University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2016

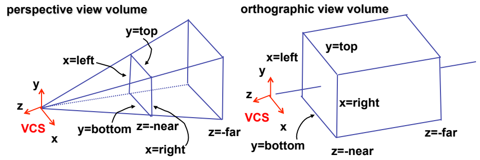
Tamara Munzner

Viewing 3

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2016

View Volumes

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

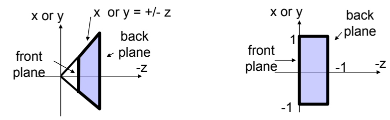


Canonical View Volumes

- standardized viewing volume representation

perspective

orthographic
orthogonal
parallel



Why Canonical View Volumes?

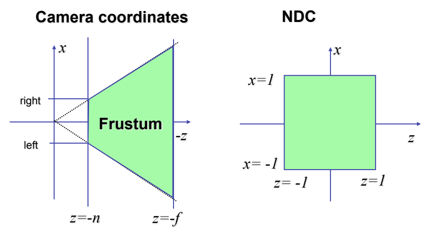
- permits standardization
  - clipping
    - easier to determine if an arbitrary point is enclosed in volume with canonical view volume vs. clipping to six arbitrary planes
  - rendering
    - projection and rasterization algorithms can be reused

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

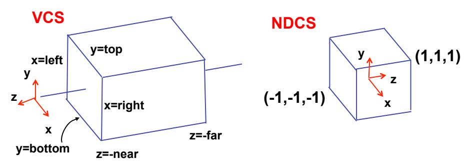
Normalized Device Coordinates

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



Understanding Z

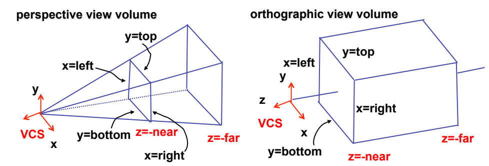
- z axis flip changes coord system handedness
  - RHS before projection (eye/view coords)
  - LHS after projection (clip, norm device coords)



Understanding Z

near, far always positive in GL calls

THREE.OrthographicCamera(left,right,bot,top,near,far);
mat4.frustum(left,right,bot,top,near,far, *projectionMatrix*);

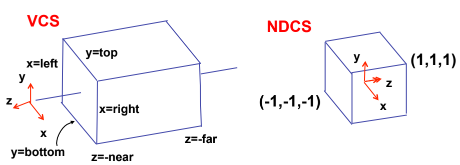


Understanding Z

- 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

Orthographic Derivation

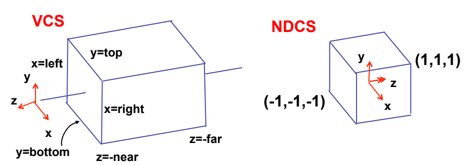
- scale, translate, reflect for new coord sys



Orthographic Derivation

- scale, translate, reflect for new coord sys

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



Orthographic Derivation

- scale, translate, reflect for new coord sys

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

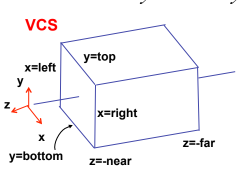
$$b = 1 - a \cdot top, b = -1 - a \cdot bot$$
$$1 - a \cdot top = -1 - a \cdot bot$$
$$1 - (-1) = -a \cdot bot - (-a \cdot top)$$
$$2 = a(-bot + top)$$
$$a = \frac{2}{top - bot}$$

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

Orthographic Derivation

- scale, translate, reflect for new coord sys

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



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

same idea for right/left, far/near

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

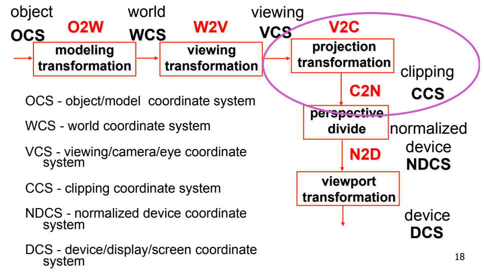
$$P' = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & -\frac{right + left}{right - left} \\ 0 & \frac{2}{top - bot} & 0 & -\frac{top + bot}{top - bot} \\ 0 & 0 & \frac{-2}{far - near} & -\frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

