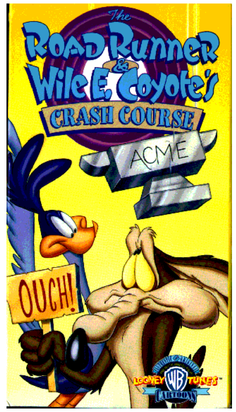
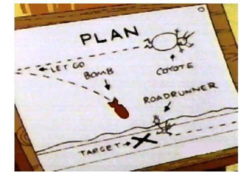
Computer Animation Fundamentals

Animation Methods
Keyframing
Interpolation
Kinematics
Inverse Kinematics



Conventional Animation

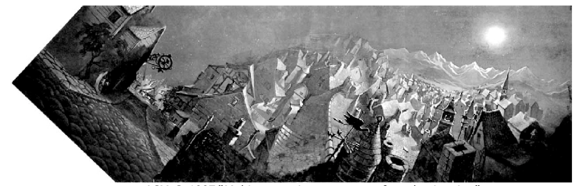


Draw each frame of the animation

- great control
- tedious

Reduce burden with cel animation

- layer
- keyframe
- inbetween
- cel panoramas (Disney's Pinocchio)
- ...

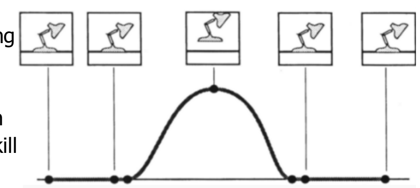


ACM © 1997 "Multiperspective panoramas for cel animation"

Computer-Assisted Animation

Keyframing

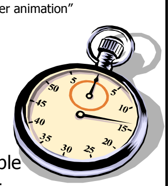
- automate the inbetweening
- good control
- less tedious
- creating a good animation still requires considerable skill and talent



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

Procedural animation

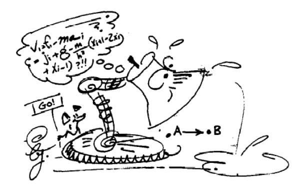
- describes the motion algorithmically
- express animation as a function of small number of parameteres
- Example: a clock with second, minute and hour hands
  - hands should rotate together
  - express the clock motions in terms of a "seconds" variable
  - the clock is animated by varying the seconds parameter



Computer-Assisted Animation

Physically Based Animation

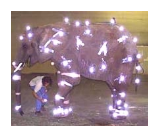
- Assign physical properties to objects (masses, forces, inertial properties)
- Simulate physics by solving equations
- Realistic but difficult to control

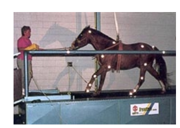


ACM© 1988 "Spacetime Constraints"

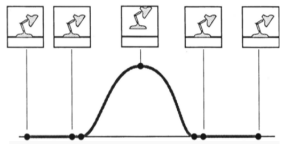
Motion Capture

- Captures style, subtle nuances and realism
- You must observe someone do something






Keyframing



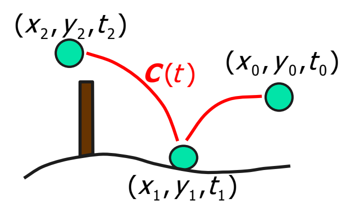
ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

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

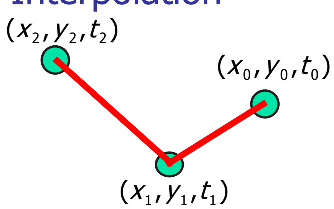
Interpolating Positions

Given positions: $(x_i, y_i, t_i),\ i = 0, \ldots, n$

find curve $\boldsymbol{C}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ such that $\boldsymbol{C}(t_i) = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$



Linear Interpolation

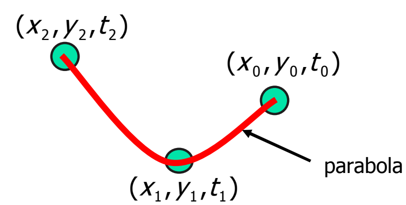


Simple problem: linear interpolation between first two points assuming t_0=0 and t_1=1: $x(t) = x_0(1-t) + x_1 t$

The x-coordinate for the complete curve in the figure:

$$x(t) = \begin{cases} \frac{t_1 - t}{t_1 - t_0} x_0 + \frac{t - t_0}{t_1 - t_0} x_1, & t \in [t_0, t_1) \\ \frac{t_2 - t}{t_2 - t_1} x_1 + \frac{t - t_1}{t_2 - t_1} x_2, & t \in [t_1, t_2] \end{cases}$$

