EROSION OF POINT CLOUDS & SPH

Alec Bernardi
Emilee Reichenbach

OVERVIEW

1. Convert mesh to point cloud offline
2. Smooth particle hydrodynamics fluid simulation
3. Rendering fluid surface with point splatting

CONVERTING MESHES TO POINT CLOUDS

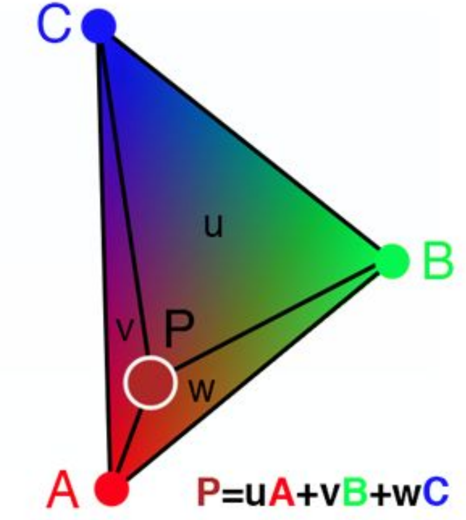
BACKGROUND

- Barycentric coordinates can express position of a point on triangle surface

$$P = uA + vB + wC$$

$$u + v + w = 1$$

- Weighted random sampling makes probability of choosing a triangle proportional to its area



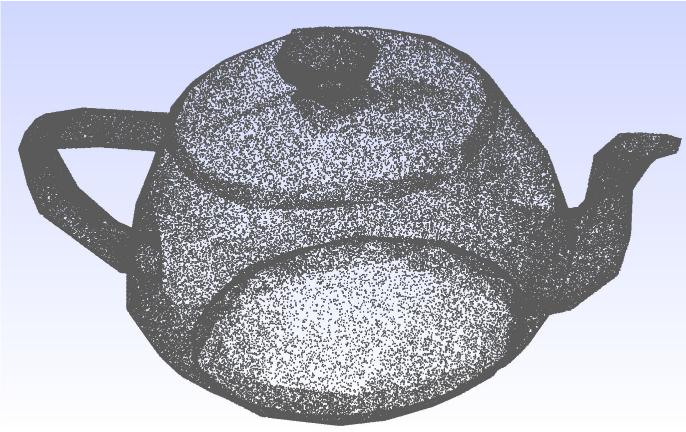
scratchapixel.com

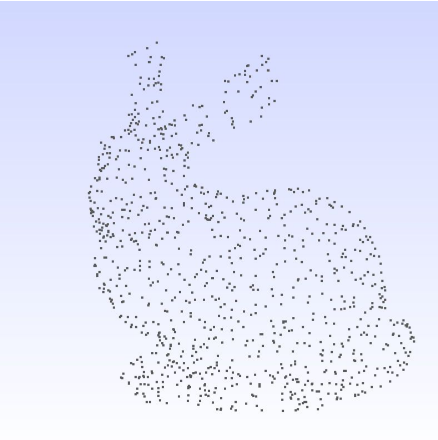
IMPLEMENTATION

Built off of HW 1 code

Input:

- Mesh (.OBJ file)
- Number of samples





1. Weighted random sampling
   - Based on triangle areas
2. Generate random point in triangle
   - Using barycentric coordinates
3. Add new point and normal

WEIGHTED RANDOM SAMPLING

1. Normalize each triangle area
2. Generate a random value from 0 to 99
3. Have a cumulative probability
4. For each triangle
   - Add triangle’s proportional area to cumulative probability
   - If triangle cumulative probability > random value, return that triangle’s index
5. If no triangle is selected, pick a random triangle index

