Animation

- Overall goals
- Traditional animation
- Computer-assisted animation
- Computer-generated animation
  - *Key framing*
  - *Forward kinematics*
  - *Inverse kinematics*
  - *Procedural animation*
  - *Motion capture*

Our Approach to Animation

We create motion just like movie projectors

- Display a sequence of still images in rapid succession
- Creates the illusion of continuous motion
- Typically want 30 frames/sec (fps); definitely more than 10 fps

Given some parameterized geometric model

- For every frame, we calculate the correct parameter values
- And we draw the scene in its current state

We "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

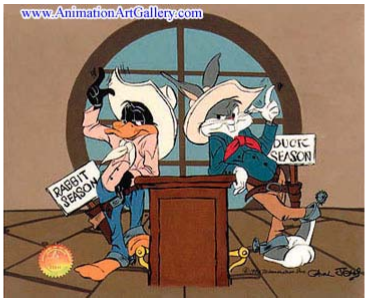
Flexibility and expressiveness

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

Ease of control

- A model that's impossible to control does us little good

Traditional (Manual) Animation



Every frame is created individually by a human

- That's 24 frames/sec at traditional movie speeds
  - *Roughly 130,000 frames for a 1.5 hr movie*

A general pipeline evolved to support efficiency

- Start with a ***storyboard***
  - *A set of drawings outlining the animation*
- Senior artists sketch important frames – ***Keyframes***
  - *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 it can be reused*
- Photograph finished ***cel-stack*** onto film

Computer-Assisted Cel Animation

Cel animation has been in use 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 generate the in-betweens automatically

Computer-assisted systems are now common

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

Computer-Generated Animation

This is the kind of animation in which we are most interested

- Start with some 3D 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?

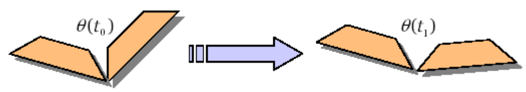
- ***Manually*** set them for each frame
- ***Key-framing***
- Generate them ***procedurally***
- ***Motion capture***
- ***Physical simulation***
- ***Behavioral animation*** (e.g., follow the fish ahead)

Key-Framing

We've associated a set of parameters with our model

- Positions, orientations, joint-angles, 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

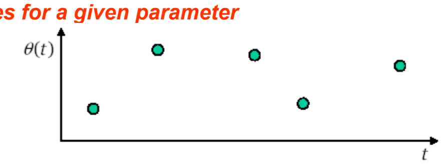


How do we accomplish this interpolation?

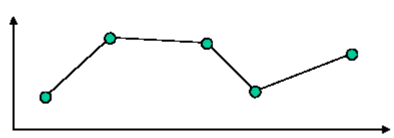
Interpolating Motion Parameters

We have specified some fixed values for a given parameter

- These are the key-frame values



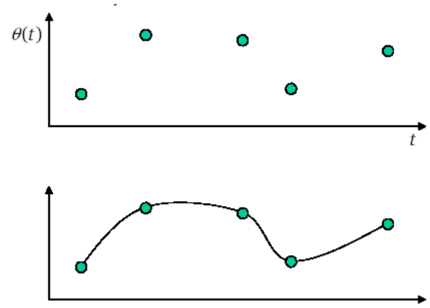
- Try linear interpolation



- But this generally produces undesirable motion
  - *During each interpolated span, we move with constant velocity and then change velocity at each key point*
  - *This is highly non-physical*
- What else might we try?

Interpolating Motion Parameters

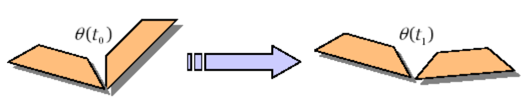
We want some higher-order interpolating curve



Creating Key-Frames

What about the specification of these parameters?

- How do I get my articulated figure into my desired pose?

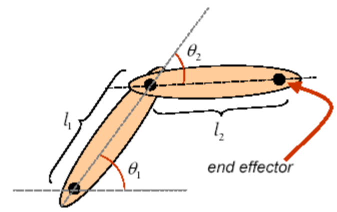


Forward Kinematics

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

- More formally: **x** = **f**(θ)
  - ***x***: *position of end effector*
  - *θ: angles of joints*
- For this simple 2D example:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} g(\theta_1,\theta_2) \\ h(\theta_1,\theta_2) \end{bmatrix}$$



Forward Kinematics

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

- Rotate the shoulder into position
- And then the elbow
- And then the wrist
- Etc.
- And finally re-balance other parts of the body

This is tedious

- We'd much rather directly move the hand

We can use inverse kinematics

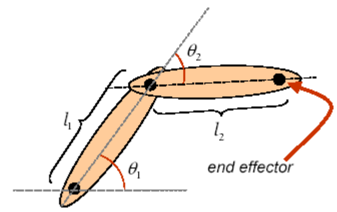
- Let the user drag the tip of the hand into place
- Determine joint angles from hand position

Inverse Kinematics

Automatically derive joint angles from end effector position

- Forward kinematics: $\mathbf{x} = \mathbf{f}(\theta)$
- Inverse kinematics: $\theta = \mathbf{f}^{-1}(\mathbf{x})$
- For this simple 2D example:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} g^{-1}(x,y) \\ h^{-1}(x,y) \end{bmatrix}$$



Inverse Kinematics

Real humans are much more complex than our simple example

- A human has around 200 degrees of freedom
- 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 parameter settings that work*
    - Need to pick a “best” one
      - minimize work
      - maintain balance
      - etc.
  - *Alternatively, 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 the floor when walking*

Procedural Animation

To specify procedural animation

- Write some code – the animator as programmer
  - *Input: current time*
  - *Output: 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 and interaction

- If it's simple, we can probably quickly do it interactively
- If its complex and regular, coding is probably quicker

Demo: Ken Perlin's procedural actors

- www.mrl.nyu.edu/~perlin/experiments/emotive-actors

Motion Capture

Currently a popular way of creating motions

- Strap a bunch of sensors on a person and record their motion
  - *Several technologies available*
    - Instrumented exoskeletons
    - Magnetic
    - Optical

- Track the location of several reference points
- Convert this to joint angles and map to articulated model
- But it can be hard to edit

Motion Capture

Computer-Generated Animation

Physical simulation

- Particle dynamics
- Spring-mass systems
- Fluids
- Rigid-body dynamics
- Articulated dynamics

Behavioral animation

Physical Simulation

We usually want realistic looking motion

- People are extremely experienced at observing body language
- They pick up on unnatural human motion instantly

Some of the methods we've discussed can achieve realism

- If our animator makes good enough key frames
- Or we write good enough procedural scripts
- Or we strap a bunch of sensors on an actor

But there's another good alternative

- Why not just ***simulate the relevant physical laws*** ?
- Then we'll know that the motion is natural
- And we'll still have decent control over it

Dynamics

Direct physical simulation (e.g., with Newton's laws of motion)

- Specify positions, masses, forces,…
- Apply relevant laws to compute accelerations, velocities, positions as a function through time
  - *We can express the relevant laws as differential equations*

$$\mathbf{f} = m\,\mathbf{a} \rightarrow \frac{d^2\mathbf{x}}{dt^2} = \frac{\mathbf{f}}{m} \quad \text{or} \quad \ddot{\mathbf{x}} = \frac{\mathbf{f}}{m}$$

- And in general we must solve them numerically

Caveat

Keep in mind that we don't always want to mimic nature

- Exact replication of reality isn't always artistically interesting
- We can invent our own physical laws
  - *"Cartoon laws of physics"*
  - *In particular, phantom forces are easy to add*

We may want to mimic the motion of a golf ball

- But how do we make sure it lands in the hole for a "hole-in-one", if that is what the animator wants his golfer to do?