CS559: Computer Graphics

Lecture 9: Projection

Li Zhang

Spring 2008

Today

- Projection

- Reading:
  - Shirley ch 7

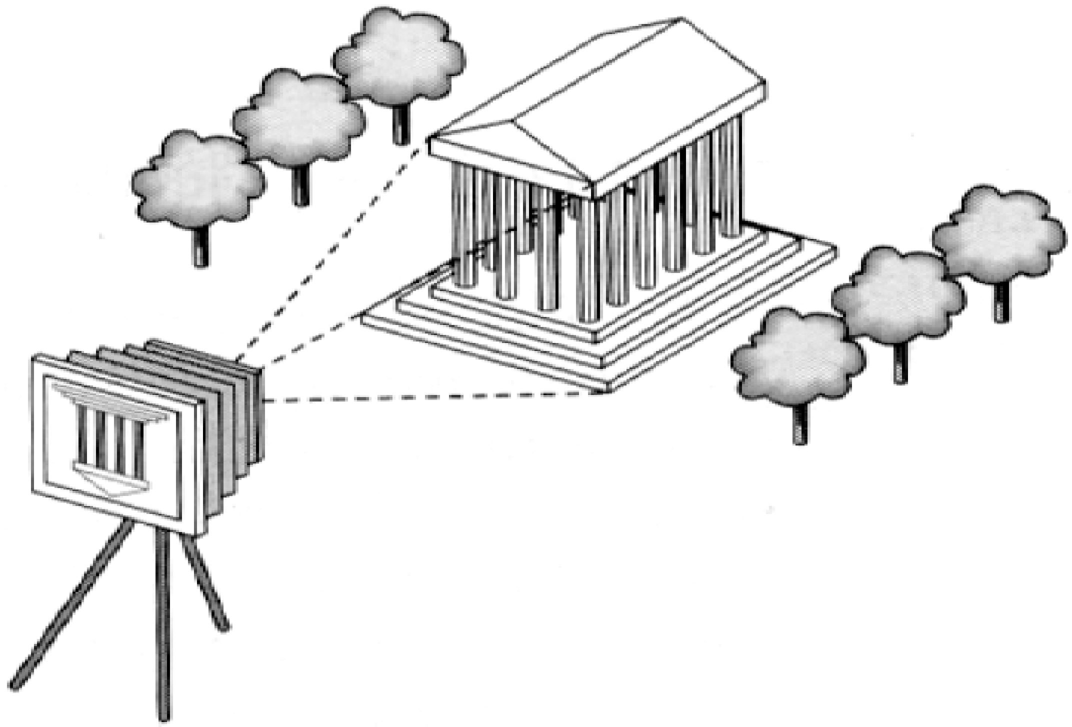
Geometric Interpretation 3D Rotations

- Can construct rotation matrix from 3 orthonormal vectors
- Rows of matrix are 3 unit vectors of new coord frame
- Effectively, projections of point into new coord frame

$$R_{uvw} = \begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{pmatrix}$$

$$Rp = \begin{pmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = ? \quad \begin{pmatrix} u \bullet p \\ v \bullet p \\ w \bullet p \end{pmatrix}$$

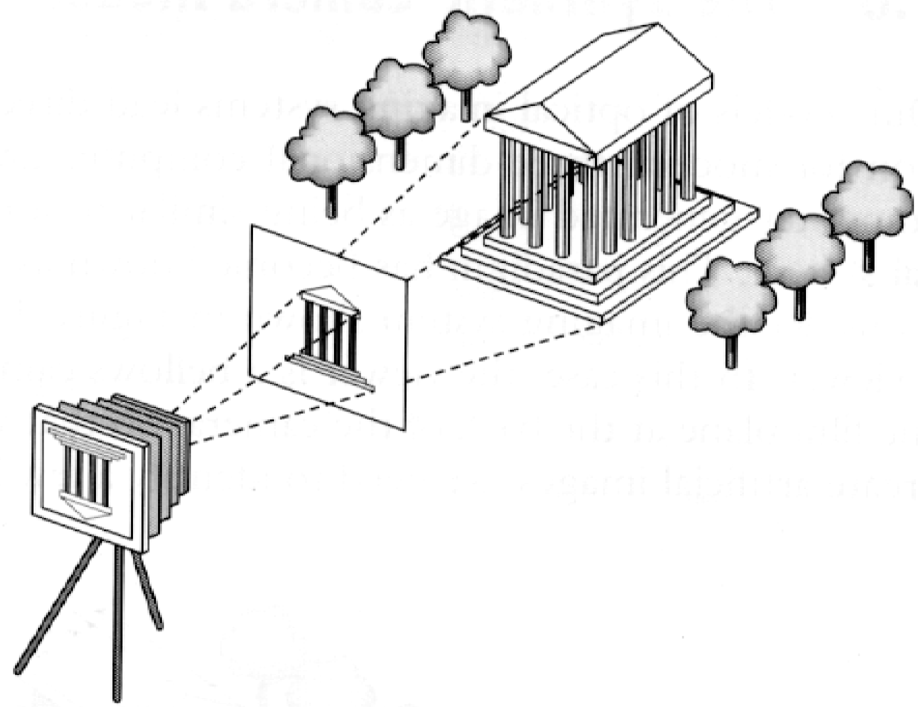
The 3D synthetic camera model



The **synthetic camera model involves two components,** specified *independently:*

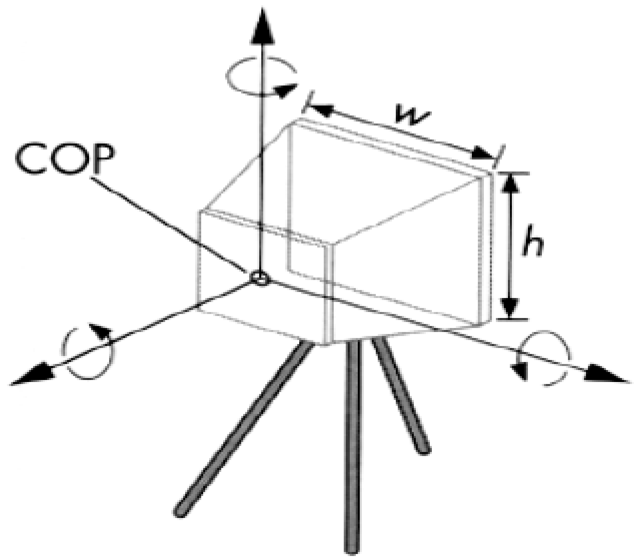
- objects (a.k.a. **geometry)**
- viewer (a.k.a. **camera)**

Imaging with the synthetic camera



The image is rendered onto an **image plane or projection plane (usually** in front of the camera).
Projectors emanate from the center of projection (COP) at the center of the lens (or pinhole).
The image of an object point *P is at the intersection of the projector* through *P and the image plane.*

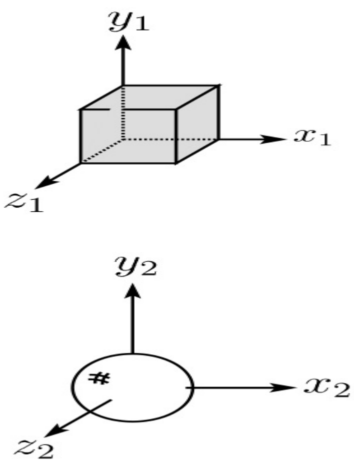
Specifying a viewer



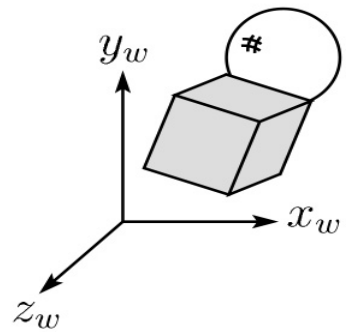
Camera specification requires four kinds of parameters:

- *Position: the COP.*
- *Orientation: rotations about axes with origin at the COP.*
- *Focal length: determines the size of the image on the film plane, or* the **field of view.**
- *Film plane: its width and height.*

3D Geometry Pipeline



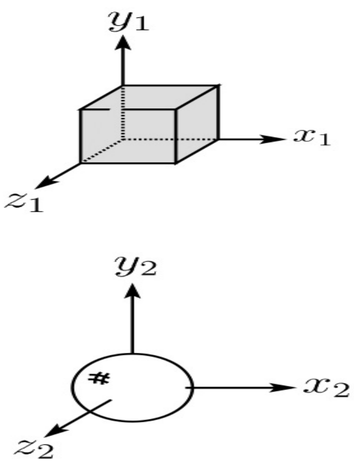
Model Space
(Object Space)



World Space

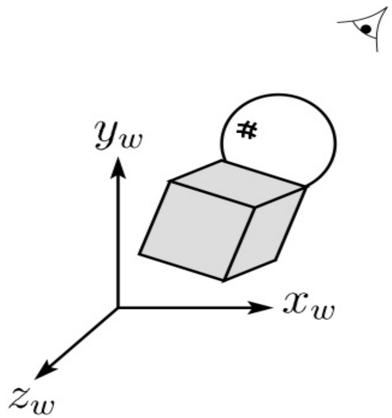
Rotation
Translation
Resizing

3D Geometry Pipeline



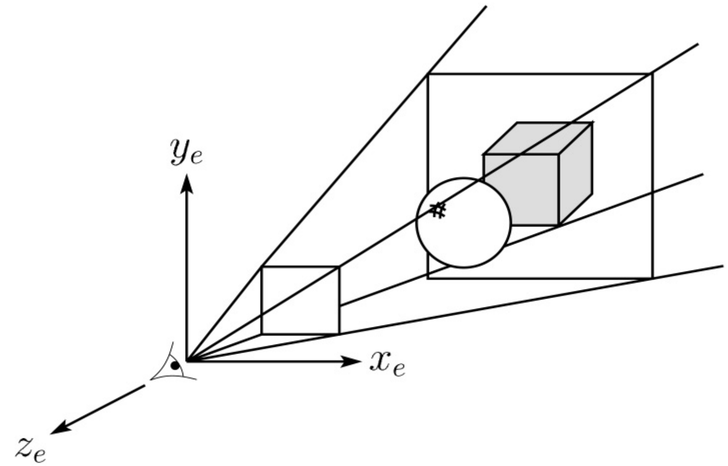
Model Space
(Object Space)

