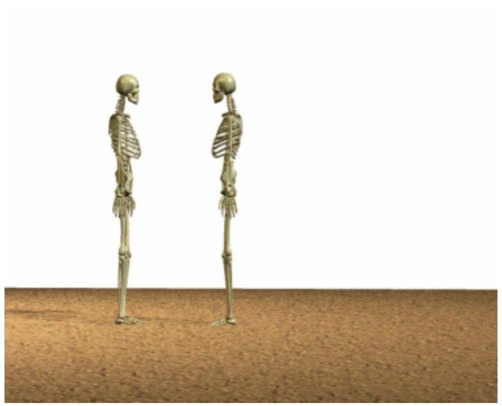
Affine Transformations in 3D



Recap from Last Lecture

- ***2D Linear Transforms***
- Homogeneous Coordinates
- 2D Affine Transforms
- Composite Transforms

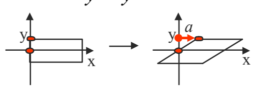
Elementary Linear Transformations

2D Transformations are represented as matrix multiplications

$$\mathbf{x}' = \mathbf{M}\mathbf{x} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Longleftrightarrow \begin{aligned} x' &= ax + by \\ y' &= cx + dy \end{aligned}$$

- Scaling: $\mathbf{M} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$ $\quad \begin{aligned} x' &= ax \\ y' &= by \end{aligned}$
- Shearing: $\begin{aligned} x' &= x + ay \\ y' &= y \end{aligned}$

$$\mathbf{M}_y = \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \quad \mathbf{M}_x = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$$



- Rotation: $\mathbf{M} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ $\quad \begin{aligned} x' &= x\cos(\theta) - y\sin(\theta) \\ y' &= x\sin(\theta) + y\cos(\theta) \end{aligned}$

Inverse Linear Transformations, M^{-1}

- Scaling: $\mathbf{M} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad \mathbf{M}^{-1} = \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix}$
- Shearing:

$$\mathbf{M}_x = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \quad \mathbf{M}_x^{-1} = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}$$

- Rotation:

$$\mathbf{M} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad \mathbf{M}^{-1} = \begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix}$$

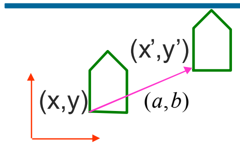
Summary: Linear Transformations

- ***Any*** linear transformation is a combination of a
  - Scale
  - Shear
  - Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Leftrightarrow \begin{matrix} x' = ax + by \\ y' = cx + dy \end{matrix}$$

- properties of linear transformations
  - satisifes T($s\mathbf{p}_1$+$t\mathbf{p}_2$) = s T($\mathbf{p}_1$) + t T($\mathbf{p}_2$)
  - origin maps to origin
  - lines map to lines
  - parallel lines remain parallel
  - relative ratios of points on a line are preserved
  - closed under composition– combination of linear transformations is a linear transform

2D Translation



vector addition

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x+a \\ y+b \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

matrix multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

matrix multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\text{rotation matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{\text{Translation as a multiplication matrix??}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Challenge

- Transforms represented as matrix multiplication
  - for everything except translation
  - why do everything with multiplication?
    - If every transform is a matrix multiplication then one can create composite transformations which are just a series of matrix multiplications, no special cases
- Create translation as a matrix multiplication
  - ***homogeneous coordinates*** trick
  - represent 2D coordinates (x,y) with 3-tuple (x,y,1)

$$\begin{bmatrix} x+a \\ y+b \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Recap from Last Lecture

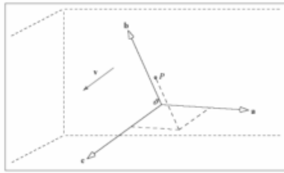
- 2D Linear Transforms
- ***Homogeneous Coordinates***
- 2D Affine Transforms
- Composite Transforms

Homogeneous Coordinates

Vectors and Points are represented as column matrices

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} \qquad P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Homogeneous Representation: Points & Vectors



Coordinate Frame defined by:

$$[\mathbf{a} \;\; \mathbf{b} \;\; \mathbf{c} \;\; O] = \begin{bmatrix} a_x & b_x & c_x & o_x \\ a_y & b_y & c_y & o_y \\ a_z & b_z & c_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c} \;\longrightarrow\; \mathbf{v} = [\mathbf{a} \;\, \mathbf{b} \;\, \mathbf{c} \;\, O]\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c} \;\longrightarrow\; P = [\mathbf{a} \;\, \mathbf{b} \;\, \mathbf{c} \;\, O]\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Does the Homogeneous Representation Support Operations?

Operations :

- $\mathbf{v} + \mathbf{w} = [v_1,v_2,v_3,0]^T + [w_1,w_2,w_3,0]^T$
  $= [v_1+w_1,\ v_2+w_2,\ v_3+w_3,\ 0]^T$ Vector!
- $a\mathbf{v} = a[v_1,v_2,v_3,0]^T = [av_1,\ av_2,\ av_3,\ 0]^T$ Vector!
- $a\mathbf{v} + b\mathbf{w} = a[v_1,v_2,v_3,0]^T + b[w_1,w_2,w_3,0]^T$
  $= [av_1+bw_1,\ av_2+bw_2,\ av_3+bw_3,\ 0]^T$ Vector!
- $P+\mathbf{v} = [p_1,p_2,p_3,1]^T + [v_1,v_2,v_3,0]^T$
  $= [p_1+v_1,\ p_2+v_2,\ p_3+v_3,\ 1]^T$ Point!

Does the Homogeneous Representation Support Operations?

- Valid operation if the last coordinate is 0 or a 1
  - **vector + vector = vector**
  - **point – point = vector**
  - **point + vector = point**
  - **point + point = ??**

Linear Combination of Points

Points P, Q scalars f, g*:*

$$fP+gQ = f\,[p_1,p_2,p_3,1]^T + g[q_1,q_2,q_3,1]^T$$
$$= [fp_1+gq_1,\ fp_2+gq_2,\ fp_3+gq_3,\ f+g]^T$$

What is this?

- If $(f+g) = 0$ then vector!
- If $(f+g) = 1$ then point!

Linear Combination of Points

Points ***P, Q*** *scalars* f, g*:*

$$f\mathbf{P}+g\mathbf{Q} = f\,[p_1,p_2,p_3,1]^T + g[q_1,q_2,q_3,1]^T$$
$$= [fp_1+gq_1,\ fp_2+gq_2,\ fp_3+gq_3,\ f+g]^T$$

What is this?

