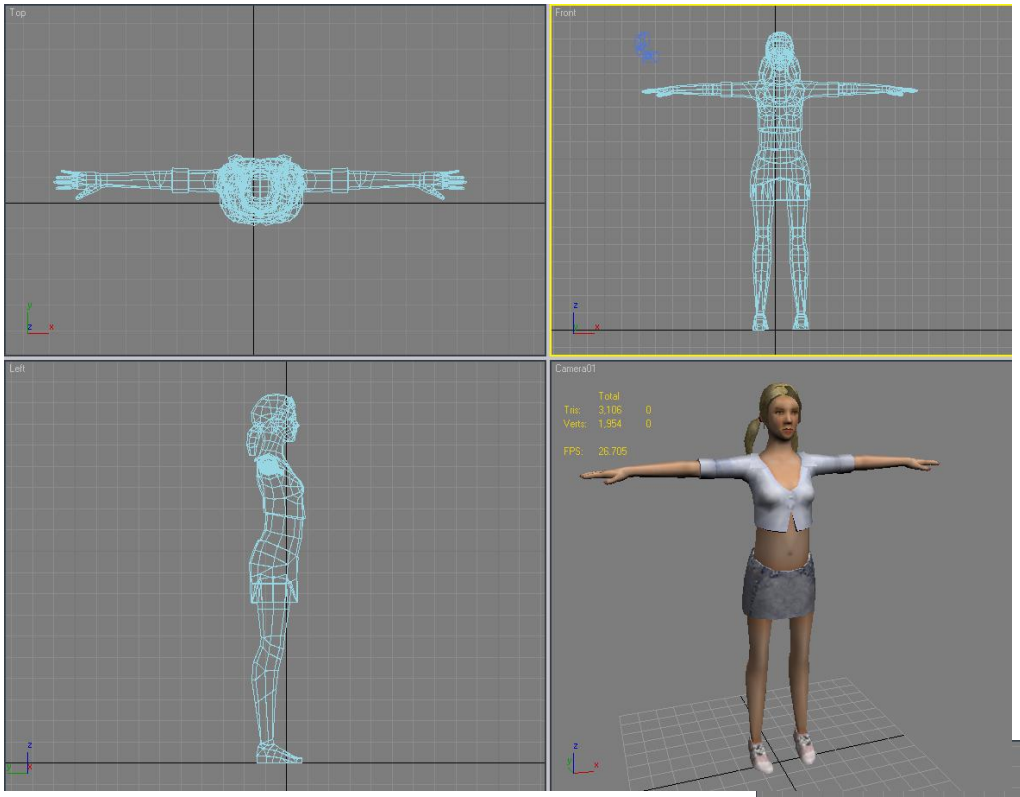
Rigging / Skinning

based on

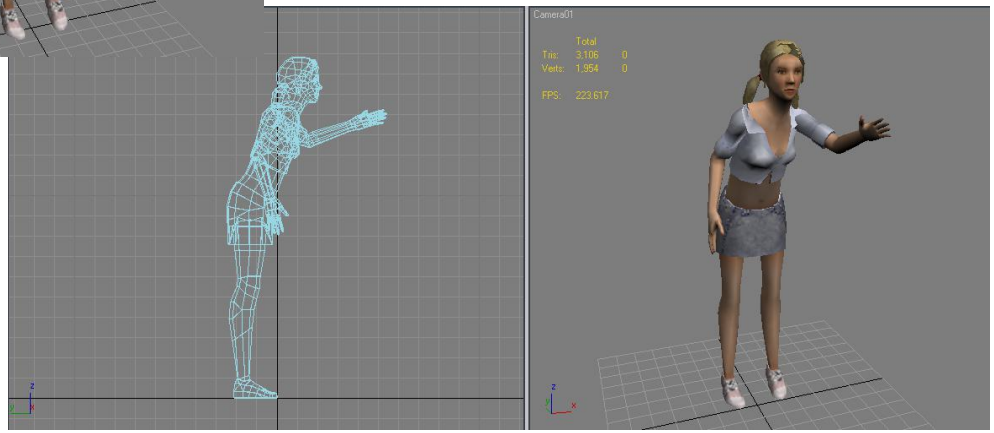
Taku Komura, Jehee Lee and

Charles B.Own's slides

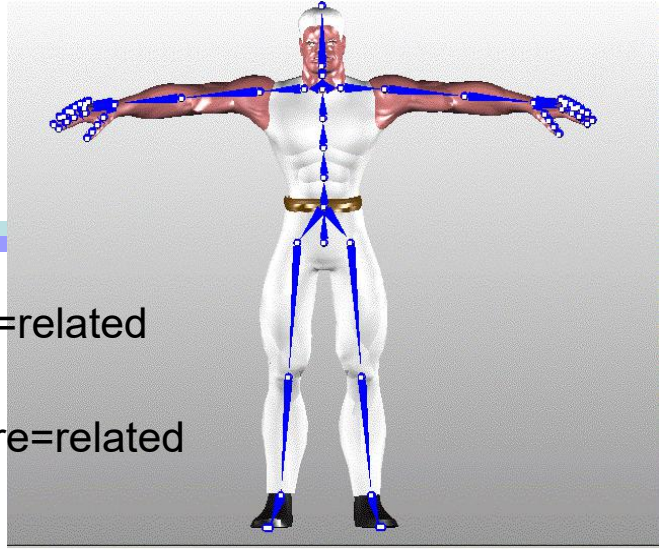
Skeletal Animation



Victoria



Skinning



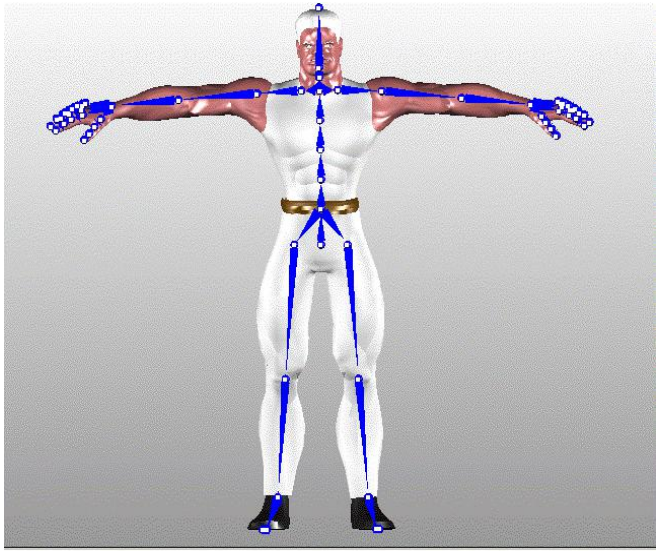
http://www.youtube.com/watch?v=dniWVu55PEc&feature=related

http://www.youtube.com/watch?v=z0QWBdk8MCA&feature=related

- Assuming the skin mesh is given
- 1. Kinematics : How the motions of the skeleton are given
- 2. Skinning : How the character's skin deform according to the motion of the skeleton

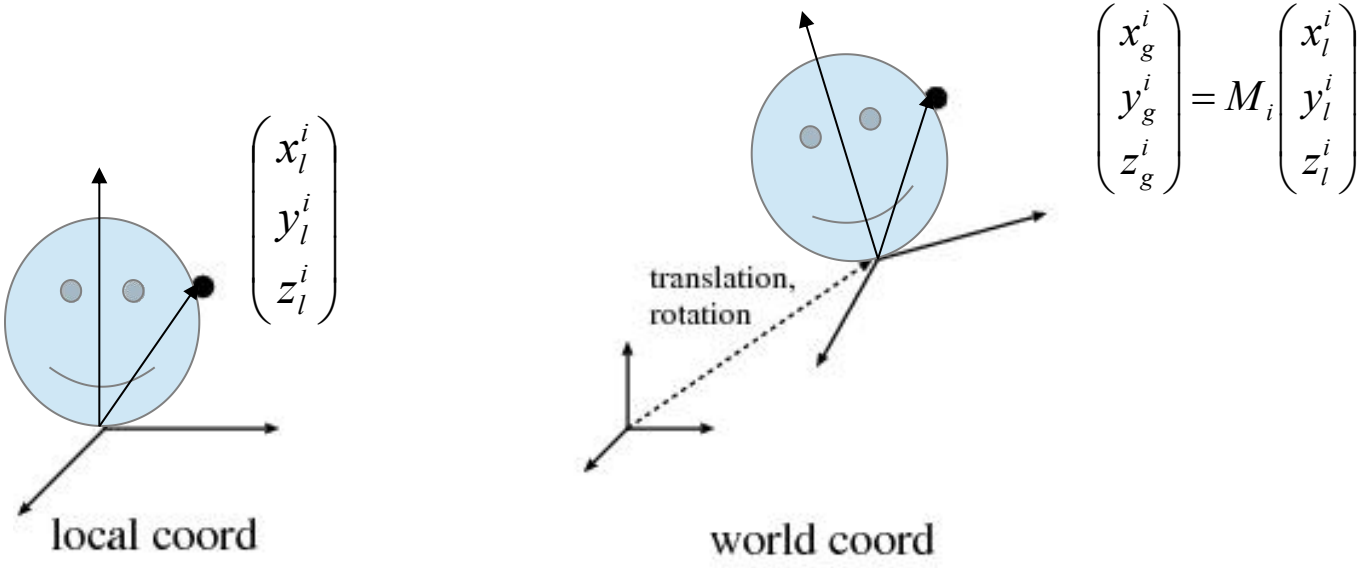
Kinematics

- How to animate skeletons (articulated figures)
- ***Kinematics*** is the study of motion (without regard to the forces that caused it)

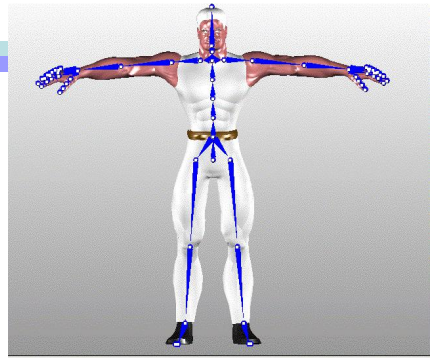


Backgrounds

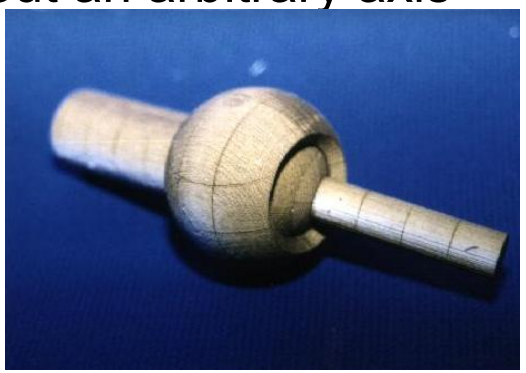
- A transformation matrix (4x4) is defined per bone that converts local coordinates to world coordinates



Hierarchical Models

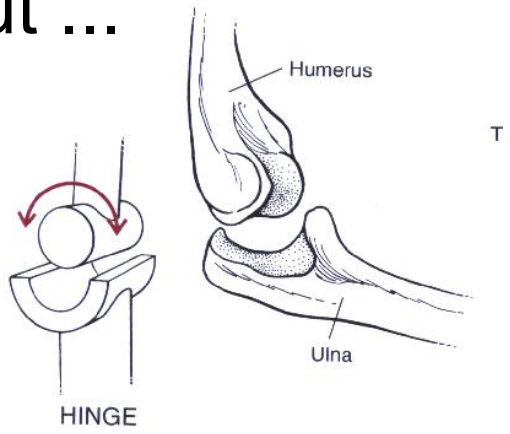


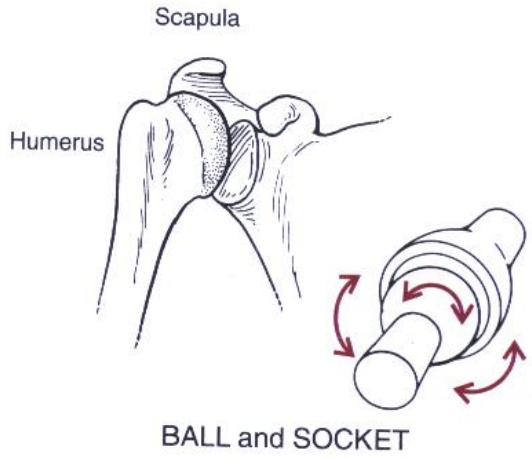
- Tree structure of joints and bones
  - The root bone can be chosen arbitrarily
  - Usually the pelvis bone is the root bone
- A Joint connect two bones
  - Revolute (hinge) joint allows rotation about a fixed axis
  - Prismatic joint allows translation along a line
  - Ball-and-socket joint allows rotation about an arbitrary axis

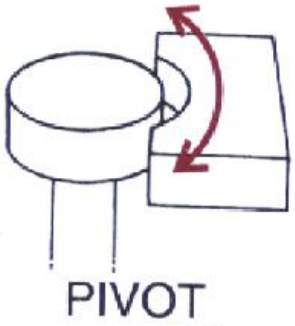


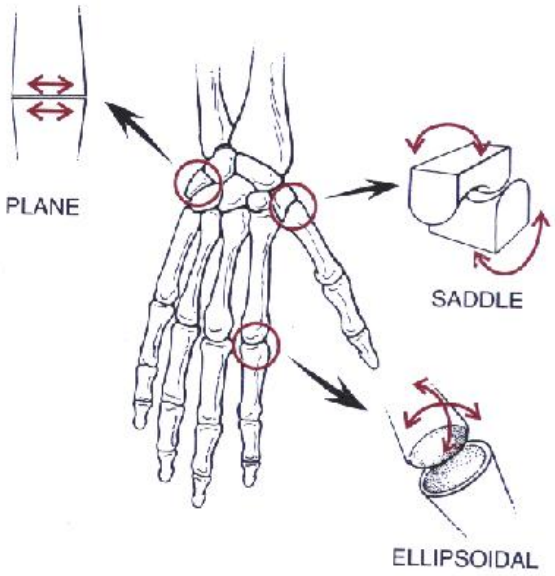
Human Joints

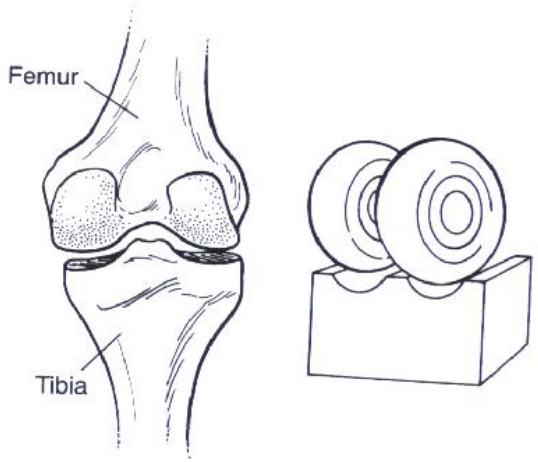
- Human joints are actually much more complicated, but ...