Rotation
Translation
Resizing



World Space

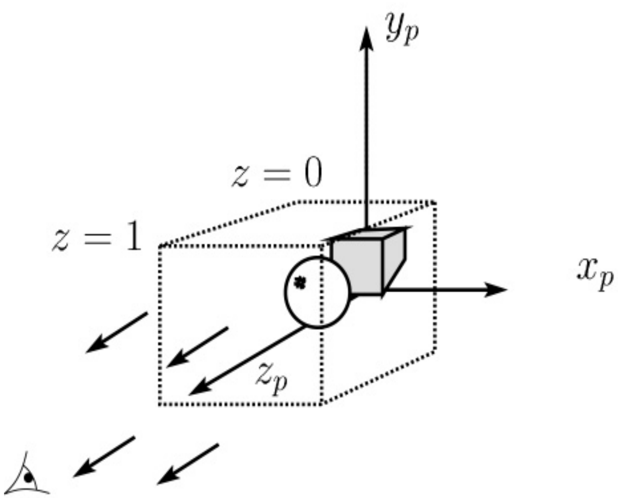
Rotation
Translation

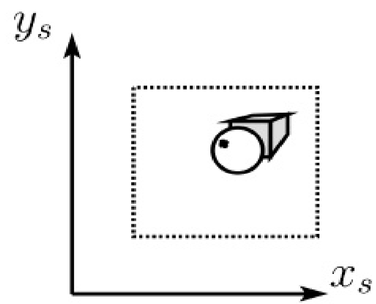


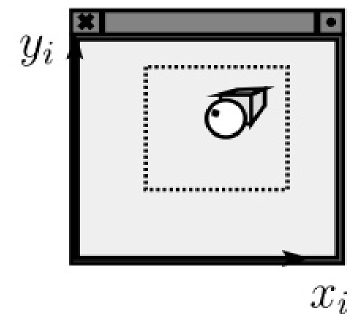
Eye Space
(View Space)

Projective,
Scale
Translation

3D Geometry Pipeline (cont'd)



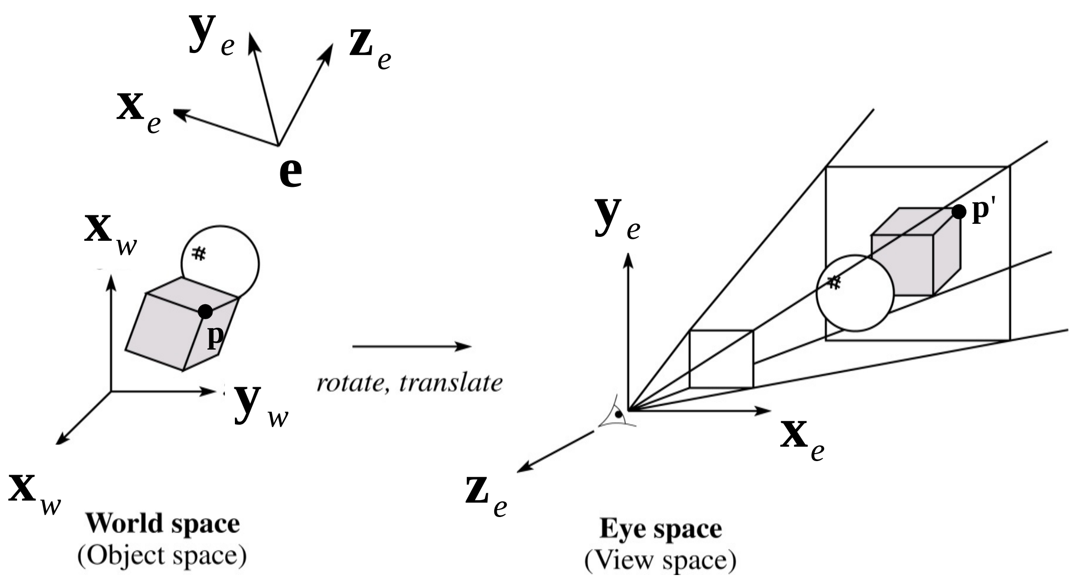




Normalized Projection Space → Project → Screen Space (2D) → Scale, Translation → Image Space (pixels) Raster Space

World -> eye transformation

- Let's look at how we would compute the world->eye transformation.

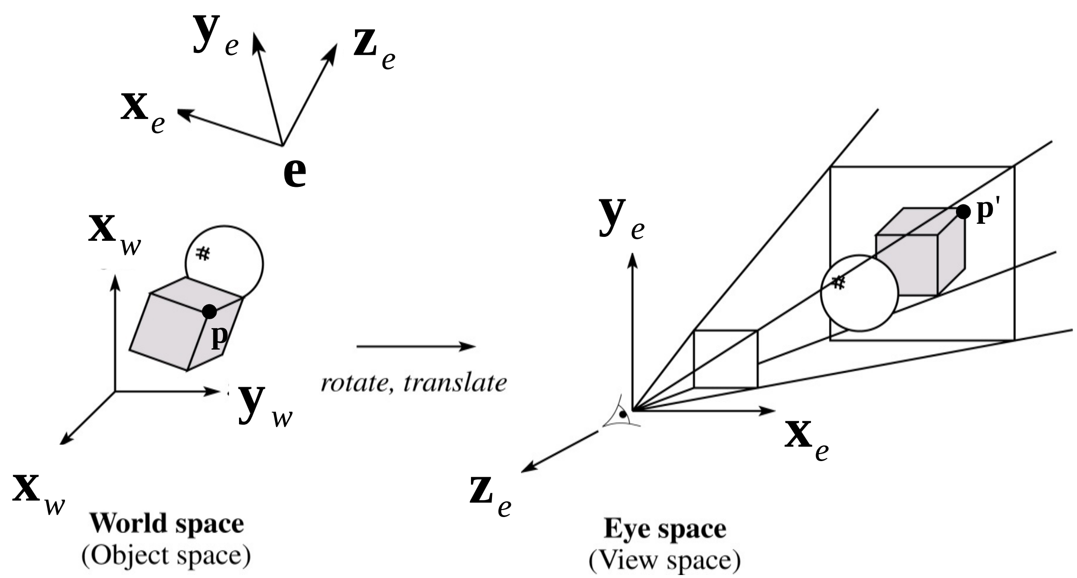


$$\mathbf{p}' = ?$$

$$x_e = \mathbf{x}_e \bullet \mathbf{p}\,?$$

World -> eye transformation

- Let’s look at how we would compute the world->eye transformation.



$$\mathbf{p}' = ?$$

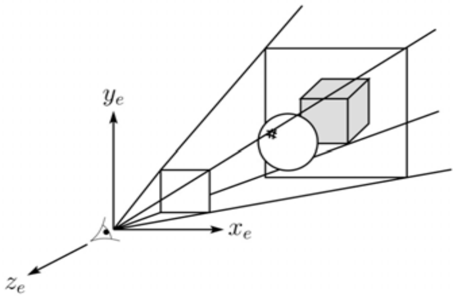
$$x_e = \mathbf{x}_e \bullet (\mathbf{p} - \mathbf{e})$$

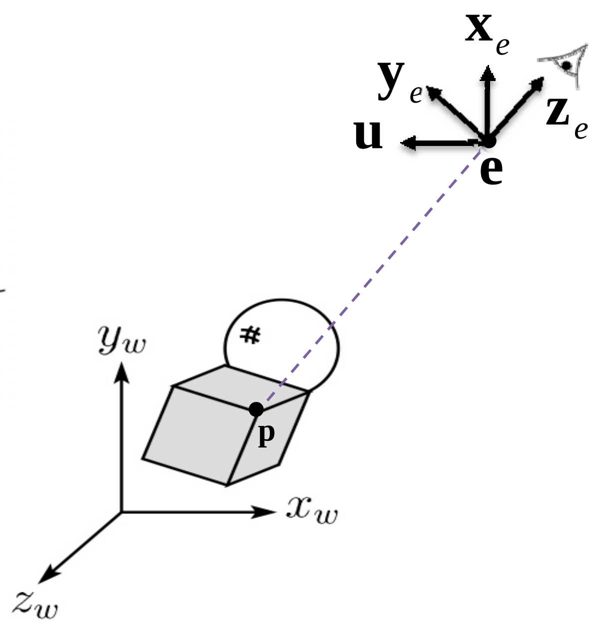
$$y_e = \mathbf{y}_e \bullet (\mathbf{p} - \mathbf{e})$$

$$z_e = \mathbf{z}_e \bullet (\mathbf{p} - \mathbf{e})$$

$$\mathbf{M}_v = \begin{bmatrix} - & \mathbf{x}_e^{\mathrm{T}} & - & 0 \\ - & \mathbf{y}_e^{\mathrm{T}} & - & 0 \\ - & \mathbf{z}_e^{\mathrm{T}} & - & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & | \\ 0 & 1 & 0 & -\mathbf{e} \\ 0 & 0 & 1 & | \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

How to specify eye coordinate?



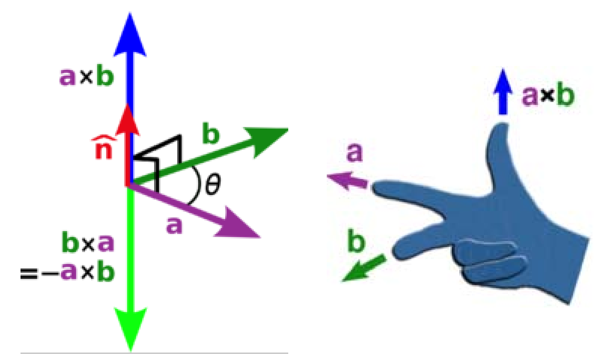


$$\mathbf{z}_e = -\frac{\mathbf{p}-\mathbf{e}}{|\mathbf{p}-\mathbf{e}|}$$

$$\mathbf{x}_e = \frac{\mathbf{u}\times\mathbf{z}_e}{|\mathbf{u}\times\mathbf{z}_e|}$$

$$\mathbf{y}_e = \frac{\mathbf{z}_e\times\mathbf{x}_e}{|\mathbf{z}_e\times\mathbf{x}_e|}$$

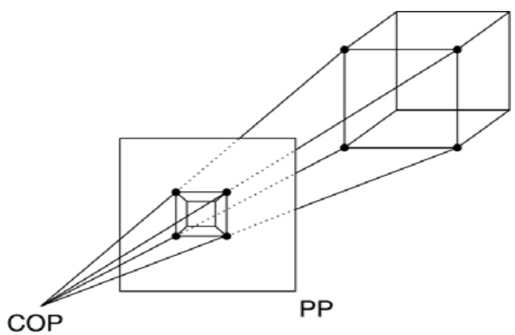
1. Give eye location e
2. Give target position p
3. Give upward direction u

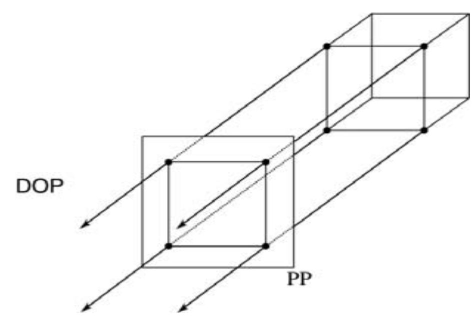


OpenGL has a helper command:
gluLookAt (*eyex*, *eyey*, *eyez*, *px*, *py*, *pz*, *upx*, *upy*, *upz*)

Projections

- **Projections** transform points in *n*-space to *m*-space, where $m<n$.
- In 3-D, we map points from 3-space to the **projection plane** (PP) (a.k.a., image plane) along **projectors** (a.k.a., viewing rays) emanating from the center of projection (COP):





- There are two basic types of projections:
  - Perspective – distance from COP to PP finite
  - Parallel – distance from COP to PP infinite

Parallel projections

- For parallel projections, we specify a **direction of projection** (DOP) instead of a COP.
- We can write orthographic projection onto the *z=0* plane with a simple matrix, such that x'=x, y'=y.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \qquad\qquad \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Parallel projections

- For parallel projections, we specify a **direction of projection** (DOP) instead of a COP.
- We can write orthographic projection onto the *z=0* plane with a simple matrix, such that x'=x, y'=y.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Normally, we do not drop the z value right away. Why not?

