Computer Animation

Tom Funkhouser
Princeton University
COS 426, Spring 2007

Syllabus

I. Image processing
II. Modeling
III. Rendering
IV. Animation

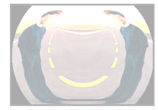
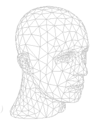
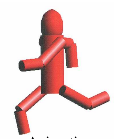


Image Processing
(Rusty Coleman, CS426, Fall99)

Rendering
(Michael Bostock, CS426, Fall99)


Modeling
(Dennis Zorin, CalTech)


Animation
(Angel, Plate 1)

Computer Animation

- What is animation?
  - Make objects change over time according to scripted actions


Pixar

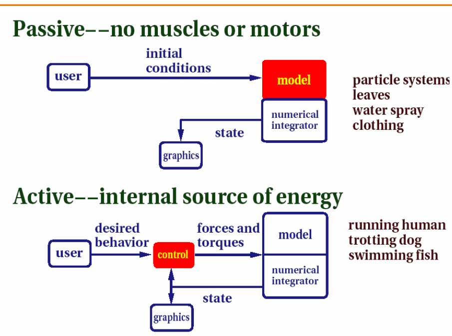
- What is simulation?
  - Predict how objects change over time according to physical laws


University of Illinois

Simulation

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

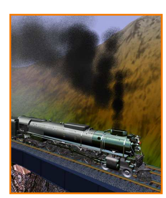
Dynamics



Hodgins

Passive Dynamics

- No muscles or motors
  - Smoke
  - Water
  - Cloth
  - Fire
  - Fireworks
  - Dice



McAllister

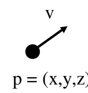
Passive Dynamics

- Physical laws
  - Newton's laws
  - Hook's law
  - Etc.
- Physical phenomena
  - Gravity
  - Momentum
  - Friction
  - Collisions
  - Elasticity
  - Fracture

McAllister

Particle Systems

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



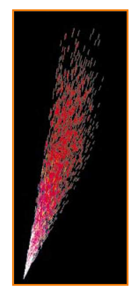
- Use lots of particles to model complex phenomena
  - Keep array of particles
  - Newton's laws

Particle Systems

- For each frame:
  - Create new particles and assign attributes
  - Delete any expired particles
  - Update particles based on attributes and physics
  - Render particles

Creating Particles

- Where to create particles?
  - Predefined source
  - Surface of shape
  - Where particle density is low
  - etc.



McAllister

Creating Particles

- Where to create particles?
  - Predefined source
  - Surface of shape
  - Where particle density is low
  - etc.

Reeves

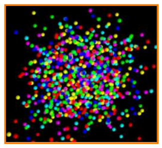
Deleting Particles

- When to delete particles?
  - Predefined sink
  - Surface of shape
  - Where density is high
  - Life span
  - Random

McAllister

Rendering Particles

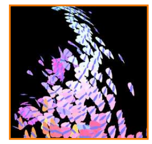
- Rendering styles
  - Points
  - Polygons
  - Shapes
  - Trails
  - etc.

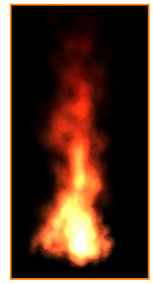


McAllister

Rendering Particles

- Rendering styles
  - Points
  - Polygons
  - Shapes
  - Trails
  - etc.





McAllister

Rendering Particles

- Rendering styles
  - Points
  - Polygons
  - Shapes
  - Trails
  - etc.

McAllister

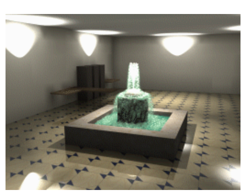
Rendering Particles

- Rendering styles
  - Points
  - Polygons
  - Shapes
  - Trails
  - etc.

McAllister

Particle Systems

- For each frame:
  - Create new particles and assign attributes
  - Delete any expired particles
  - Update particles based on attributes and physics
  - Render particles



McAllister

Equations of Motion

- Newton's Law for a point mass
  - f = ma

- Computing particle motion requires solving second-order differential equation

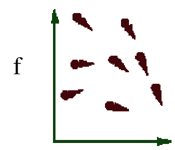
$$\ddot{x} = \frac{f(x, \dot{x}, t)}{m}$$

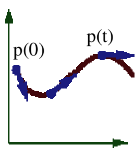
- Add variable v to form coupled first-order differential equations

$$\begin{cases} \dot{x} = v \\ \dot{v} = \dfrac{f}{m} \end{cases}$$

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

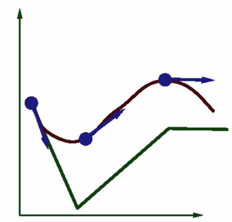




Hodgins

Solving the Equations of Motion

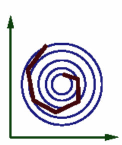
- Euler integration
  - $p(t+\Delta t)=p(t) + \Delta t\, f(x,t)$

