

Composition of Transformations

①

Scaling:

$$S_1 \circ S_2 = \underbrace{\begin{bmatrix} s_{x_1} & 0 & 0 \\ 0 & s_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S_1} \begin{bmatrix} s_{x_2} & 0 & 0 \\ 0 & s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x_1} s_{x_2} & 0 & 0 \\ 0 & s_{y_1} s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation:

$$R_2 \circ R_1 = \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 \\ \sin \theta_2 & \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2 & & \\ & & \\ & & \end{bmatrix} \text{ not enough space...}$$

$$= \begin{bmatrix} \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2 & -\sin \theta_1 \cos \theta_2 - \cos \theta_1 \sin \theta_2 & 0 \\ \sin \theta_1 \cos \theta_2 + \cos \theta_1 \sin \theta_2 & -\sin \theta_1 \sin \theta_2 + \cos \theta_1 \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

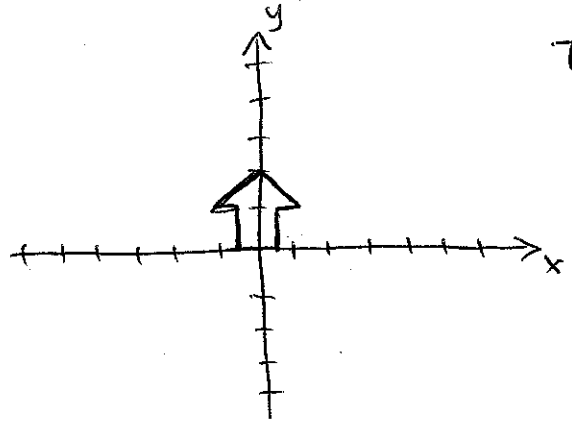
Remember back in high school:

$$\begin{aligned} \sin(A+B) &= \sin A \cos B + \cos A \sin B \\ \cos(A+B) &= \cos A \cos B - \sin A \sin B \end{aligned}$$

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

• Be very careful about the order of the transformations!

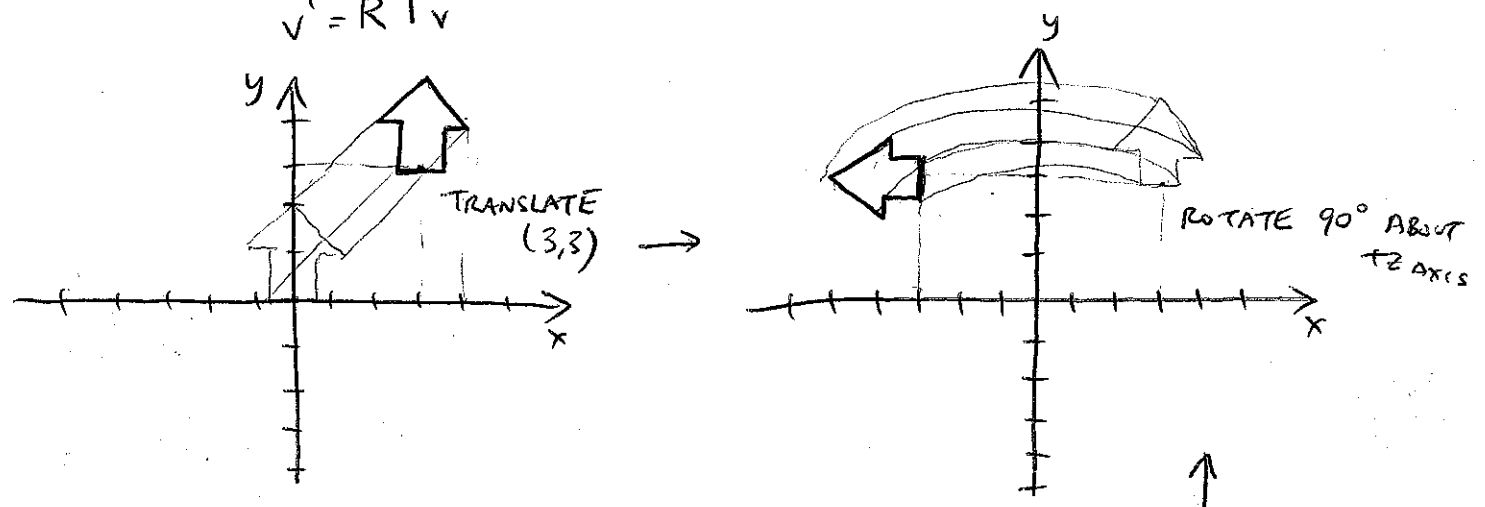
Ex:



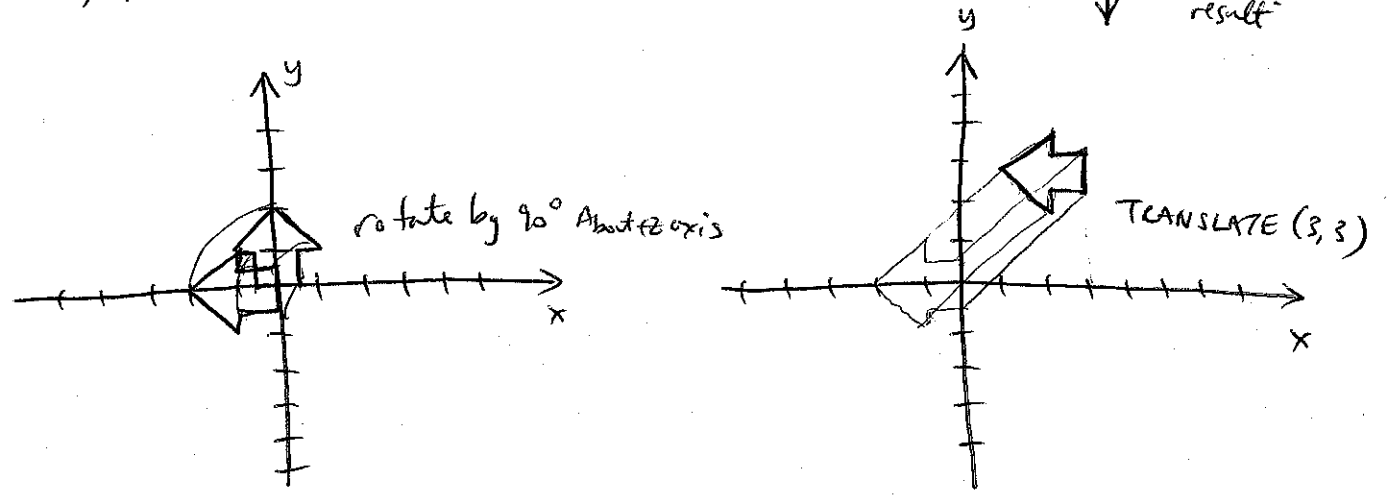
$T = \text{Translate } (3, 3)$
 $R = \text{Rotate } (90^\circ)$ ↻

1) First Translate, then rotate:

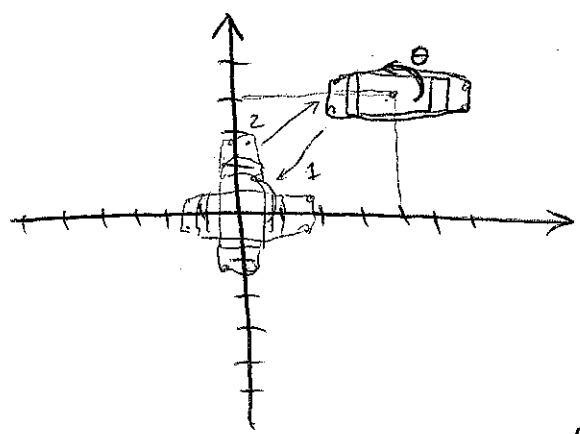
$$v' = RTv$$



2) First Rotate then translate:



How do you rotate about an arbitrary point?



Solution: first translate to origin, do your rotation, then translate back.

$$R_{(x_p, y_p)}(\theta) = T(x_p, y_p) R(\theta) T(-x_p, -y_p)$$

$$= \begin{bmatrix} 1 & 0 & x_p \\ 0 & 1 & y_p \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_p \\ 0 & 1 & -y_p \\ 0 & 0 & 1 \end{bmatrix}$$

rotation about arbitrary point (x_p, y_p)

Extension to 3D transformations:

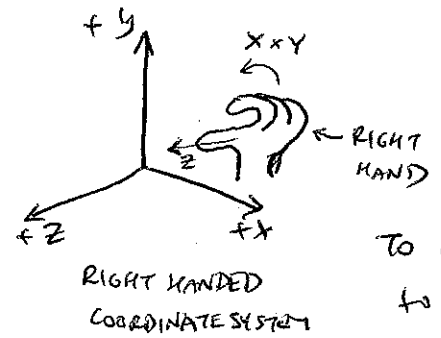
3D cartesian coordinates \rightarrow 4D homogeneous coordinates

$$\underbrace{(x, y, z)}_{\text{Euclidean space}} \rightarrow \underbrace{(x, y, z, w)}_{\text{Projective space}}$$

To Homogenize the point you still divide by w to get $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1)$ which projects the point to 3D Euclidean space ($w=1$)

\therefore Cartesian coordinates are the $w=1$ subspace in 4D projective space

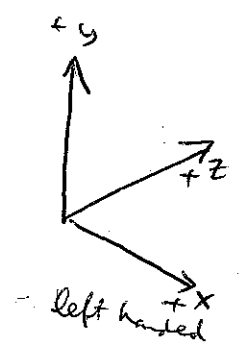
Right-Handed coordinate system



OPENGL IS RIGHT HANDED

DIRECT X IS LEFT HANDED

To convert from right handed to left handed (or vice-versa), simply flip the sign of Z :



$$\begin{bmatrix} L.H. \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R.H. \end{bmatrix}$$

3-D Transformations

Translation:

$$T(a, b, c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling:

$$S(a, b, c) = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations:

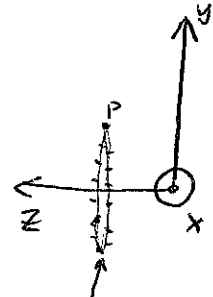
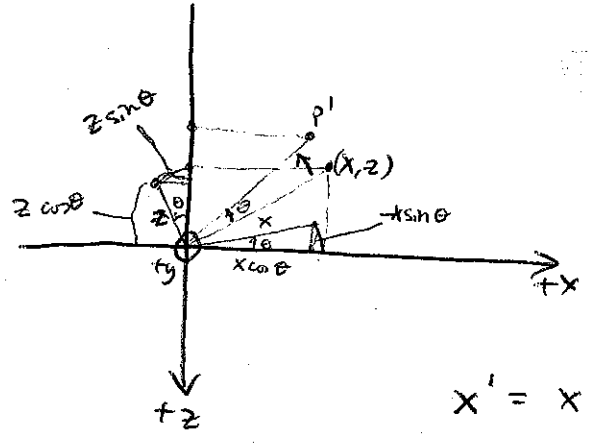
In 2D we had done rotation about the +z axis, so that's the same extended to 3D:

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

note how the z coordinate of the point does not change!

Makes sense if you look at it from the side

Rotate about +y:



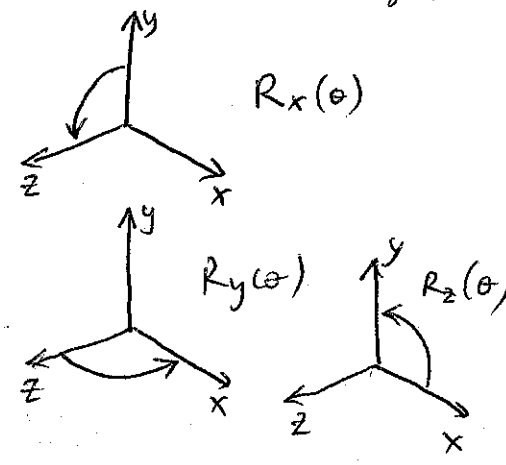
z value of P stays the same as you rotate!

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

$$z' = -x \sin \theta + z \cos \theta$$

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

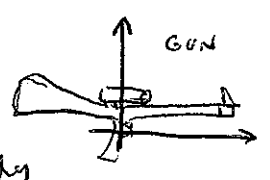
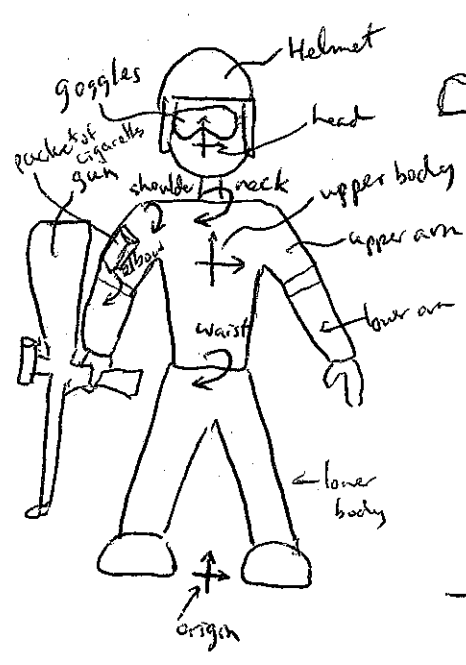
Relationships: (positive angle)



$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \leftarrow \text{know how to derive this!}$$

How do you rotate about an arbitrary axis?

A model hierarchy using a matrix stack



Transformations
Vertex on lower body to world:

$$\begin{bmatrix} \text{lower body vertex} \end{bmatrix} = \begin{bmatrix} \text{lower body to world} \end{bmatrix} \begin{bmatrix} \text{lower body vertex} \end{bmatrix}$$

Upper body to world:

$$\begin{bmatrix} \text{upper body vertex} \end{bmatrix} = \begin{bmatrix} \text{lower body to world} \\ \text{"dude to world"} \end{bmatrix} \begin{bmatrix} \text{Waist to lower body} \end{bmatrix} \begin{bmatrix} \text{Rotate waist} \end{bmatrix} \begin{bmatrix} \text{Translate to waist} \end{bmatrix} \begin{bmatrix} \text{upper body vertex} \end{bmatrix}$$

Head vertex to world:

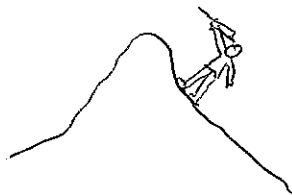
$$\begin{bmatrix} \text{head vertex in world} \end{bmatrix} = \begin{bmatrix} \text{upper body to world} \end{bmatrix} \begin{bmatrix} \text{neck to upper body} \end{bmatrix} \begin{bmatrix} \text{neck rotation} \end{bmatrix} \begin{bmatrix} \text{translate to neck} \end{bmatrix} \begin{bmatrix} \text{head vertex} \end{bmatrix}$$

Goggle vertex to world

$$\begin{bmatrix} \text{goggle in world space} \end{bmatrix} = \begin{bmatrix} \text{Head to world} \end{bmatrix} \begin{bmatrix} \text{translate goggles to head} \end{bmatrix} \begin{bmatrix} \text{goggle vertex} \end{bmatrix}$$

etc.

Using the matrix stack

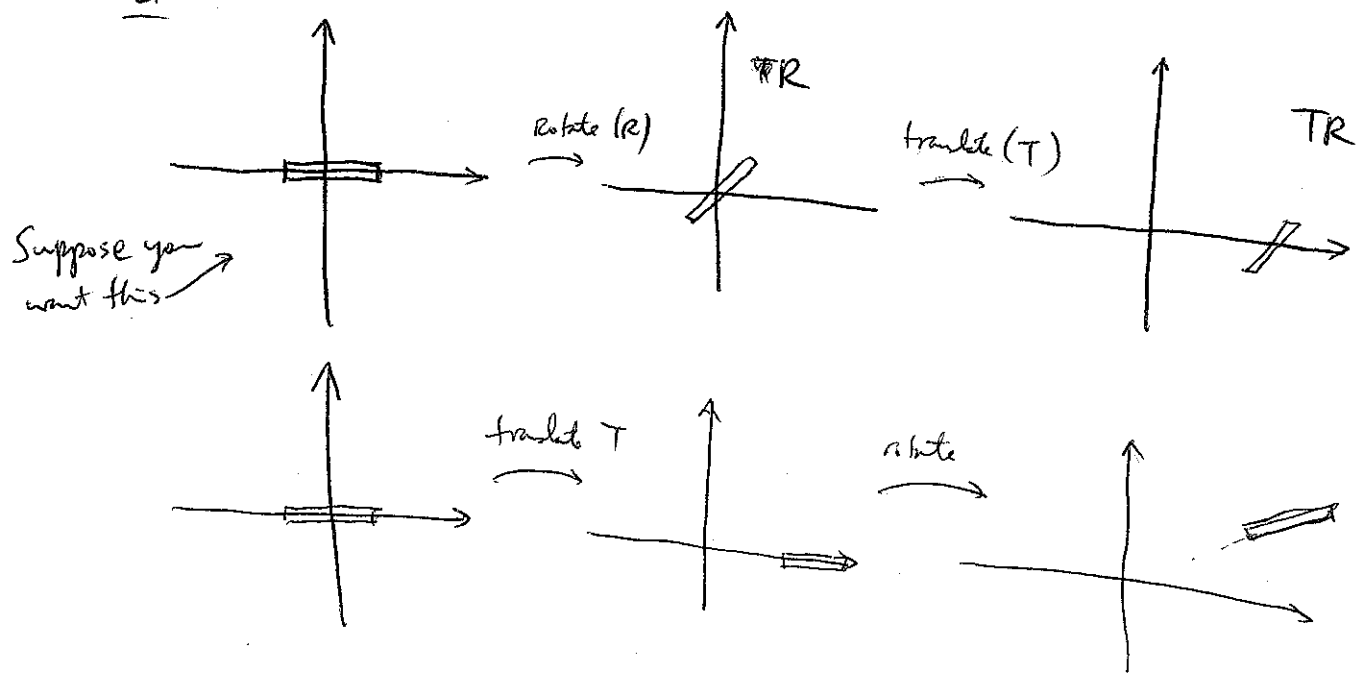


| OPERATION | STACK | Image |
|---|--|-------|
| Load I (world space) | → [I] | |
| ✓ Draw mountain | | |
| Concat Dude-to-World (D) | → [D] | |
| ✓ Draw lower body | | |
| Push | → [D] → [D] | |
| Concat Waist-to-lowerbody (W) | → [D · W] → [D] | |
| Concat Rotate waist (W _R) | → [D · W · W _R] → [D] | |
| Concat Waist to upperbody (U) | → [D · W · W _R · U] → [D] | |
| Push | → [D · W · W _R · U] → [D · W · W _R · U] → [D] | |
| Concat Neck-to-Upper-Body (N) | → [D · W · W _R · U · N] → [D · W · W _R · U] → [D] | |
| Concat Neck rotation (N _R) | → [D · W · W _R · U · N · N _R] → [D · W · W _R · U] → [D] | |
| Concat Head to neck (H) | → [D · W · W _R · U · N · N _R · H] → [D · W · W _R · U] → [D] | |
| Push | → [D · W · W _R · U · N · N _R · H] → [] | |
| Concat Goggles to Head (G) | → [D · W · W _R · U · N · N _R · H · G] | |
| Draw goggles | | |
| Pop | → [D · W · W _R · U · N · N _R · H] | |
| Push | → [D · W · W _R · U · N · N _R · H] → [] | |
| Concat Helmet to head (H _e) | → [D · W · W _R · U · N · N _R · H · H _e] | |
| Draw helmet | | |
| Pop | → [D · W · W _R · U · N · N _R · H] | |
| Draw Head | | |

Two ways of thinking about it

- 1. Grant, fixed coordinate system → ~~apply~~ ^{issue} transformations in reverse order that they are applied
- 2. Moving local coordinate system → issue transformations in the same order that they are applied

Ex:



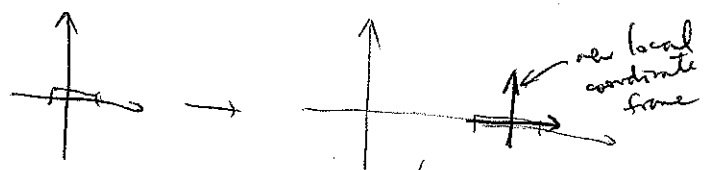
- 1. Grant fixed scheme (everything happens w/r to global origin)
 - 1. rotate
 - 2. translate

issue in reverse order:

- g1 Mult Matrix (T);
- g1 Mult Matrix (R);

- 2. Local reference frame (everything happens locally)

- 1. Translate



- 2. Rotate

Issue instructions in order

- g1 Mult Matrix (T);
- g1 Mult Matrix (R);