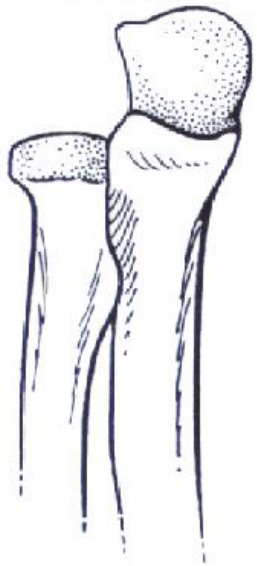




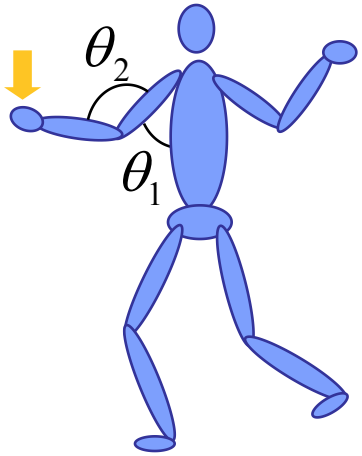






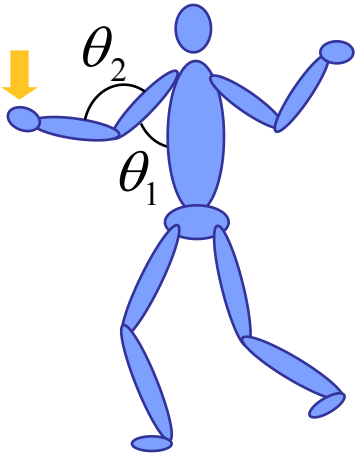


Forward Kinematics



$$(\mathbf{p},\mathbf{q}) = \mathrm{F}(\theta_i)$$

Forward Kinematics

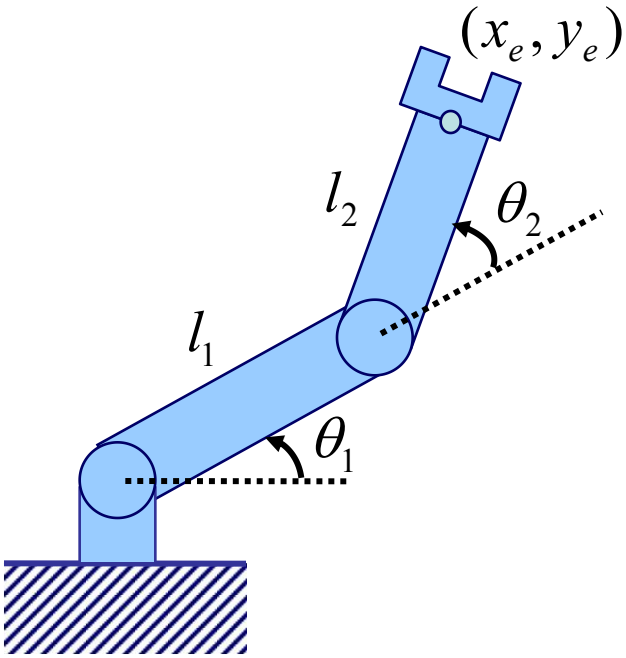


$$\theta_i = \mathrm{F}^{-1}(\mathbf{p},\mathbf{q})$$

Inverse Kinematics

Forward Kinematics: A Simple Example

- A simple robot arm in 2-dimensional space
  - 2 revolute joints
  - Joint angles are known
  - Compute the position of the end-effector



$$x_e = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y_e = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

3D Transformations

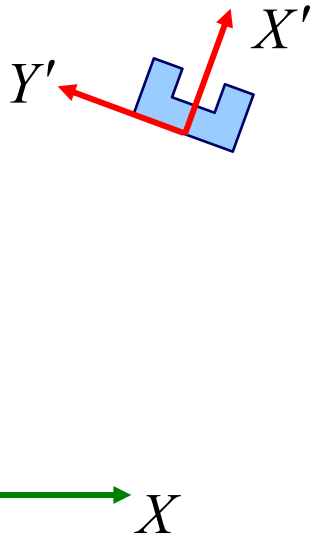
- The position and orientation of an object is represented as a rigid transformation

- Vector & 3x3 Matrix

$$\mathbf{Rv} + \mathbf{p}$$

- 4x4 Matrix

$$\mathbf{Tv} = \begin{pmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

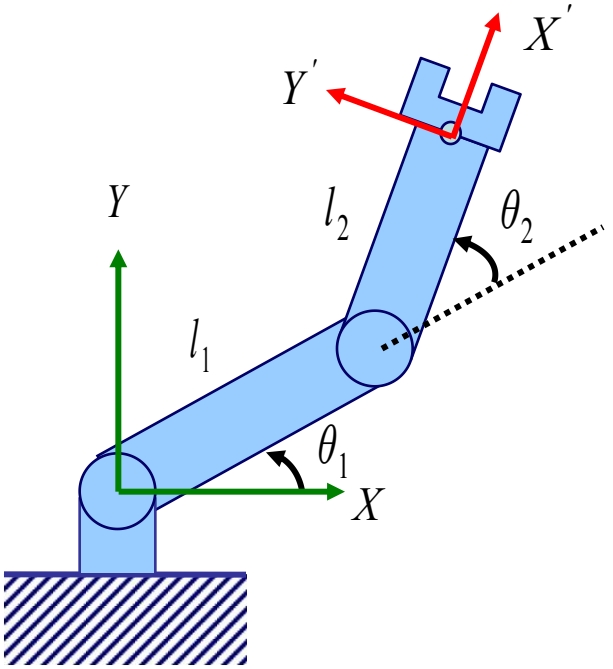


- Vector & Quaternion

$$\mathbf{q}\mathbf{v}\mathbf{q}^{-1} + \mathbf{p}$$

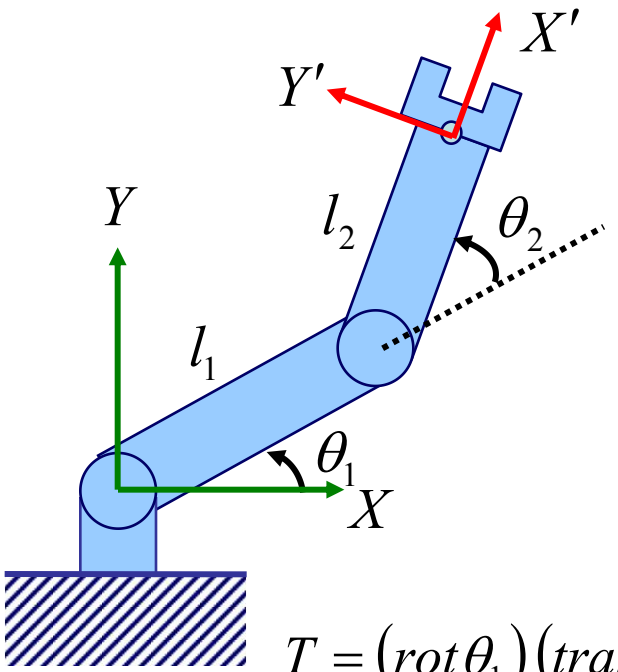
Forward Kinematics: A Simple Example

- Forward kinematics map as a coordinate transformation
  - that transforms the position and orientation of the end-effector according to joint angles



$$\begin{pmatrix} x_e \\ y_e \\ 1 \end{pmatrix} = \begin{pmatrix} & T & \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

A Chain of Transformations



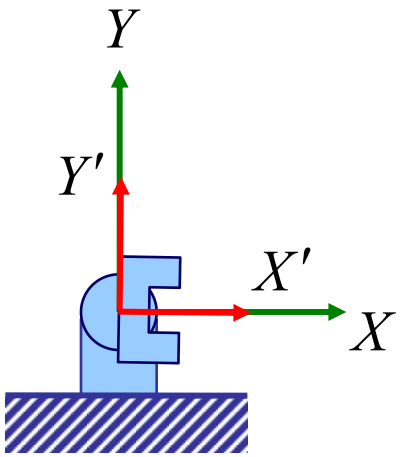
$$\begin{pmatrix} x_e \\ y_e \\ 1 \end{pmatrix} = \begin{pmatrix} & T & \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

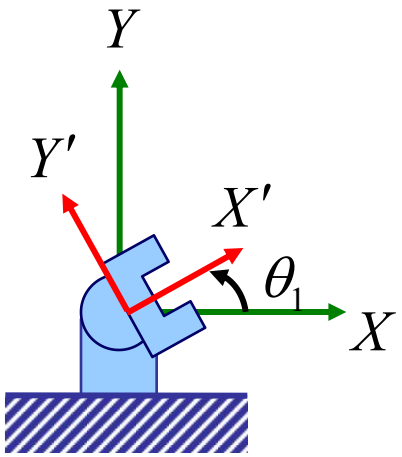


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

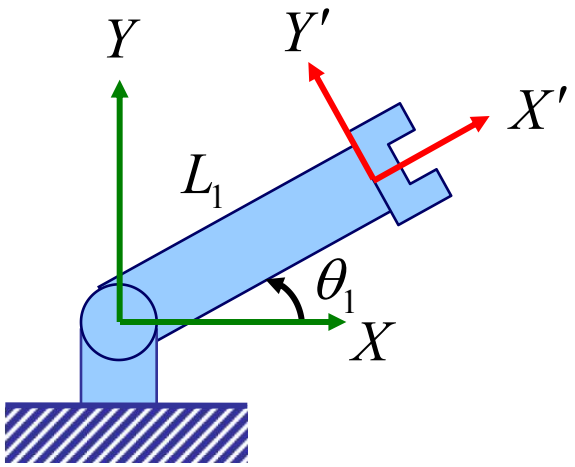


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

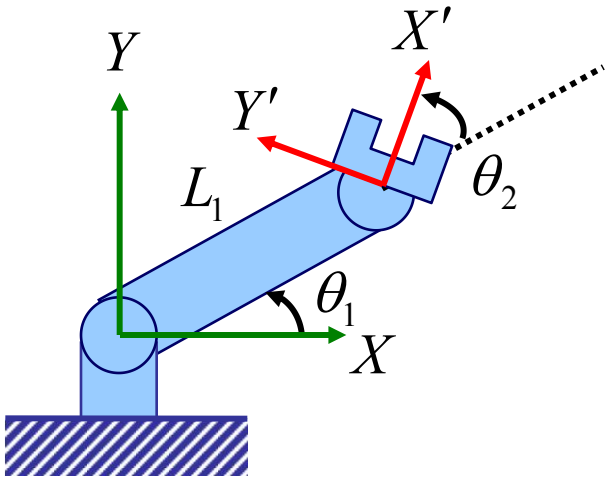


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

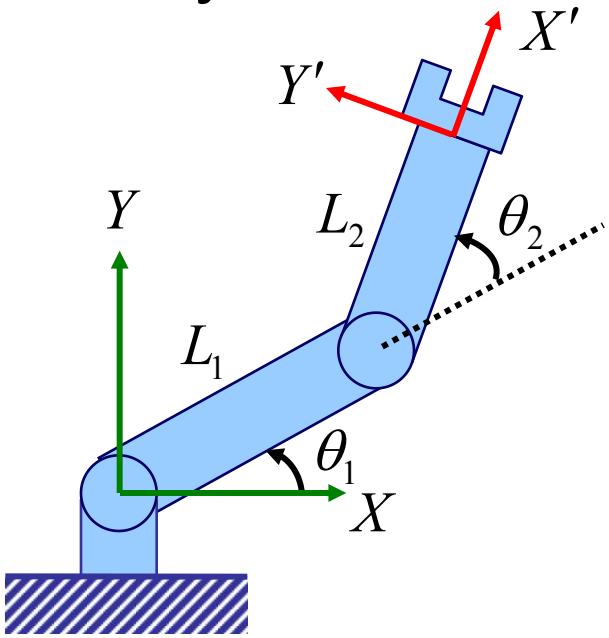


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of body-attached coordinate system

