellation)이라고 한다. ____T____
4) 중간각벡터 (half-way vector)는 정반사 (specular reflection)에 의한 음영 계산을 빨리 해주기 위해서 사용된다. ____T___
5) 난반사 (diffuse reflection)에서 입사각 (광원벡터와 표면의 법선벡터 사이각)이 직각을 이룰 때 표면에 빛이 가장 밝게 나타난다. ____F___
6) 알파 블렌딩 (alpha blending) 사용시 불투명과 투명을 같이 그릴 때 투명을 먼저 그리고 불투명을 나중에 그린다. ____F___
7) Sutherland-Hodgeman 알고리즘은 4비트와 6비트 외곽부호 (outcode)를 사용하여 2차원, 3차원 클리핑 (clipping)하는 알고리즘이다. ___F___
8) 투영 정규화 (projection normalization)은 객체들을 사전 왜곡시켜 모든 투영을 직교투영으로 변환시킨다. ___T___
9) 라디오 시티 (radiosity)와 광선추적 (ray tracing)은 직접 조명 모델 (direct illumination model)이다. ____F___
10) 기하학적 도형을 프레임 버퍼 안의 픽셀의 색, 위치로 변환시키는 것을 래스터화 (rasterization)라고 한다. ___T____

2. 다음 문제에 답하시오. (50점)

1) 물체를 표현하는데 있어 가장 기본이 되는 요소인 기하요소 (geometry primitives)를 3가지 이상 적으시오.

점 (point)
선 (line), 선분 (line segment), 광선 (ray)
원 (circle), 구 (sphere), 원기둥 (cylinder), 원뿔 (cone)
삼각형 (triangle), 평면 (plane), 입방체 (cube), 직육면체 (rectilinear box)

학과 _____ 학번 _____ 이름 _____

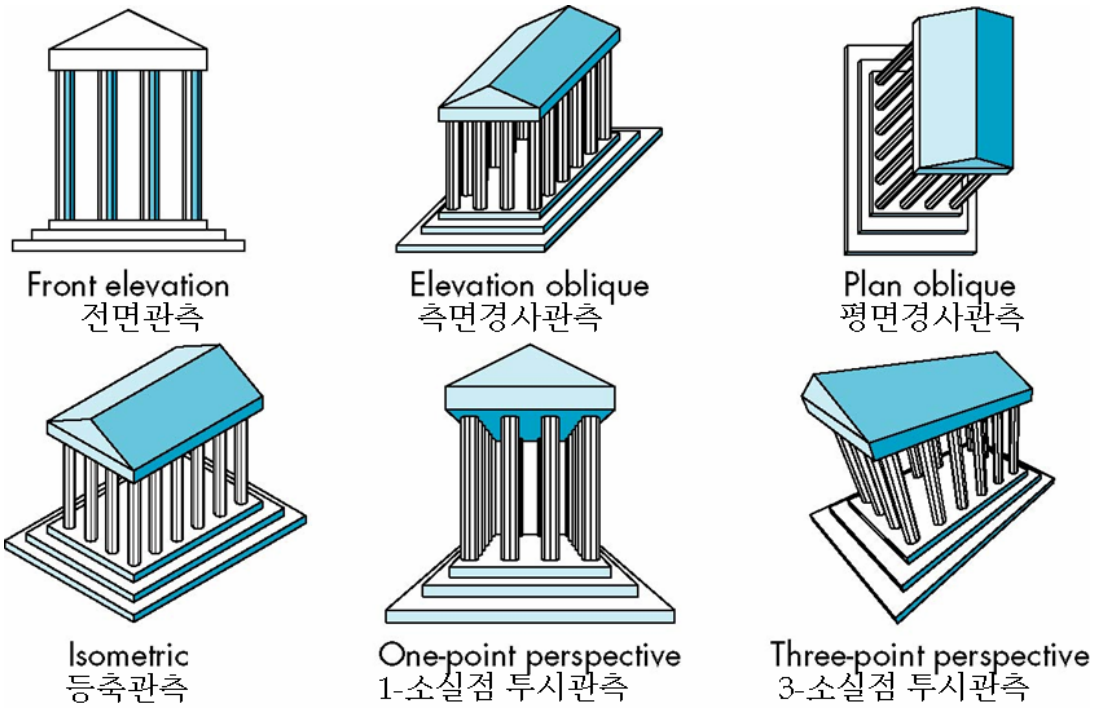
2) 3개의 기본적인 기하학적 객체의 변환 (geometric transformation)을 서술하고, OpenGL 변환 함수를 적으시오.

