Navier-Stokes
&
Flow Simulation

Last Time?

Today

- Flow Simulations in Computer Graphics
  - water, smoke, viscous fluids
- Navier-Stokes Equations
- Fluid Representations
- Basic Algorithm
- Data Representation

Flow Simulations in Graphics

- Random velocity fields
  - with averaging to get simple background motion
- Shallow water equations
  - height field only, can't represent crashing waves, etc.
- Full Navier-Stokes

- *note: typically we ignore surface tension and focus on macroscopic behavior*

Today

- Flow Simulations in Computer Graphics
- Navier-Stokes Equations
  - incompressibility, conservation of mass
  - conservation of momentum & energy
- Fluid Representations
- Basic Algorithm
- Data Representation

Flow in a Voxel Grid

- conservation of mass:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

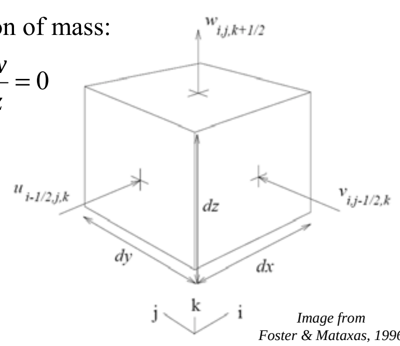


Image from Foster & Mataxas, 1996

Navier-Stokes Equations

- conservation of momentum:

gravity (& other external forces)

pressure viscosity

$$\begin{aligned}
\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} &= -\frac{\partial p}{\partial x} + g_x + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) \\
\frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial v^2}{\partial y} + \frac{\partial vw}{\partial z} &= -\frac{\partial p}{\partial y} + g_y + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) \\
\underbrace{\frac{\partial w}{\partial t}}_{\text{acceleration}} + \underbrace{\frac{\partial wu}{\partial x} + \frac{\partial wv}{\partial y} + \frac{\partial w^2}{\partial z}}_{\text{convection}} &= -\frac{\partial p}{\partial z} + g_z + \nu\underbrace{\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right)}_{\text{drag}}
\end{aligned}$$

Questions?

Today

- Flow Simulations in Computer Graphics
- Navier-Stokes Equations
- Fluid Representations
- Basic Algorithm
- Data Representation

Modeling the Water Surface

- Volume-of-fluid tracking
  - a scalar saying how "full" each cell is
- Particle In Cell (PIC)
  - the particles have mass
- Marker and Cell (MAC)
  - the particles don't effect computation, just identify which cells the surface passes through
  - Harlow & Welch, "Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface", *The Physics of Fluids*, 1965.

PIC method (2-dimensional): At time t_n the state of the fluid is represented by N particles with the property vectors $\mathbf{u}_k^n \equiv \{\mathbf{r}_k^n, \mathbf{v}_k^n, c_k^n\}$ $(k = 1, \dots N)$. In each Eulerian cell of side length Δl there should be at least ≈ 100 particles.

1. Compute, for each Euler cell (i, j), the cell properties

$$\begin{aligned}
\rho_{i,j}^n &= \frac{m}{(\Delta l)^2} \sum_{k=1}^{N} \delta\left[\mathbf{r}_k^n(i,j)\right] \\
(\rho\mathbf{v})_{i,j}^n &= \frac{m}{(\Delta l)^2} \sum_{k=1}^{N} \mathbf{v}_k^n \delta\left[\mathbf{r}_k^n(i,j)\right] \\
(\rho c)_{i,j}^n &= \frac{m}{(\Delta l)^2} \sum_{k=1}^{N} c_k^n \delta\left[\mathbf{r}_k^n(i,j)\right]
\end{aligned}$$

2. Using the equation of state to evaluate cell pressures $p_{i,j}$, compute new (preliminary) flow velocities according to

$$\begin{aligned}
v_{x,i,j}^{n+1} &= v_{x,i,j}^n - a\left(p_{i+1,j}^n - p_{i-1,j}^n\right) \\
v_{y,i,j}^{n+1} &= v_{y,i,j}^n - a\left(p_{i,j+1}^n - p_{i,j-1}^n\right)
\end{aligned}$$

with $a \equiv \Delta t / 2(\Delta l)\rho_{i,j}^n$. For each fluid particle k we now have $\mathbf{v}_k^{n+1} = \mathbf{v}_{i,j}^{n+1}$ and $c_k^{n+1} = p_{i,j}^n / (\rho_{i,j}^n)^\gamma$.

3. From the time-centered cell velocities $\mathbf{v}_{i,j}^{n+1/2} \equiv \left[\mathbf{v}_{i,j}^{n+1} + \mathbf{v}_{i,j}^n\right] / 2$, compute for each particle k an intermediate velocity

$$\mathbf{v}_k^{n+1/2} = \frac{1}{(\Delta l)^2} \sum_{(ij)} a_{(ij)} \mathbf{v}_{(ij)}^{n+1/2}$$

using suitable weights $a_{(ij)}$ (see text); calculate new particle positions

$$\mathbf{r}_k^{n+1} = \mathbf{r}_k^n + \Delta t\, \mathbf{v}_k^{n+1/2}$$

4. Each particle is now given the new state vector

$$\mathbf{u}_k^{n+1} \equiv \left\{\mathbf{r}_k^{n+1}, \mathbf{v}_k^{n+1}, c_k^{n+1}\right\}$$

Comparing Representations

- How do we render the resulting surface?
- Are we guaranteed not to lose mass/volume? (is the simulation incompressible?)
- How is each affected by the grid resolution and timestep?
- Can we guarantee stability?

Demos

- Nice Marker and Cell (MAC) videos at:

`http://panoramix.ift.uni.wroc.pl/~maq/eng/cfdthesis.php`

Questions?

Today

- Flow Simulations in Computer Graphics
- Navier-Stokes Equations
- Fluid Representations
- Basic Algorithm
- Data Representation

Each Grid Cell Stores:

- Velocity at the cell faces (offset grid)
- Pressure
- List of particles

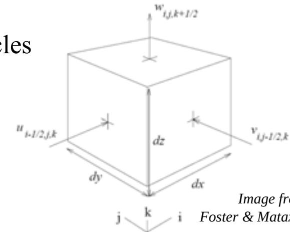


Image from Foster & Mataxas, 1996

Initialization

- Choose a voxel resolution
- Choose a particle density
- Place the particles
- Initialize pressure & velocity of each cell
- Set the viscosity & gravity
- Choose a timestep & go!

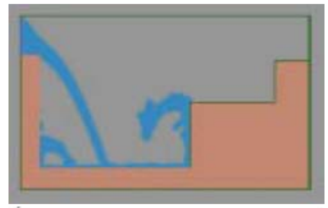
At each Timestep:

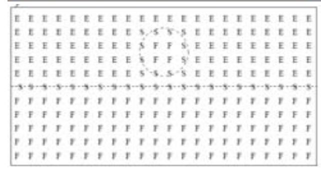
- Identify which cells are Empty, Full, or on the Surface
- Compute new velocities
- Adjust the velocities to maintain an incompressible flow
- Move the particles
  - Interpolate the velocities at the faces
- Render the geometry and repeat!

At each Timestep:

- Identify which cells are Empty, Full, or on the Surface
- Compute new velocities
- Adjust the velocities to maintain an incompressible flow
- Move the particles
  - Interpolate the velocities at the faces
- Render the geometry and repeat!

Empty, Surface & Full Cells





Images from Foster & Mataxas, 1996

At each Timestep:

- Identify which cells are Empty, Full, or on the Surface
- Compute new velocities
- Adjust the velocities to maintain an incompressible flow
- Move the particles
  - Interpolate the velocities at the faces
- Render the geometry and repeat!

Compute New Velocities

$$\begin{aligned}\tilde{u}_{i+1/2,j,k} = u_{i+1/2,j,k} + \delta t\{&(1/\delta x)[(u_{i,j,k})^2 - (u_{i+1,j,k})^2] \\ &+(1/\delta y)[(uv)_{i+1/2,j-1/2,k} - (uv)_{i+1/2,j+1/2,k}] \\ &+(1/\delta z)[(uw)_{i+1/2,j,k-1/2} - (uw)_{i+1/2,j,k+1/2}] + g_x \\ &+(1/\delta x)(p_{i,j,k} - p_{i+1,j,k}) + (\nu/\delta x^2)(u_{i+3/2,j,k} \\ &-2u_{i+1/2,j,k} + u_{i-1/2,j,k}) + (\nu/\delta y^2)(u_{i+1/2,j+1,k} \\ &-2u_{i+1/2,j,k} + u_{i+1/2,j-1,k}) + (\nu/\delta z^2)(u_{i+1/2,j,k+1} \\ &-2u_{i+1/2,j,k} + u_{i+1/2,j,k-1})\},\end{aligned}$$

Note: some of these values are the *average velocity* within the cell rather than the velocity at a cell face

At each Timestep:

- Identify which cells are Empty, Full, or on the Surface
- Compute new velocities
- Adjust the velocities to maintain an incompressible flow
- Move the particles
  - Interpolate the velocities at the faces
- Render the geometry and repeat!

Adjusting the Velocities

- Calculate the *divergence* of the cell (the extra in/out flow)
- The divergence is used to update the *pressure* within the cell
- Adjust each face velocity uniformly to bring the divergence to zero
- Iterate across the entire grid until divergence is < ε

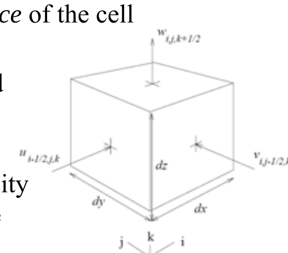
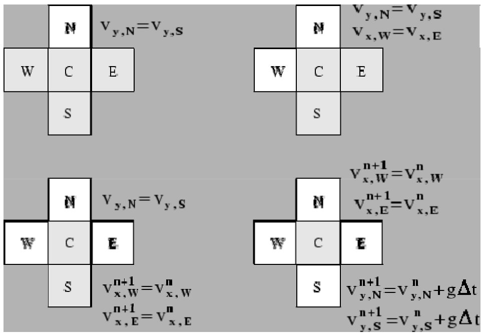


Image from Foster & Mataxas, 1996

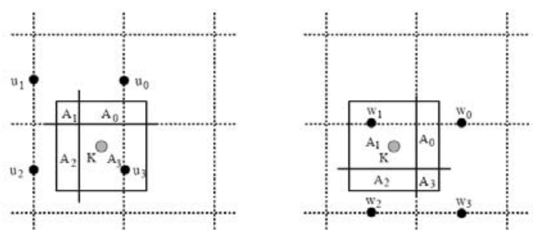
Handing Boundaries with MAC



At each Timestep:

- Identify which cells are Empty, Full, or on the Surface
- Compute new velocities
- Adjust the velocities to maintain an incompressible flow
- Move the particles
  - Interpolate the velocities at the faces
- Render the geometry and repeat!

Velocity Interpolation

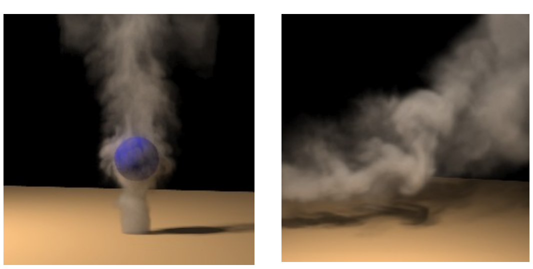


$u_k = A_0 u_0 + A_1 u_1 + A_2 u_2 + A_3 u_3$ $w_k = A_0 w_0 + A_1 w_1 + A_2 w_2 + A_3 u_3$

Image from Foster & Mataxas, 1996

Questions?

Efficient Smoke Simulation



"Visual Simulation of Smoke"
Fedkiw, Stam & Jensen
SIGGRAPH 2001

Solid/Liquid: Time-varying viscosity

"Melting and Flowing"
Carlson, Mucha, Van Horn III & Turk
Symposium on Computer Animation 2002