CS 247 – Scientific Visualization
Lecture 28: Vector / Flow Visualization, Pt. 7

Markus Hadwiger, KAUST

Reading Assignment #14++ (1)

Reading suggestions:

- Data Visualization book, Chapter 6.7
- J. van Wijk: *Image-Based Flow Visualization*,
  ACM SIGGRAPH 2002

  `http://www.win.tue.nl/~vanwijk/ibfv/ibfv.pdf`

- T. Günther, A. Horvath, W. Bresky, J. Daniels, S. A. Buehler:
  Lagrangian Coherent Structures and Vortex Formation in High Spatiotemporal-Resolution Satellite Winds of an Atmospheric Karman Vortex Street, 2021

  `https://www.essoar.org/doi/10.1002/essoar.10506682.2`

- H. Bhatia, G. Norgard, V. Pascucci, P.-T. Bremer:
  The Helmholtz-Hodge Decomposition – A Survey, TVCG 19(8), 2013

  `https://doi.org/10.1109/TVCG.2012.316`

- Work through online tutorials of multi-variable partial derivatives, grad, div, curl, Laplacian:

  `https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives`
  `https://www.youtube.com/watch?v=rB83DpBJQsE` (3Blue1Brown)

- Matrix exponentials:

  `https://www.youtube.com/watch?v=O85OWBJ2ayo` (3Blue1Brown)

Reading Assignment #14++ (2)

Reading suggestions:

- Tobias Günther, Irene Baeza Rojo:

  Introduction to Vector Field Topology

  https://cgl.ethz.ch/Downloads/Publications/Papers/2020/Gun20b/Gun20b.pdf

- Roxana Bujack, Lin Yan, Ingrid Hotz, Christoph Garth, Bei Wang:

  State of the Art in Time-Dependent Flow Topology: Interpreting Physical Meaningfulness Through Mathematical Properties

  https://onlinelibrary.wiley.com/doi/epdf/10.1111/cgf.14037

- B. Jobard, G. Erlebacher, M. Y. Hussaini:

  Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization

  http://dx.doi.org/10.1109/TVCG.2002.1021575

- Anna Vilanova, S. Zhang, Gordon Kindlmann, David Laidlaw:

  An Introduction to Visualization of Diffusion Tensor Imaging and Its Applications

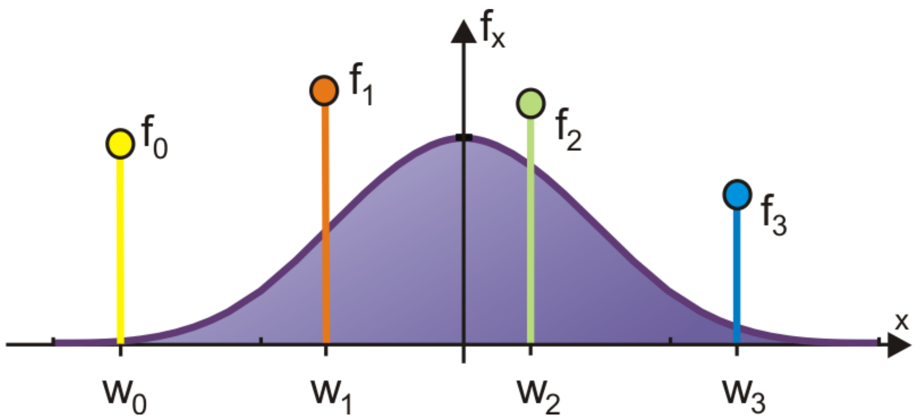
  http://vis.cs.brown.edu/docs/pdf/Vilanova-2005-IVD.pdf

Interlude:
Derivatives via Convolution

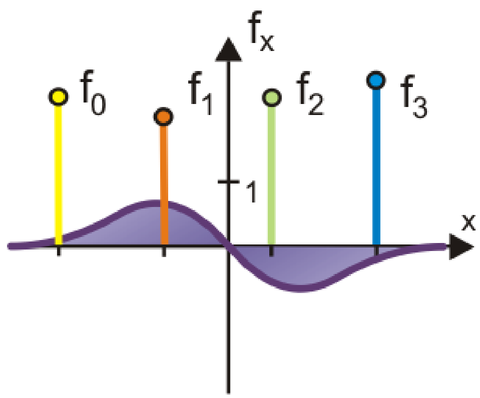
Convolve with Derivatives of Kernel

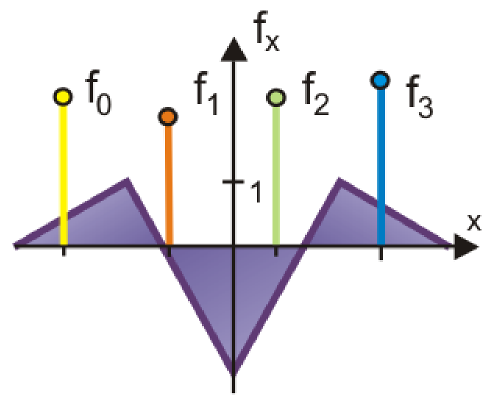
Example

- Cubic B-spline and derivatives
- Use 1D kernels and tensor product for tri-cubic



- Well-suited for curvature computation [Kindlmann et al., 2003]
- Expensive convolution?



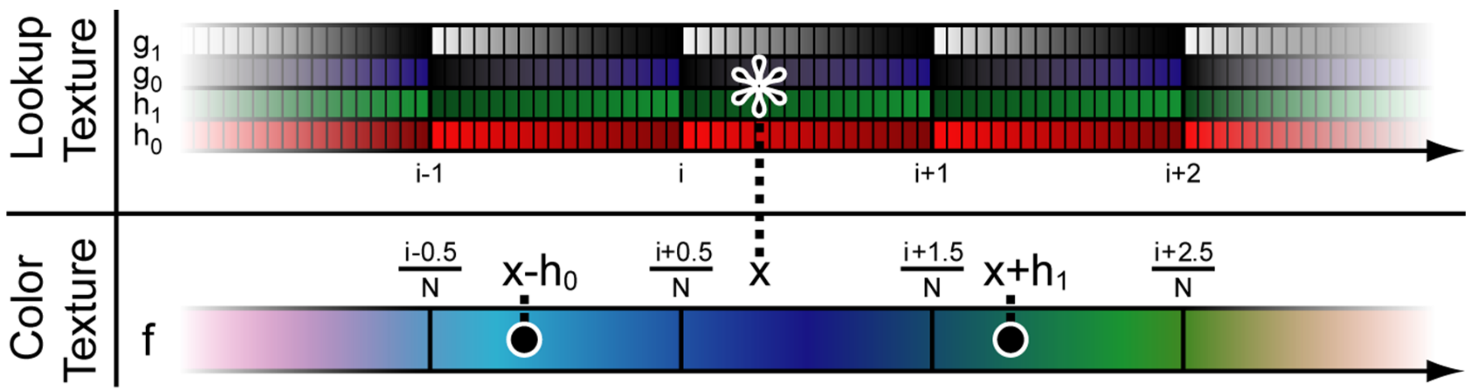


Fast Tri-Cubic Filtering on GPUs

Cubic: Need 64 neighbors; usually means 64 nearest-neighbor lookups

- But on GPUs 8 tri-linear lookups suffice for tri-cubic B-spline
- Kernels are transformed into 1D look-up textures (or simple equations)

[Sigg and Hadwiger, 2005] (GPU Gems 2)



$$g_0 \cdot f_{x-h_0} + g_1 \cdot f_{x+h_1}$$

- Newer: procedural kernel computation (see NVIDIA CUDA SDK)

Lagrangian vs. Eulerian Perspective of Fluid Flow

Lagrangian vs. Eulerian

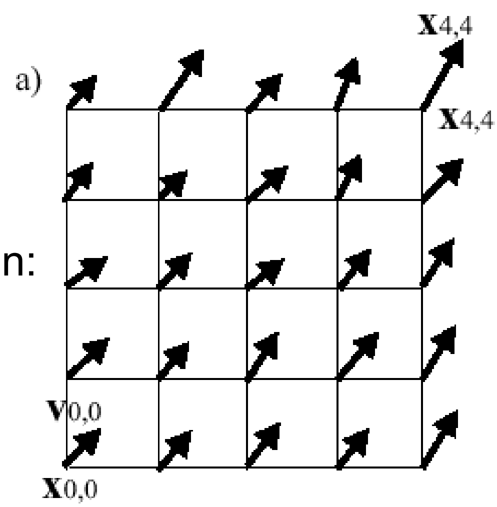
Eulerian

- Flow properties given at fixed spatial positions (grid points)
- Partial time derivative

Lagrangian

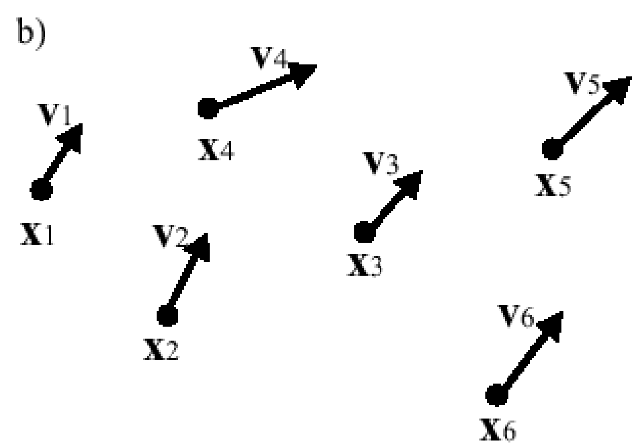
- Flow properties given for each particle (particles are moving)
- Material time derivative

Eulerian specification:



vectors given at grid points
(grid points do not move)

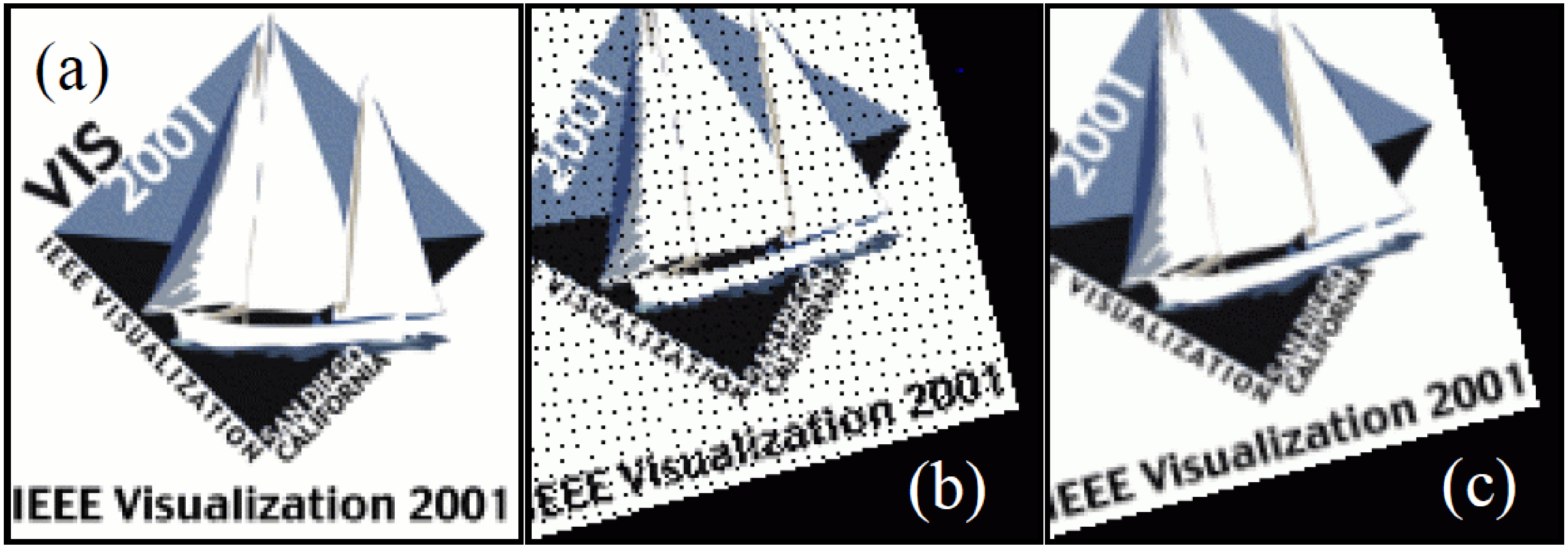
Lagrangian specification:



vectors given at particle positions
(particle positions do move)

Lagrangian vs. Eulerian

- Lagrangian: move along with the particle
- Eulerian: consider fixed point in space, look at particles moving through



- Example for pixels: rotate image (a),
  Lagrangian: move pixels forward (b),
  Eulerian: fetch pixels from backward dir. (c) (see semi-Lagrangian algo.)

Material Derivative (1)

The material time derivative (convective derivative) gives the rate of change when following a particle in the flow

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\mathbf{u} \cdot \nabla)$$

Material Derivative (1)

The material time derivative (convective derivative) gives the rate of change when following a particle in the flow

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\mathbf{u} \cdot \nabla)$$

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla) T$$

Material Derivative (1)

The material time derivative (convective derivative) gives the rate of change when following a particle in the flow

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (\mathbf{u} \cdot \nabla)$$

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla)\, T$$

$$\frac{DT}{Dt} = \frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} + w \frac{\partial T}{\partial z}$$

Material Derivative (2)

Actually, nothing else than application of the multi-variable chain rule:

$$dT = \frac{\partial T}{\partial t} dt + \frac{\partial T}{\partial x} dx + \frac{\partial T}{\partial y} dy + \frac{\partial T}{\partial z} dz$$

Material Derivative (2)

Actually, nothing else than application of the multi-variable chain rule:

$$dT = \frac{\partial T}{\partial t}dt + \frac{\partial T}{\partial x}dx + \frac{\partial T}{\partial y}dy + \frac{\partial T}{\partial z}dz$$

$$dT = \frac{\partial T}{\partial t}dt + \frac{\partial T}{\partial x}\frac{dx}{dt}dt + \frac{\partial T}{\partial y}\frac{dy}{dt}dt + \frac{\partial T}{\partial z}\frac{dz}{dt}dt$$

Material Derivative (2)

Actually, nothing else than application of the multi-variable chain rule:

$$dT = \frac{\partial T}{\partial t}dt + \frac{\partial T}{\partial x}dx + \frac{\partial T}{\partial y}dy + \frac{\partial T}{\partial z}dz$$

$$dT = \frac{\partial T}{\partial t}dt + \frac{\partial T}{\partial x}\frac{dx}{dt}dt + \frac{\partial T}{\partial y}\frac{dy}{dt}dt + \frac{\partial T}{\partial z}\frac{dz}{dt}dt$$

$$\frac{dT}{dt} = \frac{\partial T}{\partial t} + \frac{\partial T}{\partial x}\frac{dx}{dt} + \frac{\partial T}{\partial y}\frac{dy}{dt} + \frac{\partial T}{\partial z}\frac{dz}{dt}$$

Material Derivative (2)

Actually, nothing else than application of the multi-variable chain rule:

$$dT = \frac{\partial T}{\partial t}dt + \frac{\partial T}{\partial x}dx + \frac{\partial T}{\partial y}dy + \frac{\partial T}{\partial z}dz$$

$$dT = \frac{\partial T}{\partial t}dt + \frac{\partial T}{\partial x}\frac{dx}{dt}dt + \frac{\partial T}{\partial y}\frac{dy}{dt}dt + \frac{\partial T}{\partial z}\frac{dz}{dt}dt$$

$$\frac{dT}{dt} = \frac{\partial T}{\partial t} + \frac{\partial T}{\partial x}\frac{dx}{dt} + \frac{\partial T}{\partial y}\frac{dy}{dt} + \frac{\partial T}{\partial z}\frac{dz}{dt}$$

$$u := \frac{dx}{dt}, \qquad v := \frac{dy}{dt}, \qquad w := \frac{dz}{dt}$$

Material Derivative (2)

Actually, nothing else than application of the multi-variable chain rule:

We are given $T(x, y, z, t)$ with four independent variables;

But now we want to go along a parameterized path with parameter t, so x, y, z become dependent variables: $x(t),\ y(t),\ z(t)$

Along this path, our goal is now to compute the derivative of the function

$T\left(x(t), y(t), z(t), t\right)$ with t as only independent variable:

$$\frac{d}{dt} T\left(x(t), y(t), z(t), t\right) =$$

$$\frac{\partial}{\partial t} T(x,y,z,t) + \frac{\partial}{\partial x} T(x,y,z,t) \frac{d}{dt} x(t) + \frac{\partial}{\partial y} T(x,y,z,t) \frac{d}{dt} y(t) + \frac{\partial}{\partial z} T(x,y,z,t) \frac{d}{dt} z(t)$$

$$u(t) := \frac{dx(t)}{dt}, \qquad v(t) := \frac{dy(t)}{dt}, \qquad w(t) := \frac{dz(t)}{dt}$$

Advection

Advection equation; velocity field **u(** *x, y, z, t* **)**,
no change following particle, just advection:
set material derivative = 0:

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

In the Navier-Stokes equations: “self-advection” of velocity

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

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

this is equivalent to saying that the acceleration is zero!

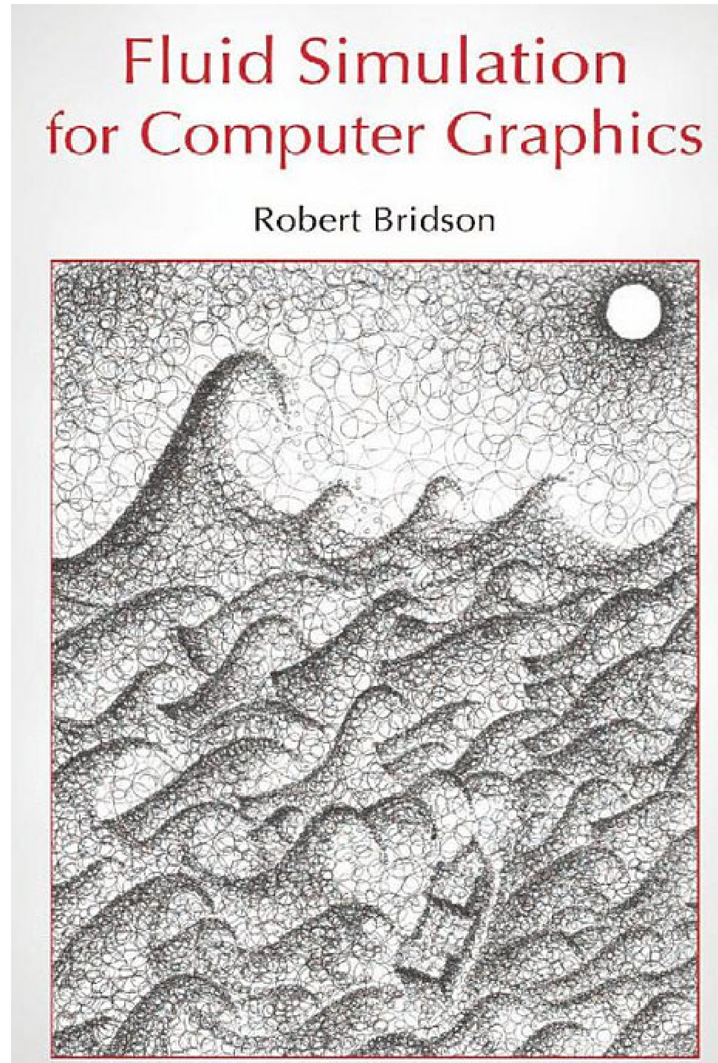
Fluid Simulation Basics

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



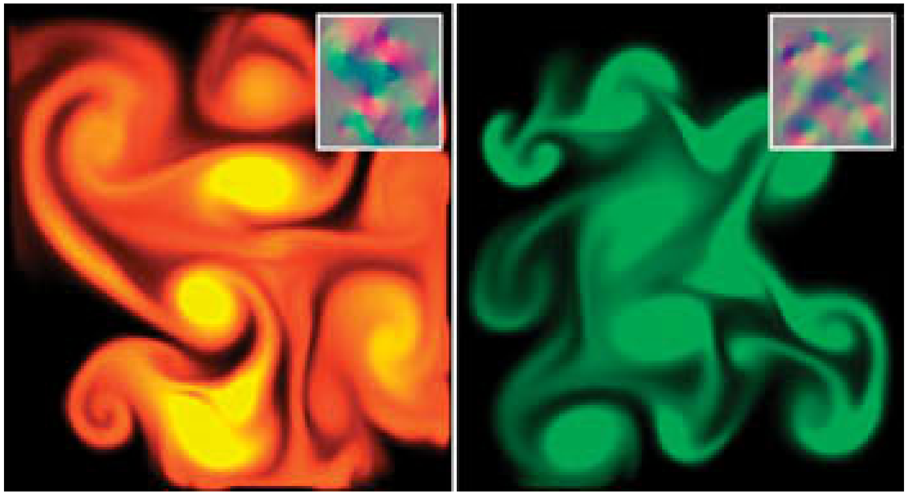
Fluid Simulation and Rendering

Compute advection of fluid

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

Discretized domain

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

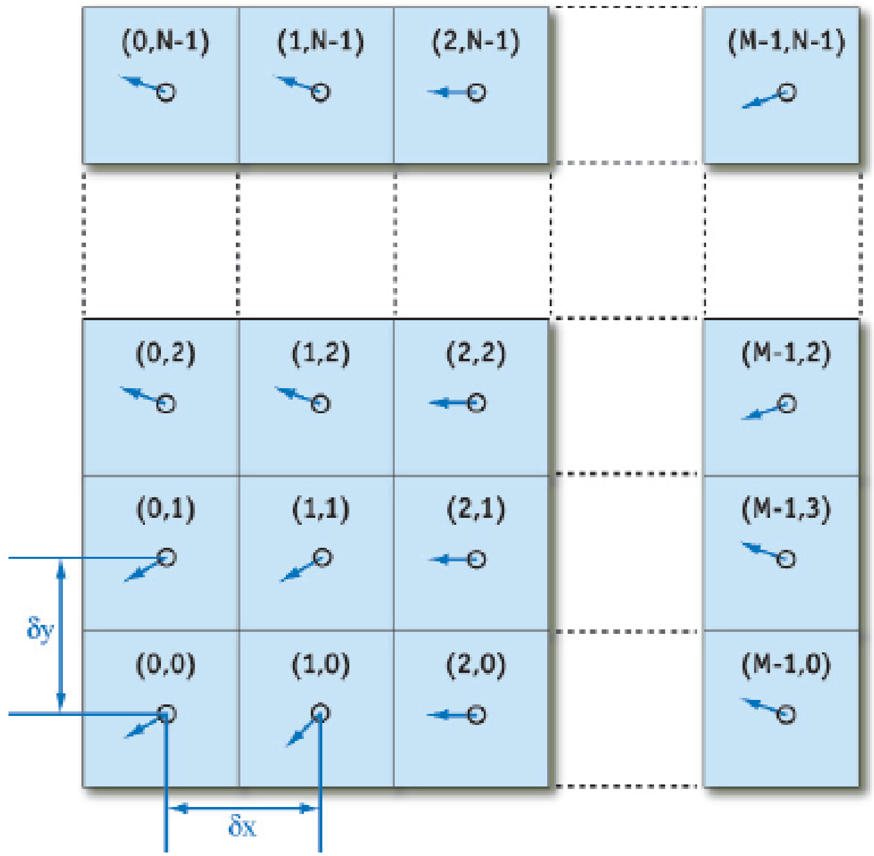


Courtesy Mark Harris

Velocity Field

2D or 3D vector field

- Stored in 2D or 3D texture/array



Vector Calculus

Gradient

- Scalar field → 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 → scalar field
- Density exit rate (source?, sink?)

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

Laplacian

- Scalar field → scalar field
- Divergence of gradient

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

Fluid Simulation: Navier Stokes (1)

Incompressible (divergence-free) Navier Stokes equations

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

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

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 (1)

Incompressible (divergence-free) Navier Stokes equations

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

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

this is the velocity gradient tensor!

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)

Given a (Cartesian) coordinate system, the momentum equation can be seen as a system of equations (2 equations in 2D, 3 equations in 3D)

For 2D (Cartesian):

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

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

these are PDEs!

Fluid Simulation: Navier Stokes (2)

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

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

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

Advection

Advection operator, with velocity field **u(**t;*x,y,z***)**

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

- Advection of scalar quantity, here: **a(**t;*x,y,z***)**, with incomp. flow:

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

Self-advection of velocity

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

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

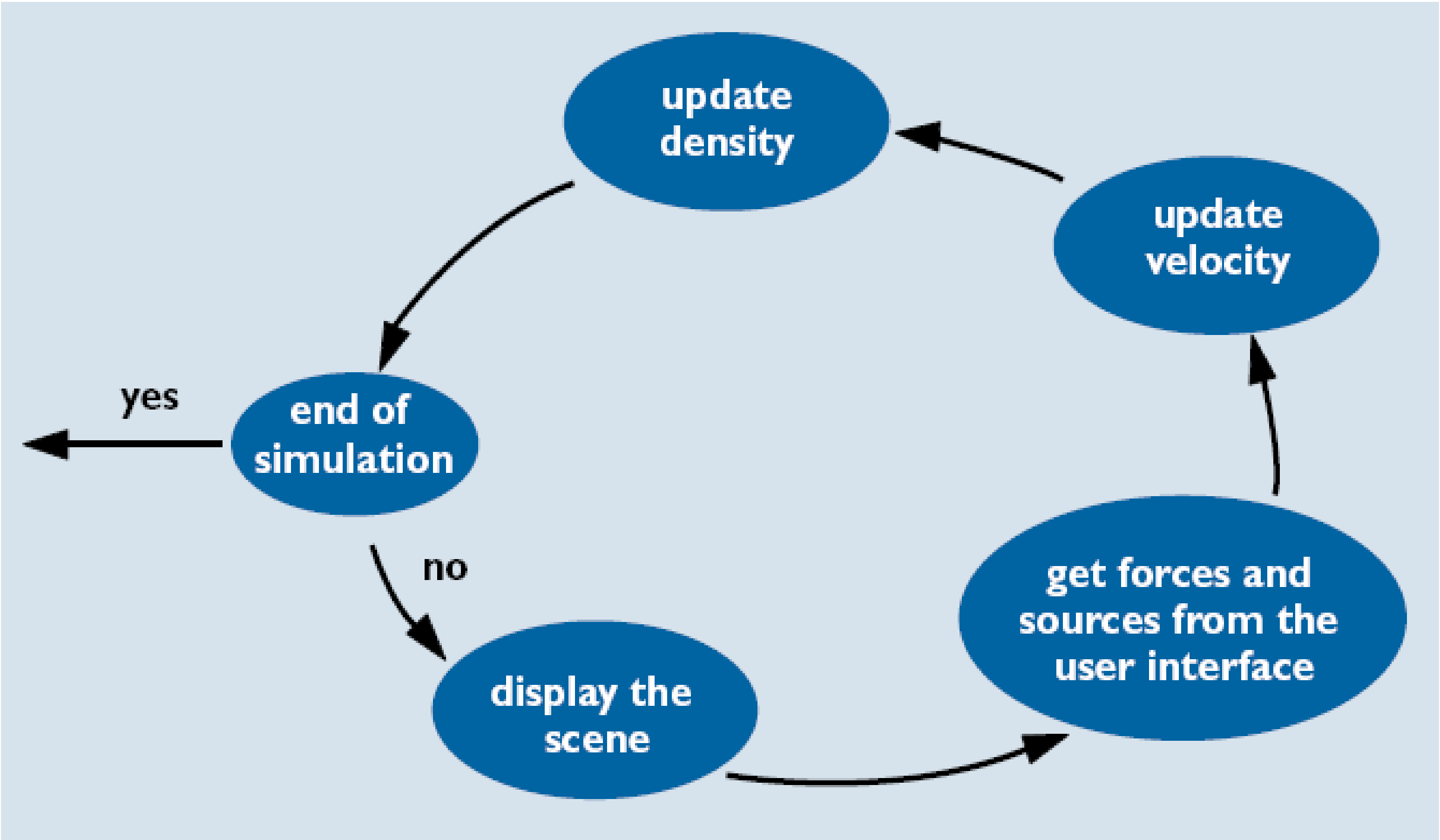
Vector Calculus: Finite Difference Approximations

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	$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}$	$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\delta y)^2}$

Stable Fluids Solver Overview



Courtesy Jos Stam

Stable Fluids (1)

$$\mathbf{w}_0(\mathbf{x}) \overbrace{\longrightarrow}^{\text{add force}} \mathbf{w}_1(\mathbf{x}) \overbrace{\longrightarrow}^{\text{advect}} \mathbf{w}_2(\mathbf{x}) \overbrace{\longrightarrow}^{\text{diffuse}} \mathbf{w}_3(\mathbf{x}) \overbrace{\longrightarrow}^{\text{project}} \mathbf{w}_4(\mathbf{x})$$

- Add force

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

- Advect

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

- Diffuse

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

- Project: solve for pressure

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

- Project: sub. pressure gradient

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

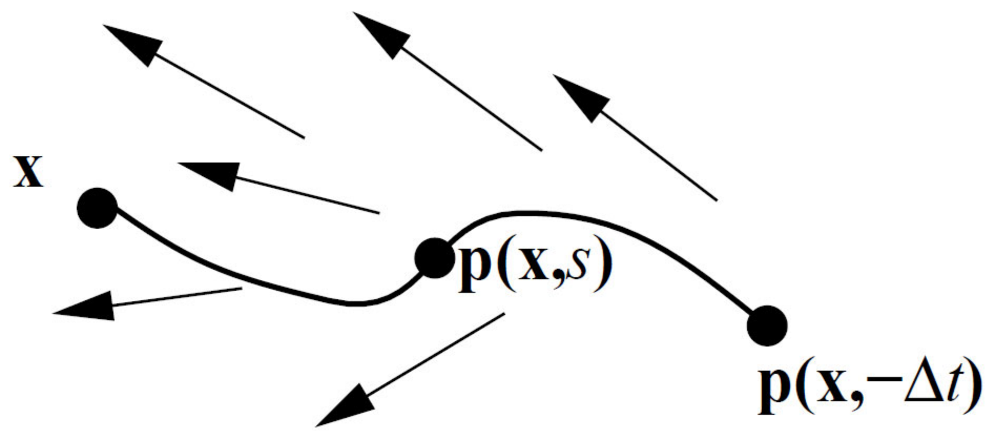
Stable Fluids (2)

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



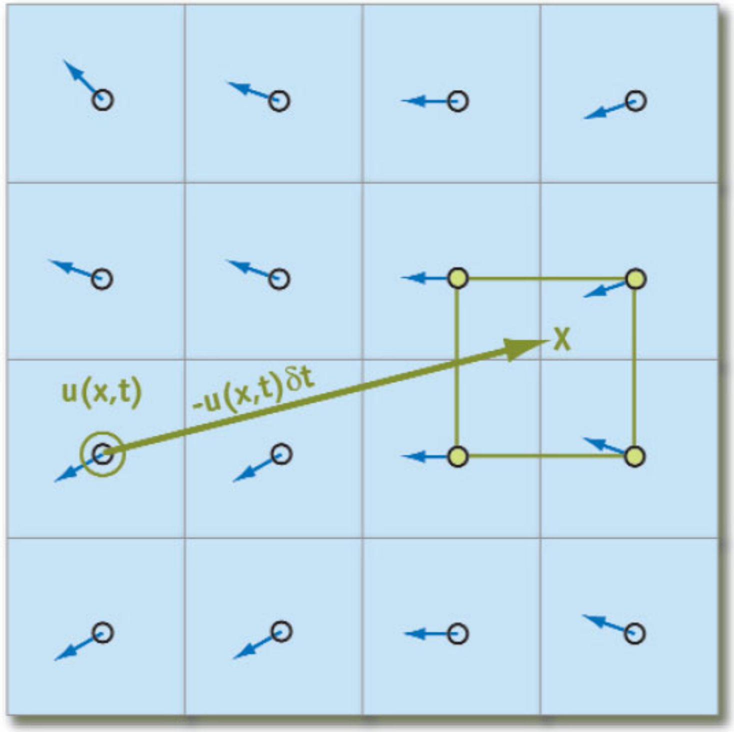
Stable Fluids (2)

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



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

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

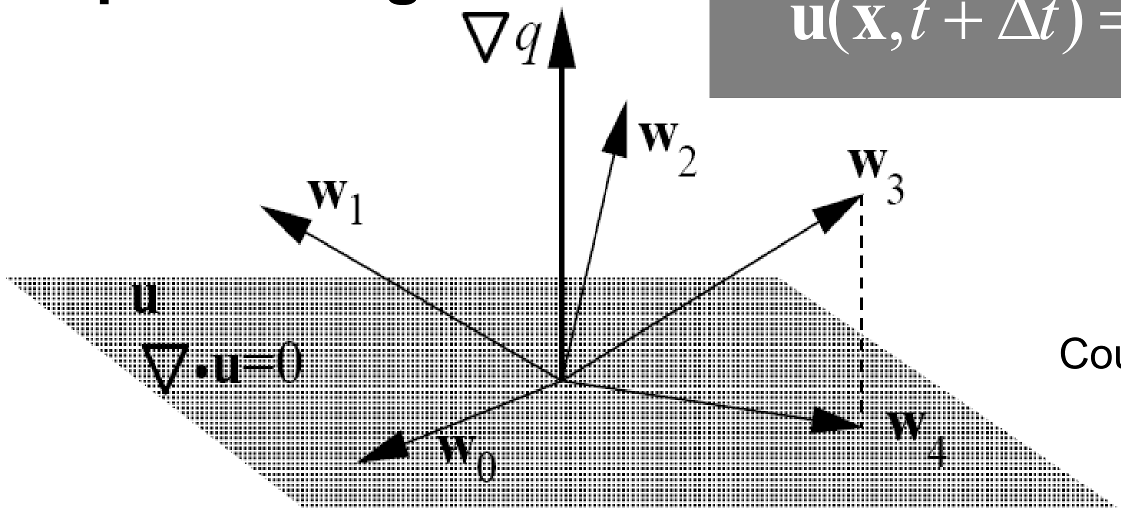
Solve Poisson equation for pressure

- 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

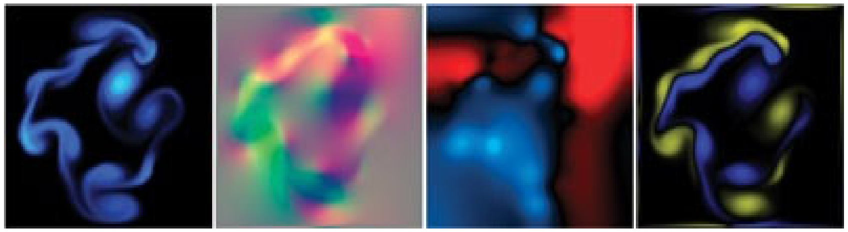
$$\mathbf{u}(\mathbf{x}, t+\Delta t) = \mathbf{w}_3 - \nabla p$$



Courtesy Jos Stam

Obtain final divergence-free velocity field

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



Distance Fields and Level Sets

- Additional volume: distance field

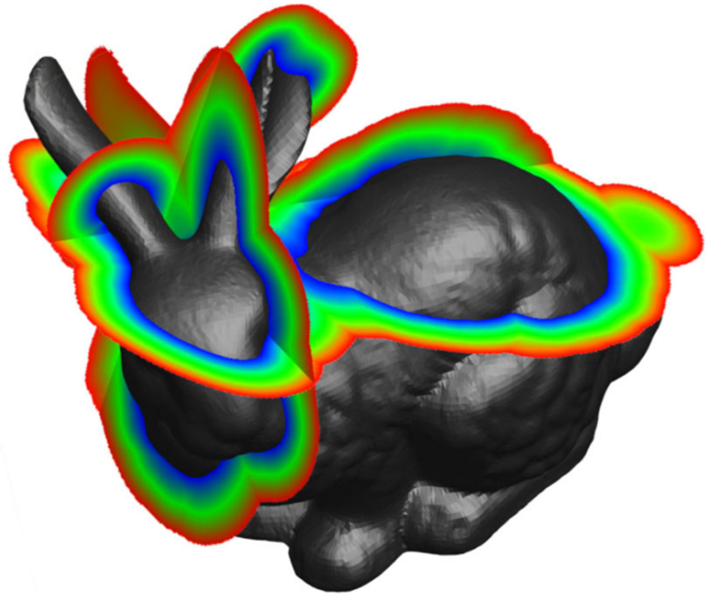
$$\phi : \mathbb{R}^3 \mapsto \mathbb{R}$$

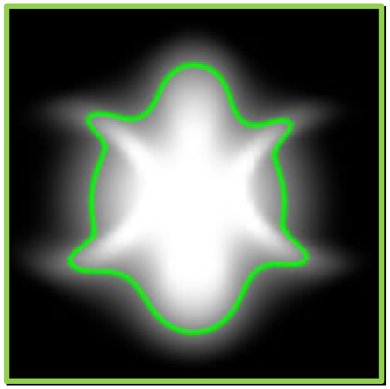
$$S = \{\mathbf{x} | \phi(\mathbf{x}) = 0\}$$

- Solve PDE for every sample

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

- Speed function F determines evolution/deformation

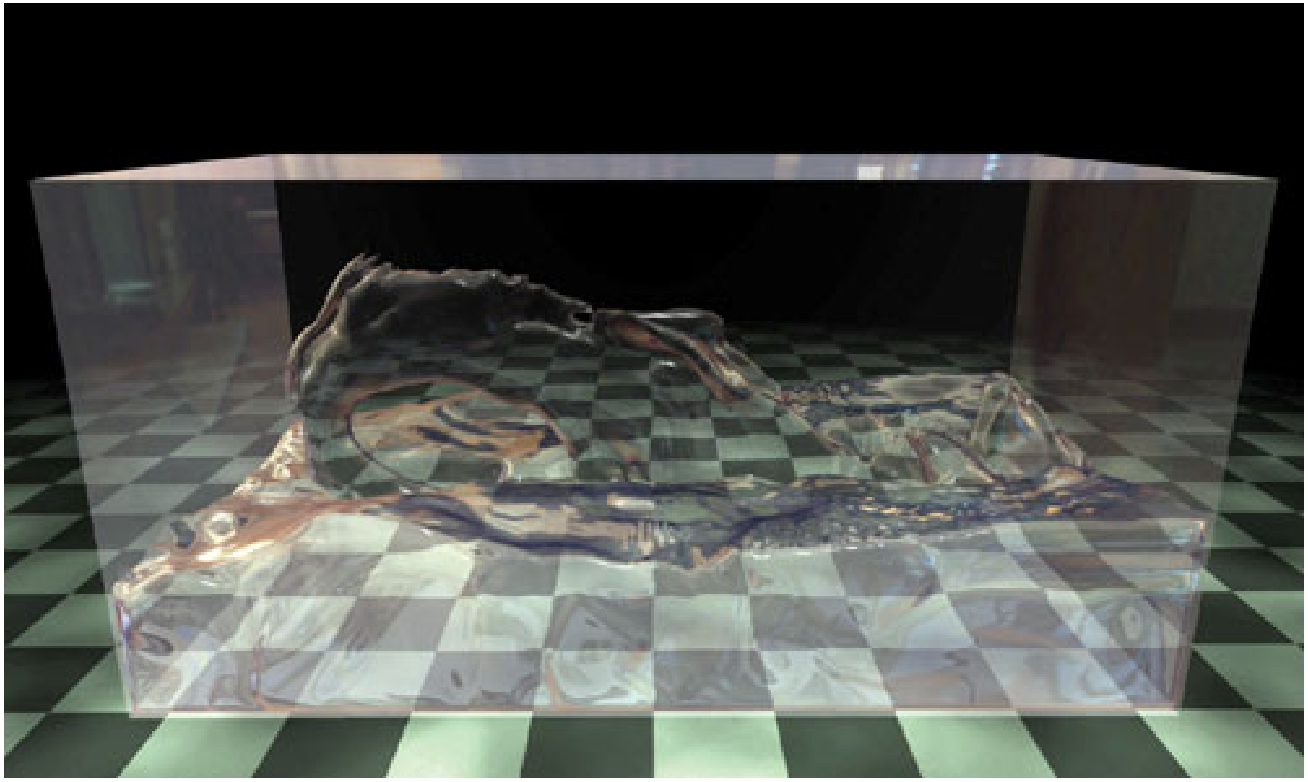




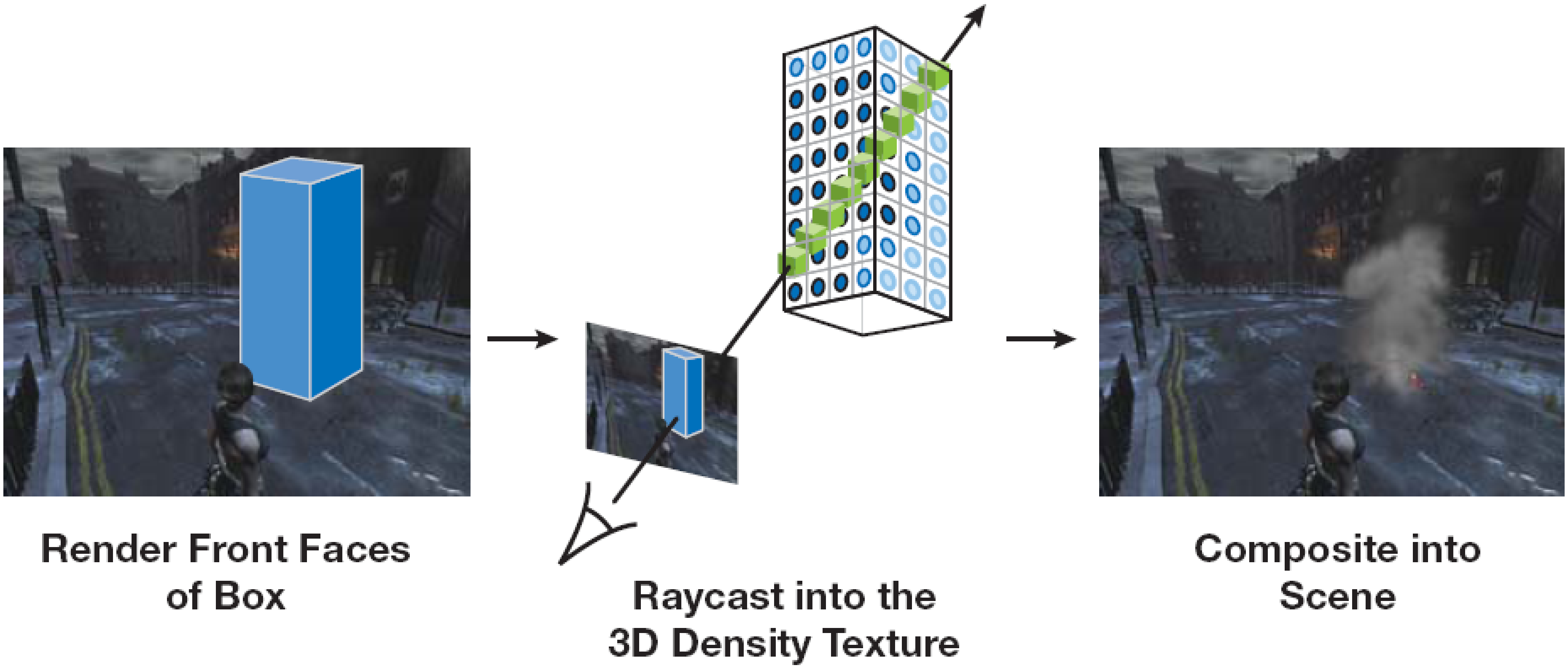
Water Surface Represented as Distance Field

Advection pushes around signed distances

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



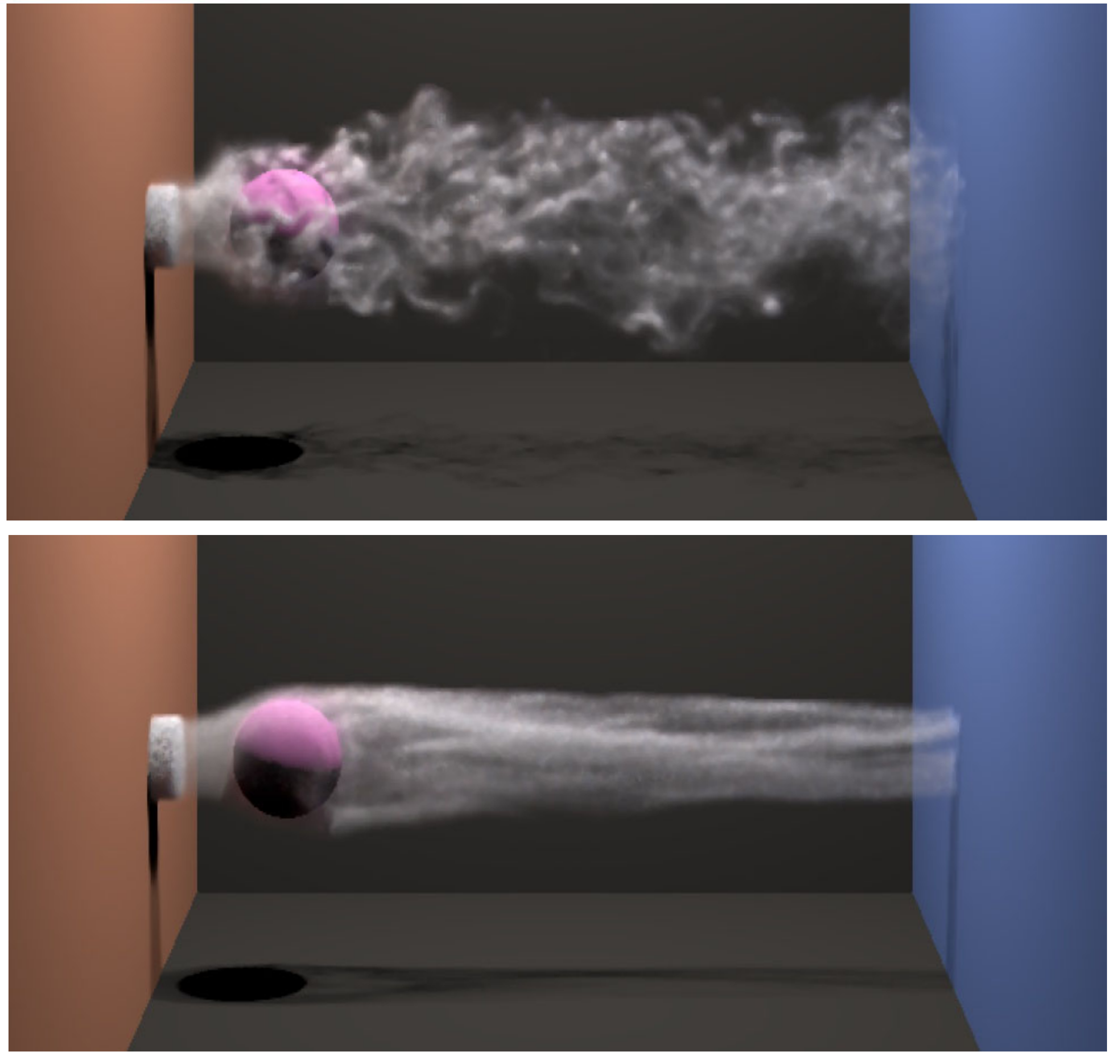
Volume Rendering



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

Energy-Preserving Integrators

- Eulerian scheme
- No numerical dissipation
- Easier to control intended viscosity



- [Mullen et al., SIGGRAPH 2009]

Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama