

Molecular Dynamics - Assignment 3

Alex Hocks Jan Hampe Johannes Riemenschneider

Technische Universität München

TUM CIT

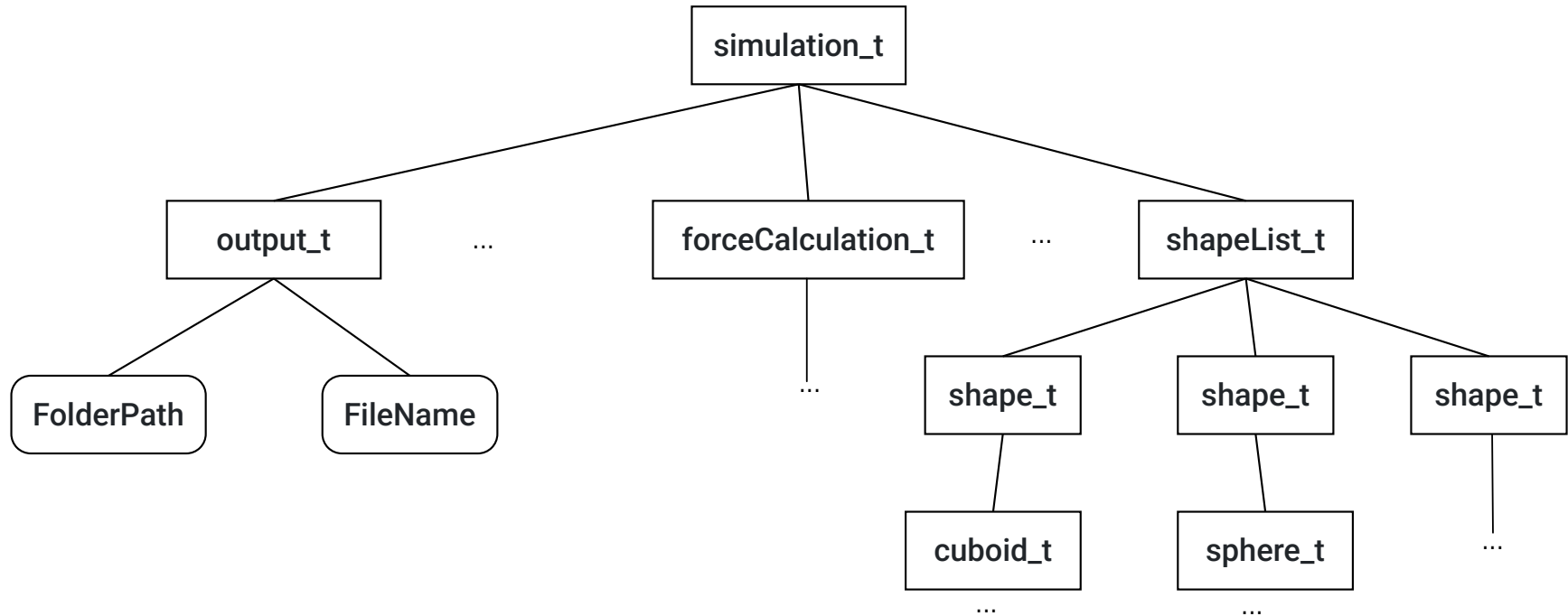
Lehrstuhl für wissenschaftliches Rechnen

9. Dezember 2022



TUM Uhrenturm

XSD- tree-like definition of Datastructures



XSD- Code Snippet

- simulation_t stores all the necessary parameters
- utilizes defined other tree-like Datastructures
- “simulation_t is root“

