Overview

- Kinematics
  - Considers only motion
  - Determined by positions, velocities, accelerations
- Dynamics
  - Considers underlying forces
  - Compute motion from initial conditions and physics

Example: 2-Link Structure

- Two links connected by rotational joints

  \[ X = (x,y) \]

  \[ l_1 \]

  \[ l_2 \]

  \[ (0,0) \]

  \[ \theta_1 \]

  \[ \theta_2 \]

  \[ \text{“End-Effector”} \]

Forward Kinematics

- Animator specifies joint angles: \( \theta_1 \) and \( \theta_2 \)
- Computer finds positions of end-effector: \( X \)

\[ X = (l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2), l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2)) \]

Forward Kinematics

- Joint motions can be specified by spline curves

Forward Kinematics

- Joint motions can be specified by initial conditions and velocities

\[ \theta_1(0) = 60^\circ \quad \theta_2(0) = 250^\circ \]

\[ \frac{d\theta_1}{dt} = 1.2 \quad \frac{d\theta_2}{dt} = -0.1 \]
Example: 2-Link Structure

- What if animator knows position of "end-effector"?

X = (x,y)

\[ \begin{align*}
\theta_1 & \quad \text{Joint angle 1} \\
\theta_2 & \quad \text{Joint angle 2}
\end{align*} \]

End-Effector

Inverse Kinematics

- Animator specifies end-effector positions: X
- Computer finds joint angles: \( \theta_1 \) and \( \theta_2 \):

\[
\begin{align*}
\theta_1 &= -\frac{(l_2 \sin(\theta_2))x + (l_1 + l_2 \cos(\theta_2))y}{(l_2 \sin(\theta_2))y + (l_1 + l_2 \cos(\theta_2))x} \\
\theta_2 &= \cos^{-1} \left( \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right)
\end{align*}
\]

Inverse Kinematics

- End-effector positions specified by spline curves

Inverse Kinematics

- Problem for more complex structures
  - System of equations is usually under-defined
  - Multiple solutions

Inverse Kinematics

- Solution for more complex structures:
  - Find best solution (e.g., minimize energy in motion)
  - Non-linear optimization

Inverse Kinematics

- Style-based IK: optimize for learned style

Growchow 04
Summary of Kinematics

- **Forward kinematics**
  - Specify conditions (joint angles)
  - Compute positions of end-effectors
- **Inverse kinematics**
  - "Goal-directed" motion
  - Specify goal positions of end effectors
  - Compute conditions required to achieve goals

Inverse kinematics provides easier specification for many animation tasks, but it is computationally more difficult.

Overview

- **Kinematics**
  - Considers only motion
  - Determined by positions, velocities, accelerations
- **Dynamics**
  - Considers underlying forces
  - Compute motion from initial conditions and physics
  - Active dynamics: objects have muscles or motors
  - Passive dynamics: external forces only

Spacetime Constraints

- **Computer finds the “best” physical motion satisfying constraints**
- **Example: particle with jet propulsion**
  - $x(t)$ is position of particle at time $t$
  - $f(t)$ is force of jet propulsion at time $t$
  - Particle’s equation of motion is:
    $$mx'' - f - mg = 0$$
  - Suppose we want to move from $a$ to $b$ within $t_0$ to $t_1$ with minimum jet fuel:
    Minimize $\int_{t_0}^{t_1} |f(t)|^2 dt$ subject to $x(t_0) = a$ and $x(t_1) = b$

Spacetime Constraints

- **Discretize time steps:**
  \[
  x_i' = \frac{x_i - x_{i-1}}{h} \\
  x_i'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} \\
  m\left(x_i'' - \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}\right) - f_i - mg = 0
  \]
  Minimize $h \sum |f_i|$ subject to $x_0 = a$ and $x_f = b$
Spacetime Constraints

- Solve with iterative optimization methods

Witkin & Kass '88

Advantages:
- Free animator from having to specify details of physically realistic motion with spline curves
- Easy to vary motions due to new parameters and/or new constraints

Challenges:
- Specifying constraints and objective functions
- Avoiding local minima during optimization

Adapting motion:

Original Jump

Heavier Base

Witkin & Kass '88

Adapting motion:

Hurdle

Witkin & Kass '88

Motion Sketching

- Plausible motion matches sketched constraints

Pepovic 03

Adapting motion:

Ski Jump

Witkin & Kass '88
**Spacetime Constraints**

**Advantages:**
- Free animator from having to specify details of physically realistic motion with spline curves
- Easy to vary motions due to new parameters and/or new constraints

**Challenges:**
- Specifying constraints and objective functions
- Avoiding local minima during optimization

**Passive Dynamics**

**Other physical simulations:**
- Rigid bodies
- Soft bodies
- Cloth
- Liquids
- Gases
- etc.

**Other physical simulations:**
- Hot Gases (Foster & Metaxas `97)
- Cloth (Baraff & Witkin `98)

**Particle Systems**

**A particle is a point mass**
- Mass
- Position
- Velocity
- Acceleration
- Color
- Lifetime

**Use lots of particles to model complex phenomena**
- Keep array of particles

**Creating/Deleting Particles**

**Where to create particles?**
- Around some center
- Along some path
- Surface of shape
- Where particle density is low

**When to delete particles?**
- Where particle density is high
- Life span
- Random

**Example: Wrath of Khan**

![Diagram of particle system on the planet's surface]
Equations of Motion

- Newton’s Law for a point mass
  \( f = ma \)

- Update every particle for each time step
  \( a(t+\Delta t) = g \)
  \( v(t+\Delta t) = v(t) + a(t) \Delta t \)
  \( p(t+\Delta t) = p(t) + v(t) \Delta t + a(t)^2 \Delta t/2 \)

Solving the Equations of Motion

- Initial value problem
  - Know \( p(0), v(0), a(0) \)
  - Can compute force at any time and position
  - Compute \( p(t) \) by forward integration

- Euler integration
  \( p(t+\Delta t) = p(t) + \Delta t f(x,t) \)

- Problem:
  - Accuracy decreases as \( \Delta t \) gets bigger
Solving the Equations of Motion

- Midpoint method (2nd order Runge-Kutta)
  - Compute an Euler step
  - Evaluate $f$ at the midpoint
  - Take an Euler step using midpoint force
    $$p(t+\Delta t) = p(t) + \Delta t f(p(t) + 0.5\Delta t f(t), t)$$

Adapting step size

- Compute $p_a$ by taking one step of size $h$
- Compute $p_b$ by taking 2 steps of size $h/2$
- Error $= |p_a - p_b|$
- Adjust step size by factor $(\epsilon/\text{error})^{1/f}$

Particle System Forces

- Force fields
  - Gravity, wind, pressure
- Viscosity/damping
  - Liquids, drag
- Collisions
  - Environment
  - Other particles
- Other particles
  - Springs between neighboring particles (mesh)
  - Useful for cloth

Rendering Particles

- Volumes
  - Ray casting, etc.
- Points
  - Render as individual points
- Line segments
  - Motion blur over time

Example: Fountain

More Passive Dynamics Examples

- Spring meshes
- Level sets
- Collisions
- etc.
Example: Cloth

Example: Smoke

Example: Water

Example: Water

Example: Rigid Body Contact

Summary

• Kinematics
  o Forward kinematics
    » Animator specifies joints (hard)
    » Compute end-effectors (easy - assn 4!)
  o Inverse kinematics
    » Animator specifies end-effectors (easier)
    » Solve for joints (harder)

• Dynamics
  o Space-time constraints
    » Animator specifies structures & constraints (easiest)
    » Solve for motion (hardest)
  o Also other physical simulations