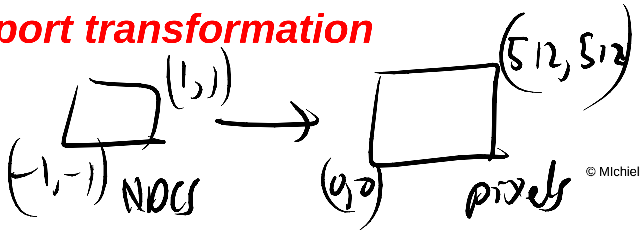
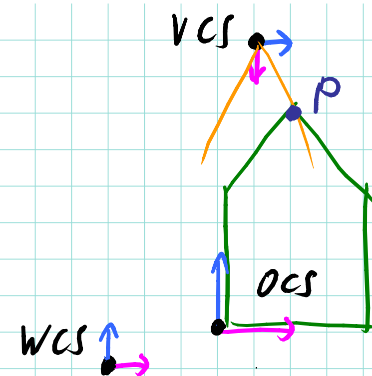
University of British Columbia

# Viewing and Projection Transformations

- viewing transformation — camera model (position and orient)
- intro to projection transformations
- view volumes
- viewport transformation






$P_{obj} = (1,3)$

$$\begin{bmatrix}5\\7\\1\end{bmatrix}_{wcs} = \begin{bmatrix}2&0&3\\0&2&1\\0&0&1\end{bmatrix}\begin{bmatrix}1\\3\\1\end{bmatrix}_{ocs}$$

$$\begin{bmatrix}2\\1\\1\end{bmatrix}_{vcs} = \underbrace{\begin{bmatrix}0&-1&9\\1&0&-4\\0&0&1\end{bmatrix}}_{M_{view}}\begin{bmatrix}5\\7\\1\end{bmatrix}_{wcs}$$

$$P_{ocs} \rightarrow \boxed{M_{mod}} \xrightarrow{P_{wcs}} \boxed{M_{view}} \xrightarrow{} P_{vcs}$$

$$P_{vcs} = M_{view} M_{mod} P_{ocs}$$

University of British Columbia

# Projective Rendering Pipeline

glVertex(p)

camera coords

$P_{ocs}$ → OCS → **modeling transformation** (4x4) → WCS → **viewing transformation** (4x4) → VCS → **projection transformation** (4x4) → ~~CCS~~ → **/ h** → NDCS → **viewport transformation** → DCS

