CS195V Week 7

Fluids?!

Overview

- Hopefully you're making good progress on the NBody project
  - Or at least started, maybe? (No you haven't, who are we kidding)
  - No really, you should start
- Now we have this nice particle engine, why not use it for other things?
- How about simulating fluids?

Fluids, in General

- Matter made up of atoms/molecules
- When in a liquid or gaseous state, such matter dynamically changes shape based on environment
- Fluids can be described by the Navier Stokes Equations
- Two primary ways to model fluids in gfx
  - Grid based methods
  - Particle based methods
- You already have a particle sim from NBody, so we're going to talk about particle methods

Grids vs Particles

- Grid based methods must subdivide the region into a grid and compute flow fields for each cell in the grid
  - This has some limitations including limited spatial extent, choice of grid size, extension to multiple fluids, and collision / boundary conditions
- Particle based methods only calculate forces at particle locations and can trivially be extended to incorporate multiple fluids into one simulation

Smoothed Particle Hydrodynamics

- Model the fluid as a set of discrete units, or particles
- The properties of each particle are determined by applying some kernel function to each of its neighbors
  - Much like a filter kernel (we know how much you guys love those)
  - Sounds parallelizable!


REALFLOW
2012

SPH

- Smoothed particle hydrodynamics originates from computational astrophysics and is designed for compressible flow problems
- SPH can approximate derivatives at any location by operating on arbitrary particle locations
  - It is basically an interpolation method
- Some notation
  - rho - density
  - p - pressure
  - **r** - a point (x, y) or (x, y, z)

Fluids

The acceleration for each particle

a = du/dt = F / rho

where **u** is the velocity and **F** is the total force on the particle and **rho** is the mass density

SPH Equations

The General SPH Equation (see Kelager 06 Section 3 for derivation) :

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

- A is the quantity you want to find for a particle (it is the integral interpolant over a delta function, which in this case is approximated using the kernel W)
- m is the mass of a given particle
- W is the kernel function (takes in a distance)
  - h is known as the core radius, or width of the kernel; it controls the smoothness or roughness of the kernel

More SPH

- What does this equation mean?
- A is only a function of **r**, the position, which means that even though we have a discrete set of particles, we can calculate a property at any arbitrary position
- What do we need to solve the equation?
  - Mass of particle, just some control variable
  - Density (how to calculate?)
  - Kernel function W

Calculating Density

- We want the density of the fluid at any given location
- What happens if we plug in density for the quantity A in the original equation?
- We get...

$$\rho_i = \rho(\mathbf{r}_i) = \sum_j m_j \frac{\rho_j}{\rho_j} W(|\mathbf{r}_i - \mathbf{r}_j|, h) = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h),$$

- Easy to calculating knowing mass, positions, and kernel function

Kernel Functions

- They're back again...
- Just a weighting function which determines how you sum up the contributions from the neighbors
- You could use a box if you are lame
- Generally people use Gaussians or cubic splines
    - Concern with speed and locality of neighborhood
- However, you may find that certain quantities give better results with certain kernels

Calculating Forces

Alright we're going to look at the paper because there's a lot of math.

On Fluids

- Now we have the general foundation, how to apply to fluids?
- We need to keep track of position and velocity of each particle, and update these quantities based on the forces from fluid dynamics
- See http://www.inf.ufrgs.br/cgi2007/cd_cgi/papers/harada.pdf for detailed explanation, including how to deal with walls

Applying to NBody

- You already should have a way to keep track of particles (positions and velocities)
- Just need to change your simulation calculation to use these fluid calculations rather than the gravitation equations

Rendering

- Point sprites are all well and good, but how would we render our particle system as a convincing fluid?
- Have to generate some kind of 3D mesh for the fluid surfaces
    - This is actually kind of difficult...

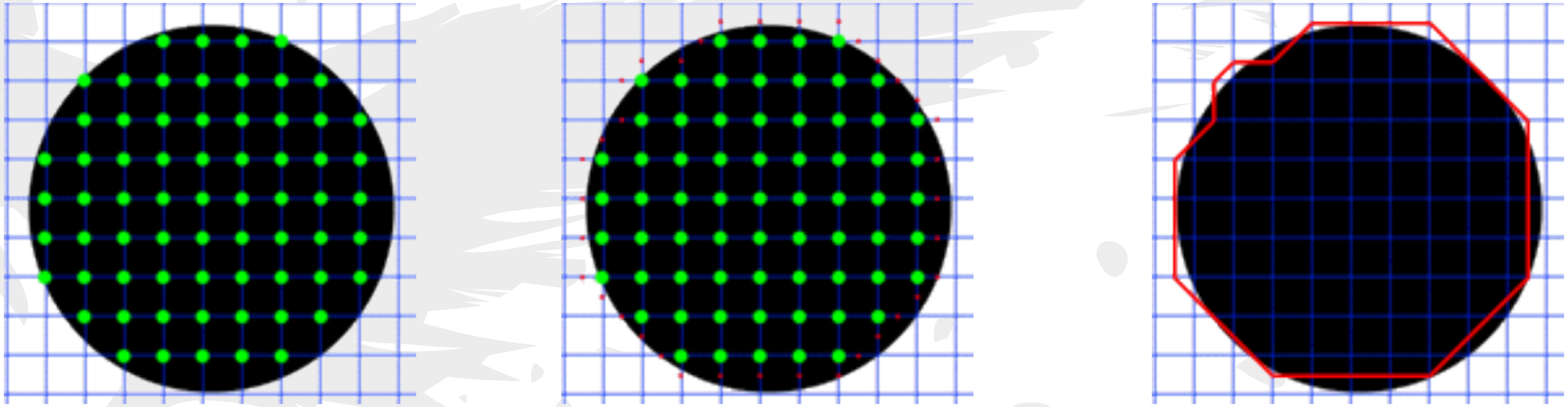
Marching Cubes

Case Study

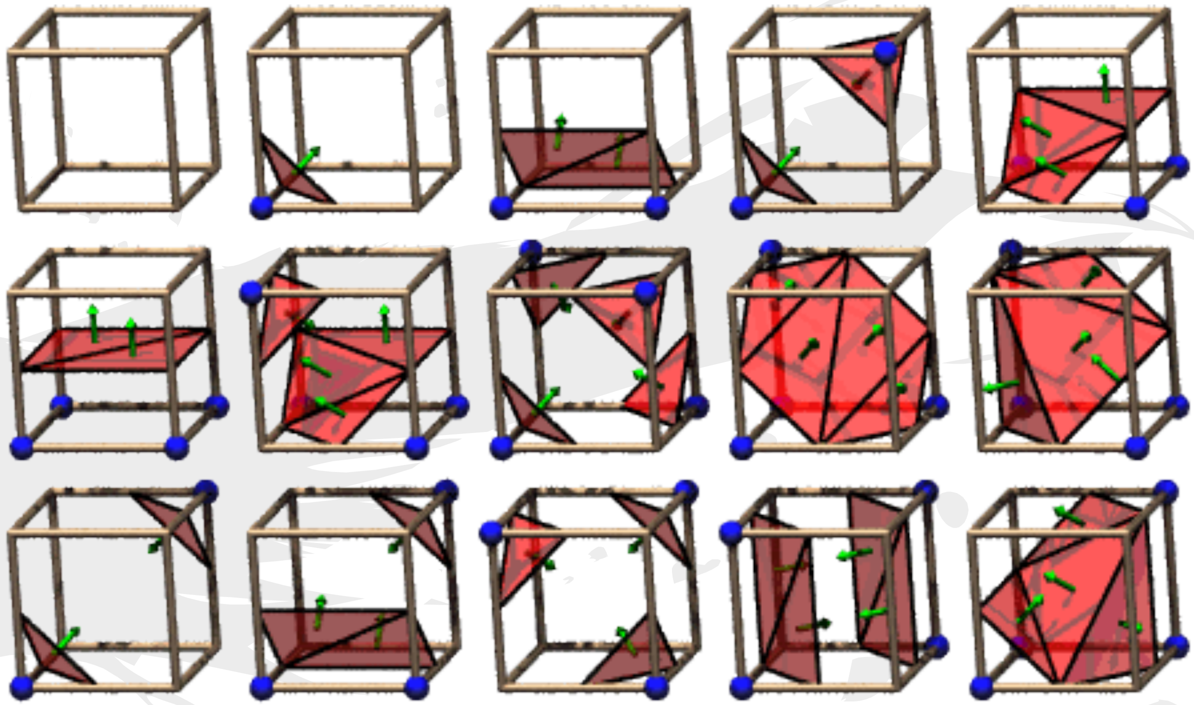
images from http://www.exaflop.org/docs/marchcubes/

Marching Cubes

- First developed in 1987 for visualizing MRI and CT scans
- Generates a 3D mesh from a 3D value field
  - Certain points in the field defined as inside and outside the shape
- For each cube, check which of its vertices are inside and outside the shape, then generate the polygons for the cube based on this configuration

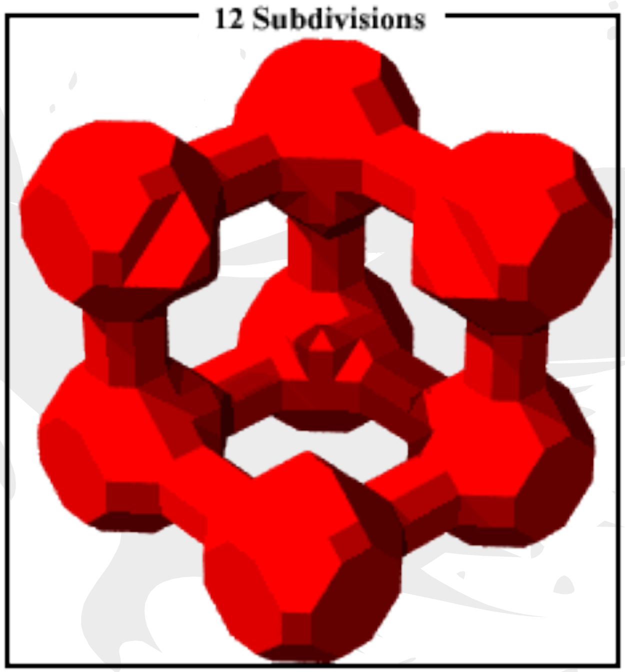


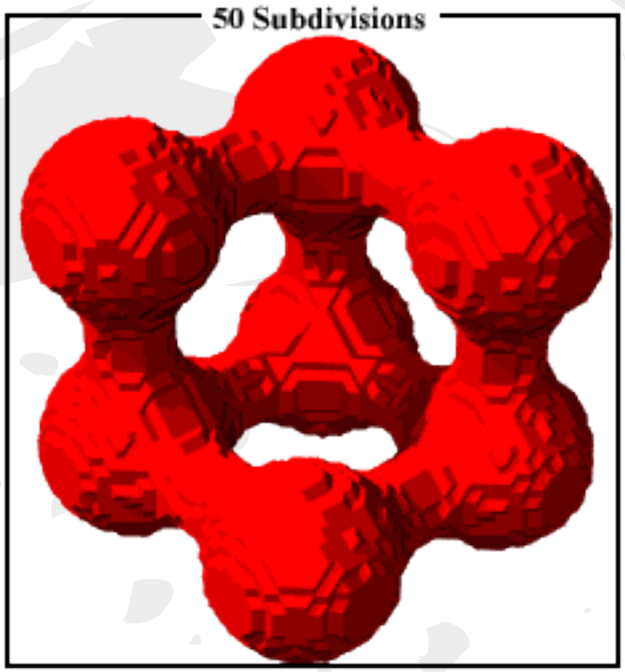
Stupid 2D Example



The 15 Cube Combinations

Polygons generated from different vertex combinations

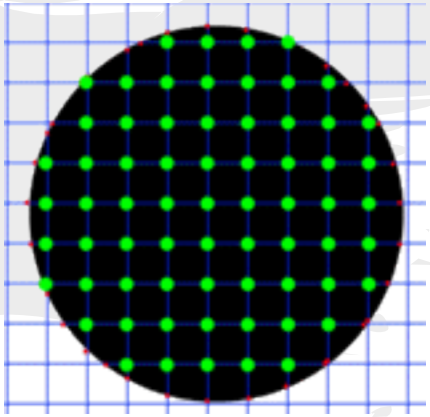


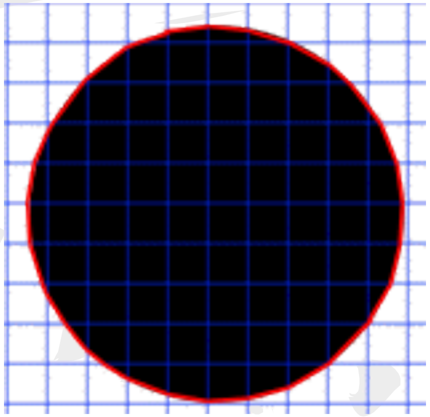


3D example

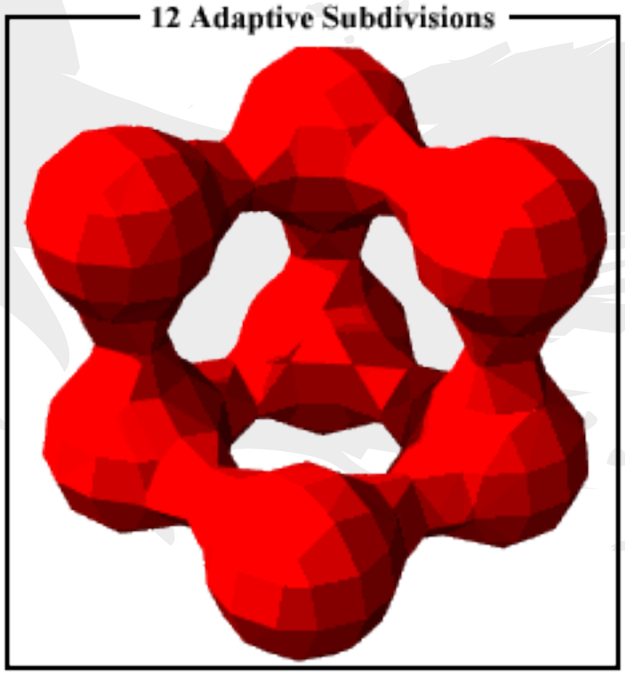
It's all jagged and stuff...

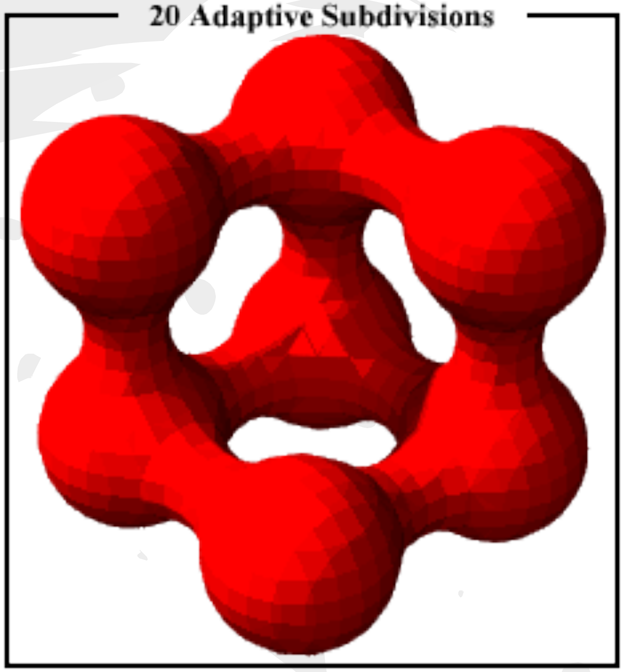
- How do we improve it?
- For each generated polygon vertex, displace it to the actual surface of the shape
- This is called adaptive marching cubes





2D example

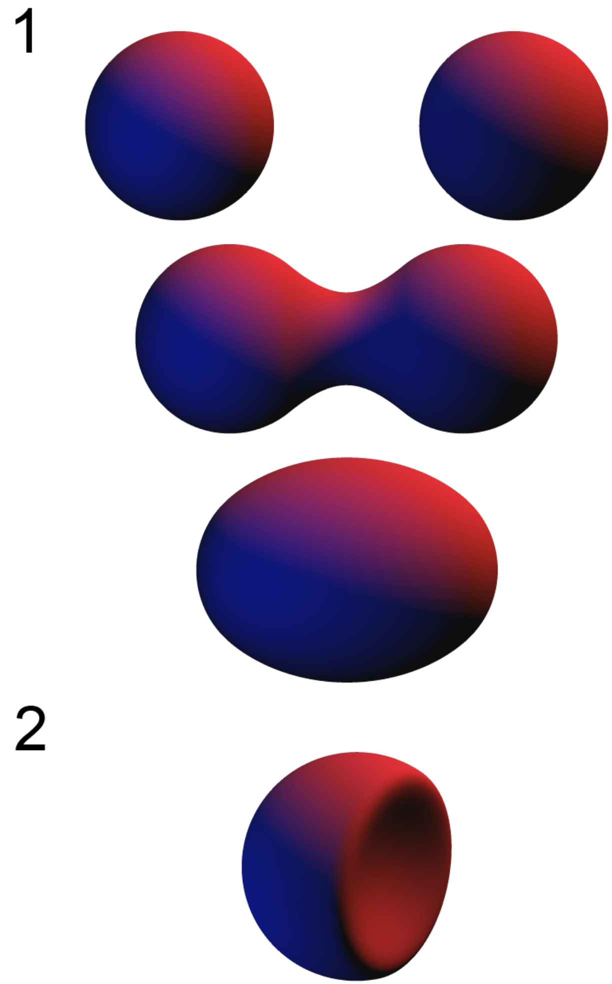




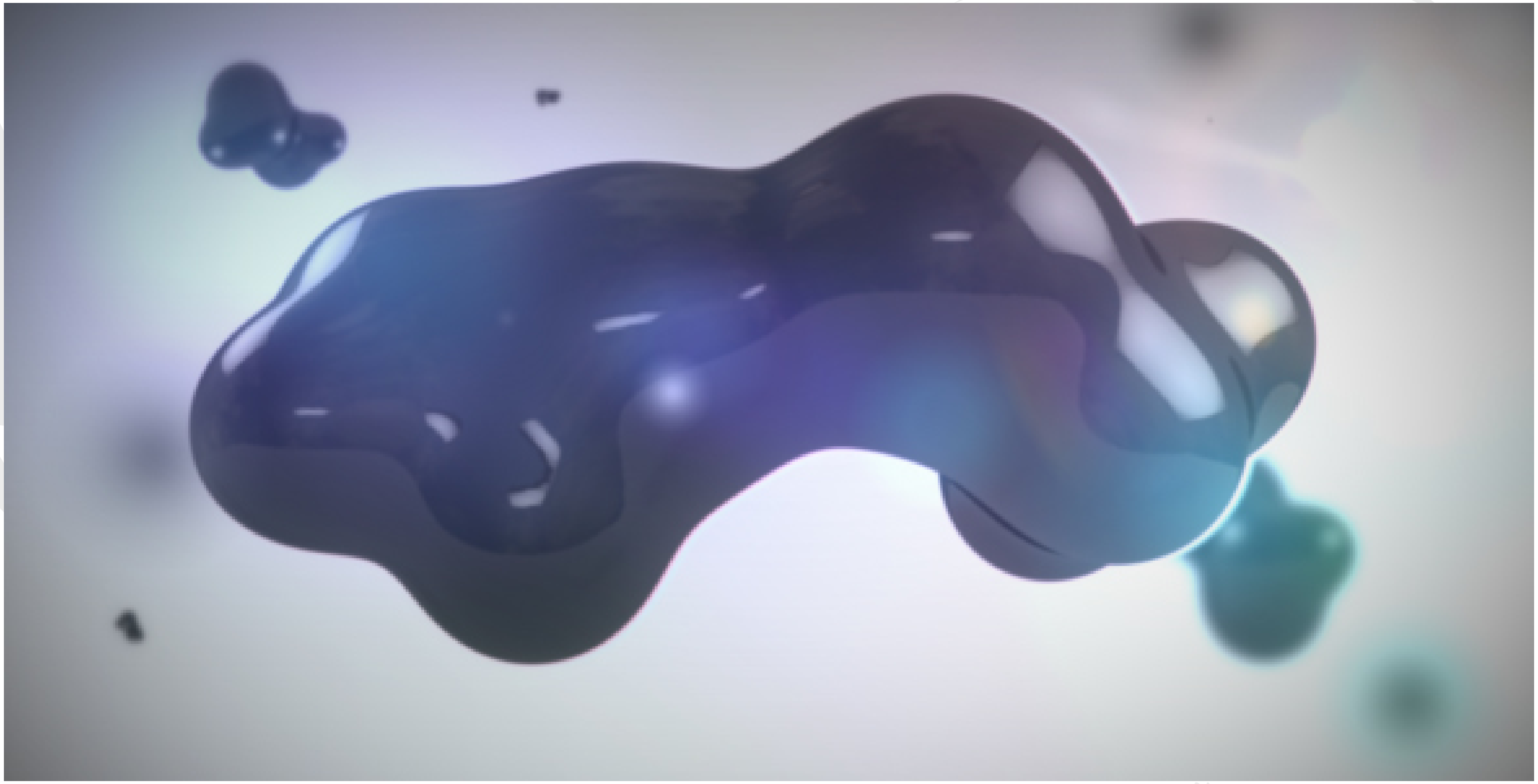
3D example

Marching Cubes on SPH

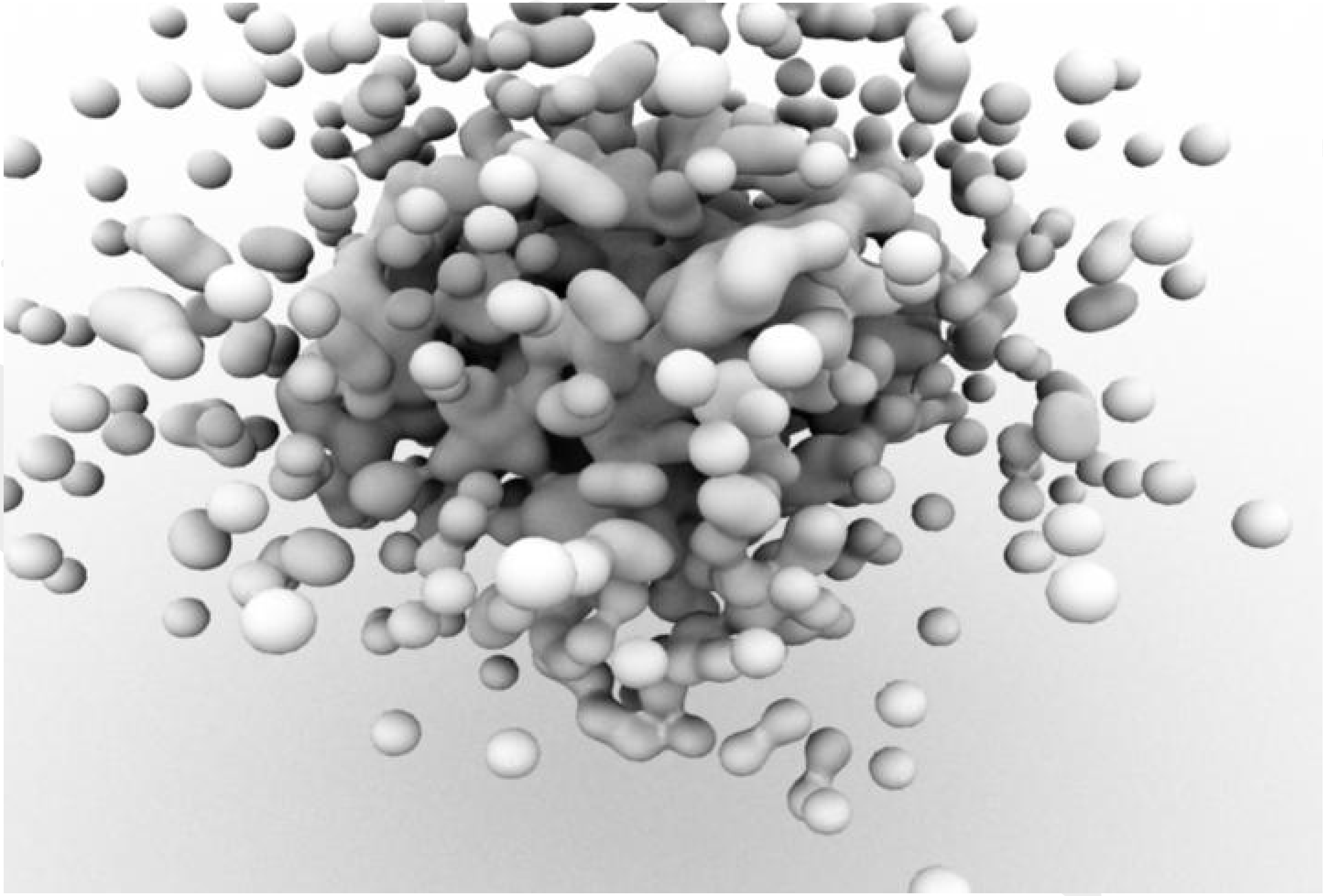
- How to determine if a given point is "inside" the fluid?
- Many methods represent the particles as metaballls
    - A set of metaballs can blend together to form a 3D value function
- Also, a relatively new attempt:
    - http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05613495
    - Render using 3D volume textures and perspective grids



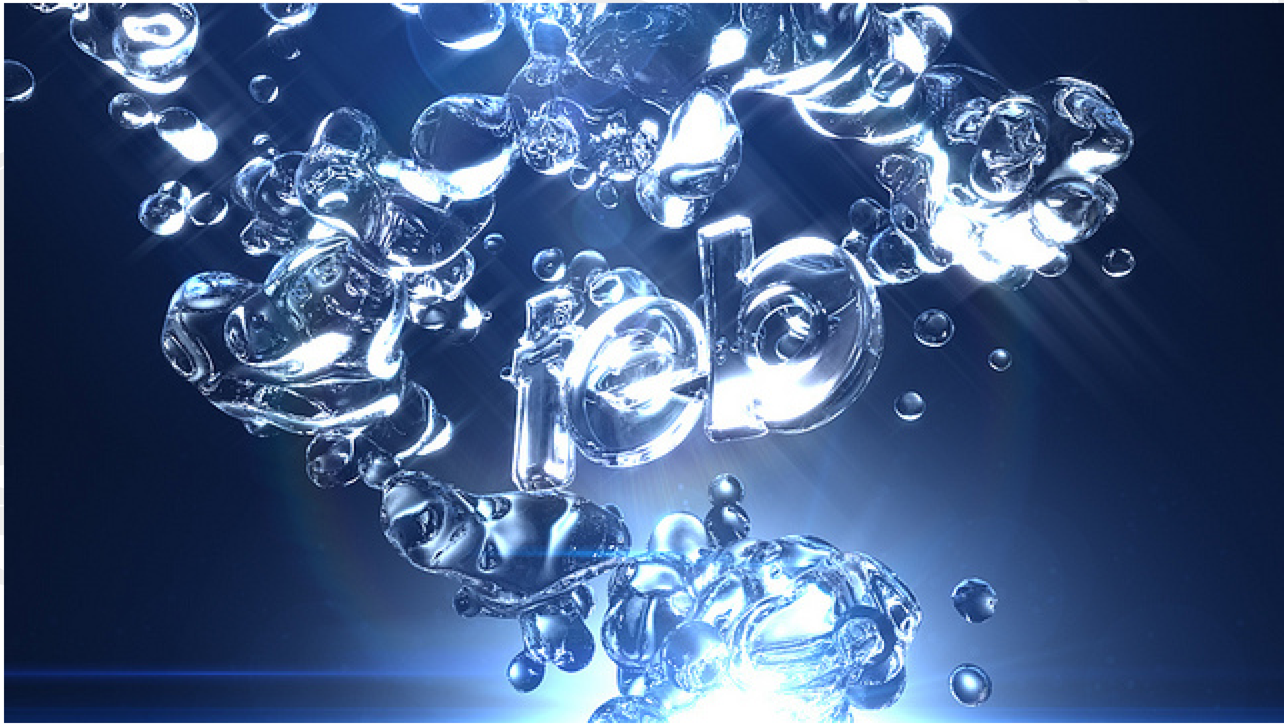
Basic Metaballs



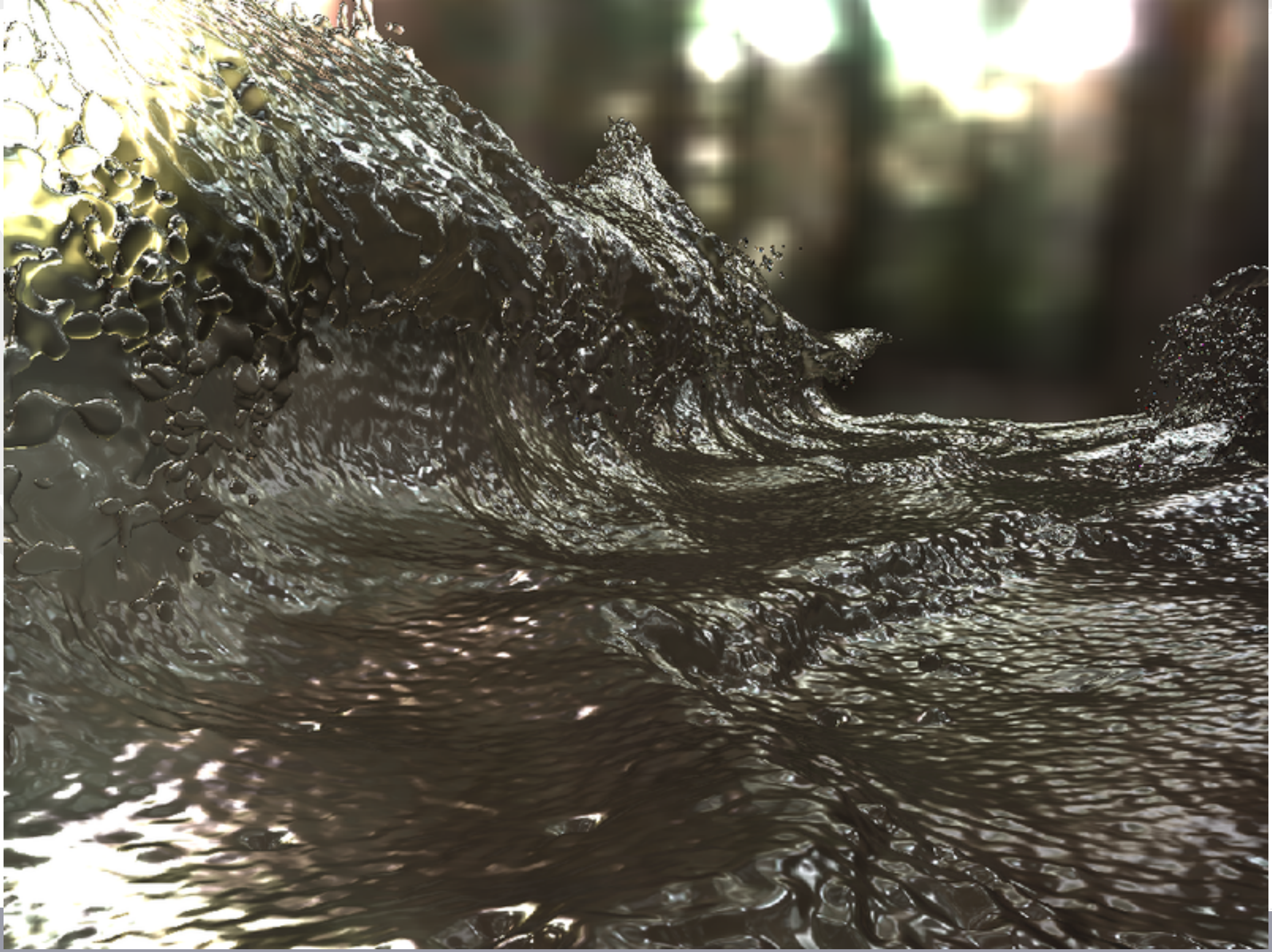
Metaballs



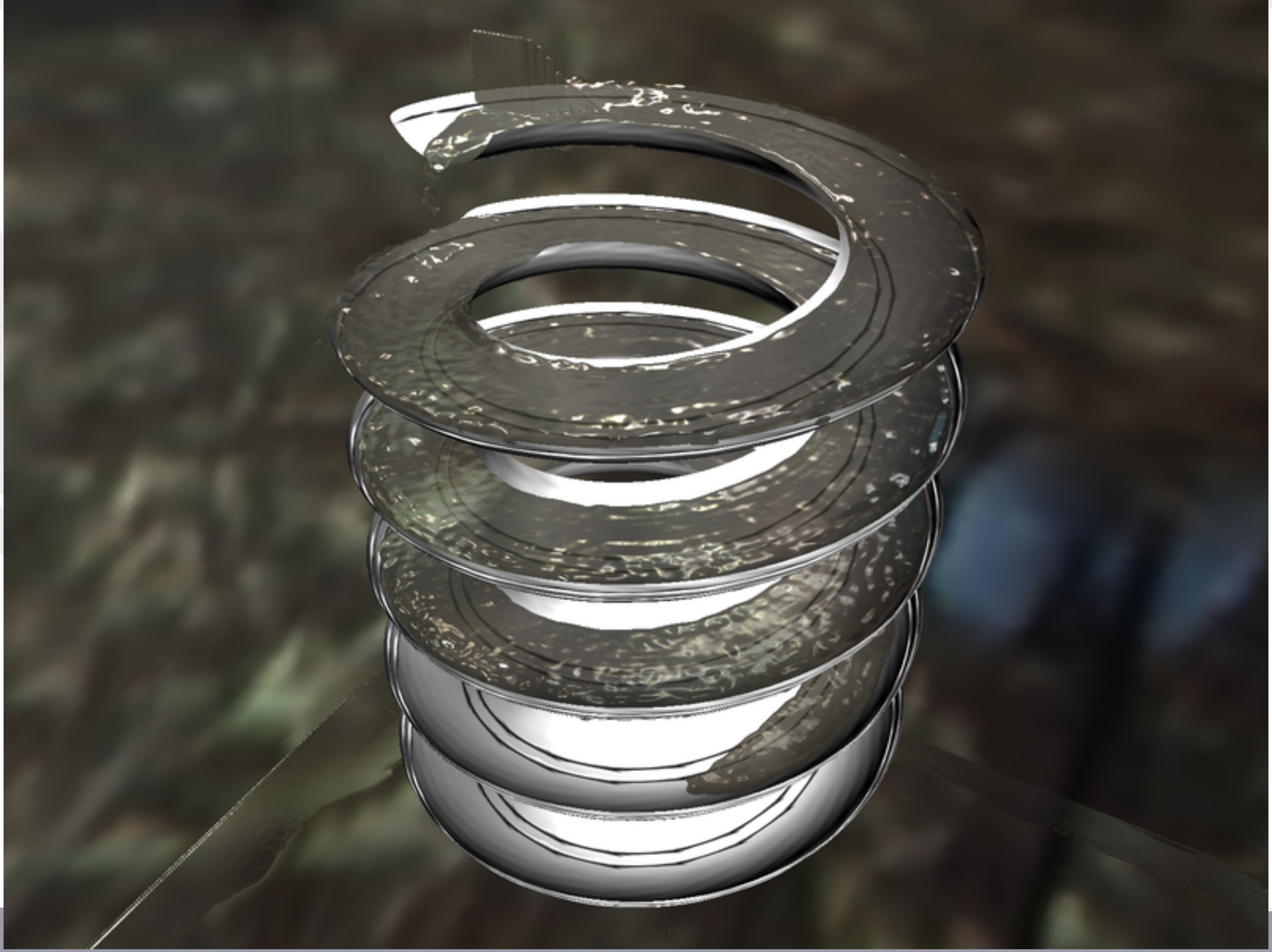
More



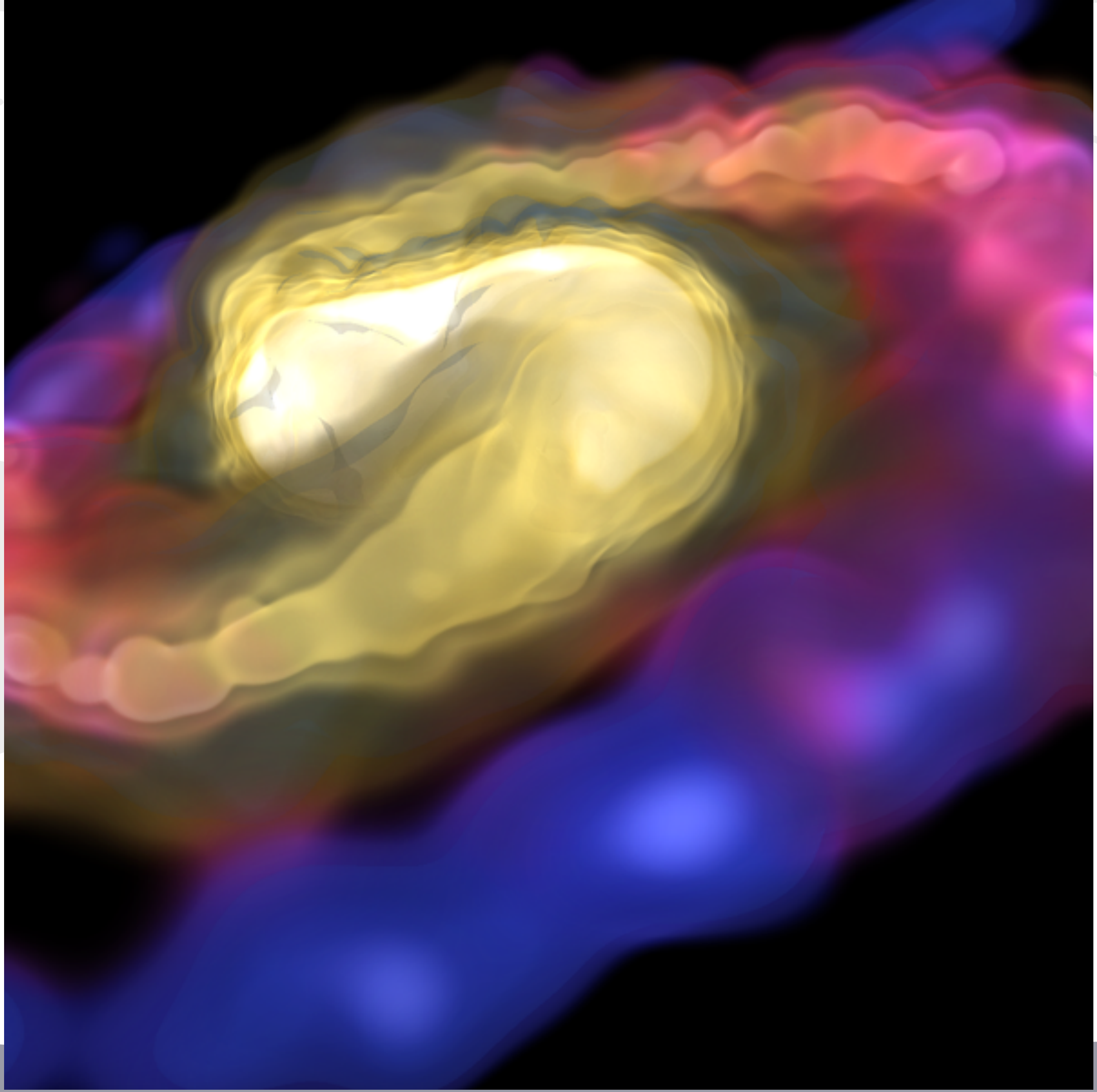
More



Perspective Grid Method



More



More

Screen Space Fluid Rendering?!

Case Study

http://developer.download.nvidia.com/presentations/2010/gdc/Direct3D_Effects.pdf