



# **CS 380 - GPU and GPGPU Programming**

## **Lecture 24: GPU Fluid Simulation and Rendering**

Markus Hadwiger, KAUST

# Reading Assignment++



Read (**optional** :):

- **Larrabee: A Many-Core x86 Architecture for Visual Computing**

*Larry Seiler et al.*, Siggraph 2008

<http://software.intel.com/file/18198/>

**Larrabee Architecture @ Intel**

<http://software.intel.com/en-us/articles/larrabee/>

**Rasterization on Larrabee**

<http://software.intel.com/en-us/articles/rasterization-on-larrabee/>

- **OpenCL 1.0 Specification**

Khronos Group

<http://www.khronos.org/opencv/>

# Fluid Simulation in Computer Graphics

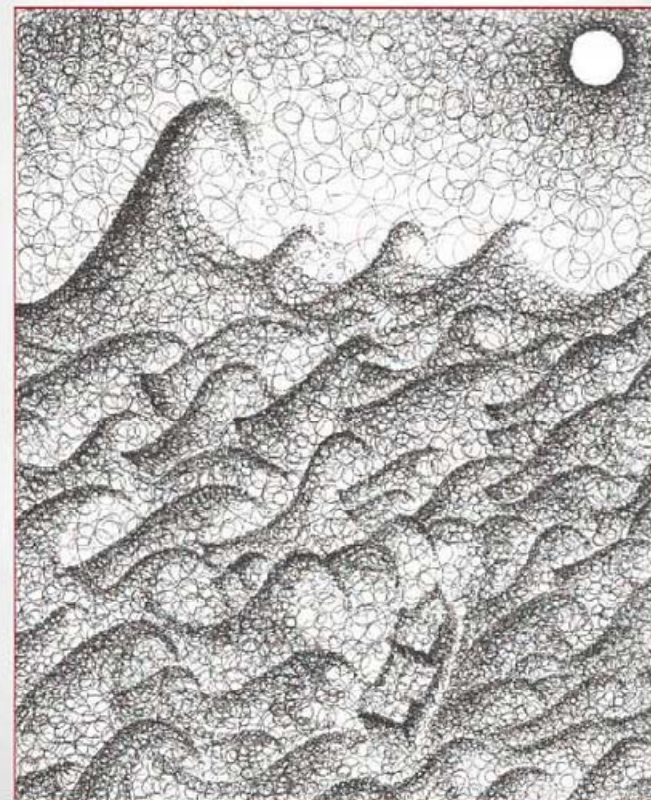


## Goal

- Visually appealing and convincing results
  - Physically based (Navier-Stokes)
  - But not necessarily physically accurate
- Effects for movies and games
- Lots of publications in computer graphics community (SIGGRAPH, ...)
- Very good overview:  
Robert Bridson, *Fluid Simulation for Computer Graphics*, AK Peters 2008

## Fluid Simulation for Computer Graphics

Robert Bridson



# Fluid Simulation and Rendering



Compute advection of fluid

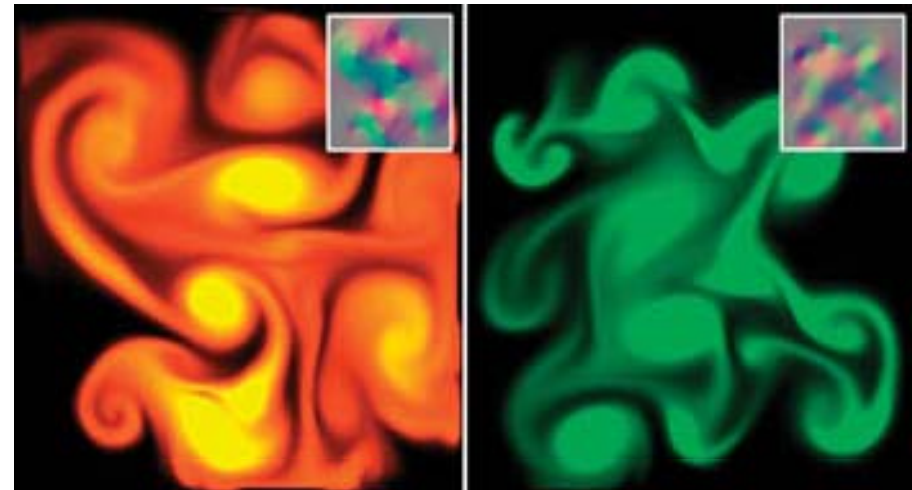
- (Incompressible) Navier-Stokes solvers
- Lattice Boltzmann Method (LBM)

Discretized domain; stored in 2D/3D textures

- Velocity, pressure
- Dye, smoke density, vorticity, ...

