Viewing

321190
2013년 봄학기
5/7/2013
박경신

Camera Movement

- OpenGL에서 카메라 효과를 주기 위하여, display함수의 시작부분에 카메라의 움직임에 반대되는 변환행렬을 적용시키면 된다.
- 예를 들어, 카메라를 원점에서 10 units만큼 +Z로 움직이려면, world를 -10 units만큼 움직이면 된다.

```
void display()
{
    Projection = glm::perspective(45, 1, 0.1, 1000);
    View = glm::mat4(1.0f); // Identity matrix
    World = glm::translate(glm::mat4(1.0f), glm::vec3(0, 0, -10));
    drawObjects();
}
```

Camera Movement

- 일반적인 카메라 움직임 (Camera movement)은 카메라의 위치와 방향 (Camera position & orientation)을 world에 역변환행렬 (Inverse transformation)로 적용한다.
- 예

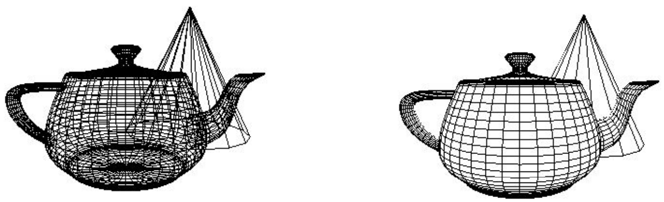
```
float cameraX, cameraY, cameraZ, cameraHeading;
void display()
{
    Projection = perspective(45, 1, 0.1, 1000);
    View = mat4(1.0f); // Identity matrix
    World = rotate(mat4(1.0f), -cameraHeading, vec3(0, 1, 0))
            * translate(mat4(1.0f), vec3(-cameraX, -cameraY, -cameraZ));
    drawObjects();
}
```

Camera Movement

- Navigation
  - Fly-through (6DOF)
    - Yaw (y), pitch (x), roll (z) orientation
    - Walk forward/backward (z), strafe right/left (x), fly up/down (y) movement
  - Walk-through (2DOF)
    - Pan (y)
    - Walk forward/backward (z)

Hidden Surface

- 은면(Hidden surfaces)은 occlusion depth cue를 제공한다.
- 컴퓨터 그래픽스에서, 가려짐 (occlusion)이란 용어는 뷰포트로부터 가까운 물체가 뷰포트에서 멀리있는 물체를 가리는 것을 말한다.
- 그래픽스 파이프라인에서 occlusion culling으로 세이딩(shading)과 래스터화(rasterization)하기 전에 은면 제거(hidden surface removal)을 한다.



