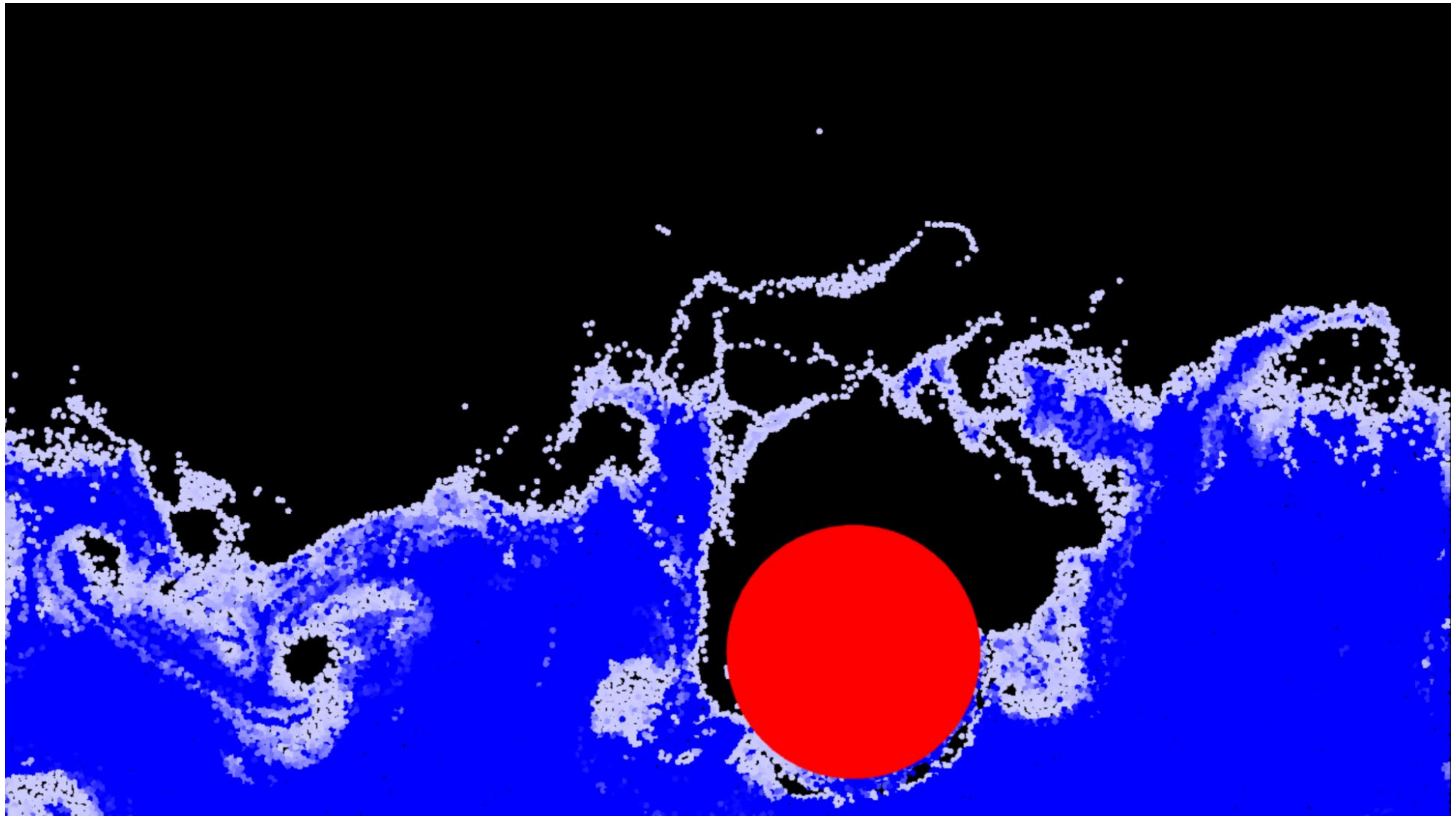


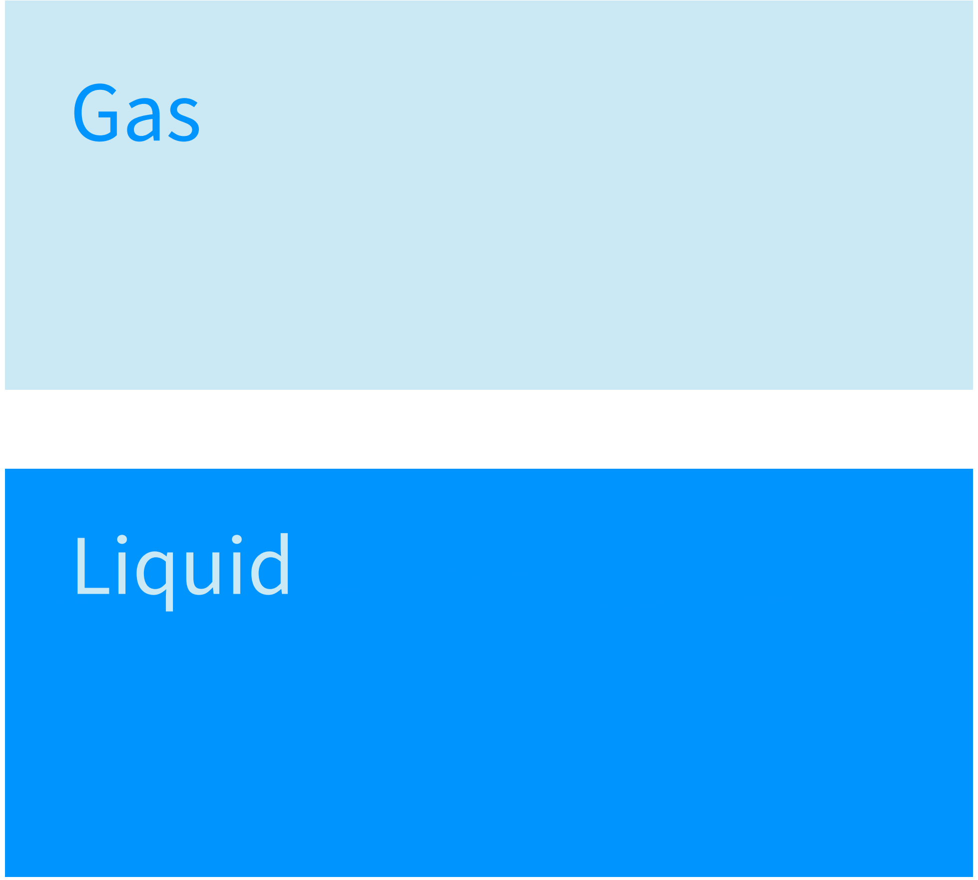
How to write a FLIP Water Simulator

Matthias Müller, Ten Minute Physics

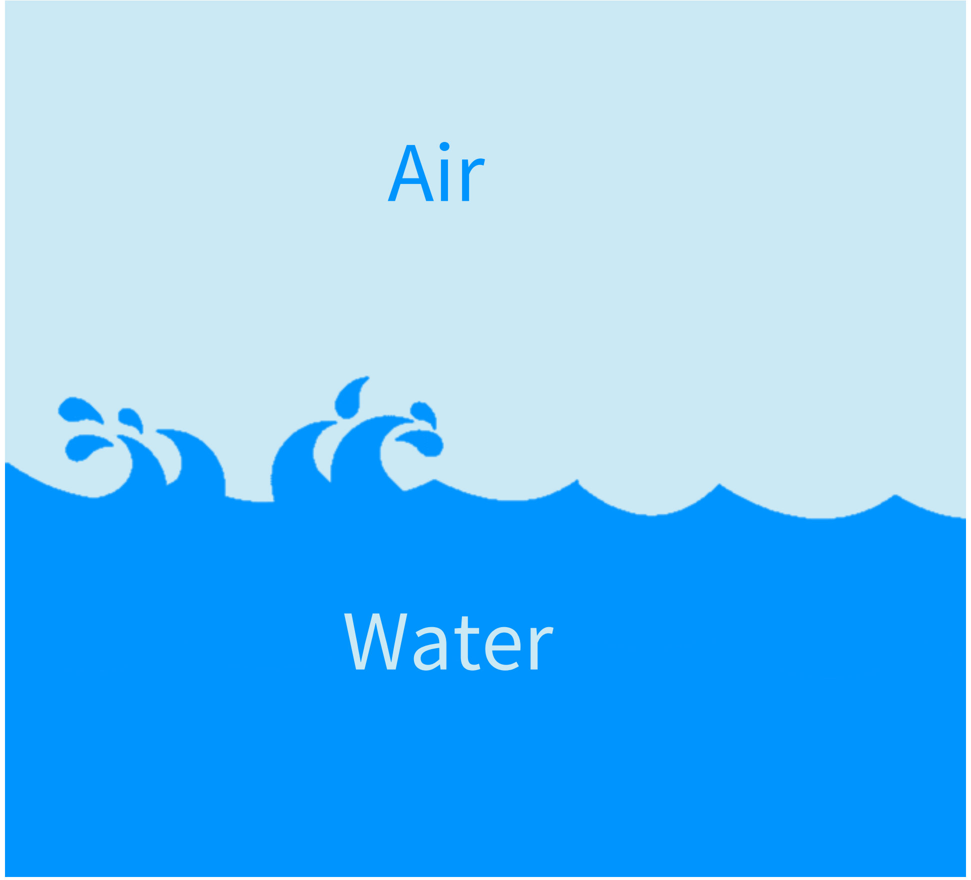
For the code and the demo see:

www.matthiasmueller.info/tenMinutePhysics

Goal