이동 (translation) – 주어진 방향으로 일정한 거리만큼 점을 이동. glTranslatef(x, y, z)

회전 (rotation) – 회전축에 대해 각도 (angle)만큼 회전. glRotatef(angle, x, y, z)

크기변환 (scaling) – 크기변환값 (scaling factor)를 사용하여 크거나 작게 크기변환. glScalf(x, y, z)

3) 다음 그림은 고전적 관측들을 보여주고 있다. 평행 관측 (parallel viewing)에 속하는 것에 모두 동그라미 표시 하시오.



4) 깊이버퍼 알고리즘 (Depth-buffer algorithm)의 작동원리에 대하여 설명하시오.

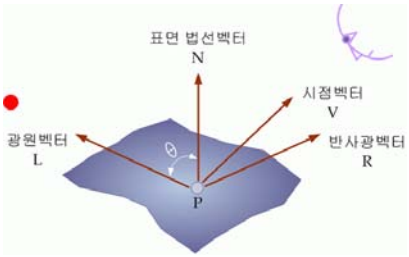
가장 일반적으로 사용되는 이미지 공간 은면제거 기법으로, 물체의 가시성 (visibility)를 화소 (pixel) 단위로 조사하여 z (깊이)값이 가장 작은 평면의 값을 그린다.

깊이 버퍼 (depth-buffer)는 그리고자 하는 화소 당 깊이 정보 (depth value, 즉 z값)을 저장한다.

새로운 화소를 그릴 때 마다, 새로운 깊이 정보를 깊이 버퍼 (depth buffer)안에 있는 깊이 값 (depth value)과 비교한다.

학과 _____ 학번 _____ 이름 _____

5) 다음은 직접조명모델 (Direct illumination model)을 보여주고 있다. 이 공식에서 환경광 (ambient light), 확산광 (diffuse light), 경면광 (specular light), 방출광 (emissive light)이 어느 것인지 표시하라. 그리고 이 공식에서 α 계수가 무엇을 의미하는지도 설명하라.



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

Ambient: 광원에 직접 노출되지 않는 면에 밝기를 부여, K_aI_a
Diffuse: 면이 서있는 방향에 따라 차등적 밝기를 부여함, K_dI_d (N ● L)
Specular: 반짝이는 표면에서 반사되는 빛을 모델링함, K_sI_s $(R \bullet V)^{\alpha}$
Emissive: 자체가 빛을 발하는 방출조명, E
α 계수: 광택계수 (shininess coefficient) 하이라이트 크기를 결정

6) 플랫 세이딩 (Flat shading), 구로우 세이딩 (Gouraud shading), 퐁 세이딩 (Phong shading)을 각각 설명하고 장단점을 비교하여라.

플랫 세이딩 (Flat shading)은 주어진 하나의 다각형 전체를 동일한 색으로 칠함. 빠르고 간단함. 그러나 곡면의 경우 경계선 부근에서 명암이 대조를 이루어 각이 진 모습을 보임.
구로우 세이딩 (Gouraud shading)은 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).
퐁 세이딩 (Phong shading)은 정점의 색 대신 정점의 법선벡터를 보간함. 이때 곡면의 기울기가 복원됨. 경면광을 부여할 수 있음.

7) 현재 프레임버퍼에 있는 픽셀 값 RGB (0.2, 0.2, 0.2) 위에 그리고자 하는 물체의 픽셀 값 RGBA (0.7, 0.5, 0.0, 0.5)을 가장 보편적 블렌딩 함수인 glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)를 사용할 때 아래의 블렌딩 공식을 사용한다. 알파 블렌딩 (alpha blending)되어 화면에 출력되는 RGB 색을 계산하시오.

