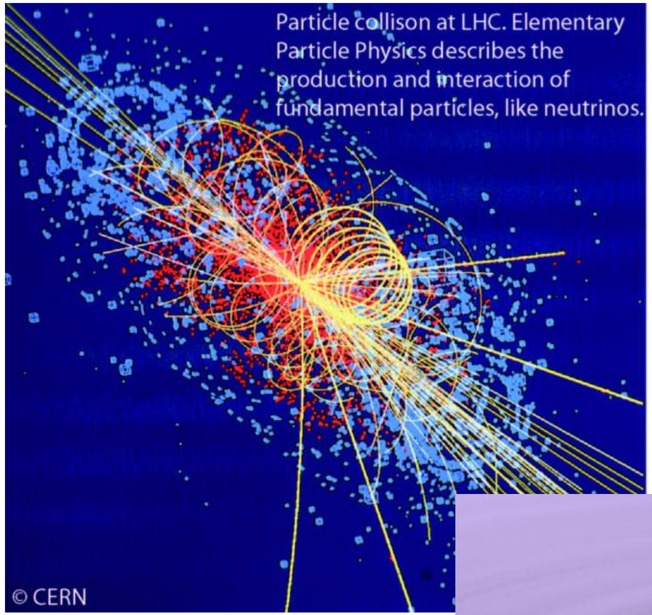
PHYSICALLY BASED ANIMATION

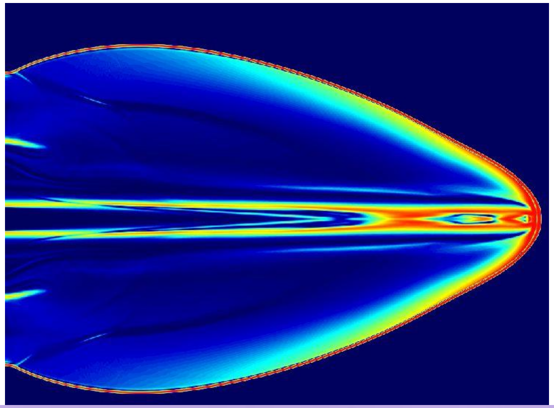
CS148
Introduction to Computer Graphics and Imaging

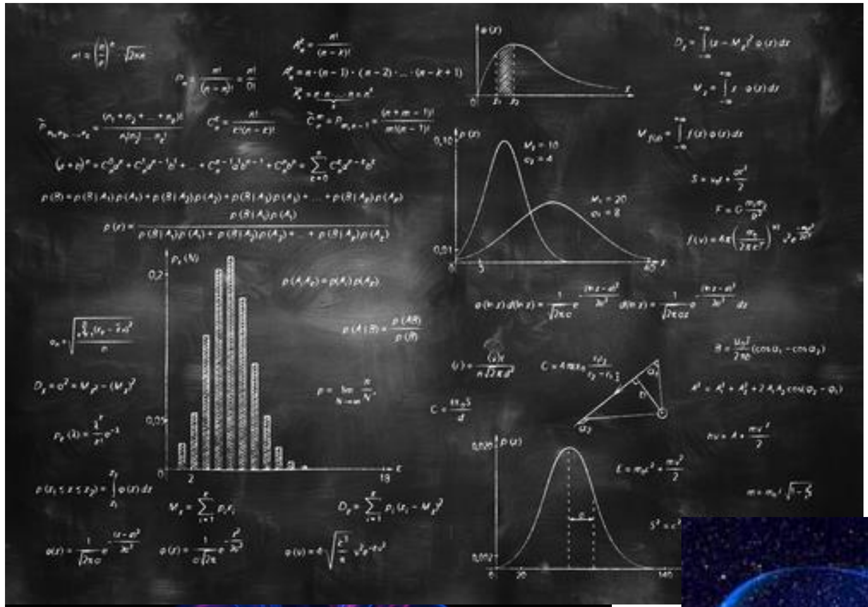
David Hyde

August 2nd, 2016

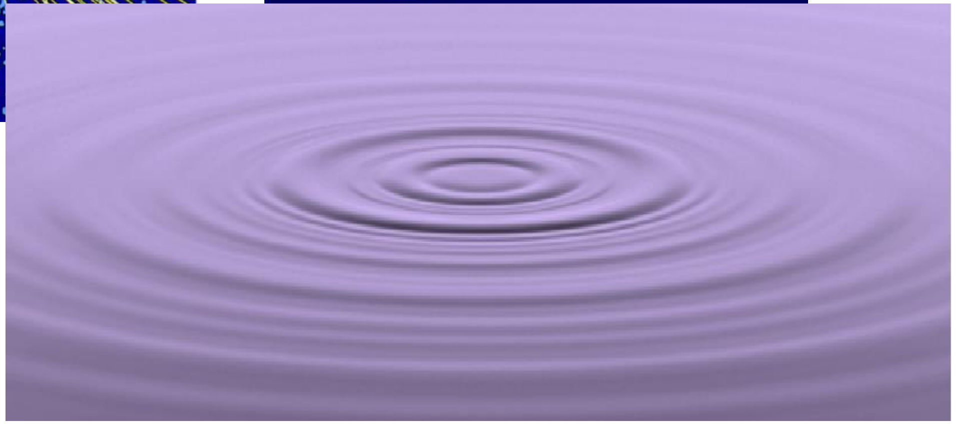
WHAT IS PHYSICS?

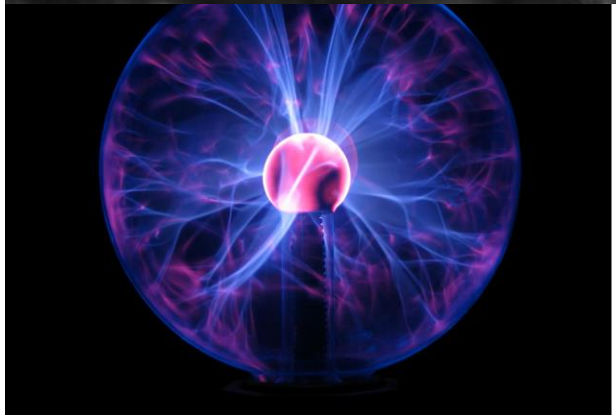


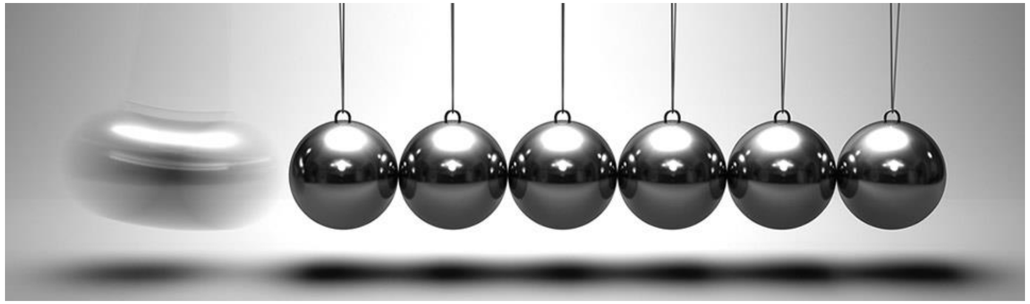




the study of everything?

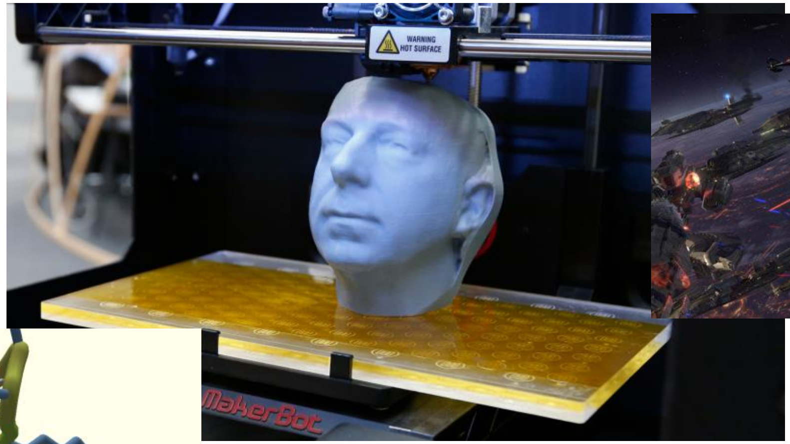




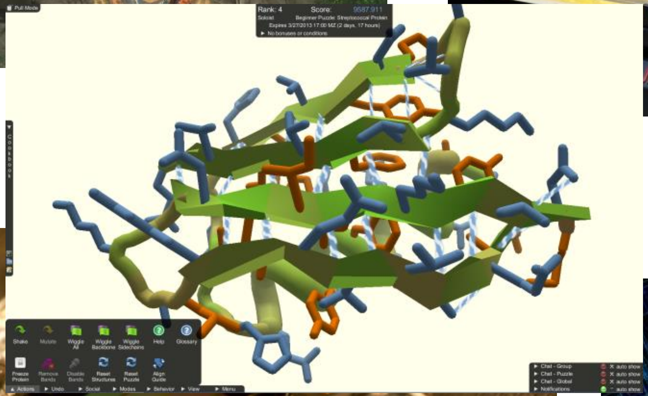


WHAT IS COMPUTATION?

