



University of British Columbia
 CPSC 314 Computer Graphics
 Jan-Apr 2005

Tamara Munzner

Projections

Week 4, Mon Jan 24

<http://www.ugrad.cs.ubc.ca/~cs314/Vjan2005>

News

- make sure you've signed up for demo slot

2

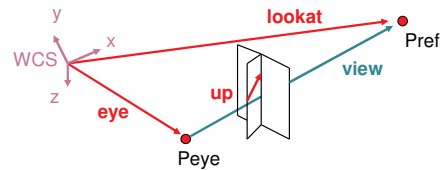
Review: Display Lists

- reuse block of OpenGL code
 - efficiency
 - multiple instances of same object
 - static objects redrawn often
 - exploit hierarchical structure when possible
- set up list once with glNewList/glEndList
 - call multiple times

3

Review: Camera Motion

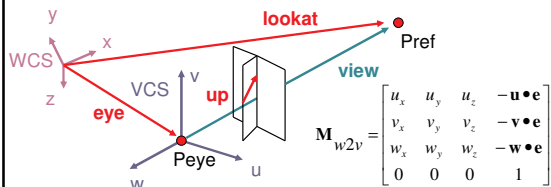
- rotate/translate/scale difficult to control
- arbitrary viewing position
 - eye point, gaze/lookat direction, up vector



4

Review: World to View Coordinates

- translate **eye** to origin
- rotate **view** vector (**lookat** - **eye**) to **w** axis
- rotate around **w** to bring **up** into **vw**-plane



5




University of British Columbia
 CPSC 314 Computer Graphics
 Jan-Apr 2005

Tamara Munzner


Projections

Pinhole Camera


- ingredients
 - box
 - film
 - hole punch
- results
 - pictures!



www.kodak.com



www.pinhole.org

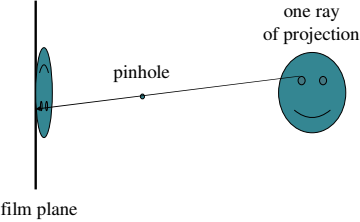


www.debevec.org/Pinhole

7

Pinhole Camera

- theoretical perfect pinhole



one ray of projection

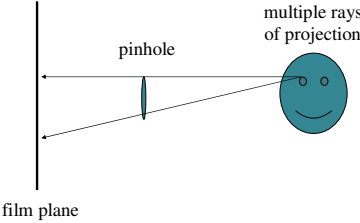
pinhole

film plane

8

Pinhole Camera

- non-zero sized hole



multiple rays of projection

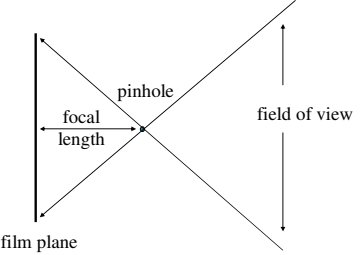
pinhole

film plane

9

Pinhole Camera

- field of view and focal length



field of view

pinhole

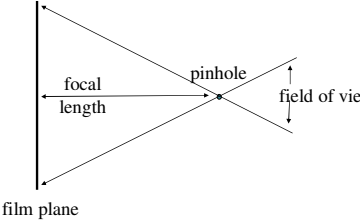
focal length

film plane

10

Pinhole Camera

- field of view and focal length



field of view

pinhole

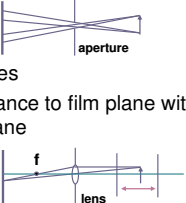
focal length

film plane

11

Real Cameras

- pinhole camera has small **aperture** (lens opening)
 - hard to get enough light to expose the film
- real pinhole camera**
- lens permits larger apertures
- lens permits changing distance to film plane without actually moving the film plane
- camera**



aperture

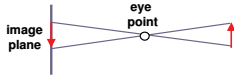
lens

price to pay: limited depth of field

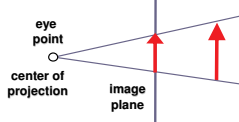
12

Graphics Cameras

- real pinhole camera: image inverted



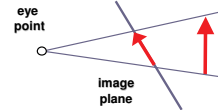
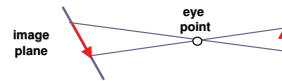
- computer graphics camera: convenient equivalent



13

General Projection

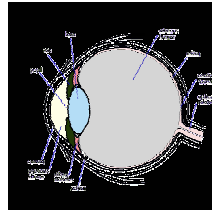
- image plane need not be perpendicular to view plane



14

Perspective Projection

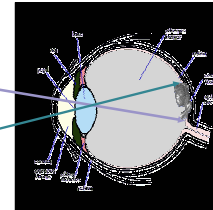
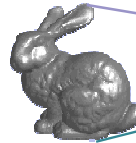
- our camera must model perspective



15

Perspective Projection

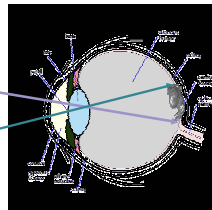
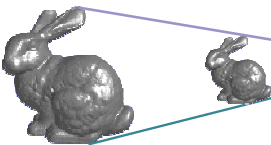
- our camera must model perspective



