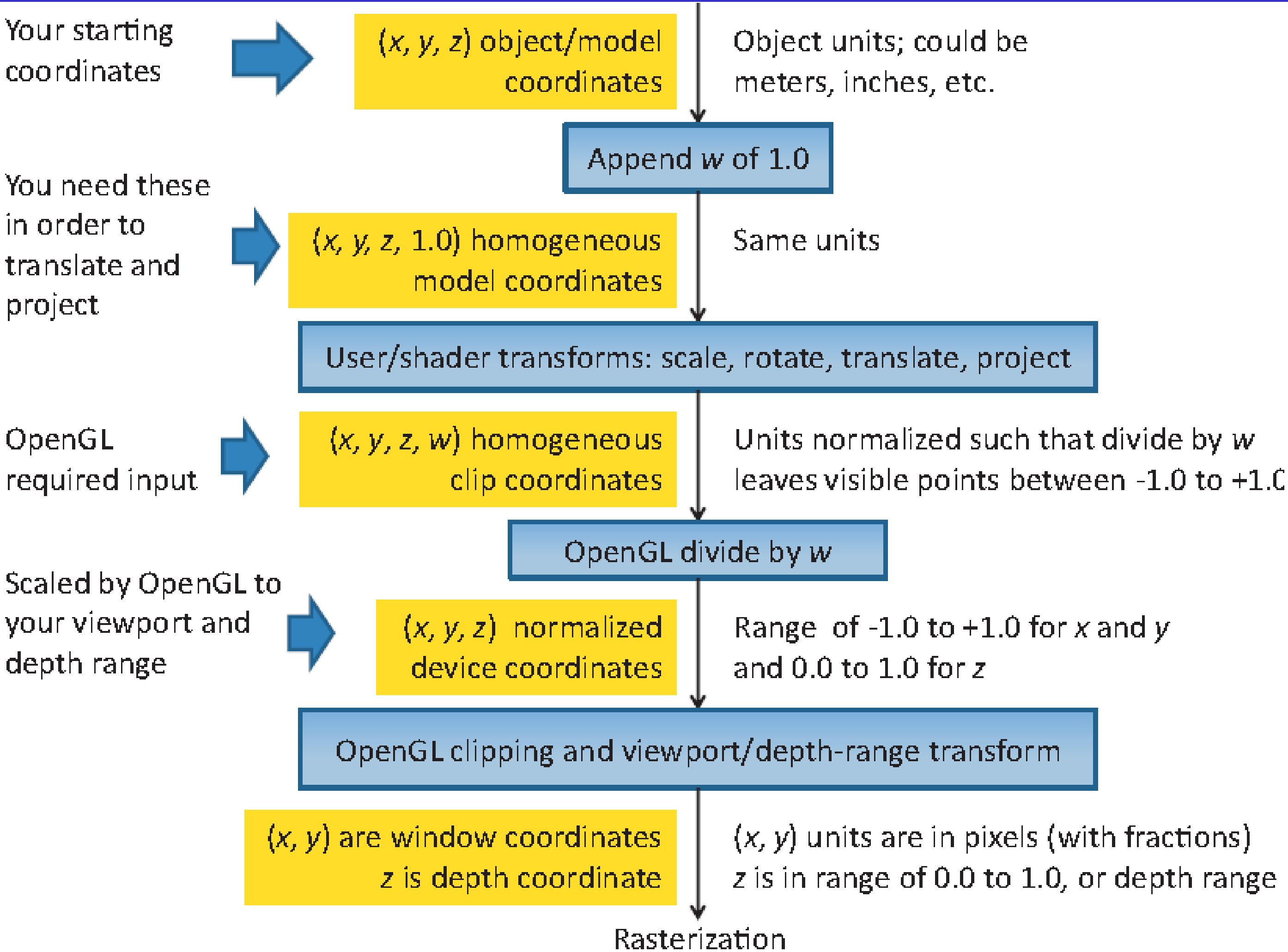
Topics

Perspective Projection

Coordinate Systems in OpenGL



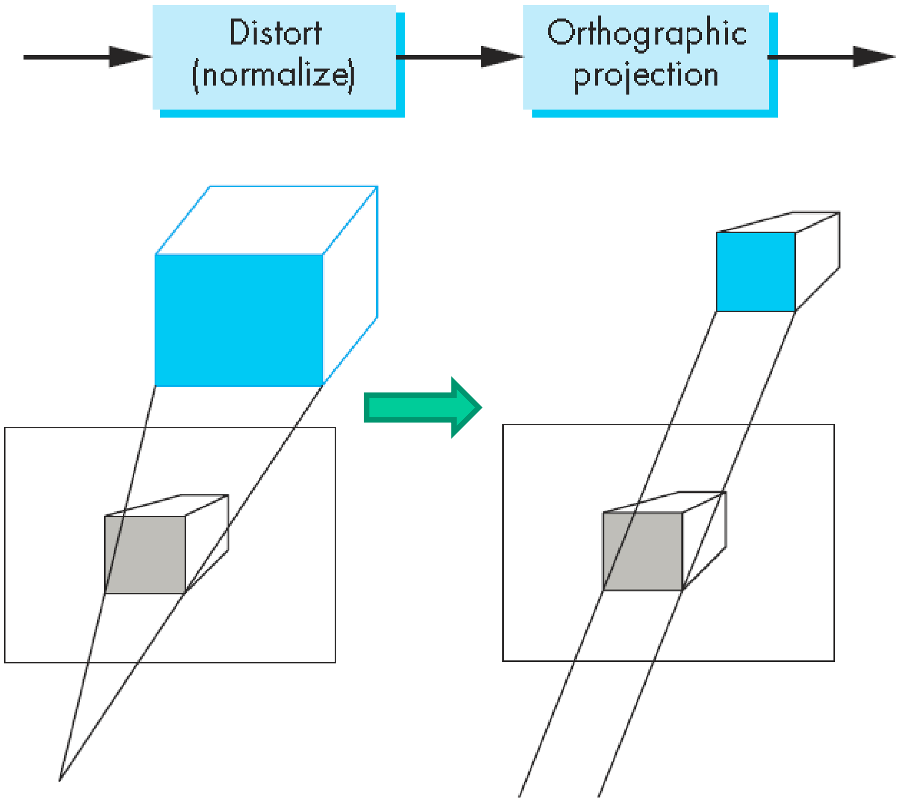
Shreiner et al.

