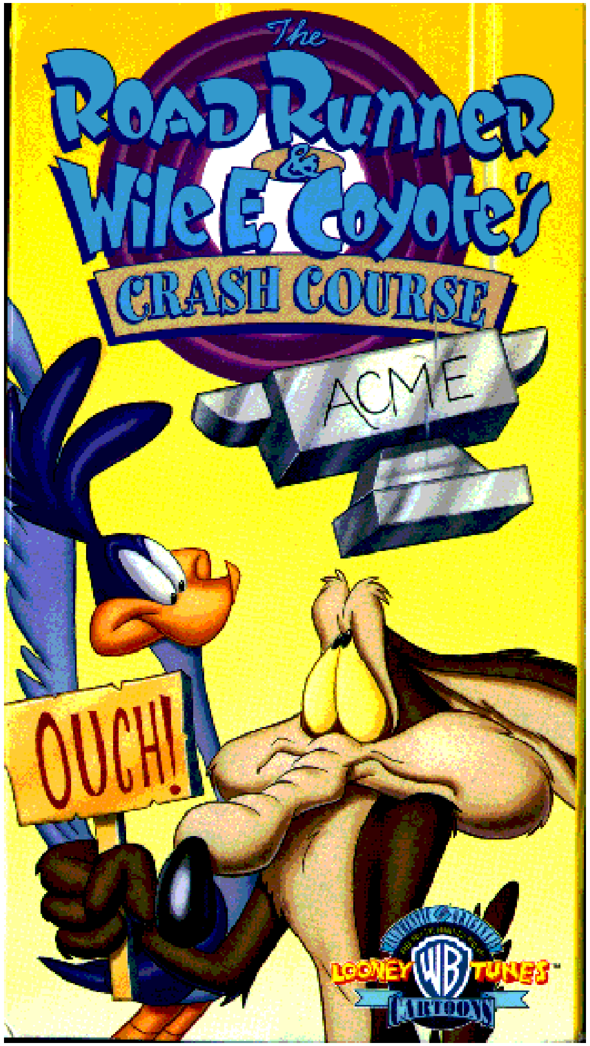
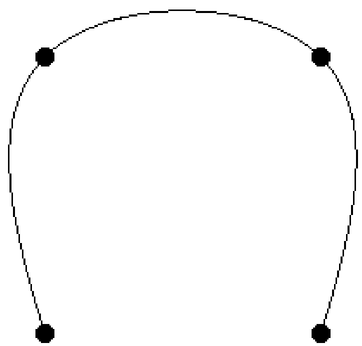
Computer Animation
Particle systems



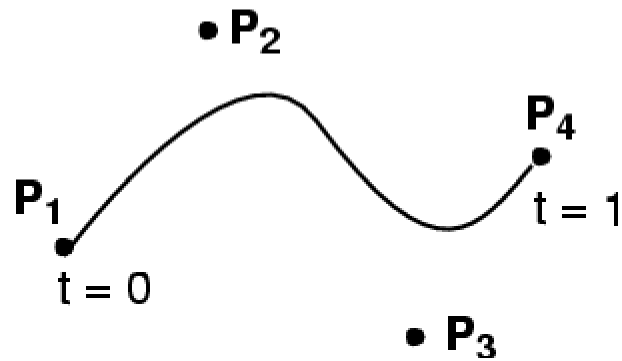
- Some slides courtesy of Jovan Popovic & Ronen Barzel

Last time?

- Splines

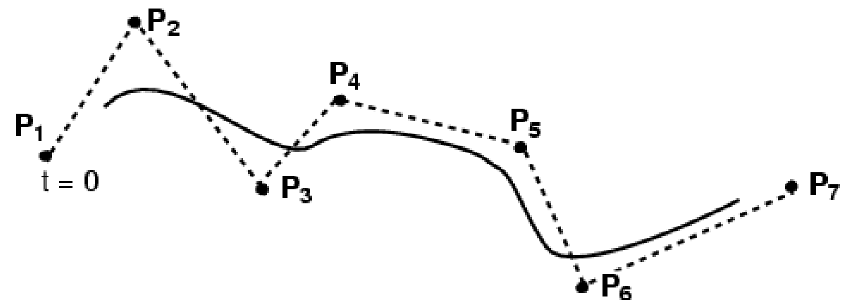


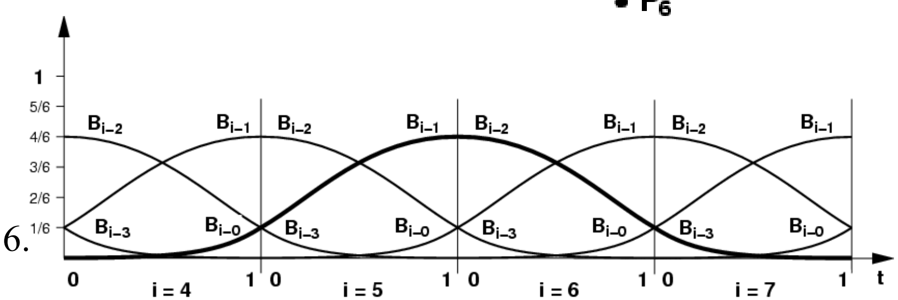
Interpolation



Bézier (approximation)

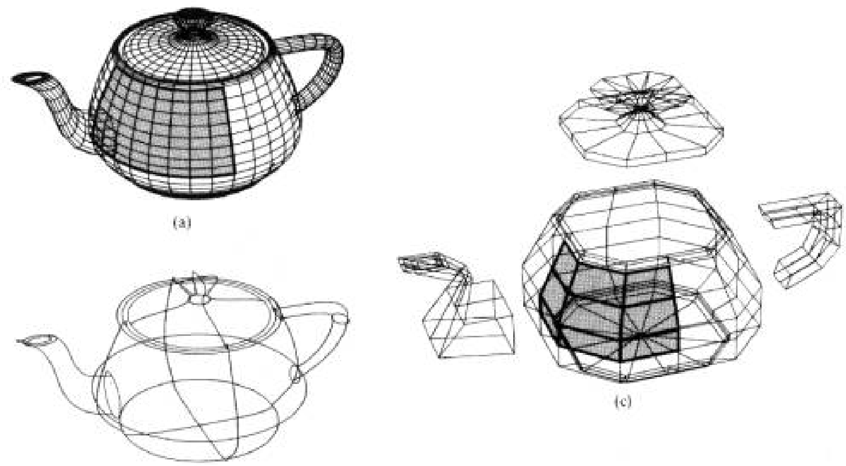
BSpline (approximation)

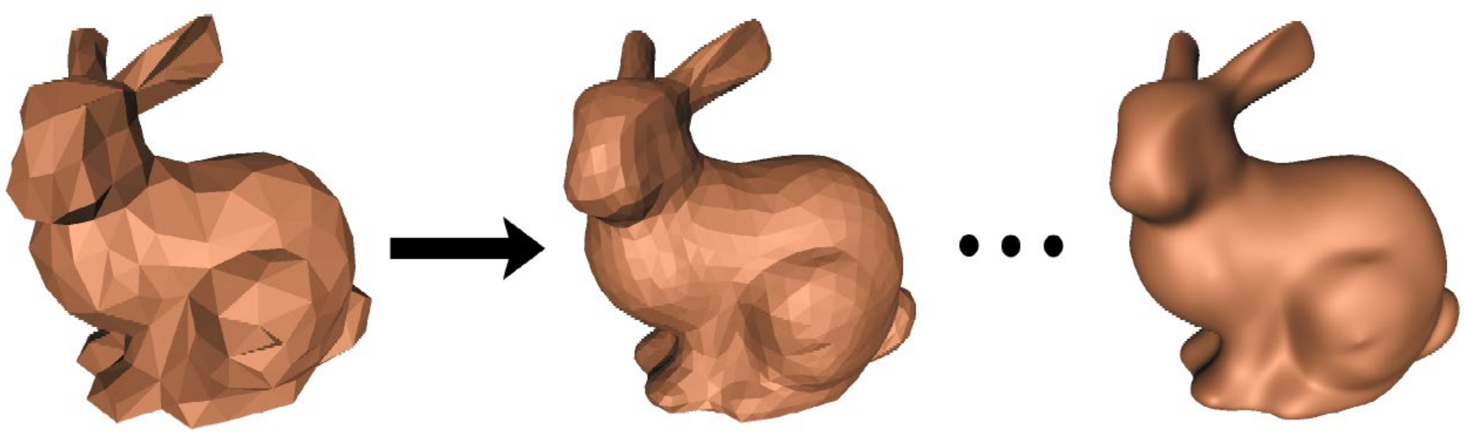




Last time?

- Splines
- Basis functions, conversion (4x4 matrix)
- de Casteljau's algorithm
- C1 continuity
- Bicubic patches
- Subdivision surfaces





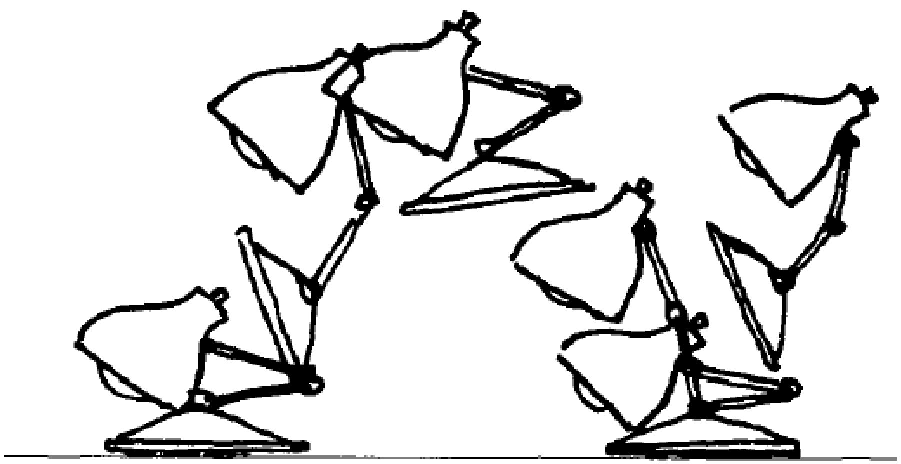
Assignment 10

- Proposal due this Friday
- Assignment due Dec 3

- You have only 10 days
- Be specific in your goals
- Avoid risky exploratory subjects

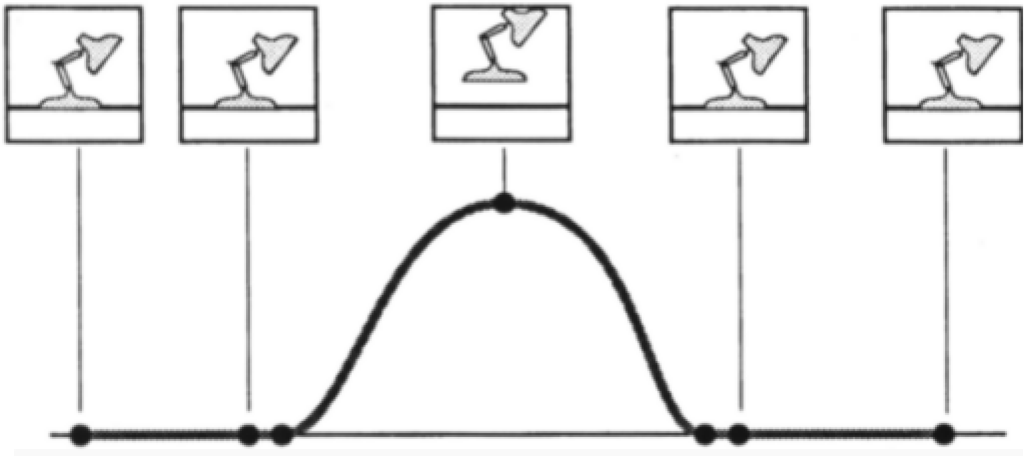
Today: animation

- So far, we have focused on synthesizing the image given a geometric and photometric description
- Now, how do we specify or generate the motion of the objects?



Keyframing

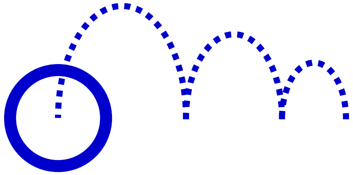
- automate the inbetweening
- use spline curves
- good control
- less tedious
- creating a good animation still requires considerable skill and talent



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

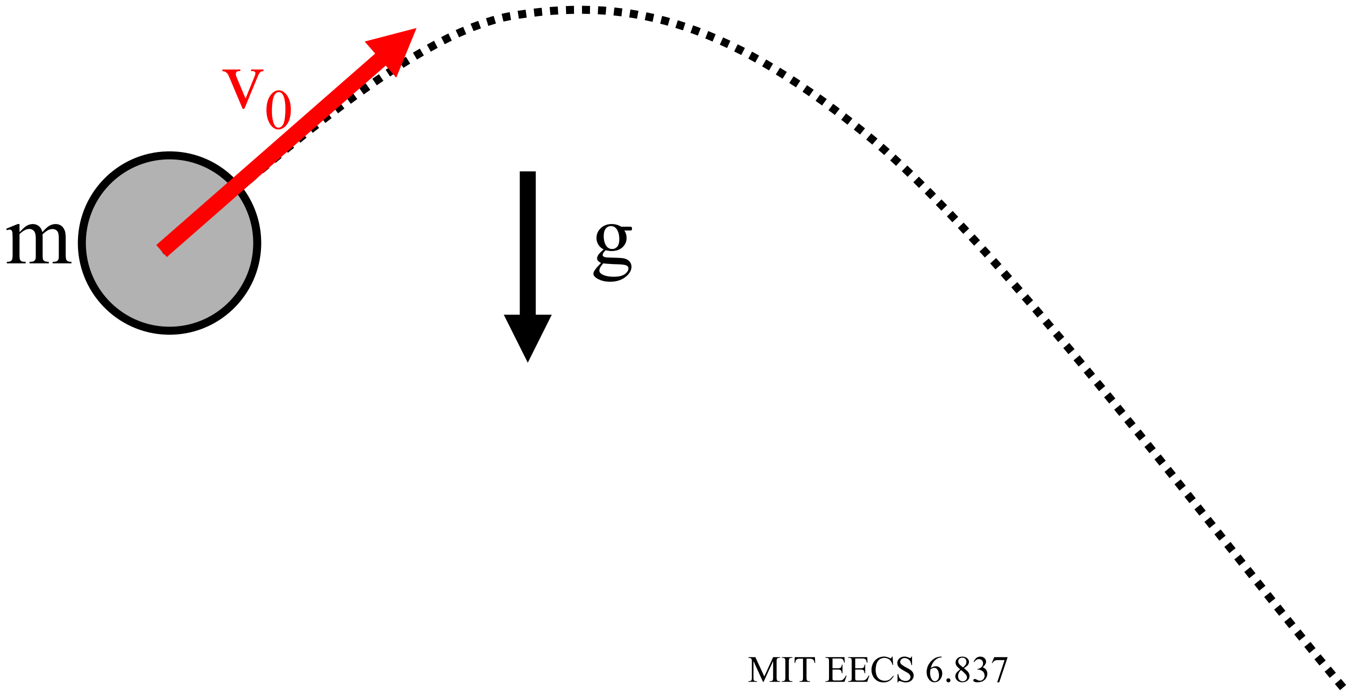
Procedural animation

- 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
- Example 2: A bouncing ball
  - $\mathrm{Abs}(\sin(\omega t+\theta_0))*e^{-kt}$



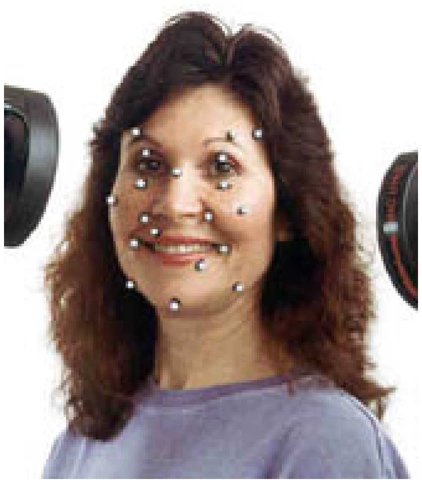
Physically Based Animation

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



Motion Capture

- Usually uses optical markers and multiple high-speed cameras
- Triangulate to get marker 3D position
- Faces or joints
- Captures style, subtle nuances and realism
- You must observe someone do something



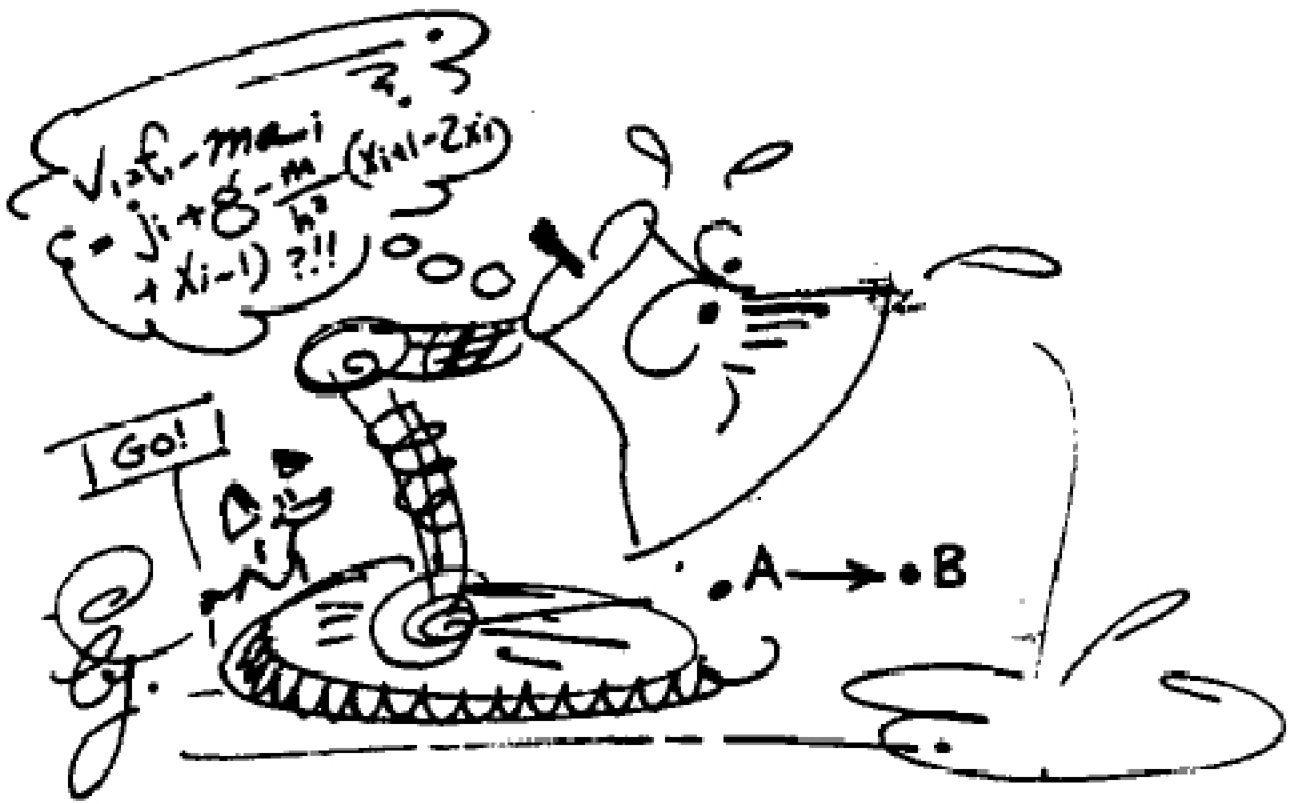
See also

- http://www.pixar.com/howwedoit/index.html

Questions?

Now

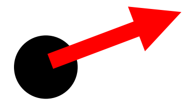
Dynamics



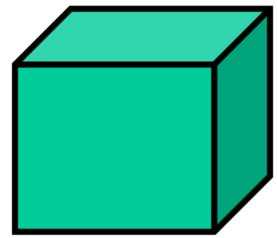
ACM© 1988 “Spacetime Constraints”

Types of dynamics

- Point



- Rigid body

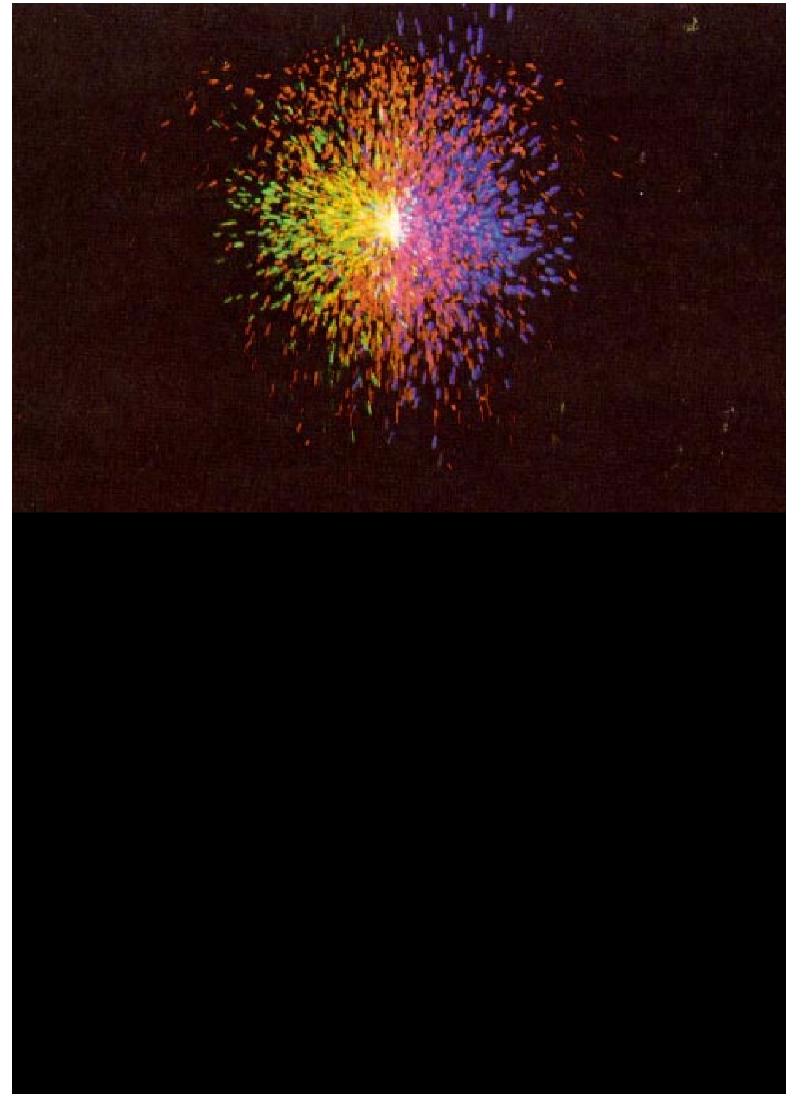


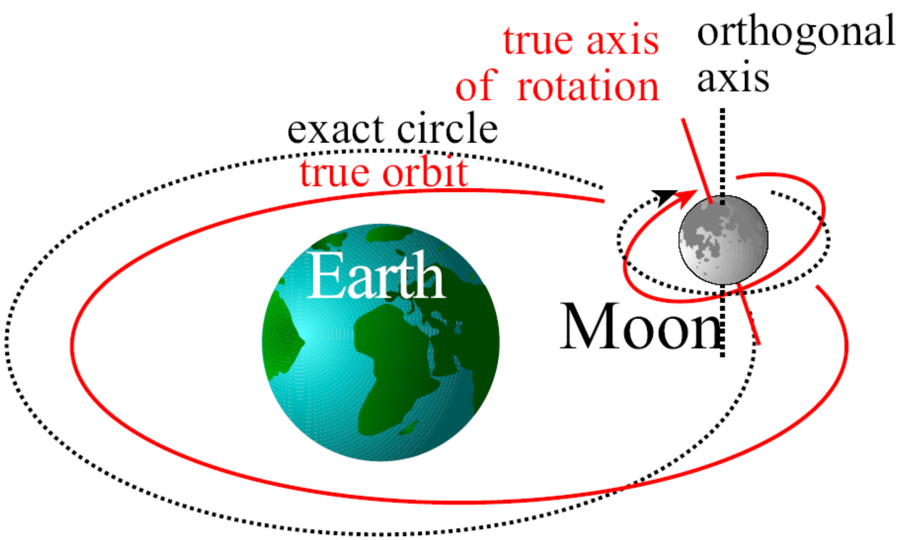
- Deformable body
  (include clothes, fluids, smoke, etc.)

Animation by Mark Carlson

Today we focus on point dynamics

- But we use tons of points
- Particles systems





Overview

- Generate tons of particles
- Describe the external forces with a force field
- Integrate the laws of mechanics
  - Lots of differential equations ;-(

- Each particle is described by its state
  - Position, velocity, color, mass, lifetime, shape, etc.

- More advanced versions exist: flocks, crowds

What is a particle system?

- Collection of many small simple particles
- Particle motion influenced by force fields
- Particles created by *generators*
- Particles often have *lifetimes*
- Used for, e.g:
  – sand, dust, smoke, sparks, flame, water, …

Videos

- Demos from http://users.rcn.com/mba.dnai/software/flow/
- **flow** is a particle animation application under development by Mark B. Allan and theReptileLabourProject.

Questions?

Particle motion

- mass m, position x, velocity v
- equations of motion:

$$\frac{d}{dt}x(t) = v(t)$$

$$\frac{d}{dt}v(t) = \frac{1}{m}F(x, v, t)$$

- Ordinary Differential Equation:

$$\mathbf{X} = \begin{pmatrix} x \\ v \end{pmatrix} \qquad f(X, t) = \begin{pmatrix} v \\ \frac{1}{m}F(x, v, t) \end{pmatrix}$$

What is an ODE?

- Ordinary Differential Equation
- Relates value of a function to its derivatives:

$$\frac{dx}{dt} = -k\,x(t)$$

$$m\ddot{x} - \lambda\dot{x} - g = -k(x - p)$$

$$\begin{cases} y' = x \\ x' = -y \end{cases}$$

- "Ordinary" = function of one variable
- Partial Differential Equation (PDE): more variables

Standard ODE

- Generic form for first-order ODE:

$$\frac{d\,\mathbf{X}(t)}{dt} = f(\mathbf{X}(t), t)$$

$$\mathbf{X} : \mathbb{R} \to \mathbb{R}^n$$

$$f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$$

- Note:
  - typically t is time
  - sometimes use **Y** instead of **X,** sometimes x instead of t
  - names sometimes confusing: often $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$

Why do we care?

- Differential equations describe (almost) everything
- in the world:
  - physics
  - chemistry
  - engineering
  - ecology
  - economy
  - weather
  - …
- Also useful for animation!
- ODEs are fundamental. PDEs build on ODEs

Solving differential equations

- Analytic solutions to differential equations
- Many standard forms, e.g.
- E.g. $x' = kx$! $x(t)=x_0e^{k}t$

$$a\ddot{x}+b\dot{x}+cx=0 \quad \Rightarrow \quad x=\begin{cases} c_1e^{r_1t}+c_2e^{r_2t} & b^2>4ac \\ c_1e^{rt}+c_2xe^{rt} & b^2=4ac \\ c_1e^{\alpha t}\cos(\beta t)+c_2xe^{\alpha t}\sin(\beta t) & b^2<4ac \end{cases}$$

- But most can't be solved analytically:
  - 3-body problem

Numerical solutions to ODEs

$$\frac{d\,\mathbf{X}(t)}{dt} = f(\mathbf{X}(t), t)$$

- Given a function $f(\mathbf{X},t)$ compute $\mathbf{X}(t)$
- Typically, *initial value problems*:
  - Given values $\mathbf{X}(t_0)=\mathbf{X}_0$
  - Find values $\mathbf{X}(t)$ for $t > t_0$
- Also, boundary value problems, constrained problems, …

Solving ODEs for animation

$$\mathbf{X}(t) = \mathbf{X}_0 \qquad t = t_0$$

$$\tfrac{d}{dt}\mathbf{X}(t) = f(\mathbf{X}(t), t) \qquad t \geq t_0$$

- For animation, want a series of values:

$$\mathbf{X}(t_i) \qquad t_i = t_0, t_1, t_2, \ldots$$

  – samples of the continuous function $\mathbf{X}(t)$
  – i.e., frames of an animation

Questions?

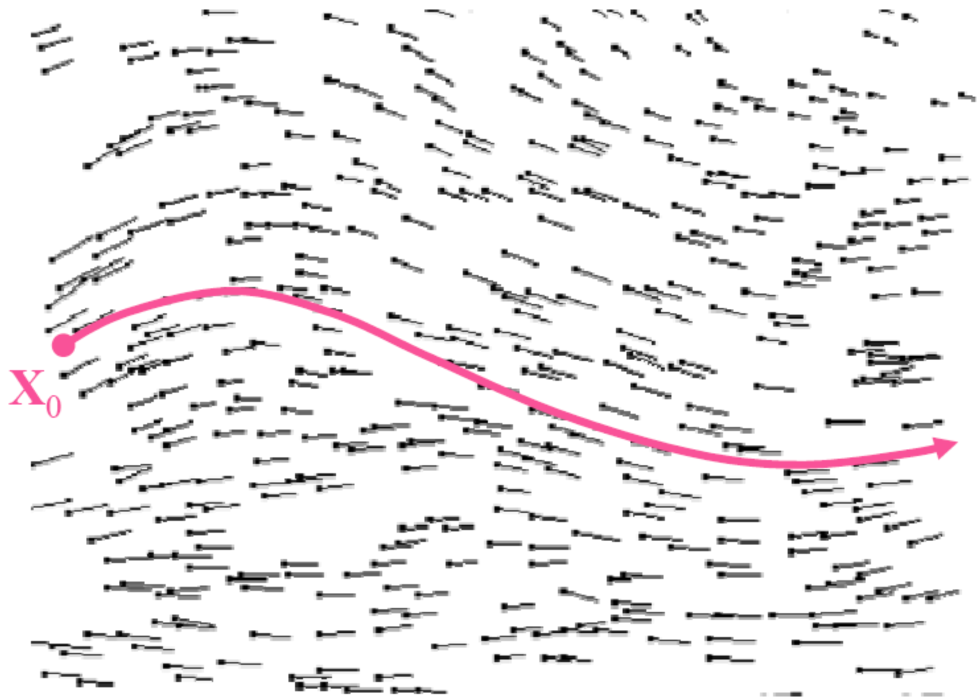
Path through a field

- $f(\mathbf{X},t)$ is a vector field defined everywhere
  - E.g. a velocity field

- it may change based on t

Path through a field

- *f(**X**,t)* is a vector field defined everywhere
  - E.g. a velocity field



- ***X**(t)* is a path through the field

Higher order ODEs

- E.g., Mechanics has 2nd order ODE:

$$\frac{d^2}{dt^2}x = \frac{1}{m}F$$

- Express as 1st order ODE by defining *v(t)*:

$$\frac{d}{dt}x(t) = v(t)$$

$$\frac{d}{dt}v(t) = \frac{1}{m}F(x, v, t)$$

$$\mathbf{X} = \begin{pmatrix} x \\ v \end{pmatrix} \qquad f(X, t) = \begin{pmatrix} v \\ \frac{1}{m}F(x, v, t) \end{pmatrix}$$

E.g., for a 3D particle

- We have a 6 dimension ODE problem:

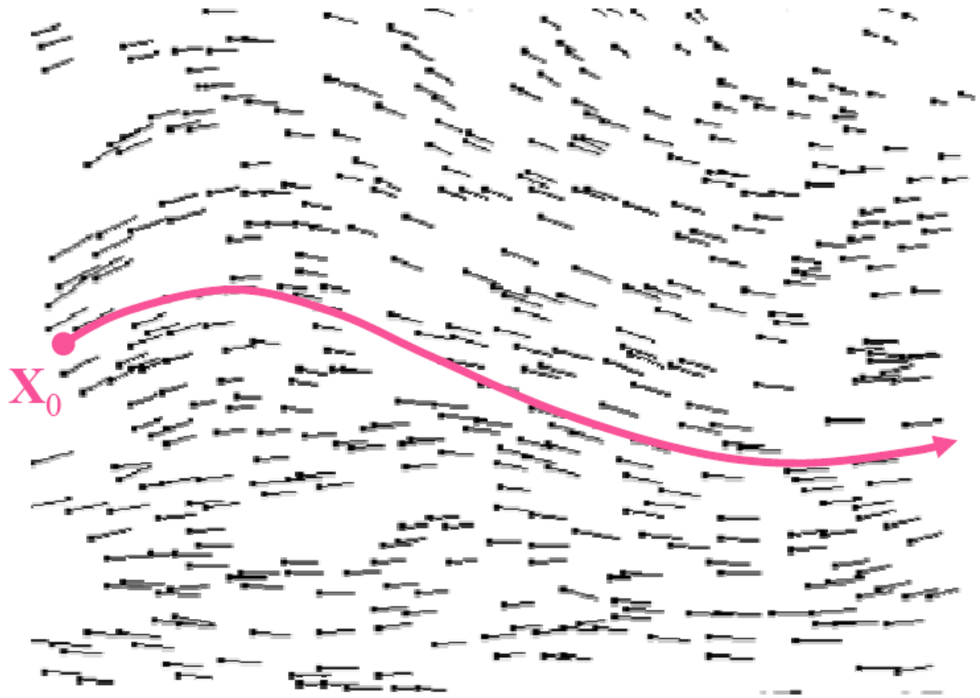
$$\mathbf{X} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{pmatrix} \qquad f(\mathbf{X}, t) = \begin{pmatrix} v_x \\ v_y \\ v_z \\ \frac{1}{m} F_x(\mathbf{X}, t) \\ \frac{1}{m} F_y(\mathbf{X}, t) \\ \frac{1}{m} F_z(\mathbf{X}, t) \end{pmatrix}$$

For a collection of 3D particles…

$$\mathbf{X} = \begin{pmatrix} p_x^{(1)} \\ p_y^{(1)} \\ p_z^{(1)} \\ v_x^{(1)} \\ v_y^{(1)} \\ v_z^{(1)} \\ p_x^{(2)} \\ p_y^{(2)} \\ p_z^{(2)} \\ v_x^{(2)} \\ v_y^{(2)} \\ v_z^{(2)} \\ \vdots \end{pmatrix} \qquad f(\mathbf{X},t) = \begin{pmatrix} v_x^{(1)} \\ v_y^{(1)} \\ v_z^{(1)} \\ \frac{1}{m_1} F_x^{(1)}(\mathbf{X},t) \\ \frac{1}{m_1} F_y^{(1)}(\mathbf{X},t) \\ \frac{1}{m_1} F_z^{(1)}(\mathbf{X},t) \\ v_x^{(2)} \\ v_y^{(2)} \\ v_z^{(2)} \\ \frac{1}{m_2} F_x^{(2)}(\mathbf{X},t) \\ \frac{1}{m_2} F_y^{(2)}(\mathbf{X},t) \\ \frac{1}{m_2} F_z^{(2)}(\mathbf{X},t) \\ \vdots \end{pmatrix}$$

Still, a path through a field:

- ***X(t)***: path in multidimensional state space

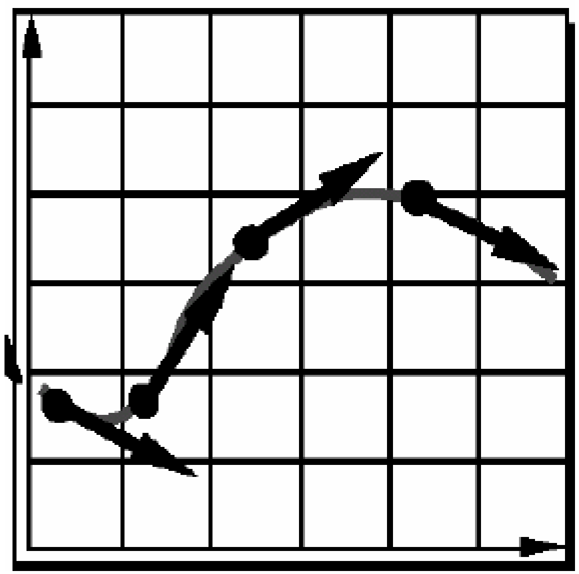


- For ODE, it's an array of numbers.

Questions?

Intuitive solution: take steps

- Current state X
- Examine f(X,t) at (or near) current state
- Take a step to new value of X
- Most solvers do some form of this



Euler's method

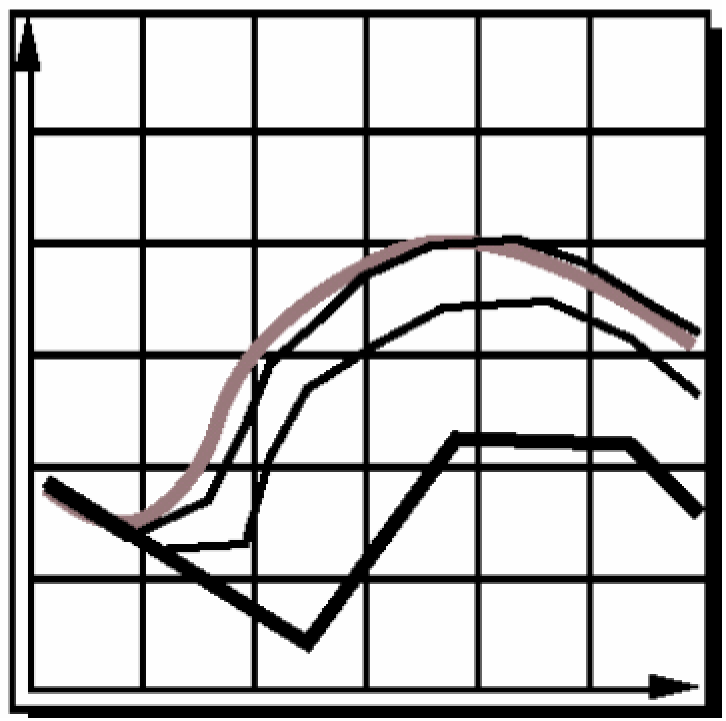
- Simplest and most intuitive.
- Define **step size** h
- Given $\mathbf{X}_0=\mathbf{X}(t_0)$, take step:

$$t_1 = t_0 + h$$
$$\mathbf{X}_1 = \mathbf{X}_0 + h\, f(\mathbf{X}_0, t_0)$$

- Piecewise-linear approximation to the curve

Effect of step size

- Step size controls accuracy
- Smaller steps more closely follow curve
- For animation, may need to take many small steps per frame



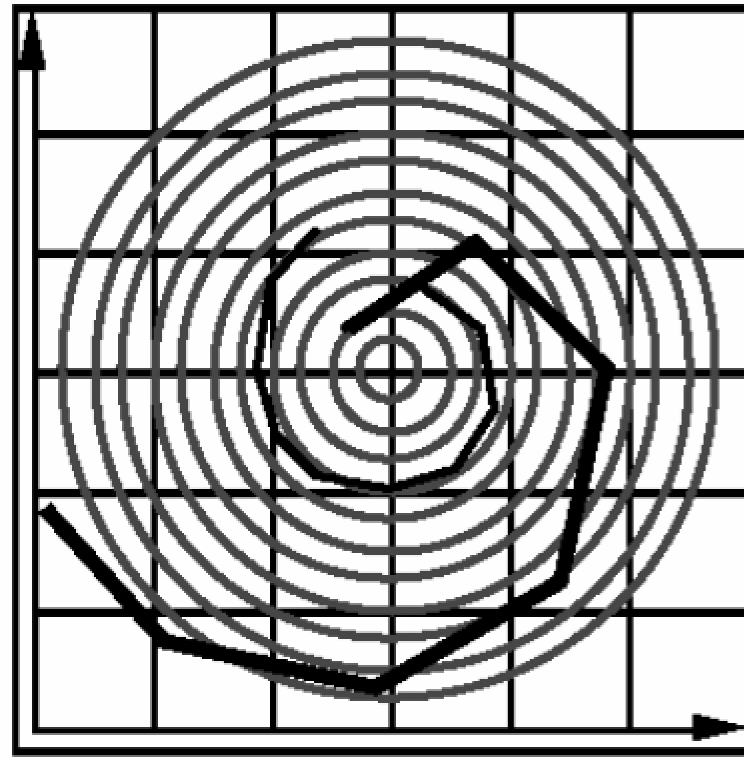
Euler's method: inaccurate

- Moves along tangent; can leave curve, e.g.:

$$f(\mathbf{X},t)=\begin{pmatrix}-y\\ x\end{pmatrix}$$

- Exact solution is circle:

$$\mathbf{X}(t)=\begin{pmatrix}r\cos(t+k)\\ r\sin(t+k)\end{pmatrix}$$



- Euler's spirals outward
- no matter how small h is

Euler's method: unstable

$$f(x,t) = -kx$$

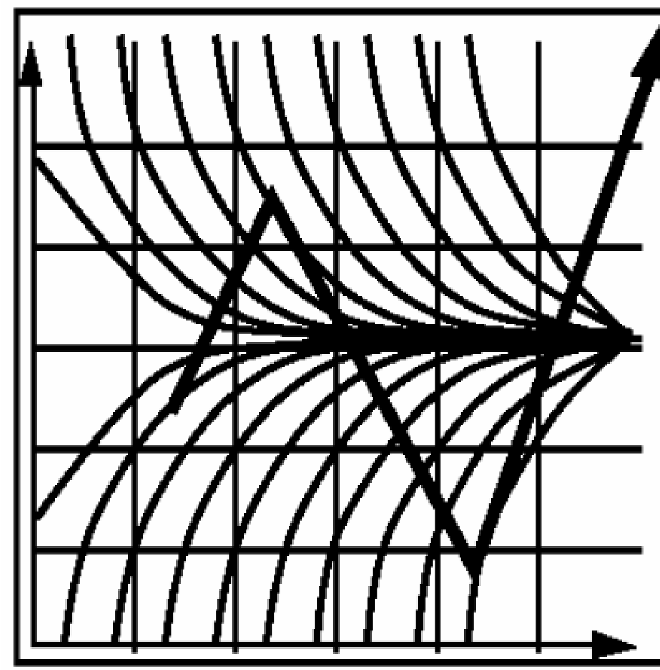
- Exact solution is decaying exponential:

$$x(t) = x_0 e^{-kt}$$

- Limited step size:

$$x_1 = x_0(1 - hk)$$

$$\begin{cases} h \le 1/k & \text{ok} \\ h > 1/k & \text{oscillates} \pm \\ h > 2/k & \text{explodes} \end{cases}$$



- If k is big, h must be small

Analysis: Taylor series

- Expand exact solution **X**(t)

$$\mathbf{X}(t_0+h)=\mathbf{X}(t_0)+h\left(\tfrac{d}{dt}\mathbf{X}(t)\right)\Big|_{t_0}+\tfrac{h^2}{2!}\left(\tfrac{d^2}{dt^2}\mathbf{X}(t)\right)\Big|_{t_0}+\tfrac{h^3}{3!}\left(\cdots\right)+\cdots$$

- Euler's method approximates:

$$\mathbf{X}(t_0+h)=\mathbf{X}_0+h\,f(\mathbf{X}_0,t_0) \qquad \ldots+O(h^2)\,\text{error}$$

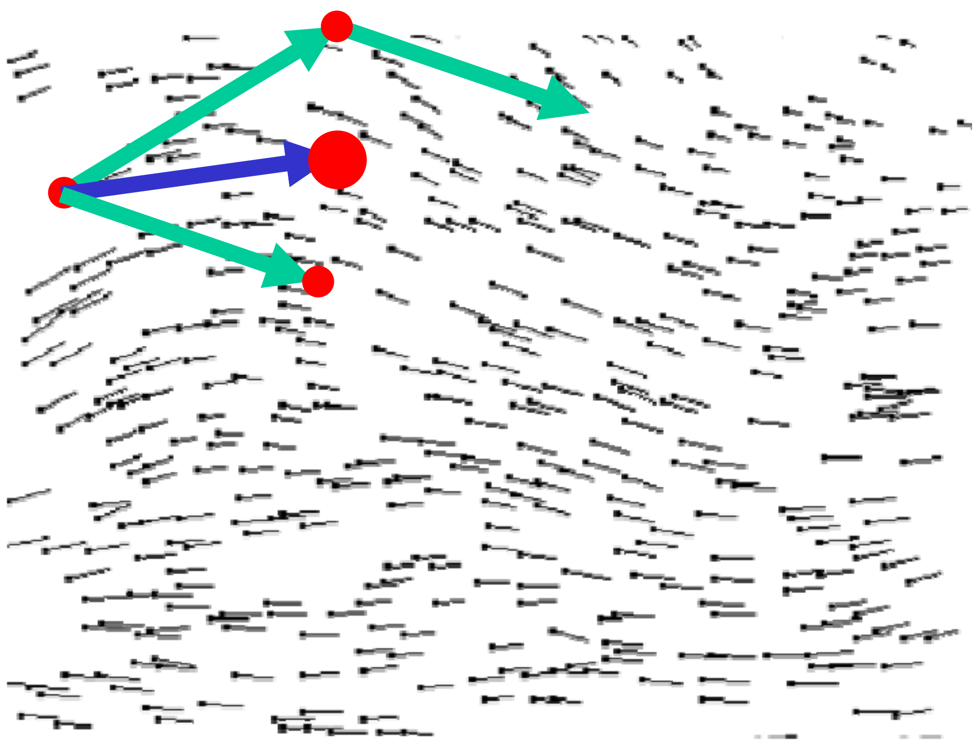
$$h\to h/2 \;\Rightarrow error\to error/4\,\text{per step}\times\text{twice as many steps}$$
$$\to error/2$$

- First-order method: Accuracy varies with h
- To get 100x better accuracy need 100x more steps

Questions?

Can we do better?

- Problem: f has varied along the step
- Idea: look at f at the arrival of the step and compensate for variation



2nd order methods

- Let
$$\boxed{\begin{aligned} f_0 &= f(\mathbf{X}_0, t_0) \\ f_1 &= f(\mathbf{X}_0 + h f_0, t_0 + h) \end{aligned}}$$

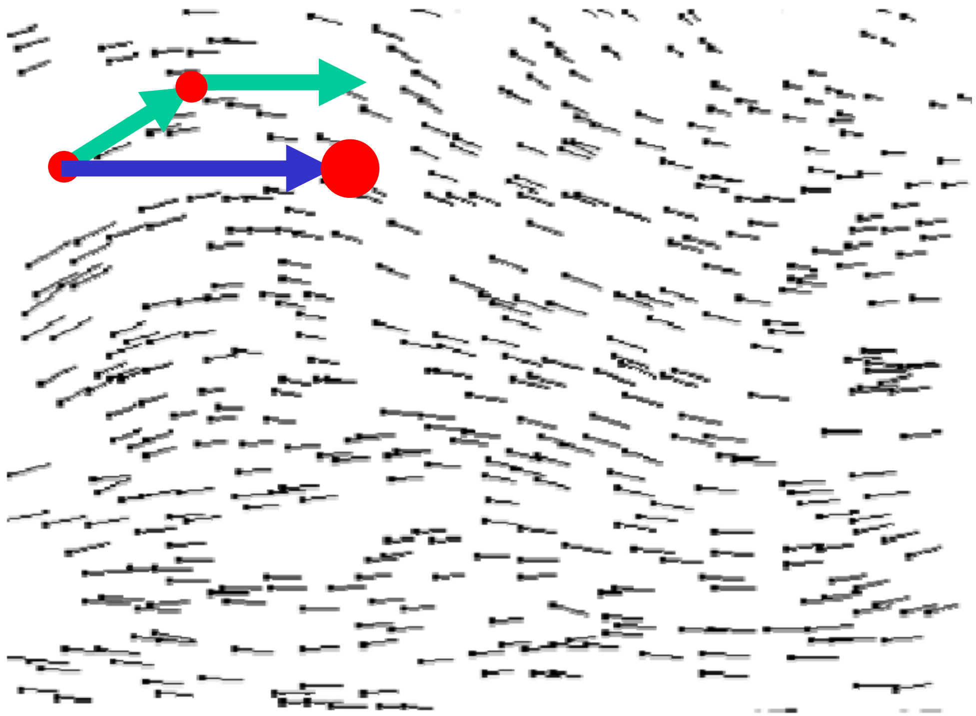
- Then
$$\boxed{\mathbf{X}(t_0 + h) = \mathbf{X}_0 + \tfrac{h}{2}(f_0 + f_1)} + O(h^3)$$

- This is the *trapeziod method*,
- AKA *improved Euler's method*

- Analysis omitted (see 6.839)

Can we do better?

- Problem: f has varied along the step
- Idea: look at f at the arrival of the step and compensate for variation



2nd-order methods continued…

- Could also have chosen

$$\Delta_{\mathbf{X}} = \tfrac{h}{2} f(\mathbf{X}_0, t_0) \text{ and } \Delta_t = \tfrac{h}{2}$$

- then rearrange the same way, let

$$\boxed{\begin{aligned} f_0 &= f(\mathbf{X}_0, t_0) \\ f_m &= f(\mathbf{X}_0 + \tfrac{h}{2} f_0, t_0 + \tfrac{h}{2}) \end{aligned}}$$

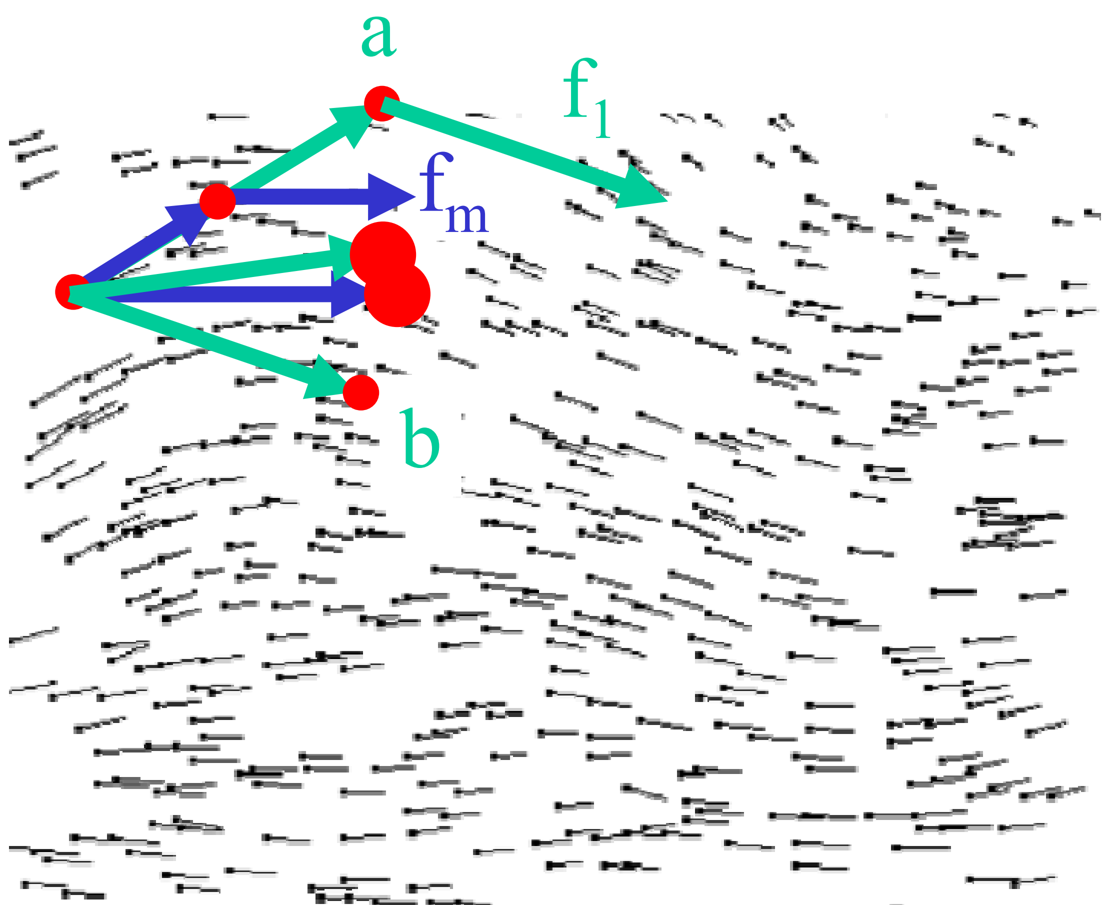
- and get

$$\boxed{\mathbf{X}(t_0 + h) = \mathbf{X}_0 + h\, f_m} + O(h^3)$$

- This is the *midpoint method*

Comparison

- Midpoint:
  - ½ Euler step
  - evaluate f_m
  - full step using f_m
- Trapezoid:
  - Euler step (a)
  - evaluate f_1
  - full step using f_1 (b)
  - average (a) and (b)
- Not exactly same result
- Same order of accuracy



Questions?

Overview

- Generate tons of particles
- Describe the external forces with a force field
- Integrate the laws of mechanics
  - Lots of differential equations ;-(

Done!

- Each particle is described by its state
  - Position, velocity, color, mass, lifetime, shape, etc.
- More advanced versions exist: flocks, crowds

Particle Animation

```
AnimateParticles(n, y0, t0, tf)
{
  y = y0
  t = t0
  DrawParticles(n, y)
  while(t != tf) {
   f = ComputeForces(y, t)
   dydt = AssembleDerivative(y, f)
      //there could be multiple force fields
   {y, t } = ODESolverStep(6n, y, dy/dt)
   DrawParticles(n, y)
  }
}
```

What is a force?

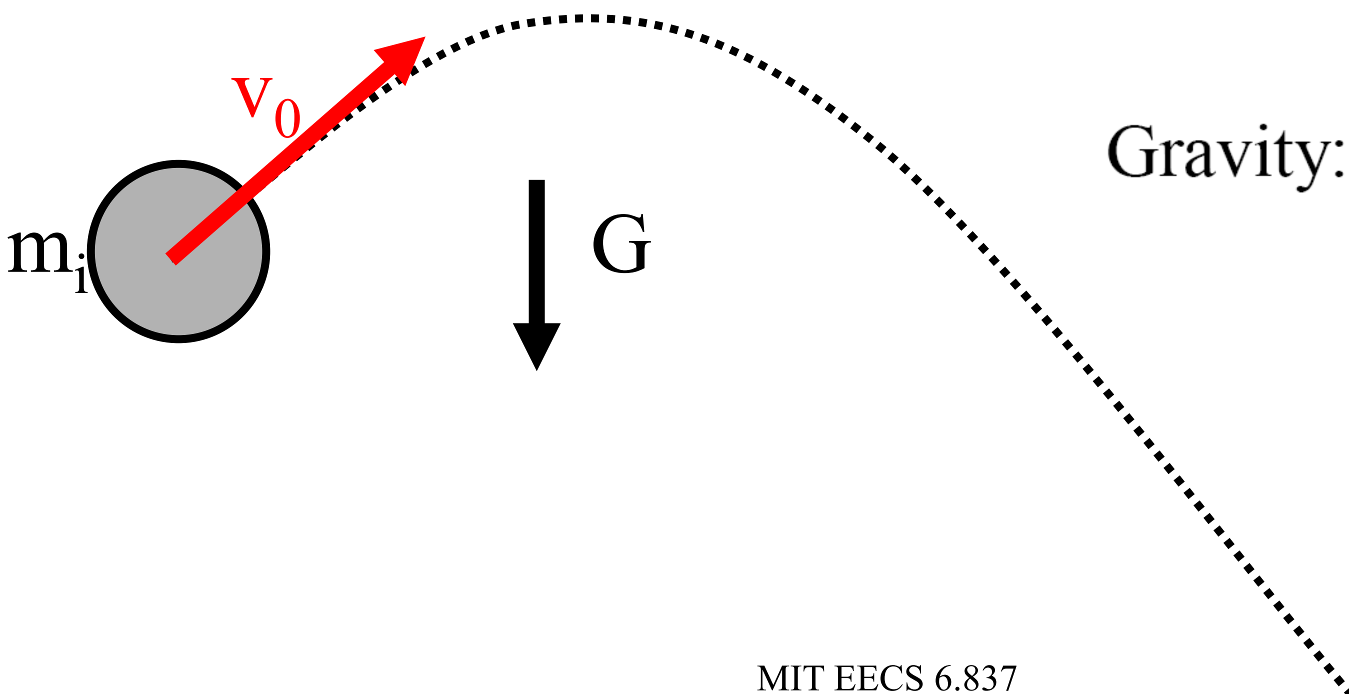
- Forces can depend on location, time, velocity

Implementation:

- **Force** a class
  - Computes force function for each particle **p**
  - Adds computed force to total in **p.f**
- There can be multiple force sources

Forces: gravity on Earth

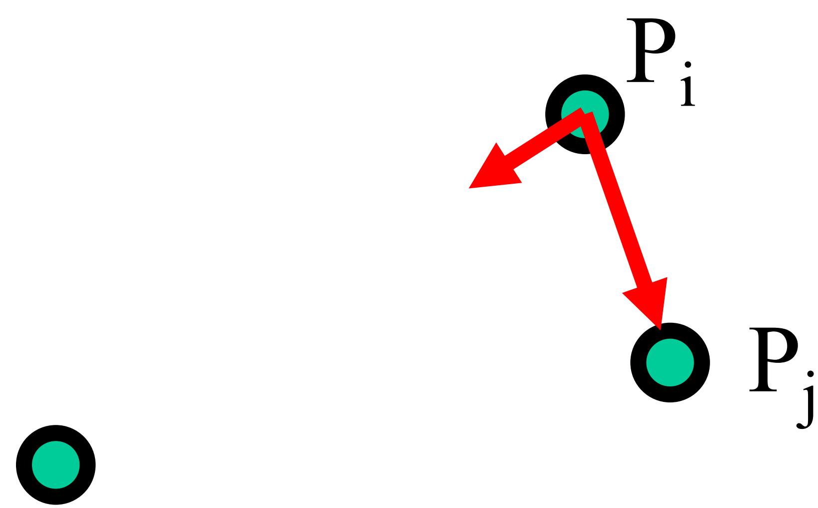
- depends only on particle mass:
- $f(\mathbf{X},t)$ = constant
- for smoke, flame: make gravity point up!



$$\text{Gravity: } f^{(i)} = \begin{pmatrix} 0 \\ 0 \\ -m_i G \end{pmatrix}$$

Forces gravity for N-body problem

- Depends on all other particles
- Opposite for pairs of particles
- Force in the direction of $p_i p_j$ with magnitude inversely proportional to square distance
- $F_{ij} = G\, m_i m_j / r^2$



Forces: damping

$$f^{(i)} = -dv^{(i)}$$

- force on particle i depends only on velocity of I
- force opposes motion
- removes energy, so system can settle
- small amount of damping can stabilize solver
- too much damping makes motion like in glue

Forces: spatial fields

Spatial fields: $f^{(i)} = f(x^{(i)}, t)$

- force on particle i depends only on position of i
- arbitrary functions:
  - wind
  - attractors
  - repulsers
  - vortexes
- can depend on time
- note: these add energy, may need damping, so

$$f^{(i)} = f(x^{(i)}, v^{(i)}, t)$$

Forces: spatial interaction

$$\text{Spatial interaction: } f^{(i)} = \sum_j f(x^{(i)}, x^{(j)})$$

- e.g., approximate fluid: Lennard-Jones force:

$$f(x^{(i)}, x^{(j)}) = \frac{k_1}{\left|x^{(i)} - x^{(j)}\right|^m} - \frac{k_2}{\left|x^{(i)} - x^{(j)}\right|^n}$$

- $O(N^2)$ to test all pairs
  - usually only local
  - Use buckets to optimize. Cf. 6.839

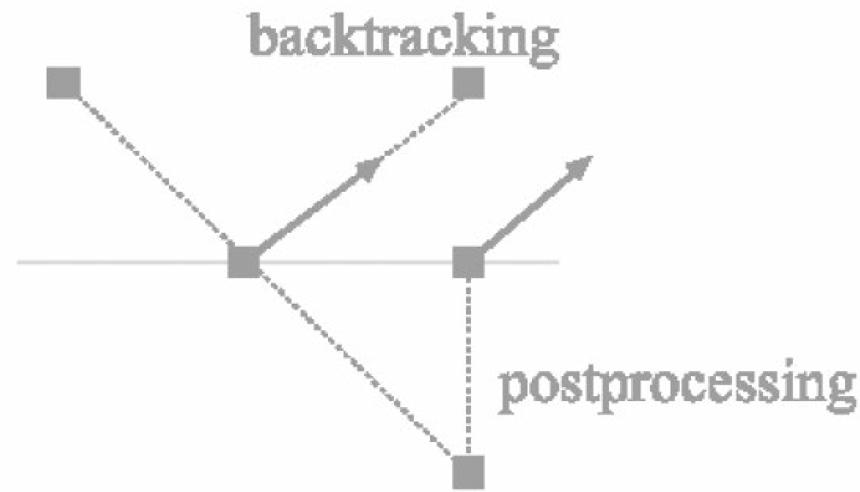
Computing f(X,t)

```
getDerivative(X,t)
  for each p in Particles {
    p.setState(X)
    p.f = 0
  }
  for each f in Forces {
    f.apply(t) // adds to each p.f
  }
  for each p in Particles {
    p.computeDerivative(F)
  }
  return F
```

Questions?

Collisions

- Usually don't test particle-particle collision
- Test collision with environment, e.g. ground
- Note: step overshoots collision
  - Test for penetration
  - back up or fixup

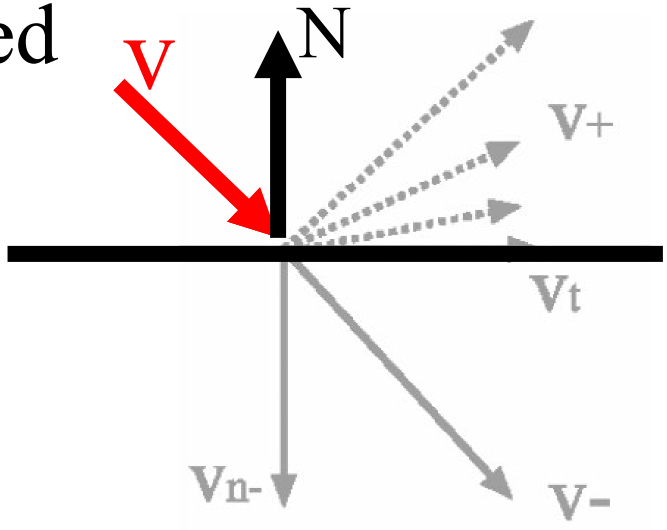


Handling collisions

- tangential velocity v_b unchanged
- normal velocity v_n reflects:

$$v = v_t + v_n$$

$$v \leftarrow v_t - \varepsilon v_n$$



- coefficient of restitution
- change of velocity = $-(1+\epsilon)v$
- change of momentum *Impulse* = $-m(1+\epsilon)v$
- Remember mirror reflection? Can be seen as photon particles

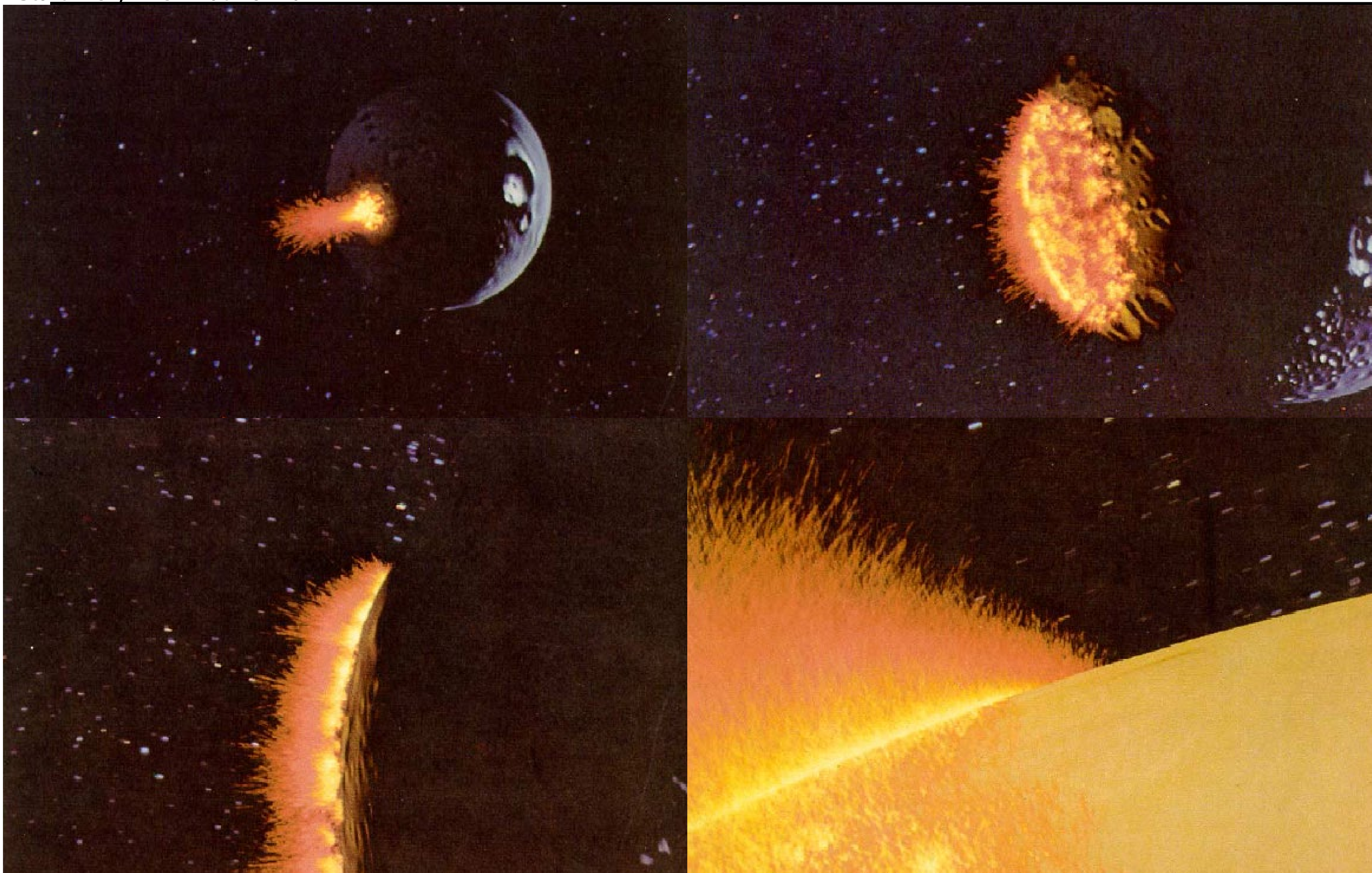
Detecting collisions

- Easy with implicit equations of surfaces
- $H(x,y,z)=0$ at surface
- $H(x,y,z)<0$ inside surface
- So just compute H and you know that you're inside if it's negative

- More complex with other surface definitions

Particle Animation [Reeves et al. 1983]

Start Trek, The Wrath of Kahn



Additional references

- http://www.cse.ohio-state.edu/~parent/book/outline.html
- http://www.pixar.com/companyinfo/research/pbm2001/
- http://www.cs.unc.edu/~davemc/Particle/