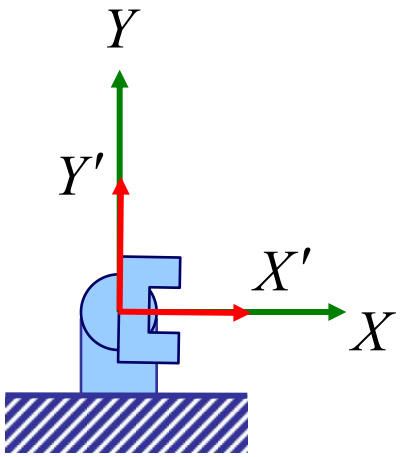


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of global coordinate system

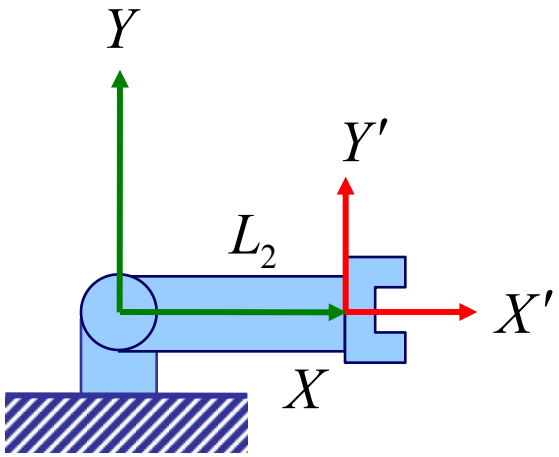


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of global coordinate system

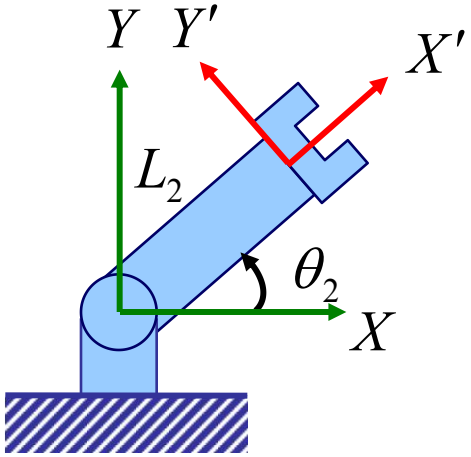


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of global coordinate system

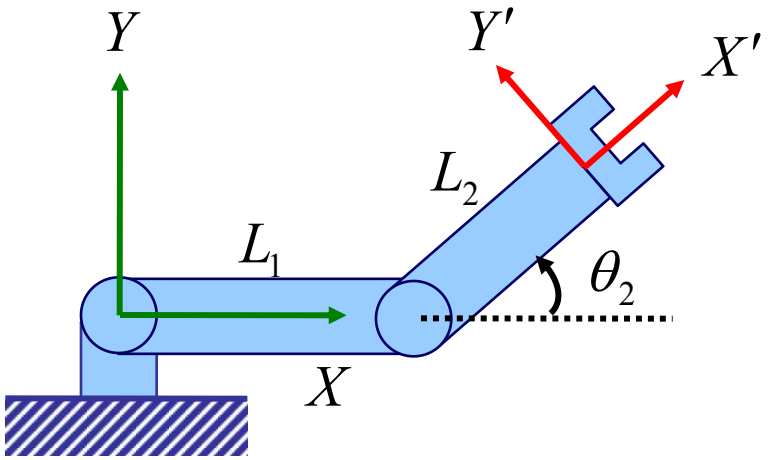


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of global coordinate system

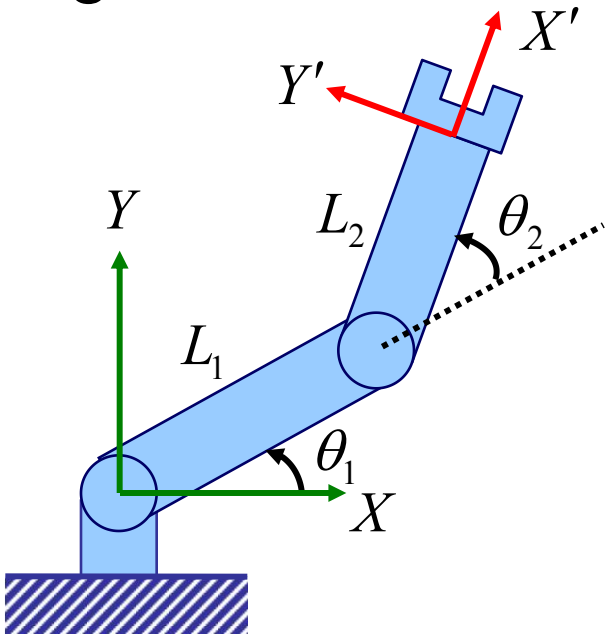


$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Thinking of Transformations

- In a view of global coordinate system



$$T = (rot\,\theta_1)(transl_1)(rot\,\theta_2)(transl_2)$$

$$= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & l_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

How to Handle Ball-and-Socket Joints ?

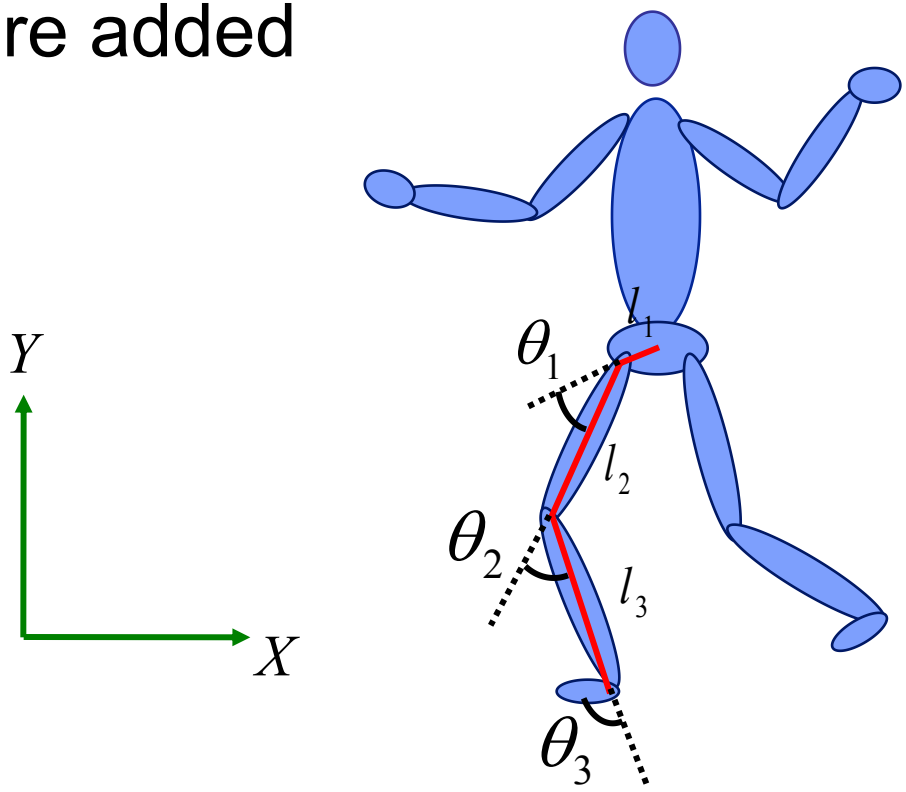
- Three revolute joints whose axes intersect at a point (equivalent to Euler angles), or
- 3D rotation about an arbitrary axis

$$T = (transl_1)(rot\theta_x)(rot\theta_y)(rot\theta_z)(transl_2)$$

$$= \cdots \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdots$$

Floating Base

- The position and orientation of the root segment are added



$$T_2 = (transl_r)(rot\ \theta_r)(transl_1)(rot\ \theta_1)(transl_2)(rot\ \theta_2)$$

Joint & Link Transformations

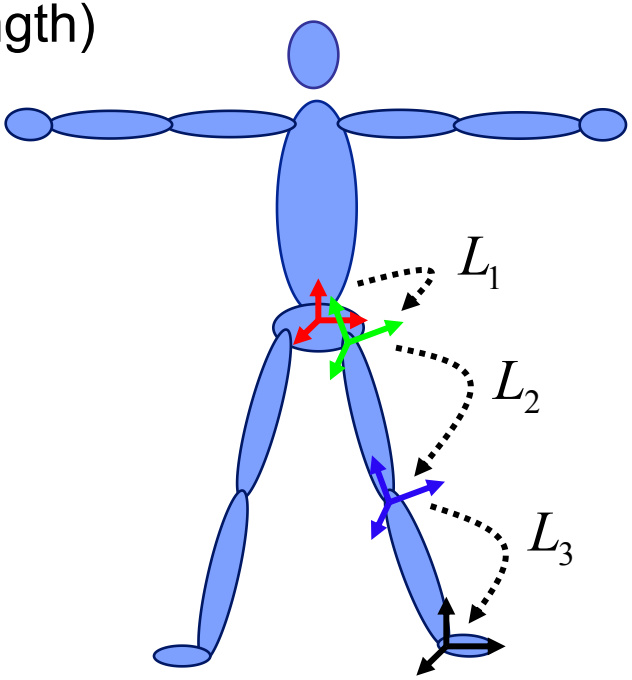
- Each segment has its own coordinate frame
- Forward kinematics map is an alternating multiple of
  - ***Joint transformations*** : represents joint movement
    - Usually rotations
  - ***Link transformations*** : defines a frame relative to its parent
    - Usually translations (bone-length)

1st link transformation

$$T = J_0 L_1 J_1 L_2 J_2 L_3 J_3$$

1st joint transformation

The position and orientation of the root segment



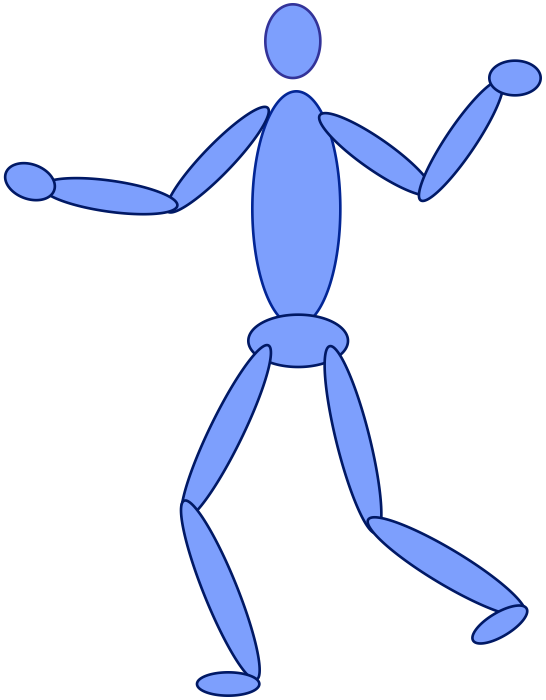
Joint & Link Transformations

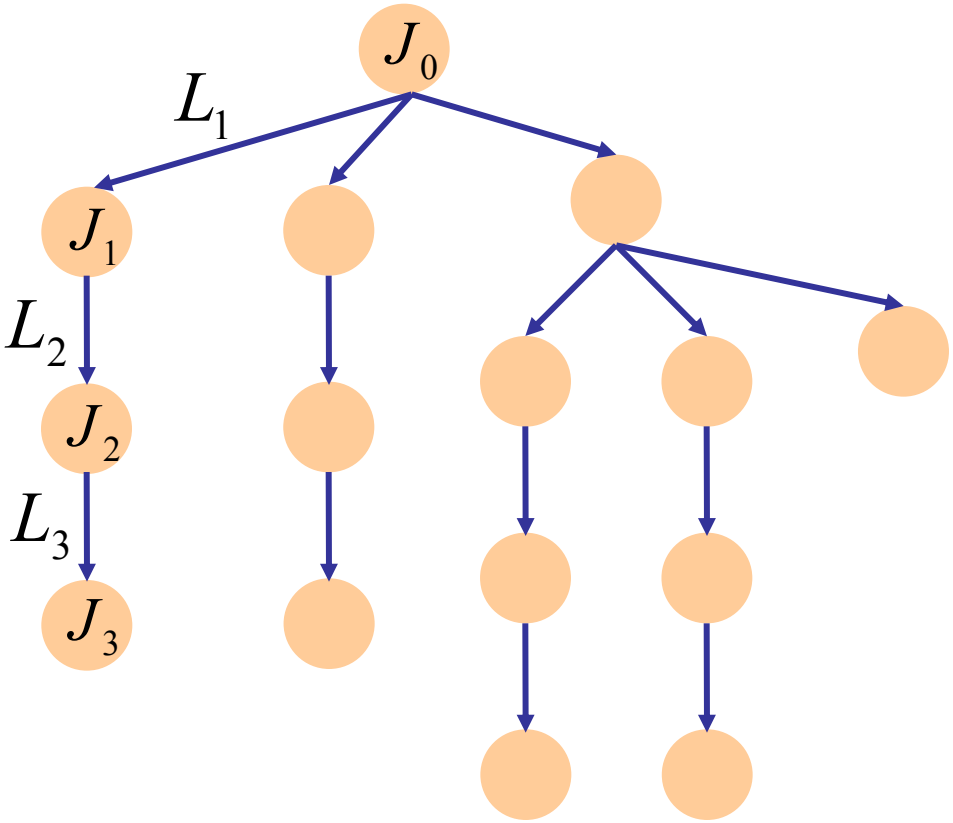
Note that

- Joint transformations may include translation
  - Human joints are not ideal hinges
- Link transformations may include rotation
  - Some links are twisted

Representing Hierarchical Models

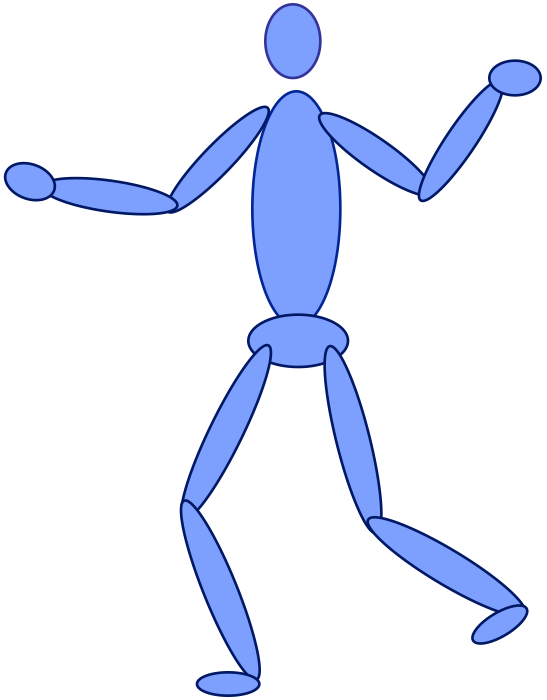
- A tree structure
  - A node contains a joint transformation
  - An arc contains a link transformation

