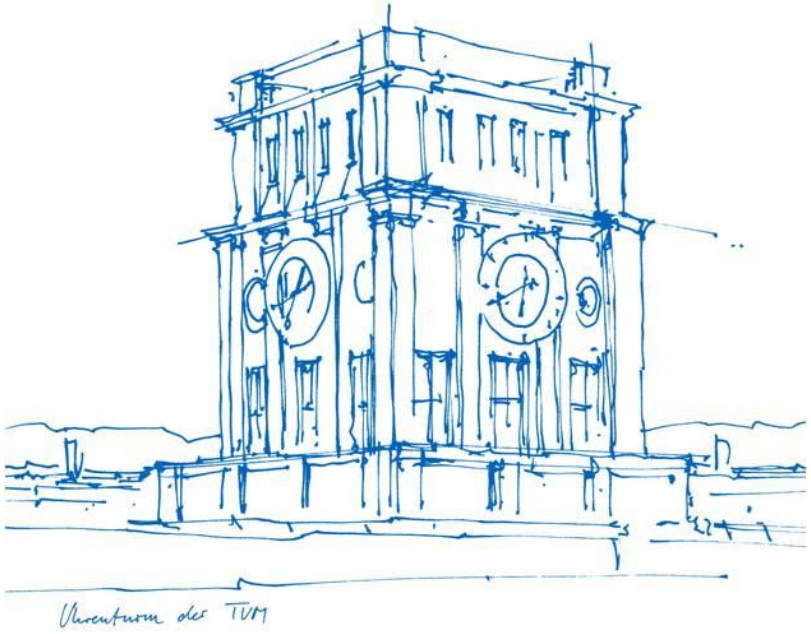
PSE Molecular Dynamics: Worksheet 2

Group C, 15.11.2024

Luca-Dumitru Drîndea

Mara Godeanu

Flavius Schmidt



Unit Tests

What to test?

- Expected Functionality (constructors, overloaded operators, physics calculations...)
- Edge Cases (e.g. integer limits, reading empty files...)
- Invalid Input (e.g. converting something that isn't a number to a number...)

Simulations:

Test different physics calculations against **eachother** and against **precomputed values**.

Mocking?

No thanks. (tedious for non-virtual functions, little performance gain, inter-class dependencies)

Unit Tests

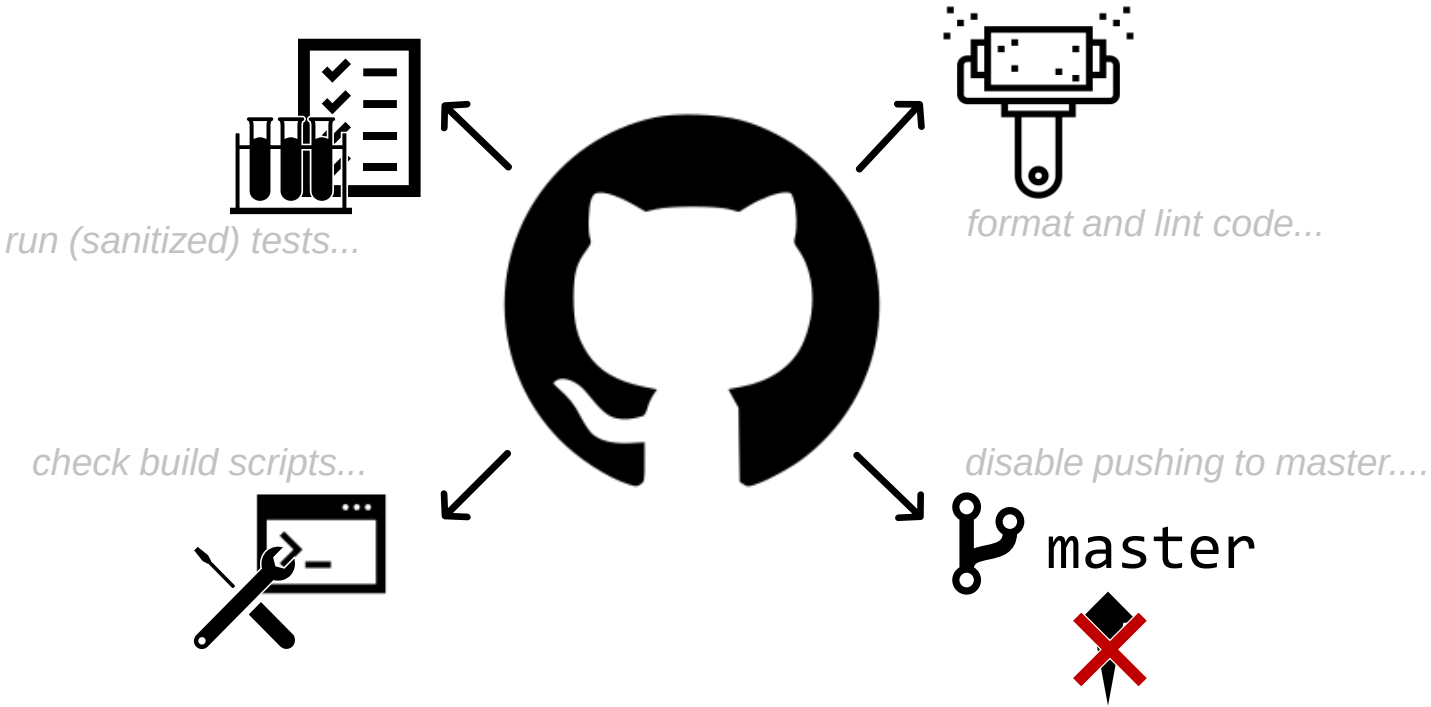
Problem: What about file input? How should the executable find the test input files?
~~relative file paths~~ what if the location of the executable changes?

Solution: Find them relative to the project's root directory.

Step 1: Find root directory... `…/MolSim/build/tests/tests`
Step 2: Find test directory... `…/MolSim/`
Step 3: Profit. `…/MolSim/tests/...`

Still not perfect, but better than before...

Continuous Integration



Logging

TRACE: *Very* granular, low-level calls. (C++ internals more than anything...)
→ used for monitoring operations with potentially high performance impact, disabled by default

```
[trace] Generated Particle (copy) ...
[trace] Destroyed ParticleContainer.
```

DEBUG: Slightly higher-level. (Program semantics...)
→ used to check program functionality and verify proper execution

```
[debug] Read line: # this was a triumph
[debug] Chose physics calculations for LJ simulation: ...
```

Logging

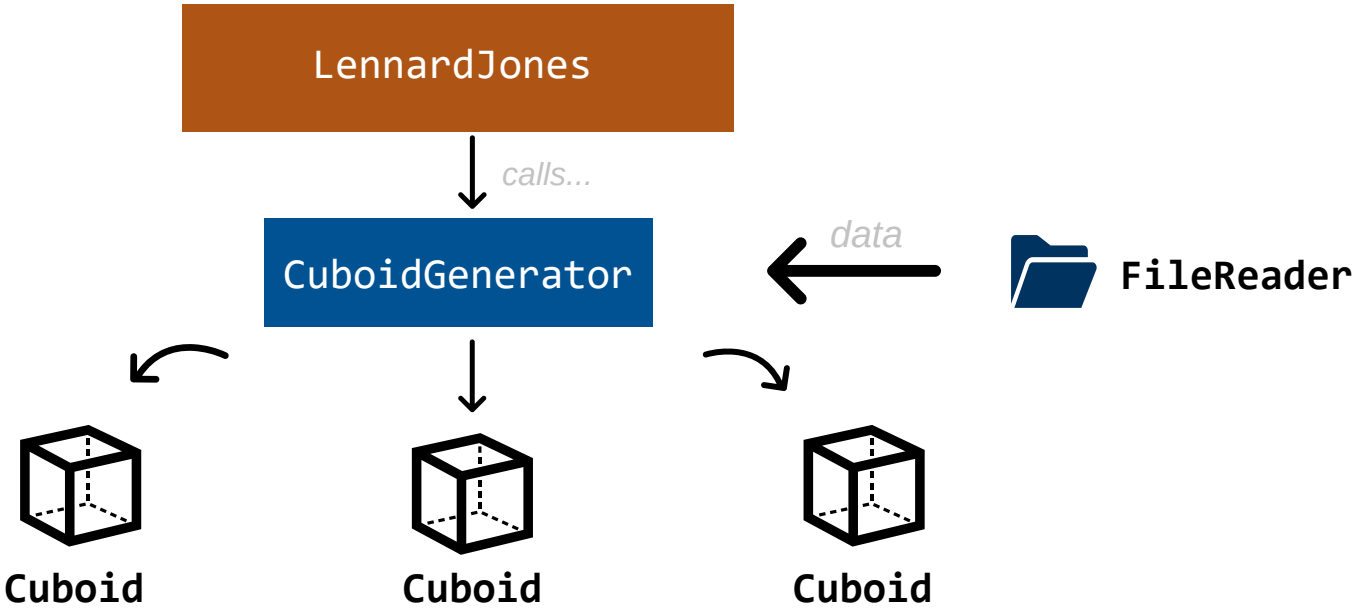
INFO: Log interesting functionality. (stuff that the user wants to see...)
→ used to show the user that the program is working and doing something

```
[info] Running Verlet simulation with parameters: ...
[info] Wrote contents to VTK file: ...
```

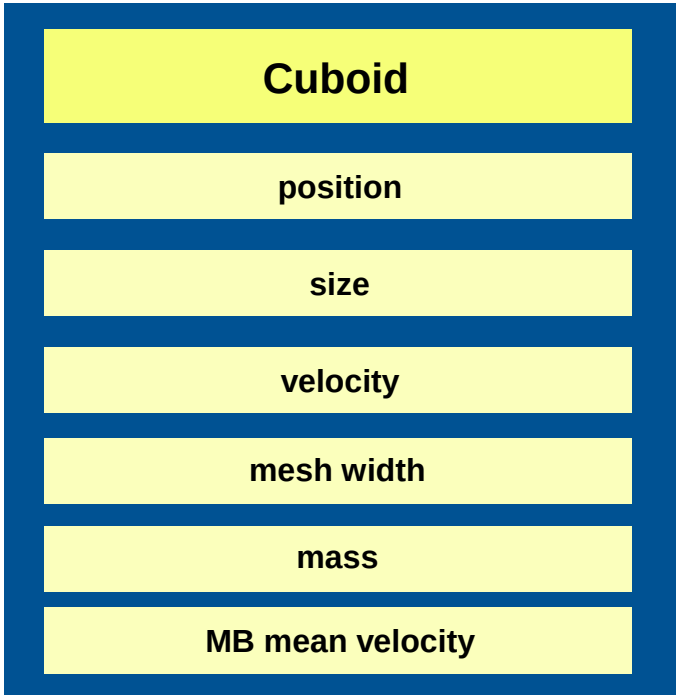
ERROR: Self-explanatory. (program usually terminates afterwards...)
→ used for errors typically caused by the user

```
[error] Number out of conversion range: ...
[error] Malformed input file! ...
```

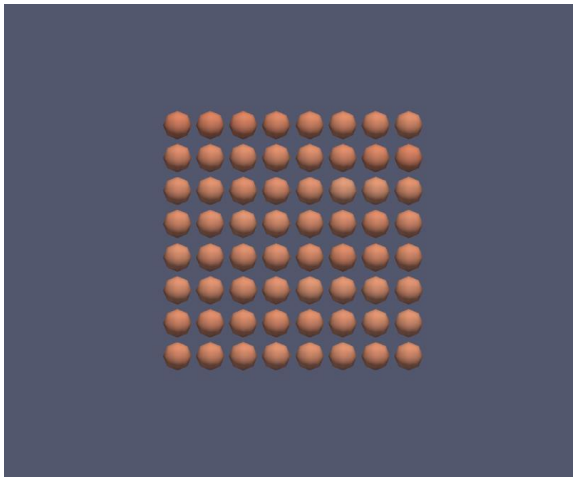
Cuboid and **CuboidGenerator**



Cuboid



initializeParticles()

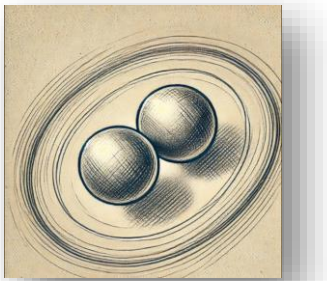


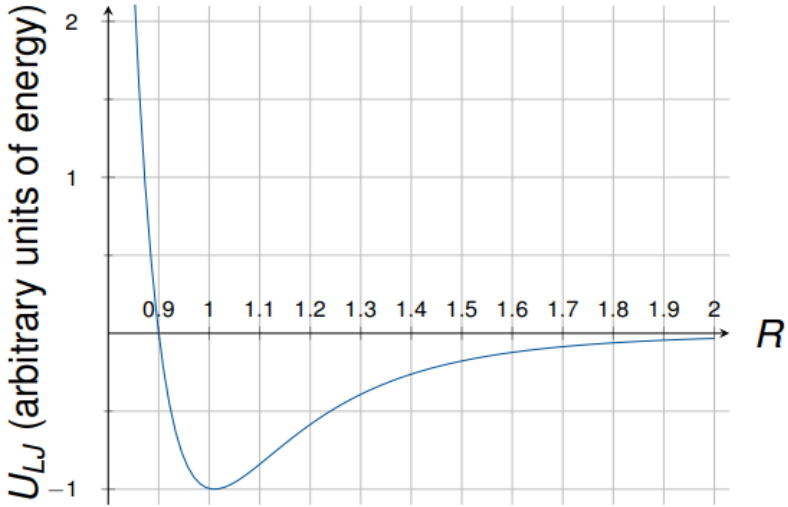
! All particles stored contiguously

Lennard-Jones Force

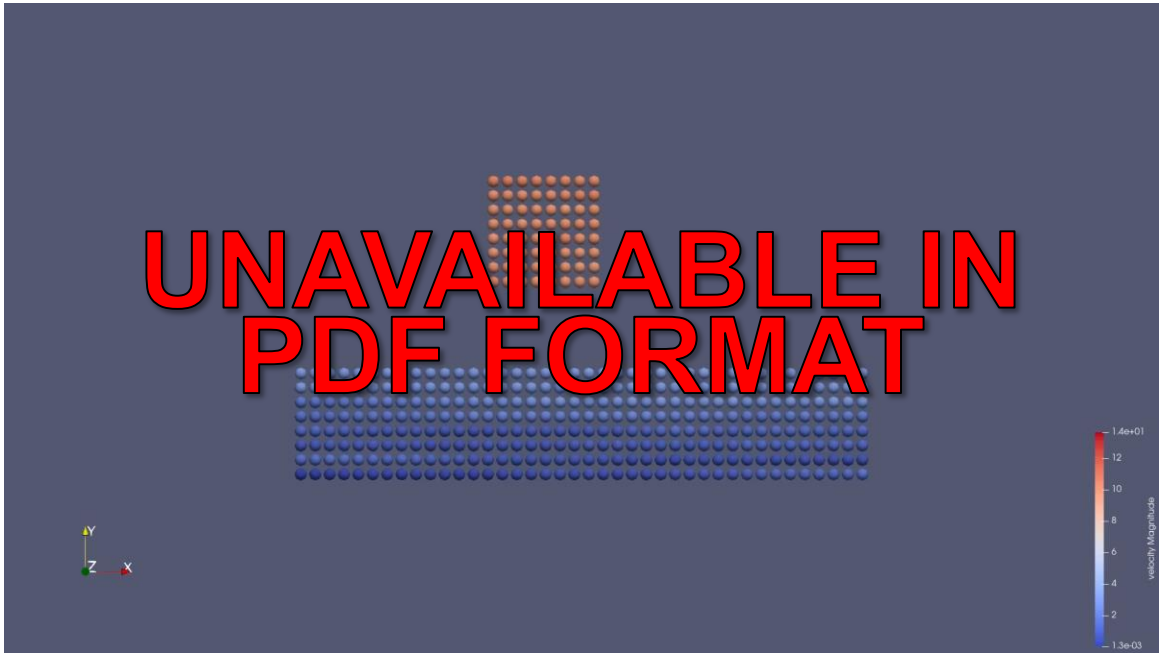
- Better suited than normal force at **molecular** level

- Normal force lacks **short range repulsion**
  → particles can **collapse** into each other

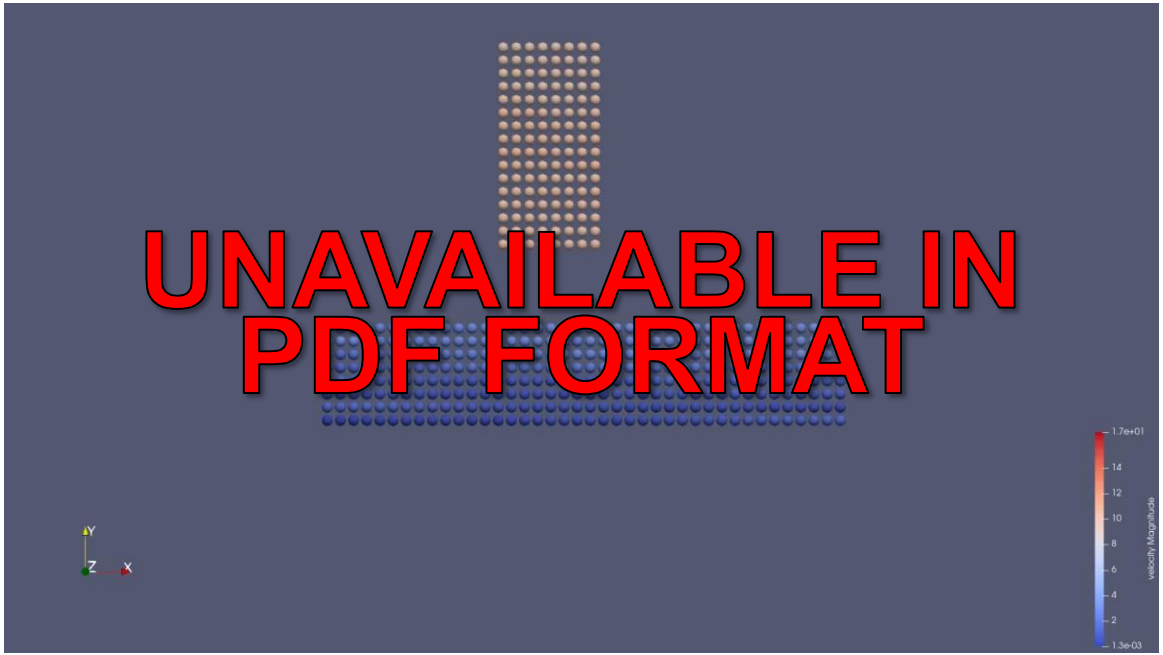




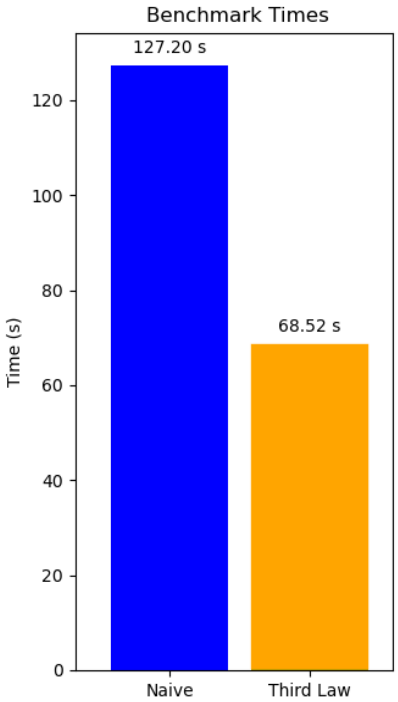
Small Simulation



Slightly Bigger Simulation



Benchmarking (using Google Benchmark)



Naïve Implementation: ~127 seconds!

Really slow.

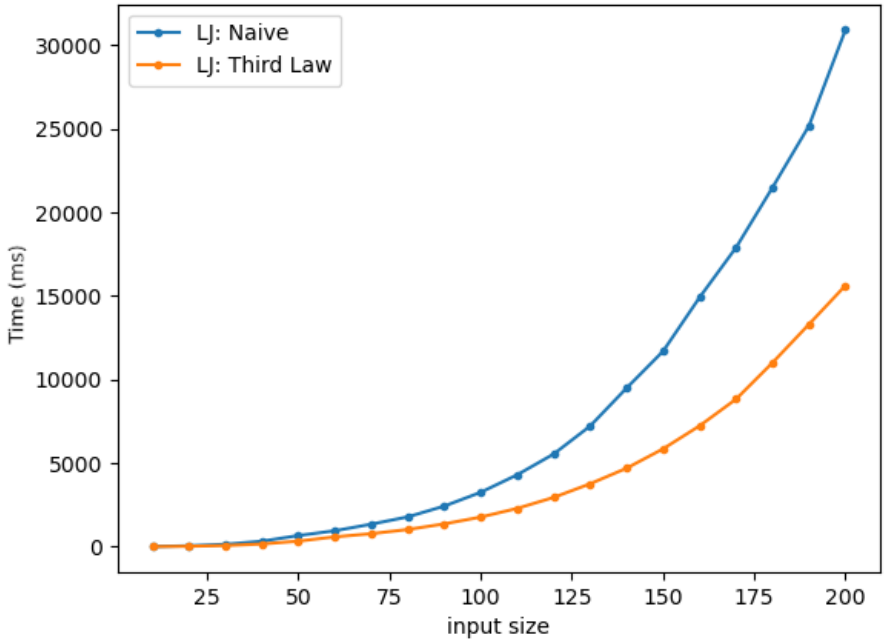
Optimized Implementation: ~69 seconds...

Expected, since we effectively halve the amount of iterations.

But still *really* slow.

Ubuntu 24.04 LTS, Linux 6.8, Clang++ 18.1.3, -O3
16GB RAM, AMD Ryzen 7 5800U (8 cores, 16 threads) @ 4.51 GHz

Benchmarking (using Google Benchmark)



Running the simulation for various amounts of particles...

Miscellaneous: Build Script

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

Before...

```
311  echo "[BUILD] Running tests..."
312  cd tests
313  ctest
314 fi
315 # post script completed
316 echo "[BUILD] Script executed!"
```

After...

Miscellaneous: Build Script

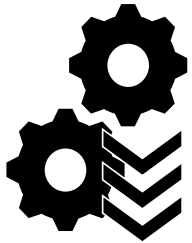
Shorthand options!

~~-DCMAKE_BUILD_TYPE~~

-b

~~-DENABLE_BENCHMARKING~~

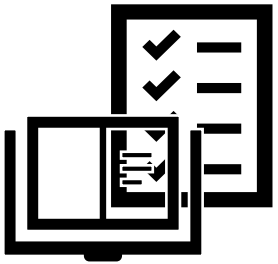
-c



Automatically installs **missing dependencies** system-wide.

\+ faster compilation

\+ smaller build directory



Runs **tests** and builds **documentation**.