University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2010

Tamara Munzner

Viewing/Projection IV

Week 4, Fri Jan 29

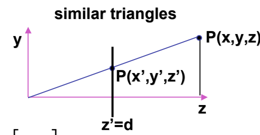
http://www.ugrad.cs.ubc.ca/~cs314/Vjan2010

News

- extra TA office hours in lab 005
  - Fri 2-4 (Garrett)
- Tamara's usual office hours in lab
  - Fri 4-5
- hand in H1 here/now or in box next to 005 lab by 5pm
  - correction: problem 6 worth 54 not 60 marks

2

Review: Basic Perspective Projection



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

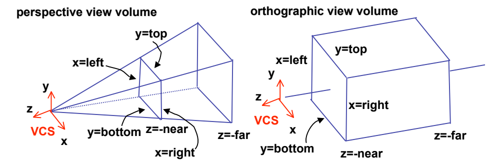
$$x' = \frac{x \cdot d}{z} \qquad z' = d$$

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

3

Review: View Volumes

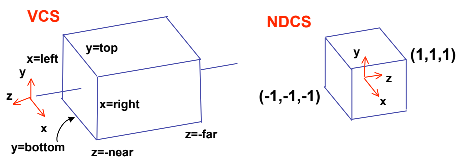
- specifies field-of-view, used for clipping
- restricts domain of ***z*** stored for visibility test



4

Review: Understanding Z

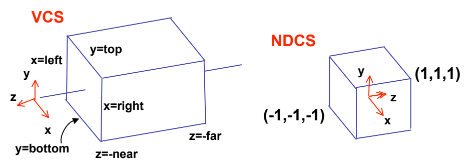
- z axis flip changes coord system handedness
  - RHS before projection (eye/view coords)
  - LHS after projection (clip, norm device coords)



5

Review: Orthographic Derivation

- scale, translate, reflect for new coord sys



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} e & 0 & 0 & f \\ 0 & a & 0 & b \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$y' = a \cdot y + b$$
$$y = top \rightarrow y' = 1$$
$$y = bot \rightarrow y' = -1$$

$$a = \frac{2}{top - bot}$$
$$b = -\frac{top + bot}{top - bot}$$

6

Review: Orthographic Derivation

- scale, translate, reflect for new coord sys

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

7

Demo

- Robins demo: projection
  - orthographic
  - perspective

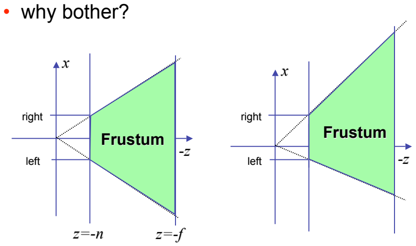
8

Projections II

9

Asymmetric Frusta

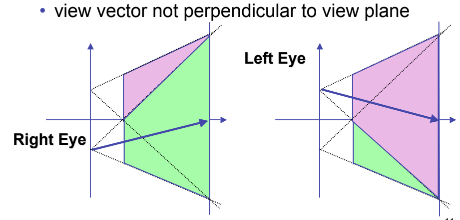
- our formulation allows asymmetry
  - why bother?



10

Asymmetric Frusta

- our formulation allows asymmetry
  - why bother? binocular stereo
    - view vector not perpendicular to view plane



11

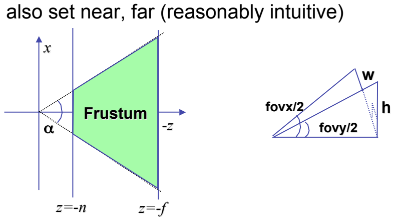
Simpler Formulation

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

12

Field-of-View Formulation

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



13

Perspective OpenGL

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

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

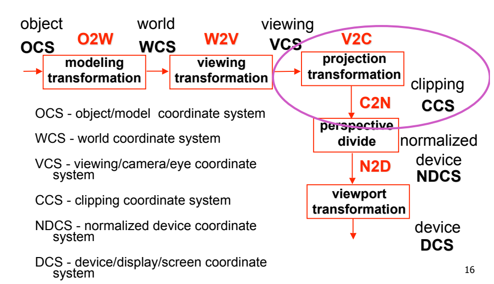
14

Demo: Frustum vs. FOV

- Nate Robins tutorial (take 2): projection
  - frustum vs perspective

15

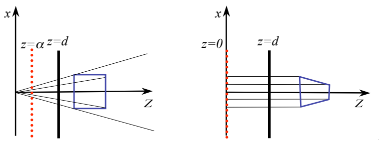
Projective Rendering Pipeline



16

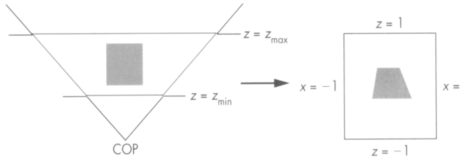
Perspective Warp

- warp perspective view volume to orthogonal view volume
  - render all scenes with orthographic projection!
  - aka perspective normalization

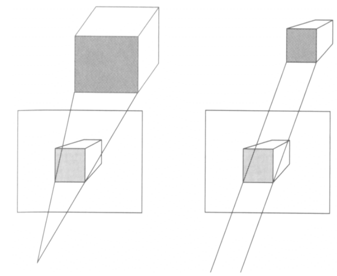


Perspective Warp

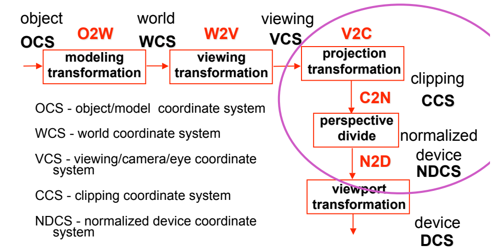
- perspective viewing frustum transformed to cube
- orthographic rendering of warped objects in cube produces same image as perspective rendering of original frustum



Predistortion



Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

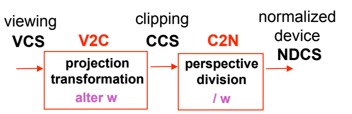
VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

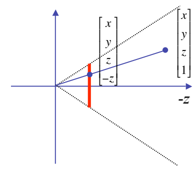
Separate Warp From Homogenization



- warp requires only standard matrix multiply
  - distort such that orthographic projection of distorted objects shows desired perspective projection
    - w is changed
  - clip after warp, before divide
  - division by w: homogenization

Perspective Divide Example

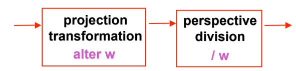
- specific example
- assume image plane at $z = -1$
- a point $[x,y,z,1]^T$ projects to $[-x/z, -y/z, -z/z, 1]^T \equiv [x, y, z, -z]^T$



Perspective Divide Example

$$T\begin{pmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\end{pmatrix} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&-1&0\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix} = \begin{bmatrix}x\\y\\z\\-z\end{bmatrix} = \begin{bmatrix}-x/z\\-y/z\\-1\\1\end{bmatrix}$$

- after homogenizing, once again w=1



Perspective Normalization

- matrix formulation

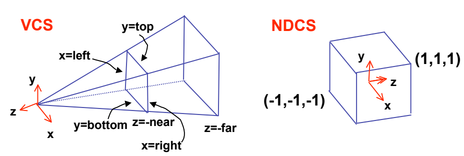
$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&\frac{d}{d-a}&\frac{-a\cdot d}{d-a}\\0&0&\frac{1}{d}&0\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix} = \begin{bmatrix}x\\y\\\frac{(z-a)\cdot d}{d-a}\\\frac{z}{d}\end{bmatrix} \qquad \begin{bmatrix}x_p\\y_p\\z_p\end{bmatrix} = \begin{bmatrix}\frac{x}{z/d}\\\frac{y}{z/d}\\\frac{d^2}{d-a}\left(1-\frac{a}{z}\right)\end{bmatrix}$$

- warp and homogenization both preserve relative depth (z coordinate)

Demo

- Brown applets: viewing techniques
  - parallel/orthographic cameras
  - projection cameras
- http://www.cs.brown.edu/exploratories/freeSoftware/catalogs/viewing_techniques.html

Perspective To NDCS Derivation



Perspective Derivation

simple example earlier:

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

complete: shear, scale, projection-normalization

$$\begin{bmatrix}x'\\y'\\z'\\w'\end{bmatrix} = \begin{bmatrix}E&0&A&0\\0&F&B&0\\0&0&C&D\\0&0&-1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Perspective Derivation

earlier:

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

complete: shear, scale, projection-normalization

$$\begin{bmatrix}x'\\y'\\z'\\w'\end{bmatrix} = \begin{bmatrix}E&0&A&0\\0&F&B&0\\0&0&C&D\\0&0&-1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Perspective Derivation

earlier:

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

complete: shear, scale, projection-normalization

$$\begin{bmatrix}x'\\y'\\z'\\w'\end{bmatrix} = \begin{bmatrix}E&0&A&0\\0&F&B&0\\0&0&C&D\\0&0&-1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Perspective Derivation

$$\begin{bmatrix}x'\\y'\\z'\\w'\end{bmatrix} = \begin{bmatrix}E&0&A&0\\0&F&B&0\\0&0&C&D\\0&0&-1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} \qquad \begin{aligned}x' &= Ex + Az\\y' &= Fy + Bz\\z' &= Cz + D\\w' &= -z\end{aligned}$$

$$\begin{aligned}x = left &\rightarrow x'/w' = 1\\x = right &\rightarrow x'/w' = -1\\y = top &\rightarrow y'/w' = 1\\y = bottom &\rightarrow y'/w' = -1\\z = -near &\rightarrow z'/w' = 1\\z = -far &\rightarrow z'/w' = -1\end{aligned}$$

$$y' = Fy + Bz, \quad \frac{y'}{w'} = \frac{Fy + Bz}{w'}, \quad 1 = \frac{Fy + Bz}{w'}, \quad 1 = \frac{Fy + Bz}{-z},$$

$$1 = F\frac{y}{-z} + B\frac{z}{-z}, \quad 1 = F\frac{y}{-z} - B, \quad 1 = F\frac{top}{-(-near)} - B,$$

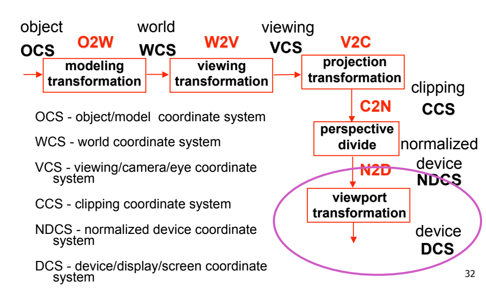
$$1 = F\frac{top}{near} - B$$

Perspective Derivation

- similarly for other 5 planes
- 6 planes, 6 unknowns

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

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

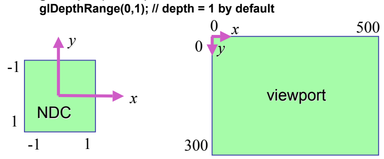
NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

NDC to Device Transformation

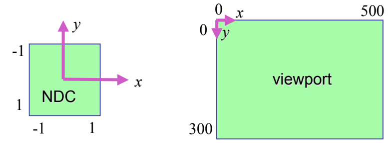
- map from NDC to pixel coordinates on display
  - NDC range is x = -1...1, y = -1...1, z = -1...1
  - typical display range: x = 0...500, y = 0...300
    - maximum is size of actual screen
    - z range max and default is (0, 1), use later for visibility

```
glViewport(0,0,w,h);
glDepthRange(0,1); // depth = 1 by default
```



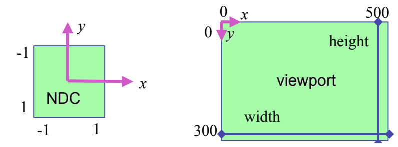
Origin Location

- yet more (possibly confusing) conventions
  - OpenGL origin: lower left
  - most window systems origin: upper left
- then must reflect in y
- when interpreting mouse position, have to flip your y coordinates



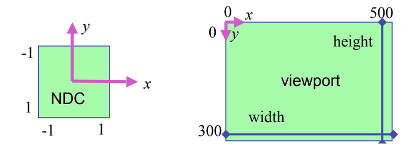
N2D Transformation

- general formulation
  - reflect in y for upper vs. lower left origin
  - scale by width, height, depth
  - translate by width/2, height/2, depth/2
    - FCG includes additional translation for pixel centers at (.5, .5) instead of (0,0)



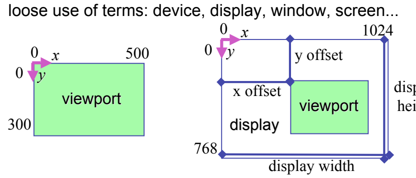
N2D Transformation

$$\begin{bmatrix} x_D \\ y_D \\ z_D \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \frac{width}{2} - \frac{1}{2} \\ 0 & 1 & 0 & \frac{height}{2} - \frac{1}{2} \\ 0 & 0 & 1 & \frac{depth}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{width}{2} & 0 & 0 & 0 \\ 0 & \frac{height}{2} & 0 & 0 \\ 0 & 0 & \frac{depth}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_N \\ y_N \\ z_N \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{width(x_N+1)-1}{2} \\ \frac{height(-y_N+1)-1}{2} \\ \frac{depth(z_N+1)}{2} \\ 1 \end{bmatrix}$$

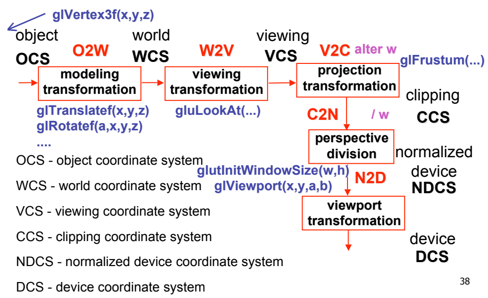


Device vs. Screen Coordinates

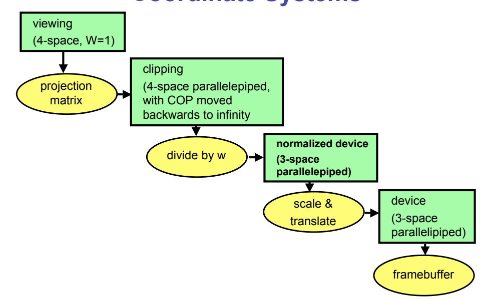
- viewport/window location wrt actual display not available within OpenGL
  - usually don't care
    - use relative information when handling mouse events, not absolute coordinates
  - could get actual display height/width, window offsets from OS
- loose use of terms: device, display, window, screen...



Projective Rendering Pipeline



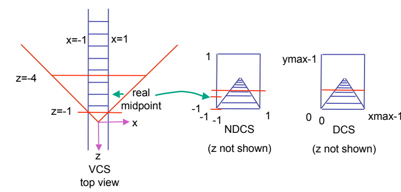
Coordinate Systems



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



Perspective Example

view volume

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

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

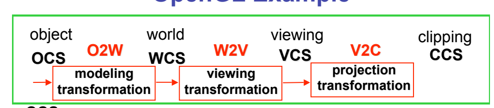
Perspective Example

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

/ w

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

OpenGL Example




```
CCS   glMatrixMode( GL_PROJECTION );
      glLoadIdentity();
      gluPerspective( 45, 1.0, 0.1, 200.0 );
VCS   glMatrixMode( GL_MODELVIEW );
      glLoadIdentity();
      glTranslatef( 0.0, 0.0, -5.0 );
WCS   glPushMatrix()
      glTranslate( 4, 4, 0 );  W2O
OCS1  glutSolidTeapot(1);
      glPopMatrix();
      glTranslate( 2, 2, 0);  W2O
OCS2  glutSolidTeapot(1);
```

- transformations that are applied to object first are specified last

Reading for Next Time

- RB Chap Color
- FCG Sections 3.2-3.3
- FCG Chap 20 Color
- FCG Chap 21.2.2 Visual Perception (Color)