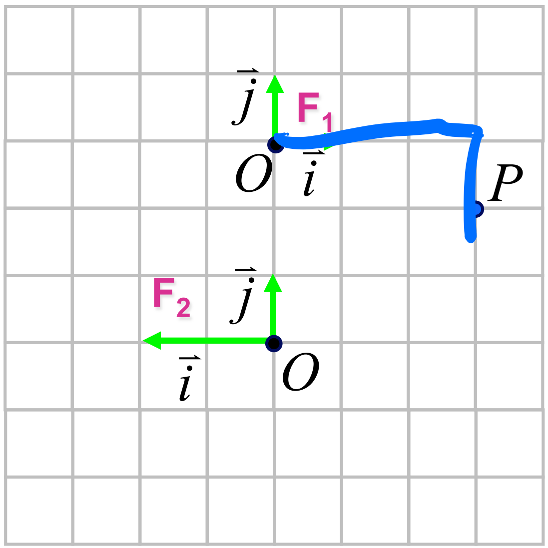
Transformations as a change of basis



$P_1 = (3,-1)$ $P_2 = (-1.5, 2)$ **Goal:** $P_2 = M P_1$

$$P = O + x\vec{i} + y\vec{j}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}_1 + x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix}_1 + y_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}_1$$

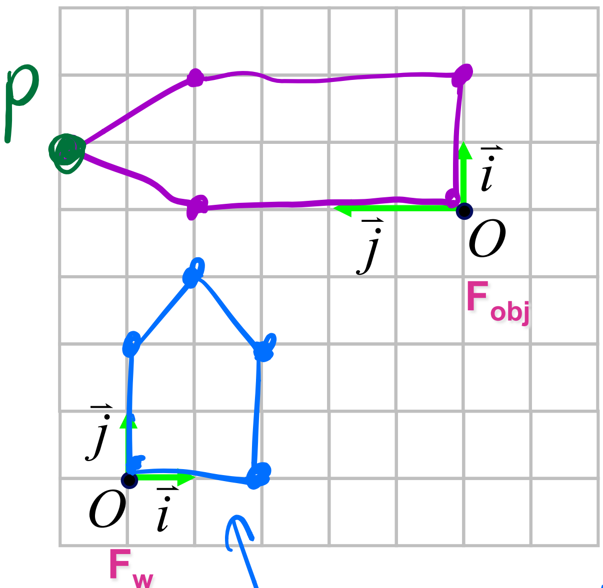
$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}_2 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}_2 + x_1 \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}_2 + y_1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}_2$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

check:

$$\begin{bmatrix} -1.5 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -1 \\ 1 \end{bmatrix}$$

Transformations as a change of basis



Goal: $P_w = M \, P_{obj}$

$$
\begin{bmatrix} -1 \\ 5 \\ 1 \end{bmatrix} = \begin{array}{ccc} i_{obj} & j_{obj} & O_{obj} \\ \end{array} \begin{bmatrix} 0 & -2 & 5 \\ 1 & 0 & 4 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}}_{P_{obj}}
$$

$$
\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}
$$

untransformed house, i.e., $F_w = F_h$

$$P_{obj} = O_{obj} + x_{obj} \, i_{obj} + y_{obj} \, j_{obj}$$

3D Transformations

Affine transformations

- linear transformation + translations
- can be expressed as a 3x3 matrix + 3 vector

$$P' = M \cdot P + T$$

4x4 matrices

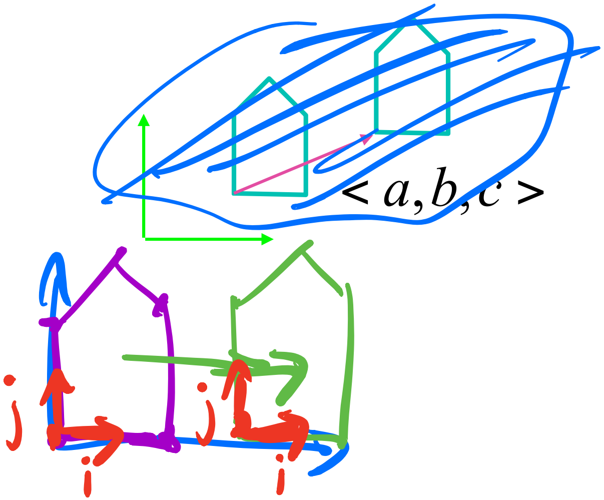
$$\begin{array}{c} \\ \\ \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \end{array} = \begin{array}{c} \begin{array}{cccc} i & j & k & O \end{array} \\ \begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{array}{c} \\ \\ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{array}$$

h=1

$$P = 1O + x\vec{i} + y\vec{j} + z\vec{k}$$

Transformations

Translation



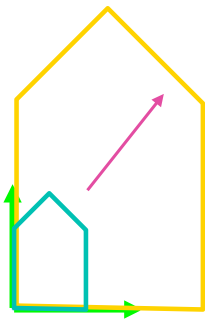
Translate(a,b,c)
Trans(a,b,c)

$$x' = x + a$$
$$y' = y + b$$
$$z' = z + c$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformations

Scaling



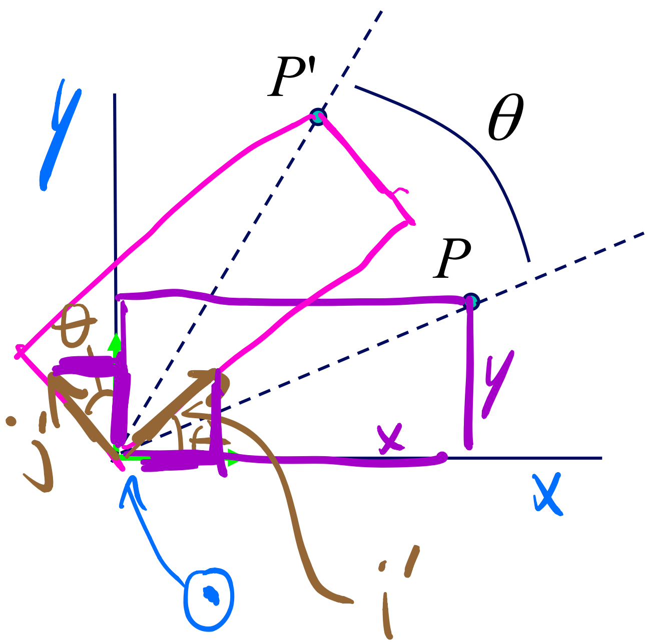
Scale(a,b,c)

$$x' = a x$$
$$y' = b y$$
$$z' = c z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Transformations

Rotation



$Rotate(z,\theta)$

$Rot(z,\theta)$

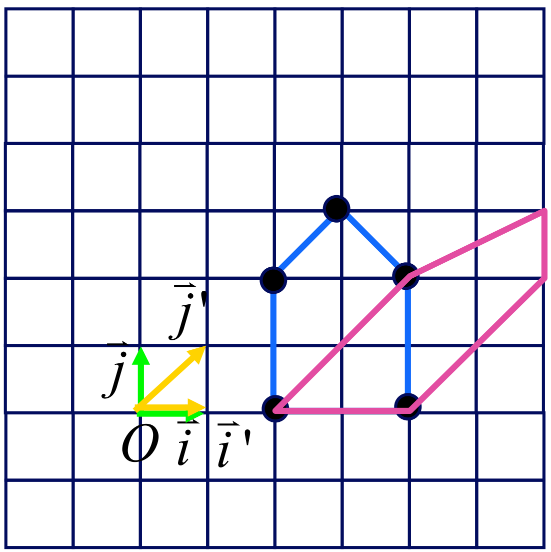
CCW is +ve

$$\begin{bmatrix} i' & j' & k' & o' \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

