

## hierarchical transformations

- we will have a robot with moving limbs
- we will draw each part as a squashed unit sphere
- our end goal is to have the correct part frame  $\vec{\mathbf{f}}^t = \vec{\mathbf{w}}^t F$
- and then  $MVM = E^{-1}F$
- $\vec{\mathbf{f}}^t$  will not be an orthonormal frame!

## start with center body

- suppose  $\vec{\mathbf{o}}^t = \vec{\mathbf{w}}^t O$  is an orthonormal frame centered at the center of the body with one axis, say  $y$  pointing up to the neck, and  $x$  pointing to the robot's right
- for an appropriate xy-translation  $T_0$  wrt  $\vec{\mathbf{o}}^t$  we have a frame  $\vec{\mathbf{b}}^t = \vec{\mathbf{w}}^t O T_0$  centered at the center of the right shoulder of the body

## at the shoulder

- suppose we have stored two euler angles  $x_0, y_0$ , describing the rotation of the sholder, wrt  $\vec{\mathbf{b}}^t$
- then  $\vec{\mathbf{c}}^t = \vec{\mathbf{w}}^t O T_0 R_{y_0} R_{x_0}$  is a frame at the shoulder with  $-y$  pointing down the upper arm

## at the beginning of upper arm

- the shoulder is the same as the beginning of the upper arm
- for an appropriate -y-translation  $T_1$  wrt  $\vec{\mathbf{c}}^t$  we have a frame at the center of the upper arm  $\vec{\mathbf{d}}^t = \vec{\mathbf{w}}^t O T_0 R_{y_0} R_{x_0} T_1 = \vec{\mathbf{w}}^t D$

## at the center of upper arm

- we use an appropriate non-uniform scale  $S_1$  wrt  $\vec{\mathbf{d}}^t$  so that a canonical unit sphere connects to the shoulder, and has desired width  $\vec{\mathbf{f}}^t = \vec{\mathbf{w}}^t O T_0 R_{y_0} R_{x_0} T_1 S_1 = \vec{\mathbf{w}}^t F$
- we can use  $F$  as the part's object matrix to create the MVM.

## orthonormal to continue

- suppose we now wanted to go to the forearm
- we can continue where we left off
- but we do not want to use  $\vec{\mathbf{f}}^t$ , which is not orthonormal
  - since rotation matrices will not have rotational effect
- so we should continue on with  $\vec{\mathbf{d}}^t$  (or  $\vec{\mathbf{c}}^t$ )
- then reading left to right, we will translate, rotate, translate, scale to get the part's frame.

## picking

- set `glReadBuffer(GL_BACK);`
- draw to back buffer
- code the id of each part as the vertex colors
- use flat shading
- read the pixel under the mouse from the buffer
- do not swap the buffers