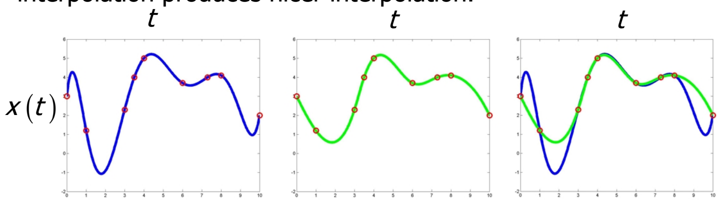
Polynomial Interpolation



An n-degree polynomial can interpolate any n+1 points. The Lagrange formula gives the n+1 coefficients of an n-degree polynomial that interpolates n+1 points. The resulting interpolating polynomials are called Lagrange polynomials. On the previous slide, we saw the Lagrange formula for n = 1.

Spline Interpolation

Lagrange polynomials of small degree are fine but high degree polynomials are too wiggly. Spline (piecewise cubic polynomial) interpolation produces nicer interpolation.



8-degree polynomial | spline | spline vs. polynomial

How many n-degree polynomials interpolate n+1 points?

How many splines interpolate n+1 points?

Spline Interpolation

A cubic polynomial between each pair of points:

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

Four parameters (degrees of freedom) for each spline segment.

Number of parameters:

n+1 points $\Rightarrow$ n cubic polynomials $\Rightarrow$ 4n degrees of freedom

Number of constraints:

- interpolation constraints

  n+1 points $\Rightarrow$ 2 + 2 (n-1) = 2n interpolation constraints
- continuous velocity

  n+1 points $\Rightarrow$ n-1 velocity constraints (one for each interior point)
- continuous acceleration

  n+1 points $\Rightarrow$ n-1 acceleration constraints (one for each interior point)

Interpolation of Positions

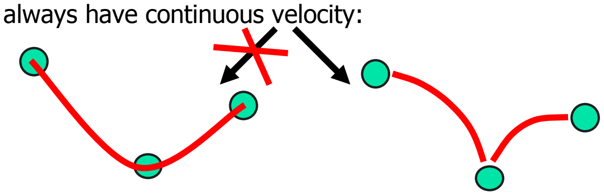
Solve an optimization to set remaining degrees of freedom:

$$\max_{\text{dof}} \ \text{quality}\left(\boldsymbol{C}(t)\right) \quad \equiv \quad \min_{\text{dof}} \ \text{badness}\left(\boldsymbol{C}(t)\right)$$

subject to constraints $\uparrow$ subject to constraints

$$\text{badness}\left(\boldsymbol{C}(t)\right) = -\text{quality}\left(\boldsymbol{C}(t)\right)$$

We want to support general constraints: not just smooth velocity and acceleration. For example, a bouncing ball does not always have continuous velocity:

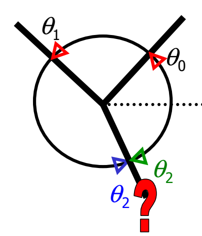


Interpolating Angles

Given angles $(\theta_i, t_i), \ i = 0, \ldots, n$

find curve $\theta(t)$ such that $\theta(t_i) = \theta_i$

Angle interpolation is ambiguous. Different angle measurements will produce different motion:



Keyframing

Given keyframes $K_i = (x_i, y_i, \theta_i, t_i),\ i = 0, \ldots, n$

find curve $\boldsymbol{K}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix}$ such that $\boldsymbol{K}(t_i) = \boldsymbol{K}_i$

Interpolate each parameter separately

Traditional Animation Principles

The in-betweening, was once a job for apprentice animators. We described the automatic interpolation techniques that accomplish these tasks automatically. However, the animator still has to draw the key frames. This is an art form and precisely why the experienced animators were spared the in-betweening work even before automatic techniques.

The classical paper on animation by John Lasseter from Pixar surveys some the standard animation techniques:

"*Principles of Traditional Animation Applied to 3D Computer Graphics,*" **SIGGRAPH'87**, pp. 35-44.

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

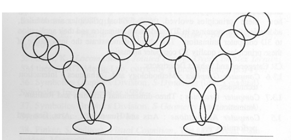


FIGURE 2. Squash & stretch in bouncing ball.

