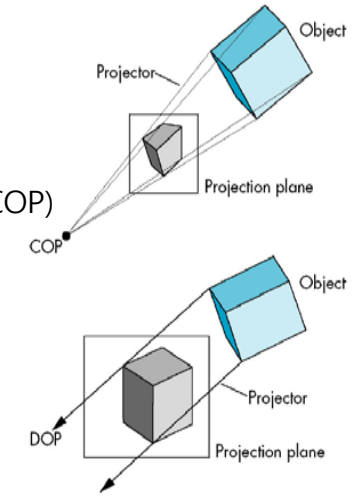
Viewing

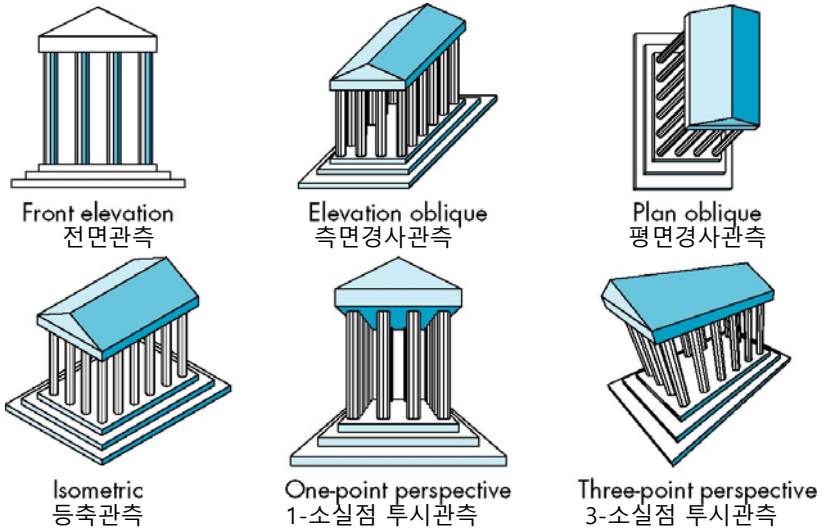
514780
2019년 가을학기
11/4/2019
단국대학교 박경신

Viewing

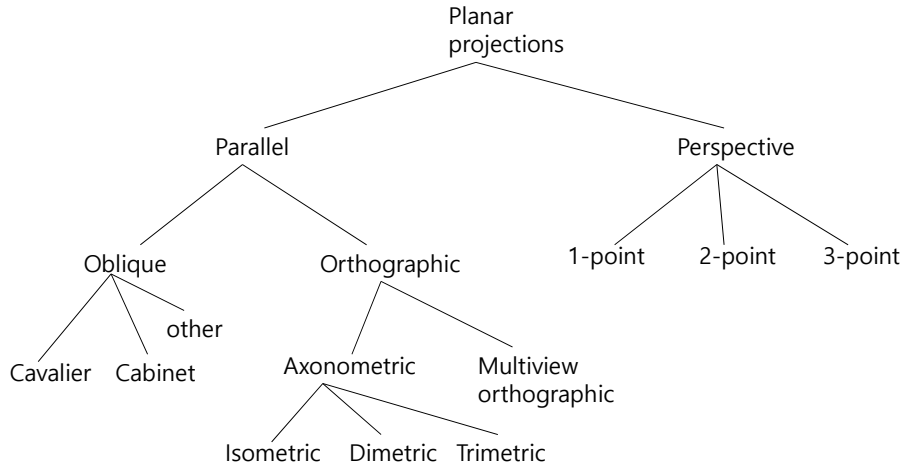
- 관측의 기본요소
  - 객체 (Objects)
  - 관측자 (Viewer)
  - 투영선 (Projector)
  - 투영면 (Projection plane)
- 투영중심 (Center of Projection: COP)
  - COP가 유한한 경우 –
    투시관측 (Perspective views)
  - COP가 무한한 경우 –
    평행관측 (Parallel views)



