CS**5643**

15 Hybrid Eulerian/Lagrangian Fluids

Steve Marschner
Cornell University
Spring 2025

Fluid solver landscape

Grid based Eulerian solvers

- incompressibility is readily achieved by solving global systems for pressure
- advection is tricky; stable methods introduce dissipation
- free surfaces require complex additional mechanisms

Particle fluids (SPH)

- advection is easy: just move particles
- free surfaces are simpler to handle by simply letting particles not fill space
- incompressibility is difficult to achieve
  - allowing compressibility creates stiffness that slows things down
  - enforcing incompressibility as a constraint is tricky to do well

Hybrid simulation

Also a quite early idea

- Particle-in-Cell (PIC) [Harlow 1963] and marker-and-cell (MAC) [Harlow & Welch 1965]
- Fluid Implicit Particle (FLIP) developed in 80s/90s

Strategy: resample onto a grid

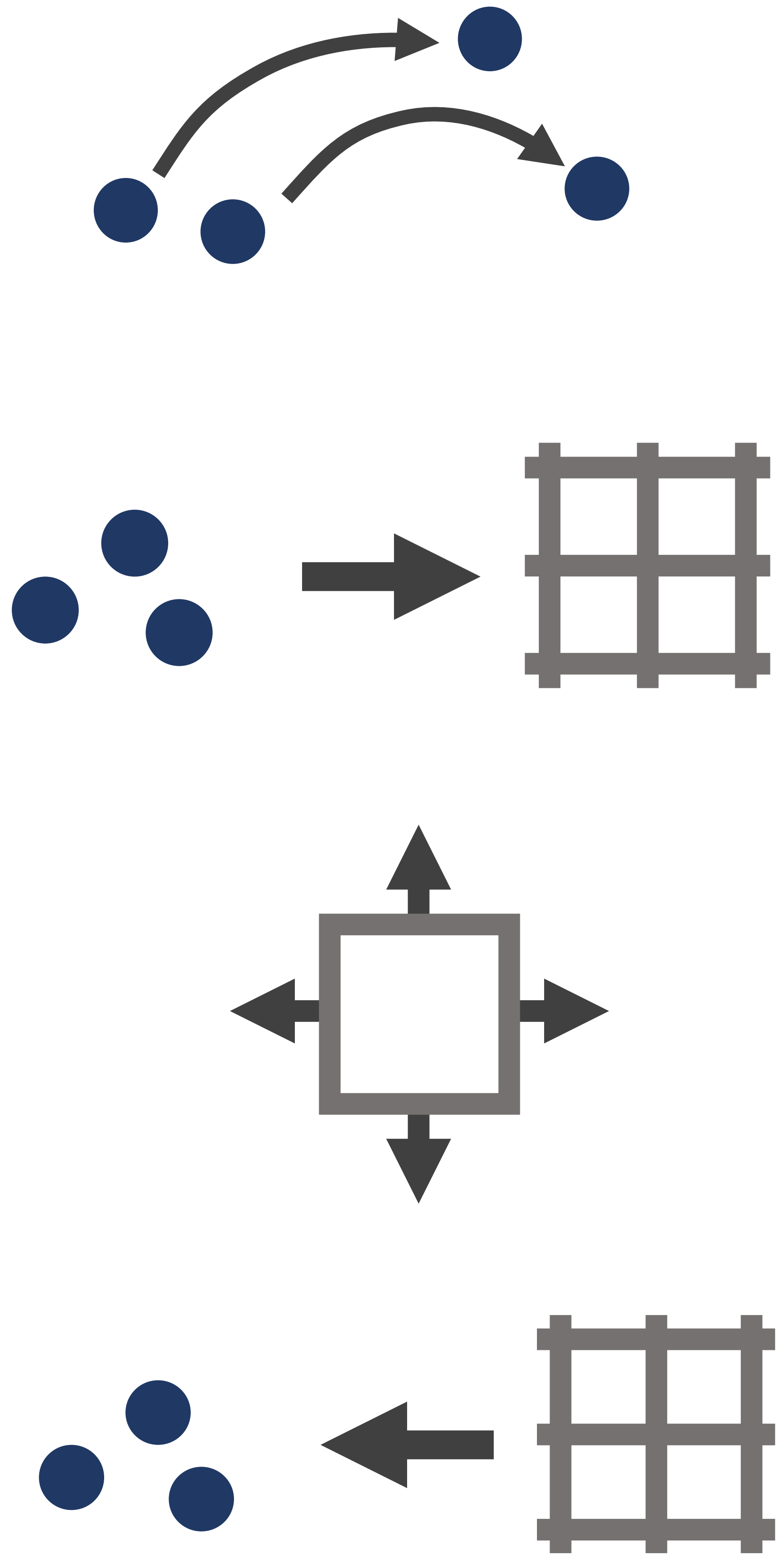
- Recall particles are sample points for the velocity field
- They are an inconvenient grid for doing velocity/pressure constraint solving
- But a signal can always be resampled…

Structure of a timestep

- Simulate particle motion under gravity using their own velocities
- Resample velocity onto a grid (“particle to grid”)
- Compute effects of pressure on the grid, resulting in new velocities
- Resample velocity back to particles (“grid to particle”)

PIC Method

- **Simulate particles**
- **Velocity transfer: Particles → Grid**
- **Make the grid velocities incompressible**
- **Velocity transfer: Grid → Particles**



- Particles carry velocity → can skip grid advection!

Slide from Matthias Müller, Ten Minute Physics

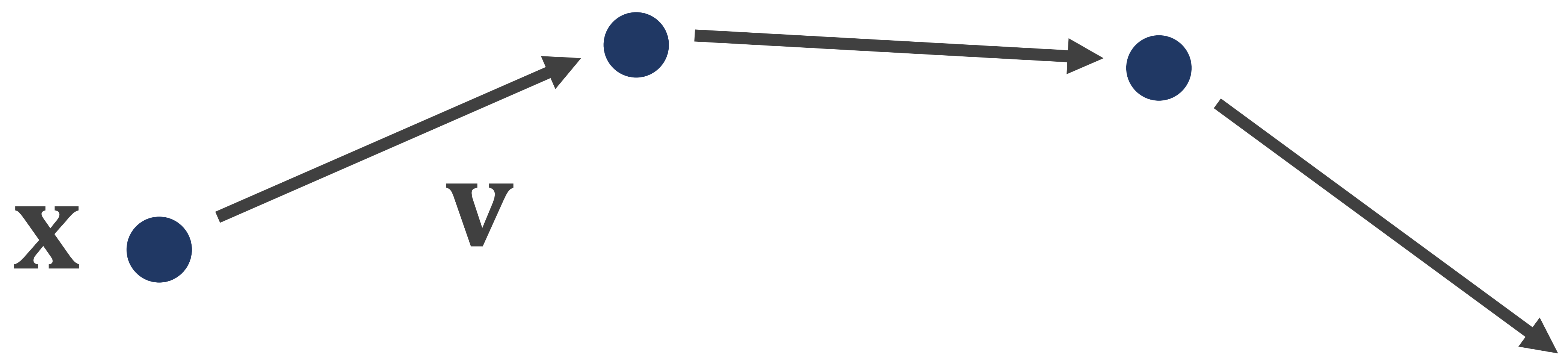
Simulate Particles

- Particles store a position $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ and a velocity $\mathbf{v} = \begin{bmatrix} u \\ v \end{bmatrix}$

