Scientific Computing (PSE) Molecular Dynamics
Group D

Worksheet 3

02

PROJECT TASKS

Task 1 to 3 -Worksheet 3

XML input

Create an XML Parser using the XSD tool

Linked-Cell algorithm

performance enhancing through simplification of force calculation

Boundary conditions

More realistic approach to behaviour of particles on the borders of a defined domain

TIMELINE

Linked-cells algorithm

Implementing another type of more performant calculation where only necessary calculations are made

Boundary conditions & first simulations

Reflective behaviour on “walls” or outflowing of particles that leave the domain

XML Parser connection instead of txt. file based

XML files for all worksheet simulations, parser and config generator that extracts all necessary information

We implemented the functional/simulation part quite parallel to the XML parsing part without connecting it right then, but just at the end when all new implementations were running and tested for proper functionality

04

SOMULATION OF FALLING DROP - WALL

Task 4

Particle generator

became a lot simpler and slimmer because of the new XML input classes

Disc as drop representation

A new simulation body of disc is introduced

Simulation

perform 2D simulation of the collision with the given parameters

Helpful XSD annotations for us

```
<xs:choice>
</xs:choice>

    <xs:element name="cuboid">
    </xs:element>

        <xs:element name="disc">
        </xs:element>

            <xs:element name="sphere">
            </xs:element>


                <xs:atrribute name="sigma"/>


                    <xs:sequence/>
```

XML Parsing

Advantages

ready to use library XSD that generates corresponding cxx and hxx files

integrated input validation through good schema definition

Key features

different elements and types that allow for a very specific definition of valid inputs and little overhead

```xml
<?xml version="1.0" encoding="UTF-8"?>

<Data xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
      xs:noNamespaceSchemaLocation="../src/lib/simulator/io/xml_reader/SimulationInputSchema.xsd">

    <header base_name="Simulation of a falling drop - Wall" output_file_name="falling_drop_simulation"
            output_frequency="100" t_end="10" seed="1234"/>

    <linked_cells>
        <domain_size>
            <value>0</value>
            <value>180</value>
            <value>90</value>
        </domain_size>
    </linked_cells>

    <lennard_jones>
        <settings delta_t="0.00005" sigma="1" epsilon="5" mass_m="1"
                  distance_h="1.1225" brown_motion="0.1" cutoff_radius="0"/>
        <discs>
            <Disc>
                <coordinate>
                    <value>60</value>
                    <value>25</value>
                    <value>0</value>
                </coordinate>
                <velocity>
                    <value>0</value>
                    <value>-10</value>
                    <value>0</value>
                </velocity>
                <radius>15</radius>
            </Disc>
        </discs>
    </lennard_jones>
</Data>
```

header

particle model

settings

disc

TASK 1

XML Input

XML file for disc

choice of particle model
choice of force calculation model

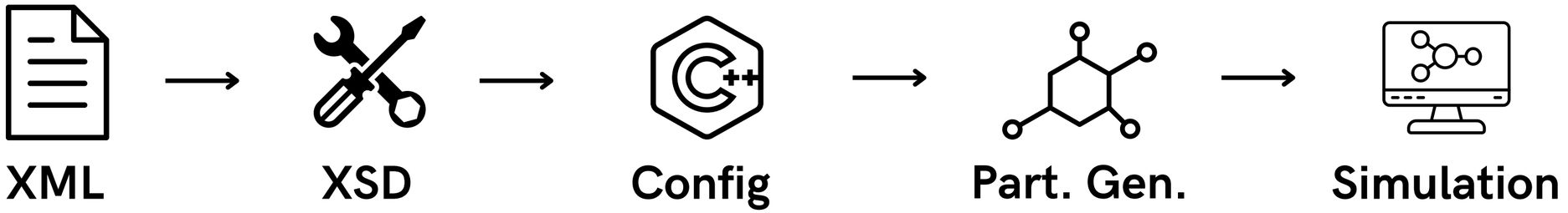
header and settings for globally defined simulation variables

multiple discs possible if necessary

Simulation Config

Class that stores all parameters for the simulation using the following mechanic

```
config.sigma = data->lennard_jones()->settings().sigma()
config.end_time = data->header->t_end()
```

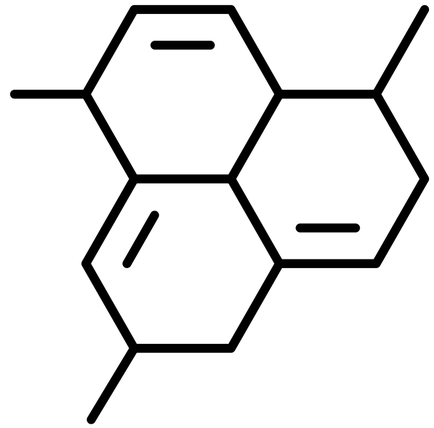


Linked-cells algorithm

New particle container
implements new linked-cell algorithm so that previous functionality is preserved