projection

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

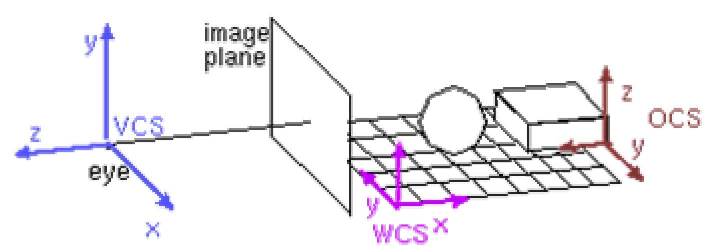
Device coords
pixel coords



University of British Columbia

# Viewing Transformation

***Positioning the camera***

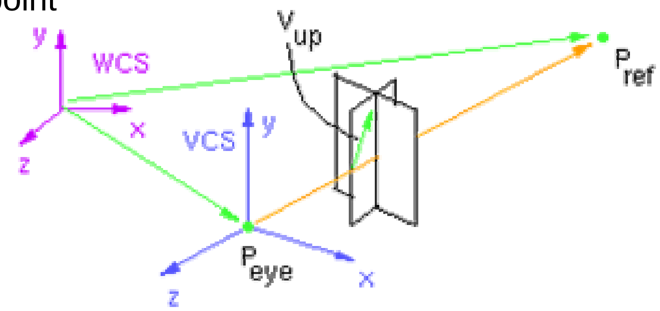


# Viewing Transformation

University of British Columbia

## Defining the camera position

- eye point
- reference point
- up vector






# Viewing Transformation

University of British Columbia

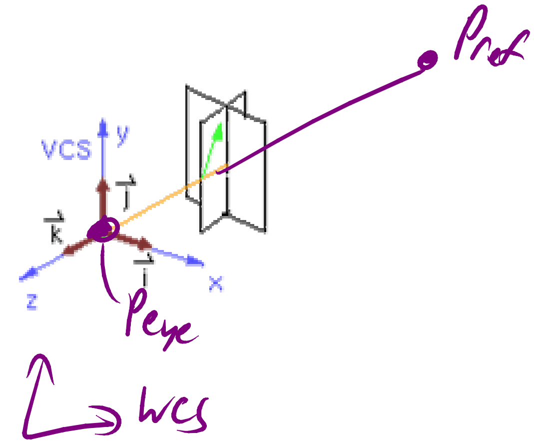
## Computing Mcam

$$O_{vcs} = P_{eye}$$

$$\vec{k}_{vcs} = \frac{P_{eye} - P_{ref}}{|P_{eye} - P_{ref}|}$$

$$\vec{i}_{vcs} = \frac{\vec{V}_{up} \times \vec{k}_{vcs}}{|\vec{V}_{up} \times \vec{k}_{vcs}|}$$

$$\vec{j}_{vcs} = \vec{k}_{vcs} \times \vec{i}_{vcs}$$






# Viewing Transformation

University of British Columbia

## Computing Mcam

$$M_{cam} = \begin{bmatrix} 1 & & & P_{eye,x} \\ & 1 & & P_{eye,y} \\ & & 1 & P_{eye,z} \\ & & & 1 \end{bmatrix} \begin{array}{c} \begin{array}{cccc} i_{vcs} & j_{vcs} & k_{vcs} & \end{array} \\ \begin{bmatrix} i_x & j_x & k_x & \\ i_y & j_y & k_y & \\ i_z & j_z & k_z & \\ & & & 1 \end{bmatrix} \end{array} \qquad (A\,B)^{-1}$$

Camera → World

$$M_{view} \quad M_{cam}^{-1} = \begin{bmatrix} i_x & i_y & i_z & \\ j_x & j_y & j_z & \\ k_x & k_y & k_z & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & -P_{eye,x} \\ & 1 & & -P_{eye,y} \\ & & 1 & -P_{eye,z} \\ & & & 1 \end{bmatrix} \qquad B^{-1} A^{-1}$$

World → Camera



# Viewing Transformation

University of British Columbia

OCS → **modeling transformation** → WCS → **viewing transformation** → VCS

$$\underbrace{M_{mod} \qquad M_{cam}^{-1}}_{\text{OpenGL ModelView matrix}}$$

$$P_{WCS} = M_{mod} \cdot P_{OCS}$$

$$P_{WCS} = M_{cam} \cdot P_{VCS}$$

$$P_{VCS} = \boxed{M_{cam}^{-1} \cdot M_{mod}} \cdot P_{OCS}$$

# Viewing Transformation

University of British Columbia

## *OpenGL*

- `gluLookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz)`

  but this postmultiplies the current matrix; therefore usually use as follows:

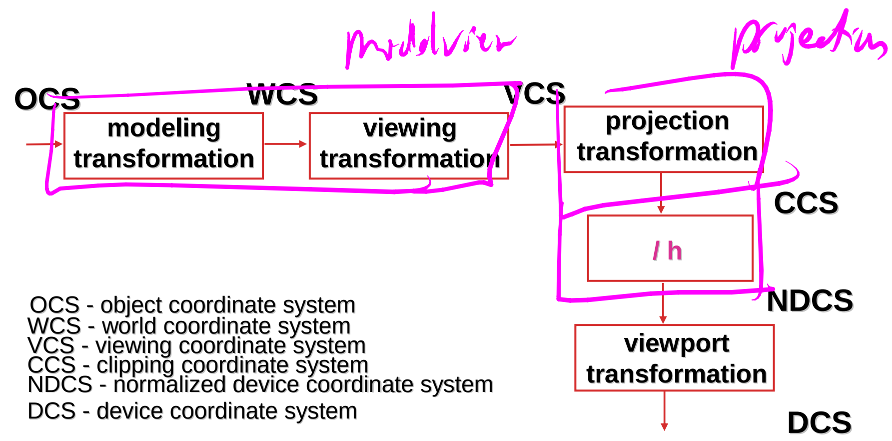
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz)