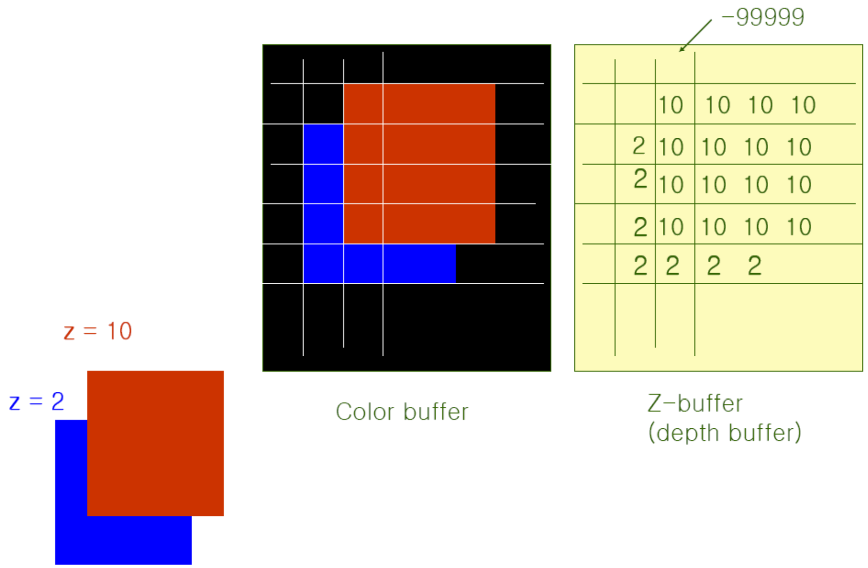
Hidden Surface Removal

- 은면 제거 (Hidden Surface Removal) 알고리즘
  - 객체 공간기법 – 객체나 객체 부분들을 서로 비교하여 전체적으로 어느 면과 선이 보이지 않는 것인지 결정
    - 깊이 정렬 알고리즘 (Depth-sorting algorithm) – 폴리곤의 각 면을 깊이에 따라 정렬한 뒤, 먼 것부터 투영하여 그린다. Painter's algorithm라고도 불린다.
    - Binary Space Partitioning (BSP) tree – BSP tree를 사용하여 관측 방향에 따라 front, back을 구분하여 공간을 계속적으로 분할한다.
  - 이미지 공간 기법 – 투영 과정의 일부분으로 동작하여, 각 투영선 위의 객체 화소 위치에서 점 단위로 가시성이 결정
    - Z-buffer (depth buffer) – 가장 일반적으로 사용되는 이미지 공간 기법으로, 물체의 가시성을 화소 단위로 조사하여 z (깊이) 값이 가장 작은 평면의 값을 그린다. Z값을 저장하는 깊이버퍼 (z-buffer)가 필요하다.
    - Ray-casting – 시점에서 투영면의 각 화소를 통해 빛 (ray)를 투사하고, 이 빛과 처음으로 만나는 객체를 선택하여 해당 픽셀을 그린다. 임의의 곡면과 같은 표면에서 효과적인 은면 제거 방법이다.

Hidden Surface Removal

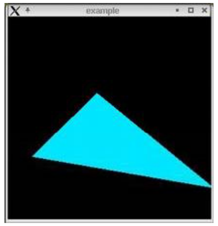
- 은면 제거가 적용된 다양한 방법
  - 깊이 정보 테스트
    - glEnable(GL_DEPTH_TEST);
  - 표면/이면 제거
    - glEnable(GL_CULL_FACE);
    - glCullFace(GL_FRONT);
    - glCullFace(GL_BACK);

Z-buffer

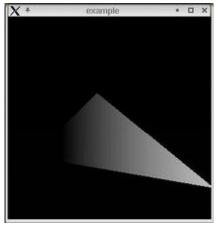


Z-buffer

- 폴리곤 렌더링이란 결국 픽셀로 채워지는 것을 의미한다.
- 컬러 버퍼 (Color buffer)는 그리고자 하는 픽셀당 RGB 색 정보를 가진다.
- 깊이 버퍼 (Z-buffer, depth buffer)는 그리고자 하는 픽셀당 깊이 정보 (depth value)를 가진다.



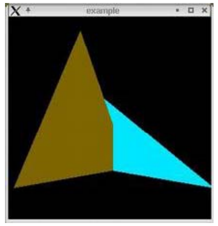
Color buffer



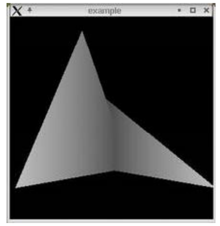
Depth buffer

Z-buffer Algorithm

- Z-buffer algorithm은 새로운 픽셀을 그릴 때마다, 새로운 깊이 정보를 깊이 버퍼 (z-buffer) 안에 있는 깊이(depth) 정보와 비교한다.
- 폴리곤 (Polygons)은 어떠한 방향에서도 그려질 수 있으며 교차할 수도 있다.



Color buffer



Depth buffer

OpenGL Z-buffering

- OpenGL에서 z-buffer를 사용하려면 먼저 깊이 버퍼를 초기화하고, 깊이정보 테스트를 활성화한다.

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
```

- 매 프레임마다 깊이 버퍼를 지운다.

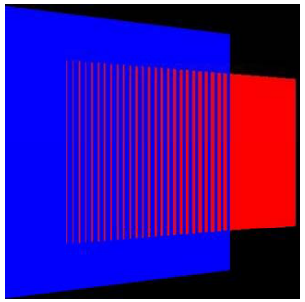
```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
```

- 정육면체와 같은 객체의 경우 관측자로부터 멀어지는 방향을 향하는 모든 면을 제거하고자 할 때 사용한다.

