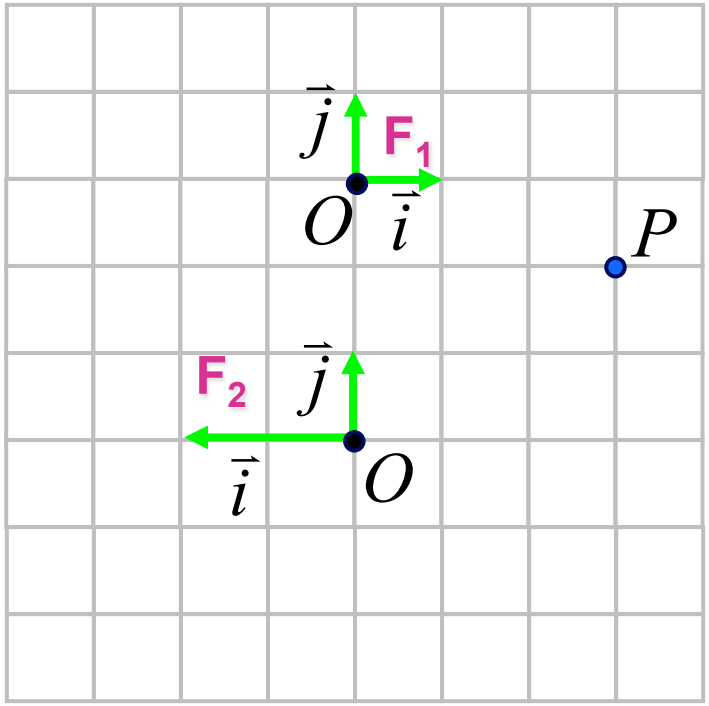
Transformations as a change of basis

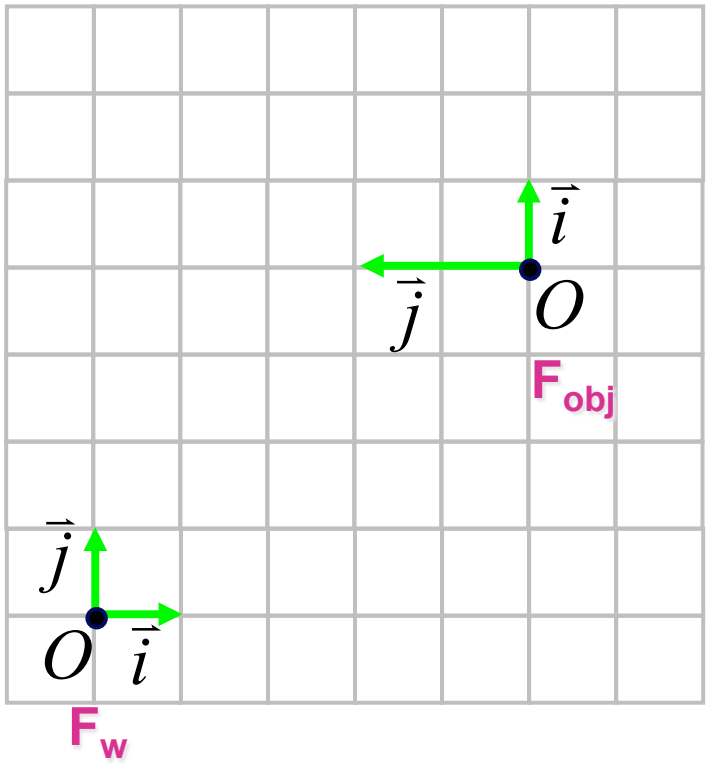


$P_1 =$ $P_2 =$ **Goal:**

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

check:

Transformations as a change of basis



Goal:

3D Transformations

Affine transformations

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

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

4x4 matrices

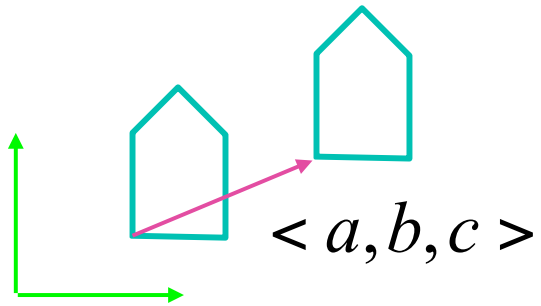
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & T_x \\ m_{21} & m_{22} & m_{23} & T_y \\ m_{31} & m_{32} & m_{33} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

h=1

Transformations

Translation

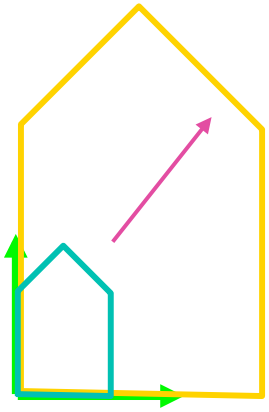
Translate(a,b,c)
Trans(a,b,c)