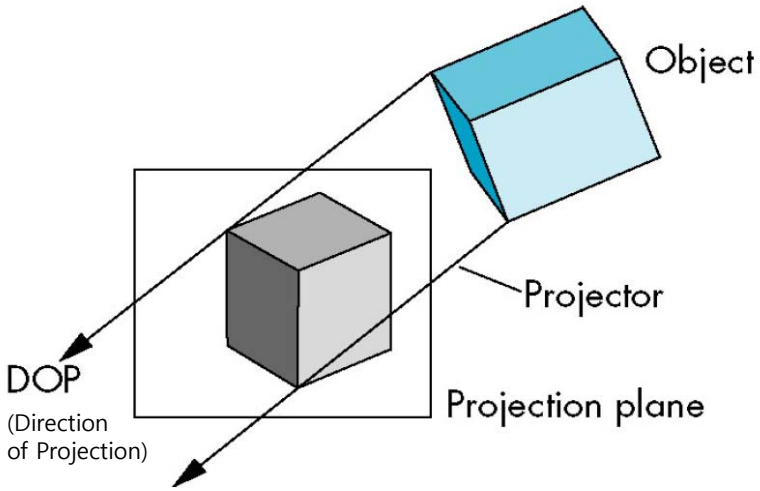
Classical Viewing



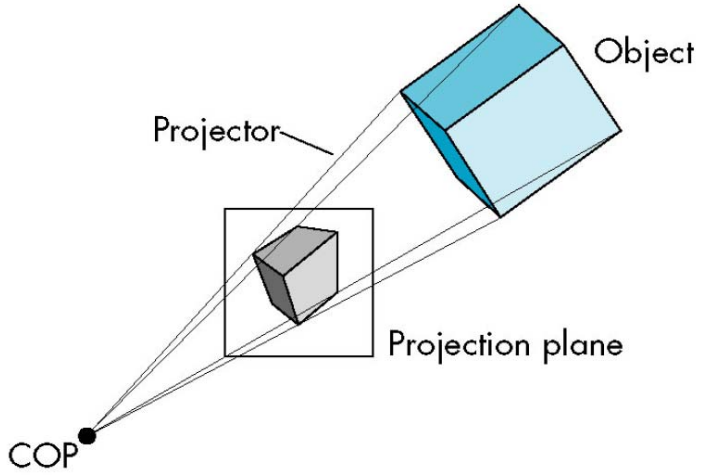
Classical Viewing



Parallel Viewing

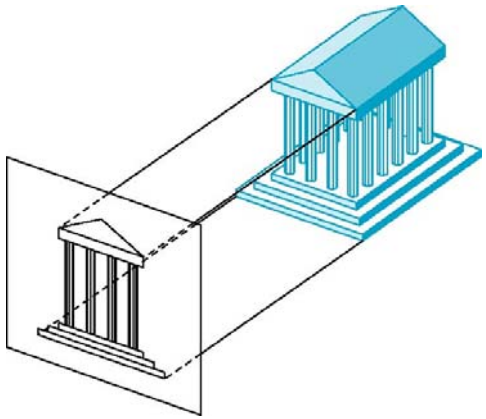


Perspective Viewing



Orthographic Projection

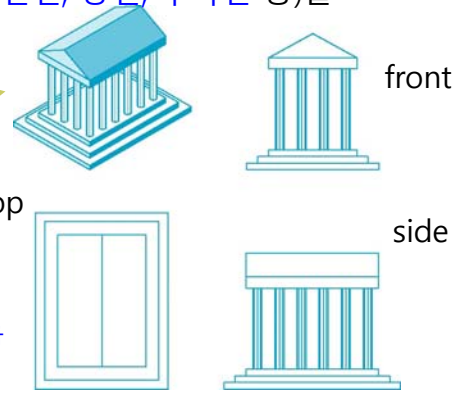
- 직교 투영 (Orthographic Projection)에서는 투영선 (Projector)은 투영면 (Projection plane)에 수직이다.



Multiview Orthographic Projection

- 다중 관측 직교 투영 (Multiview Orthographic Projection) 에서는 여러 개의 투영면을 만드는데, 각각은 객체의 주면 (principal face) 중 하나와 평행하다.
- 일반적으로 세 개의 관측 (전면, 상면, 우측면 등)을 표시한다.

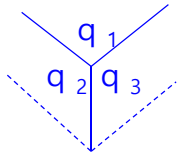
Isometric (not multiview orthographic view)

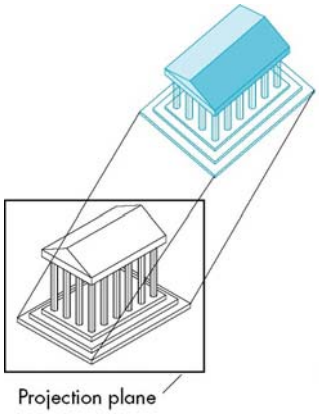


- 거리와 각이 모두 보존됨
- 또한, 거리와 모양의 왜곡이 없음
- 그러나, 다중관측직교투영으로부터 객체가 어떻게 생겼는지 추측이 어려움
- 그래서, Isometric view과 함께 제공함

Axonometric Projections

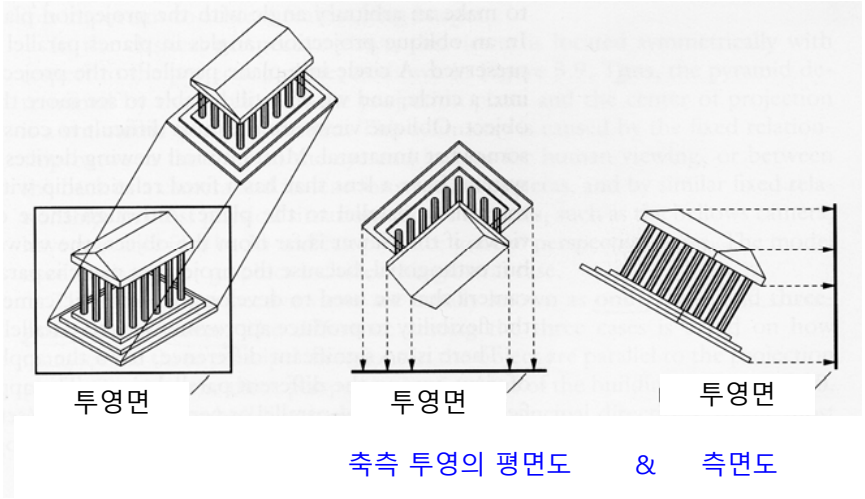
- 축측 관측 (Axonometric view)에서 투영선은 투영면에 수직이지만, 투영면은 객체에 대해 어떠한 방향에도 존재할 수 있다.