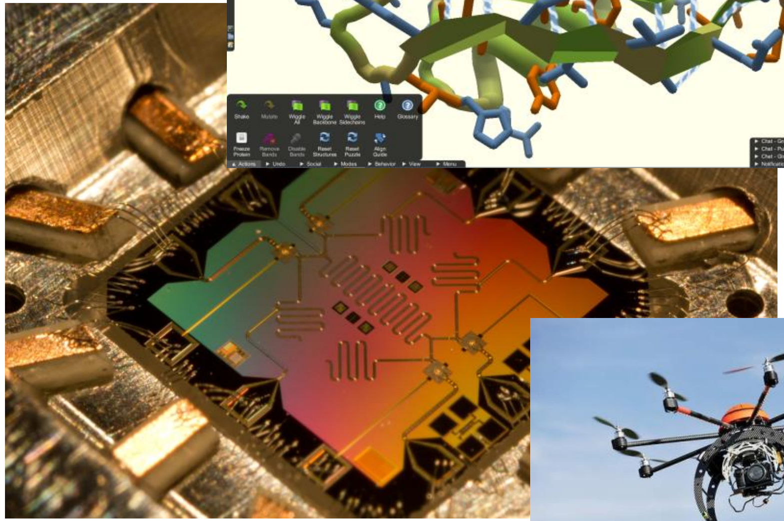
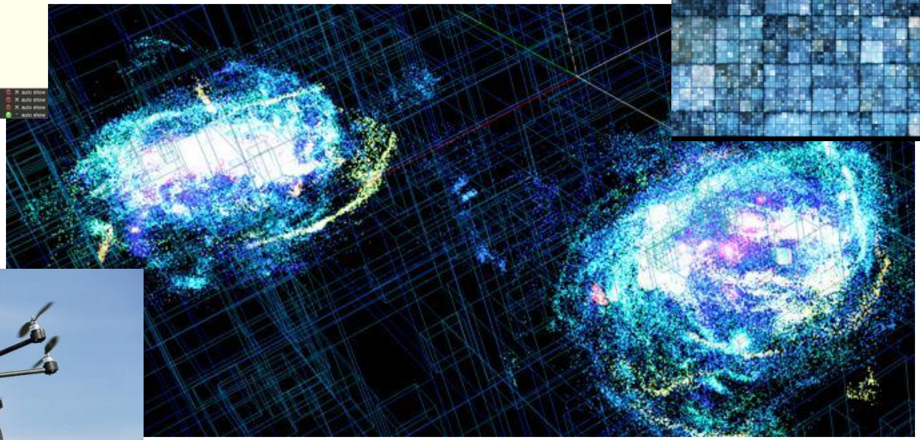
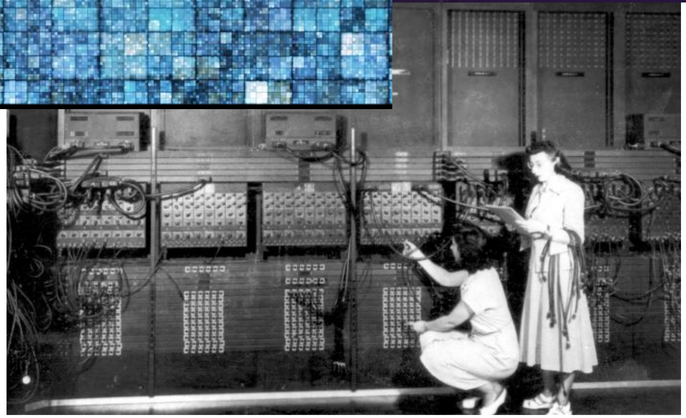









the study of everything?


OUTLINE

I Computational Physics

- History
- Today
- Examples

II Fluid Simulation

- Particle-based simulation
- Grid-based simulation
- Using Tools
- Rendering Considerations

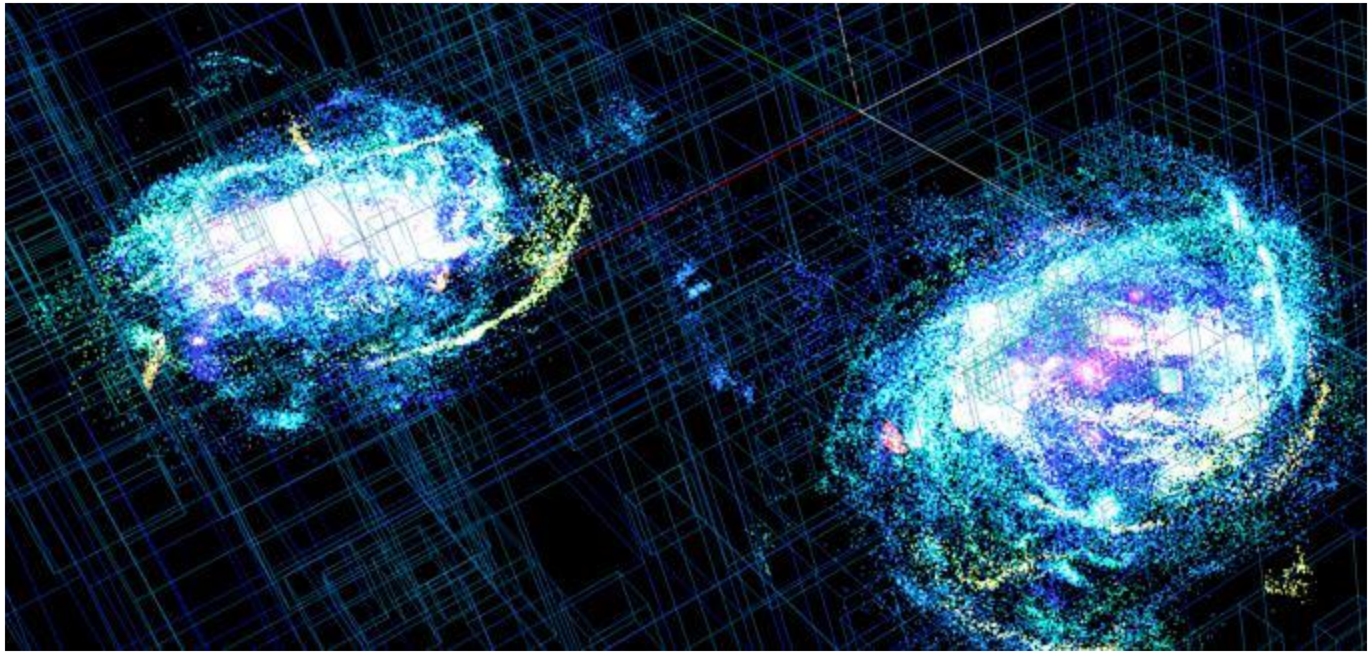
III Cloth Simulation

- Baraff and Witkin

A BRIEF HISTORY OF COMPUTATIONAL PHYSICS

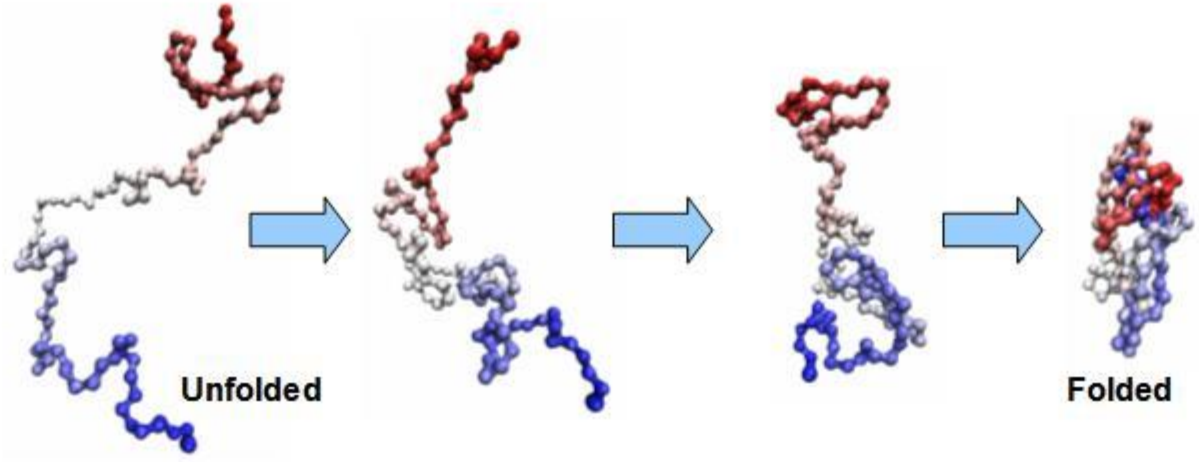
HOW COMPUTATIONAL PHYSICS IS USED TODAY

Computational astrophysics



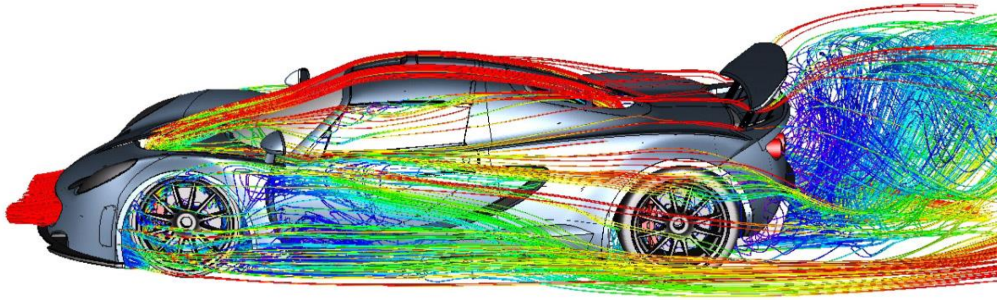
HOW COMPUTATIONAL PHYSICS IS USED TODAY

Protein folding / biology



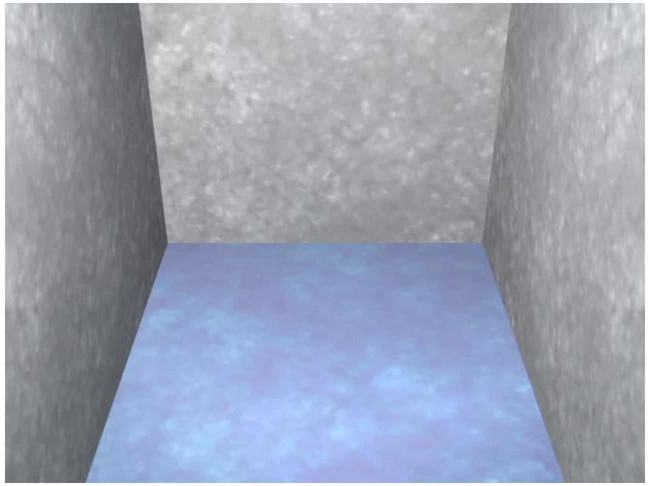
HOW COMPUTATIONAL PHYSICS IS USED TODAY

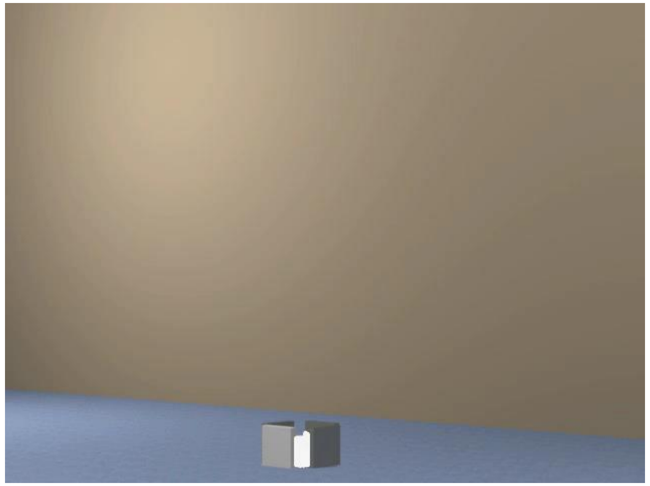
Computational fluid dynamics



HOW COMPUTATIONAL PHYSICS IS USED TODAY

Computational fluid dynamics





HOW COMPUTATIONAL PHYSICS IS USED TODAY

Graphics

HOW COMPUTATIONAL PHYSICS IS USED TODAY

Graphics

HOW COMPUTATIONAL PHYSICS IS USED TODAY

Graphics

A SIMPLE COMPUTATIONAL PHYSICS EXAMPLE

Simulating an object falling due to gravity:

$$x = x_0 + v_0 \Delta t + \frac{1}{2} a \Delta t^2$$

1. Pick a "time step" Δt
2. Solve equation to update x
3. Use new x and old x to update v
4. Repeat steps 2-4

Live Demo

MAKING COMPUTATIONAL PHYSICS WORK

1. Figure out what physical laws apply to what you want to simulate (reading, thinking, doing math)
2. Figure out how to solve those equations on a computer (reading, thinking, math)
3. Write a computer program that solves the equations (programming)
4. Debug
5. Make a finished product
    1. Render results and make cool pictures/animations (programming, art)
    2. Compare to real-world experiments and other people's work (programming, reading)
6. Use results to gain insight into universe and to guide future research

MAKING COMPUTATIONAL PHYSICS WORK

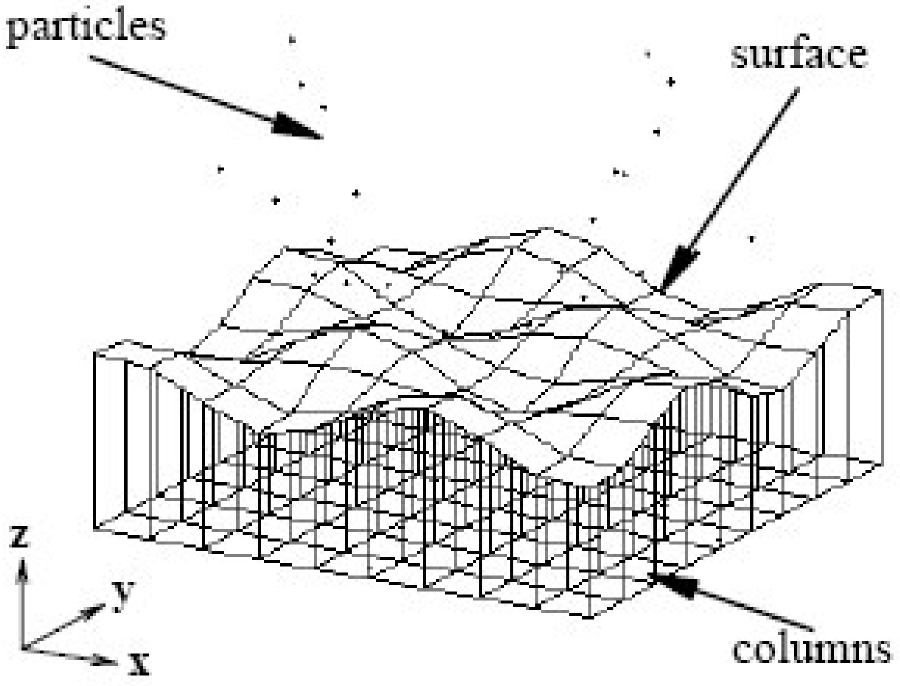
1. Figure out what physical laws apply to what you want to simulate (reading, thinking, doing math)

e.g. Navier-Stokes equations (fluid dynamics):

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho} \nabla \bar{\mathrm{p}} + \nu \nabla^2 \boldsymbol{u} + \frac{1}{3} \nu \nabla (\nabla \cdot \boldsymbol{u}) + \boldsymbol{g}$$