for all particles i

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \cdot \mathbf{g}$$

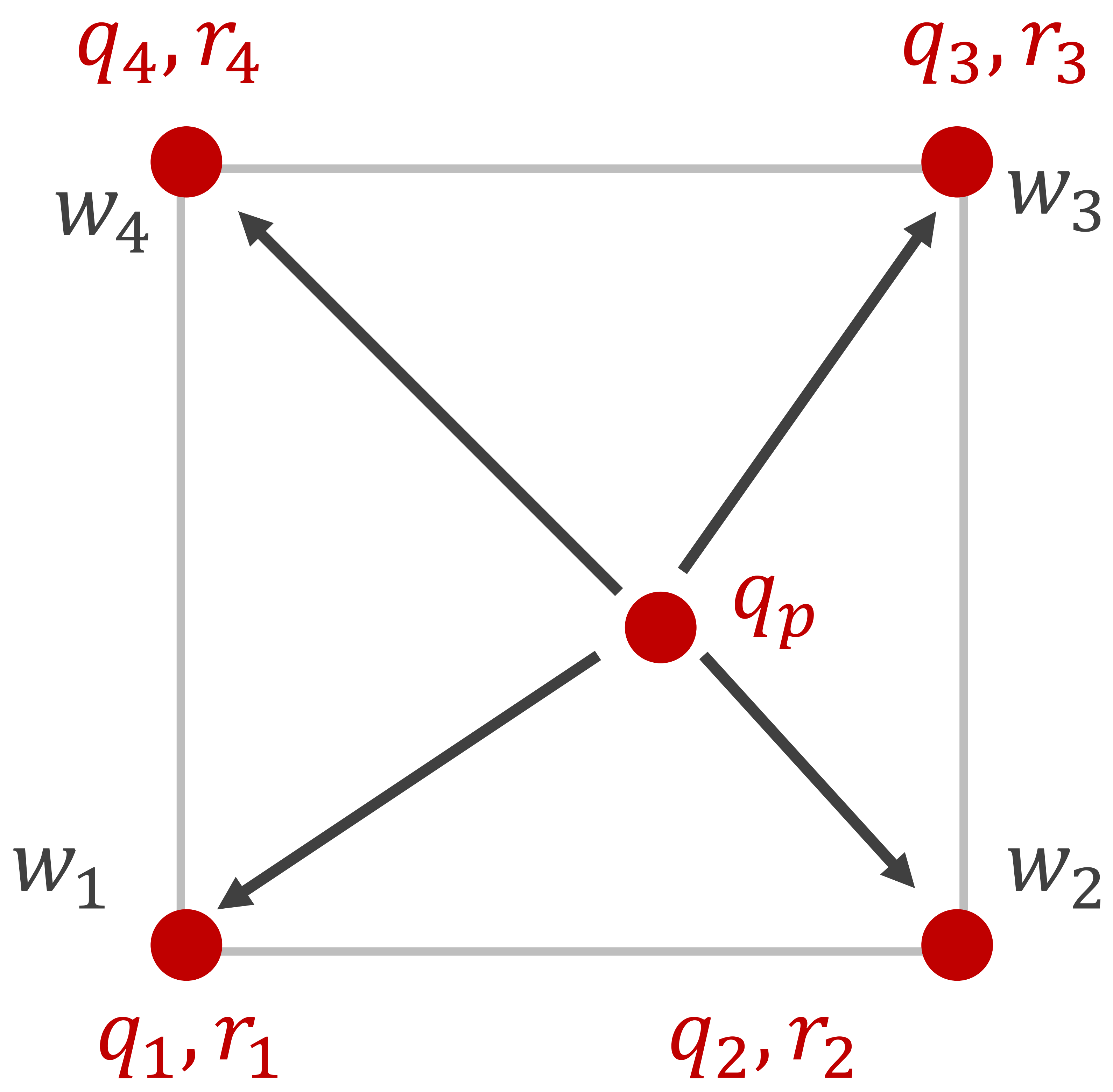
$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \cdot \mathbf{v}_i$$



- Push particles out of obstacles!

- Gravity $\mathbf{g} \approx \begin{bmatrix} 0 \\ -9.81 \end{bmatrix} \frac{m}{s^2}$
- Timestep Δt (e. g. $\frac{1}{30} s$)

From Particles to Grid



clear q and r of all cells

for all particles

$q_1 \leftarrow q_1 + w_1 q_p \qquad r_1 \leftarrow r_1 + w_1$

$q_2 \leftarrow q_2 + w_2 q_p \qquad r_2 \leftarrow r_2 + w_2$

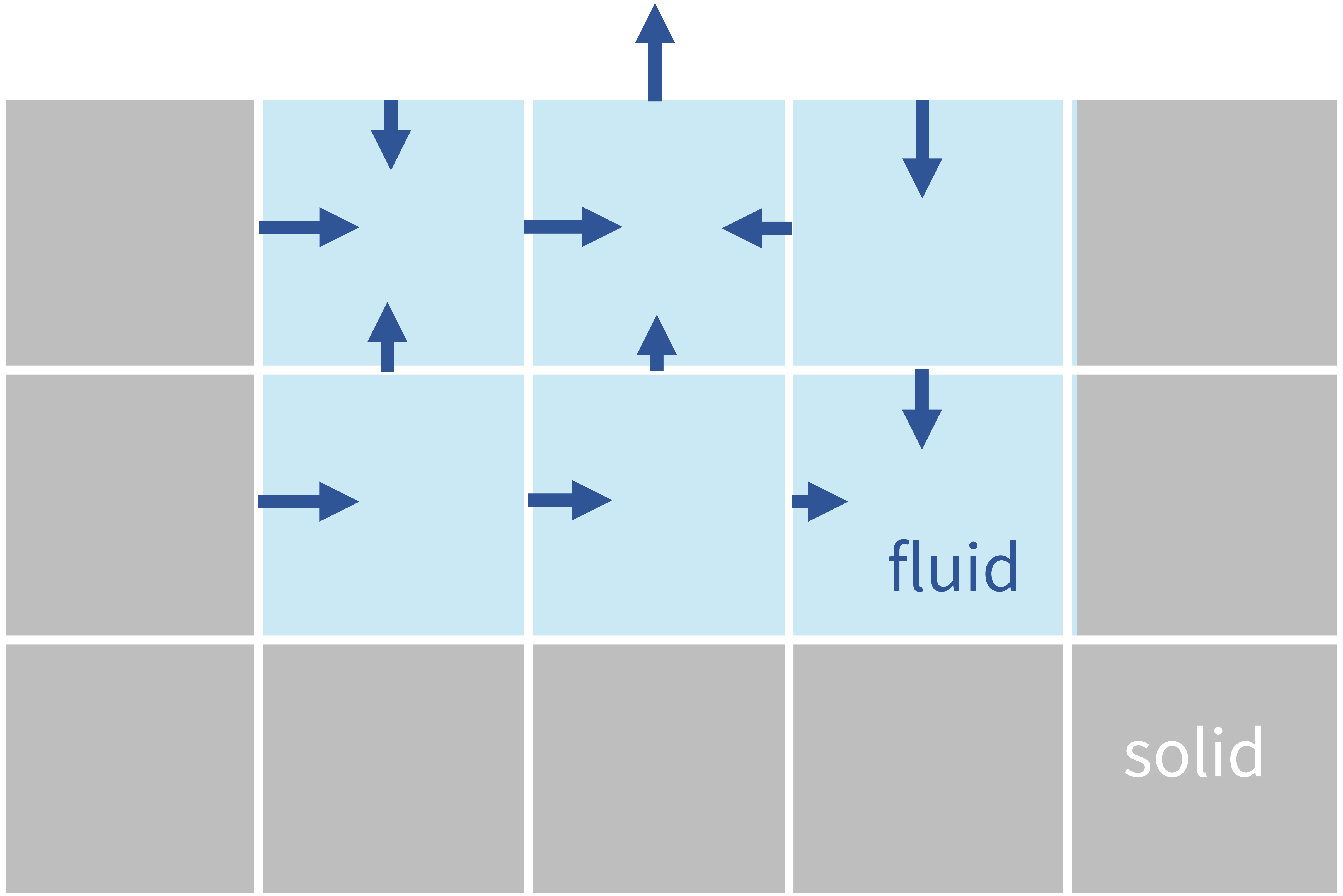
$q_3 \leftarrow q_3 + w_3 q_p \qquad r_3 \leftarrow r_3 + w_3$

