Computer Animation

Lecture 2.

Basics of Character Animation

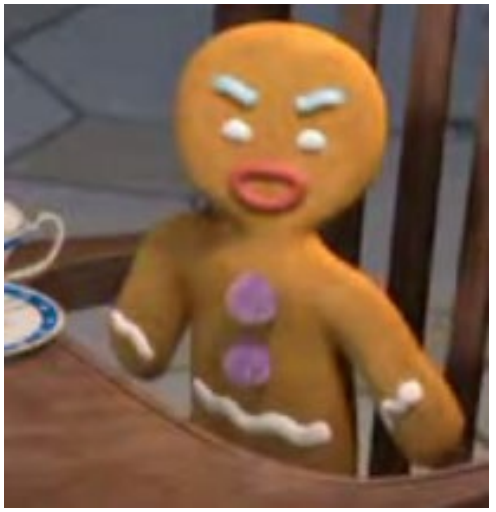
Taku Komura

Overview

- Character Animation
  - Posture representation
    - Hierarchical structure of the body
    - Joint types
      - Translational, hinge, universal, gimbal, free
    - Euler angles
      - Gimbal lock
    - Quaternions
    - Coordinate Transformations
- Creating the animation (Keyframe animation)
  - Interpolation
  - Inverse Kinematics
    - Analytical
    - CCD
    - Pseudo Inverse

Characters include

- Human models
- Virtual characters
- Animal models



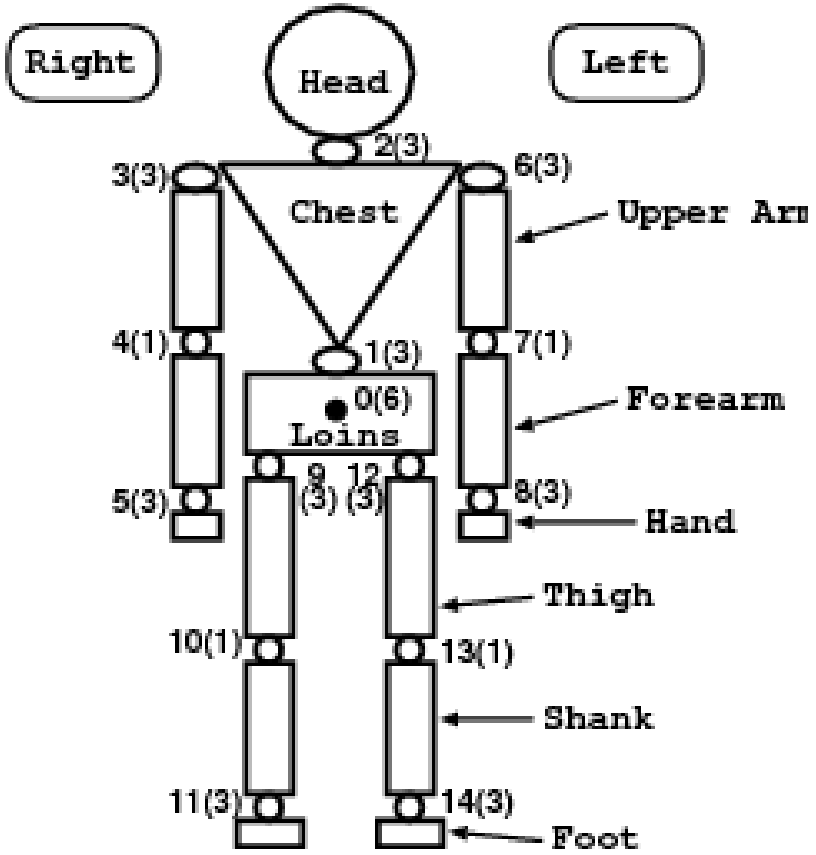
Controlling the Skeleton

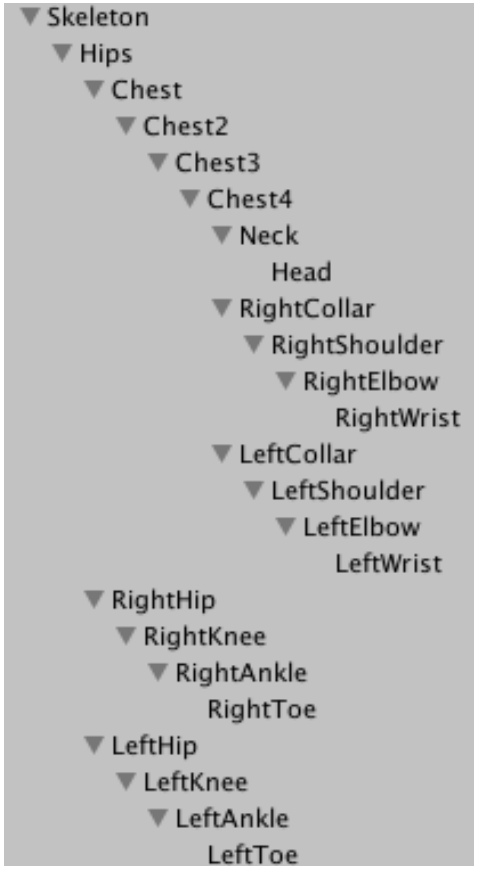
- We control the skeleton of the character
  → *skeletal motion*

- The skin follows the movement of the skeleton
  → *skinning deformation*

Representation of Postures

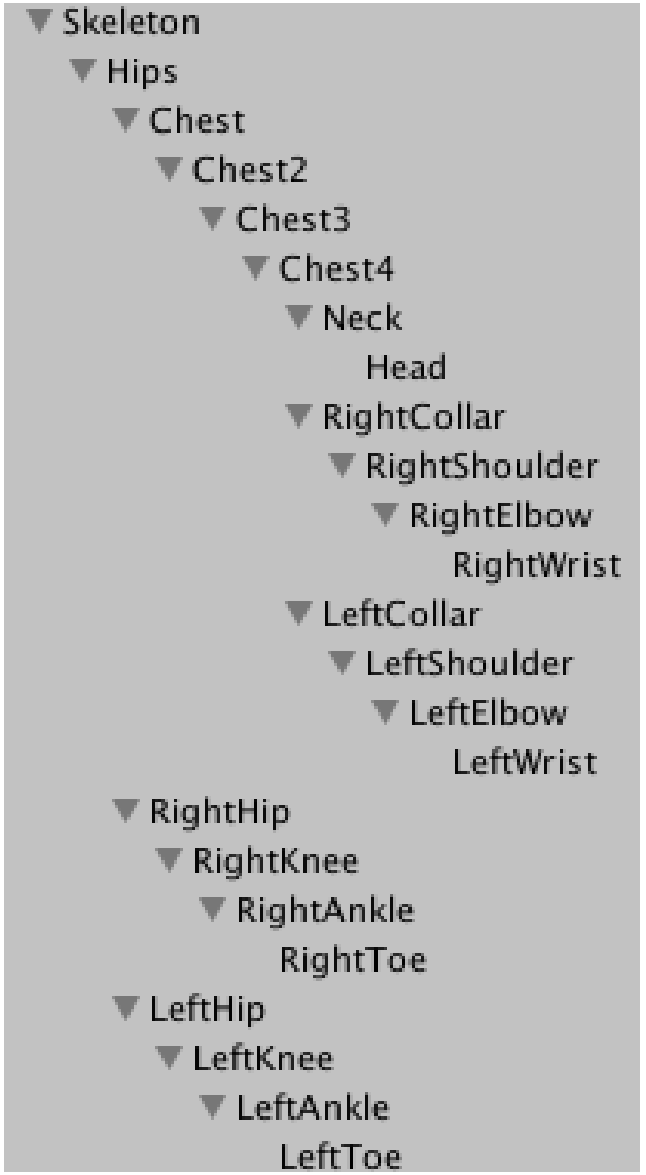
- The body has a hierarchical structure
- Many & different types of joints





