

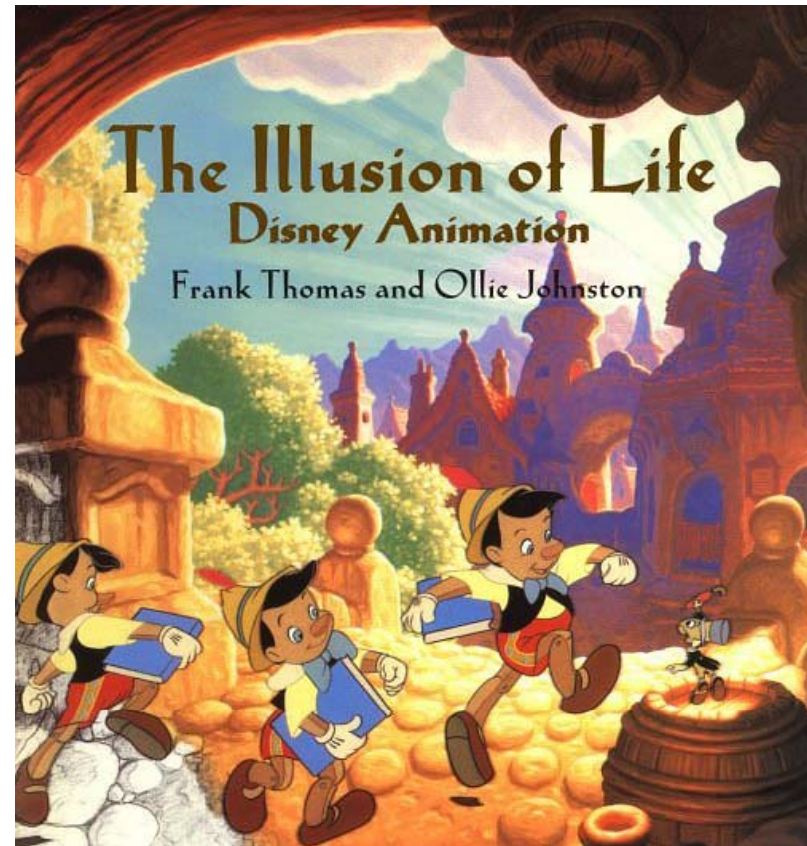
Keyframe animation

- Keyframing: Classical
- Interpolation
- Kinematics
- Inverse Kinematics

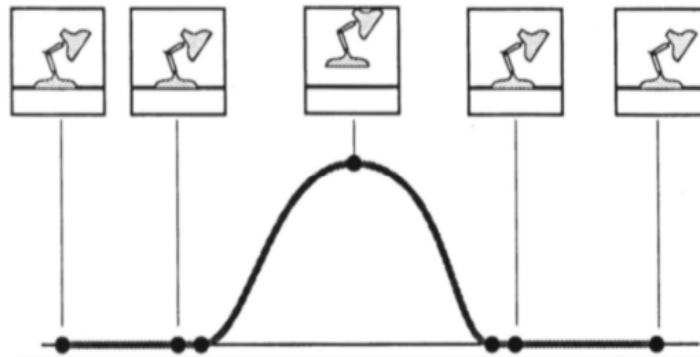
•*Some slides borrowed from Ronen Barzel*

Keyframe animation: classical

- Animator draws character at “extreme” poses,
- Apprentice animators fill the “in-between” frames.



Keyframe animation: CG



ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

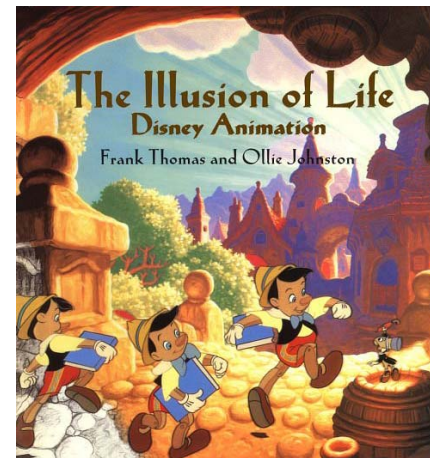
Describe motion of objects as a function of time from a set of key object positions. In short, compute the inbetween frames.

Keyframe animation: CG

- Model parameters
 - Position (x,y,z), joint angles, deformations, ...
- Key frames = data points
- In-between = interpolation
- Many ways to interpolate:
 - *linear*
 - *splines* (polynomial curves)

Animation Principles

- From
 - “Principles of Traditional Animation Applied to 3D Computer Animation”
John Lasseter, ACM Computer Graphics, 21(4), 1987
- In turn from
 - “The Illusion of Life”

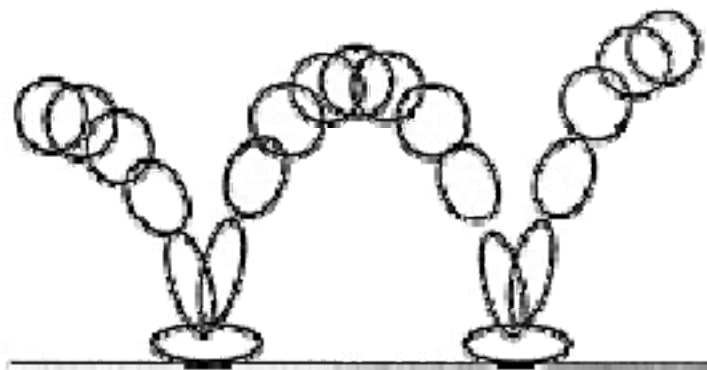


Squash and stretch

Squash: flatten an object or character by pressure or by its own power

Stretch: used to increase the sense of speed and emphasize the squash by contrast

Determines feeling of weight, flexibility, response to pressure. Maintains volume



Timing

Timing affects weight and emotion:

- Light/happy object move quickly
- Heavier/sad objects move slower

Timing completely changes the interpretation of the motion. Because the timing is critical, the animators used the draw a time scale next to the keyframe to indicate how to generate the in-between frames.

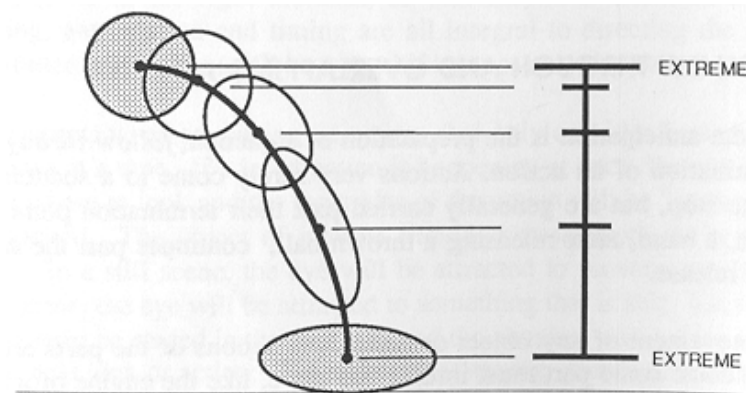


FIGURE 9. Timing chart for ball bounce.

Anticipation

- An action breaks down into:
 - Anticipation
 - Action
 - Reaction
- Anatomical motivation: a muscle must extend before it can contract.
- Prepares audience for action so they know what to expect.
- Directs audience's attention.
- Amount of anticipation can affect perception of speed and weight.

Follow Through and Overlap

- Follow through is the termination part of an action.
 - E.g. throwing a ball - the hand continues to move after the ball is released.
- Heavier parts lag farther and stop slower.
 - E.g. the antennae of an insect - they will lag behind and then move quickly to indicate the lighter mass.
- Overlapping means to start a second action before the first action has completely finished.
 - This keeps the interest of the viewer, since there is no dead time between actions.
- *"When a character knows what he is going to do he doesn't have to stop before each individual action and think to do it. He has it planned in advance in his mind."* Walt Disney

Staging

- Presentation of an idea so that it is clear
 - can be an action, a personality, an expression, or a mood.
- Lead the viewer's eye to where the action will occur so that they do not miss anything
 - the main object or action should be contrasted in some way with the rest of the scene.
- Ensure that the viewer is looking at the correct object at the correct time.

Action

- "Straight ahead action" means drawing out a scene frame by frame from beginning to end
 - creates a more fluid, dynamic illusion of movement, and is better for producing realistic action sequences
 - hard to maintain proportions, and to create exact, convincing poses along the way
- "pose to pose" involves starting with drawing a few, key frames, and then filling in the intervals later
 - works better for dramatic or emotional scenes, where composition and relation to the surroundings are of greater importance
- A combination of the two techniques is often used
- Computer animation removes the problems of proportion related to "straight ahead action" drawing; however, "pose to pose" is still used for computer animation, because of the advantages it brings in composition

Ease-in and Ease-Out

- Movement doesn't start & stop abruptly.
- Also contributes to weight and emotion
- The movement of the human body, and most other objects, needs time to accelerate and slow down.
 - For this reason, an animation looks more realistic if it has more frames near the beginning and end of a movement, and fewer in the middle.
 - For characters moving between two extreme poses such as sitting down and standing up, or for inanimate, moving objects, like a bouncing ball.

Arcs

- Move in curves, not in straight lines
- Most human and animal actions occur along an arched trajectory
 - animation should reproduce these movements for greater realism.
- This can apply to a limb moving by rotating a joint, or a thrown object moving along a parabolic trajectory.
- The exception is mechanical movement, which typically moves in straight lines.

