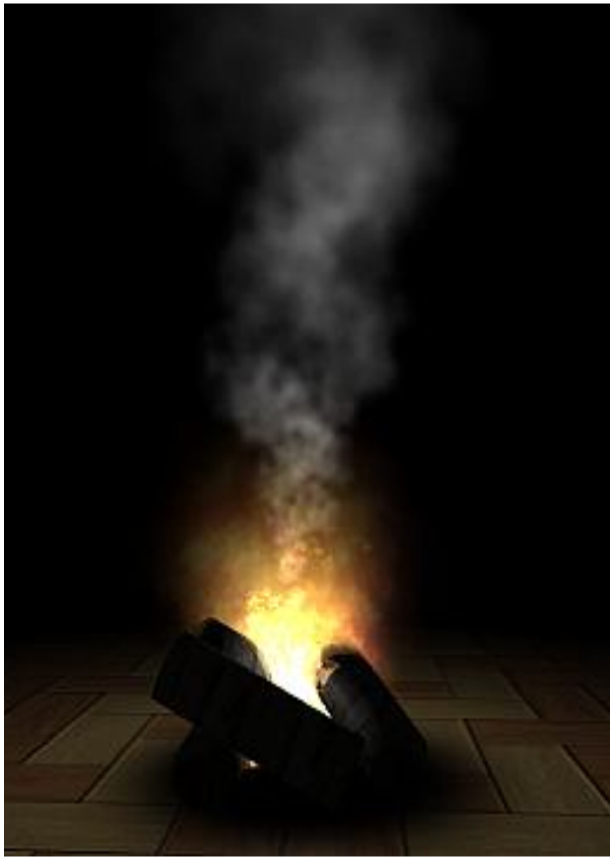
Particle Systems

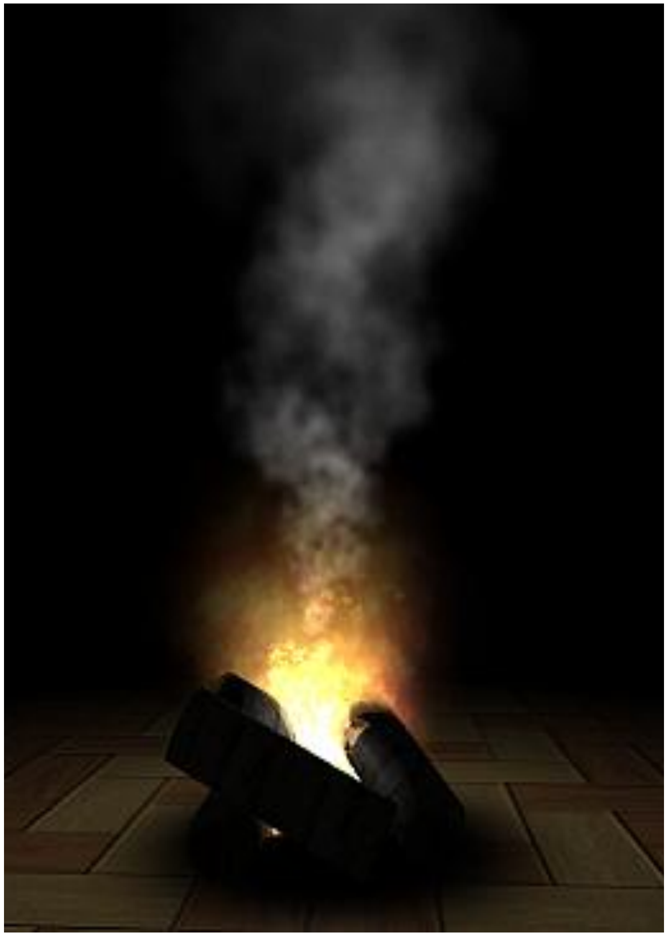


CS 312

Based on slides from Addison-Wesley and Open Courseware

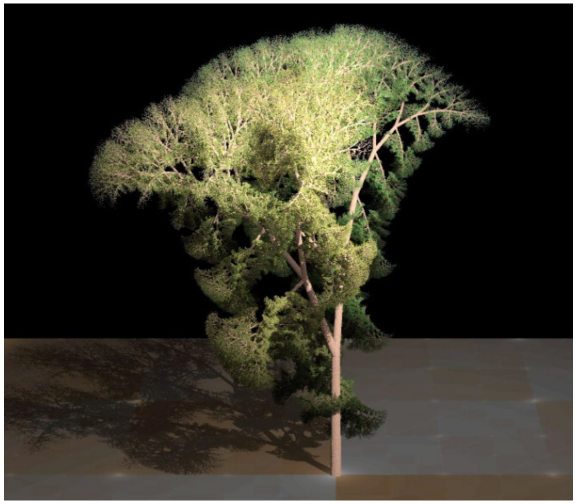
Introduction

- So far, “polygon based models” only
  - Which have achieved extraordinary success
  - … with implementation and use highly hardware supported, thus, furthering success
  - … but, there are other ways …
    - E.g, see global illumination
- Some things just are not handled well
    - Clouds, terrain, plants, crowd scenes, smoke, fire
  - Physical constraints and complex behavior not part of polygonal modeling
- Procedural methods
  - Generate geometric objects in different way
  - 1. Describe objects in an algorithmic manner
  - 2. Generate polygons only when needed as part of rendering process

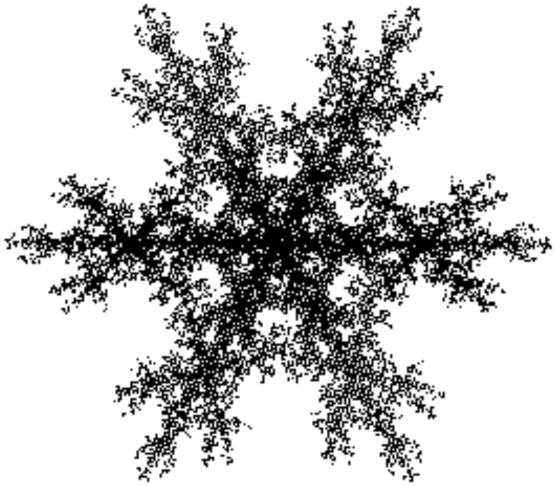


Procedural Methods

3 Approaches



- Particles that obey Newton's laws
  - Systems of thousands of particles capable of complex behaviors
  - Solving sets of differential equations
- Language based models
  - Formal language for producing objects
  - E.g., productions rules
- Fractal geometry
  - Based on self-similarity of seen in natural phenomena
  - Means to generate models to any level of detail
- Procedural noise
  - Introduce "controlled randomness" in models
    - Turbulent behavior, realistic motion in animation, fuzzy objects



Physically-based Models

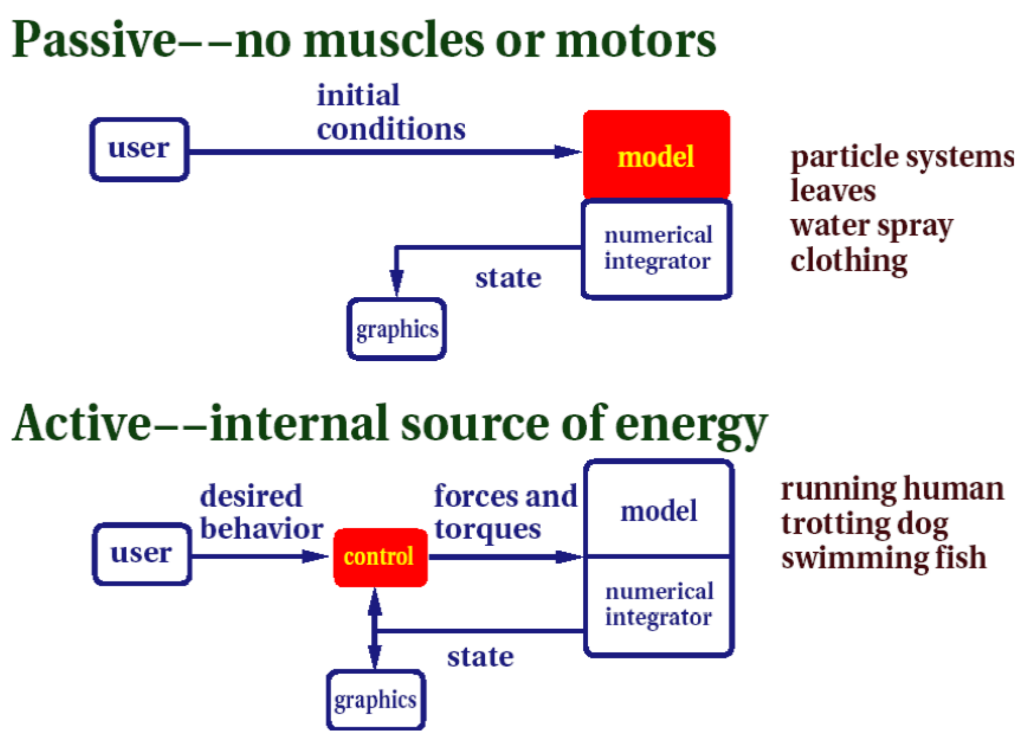
- Recall, “biggest picture” of computer graphics
  - Creating a world unconstrained by, well, anything
  - Is a good thing, e.g., scientific visualization of mathematic functions, subatomic particles and fields, visual representation of designs perhaps not realizable (yet)
  - Have seen series of techniques, e.g., Phong shading, that “look right”
    - And even had a glimpse at what about viewer makes things look right (actually *not* look right), e.g., Mach banding

- Physically-based modeling
  - Can now feasibly investigate cg systems that fully model objects obeying all physical laws … still a research topic

- Hybrid approach
  - Combination of basic physics and mathematical constraints to control dynamic behavior of objects
  - Will look at *particle systems* as an example
    - Dynamic behavior of (point) masses determined by solution of sets of coupled differential equations – will use an easily implemented solution

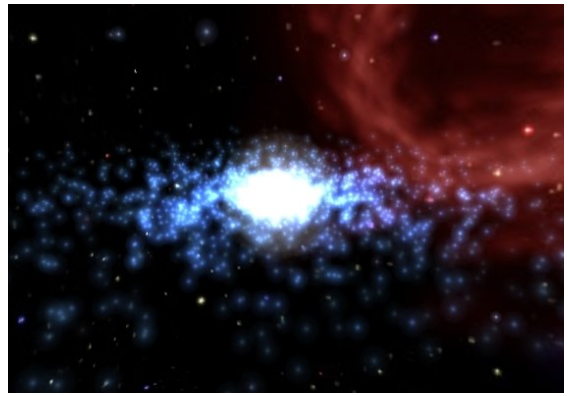
Kinematics and Dynamics

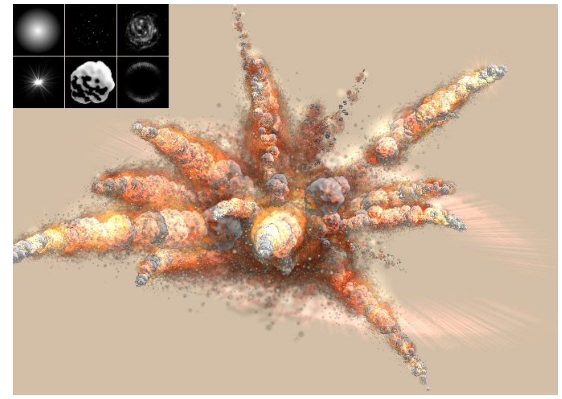
- Kinematics
  - Considers only motion
  - Determined by positions, velocities, accelerations
- Dynamics
  - Considers underlying forces
  - Compute motion from initial conditions and physics
- Dyamics - Active vs. Passive

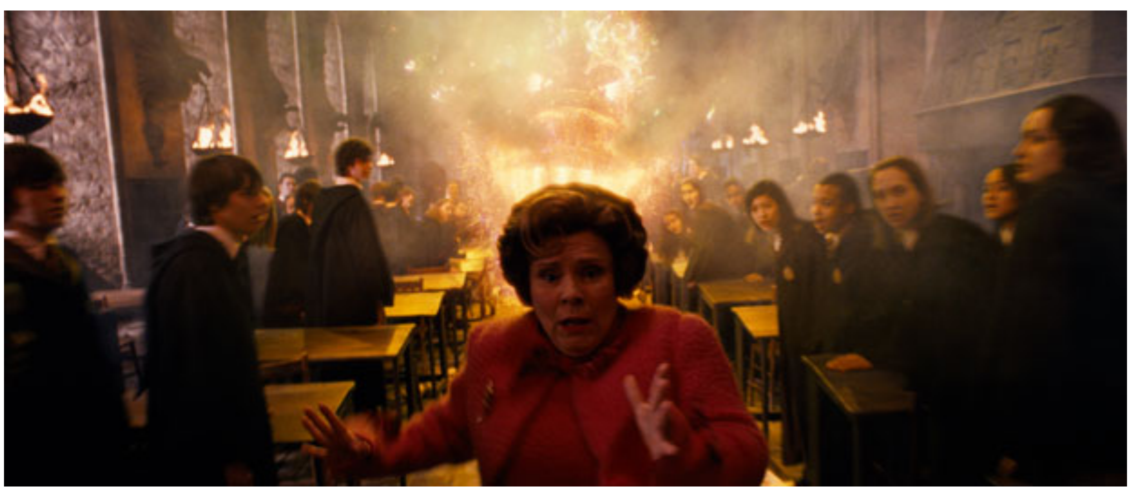


Particle Systems

- Used to model:
  - Natural phenomena
    - Clouds
    - Terrain
    - Plants
  - Group behavior
    - Animal groups, crowds
  - Real physical processes
- Individual elements –
  - forces, direction attributes







Particle System History

- 1962: Pixel clouds in “Spacewar!”
  - 2nd (or so) digital video game
- 1978: Explosion physics simulation in “Asteroids”
- 1983: “Star Trek II: Wrath of Kahn”
  - William T. Reeves
  - 1st cg paper about particle systems
  - More later
- Now: Everywhere
  - Programmable and in firmware
  - Tools for creating, e.g., Maya

Particle System History

1983, Reeves, Wrath of Khan

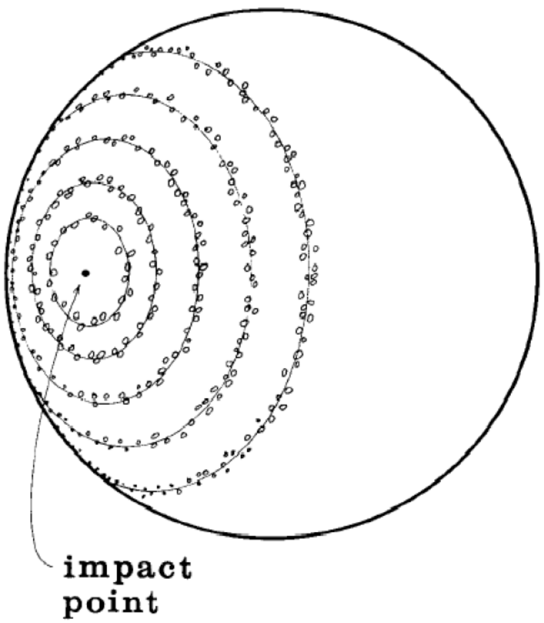
- Some 400 particle systems
  - "Chaotic effects"
  - To 750k particles
  - Genesis planet
- Each Particle Had:
  - Position
  - Velocity
  - Color
  - Lifetime
  - Age
  - Shape
  - Size
  - Transparency

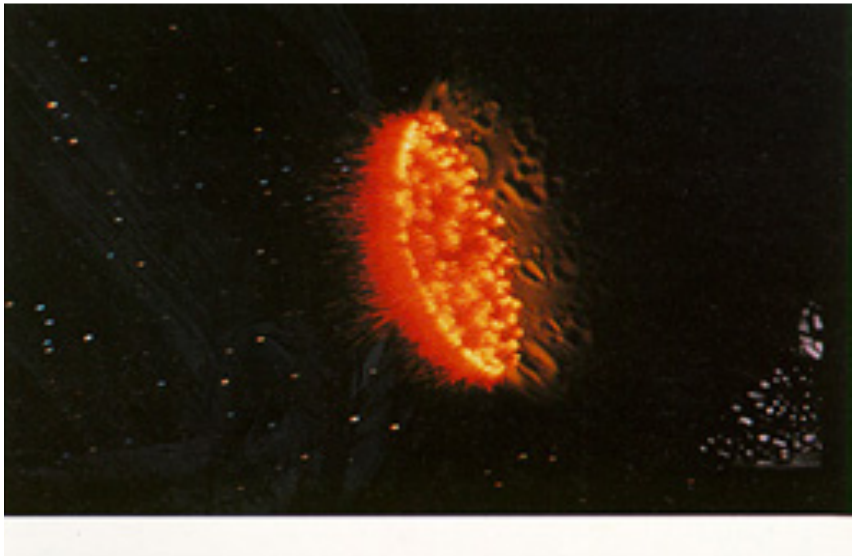
- Reeves1983: Reeves, William T.; Particle Systems – Technique for Modeling a Class of Fuzzy Objects. In SIGGRAPH Proceedings 1983, http://portal.acm.org/citation.cfm?id=357320

Example: Wrath of Khan

Distribution of particles on planet surface

- Model spread

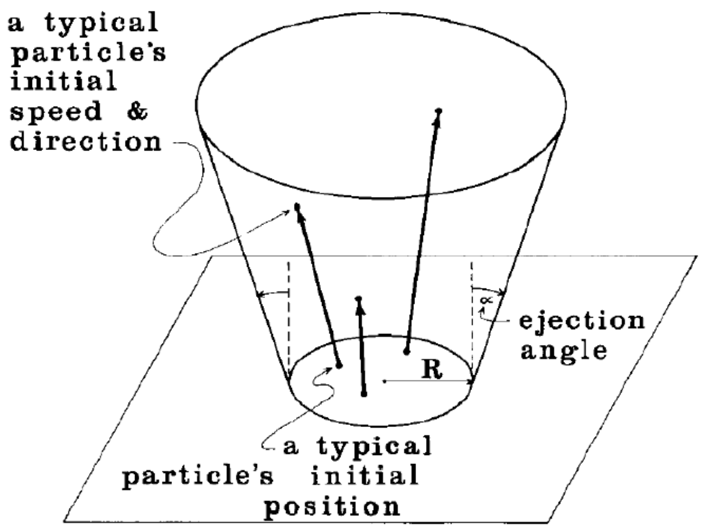




Example: Wrath of Khan

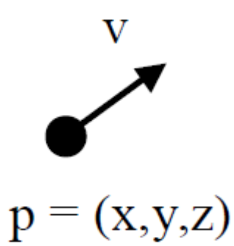
Ejection of particles from planet surface

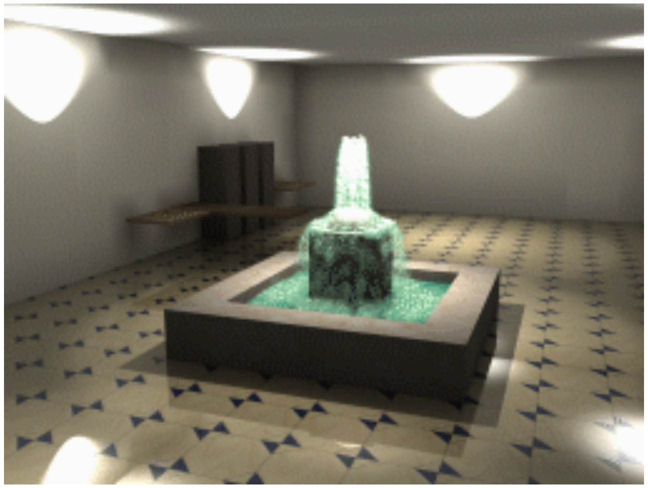
- Will see same approach for collisions



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

Newtonian Particles

Angel

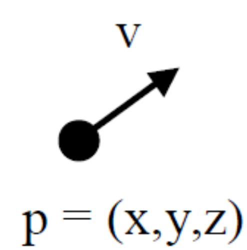
- Will model a set of particles subject to Newton's laws
  - Though could use any "laws", even your own
- Particles will obey Newton's second law:
  - The mass of the particle (m) times the particle's acceleration ($\mathbf{a}$) is equal to the sum of the forces ($\mathbf{f}$) acting on the particle
  - $m\mathbf{a} = \mathbf{f}$
  - Note that both acceleration, **a**, and force, **f**, are vectors (usually x, y, z)
  - With mass concentrated at a single point (an ideal point mass particle), state is determined completely by it position and velocity
  - Thus, in a 3d space ideal particle has 6 degrees of freedom, and a system of n particles has $6n$ state variables, position and velocity of all particles

Newtonian Particle

Angel

- Particle system is a set of particles
  - Each particle is an ideal point mass

- Six degrees of freedom
  - Position
  - Velocity
- Each particle obeys Newton's law
  - $\mathbf{f} = \mathbf{m}a$
- Particle equations
  - $\mathbf{p}_i = (x_i, y_i\ z_i)$
  - $\mathbf{v}_i = d\mathbf{p}_i /dt = \mathbf{p}_i' = (dx_i /dt, dy_i /dt , z_i /dt)$
  - $m\ \mathbf{v}_i' = \mathbf{f}_i$



- Hard part is defining force vector

Newtonian Particles

Details

- State of the i^{th} particle is given by
  - Position matrix:

$$\mathrm{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

  - -- Velocity matrix:

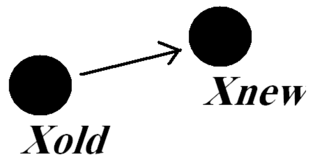
$$\mathrm{v}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{dz}{dt} \end{bmatrix}$$

- Acceleration is the derivative of velocity and velocity is the derivative of position, so can write Newton's second law for a particle as the 6*n* coupled first order differential equations
  - $\dot{\mathrm{p}}_i = v_i$
  - $\dot{\mathrm{v}}_i = \frac{1}{m_i} f_i(t)$

Simply Put, ...

Particle Dynamics

- Again, for each frame:
  - Create new particles and assign attributes
  - Delete any expired particles
  - *Update particles based on attributes & physics*
  - Render particles



- Particle's position in each succeeding frame can be computed by knowing its velocity
  - speed and direction of movement

- This can be modified by an acceleration force for more complex movement, e.g., gravity simulation

Solution of Particle Systems

Angel

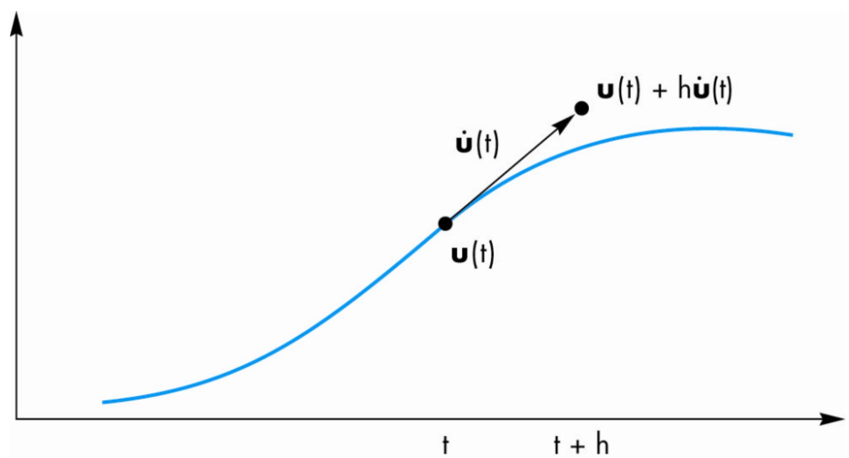
```
float time, delta state[6n], force[3n];
   state = initial_state();
   for(time = t0; time<final_time, time+=delta) {
        force =  force_function(state, time);
        state = ode(force, state, time, delta);
        render(state, time)
        }
```

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

Solution of ODEs

Angel details

- Particle system has 6*n* ordinary differential equations
- Write set as $d\mathbf{u}/dt = g(\mathbf{u},t)$
- Solve by approximations using Taylor's Theorem



Solution of ODEs, 2

Angel details

- Euler's Method

$$\mathbf{u}(t + h) \approx \mathbf{u}(t) + h\, d\mathbf{u}/dt = \mathbf{u}(t) + h\mathbf{g}(\mathbf{u}, t)$$

Per step error is $O(h^2)$

Require one force evaluation per time step

Problem is numerical instability - depends on step size

- Improved Euler

$$\mathbf{u}(t + h) \approx \mathbf{u}(t) + h/2(\mathbf{g}(\mathbf{u}, t) + \mathbf{g}(\mathbf{u}, t{+}h))$$

Per step error is $O(h^3)$

Also allows for larger step sizes

But requires two function evaluations per step

Also known as Runge-Kutta method of order 2

Force Vector

Particle System Forces

- A number of means to specify forces have been developed
- Most simply, independent particles – no interaction with other particles
  - Gravity, wind forces
  - $O(n)$ calculation
- Coupled Particles $O(n)$
  - Meshes
    - Useful for cloth
  - Spring-Mass Systems
- Coupled Particles $O(n^2)$
  - Attractive and repulsive forces

Simple Forces

E.g., Gravity

- Particle field forces
  - Usually can group particles into equivalent point masses
  - E.g.,Consider simple gravity
  - Not compute forces due to sun, moon, and other large bodies
  - Rather, we use the gravitational field
  - Same for wind forces, drag, etc.

- Consider force on particle i

$$\mathbf{f}_i = \mathbf{f}_i(\mathbf{p}_i, \mathbf{v}_i)$$

- Gravity $\mathbf{f}_i = \mathbf{g}$
  - Really easy

$$\mathbf{g}_i = (0, -g, 0)$$

$\mathbf{p}_i(t_0)$, $\mathbf{v}_i(t_0)$

More Complex Force

- Local Force – Flow Field
- Stokes Law of drag force on a sphere

  $F_d = 6\Pi\eta r(v-v_{fl})$

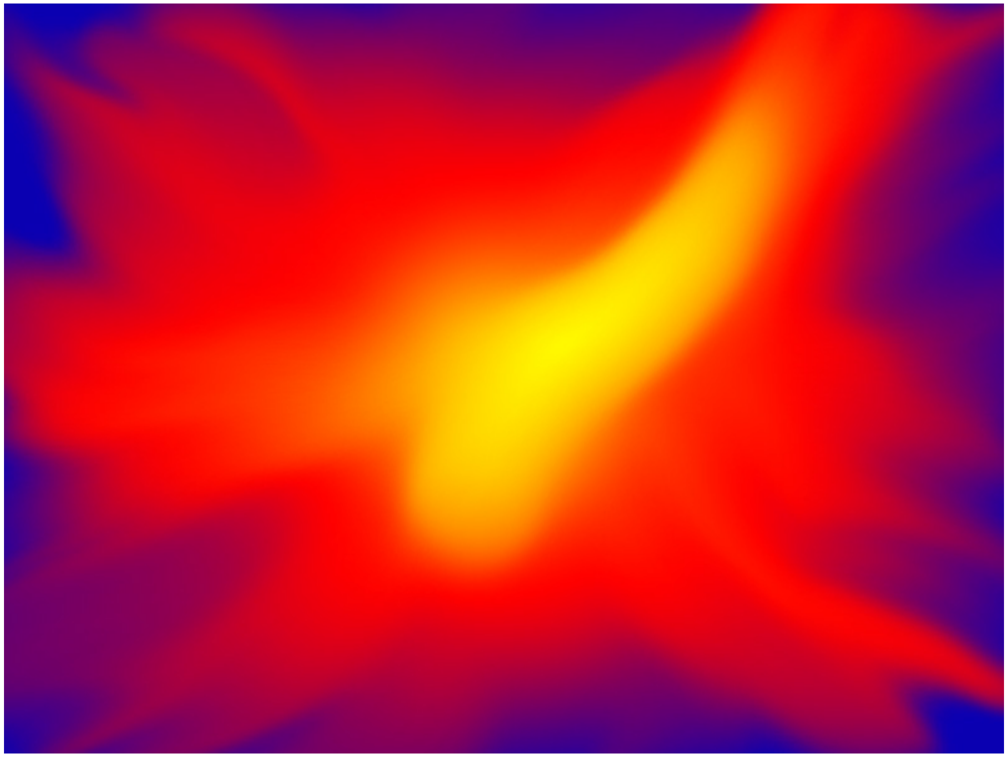
  η = viscosity

  r = radius of sphere

  $C = 6\Pi\eta r$ (constant)

  v = particle velocity

  v_{fl} = flow velocity

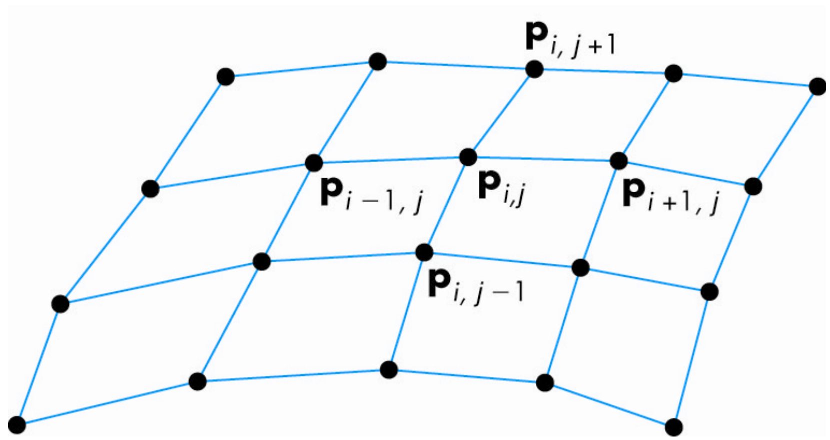


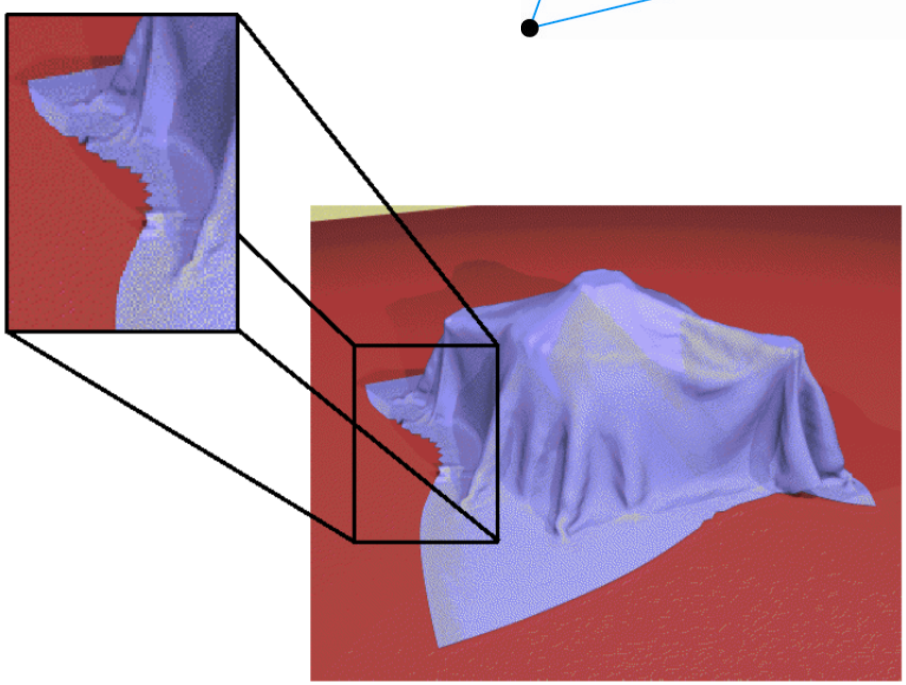
Sample Flow Field

Meshes

- Connect each particle to its closest neighbors
  - O(n) force calculation

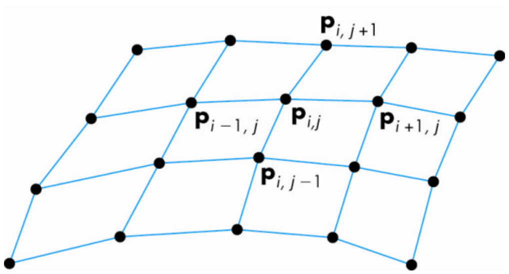
- Use spring-mass system



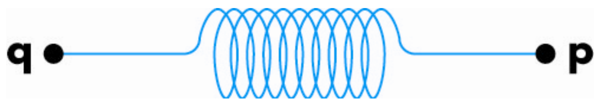


Spring Forces

- Used modeling forces, e.g., meshes
  - Particle connected to its neighbor(s) by a spring
  - Interior point in mesh has four forces applied to it
  - Widely used in graph layout



- Hooke's law: force proportional to distance ($d = \|\mathbf{p} - \mathbf{q}\|$) between points



- Let s be distance when no force

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

k_s is the spring constant

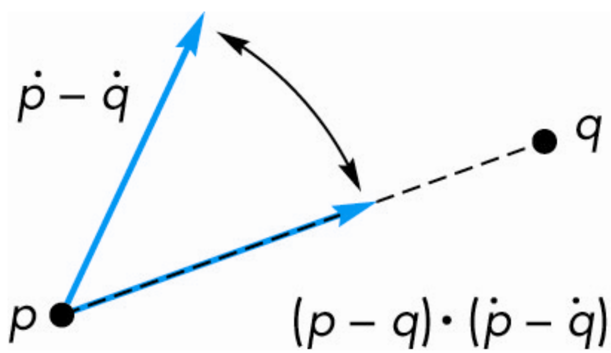
$\mathbf{d}/|\mathbf{d}|$ is a unit vector pointed from $\mathbf{p}$ to $\mathbf{q}$

Spring Damping

- A pure spring-mass will oscillate forever
- Must add a damping term

$$\mathbf{f} = -(k_s(|\mathbf{d}| - s) + k_d\, \dot{\mathbf{d}} \cdot \mathbf{d}/|\mathbf{d}|)\mathbf{d}/|\mathbf{d}|$$

- Must project velocity



Attraction and Repulsion

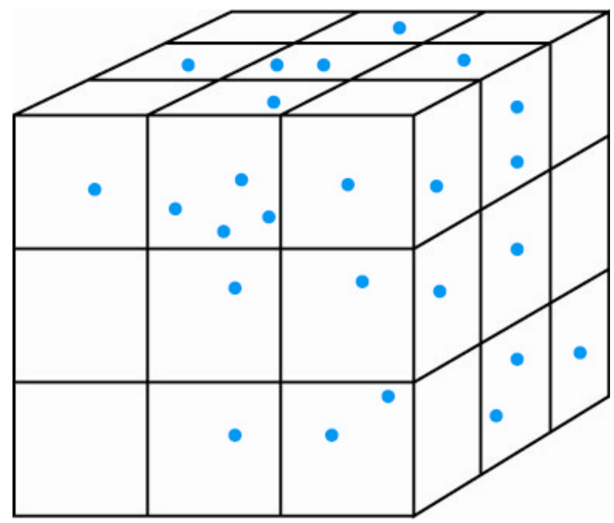
- Inverse square law

  $$\mathbf{f} = -k_r\mathbf{d}/|\mathbf{d}|^3$$

- General case requires $O(n^2)$ calculation
- In most problems, the drop off is such that not many particles contribute to the forces on any given particle
- Sorting problem: is it $O(n \log n)$?

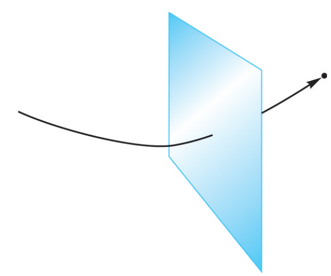
Boxes

- $O(n^2)$ algs when consider interactions among all particles
- Spatial subdivision technique
- Divide space into boxes
- Particle can only interact with particles in its box or the neighboring boxes
- Must update which box a particle belongs to after each time step

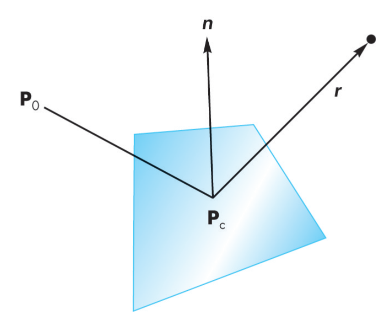


Constraints and Collisions

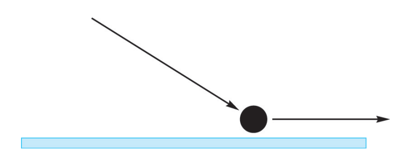
- Constraints
  - Easy in cg to ignore physical reality
    - Surfaces are virtual
  - Must detect collisions if want exact solution
    - $O(n^2)$ limiting, $O(6)$ for box sides!
  - Can approximate with repulsive forces



- Collisions
  - Once detect a collision, we can calculate new path
  - Use coefficient of restitution
  - Reflect vertical component
  - May have to use partial time step



- Contact forces



Grass / Hair / Fur

- Entire trajectory of a particle over its lifespan is rendered to produce a static image
- Green and dark green colors assigned to the particles which are shaded on the basis of the scene's light sources
- Each particle becomes a blade of grass
- Also works to create hair, fur, etc.

white.sand by Alvy Ray Smith
(he was also working at Lucasfilm)

Tools - Alias|Wavefront's Maya

- Tutorial
  - http://dma.canisius.edu/~moskalp/tutorials/Particles/ParticlesWeb.mov