단국대학교 멀티미디어공학 그래픽스 프로그래밍 중간고사 (2009년 봄학기) 2009년 4월 21일

학과 _____ 학번 _____ 이름 _____

# 중간고사

**담당교수: 단국대학교 멀티미디어공학전공 박경신**

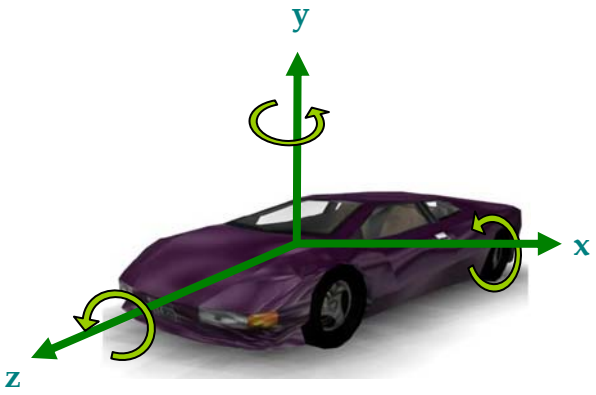
- **답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.**

**1. 맞으면 true, 틀리면 false를 적으시오. (20점)**

1) OpenGL에서 다각형의 winding order는 반시계 방향이다. ___T____
2) 프레임 버퍼는 벡터 그래픽 시스템에서 사용한다. ____F___
3) 볼록한(convex) 객체는 객체 내에 임의의 두 점을 연결하는 선분 위에 놓인 임의의 점이 객체 내에 있다. ____T___
4) 아핀 변환(Affine Transformation)은 선형성 (Colinearity)과 거리의 비례 (Ratio of distance)를 유지한다. ___T____
5) 벡터는 공간 내에 고정된 위치를 갖지 않는다. ___T____
6) 정방행렬 (Square Matrix)는 반드시 역행렬 (Inverse Matrix)를 갖는다. ____F___
7) glutPostRedisplay() 함수는 윈도우가 새로 그려져야할 필요가 있는 경우를 표시할 때 사용된다. ____T___
8) 샘플 모드 (Request mode) 입력장치에서는 장치가 트리거 될 때까지 측정치(Measure)가 프로그램으로 반환되지 않는다. ___F____
9) glutBitmapCharacter() 함수를 사용한 문자는 GL의 변환(즉, 위치, 크기, 방향)에 의해 영향을 받는다. ___F____
10) 사원수 (Quaternion)의 곱은 교환법칙이 성립된다. ___F____

**2. 다음 문제에 답하시오. (30점)**

1) 아래의 자동차 그림 위에 OPENGL의 좌표계 (x, y, z축)과 회전 (rotation) 방향을 표시하시오.



학과 _____ 학번 _____ 이름 _____

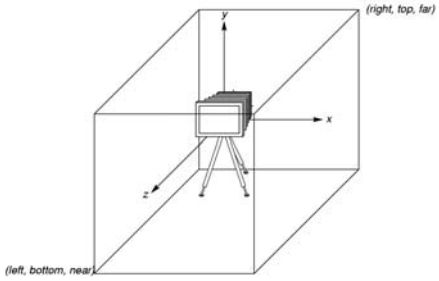
2) 아래의 빈칸을 채우시오.

점 p가 평면 (n, d)에서 만약 $n \bullet p + d < 0$ 이라면 p는 평면 __안쪽____에 있다.
점 p가 평면 (n, d)에서 만약 $n \bullet p + d = 0$ 이라면 p는 평면 __위____에 있다.
점 p가 평면 (n, d)에서 만약 $n \bullet p + d > 0$ 이라면 p는 평면 __바깥쪽____에 있다.

3) glutDisplayFunc(void (func*)(void)) 함수를 설명하시오.

glutDisplayFunc(display)의 void display() 답신함수는 GLUT에서 윈도우가 새로 그리기(window refresh)를 요할 때 불려진다. 모든 GLUT 프로그램에서 반드시 불리는 함수이다.
-윈도우가 처음으로 열렸을 때
-윈도우가 재구성이 필요할 때
-윈도우가 expose됐을 때
-사용자 프로그램에서 디스플레이가 바뀌길 원할 때

4) OpenGL에서 기본 카메라(Default Camera)의 위치와 관측공간을 간단히 설명하고 그림으로 표시하라.




