Artistic Multiprojection Rendering

Maneesh Agrawala
Stanford University
Denis Zorin
New York University
Tamara Munzner
Stanford University

CS-184: Computer Graphics

Lecture 8: Projection

Maneesh Agrawala
University of California, Berkeley

Slides based on those of James O'Brien and Stebve Marschner

Announcements

Assignments 1 and 2 results posted

Assignment 4: due Fri Oct 8 by 11pm

Today

History and Definitions

Rendering with Projections

- Windows and viewports
- Orthographic projection
- Perspective projection

History and Definitions

History of Projection

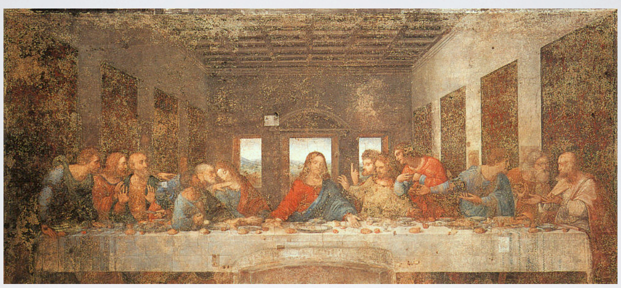
Ancient times: Greeks wrote about laws of perspective

Renaissance: Perspective is adopted by artists

Duccio c. 1308

History of Projection

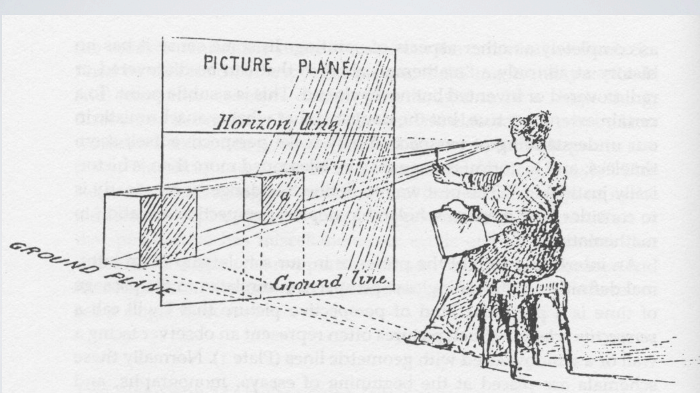
Later Renaissance: Perspective formalized precisely



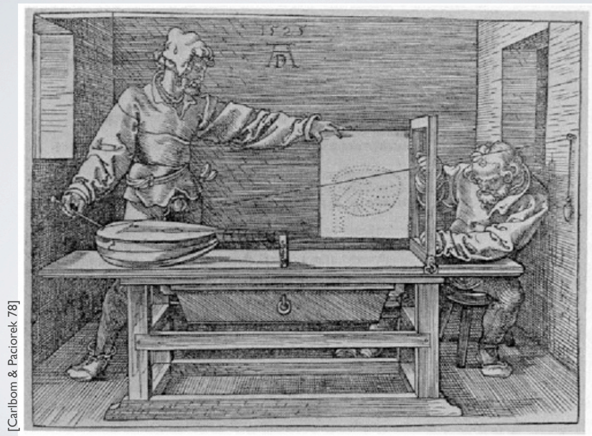
da Vinci c. 1498

Plane Projection in Drawing

Trace rays from eye through image plane into scene



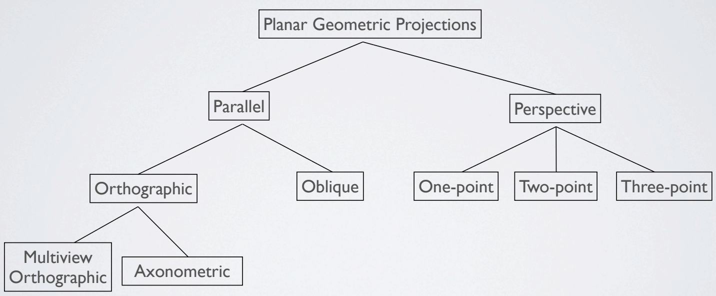
Plane Projection in Drawing



Classical Projections

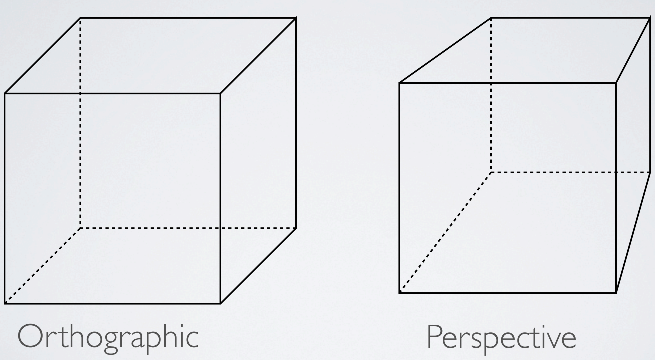
Emphasis on cube-like objects

- traditional in mechanical and architectural drawing

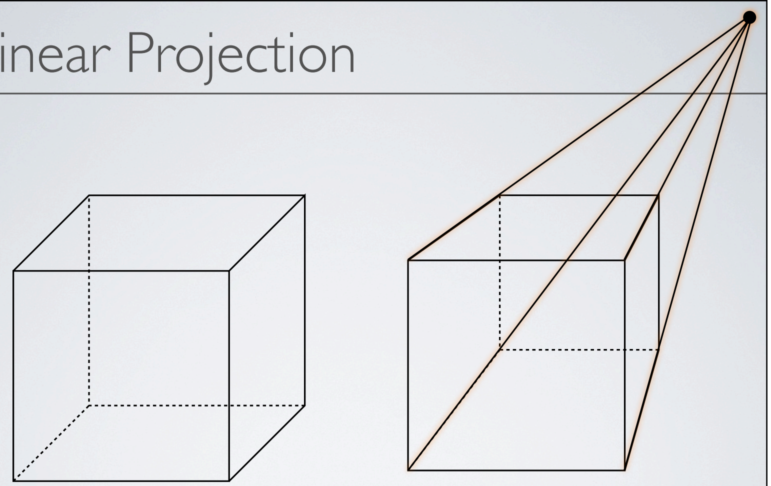


[after Carlbom & Paciorek 78]

