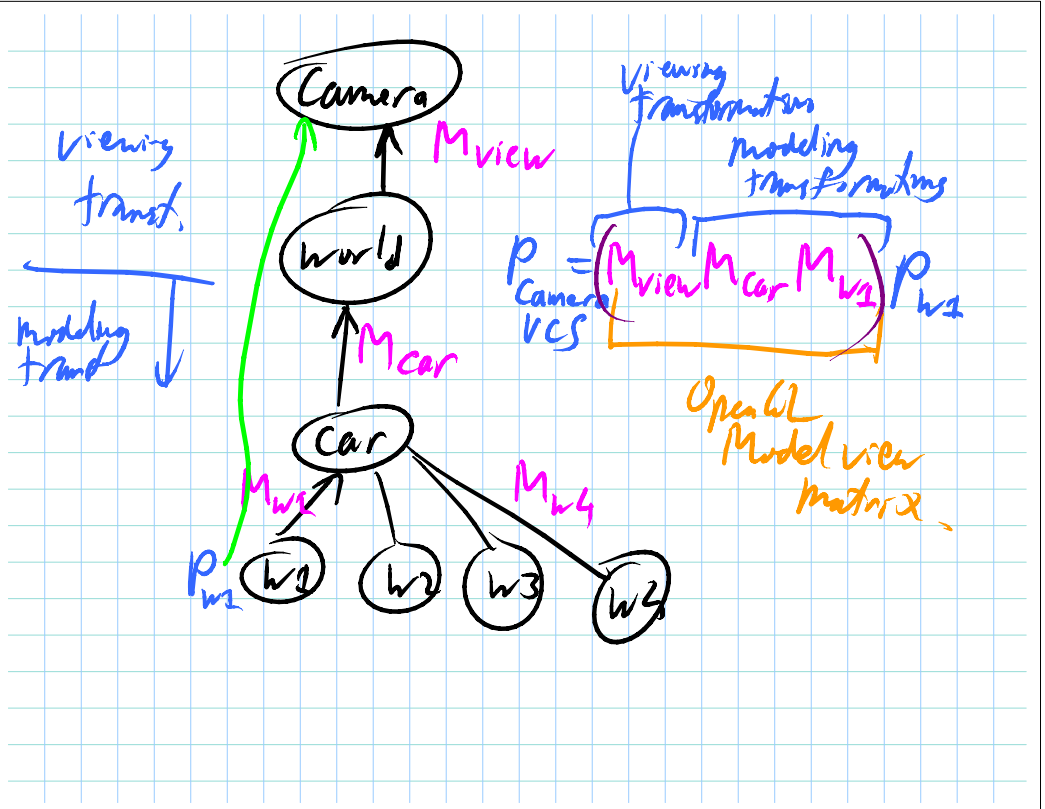
University of British Columbia

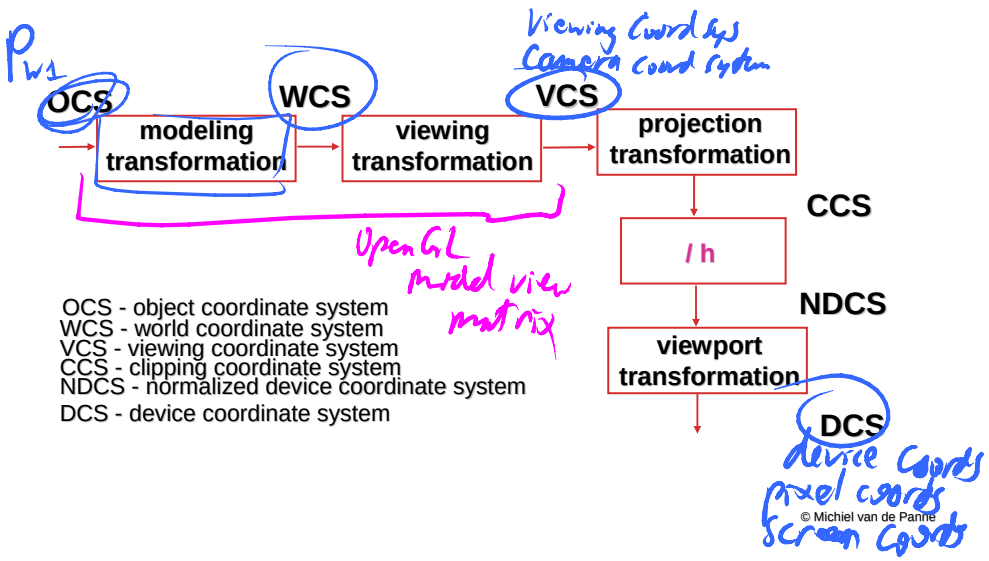
Viewing and Projection Transformations

- *viewing transformation*
- *intro to projection transformations*
- *view volumes*
- *viewport transformation*



University of British Columbia

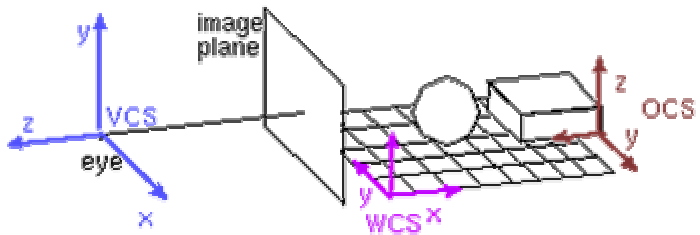
Projective Rendering Pipeline



University of British Columbia

Viewing Transformation

Positioning the camera



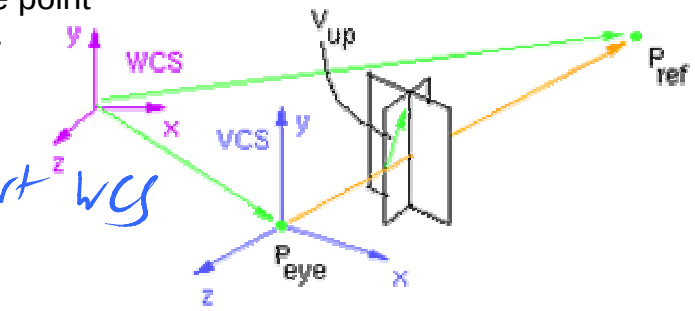
Viewing Transformation

University of British Columbia

Defining the camera position

- eye point
- reference point
- up vector

specified wrt WCS



Viewing Transformation

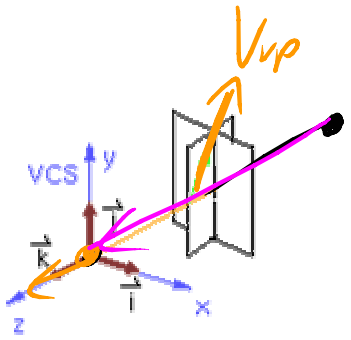
University of British Columbia

Computing Mcam

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

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

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



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

Viewing Transformation

University of British Columbia

Computing Mcam

$$M_{cam} = \begin{bmatrix} 1 & & & P_{eye,x} \\ & 1 & & P_{eye,y} \\ & & 1 & P_{eye,z} \\ & & & 1 \end{bmatrix} \begin{bmatrix} i_x & j_x & k_x & \\ i_y & j_y & k_y & \\ i_z & j_z & k_z & \\ & & & 1 \end{bmatrix}$$

i_{vcs} j_{vcs} k_{vcs}

$(AB)^{-1}$

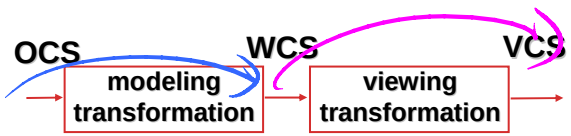
$B^{-1} A^{-1}$

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

for an orthonormal matrix, the inverse is the transpose

Viewing Transformation

University of British Columbia



$$M_{\text{mod}} \qquad M_{cam}^{-1}$$

OpenGL ModelView matrix

M_{view}

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

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

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

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:

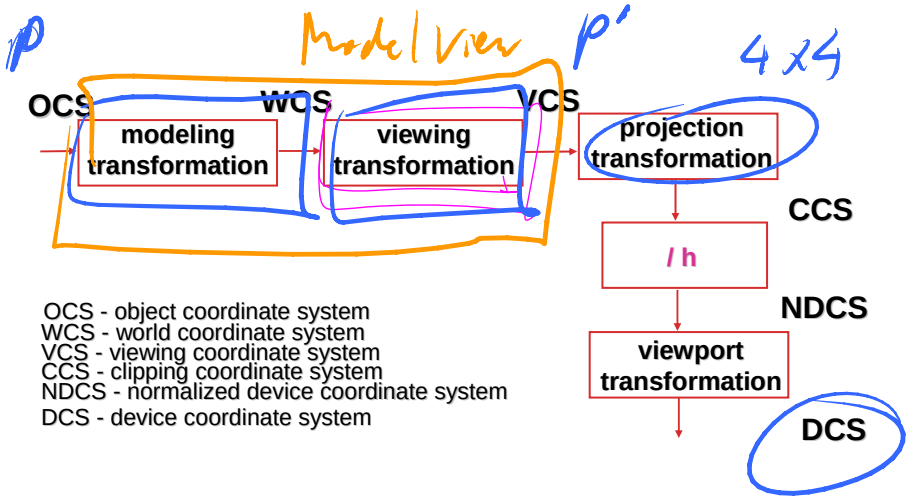
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz)

// now ok to setup modeling transformations
```

build M_{view} and postmultiply

$I * M_{view} = M_{view}$

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

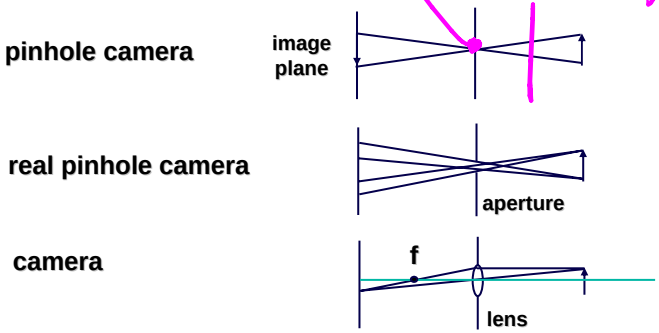
Projection

University of British Columbia

Pinhole camera

center of projection

I for graphics



Projection

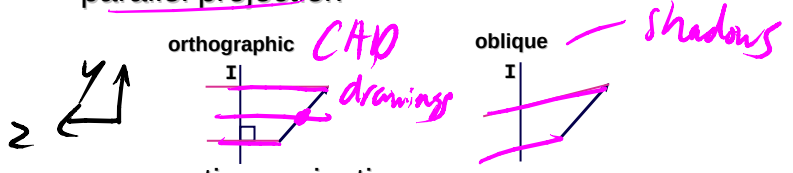
University of British Columbia

- definition

  mapping $f : \Re^n \rightarrow \Re^m, \quad m < n$

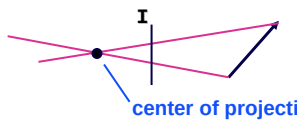
  3 2 3

- parallel projection



- perspective projection

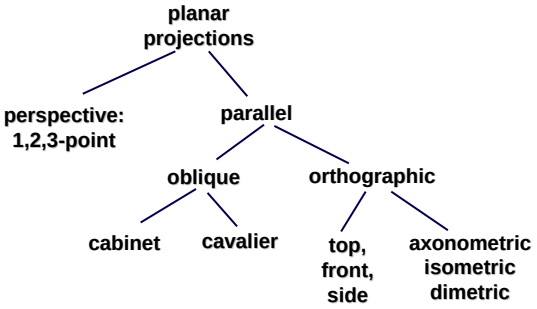
real cameras



Projections

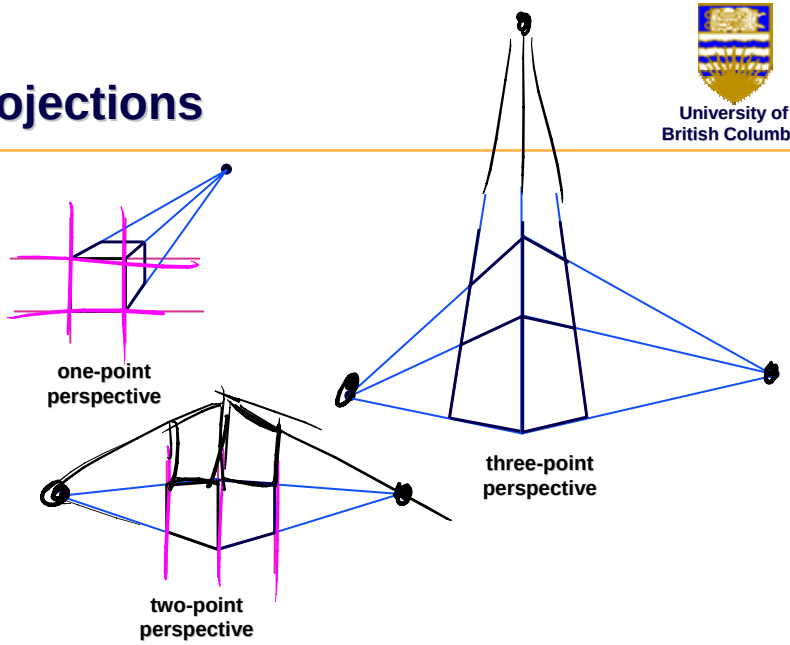
University of British Columbia

Taxonomy

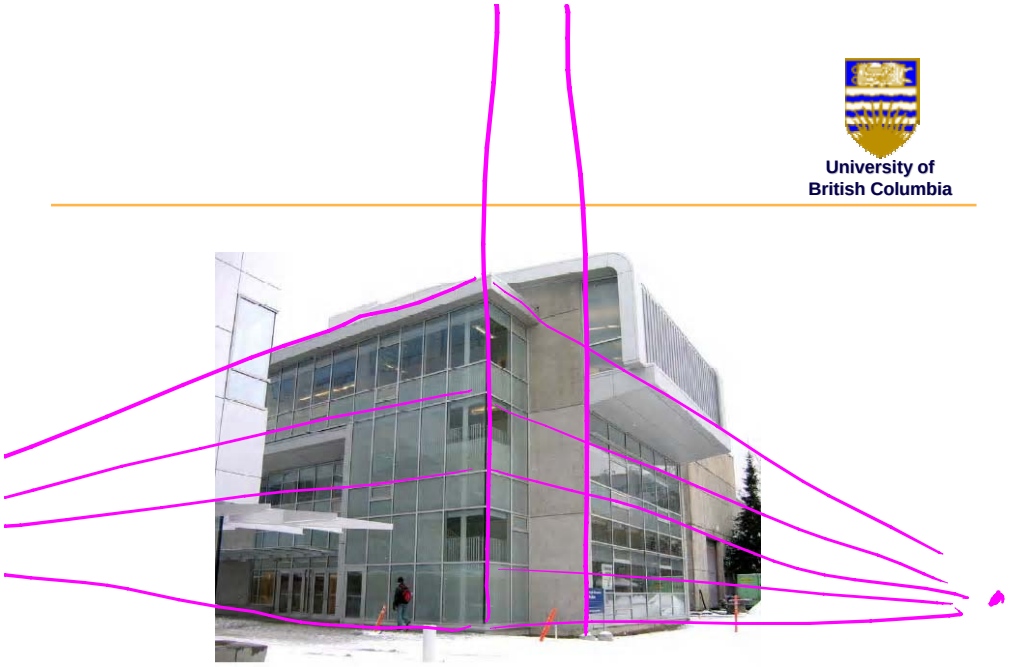


Projections

University of British Columbia

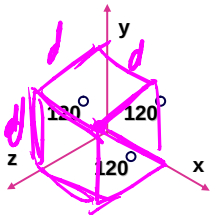


University of British Columbia

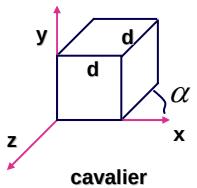


Projections

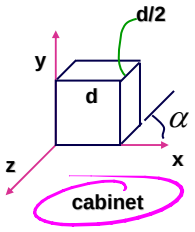
University of British Columbia



isometric

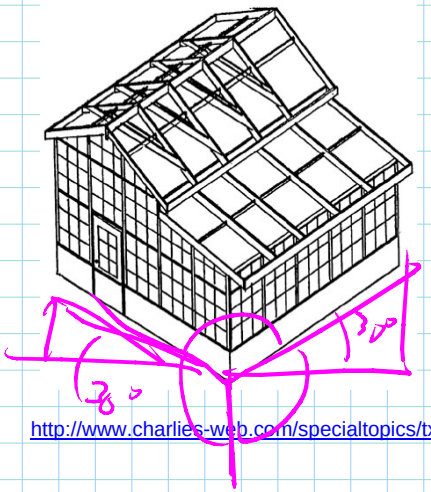


cavalier

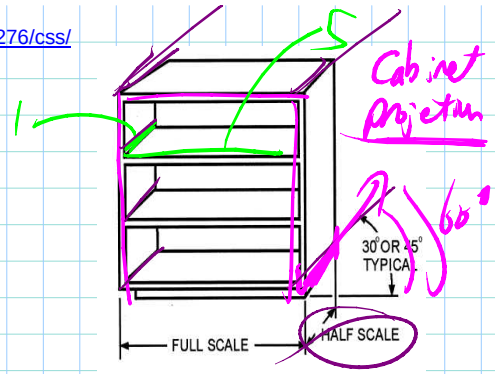


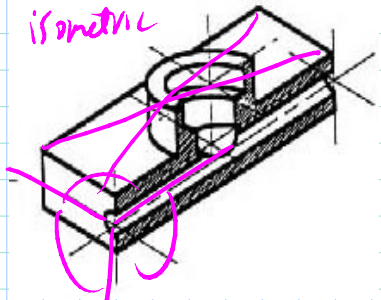
cabinet

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



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

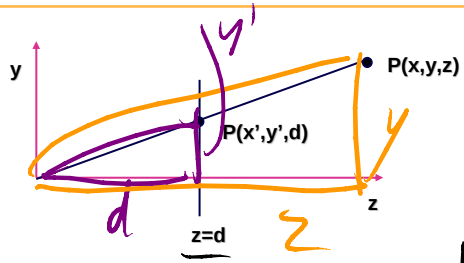