OpenGL에서는 카메라가 물체의 공간(drawing coordinates)의 원점(origin)에 위치하며 z- 방향으로 향하고 있다. 관측공간을 지정하지 않는다면, 디폴트로 2x2x2 입방체의 viewing volume을 사용한다.

5) 다음 3차원 회전행렬 공식에서 Symmetric Matrix를 유도하라.

$$R = \mathrm{I}\cos\theta + \mathbf{Symmetric}\ (1-\cos\theta) + \mathbf{Skew}\ \sin\theta$$

$$\mathrm{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \bar{a}(\bar{a}\cdot\bar{x})$$

$$\mathbf{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right)\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \bar{a}(\bar{a}\cdot\bar{x}) = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\left(a_x x + a_y y + a_z z\right)$$

$$x' = a_x^2 x + a_x a_y y + a_x a_z z$$

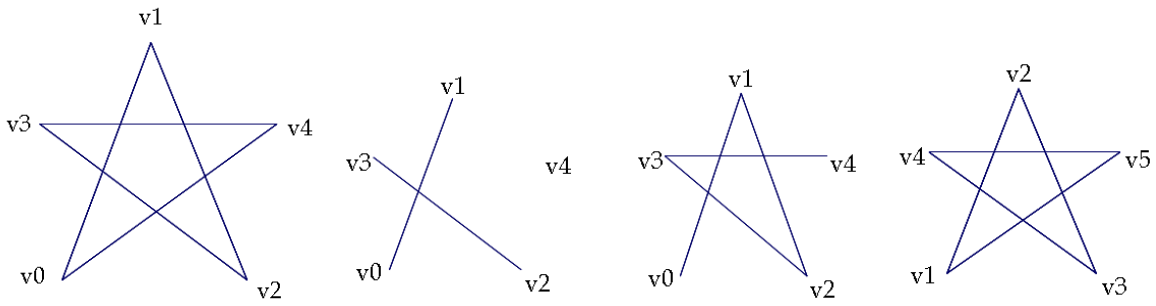
$$y' = a_x a_y x + a_y^2 y + a_y a_z z$$

$$z' = a_x a_z x + a_y a_z y + a_z^2 z$$

$$\mathbf{Symmetric} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\begin{bmatrix} a_x & a_y & a_z \end{bmatrix} = \begin{bmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{bmatrix}$$

6) 아래의 그림을 참고하여 OPENGL Geometry Primitives 예제에서 기하학적 객체의 모드로 GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP를 그렸을 때 차이점을 설명하라.

```
glBegin(GL_LINES);
for (int i=0; i<5; i++)
        glVertex2fv(v[i]);
glEnd();

glBegin(GL_LINE_STRIP);
for (int i=0; i<5; i++)
        glVertex2fv(v[i]);
glEnd();

glBegin(GL_LINE_LOOP);
for (int i=0; i<5; i++)
        glVertex2fv(v[i]);
glEnd();
```



