BSP Project Description: OrbitCalc - A simplified version of the solar system

Monday 16th October, 2023 - 10:47

Henrik Klasen
University of Luxembourg
Email: henrik.klasen.001@student.uni.lu

This report has been produced under the supervision of:
Gabriel Garcia
University of Luxembourg
Email: gabriel.garcia@uni.lu

***Abstract*—With scientific endeavours becoming more cost intensive, there is a need to plan specific missions into minor details. For having the means to plan such missions, for example to the edge of the solar system, to the depth of the oceans and onto the surface of other celestial bodies, there has to be a computational model, which virtually has any property, any side factor and any physical property of the environment the mission planners expect to encounter. The development of one such computational model with a potential use case in space exploration is the topic of this Bachelor Semester Project. (100 words)**
In total 1224 words, ca. 60%

1. Main required competencies (107/100-200 words 5.35%)

1.1. Scientific main required competencies (44 words)

Before starting the Bachelor Semester Project, the student had some experience in orbital mechanics, which includes the basics of (gravitational) force vectors.
Next to that, the student had some experience in constructing mathematical and physical models with certain simplifications for computational ease.

1.2. Technical main required competencies (63 words)

As the technical deliverable is about the implementation of a basic solar system simulation in Unity, the student already knows how to work with Unitys' Scaling, Transformation and Rotation system and their Script APIs in C#. Also the basics of game and simulation development are known, as it was part of a small course in the Computer Science Summer Camp at Hochschule Trier.

2. A computational model for a solar system simulation (664/800 words)

The scientific part of this Bachelor Semester Project is about modelling a suitable, simplified model of the solar system.
It will consist of three parts. The first part being about the fundamentals of computational modelling, which will take a look at the approach of breaking real world data down to a computable model, which can be emulated by the computer. As the title of this Bachelor Semester Project implies, that it is about the simulation of orbital mechanics in a virtual solar system, the second part of the scientific part is about modelling a model with some constraints and simplifications for implementing it later in the technical part.
The domains of this project are:

- *Computational Modelling*: This domain is all about breaking down real life environments and behaviours into equations and algorithms for simulating the environment as good as possible with certain simplifications.
- *Simulations*: Simulations are a computational approach to emulate real life behaviours of objects and environments on computers. They are used for example for optimizing the trajectories in space, simulating fluid dynamics or aerodynamics for optimizing the fuel efficiency of cars.
- *Scientific Computing*: Scientific Computing is an interdisciplinary discipline bound to develop approaches for algorithms, models and software for simulating real world behaviour as close as possible. This means, that computational modelling and simulations are both a subset of scientific computing, which is the broader scientific topic.

The scientific part will tackle the following scientific question: *How can one model a simplified simulation of the solar system?*

To find a satisfying answer to this question, it is necessary to answer the related questions:

- *What is a computational model?*: This question aims on answering the most basic and fundamental question of this project, as it is required to later on build and model the computational model.
- *What environmental factors of the solar system have to be considered?*: This question aims on answering, *how* the model will look like, what simplifications there are and which approaches are taken. It will also reason, why the approaches chosen have been chosen.
- *How can the model be modelled?* : With this question, the goal is to find equations and algorithms that enable us to model the computational model properly. This part is meant to include all the orbital mechanics theory needed.

As the goal is to find satisfying answers to all questions, it is necessary to use a few resources on these topics. This will include the following papers and books:

- [KaufmannFranzinMenegaisPozzer]: This publication is about the computational cost of large scale physics simulations. It will mainly be used in the section for the question "*What environmental factors of the solar system have to be considered?*", as it is about modelling the computational model itself.
- [MurinKompisKutis]: The main point in using this publication is to find out a way to properly model a computational model. It provides many examples for computational models for several different simulations, which might help when modelling the solar system later in the project
- [HeisterRebholz]: This resource is about scientific computing. As its target is to teach undergraduate students the fundamentals of scientific computing and numerical methods to model a simulation or whichever model needed, it is well suitable for this Bachelor Semester Project. It will be used for getting some insights on how scientific computing is used today and how it *can* be used in this project.

With the use of these resources, the scientific part will focus on delivering the most useful and crucial information to develop a satisfying model of the solar system. The results of this part will later on be used in the technical part of this bachelor semester project, which is all about the implementation of a three dimensional solar system in Unity. In the last section of this part of the project, we will take a look at which factors cannot be simulated in the model developed and which parts could possibly be added in future extensions, but which exceed the scope of this project.

3. Implementation of a virtual solar system (353/800 words)

The technical part of this Bachelor Semester Project is about the implementation of the computational model developed in the scientific part. In fact, it is about the implementation of a virtual version of the solar system.
While the scientific part is more about the "*What do we want to simulate?*", this part is more about the "*How do we implement what we want to simulate?*". The goal with this deliverable is, to get to know how to implement scientific computing models in a given environment.
The domains of this deliverable are:

- *Unity*: A state of the art game engine for three dimensional and two dimensional games. With its powerful, built-in physics engine, it is able to also create realistic simulations, such as digital twinning, fluid simulations, as well as large field gravitational simulations, which is the goal in this project.
- *Visual Studio*: A powerful code editor for all of the programming languages in the .NET Framework. This IDE has Unity integration, which simplifies the development of programs in Unity by a lot.
- *C#*: A language in the .NET framework, which was heavily influenced by C++ and Java. It is object-oriented and Unity provides an API to its editor to this language.

As for the technical deliverable, there is not as much literature such as official scientific papers or publications, the main source of information will be the scientific deliverable, e.g. for the equations describing the gravitational law. Next to that, the official Unity Documentation (linked in [Unity]) will be used.
The technical deliverable will provide some key features:

- *Gravity*: According to the Newtonian gravitational law. Further details will be in the scientific part of the report.
- *Real scale*: The scale of the objects in the solar system will correspond to their actual scale in the real solar system. This will (hopefully) ensure a higher degree of realism and ease gravitational calculations, which are based on the distances.
- *Realistic lighting*: As the sun is more or less the only high intensity light source in the solar system, the simulation will use a point light source which will shine light from the suns position.

4. Plagiarism statement

I declare that I am aware of the following facts:

- As a student at the University of Luxembourg I must respect the rules of intellectual honesty, in particular not to resort to plagiarism, fraud or any other method that is illegal or contrary to scientific integrity.
- My report will be checked for plagiarism and if the plagiarism check is positive, an internal procedure will be started by my tutor. I am advised to request a pre-check by my tutor to avoid any issue.
- As declared in the assessment procedure of the University of Luxembourg, plagiarism is committed whenever the source of information used in

an assignment, research report, paper or otherwise published/circulated piece of work is not properly acknowledged. In other words, plagiarism is the passing off as one's own the words, ideas or work of another person, without attribution to the author. The omission of such proper acknowledgement amounts to claiming authorship for the work of another person. Plagiarism is committed regardless of the language of the original work used. Plagiarism can be deliberate or accidental. Instances of plagiarism include, but are not limited to:

1) Not putting quotation marks around a quote from another person's work
2) Pretending to paraphrase while in fact quoting
3) Citing incorrectly or incompletely
4) Failing to cite the source of a quoted or paraphrased work
5) Copying/reproducing sections of another person's work without acknowledging the source
6) Paraphrasing another person's work without acknowledging the source
7) Having another person write/author a work for oneself and submitting/publishing it (with permission, with or without compensation) in one's own name ('ghost-writing')
8) Using another person's unpublished work without attribution and permission ('stealing')
9) Presenting a piece of work as one's own that contains a high proportion of quoted/copied or paraphrased text (images, graphs, etc.), even if adequately referenced

Auto- or self-plagiarism, that is the reproduction of (portions of a) text previously written by the author without citing that text, i.e. passing previously authored text as new, may be regarded as fraud if deemed sufficiently severe.

References

[BiCS(2021)] BiCS Bachelor Semester Project Report Template. https://github.com/nicolasguelfi/lu.uni.course.bics.global University of Luxembourg, BiCS - Bachelor in Computer Science (2021).

[BiCS(2021)] Bachelor in Computer Science: BiCS Semester Projects Reference Document. Technical report, University of Luxembourg (2021)

[Armstrong and Green(2017)] J Scott Armstrong and Kesten C Green. Guidelines for science: Evidence and checklists. *Scholarly Commons*, pages 1–24, 2017. https://repository.upenn.edu/marketing_papers/181/

[KaufmannFranzinMenegaisPozzer] Lorenzo Schwertner Kaufmann, Flavio Paulus Franzin, Roberto Menegais, Cesar Tadeu Pozzer. Accurate Real-Time Physics Simulation for Large Worlds. *Universidade Federal de Santa Maria, Santa Maria, Brazil*, 2021.

[MurinKompisKutis] Justin Murin, Vladimir Kompis, Vladimir Kutis. Computational Modelling and Advanced Simulations. *Springer*, 978-94-007-0317-9-1, 2011.

[HeisterRebholz] Timo Heister, Leo G. Rebholz. Scientific Computing for Scientists and Engineers. *De Gruyter*, 2023.

[Unity] https://docs.unity3d.com/Manual/index.html

5. Appendix

All images and additional material go there.

5.1. Source Code

The following environment shows the correct and mandatory way to insert your code.

Listing 1: Caption example.

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int) #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```