학과 _____ 학번 _____ 이름 _____

중간고사

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. 다음 문제에 답하시오. (총 25점)

1) **깊이 버퍼(Depth buffer)가 무엇인지? OpenGL/GLUT에서 깊이 버퍼를 사용하기 위해 필요로 하는 설정을 서술하라. (5점)**

일명 z-buffer라고도 불리며, 색 버퍼(color buffer)와 동일한 크기를 가지며 픽셀단위로 기하개체의 깊이값(즉, z-값)이 가장 작은 것이 앞에 그리도록 깊이 정보를 축적하는 버퍼이다.
OpenGL/GLUT에서 깊이 버퍼를 사용하기 위해서 깊이버퍼를 사용한다는 초기화가 필요하며, 깊이버퍼 테스트를 활성화해야 하고, display callback 함수 내에서 매프레임마다 맨 처음으로 색 버퍼를 지울 때 깊이 버퍼도 같이 지워야 한다.

2) **더블 버퍼링 (Double buffering)이 무엇인지? OpenGL/GLUT에서 더블 버퍼링을 사용하기 위해서 필요로 하는 설정을 서술하라. (5점)**

더블 버퍼링은 2 개의 색 버퍼를 사용하는 것으로, 전면 버퍼와 후면 버퍼로 나누어서 전면 버퍼는 화면에 이미지를 그리기를 수행하고 후면 버퍼는 그 동안 프레임 버퍼에 새로운 이미지를 축적한다. 둘 다 수행을 완료한 후 서로 스왑버퍼링을 한다. 싱글버퍼링보다 부드러운 애니메이션을 그릴 수 있다.
OpenGL/GLUT에서 더블 버퍼링을 사용하기 위해서 싱글 버퍼링 대신 더블 버퍼링을 사용한다는 초기화가 필요하며, display callback 함수 내에서 매프레임마다 마지막으로 스왑 버퍼링(Swap Buffer)을 해야 한다.

3) **OpenGL에서 기본 카메라(default camera)에 대하여 간단히 서술하라. (5점)**

OpenGL에서 기본 카메라는 World에서 원점(0, 0, 0)에 위치하며, -z의 방향으로 향하고 있다. 관측 공간은 2x2x2 입방체의 viewing volume을 사용한다.

4) **glutDisplayFunc()와 glutRedisplay() 함수가 무엇인지 간단히 서술하라. (5점)**

glutDisplay()는 display callback 함수를 등록하는 함수이다. 디스플레이 이벤트 시 호출되는 함수이며, 그리기를 수행한다. 모든 GLUT 프로그래밍에서 반드시 사용해야 하는 함수이다.

glutRedisplay()는 윈도우가 새로 그려져야 할 필요가 있는 경우에 호출해서 사용한다. Display callback 함수를 직접 호출하는 것보다 glutRedisplay() 함수를 사용하여 display callback이 다시 그리기를 지시하는 방식이다. 예를 들어 키보드나 마우스 이벤트를 사용하여 도형의 색상을 바꾸거나 움직임을 주거나 한 후 glutRedisplay() 함수를 사용하여 바뀌어진 도형의 색상이나 움직임으로 다시 그리기를 수행하게 한다.

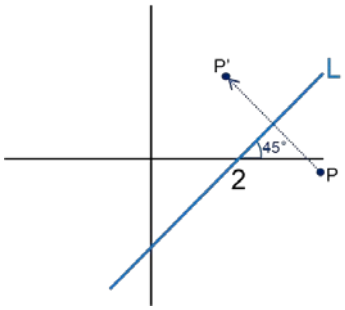
5) **Modern OpenGL 렌더링을 하기 위해 사용하는 Vertex Array Object (VAO), Vertex Buffer Object (VBO), Index Buffer Object (IBO)이 무엇인지 간단히 설명하라. (5점)**

VAO (Vertex Array Object) – 모든 정점자료(즉, position, color, ...)를 하나로 묶어줌. 보통 하나의 Mesh (즉, 기하객체)마다 하나의 VAO를 사용함. 즉, 하나의 VAO안에는 하나 또는 여러 개의 VBO가 존재함.

VBO (Vertex Buffer Object) – 정점자료(즉, position, color, normal, 등등)의 데이터를 저장하는 메모리 버퍼. 대용량 자료를 GPU에 보내줄 수 있음.

