Geometrical Transformations

2D Linear transformations:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{aligned} x' &= a_{11}x + a_{12}y \\ y' &= a_{21}x + a_{22}y \end{aligned}$$

premultiplication form; transformation matrix is written before the position column vector

Linear because it satisfies the following two conditions necessary for any linear function $L(x)$:

1) $L(x+y) = L(x) + L(y)$ ← superposition
2) $L(cx) = cL(x)$ for any scalar c and position vectors x and y

Scaling: $\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$

Rotation: $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Horizontal shear: $\begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$

Vertical shear: $\begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$

Note that linear transformations are a sum of scaled input coords: they do not account for simple translation.

We want
$$x' = a_{11}x + a_{12}y + a_{13}$$
$$y' = a_{21}x + a_{22}y + a_{23}$$
$$\Downarrow$$
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Points are expressed in homogeneous coords.

* All transformations can now be treated as matrix multiplications.

In homogeneous coords, we add a third coord to a pt $\Rightarrow$ (x, y) is represented by (x, y, w)
w refers to the plane upon which the transformation operates.

The representation of a point is no longer unique:

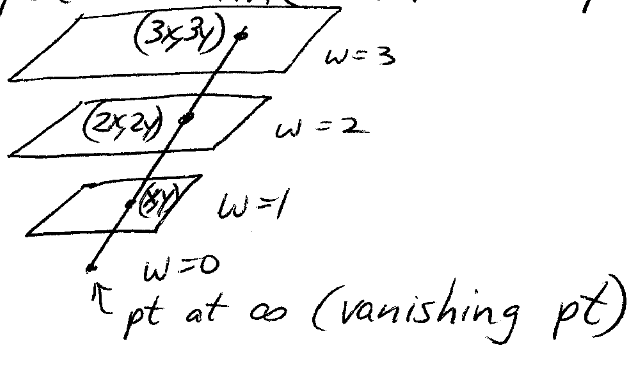
$[8,16,2] = [4,8,1] = [16,32,4]$

To recover any 2-D position vector $p = \begin{bmatrix} x \\ y \end{bmatrix}$ from $P_h = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}$, divide by the homogeneous coordinate w'.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'/w' \\ y'/w' \\ (w'/w') \end{bmatrix} \Rightarrow \begin{array}{l} x = x'/w' \\ y = y'/w' \end{array}$$

The points with $w=0$ are called points at infinity.

If we take all triples representing the same point (tx, ty, tw) with $t \neq 0$, then we get a line in 3-space.



For convenience, all input pts are made to lie on $w=1$ plane to trivially facilitate translation.

Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$ ← General representation

Note: matrix multiplication is not commutative

$$AB \neq BA$$

However,

$$(AB)^T = B^T A^T$$

Therefore, in postmultiplication form

$$[x' \; y' \; 1] = [x \; y \; 1] \begin{bmatrix} a_{11} & a_{21} & 0 \\ a_{12} & a_{22} & 0 \\ a_{13} & a_{23} & 1 \end{bmatrix}$$ ← used in Hill book

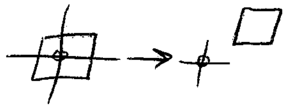
(In these notes we shall use the premult form though)

Division by w' is avoided by selecting $w = w' = 1$.

All affine mappings have $[0 \; 0 \; 1]$ as the last row.

Handles translation, rotation, scale, and shearing (skewing)

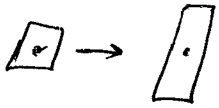
Translation: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$



Rotation: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

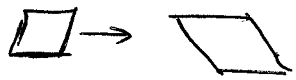
CCW rotation

Scale: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

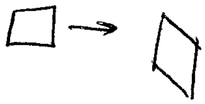


Horizontal Shear: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \alpha & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

x is linearly dep. on y as well as x

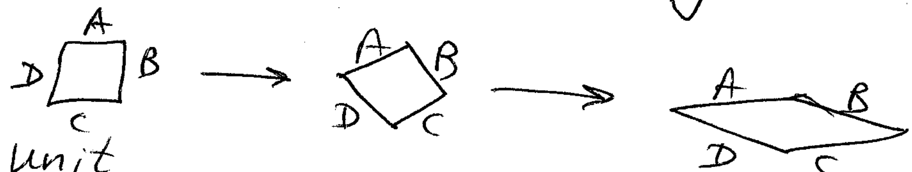


Vertical shear: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$



Properties: affine transformations preserve parallel lines, but not lengths or angles.

Ex:



unit square → 45° rotation → Scale in x, not in y

A // C
B // D
but ∠s and lengths changed

Note: We transform lines by transforming endpts + redrawing line through transformed endpts.

Composition of 2D Transformations

To rotate some object about an arbitrary point P_1 : translate such that $P_1 \rightarrow$ origin, rotate, translate such that pt at origin $\rightarrow P_1$
(premultiplication shown here)

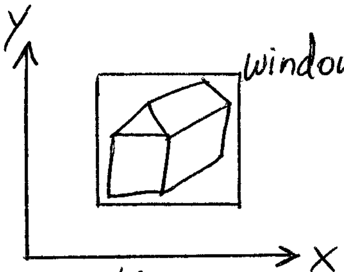
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{translate} \\ \text{origin to } P_1}} \cdot \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{rotate} \\ \text{about origin}}} \cdot \underbrace{\begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{translate} \\ P_1 \text{ to origin}}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_1(1-\cos\theta) + y_1 \sin\theta \\ \sin\theta & \cos\theta & y_1(1-\cos\theta) - x_1 \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Note: $(A \cdot (B \cdot C)) = ((A \cdot B) \cdot C)$ associative

$A \cdot B \neq B \cdot A$ not always commutative

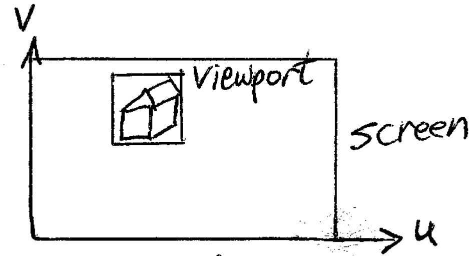
Window-to-Viewport Transformation



world coords

(uses unit meaningful to appln. program)

"world" refers to models being created or displayed to user



screen (device) Coordinates

<u>Note:</u> what we actually refer to as a window on the screen is known as a viewport in the computer graphics literature. CG preceded modern window systems.

Transformation matrix that maps world to viewport coordinates

last op — first op

$$M_{WV} = T(U_{min}, V_{min}) \cdot S\left(\frac{U_{max}-U_{min}}{X_{max}-X_{min}}, \frac{V_{max}-V_{min}}{Y_{max}-Y_{min}}\right) \cdot T(-X_{min}, -Y_{min})$$

$$= \begin{bmatrix} 1 & 0 & U_{min} \\ 0 & 1 & V_{min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{U_{max}-U_{min}}{X_{max}-X_{min}} & 0 & 0 \\ 0 & \frac{V_{max}-V_{min}}{Y_{max}-Y_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -X_{min} \\ 0 & 1 & -Y_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{U_{max}-U_{min}}{X_{max}-X_{min}} & 0 & -X_{min} \cdot \frac{U_{max}-U_{min}}{X_{max}-X_{min}} + U_{min} \\ 0 & \frac{V_{max}-V_{min}}{Y_{max}-Y_{min}} & -Y_{min} \cdot \frac{V_{max}-V_{min}}{Y_{max}-Y_{min}} + V_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

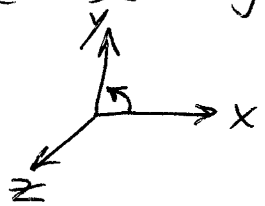
$$\boxed{P' = M_{WV} P}$$

3D Transformations

$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$ is a 3D point in homogeneous coordinates

Homogenizing $\underbrace{\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}}_{\text{homogeneous coord.}}$ gives us $\underbrace{\begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}}_{\text{3D coord}}$

We use right-handed (RH) coordinate system



$$x \quad y \quad z$$

$$x \times y \rightarrow z \qquad \overset{1}{x} \quad \overset{2}{y} \quad \overset{3}{\boxed{z}}$$

$$y \times z \rightarrow x \qquad \overset{3}{\boxed{x}} \quad \overset{1}{y} \quad \overset{2}{z}$$

$$z \times x \rightarrow y \qquad \overset{2}{x} \quad \overset{3}{\boxed{y}} \quad \overset{1}{z}$$

cross product

Translation: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Rotation: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