Linear Projection



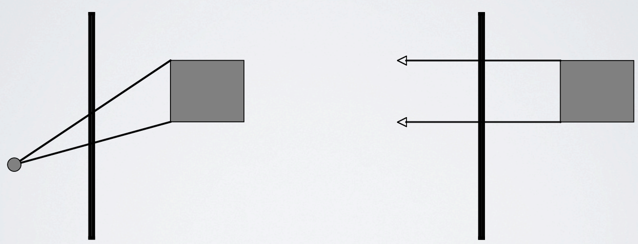
Linear Projection



Orthographic Perspective

Linear Projection

A 2D view

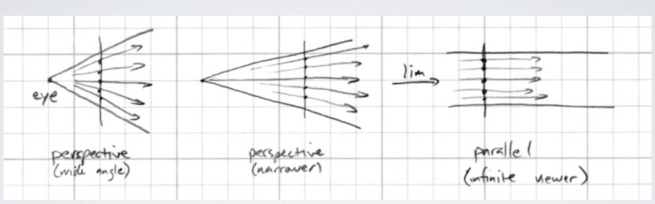


Perspective **Orthographic**

Linear Projection

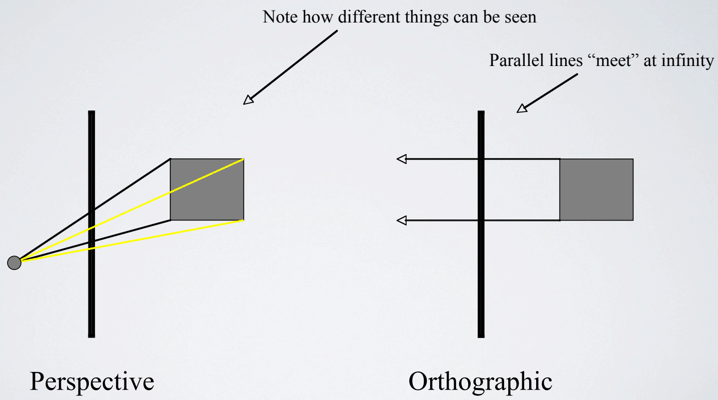
Viewing rays are parallel rather than diverging

- Like a perspective camera that's far away

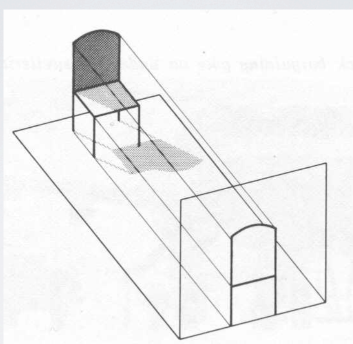


Linear Projection

A 2D view

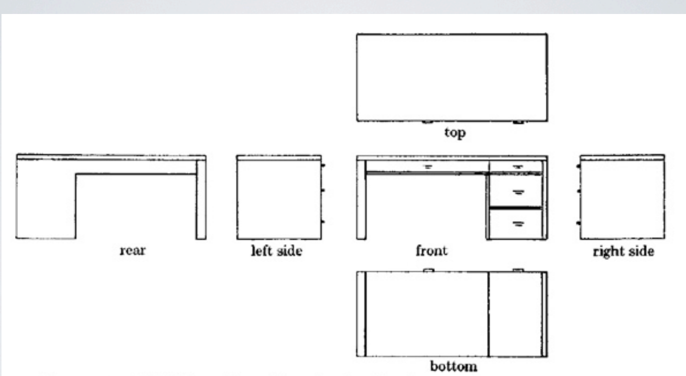


Orthographic View



- Projection plane parallel to a coordinate plane
- Projection direction perpendicular to projection plane

Multiview Orthographic



[Carlbom & Paciorek 78]

- Projection plane parallel to a coordinate plane
- Projection direction perpendicular to projection plane

Multiview Orthographic

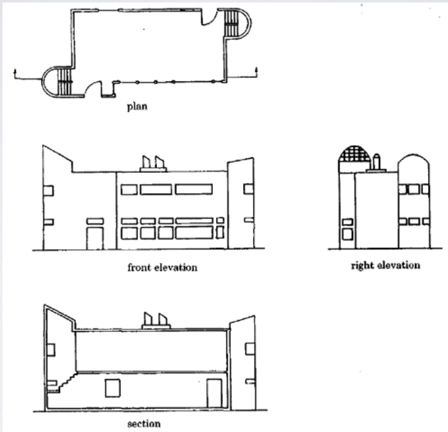
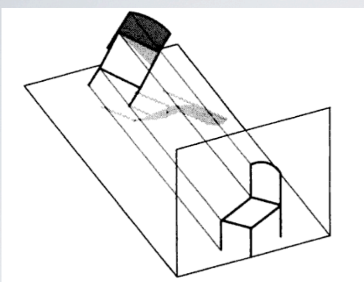
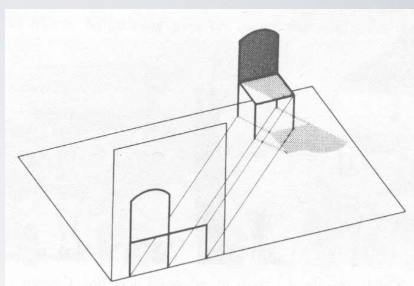


FIGURE 2-1. Multiview orthographic projection: plan, elevations, and section of a building.

[Carlbom & Paciorek 78]

Off-Axis Parallel

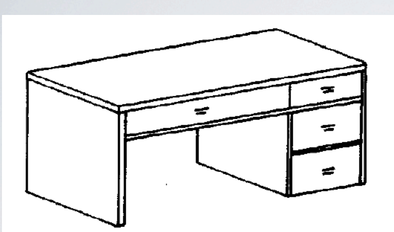


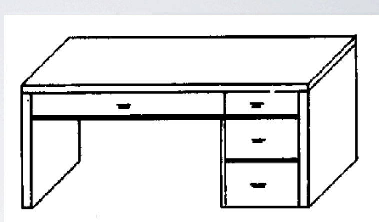


axonometric: projection plane perpendicular to projection direction but not parallel to coordinate planes

oblique: projection plane parallel to a coordinate plane but not perpendicular to projection direction.