Exaggeration

- Exaggeration is especially useful for animation
 - perfect imitation of reality can look static and dull in cartoons
- The level of exaggeration depends on whether one seeks realism or a particular style, like a caricature or the style of an artist
- The classical definition of exaggeration, employed by Disney, was to remain true to reality, just presenting it in a wilder, more extreme form.
- There should be a balance in how those elements are exaggerated in relation to each other, to avoid confusing or overawing the viewer

Secondary action

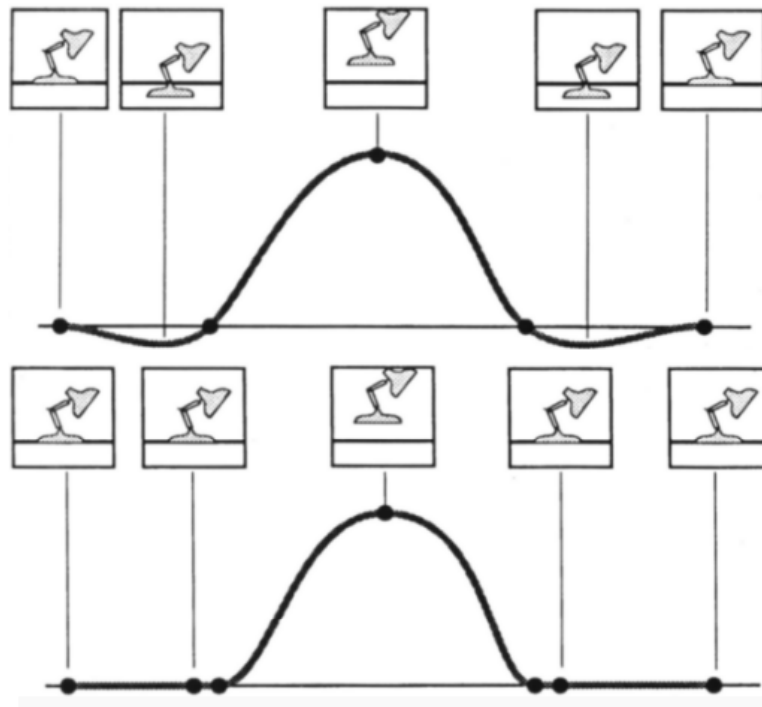
- Motion that results from some other action
- Needed for interest and realism
- Shouldn't distract from primary motion
- (today: often use simulation to compute secondary motion of hair, clothing, etc.)

Appeal

- Appeal in a cartoon character corresponds to what would be called charisma in an actor
- A character who is appealing is not necessarily sympathetic — villains or monsters can also be appealing — the important thing is that the viewer feels the character is real and interesting

Interpolating Key Frames

Interpolation is not fool proof. The splines may undershoot and cause interpenetration. The animator must also keep an eye out for these types of side-effects.



Linear interpolation

- Given points P_0 and P_1 , define curve $L(t)$:

$$L(t) = (1-t) P_0 + t P_1 \quad t \text{ in } [0, 1]$$

- Weighted average of endpoints.
- “Curve” is linear segment

Linear interpolation: N points

- Given $P_0 \dots P_N$, define segment:

$$L_i(s) = (1-s) P_i + s P_{i+1} \quad s \text{ in } [0, 1]$$

- Then define piecewise-linear curve:

$$L(t) = L_i(s) \quad \text{for } t_i < t < t_{i+1}$$

Where $s = (t-t_i)/(t_{i+1}-t_i)$

and (let's say) $t_i = i/N$

Interpolation along a curve

- The parametric representation of curves are often used to depict non-linear trajectories
 - In general for any genus of surface or curve, there is both a parametric and an implicit representation.
 - In computer graphics it is often more convenient to adopt the parametric form.

Example – definition of sphere

- For example, the implicit form of a sphere is:

$$x^2 + y^2 + z^2 - r^2 = 0$$

- In general all implicit representations of a 3-d surface are of the form:

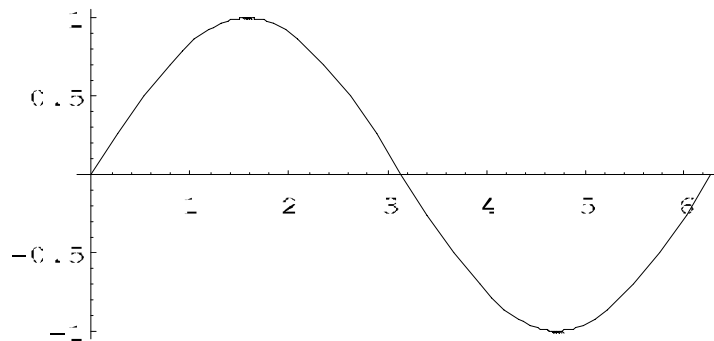
$$f(x, y, z) = 0$$

- The parametric form of a sphere is:

$$f: (\theta, \phi) \mapsto (\cos \theta \cos \phi, \sin \theta, \cos \theta \sin \phi)$$

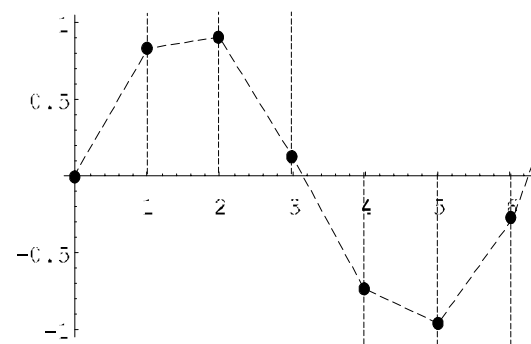
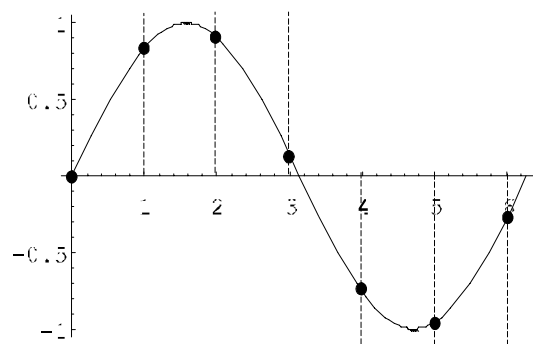
Example - 1

- Determine the representation of the function $f(\theta) = \sin(\theta)$.
- This is a parametric description of a curve in 2 dimensions with parameter θ .
- This is an example of an unbounded curve (in that we can take values of θ from $-\infty \dots +\infty$). We'll limit our curve to the domain $(0 \dots 2\pi)$. This gives the following curve:



Example - 2

- Now determine how fine or coarse a representation is needed to faithfully capture this curve.
- Sample the curve at regular intervals of θ along the length of the curve.
 - In this example, the curve is sampled at regular points 1 unit distance apart (i.e. at $\theta = 0, 1, 2, \dots$).
- Sampled points are joined by straight lines
 - This is how the curve will be used for animation or display

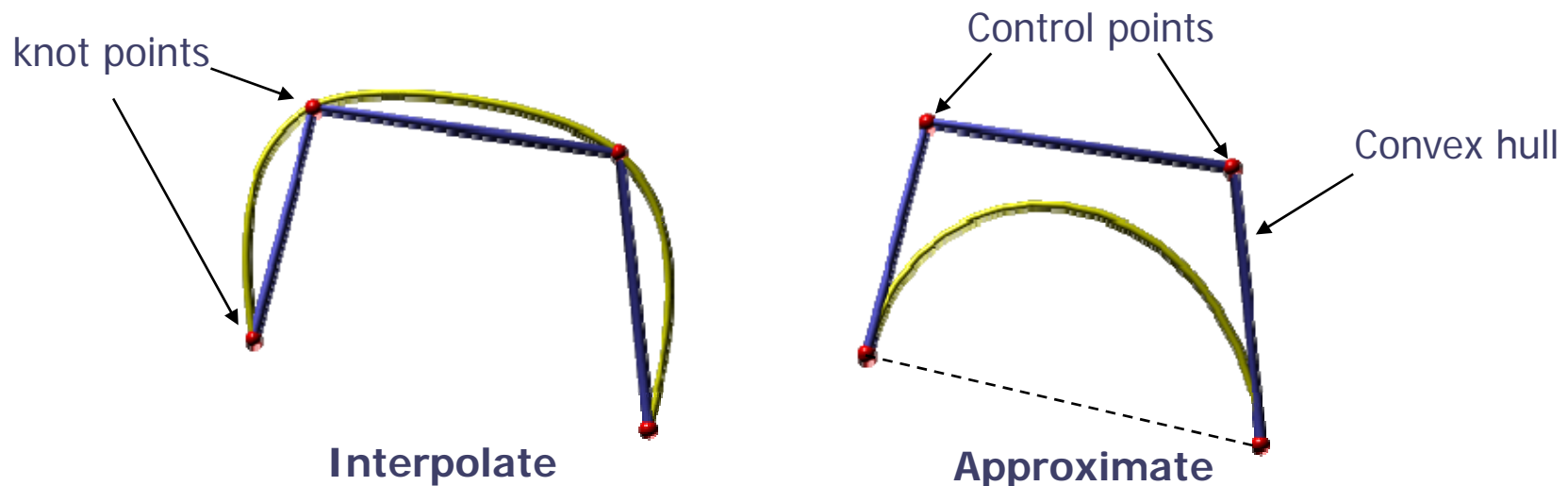


Interpolating along a curve

- There are many different methods of representing general curves (rather than attempt to model all surfaces as some existing function, say a Sine or Cosine).
- The most common are:
 - **Cubic Splines**
 - **Bezier Curves**
 - **B-splines**
 - **NURBS and β -splines**