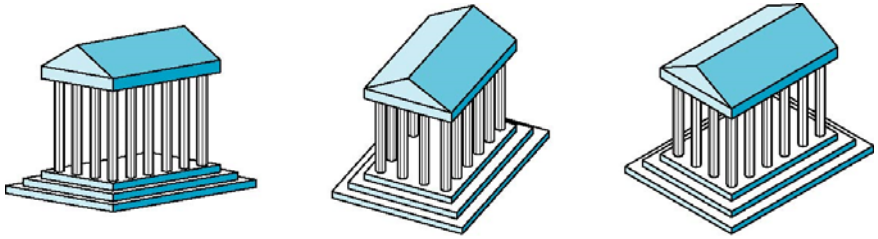
▪등축투영 (isometric) - 만일 투영면이 사각형 객체의 모서리에서 만나는 세 개의 주면에 대해서 대칭으로 놓여짐
▪이축투영(dimetric) – 투영면이 두 개의 주면이 대칭되도록 놓여짐
▪삼축투영(trimetric) – 일반적인 경우

Construction of an Axonometric Projection



축측 투영의 평면도 & 측면도

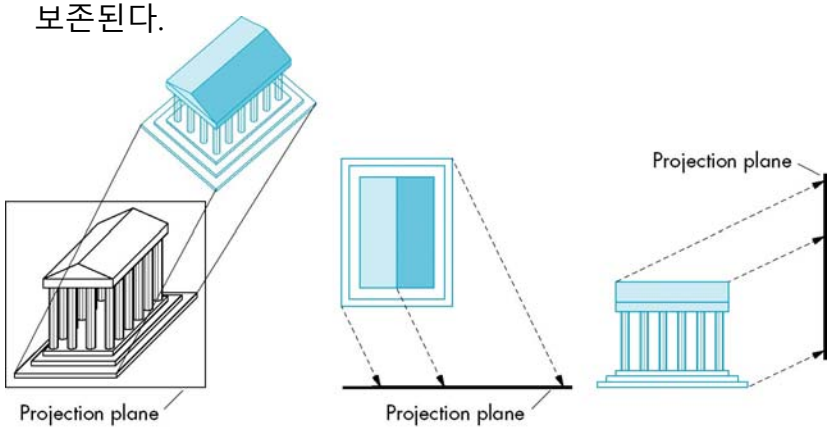
Types of Axonometric Projections



▪선들의 평행은 이미지 안에서 보존되지만 각은 보존되지 않는다.
▪등축 투영: 이미지공간에서 선분의 길이는 객체 공간에서 측정된 길이보다 짧다. 이러한 거리의 단축 (foreshortening)은 세 개의 주면에서 똑같이 발생한다. 따라서 거리 비교는 가능하다.
▪이축 투영: 두 개의 다른 단축비를 가진다.
▪삼축 투영: 세 개의 다른 단축비를 가진다.

Oblique Projection

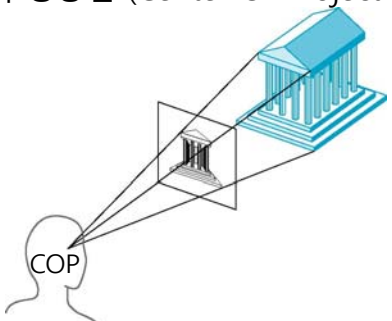
- 경사투영 (Oblique Projection)에서 투영선은 투영면과 임의의 각을 가질 수 있다. 투영면에 평행한 면내의 각은 보존된다.



경사 투영의 평면도 & 측면도

Perspective Projection

- 원근(투시) 관측 (Perspective Projection)은 객체가 관측자로부터 멀리 떨어질 수록 크기가 축소된다.
- 투영선은 투영중심 (Center Of Projection)으로 모인다.



- 원근(투시) 관측은 크기가 축소 (diminution)되는 특징을 가진다.
- 이러한 크기변화는 자연스러운 모습의 관측(realistic view)을 얻게 한다.
- 그러나, 선분의 길이가 얼마나 짧아지는 가는 그 선분이 관측자로부터 얼마나 떨어져 있는가에 의존하기 때문에, 길이 측정을 할 수 없다.

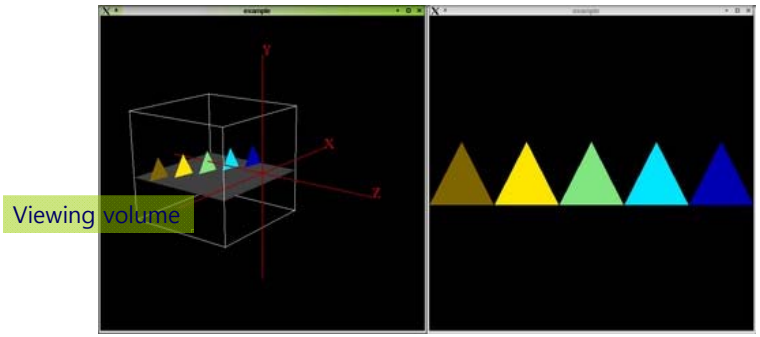
1-,2-,3-Point Perspective

- 일점, 이점, 삼점 투시 (one, two, three point perspectives) 관측의 차이는 객체의 세가지 주 방향 가운데 얼마나 많은 방향이 투영면에 평행한 가에 있다.
- 삼점 투시의 경우 세 개의 주 방향에 평행한 모든 직선들은 세 개의 소실점 (vanishing point)에서 만난다.

