Viewing

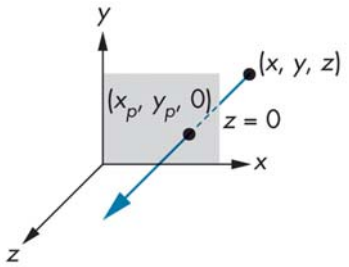
321190
2007년 봄학기
5/1/2007
박경신

Orthographic Projection

- 직교 투영 (Orthographic projection)
  - 투영선이 관측 평면에 수직인 평행투영의 특수한 경우이다.
  - 렌즈와 카메라의 뒷면이 평행하고 초점거리가 무한대이다.

Orthographic projection

$x_p = x$
$y_p = y$
$z_p = 0$
$w_p = 1$

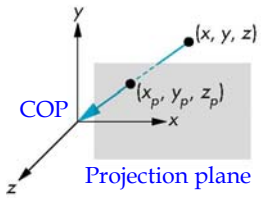


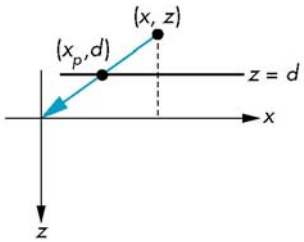
$$\mathbf{p}_p = \mathbf{M}\mathbf{p}$$

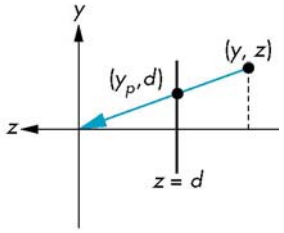
$$\mathbf{M}_{ortho} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Projection

- 원근 투영 (Perspective projection)
  - 투영중심 (Center of projection)은 원점 (Origin)
  - 투영면 (Projection plane) $z_p = d$







$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

Perspective Projection

Perspective projection

$$x_p = \frac{x}{z/d}$$

$$y_p = \frac{y}{z/d}$$

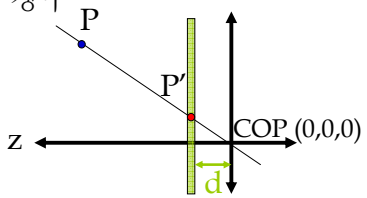
$$z_p = d$$

$$\mathbf{q} = \mathbf{M}\mathbf{p} \qquad \mathbf{M}_{pers} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Perspective Projection

- 투영면 (Projection plane, PP)이 투영중심 (Center of projection, COP) 의 앞에 있는 경우



$$\frac{x'}{d}=\frac{x}{z} \qquad x'=\frac{x}{z/d}$$

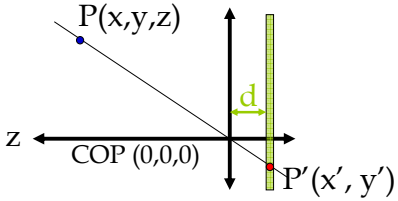
$$\frac{y'}{d}=\frac{y}{z} \qquad y'=\frac{y}{z/d}$$

$$z'=d \qquad z'=d$$

$$\mathbf{M}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1/d&0\end{bmatrix}$$

Perspective Projection

- 투영면 (Projection plane, PP)이 투영중심 (Center of projection, COP) 의 뒤에 있는 경우



$$-\frac{x'}{d}=\frac{x}{z} \qquad x'=-\frac{x}{z/d}$$

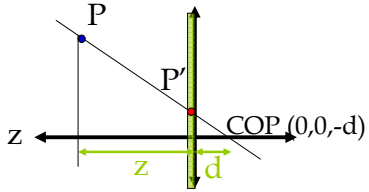
$$-\frac{y'}{d}=\frac{y}{z} \qquad y'=-\frac{y}{z/d}$$

$$z'=-d \qquad z'=-d$$

$$\mathbf{M}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&-1/d&0\end{bmatrix}$$

Perspective Projection

- 투영면 (Projection plane, PP)이 z = 0에 있고, 투영중심 (Center of projection, COP)이 z = -d에 있는 경우



$$\frac{x'}{d}=\frac{x}{z+d} \qquad x'=\frac{x}{(z+d)/d}$$

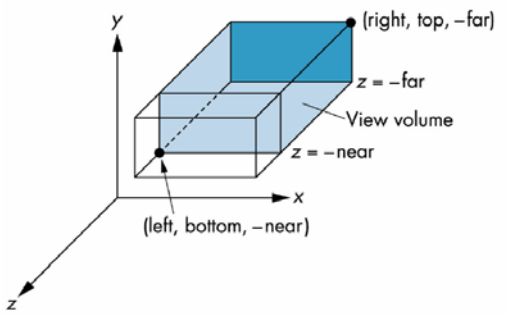
$$\frac{y'}{d}=\frac{y}{z+d} \qquad y'=\frac{y}{(z+d)/d}$$

$$z'=0 \qquad z'=0$$

$$\mathbf{M}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&1/d&1\end{bmatrix}$$