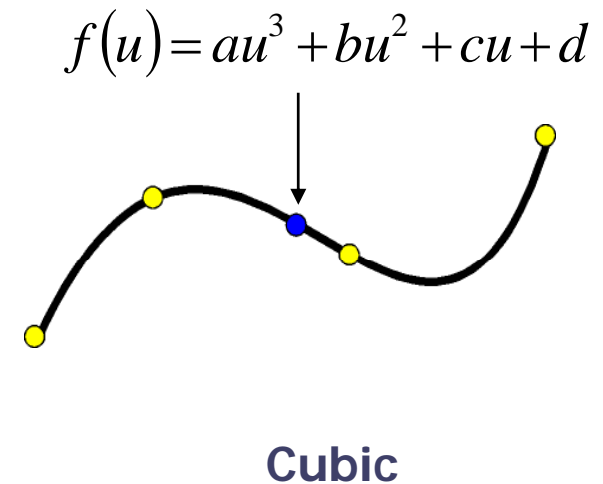
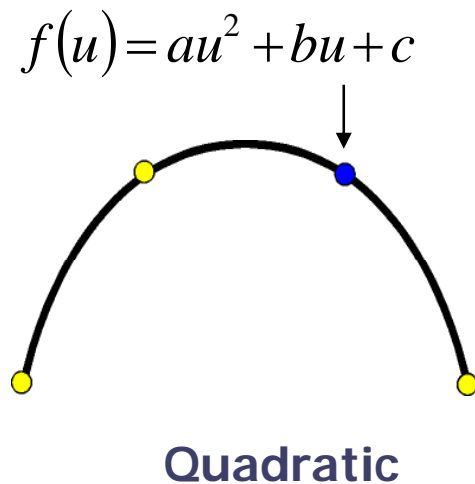
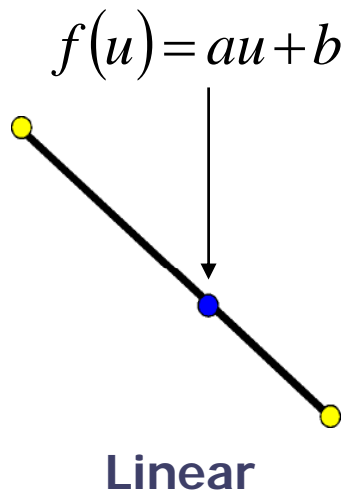
Interpolation vs. Approximation

- Given a set of n points, to create a curve we either
 - **interpolate** the points (curve passes through all points)
 - **approximate** the points (points describe *convex hull* of curve)
- Points on curve = *knot points*
- Points on convex hull (off curve) = *control points*



Splines

- To interpolate we can use a simple polynomial spline. With n points we require a polynomial of degree $n-1$ (order n polynomial).
- Let $f(u)$ be the parameterised polynomial where $0 \leq u \leq 1$



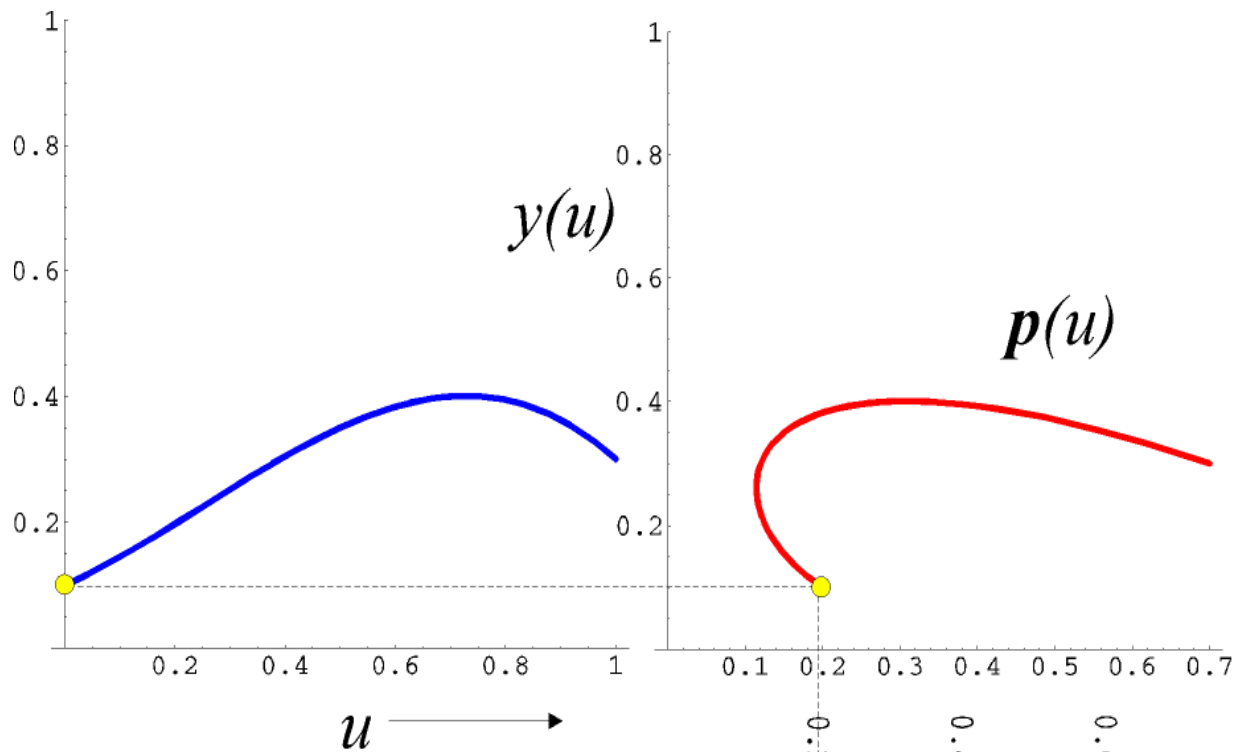
Splines

- These polynomials are plots of $f(u)$ with respect to u
 - for each u , there is one and only one $t(u)$
⇒ curve cannot turn back on itself
- Use polynomials for each axis (in 2D we have 2 polys):

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

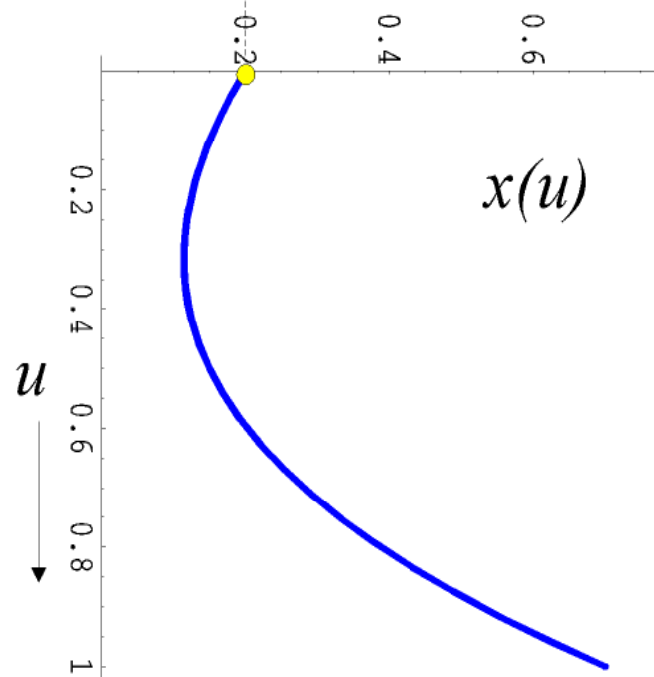
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$

- As before we limit u to $[0,1]$, although the polynomial is defined for all values of u .



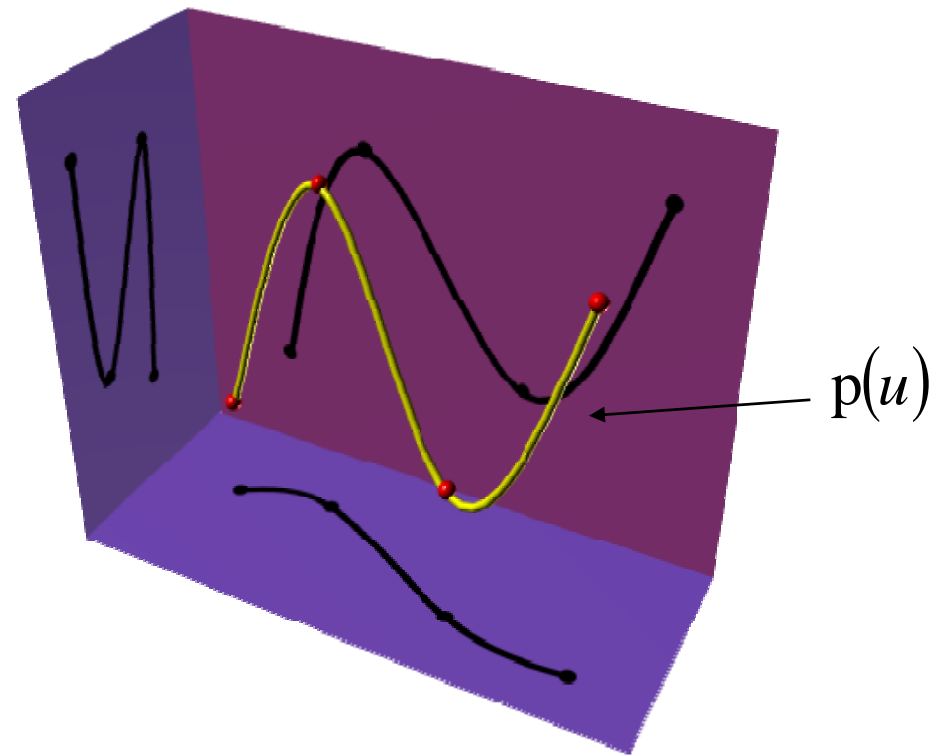
$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$