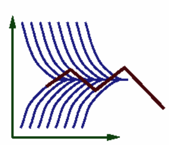


Hodgins

Solving the Equations of Motion

- Euler integration
  - $p(t+\Delta t)=p(t) + \Delta t\, f(x,t)$
- Problem:
  - Accuracy decreases as Δt gets bigger

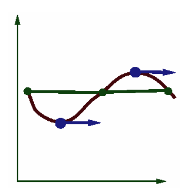




Hodgins

Solving the Equations of Motion

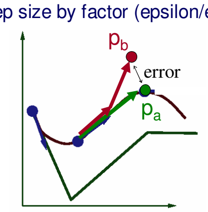
- Midpoint method (2nd order Runge-Kutta)
  - Compute an Euler step
  - Evalute f at the midpoint
  - Take an Euler step using midpoint force
    - » $p(t+\Delta t)=p(t) + \Delta t\, f(\, p(t) + 0.5^*\Delta t\, f(t),t)$



Hodgins

Solving the Equations of Motion

- 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 $(\text{epsilon}/\text{error})^{1/f}$

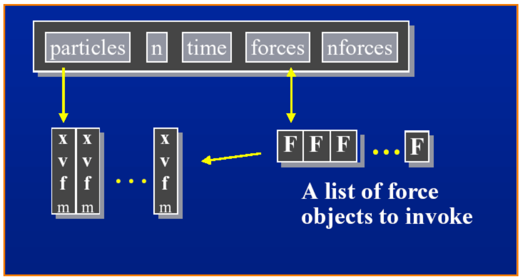


Hodgins

Particle System Forces

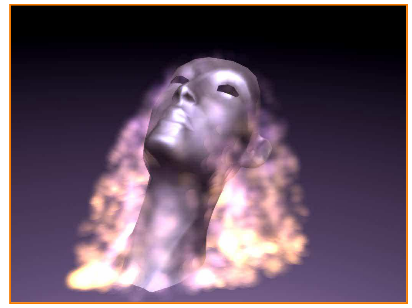
- Force fields
  - Gravity, wind, pressure
- Viscosity/damping
  - Liquids, drag
- Collisions
  - Environment
  - Other particles
- Other particles
  - Springs between neighboring particles (mesh)
  - Useful for cloth

Particle System Forces



Witkin

Example: Gravity

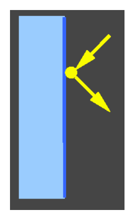


McAllister

Example: Fire

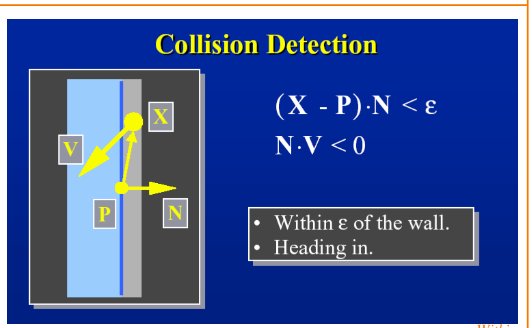
Example: Bouncing Off Wall

- Requires
  - Collision detection
  - Collision response (dynamic forces)



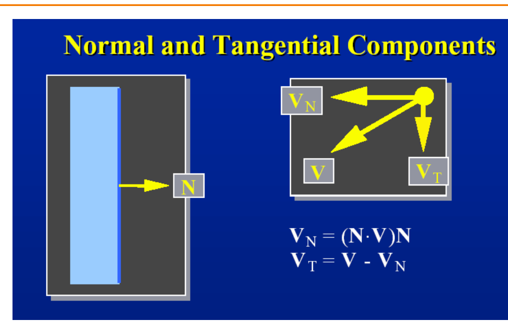
Witkin

Example: Bouncing Off Wall



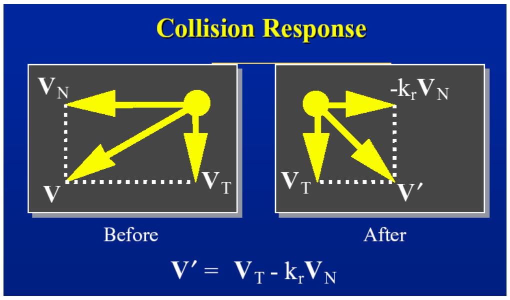
Witkin

Example: Bouncing Off Wall



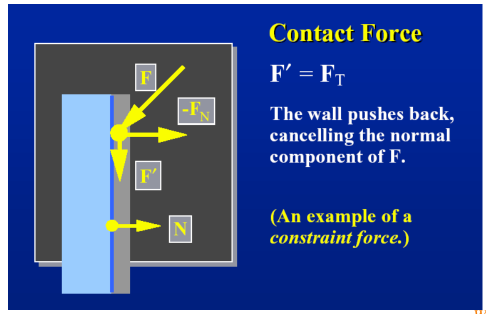
Witkin

Example: Bouncing Off Wall



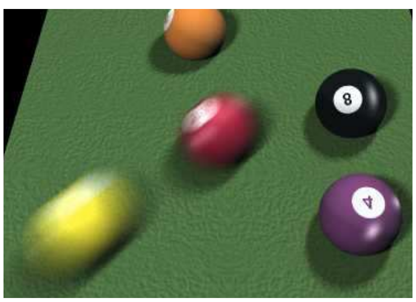
Witkin

Example: Bouncing Off Wall

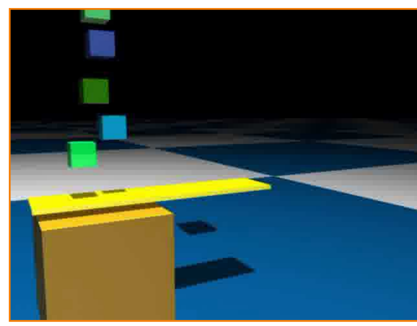


Witkin

Example: Bouncing Off Particles

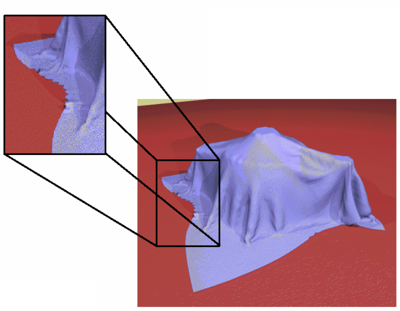


Example: More Bouncing



Bender

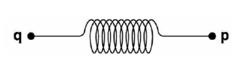
Example: Cloth



Breen

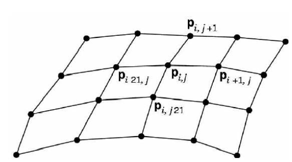
Example: Cloth

- Spring-mass mesh
- Hooke's law



$$\mathbf{f} = -k_s(|\mathbf{d}| - s)\frac{\mathbf{d}}{|\mathbf{d}|}$$

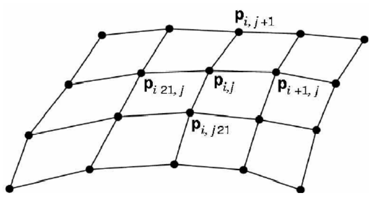
f = force
k_s = spring constant
d = p - q
s = resting length

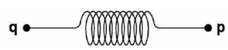


Hodgins

Example: Cloth

- Spring-mass mesh





Hodgins

Example: Cloth

- Hooke's law

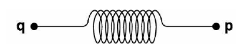
$$\mathbf{f} = -k_s(|\mathbf{d}| - s)\frac{\mathbf{d}}{|\mathbf{d}|}$$

f = force
k_s = spring constant
d = p - q
s = resting length

- Damping term

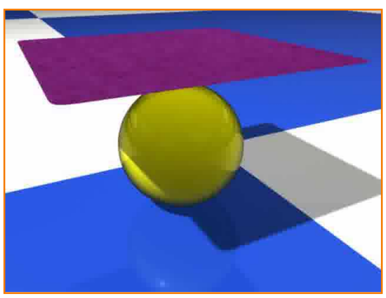
$$\mathbf{f} = -\left(k_s(|\mathbf{d}| - s) + k_d\frac{\dot{\mathbf{d}} \cdot \mathbf{d}}{|\mathbf{d}|}\right)\frac{\mathbf{d}}{|\mathbf{d}|}$$

$$\dot{\mathbf{d}} = \dot{\mathbf{p}} - \dot{\mathbf{q}}$$



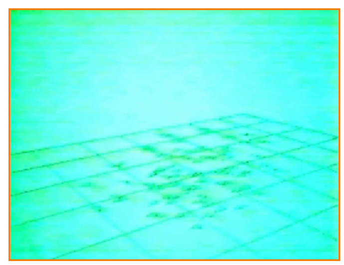
Hodgins

Example: Cloth



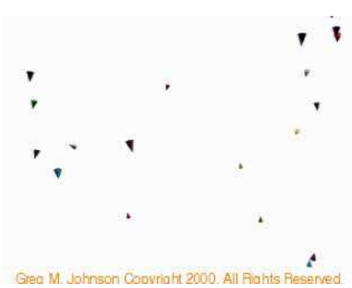
Bender

Example: Flocks & Herds



Reynolds

Example: Flocks & Herds



Summary

- Particle systems
  - Lots of particles
  - Simple physics
- Interesting behaviors
  - Waterfalls
  - Smoke
  - Cloth
  - Flocks
- Solving motion equations
  - Simplest method is Euler integration
  - Better to use adaptive step sizes