R = SourceFactor*Rs + DestinationFactor*Rd = As * Rs + (1 – As) * Rd = 0.5*0.7 + (1 – 0.5)*0.2
G = SourceFactor*Gs + DestinationFactor*Gd = As * Gs + (1 – As) * Gd = 0.5*0.5 + (1 – 0.5)*0.2
B = SourceFactor*Bs + DestinationFactor*Bd = As * Bs + (1 – As) * Bd = 0.5*0.0+(1-0.5)*0.2
(0.45, 0.35, 0.1)

8) OpenGL 함수 glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, mode)는 텍스쳐와 음영간의 상호작용을 지정할 수 있게 한다. GL_TEXTURE_ENV_MODE는 GL_MODULATE, GL_DECAL, GL_BLEND, GL_REPLACE 를 지원하는데, 각 모드를 사용했을 때 텍스쳐 값이 어떻게 나타나는지 설명하시오. (아래의 모드에 따른 텍스쳐 함수 테이블을 참고하시오)

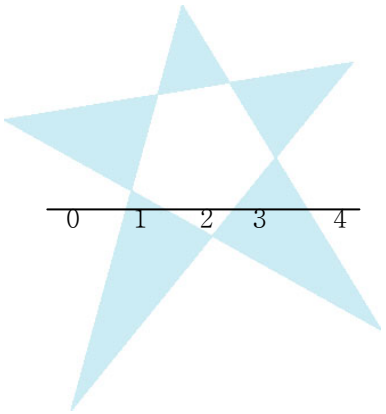
Base internal	Texture functions			
	GL_MODULATE	GL_DECAL	GL_BLEND	GL_REPLACE
GL_LUMINANCE	$C_v=L_tC_f$	undefined	$Cv=(1 - L_t)C_f + L_tC_c$	$C_v=L_t$
GL_RGB	$C_v=C_tC_f$	$C_v=C_t$	$C_v=(1 - C_t)C_f + C_tC_c$	$C_v=C_t$

GL_MODULATE: 텍스쳐의 색 성분과 음영에서 주어지는 색성분을 곱함으로써 텍스쳐 맵핑없이 할당될 음영을 변조 가능.
GL_DECAL: 텍스쳐의 색이 객체의 색을 완전히 결정

GL_BLEND: 환경색과 합성함

GL_REPLACE: 텍스쳐 색만 사용함

9) 다각형 내부의 판단 규칙 중 홀짝 규칙 (even-odd rule)은 주사선 별로 경계가 홀수 번째 교차하면 내부, 짝수 번째 교차하면 외부가 시작된다고 판단한다. 아래 그림의 다각형에 주사선 위에 홀짝 숫자를 적어 넣어 다각형의 내부와 외부를 표시하시오.



10) 반지름이 1인 구체(sphere)의 parametric form은

$x = \cos\phi \sin\theta, -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}, -\pi \le \theta \le \pi$

$y = \cos\phi \cos\theta, -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}, -\pi \le \theta \le \pi$ 로 표현된다.

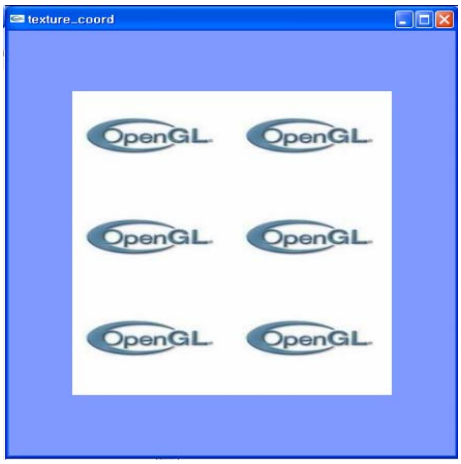
$z = \sin\phi, -\frac{\pi}{2} \le \phi \le \frac{\pi}{2}$

이 구체 표면의 법선벡터 (normal vector)를 구하라.

Normal vector: N (cosϕ sinθ, cosϕ cosθ, sinϕ) = P와 같다.

