University of British Columbia

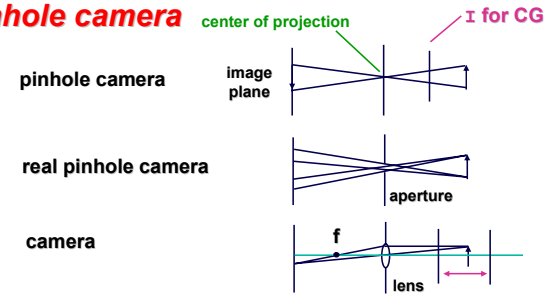
Viewing and Projection Transformations

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

Projection

University of British Columbia

Pinhole camera



price to pay: limited depth of field

Projection

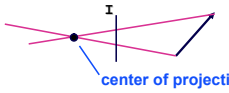
University of British Columbia

- definition

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

- parallel projection

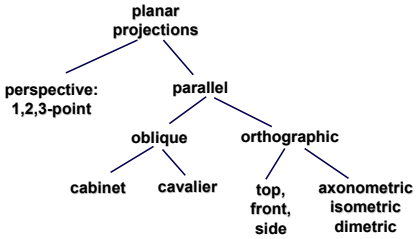
- perspective projection



Projections

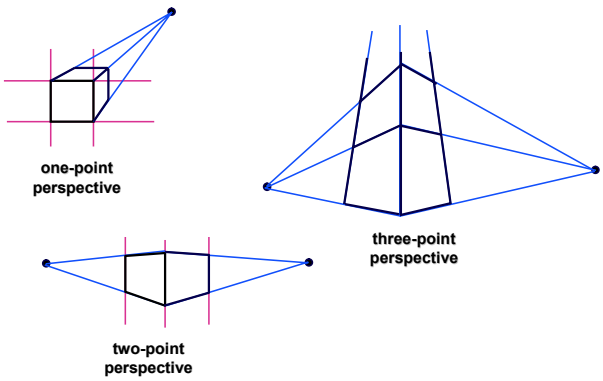
University of British Columbia

Taxonomy



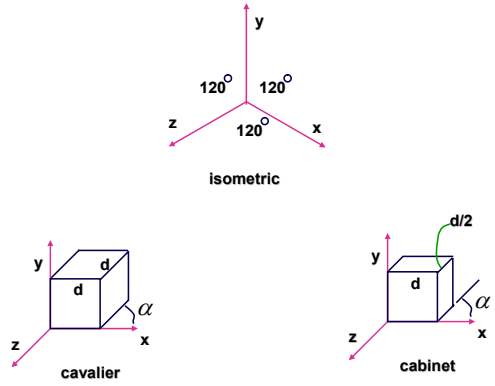
Projections

University of British Columbia



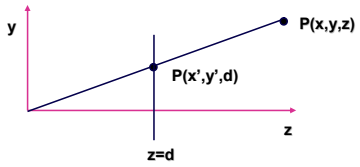
Projections

University of British Columbia



Basic Projection

University of British Columbia



similar triangles: $\frac{y'}{d} = \frac{y}{z}$ $\quad y' = \frac{y \cdot d}{z}$

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

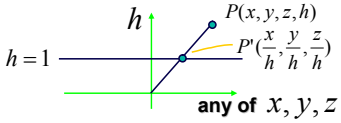
Homogeneous Coordinates

University of British Columbia

homogeneous $\qquad$ 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



equating slopes $\quad \frac{h}{x} = \frac{1}{x'}$

Homogeneous Matrices

University of British Columbia

Note:

- Multiplication of the matrix with a constant does not change the transformation!

$$\tilde{T}\left(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right) = \begin{bmatrix} m_{1,1} \cdot k & m_{1,2} \cdot k & m_{1,3} \cdot k & t_x \cdot k \\ m_{2,1} \cdot k & m_{2,2} \cdot k & m_{2,3} \cdot k & t_y \cdot k \\ m_{3,1} \cdot k & m_{3,2} \cdot k & m_{3,3} \cdot k & t_z \cdot k \\ 0 & 0 & 0 & k \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \cdot k \\ y' \cdot k \\ z' \cdot k \\ k \end{bmatrix}$$

$$\equiv \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = T\left(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right)$$

