CS 231

Basics of Computer Animation
Animation Techniques

Keyframing

Motion capture

Physics models
Keyframe animation

Highest degree of control, also difficult
Interpolation affects end result
Timing must also be considered
Constraints can help but must be set up
The keyframing process

Iterate:

- adjust trajectory
- play back motion

Depends on:
- interpolation between frames
- speed along path
- parameters for model
The basic animation engine

Motion generation

frame = 0;
Until (forever) {
    set scene(frame);
    take picture(frame);
    frame++;
}

Offline representation

mymovie.mpg = make_mpeg (all frames)
Animation systems – animate what?

Particles (or other elements)

Linked Particles (or elements)

Rigid bodies (RB)

Linked RB

Other - ? (e.g. Eulerian Grids)
Keyframe any controllable parameters:

- joint angles
- location/orientation
- scale/shear
- shape (deformations)
- materials and textures
- camera settings
- lighting (strobe, etc)
- about any parameter
Keyframing – 3D characters

Complete control

Quality relies on skilled animator
Keyframing – 3D characters

Representation

1) Skeleton
   Origin (root)
   Joint centers/
   bones lengths

2) Keyframes
   Pos/Rot Root (x)
   Joint Angles (q)
Kinematics – study of static movement

Forward: \( x = f(\alpha, \beta) \)

Inverse: \( \alpha, \beta = f^{-1}(x) \)

easy to compute

intuitive to control
positioning hand/feet
higher-level goals

Note: kinematics has no notion of dynamics
Kinematics and IK - rely on skeleton motion the basis of all human (and many non-human) characters

Kinematics solved directly

Used for character motion based on skeleton
Forward Kinematics

\[ x = l_1 \cos(\theta_1) + l_2 \cos(\theta_2) + l_3 \cos(\theta_3) \]

\[ y = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) + l_3 \sin(\theta_3) \]
Hierarchical Animation
Hierarchical Animation

A series of transformations on an object can be applied as a series of matrix multiplications:

\[ p = T(x_0, y_0, z_0)R(\theta_0)R(\phi_0)R(\sigma_0)T(0, h_0, 0)R(\theta_1)R(\phi_1)R(\sigma_1)T(0, h_1, 0)R(\theta_2)T(0, h_2, 0)R(\theta_3)R(\phi_3)x \]

- \( p \): position in the global coordinate
- \( x \): position in the local coordinate

\((h_3, 0, 0)\)
Keyframing skeleton with constraints

Joint limits
Pos limits
IK handles
Keyframe Interpolation:
Types of interpolation:

- linear
- b-spline
- cardinal
Keyframing in practice
Problems with interpolation curves

Note, non-uniform velocity
Keyframing to control velocity
Velocity depends on interpolation style

**Linear**

```
[Diagram showing linear interpolation with constant velocity]
```

**Ease In/Ease out**

```
[Diagram showing ease in/ease out interpolation with varying velocity]
```
Keyframing Ease-in/Ease out

Sinusoidal
  offset cos/sin
  half period
  piecewise

Function
  cubic
  constant acceleration
Ease-in/Ease-out (EIEO) Function

\[ s = \text{ease}(t) \]

(Normalized distance and time)
$s = ease(t) = \frac{\sin(t\pi - \pi/2) + 1}{2}$
$EIEO : Single \text{ Cubic}$

$$s = ease(t) = -2t^3 + 3t^2$$
Provide linear (constant velocity) middle segment

\[
ease(t) = \begin{cases} 
(k_1 \frac{2}{\pi} \sin \left( \frac{t \pi}{2} - \frac{\pi}{2k_1} \right) / f \\
\left( \frac{k_1}{\pi / 2} + t - k_1 \right) / f \\
\left( \frac{k_1}{\pi / 2} + k_2 - k_1 + (1 - k_2) \frac{2}{\pi} \sin \left( \frac{\pi(t - k_2)}{2(1 - k_2)} \right) \right) / f
\end{cases}
\]

\[
f = k_1 \frac{2}{\pi} + k_2 - k_1 + (1 - k_2) \frac{2}{\pi}
\]
EIEO : Sinusoidal Piecewise
EIEO : Constant Acceleration
EIEO : Constant Acceleration
EIEO : Constant Acceleration
EIEO: Constant Acceleration

Piecewise Sinusoid

Constant Acceleration
Keyframing with Ease-in/Ease out

Action Transitions
Keyframing with Ease-in/Ease out

Combining actions

![Diagram showing ease-in and ease-out over time](image)
Keyframing with Ease-in/Ease out

Directing/initiating actions and their “transition”
Rigging

Skeleton = “bones”
Geometry = “skin”
Rigging
Rigging

**Rigid Bind:**
Points on geometry get same transforms as bones

**Smooth Bind:**
Average weights of bones
Weights controlled by user or based on distance from joint
Free-form deformation

Deformation of the lattice then implicitly defines a deformation of the space.

Track positions of key points within the deformed sequence of lattices.
Free-form deformation – Example, single line...

Get object
Draw polyline
Map vertices to polyline
Warp polyline
Reposition vertices to polyline
Free-form deformation
Free-form deformation
Free-form deformation
Grid Deformation

Overlay 2D grid on top of object

Map object vertices to grid cells (create local coordinate system)
User distorts 2D grid vertices

Object vertices are remapped to local coordinate system of 2D grid by using bilinear interpolation
Grid Deformation
Grid Deformation

For each vertex
Identify cell
Local u,v coordinate
Grid Deformation

Bilinear interpolation

Pu0 = (1-u)*P00 + u*P10
Pu1 = (1-u)*P01 + u*P11
Puv = (1-v)*Pu0 + v*Pu1
Grid Deformation
Grid Deformation
Free-Form Deformations

Define local coordinate system for deformation

\[ (S, T, U) \]
FFD – position of point

![Diagram of a cube with labeled points S, T, and U]
FFD - 3D example
Rigging

Deform a character arm
Rigging – issues
Rigging - advanced

Other bindings
Muscles bulge
How to set up?

Automation
Compute skeleton from model

Issues
Joint distance problems
Creases
Keyframing

Complete control

Quality relies on skilled animator
Procedural Animation

Rule based animation that changes/evolves over time.

Types of procedural systems:
- Function-based
- Animating with noise
- Grammars and fractals
Procedural Animation

Function-based
Animating with mathematical rules:
sine, cos, exp, ln, etc.
Often depends on geometry

Scripting
Animating with noise

Perlin “invents” noise

What is noise()?

A 'random' function that has statistical properties perceived by the human visual system to include 'appropriate' detail at a variety of scales.
Animating with noise

Noise adds lifelike feel to models and movement

Powerful tool in layering and combining

Must model constraints explicitly

Animators skill play a heavy hand in success
Yoichiro Kawaguchi
Procedural Animation

Animation example with kinematics

Procedural motion
Hip, knee, ankle

Rules turn on and off prescribed curves (scaled sin, cos) to create walk cycle
Particle systems

Animating complicated systems with many small elements and simple rules

Very common method for doing many effects: water spray, explosions, fireworks

Rules can include:
- Distribution rate
- Movement
- Color, size, shape
- Life span, age-related changes
Particle systems
Moving particles

\((X, V)\)
Particle dynamics

Newtonian particles: $f = m \, a$

$$a = \frac{dv}{dt} = \frac{d^2x}{dt^2}$$

Given $f$, find the position $x$ so we integrate

Can use a simple approximation:

$$v_{t+1} = a_t \, Dt + v_t$$

$$x_{t+1} = v_t \, Dt + x_t$$

But *physics isn’t necessary* for particles…
Particle systems – Procedural

Applying an acceleration to the particles procedural types include:

Random accelerations
Acceleration toward/away from a point
Acceleration toward/away from a line
Spirals, Vortices, Tornadoes

Particles move in a coherent fashion to produce overall effect

Other prescribed motions (sin(t), etc.)
Rendering particle effects
Particle systems – Rigging to kinematic model

Starting from particle, "rig" follows

Procedurally add walk cycle based on velocity turning radius
Particle systems – Rigging to kinematic model
Particle systems – Rigging to kinematic model

Starting from particle, "rig" follows

Procedurally add walk cycle based on velocity turning radius

How to animate legs?