



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

Tamara Munzner

## Projections II

Week 4, Wed Jan 26

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

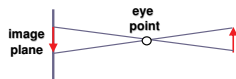
## Reading (Mon and today)

- n FCG
  - n Section 5.3.1
  - n rest of Chapter 6
- n RB
  - n rest of Chapter Viewing
  - n rest of Appendix Homogeneous Coords

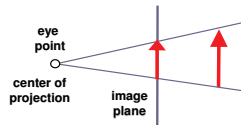
2

## Review: Graphics Cameras

- n real pinhole camera: image inverted

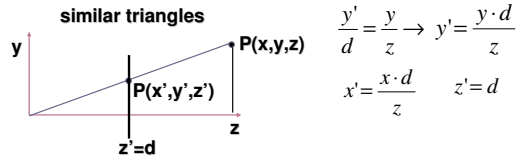


- n computer graphics camera: convenient equivalent



3

## Review: Basic Perspective Projection



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

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

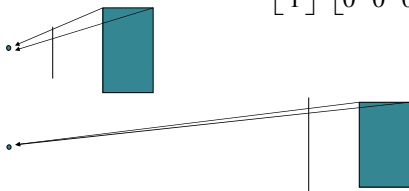
$$\begin{bmatrix} x \\ z/d \\ y \\ z/d \\ d \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

4

## Review: Orthographic Cameras

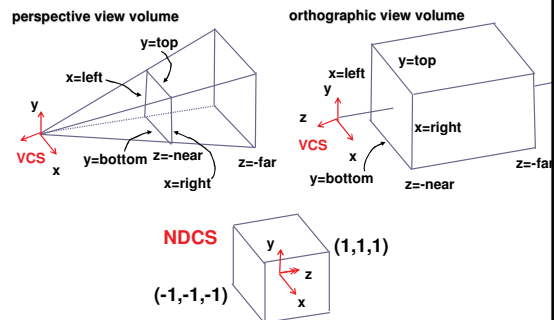
- n center of projection at infinity
- n no perspective convergence
- n just throw away z values

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



5

## Review: Transforming View Volumes



6

### Review: Ortho to NDC Derivation

- scale, translate, reflect for new coord sys

**VCS**

**NDCS**

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

7

### NDC to Viewport Transformation

- generate pixel coordinates
- map x, y from range -1...1 (NDC) to pixel coordinates on the display
- involves 2D scaling and translation

8

### NDC to Viewport Transformation

- 2D scaling and translation

$$x_{DCS} = w \frac{(x_{NDCS} + 1)}{2}$$

$$y_{DCS} = h \frac{(y_{NDCS} + 1)}{2}$$

$$z_{DCS} = \frac{(z_{NDCS} + 1)}{2}$$

**OpenGL:**  
`glViewport(x, y, a, b);`  
**default:**  
`glViewport(0, 0, w, h);`

9

### Origin Location

- yet more possibly confusing conventions
- OpenGL: lower left
- most window systems: upper left
- often have to flip your y coordinates
- when interpreting mouse position