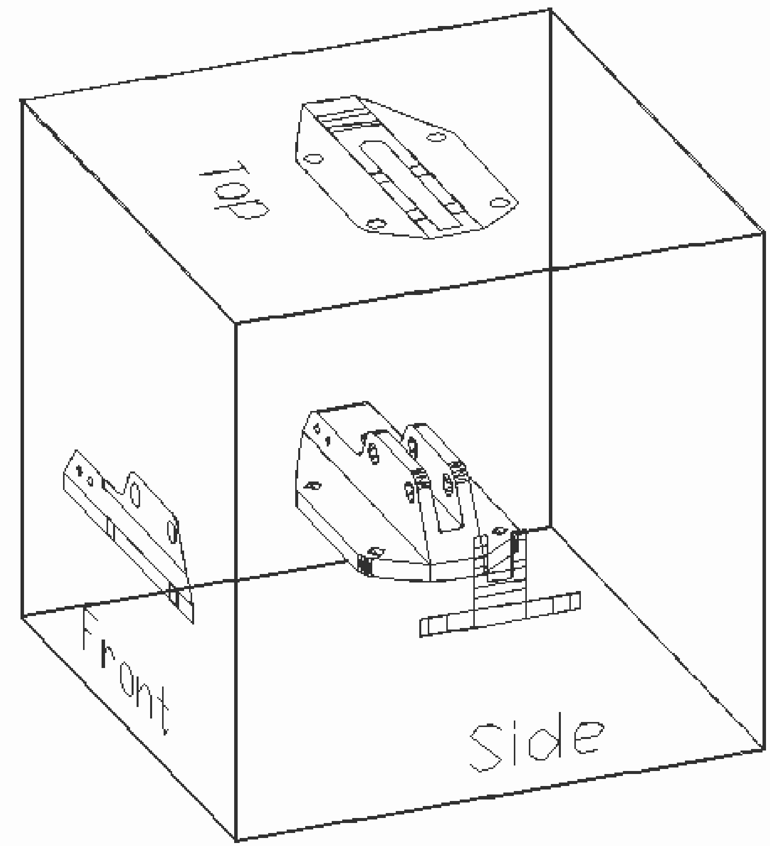
Properties of parallel projection

- Properties of parallel projection:
  - Are actually a kind of affine transformation
    - Parallel lines remain parallel
    - Ratios are preserved
    - Angles not (in general) preserved
  - Not realistic looking
  - Good for exact measurements,

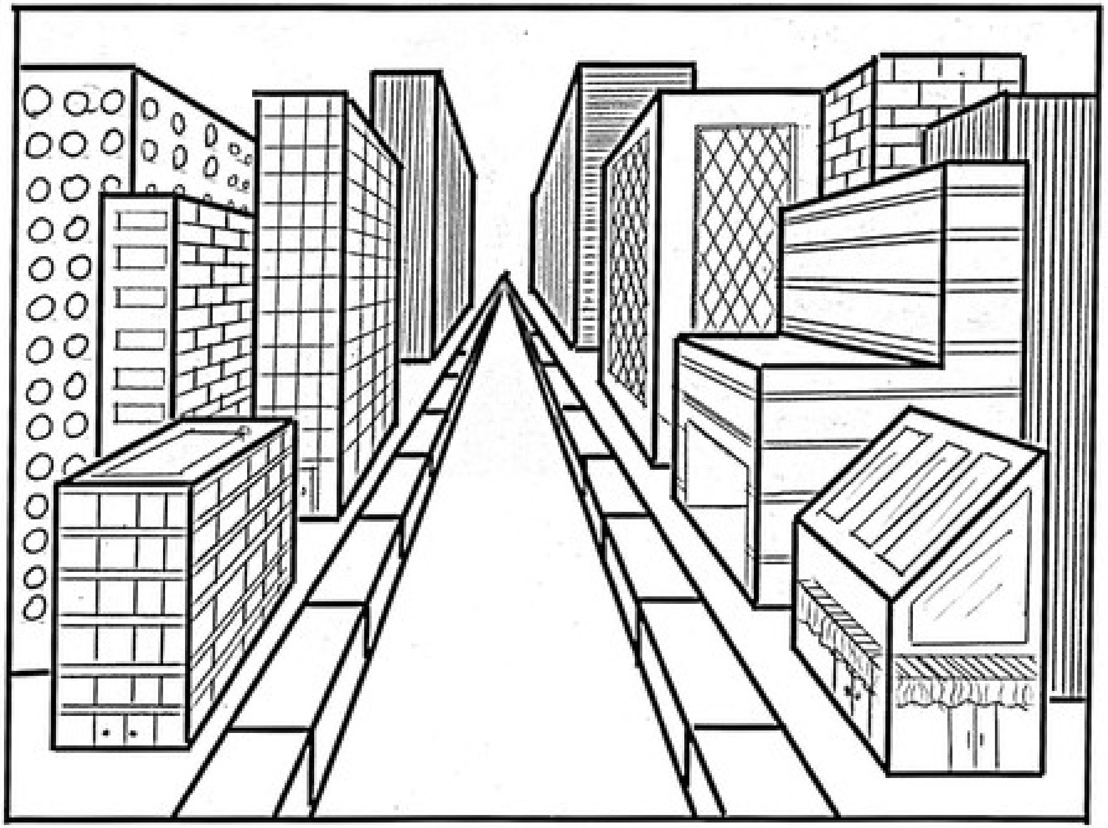
    Most often used in
    - CAD,
    - architectural drawings,
    - etc.,