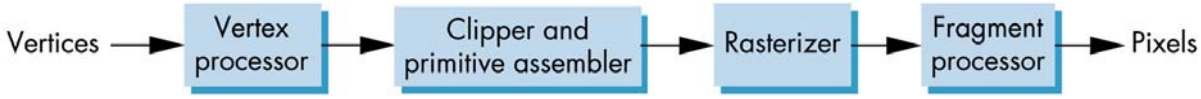

**3. 다음은 실습숙제1에서 OPENGL Geometry를 사용하여 우주선 (Spaceship)을 그리는 함수를 보여주고 있다. 우주선의 색을 표시하여 출력 결과를 그려라. (10점)**

```
// draw the ship where (cx, cy) is the center position of the spaceship
void Ship::draw()
{
    glBegin(GL_POLYGON);      // base
        glColor3f(0.0, 0.0, 1.0);
        glVertex2f(cx,     cy + 20);
        glVertex2f(cx - 4, cy + 15);
        glVertex2f(cx - 4, cy + 5);
        glVertex2f(cx + 4, cy + 5);
        glVertex2f(cx + 4, cy + 15);
    glEnd();
```

```
    glBegin(GL_POLYGON);      // base left wing
        glColor3f(0.0, 1.0, 0.0);
        glVertex2f(cx - 4, cy + 5);
        glVertex2f(cx - 4, cy + 15);
        glVertex2f(cx - 13, cy + 5);
    glEnd();

    glBegin(GL_POLYGON);      // base right wing
        glColor3f(0.0, 1.0, 0.0);
        glVertex2f(cx + 4, cy + 5);
        glVertex2f(cx + 13, cy + 5);
        glVertex2f(cx + 4, cy + 15);
    glEnd();

    glBegin(GL_POLYGON);      // left wing
        glColor3f(1.0, 1.0, 1.0);
        glVertex2f(cx - 4, cy + 5);
        glVertex2f(cx - 4, cy + 15);
        glVertex2f(cx - 10, cy + 2);
    glEnd();

    glBegin(GL_POLYGON);      // right wing
        glColor3f(1.0, 1.0, 1.0);
        glVertex2f(cx + 4, cy + 5);
        glVertex2f(cx + 10, cy + 2);
        glVertex2f(cx + 4, cy + 15);
    glEnd();

    glBegin(GL_POLYGON);      // big fire
        glColor3f(1.0, 1.0, 0.0);
        glVertex2f(cx,      cy);
        glVertex2f(cx + 4, cy + 5);
        glVertex2f(cx - 4, cy + 5);
    glEnd();

    glBegin(GL_POLYGON);      // small fire
        glColor3f(1.0, 0.0, 0.0);
        glVertex2f(cx,      cy + 2);
        glVertex2f(cx + 2, cy + 5);
        glVertex2f(cx - 2, cy + 5);
    glEnd();
}
```

학과 _____ 학번 _____ 이름 _____

**4. 다음은 그래픽스 기하 파이프라인 (Geometric Pipeline) 구조를 보여주고 있다. 각각의 단계를 간단히 설명하라. (10점)**




**Vertex Processor (정점 처리):**
각 정점 (Vertex)의 좌표계 변환을 수행하고 (예: model transformation, viewing transformation 등) 각 정점의 색을 계산 처리(예: 광원의 특성과 객체 표면의 물리적인 특성을 고려하여 색을 계산)한다.

**Clipper and Primitive Assembler (클리핑과 기본요소로 조립):**
카메라가 찍을 수 있는 시야각 영역, 즉 클리핑 볼륨 (Clipping Volume)을 두어, 이것의 밖에 투영되는 객체는 영상으로 나타나지 않고 클리핑되도록 하고, 클리핑 볼륨에 걸친 객체들은 영상에 부분적으로 보이도록 한다. 클리핑을 위하여 정점의 집합을 선분과 다각형과 같은 기본요소로 조립한다. 이 단계의 출력은 투영변환(Projection)되어 영상으로 표시되는 기본 요소의 집합이다.

**Rasterizer (래스터화):**
클리핑기로부터 나온 기본 요소의 집합은 정점으로 표현되어 있는데 이를 프레임 버퍼의 픽셀로 변환되는 과정이다. 이 단계의 출력은 각 기본 요소 단편 (fragment)의 집합이다. 단편은 색과 위치 정보를 전달하는 잠정적인 픽셀값이다.

**Fragment Processor (단편 처리):**
단편처리기에서는 단편을 받아들여 프레임 버퍼 안에 있는 픽셀을 갱신한다. 단편 처리기에서 깊이버퍼를 사용한 은면제거와 택스쳐 맵핑 (혹은 범프 매핑), 반투명 효과를 주는 알파 블렌딩 등이 수행된다.

