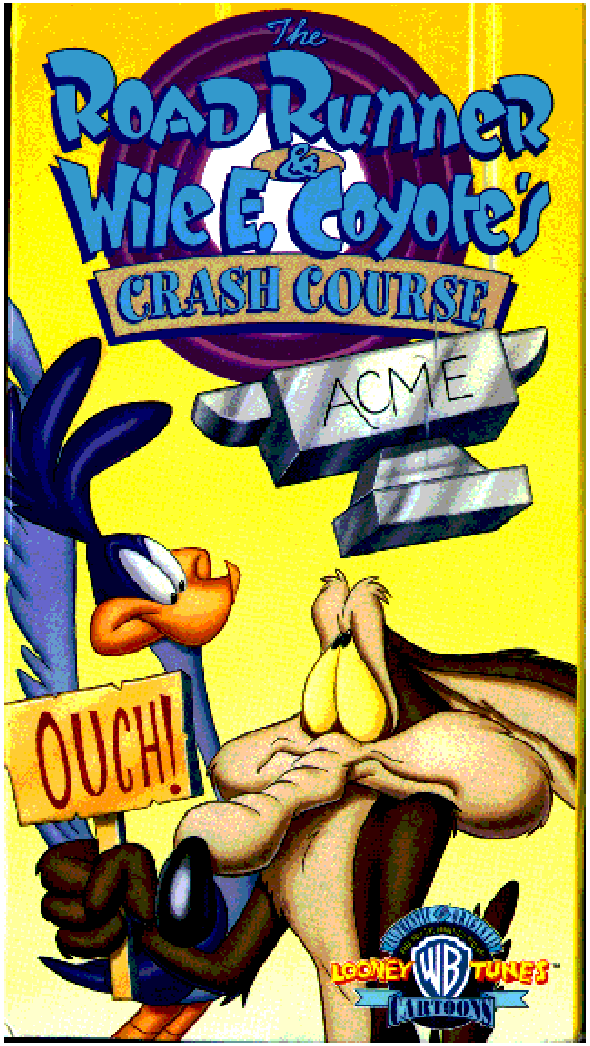
Computer Animation
Particle systems
& mass-spring



- Some slides courtesy of Jovan Popovic, Ronen Barzel

Last time?

- Animation
  - Keyframe, procedural, physically-based, motion capture
- Particle systems
  - Generate tons of points
  - Force field
- ODE integration
  - Take small step in the direction of derivatives
  - Euler $O(h)$, midpoint and trapezoid $O(h^2)$

Assignment 10

- Proposal due tomorrow
- Assignment due Dec 3

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

What is a particle system?

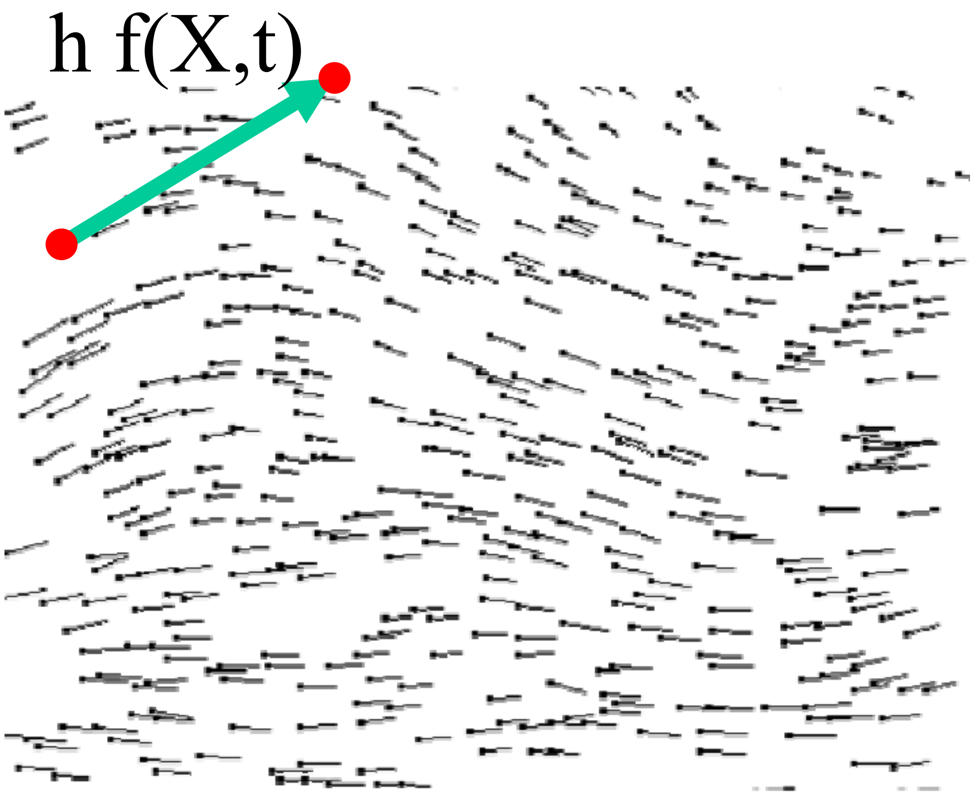
- Collection of many small simple particles
- Particle motion influenced by force fields
- Particles created by *generators*
- Particles often have *lifetimes*
- Used for, e.g:
  - sand, dust, smoke, sparks, flame, water, …

For a collection of 3D particles…

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

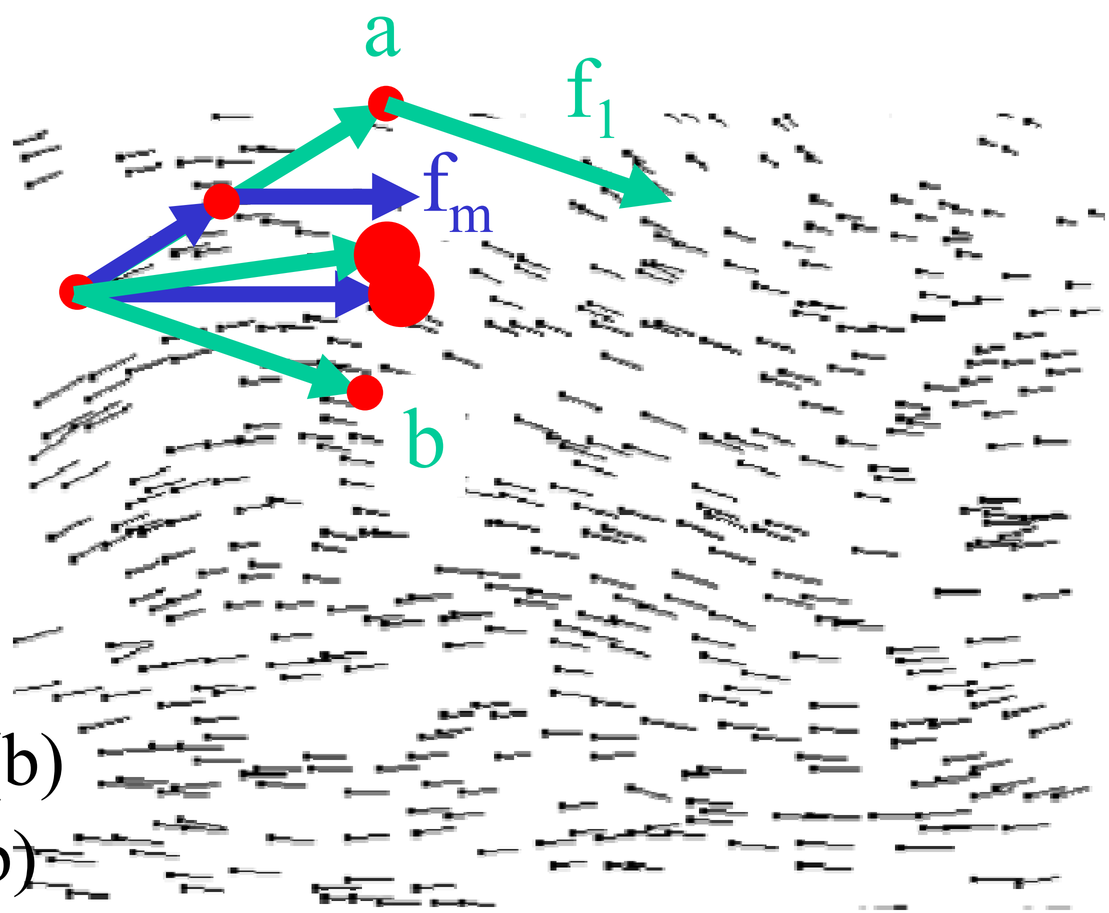
Euler

- Timestep h, move in the direction of f(X, t)



2^{nd} order methods

- 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)
- Both $O(h^2)$



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?