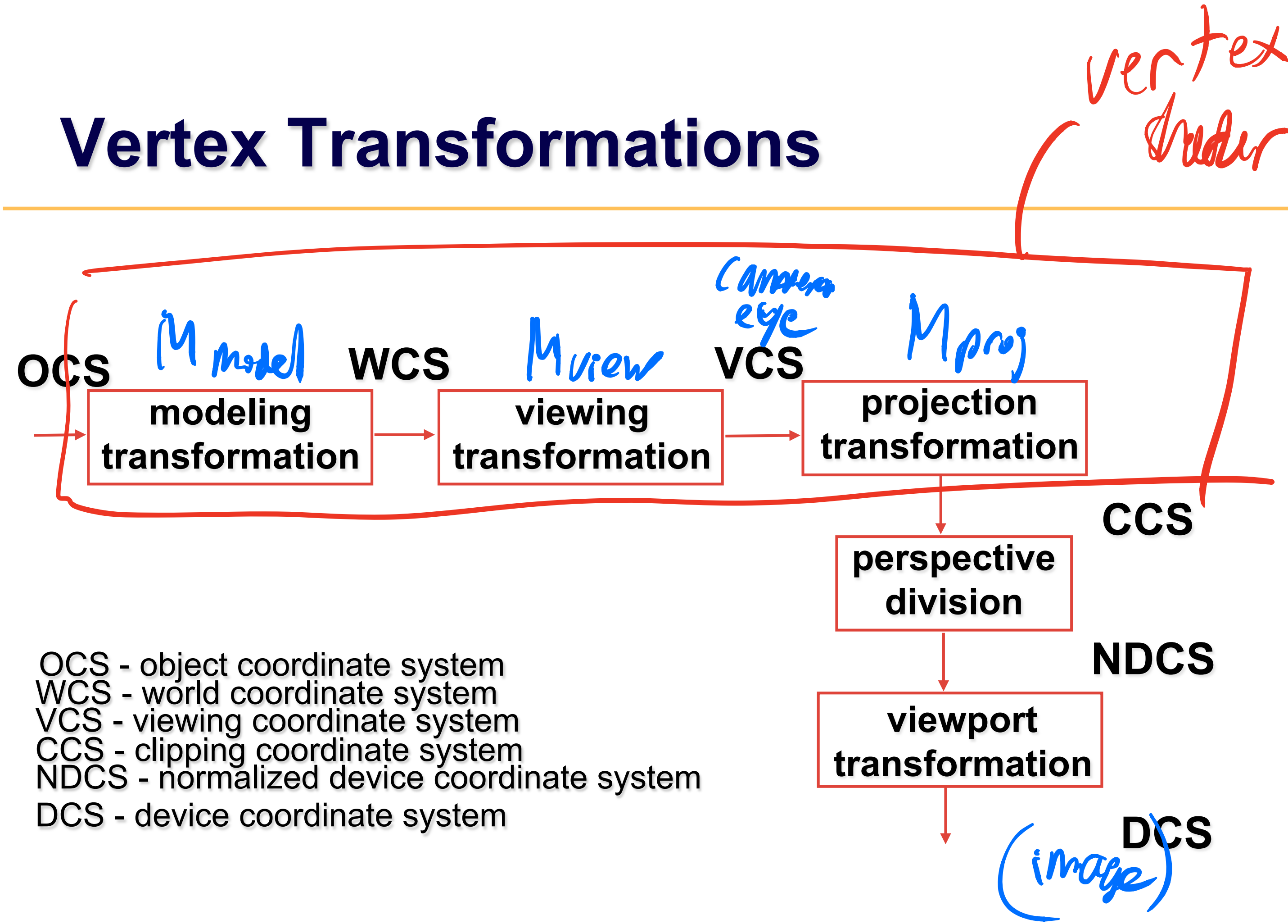
Transformations

Shear



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

Vertex Transformations



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

Composition of Transformations

reminder:

translate(a,b,c)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

scale(a,b,c)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$Rotate(z,\theta)$$

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

or build 4x4 matrix directly

$$\begin{bmatrix} i & j & k & O \\ 0 & 0 & 0 & 1 \end{bmatrix} \Big\} \text{ i, j, k, O of obj expressed wrt world.}$$

Simple Compositions

translate(a,b,c) translate(d,e,f) $= \text{translate}(a+d, b+e, c+f)$

$$\begin{bmatrix} 1 & & & a \\ & 1 & & b \\ & & 1 & c \\ & & & 1 \end{bmatrix}\begin{bmatrix} 1 & & & d \\ & 1 & & e \\ & & 1 & f \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & a+d \\ & 1 & & b+e \\ & & 1 & c+f \\ & & & 1 \end{bmatrix}$$

scale(a,b,c) scale(d,e,f)

$$\begin{bmatrix} a & & & \\ & b & & \\ & & c & \\ & & & 1 \end{bmatrix}\begin{bmatrix} d & & & \\ & e & & \\ & & f & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} ad & & & \\ & be & & \\ & & cf & \\ & & & 1 \end{bmatrix} = \text{scale}(ad, be, cf)$$

$Rotate(z,\theta_1)\ Rotate(z,\theta_2)$

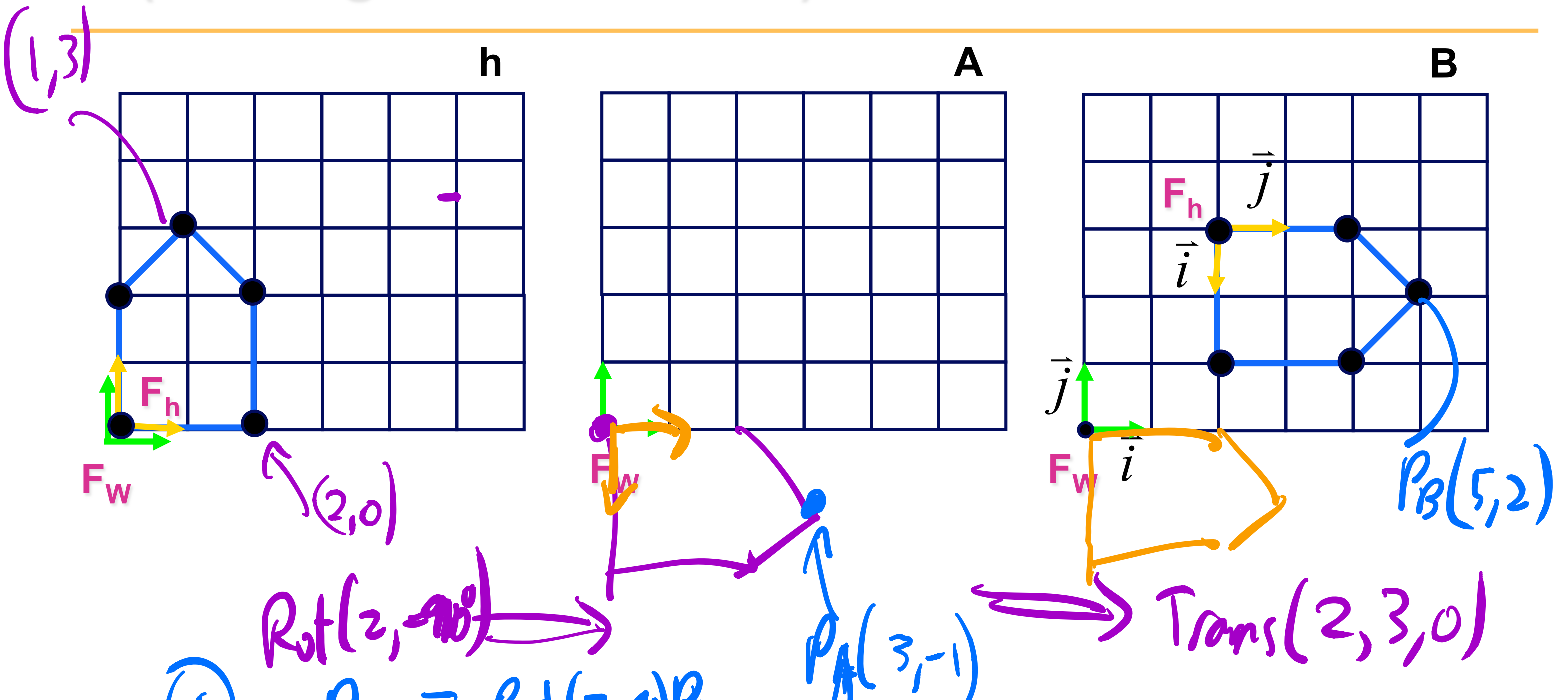
$$\begin{bmatrix} c_1 & -s_1 & & \\ s_1 & c_1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}\begin{bmatrix} c_2 & -s_2 & & \\ s_2 & c_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} = \begin{bmatrix} c_1c_2 - s_1s_2 & -s_1c_2 - c_1s_2 & & \\ s_1c_2 + c_1s_2 & c_1c_2 - s_1s_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$c_{12} = \cos(\theta_1 + \theta_2)$