학과 _____ 학번 _____ 이름 _____

**5. 다음은 디스플레이 리스트 (Display List)로 구성한 3차원 객체와 지적 (Picking)이 사용된 OPENGL 프로그램 일부를 보여주고 있다. 디스플레이 리스트를 사용하여 구성한 3차원 객체의 출력 결과 (즉, drawObject)와 전체적인 장면의 출력 결과를 그림으로 나타내라 (10점). 그리고, 각 함수에 작동원리를 주석으로 달아라 (10점).**

```
// __디스플레이 리스트를 사용하여 눈사람 객체를 구성_____________________________
void initDL()
{
	GLfloat white[] = {1, 1, 1, 1};
	GLfloat black[] = {0, 0, 0, 1};
	GLfloat orange[] = {1.0f, 0.5f , 0.5f};

	glNewList(1, GL_COMPILE);

	glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, white);
	// base body
	glPushMatrix();
	glTranslatef(0.0f, 1.0f, 0.0f);
	gluSphere(quadric, 1.5f, 20, 20);

		// upper body
		glPushMatrix();
		glTranslatef(0.0f, 2.0f, 0.0f);
		gluSphere(quadric, 1.0f, 20, 20);

			// head
			glPushMatrix();
			glTranslatef(0.0f, 1.3f, 0.0f);
			gluSphere(quadric, 0.6f, 20, 20);

				// eyes
				glPushMatrix();
				glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, black);
				glTranslatef(0.2f, 0.1f, 0.6f);
				gluSphere(quadric, 0.1f, 10, 10);
				glPopMatrix();

				glPushMatrix();
				glTranslatef(-0.2f, 0.1f, 0.6f);
				gluSphere(quadric, 0.1f, 10, 10);
				glPopMatrix();

				// nose
				glPushMatrix();
				glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, orange);
				glTranslatef(0.0f, -0.1f, 0.6f);
				glRotatef(0.0f,1.0f, 0.0f, 0.0f);
				gluCylinder(quadric, 0.1f, 0.0, 0.35, 10, 2);
				glPopMatrix();

			glPopMatrix();
		glPopMatrix();
	glPopMatrix();
```

```
	glEndList();
}

// _디스플레이 리스트를 사용하여 객체를 그림_____________________________________
void drawObject(float x, float y, float z)
{
	glPushMatrix();
	glTranslatef(x, y, z);
	glCallList(1);
	glPopMatrix();
}

// __5개의 객체를 일반 렌더링 모드로 그리거나, SELECT 모드에선 네임스택에 ID를 지정____
void drawObjects(GLenum mode)
{
	if (mode == GL_SELECT) glLoadName(1);
	drawObject(0, -3, -3);
	if (mode == GL_SELECT) glLoadName(2);
	drawObject(-3, -3, 3);
	if (mode == GL_SELECT) glLoadName(3);
	drawObject(3, -3, 0);
	if (mode == GL_SELECT) glLoadName(4);
	drawObject(-6, -3, 3);
	if (mode == GL_SELECT) glLoadName(5);
	drawObject(6, -3, -3);
}

// __지면을 그림_____________________________________________________________
void drawGround()
{
	glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, darkbrown);

	glBegin(GL_QUADS);
	for(int i = -10;i <= 10; ++i) {
		glNormal3f(0, 1, 0);
		glVertex3f(i*5, -3, -50);
		glVertex3f(i*5, -3, 50);

		glVertex3f(50, -3, i*5);
		glVertex3f(-50, -3, i*5);
	}
	glEnd();
}

// _OPENGL 초기화 함수에서 display list를 초기화_____________________________
void init(void)
{
	glClearColor(0.0f, 0.0f, 0.0f, 0.0f); // Clear The Background Color To Blue
	glClearDepth(1.0);                    // Enables Clearing Of The Depth Buffer
	glDepthFunc(GL_LESS);                 // The Type Of Depth Test To Do
	glEnable(GL_DEPTH_TEST);              // Enables Depth Testing
	glShadeModel(GL_SMOOTH);              // Enables Smooth Color Shading

	// lighting
	GLfloat lightPos[4] = {-1, 1, 1, 0};
```

```
	glLightfv(GL_LIGHT0, GL_POSITION, lightPos);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, white);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);

	// quadric
	quadric = gluNewQuadric();
	gluQuadricNormals(quadric, GLU_SMOOTH);
	gluQuadricDrawStyle(quadric, GLU_FILL);

	for (int i = 0; i < 5; i++)
		initDL();
}

// _지면과 일반 렌더링 모드에서 객체를 그림_____________________________________
void display()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(60, 1, 0.1, 1000);
	glMatrixMode(GL_MODELVIEW);
	// 중간 생략..

	drawGround();
	drawObjects(GL_RENDER);

	glutSwapBuffers();
}

// _다수의 hits가 존재할 시 가까운 것으로 선택_____________________________________
int processHits (GLint hits, GLuint buffer[])
{
	int i, j, choose, depth;

	// if there is more than 0 hits
	if (hits > 0) {
		printf ("hits = %d\n", hits);
		choose = buffer[3];				// make the selection the first object
		depth = buffer[1];				// store how far away it is

		for (i = 1; i < hits; i++){			//   for all detected hits
			// if this object is closer to us than the one we selected
			if (buffer[i*4+1] < GLuint(depth)) {
				choose = buffer[i*4+3];	// select the closer object
				depth = buffer[i*4+1];	// store how far away it is
			}
		}
		return choose;
	}
	return -1;
}
```