Transformations

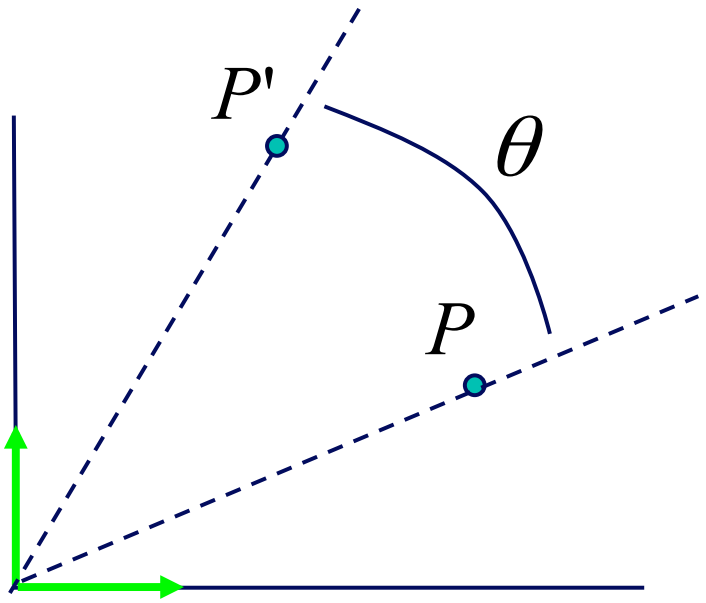
Scaling

Scale(a,b,c)