- If $(f+g) = 0$ then vector!
- If $(f+g) = 1$ then point!

Homogeneous Combinations of Points

Definition:

Points P_i: i = 1,…,n

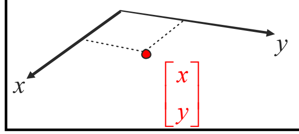
Scalars f_i: i = 1,…,n

$$f_1P_1 + \ldots + f_nP_n \quad \text{iff} \quad f_1 + \ldots + f_n = 1$$

Example: $0.5P_1 + 0.5P_2$

Homogeneous Coordinates Geometrically

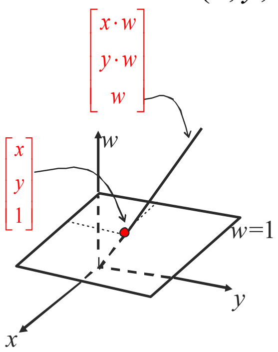
- point in 2D cartesian



Homogeneous Coordinates Geometrically

homogeneous **cartesian**

$$(x, y, w) \xrightarrow{/\,w} \left(\frac{x}{w}, \frac{y}{w}\right)$$



- point in 2D cartesian + weight w = point P in 3D homog. coords
- multiples of (x,y,w)
  - form a line L in 3D
  - all homogeneous points on L represent same 2D cartesian point
  - example: (2,2,1) = (4,4,2) = (1,1,0.5)

Recap from Last Lecture

- 2D Linear Transforms
- Homogeneous Coordinates
- ***2D Affine Transforms***
- Composite Transforms

Affine Transformations

- Affine transformation = linear transformation + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} a & b \\ c & d \end{pmatrix}}_{\text{Rotation, Scale, Shear}} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \underbrace{\begin{pmatrix} t_x \\ t_y \end{pmatrix}}_{\text{Translation}}$$

- Affine transformation using homogenous coordinates is only a matrix multiplication:

Rotation, Scale Shear — Translation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \boxed{\begin{matrix} a & b \\ c & d \end{matrix}} & \boxed{\begin{matrix} t_x \\ t_y \end{matrix}} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Linear Transformation → Affine Transformation

- Create elementary affine transformations from linear ones as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ d & e & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Affine Transformations Around the Origin

- our 2D transformation matrices are now 3x3:

x' = Mx

$$\mathbf{M}_{translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- use rightmost column

$$\mathbf{M}_{scale} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M}_{shear\_x} = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M}_{shear\_y} = \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

Inverse Affine Transformations Around the Origin

- our 2D transformation matrices are now 3x3:

$$\mathbf{M}^{-1}_{translation} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M}^{-1}_{scale} = \begin{bmatrix} 1/a & 0 & 0 \\ 0 & 1/b & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}^{-1}_{shear\_x} = \begin{bmatrix} 1 & -shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{M}^{-1}_{shear\_y} = \begin{bmatrix} 1 & 0 & 0 \\ -shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}^{-1}_{rotation} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformations

- ***Any*** affine transformation is a combination of
  - translation and/or
  - linear transformations:
    - scale, shear, rotation

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- properties of affine transformations
  - origin does not necessarily map to origin
  - lines map to lines
  - parallel lines remain parallel
  - relative ratios of points on a line are preserved
  - closed under composition - combination of affine transformations is an affine transform

Recap from Last Lecture

- 2D Linear Transforms
- Homogeneous Coordinates
- 2D Affine Transforms
- ***Composite Transforms***

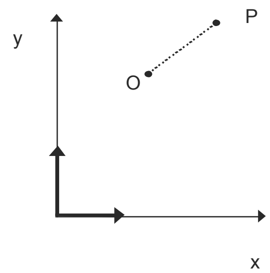
Composite 2D Transformations

The following are composite 2D transformations:

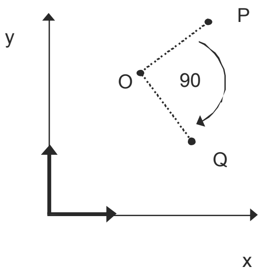
- Rotation about an arbitrary pivot point
- Scaling around an arbitrary point
- Reflection
- Reflection about a tilted line

Example of 2D Composite Transformation

Rotate -90 deg around an arbitrary point O:

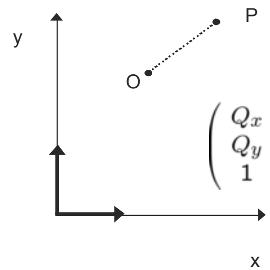


Rotate Around an Arbitrary Point



Rotate Around an Arbitrary Point

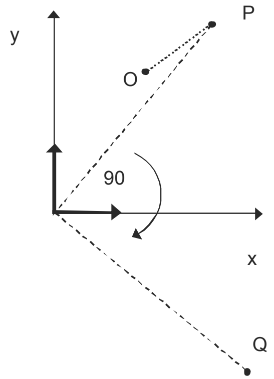
We know how to rotate around the origin



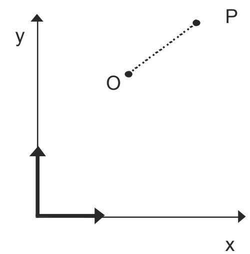
$$\begin{pmatrix} Q_x \\ Q_y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix}$$

Rotate Around an Arbitrary Point

…but that is not what we want to do!

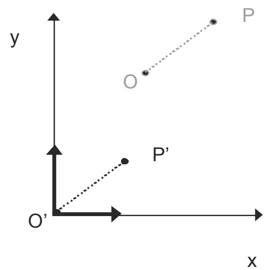


So What Do We Do?