    where taking exact measurement

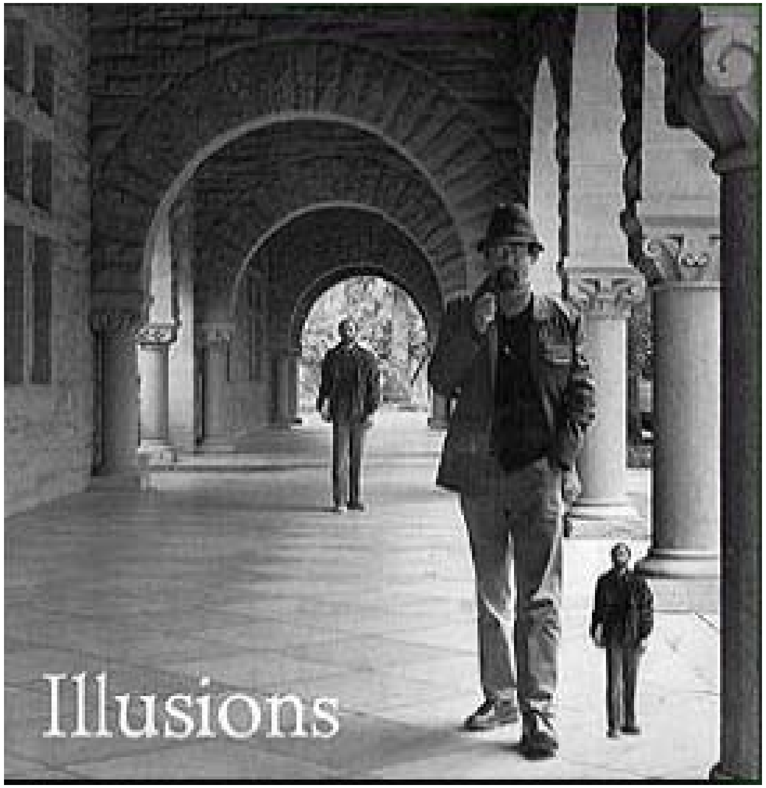
    is important



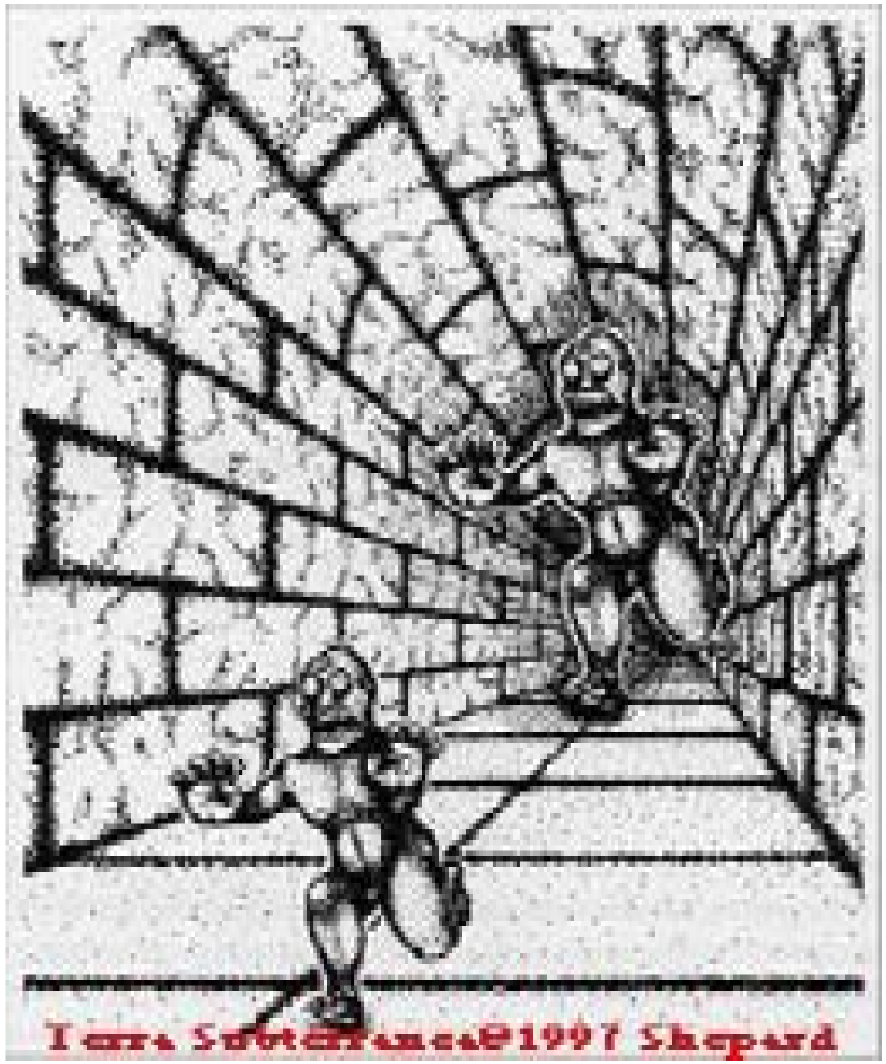
Perspective effect



Perspective Illusion

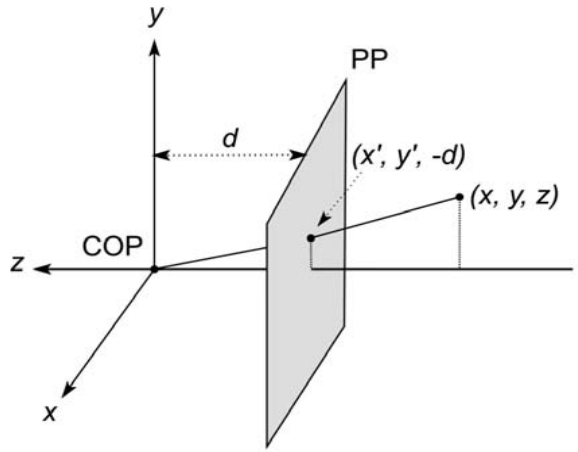


Perspective Illusion



Derivation of perspective projection

Consider the projection of a point onto the projection plane:



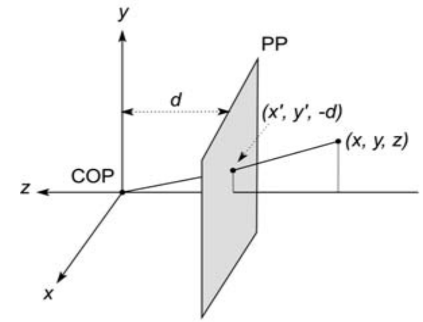
By similar triangles, we can compute how much the *x* and *y* coordinates are scaled:

$$\frac{y'}{d} = \frac{y}{z} \Rightarrow y' = \frac{d}{z} y$$

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

Homogeneous coordinates and perspective projection

Now we can re-write the perspective projection as a matrix equation:



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} (d/z)x \\ (d/z)y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

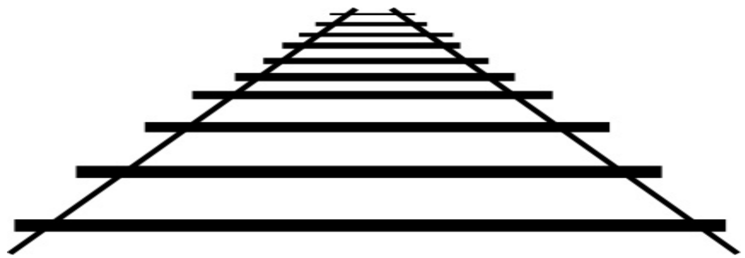
Orthographic projection

$$= \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} (d/z)x \\ (d/z)y \\ d \\ 1 \end{bmatrix}$$

Vanishing points

What happens to two parallel lines that are not parallel to the projection plane?

Think of train tracks receding into the horizon...



The equation for a line is:

$$\mathbf{l} = \mathbf{p} + t\,\mathbf{v} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} + t \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

After perspective transformation we get:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} p_x + tv_x \\ p_y + tv_y \\ (p_z + tv_z)/d \end{bmatrix}$$

Vanishing points (cont'd)

Dividing by *w*:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} p_x + tv_x \\ p_y + tv_y \\ (p_z + tv_z)/d \end{bmatrix} = \begin{bmatrix} \frac{p_x + tv_x}{p_z + tv_z} d \\ \frac{p_y + tv_y}{p_z + tv_z} d \\ 1 \end{bmatrix}$$

Letting *t* go to infinity: $= \begin{bmatrix} (v_x / v_z) d \\ (v_y / v_z) d \\ 1 \end{bmatrix}$

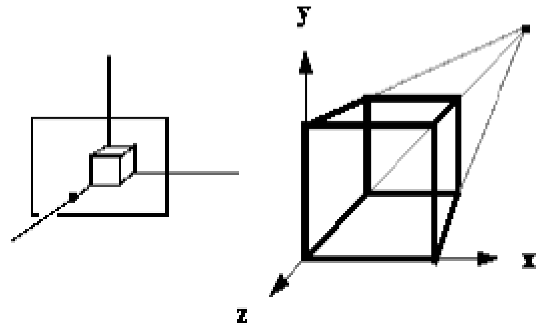
We get a point!

What happens to the line **l** = **q** + *t***v**?

Each set of parallel lines intersect at a **vanishing point** on the PP.

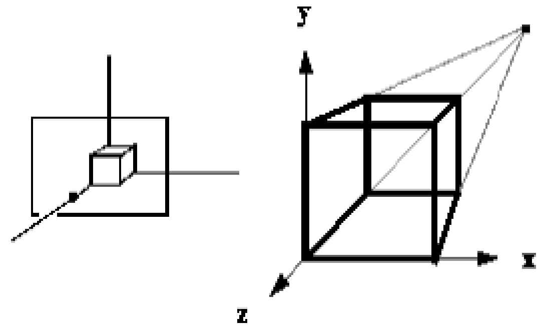
Q: How many vanishing points are there?

Vanishing points

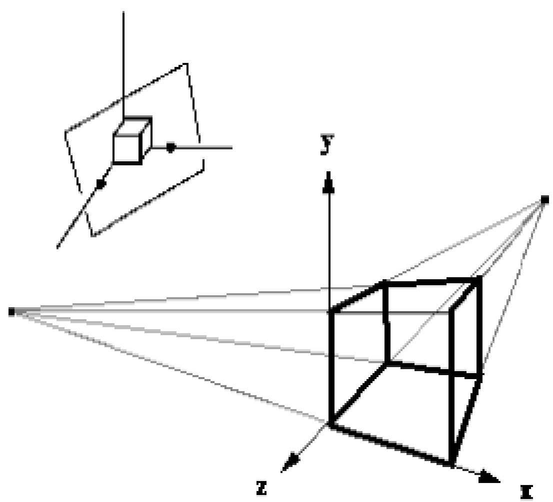


One Point Perspective
(z-axis vanishing point)

Vanishing points

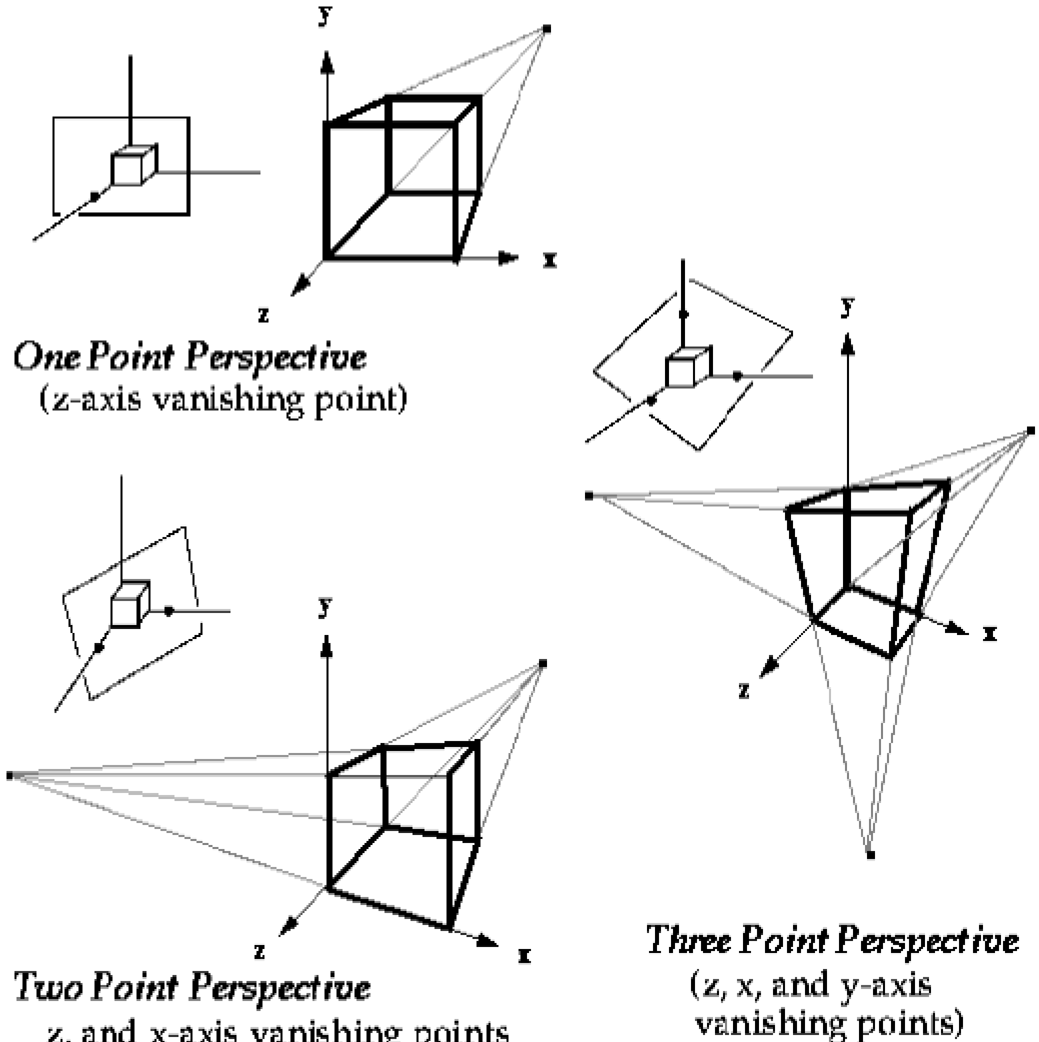


One Point Perspective
(z-axis vanishing point)



Two Point Perspective
z, and x-axis vanishing points

Vanishing points



General Projective Transformation

$$\text{affine transform} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & g & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

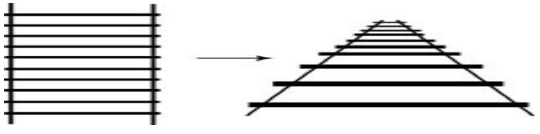
$$\text{projective transform} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & g & k & l \\ m & n & o & p \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The perspective projection is an example of a **projective transformation**.

Properties of perspective projections

Here are some properties of projective transformations:

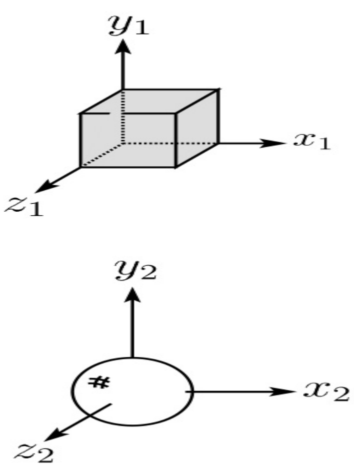


- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved

One of the advantages of perspective projection is that size varies inversely with distance – looks realistic.

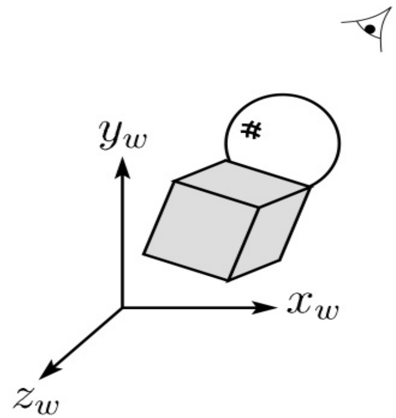
A disadvantage is that we can't judge distances as exactly as we can with parallel projections.

3D Geometry Pipeline



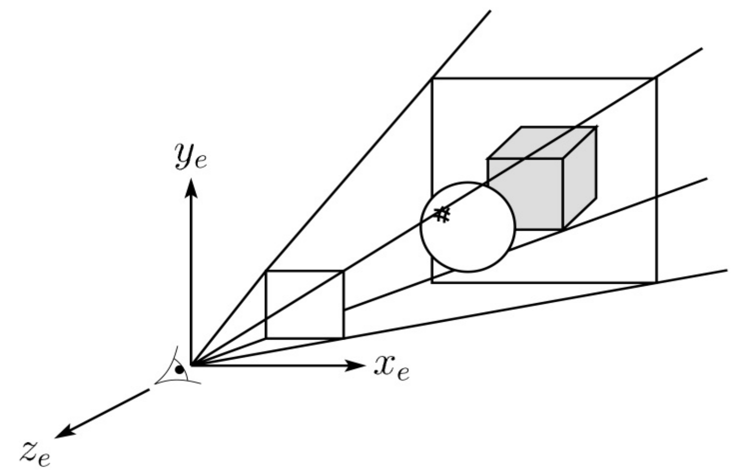
Model Space
(Object Space)

Rotation
Translation
Resizing



World Space

Rotation
Translation



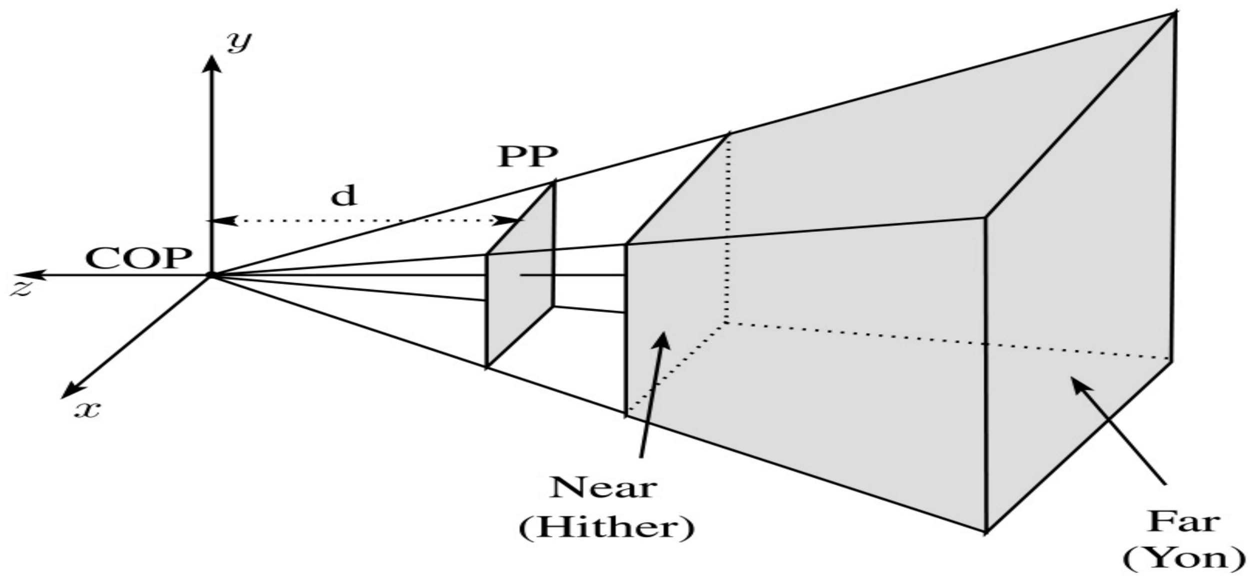
Eye Space
(View Space)

Projective,
Scale
Translation

Clipping and the viewing frustum

The center of projection and the portion of the projection plane that map to the final image form an infinite pyramid. The sides of the pyramid are **clipping planes**.

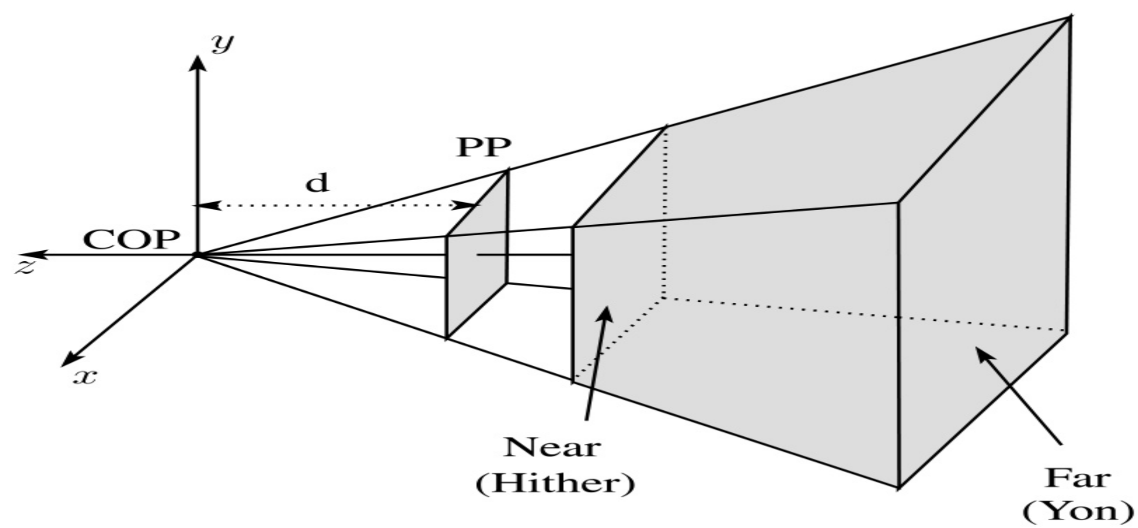
Frequently, additional clipping planes are inserted to restrict the range of depths. These clipping planes are called the **near** and **far** or the **hither** and **yon** clipping planes



All of the clipping planes bound the **viewing frustum**.

Clipping and the viewing frustum

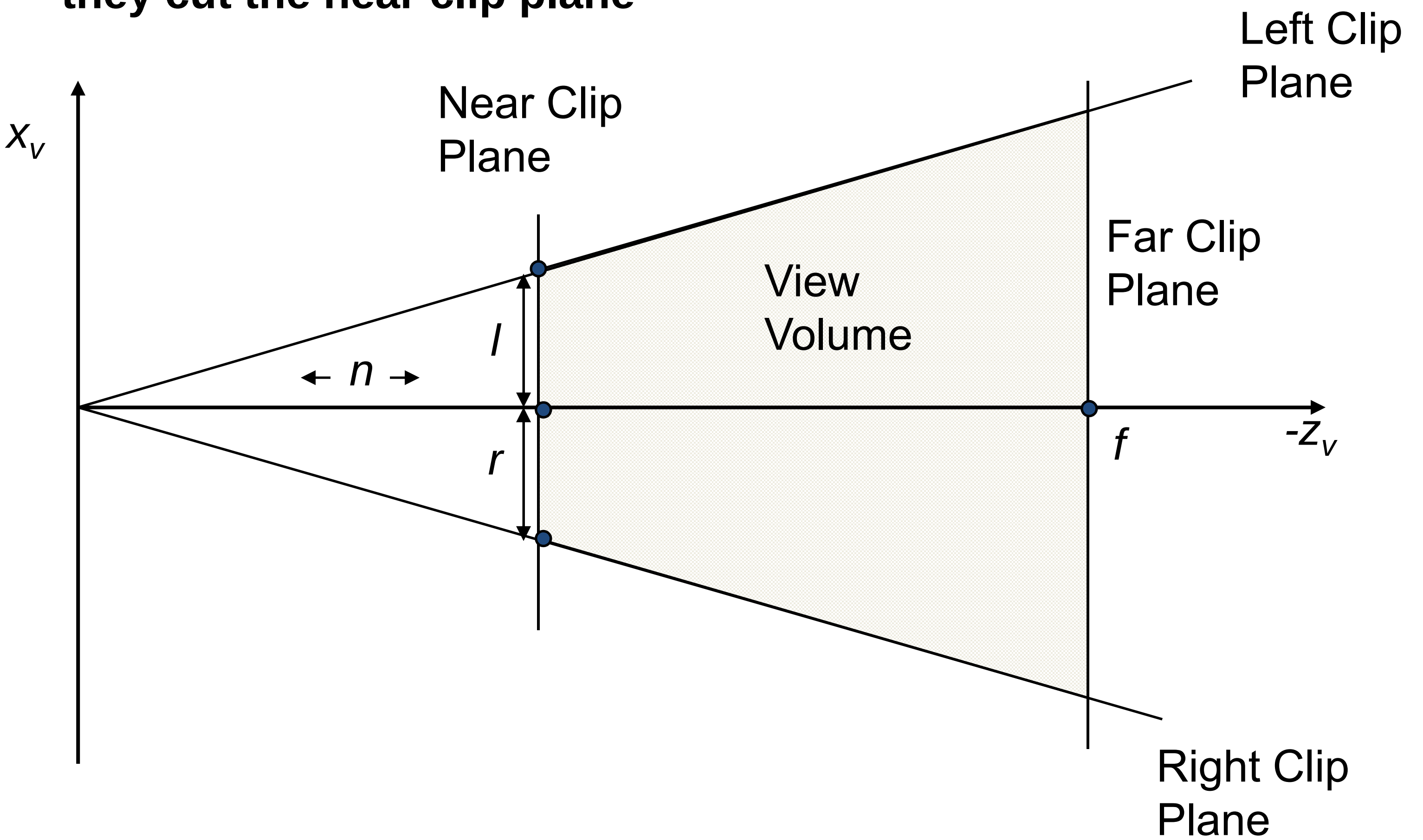
- Notice that it doesn't really matter where the image plane is located, once you define the view volume
  - You can move it forward and backward along the *z* axis and still get the same image, only scaled



Usually d = near

Clipping Planes

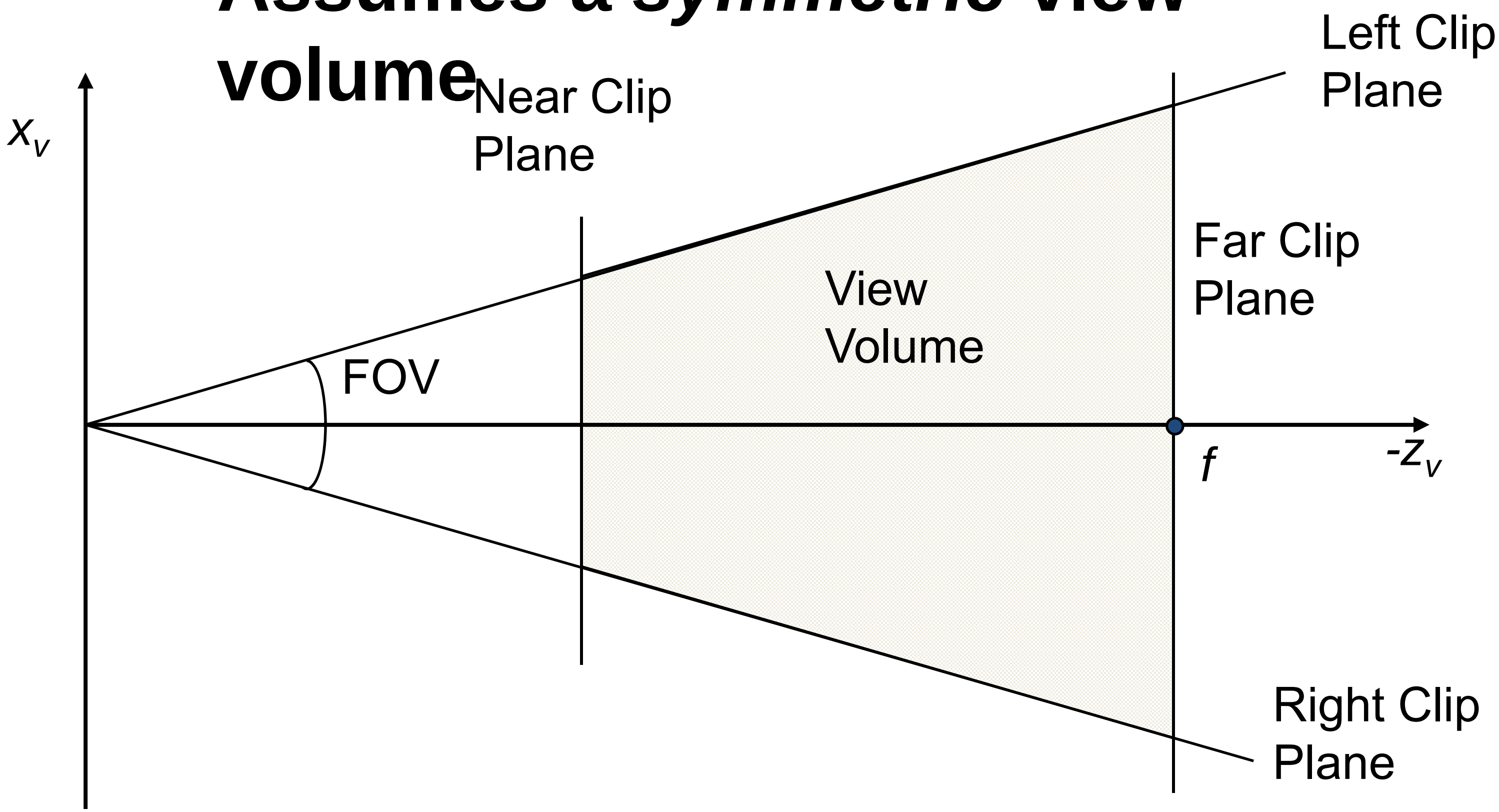
The left/right/top/bottom planes are defined **according to where they cut the near clip plane**



Need 6 parameters,
Or, define the left/right and top/bottom clip planes by the *field of view*

Field of View

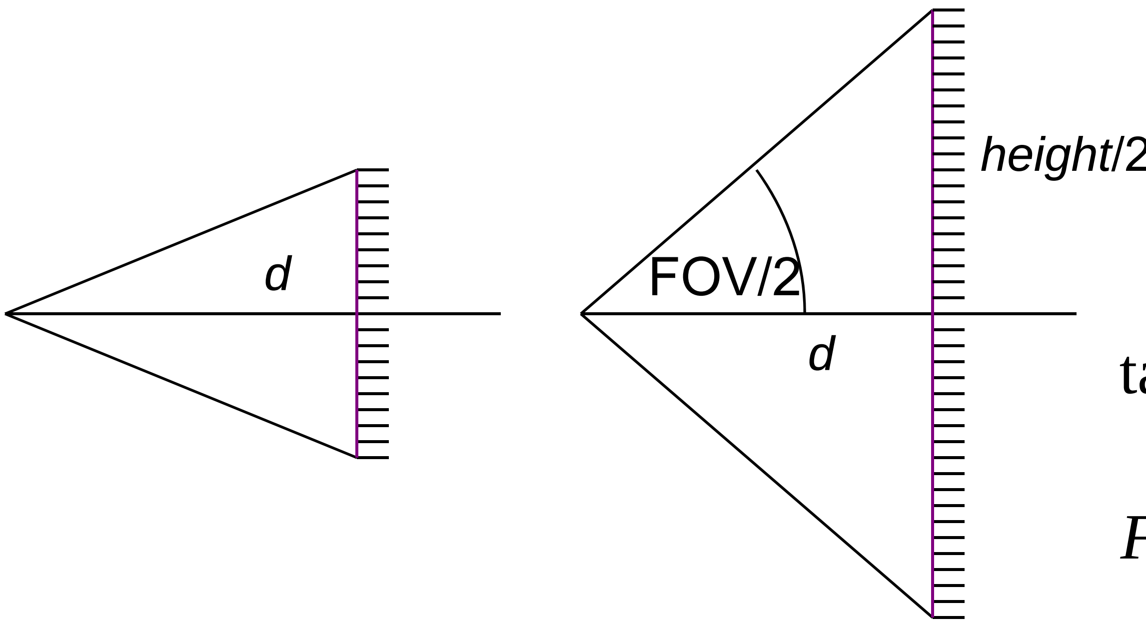
Assumes a *symmetric* view volume



Need 4 parameters,
Or, define the near, far, fov, aspect ratio

Focal Distance to FOV

- You must have the image size to do this conversion
  - Why? Same *d*, different image size, different FOV



$$\tan\left(\frac{FOV}{2}\right) = \frac{height}{2d}$$

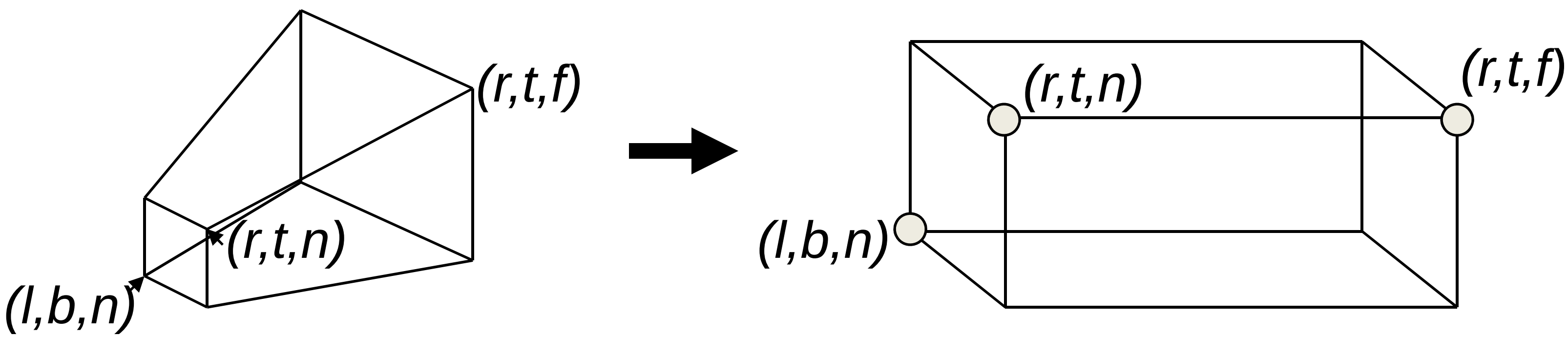
$$FOV = 2\tan^{-1}\left(\frac{height}{2d}\right)$$

Perspective Parameters

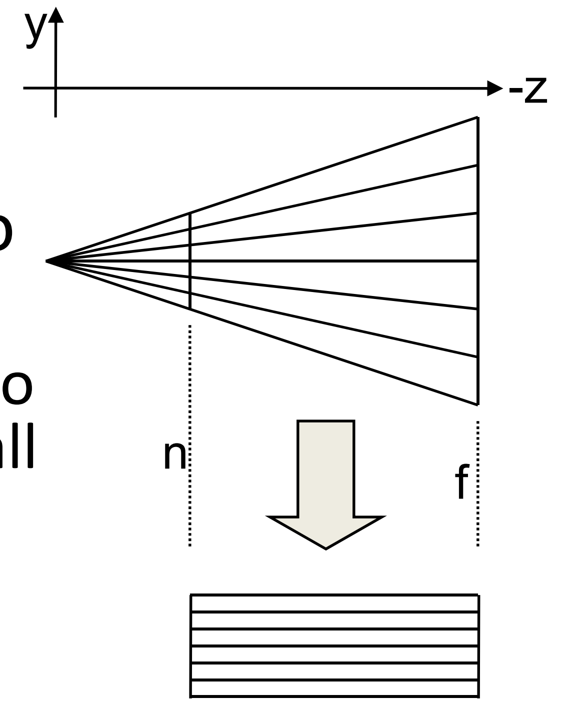
- We have seen three different ways to describe a perspective camera
  - Clipping planes, Field of View, Focal distance,
- The most general is clipping planes – they directly describe the region of space you are viewing
- For most graphics applications, field of view is the most convenient
- You can convert one thing to another