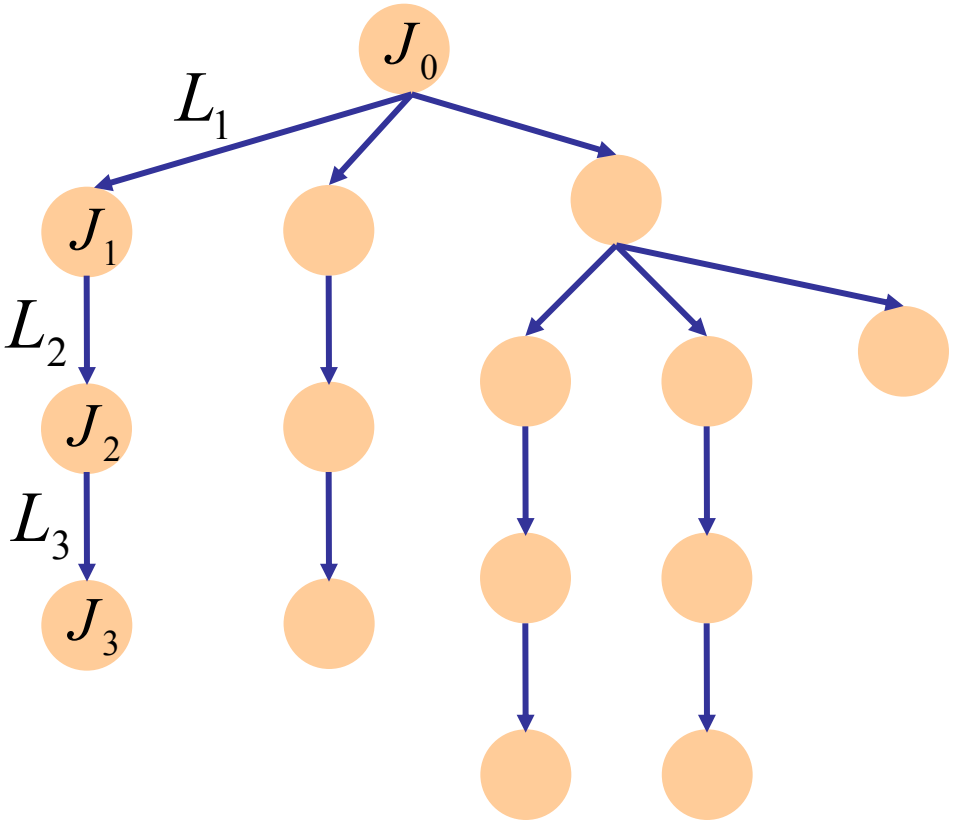




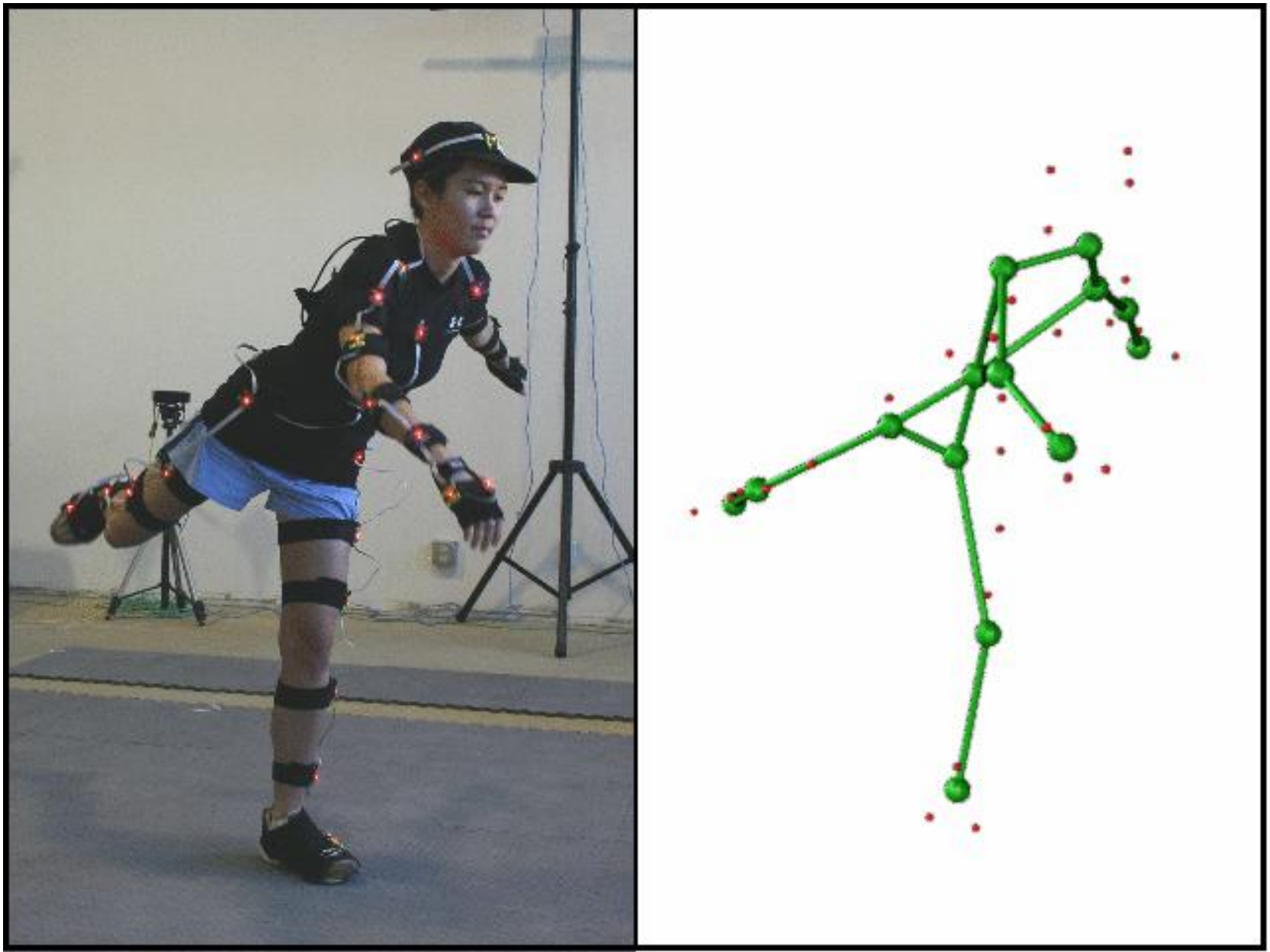
Depth-First Tree Traversal

- Draw graphics needs to compute the position and orientation of all links
- OpenGL's matrix stack is useful





Motion Capture



BVH mocap file format

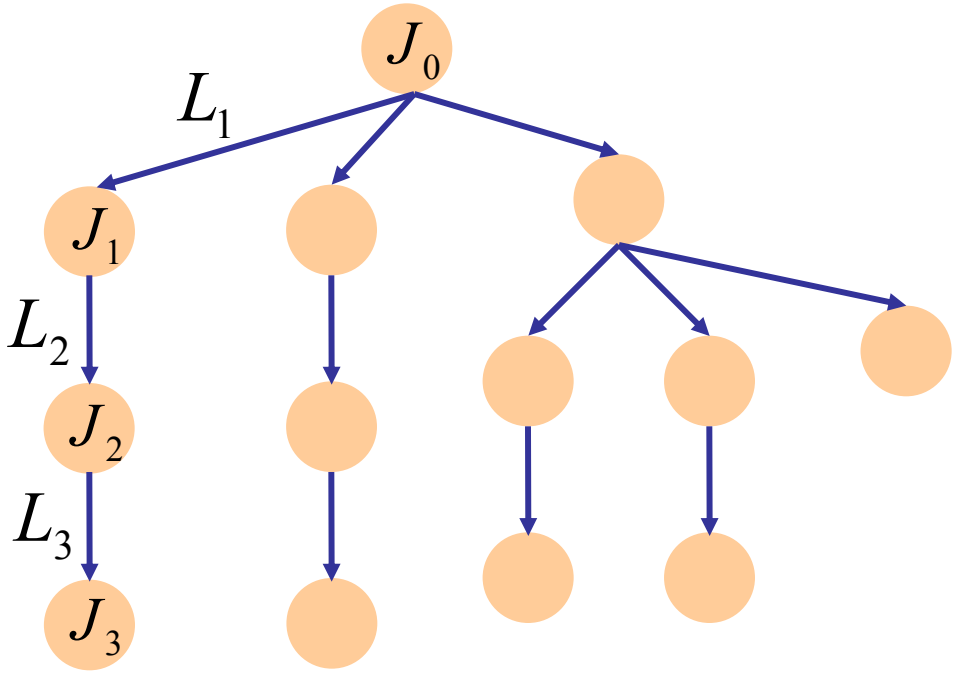
- The **Biovision Hierarchy (BVH)** file format was developed by Biovision, a motion capture company.
- It stores motion capture data.
- It's a text file.
- It matches with OpenGL well.
- It has two parts: hierarchy and data.

Part 1: Hierarchy

- The hierarchy is a joint tree.
- Each joint has an offset and a channel list.

- Joint 1:
- Offset: L1
- The channel list

defines a sequence of transformations of J1



An example

```
JOINT chest
{
        OFFSET  0.096536 3.475309 -0.289609
        CHANNELS 3 Xrotation Zrotation Yrotation
        JOINT neck
        {
                OFFSET  -0.096536 13.901242 -2.027265
                CHANNELS 3 Xrotation Zrotation Yrotation
                JOINT head
                {
                        OFFSET  -0.166775 1.448045 0.482682
                        CHANNELS 3 Xrotation Zrotation Yrotation
                        JOINT leftEye
                        {
```

An example

```
JOINT chest
{
        OFFSET  0.096536 3.475309 -0.289609
        CHANNELS 3 Xrotation Zrotation Yrotation
        JOINT neck
        {
                OFFSET  -0.096536 13.901242 -2.027265
                CHANNELS 3 Xrotation Zrotation Yrotation
                JOINT head
                {
                        OFFSET  -0.166775 1.448045 0.482682
                        CHANNELS 3 Xrotation Zrotation Yrotation
                        JOINT leftEye
```

Chest Joint and its local coordinate system

Neck Joint and its local coordinate system

Neck's offset from chest (like a bone)

Channel list: Transformation from chest coordinate system to neck coordinate system

Part 2: Data

MOTION
Frames: 4620
Frame Time: 0.008333
0.4954 -10.3314 36.8439 -1.0364 -2.0052 92.3287 6.5990 -0.8348 14.2185 0.0000 13.7079 12.9074 -5.0440 -2.9531 -19.4662 -0.0000 -36.9689 -0.0000 -5.8559 -2.0372 -19.6725 0.0000 16.1210 -14.2594 6.5762 2.0311 25.2317 -0.0000 -40.3850 -0.0000 0.4920 -11.6100 42.3449 1.5698 -10.2585 -1.3778 -1.6618 12.8953 -7.8738 -6.1442 0.0000 22.3789 19.4173 -9.7285 -32.4893 0.0000 -24.5218 -6.5960 5.5760 2.4836 -0.0000 18.7680 -0.0000 -2.8578 ...

Part 2: Data

```
JOINT chest
{
        OFFSET  0.096536 3.475309 -0.289609
        CHANNELS 3 Xrotation Zrotation Yrotation
        JOINT neck
        {
                OFFSET  -0.096536 13.901242 -2.027265
                CHANNELS 3 Xrotation Zrotation Yrotation
                JOINT head
                {
                        OFFSET  -0.166775 1.448045 0.482682
                        CHANNELS 3 Xrotation Zrotation Yrotation
```

Variable 1 Variable 2 Variable 3

Variable 4 Variable 5 Variable 6

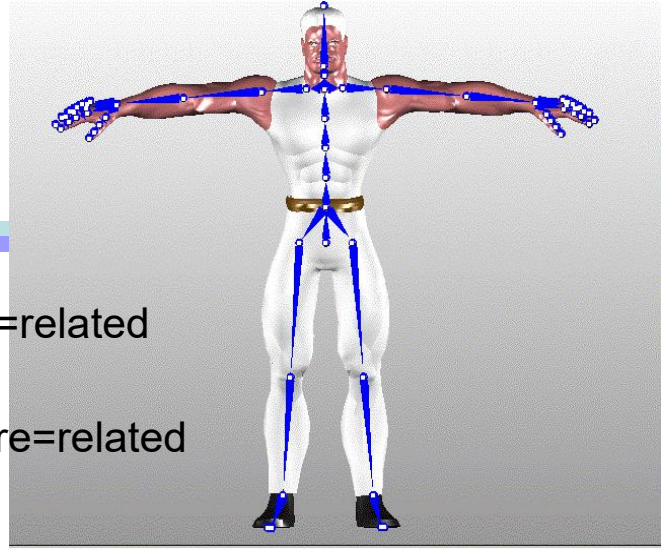
Variable 7 Variable 8 Variable 9

- The data part stores a number of frames.
- Each frame is a list of variables.
- The model is animated, when each frame uses variables.

Summary

- Kinematics is the study of motion of articulated figures
  - Kinematics does not consider physics (forces, mass, …)
- Forward kinematics is straightforward
  - Forward kinematics map can be considered as a chain of coordinate transformations

Skinning



http://www.youtube.com/watch?v=dniWVu55PEc&feature=related

http://www.youtube.com/watch?v=z0QWBdk8MCA&feature=related

- Assuming the skin mesh is given
- 1. Kinematics : How the motions of the skeleton are given
- 2. Skinning : How the character's skin deform according to the motion of the skeleton

For robots, skinning is trivial

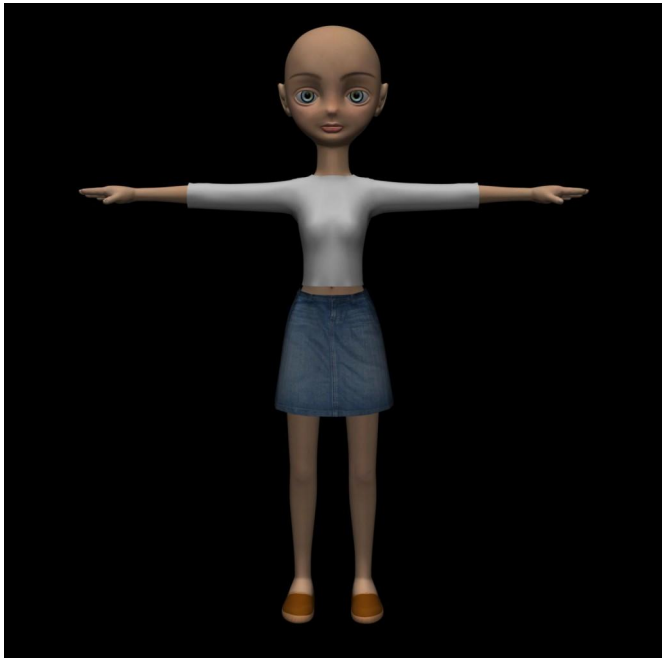
- Compute the local-to-global matrix for all body segments : **M**
- Multiply the local coordinates **v** of every body segment to this matrix to calculate its global position :

$$\hat{v} = \mathbf{M}\mathbf{v}$$

- Then, the global position of all the points can be computed

What about for characters?