Transform it to the Known Case

Translate***(-O$_x$,-O$_y$)***

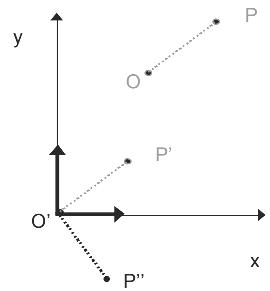


$$P' = \mathbf{T}\left(-O_x, -O_y\right)P$$

Second Step: Rotation

Translate(-***O$_x$***,-***O$_y$***)

Rotate(-90)



$$P' = \mathbf{T}\left(-O_x, -O_y\right)P$$

$$P'' = \mathbf{R}(90)P'$$

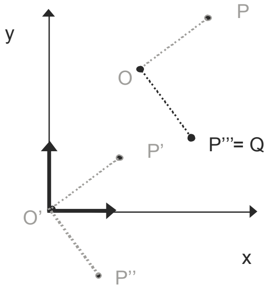
$$P'' = \mathbf{R}(90)\left(\mathbf{T}\left(-O_x, -O_y\right)P\right)$$

Finally, Put Everything Back

Translate($-O_x$,$-O_y$)

Rotate(-90)

Translate(O_x,O_y)



$$P' = \mathbf{T}(-O_x, -O_y)\,P$$

$$P'' = \mathbf{R}(90)P'$$

$$P'' = \mathbf{R}(90)\big(\mathbf{T}(-O_x, -O_y)\,P\big)$$

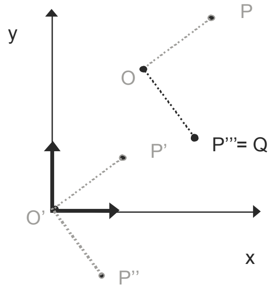
$$Q = \mathbf{T}(O_x, O_y)\big(\mathbf{R}(-90)\big(\mathbf{T}(-O_x, -O_y)\,P\big)\big)$$

Rotation About Arbitrary Point

Composite transformation representation:

$$\mathbf{M} = \mathbf{T}(O_x, O_y)\mathbf{R}(-90)\mathbf{T}(-O_x, -O_y)$$

Order is IMPORTANT!



$$Q = \mathbf{T}(O_x, O_y)\big(\mathbf{R}(-90)\big(\mathbf{T}(-O_x, -O_y)\,P\big)\big)$$

$$Q = \big(\mathbf{T}(O_x, O_y)\mathbf{R}(-90)\mathbf{T}(-O_x, -O_y)\big)\,P$$

Do not flip the order

Recall: Matrix order matters,

$$\mathbf{AB} \neq \mathbf{BA}$$

but multiplication order does not

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$

Transformations 2

- Reading: Chapter 6

Lecture 4: Outline

- **3D Affine Transforms**
- Composite Transforms
  - Composite Rotations
  - Gimble lock
  - Rotation around an arbitrary axis
  - Composite Translations, Scales, Shears
- Transformation Interpretation:
  - Right to Left Interpretation: – Changing Location of Objects
  - Left to Right Interpretation: -- Changing the Coord. System
- Transformation Hierarchies

Affine Transformations in 3D

General form

$$\begin{bmatrix} Q_x \\ Q_y \\ Q_z \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Or:

$$Q = \mathrm{M}P$$

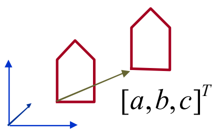
General Form

Rotation, Scaling, Shear

Translation

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

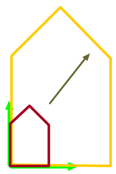
3D Translation



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

Translate(a,b,c);

3D Scaling



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

Scale(a,b,c);

3D Shear

- General shear

$$\mathbf{Shear}(hxy, hxz, hyx, hyz, hzx, hzy) = \begin{bmatrix} 1 & hyx & hzx & 0 \\ hxy & 1 & hzy & 0 \\ hxz & hyz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

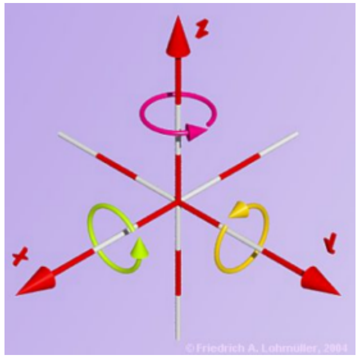
  - to avoid ambiguity, always say "shear along <axis> in direction of <axis>"

$$\text{Shear}_{\text{AlongXinDirectionOfY}}(h) = \begin{bmatrix} 1 & h & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Shear}_{\text{AlongXinDirectionOfZ}}(h) = \begin{bmatrix} 1 & 0 & h & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

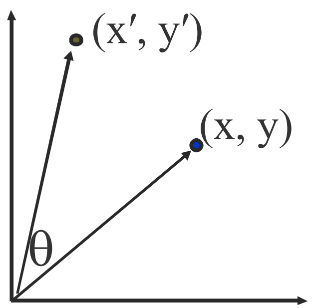
$$\text{Shear}_{\text{AlongYinDirectionOfX}}(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ h & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Shear}_{\text{AlongYinDirectionOfZ}}(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & h & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Shear}_{\text{AlongZinDirectionOfX}}(h) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ h & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Shear}_{\text{AlongZinDirectionOfY}}(\text{h}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & h & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Three Axes to Rotate Around



3D Rotation Around Z Axis



$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

$$z' = z$$

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

- general OpenGL command
  glRotatef(angle,x,y,z);
- rotate in z
  RotateZ(angle);

3D Rotation in X, Y

around x axis: **glRotatef(angle,1,0,0);** **RotateX(angle);**

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

around y axis: **glRotatef(angle,0,1,0);** **RotateY(angle);**

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

Summary: Transformations

Translate

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

Scale

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

General shear -- to avoid ambiguity, always say "shear along <axis> in direction of <axis>"

$$\mathbf{Shear}(hxy, hxz, hyx, hyz, hzx, hzy) = \begin{bmatrix} 1 & hyx & hzx & 0 \\ hxy & 1 & hzy & 0 \\ hxz & hyz & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate X

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

Rotate Y

$$\begin{bmatrix} \cos\theta & & \sin\theta & \\ & 1 & & \\ -\sin\theta & & \cos\theta & \\ & & & 1 \end{bmatrix}$$

Rotate Z

$$\begin{bmatrix} \cos\theta & -\sin\theta & & \\ \sin\theta & \cos\theta & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

Inverse Transformations

$$\mathbf{T}(x,y,z)^{-1} = \mathbf{T}(-x,-y,-z)$$
$$\mathbf{T}(x,y,z)\ \mathbf{T}(-x,-y,-z) = \mathbf{I}$$

$$\mathbf{S}(sx,sy,sz)^{-1} = \mathbf{S}(\frac{1}{sx},\frac{1}{sy},\frac{1}{sz})$$
$$\mathbf{S}(sx,sy,sz)\mathbf{S}(\frac{1}{sx},\frac{1}{sy},\frac{1}{sz}) = \mathbf{I}$$

$$\mathbf{R}(z,\theta)^{-1} = \mathbf{R}(z,-\theta) = \mathbf{R}^{\mathrm{T}}(z,\theta)$$
$$\mathbf{R}(z,\theta)\ \mathbf{R}(z,-\theta) = \mathbf{I}$$