Off-Axis Parallel





[Carlbom & Paciorek 78]

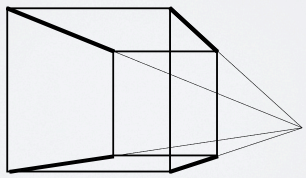
axonometric: projection plane perpendicular to projection direction but not parallel to coordinate planes

oblique: projection plane parallel to a coordinate plane but not perpendicular to projection direction.

Perspective Projection

Vanishing points

- Depend on the scene
- Not intrinsic to camera

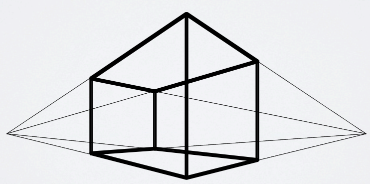


"One point perspective"

Perspective Projection

Vanishing points

- Depend on the scene
- Nor intrinsic to camera

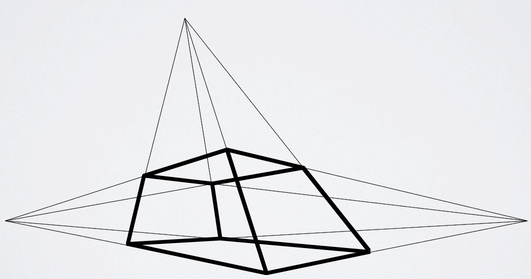


"Two point perspective"

Perspective Projection

Vanishing points

- Depend on the scene
- Not intrinsic to camera



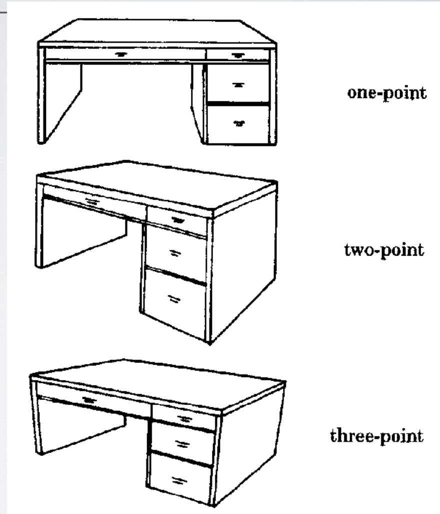
"Three point perspective"

Perspective

one-point: projection plane parallel to a coordinate plane (to two coordinate axes)

two-point: projection plane parallel to one coordinate axis

three-point: projection plane not parallel to a coordinate axis



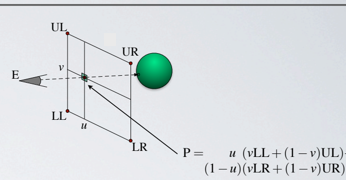
[Carlbom & Paciorek 78]

Rendering With Projections

Ray Generation vs. Projection

Viewing in ray tracing

- start with image point
- compute ray that projects to that point
- do this using geometry



Viewing by projection (primarily used in scan conversion)

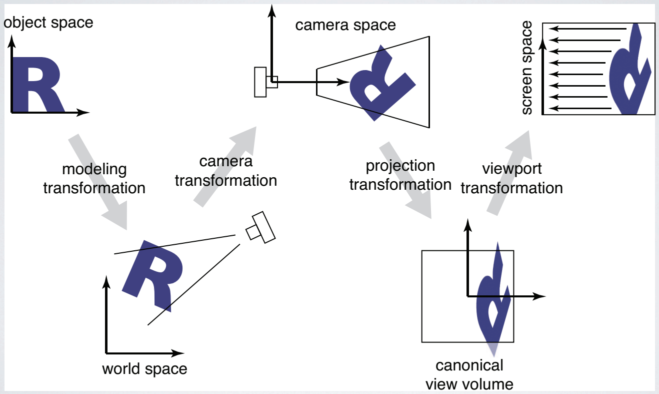
- start with 3D point
- compute image point that it projects to
- do this using transforms

Inverse processes

- ray gen. computes the preimage of projection

Pipeline of Transformations

Standard sequence of transforms



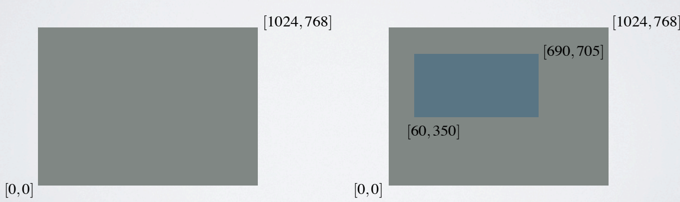
Screen Space

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

Sometimes odd

- Upside down
- Hexagonal

Screen Space

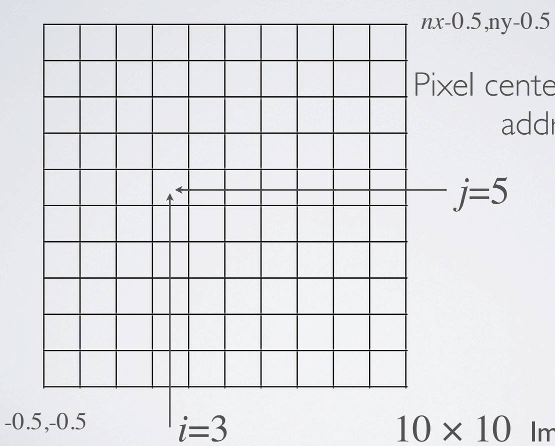
Viewport is somewhere on screen

- 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)

