SPH Fluid Simulation

Stephen J. Guy

Feb 26, 2020

1

Quiz 1

- Next Class!
- Conceptual questions:
  - E.g., When to use SPH vs Shallow Water?
- Topics:
  - Physical Sim, Numerical Integration, Particle Systems, Cloth Simulation, Drag, Numerical Stability, SPH, Shallow Water, Eulerian vs. Lagrangian, Spatial Data Structures

2

2

Spatial Data Structures

3

3

Fluid Simulation

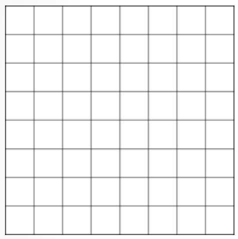
- Lagrangian vs Eulerian
- Lagrangian
  - o Follow single parcel of fluid over time
- Eulerian
  - o Follow changes in flow over a single location in space
- Frame of reference
  - o Lagrangian – moving with particle
  - o Eulerian – fixed in space
- Lagrangian – very similar to mass-spring

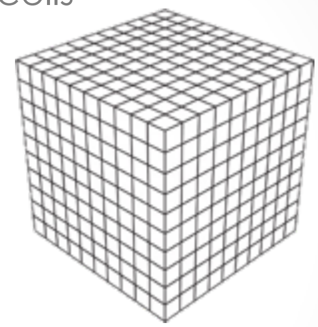
4

4

Eulerian Dynamics

- Discretize space into cells





- Compute dynamics for each grid cell
  - Density and Velocity

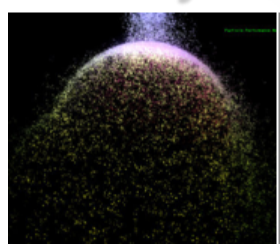
$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f}$$

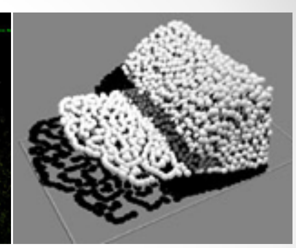
5

5

Lagrangian Dynamics

- Examples
  - SPH fluids
  - Particle systems





- Must derive how particle state changes over time

$$\boxed{F = m*A = G}$$

$$\boxed{\frac{d\mathbf{v}_a}{dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) \nabla W_{ab}}$$

6

6

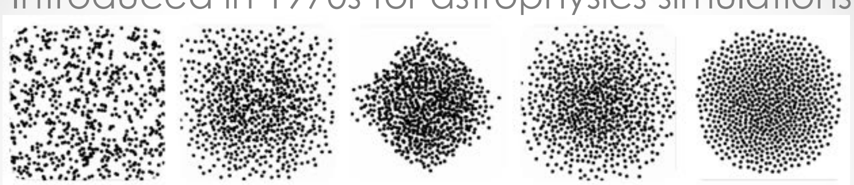
Lagrangian (SPH)

7

7

SPH - Background

- Introduced in 1970s for astrophysics simulations



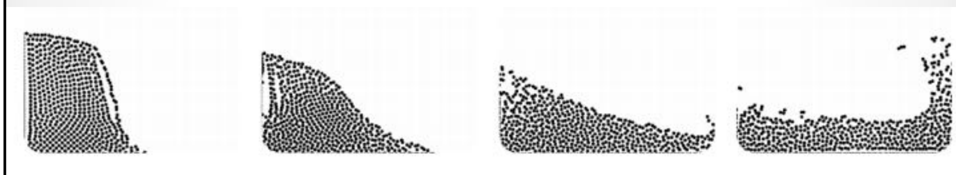
- Gained popularity quickly in other areas
  - Esp. Computer graphics
- Represents fluids as a collection of particles
  - Particles represent fluid attributes
  - Lagrangian simulation method

8

8

Discussion

- How might you make a particle-based fluid simulation?
- What forces effect a drop of water?