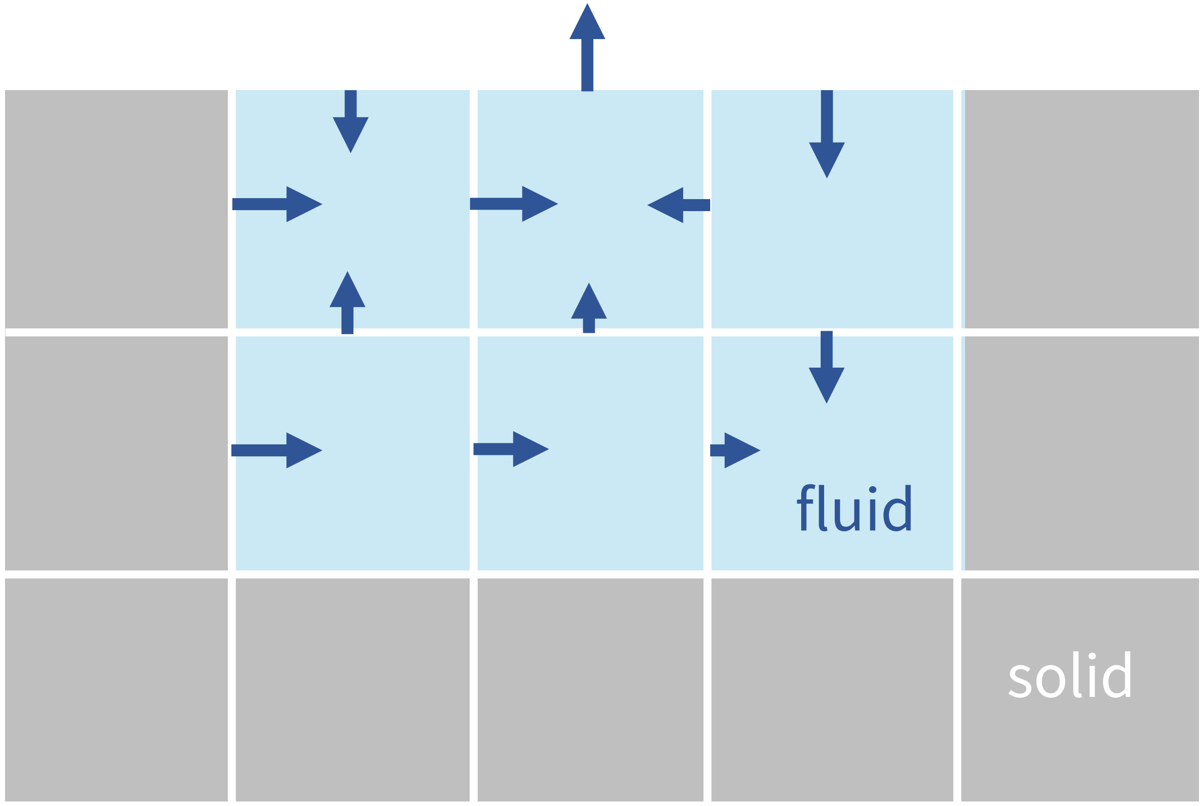


Last tutorial: separate simulations



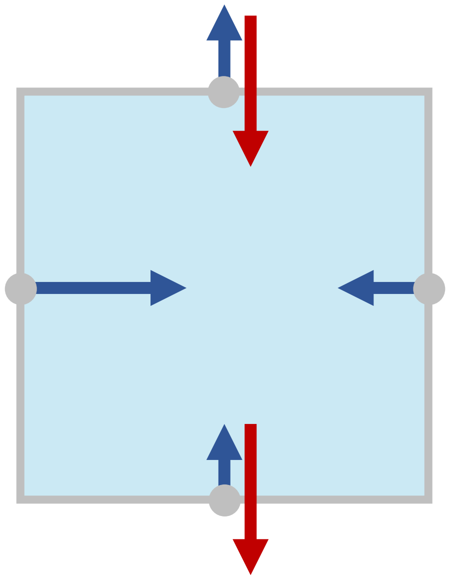
This tutorial: combined simulation

Eulerian Simulation Recap

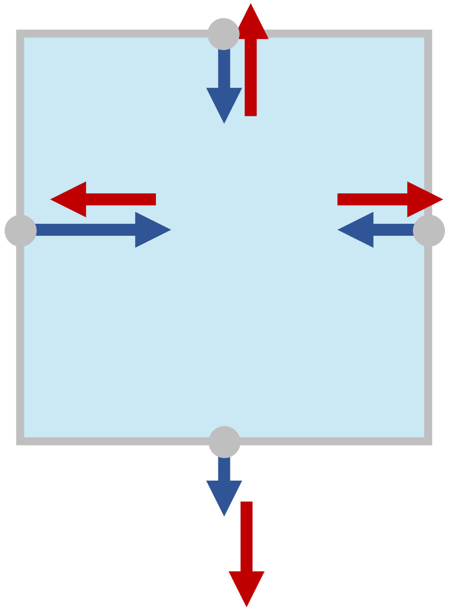


- Fluid as a velocity field stored in a staggered grid
- Two types of cells: fluid and solid

