University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2007

Tamara Munzner

Viewing/Projections IV

Week 4, Fri Feb 2

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2007

Reading for Today

- FCG Chapter 7 Viewing
- FCG Section 6.3.1 Windowing Transforms

- RB rest of Chap Viewing
- RB rest of App Homogeneous Coords

Reading for Next Time

- RB Chap Color

- FCG Sections 3.2-3.3
- FCG Chap 20 Color
- FCG Chap 21 Visual Perception

News

- my office hours reminder (in lab): today 11-12
- homework 1 due 3pm in box marked 314
  - same hallway as lab
- project 1 due 6pm, electronic handin
  - no hardcopy required
  - demo signup sheet going around again
    - Mon 1-12; Tue 10-12:30, 4-6; Wed 10-12, 2:30-4
    - arrive in lab 10 min before for your demo slot
    - be logged in and ready to go at your time
    - note to those developing in Windows/Mac
      - your program **must** compile and run on lab machines
      - test before the last minute, no changes after handin

Homework 1 News

- don’t forget problem 11 (on back of page)

- Problem 3 is now extra credit
  - Write down the 4x4 matrix for shearing an object by 2 in y and 3 in z.

Correction (Transposed Before): 3D Shear

- shear in x
  - shear due to x along y and z axes
- shear in y
- shear in z
- general shear

$$\begin{bmatrix} 1 & sy & sz & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ sx & 1 & sz & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sx & sy & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$xshear(hxy, hxz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ hxy & 1 & 0 & 0 \\ hxz & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$yshear(hyx, hyz) = \begin{bmatrix} 1 & hyx & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & hyz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$zshear(hzx, hzy) = \begin{bmatrix} 1 & 0 & hzx & 0 \\ 0 & 1 & hzy & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

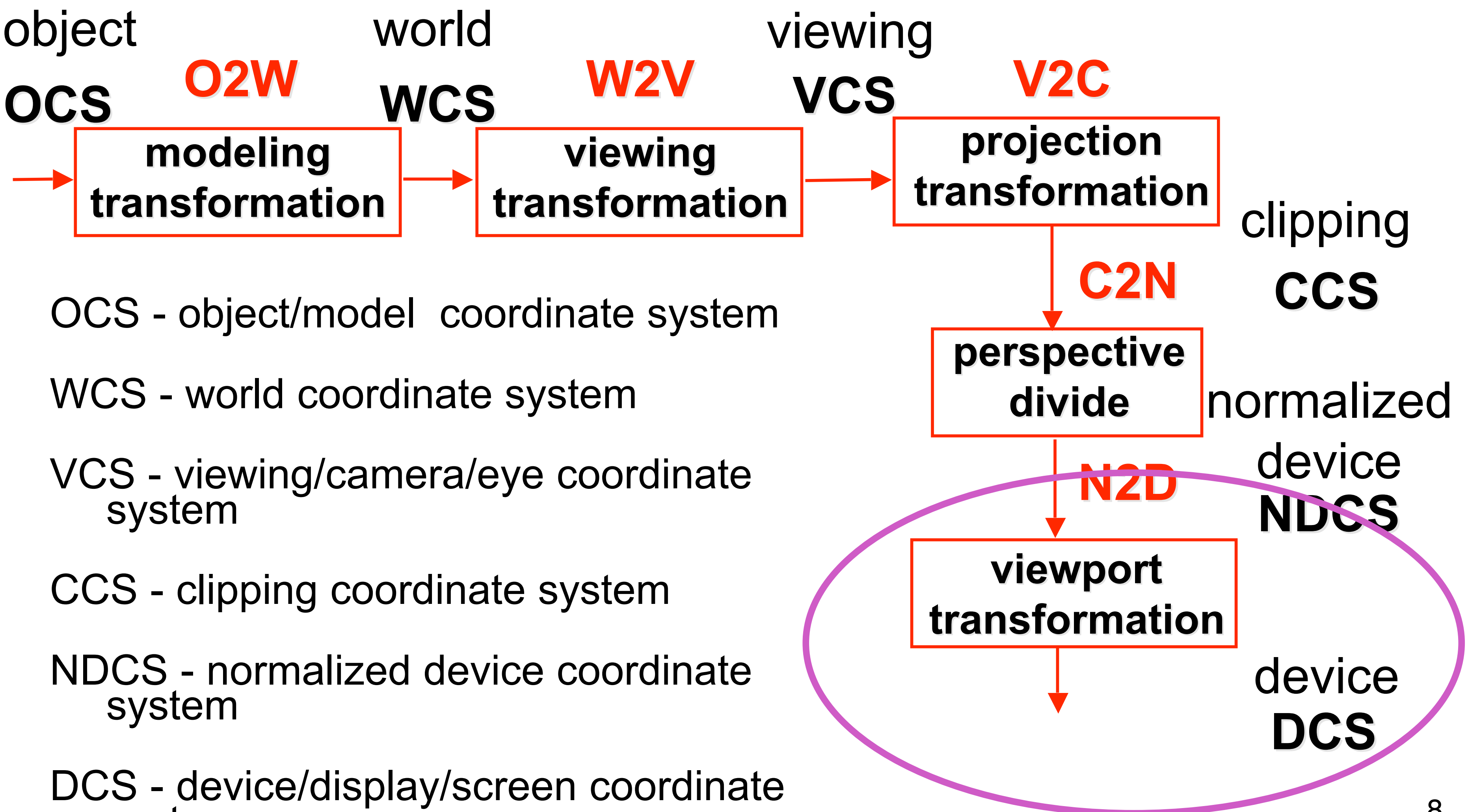
$$shear(hxy, hxz, hyx, hyz, hzx, hzy) = \begin{bmatrix} 1 & hyx & hzx & 0 \\ hxy & 1 & hzy & 0 \\ hxz & hyz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

News

- midterm Friday Feb 9
  - closed book
  - no calculators
  - allowed to have one page of notes
    - handwritten, one side of 8.5x11” sheet
  - this room (DMP 301), 11-11:50

  - material covered
    - transformations, viewing/projection

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

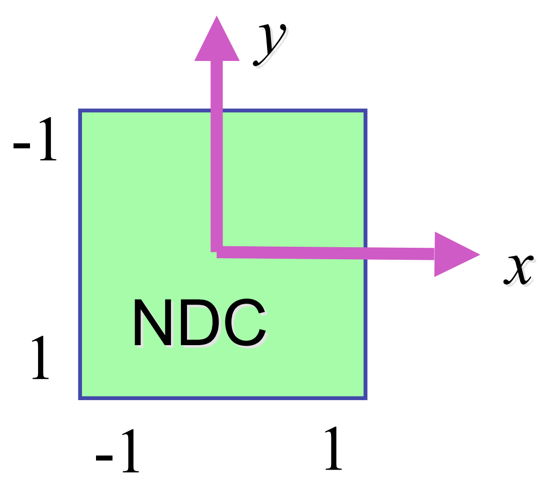
NDCS - normalized device coordinate system

