

Recall Our Approach To Animation

We create animated behavior just like movie projectors

- display a sequence of still images in rapid succession
- creates the illusion of continuous motion
- typically want 30 frames/sec, and definitely higher than 10

Given some parameterized geometric model

- for every frame, we calculate the correct parameter values
- and we draw the scene in its current state
- just need to figure out how to specify/control these parameters

Some Overall Goals in Animation

Realistic motion

- a special case of the overall photorealism motive

Flexibility & Expressiveness

- want to support the widest possible range of animation
- system that can only produce a single walking motion is boring

Ease of control

- something nigh impossible to control does us little good

Traditional (Manual) Animation

Every frame created individually by a human

- that's 24 frames/second at traditional movie speeds
- roughly 130,000 frames for a 1.5 hour movie

Over time, a general process evolved to support efficiency

- start with set of drawings outlining clip — [storyboard](#)
- senior artists sketch important frames — [key frames](#)
 - typically occur when motion changes
- lower-paid artists draw the rest of the frames — [in-betweens](#)
- all line drawings are painted on cels
 - generally composed in layers, hence the use of acetate
 - background changes infrequently, so can be reused
- photograph finished cel-stack onto film

Computer-Assisted Cel Animation

Cel animation has been in use for a long time (e.g., at Disney)

- but we can use computers to help expedite the process
- draw sketches with digital systems
- use digital paint programs for coloring
- can even try to automatically generate in-betweens

Computer-assisted systems are now quite common

- Disney makes heavy of digital drawing, painting, compositing
- 2-D in-betweening is hard to get right
- morphing a 2-D sketch doesn't give the impression of 3-D
- humans are still much better at this

Computer-Generated Animation

This is the kind of animation we're really interested in here

- start with some 3-D model of the scene
- vary parameters to generate desired pose for all objects
- render scene to produce one frame
- repeat for all 130,000 frames

So how will we control these parameters?

- manually set them for each frame
- key-framing
- generate them procedurally
- execute behavioral scripts (e.g., follow the fish in front of you)
- motion capture
- physical simulation

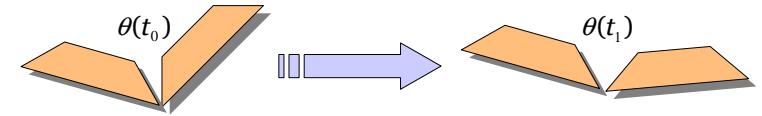
Key-Framing

We've associated a set of parameters with our model

- joint angles, positions, etc.
- we view each of these as a function of time

In key-framing, we specify parameter values at specific times

- and let the computer interpolate the in-between values

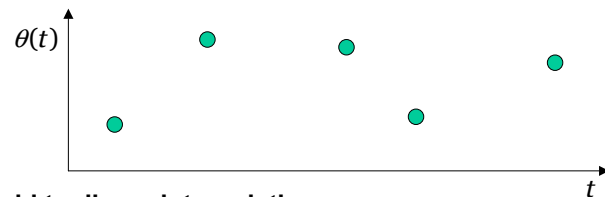


How do we accomplish this interpolation?

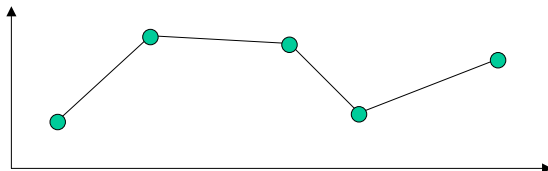
Interpolating Motion Parameters

For a given parameter, we've specified some fixed values

- these are the key frame values



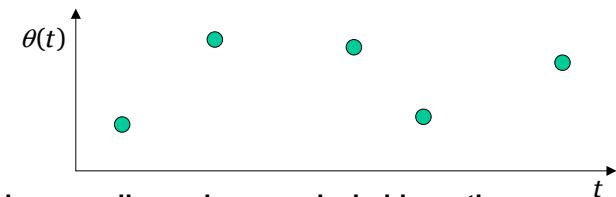
We could try linear interpolation



Interpolating Motion Parameters

For a given parameter, we've specified some fixed values

- these are the key frame values



But this generally produces undesirable motion

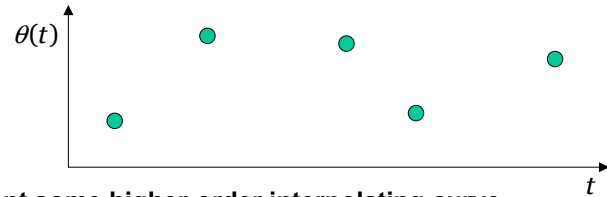
- during each interpolated span, we move with constant velocity
- and then instantaneously change at each key point
- but this is highly non-physical!

What else might we try?