Perspective Projection Matrices

- We want a matrix that will take points in our perspective view volume and transform them into the orthographic view volume
  - This matrix will go in our pipeline before an orthographic projection matrix



Mapping Lines

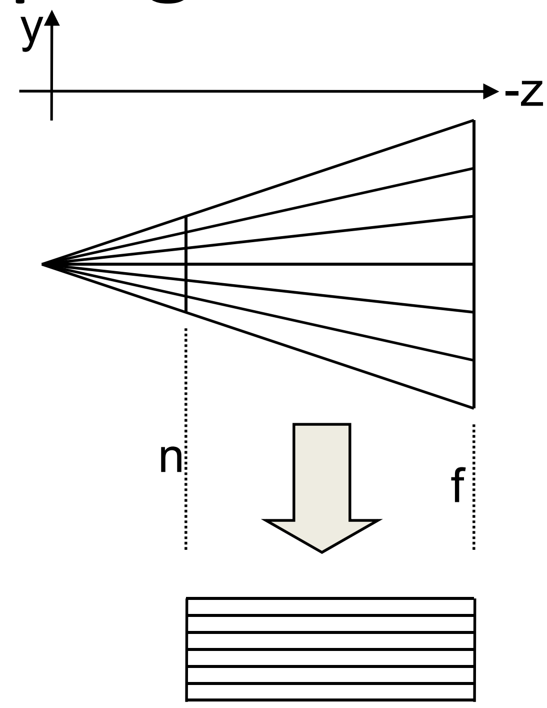


- We want to map all the lines through the center of projection to parallel lines
  - This converts the perspective case to the orthographic case, we can use all our existing methods
- The relative intersection points of lines with the near clip plane should not change
- The matrix that does this looks like the matrix for our simple perspective case

How to get this mapping?

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/n & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} (n/z)x \\ (n/z)y \\ n \\ 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 1/n & 0 \end{bmatrix}$$



$$M \begin{bmatrix} x \\ y \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ (An+B) \\ 1 \end{bmatrix} \quad (An+B) = n$$

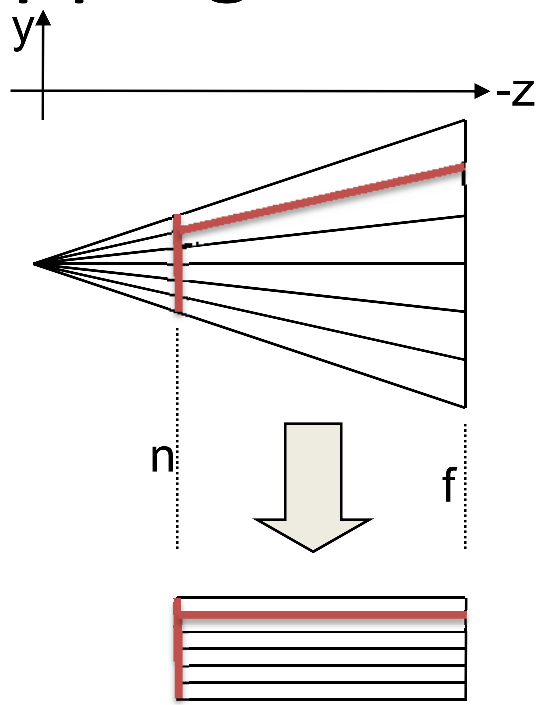
$$M \begin{bmatrix} x \\ y \\ f \\ 1 \end{bmatrix} = \begin{bmatrix} (n/f)x \\ (n/f)y \\ (n/f)(Af+B) \\ 1 \end{bmatrix} \quad (n/f)(Af+B) = f$$

$$\Rightarrow \quad A = 1 + \frac{f}{n}, \quad B = -f$$

Properties of this mapping

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/n & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} (n/z)x \\ (n/z)y \\ n \\ 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (n+f)/n & -f \\ 0 & 0 & 1/n & 0 \end{bmatrix}$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{nx}{z} \\ \frac{ny}{z} \\ f + n - \frac{fn}{z} \\ 1 \end{bmatrix}$$

If z = n, M(x,y,z,1) = [x,y,z,1]
near plane is unchanged

If x/z = c1, y/z = c2, then x'=n*c1, y'=n*c2
bending converging rays to parallel rays

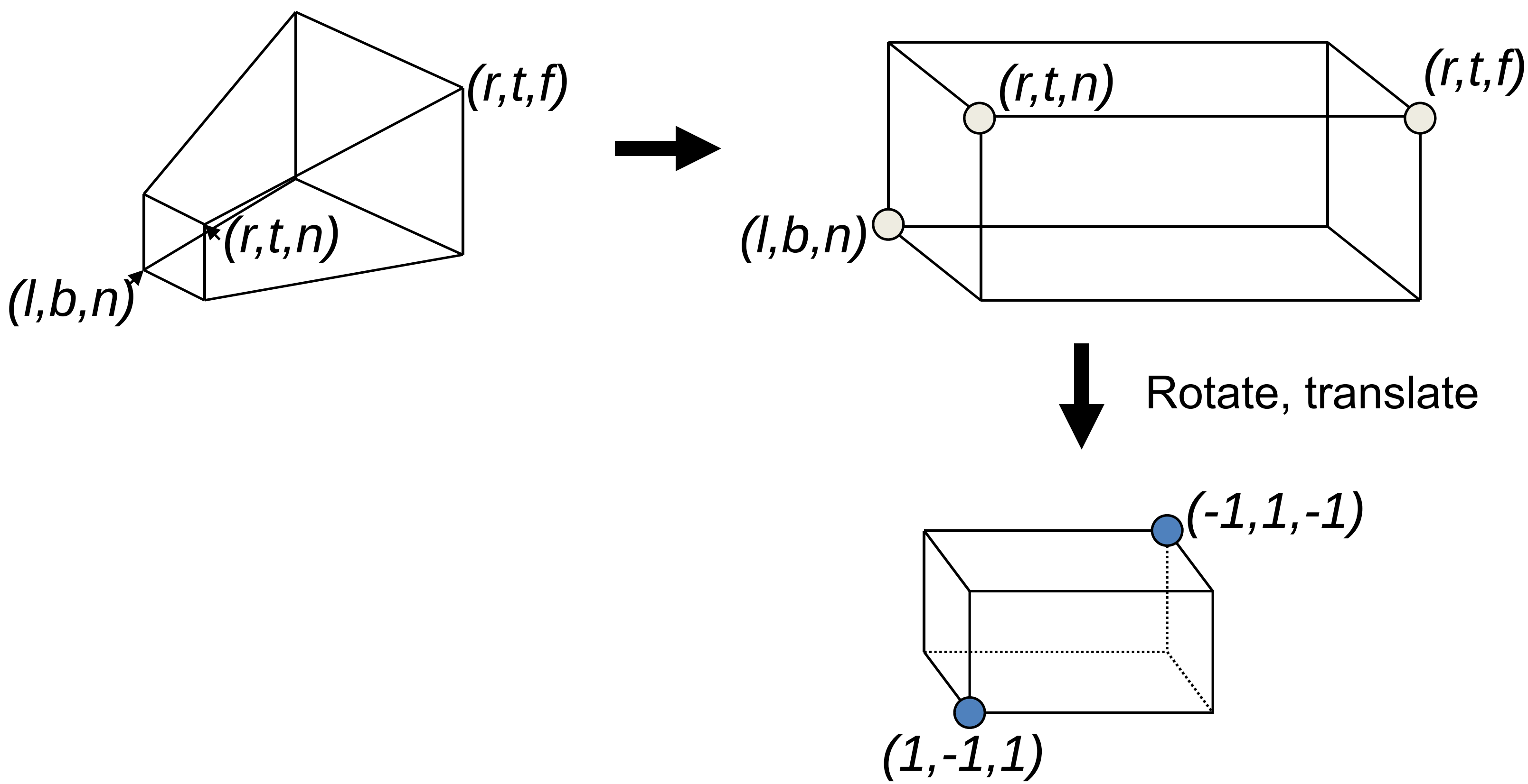
If z1 < z2, z1' < z2'
z ordering is preserved

General Perspective

$$\mathbf{M}_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (n+f)/n & -f \\ 0 & 0 & 1/n & 0 \end{bmatrix} \equiv \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

To Canonical view volume

- Perspective projection transforms a pyramid volume to an orthographic view volume



Canonical view volume [-1,1]^3

Orthographic View to Canonical Matrix

$$\begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} 2/(r-l) & 0 & 0 & 0 \\ 0 & 2/(t-b) & 0 & 0 \\ 0 & 0 & 2/(n-f) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -(r+l)/2 \\ 0 & 1 & 0 & -(t+b)/2 \\ 0 & 0 & 1 & -(n+f)/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic View to Canonical Matrix

$$\begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix} = \begin{bmatrix} 2/(r-l) & 0 & 0 & 0 \\ 0 & 2/(t-b) & 0 & 0 \\ 0 & 0 & 2/(n-f) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -(r+l)/2 \\ 0 & 1 & 0 & -(t+b)/2 \\ 0 & 0 & 1 & -(n+f)/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2/(r-l) & 0 & 0 & -(r+l)/(r-l) \\ 0 & 2/(t-b) & 0 & -(t+b)/(t-b) \\ 0 & 0 & 2/(n-f) & -(n+f)/(n-f) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \\ 1 \end{bmatrix}$$

$$\mathbf{x}_{canonical} = \mathbf{M}_{view->canonical}\mathbf{x}_{view}$$

Complete Perspective Projection

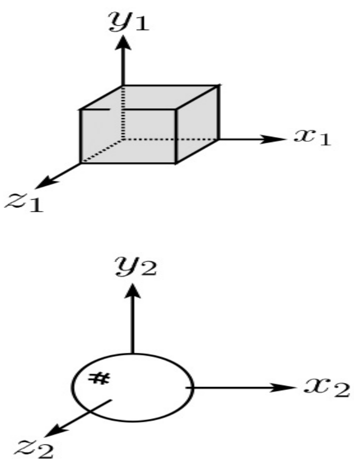
- After applying the perspective matrix, we map the orthographic view volume to the canonical view volume:

$$\mathbf{M}_{view->canonical} = \mathbf{M}_O \mathbf{M}_P = \begin{bmatrix} \frac{2}{(r-l)} & 0 & 0 & \frac{-(r+l)}{(r-l)} \\ 0 & \frac{2}{(t-b)} & 0 & \frac{-(t+b)}{(t-b)} \\ 0 & 0 & \frac{2}{(n-f)} & \frac{-(n+f)}{(n-f)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & (n+f) & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{M}_{world->canonical} = \mathbf{M}_{view->canonical} \mathbf{M}_{world->view}$$