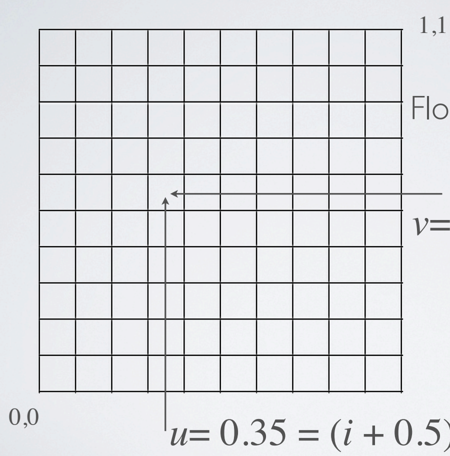
Screen Space



Pixel centers at integer addresses

10×10 **Image Resolution**

Screen Space



Float Pixel Coordinates

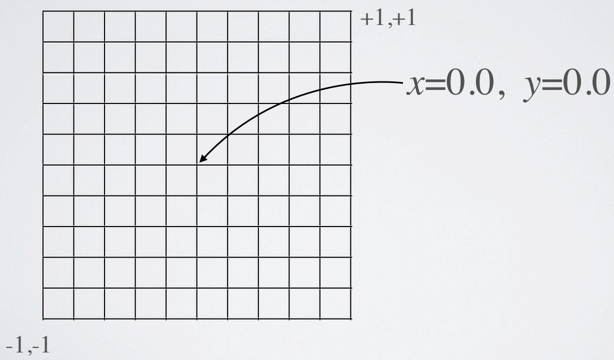
$v = 0.55 = (j + 0.5)/ny$

$u = 0.35 = (i + 0.5)/nx$

2D Canonical View Space

Canonical view region

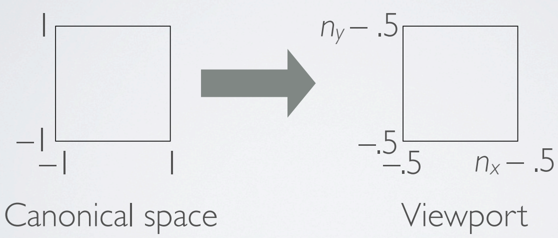
- 2D: [-1,-1] to [+1,+1]



From Shirley textbook.

2D Canonical Space to Viewport

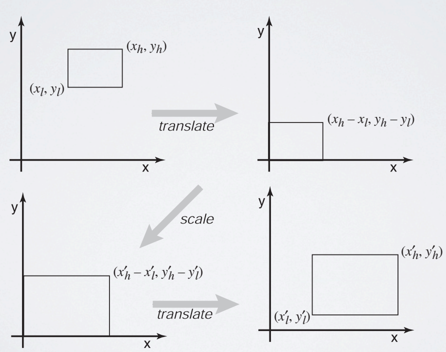
To draw in image, need coordinates in pixel units, though



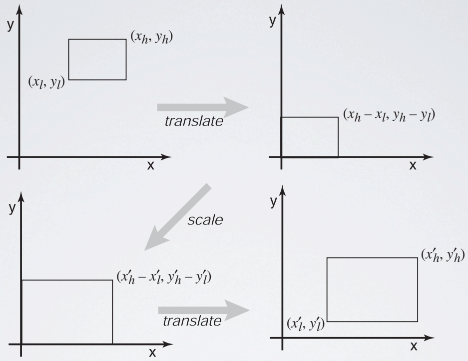
Windowing Transform

This transformation is worth generalizing: take one axis-aligned rectangle or box to another

- A useful, if mundane, piece of a transformation chain

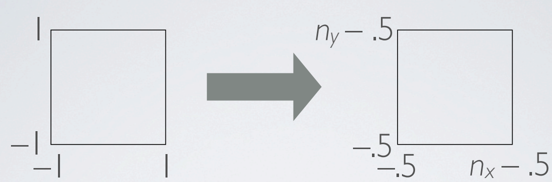


Windowing Transform



$$\begin{bmatrix}1 & 0 & x'_l\\0 & 1 & y'_l\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}\frac{x'_h-x'_l}{x_h-x_l} & 0 & 0\\0 & \frac{y'_h-y'_l}{y_h-y_l} & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -x_l\\0 & 1 & -y_l\\0 & 0 & 1\end{bmatrix} = \begin{bmatrix}\frac{x'_h-x'_l}{x_h-x_l} & 0 & \frac{x'_l x_h - x'_h x_l}{x_h-x_l}\\0 & \frac{y'_h-y'_l}{y_h-y_l} & \frac{y'_l y_h - y'_h y_l}{y_h-y_l}\\0 & 0 & 1\end{bmatrix}$$

2D Viewport Transformation



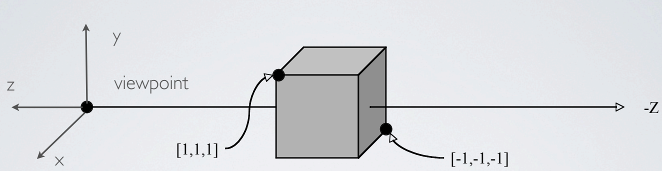
Canonical space

Viewport

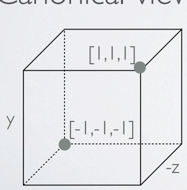
$$\begin{bmatrix}x_{\text{screen}}\\y_{\text{screen}}\\1\end{bmatrix} = \begin{bmatrix}\frac{n_x}{2} & 0 & \frac{n_x-1}{2}\\0 & -\frac{n_y}{2} & \frac{n_y-1}{2}\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_{\text{canonical}}\\y_{\text{canonical}}\\1\end{bmatrix}$$

Add minus sign if necessary to invert y

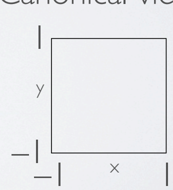
3D Canonical View Volume



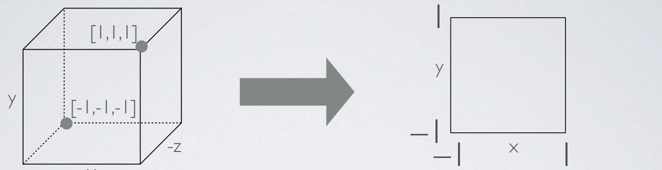
3D Canonical view volume



2D Canonical view space



3D Canonical to 2D Canonical



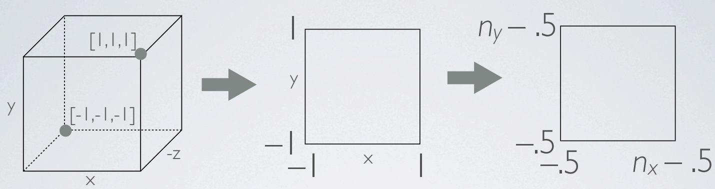
3D Canonical view volume

2D Canonical view space

to implement basic orthographic projection , just toss out z:

$$\begin{bmatrix} x_{\text{canonical 2D}} \\ y_{\text{canonical 2D}} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical 3D}} \\ y_{\text{canonical 3D}} \\ z_{\text{canonical 3D}} \\ 1 \end{bmatrix}$$