// now ok to setup modeling transformations
```



# Projective Rendering Pipeline

University of British Columbia



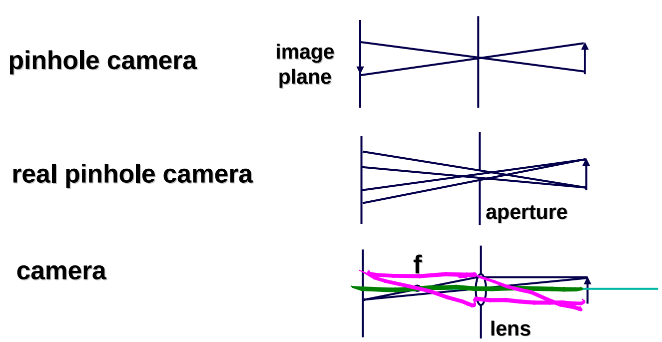

OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system



# Projection

University of British Columbia

## *Pinhole camera*






# Projection

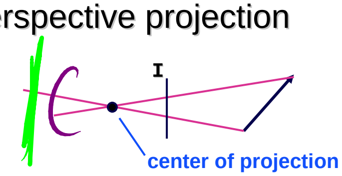
University of British Columbia

- definition 3D→2D

  **mapping** $f:\Re^n \to \Re^m,\ m<n$

- parallel projection

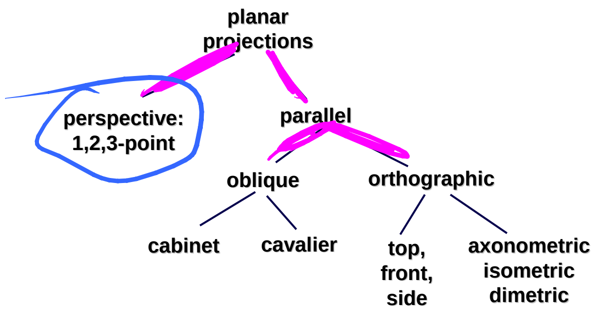

- perspective projection



# Projections

University of British Columbia

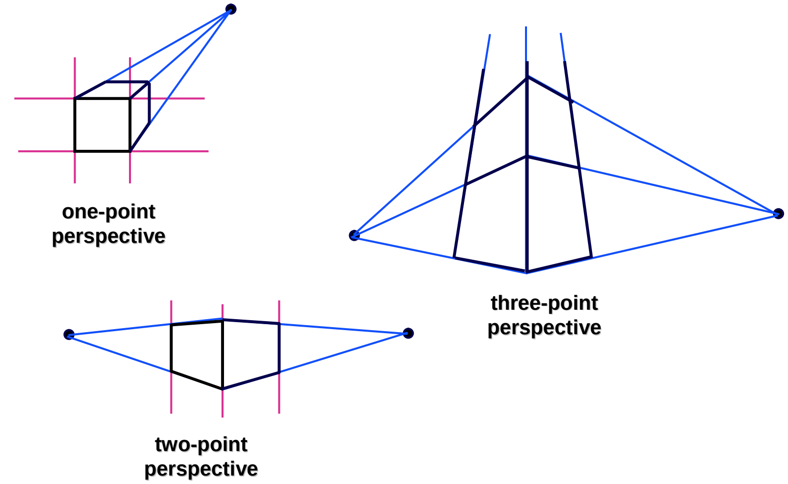
## *Taxonomy*





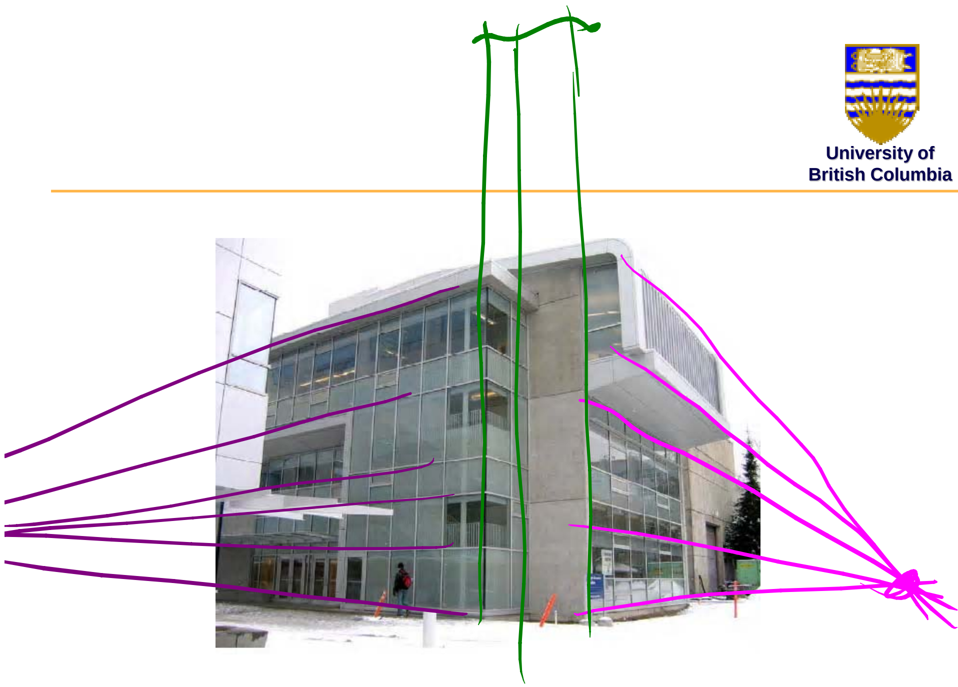

# Projections

University of British Columbia




© Michiel van de Panne

University of British Columbia



© Michiel van de Panne

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & -\frac{c}{2} & \\ & 1 & -\frac{s}{2} & \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

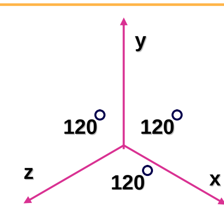
# Projections

University of British Columbia

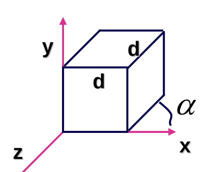
$$x' = x - \frac{z}{2}\cos\alpha$$

$$y' = y - \frac{z}{2}\sin\alpha$$

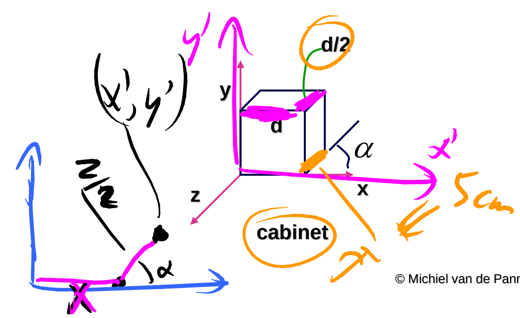
$$z' = 0$$



