Transforms (cont.)

4x4 matrix transforms are called homogeneous transformations because the operate on homogeneous coordinates

Notes: homogeneous transforms can be represented as 2x2 block matrices

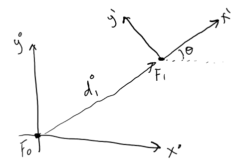
$$\left[\begin{array}{c|c} R & d \\ \hline 0 & 1 \end{array}\right]$$

homogeneous coordinates can be represented as 2x1 block matrices

points $\left[\begin{array}{c} p \\ \hline 1 \end{array}\right]$ and vector $\left[\begin{array}{c} v \\ \hline 0 \end{array}\right]$

We can combine these block representations the same as regular matrix + vector arithmetic

$\boxed{EX}$ Consider our frame of reference from last class



The transformation T_1^0 which converts coordinates from frame 1 (F_1) to frame $\emptyset$ (F_0) is

$$T_1^0 = \left[\begin{array}{c|c} R_1^0 & d_1^0 \\ \hline 0 & 1 \end{array}\right]$$

Suppose $R_1^0 = R_Z(45)$ and $d_1^0 = (5,2,0)^T$.
What is the location of the origin of F_1 w.r.t. F_0?

Aside: F_1 is an example of a local coordinate system

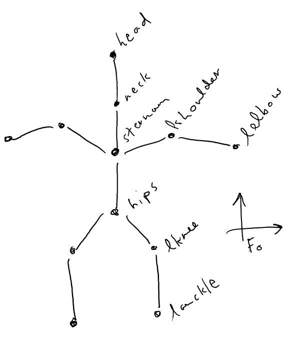
Let $p' = (0,0,0)^T$, then $p^0 = T_1^0 p'$

$$= \left[\begin{array}{c|c} R_1^0 & d_1^0 \\ \hline 0 & 1 \end{array}\right]\left[\begin{array}{c} p' \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} R_1^0 p' + d_1^0 \\ \hline 0 \cdot p' + 1 \end{array}\right] = \left[\begin{array}{c} R_1^0 p' + d_1^0 \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} R_Z(45)\begin{pmatrix}0\\0\\0\end{pmatrix} + \begin{pmatrix}5\\2\\0\end{pmatrix} \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} \begin{pmatrix}5\\2\\0\end{pmatrix} \\ \hline 1 \end{array}\right]$$

What is the position of $p' = (1,0,0)^T$ w.r.t. F_0?

$$p^0 = T_1^0 p' = \left[\begin{array}{c} R_1^0 p' + d_1^0 \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} R_Z(45)\begin{pmatrix}1\\0\\0\end{pmatrix} + \begin{pmatrix}5\\2\\0\end{pmatrix} \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} \begin{pmatrix}\sqrt{2}/2\\ \sqrt{2}/2\\0\end{pmatrix} + \begin{pmatrix}5\\2\\0\end{pmatrix} \\ \hline 1 \end{array}\right] = \begin{bmatrix} \frac{\sqrt{2}}{2} + 5 \\ \sqrt{2}/2 + 2 \\ 0 \\ \hline 1 \end{bmatrix}$$

Articulated Characters



→ Represented as a hierarchy of joints (aka bones)
→ hierarchy is called a skeleton
→ root of the hierarchy is usu. the hips
→ the root transform is usu. w.r.t. to the world transform

→ each joint stores a transform w.r.t to its parent

→ joints with no children are called end effectors

Degrees of Freedom (DOF): the "# ways" an object can move

$\boxed{EX}$ the "hips" have 6 DOF: translation in XYZ
rotation around X Y Z axes

$\boxed{EX}$ the "lknee" might have 3 DOF: rotation around XYZ

Joints: Several types of joints

1. Ball joint (ex. Shoulder) ← all joints in our class are these (3 DOF, rotation around XYZ)
2. Hinge joint (ex. elbow) ← 1 DOF joint (rotational)
3. Prismatic joint (ex. selfie stick) ← 1 DOF (translational)

Implementation details (base code):

Skeleton maintains tree of joints

Each joint stores

2 transforms: ① local2parent (F_i^j): converts from the joint's frame to its parent frame

② local2global (F_i^0) converts from the joint's frame to the global frame

name (ex. "hips")

id (0, 1, 2, ..., numJoints-1)

ptr to parent

list of children ptrs.

Kinematics: refers to the motion of objects over time

Forward Kinematics: Process of computing global positions of each joint given the current state of the skeleton.

The current state of a skeleton is called a "pose".

Idea: A pose contains a value for every DOF in the character.

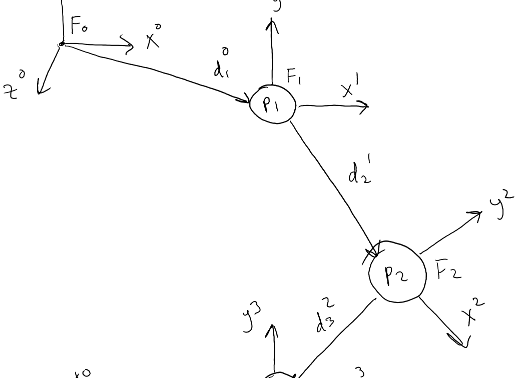
$\boxed{EX}$ The above skeleton has 6 + 11*3 DOFs (root + other joints)

Therefore, our pose have 39 values in it

$$\Theta = [\underbrace{root_x, root_y, root_z}_{\text{root translation}}, \underbrace{root\,\theta_x, root\,\theta_y, root\,\theta_z}_{\text{root rot}}, \underbrace{lknee\,\theta_x, lknee\,\theta_y}, \dots]$$

all values in local coordinates

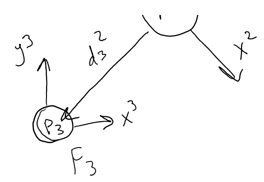
$\boxed{EX}$ Kinematic Chain



Goal: Compute the global positions of P_1, P_2, P_3 given the local transforms at P_1 P_2 P_3, e.g. T_1^0, T_2^1, T_3^2

Approach: Compute T_1^0, T_2^0, and T_3^0 to get the global positions of P_1 P_2 & P_3.

of p_1 p_2 is.



p_1 wrt to F_0

$$P_1^0 = T_1^0 p_1^1 \quad \text{, where } p_1^1 = (0,0,0,1)^T$$

$$P_2^0 = T_2^0 p_2^2 = T_1^0 T_2^1 p_2^2 \leftarrow p_2^2 = (0,0,0,1)^T$$

$$P_3^0 = T_3^0 p_3^3 = T_1^0 T_2^1 T_3^2 p_3^3$$

$\boxed{EX}$ Suppose $d_1^0 = (1,-1,1)^T$ $\quad R_1^0 = R_z(-45)$

$d_2^1 = (2,0,0)^T$ $\quad R_2^1 = R_z(-30)$

$d_3^2 = (2,-1,0)^T$ $\quad R_3^2 = I$

ϕ is a 3×1 vector

$$P_3^0 = T_3^0 p_3^3 = T_1^0 T_2^1 T_3^2 p_3^3$$

$$= \left[\begin{array}{c|c} R_1^0 & d_1^0 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} R_2^1 & d_2^1 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} R_3^2 & d_3^2 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c} \phi \\ \hline 1 \end{array}\right]$$

$$= \left[\begin{array}{c|c} R_1^0 R_2^1 & R_1^0 d_2^1 + d_1^0 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} R_3^2 & d_3^2 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c} \phi \\ \hline 1 \end{array}\right]$$

$$= \left[\begin{array}{c|c} R_1^0 R_2^1 R_3^2 & R_1^0 R_2^1 d_3^2 + R_1^0 d_2^1 + d_1^0 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c} \phi \\ \hline 1 \end{array}\right]$$

$=$ do later

<u>Forward Kinematics Alg</u> (FK): generalizes process we just did w/ the kinematic chain to any skeleton