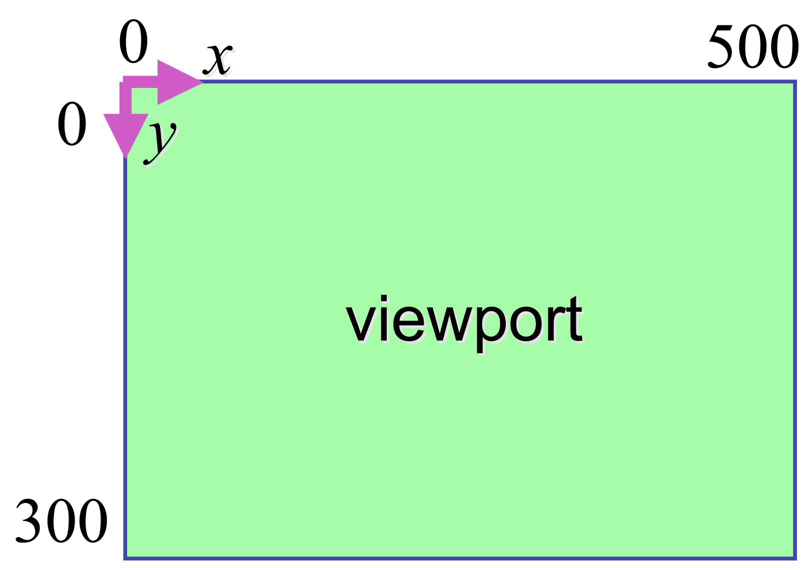
DCS - device/display/screen coordinate system

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

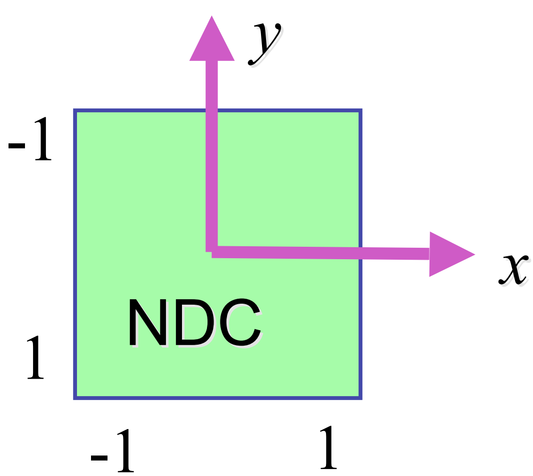
```
glViewport(0,0,w,h);
glDepthRange(0,1); // depth = 1 by default
```

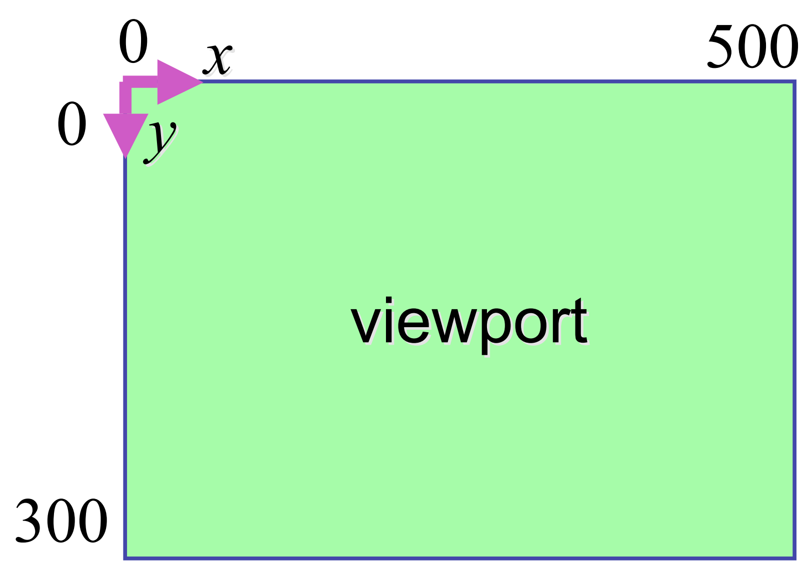




Origin Location

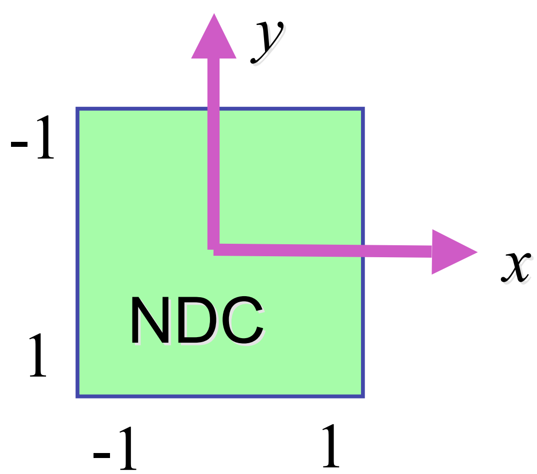
- yet more (possibly confusing) conventions
  - OpenGL origin: lower left
  - most window systems origin: upper left
- then must reflect in y
- when interpreting mouse position, have to flip your y coordinates