Projections and View Normalization

The default projection is **orthogonal (orthographic) projection**

Most graphics systems use ***view normalization***

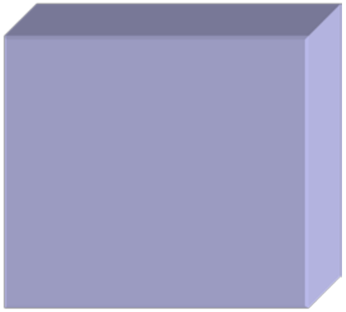
- All other views are converted to the orthographic view by distorting the objects -- ***normalization***
- Allows use of the same pipeline for all views



E. Angel and D. Shreiner

Oblique Projections

The OpenGL projection functions cannot produce general parallel projections – the oblique projection

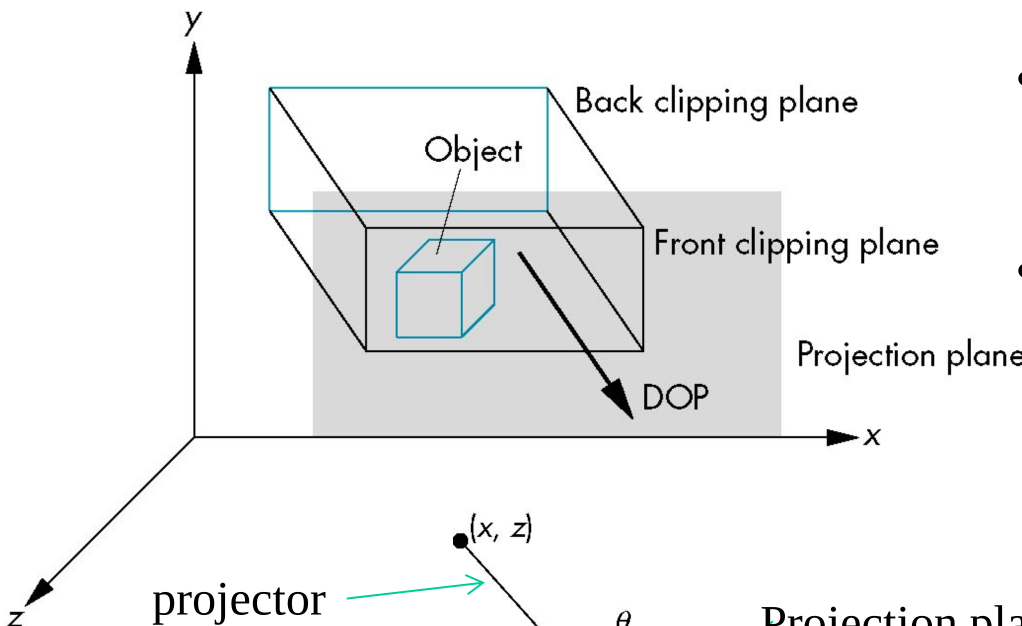


E. Angel and D. Shreine

It seems the cube has been sheared

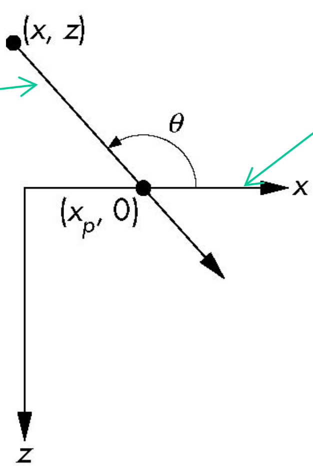
Oblique Projection = Shear + Orthogonal Projection

General Shear

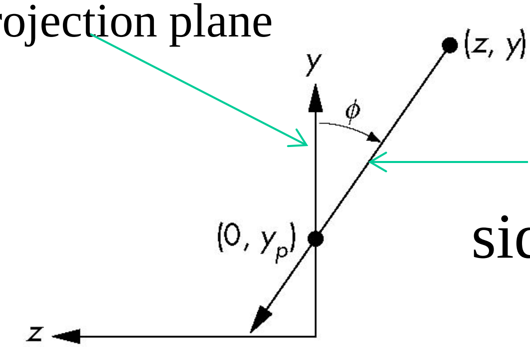


- The far and near clipping planes are parallel to the view plane
- The other four planes are parallel to the projection direction

projector



Projection plane



projector

top view

side view

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

$$\Rightarrow \quad x_p = x + z\cot\theta$$

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

E. Angel and D. Shreiner

Shear 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

$$\mathbf{P} = \mathbf{M}_{\mathrm{orth}}\,\mathbf{H}(\theta,\phi)$$

Shear Matrix

General case:

$$\mathbf{P} = \mathbf{M}_{\mathrm{orth}}\ \mathbf{STH}(\theta,\phi)$$

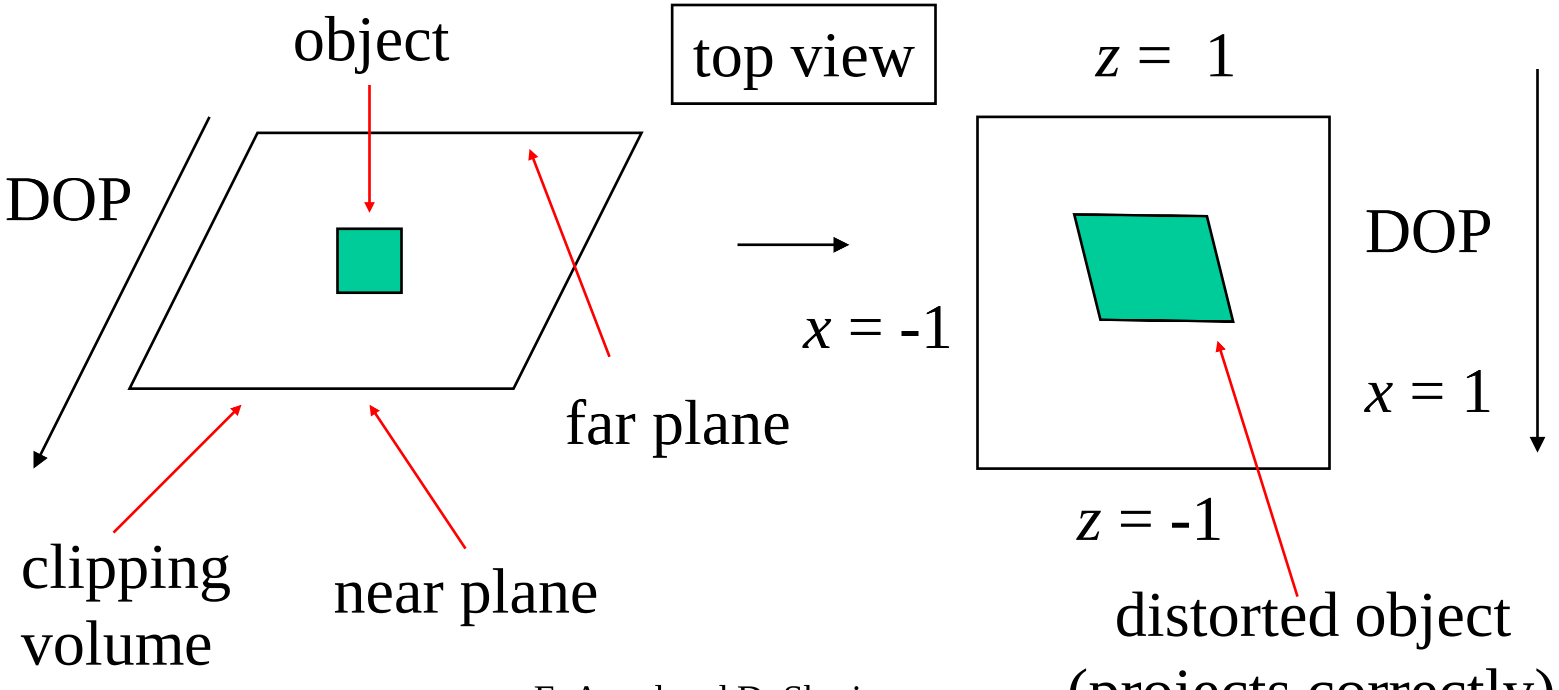
$$\mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{near-far} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$left = x_{min} - near * \cot\theta$$
$$right = x_{max} - near * \cot\theta$$
$$bottom = y_{min} - near * \cot\phi$$
$$top = y_{max} - near * \cot\phi$$

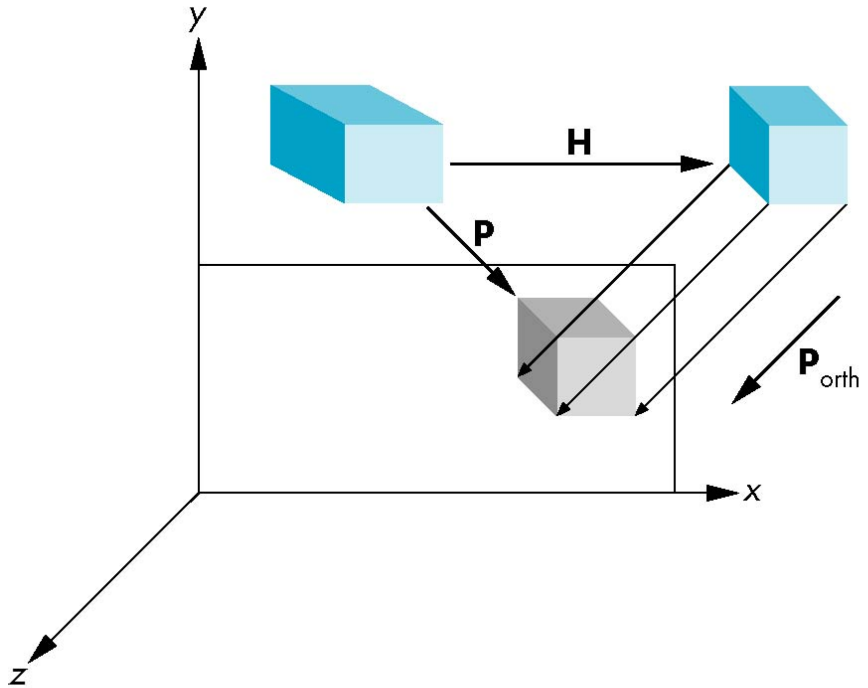
$x_{min}, x_{max}, y_{min}, y_{max}$ are determined by intersections of the four side planes with the *near* plane

Effect on Clipping

The projection matrix $\mathbf{P = STH}$ transforms the original clipping volume to the default clipping volume



Equivalency

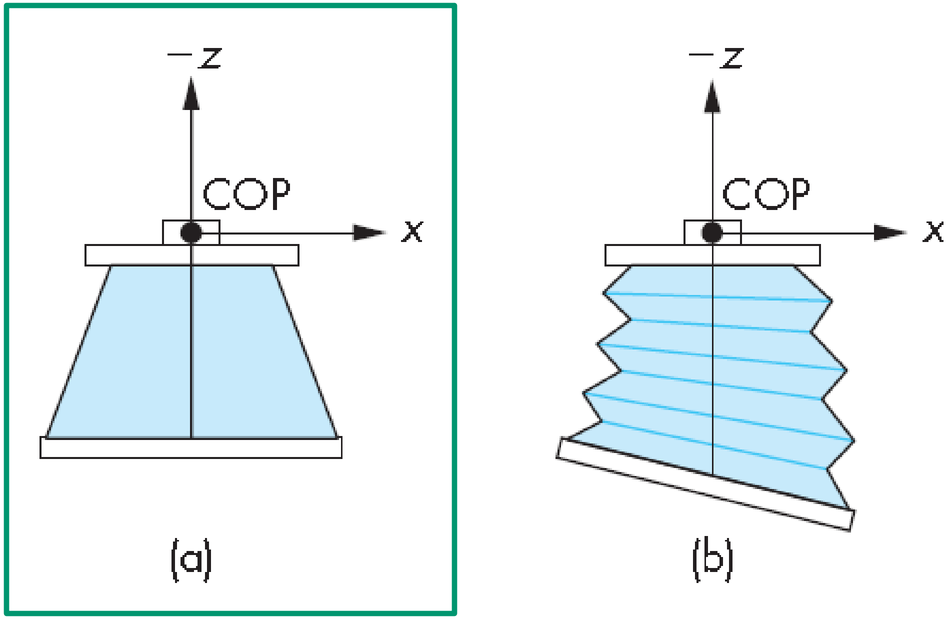


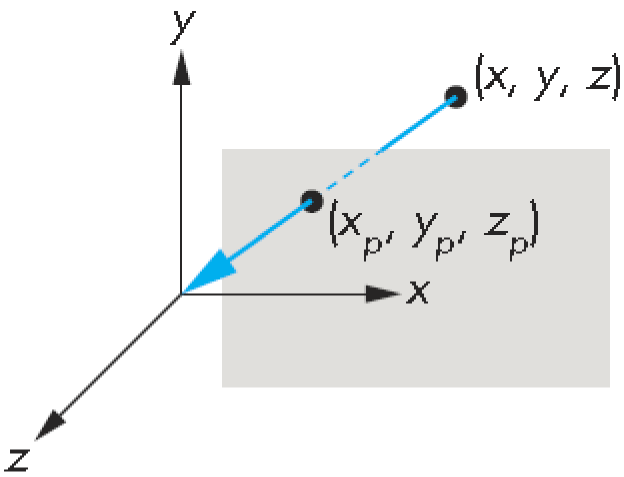
E. Angel and D. Shreiner

Simple Perspective

Center of projection at the origin, projection plane is orthogonal to the z-direction and is parallel to the lens

Projection plane $z = d, d < 0$

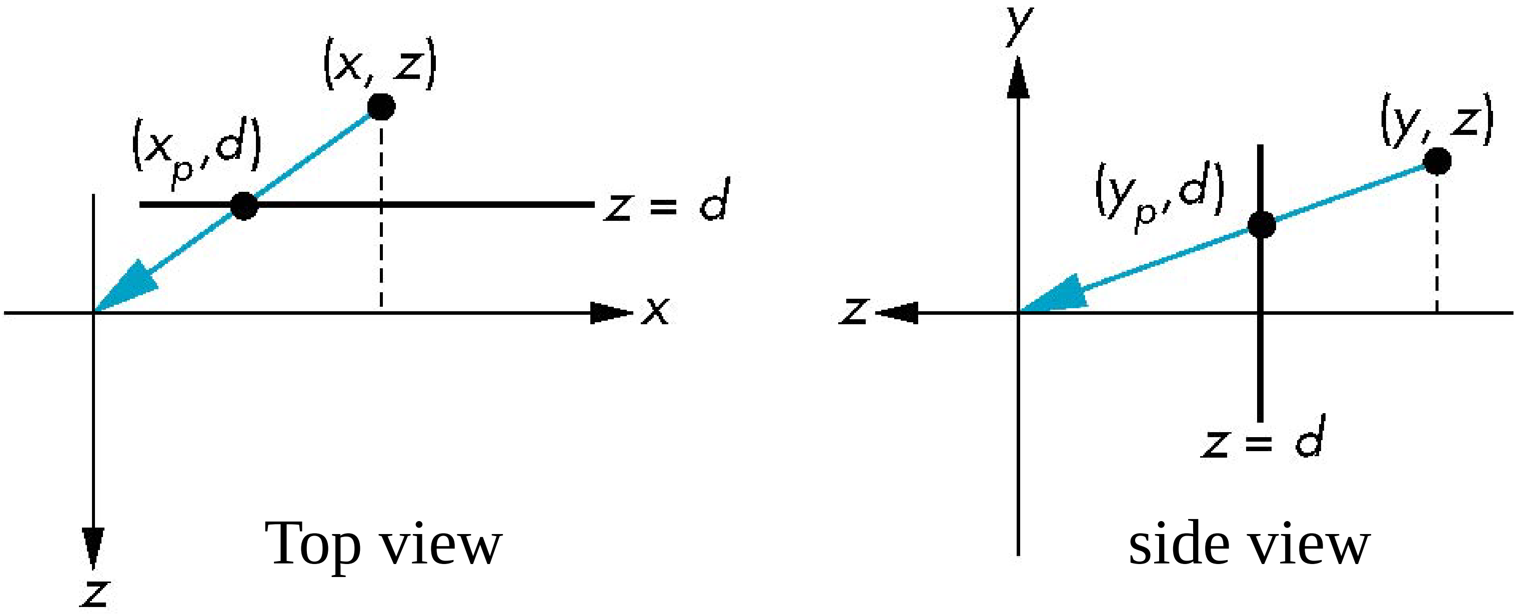




E. Angel and D. Shreiner

Perspective Equations

Consider top and side views



E. Angel and D. Shreiner

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

Nonuniform foreshortening

Perspective Transformation

Perspective transformation is

- **Not linear**
- **Not affine**
- **Not reversible**

Homogeneous Coordinate Form

Consider $P_p = \mathbf{M}P_c$ **where**

A point measured in the clipping frame

The corresponding point measured in the camera frame

$$P_c = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \quad \Rightarrow \quad P_p = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Perspective Division

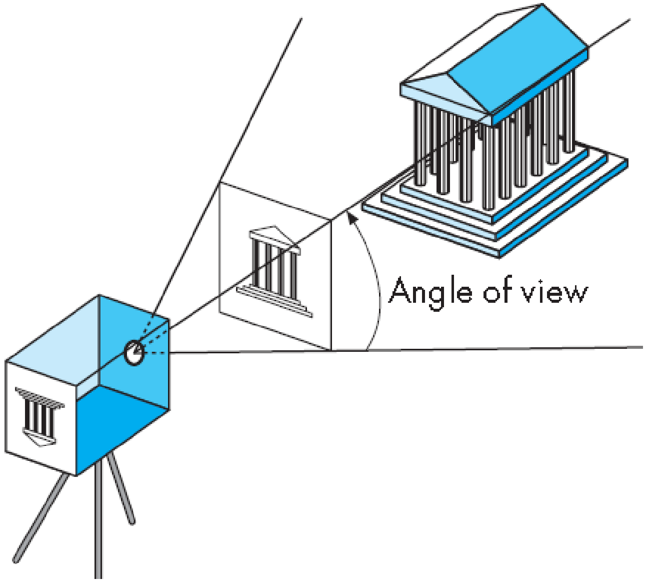
Note that $w \neq 1$, so we must divide by w to return from homogeneous coordinates

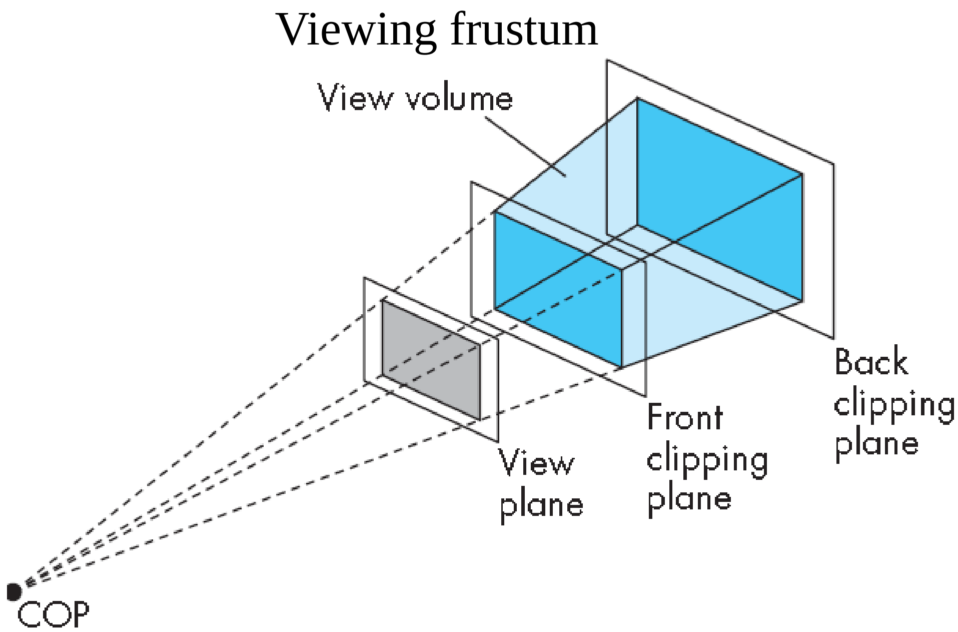
This ***perspective division*** yields the desired perspective equations

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

Perspective with OpenGL

View volume is determined by the angle of view (field of view)