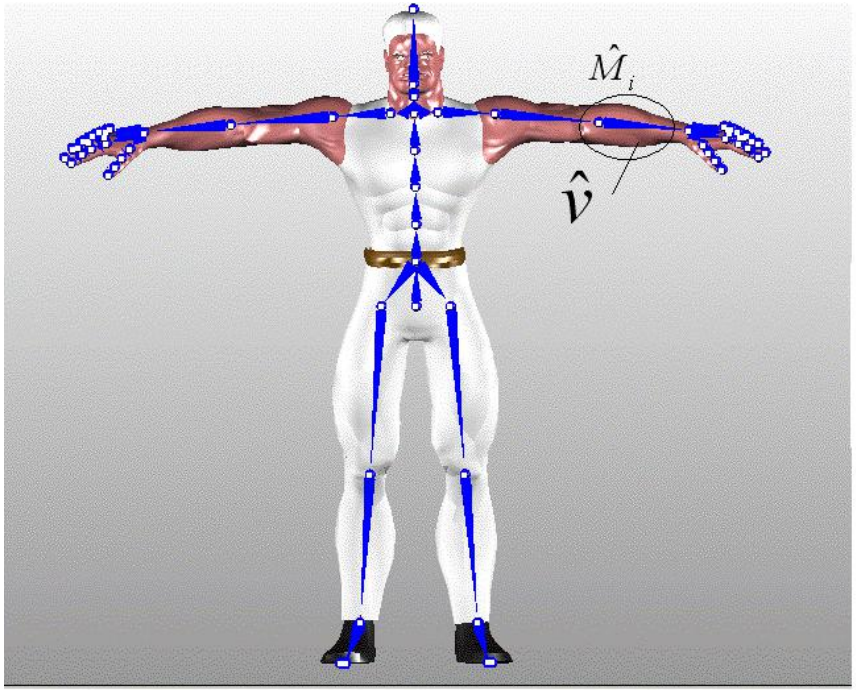
A single mesh is modeled : the rest pose



Rest Pose to Bone Coordinate

- the global position of a particular vertex, *v*, in the rest pose is written as $\hat{v}$
- The transformation matrix associated with bone *i* in the rest pose is written as $\hat{M}_i$

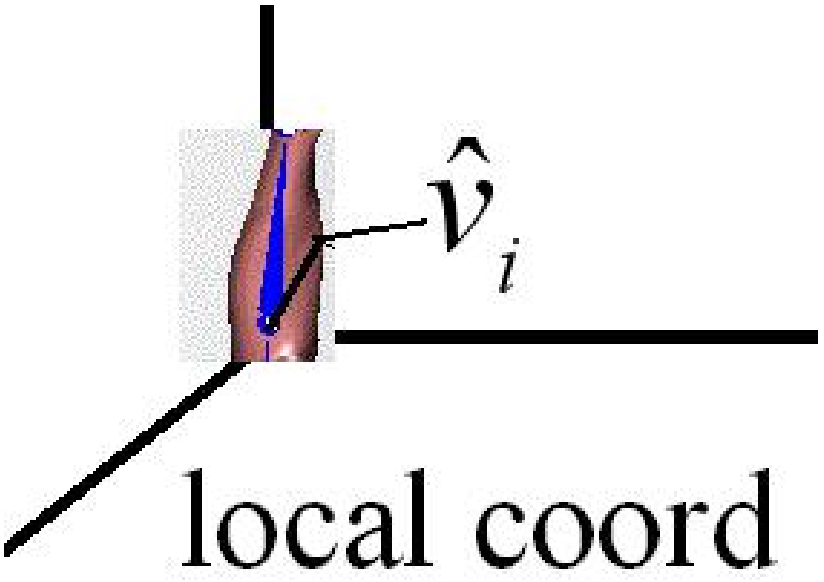
Rest pose:
The pose in which surface mesh is modeled



Rest Pose to Bone Coordinate

- for each bone, i, the position of the vertex in the rest pose is first transformed from model coordinates ($\hat{v}$) to bone coordinates ($\hat{v}_i$) by applying the inverse of the rest pose bone transformation:

$$\hat{\mathbf{v}}_\mathbf{i} = \hat{\mathbf{M}}_\mathbf{i}^{-1}\hat{\mathbf{v}}$$

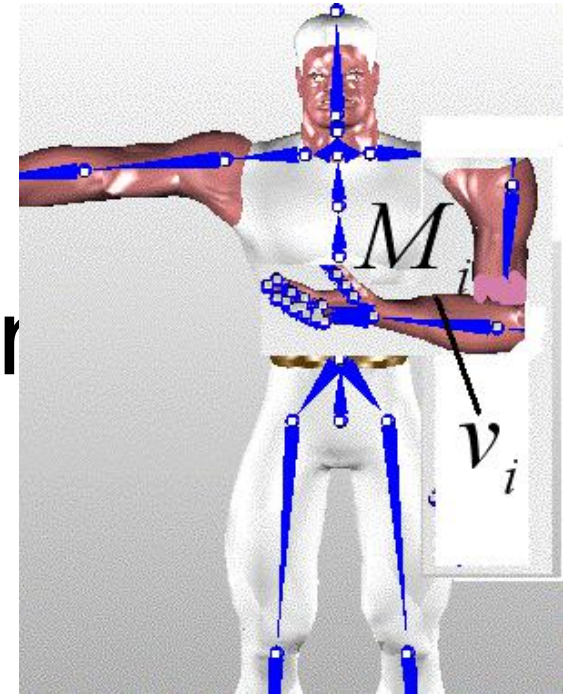


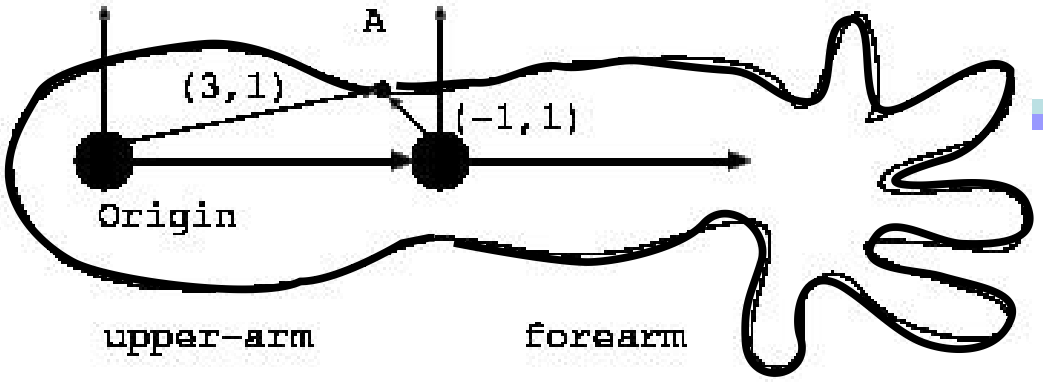
Bone Coordinate to World Coordinate

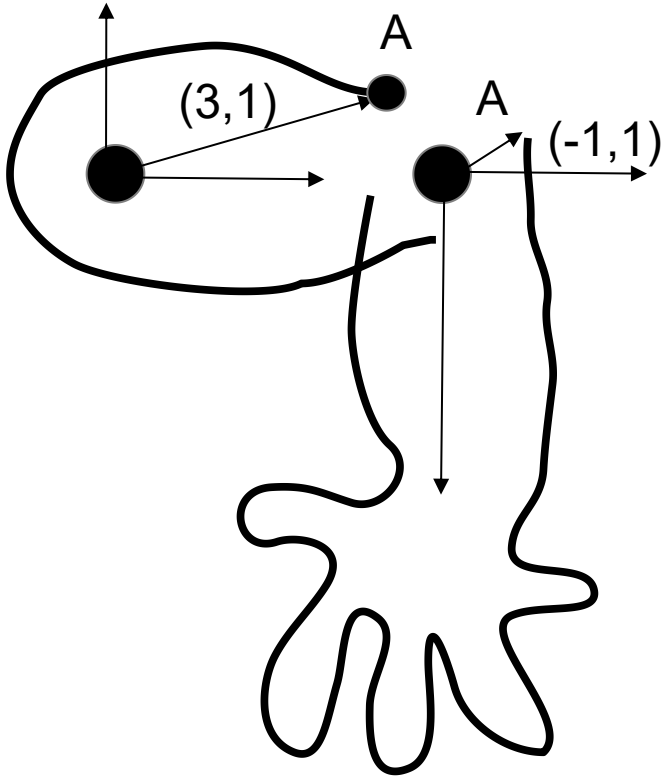
- The vertex in bone coordinates, $\hat{v}_i$ is then transformed back into world coordinates by applying the transformation of the bone in its new skeletal configuration:

$$v_i = M_i \hat{v}_i = M_i M_i^{-1} \hat{v}$$

M_i : the local-to-global matrix in the new posture

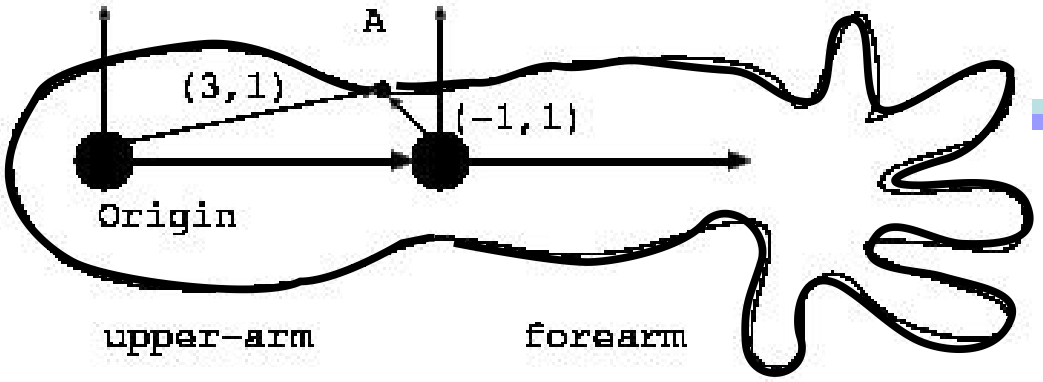


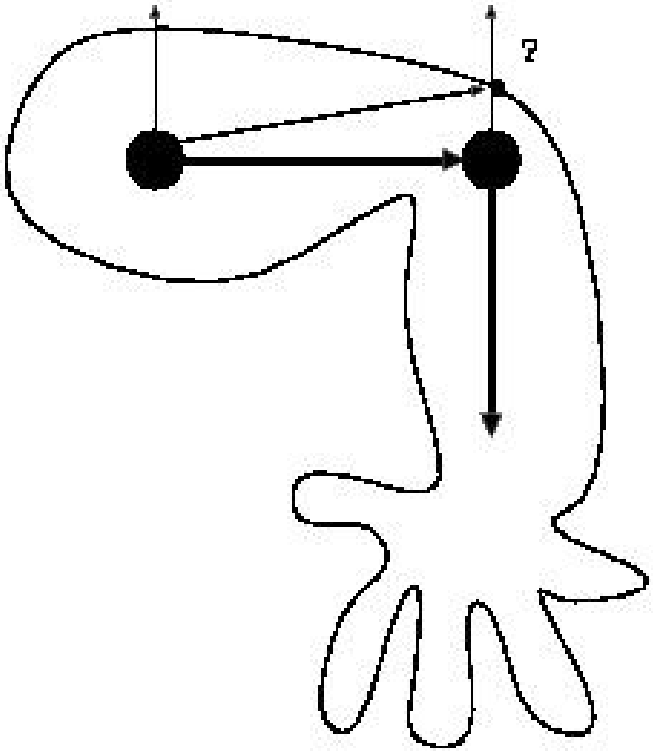


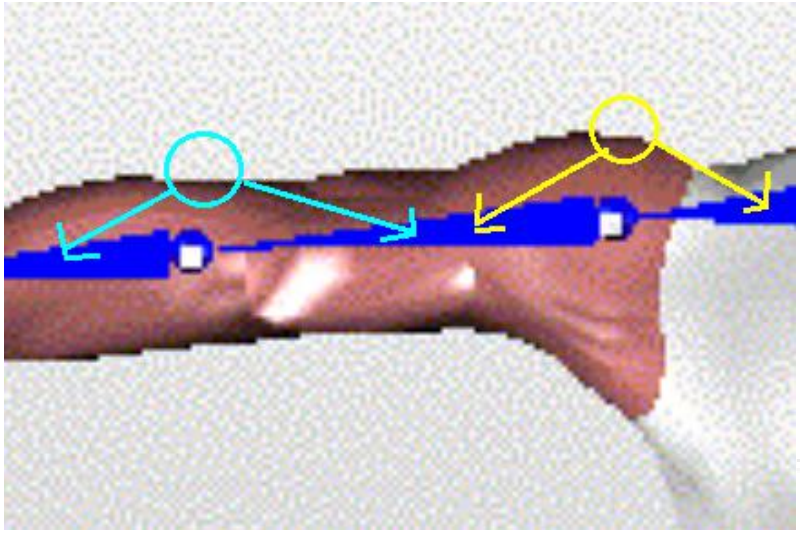


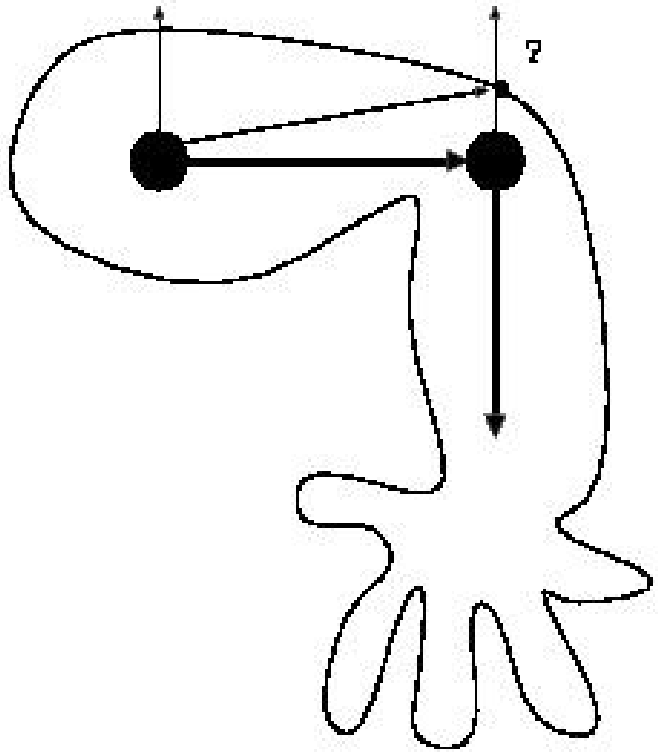
$$v_i = M_i \hat{v}_i = M_i \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

M_i : The forearm matrix in the new pose


A
(3,1)
(-1,1)
Origin
upper-arm
forearm


?



