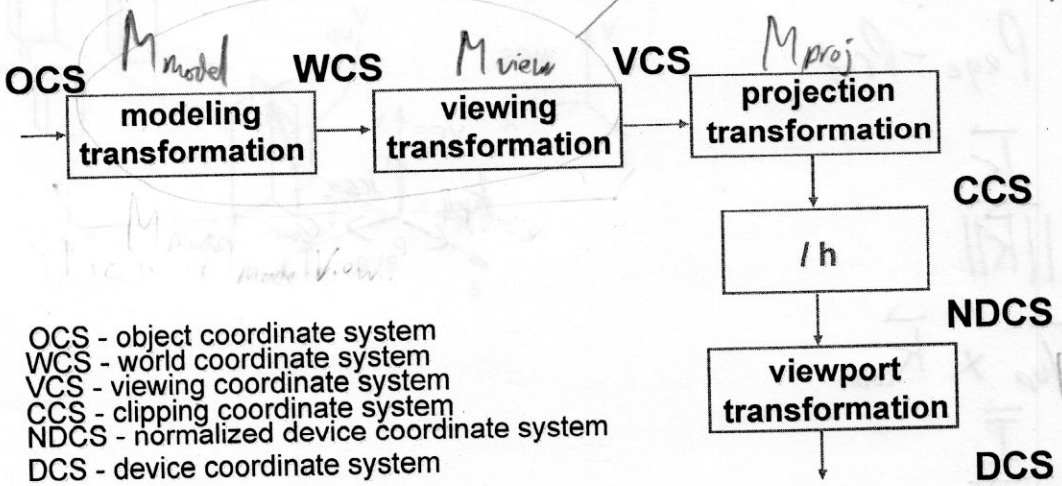
Viewing and Projection Transformations

Projective Rendering Pipeline

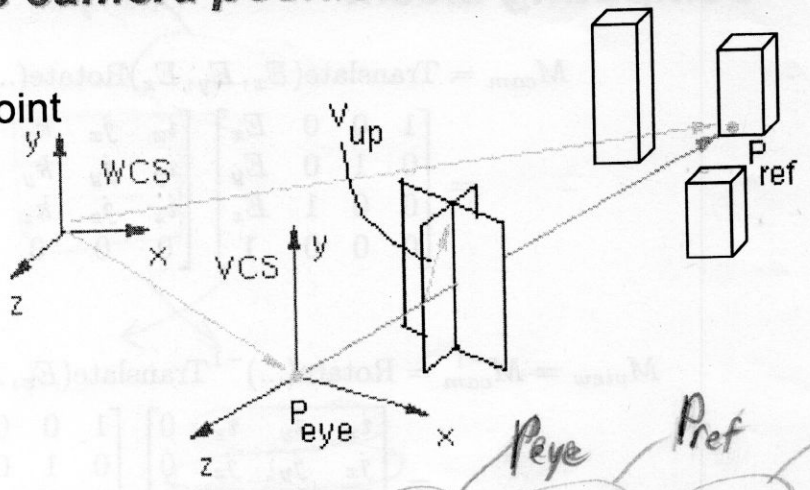


OCS - object coordinate system
WCS - world coordinate system
VCS - viewing coordinate system
CCS - clipping coordinate system
NDCS - normalized device coordinate system
DCS - device coordinate system

Viewing Transformation

Defining the camera position and orientation

- eye point
- reference point
- up vector



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

three.js: `matrix.lookAt(eye,ref,up)`

Computing i,j,k

$\vec{K}_{cam} = P_{eye} - P_{ref}$

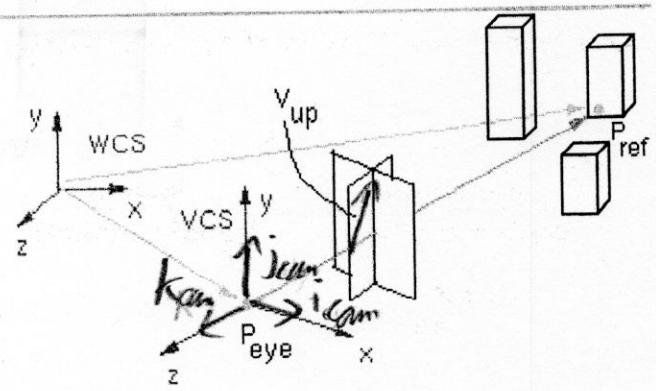
$\vec{k}_{cam} = \frac{\vec{K}}{\|\vec{K}\|}$

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

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

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

$O_{cam} = P_{eye}$.



Viewing Transformation

Computing Mcam

P_{eye}.