Splines

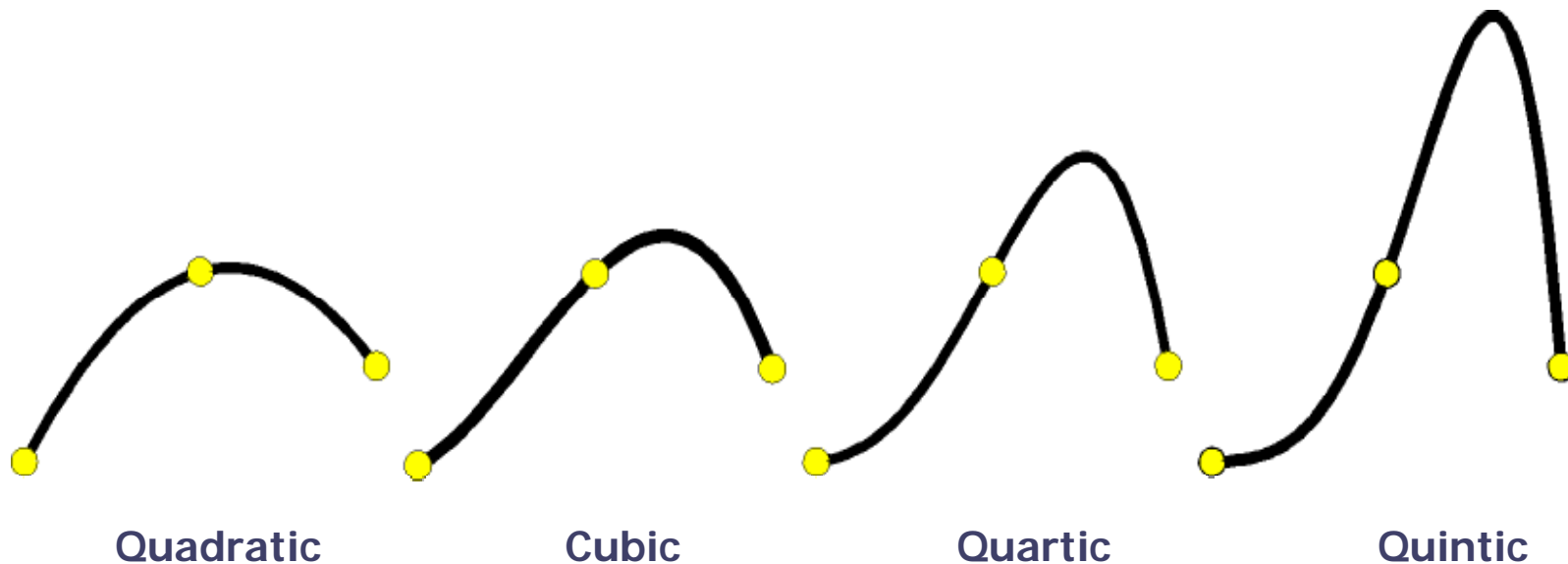
- For a 3D spline, we have 3 polynomials:



The point $p(u)$ is a weighted sum of 4 points or vectors in 3-space

Splines

- If we have more than 4 points we require a polynomial of higher degree
 - higher degree polynomials are more difficult to control
 - they exhibit unwanted *wiggles* (oscillations)

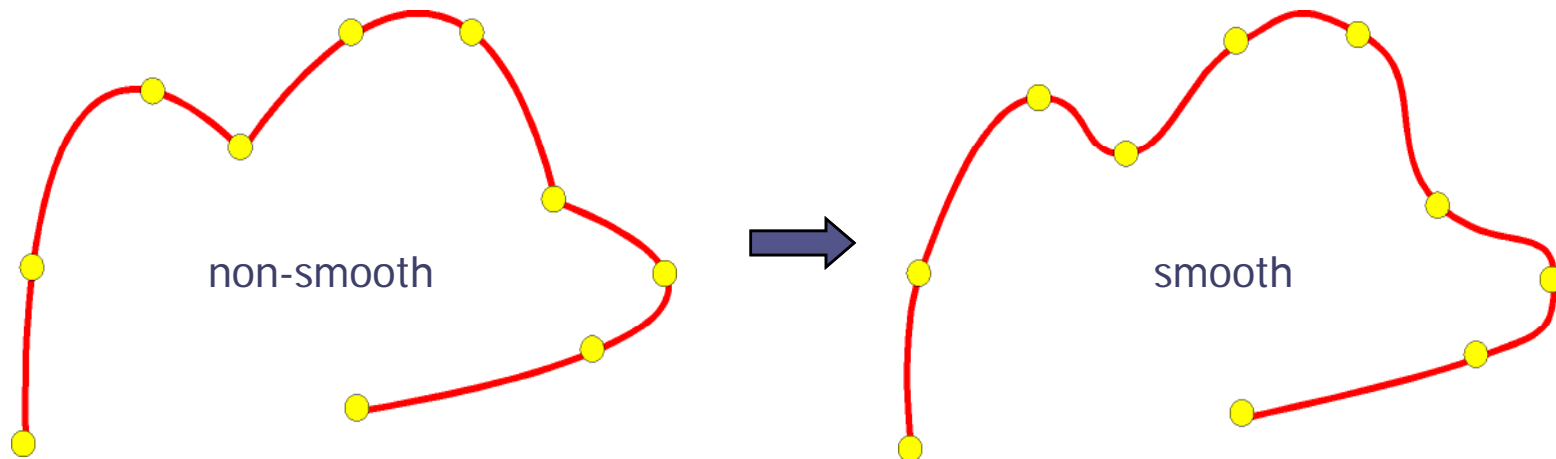


Splines

- In general we use cubic polynomials for curves in CG:
 - minimal wiggles and faster to compute than high degree polynomials
 - lowest degree which allows *non-planar curves* (quadratics require 3 points, 3 points always lie in the same plane)

Defining the Cubic Spline

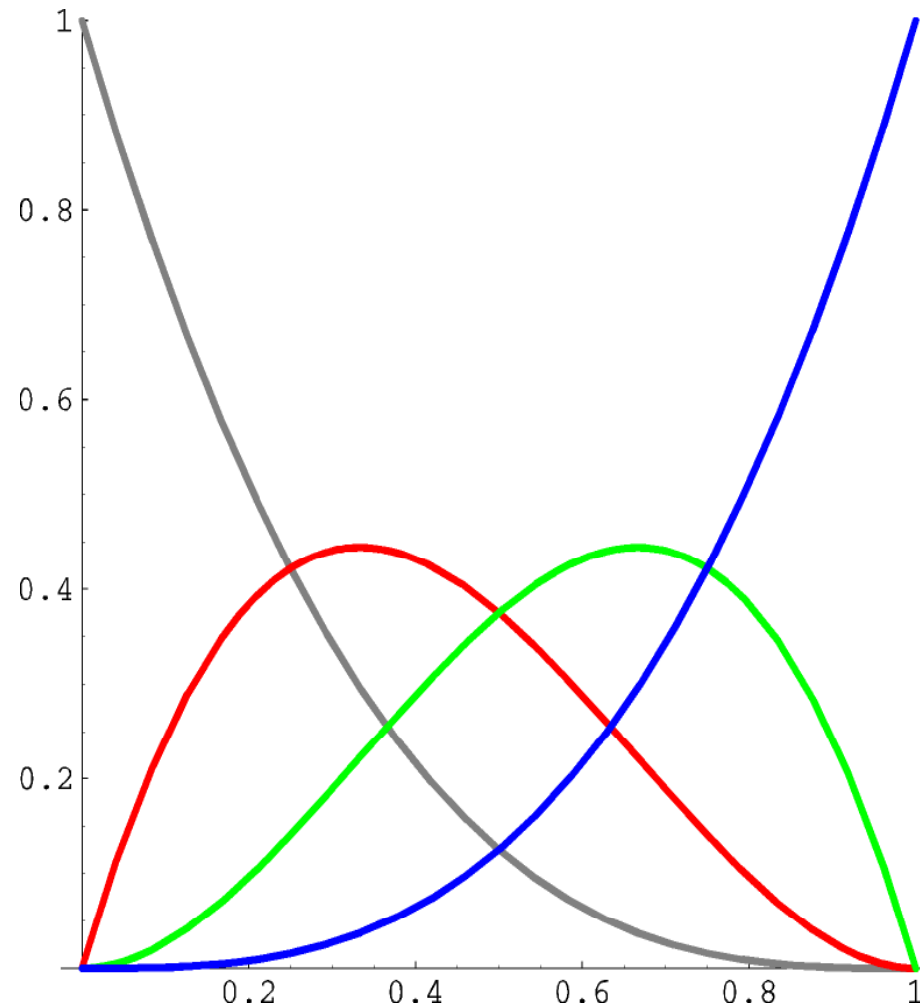
- Normally we supply 4 points we wish the spline to pass through.
- If we have more than 4 points we must employ more than 1 spline \Rightarrow use a *piecewise cubic polynomial*
 - for n points, we have $(n-1)/3$ individual cubic segments
 - without further constraints these will not join smoothly



Blending Functions

- The curve is a weighted sum of the four points or vectors
- The weights are each cubic polynomials of the parameter u , and are called the blending functions.
- This is similar to linear interpolation, for which only two geometric constraints (i.e. the endpoints of the line) are needed.
- Parametric cubics are really just a generalization of straight-line approximations:
 - The cubic curve $\mathbf{p}(u)$ is a combination of the **4** points or vectors, whereas the line is an affine combination of **2** points.

Sample Blending Polynomials



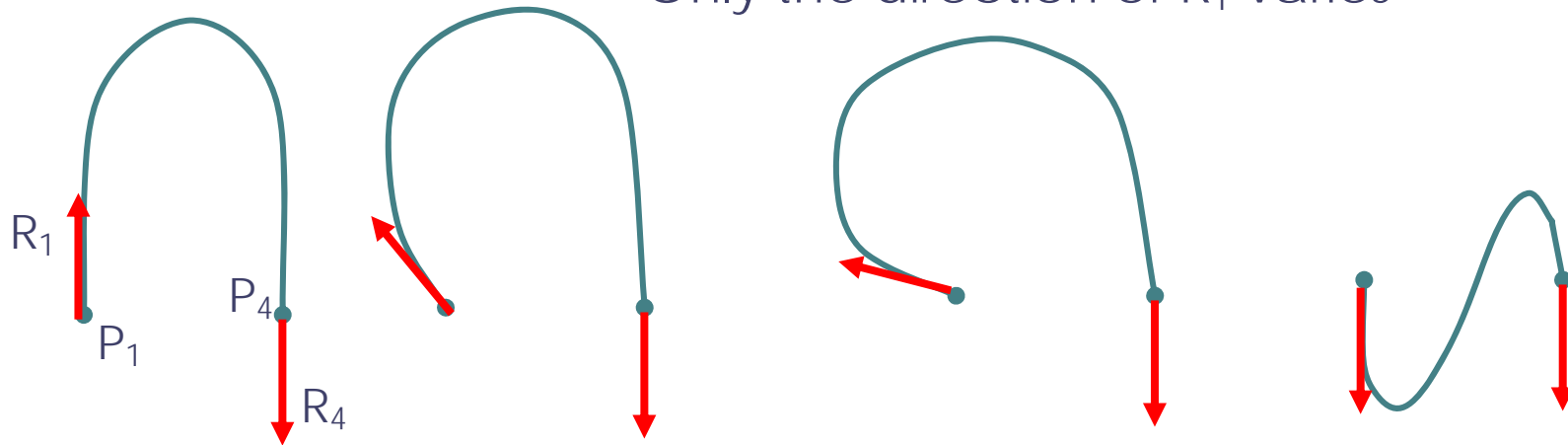
The Bernstein polynomials, weighting functions for Bézier curves