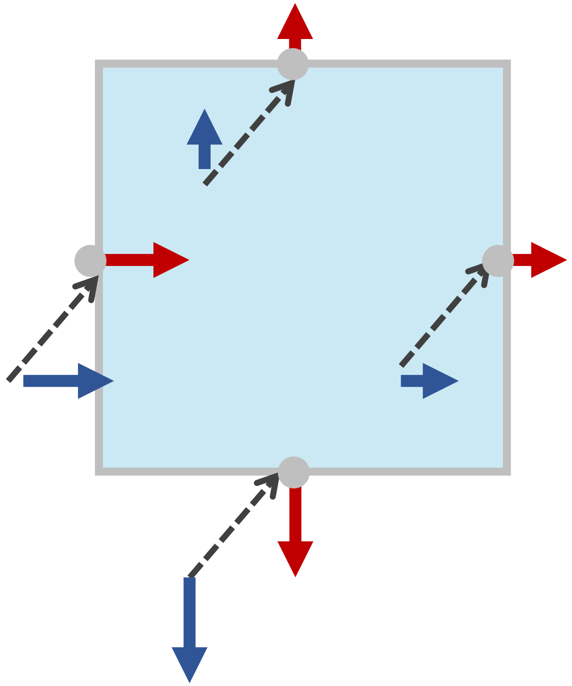
Eulerian Simulation Recap



add gravity

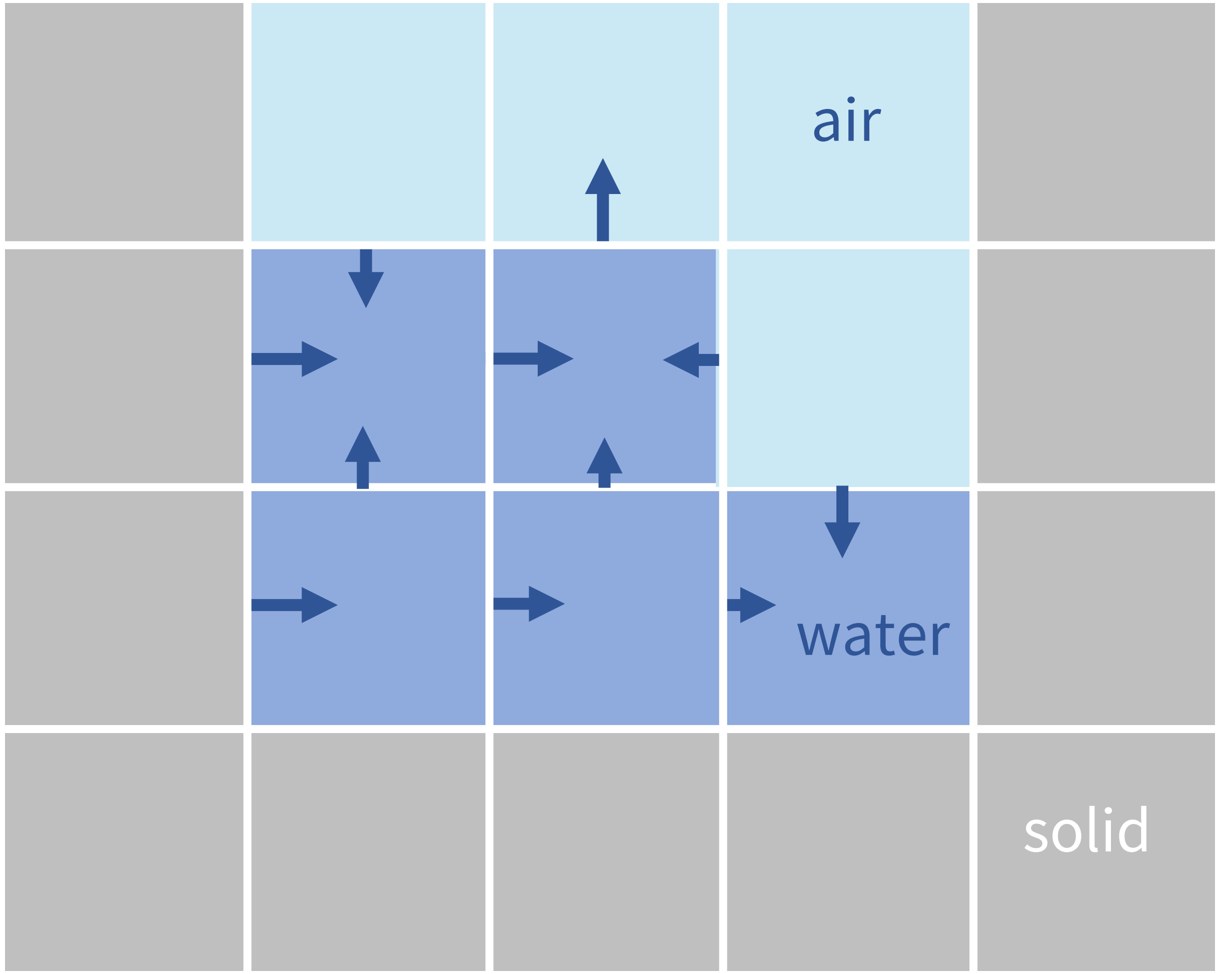


make incompressible

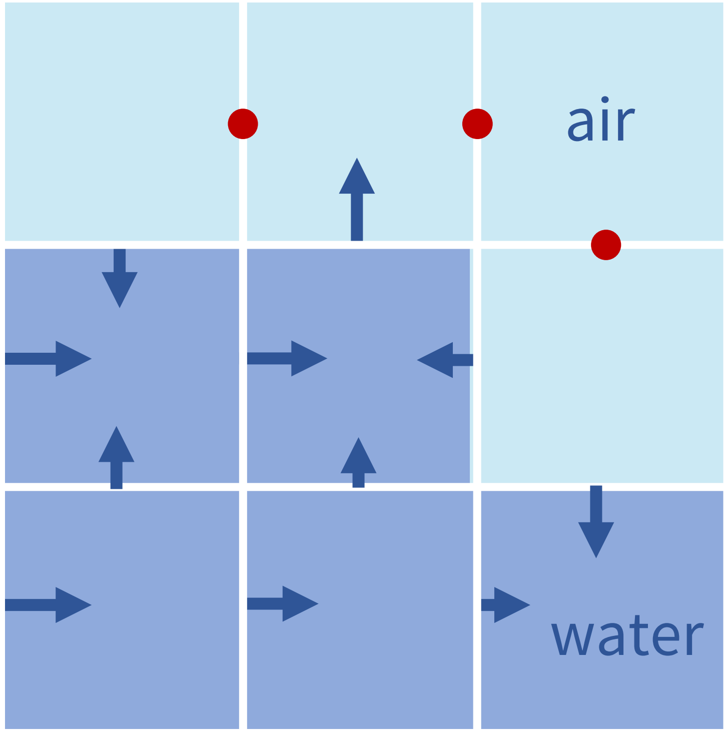


advect

Two Phase Simulation



Two Phase Simulation

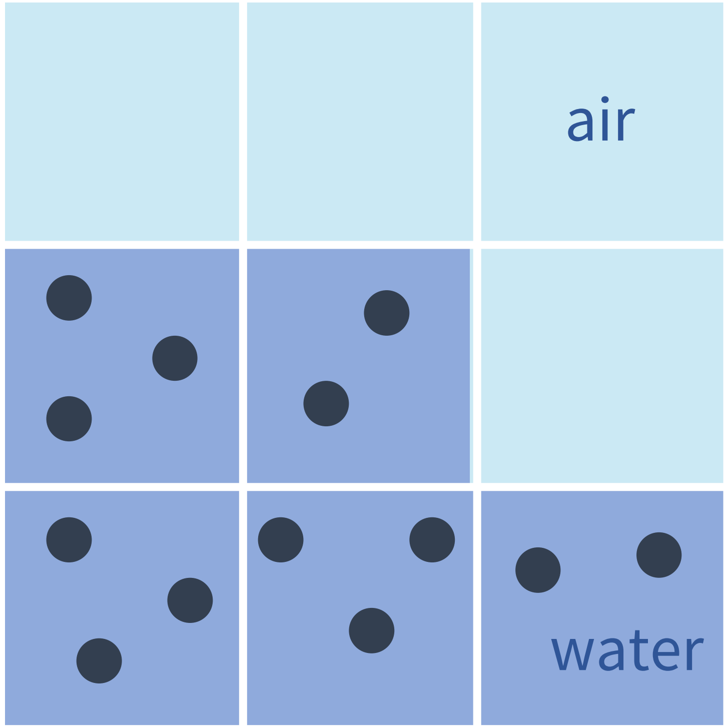


- Density of water ≈ 1000 kg/m^3
- Density of air ≈ 1 kg/m^3
- Treat air as *nothing*
- Velocities between air cells are *undefined* (not zero)!