$q_4 \leftarrow q_4 + w_4 q_p \qquad r_4 \leftarrow r_4 + w_4$

for all cells

$q \leftarrow q/r$

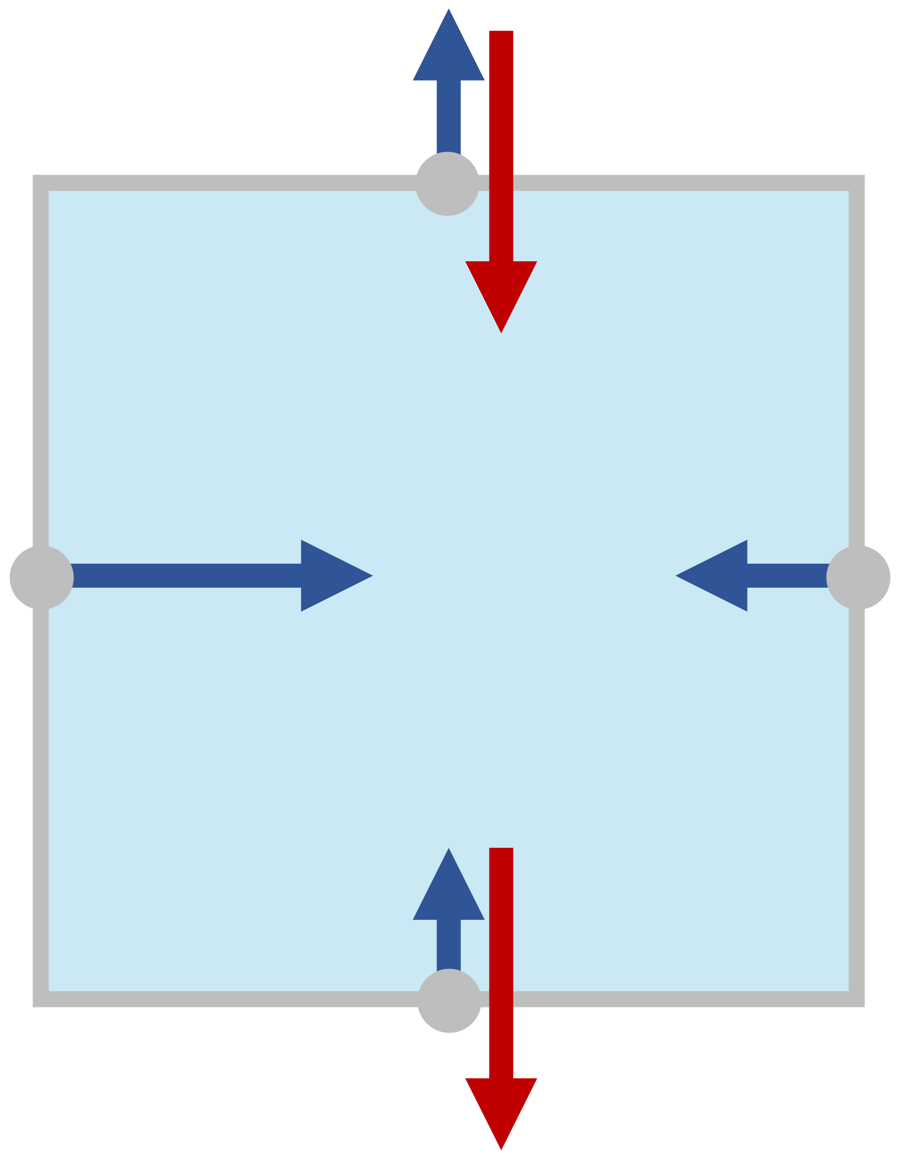
Eulerian Simulation Recap



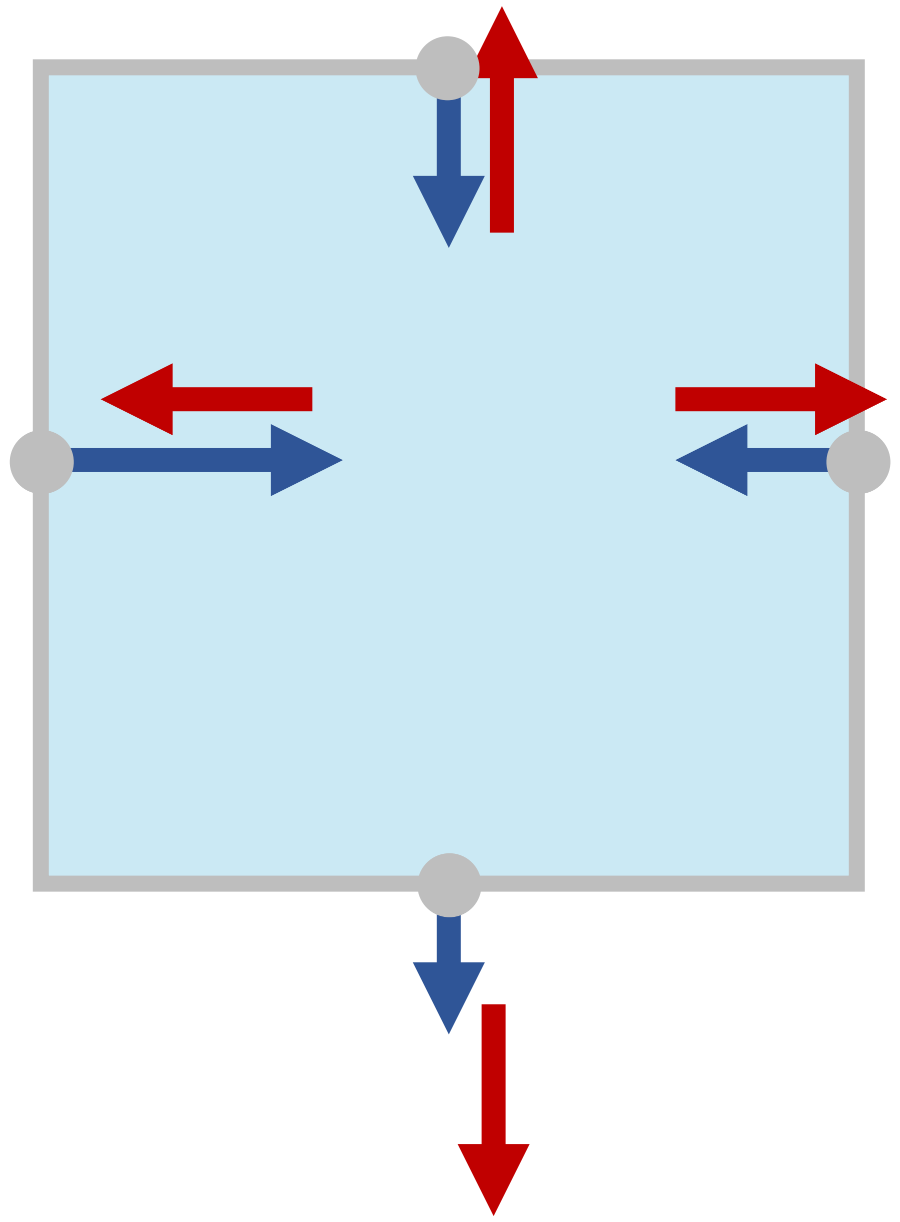
- Fluid as a velocity field stored in a staggered grid
- Two types of cells: fluid and solid