R_z (rotation about z-axis)

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling: $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

All these matrices have inverses:

$$\underset{\substack{\text{identity}\\ \text{matrix}}}{I} = \underset{\substack{\text{inverse of}\\ \text{translation}}}{T^{-1}} \cdot \underset{\text{translation}}{T}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

← 2D case (same applies to 3D)

$$I = S^{-1} \cdot S$$

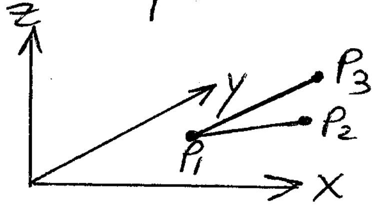
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$I = R^{-1} \cdot R$$

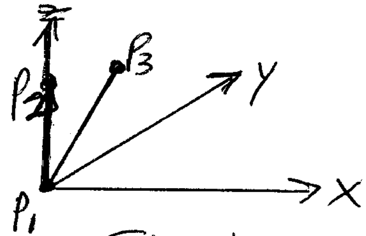
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) & 0 \\ \sin(-\theta) & \cos(-\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

<u>Note:</u> $\cos(-\theta) = \cos\theta$
$\sin(-\theta) = -\sin\theta$

Composition of 3D Transformations



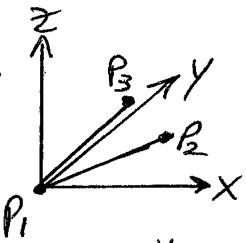
Initial



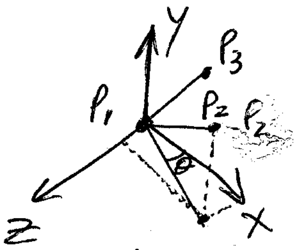
Final

1) Translate P_1 to origin
2) Rotate about y-axis such that P_1P_2 lies in yz-plane
3) Rotate about x-axis such that P_1P_2 lies on z-axis
4) Rotate about z-axis such that P_1P_3 lies on yz-plane

After step (1):

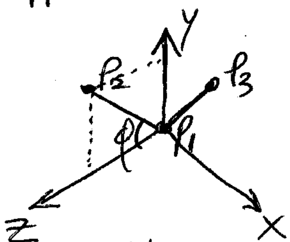


OR

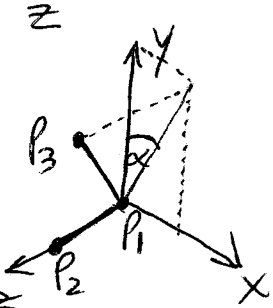


Note: $-(90-\theta)=\theta-90$

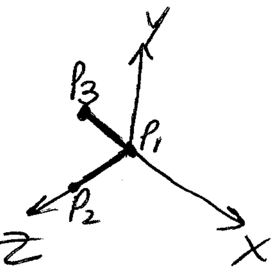
After step (2):



After step (3):



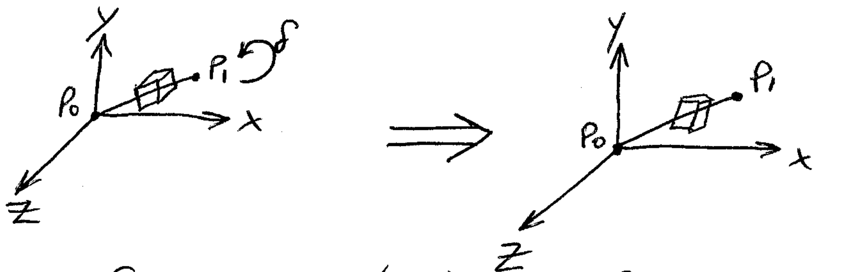
After step (4):



Composite matrix:

$$R_z(\alpha)\cdot R_x(\phi)\cdot R_y(\theta-90)\cdot T(-x_1,-y_1,-z_1)$$

Rotation About An Arbitrary Axis



So far, we only know R_x, R_y, R_z but not $R_{\text{arbitrary axis}}$

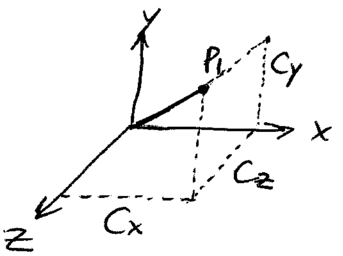
Strategy:

align arb. axis with z, Do same with object:
1) Translate P_0 to origin
2) Perform up to 2 rotations to make P_1 lie on z-axis (make rot. axis coincident with z-axis)

rotate:
3) Rotate about z-axis by angle δ

bring axis + obj back to original coord sys:
4) Perform inverse of (2)
5) Perform inverse of (1)

step (2):



To bring $P_1(x_1, y_1, z_1)$ onto z-axis, we must have x_1, y_1 become 0.

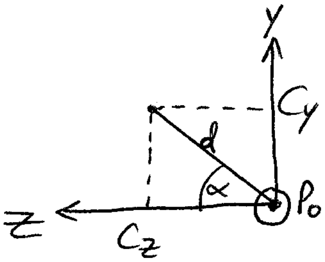
Rotate about x: $P_1 \to \underset{y=0}{xz}$ or $\overset{z=0}{xy}$ plane

Rotate P_1' about y: $P_1' \to \underset{x=0}{z}$

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow$$

x remains the same

what is α?

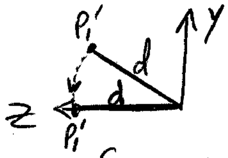


View from right

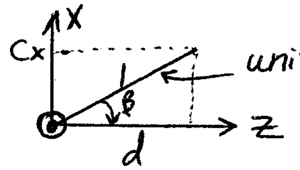
$\Longrightarrow$ $d = \sqrt{C_y^2 + C_z^2}$ (length of projection onto yz plane)

$\cos\alpha = \frac{C_z}{d}$ $\qquad \sin\alpha = \frac{C_y}{d}$

After rotation about x-axis, x rts



view from right



$y=0$

$P_1'(C_x, 0, d)$

$$1 = \sqrt{C_x^2 + d^2}$$

$$1 \stackrel{\checkmark}{=} \sqrt{C_x^2 + C_y^2 + C_z^2}$$

direction cosines picked so that $C_x^2 + C_y^2 + C_z^2 = 1$

$\cos\beta = d \qquad \sin\beta = C_x$

$$R_y = \begin{bmatrix} \cos(-\beta) & 0 & \sin(-\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-\beta) & 0 & \cos(-\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

← use $-\beta$ for clockwise rotation about y (not ccw)

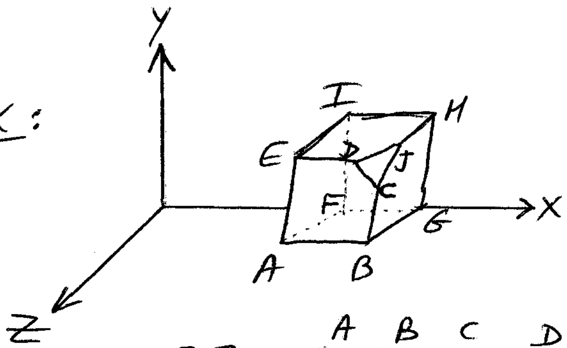
Now P_0P_1 lies on z-axis; rotate by desired δ

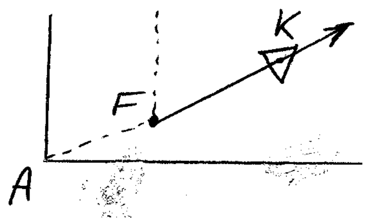
$$R_z = \begin{bmatrix} \cos\delta & -\sin\delta & 0 & 0 \\ \sin\delta & \cos\delta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underset{\substack{\uparrow \\ \text{composite} \\ \text{matrix}}}{M} = \underset{\substack{\uparrow \\ (x_0, y_0, z_0)}}{T^{-1}} \cdot \underset{\substack{\uparrow \\ -\alpha}}{R_x^{-1}} \cdot \underset{\substack{\uparrow \\ \beta}}{R_y^{-1}} \cdot \underset{\substack{\uparrow \\ \delta}}{R_z} \cdot \underset{\substack{\uparrow \\ -\beta}}{R_y} \cdot \underset{\substack{\uparrow \\ \alpha}}{R_x} \cdot \underset{\substack{\uparrow \\ (-x_0, -y_0, -z_0)}}{T}$$

$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \frac{\begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \\ z_1 - z_0 \end{bmatrix}}{\sqrt{(x_1-x_0)^2 + (y_1-y_0)^2 + (z_1-z_0)^2}}$$