Orthographic Derivation

- scale, translate, reflect for new coord sys

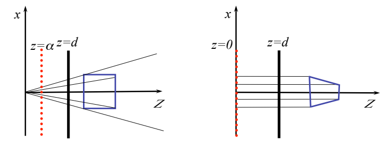
$$P' = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bot} & 0 & -\frac{top+bot}{top-bot} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} P$$

Projective Rendering Pipeline



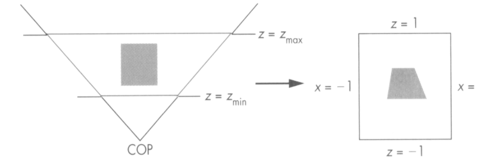
Projection Warp

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

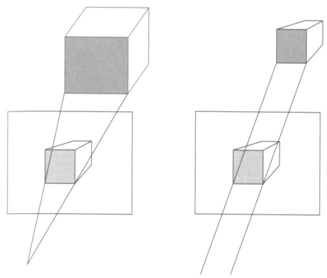


Perspective Warp

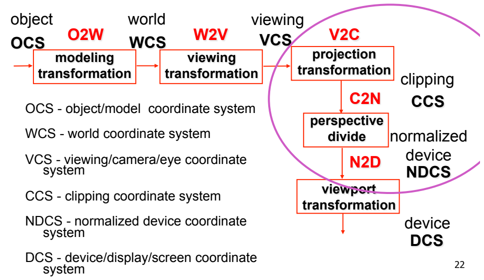
- perspective viewing frustum transformed to cube
- orthographic rendering of cube produces same image as perspective rendering of original



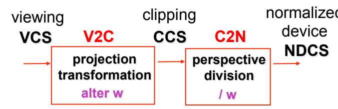
Predistortion



Projective Rendering Pipeline



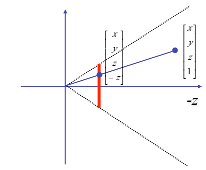
Separate Warp From Homogenization



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

Perspective Divide Example

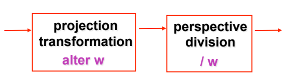
- specific example
- assume image plane at $z = -1$
- a point $[x,y,z,1]^T$ projects to $[-x/z,-y/z,-z/z,1]^T \equiv [x,y,z,-z]^T$



Perspective Divide Example

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

- after homogenizing, once again w=1



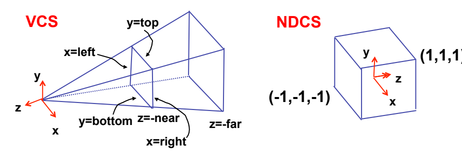
Perspective Normalization

- matrix formulation

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

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

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} E & 0 & A & 0 \\ 0 & F & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{aligned} x' &= Ex + Az \\ y' &= Fy + Bz \\ z' &= Cz + D \\ w' &= -z \end{aligned} \qquad \begin{aligned} &x = left \rightarrow x'/w' = -1 \\ &x = right \rightarrow x'/w' = 1 \\ &y = top \rightarrow y'/w' = 1 \\ &y = bottom \rightarrow y'/w' = -1 \\ &z = -near \rightarrow z'/w' = -1 \\ &z = -far \rightarrow z'/w' = 1 \end{aligned}$$