Updates in multiple passes

Render current frame



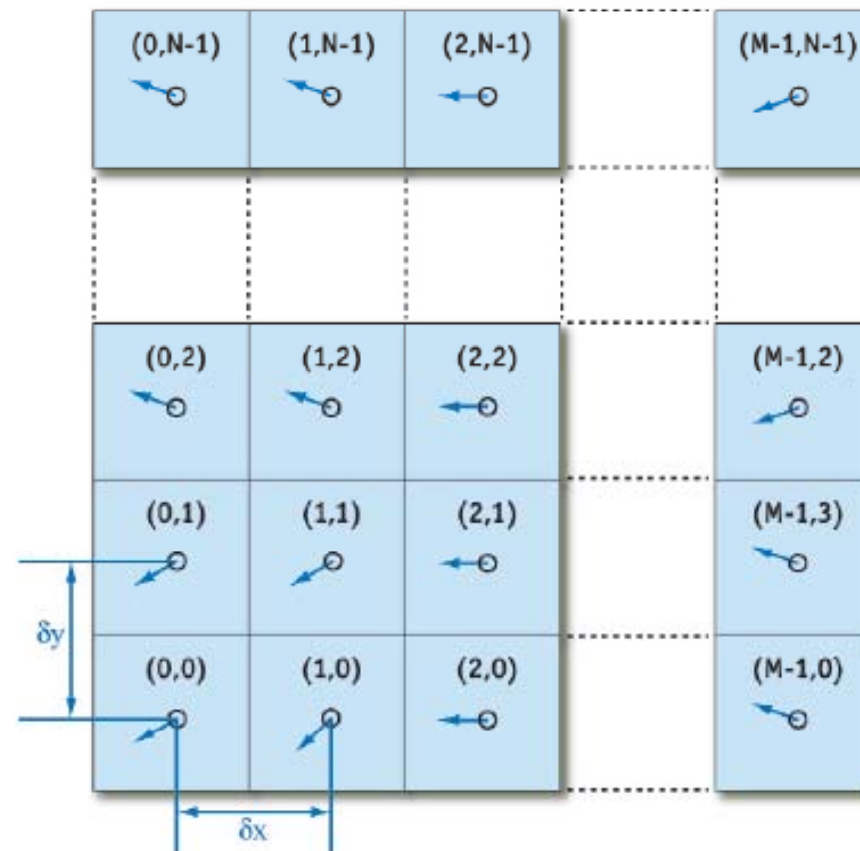
Courtesy Mark Harris

# Velocity Field



2D or 3D vector field

- Stored in 2D or 3D texture/array



# Fluid Simulation: Navier Stokes (1)



Incompressible Navier Stokes (divergence-free)

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F},$$

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

Components:

- Self-advection of velocity (i.e., advection of velocity according to velocity)
- Pressure gradient (force due to pressure differences)
- Diffusion of velocity due to viscosity (for viscous fluids, i.e., not inviscid)
- Application of (arbitrary) external forces, e.g., gravity, user input, etc.

# Fluid Simulation: Navier Stokes (2)



Actually, the momentum equation is a system of equations  
(2 equations in 2D, 3 equations in 3D)

$$\frac{\partial u}{\partial t} = -(\mathbf{u} \cdot \nabla) u - \frac{1}{\rho} \nabla p + \nu \nabla^2 u + f_x,$$

$$\frac{\partial v}{\partial t} = -(\mathbf{u} \cdot \nabla) v - \frac{1}{\rho} \nabla p + \nu \nabla^2 v + f_y.$$

# Vector Calculus



## Gradient

- Scalar field  $\rightarrow$  vector field
- Points in direction of highest change

$$\nabla p = \left( \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y} \right)$$

## Divergence

- Vector field  $\rightarrow$  scalar field
- Density exit rate (source?, sink?)

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

## Laplacian

- Scalar field  $\rightarrow$  scalar field
- Divergence of gradient

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$$

# Vector Calculus: Finite Difference Approximations



## Differences between neighboring points

- Result of Taylor expansion
- Discretization leads to diagonal matrix for the whole system of eqs.

Operator	Definition	Finite Difference Form
Gradient	$\nabla p = \left( \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y} \right)$	$\frac{p_{i+1,j} - p_{i-1,j}}{2\delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{2\delta y}$
Divergence	$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$	$\frac{u_{i+1,j} - u_{i-1,j}}{2\delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\delta y}$
Laplacian	$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$	$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\delta y)^2}$

# Why Sparse Linear Systems?



- Discretization using finite differences

$$x_{i,j}^{(k+1)} = \frac{x_{i-1,j}^{(k)} + x_{i+1,j}^{(k)} + x_{i,j-1}^{(k)} + x_{i,j+1}^{(k)} + \alpha b_{i,j}}{\beta},$$

$4\alpha+1$	$-\alpha$		$-\alpha$				
$-\alpha$	$4\alpha+1$	$-\alpha$		$-\alpha$			
	$-\alpha$	$4\alpha+1$			$-\alpha$		
$-\alpha$			$4\alpha+1$	$-\alpha$		$-\alpha$	
	$-\alpha$		$-\alpha$	$4\alpha+1$	$-\alpha$		$-\alpha$
		$-\alpha$		$-\alpha$	$4\alpha+1$		$-\alpha$
			$-\alpha$			$4\alpha+1$	$-\alpha$
				$-\alpha$		$-\alpha$	$4\alpha+1$
					$-\alpha$		$-\alpha$
						$-\alpha$	$4\alpha+1$

$x_1^{t+1}$
$x_2^{t+1}$
$x_3^{t+1}$
$x_4^{t+1}$
$x_5^{t+1}$
$x_6^{t+1}$
$x_7^{t+1}$
$x_8^{t+1}$
$x_9^{t+1}$

$c_1^t$
$c_2^t$
$c_3^t$
$c_4^t$
$c_5^t$
$c_6^t$
$c_7^t$
$c_7^t$
$c_9^t$

Example:  
Navier-Stokes  
pressure-Poisson solve /  
viscous diffusion

Example:  
Solve wave equation using  
Crank-Nicolson scheme  
Courtesy Jens Krüger

# Advection



Advection operator, with velocity field  $\mathbf{u}(t;x,y,z)$

$$\mathbf{u} \cdot \nabla = u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} + w \frac{\partial}{\partial z}$$

- Advect a scalar quantity, here:  $\mathbf{a}(t;x,y,z)$

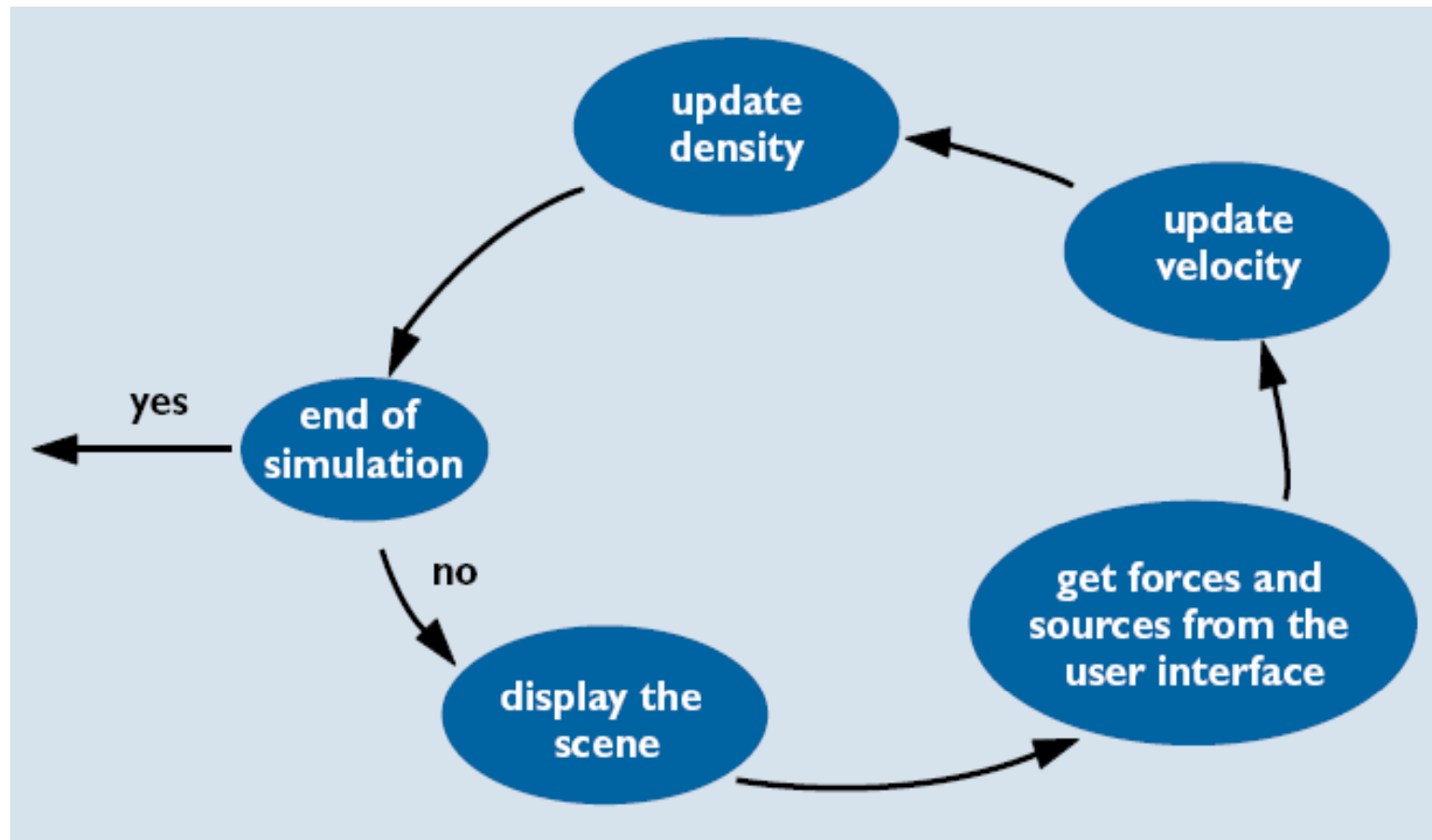
$$\frac{\partial \mathbf{a}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{a} = 0.$$

Self-advection of velocity

- Advect scalar components of velocity field individually (actually two equations in 2D, three equations in 3D)

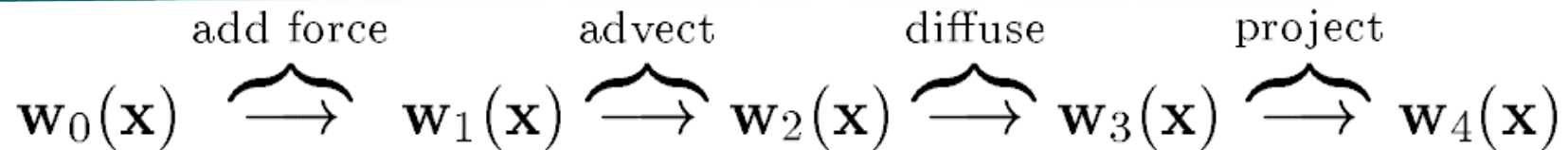
$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u}$$

# Stable Fluids Solver Overview



Courtesy Jos Stam

# Stable Fluids (1)



- Add force
- Advect
- Diffuse
- Project: solve for pressure
- Project: sub. pressure gradient

$$\mathbf{w}_1 = \mathbf{u}(\mathbf{x}, t) + \mathbf{f}(\mathbf{x}, t)\Delta t$$

$$\mathbf{w}_2 = \mathbf{w}_1(\mathbf{x} - \mathbf{w}_1\Delta t)$$

$$\left(\mathbf{I} - \nu\Delta t\nabla^2\right)\mathbf{w}_3 = \mathbf{w}_2$$

$$\nabla^2 p = \nabla \cdot \mathbf{w}_3$$

$$\mathbf{u}(\mathbf{x}, t + \Delta t) = \mathbf{w}_3 - \nabla p$$

# Stable Fluids (2)

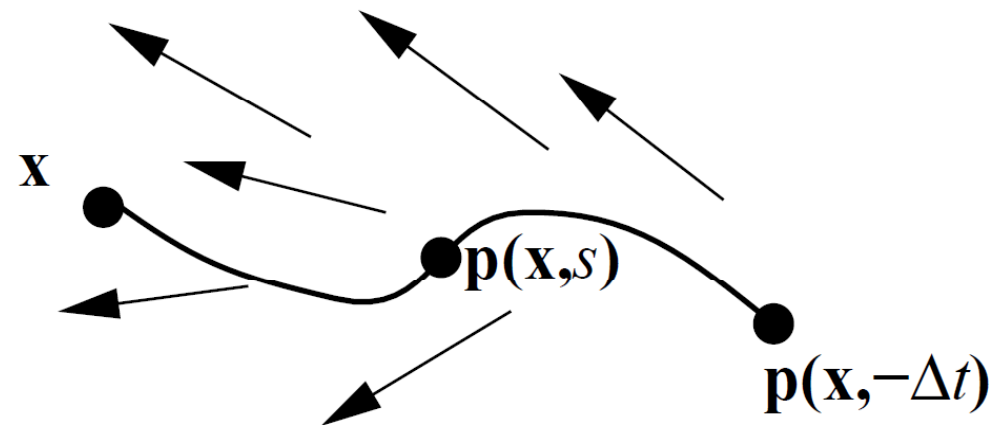


## Advect

$$\mathbf{w}_1 = \mathbf{u}(\mathbf{x}, t) + \mathbf{f}(\mathbf{x}, t)\Delta t$$

## Semi-Lagrangian advection

- Trace backwards in time
- First order scheme



# Stable Fluids (2)

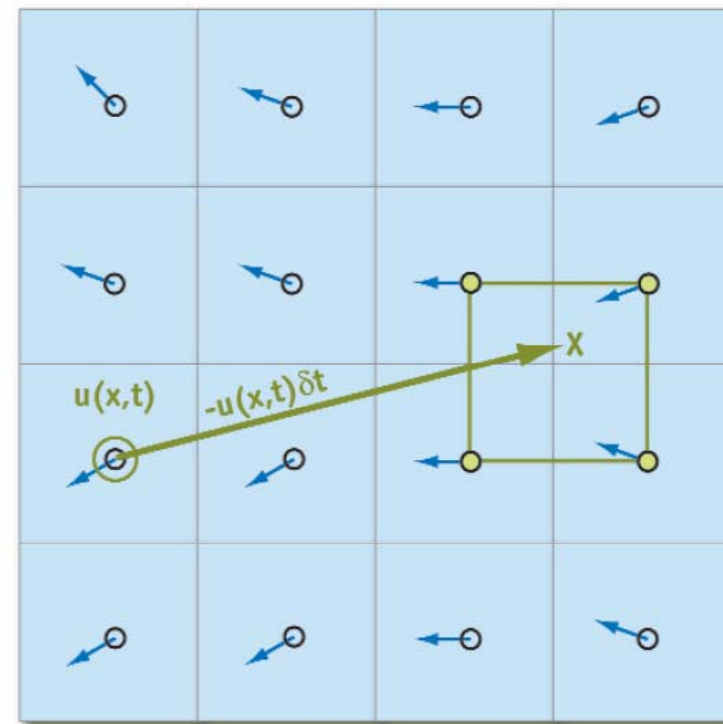


## Advect

$$\mathbf{w}_1 = \mathbf{u}(\mathbf{x}, t) + \mathbf{f}(\mathbf{x}, t)\Delta t$$

## Semi-Lagrangian advection

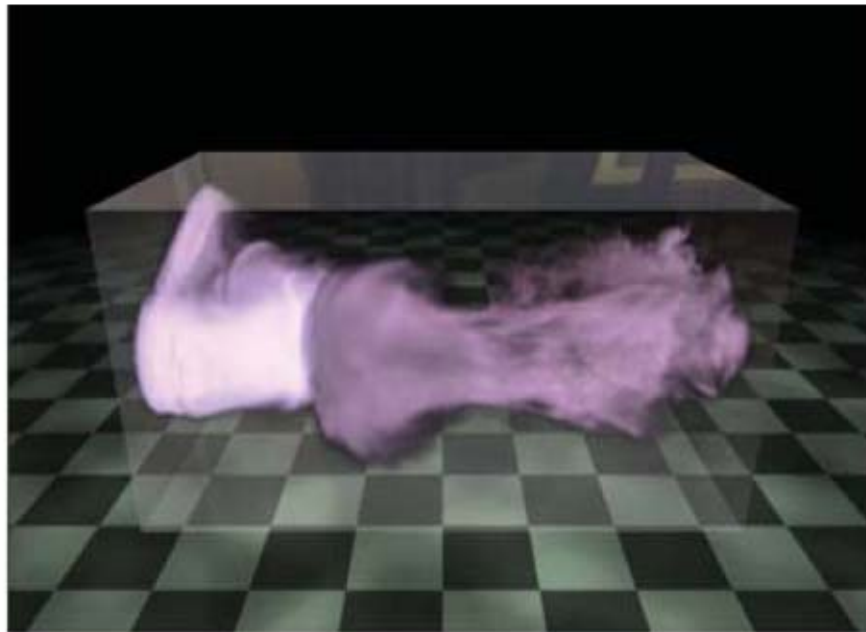
- Trace backwards in time
- First order scheme



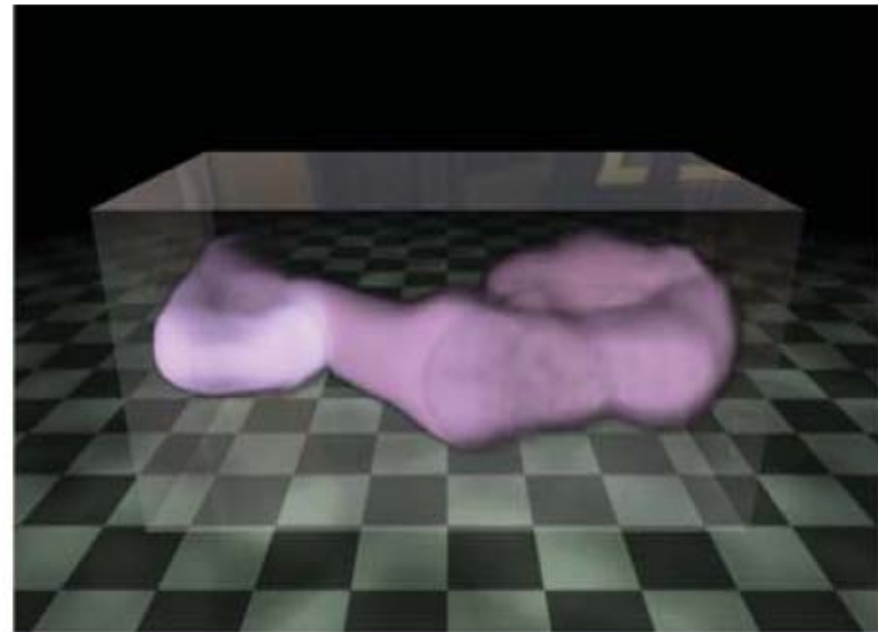
# MacCormack Advection vs. Semi-Lagrangian



Keenan Crane et al., GPU Gems 3



MacCormack advection



Semi-Lagrangian advection

Accuracy can be improved by second order scheme,  
e.g., MacCormack

- Less numerical dissipation/"diffusion" (but not unconditionally stable)

# Stable Fluids (3)



## Viscous diffusion

$$\left(\mathbf{I} - \nu\Delta t\nabla^2\right)\mathbf{w}_3 = \mathbf{w}_2$$

## Solve Poisson equation for velocity

- Discretization yields sparse system
- Jacobi [GPU Gems], Gauss-Seidel [Krüger and Westermann, 2003]
- Multigrid [Bolz et al., 2003; Goodnight et al., 2003]
- CG (conjugate gradient) [Krüger and Westermann, 2003]
- Pre-conditioned CG,  
e.g., modified incomplete Cholesky CG [Bridson, 2006, 2008]

# Stable Fluids (4)



## Solve for pressure

$$\nabla^2 p = \nabla \cdot \mathbf{w}_3$$

## Solve Poisson equation for pressure

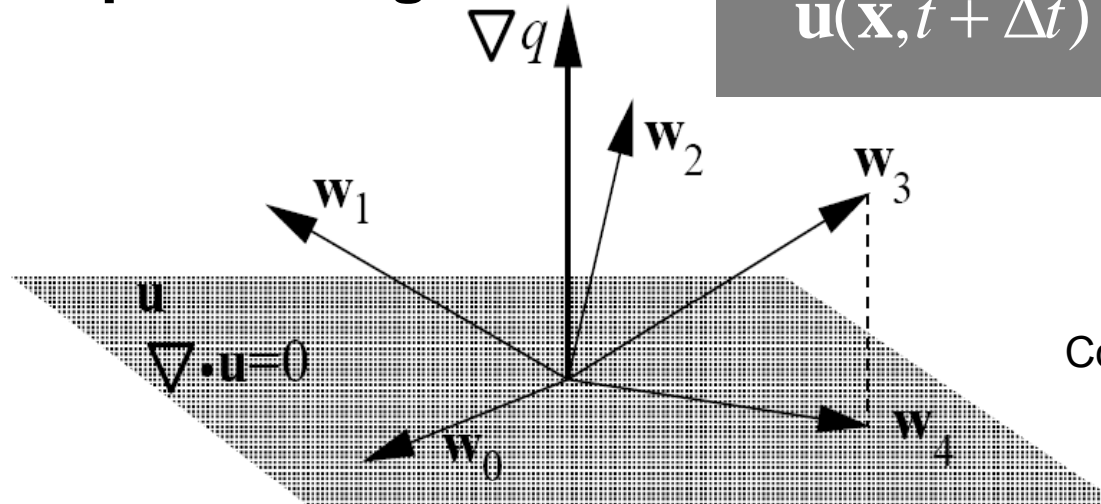
- Discretization yields sparse system
- Jacobi [GPU Gems], Gauss-Seidel [Krüger and Westermann, 2003]
- Multigrid [Bolz et al., 2003; Goodnight et al., 2003]
- CG (conjugate gradient) [Krüger and Westermann, 2003]
- Pre-conditioned CG,  
e.g., modified incomplete Cholesky CG [Bridson, 2006, 2008]

# Stable Fluids (5)



Subtract pressure gradient

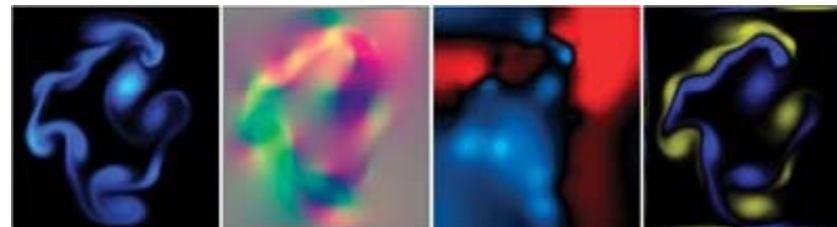
$$\mathbf{u}(\mathbf{x}, t + \Delta t) = \mathbf{w}_3 - \nabla p$$



Courtesy Jos Stam

Obtain final divergence-free velocity field

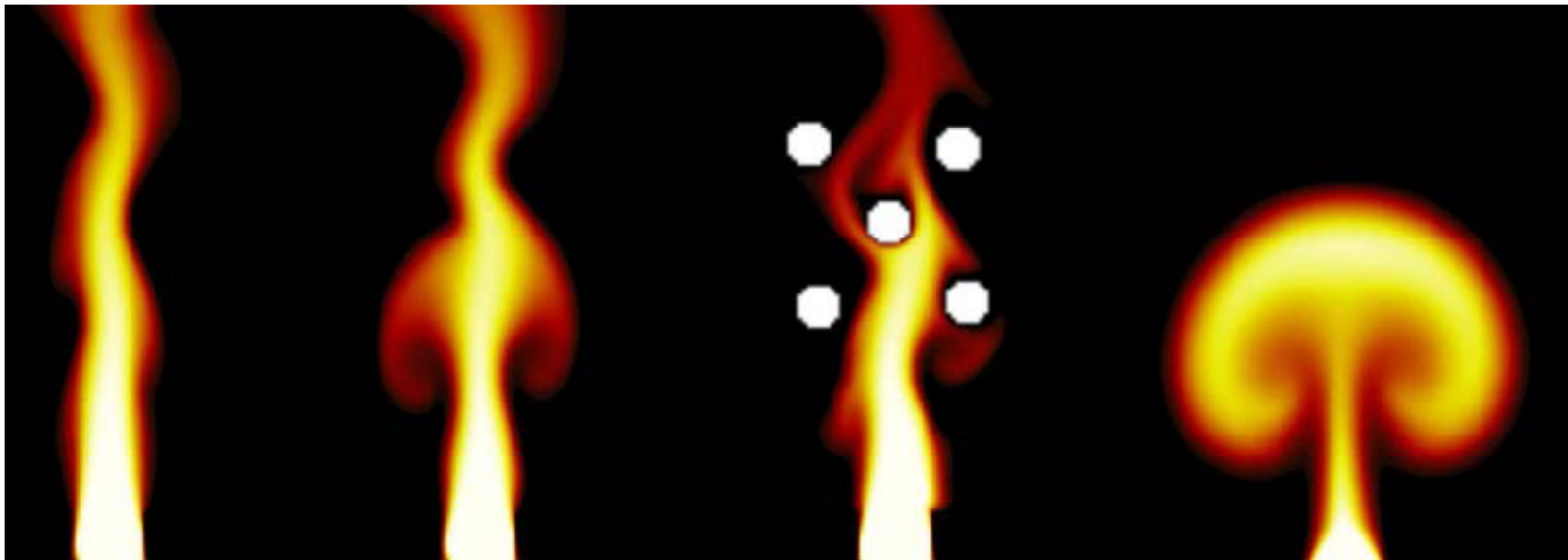
Advect other quantities (dye, smoke density, temperature, ...) using this velocity field



# GPU Fluid Flow Simulation (1)



- Solve incompressible Navier Stokes
- Use GPU matrix / linear systems solver

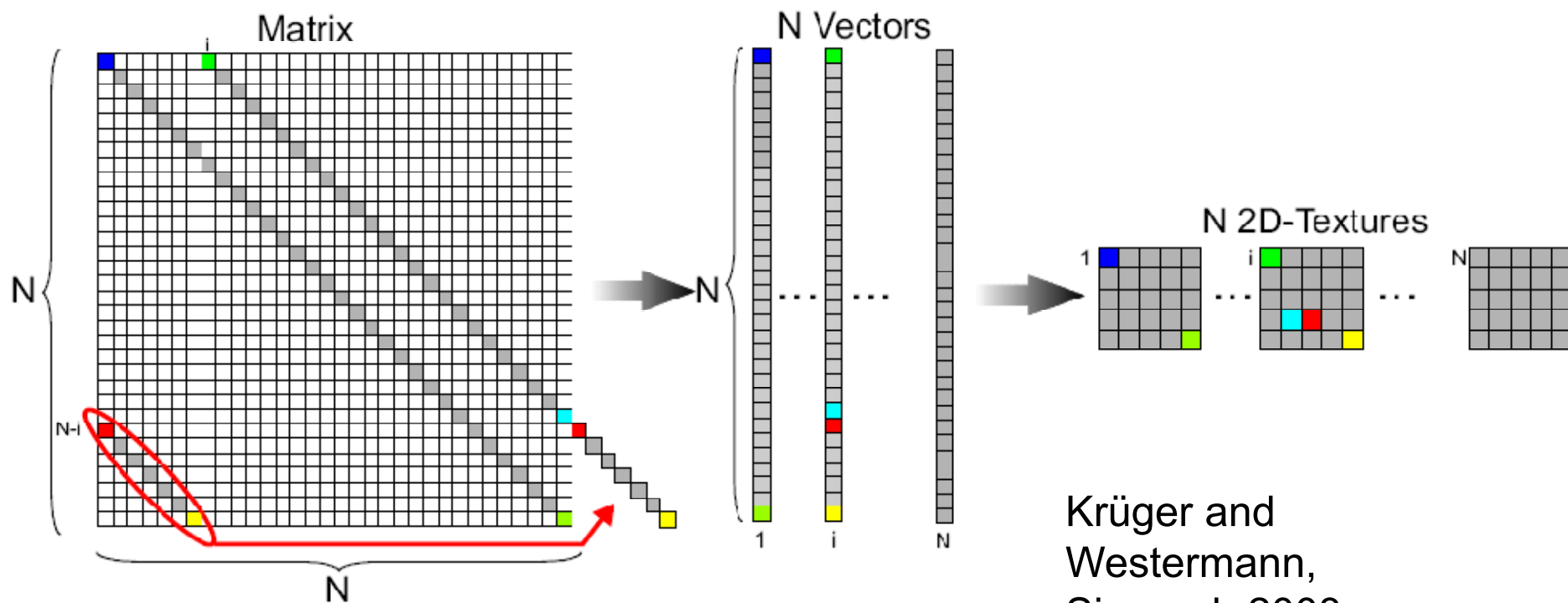


Krüger and Westermann

# GPU Fluid Flow Simulation (2)



- Matrices: sets of vectors; vectors stored in textures
- Linear algebra via texture multiplication/addition

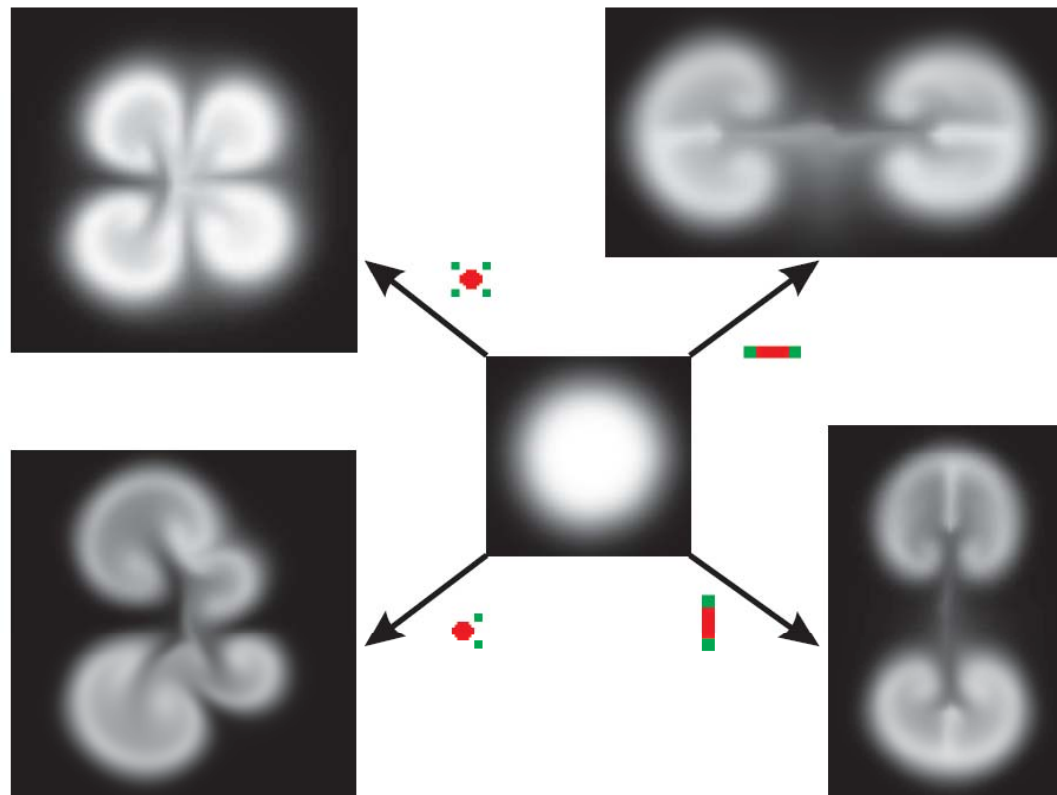


Krüger and  
Westermann,  
Siggraph 2003

# Example Velocity Field Generation



- Pressure templates



Krüger and Westermann

# Integrate Other Approaches



- Particle systems for complicated structures / sparsely populated domains
- Simulation computed on vertex buffer storing particle positions



Krüger and Westermann

# Distance Fields and Level Sets



- Additional volume: distance field

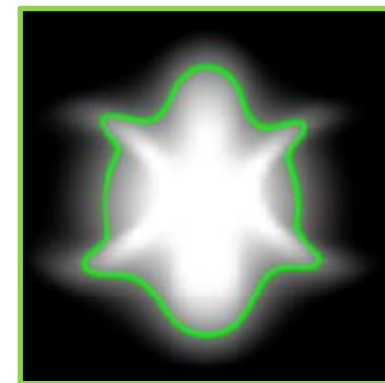
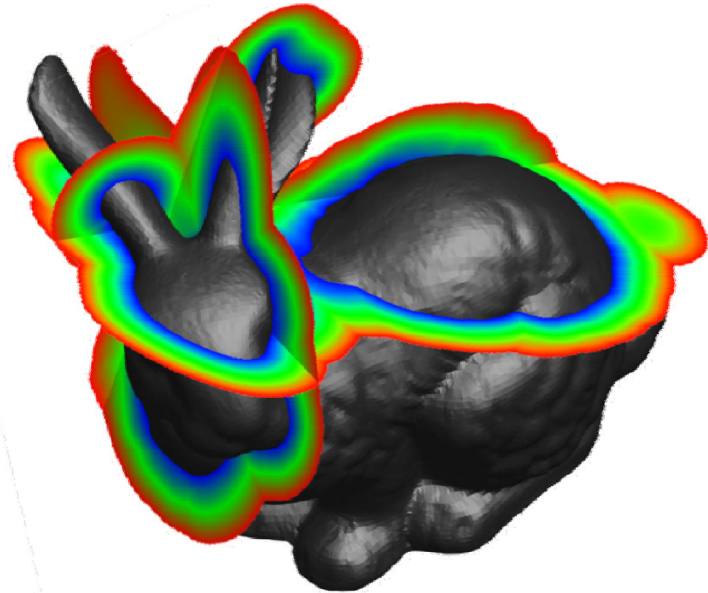
$$\phi : \mathbb{R}^3 \mapsto \mathbb{R}$$

$$S = \{\mathbf{x} | \phi(\mathbf{x}) = 0\}$$

- Solve PDE for every sample

$$\frac{\partial \phi}{\partial t} = F |\nabla \phi|$$

- Speed function F determines evolution/deformation



# Water Surface Represented as Distance Field

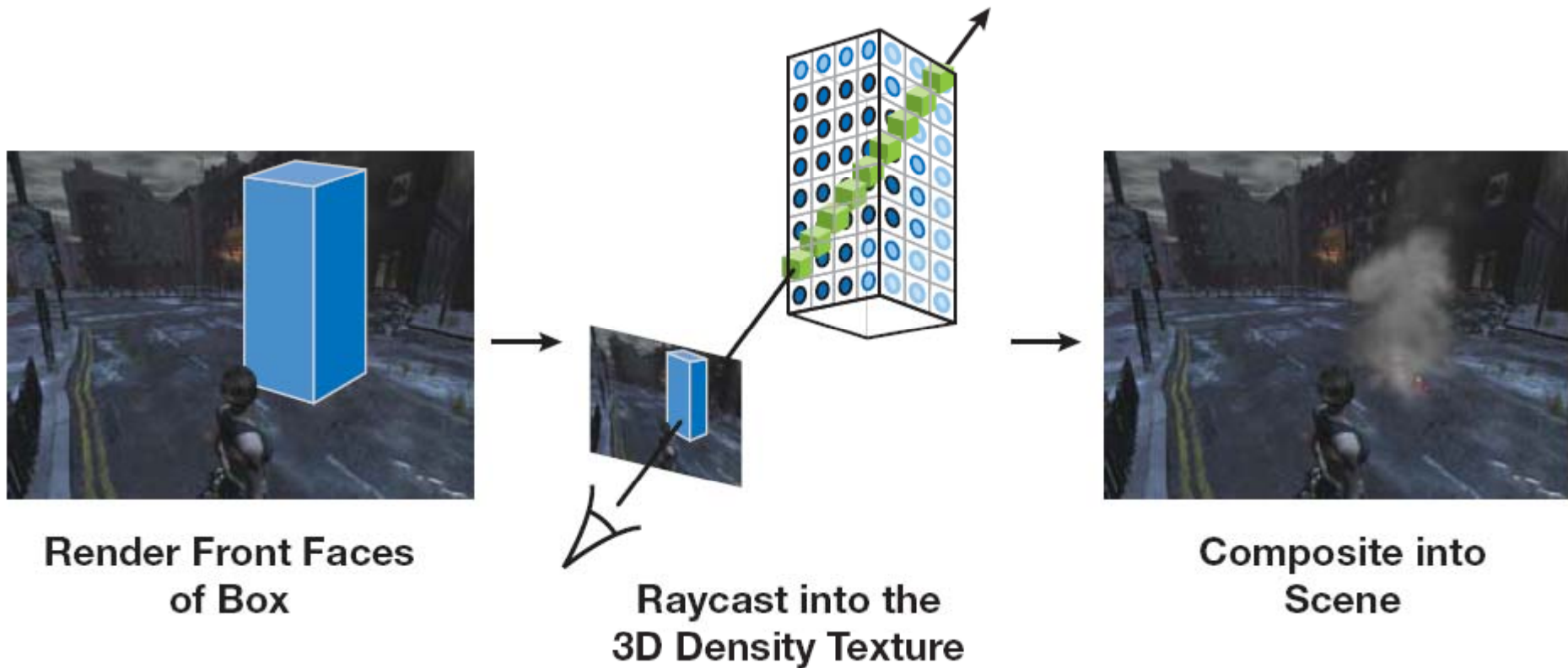


Advection pushes around signed distances

Ray-casting displays current zero level set (distance 0)



# Volume Rendering



Hellgate London / GPU Gems 3 chapter [Crane et al., 2007]

# Thank you.

Thanks for slides and images

- Mark Harris
- Keenan Crane, Ignacio Llamas, Sarah Tariq
- Jos Stam
- Christian Sigg
- Jens Krüger, Rüdiger Westermann
- Robert Bridson