http://pergatory.mit.edu/2.007/Resources/drawings/

Basic Projection

University of British Columbia



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

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

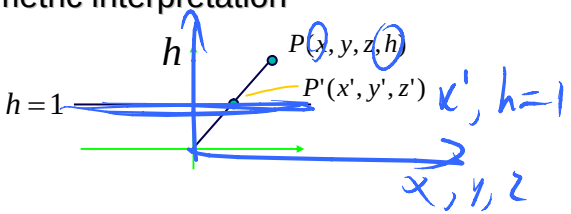
$$x' = \frac{d \cdot x}{z}$$

Homogeneous Coordinates

University of British Columbia

homogeneous		cartesian
(x, y, z, h)	$\xrightarrow{/\,h}$	$(\frac{x}{h}, \frac{y}{h}, \frac{z}{h})$

- redundant representation
- h=0: point at infinity (direction)
- geometric interpretation



Basic Projection

University of British Columbia

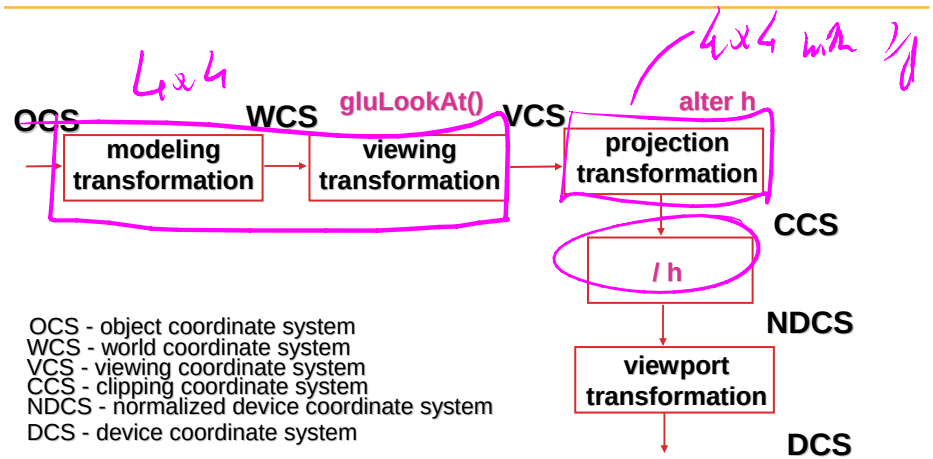
Using h and 4x4 matrices

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

$$\xrightarrow{/\,h} \begin{bmatrix} xd/z \\ yd/z \\ d \end{bmatrix}$$

Projective Rendering Pipeline

University of British Columbia

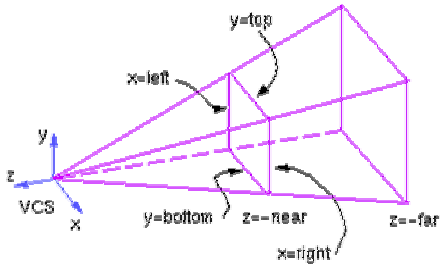


OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

View Volumes

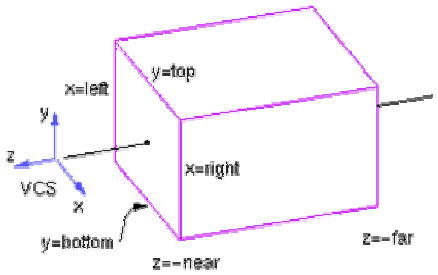
University of British Columbia

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



perspective view volume

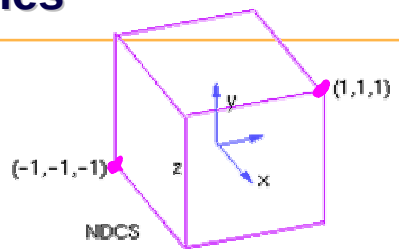
orthographic view volume

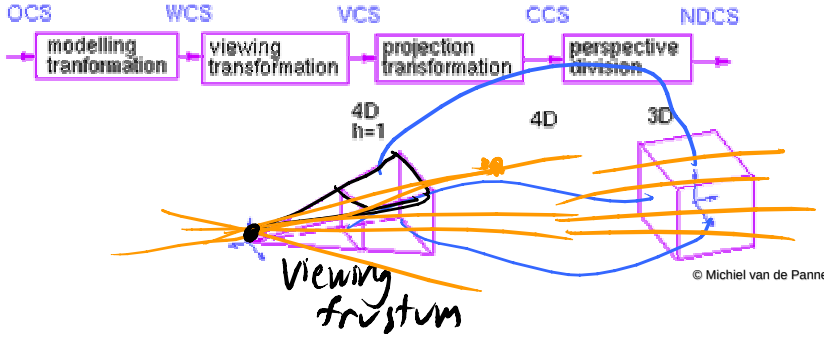


View Volumes

University of British Columbia

OpenGL
canonical view volume





View Volumes

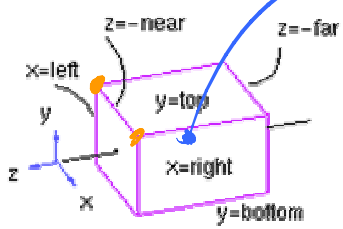