Container for iteration
to allow iteration over boundary and halo particles and enable their deletion

XML input
adapted the XSD schema accordingly to pass domain size and cutoff radius



TASK 3

Linked-cell algorithm

Boundary Conditions

Outflow condition

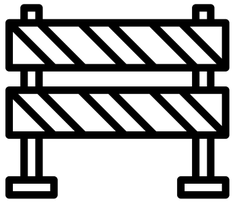
to allow iteration over boundary and halo particles and enable their deletion

Reflection condition

to allow iteration over boundary and halo particles and enable their deletion

XML input

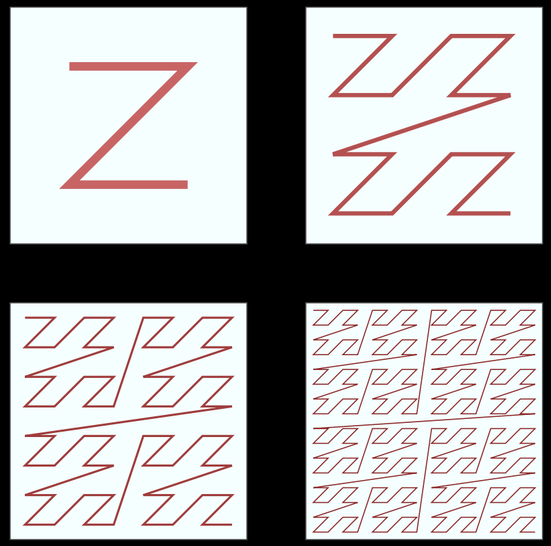
adapted the XSD schema accordingly to pass the condition choice and necessary boundary parameters for the simulation



Testing

Ensured proper functionality by testing each boundary condition for its stability

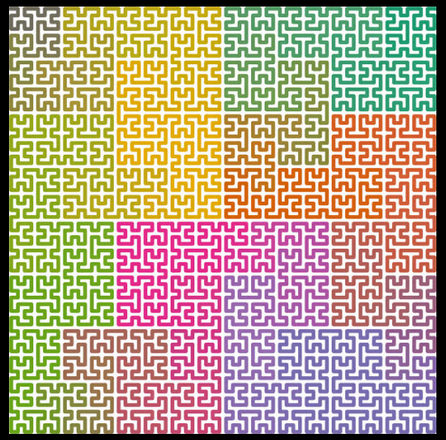
OPTIMZATION



Z-curves

Tried out the implementation of Z-curves for performance enhanced iteration over arrays and therefore better runtime benchmarks

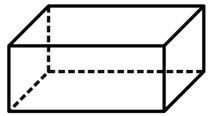
Mathematical approach of efficient memory / cache access



Hilbert curves

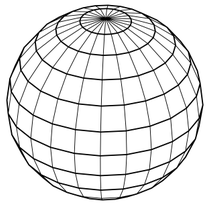
Since Z-curves did not deliver the expected result we did not implement Hilbert curves (which are more costly to calculate)

