CS-184: Computer Graphics

Lecture #5: Projection

Prof. James O'Brien
University of California, Berkeley

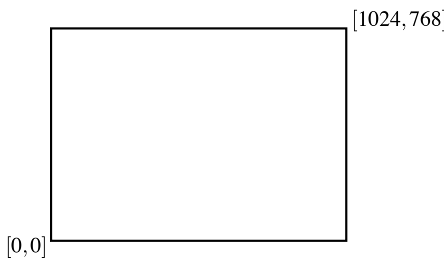
V2006-S-05-1.0

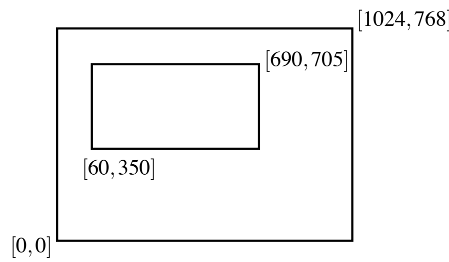
Today

- Windowing and Viewing Transformations
  - Windows and viewports
  - Orthographic projection
  - Perspective projection

Screen Space

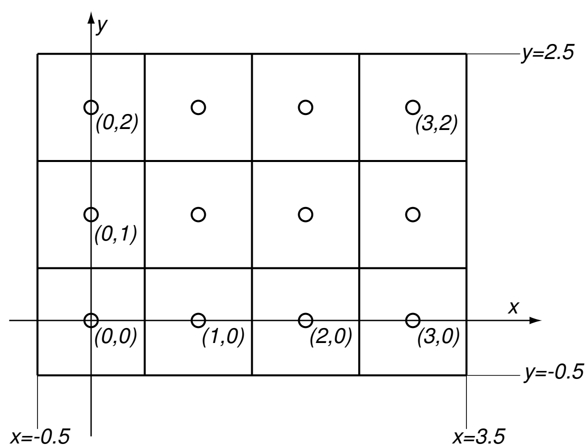
- Monitor has some number of pixels
  - e.g. 1024 x 768
- Some sub-region used for given program
  - You call it a window
  - Let's call it a viewport instead





Screen Space

- May not really be a "screen"
  - Image file
  - Printer
  - Other
- Little pixel details
- Sometimes odd
  - Upside down
  - Hexagonal



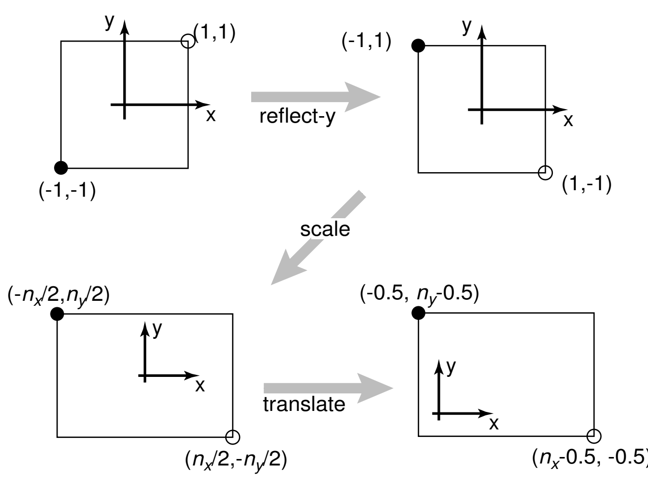
From Shirley textbook.

Screen Space

- Viewport is somewhere on screen
  - You probably don't care where
  - Window System likely manages this detail
  - Sometimes you care exactly where
- Viewport has a size in pixels
  - Sometimes you care (images, text, *etc.*)
  - Sometimes you don't (using high-level library)

Canonical View Space

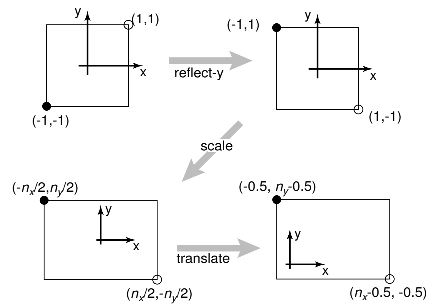
- Canonical view region
  - 2D: [-1,-1] to [+1,+1]



From Shirley textbook.

Canonical View Space

- Canonical view region
  - 2D: [-1,-1] to [+1,+1]



From Shirley textbook.

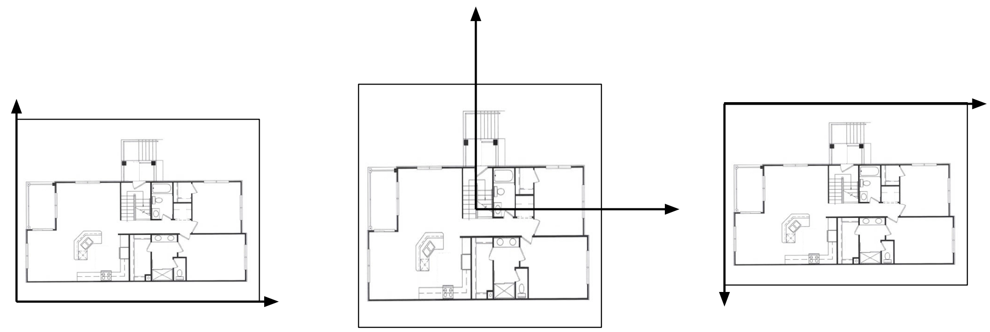
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{-n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Canonical View Space

- Canonical view region
  - 2D: [-1,-1] to [+1,+1]
- Define arbitrary *window* and define objects
- Transform window to canonical region
- Do other things (we'll see clipping latter)
- Transform canonical to screen space
- Draw it.

From Shirley textbook.

Canonical View Space



World Coordinates (Meters) | Canonical | Screen Space (Pixels)

Note distortion issues...

Projection

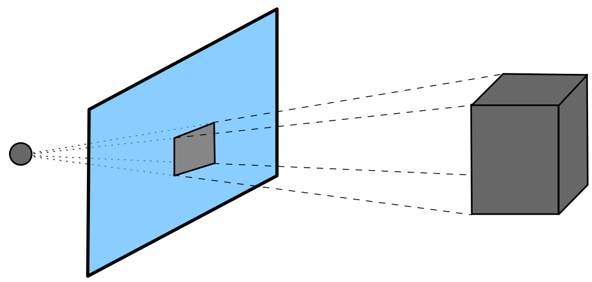
- Process of going from 3D to 2D
- Studies throughout history (e.g. painters)
- Different types of projection
  - Linear
    - Orthographic
    - Perspective
  - Nonlinear

Many special cases in books just one of these two...

Orthographic is special case of perspective...

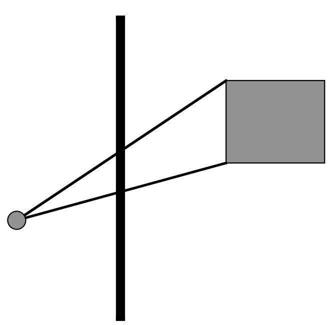
Linear Projection

- Projection onto a planar surface
- Projection directions either
  - Converge to a point
  - Are parallel (converge at infinity)

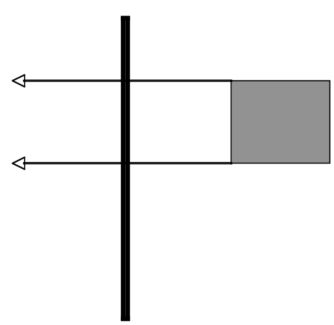


Linear Projection

- A 2D view

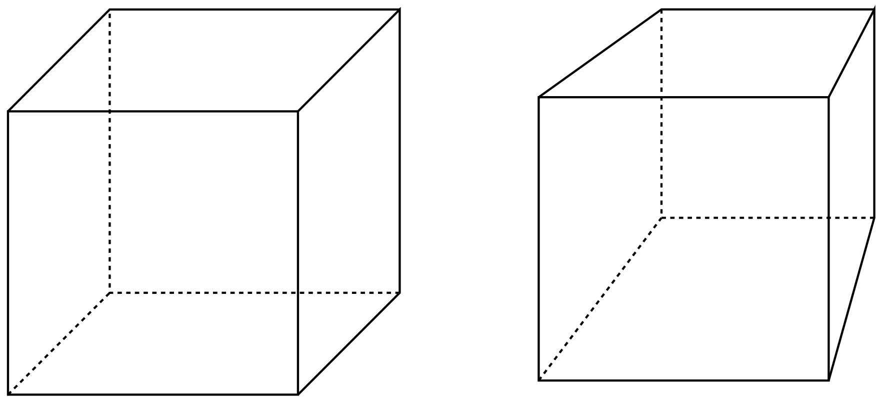


Perspective



Orthographic

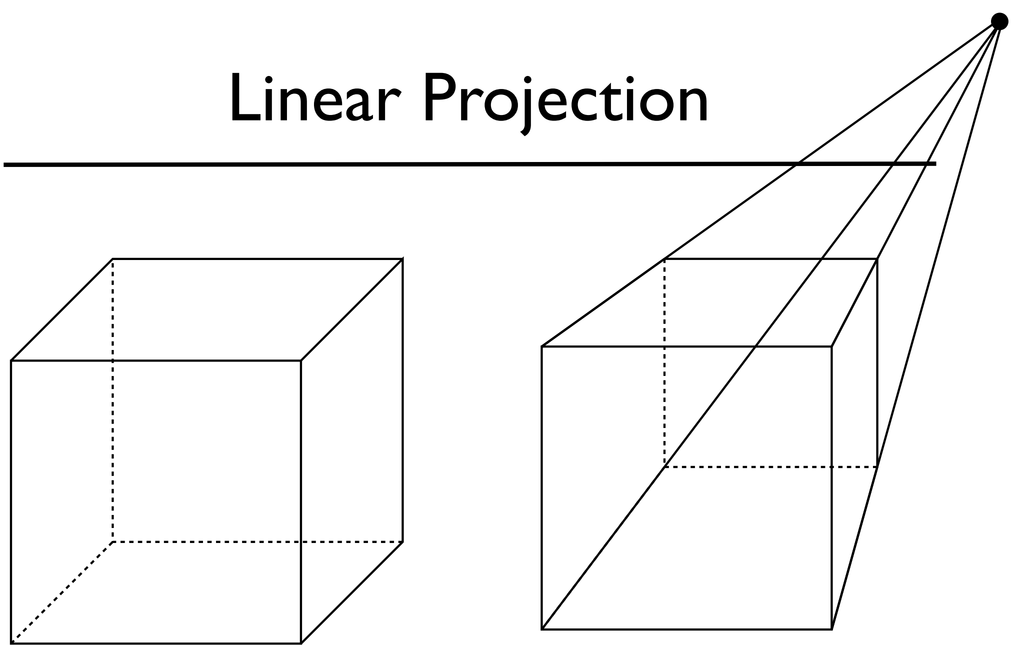
Linear Projection



Orthographic

Perspective

Linear Projection

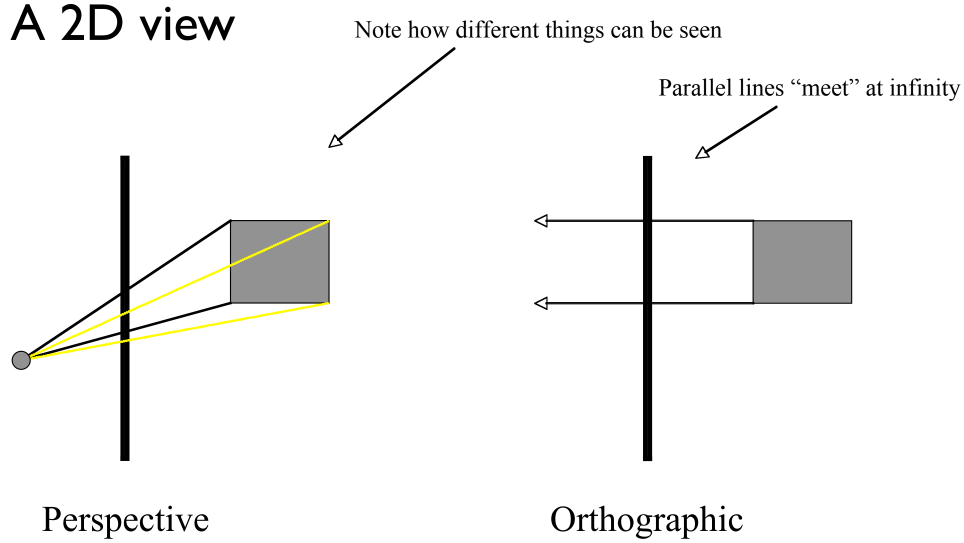


Orthographic

Perspective

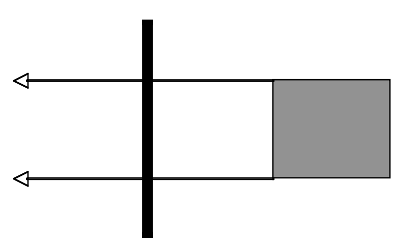
Linear Projection

- A 2D view



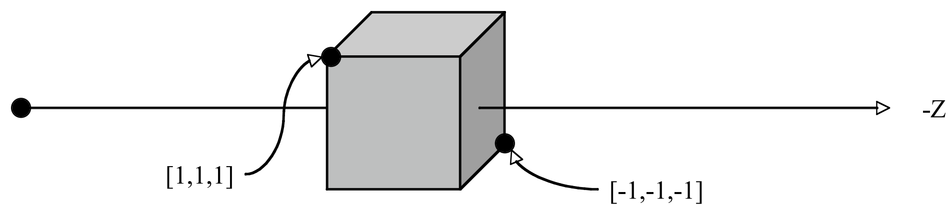
Orthographic Projection

- No foreshortening
- Parallel lines stay parallel
- Poor depth cues



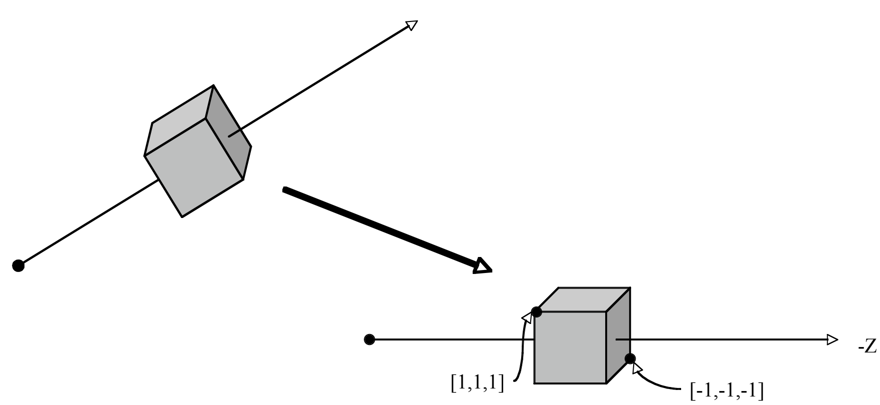
Canonical View Space

- Canonical view region
  - 3D: [-1,-1,-1] to [+1,+1,+1]
- Assume looking down *-Z* axis
  - Recall that "Z is in your face"

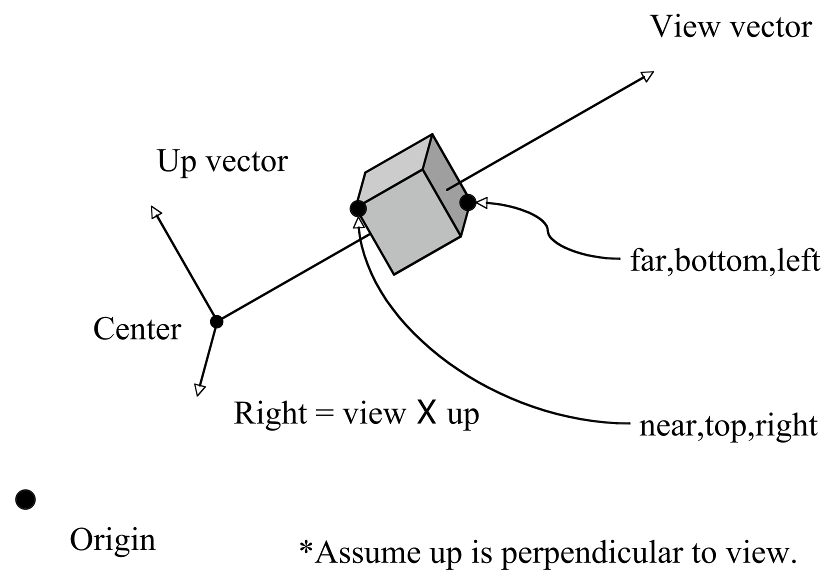


Orthographic Projection

- Convert arbitrary view volume to canonical



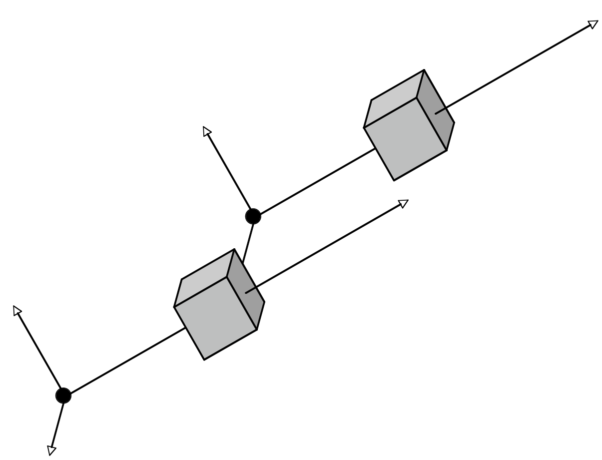
Orthographic Projection



*Assume up is perpendicular to view.

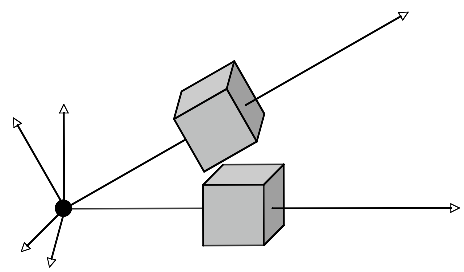
Orthographic Projection

- Step 1: translate center to origin



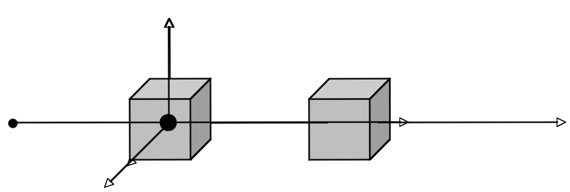
Orthographic Projection

- Step 1: translate center to origin
- Step 2: rotate *view* to **-Z** and *up* to **+Y**



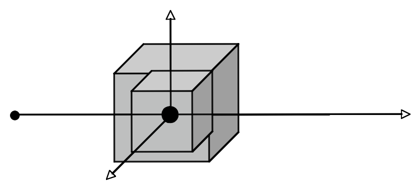
Orthographic Projection

- Step 1: translate center to origin
- Step 2: rotate *view* to **-Z** and *up* to **+Y**
- Step 3: center view volume



Orthographic Projection

- Step 1: translate center to origin
- Step 2: rotate *view* to **-Z** and *up* to **+Y**
- Step 3: center view volume
- Step 4: scale to canonical size

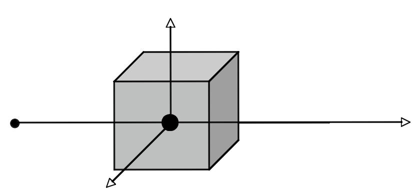


Orthographic Projection

- Step 1: translate center to origin
- Step 2: rotate *view* to **-Z** and *up* to **+Y**
- Step 3: center view volume
- Step 4: scale to canonical size

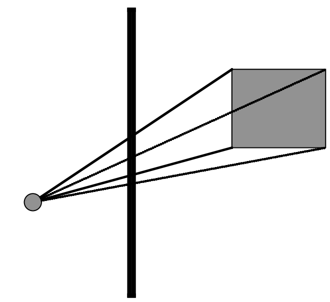
$$\mathbf{M} = \underline{\mathbf{S}\cdot\mathbf{T}_2}\cdot\underline{\mathbf{R}\cdot\mathbf{T}_1}$$

$$\mathbf{M} = \mathbf{M}_o \cdot \mathbf{M}_v$$



Perspective Projection

- Foreshortening: further objects appear smaller
- Some parallel line stay parallel, most don't
- Lines still look like lines
- $\mathbf{Z}$ ordering preserved (where we care)



Perspective Projection

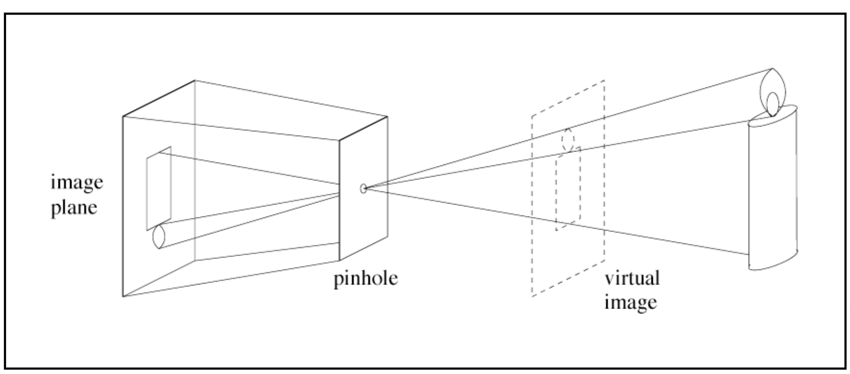


Image from D. Forsyth

Pinhole *a.k.a* center of projection

Perspective Projection

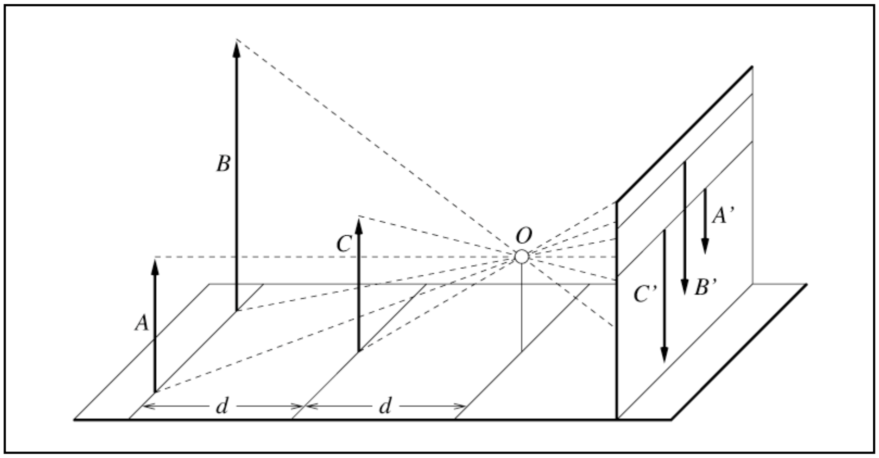
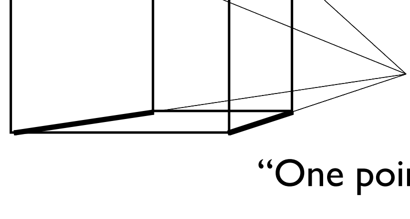


Image from D. Forsyth

Foreshortening: distant objects appear smaller

Perspective Projection

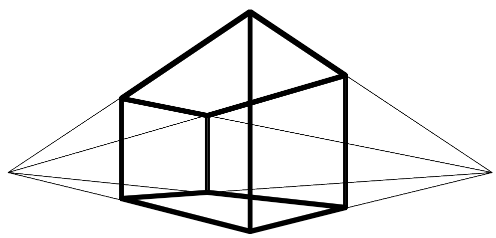
- Vanishing points
  - Depend on the scene
  - Not intrinsic to camera



"One point perspective"

Perspective Projection

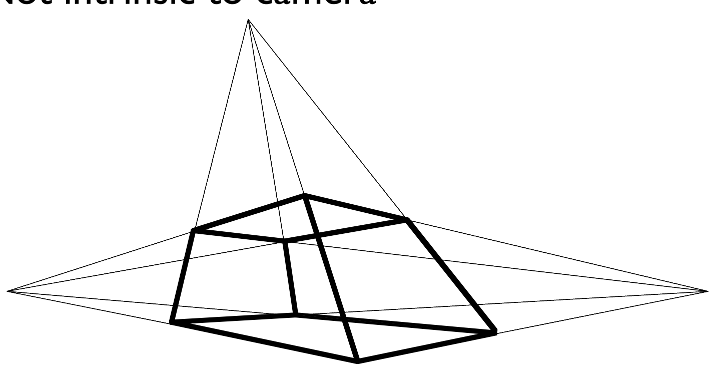
- Vanishing points
  - Depend on the scene
  - Nor intrinsic to camera



"Two point perspective"

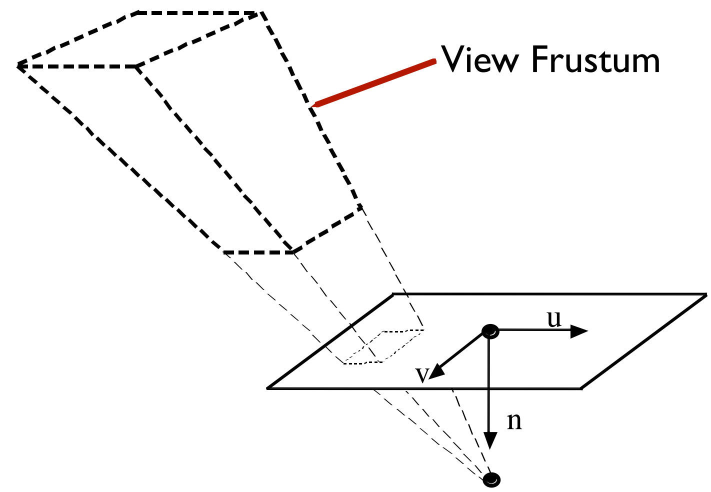
Perspective Projection

- Vanishing points
  - Depend on the scene
  - Not intrinsic to camera

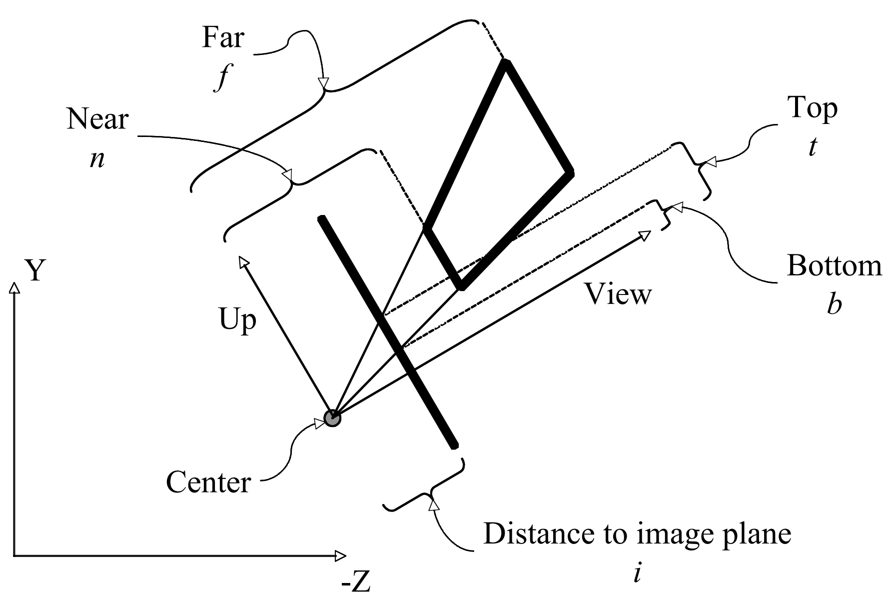


"Three point perspective"

Perspective Projection

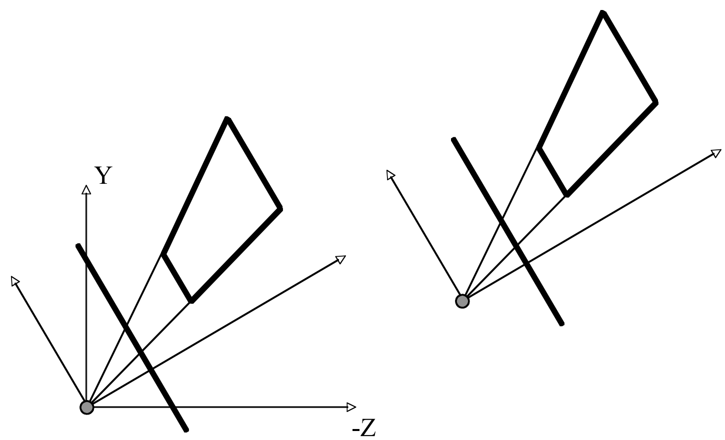


Perspective Projection



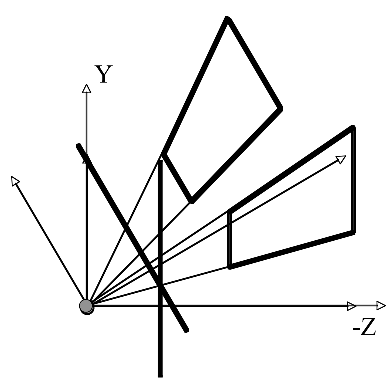
Perspective Projection

- Step 1: Translate *center* to origin



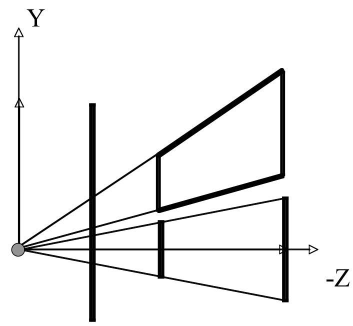
Perspective Projection

- Step 1: Translate *center* to origin
- Step 2: Rotate *view* to **-Z**, *up* to **+Y**



Perspective Projection

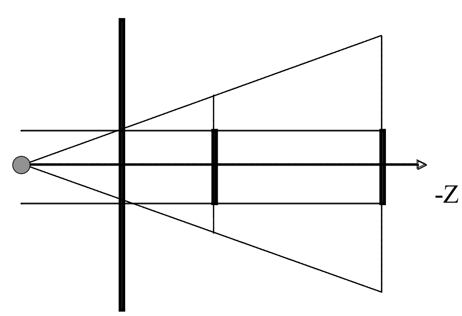
- Step 1: Translate *center* to origin
- Step 2: Rotate *view* to **-Z**, *up* to **+Y**
- Step 3: Shear center-line to **-Z** axis



Perspective Projection

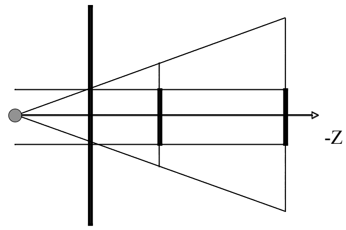
- Step 1: Translate *center* to origin
- Step 2: Rotate *view* to **-Z**, *up* to **+Y**
- Step 3: Shear center-line to **-Z** axis
- Step 4: Perspective

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{i+f}{i} & f \\ 0 & 0 & \frac{-1}{i} & 0 \end{bmatrix}$$



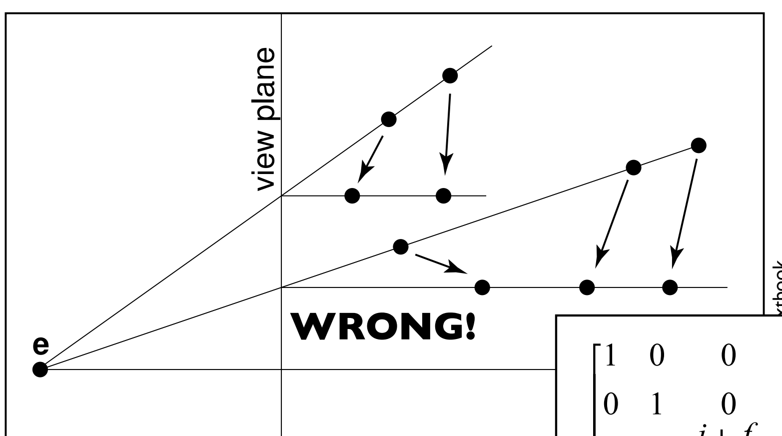
Perspective Projection

- Step 4: Perspective
  - Points at z=-i stay at z=-i
  - Points at z=-f stay at z=-f
  - Points at z=0 goto z=$\pm\infty$
  - Points at z=-∞ goto z=-$(i+f)$
  - x and y values divided by -z/i
  - Straight lines stay straight
  - Depth ordering preserved in [-i,-f]
  - Movement along lines distorted



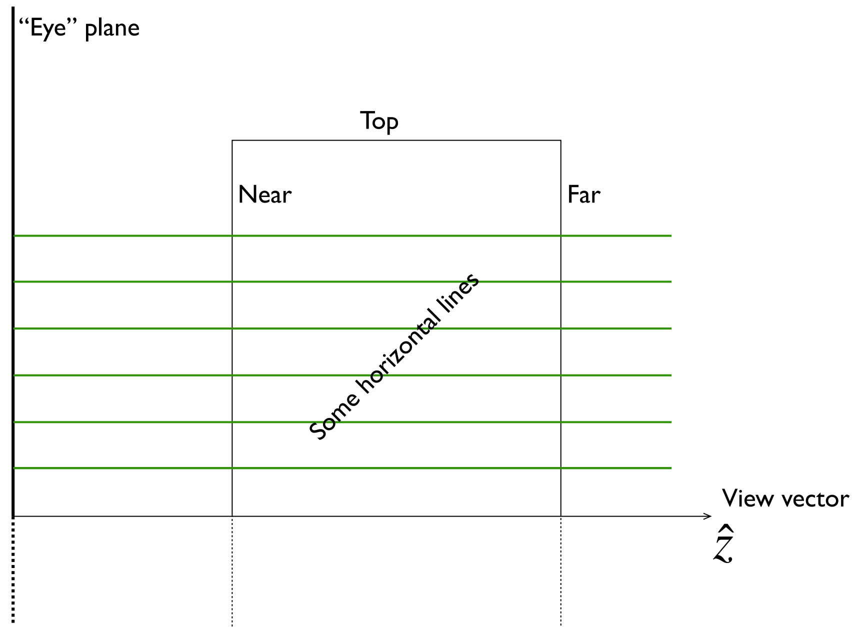
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{i+f}{i} & f \\ 0 & 0 & \frac{-1}{i} & 0 \end{bmatrix}$$

Perspective Projection



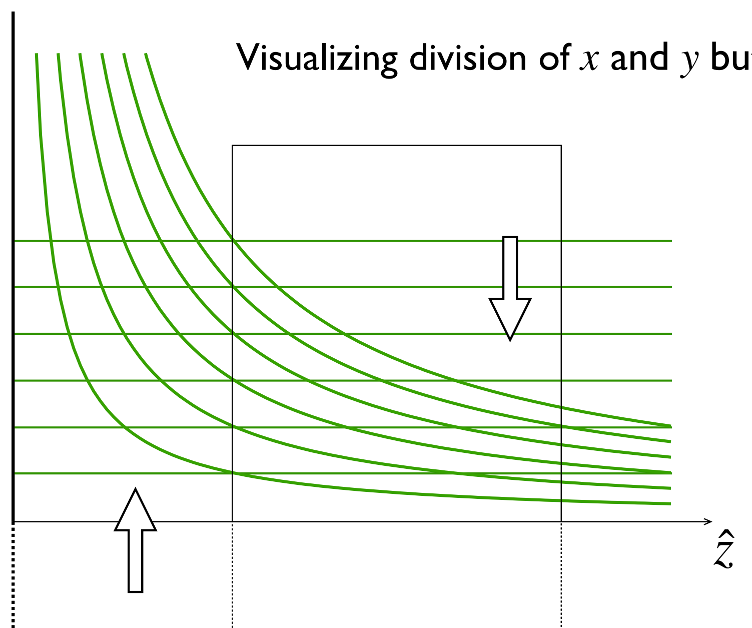
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{i+f}{i} & f \\ 0 & 0 & \frac{-1}{i} & 0 \end{bmatrix}$$

Perspective Projection

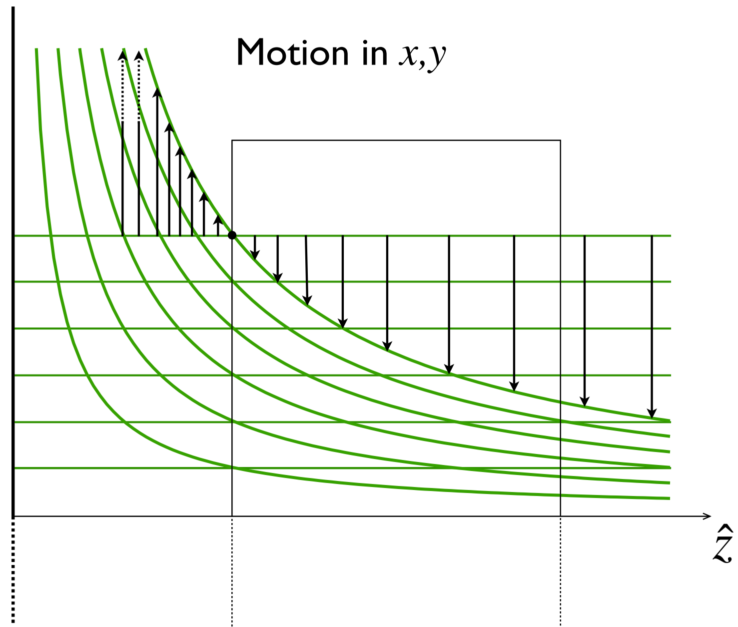


Perspective Projection

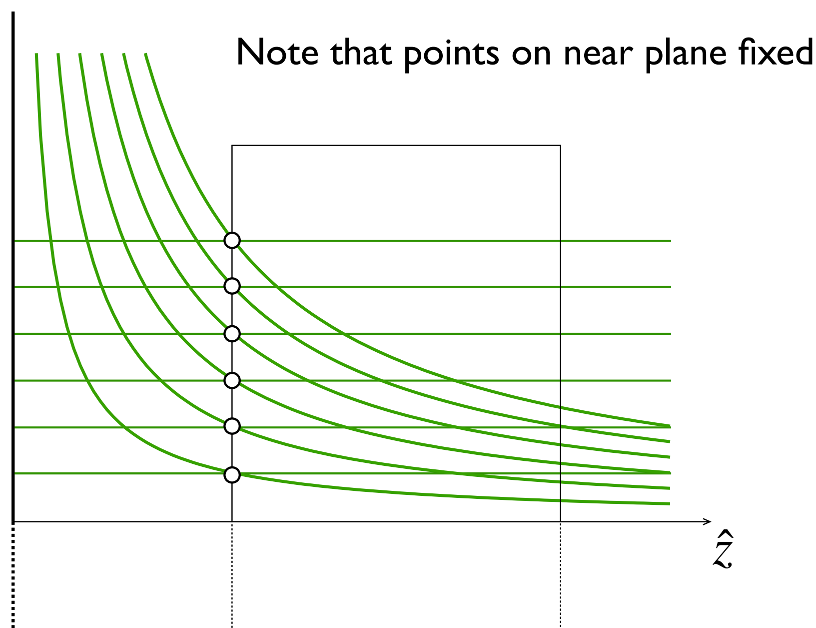
Visualizing division of x and y but not z



Perspective Projection

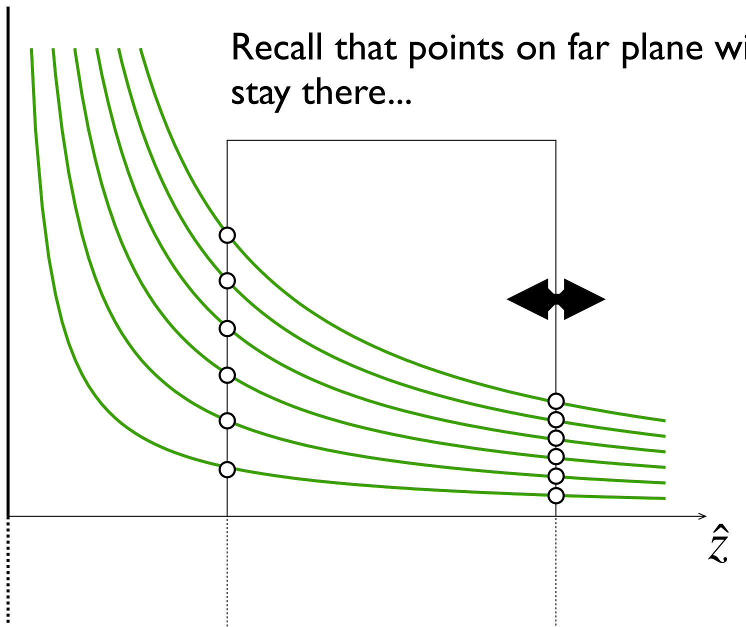


Perspective Projection



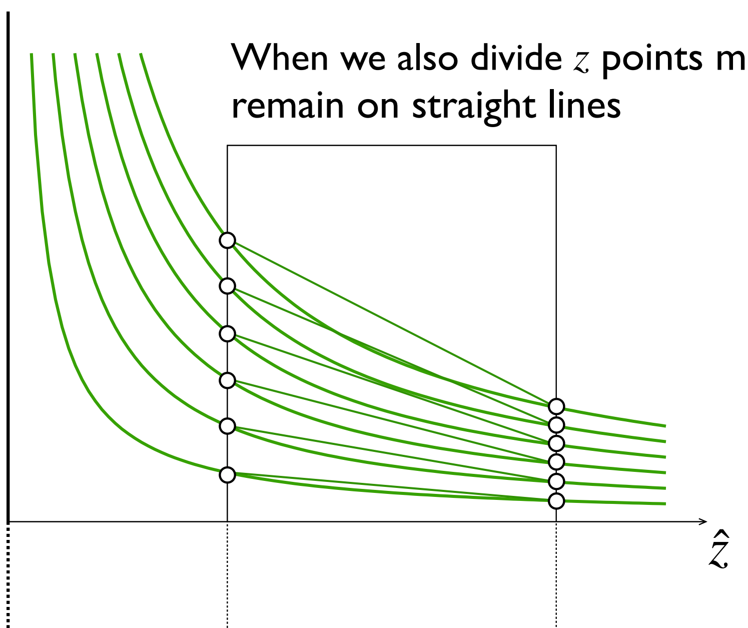
Perspective Projection

Recall that points on far plane will stay there...



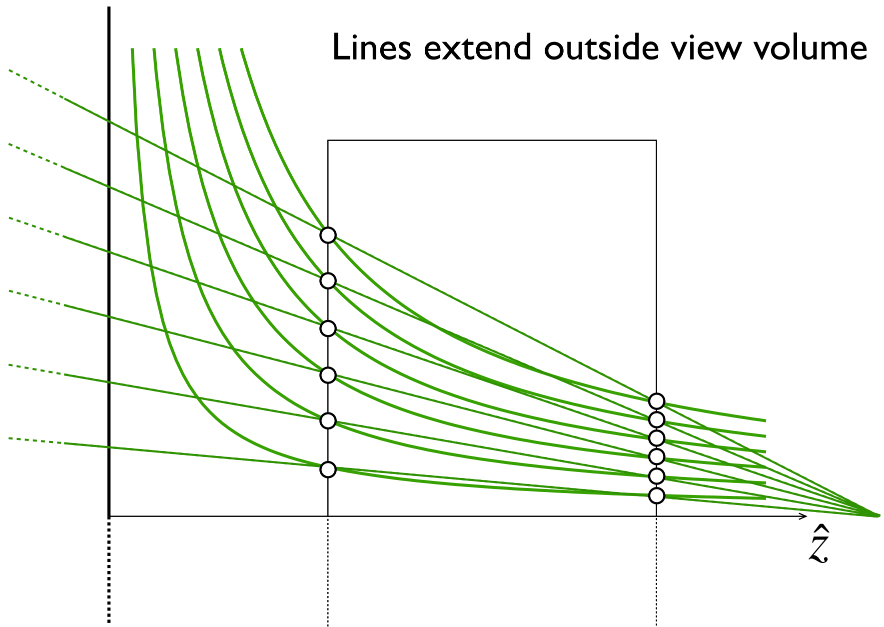
Perspective Projection

When we also divide z points must remain on straight lines



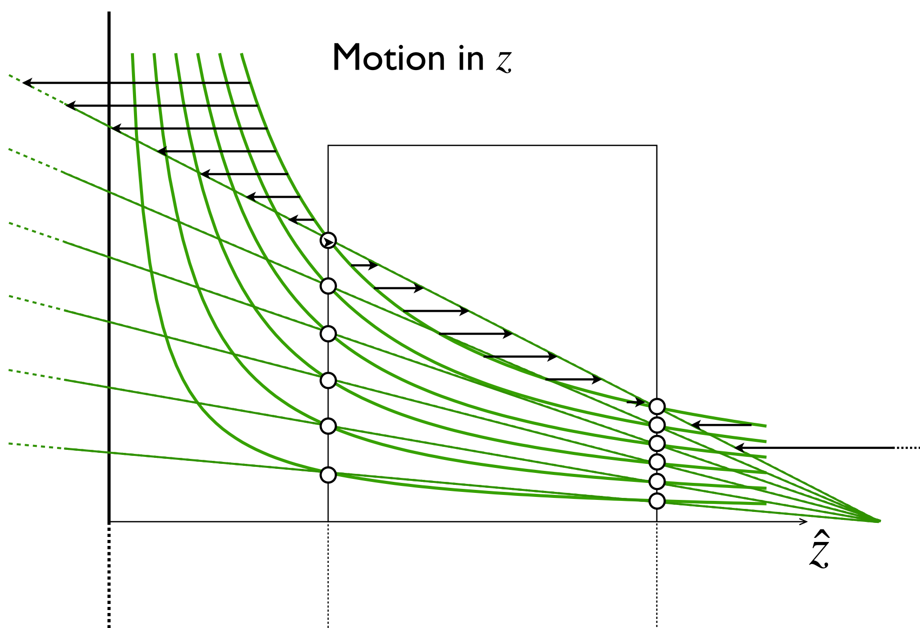
Perspective Projection

Lines extend outside view volume

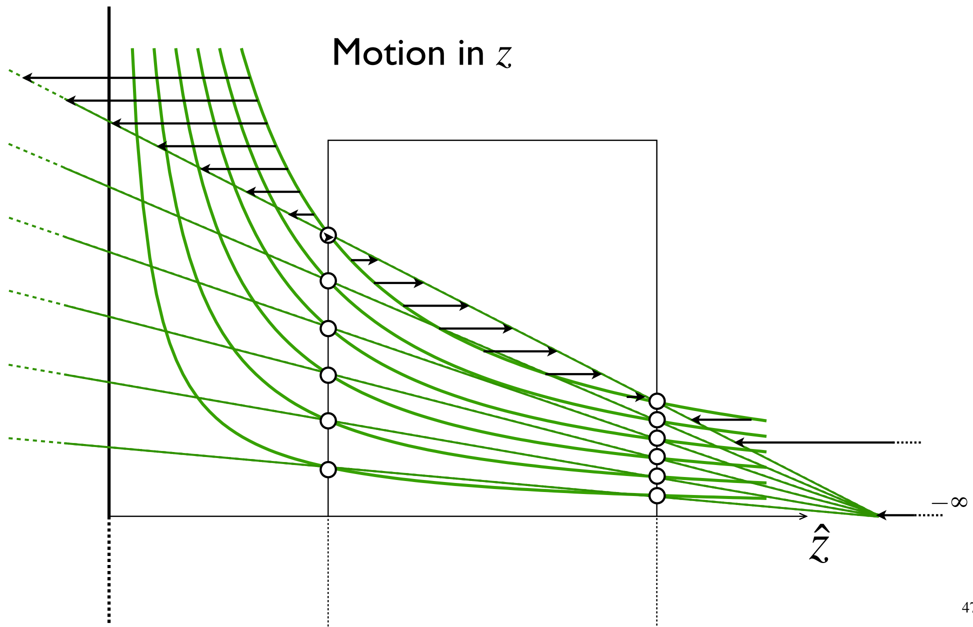


Perspective Projection

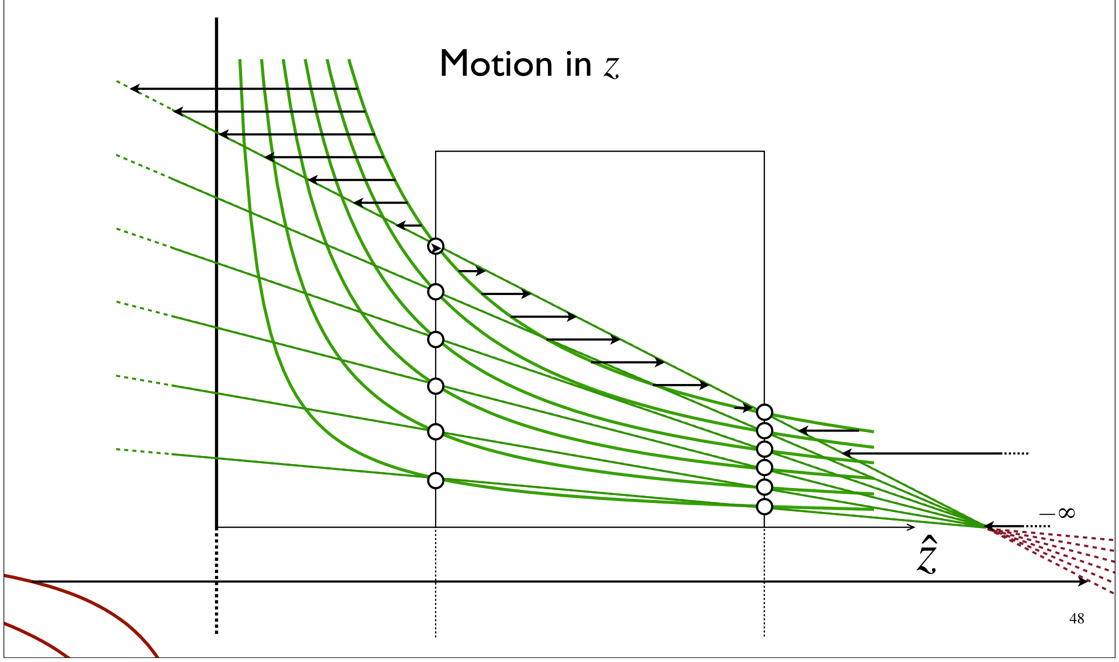
Motion in z



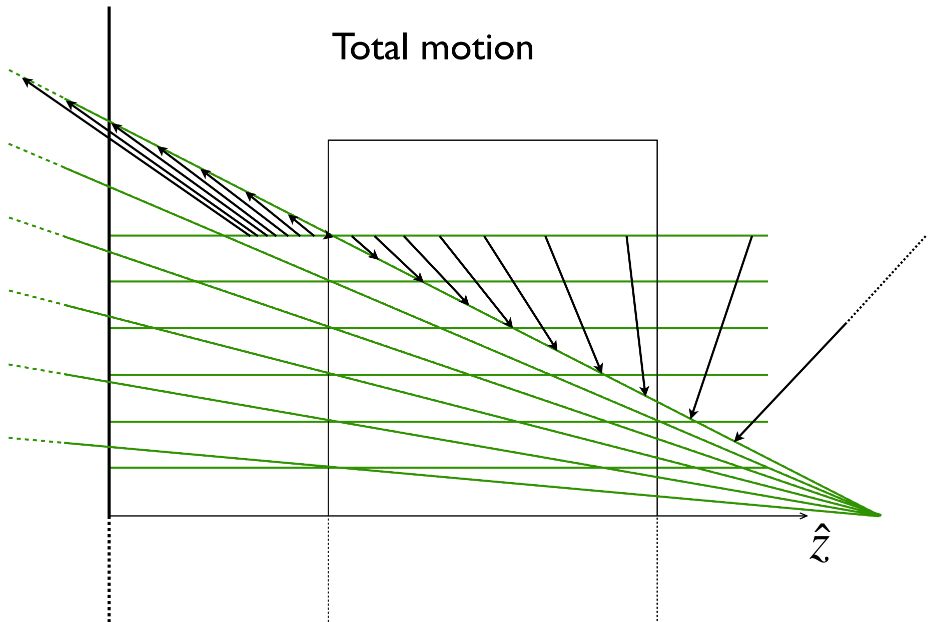
Perspective Projection



Perspective Projection

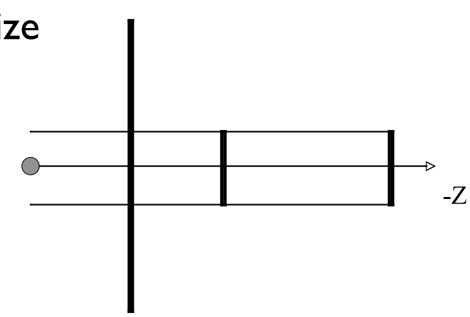


Perspective Projection



Perspective Projection

- Step 1: Translate *center* to orange
- Step 2: Rotate *view* to **-Z**, *up* to **+Y**
- Step 3: Shear center-line to **-Z** axis
- Step 4: Perspective
- Step 5: center view volume
- Step 6: scale to canonical size



Perspective Projection

- Step 1: Translate *center* to orange
- Step 2: Rotate *view* to **-Z**, *up* to **+Y**

$\} \mathbf{M}_v$

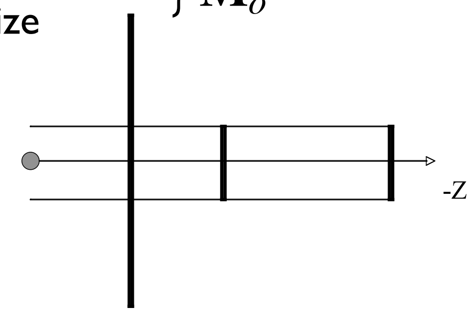
- Step 3: Shear center-line to **-Z** axis
- Step 4: Perspective

$\} \mathbf{M}_p$

- Step 5: center view volume
- Step 6: scale to canonical size

$\} \mathbf{M}_o$

$$\mathbf{M} = \mathbf{M}_o \cdot \mathbf{M}_p \cdot \mathbf{M}_v$$



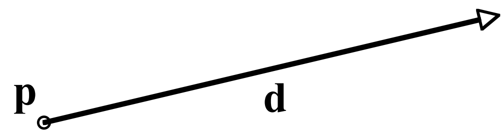
Perspective Projection

- There are other ways to set up the projection matrix
  - View plane at $z=0$ zero
  - Looking down another axis
  - *etc...*
- Functionally equivalent

Vanishing Points

- Consider a ray:

$$\mathbf{r}(t) = \mathbf{p} + t\,\mathbf{d}$$



Vanishing Points

- Ignore $\mathbf{Z}$ part of matrix
- $\mathbf{X}$ and $\mathbf{Y}$ will give location in image plane
- Assume image plane at z=-i

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ & \text{whatever} & & \\ 0 & 0 & -1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} I_x \\ I_y \\ I_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Vanishing Points

$$\begin{bmatrix} I_x \\ I_y \\ I_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z \end{bmatrix}$$

$$\begin{bmatrix} I_x / I_w \\ I_y / I_w \end{bmatrix} = \begin{bmatrix} -x/z \\ -y/z \end{bmatrix}$$

Vanishing Points

- Assume $d_z = -1$

$$\begin{bmatrix} I_x / I_w \\ I_y / I_w \end{bmatrix} = \begin{bmatrix} -x/z \\ -y/z \end{bmatrix} = \begin{bmatrix} \dfrac{p_x + td_x}{-p_z + t} \\ \dfrac{p_y + td_y}{-p_z + t} \end{bmatrix}$$

$$\lim_{t \to \pm\infty} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Vanishing Points

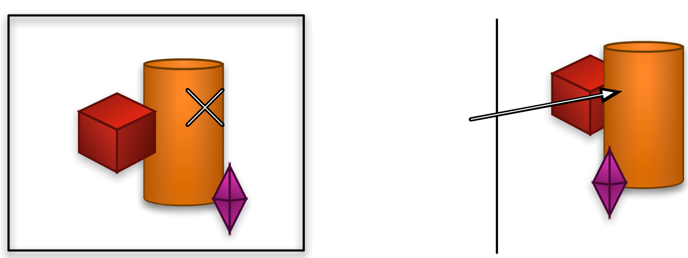
$$\lim_{t \to \pm\infty} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- All lines in direction $\mathbf{d}$ converge to same point in the image plane -- the vanishing point
- Every point in plane is a v.p. for some set of lines
- Lines parallel to image plane ($d_z = 0$) vanish at infinity

What's a horizon?

Ray Picking

- Pick object by picking point on screen



- Compute ray from pixel coordinates.

Ray Picking

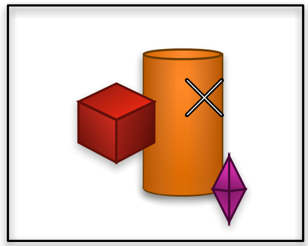
- Transform from World to Screen is:

$$\begin{bmatrix} I_x \\ I_y \\ I_z \\ I_w \end{bmatrix} = \mathbf{M} \begin{bmatrix} W_x \\ W_y \\ W_z \\ W_w \end{bmatrix}$$

- Inverse:

$$\begin{bmatrix} W_x \\ W_y \\ W_z \\ W_w \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} I_x \\ I_y \\ I_z \\ I_w \end{bmatrix}$$

- What $\mathbf{Z}$ value?



Ray Picking

- Recall that:
  - Points at z=-i stay at z=-i
  - Points at z=-f stay at z=-f

Depends on screen details, YMMV
General idea should translate...

$$\mathbf{r}(t) = \mathbf{p} + t\,\mathbf{d}$$

$$\mathbf{r}(t) = \mathbf{a}_w + t(\mathbf{b}_w - \mathbf{a}_w)$$

$$\mathbf{a}_s = [s_x, s_y, -i]$$

$$\mathbf{b}_s = [s_x, s_y, -f]$$