```
glEnable(GL_CULL);
```

Depth Fighting

- Z-buffer의 깊이 값은 한정된 해상도를 갖고 있다.
- 깊이 버퍼에서 아주 가까운 깊이 값(depth value)을 가지는 폴리곤의 중첩(overlap)은 “depth-fighting”을 만든다.
- 폴리곤이 그려질 때 부동 소수점 반올림 에러 (floating point round-off errors) 때문에 생기는 현상으로, 폴리곤 임의의 부분이 서로 렌더링하려는 현상이다.

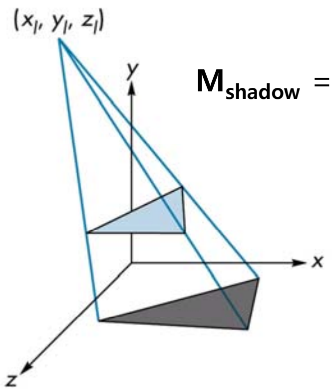


Projections and Shadows

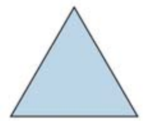
- 물리적으로 그림자는 하나 이상의 광원을 요구 - 즉, 광선과 재질간의 상호작용이 필요하다.
- 그림자 행렬 (Shadow Matrix)
  - 그림자는 y=0에 떨어져있다고 가정하고 그림자 다각형을 생성한다.
  - 임의의 위치에 있는 광원에서 시작하여 광원이 원점에 있도록 하면 원점을 통한 간단한 투시 투영을 획득한다.
  - 다시 원래의 위치로 이동 (즉, 이동->투시투영 획득->이동)을 통해서 특정 도형에 대한 그림자를 획득할 수 있다.
  - 그리고, 같은 다각형에 대해 두 번의 렌더링(즉, 정상적인 다각형 렌더링 & 그림자 행렬을 적용한 다각형 렌더링)을 통해 그림자를 획득한다.

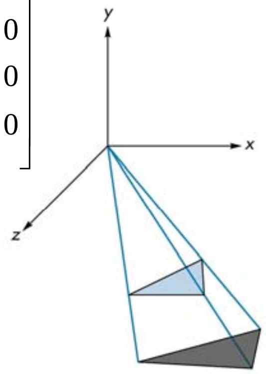
Projections and Shadows

- 투시투영을 사용한 간단한 그림자 생성



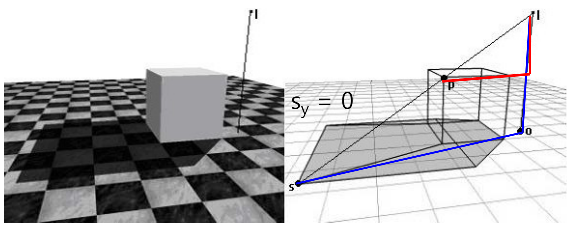
$$\mathbf{M}_{shadow} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{-y_l} & 0 & 0 \end{bmatrix}$$





(b)

Planar Shadow [J. Blinn, 88]



$$\frac{l_y - p_y}{p_x - l_x} = \frac{l_y - 0}{\mathbf{s}_x - l_x}$$

$$\Rightarrow \frac{\mathbf{s}_x - l_x}{p_x - l_x} = \frac{l_y - 0}{l_y - p_y}$$

$$\mathbf{s}_x = \frac{l_y(p_x - l_x)}{l_y - p_y} + l_x$$

$$t = \frac{l_y}{l_y - p_y}$$

$$\begin{bmatrix} s_x \\ 0 \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$\mathbf{s} = l + t(p - l)$$
$$\mathbf{n} \cdot \mathbf{s} + d = 0$$
$$t = \frac{l_y}{l_y - p_y}$$

Planar Shadow [J. Blinn, 88]

$$\mathbf{s} = l + \frac{l_y}{l_y - p_y}(p - l)$$

$$s_x = l_x + \frac{l_y}{l_y - p_y}(p_x - l_x)$$

$$s_y = 0$$

$$s_z = l_z + \frac{l_y}{l_y - p_y}(p_z - l_z)$$

$$s_x = l_x + \frac{l_y}{l_y - p_y}(p_x - l_x) = \frac{l_x(l_y - p_y) + l_y(p_x - l_x)}{l_y - p_y} = \frac{l_y p_x - l_x p_y}{l_y - p_y}$$

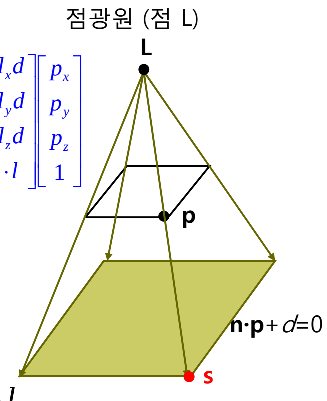
$$s_z = l_z + \frac{l_y}{l_y - p_y}(p_z - l_z) = \frac{l_z(l_y - p_y) + l_y(p_z - l_z)}{l_y - p_y} = \frac{-l_z p_y + l_y p_z}{l_y - p_y}$$

$$\begin{bmatrix} s_x \\ 0 \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$s_x = l_y p_x - l_x p_y + 0 p_z$
$s_y = 0$
$s_z = 0 p_x - l_z p_y + l_y p_z$
$w = 0 p_x - p_y + 0 p_z + l_y$

Projection Shadow

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n\cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n\cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n\cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n\cdot l \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$



$$\mathbf{s} = l + t(p - l)$$

$$\mathbf{n}\cdot\mathbf{s} + d = 0$$

$$\mathbf{n}\cdot(l + t(p-l)) + d = 0$$

$$\mathbf{n}\cdot l + t(\mathbf{n}\cdot(p-l)) = -d$$

$$t(\mathbf{n}\cdot p - \mathbf{n}\cdot l) = -d - \mathbf{n}\cdot l$$

$$t = \frac{-d - \mathbf{n}\cdot l}{\mathbf{n}\cdot p - \mathbf{n}\cdot l} \qquad \therefore\ \mathbf{s} = l + \left[\frac{\mathbf{n}\cdot l + d}{\mathbf{n}\cdot l - \mathbf{n}\cdot p}\right](\mathbf{p} - l)$$

Projection Shadow

$$\mathbf{s} = l + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p - l) \qquad s_x = l_x + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_x - l_x)$$

$$s_x = l_x + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_x - l_x) \qquad = \frac{l_x(n\cdot l - n\cdot p) + (n\cdot l + d)p_x - (n\cdot l + d)l_x}{n\cdot l - n\cdot p}$$

$$s_y = l_y + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_y - l_y) \qquad = \frac{l_x n\cdot l - l_x n_x p_x - l_x n_y p_y - l_x n_z p_z + (n\cdot l + d)p_x - l_x n\cdot l - l_x d}{-n_x p_x - n_y p_y - n_z p_z + n\cdot l}$$

$$s_z = l_z + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_z - l_z) \qquad = \frac{(n\cdot l + d - l_x n_x)p_x - l_x n_y p_y - l_x n_z p_z - l_x d}{-n_x p_x - n_y p_y - n_z p_z + n\cdot l}$$

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n\cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n\cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n\cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n\cdot l \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Projection Shadow Matrix

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n\cdot l - l_x n_x & -l_x n_y & -l_x n_z & -l_x n_w \\ -l_y n_x & n\cdot l - l_y n_y & -l_y n_z & -l_y n_w \\ -l_z n_x & -l_z n_y & n\cdot l - l_z n_z & -l_z n_w \\ -l_w n_x & -l_w n_y & -l_w n_z & n\cdot l - l_w n_w \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

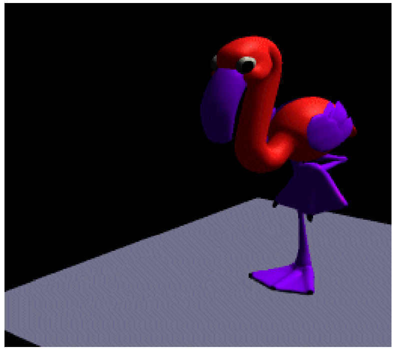
투영평면 $n = [n_x, n_y, n_z, n_w]$

광원 $l = [l_x, l_y, l_z, l_w]$ *where* $l =$ 평행광원이면 $l_w = 0,\ l =$ 점광원이면 $l_w = 1$

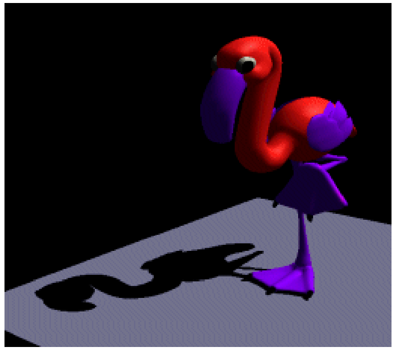
Projection Shadow Matrix

