Animation

CMSC 435/634
Prof. Marc Olano

Keyframe Animation

- From hand drawn animation
  - Lead animator draws poses at key frames
  - Inbetweener draws frames between keys
- Computer animation
  - Can have separate keys for different attributes
  - Interpolate between values at key frames

How to Interpolate

- Linear interpolation
  - Value V_0 at time T_0, V_1 at time T_1
  - Fraction of the way from T_0 to T_1

$$t = (T - T_0)/(T_1 - T_0)$$

  - Lerp/mix equation

$$v = (1-t)V_0 + t\,V_1$$

Spline

- Set of polynomials

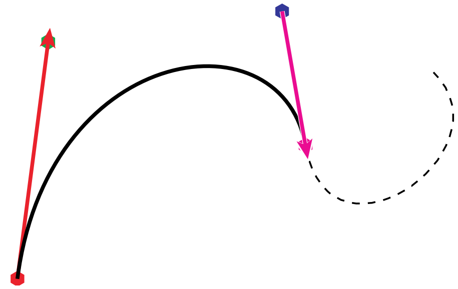
$$\vec{p}(t) = \vec{a}\,t^3 + \vec{b}\,t^2 + \vec{c}\,t + \vec{d}$$

- 1 constraint per coefficient
  - Positions $\vec{p}(t) = \vec{a}\,t^3 + \vec{b}\,t^2 + \vec{c}\,t + \vec{d}$
  - Velocities $\vec{p'}(t) = 3\,\vec{a}\,t^2 + 2\,\vec{b}\,t + \vec{c}$
  - Acceleration $\vec{p''}(t) = 6\,\vec{a}\,t + 2\,\vec{b}$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p'}(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p'}(1) = 3(\vec{p}_3 - \vec{p}_2)$$



Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p'}(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p'}(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{aligned} \vec{p}_0 &= \vec{a}\,0^3 + \vec{b}\,0^2 + \vec{c}\,0 + \vec{d} \\ \vec{p}_3 &= \vec{a}\,1^3 + \vec{b}\,1^2 + \vec{c}\,1 + \vec{d} \\ 3(\vec{p}_1 - \vec{p}_0) &= 3\,\vec{a}\,0^2 + 2\,\vec{b}\,0 + \vec{c} \\ 3(\vec{p}_3 - \vec{p}_2) &= 3\,\vec{a}\,1^2 + 2\,\vec{b}\,1 + \vec{c} \end{aligned}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p'}(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p'}(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} \vec{p}_0 \\ \vec{p}_3 \\ 3(\vec{p}_1 - \vec{p}_0) \\ 3(\vec{p}_3 - \vec{p}_2) \end{bmatrix} = \begin{bmatrix} \vec{a}\,0^3 + \vec{b}\,0^2 + \vec{c}\,0 + \vec{d} \\ \vec{a}\,1^3 + \vec{b}\,1^2 + \vec{c}\,1 + \vec{d} \\ 3\,\vec{a}\,0^2 + 2\,\vec{b}\,0 + \vec{c} \\ 3\,\vec{a}\,1^2 + 2\,\vec{b}\,1 + \vec{c} \end{bmatrix}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p'}(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p'}(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} \vec{p}_0 \\ \vec{p}_3 \\ 3(\vec{p}_1 - \vec{p}_0) \\ 3(\vec{p}_3 - \vec{p}_1) \end{bmatrix} = \begin{bmatrix} \vec{d} \\ \vec{a} + \vec{b} + \vec{c} + \vec{d} \\ \vec{c} \\ 3\,\vec{a} + 2\,\vec{b} + \vec{c} \end{bmatrix}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p}'(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p}'(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} \vec{p}_0 \\ \vec{p}_3 \\ 3(\vec{p}_1 - \vec{p}_0) \\ 3(\vec{p}_3 - \vec{p}_2) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p}'(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p}'(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} \vec{p}_0 & & & \\ & & & \vec{p}_3 \\ -3\,\vec{p}_0 & +3\,\vec{p}_1 & & \\ & & -3\,\vec{p}_2 & +3\,\vec{p}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p}'(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p}'(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p}'(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p}'(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix}$$

Bezier Spline

- All constraints from *control points*:

$$\vec{p}(0) = \vec{p}_0; \qquad \vec{p}(1) = \vec{p}_3;$$
$$\vec{p'}(0) = 3(\vec{p}_1 - \vec{p}_0); \quad \vec{p'}(1) = 3(\vec{p}_3 - \vec{p}_2)$$

- Resulting equations:

$$\begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix}$$

Bezier Basis Functions

- Computing position

$$\vec{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$$

Bezier Basis Functions

- Computing position

$$\vec{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix}$$

Bezier Basis Functions

- Group by t^i: coefficients

$$\vec{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \left(\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix} \right)$$

Bezier Basis Functions

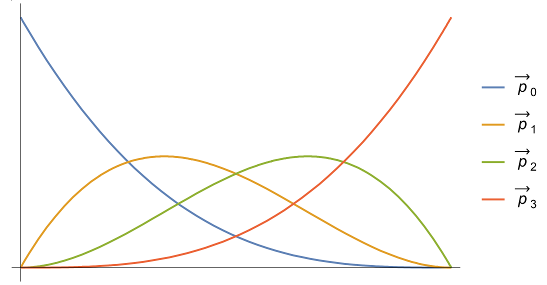
- Group by p_i: Basis Functions

$$\vec{p}(t) = \left(\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix}$$

Bezier Basis Functions

- Cubic Bezier basis functions

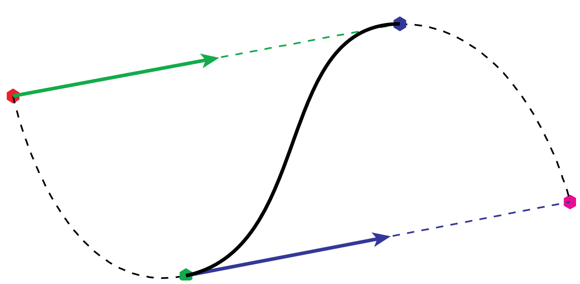
$$\begin{aligned} B_0^3(t) &= -t^3 + 3t^2 - 3t + 1 &&= (1-t)^3 \\ B_1^3(t) &= 3t^3 - 6t^2 + 3t &&= 3(1-t)^2\, t \\ B_2^3(t) &= -3t^3 + 3t^2 &&= 3(1-t)\, t^2 \\ B_3^3(t) &= t^3 &&= t^3 \end{aligned}$$



Catmull-Rom Spline

- Constraints

$$\begin{aligned} \vec{p}(0) &= \vec{p}_1; & \vec{p}(1) &= \vec{p}_2; \\ \vec{p}'(0) &= (\vec{p}_2 - \vec{p}_0)/2; & \vec{p}'(1) &= (\vec{p}_3 - \vec{p}_1)/2 \end{aligned}$$



Catmull-Rom Spline

- Constraints

$$\begin{aligned} \vec{p}(0) &= \vec{p}_1; & \vec{p}(1) &= \vec{p}_2; \\ \vec{p}'(0) &= (\vec{p}_2 - \vec{p}_0)/2; & \vec{p}'(1) &= (\vec{p}_3 - \vec{p}_1)/2 \end{aligned}$$

- Resulting equations:

$$\begin{bmatrix} \vec{p}_1 \\ \vec{p}_2 \\ (\vec{p}_2 - \vec{p}_0)/2 \\ (\vec{p}_3 - \vec{p}_1)/2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$$

Catmull-Rom Spline

- Constraints

$$\vec{p}(0) = \vec{p}_1; \qquad \vec{p}(1) = \vec{p}_2;$$
$$\vec{p'}(0) = (\vec{p}_2 - \vec{p}_0)/2; \quad \vec{p'}(1) = (\vec{p}_3 - \vec{p}_1)/2$$

- Resulting equations:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -{}^1\!/_2 & 0 & {}^1\!/_2 & 0 \\ 0 & -{}^1\!/_2 & 0 & {}^1\!/_2 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$$

Catmull-Rom Spline

- Constraints

$$\vec{p}(0) = \vec{p}_1; \qquad \vec{p}(1) = \vec{p}_2;$$
$$\vec{p'}(0) = (\vec{p}_2 - \vec{p}_0)/2; \quad \vec{p'}(1) = (\vec{p}_3 - \vec{p}_1)/2$$

- Resulting equations:

$$\begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -{}^1\!/_2 & 0 & {}^1\!/_2 & 0 \\ 0 & -{}^1\!/_2 & 0 & {}^1\!/_2 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix}$$

Catmull-Rom Spline

- Constraints

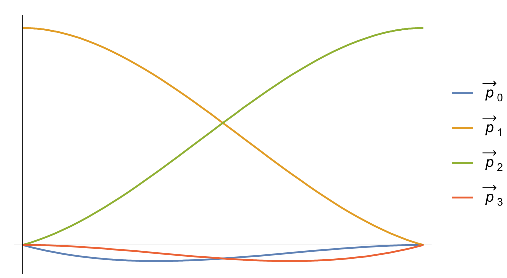
$$\vec{p}(0) = \vec{p}_1; \qquad \vec{p}(1) = \vec{p}_2;$$
$$\vec{p'}(0) = (\vec{p}_2 - \vec{p}_0)/2; \quad \vec{p'}(1) = (\vec{p}_3 - \vec{p}_1)/2$$

- Resulting equations:

$$\begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix} = \begin{bmatrix} -{}^1\!/_2 & {}^3\!/_2 & -{}^3\!/_2 & {}^1\!/_2 \\ 1 & -{}^5\!/_2 & 2 & -{}^1\!/_2 \\ -{}^1\!/_2 & 0 & {}^1\!/_2 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{p}_0 \\ \vec{p}_1 \\ \vec{p}_2 \\ \vec{p}_3 \end{bmatrix}$$

Catmull-Rom Basis Functions

$$B_0(t) = -t^3/2 + t^2 - t/2$$
$$B_1(t) = 3t^3/2 - 5t^2/2 + 1$$
$$B_2(t) = -3t^3/2 + 2t^2 + t/2$$
$$B_3(t) = t^3/2 - t^2/2$$



What to Interpolate

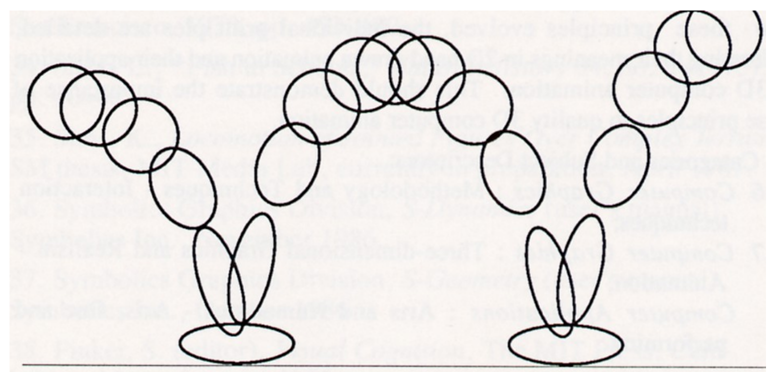
- What controls to artists need?
- How to convert those into transformations?

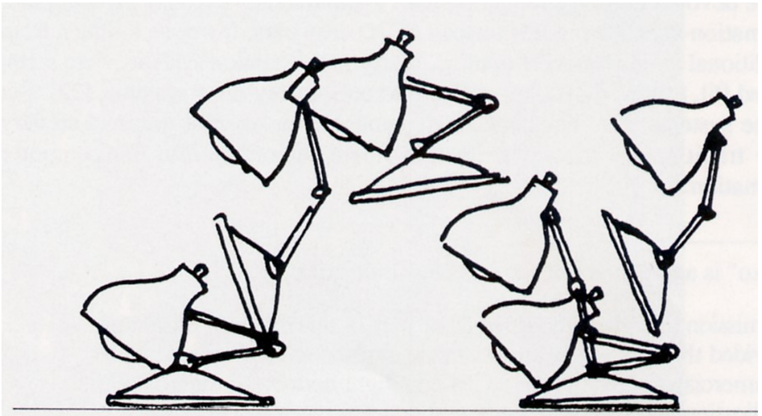
Position and Orientation

- Objects can move!
- Keys:
  - Separate control of position and orientation
  - Never interpolate matrices!
    - They won't do what you want.
  - *Quaternions* interpolate better than Euler angles $[\hat{a}_x \sin(\theta/2),\ \hat{a}_y \sin(\theta/2),\ \hat{a}_z \sin(\theta/2),\ \cos(\theta/2)]$
    - But angles make a better animation interface
    - Can still convert to quaternion for interpolation
    - Possible to use directly for rotation, or convert to matrix

Squash and Stretch

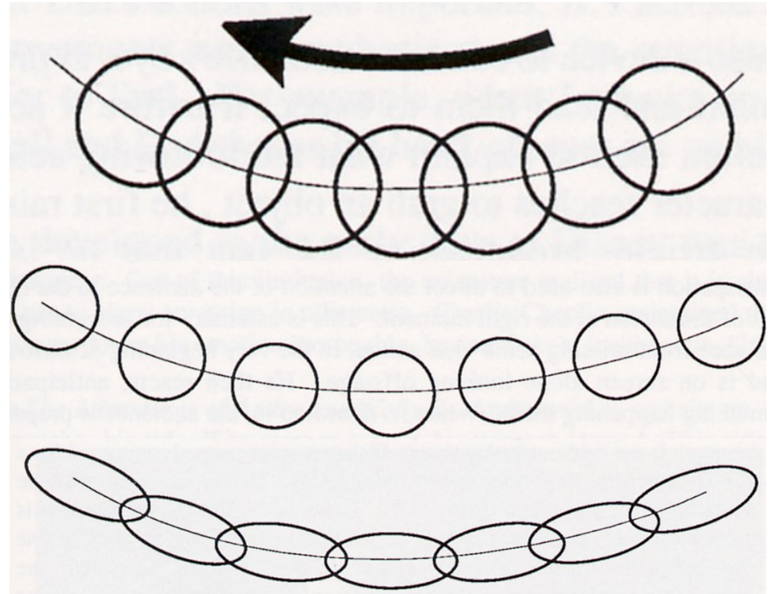
- Defining the rigidity and mass of an object by distorting its shape during an action
- Examples:
  - Ball flattening during bounce
  - Facial animation – cheeks squash during smile





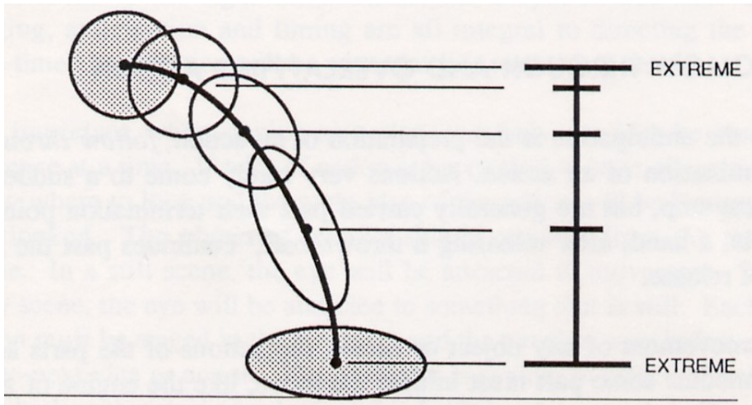
Squash and Stretch

- Keys
  - Volume constant
  - Different materials respond differently
  - Need not deform
  - Use stretching to eliminate strobing from fast action
- Method
  - Can use scale to conserve volume (up in one dimension down in others)



Slow In and Out

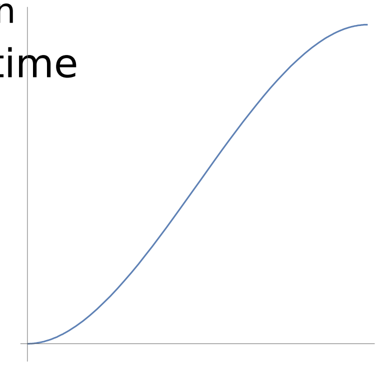
- The spacing of the in between frames to achieve subtlety of timing and movement
- Example:
  - Moving from place to place: start and end slow



Slow In and Out

- Keys
  - Think about continuity of second and third order motion
- Reparameterize time

$$t_{new} = 3\,t^2 - 2\,t^3$$



Arcs

- The visual path of action for natural movement
- Examples:
  - Thrown ball
- Keys
  - Arc movements are more natural than lines

Character Animation

- Control
  - Hierarchical model
  - Forward kinematics
  - Inverse kinematics
  - Motion capture
- Rendering
  - Skinning
  - Blend Shapes
  - Deformation

Forward Kinematics

- Given a set of joint angles, where's the hand?
  - (or foot or head or …)
  - *End effector*
- Just apply nested transforms
- We know how to do that!

Forward Kinematics

- Character is holding something in their right hand, want to shift it to the left hand
  - Forward transform up tree
  - Inverse transform back down
- Think of matrices as X_from_Y
  - X_from_Y * Y_from_Z = X_from_Z
  - $X\_from\_Y^{-1}$ = Y_from_X

Inverse Kinematics

- Find angles to match end effector position
- Few joints: system of equations
- Many joints: optimization
  - Often with constraints
    - (wrist doesn't bend that way)
  - And heuristics
    - Minimal change
    - Load support
    - Physical data

Motion Capture (mocap)

- Track markers on actor
- Infer transforms
- Often significant artistic cleanup

Skinning

- Don't like intersecting joints
- Animate "skeleton"
  - Just joint transforms, no geometry
- Each vertex in "skin"
  - Linear blend of one or more joint transforms
  - E.g. α Shoulder + β Arm
- Can *retarget* same animation to different skins

Blend Shapes

- Sculpted vertex positions in key *poses*
- Blend positions
- Good when skeletons don't work well
- Most often used for facial animation

Deformation

- Nonlinear function p' = f(p)
- Affine transform as a function of position
  - Bend = RotateX(z), twist = RotateZ(z)
- Free form deformation (FFD)
  - 3D spline: p(s,t,u)
  - Like object is embedded in jello

Physics-based Animation

- Generally: simulating the laws of physics to predict motion
- Common applications:
  - Fluids, gas
  - Cloth, hair
  - Rigid body motion
- Approach: model change as differential equations

Autonomous Objects/Groups

- Generally: create complex group behavior by defining relatively simple individual behavior
- Common applications:
  - Flocks, crowds
  - Particle systems
- Approach: leverage AI techniques