FACULTY OF ENGINEERING
AND ARCHITECTURE

E016712: Computer Graphics

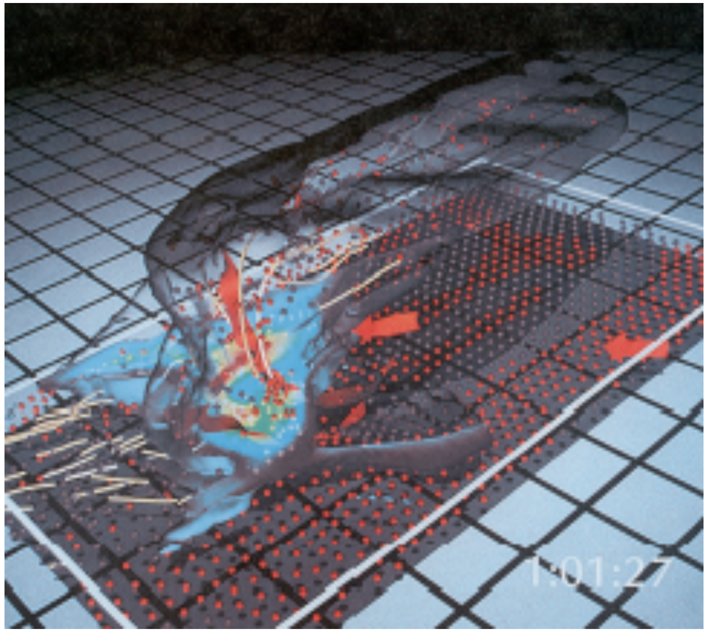
Animation Part 1

Lecturers: Aleksandra Pizurica and Danilo Babin

GHENT
UNIVERSITY

Computer Animation

- **What is Animation?**
  - Make objects change over time according to scripted actions
  - **Computer animation** is the process used for generating animated images (moving images) using computer graphics



- **What is Simulation?**
  - Predict how objects change over time according to physical laws.

First animation

- Persistence of vision: discovered about 1800s
  - Zoetrope or "wheel of life"
  - Flip-book

Source: Wikipedia

Overview

- Animating using:
  - Key frames
  - Forward kinematics
  - Inverse kinematics
  - Hierarchical kinematics
  - Dynamics

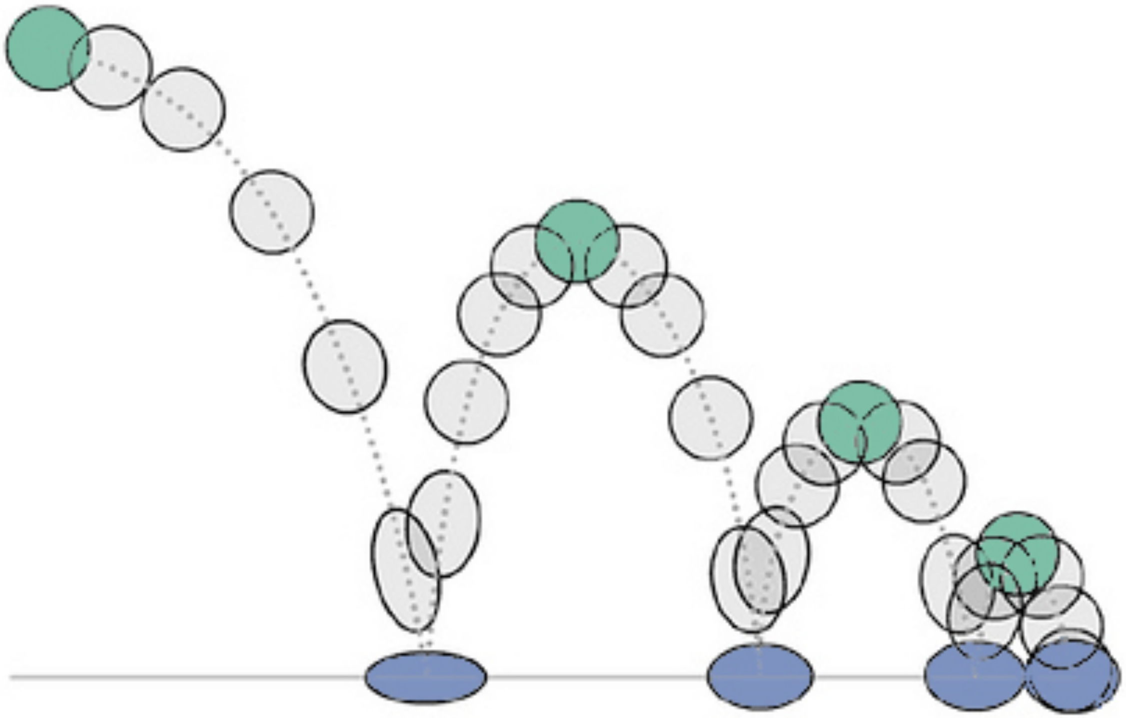
The material partially based on: E. Angel and D. Shreiner: Interactive Computer Graphics 6E © Addison-Wesley 2012

Keyframing

- *Keyframe* systems take their name from the traditional hierarchical production system first applied by Walt Disney
- Skilled animators would design or choreograph a particular sequence by drawing frames that established the animation - the so-called keyframes
- The production of the complete sequence was then passed on to less skilled artists who used the keyframes to produce ‘in-between’ frames

Keyframe animation

- Keyframe is a drawing (image) of a key moment in an animation sequence, where the motion is at its extreme
- Inbetweens fill the gaps between keyframes



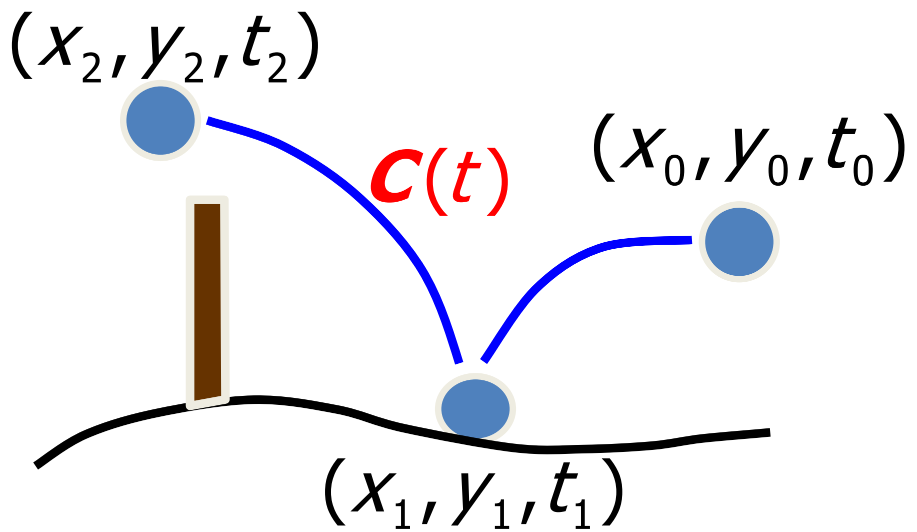
Keyframe animation

- In traditional animation, skilled animators draw keyframes; less experienced animators draw inbetweens
- In 3D computer animations, animators set up parameter values for keyframes;
- Software interpolates parameter values between keyframes for inbetweens
- Every motion is created by animators

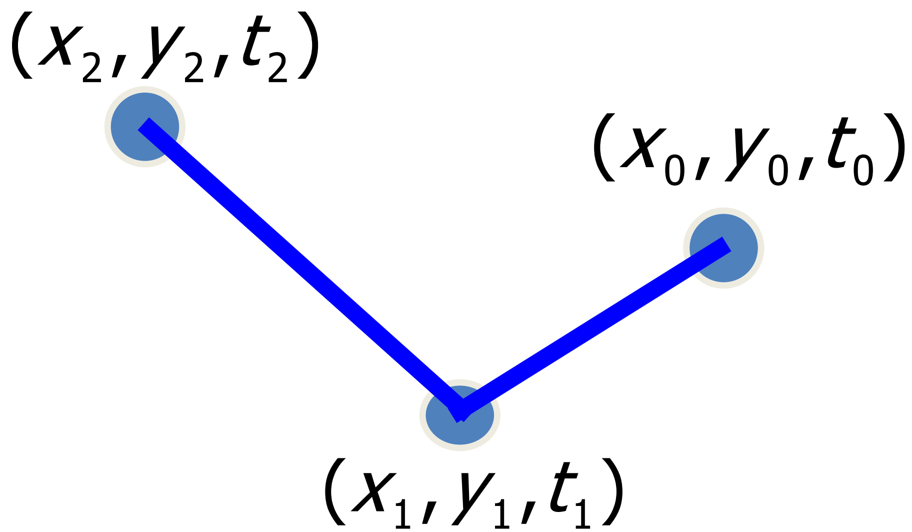
Inbetweening: interpolating positions

- Given positions: $(x_i, y_i, t_i),\ i = 0, \ldots, n$
- find a curve $\boldsymbol{c}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$
- such that

$$\boldsymbol{c}(t_i) = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$



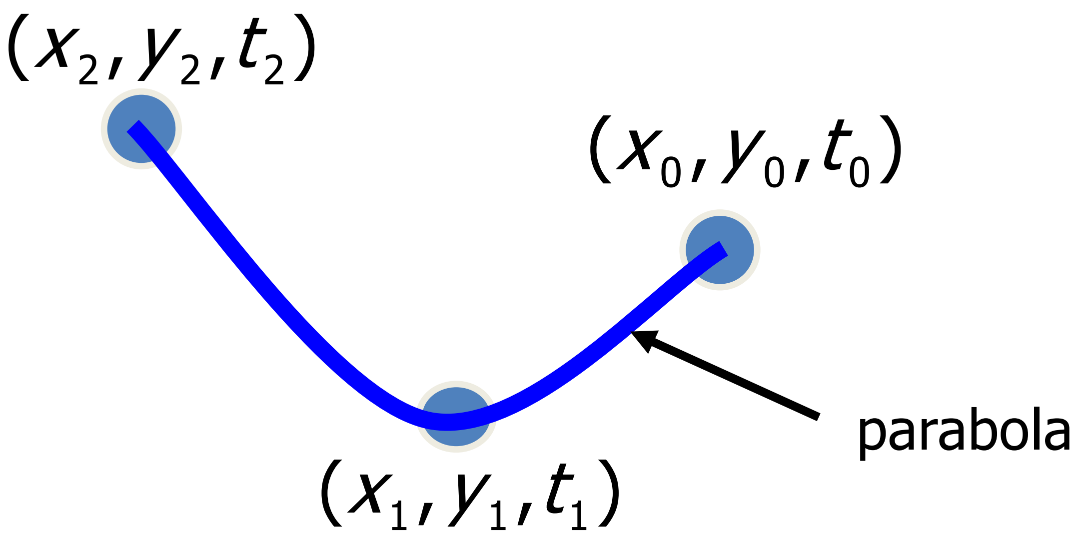
Linear Interpolation



t_0=0 and t_1=1 $\quad x(t) = x_0(1-t) + x_1 t$

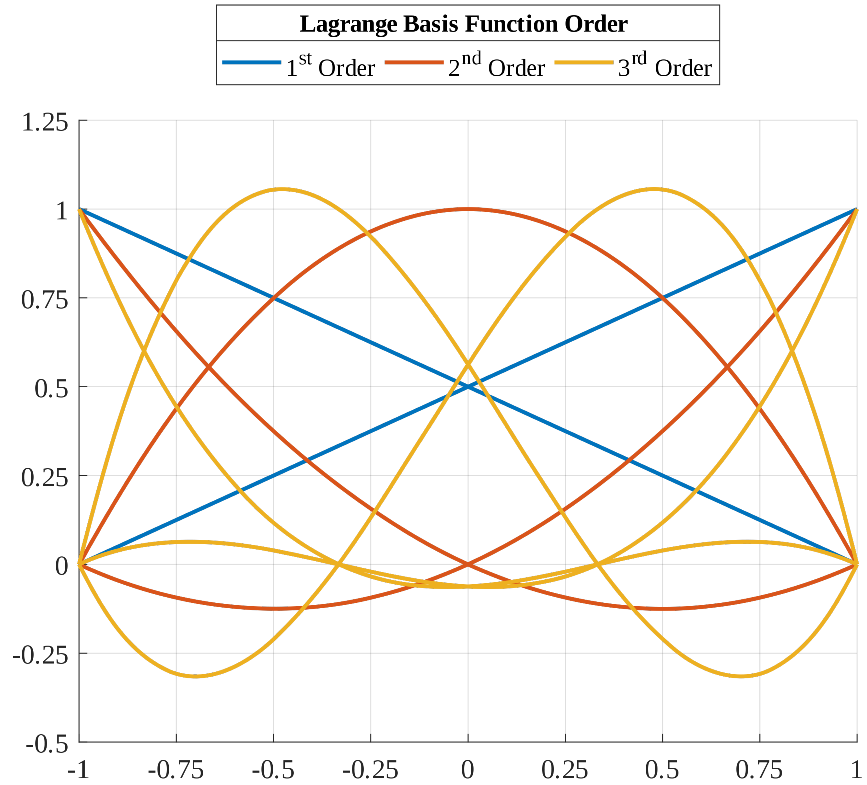
$$x(t) = \begin{cases} \dfrac{t_1 - t}{t_1 - t_0} x_0 + \dfrac{t - t_0}{t_1 - t_0} x_1, & t \in [t_0, t_1) \\ \dfrac{t_2 - t}{t_2 - t_1} x_1 + \dfrac{t - t_1}{t_2 - t_1} x_2, & t \in [t_1, t_2] \end{cases}$$

Polynomial Interpolation



- An n-degree polynomial can interpolate any n+1 points.
- The Lagrange formula gives the n+1 coefficients of an n-degree polynomial that interpolates n+1 points.
- The resulting interpolating polynomials are called Lagrange polynomials.