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

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

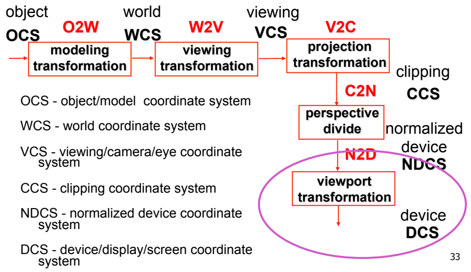
$$1 = F\frac{top}{near} - B$$

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

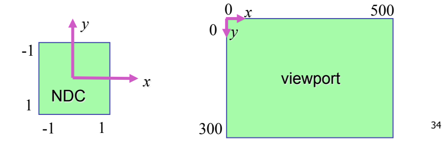
NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

NDC to Device Transformation

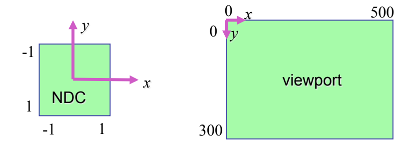
- map from NDC to pixel coordinates on display
  - NDC range is x = -1...1, y = -1...1, z = -1...1
  - typical display range: x = 0...500, y = 0...300
    - maximum is size of actual screen
    - z range max and default is (0, 1), use later for visibility

```
gl.viewport(0,0,w,h);
gl.depthRange(0,1); // depth = 1 by default
```



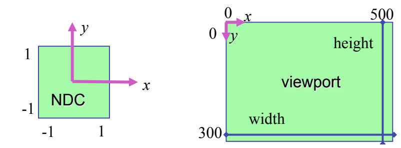
Origin Location

- yet more (possibly confusing) conventions
  - GL 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)

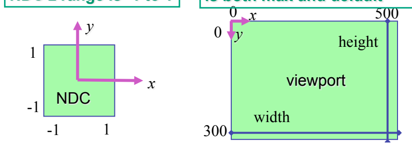


N2D Transformation

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

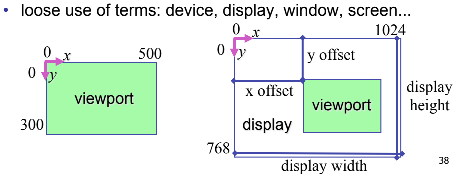
Display z range is 0 to 1. gl.depthRange(n,f) can constrain further, but *depth* = 1 is both max and default

reminder: NDC z range is -1 to 1

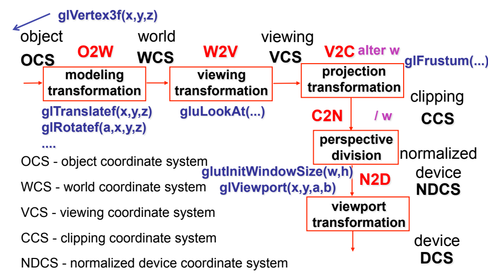


Device vs. Screen Coordinates

- viewport/window location wrt actual display not available within GL
  - 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



OCS - object coordinate system

WCS - world coordinate system

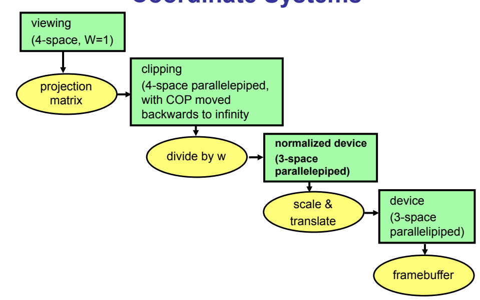
VCS - viewing coordinate system

CCS - clipping coordinate system

NDCS - normalized device coordinate system

DCS - device coordinate system

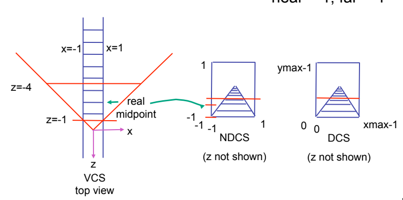
Coordinate Systems



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

$$x_{NDCS} = -1/z_{VCS}$$

$$y_{NDCS} = 1/z_{VCS}$$

$$z_{NDCS} = \frac{5}{3} + \frac{8}{3z_{VCS}}$$