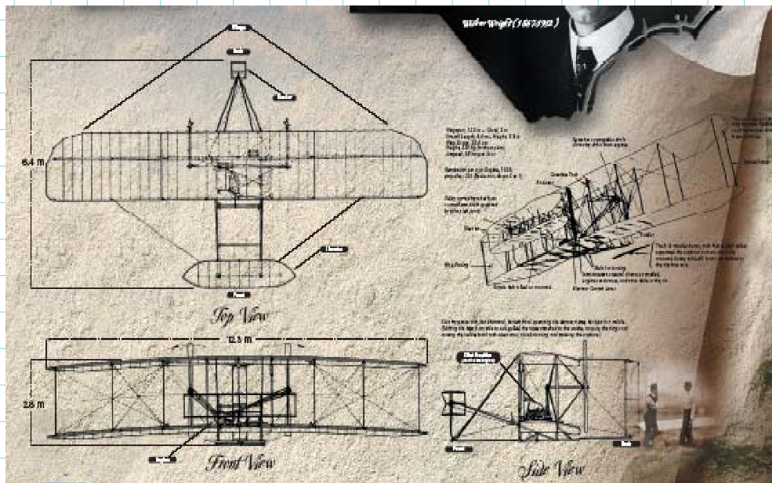

isometric

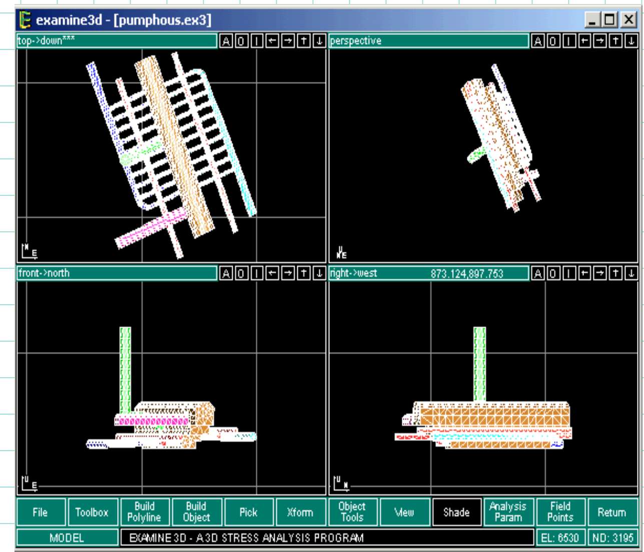



cavalier

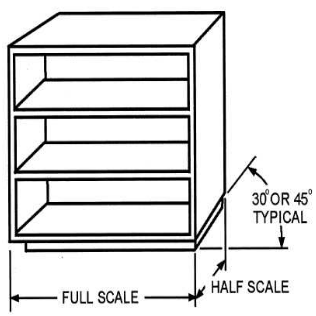


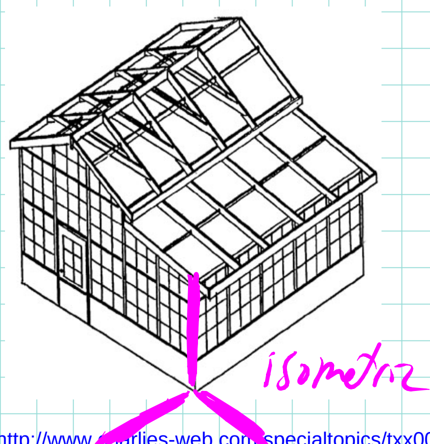

© Michiel van de Panne



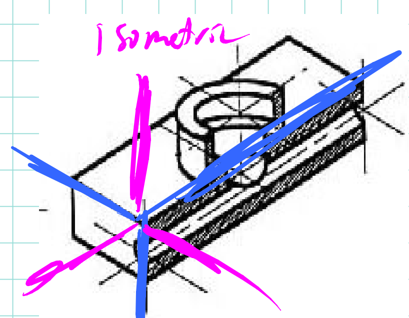




http://www.tpub.com/content/draftsman/14276/css/14276_308.htm




http://www.charlies-web.com/specialtopics/txx003.html

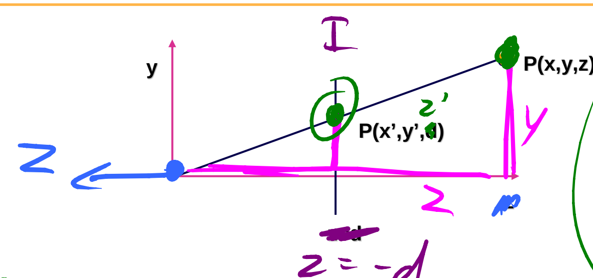



http://pergatory.mit.edu/2.007/Resources/drawings/

- similar triangles,
- /z

# Basic Projection

University of British Columbia




$P' = f(P)$

$x' = -d\frac{x}{z}$

$y' = -d\frac{y}{z}$

$z' = -d$

$$\begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ 0 & 0 & 1 & 0 \\ & & -\frac{1}{d} & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$\div h$

$x' = x/h$

$y' = y/h$

$z' = z/h$

$$\boxed{\frac{y}{z} = \frac{y'}{z'}}$$

$\Rightarrow y' = z'\frac{y}{z}$

let $h = -z/d$

© Michiel van de Panne

# Homogeneous Coordinates

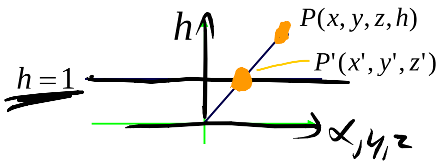


**homogeneous** **cartesian**

$$(x, y, z, h) \xrightarrow{/h} \left(\frac{x}{h}, \frac{y}{h}, \frac{z}{h}\right)$$

$$(5, 5, 5, 10) \longleftrightarrow (0.5, 0.5, 0.5, 1)$$

- redundant representation
- h=0: point at infinity (direction) → good for representing vectors or normals
- geometric interpretation






# Basic Projection



## *Using h and 4x4 matrices*

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

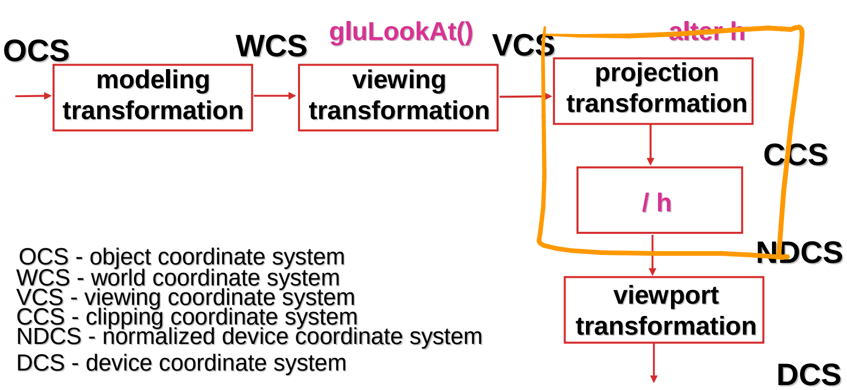
$$\xrightarrow{/\hat{h}} \begin{bmatrix} xz/d \\ yz/d \\ d \end{bmatrix}$$

assumes $d < 0$



# Projective Rendering Pipeline





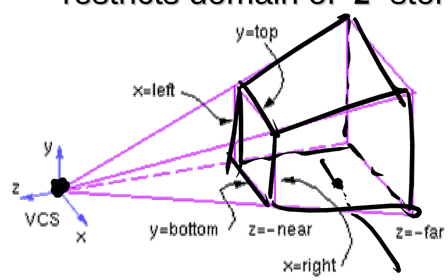

OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system



# View Frustum
# View Volumes



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

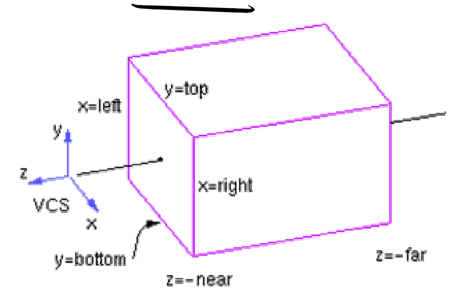



**perspective view volume**

perspective projections

orthographic proj.

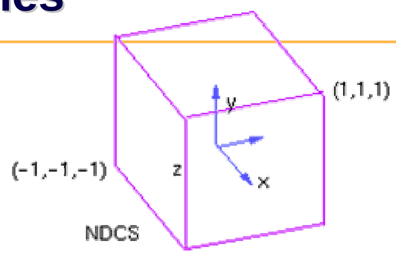
**orthographic view volume**

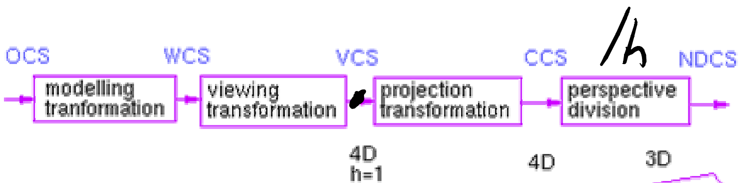


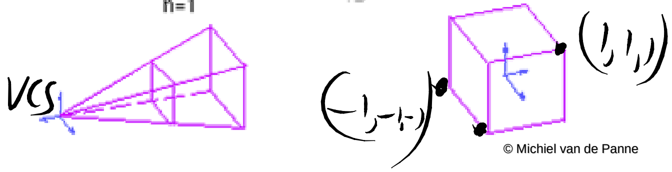
# View Volumes

University of British Columbia

**OpenGL**
**canonical view volume**



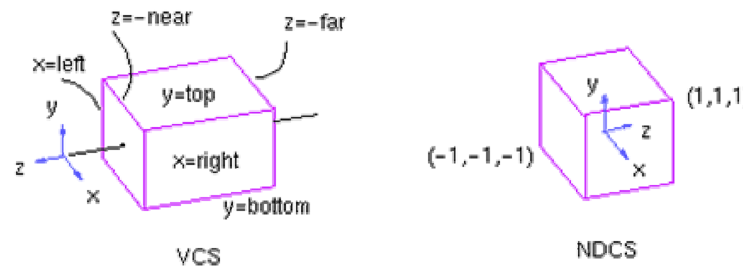






# View Volumes

University of British Columbia

## *Derivation – orthographic projections*






# View Volumes

University of British Columbia

## *Derivation – orthographic projections*

$$y' = a \cdot y + b \qquad y = top \;\rightarrow\; y' = 1$$

$$y = bot \;\rightarrow\; y' = -1$$

**solving for a and b gives:**

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



# View Volumes

University of British Columbia

## *Derivation – orthographic projections*

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

**OpenGL**

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

# View Volumes

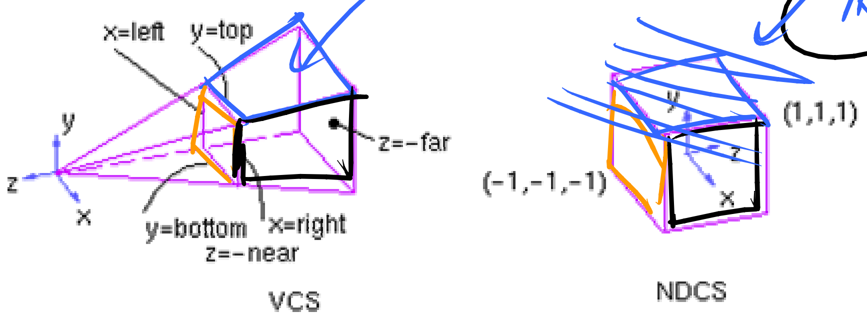
University of British Columbia

## *Derivation – Perspective case*

$y = top$
$z = -near$

$y = \frac{top}{-near} z$

$y_{NDCS} = 1$






# View Volumes

University of British Columbia

## *Derivation – Perspective case*

**earlier:**

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

$$\begin{bmatrix} d & & & \\ & d & & \\ & & d & \\ & & -1 & \end{bmatrix}$$

**with additional ability to scale, etc.:**

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

$y_{NDCS} = \frac{y'}{h'} = \frac{Fy + Bz}{-z}$

$y_{NDCS} = F\frac{y}{z} - B$

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



# View Volumes

University of British Columbia

## *Derivation – Perspective case*

**top plane:**

$$y = z\frac{top}{(-near)} \longrightarrow \frac{y'}{h'} = 1 \qquad \frac{Fy + Bz}{-z} = 1$$

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

