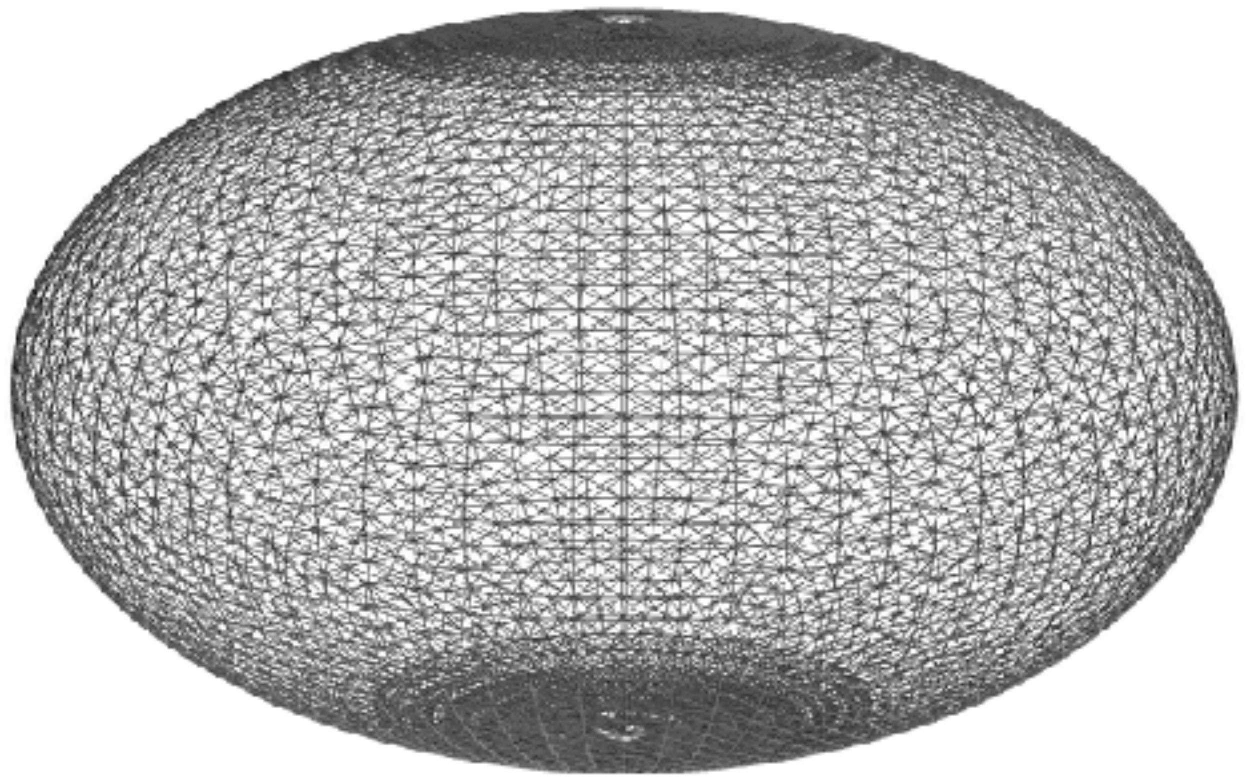
S7260: Microswimmers on Speed: Simulating Spheroidal Squirmers on GPUs

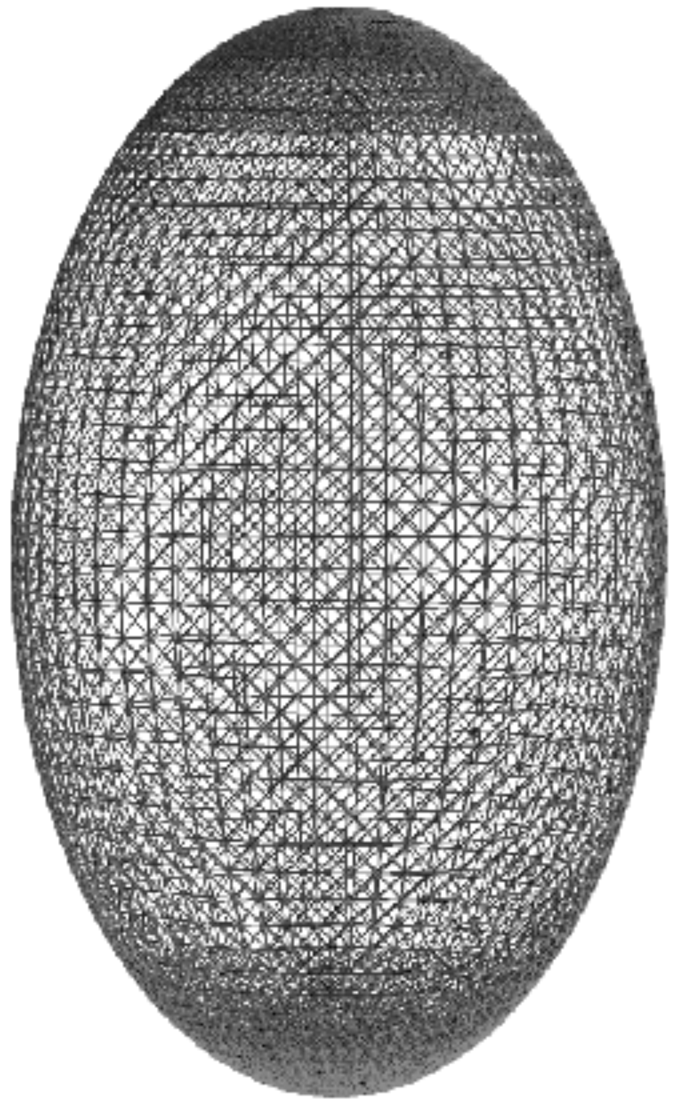
Elmar Westphal - Forschungszentrum Jülich GmbH

Spheroids

Spheroid: A volume formed by rotating an ellipse around one of its axes



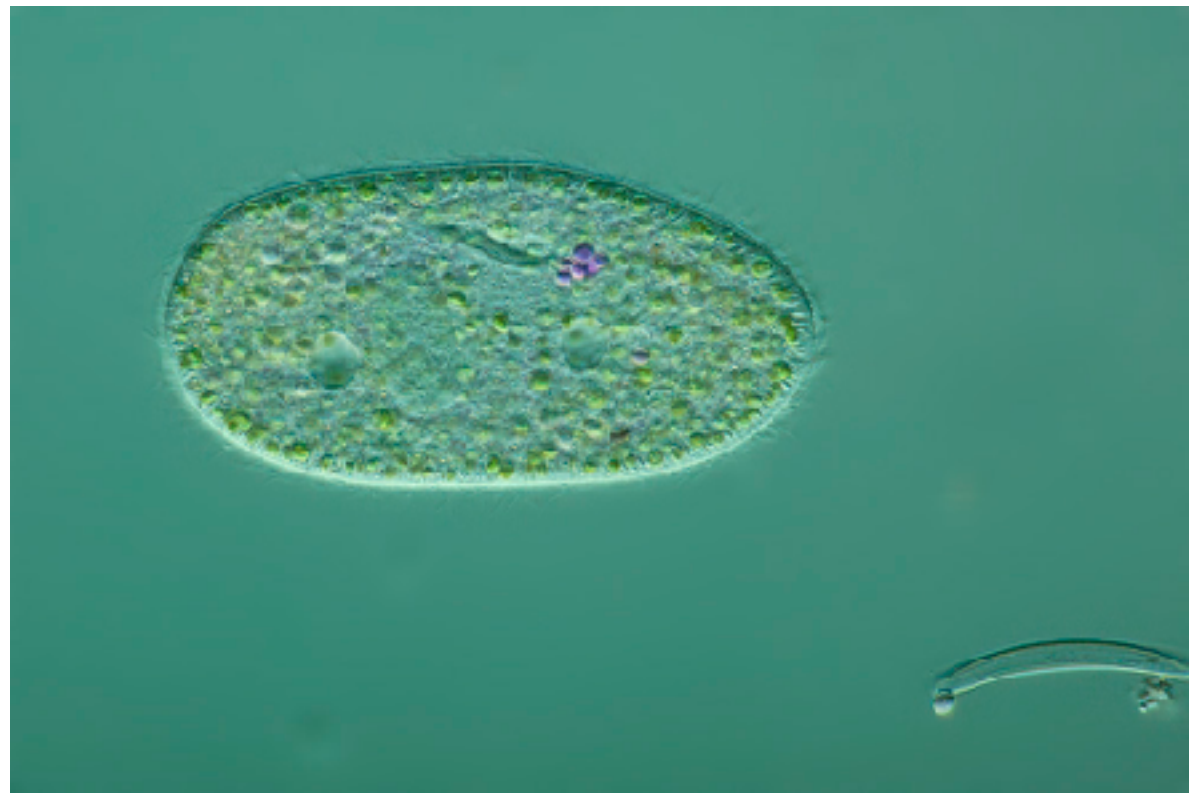
- Two kinds:
  - oblate (rotated around its shorter axis), like a pumpkin or teapot
  - prolate (rotated around its longer axis), like an American football



Squirmers

- Squirmer is a model for simulating micro swimmers
  - Model origins date back to the 1950s
  - One of today’s standard models for self propelled swimmers
  - Can use different means to swim (flagella, arm-like structures etc.), here we simulate the flow caused by surfaces covered with short cilia (filaments)

Example: Paramecium bursaria



Ein grünes Pantoffeltierchen
Frank Fox / www.mikro-foto.de

The Simulation

Our simulation is split into three parts:

- Movement of the squirmers and interactions between them
- Simulation of the liquid
- Interactions between the squirmers and the liquid

Simulation of the Squirmers

JÜLICH FORSCHUNGSZENTRUM

- Number of squirmers is low to moderate (up to a few 1000)
  - Simulation of the squirmers and their interactions is sufficiently fast on CPU
- This may change for future projects

Simulation of the Fluid

- We use discrete fluid particles and an algorithm called “Multi-Particle Collision Dynamics” (MPC) to simulate the fluid
  - MPC inherently conserves energy and momentum of the fluid
  - The phenomena we want to study also require:
    - Conservation of the angular momentum of the fluid particles
      - Adding this roughly doubles the computational effort
    - Walls at one dimension of our system to form a slit
      - This requires additional ghost particles
  - A sufficiently large simulation box for a moderate number of squirmers contains millions of fluid particles

Simulation of the Fluid

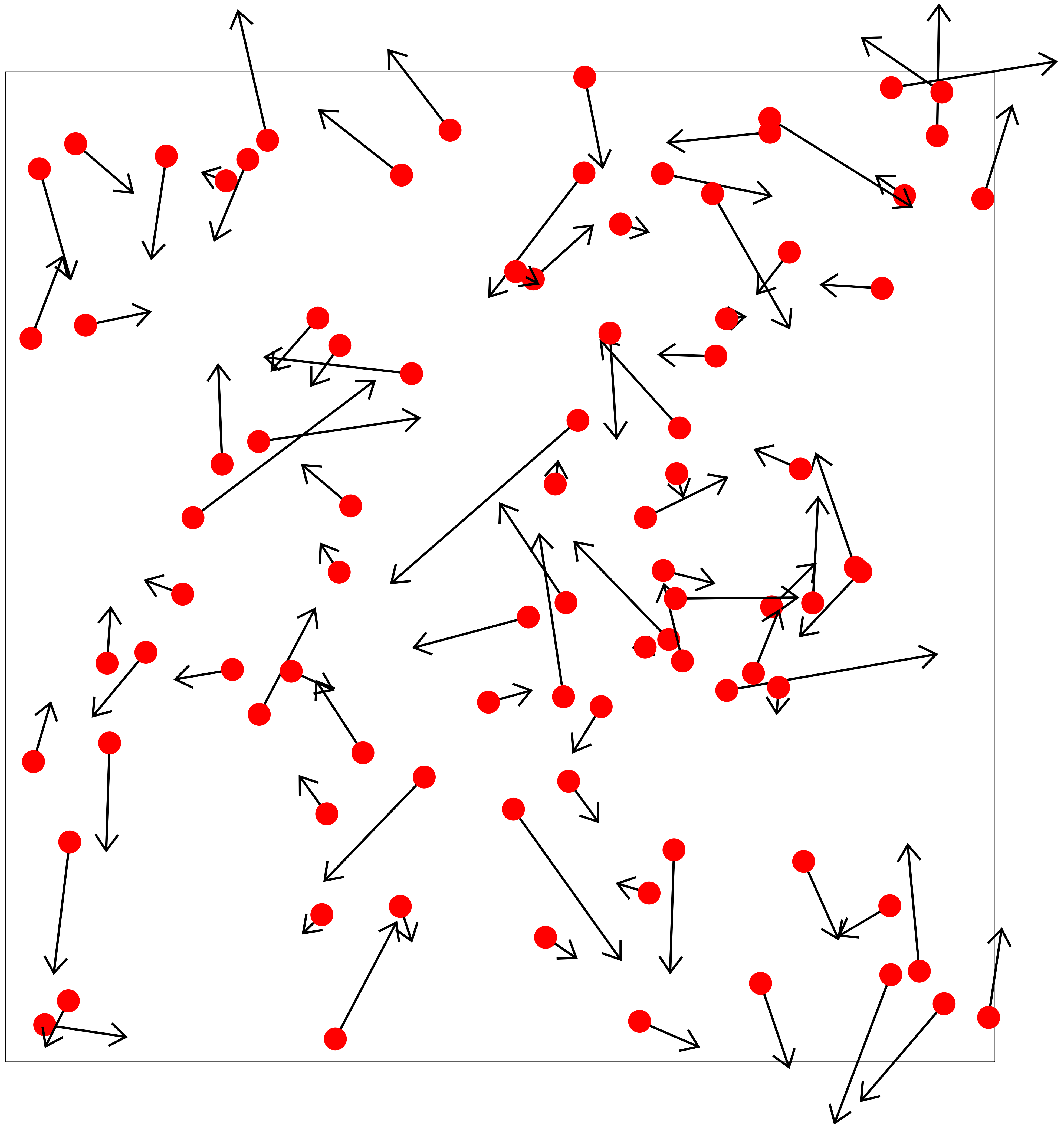
- We use discrete fluid particles and an algorithm called “Multi-Particle Collision Dynamics” (MPC) to simulate the fluid
  - MPC inherently conserves energy and momentum of the fluid
  - The phenomena we want to study also require:
    - Conservation of the angular momentum of the fluid particles
      - Adding this roughly doubles the computational effort
    - Walls at one dimension of our system to form a slit
      - This requires additional ghost particles
  - A sufficiently large simulation box for a moderate number of squirmers contains millions of fluid particles

GPU to the Rescue!

Please note that our simulation is in fact a 3-dimensional system.
To explain the algorithms, 2D-drawings are used for the sake of simplicity.

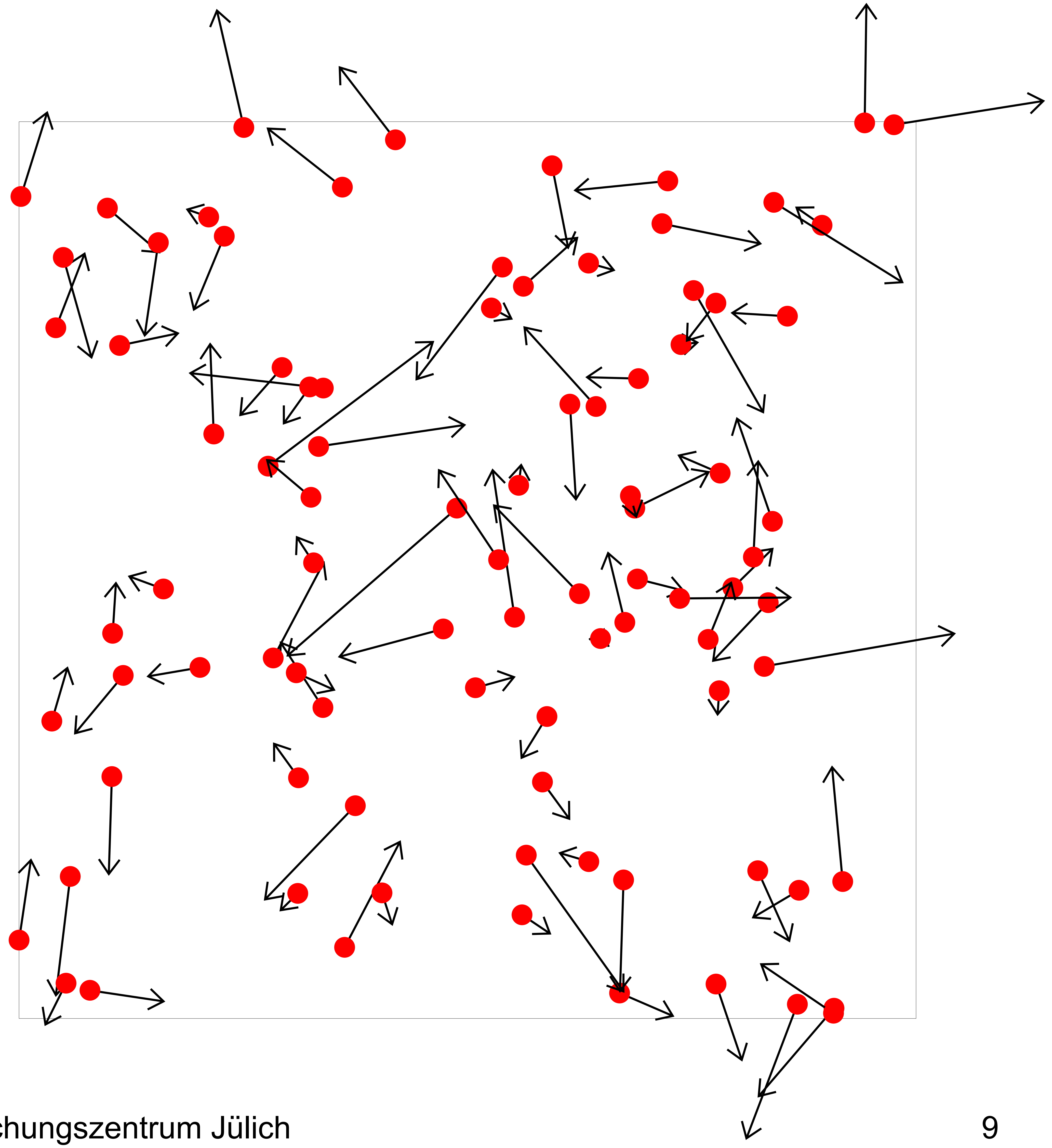
Multi-particle Collision Dynamics (MPC)

- Fluid particles



Multi-particle Collision Dynamics (MPC)

- Fluid particles are moved ballistically

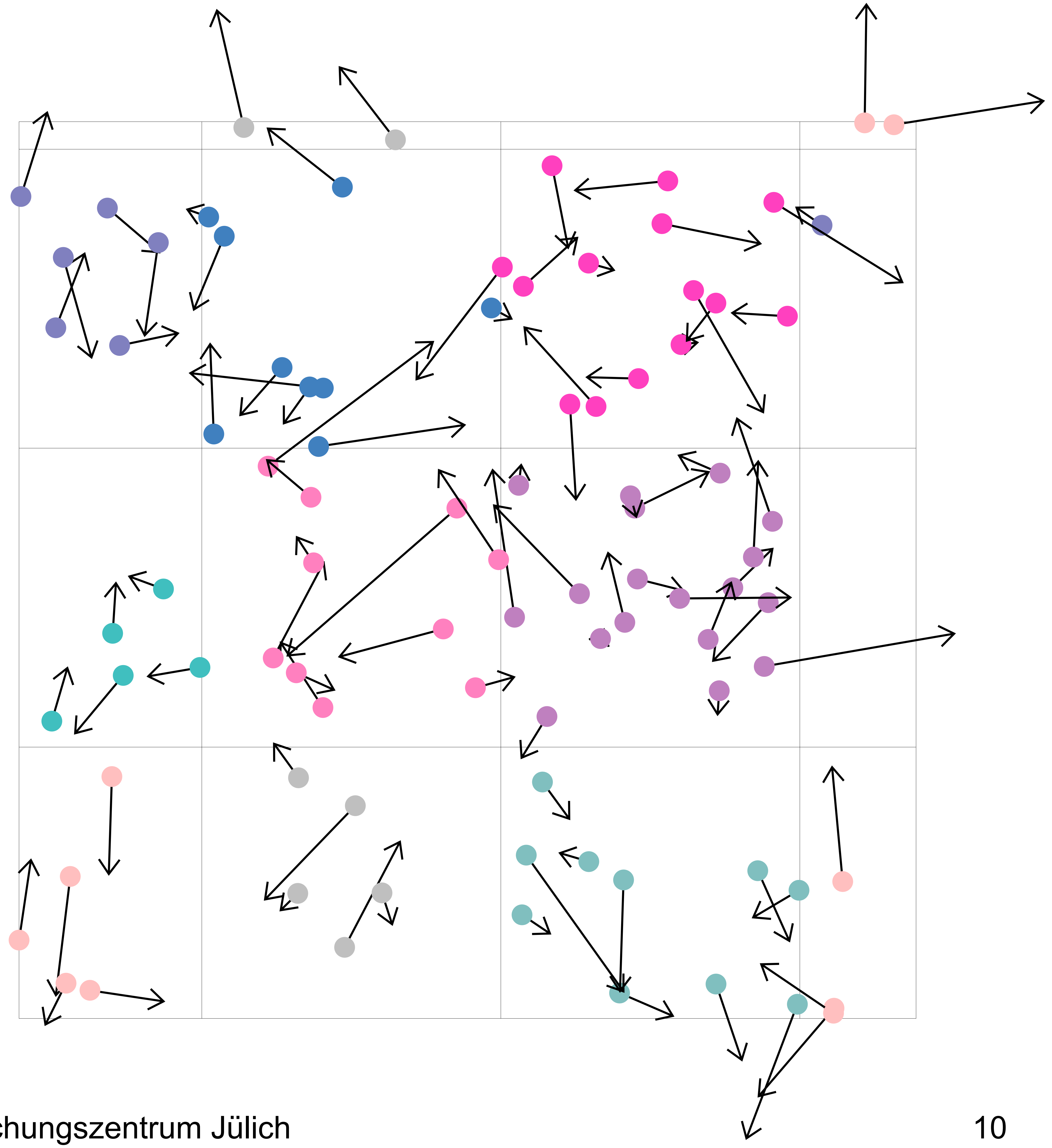


One thread per particle, memory-bound

Multi-particle Collision Dynamics (MPC)

JÜLICH FORSCHUNGSZENTRUM

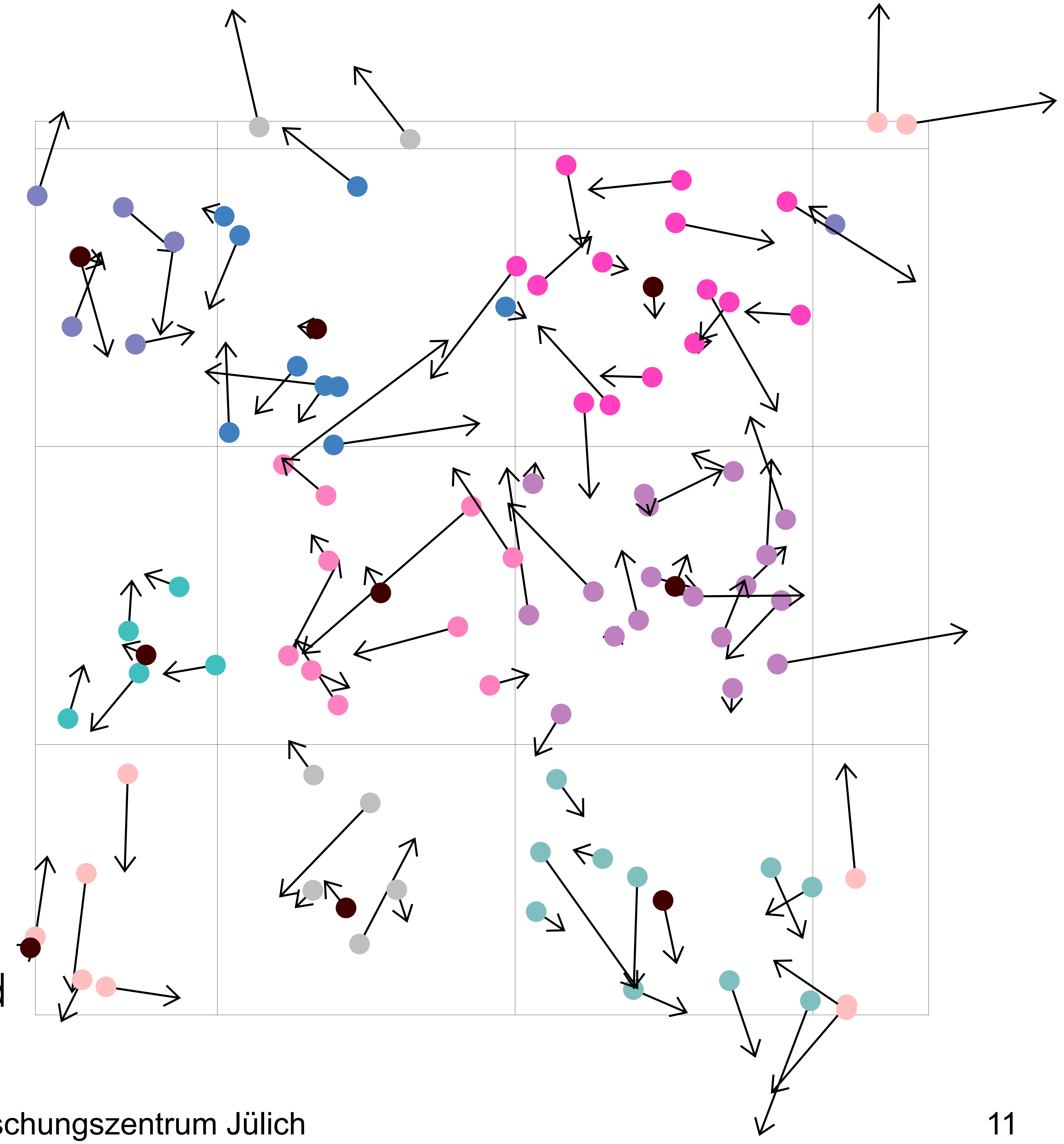
- Fluid particles are moved ballistically and sorted into cells of a randomly shifted grid



One thread per particle, memory-bound

Multi-particle Collision Dynamics (MPC)

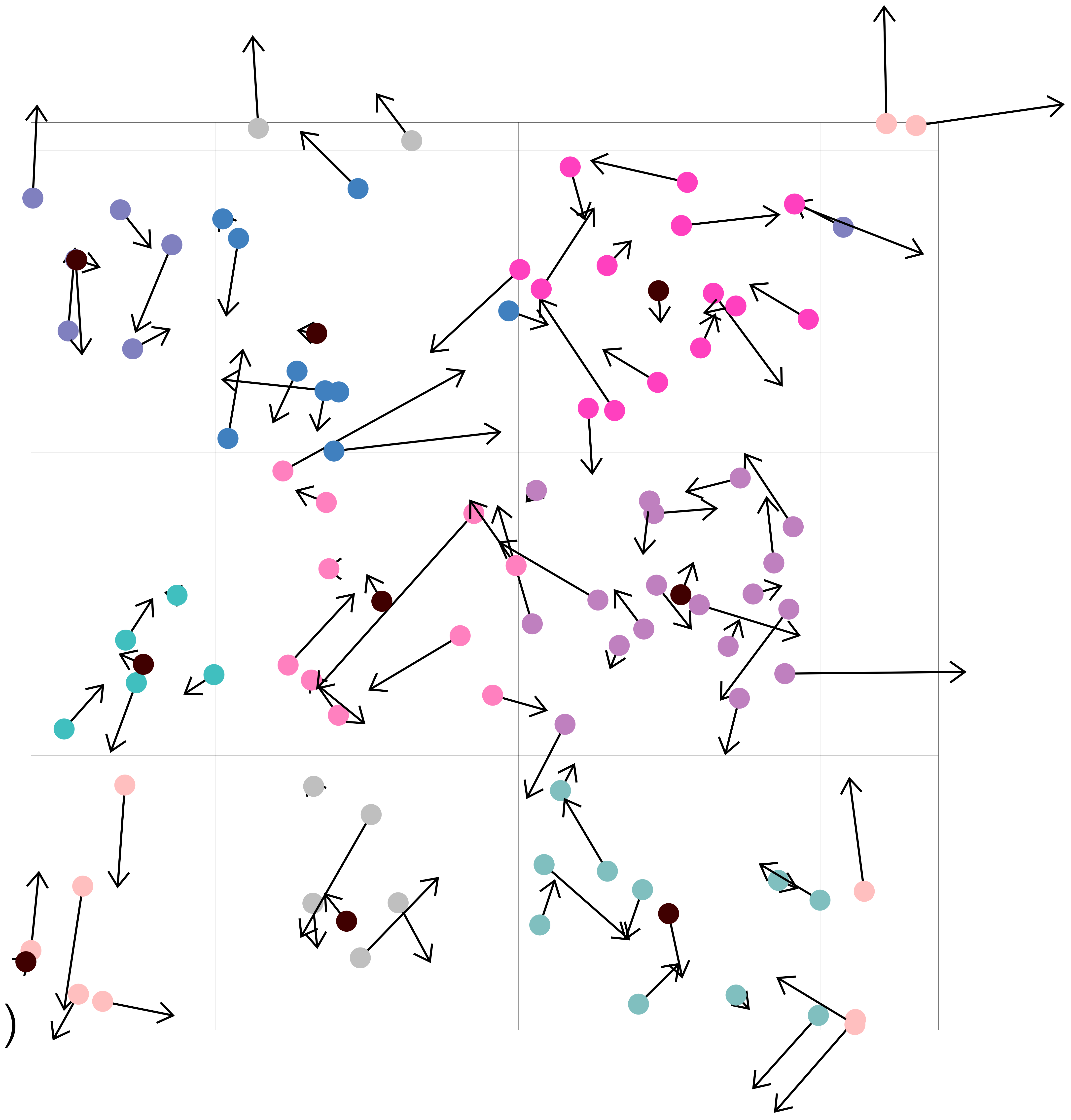
- Fluid particles are moved ballistically and sorted into cells of a randomly shifted grid
- For each cell the centre of mass, centre of mass velocity, kinetic energy and angular momentum are computed



One thread per particle, atomic-bound,
then one thread per cell, memory-bound

Multi-particle Collision Dynamics (MPC)

- Fluid particles are moved ballistically and sorted into cells of a randomly shifted grid
- For each cell the centre of mass, centre of mass velocity, kinetic energy and angular momentum are computed
- Relative velocities are calculated

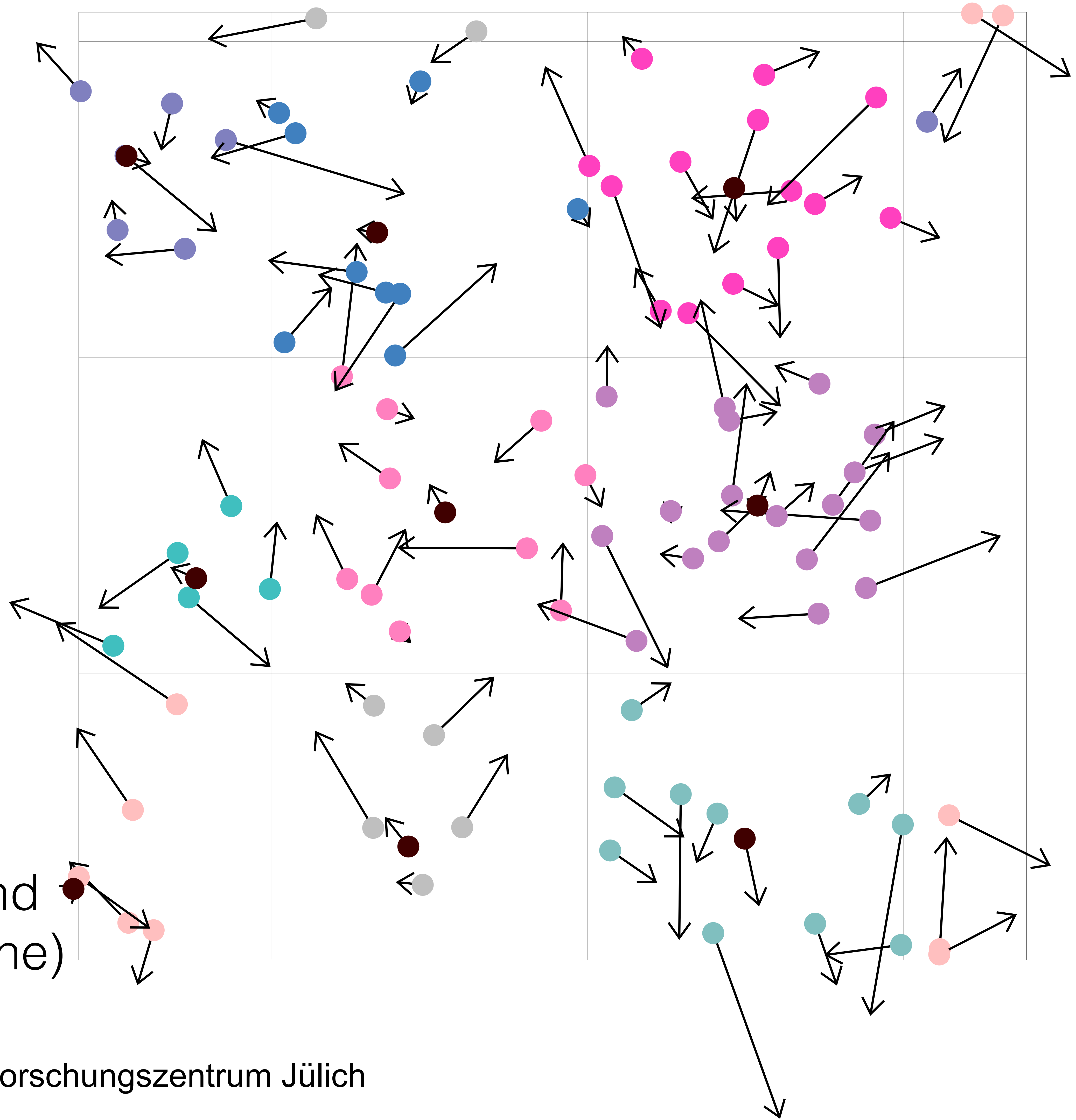


One thread per particle, memory-bound by random cell-data reads (texture cache)

Multi-particle Collision Dynamics (MPC)

JÜLICH FORSCHUNGSZENTRUM

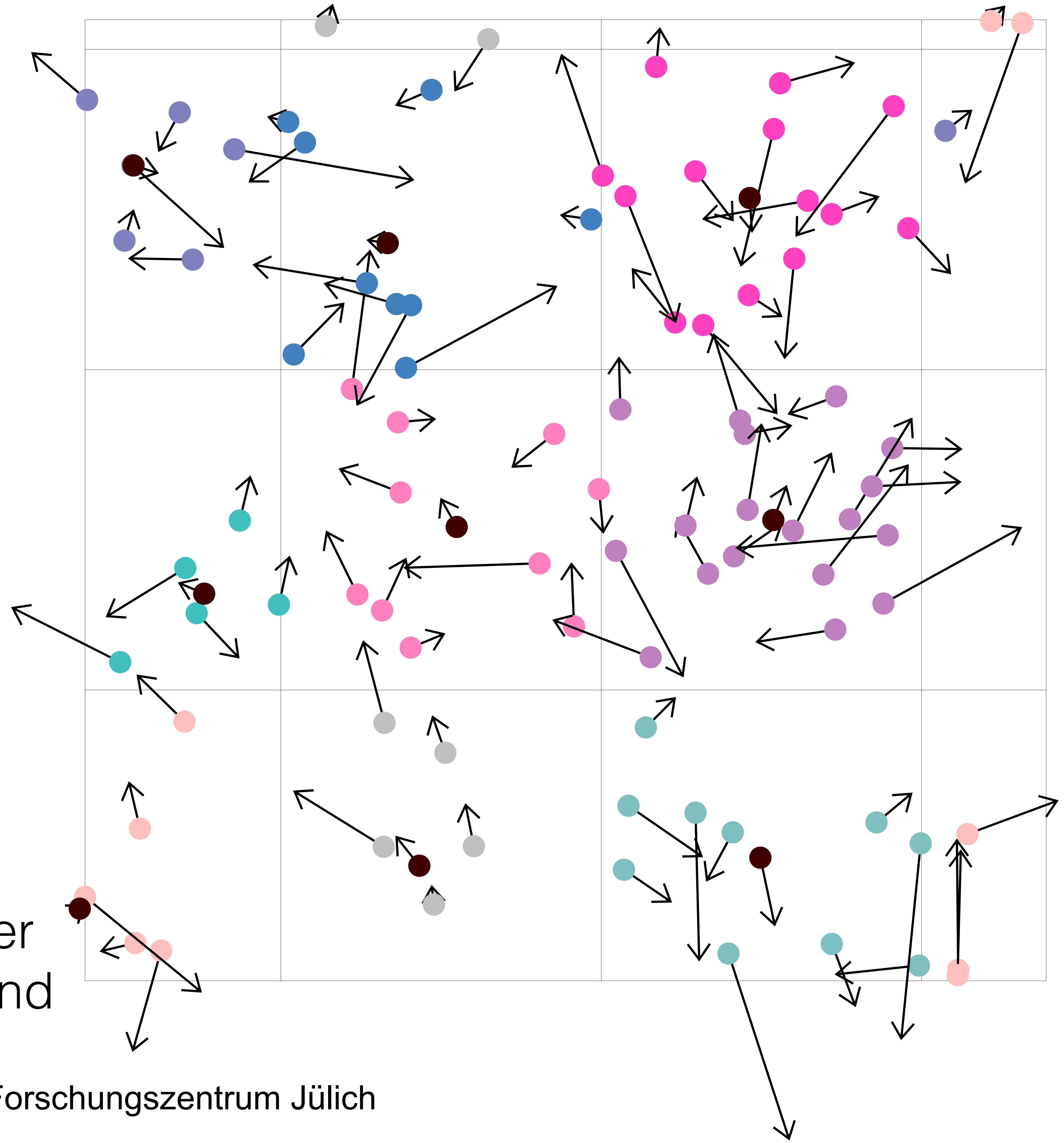
- Fluid particles are moved ballistically and sorted into cells of a randomly shifted grid
- For each cell the centre of mass, centre of mass velocity, kinetic energy and angular momentum are computed
- Relative velocities are calculated and rotated around a random axis



One thread per particle, memory-bound by random cell-data reads (texture cache)

Multi-particle Collision Dynamics (MPC)

- Fluid particles are moved ballistically and sorted into cells of a randomly shifted grid
- For each cell the centre of mass, centre of mass velocity, kinetic energy and angular momentum are computed
- Relative velocities are calculated and rotated around a random axis
- Angular momentum and kinetic energy need to be restored



several steps using one thread per particle or cell, mostly atomic-bound

MPC on GPU

- Is limited by speed of atomic operations and memory bandwidth
- Implementation uses optimisations as described in some of my earlier GTC talks
  - Reordering of particles to preserve data locality (S2036*)
  - Reducing the number of atomic operations (S5151)

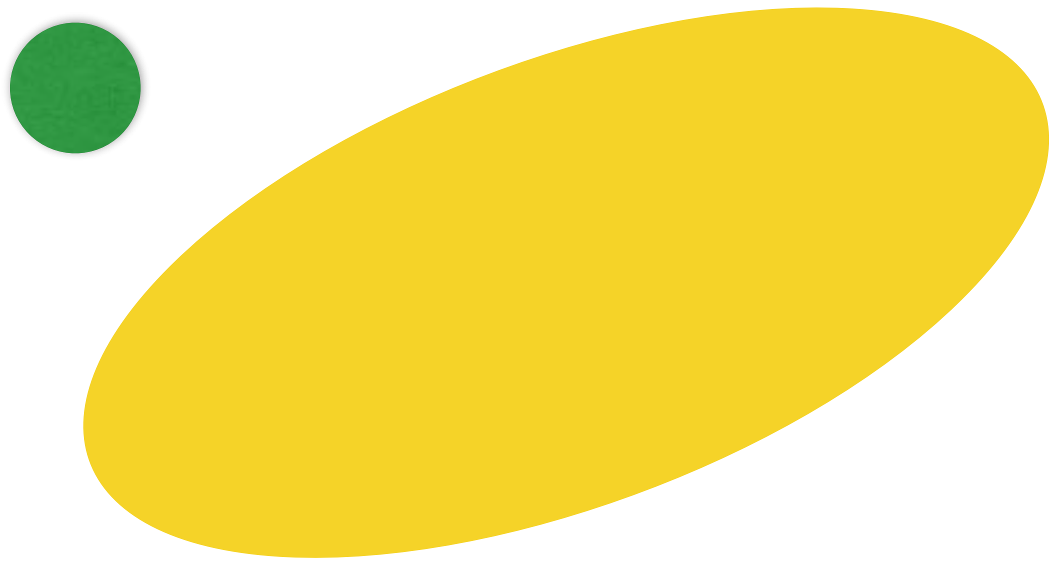
*the speed of atomic operations has improved significantly over time, so parts of the implementation described here have been abandoned

Interactions between Fluid and Squirmers

- Squirmer surfaces are considered impenetrable for the fluid and are therefore boundaries for the fluid particles
  - Collisions have to be detected
    - Their impact has to be combined and applied to the squirmer and fluid particles accordingly
      - This happens on the GPU (large number of fluid particles),
    - The total impact for each squirmer is passed to and processed by the CPU (low number of squirmers)

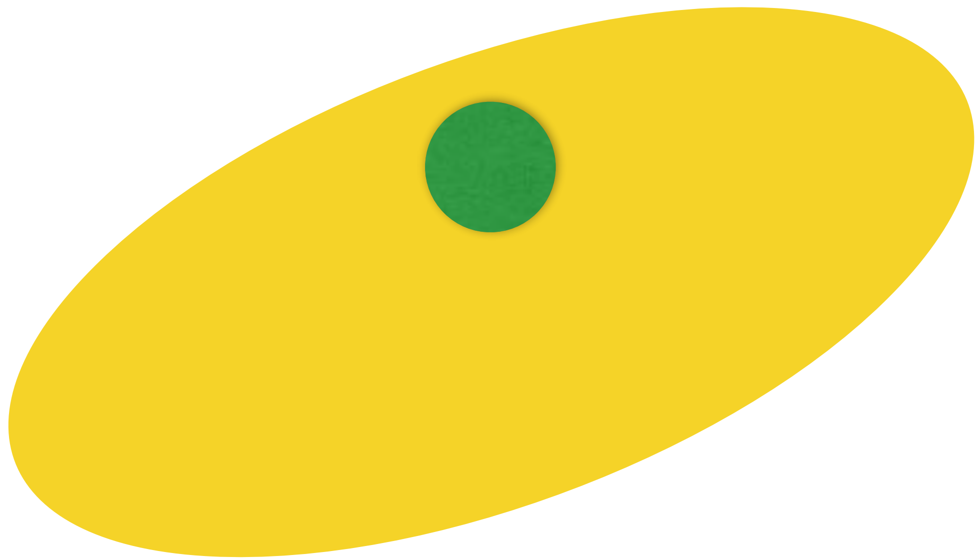
Collisions Between Fluid Particles and Squirmers

- Fluid particles and Squirmers move during each time-step



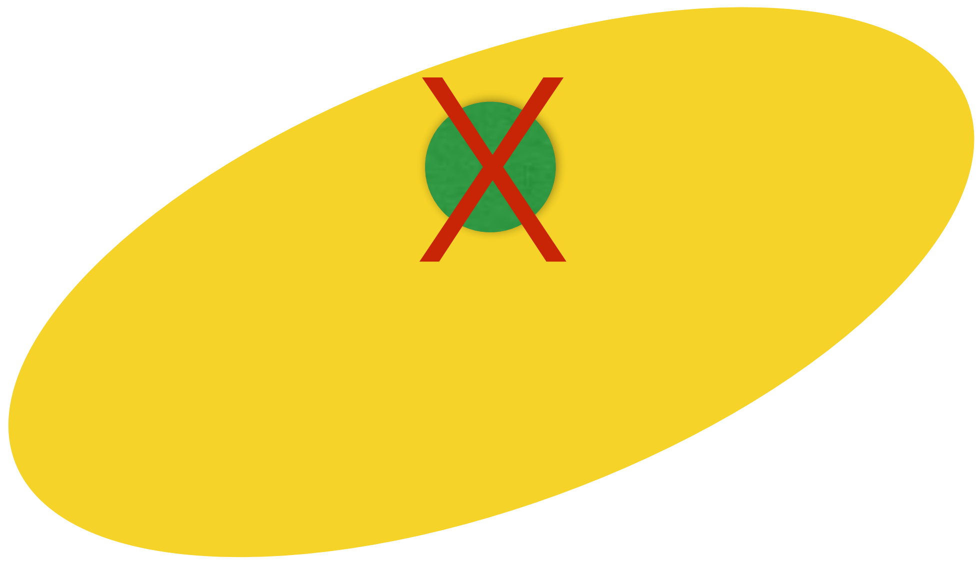
Collisions Between Fluid Particles and Squirmers

- Fluid particles and Squirmers move during each time-step



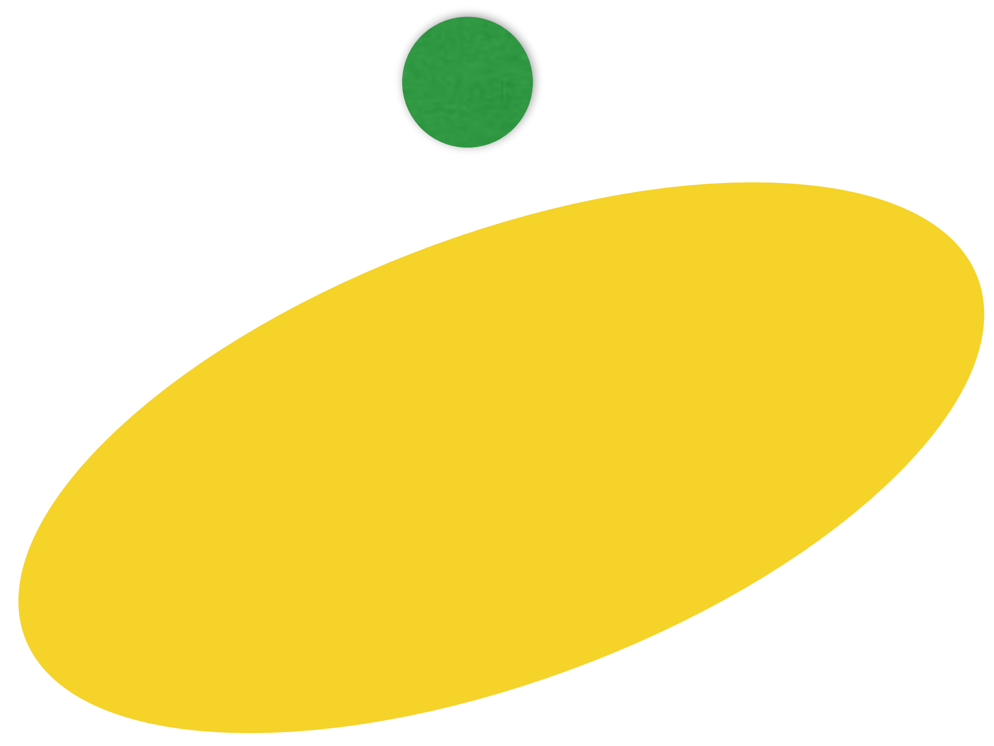
Collisions Between Fluid Particles and Squirmers

- Fluid particles and Squirmers move during each time-step
- Squirmer walls are considered impenetrable



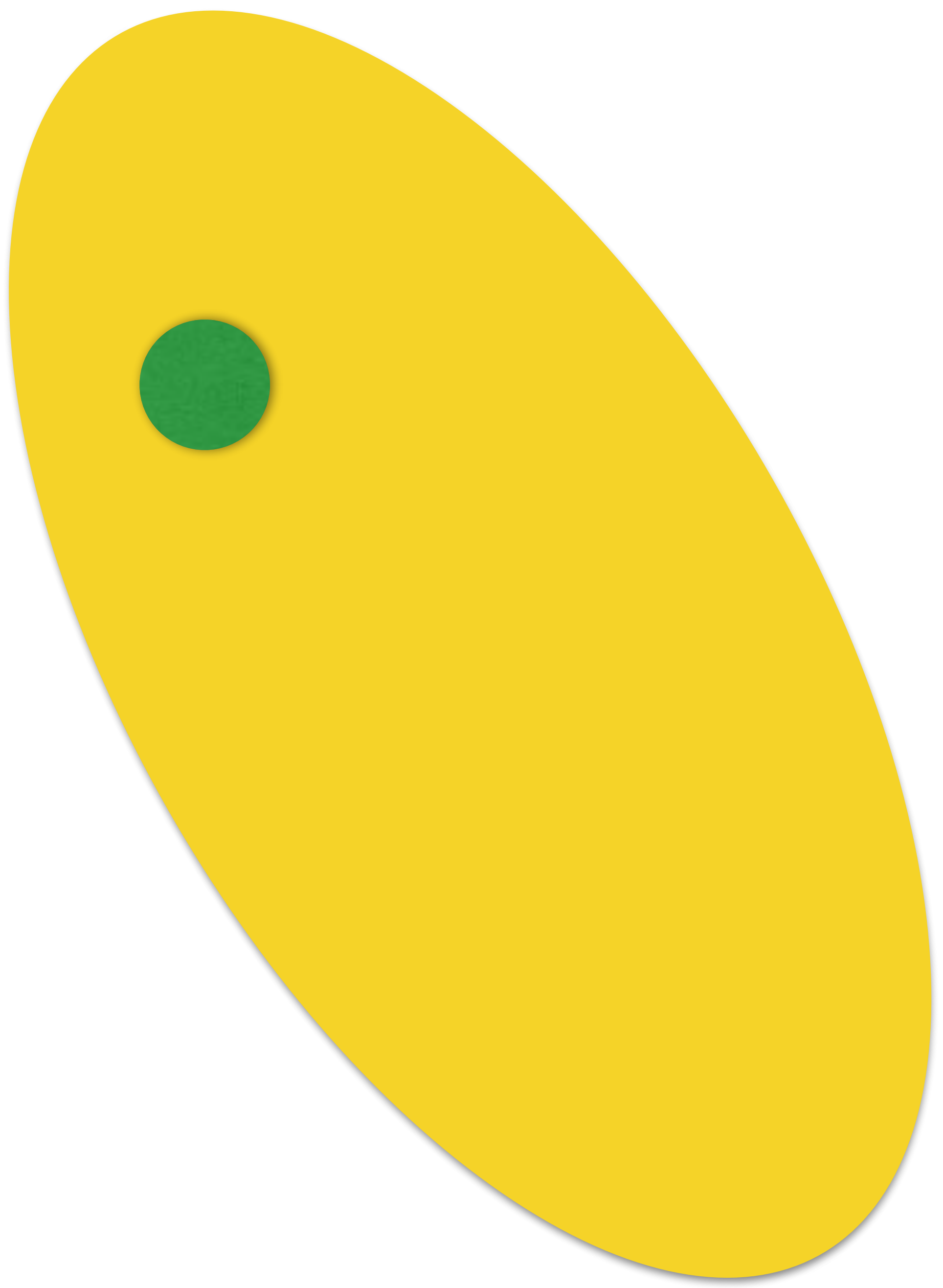
Collisions Between Fluid Particles and Squirmers

- Fluid particles and Squirmers move during each time-step
- Squirmer walls are considered impenetrable
- Fluid particles entering the squirmer have to be dealt with accordingly



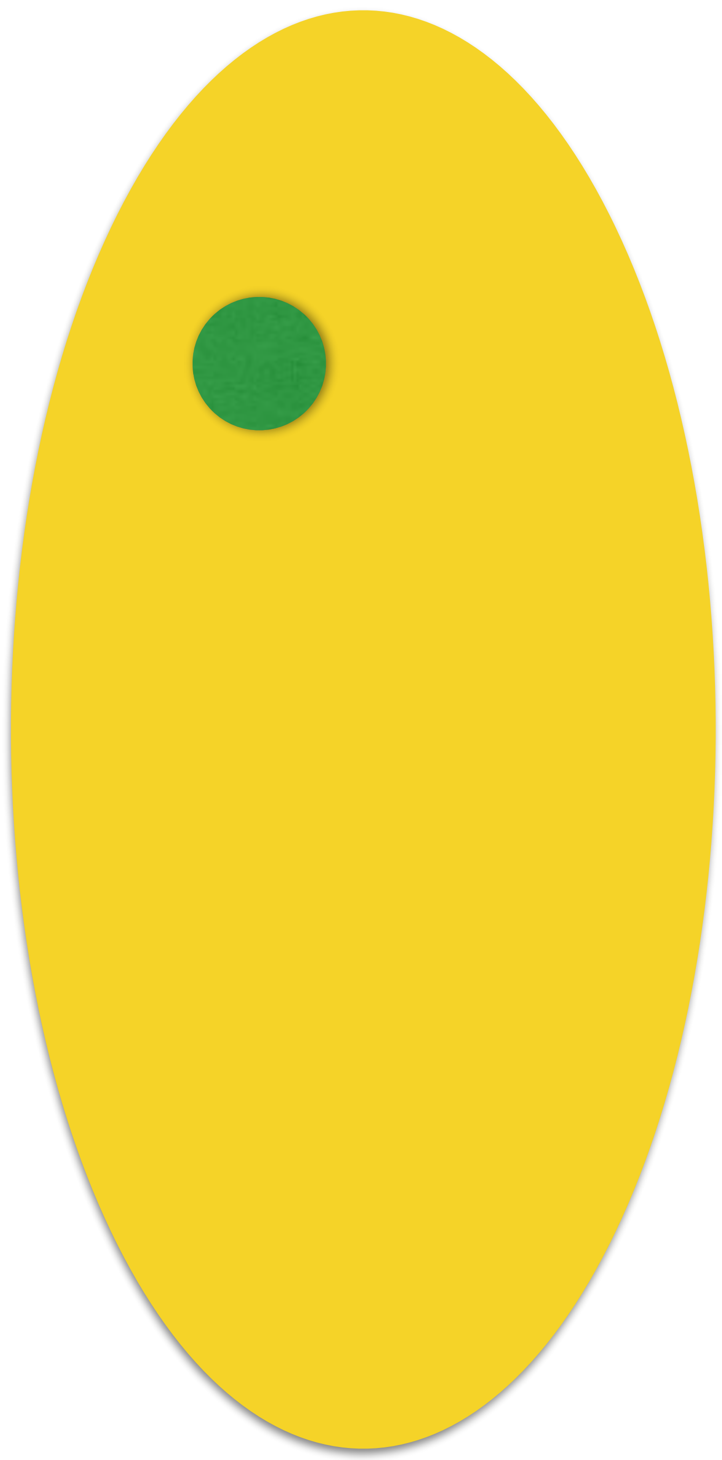
Collisions Between Fluid Particles and Squirmers

- Detecting penetration requires rotating particles into the squirmers frame of reference and scaling according to its ratio
- Checking every fluid particle against every squirmer is too much work, even for a GPU



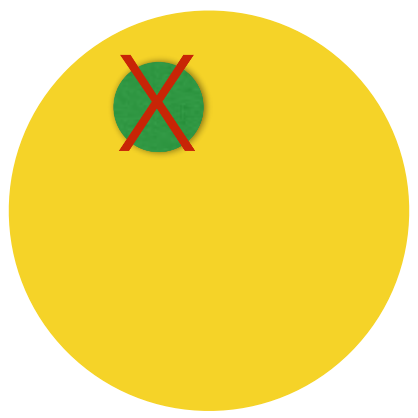
Collisions Between Fluid Particles and Squirmers

- Detecting penetration requires rotating particles into the squirmers frame of reference and scaling according to its ratio
- Checking every fluid particle against every squirmer is too much work, even for a GPU

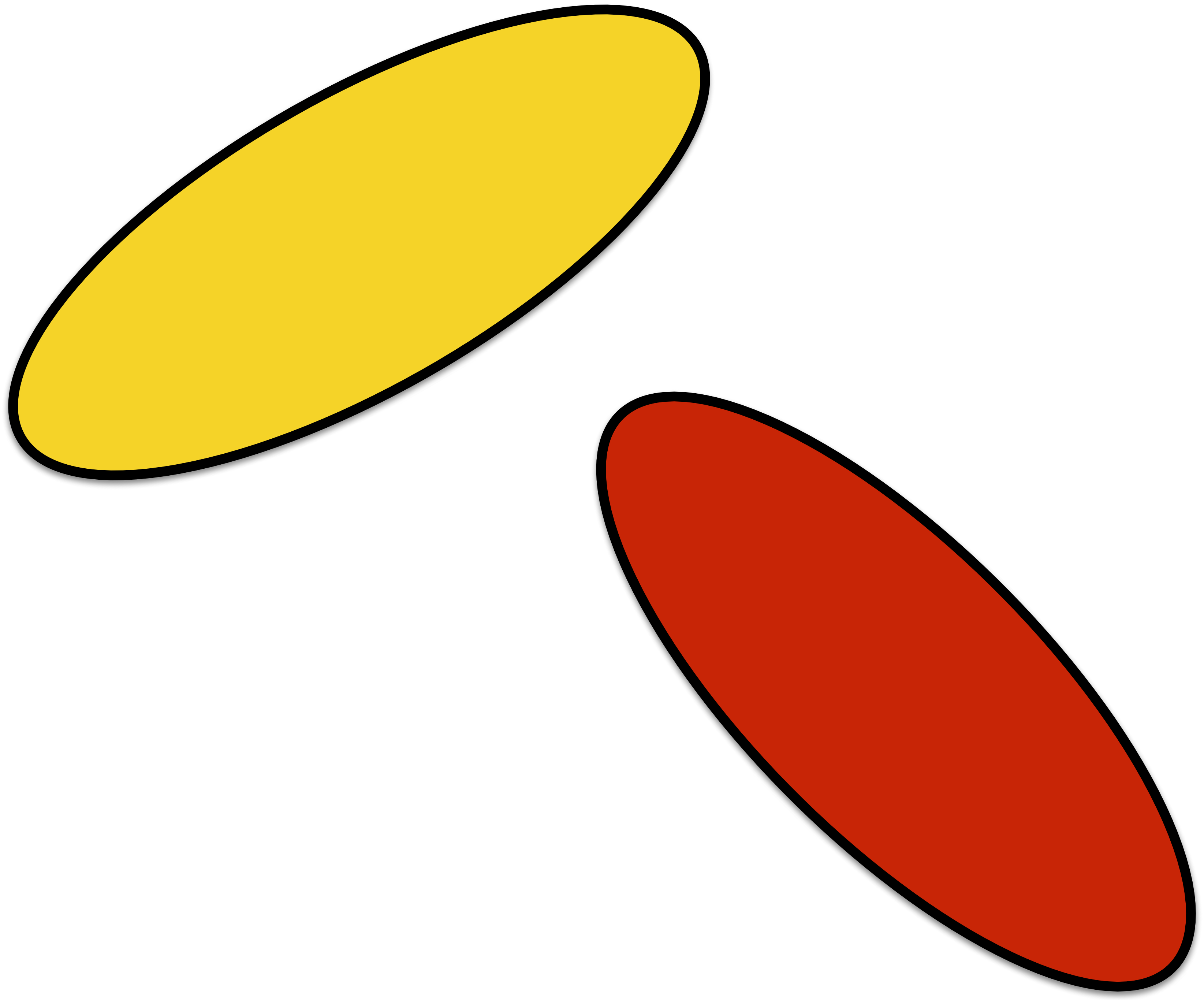


Collisions Between Fluid Particles and Squirmers

- Detecting penetration requires rotating particles into the squirmers frame of reference and scaling according to its ratio
- Checking every fluid particle against every squirmer is too much work, even for a GPU

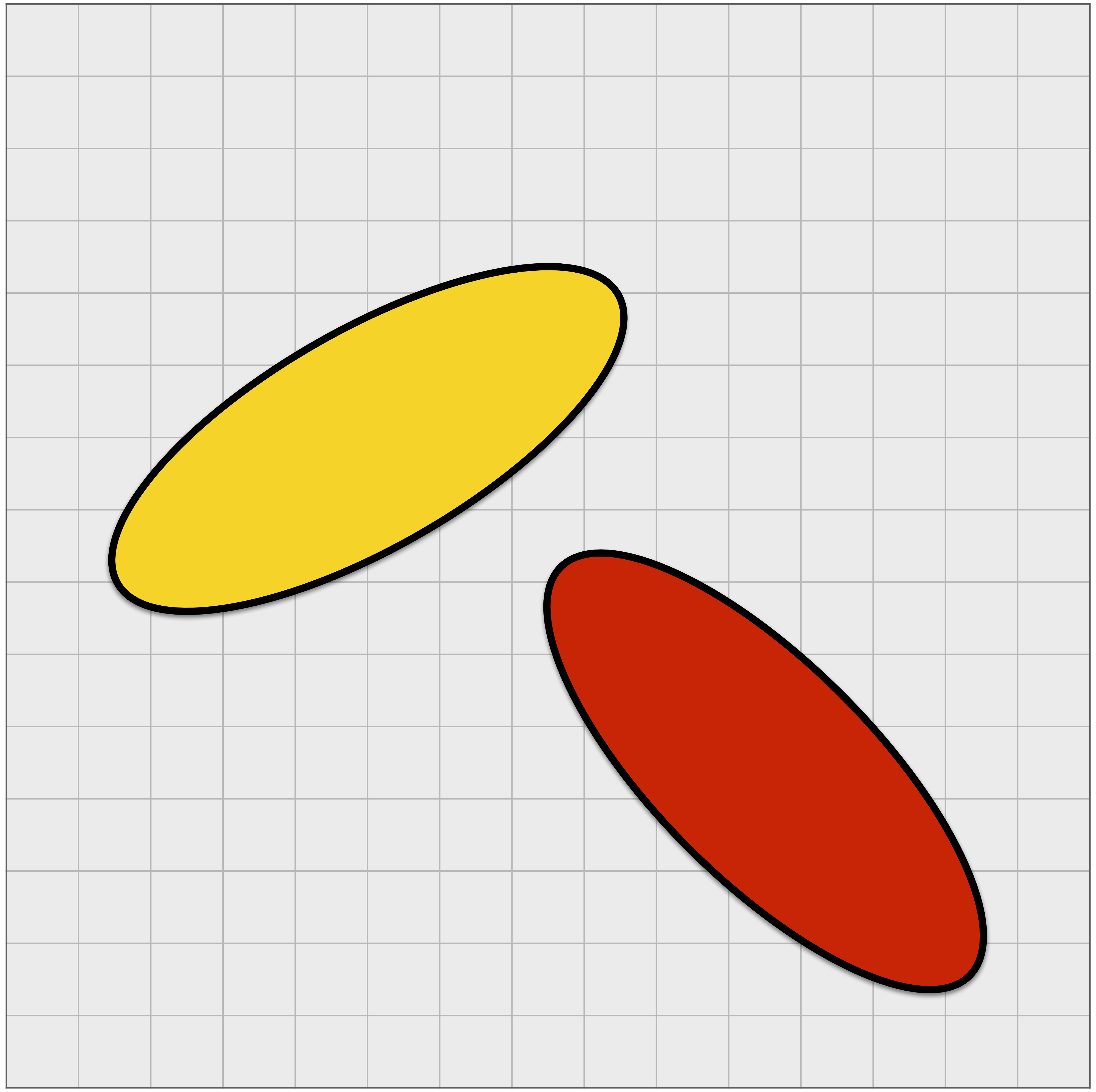


Handling Squirmer-Fluid Collisions



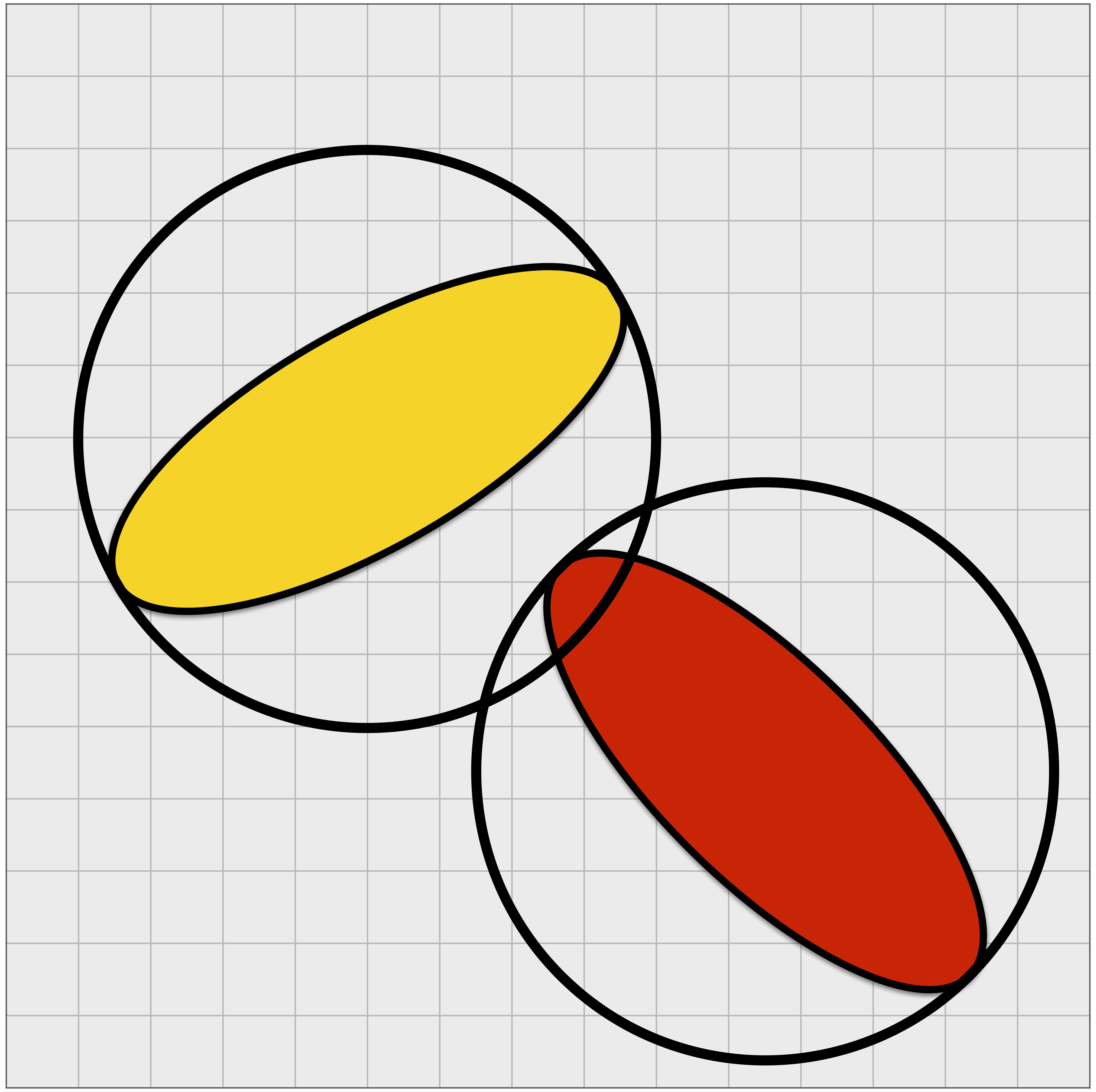
Handling Squirmer-Fluid Collisions

- The system is divided into cells



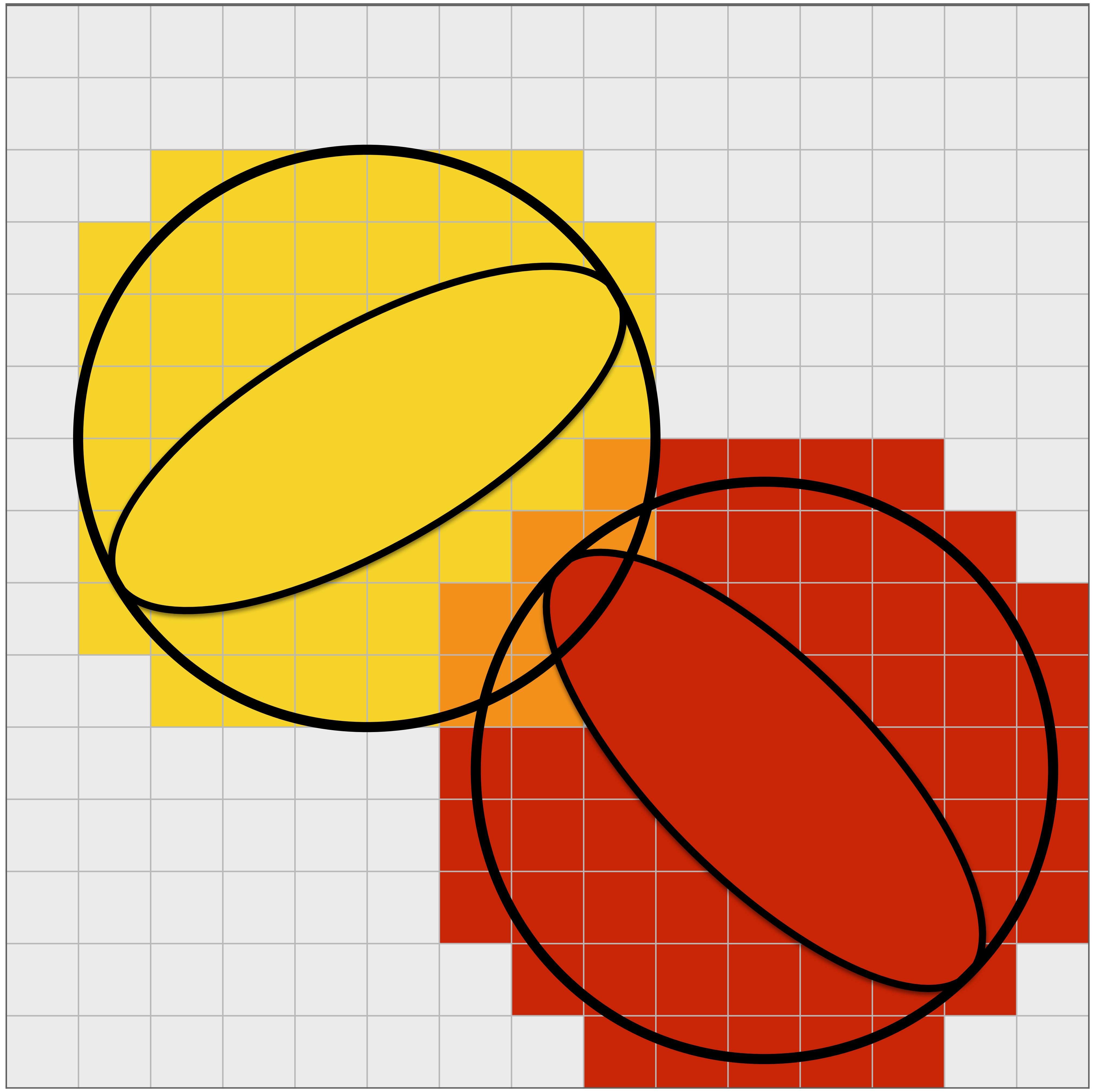
Handling Squirmer-Fluid Collisions

- The system is divided into cells
- Regardless of its orientation, a spheroid can not exceed a sphere matching its centre and largest radius



Handling Squirmer-Fluid Collisions

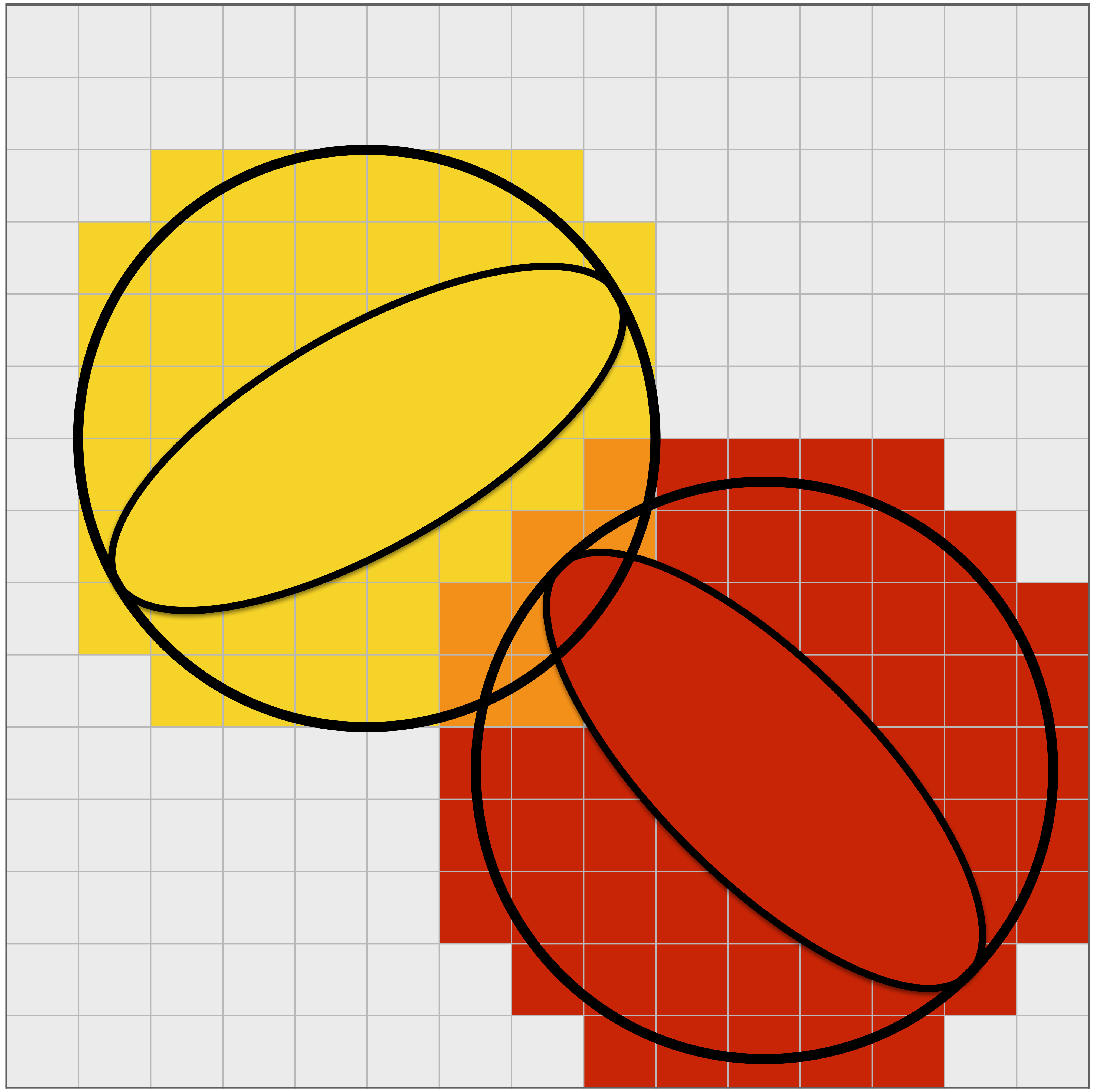
- The system is divided into cells
- Regardless of its orientation, a spheroid can not exceed a sphere matching its centre and largest radius
- Each squirmer has a list of the cells affected by its surrounding sphere (they may overlap)



Mitglied der Helmholtz-Gemeinschaft

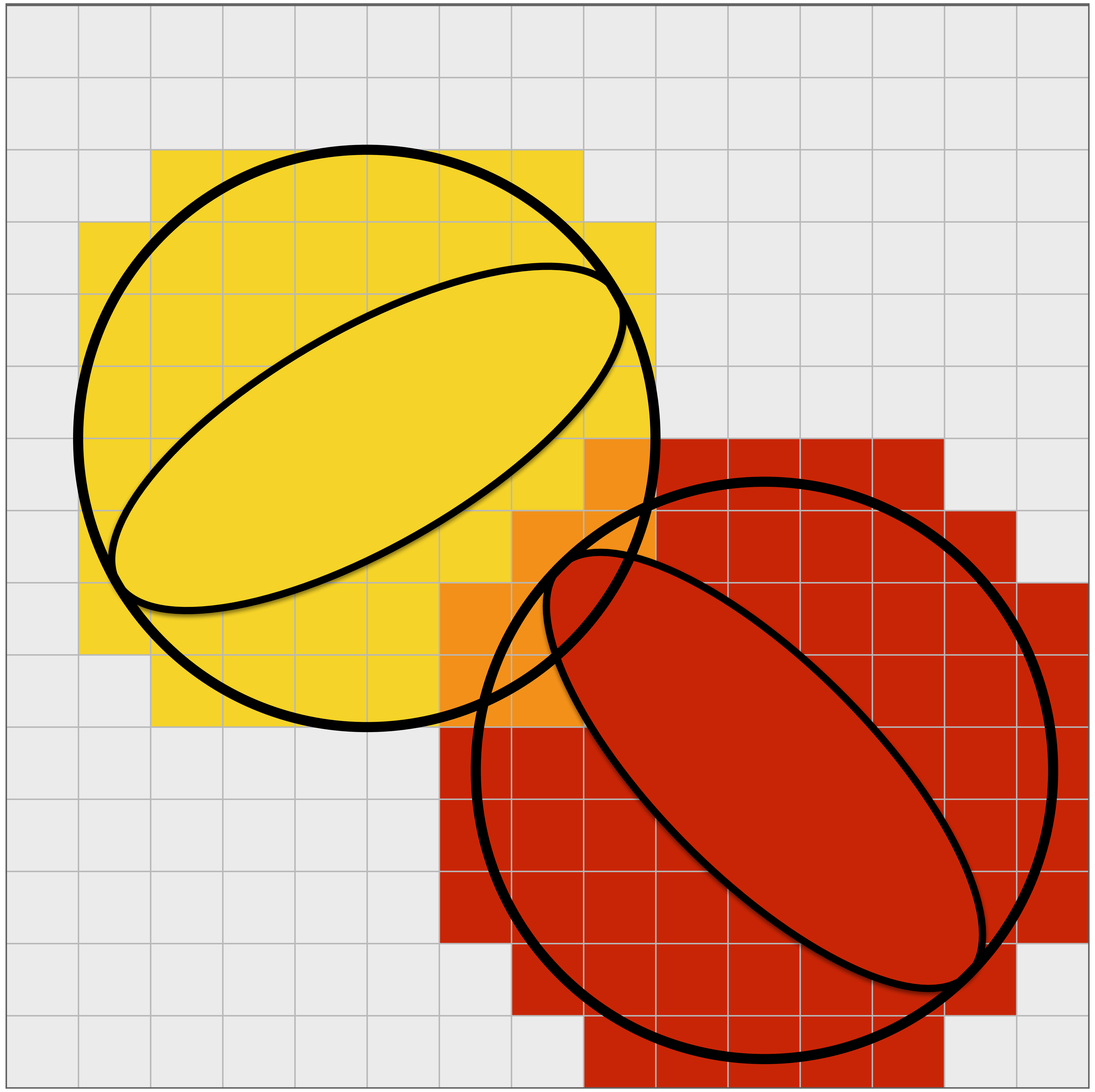
Handling Squirmer-Fluid Collisions

- The system is divided into cells
- Regardless of its orientation, a spheroid can not exceed a sphere matching its centre and largest radius
- Each squirmer has a list of the cells affected by its surrounding sphere (they may overlap)
- Each cell has a list of the fluid particles it contains



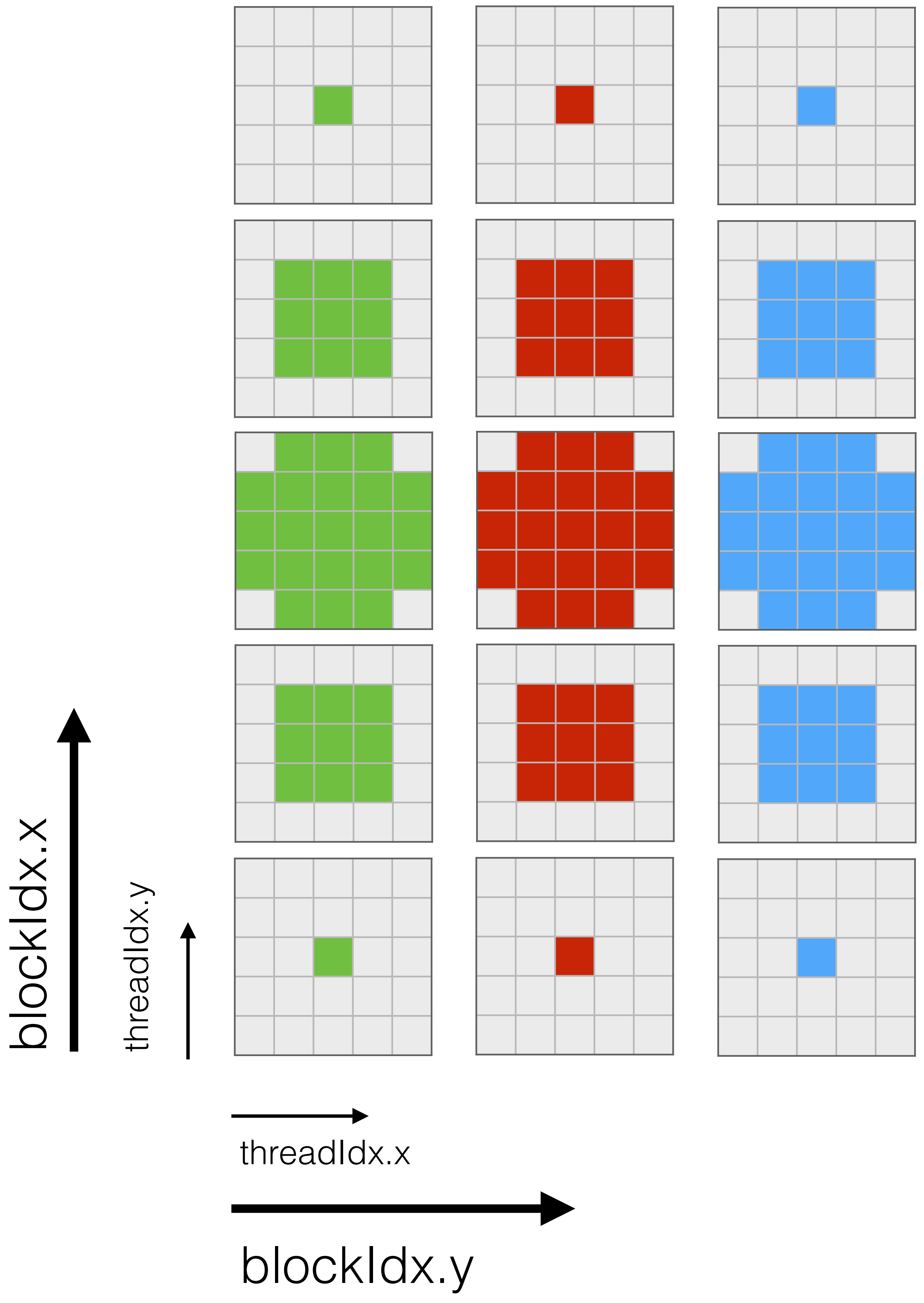
Handling Squirmer-Fluid Collisions

- The system is divided into cells
- Regardless of its orientation, a spheroid can not exceed a sphere matching its centre and largest radius
- Each squirmer has a list of the cells affected by its surrounding sphere (they may overlap)
- Each cell has a list of the fluid particles it contains
- This reduces the number of checks from N_{sq} x N_{fluid} to ~N_{sq} x V_{sq}* ρ



Handling Squirmer Surroundings on GPU

- For each squirmer, a cubical box of cells is defined that contains its surrounding sphere
- Cells can be conveniently coded into grid- and block-dimensions:
  - x- and y- directions are coded into threadIdx
  - z-direction is coded into blockIdx.x (limit of 1024 threads per block)
  - Squirmer-index is coded into blockIdx.y
- Affected cells are selected using approximate cell stencils
- Cells are processed one per thread



Handling Squirmer Surroundings on GPU

Threads from the described grid are used to

- Select affected cells
- Check particles from selected cells for collision and
  - Move colliding particles accordingly
  - Sum up collision impact to apply to squirmers
- Generate random fluid particles inside squirmers to preserve physical properties of the system

Performance Pitfalls

JÜLICH FORSCHUNGSZENTRUM

- Squirmers and MPC using walls require volumes filled with random particles
  - Their relative impact depends on:
    - Ratio of surface to height of the simulation box (wall algorithm adds an additional cell in one direction)
    - Number and size of squirmers
  - Lack of data locality in squirmer random particles has a severe impact on the performance of the MPC implementation on GPU:
    - Fewer memory accesses can be combined
    - Atomics optimisations depend on data locality

Computation Time

- Mostly due to the influence of the spatially unordered random particles inside the squirmers, computation times vary significantly with the number of squirmers (measured on Tesla K80):
  - 4.9 ns per iteration and MPC-particle for large systems with 1 squirmer
  - 13.2 ns per iteration and MPC-particle for large systems with 3692 squirmers
- Squirmer interactions done on CPU only ~4% of computation time

General Implementation

- Heavily templated C++-11 code
  - MPC parts works in 2D, 3D, different precisions and with a variety of options, generating optimised code for different applications and GPU architectures
  - Uses a class template for particle sets to allow mixed precisions and features in a single simulation
  - User managed particle sets can be injected into most steps of the process

General Implementation

- Based on a std::vector-like class template using managed memory
  - Replacing the allocator is not enough (members/ operators not defined for device)
  - Easy exchange of data between CPU- and GPU-parts
  - Uses texture caching where applicable
- Operator templates for CUDA vector-types (float4 etc.) make life a lot easier

Memory Considerations

- MPC particles use 52 bytes of memory each
- MPC cells use 160 bytes of memory each (with angular momentum conservation)
- Squirmer data uses about 600 bytes of memory
  - Random particles inside squirmers do not count, because they replace particles of the original fluid
- On recent GPUs, this allows system sizes of up to ~10M cells and ~15K squirmers

Outlook

The next version (currently testing) will definitely feature…

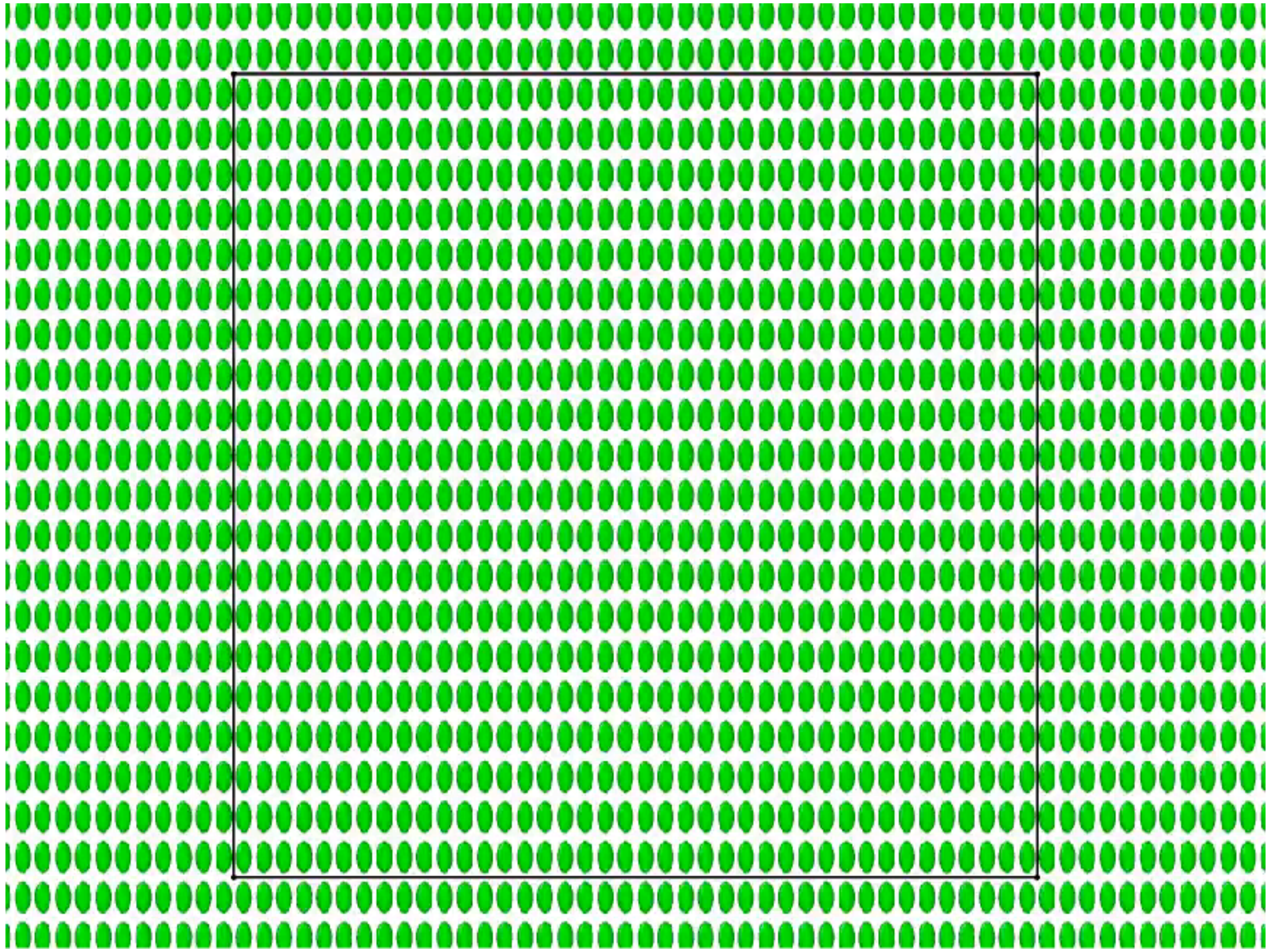
- MPI parallelisation, allowing larger simulations and/or higher speed
- Hybrid code using template magic to generate CPU or GPU based executables from the same sources (real kernels, not pragma-based)
- Fewer restraints through the use of even more templates

… and is prepared for

- Bringing some (spatial) order to the random particle chaos
- Bit-true, reproducible calculations

Example

800 Squirmers in slit,
$L_x=L_z=300$, $L_y=7$,
~7M fluid particles,
20M timesteps,
2-3 weeks walltime



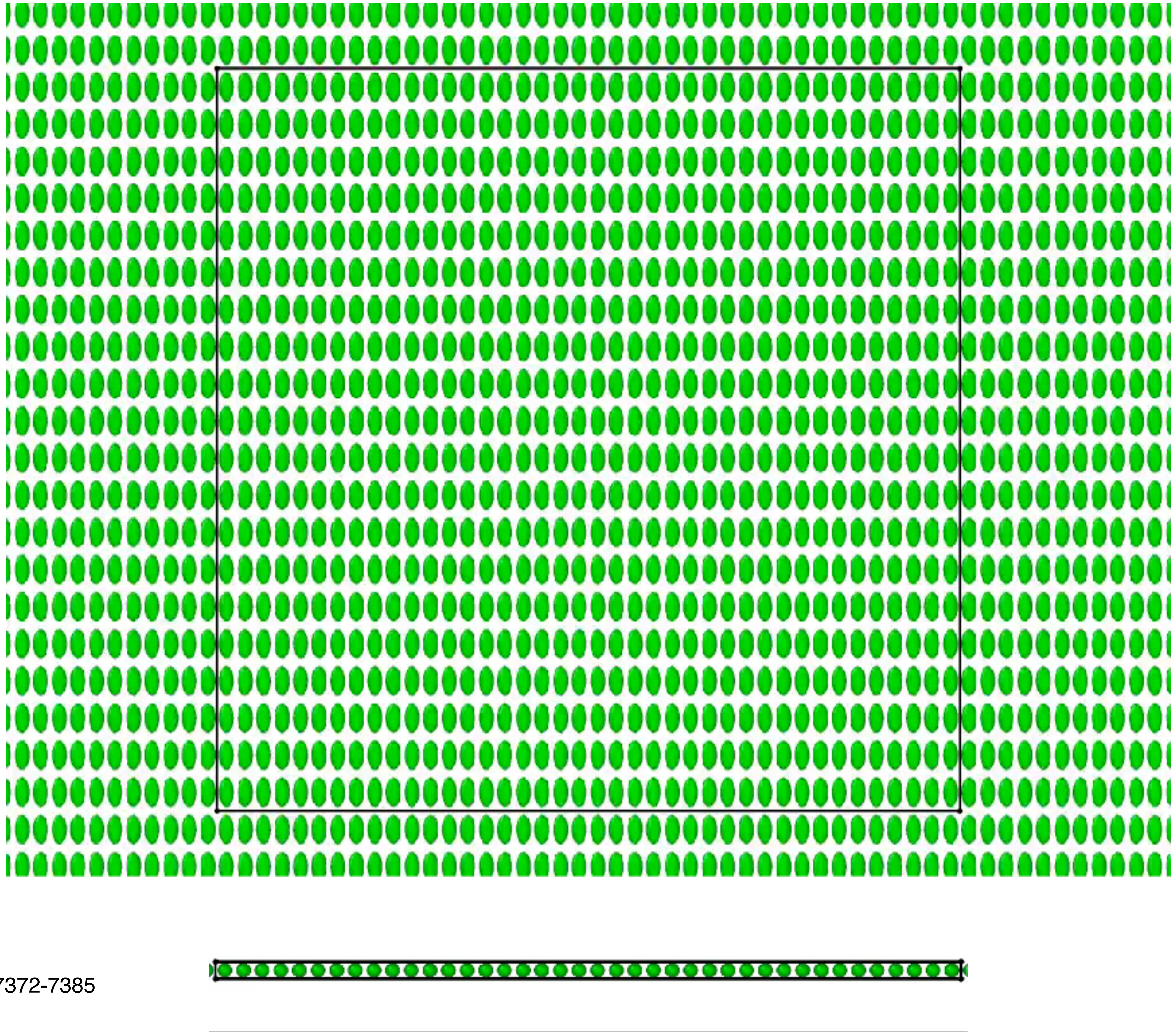
Simulation and rendering courtesy of Mario Theers

Further reading:

DOI: 10.1039/C6SM01424K (Paper) Soft Matter, 2016, 12, 7372-7385

Example

800 Squirmers in slit,
$L_x=L_z=300$, $L_y=7$,
~7M fluid particles,
20M timesteps,
2-3 weeks walltime



Simulation and rendering courtesy of Mario Theers

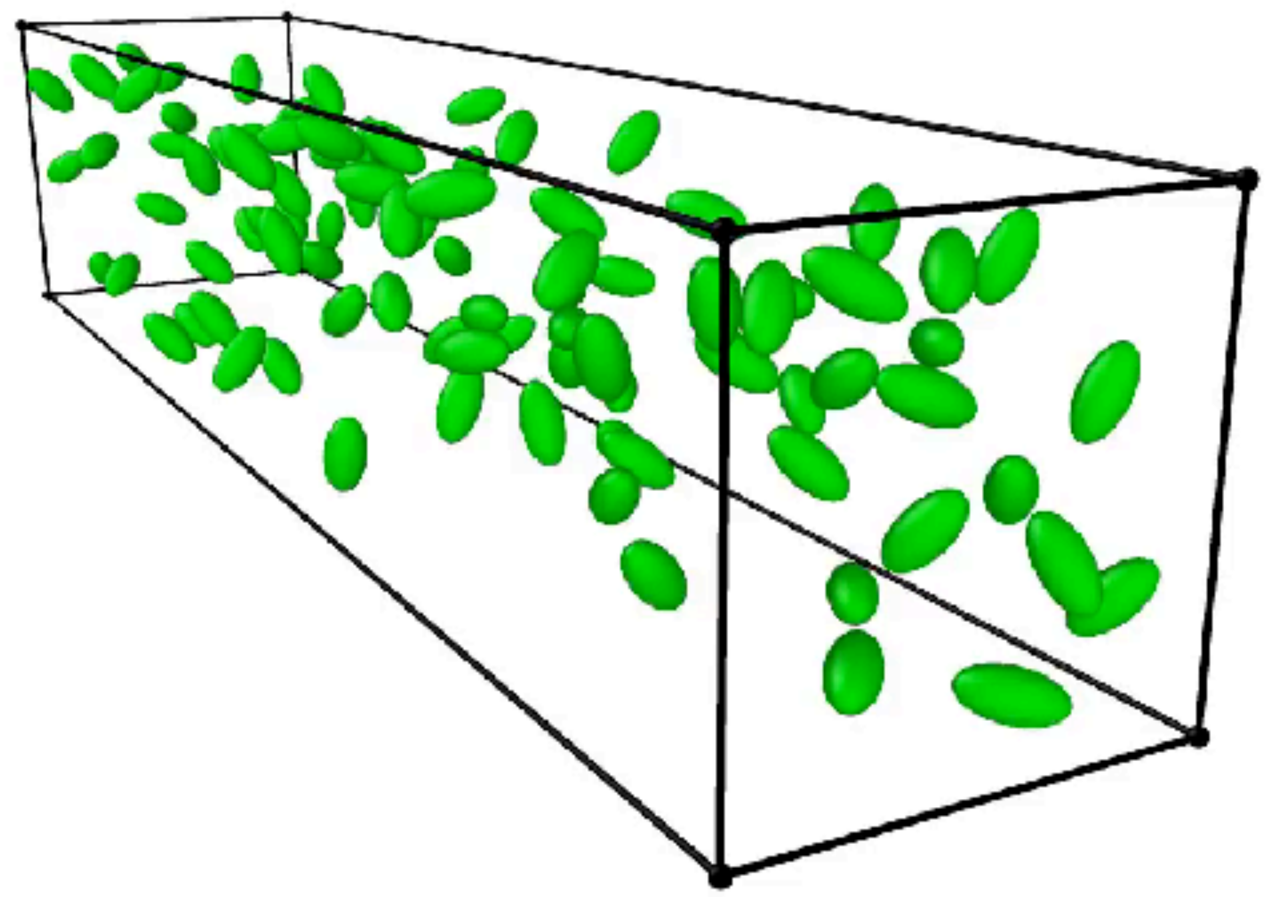
Further reading:

DOI: 10.1039/C6SM01424K (Paper) Soft Matter, 2016, 12, 7372-7385

Thank you for your time

Questions?

~100 Squirmers in flow,
L_x=300, L_y=L_z=60,
~11M fluid particles,
1M timesteps,
14h walltime

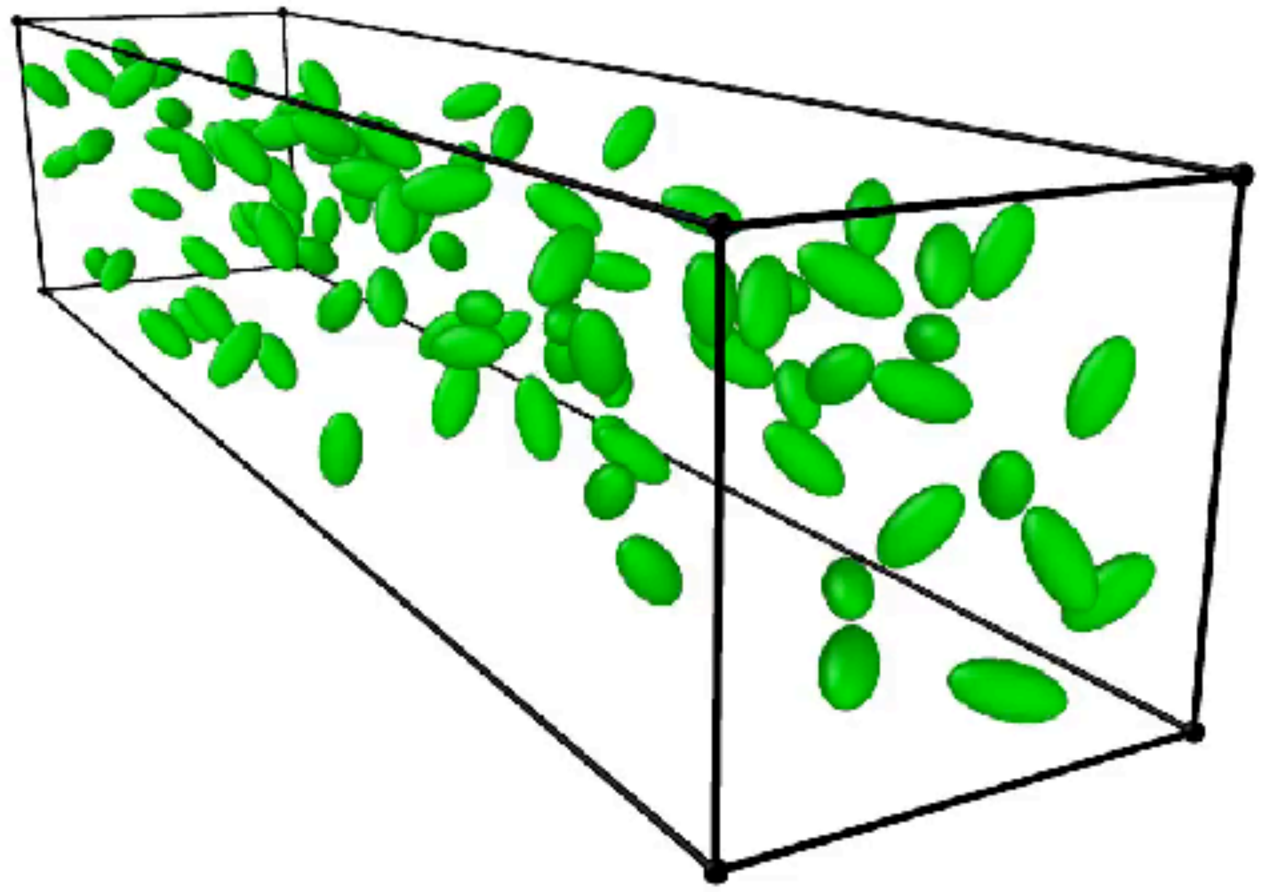


Simulation courtesy of Hemalatha Annepu,
rendering courtesy of Mario Theers

Thank you for your time

Questions?

~100 Squirmers in flow,
$L_x=300$, $L_y=L_z=60$,
~11M fluid particles,
1M timesteps,
14h walltime



Simulation courtesy of Hemalatha Annepu,
rendering courtesy of Mario Theers