16

Perspective Projection

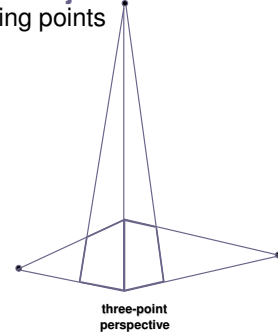
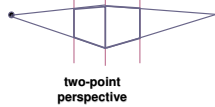
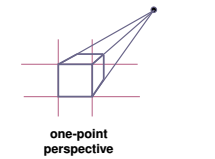
- our camera must model perspective



17

Perspective Projections

- classified by vanishing points



18

Projective Transformations

- planar geometric projections
- planar: onto a plane
- geometric: using straight lines
- projections: 3D -> 2D
- aka projective mappings
- counterexamples?

19

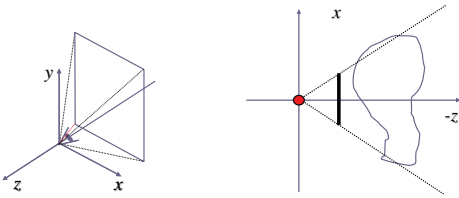
Projective Transformations

- properties
- lines mapped to lines and triangles to triangles
- parallel lines do NOT remain parallel
 - e.g. rails vanishing at infinity
- affine combinations are NOT preserved
 - e.g. center of a line does not map to center of projected line (perspective foreshortening)

20

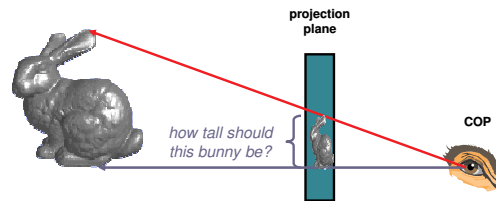
Perspective Projection

- project all geometry
- through common center of projection (eye point)
- onto an image plane



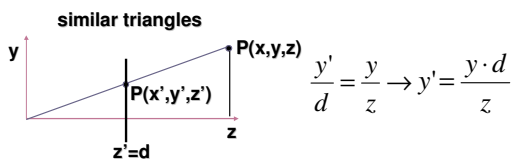
21

Perspective Projection



22

Basic Perspective Projection



$$\frac{y'}{d} = \frac{y}{z} \rightarrow y' = \frac{y \cdot d}{z}$$

$$\frac{x'}{d} = \frac{x}{z} \rightarrow x' = \frac{x \cdot d}{z} \quad \text{but } z' = d$$

- nonuniform foreshortening
- not affine

23

Perspective Projection

- desired result for a point $[x, y, z, 1]^T$ projected onto the view plane:

$$\frac{x'}{d} = \frac{x}{z}, \quad \frac{y'}{d} = \frac{y}{z}$$

$$x' = \frac{x \cdot d}{z} = \frac{x}{z/d}, \quad y' = \frac{y \cdot d}{z} = \frac{y}{z/d}, \quad z = d$$

- what could a matrix look like to do this?

24

Perspective Projection Matrix

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

25

Perspective Projection Matrix

$$\begin{bmatrix} x \\ z/d \\ y \\ z/d \\ d \end{bmatrix} \text{ is homogenized version of } \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

where $w = z/d$

26

Perspective Projection Matrix

$$\begin{bmatrix} x \\ z/d \\ y \\ z/d \\ d \end{bmatrix} \text{ is homogenized version of } \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

where $w = z/d$

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

27

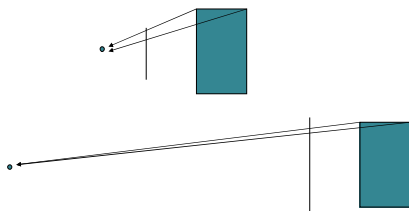
Perspective Projection

- n expressible with 4x4 homogeneous matrix
 - n use previously untouched bottom row
- n perspective projection is irreversible
 - n many 3D points can be mapped to same (x, y, d) on the projection plane
 - n no way to retrieve the unique z values

28

Moving COP to Infinity

- n as COP moves away, lines approach parallel
- n when COP at infinity, **orthographic** view



29

Orthographic Camera Projection

- n camera's back plane parallel to lens
- n infinite focal length
- n no perspective convergence

$$\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

- n just throw away z values

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

30

Perspective to Orthographic

- n transformation of space
- n center of projection moves to infinity
- n view volume transformed
 - o from frustum (truncated pyramid) to parallelepiped (box)

View Volumes

- n specifies field-of-view, used for clipping
- n restricts domain of z stored for visibility test

View Volume

- n convention
- n viewing frustum mapped to specific parallelepiped
 - o Normalized Device Coordinates (NDC)
 - o same as clipping coords
- n only objects inside the parallelepiped get rendered
- n which parallelepiped?
 - o depends on rendering system

Normalized Device Coordinates

left/right $x = +/- 1$, top/bottom $y = +/- 1$, near/far $z = +/- 1$

Understanding Z

- n z axis flip changes coord system handedness
- n RHS before projection (eye/view coords)
- n LHS after projection (clip, norm device coords)