Transformations

Rotation

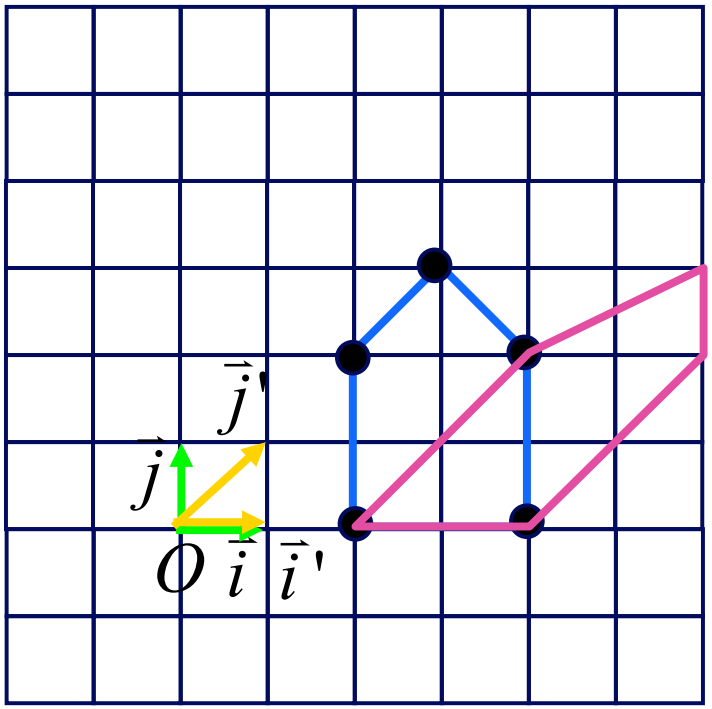


$Rotate(z,\theta)$

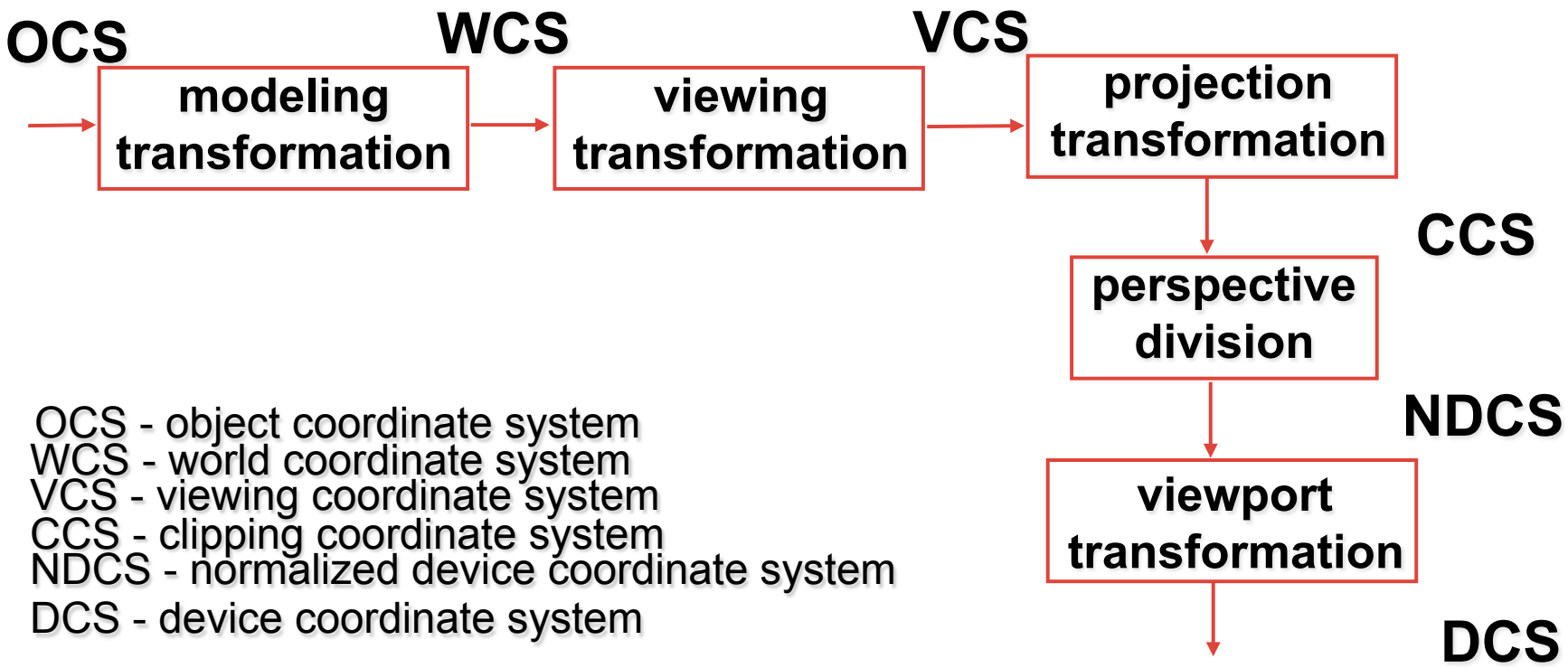
$Rot(z,\theta)$

Transformations

Shear



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)