**repeat for bot plane to get another eqn, then solve for F and B**

**similar process for solving for the other unknowns, using the left/right and near/far planes**



# View Volumes

University of British Columbia

$n = near$ $\quad t = top$
$r = right$ $\quad f = far$
$l = left$ $\quad b = bottom$

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

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

# Perspective Transform

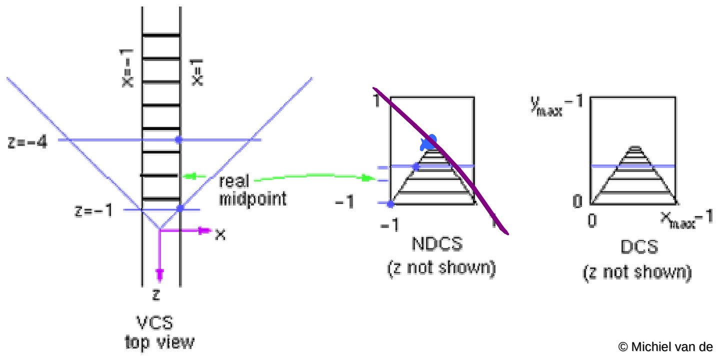
University of British Columbia

## *Example*

Camera is 1 unit above the tracks

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 Transform

University of British Columbia

## *Example*

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

x=1
y=-1

right hand rail

/ h

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

$y_{NDCS}$ $x_{NDCS}$ = -1



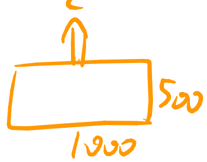
# Perspective Transform

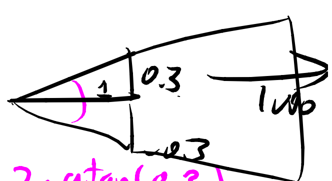
University of British Columbia

## *OpenGL*

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
        -0.3  0.3  -0.3 0.3  1   1000
glFrustum(left,right,bot,top,near,far);
  or
glPerspective(fovy,aspect,near,far);
           30°    2    0.1  1000
```

perspective projections



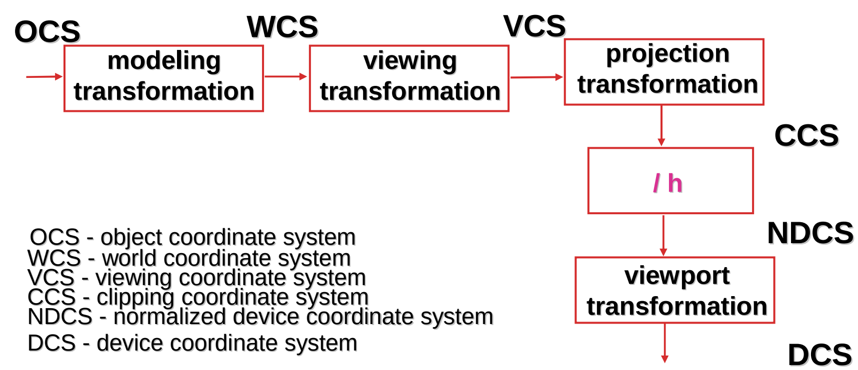




fov = 2 · atan(0.3)



# Projective Rendering Pipeline

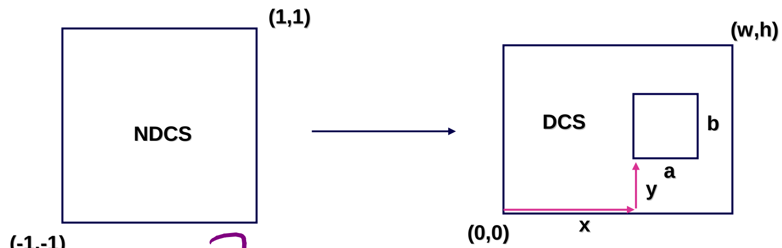
University of British Columbia




- 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

# Viewport Transformation

University of
British Columbia




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

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

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

$z_{DCS} \in [0,1]$

near plane

far plane

OpenGL

```
glViewport(x,y,a,b);
```

fancy

default:

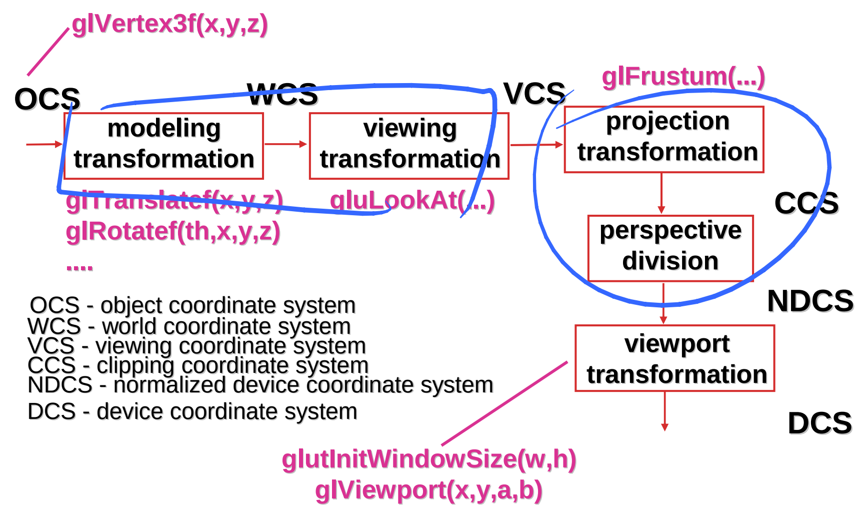
```
glViewport(0,0,w,h);
```

plain



# Projective Rendering Pipeline

University of
British Columbia




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



# Coming Up…

University of
British Columbia

- clipping and culling
- visibility
- scan conversion