SPH Principles

- F = ma → a = F/m
- Potential Forces:
  - Gravity
  - Incompressibility -> Pressure
  - Surface tension
- Mass:
  - Density

Fluid Dynamics

- Euler equation(inviscid fluid)

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla P + \mathbf{b},$$

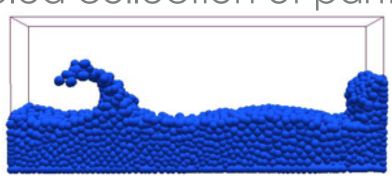
  - v – velocity, ρ – density
  - P – pressure,∇ - gradient
  - b – external force (e.g., gravity)
- SPH Simulation:
  - Represent fluid as several particles
  - Estimate values locally around each particle
  - Update each particle according to dv/dt

11

11

SPH – Evaluating Quantities

- Fluid is sampled collection of particles



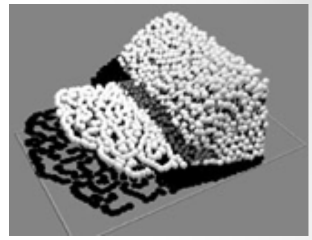
- Each particle has some attributes (mass, density, position, …)
- How to evaluate the attribute an a arbitrary location r:
  - Weighted, spatial average – smoothing function W

$$A(\mathbf{r}) \approx \sum_b \frac{A_b}{\rho_b} W(\mathbf{r} - \mathbf{r}_b, h) m_b$$

12

12

SPH - cont



- Computing particle density

$$\rho_a = \sum_b m_b W(\mathbf{r}_a - \mathbf{r}_b, h)$$

- Approximating derivatives

$$\nabla A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(|\mathbf{r} - \mathbf{r}_j|, h).$$

- Pairwise approximations to conserve momentum
  - o E.g.

$$\frac{d\mathbf{v}_a}{dt} = -\sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) \nabla W_{ab}$$

where $\nabla W_{ab} = \nabla W(\mathbf{r}_a - \mathbf{r}_b, h)$

13

13

SPH – Neighbor Search

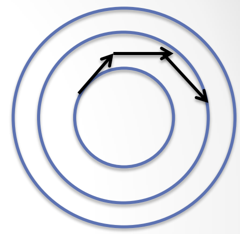
- Finding neighbors is very time consuming $O(n^2)$
- Not every particle has a meaningful effect on every other particle
- Close neighbors effect more
  - o Only compute values over close neighbors
- Use Spatial Data Structure

14

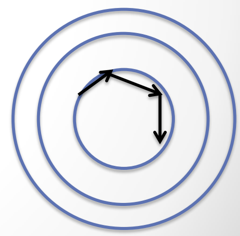
14

SPH - Damping

- Eulerian integration is bad!
  - $x(t+\Delta t) \quad = \quad x(t) + f(t)\Delta t$
  - Unstable



- Damp changes
  - Create fictional damping force



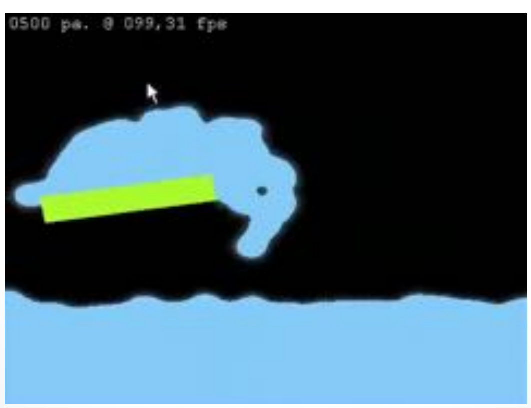
$$
\begin{aligned}
\frac{d\mathbf{x}_a}{dt} &= \mathbf{v}_a \\
\frac{d\mathbf{v}_a}{dt} &= -\nu\mathbf{v}_a - \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2}\right) \nabla W_{ab} + \mathbf{b}_d
\end{aligned}
$$

15

15

Putting It All Together

- http://www.youtube.com/watch?v=0bL80G1HX9w
- C# code at - http://rene-schulte.info/

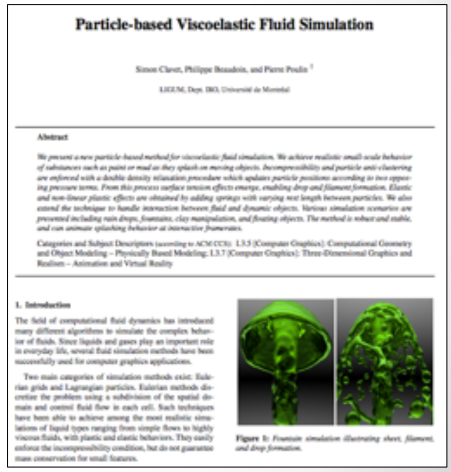


16

16

Particle-based Viscoelastic Fluid Sim.