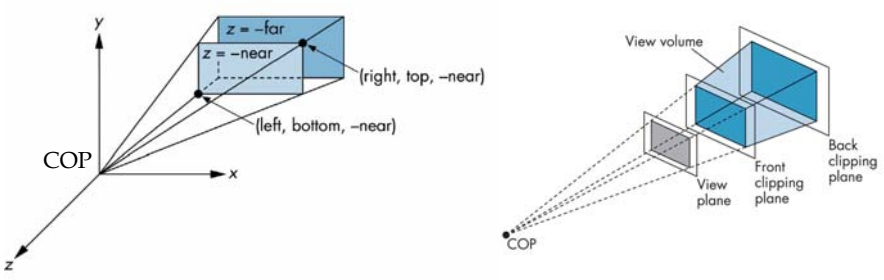
OpenGL Orthographic Projection

- glOrtho(left, right, bottom, top, near, far)
  - 이 함수의 매개변수는 glFrustum의 매개변수와 동일하다.
  - 관측공간은 직육면체이다.



OpenGL Perspective Projection

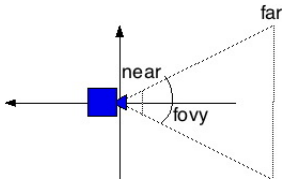
- OpenGL에서 투시투영 (perspective projection)은 바늘구멍 카메라 (pin-hole camera)가 원점(origin)에 위치하고 있으며 -Z축을 바라보고 있다.
- glFrustum(left, right, bottom, top, near, far)
  - 앞면과 뒷면의 거리는 양수이어야 하며, COP에서 평면까지의 거리로 측정된다.
  - 관측공간은 절두체 (frustum, i.e. truncated pyramid)이다.

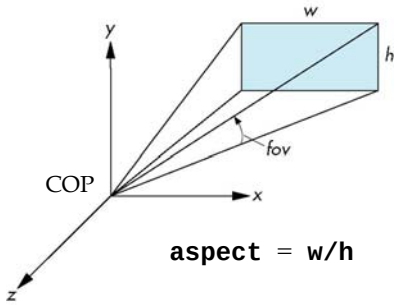


OpenGL Perspective Projection

- gluPerspective(fovy, aspect, near, far)
  - fovy – Y-축 방향에서의 시야 (field of view) 각도
  - aspect – 투영면의 (너비를 높이로 나눈) 종횡비 (aspect ratio)
  - near – 앞쪽 클리핑면
  - far – 뒤쪽 클리핑면