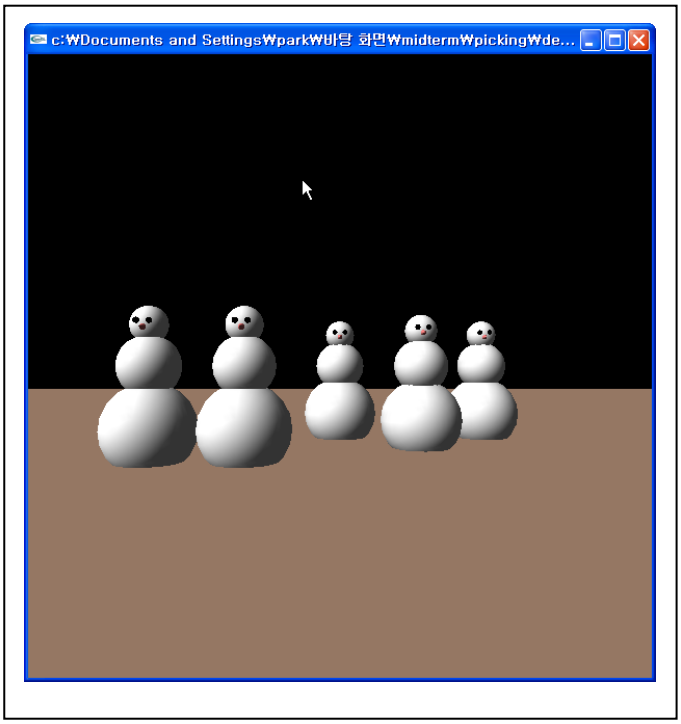
// __GL_SELECT 모드로 전환하고, 마우스 위치의 영역과 객체를 렌더링한 후
_ 다시 GL_RENDER 모드로 전환하면 다수의 hits가 넘겨져 오고,
_ processHits 함수를 호출하여 어느 것이 선택되었는지 알아냄_____

```
void pick(int x, int y)
{
	GLuint selectBuf[512];
	GLint viewport[4];
	GLint hits;
	int id;

	glSelectBuffer(512, selectBuf);		// selectBuf is where to store the hits
	glRenderMode(GL_SELECT);		// GL_SELCT mode
	glInitNames();				// clear name stack
	glPushName(0);				// fill the stack with one element

	// modify the viewing volume, restricting selection area around the cursor
	glMatrixMode(GL_PROJECTION);		// select the projection matrix
	glPushMatrix();				// push the projection matrix
		glLoadIdentity();
		glGetIntegerv(GL_VIEWPORT, viewport);
		gluPickMatrix((GLdouble)x, (GLdouble)(viewport[3]-y), 1, 1, viewport);
		gluPerspective(60, (float)viewport[2]/(float)viewport[3], 0.1, 1000);

		glMatrixMode(GL_MODELVIEW);	// draw object onto the screen
		drawObjects(GL_SELECT);		// i.e., draw only the names in the stack

		glMatrixMode(GL_PROJECTION);	// select the projection matrix
	glPopMatrix();				// pop the projection matrix
	glMatrixMode(GL_MODELVIEW);		// select the modelview matrix
	glFlush();

	// swith to render mode, get hit info
	hits = glRenderMode(GL_RENDER);
	id = processHits(hits, selectBuf);
	printf ("You picked object %d\n", id);
}
```

// __왼쪽 마우스 버튼이 눌리면, pick 연산이 호출_____

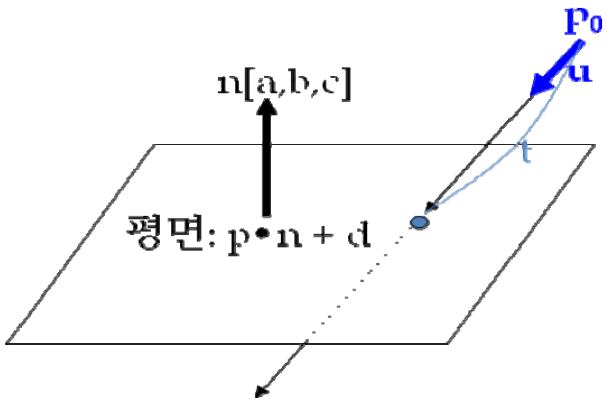
```
void mouse(int button,int state,int x,int y)
{
	prevX = x;
	prevY = y;
	switch(button)
	{
	case GLUT_LEFT_BUTTON:
		mbuttons[0] = ((GLUT_DOWN==state)?1:0);
		if (mbuttons[0]) pick(x, y);
		break;
	case GLUT_MIDDLE_BUTTON:
		mbuttons[1] = ((GLUT_DOWN==state)?1:0);
		break;
	case GLUT_RIGHT_BUTTON:
		mbuttons[2] = ((GLUT_DOWN==state)?1:0);
		break;
	default:
```

```
                    break;
            }
            glutPostRedisplay();
}

int main(int argc, char** argv)
{
            glutInit(&argc, argv);
            glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
            glutInitWindowSize(600, 600);
            glutInitWindowPosition(0,0);
            glutCreateWindow(argv[0]);
            glutReshapeFunc(reshape);
            glutDisplayFunc(display);
            glutKeyboardFunc(keyboard);
            glutSpecialFunc(specialkey);
            glutMouseFunc(mouse);
            glutMotionFunc(motion);

            init();
            glutMainLoop();
            return 0;
}
```



학과 _____ 학번 _____ 이름 _____

**6. 다음은 광선(Ray)이 평면(Plane)과 교차하는 공식을 표현한 간단한 코드이다. 빈칸에 계산한 값을 적으시오. (10점)**




```
class plane {
public:
        float a, b, c, d;
        // constructor..
        plane(float a_, float b_, float c_, float d_);

        // static utility methods (plane p: a, b, c, d & vector3 v: x, y, z)
        static float dotNormal(const plane & p, const vector3 & v);    // a*x + b*y + c*z + d*0
        static float dotCoord(const plane & p, const vector3 & v);     // a*x + b*y + c*z + d*1

        // 중간생략...
};

typedef struct _RAY {
        vector3 p0;         // 시작점
        vector3 u;          // 방향
} RAY;

bool RayPlaneIntersection(plane p, RAY line, vector3& out)
{
    float denom = plane::dotNormal(p, line.u);
    if (denom == 0)
        return false;

    float t = -plane::dotCoord(p, line.p0)/denom;
```

```
    if (t < 0)
        return false;

    out = line.p0 + t*(line.u);      // return vector

    return true;
}

int main(int argc, char *argv[])
{
        plane plane0(1, 0, 0, 1);
        RAY ray1, ray2, ray3;
        ray1.p = vector3(1, 0, 0);
        ray1.u = vector3(-1, -1, 0);
        ray2.p = vector3(0, 0, -2);
        ray2.u = vector3(1, 1, 1);
        ray3.p = vector3(1, 1, 0);
        ray3.u = vector3(1, 1, 1);
        vector3 out1, out2, out3;

        if (RayPlaneIntersection(plane0, ray1, out1))
                cout << "out1          = " << out1 << endl; // __out1 = (-1, -2, 0)_____
        if (RayPlaneIntersection(plane0, ray2, out2))
                cout << "out2          = " << out2 << endl; // ___nothing_____
        if (RayPlaneIntersection(plane0, ray3, out3))
                cout << "out3          = " << out3 << endl; // ___nothing_____

        return 0;
}
```