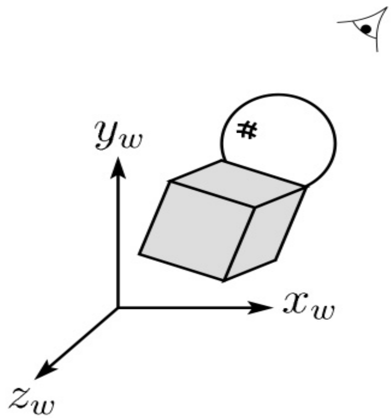
$$\mathbf{x}_{canonical} = \mathbf{M}_{world->canonical} \mathbf{x}_{world}$$

3D Geometry Pipeline



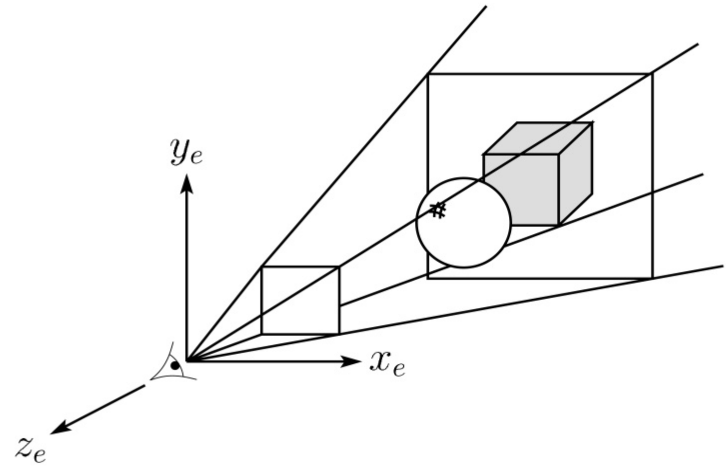
Model Space
(Object Space)

Rotation
Translation
Resizing



World Space

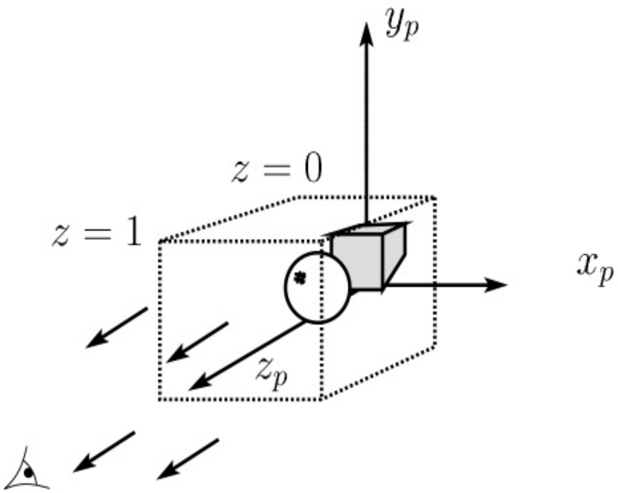
Rotation
Translation

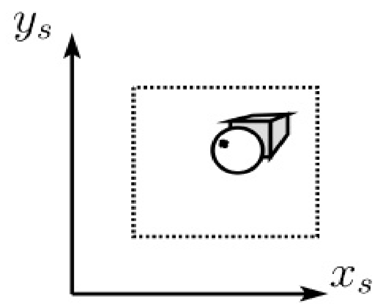


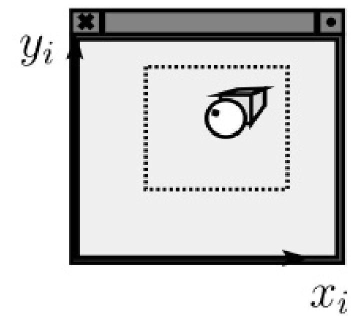
Eye Space
(View Space)

Projective,
Scale
Translation

3D Geometry Pipeline (cont’d)







Normalized Projection Space

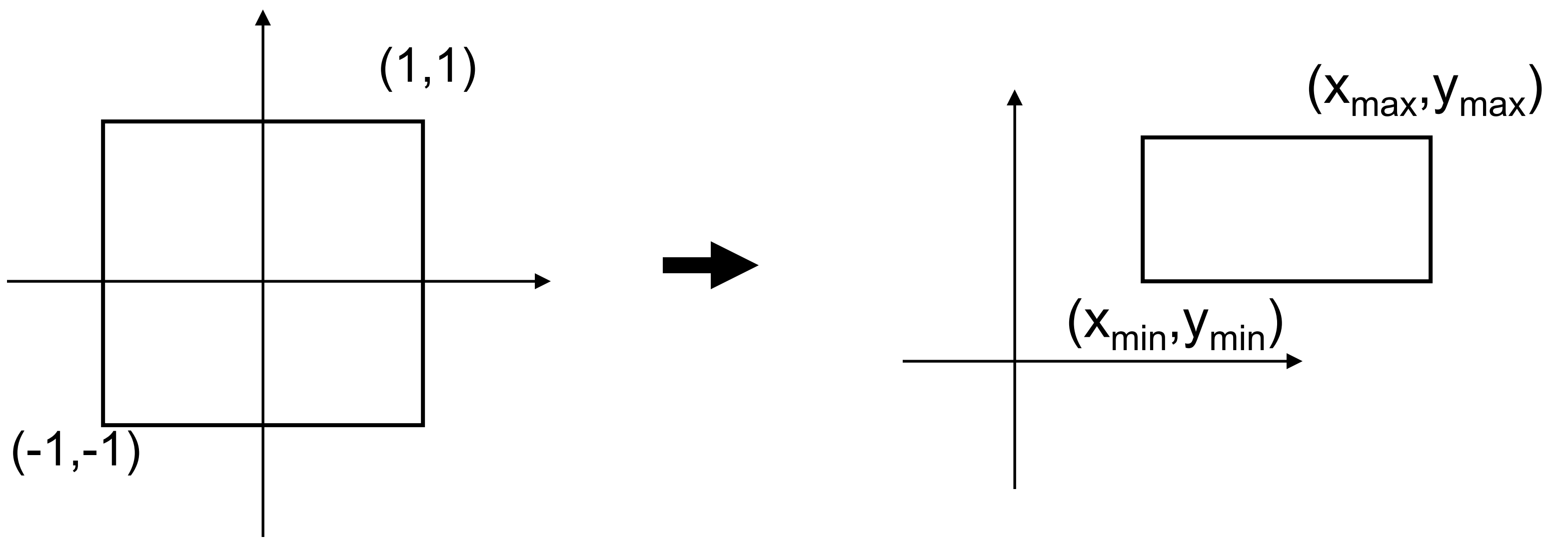
Project

Screen Space (2D)

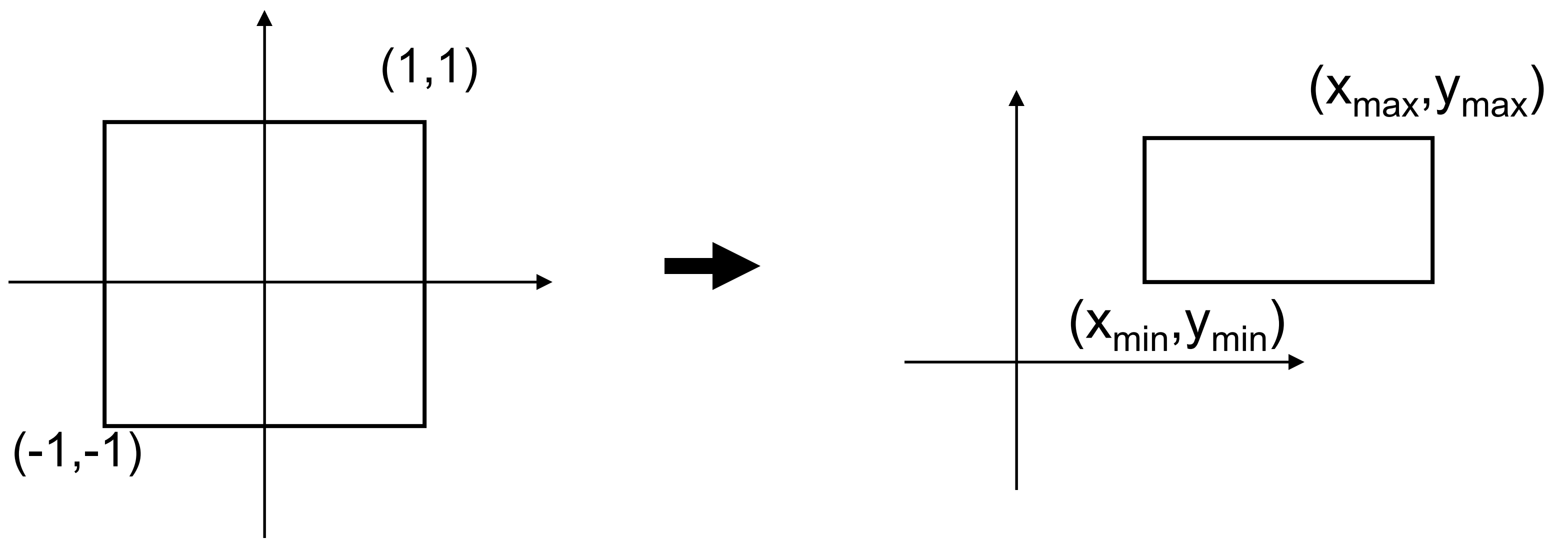
Scale
Translation

Image Space (pixels)
Raster Space

Canonical $\rightarrow$ Window Transform



Canonical → Window Transform



$$\begin{bmatrix} x_{pixel} \\ y_{pixel} \\ z_{pixel} \\ 1 \end{bmatrix} = \begin{bmatrix} (x_{\max} - x_{\min})/2 & 0 & 0 & (x_{\max} + x_{\min})/2 \\ 0 & (y_{\max} - y_{\min})/2 & 0 & (y_{\max} + y_{\min})/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{bmatrix}$$

Mapping of Z is nonlinear

$$\left(\frac{Az+B}{-z}\right) = -A - \frac{B}{z}$$

- Many mappings proposed: all have nonlinearities
- Advantage: handles range of depths (10cm – 100m)
- Disadvantage: depth resolution not uniform
- More close to near plane, less further away
- Common mistake: set near = 0, far = infty. Don’t do this. Can’t set near = 0; lose depth resolution.