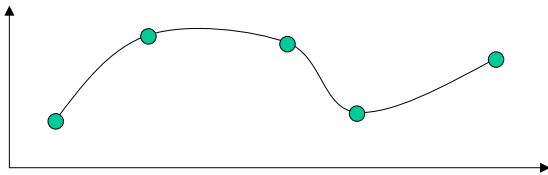
Interpolating Motion Parameters

For a given parameter, we've specified some fixed values

- these are the key frame values



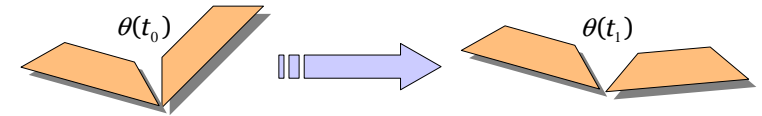
We want some higher-order interpolating curve



Creating Key-Frames

In key-framing, we specify parameter values at specific times

- and let the computer interpolate the in-between values



We discussed how to do the interpolation

- but what about the specification of these parameters?
- how do I get my articulated action figure into my favorite pose?

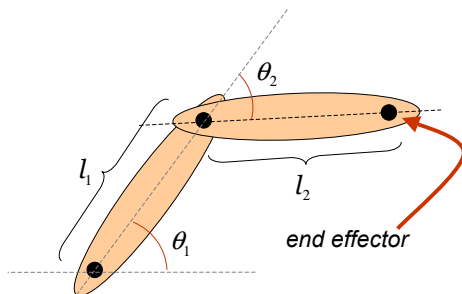
Forward Kinematics

Position of end effector is a function of the state of all joints

more formally: $\mathbf{x} = f(\Theta)$

\mathbf{x} : position of end effector

Θ : angles, positions, ... of joints



For this simple 2D example

$$\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} g(\theta_1, \theta_2) \\ h(\theta_1, \theta_2) \end{Bmatrix}$$

Forward Kinematics

Given an articulated human, how do we make him wave?

- rotate the shoulder into position
- and then the elbow and wrist
- and finally rebalance other parts of the body

This is tedious. We'd much rather directly move the hand

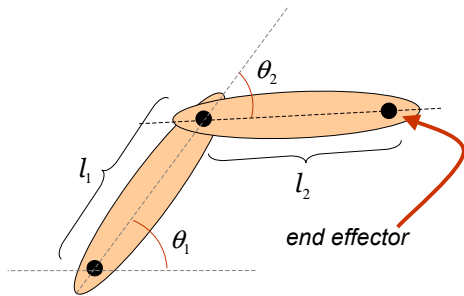
- we can use inverse kinematics
 - kinematics: derive motion from positions, angles, velocities, ...
 - inverse kinematics: determine joint angles from positions
- let the user drag the tip of the hand into place

Inverse Kinematics

Automatically derive joint angles from end effector position

forward kinematics: $\mathbf{x} = f(\Theta)$

inverse kinematics: $\Theta = f^{-1}(\mathbf{x})$



Can write formulas

$$\theta_1 = g(x_1, x_2)$$

$$\theta_2 = h(x_1, x_2)$$

for this simple 2D example.

Inverse Kinematics

Real models are much more complex than our simple example

- a human has around 200 degrees of freedom (parameters)
- the mapping of parameters to effector positions is non-linear
- inverting this function is not possible
- must rely on numerical methods

Suppose we specify locations for end effectors

- we need to compute a model configuration to achieve pose
 - there may be many possible parameter settings that work
 - need to pick “best” one — minimize work, maintain balance, ...
 - similarly, there may not be any parameter settings that work
 - need to pick one that is “close enough”
- both involve some kind of optimization algorithm

What Key Framing Doesn't Address

Control point selection

- how often do we need to specify a key value?
- what precise key values work best?

Key value interpolation

- what interpolation method will give us what we want?

Physical constraints

- how do we avoid non-physical motion?
 - 360° head twists, infinite instantaneous accelerations, ...
- how do we know that two objects don't interpenetrate?
- how do we maintain contact at appropriate points?
 - feet must touch floor when walking

Procedural Animation

To specify a procedural animation

- sit down and write some code — the animator as programmer
- input: current time
- output: correct parameter value
- usually combine lots of little procedures together
 - one procedure for walk, one for run, one for hop, ...

There is a clear tradeoff between procedures & interaction

- think of the analogy of creating a 2-D illustration
- if it's simple, we can probably quickly do it interactively
- but if it's very complex & regular, coding is probably quicker

Behavioral Animation

Closely related to procedural animation

- specify how object reacts to the world around it

A common example of this approach is “flocking”

- motion of an object is directly determined by others nearby
 - analogous to Conway’s Life
- simple rules lead to interesting emergent behaviors
- very helpful for choreographing large-scale action
 - wildebeests in *Lion King*
 - army of mounted soldiers in *Mulan*

Motion Capture

One of the most common ways of creating surface models:

- build something in clay and put it in a laser scanner

One of the most common ways of creating motion:

- strap a bunch of sensors on a person & record their motion
 - track the location of several reference points
 - convert this to joint angles & map to articulated model
- several technologies available
 - optical
 - magnetic tracking
 - instrumented exoskeletons

But it can be hard to edit