SMOOTHED PARTICLE HYDRODYNAMICS

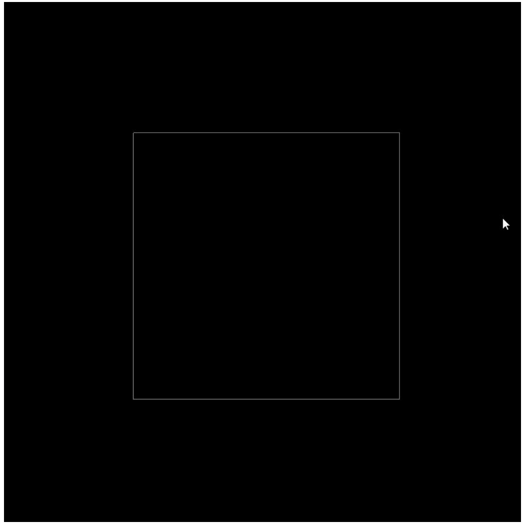
BACKGROUND

Meshfree Lagrangian integration of Navier Stokes:

- No adjacency information
- Approximation by interpolation over nearby particles

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \rho \mathbf{g} + \mu \nabla^2 \mathbf{v}, \qquad (7)$$

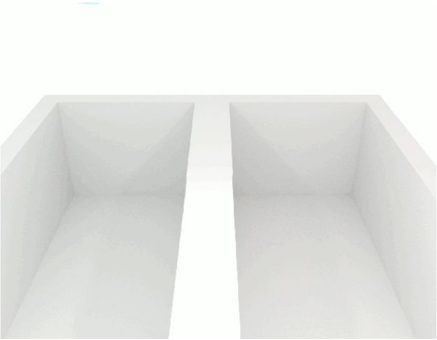
$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{\mathbf{f}_i}{\rho_i}, \qquad (8)$$



FLUID PROPERTIES

Properties that exert forces:

- Pressure
- Viscosity
- Surface tension



Wikipedia

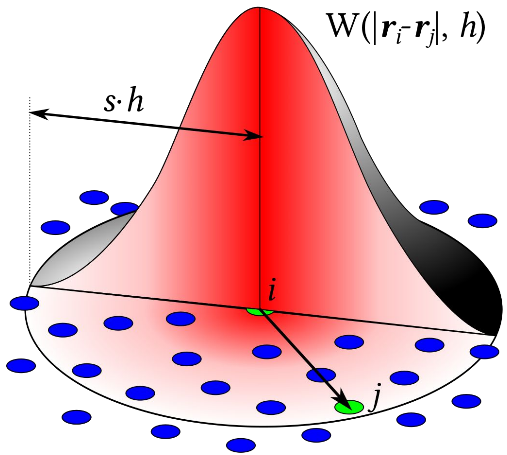
Can be computed with:

- Molar mass
- Rest density
- Rest pressure
- Viscosity
- Speed of sound
- Cohesion

INTERPOLATION

$$A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h), \qquad (1)$$

Wikipedia



Desirable kernel properties:

- Even
- Radially symmetric
- Normalized
- Compact support

$$W_{\text{poly6}}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \le r \le h \\ 0 & \text{otherwise} \end{cases} \qquad (20)$$

$$W_{\text{spiky}}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3 & 0 \le r \le h \\ 0 & \text{otherwise,} \end{cases} \qquad (21)$$

$$W_{\text{viscosity}}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & 0 \le r \le h \\ 0 & \text{otherwise.} \end{cases} \qquad (22)$$

COMPUTING PROPERTIES

$$\rho_S(\mathbf{r}) = \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h). \quad (3)$$

$$p = k(\rho - \rho_0), \quad (12)$$

COMPUTING FORCES

$$\mathbf{f}_i^{\text{pressure}} = -\nabla p(\mathbf{r}_i) = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h). \qquad (9)$$

$$\mathbf{f}_i^{\text{viscosity}} = \mu \nabla^2 \mathbf{v}(\mathbf{r}_a) = \mu \sum_j m_j \frac{\mathbf{v}_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h). \qquad (13)$$

COMPUTING SYMMETRIC FORCES

$$\mathbf{f}_i^{\text{pressure}} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h).. \qquad (10)$$

$$\mathbf{f}_i^{\text{viscosity}} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h). \qquad (14)$$

