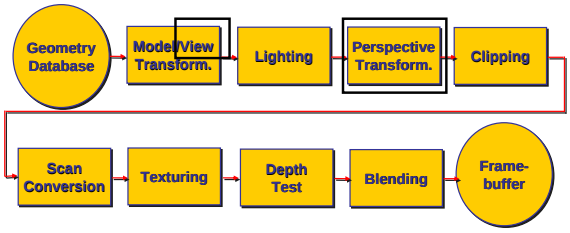
University of British Columbia

Chapter 6

Viewing/Perspective Transformations

transformations

Rendering Pipeline



- Specify view point (change of coordinate system)
- Project from 3D to 2D (introduce perspective)

University of British Columbia

Rendering Pipeline

Scene graph Object geometry

Modelling Transforms

Viewing Transform

Projection Transform

University of British Columbia

Rendering Pipeline

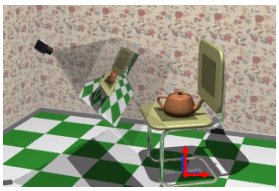
Scene graph Object geometry

Modelling Transforms

Viewing Transform

Projection Transform

- result
  - all vertices of scene in shared 3D world coordinate system



University of British Columbia

Rendering Pipeline

Scene graph Object geometry

Modelling Transforms

Viewing Transform

Projection Transform

- result
  - scene vertices in 3D view (camera) coordinate system

University of British Columbia

Rendering Pipeline

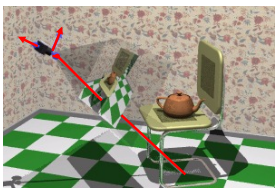
Scene graph Object geometry

Modelling Transforms

Viewing Transform

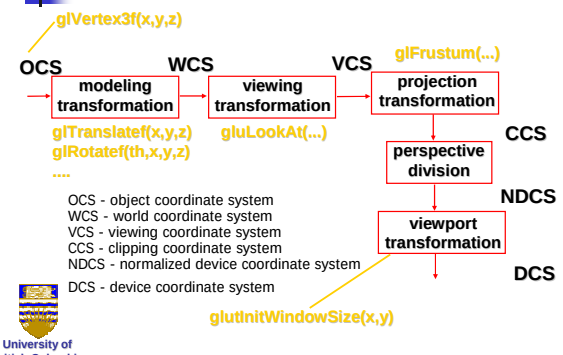
Projection Transform

- result
  - 2D screen coordinates of clipped vertices



University of British Columbia

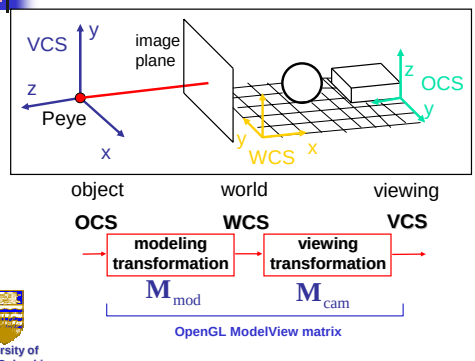
Projective Rendering Pipeline



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

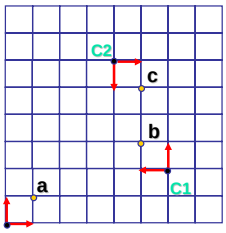
University of British Columbia

Viewing Transformation



University of British Columbia

World vs. Camera Coordinates



$a = (1,1)_W$

$b = (1,1)_{C1} = (5,3)_W$

$c = (1,1)_{C2} = (1,3)_{C1} = (5,5)_W$

University of British Columbia

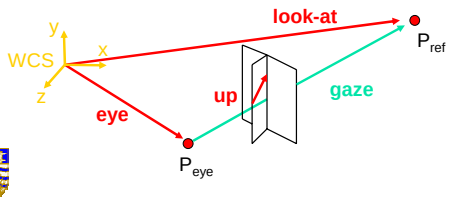
OpenGL Basic Viewing

- starting spot - OpenGL
  - camera at world origin
    - probably inside an object
  - y axis is up
  - looking down negative z axis
    - why? RHS with x horizontal, y vertical, z out of screen
- To position – coordinate frame change
- Intuitive description
  - eye point, look-at point, up vector

University of British Columbia

Camera Description

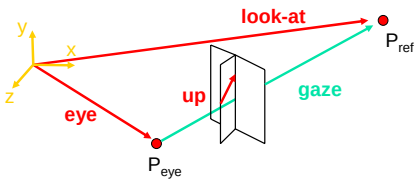
- arbitrary viewing position
  - eye point, look-at direction, up vector



