

# Stable Fluids

Jos Stam

Alias|wavefront

Seattle, WA USA

# Fluids in Computer Graphics

- Interactive modeling of fluids
- Fast
- Capture visuals of fluids

# Fluid Mechanics

- Natural framework for fluid modeling
  - Full Navier-Stokes Equations
- Has a long history
  - reuse code/algorithms
- Equations are hard to solve
  - non-linear

# Previous Work (computer graphics)

## Two dimensions:

- Yaeger & Upson 86 + Gamito et al. 95 (vortex blobs)
- Chen et al. 97 (explicit in time, finite differences)

## Three-dimensions:

- Foster & Metaxas 97 (explicit in time, finite differences)

**unstable**

Inaccurate schemes can be useful

# Main Contribution

Stable Navier-Stokes solver

Any time step can be used

Bigger time steps = faster simulations

NOT accurate

# Application

Use velocity to move densities:

```
While ( simulating )  
    Get force from UI  
    Get density from UI  
    Update velocity  
    Update density  
    Display density
```

# Equations

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla) \rho + \kappa \nabla^2 \rho + S$$

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

+ velocity should conserve mass

Equations very similar

# Equations

Evolution of density (assume velocity known)

$$\boxed{\frac{\partial \rho}{\partial t}} = -(\mathbf{u} \cdot \nabla) \rho + \kappa \nabla^2 \rho + S$$

Over a time step...

# Equations

Evolution of density (assume velocity known)

$$\frac{\partial \rho}{\partial t} = \boxed{-(\mathbf{u} \cdot \nabla) \rho} + \kappa \nabla^2 \rho + S$$

Density changes in the direction of the flow

# Equations

Evolution of density (assume velocity known)

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla) \rho + \boxed{\kappa \nabla^2 \rho} + S$$

Density diffuses over time

# Equations

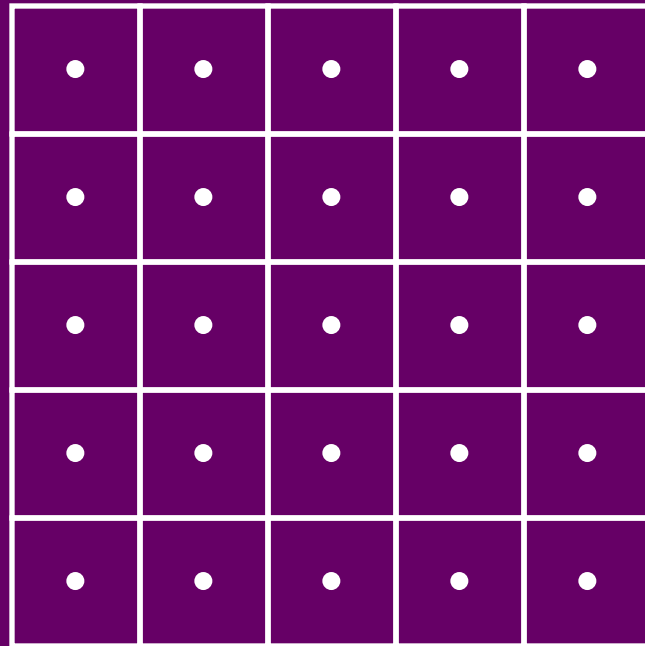
Evolution of density (assume velocity known)

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla) \rho + \kappa \nabla^2 \rho + \boxed{S}$$

Increases due to sources from the UI

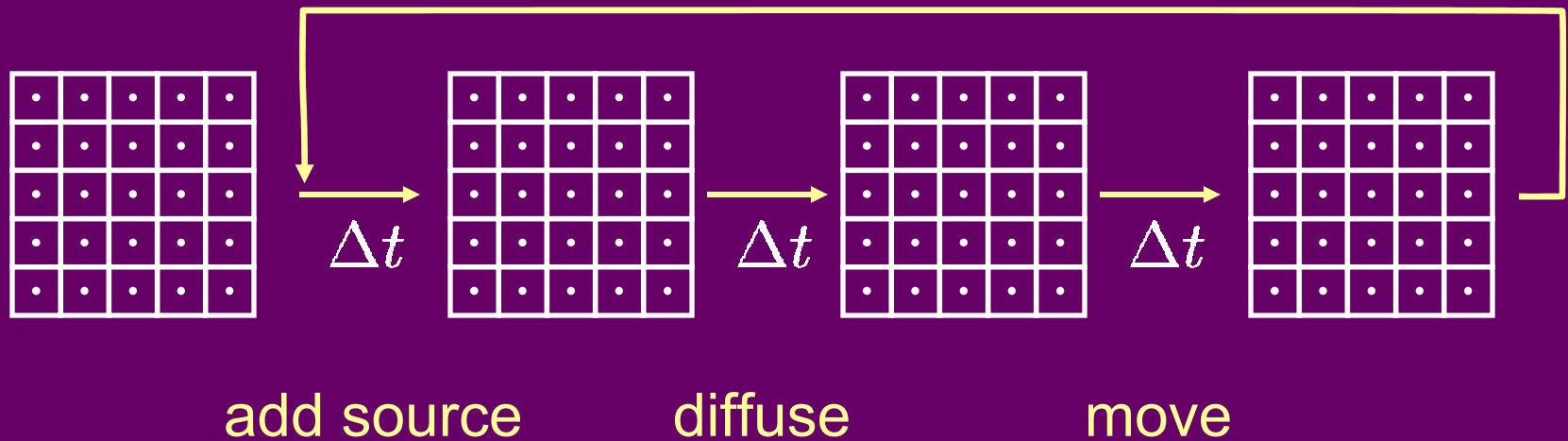
# Algorithm

Subdivide space into voxels



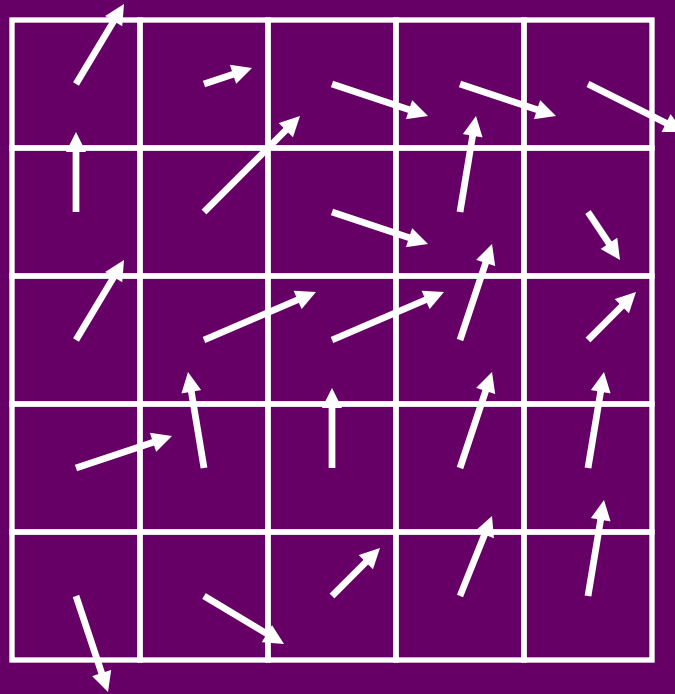
Velocity + density defined in the center of each voxel

# Algorithm



$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla) \rho + \kappa \nabla^2 \rho + S$$

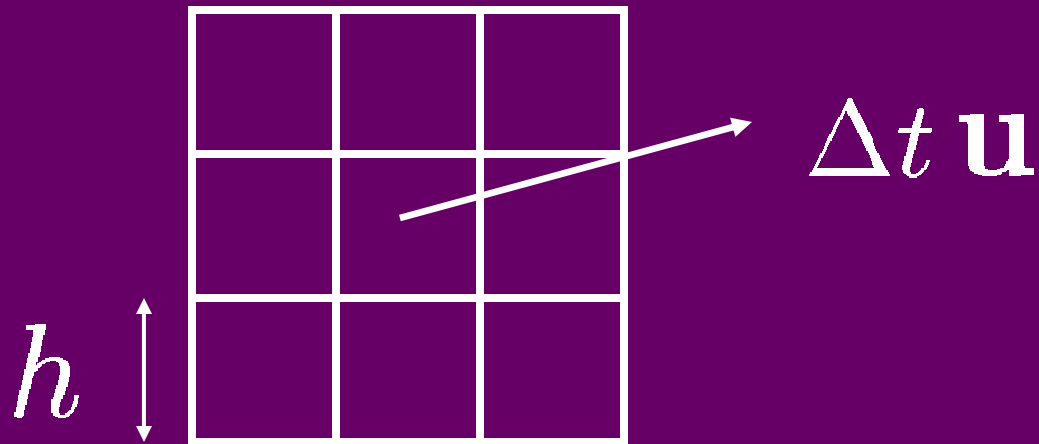
# Moving Densities



Velocity known

# Moving Densities

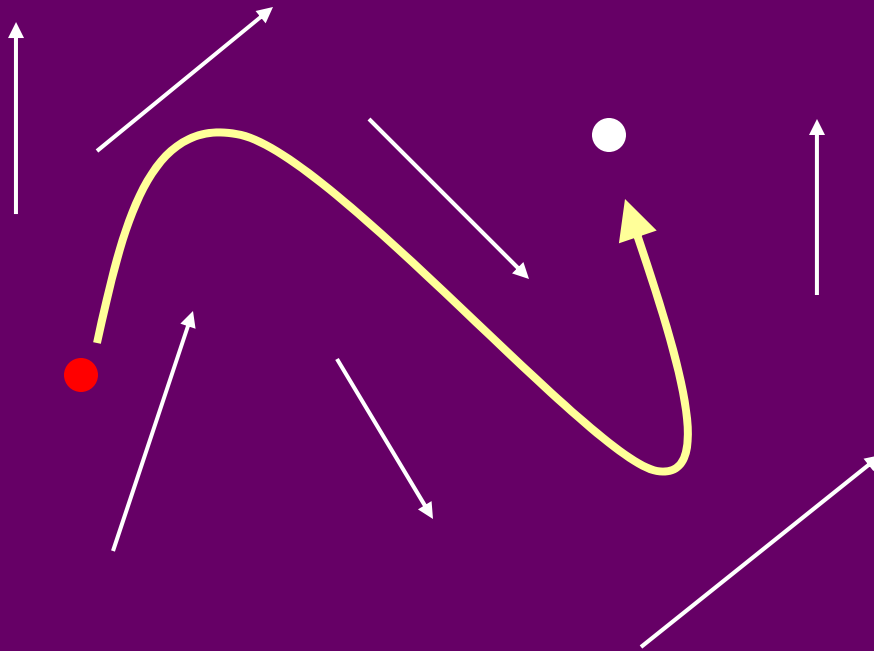
Finite Differences: transfer only between neighbors



Unstable when  $\Delta t |\mathbf{u}| > h$

# Moving Densities

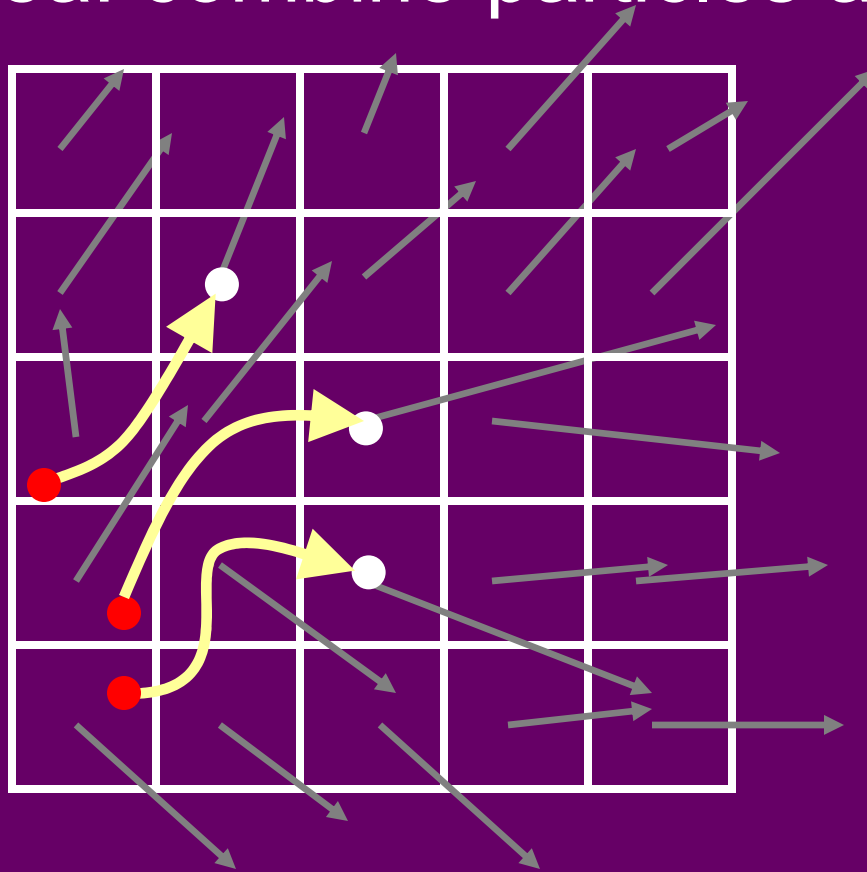
Easy if density defined on particles



Any time step ok

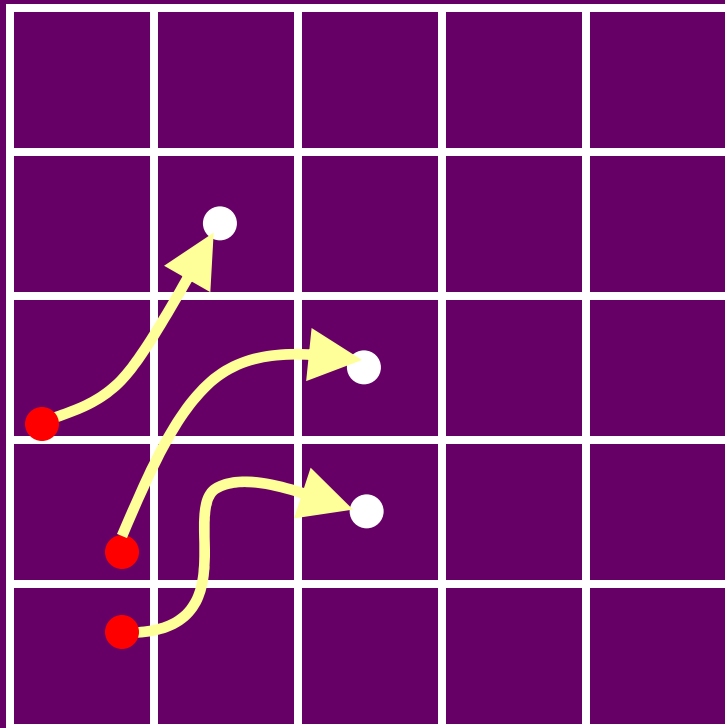
# Moving Densities

Key Idea: combine particles and grids



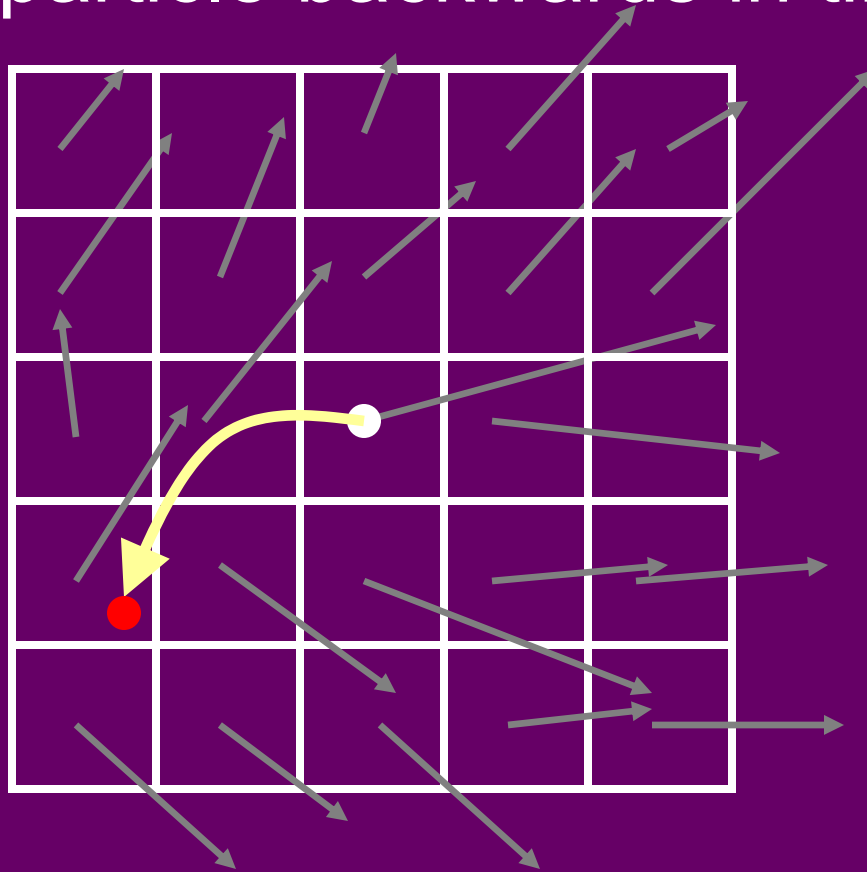
# Moving Densities

Key Idea: combine particles and grids



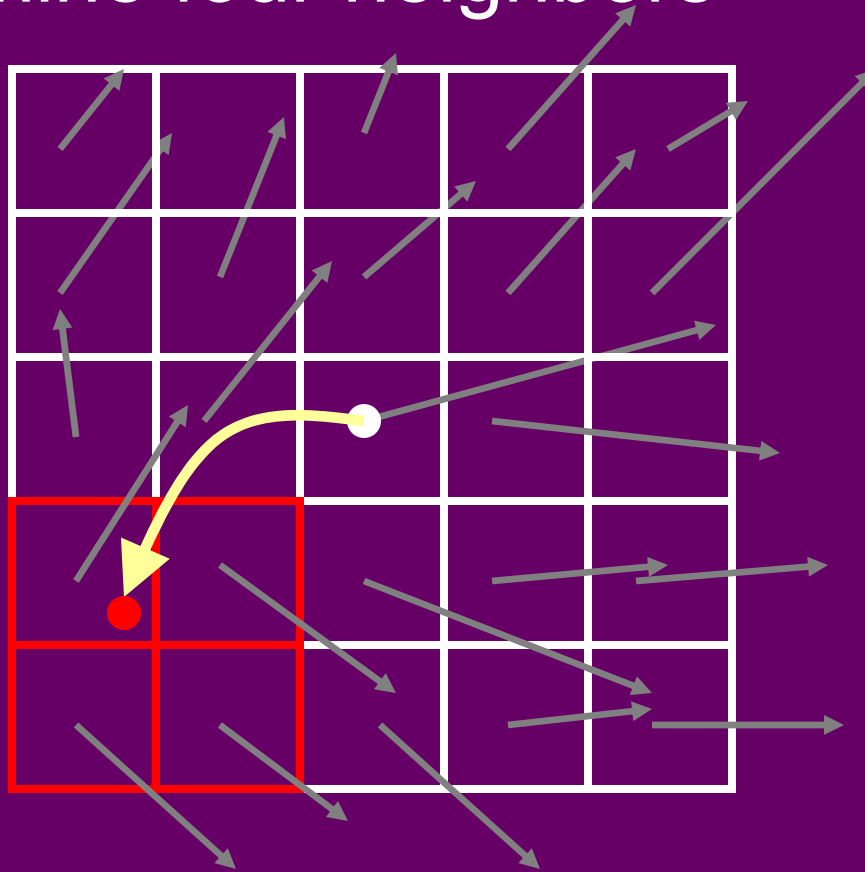
# Moving Densities

Trace particle backwards in time



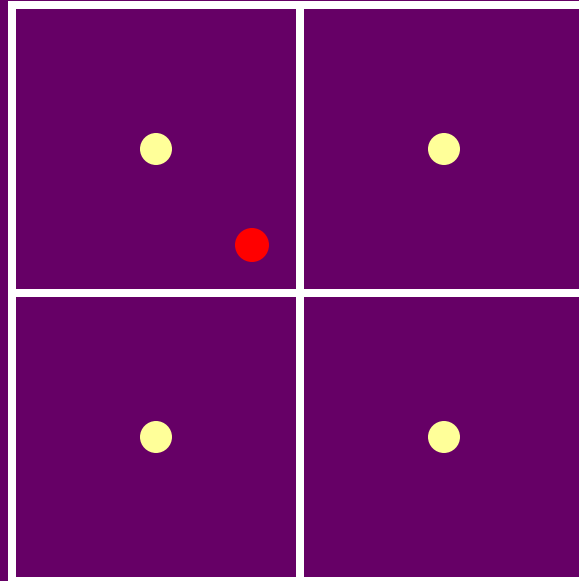
# Moving Densities

Determine four neighbors



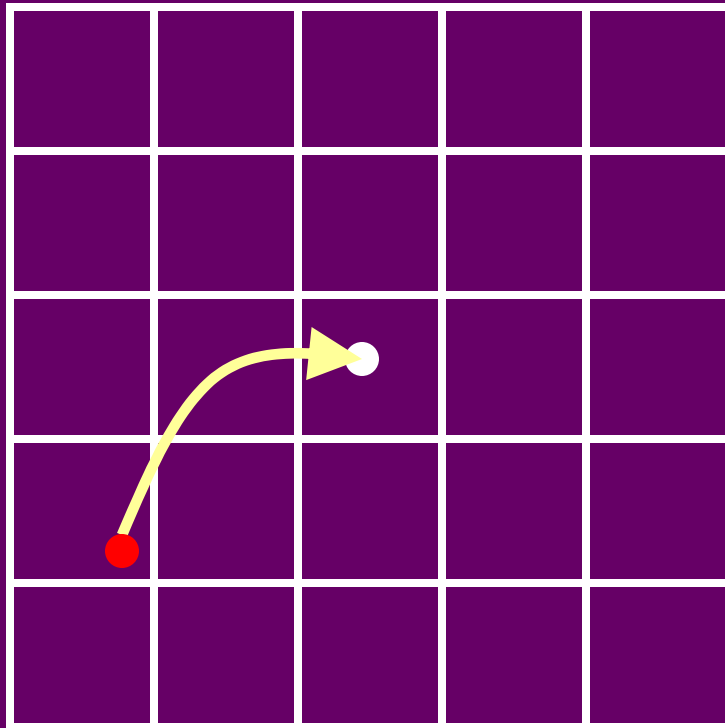
# Fluid Mechanics

Interpolate the density at new location



# Moving Densities

Set interpolated density at grid location



Requires two grids

# Moving Densities

This scheme is unconditionally stable:

$$\rho_{int} = (1 - s)\rho_0 + s \rho_1$$

$$\rho_0, \rho_1 \leq \rho_{max}$$

$$\rho_{int} \leq (1 - s + s)\rho_{max} \leq \rho_{max}$$

→ density is always bounded

# Computing Velocities

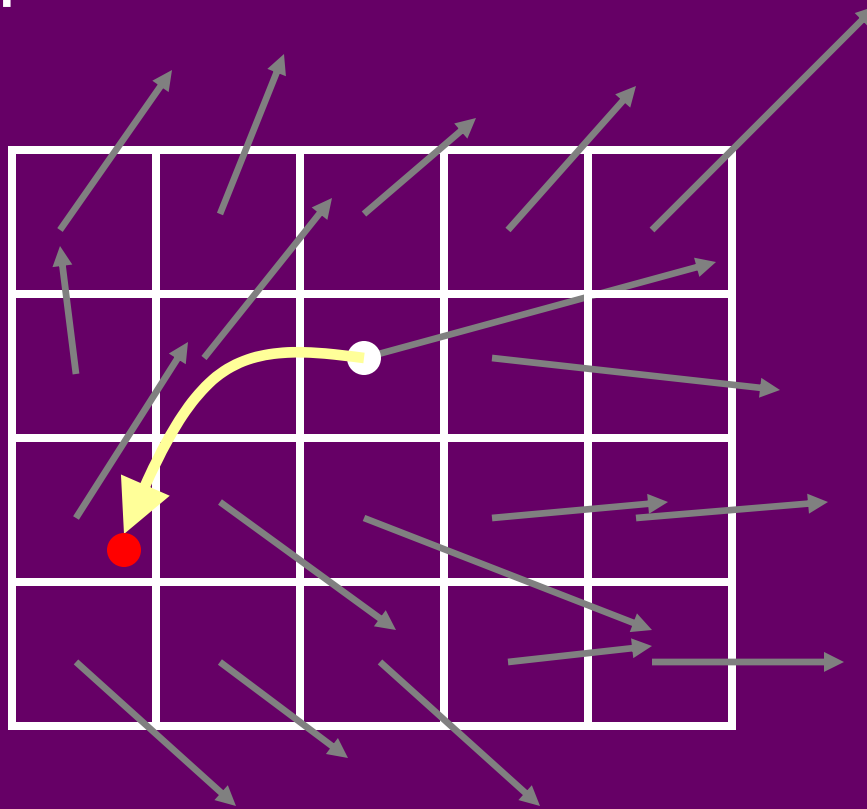
Use same algorithms as for density

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla) \rho + \kappa \nabla^2 \rho + S$$
$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

Velocity is moved by itself

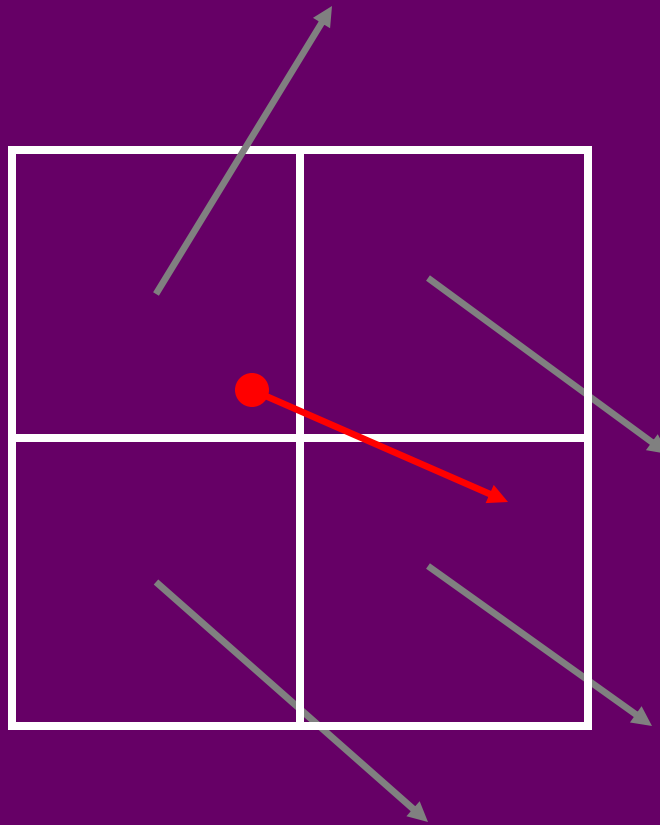
# Moving Velocity

Trace particle backwards in time



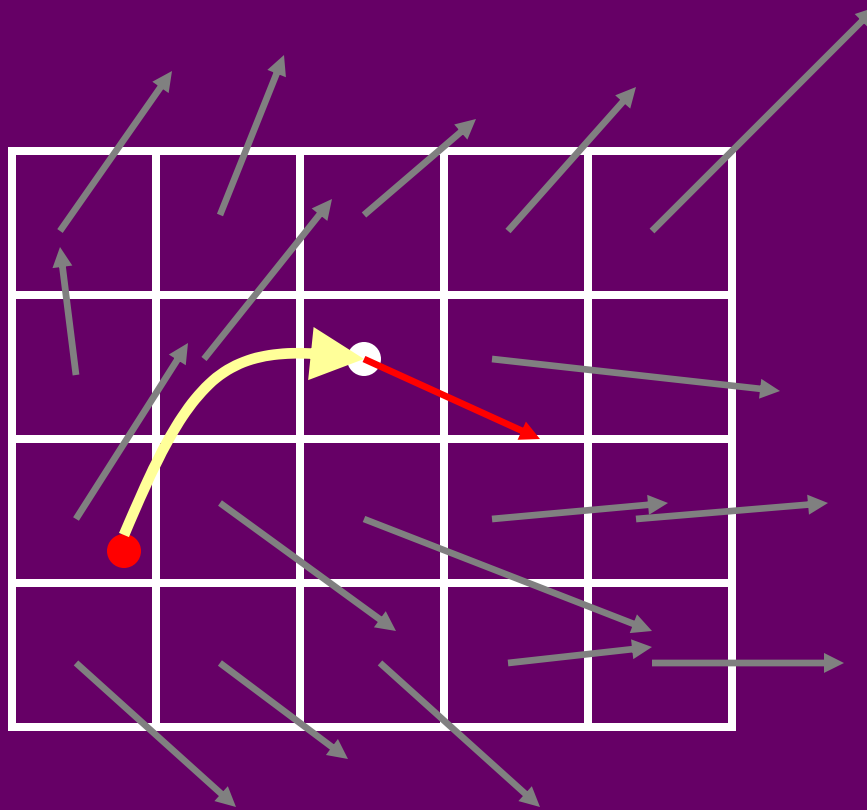
# Moving Velocity

Interpolate the velocity at new location



# Moving Velocity

Set interpolated velocity at grid location



Requires two grids

# Summary

`UpdateVelocity (U1, U0, F, visc, dt)`

`AddForce (U1, U0, F, dt)`

`Diffuse (U0, U1, visc, dt)`

`Move (U1, U0, U0, dt)`

`ConserveMass (U1, dt)`

Very easy to code. Only need:

Particle tracer + grid interpolator

PDE solver (FISHPAK)

# Results

Demo time....

Hardware:

sgi Octane Workstation

R12K 300Mhz CPU

3D texture hardware

# Future Work

- Handle free boundaries (water)
- Parallel implementation
- Other texture maps
- Handle complex boundaries
- Can use same algorithm for boundary fitted finite element meshes