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

Tamara Munzner

Viewing III

Week 4, Wed Jan 27

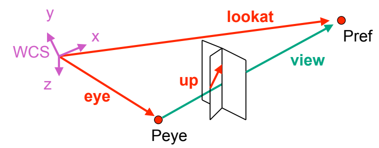
http://www.ugrad.cs.ubc.ca/~cs314/Vjan2010

News: Reminder

- extra TA office hours in lab 005
  - Tue 2-5 (Kai)
  - Wed 2-5 (Garrett)
  - Thu 1-3 (Garrett), Thu 3-5 (Kai)
  - Fri 2-4 (Garrett)
- Tamara's usual office hours in lab
  - Fri 4-5

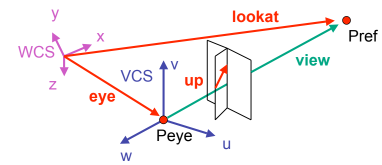
Review: Convenient Camera Motion

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



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



Review: W2V vs. W2V

- M_{W2V}=TR

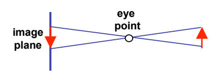
$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- we derived position of camera in world
  - invert for world with respect to camera
- $M_{V2W}=(M_{W2V})^{-1}=R^{-1}T^{-1}$

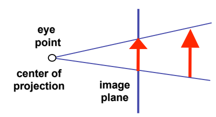
$$\mathbf{M}_{view2world} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & -e_x \\ v_x & v_y & v_z & -e_y \\ w_x & w_y & w_z & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Review: Graphics Cameras

- real pinhole camera: image inverted



- computer graphics camera: convenient equivalent



Review: Projective Transformations

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

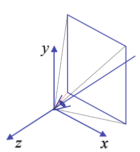
Projective Transformations

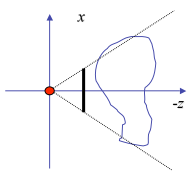
- properties
  - lines mapped to lines and triangles to triangles
  - parallel lines do NOT remain parallel
    - e.g. rails vanishing at infinity

  - affine combinations are NOT preserved
    - e.g. center of a line does not map to center of projected line (perspective foreshortening)

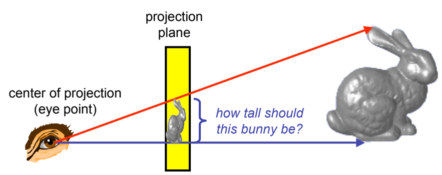
Perspective Projection

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

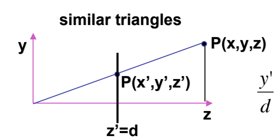




Perspective Projection



Basic Perspective Projection



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

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

- nonuniform foreshortening
  - not affine

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?

Simple Perspective Projection Matrix

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

Simple Perspective Projection Matrix

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

where w = z/d

Simple Perspective Projection Matrix

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

where w = z/d

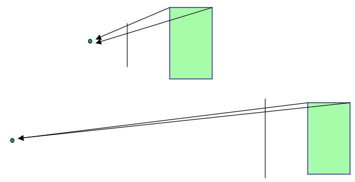
$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Projection

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

Moving COP to Infinity

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



17

Orthographic Camera Projection

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

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

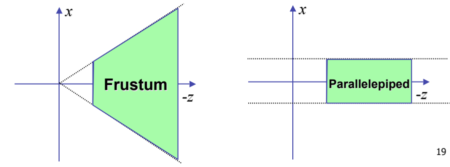
- just throw away z values

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

18

Perspective to Orthographic

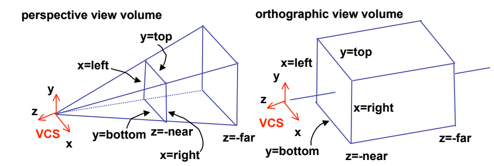
- transformation of space
  - center of projection moves to infinity
  - view volume transformed
    - from frustum (truncated pyramid) to parallelepiped (box)



19

View Volumes

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



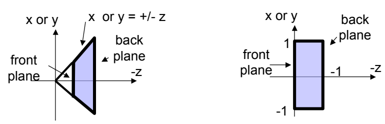
20

Canonical View Volumes

- standardized viewing volume representation

perspective

orthographic
orthogonal
parallel



21

Why Canonical View Volumes?

- permits standardization
  - clipping
    - easier to determine if an arbitrary point is enclosed in volume with canonical view volume vs. clipping to six arbitrary planes
  - rendering
    - projection and rasterization algorithms can be reused

22

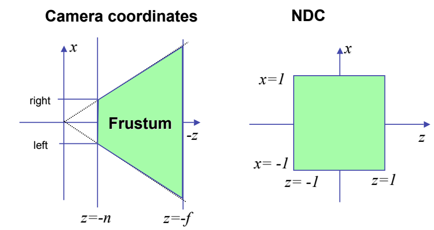
Normalized Device Coordinates

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

23

Normalized Device Coordinates

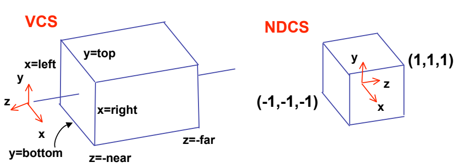
left/right x =+/- 1, top/bottom y =+/- 1, near/far z =+/- 1



24

Understanding Z

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

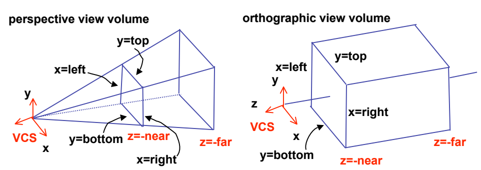


25

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);
```



26

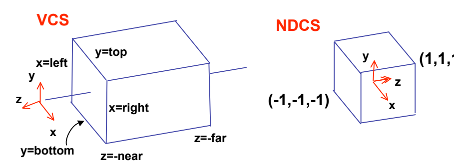
Understanding Z

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

27

Orthographic Derivation

- scale, translate, reflect for new coord sys

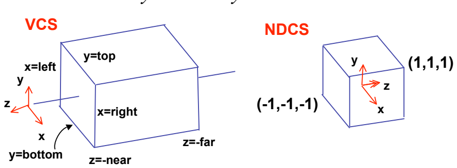


28

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad \begin{aligned} y = top &\rightarrow y' = 1 \\ y = bot &\rightarrow y' = -1 \end{aligned}$$



29

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad \begin{aligned} y = top &\rightarrow y' = 1 \\ y = bot &\rightarrow y' = -1 \end{aligned} \qquad \begin{aligned} 1 &= a \cdot top + b \\ -1 &= a \cdot bot + b \end{aligned}$$

$$b = 1 - a \cdot top, b = -1 - a \cdot bot$$
$$1 - a \cdot top = -1 - a \cdot bot$$
$$1 - (-1) = -a \cdot bot - (-a \cdot top)$$
$$2 = a(-bot + top)$$
$$a = \frac{2}{top - bot}$$

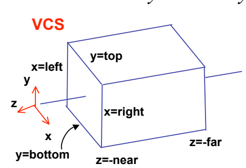
$$1 = \frac{2}{top - bot} top + b$$
$$b = 1 - \frac{2 \cdot top}{top - bot}$$
$$b = \frac{(top - bot) - 2 \cdot top}{top - bot}$$
$$b = \frac{-top - bot}{top - bot}$$

30

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad \begin{aligned} y = top &\rightarrow y' = 1 \\ y = bot &\rightarrow y' = -1 \end{aligned}$$



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

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

same idea for right/left, far/near

31

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

32

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

Orthographic OpenGL

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

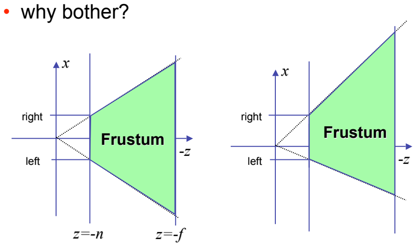

Demo

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

Projections II

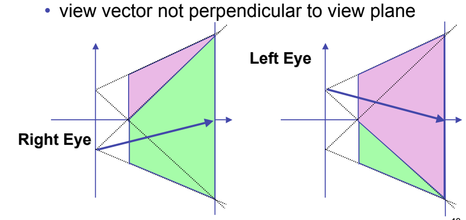
Asymmetric Frusta

- our formulation allows asymmetry
- why bother?



Asymmetric Frusta

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

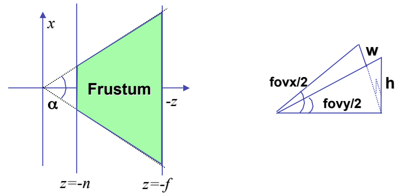


Simpler Formulation

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

Field-of-View Formulation

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



Perspective OpenGL

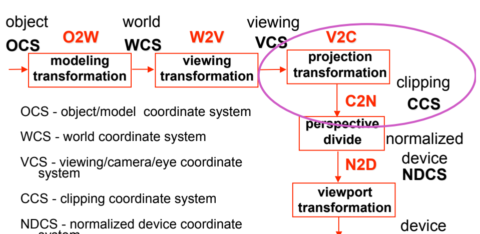
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

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


Demo: Frustum vs. FOV

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

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

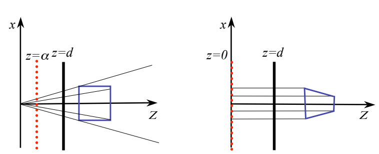
CCS - clipping coordinate system

NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

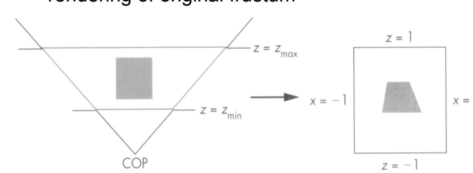
Perspective Warp

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

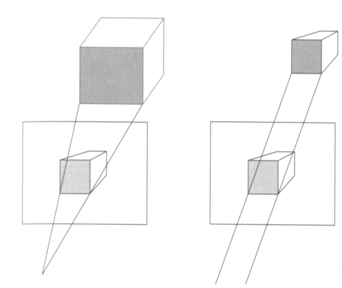


Perspective Warp

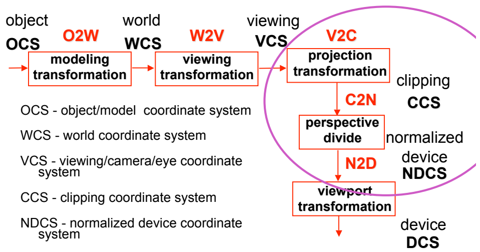
- perspective viewing frustum transformed to cube
- orthographic rendering of warped objects in cube produces same image as perspective rendering of original frustum



Predistortion



Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

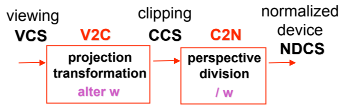
VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

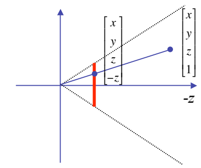
Separate Warp From Homogenization



- 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

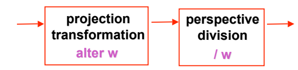
- specific example
  - assume image plane at $z = -1$
  - a point $[x,y,z,1]^T$ projects to $[-x/z,-y/z,-z/z,1]^T \equiv [x,y,z,-z]^T$



Perspective Divide Example

$$T\left(\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\right)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&-1&0\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}=\begin{bmatrix}x\\y\\z\\-z\end{bmatrix}=\begin{bmatrix}-x/z\\-y/z\\-1\\1\end{bmatrix}$$

- 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