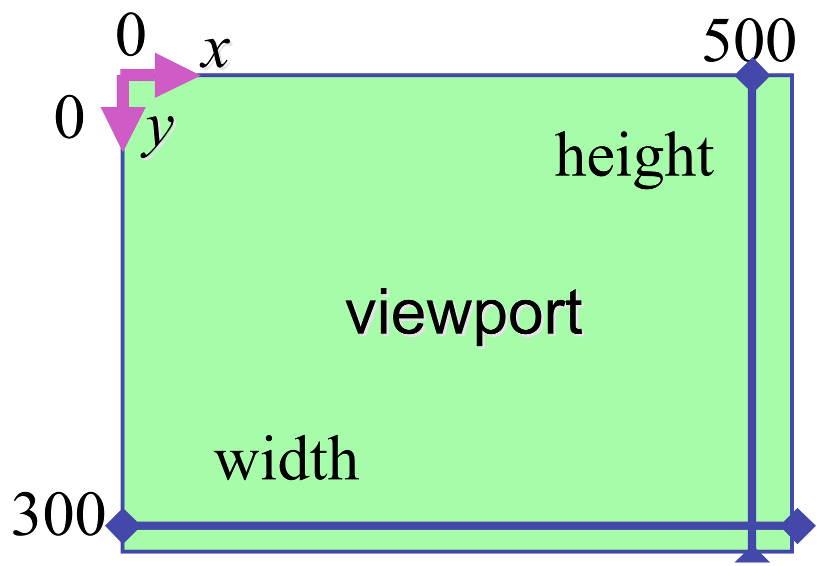




N2D Transformation

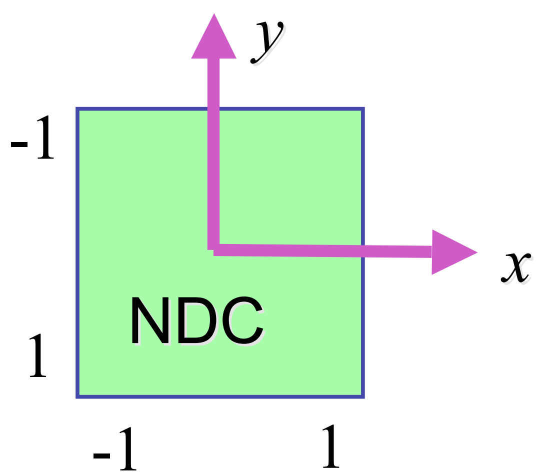
- general formulation
  - reflect in y for upper vs. lower left origin
  - scale by width, height, depth
  - translate by width/2, height/2, depth/2
    - FCG includes additional translation for pixel centers at (.5, .5) instead of (0,0)

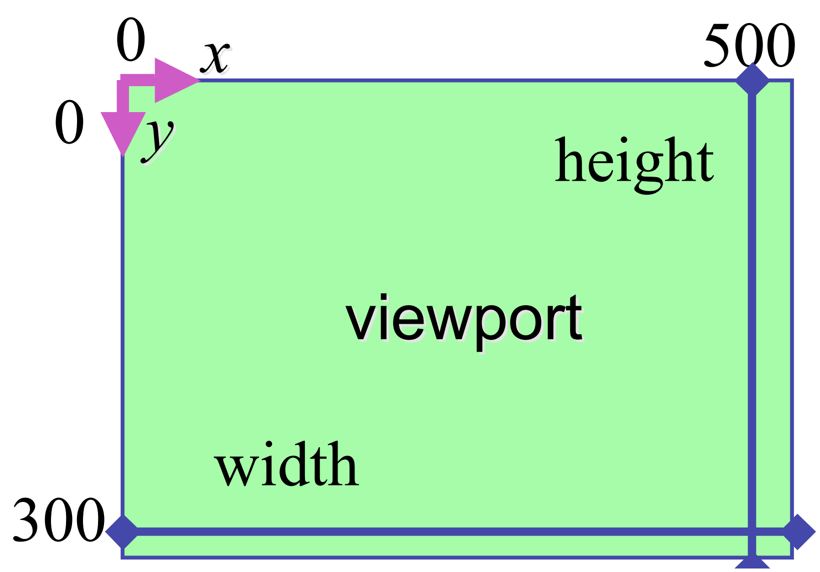




N2D Transformation

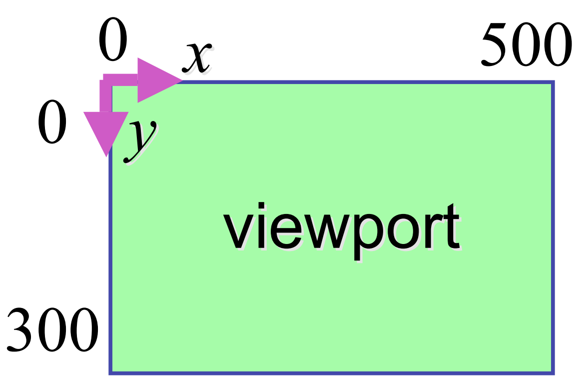
$$
\begin{bmatrix} x_D \\ y_D \\ z_D \\ 1 \end{bmatrix} =
\begin{bmatrix} 1 & 0 & 0 & \frac{width}{2} - \frac{1}{2} \\ 0 & 1 & 0 & \frac{height}{2} - \frac{1}{2} \\ 0 & 0 & 1 & \frac{depth}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} \frac{width}{2} & 0 & 0 & 0 \\ 0 & \frac{height}{2} & 0 & 0 \\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