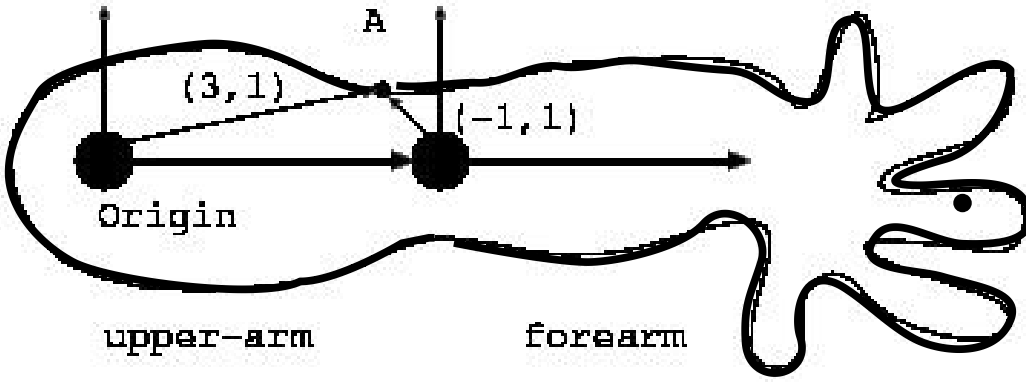


Problem :

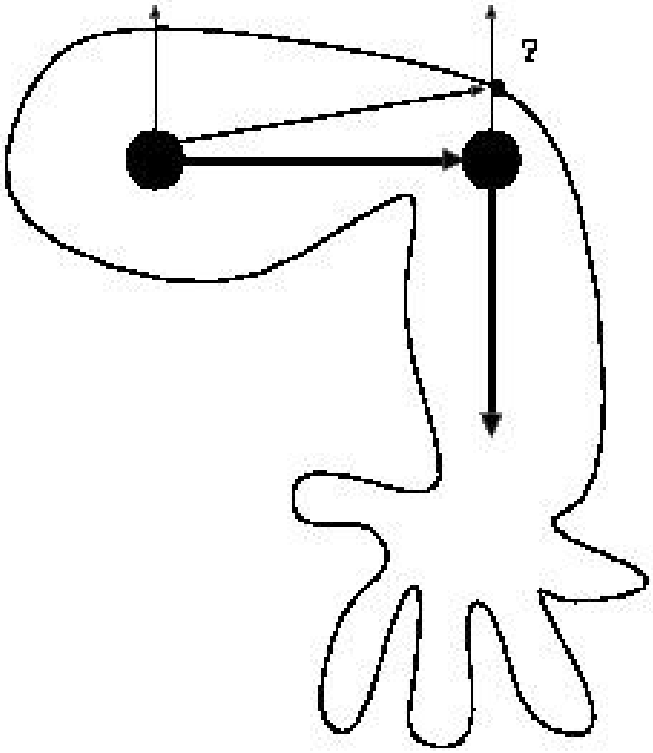
For some points, we don’t know to which body segment it belongs to

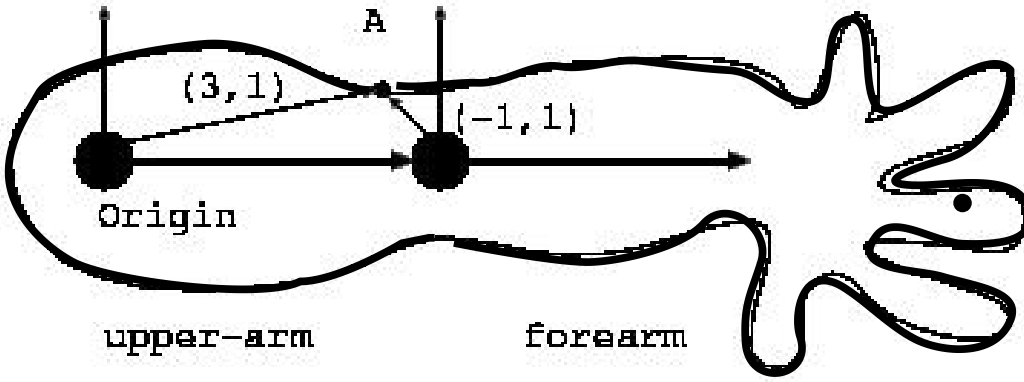
For the points near the elbow joint, it might belong to the upper arm , or the forearm or maybe both

We want both segments to affect its movements

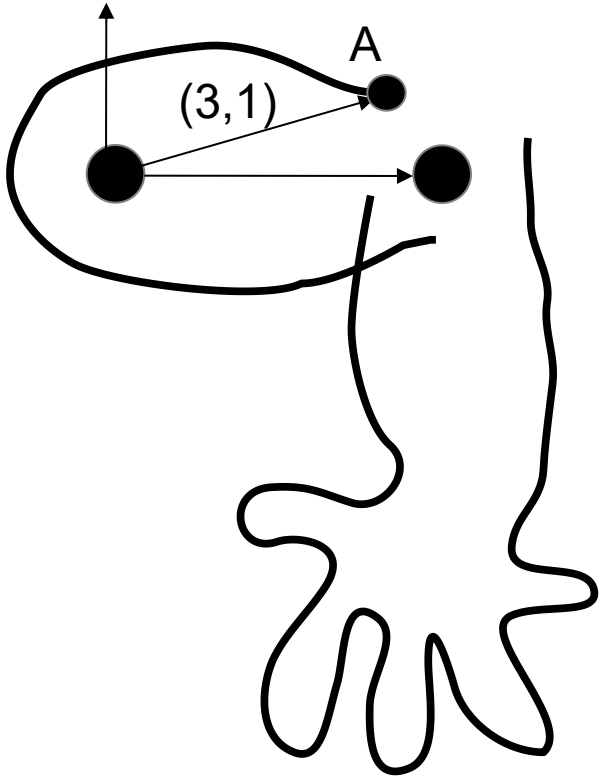


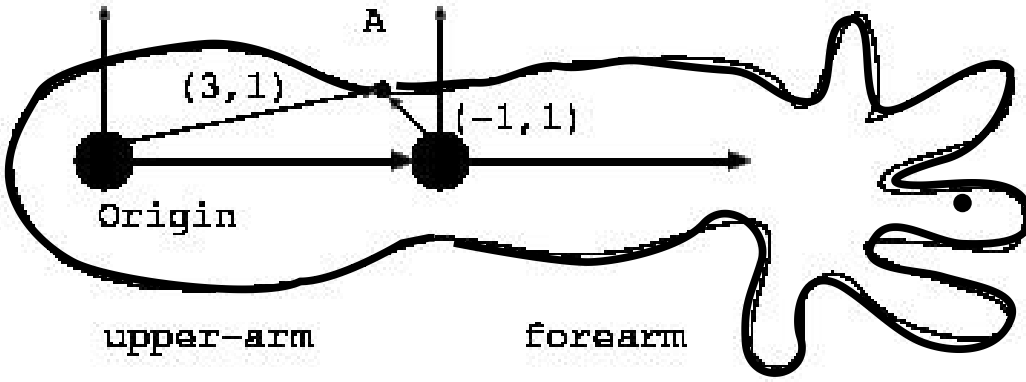
What is the position of point A after the elbow is bent 90 degrees?

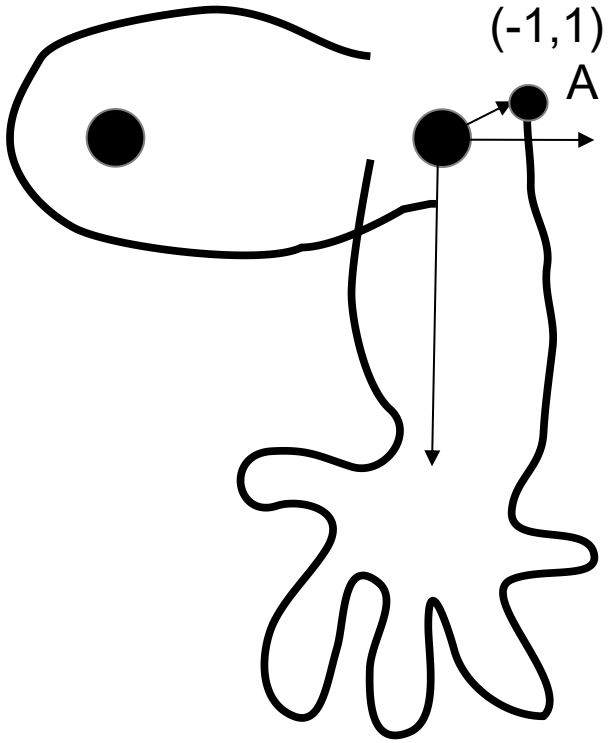




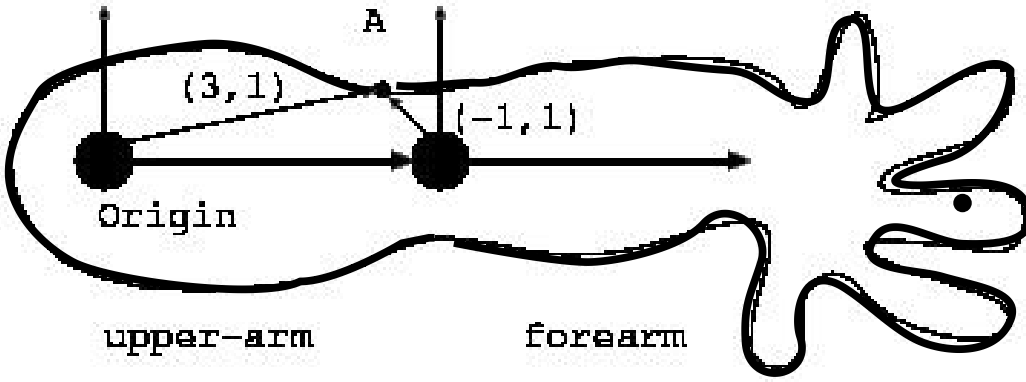
- What is the position of point A after the elbow is bent 90 degrees?
- Assuming it is a point of the upper arm

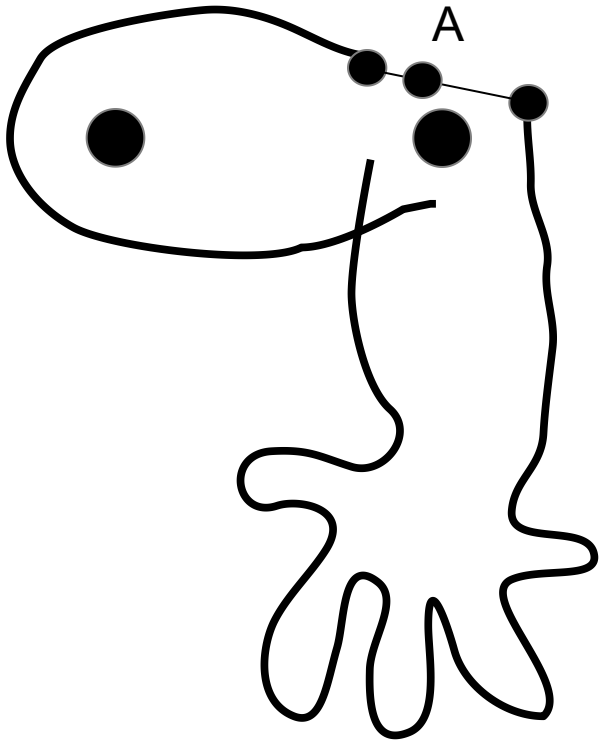






- What is the position of point A after the elbow is bent 90 degrees?
  - Assuming it is a point of the upper arm
  - Assuming it is a point of the lower arm





- What is the position of point A after the elbow is bent 90 degrees?
- Assuming it is a point of the upper arm
- Assuming it is a point of the lower arm
- Assuming the weight is 0.8 for the upper-arm and 0.2 for the forearm

Solution: Linear Blending

- Linear Blending determines the new position of a vertex by linearly combining the results of the vertex transformed rigidly with each bone.
- A scalar weight, w_i, is given to each influencing bone and the weighted sum gives the vertex's position, v, in the new pose, as follows:

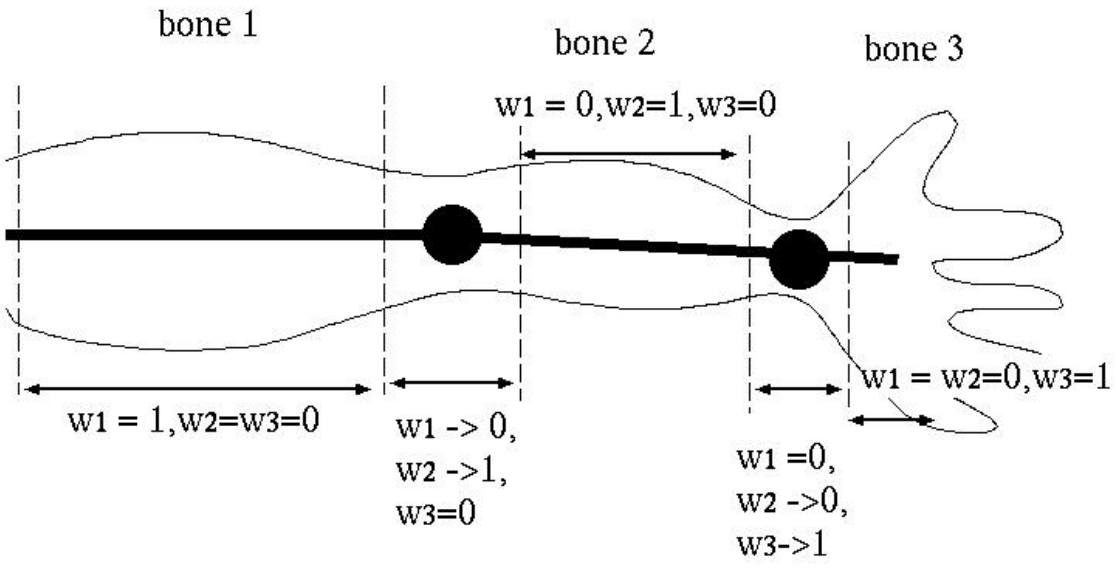
$$\mathbf{v_i} = \sum_{i=1}^{b} w_i \mathbf{M_i} \hat{\mathbf{M}}_\mathbf{i}^{-1} \hat{\mathbf{v}} \qquad \sum_{i=1}^{b} w_i = 1$$

b is the number of bones influencing the position of v

How to decide the weights?