Hermite Curves

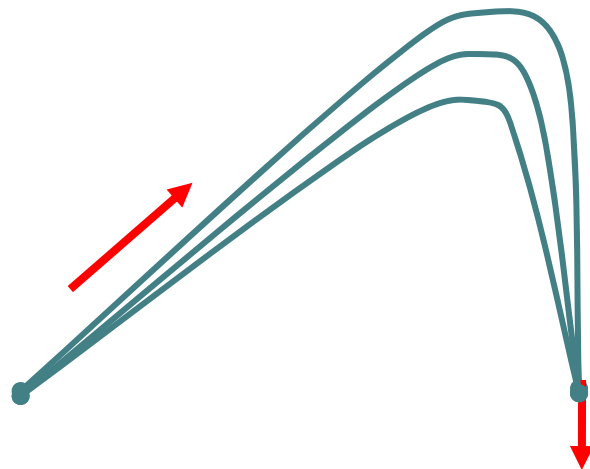
- The key to defining a parametric cubic curve therefore lies in these blending functions.
- Depending on the nature of these functions, specific forms of curves may be created.
- The Hermite form of a cubic polynomial curve segment is determined by constraints on the endpoints P_1 and P_4 , and tangent vectors at the endpoints R_1 and R_4 .

Hermite Curves - Examples

Only the direction of R_1 varies



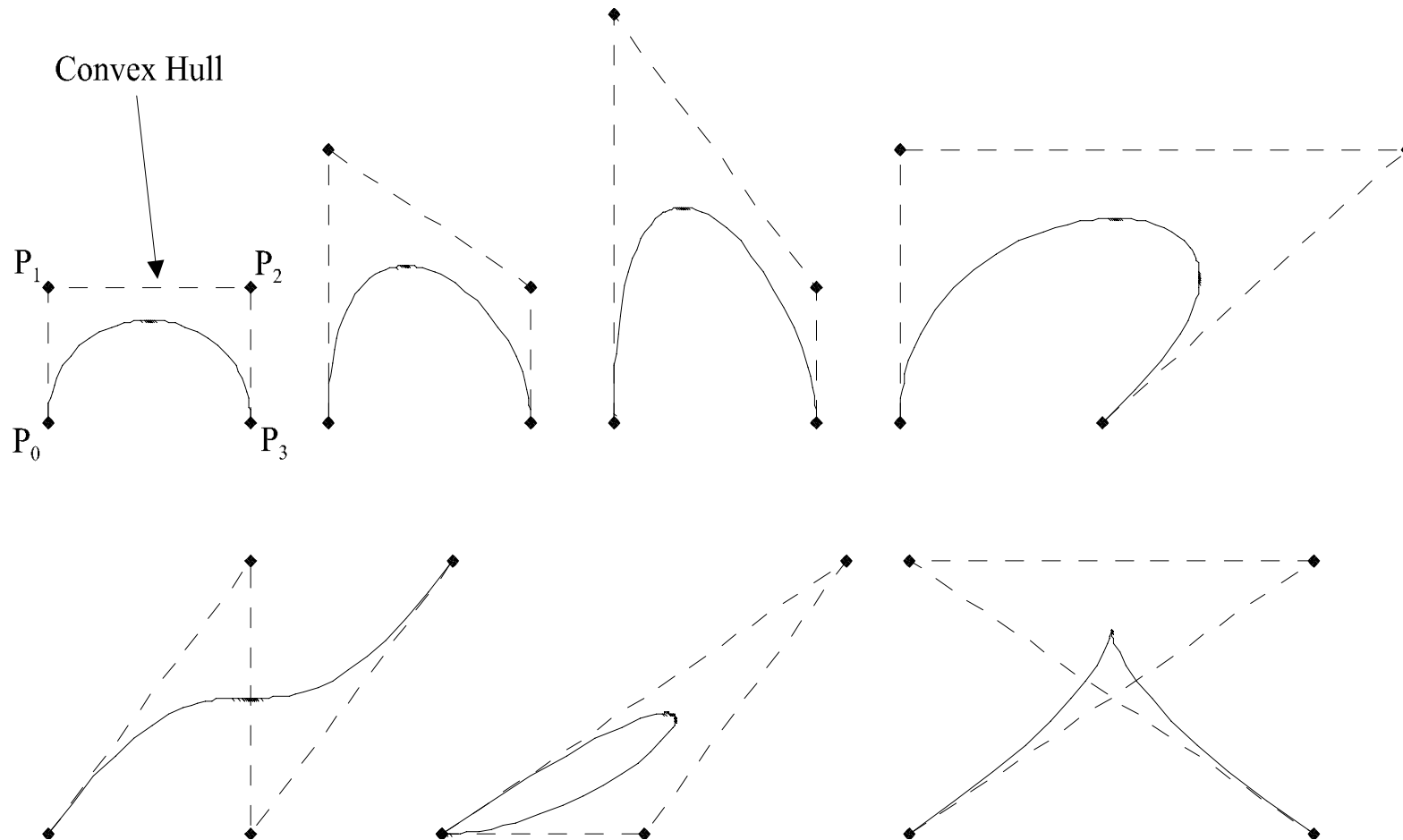
Only the magnitude of R_1 varies



Hermite Blending Functions

$$\begin{aligned} p(t) = & (2t^3 - 3t^2 + 1)P_1 + \\ & (-2t^3 + 3t^2)P_4 + \\ & (t^3 - 2t^2 + t)R_1 + \\ & (t^3 - t^2)R_4 \end{aligned}$$

Examples of Bézier Curves



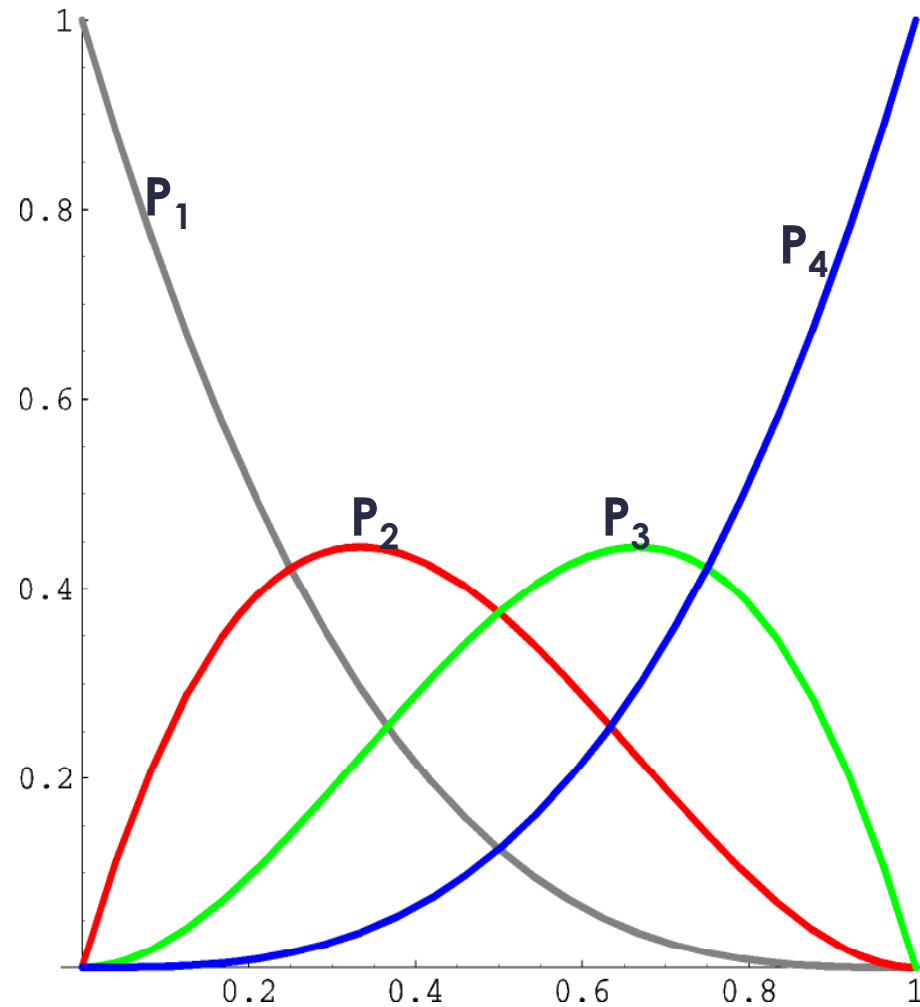
Bernstein Polynomials

- A Bezier spline is defined as:

$$p(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$$

- The four weights are known as the Bernstein Polynomials

Bernstein Polynomials



Labels show which geometry element is weighted.