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

$Rotate(z,\theta)$

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

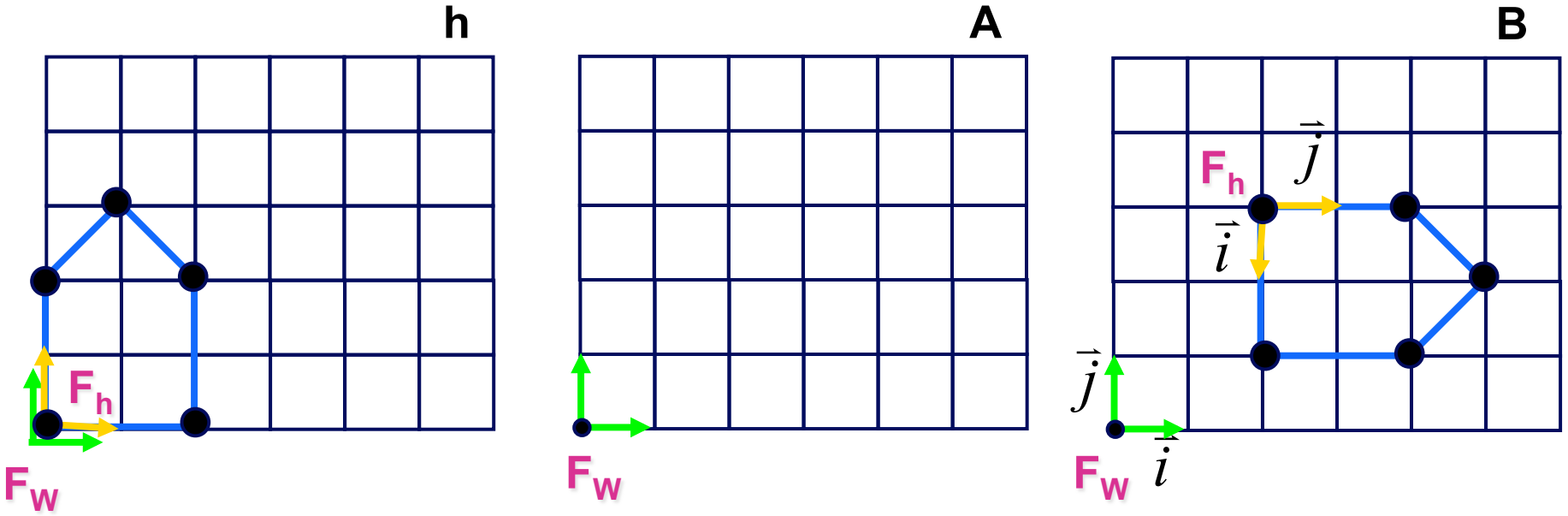
Simple Compositions

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

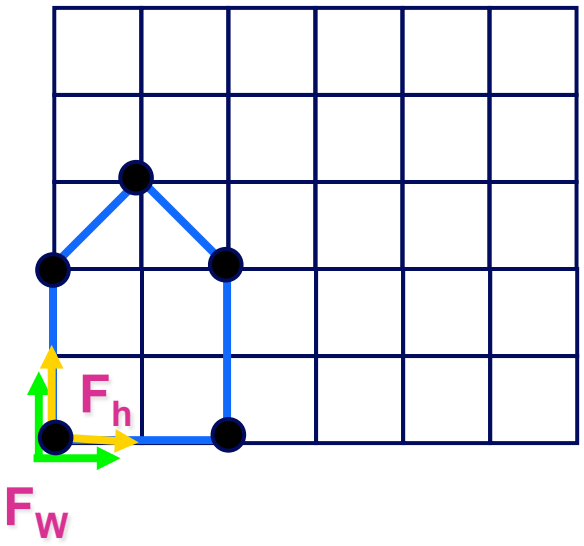
scale(a,b,c) scale(d,e,f)

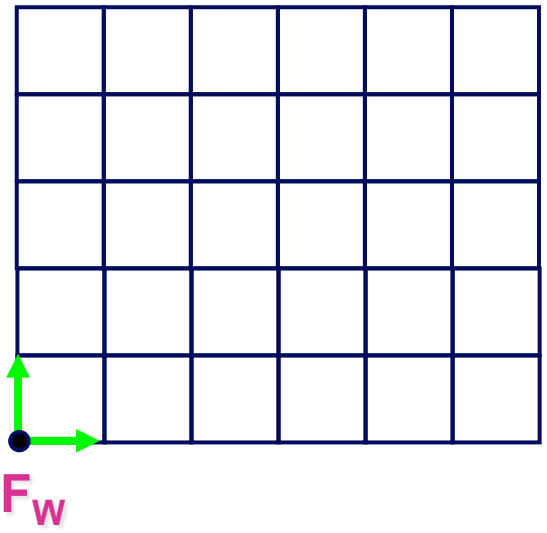
$Rotate(z,\theta_2)\ Rotate(z,\theta_2)$

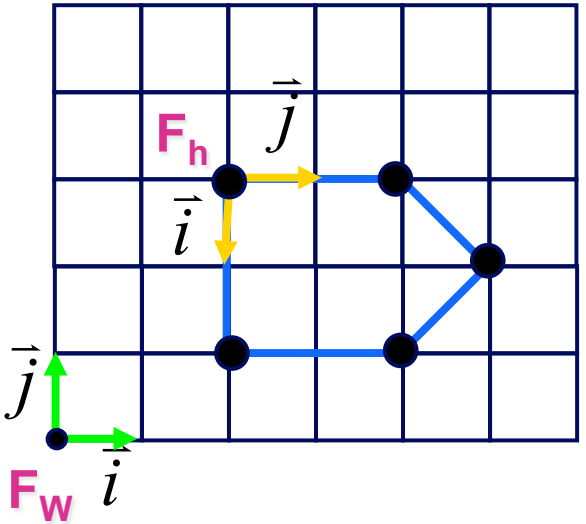
Composing Transformations
(thinking in fixed coords)



Composing Transformations
(thinking in local coords)