10

### Perspective Example

tracks in VCS:  
left x=-1, y=-1  
right x=1, y=-1

view volume  
left = -1, right = 1  
bot = -1, top = 1  
near = 1, far = 4

11

### Viewing Transformation

object

OCS

modeling transformation

$M_{mod}$

world

WCS

viewing transformation

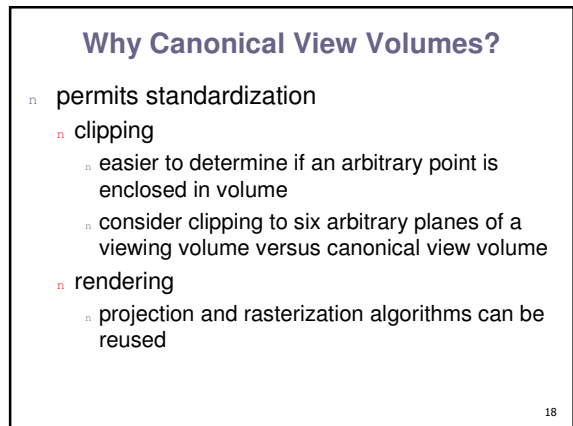
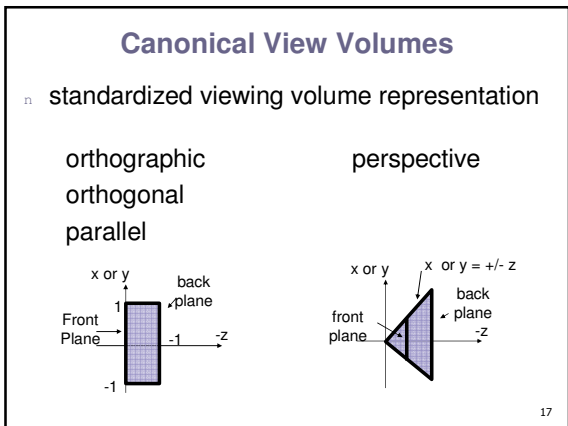
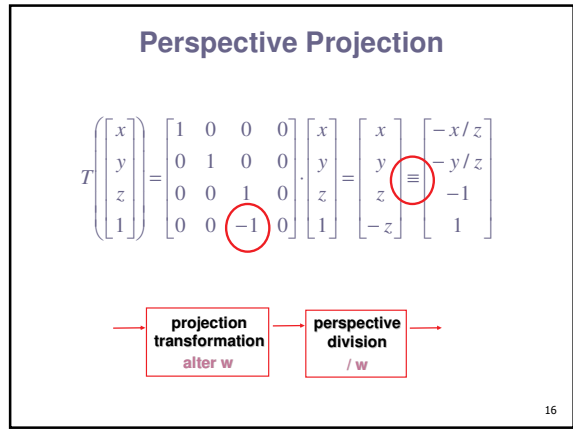
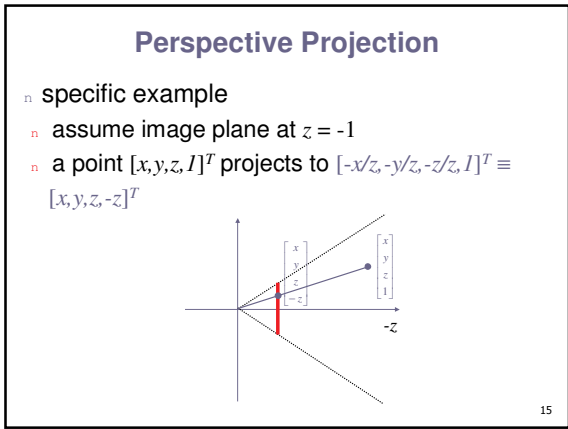
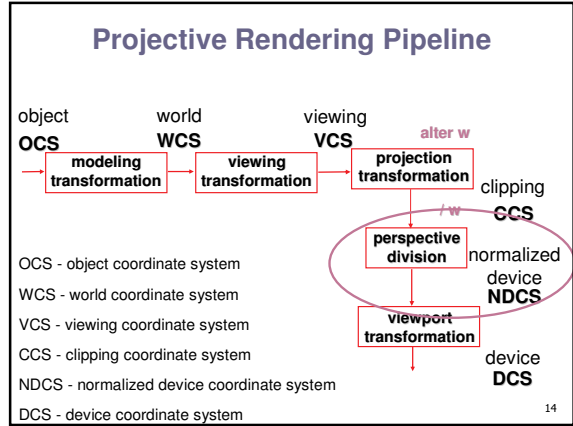
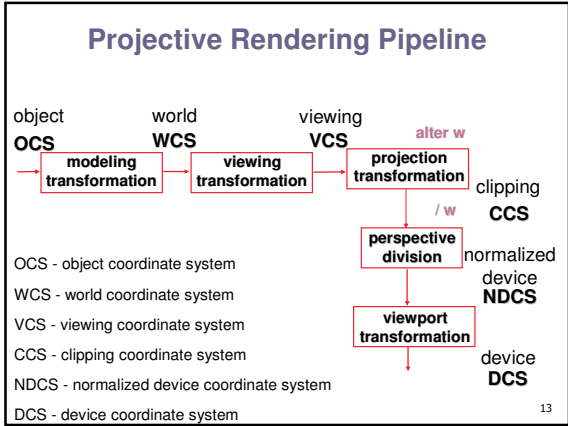
$M_{cam}$