IBO (Index Buffer Object) – 인덱스자료(즉, 정점의 index) 데이터를 저장하는 버퍼

2. 다음은 2차원 공간의 점 P(x, y)를 직선 L에 대하여 반사(reflection)된 점 P'(x', y')에 대한 2차원 아핀 변환 행렬 (3x3 matrix)에 관한 문제이다. 아래의 물음에 답하라. (각 5점 총 25점)



1) **2차원 공간에 임의의 두 점을 선택하여 직선 L에 대한 반사된 점들을 계산하라.**

y = x – 2에 반사된 점은
P1 (3, 0) => (2, 1)
P2 (4, 0) => (2, 2)

2) **2차원 공간에 임의의 점 P을 직선 L에 대하여 반사해서 점 P'으로 변환시키는 아핀 변환 행렬 M (3x3)을 유도하라.**

x' = y + 2
y' = x – 2

$$M = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

3) **위(2.2)의 아핀 변환 행렬 M을 GLM translate(이동), rotate(회전), scale(크기) 등의 기본 변환 행렬들의 곱으로 적절히 표현하라. 행렬의 곱의 순서에 주의할 것. 아래는 GLM translate, rotate, scale 변환 함수의 사용법을 보여준다.**

```
glm::mat4 M1 = glm::translate(glm::mat4(1), glm::vec3(dx, dy, dz));
glm::mat4 M2 = glm::rotate(glm::mat4(1), glm::radian(angle), glm::vec3(ax, ay, az));
glm::mat4 M3 = glm::scale(glm::mat4(1), glm::vec3(sx, sy, sz));
```

```
glm::mat4 R1 = glm::rotate(glm::mat4(1), glm::radian(-45), glm::vec3(0, 0, 1)); // z-축 -45도 회전
glm::mat4 S = glm::scale(glm::mat4(1), glm::vec3(1, -1, 1)); // x-축 반사
glm::mat4 R2 = glm::rotate(glm::mat4(1), glm::radian(45), glm::vec3(0, 0, 1)); // z-축 45도 회전
glm::mat4 T = glm::translate(glm::mat4(1), glm::vec3(2, -2, 0)); // 이동
glm::mat4 M = T * R2 * S * R1
```

4) 위(2.3)의 행렬을 직접 행렬의 곱을 계산하여 (2.2)와 같음을 증명한다. 3x3 아핀 변환 행렬을 사용할 것.

$$M = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ -\sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45 & -\sin 45 & 2 \\ \sin 45 & \cos 45 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & \sin 45 & 0 \\ \sin 45 & -\cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos 45 \cos 45 - \sin 45 \sin 45 & \cos 45 \sin 45 + \sin 45 \cos 45 & 2 \\ \sin 45 \cos 45 + \cos 45 \sin 45 & \sin 45 \sin 45 - \cos 45 \cos 45 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

5) 임의의 두 점(2.1)을 이용하여 직선 L에 반사되는 행렬 M(2.2)을 곱하여 변환된 점들이 (2.1)과 같은 값이 나옴을 증명한다.

$$P_1' = M * P_1 = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

$$P_2' = M * P_2 = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

3. 다음은 3차원 공간에서 임의의 축 (arbitrary axis) (1, 1, 0)에 대한 45도 각도 회전 Rotation Vector를 계산하는 과정이다. 아래의 질문에 답하라. (각 3점 총 18점)

1) 임의의 축 a(1, 1, 0)를 정규화한 $\bar{a}$을 구하라.

$$\boxed{\bar{a} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0 \right)}$$

2) $\bar{a}$와 정점 X(3, 3, -1)간의 외적(cross product) w를 계산하라.

$$w = \bar{a} \times x = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right) \times (3,3,-1) = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$$

3) 정점 X(3, 3, -1)이 $\bar{a}$에 평행한 벡터 $X_{\parallel}$를 계산하라.

$$x_{||} = (\bar{a} \bullet x)\bar{a} = 3\sqrt{2}\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right) = (3,3,0)$$

4) 정점 X(3, 3, -1)이 $\bar{a}$에 수직인 벡터 $X_{\perp}$를 계산하라.

$$x_{\perp} = x - (\bar{a} \bullet x)\bar{a} = (3,3,-1) - (3,3,0) = (0,0,-1)$$

5) 벡터 $X_{\perp}$가 $\bar{a}$에 45도 회전한 벡터 $R(X_{\perp})$를 계산하라.

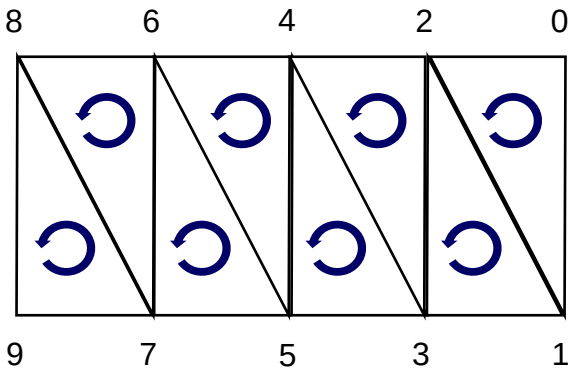
$$\begin{aligned} R(x_{\perp}) &= \cos\theta x_{\perp} + \sin\theta w \\ &= \frac{1}{\sqrt{2}}(0,0,-1) + \frac{1}{\sqrt{2}}(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0) \\ &= (0,0,-\frac{1}{\sqrt{2}}) + (-\frac{1}{2}, \frac{1}{2}, 0) \\ &= (-\frac{1}{2}, \frac{1}{2}, -\frac{1}{\sqrt{2}}) \end{aligned}$$

6) 정점 X(3, 3, -1)이 임의의 축 a 에 45 회전한 벡터 R(X)를 계산하라.

$$\begin{aligned} R(x) &= R(x_{||}) + R(x_{\perp}) \\ &= (3,3,0) + (-\frac{1}{2}, \frac{1}{2}, -\frac{1}{\sqrt{2}}) \\ &= (\frac{5}{2}, \frac{7}{2}, -\frac{1}{\sqrt{2}}) \end{aligned}$$

학과 _____ 학번 _____ 이름 _____

4. GL_LINES, GL_LINE_STRIP, GL_TRIANGLES, GL_TRIANGLE_STRIP을 사용하여, 다음 도형을 wireframe/fill 그리기 위한 정점 리스트(정점 인덱스 사용)를 적어라. (각 3점 총 12점)



GL_LINES	GL_LINE_STRIP	GL_TRIANGLES	GL_TRIANGLE_STRIP
1 2	1	0 1 2	0
2 3	2	1 2 3	1
3 4	3	2 3 4	2
4 5	4	3 4 5	3
5 6	5	4 5 6	4
6 7	6	5 6 7	5
7 8	7	6 7 8	6
8 9	8	7 8 9	7
9 7	9		8
7 5	7		9
5 3	5		
3 1	3		
1 0	1		
0 2	0		
2 4	2		
4 6	4		
6 8	6		
	8		

학과 _____ 학번 _____ 이름 _____

5. 다음 물음에 답하시오. (20점)

1) SimpleCar 클래스의 계층적 변환(hierachical transformation) 구조를 3차원 공간에서 정확한 척도로 표시하라. 또한 이 계층적 변환 구조에서 중심의 위치를 표시해준다. (10점)