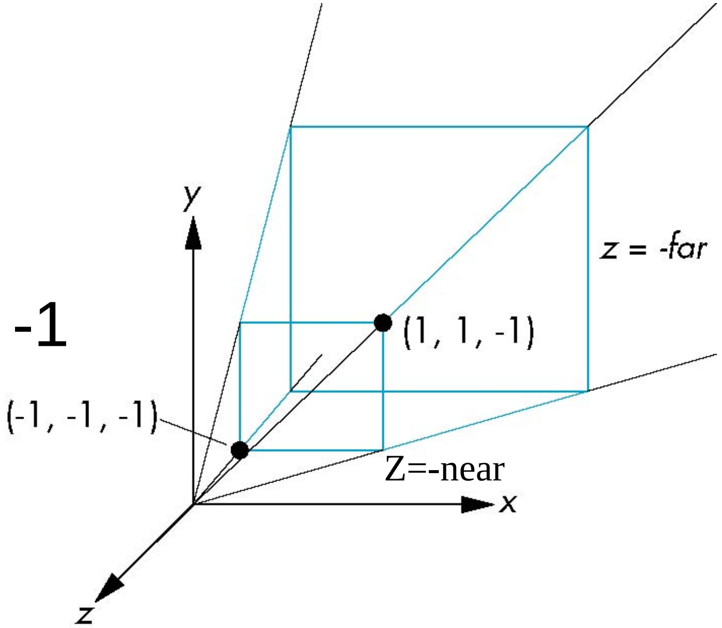
Simple Perspective with OpenGL

Consider a simple perspective with

- the COP at the origin,
- the near clipping plane at z = -1, and
- a 90 degree field of view determined by the planes $x = \pm z$, $y = \pm z$
- Perspective projection matrix is

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

where d = -1



E. Angel and D. Shreiner:

Simple Perspective with OpenGL

A point P (x, y ,z, 1) is projected to a new point Q

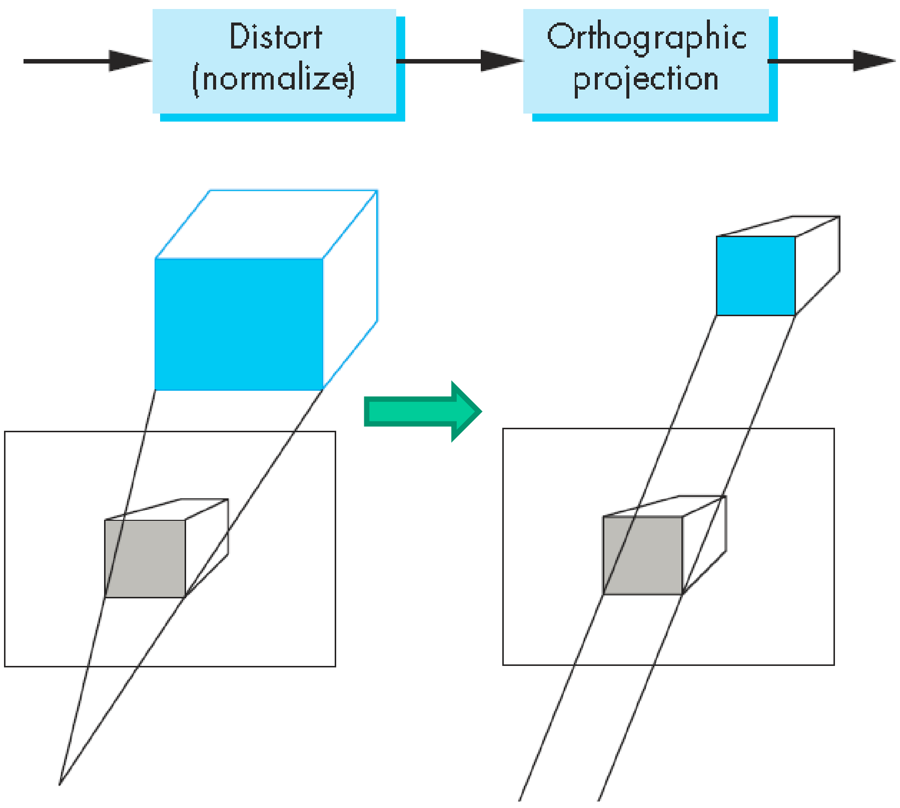
$$Q = \mathbf{M}P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -z \end{bmatrix} = \begin{bmatrix} -x/z \\ -y/z \\ -1 \\ 1 \end{bmatrix}$$

Recall View Normalization

The default projection is **orthogonal (orthographic) projection**

Most graphics systems use ***view normalization***

- All other views are converted to the orthographic view by distorting the objects -- ***normalization***
- Allows use of the same pipeline for all views



E. Angel and D. Shreiner

Perspective Projection and Normalization

We will show the projection can be achieved by ***view normalization*** and an ***orthographic projection***

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

A point P=(x, y, z, 1) is transformed to a new point P'= (x', y', z', w') as

$$\mathbf{P}' = \mathbf{NP}$$

$$\begin{aligned} x' &= x \\ y' &= y \\ z' &= \alpha z + \beta \\ w' &= -z \end{aligned}$$

Perspective Projection and Normalization

After perspective division, we can have P' represented in 3D

P'= (x'', y'', z'')

$x'' = -x/z$

$y'' = -y/z$

$z'' = -(\alpha+\beta/z)$

Then, apply an orthographic projection along the z-axis, we have

$$Q = \mathbf{M}_{\mathrm{orth}} P' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -x/z \\ -y/z \\ -(\alpha+\beta/z) \\ 1 \end{bmatrix} = \begin{bmatrix} -x/z \\ -y/z \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ -z \end{bmatrix}$$

The result is exactly the same as performing perspective projection directly!