3점 투시 2점 투시 1점 투시

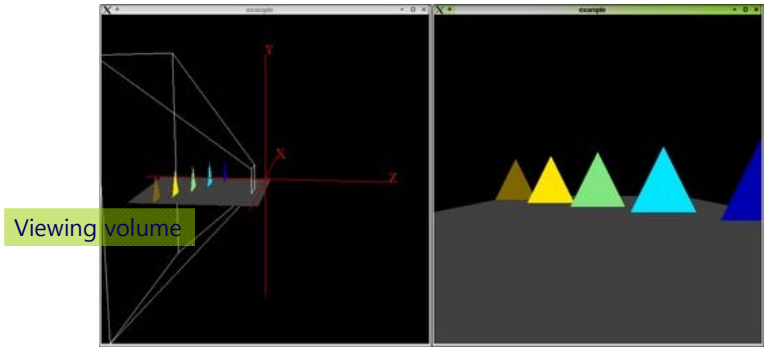
Orthographic Projection

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



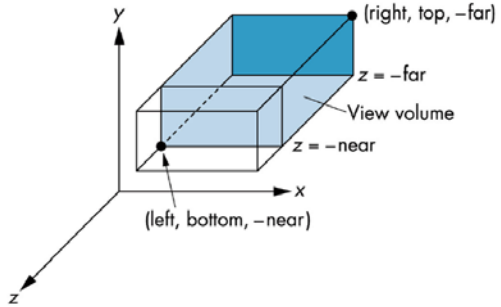
Perspective Projection

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



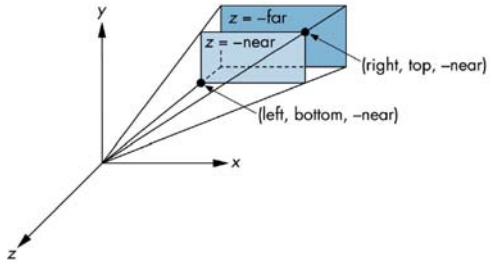
OpenGL Orthographic Projection

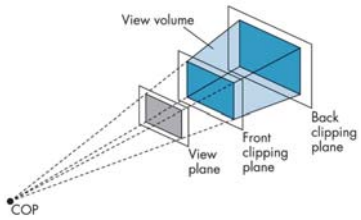
- **glm::ortho(left, right, bottom, top, near, far)**
  - 이 함수의 매개변수는 glFrustum의 매개변수와 동일하다.
  - 관측공간은 직육면체이다.
  - near와 far는 양수만 받는다. 내부에서 음수로 바꿔서 사용한다.



OpenGL Perspective Projection

- OpenGL에서 투시투영 (perspective projection)은 카메라가 원점(origin)에 위치하고 있으며 –Z축을 바라보고 있다.
- **glm::frustum(left, right, bottom, top, near, far)**
  - 앞면과 뒷면의 거리는 양수이어야 하며, COP에서 평면까지의 거리로 측정된다.
  - 관측공간은 절두체 (frustum, i.e. truncated pyramid)이다.

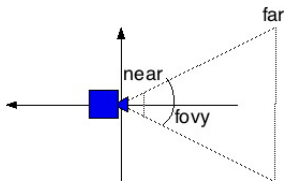


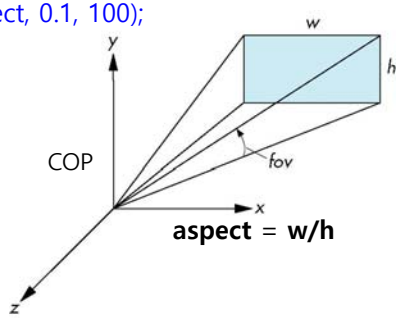


OpenGL Perspective Projection

- **glm::perspective(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

Projection = glm::perspective(45, aspect, 0.1, 100);



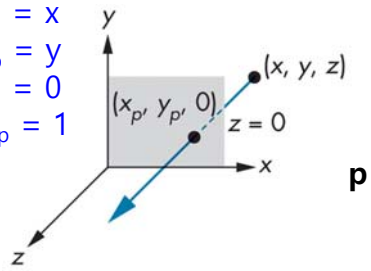


Orthographic Projection

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

Orthographic projection

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

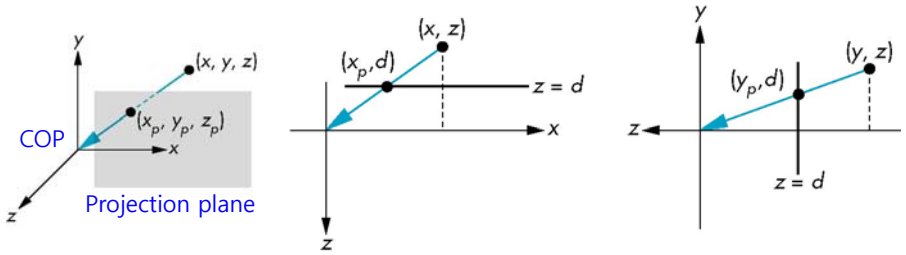


$$\mathbf{M}_{ortho} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{q}=\mathbf{Mp}$$

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \mathbf{M}_{ortho} \Rightarrow \quad \mathbf{q} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