3D Canonical to Viewport



3D Canonical

2D Canonical

Viewport

$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical 3D}} \\ y_{\text{canonical 3D}} \\ z_{\text{canonical 3D}} \\ 1 \end{bmatrix}$$

3D Canonical to Viewport

$$\begin{bmatrix} x_{\text{screen}} \\ y_{\text{screen}} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & \frac{n_y-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{canonical 3D}} \\ y_{\text{canonical 3D}} \\ z_{\text{canonical 3D}} \\ 1 \end{bmatrix}$$

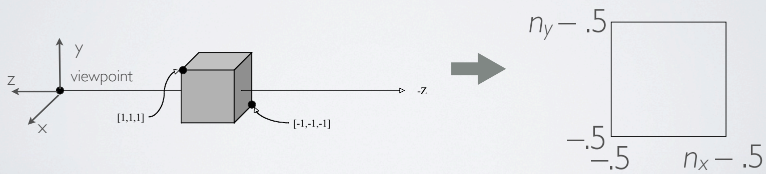
Viewport transform dropping z

$$\begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Viewport Transformation

But z will be useful later to let's carry it along for the ride

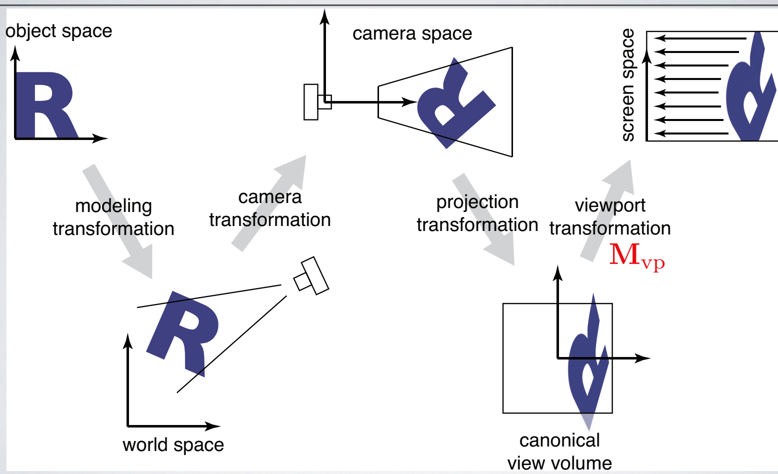
$$\mathbf{M}_{\mathrm{vp}} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3D Canonical

Viewport

Pipeline of Transformations



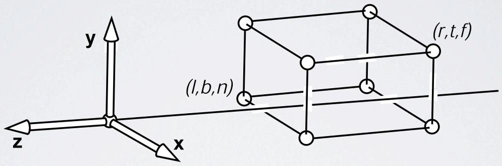
$\mathbf{M}_{\mathrm{vp}}$:3D transform, but mostly just 2D translation & scale

Orthographic Projection

Camera Space

Generalize canonical view volume

- View volume is rectangular in camera space for orthographic projection

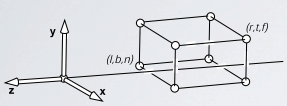


- Still assume looking down -z axis
- Specify **left, right, top, bottom** (as in ray tracing) and **near, far**

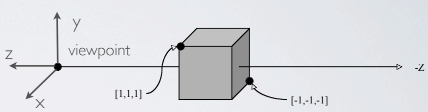
Camera Space to Canonical

Canonical view region

- Transform from 3D cube [l,b,n],[r,t,f] to canonical 3D cube [-1,-1,-1],[1,1,1]



Camera space

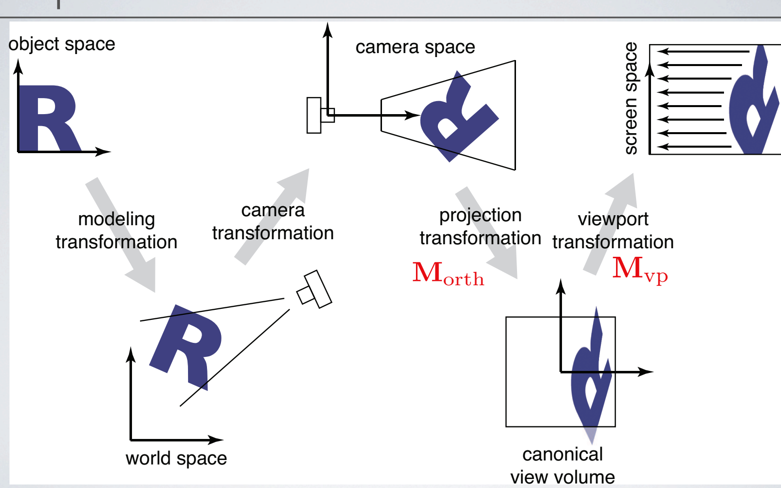


3D Canonical

Apply 3D version of windowing transform

$$\begin{bmatrix} \frac{x_h'-x_l'}{x_h-x_l} & 0 & 0 & \frac{x_l'x_h-x_h'x_l}{x_h-x_l} \\ 0 & \frac{y_h'-y_l'}{y_h-y_l} & 0 & \frac{y_l'y_h-y_h'y_l}{y_h-y_l} \\ 0 & 0 & \frac{z_h'-z_l'}{z_h-z_l} & \frac{z_l'z_h-z_h'z_l}{z_h-z_l} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{M}_{\mathrm{orth}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pipeline of Transformations

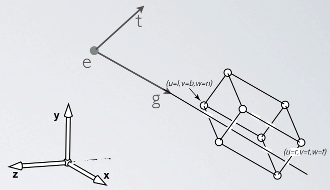


$\mathbf{M}_{\mathrm{orth}}$: Projection from camera space to canonical view vol.