(R is orthonormal)

Inverse of Rotations

Pure rotation only, no scaling or shear

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$\mathrm{M}^{-1} = \mathrm{M}^{T}$$

Since the rotation matrix *M* is an orthonormal matrix

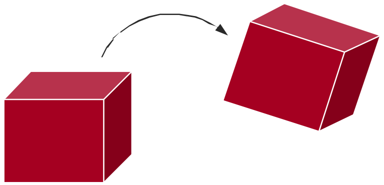
Properties of Affine Transformations

1. Affine transformations are composed of elementary ones
2. Preservation of affine combinations of points
3. Preservation of lines and planes
4. Preservation of parallelism of lines and planes
5. Relative ratios are preserved

Rigid Body Transformations

Translations and rotations

Preserves lines, angles and distances



Affine Combinations of Points

$$W = a_1 P_1 + a_2 P_2$$

$$T(W) = T(a_1 P_1 + a_2 P_2) = a_1 T(P_1) + a_2 T(P_2)$$

Proof: from linearity of matrix multiplication

$$\mathbf{M}W = \mathbf{M}(a_1 P_1 + a_2 P_2) = a_1 \mathbf{M} P_1 + a_2 \mathbf{M} P_2$$

Preservations of Lines and Planes

$$L(t) = (1-t)P_1 + tP_2$$

$$T(L) = (1-t)T(P_1) + tT(P_2) = (1-t)\mathbf{M}P_1 + t\mathbf{M}P_2$$

$$Pl(s,t) = (1-s-t)P_1 + tP_2 + sP_3$$

$$\begin{aligned} T(Pl) &= (1-s-t)T(P_1) + tT(P_2) + sT(P_3) \\ &= (1-s-t)\mathbf{M}P_1 + t\mathbf{M}P_2 + s\mathbf{M}P_3 \end{aligned}$$

Preservation of Parallelism

$$L(t) = P + t\mathbf{u}$$

$$\mathbf{M}L = \mathbf{M}(P + t\mathbf{u}) = \mathbf{M}P + \mathbf{M}(t\mathbf{u}) \rightarrow$$
$$\mathbf{M}L = \mathbf{M}P + t(\mathbf{M}\mathbf{u})$$

$\mathbf{Mu}$ independent of P

Similarly for planes

Lecture Outline

- 3D Affine Transforms
- ***Composite Transforms***
  - Composite Rotations
  - Gimble lock
  - Rotation around an arbitrary axis
  - Composite Translations, Scales, Shears
- Transformation Interpretation:
  - Right to Left Interpretation: – Changing Location of Objects
  - Left to Right Interpretation: -- Changing the Coord. System
- Transformation Hierarchies

Composition of 3D Affine Transformations

The composition of affine transformations is an affine transformation

Any 3D affine transformation can be performed as a series of elementary affine transformations

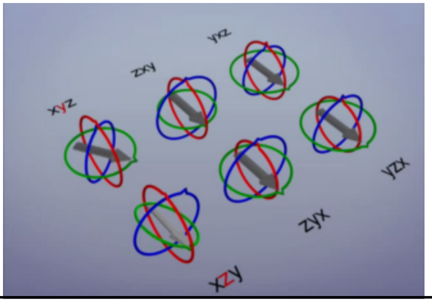
Composite 3D Rotation About the Orig

$$\mathbf{R}(\theta_1, \theta_2, \theta_3) = \mathbf{R}_z(\theta_3)\mathbf{R}_y(\theta_2)\mathbf{R}_x(\theta_1)$$

- ***This is known as the "Euler angle" representation of 3D rotations***
- ***The order of the rotation matrices is important !!***
- ***Note: The Euler angle representation suffers from singularities***

Gimbal Lock

- When the middle rotation is 90, the last rotational axis is aligned with the first rotational axis. Thus losing a degree of freedom and resulting in a gimble lock.



Gimbal Lock

$$\mathbf{R}(\theta_1,\theta_2,\theta_3) = \mathbf{R}_z(\theta_3)R_y(\theta_2)\mathbf{R}_x(\theta_1)$$

$$\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}(\theta_1, 90^o, \theta_3) = \mathbf{R}_z(\theta_3)\mathbf{R}_y(90^o)\mathbf{R}_x(\theta_1)$$

$$\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Loss of a Rotational Degree of Freedom

$$\mathbf{R}(\theta_1, 90^o, \theta_3) = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & \cos(\theta_3)\sin(\theta_1) - \sin(\theta_3)\cos(\theta_1) & \cos(\theta_3)\cos(\theta_1) + \sin(\theta_3)\sin(\theta_1) & 0 \\ 0 & \cos(\theta_3)\cos(\theta_1) + \sin(\theta_3)\sin(\theta_1) & -\cos(\theta_3)\sin(\theta_1) + \sin(\theta_3)\cos(\theta_1) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & \sin(\theta_1-\theta_3) & \cos(\theta_1-\theta_3) & 0 \\ 0 & \cos(\theta_1-\theta_3) & -\sin(\theta_1-\theta_3) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{R}(\theta) \qquad (\theta_1,\theta_3) \to \theta = (\theta_1 - \theta_3)$$

There are Alternatives

It is often convenient to use other representations of 3D rotations that do not suffer from Gimbal Lock

- Advanced concepts
  - Quaternions
  - Exponential Maps

Rotation Around an Arbitrary Axis

Euler's theorem:

Any rotation or sequence of rotations around a point is equivalent to a single rotation around an axis that passes through the point

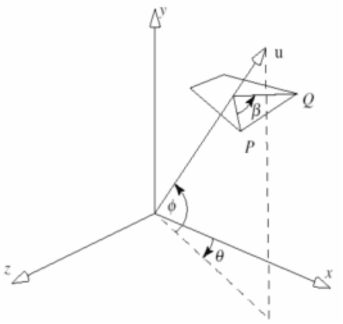
What does the matrix look like?

Rotation Around an Arbitrary Axis

Vector (axis): **u**
Rotation angle: β
Point: P

Method:

1. Two rotations to align ***u*** with x-axis
2. Do x-roll by β
3. Undo the alignment



Derivation

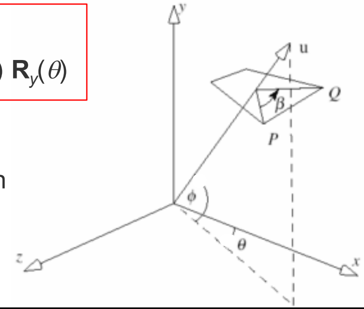
1. $\mathbf{R}_z(-\phi)\ \mathbf{R}_y(\theta)$
2. $\mathbf{R}_x(\beta)$
3. $\mathbf{R}_y(-\theta)\ \mathbf{R}_z(\phi)$

$$\cos(\theta) = u_x/\sqrt{u_x^2+u_z^2}$$
$$\sin(\theta) = u_z/\sqrt{u_x^2+u_z^2}$$
$$\sin(\phi) = u_y/|\mathbf{u}|$$
$$\cos(\phi) = \sqrt{u_x^2+u_z^2}/|\mathbf{u}|$$

All together:

$\mathbf{R}_\mathbf{u}(\beta) = \mathbf{R}_y(-\theta)\ \mathbf{R}_z(\phi)\ \mathbf{R}_x(\beta)\ \mathbf{R}_z(-\phi)\ \mathbf{R}_y(\theta)$

We should add translation too if the axis is not through the origin



Composing Transformations

- Translation

$$\mathbf{T}_1 = \mathbf{T}(dx_1, dy_1) = \begin{bmatrix} 1 & & & dx_1 \\ & 1 & & dy_1 \\ & & 1 & \\ & & & 1 \end{bmatrix} \qquad \mathbf{T}_2 = \mathbf{T}(dx_2, dy_2) = \begin{bmatrix} 1 & & & dx_2 \\ & 1 & & dy_2 \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$$P'' = \mathbf{T}_2 P' = \mathbf{T}_2(\mathbf{T}_1 P) = (\mathbf{T}_2\mathbf{T}_1)P, \textit{ where}$$

$$\mathbf{T}_2\mathbf{T}_1 = \begin{bmatrix} 1 & & & dx_1+dx_2 \\ & 1 & & dy_1+dy_2 \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

so translations add

Composing Transformations

- Scaling

$$\mathbf{S}_2\mathbf{S}_1 = \begin{bmatrix} sx_1 * sx_2 & & & \\ & sy_1 * sy_2 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

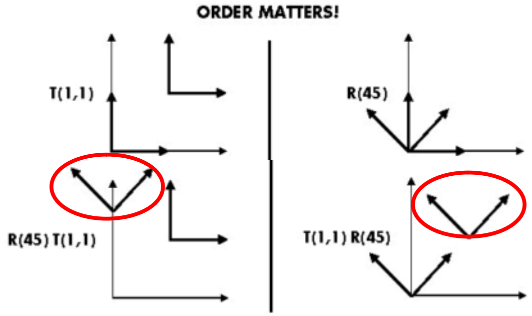
so scales multiply

- Rotation

$$\mathbf{R}_2\mathbf{R}_1 = \begin{bmatrix} \cos(\theta 1+\theta 2) & -\sin(\theta 1+\theta 2) & & \\ \sin(\theta 1+\theta 2) & \cos(\theta 1+\theta 2) & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

so rotations add

Composing Transformations

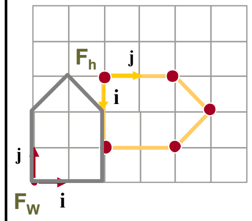


$T_a\ T_b = T_b\ T_a$, but $R_a\ R_b \ != R_b\ R_a$ and $T_a\ R_b \ != R_b\ T_a$

- translations commute
- rotations around same axis commute
- rotations around different axes do not commute
- rotations and translations do not commute

Composing Transformations

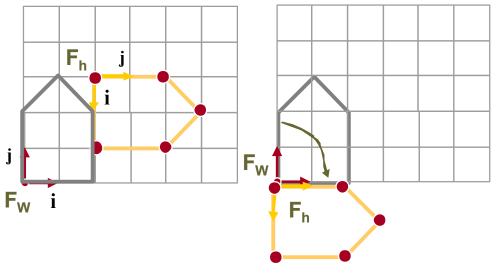
suppose we want



Composing Transformations

suppose we want

Rotate(z,-90)



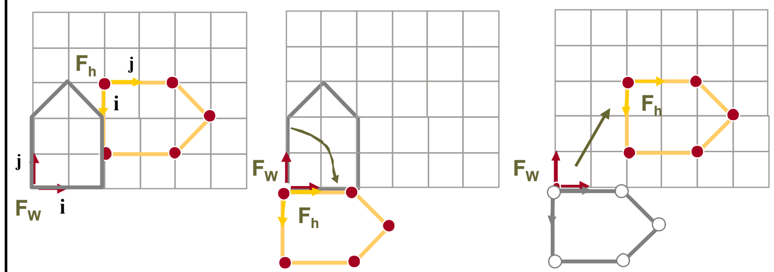
$$\mathbf{p}' = \mathbf{R}(z,-90)\,\mathbf{p}$$

Composing Transformations

suppose we want

Rotate(z,-90)

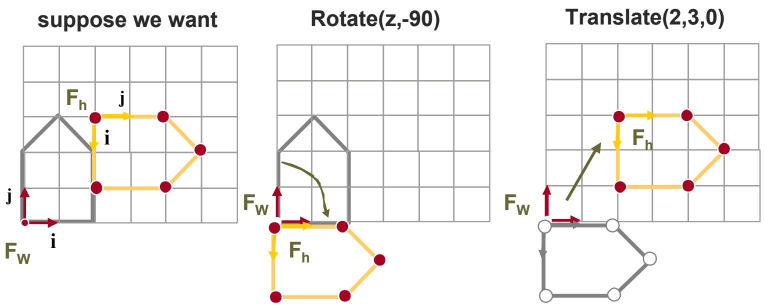
Translate(2,3,0)