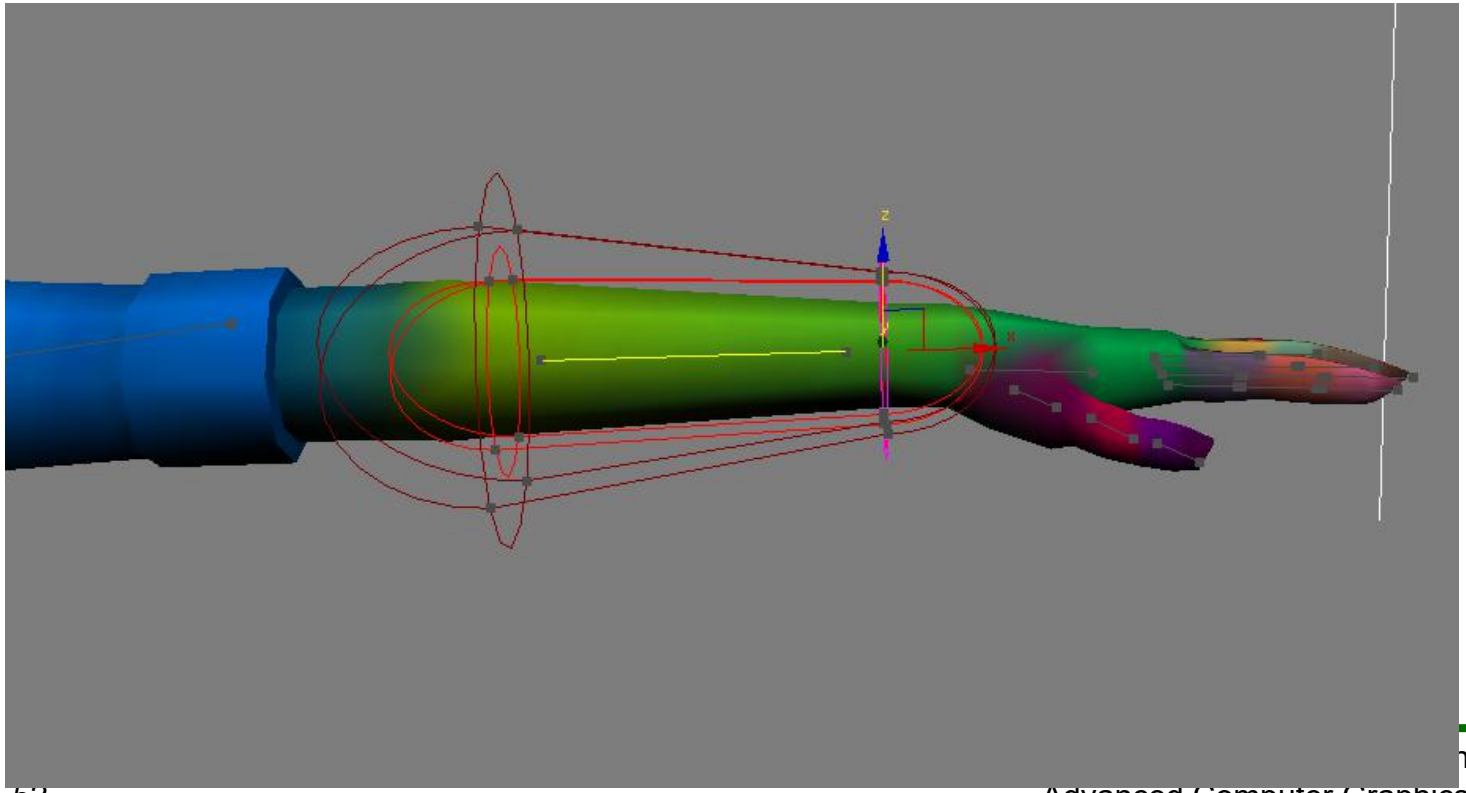
Decide the mapping of the vertex to the bone

- If vertex v is in the middle of bone i, then $w_i = 1$ and for the rest $w_{j \neq i} = 0$
- If the vertex is near the border of bone i and i+1, *wi* will gradually decrease to 0 and *wi+1* will gradually increase to 1
- If the vertex is affected by more than three bones, the weight can be determined according to its distance to each bone

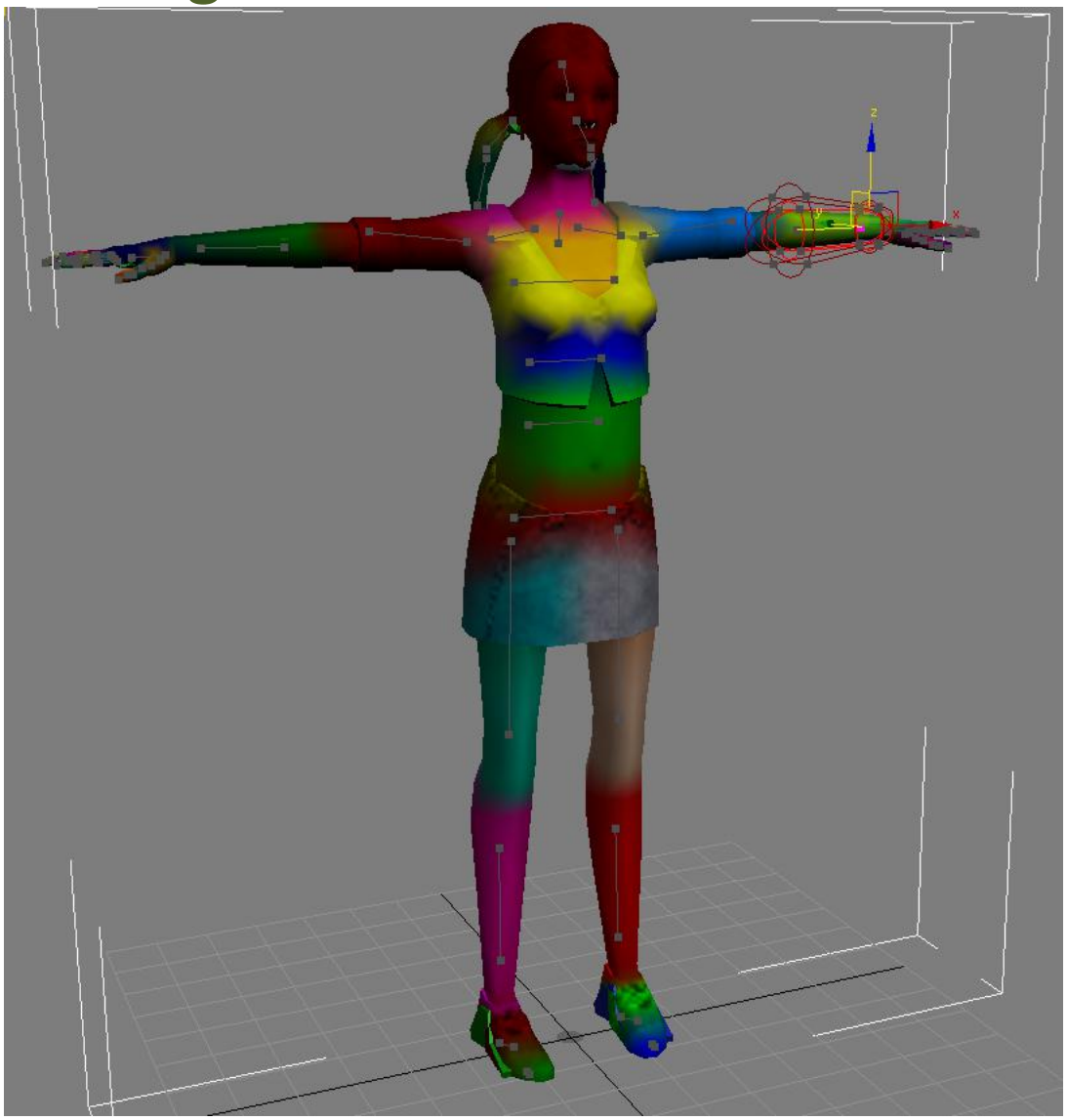


Containment Binding (3DSMax)

- Volume primitives around the bones
  - Boxes, cylinders, etc.
  - Vertex weights assigned based on which primitives it is in



Smooth Skin Algorithm



Problems with Linear Blending

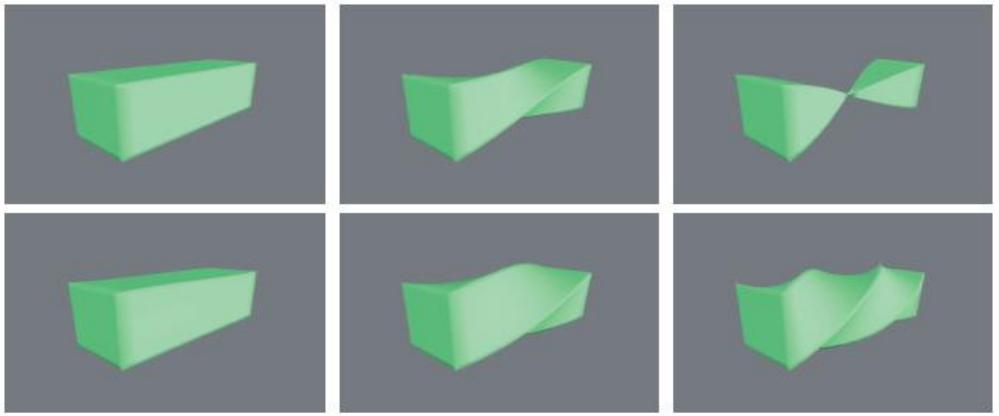
- The meshes exhibit volume loss as joints are rotated to extreme angles.
- These are called “joint collapse” and “candy wrapper” effect
- These undesirable results occur because of a lack of flexibility in the framework.

Linear blending

Non-linear blending

90 degree

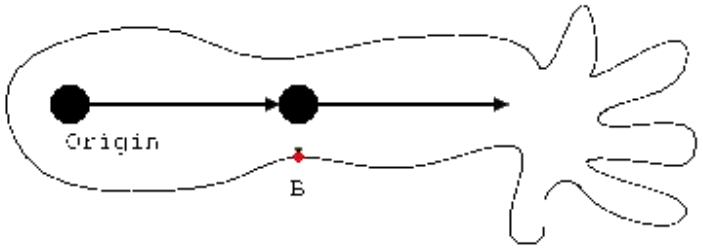
180 degree

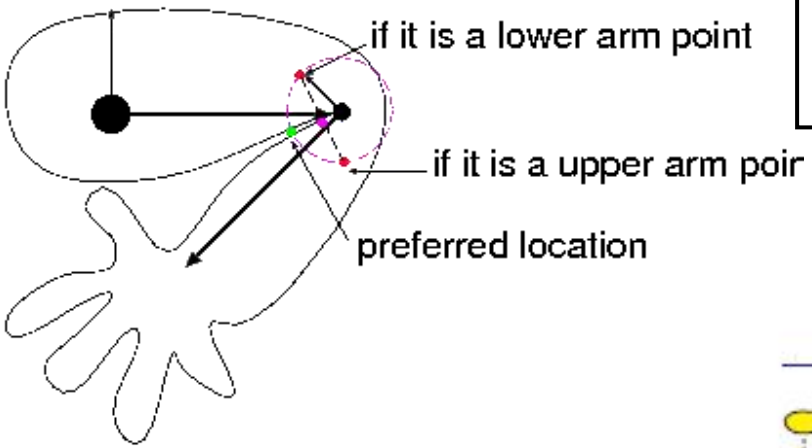


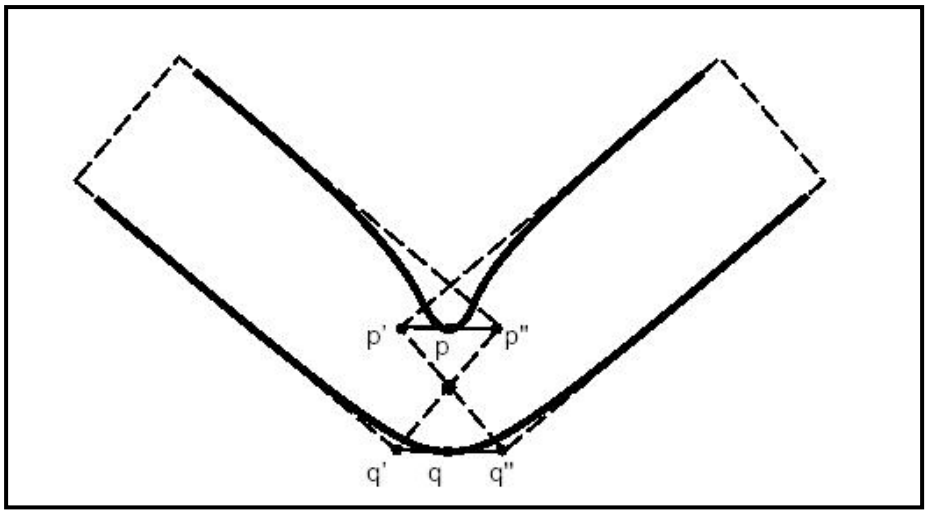
Limitations of Smooth Skin

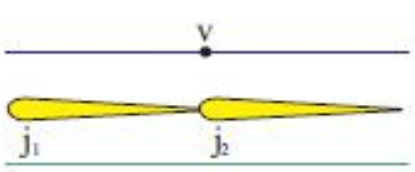
Why does it happens?