SURFACE TENSION

$$c_S(\mathbf{r}) = \sum_j m_j \frac{1}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h). \tag{15}$$

$$\mathbf{n} = \nabla c_s \tag{16}$$

$$\mathbf{f}^{\text{surface}} = \sigma \kappa \mathbf{n} = -\sigma \nabla^2 c_S \frac{\mathbf{n}}{|\mathbf{n}|} \tag{19}$$

IMPLEMENTATION

1. Compute per-particle neighborhoods
2. Compute scalar densities and pressures
3. Compute net forces
    a. Also computes surface normal
4. Integrate new positions

COMPUTING NEAREST NEIGHBORS

1. Discretize particle positions onto lattice points
   - Grid size of kernel support radius
2. Sort by flattened index
   - Can use linear radix sort for integer keys
3. Each particle checks 27 neighboring cells

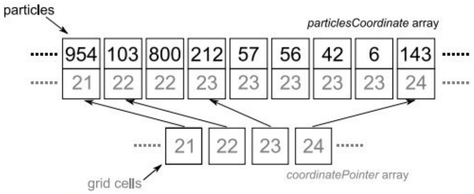


Figure 1. The sorted *particleCoordinate* array and *coordinatePointer* array.

"GPU-based neighbor-search algorithm for particle simulations", Serkan Bayraktar, U˘gur Güdükbay, and Bülent Özgüç

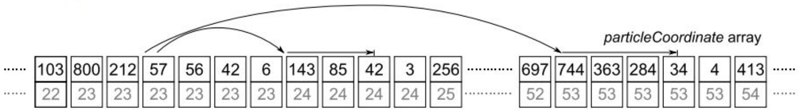


Figure 2. Potential neighbors of particle 57 are being searched.

POINT SPLATTING

BACKGROUND - OVERVIEW

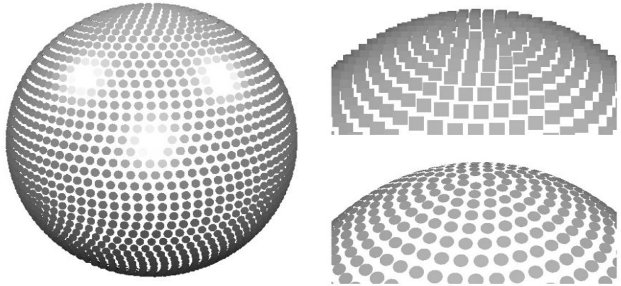
- We have a bunch of surface points
- No adjacency information
- How do we draw the object then?
  - Splatting

"Phong Splatting", Mario Botsch, Michael Spernat, Leif Kobbelt

"High-Quality Point-Based Rendering on Modern GPUs", Mario Botsch, Leif Kobbelt

IMPLEMENTATION

“High-Quality Point-Based Rendering on Modern GPUs”, Mario Botsch, Leif Kobbelt



Input: point cloud data

1. Send data to vertex shader
2. GL_POINTS generates image-space squares
3. Fragment shader projects fragments onto plane defined by splat center and splat normal
4. Compute distance from fragment to splat center
   - If greater than splat radius, then discard
   - Otherwise color fragment with Phong shading

Output: elliptical splats

PERFORMANCE ANALYSIS

Interactive performance

- 2^{15} (~32k) particles at ~6 fps (h=1/20, Δt=0.002)
- 2^{14} (~16k) particles at ~12 fps (h=1/16, Δt=0.002)
- 2^{13} (~8k) particles at ~30 fps (h=1/16, Δt=0.0001)

Heavily CPU-bound

- Parallelization
- GPU only splats

FUTURE WORK

- Import model point clouds
- Extend parallelization with compute shaders
- Surface particle interpolation
- Improve numerical stability
- Multi-phase simulation
  - Rigid-body particles
- Splat filtering and depth correction
- Multipass rendering for splat interpolation

DEMO & QUESTIONS