Lagrange polynomials



Given a set of $k+1$ data points

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$$

where no two x_j are the same, the **interpolation polynomial in the Lagrange form** is a linear combination

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

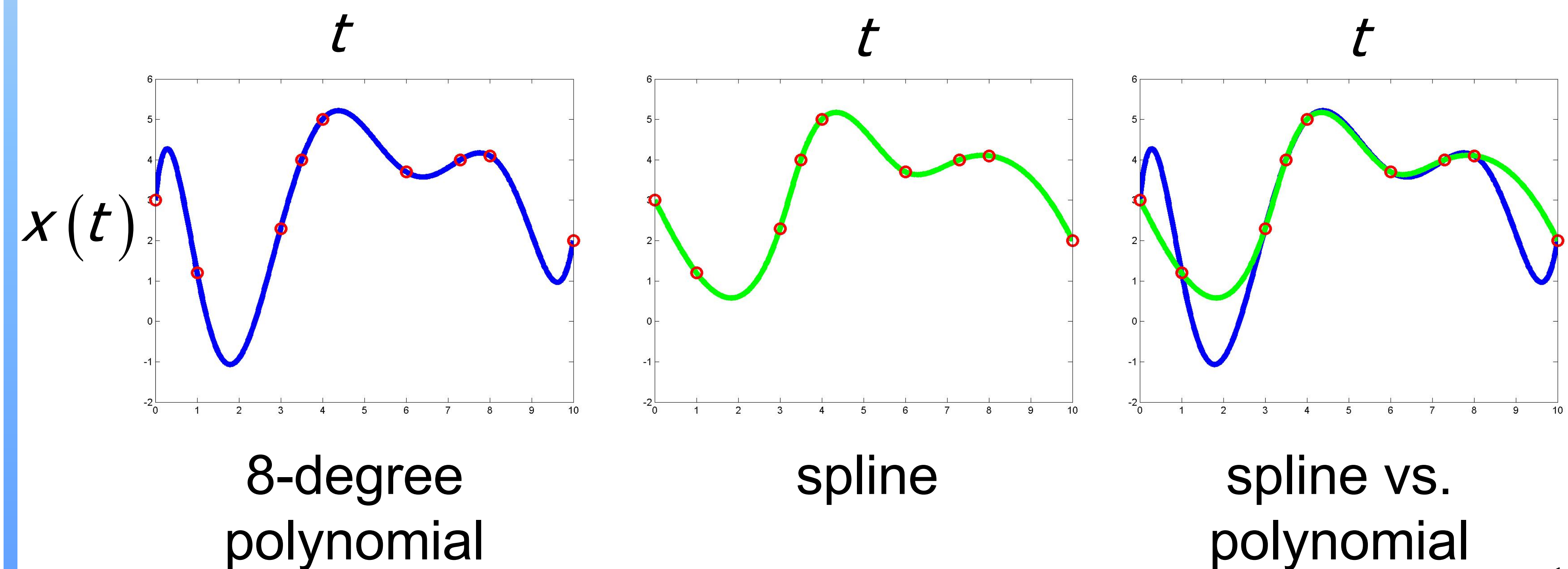
of Lagrange basis polynomials

$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)},$$

Source: Wikipedia

Spline Interpolation

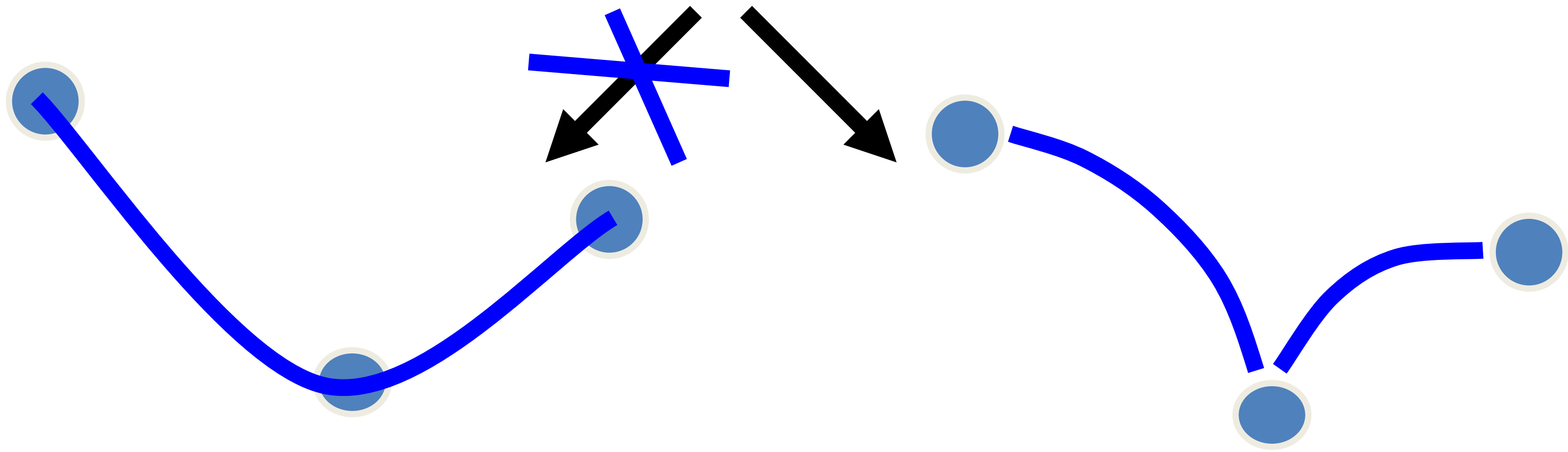
- Lagrange polynomials of small degree are fine but high degree polynomials are too wiggly.
- Spline (piecewise cubic polynomial) interpolation produces nicer interpolation. $x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$



Interpolation of Positions

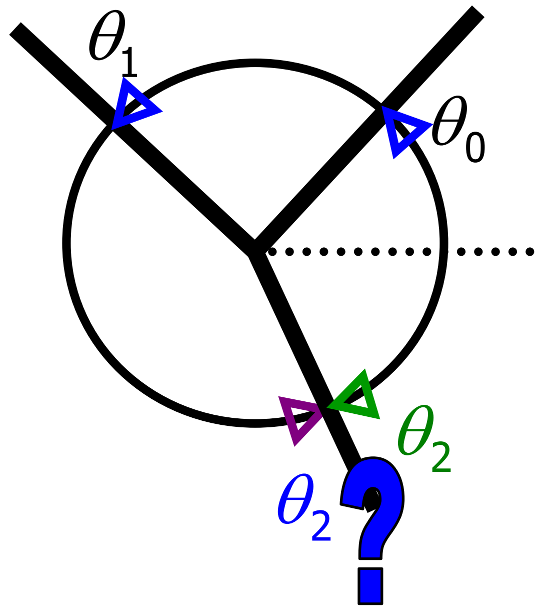
- We want to support general constraints: not just smooth velocity and acceleration.

- For example, a bouncing ball does not always have continuous velocity:



Interpolating angles

- Given angles $(\theta_i, t_i),\ i = 0, \ldots, n$
- find curve $\theta(t)$
- such that $\theta(t_i) = \theta_i$



- Angle interpolation is ambiguous.
- Different angle measurements will produce different motion

View interpolation problem statement:

Solution:

View interpolation example

View interpolation example

Keyframing drawbacks

- The keyframing approach carries certain disadvantages:
  - It is suitable for simple motion of rigid bodies
  - Care must be taken to ensure that no unwanted motion is introduced by the interpolation.

- None the less, interpolation of key frames remains fundamental to many animation systems

Kinematics and Dynamics

- Kinematics:
  - Motion parameters such as position, velocity and acceleration are specified without reference to the forces.
- Inverse kinematics:
  - Initial and final positions of objects at specified times and from that motion parameters .
- Dynamics:
  - The forces that produce the velocities and accelerations are specified (physically based modeling).
  - It uses laws such as Newton's laws of motion, Euler or Navier - Stokes equations.

Animating Articulated Structures

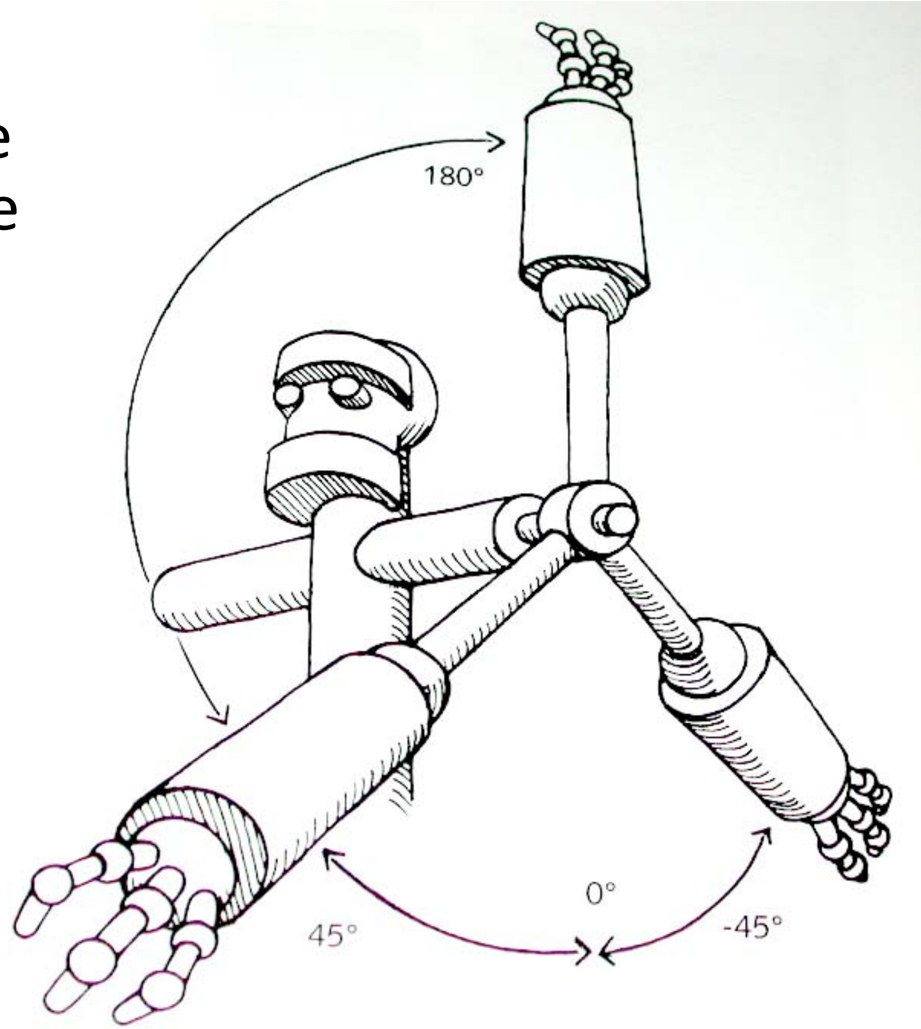
- The characters themselves are constructed out of skeletons which resemble the articulated structures found in robotics
- Articulated figure: a structure consisting of rigid links connected at joints
- Degrees of freedom (DOF): The number of independent joint variables specifying the state of the structure
- End Effector: end of a chain of links, e.g. a hand or a foot
- State vector: set of independent parameters which define a particular state of the articulated structure.
- E.g. state vector Q = (Q1, Q2, ..., QN) has N degrees of freedom.