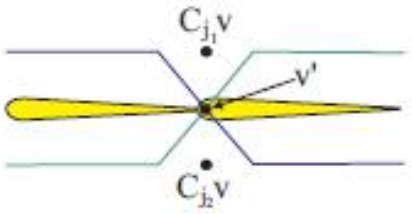
- Simple linear blending in the Cartesian space causes the artefacts





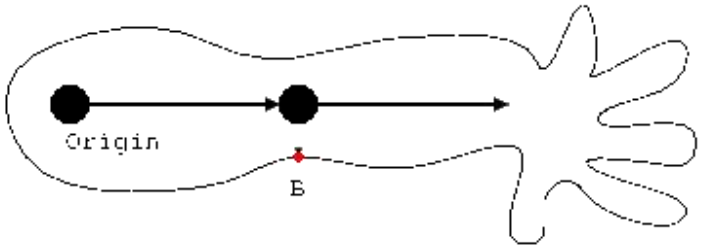


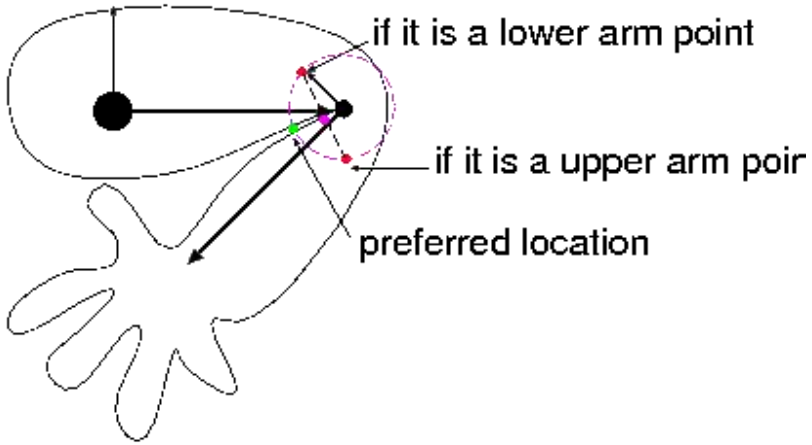


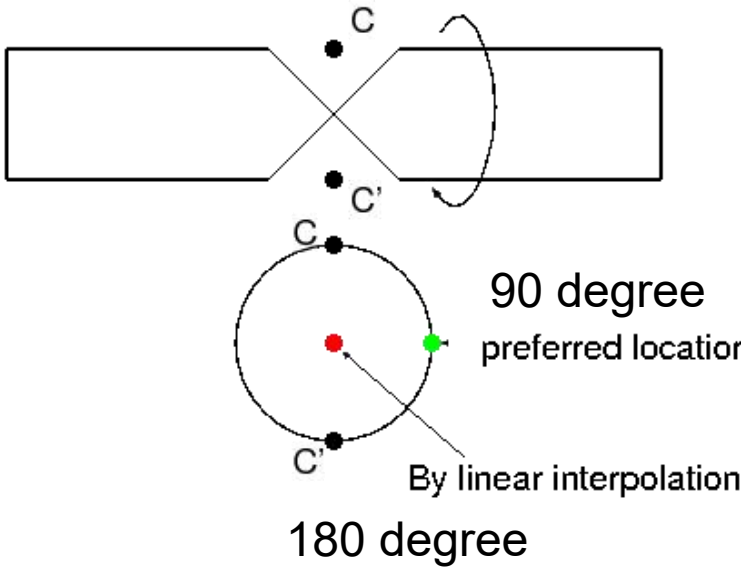


Solution?

- Interpolating the homogeneous transformations
  - Kavan et al. [SIGGRAPH08], etc.



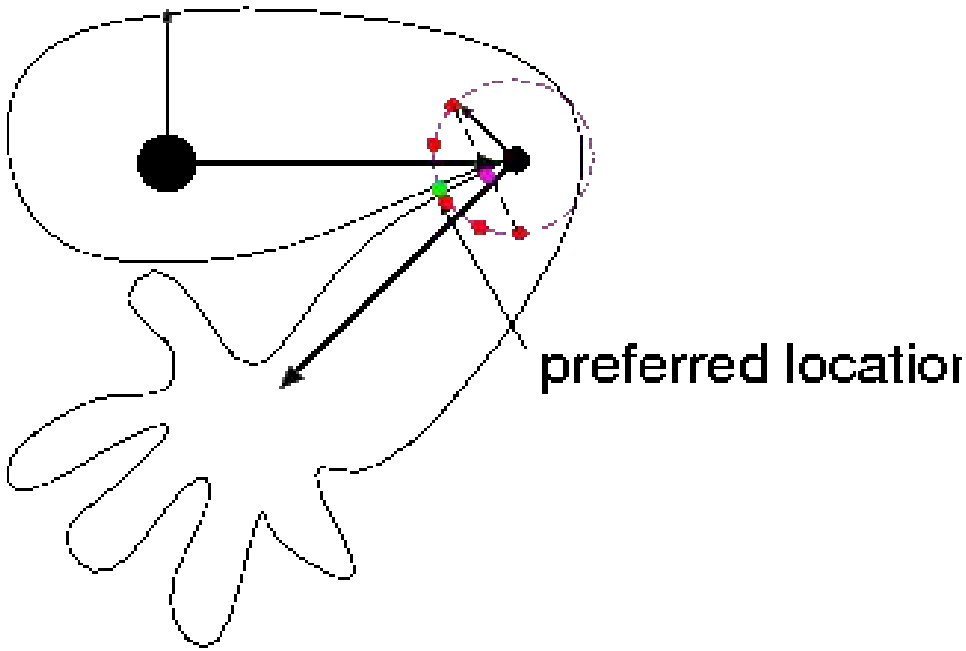




How to solve volume loss?

Add additional joints that half interpolate the rotations

- Each joint should be only rotated a little, not reaching the extreme angle

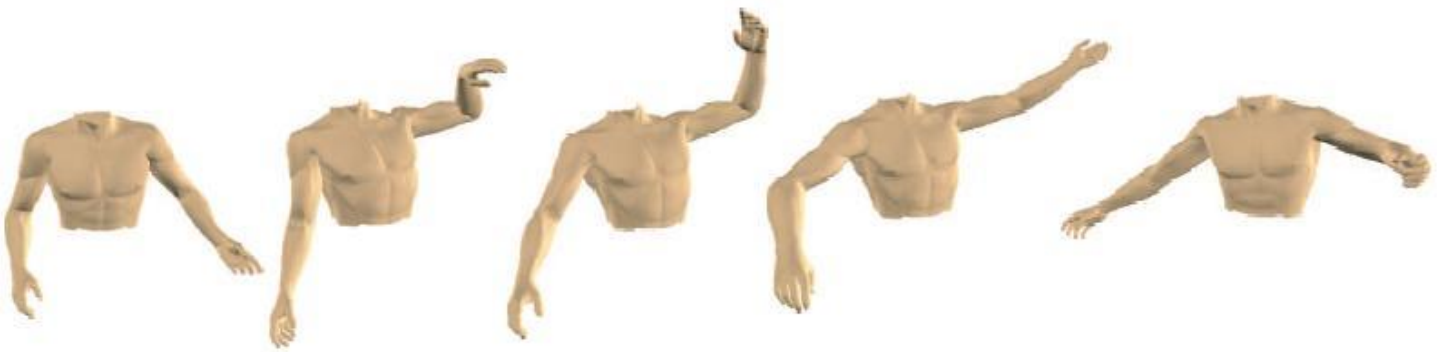


Remaining difficulties

- Deciding Weights
- Deciding Skeletons
- Muscle buldging
  - Bulging the biceps when the elbow is bent

Deciding Weights from Examples

- Instead of manually tuning the weights, we can let the system learn them from examples

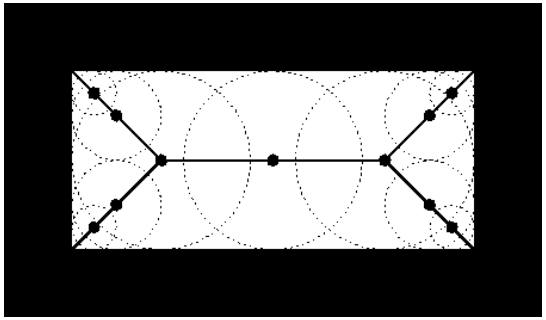


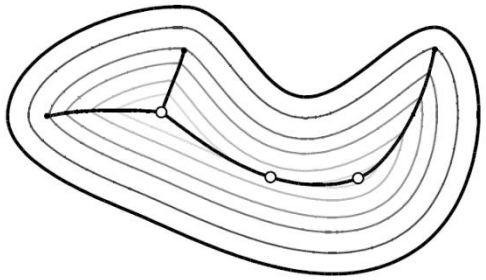
- Alex Mohr and Michael Gleicher [SIGGRAPH'03]

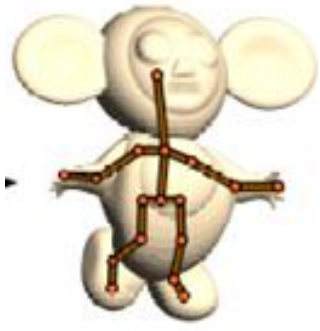
http://www.youtube.com/watch?v=e6y5plTq6bo

Automatic Rigging

- How can we automatically compute the skeleton from the polygon data?
- Can make use of the medial axis
- Medial axis is where the distance to the surface is C1 discontinuous
- Can be computed by circle / sphere fitting

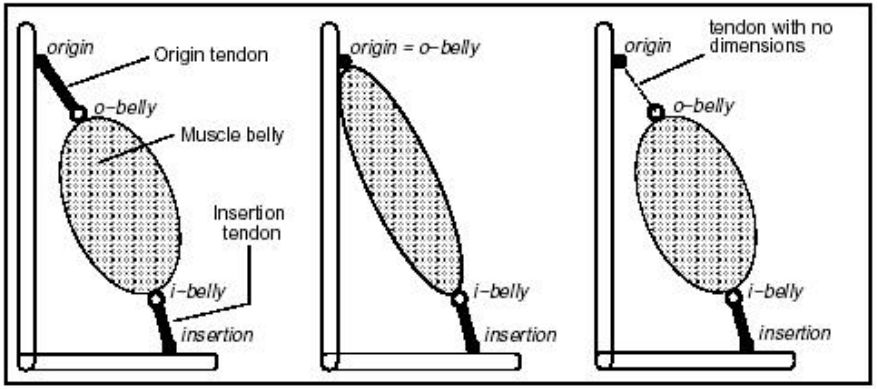


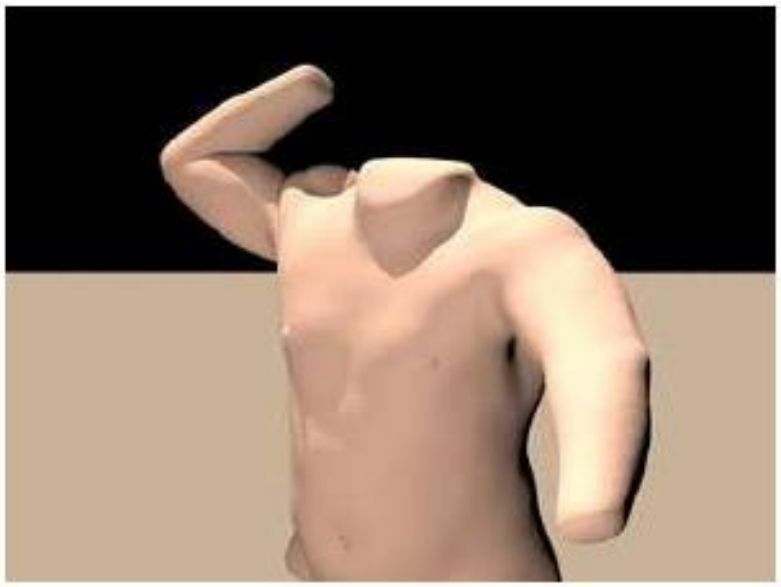




Anatomical models

- Model the body by
  - Muscles
  - Fat
  - Skin

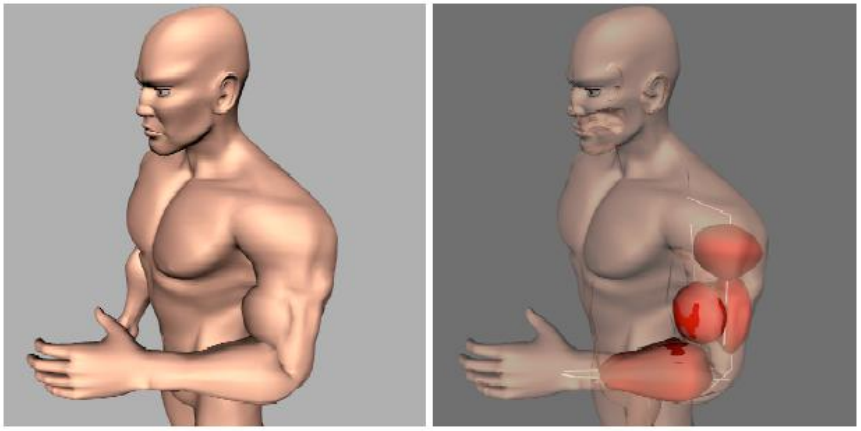




Detailed Anatomical Models

Method

1. When the joints are bent, the muscles contract
2. The distance between the origin and insertion point decreases
3. The volume of the muscles are kept the same, so they pump up
4. The skin is deformed to cover the muscles



Summary

- Kinematics
  - Hierarchical Model
  - Motion Capture
- Skinning
  - Linear Blending
  - Example-based method
  - Automatic Rigging
  - Anatomical models

Readings (beyond the scope of this class)

Skinning

- **A Comparison of Linear Skinning Techniques for Character Animation Afrigraph 2007**
- **Multi-Weight Enveloping: Least-Squares Approximation Techniques for Skin Animation**
  - Wang and Phillips, SCA 02
  - http://portal.acm.org/citation.cfm?id=545283

- Guessing the weights from examples
  - Alex Mohr Michael Gleicher **Building Effcient, Accurate Character Skins from Examples.** SIGGRAPH 2003
- **Automatic Rigging and Animation of 3D Characters** Ilya Baran Jovan Popovi ´ c, SIGGRAPH 2007
- http://www.mit.edu/~ibaran/autorig/pinocchio.html
- Geometric Skinning with Approximate Dual Quaternion Blending, SIGGRAPH 2008