Geo1000 – Python Programming for Geomatics

Assignment 4

Deadline: Friday, October 29, 2021, 18h00 CEST

Introduction

In this assignment you are asked to make the output functions of a n-body simulation. The outputs will be the 3D point coordinates of the bodies for each time step simulated. They can be visualized in QGIS (at least, the 2D projections). You will do so, by starting from both a Python script and a C++ program.

This assignment is *preferably* made in groups of 2 (enroll with your group or individually in Brightspace), and your mark will count for your own final grade of the course. Helping each other is fine. However, make sure that your implementation is your *own*.

Learning objectives

The goal of the assignment is that a student after this assignment:

- knows how to set up and use a C++ development environment
- gets more familiar with the simlarities/differences of/between Python and C++
- knows how to publish source code using the Git version management system

N-Body simulation

The following description is taken from: http://www.new-npac.org/projects/cdroms/cewes-1999-06-vol2/cps615course/nbody-materials/nbody-simulations.html

> "The goal of N-Body problems is to determine the motion over time of bodies (particles) due to forces from other bodies. Typical calculated forces include electrostatic force and gravity. The basic method used for solving the problem is to loop forever, stepping discretely through time and doing the following at each timestep:

- Update positions using velocities ($\overline{x}_{i+1} = \overline{x}_i + \Delta(t)\overline{v}_i$)
- Calculate forces $\overline{F}_i$
- Update velocities ($\overline{v}_{i+1} = \overline{v}_i + (1/m)\Delta(t)\overline{F}_i$)

[...] Note that it is possible to use multiple resolutions for time, i.e. different $\Delta(t)$ for different particles, but we will only consider uniform $\Delta(t)$."

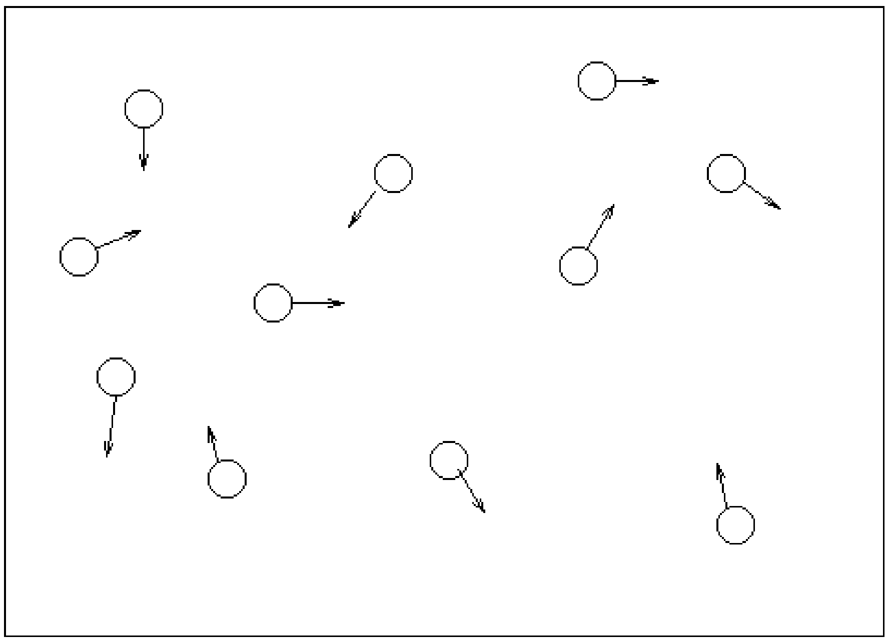

Figure 1: Particles in Motion

Preparation

Before starting on this assignment, make sure to read the online book 'C++ for Python Programmers' (go over at least Chapters 1–7) at https://runestone.academy/runestone/books/published/cpp4python/index.html Also, make sure to answer all interactive questions in these Chapters (to test yourself if you've mastered the material).

C++ development environment

Make sure you have an environment for programming with Python and C++:

- Install a C++ Compiler for your platform
- Install Clion

These steps are documented in the manual for installing CLion prepared by AdHok. You find the correct manual at: https://adhok.bk.tudelft.nl/manuals Please, let us know (e.g. in your report) if certain steps are missing or unclear (this way we can improve the manuals for next year). In case you get stuck, also the developer of CLion (JetBrains) has extensive documentation online: https://www.jetbrains.com/help/clion/installation-guide.html

Changes you have to make to the Python / C++ code

Clone the Git repository from Github: https://github.com/bmmeijers/nbody

This gives you a CLion project, that you should be able to open in CLion. Note, CLion also has a Git client, so you should be able to use it, to clone the project as well.

Now, first get familiar with CLion and the code given to you. See how to execute the C++ code in CLion (build and run) and also set up the Python interpreter you want to use. Try to understand what happens in the n-body simulation (although it is not necessary for the assignment to grasp all the math behind the simulation).

Adapt both the Python *and* the C++ code to output a CSV file, that stores (at least) the 3D position of the bodies in the simulation for each time step at which the simulation is calculated. Per time step per body (Sun, Earth, . . .) a line should be present in the CSV file with on every line:

> name of the body, position x, position y, position z

Make sure there also is a header line in the output files you write with descriptive column names. The delimiter you should use is a semi column (;). It should be possible to load the CSV files produced by both your Python and C++ program in QGIS.

If you want, you can make the output more 'fancy', e.g. you can output more attributes as well (e.g. components of the velocity vector, or the mass of each body), or you can add a boolean program argument that can enable or disable writing the output file.

Use Git

Record the history of the modifications you make to the source code in a Git repository of your own, and make sure that all the members in your group have made at least one commit (and thus finally appear in the Git log).

Benchmark

Give an indication of the runtime of both of your programs. The number of simulation steps should be varied in 4 sizes: n = 5'000, 500'000 and 50'000'000 iterations. For the C++ program: Compile your C++ code both in Debug, as well as in Release mode and include both the timings in your report.

Note: You could write a small Python script that invokes the other programs. More details on how this can be done can be found in the Python book in Section 14.8, or using os.system, together with: time.perf_counter.

What to hand in?

Put the modified code in an online and public Git repository of your own (e.g. on Github).

Next to the code, hand in a report (as PDF file) on Brightspace that contains:

- Your name(s) and student number(s)
- A link to the Git repository that hosts your modified code
- The number of words your report uses.
- A paragraph (max. 200 words) explaining how the Python and C++ programs are different (How is the information in the simulation represented: Which data types do the programs use to represent the data for the simulation?)
- A paragraph reflection on how you went about solving the task (max. 400 words). Which steps did you take? How did you measure the runtime? How did collaboration go with Git? Did you get stuck? Which sources did you consult when you got stuck? Did you expect the results you obtained? Etc.
- Timings for Python, C++ Debug and C++ Release, in a table and visualised in a chart (x-axis: instance size/y-axis: run time).
- One (or more, e.g. 1 overview and 1 close-up) screenshot of QGIS, where you have loaded the CSV files your code did produce for 5'000 iterations (for both C++ as well as Python programs)

Grading

You can get 100 points for this assignment. The aspects that we will look at are: use of Git (10 points), your (modified) code (40 points) and your report (50 points).

Note that if you submit your assignment after the deadline, some points will be removed. For the first day that a submission is late, 10 points will be removed before marking. For every day after that, another 20 points will be removed.

Appendix

C++ reference

The go to website for C++: https://en.cppreference.com/w/

CMake

CMake helps defining the steps for building software (i.e. the steps the compiler needs to take, while compiling your C++ source code). For this, a text file named CMakeLists.txt has to exist within your project. You can read more about it at: https://www.jetbrains.com/help/clion/quick-cmake-tutorial.html