3. 다음은 아래의 opengl.jpg 이미지를 사용하여 GL_REPEAT을 사용한 텍스쳐 맵핑 프로그램을 보여주고 있다. 빈칸을 채워라. (10점)


```
glBegin(GL_QUADS);
    glTexCoord2f(__0___, ___0__);
    glVertex3f(-1.0, -1.0, 0.0);
    glTexCoord2f(__2___, ___0__);
    glVertex3f(1.0, -1.0, 0.0);
    glTexCoord2f(__2___, ___3__);
    glVertex3f(1.0, 1.0, 0.0);
    glTexCoord2f(__0___, ___3__);
    glVertex3f(-1.0, 1.0, 0.0);
glEnd();
```



학과 _____ 학번 _____ 이름 _____

4. 아래의 drawObjects() 함수를 설명하고, display() 함수에서 glOrtho 직교 투영 (orthographic projection)을 사용할 때와 gluPerspective 원근 투영 (perspective projection) 사용할 때의 차이를 설명하시오.

```
void drawObjects()
{
	glPushMatrix();
	glTranslatef(-2.0, 0.0, 0.0);
	glBegin(GL_TRIANGLES);
		glColor3f(1.0, 0.0, 0.0);
		glVertex3f(-0.4, 0.0, 0.0);
		glVertex3f(0.4, 0.0, 0.0);
		glVertex3f(0.0, 0.8, 0.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glTranslatef(-1.0, 0.0, -1.0);
	glBegin(GL_TRIANGLES);
		glColor3f(0.0, 1.0, 0.0);
		glVertex3f(-0.4, 0.0, 0.0);
		glVertex3f(0.4, 0.0, 0.0);
		glVertex3f(0.0, 0.8, 0.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glTranslatef(0.0, 0.0, -2.0);
	glBegin(GL_TRIANGLES);
		glColor3f(0.0, 0.0, 1.0);
		glVertex3f(-0.4, 0.0, 0.0);
		glVertex3f(0.4, 0.0, 0.0);
		glVertex3f(0.0, 0.8, 0.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glTranslatef(1.0, 0.0, -3.0);
	glBegin(GL_TRIANGLES);
		glColor3f(1.0, 1.0, 0.0);
		glVertex3f(-0.4, 0.0, 0.0);
		glVertex3f(0.4, 0.0, 0.0);
		glVertex3f(0.0, 0.8, 0.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glTranslatef(2.0, 0.0, -4.0);
	glBegin(GL_TRIANGLES);
		glColor3f(1.0, 0.0, 1.0);
		glVertex3f(-0.4, 0.0, 0.0);
		glVertex3f(0.4, 0.0, 0.0);
		glVertex3f(0.0, 0.8, 0.0);
	glEnd();
	glPopMatrix();
}

void display()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glLoadIdentity();

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	if (ortho)
		glOrtho(-10, 10, -10, 10, -10, 10);
	else
		gluPerspective(60.0, 1.0, 0.1, 100.0);
	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);

        drawObjects();

        glutSwapBuffers();
}
```

drawObject()은 왼쪽으로부터 빨간색, 녹색, 파란색, 노란색, 보라색 삼각형이 z-축으로 -1만큼씩 들어간 모습을 그린다.

glOrtho 직교투영은 직육면체 (rectilinear box)의 관측공간을 화면에 투영한다. 객체의 크기가 거리에 따라 변하지 않는다.

gluPerspective 원근투영은 절두체 (view frustum) 관측공간을 화면에 투영한다. 가까운 객체는 크게 나타나고, 멀리있는 객체는 작게 나타난다.