University of British Columbia

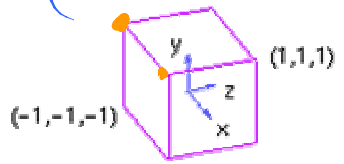
$z_{mid} = \left(\frac{-near + -far}{2}\right)$

Derivation – orthographic projections

$x_{mid} = (left + right)/2$

$y_{mid} = (top + bot)/2$





$scale\left(\frac{2}{right - left}, \ldots, \ldots\right) trans\left(-x_{mid}, -y_{mid}, -z_{mid}\right)$

View Volumes

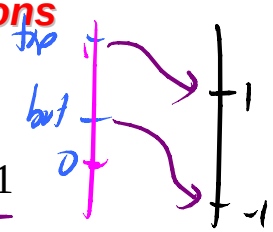
University of British Columbia

Derivation – orthographic projections

$$y' = a\,y + b$$

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

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



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

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

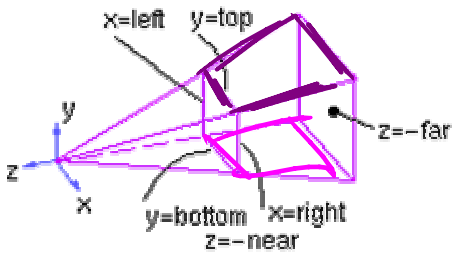
OpenGL

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

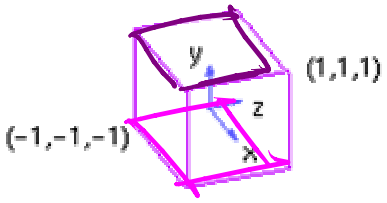

View Volumes

University of British Columbia

Derivation – Perspective case



VCS



NDCS

View Volumes

University of British Columbia

Derivation – Perspective case

earlier:

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

with additional ability to scale, etc.:

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

View Volumes

University of British Columbia

Derivation – Perspective case

top plane:

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

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

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

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

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

glFrustum(l, r, b, t, n, f)
gluPerspective(fovy, aspect, n, f)

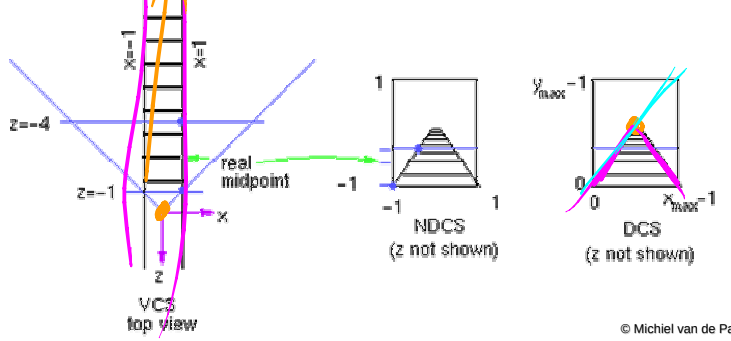
Perspective Transform

University of British Columbia

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 Transform

University of British Columbia

Example

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

/ h

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

Perspective Transform

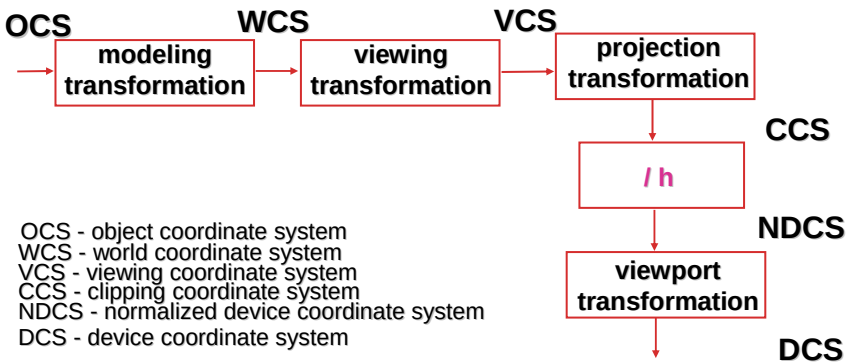
University of British Columbia

OpenGL

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

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


Projective Rendering Pipeline

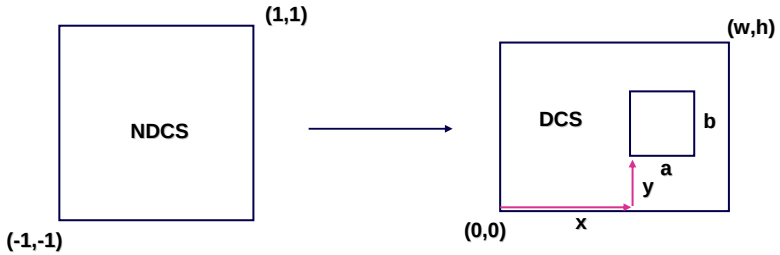
University of British Columbia



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



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

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

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

OpenGL

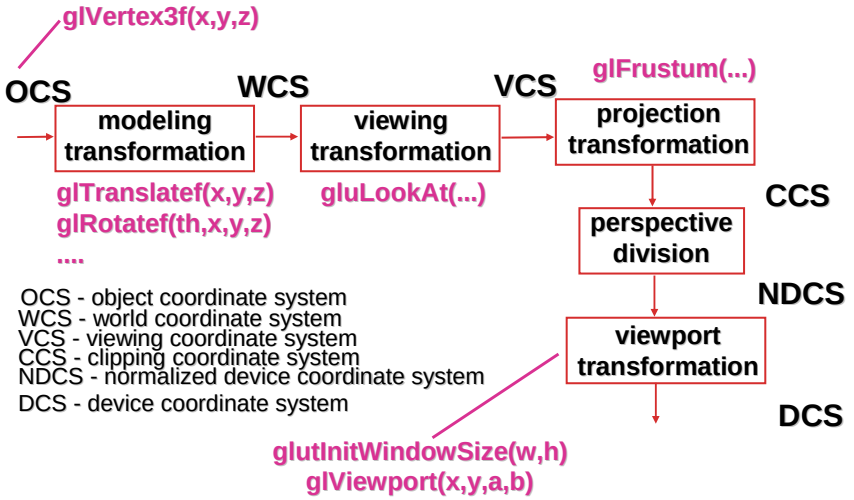
```
glViewport(x,y,a,b);
```

default:

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


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