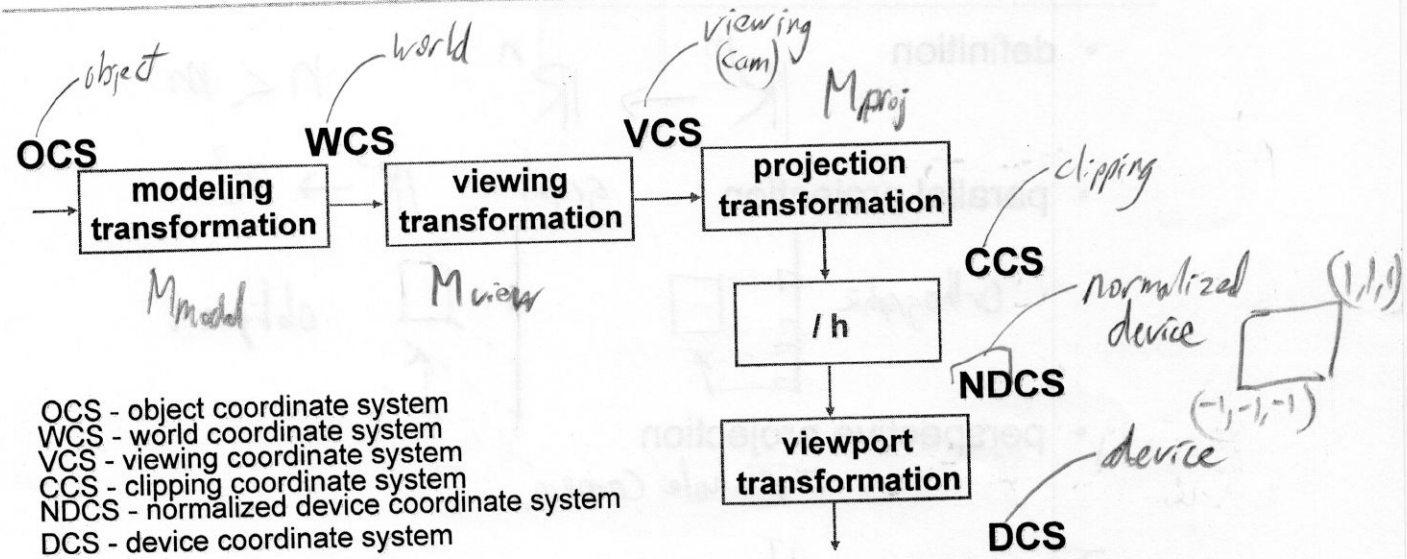
$$M_{cam} = \text{Translate}(E_x, E_y, E_z)\text{Rotate}(...)$$

$$= \begin{bmatrix} 1 & 0 & 0 & E_x \\ 0 & 1 & 0 & E_y \\ 0 & 0 & 1 & E_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_x & j_x & k_x & 0 \\ i_y & j_y & k_y & 0 \\ i_z & j_z & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} i & j & k & E \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{view} = M_{cam}^{-1} = \text{Rotate}(...)^{-1}\text{Translate}(E_x, E_y, E_z)^{-1}$$

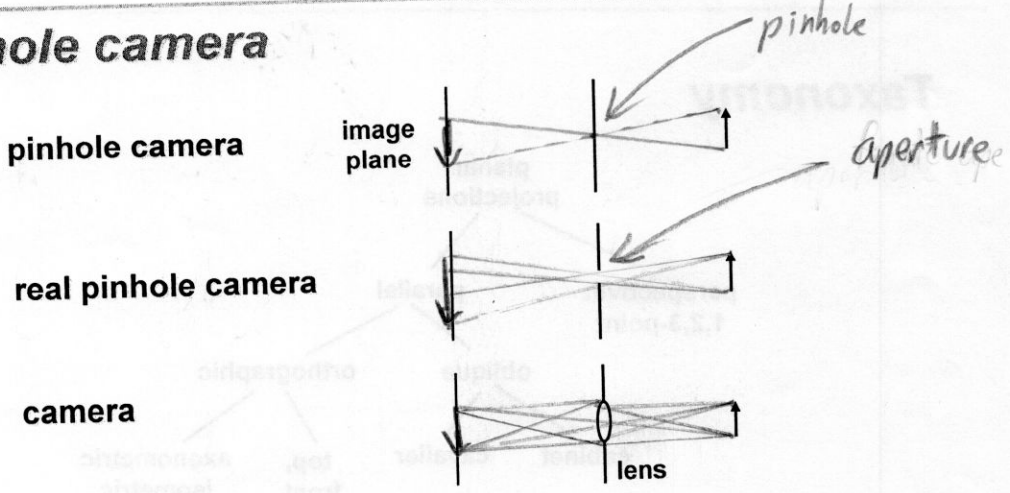
$$= \begin{bmatrix} i_x & i_y & i_z & 0 \\ j_x & j_y & j_z & 0 \\ k_x & k_y & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -E_x \\ 0 & 1 & 0 & -E_y \\ 0 & 0 & 1 & -E_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projective Rendering Pipeline



