Bachelor-Praktikum - Scientific Computing (PSE) Molecular Dynamics

Worksheet 1 – Planetary Simulation

Group F: Alp Kaan Aksu Berke Saylan Feryal Ezgi Aşkın

30.10.2023

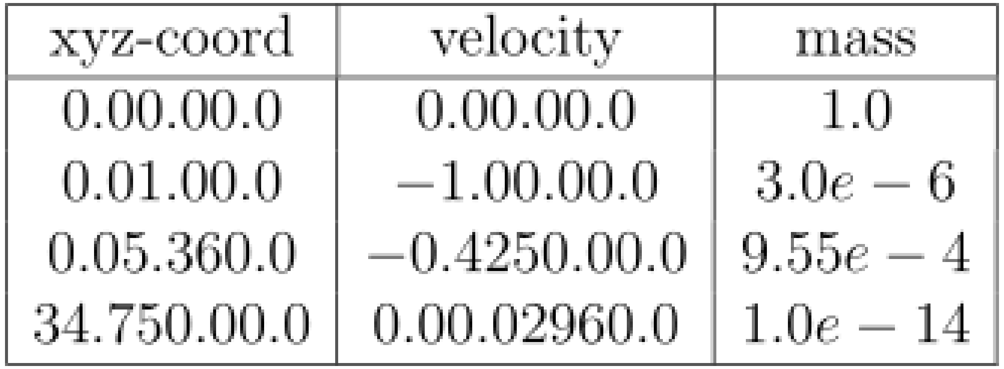
Celestial Body Identification

Challenge: Planets are unlabeled within the provided input file

Solution: Sun has the biggest mass and in the context of our solar system, it is often referred as the center (origin point)

Masses of the others scaled relative to the Sun's mass (normalized)

Position and velocity vectors are in astronomical units (relative to the distance between the Sun and Earth)



xyz-coord	velocity	mass
0.00.00.0	0.00.00.0	1.0
0.01.00.0	$-1.00.00.0$	$3.0e-6$
0.05.360.0	$-0.4250.00.0$	$9.55e-4$
34.750.00.0	0.00.02960.0	$1.0e-14$

Particle Container

Challenge: Create a class that encapsulates particles, providing methods for effective iteration

- Need for potential but not frequent dynamic resizing, efficient element access

➡ Use of vector data structure to store particles

- Implementation of insert(), delete(), get() etc. functions for future use

Particle Container

- Enable iteration over all particles

→ Iterator Pattern

- While iterating allow selecting the function to apply dynamically

→ Strategy Pattern

calculateF(), calculateX(), calculateV() iterate through all particles and apply different formulas

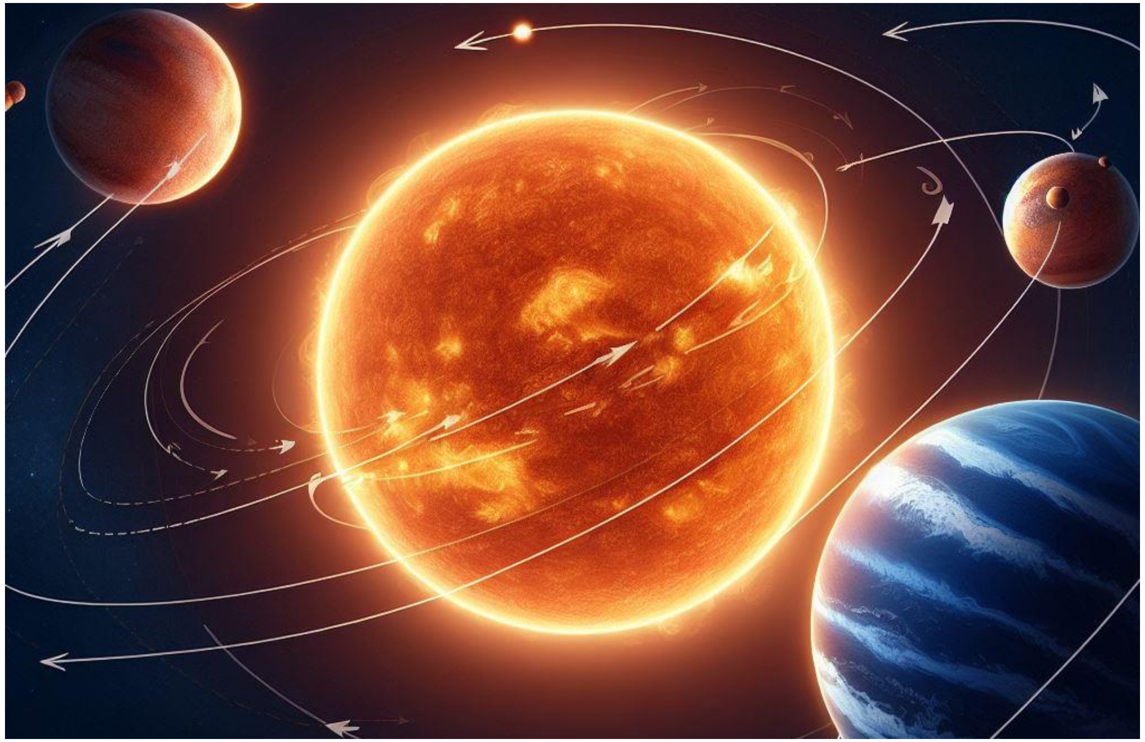
Let’s create a method that takes a function parameter and iterates through all particles and applies the parameter!

About Planet Rotations

- In Paraview it’s possible to observe anomalies with rotations, if settings are incorrect

→ Stride size is important to get the expected view

- WASP-17b is a planet outside of the solar system that rotates retrograde



Model

Challenge: Different (potentially more efficient/simple) formulas might be used for force, velocity etc. calculations in the future

- Need for dynamic selection for which formula to apply in each simulation iteration

➡ We created a new class named Model to only store different formulas

- force, velocity, position attributes as anonymous functions that can take different functions to apply different formulas

Problems encountered

- Updating the position instead of velocity in the velocity-setter

→ On Paraview planet movements deviated from expected, we thought we applied the formula wrong

- Unrealistic scaling of the planets on Paraview

→ Good camera angle needed for observation of planet movements

- Exporting the simulation from Paraview takes too long

→ Patience training ☺

Or maybe there is another solution?

Simulation

We modified the molsim.cpp and added a new Simulation class, WHY?

- Model
- End time, time delta
- Video duration, frame rate
- Input filepath, output filepath
- Output type

References

All images are generated by us using Microsoft Bing AI, so no copyright