Slide from Matthias Müller, Ten Minute Physics

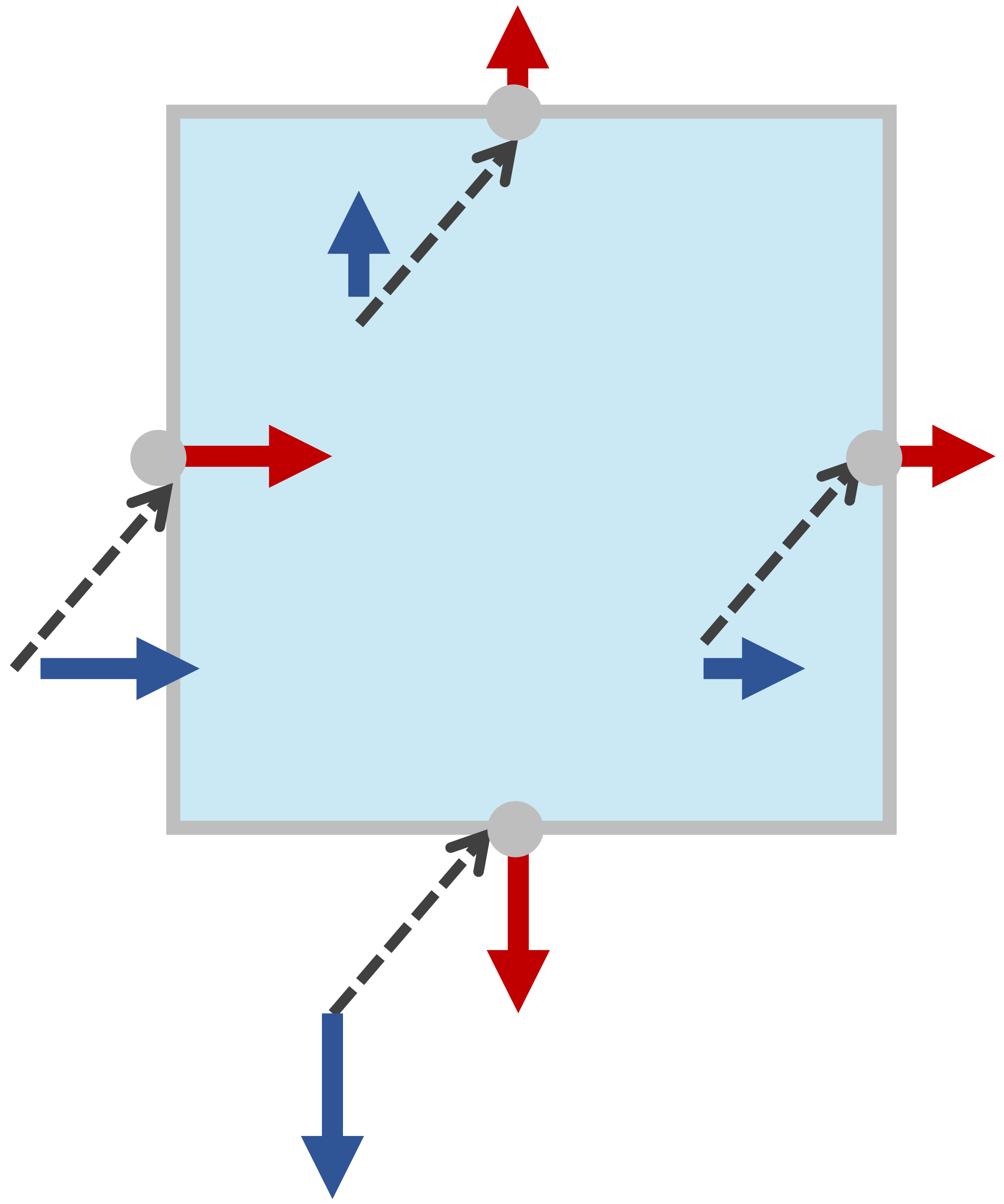
Eulerian Simulation Recap



add gravity



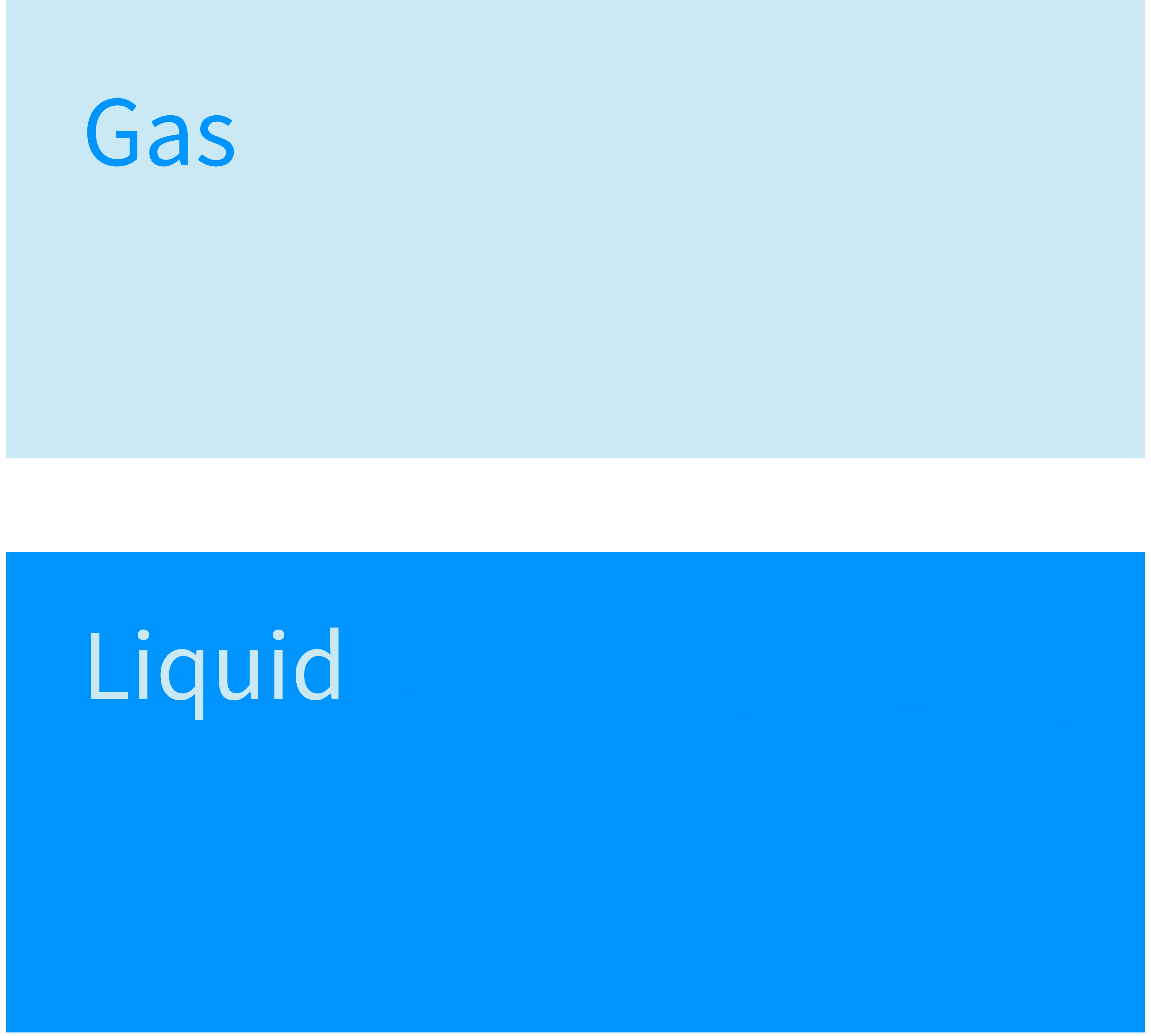
make incompressible



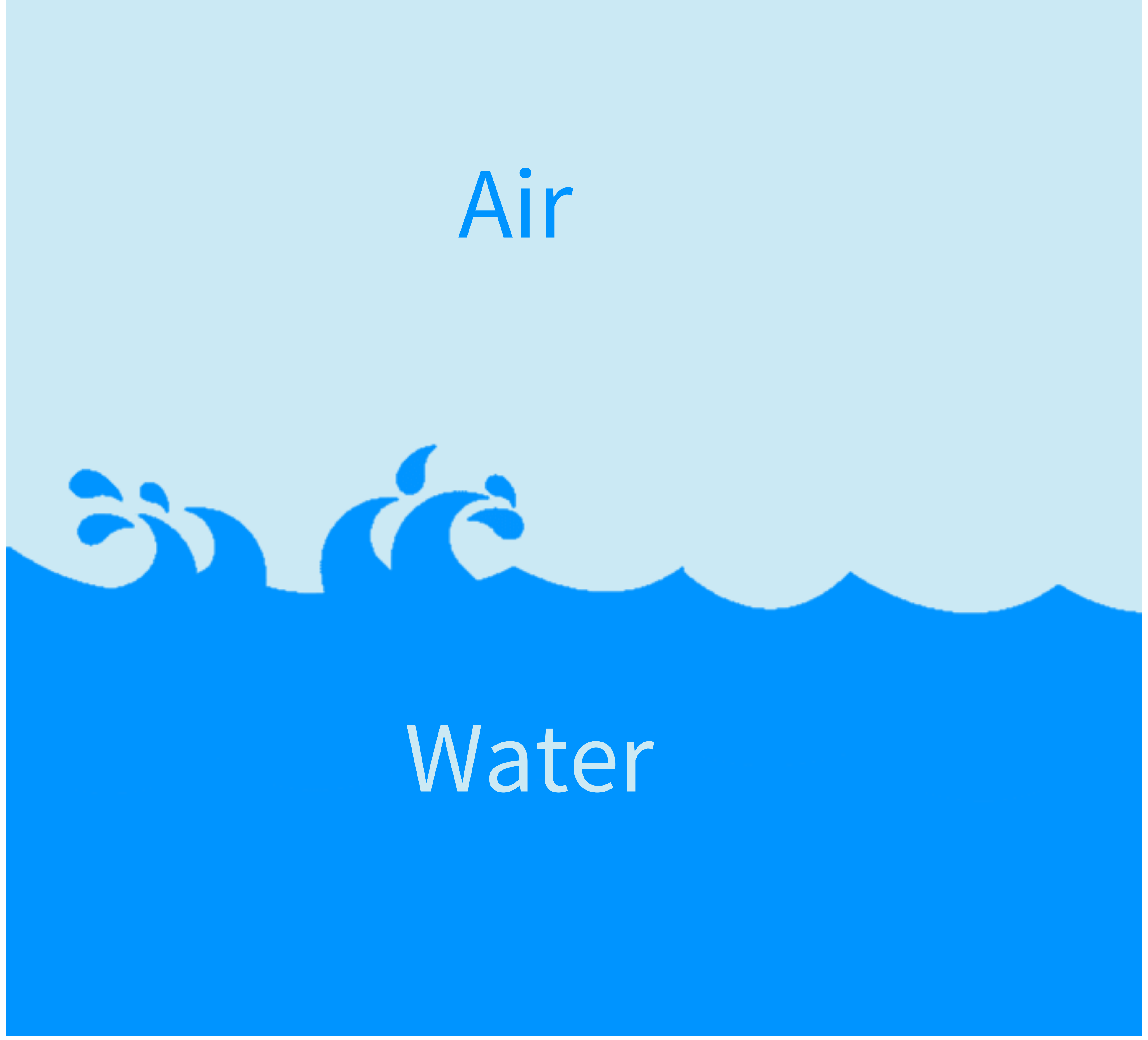
advect

Slide from Matthias Müller, Ten Minute Physics

Goal



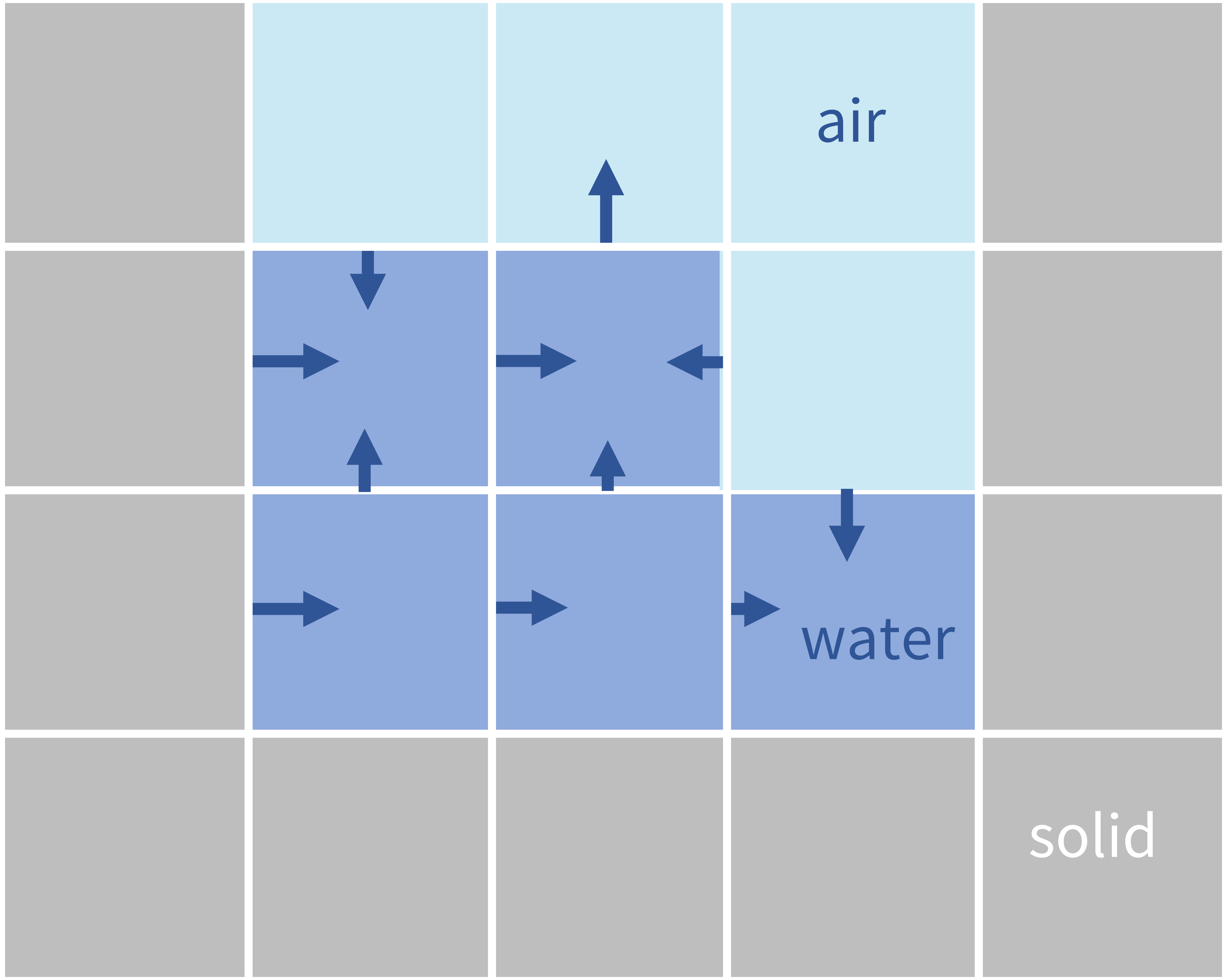
Last tutorial: separate simulations



This tutorial: combined simulation

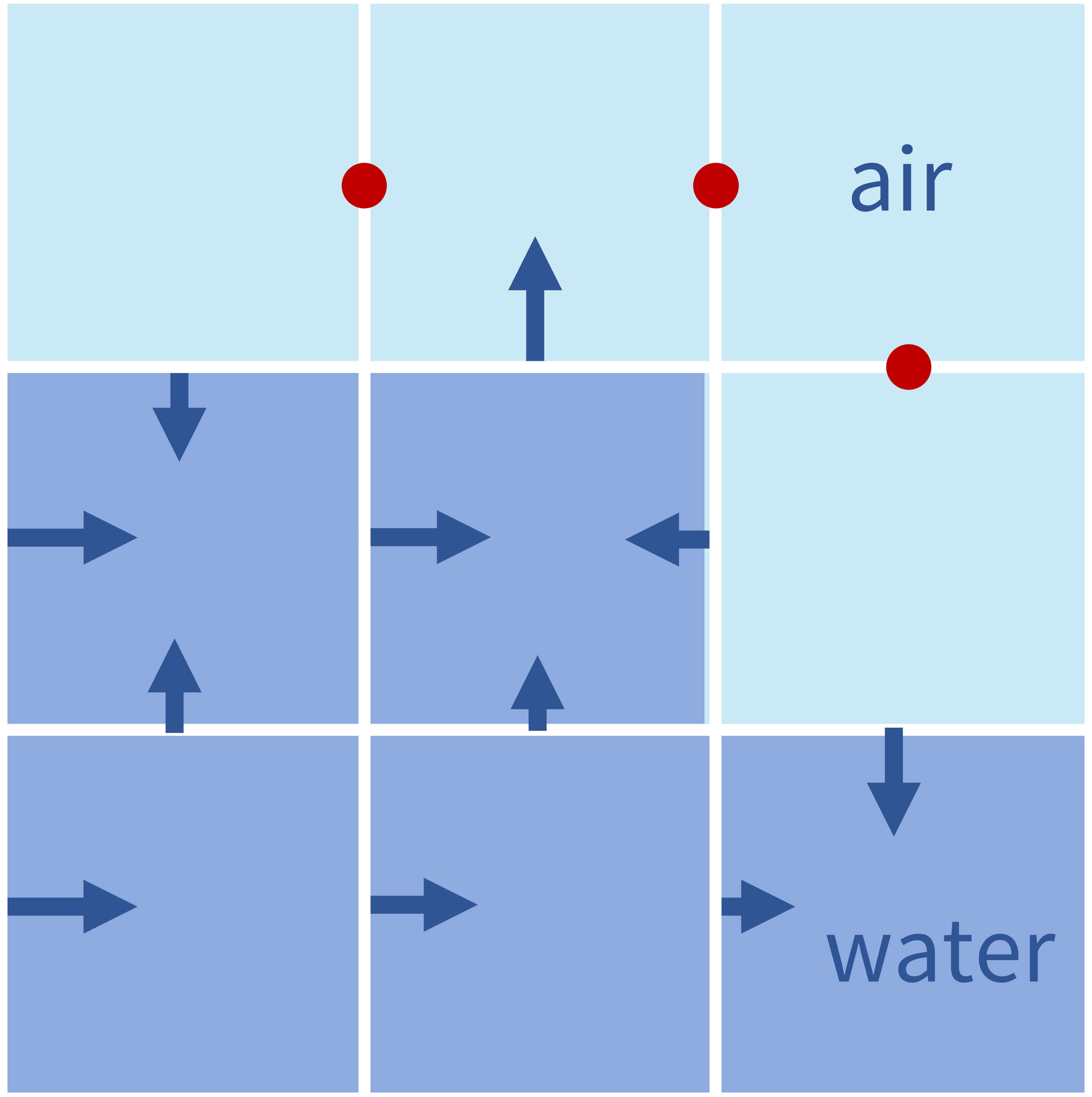
Slide from Matthias Müller, Ten Minute Physics

Two Phase Simulation



Slide from Matthias Müller, Ten Minute Physics

Two Phase Simulation

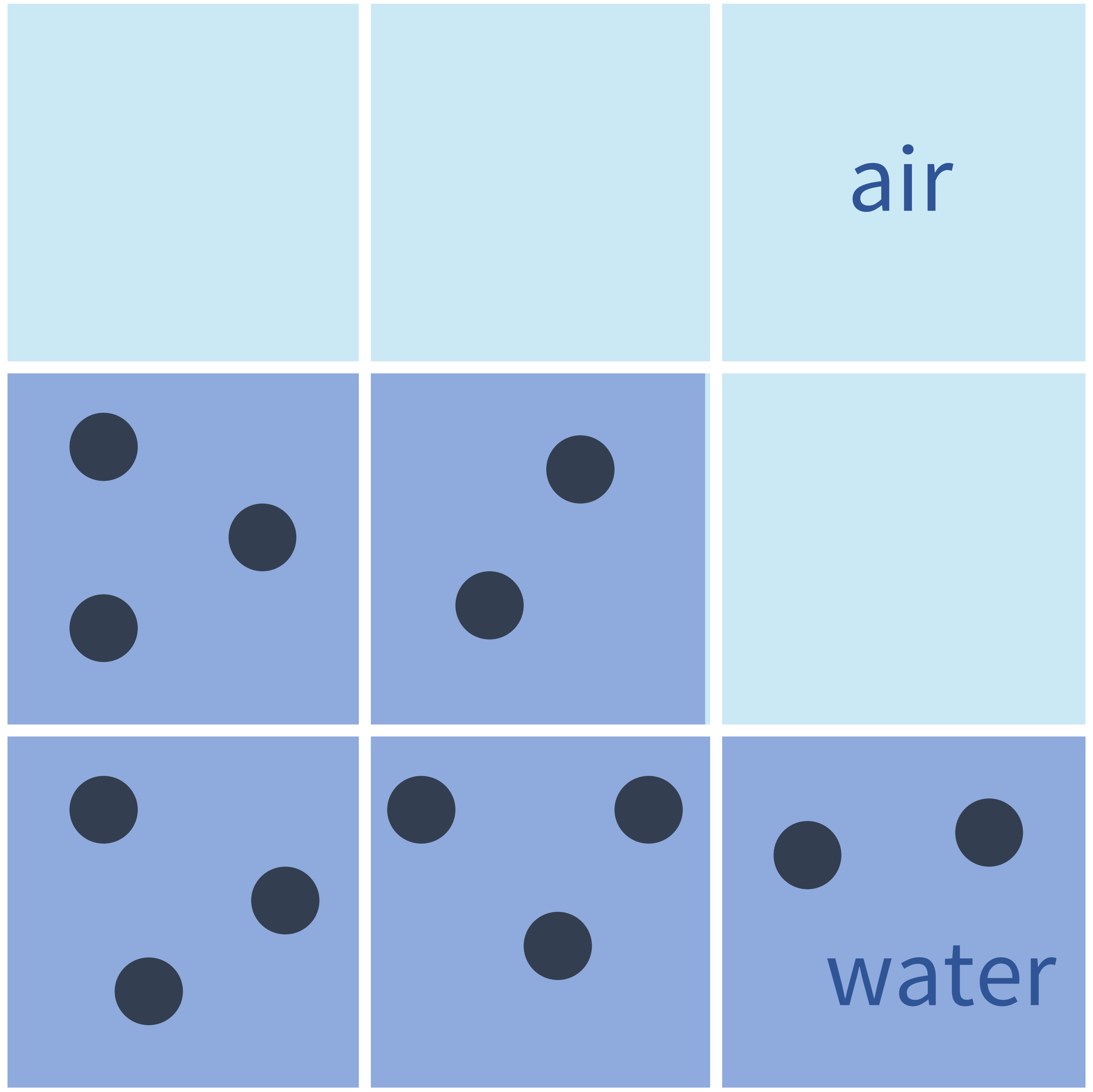


- Density of water $\approx 1000 \text{ kg/m}^3$
- Density of air $\approx 1 \text{ kg/m}^3$
- Treat air as *nothing*
- Velocities between air cells are *undefined* (not zero)!

