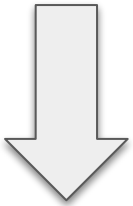
PSE Molekulardynamik

Team D - Sheet 1 : First steps towards a molecular dynamics simulation

Simplifying the calculations

- addition
- subtraction
- scalar multiplication
- euclidean norm

```
for(int i = 0; i < 3; i++){
    result[i] = vec1[i] + vec2[i];
}
```



```
result = vec1 + vec2;
```

Program Functionality

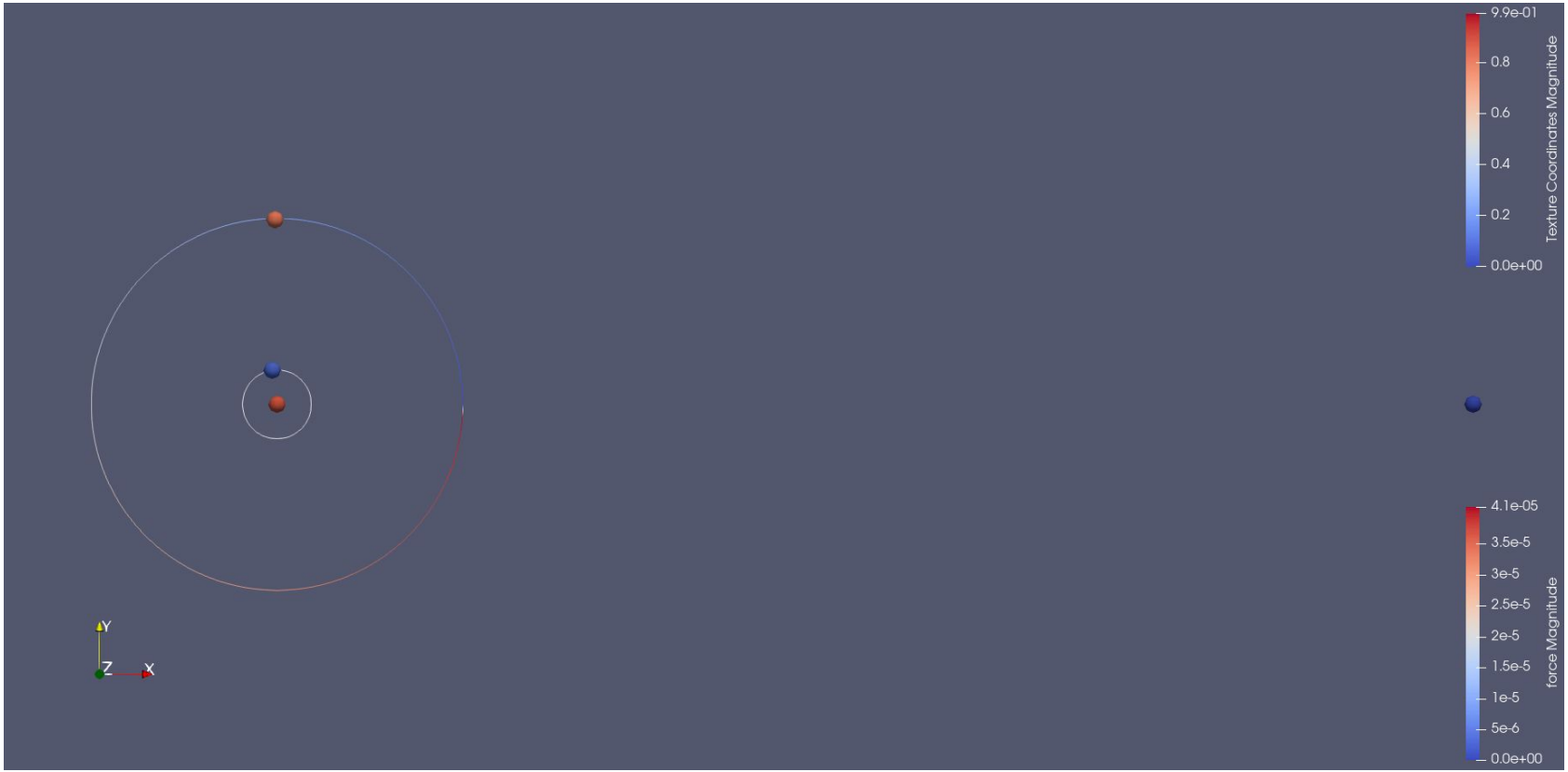
- created VTK output for visualization by utilizing the provided VTKWriter class
- Argument Passing with Boost

[1]

Why Boost?

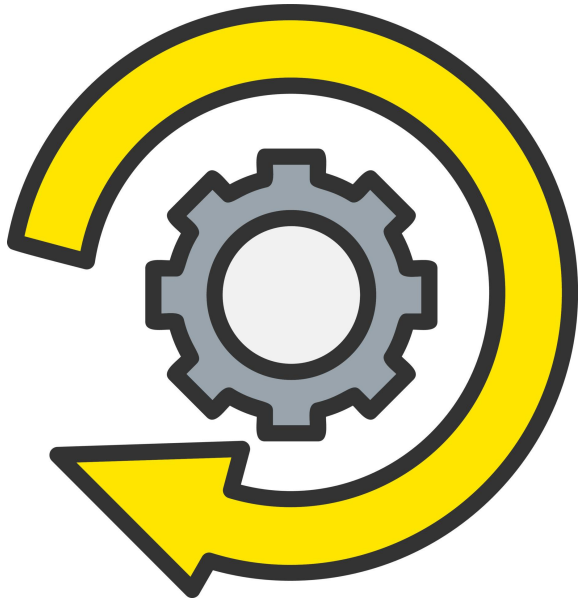
- User friendly command line interfaces
- Flexibility and customization
- Integration with standard C++ code

Simulation Unleashed: Exploring Cosmic

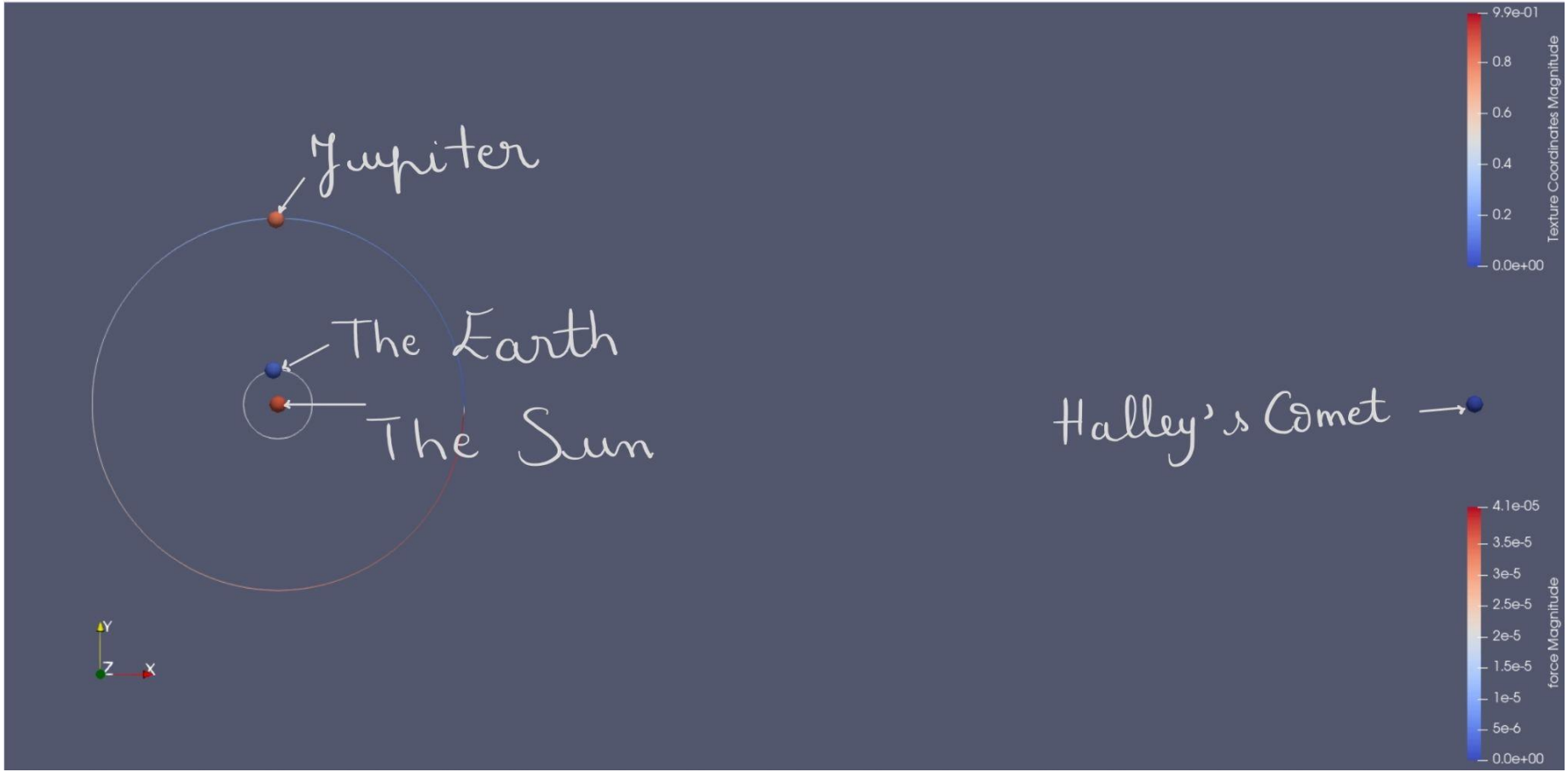


Analysis of the celestial bodies

- employed a systematic approach based on their coordinates, masses provided in the input file
- first body, stable →the Sun
- second body, possessing a mass which is s approximately 333.000 times smaller than the Sun's→ Earth