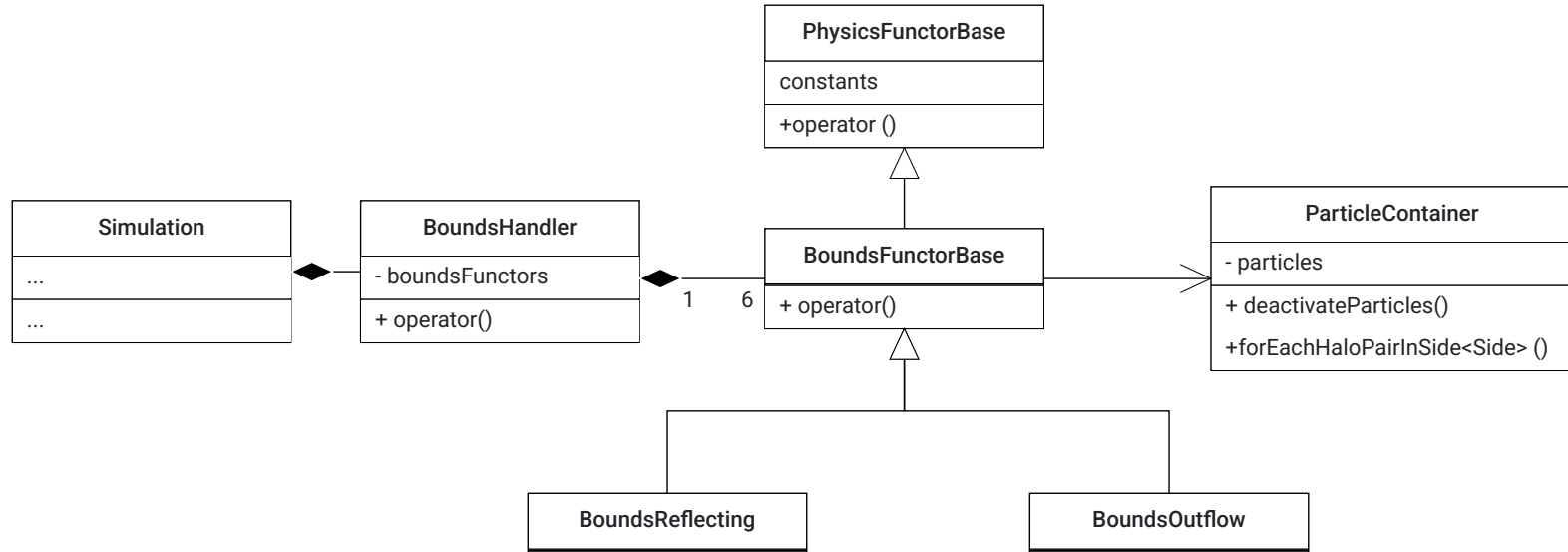
```
<xsd:complexType name="simulation_t">
  <xsd:sequence>
    <xsd:element name="OutputFile" type="output_t" minOccurs="0"/>
    ...
    <xsd:element name="ShapeList" type="shapeList_t"/>
  </xsd:sequence>
</xsd:complexType>
```

XSD- Code Snippet

Trade-off convenience \leftrightarrow complexity in action:

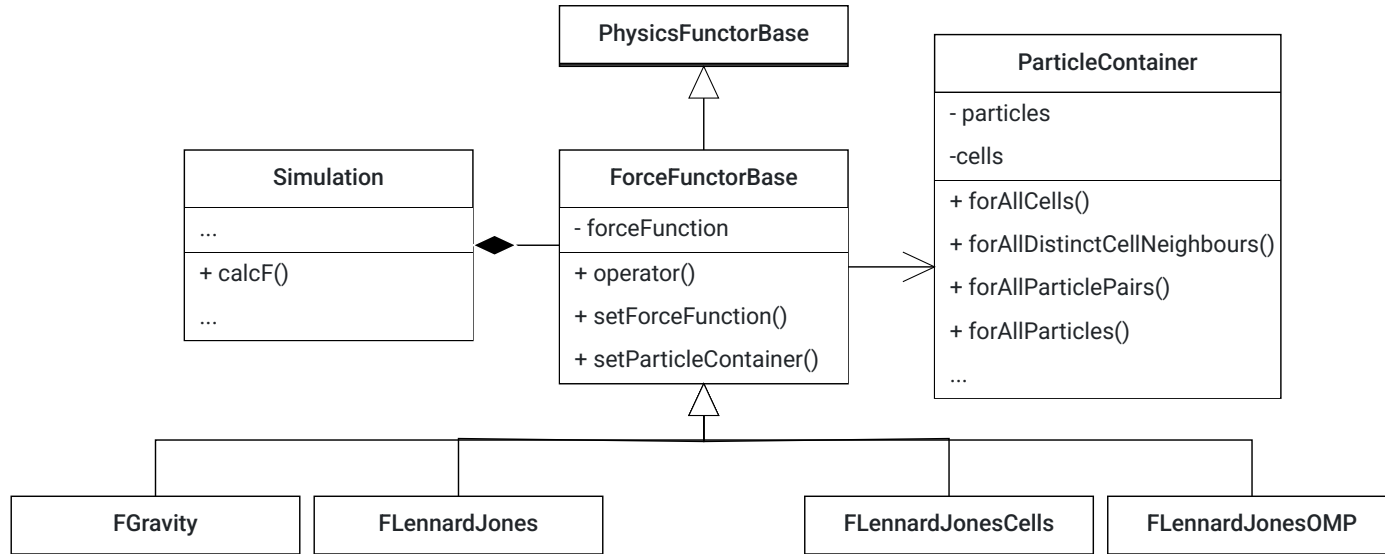
```
<xsd:complexType name="shape_t">
  <xsd:choice>
    <xsd:element name="Particle" type="particle_t"/>
    <xsd:element name="Cuboid" type="cuboid_t"/>
    <xsd:element name="Sphere" type="sphere_t"/>
  </xsd:choice>
</xsd:complexType>
```