Picking α and β

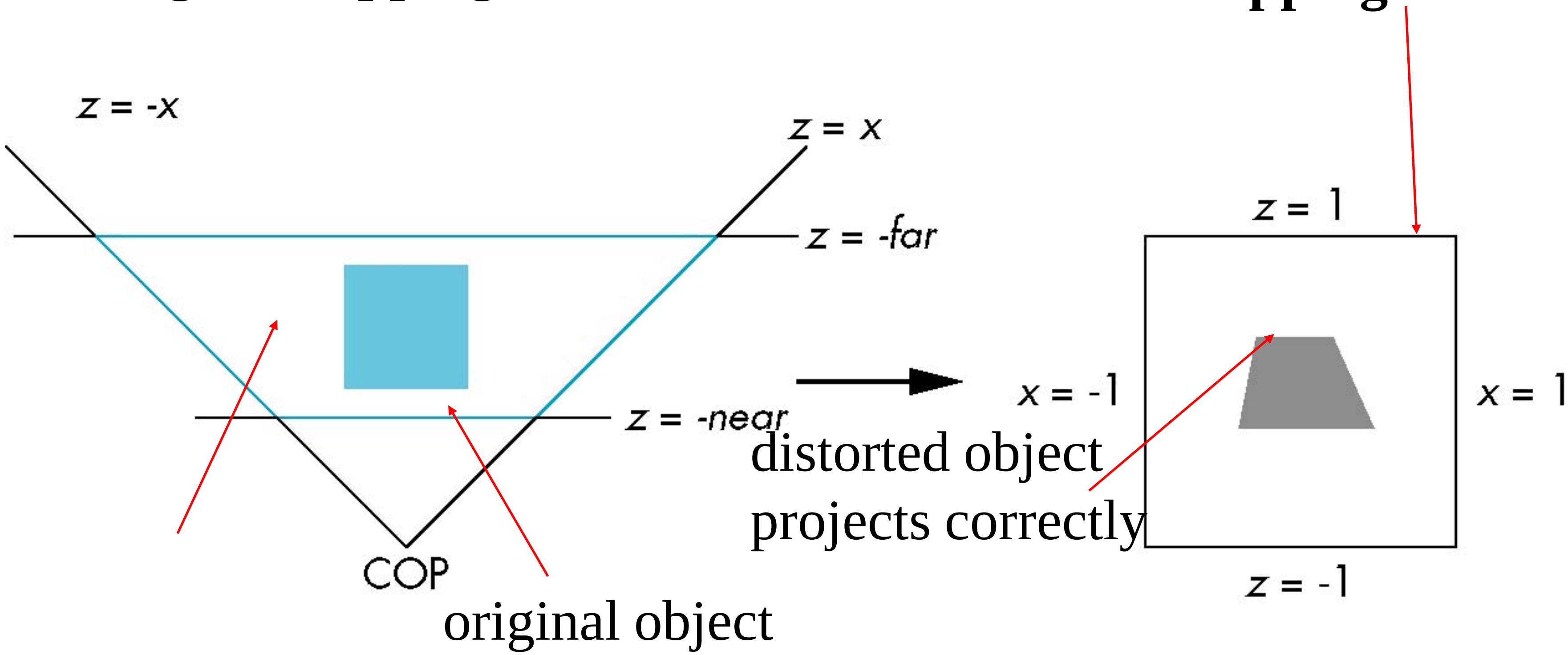
What are α and β for?

After applying view normalization, the new clipping volume should be transformed to the default clipping volume

- The near plane z= - near needs to be mapped to $z'' = -1$
- The far plane z= - far needs to be mapped to $z'' = 1$
- The sides x = $\pm$z and y = $\pm$z needs to be mapped to $x'' = \pm 1$, $y'' = \pm 1$

Normalization Transformation

Original clipping volume **Normalized clipping volume**



Picking α and β

$$z'' = -(\alpha+\beta/z)$$

z =-near will transformed to

$$z'' = -(\alpha+\beta/z) = -(\alpha+\beta/(-near)) = -1$$

z =-far will transformed to

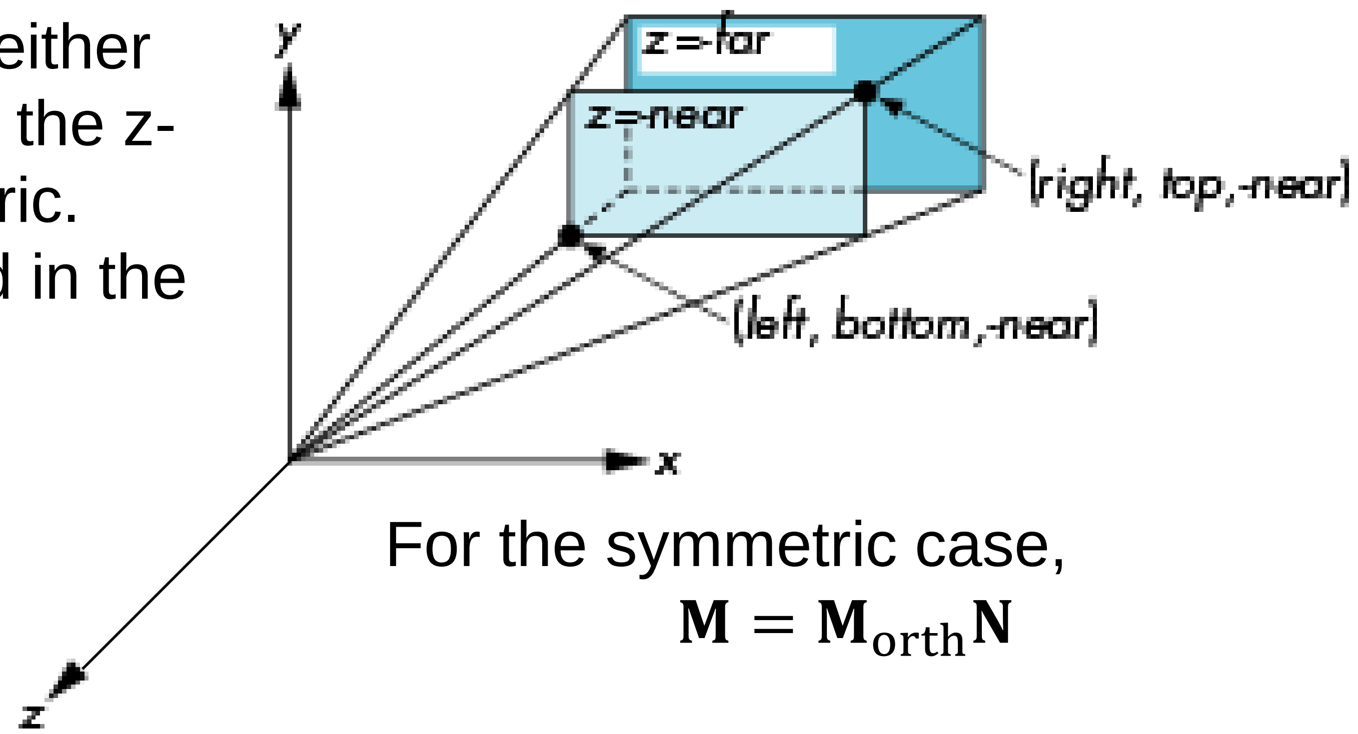
$$z'' = -(\alpha+\beta/z) = -(\alpha+\beta/(-far)) = 1$$

$$\Rightarrow \quad \alpha = \frac{\mathrm{near} + \mathrm{far}}{\mathrm{near} - \mathrm{far}} \quad \text{and} \quad \beta = \frac{2\mathrm{near} * \mathrm{far}}{\mathrm{near} - \mathrm{f}ar}$$

OpenGL Perspective

`Frustum(left,right,bottom,top,near,far)`

- Frustum can be either symmetric about the z-axis or asymmetric.
- All are measured in the *camera frame.*



For the symmetric case,

$$\mathbf{M} = \mathbf{M}_{\text{orth}}\mathbf{N}$$

OpenGL Perspective

How do we handle the asymmetric frustum?

Convert the frustum to a symmetric one by performing a shear followed by a scaling to get the normalized perspective volume.

Step 1 Shear: Transform the point $\left(\frac{left+right}{2}, \frac{top+bottom}{2}, -near\right)$ to $(0,0,-near)$

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

where $\cot\theta = \frac{\text{left}+\text{right}}{2\text{near}}$ and $\cot\varphi = \frac{top+bottom}{2\text{near}}$

OpenGL Perspective

After shearing, the resulting frustum is described by

4 sides $\quad x = \pm \frac{\text{right} - \text{left}}{-2\text{near}} \quad y = \pm \frac{\text{top} - \text{bottom}}{-2\text{near}}$

Near plane $\quad z = -near$

Far plane $\quad z = -far$

OpenGL Perspective

Step 2: Scaling

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

Step 3: Perspective normalization N

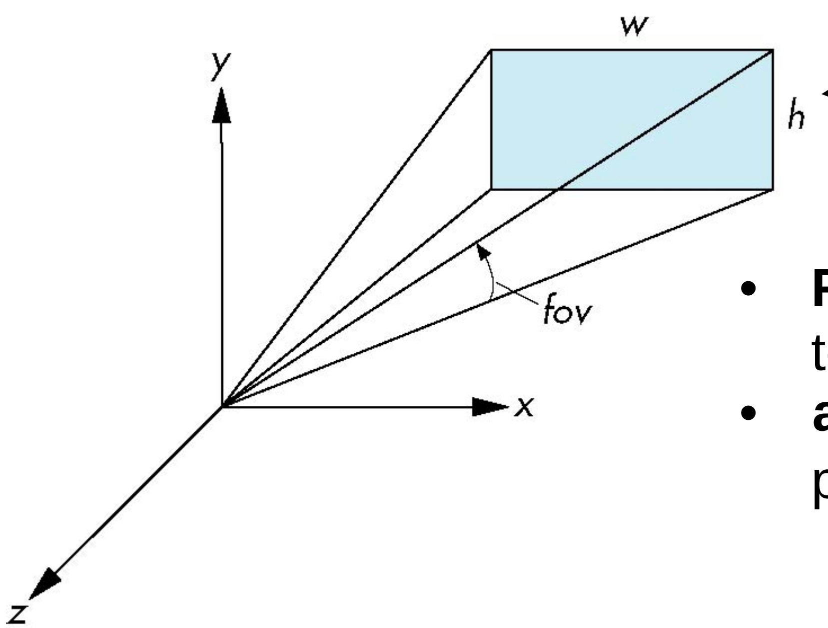
The final perspective matrix

$$\mathbf{M}_{\boldsymbol{p}} = \mathbf{NSH} = \begin{bmatrix} \frac{2near}{right-left} & 0 & \frac{left+right}{right-left} & 0 \\ 0 & \frac{2near}{top-bottom} & \frac{bottom+top}{top-bottom} & 0 \\ 0 & 0 & \frac{near+far}{near-far} & \frac{2near*far}{near-far} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Using Field of View: Perspective()

An alternative and more convenient way is to use the field of view

`Perspective(fovy, aspect, near, far)` often provides a better interface



projection plane

- **Fovy** is the angle between the top and the bottom planes
- **`aspect = w/h`** of projection plane

Using Field of View: Perspective()

Enforce a symmetric frustum

$$left = -right$$
$$bottom = -top$$

Frustum() ⟺Perspective()

$$fovy = 2\tan^{-1}\frac{top - bottom}{2near}$$

$$left = aspect * bottom$$
$$top = \tan\left(\frac{fovy}{2}\right) * near$$