1. Do not process air cells

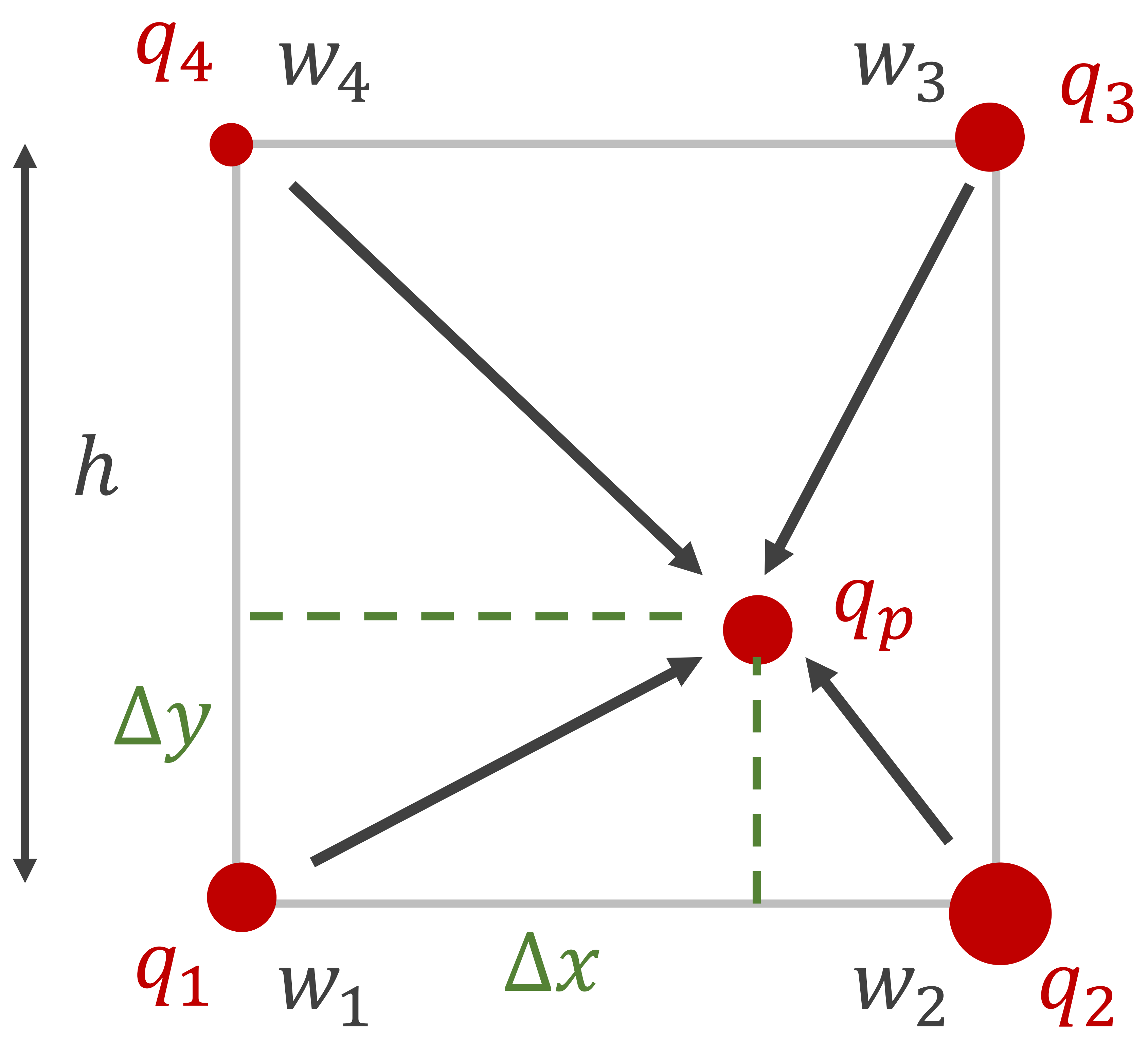
2. Do not access velocities between air cells!

Cell Type Determination



- Use simulated particles storing a position and velocity!
- Water cells: non-solid cells that contain particles
- PIC: *Particle In Cell* method

From Grid to Particles



$$w_1 = \left(1 - \frac{\Delta x}{h}\right)\left(1 - \frac{\Delta y}{h}\right) \qquad w_2 = \frac{\Delta x}{h}\left(1 - \frac{\Delta y}{h}\right)$$

$$w_3 = \frac{\Delta x}{h}\frac{\Delta y}{h} \qquad w_4 = \left(1 - \frac{\Delta x}{h}\right)\frac{\Delta y}{h}$$

for all particles:

$$q_p = \frac{w_1 q_1 + w_2 q_2 + w_3 q_3 + w_4 q_4}{w_1 + w_2 + w_3 + w_4}$$

If q_2 is undefined: $$q_p = \frac{w_1 q_1 + w_3 q_3 + w_4 q_4}{w_1 + w_3 + w_4}$$

Slide from Matthias Müller, Ten Minute Physics

Problems with PIC

Resampling causes smoothing

- particle velocities are averaged at grid points
- grid velocities are averaged at particles

Result: severe damping of motion

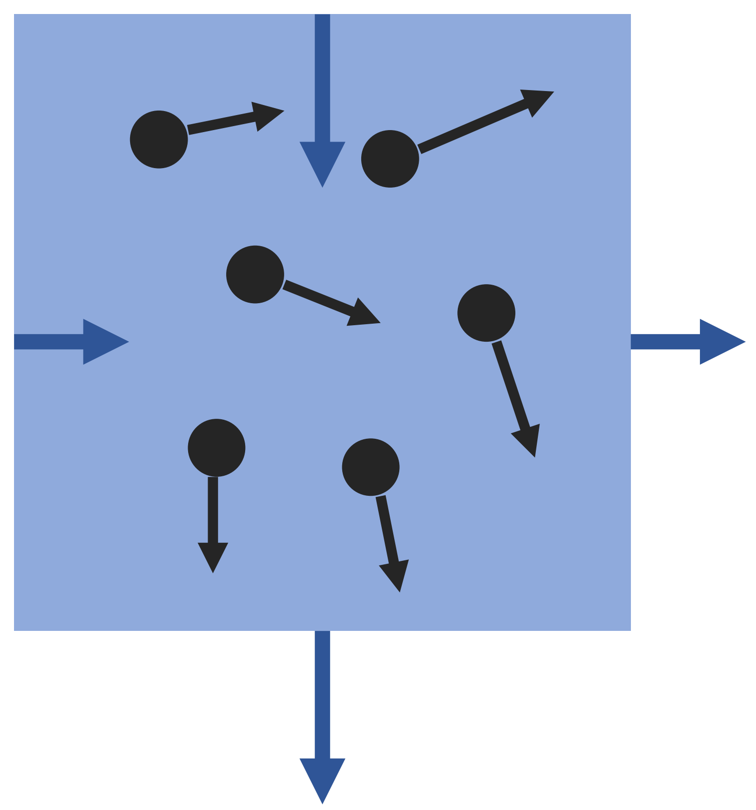
- overall effect like viscosity but with angular momentum loss

Smoothing introduces divergence

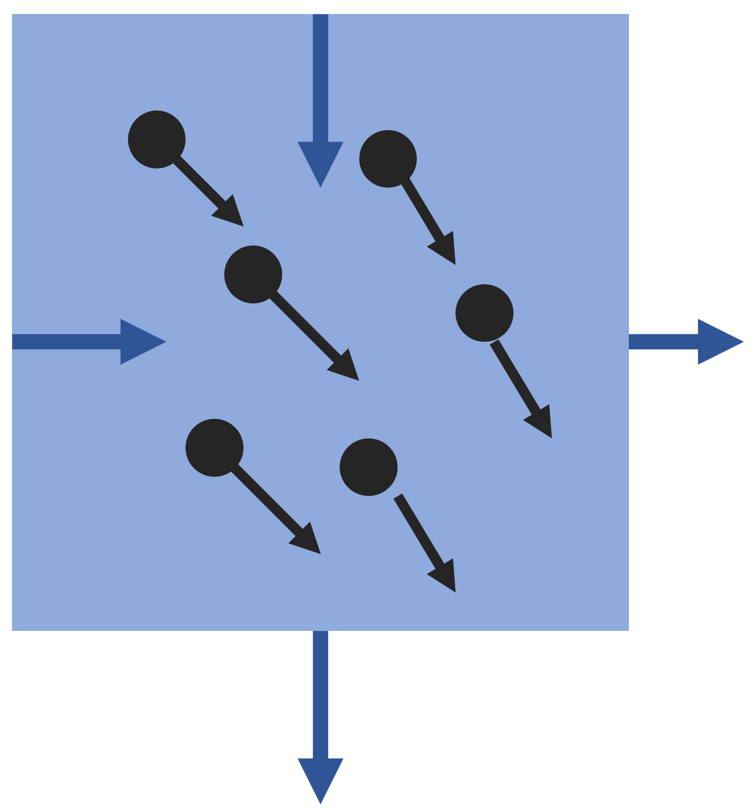
- small errors in particle trajectories accumulate
- particles density becomes wrong

Solutions:

- damping: FLIP, APIC; drift: position-based nudges



before grid → particles

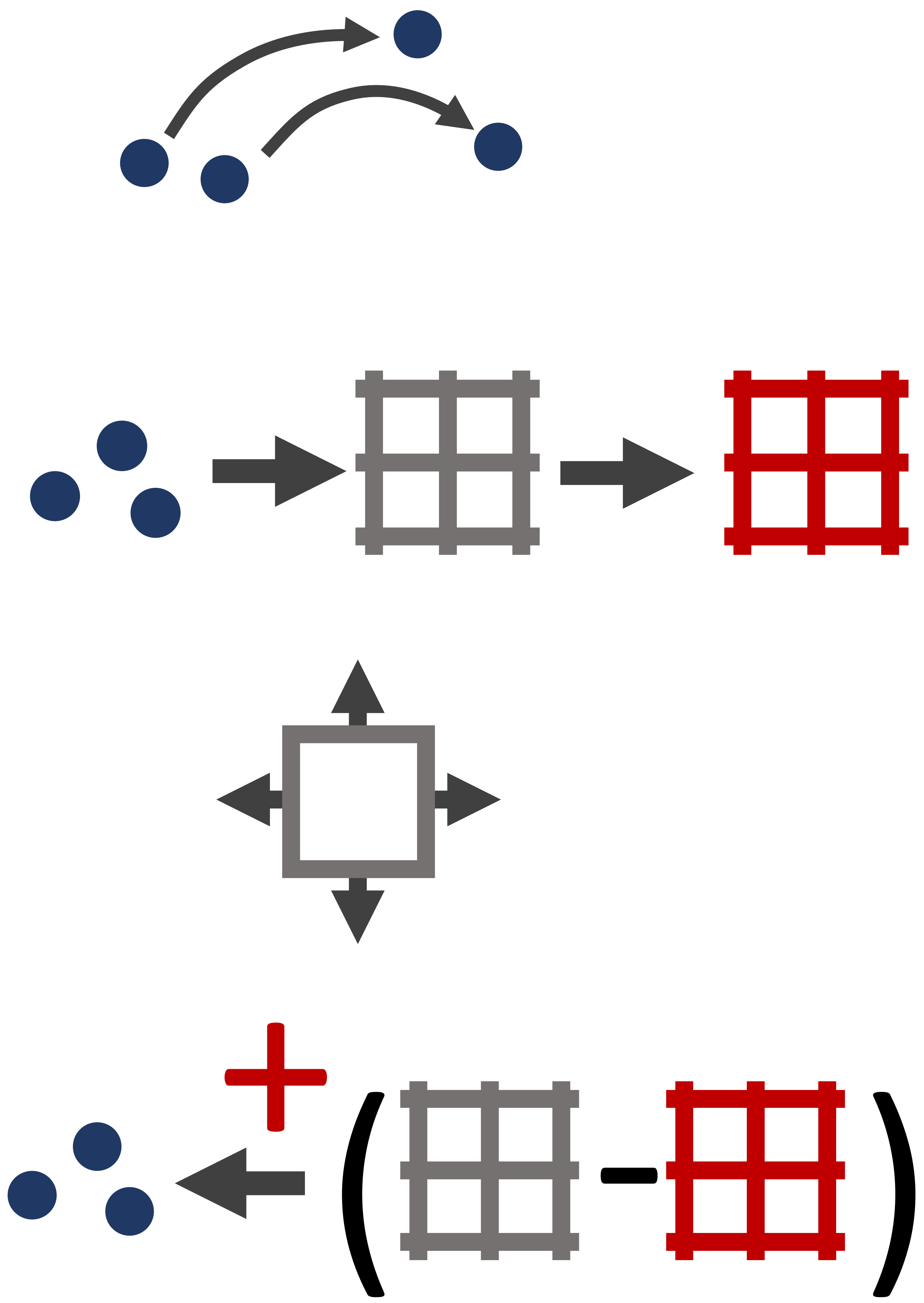


after grid → particles

Figure from Matthias Müller, Ten Minute Physics

FLIP Method

- Simulate particles
- Particles velocity → Grid | Make a **copy**
- Make the grid velocities incompressible
- **Add change** to the particles



- More detail but also more noise! → mix: 0.1 * PIC + 0.9 * FLIP

Slide from Matthias Müller, Ten Minute Physics

FLIP (**FL**uid **I**mplicit **P**article)

Small change from PIC

- filter velocities to grid as always
- project them to divergence-free as always
- filter the *change* in velocity back to the particles
  rather than replacing the old velocity

Advantage: only smooths at the grid scale and up

- smaller details in velocity remain unchanged
- much better at preserving rotational motions and turbulent flow patterns

Disadvantage: only smooths at the grid scale and up

- small scale noise can grow unstably

Alternative: keep higher order information (APIC, MPM)

Density drift

As with other constraints, enforcing divergence-free flow with even small errors allows density to drift over time

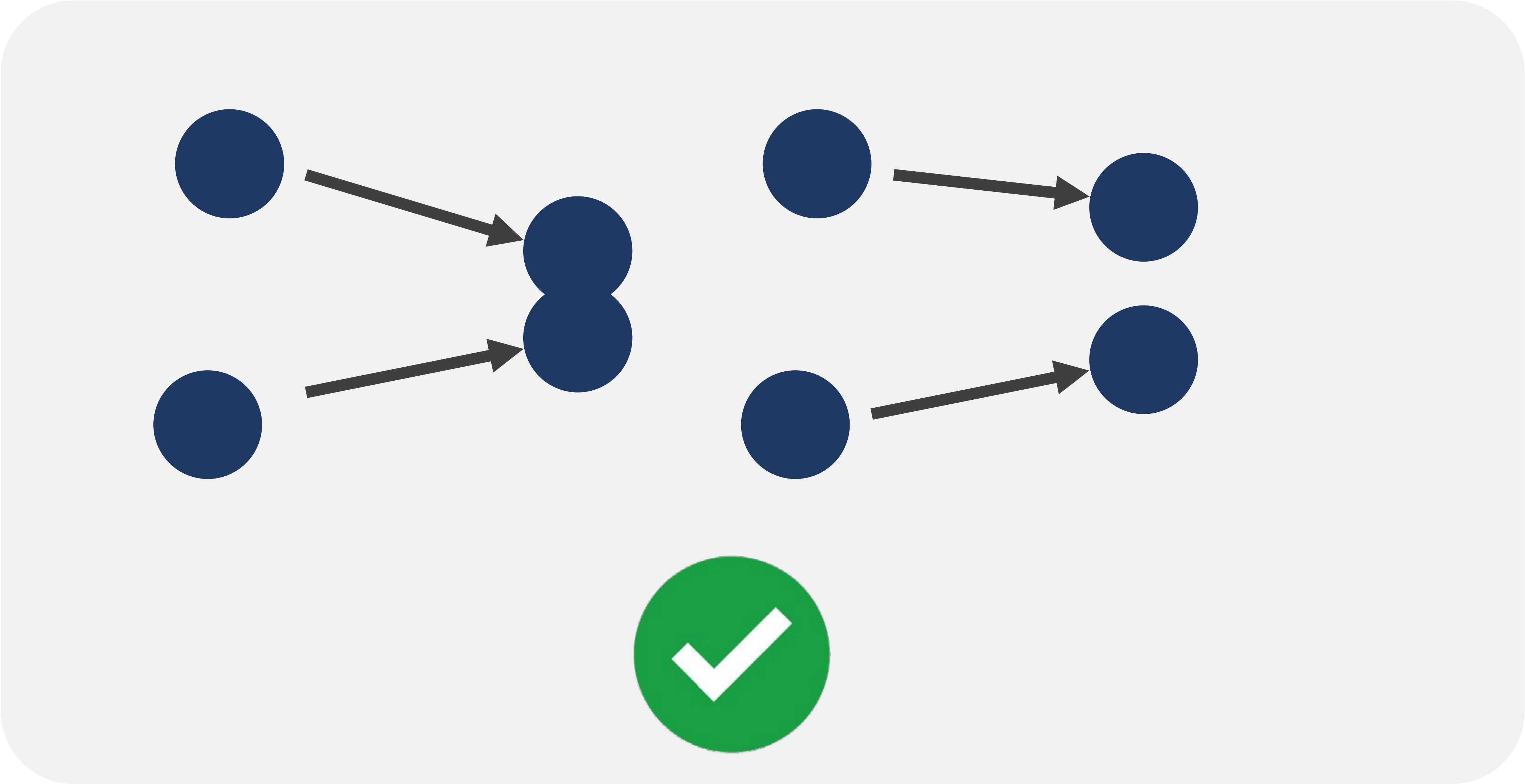
- analogous to constraint drift in our rigid body simulator
- this is a property of any simulation that adjusts velocities or forces without ever looking at the positions