```
// create a shadow matrix that will project the desired shadow
void ShadowMatrix(GLfloat shadowMat[16], GLfloat plane[4], GLfloat lightpos[4])
{
    GLfloat dot; // dot product of light position and ground plane normal
    dot = plane[0] * lightpos[0] + plane[1] * lightpos[1] + plane[2] * lightpos[2]
          + plane[3] * lightpos[3];
    shadowMat[0] = dot - lightpos[0] * plane[0];
    shadowMat[1] = 0.f - lightpos[0] * plane[1];
    shadowMat[2] = 0.f - lightpos[0] * plane[2];
    shadowMat[3] = 0.f - lightpos[0] * plane[3];
    shadowMat[4] = 0.f - lightpos[1] * plane[0];
    shadowMat[5] = dot - lightpos[1] * plane[1];
    shadowMat[6] = 0.f - lightpos[1] * plane[2];
    shadowMat[7] = 0.f - lightpos[1] * plane[3];
    shadowMat[8] = 0.f - lightpos[2] * plane[0];
    shadowMat[9] = 0.f - lightpos[2] * plane[1];
    shadowMat[10] = dot - lightpos[2] * plane[2];
    shadowMat[11] = 0.f - lightpos[2] * plane[3];
    shadowMat[12] = 0.f - lightpos[3] * plane[0];
    shadowMat[13] = 0.f - lightpos[3] * plane[1];
    shadowMat[14] = 0.f - lightpos[3] * plane[2];
    shadowMat[15] = dot - lightpos[3] * plane[3];
}
```

Shadow

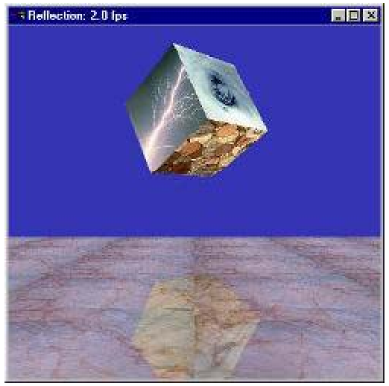


Render without shadow



Render with shadow

Reflection



http://www.gamasutra.com/features/19990723/opengl_texture_objects_02.htm

Planar Reflection

- 거울 평면 (n, d)에 대해 점 q=(x_0, y_0, z_0)의 반사 포인트 q'=(x_0', y_0', z_0')를 계산하는 방법

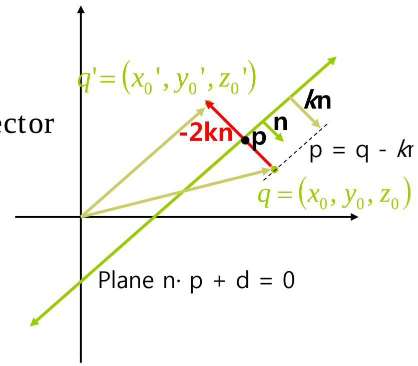
$$q' = q - 2k\mathrm{n}$$

$$= q - 2\frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2+b^2+c^2}}\mathrm{n}$$

$$= q - 2(n \cdot q + d)\mathrm{n} \text{ when n is unit vector}$$

$$q' = Rq$$

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



점 q에서 plane로의 signed shortest distance는 n이 unit vector인 경우, k = n•q + d

Planar Reflection

$$\begin{bmatrix} x_0' \\ y_0' \\ z_0' \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} - 2(ax_0 + by_0 + cz_0 + d)\begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad when\ n\ is\ unit\ vector\ (a^2+b^2+c^2=1)$$

$$x_0' = x_0 - 2(ax_0 + by_0 + cz_0 + d)a$$
$$= (1-2a^2)x_0 - 2aby_0 - 2acz_0 - 2ad$$
$$y_0' = y_0 - 2(ax_0 + by_0 + cz_0 + d)b$$
$$= 2abx_0 + (1-2b^2)y_0 - 2bcz_0 - 2bd$$
$$z_0' = z_0 - 2(ax_0 + by_0 + cz_0 + d)c$$
$$= -2acx_0 - 2bcy_0 + (1-2c^2)z_0 - 2cd$$

$$\begin{bmatrix} x_0' \\ y_0' \\ z_0' \end{bmatrix} = R\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

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

Planar Reflection

- 세 가지 특수한 경우
  - 표준 좌표 평면(*yz, xz, xy* 평면)에 대한 반사 변환 행렬

yz평면 Plane(1,0,0,0)

$$\mathbf{R}_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

xz평면 Plane(0,1,0,0)

$$\mathbf{R}_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

xy평면 Plane(0,0,1,0)

$$\mathbf{R}_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Planar Reflection Matrix

```
void ReflectionMatrix(GLfloat reflectionMat[16], GLfloat plane[4]) // create a reflection matrix
{
    reflectionMat[0] = 1 - 2 * plane[0] * plane[0];
    reflectionMat[1] =     - 2 * plane[0] * plane[1];
    reflectionMat[2] =     - 2 * plane[0] * plane[2];
    reflectionMat[3] =     - 2 * plane[0] * plane[3];

    reflectionMat[4] =     - 2 * plane[1] * plane[0];
    reflectionMat[5] = 1 - 2 * plane[1] * plane[1];
    reflectionMat[6] =     - 2 * plane[1] * plane[2];
    reflectionMat[7] =     - 2 * plane[1] * plane[3];

    reflectionMat[8] =      - 2 * plane[2] * plane[0];
    reflectionMat[9] =      - 2 * plane[2] * plane[1];
    reflectionMat[10] = 1 - 2 * plane[2] * plane[2];
    reflectionMat[11] =     - 2 * plane[2] * plane[3];

    reflectionMat[12] = 0.0;
    reflectionMat[13] = 0.0;
    reflectionMat[14] = 0.0;
    reflectionMat[15] = 1.0;
}
```