$s_{12} = \sin(\theta_1 + \theta_2)$

Composing Transformations

(thinking in fixed coords)

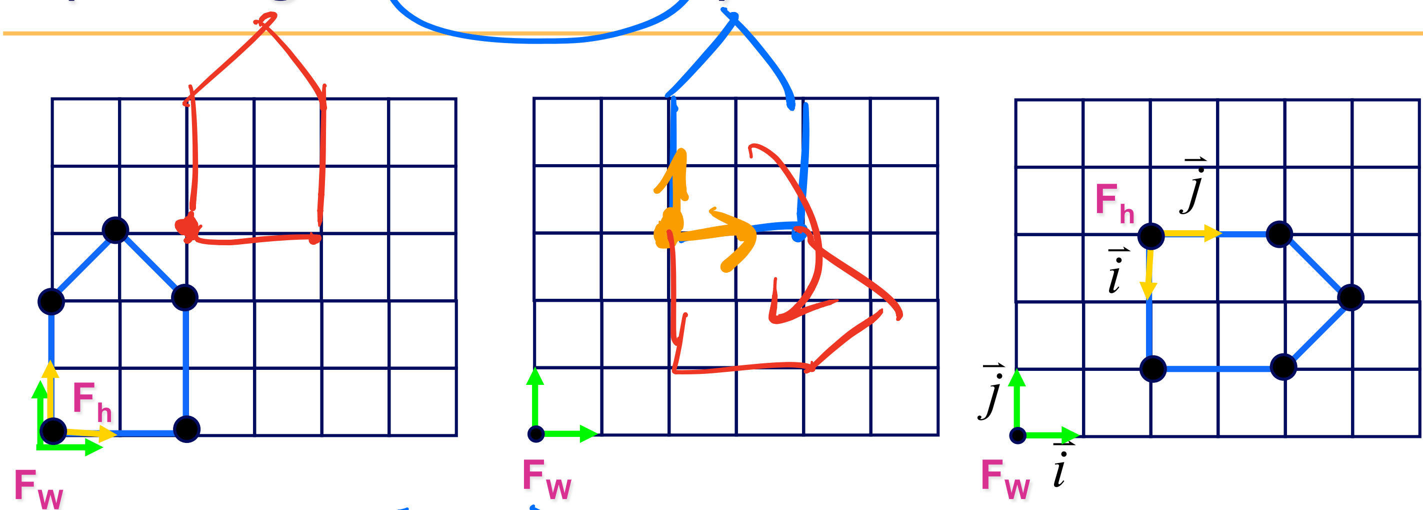


① $P_A = Rot(z,-90)\,P_h$ ✓

② $P_B = Trans(2,3,0)\,P_A$

$P_B = Trans(2,3,0)\,Rot(z,-90°)\,P_h$

Composing Transformations
(thinking in local coords)



Trans(2,3,0)

Rot(z,-90°)

$P_w = \text{Trans}(2,3,0)\ \text{Rot}(z,-90^\circ)\ P_h$

right multiply

Composing Transformations

(a) $P_w = Trans(2,3,0)\, Rot(z,-90°)\, P_h$ prev two slides

or (b) $P_w = Rot(z,-90°)\, Trans(-3,2,0)\, P_h$

- left multiply: R-to-L
  - *interpret operations wrt fixed coords*
- right multiply: L-to-R (default for **code**)
  - *interpret operations wrt local coords*

(a)

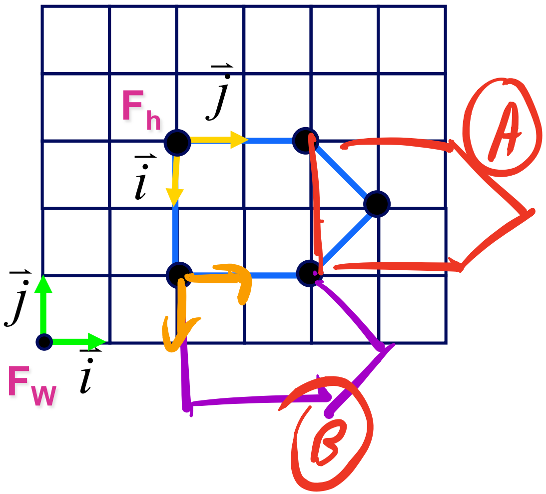
M = I // M

M.translate(2,3,0); // $M \leftarrow M \cdot T$

M.rotate(z,-90°) // $M \leftarrow M \cdot R$

$M = T \cdot R$

Summary Example



$T = Trans(2,0,0)$

$$P' = M\,P$$

left multiply (fixed) A

$P' = T \cdot M \cdot P$

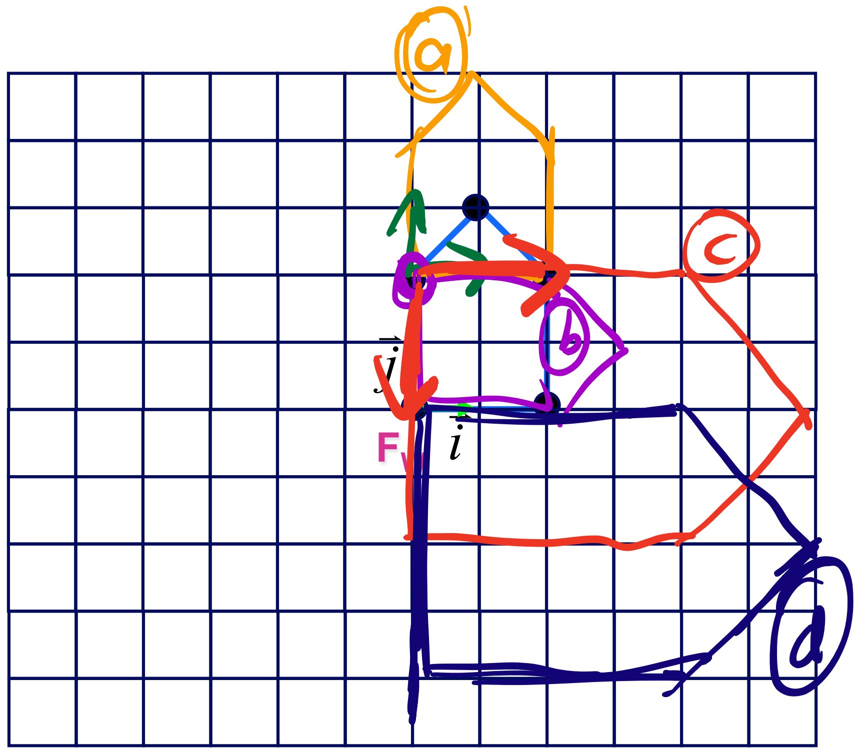
right multiply (local) B

$P' = M \cdot T \cdot P$

$P' = MTP$

Test yourself ...

assume graphics API right multiplies. (local)



a Translate(0,2,0);
b Rotate(z,-90);
c Scale(2,2,2);
d Translate(1,0,0);
DrawHouse();

$$M = \underbrace{M_a M_b M_c M_d}$$

$$P_w = M P_h$$

Note: do non-uniform scales always as a last step.

Test yourself

$$M = \begin{array}{c} \begin{array}{cccc} i & j & & O \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 1 \\ 1 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

(a) Sketch the origin and basis vectors of the transformed house

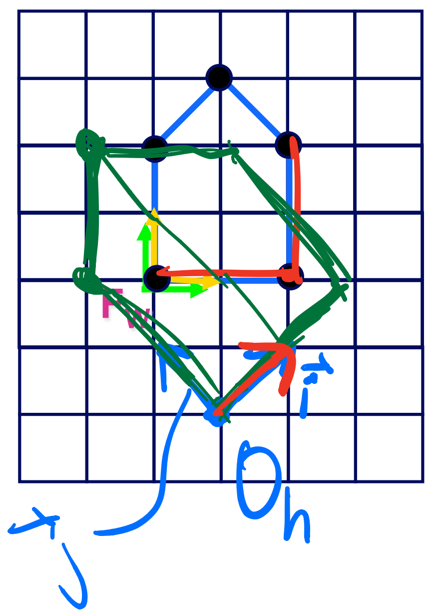
(b) Draw the transformed house

(c) Give a sequence of translate(), rotate(), and scale() that implements this

Translate(1, -2, 0)

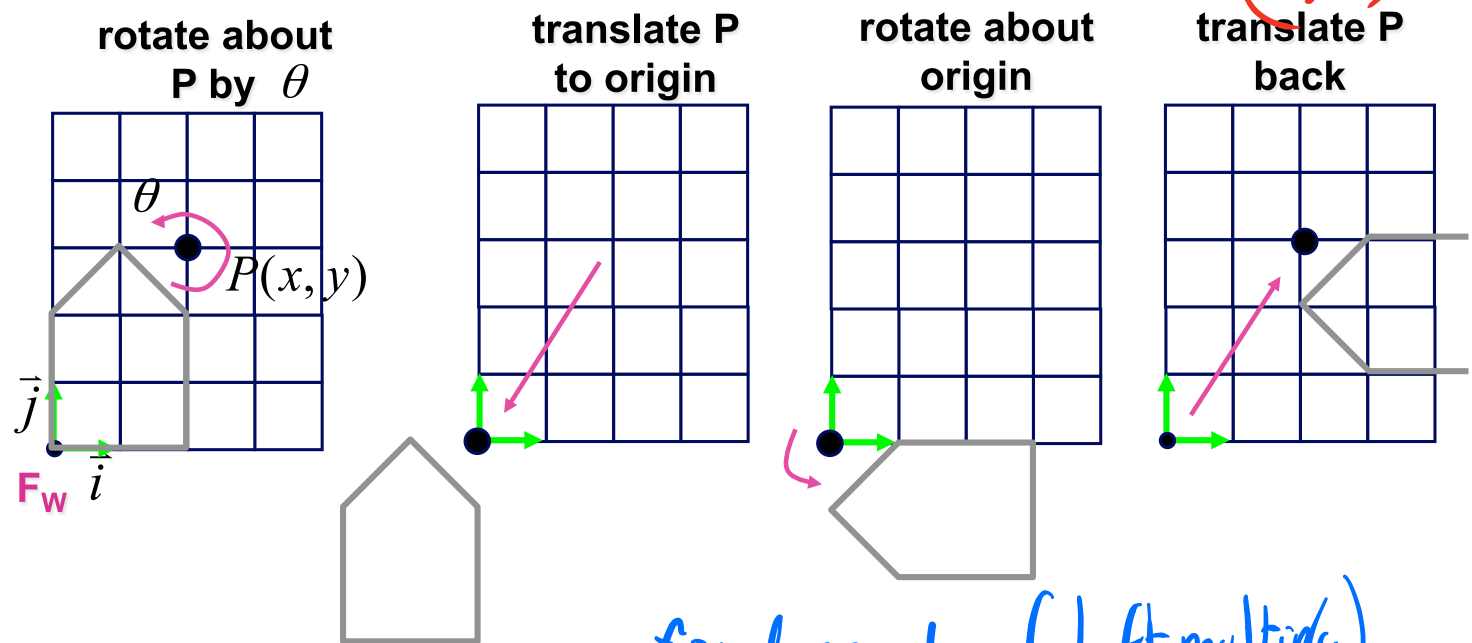
Rotate(z, 45°)

Scale($\sqrt{2}$, $\sqrt{2}$, 1)



Rotation about a point

e.g. Rot(z, 90°) about (2, 3)



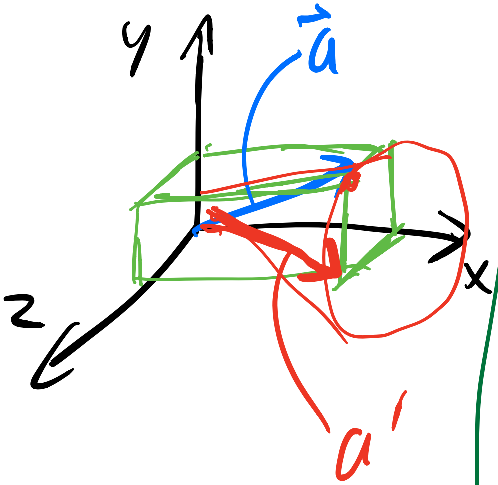
Think about this in fixed coords (left multiply)

$$M = Trans(x, y, 0)\, Rot(z, \theta)\, Trans(-x, -y, 0)$$

Rotation about an arbitrary axis

Rotate(~~angle, x, y~~, z);

Rotate$(\vec{a}, \theta)$



1. Rotate(x, ϕ) until $\vec{a}$ lies in the xz-plane

   $\phi = \operatorname{atan}(a_y / a_z)$

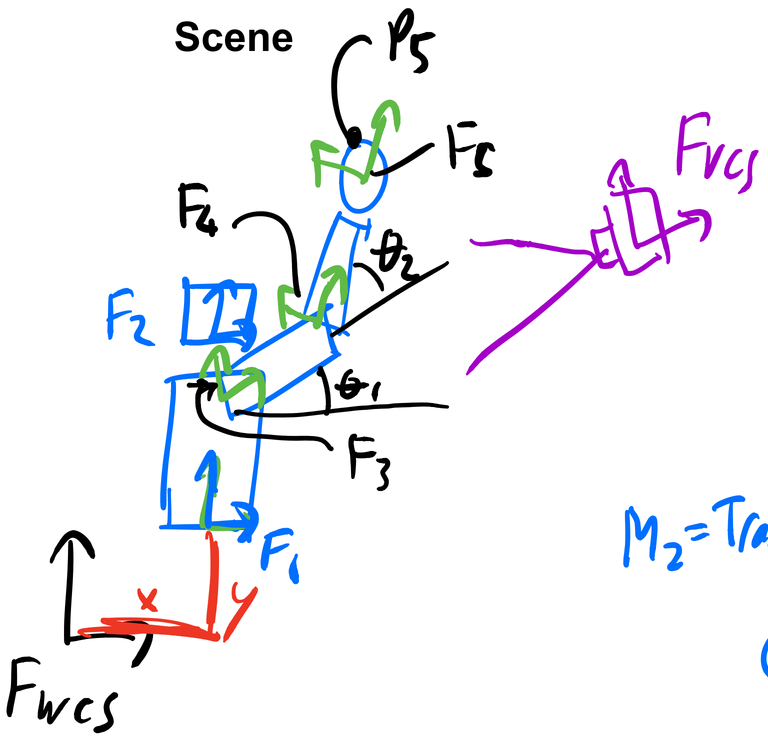
2. Rotate$(y, -\gamma)$ until a' is aligned with z-axis

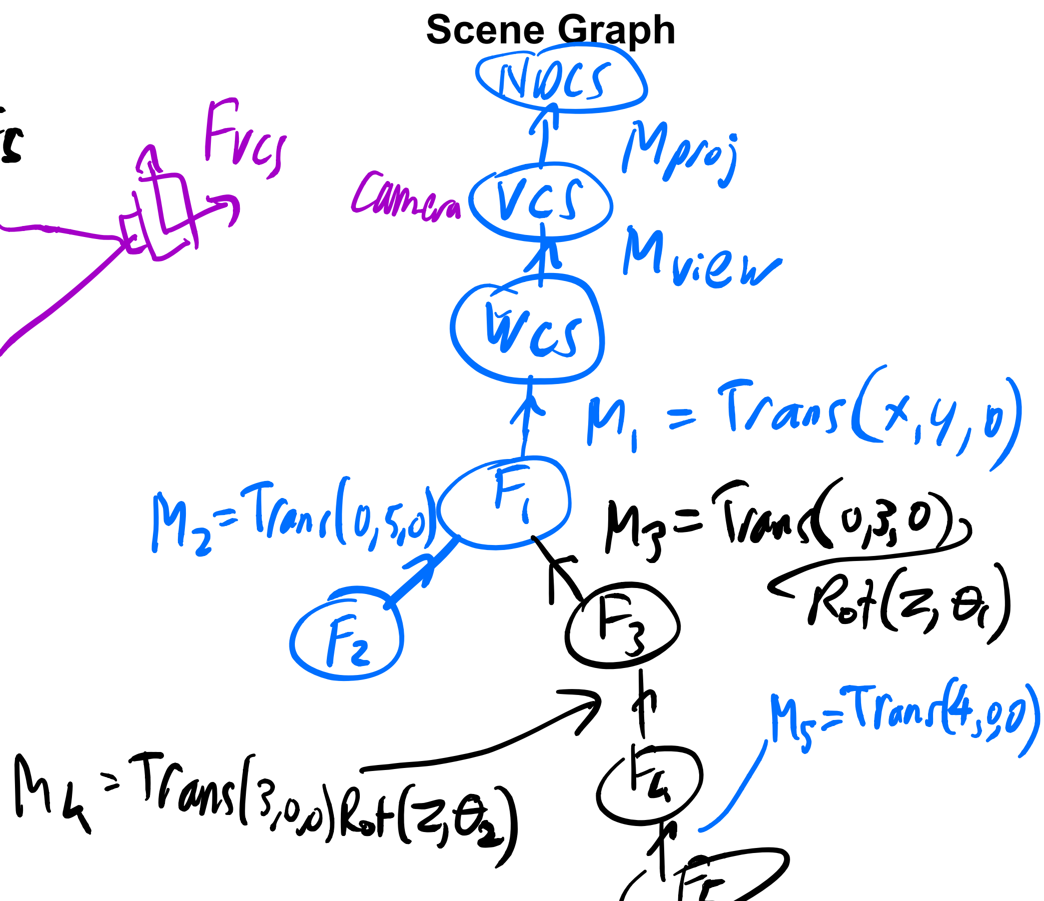
   $\gamma = \operatorname{atan}\left(a_x / \sqrt{a_y^2 + a_z^2}\right)$

3. Rotate(z, θ)
4. Rotate(y, γ) undo 2
5. Rotate$(x, -\phi)$ undo 1

$M = M_5 M_4 M_3 M_2 M_1$

Transformations in Scene Graphs (1)





Transformations in Scene Graphs (2)

Transforming Vertices

Math

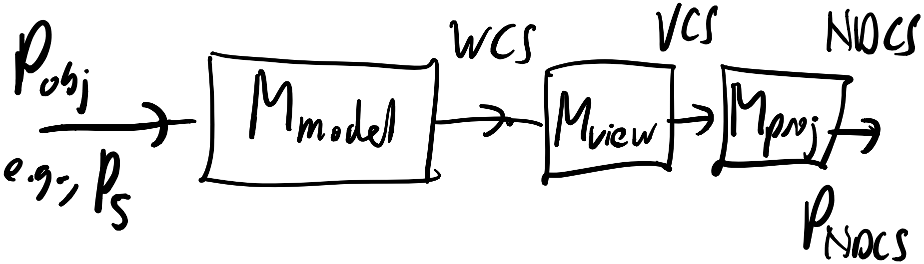
$$P_{NDCS} = M_{proj} M_{view} \overset{M_{model}}{\boxed{M_1 M_3 M_4 M_5}} P_5$$

Code

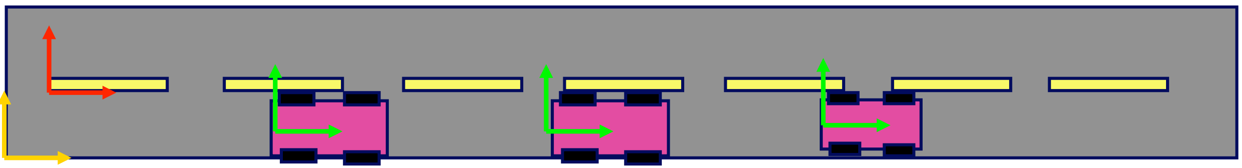
```
// assume all operations
   right multiply

M = I
M.setPerspective(...)   // M_proj
M.lookAt(...)           // M_view
M.translate(4,3,0);     // M_1
M.translate(0,3,0);
M.rotate(z, θ_1);       // M_3
   ⋮
drawBall();
```

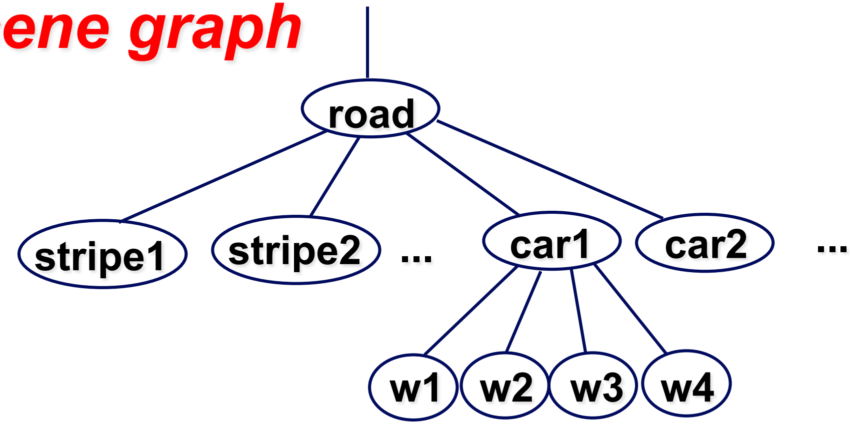
how we'll usually draw it:



Transformation Hierarchy

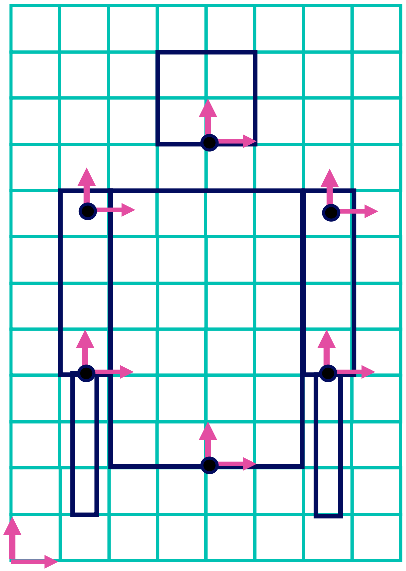


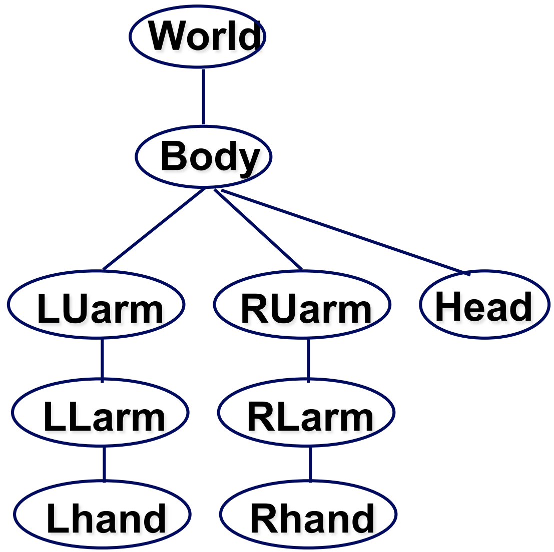
scene graph



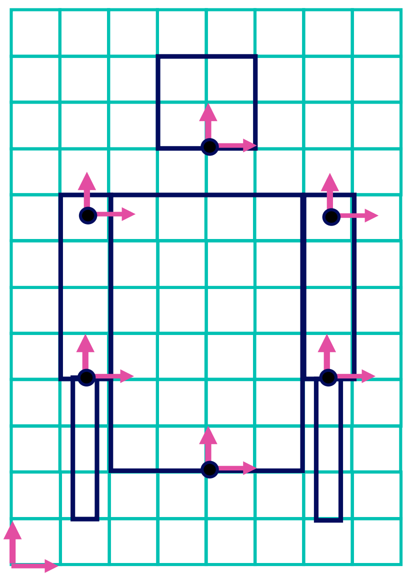
Transformation Hierarchy

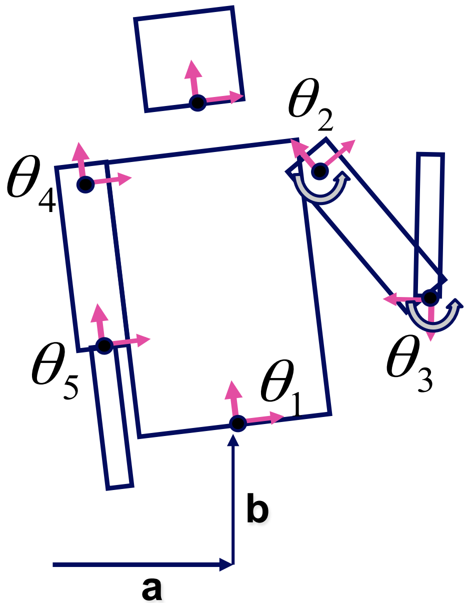
A matrix stack allows for convenient return to a previous coordinate frame.





Code to Draw using a Matrix Stack





looking at character from behind

```
M.Translate(a,b,0);
M.Rotatef(θ1,0,0,1);
DrawBody();
PushMatrix(M);
   M.Translate(0,7,0);
   M.DrawHead();
M=PopMatrix();
PushMatrix(M);
   M.Translate(2.5,5.5,0);
   M.Rotate(θ2,0,0,1);
   DrawRUarm();
   M.Translate(0,-3.5,0);
   M.Rotate(θ3,0,0,1);
   DrawRLarm();
M=PopMatrix();
... (draw left arm)
```