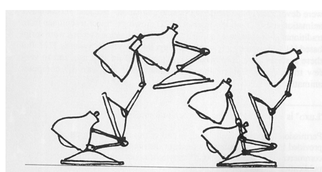


FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

Timing

Timing affects weight:

- Light object move quickly
- Heavier 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.

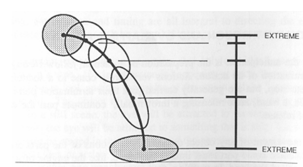


FIGURE 9. Timing chart for ball bounce.

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.

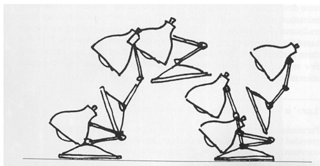
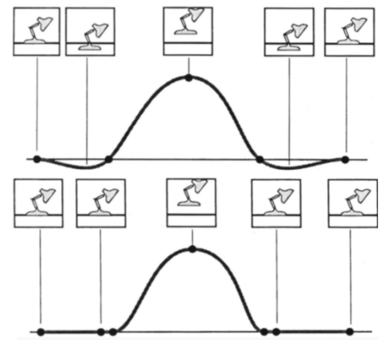


FIGURE 3. Squash & stretch in Luxo Jr.'s hop.

Interpolating Key Frames

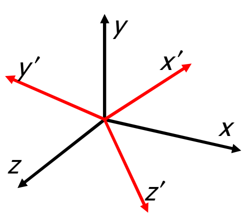
Interpolation is not fool proof. The splines may undershoot and cause interpenetration. The animator must also keep an eye out for these types of side-effects.



Interpolating Orientations in 3-D

Rotation matrices

Given rotation matrices M_i and time t_i, find $M(t)$ such that $M(t_i)=M_i$.



$$M = \begin{bmatrix} | & | & | \\ \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \\ | & | & | \end{bmatrix}$$

Flawed Solution

Linearly interpolate each entry independently

Example: M_0 is identity and M_1 is 90-deg rotation around x-axis

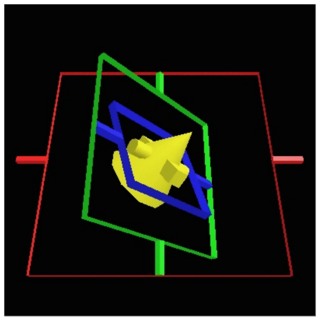
$$\text{Interpolate}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & -0.5 & 0.5 \end{bmatrix}$$

Is the result a rotation matrix?

The result R is not a rotation matrix. For example, check that RR^T does not equal identity. In short, this interpolation does not preserve the rigidity (angles and lengths) of the transformation.

Euler Angles

An euler angle is a rotation about a single axis. Any orientation can be described composing three rotation around each coordinate axis. We can visualize the action of the Euler angles: each loop is a rotation around one coordinate axis.

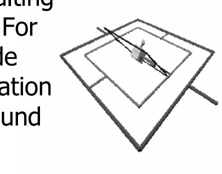


Interpolating Euler Angles

Natural orientation representation: three angles for three degrees of freedom

Unnatural interpolation: A rotation of 90-degrees first around the z-axis and then around the y-axis has the effect of a 120-degree rotation around the axis (1, 1, 1). But rotation of 30-degrees around the z- and y-axis does not have the effect of a 40-degree rotation around the axis (1, 1, 1).

Gimbal lock: two or more axis align resulting in a loss of rotation degrees of freedom. For example, if the green loop in previous slide aligns with the red loop then both the rotation around the blue loop and the rotation around the red loop produces identical rotation.



Quaternion Interpolation

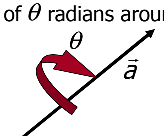
Linear int**erp**olation (lerp) of quaternion representation of orientations gives us something better:

$$\text{lerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \mathbf{q}_0(1-t) + \mathbf{q}_1 t$$

Quaternion Refresher

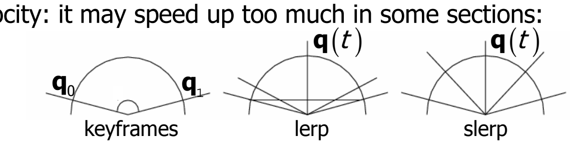
- a *general* quaternion $\mathbf{q}$ consists of four numbers: a scalar s and a 3-D vector $\vec{v} = (\vec{v}_1, \vec{v}_2, \vec{v}_3)$: $\mathbf{q} = (s, \vec{v})$
- two general quaternions are multiplied by a special rule:
  $$\mathbf{q}_1\mathbf{q}_2 = \left(s_1 s_2 - (\vec{v}_1 \cdot \vec{v}_2), s_1\vec{v}_2 + s_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2\right)$$
- a *unit* quaternion $\mathbf{q} = (\cos(\theta/2), \sin(\theta/2)\vec{a})$

can represent a rotation of θ radians around the unit axis vector $\vec{a}$



Quaternion Interpolation

The only problem with **l**inear int**erp**olation (lerp) of quaternions is that it interpolates the straight line (the secant) between the two quaternions and not their spherical distance. As a result, the interpolated motion does not have smooth velocity: it may speed up too much in some sections:



Spherical **l**inear int**erp**olation (slerp) removes this problem by interpolating along the arc lines instead of the secant lines.

$$\text{slerp}(\mathbf{q}_0, \mathbf{q}_1, t) = \mathbf{q}(t) = \frac{\mathbf{q}_0 \sin((1-t)\omega) + \mathbf{q}_1 \sin(t\omega)}{\sin(\omega)},$$

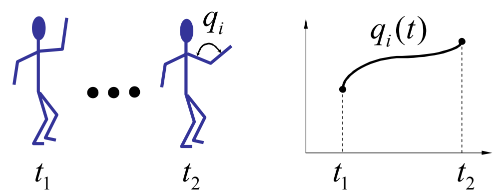
where $\omega = \cos^{-1}(\mathbf{q}_0 \cdot \mathbf{q}_1)$

Articulated Models

Articulated models:

- rigid parts
- connected by joints

They can be animated by specifying the joint angles (or other display parameters) as functions of time.

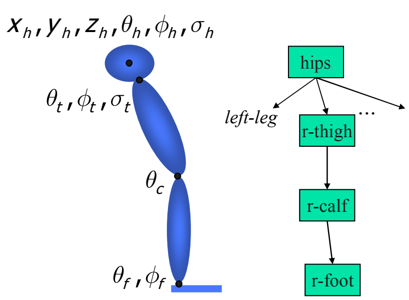


Forward Kinematics

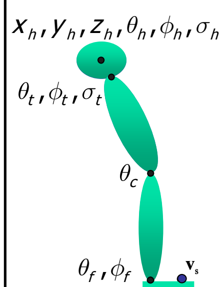
Describes the positions of the body parts as a function of the joint angles.

Skeleton Hierarchy

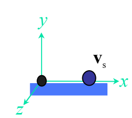
Each bone transformation described relative to the parent in the hierarchy:



Forward Kinematics



Transformation matrix for a sensor/effecter $\mathbf{v}_s$ is a matrix composition of all joint transformation between the sensor/effecter and the root of the hierarchy.



$$\mathbf{v}_w = \mathbf{T}(x_h,y_h,z_h)\mathbf{R}(\theta_h,\phi_h,\sigma_h)\ \mathbf{TR}(\theta_t,\phi_t,\sigma_t)\ \mathbf{TR}(\theta_c)\ \mathbf{TR}(\theta_f,\phi_f)\ \mathbf{v}_s$$

$$v_w = \mathbf{S}\left(\underbrace{x_h,y_h,z_h,\theta_h,\phi_h,\sigma_h,\theta_t,\phi_t,\sigma_t,\theta_c,\theta_f,\phi_f}_{\mathbf{p}}\right) v_s = \mathbf{S}(\mathbf{p})\, v_s$$

Inverse Kinematics

Forward Kinematics

- Given the skeleton parameters (position of the root and the joint angles) $\boldsymbol{p}$ and the position of the sensor/effecter in local coordinates v_s, what is the position of the sensor in the world coordinates v_w.
- Not too hard, we can solve it by evaluating $\mathbf{S}(\mathbf{p})v_s$

Inverse Kinematics

- Given the the position of the sensor/effecter in local coordinates v_s and the position of the sensor in the world coordinates v_w, what are the skeleton parameters $\boldsymbol{p}$.
- Much harder requires solving the inverse of

the non-linear function $\mathbf{S}(\mathbf{p})$

- We can solve it by root-finding $\mathbf{p}$? such that $\mathbf{S}(\mathbf{p})v_s - \mathrm{v_w} = 0$
- We can solve it by optimization $\underset{\mathbf{p}}{\text{minimize}} \left(\mathbf{S}(\mathbf{p})v_s - \mathrm{v_w}\right)^2$

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.

Next Time

Dynamics

ACM© 1988 "Spacetime Constraints"