normalize: divide by w

Basic Projection

University of British Columbia

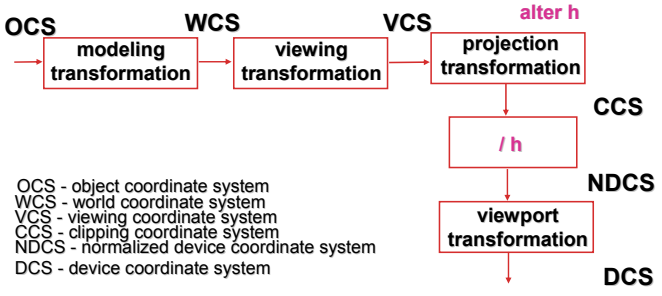
Using h and 4x4 matrices

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

$$\begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \xrightarrow{/\,h} \begin{bmatrix} x \cdot d/z \\ y \cdot d/z \\ d \end{bmatrix}$$

Projective Rendering Pipeline

University of British Columbia

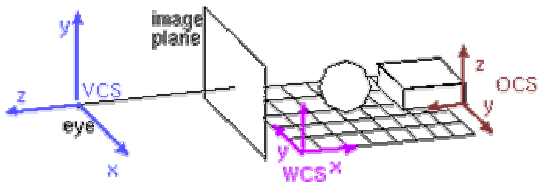


OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

Viewing Transformation

University of British Columbia

Positioning the camera

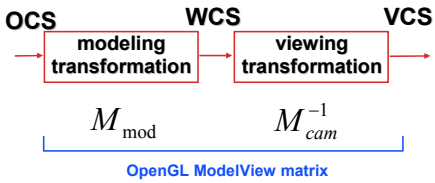


$$P_{WCS} = M_{\text{mod}} \cdot P_{OCS}$$
$$P_{WCS} = M_{cam} \cdot P_{VCS}$$

OpenGL ModelView matrix

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

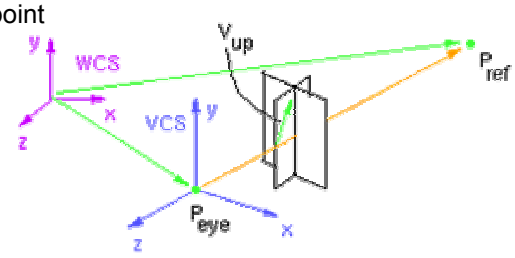
Viewing Transformation



Viewing Transformation

Defining the camera position

- eye point
- reference point
- up vector



Viewing Transformation

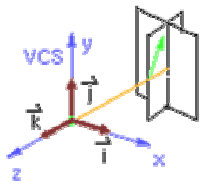
Computing Mcam

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

$$\vec{I} = \vec{V}_{up} \times \vec{k}$$

$$\vec{i} = \frac{\vec{I}}{|\vec{I}|}$$

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



Viewing Transformation

Computing Mcam

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

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

Viewing Transformation

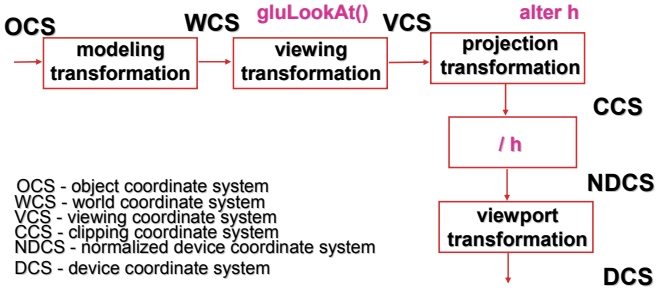
OpenGL

- `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
```

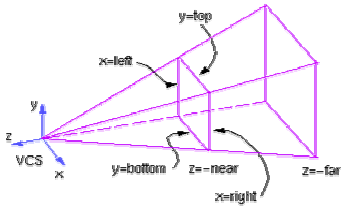
Projective Rendering Pipeline



View Volumes

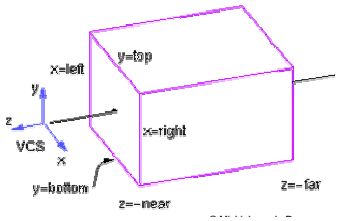
University of British Columbia

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



perspective view volume

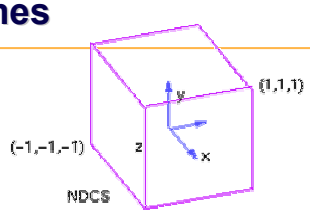
orthographic view volume

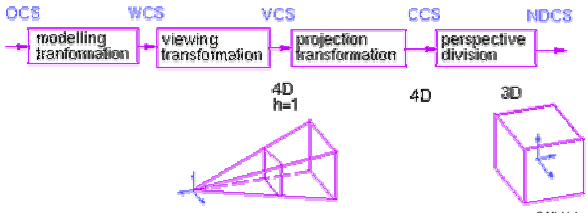


View Volumes

University of British Columbia

OpenGL canonical view volume

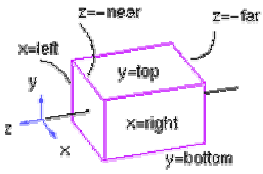




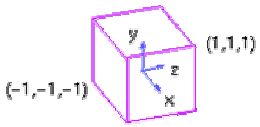
View Volumes

University of British Columbia

Derivation – orthographic projections



VCS



NDCS

View Volumes

University of British Columbia

Derivation – orthographic projections

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

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

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

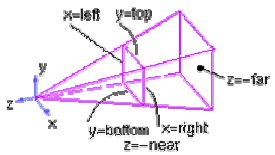
OpenGL

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

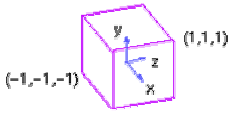
View Volumes

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:

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

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

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

similar process for solving 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

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

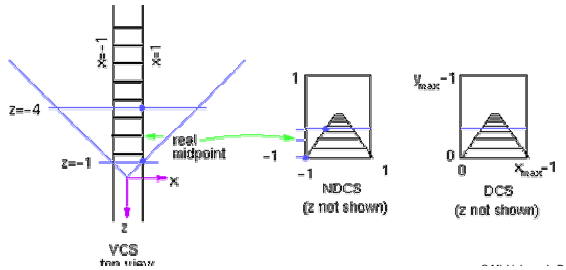
Perspective Transform

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

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

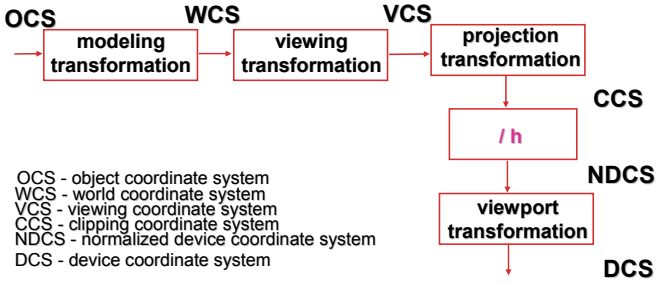
Perspective Transform

University of British Columbia

OpenGL

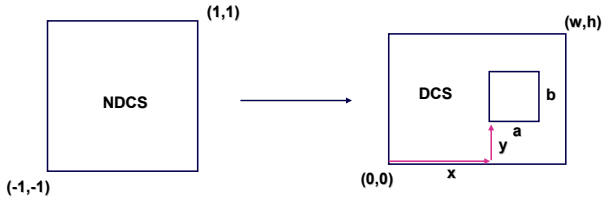
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

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

Projective Rendering Pipeline



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



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

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

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

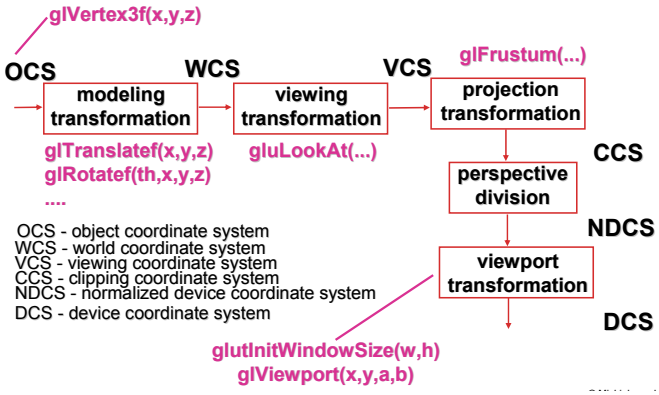
OpenGL

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

default:

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

Projective Rendering Pipeline



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…

- clipping and culling
- visibility
- scan conversion