1. Do not process air cells

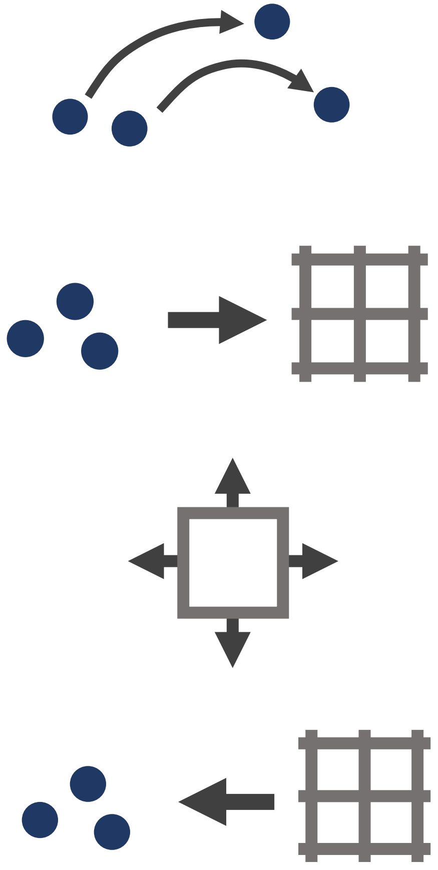
2. Do not access velocities between air cells!

Cell Type Determination



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

PIC Method



Simulate particles

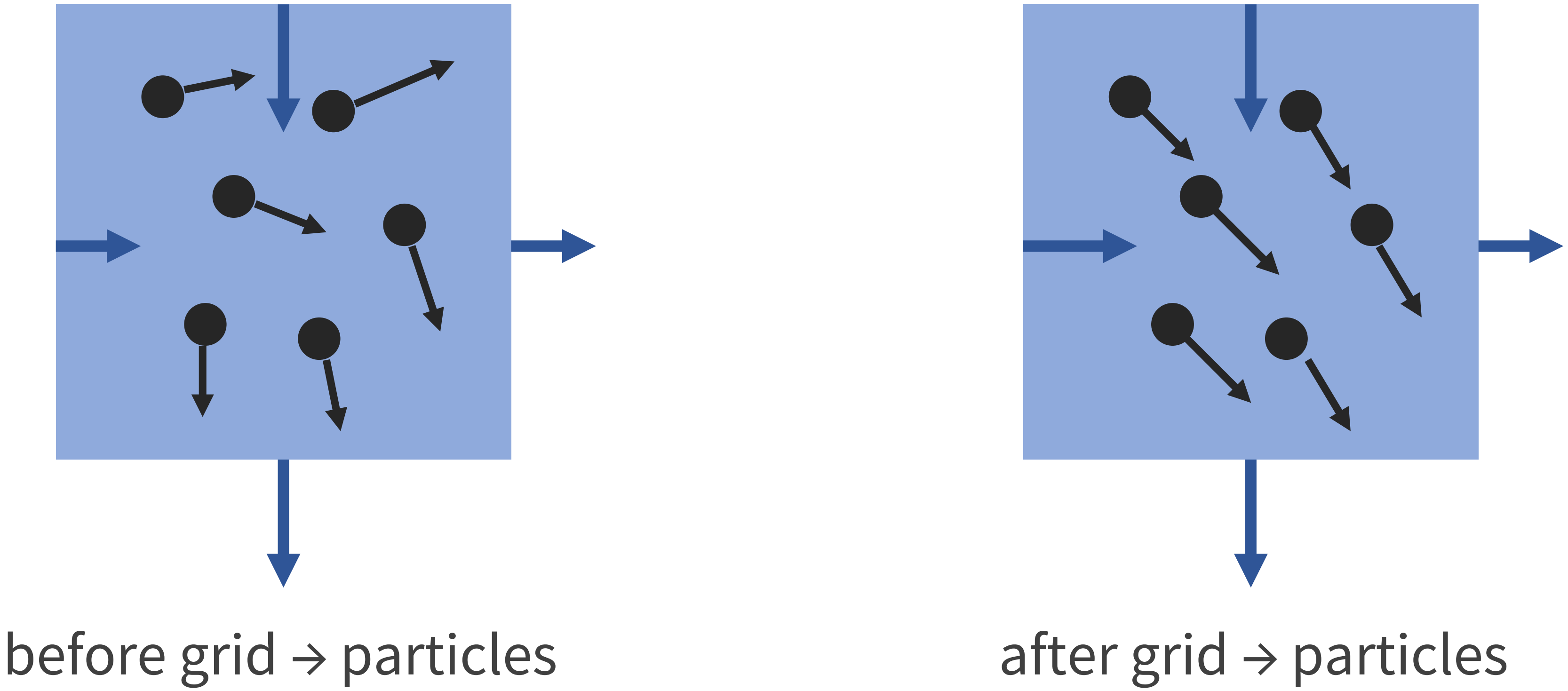
Velocity transfer: Particles → Grid

Make the grid velocities incompressible

Velocity transfer: Grid → Particles

- Particles carry velocity → can skip grid advection!