Bounds Handling



ForceFuncctors

- Force function used gets determined on runtime via input parameters
- Force functor defines the algorithm (Linked-Cell algorithm/ All-Pairs algorithm) used



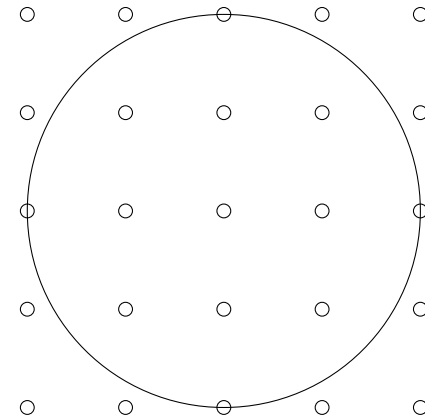
Spheres

Expansion of the Body-struct utilized in Assignment 2

```
enum Shape {cuboid , sphere};
struct Body {
    Shape shape;
    Eigen::Vector3d fixpoint;
    Eigen::Vector3d dimensions;
    double distance;
    double mass;
    Eigen::Vector3d start_velocity;
} ;
```

$$\sqrt{x^2 + y^2 + z^2} \leq r$$

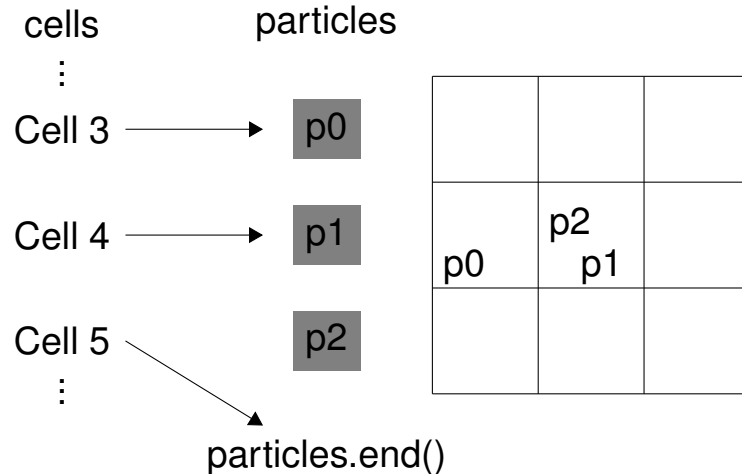
$$\iff x^2 + y^2 + z^2 \leq r^2$$



The Cell Data-Structure - Approach 1

Idea:

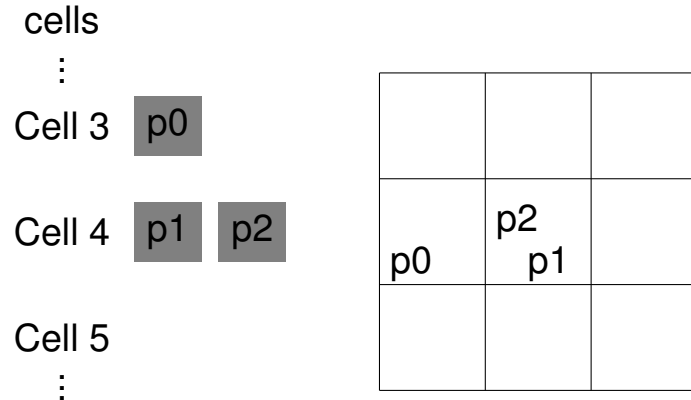
- Sort Particles in accordance to their Cell Position
- save which part of the particles-Vector corresponds to which cell



The Cell Data Structure - Approach 2

Idea:

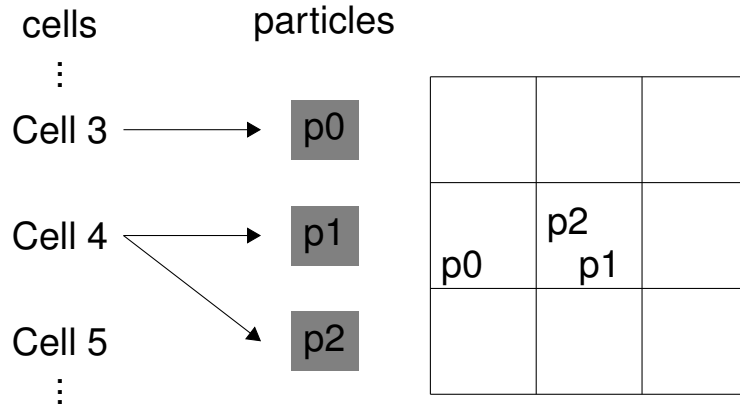
Approach 1.1 stored multiple virtual vectors in one vector \rightarrow let's actually store the particles in vectors corresponding to their cell



The Cell Data Structure - Approach 3

Idea:

- Each Cell only keeps references to their members
- No sorting or copying of entire particles required



Approach Comparison

Approach 1	Approach 2	Approach 3
<ul style="list-style-type: none">+ Easy to implement+ Interface for old Assignments remains unchanged– Expensive struct swaps during sorting+ Direct access to particles for calculations	<ul style="list-style-type: none">+ Easy to implement+ New Implementation of some methods needed– Expensive struct copies with potential reallocs needed+ Direct access to particles for calculations	<ul style="list-style-type: none">+ Easy to implement– Interface for old Assignments remains unchanged+ References are cheap+ Dereferencing needed

Approach Comparison

Approach 1	Approach 2	Approach 3
<ul style="list-style-type: none">+ Easy to implement+ Interface for old Assignments remains unchanged– Expensive struct swaps during sorting+ Direct access to particles for calculations	<ul style="list-style-type: none">+ Easy to implement+ New Implementation of some methods needed– Expensive struct copies with potential reallocs needed+ Direct access to particles for calculations	<ul style="list-style-type: none">+ Easy to implement– Interface for old Assignments remains unchanged+ References are cheap+ Dereferencing needed

In the end we decided to implement approach 3.

ParticleContainer's new methods

```
class ParticleContainer{  
    ...  
  
    void forAllPairsInSameCell(void (*fun) (Particle& p1, Particle& p1));  
  
    void forAllPairsInNeighbouringCell(void (*fun) (Particle& p1, Particle& p1));  
  
    void forAllCells(void (*fun)(...));  
  
    void forAllDistinctNeighbouringCells(void (*fun) (...));  
  
    ...  
}
```

- Functionality of first two methods is sufficient but hard to optimize
- Functionality of last two methods results in higher cohesion, but potential for runtime improvement

Runtime measurement setup

- Particles initiated in a square with varying dimensions

Runtime measurement setup

- Particles initiated in a square with varying dimensions
- Space between square and domain border kept at 10

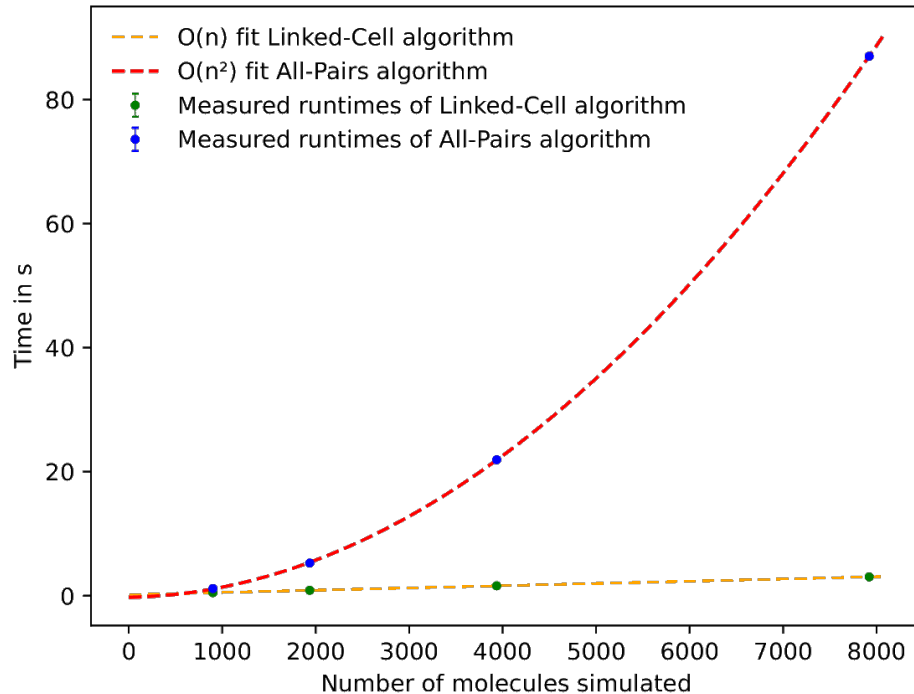
Runtime measurement setup

- Particles initiated in a square with varying dimensions
- Space between square and domain border kept at 10
- $r_{cutoff} = \text{const.}$ (aswell as other variables like ϵ , σ , brown , etc.)

Runtime measurement setup

- Particles initiated in a square with varying dimensions
- Space between square and domain border kept at 10
- $r_{cutoff} = \text{const.}$ (aswell as other variables like ϵ , σ , brown , etc.)
- exact commands to recreate the results are in README and at the end of this presentation

Runtime Comparison of different algorithms



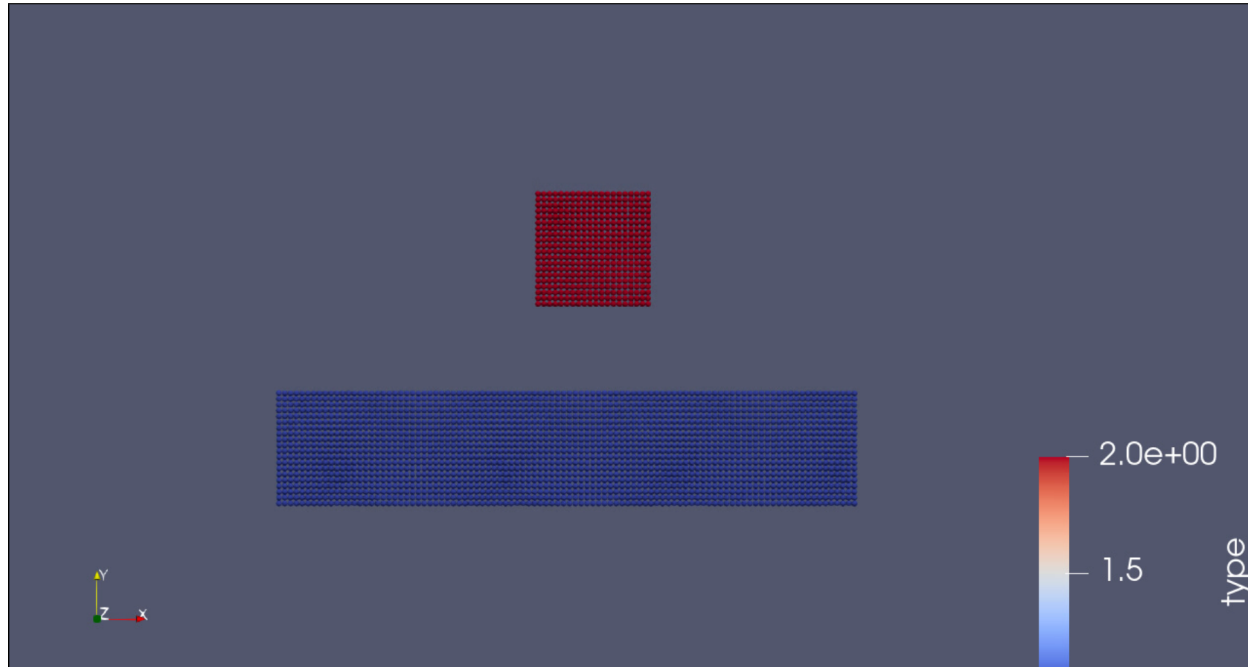
Hardware details:

Ubuntu 20.04 LTS

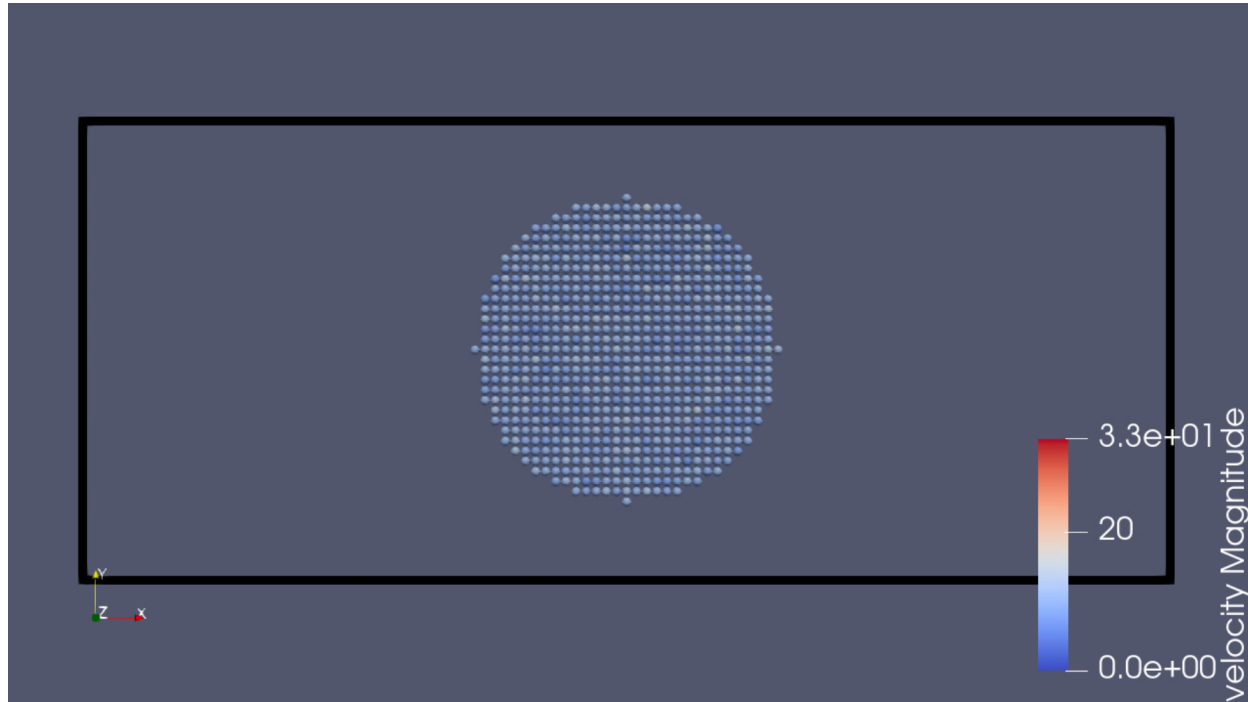
i7 –12700KF @5,0GHz

64GB RAM @ 3200MHz

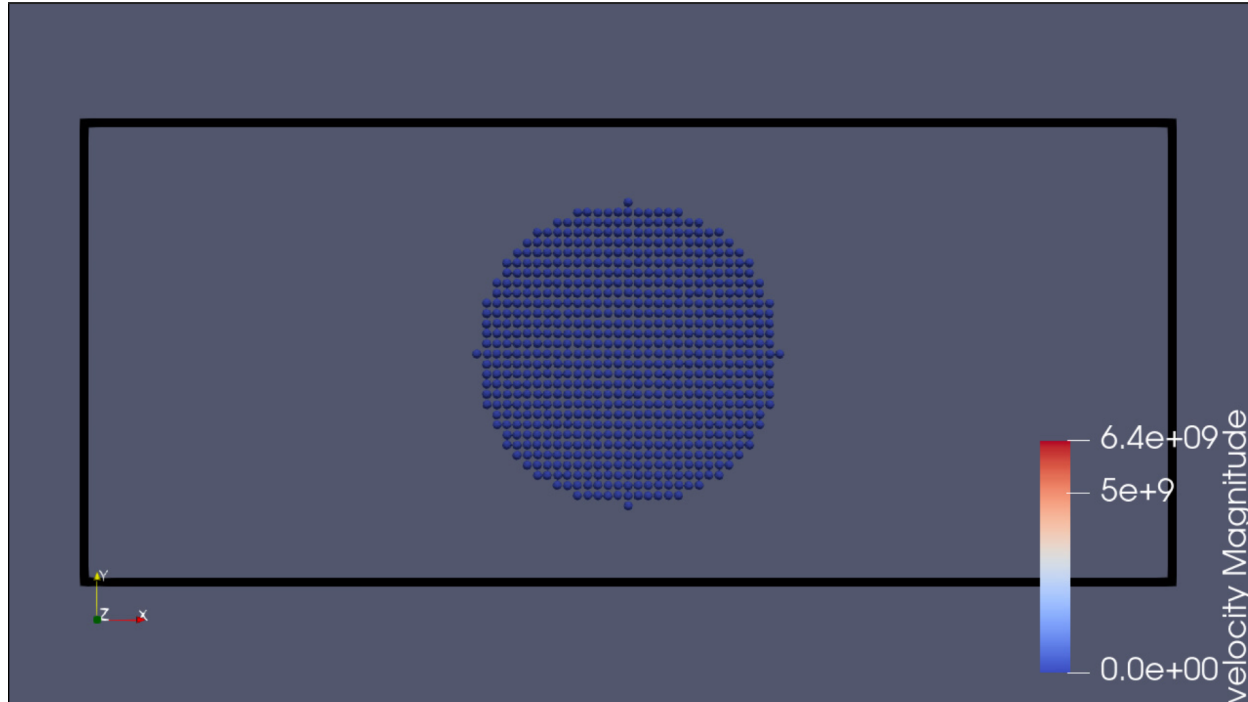
Collision of two bodies



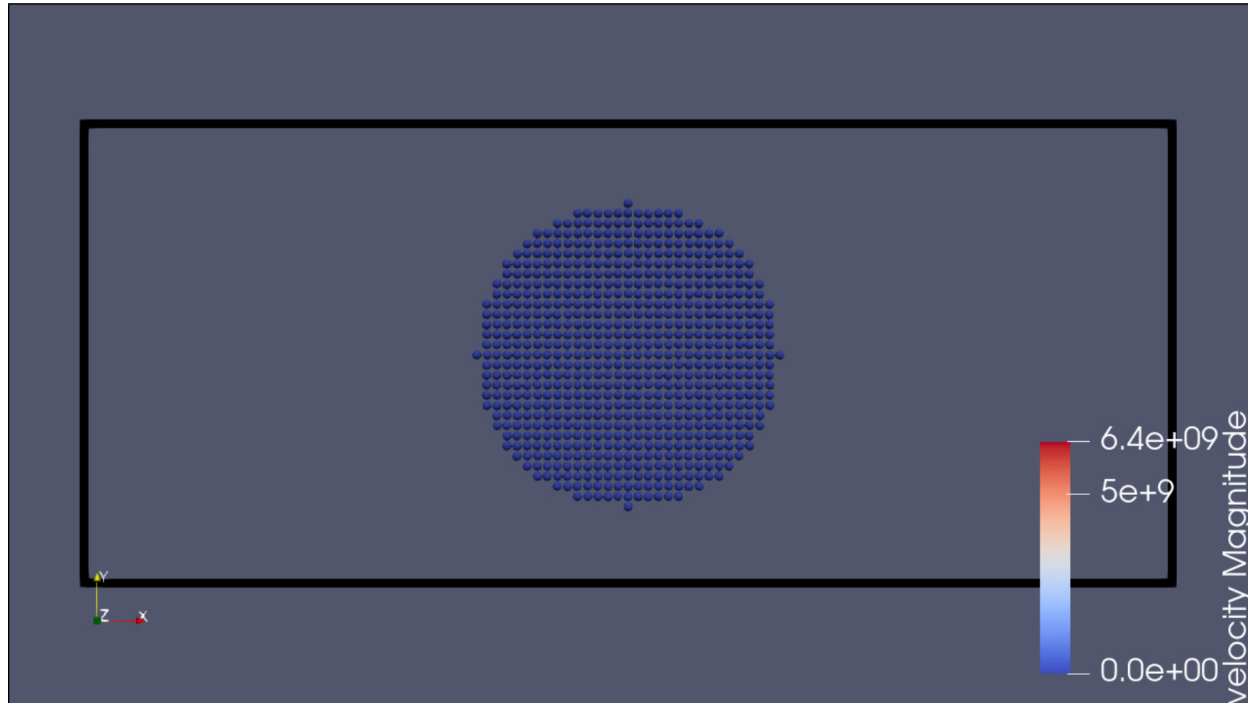
Our Simulation



Increasing Δt

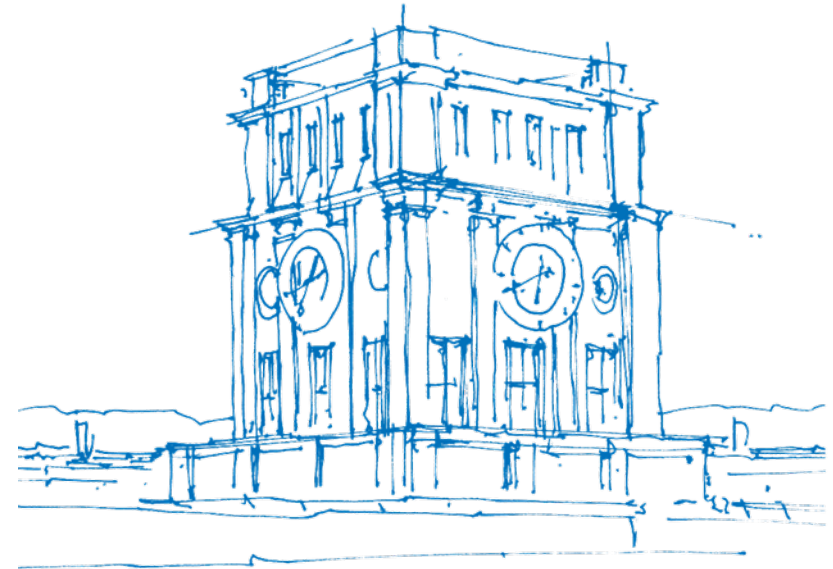


Further increasing Δt



Roadblocks and lessons learned

- Play the objective!
- Getting together and figuring out an elegant solution for one hour is an hour well spent
- Clearly define Interfaces



TUM Uhrenturm

Recreating benchmarks

Setup and compilation:

1. mkdir build output
2. cd build
3. cmake ..
4. make

Example of benchmark commands:

Linked-Cell algorithm:

```
./MolSim ../input/square1.txt -dt 0.0005 -et 0.5 -lc 1 -bbox0 50.0 -bbox1 50.0 -f lennardjonescell -rc 3.0  
-bench file -i 10 > ../output/lc_square1.txt
```

All-Pairs algorithm:

```
./MolSim ../input/square1.txt -dt 0.0005 -et 0.5 -lc 0 -bbox0 50.0 -bbox1 50.0 -f lennardjones -rc 3.0 -bench  
file -i 10 > ../output/ap_square1.txt
```

Recreating benchmarks

- Results can be found in output-folder
- New benchmark files can be written and executed easily
- for other testsizes change input file, output file and bbox-size accordingly

bbox-sizes needed: (50,50), (64,64), (83,83), (109,109)