WHAT IS PHYSICS?

the study of everything?
WHAT IS COMPUTATION?

the study of everything?
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” \( \Delta t \)
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
1. Figure out what physical laws apply to what you want to simulate (reading, thinking, doing math)

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

\[
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla \bar{p} + \nu \nabla^2 \mathbf{u} + \frac{1}{3} \nu \nabla (\nabla \cdot \mathbf{u}) + \mathbf{g}
\]
2. Figure out how to solve those equations on a computer (reading, thinking, math)
3. Write a computer program that solves the equations (programming)
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
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!
The algorithm:
- Initialize particles
- For each time step $\Delta t$:
  - 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 $\Delta t$: How to choose $\Delta t$?
  - For each particle $p_j$:
    - Get neighbors $N_j$ of $p_j$ How to get neighbors?
    - 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?
    - Correct for collisions
    - Add new particles if necessary (e.g. source terms)
    - Remove particles if necessary (e.g. outside domain)

- Naively compute inter-particle distances?
- Use a kernel, e.g. $W(d) = \frac{1}{\pi^2 h^3} e^{\frac{r^2}{h^2}}$?
- Acceleration structures?

- Smaller = more accurate, larger = faster

- Forward Euler?
- Backward Euler?
- RK4?
One possible acceleration structure: spatial grid

Extension: adaptive grids
- Quadtrees, octrees
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:**

$$\frac{v^{n+1} - v^n}{\Delta t} = Fnm^{-1} \implies v^{n+1} = v^n + \Delta tFnm^{-1}$$

- Trivial to solve; unstable

**Backward Euler:**

$$\frac{v^{n+1} - v^n}{\Delta t} =Fn+1m^{-1} \implies v^{n+1} = v^n + \Delta tFn+1m^{-1}$$

- Requires inversion/iteration to solve; stable

**Trapezoidal:**

$$\frac{v^{n+1} - v^n}{\Delta t} = \frac{1}{2}(F^n + Fn+1)m^{-1} \implies v^{n+1} = v^n + \frac{\Delta t}{2}(F^n + Fn+1)m^{-1}$$

More?: Higher-order methods etc.
RENDERING SPH FLUID

Real fluid is not made of particles
Can’t simulate infinite number of small particles

[Diagram showing the process from unstructured point cloud, radius of influence per particle ("blobbies"), implicit surface generated, to final fluid surface]

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
Choosing a time step
CFL condition:

$$\Delta t = \frac{\Delta h}{u_{max}}$$

(constants out in front?)

(adaptive time steps?)
Semi-Lagrangian advection for a fluid quantity $Q$ (e.g. density)

1. For each grid cell with index $i, j, k$
   
   Calculate $-\frac{\partial Q}{\partial t}$

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

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

   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)
WALKTHROUGH: GRID-BASED FLUID SIM

Projection and collision handling:

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

\[ \vec{u}^{n+1} \cdot \hat{n} = \vec{u}_{solid} \cdot \hat{n} \]
RENDEERING 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

<table>
<thead>
<tr>
<th></th>
<th>Particle-Based</th>
<th>Grid-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed?</td>
<td>Faster</td>
<td>Slower</td>
</tr>
<tr>
<td>Parallelization?</td>
<td>Trivial</td>
<td>Non-trivial</td>
</tr>
<tr>
<td>Accuracy?</td>
<td>Less accurate</td>
<td>More accurate</td>
</tr>
<tr>
<td>Visual appearance?</td>
<td>Worse</td>
<td>Better</td>
</tr>
</tbody>
</table>
FLUID SIMS: RENDERING CONSIDERATIONS

With fluid as triangle mesh, can apply normal mapping
FLUID SIMS: PERFORMANCE CONSIDERATIONS

Parallelism
- GPU?
- MPI?
- OpenMP?
LIVE DEMO: FLUID SIMULATION IN MAYA

>Show rendered result afterwards)
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
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**

\[
\begin{align*}
-\omega^2 \hat{x}_1 \cos(\omega t - \phi) &= \left(-2 \omega_0^2 \hat{x}_1 + \omega_0^2 \hat{x}_2 \right) \cos(\omega t - \phi), \\
-\omega^2 \hat{x}_2 \cos(\omega t - \phi) &= \left(\omega_0^2 \hat{x}_1 - 2 \omega_0^2 \hat{x}_2 \right) \cos(\omega t - \phi),
\end{align*}
\]
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
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

\[ E_C(x) = \frac{1}{2} k C(x)^T C(x) \]

(looks like kinetic energy)
Condition $C(x)$ used for the stretch energy:

$$C(x) = a \left( \frac{\|w_u(x)\|}{\|w_v(x)\|} - b_u \right)$$

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

\[
\begin{pmatrix}
\Delta x \\
\Delta v
\end{pmatrix} = h \begin{pmatrix}
v_0 + \Delta v \\
M^{-1}f(x_0 + \Delta x, v_0 + \Delta v)
\end{pmatrix}
\]

(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?