- Clavet et al., Symp. Comp. Animation 2005
- Fun & approachable SPH formulation
  - Flexible system that captures many phenomena
- If you implement SPH start here!



Particle-based Viscoelastic Fluid Simulation

Simon Clavet, Philippe Beaudoin, and Pierre Poulin [1]

LIGUM, Dept. IRO, Université de Montréal

Abstract

We present a new particle-based method for viscoelastic fluid simulation. We achieve realistic small-scale behavior of substances such as paint or mud as they splash on moving objects. Incompressibility and particle anti-clustering are enforced with a double density relaxation procedure which updates particle positions according to two opposing pressure terms. From this process surface tension effects emerge, enabling drop and filament formation. Elastic and non-linear plastic effects are obtained by adding springs with varying rest length between particles. We also extend the technique to handle interaction between fluid and dynamic objects. Various simulation scenarios are presented including rain drops, fountains, clay manipulation, and floating objects. The method is robust and stable, and can animate splashing behavior at interactive framerates.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Animation and Virtual Reality

1. Introduction

The field of computational fluid dynamics has introduced many different algorithms to simulate the complex behavior of fluids. Since liquids and gases play an important role in everyday life, several fluid simulation methods have been successfully used for computer graphics applications.

Two main categories of simulation methods exist: Eulerian grids and Lagrangian particles. Eulerian methods discretize the problem using a subdivision of the spatial domain and control fluid flow in each cell. Such techniques have been able to achieve among the most realistic simulations of liquid types ranging from simple flows to highly viscous fluids, with plastic and elastic behaviors. They easily enforce the incompressibility condition, but do not guarantee mass conservation for small features.

Figure 1: *Fountain simulation illustrating sheet, filament, and drop formation.*

17

17

Computing Fluid State

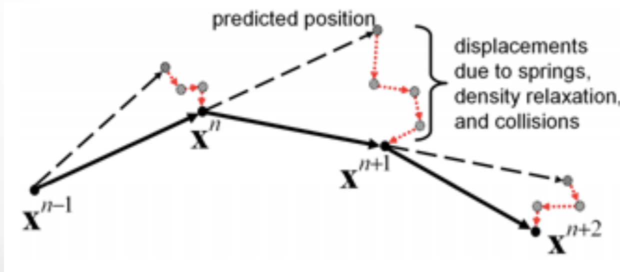
- Density: $\rho_i = \sum_{j \in N(i)} (1 - r_{ij}/h)^2$
- Pressure: $\boxed{P_i = k(\rho_i - \rho_0)}$!?!?
- Displacement: $\mathbf{D}_{ij} = \Delta t^2 P_i (1 - r_{ij}/h)\hat{\mathbf{r}}_{ij}$
- Assumptions:
  - "Weekly compressible" fluid
    - True water is incompressible
  - All particles, have same mass
  - User will tune k and ρ_0 until it looks good

18

18

Prediction-Relaxation

- Improve integration through a prediction-relaxation step
- 1st Integrate previous velocity over time to guess new positions
- 2nd Apply spring / penalty forces to refine the updated position
- 3rd Use this updated position to infer velocities



19

19

Psedocode

```
FOREACH PARTICLE p:
 p.vel = (p.pos – p.posOld) / dt
 p.posOld = pos
 p.vel += gravity * dt
 p.pos += p.vel * dt
 p.dens = 0

FOREACH PARTICLE p:
 FOREACH PARTICLE p2
   if (dist(p,p2) < K_smoothingRadius)
    createPair(p, p2)

FOREACH PAIR p (of particles A B)
 p.q = 1 - dist(A, B) / K_smoothingRadius
 p.q2 = p.q^2
 A.dens += p.q2
 B.dens += p.q2
```

```
FOREACH PARTICLE p
 p.press = K_stiff * (p.dens - K_restDensity)

FOREACH PAIR p (of particles A B)
 press = A.press + B.press
 displace = (press*p.q) *(dt^2)
 a2bN = directionNormal(A, B)
 A.pos += displace * a2bN
 B.pos += -displace * a2bN
```

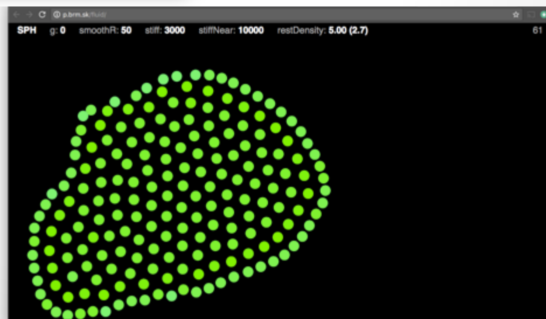
Wrong!

Clavet et al. 2005, "Particle-Based Viscoelastic Fluid Simulation"

20

20

Wrong: No Cohesion
Clustering
Uneven density

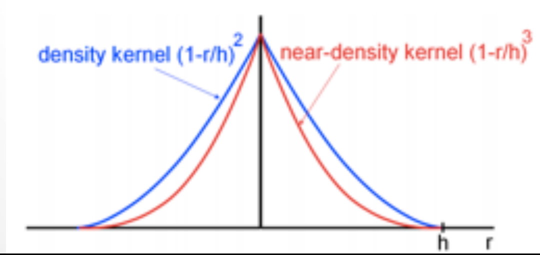


Goal: Uniform behavior

21

Issue

- All the particles rush in small clusters
  - Minimum energy solution
- Idea: Add cohesion force to nearby neighbors
  - 2nd pressure computation
  - 2nd ρ_0 of 0 (always stick together)
  - 2nd density function which drops to zero faster



22

22

Improved Psedocode

```
FOREACH PARTICLE p:
 p.vel = (p.pos - p.posOld) / dt
 p.posOld = pos
 p.vel += gravity * dt
 p.pos += p.vel * dt
 p.dens = p.densN = 0

FOREACH PARTICLE p:
 FOREACH PARTICLE p2
  if (dist(p,p2) < K_smoothingRadius)
   createPair(p, p2)

FOREACH PAIR p (of particles A B)
 p.q = 1 - dist(A, B) / K_smoothingRadius
 p.q2 = p.q^2
 p.q3 = p.q^3
 A.dens += p.q2
 B.dens += p.q2
 A.densN += p.q3
 B.densN += p.q3
```

```
FOREACH PARTICLE p
 p.press = K_stiff * (p.dens - K_restDensity)
 p.pressN = K_stiffN * p.densN

FOREACH PAIR p (of particles A B)
 press = A.press + B.press
 pressN = A.pressN + B.pressN
 displace = (press*p.q + pressN*p.q2) *(dt^2)
 a2bN = directionNormal(A, B) // or b2a
 A.pos += displace * a2bN
 B.pos += -displace * a2bN
```

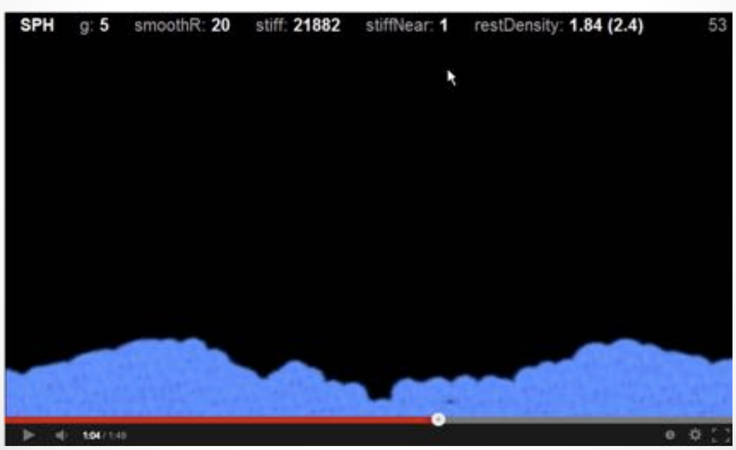
Clavet et al. 2005, "Particle-Based Viscoelastic Fluid Simulation"