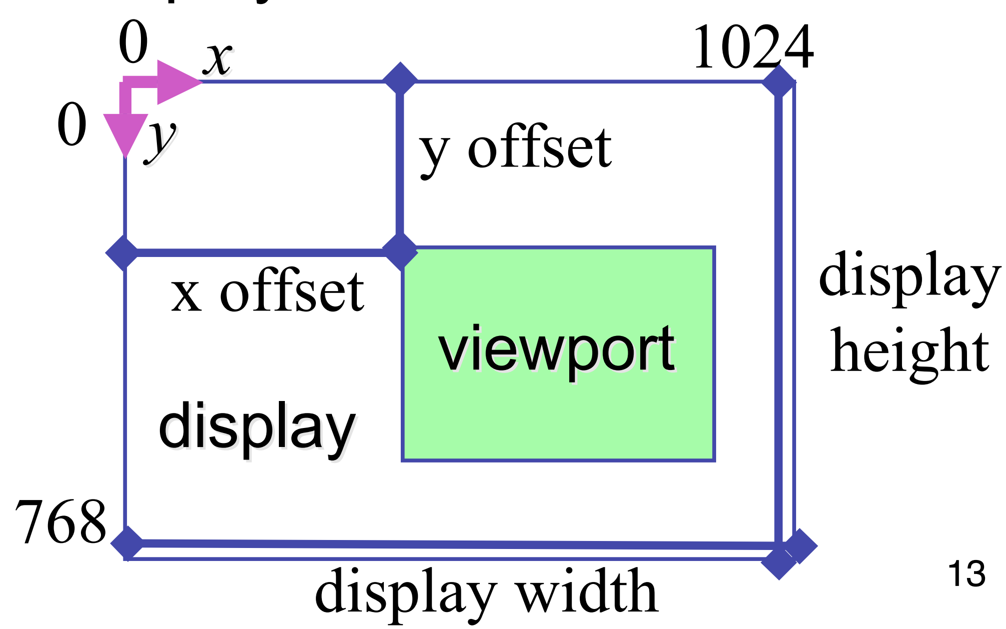




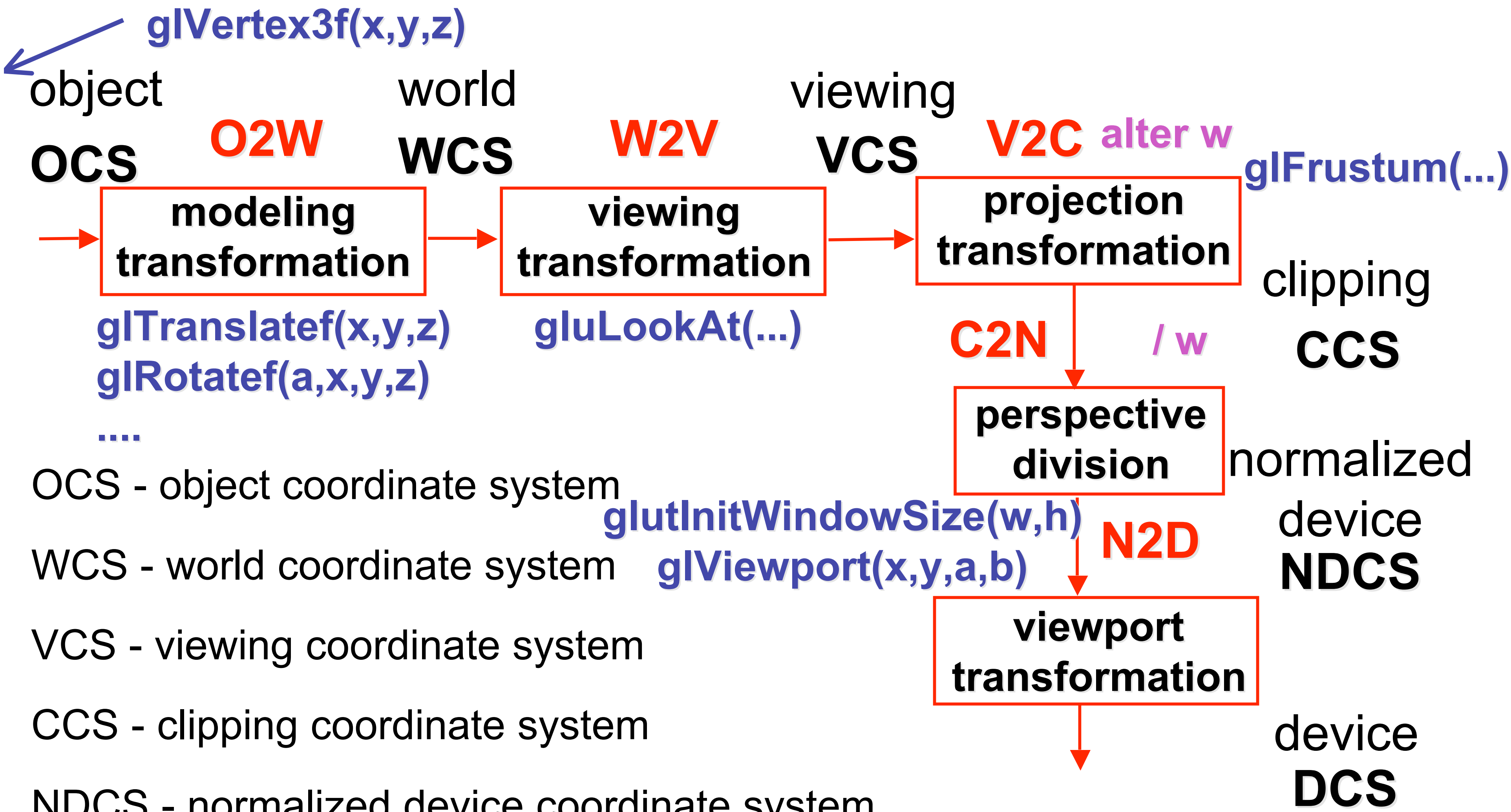
Device vs. Screen Coordinates

- 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



OCS - object coordinate system

WCS - world coordinate system

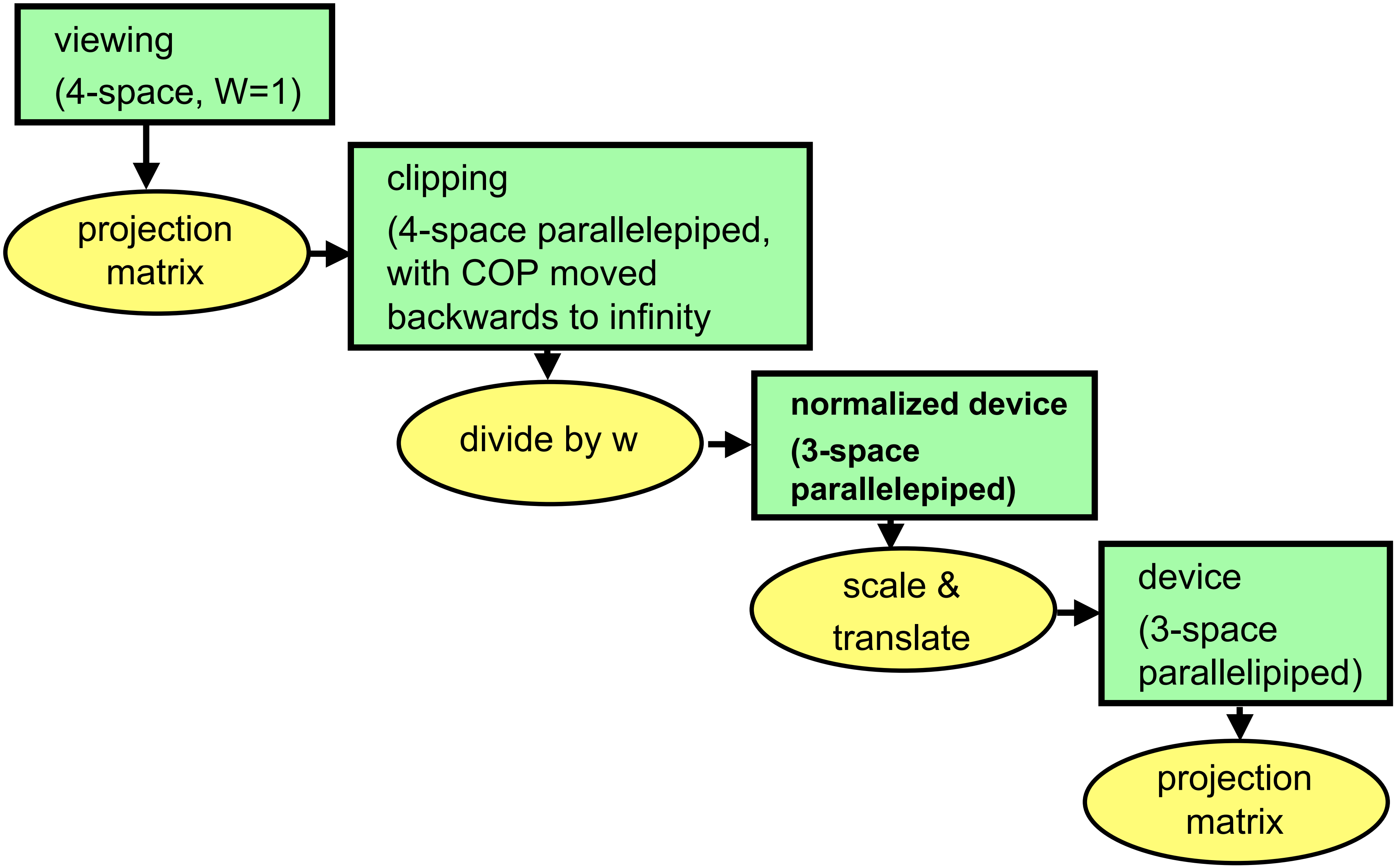
VCS - viewing coordinate system

CCS - clipping coordinate system

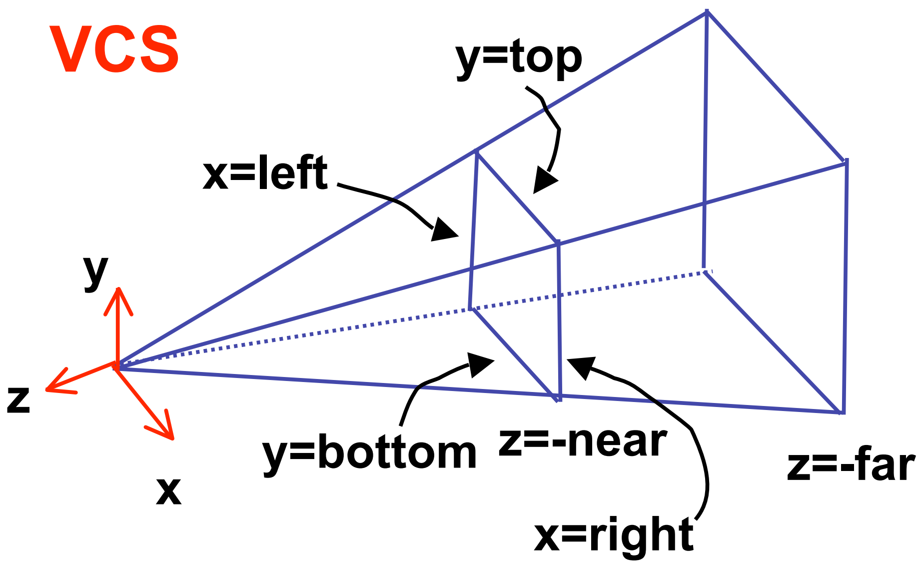
NDCS - normalized device coordinate system

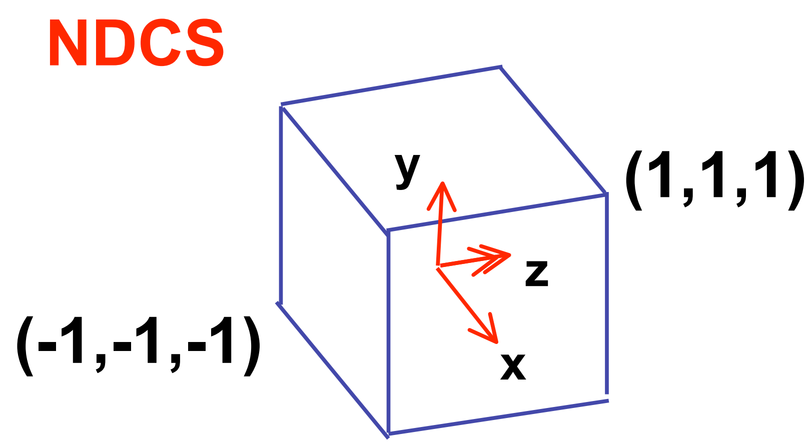
DCS - device coordinate system

Coordinate Systems



Perspective 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

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

$$x' = Ex + Az$$

$$y' = Fy + Bz$$

$$z' = Cz + D$$

$$w' = -z$$

$$x = left \rightarrow x'/w' = 1$$

$$x = right \rightarrow x'/w' = -1$$

$$y = top \rightarrow y'/w' = 1$$

$$y = bottom \rightarrow y'/w' = -1$$

$$z = -near \rightarrow z'/w' = 1$$

$$z = -far \rightarrow z'/w' = -1$$

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

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

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

Perspective Derivation

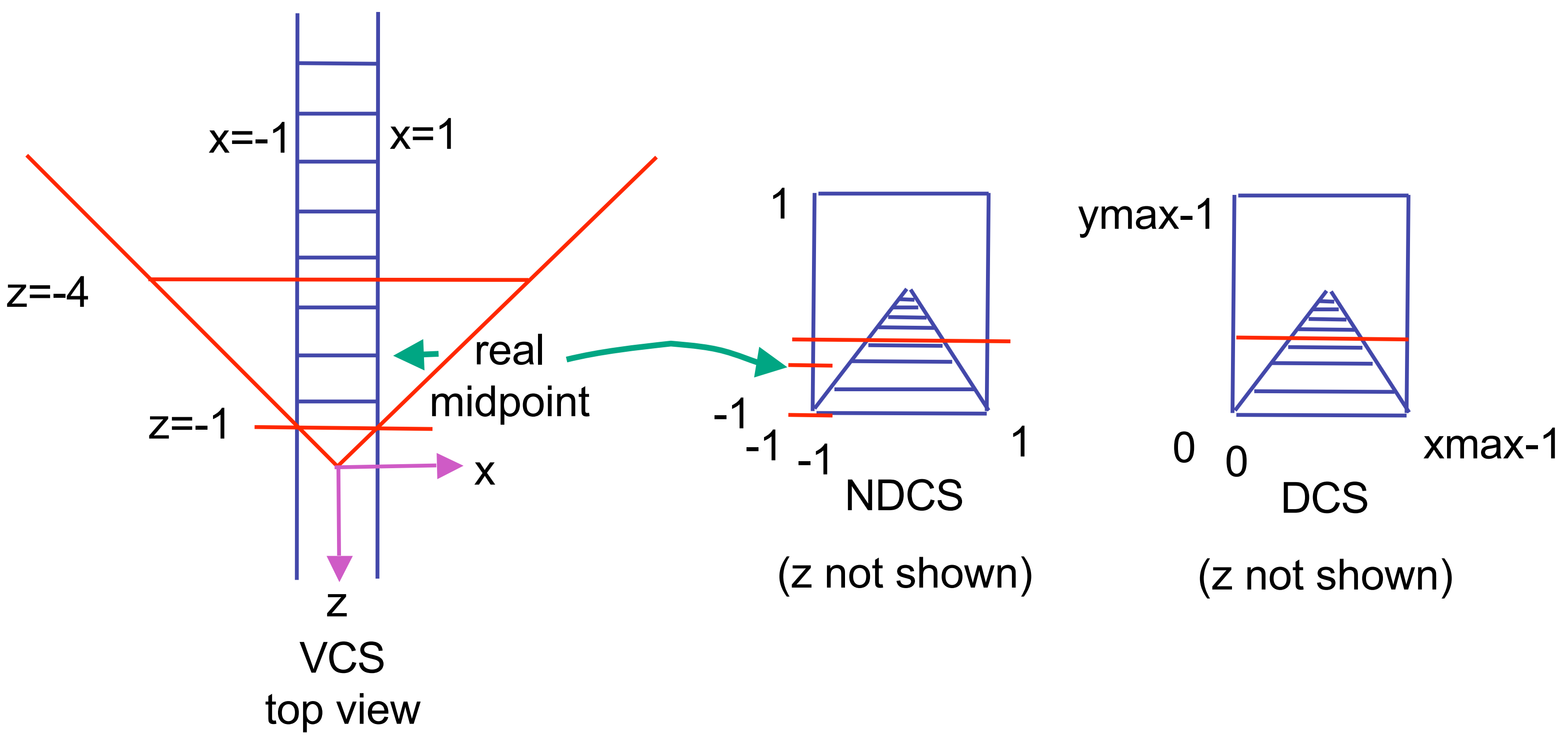
- similarly for other 5 planes
- 6 planes, 6 unknowns

