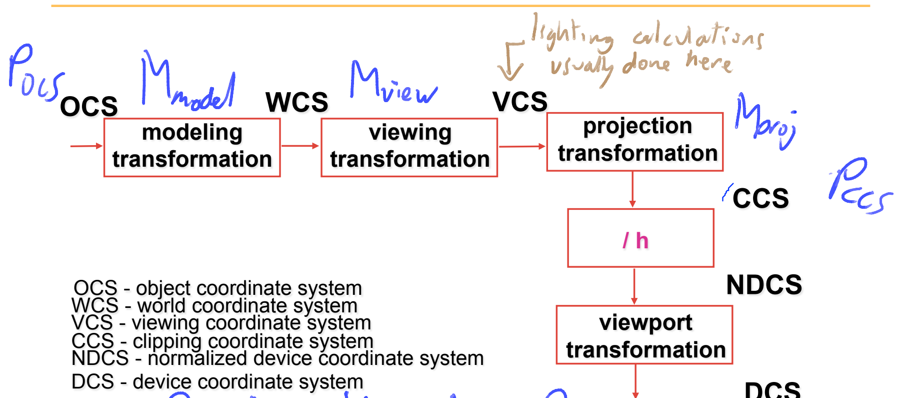
University of British Columbia

Viewing and Projection Transformations

University of British Columbia

Projective Rendering Pipeline



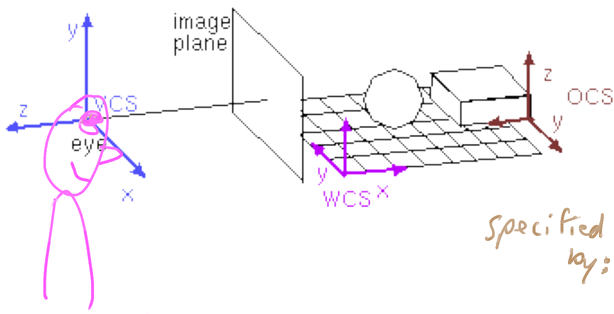
OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

$$P_{CCS} = \underbrace{M_{proj} \underbrace{M_{view} M_{model}}_{M_{MV}}}_{M_{MVP}} P_{OCS}$$

Viewing Transformation

University of British Columbia

Positioning the camera



- eye at origin of VCS
- looking down -z axis

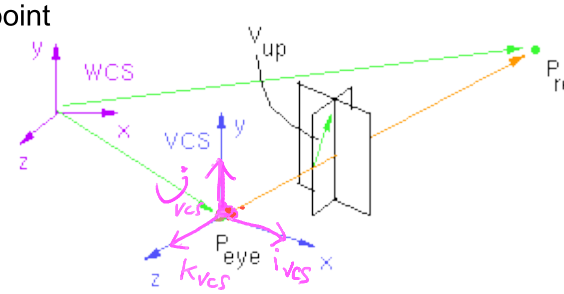
specified by: P_{eye}, P_{ref}, V_{up}

Viewing Transformation

University of British Columbia

Defining the camera position and orientation

- eye point
- reference point
- up vector



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

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

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

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

Viewing Transformation

University of British Columbia

Computing Mcam

$$\begin{array}{c} O_{VCS} \\ \\ \end{array}$$

$$M_{cam} = \begin{bmatrix} i & j & k & O_{VCS} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$M_{view} = M_{cam}^{-1} = \begin{bmatrix} & i & & -P_{eye} \cdot i \\ & j & & -P_{eye} \cdot j \\ & k & & -P_{eye} \cdot k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewing Transformation

University of British Columbia

graphics libraries:

- `matrix.setLookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz)`

  M

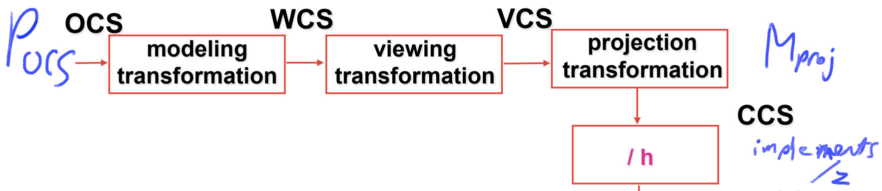
  $M \leftarrow M_{view}$

- `matrix.lookAt(ex,ey,ez,rx,ry,rz,ux,uy,uz`

  $M \leftarrow M * M_{view}$

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

M.setPerspective( )
M.lookAt( )
M.translate( ) etc.