General Bernstein Form for Bézier Curves

- The Bezier curve $\mathbf{p}(t)$ based on the $(L+1)$ points P_0, P_1, \dots, P_L is given by:

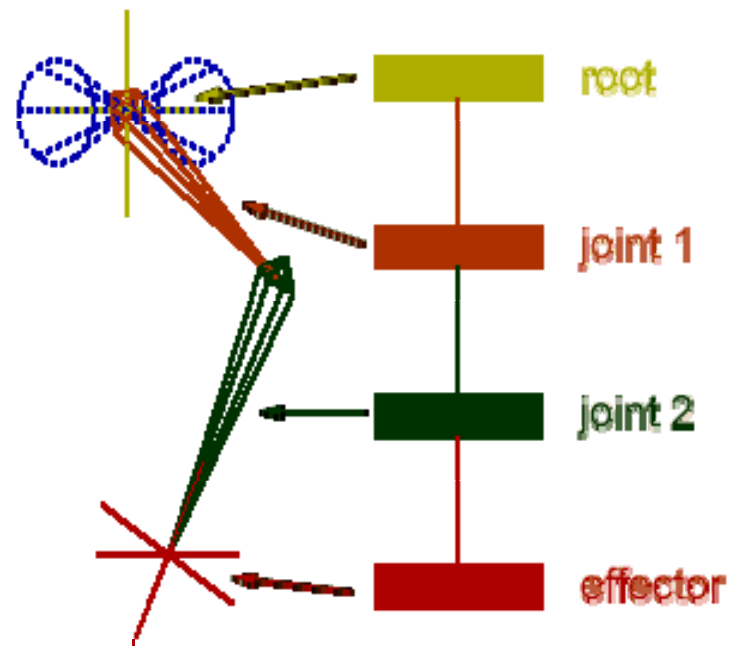
$$\mathbf{p}(t) = \sum_{k=0}^L P_k B_k^L(t)$$

- where $B_k^L(t)$ are the Bernstein polynomials, and the k -th Bernstein polynomial is defined as:

$$B_k^L(t) = \binom{L}{k} (1-t)^{L-k} t^k \quad \text{and} \quad \binom{L}{k} = \frac{L!}{k!(L-k)!} \text{ for } L \geq k$$

Skeleton

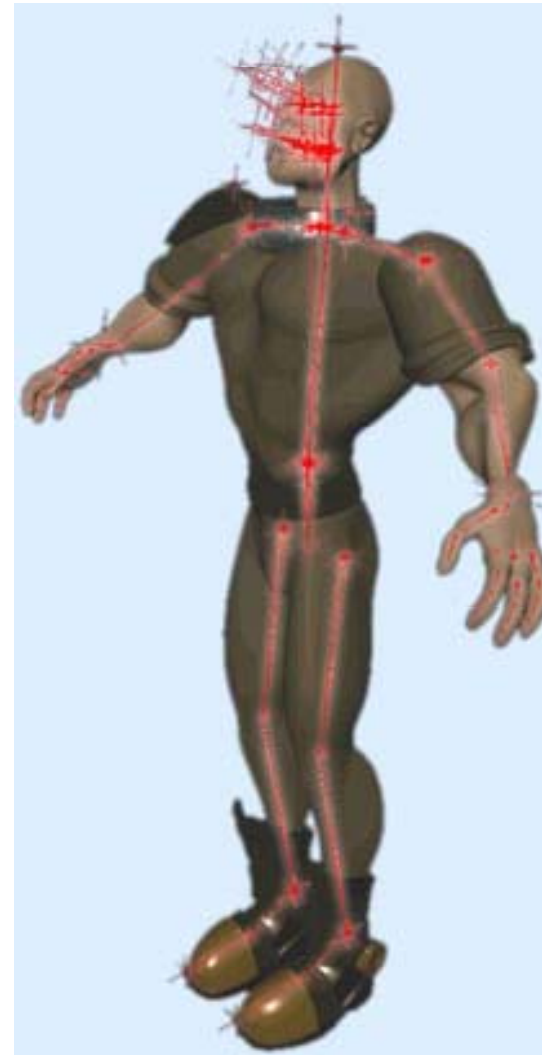
- restricted transformations
- no drawn geometry, just “bones”



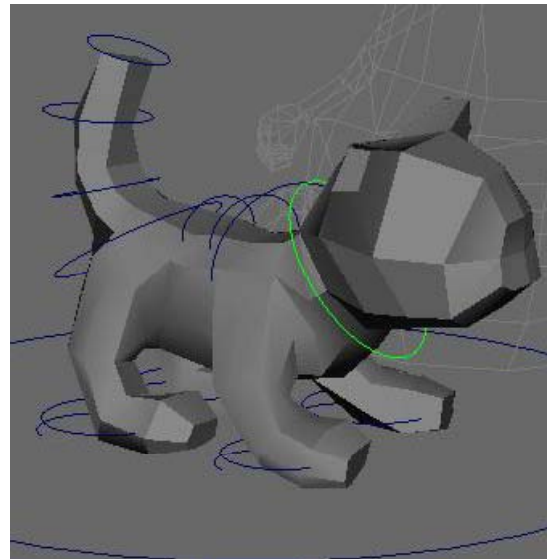
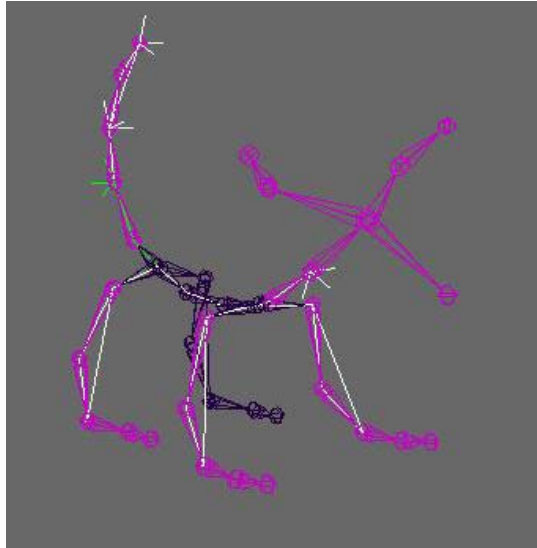
Use of skeleton

- support for direct manipulation
- support for inverse kinematics
- skinning to build models

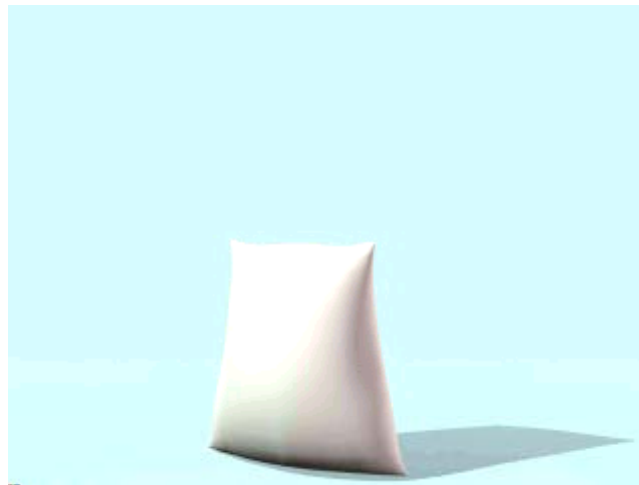
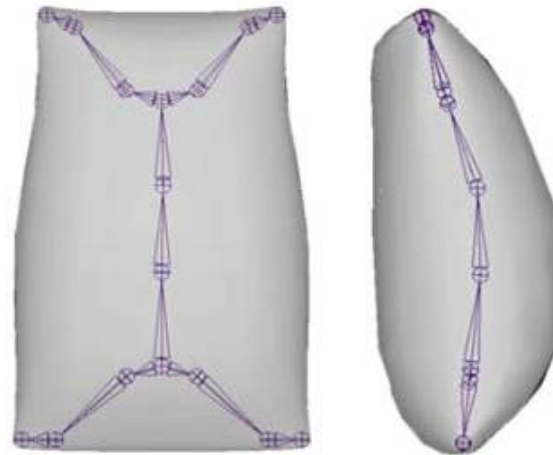
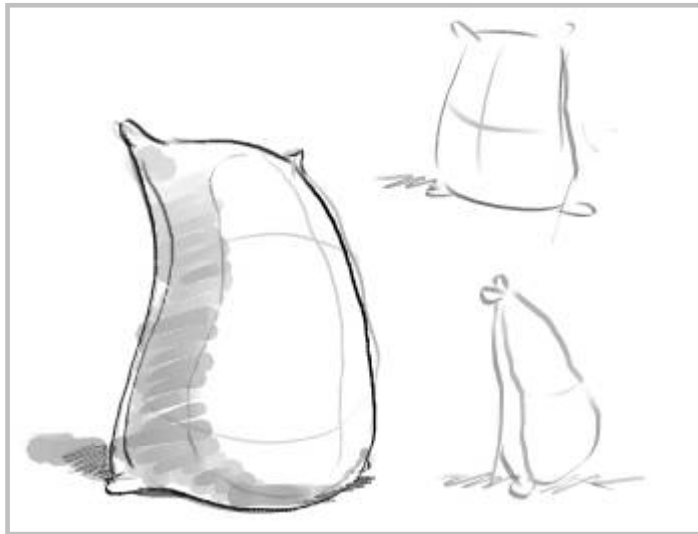
Skinning characters



Even for non-humans



Even for non-animals...