23

23

Javascript Implimentation

- http://www.youtube.com/watch?v=2aWuXfwZ8zg
  - http://p.brm.sk/fluid/ <- Live Demo!

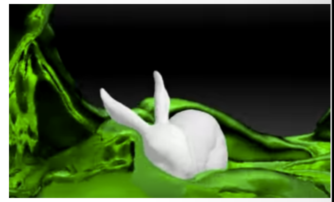


24

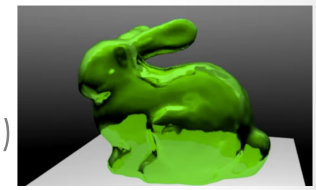
24

Other Cool Ideas

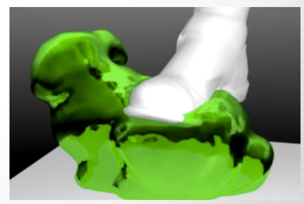
- Object/fluid Interaction:
  - o Allow particles in impart an impulse on objects
  - o (add velocity)



- Elasticity (Squishy/firm objects):
  - o Add springs between particles
  - o Rest-length = interaction radius(h)



- Plasticity (Deformation):
  - o Rest length stretches based on length or force



25

SPH – Wrap-up

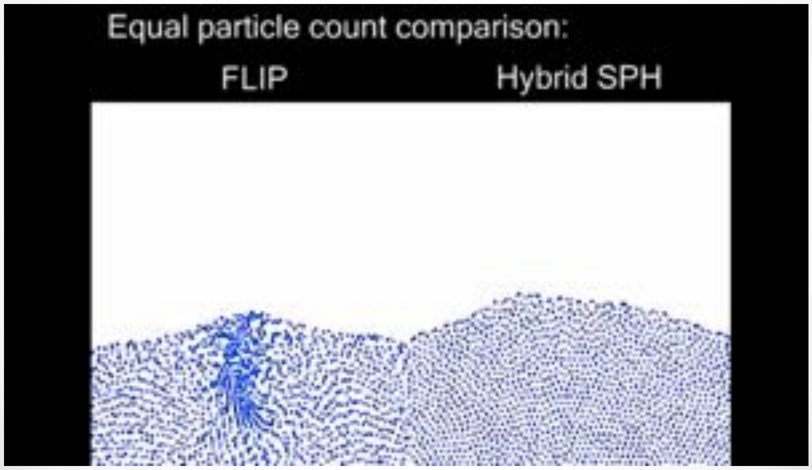
- Concept is very simple
- Lots of great research in this area
- Implementation is very easy to get wrong
  - o Lots of parameters to tune:
    - Shape of W, h
    - Hacky forces for surface tension, or to deal with timestep issues
- You will need a very small timestep
- It will run surprisingly slowly
- Start with 2D!

26

26

More

- http://www.youtube.com/watch?v=pxDeVrJO5yY

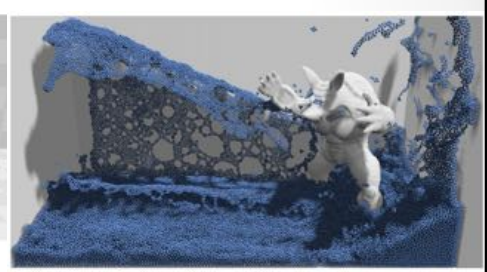


27

27

More Recent Work

- *Position Based Fluids*, Siggraph 2013 http://mmacklin.com/pbf_sig_preprint.pdf
- Large-scale, realtime fluid simulation
- Clear pseudocode!



Great Demos: http://blog.mmacklin.com/2013/07/25/siggraph-slides/

28

28