World Space

Camera (eye) coord system

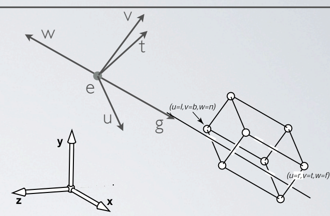
- e = eye position (any location)
- g = gaze direction (any direction)
- t = view up vector
  (any upward vector in plane bisecting viewer's head)



World Space

Camera (eye) coord system

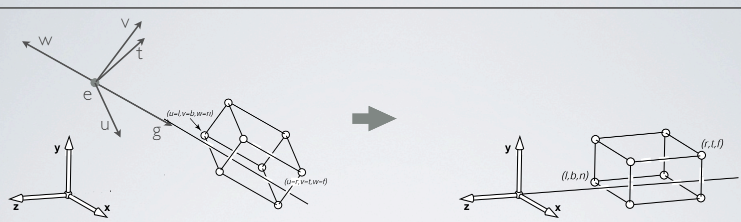
- e = eye position (any location)
- g = gaze direction (any direction)
- t = view up vector
  (any upward vector in plane bisecting viewer's head



Construct orthonormal camera frame in world space

$$w = -\frac{g}{||g||} \quad u = -\frac{t \times w}{||t \times w||} \quad v = w \times u$$

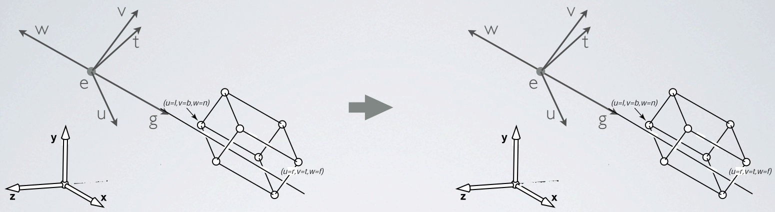
World Space to Camera Space



World space

Camera space

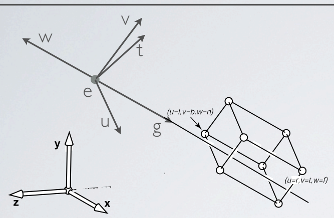
Translate Viewpoint to Origin



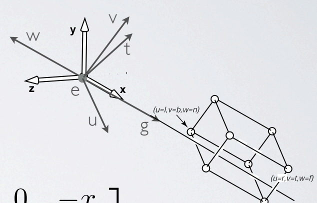
World space

$$\begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate into Camera Space



Camera space

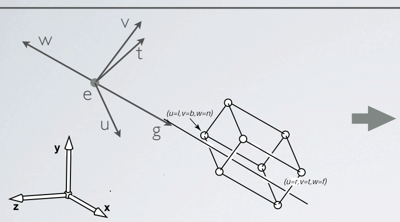


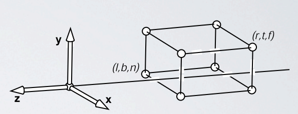
World space

$$= \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Change of coordinate systems (rotation) : camera frame into world frame

World Space to Camera Space



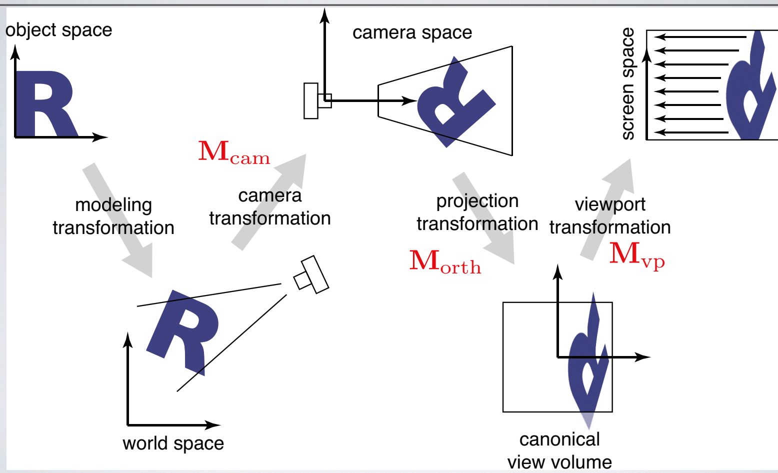


World space

Camera space

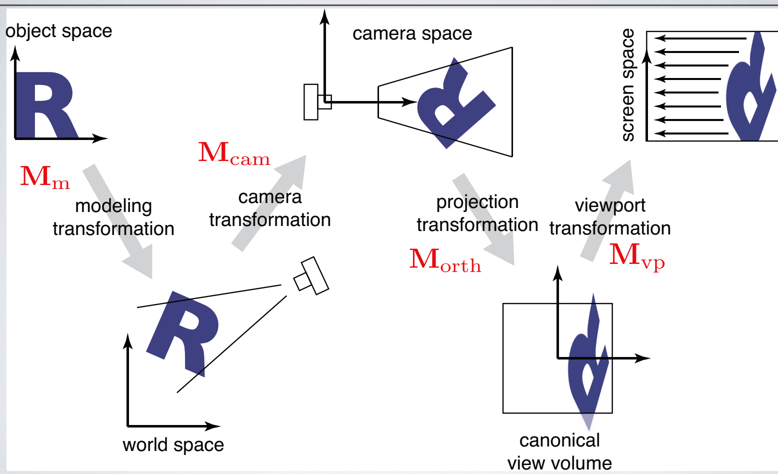
$$\mathbf{M}_{\text{cam}} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}$$

Pipeline of Transformations



$\mathbf{M}_{\mathrm{cam}}$: Translate and rotate world space into camera space

Pipeline of Transformations



$\mathbf{M}_{\mathrm{m}}$: Local object space transform (from scene graph) to world space

Orthographic Transformation

Start with coordinates in object's local coordinates

Transform into world coords (modeling transform, M_m)

Transform into eye coords (camera or viewing transform, M_{cam})

Orthographic projection, M_{orth}

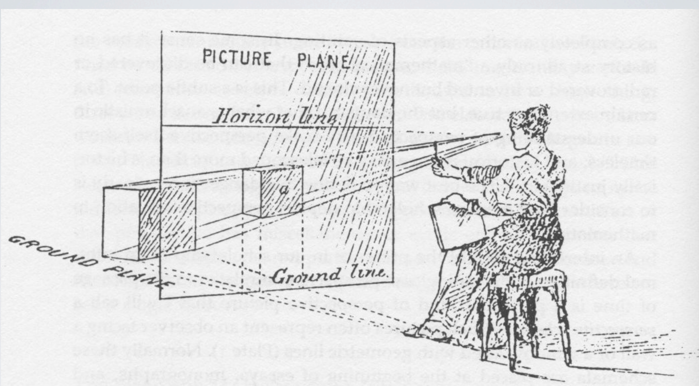
Viewport transform, M_{vp}

$$\mathbf{p}_s = \mathbf{M}_{\mathrm{vp}}\mathbf{M}_{\mathrm{orth}}\mathbf{M}_{\mathrm{cam}}\mathbf{M}_{\mathrm{m}}\mathbf{p}_o$$

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{M}_{\mathrm{m}} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

Perspective Projection

Perspective Projection



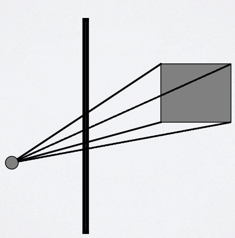
Perspective Projection

Foreshortening: further objects appear smaller

Some parallel line stay parallel, most don't

Lines still look like lines

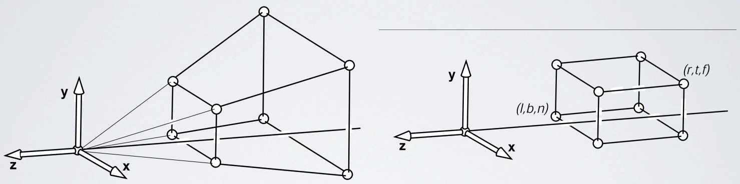
Z ordering preserved (where we care)



Perspective Camera Space

Generalize canonical view volume

- View volume is a frustum for perspective projection



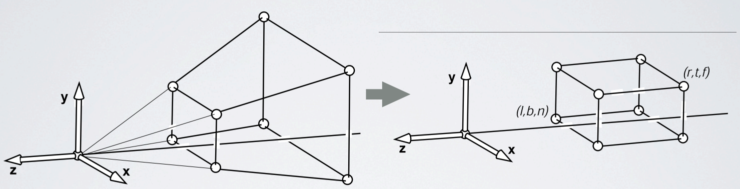
Perspective camera space   Orthographic camera space

- Sides of frustum converge at viewpoint (eye)
- But otherwise very similar to orthographic case

Frustum to Rectangular Volume

Approach

- Transform frustum into rectangular volume then use previous machinery



Perspective camera space   Orthographic camera space

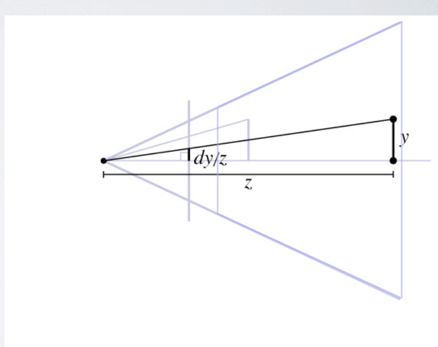
Orthographic: $\mathbf{p}_s = \mathbf{M}_{\mathrm{vp}}\mathbf{M}_{\mathrm{orth}}\mathbf{M}_{\mathrm{cam}}\mathbf{M}_{\mathrm{m}}\mathbf{p}_o$

Perspective: $\mathbf{p}_s = \mathbf{M}_{\mathrm{vp}}\mathbf{M}_{\mathrm{orth}}\mathbf{P}\mathbf{M}_{\mathrm{cam}}\mathbf{M}_{\mathrm{m}}\mathbf{p}_o$

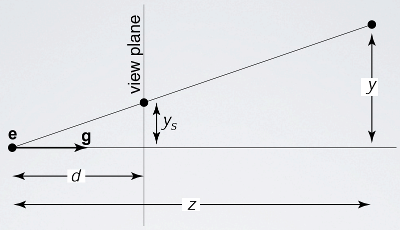
Perspective Projection (normal)

Perspective is projection by lines through a point;
"normal" = plane perpendicular to view direction

- Magnification determined by:
  - image height
  - object depth
  - image plane distance
- FOV α = 2 atan(h/(2d))



Perspective Projection



similar triangles:

$$y_s = \frac{d}{z} y$$

Homogeneous Coordinates

Perspective requires division

"True" purpose of homogeneous coords: projection

Homogeneous Coordinates

Introduced ***w*** = 1 coordinate as a placeholder

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Used as a convenience for unifying translation with linear

Can also allow arbitrary ***w***

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \sim \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

Implications of ***w***

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \sim \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

All scalar multiples of a 4-vector are equivalent

When ***w*** is not zero, can divide by ***w***

- These points represent normal *affine* points

When ***w*** is zero, it's a point at infinity, a.k.a. a direction

- This is the point where parallel lines intersect
- Can also think of it as the vanishing point

Projective Transform (Homography)

Allows for division necessary for perspective

$$\begin{bmatrix} \frac{\tilde{x}}{w} \\ \frac{\tilde{y}}{w} \\ \frac{\tilde{z}}{w} \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & x_t \\ a_{21} & a_{22} & a_{23} & y_t \\ a_{31} & a_{32} & a_{33} & z_t \\ e & f & g & h \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

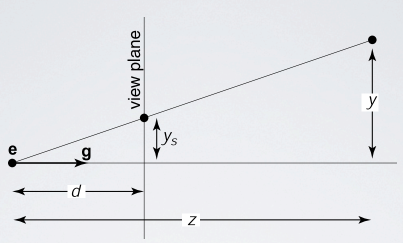
projective transform

Compare to affine transform which leaves w=1

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & x_t \\ a_{21} & a_{22} & a_{23} & y_t \\ a_{31} & a_{32} & a_{33} & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

affine transform

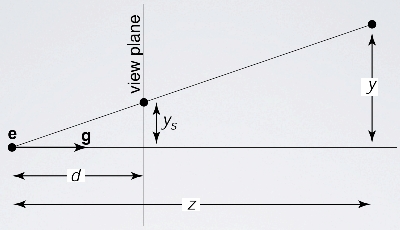
Perspective Projection



similar triangles:

$$y_s = \frac{d}{z} y$$

Perspective Projection w/o Z



to implement perspective, just move z to w:

