Projection and 3D Transformations

CS-184: Computer Graphics

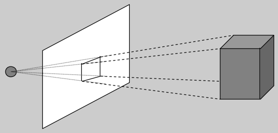
Prof. James O’Brien

Projection

- Process of going from 3D scene to 2D scene
- Studied throughout history (*e.g.* painters)
- Different types of projection
  - Linear
    - Orthographic
    - Perspective
  - Nonlinear
- Many other “types” of linear mentioned in books
  - Just special cases of orthographic or perspective

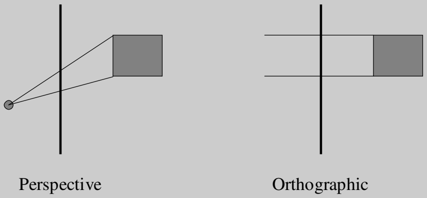
Linear Projection

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



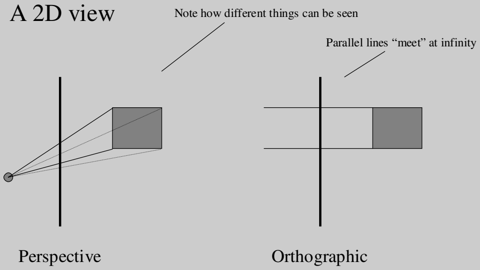
Linear Projection

- A 2D view



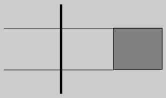
Linear Projection

- A 2D view


Note how different things can be seen

Parallel lines “meet” at infinity

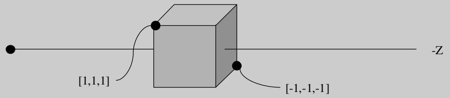
Orthographic Projection



- No foreshortening
- Parallel lines stay parallel
- Examples:

Orthographic Projection

- Assume looking down –Z axis
  - “Z is in your face”
- View center at the origin
- View region is box defined by [-1,-1,-1] and [1,1,1]



-Z

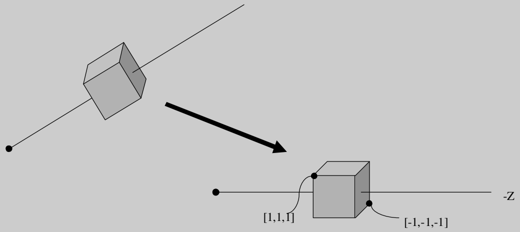
[1,1,1]

[-1,-1,-1]

Throw X and Y coordinates map to normalized view port

Orthographic Projection

- Converting to canonical view setup

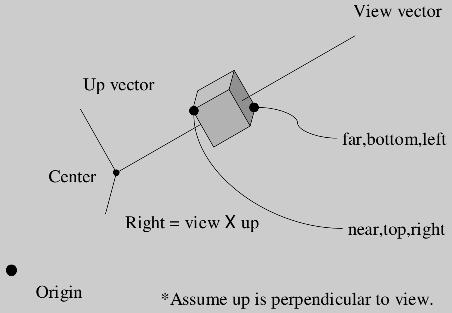


-Z

[1,1,1]

[-1,-1,-1]

Orthographic Projection



View vector

Up vector

far,bottom,left

Center

Right = view X up

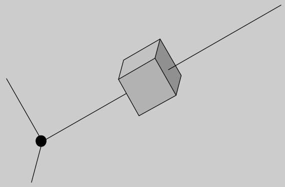
near,top,right

Origin

*Assume up is perpendicular to view.

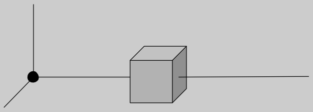
Orthographic Projection

Step 1: translate center to origin



Orthographic Projection

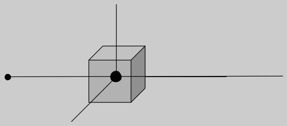
Step 2: Rotate so that view aligns with –Z axis and up with +Y axis



Orthographic Projection

Step 3: Center view volume

Step 4: Scale view volume



Orthographic Projection

Step 1: translate center to origin

Step 2: Rotate so that view aligns with –Z axis and up with +Y axis

Step 3: Center view volume

Step 4: Scale view volume

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

Orthographic Projection

Step 1: translate center to origin

Step 2: Rotate so that view aligns with –Z axis and up with +Y axis

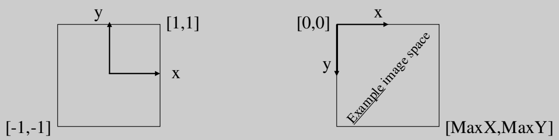
Step 3: Center view volume

Step 4: Scale view volume

$$\mathbf{M} = \overbrace{\mathbf{S}\mathbf{T}_2}^{\mathbf{M}_\mathbf{o}}\underbrace{\mathbf{R}\mathbf{T}_1}_{\mathbf{M}_\mathbf{v}}$$

Window Transformation

- Convert from [-1,-1],[+1,+1] window region to image space



Pixel centers offset by 0.5 (*e.g.* 0.5, 1.5, 2.5 … MaxX-0.5)

$\mathbf{M}_\mathbf{w}$ = Translate and scale

Detour: 3D Transformations

- With the exception of rotations, basically the same as in 2D:

2D

$$\tilde{\mathbf{A}} = \begin{bmatrix} & & t_x \\ \mathbf{A} & & \\ & & t_y \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{A} \text{ is } 2\times 2$$

Detour: 3D Transformations

- With the exception of rotations, basically the same as in 2D:

3D

$$\tilde{\mathbf{A}} = \begin{bmatrix} & & & t_x \\ & \mathbf{A} & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A} \text{ is } 3\times 3$$

Detour: 3D Transformations

- Axis-aligned scales are still diagonal
- Rotations still orthonormal w/ Det = +1
- Shear is a composition of rotation and scale
- SVD and polar decomposition have the same properties

BUT:

- More than one way to rotate
- Can rotate about any axis in space
- 3 DOF for rotation, not just 1

Detour: 3D Rotations

- 2D implicitly rotating about axis “out of the page”

$$\begin{bmatrix} \text{Cos}(\theta) & -\text{Sin}(\theta) \\ \text{Sin}(\theta) & \text{Cos}(\theta) \end{bmatrix}$$

Detour: 3D Rotations

- In 3D can rotate about one of coordinate axes

Example: rotation about *X* axis. (Other axes similar, see text.)

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \text{Cos}(\theta) & -\text{Sin}(\theta) \\ 0 & \text{Sin}(\theta) & \text{Cos}(\theta) \end{bmatrix}$$

- Or about arbitrary axis (we’ll see shortly…)

Detour: 3D Rotations

Rotation Matrix Trivia:

- AKA direction-cosine matrices
- Orthonormal
- Det = +1
- One real eigenvalue = 1
- Corresponding eigenvector is axis of rotation
- Unique

Detour: 3D Rotations

Euler Angles

- Any rotation can be composed of one rotations about each of the primary axes

$$\mathbf{R} = \mathbf{R_z R_y R_x}$$

- Allows tumbling
- Suffers from gimbal-lock
- Non-unique

Detour: 3D Rotations

Angular Displacement

- AKA: exponential map, axis-angle
- Rotate θ degrees about axis
- θ is given by the length of the vector

Detour: 3D Rotations

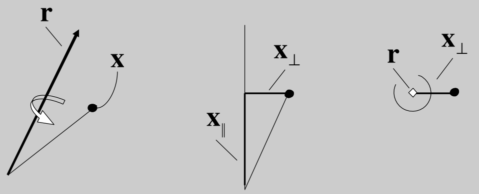
Angular Displacement

- **Method 1** to arrive at rotation matrix
  1. Rotate axis about X axis into X-Y plane
  2. Rotate axis about Z axis to align with X axis
  3. Rotate θ about X axis
  4. Undo step 2
  5. Undo step 3
  - Composite 5.4.3.2.1 together

Detour: 3D Rotations

Angular Displacement

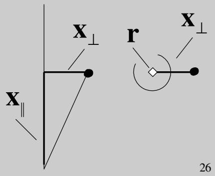
- **Method 2** to arrive at rotation matrix



Detour: 3D Rotations

Angular Displacement

- **Method 2** to arrive at rotation matrix
  - $\mathbf{x}_\parallel$ does not change
  - $\mathbf{x}_\perp$ rotates like 2D rotation



Detour: 3D Rotations

Angular Displacement

- **Method 2** to arrive at rotation matrix

$$\begin{aligned}\boldsymbol{x}' &= \hat{\boldsymbol{r}}(\hat{\boldsymbol{r}} \cdot \boldsymbol{x}) \\ &+ \sin(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times \boldsymbol{x}) \\ &- \cos(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times (\hat{\boldsymbol{r}} \times \boldsymbol{x}))\end{aligned}$$

Detour: 3D Rotations

Angular Displacement

- **Method 2** to arrive at rotation matrix

$$\begin{aligned}\boldsymbol{x}' &= \hat{\boldsymbol{r}}(\hat{\boldsymbol{r}} \cdot \boldsymbol{x}) \\ &+ \sin(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times \boldsymbol{x}) \\ &- \cos(||\boldsymbol{r}||)(\hat{\boldsymbol{r}} \times (\hat{\boldsymbol{r}} \times \boldsymbol{x}))\end{aligned}$$

Detour: 3D Rotations

Angular Displacement

- Allows tumbling
- No gimbal lock
- Orientations are space within π radius ball
- Nearly unique representation
- Singularities are shells at 2π
- Nice for interpolation

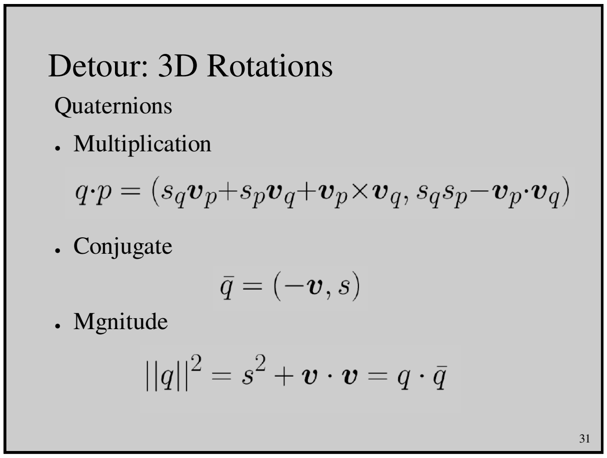
Detour: 3D Rotations

Quaternions

$$\begin{aligned}q &= (v_1, v_2, v_3, s) = (\boldsymbol{v}, s) \\ &= v_1\boldsymbol{i} + v_2\boldsymbol{j} + v_3\boldsymbol{k} + s\end{aligned}$$

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1$$

$$\begin{aligned}\boldsymbol{ij} &= k & \boldsymbol{ji} &= -k \\ \boldsymbol{jk} &= i & \boldsymbol{kj} &= -i \\ \boldsymbol{ki} &= j & \boldsymbol{ik} &= -j\end{aligned}$$



Detour: 3D Rotations

Quaternions

- Multiplication

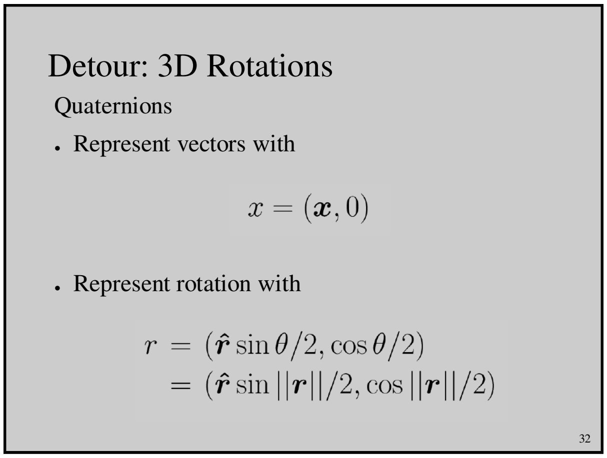
$$q\cdot p = (s_q\boldsymbol{v}_p+s_p\boldsymbol{v}_q+\boldsymbol{v}_p\times\boldsymbol{v}_q, s_q s_p-\boldsymbol{v}_p\cdot\boldsymbol{v}_q)$$

- Conjugate

$$\bar{q} = (-\boldsymbol{v}, s)$$

- Mgnitude

$$||q||^2 = s^2 + \boldsymbol{v}\cdot\boldsymbol{v} = q\cdot\bar{q}$$



Detour: 3D Rotations

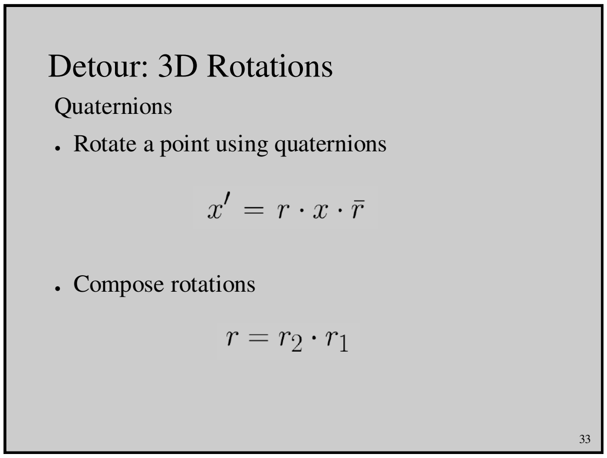
Quaternions

- Represent vectors with

$$x = (\boldsymbol{x}, 0)$$

- Represent rotation with

$$\begin{aligned} r &= (\hat{\boldsymbol{r}}\sin\theta/2, \cos\theta/2) \\ &= (\hat{\boldsymbol{r}}\sin||\boldsymbol{r}||/2, \cos||\boldsymbol{r}||/2) \end{aligned}$$



Detour: 3D Rotations

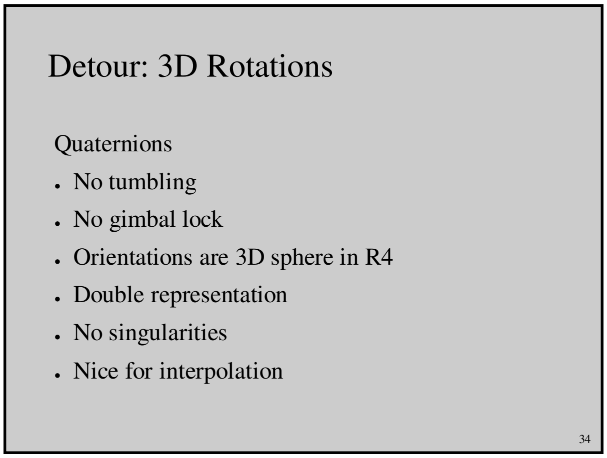
Quaternions

- Rotate a point using quaternions

$$x' = r\cdot x\cdot\bar{r}$$

- Compose rotations

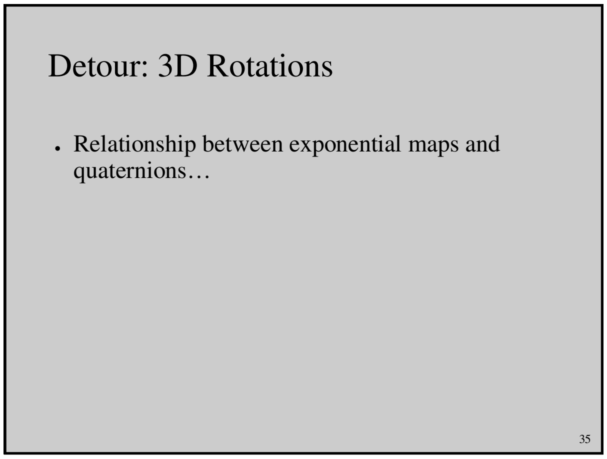
$$r = r_2\cdot r_1$$



Detour: 3D Rotations

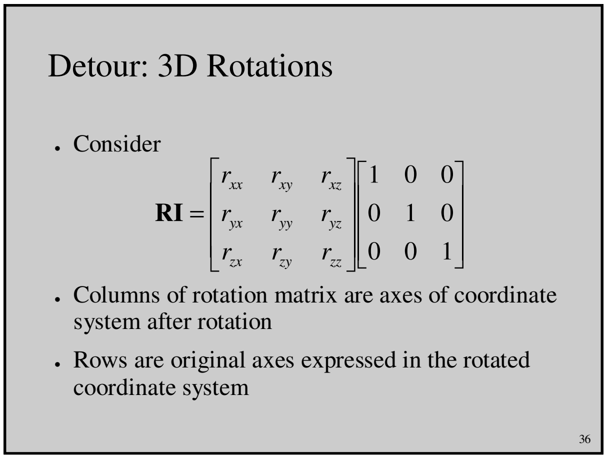
Quaternions

- No tumbling
- No gimbal lock
- Orientations are 3D sphere in R4
- Double representation
- No singularities
- Nice for interpolation



Detour: 3D Rotations

- Relationship between exponential maps and quaternions…



Detour: 3D Rotations

- Consider

$$\mathbf{RI} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Columns of rotation matrix are axes of coordinate system after rotation
- Rows are original axes expressed in the rotated coordinate system

Orthographic Projection (back from a long detour)

Step 1: translate center to origin

Step 2: Rotate so that view aligns with –Z axis and up with +Y axis

Step 3: Center view volume

Step 4: Scale view volume

$$\mathbf{M} = \underbrace{\mathbf{ST}_2}_{\mathbf{M_o}}\underbrace{\mathbf{RT}_1}_{\mathbf{M_v}}$$

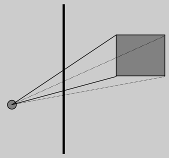
Orthographic Projection

Step 2: Rotate so that view aligns with –Z axis and up with +Y axis

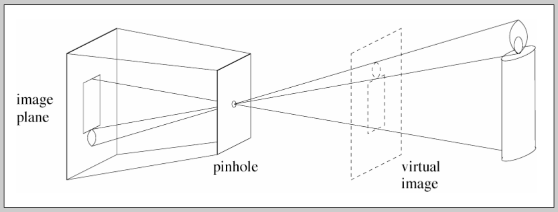
$$\mathbf{R} = \begin{bmatrix} \text{Right}_x & \text{Up}_x & -\text{View}_x \\ \text{Right}_y & \text{Up}_y & -\text{View}_y \\ \text{Right}_z & \text{Up}_z & -\text{View}_z \end{bmatrix}^{\mathsf{T}}$$

Perspective Projection

- Foreshortening: further things get smaller
- Some parallel lines stay parallel, most don’t
- Lines still look like lines
- Z ordering preserved for what we care about

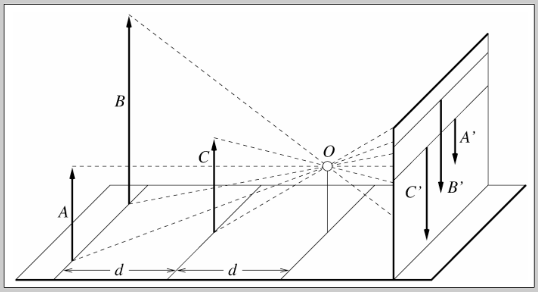


Pinhole Camera



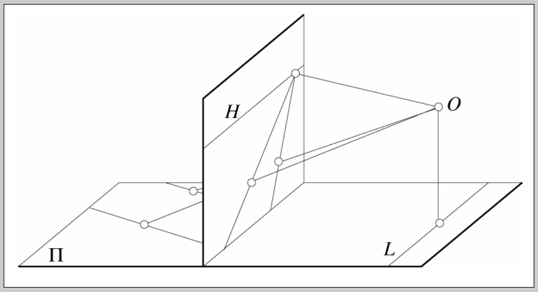
Perspective Projection

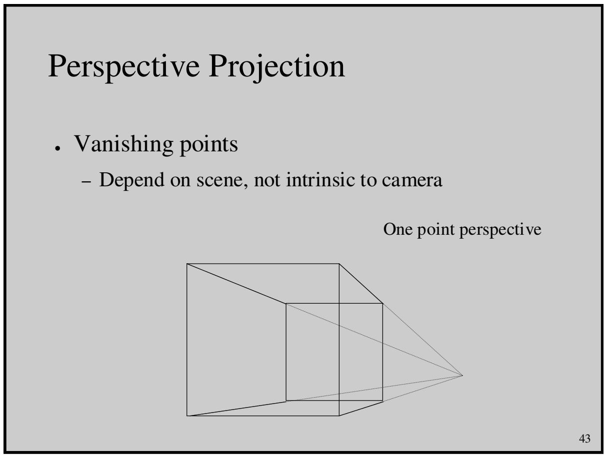
- Foreshortening: distant things are smaller



Perspective Projection

- Draw “film” in front or pinhole

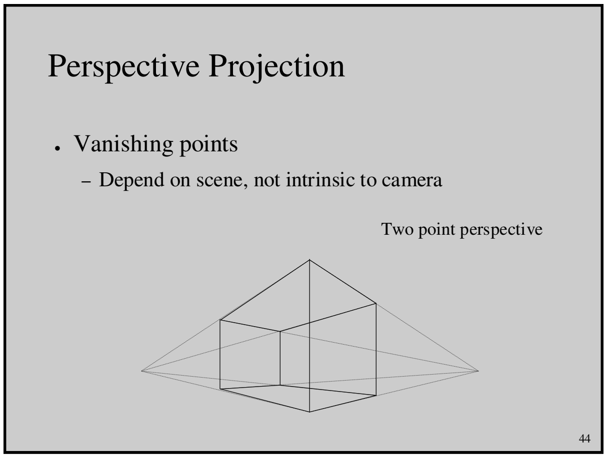




Perspective Projection

- Vanishing points
  - Depend on scene, not intrinsic to camera

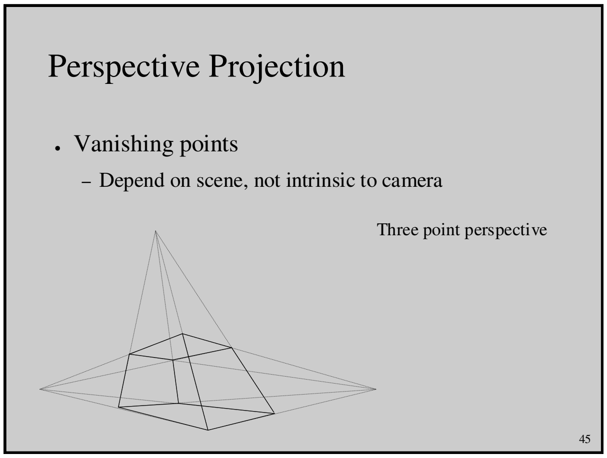
One point perspective



Perspective Projection

- Vanishing points
  - Depend on scene, not intrinsic to camera

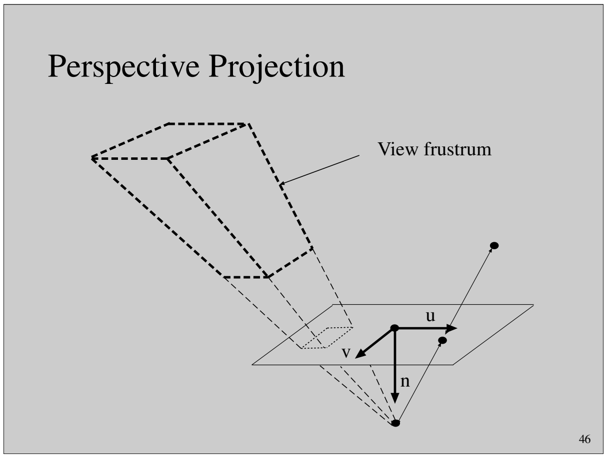
Two point perspective



Perspective Projection

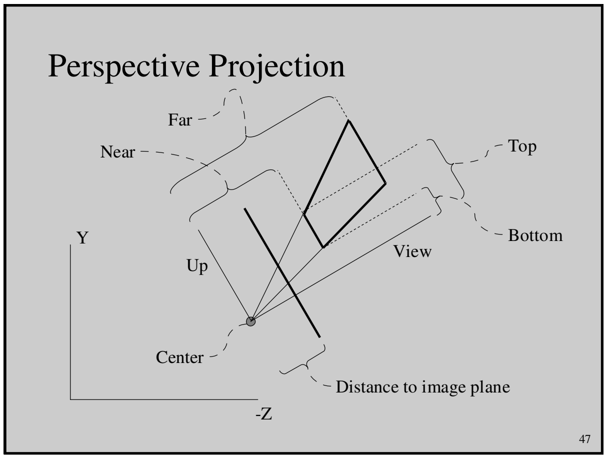
- Vanishing points
  - Depend on scene, not intrinsic to camera

Three point perspective

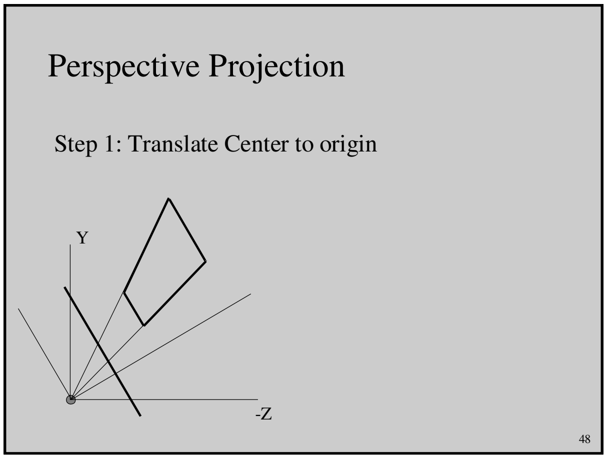


Perspective Projection

View frustrum



Perspective Projection



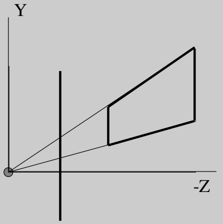
Perspective Projection

Step 1: Translate Center to origin

Perspective Projection

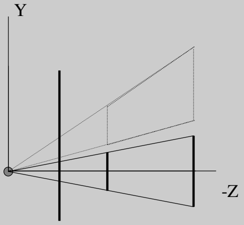
Step 2: Rotate to align

view with –Z axis
up with Y axis



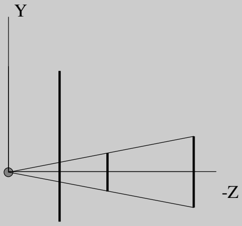
Perspective Projection

Step 3: Shear so that center-line moves to –Z axis



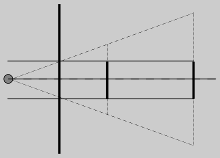
Perspective Projection

Step 4: Scale so that image plane is at Z=-1



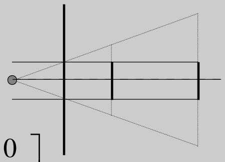
Perspective Projection

Step 5: Perspective



Perspective Projection

Step 5: Perspective



$$\mathbf{M_p} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & \frac{1}{n} & 0 \end{bmatrix}$$

Err n vs i

- Vanishing points limits