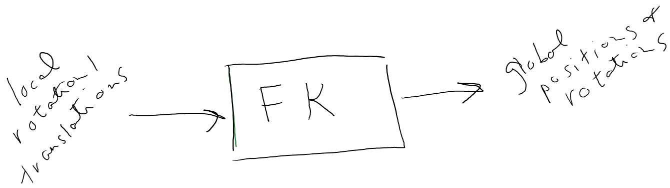
<u>FK</u>: Recursively compute local2global F_j^0 for each joint j

<u>Idea</u>: Do a tree traversal on each joint. Start at root

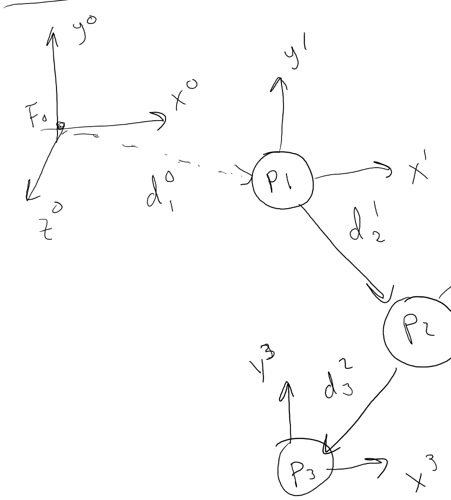
```
Joint :: fk()
  if (mParent != NULL)
    local2global = (mParent -> local2global) * local2parent
  else
    local2global = local2parent;  //root
  ...
```

F_i^0 = mParent -> local2global; F_j^i = local2parent

for each child
child → fk()



Kinematic Chain Revisited



Assume we are given local displacements & rotations at each joint, e.g.

d_1^0, d_2^1, d_3^2 ← displacement / offset / translation

R_1^0, R_2^1, R_3^2 ← orientations (usu euler angles, or quats, but 3x3 matrices here)

Q: What is the local 2 global transform, T_1^0, for joint 1?

$$T_1^0 = \left[\begin{array}{c|c} R_1^0 & d_1^0 \\ \hline 0 & 1 \end{array}\right]$$

Q: What is the local 2 global transform of joint 3?

$$T_3^0 = T_1^0 \; T_2^1 \; T_3^2 = \left[\begin{array}{c|c} R_1^0 & d_1^0 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} R_2^1 & d_2^1 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} R_3^2 & d_3^2 \\ \hline 0 & 1 \end{array}\right]$$

Q: What is the coordinate of p_3^3 in joint 2's frame?

position of joint 3 w.r.t frame 3

$$p_3^2 = T_3^2 \; p_3^3$$

$$= \left[\begin{array}{c|c} R_3^2 & d_3^2 \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c} 0 \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} R_3^2 \cdot \phi + d_3^2 \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} d_3^2 \\ \hline 1 \end{array}\right]$$

Q: What is the coordinate p_2^2 in joint 3's frame?

$$p^3 = T^3 \; p^2 = (T_3^2)^{-1} \; p_2^2$$

$$p_2^3 = \underbrace{T_2^3}_{\substack{\text{don't have this} \\ \text{but have } T_3^2}} p_2^2 = (T_3^2)^{-1} p_2^2$$

$$\text{where } (T_3^2)^{-1} = \left(\left[\begin{array}{c|c} R_3^2 & d_3^2 \\ \hline 0 & 1 \end{array} \right] \right)^{-1}$$

$$= \left(\begin{array}{c|c} (R_3^2)^T & -(R_3^2)^T d_3^2 \\ \hline 0 & 1 \end{array} \right)$$

$$\left[\begin{array}{c|c} R_3^2 & d_3^2 \\ \hline 0 & 1 \end{array} \right] \left[\begin{array}{c|c} A & b \\ \hline 0 & 1 \end{array} \right] = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & 1 \end{array} \right]$$

$$\rightarrow \left[\begin{array}{c|c} R_3^2 A & R_3^2 b + d_3^2 \\ \hline 0 & 1 \end{array} \right]$$

$\rightarrow$ Let $A = (R_3^2)^T$

$$R_3^2 b + d_3^2 = 0$$

$$\rightarrow R_3^2 b = -d_3^2$$

$$\rightarrow b = (R_3^2)^T (-d_3^2)$$

Q: What is the world origin w.r.t. joint 3's frame?

$$p_0^3 = \underbrace{T_0^3}\, p_0^0 = (T_0^3)^{-1} p_0^0$$

$$= \left[\begin{array}{c|c} (R_1^0 R_2^1 R_3^2)^{-1} & (R_1^0 R_2^1 R_3^2)^{-1} (-R_1^0 R_2^1 d_3^2 - R_1^0 d_2^1 - d_1^0) \\ \hline 0 & 1 \end{array} \right]$$

$$= \left[\begin{array}{c|c} R_2^3 R_1^2 R_0^1 & -R_2^3 d_3^2 - R_2^3 R_1^2 d_2^1 - R_2^3 R_1^2 R_0^1 d_1^0 \\ \hline 0 & 1 \end{array} \right]$$

Skeletons & Joints Revisited

Recall: A pose Θ is a vector of values corresponding to the DOF of the skeleton.

$\boxed{\text{Ex}}$ A biped w 6 DOF at the root & 3 at all other joints

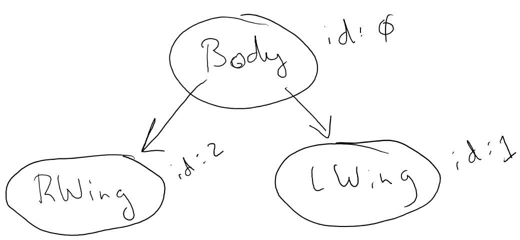
$$\Theta = \left(x_0^0\ y_0^0\ z_0^0\ \theta_{0x}^0\ \theta_{0y}^0\ \theta_{0z}^0,\ \underbrace{\theta_{1x}\ \theta_{1y}\ \theta_{1z}}_{\text{joint 1}}, \ldots, \underbrace{\theta_{Nx}\ \theta_{Ny}\ \theta_{Nz}} \right)$$

$$\Theta = \Big(\underbrace{x_0^0\; y_0^0\; z_0^0}_{\substack{\text{translation}\\ \text{joint } \phi,\\ \text{wrt to}\\ \text{world}}}\;\underbrace{\theta_{0x}^0\;\theta_{0y}^0\;\theta_{0z}^0}_{\substack{\text{rotation}\\ \text{joint } \phi\\ \text{w.rt world}}},\;\underbrace{\theta_{1x}\;\theta_{1y}\;\theta_{1z}}_{\text{rotation joint 1}},\;\ldots,\;\underbrace{\theta_{Nx}\;\theta_{Ny}\;\theta_{Nz}}_{\text{joint } N}\Big)$$

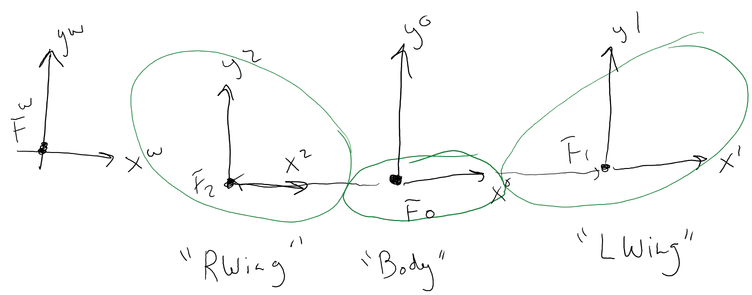
To animate an articulated character (e.g. one w/ poseable limbs), we copy the pose into the transform of each joint & call FK

$\boxed{\text{Example}}$ Butterfly

Skeleton Hierarchy:



Define how each body part relates to the other



Goal: Associate geometry w/ each transform.

Common Approach: Scene Graph

→ a tree data structure that represents an entire scene, (light, primitives, etc.)

→ nodes represent scene objects

→ each node has a transform

→ edges represent parent-child frame relationships

Matrix Stack:

$\boxed{\text{EX}}$

```
push()
translate(vec3(4,4,0)); ← translate by (4,4,0)
rotate(π/4, vec3(0,0,1)) ← rotate 45° around z
scale(vec3(2,0.5,1)); ← scale x by 2 & y by 1/2
drawSquare(...) ← draw unit square
pop()
```

2x3 is scale

a unit square has points: ... $-1/2$ 1 0 pop()

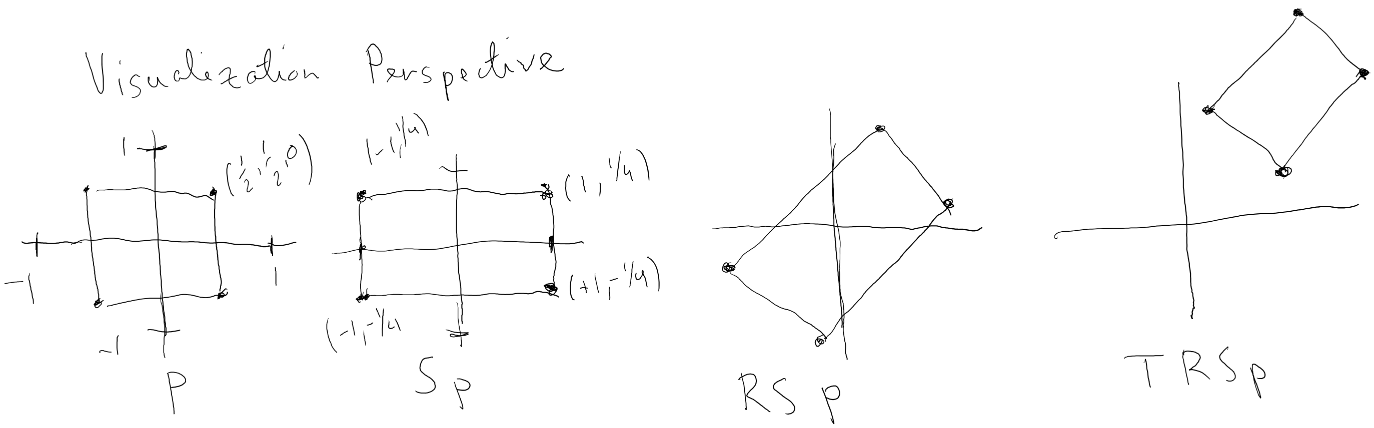
Math Perspective:

$$\underbrace{\left[\begin{array}{c|c} I & \begin{pmatrix}4\\4\\0\end{pmatrix} \\ \hline 0 & 1 \end{array}\right]}_{T} \underbrace{\left[\begin{array}{c|c} R_z(45) & 0 \\ \hline 0 & 1 \end{array}\right]}_{R} \underbrace{\left[\begin{array}{ccc|c} 2 & 0 & 0 & \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & \\ \hline & 0 & & 1 \end{array}\right]}_{S} p$$

3x3 is the scale

a unit points: ~
$[-1/2, -1/2, 0]$
$[-1/2, 1/2, 0]$
$[1/2, 1/2, 0]$
$[1/2, -1/2, 0]$

Visualization Perspective



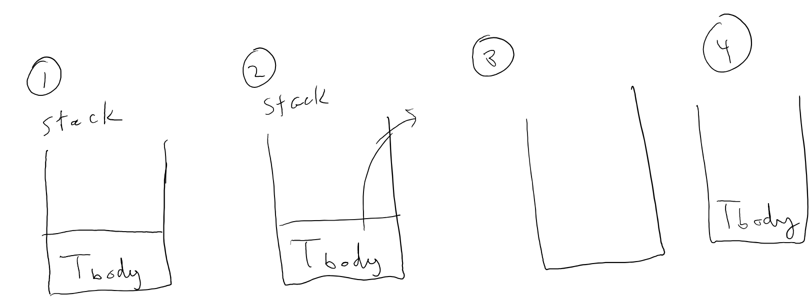
(local)

push() ⟵ pushes the current product of transforms to a stack

pop() ⟵ pops the top transform from the stack

EX
0. transform(body)
1. push()
2. transform(arm)
   drawArm()
3. pop()
4. push()
5. transform(leg)
   drawLeg()
6. pop()

0. currentTransform = T_{body}



currentTransform = T_{body}

currentTransform = $T_{body} T_{arm}$

currentTransform = T_{body}
(pop() sets currentTransform to top of the stack)

Motion:

Character Motions is a series of poses over time

recall: poses denoted Θ

Keyframed Motion: $\langle t_0, \Theta_0 \rangle, \langle t_1, \Theta_1 \rangle, \dots \langle t_n, \Theta_2 \rangle$

- times may not be uniformly spaced
- relies on artist typically to create each pose Θ_i
- use splines to interpolate poses

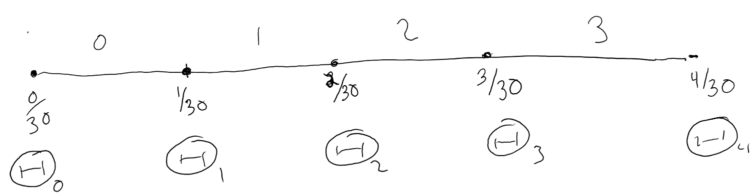
Fixed framerate Motion: $[\Theta_0, \Theta_1, \Theta_2, \dots\dots \Theta_N]$

- don't store time because the time between each key is known
- motion capture produced fixed framerate motion

$\boxed{EX}$ Motion Capture systems usu. capture poses at either 24, 30, 120 fps.

$\Rightarrow$ If the fps = 24, the time between each frame is $\frac{1}{24}$ s

$\boxed{EX}$ Suppose we have a 30 fps motion that is 2s long.
What is the pose at time 0.1 s?



Step 1: Find segment containing time

$\rightarrow$ Each segment corresponds to $\frac{1}{30}$ seconds

$\Rightarrow \Delta t = 1/30$

$\Rightarrow$ segment $= \text{floor}(0.1/\Delta t) = \text{floor}(3) = 3$

Step 2: Compute normalized time, u

Step 2: Compute normalized time, u

StartTime for segment = segment ID * Δt = $3\left(\frac{1}{30}\right) = \frac{3}{30} = 0.1$

Segment end time = (segment + 1) * Δt = $4\left(\frac{1}{30}\right) = 4/30$

$$\text{normalizedTime} = \frac{\text{time} - \text{segmentStartTime}}{\text{SegmentEndTime} - \text{segmentStartTime}} = \frac{0.1 - 0.1}{\Delta t} = 0$$

Step 3: Interpolate

Θ = Interpolate(Θ_3, Θ_4, 0)

$\boxed{EX}$ How can I play a motion twice as fast?

Approach 1: Resample a motion to have a duration, or change fps

Approach 2: During playback, you can use a "time scale" to play at different speeds w/out changing the motion, e.g.

```
update()
   time = elapsedTime();
   Θ = motion.getValue(time);
```

play at recorded speed

```
update()
   time = elapsedTime() * timeScale;
   Θ = motion.getValue(time);
```

scales time

Motion Editing:

In practice, we have motion clips for walk, stand, anything we want our character to do

Problem: We can't create motion clips for every possibility

→ too labor intensive

→ often impossible: needs special equipment, full knowledge about environments / context where motions would be used

Sol: (1) Generate new motions from existing ones

—

ex. Greeting motion + sad motion = sad greeting
ex. blending between motion clips to create transitions

2 Adopt motion clips to new settings
ex. a walk motion can be modified for uneven terrain
ex. holding a cup ; opening a door

Approach: To edit a motion, we only need to edit its keys

Editing poses:

Technige 1: Freezing a joint. (etting a constant value for a joint)

$\boxed{\text{EX}}$ Zombie Arms.

→ replace the rotation curve for the shoulders to have a constant value

Technique 2: Splicing

→ Copy upper body joints from one motion & paste them onto an- other motion

$\boxed{\text{EX}}$ Splicing a "drink water" motion onto the upper body of a walk motion