viewing

VCS

OpenGL ModelView matrix

12



### Projection Normalization

- one additional step of standardization
- warp perspective view volume to orthogonal view volume
  - render all scenes with orthographic projection!

### Predistortion

### Perspective Normalization

- perspective viewing frustum transformed to cube
- orthographic rendering of cube produces same image as perspective rendering of original

### Demos

- Tuebingen applets from Frank Hanisch
  - <http://www.gris.uni-tuebingen.de/projects/grdev/doc/html/etc/AppletIndex.html#Transformationen>

### Perspective Warp

- matrix formulation

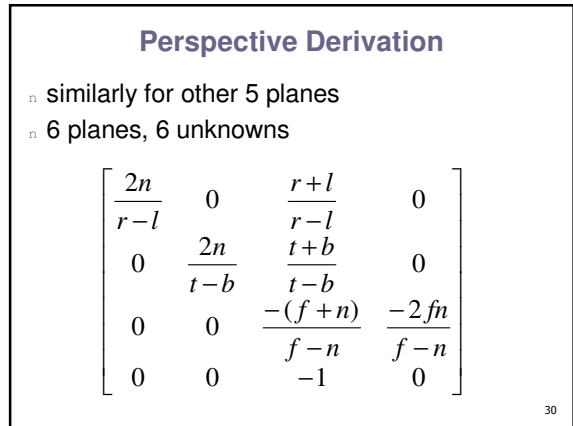
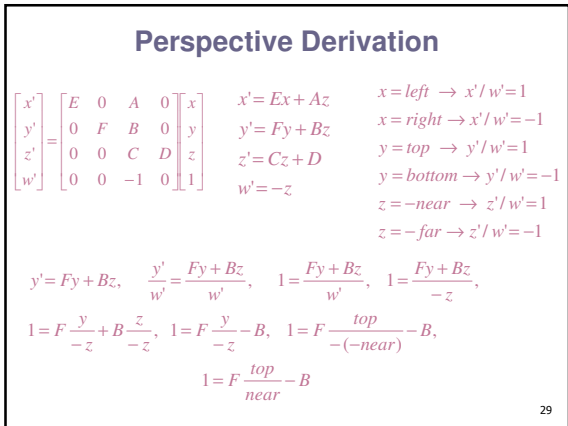
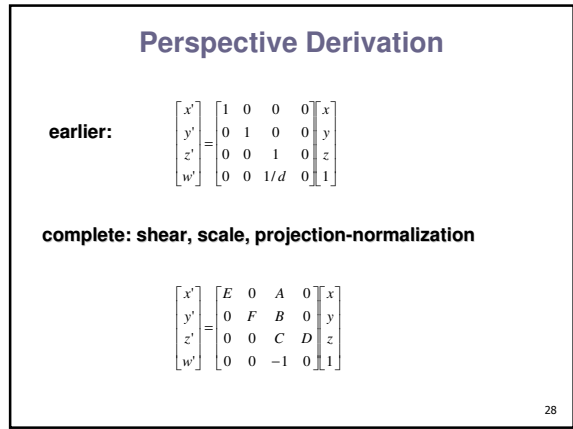
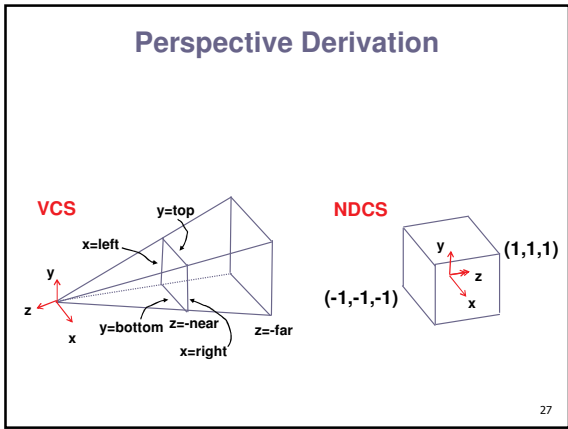
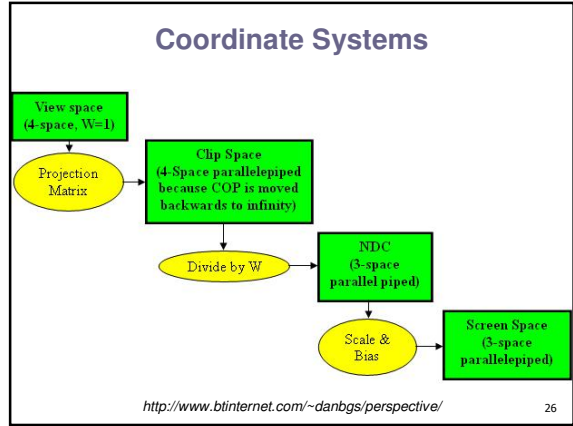
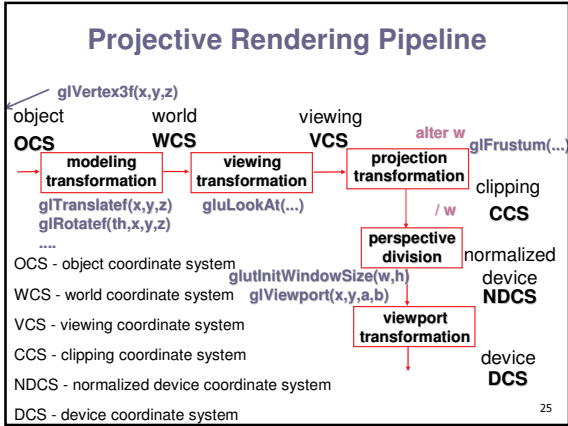
$$(x, y, z, 1) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{d}{d-\alpha} & \frac{1}{d} \\ 0 & 0 & \frac{-\alpha}{d-\alpha} & 0 \end{bmatrix} = \left( x, y, \frac{(z-\alpha) \cdot d}{d-\alpha}, \frac{z}{d} \right)$$

$$(x_p, y_p, z_p) = \left( \frac{x}{z/d}, \frac{y}{z/d}, \frac{d^2}{d-\alpha} \left( 1 - \frac{\alpha}{z} \right) \right)$$

- preserves relative depth (third coordinate)
- what does  $\alpha = 0$  mean?

### Projection Normalization

- distort such that orthographic projection of distorted objects is desired persp projection
- separate division from standard matrix multiplies
- clip after warp, before divide
- division: normalization



### Perspective Example

view volume

- left = -1, right = 1
- bot = -1, top = 1
- near = 1, far = 4

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -5/3 & -8/3 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

31

### Perspective Example

$$\begin{bmatrix} 1 & & & \\ & -1 & & \\ & -5z_{VCS}/3 - 8/3 & & \\ & -z_{VCS} & & \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & -5/3 & -8/3 & \\ & & & z_{VCS} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ z_{VCS} \\ 1 \end{bmatrix}$$

/ **w**

$$\begin{aligned} x_{NDCS} &= -1/z_{VCS} \\ y_{NDCS} &= 1/z_{VCS} \\ z_{NDCS} &= \frac{5}{3} + \frac{8}{3z_{VCS}} \end{aligned}$$

32

### Asymmetric Frusta

- our formulation allows asymmetry
- why bother?

33

### Simpler Formulation

- left, right, bottom, top, near, far
- nonintuitive
- often overkill
- look through window center
- symmetric frustum
- constraints
- left = -right, bottom = -top

34

### Field-of-View Formulation

- FOV in one direction + aspect ratio (w/h)
- determines FOV in other direction
- also set near, far (reasonably intuitive)

35

### Perspective OpenGL

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

glFrustum(left, right, bot, top, near, far);
or
glPerspective(fovy, aspect, near, far);
```

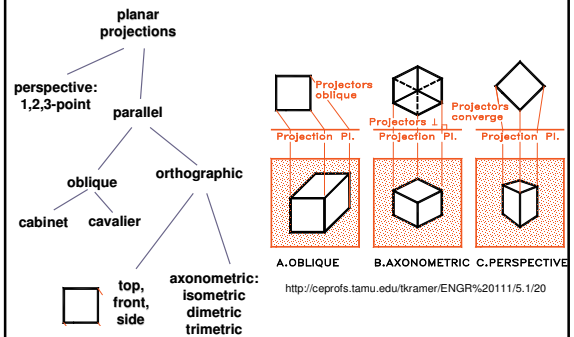
36

## Demo: Frustum vs. FOV

- n Nate Robins tutorial (take 2):
- n <http://www.xmission.com/~nate/tutors.html>

37

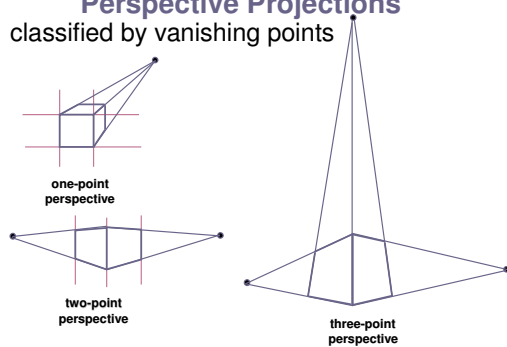
## Projection Taxonomy



38

## Perspective Projections

- n classified by vanishing points



39

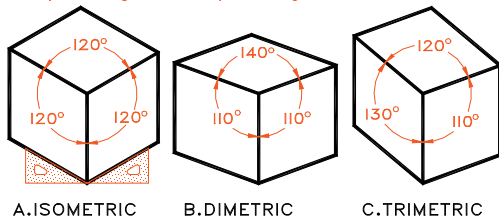
## Parallel Projection

- n projectors are all parallel
- n vs. perspective projectors that converge
- n orthographic: projectors perpendicular to projection plane
- n oblique: projectors not necessarily perpendicular to projection plane



## Axonometric Projections

- n projectors perpendicular to image plane
  - n select axis lengths
- |                |                |                |
|----------------|----------------|----------------|
| 3 Equal axes   | 2 Equal axes   | 0 Equal axes   |
| 3 Equal angles | 2 Equal angles | 0 Equal angles |

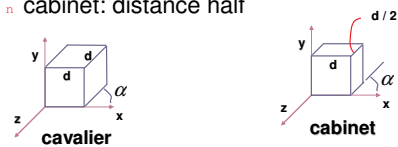


<http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20>

-1

## Oblique Projections

- n projectors oblique to image plane
- n select angle between front and z axis
- n lengths remain constant
- n both have true front view
- n cavalier: distance true
- n cabinet: distance half



42

## Demos

- n Tuebingen applets from Frank Hanisch
  - <http://www.gis.uni-tuebingen.de/projects/grdev/doc/html/etc/AppletIndex.html#Transformationen>

43