Realistic Animation of Fluids

Nick Foster and Dimitris Metaxas

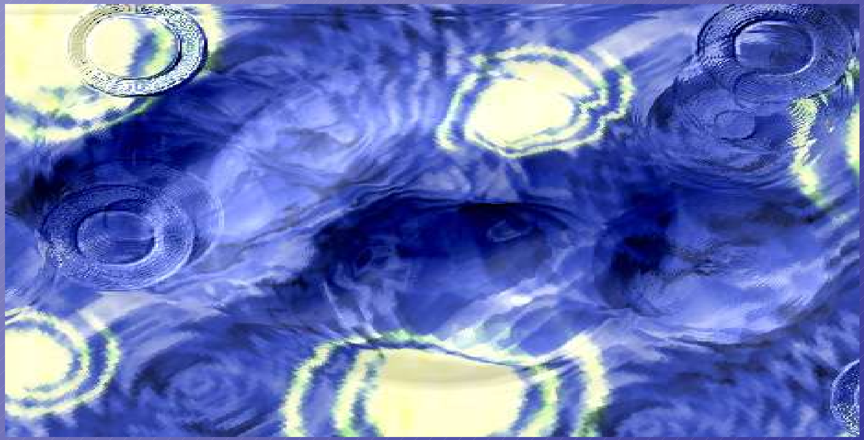
Presented by Alex Liberman
CS 667
April 19, 2005

Previous Work

• Used non physics-based methods (mostly in 2D)

• Hard to simulate effects that rely on physical phenomena and volume; rotational and pressure-based effects

• Hard to make fluid interact with dynamic objects and to simulate the buoyancy force on such objects



Quintessential Player

Inspiration from CFD

- Computational Fluid Dynamics has been used in engineering problems since the mid '60s
- Very detailed fluid models exist
- Unfortunately most methods are needlessly-complex (for graphics) and scale poorly
- End-user must know what he/she is doing (more than the average animator is interested in)

Main contributions from this paper

- Simulation – Solving the Navier-Stokes equations
- Boundary Conditions – Initializing fluid boundaries and setting up global behavior
- Surface Tracking – Tracking the evolution of the fluid surface for rendering

Simulation: Navier-Stokes (N-S)

- N-S are a set of equations that describe the behavior of various fluids (including gases)
- For graphics the "incompressible Navier-Stokes equations" are sufficient

Next Level Inc.

http://flatrock.org.nz/topics/environment/assets/nuclear_bomb_test.jpg

Simulation: Equations

Foster & Metaxas formulation:

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

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

Foster & Metaxas '96

A vector-based formulation:

$$\mathbf{u}_t = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \nu\nabla^2\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}$$

$$\nabla\cdot\mathbf{u} = 0$$

www.cc.gatech.edu/~carlson/papers/carlson-thesis.pdf

u – fluid velocity (vector field) (*u*, *v*, *w*)

u_t – fluid acceleration (vector field) (∂*u*/∂t, ∂*v*/∂t, ∂*w*/∂t)

p – fluid pressure (scalar field)

ρ – fluid density (~1.0 for water – assumed in F&M)

v – viscosity or “fluid thickness” (assumed constant)

Simulation: More N-S detail

www.cc.gatech.edu/~carlson/papers/carlson-thesis.pdf

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} + v\nabla^2\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}$$

