OrbitCalc - A simplified version of the solar system

Friday 3rd November, 2023 - 09:49

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. In order of cutting cost to expensive missions, such as satellite launches and submarine drones, it is important to plan missions in high detail. Computational models can help to find the weak points of the mission, find extreme values of the devices and can help to set up a safe margin for the mission program. The development of one such computational model with a potential use case in space exploration is the topic of this Bachelor Semester Project.**

1. Introduction (± 5% of total words)

The importance of simulations in the scientific world is rising. Not only do they provide a less cost-intensive way of testing real life scenarios in a virtual twin of the current model of the world, but they do so in a cost-effective way. It is cheaper for example, to test new aerodynamics concepts in a virtual environment, where they are well changeable, instead of using wind tunnels. For these wind tunnel tests, the components have to be produced, the wind tunnel has to be powered on and a large amount of energy has to be used for powering the wind tunnel. Not only is this way of testing resource exhaustive, but also not very environmentally friendly, as a big portion of these prototypes will end up as scrap and can partly not even be used any more.
Simulations can provide near-real world environments, or at least emulate the behaviour of certain laws in physics. With these environments, we can reduce the amount of resources we use, by checking: is it the right way to achieve our goal? Can we optimize the behaviour of our test subject before producing it?

2. Project description (± 10% of total words)

2.1. Domains

2.1.1. Scientific.

- *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. These are used 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.

2.1.2. Technical.

- *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.
- *C#*: An object-oriented, higher level programming language in the .NET framework, which was heavily influenced by C++ and Java. C# is often used in cross-platform application development, game development and in Machine Learning applications.

2.2. Targeted Deliverables

2.2.1. Scientific deliverables. 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. This model can be emulated by a computer program.
As the title of this Bachelor Semester Project implies, the second part of the scientific part is about modelling a model with some constraints and simplifications. This model will include the basics of orbital mechanics, as well as the fundamental concepts of Newtonian physics.
The scientific part will tackle the following scientific question: *How can scientists 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. The results of this question are 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. This includes the examination of the utilized simplifications, methodologies and an analysis of the factors influencing the selection of specific approaches.
- *How can the model be modelled?* : With this question, the goal is to find equations and algorithms that facilitate an accurate representation of the computational model. This part is meant to include all the orbital mechanics theory needed.

As the goal is to find satisfying answers to the questions above, a state-of-the-art study will be conducted. Based on this study, the subsequent resources are used to address the scientific question:

- [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 objective 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.
The overall goal of this deliverable is to ensure, that the reader can understand the design decisions in the technical part. The results of this section will be used as a fundamental basis of the technical deliverable and the implementation.

2.2.2. Technical deliverables. 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.
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 for any further documentation regarding Unitys API.
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.
- *User controllable Satellite*: This will help the user experience navigating a satellite in a three dimensional solar system with objects influencing the trajectory of the space probe.

The program will eventually include a Graphical User Interface, for which some parameters are needed. So far it is planned to include the parameters for two objects, mainly x-y-z position of the objects. It will be determined in this section, which data is going to be displayed in a light-weight interface, which provides the user enough information for controlling a satellite and camera (exact number of cameras

to be determined in the report). For example, the data, currently planned to be displayed are:

- Velocity of user controlled camera: Measure how fast the user is going in the solar system
- Velocity of the satellite: Enable the user to calculate where the space probe will be when.
- Distance of the satellite to the sun: Locate the position of the satellite
- Coordinates of the satellite: Locate the satellite in a three-dimensional space
- Coordinates of the camera: Locate the user controlled camera in three-dimensional space
- Distance camera – Sun: Locate the camera on a relative basis in the solar system
- Labels of the planets: Locate the planets relative to the camera
- Planet orbit paths: Locate the orbits of the planets
- Satellite trajectory: Display the points, the satellite already passed to estimate the further trajectory

As the user will be able to control the camera, it will be required for this project, to map keys on the keyboard to certain movements, to enable the user to experience the virtual solar system. As of now, the keys are mapped as follows:

- W – Forwards movement relative to current rotation
- A – Left movement relative to current rotation
- S – Backwards movement relative to current rotation
- D – Right movement relative to current rotation
- X – Upwards movement relative to current rotation
- Y – Downwards movement relative to current rotation
- Arrow Key Left – Rotation Left (Z-Axis)
- Arrow Key Right – Rotation Right (Z-Axis)
- Arrow Key Up – Rotation Upwards (X-Axis)
- Arrow Key Down – Rotation Downwards (X-Axis)

Also the satellite will be controllable by the user, to ensure, that the user gets a better understanding of the complexity of navigating in a three-dimensional space under the influence of gravitational pull to other celestial bodies.

3. Pre-requisites ($[5\%..10\%]$ of total words)

Although all concepts used in the report will be explained in the report, it could be helpful, yet not mandatory, to understand the basic notions of the following topics. These are already known by the author before starting.

3.1. Scientific pre-requisites

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.

3.2. Technical pre-requisites

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 application programming interfaces (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.

4. A computational model

For each scientific deliverable targeted in section 2.2 provide a full section with all the subsections described below.

4.1. Requirements ($\pm$ 15% of section's words)

Describe here all the properties that characterize the deliverables you produced. It should describe, for each main deliverable, what are the expected functional and non functional properties of the deliverables, who are the actors exploiting the deliverables. It is expected that you have at least one scientific deliverable (e.g. "Scientific presentation of the Python programming language", "State of the art on quality models for human computer interaction", ...) and one technical deliverable (e.g. "BSProSoft - A python/django web-site for IT job offers retrieval and analysis", ...).

4.2. Design ($\pm$ 30% of section's words)

Provide the necessary and most useful explanations on how those deliverables have been produced.

4.3. Production ($\pm$ 40% of section's words)

Provide descriptions of the deliverables concrete production. It must present part of the deliverable (e.g. source code extracts, scientific work extracts, ...) to illustrate and explain its actual production.

4.4. Assessment ($\pm$ 15% of section's words)

Provide any objective elements to assess that your deliverables do or do not satisfy the requirements described above.

5. A Technical Deliverable 1

For each technical deliverable targeted in section 2.2 provide a full section with all the subsections described below. The cumulative volume of all deliverable sections represents 75% of the paper's volume in words. Volumes below are indicated relative the the section.

5.1. Requirements (± 15% of section's words)

cf. section 5 applied to the technical deliverable

5.2. Design (± 30% of section's words)

cf. section 5 applied to the technical deliverable

5.3. Production (± 40% of section's words)

cf. section 5 applied to the technical deliverable

5.4. Assessment (± 15% of section's words)

cf. section 5 applied to the technical deliverable

Acknowledgment

The authors would like to thank the BiCS management and education team for the amazing work done.

6. Conclusion

The conclusion goes here.

7. 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/

[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

8. Appendix

All images and additional material go there.

8.1. 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
```