PIC Viscosity



- Most of the individual particle motion is lost!

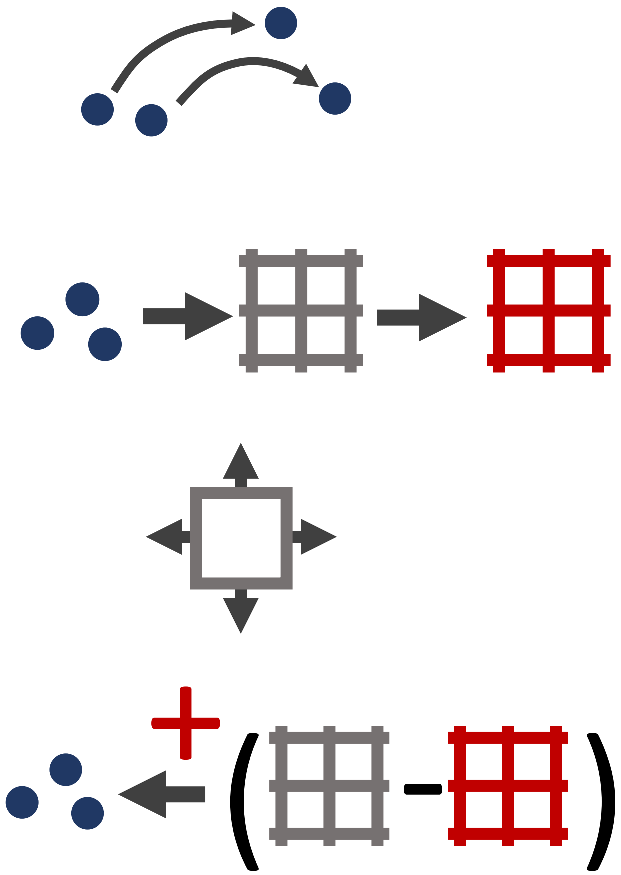
FLIP Method

Simulate particles

Particles velocity → Grid | Make a copy

Make the grid velocities incompressible

Add change to the particles



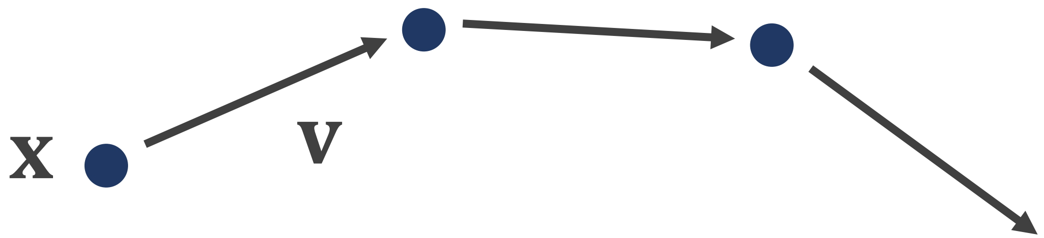
- More detail but also more noise! → mix: 0.1 * PIC + 0.9 * FLIP

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

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

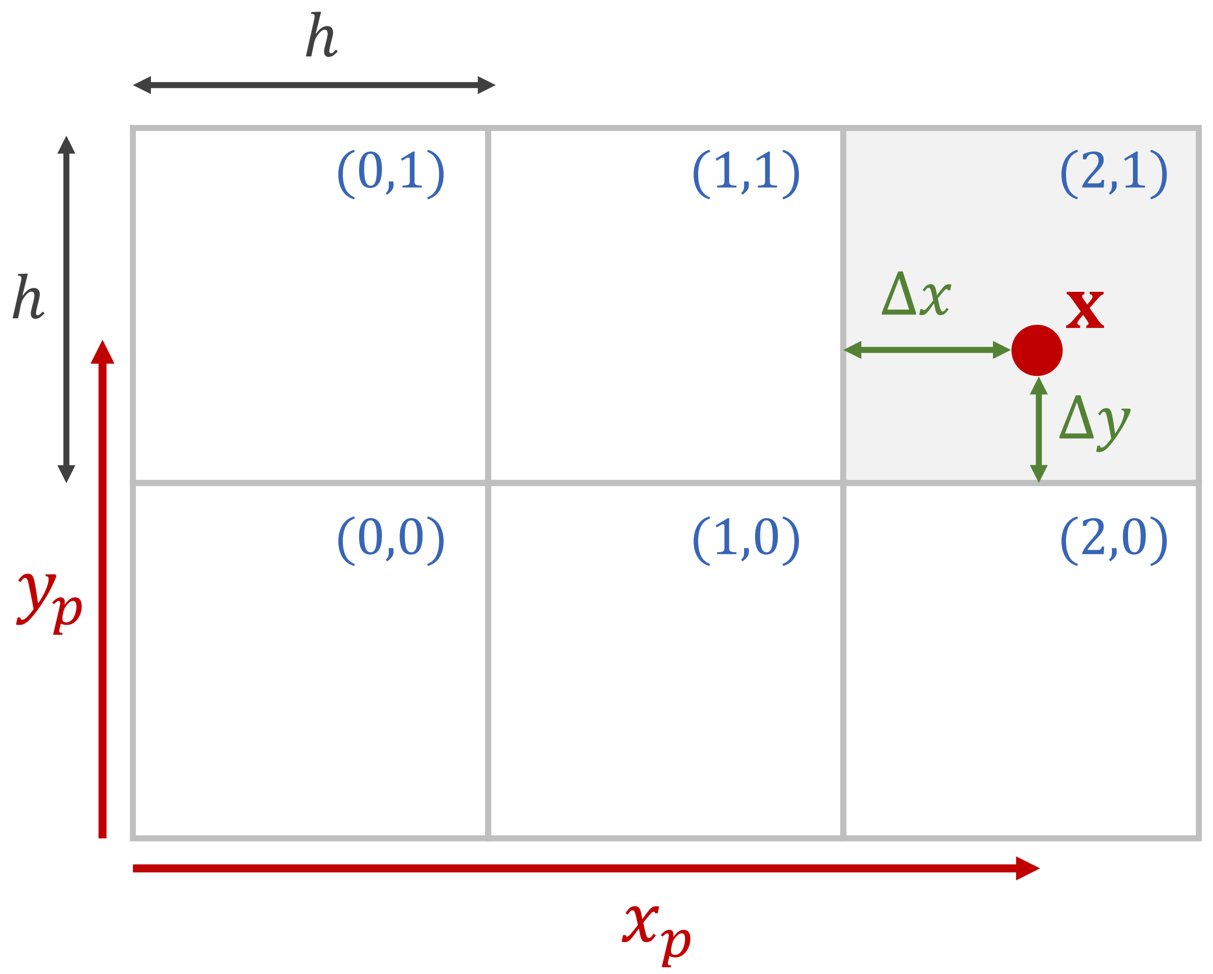


- Push particles out of obstacles!

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

Transferring the velocity field

From Particle to Cell

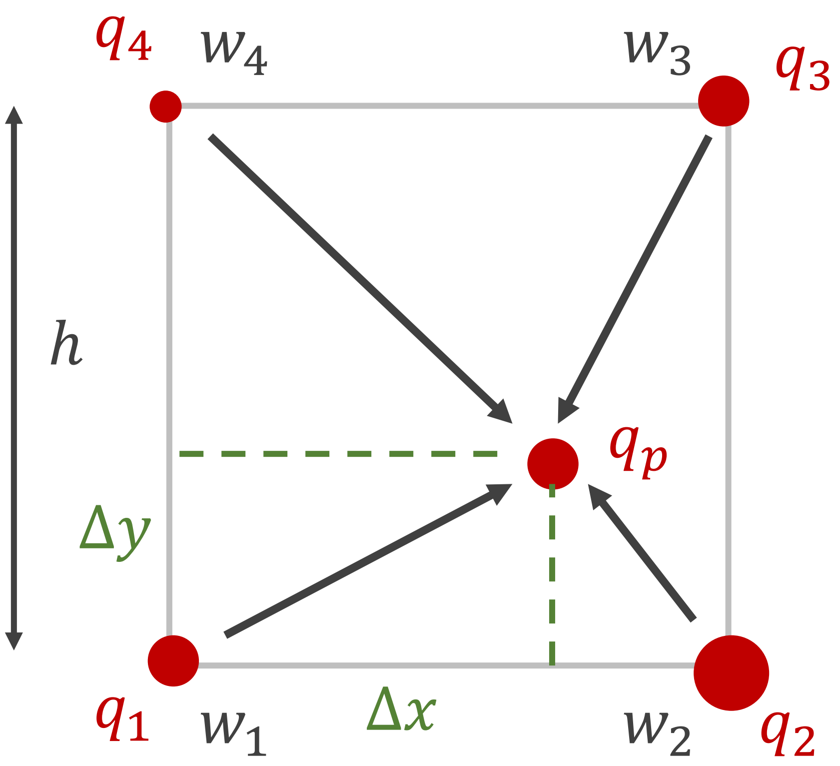


$$x_{cell} = \left\lfloor \frac{x_p}{h} \right\rfloor, \quad \Delta x = x_p - x_{cell} h$$

$$y_{cell} = \left\lfloor \frac{y_p}{h} \right\rfloor, \quad \Delta y = y_p - y_{cell} h$$

rounding down

From Grid to Particles



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

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

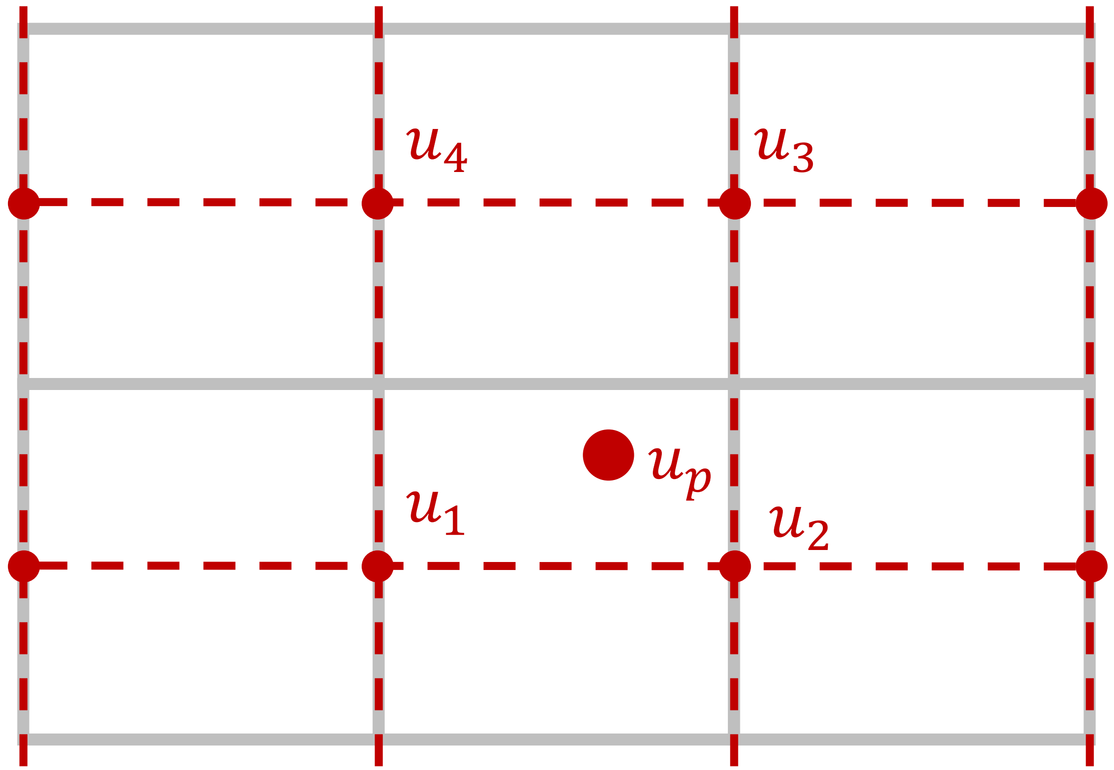
for all particles:

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

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

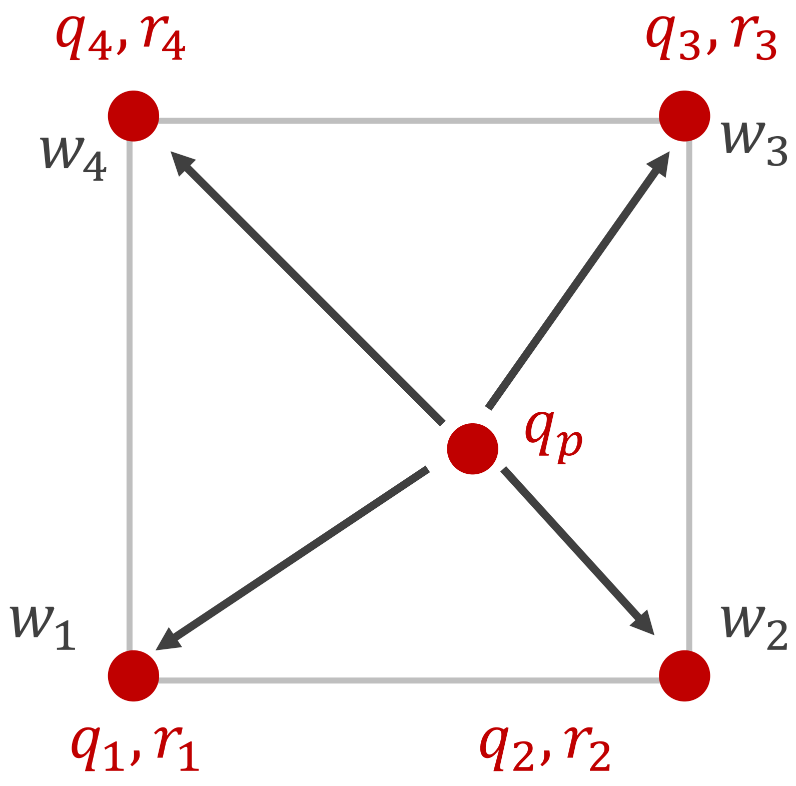
Grid Offsets

- The u component:



- Grid is shifted by $\frac{h}{2}$ in the y-direction!
- Use $\begin{bmatrix} x \\ y - \frac{h}{2} \end{bmatrix}$ as the position for p!

From Particles to Grid



clear q and r of all cells

for all particles

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

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

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

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

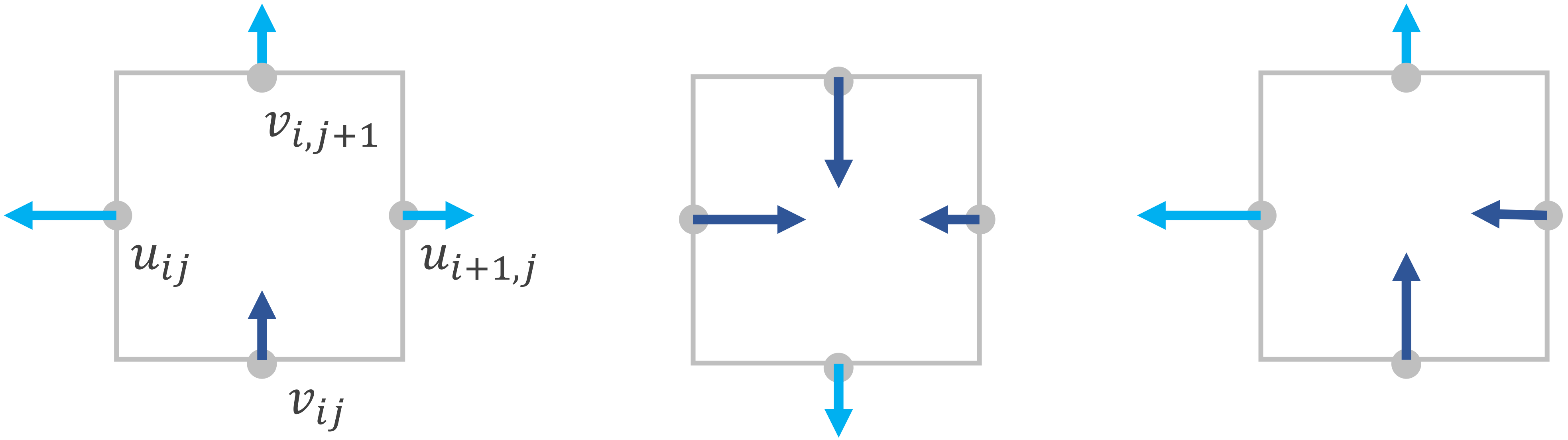
for all cells

$q \leftarrow q/r$

Making the velocity field incompressible

Divergence（Total Outflow)

$$d \leftarrow u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}$$

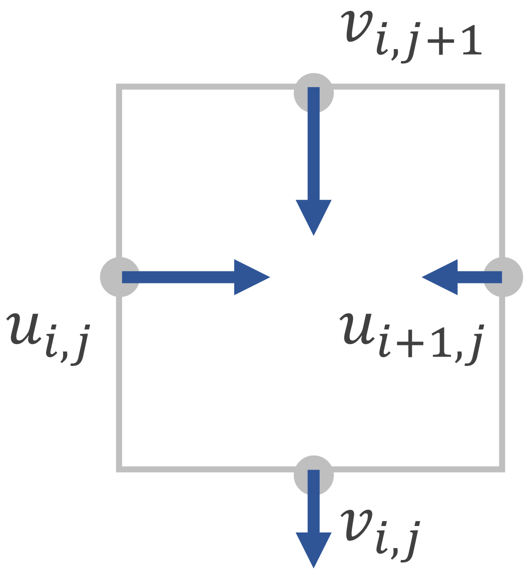


- Positive
- Too much outflow

- Negative
- Too much inflow

- Zero
- Incompressible

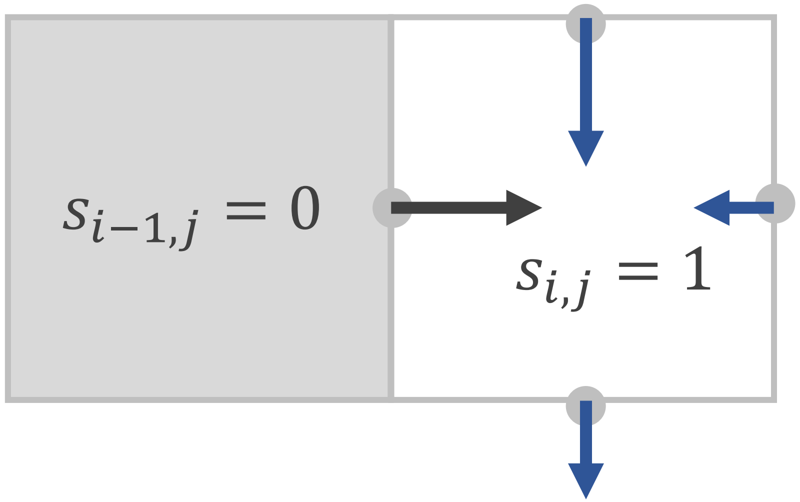
Forcing Incompressibility



$$d \leftarrow u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}$$

$$u_{i,j} \leftarrow u_{i,j} + d/4$$
$$u_{i+1,j} \leftarrow u_{i+1,j} - d/4$$
$$v_{i,j} \leftarrow v_{i,j} + d/4$$
$$v_{i,j+1} \leftarrow v_{i,j+1} - d/4$$

Obstacles / Walls



$$d \leftarrow u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}$$

$$s \leftarrow s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1}$$

$$u_{i,j} \leftarrow u_{i,j} + d\, s_{i-1,j}/s$$

$$u_{i+1,j} \leftarrow u_{i+1,j} - d\, s_{i+1,j}/s$$

$$v_{i,j} \leftarrow v_{i,j} + d\, s_{i,j-1}/s$$

$$v_{i,j+1} \leftarrow v_{i,j+1} - d\, s_{i,j+1}/s$$

Solving the Grid

for n iterations

for all water cells i, j

$$d \leftarrow u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}$$

$$s \leftarrow s_{i+1,j} + s_{i-1,j} + s_{i,j+1} + s_{i,j-1}$$

$$u_{i,j} \leftarrow u_{i,j} + d\, s_{i-1,j}/s$$

$$u_{i+1,j} \leftarrow u_{i+1,j} - d\, s_{i+1,j}/s$$

$$v_{i,j} \leftarrow v_{i,j} + d\, s_{i,j-1}/s$$

$$v_{i,j+1} \leftarrow v_{i,j+1} - d\, s_{i,j+1}/s$$

- Gauss-Seidel method
- On the boundary we access cells outside of the grid!
- Add border cells
- Set $s_{i,j} = 0$ for walls
- or copy neighbor values

Overrelaxation

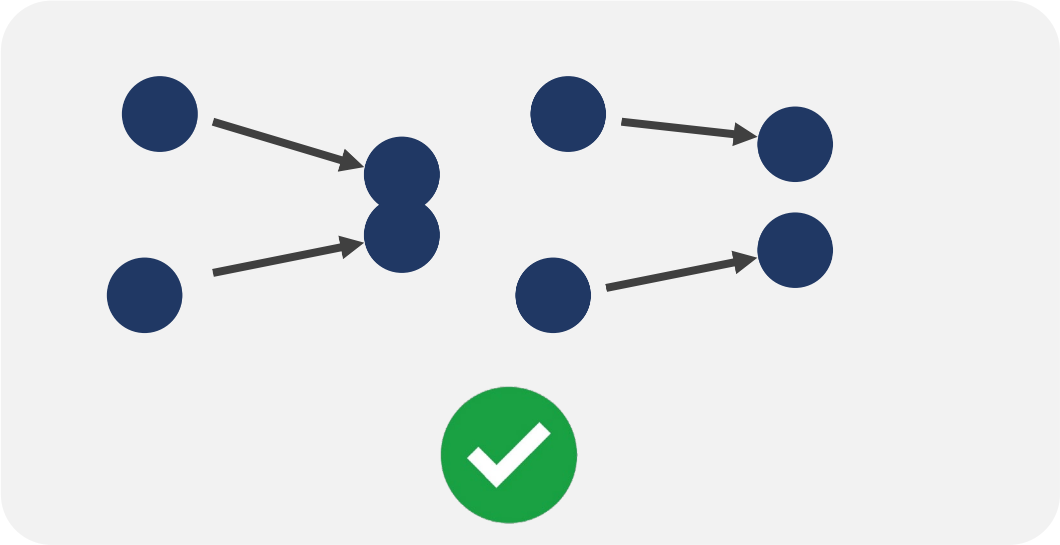
$$d \leftarrow \boldsymbol{o}\left(u_{i+1,j} - u_{i,j} + v_{i,j+1} - v_{i,j}\right)$$

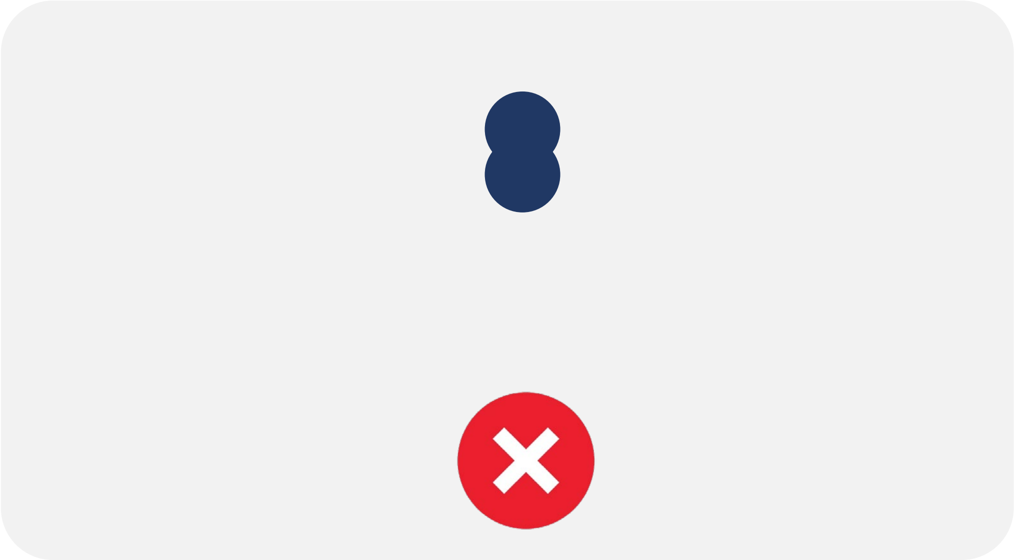
- Increasing the convergence dramatically:
- Multiply the divergence by a scalar $1 < o < 2$
- I use $o = 1.9$ in the code

Drift!

Drift

- All purely velocity-based approaches have this problem:



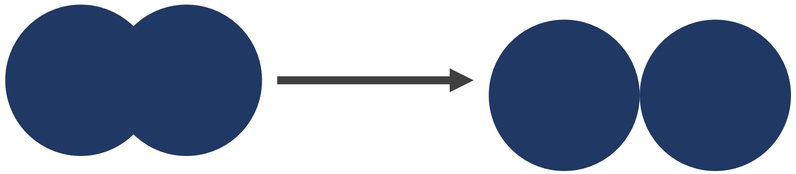


- The solver sees that:
  velocities tend to make particles collide

- The solver does not see that:
  particles are already colliding!

- Two fixes necessary…

Push Particles Apart



- Check all particle pairs is too slow!
- Using grid for speed up

Tutorial 11

Code

Make the Solver Aware of Drift

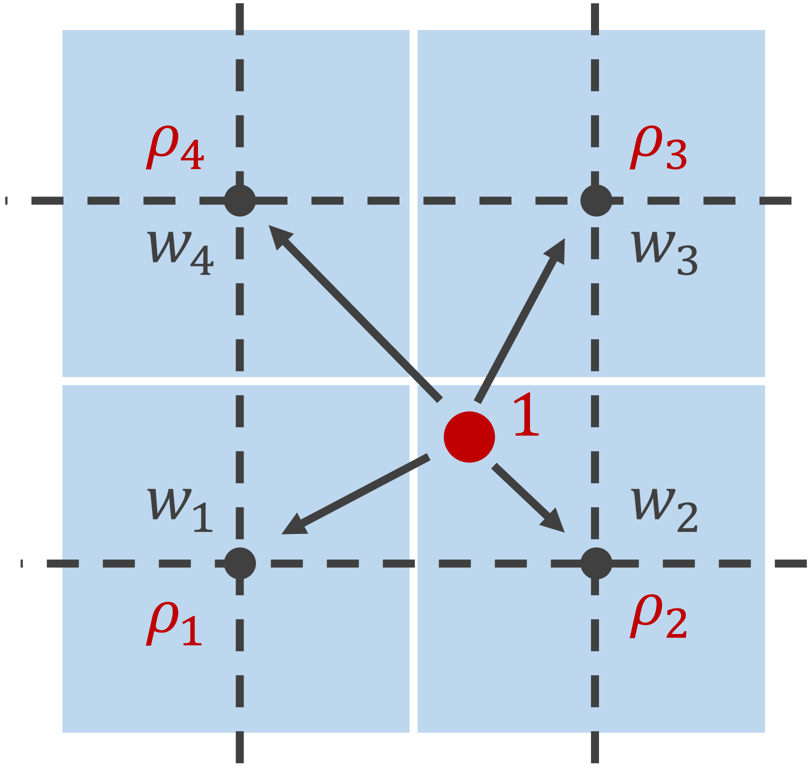
- Compute a particle density d at the center of each cell

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!

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)