Fire

- [Nguyen, Fedkiw, Jensen ‘02]
- Gaseous fuel/air mix (from a burner, or a hot piece of wood, or …) heats up
- When it reaches ignition temperature, starts to burn
  - “blue core” - see the actual flame front due to emission lines of excited hydrocarbons
- Gets really hot while burning - glows orange from blackbody radiation of smoke/soot
- Cools due to radiation, mixing
  - Left with regular smoke

Defining the flow

- Inside and outside blue core, regular incompressible flow with buoyancy
- But an interesting boundary condition at the flame front
  - Gaseous fuel and air chemically reacts to produce a different gas with a different density
  - Mass is conserved, so volume has to change
  - Gas instantly expands at the flame front
- And the flame front is moving too
  - At the speed of the flow plus the reaction speed

Interface speed

- Interface = flame front = blue core surface
- $D = V_r - S$ is the speed of the flame front
  - It moves with the fuel flow, and on top of that, moves according to reaction speed $S$
  - $S$ is fixed for a given fuel mix
- We can track the flame front with a level set $\phi$
  - Level set moves by
    \[
    \phi_t + D |\nabla \phi| = 0 \\
    \phi_t + u_{LS} \cdot \nabla \phi = 0
    \]
- Here $u_{LS}$ is $u_r - S_n$
Numerical method

- For water we had to work hard to move interface accurately
- Here it’s ok just to use semi-Lagrangian method (with reinitialization)
- Why?
  - We’re not conserving volume of blue core - if reaction is a little too fast or slow, that’s fine
  - Numerical error looks like mean curvature
  - Real physics actually says reaction speed varies with mean curvature!

Conservation of mass

- Mass per unit area entering flame front is \( \rho_f(V_f-D) \) where
  - \( V_f=\mathbf{u}_f \cdot \mathbf{n} \) is the normal component of fuel velocity
  - \( D \) is the (normal) speed of the interface
- Mass per unit area leaving flame front is \( \rho_h(V_h-D) \) where
  - \( V_h=\mathbf{u}_h \cdot \mathbf{n} \) is the normal component of hot gaseous products velocity
- Equating the two gives:
  \[
  \rho_f(V_f - D) = \rho_h(V_h - D)
  \]

Velocity jump

- Plugging interface speed \( D \) into conservation of mass at the flame front gives:
  \[
  \rho_f S = \rho_h(V_h - V_f + S)
  \]
  \[
  \rho_h V_h = \rho_h V_f + \rho_f S - \rho_h S
  \]
  \[
  V_h = V_f + \left( \frac{\rho_f}{\rho_h} - 1 \right) S
  \]

Ghost velocities

- This is a “jump condition”: how the normal component of velocity jumps when you go over the flame interface
- This lets us define a “ghost” velocity field that is continuous
  - When we want to get a reasonable value of \( \mathbf{u}_h \) for semi-Lagrangian advection of hot gaseous products on the fuel side of the interface, or vice versa (and also for moving interface)
  - When we compute divergence of velocity field
- Simply take the velocity field, add/subtract \((\rho_f/\rho_h-1)S\)
Conservation of momentum

- Momentum is also conserved at the interface
- Fuel momentum per unit area "entering" the interface is \( \rho_f V_f (V_f - D) + p_f \)
- Hot gaseous product momentum per unit area "leaving" the interface is \( \rho_h V_h (V_h - D) + p_h \)
- Equating the two gives
  \[
  \rho_f V_f (V_f - D) + p_f = \rho_h V_h (V_h - D) + p_h
  \]

Simplifying

- Make the equation look nicer by taking conservation of mass:
  \[
  \rho_f (V_f - D) = \rho_h (V_h - D)
  \]
  multiplying both sides by -D:
  \[
  \rho_f (-D)(V_f - D) = \rho_h (-D)(V_h - D)
  \]
  and adding to previous slide’s equation:
  \[
  \rho_f (V_f - D)^2 + p_f = \rho_h (V_h - D)^2 + p_h
  \]

Pressure jump

- This gives us jump in pressure from one side of the interface to the other
- By adding/subtracting the jump, we can get a reasonable continuous extension of pressure from one side to the other
  - For taking the gradient of \( p \) to make the flow incompressible after advection
- Note when we solve the Poisson equation density is NOT constant, and we have to incorporate jump in \( p \) (known) just like we use it in the pressure gradient

Temperature

- We don’t want to get into complex (!) chemistry of combustion
- Instead just specify a time curve for the temperature
  - Temperature known at flame front (\( T_{\text{ignition}} \))
  - Temperature of a chunk of hot gaseous product rises at a given rate to \( T_{\text{max}} \) after it’s created
  - Then cools due to radiation
**Temperature cont’d**

- For small flames (e.g. candles) can model initial temperature rise by tracking time since reaction: \( Y_t + u \cdot \nabla Y = 1 \) and making \( T \) a function of \( Y \)
  - For large flames ignore rise, just start flame at \( T_{\text{max}} \) (since transition region is very thin, close to blue core)
- Radiative cooling afterwards:
  \[
  T_t + u \cdot \nabla T = -c_f \left( \frac{T - T_{\text{air}}}{T_{\text{max}} - T_{\text{air}}} \right)^eta
  \]

**Smoke concentration**

- Can do the same as for temperature: initially make it a function of time \( Y \) since reaction (rising from zero)
  - And ignore this regime for large flames
- Then just advect without change, like before
- Note: both temperature and smoke concentration play back into velocity equation (buoyancy force)

**Note on fuel**

- We assumed fuel mix is magically being injected into scene
  - Just fine for e.g. gas burners
  - Reasonable for slow-burning stuff (like thick wood)
- What about fast-burning material?
  - Can specify another reaction speed \( S_{\text{fuel}} \) for how fast solid/liquid fuel turned into flammable gas (dependent on temperature)
  - Track level set of solid/liquid fuel just like we did the blue core

**SPH**

- Smoothed Particle Hydrodynamics
  - A particle system approach
- Get rid of the mesh altogether - figure out how to do \( \nabla p \) etc. with just particles
  - Each particle represents a blurry chunk of fluid (with a particular mass, momentum, etc.)
  - Lagrangian: advection is going to be easy
Mesh-free?

- Mathematically, SPH and particle-only methods are independent of meshes
- Practically, need an acceleration structure to speed up finding neighbouring particles (to figure out forces)
- Most popular structure (for non-adaptive codes, i.e. where $h=\text{constant}$ for all particles) is a mesh (background grid)

SPH

- SPH can be interpreted as a particular way of choosing forces, so that you converge to solving Navier-Stokes
- [Lucy’77], [Gingold & Monaghan ‘77], [Monaghan…], [Morris, Fox, Zhu ‘97], …
- Similar to FEM, we go to a finite dimensional space of functions
  - Basis functions now based on particles instead of grid elements
  - Can take derivatives etc. by differentiating the real function from the finite-dimensional space

Kernel

- Need to define particle’s influence in surrounding space (how we’ll build the basis functions)
- Choose a kernel function $W$
  - Smoothed approximation to $\delta$
  - $W(x)=W(|x|)$ - radially symmetric
  - Integral is 1
  - $W=0$ far enough away - when $|x|>2.5h$ for example
- Examples:
  - Truncated Gaussian
  - Splines (cubic, quartic, quintic, …)

Cubic kernel

- Use
  \[ W(x) = \frac{1}{h^3} f \left( \frac{|x|}{h} \right) \] where
  \[ f(s) = \frac{1}{\pi} \begin{cases} 1 - \frac{3}{4} s^2 + \frac{3}{4} s^3, & 0 \leq s \leq 1 \\ \frac{1}{4} (2 - s)^3, & 1 \leq s \leq 2 \\ 0, & 2 \leq s \end{cases} \]
- Note: not good for viscosity (2nd derivatives involved - not very smooth)
Estimating quantities

- Say we want to estimate some flow variable \( q \) at a point in space \( x \)
- We’ll take a mass and kernel weighted average
- Raw version:
  \[
  Q(x) = \sum_j m_j q_j W(x - x_j)
  \]
  - But this doesn’t work, since sum of weights is nowhere close to 1
  - Could normalize by dividing by \( \sum_j m_j W_j \) but that involves complicates derivatives...
  - Instead use estimate for normalization at each particle separately (some mass-weighted measure of overlap)

\[
Q(x) = \sum_j m_j q_j W(x - x_j)
\]

Smoothed Particle Estimate

- Take the “raw” mass estimate to get density:
  \[
  \langle \rho(x) \rangle = \sum_j m_j W(x - x_j)
  \]
  - [check dimensions]
- Evaluate this at particles, use that to approximately normalize:
  \[
  \langle q(x) \rangle = \sum_j q_j \frac{m_j W(x - x_j)}{\rho_j}
  \]

Incompressible Free Surfaces

- Actually, I lied
  - That is, regular SPH uses the previous formulation
  - For doing incompressible flow with free surface, we have a problem
  - Density drop smoothly to 0 around surface
  - This would generate huge pressure gradient, compresses surface layer
- So instead, track density for each particle as a primary variable (as well as mass!)
  - Update it with continuity equation
  - Mass stays constant however - probably equal for all particles, along with radius

Continuity equation

- Recall the equation is
  \[
  \rho_t + u \cdot \nabla \rho = -\rho \nabla \cdot u
  \]
- We’ll handle advection by moving particles around
- So we need to figure out right-hand side
- Divergence of velocity for one particle is
  \[
  \nabla \cdot v = \nabla \cdot (v_j W(x - x_j)) = v_j \cdot \nabla W_j
  \]
- Multiply by density, get SPH estimate:
  \[
  \langle \rho \nabla \cdot v \rangle_j = \sum_i m_i v_i \cdot \nabla W_{ij}
  \]
Momentum equation

- Without viscosity: \( u_t + u \cdot \nabla u = -\frac{1}{\rho} \nabla p + g \)
- Handle advection by moving particles
- Acceleration due to gravity is trivial
- Left with pressure gradient
- Naïve approach - just take SPH estimate as before

\[
\frac{dv_i}{dt} = \left\langle -\frac{1}{\rho} \nabla p \right\rangle = -\sum_j m_j \frac{p_j}{\rho_j^2} \nabla W_{ij}
\]

Conservation of momentum

- Remember momentum equation really came out of \( F=ma \) (but we divided by density to get acceleration)
- Previous slide - momentum is not conserved
  - Forces between two particles is not equal and opposite
- We need to symmetrize this somehow

\[
\frac{dv_i}{dt} = -\sum_j m_j \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij}
\]
- [check symmetry - also note angular momentum]

SPH advection

- Simple approach: just move each particle according to its velocity
- More sophisticated: use some kind of SPH estimate of \( v \)
  - keep nearby particles moving together
- XSPH

\[
\frac{dx_i}{dt} = v_i + \sum_j m_j \left( \frac{v_j - v_i}{\rho_i + \rho_j} \right) W_{ij}
\]

Equation of state

- Some debate - maybe need a somewhat different equation of state if free-surface involved
- E.g. [Monaghan’94]

\[
p = B \left( \frac{\rho}{\rho_0} \right)^\gamma - 1
\]
- For small variations, looks like gradient is the same [linearize]
  - But SPH doesn’t estimate -1 exactly, so you do get different results…
The End

- But my lifetime guarantee: you can ask me questions anytime about numerical physics stuff…