Hierarchical structure of the body

- The first joint is typically called the “root” of the character
- The position of the joints lower in the hierarchy are affected by those above it
- Each joints can have 1 to 6 degrees of freedom (DOF)
- For rotational joints, usually it is 1, 2, or 3
- The root of the body has 6 DOF in total (3 DOF for translation, 3DOF for rotation)



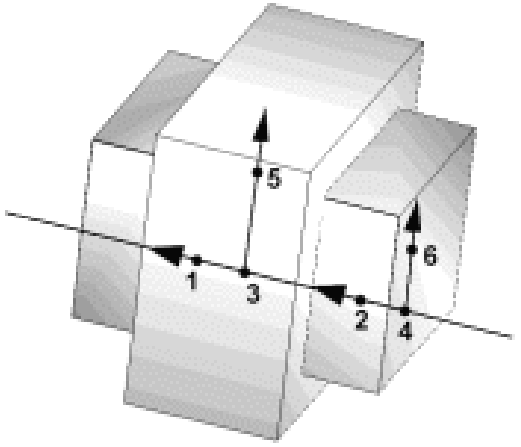
Joints

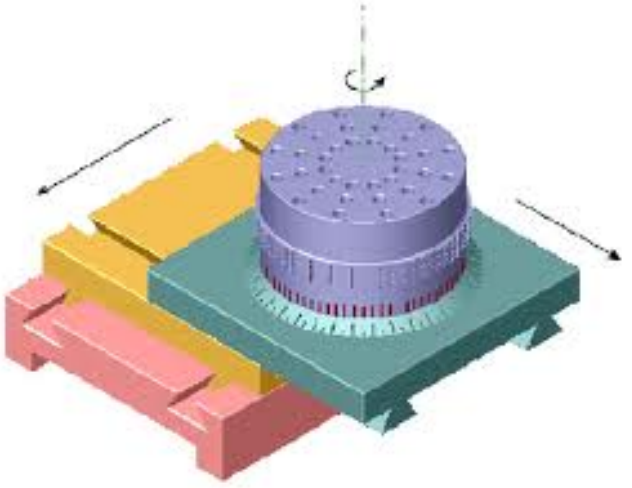
- The Degrees of Freedom (DOF) is defined for various joints

- There are several kinds of joints
  - Translational joint (1,2,3 DOF)
  - Hinge joints (1 DOF)
  - Universal joint (2 DOF)
  - Gimbal joint (3 DOF)
  - Free joint (3 DOF)

Translational joint

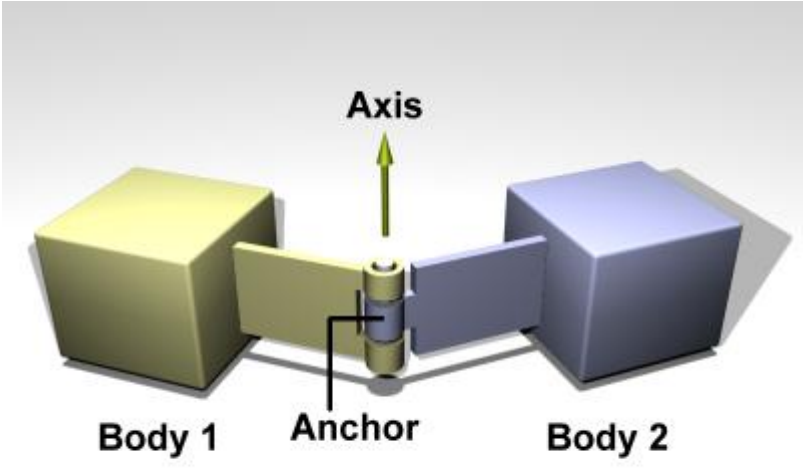
- A sliding joint
- Can be 1,2 or 3 DOF





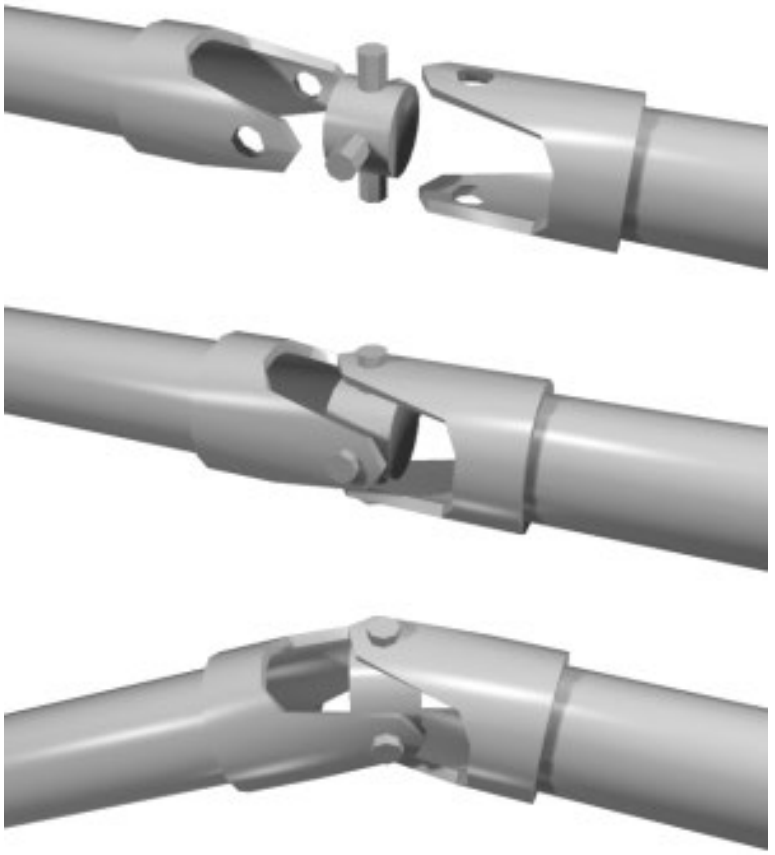
Hinge Joint

- A 1 DOF rotational joint
- Can be defined by the axis of rotation
- **Knee, elbow**



Universal Joint

- 2DOF
- Rotation around 2 axes perpendicular to each other
- **Wrist joint**

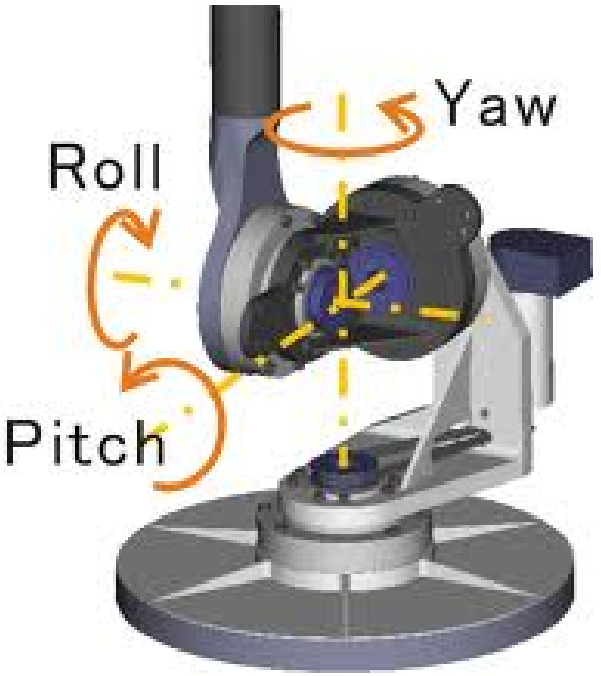


3DOF rotational joints

- Shoulder, hip, neck
- Two ways to represent the rotations
  - Gimbal joint (Euler Angles)
  - Free joint (Quaternions)

Gimbal joint: Euler Angles

- 3DOF joints
- Comes from Robotics
- 3DOF joints in robots were designed by connecting three motors pointing different axes

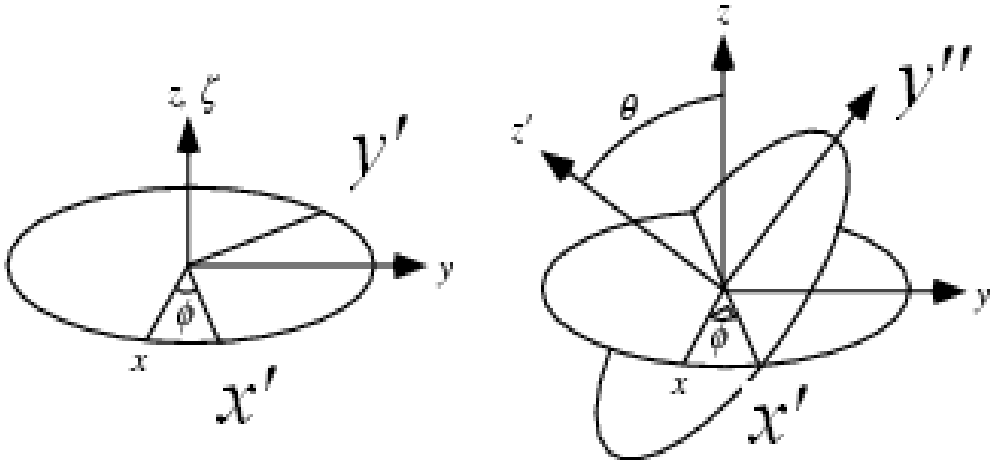


Gimbal joint: Euler Angles

- Rotation defined by the three axes and the angle of rotation around them
- the rotation order has to be specified such as

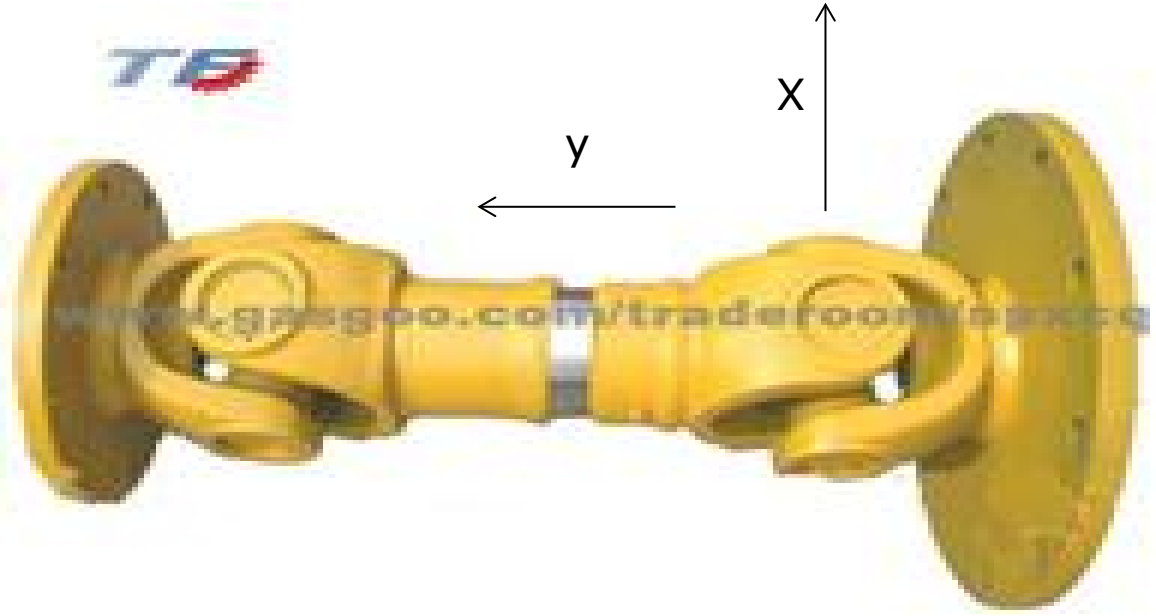
  X-Y-Z, Z-X-Y, Y-Z-X, etc

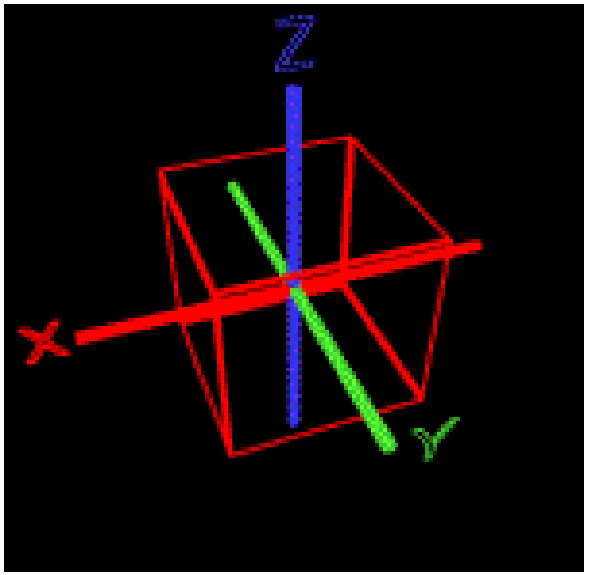
The one below is Z-X



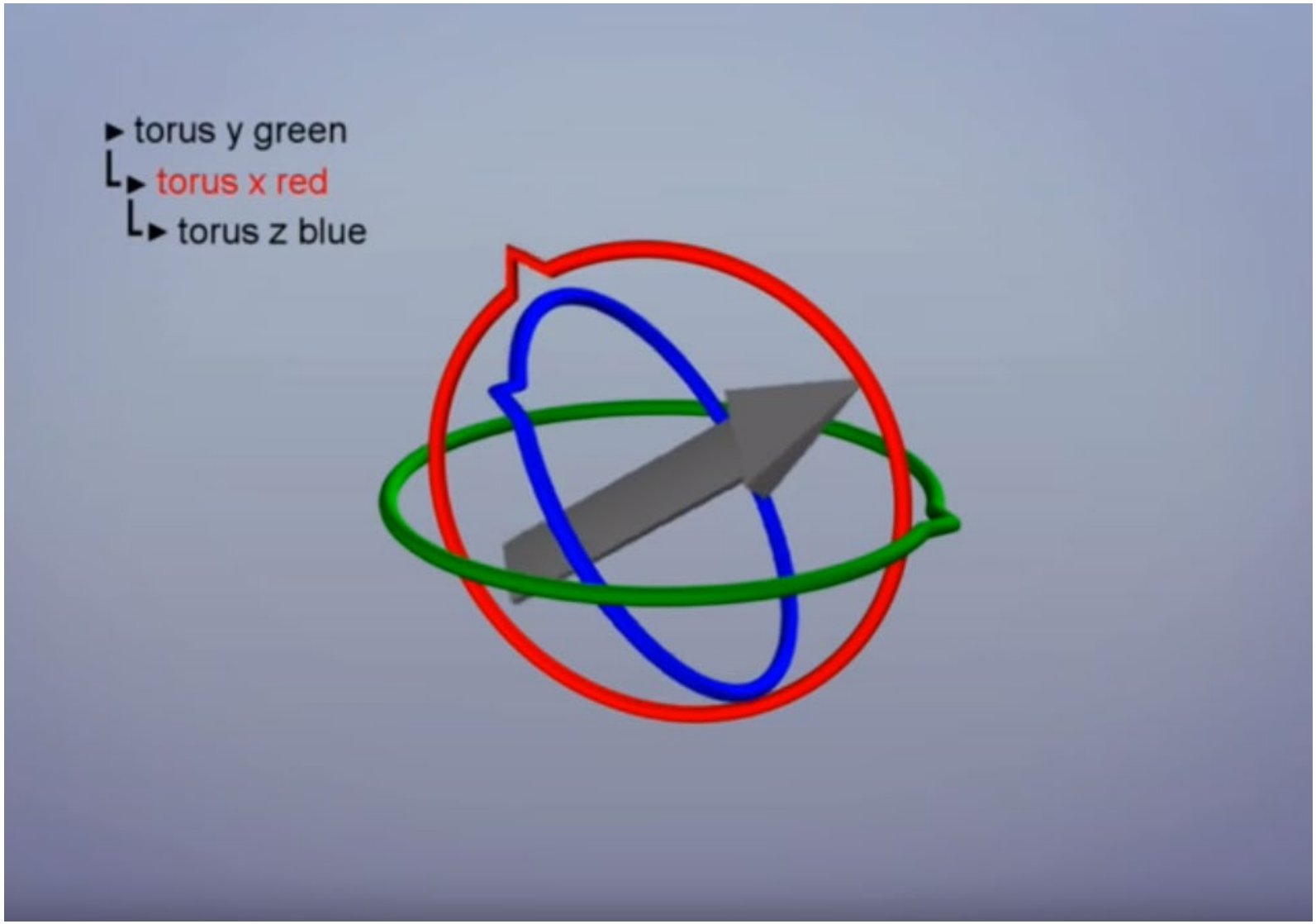
Problem: Gimbal Lock

- Two rotational axis of an object pointing in the same direction - 1DOF is lost
  - For example for rotation defined in the order of X-Y-Z
  - Gimbal lock occurs when rotating Y for 90 degrees.
  - X and Z axis get pointed down the same axis





Problem: Gimbal Lock



https://www.youtube.com/watch?v=zc8b2Jo7mno

Free joint

- A ball joint
- 3 DOF
- Do not have to worry about gimbal lock

Free joint: Quaternion

- Do not have to worry about gimbal lock
- The rotation is represented by a vector of four components (q_x, q_y, q_z, q_w)

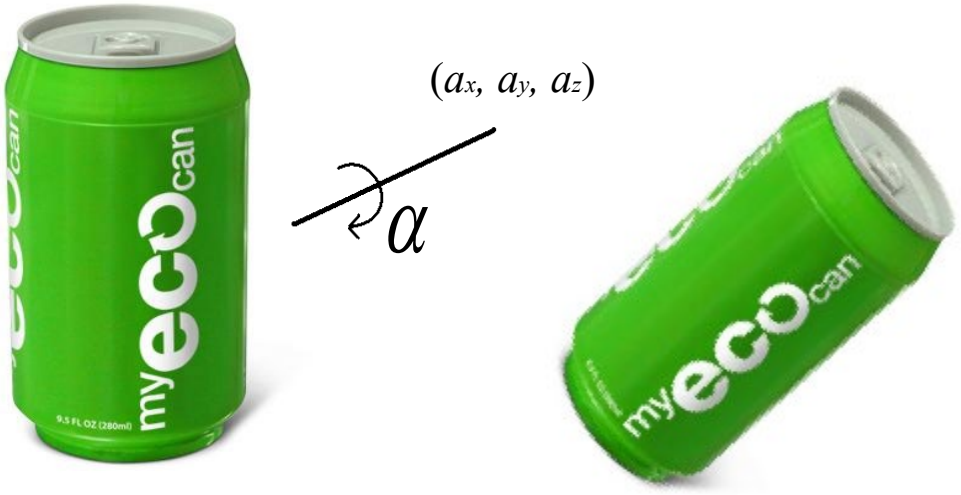
Free joint: Quaternion

- Do not have to worry about gimbal lock
- The rotation is represented by a vector of four components (q_x, q_y, q_z, q_w)

A rotation about the unit vector $\boldsymbol{q}$ by an angle alpha around an axis (a_x, a_y, a_z) makes a quaternion:

$$q_a(\alpha) = (q_x, q_y, q_z, q_w) = (a_x S\frac{\alpha}{2}, a_y S\frac{\alpha}{2}, a_z S\frac{\alpha}{2}, C\frac{\alpha}{2})$$

(S, C are sin, cos)

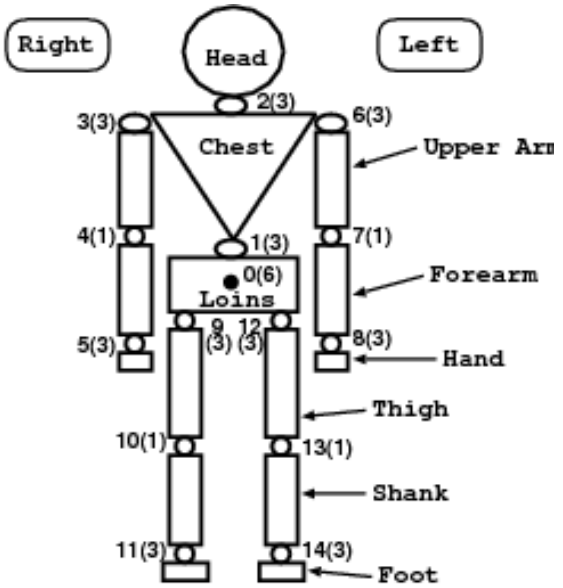


Generalized Coordinates

A vector to specify the posture of the body

$$q=(q_1,q_2,q_3,q_4,q_5,q_6,q_7,\ldots,q_n)$$

Usually, the first three numbers: *location of the root*
The next three numbers: *orientation of root*
The rest: *the joint angles of the body*



Overview

- Character Animation
  - Posture representation
    - Hierarchical structure of the body
    - Joint types
      - Translational, hinge, universal, gimbal, free
    - Euler angles
      - Gimbal lock
    - Quaternions
    - Coordinate Transformations
- **Creating the animation (Keyframe animation)**
  - Interpolation
  - Inverse Kinematics
    - Analytical
    - CCD
    - Pseudo Inverse

Keyframe Animation

- The keyframe postures are designed by the animator
- The inbetween motion is created by interpolation

Keyframe Design

- Each postures are created by the user interface
  - Forward Kinematics (FK, Virtual Track Ball)
    - The user clicks the segment and rotates it around the joint origin
    - The movement of the mouse is mapped to the rotation of the joint
    - Example: https://www.youtube.com/watch?v=pGuR4xyi7MQ
  - Inverse Kinematics (IK)
    - The user clicks the segment and drags it in the 3D coordinate
    - The motion of the mouse is mapped to the translation in 3D coordinate
    - The movement of the segment is achieved by moving each joint of the body
    - Example: http://www.youtube.com/watch?v=--5hWniftyI
    - Interactive Demo: http://www.starke-consult.de/BioIK-Demo/index.html

Keyframe animation by Poser

- Poser is a commercial software to generate human animation
- There is another free software called MikuMikuDance
- Blender plugin called “Manuel Bastioni Labs”
- Unity3D In-Built IK & Keyframe-Animation

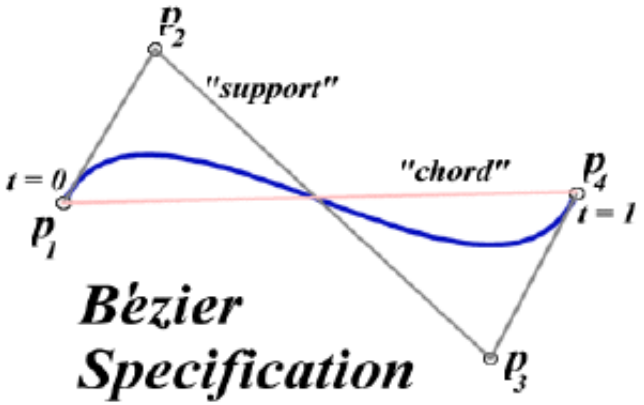
https://www.youtube.com/watch?v=QAXOdj8mr7s

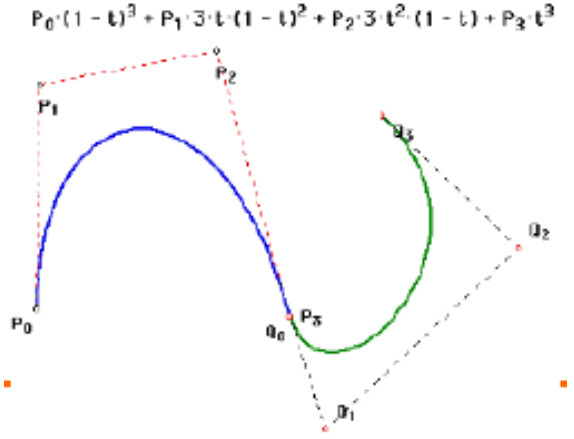
Interpolation

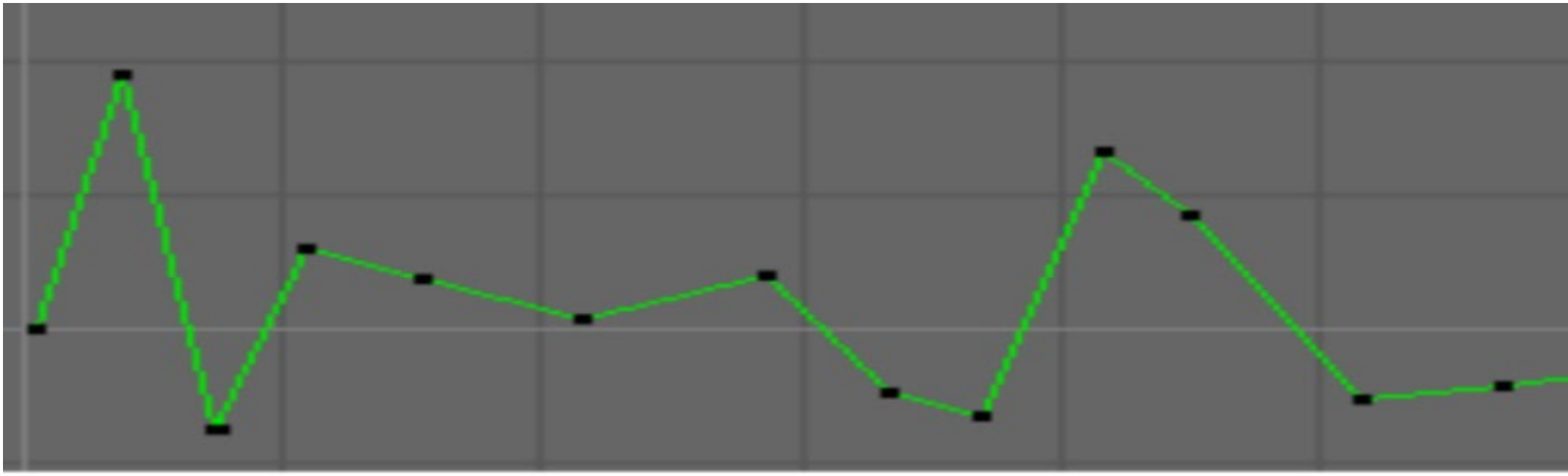
The generalized coordinates can be interpolated by

- Linear interpolation

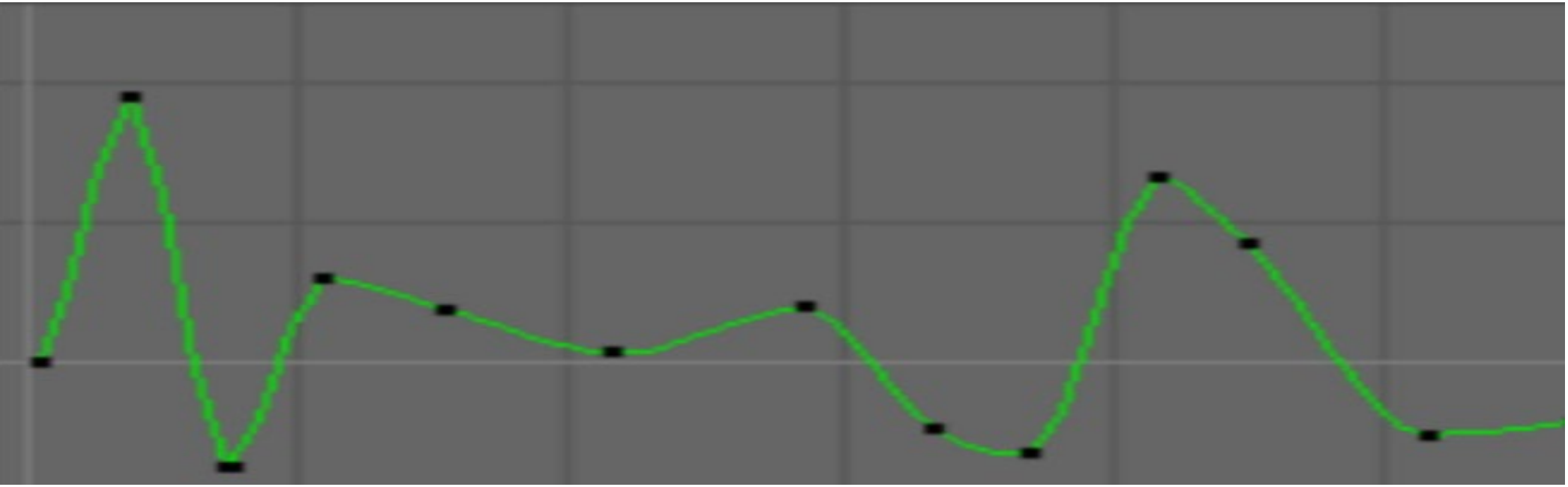
- High-order polynomials (e.g. Bsplines, Bezier)
  Bezier: 2 end points, two points to control the tangent vector







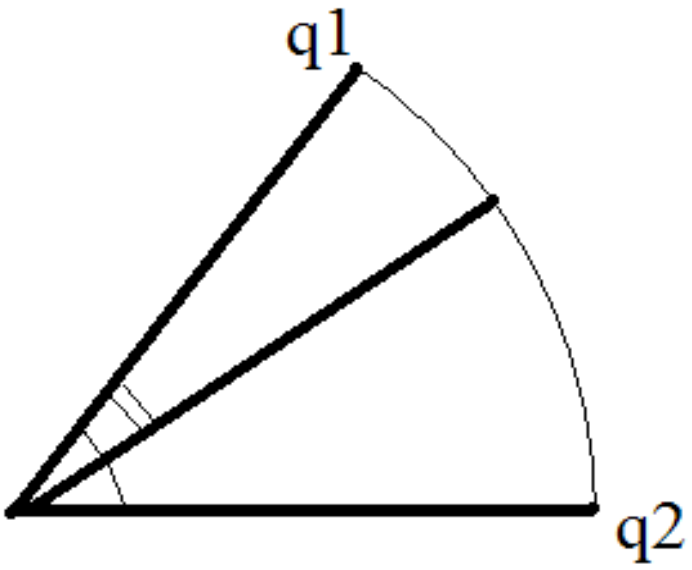
Linear interpolation



Bsplines

Interpolation of Quaternions

- Interpolation of two rotations (SLERP)
- Changing the orientation from q1 to q2 by rotating around a single axis ***u***
- angle of rotation around ***u*** to change from q1 to q2



Problems with interpolating the generalized coordinates

- Important constraints might not be satisfied

→ The feet on the ground can slide

→ Mainly reasoned by interpolating the root wrong
  → "Error-propagation" along the hierarchy

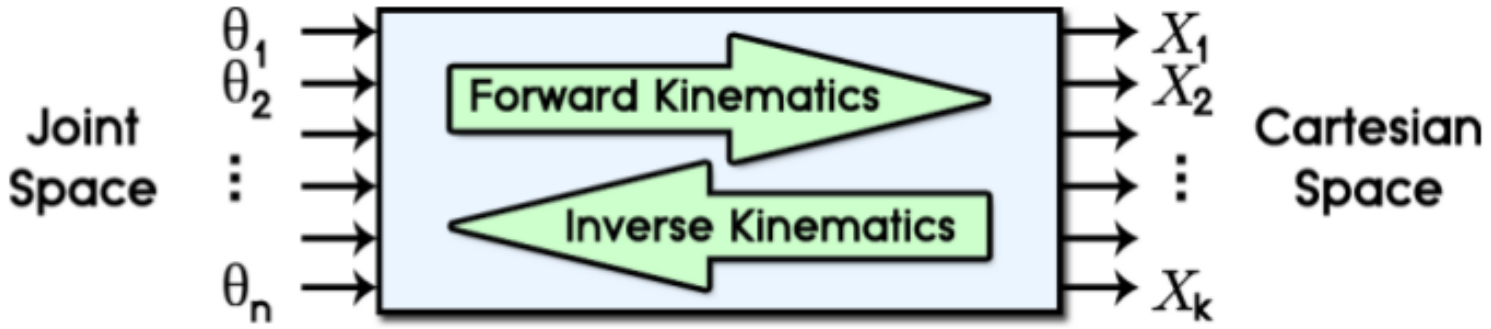
- Solution

→Specify the position of the joints and use IK
  to calcualte the joint angles

Forward and Inverse Kinematics

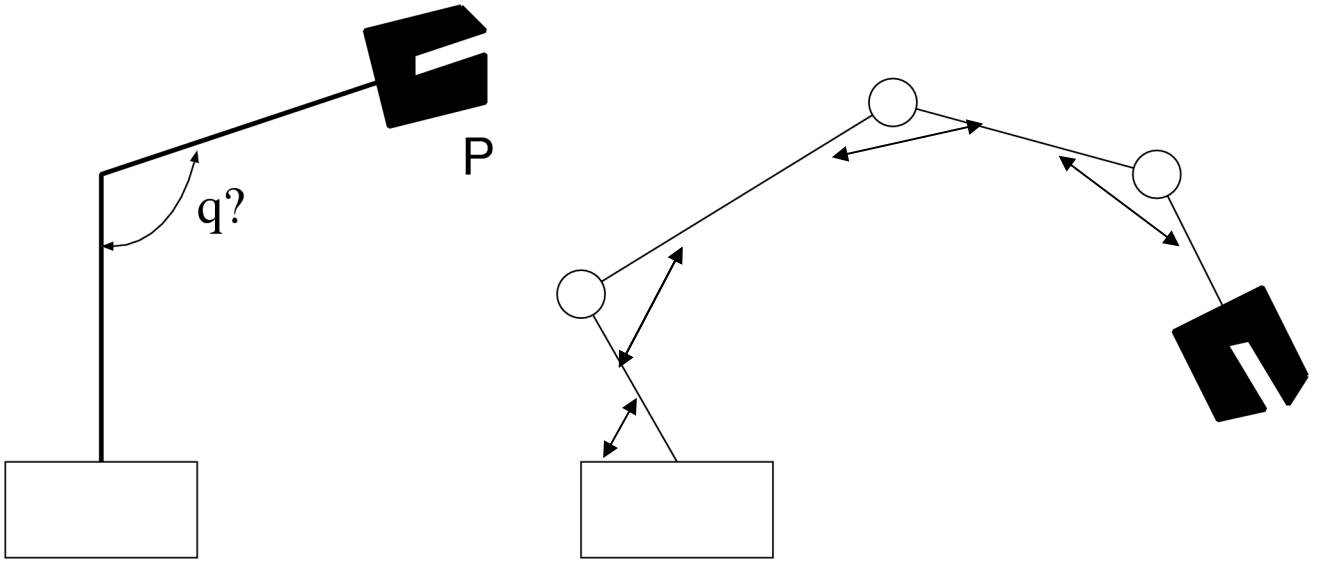
- Forward kinematics solution is straightforward to compute and unique. However, hard to control.
  → "We don't know the joint angles to reach a location."

- Inverse kinematics can have zero to infinite solutions. Solutions tell us the joint angles.



Inverse Kinematics Problem

- Calculate the joint angles from the position of the end effector



Inverse Kinematics

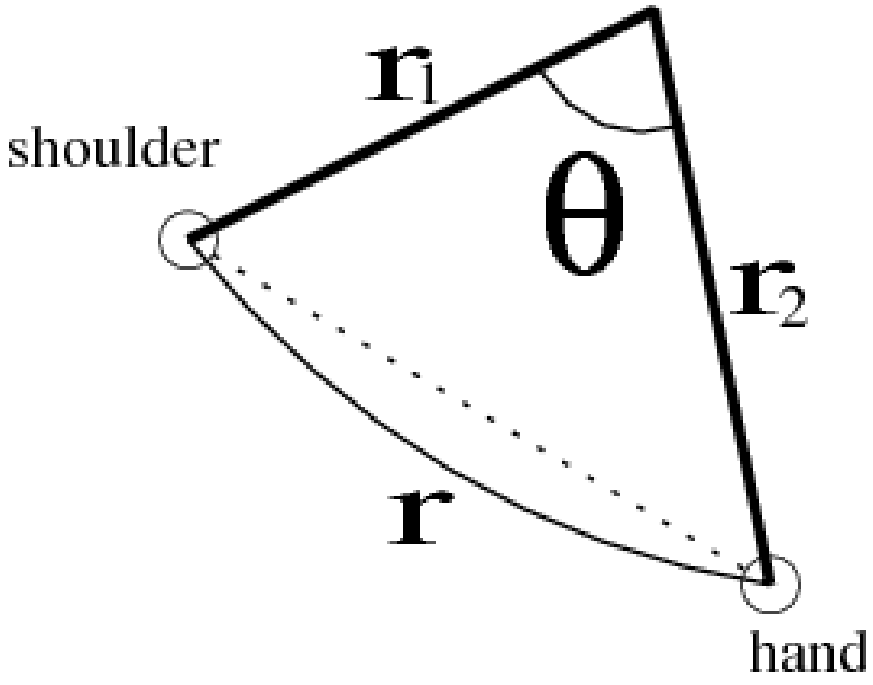
- Forward Kinematics: calculating the joint positions from the joint angles
- **Inverse Kinematics : Calculate the joint angles based on the joint positions**

  - Many different approaches
    - Analytical approaches (analytical solution exists)
    - CCD (Cyclic Coordinate Descent)
    - Jacobian-based methods (compute by optimization)
    - http://thewanderingtech.50webs.com/Flash/IK%20Comparison%20Application/IKCompare.html
  - Originated from robotics, but also essential in animation. Main difference: speed vs. accuracy

Analytical Approaches

- Using an analytical solver for calculating the joint angles
- e.g. suppose the positions of the wrist and shoulder are given, calculate the elbow angle

$$\theta = \arccos \frac{r_1^2 + r_2^2 - r^2}{2 r_1 r_2}$$



Cyclic-Coordinate Descent

- Moving the joints closest to the end effectors first and minimize the distance between the end effector and the target
- Move up the hierarchy and move the next joint to minimize the distance between the end effector and the target
- Repeat the process until the base is reached
- Move to the first joint again and repeat the same process until the end effector reaches the target or max iterations reached
  - This is needed to handle cases that the target is not reachable
  - 2D Tutorial: http://www.ryanjuckett.com/programming/cyclic-coordinate-descent-in-2d/

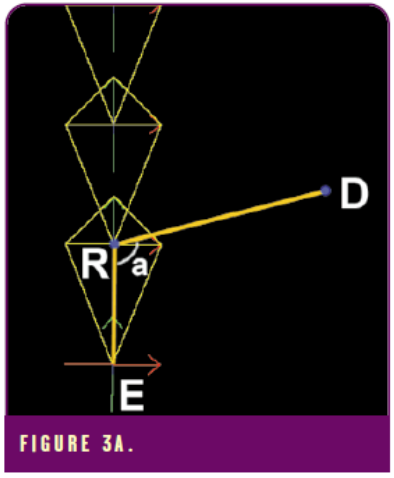


FIGURE 3A.

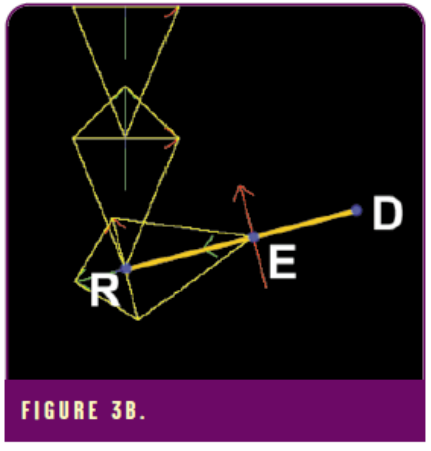


FIGURE 3B.

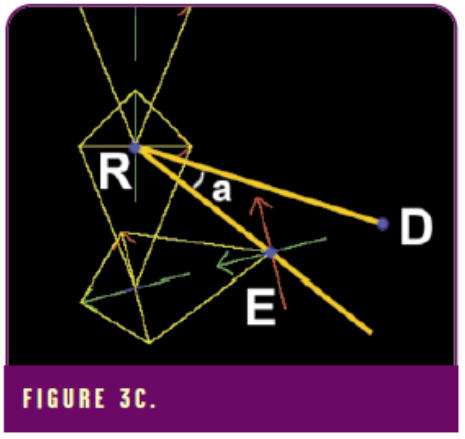


FIGURE 3C.

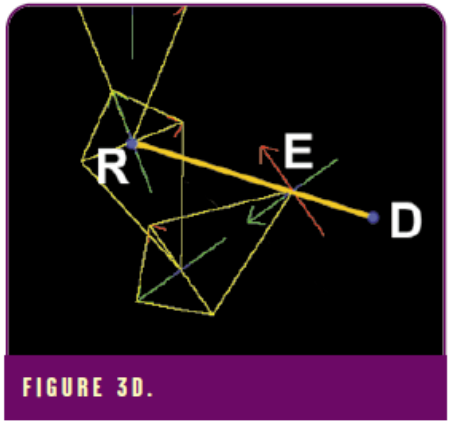


FIGURE 3D.

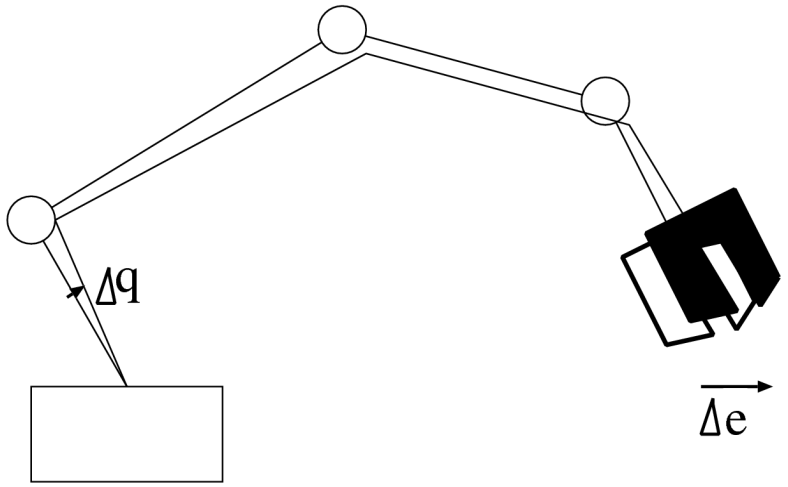
Pseudo-inverse method

- The relationship between the end effector position and the joint angles are non-linear

$$\mathbf{e} = f(\mathbf{q})$$

- But for small movements, their relation can be considered linear

$$\Delta\mathbf{e} = \mathbf{J}\,\Delta\mathbf{q}$$



Jacobian matrix

- Correlates the movement of the end effector Δe with movements of the joints $\mathbf{\Delta q_i}$
- Each column describing how much $\mathbf{\Delta e}$ changes when the $\mathbf{\Delta} qi$ is changed

$$\Delta e = \mathbf{J}\Delta q$$

$$\text{Jacobian} \quad J = \frac{\partial e}{\partial q} \approx \begin{pmatrix} \frac{\Delta e_x}{\Delta q_1} & \cdots & \frac{\Delta e_x}{\Delta q_n} \\ \frac{\Delta e_y}{\Delta q_1} & \cdots & \frac{\Delta e_y}{\Delta q_n} \\ \frac{\Delta e_z}{\Delta q_1} & \cdots & \frac{\Delta e_z}{\Delta q_n} \end{pmatrix}$$

Pseudo-inverse method

- The relationship between the end effector position and the joint angles are non-linear

$$\mathbf{e} = f(\mathbf{q})$$

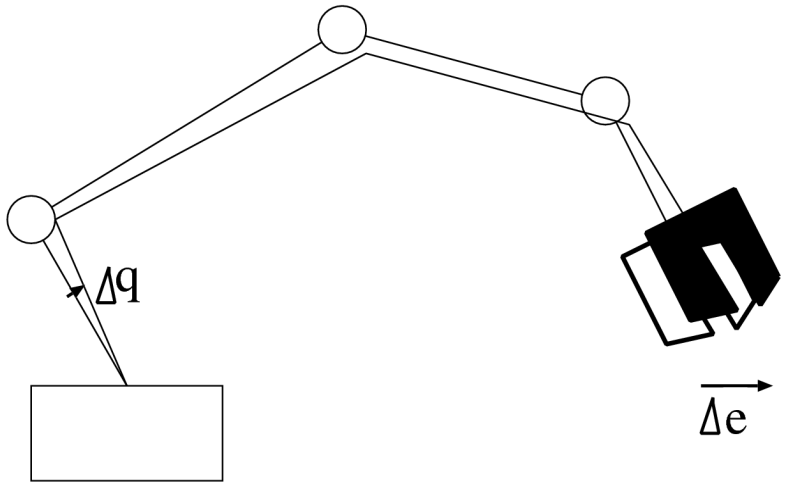
- But for small movements, their relation can be considered linear

$$\Delta\mathbf{e} = \mathbf{J}\,\Delta\mathbf{q}$$

This is Forward mapping

Inverse mapping?

$$\Delta\mathbf{e} \longrightarrow \Delta\mathbf{q}$$



Pseudo Inverse Matrix

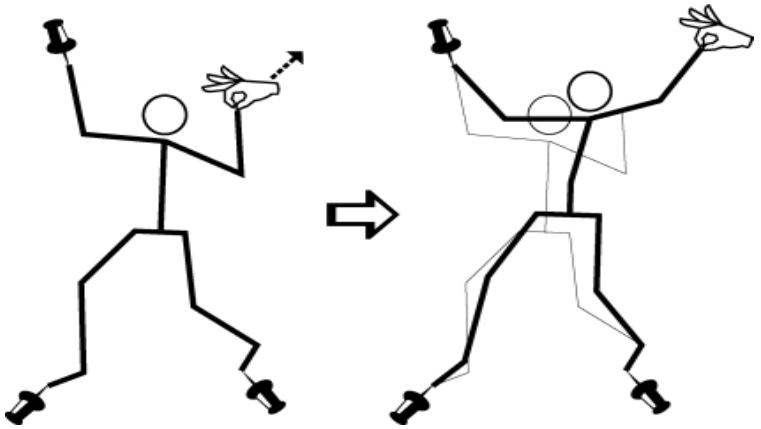
- The minimal joint angle movements that achieves the end effector movement e can be computed by the pseudo inverse matrix

Pseudo-inverse Method

Iteratively updating the generalized coordinates so that the position constraints are satisfied

e: end effector position

g: target location



while (Δe is too far from **g**) {

compute the Jacobian matrix **J**

$$\Delta \mathbf{e} = \mathbf{J}\Delta\mathbf{q}$$

compute the pseudoinverse of the Jacobian matrix $\mathbf{J}^{+}$

compute change in joint DOFs: $\Delta\mathbf{q} = \mathbf{J}^{+}\Delta\mathbf{e}$

apply the change to DOFs, move a small step of

$$\mathbf{q} = \mathbf{q} + \Delta\mathbf{q}$$

}

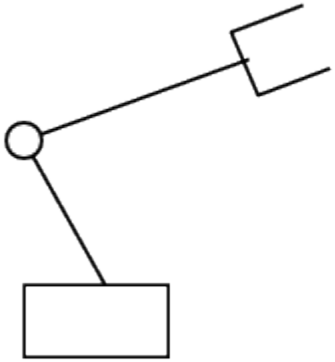
Singularity Problem

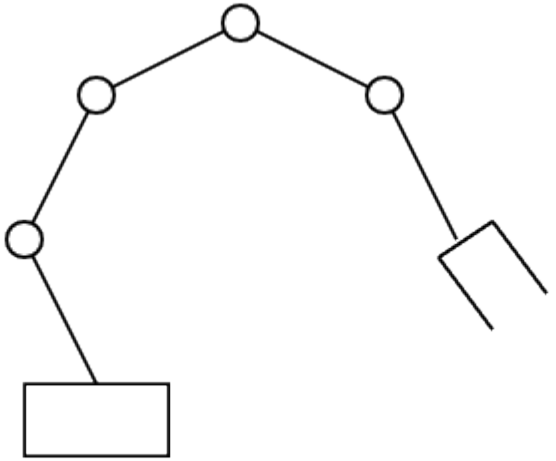
- There can be postures that the end effector cannot be moved to some directions
  - i.e. when all the joints are fully extended
- The system becomes unstable
- Test: http://thewanderingtech.50webs.com/Flash/IK%20Comparison%20Application/IKCompare.html

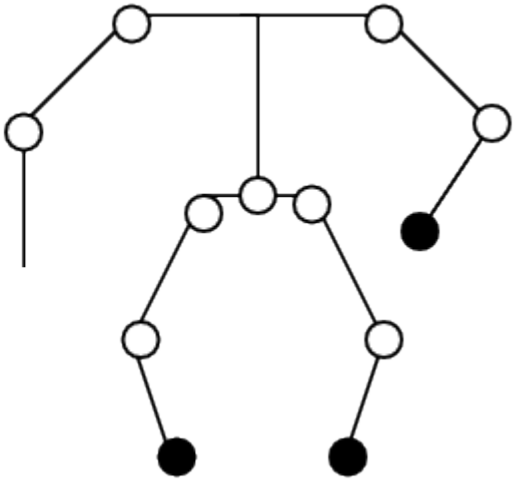
- Solution: **Damped Least Squares**

(imposing soft constraints): $J^{\star} = J^{T}(JJ^{T} + k^{2}I)^{-1}$

Which method will be good for controlling each of the below?







Cons and Pros of IK

- Analytical : Fast, but for most cases have no analytical solution
- CCD : simple, fast, easy to implement, can take into account the limit of the joint angles, may have oscillation problems
- Pseudo Inverse:
  - Used in robotics often, can handle any topological structure, multiple constraints
  - Can incorporate physics
  - Singularity problems (unstable when the limb is fully extended) -> damped least squares

Summary

- Representation of the Posture
  - Euler angles, generalized coordinates, quaternions
- Character Animation by Interpolation
- Inverse Kinematics
  - Analytical
  - CCD
  - Jacobian Pseudoinverse
  - Damped-Least-Squares

Links, Readings

- Poser http://poser.smithmicro.com/poser.html
- MikuMikuDance
  - https://sites.google.com/view/vpvp/
- About CCD http://graphics.cs.cmu.edu/nsp/course/15-464/Fall09/assignments/asst2/jlander_gamedev_nov98.pdf
- IK introduction
  - http://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf
- A robotics textbook (can be downloaded from within the university, see chapter 3)
  - http://link.springer.com/content/pdf/10.1007%2F978-1-84628-642-1.pdf