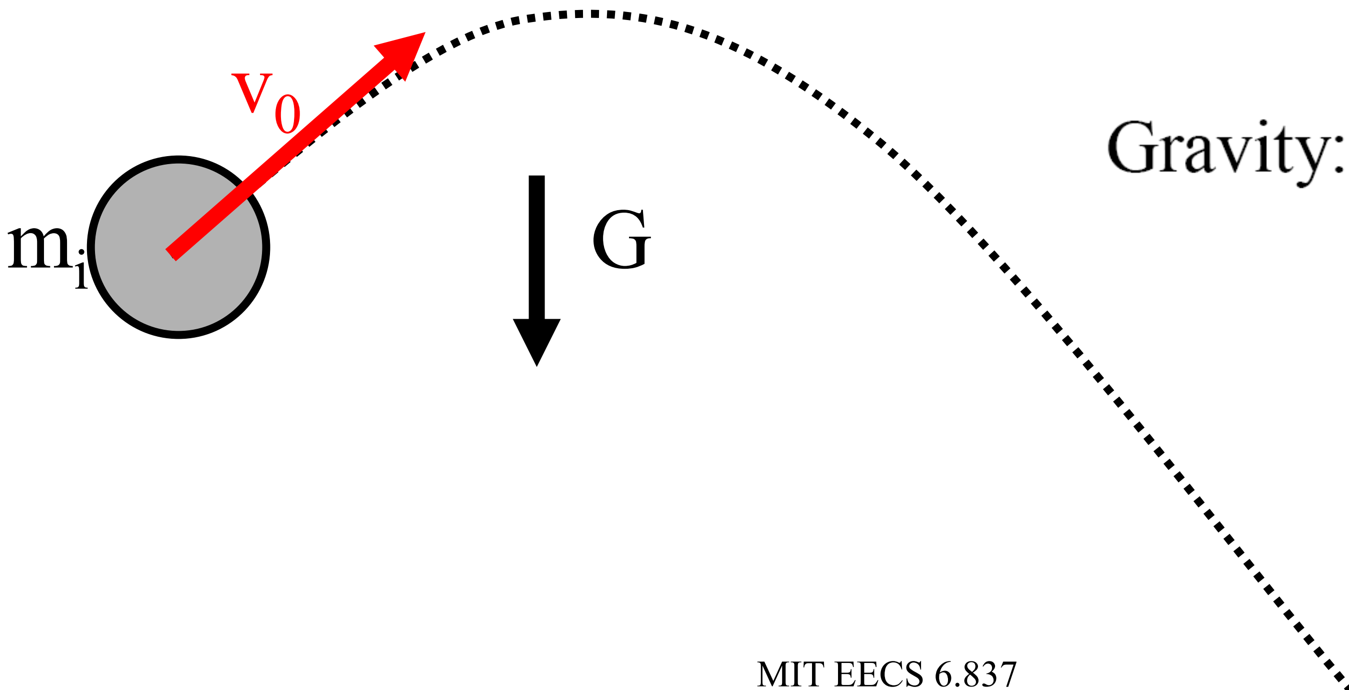
- Forces can depend on location, time, velocity

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

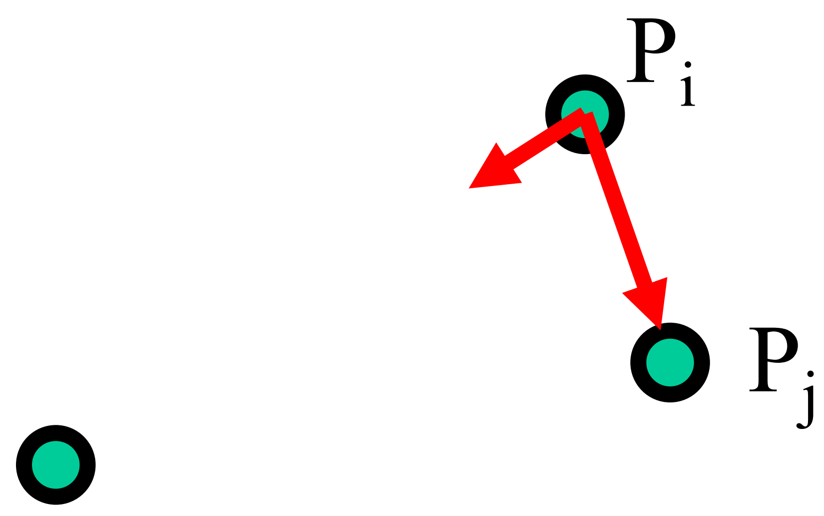
- depends only on particle mass:
- $f(\mathbf{X},t)$ = constant
- for smoke, flame: make gravity point up!



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

Forces gravity for N-body problem

- Depends on all other particles
- Opposite for pairs of particles
- Force in the direction of p_ip_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: $f^{(i)} = f(x^{(i)}, t)$

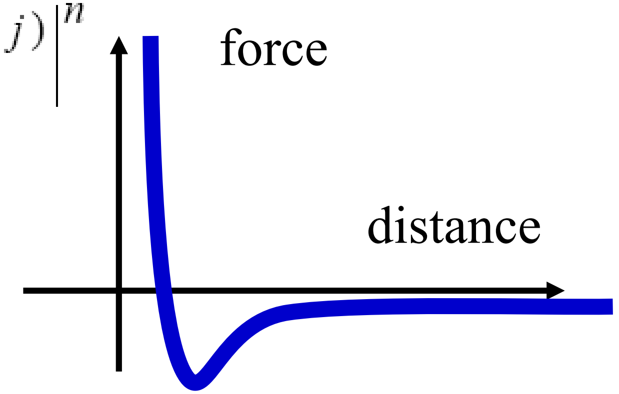
- force on particle i depends only on position of i
- arbitrary functions:
  - wind
  - attractors
  - repulsers
  - vortexes
- can depend on time
- note: these add energy, may need damping

Forces: spatial interaction

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

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

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

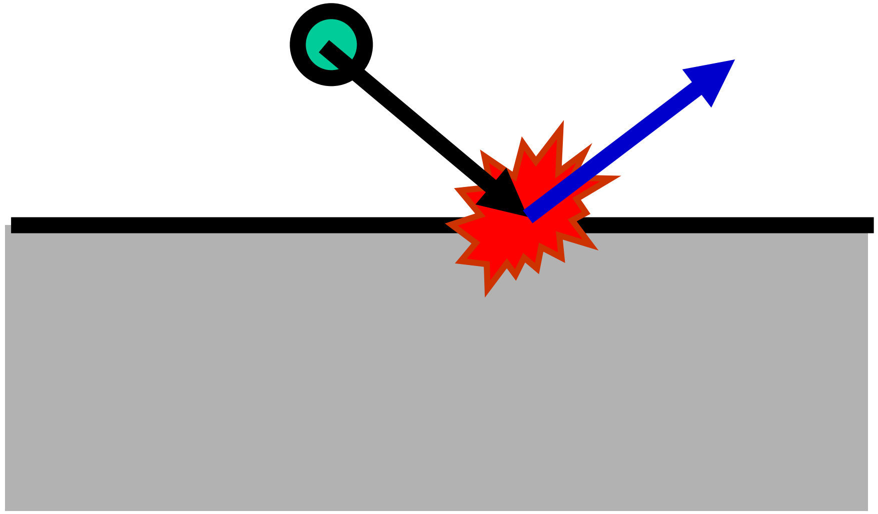


- Repulsive + attractive force
- $O(N^2)$ to test all pairs
  - usually only local
  - Use buckets to optimize. Cf. 6.839

Questions?

Collisions

- Detection
- Response
- Overshooting problem (when we enter the solid)



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

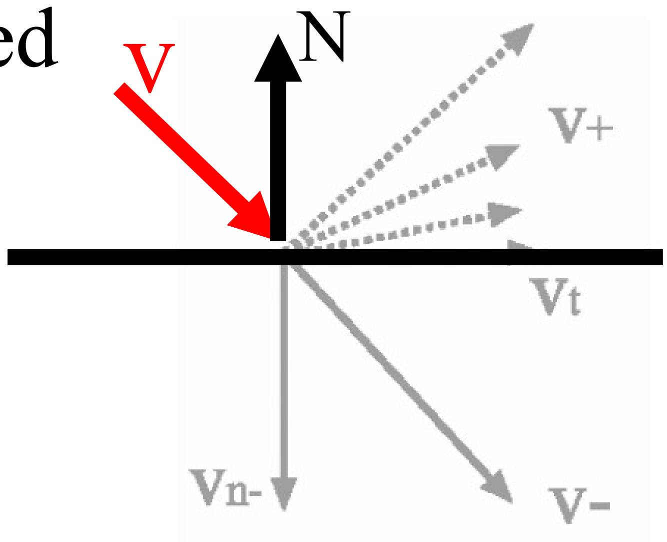
- More complex with other surface definitions

Collision response

- tangential velocity v_b unchanged
- normal velocity v_n reflects:

$$v = v_t + v_n$$

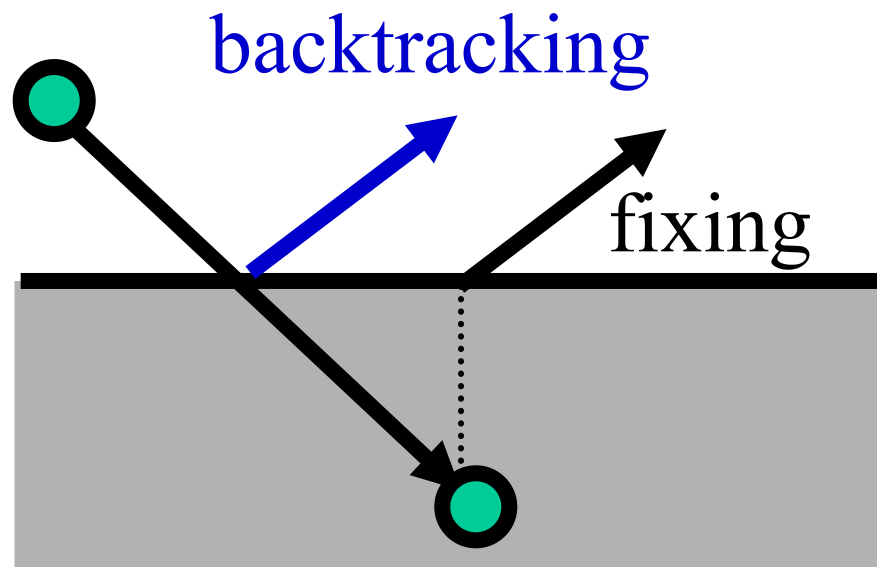
$$v \leftarrow v_t - \varepsilon v_n$$



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

Collisions - overshooting

- Usually, we detect collision when it's too late: we're already inside
- Solutions: back up
    - Compute intersection point
    - Ray-object intersection!
    - Compute response there
    - Advance for remaining fractional time step
- Other solution: Quick and dirty fixup
    - Just project back to object closest point



Questions?

Where do particles come from?

- Often created by *generators* (or "emitters")
  - can be attached to objects in the model
- Given rate of creation: particles/second
  - record t_{last} of last particle created

$$n = \lfloor (t - t_{last})\, rate \rfloor$$

  - create n particles.
    update t_{last} if $n > 0$
- Create with (random) distribution of initial x and v
  - if creating $n > 1$ particles at once, spread out on path

Particle lifetimes

- Record time of “birth” for each particle
- Specify lifetime
- Use particle age to:
  - remove particles from system when too old
  - often, change color
  - often, change transparency (old particles fade)
- Sometimes also remove particles that are offscreen

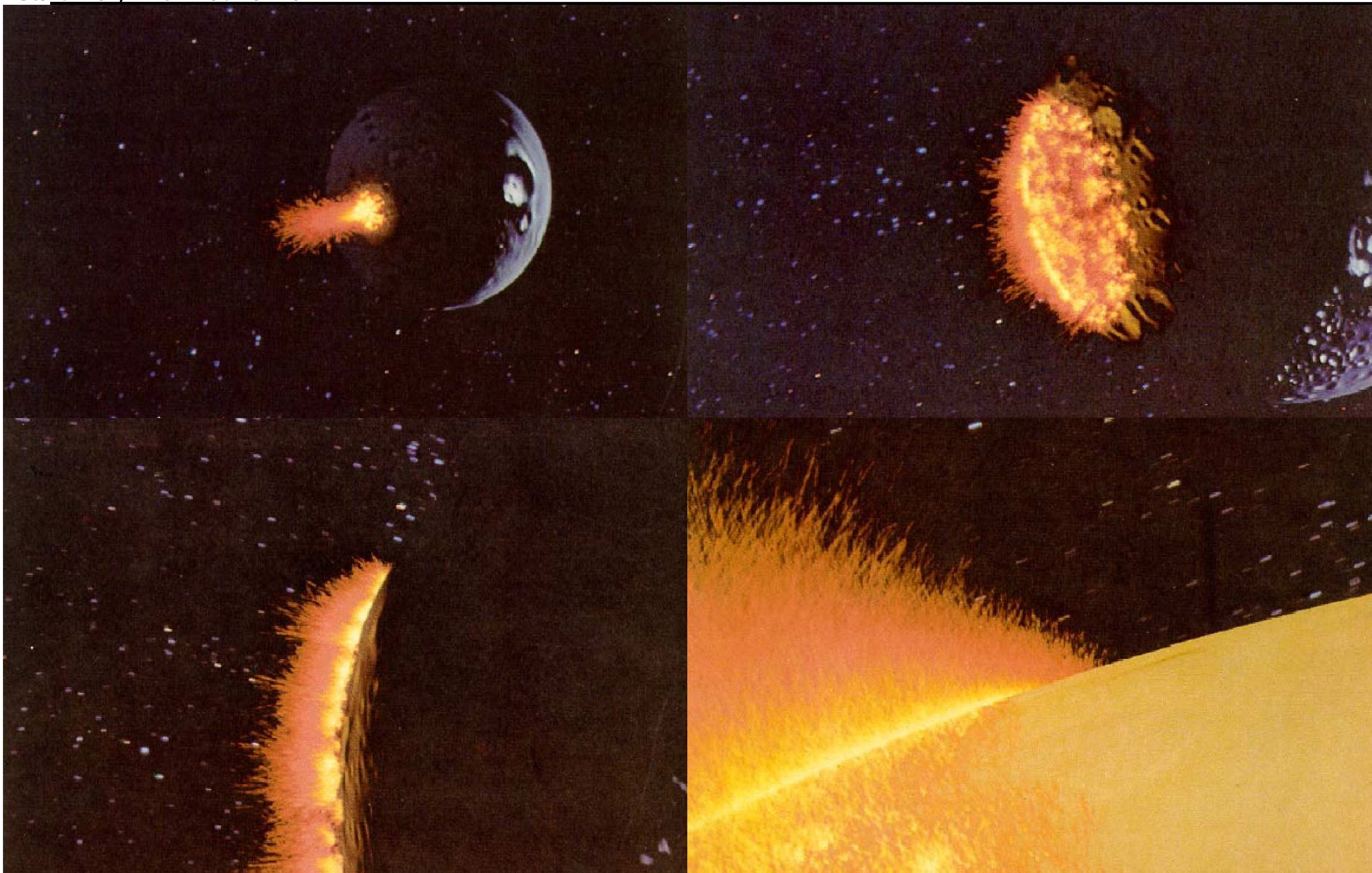
Rendering and motion blur

- Particles are usually not shaded (just emission)
- Often, they don't contribute to the z-buffer (rendered last with z-buffer disabled)
- Draw a line for motion blur
  - (x, x+vdt)
- Sometimes use texture maps (fire, clouds)

Questions?

Particle Animation [Reeves et al. 1983]

Start Trek, The Wrath of Kahn



How they did it?

- One big particle system at impact
- Secondary systems for rings of fire.

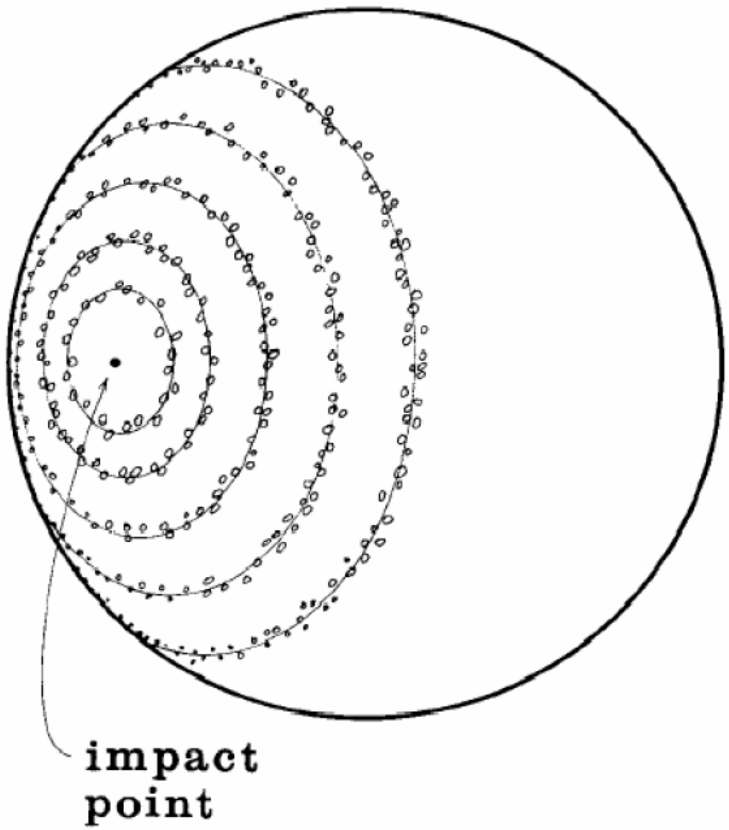
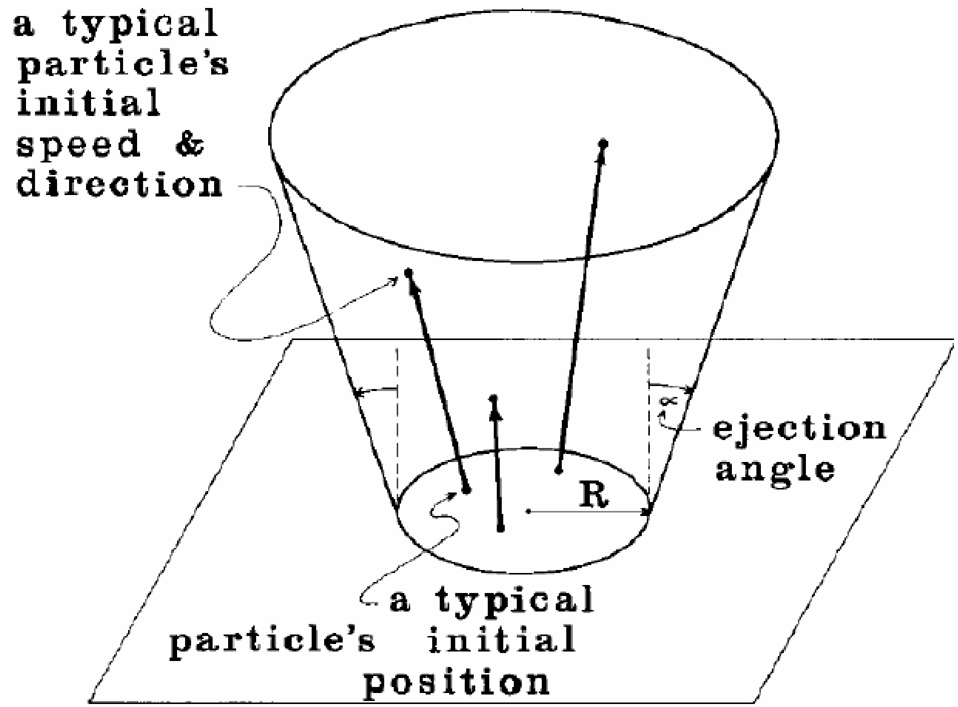


Fig. 2. Distribution of particle systems on the planet's surface.



Particle Modeling [Reeves et al. 1983]

- The grass is made of particles

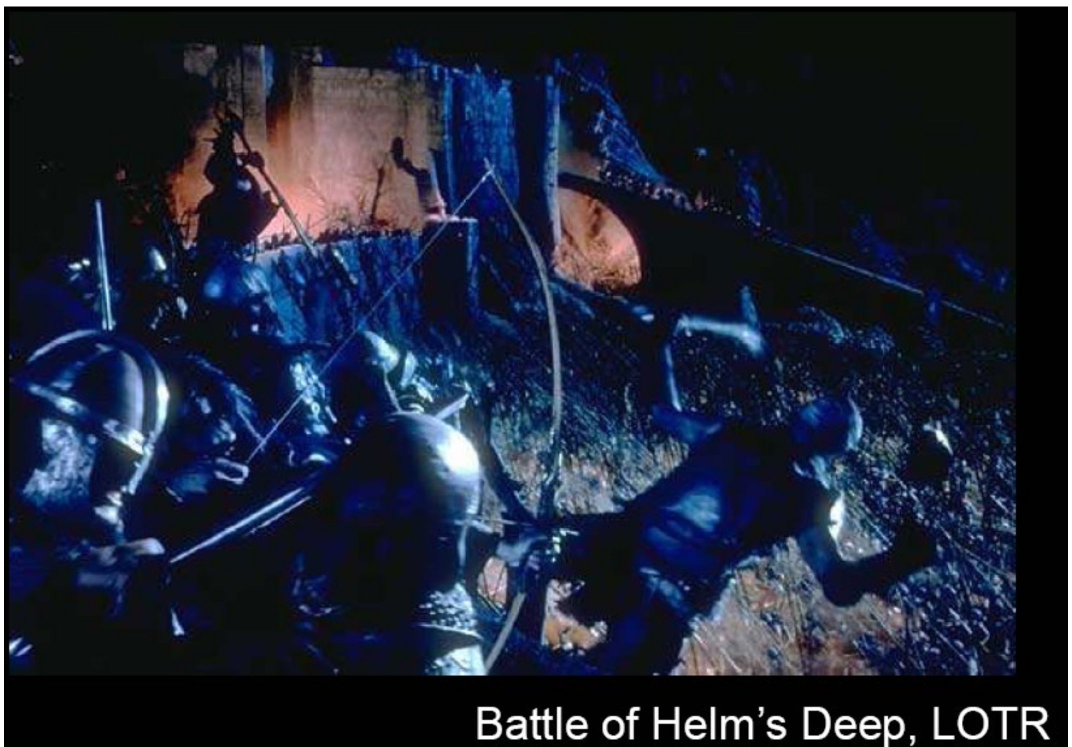
Other uses of particles

- Water
  - E.g. splashes in lord of the ring river scene
- Explosions in games
- Grass modeling
- Snow, rain
- Screen savers

Questions?

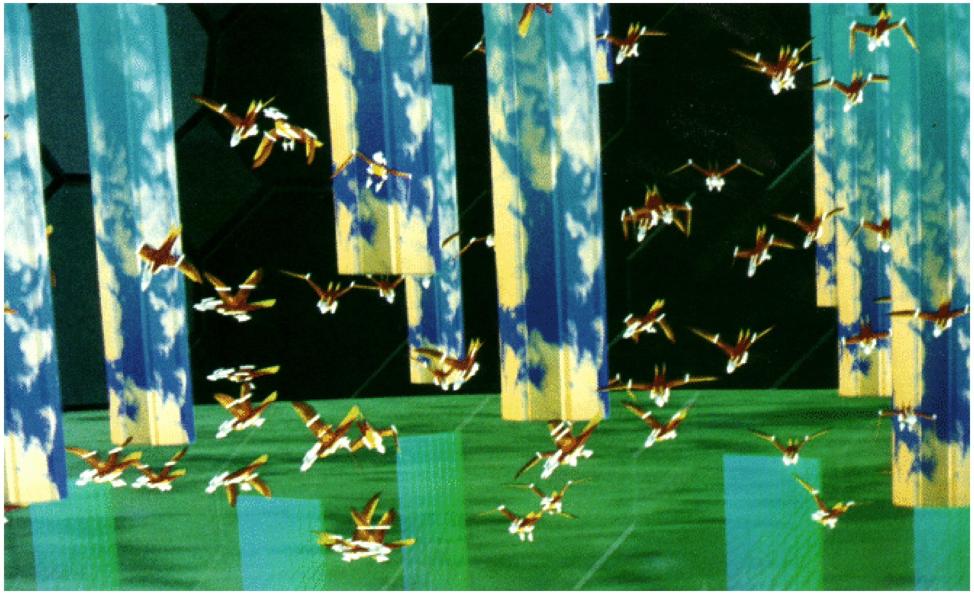
More advanced version

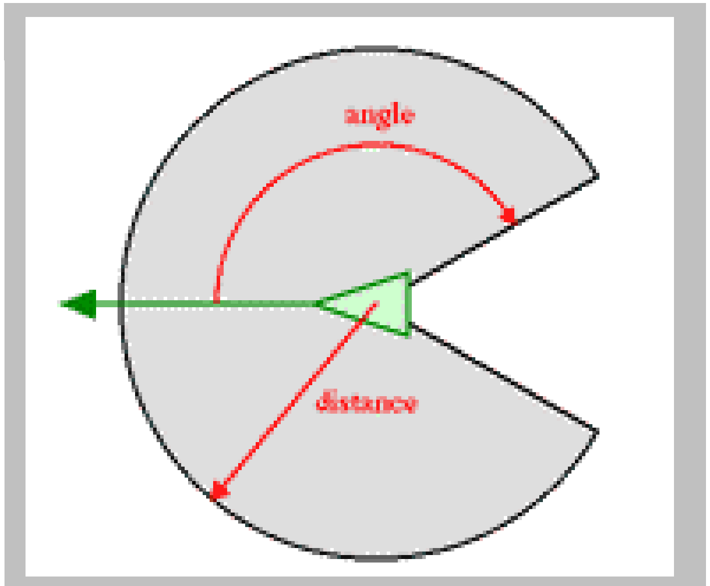
- Flocking birds, fish school
  - http://www.red3d.com/cwr/boids/
- Crowds



Flocks

- From Craig Reynolds
- Each bird modeled as a complex particle (“boid”)
- A set of forces control its behavior
- Based on location of other birds and control forces

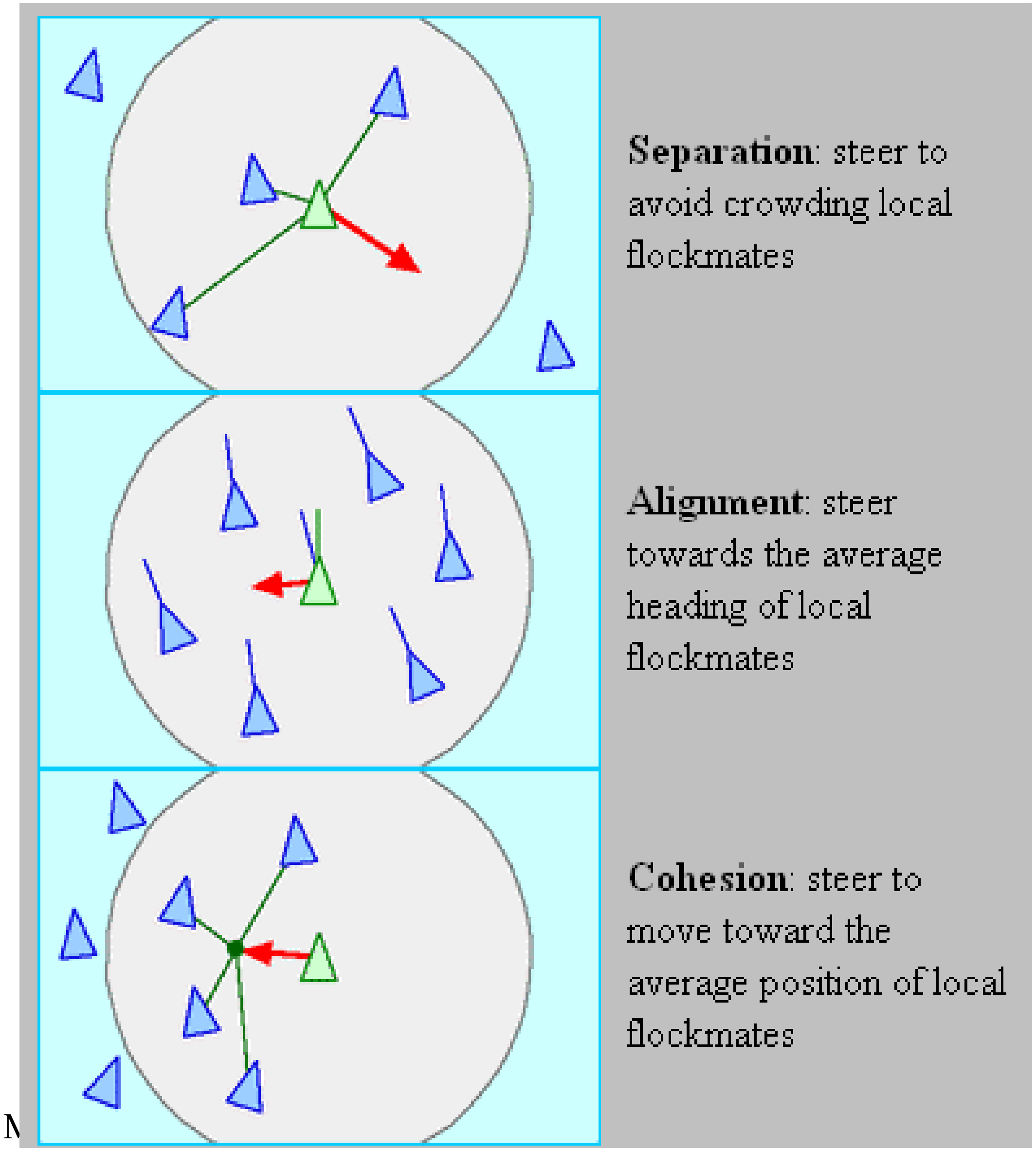




a boid's neighborhood

Flocks

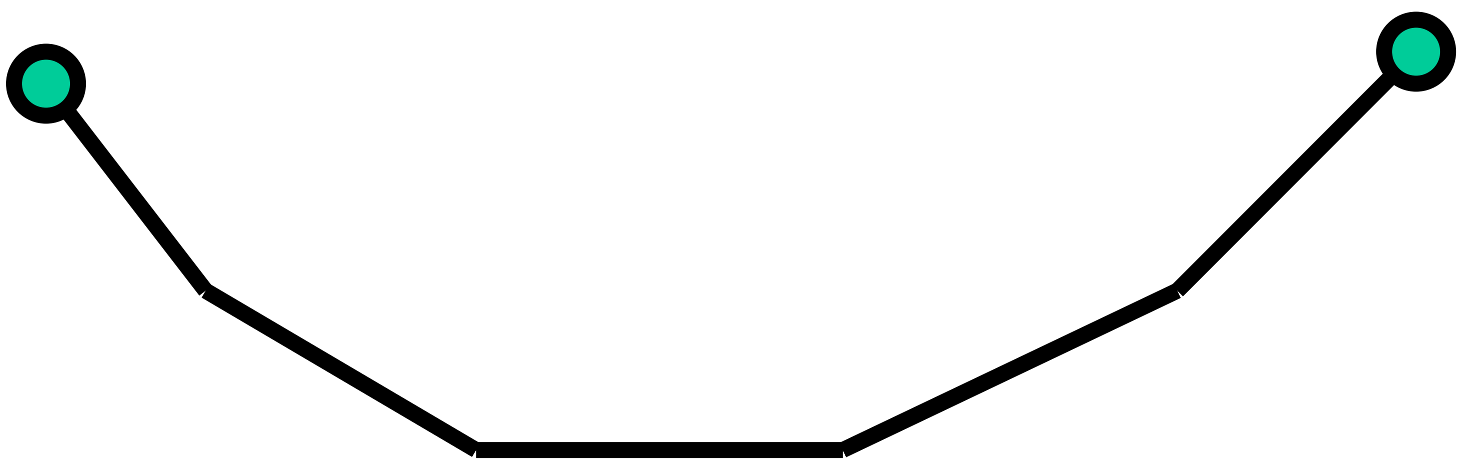
- From
  Craig Reynolds
- “Boid" was an abbreviation of "birdoid", as his rules applied equally to simulated flocking birds, and schooling fish.



Questions?

How would you simulate a string?

- Each particle is linked to two particles
- Forces try to keep the distance between particles constant
- What force?

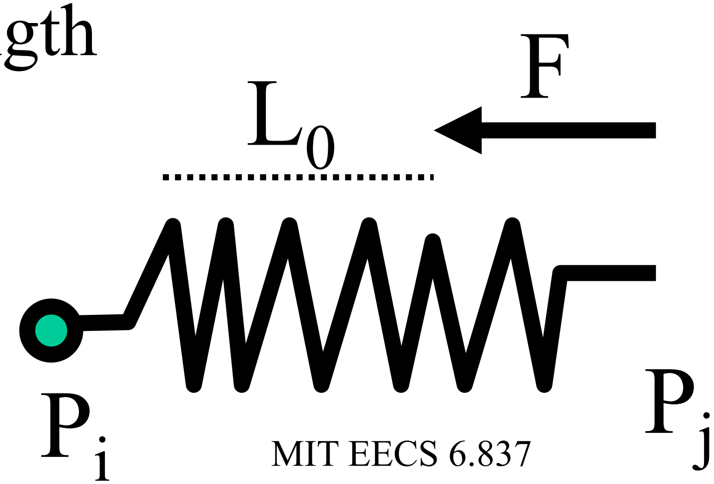


Spring forces

- Force in the direction of the spring and proportional to difference with rest length

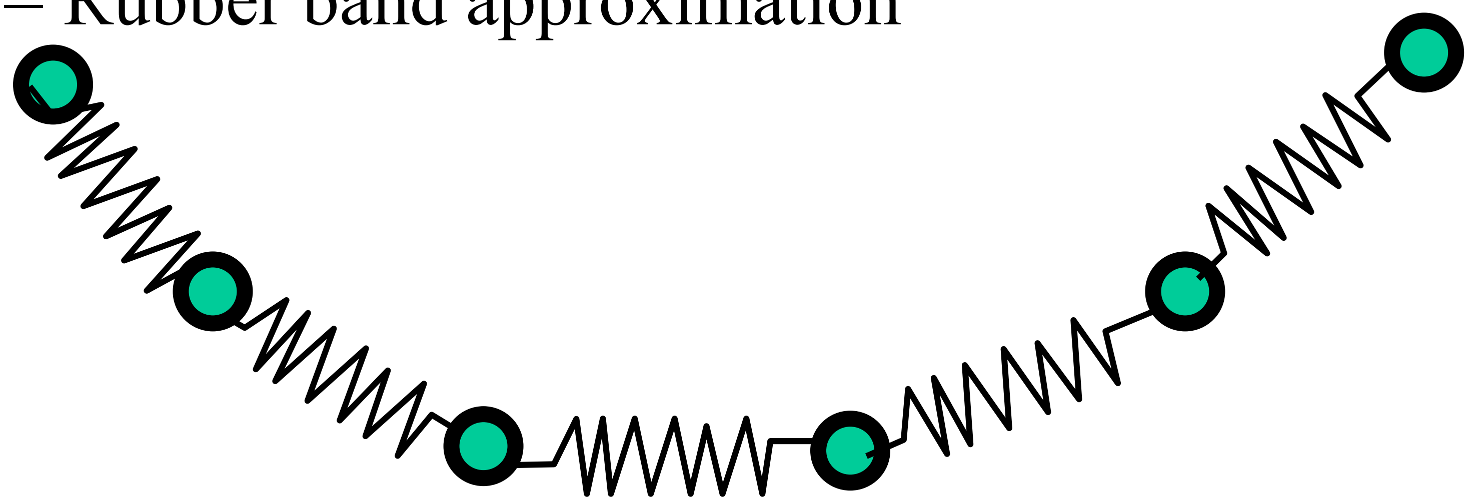
$$F(P_i, P_j) = K(L_0 - ||\vec{P_iP_j}||)\frac{\vec{P_iP_j}}{||\vec{P_iP_j}||}$$

- K is the stiffness of the spring
  - When K gets bigger, the spring really wants to keep its rest length



How would you simulate a string?

- Springs link the particles
- Springs try to keep their rest lengths and preserve the length of the string
- Not exactly preserved though, and we get numerical oscillation
  - Rubber band approximation



Questions?

Mass-spring

- Interaction between particles
- Create a network of spring forces that link pairs of particles
- Used for strings, clothes, hair

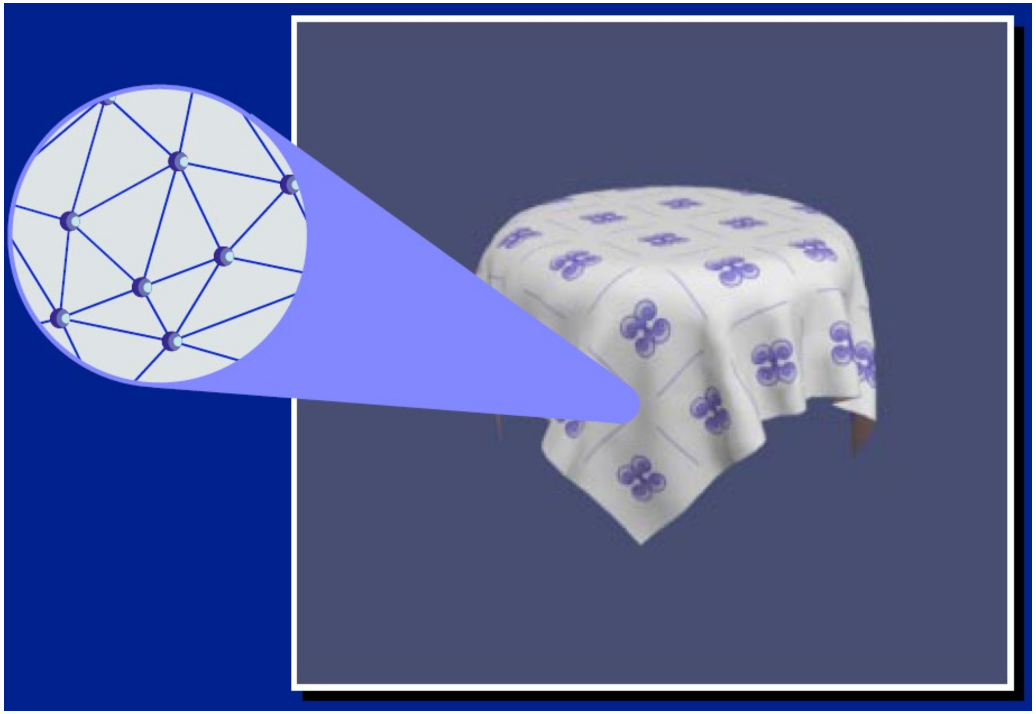


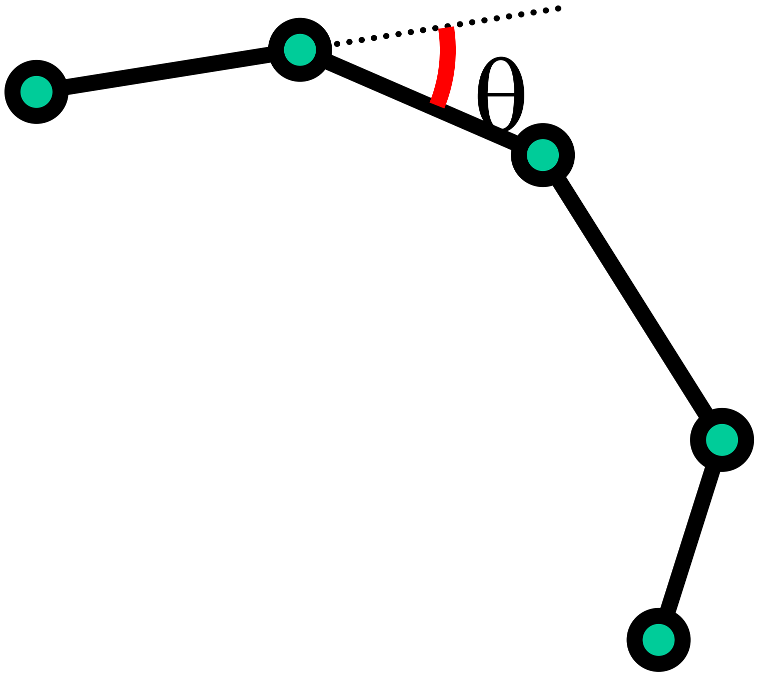
Image Michael Kass

Three types of forces

- Structural forces
  - Try to enforce invariant properties of the system
  - E.g. force the distance between two particles to be constant
  - Ideally, these should be constraints, not forces
- Internal Deformation forces
  - E.g. a string deforms, a spring board tries to remain flat
- External forces
  - Gravity, etc.

Hair

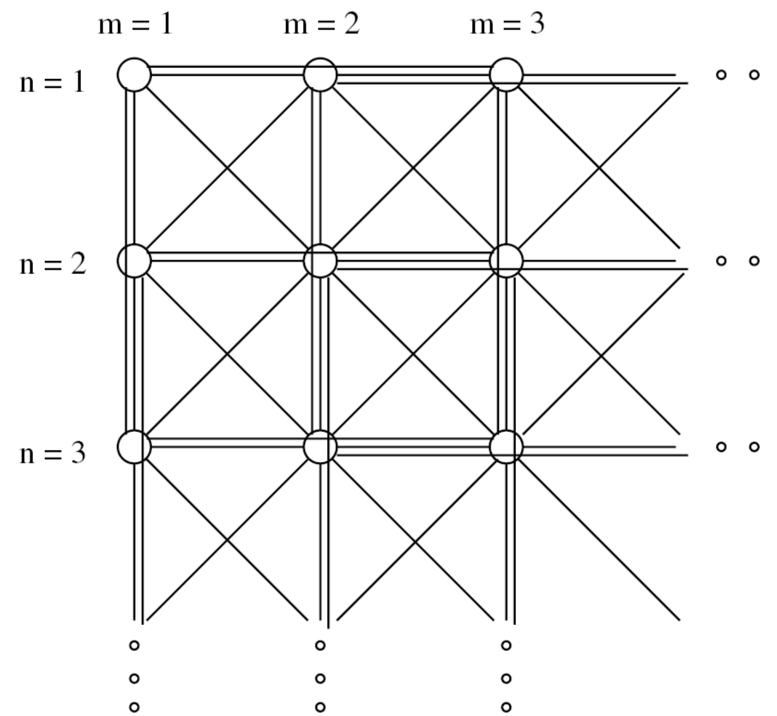
- Linear set of particles
- Length structural force
- Deformation forces proportional to the angle between segments



Questions?

Cloth using mass-spring

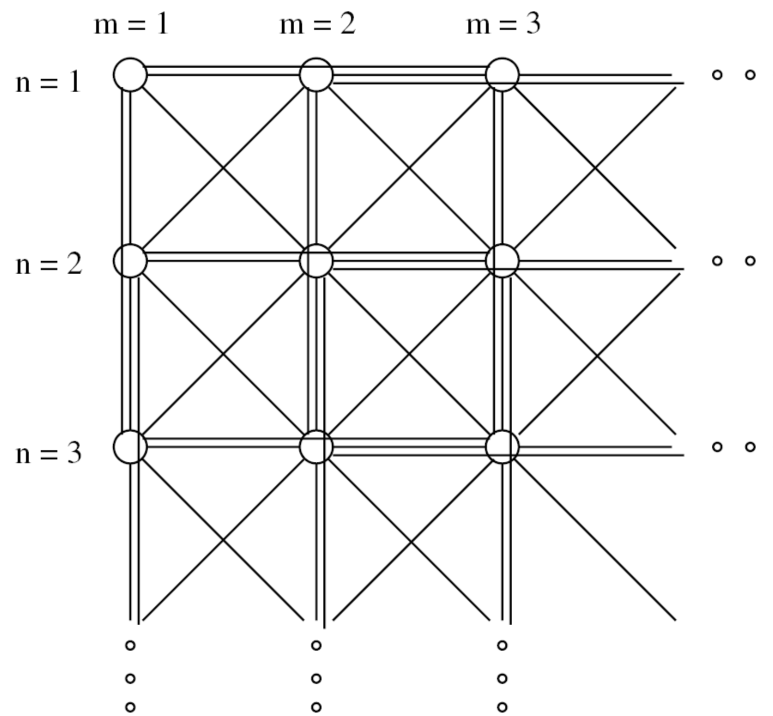
- Network of masses and springs
- Structural springs:
  - link (i j) and (i+1, j); and (i, j) and (i, j +1)
- Shear springs
  - (i j) and (i+1, j+1)
  - masses i! j and i j ! will be referred to
- Flexion springs
  - (i,j) and (i+2,j); (i,j) and (i,j+2)



From Provost 95

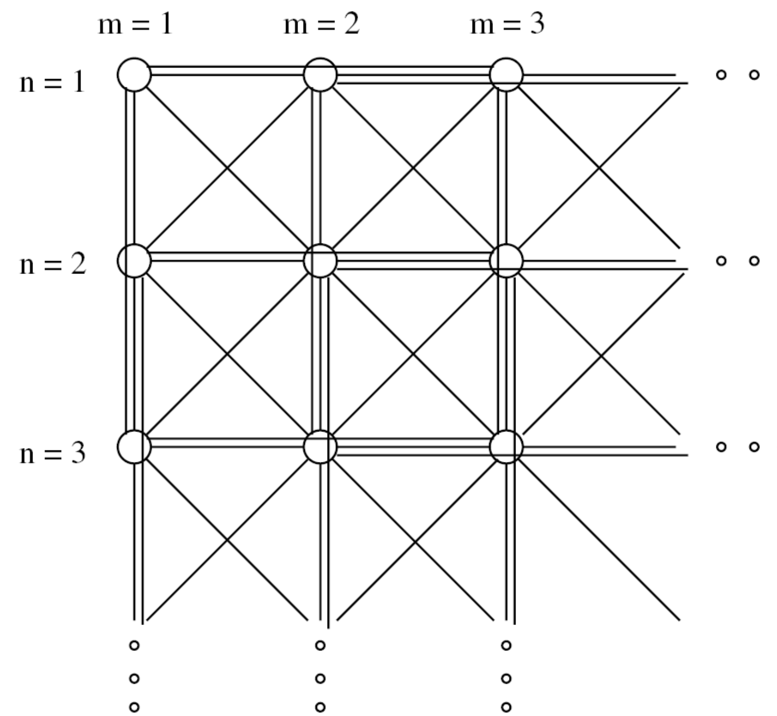
External forces

- Gravity Gm
- Viscous damping Cv
- Wind, etc.



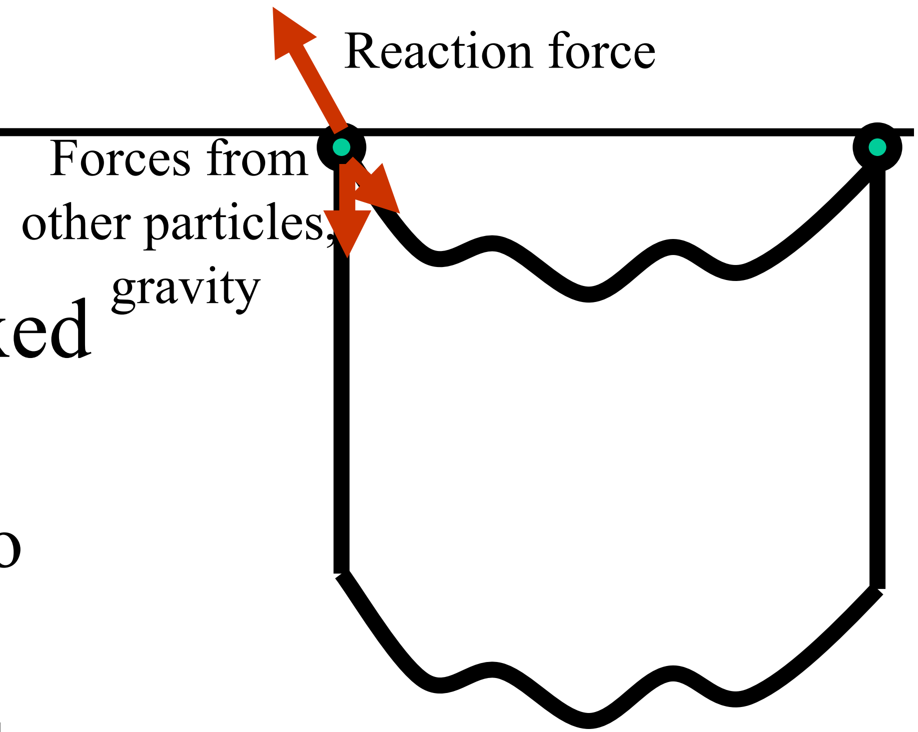
Cloth simulation

- Then, the all trick is to set the stiffness of all springs to get realistic motion!
- Remember that forces depend on other particles (coupled system)
- But it is sparse (only neighbors)

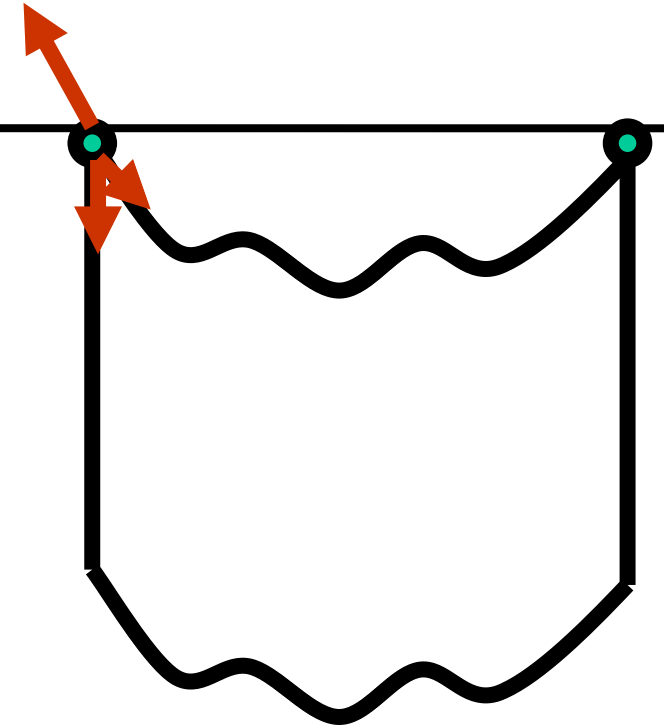


Contact forces

- Hanging curtain:
- 2 contact points stay fixed
- What does it mean?
  - Sum of the forces is zero
- How so?
  - Because those point undergo an external force that balances the system
- What is the force at the contact?
  - Depends on all other forces in the system
  - Gravity, wind, etc.



Contact forces



- How can we compute the external contact force?
  - Inverse dynamics!
  - Sum all other forces applied to point
  - Take negative
- Do we really need to compute this force?
  - Not really, just ignore the other forces applied to this point!

Example

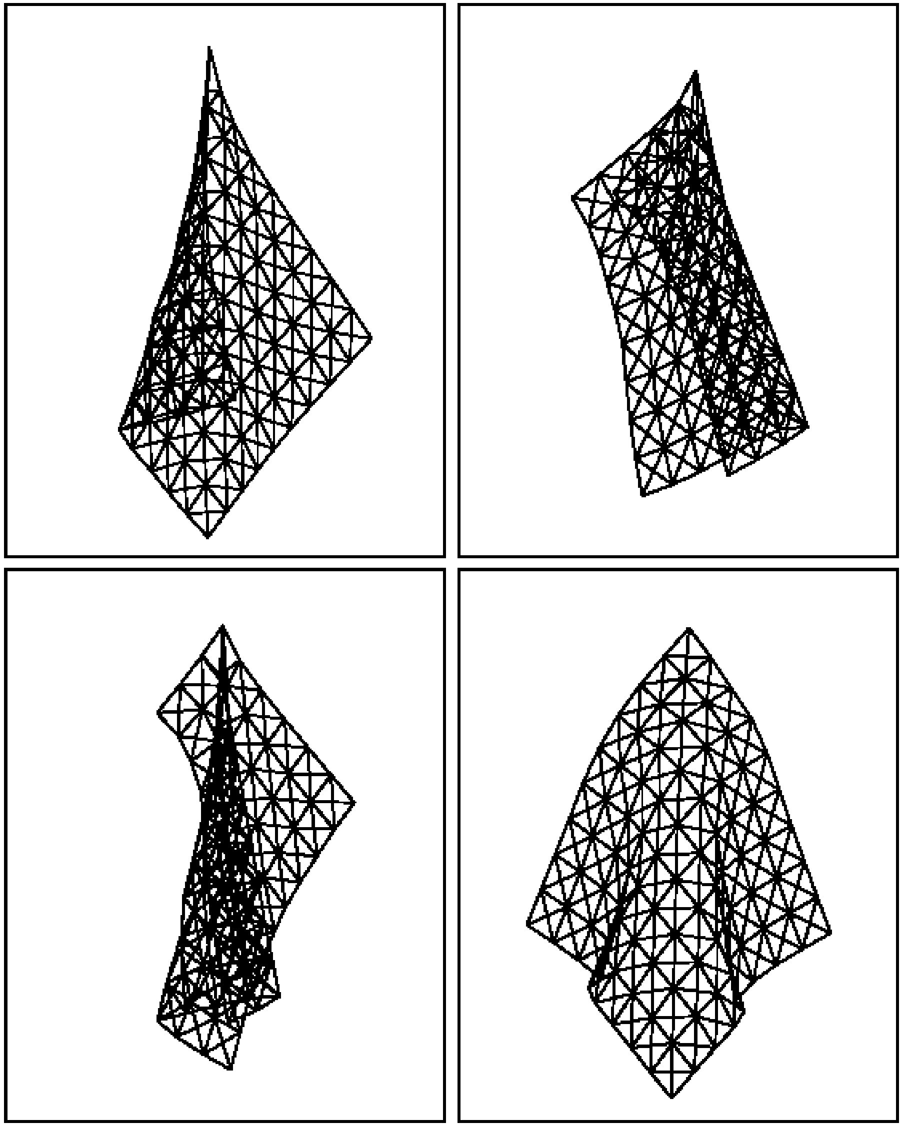
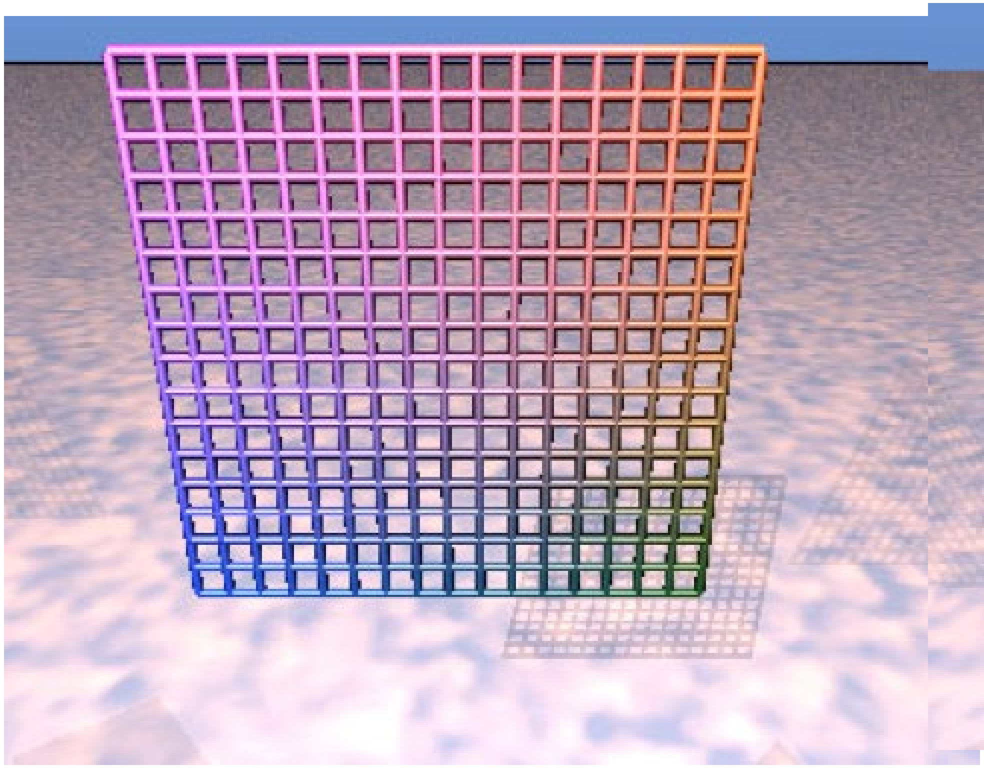


Image from Meyer et al. 2001

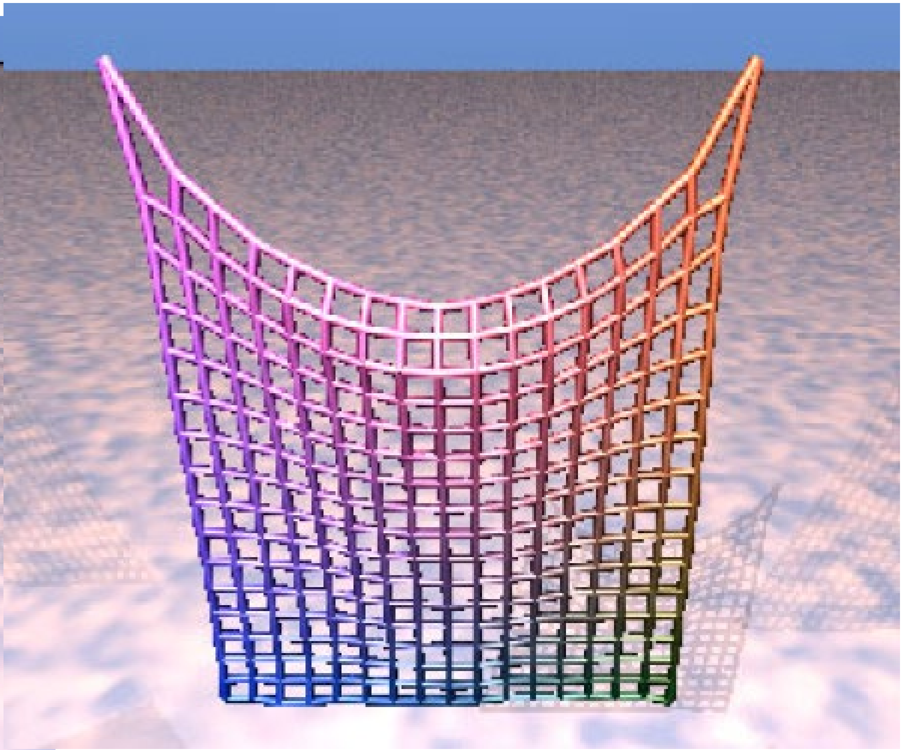
Questions?

Example

- Excessive deformation:
  the strings are not stiff enough



Initial position



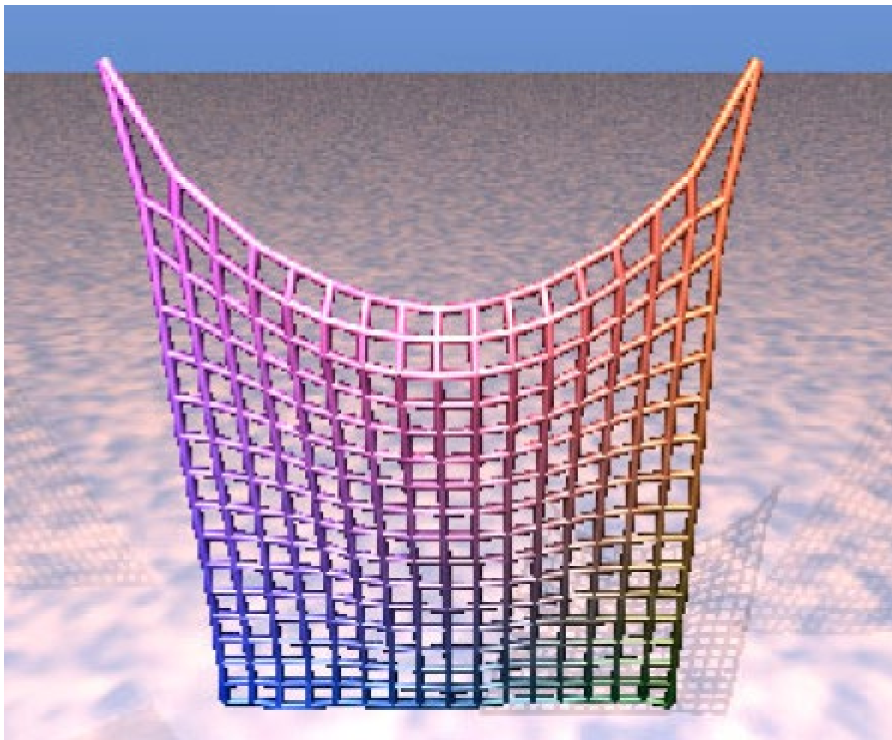
After 200 iterations

The stiffness issue

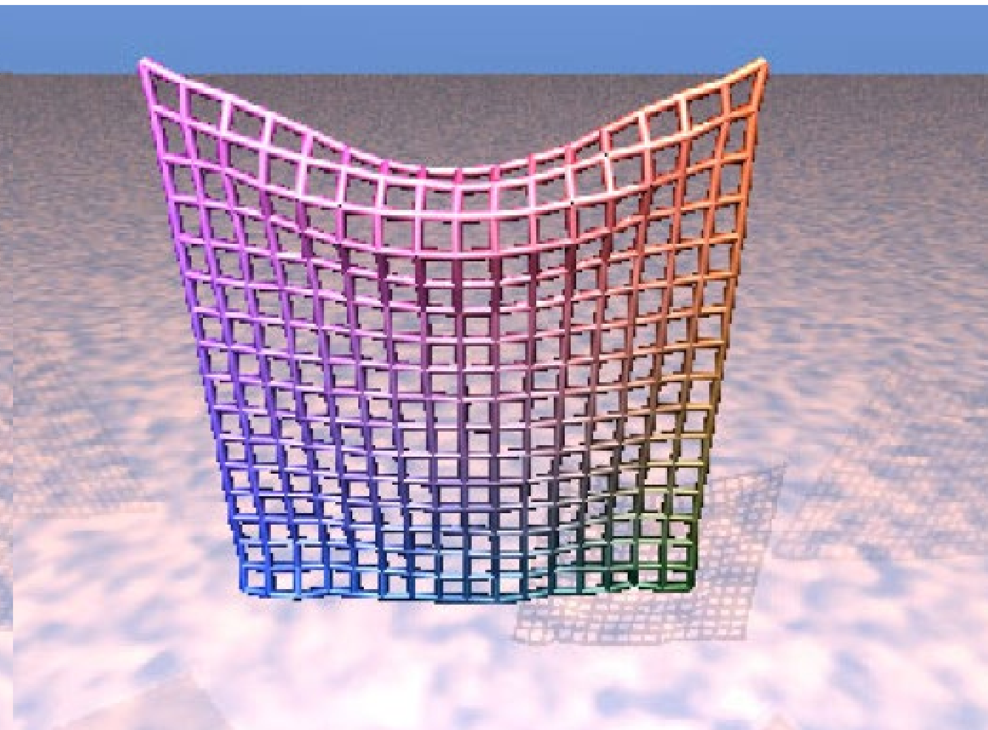
- We use springs while we mean constraint
  - Spring should be super stiff, which requires tiny Δt
  - remember $x' = -kx$ system
- Even though clothes are a little elastic, they usually don't deform more than 10%
- Many numerical solutions
  - Reduce Δt
  - Actually use constraints
  - Implicit integration scheme (see 6.839)

One solution

- Constrain length to increase by less than 10%



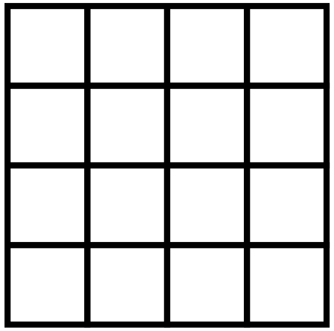
Simple mass-spring system

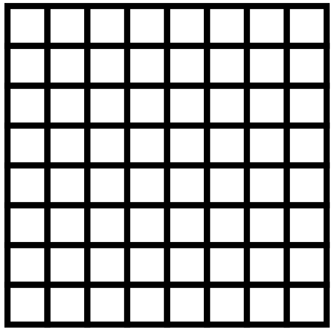


Improved solution
(see Provot Graphics Interface 1995)

The discretization problem

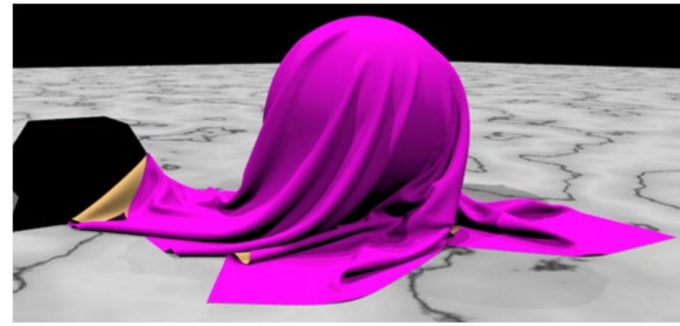
- What happens if we discretize our cloth more finely?
- Do we get the same behavior?
- Usually not! It takes a lot of effort to design a scheme that does not depend on the discretization.

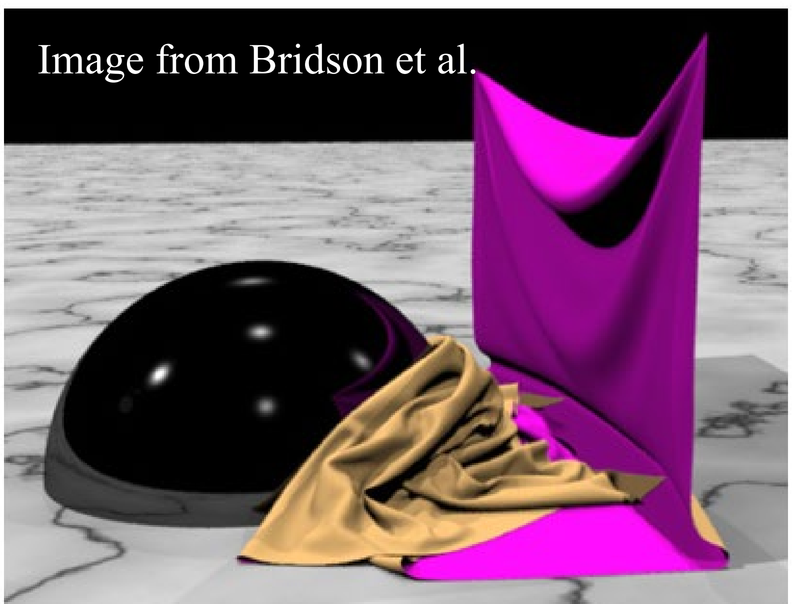




The collision problem

- A cloth has many points of contact
- Stays in contact
- Requires
  - Efficient collision detection
  - Efficient numerical treatment (stability)





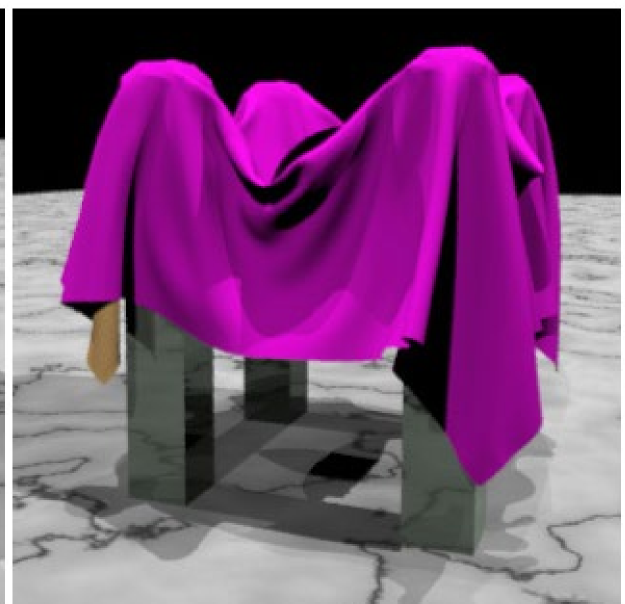


Image from Bridson et al.