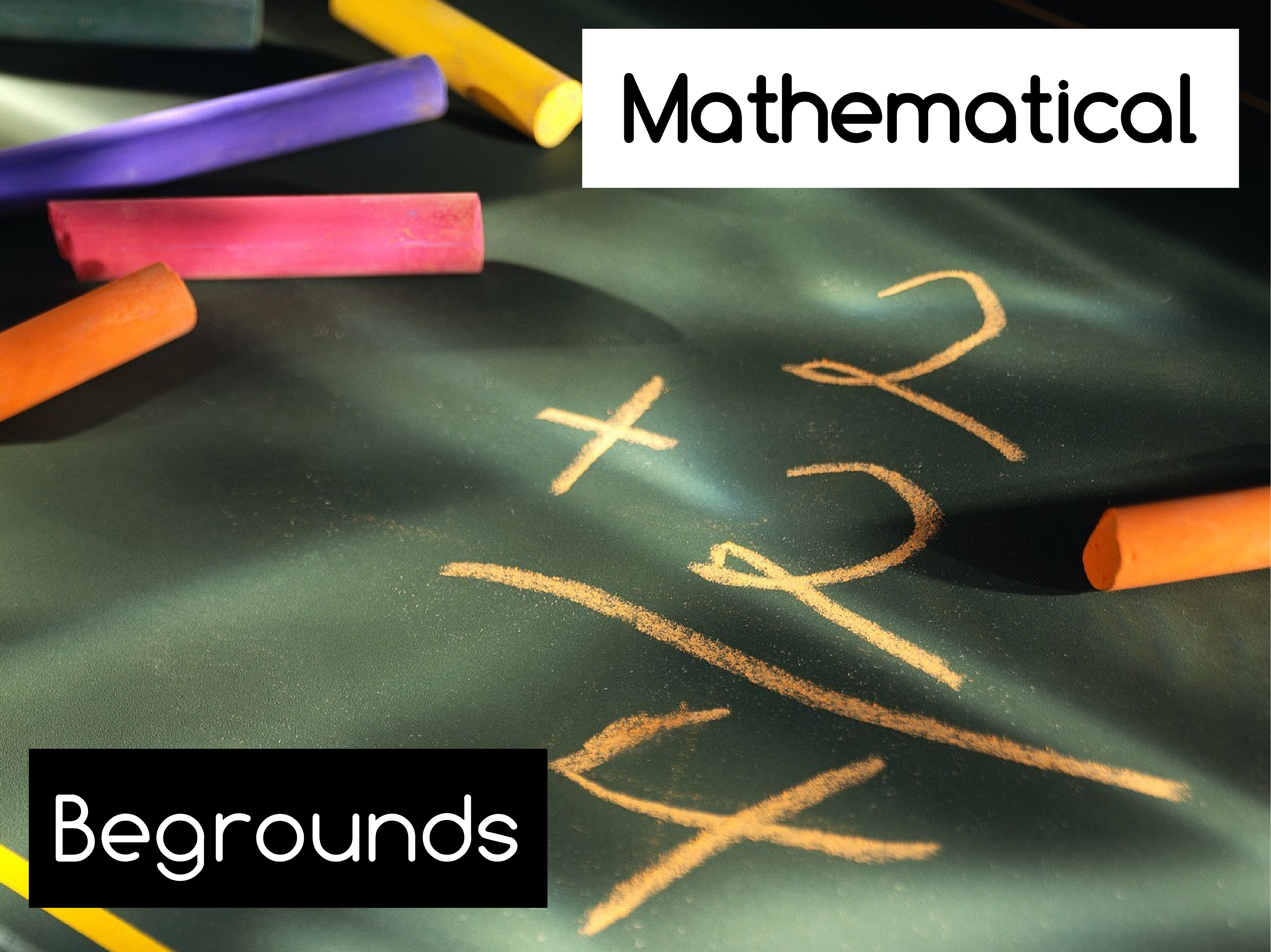
Lesson 10

Fluid Fire and Smoke

Lesson 09 Outline

- Problem definition and motivations
- Mathematical Begrounds
- Fluid dynamics and Navier-Stokes equations
- Grid based MAC method
- Particle based SPH method
- Neighbor search for coupled particles
- Demos / tools / libs


Mathematical
Begrounds

Motivations

- Dynamics of incompressible fluids is governed by the following Navier-Stokes equations

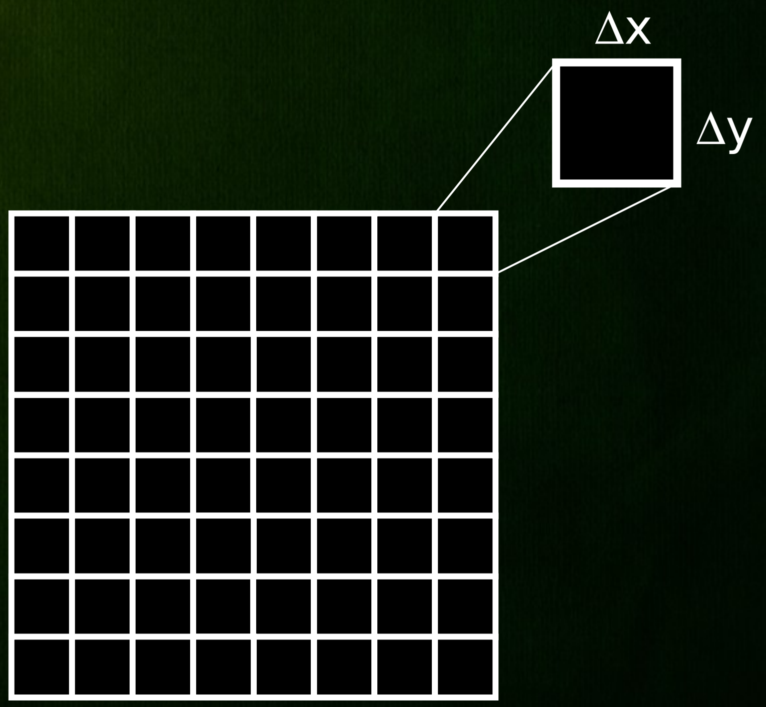
$$\nabla \circ \mathbf{u} = \mathbf{0}$$

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

- Motivation: We need to understand the **math** behind !

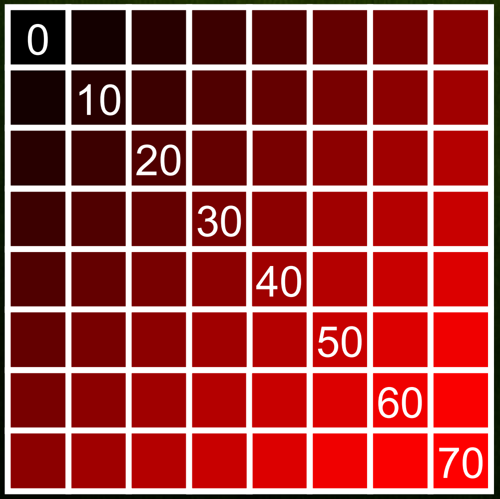
Spatial Discretization

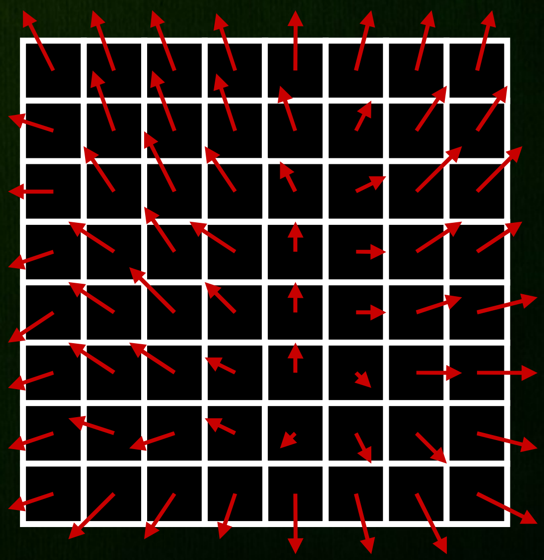
- Virtually split simulation space into finite elements
- Irregular finite elements
  - Octrees, tetrahedral meshes, ...
- Regular finite elements
  - Regular grids



Scalar and Vector Fields

- Scalar field is a function mapping a location in the simulation space to a scalar value
- Vector field is a function mapping a location in the simulation space to a vector value





Scalar and Vector Field Notation

- Scalar field
  - $f: R^n \rightarrow R$
  - $f(\mathbf{x}) = a$
- Vector field
  - $F: R^n \rightarrow R^m$
  - $F(\mathbf{x}) = \mathbf{a}$
- 2D/3D Scalar fields
  - $f(x, y) = a$
  - $f(x, y, z) = a$
- 2D/3D Vector fields
  - $F(x, y) = (u, v)$
  - $F(x, y, z) = (u, v, w)$
  - $u(x, y, z) = a$
  - $v(x, y, z) = b$
  - $w(x, y, z) = c$

Calculus – Partial Derivative

- Partial Derivative (∂) of a function of several variables is its derivative with respect to one of those variables with the others held constant

$$f_x(x,y,z) = \frac{\partial f(x,y,z)}{\partial x} = \lim_{h\to 0}\frac{f(x+h,y,z)-f(x-h,y,z)}{2\mathrm{h}}$$

$$f_y(x,y,z) = \frac{\partial f(x,y,z)}{\partial y} = \lim_{h\to 0}\frac{f(x,y+h,z)-f(x,y-h,z)}{2\mathrm{h}}$$

$$f_z(x,y,z) = \frac{\partial f(x,y,z)}{\partial z} = \lim_{h\to 0}\frac{f(x,y,z+h)-f(x,y,z-h)}{2\mathrm{h}}$$

Calculus – Finite Differences

- Forward derivative

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

- Forward difference

$$f_x^+ = \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

- Backward derivative

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x, y, z) - f(x-h, y, z)}{h}$$

- Backward difference

$$f_x^- = \frac{f(x, y, z) - f(x-h, y, z)}{h}$$

- Central derivative

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y, z) - f(x-h, y, z)}{2h}$$

- Central difference

$$f_x^0 = \frac{f(x+h, y, z) - f(x-h, y, z)}{2h}$$

Calculus – Gradient Operator

- Gradient of a scalar field is a vector field which points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change.
- Gradient operator (∇) is a vector of partial derivatives

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \qquad \nabla \mathbf{u} = \left(\frac{\partial \mathbf{u}}{\partial x}, \frac{\partial \mathbf{u}}{\partial y}, \frac{\partial \mathbf{u}}{\partial z}\right)$$

Calculus – Gradient Operator

- First-order finite differences

$$u_x(x,y,z) = \frac{u(x+h,y,z)-u(x,y,z)}{h}$$

$$v_y(x,y,z) = \frac{v(x,y+h,z)-v(x,y,z)}{h}$$

$$w_z(x,y,z) = \frac{w(x,y,z+h)-w(x,y,z)}{h}$$

- Finite difference of Gradient Operator

$$\mathbf{u} = (u,v,w) \qquad \mathbf{u}(x,y,z) = \left(u(x,y,z),v(x,y,z),w(x,y,z)\right)$$

$$\nabla\mathbf{u}(x,y,z) = \left(u_x(x,y,z),v_y(x,y,z),w_z(x,y,z)\right) =$$
$$\left(\frac{u(x+h,y,z)-u(x,y,z)}{h},\frac{v(x,y+h,z)-v(x,y,z)}{h},\frac{w(x,y,z+h)-w(x,y,z)}{h},\right)$$