University of British Columbia

From World to View Coordinates: W2V

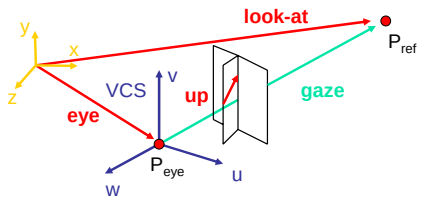
- Setup *uvw* basis with origin at eye
- translate **eye** to origin
- rotate *uvw* to align with *xyz*



University of British Columbia

Deriving W2V Transformation

- Gaze is (look-at – eye)
- Use opposite of gaze $\mathbf{w} = -\hat{\mathbf{g}} = -\frac{\mathbf{g}}{\|\mathbf{g}\|}$

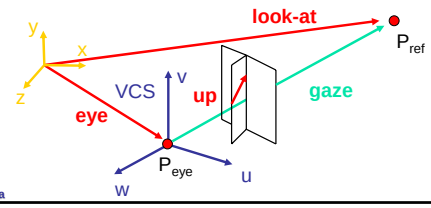


University of British Columbia

Deriving W2V Transformation

- U - perpendicular to W and Up vector
- V – perpendicular to W, V

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|} \qquad \mathbf{v} = \mathbf{w} \times \mathbf{u}$$

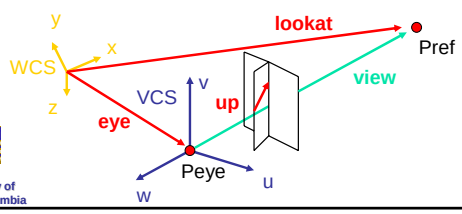


University of British Columbia

Deriving W2V Transformation

- translate **eye** to origin

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



University of British Columbia

Deriving W2V Transformation

- rotate from WCS *xyz* into *uvw* coordinate system with matrix that has rows u, v, w

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}$$

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

$$\mathbf{w} = -\hat{\mathbf{g}} = -\frac{\mathbf{g}}{\|\mathbf{g}\|}$$

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

- reminder:
  - rotate from uvw to xyz coord sys with matrix M that has columns u,v,w
  - rotate from xyz coord sys to uvw coord sys with matrix M^T that has rows u,v,w

University of British Columbia

Deriving W2V Transformation

- M=RT

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

$$\mathbf{M}_{world->view} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & -\mathbf{u}\cdot\mathbf{e} \\ v_x & v_y & v_z & -\mathbf{v}\cdot\mathbf{e} \\ w_x & w_y & w_z & -\mathbf{w}\cdot\mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

University of British Columbia

OpenGL Viewing Transformation

```
gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)
```

- postmultiplies current matrix, so to be safe initialize first

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(ex,ey,ez,lx,ly,lz,ux,uy,uz)
// now ok to do model transformations
```

- demo: Nate Robins tutorial *projection*

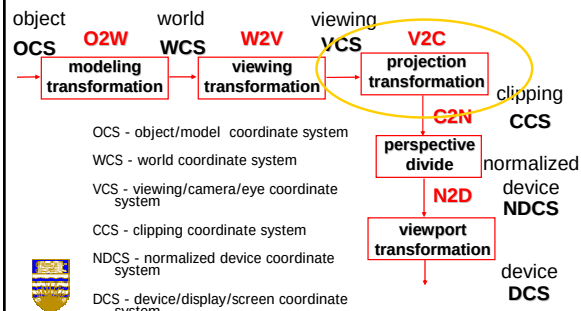
University of British Columbia

Moving the Camera or the World?

- two equivalent operations
  - move camera one way vs. move world other way
- example
  - initial OpenGL camera: at origin, looking along -z axis
  - create a unit square parallel to camera at z = -10
  - translate in z by 3 possible in two ways
    - camera moves to z = -3
      - Note OpenGL models viewing in left-hand coordinates
    - camera stays put, but square moves to -7
  - resulting image same either way
    - possible difference: are lights specified in world or view coordinates?

University of British Columbia

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

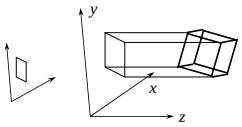
CCS - clipping coordinate system

NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

University of British Columbia

Projection Transformations



- Question: How to draw 3D object on 2D screen?
- Answer:
  - Project transformed object along Z axis onto XY plane - and from there to screen (space)
  - Canonical projection:

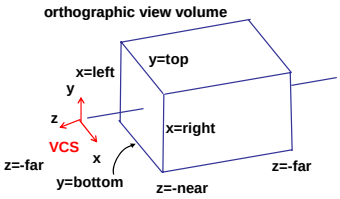
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- "ignore" *z* coordinate – use *x* and *y* coordinates for screen coordinates

University of British Columbia

View Volumes

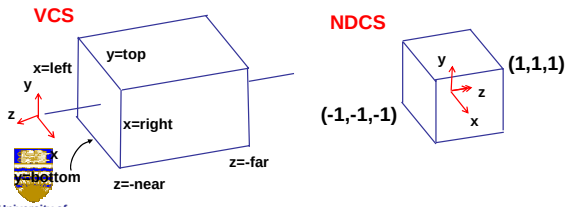
- specifies field-of-view, used for clipping
- restricts domain of ***z*** stored for visibility test



University of British Columbia

Understanding Z

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



University of British Columbia

Understanding Z

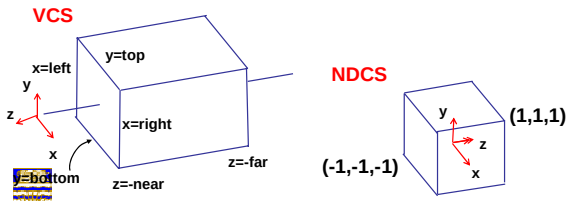
- why near and far plane?
  - near plane:
    - avoid singularity for perspective (division by zero, or very small numbers)
  - far plane:
    - store depth in fixed-point representation (integer), thus have to have fixed range of values (0...1)
    - avoid/reduce numerical precision artifacts for distant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

University of British Columbia

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad top \to 1$$
$$bottom \to -1$$



University of British Columbia

Orthographic Derivation

- scale, translate, reflect for new coord sys

$$y' = a \cdot y + b \qquad top \to 1$$
$$bottom \to -1$$

- Solve two eq.

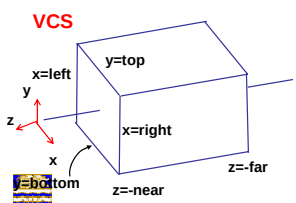
$$1 = a \cdot top + b \qquad a = \frac{2}{top - bottom}$$
$$-1 = a \cdot bottom + b \qquad b = \frac{-top - bottom}{top - bottom}$$

University of British Columbia

Orthographic Derivation

- scale, translate, reflect for new coord sys

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



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

same idea for right/left, far/near

University of British Columbia

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

University of British Columbia

Orthographic Derivation

- scale translate, reflect for new coord sys

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

University of British Columbia

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

University of British Columbia

Orthographic Derivation

- scale, translate, reflect for new coord sys

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

University of British Columbia

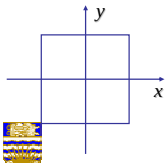
Orthographic OpenGL

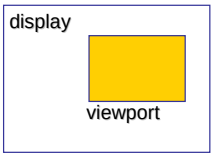
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left,right,bot,top,near,far);
```

University of British Columbia

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

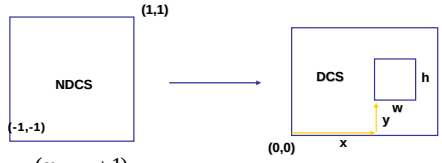




University of British Columbia

NDC to Viewport Transformation

- 2D scaling and translation



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

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

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

OpenGL

```
glViewport(x,y,w,h);
```

default:

```
glViewport(0,0,w_wnd,h_wnd);
```

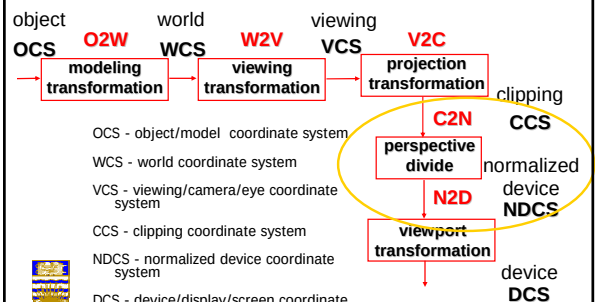
University of British Columbia

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

University of British Columbia

Projective Rendering Pipeline



OCS - object/model coordinate system

WCS - world coordinate system

VCS - viewing/camera/eye coordinate system

CCS - clipping coordinate system

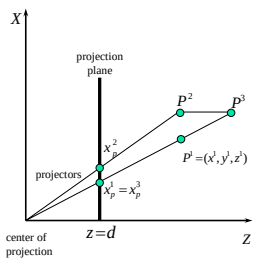
NDCS - normalized device coordinate system

DCS - device/display/screen coordinate system

University of British Columbia

Perspective Projection

- Viewing is from *point at finite distance*
- Without loss of generality:
  - Viewpoint at origin
  - Viewing plane is $z=d$
- Given P=(x,y,z) triangle similarity gives:



$$\frac{x}{z}=\frac{x_p}{d} \text{ and } \frac{y}{z}=\frac{y_p}{d} \Rightarrow x_p=\frac{x}{z/d} \text{ and } y_p=\frac{y}{z/d}$$

University of British Columbia

Perspective Projection (cont'd)

- In matrix notation with homogeneous coordinates:

$$P(x,y,z,1)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1/d&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}=\begin{bmatrix}x\\y\\z\\z/d\end{bmatrix}.$$

- In Euclidean coordinates:

$$\left(\frac{x}{z/d},\frac{y}{z/d},\frac{z}{z/d}\right)=\left(\frac{x}{z/d},\frac{y}{z/d},d\right)=(x_p,y_p,d).$$

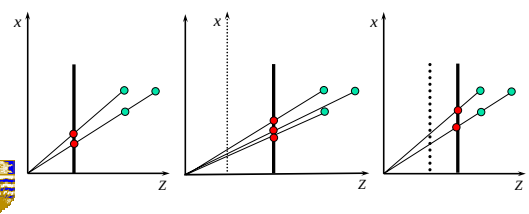
- P singular: $det(P)=0$

University of British Columbia

Perspective Projection (cont'd)

- What is (if any) is the difference between:
  - Moving projection plane
  - Moving viewpoint (center of projection)?

camtrans



University of British Columbia

How to make non-degenerate?

$$P(x,y,z,1)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&1/d&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}=\begin{bmatrix}x\\y\\z\\z/d\end{bmatrix}.$$

$$\left(\frac{x}{z/d},\frac{y}{z/d},\frac{z}{z/d}\right)=\left(\frac{x}{z/d},\frac{y}{z/d},d\right)=(x_p,y_p,d).$$

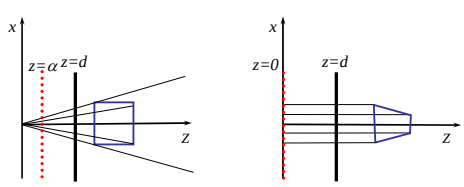
- z' monotonically increasing function of z

$$P(x,y,z,1)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&a&b\\0&0&1/d&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}=\begin{bmatrix}x\\y\\az+b\\z/d\end{bmatrix}. \qquad z'=da+\frac{b}{z}$$

University of British Columbia

Perspective Warp –Geometric Meaning

- Warp viewing frustum (world)
- Then use parallel projection



University of British Columbia

Perspective Warp

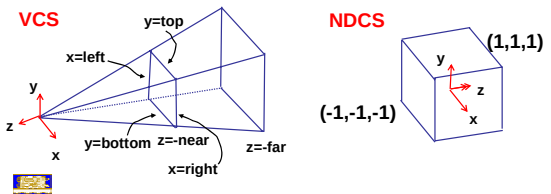
- Matrix formulation

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

- Preserves relative depth (third coordinate)
- What does $\alpha=0$ mean?

University of British Columbia

OpenGL Perspective Derivation



University of British Columbia

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

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

University of British Columbia

Perspective Derivation

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

$$x' = Ex + Az$$
$$y' = Fy + Bz$$
$$z' = Cz + D$$
$$w' = -z$$

$$x = left \rightarrow x'/w' = 1$$
$$x = right \rightarrow x'/w' = -1$$
$$y = top \rightarrow y'/w' = 1$$
$$y = bottom \rightarrow y'/w' = -1$$
$$z = -near \rightarrow z'/w' = 1$$
$$z = -far \rightarrow z'/w' = -1$$

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

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

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

University of British Columbia

Perspective Derivation

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

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

University of British Columbia

Perspective Example

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

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

University of British Columbia

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

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

University of British Columbia

Perspective OpenGL

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

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

University of British Columbia

Another Transformations Quiz

- What does each transformation preserve?

	lines	parallel lines	distance	angles	normals	convexity	conics
scaling							
rotation							
translation							
shear							
perspective							

University of British Columbia