Forward Kinematics

- In forward kinematics the motion of all the joints in the structure are explicitly specified which yields the end effector position

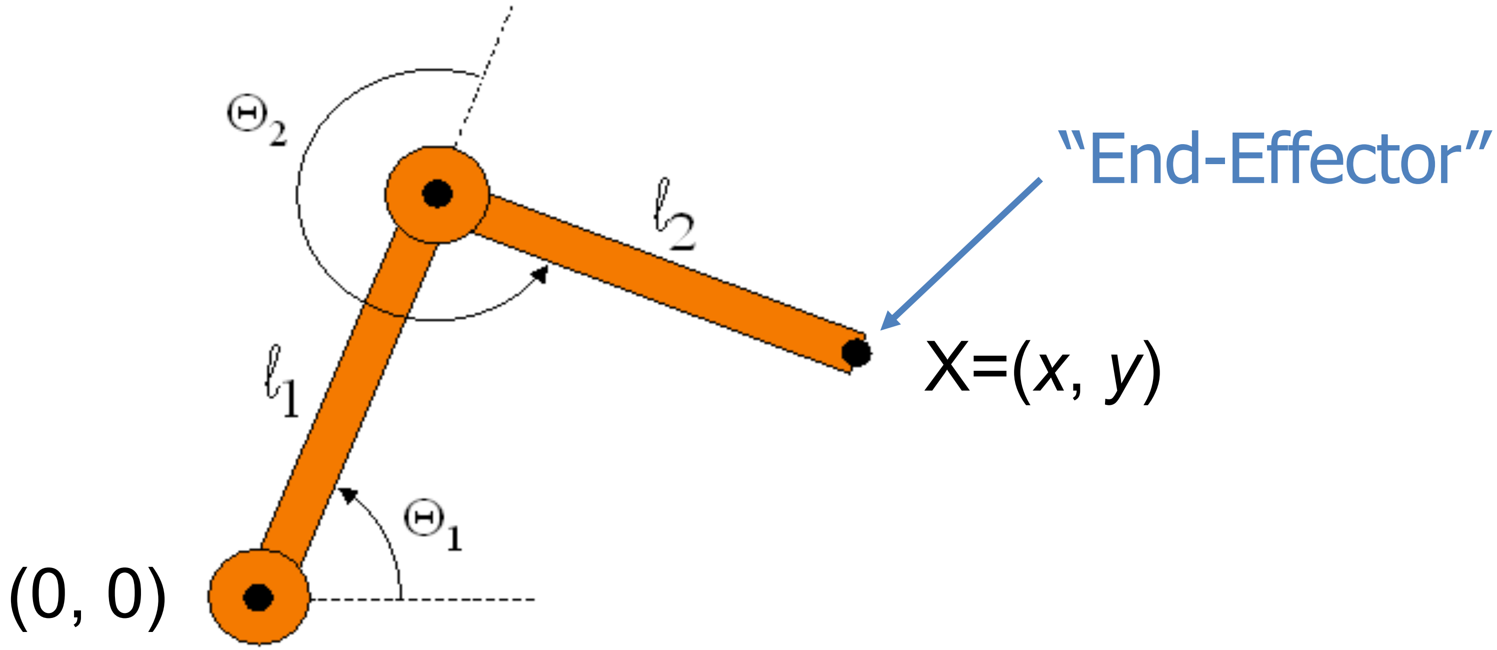
- The end effector position X is a function of the state vector of the structure:

$$X = f(Q)$$



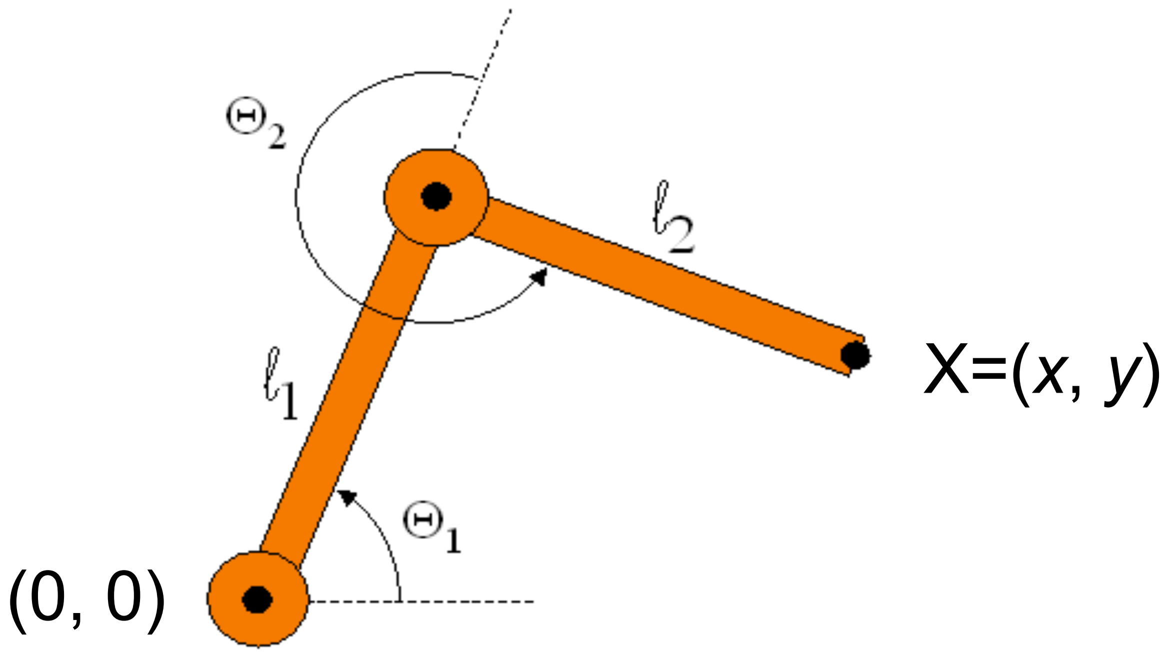
Example: 2-Link Structure

- Consider 2 links connected by rotational joints
- Links can only move in the plane of the page



Forward Kinematics

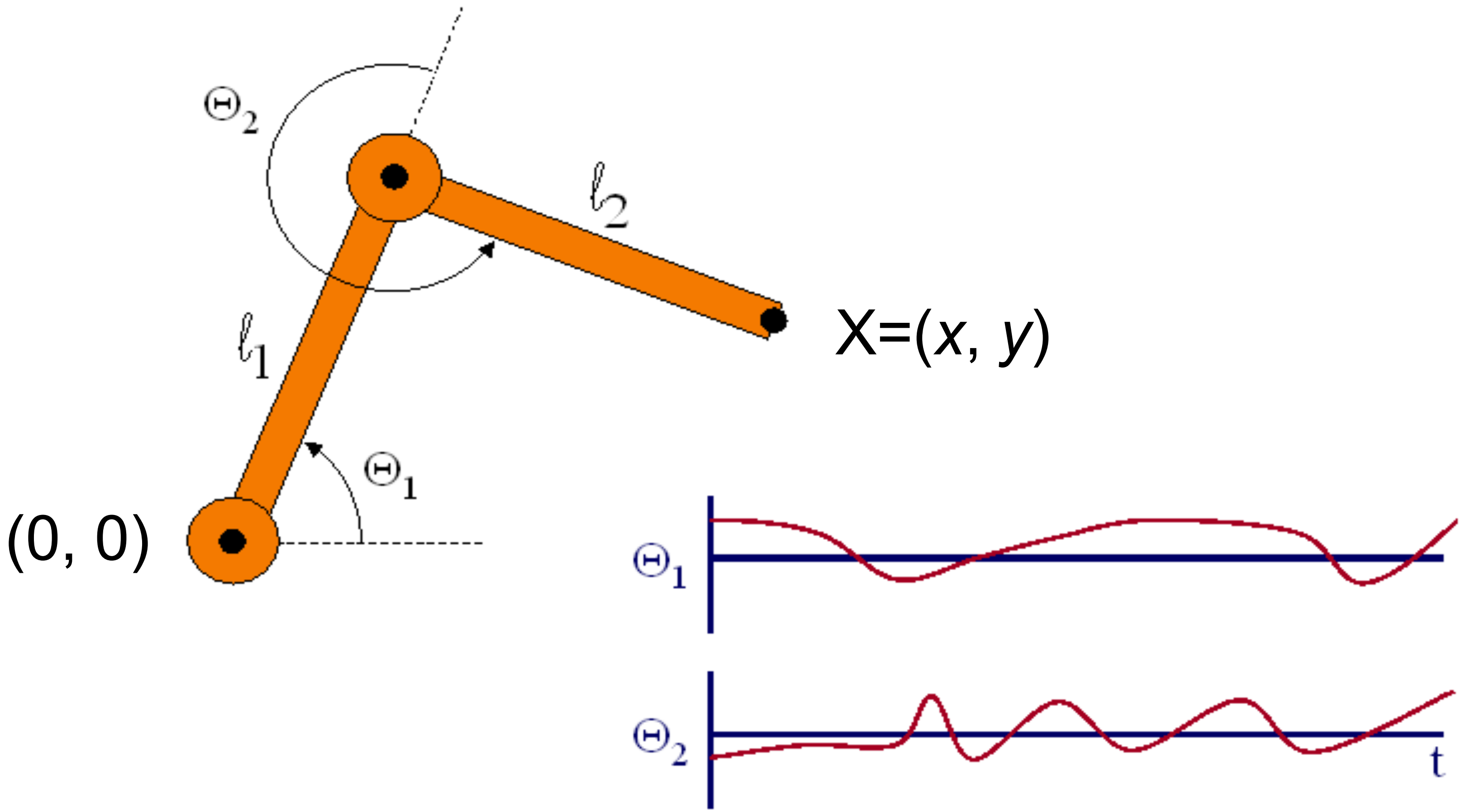
- Animator specifies joint angles: Θ_1 and Θ_2
- Computer finds positions of end-effector: X



$$X=(l_1\cos\Theta_1 + l_2\cos(\Theta_1+\Theta_2),\ l_1\sin\Theta_1 + l_2\sin(\Theta_1+\Theta_2))$$

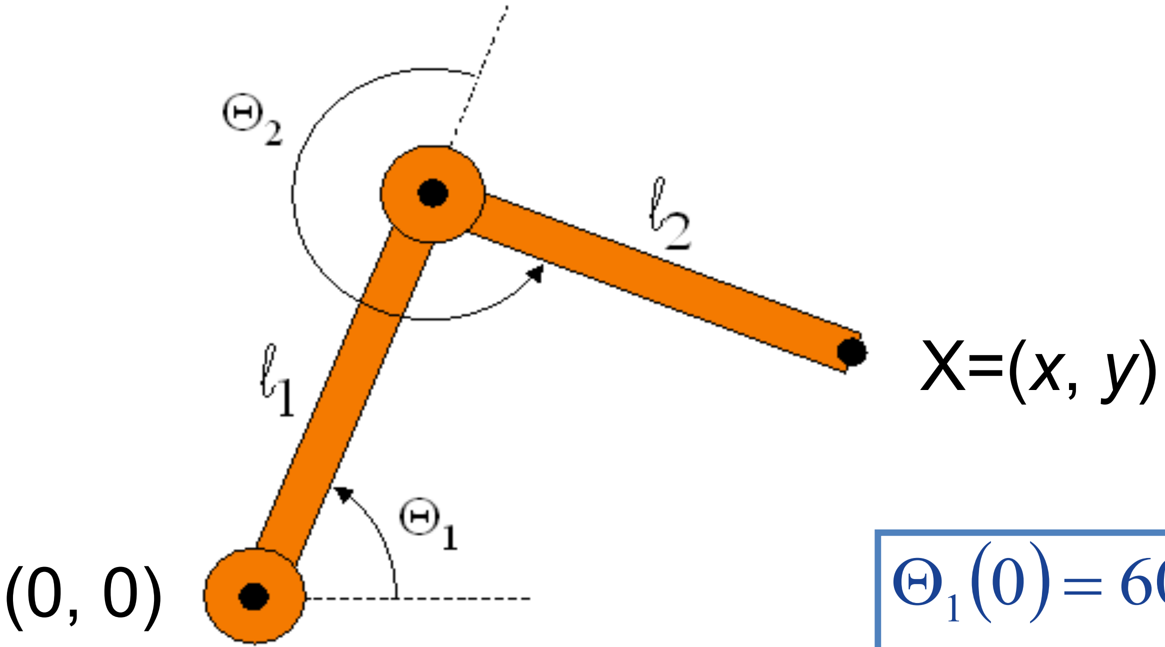
Forward Kinematics

- Joint motions can be specified by Spline Curves



Forward Kinematics

- Joint motions can be specified by initial conditions and velocities



$$\Theta_1(0) = 60° \qquad \Theta_2(0) = 250°$$

$$\frac{d\Theta_1}{dt} = 1.2 \qquad \frac{d\Theta_2}{dt} = -0.1$$

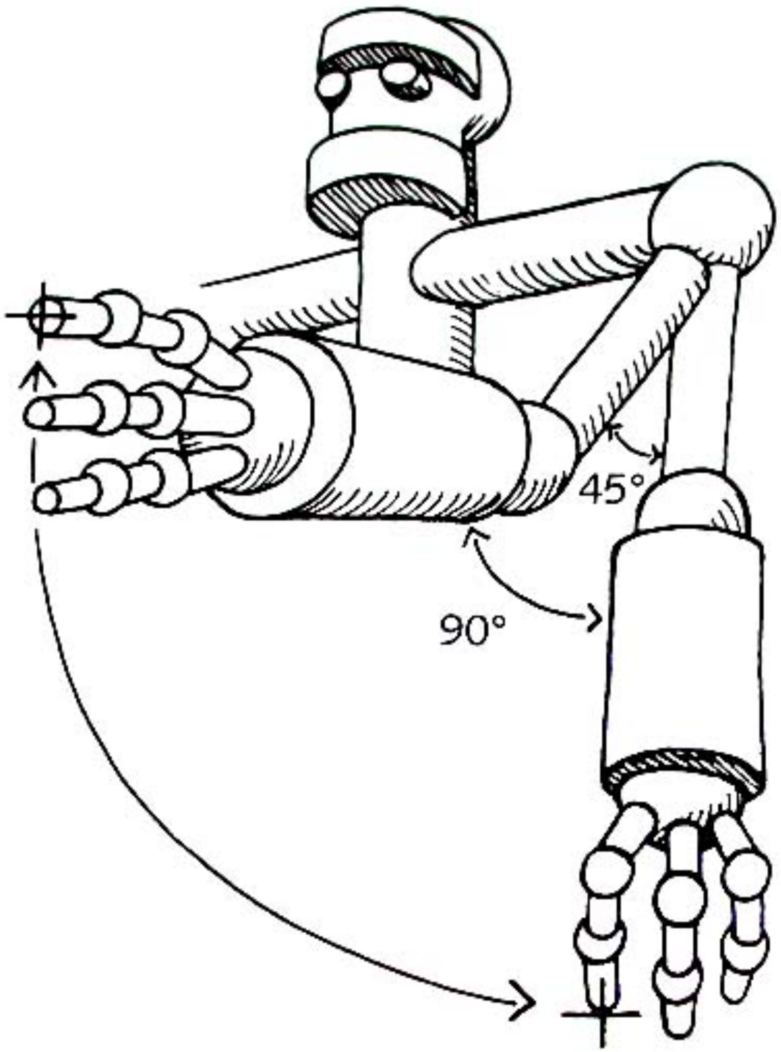
Inverse Kinematics

- In inverse kinematics (also known as "goal directed motion") the end effector's position is all that is defined
- Given the end effector position, we must derive the state vector of the structure which produced that end effector position
- Thus the state vector is given by:

$$Q = f^{-1}(X)$$

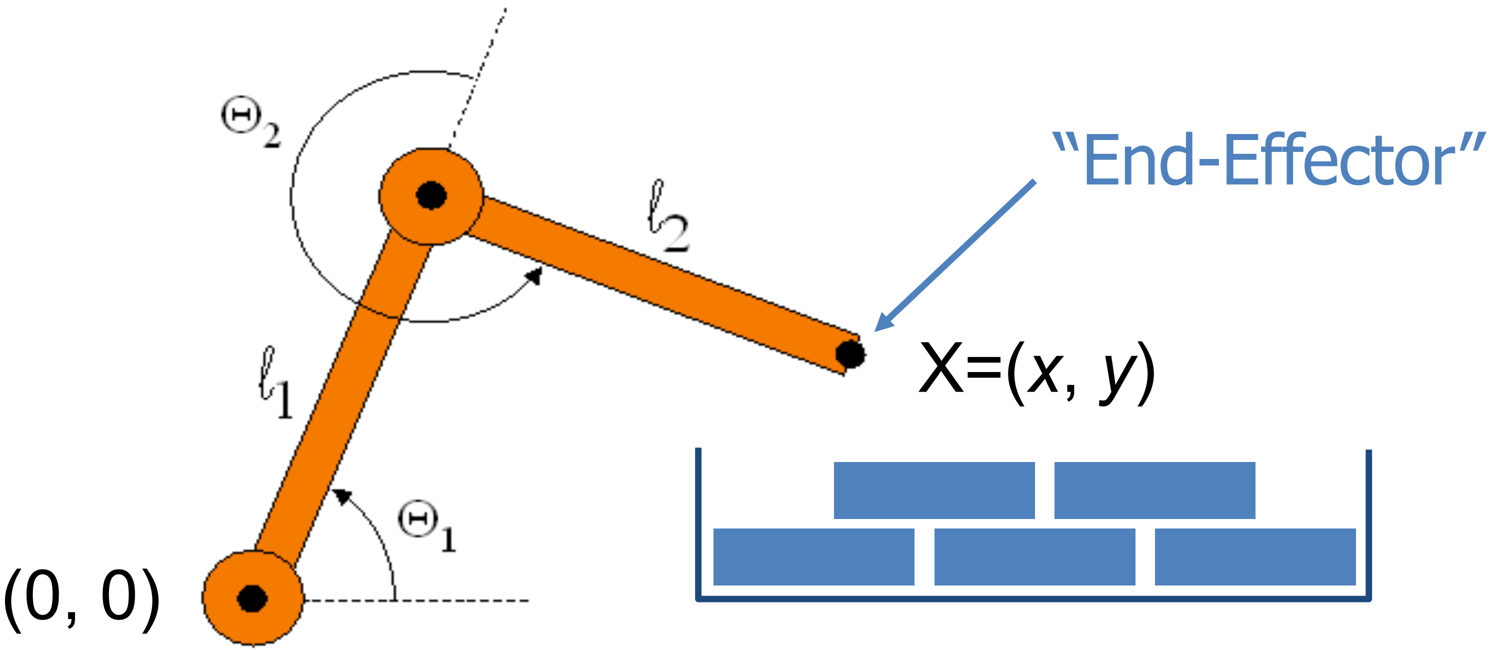
Inverse Kinematics

- Given the end-effector position (x,y) we can find the joint angles Θ1 and Θ2
  - Once again use simple geometry
- Increasing degrees of freedom allows more motion, but makes the geometry more difficult (for inverse kinematics, there will be multiple solutions)
- Suppose you want the robot to pick up a can of oil to drink. How?



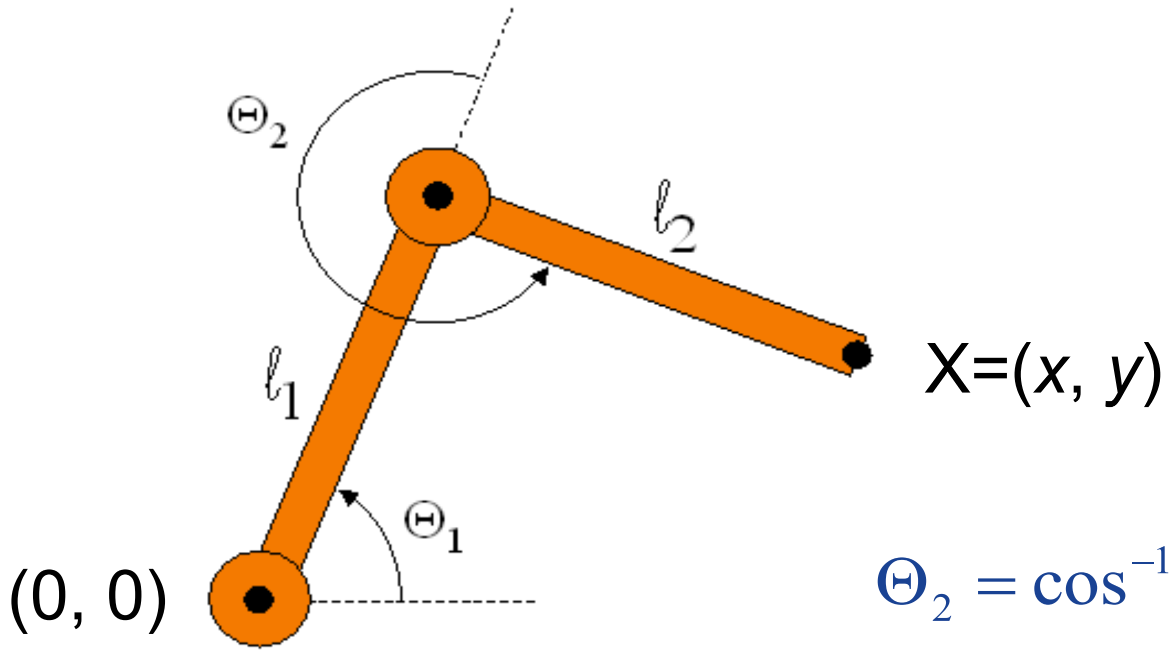
Example: 2-Link Structure

- What If Animator Knows Position of "End-Effector"



Inverse Kinematics

- Animator specifies end-effector position X
- Computer finds joint angles: Θ_1 and Θ_2