$$\mathbf{p}' = \mathbf{R}(z,-90)\,\mathbf{p}$$

$$\mathbf{p}'' = \mathbf{T}(2,3,0)\,\mathbf{p}'$$

Composing Transformations



$\mathbf{p}' = \mathbf{R}(z,-90)\,\mathbf{p}$

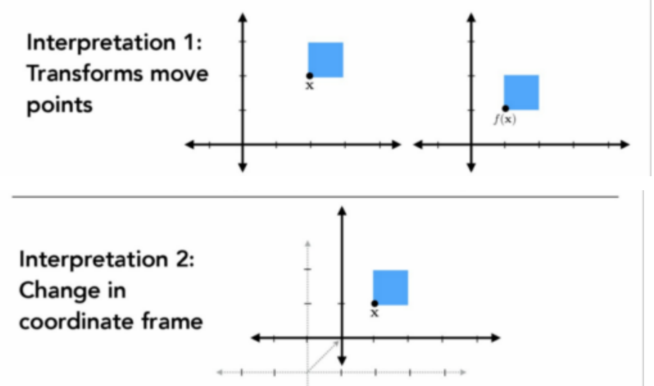
$\mathbf{p}'' = \mathbf{T}(2,3,0)\,\mathbf{p}'$

$\mathbf{p}'' = \mathbf{T}(2,3,0)\,\mathbf{R}(z,-90)\mathbf{p} = \mathbf{TRp}$

Lecture Outline

- 3D Affine Transforms
- Composite Transforms
  - Composite Rotations
  - Gimble lock
  - Rotation around an arbitrary axis
  - Composite Translations, Scales, Shears
- Transformation Interpretation:
  - Right to Left Interpretation: – Changing Location of Objects
  - Left to Right Interpretation: -- Changing the Coord. System
- Transformation Hierarchies

Transformation Interpretation



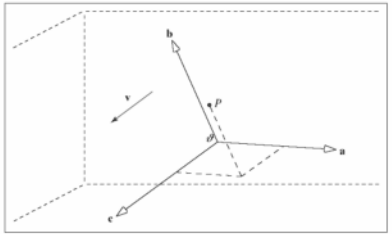
- https://processing.org/tutorials/transform2d/

Transformations of Coordinate Frames

Coordinate frames consist of vectors and an origin (point), therefore we can transform them just like points and vectors

This provides an alternative way to think of transformations—as changes of coordinate systems

Reminder: Points, Vectors, Coordinate Frames



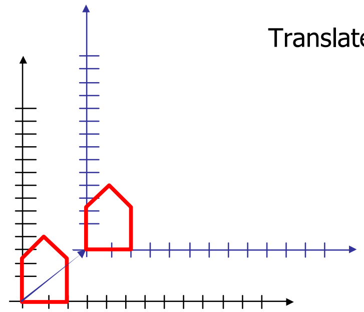
Coordinate Frame defined by:

$$[\mathbf{a} \quad \mathbf{b} \quad \mathbf{c} \quad O] = \begin{bmatrix} a_x & b_x & c_x & o_x \\ a_y & b_y & c_y & o_y \\ a_z & b_z & c_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{v} = v_1\mathbf{a} + v_2\mathbf{b} + v_3\mathbf{c} \quad \Rightarrow \quad \mathbf{v} = [\mathbf{a}, \mathbf{b}, \mathbf{c}, O] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

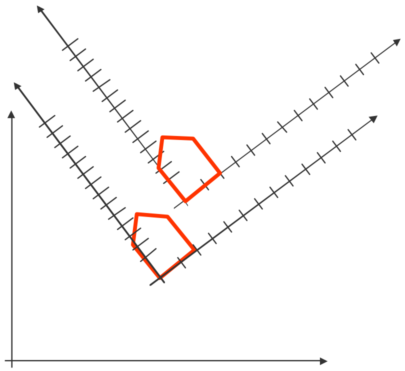
$$P = O + p_1\mathbf{a} + p_2\mathbf{b} + p_3\mathbf{c} \quad \Rightarrow \quad P = [\mathbf{a}, \mathbf{b}, \mathbf{c}, O] \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix}$$

Translate Coordinate Frame



Translate (3,3)?

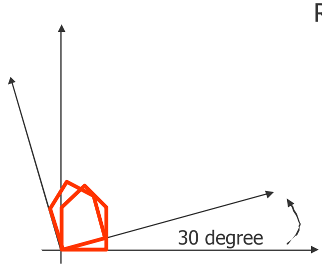
Translate Coordinate Frame



Translate (3,3)?

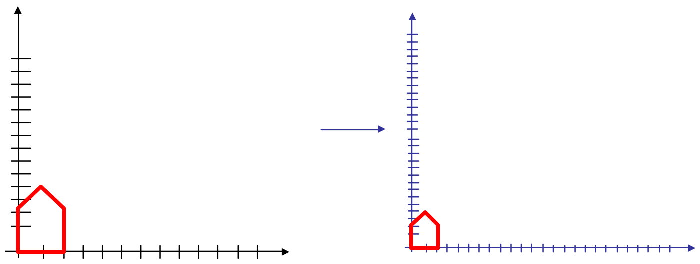
Rotate Coordinate Frame

Rotate 30 degree?



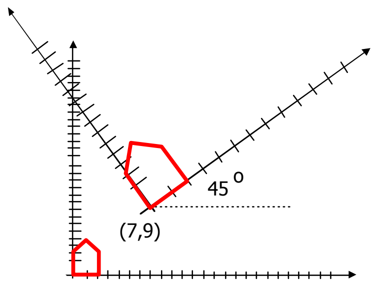
Scale Coordinate Frame

Scale (0.5,0.5)?



Compose Transformations

Coordinate Frame Transformations?