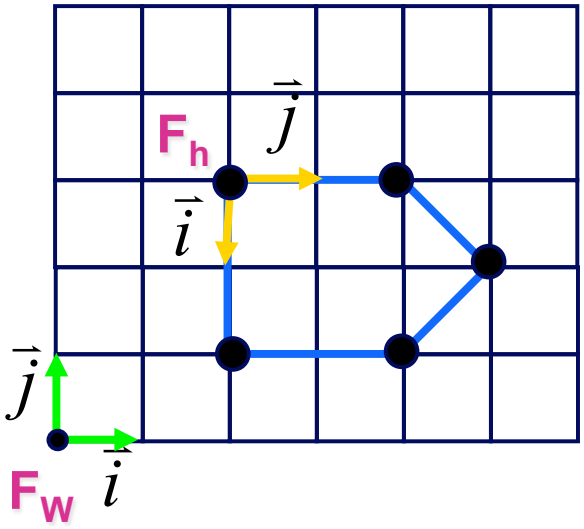




Composing Transformations

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

Summary Example

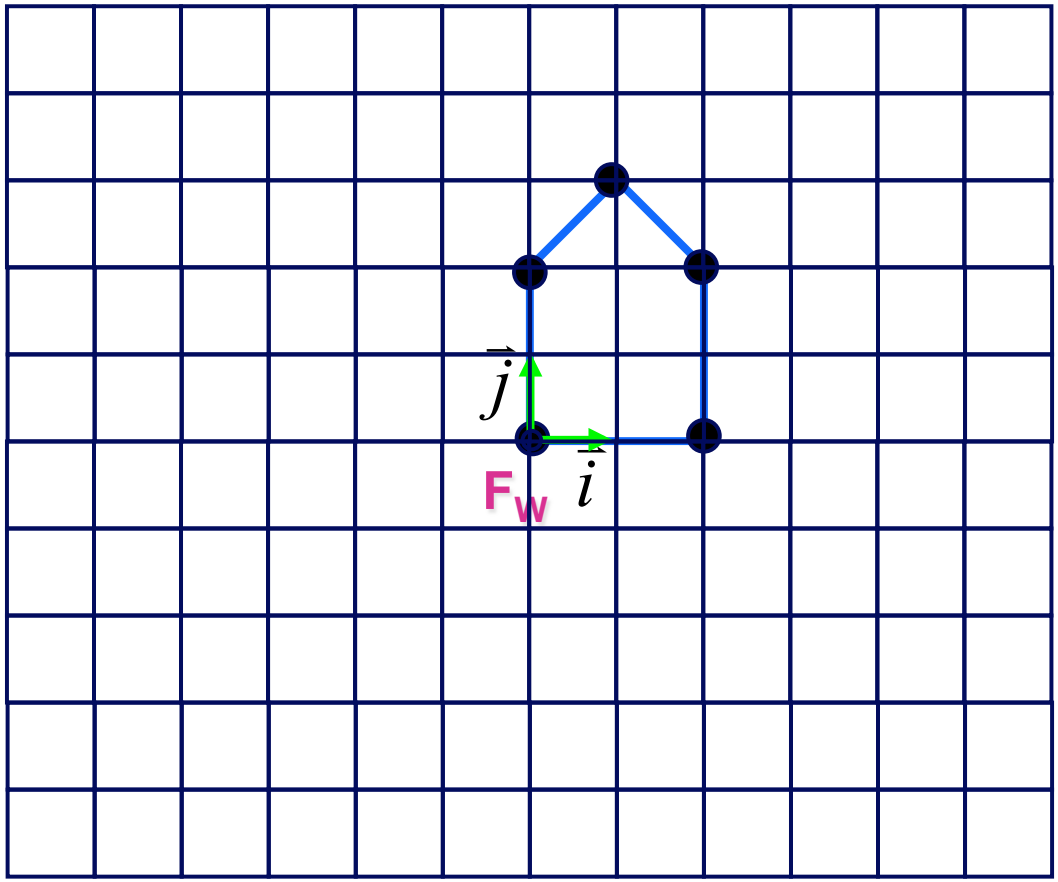


$$P' = M\,P$$

left multiply

right multiply

Test yourself ...

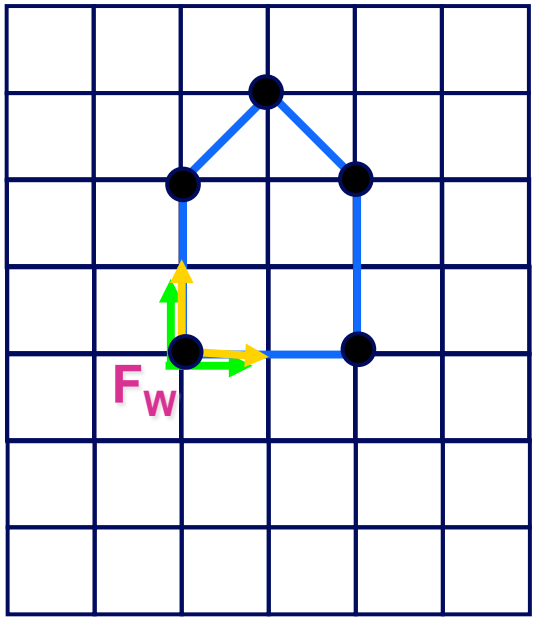



```
Translate(0,2,0);
Rotate(z,-90);
Scale(2,2,2);
Translate(1,0,0);
DrawHouse();
```