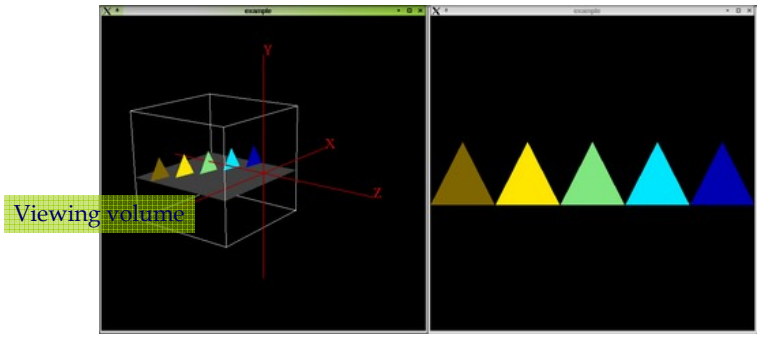
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45, 1.333, 0.1, 100);
glMatrixMode(GL_MODELVIEW);
```





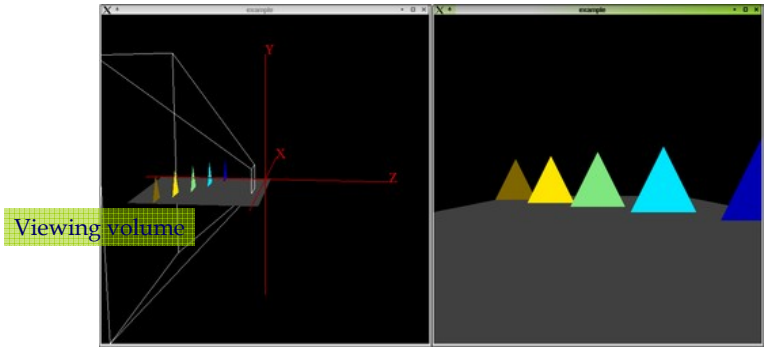
Orthographic Projection

- 직교 투영 (Orthographic projection)은 직육면체 (rectilinear box)의 관측공간을 화면에 투영한다.
- 객체의 크기가 거리에 따라 변하지 않는다.



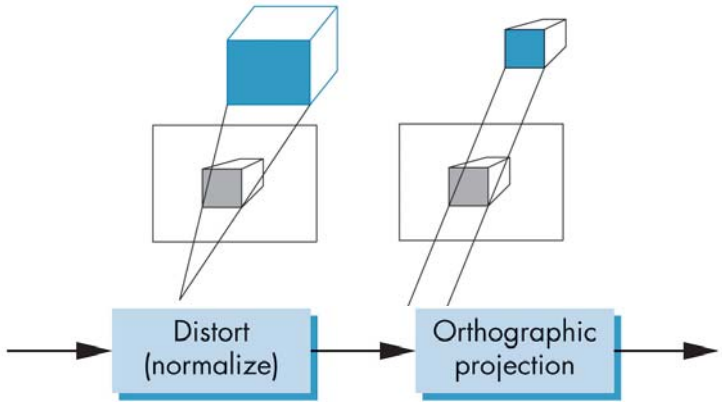
Perspective Projection

- 원근투영 (Perspective projection)은 절두체 (*frustum, i.e., truncated pyramid*) 관측공간을 화면에 투영한다.
- 가까운 객체는 크게 나타나고, 멀리 있는 객체는 작게 나타난다.



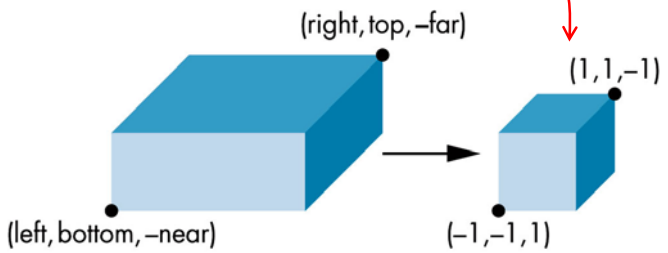
Projection Normalization

- 투영 정규화 (projection normalization)는 왜곡된 객체의 직교 투영이 원래 객체의 원하는 투영이 되도록, 객체들을 사전 왜곡시킴으로써, 모든 투영을 직교 투영으로 변환시키는 작업이다.



Orthogonal Projection Matrix

- 관측 공간(View volume)을 정규관측공간 (Canonical view volume)으로 매핑한다.



직육면체 -> 정육면체
직교투영 -> 직교투영

Orthogonal Projection Matrix

- 지정된 관측공간의 중심을 정규관측공간의 중심으로 이동

$$T\left(\frac{-(left+right)}{2}, \frac{-(top+bottom)}{2}, \frac{(far+near)}{2}\right)$$

- 지정된 관측공간의 변을 길이가 2가 되도록 크기 변환

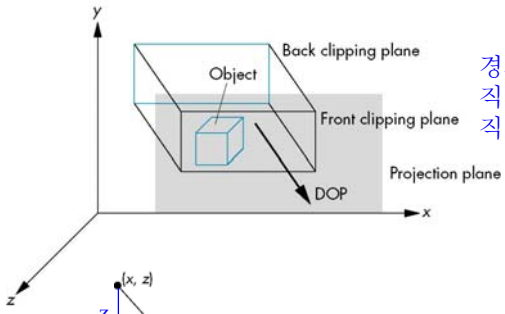
$$S\left(\frac{2}{(right-left)}, \frac{2}{(bottom-top)}, \frac{2}{(far-near)}\right)$$

- Projection matrix:

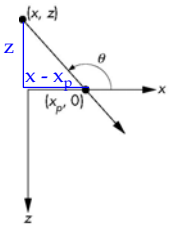
$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- General case: $\mathbf{P} = \mathbf{M}_{orth}\mathbf{ST}$

Oblique Projection Matrix



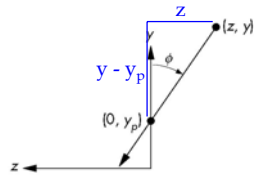
경사투영-> 직교투영
직육면체 -> 정육면체
직교투영 -> 직교투영



top view

$$\tan\theta = \frac{z}{x - x_p}$$

$$x_p = x - z\cot\theta$$



side view

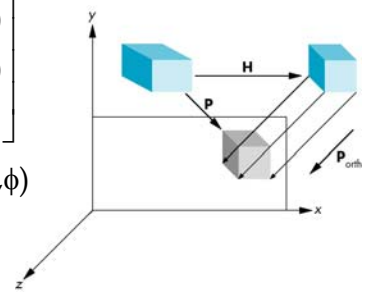
$$\tan\phi = \frac{z}{y - y_p},$$

$$y_p = y - z\cot\phi$$

Oblique Projection Matrix

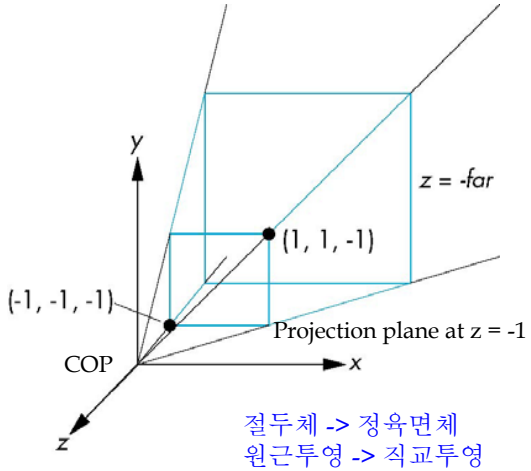
- *xy* shear (*z* values unchanged)

$$\mathbf{H}(\theta,\phi) = \begin{bmatrix} 1 & 0 & -\cot\theta & 0 \\ 0 & 1 & -\cot\varphi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- Projection matrix: $\mathbf{P} = \mathbf{M}_{orth}\,\mathbf{H}(\theta,\phi)$
- General case: $\mathbf{P} = \mathbf{M}_{orth}\,\mathbf{ST}\,\mathbf{H}(\theta,\phi)$

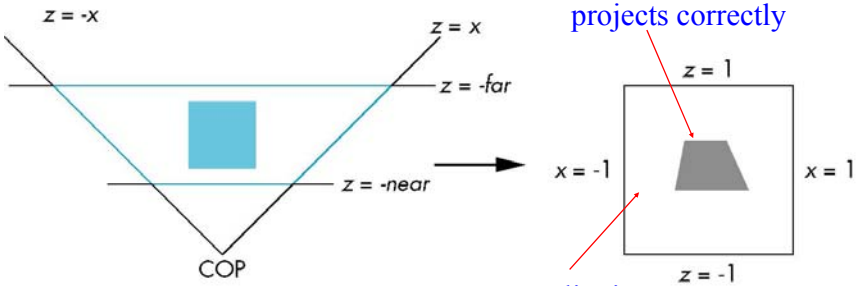
Perspective Projection Matrix



절두체 -> 정육면체
원근투영 -> 직교투영
한 점으로 모이던 투영선들이 평행해짐 - 원근감생성

Perspective Projection Matrix

- 원근 정규화 (Perspective normalization)



x = ±*z* → ±1
y = ±*z* → ±1
z = *near/far* → ±1

Perspective Projection Matrix

- 원근 정규화 (Perspective normalization)는 원근투영을 직교투영으로 바꾸는 작업이다.
  - 투영면(PP)이 z = -1, 투영중심(COP)이 원점인 원근투영 행렬, M

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

  - 관측공간의 측면이 투영면을 45도로 교차하도록 함으로써 시야를 90도로 고정

$x = \pm z$

$y = \pm z$

Perspective Projection Matrix

- N 행렬:

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

- p′=Np:

$$x' = x,\ \ y' = y,\ \ z' = \alpha z + \beta,\ \ w' = -z$$

- 원근 나눗셈 (Perspective division)한 후, p′->p′′:

$$\Rightarrow x'' = -\frac{x}{z},\ \ y'' = -\frac{y}{z},\ \ z'' = -\left(\alpha + \frac{\beta}{z}\right)$$

Perspective Projection Matrix

- x = ± z 이면, x′′ = ± 1
- y = ± z 이면, y′′ = ± 1
- Far plane z = z_{max} 이면,

$$z'' = -\left(\alpha + \frac{\beta}{z_{max}}\right)$$

- Near plane z = z_{min} 이면,

$$z'' = -\left(\alpha + \frac{\beta}{z_{min}}\right)$$

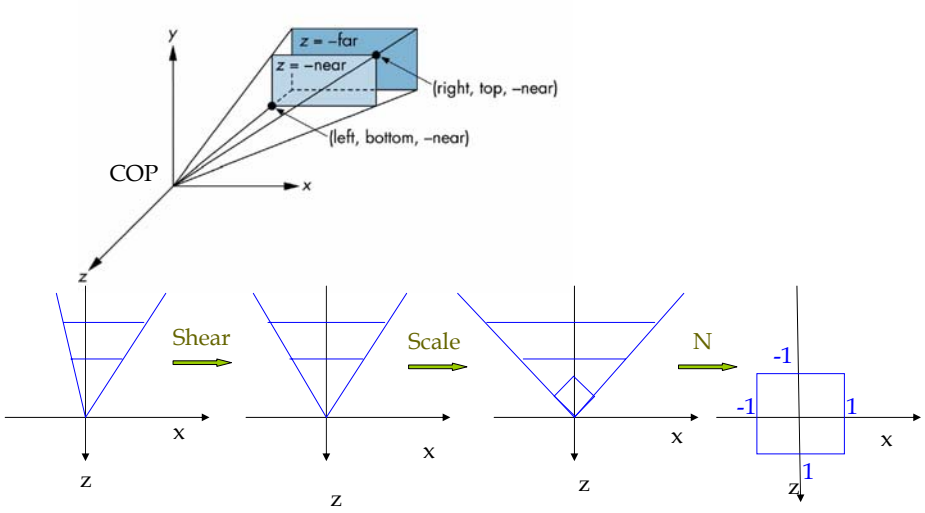
- z_{min} -> -1 그리고 z_{max} -> 1 매핑하도록 α와 β를 선정

$$\alpha = \frac{z_{max} + z_{min}}{z_{max} - z_{min}}$$

$$\beta = \frac{2 z_{max} z_{min}}{z_{max} - z_{min}}$$

OpenGL Perspective Projection

- glFrustum(left, right, bottom, top, near, far)



OpenGL Perspective Projection

- Shear

$$H(\cot\theta, \cot\phi) = H\left(\frac{right + left}{2 z_{min}}, \frac{top + bottom}{2 z_{min}}\right)$$

- Then,

$$x = \pm\frac{right - left}{2 z_{min}},\ y = \pm\frac{top - bottom}{2 z_{min}},\ z = z_{max},\ z = z_{min}$$

- Scale $x = \pm z,\ y = \pm z$

- $\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{bmatrix}$

$$\alpha = \frac{z_{max} + z_{min}}{z_{max} - z_{min}}$$

$$\beta = \frac{2 z_{max} z_{min}}{z_{max} - z_{min}}$$

OpenGL Perspective Projection

$$\mathbf{P} = \mathbf{NSH} = \begin{bmatrix} \frac{2z_{min}}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2z_{min}}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{z_{max}+z_{min}}{z_{max}-z_{min}} & \frac{2z_{max}z_{min}}{z_{max}-z_{min}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

z_{max} = far
z_{min} = near

Camera Movement

- The OpenGL camera is always positioned at the origin, looking down the –Z axis.
- The camera itself never moves.
- Looking at the final image produced, moving a camera is equivalent to moving the world in the opposite direction.
- E.g. moving everything to the right will produce the same image as moving the camera to the left.

Camera Movement

- To give the effect of moving the perspective viewpoint in OpenGL, use a transformation at the beginning of the frame, which affects all objects.
- For example, to move the camera 10 units from the origin in the +Z direction, translate the world by -10 in Z

```
void display()
{
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(45, 1, 0.1, 100);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glTranslatef(0, 0, -10);
   drawObjects();
}
```

Camera Movement

- For general camera movement
  - Keep track of camera position & orientation
  - Apply inverse transformation to the world
- Example

```
float cameraX, cameraY, cameraZ, cameraHeading;
void display()
{
   glLoadIdentity();
   glRotatef(-cameraHeading, 0, 1, 0);
   glTranslatef(-cameraX, -cameraY, -cameraZ);

   drawObjects();
}
```