$$a = \frac{F_{net}}{m}$$

$$\nabla \cdot \mathbf{u} = 0$$

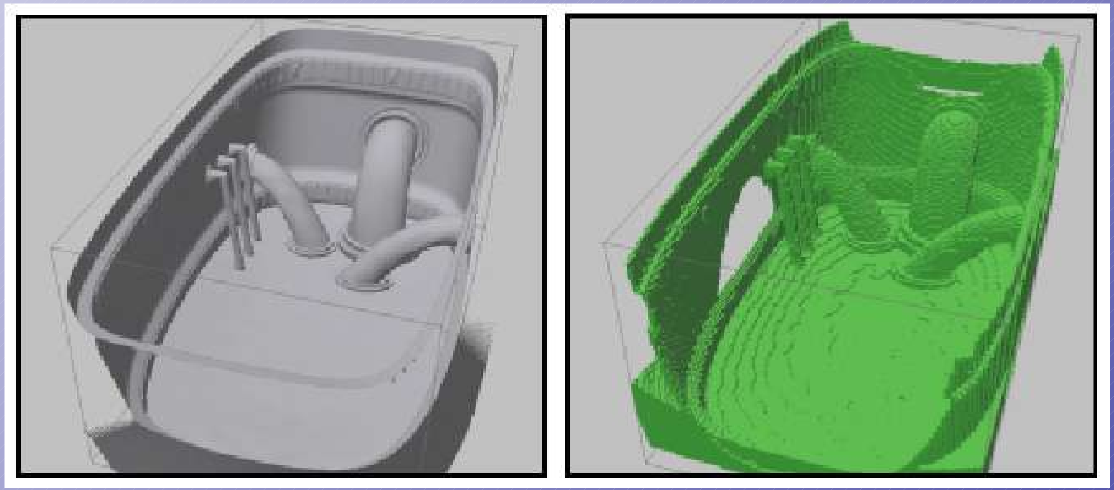
Mass Conservation:
net fluid flow is 0

$(u \cdot \nabla) u$ Advection: Describes in what direction a “neighboring” region of water pushes water at **u**

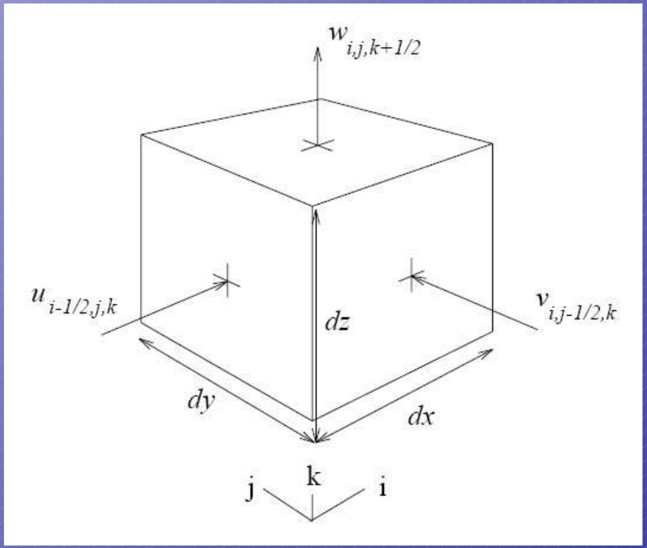
$(v \nabla^2 u)$ Momentum Diffusion: Describes how quickly variations in velocity are damped-out; depends on fluid viscosity

$(\frac{1}{\rho} \nabla p)$ Pressure Gradient: Describes in which direction fluid at **u** is pushed to reach a lower pressure area

Simulation: Discretization



Foster & Metaxas '00



Foster & Metaxas '96

Simulation: Velocity field update

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

Intimidating, but really means...

$$1 > max[u\frac{\delta t}{\delta x}, v\frac{\delta t}{\delta y}, w\frac{\delta t}{\delta z}]$$

Foster & Metaxas '96

CFL Condition: traverse at most one cell in a single time-step

Simulation: Velocity field update (ctd)

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

$$1 > max[u\frac{\delta t}{\delta x}, v\frac{\delta t}{\delta y}, w\frac{\delta t}{\delta z}]$$

Foster & Metaxas '96

CFL Condition: traverse at most one cell in a single time-step

Intimidating, but really means...

$$u(t+1) = u(t) + h(u_t(t))$$

$$\tilde{u} = u + \cdots$$

- Is an Euler step that uses a finite-difference approximation of $u_t(t)$
- Maximum value of h is dictated by the CFL

Simulation: Finite differences

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

Foster & Metaxas '96

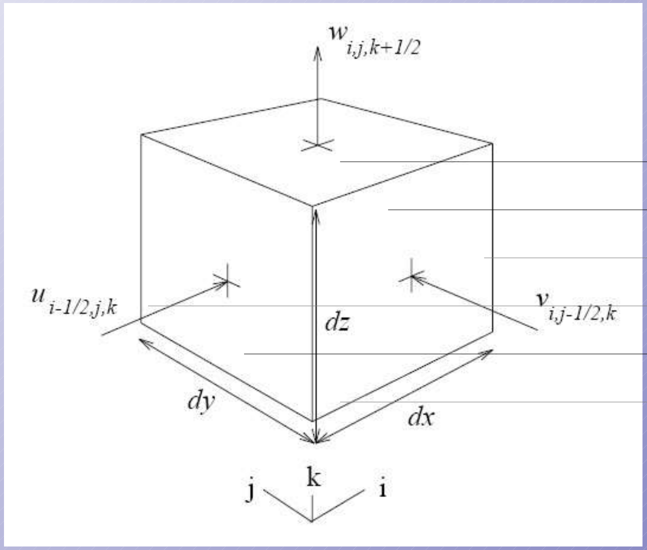
$$(\nabla \cdot \mathbf{u})_{i,j,k} = \frac{u_{i+1,j,k} - u_{i,j,k} + v_{i,j+1,k} - v_{i,j,k} + w_{i,j,k+1} - w_{i,j,k}}{\Delta x}$$

$$(\nabla p)_{i,j,k} = \left(\frac{p_{i,j,k} - p_{i-1,j,k}}{\Delta x}, \frac{p_{i,j,k} - p_{i,j-1,k}}{\Delta x}, \frac{p_{i,j,k} - p_{i,j,k-1}}{\Delta x}\right)$$

$$\nabla^2 u_{i,j,k} = \frac{u_{i-1,j,k} + u_{i,j-1,k} + u_{i,j,k-1} - 6u_{i,j,k} + u_{i+1,j,k} + u_{i,j+1,k} + u_{i,j,k+1}}{\Delta x^2}$$

www.cc.gatech.edu/~carlson/papers/carlson-thesis.pdf

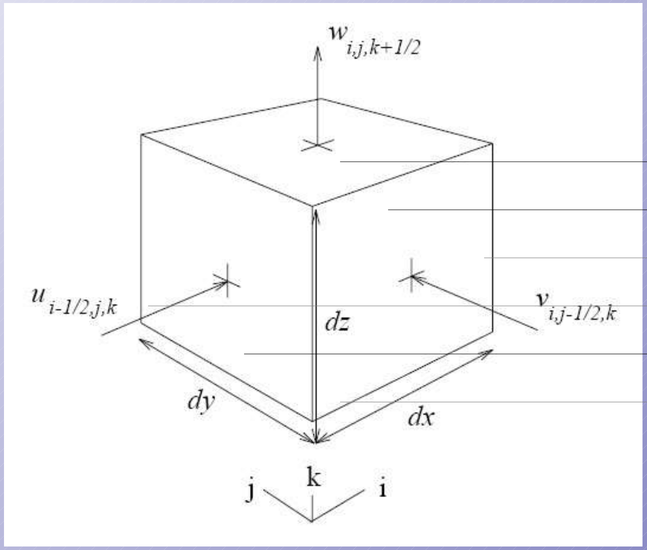
Simulation: Problem



Foster & Metaxas '96

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

Simulation: Problem – compressible fluids?

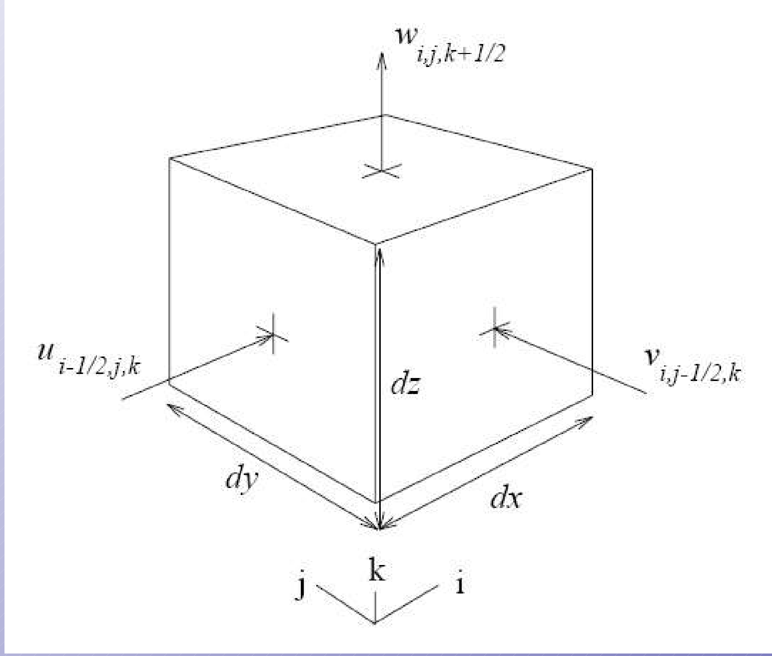


Foster & Metaxas '96

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

However: $\nabla \cdot u \neq 0$

Simulation: Enforcing incompressibility



Foster & Metaxas '96

- Compute divergence

$$D = \nabla \cdot u_{i,j,k}$$

- Computer pressure correction

$$\delta p_{i,j,k} = \beta D_{i,j,k}$$

- Update cell-face velocities

$$u_{i+1/2,j,k} = u_{i+1/2,j,k} + (\delta t / \delta x) \delta p_{i,j,k}$$

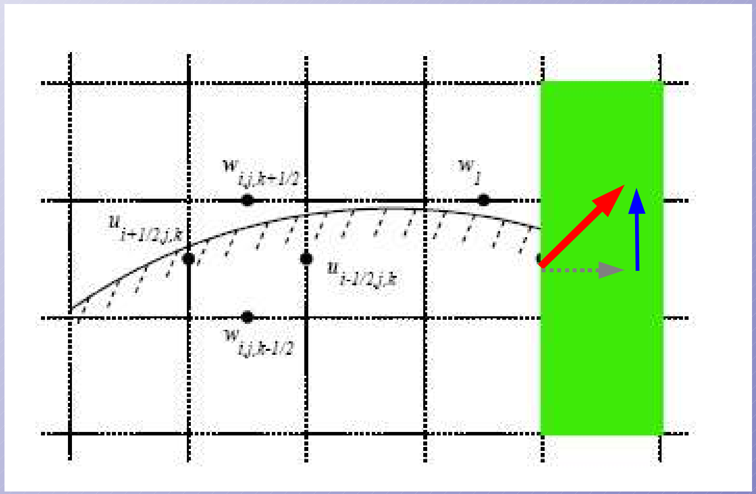
$$u_{i-1/2,j,k} = u_{i-1/2,j,k} - (\delta t / \delta x) \delta p_{i,j,k}$$

...

- Iterate the above steps until $D \approx 0$
- Update pressure field: $\tilde{p}_{i,j,k} = p_{i,j,k} + \delta p_{i,j,k}$

Boundary Conditions

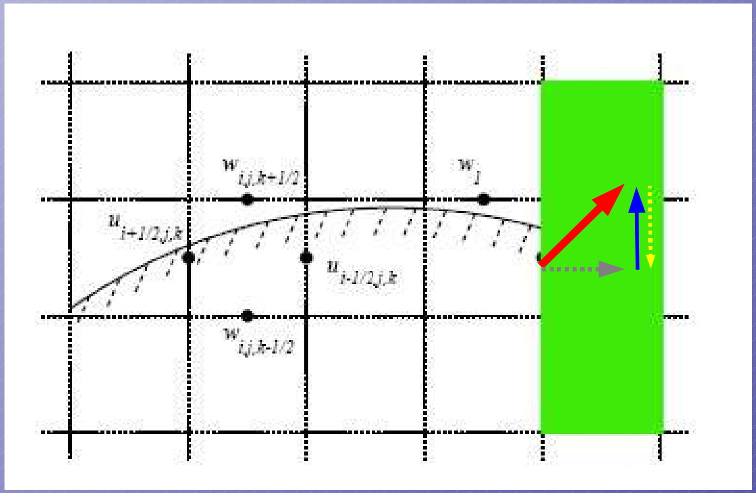
free-slip boundary



Foster & Metaxas '96

- Fluid is free to move parallel to the boundary

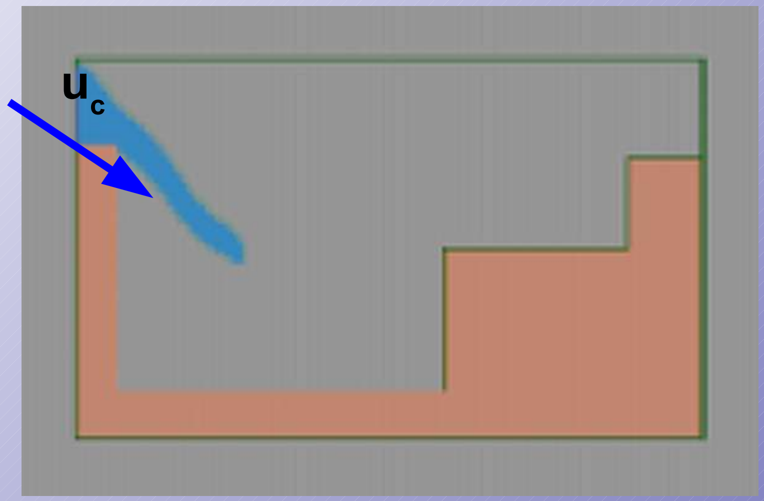
non-slip boundary



- Fluid is fixed to boundary

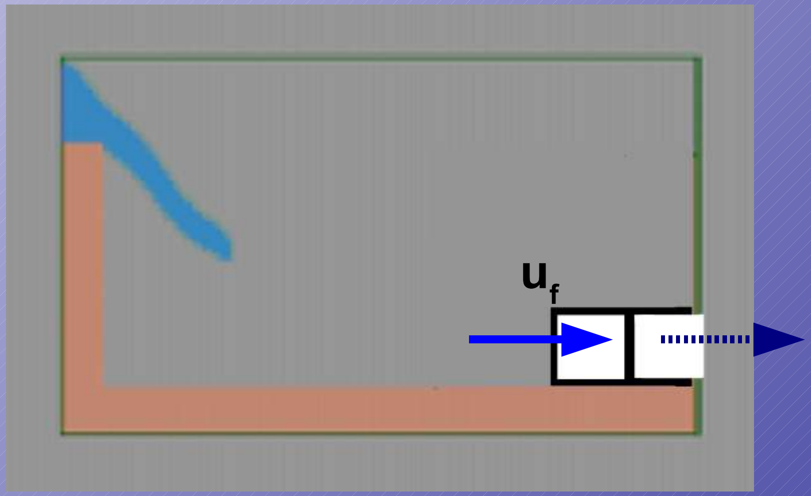
Inflow/Outflow conditions (sources and sinks)

inflow condition



Foster & Metaxas '96

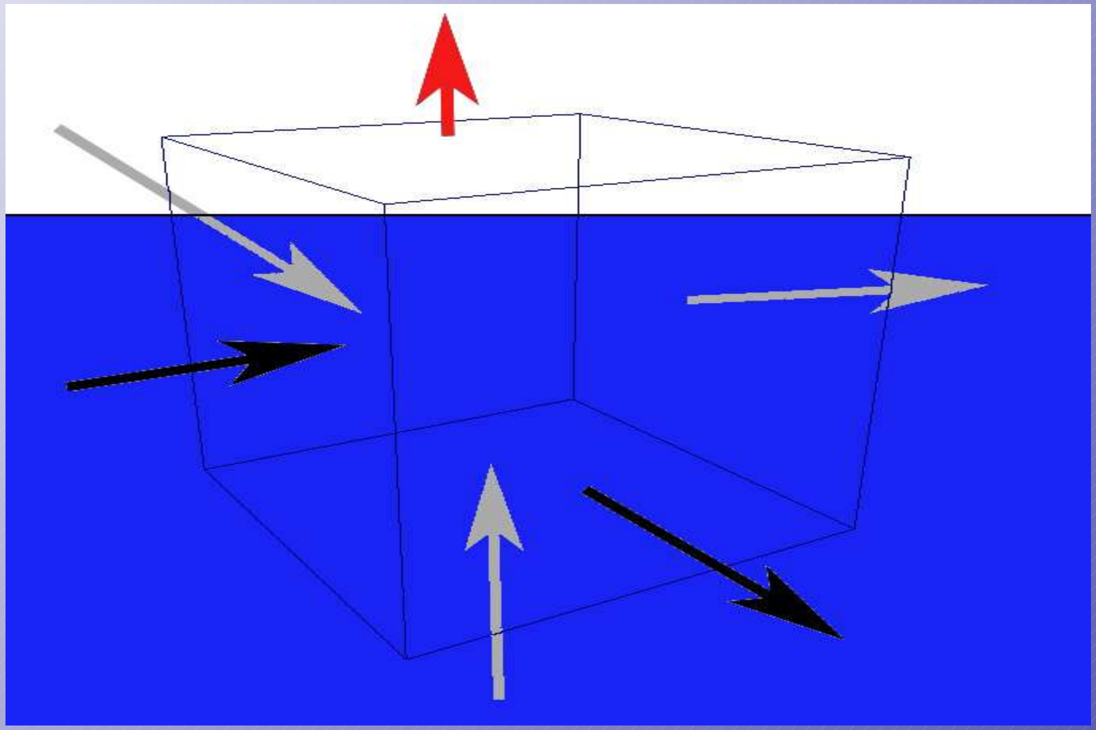
outflow condition



- Set a constant fluid velocity $\mathbf{u}_c$ on the face of a source cell

- Set sink cell's face velocity to adjacent fluid cell's velocity, and allow it to “relax”

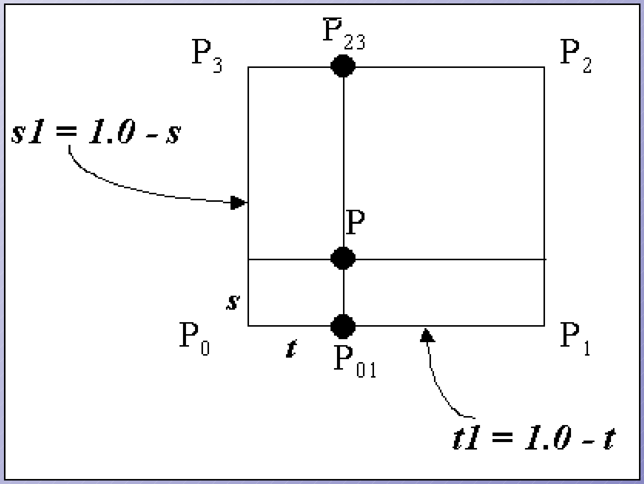
Boundary condition for free surface cells



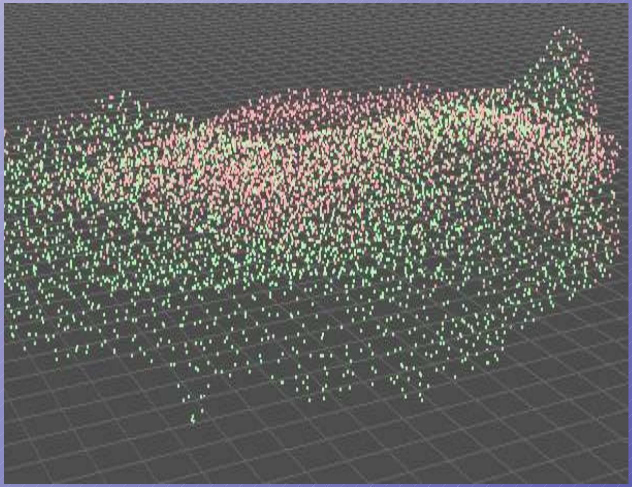
- Surface cell's interface's velocity = sum of the remaining velocities
- This maintains the condition

$$\nabla \cdot u \approx 0$$

Surface Tracking: Marker Particles



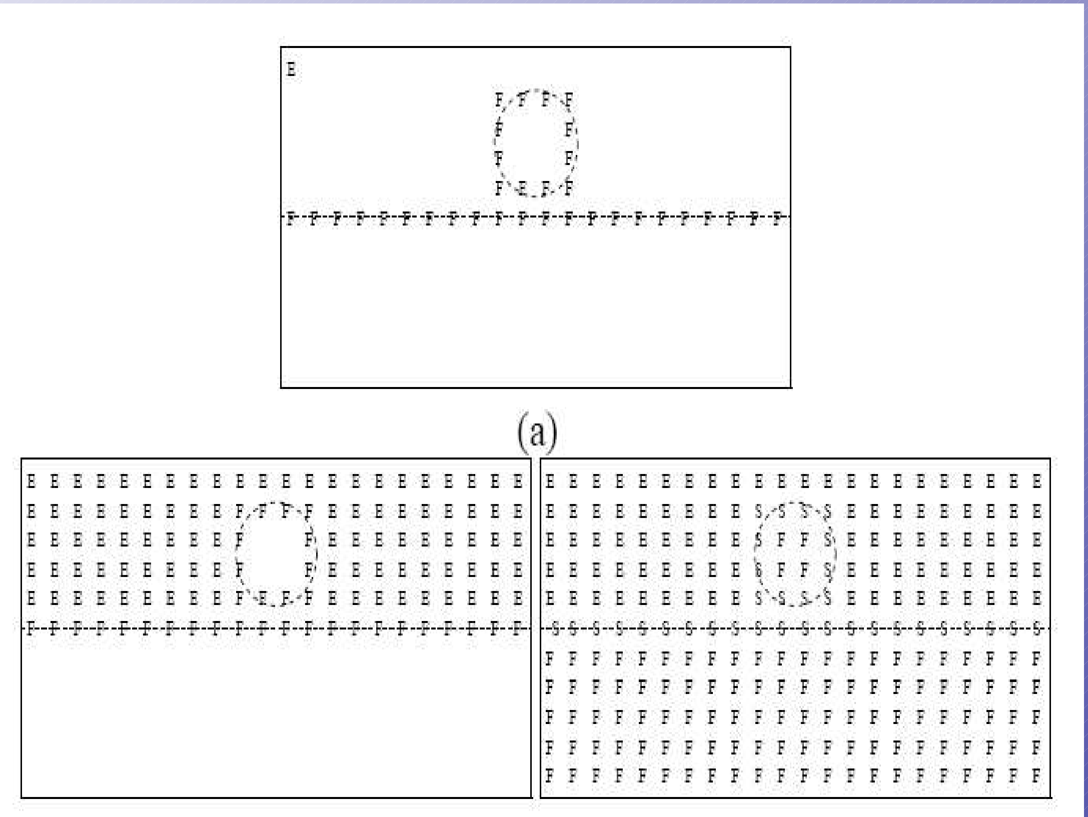
http://developer.intel.com/technology/itj/q21999/images/art5fig7.gif



McFluid

- Generate marker particles at sources (and render them only in cells which contain fluid)
- Use weighted (bilinear) interpolation to compute a marker's velocity from four nearest cell velocities
- Propagate particles using standard integration techniques
- Easy to simulate violent effects like splashing and overturning

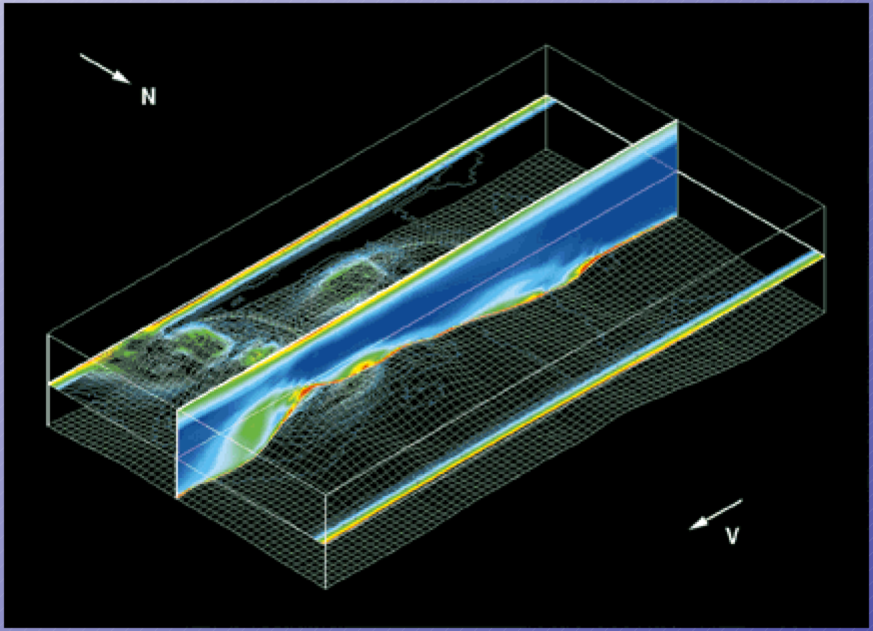
Surface Tracking: Free Surface Particles



Foster & Metaxas '96

- Start at known "Full" or "Empty" regions and grow particles via 4-connected search
- Insert particles when too sparse, remove when too dense
- Problematic: how to efficiently detect sparse/dense areas?

Surface Tracking: Height Field



http://www.sdsc.edu/GatherScatter/gsjan94/Images/GS.CFD1.gif

For smooth fluids (ocean, lake, puddles) w/o splashes

$$H_{t+1} = H_t + \delta t \left(w_{t+\delta t} + \delta H \cdot un_{t+1} \right)$$

un_{t+1} – velocity field values of neighbor cells

Miscellaneous & Questions?

- Better Explanation and Implementation Guide – Available as part of Mark Carlson's PhD thesis: www.cc.gatech.edu/~carlson/papers/carlson-thesis.pdf
- Other Simulation Techniques – SPH; simulate motion of marker particles directly
- CFD Math & Derivations – Pain available at: www.math.colostate.edu/~pauld/M646.html