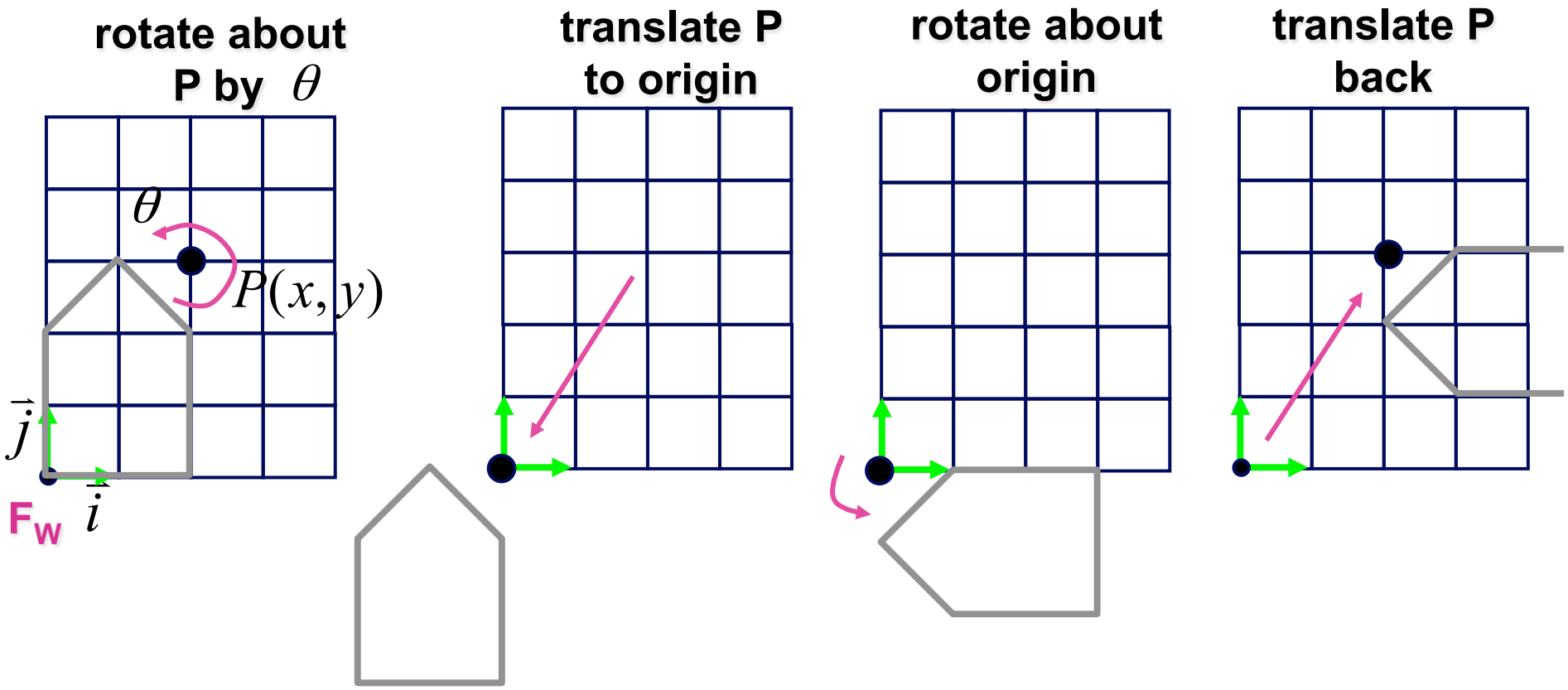
Test yourself

$$M = \begin{bmatrix} 1 & -1 & 0 & 1 \\ 1 & 1 & 0 & -2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(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



Rotation about a point



Rotation about an arbitrary axis

Rotate(angle, x, y, z);

Transformations in Scene Graphs (1)

Scene

Scene Graph

Transformations in Scene Graphs (2)