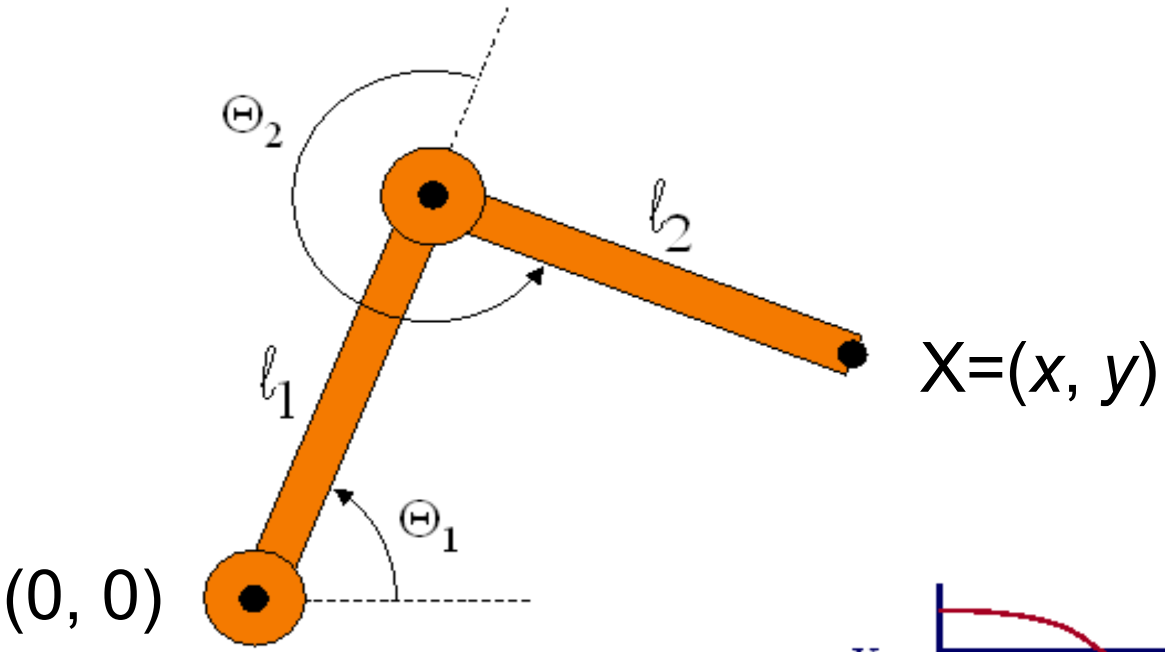


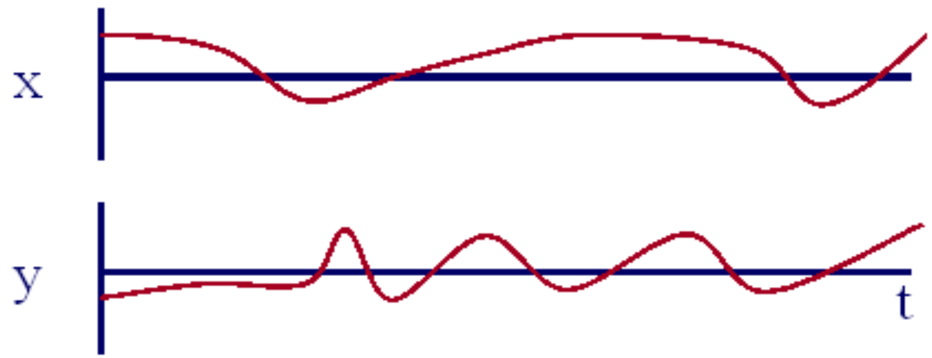
$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin \Theta_2)x + (l_1 + l_2 \cos \Theta_2)y}{(l_2 \sin \Theta_2)y + (l_1 + l_2 \cos \Theta_2)x}$$

Inverse Kinematics

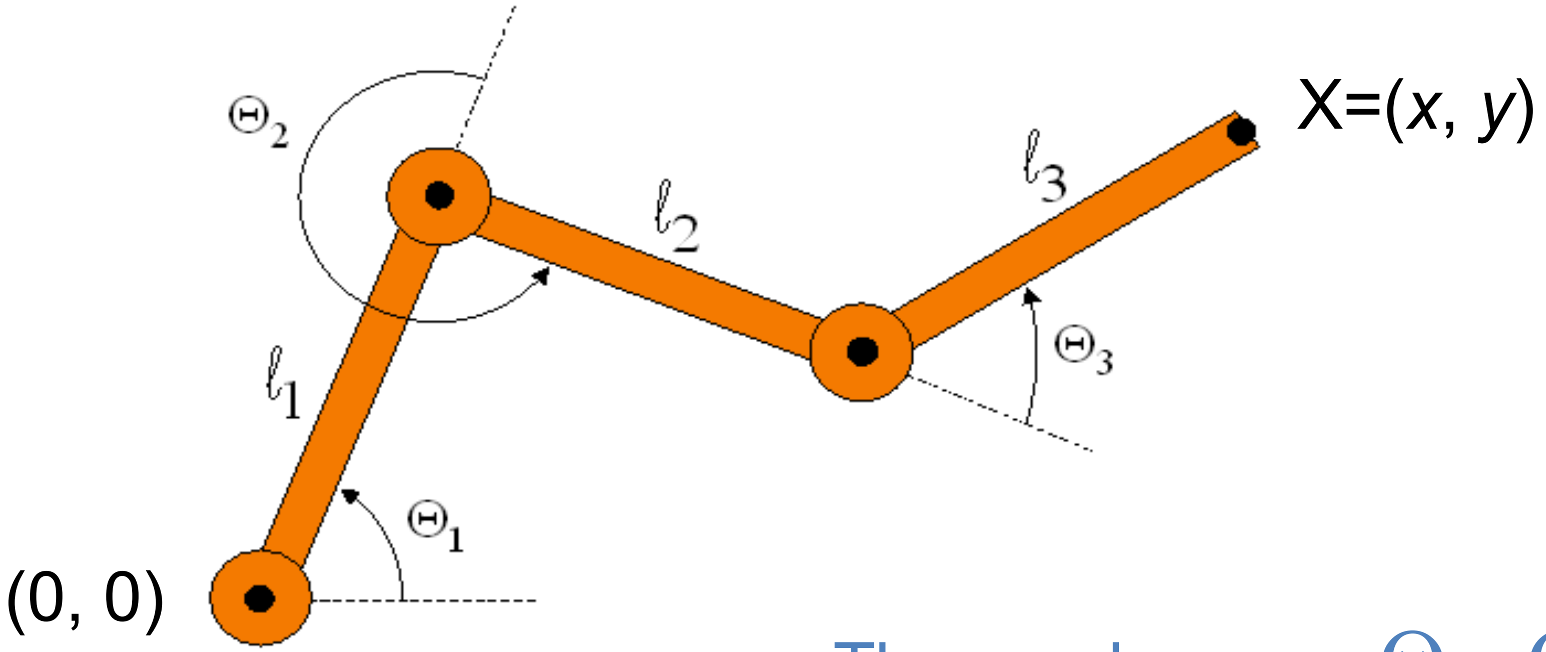
- End-Effector positions can be specified by spline curves





Inverse Kinematics

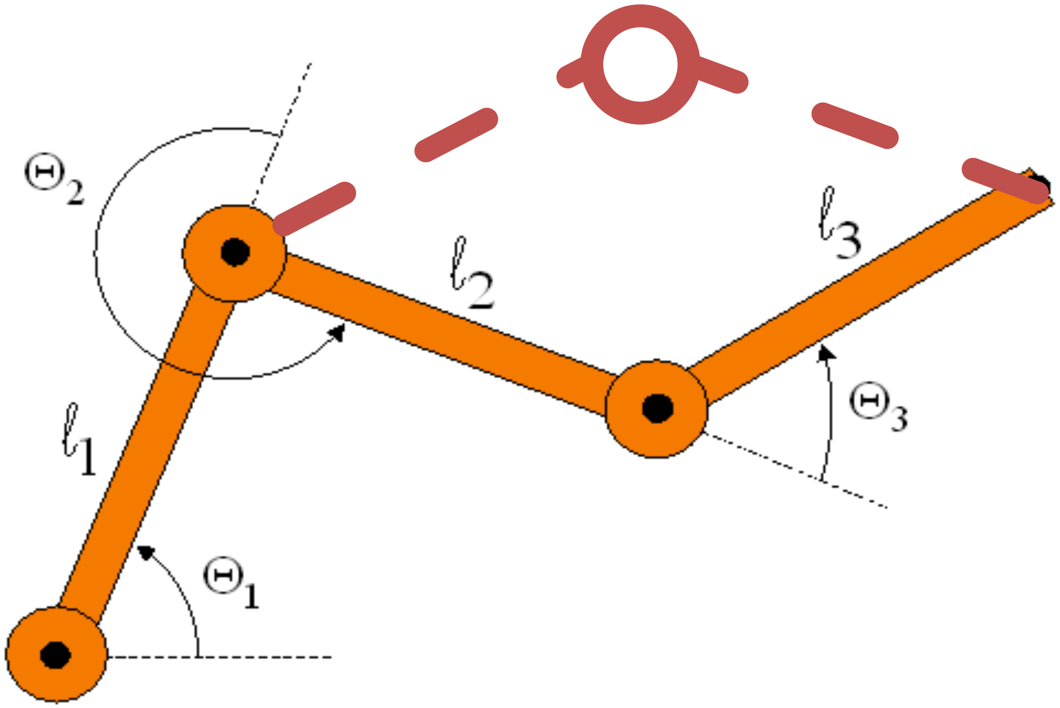
- Problem for More Complex Structures
  - System of equations is usually under-defined
  - Multiple solutions



Three unknowns: $\Theta_1, \Theta_2, \Theta_3$
Two equations: x, y

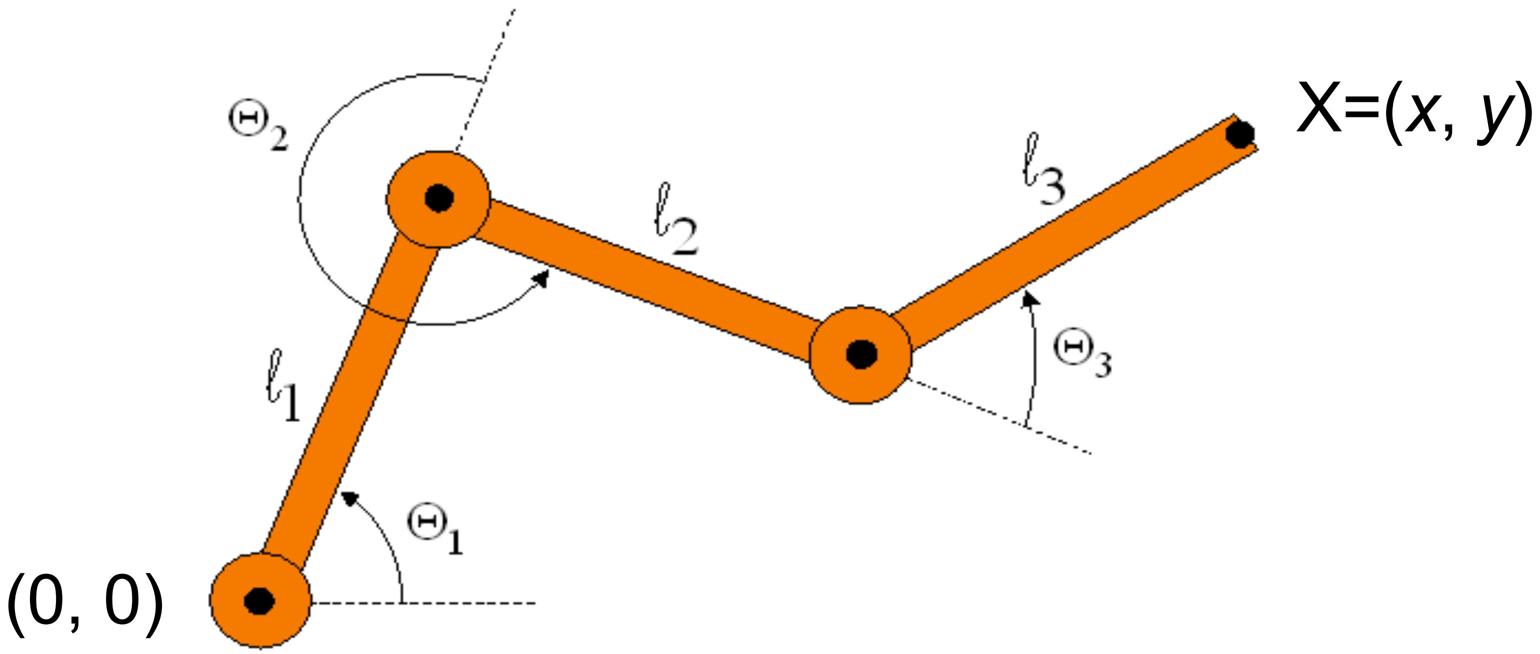
What makes inverse kinematics hard

- Redundancy



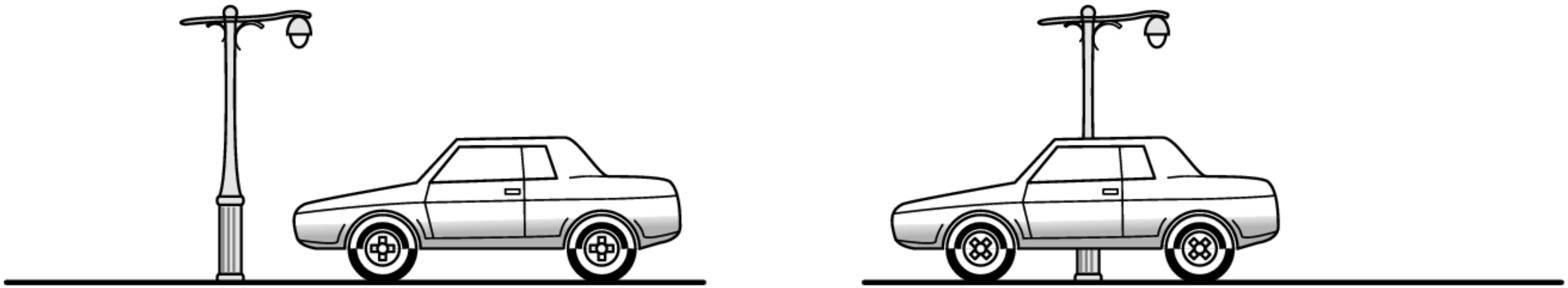
Inverse Kinematics

- Solution for More Complex Structures
  - Find best solution (e.g., minimize energy in motion)
  - Non-linear optimization



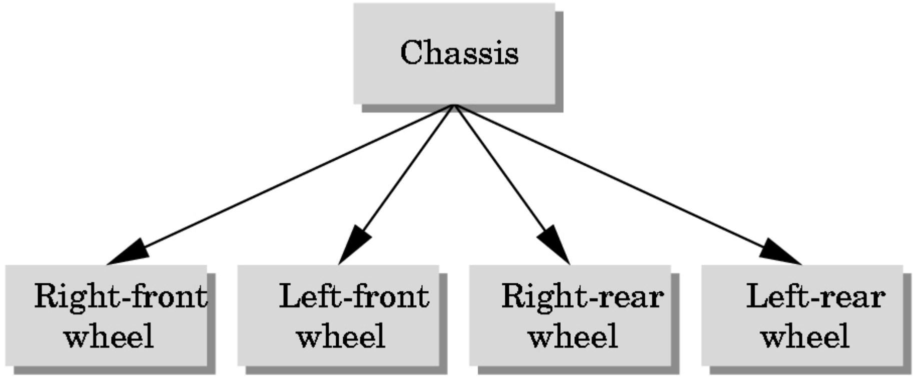
Hierarchical models

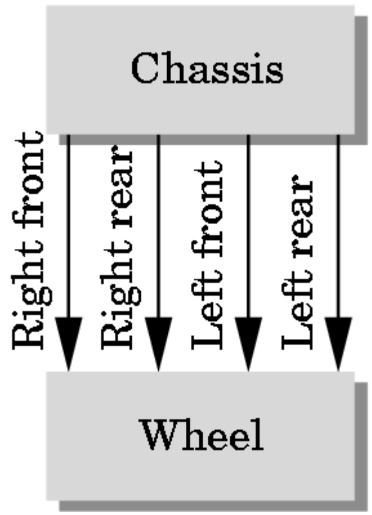
- When animation is desired, objects may have parts that move with respect to each other
  - Object represented as hierarchy
  - Often there are joints with motion constraints
  - Example: represent wheels of car as sub-objects with rotational motion



Directed Acyclic Graph (DAG) models

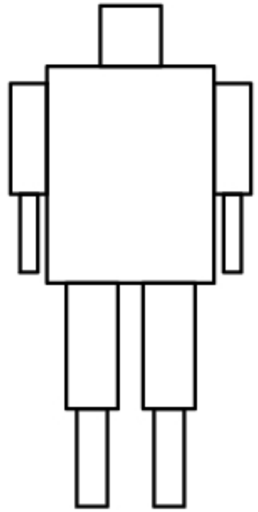
- Could use tree to represent object
- DAG (directed acyclic graph) is better: can re-use objects
- Note that each arrow needs a separate modeling transform
- In object-oriented graphics, also need motion constraints with each arrow

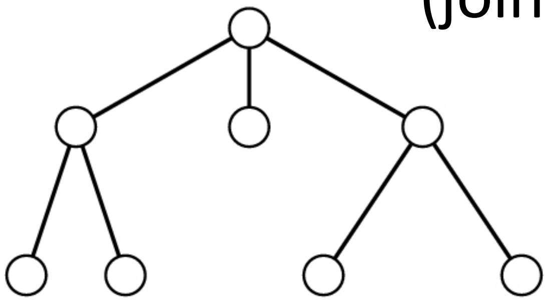


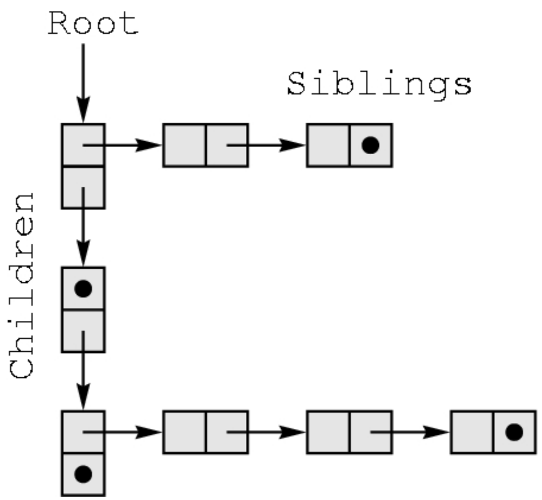


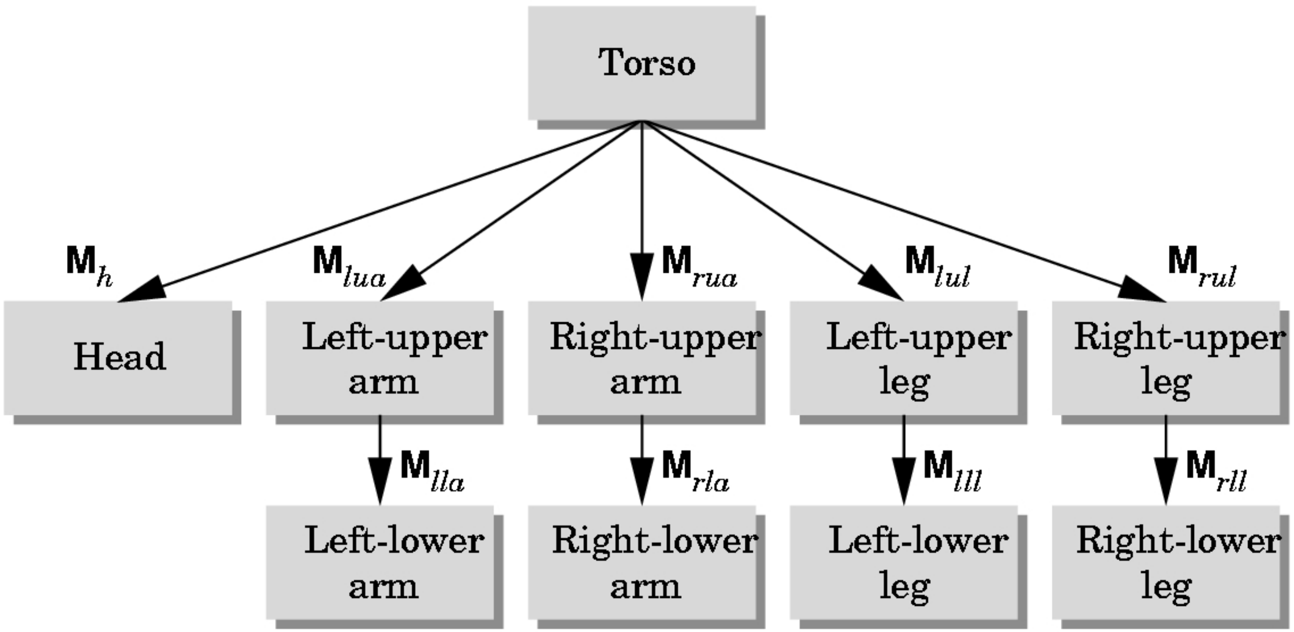
Example: Robot

- Traverse tree (or DAG) using DFS (or BFS)
- Push and pop matrices along the way
  (e.g. left-child right-sibling)
  (joint position parameters?)

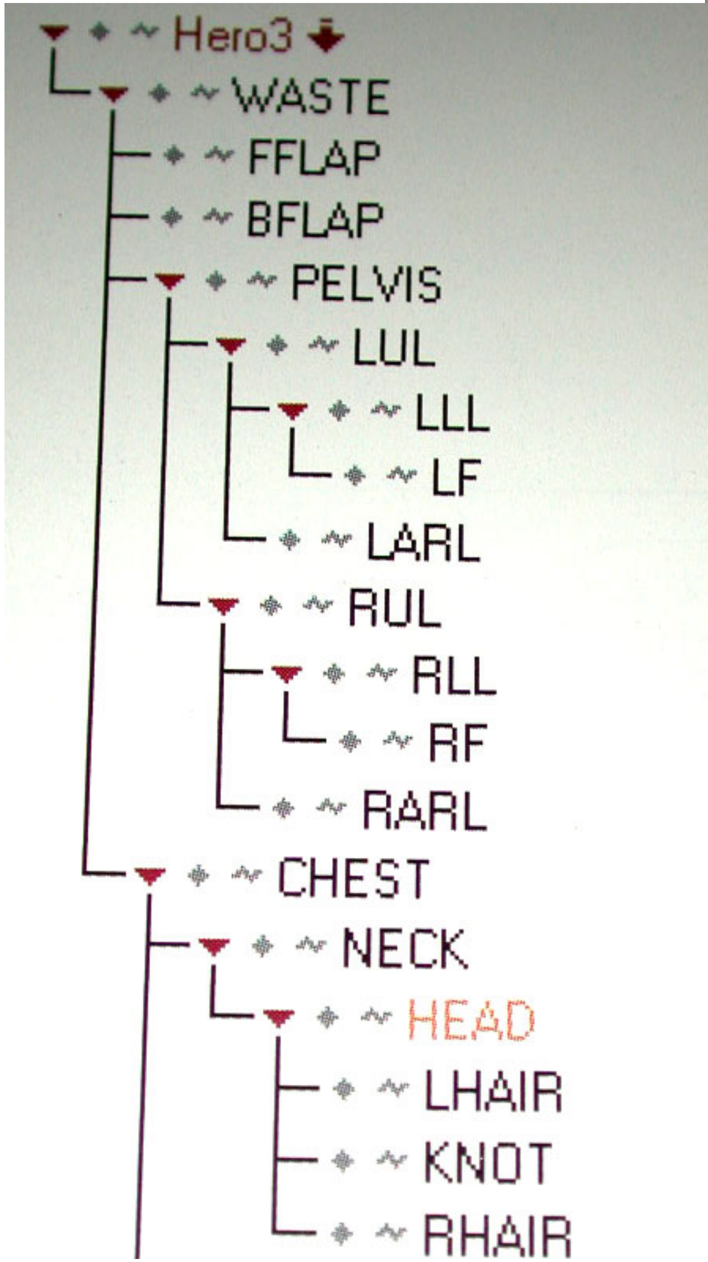






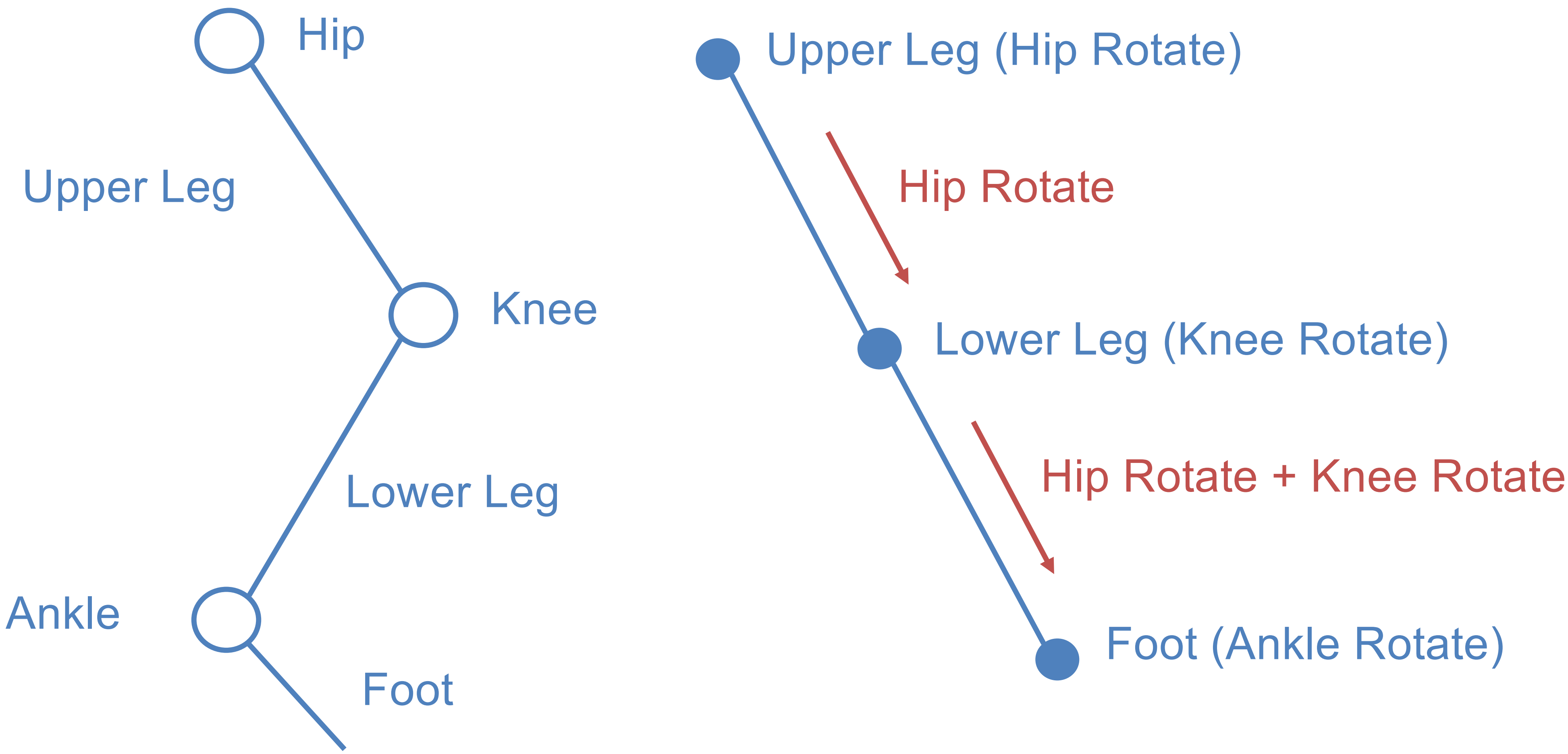


Example: Character



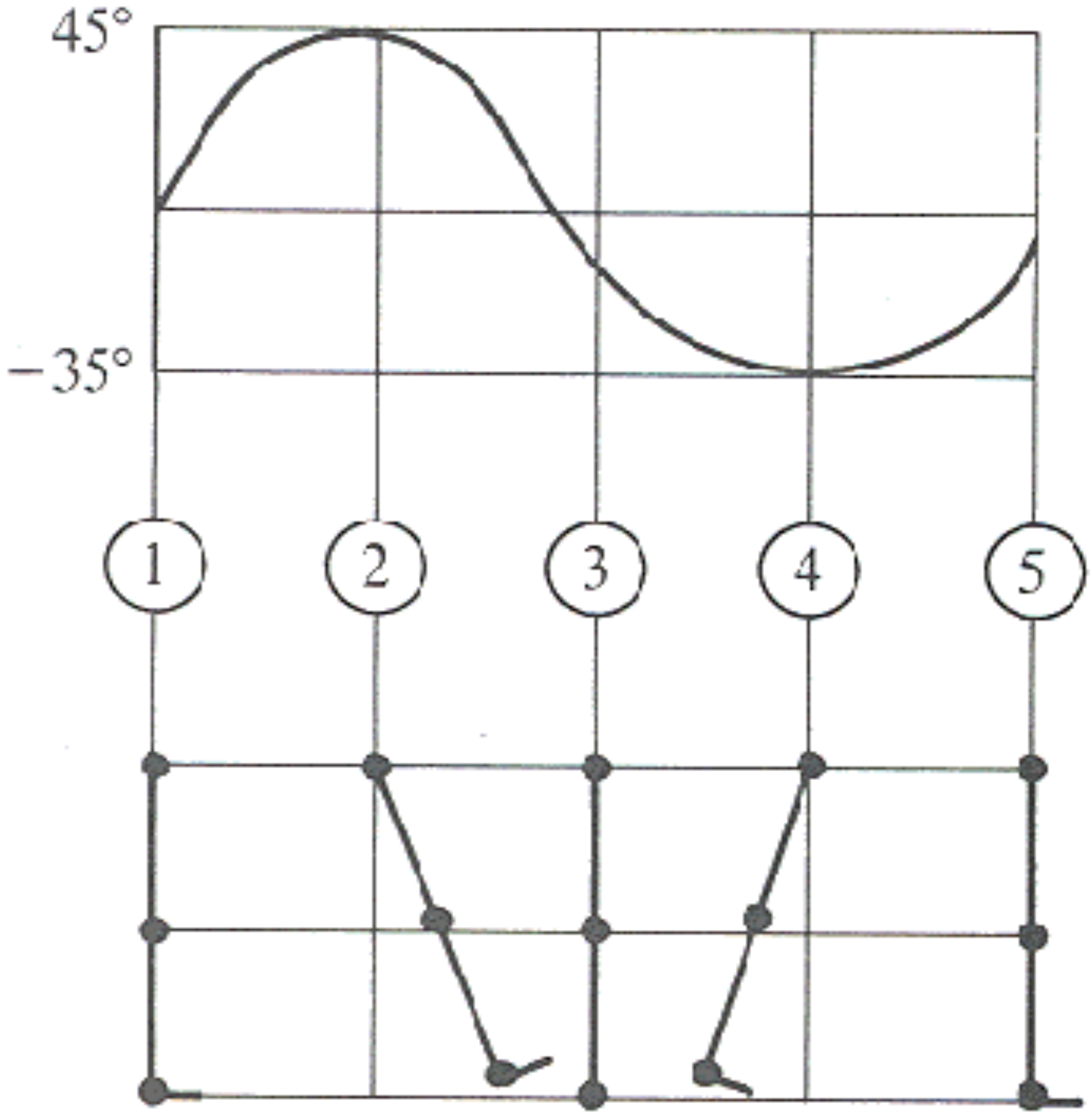
Example: Walk Cycle

- Leg:



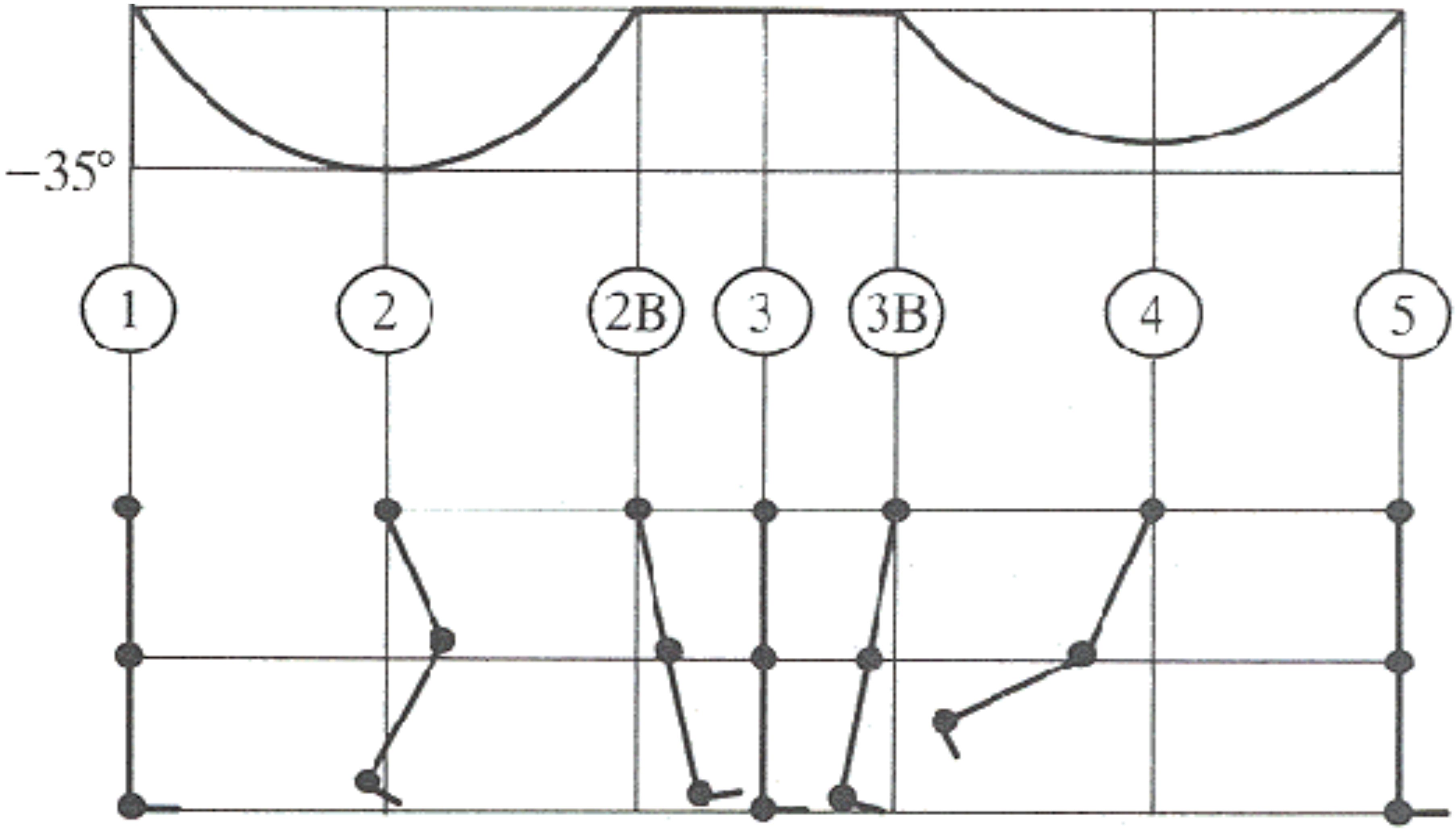
Example: Walk Cycle

- Hip Joint Orientation:



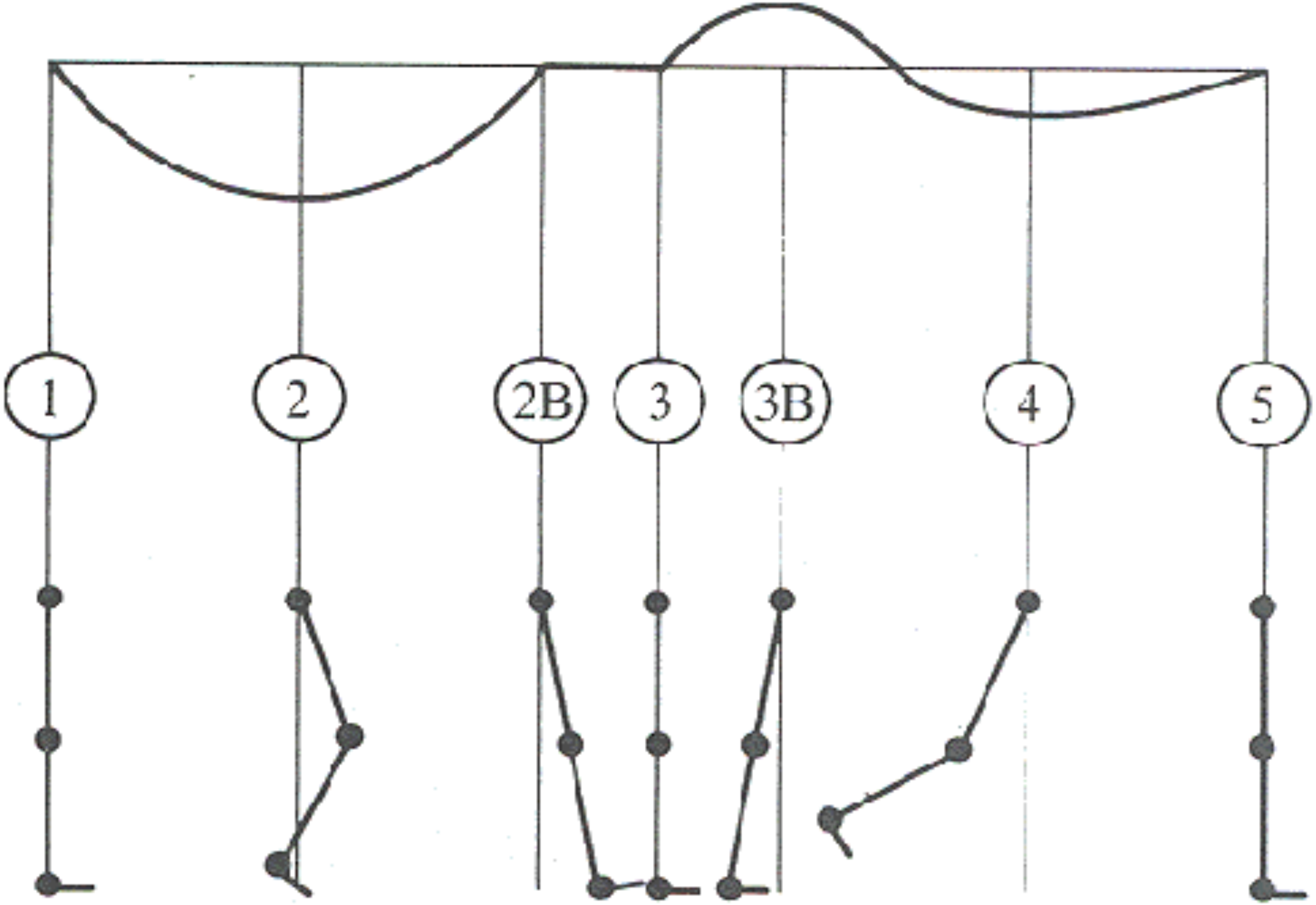
Example: Walk Cycle

- Knee Joint Orientation :



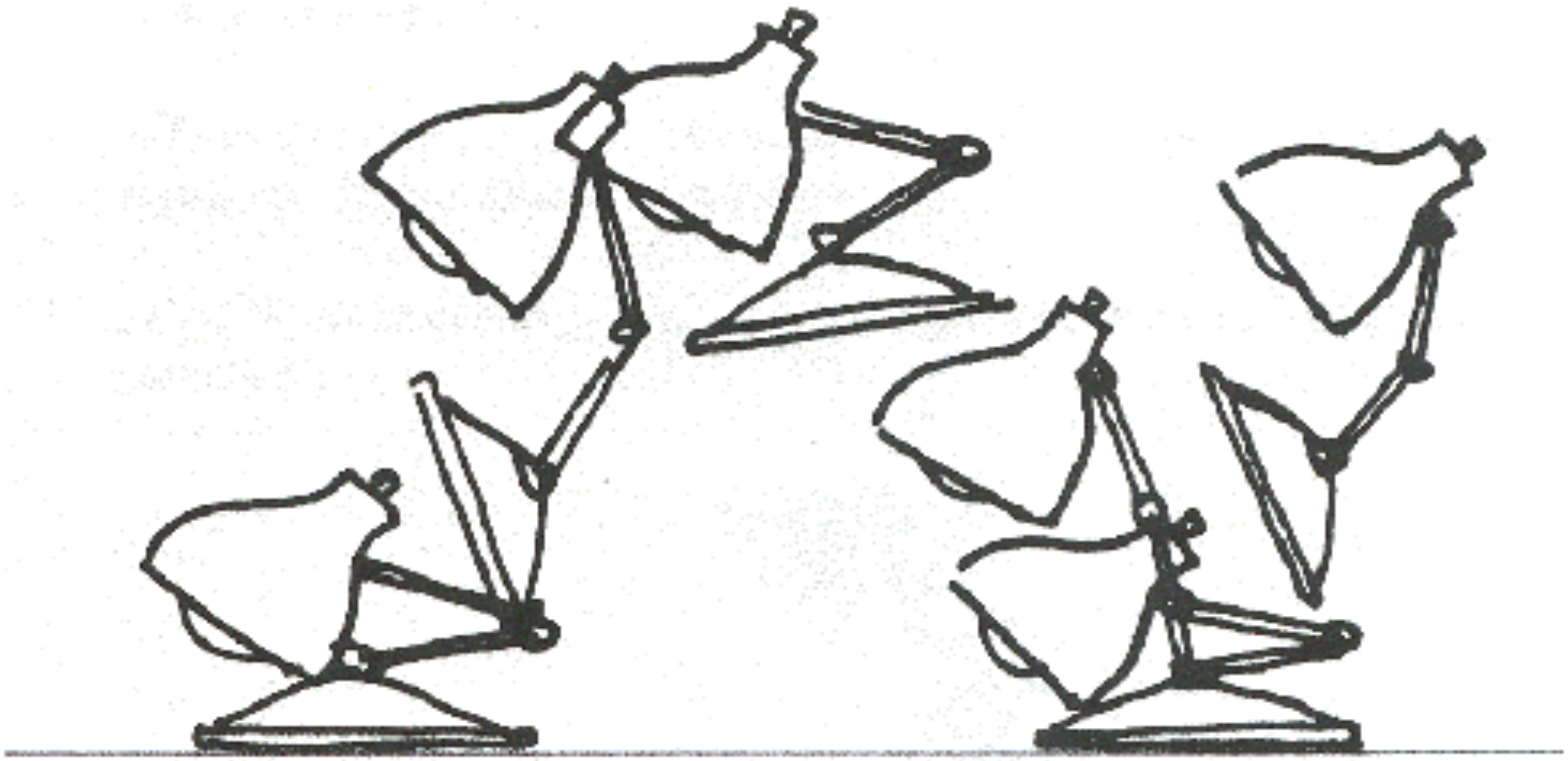
Example: Walk Cycle

- Ankle Joint Orientation:



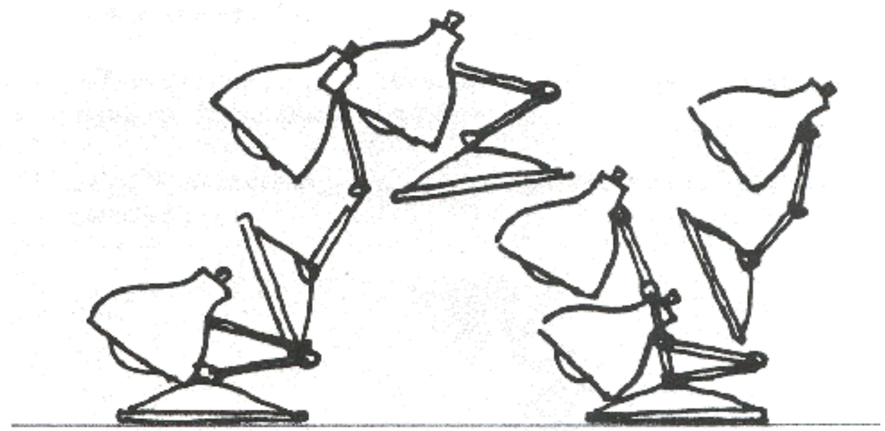
Dynamics

- Simulation of physics insures realism of motion



Space Time Constraints

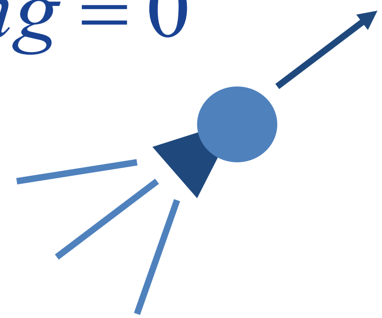
- Animator Specifies Constraints
  - What the character's physical structure is (e.g. articulated figure)
  - What the character has to do (e.g., jump from here to there within time *t)*
  - What other physical structures are present (e.g. floor to push off and land)
  - How the motion should be performed (e.g. minimize energy).



Space Time Constraints

- Compute the optimal physical motion satisfying constraints

- Example: particle with jet propulsion
  - $x(t)$ is position of particle at time t
  - $f(t)$ is force of jet propulsion at time t
  - Particle's equation of motion is: $mx'' - f - mg = 0$



  - Suppose we want to move from a to b within t_0 to t_1 with minimum jet fuel:

$$\text{Minimize} \quad \int_{t_0}^{t_1} |f(t)|^2 \, dt \text{ subject to } x(t_0) = a \text{ and } x(t_1) = b$$

Space Time Constraints

- Discretize Time Steps

$$x' = \frac{x_i - x_{i-1}}{h}$$

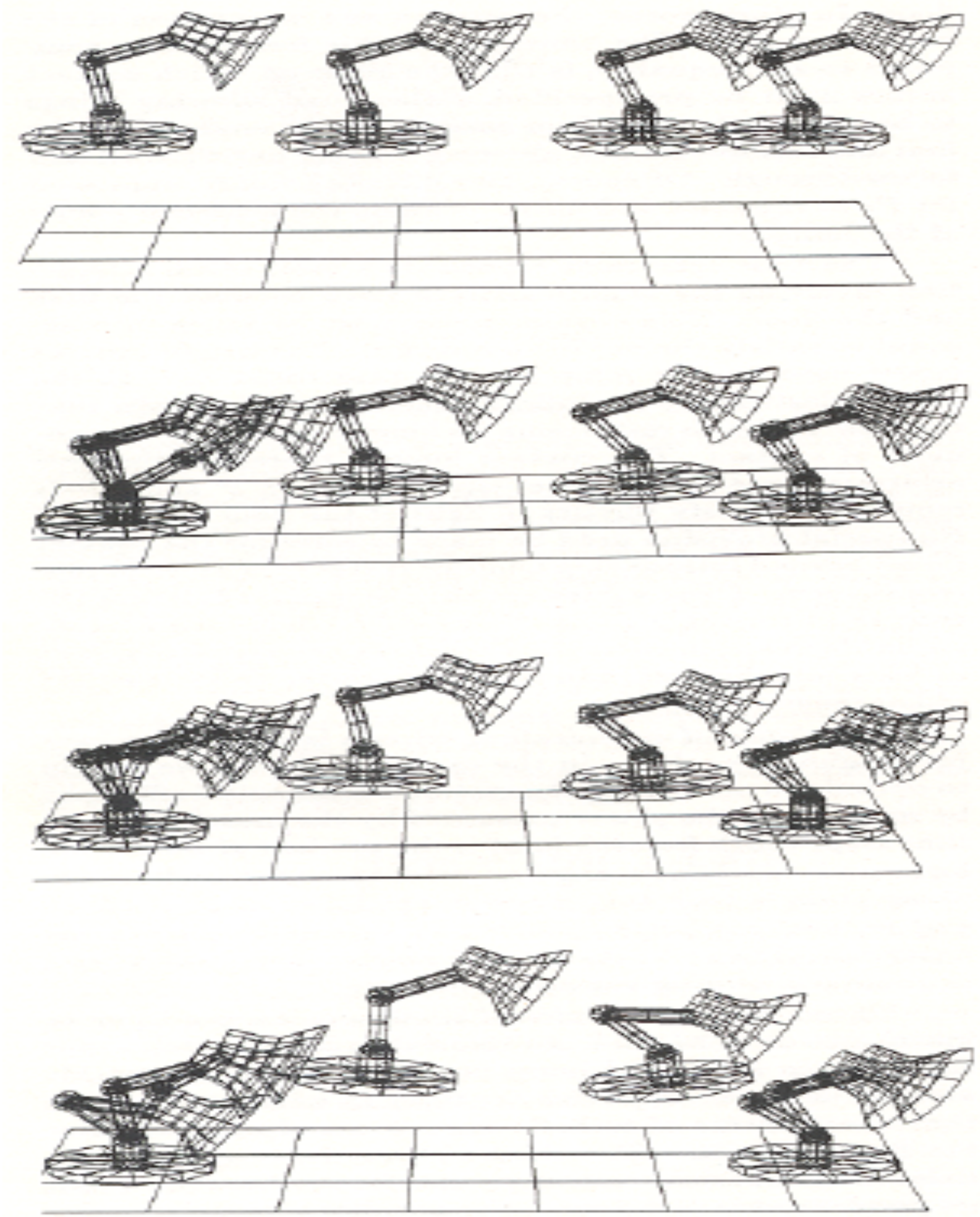
$$x'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}$$

$$m\left(x'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} \right) - f_i - mg = 0$$

$$\text{Minimize} \quad h\sum_i |f_i|^2 \text{ subject to } x_0 = a \text{ and } x_1 = b$$

Space Time Constraints

- Solve with iterative optimization methods



Space Time Constraints

- Advantages:
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints

- Challenges:
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization