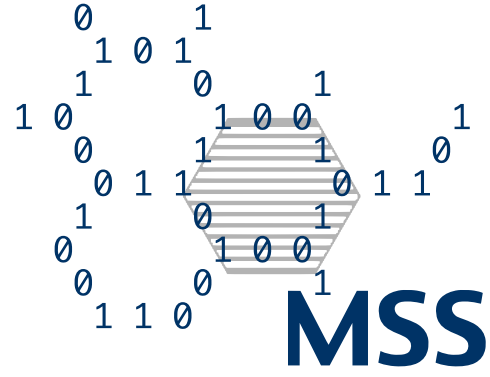
High End Simulation in Practice: Molecular Dynamics Part 2

Severin Strobl
Institute for Multiscale Simulation

2014-05-07



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Part 1: Neighbour Lists

Part 2: Smoothed Particle Hydrodynamics

Part I

Neighbour Lists

Outline

- In the previous lecture, we learned how a simple MD program might work.
- Let us recap...

Algorithm Outline

A simple molecular dynamics algorithm can be broken down as follows.

1. Take N particles, set initial positions (e.g. regular grid) and velocities (e.g. Gaussian distributed).
2. Calculate forces on the particles.

$$\mathbf{F}_i = \mathbf{F}_{body}(\mathbf{x}_i, \mathbf{v}_i) + \sum_{j \neq i} \mathbf{F}_2(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j)$$

3. Integrate to the next time step.

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{\mathbf{a}_i(t)\Delta t^2}{2}$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \frac{\mathbf{a}_i(t) + \mathbf{a}_i(t + \Delta t)}{2}\Delta t$$

4. If we've simulated long enough, stop, otherwise goto 2.

Force Models

- We had two examples for force laws in our last lecture:

$$\mathbf{F}_2^{LJ}(\mathbf{x}_i, \mathbf{x}_j) = 24\,\varepsilon \left(\frac{\sigma}{x_{ij}}\right)^6 \left[2\left(\frac{\sigma}{x_{ij}}\right)^6 - 1\right] \frac{\mathbf{x}_{ij}}{x_{ij}^2}$$

$$\mathbf{F}_2^{SD}(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j) = K\,\boldsymbol{\xi} + \gamma\,\dot{\boldsymbol{\xi}}$$

- Another force commonly used for granular matter is the viscoelastic Hertz model:

$$\mathbf{F}_2^{Hertz}(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j) = \frac{2Y\sqrt{R_{eff}}}{3\left(1-\nu^2\right)} \left(\sqrt{\xi}\boldsymbol{\xi} + \frac{3}{2}A\sqrt{\xi}\dot{\boldsymbol{\xi}}\right),$$

with Y, ν and R_{eff} denoting the Young's modulus, the Poisson's ratio and the effective radius, respectively. A is a dissipative constant, controlling how much energy is transfered into internal energy in each collision.

Outline

Algorithmic Complexity

Let's have a look at the computational complexity of the different steps in a MD simulation:

1. $\mathcal{O}(N)$: Take N particles, on a regular grid with Gaussian distributed velocities.
2. $\mathcal{O}(N^2)$: Calculate forces on the particles.

$$\mathbf{F}_i = \mathbf{F}_{body}(\mathbf{x}_i, \mathbf{v}_i) + \sum_{j \neq i} \mathbf{F}_2(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j)$$

3. $\mathcal{O}(N)$: Integrate to the next time step.

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{\mathbf{a}_i(t)\Delta t^2}{2}$$
$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \frac{\mathbf{a}_i(t) + \mathbf{a}_i(t + \Delta t)}{2}\Delta t$$

4. If we've simulated long enough, stop, otherwise goto 2.

Algorithmic Complexity

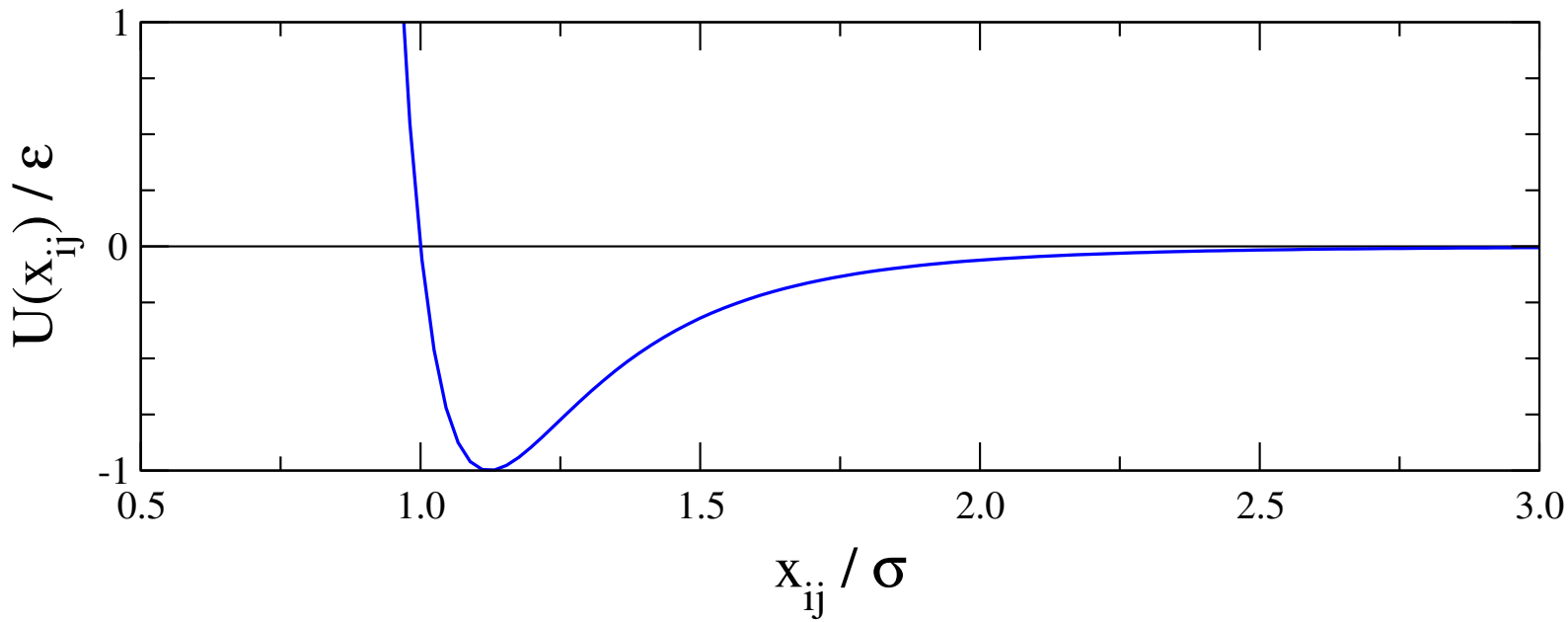
2. $\mathcal{O}(N^2)$: Calculate forces on the particles.

$$\mathbf{F}_i = \mathbf{F}_{body}(\mathbf{x}_i, \mathbf{v}_i) + \sum_{j \neq i} \mathbf{F}_2(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j)$$

- The force calculation is by far the most expensive step.
- It scales as $\mathcal{O}(N^2)$, as we sum N terms for each of the N particles.
- How can we reduce this cost?

Short Range Forces

- Lets take a look at the Lennard-Jones potential again.



- The gradient of the potential is pretty shallow for larger distances x_{ij}.
- Therefore the force tends towards zero with increasing separation distance.

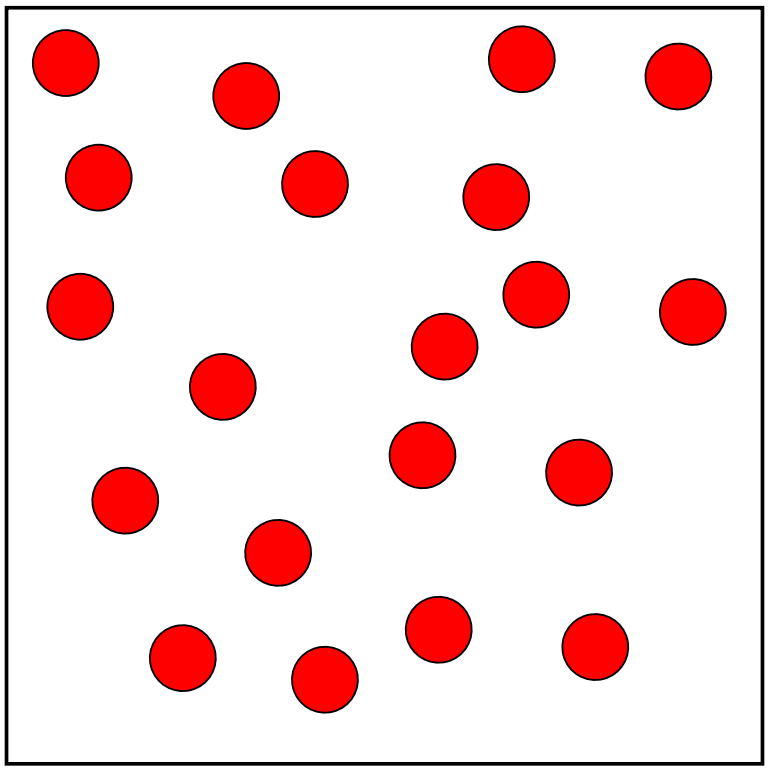
Short Range Forces

- We can then approximate the Lennard-Jones force term by truncating at a certain distance r_c:

$$\mathbf{F}^{LJ}_{2,\,\text{trunc}}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 24\,\varepsilon \left(\frac{\sigma}{x_{ij}}\right)^6 \left[2\left(\frac{\sigma}{x_{ij}}\right)^6 - 1\right] \frac{\mathbf{x}_{ij}}{x_{ij}^2} & \text{for } x_{ij} \leq r_c \\ 0 & \text{for } x_{ij} > r_c \end{cases}$$

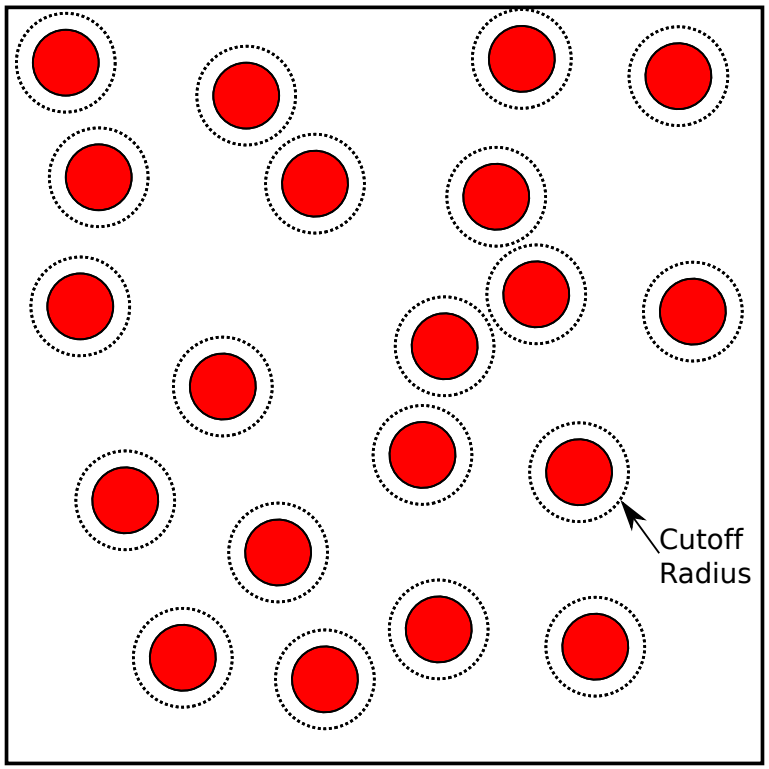
- This distance r_c where the potential is truncated is commonly referred to as the *cut-off distance* or *cut-off radius*.
- A popular choice for r_c is $2.5\,\sigma$, which is a good compromise between computational cost and the error introduced by the truncation of the potential.
- When truncating the potential, we only need to sum up the forces between neighboring particles.
- So now we need a way to find neighboring particles quickly, we need a *neighbor list*!

The System



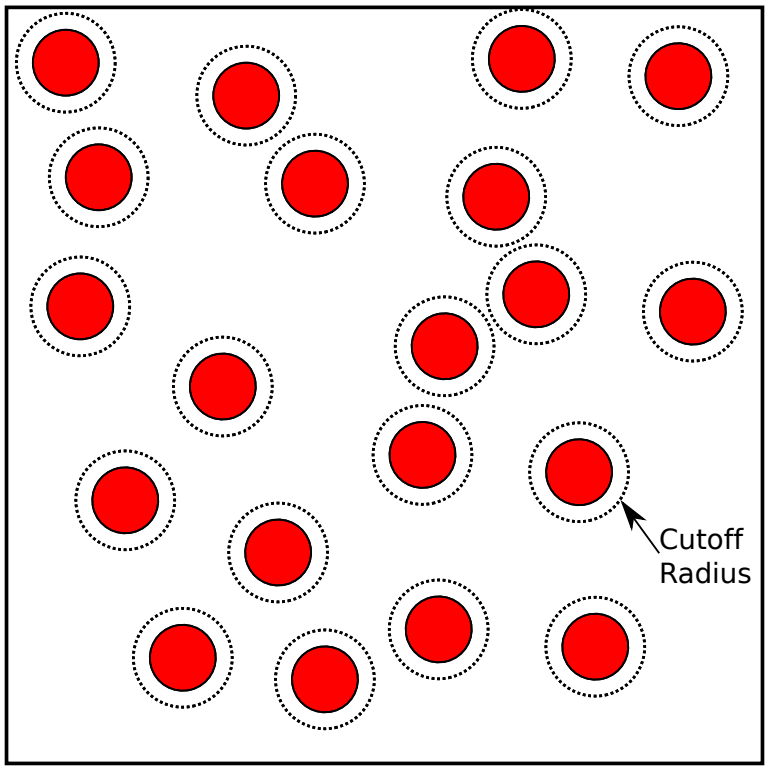
- We have a box of particles.

The System



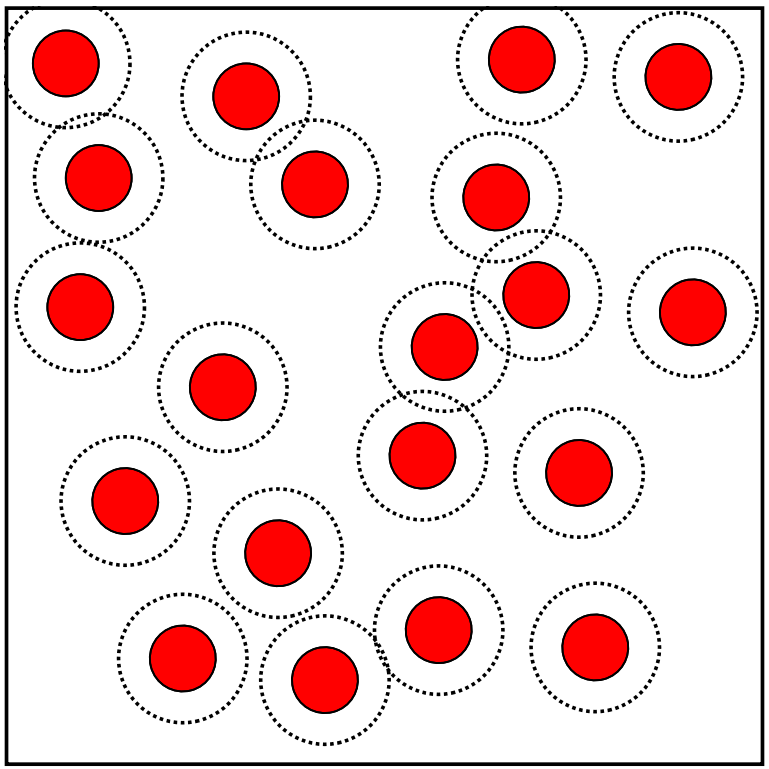
- These particles only have short range interactions with each other.
- Before we performed the force summations by checking all particles against each other.

The System



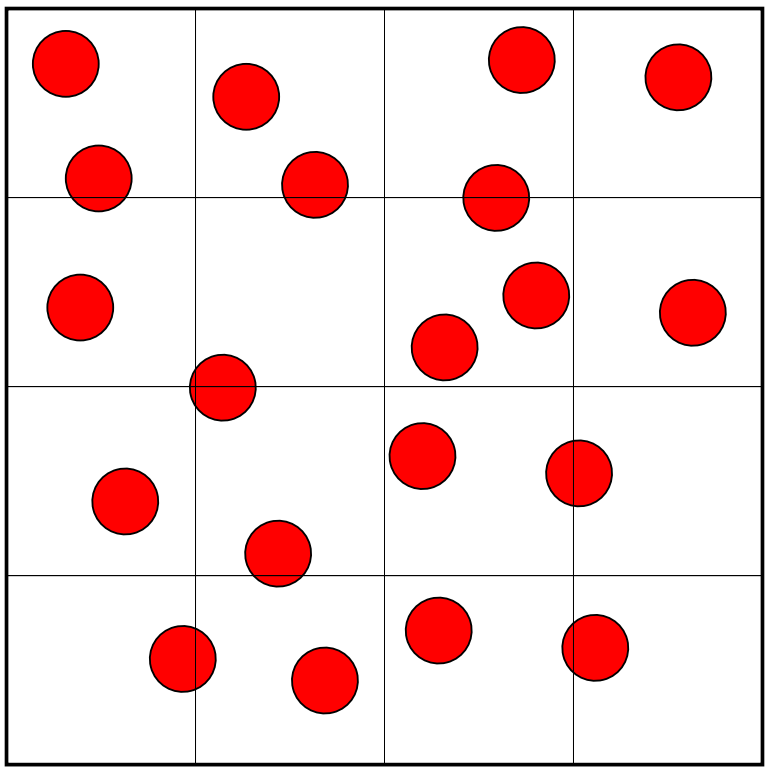
- But this is a huge waste of computation for short range forces.
- If the interactions were as short ranged as in the image, no forces would have to be computed at all!

The System



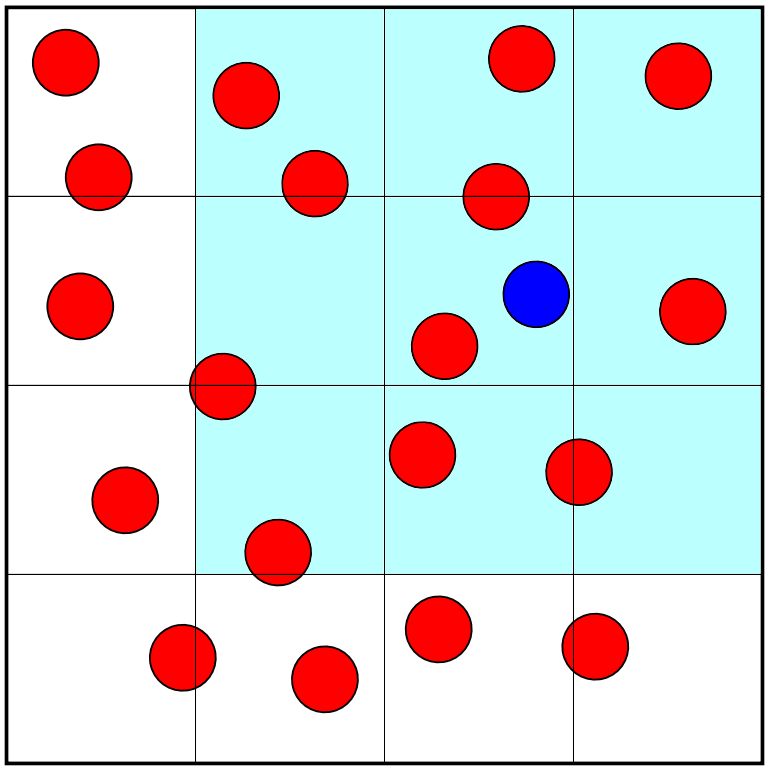
- Typically they're not quite so short ranged.
- But using a neighbor list reduces the force calculations from $O(N^2)$ to $O(N) \times N_{neighbors}$, so it almost always pays off.

The Idea



- We start by sorting the particles into a grid.
- This lets us find neighbouring (localised) particles quickly.

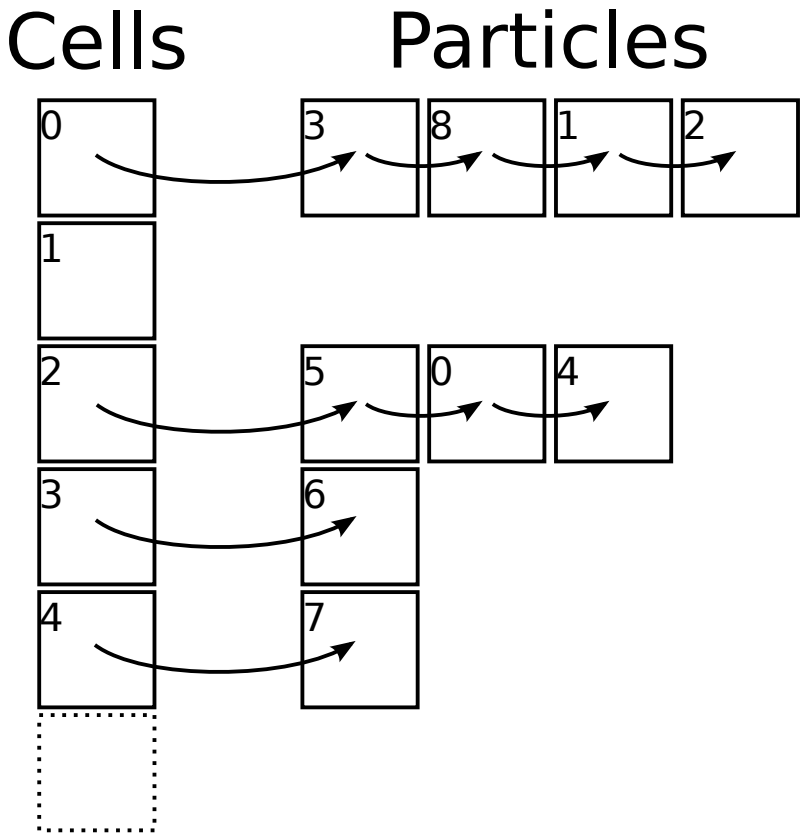
The Idea



- The width of a single cell must be $\geq r_{cutoff}$.
- Therefore, all particles within r_{cutoff} of a single particle (blue) are also within a $3 \times 3 \times 3$ group of cells.

Outline

The Idea



- The particles in each cell are stored in a linked list.
- These lists are updated as particles move from cell to cell.

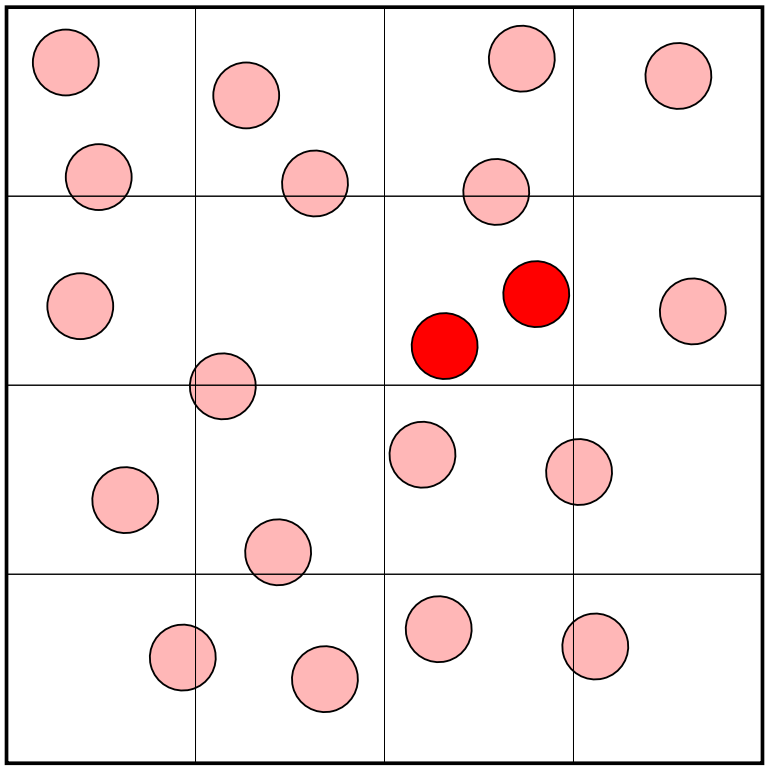
The Idea

- Finding the cell a particle belongs to is easy: division and rounding to integer.
- Adding a particle to a cell is difficult in a parallel environment like OpenCL:
  - The list of particles belonging to a cell must be available to all threads at the same time and can be large.
  - But what if multiple threads try to add a particle to the same cell at the same time? Particles might be lost, data structures corrupted,...
- Atomic operations are here to save the day!

The Solution

- How do we perform this sorting into cells using multiple threads?
- Do we make cells look for their contained particles? ($\mathcal{O}(N^2)$)
- Do we make particles look for their containing cells? ($\mathcal{O}(N)$, but requires locks or atomic operations)
- Here we'll do the latter as it is $\mathcal{O}(N)$, even with the slowdown of atomic operations it will outperform the first option even at moderate system sizes.

The Solution



By only considering the particles center point we can state:

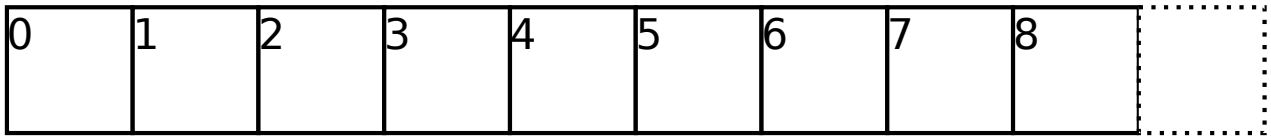
- Each particle belongs to only one cell.
- Cells can contain multiple particles.

The Data Structure

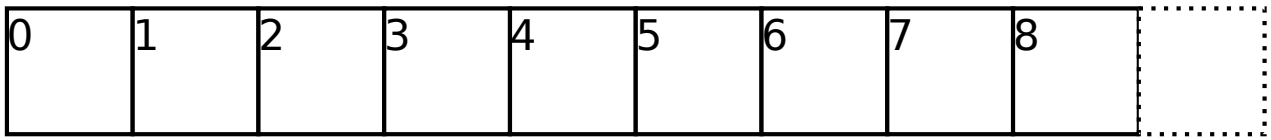
- These statements allow us to choose our data structure.
- We could use a singly linked list to store the contents of a single cell.
- Cells are assigned IDs from 0 to $N_{cells} - 1$.
- Particles are assigned IDs from 0 to $N - 1$.
- An integer is then needed for each particle and cell.

The Data Structure

Cells

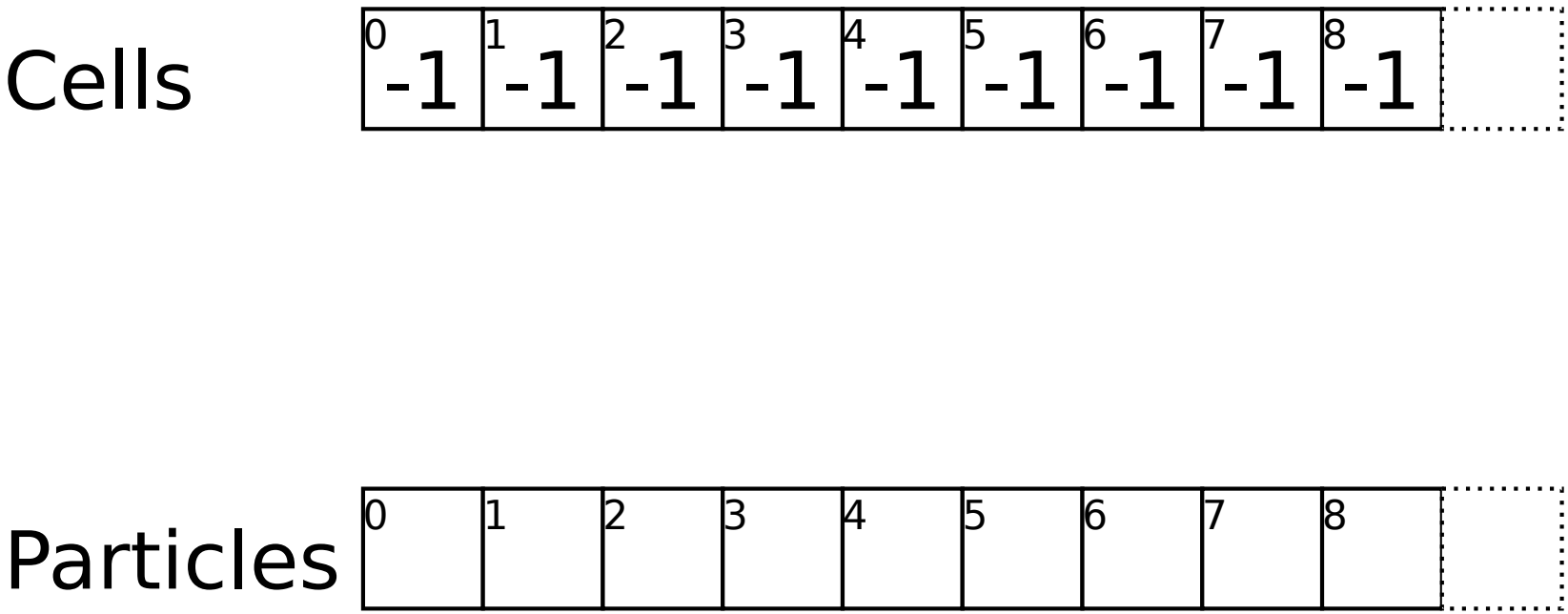


Particles



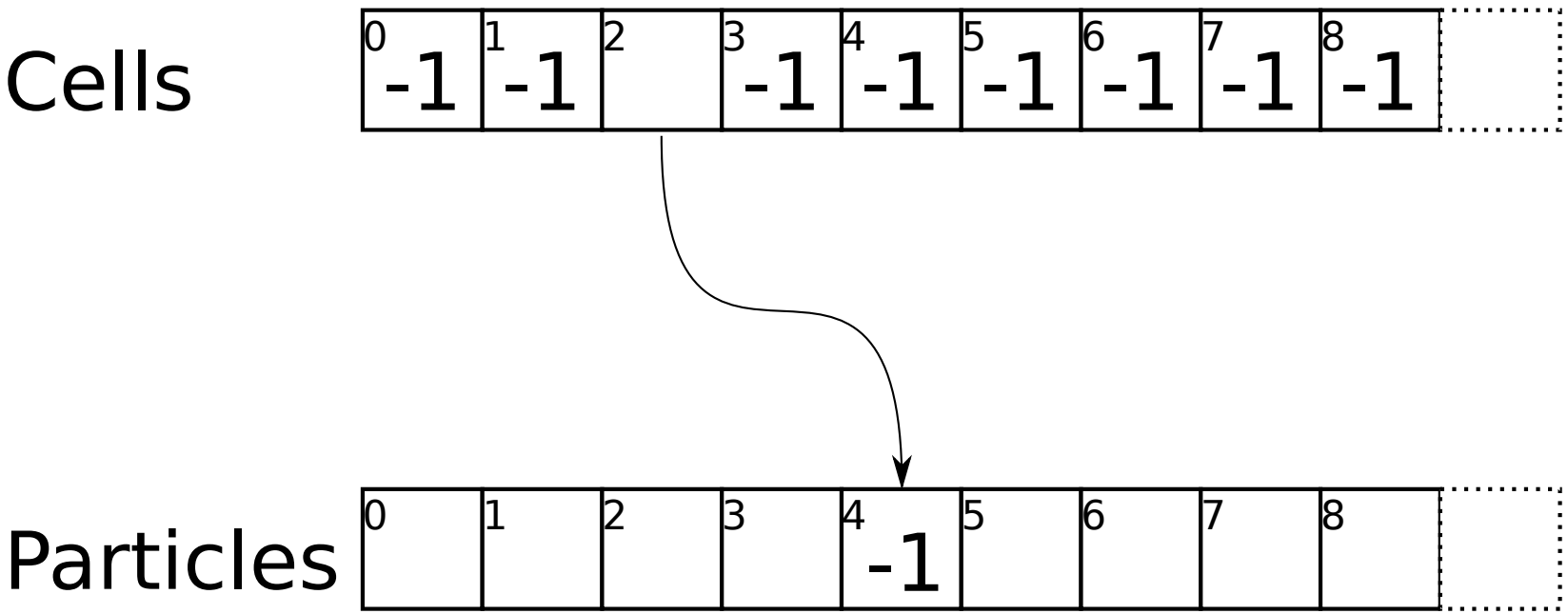
- Two arrays in memory hold the linked list integers of the cell and the particles.

The Data Structure



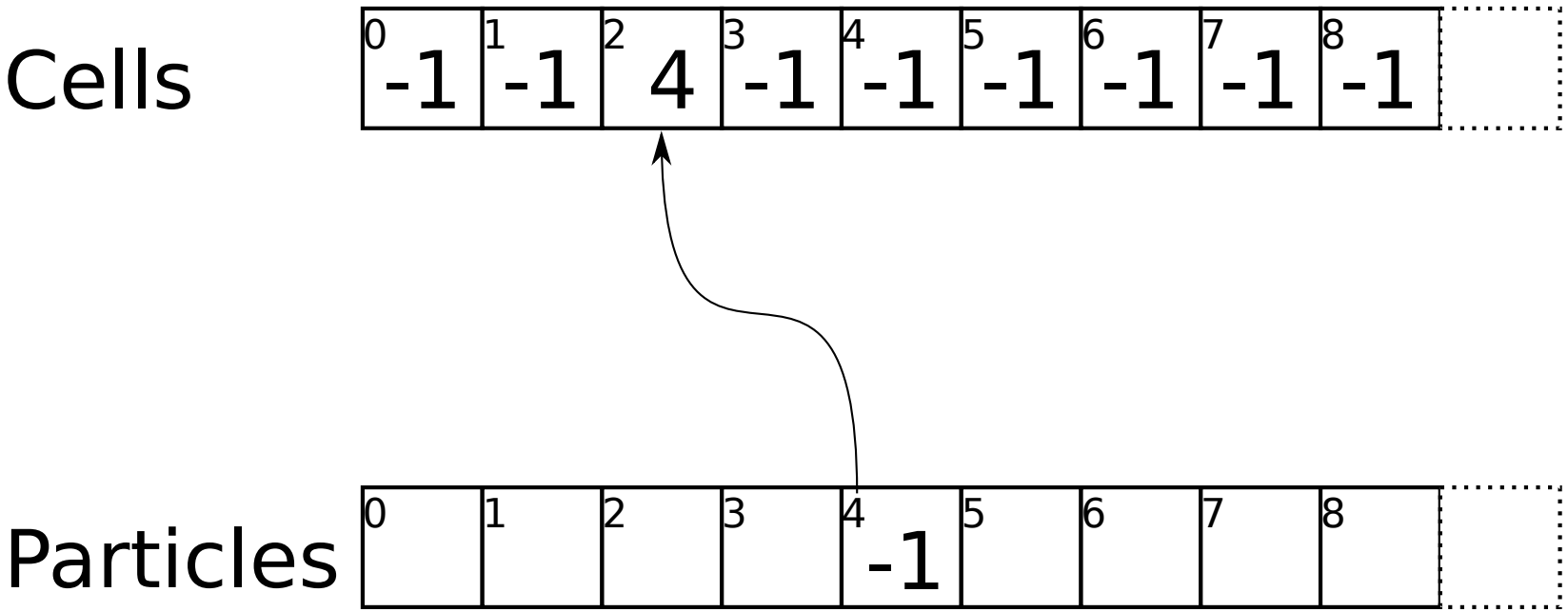
- The cells are initialized to an empty state (-1).

The Data Structure



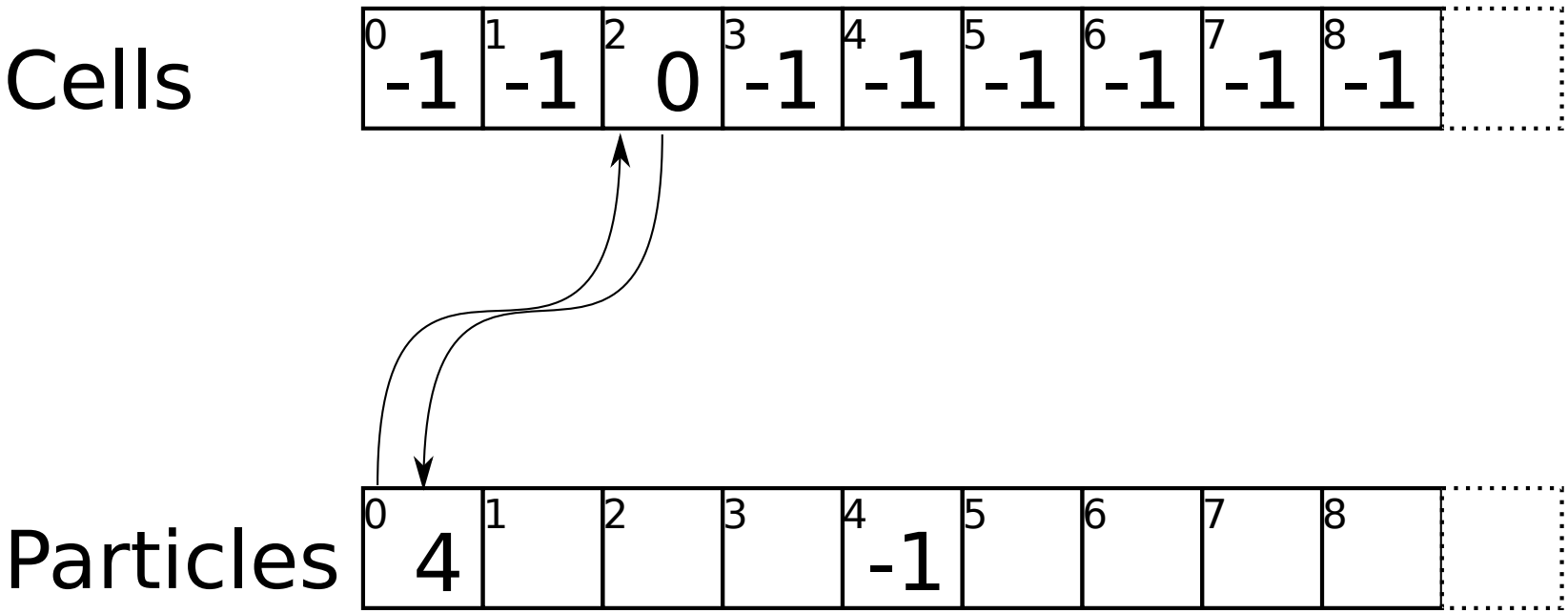
- We want to add particle 4 to cell number 2.
- We move the value of cell 2 to particle 2's integer.

The Data Structure



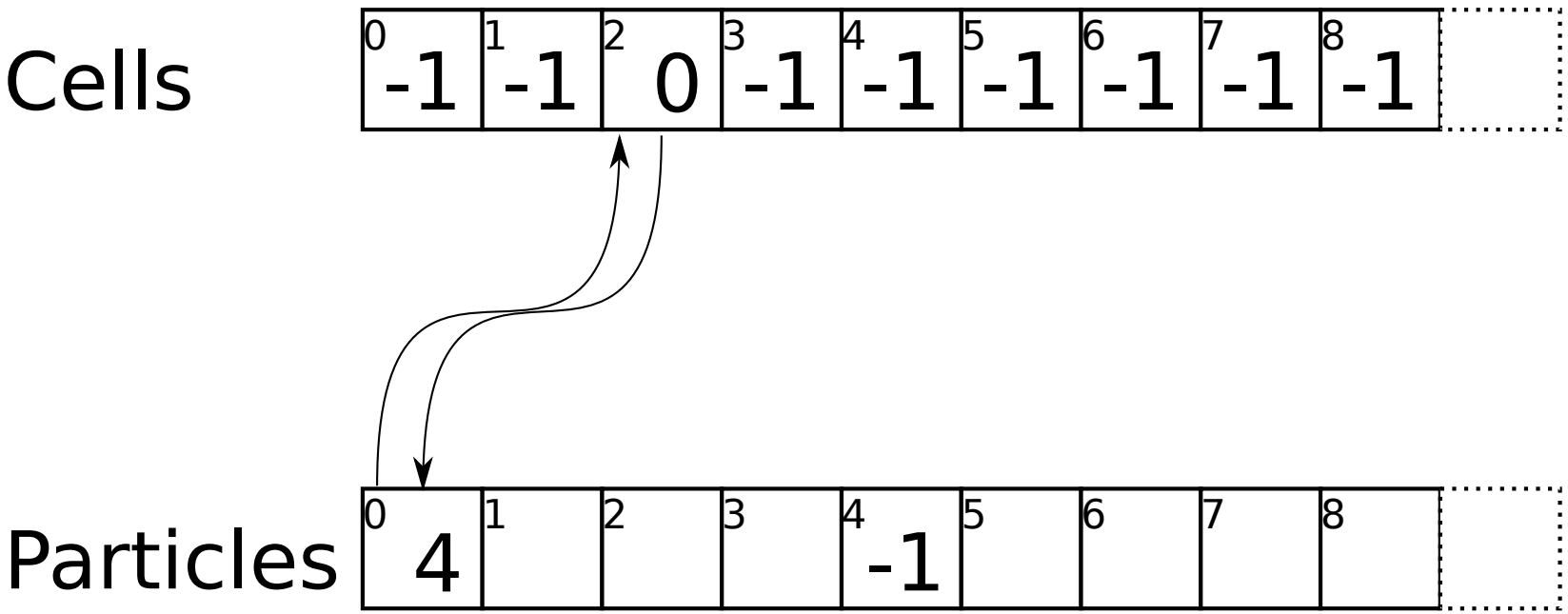
- Now we write the ID of particle 4 in the integer of cell 2.
- Particle 4 is now added to cell 2.

The Data Structure



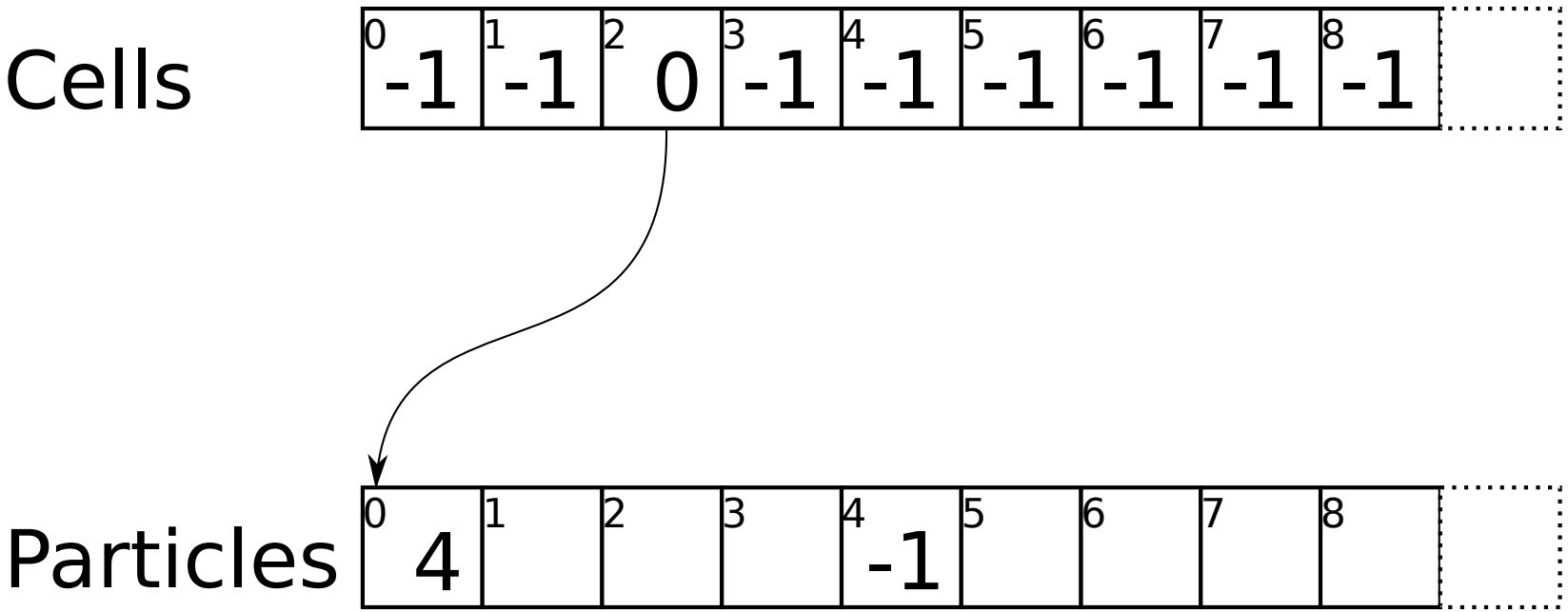
- This can continue and multiple particles can be added to a single cell.
- Essentially, each particle's number is pushed onto the start of a cell's singly linked list.

The Data Structure



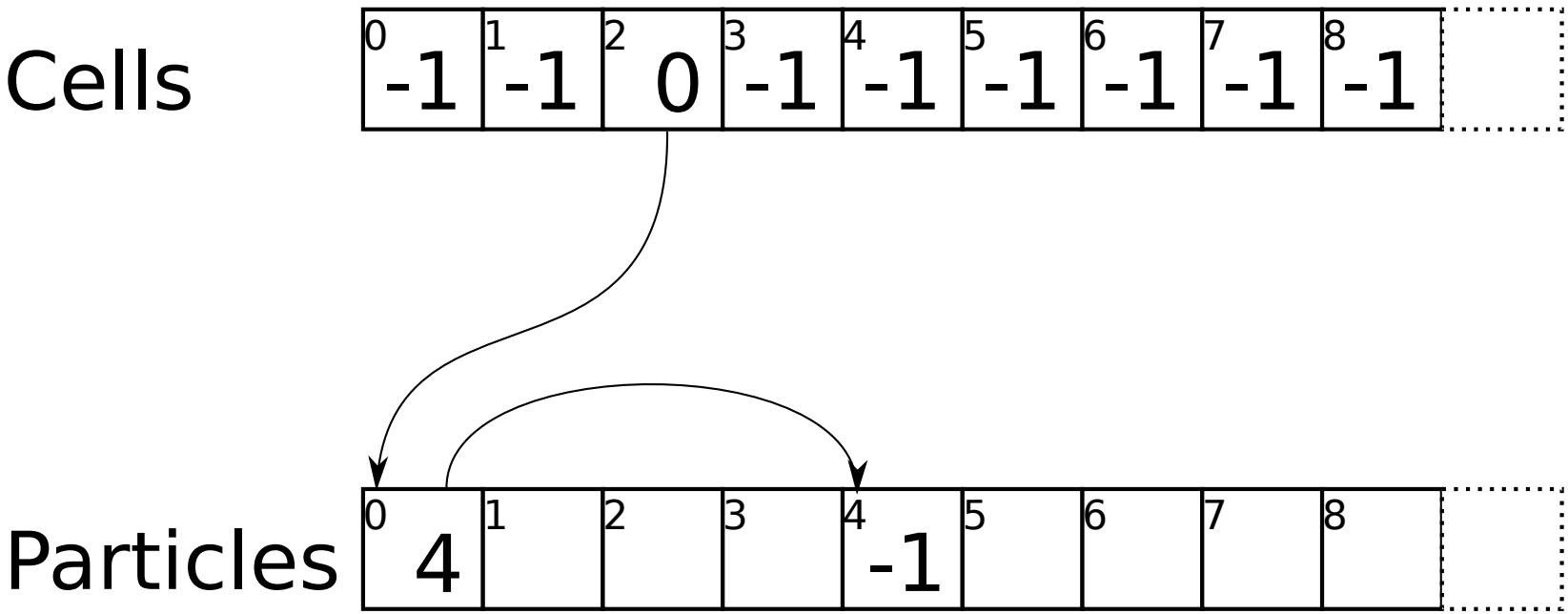
- Adding a particle is a simple exchange procedure, but what about reading a cell's contents?

The Data Structure



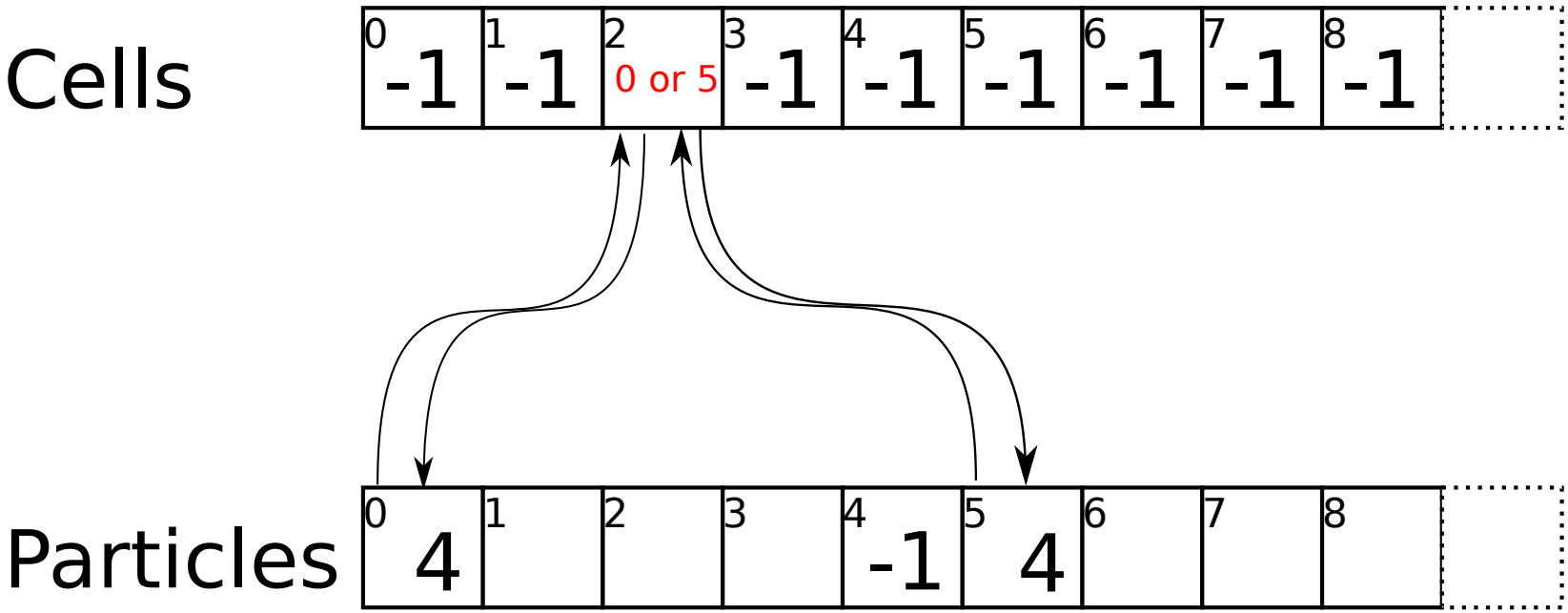
- To read what particles are in cell 2, we check its integer for the first particle (0).

The Data Structure



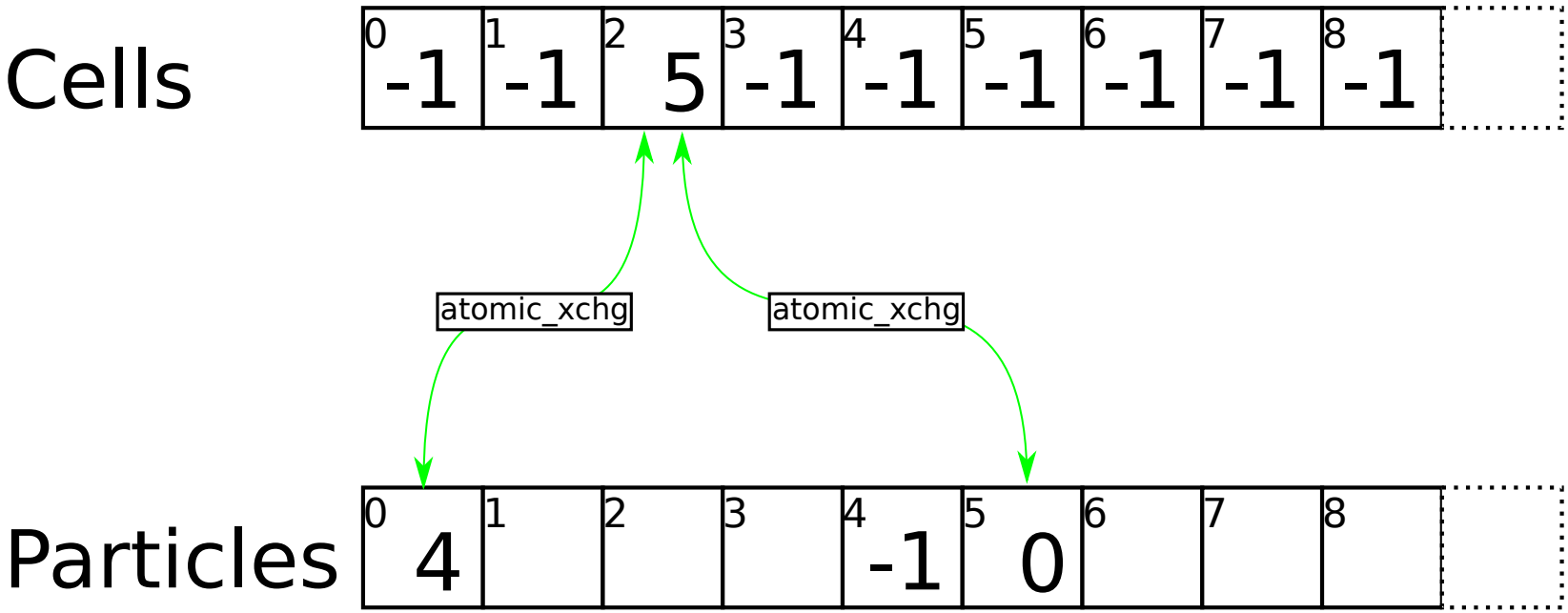
- The integer of the first particle (0), then leads us to the integer of particle 4.
- As soon as we encounter a -1, the end of the list has been reached.

The Data Structure



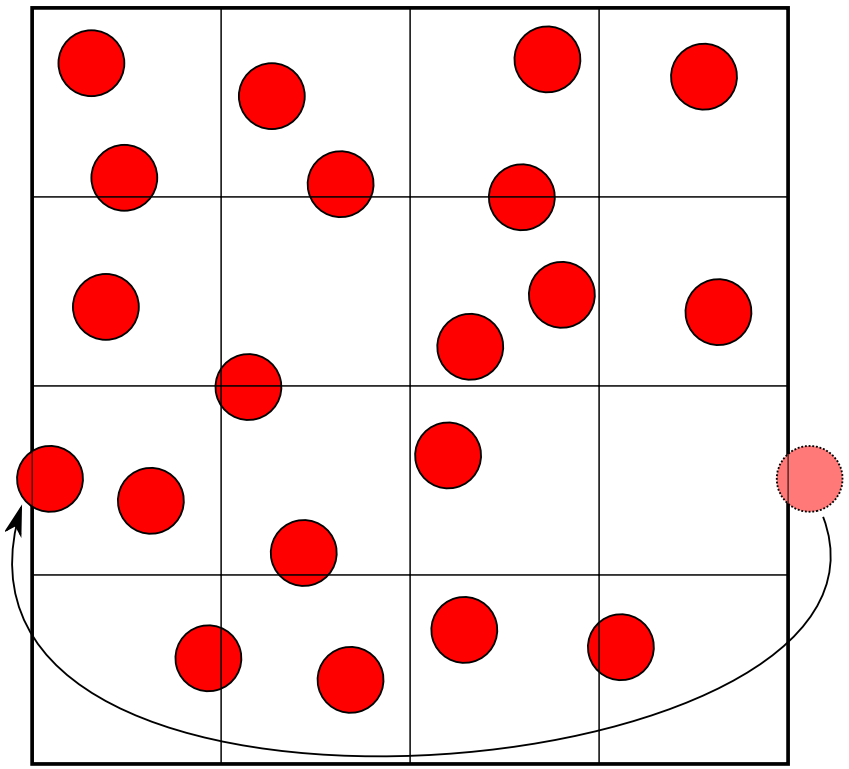
- A problem arises if multiple threads try to add a particle to the same cell.
- Their exchange operations might overlap, leading to a invalid state.

The Data Structure



- We solve this by making the exchange operation "atomic" (`atomic_xchg` in OpenCL).
- This ensures that our linked list is always valid.

Final Notes



- For our kernel to work, all our particles must be within the gridded area, so we have to ensure they stay within the system boundaries.
- We use periodic boundary conditions for this and every time step we must ensure the particles are within the primary image.

Final Notes

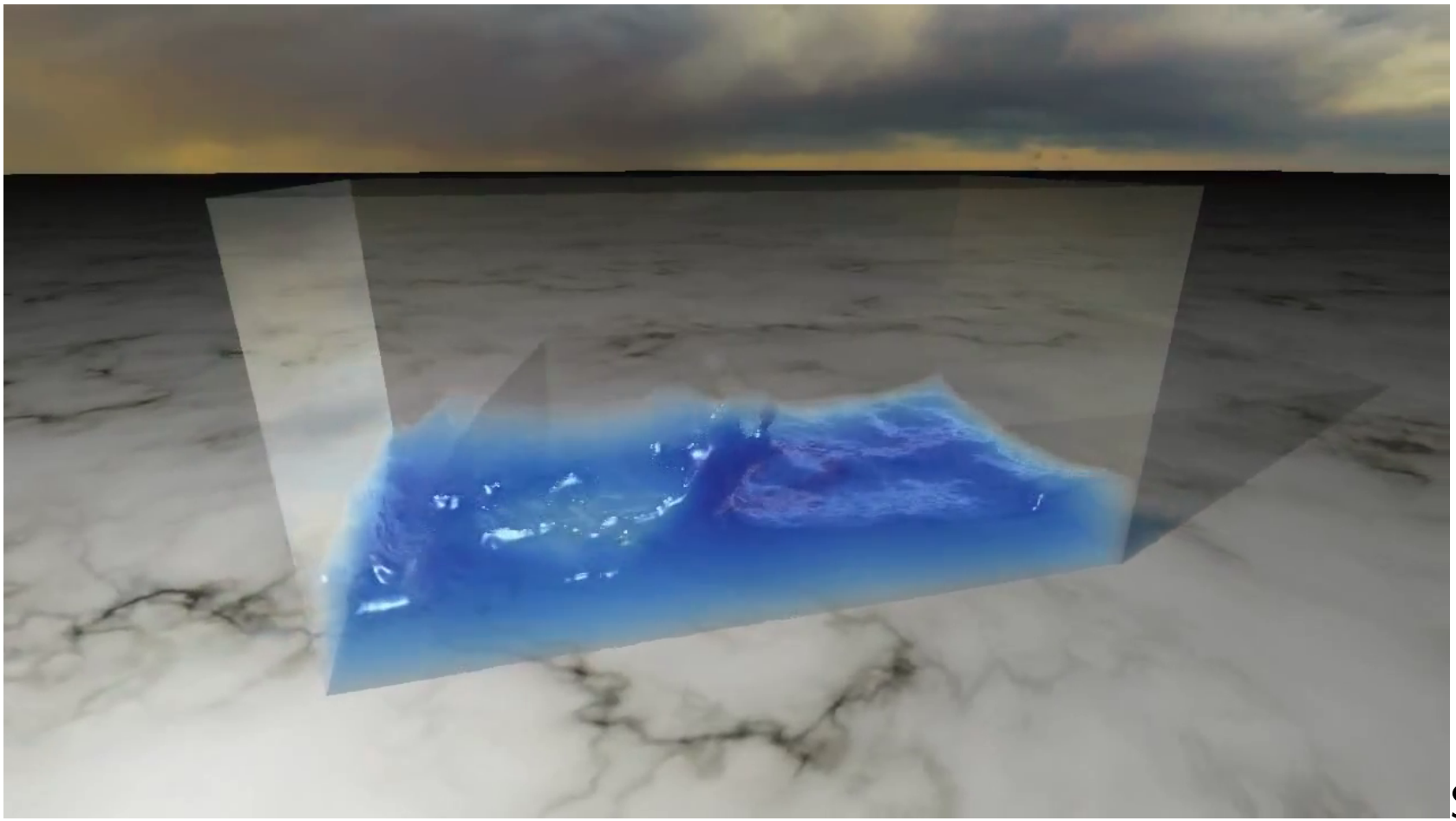
- This neighbor list approach allows our simulation to scale to much larger system sizes.
- There are still some speed-ups to be gained:
  - You can make the neighborhood area larger than needed and then the neighbor lists can remain valid for multiple time steps.
  - The cell neighbor list can be used to create a spherical neighbor list to reduce the number of neighbors even further.

Part II

Smoothed Particle Hydrodynamics

Outline

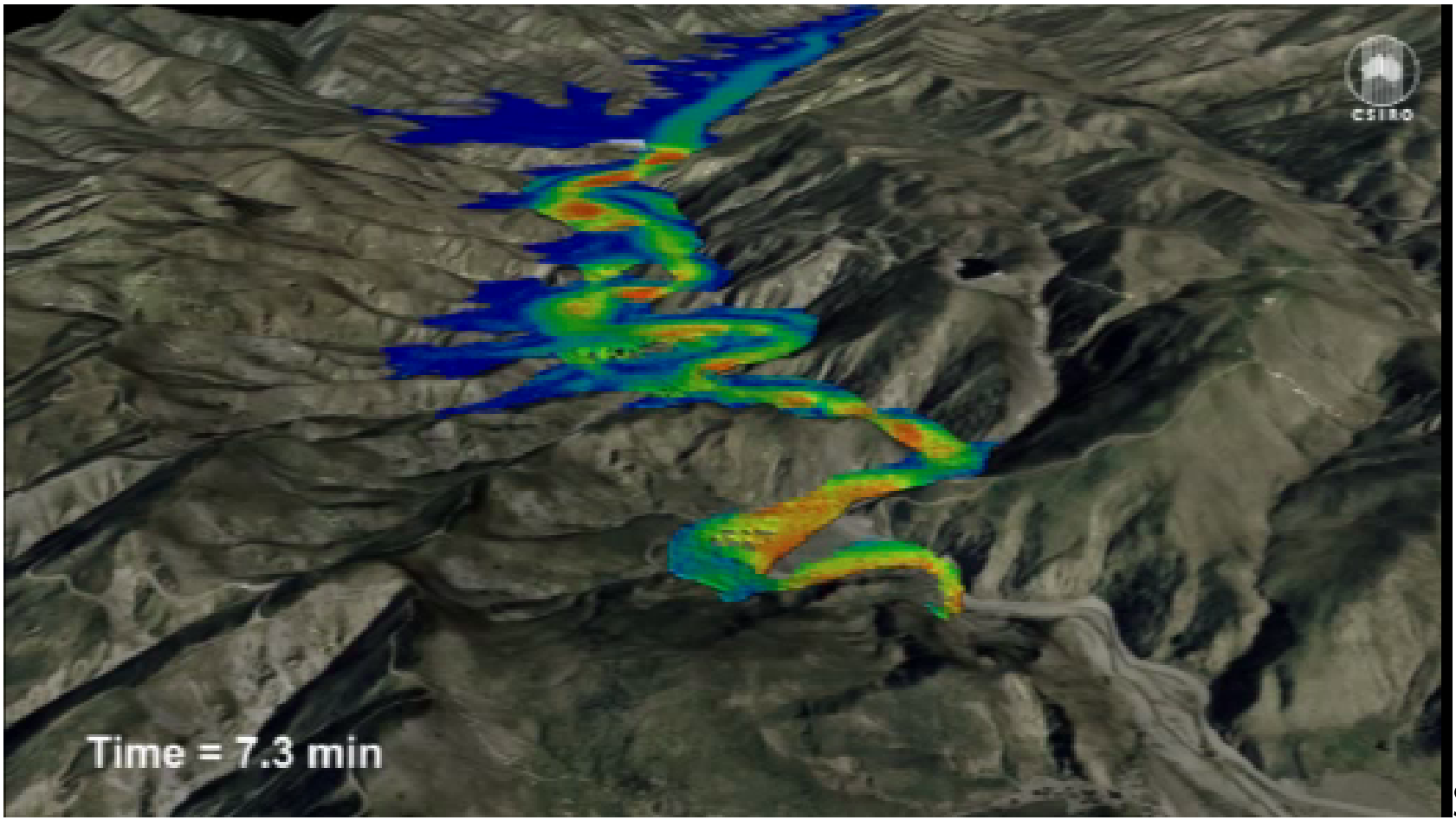
Applications for Smoothed Particle Hydrodynamics (SPH)



Source:

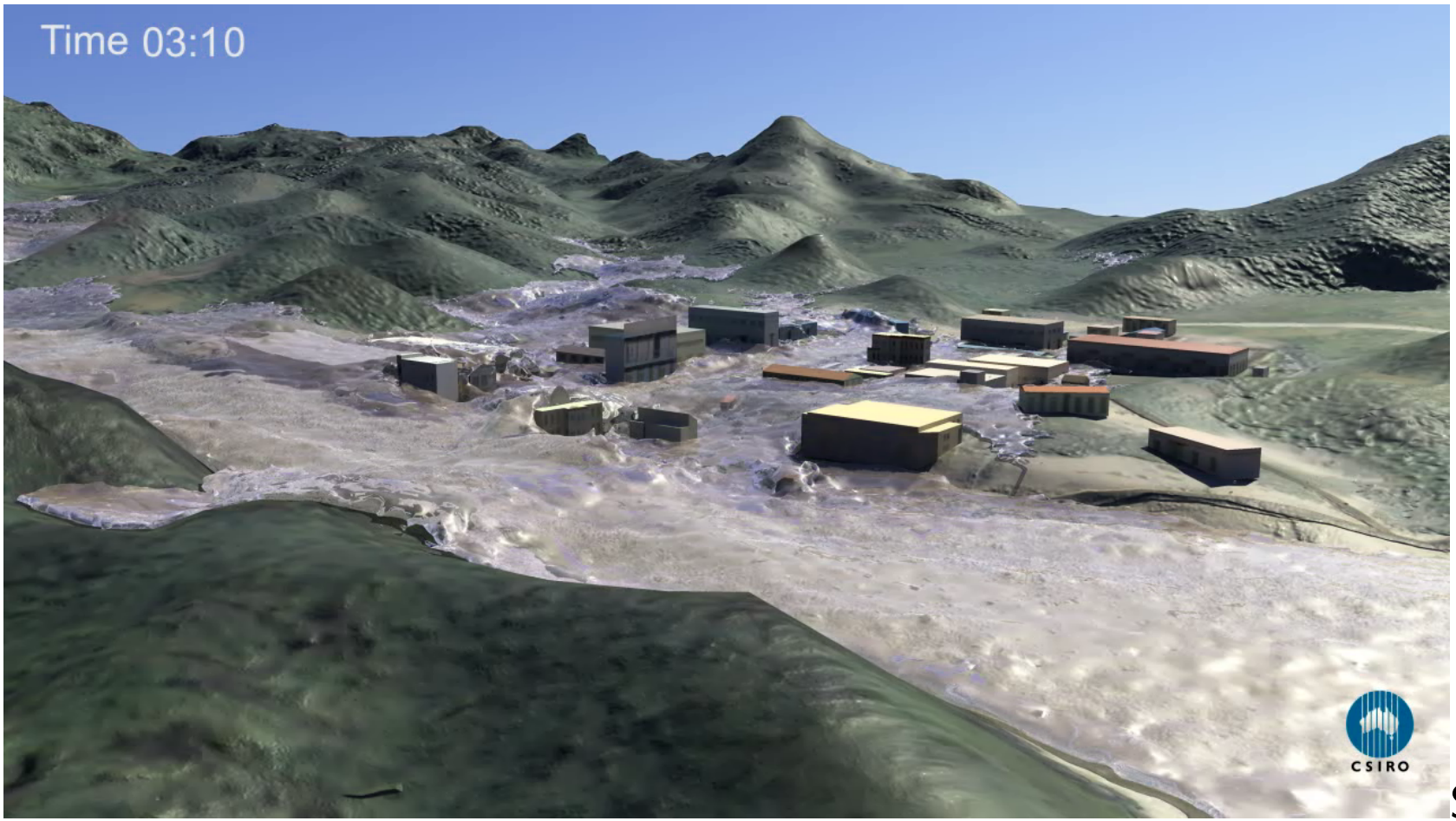
NVIDIA PhysX Fluids Demo running on a GF100

Applications for Smoothed Particle Hydrodynamics (SPH)



Source: CSIRO Mathematics, Informatics & Statistics, Clayton, AU

Applications for Smoothed Particle Hydrodynamics (SPH)



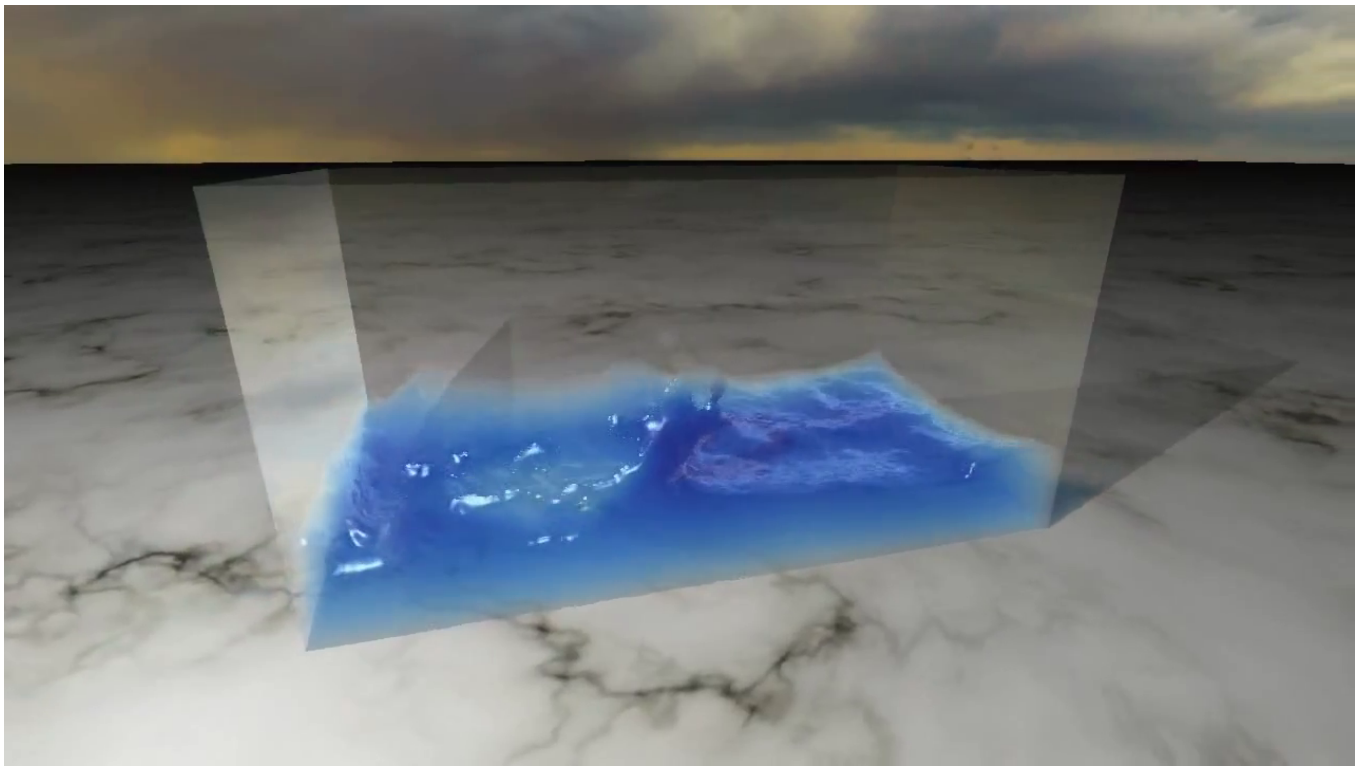
Source: CSIRO Mathematics, Informatics & Statistics, Clayton, AU

Smoothed Particle Hydrodynamics (SPH)

- We're going to describe SPH in a rough and general fashion, as the underlying theory is not in the scope of this lecture.
- You can read several more thorough derivations on the internet or in one of the many books on the subject.
- This description is not complete and misses much/all of the necessary approximations required to derive SPH, so **Buyer Beware**.
- The presentation of the SPH method is based on the work of T. Harada et al.: Smoothed Particle Hydrodynamics on GPUs, 2007.

Outline

Smoothed Particle Hydrodynamics



- Smoothed Particle Hydrodynamics is a numerical method to solve a set of **hydrodynamic** equations.
- **Hydrodynamics** is the theory describing the motion of fluids.

Smoothed Particle Hydrodynamics

- Hydrodynamics generally solves balance equations for the density,momentum and energy.
  - For example, the conservation of mass gives the continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \, \mathbf{v}) = 0$$

  - This equation simply states, that for an **element** (small volume) of fluid to change its **density** ρ (lose or gain some fluid and by that change its mass), that mass must flow **out** or flow **in** from/to the element.
  - But we have two unknowns, and only one equation. In addition we need an expression for the **flow velocity v**.

Smoothed Particle Hydrodynamics

- Hydrodynamics generally concerns solving balance equations for the density, momentum and energy.
  - A momentum balance for the fluid element helps us here:

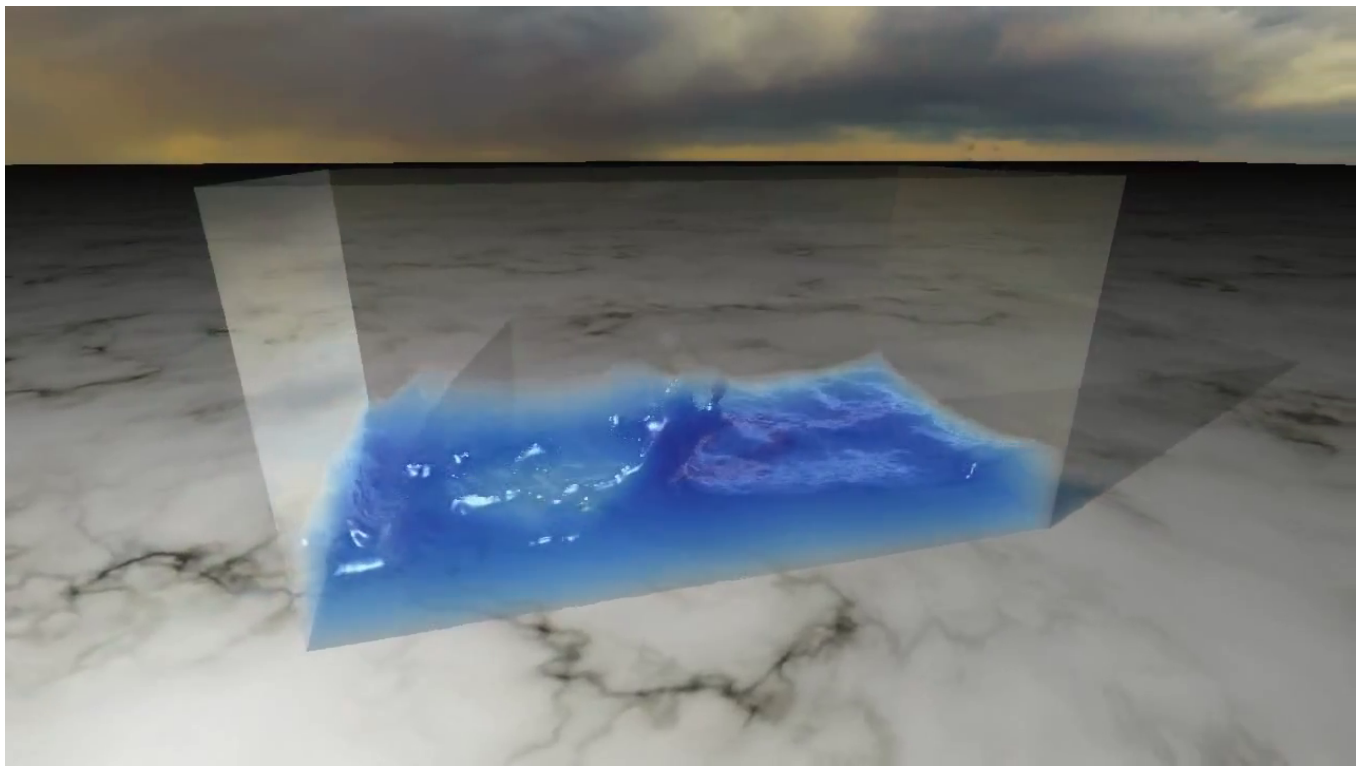
$$\rho \frac{D\mathbf{v}}{Dt} = \rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \nabla \cdot \mathbf{T} + \mathbf{F}_{other},$$

    with p denoting the pressure, $\mathbf{T}$ being the deviatoric stress tensor and $\mathbf{F}_{other}$ representing body forces acting on the fluid element (like gravity).
  - This bares more than a passing resemblance to:

$$m\,\mathbf{a} = \mathbf{F} = \mathbf{F}_{pressure} + \mathbf{F}_{viscous} + \mathbf{F}_{other}$$

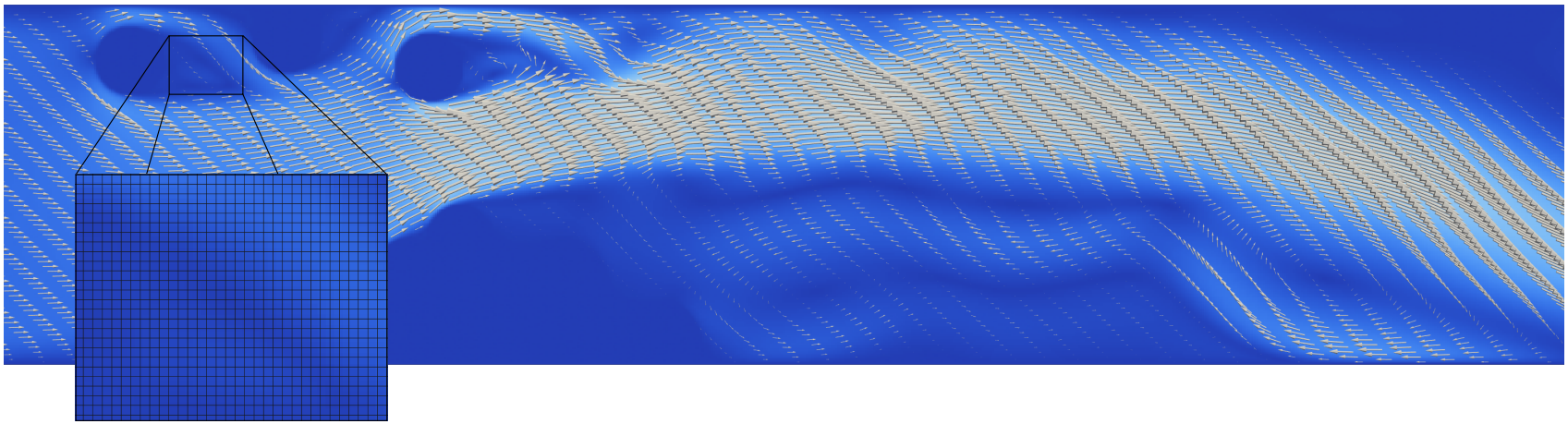
    As they should, fluids obey Newtons equation of motion just like everything else (e.g. atoms in molecular dynamics).
  - For an incompressible Newtonian fluid the stress term $\nabla \cdot \mathbf{T}$ simplifies to $\mu \nabla^2 \mathbf{v}$, so we get $\mathbf{F}_{viscous} = \mu \nabla^2 \mathbf{v}$, with μ being the viscosity.

Smoothed Particle Hydrodynamics



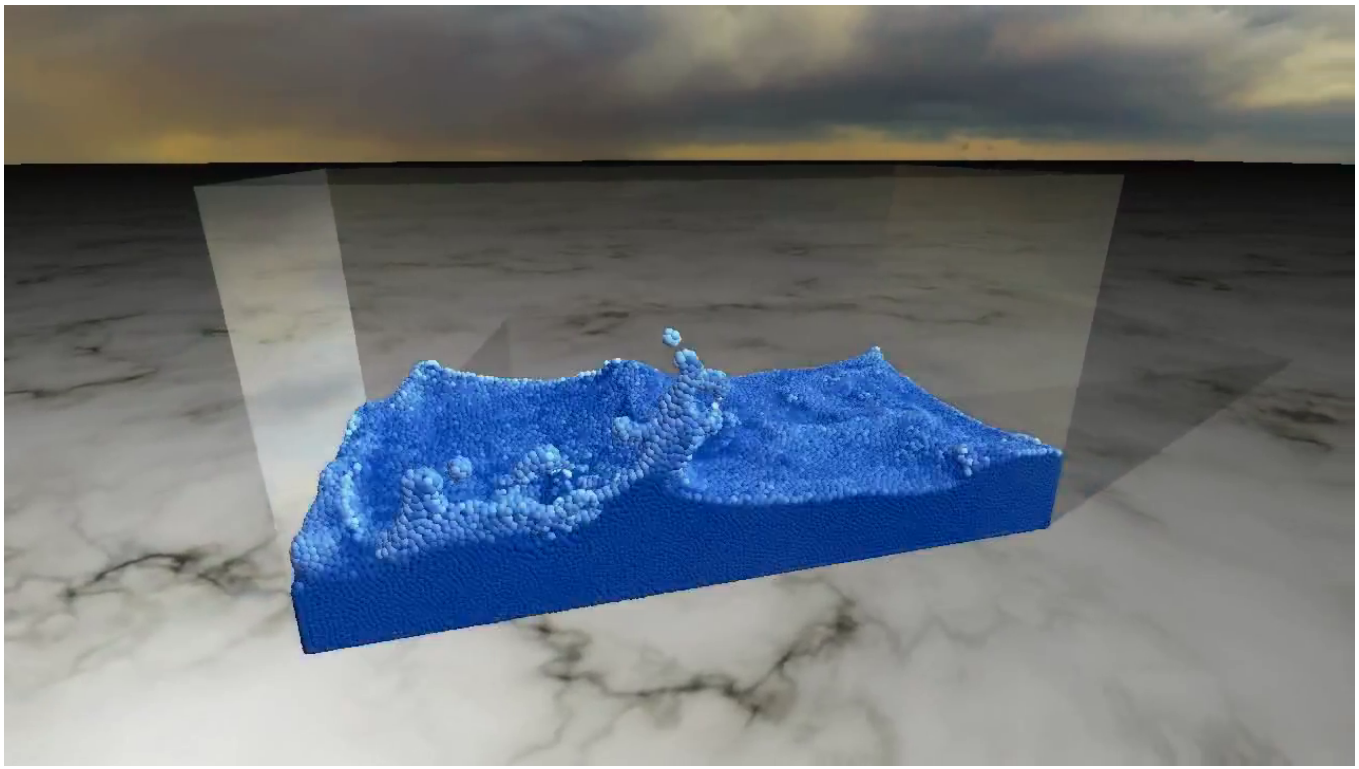
- Mathematically speaking these equations are only valid for a point of fluid (a differential element).
- But we have a finite volume of fluid to simulate and numerically we can only handle a finite number of “fluid points” or elements.

Smoothed Particle Hydrodynamics



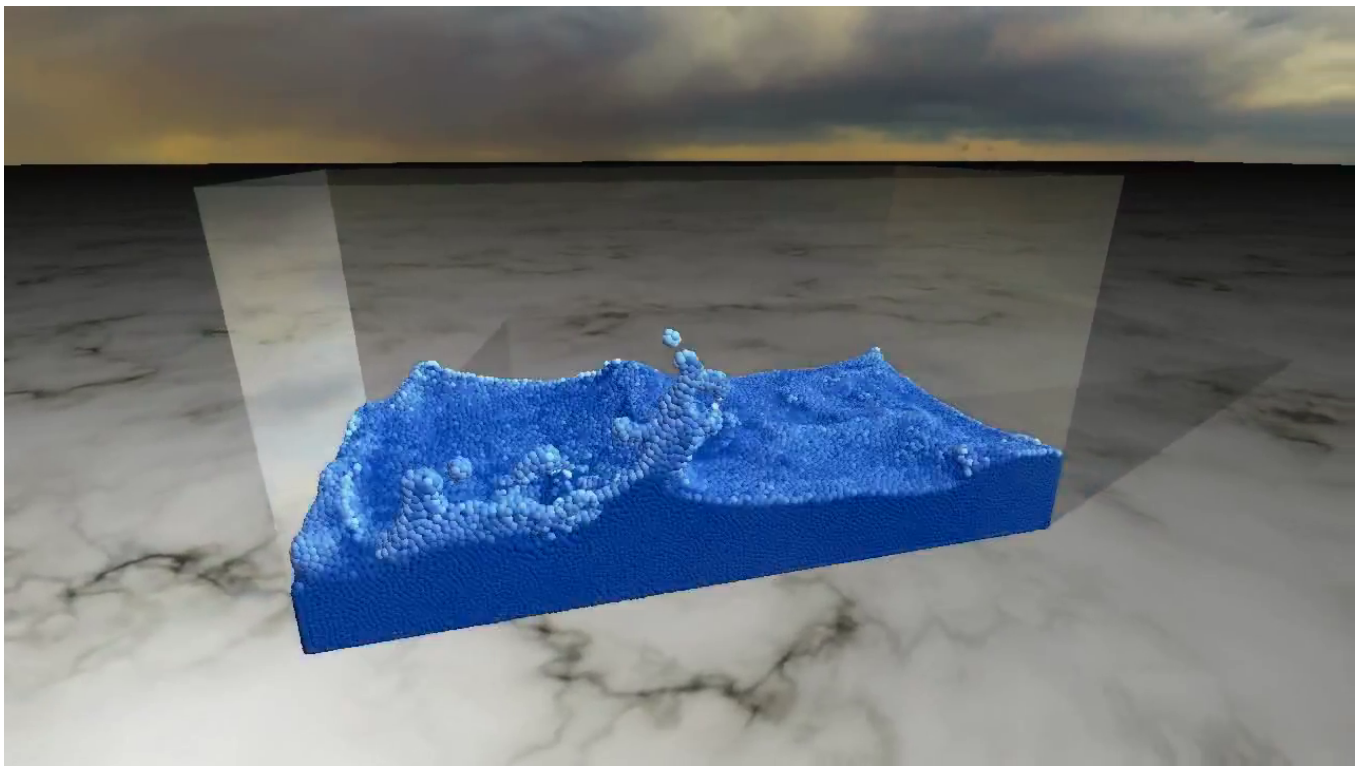
- A common approach is to use a grid of nodes to approximate the fluid.
- The shape and topology of this grid is a huge area of research as it must be:
  - Fine enough (many nodes) to capture all the essential behavior.
  - Coarse enough (few nodes) to have a manageable number of nodes and remain efficient.
- Only on the nodes of the grid are the hydrodynamic variables calculated or stored (density, temperature, pressure, velocity).

Smoothed Particle Hydrodynamics



- SPH still utilizes a grid, but the nodes of the grid **move with the fluid**.
- Each node of the grid represents a finite mass of fluid.

Smoothed Particle Hydrodynamics



- This is the "Particle" part in the name of Smoothed Particle Hydrodynamics.
- We have particles of fluid, each with a mass and interacting according to the hydrodynamic equations.

Smoothed Particle Hydrodynamics

- A problem with this SPH grid is that the neighbors of each node/particle of the grid change continuously.
- With a fixed grid, we could determine spatial derivatives (needed for the fluxes) through looking at the well defined neighboring nodes.
- But what are the neighboring nodes of a constantly shifting grid?
- And nodes must be correctly weighted too for the spatial derivative, depending on orientation and distance!

Smoothed Particle Hydrodynamics

- To combat this difficulty, we will use a smoothing function $W(\mathbf{x} - \mathbf{x}_i, h)$ to map our "particle" description to continuous hydrodynamic fields.
- This weights the contribution of nearby particles, $\mathbf{x}_i$, to the hydrodynamic properties at some point $\mathbf{x}$.

Smoothed Particle Hydrodynamics

- This smoothing function $W(\mathbf{x} - \mathbf{x}_i, h)$ also has a smoothing length, h, which describes how near particles have to be to contribute.
- This is the "Smoothed" part in the name of Smoothed Particle Hydrodynamics.

Smoothed Particle Hydrodynamics

- The smoothing function is both a strength and a weakness of SPH.
- We don't have a grid to maintain, the fluid naturally clusters grid points near dense regions. We can even have free surface boundaries!
- But this smoothing function is also a curse for low density fluids, our approximation breaks down as SPH particles don't overlap. Our low density regions will have either 0 or small spikes in the density!

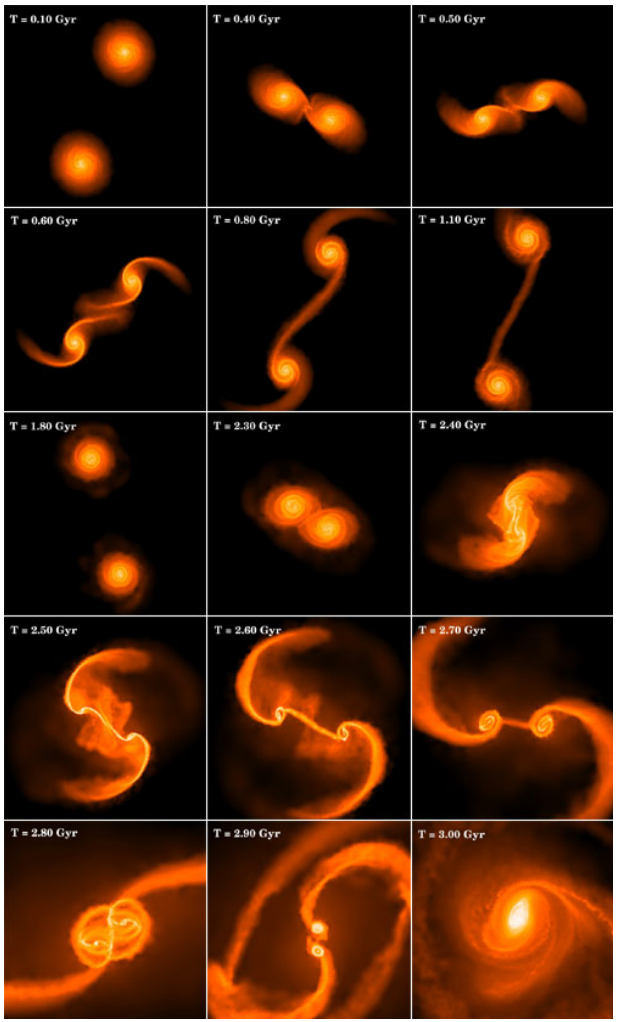


Image by Stelios Kazantzidis

http://kicp.uchicago.edu/research/highlights/highlight_2006-03-21.html

- This can be counteracted by varying the smoothing length depending on the local density, but we won't cover this here.
- Varying the smoothing length is crucial in galactic simulations, a popular application of SPH.

Smoothed Particle Hydrodynamics

- With a smoothing function we can define "smoothed" hydrodynamic properties

$$\langle A(\mathbf{x}) \rangle = \int A(\boldsymbol{\tau}) W(\mathbf{x} - \boldsymbol{\tau}, h) \, \mathrm{d}\boldsymbol{\tau}$$

  where A is a hydrodynamic property, like the density.
- As our smoothing length vanishes ($\lim_{h \to 0}$) and our number of SPH particles increases, our "smoothed" properties approach the true hydrodynamic properties.

Smoothed Particle Hydrodynamics

- As the fluid is represented by particles, the previous integral over all space can be reduced to a sum over all particles:

$$\langle A(\mathbf{x}) \rangle = \sum_i \frac{m_i}{\rho_i} A(\mathbf{x}_i) W(\mathbf{x} - \mathbf{x}_i, h),$$

  where $A(\mathbf{x}_i)$ is the property of particle i (like the mass).
- This equation lets us calculate any property, at any point, using only the data at each particle's position.
- For example, as the mass of a single SPH particle is defined, the smoothed density of the fluid at any given point is given by:

$$\langle \rho(\mathbf{x}) \rangle = \sum_i \frac{m_i}{\rho_i} \rho_i W(\mathbf{x} - \mathbf{x}_i, h) = \sum_i m_i W(\mathbf{x} - \mathbf{x}_i, h)$$

Smoothed Particle Hydrodynamics

- Even more crucially, spatial derivatives in a property (like the pressure term $\mathbf{F}_{pressure} = -\nabla p$), can be rewritten in terms of gradients of the smoothing function!

$$\langle \nabla A(\mathbf{x}) \rangle = \sum_i \frac{m_i}{\rho_i} A(\mathbf{x}_i) \nabla W(\mathbf{x} - \mathbf{x}_i, h)$$

- Thus we can solve our hydrodynamic equations using SPH, we just need an appropriate smoothing function.
- If we solve the momentum balance equation at the point of an SPH particle, then we know how the particle moves (the particle follows the flow)!

Outline

Smoothed Particle Hydrodynamics

- We want a spherical smoothing function that has the properties

$$\int W(\mathbf{x}, h)\, d\mathbf{x} = 1 \quad \textit{and}$$

$$\lim_{h \to 0} W(\mathbf{x} - \mathbf{x}_i, h) = \delta(\mathbf{x} - \mathbf{x}_i)$$

- We could use a Gaussian, but this would mean that all particles would contribute to all points.
- Instead, we want a localized function that is zero if $|\mathbf{x} - \mathbf{x}_i| \geq r_{cutoff} = h$ (for the neighbor lists).
- Often people use three piece splines.
- Here, we'll use the approach from the paper by T. Harada et al. and use the following function:

$$W(\mathbf{x} - \mathbf{x}_i, h) = \begin{cases} \frac{315}{64\,\pi\,h^9} \left(h^2 - |\mathbf{x} - \mathbf{x}_i|^2\right)^3 & \text{for } |\mathbf{x} - \mathbf{x}_i| \leq h \\ 0 & \text{for } |\mathbf{x} - \mathbf{x}_i| > h \end{cases}$$

Smoothed Particle Hydrodynamics

- For terms including gradients (e.g. pressure force, a function of the gradient of the pressure) we need the gradient of the smoothing function.
- For the pressure term we can use the gradient of a modified smoothing kernel (for details see M. Müller et al.: Particle-based Fluid Simulation for Interactive Applications, 2003):

$$\nabla W_p(\mathbf{x} - \mathbf{x}_i, h) = \begin{cases} -\frac{45}{\pi\, h^6} \left(h - |\mathbf{x} - \mathbf{x}_i|\right)^2 \frac{\mathbf{x}-\mathbf{x}_i}{|\mathbf{x}-\mathbf{x}_i|} & \text{for } |\mathbf{x} - \mathbf{x}_i| \leq h \\ 0 & \text{for } |\mathbf{x} - \mathbf{x}_i| > h \end{cases}$$

Smoothed Particle Hydrodynamics

- The pressure force is then given by

$$\mathbf{F}_i^{pressure} = -\langle \nabla p(\mathbf{x}_i) \rangle = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W_p(\mathbf{x}_i - \mathbf{x}_j, h)$$

- To ensure that momentum is always conserved, this expression was made symmetric in i and j.
- In order to increase the numerical stability of the SPH method, the pressure at the location of a particle is not calculated via the ideal gas equation of state as

$$p = \rho\, k_B\, T = \rho\, k,$$

with k_B being the gas constant and T the temperature, but instead as

$$p = (\rho - \rho_0) k_B\, T = (\rho - \rho_0) k$$

The offset ρ_0 is the rest density and mathematically has no effect on the pressure forces.

Smoothed Particle Hydrodynamics

- We would like a viscous term too, and we define this now.
- The expression for the viscous force with the necessary modifications to make it symmetric is:

$$\mathbf{F}_i^{viscous} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W_v(\mathbf{x}_i - \mathbf{x}_j, h)$$

- Where we use $\nabla^2 W_v(\mathbf{x} - \mathbf{x}_i, h)$ according to M. Müller et al.:

$$\nabla^2 W_v(\mathbf{x} - \mathbf{x}_i, h) = \begin{cases} \frac{45}{\pi h^6} \left(h - |\mathbf{x} - \mathbf{x}_i|\right) & \text{for } |\mathbf{x} - \mathbf{x}_i| \leq h \\ 0 & \text{for } |\mathbf{x} - \mathbf{x}_i| > h \end{cases}$$

Smoothed Particle Hydrodynamics

- This completes our inter-particle, hydrodynamic force definitions.
- Of course other (external) forces like gravity can also be added.
- We then need to integrate our system forward in time, just like the MD example...

Outline

Smoothed Particle Hydrodynamics

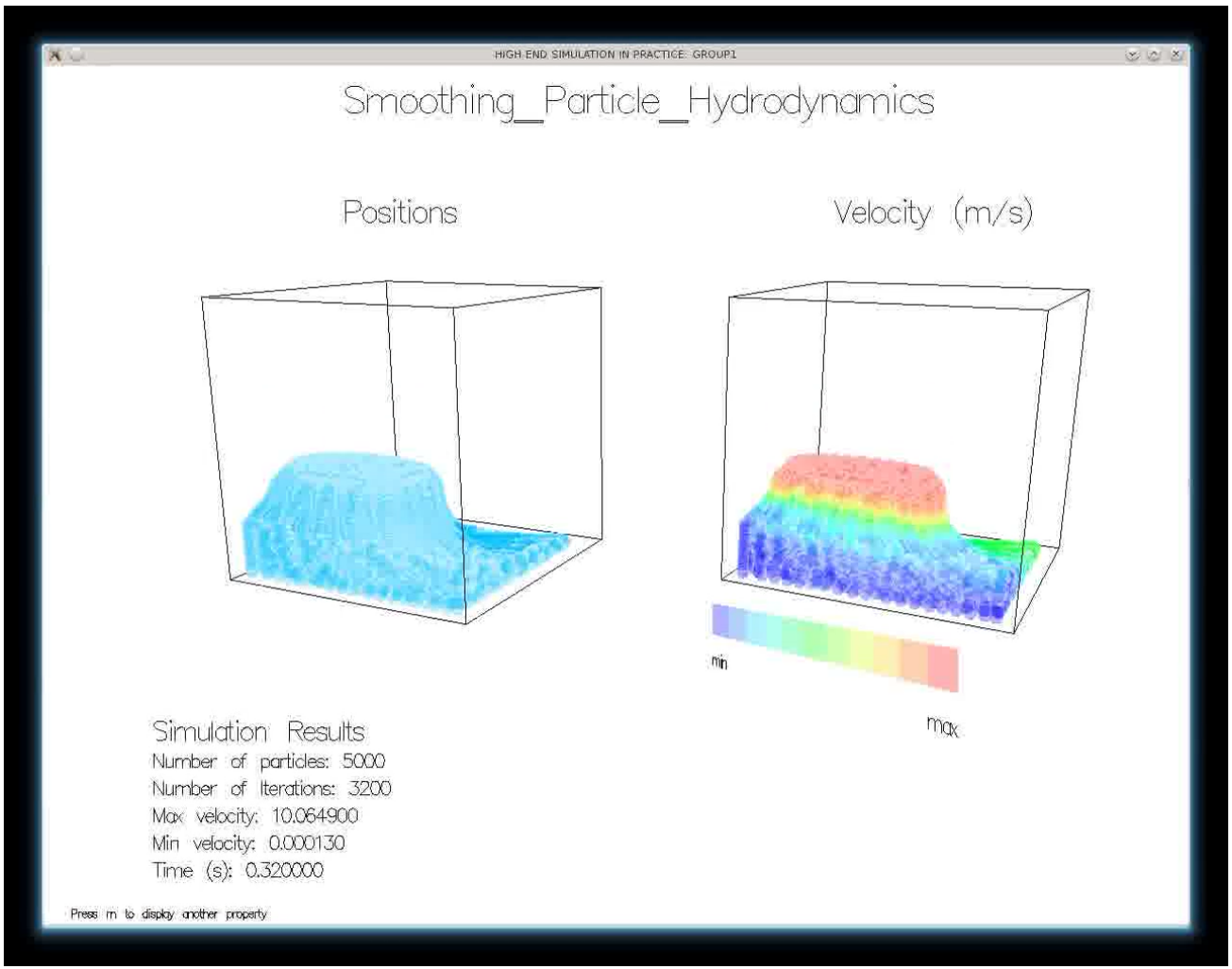
- We'll only use a simple Euler integrator to evolve the system forward in time as the system is rather dissipative anyway

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \Delta t\, \mathbf{v}_i$$
$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t\, \mathbf{a}_i$$

- The Velocity Verlet algorithm we used previously is only correct for forces which are not a function of the velocity, but the viscous force terms in SPH are.

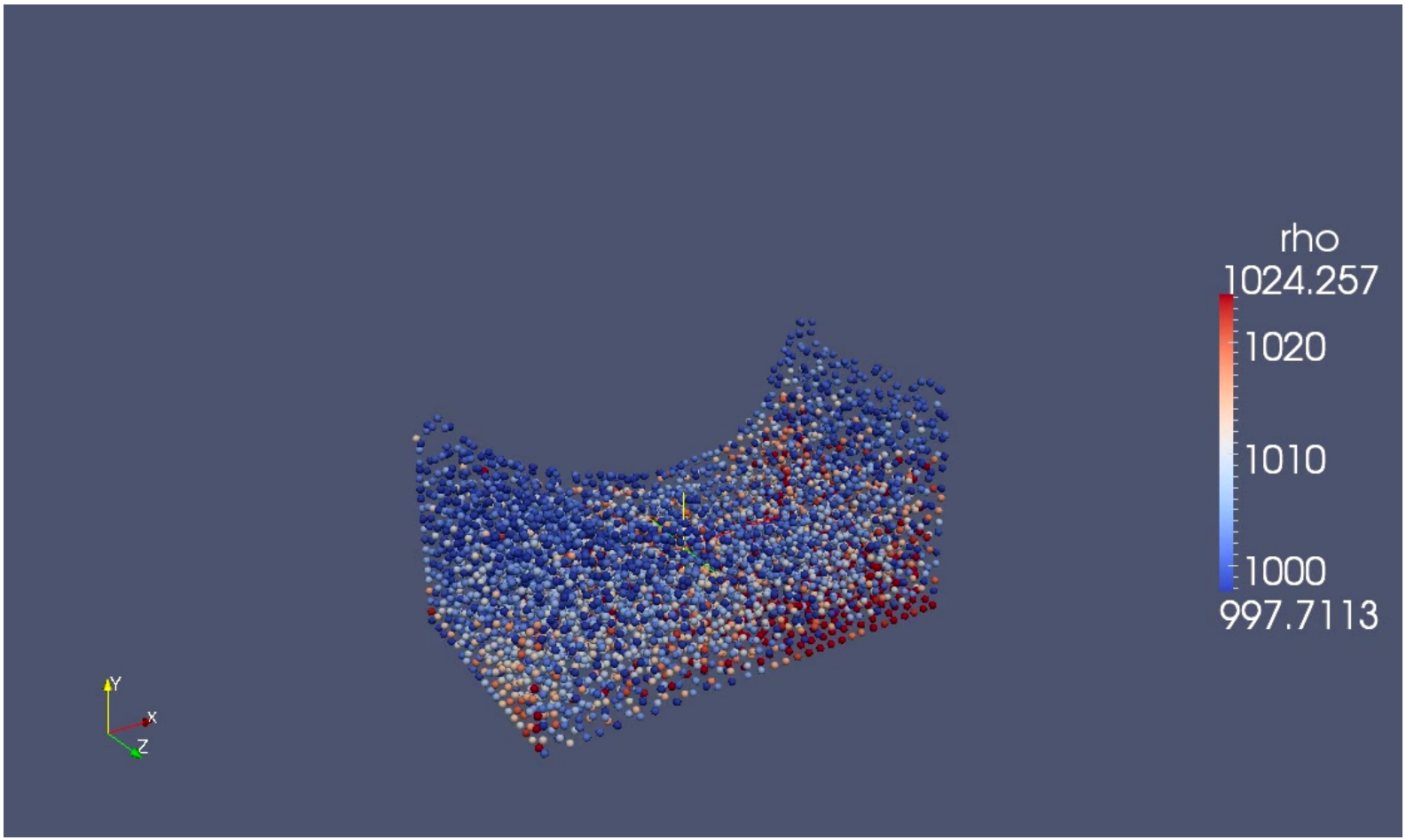
Outline

HESP Project SPH #1



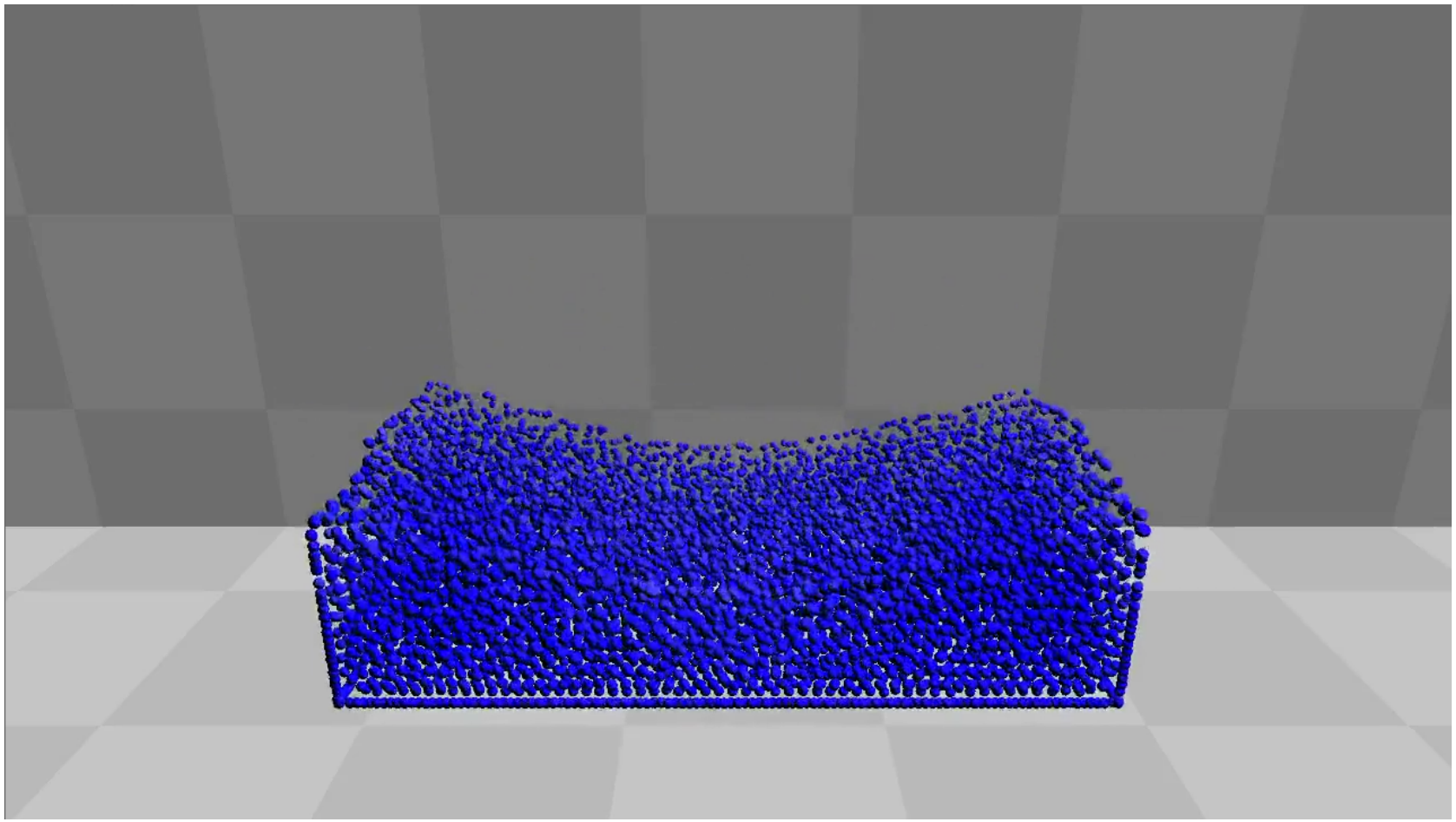
HESP 2013 - Group 1

HESP Project SPH #1



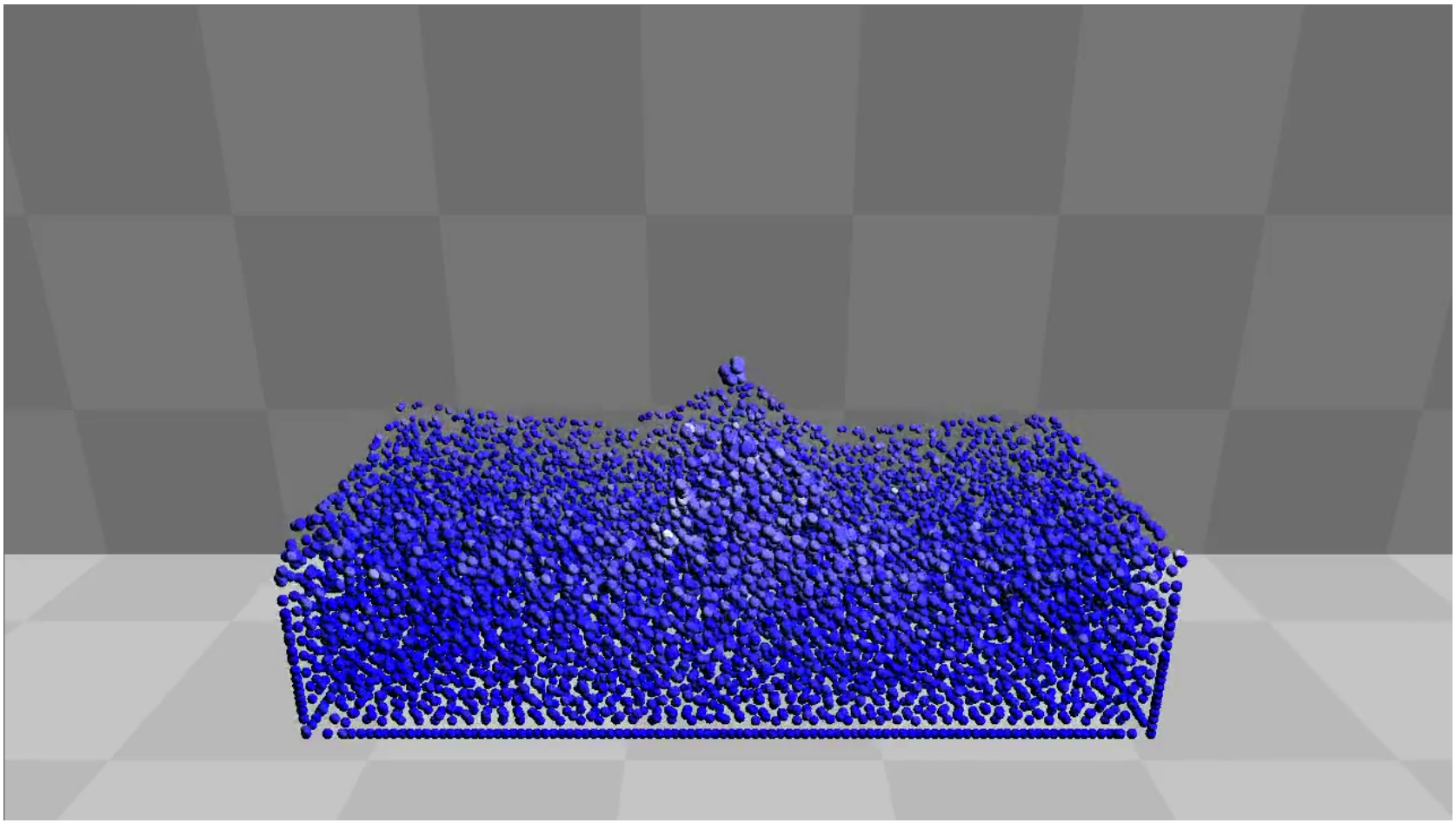
HESP 2013 - Group 1

HESP Project SPH #2



HESP 2013 - Group 2

HESP Project SPH #2



HESP 2013 - Group 2

Outline

Summary

- Today you learned about neighbour lists, an important optimization for any MD implementation.
- In the next exercise you will extend your own MD implementation to make use of this.
- We showed you SPH, basically a modified MD with different force models.
- In the project phase one or more groups might actually implement SPH.