Cuboid

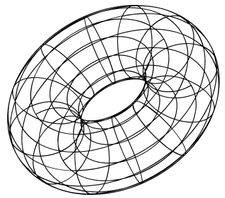


Disc

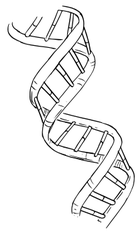
Sphere



Torus



Double Helix



DIFFERENT BODIES

Worksheet 2 & 3

Cuboid & Disc

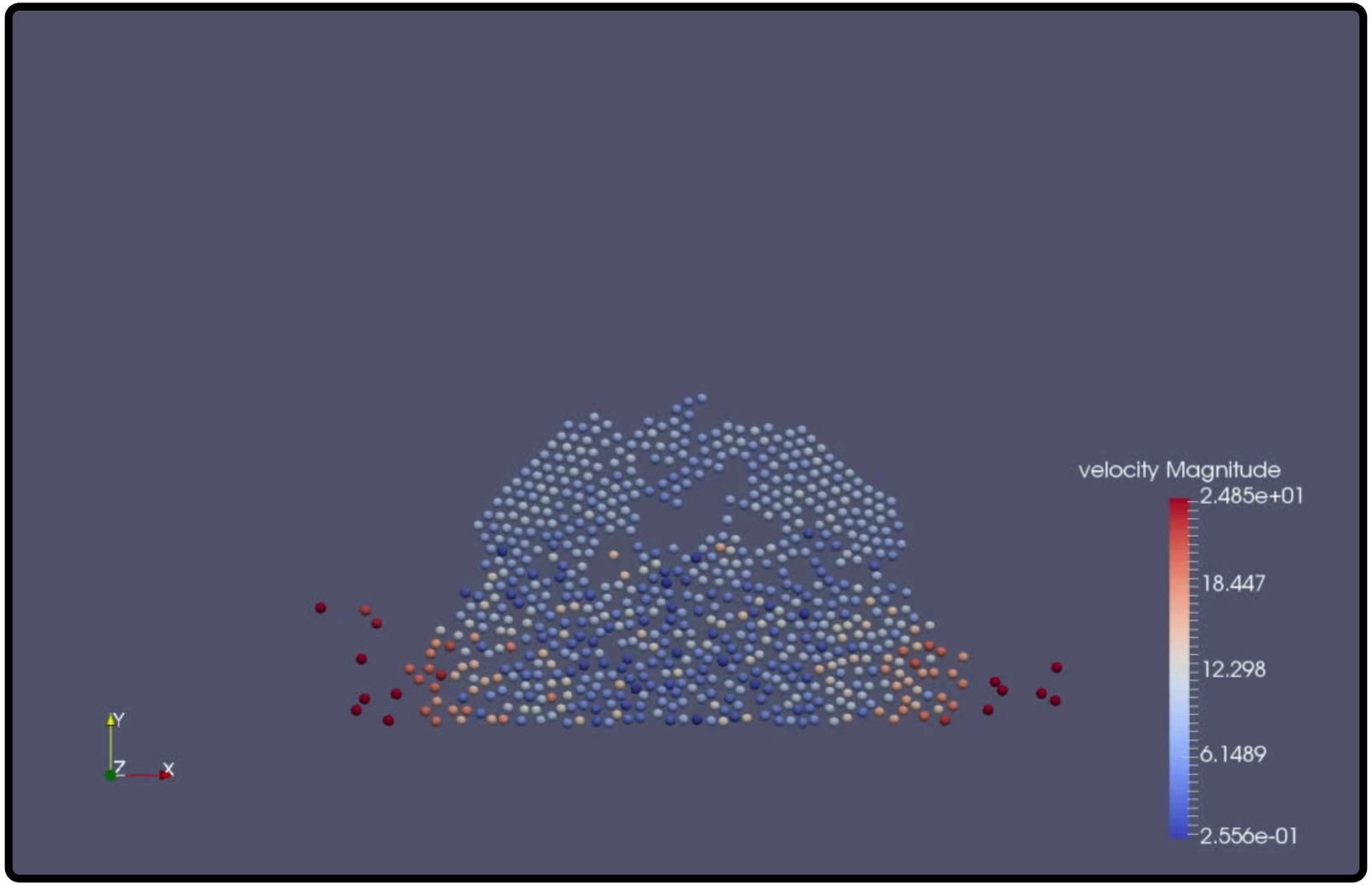
Our addition

Spheres, Tori and Helices

SIMULATION 1

Velocity magnitude as color code

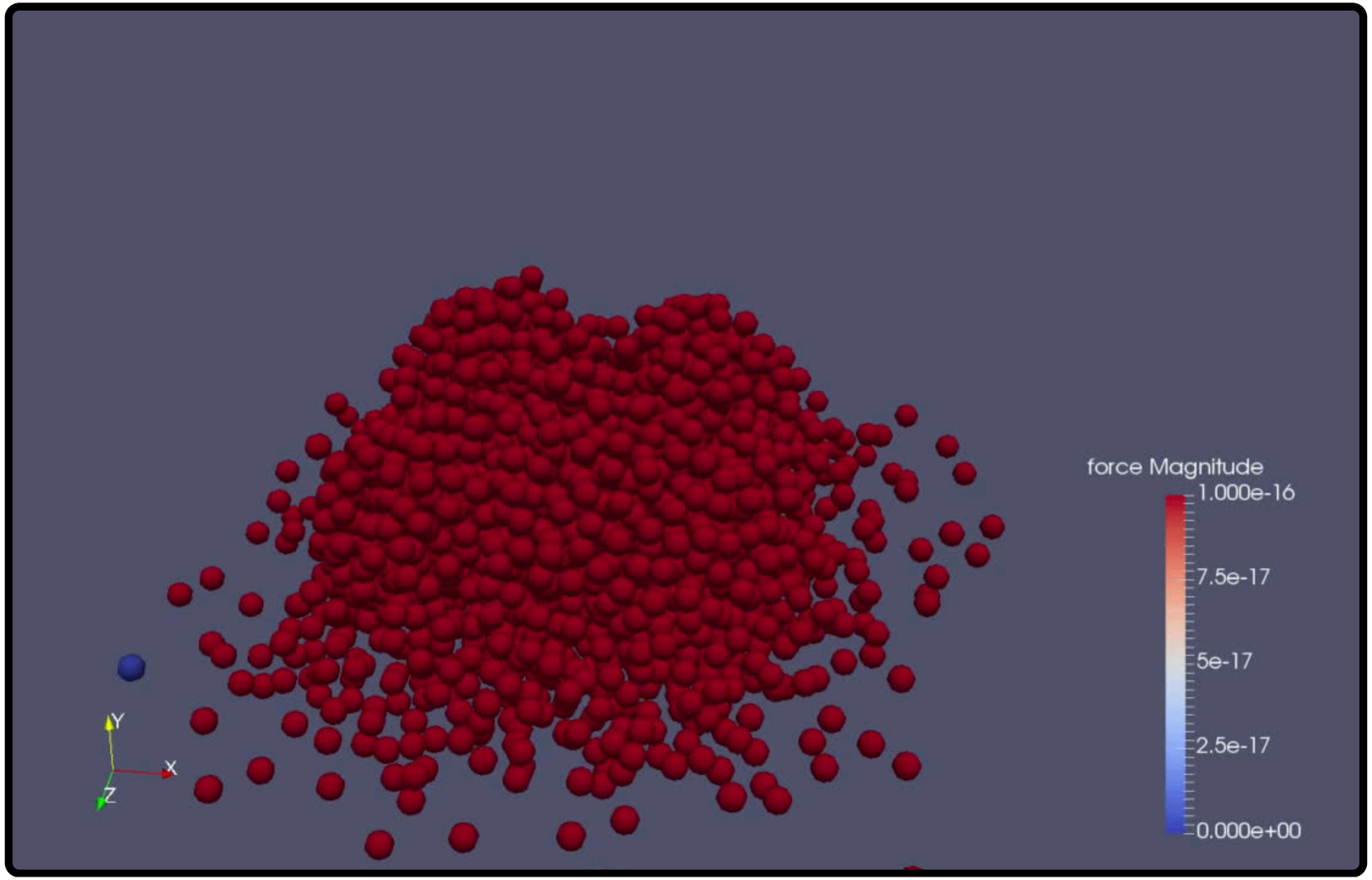
12



SIMULATION 2

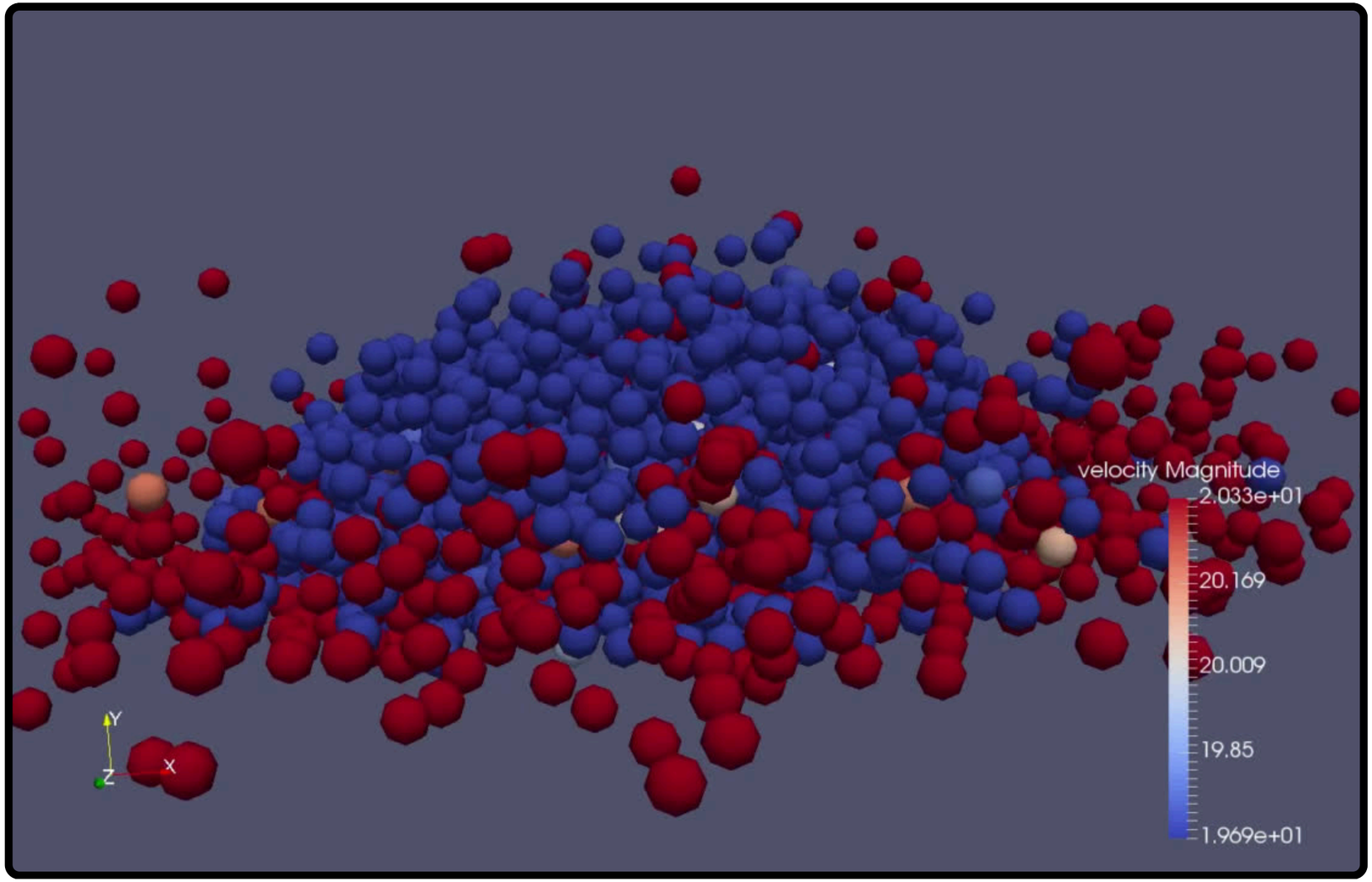
Force magnitude as color code

13



SIMULATION 3

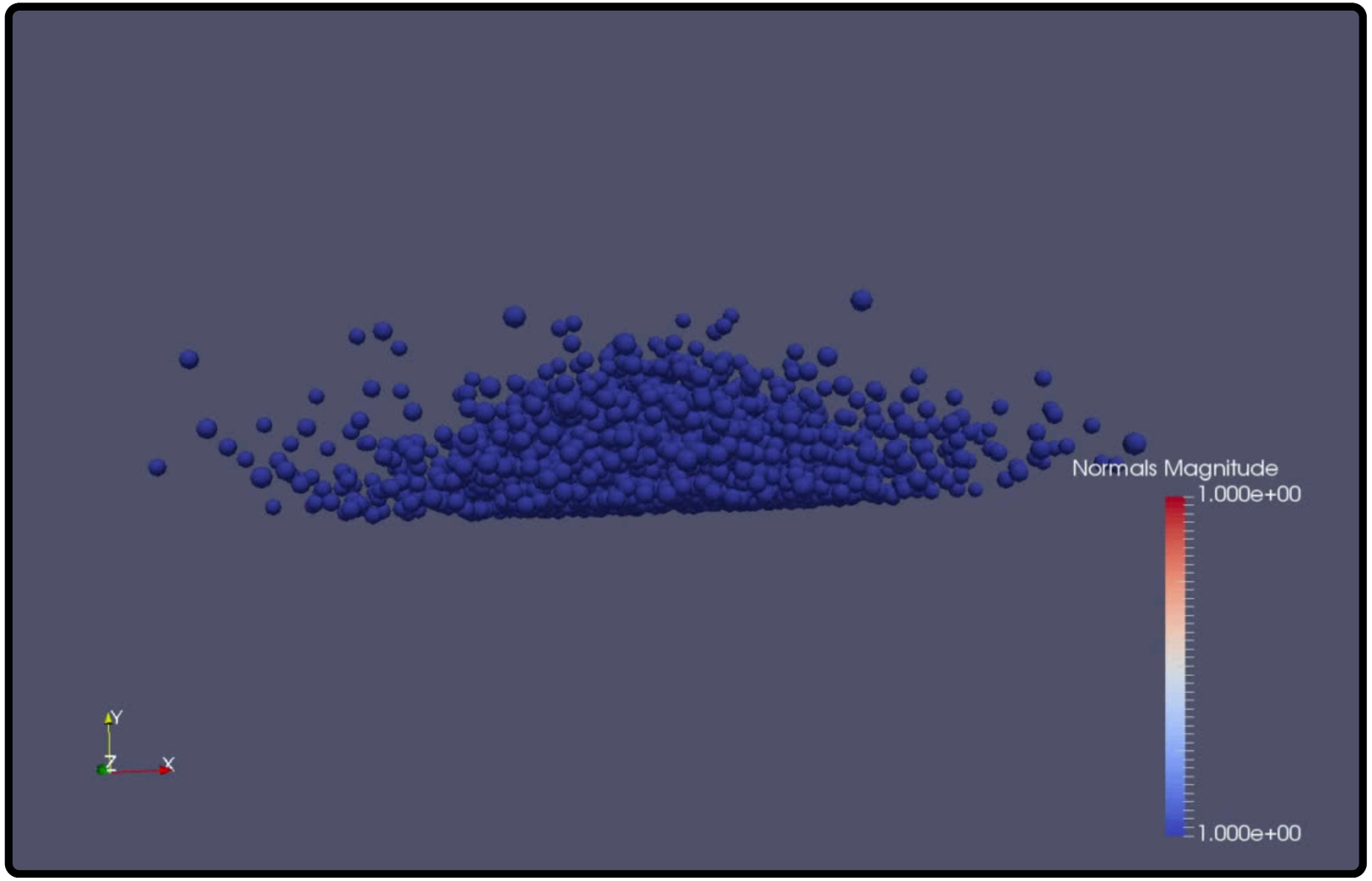
Velocity magnitude as color code



SIMULATION 4

Normals magnitude as color code

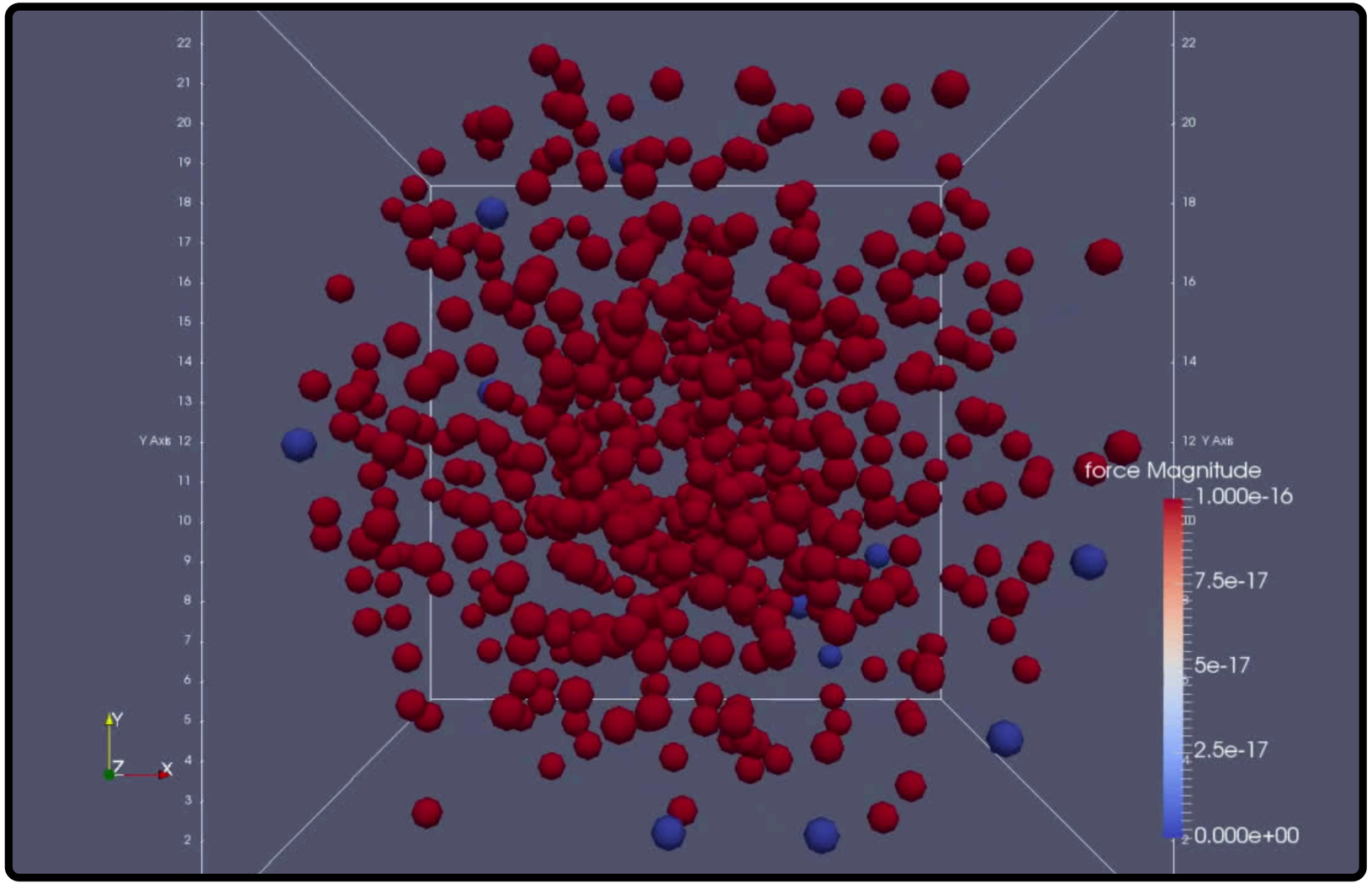
15



SIMULATION 5

Force magnitude as color code

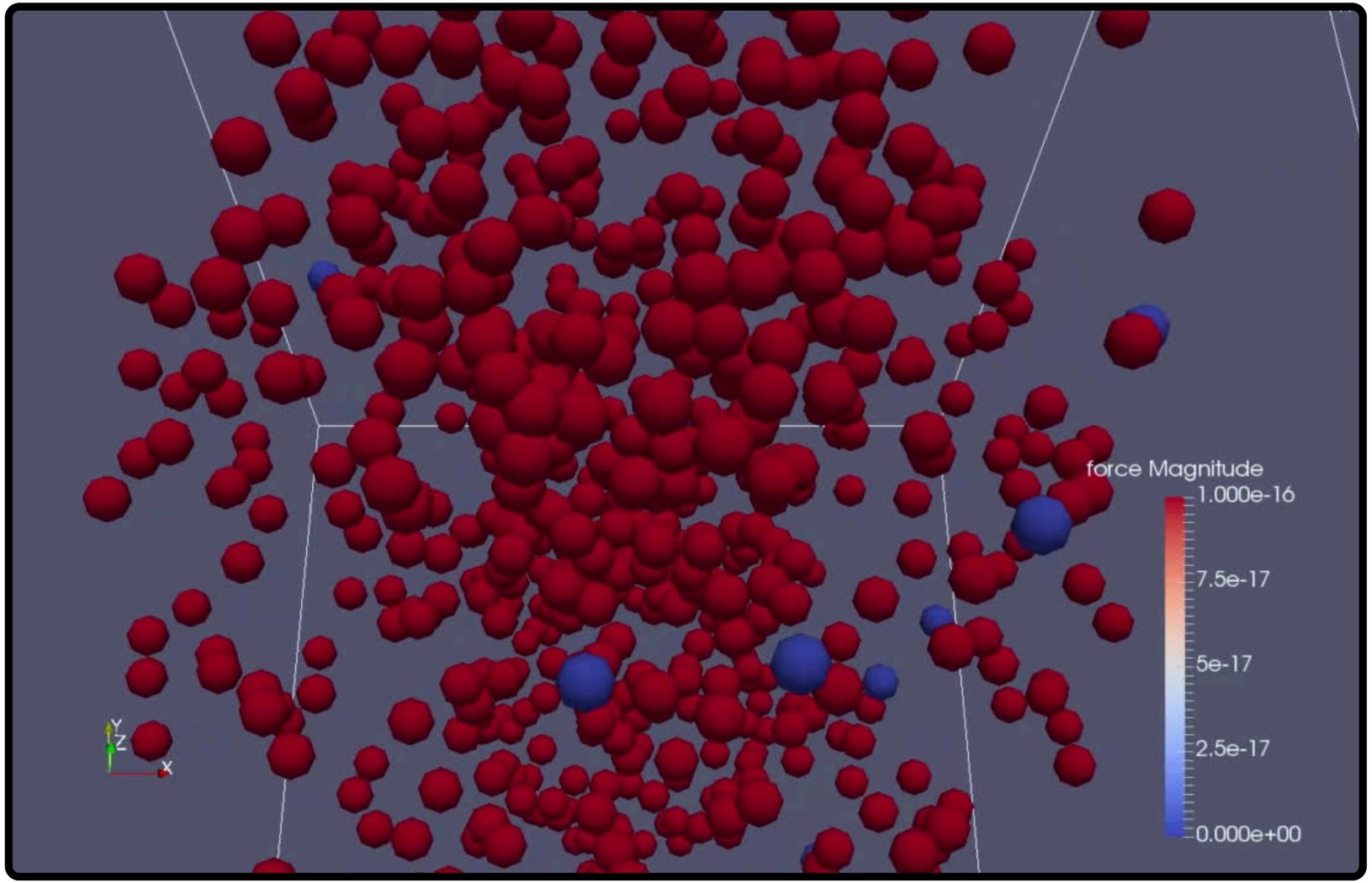
16



SIMULATION 6

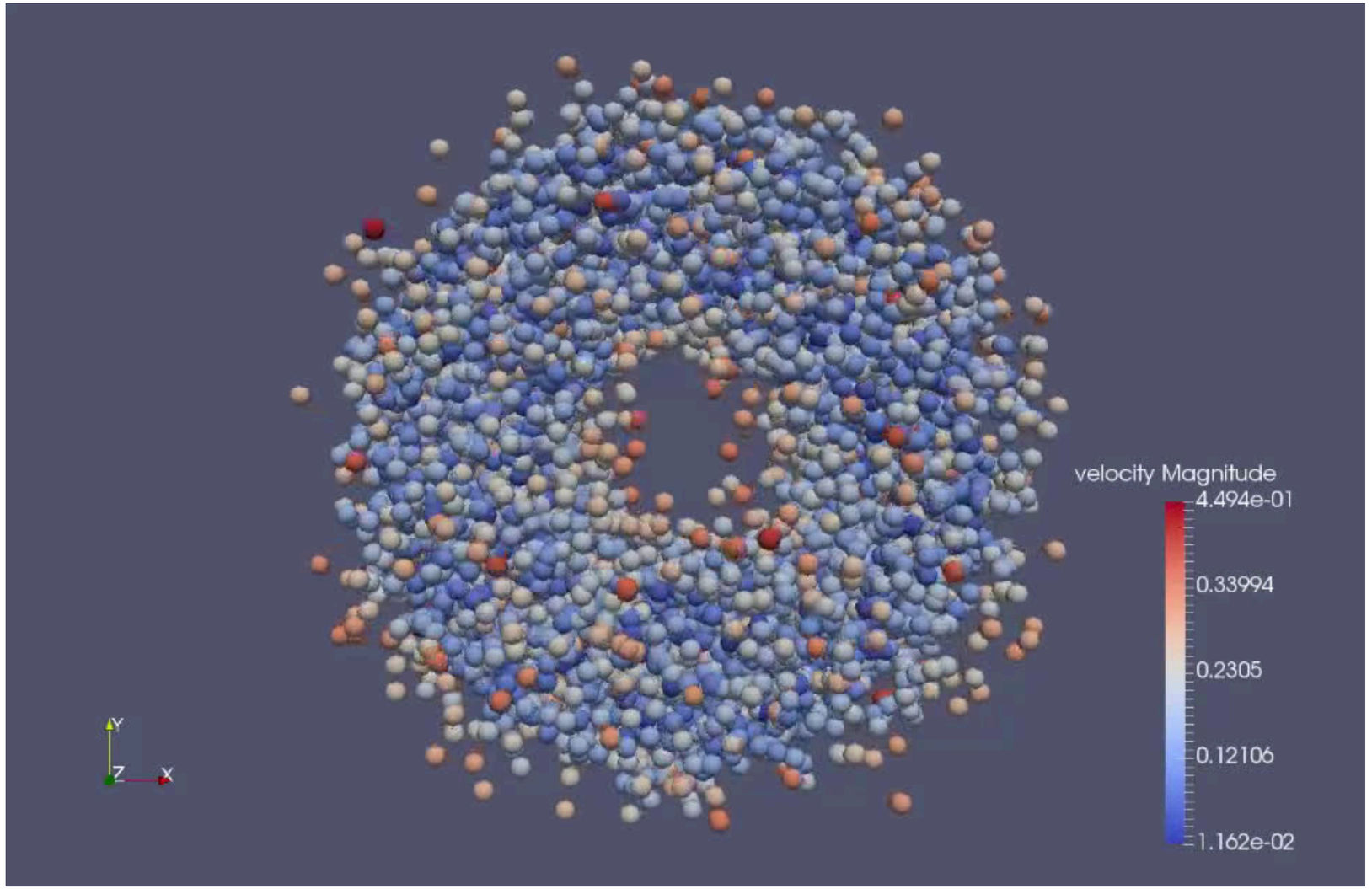
Force magnitude as color code

17



SIMULATION 7

Velocity magnitude as color code



SIMULATION TIMES

Hardware: Intel i5-12600K 10K/16T, 32GB RAM

NAIVE APPROACH

01:22:23 min
1000 molecules

05:12:21 min
2000 molecules

20:53:02 min
4000 molecules

LINKED CELLS

00:09:50 min
1000 molecules

00:05:21 min
2000 molecules

00:38:53 min
4000 molecules

SUMMARY & LEARNINGS

TASK 1

XML parsing single big time investment to set up schema, config generator and differentiate between choices and types, compared to text or command line input -> but still worthwile for the ease of use and adaptabilty

TASK 2

as expected, big performance improvements because calculations are only done when considered necessary. Realistic enough approach to be considered accurate but still quickly computable

TASK 3

Very neat looking simulations, a lot more possibilities to test out, also with other bodies like spheres, helices and tori

REFACTORING

In parallel, we also did some refactoring, cleaned up and deleted old code (like the old text parser) to keep the project slim

FUTURE IMPROVEMENTS

PERFORMANCE

Parallelization, use of multi-core and threading
Streamline calculation

Try out Hilbert curves nonetheless or similar memory-efficient approaches

TESTING

More edge case tests
Writing tests while implementing new methods

REFACTORING

Reduce line of codes by replacing old code with new implementations
and deleting old features/implementations

Scientific Computing (PSE) Molecular Dynamics
Group D

Johannes H. | Julius K. | Tim S.

THANK YOU FOR LISTENING