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{dx}{z} \\ \frac{dy}{z} \\ 1 \end{bmatrix} \sim \begin{bmatrix} dx \\ dy \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Projection with Z

Straightforward extension doesn't preserve z coordinates

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\tilde{z} = z \text{ so } z_s = \frac{\tilde{z}}{z} = 1$$

To carry through **z**-coordinates use alternative formulation

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Projection with Z

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

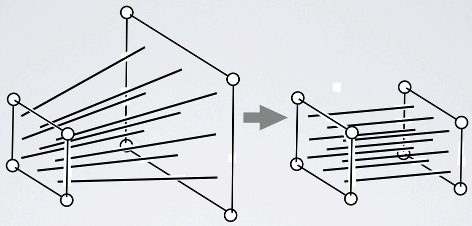
Here $\tilde{z} = az + b$ and $z_s = \dfrac{az+b}{z}$

Set $d = n$ and preserve depths at near and far planes

- For $z = n$ we want $z_s = n$
- For $z = f$ we want $z_s = f$
- Solve for **a** and **b** we obtain

$$\text{result: } a = n + f \text{ and } b = -fn \text{ (try it)}$$

Perspective Matrix



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

Perspective Transformation Chain

Transform into world coords (modeling transform, M_m)

Transform into eye coords (camera or viewing transform M_{cam})

Perspective matrix, P

Orthographic projection, M_{orth}

Viewport transform, M_{vp}

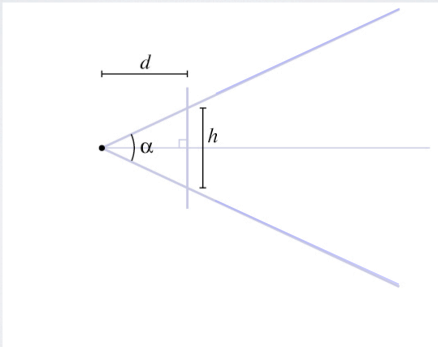
$$\mathbf{p}_s = \mathbf{M}_{\text{vp}}\mathbf{M}_{\text{orth}}\mathbf{P}\mathbf{M}_{\text{cam}}\mathbf{M}_{\text{m}}\mathbf{p}_o$$

$$\begin{bmatrix} x_s \\ y_s \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{M}_{\text{cam}}\mathbf{M}_{\text{m}} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix}$$

Issues with Perspective

Normal Perspective and FOV

FOV $\alpha = 2\ \mathrm{atan}(\boldsymbol{h}/(2\boldsymbol{d}))$



Field of View

Angle between rays along opposite edges of perspective image

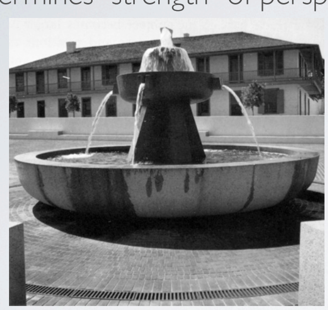
- Easy to compute only for "normal" perspective
- Have to decide to measure vert., horiz., or diag.

In cameras, determined by focal length

- Confusing because of many image sizes
- For 35mm format (36mm by 24mm image)
  - 18mm = 67° v.f.o.v. — super-wide angle
  - 28mm = 46° v.f.o.v. — wide angle
  - 50mm = 27° v.f.o.v. — "normal"
  - 100mm = 14° v.f.o.v. — narrow angle ("telephoto")

Field of View

Determines "strength" of perspective effects



close viewpoint
wide angle
prominent
foreshortening

far viewpoint
narrow angle
little foreshortening

[Ansel Adams]

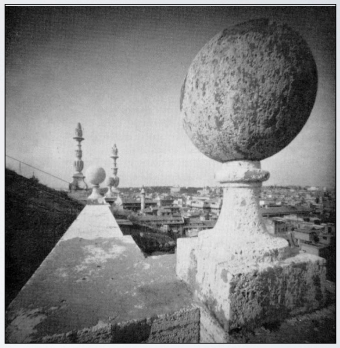
Choice of Field of View

In photography, wide angle lenses are specialty tools

- "hard to work with"
- easy to create weird-looking effects

In graphics, you can type in whatever FOV you want

- People often type in big numbers!



Correct wide angle image

[Denis Zorin 95]

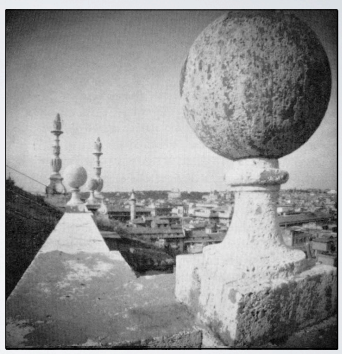
Choice of Field of View

In photography, wide angle lenses are specialty tools

- "hard to work with"
- easy to create weird-looking effects

In graphics, you can type in whatever FOV you want

- People often type in big numbers!



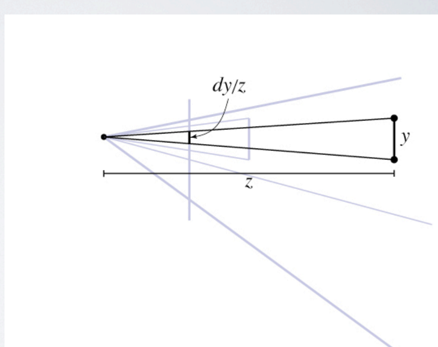
Correct telephoto image

[Denis Zorin 95]

Shifted Perspective Projection

Perspective but proj. plane not perpendicular to view direction

- Additional parameter projection plane normal
- Equivalent to cropping out off-center rect. from larger "normal" perspective
- Corresponds to ***view camera*** in photography



Why Shifted Perspective?

Control convergence of parallel lines

Standard example: architecture

- Buildings are taller than you, so you look up
- Top of building is farther away, so it looks smaller

Solution: make projection plane parallel to facade

- Top of building is the same distance ***from the projection plane***

[Philip Greenspun]

camera tilted up: converging vertical lines

[Philip Greenspun]

lens shifted up: parallel vertical lines