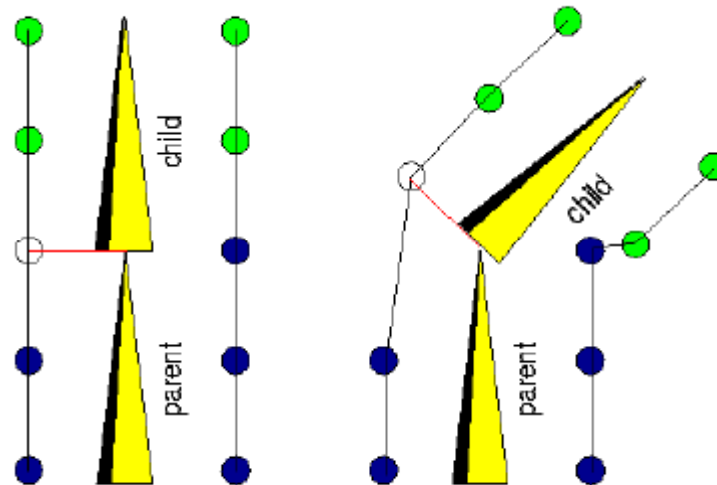
Skeleton deformation

- Skeleton (IK) inside the "skin"

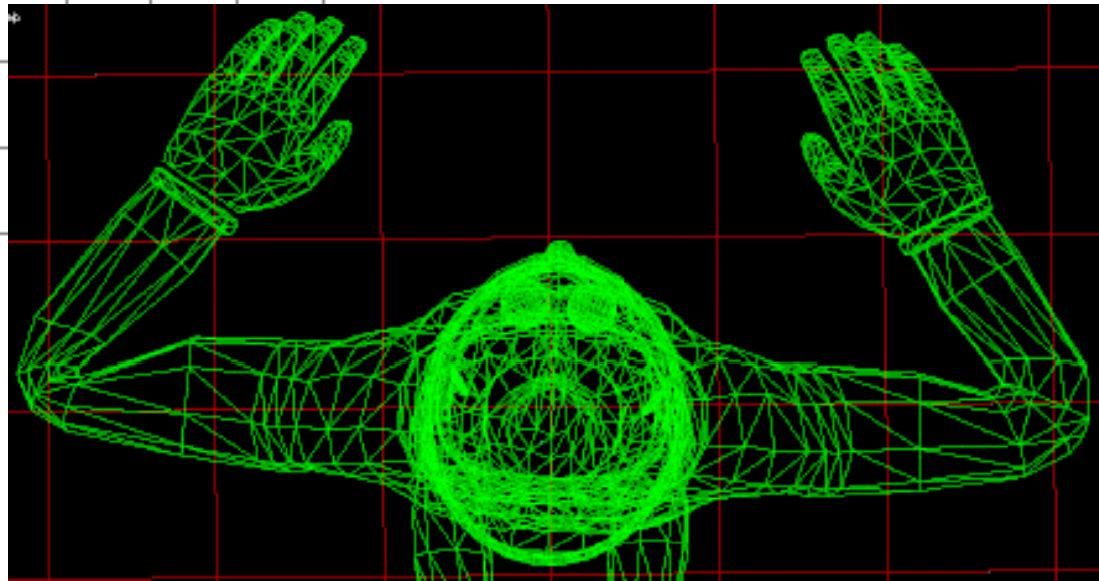
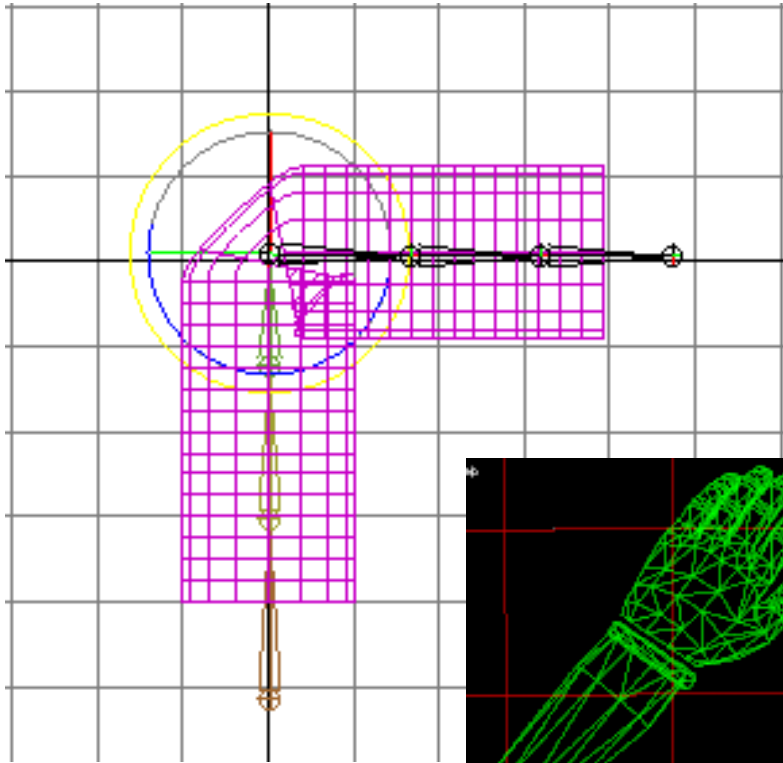


Skeleton deformation

- Associate each point with nearest link
- When link moves, transform its points.

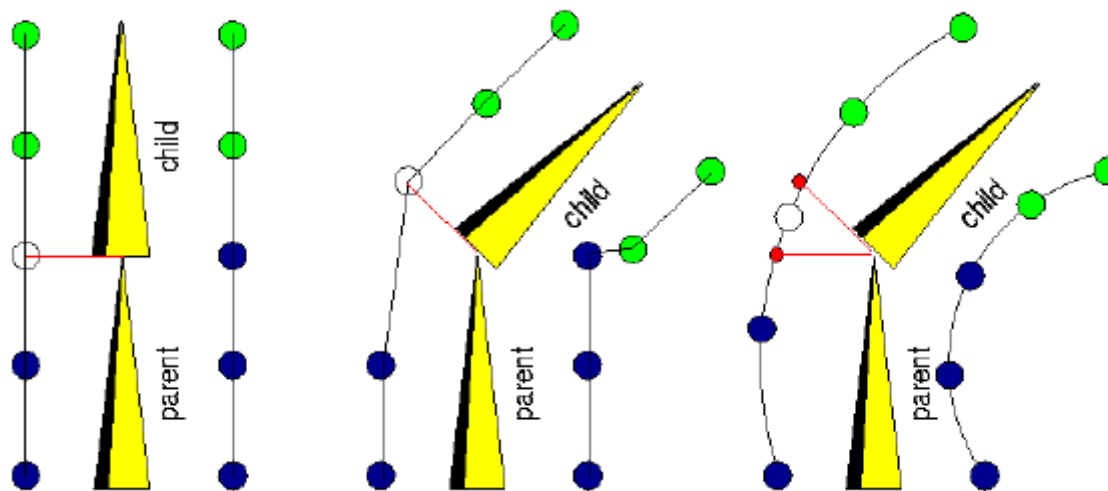


Problem: collapsing, kinking



Point weights

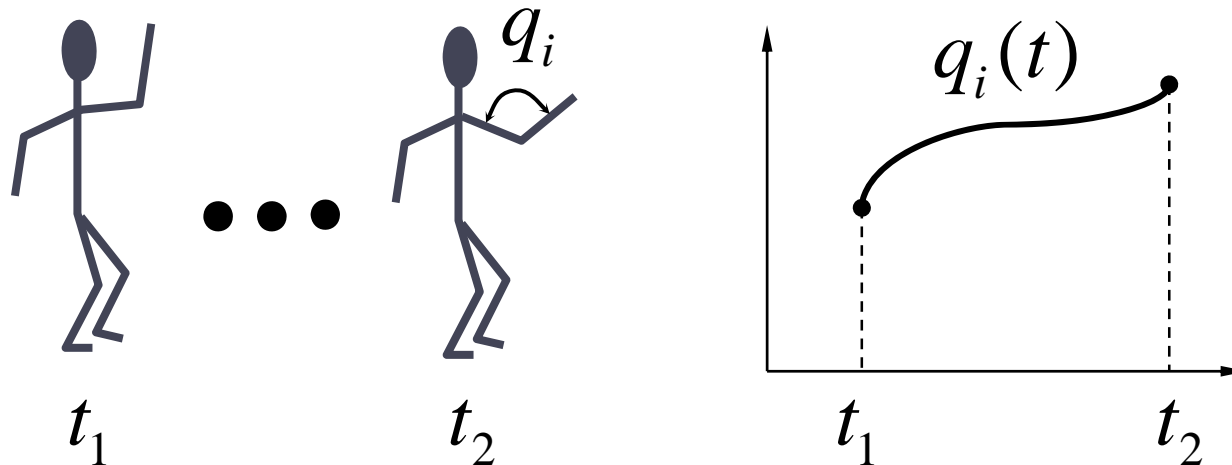
- Each point gets affected by several links
- Take weighted average
- Adjust the weights until it looks good



Animating articulated Models

- rigid parts
- connected by joints

They can be animated by specifying the joint angles, or the end-effectors (e.g., hand, foot) as functions of time.



Kinematics vs. Dynamics

Kinematics

Describes the positions of the body parts as a function of the joint angles.

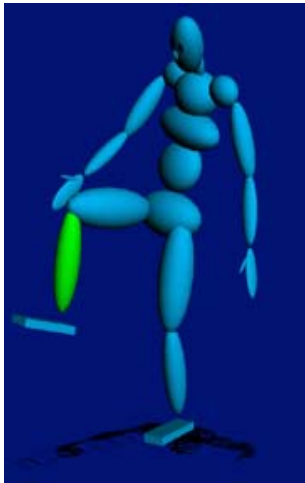
Dynamics

Describes the positions of the body parts as a function of the applied forces.

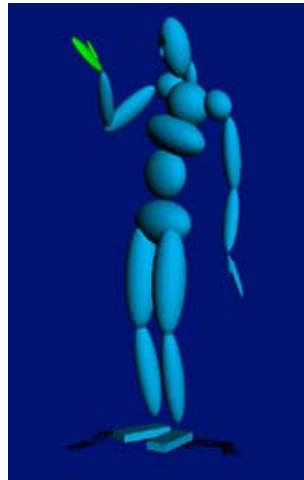
Forward Kinematics

Describes the positions of the body parts as a function of the joint angles.