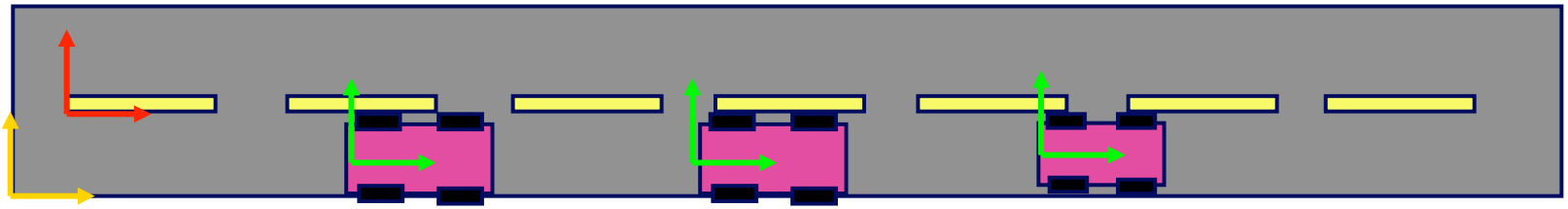
Transforming Vertices

Math

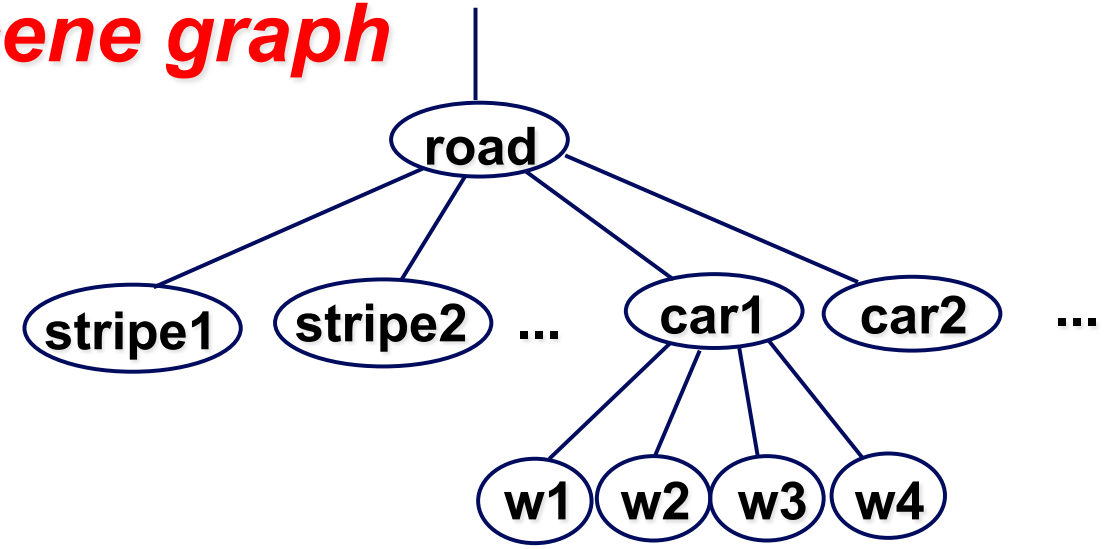
Code

how we'll usually draw it:

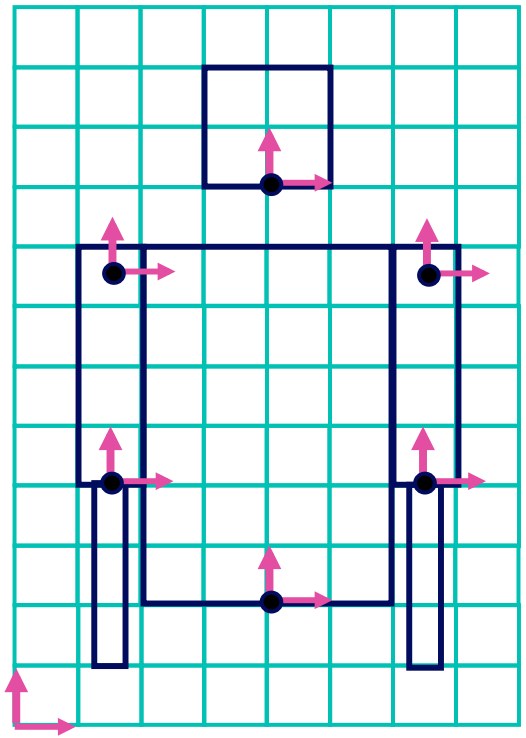
Transformation Hierarchy



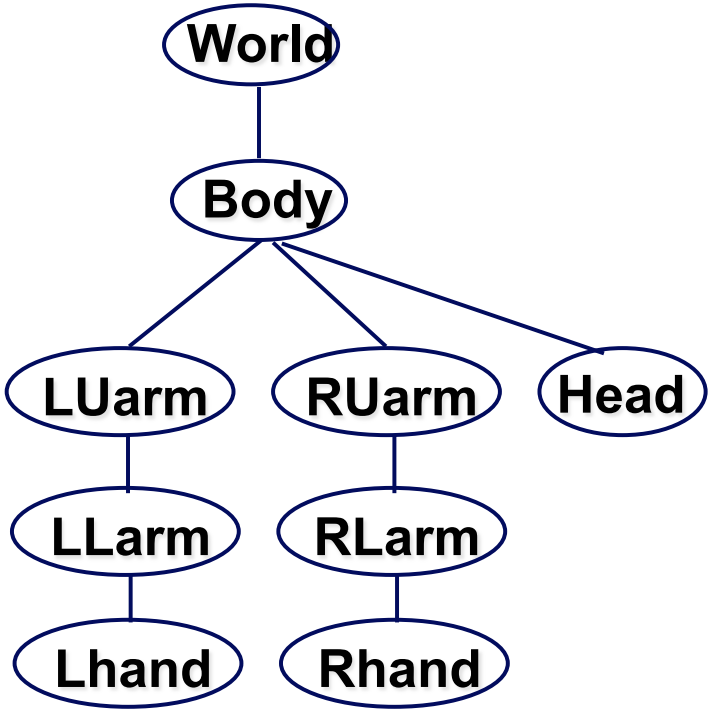
scene graph



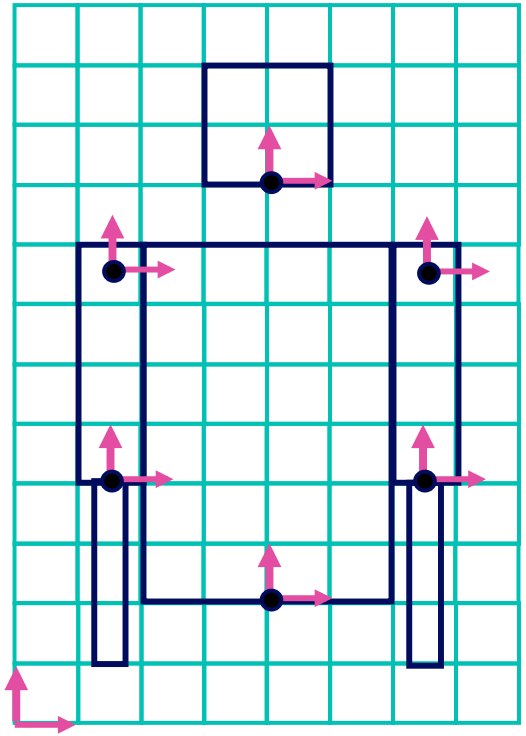
Transformation Hierarchy

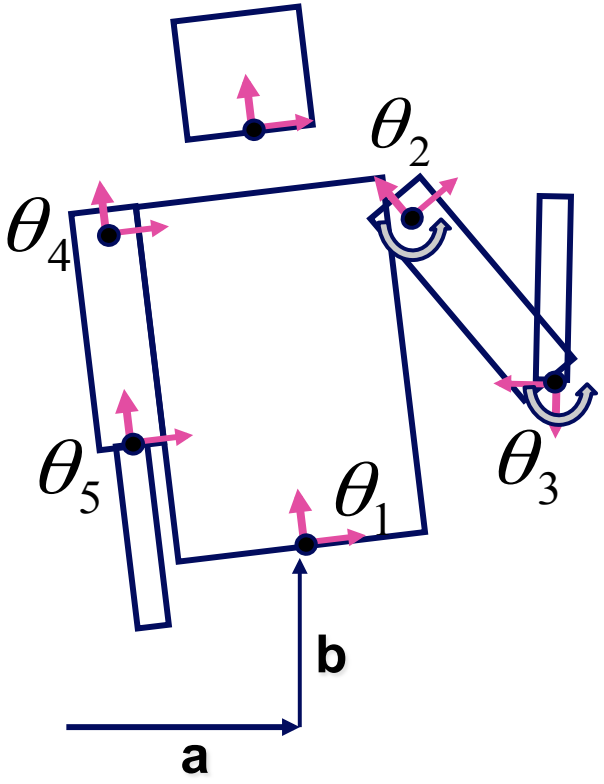


A matrix stack allows for convenient return to a previous coordinate frame.



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)
```