

01

Scientific Computing (PSE) Molecular Dynamics  
Group D

# Worksheet 2



02

# PROJECT TASKS

Task 1 to 3 -Worksheet 2

## Unit Testing

Integrated testing in CMake using the googletest suite

## Continuous Integration

CI tools provided by GitHub

## Logging

adding meaningful log messages in runtime using spdlog

## Testing and CI

googletest integration for previous methods

## Restructuring and Logging

to make the project ready for new functionality

## New implementation

File parser, Lennard-Jones forces and simulation

## TIMELINE

We used most of the time for refactoring, restructuring, testing, logging and performance improvements so that the simulation task itself was fairly easy to implement and quickly done

04

# Particle generator

particle generator that can initialize the simulation

## Lennard-Jones forces

The interaction between two molecules  $i$  and  $j$

# COLLISION SIMULATION TASK

## Simulation

perform 2D simulation of the collision with the given parameters

Task 4

# **EXPECT\_**

`EXPECT_TRUE(cond1)`

`EXPECT_EQ(val1, val2)`

`EXPECT_NE(val1, val2)`

`EXPECT_DOUBLE_EQ(val1, val2)`

**VS.**

# **ASSERT\_**

`ASSERT_FALSE(cond1)`

`ASSERT_GE(val1, val2)`

`ASSERT_FLOAT_EQ(val1, val2)`

`ASSERT_ANY_THROW(statement)`

# **GOOGLETEST**

Advantages

easy to use unit testing with CMake  
integration

import via FetchContent

Key features

ready-to-use framework, easy setup,  
automatic executing with ctest  
asserts() and expects()

# Gravitational force testing

L2-norm =  $\sqrt{x^2 + y^2 + z^2}$  = Euclidean norm

Multiplied mass of particles = mass1 \* mass2 = 1005.34

L2-norm)<sup>3</sup>

compute actual forces (by hand)

```
EXPECT_TRUE(calculated_force[0] - actual_force[0] < 0.0000001)
```

```
EXPECT_EQ(calculated_force, zero_force)
```

# Lennard-Jones force testing

compute displacement of vectors  $r_{ij} = ||x_i - x_j||$

manually compute the Lennard-Jones potential by hand using the formula

manually compute the Lennard-Jones forces using the formula

```
EXPECT_DOUBLE_EQ(calculated_force[0], actual_force[0])
```

```
EXPECT_TRUE(calculated_force[0] - actual_force[0] < 0.0000001)
```

```
EXPECT_TRUE(std::isnan(force))
```

# REFACTORING

## Structure

strongly adjusted the project structure to fit our needs and make the project easier to manage and extend in the coming weeks

split CMakeLists into several files with each one only configuring a specific part like testing, logging, etc. with one high-level list

## Naming

snake\_case style naming for variables, functions etc.

```
> ⌂ .github
> ⌂ build-debug
> ⌂ build-release
> ⌂ cmake
< hooks
  ⌂ pre-commit
< input
  ⌂ eingabe-collision.txt
  ⌂ eingabe-sonne.txt
< libs
  > ⌂ libxsd
  > ⌂ spdlog
< src
  > ⌂ bin
  > ⌂ lib
< submission
  > ⌂ worksheet1
  > ⌂ worksheet2
> ⌂ test
> ⌂ toolchains
  ⌂ .clang-format
  ⌂ .clang-tidy
  ⌂ .gitignore
  ⌂ .gitmodules
  ⌂ CMakeLists.txt
  ⌂ Doxyfile
  ⌂ README.md
> ⌂ External Libraries
> ⌂ Scratches and Consoles
```

# LOGGING

## Log Functions

```
spdlog::info("Message");
```

**vs.**

## Log Macros

```
SPDLOG_INFO("Message")
```

Advantages

Fast, header-only logging library for C++

Key features

Performance, ease of use, flexibility

## In use

```
switch (model) {  
    case physics::ForceModel::LennardJones: {  
        spdlog::info( msg: "Processing Lennard-Jones model.");  
        auto cuboids :vector<...> = *parse_cuboids( & input_buf);  
        spdlog::debug( fmt: "Parsed {} cuboids.", cuboids.size());  
        auto p :vector<Particle> = generate_cuboids(cuboids, config->seed);  
        spdlog::debug( fmt: "Generated {} particles for Lennard-Jones model.", p.size());  
        particles = ParticleContainer( & p);  
        break;  
    }  
}
```

## For performance tracing / timings

SPDLOG\_LEVEL=ON

SPDLOG\_LEVEL=OFF

# LOGGING

## Advantages

Fast, header-only logging library for C++

## Key features

Performance, ease of use, flexibility

# MolSim Usage

via command line

```
--output      -o    specify output file  
--input       -i    specify input file  
--help        -h    open usage page  
--delta_t     -d    set delta_t  
--start_time  -s    set start_time  
--seed         -r    set seed for rng  
--end_time    -e    set end_time  
--io_interval -l    set plot interval, 0  
disables plotting
```

# Running simulations...

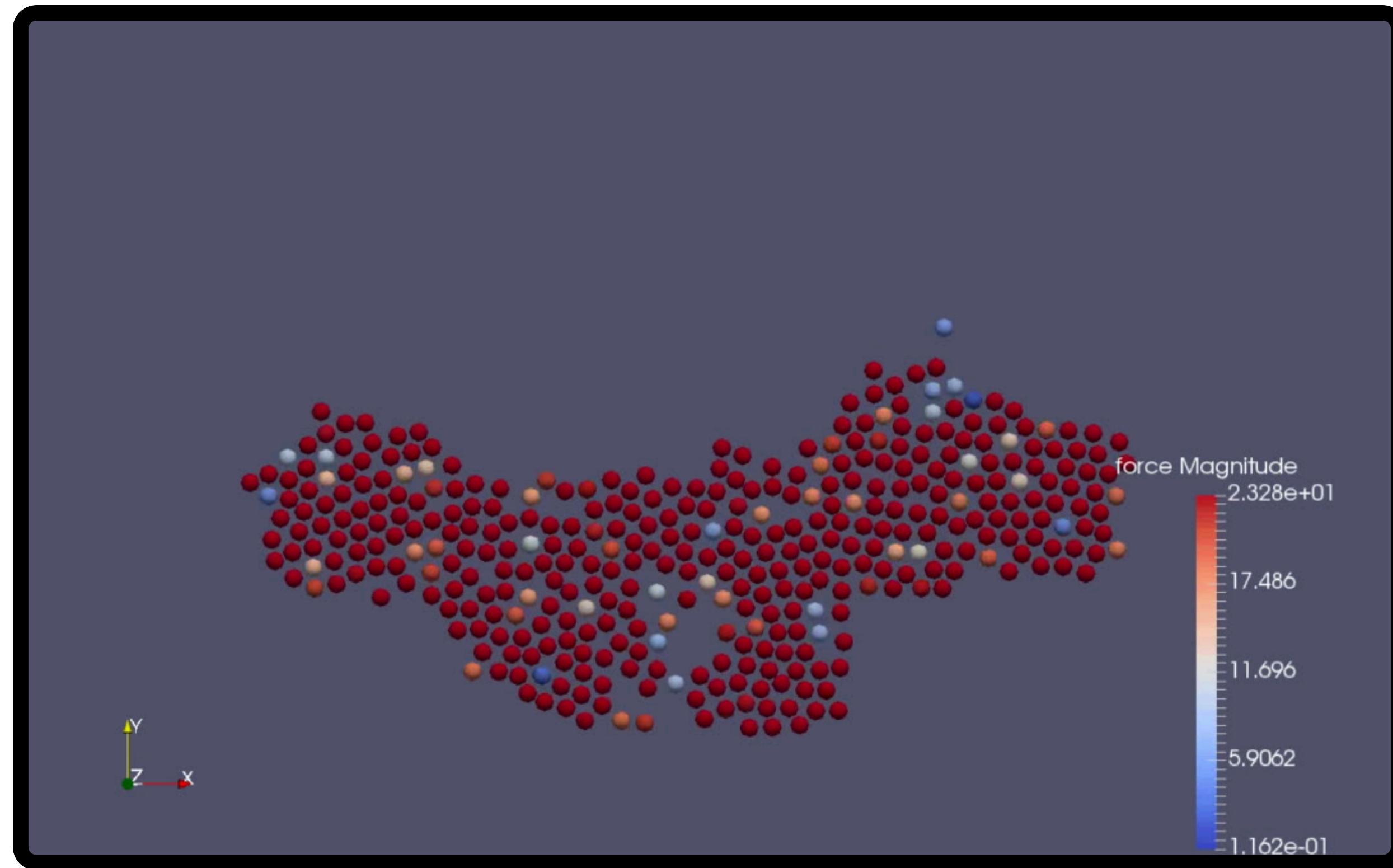
**for simulation on worksheet**

```
Call: ./MolSim  
      -o LennardJones  
      -i "../input/eingabe-collision.txt"  
      -d 0.0002  
      -e 5
```

# SIMULATION 1

Force magnitude as color code

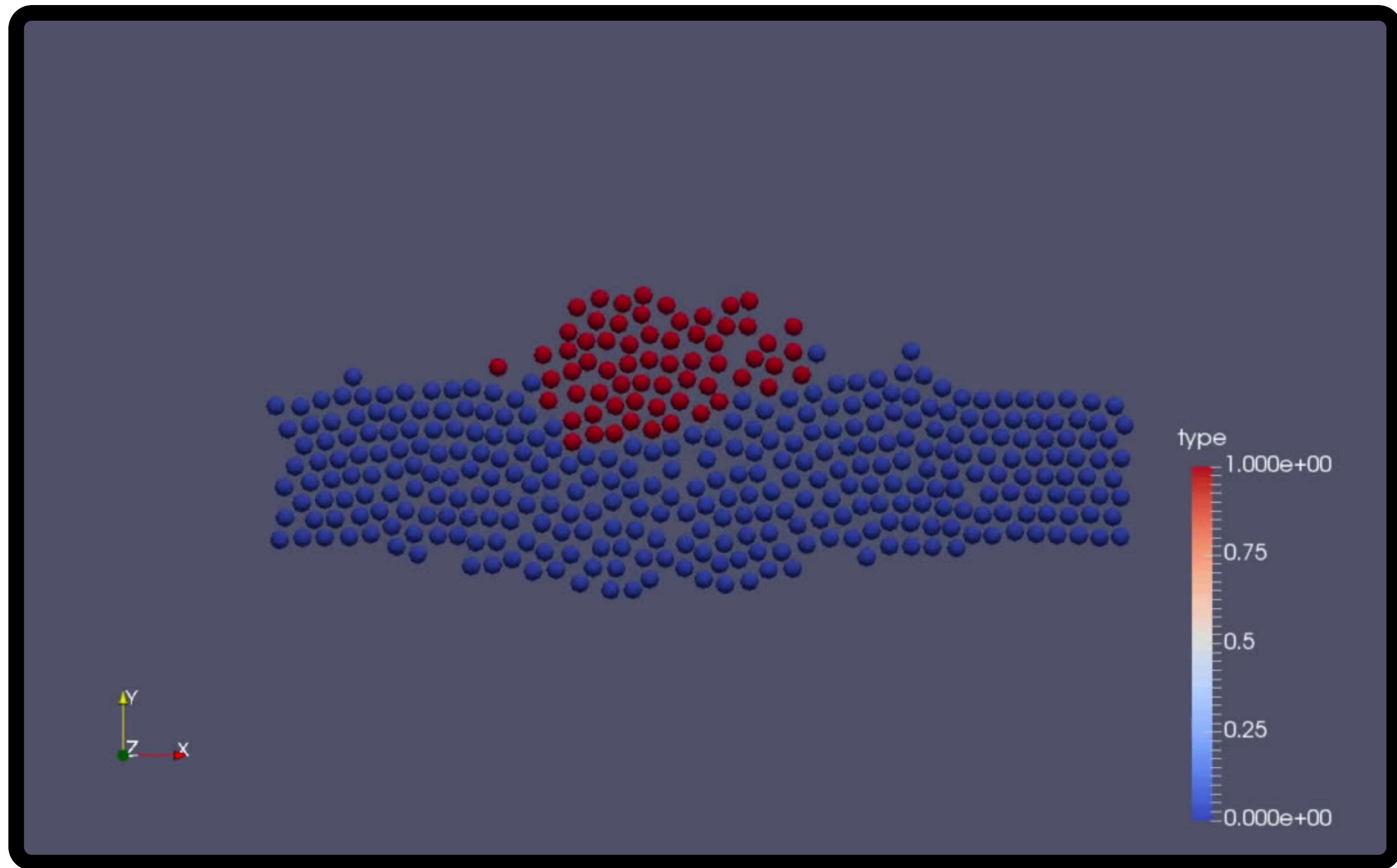
12



# SIMULATION 2

Molecule type as color code

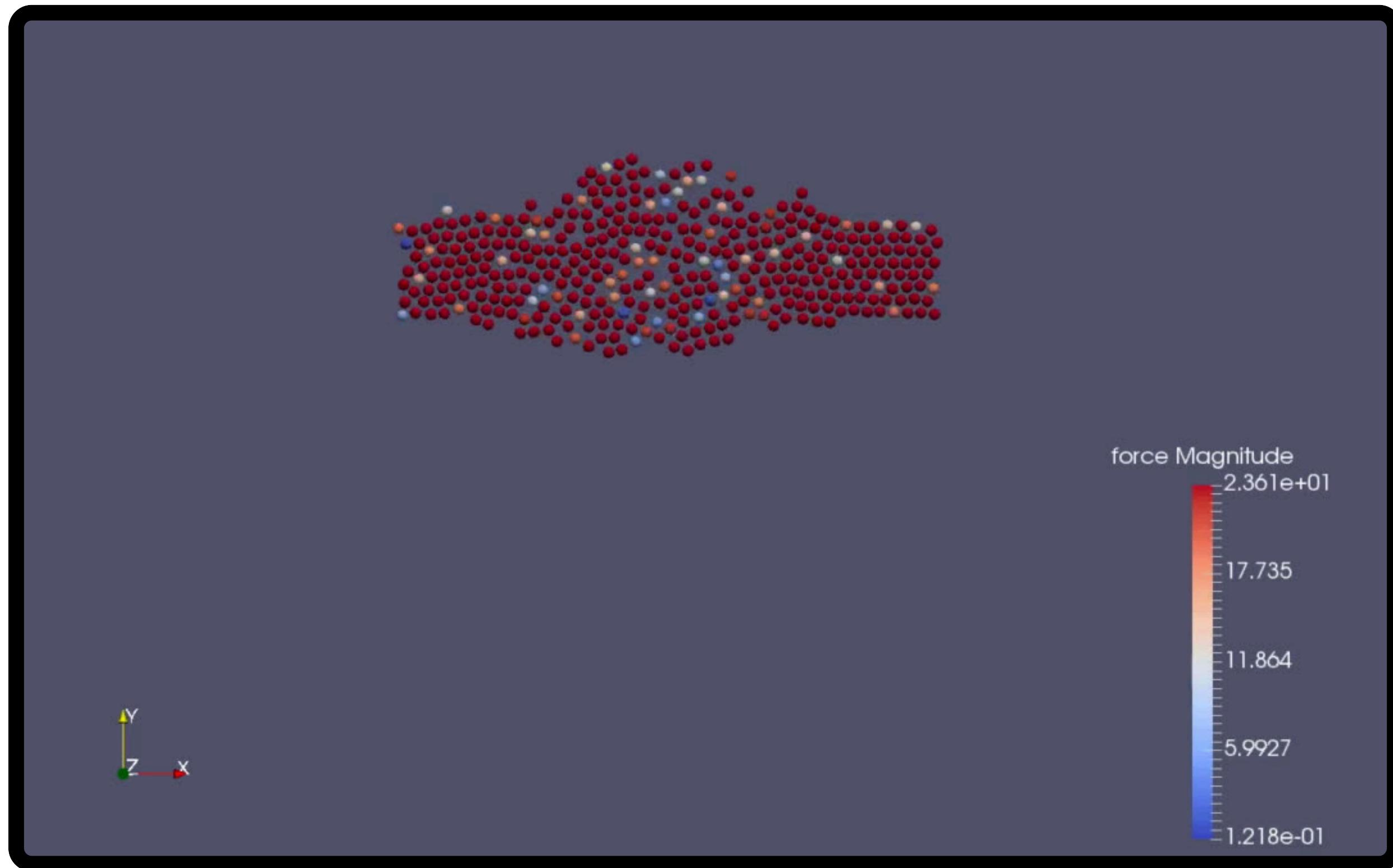
13



# SIMULATION 3

Force magnitude as color code

14



# 01:23:652 min

Hardware: AMD Ryzen 9 7950X3D 16C/32T, 64GB RAM  
WSL on Windows 11

# 01:24:964 min

Hardware: Intel i5-12600K 10K/16T, 32GB RAM  
WSL on Windows 11

# 01:33:020 min

Hardware: Intel i7-8750H 6C/12T, 32GiB  
Linux only system

## Function Call

```
./MolSim
-o LennardJones
-i "../input/eingabe-collision.txt"
-d 0.0002
-e 5
```

## SIMULATION TIMES

on our different setups

## **TASK 1**

testing for correct behavior with pre-calculated result checking  
testing is important and helpful and can save a lot of time (and also take  
a lot of time...)

## **TASK 2**

checking for correct formats, running tests, checking code quality with  
clang-tidy, sanitizers  
leads to clean repository and prevents bad/non-functioning code from  
being merged into the main branch

## **TASK 3**

cleaner faster and uniform possibility to print to terminal, way better  
than printing out single values and steps

## **REFACTORING**

meaningful naming, tidy structure, saves lots of time and makes working  
in a group more efficient

## **SUMMARY & LEARNINGS**

# FUTURE IMPROVEMENTS

## PERFORMANCE

parallelization, use of multi-core and threading

streamline calculation

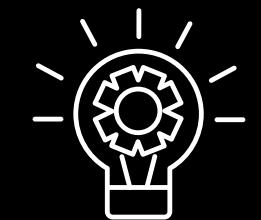
switch logging to macros for better runtime performance which did not work properly for us at this point



## TESTING

more edge case tests

writing tests while implementing new methods



## ADJUST PARSER

we thought about implementing an XML parser already but did not yet will be necessary for next week's worksheet





18

Scientific Computing (PSE) Molecular Dynamics  
Group D

Johannes H. | Julius K. | Tim S.

**THANK YOU FOR  
LISTENING**