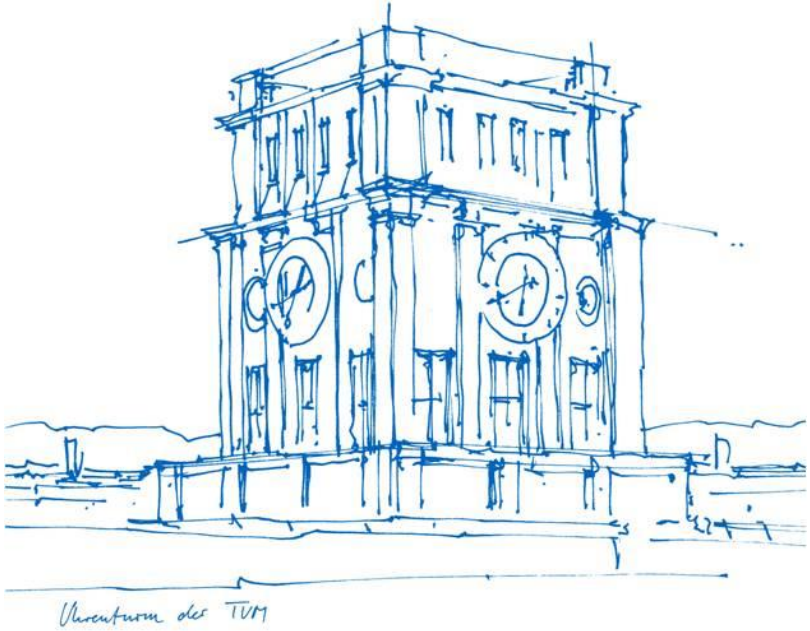
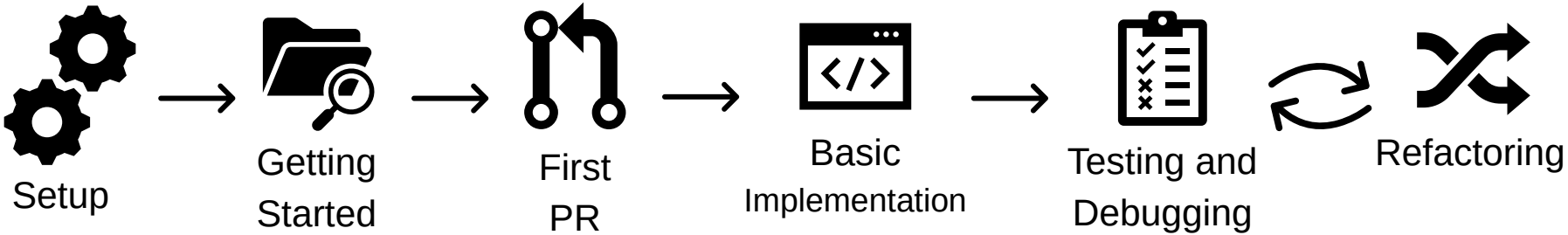
PSE Molecular Dynamics: Worksheet 1

Group C, 31.10.2024

Luca-Dumitru Drîndea
Mara Godeanu
Flavius Schmidt

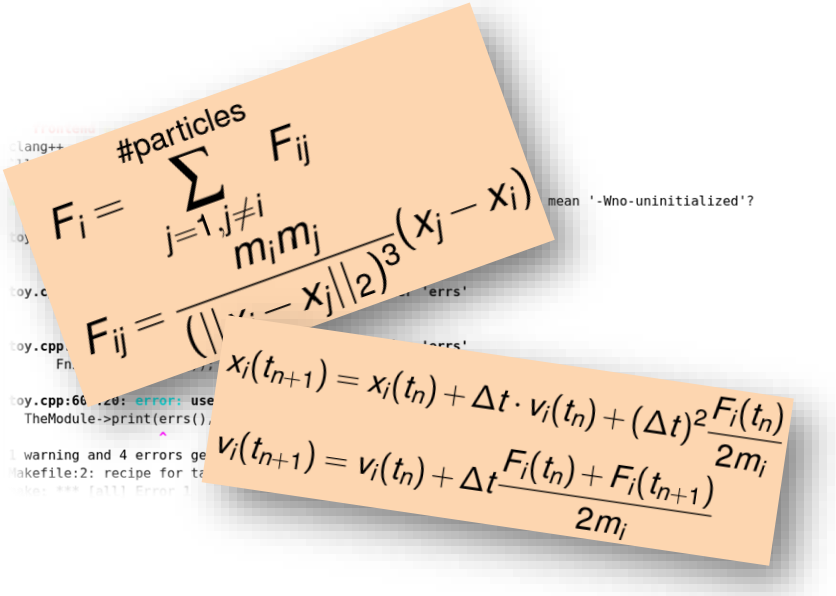


Workflow

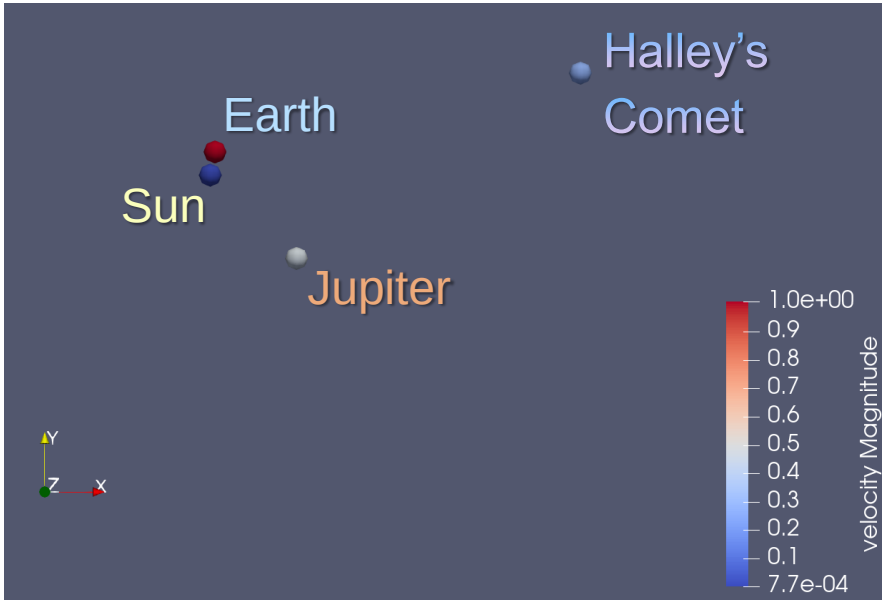

Setup
Getting Started
First PR
Basic Implementation
Testing and Debugging
Refactoring

Implementing the Formulas

- Implemented the **formulas**
- **Visualized** data
- **Debugging**
- Obtained a **working simulation**

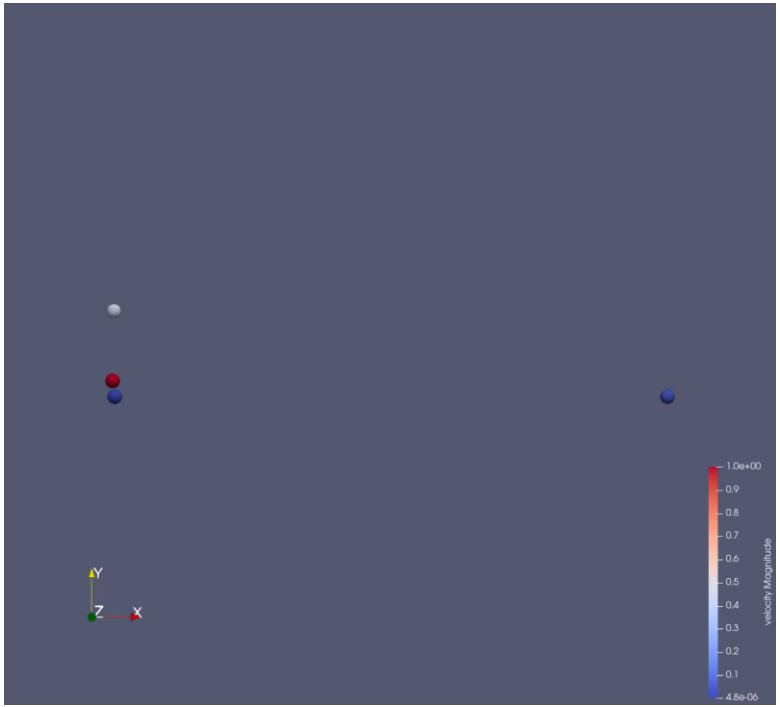


Simulation – First Data



- Based on **orbital period**
- Confirmed by **input data** (`eingabe-sonne.txt`)

Video



Command Line Interface

Which options should we be able to configure?

t_{start}?

Δt?

simulation type?

t_{end}?

output frequency?

output type?

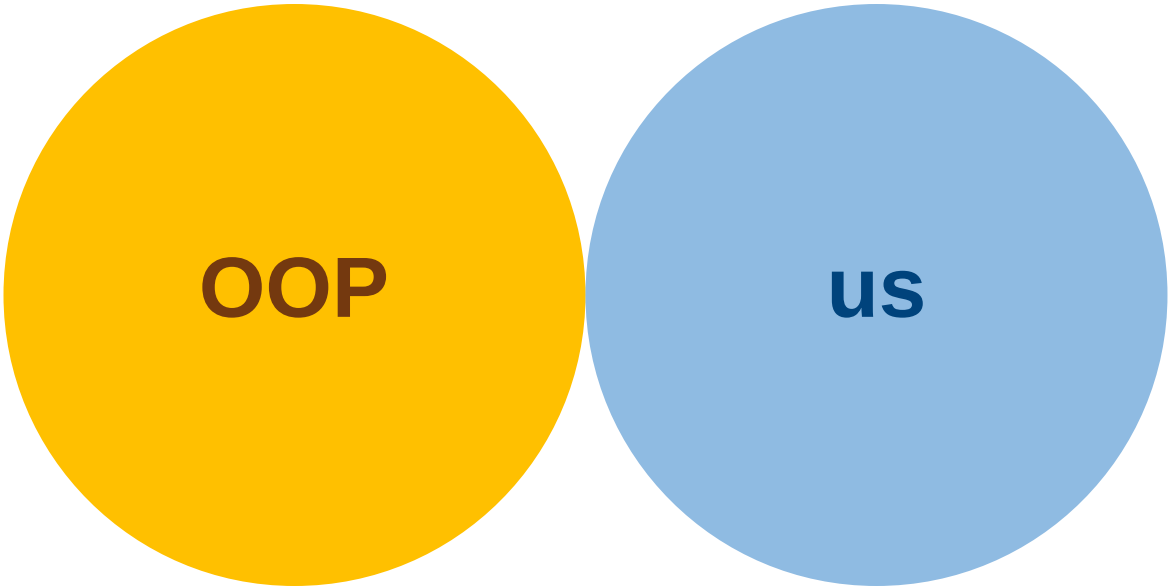
Answer: All of them!

Command Line Interface

How did we implement command line argument parsing?

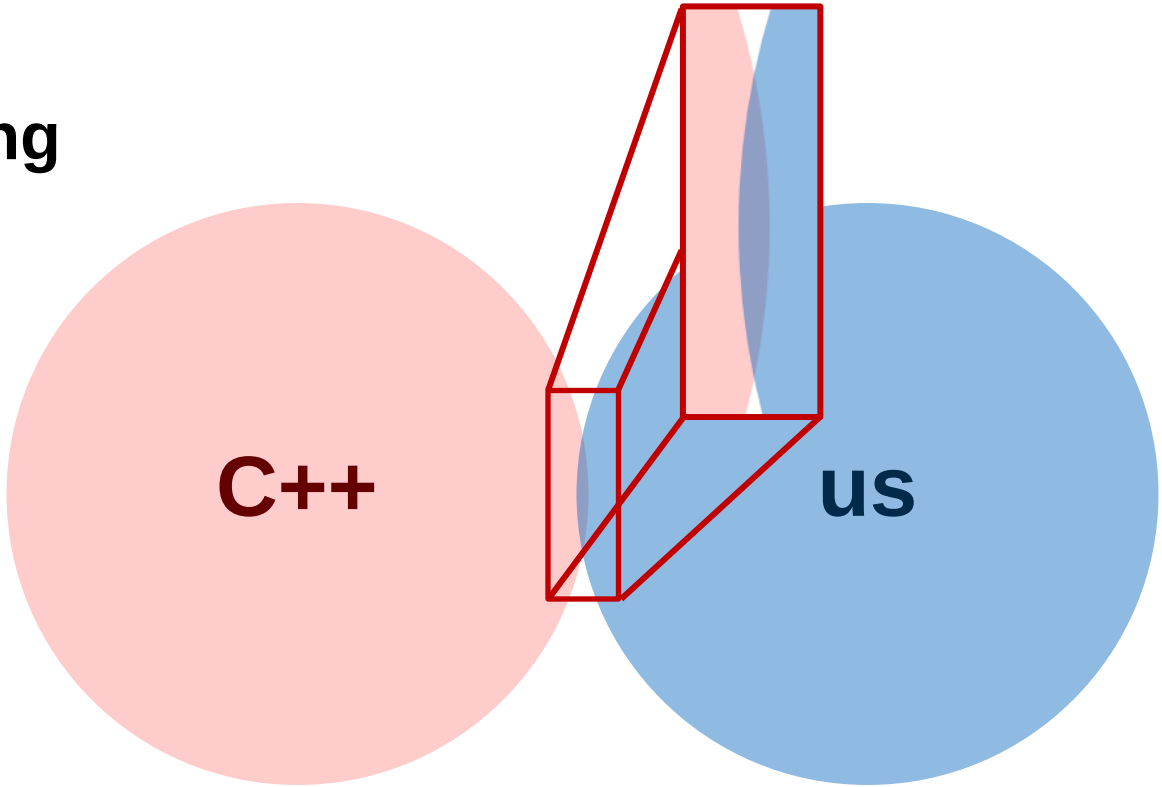
Method	Pros	Cons
Doing it ourselves	Highly **customizable**	**Tedious** to implement
Using some **external library** (e.g. Boost)	**Easiest** to use	**Overkill** (*whole* library for *one* feature…)
Using C's `getopt`	Universally **available**, already **familiar**	**Literally everything else**

Refactoring



Venn diagram showing our OOP skills.

Refactoring



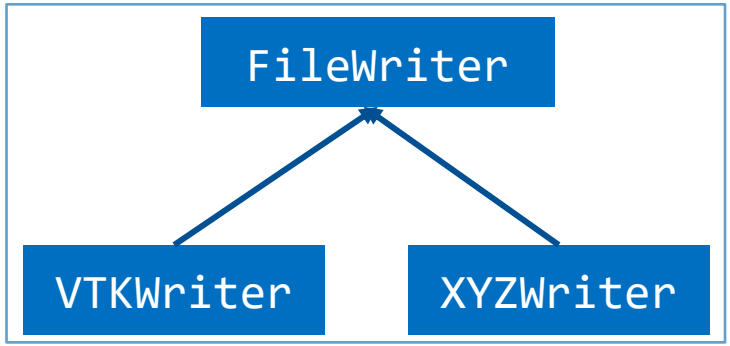
Venn diagram showing our C++ skills!

Folder Structure (`src`)

- **io** File input, output and command line parsing
- **objects** Physical objects used for simulations
- **simulations** The simulations themselves
- **utils** Helper / utility functions
- MolSim.cpp The main program

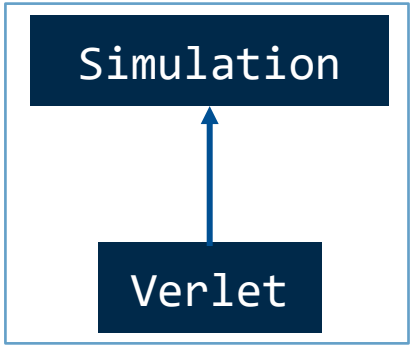
Class Hierarchy (notable examples)

has factory class

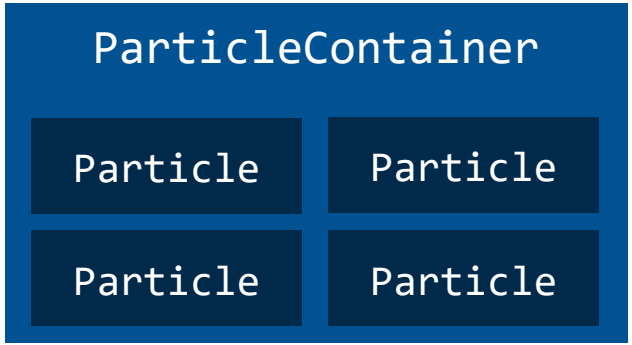


Basic Inheritance,
especially for I/O…

has factory class



Future-proofing…



Abstraction…

→ **easily expandable!**

Storing Multiple Particles

1. Standard library vector iterator *(normal, const)*

2. Custom PairIterator class
   - Two nested iterators
   - New functions *(beginPair(), endPair())*
   - Overriding operators *(++, == etc)*

```
iterator(), PairIterator()
```

Neat Doxygen Features

◆ calculateX()

void Verlet::calculateX ( )

Calculates the position x for all particles.

The position x_i of a given particle i at the next unit of time t_{n+1} is calculated using the formula

$$x_i(t_{n+1}) = x_i(t_n) + \Delta t \cdot v_i(t_n) + (\Delta t)^2 \frac{F_i(t_n)}{2m_i}.$$

Bug List

Member CLIParser::parseArguments (int argc, char **argv, Arguments &args)

The input file path doesn't have to be the last argument passed via the command line.

LaTeX rendering using **MathJax**

→ keeps the math all in one place...

Bug tracking using **@bug** …

→ easy to follow…

Other Minor Features and Observations…

- **Build script** to automate CMake build process…

```
rm -rf build
mkdir build
cd build
cmake ..
make
```

→ `./build.sh`

- Using **references** or **pointers** wherever possible (get rid of copies entirely)

```
Particle generated!
Particle generated by copy!
Particle generated by copy!
Particle generated!
Particle generated by copy!
…
```

→

```
Particle generated!
Particle generated!
```

Other Minor Features and Observations…

- **GitHub Workflow** to validate **code formatting** on pull request to `master`

```
            k;double sin()
         ,cos();main(){float A=
       0,B=0,i,j,z[1760];char b[
     1760];printf("\x1b[2J");for(;;
  ){memset(b,32,1760);memset(z,0,7040)
  ;for(j=0;6.28>j;j+=0.07)for(i=0;6.28
 >i;i+=0.02){float c=sin(i),d=cos(j),e=
 sin(A),f=sin(j),g=cos(A),h=d+2,D=1/(c*
 h*e+f*g+5),l=cos      (i),m=cos(B),n=s\
in(B),t=c*h*g-f*        e;int x=40+30*D*
(l*h*m-t*n),y=            12+15*D*(l*h*n
+t*m),o=x+80*y,          N=8*((f*e-c*d*g
 )*m-c*d*e-f*g-l        *d*n);if(22>y&&
 y>0&&x>0&&80>x&&D>z[o]){z[o]=D;;;b[o]=
 ".,-~:;=!*#$@"[N>0?N:0];}}/*#****!!-*/
  printf("\x1b[H");for(k=0;1761>k;k++)
   putchar(k%80?b[k]:10);A+=0.04;B+=
     0.02;}}/*****####*******!!=;:~
       ~::==!!!**********!!!==::-
         .,~~;;;========;;;:~-.
             ..,--------,*/
```

→ clang-format → FAIL!

Summary

1. Implemented the **basic functionality** → Realistic results and nice animations!

2. **Refactored** the code → Future-proof

3. Implemented **cool features** → Quality of life

forbes.com

Same thing?