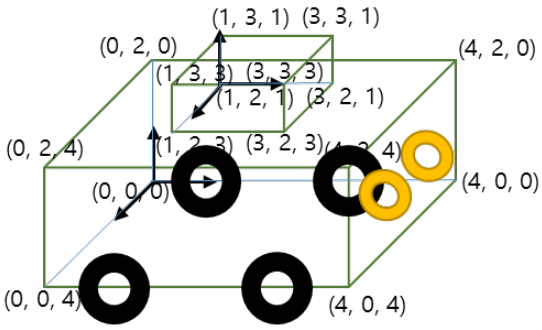
```
// SimpleCar.cpp
void SimpleCar::init()
{
	geo[0] = Parallelepiped(glm::vec3(0, 0, 0), glm::vec3(1, 0, 0), glm::vec3(0, 0, 1), glm::vec3(0, 1, 0));
	geo[0].setColor(glm::vec3(0, 1, 0)); // green parallelepiped
	geo[1] = Torus(glm::vec3(0, 0, 0), 0.5, 0.1, 16, 16);
	geo[1].setColor(glm::vec3(1, 1, 0)); // yellow torus
	geo[2] = Torus(glm::vec3(0, 0, 0), 0.5, 0.1, 16, 16);
	geo[2].setColor(glm::vec3(0, 0, 0)); // black torus
}

void SimpleCar::draw(Program* p, glm::mat4& projection, glm::mat4& view, glm::mat4& model)
{
	p->useProgram();
	p->setUniform("gProjection", projection);
	p->setUniform("gView", view);

	mat4 mMatrix = model * scale(mat4(1), vec3(4, 2, 4));
	p->setUniform("gModel", mMatrix);
	geo[0].draw(); // (1) lower body
	mMatrix = model * translate(mat4(1), vec3(1, 2, 1)) * scale(mat4(1), vec3(2, 1, 2));
	p->setUniform("gModel", mMatrix);
	geo[0]->draw(); // (2) upper body

	mMatrix = model * translate(mat4(1), vec3(4, 1, 1)) * rotate(mat4(1), M_PI/2, vec3(0, 1, 0)) * scale(mat4(1), vec3(0.5, 0.5, 2));
	p->setUniform("gModel", mMatrix);
	geo[1]->draw(); // (3) left head light
	mMatrix = model * translate(mat4(1), vec3(4, 1, 3)) * rotate(mat4(1), M_PI/2, vec3(0, 1, 0)) * scale(mat4(1), vec3(0.5, 0.5, 2));
	p->setUniform("gModel", mMatrix);
	geo[1]->draw(); // (4) right head light

	mMatrix = model * translate(mat4(1), vec3(1, 0, 0));
	p->setUniform("gModel", mMatrix);
	geo[2]->draw(); // (5) left back wheel
	mMatrix = model * translate(mat4(1), vec3(1, 0, 4));
	p->setUniform("gModel", mMatrix);
	geo[2]->draw(); // (6) right back wheel
	mMatrix = model * translate(mat4(1), vec3(3, 0, 0));
	p->setUniform("gModel", mMatrix);
	geo[2]->draw(); // (7) left front wheel
	mMatrix = model * translate(mat4(1), vec3(3, 0, 4));
	p->setUniform("gModel", mMatrix);
	geo[2]->draw(); // (8) right front wheel
}
```



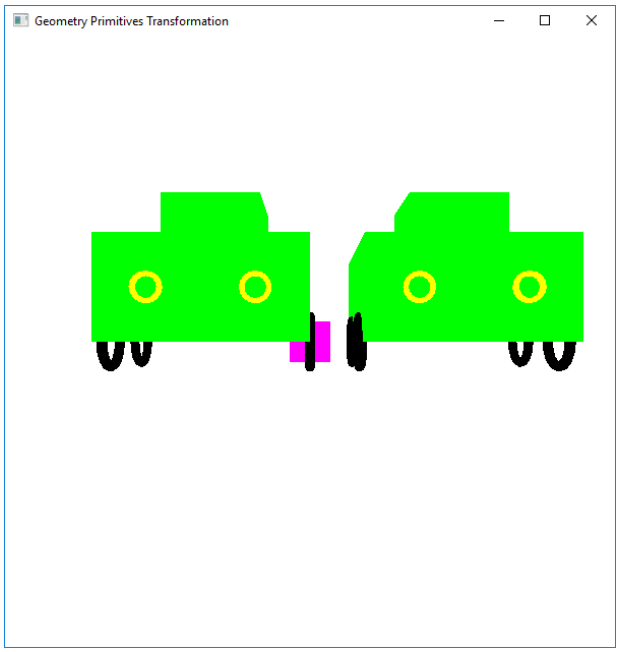
2) main.cpp 의 draw()함수의 실행 결과를 보여라. 3차원 공간에서 정확한 척도로 표시한다. 또한 car의 중심 위치를 표시해준다. (10점)

```
// main.cpp
void draw()
{
        glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

        // MVP matrix
        Projection = glm::perspective(g_fovy, g_aspect, g_zNear, g_zFar);
        View = glm::lookAt(g_eye, g_at, g_up);
        spMain.useProgram();
        spMain.setUniform("gProjection", Projection);
        spMain.setUniform("gView", View);

        glm::mat4 R = glm::rotate(glm::mat4(1.0f), glm::radians(-90.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        World = R;
        spMain.setUniform("gModel", World);
        car->draw(&spMain, Projection, View, World); //(1)

        glm::mat4 T2 = glm::translate(glm::mat4(1.0f), glm::vec3(5.0f, 0.0f, 0.0f));
        World = T2 * R;
        car->draw(&spMain, Projection, View, World); //(2)
}
```



- 끝 -