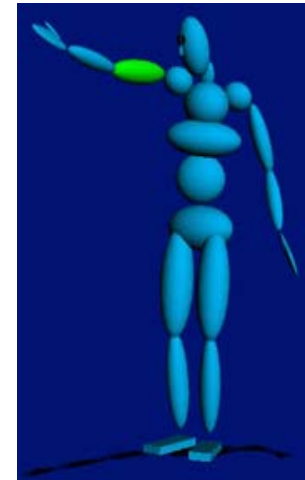
1 DOF: knee



2 DOF: wrist

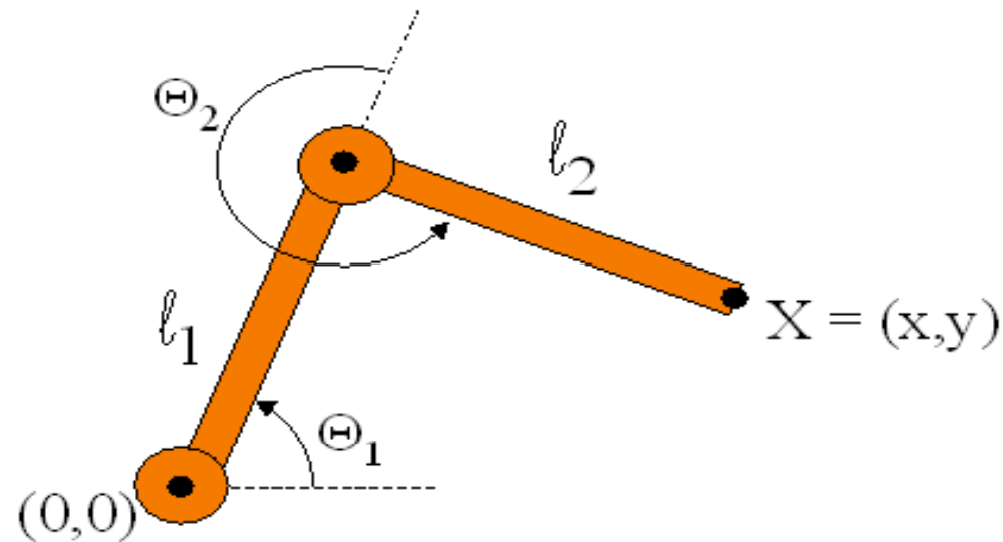


3 DOF: arm



Forward Kinematics

- Animator specifies joint angles
- Computer finds position of end-effector: X



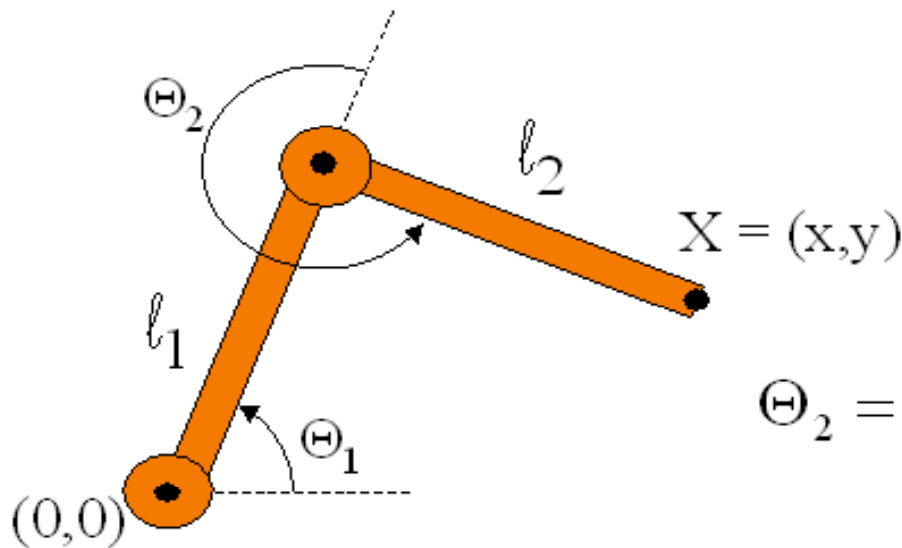
$$X = (l_1 \cos \Theta_1 + l_2 \cos(\Theta_1 + \Theta_2), l_1 \sin \Theta_1 + l_2 \sin(\Theta_1 + \Theta_2))$$

Inverse kinematics

- Goals
 - Keep end of limb fixed while body moves
 - Position end of limb by direct manipulation
 - (More general: arbitrary constraints)

Inverse Kinematics

- Animator specifies end-effector positions
- Computer finds joint angles



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

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

Kinematics Summary

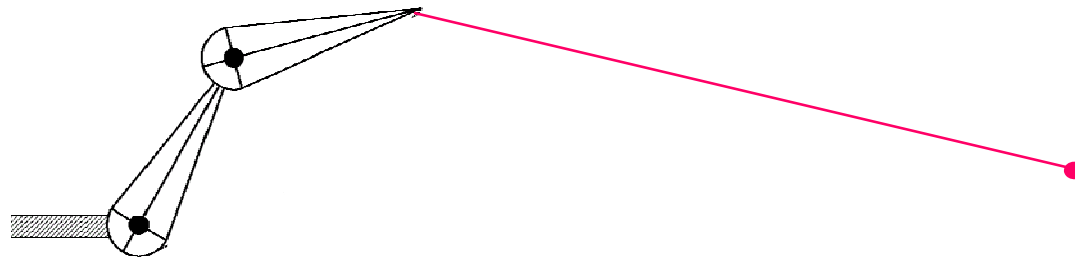
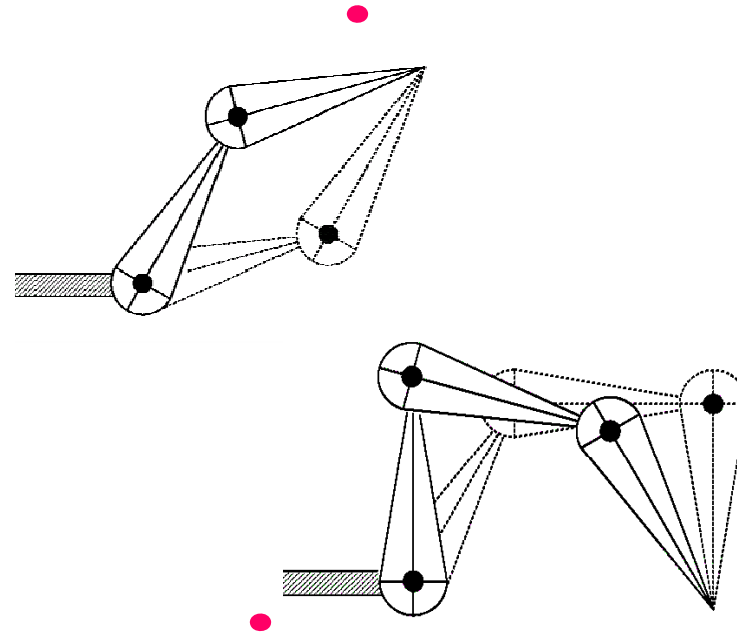
- Forward Kinematics
 - Animator has complete freedom over the movements of any part of the structure
 - Amount of work is function of the complexity of the structure, much more expensive when dealing with complex structures
- Inverse Kinematics
 - Does not leave much scope for the animator to inject 'character' into the movements
 - Model does not possess the interpretation of a human being
 - Overall movement of structure depends on the formula -> same footprints, same movements

What makes IK hard?

- Not always a unique solution
- Not always well-behaved
- Nonlinear problem
- Joint limits

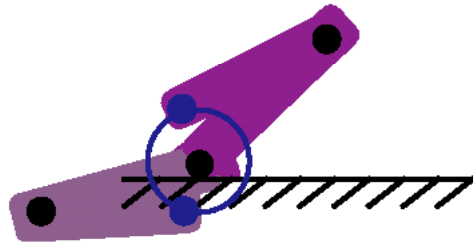
Not always a unique solution

- Disjoint solutions:
- Continuum of solutions:
- No solution:

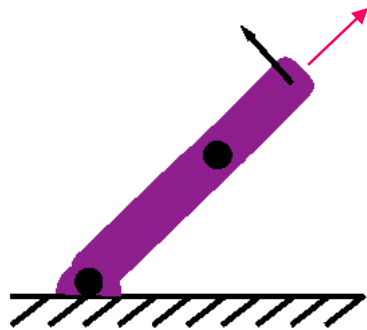


Not always well-behaved

- Small change in X can cause big change in θ



- Changing θ might not move end towards X



Forward Kinematics - details

- The local and world matrix construction within the skeleton is an implementation of *forward kinematics*
- Forward kinematics refers to the process of computing world space geometric descriptions (matrices...) based on joint DOF values (usually rotation angles and/or translations)

Kinematic Chains

- For your spider animation, you only have to worry about linear kinematic chains, rather than the more general hierarchies (i.e., only legs rather than an entire body with multiple branching chains)

Forward Kinematics

- The following vector represents the array of M joint DOF values

$$\mathbf{\Phi} = [\phi_1 \quad \phi_2 \quad \dots \quad \phi_M]$$

- The following vector represents an array of N DOFs that describe the end effector in world space. :

$$\mathbf{e} = [e_1 \quad e_2 \quad \dots \quad e_N]$$

For example, if our end effector is a full joint with orientation, \mathbf{e} would contain 6 DOFs: 3 translations and 3 rotations. If we were only concerned with the end effector position, \mathbf{e} would just contain the 3 translations.

Forward Kinematics

- The forward kinematic function $f()$ computes the world space end effector DOFs from the joint DOFs:

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

Inverse Kinematics

- The goal of inverse kinematics is to compute the vector of joint DOFs that will cause the end effector to reach some desired goal state
- In other words, it is the inverse of the forward kinematics problem

$$\Phi = f^{-1}(\mathbf{e})$$

Inverse Kinematics Issues

- IK is challenging because while $f()$ may be relatively easy to evaluate, $f^{-1}()$ usually isn't
- For one thing, there may be several possible solutions for Φ , or there may be no solutions
- Even if there is a solution, it may require complex and expensive computations to find it
- As a result, there are many different approaches to solving IK problems

Analytical vs. Numerical Solutions

- One major way to classify IK solutions is into analytical and numerical methods
- Analytical methods attempt to mathematically solve an exact solution by directly inverting the forward kinematics equations. This is only possible on relatively simple chains.
- Numerical methods use approximation and iteration to converge on a solution. They tend to be more expensive, but far more general purpose.
 - Jacobian methods
 - Method of Cyclic Coordinate Descent (CCD)

Resources

- http://graphics.ucsd.edu/courses/cse169_w04/welman.pdf
- Lander: Oh My God, I Inverted Kine – download from here:
graphics.cs.cmu.edu/nsp/course/15.../lander_gamedev_sept98.pdf
- *Lander: Making Kine More Flexible:*
graphics.cs.cmu.edu/nsp/.../lander_gamedev_nov98.pdf