Projection

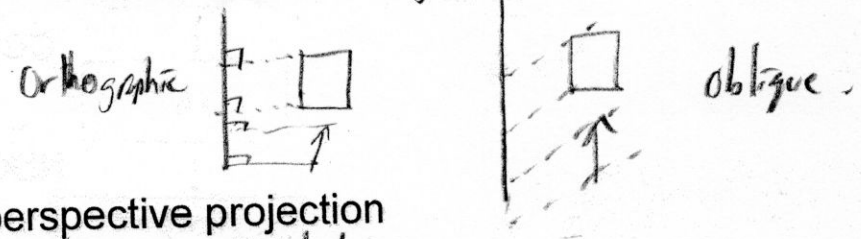
Pinhole camera



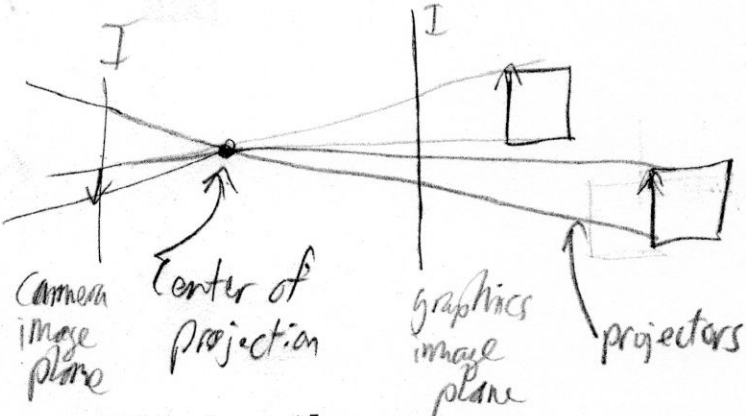
Projection

p p' $p' = f(p)$

- definition $\mathbb{R}^m \rightarrow \mathbb{R}^n$ $n < m$
- parallel projection graphics: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$

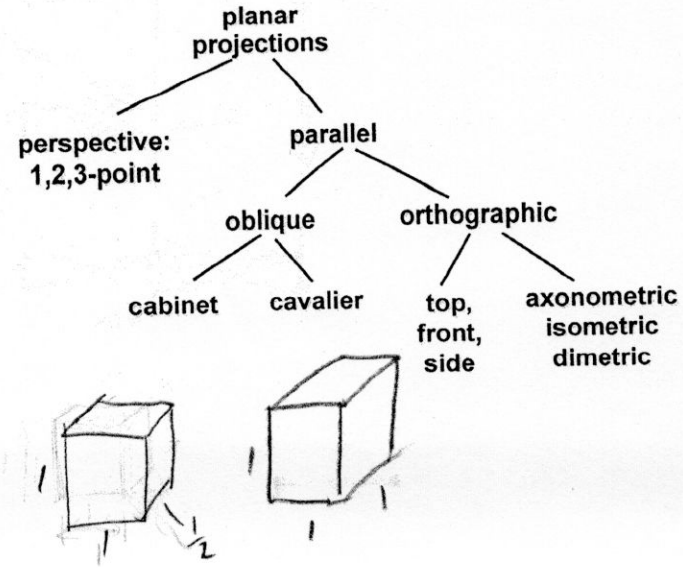


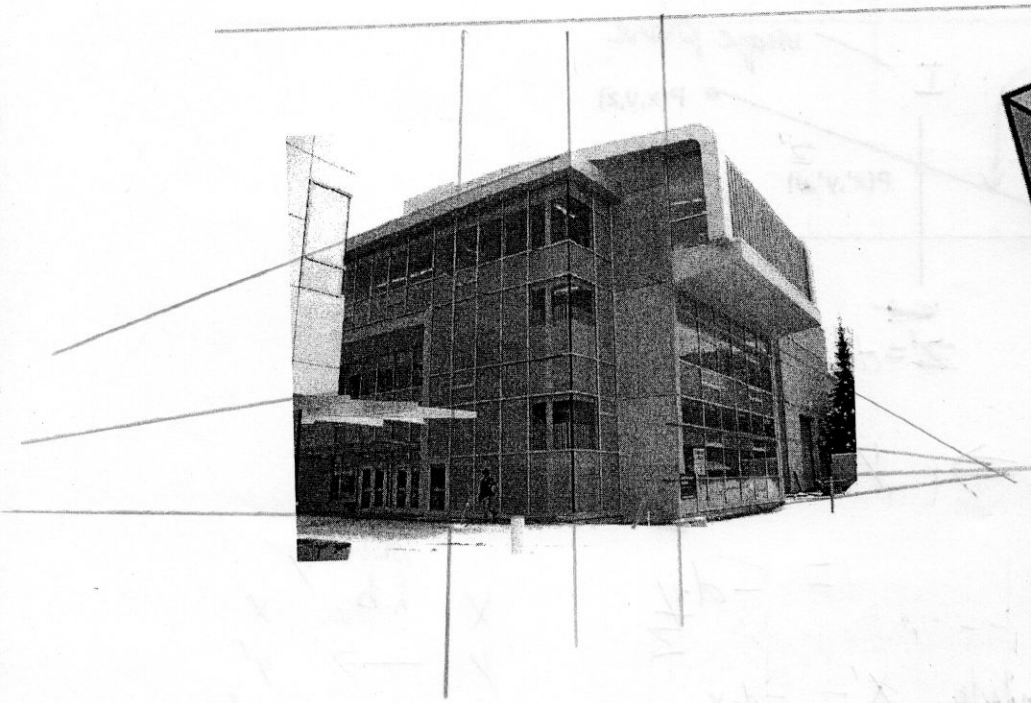
- perspective projection
  = pinhole camera.



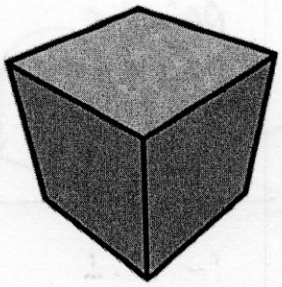
Projections

Taxonomy

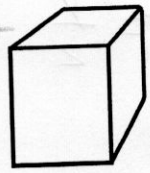




almost a 2-point perspective



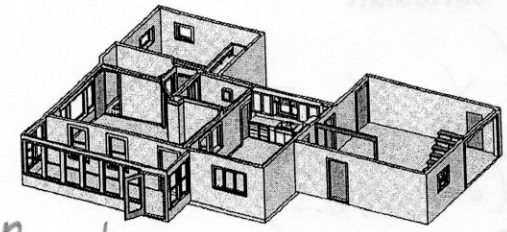
3-point



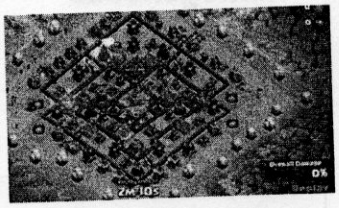
1-point

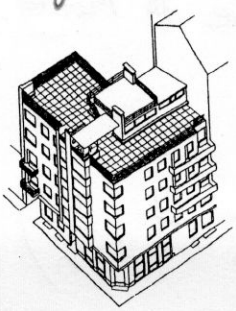
a rock

meaningless; only applies to scenes with ortho edges.

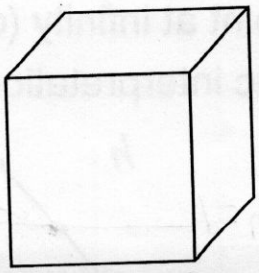


orthographic



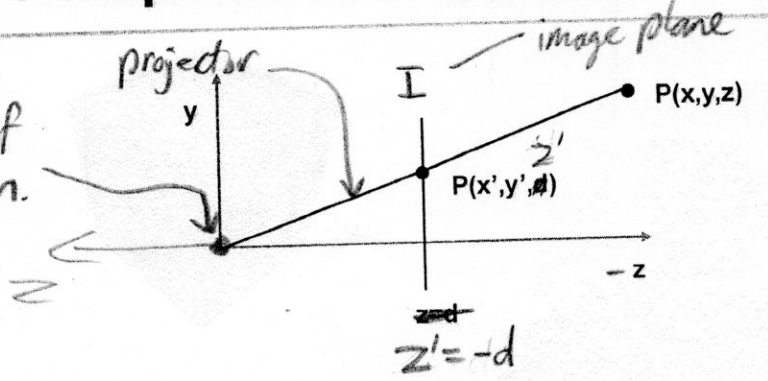


orthographic



oblique (cabinet)

Perspective Projection



Similar triangles, same slope

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

Similarly, $x' = -d \cdot \frac{x}{z}$

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

call this h: $h = \frac{z}{-d}$

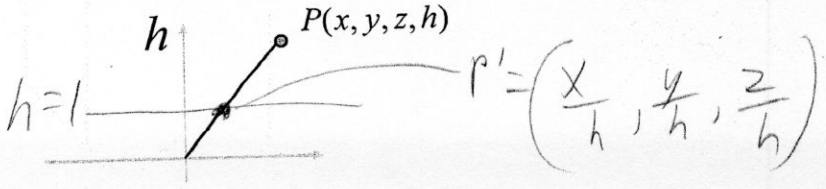
divide by h

$\begin{matrix} x \\ y \\ z \end{matrix} \longrightarrow \begin{matrix} x' \\ y' \\ z' \end{matrix}$

Homogeneous Coordinates

homogeneous	cartesian
(x, y, z, h)	$\left(\frac{x}{h}, \frac{y}{h}, \frac{z}{h}\right)$
$(5, 5, 5, 10)$	$(0.5, 0.5, 0.5)$

- redundant representation
- h=0: point at infinity (direction) $\Rightarrow$ use to model vectors.
- geometric interpretation



Perspective Projection

Using h and 4x4 matrices

Simple version of M_{proj} that implements the previous perspective projection

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

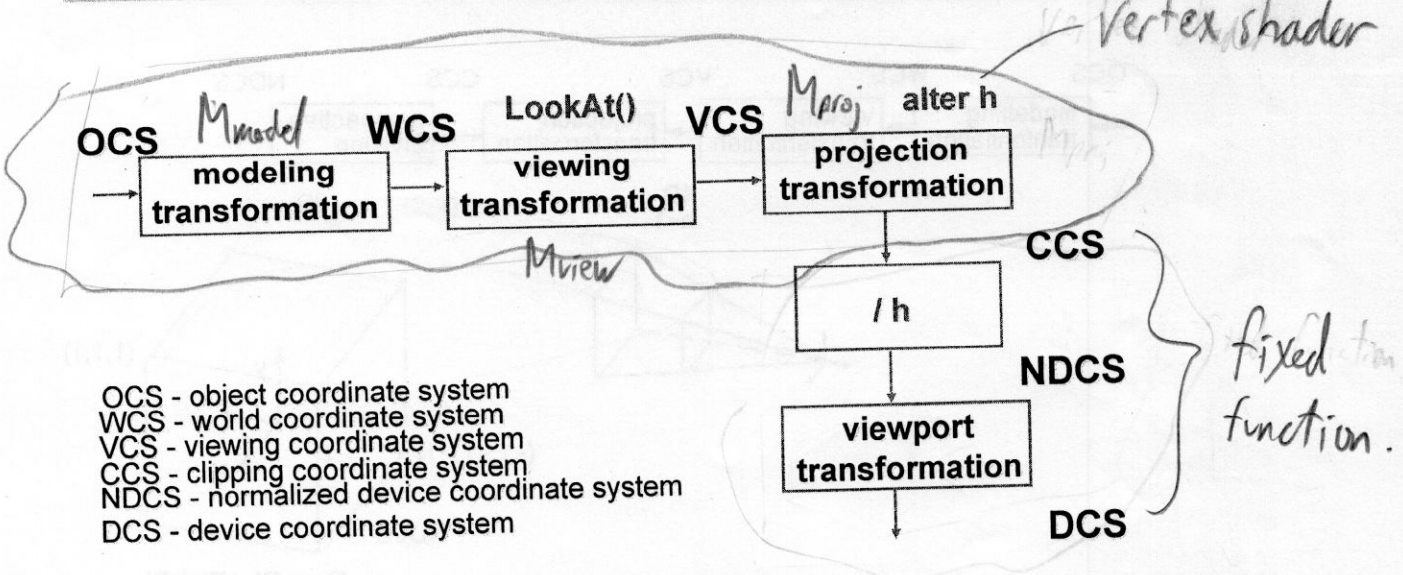
$/h$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -d\frac{x}{z} \\ -d\frac{y}{z} \\ -d\frac{z}{z} \end{bmatrix}$$

Note: The projected image size is inversely proportional to its z value

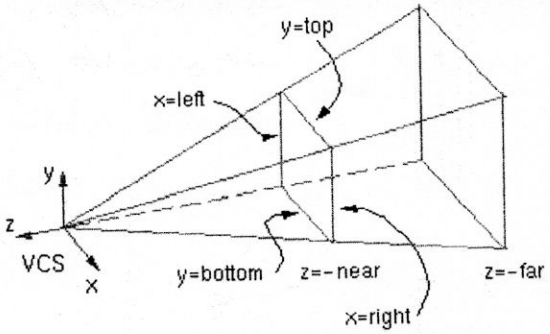
larger z value ⇒ further away ⇒ smaller x, y values.
(more negative)

Projective Rendering Pipeline



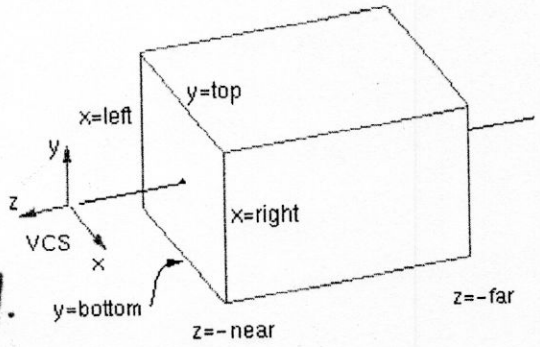
View Volumes

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



perspective view volume

"frustum": pyramid with the tip removed.

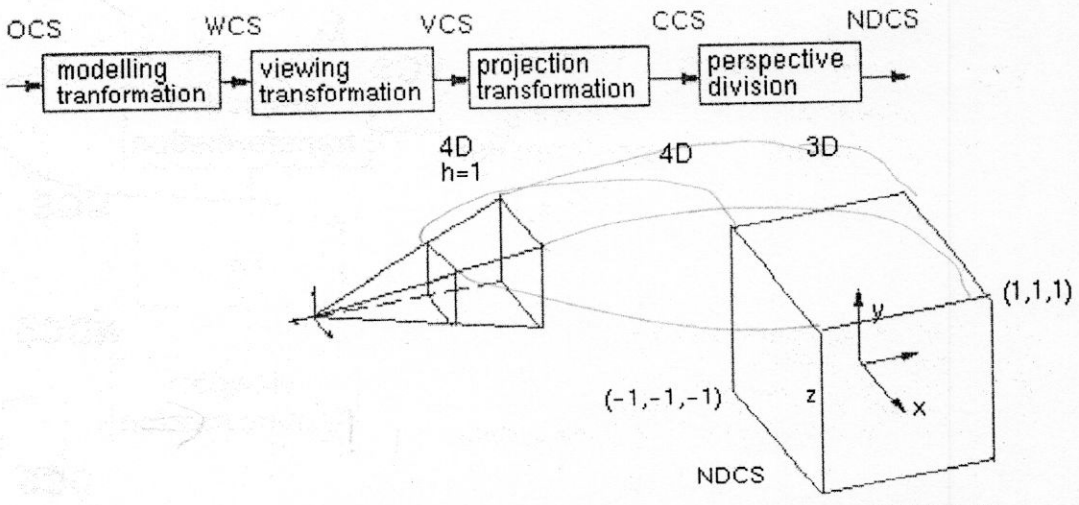


orthographic view volume

(interactive demos: WebGL, threejs.org)

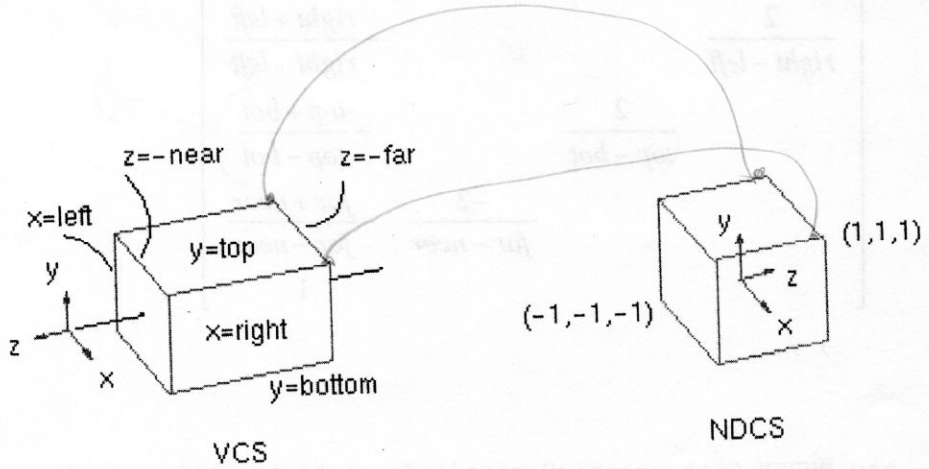
Note: the near and far plane remove objects that are too close or too far.

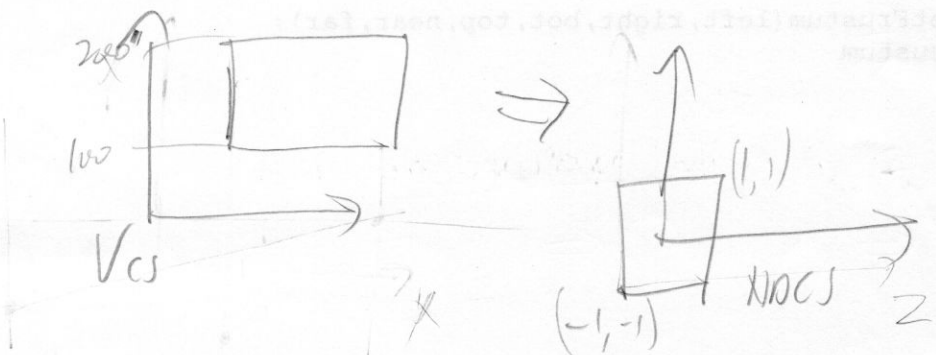
View Volumes



OpenGL / WebGL canonical view volume

Orthographic View Volume





Orthographic View Volume

Derivation

linear form: $y' = a \cdot y + b$

substitute:
$y = top \Leftrightarrow y' = 1$
$y = bot \Leftrightarrow y' = -1$

two eqns in two unknowns:
$$\begin{cases} 1 = a \cdot top + b \\ -1 = a \cdot bot + b \end{cases}$$

solving for a and b gives:

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

Orthographic View Volume

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

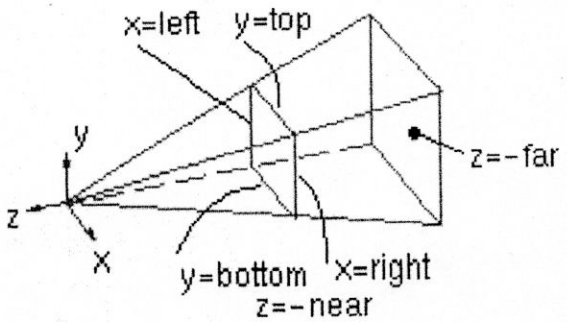
three.js

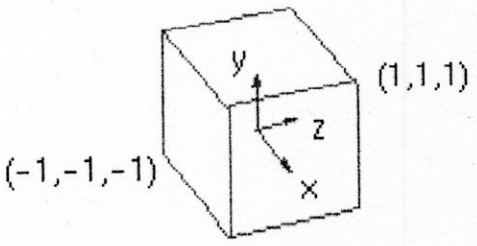
```
var cam = new THREE.OrthographicCamera(left,right,top,bot,near,far)
```

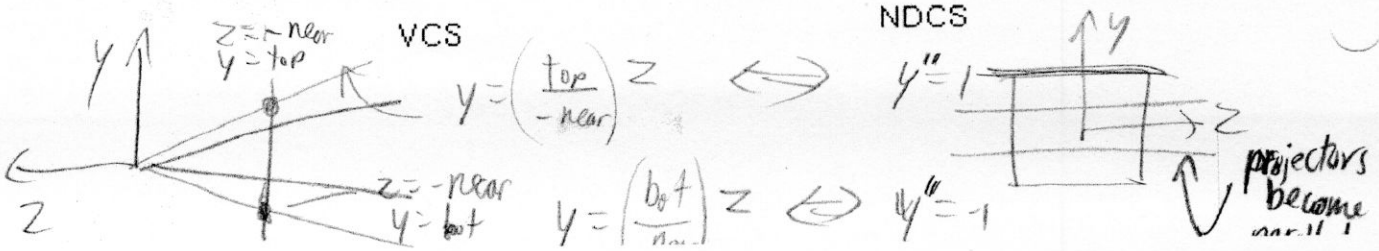
cuon-matrix.js

```
matrix.setFrustum(left,right,bot,top,near,far);
matrix.Frustum
```

Perspective View Volume







Perspective View Volume

Derivation

earlier:

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

$\div h$

$$\begin{bmatrix} -dx/z \\ -dy/z \\ -d \end{bmatrix}$$

with additional ability to scale, etc.:

$$\begin{bmatrix} Ex+Az \\ Fy+Bz \\ Cz+D \\ -z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ h' \end{bmatrix} = \begin{bmatrix} E & & A & \\ & F & B & \\ & & C & D \\ & & -1 & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

M_{proj}

$\div h$

$$\begin{bmatrix} -\frac{Ex}{z} - A \\ -\frac{Fy}{z} - B \\ -C - \frac{D}{z} \end{bmatrix}$$

Perspective View Volume

Derivation

top plane:

VCS: $y = \left(\frac{top}{-near}\right) z$

NDCS: $y'' = 1$

From above, we have:

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

substitute:

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

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

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

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

two eqns, solve for F, B

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

Perspective View Volume

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

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

three.js

```
var camera = new THREE.PerspectiveCamera(fov, aspect, near, far)
```

cuon-matrix.js

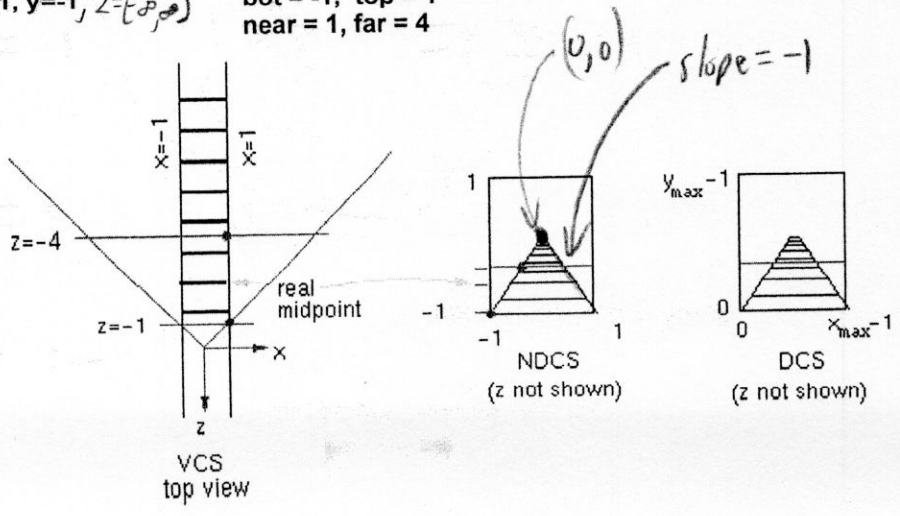
```
matrix.setFrustum(left,right,bot,top,near,far);
matrix.Frustum
```

Perspective Projection -- Example

Example

tracks in VCS:
left x=-1, y=-1, $z=(-\infty,\infty)$
right x=1, y=-1, $z=(-\infty,\infty)$

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



Perspective Projection -- Example

Example

M_{proj}

← points on the right railway track

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

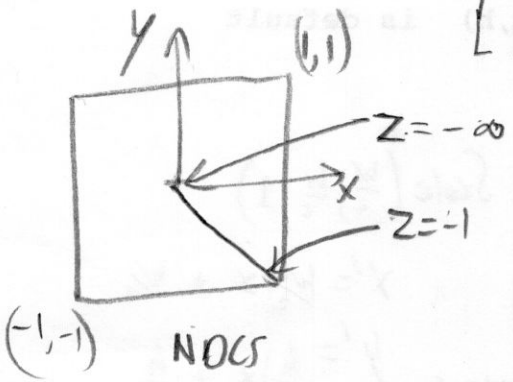
/ h

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

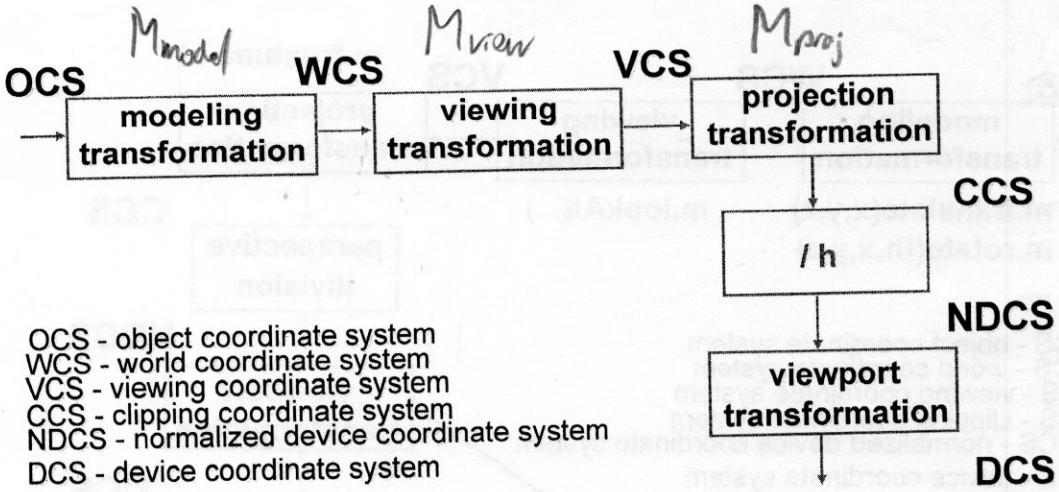
$z = -1 \Rightarrow \begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$z = -\infty \Rightarrow \begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$slope = \frac{y''}{x''} = \frac{-\frac{1}{z}}{\frac{1}{z}} = -1$

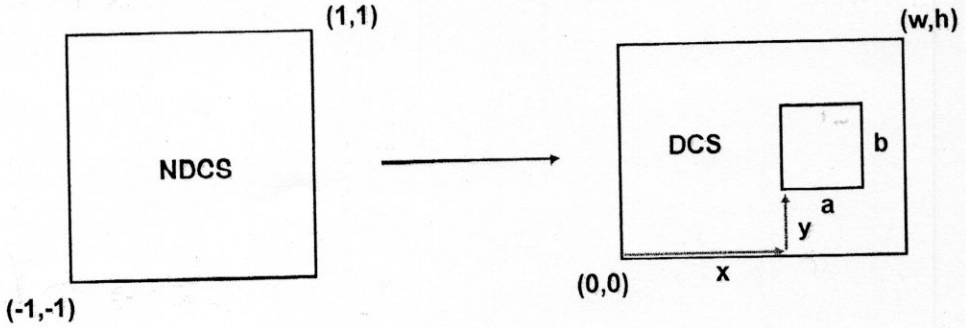


Projective Rendering Pipeline



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




```
three.js:   renderer.setViewport(x,y,a,b);
WebGL:      gl.viewport(x,y,a,b);
            gl.viewport(0,0,w,h)  is default
```

$$P_{DCS} = Trans\left(\frac{w}{2}, \frac{h}{2}, 0\right) Scale\left(\frac{w}{2}, \frac{h}{2}, 1\right)$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}_{DCS} = \begin{bmatrix} w/2 & 0 & 0 & w/2 \\ 0 & h/2 & 0 & h/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{NDCS}$$

$$x' = \frac{w}{2}x + \frac{w}{2}$$

$$y' = \frac{h}{2}x + \frac{h}{2}$$

Projective Rendering Pipeline