Perspective Projection

- 원근/투시 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

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

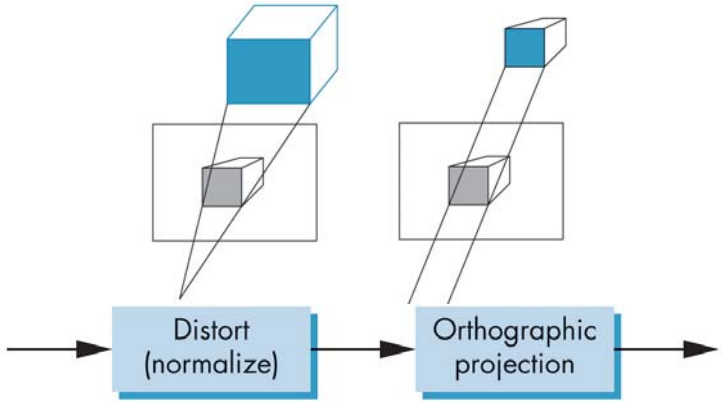
q = **Mp**

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

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

Projection Normalization

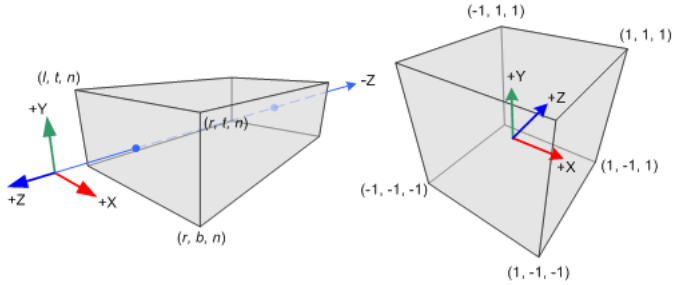
- 투영 정규화 (projection normalization)는 왜곡된 객체의 직교 투영이 원래 객체의 원하는 투영이 되도록, 객체들을 사전 왜곡시킴으로써, 모든 투영을 직교 투영으로 변환시키는 작업이다.



Orthogonal Projection Matrix

- Orthogonal projection은 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: $\mathbf{P} = \mathbf{M}_{ortho}\mathbf{ST}$

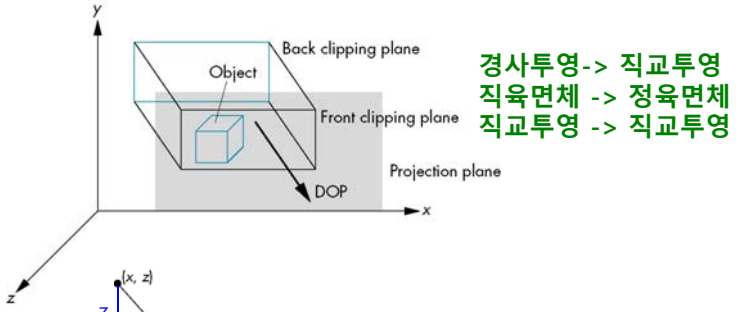
Orthogonal Projection Matrix

ortho=ST

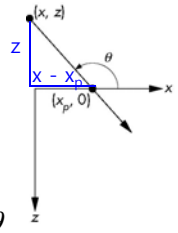
$$= \begin{pmatrix} \frac{2}{(right-left)} & 0 & 0 & 0 \\ 0 & \frac{2}{(top-bottom)} & 0 & 0 \\ 0 & 0 & -\frac{2}{(far-near)} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\frac{(right+\text{left})}{2} \\ 0 & 1 & 0 & -\frac{(top+bottom)}{2} \\ 0 & 0 & 1 & \frac{(far+near)}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{2}{(right-left)} & 0 & 0 & -\frac{(right+left)}{(right-left)} \\ 0 & \frac{2}{(top-bottom)} & 0 & -\frac{(top+bottom)}{(top-bottom)} \\ 0 & 0 & -\frac{2}{(far-near)} & -\frac{(far+near)}{(far-near)} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Oblique Projection Matrix



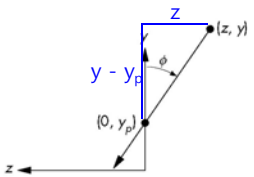
경사투영-> 직교투영
직육면체 -> 정육면체
직교투영 -> 직교투영



top view

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

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



side view

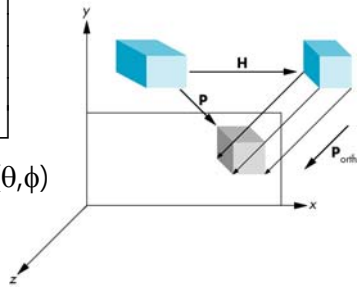
$$\tan\phi = \frac{z}{y - y_p},$$

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

Oblique Projection Matrix

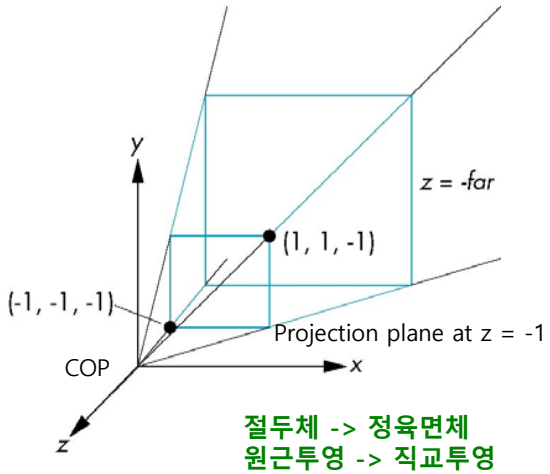
- *xy* shear (*z* values unchanged)