Calculus – Divergence of field

- Divergence ($\nabla\cdot$) is an operator that measures the magnitude of a vector field's source or sink at a given point
- Divergence of a vector field is a (signed) scalar

$$\mathbf{u} = (u, v, w)$$

$$\nabla \circ \mathbf{u} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \circ (u, v, w)$$

$$= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = u_x + u_y + u_z$$

Calculus – Divergence of field

- First-order finite differences

$$u_x(x,y,z) = \frac{u(x+h,y,z)-u(x,y,z)}{h}$$

$$v_y(x,y,z) = \frac{v(x,y+h,z)-v(x,y,z)}{h}$$

$$w_z(x,y,z) = \frac{w(x,y,z+h)-w(x,y,z)}{h}$$

- Finite difference of Gradient Operator

$$\mathbf{u} = (u,v,w) \qquad \mathbf{u}(x,y,z) = (u(x,y,z),v(x,y,z),w(x,y,z))$$

$$\nabla\circ\mathbf{u}(x,y,z) = u_x(x,y,z)+v_y(x,y,z)+w_z(x,y,z) =$$
$$\frac{u(x+h,y,z)-u(x,y,z)+v(x,y+h,z)-v(x,y,z)+w(x,y,z+h)-w(x,y,z)}{h}$$

Calculus – Laplacian operator

- Laplacian roughly describes how much values in the original field differ from their neighborhood average
- Laplacian operator (∇^2) is defined as the divergence of a gradient

$$\nabla^2 = \nabla \circ \nabla = \frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \frac{\partial^2}{\partial z^2}$$

- Laplacian of a scalar *u* and vector **u** field

$$\nabla \circ \nabla u = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \circ \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}\right) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

$$\nabla^2 \mathbf{u} = \ldots = \left(\nabla^2 u, \nabla^2 v, \nabla^2 w\right)$$

Calculus – Laplacian operator

- Second-order finite differences

$$u_{xx}(x,y,z) = \frac{u(x+h,y,z)+u(x-h,y,z)-2\mathrm{u}(x,y,z)}{h^2}$$

$$v_{yy}(x,y,z) = \frac{u(x,y+h,z)+u(x,y-h,z)-2\mathrm{u}(x,y,z)}{h^2}$$

$$w_{zz}(x,y,z) = \frac{u(x,y,z+h)+u(x,y,z-h)-2\mathrm{u}(x,y,z)}{h^2}$$

- Finite difference of Laplacian operator

$$\nabla^2 u(x,y,z) = u_{xx}(x,y,z)+u_{yy}(x,y,z)+u_{zz}(x,y,z) =$$

$$\frac{u(x+h,y,z)+u(x-h,y,z)+u(x,y+h,z)+u(x,y-h,z)+u(x,y,z+h)+u(x,y,z-h)-6\mathrm{u}(x,y,z)}{h^2}$$

Fluid Dynamics

Motivations

- Dynamics of incompressible fluids is governed by the following Navier-Stokes equations

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

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

- Motivation: We need to understand the **physics** behind !

Nomenclature

- Velocity vector field (**u**)
- Pressure scalar field (p)
- Density of fluid (ρ)
- Viscosity of fluid (υ)
- External force field (**F**)

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

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

Navier-Stokes Equations

- Set of two Partial differential equations
- Continuity Equation – The rate at which mass enters a system is equal to the rate at which mass leaves the system.

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

- Momentum equation – Application of Newton's second law to fluid motion

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

Continutity Equation

- Total mass must be always conserved.
- The rate at which mass enters a system is equal to the rate at which mass leaves the system.
- The divergence of the velocity field must always be zero

$$\mathbf{u} = (u, v, w)$$

$$\nabla \circ \mathbf{u} = u_x + u_y + u_z = \mathbf{0}$$

Momentum Equation

- Velocity field of fluid changes over time due to:

$$\boxed{\frac{\partial \mathbf{u}}{\partial t} =}$$

Momentum Equation

- Velocity field of fluid changes over time due to:

- **Self advection force**

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

Momentum Equation

- Velocity field of fluid changes over time due to:

- Self advection force
- **Pressure gradient force**

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \circ \nabla)\mathbf{u} \boxed{- \frac{1}{\rho}\nabla p}$$

Momentum Equation

- Velocity field of fluid changes over time due to:

- Self advection force
- Pressure gradient force
- **Internal viscosity force**

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \circ \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p \boxed{+ \upsilon \nabla^2 \mathbf{u}}$$

Momentum Equation

- Velocity field of fluid changes over time due to:

- Self advection force
- Pressure gradient force
- Internal viscosity force
- **External body forces**

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \circ \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \upsilon \nabla^2 \mathbf{u} \boxed{+ \ \mathbf{F}}$$

Time Derivative of Velocity

- At every location velocity field of fluid changes due to several internal and external forces acting on fluids body
- It's time derivative simple measures the evaluation of the velocity field in time

$$\boxed{\frac{\partial \mathbf{u}}{\partial t} =}$$

Advection Term

- Advection term represents internal rate of change of momentum due to velocity itself. To conserve momentum it must moved (self advected) through the space along with the fluid
- Mathematically advection is the scaled velocity by it's divergence

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

Pressure term

- Pressure term defines internal forces generated due to the pressure differences within the fluid
- For incompressible fluid pressure will be directly coupled with conservation of mass (continuity equation)

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \circ \nabla)\mathbf{u} \boxed{- \frac{1}{\rho}\nabla p}$$

Viscosity term

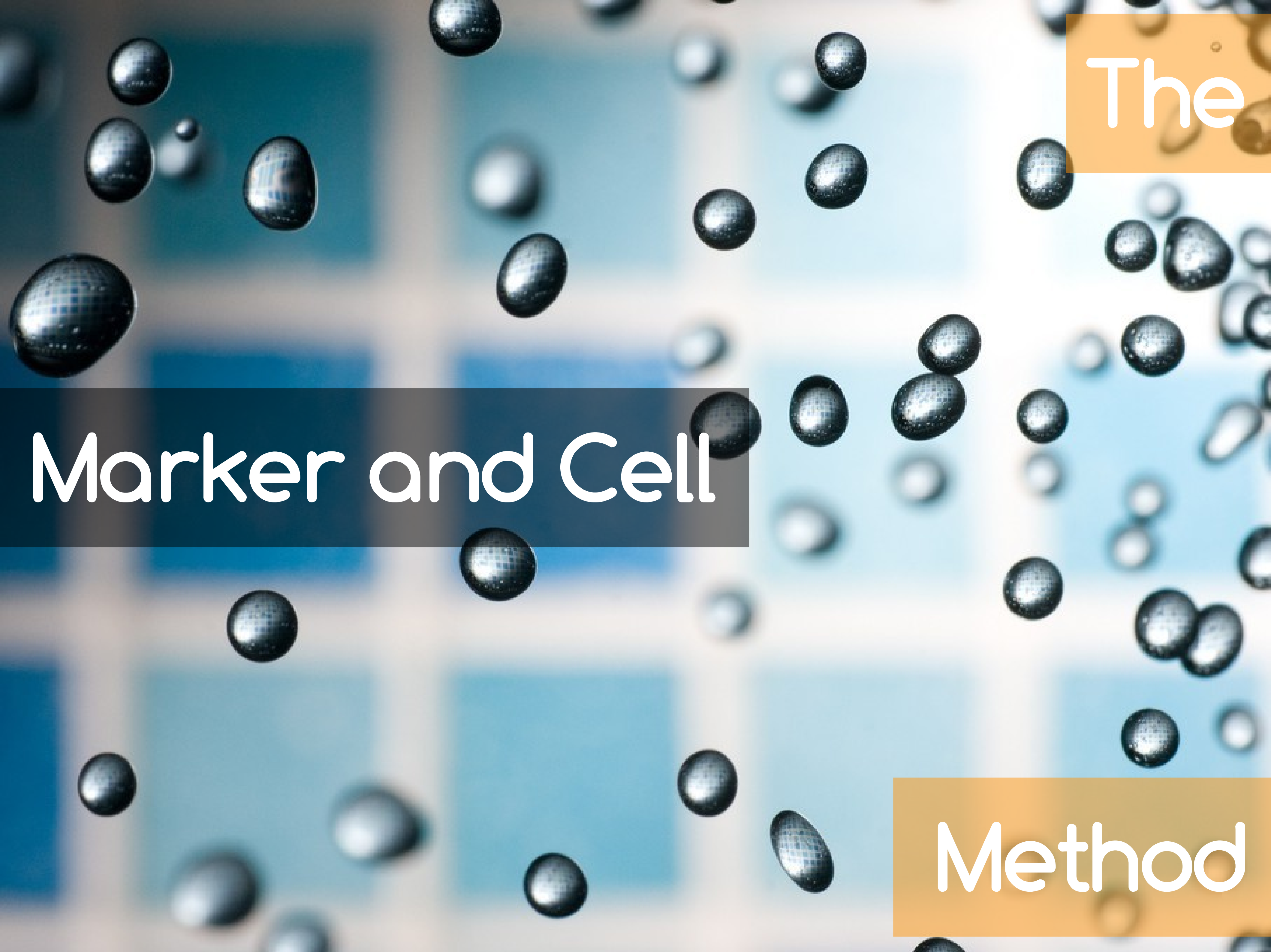
- Viscosity term captures internal friction forces due to material friction.
- Viscosity forces cause the velocity of fluid to move toward the neighbor average, see the Laplacian operator

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \circ \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p \boxed{+ \upsilon \nabla^2 \mathbf{u}}$$

External forces

- External forces usually contain gravity, wind, user drag, contact forces or any other body forces.
- Simply we can modify the velocity field by any external force while keeping natural motion of fluid

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \circ \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \upsilon \nabla^2 \mathbf{u} \boxed{+ \mathbf{F}}$$


The
Marker and Cell
Method

Fluid simulation techniques

- Eulerian techniques
  - Marker and Cell (MAC)
  - Lattice Boltzmann Model (LBM)
  - Other Finite Element/Difference Methods (FEM/FDM)

- Lagrangian techniques
  - Smoothed Particle Hydrodynamics (SPH)
  - Fluid Implicit Particle (FLIP)
  - Particle in Cell (PIC)
  - Moving Particle Semi Implicit (MPS)

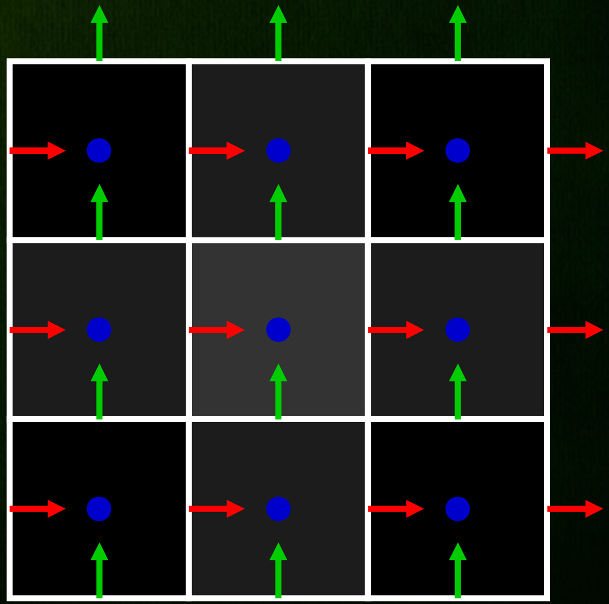
Marker and Cell (MAC) Simulation

- Popular Eulerian fluid simulation technique in CG
- Originally invented by Harlow and Welch in 1965

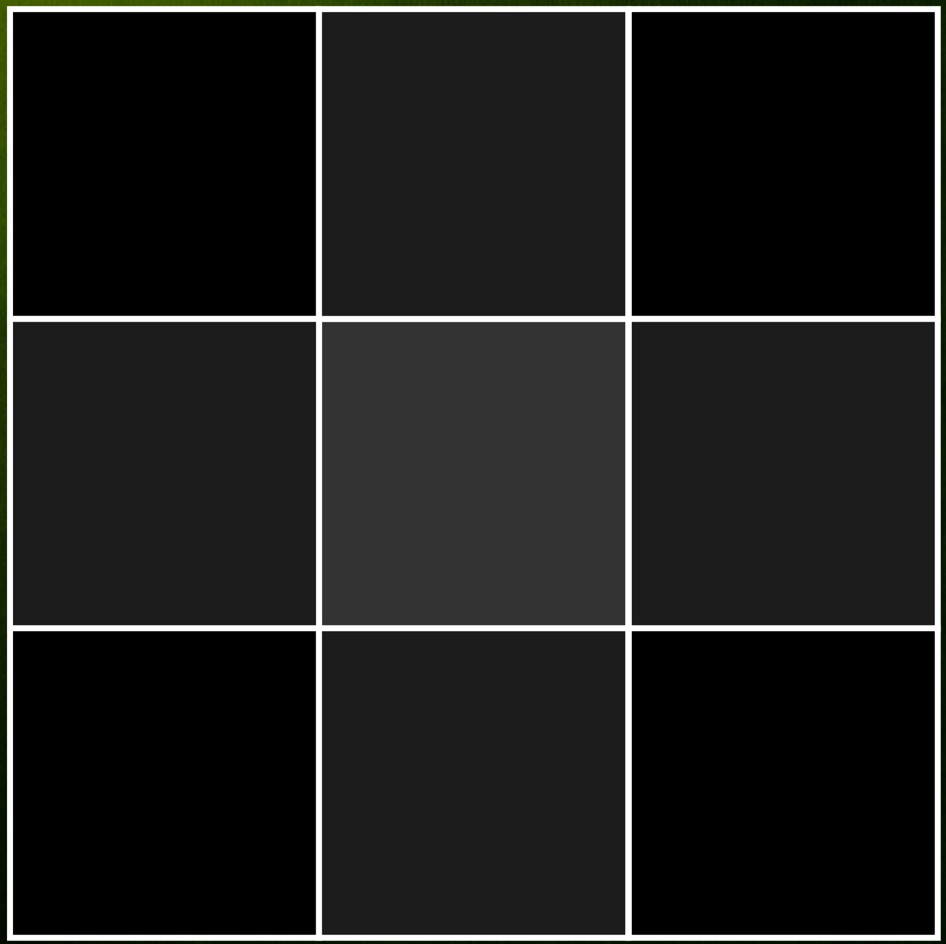
- Key ideas
  - Discretize simulation space into cubical grid
  - Store fluid variables in a staggered fashion
  - Numerically evolve Navies Stokes eq. on grid in time
  - Advect mass-less marker particles in velocity field
  - Update type (solid, fluid, empty) of cells according to the location of marker particles

Staggered MAC grid

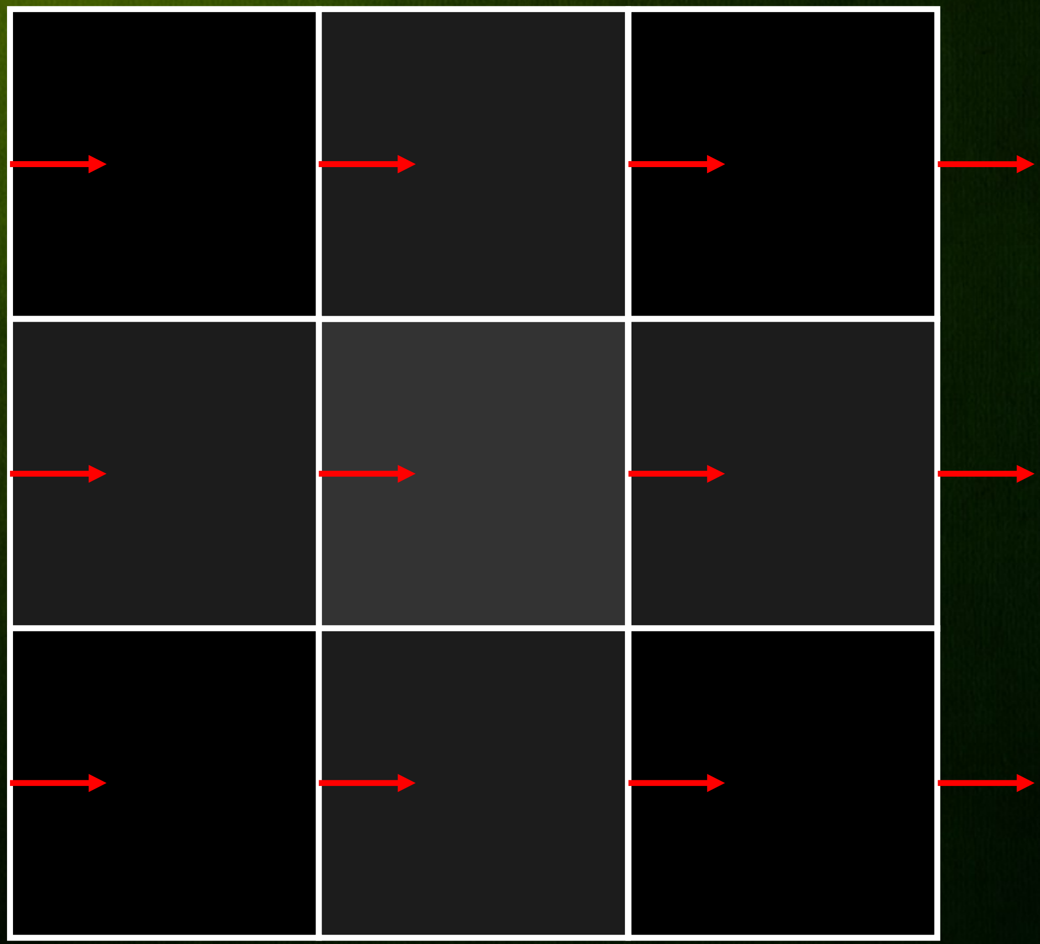
- Virtually decompose velocity vector field **u** into three respective scalar fields (u,v,w)
- Store each velocity component on face center of grid cell parallel to face normal
- In 2D - Vertical faces store horizontal component and vice versa
- Store pressure in the center of grid cell



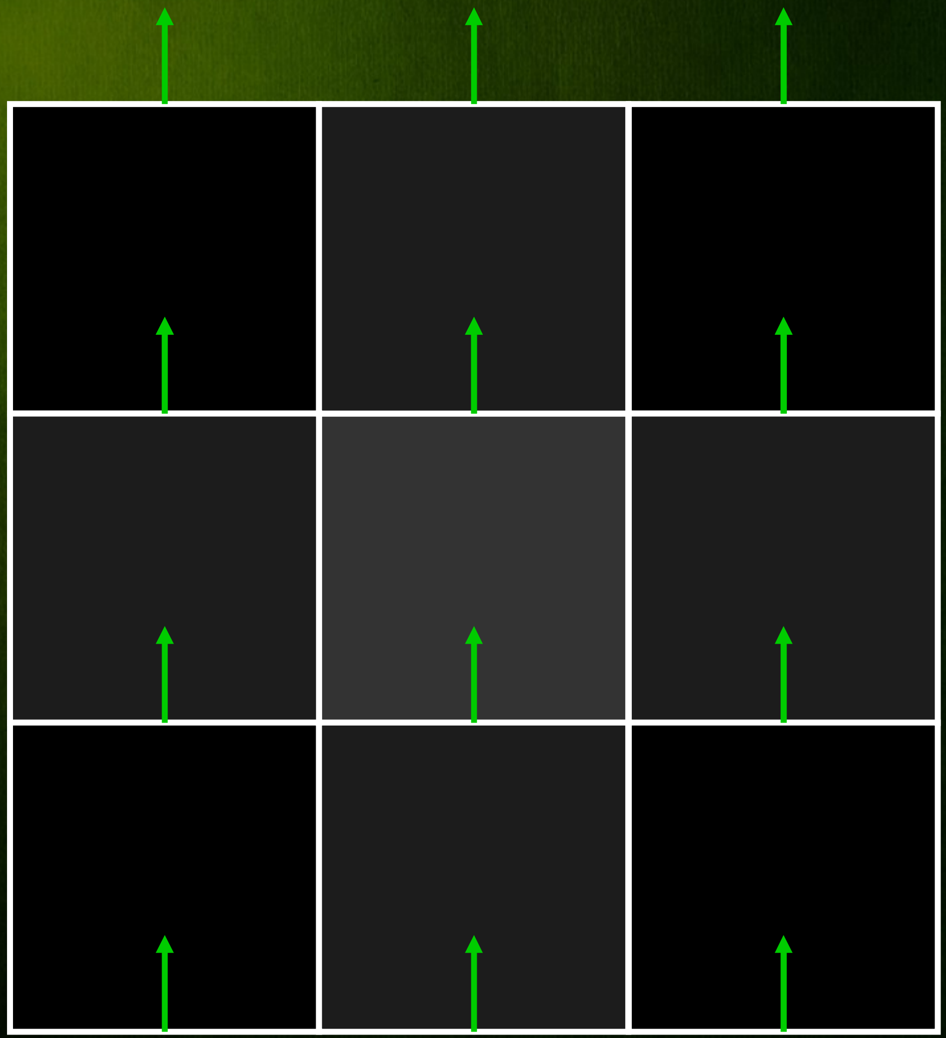
MAC Grid: Cells



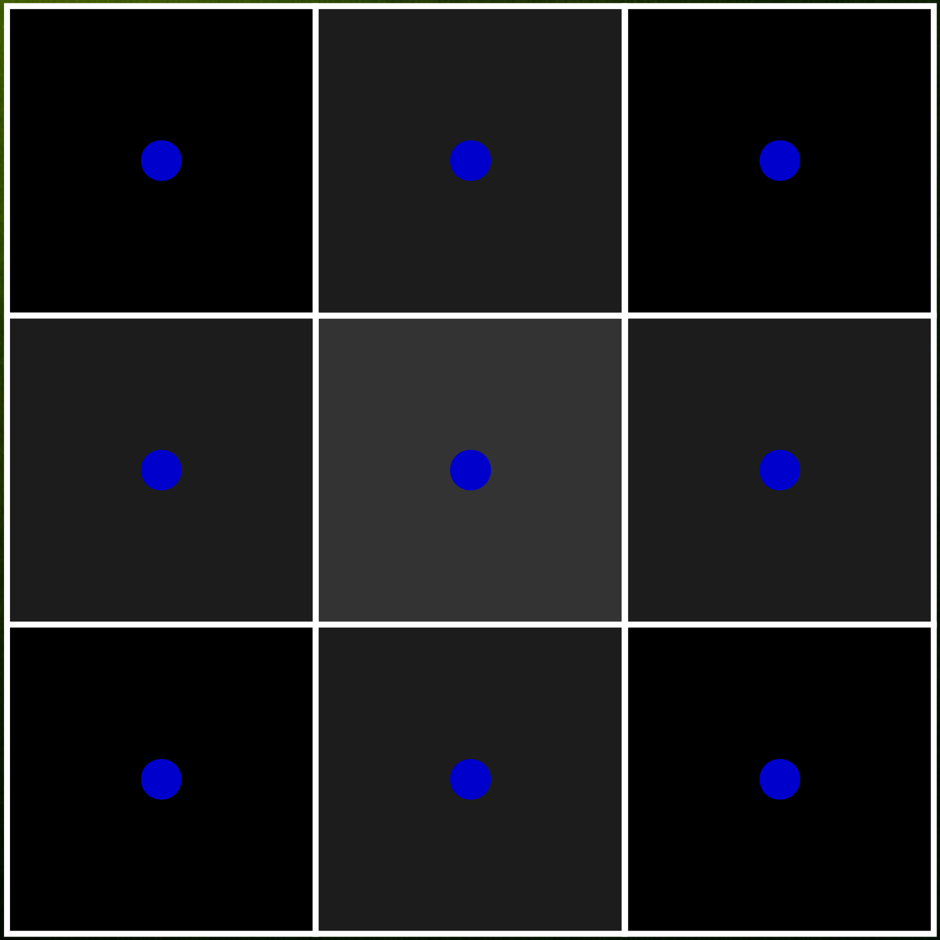
MAC Grid: u-velocity



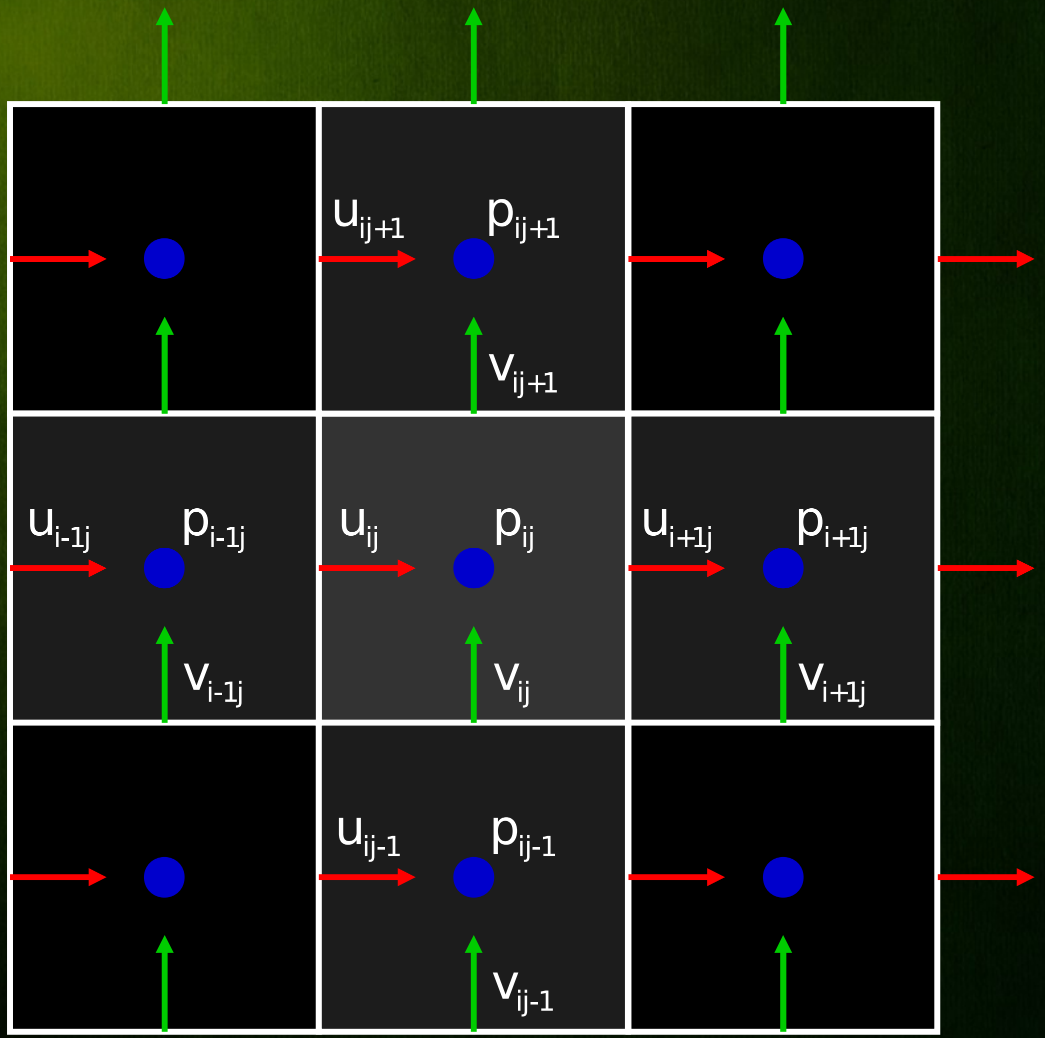
MAC Grid: v-velocity



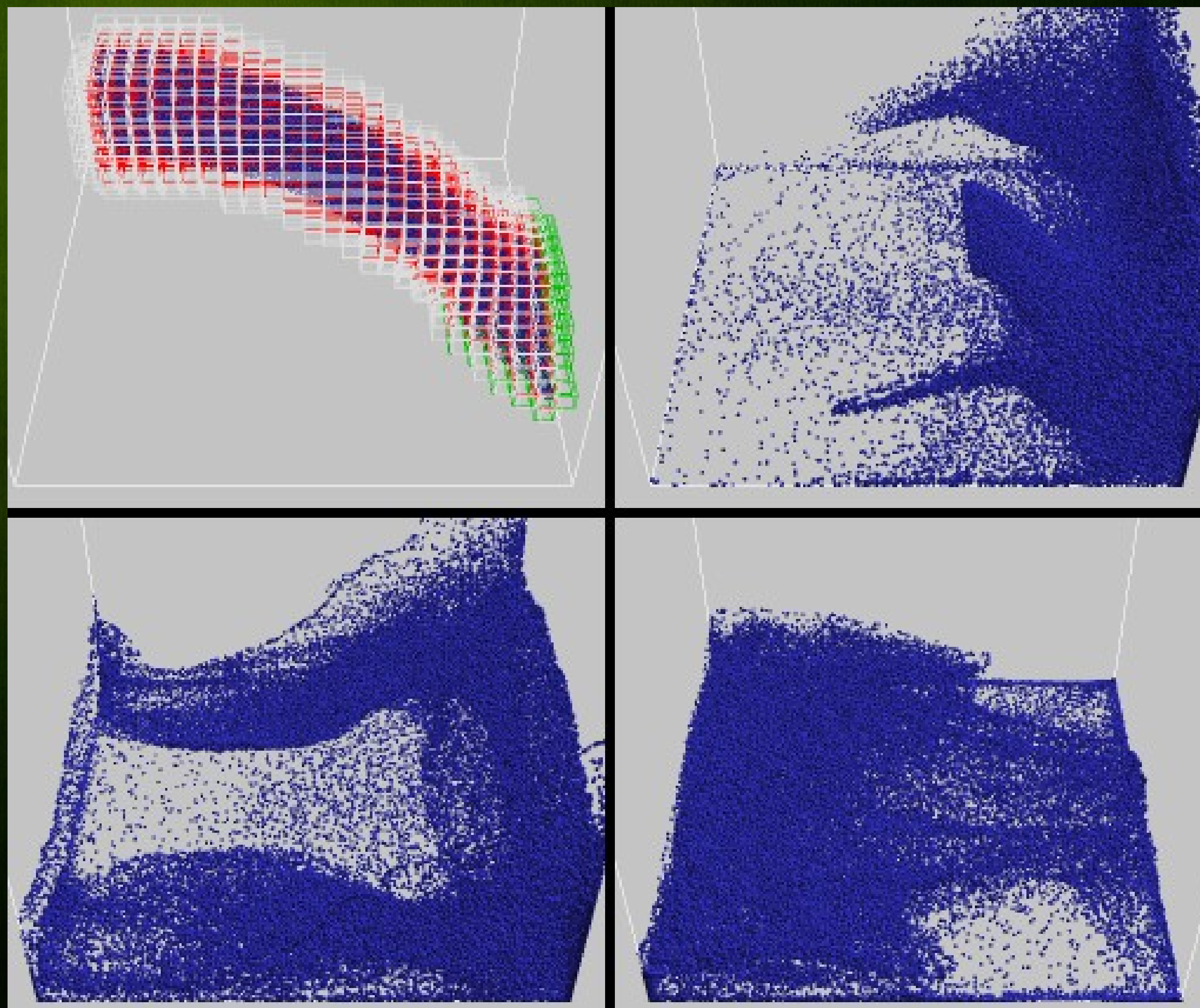
MAC Grid: pressure



Staggered MAC Grid



MAC Simulation



Stable MAC Algorithm

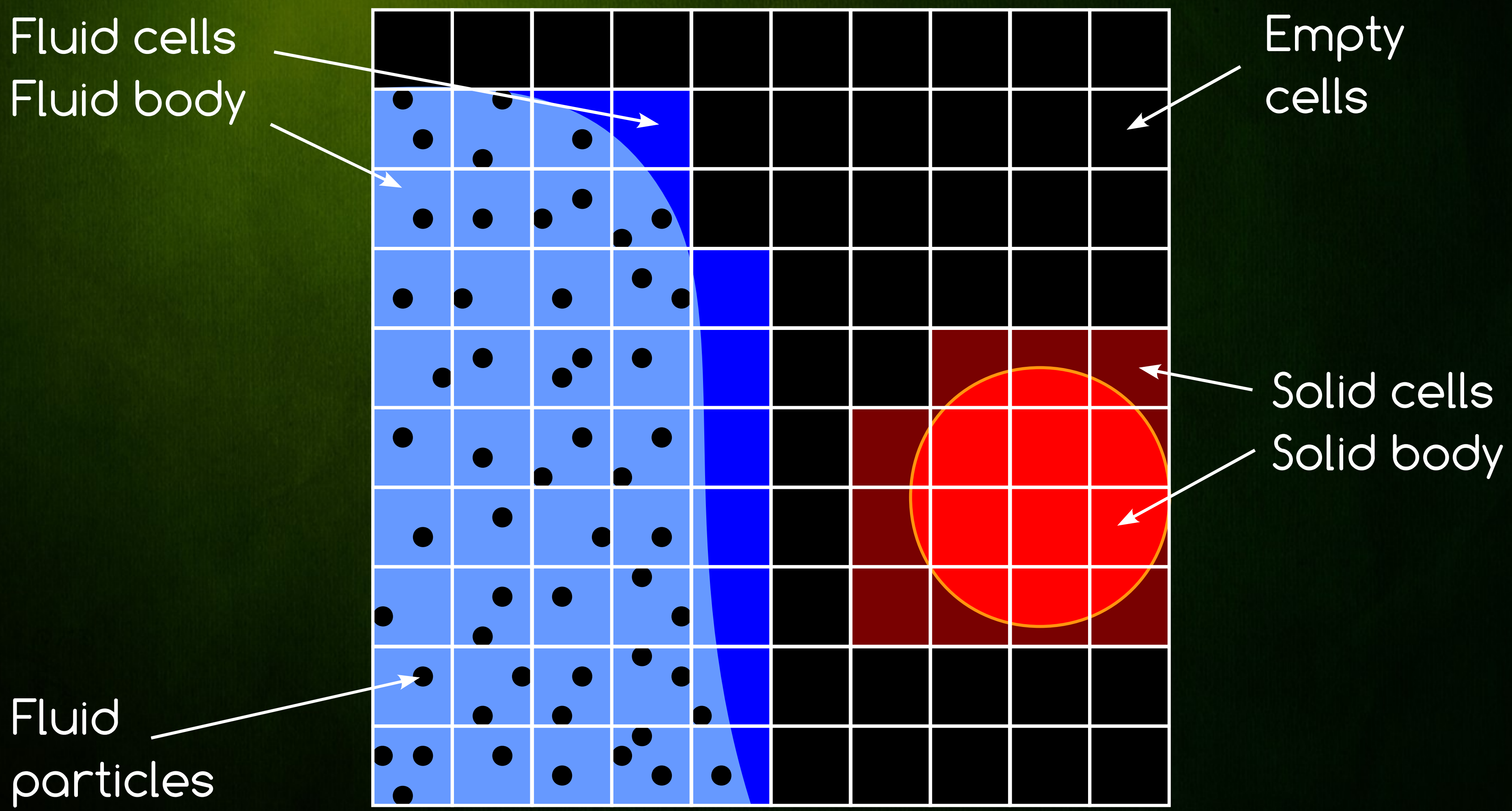
- Initialization
  - Grid initialization
  - Particle seeding
- Simulation loop
  - Time step estimation
  - Particle advection
  - Grid update
  - Boundary conditions
  - Velocity update

MAC – Initialization

- **Grid Initialization**
- Set all velocities to zero
- Define initial (static) environment
- Label cells as Fluid, Solid or Empty

- **Particle seeding**
- Randomly seed mass-less marker particles inside fluid body

MAC Initialization



MAC Simulation Loop

* Calculate (set) simulation time step Δt
* Advect marker particles along fluid velocity
* Update grid by marker particles
* Apply boundary conditions
* Advance the velocity field **u**

MAC – Time Step Estimation

- We need to achieve enough
- 1) Stability prevent blow up
- 2) Accuracy to simulate plausible

- Use Courant-Friedrichs-Lewy (CFL) condition
  - The CFL condition states that the time step must be small enough to make sure information does not travel across more than one cell at a time.

$$\Delta t < \frac{\Delta x}{\max(|u|, |v|, |w|)}$$

MAC – Particle Advection

- Given velocity field and time step we can freely advect particles using some explicit scheme
- Standard Euler integration step

$$x^{new} = x + \Delta t u(x)$$

- Modified Euler (midpoint method)

$$x^{*} = x + \Delta t u(x)$$
$$x^{new} = x + 0.5\Delta t[u(x) + u(x^{*})]$$

MAC – Grid update

- Particles have new locations
- Cell types must be updated
- Each cell containing at least one particle is marked as fluid cell
- Solid cells are unchanged
- Other cells are marked as empty (air) cells

MAC – Boundary Conditions

- Two types of boundary condition
  - Fluid / Solid boundary conditions
  - Fluid / Air boundary conditions
- We need to satisfy them both for velocity and pressure
- Velocity boundary conditions uses slip-conditions and continuity conditions
- Pressure boundary conditions uses Dirichlet and Neumann conditions (see Pressure calculation)

MAC – Velocity boundary conditions

- Free-slip fluid/solid condition:
- Fluid is freely allowed to slip along the solid/fluid boundary face

- No-slip fluid/solid condition:
- Fluid is not allowed to slip along the solid/fluid boundary face

MAC – Velocity Field Update

- Evaluate velocity with operator splitting in four steps:

- 1) Force - Apply external forces
- 2) Advect - Apply advection
- 3) Diffuse - Apply viscosity
- 4) Project - Calculate and apply pressure

$$u(x, t) = w_0^{\text{force}} \longrightarrow w_1^{\text{advect}} \longrightarrow w_1^{\text{diffuse}} \longrightarrow w_1^{\text{project}} \longrightarrow w_4 = u(x, t+h)$$

MAC – Apply External Forces

- Use simple explicit Euler to integrate force fields
- Force field is usually gravity or wind body force

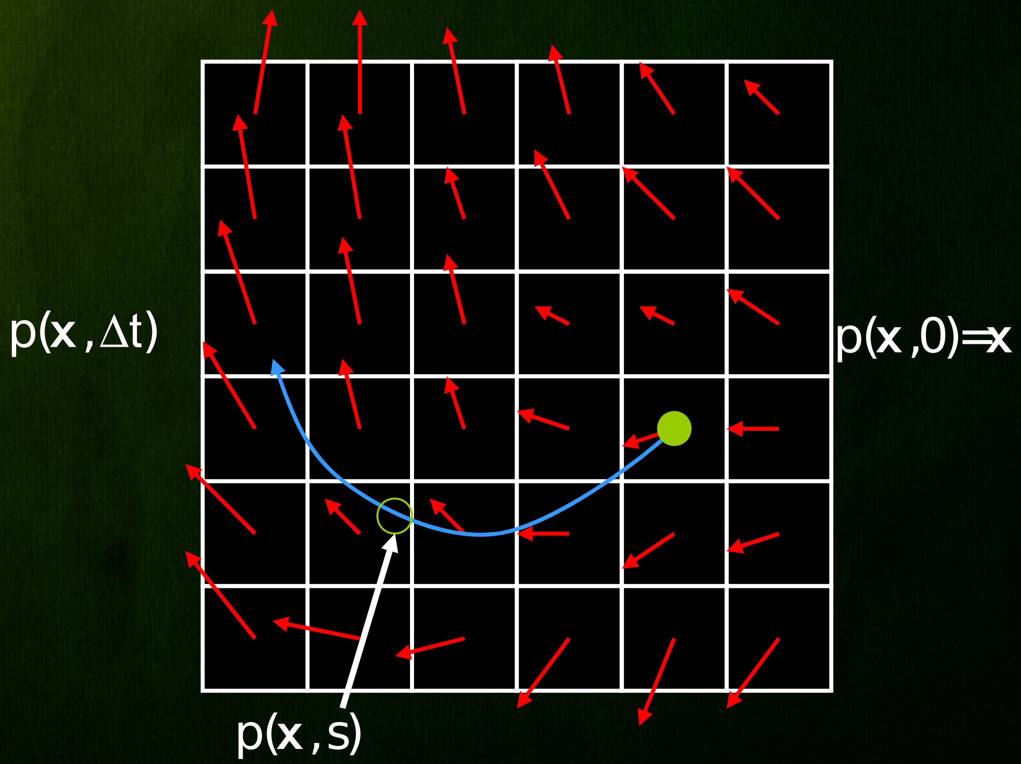
$$w_1(x) = w_0(x) + \Delta t F(x,t)$$

MAC – Apply Velocity Advection

- We want to know how will be the velocity advected over the time step
- Simple Euler scheme brings instability or extremely small time steps must be taken
- Method of characteristics is unconditionally stable, allows large time steps – semi Implicit advection

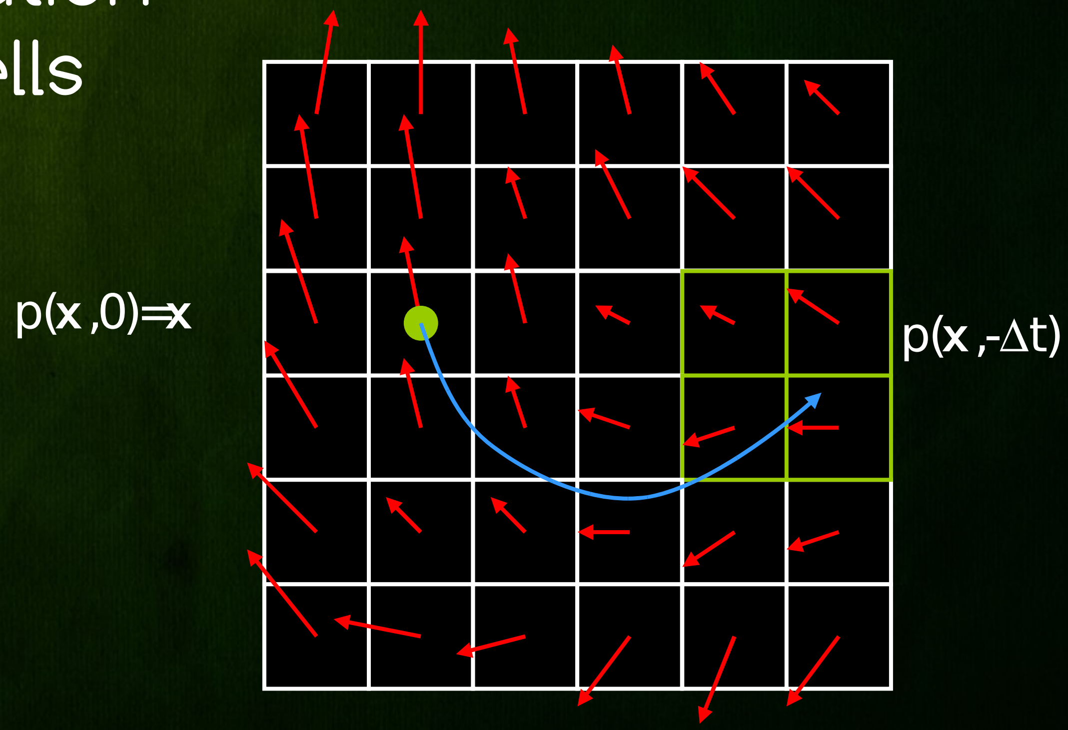
MAC – Semi-implicit Advection

- Suppose simple particle advection
- During time step particle will travel along the blue path in the velocity field and can carry any scalar/vector with it
- Let p(**x**,s) be the location of particle at time s



MAC – Semi-implicit Advection

- Key idea – trace particle in negative velocity and find which velocity will be advected to particles location
- Use bilinear interpolation of values in green cells



MAC – Semi-implicit Advection

- Bilinear interpolation is always bounded, advection is unconditionally stable
- Particle back-tracing must be done separately for each velocity dimension (scalar field)
- If particle tracer is simple Euler with Δt time step semi-implicit advection can be written as

$$w_2(x) = w_1(p(x, -\Delta t))$$

$$w_2(x) = w_1(x - \Delta t w_1(x))$$

MAC – Applying Viscosity

- Explicit and Implicit Euler Scheme

$x(t + \Delta t) = x(t) + \Delta t\, x'(t)$ (Explicit Euler)

$x(t + \Delta t) - \Delta t\, x'(t) = x(t)$ (Implicit Euler)

- Implicit viscosity application (sparse lin. eq. Solver)

$dw_2(x)/dt = \nabla^2 w_2(x)$

$w_3(x) - \Delta t\, \nabla^2 w_3(x) = w_2(x)$

$(I - \Delta t\, \nabla^2)\, w_3(x) = w_2(x)$

$Ax = b$ where $A = (I - \Delta t\, \nabla^2)$ (Sparse system)

MAC – Calculating Pressure

- For solving pressure we use implicit Euler and continuity condition

$dw_3(x)/dt = -\nabla p(x)$

$u(x) = w_4(x) = w_3(x) - \Delta t \nabla p(x)$

$0 = \nabla \bullet u = \nabla \bullet w_4(x) = \nabla \bullet w_3(x) - \Delta t \nabla^2 p(x)$

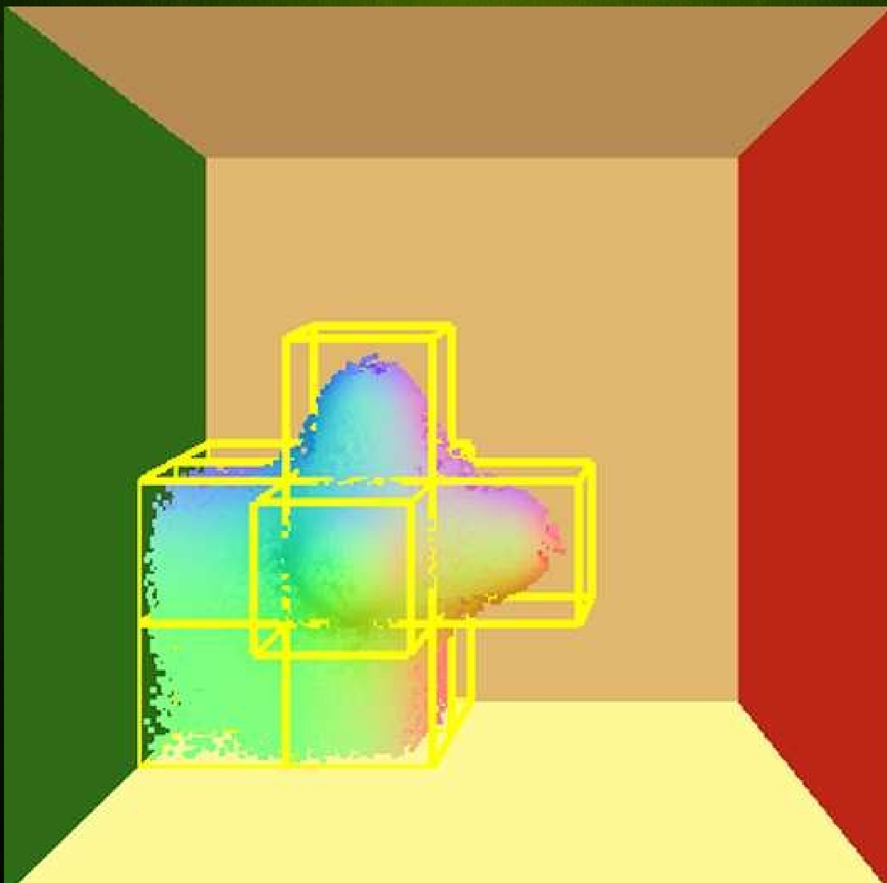
$\nabla^2 p(x) = \nabla \bullet w_3(x)/\Delta t$ (Poisson Equation)

$Ax=b$ where $A=\nabla^2$ (Sparse system)

MAC – Pressure Boundary Conditions

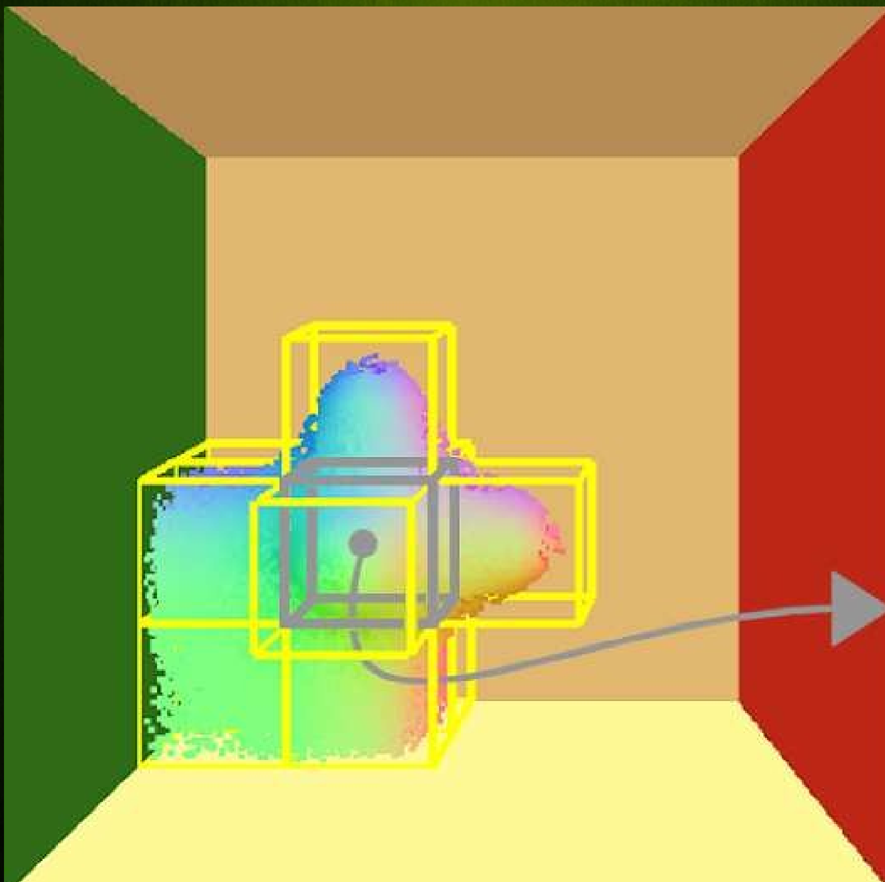
- Neumann boundary condition
  - Set pressure in solid cells equal to fluid pressure in neighbor fluid cell
  - Pressure gradient along boundary face will be zero = Neumann boundary condition
- Dirichlet boundary condition
  - Set pressure in empty (air) cells to zero = Dirichlet boundary condition
- Next slides demonstrate Poisson equation evaluation satisfying Neumann and Dirichlet boundary conditions

MAC – Poisson equation



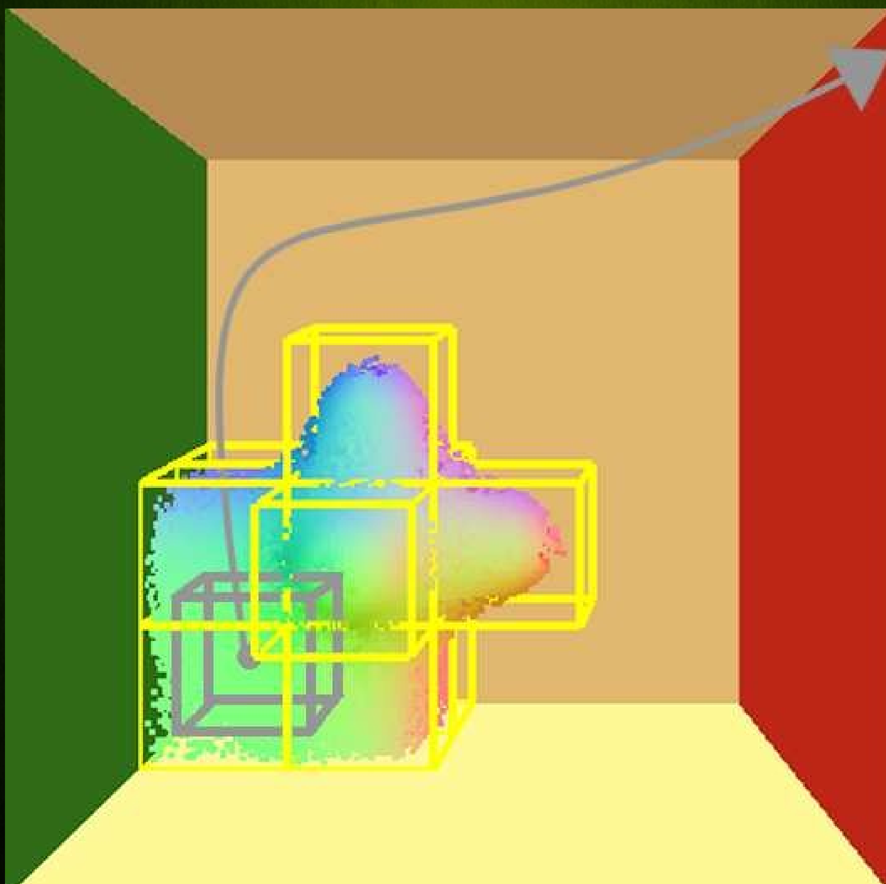
0	3	-1	-1		-1						
1	-1	4		-1		-1					
2	-1		4	-1			-1				
3		-1	-1	5				-1			
4	-1				4	-1	-1				
5		-1			-1	5		-1			
6			-1		-1		5	-1			
7				-1		-1	-1	6	-1	-1	-1
8								-1	6		
9								-1		6	
10								-1			6

MAC – Poisson equation



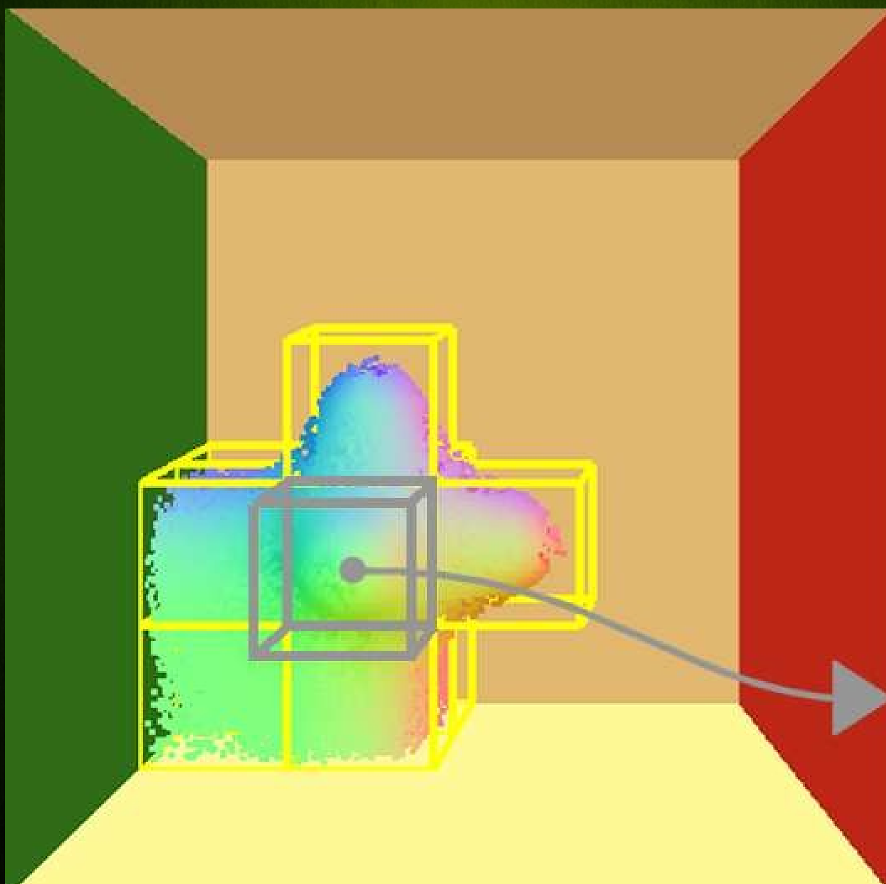
0	3	-1	-1		-1						
1	-1	4		-1		-1					
2	-1		4	-1			-1				
3		-1	-1	5				-1			
4	-1				4	-1	-1				
5		-1			-1	5		-1			
6			-1		-1		5	-1			
7				-1		-1	-1	6	-1	-1	-1
8								-1	6		
9								-1		6	
10								-1			6

MAC – Poisson equation



0	3	-1	-1		-1						
1	-1	4		-1		-1					
2	-1		4	-1			-1				
3		-1	-1	5				-1			
4	-1				4	-1	-1				
5		-1			-1	5		-1			
6			-1		-1		5	-1			
7				-1		-1	-1	6	-1	-1	-1
8								-1	6		
9								-1		6	
10								-1			6

MAC – Poisson equation



0	3	-1	-1		-1						
1	-1	4		-1		-1					
2	-1		4	-1			-1				
3		-1	-1	5				-1			
4	-1				4	-1	-1				
5		-1			-1	5		-1			
6			-1		-1		5	-1			
7				-1		-1	-1	6	-1	-1	-1
8								-1	6		
9								-1		6	
10								-1			6

MAC – Applying Pressure

- Once the pressure is known we use explicit Euler to find new velocity

$$dw_3(x)/dt = -\nabla p(x)$$

$$u(x) = w_4(x) = w_3(x) - \Delta t \nabla p(x)$$

Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics

- Historical origin
  - Invented by Monaghan and Lucy in astrophysics for Simulating flow of interstellar gas
- Classification
  - Lagrangian mesh-less particle-based
  - Based on local integral function representation (convolution)
- Principles
  - Represent fluid with finite number of particles
  - Store all quantities only on particle positions only
  - Approximate field quantities by kernel convolution
  - Use Lagrangian formulation of Navies-Stokes equations for particle dynamics

SPH – Method Overview

- Benefits
  - Mesh-less (grid-less) particle-based
  - No advection term in Navier Stokes equations
  - Inherently mass conserving (finite number of particles)
  - Straightforward multiphase extension
  - Spatially unlimited simulation domain
  - Suitable for interactive applications
- Drawbacks
  - Difficult to achieve incompressible fluid
  - Time consuming Neighbor search algorithm
  - Boundary deficiency (e.g. in density estimation)

SPH – Approximation Principle

- Assume the following notation:
- $A(r)$ – Scalar (or vector) field, $A_i = A(r_i)$
- $\delta(r)$ – Dirac delta function
- $W_h(r)$ – Radial symmetric smoothing kernel
- r_i – Position of i-th particle
- V_i – Volume of i-th particle

SPH – Approximation Principle

- Integral representation of function

  $A(r) = \int_r A(r')\delta(r - r')dr' = A * \delta$

- Approximation of function by convolution

  $A(r) \approx A * W_h = \int_r A(r')W_h(r - r')dr'$

- Particle-base approximation of function

  $\langle A(r) \rangle = \sum_j V_j A_j W_h(r - r_j) \approx A * W_h \approx A(r)$

SPH – Gradient and Laplacian

- Basic Gradient Approximation Formula (BGAF)

$$\nabla_b(A) = \langle \nabla A(r) \rangle = \sum_j V_j A_j \nabla W_h(r - r_j)$$

- Basic Laplacian Approximation Formula (BLAF)

$$\nabla^2{}_b(A) = \langle \nabla^2 A(r) \rangle = \sum_j V_j A_j \nabla^2 W_h(r - r_j)$$

SPH – Gradient and Laplacian

- Difference Gradient Approximation Formula (DGAF)

$$\nabla_b(A) = (1/\rho)\sum_j V_j \rho_j (A_j - A)\nabla W_h(r - r_j)$$

- Symmetric Gradient Approximation Formula (SGAF)

$$\nabla_s(A) = \rho\sum_j V_j \rho_j (A_j/\rho_j + A/\rho)\nabla W_h(r - r_j)$$

- Zero Laplacian Approximation Formula (ZLAF)

$$\nabla^2_z(A) = \sum_j V_j (A_j - A)\nabla^2 W_h(r - r_j)$$

SPH – Kernel functions: $W_h(r)$

- Basic kernel function properties
  - Compact support
  - Partition of unity
  - Symmetry
  - Limit to delta function

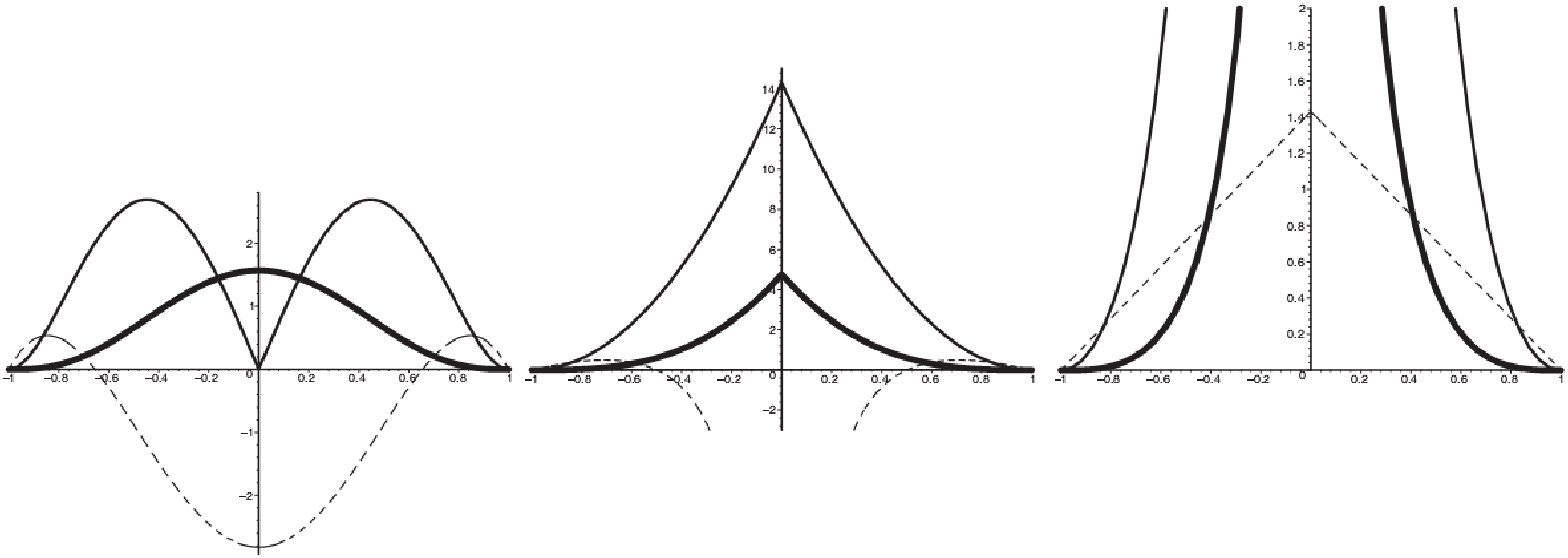
- $|r| \geq h \rightarrow W_h(r) = 0$ (Compact Support)
- $\int_r W_h(r)\,dr = 1$ (Partition of unity)
- $\int_r r W_h(r)\,dr = 0$ (Symmetry)
- $\mathrm{Lim}_{h \rightarrow 0} W_h(r) = \delta(r)$ (Limit to delta function)

SPH – Kernel functions

Kernel function

Kernel function derivative

Kernel function second derivative



W^{poly} W^{press} W^{visco}

SPH – Navier Stokes Equations

- Eulerian formulation

$\partial\rho/\partial t + v \bullet \nabla\rho = -\rho\nabla \bullet v = 0$

$\rho(\partial v/\partial t + v \bullet \nabla v) = -\nabla P + \mu\nabla^2 v + \rho f$

- Lagrangian formulation

$d\rho/dt = \partial\rho/\partial t + v \bullet \nabla\rho = -\rho\nabla \bullet v = 0$

$dv/dt = \partial v/\partial t + v \bullet \nabla v = -\nabla P/\rho + \mu\nabla^2 v/\rho + a =$

$= a^{press} + a^{visco} + a^{ext}$

SPH – Evaluating Fluid Properties

- Density and pressure estimations

$$\rho(r_i) = \langle\rho(r_i)\rangle = \sum_j V_j \rho_j W_h(r - r_j) = \sum_j m_j \rho_j W_h(r - r_j)$$

$$P(r_i) = k^{gas}((\rho_i/\rho_0)^{\gamma} - 1) \qquad \text{(State equation)}$$

- Pressure, viscosity and external forces

$$f^{press}(r_i) = -(m_i/\rho_i)\nabla_s(\rho) = \sum_j m_i m_j (P_j/\rho_j + P_i/\rho_i)\nabla W_h^{press}(r_i - r_j)$$

$$f^{visco}(r_i) = -(m_i/\rho_i)\nabla^2_z(\mu v) = \sum_j V_i V_j (v_j - v_i)\nabla^2 W_h^{visco}(r_i - r_j)$$

$$f^{ext}(r_i) = m_i a_i = f^{int} + f^{grav} + \ldots$$

SPH – Fluid Simulation Algorithm

- Collision Detection
  - Find approximate and precise neighbor particle pairs
  - Find closest points on boundaries
- SPH Dynamics
  - Accumulate densities
  - Calculate pressure
  - Accumulate pressure, viscosity forces and color field
  - Apply surface tension force
  - Apply boundary collision forces
- Time integration (ODE)
  - Use leap-frog to integrate positions and velocities

In: support length h, subdivision factor H and delta time Δt

function $\mathrm{SPH}(h, \Delta t)$

$\textsc{Neighbours} \leftarrow \textsc{ReportAllNeighbors}(h)$

foreach $\mathcal{P}_i$ **in** $\textsc{Particles}$ **do**

$\rho_i \leftarrow 0; \quad \nabla C_i \leftarrow \mathbf{0}; \quad \nabla^2 C_i \leftarrow 0; \quad \mathbf{f}_i \leftarrow \mathbf{f}_i^{\text{grav}}$ `/* initialize */`

foreach $\mathcal{P}_j$ **in** $\textsc{Neighbours}(\mathcal{P}_i)$ **do** `/* accumulate density */`

$\rho_i \leftarrow \rho_i + m_j W_h^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j)$

end

$p_i \leftarrow k^{\text{gas}} \left(\left(\frac{\rho_i}{\rho_0} \right)^{\gamma} - 1 \right)$ `/* calculate pressure */`

foreach $\mathcal{P}_j$ **in** $\textsc{Neighbours}(\mathcal{P}_i)$ **do** `/* accumulate forces */`

$\mathbf{f}_i \leftarrow \mathbf{f}_i - m_i m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_h^{\text{press}}(\mathbf{r}_i - \mathbf{r}_j)$ `/*` $(= \mathbf{f}_i^{\text{press}})$ `*/`

$\mathbf{f}_i \leftarrow \mathbf{f}_i + V_i V_j \mu (v_j - v_i) \nabla^2 W_h^{\text{visco}}(\mathbf{r}_i - \mathbf{r}_j)$ `/*` $(= \mathbf{f}_i^{\text{visco}})$ `*/`

$\nabla C_i \leftarrow \nabla C_i + V_j c_j^{\text{int}} \nabla W_h^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j)$ `/*` $(= \nabla C_i^{\text{int}})$ `*/`

$\nabla^2 C_i \leftarrow \nabla^2 C_i + V_j c_j^{\text{int}} \nabla^2 W_h^{\text{poly}}(\mathbf{r}_i - \mathbf{r}_j)$ `/*` $(= \nabla^2 C_i^{\text{int}})$ `*/`

end

$\mathbf{f}_i \leftarrow \mathbf{f}_i - \sigma^{\text{int}} \nabla^2 C_i^{\text{int}} \frac{\nabla C_i^{\text{int}}}{|\nabla C_i^{\text{int}}|}$ `/*` $(= \mathbf{f}_i^{\text{int}})$ `*/`

end

foreach $\mathcal{P}_i$ **in** $\textsc{Particles}$ **do** `/* Leap-Frog */`

$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \frac{\mathbf{f}_i}{m_i}$

$\mathbf{r}_i \leftarrow \mathbf{r}_i + \Delta t \mathbf{v}_i$

end

end

Neighbor search with Z-indexing

- Neighbor search: Given a particle find all particles whose distance to this particle is less than some threshold (support radius in SPH)
  - This can be $O(n^2)$ problem → very expensive for large number of particles
  - In SPH simulations it is in average case an $O(n)$ problem
- Proposed solution: Z-indexing and radix sort
- Z-indexing: A strategy create a linear index of particles in a 3D grid while maintaining good spatial locality of particles enumerated in index order.
- Radix-sort: $O(n)$ sort for bounded values

Z-indexing : Index order

	x = 0 000	x = 1 001	x = 2 010	x = 3 011	x = 4 100	x = 5 101	x = 6 110	x = 7 111
y = 0 000	000000	000001	000100	000101	010000	010001	010100	010101
y = 1 001	000010	000011	000110	000111	010010	010011	010110	010111
y = 2 010	001000	001001	001100	001101	011000	011001	011100	011101
y = 3 011	001010	001011	001110	001111	011010	011011	011110	011111
y = 4 100	100000	100001	100100	100101	110000	110001	110100	110101
y = 5 101	100010	100011	100110	100111	110010	110011	110110	110111
y = 6 110	101000	101001	101100	101101	111000	111001	111100	111101
y = 7 111	101010	101011	101110	101111	111010	111011	111110	111111

Z-Indexing: Index Structure

- Given (8-bit) coordinates (i,j,k) of some cell
  - $i = i_7 i_6 i_5 i_4 i_3 i_2 i_1 i_0$ (eg 45 = 00101101)
  - $j = j_7 j_6 j_5 j_4 j_3 j_2 j_1 j_0$ (eg 135 = 10000111)
  - $k = k_7 k_6 k_5 k_4 k_3 k_2 k_1 k_0$ (eg 209 = 11010001)
- The interleaved (24-bit) Z-index of cell (i,j,k) is:
  - Index = $k_7 j_7 i_7 k_6 j_6 i_6 k_5 j_5 i_5 k_4 j_4 i_4 k_3 j_3 i_3 k_2 j_2 i_2 k_1 j_1 i_1 k_0 j_0 i_0$
  - Index = 110 100 001 100 001 011 010 111
- We precompute tables i_{24}, j_{24} and k_{24} and get index
- Index = i_{24} **or** j_{24} **or** k_{24} (**or** is bit-wise or operation)
- Tables i_{24}, j_{24} and k_{24} are stored as CUDA textures

Z-Indexing: Index Structure

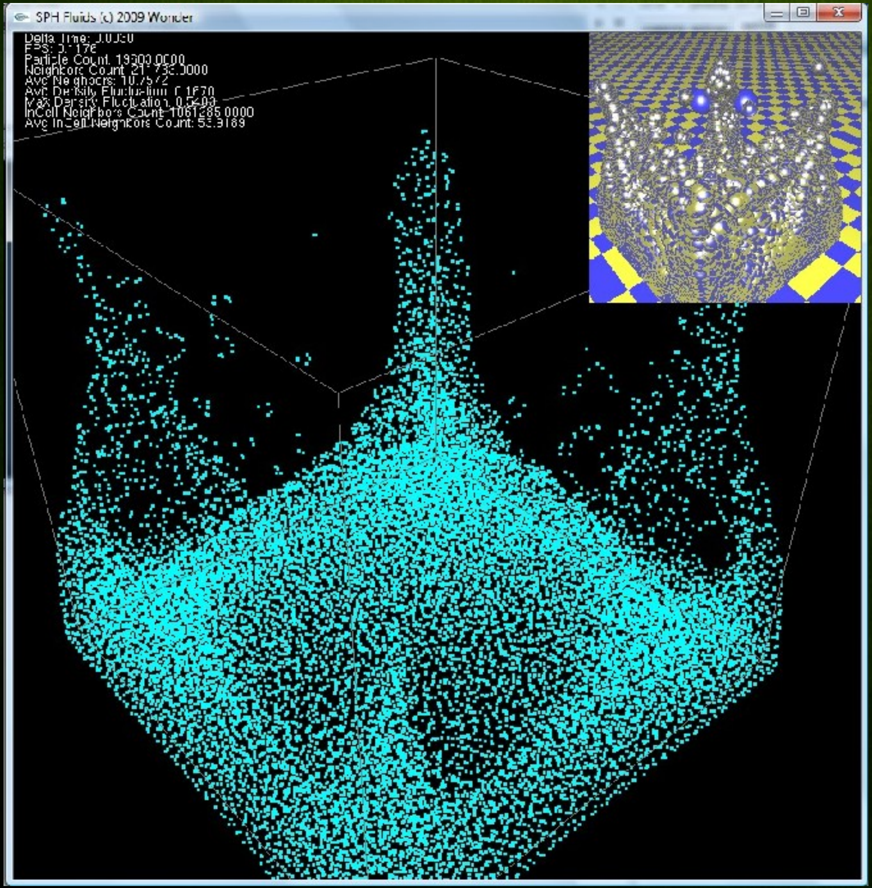
- For each i $(0..2^n)$ precompute i_{24} as
  - i_{24} = $00i_7 00i_6 00i_5 00i_4 00i_3 00i_2 00i_1 00i_0$
  - i_{24} = 000000001000001001000001
- For each j $(0..2^n)$ precompute j_{24} as
  - j_{24} = $0j_7 00j_6 00j_5 00j_4 00j_3 00j_2 00j_1 00j_0 0$
  - j_{24} = 010000000000000010010010
- For each k $(0..2^n)$ precompute k_{24} as
  - k_{24} = $k_7 00k_6 00k_5 00k_4 00k_3 00k_2 00k_1 00k_0 00$
  - k_{24} = 100100001000000000000100

Z-Indexing: Summary

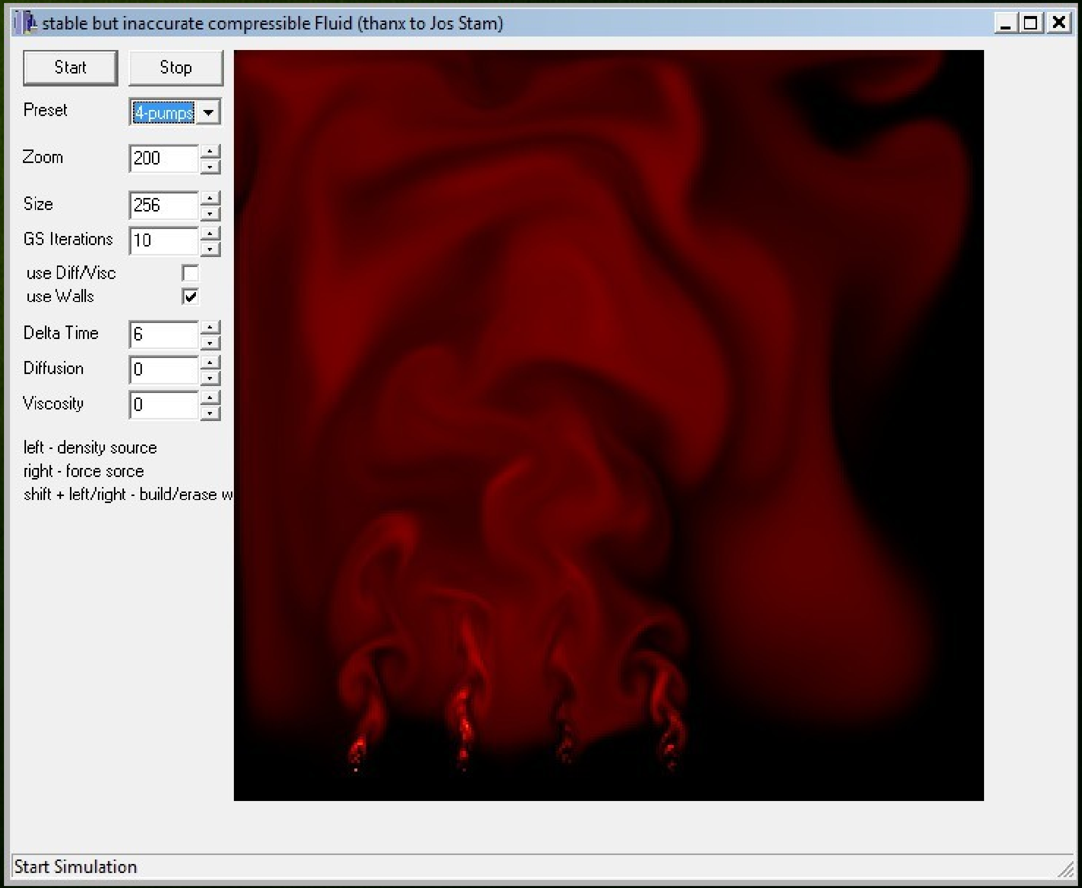
- The simulation domain is divided into a virtual indexing grid
- Grid location of a particle is used to determine its bit-interleaved Z-index
- The Z-indices are computed very efficiently in parallel using a table look-up approach and binary "or"
- Z-indices of particles being within some 2^n spatial block are contiguous
- Before NB particles are sorted based on Z-indices using parallel CUDA radix-sort

Demos / Tools / Libs

- SPH water demo



- MAC fire/smoke demo



... fire and smoke next time :) ...

The End

... endless torture is over ...