TUM School of Computation, Information and Technology WS 2023/24
Scientific **C**omputing in **C**omputer **S**cience
Prof. Dr. Hans-Joachim Bungartz
Samuel Newcome, M.Sc.
Markus Mühlhäußer, M.Sc.

PSE Molekulardynamik
Sheet 1: First steps towards a molecular dynamics simulation

Exercise on 20. October 2023

General information

- Work in groups of (2 or) 3 students
- The program frame is located at:
  `https://github.com/TUM-I5/MolSim`
- Deliverables:
  - Code in a git-repository.
    ⇒ Add the supervisor to your project as a "Reporter".
    ⇒ Should also contain all files needed to execute your code.
  - Moodle Hand-in containing at least:
    * Slides (pdf) for the presentation during the next meeting.
    * Pictures and animations of all runs.
    * A readme or report including answering any questions, analysing results, how to build and use your code.
    * See the example submission on Moodle, particularly the Readme file for further details.
- Deliver latest by the date given at the end of the sheets via moodle. Also, see instructions on moodle.
- If you cannot finish all tasks or still have bugs in your code, submit videos and pictures with explanations of the errors anyway. This helps during the grading. Otherwise, it might be assumed that you did not do these tasks at all!

Task 1 "First steps"

- Have a look at our Student Starter Clues especially the tools section. Think about what tools might be usefull for your project and how you want to employ them. `https://gitlab.lrz.de/tum-i05/public/studentstarterclues`
- Download and install CLion which is free for Students! (or any IDE you like) (`https://www.jetbrains.com/student/`).
- Setup GIT (`http://git-scm.com/`) and a repository at `https://github.com/`.
  ⇒ add a `.gitignore` to avoid commiting documentation or binaries!
- Download the program frame from GitHub, compile and execute it.
- Download and install VTK/Paraview (`http://paraview.org`).
- Download and install Doxygen (`http://www.doxygen.org`). Create a preliminary documentation of the code frame.
- Set the project name in CMakeLists.txt to `PSEMolDyn_GroupX` where X is your group number / letter.

Note:

- Instead of installing from the websites use your package manager (apt, pacman, yum, ...) if applicable.
- CMake allows weird ways to achieve any goal. Inform yourself about best practices and apply them throughout the course.
- Look at CMakeLists.txt:
  - The directory libxsd/xsd has to be on the include path.
  - You have to link against the Xerces XML Parser (flag -lxerces-c).
- If you insist on using Windows 10/11 you can run CLion and compile and execute the code through the *Windows Subsystem for Linux*. This probably results in higher performance than using a Virtual Machine.

Windows Subsystem for Linux (WSL) Guide

- Prepare the WSL:
  1. Install Windows Subsystem for Linux (Probably via Windows Store).
  2. Install build essentials (gcc, make, etc...) in WSL.
     → `sudo apt install build-essential`
  3. Install cmake (incl. terminal / curses gui) in WSL.
     → `sudo apt install cmake-curses-gui`
  4. Prepare rest via JetBrains script [1]
- Prepare Windows:
  1. Install CLion (e.g. through idea toolbox[2]).
  2. Set toolchain to WSL
     → Settings
     → Build, Execution, Deployment
     → Toolchains
     → Environment: WSL
     → Complete credentials: username, port (usually 2222), password
- Profit
  Now you can use a project in CLion with the source files on the windows file system and use CLion to compile it on the WSL!
- (all based on this[3] guide:)

Task 2 "Completion of the program frame"

- As we discussed in the meeting, the basic algorithm of the molecular dynamics simulation consists of the following steps:
  - Force calculation.
  - Calculation of the new position according to these forces.
  - Calculation of the new velocities according to these forces.
- Complete the steps of the simulation in the program frame.
- Create VTK output for visualization with the `VTKWriter` class.
- (Optional) Also enable binary VTK output. What is the advantage?
- Pass the parameters `t_end` and `delta_t` via the command line!

Task 3 "Simulation of Halley's Comet"

- Run the simulation from the input file `eingabe-sonne.txt`. As simulation parameters use at least the following:
  $\Delta t = 0.014 \quad t_{end} = 1000$
- Visualize the particles in Paraview (e.g. with a *glyph filter*).
- Which particle represents which celestial body?

Note:

- It is possible to export videos (.ogg or .avi) with "File" → "save Animation"
  If you have problems with the file size you can convert and hand in your videos in any format that is playable with the VLC player and has comparable or better quality than avi.

See next page

[1] `wget https://raw.githubusercontent.com/JetBrains/clion-wsl/master/ubuntu_setup_env.sh && bash ubuntu_setup_env.sh`
[2] `https://www.jetbrains.com/toolbox/`
[3] `https://blog.jetbrains.com/clion/2018/01/clion-and-linux-toolchain-on-windows-are-now-friends/`

Task 4 "Refactoring and documentation"

The code frame in its current state is not very suitable as a base for building up a molecular dynamics simulator. Refactor it, especially concerning the following issues:

- Encapsulate the molecules into a class `ParticleContainer`. It should be possible to iterate over the particles as well as the particle pairs in a simple way.
  → Which software design patterns are suitable?
- During this course, we will use different methods for I/O and for calculating the force between particles.
  → What might be an easy and extensible way to do that?
- Think about what data structures you use, especially for storing particles. Also, pay attention where you use copies or references!
  → This is highlighted by the output produced by the constructor and destructor of the Particle.
- Make yourself familiar with Doxygen.
  → What can you do with a system like Doxygen, what is it useful for?
  → Document the interfaces and implementations *you created* with Doxygen.
  → Set the main page of your Doxygen documentation to your projects' Readme file.
  → Do not commit the Doxygen output to your repository.
- Add a CMake module with a custom CMake target for invoking Doxygen. Required features are:
  → Create the documentation by calling `make doc_doxygen`.
  → Exclude the Doxygen target from the `all` / default target.
  → An CMake Option to disable creating the Doxygen target.
  (Useful if you build your code on a machine that does not have Doxygen).
- Pay attention to compiler warnings and IDE hints!
  → Consider using the latest Clang compiler: `https://apt.llvm.org/`.
  → State which compiler you used in your README.

Have fun!

Deliver by **3. November 2023, 3:00 AM** via moodle!