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



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

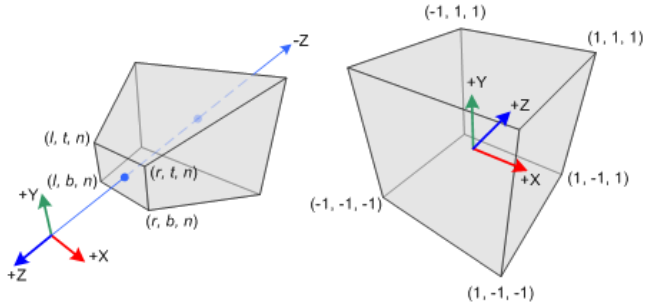
Perspective Projection Matrix



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

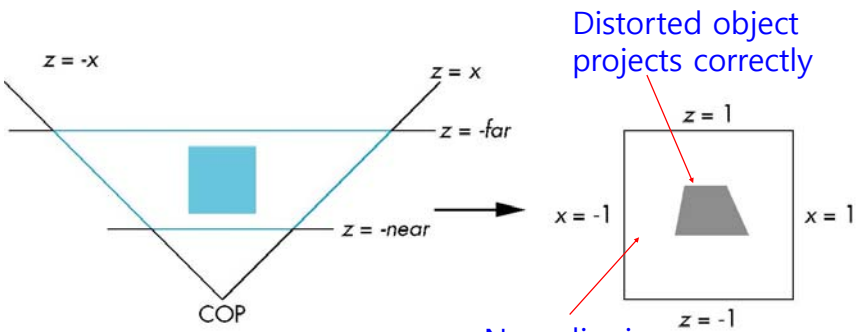
Perspective Projection Matrix

- Perspective projection은 관측 공간(View volume)을 정규관측공간 (Canonical view volume)으로 매핑한다.
  [l, r] => [-1, 1], [b, t] => [-1, 1], [-n, -f] => [-1, 1]
  [n, f] => [1, -1]



Perspective Projection Matrix

- 원근 정규화 (Perspective normalization)



$x = \pm z \rightarrow \pm 1$
$y = \pm z \rightarrow \pm 1$
$z = near/far \rightarrow \pm 1$

Perspective Projection Matrix

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

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

  - 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″:

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

Perspective Projection Matrix

- x = ± z 이면, x″ = ± 1
- y = ± z 이면, y″ = ± 1
- Far plane z = -far이면, $z'' = \dfrac{\alpha(-far) + \beta}{far} = 1$

  Near plane z = -near이면, $z'' = \dfrac{\alpha(-near) + \beta}{near} = -1$

- z″ -> ± 1 매핑하도록 α와 β를 선정: (-near, -1) & (-far, 1)

$$\alpha = -\frac{far + near}{far - near}$$

$$\beta = -\frac{2\,far\ \ near}{far - near}$$

$\alpha(-far) + \beta = far$ & $\alpha(-near) + \beta = -near$

$\beta = -near + \alpha near$

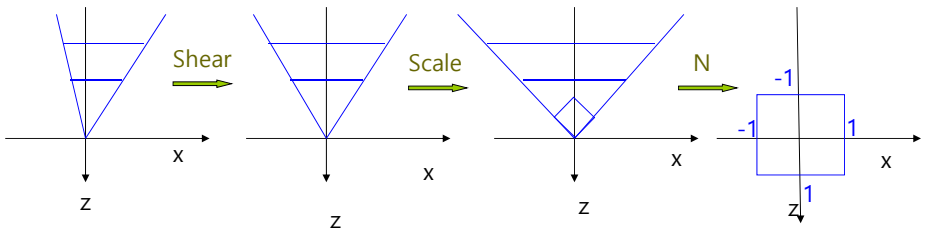
$\alpha(-far) + (-near + \alpha near) = far$

$\alpha(near - far) = near + far$

$\alpha = \dfrac{near + far}{near - far} = -\dfrac{far + near}{far - near}$

OpenGL Perspective Projection

- glm::frustum(left, right, bottom, top, near, far)



OpenGL Perspective Projection

- Shear $H(\cot\theta, \cot\phi) = H\left(\dfrac{right + left}{-2near}, \dfrac{top + bottom}{-2near}\right)$
- Then, $x = \pm\dfrac{right - left}{-2near}, y = \pm\dfrac{top - bottom}{-2near}, z = -near, z = -far$
- Scale $S = S\left(\dfrac{-2near}{right - left}, \dfrac{-2near}{top - bottom}, 1\right)$
- Then, $x = \pm z, y = \pm z$
- Normalize

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

$$\alpha = -\frac{far + near}{far - near}$$

$$\beta = -\frac{2far\ near}{far - near}$$

OpenGL Perspective Projection Matrix

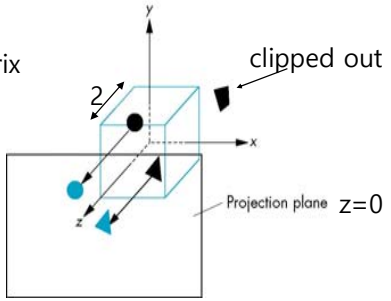
frustum=NSH

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{(far+near)}{(far-near)} & \frac{(far+near)}{2} \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \frac{2near}{(right-left)} & 0 & 0 & 0 \\ 0 & \frac{2near}{(top-bottom)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \frac{(right+\text{left})}{2near} & 0 \\ 0 & 1 & \frac{(top+bottom)}{2near} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{2near}{(right-left)} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2near}{(top-bottom)} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{(far+near)}{(far-near)} & -\frac{2far\,near}{(far-near)} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Computer Viewing

- 컴퓨터에서의 관측은 다음과 같이 구성된다.
  - 카메라의 위치와 방향을 잡아준다.
    - Model-view transformation matrix
  - 투영변환을 적용한다.
    - Projection transformation matrix
  - 클리핑 (Clipping) 한다.
    - View volume
- OpenGL에서 초기 카메라는
  - 객체 프레임의 원점에 놓이고,
  - z-축의 음수방향을 향한다.
  - 직교관측으로 설정되어 있고,
  - 원점을 중심으로 한 각 변의 길이가 2인 정육면체로 된 관측 공간을 가진다.
  - 기본 투영면은 z=0인 면이고, 투영방향은 z축과 나란하다.



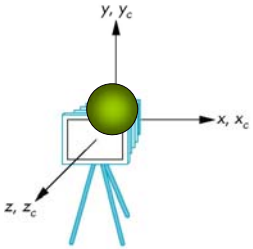
Positioning the Camera Frame

- 모델-관측 변환행렬 (model-view transformation matrix) 방법
- 관측 참조점(VRP), 관측면 법선(VPN), 관측 상향벡터(VUP)를 이용한 관측-방향 행렬 (view-orientation matrix) 방법
- Look-at 함수

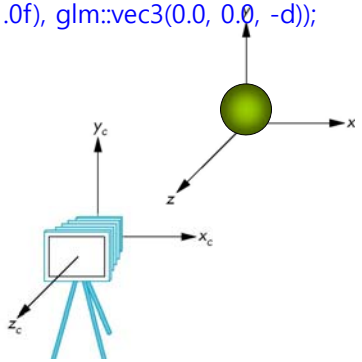
Positioning the Camera Frame

- OpenGL에서 카메라 위치 지정 방법
  - 카메라를 원점으로부터 뒤로 이동시키는 방법
  - 또는 물체를 카메라의 앞으로 이동시키는 방법을 사용한다.

World = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, 0.0, -d));



World frame = Camera frame

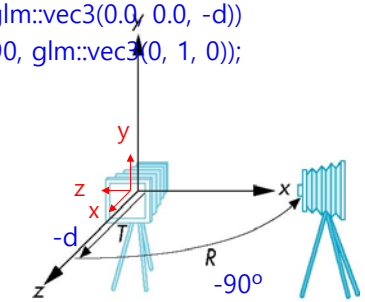


Moving the camera frame after translation by –d, d > 0

Positioning the Camera

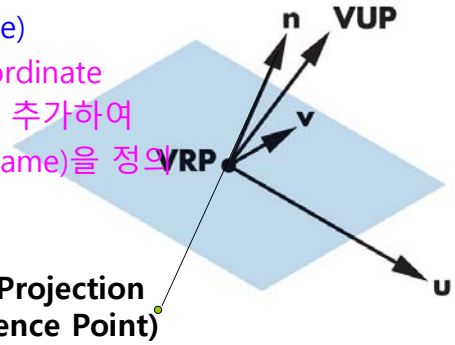
- 연속된 회전 (rotation)과 이동 (translation)으로 카메라 위치를 지정할 수 있다.
- 예제: x축 방면에서 바라보는 관측
  - R = 카메라를 y축을 중심으로 회전
  - T = 카메라를 원점에서 먼 곳으로 이동
  - Model-view matrix C = TR

```
World = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, 0.0, -d))
        * glm::rotate(glm::mat4(1.0f), -90, glm::vec3(0, 1, 0));
```



Camera Frame

- 관측 참조점 (View reference point, VRP)
- 관측면 법선 (View plane normal, VPN) n = PRP - VRP
- 관측 상향벡터 (View-up vector, VUP)
- 측면벡터 (Side vector) u = VUP x n
- 상향벡터 (Up vector) v = n x u
- u, v, n를 정규화(normalize)
- 관측 좌표계 (viewing coordinate system, u′-v′-n′)와 VRP를 추가하여 카메라 프레임 (camera frame)을 정의



Camera Frame

- View-orientation matrix, M

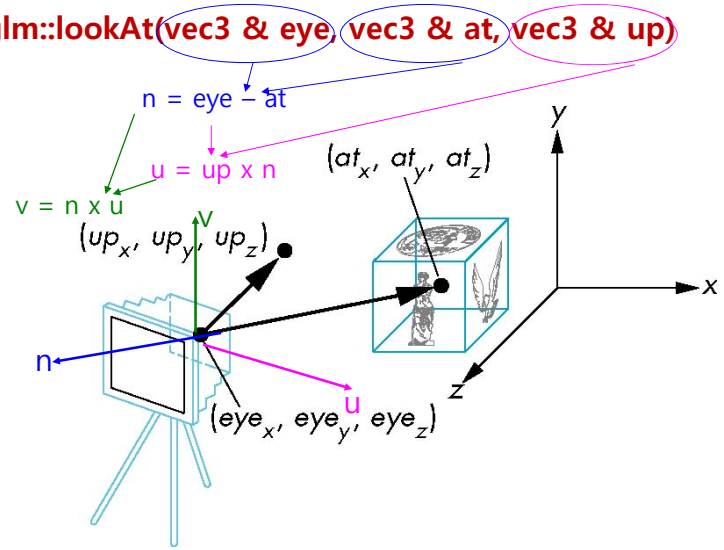
$$\mathbf{M} = \begin{bmatrix} u'_x & v'_x & n'_x & 0 \\ u'_y & v'_y & n'_y & 0 \\ u'_z & v'_z & n'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation matrix, $M^{-1} = M^T = R$
- World frame에서 카메라의 위치 지정: V = RT

$$\begin{bmatrix} u'_x & u'_y & u'_z & 0 \\ v'_x & v'_y & v'_z & 0 \\ n'_x & n'_y & n'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u'_x & u'_y & u'_z & -e \bullet u' \\ v'_x & v'_y & v'_z & -e \bullet v' \\ n'_x & n'_y & n'_z & -e \bullet n' \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

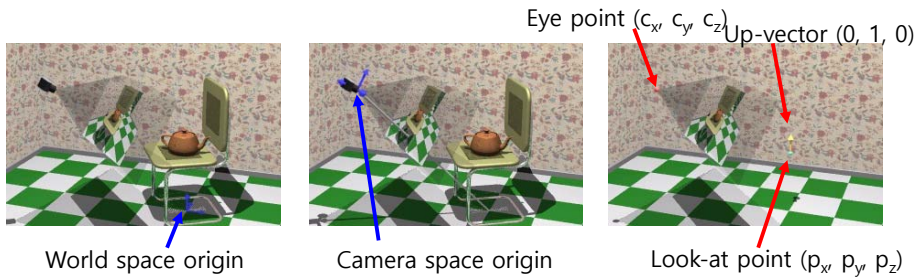
lookAt

- **glm::lookAt(vec3 & eye, vec3 & at, vec3 & up)**



lookAt

- *Eye Point* : 카메라의 원점 (월드 좌표계)
- *Look-At* : 카메라가 쳐다보고 있는 위치 (카메라 이미지의 중심이 되는 위치)
- *Up-Vector* : 월드 좌표계에서 카메라가 보는 up 벡터 (카메라 이미지에서 어디로 향하는지에 대한 방향벡터)

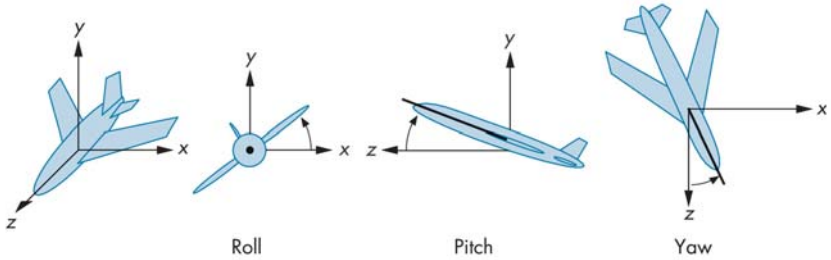


gluLookAt

```
void gluLookAt(GLdouble ex, GLdouble ey, GLdouble ez, GLdouble ax, GLdouble ay, GLdouble az,
    GLdouble ux, GLdouble uy, GLdouble uz) {
    GLdouble M[16]; GLdouble u[3], v[3], n[3]; GLdouble mag;

    n[0] = ex – ax; n[1] = ey – ay; n[2] = ez – az;          // n (camera frame Z)
    mag = sqrt(n[0]*n[0] + n[1]*n[1] + n[2]*n[2]);
    if (mag) { n[0] /= mag; n[1] /= mag; n[2] /= mag; }

    v[0] = ux; v[1] = uy; v[2] = uz;                          // u (camera frame X)
    u[0] = v[1]*n[2] – v[2]*n[1]; u[1] = -v[0]*n[2] + v[2]*n[0]; u[2] = v[0]*n[1] - v[1]*n[0];
    mag = sqrt(u[0]*u[0] + u[1]*u[1] + u[2]*u[2]);
    if (mag) { u[0] /= mag; u[1] /= mag; u[2] /= mag; }

    v[0] = n[1]*u[2] – n[2]*u[1]; v[1] = -n[0]*u[2] + n[2]*u[0]; v[2] = n[0]*u[1] - n[1]*u[0];  // v (camera
    frame Y)
    mag = sqrt(v[0]*v[0] + v[1]*v[1] + v[2]*v[2]);
    if (mag) { v[0] /= mag; v[1] /= mag; v[2] /= mag; }

    M[0] = u[0];  M[4] = u[1];  M[8]  = u[2];   M[12] = 0.0;          // R
    M[1] = v[0];  M[5] = v[1];  M[9]  = v[2];   M[13] = 0.0;
    M[2] = n[0];  M[6] = n[1];  M[10] = n[2];   M[14] = 0.0;
    M[3] = 0.0;   M[7] = 0.0;   M[11] = 0.0;    M[15] = 1.0;
    glMultMatrix(M);

    glTranslated(-ex, -ey, -ez);                              // RT
}
```

Yaw, Pitch, Roll

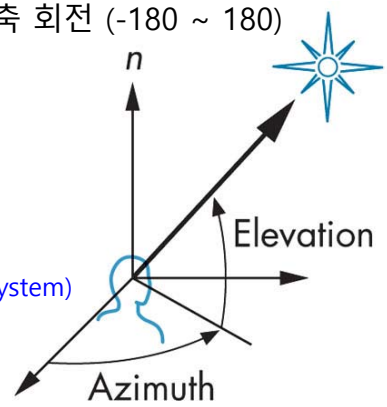
- 편요 (Yaw) – Y축 회전
- 종전 (Pitch) – X축 회전
- 횡전 (Roll) – Z축 회전



Elevation and Azimuth

- 방위각 (Azimuth) – X축 회전 (-180 ~ 180)
- 앙각 (Elevation) – Y축 회전 (-90 ~ 90)
- 꼬임각 (Twist angle) – Z축 회전 (-180 ~ 180)

3차원 극좌표계
(Spherical Polar Coordinates System)



Reference

- 원근 정규화
  http://www.songho.ca/opengl/gl_projectionmatrix.html