MAKING COMPUTATIONAL PHYSICS WORK

2. Figure out how to solve those equations on a computer (reading, thinking, math)



MAKING COMPUTATIONAL PHYSICS WORK

3. Write a computer program that solves the equations (programming)

```
%==========================================================================
% code1.m
% A very simple Navier-Stokes solver for a drop falling in a rectangular
% domain. The viscosity is taken to be a constant and a forward in time,
% centered in space discretization is used. The density is advected by a
% simple upwind scheme.
%==========================================================================
%domain size and physical variables
Lx=1.0;Ly=1.0;gx=0.0;gy=-100.0; rho1=1.0; rho2=2.0; m0=0.01; rro=rho1;
unorth=0;usouth=0;veast=0;vwest=0;time=0.0;
rad=0.15;xc=0.5;yc=0.7; % Initial drop size and location

% Numerical variables
nx=32;ny=32;dt=0.00125;nstep=100; maxit=200;maxError=0.001;beta=1.2;

% Zero various arrys
u=zeros(nx+1,ny+2); v=zeros(nx+2,ny+1); p=zeros(nx+2,ny+2);
ut=zeros(nx+1,ny+2); vt=zeros(nx+2,ny+1); tmp1=zeros(nx+2,ny+2);
uu=zeros(nx+1,ny+1); vv=zeros(nx+1,ny+1); tmp2=zeros(nx+2,ny+2);

% Set the grid
dx=Lx/nx;dy=Ly/ny;
for i=1:nx+2; x(i)=dx*(i-1.5);end; for j=1:ny+2; y(j)=dy*(j-1.5);end;

% Set density
r=zeros(nx+2,ny+2)+rho1;
for i=2:nx+1,for j=2:ny+1;
  if ( (x(i)-xc)^2+(y(j)-yc)^2 < rad^2), r(i,j)=rho2;end,
end,end
%=================== START TIME LOOP=======================================
 for is=1:nstep,is
    % tangential velocity at boundaries
    u(1:nx+1,1)=2*usouth-u(1:nx+1,2);u(1:nx+1,ny+2)=2*unorth-u(1:nx+1,ny+1);
    v(1,1:ny+1)=2*vwest-v(2,1:ny+1);v(nx+2,1:ny+1)=2*veast-v(nx+1,1:ny+1);

    for i=2:nx,for j=2:ny+1      % TEMPORARY u-velocity
      ut(i,j)=u(i,j)+dt*(-0.25*(((u(i+1,j)+u(i,j))^2-(u(i,j)+   ...
          u(i-1,j))^2)/dx+((u(i,j+1)+u(i,j))*(v(i+1,j)+         ...
          v(i,j))-(u(i,j)+u(i,j-1))*(v(i+1,j-1)+v(i,j-1)))/dy)+ ...
          m0/(0.5*(r(i+1,j)+r(i,j)))*(                          ...
                  (u(i+1,j)-2*u(i,j)+u(i-1,j))/dx^2+            ...
                  (u(i,j+1)-2*u(i,j)+u(i,j-1))/dy^2 )+gx    );
    end,end
```

OUTLINE

I Computational Physics

- History
- Today
- Examples

<u>II Fluid Simulation</u>

- Particle-based simulation
- Grid-based simulation
- Using Tools
- Rendering Considerations

III Cloth Simulation

- Baraff and Witkin

PARTICLE-BASED FLUID SIMULATION

Demo Video

WALKTHROUGH: PARTICLE-BASED FLUID SIM

Why approximate fluid as particles?

- Simplicity, speed

What must a particle know?

- Position, velocity, mass, density, pressure, force, etc.

How do particles move?

- Newton's second law ($F = ma$)

What next?

- Write the simulation loop!

WALKTHROUGH: PARTICLE-BASED FLUID SIM

The algorithm:

- Initialize particles
- For each time step Δt:
  - For each particle p_j:
    - Get neighbors N_j of p_j
    - Compute density at p_j from N_j
    - Compute pressure at p_j from N_j
    - Use density, pressure, and other forces like gravity to compute acceleration of p_j
    - Update particle velocity and position due to acceleration
    - Correct for collisions
  - Add new particles if necessary (source term)
  - Remove particles if necessary (e.g. outside domain)

WALKTHROUGH: PARTICLE-BASED FLUID SIM

Q&A:

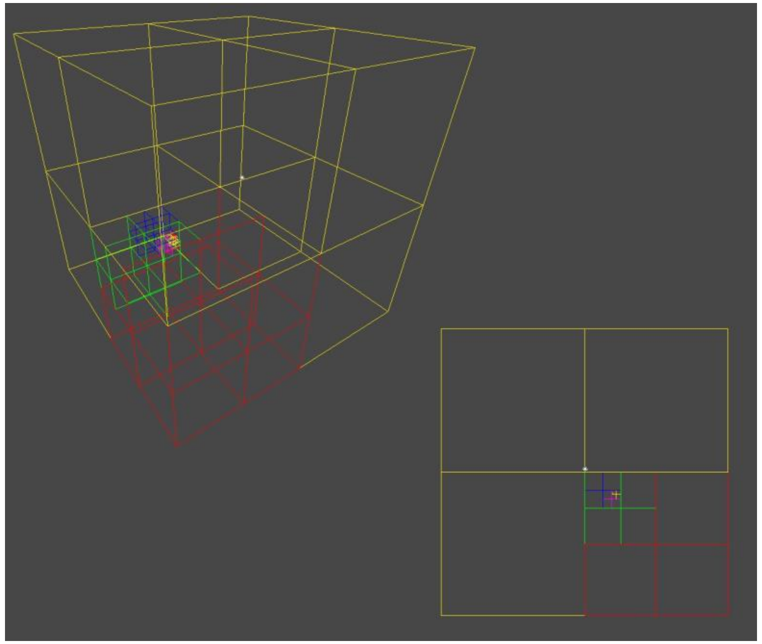
- Initialize particles
- For each time step Δt: **How to choose Δt?**
  - Smaller = more accurate, larger = faster
  - For each particle p_j:
    - Get neighbors N_j of p_j **How to get neighbors?**
      - Naively compute inter-particle distances?
      - Use a kernel, e.g. $W(d) = \frac{1}{\pi^{\frac{3}{2}} h^3} e^{\frac{r^2}{h^2}}$?
      - Acceleration structures?
    - Compute density at p_j from N_j
    - Compute pressure at p_j from N_j
    - Use density, pressure, and other forces like gravity to compute acceleration of p_j
    - Update particle velocity and position due to acceleration **How to update?**
      - Forward Euler?
      - Backward Euler?
      - RK4?
    - Correct for collisions
  - Add new particles if necessary (e.g. source terms)
  - Remove particles if necessary (e.g. outside domain)

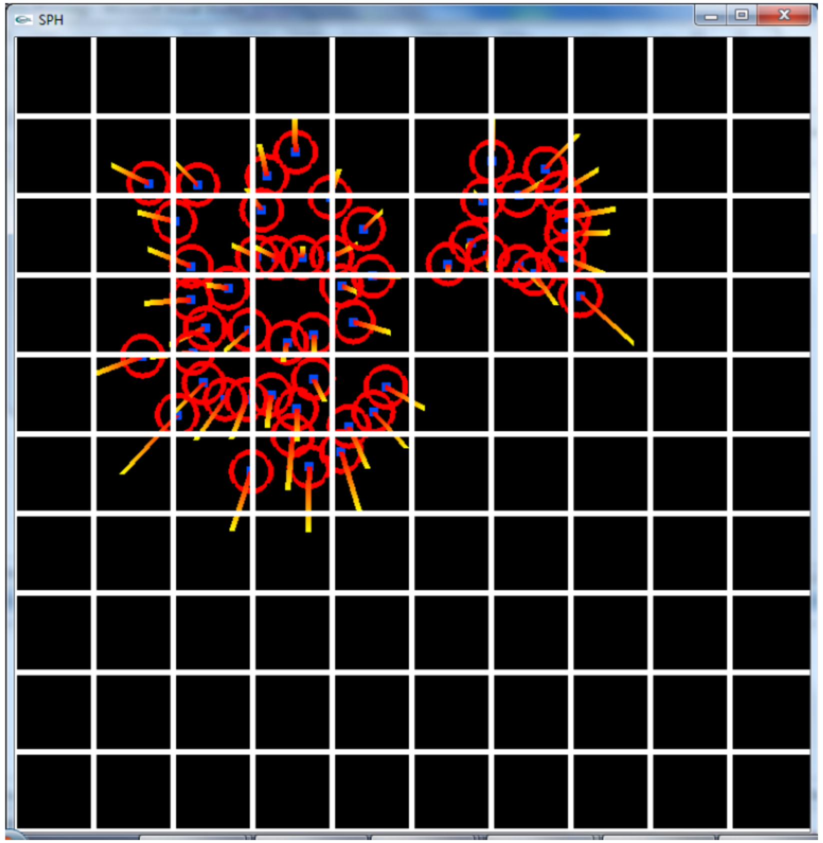
ACCELERATION STRUCTURES FOR SPH SIMS

One possible acceleration structure: spatial grid

Extension: adaptive grids

- Quadtrees, octrees





TIME INTEGRATION / 205A FREE PREVIEW

How to numerically solve an equation like $F = ma$? (Assume mass is constant.)