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

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

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

Perspective Example

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

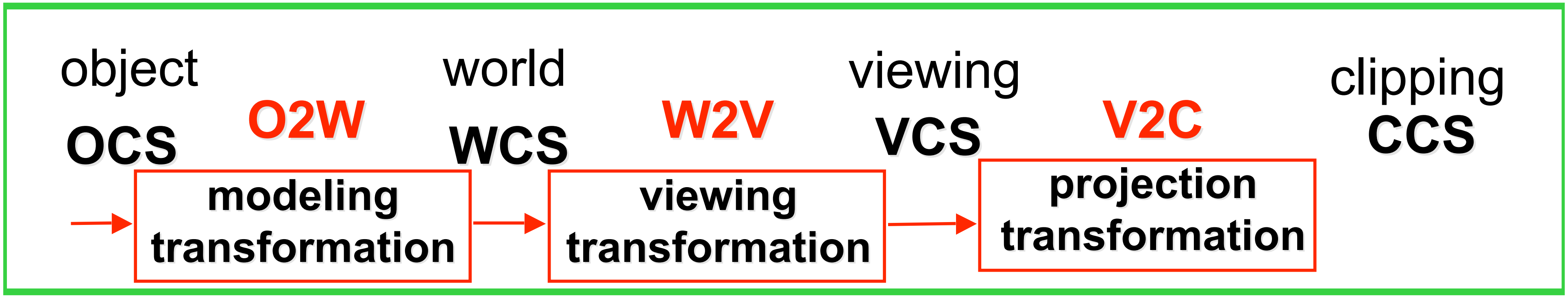
/ w

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

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

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

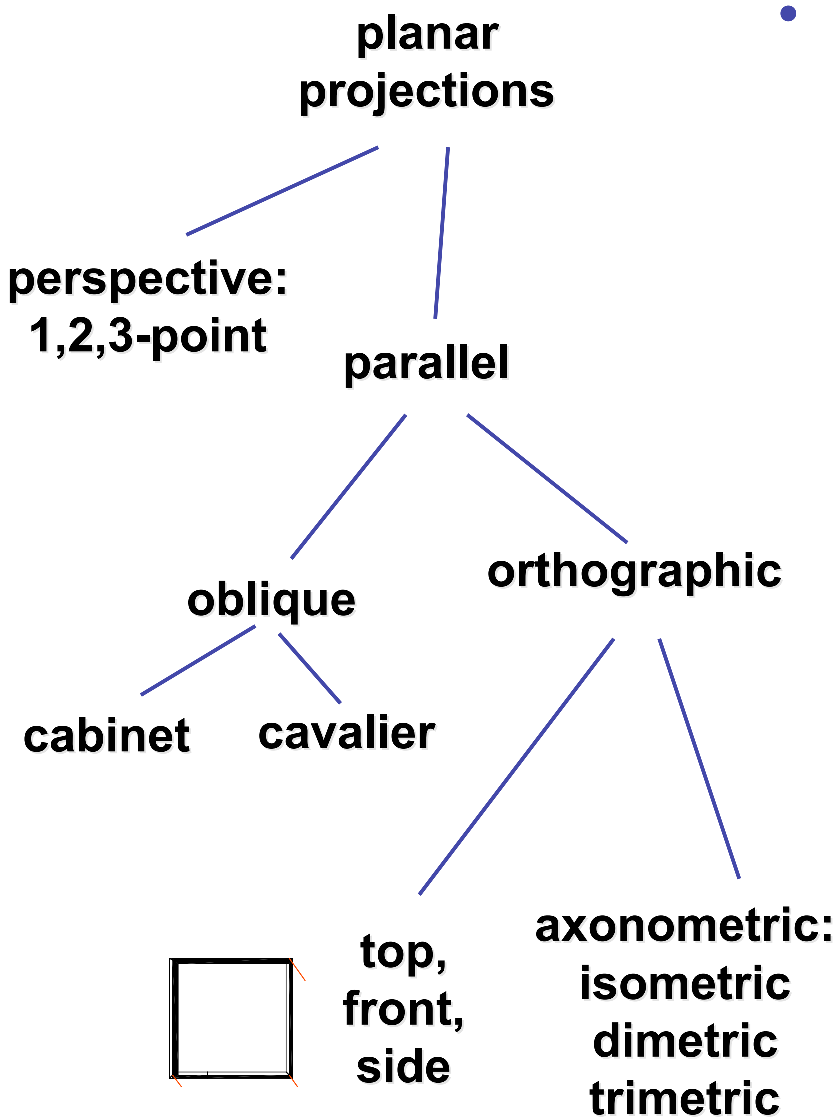
OpenGL Example



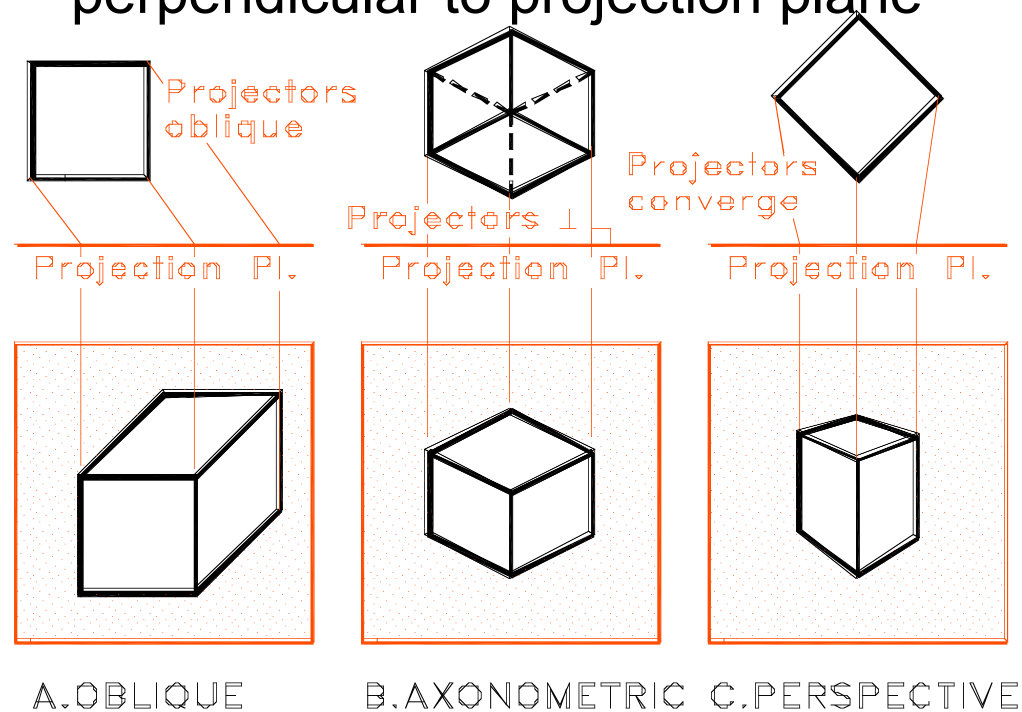

```
CCS   glMatrixMode( GL_PROJECTION );
      glLoadIdentity();
      gluPerspective( 45, 1.0, 0.1, 200.0 );
VCS   glMatrixMode( GL_MODELVIEW );
      glLoadIdentity();
      glTranslatef( 0.0, 0.0, -5.0 );
WCS   glPushMatrix()
      glTranslate( 4, 4, 0 );   W2O
OCS1  glutSolidTeapot(1);
