Notes

- Added a 2D cross-section viewer for assignment 6
  - Not great, but an alternative if the full 3d viewer isn't working for you
- Warning about the formulas in Fedkiw, Stam, and Jensen - maybe not right
  - Rederive the limited Catmull-Rom formulas or check around on the web…
- Please read Foster & Metaxas, "Realistic animation of liquids", 1996
- Thursday: decide your final project!

Water

- This week: extend our 3D flow solver to full 3D water
- We need to add two things:
  - Keep track of where the water is
  - Figure out the right boundary conditions for water surface

Free surface

- As before with waves, we'll ignore what the air is doing
  - Our model of air is pressure=0
- Comparison:
  - ρ_{water}=1000kg/m^3, ρ_{air}=1.3kg/m^3 (approximate, at sea level)
  - Air moves out of the way of water pretty fast!
  - Momentum of air isn't a big deal, pressure variation small
    - Of course, the wind does makes the waves go…
- Instead of 2 phase flow (water+air), we're doing free surface flow (water+vacuum)

Boundary conditions

- All that's new is the free surface boundary (water-"air")
- We know p=0 outside the water
  - So use this BC for the pressure solve
- What about velocity?
  - For figuring out divergence etc.
- Let's think about real water-air interface

Real velocity

- The molecules of water at the interface basically move at the same speed as the molecules of air at the interface
  - Normal to the interface: if moving at different speeds, either compress together or leave a gaping hole…
  - Mathematically this translates to $\partial u/\partial n=0$
  - In tangential direction, things are a little more complicated - applied traction due to viscous stress…
  - Simplify by saying no viscosity, which means tangential components of u not coupled across the boundary

Velocity boundary condition

- If we don't have air, just a free surface, then just extrapolate u
  - u outside = u at closest point inside water
  - Then $\partial u/\partial n=0$, and we get reasonable values for tangential components
- So for example, when we compute divergence near free surface, don't include differences across the interface
  - [draw it]

Tracking the interface

- We know the normal component of u is continuous across interface (so it's well defined on the interface)
- The water and air molecules just on either side then have the same normal velocity
- So interface moves in the normal direction at that speed
- Can the interface move in the tangential direction?
  - No - that doesn't make sense…
  - Ignore the tangential component of velocity

Numerical methods

- Two approaches:
  - "tracking": Lagrangian view point, actually tag material and follow it around
  - "capturing": Eulerian view point, just keep track of whether each grid point is water or not
- Lots of different algorithms…

Parameterized tracking

- Example: see [Foster & Metaxas '96]
- 1st try: use a heightfield again
  - But if heightfield geometry is reasonable, probably 2D physics simplification is fine too
- So then generalize to a parameterized surface
  - E.g. a mesh, or a spline surface, …
  - Delineates the water surface - just what we need for rendering
  - Each vertex of the mesh should move at the speed of the water (extrapolated if needed)

Adaptivity

- Want to start with, e.g., one mesh vertex per surface voxel
- But as water sloshes around, sampling will change
  - Some regions over-resolved - could get numerical noise in mesh
  - Some regions under-resolved - bad bad bad
- Need to resample - delete points, add points, maybe even move points
  - In 1D/2D pretty easy
  - In 2D/3D pretty hard (but do-able: 533A)

Topology changes

- When a wave crashes down, or a drop hits, or a drop separates, or…
  - Topology changes
  - Old parameterization does not apply
- Need to detect collisions, reparameterize (mesh surgery)
  - 1D/2D: painful, but do-able
  - 2D/3D: you don't want to go there
- Bottom-line: parameterized surfaces are not a good idea for interface tracking

Phase-field

- Mathematically could define characteristic function $\chi(x)$
  - 1 for water, 0 for air
- Discretize this on a grid θ_{ijk}, advect it around in the velocity field like any other scalar
  - Called a phase field (tells us which phase)

$$\theta_t + u \cdot \nabla \theta = 0$$

- Two immediate problems:
  - Stair-step problem (smooth water surface is now voxelized)
  - Initial discontinuity gets blurred out - we lose 0/1 values

Fixing phase fields

- Smearing things out is actually good!
  - Around interface go smoothly from 0 to 1
  - Pick 1/2 to be the threshhold for what is water, what is not
  - Render smooth implicit surface
- How much smearing?
  - Say 2-3 grid cells…
- Problems:
  - Over time, smearing spreads and gets distorted
  - Mass is not conserved discretely

Level sets

- Naturally leads to level set method
- Now use signed distance on a grid, with $\phi=0$ marking the interface
- We know exactly how much "smearing": we want $|\nabla\phi|=1$
- Interface is always sharply defined
- Move it around as before:

$$\phi_t + u \cdot \nabla\phi = 0$$

- But problems remain:
  - Over time, signed distance gets distorted
  - Mass isn't guaranteed to be conserved

Interface velocity

- Remember the interface only cares about normal component of velocity
- It also only cares about velocity at the interface
  - But Eulerian schemes move entire field using velocity everywhere…
- Significantly improve level set method by changing velocity field
  - Just keep normal component of velocity from closest point on interface

$$u_{LS}(x) = u(x - \phi\nabla\phi) \cdot \hat{n}\,\hat{n}$$

Distortion

- This delays, but doesn't stop, the problem of signed distance getting distorted
  - If it's distorted too much, get very unreliable normals and closest point estimates…
- But remember: we only care about interface
  - Values of ϕ far from interface may be changed for our convenience without changing interface: there's nothing sacred about them
- Thus we need to reinitialize ϕ to be signed distance

Reinitialization

- Idea: we have a distorted ϕ, $|\nabla\phi| \neq 1$
- Want to return to $|\nabla\phi|=1$ without disturbing the location of the interface
- If we're not too far from $|\nabla\phi|=1$, makes sense to use an iterative method
  - We can even think of each iteration as a pseudo-time step
  - Information should flow outward from interface
  - Advection in direction sign(ϕ)n and with rate of change sign(ϕ):

$$\phi_t + \left(sign(\phi) \frac{\nabla \phi}{|\nabla \phi|} \right) \cdot \nabla \phi = sign(\phi)$$

Reinitialization cont'd

- Simplifying this we get:

$$\phi_t + \left(sign(\phi) - 1 \right) |\nabla \phi| = 0$$

- This is another Hamilton-Jacobi equation…
  - If we want $|\nabla\phi|=1$ to very high order accuracy, can use high-order HJ methods

Discretization

- When we discretize (e.g. with semi-Lagrangian) we'll end up interpolating with values on either side of interface
- Limit the possibility for weird stuff to happen, like ϕ changing sign
- So instead of sign(ϕ), use $S(\phi_0)$
  - Can never flip sign
  - Sign function smeared out to be smooth:

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + |\nabla \phi_0|^2 (\Delta x)^2}}$$

Aside: initialization

- This works well if we're already close to signed distance
- What if we start from scratch at t=0?
  - For very simple geometry, may construct ϕ analytically
  - More generally, need to numerically approximate
- One solution - if we can at least get inside/outside on the grid, can run reinitialization equation from there (1st order accurate)

Fast methods

- Problem with reinitialization from scratch - to get full field, need to take O(n) steps, each costs $O(n^3)$
- Can speed up with local level set method
  - Only care about signed distance near interface, so only compute those $O(n^2)$ values in O(1) steps
  - Gives optimal $O(n^2)$ complexity (but the constant might be big!)
- If we really want full grid, but fast:
  - Fast Marching Method $O(n^3 \log n)$
  - Fast Sweeping Method $O(n^3)$
  - But not very accurate

Velocity extrapolation

- We can exploit level set to extrapolate velocity field outside water
  - Not a big deal for pressure solve - can directly handle extrapolation there
  - But a big deal for advection - with semi-Lagrangian method might be skipping over, say, 5 grid cells
  - So might want velocity 5 grid cells outside of water
- Simply take the velocity at an exterior grid point to be interpolated velocity at closest point on interface

Pure level set algorithm

- Use current ϕ (for velocity extrapolation and pressure-solve boundary conditions) to get next velocity field
- Advect ϕ
- Every so often (20 time steps?) reinitialize ϕ for a few (5?) pseudo-time steps

Mass conservation

- Problem: it doesn't work
- Visual artifacts - water droplets vanish in mid-air
- Mass is not guaranteed to be conserved
- Reinitialization makes it even worse
- [example]
- In the limit, works ok, but not on coarse grids
  - Even if we use 5th order accurate HJ-WENO…

Volume-of-fluid (VOF)

- Another Eulerian approach: directly enforce conservation of mass
  - Account for every last drop of water
- At each grid cell, keep track of how much water is in it (as a fraction of the cell): 0=empty, 1=full
  - Like phase-field, only physical meaning for intermediate values
- Treat advection as a conservation law - make sure water is conserved

VOF problems

- Discontinuous interface (which should be handled by p=0, extrapolated u) is smeared out and made erroneously continuous
- Hard to figure out what to do with accumulation of partially-filled cells
- Hard to reconstruct nice interface, e.g. for rendering
  - [draw it]

Back to particles!

- Harlow and Welch, 1965: MAC method
  - Marker-and-Cell
- Instead of moving surface particles around, move water particles ("marker particles")
- Forget about a mesh
  - Only need to know where water is and isn't (worry about rendering later - e.g. blobby implicit surface wrapped around particles)
- Any grid cell with marker particles in it is water, rest are not

MAC

- Seed particles in grid cells where there is water (e.g. 8 to a grid cell in 3D)
- Mark grid cells as water/air according to whether or not they have particles
- Solve for new velocity/pressure
- Move particles in velocity field
  - Need CFL limit for accuracy

Issues

- Mass conservation?
  - Not exact, but close - if velocity field is divergence-free
  - Can never lose water in mid-air
- Smearing, distortion? Doesn't apply
- The only downside is noisy surface
  - Discrete particles don't do a good job at representing smooth water
  - But great for rough foamy splashing!

Surface tension

- Critical for small-scale water
- We model it by adding to pressure boundary condition p=0 at free surface:

$$p_{fs} = \sigma\kappa$$

  - σ is surface tension parameter, κ is mean curvature
  - Recall κ is based on second derivatives of surface
- If we have a noisy surface from blobby-wrapped marker particles, curvature estimate is extremely noisy - useless