Ex:





rot. axis goes thru F and K

$$[X] = \begin{array}{ccccccccccc} A & B & C & D & E & F & G & H & I & J & K \end{array}$$
$$\begin{bmatrix} 2 & 3 & 3 & 2.5 & 2 & 2 & 3 & 3 & 2 & 3 & 3 \\ 1 & 1 & 1.5 & 2 & 2 & 1 & 1 & 2 & 2 & 2 & 3 \\ 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1.5 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

1) Compute C_x, C_y, C_z :
$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \frac{\begin{bmatrix} 3-2 \\ 2-1 \\ 2-1 \end{bmatrix}}{\sqrt{(3-2)^2+(2-1)^2+(2-1)^2}} = \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$$

2) Translate F to origin :
$$T = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3) Rotate about x :
$$d = \sqrt{C_y^2 + C_z^2} = \sqrt{\tfrac{1}{3} + \tfrac{1}{3}} = \sqrt{\tfrac{2}{3}}$$
$$\cos\alpha = \frac{C_z}{d} = \frac{1}{\sqrt{2}} \quad ; \quad \sin\alpha = \frac{C_y}{d} = \frac{1}{\sqrt{2}}$$
$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4) Rotate about y:
$$\cos\beta = d \qquad \sin\beta = C_x$$
We must use $-\beta \Rightarrow \cos(-\beta) = d \quad \sin(-\beta) = -C_x$
$$R_y = \begin{bmatrix} d & 0 & -C_x & 0 \\ 0 & 1 & 0 & 0 \\ C_x & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2/3} & 0 & -\frac{1}{\sqrt{3}} & 0 \\ 0 & 1 & 0 & 0 \\ 1/\sqrt{3} & 0 & \sqrt{2/3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow$$

5) Rotate about z:

$$R_z = \begin{bmatrix} \cos\delta & -\sin\delta & 0 & 0 \\ \sin\delta & \cos\delta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

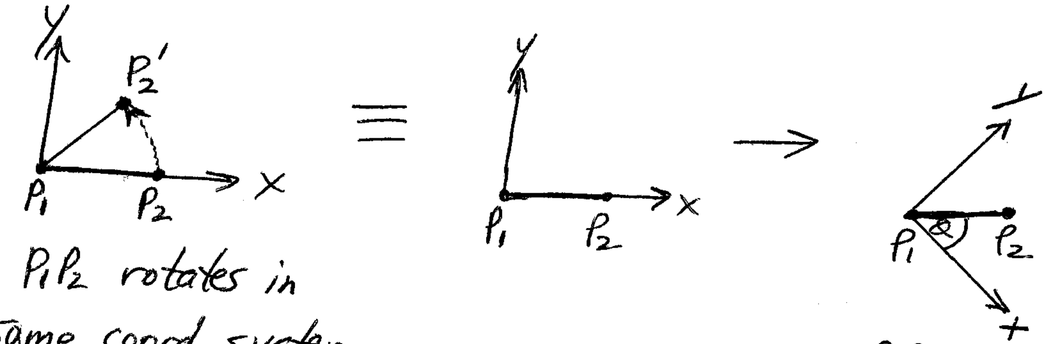
$$M = T^{-1} \cdot R_x^{-1} \cdot R_y^{-1} \cdot R_z \cdot R_y \cdot R_x \cdot T$$

Transform all object vertices:

$$[X'] = [M][X]$$

Coordinate System Transformation

Thus far, we have transformed object points in the same coordinate system.



P_1P_2 rotates in same coord system

P_1P_2 remains fixed different coord system

Change of coordinate system is useful when multiple objects, each defined in its local coordinate system, are combined and we wish to express these objects' coordinates in a single global coordinate system (known as the world coordinate system)

<u>Def:</u> $M_{i \leftarrow j}$ is a transformation matrix that converts the representation of a point in coordinate system j into its representation in coordinate system i.
$p^{(i)}$ is the representation of a point in coordinate system i

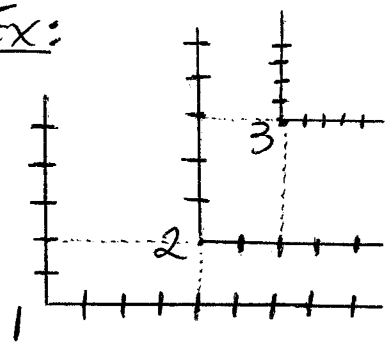
$$p^{(i)} = M_{i \leftarrow j}\, p^{(j)} \qquad p^{(j)} = M_{j \leftarrow k}\, p^{(k)}$$

$$p^{(i)} = M_{i \leftarrow j} \cdot \overbrace{M_{j \leftarrow k} \cdot p^{(k)}} = M_{i \leftarrow k}\, p^{(k)}$$

$$\therefore$$

$$M_{i \leftarrow k} = M_{i \leftarrow j} \cdot M_{j \leftarrow k}$$

<u>Ex:</u>



$$M_{1 \leftarrow 2} = T(4,2)$$

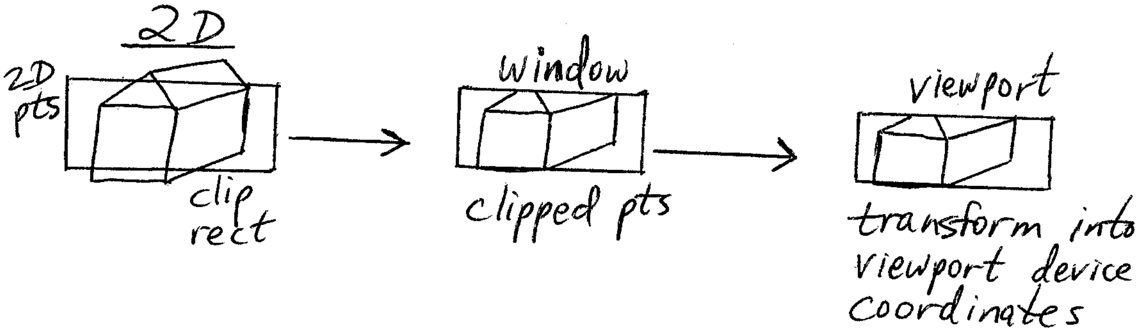
$$(0,0)^{(2)} \rightarrow (4,2)^{(1)}$$

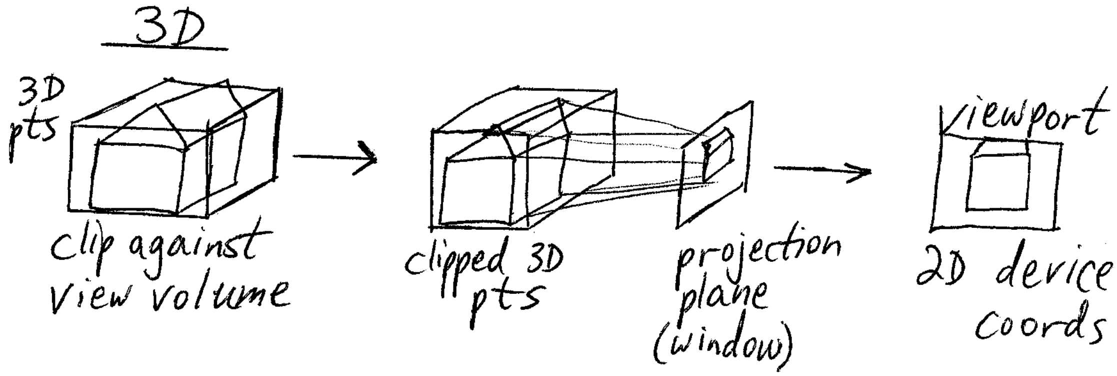
$$M_{2 \leftarrow 3} = T(2,3) \cdot S(.5,.5)$$

<u>Note</u>: first scale by $(.5,.5)$ and then translate by $(2,3)$. Scaling about origin leaves origin alone so that it can later be translated $(2,3)^{(2)}$

$$M_{1 \leftarrow 3} = M_{1 \leftarrow 2} \cdot M_{2 \leftarrow 3} = T(4,2) \cdot T(2,3) \cdot S(.5,.5)$$

Projections (3D to 2D)





Projections transform points in a coord system of dimension n into pts. in a coord system of dimension $< n$ (e.g. 3D to 2D)