Understanding Z

near, far always positive in OpenGL calls
`glOrtho(left,right,bot,top,near,far);`
`glFrustum(left,right,bot,top,near,far);`
`glPerspective(fovy,aspect,near,far);`

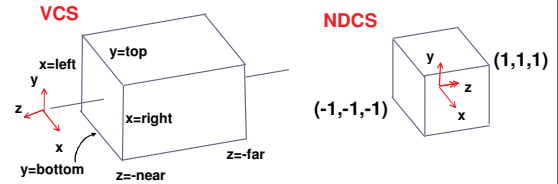
Understanding Z

- n why near and far plane?
- n near plane:
 - n avoid singularity (division by zero, or very small numbers)
- n far plane:
 - n store depth in fixed-point representation (integer), thus have to have fixed range of values (0...1)
 - n avoid/reduce numerical precision artifacts for distant objects

37

Orthographic Derivation

- n scale, translate, reflect for new coord sys

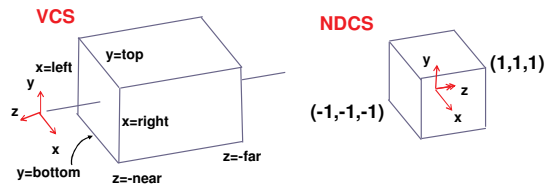


38

Orthographic Derivation

- n scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \quad \begin{array}{l} y = \text{top} \rightarrow y' = 1 \\ y = \text{bot} \rightarrow y' = -1 \end{array}$$



39

Orthographic Derivation

- n scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \quad \begin{array}{l} y = \text{top} \rightarrow y' = 1 \quad 1 = a \cdot \text{top} + b \\ y = \text{bot} \rightarrow y' = -1 \quad -1 = a \cdot \text{bot} + b \end{array}$$

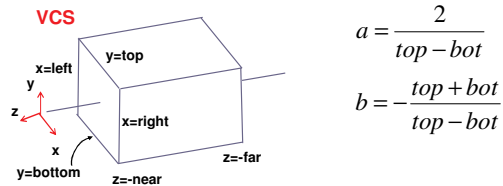
$$\begin{aligned} b &= 1 - a \cdot \text{top}, b = -1 - a \cdot \text{bot} & 1 &= \frac{2}{\text{top} - \text{bot}} \text{top} + b \\ 1 - a \cdot \text{top} &= -1 - a \cdot \text{bot} & b &= 1 - \frac{2 \cdot \text{top}}{\text{top} - \text{bot}} \\ 1 - (-1) &= -a \cdot \text{bot} - (-a \cdot \text{top}) & b &= \frac{(\text{top} - \text{bot}) - 2 \cdot \text{top}}{\text{top} - \text{bot}} \\ 2 &= a(-\text{bot} + \text{top}) & a &= \frac{2}{\text{top} - \text{bot}} \\ a &= \frac{2}{\text{top} - \text{bot}} & b &= \frac{-\text{top} - \text{bot}}{\text{top} - \text{bot}} \end{aligned}$$

40

Orthographic Derivation

- n scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \quad \begin{array}{l} y = \text{top} \rightarrow y' = 1 \\ y = \text{bot} \rightarrow y' = -1 \end{array}$$



$$\begin{aligned} a &= \frac{2}{\text{top} - \text{bot}} \\ b &= -\frac{\text{top} + \text{bot}}{\text{top} - \text{bot}} \end{aligned}$$

same idea for right/left, far/near

41

Orthographic Derivation

- n scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{\text{right} - \text{left}} & 0 & 0 & -\frac{\text{right} + \text{left}}{\text{right} - \text{left}} \\ 0 & \frac{2}{\text{top} - \text{bot}} & 0 & -\frac{\text{top} + \text{bot}}{\text{top} - \text{bot}} \\ 0 & 0 & \frac{-2}{\text{far} - \text{near}} & -\frac{\text{far} + \text{near}}{\text{far} - \text{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

42

Orthographic Derivation

n scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & -\frac{right + left}{right - left} \\ 0 & \frac{2}{top - bot} & 0 & -\frac{top + bot}{top - bot} \\ 0 & 0 & \frac{-2}{far - near} & -\frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

43

Orthographic Derivation

n scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & \frac{right + left}{right - left} \\ 0 & \frac{2}{top - bot} & 0 & -\frac{top + bot}{top - bot} \\ 0 & 0 & \frac{-2}{far - near} & -\frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

44

Orthographic Derivation

n scale, translate, reflect for new coord sys

$$P' = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & -\frac{right + left}{right - left} \\ 0 & \frac{2}{top - bot} & 0 & -\frac{top + bot}{top - bot} \\ 0 & 0 & \frac{-2}{far - near} & -\frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

45

Orthographic OpenGL

```
glMatrixMode (GL_PROJECTION) ;
glLoadIdentity () ;
glOrtho (left, right, bot, top, near, far) ;
```

46