5. Bresenham 알고리즘을 이용해서 점 (3,7) 부터 점 (12, 15)까지 선을 그리고자 한다. 아래의 조건을 이용해서 (혹은 조건을 수정해서) 각 단계별로 P_k를 구하고 점 (vertex)이 그려져야 하는 x, y좌표의 값을 구하라. 표에 값이 전부 채워지지 않을 수도 있으며 모자라는 경우에는 행을 추가해서 숫자를 넣도록 한다 (10점) dx = 12 – 3 = 9; dy = 15 – 7 = 8

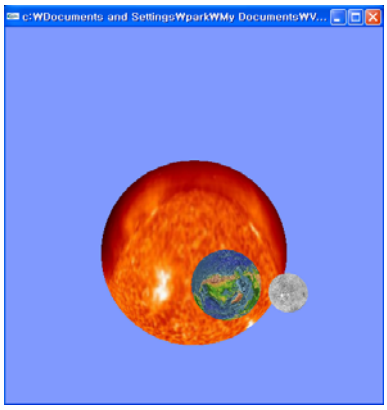
- Initialization
  - $D_0 = 2\Delta y - \Delta x$ $D_0 = 2*8 - 9 = 7$
- Stepwise process
  - If $D_k < 0$, $D_{k+1} = D_k + 2\Delta y$
  - Otherwise, $D_{k+1} = D_k + 2\Delta y - 2\Delta x$, y++ $D_1 = 7 + 2*8 - 2*9 = 5$

D_k	x	y
D0 = 7	3	7
D1 = 5	4	8
D2 = 3	5	9
D3 = 1	6	10
D4 = -1	7	11
D5 = 15	8	11
D6 = 13	9	12
D7 = 11	10	13
D8 = 9	11	14
D9 = 7	12	15

6. 다음은 수업시간에 배운 Simple Solar System 예제를 보여주고 있다. display()함수의 빈칸

학과 _____ 학번 _____ 이름 _____

으로 남아 있는 회전 (rotate)과 이동 (translate) 변환을 완성하라. 태양 (sun)은 자전 (spin)하고 있고, 지구 (earth)는 자전 (spin)하고 있으면서 태양주위를 공전 (revolve)하고 있다. 또한, 달 (moon)은 자전 (spin)하고 있으면서 지구 주위를 공전 (revolve)하고 있는 모습이다. (보너스 문제 extra 10점)



```
float g_SunRadius = 5.0f;
float g_EarthRadius = 1.0f;
float g_MoonRadius = 0.5f;

float g_EarthDistanceFromSun = -12.0f;
float g_MoonDistanceFromEarth = -2.0f;

void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();

    gluLookAt( 0.0, 2.0, 25.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 );

    static float SunSpin    = 0.0f;
    static float EarthSpin  = 0.0f;
    static float EarthOrbit = 0.0f;
    static float MoonSpin   = 0.0f;
    static float MoonOrbit  = 0.0f;

    if( g_OrbitOn == true )
    {
        SunSpin -= g_Speed * (g_ElpasedTime * 10.0f);   // sun rotate

        EarthSpin  -= g_Speed * (g_ElpasedTime * 100.0f);      // earth rotate
        EarthOrbit -= g_Speed * (g_ElpasedTime * 20.0f);       // earth revolve

        MoonSpin   -= g_Speed * (g_ElpasedTime * 50.0f);       // moon rotate
        MoonOrbit  -= g_Speed * (g_ElpasedTime * 200.0f);      // moon revolve
    }

    // The Sun (spins by rotating it about y-axis)
    glPushMatrix();
        glRotatef(SunSpin, 0.0f, 1.0f, 0.0f);

    glutSolidSphere( g_SunRadius, 24, 24 );
    glPopMatrix();
```

```
    // The Earth spins on its own axis and orbit the Sun.
    glPushMatrix();
        glRotatef(EarthOrbit, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.0f, 0.0f, g_EarthDistanceFromSun);
        glRotatef(EarthSpin, 0.0f, 1.0f, 0.0f);

    glutSolidSphere( g_EarthRadius, 24, 24 );
    glPopMatrix();

    // The Moon spins on its own axis and orbit the Earth.
    glPushMatrix();
        glRotatef(EarthOrbit, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.0f, 0.0f, g_EarthDistanceFromSun);
        glRotatef(MoonOrbit, 0.0f, 1.0f, 0.0f);
        glTranslatef(0.0f, 0.0f, g_MoonDistanceFromEarth);
        glRotatef(MoonSpin, 0.0f, 1.0f, 0.0f);

    glutSolidSphere( g_MoonRadius, 24, 24 );
    glPopMatrix();

    glutSwapBuffers();
}
```