Can express above as ordinary differential equation (ODE):

$$\frac{dv}{dt} = Fm^{-1}$$

Forward Euler: $\frac{v^{n+1} - v^n}{\Delta t} = F^n m^{-1} \Rightarrow v^{n+1} = v^n + \Delta t F^n m^{-1}$

- Trivial to solve; unstable

Backward Euler: $\frac{v^{n+1} - v^n}{\Delta t} = F^{n+1} m^{-1} \Rightarrow v^{n+1} = v^n + \Delta t F^{n+1} m^{-1}$

- Requires inversion/iteration to solve; stable

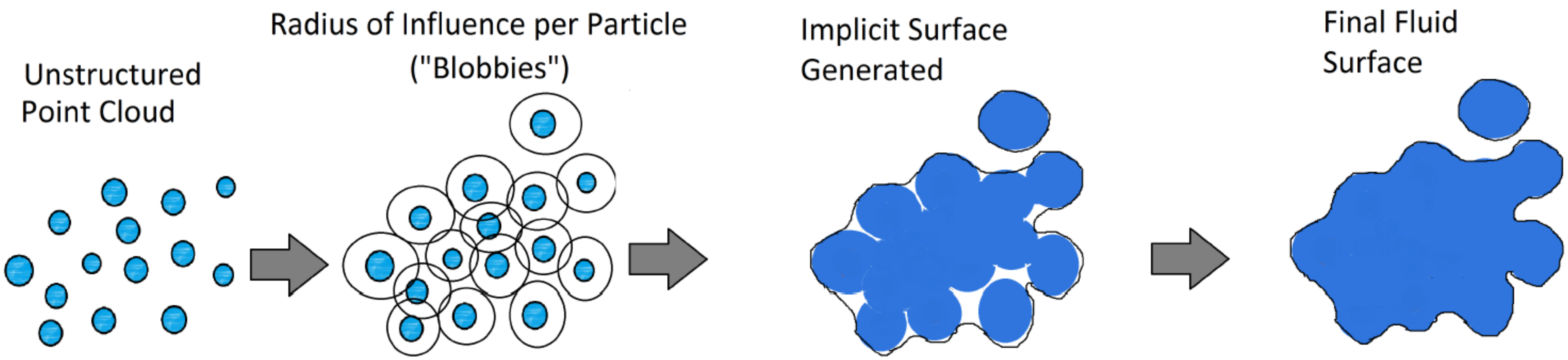
Trapezoidal: $\frac{v^{n+1} - v^n}{\Delta t} = \frac{1}{2}(F^n + F^{n+1})m^{-1} \Rightarrow v^{n+1} = v^n + \frac{\Delta t}{2}(F^n + F^{n+1})m^{-1}$

More?: Higher-order methods etc.

RENDERING SPH FLUID

Real fluid is not made of particles

Can't simulate infinite number of small particles



Demo Video

RENDERING SPH FLUID

Make it look more realistic?

Lighting, shading

Reflection, refraction

Foam, turbulence

- Color by vorticity: demo video

WALKTHROUGH: GRID-BASED FLUID SIM

Demo video

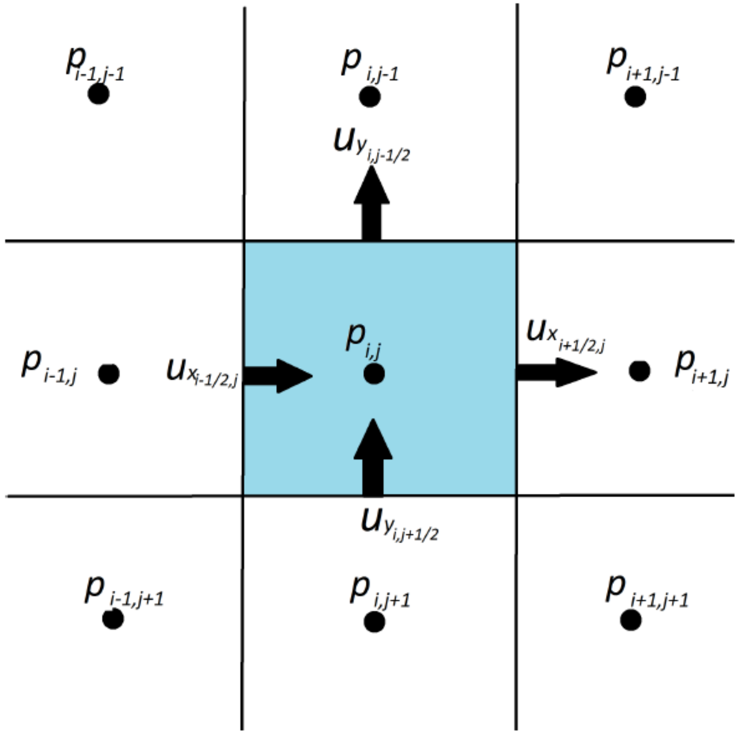
WALKTHROUGH: GRID-BASED FLUID SIM

The idea:

Discretize space into a grid

Store fluid quantities at different positions on the grid

- MAC grid
- Order of accuracy
  - $O(\Delta x), O(\Delta x^2)$, etc.
- Update fluid quantities with advection and projection steps



WALKTHROUGH: GRID-BASED FLUID SIM

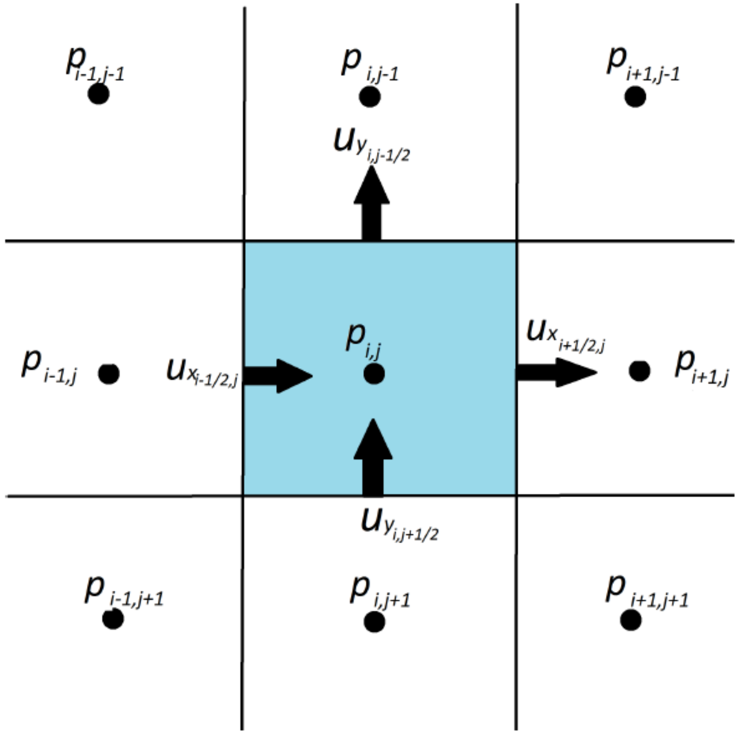
Choosing a time step

CFL condition:

$$\Delta t = \frac{\Delta h}{\vec{u}_{max}}$$

(constants out in front?)

(adaptive time steps?)



WALKTRHOUGH: GRID-BASED FLUID SIM

Semi-Lagrangian advection for a fluid quantity Q (e.g. density)

1. For each grid cell with index i, j, k

   Calculate $-\frac{\partial Q}{\partial t}$

   Calcluate the spatial position of $Q_{i,j,k}$, store it in $\vec{X}$

   Calculate $\vec{X}_{prev} = \vec{X} - \frac{\partial Q}{\partial t} * \Delta t$

   Set the gridpoint for Q^{n+1} that is nearest to $\vec{X}_{prev}$ equal to $Q_{i,j,k}$

2. Set $Q = Q^{n+1}$

(note: Forward Euler)

WALKTRHOUGH: GRID-BASED FLUID SIM

Projection and collision handling:

$$\nabla \cdot \vec{u}^{n+1} = 0$$

$$\vec{u}^{n+1} \cdot \hat{n} = \vec{u}_{solid} \cdot \hat{n}$$

RENDERING EULERIAN FLUID

Level set method

Initialize level set as *signed distance function*

- Solve Eikonal equation $|\nabla\phi| = 1$

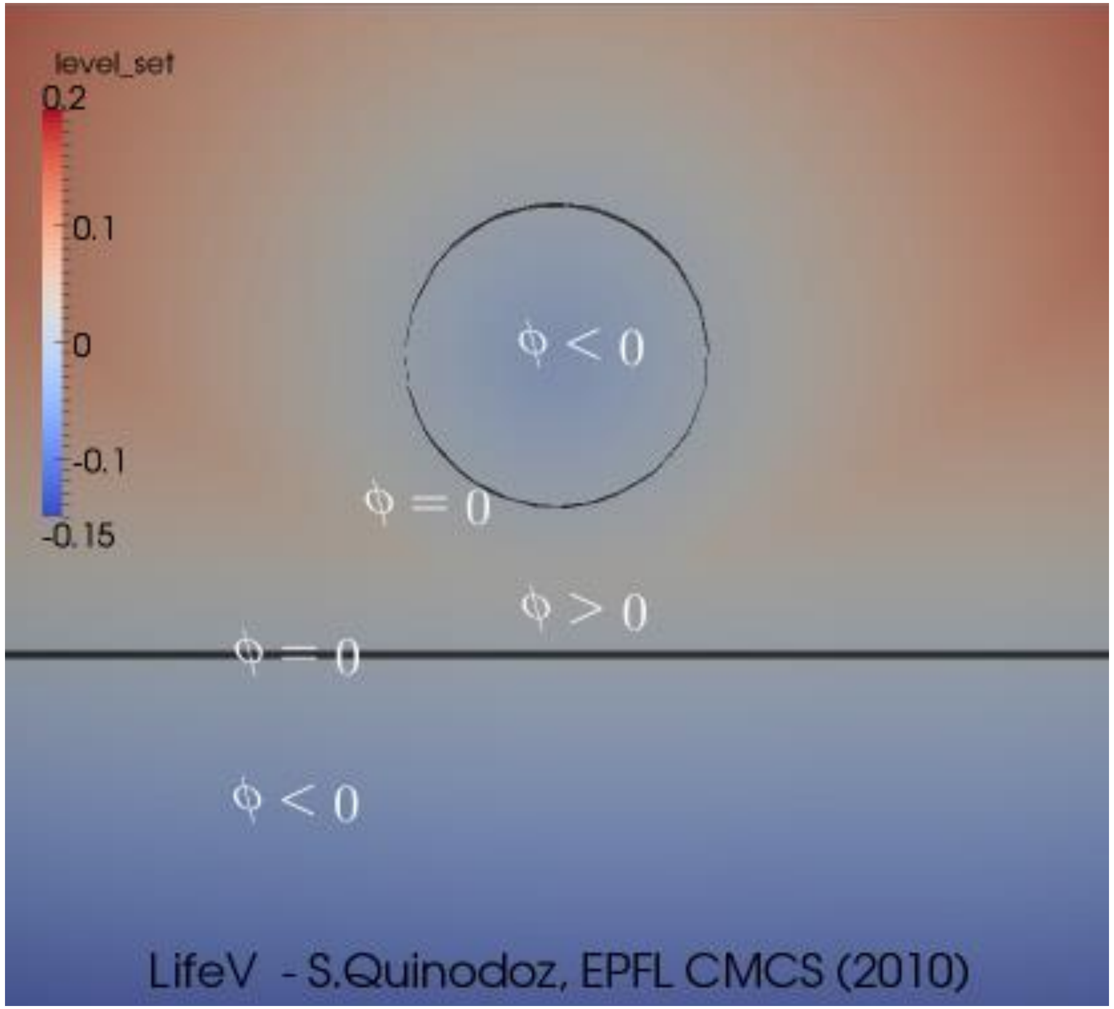
Advect level set along with fluid!

- $\frac{\partial\phi}{\partial t} = v|\nabla\phi|$

Reinitialize level set occasionally

Demo Video 1

Demo Video 2

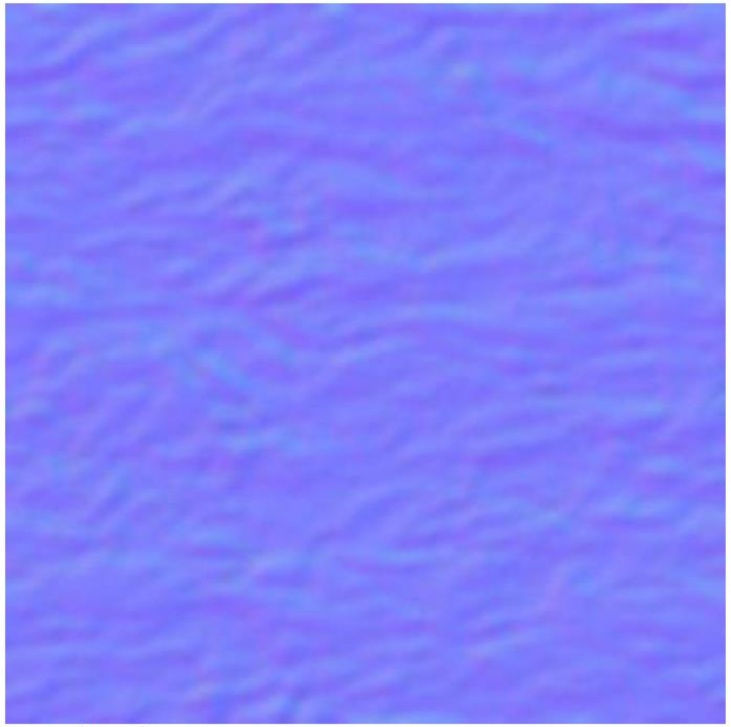


FLUID SIMULATIONS: BRIEF COMPARISON

	Particle-Based	Grid-Based
Speed?	**Faster**	Slower
Parallelization?	**Trivial**	Non-trivial
Accuracy?	Less accurate	**More accurate**
Visual appearance?	Worse	**Better**

FLUID SIMS: RENDERING CONSIDERATIONS

With fluid as triangle mesh, can apply normal mapping

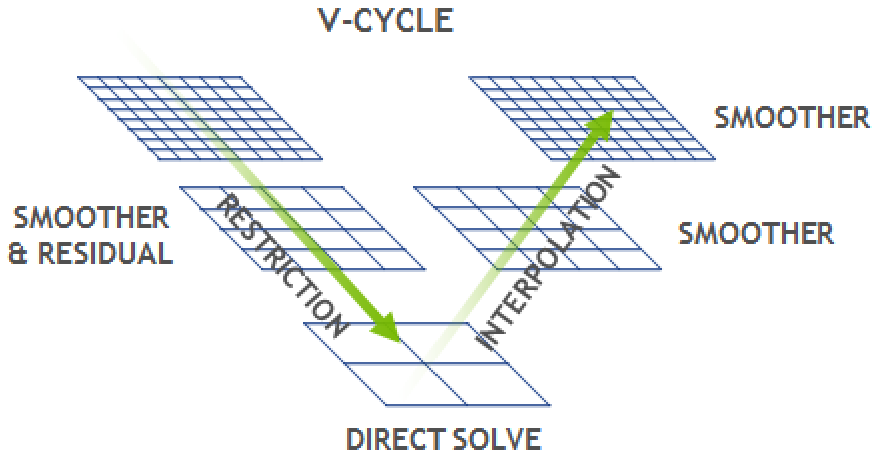


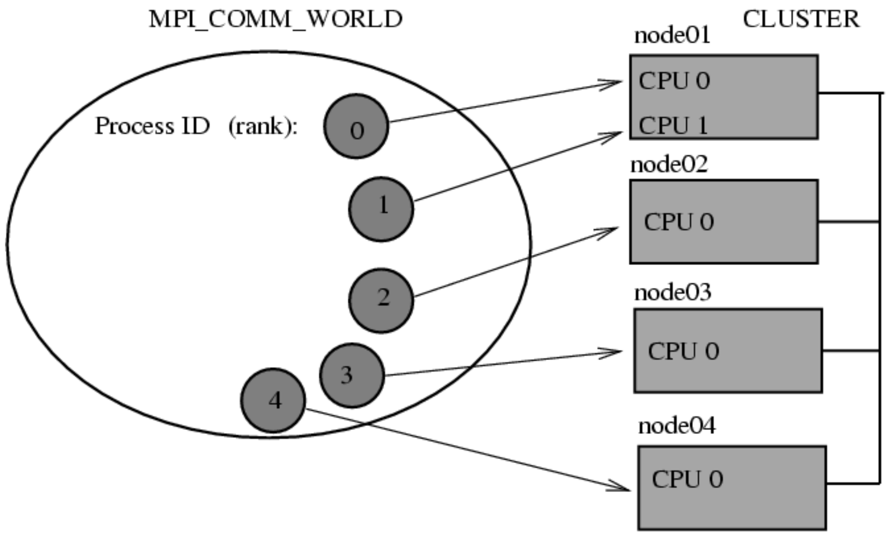
NormalMap.bmp

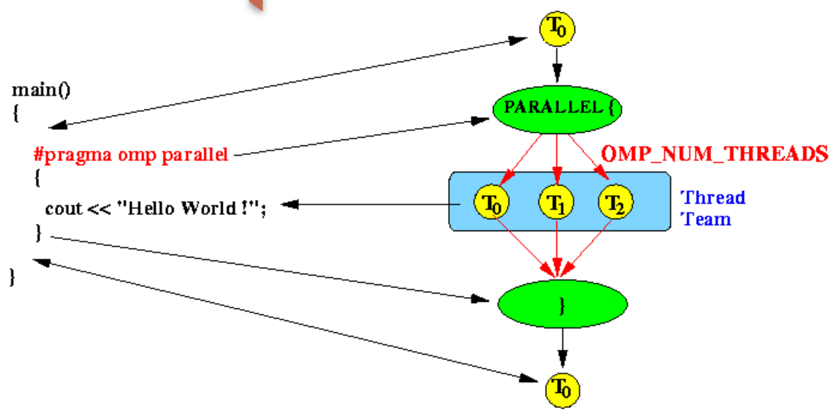
FLUID SIMS: PERFORMANCE CONSIDERATIONS

Parallelism

- GPU?
- MPI?
- OpenMP?







LIVE DEMO: FLUID SIMULATION IN MAYA

(Show rendered result afterwards)

OUTLINE

I Computational Physics

- History
- Today
- Examples

II Fluid Simulation

- Particle-based simulation
- Grid-based simulation
- Using Tools
- Rendering Considerations

III Cloth Simulation

- Baraff and Witkin

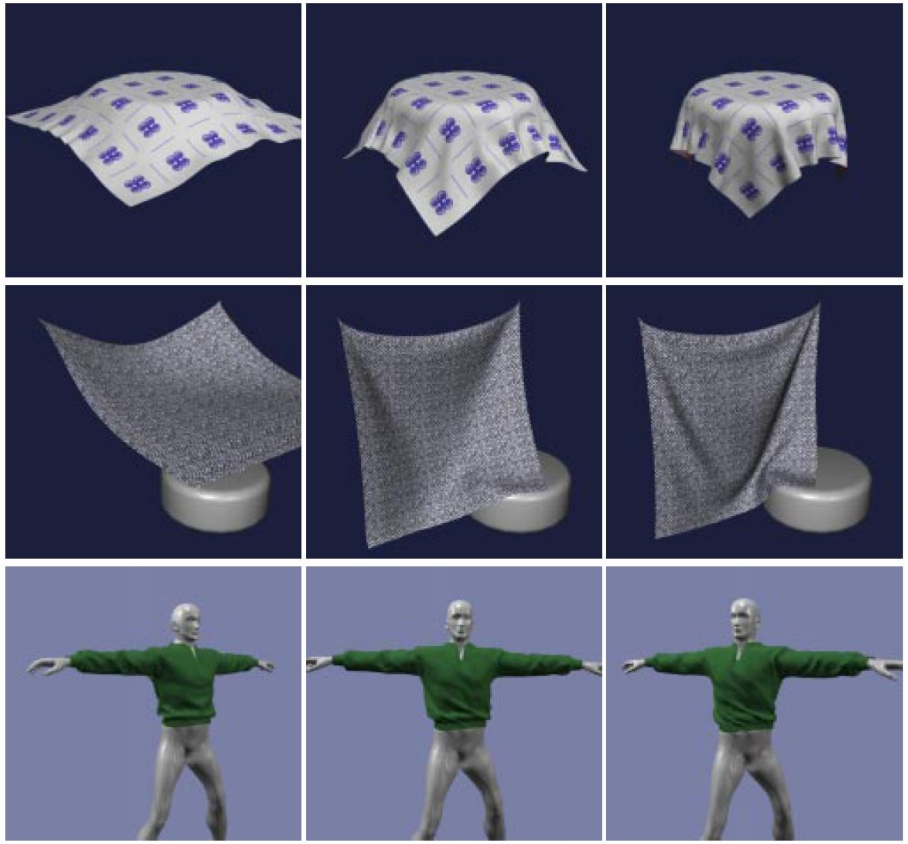
SIMULATING CLOTH

SIGGRAPH 98, Orlando, July 19–24 — COMPUTER GRAPHICS Proceedings, Annual Conference Series, 1998

Large Steps in Cloth Simulation

David Baraff   Andrew Witkin

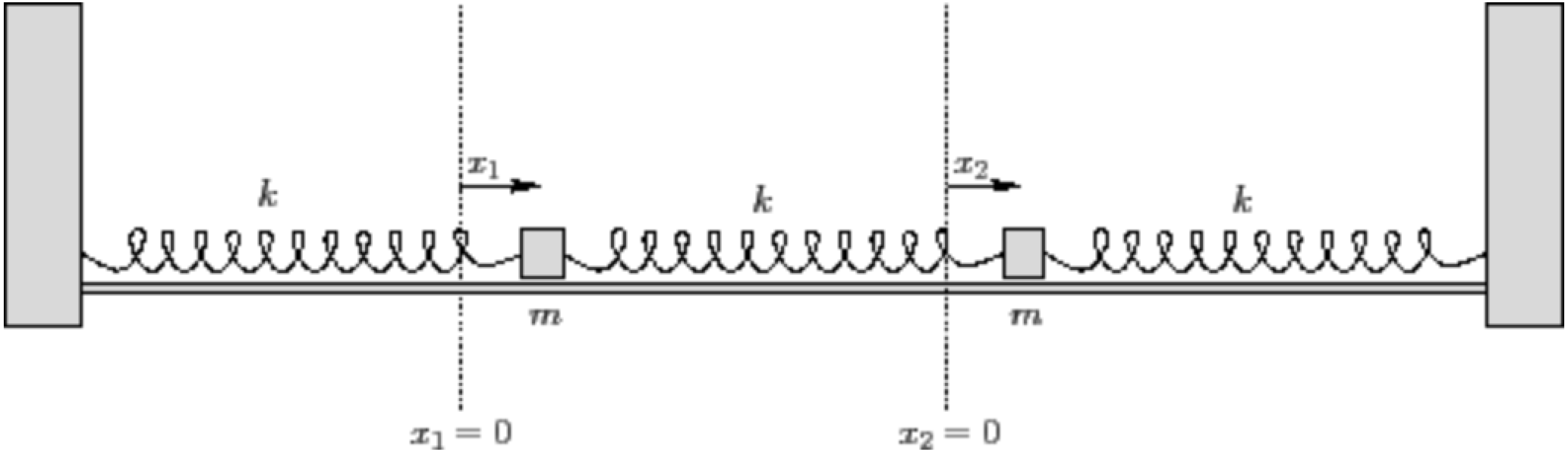
Robotics Institute
Carnegie Mellon University



A SINGLE MASS-SPRING SYSTEM

Live Demo

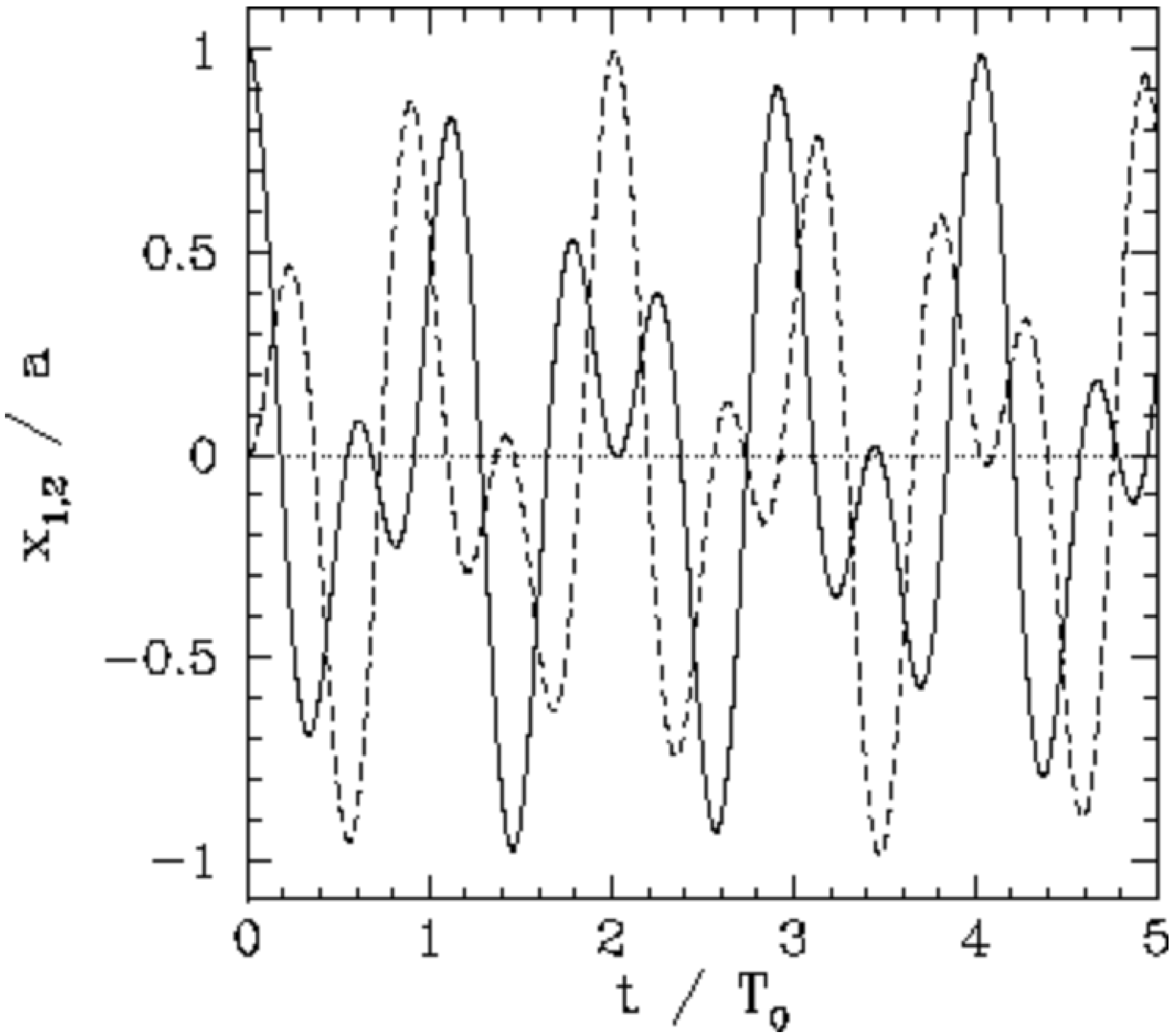
COUPLED MASS-SPRING SYSTEM



$$-\omega^2\,\hat{x}_1\,\cos(\omega\,t-\phi) = \left(-2\,\omega_0^2\,\hat{x}_1+\omega_0^2\,\hat{x}_2\right)\cos(\omega\,t-\phi),$$

$$-\omega^2\,\hat{x}_2\,\cos(\omega\,t-\phi) = \left(\omega_0^2\,\hat{x}_1-2\,\omega_0^2\,\hat{x}_2\right)\cos(\omega\,t-\phi),$$

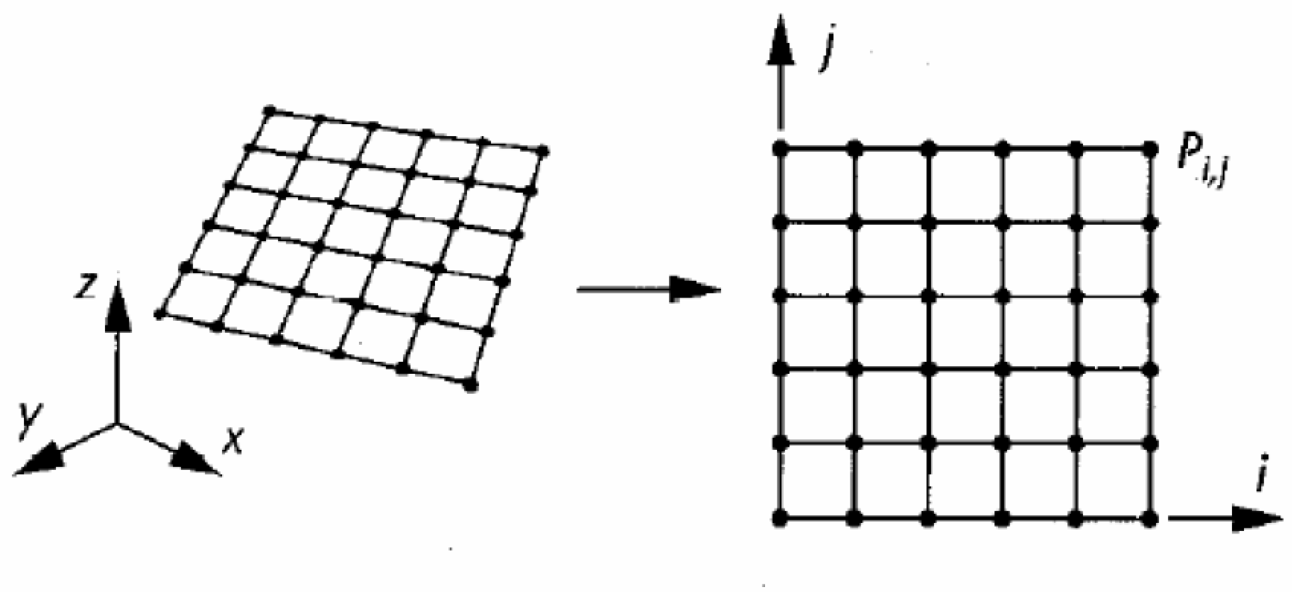
COUPLED MASS-SPRING SYSTEM



SIMULATING CLOTH

How to model a piece of cloth?

Set of nodes/vertices connected by springs (Hooke's Law: $F = kx$)



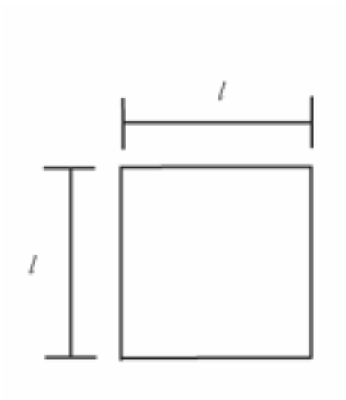
SIMULATING CLOTH

How to model a piece of cloth?

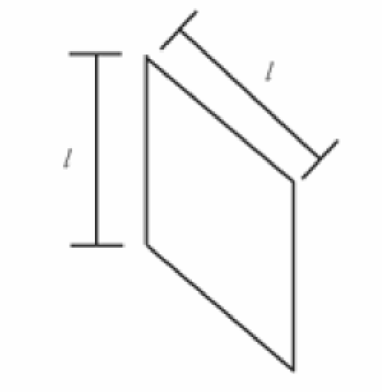
Apply various forces to cloth/springs

Restorative forces to prevent cloth craziness

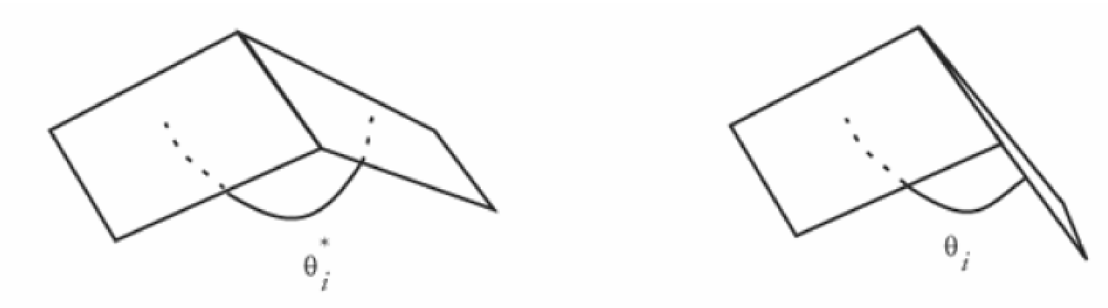
Stretch



Shear



Bending



SIMULATING CLOTH

The math

Governing ODE:

$$\ddot{x} = M^{-1}\left(-\frac{\partial E}{\partial x} + F\right)$$

where M is mass distribution of cloth, E is cloth's internal energy, and F captures forces like air drag, contact, bending, internal damping, etc.

"Energy" idea: for a (vector) condition C we want to be zero, associate with it an energy function

$$E_C(x) = \frac{1}{2}kC(x)^T C(x)$$

(looks like kinetic energy)

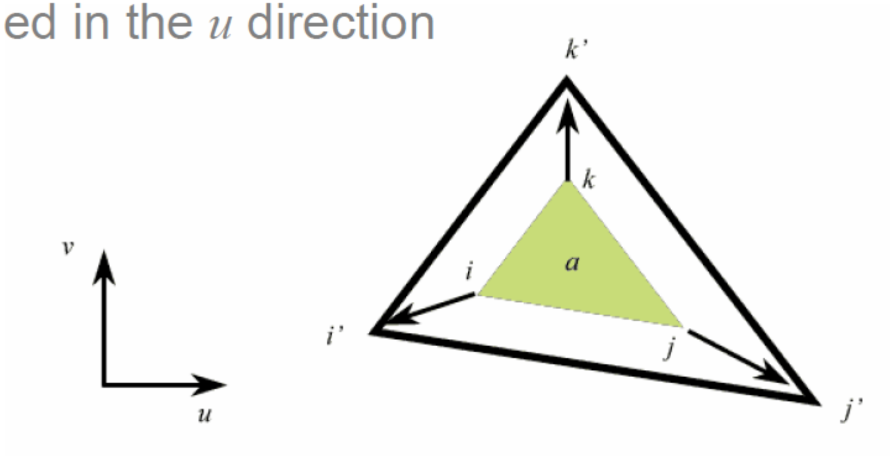
SIMULATING CLOTH

Condition *C(x)* used for the stretch energy:

magnitude stretched in the u direction

$$C(x) = a\begin{pmatrix} \|w_u(x)\| - b_u \\ \|w_v(x)\| - b_v \end{pmatrix}$$

magnitude stretched in the v direction



where a = triangle's area in uv coordinates

$b_u = b_v$ = rest length = 1

SIMULATING CLOTH

Implicit (Backward Euler) time integration method

- Stability means we can take large time steps

$$\begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{pmatrix} = h \begin{pmatrix} \mathbf{v}_0 + \Delta\mathbf{v} \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}, \mathbf{v}_0 + \Delta\mathbf{v}) \end{pmatrix}$$

(h is step size)

SIMULATING CLOTH

Collision handling?

Demo Video

CONCLUSIONS

Physics and computation

Particle-based and grid-based fluid simulation

Cloth simulation

Where to go from here?

THANK YOU!

Any questions?