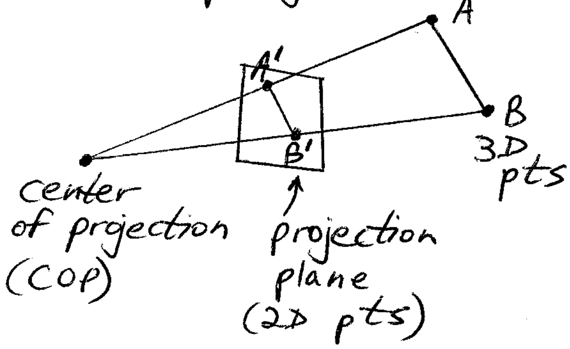
We will deal with planar geometric projections (PGP)

planar: onto a plane

geometric: uses straight projectors

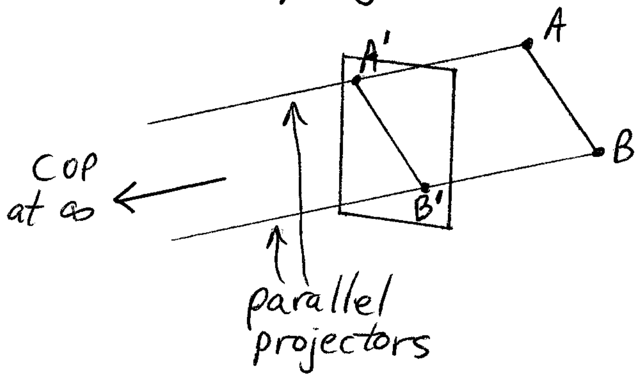
PGP can be divided into two basic classes: perspective and parallel.

Perspective projection



* Distance between COP and projection plane is <u>finite</u>

Parallel projection



* Distance between COP and projection plane is <u>infinite</u>

When defining a perspective projection, we must specify its center of proj.

When defining a parallel projection, we must specify its direction of proj.

Center of projection (COP) : $(x,y,z,1)$ ← 3D pt rep. with homogeneous coords

Direction of projection is a vector (difference between pts):

$$d = (x,y,z,1) - (x',y',z',1)$$
$$= (a,b,c,0)$$

$\hookrightarrow$ pt at ∞ in direction (a,b,c)

∴ direction and pts at ∞ correspond naturally

A perspective projection whose COP is a pt at ∞ → parallel projection

Hierarchy of projections

→ more complex

↓ more restrictive

Plane geometric projections

- parallel
  - orthographic
  - axonometric
    - trimetric
    - dimetric
    - isometric
  - oblique
    - cavalier
    - cabinet
- perspective
  - single pt.
  - 2-pt
  - 3-pt

PGP: intersection of lines (projectors) with proj. plane
Projectors are lines from an arbitrary pt. (COP) through each point in an object

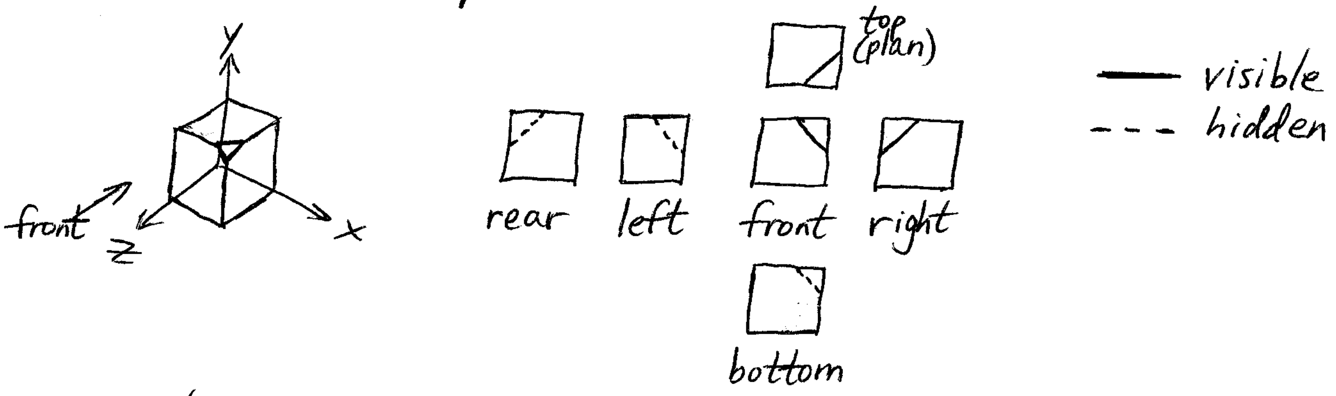
Perspective: COP is located at a finite pt. in 3-space
Parallel: COP is located at infinity ⇒ all projectors are parallel
Orthographic: projection onto one of the 3 coordinate planes ($x=0$, $y=0$, or $z=0$)

$$P_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad P_x = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad P_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

projection onto $z=0$ plane

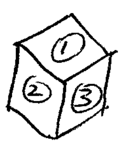
z drops out ($z=0$)

Multiple orthographic projections are needed to reconstruct shape of object:



Orthographic projections show "true" size and shape of a single object face if it is parallel to the coordinate plane (proj. plane)
Parallel lines and angles are preserved

Axonometric: rotate/translate object before ortho. proj. such that at least 3 adjacent faces are shown. This is equivalent to using a projection plane that is not normal to a principal axis. Again, true shape of face is not given unless it is parallel to the projection plane. However, the relative lengths of parallel lines remain constant (parallel lines are equally foreshortened). Angles are not preserved.



3 adjacent faces are shown

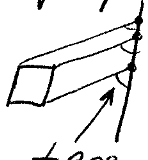
Trimetric: an axonometric projection whereby the foreshortening ratios for each projected principal axis (x,y,z) are all different.

Dimetric: a trimetric projection with 2 of the 3 foreshortening factors equal.

Isometric: a dimetric projection with all 3 principal axes equally foreshortened.
The principal axes project so as to make equal angles with one another.

8 possible directions satisfy this condition: angles $\pm 45°$, $\pm 35.26°$

Oblique: parallel projectors are not perpendicular to the plane of projection
Only faces parallel to projection plane are shown at their true size and shape (angles + lengths are preserved for these faces only, as in ortho. proj.)
Otherwise, size and shape are distorted

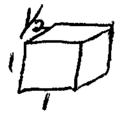


$\neq 90°$

Cavalier: angle between oblique projectors and proj. plane is $45°$

The foreshortening factors for all 3 dirs are equal.
The resulting figure appears too thick.

Cabinet: foreshortening factor for edges $\perp$ to plane of proj. $= \frac{1}{2}$



Used to correct deficiency of cavalier proj.

Perspective:

perspective transformation (not projection)

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow x' = \frac{x}{rz+1},\; y' = \frac{y}{rz+1},\; z' = \frac{z}{rz+1}$$

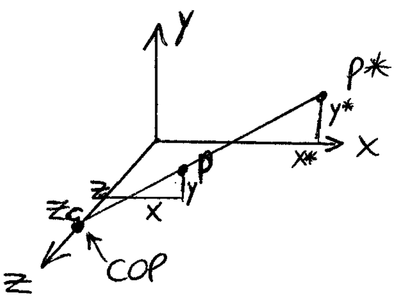
no longer $[0\,0\,0\,1]$

perspective foreshortening: the size of the projection varies inversely with the distance of the object from COP

Parallel lines converge

A perspective projection is obtained by concatenating an orthographic proj. w/ persp. transformation:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{P_z} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \begin{aligned} x' &= \frac{x}{rz+1} \\ y' &= \frac{y}{rz+1} \\ z' &= 0 \end{aligned}$$



similar triangles:

$$\frac{x^*}{z_c} = \frac{x}{z_c - z} \Rightarrow \boxed{x^* = \frac{x}{1 - \frac{z}{z_c}}}$$

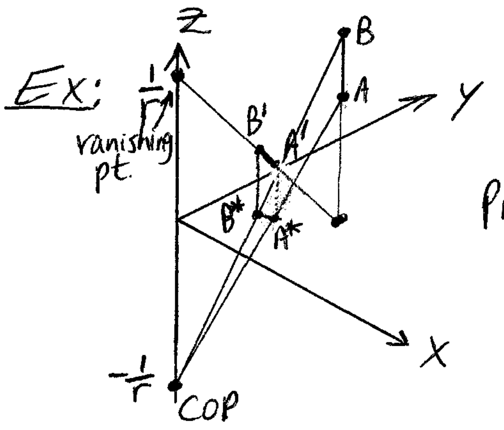
$$\frac{y^*}{\sqrt{x^{*2} + z_c^2}} = \frac{y}{\sqrt{x^2 + (z_c - z)^2}} \Rightarrow \boxed{y^* = \frac{y}{1 - \frac{z}{z_c}}}$$

SAME

Let $r = -\frac{1}{z_c}$ in matrix above $\Rightarrow \boxed{x' = \frac{x}{1-\frac{z}{z_c}} \quad y' = \frac{y}{1-\frac{z}{z_c}}}$

Interpret COP to be $-\frac{1}{r}$

Notice, as $r \to 0$, COP $\to \infty$, parallel projection results.
Also, pts on plane of proj ($z=0$) are not distorted (r.t.s.)



initial line AB $\longrightarrow$ persp. transf. A'B' $\longrightarrow$ persp. proj. A*B*

Proj. plane $z=0$

Observations

1) A'B' intersects $z=0$ plane at same pt as AB
2) A'B' intersects z-axis at $z=\frac{1}{r}$
3) Perspective transformation has transformed the intersection pt. at ∞ of AB and z-axis to finite pt. $z=\frac{1}{r}$

   This pt. is called the <u>vanishing pt.</u> (parallel lines converge to it)
4) Vanishing pt. lies equal distance on the opposite side of proj. plane from COP

Confirmation:

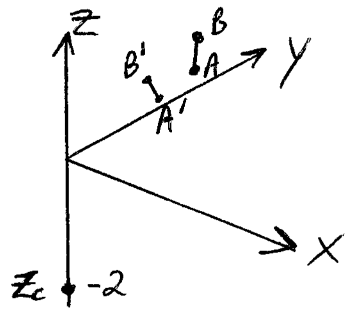
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}}_{\text{pt. at } \infty \text{ on } +z\text{-axis}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ r \end{bmatrix}}_{\text{finite pt. on } +z\text{-axis}} \Rightarrow \begin{array}{l} x'=0 \\ y'=0 \\ z'=\frac{1}{r} \end{array} \text{(vanishing pt.)}$$

1 v.p. $\leftarrow$ 1-pt perspective: last row has only 2 0's

2 v.p. $\leftarrow$ 2-pt perspective: last row has only 1 0

3 v.p. $\leftarrow$ 3-pt perspective: last row has no zeros

Ex:



$A = \begin{bmatrix} 3 \\ 2 \\ 4 \\ 1 \end{bmatrix}$ $B = \begin{bmatrix} 3 \\ 2 \\ 8 \\ 1 \end{bmatrix}$

Perform: persp. proj onto $z=0$ plane from a COP at $z_c = -2$

$r = -\frac{1}{z_c} = .5$

$$\begin{array}{cc} A' & B' \end{array} \qquad\qquad\qquad\qquad\qquad\qquad \begin{array}{cc} A & B \end{array}$$
$$\begin{bmatrix} 1 & .6 \\ .667 & .4 \\ 1.333 & 1.6 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 2 & 2 \\ 4 & 8 \\ 3 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & .5 & 1 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 2 & 2 \\ 4 & 8 \\ 1 & 1 \end{bmatrix}$$

3D pts after persp. transformation

3D pts before persp. transf.

Parametric equation of $A'B'$:

$$P(t) = A' + (B' - A')t \qquad 0 \le t \le 1$$

$$= \begin{bmatrix} 1 \\ .667 \\ 1.333 \\ 1 \end{bmatrix} + \begin{bmatrix} -.4 \\ -.267 \\ .267 \\ 0 \end{bmatrix} t$$

Intersection of $A'B'$ with the $x=0$, $y=0$, and $z=0$ planes yields:

$x(t) = 0 = 1 - .4t \Rightarrow t = 2.5$

$y(t) = 0 = .667 - .267t \Rightarrow t = 2.5$

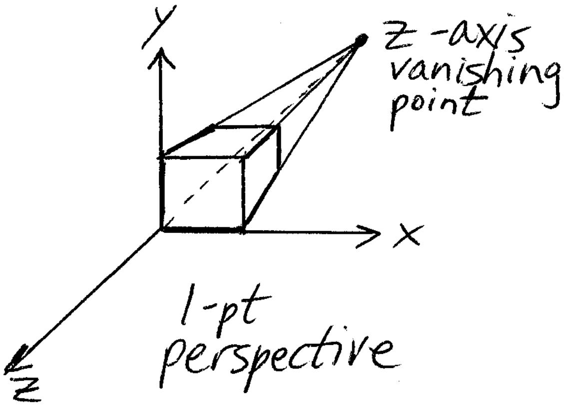
$z(t) = 0 = 1.333 + .267t \Rightarrow t = -5$

Vanishing pt: $z(2.5) = 1.333 + (.267)(2.5) = 2.0 \Rightarrow$ intersection of $A'B'$ with z-axis at $z = +\frac{1}{r}$ (opposite COP)

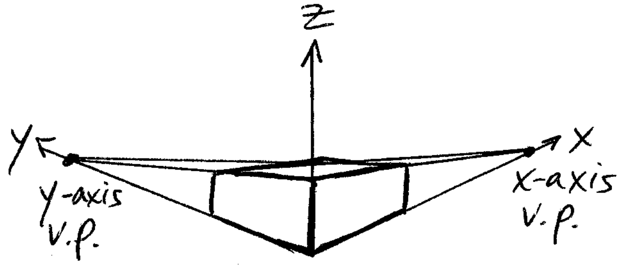
Intersection of $A'B'$ with $z=0$ plane: $x(-5) = 1 - (.4)(-5) = 3$, $y(-5) = .667 - (.267)(-5) = 2$ $\Rightarrow$ Same as intersection of AB with $z=0$ plane

Proj. of $A'B'$ onto $z=0$ plane:

$$\begin{array}{cc} A^* & B^* \end{array} \qquad\qquad\qquad\qquad\qquad \begin{array}{cc} A' & B' \end{array}$$
$$\begin{bmatrix} 1 & .6 \\ .667 & .4 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & .6 \\ .667 & .4 \\ 1.333 & 1.6 \\ 1 & 1 \end{bmatrix}$$



1-pt
perspective



2-pt
perspective

lines parallel to x and y axes do not converge; only lines parallel to z-axis do so.

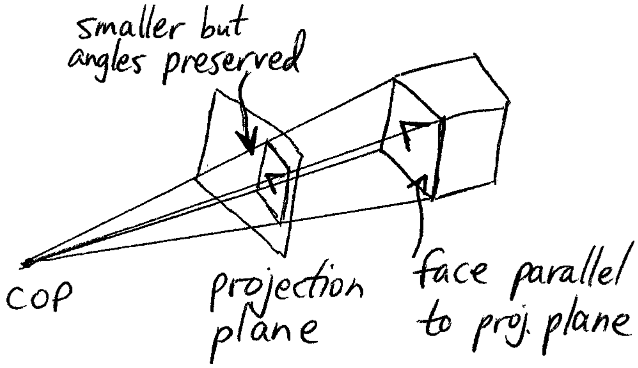
lines parallel to x- and y-axes converge

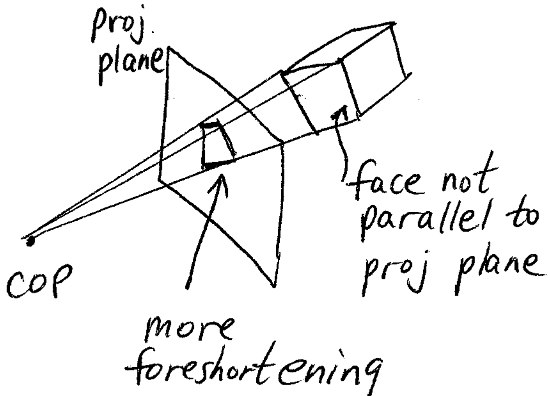
3-pt perspective is used less since it adds little realism beyond 2-pt perspective.

Properties of perspective projections:

angles are preserved only on those faces of the object that are parallel to the projection plane.

Parallel lines do not in general project to parallel lines.

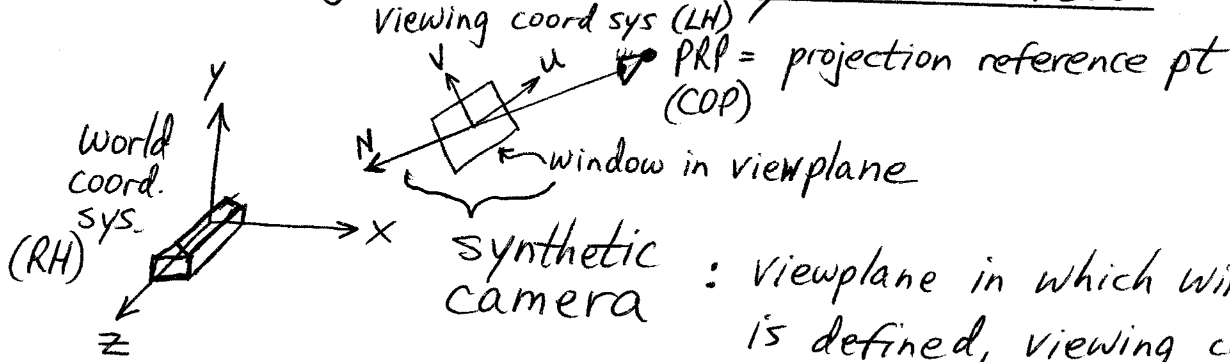




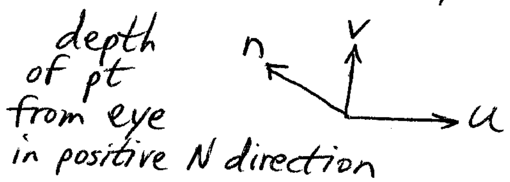
(unequal because some object parts are closer to proj. plane than others)

Angles are not preserved

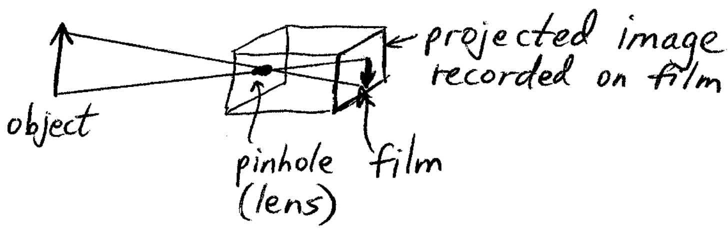
Specifying an Arbitrary 3D View



synthetic camera : Viewplane in which window is defined, viewing coord system (LH sys is more natural), and eye



Pin-hole (real) camera:

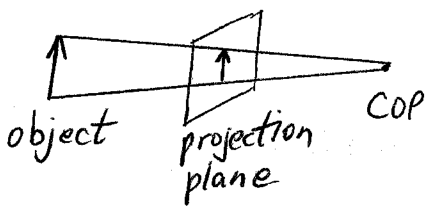


Note: Pinhole is COP and it lies between object and projection plane (film).

$\Downarrow$

Image is upside down

Synthetic camera:



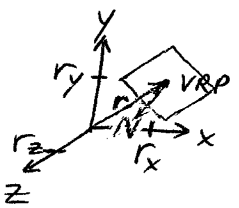
Note: projection plane lies between object and COP.

$\Downarrow$

Image is upright

Mathematical description of synthetic camera: specify viewing reference coord (VRC) system, position + orientation of viewplane, window borders, and position of eye (PRP).

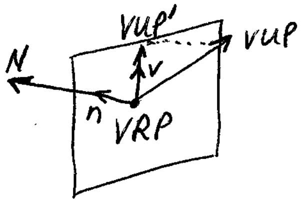
1) View plane (≡ projection plane)



a) define view reference point (VRP) to be origin of coord sys
It is at (r_x, r_y, r_z) in WC (world coords)

b) define viewplane normal (VPN); given as a unit vector n with WC components (n_x, n_y, n_z). VPN establishes n-axis

2) Viewing coordinate system



a) define VUP vector to designate view "up" direction. This points in the general direction but is not guaranteed to be ⊥ to VPN (N)

b) project VUP onto VP along N. This continues to point up but also ⊥ to n. Call this projected line (from VRP to VUP') the <u>V-axis</u>
$VUP' = VUP - \underbrace{(VUP \cdot n)}_{\text{scalar}} n$; $v = \dfrac{VUP'}{|VUP'|}$
Fails if VUP ∥ N (up cannot be along N)

c) Make <u>u-axis</u> ⊥ n and v: $u = n \times v$ ← forms LH sys
(↑ cross product)

3) Window borders : (U_{min}, V_{min}) to (U_{max}, V_{max}) in VRC system. VRP is not, in general, the center of the window. The window borders specify what part of the view plane is to be displayed.

4) Eye position (e_u, e_v, e_n) is given in VRC sys. It can be anywhere but usually on -n-axis. Otherwise, the view would be oblique

Flexibility of Camera Model

Fly by: move VRP (position of viewer). Akin to moving head without changing direction of looking

Look around: move VPN. Akin to swiveling/pivoting head

Head tilting: change VUP. Akin to keeping eye fixed on object while tilting head. Appears rotated.

Hints: VRP is usually near center of object to be viewed. Normalize VPN : $n = \frac{norm}{|norm|}$
Have VPN point to WC origin or point of interest.

Transform Object Points into VRC

Represent all points in VRC for consistency with eye position and window boundaries already given in VRC.

Find: viewing coords (a,b,c) of WC pt. $p(x,y,z)$.

$$\overbrace{(x,y,z)}^{p} = M_{VW}\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} + \underbrace{\begin{bmatrix} r_x \\ r_y \\ r_z \\ 0 \end{bmatrix}}_{r} \quad \text{where } M_{VW} = \begin{pmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} = M_{VW}^{-1}(p-r) = M_{VW}^{T}p \underbrace{- M_{VW}^{T}r}_{\text{translation } r'}$$

$r' = (-r \cdot u, -r \cdot v, -r \cdot n)$

$$M_{WV} = \begin{pmatrix} u_x & u_y & u_z & r'_x \\ v_x & v_y & v_z & r'_y \\ n_x & n_y & n_z & r'_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\uparrow$ WC to VRC

Ex: $u = (-1,0,0)$ $\quad v = (0, 4/5, 3/5)$ $\quad n = (0, -\frac{3}{5}, \frac{4}{5})$

$-r \cdot u = 2$ $\quad -r \cdot v = -\frac{9}{5}$ $\quad -r \cdot n = \frac{13}{5}$ $\quad r = (2,3,-1)$

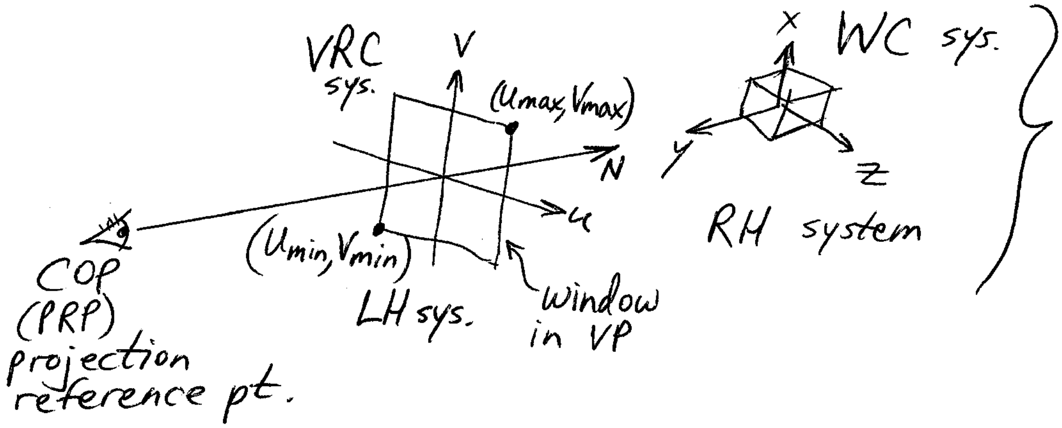
$$M_{WV} = \begin{pmatrix} -1 & 0 & 0 & 2 \\ 0 & 4/5 & 3/5 & -9/5 \\ 0 & -3/5 & 4/5 & 13/5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{bmatrix} 4 \\ 7 \\ 2 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} -2 \\ 5 \\ 0 \\ 1 \end{bmatrix}$$

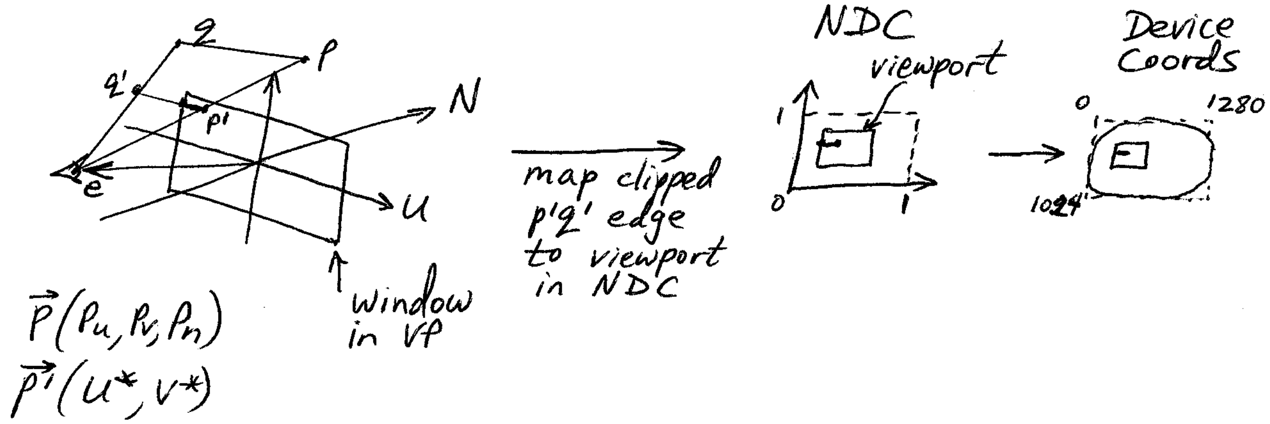
$\rightarrow$ on viewplane

Purpose: Convert from WC to VRC to simplify remaining operations of clipping, projection, and rendering pictures of a scene.

```
for all vertices {
    UVNvert[i] = worldToView(vert[i], u, v, n, r);
}
```



Convert everything to VRC



Ray from eye at $e = (e_u, e_v, e_n)$ to p is:

$$\vec{r}(t) = \vec{e}(1-t) + \vec{p}t$$

It pierces VP at t' when n-component $= 0$:

$$e_n(1-t') + p_n t' = 0 \Rightarrow t' = \frac{e_n}{e_n - p_n}$$

Note: this fails if $e_n = p_n$ (if $\vec{r} \perp N$)

only if we look parallel to VP

Usual case: $e_n < 0$, $p_n > e_n \Rightarrow 0 < t' < 1$

Plugging t' into $\vec{r}(t)$ yields the u^*, v^* components:

General soln. (COP may be anywhere)

$$u^* = e_u(1-t') + p_u t' = e_u\left(\frac{-p_n}{e_n - p_n}\right) + p_u\left(\frac{e_n}{e_n - p_n}\right)$$

$$= \frac{e_n p_u - e_u p_n}{e_n - p_n}$$

$$v^* = e_v(1-t') + p_v t' = e_v\left(\frac{-p_n}{e_n - p_n}\right) + p_v\left(\frac{e_n}{e_n - p_n}\right)$$

$$= \frac{e_n p_v - e_v p_n}{e_n - p_n}$$

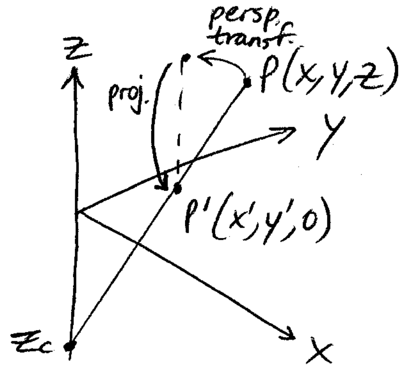
Special case: eye on N-axis (usual case)

$e_u = e_v = 0$

$$u^* = \frac{e_n p_u - \cancel{e_u p_n}^{\,0}}{e_n - p_n} = \frac{e_n p_u}{e_n - p_n} = \frac{p_u}{1 - \frac{p_n}{e_n}}$$

$$v^* = \frac{e_n p_v - e_v p_n}{e_n - p_n} = \frac{e_n p_v}{e_n - p_n} = \frac{p_v}{1 - \frac{p_n}{e_n}}$$

Fails if $e_n = p_n$ ($r \perp N$) or $e_n = 0$ (eye on VP)



Let $r = \frac{-1}{z_c}$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & r & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\Downarrow$$

$$x' = \frac{x}{1+rz} = \frac{x}{1 - \frac{z}{z_c}}$$

$$y' = \frac{y}{1+rz} = \frac{y}{1 - \frac{z}{z_c}}$$

$e_n < 0$

SAME RESULT

$z_c < 0$

Foreshortening factor is $\frac{1}{1 - \frac{p_n}{e_n}}$ $\Rightarrow$ No foreshortening if $p_n = 0$ (on VP)

since $e_n < 0$, $p_n > 0$, $\left(\frac{-p_n}{e_n}\right) > 0$

$\underbrace{\text{As } p_n \uparrow \text{ foresh. factor} \downarrow}_{\text{inverse relationship betw. distance + size}}$

We can rewrite above matrix as:

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{e_n} & 1 \end{bmatrix}$$

perspective transformation

Back to General Case: eye located off N-axis

Numerators of u^*, v^* are linear combinations of p:

$$p^* = M_p M_s \begin{bmatrix} p_u \\ p_v \\ p_n \\ 1 \end{bmatrix}$$

where $M_s = \begin{bmatrix} 1 & 0 & -e_u/e_n & 0 \\ 0 & 1 & -e_v/e_n & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

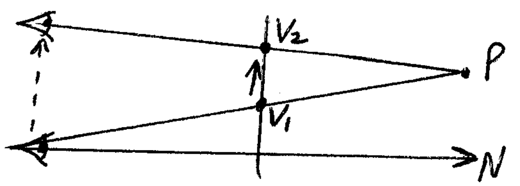
$\uparrow$ shear transf.

Verify:

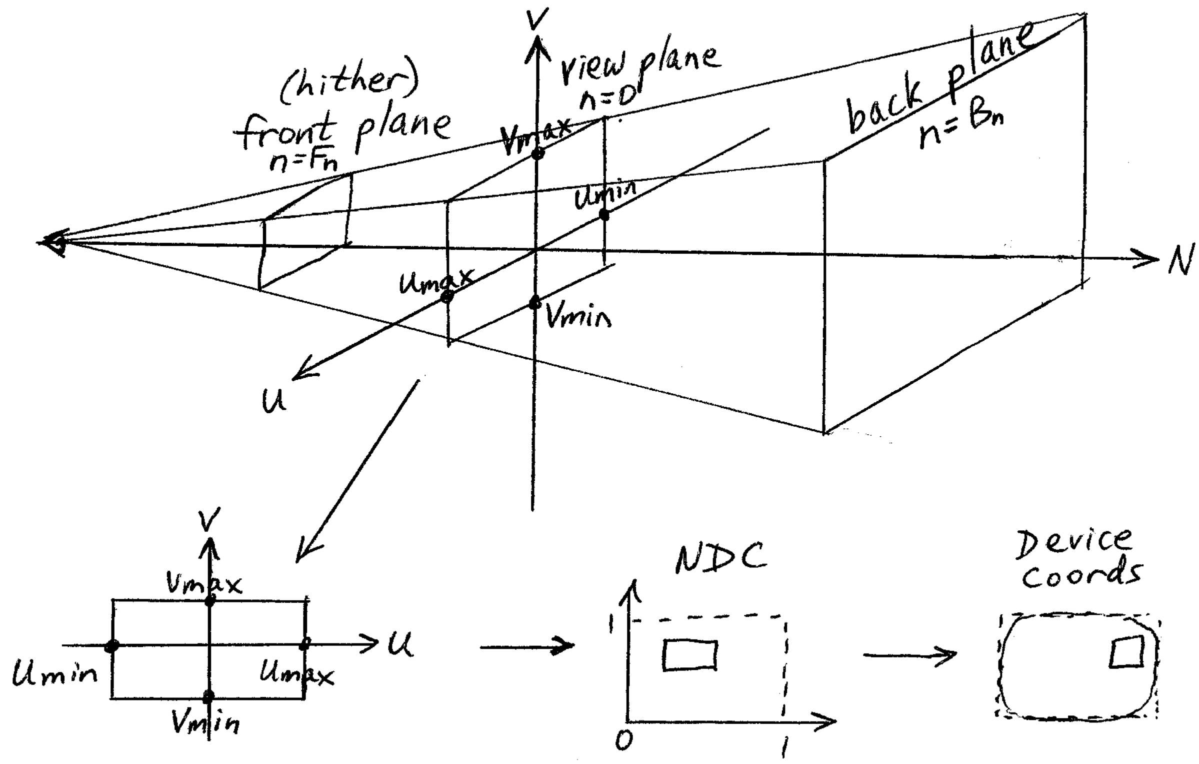
$$\begin{bmatrix} e_n p_u - e_u p_n \\ e_n p_v - e_v p_n \\ e_n p_n \\ e_n \end{bmatrix} = \begin{bmatrix} p_u - \frac{e_u p_n}{e_n} \\ p_v - \frac{e_v p_n}{e_n} \\ p_n \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\frac{e_u}{e_n} & 0 \\ 0 & 1 & -\frac{e_v}{e_n} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_u \\ p_v \\ p_n \\ 1 \end{bmatrix}$$

$$u^* \stackrel{\checkmark}{=} \frac{e_n p_u - e_u p_n}{e_n - p_n} \qquad v^* \stackrel{\checkmark}{=} \frac{e_n p_v - e_v p_n}{e_n - p_n} \Longleftarrow \begin{bmatrix} e_n p_u - e_u p_n \\ e_n p_v - e_v p_n \\ e_n p_n \\ e_n - p_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{e_n} & 1 \end{bmatrix} \begin{bmatrix} e_n p_u - e_u p_n \\ e_n p_v - e_v p_n \\ e_n p_n \\ e_n \end{bmatrix}$$

$\therefore$ effect of moving eye off N-axis is to shear the image. $+e_v$ shifts projected pt upward



View Volume



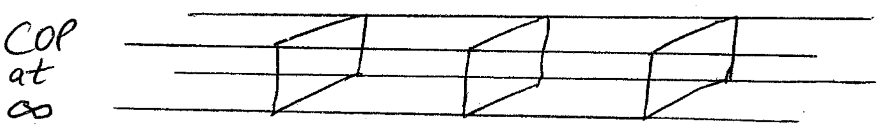
View volume is region in space that is to be projected and drawn. It is part of the synthetic camera $\Rightarrow$ defined in VRC

F and B planes are parallel to viewplane.

They chop pyramid into frustum.

Lines are clipped to view volume. $e_n < F_n < B_n$

For parallel projection, view volume becomes parallelepiped:



Summary Thus Far:

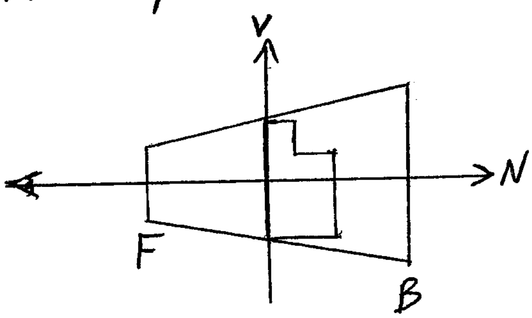
WC → VC → shear → perspective

$$\Downarrow$$

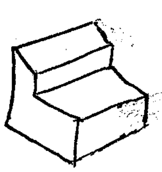
$$\begin{bmatrix} P_u \\ P_v \\ P_n \\ P_w \end{bmatrix} = M_p M_s M_{wv} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

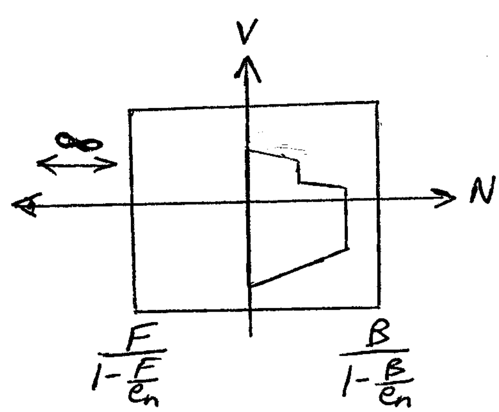
$$\begin{bmatrix} u^* \\ v^* \\ n^* \end{bmatrix} = \begin{bmatrix} P_u/P_w \\ P_v/P_w \\ P_n/P_w \end{bmatrix} \quad \text{and} \quad P_w = 1 - \frac{P_n}{e_n} = \frac{e_n - P_n}{e_n}$$

The effect of $M_p M_s$ is called prewarping.
This moves each vertex of the object in 3 dimensions such that its (u,v) coordinates are in their final position.

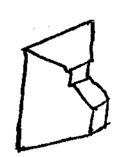


Before prewarping





After prewarping



The front of the block is on the viewplane so its projection remains the same (no distortion).

The rear appears smaller.

Once objects have been prewarped, we perform orthographic projection. This sets the n-component to 0.

Perspective projection consists of:

1) perspective transformation
2) recover Cartesian coordinates (divide by w)
3) orthographic projection

Notice that prewarping transforms view volume from frustum to parallelepiped, a much simpler shape!

Note that:

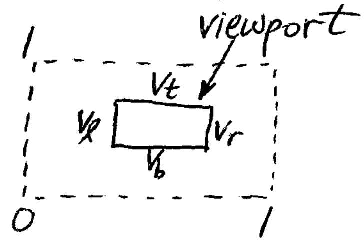
1) prewarping preserves planes (F, VP, B)
2) planes parallel to VP are just shifted to $\frac{F}{1-\frac{F}{e_n}}$ and $\frac{B}{1-\frac{B}{e_n}}$
3) the 4 side walls of the view volume are warped into 4 parallel planes: $u=u_{min}$, $u=u_{max}$, $v=v_{min}$, $v=v_{max}$

Normalized (Canonical) View Volume

A view volume is used to map coordinates to viewport, given in NDC.
Normalizing it helps simplify clipping.

$$M_N = \begin{pmatrix} S_u & 0 & 0 & r_u \\ 0 & S_v & 0 & r_v \\ 0 & 0 & S_n & r_n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

normalization matrix
(VRC → NDC)



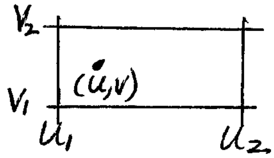
where

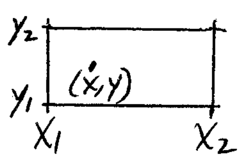
$$S_u = \frac{V_r - V_l}{U_{max} - U_{min}} \qquad r_u = \frac{V_l U_{max} - V_r U_{min}}{U_{max} - U_{min}}$$

$$S_v = \frac{V_t - V_b}{V_{max} - V_{min}} \qquad r_v = \frac{V_b V_{max} - V_t V_{min}}{V_{max} - V_{min}}$$

$$S_n = \frac{(e_n - B)(e_n - F)}{e_n^2 (B - F)} \qquad r_n = \frac{F(e_n - B)}{e_n(F - B)}$$

Recall:





$$x = S_u u + t_u$$
$$y = S_v V + t_v$$

where

$$S_u = \frac{x_2 - x_1}{u_2 - u_1} \qquad S_v = \frac{y_2 - y_1}{V_2 - V_1}$$

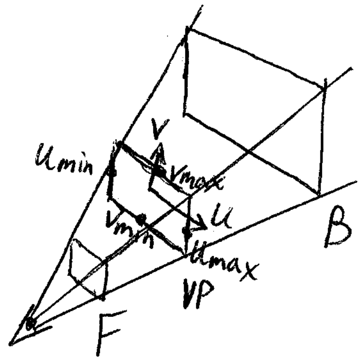
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} u_1 & 1 \\ u_2 & 1 \end{bmatrix}}_{M} \begin{bmatrix} S_u \\ t_u \end{bmatrix} \rightarrow \quad t_x = \frac{x_1 u_2 - u_1 x_2}{u_2 - u_1} \qquad t_y = \frac{y_1 V_2 - V_1 y_2}{V_2 - V_1}$$

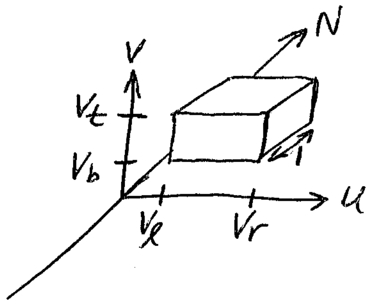
$$M^{-1} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{M^{-1} M}_{I} \begin{bmatrix} S_u \\ t_u \end{bmatrix}$$

$$\frac{1}{u_1 - u_2} \begin{bmatrix} 1 & -1 \\ -u_2 & u_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} S_u \\ t_u \end{bmatrix}$$

Note: If $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

then $M^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$





Composite matrix M_c

$$M_C = M_N M_P M_S M_{WV}$$

$$P(u,v) = M_C P(x,y,z)$$

transforms pts from WC to NDC

Clipping

Clipping to a normalized view volume is now easy. Do clipping in homogeneous coords to avoid division on clipped points.
Use 3D Liang-Barsky algorithm.