Answer:

1. Translate(7,9)
2. Rotate 45
3. Scale (2,2)

Transform Objects

- What does coordinate frame transformation have anything to do with object transformation?
- You can view transformation as to tie the object to a local coordinate frame and move that coordinate frame

Composing Transformations

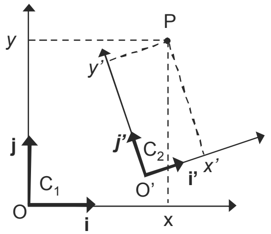
$$\mathbf{p}_{transformed} = \mathbf{TRp}$$

- Interpretation:
  - right to left
    - interpret operations wrt fixed coordinate frame
    - moving the object
  - left to right
    - interpret operations wrt local coordinate frame
    - changing the local coordinate system

Algebraic Derivation: Transforming C_1 into C_2

- What is the relationship between
  - **P in C_2** and
  - **P in C_1**

if **C_2 = T(C_1)**?



$$C_1 : P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$C_2 : P = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

$$O' = T(O),$$
$$\mathbf{i}' = T(\mathbf{i}),$$
$$\mathbf{j}' = T(\mathbf{j}),$$
$$\mathbf{k}' = T(\mathbf{k})$$

Derivation

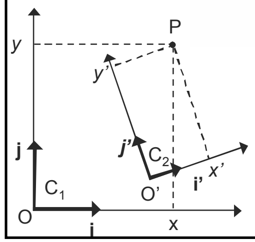
By definition P is the linear combination of vectors $\mathbf{i}', \mathbf{j}', \mathbf{k}'$ and point O'.

$$P \;=\; x'\mathbf{i}' + y'\mathbf{j}' + z'\mathbf{k}' + O'$$

In coordinate frame, C_1:

$$P_{C_1} \;=\; x'\mathbf{i}'_{C_1} + y'\mathbf{j}'_{C_1} + z'\mathbf{k}'_{C_1} + O'_{C_1}$$

$$[\mathbf{i}'_{C_1}, \mathbf{j}'_{C_1}, \mathbf{k}'_{C_1}, O'_{C_1}] = T([\mathbf{i}, \mathbf{j}, \mathbf{k}, O])$$



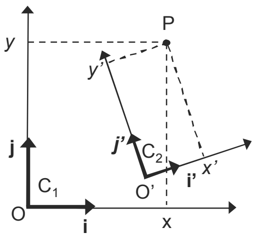
Derivation

$$P_{C_1} \;=\; x'\mathbf{i}'_{C_1} + y'\mathbf{j}'_{C_1} + z'\mathbf{k}'_{C_1} + O'_{C_1}$$

We know that $[\mathbf{i}'_{C_1}, \mathbf{j}'_{C_1}, \mathbf{k}'_{C_1}, O'_{C_1}] = T([\mathbf{i}, \mathbf{j}, \mathbf{k}, O])$

$\boxed{C_2 = \mathbf{M}\, C_1}$

$$\begin{aligned} P_{C_1} &= x'T(\mathbf{i}) + y'T(\mathbf{j}) + z'T(\mathbf{k}) + T(O) \\ &= x'\mathrm{M}\mathbf{i} + y'\mathrm{M}\mathbf{j} + z'\mathrm{M}\mathbf{k} + \mathrm{M}O \\ &= x'\mathrm{M}\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + y'\mathrm{M}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} + z'\mathrm{M}\begin{bmatrix}0\\0\\1\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \\ &= \mathrm{M}\begin{bmatrix}x'\\0\\0\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\y'\\0\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\0\\z'\\0\end{bmatrix} + \mathrm{M}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \\ &= \mathrm{M}\left(\begin{bmatrix}x'\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0\\y'\\0\\0\end{bmatrix} + \begin{bmatrix}0\\0\\z'\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}\right) = \mathrm{M}\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix} \end{aligned}$$



C_2 = **M** C_1

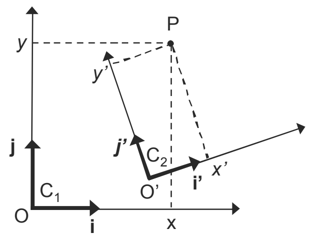
$\boxed{C_1 = \mathbf{M}^{-1}\, C_2}$

$$\boxed{P_{C_1} = \mathrm{M}P_{C_2}}$$

P in C_1 *vs* P in C_2

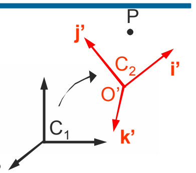
$$P_{C_1} = \mathrm{M}P_{C_2}$$

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



Transformations as a Change of Basis

We know the basis vectors and we know that



$$P_{C_1} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}P_{C_2}$$

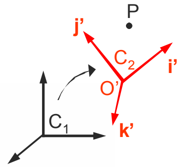
Now, what is M with respect to the basis vectors?

$$P_{C_2} = x'\mathbf{i}'_{C_2} + y'\mathbf{j}'_{C_2} + z'\mathbf{k}'_{C_2} + O'_{C_2} = x'\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + y'\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + z'\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$P_{C_1} = x'\mathbf{i}'_{C_1} + y'\mathbf{j}'_{C_1} + z'\mathbf{k}'_{C_1} + O'_{C_1} = x'\begin{bmatrix} i'_x \\ i'_y \\ i'_z \\ 0 \end{bmatrix} + y'\begin{bmatrix} j'_x \\ j'_y \\ j'_z \\ 0 \end{bmatrix} + z'\begin{bmatrix} k'_x \\ k'_y \\ k'_z \\ 0 \end{bmatrix} + \begin{bmatrix} O'_x \\ O'_y \\ O'_z \\ 1 \end{bmatrix}$$

$$P_{C_1} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} i'_x & j'_x & k'_x & O'_x \\ i'_y & j'_y & k'_y & O'_y \\ i'_z & j'_z & k'_z & O'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}P_{C_2}$$

Transformations as a Change of Basis



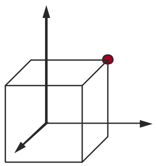
$$P_{C_1} = \mathrm{M}P_{C_2}$$

$$P_{C_1} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} i'_x & j'_x & k'_x & O'_x \\ i'_y & j'_y & k'_y & O'_y \\ i'_z & j'_z & k'_z & O'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}P_{C_2}$$

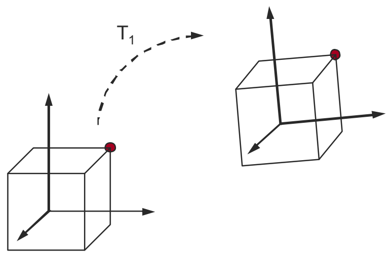
That is:

We can view transformations as a change of coordinate system

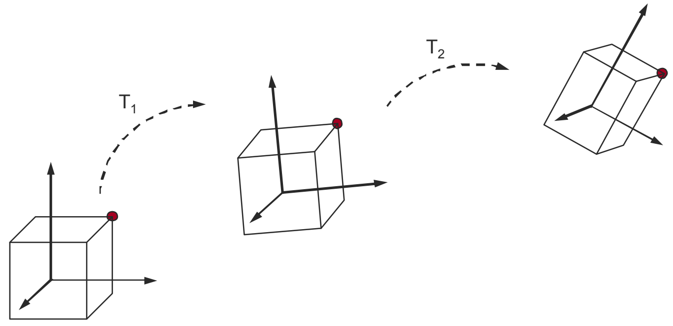
Transforming a Point by Transforming Coordinate Systems



Transforming a Point by Transforming Coordinate Systems

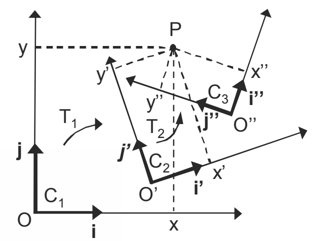


Transforming a Point by Transforming Coordinate Systems



Successive Transformations of the Coordinate System

$C_1 \rightarrow C_2 \rightarrow C_3$
$\quad T_1 \quad\;\; T_2$



Working backwards:

$$P_{C_2} = \mathrm{M}_2 P_{C_3} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}_2 \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

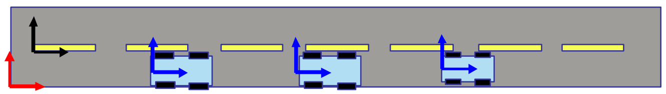
$$P_{C_1} = \mathrm{M}_1 P_{C_2} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathrm{M}_1 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \mathrm{M}_1 \mathrm{M}_2 \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix}$$

Lecture Outline

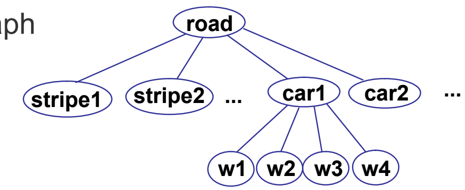
- 3D Affine Transforms
- Composite Transforms
  - Composite Rotations
  - Gimble lock
  - Rotation around an arbitrary axis
  - Composite Translations, Scales, Shears
- Transformation Interpretation:
  - Right to Left Interpretation: – Changing Location of Objects
  - Left to Right Interpretation: -- Changing the Coord. System
- Transformation Hierarchies

Transformation Hierarchies

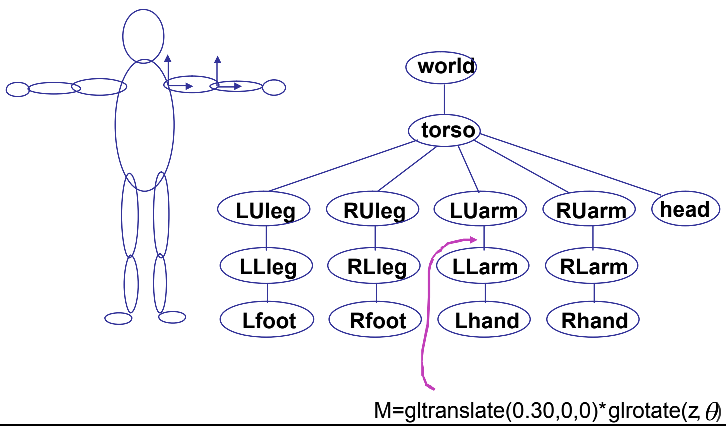
- scene may have a hierarchy of coordinate systems
  - stores matrix at each level with incremental transform from parent's coordinate system



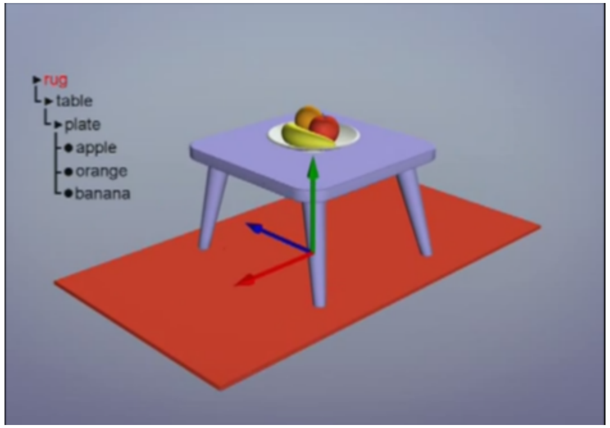
- scene graph



Transformation Hierarchy

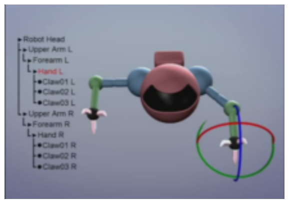


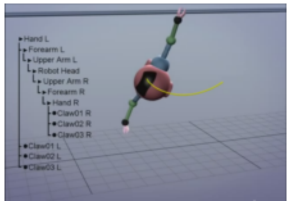
Hierarchies

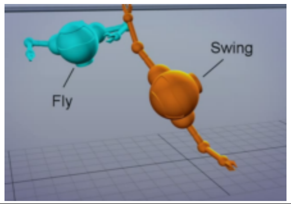


Hierarchies: Building a Robot

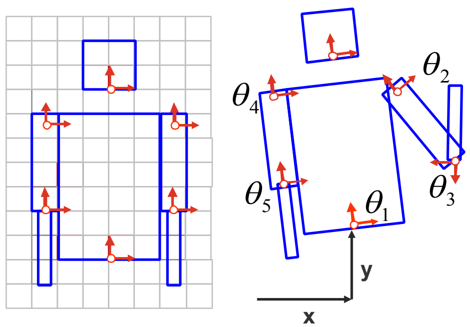
- Sometimes you need different hierarchies for different actions







Transformation Hierarchy Example 5




```
glTranslate3f(x,y,0);
glRotatef( θ1,0,0,1);
DrawBody();
glPushMatrix();
   glTranslate3f(0,7,0);
   DrawHead();
glPopMatrix();
glPushMatrix();
   glTranslate(2.5,5.5,0);
   glRotatef( θ2,0,0,1);
   DrawUArm();
   glTranslate(0,-3.5,0);
   glRotatef( θ3,0,0,1);
   DrawLArm();
glPopMatrix();
... (draw other arm)
```

Hierarchical Modelling

- Advantages
  - define object once, instantiate multiple copies
  - transformation parameters often good control knobs
  - maintain structural constraints if well-designed
- Limitations
  - expressivity: not always the best controls
  - can't do closed kinematic chains
    - keep hand on hip
  - can't do other constraints
    - collision detection
      - self-intersection
      - walk through walls