      glPopMatrix();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

- transformations that are applied first are specified last

Projection Taxonomy



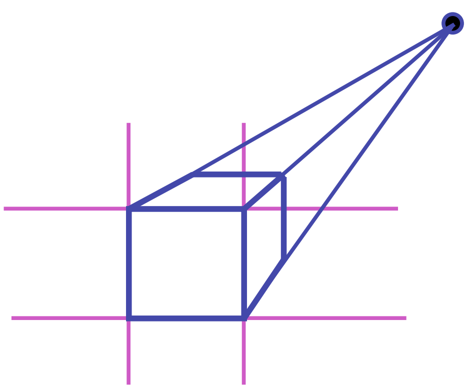
- perspective: projectors converge
  - orthographic, axonometric: projectors parallel and perpendicular to projection plane
  - oblique: projectors parallel, but not perpendicular to projection plane



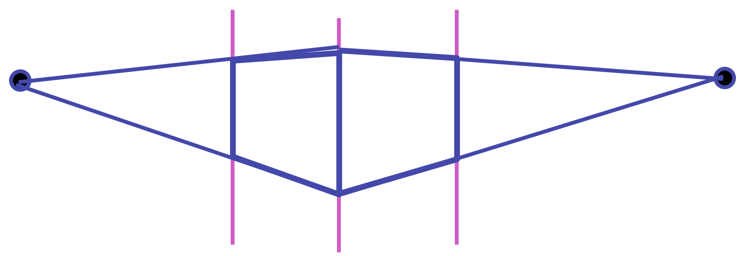
http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20

Perspective Projections

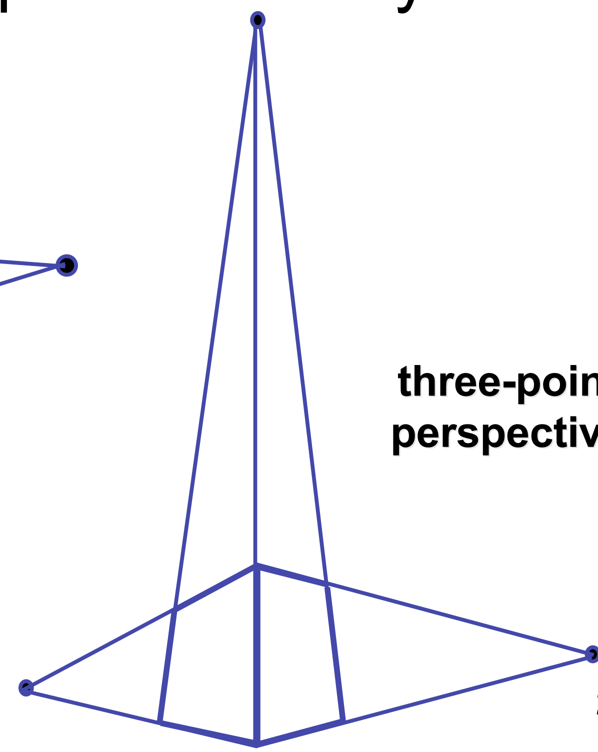
- projectors converge on image plane
- select how many vanishing points
  - one-point: projection plane parallel to two axes
  - two-point: projection plane parallel to one axis
  - three-point: projection plane not parallel to any axis



one-point perspective



two-point perspective

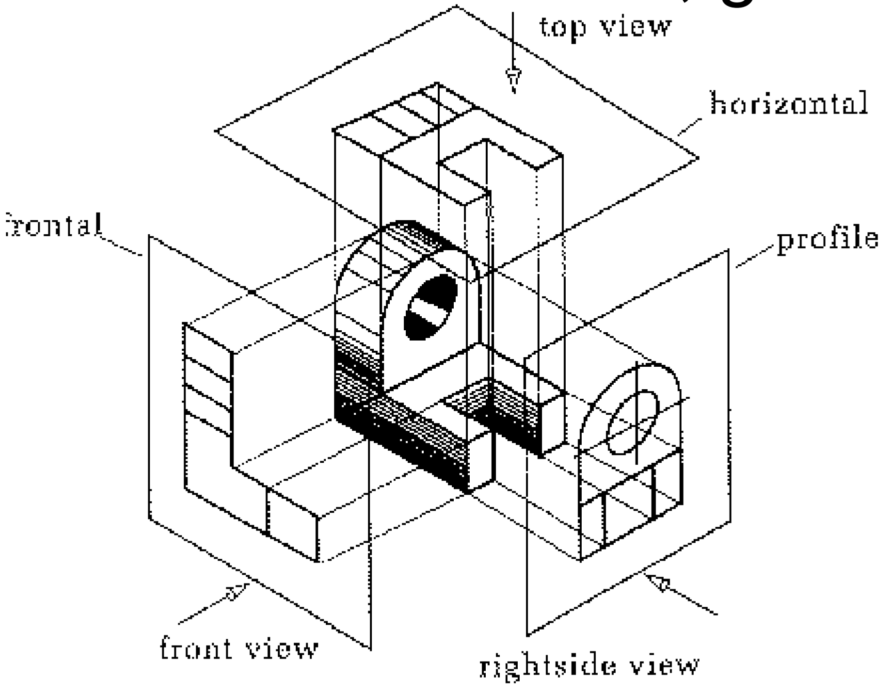


three-point perspective

Tuebingen demo: vanishingpoints

Orthographic Projections

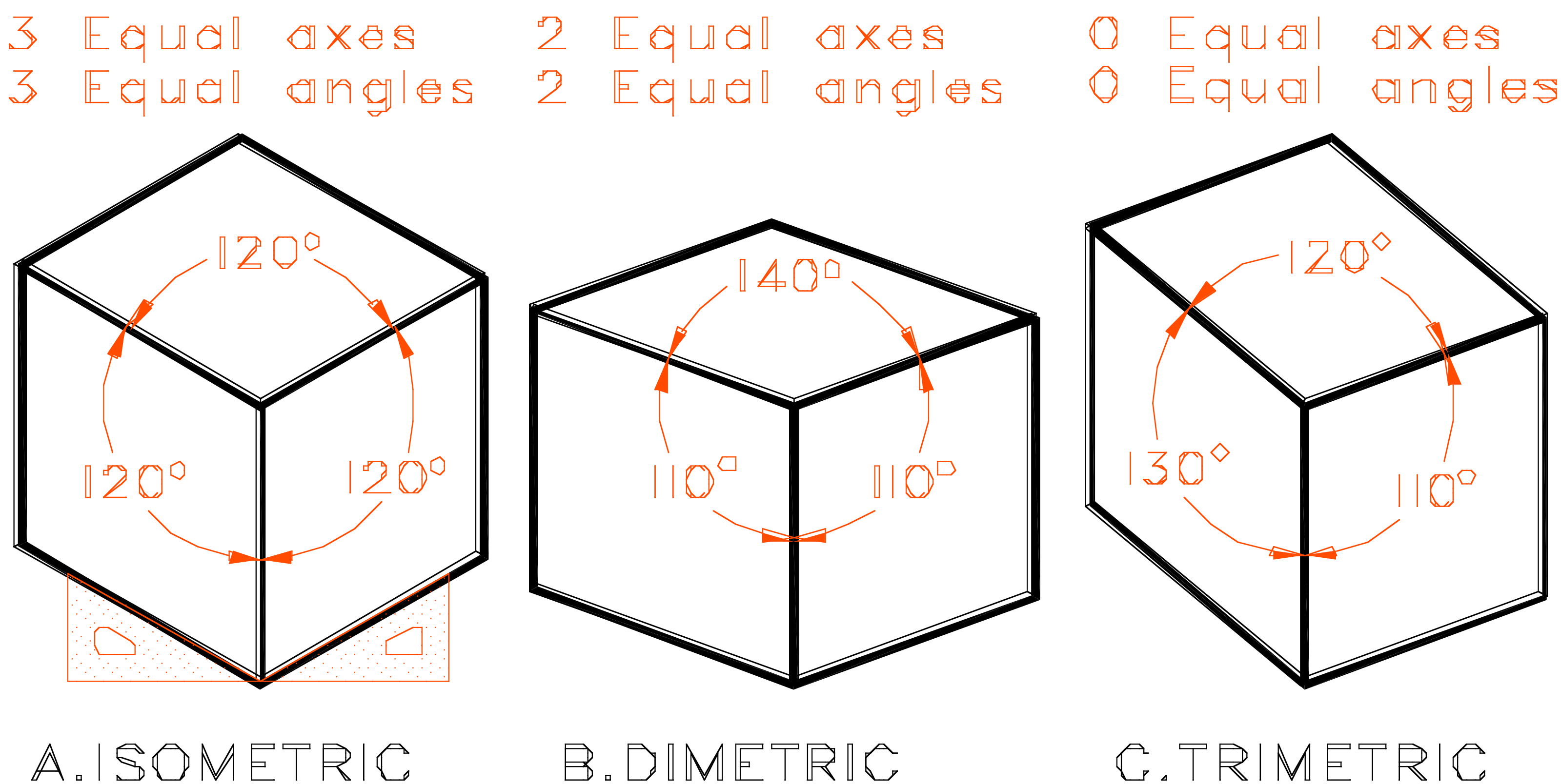
- projectors parallel, perpendicular to image plane
- image plane normal parallel to one of principal axes
- select view: top, front, side
- every view has true dimensions, good for measuring



http://www.cs.fit.edu/~wds/classes/cse5255/thesis/images/proj/orthoProj.gif

Axonometric Projections

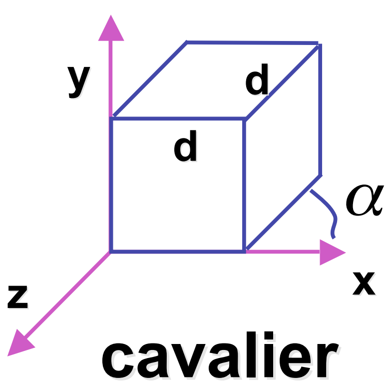
- projectors parallel, perpendicular to image plane
- image plane normal not parallel to axes
- select axis lengths
- can see many sides at once

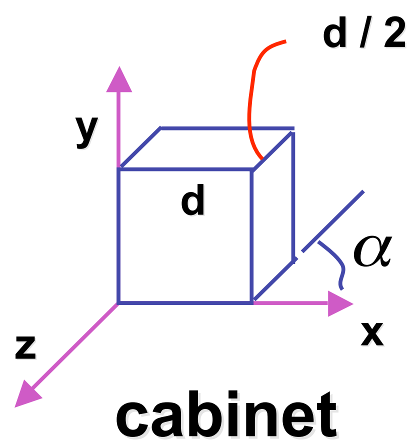


http://ceprofs.tamu.edu/tkramer/ENGR%20111/5.1/20

Oblique Projections

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





Tuebingen demo: oblique projections