$M = M_{proj}$

$M = M_{proj} \cdot M_{view}$

$M = M_{proj} \, M_{view} \cdot M_{model}$

Projection

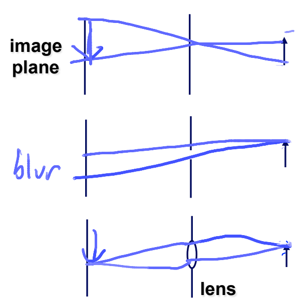
University of British Columbia

Pinhole camera

pinhole camera

real pinhole camera

camera



Projection

University of British Columbia

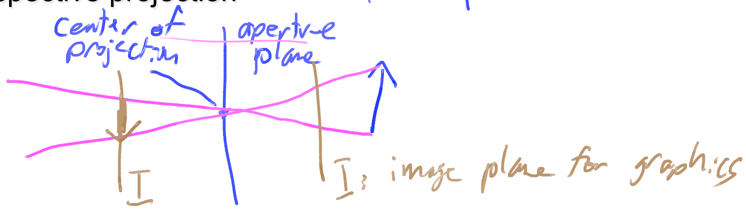
- definition

mapping $f: \mathbb{R}^n \to \mathbb{R}^m$ $m<n$ $\qquad p' = f(p)$

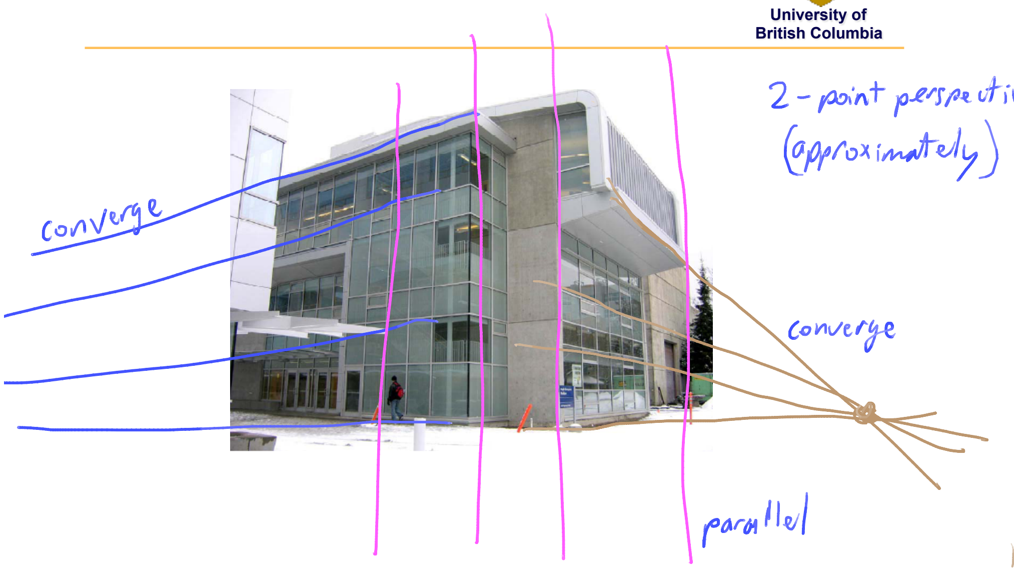
$\mathbb{R}^3 \to \mathbb{R}^2$

- parallel projection

- perspective projection

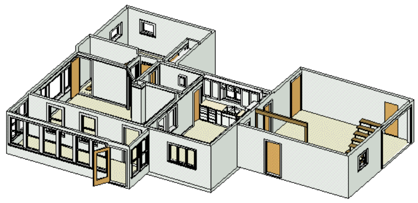


University of British Columbia

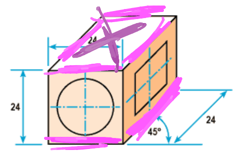


2-point perspective (approximately)

CAD drawings : parallel projections



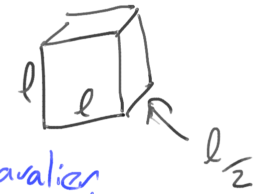
fortunecity.com



Cavalier projection

http://metal.brightcookie.com

cabinet projection



cavalier projection

→ distances on paper are preserved along the 3 axes

Projections

University of British Columbia

Taxonomy

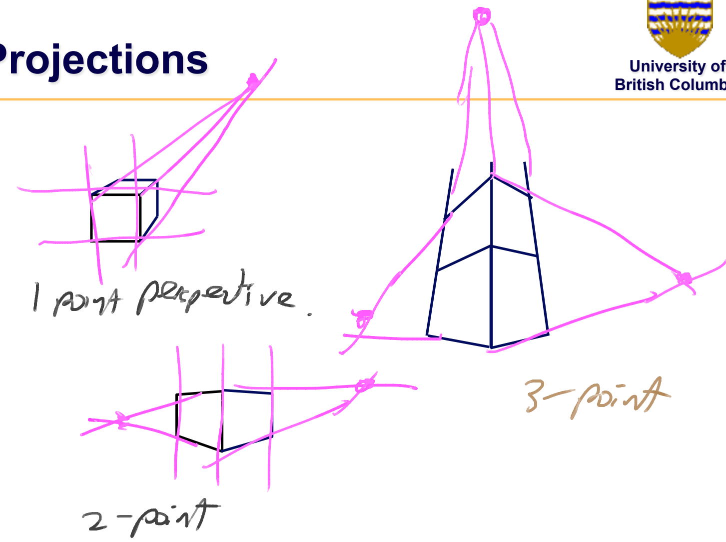
All these can be implemented by:

M_{proj} 4x4 matrix

- planar projections
  - perspective: 1,2,3-point
    - Camera and most CG
  - parallel
    - oblique
      - cabinet
      - cavalier
        - Some design drawings
    - orthographic
      - top, front, side
        - most design drawings
      - axonometric isometric dimetric
        - some design drawings

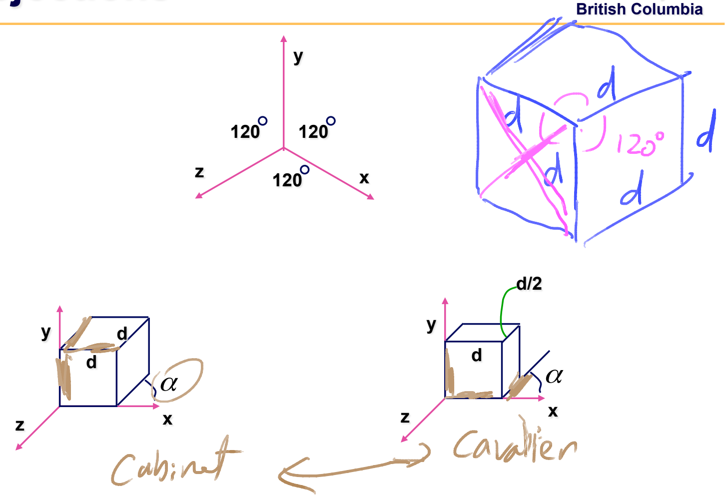
Projections

University of
British Columbia



Projections

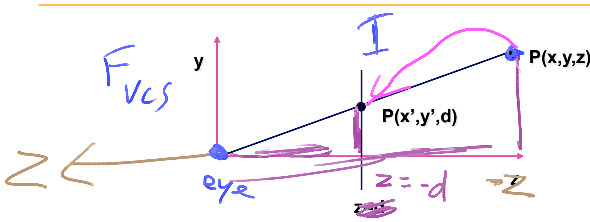
University of
British Columbia



Perspective

Basic Projection

University of
British Columbia



$p' = f(p)$

Similar triangles: $\frac{\text{rise}}{\text{run}} = \frac{y}{z} = \frac{y'}{-d} \Rightarrow$

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

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

$z' = -d$

we will be calling this

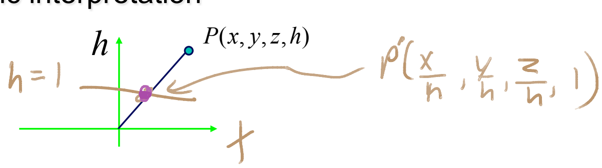
$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{VCS} \xrightarrow{\text{divide by } \frac{-z}{d}} \begin{bmatrix} -d\frac{x}{z} \\ -d\frac{y}{z} \\ -d \end{bmatrix}$

Homogeneous Coordinates

University of
British Columbia

homogeneous		cartesian
$(x, y, z, 1)$		
(x, y, z, h)	$\rightarrow$	$\left(\boxed{\frac{x}{h}, \frac{y}{h}, \frac{z}{h}}, 1\right)$

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



Basic Projection

University of British Columbia

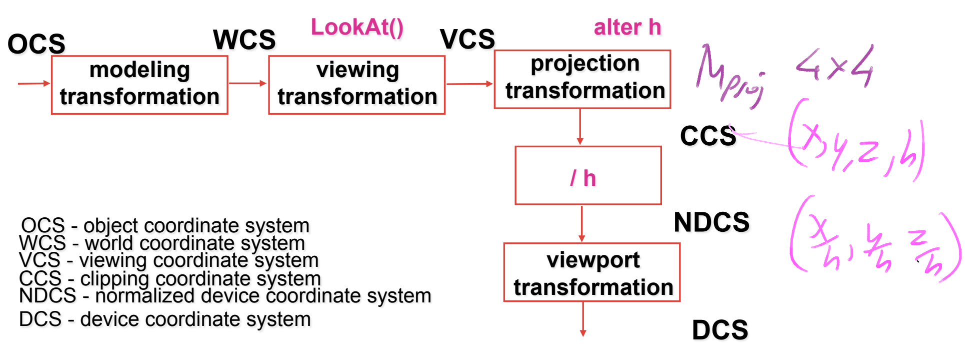
Using h and 4x4 matrices

$$\begin{bmatrix} x \\ y \\ z \\ -z/d \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & -1/d & 0 \end{bmatrix}}^{M_{proj}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad VCS$$

$$\xrightarrow{/h} \begin{bmatrix} \frac{-d}{z} x \\ \frac{-d}{z} y \\ \cancel{\frac{-d}{z}} \cancel{z} = -d \end{bmatrix}$$

Projective Rendering Pipeline

University of British Columbia

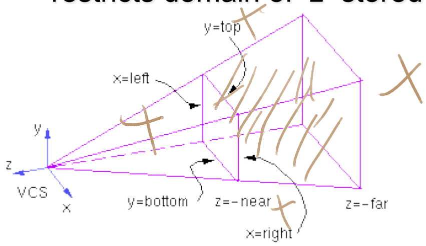


OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

View Volumes

University of British Columbia

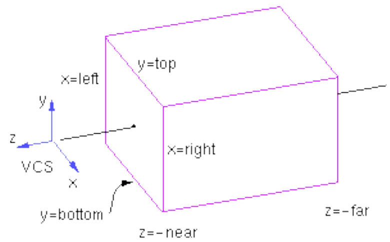
- specifies field-of-view, used for clipping
- restricts domain of ***z*** stored for visibility test



perspective view ~~volume~~

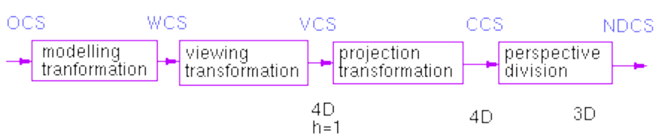
frustum

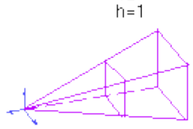
orthographic view volume

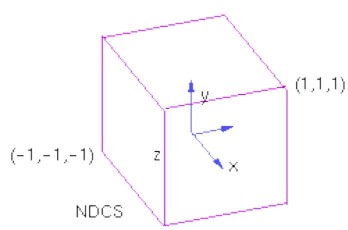


View Volumes

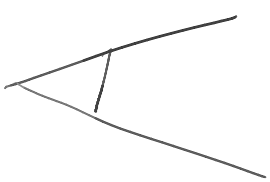
University of British Columbia







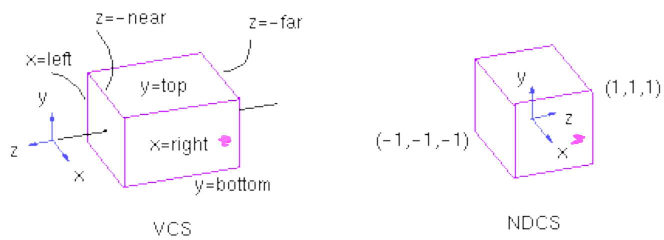
OpenGL
canonical view volume

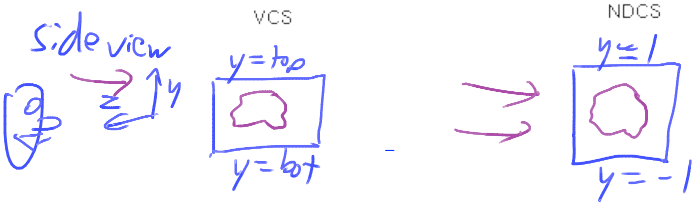


View Volumes

University of British Columbia

Derivation – orthographic projections





View Volumes

University of British Columbia

y' : y_{NDCS}

y : y_{VCS}

Derivation – orthographic projections

$\Rightarrow y' = a \cdot y + b$

$y = top \Rightarrow y' = 1$

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

$1 = a \cdot (top) + b$

$-1 = a \cdot (bot) + b$

solving for a and b gives:

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

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

View Volumes

University of British Columbia

Derivation – orthographic projections

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

a

b

$P_{NDCS} = P_{CCS} = M_{proj} P_{VCS}$

OpenGL

```
matrix.Ortho(left,right,bot,top,near,far);
matrix.setOrtho(left,right,bot,top,near,far);
```

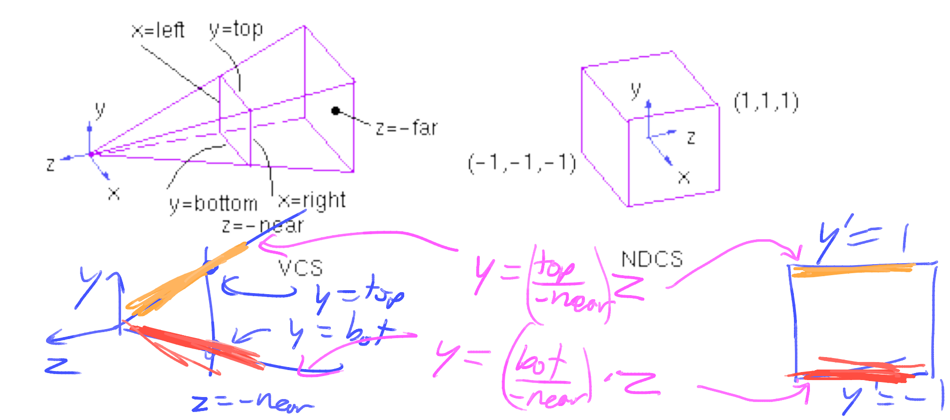
$M \leftarrow M \cdot M_{proj}$

$M \leftarrow M_{proj}$

View Volumes

University of British Columbia

Derivation – Perspective case



View Volumes

University of British Columbia

Derivation – Perspective case

earlier:

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

or equivalently, multiply by $-d$: $\begin{bmatrix} -d & & & \\ & -d & & \\ & & -d & \\ & & -1 & \end{bmatrix}$

with additional ability to scale, etc.:

CCS NDCS

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

$\Leftarrow /h$ CCS VCS

$x' = Ex + Az$ $/h'$

$y' = Fy + Bz$

$z' = Cz + D$

$h' = -z$

$\Rightarrow$

$x'' = E\frac{x}{-z} - A$

$y'' = -F\frac{y}{z} - B$

$z'' = -C - \frac{D}{z}$

View Volumes

University of British Columbia

Derivation – Perspective case

top plane: $y = \frac{top}{-near} \cdot z \Rightarrow y'' = 1$

$1 = -F\frac{y}{z} - B$

$1 = -F\left(\frac{top}{-near}\frac{z}{z}\right) - B$

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

two equations, two unknowns

$$\begin{cases} 1 = -F\frac{top}{-near} - B \\ -1 = -F(\quad) - B \end{cases}$$

View Volumes

University of
British Columbia

n = near
f = far
l = left
r = right

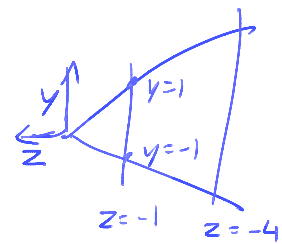
t = top
b = bottom

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

Example

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

F B



Perspective Transform

University of
British Columbia

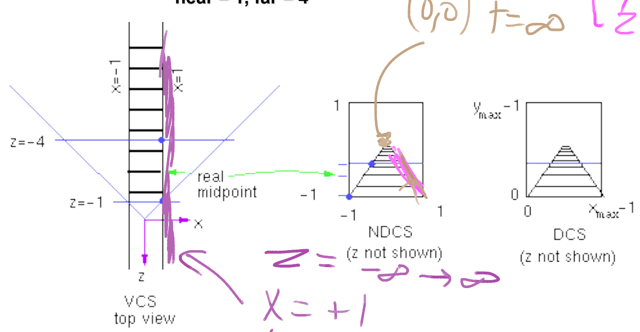
Example

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

(0,0) at t = ∞

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{NDCS} = \begin{bmatrix} -\frac{1}{z} \\ \frac{1}{z} \\ \frac{5}{3} + \frac{8}{3}\frac{1}{z} \end{bmatrix}$$



z = -∞ → ∞
x = +1
y = -1

Perspective Transform

University of British Columbia

Example

$$\begin{bmatrix} 1 \\ -1 \\ -\frac{5}{3}z-\frac{8}{3} \\ -z \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -\frac{5}{3} & -\frac{8}{3} \\ & & -1 & \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ z \\ 1 \end{bmatrix}$$

/ h

$$\begin{bmatrix} -\left(\frac{1}{z}\right) \\ \left(\frac{1}{z}\right) \\ \frac{5}{3}+\frac{8}{3}\left(\frac{1}{z}\right) \end{bmatrix} \text{NDCS}$$

center of screen

$$z = -\infty \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 5/3 \end{bmatrix}$$

Perspective Transform

University of British Columbia

OpenGL

old OpenGL

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

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

fovy: field of view y

aspect: w/h of display

m.setFrustum() $M \leftarrow M_{proj}$

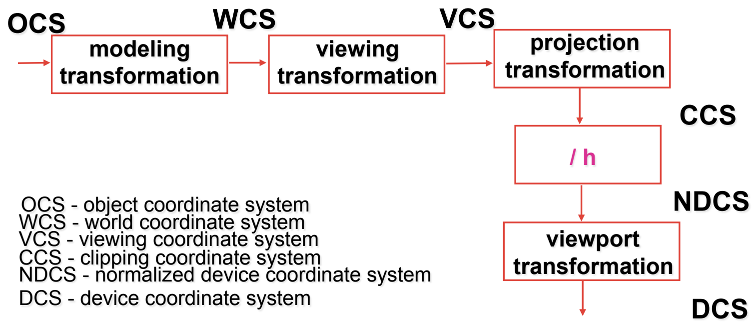
m.Frustum(left, right ...) $M \leftarrow M \cdot M_{proj}$

m.setPerspective(fovy, aspect, near, far) $M \leftarrow M_{proj}$

m.Perspective(...) $M \leftarrow M \cdot M_{proj}$

Projective Rendering Pipeline

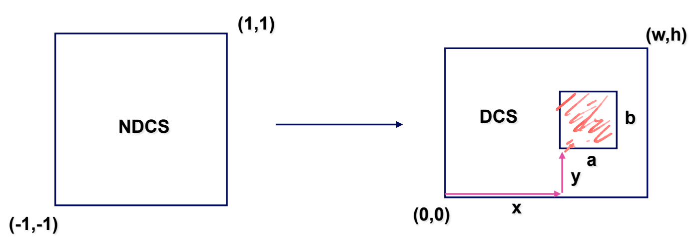
University of British Columbia



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



WebGL

```
gl.viewport(x,y,a,b);
```

default:

```
gl.viewport(0,0,w,h);
```

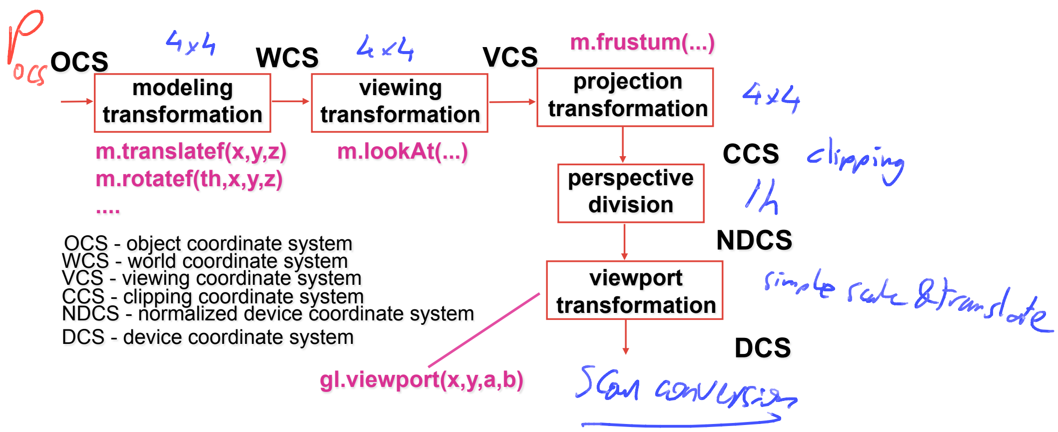
typical

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

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

Projective Rendering Pipeline

University of British Columbia



Coming Up…

University of British Columbia

- clipping and culling
- visibility
- scan conversion