Passive Dynamics

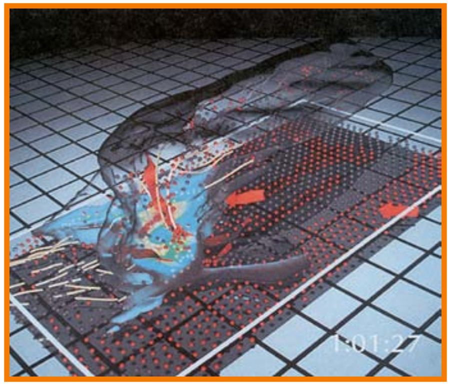
COS 426

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



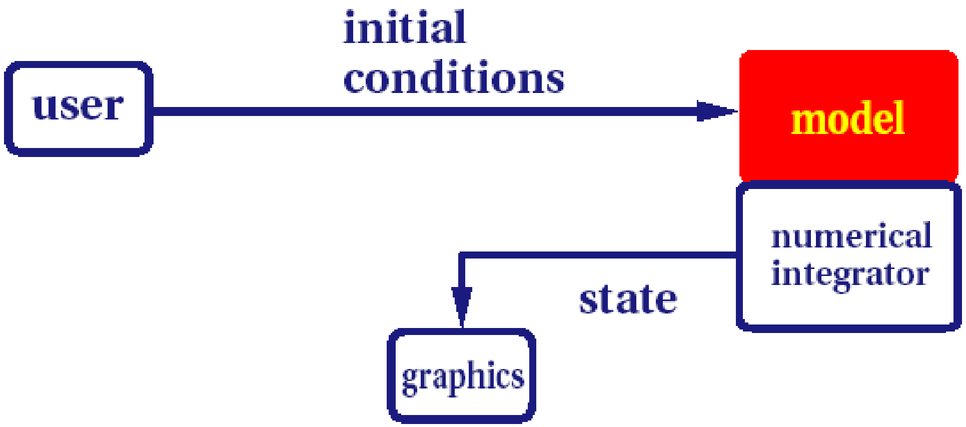
University of Illinois

Simulation

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

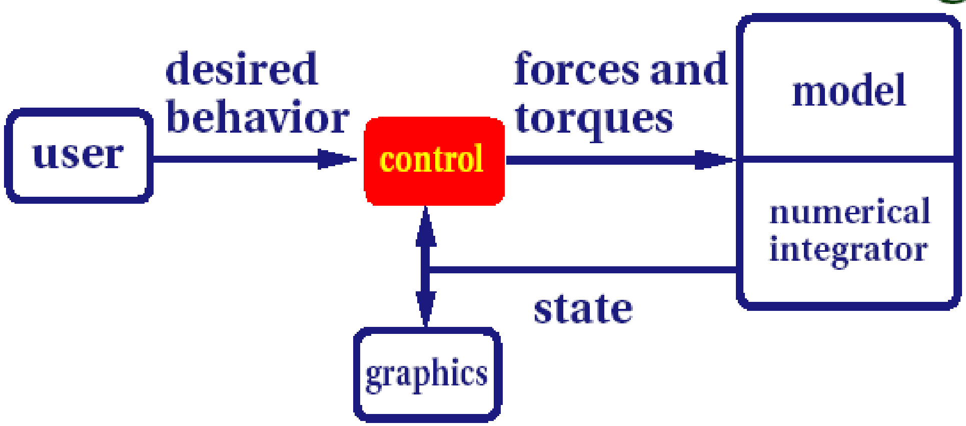
Dynamics

Passive--no muscles or motors



particle systems
leaves
water spray
clothing

Active--internal source of energy

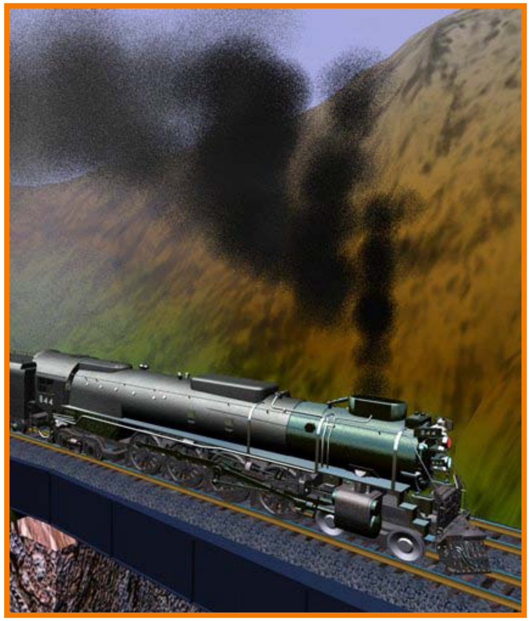


running human
trotting dog
swimming fish

Hodgins

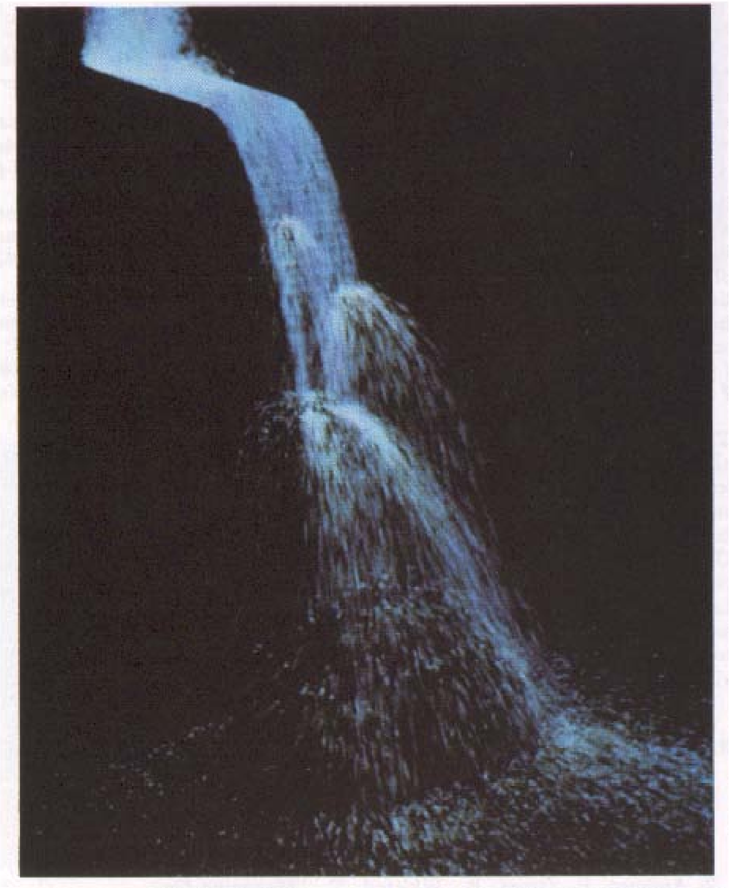
Passive Dynamics

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



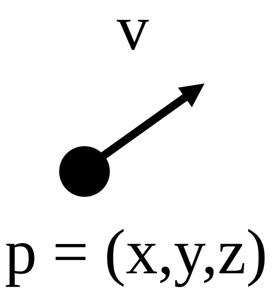
Passive Dynamics

- Physical laws
  - Newton’s laws
  - Hooke’s law
  - Etc.
- Physical phenomena
  - Gravity
  - Momentum
  - Friction
  - Collisions
  - Elasticity
  - Fracture



Particle Systems

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



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

Particle Systems

- For each frame:
  - For each simulation step (Δt)
    - Update particles based on attributes and physics
    - Delete any expired particles
    - Create new particles and assign attributes
  - Render particles

Particle Systems

- For each frame:
  - For each simulation step (Δt)
    - Update particles based on attributes and physics
    - Delete any expired particles
    - Create new particles and assign attributes
  - Render particles

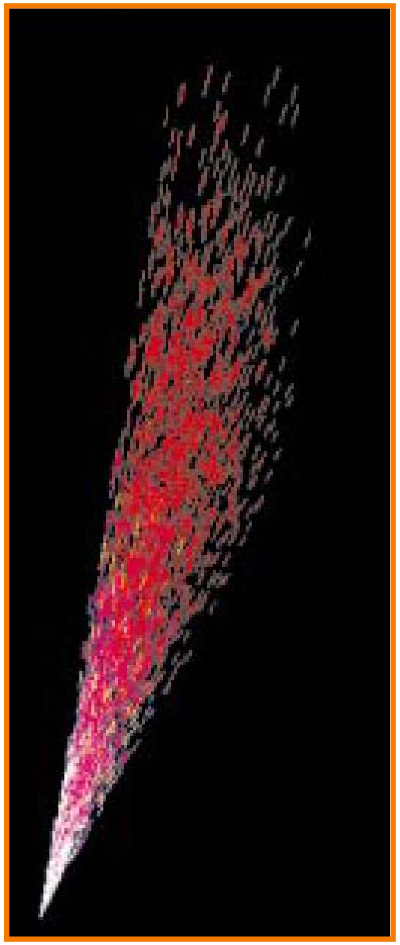
Creating Particles

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

Reeves

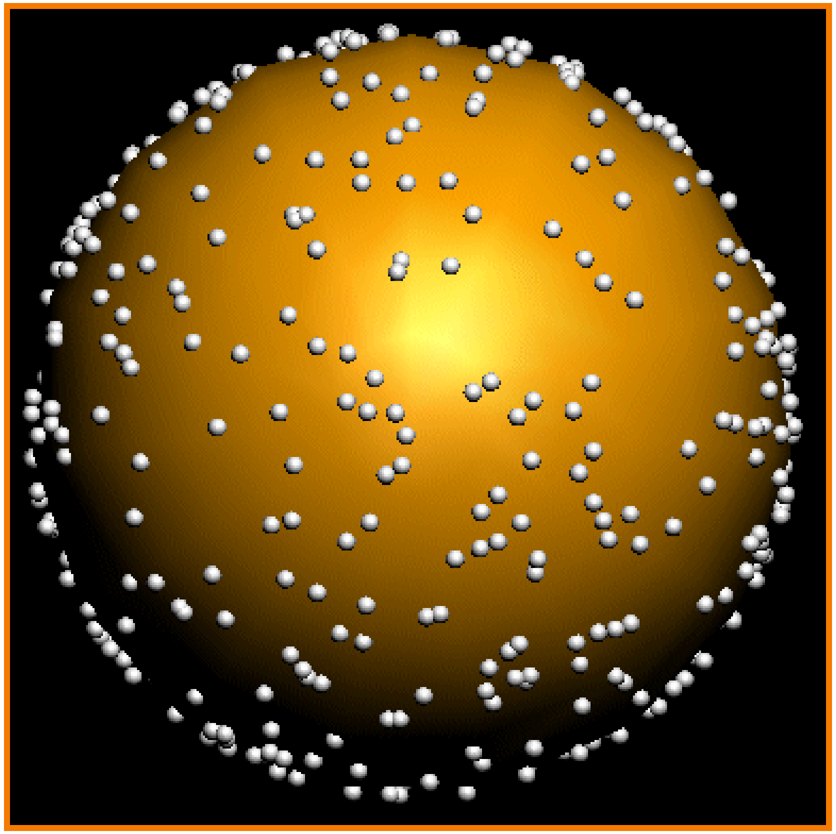
Creating Particles

- Example: particles emanating from shape
  - Line
  - Box
  - **Circle**
  - **Sphere**
  - Cylinder
  - Cone
  - Mesh



Creating Particles

- Example: particles emanating from sphere

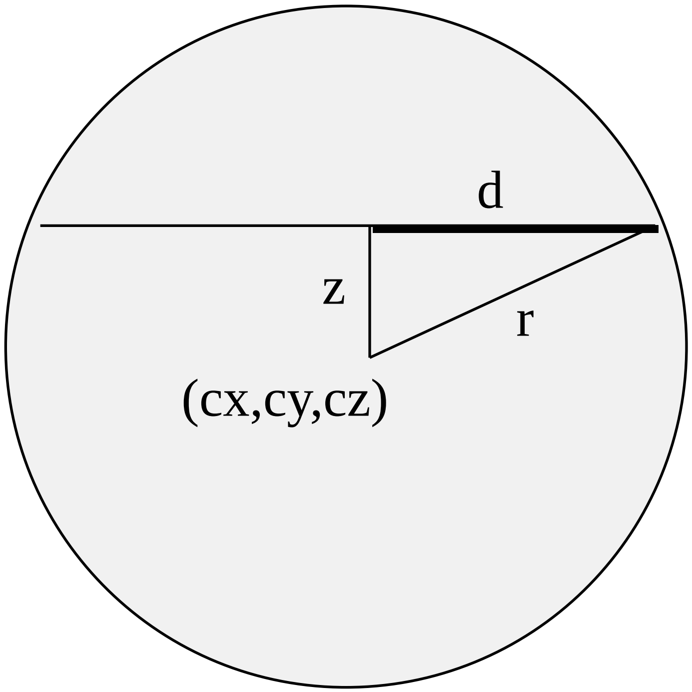


Creating Particles

- Example: particles emanating from sphere

Selecting random position on surface of sphere

1. z = random $[-1,1]$
2. t = random $[0, 2\pi)$
3. d = sqrt($r^2 - z^2$)
4. px = cx + d * cos(t)
5. py = cy + d * sin(t)
6. pz = cz + z

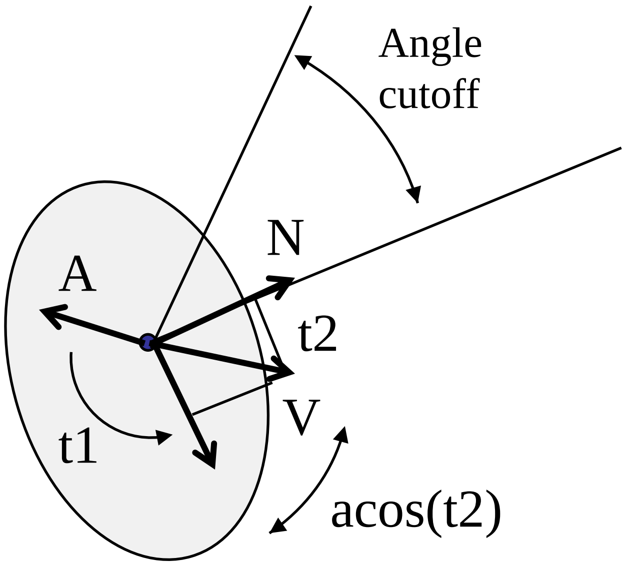


Creating Particles

- Example: particles emanating from sphere

Selecting random direction within angle cutoff of normal

1. N = surface normal
2. A = any vector on tangent plane
3. t1 = random [0, 2π)
3. t2 = random [0, sin(angle cutoff))
4. V = rotate A around N by t1
5. V = rotate V around VxN by acos(t2)

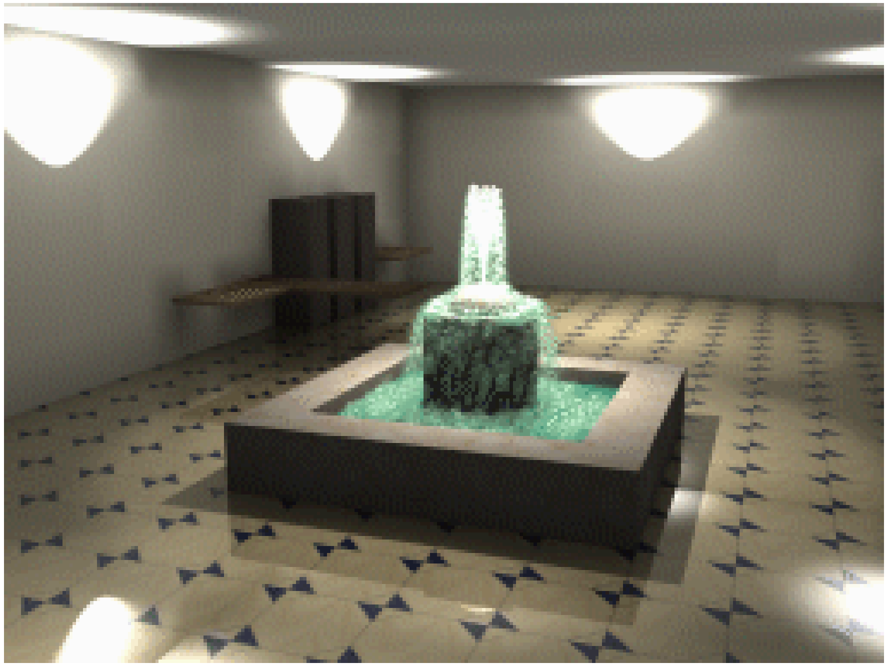


Particle Systems

- For each frame:
  - For each simulation step (Δt)
    - Update particles based on attributes and physics
    - Delete any expired particles
    - Create new particles and assign attributes
  - Render particles

Deleting Particles

- When to delete particles?
  - When life span expires
  - When intersect predefined sink surface
  - Where density is high
  - Random



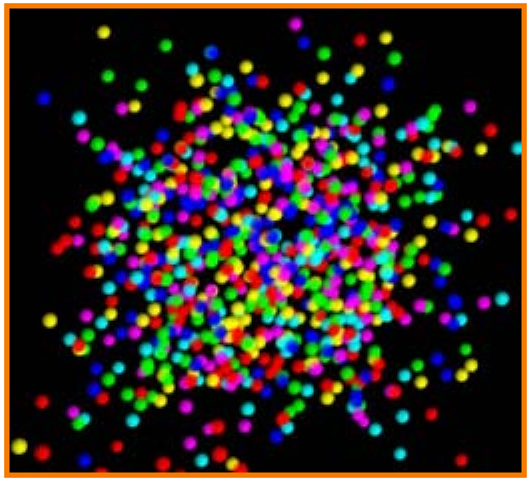
McAllister

Particle Systems

- For each frame:
  - For each simulation step (Δt)
    - Update particles based on attributes and physics
    - Delete any expired particles
    - Create new particles and assign attributes
  - Render particles

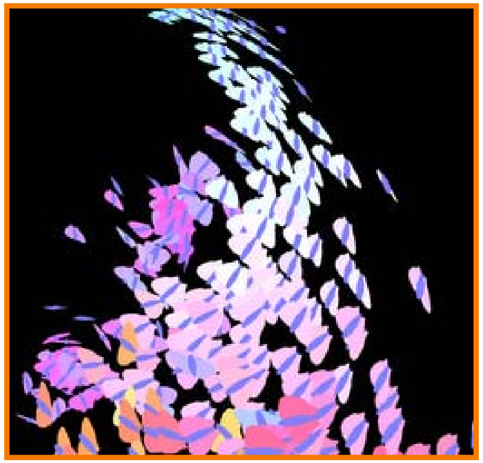
Rendering Particles

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



Rendering Particles

- Rendering styles
  - Points
  - ➢ Textured polygons
  - Shapes
  - Trails
  - etc.



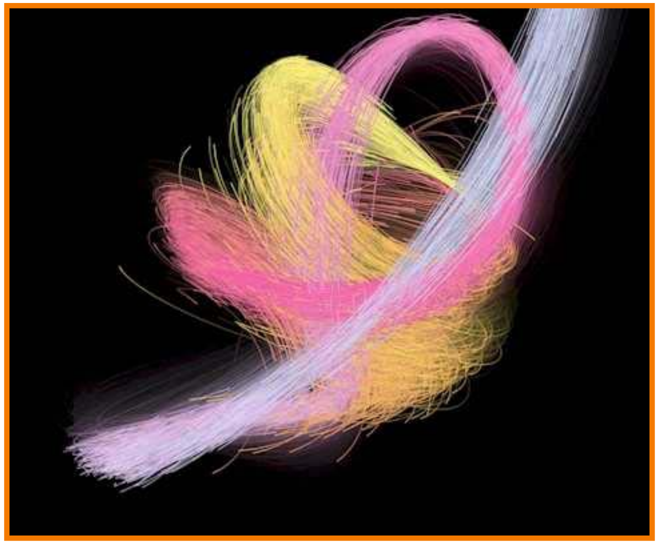
Rendering Particles

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

Rendering Particles

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

McAllister



Particle Systems

- For each frame:
  - For each simulation step (Δt)
    - Update particles based on attributes and physics
    - Delete any expired particles
    - Create new particles and assign attributes
  - Render particles

Equations of Motion

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

- Computing particle motion requires solving second-order differential equation

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

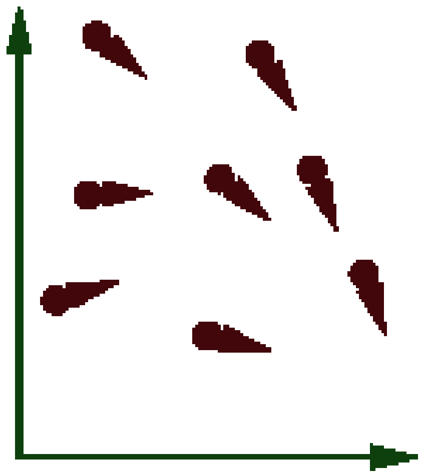
- Add variable v to form coupled first-order differential equations: "state-space form"

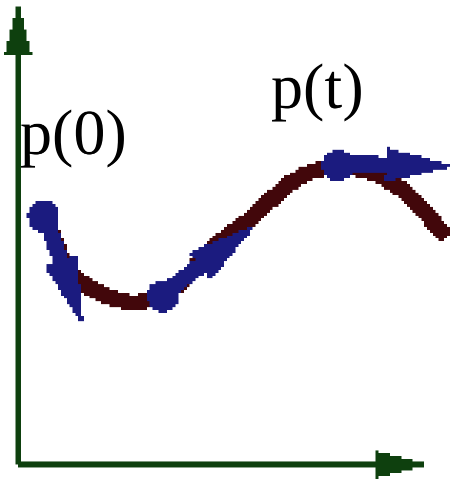
$$\begin{cases} \dot{x} = v \\ \dot{v} = \dfrac{f}{m} \end{cases}$$

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

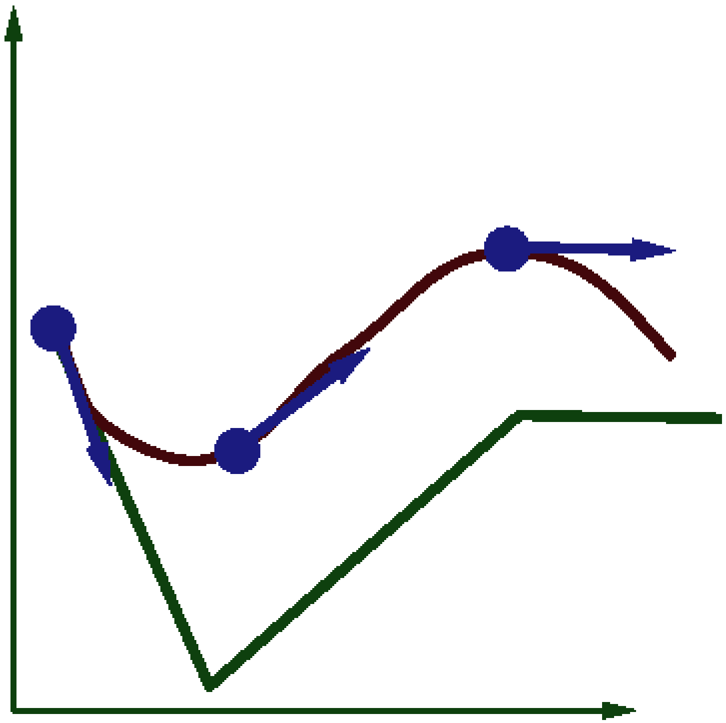
f





Solving the Equations of Motion

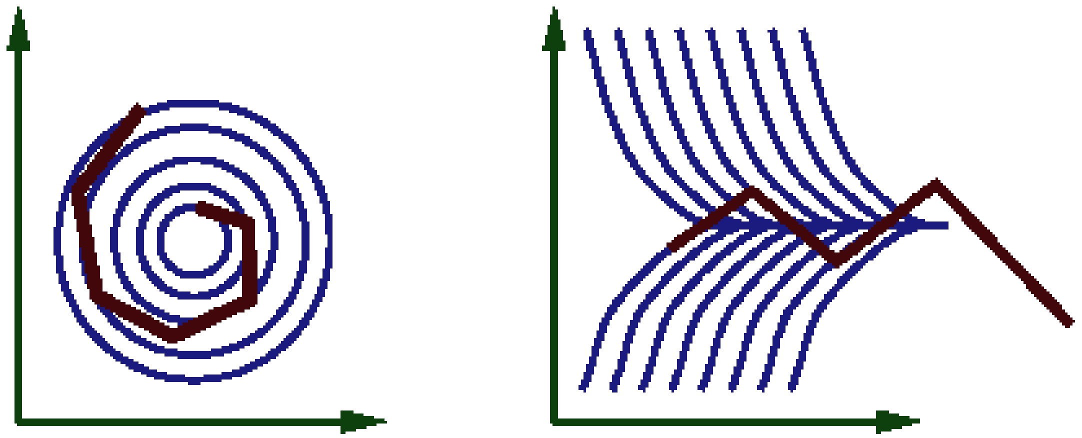
- Euler integration
  - $p(t+\Delta t)=p(t) + \Delta t\ v(t)$
  - $v(t+\Delta t)=v(t) + \Delta t\ f(p(t),t)/m$



Hodgins

Solving the Equations of Motion

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

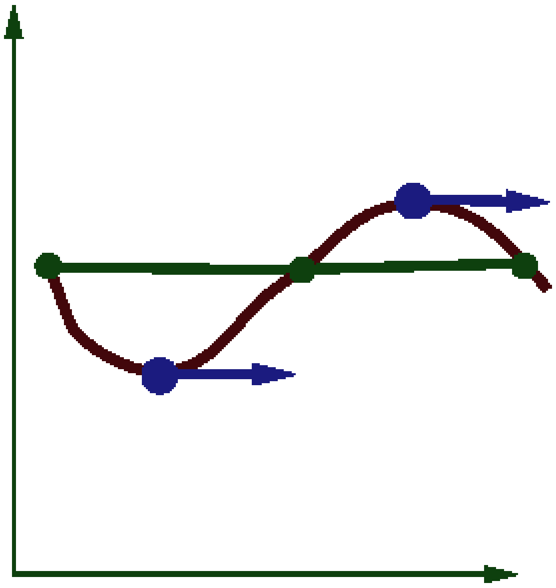


Hodgins

Solving the Equations of Motion

- Midpoint method (2nd order Runge-Kutta)
  - Compute an Euler step
  - Evaluate f at the midpoint of Euler step
  - Compute new velocity using midpoint force

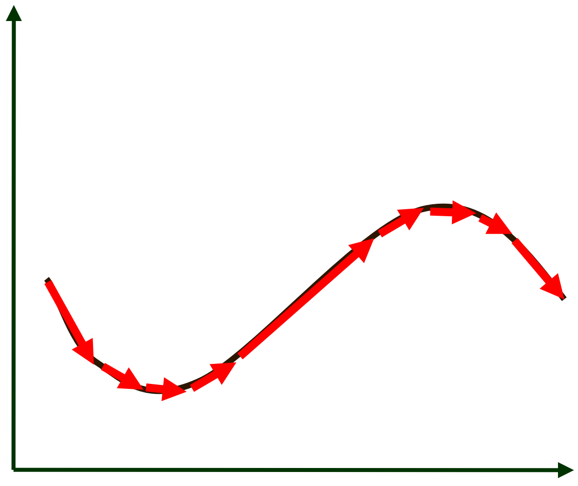
    $v(t+\Delta t) = v(t) + \Delta t\ f(\text{midpoint}) / m$

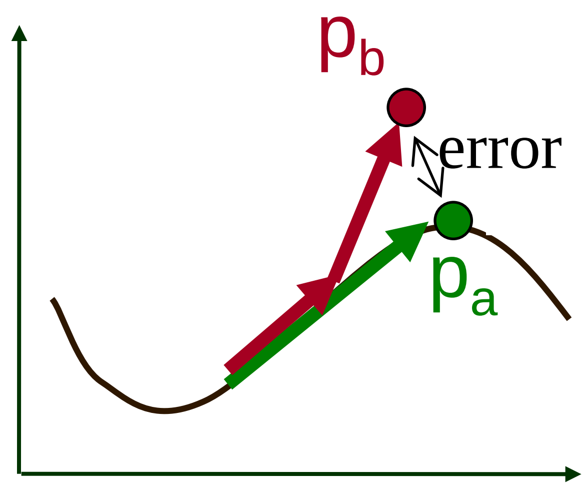


Hodgins

Solving the Equations of Motion

- Adaptive step size
  - Repeat until error is below threshold
    1. Compute p_a by taking one step of size h
    2. Compute p_b by taking 2 steps of size h/2
    3. Compute error = | p_a - p_b |
    4. If (error < threshold) break
    5. Otherwise, reduce step size





Particle System Forces

- Force fields
  - Gravity, wind, pressure
- Viscosity/damping
  - Drag, friction
- Collisions
  - Static objects in scene
  - Other particles
- Attraction and repulsion
  - Springs between neighboring particles (mesh)
  - Gravitational pull, charge

Particle System Forces

- Gravity
  - Force due to gravitational pull (of earth)
  - g = acceleration due to gravity (m/s^2)

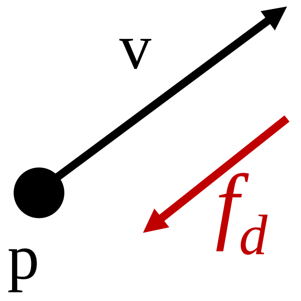
$$f_g = mg$$

g = (0, -9.80665, 0)

Particle System Forces

- Drag
  - Force due to resistance of medium
  - k_{drag} = drag coefficient (kg/s)

$$f_d = -k_{drag} v$$

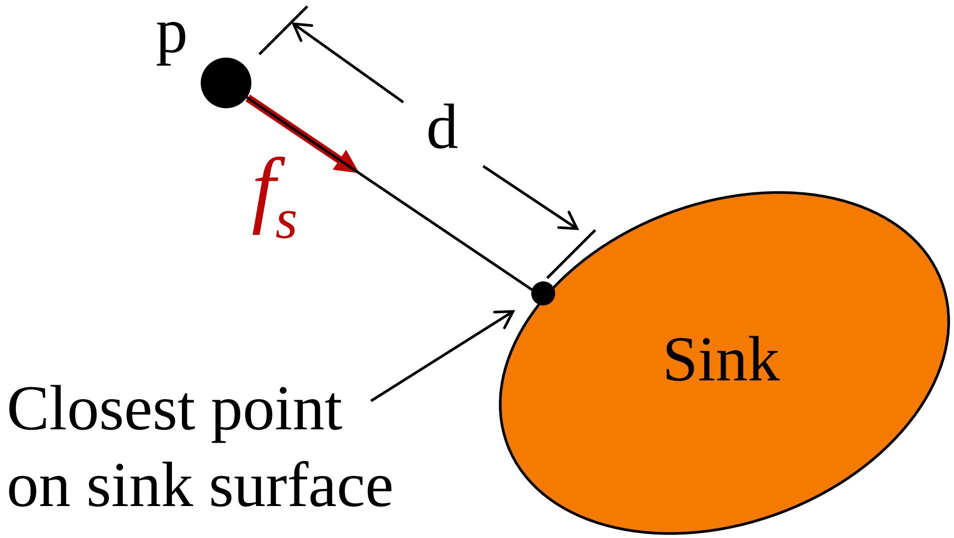


  - Air resistance sometimes taken as proportional to v^2

Particle System Forces

- Sinks
  - Force due to attractor in scene

$$f_s = \frac{\text{intensity}}{ca + la \cdot d + qa \cdot d^2}$$

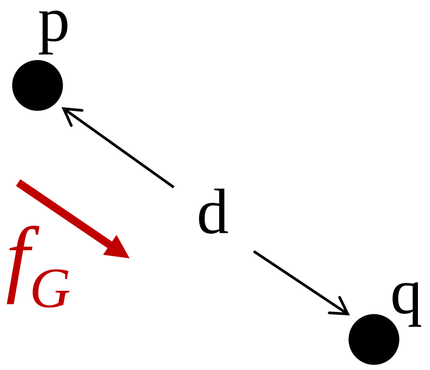


Particle System Forces

- Gravitational pull of other particles
  - Newton's universal law of gravitation

$$f_G = G \frac{m_1 \cdot m_2}{d^2}$$

$$G = 6.67428 \times 10^{-11} \ \mathrm{N\,m^2\,kg^{-2}}$$



Particle System Forces

- Springs
  - Hooke's law

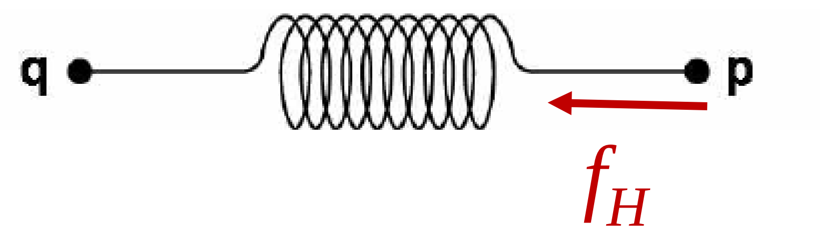
$$f_H(p) = k_s(d(p,q) - s)\ D$$

$$D = (q-p)/\|q-p\|$$

$$d(p,q) = \|q-p\|$$

$$s = \text{resting length}$$

$$k_s = \text{spring coefficient}$$



Particle System Forces

- Springs
  - Hooke's law with damping

$$f_H(p) = \left(k_s(d(p,q) - s) + k_d(v(q) - v(p) \bullet D)\right) D$$

$$D = (q - p)/\|q - p\|$$

$$d(p,q) = \|q - p\|$$

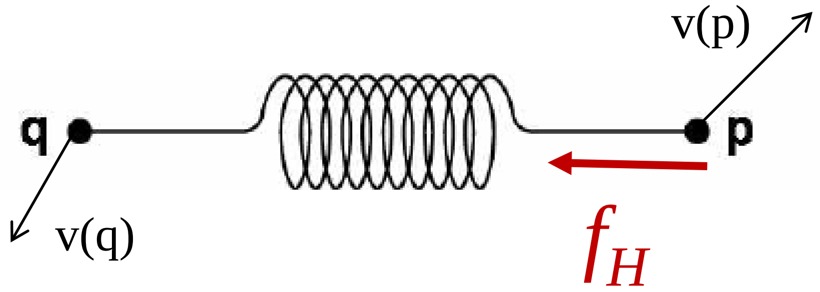
s = resting length

k_s = spring coefficient

k_d = damping coefficient

$v(p)$ = velocity of p

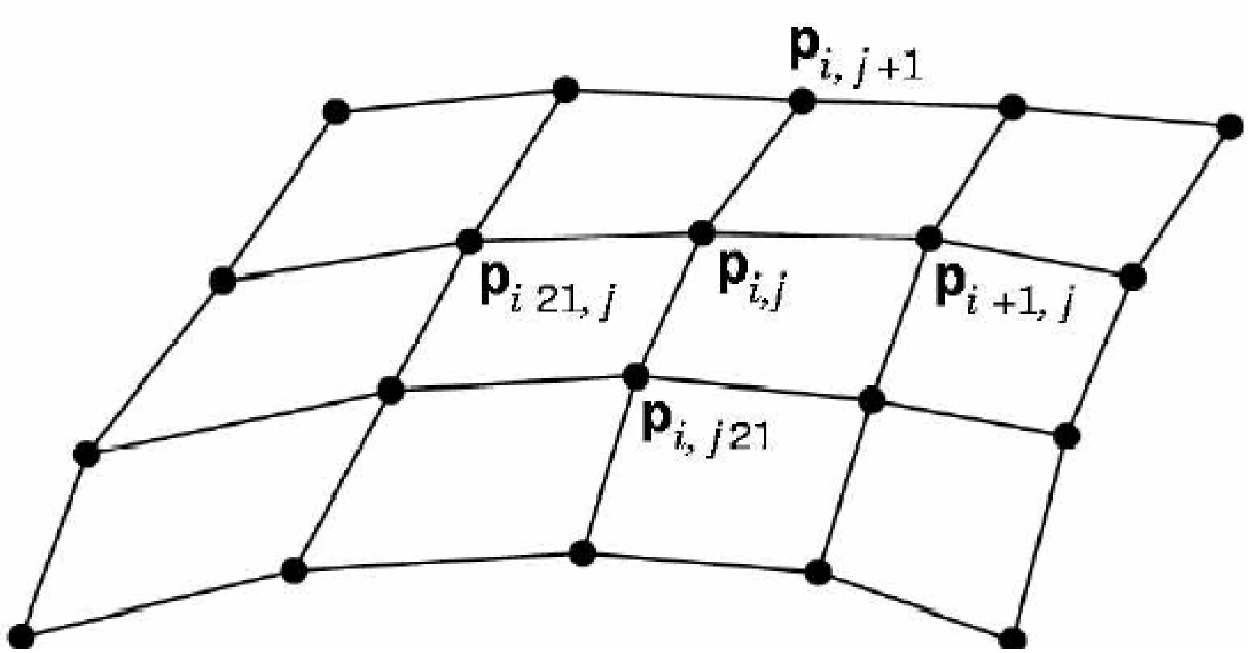
$v(q)$ = velocity of q

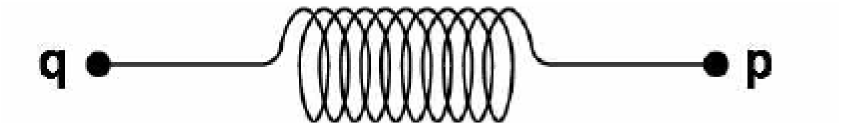


$$k_d \sim 2\sqrt{mk_s}$$

Particle System Forces

- Spring-mass mesh

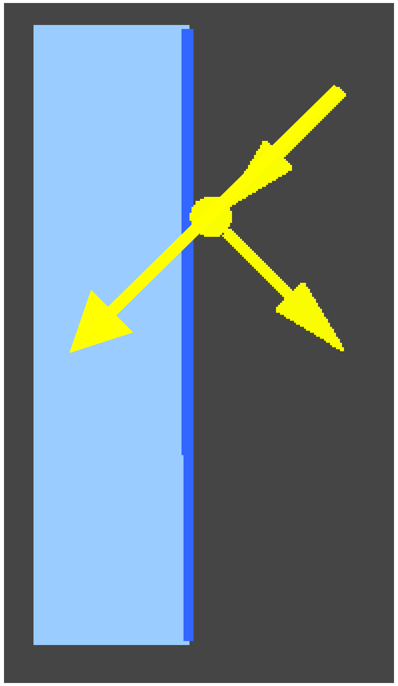




Hodgins

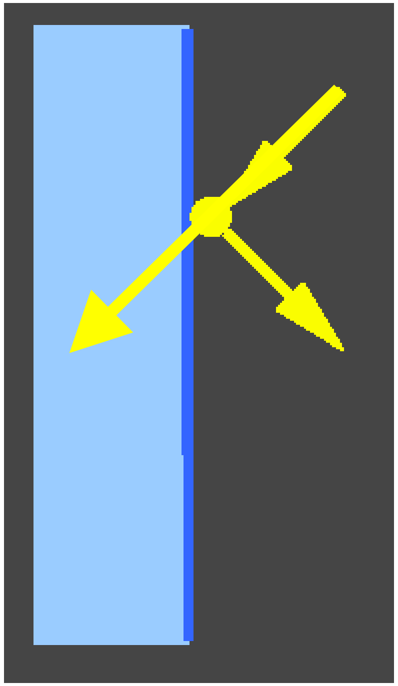
Particle System Forces

- Collisions
  - Collision detection
  - Collision response



Particle System Forces

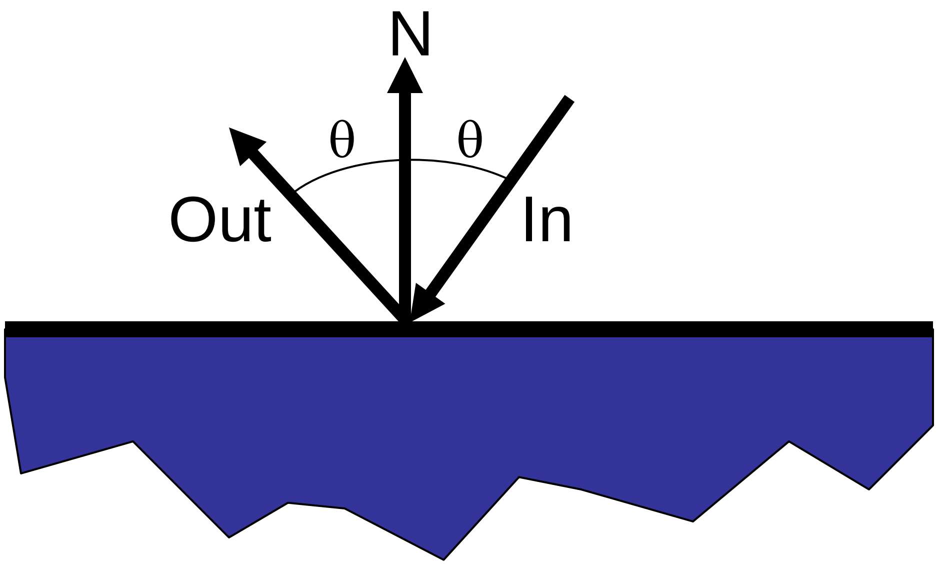
- Collision detection
  - Intersect ray with scene
  - Compute up to Δt at time of first collision, and then continue from there



Witkin

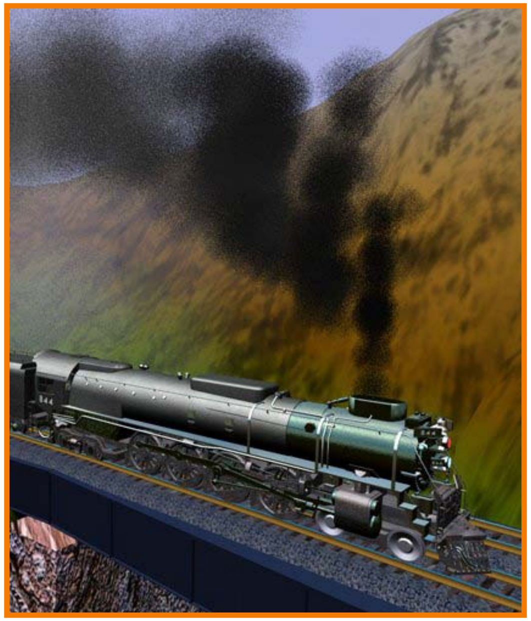
Particle System Forces

- Collision response
  - No friction = specular reflection



Passive Dynamics

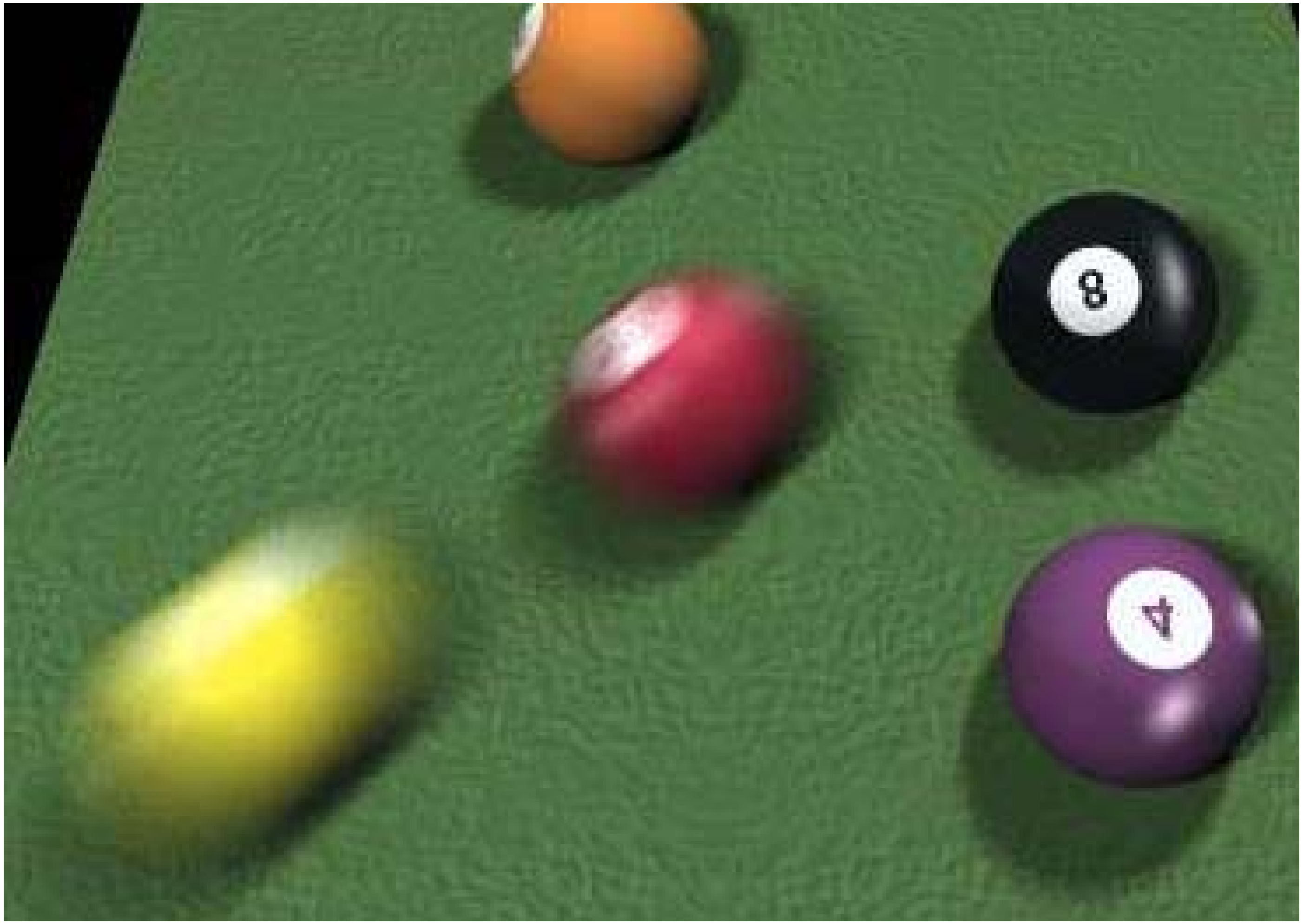
- Examples
  - Smoke
  - Water
  - Cloth
  - Fire
  - Fireworks
  - Dice



Example: Gravity

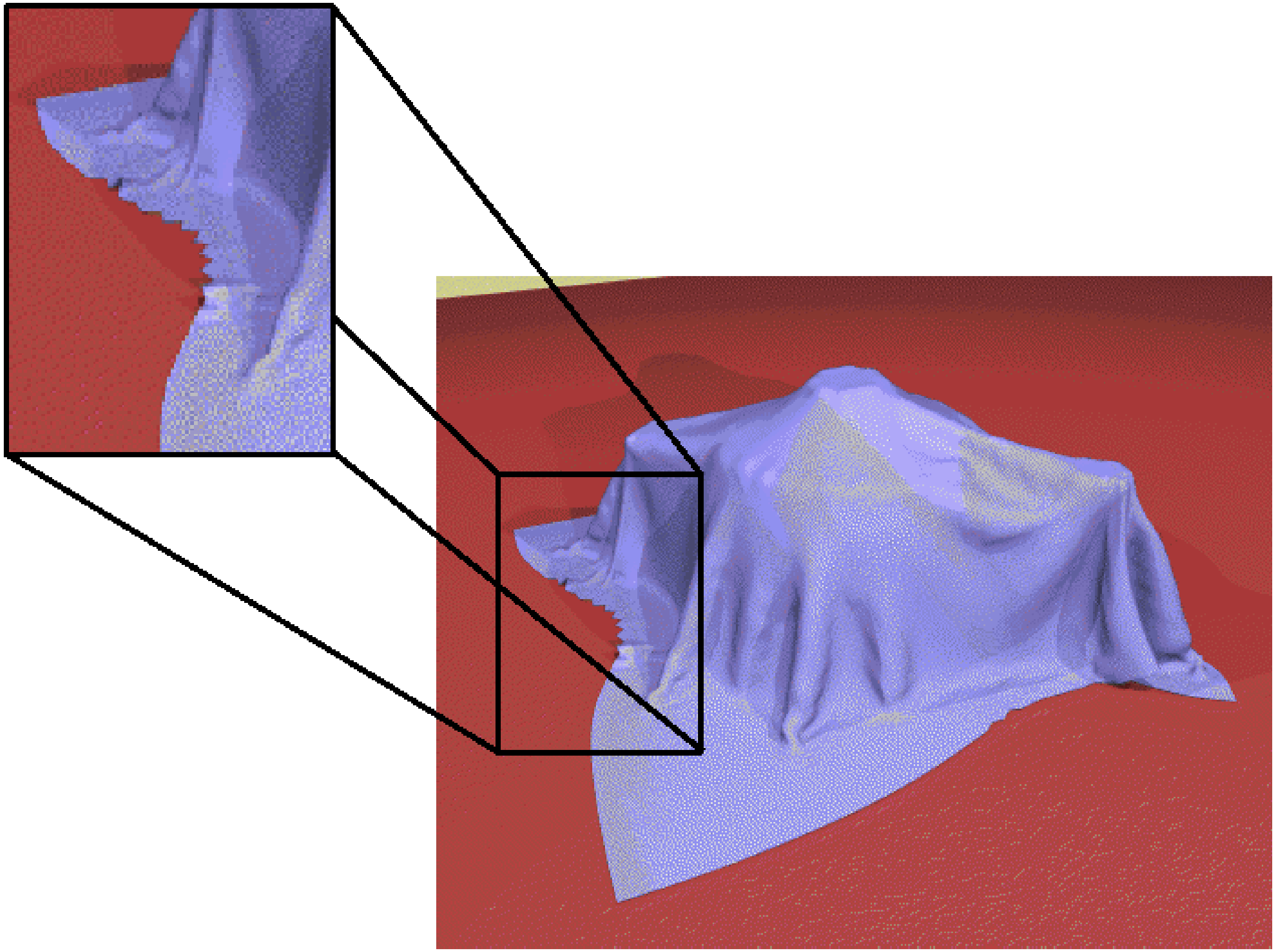
Example: Fire

Example: Bouncing Off Particles



Example: More Bouncing

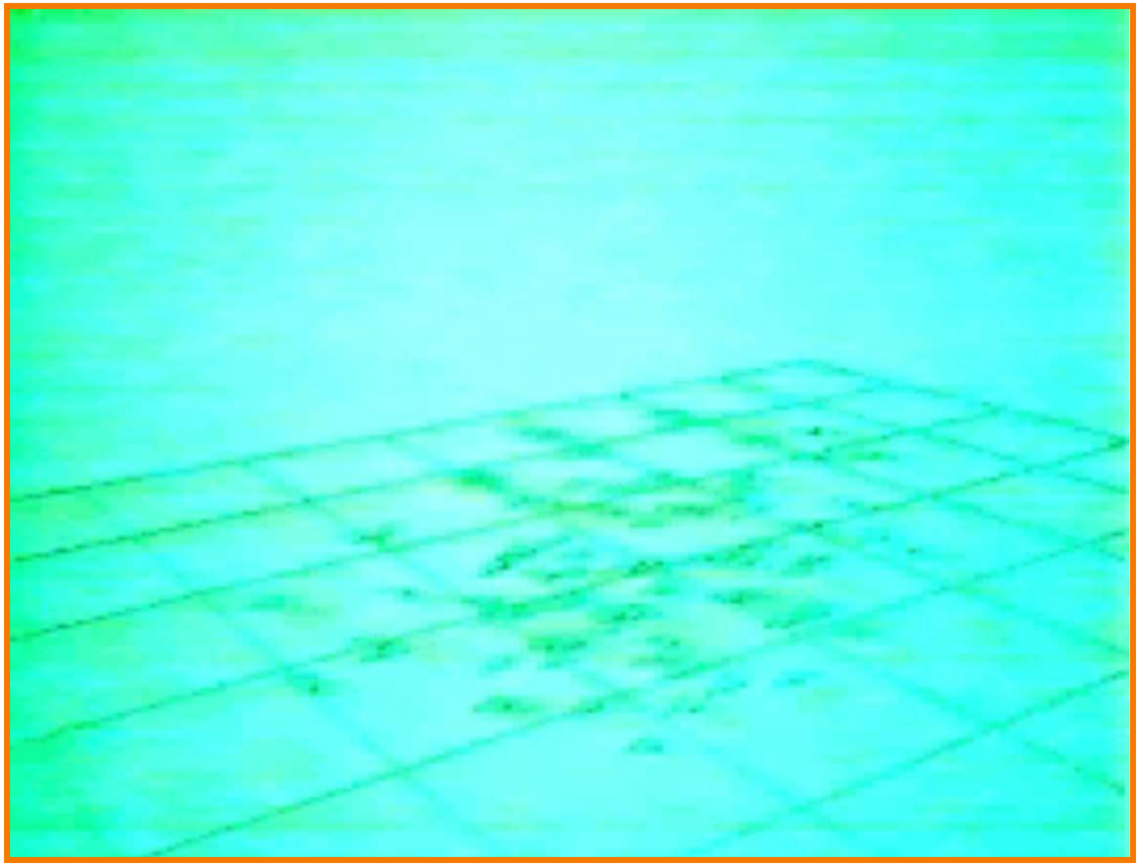
Example: Cloth



Breen

Example: Cloth

Example: Flocks & Herds



Reynolds

Summary

- Particle systems
  - Lots of particles
  - Simple physics
- Interesting behaviors
  - Waterfalls
  - Smoke
  - Cloth
  - Flocks
- Solving motion equations
  - For each step, first sum forces, then update position and velocity