- third body's mass, approximately 1047 times smaller than the Sun's →Jupiter
- fourth body, located far away along the x-axis and characterized by an exceptionally small mass, coupled with its elliptic trajectory →Halley's Comet



[2]

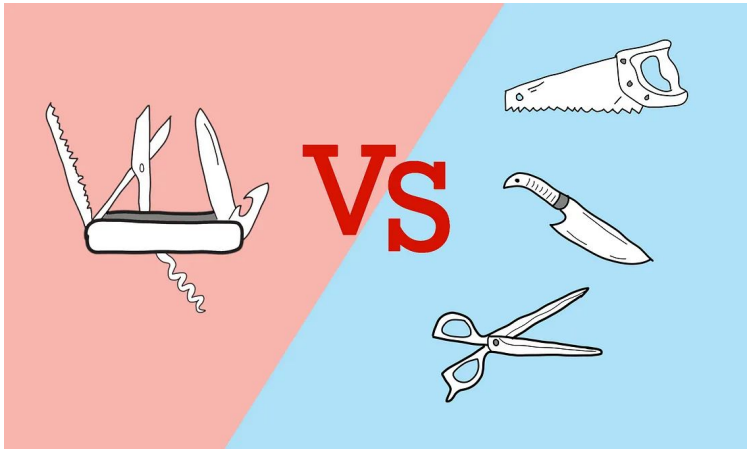

Jupiter
The Earth
The Sun
Halley's Comet
Y
Z
X
9.9e-01
0.8
0.6
0.4
0.2
0.0e+00
Texture Coordinates Magnitude
4.1e-05
3.5e-5
3e-5
2.5e-5
2e-5
1.5e-5
1e-5
5e-6
0.0e+00
force Magnitude

https://github.com/klaramozna/PSEMolDyn_GroupD/assets/101558922/00c0e27b-faaa-46da-9583-b0e2a427c47a

[3]

Refactoring the code: Simulation

Single Responsibility Principle:



[4] Multi Responsibility vs. SRP

A simulation class should only incorporate calculations that update the particle system

We therefore need that the particle system is composed onto a simulation class

Refactoring the code: ParticleContainer

- **Two iterators:** simple and two-level

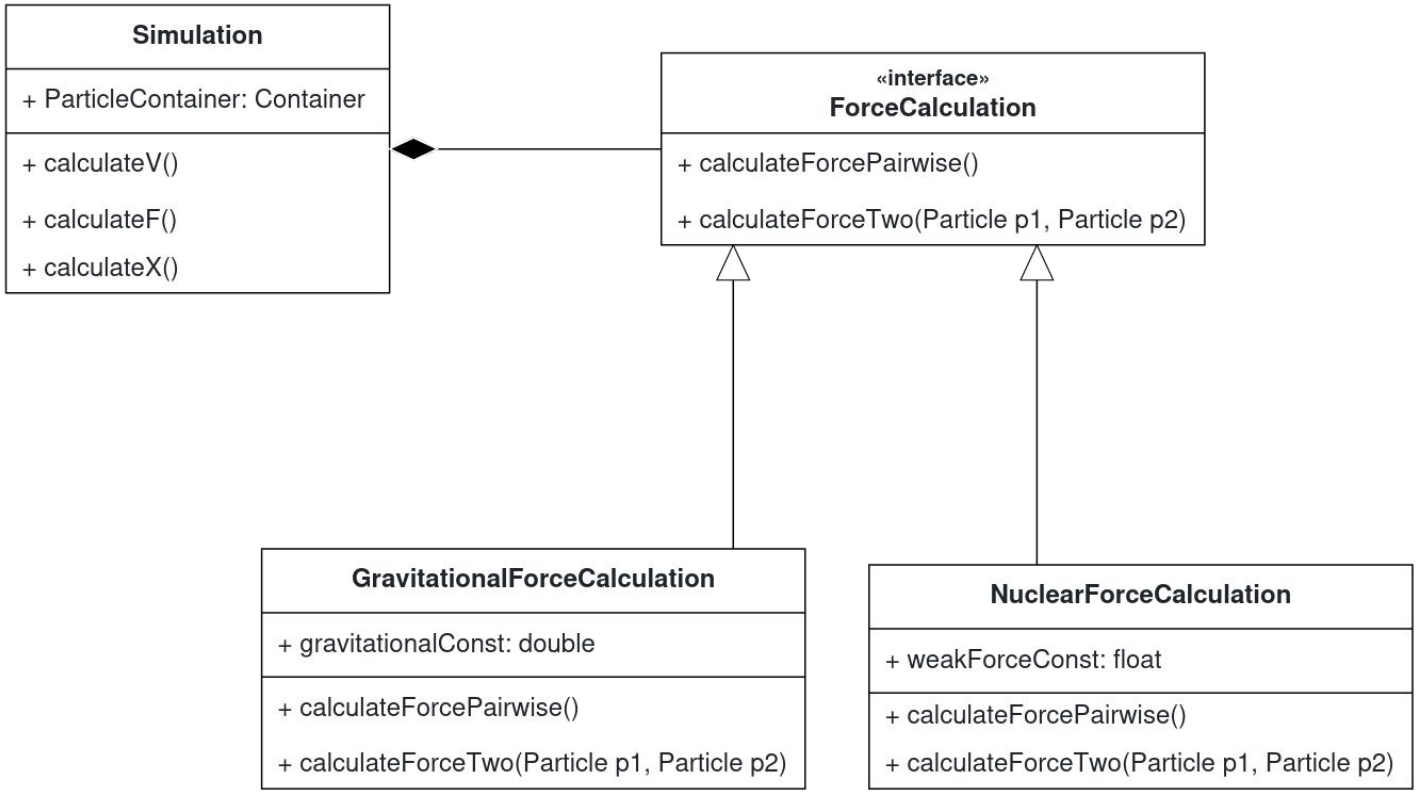
Simple iterator:

- enables use of range-based for-loops
- reuses functionality of std::vector

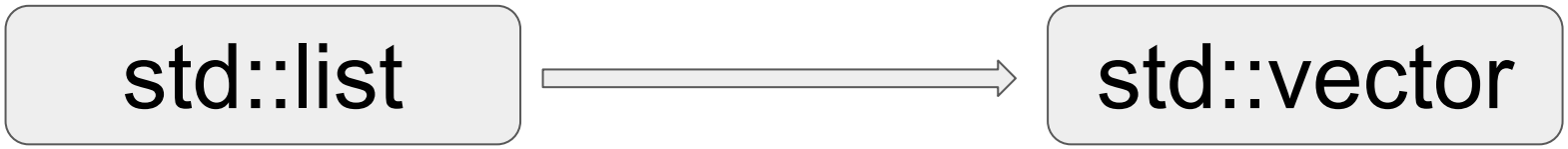
Two-level iterator:

- simulates a nested for loop

Refactoring the code: Expanding IO and Force Calculation



Refactoring the code: Data Structures



- **Advantage:** Better use of caches due to iterating from beginning to end
- **Why not std::array?**

-> no knowledge about number of particles at compile time

Doxygen docs online ?

[5]

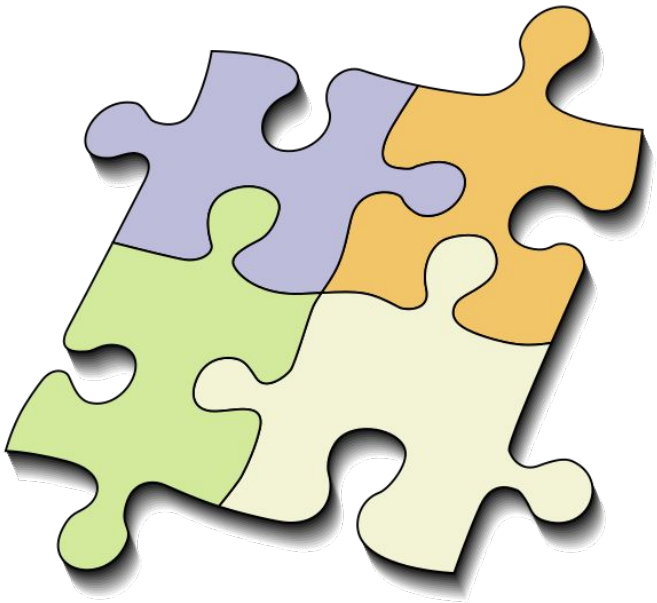
GitHub Actions

[6]

Link to our generated documentation:

https://klaramozna.github.io/PSEMolDyn_GroupD/

CMake



[7] Jigsaw Puzzle

Good design is modular without compromising functionality

With CMake, it’s possible to separate concerns and have each ‘CMakeLists.txt’ be responsible for a module.

We decided for 3 - one at the top-level, one for ‘src’ and one for ‘test’

Summary

Today we learned about…

- Simplifying vector calculations
- How to import and export the results of the simulation
- A toy example with Halley’s comet
- What choices come with refactoring

References

[1] https://en.m.wikipedia.org/wiki/File:Boost.png

[2] https://www.vecteezy.com/vector-art/16848747-reverse-engineering-vector-icon

[3] https://seeklogo.com/vector-logo/297768/movie-time-cinema

[4] https://levelup.gitconnected.com/the-single-responsibility-principle-made-simple-4e1597a44d7d

[5]https://raw.githubusercontent.com/github/explore/80688e429a7d4ef2fca1e82350fe8e3517d3494d/collections/github-pages-examples/github-pages-examples.png

[6]https://user-images.githubusercontent.com/95834130/197115277-e8bc27f0-fa68-47c4-a734-8b10350b078f.png

[7]https://pt.wikipedia.org/wiki/Quebra-cabe%C3%A7a#/media/Ficheiro:Jigsaw.svg