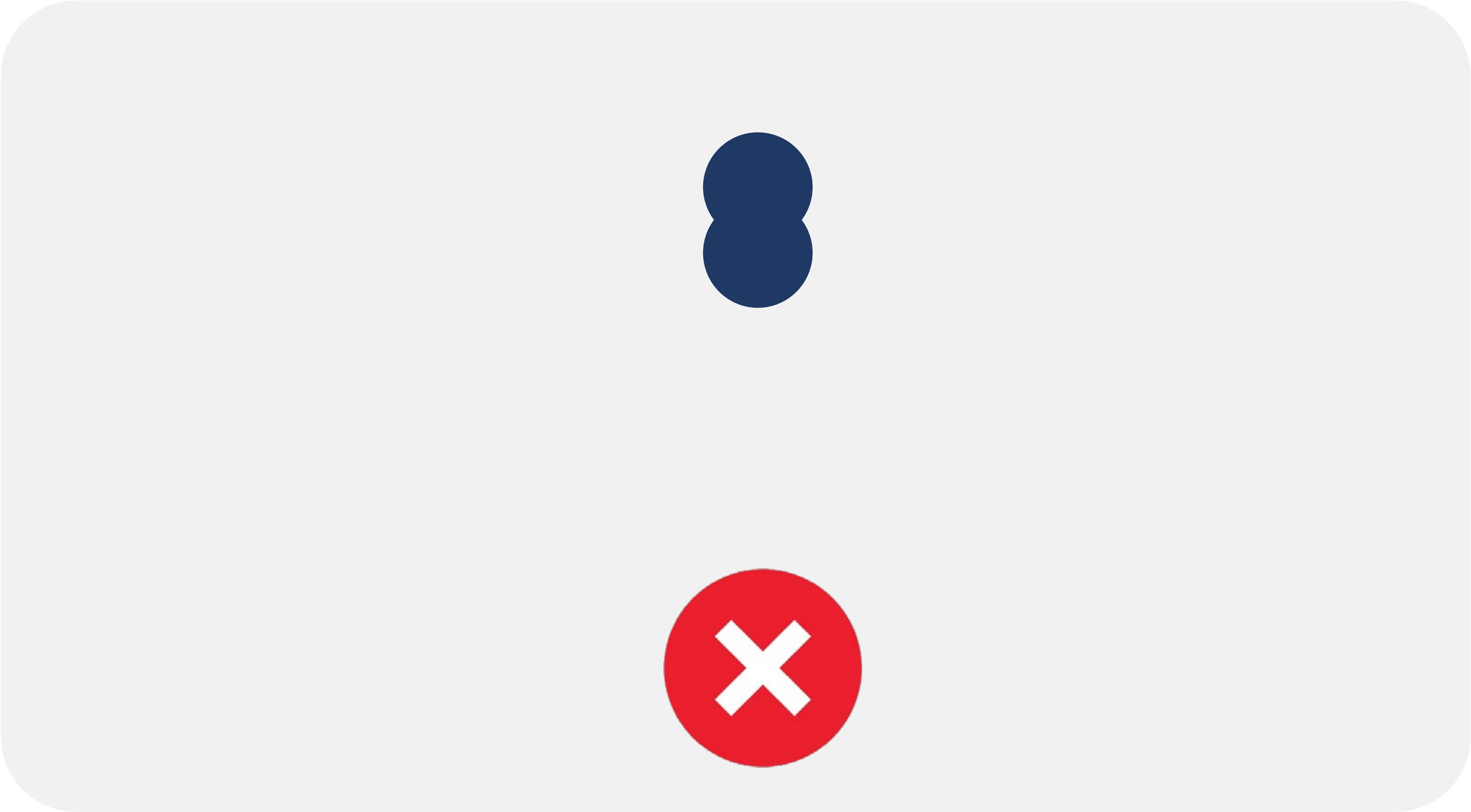
Solutions

- use high enough accuracy to keep things OK for the simulation duration
- introduce ad hoc drift repair mechanisms
  - monitor a property (e.g. particle density) that should be conserved
  - introduce a small force to nudge the system towards goal

Drift

- All purely velocity-based approaches have this problem:





- The solver sees that:

  velocities tend to make particles collide

- The solver does not see that:

  particles are already colliding!

- Two fixes necessary…

Slide from Matthias Müller, Ten Minute Physics

Make the Solver Aware of Drift

- Compute a particle density d at the center of each cell

clear ρ of all cells

for all particles

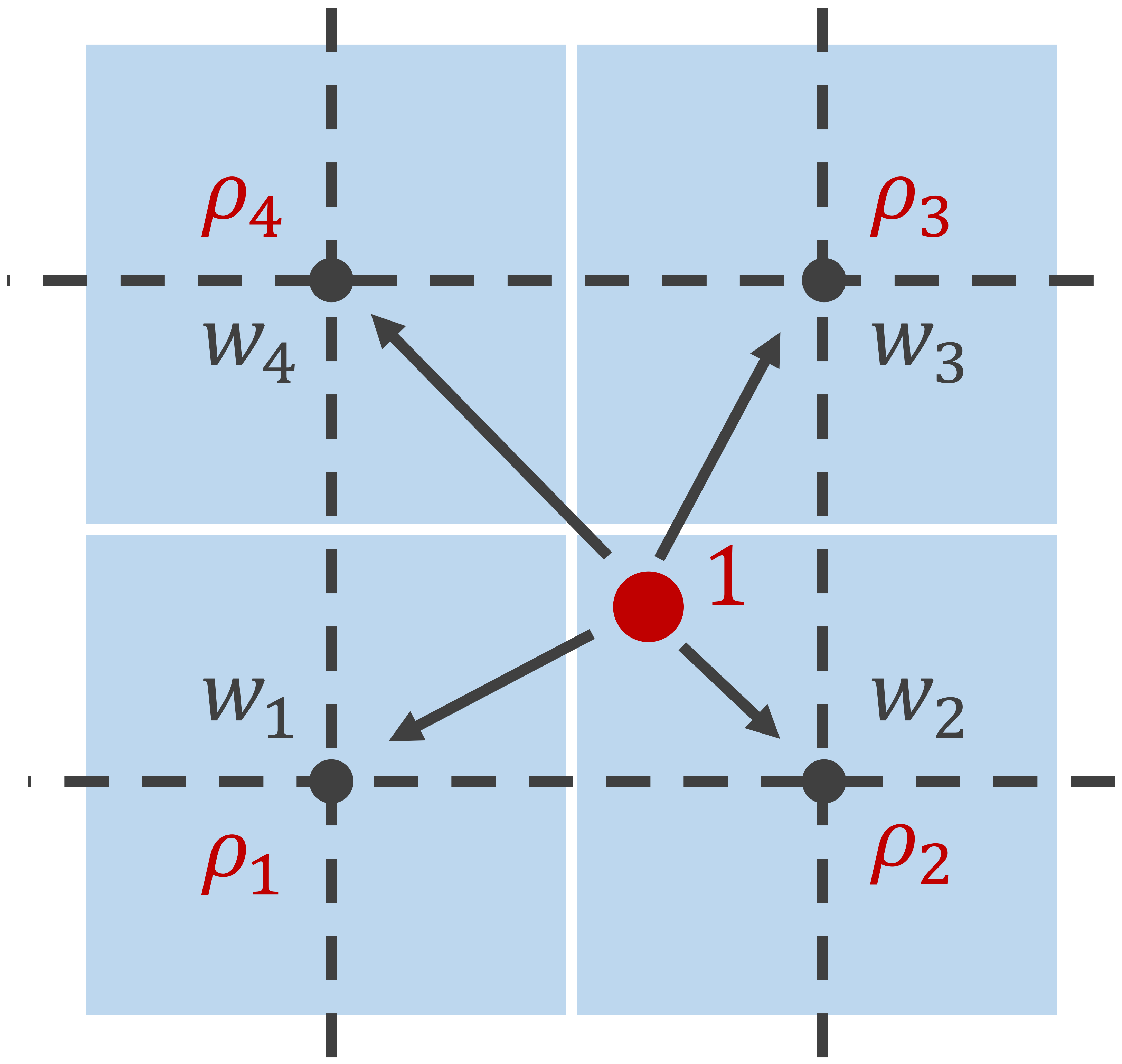
$\rho_1 \leftarrow \rho_1 + w_1$

$\rho_2 \leftarrow \rho_2 + w_2$

$\rho_3 \leftarrow \rho_3 + w_3$

$\rho_4 \leftarrow \rho_4 + w_4$

- $w_1 + w_2 + w_3 + w_4 = 1$



- Grid is shifted by $\frac{h}{2}$ in both directions!

Slide from Matthias Müller, Ten Minute Physics

Modify Divergence

- Reduce divergence in dense regions:

$$d \leftarrow \boldsymbol{o}\left(u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}\right) - k(\rho - \rho_0)$$

- Causes more outward push
- Rest density ρ_0 is the average density of water cells before the simulation starts
- Parameter k is a stiffness coefficient (1 in my code)

Affine Particle-in-Cell Method (APIC)

Goal: eliminate angular velocity loss

- Makes changes to the particle ↔ grid transfers
- Carries additional information with particles

Idea: particles sample spatial derivative of velocity

- $\mathbf{v} : \mathbb{R}^2 \to \mathbb{R}^2$ — the fluid velocity; particle velocities are samples of it
- $\nabla \mathbf{v} \in \mathbb{R}^{2\times 2}$ — an affine transformation that describes how the fluid is deforming nearby
- in APIC, each particle carries an affine transformation $\mathbf{C}$ as well as a velocity
  - e.g. in a fluid undergoing a pure rotation, $\mathbf{C} = \omega^{\times}$ (skew sym. derivative of rotation)
  - e.g. in a shear layer $\mathbf{C}$ would be a shear transformation


APIC
our method
RPIC
PIC
FLIP
FLIP .99
FLIP .95


APIC
FLIP
PIC
FLIP .95
©Disney


©Disney