OrbitCalc - A simplified version of the solar system

Monday 4th December, 2023 - 15:15

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. (3100)**

1. Introduction (190/400 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
- Shift-Left – Upwards movement relative to current rotation
- Ctrl-Left – 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

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

4.1. Requirements (122/150 words)

The requirement for this section is to explain with the given resources, the concept of a computational model. This includes the fundamental basics of how it came to be, as well as some insights on how they are usually developed and the scientific significance of them. For the explanation, it should be well understandable to any person, who understands the basics of computer science and is willing to read this bachelor semester project. This is to ensure that the author has well understood the topics of this section, as it will be the fundament of the upcoming sections and the technical part.
The results of this section will provide a step by step way to build a computational model of any kind.

4.2. Design (213/300 words)

As there are only few resources providing a general overview on computational models, this section of the report will mainly be consisting of information found in [Garrido]. Here, the following chapters will be used for finding results to the above scientific question:

- Chapter 1: Problem Solving and Computing:

  - 1.1: This is useful for finding the motivation for computational models. It explains the fundamental issue and already hints at, what is needed to develop such a model.
  - 1.2: As it is also the goal of this report to find a step by step set of instructions for building a computational model, this section already provides a basic skeleton for this set of instructions.
    Regarding 4, the outline of steps will be used to construct a more detailed instruction set than given in the resource.
  - 1.3: This section will be used to find different ways to tackle different mathematical difficulties. It will help to set up a guideline for transforming the mathematical model with formulas into a computational model with algorithmic approaches.

Next to these information, the observations made during the first weeks of the semester in the development of the application will be used. This will not only provide meaningful results, but also show the real world development of the models.

4.3. Production (514/400 words)

A computational model is a mathematical and algorithmic construct describing real world behaviours in formulas and procedures. With this in mind, finding a computational model needs a few steps for developing the model. This process is called *computational modelling* and involves a numerous count of observations to be made.
But all in all, the steps for setting up a computational model are quite general, as they are usually applied universally and then tuned to the individual model depending on the models complexity.

4.3.1. Observation.
The basis of modelling a computational model emulating an environment is observation. This enables to understand the challenges of the environment and to understand the problem. With these in mind, it is also useful to observe certain behaviours of different objects and note them down as much as possible, as it will help in 4.3.3 to find appropriate formulas to describe these behaviours.
Observation also enables to figure out factors with a minor impact, which might have to be left out for simplifying the model.

4.3.2. Problem Description.
(acc. to [Garrido], Ch. 1 Section 2) The problem description is one of the most important steps of designing a computational model. By providing a detailed problem description, taking into account all the important aspects, the model is supposed to take into account. This helps to prioritize different observations for later considerations of which aspects are must-have and which ones are nice-to-have aspects.

4.3.3. Mathematical foundations to a model.
To further advance in the model, it is usually required to find formulas describing the observed behaviours of the model.
As most models are used in interdisciplinary research, such as physics, mathematics, chemistry and biology, it often implies using existing formulas of this discipline to find a well-working environment for the computational model. The formulas for this have to be as complete as possible, as it will be the basis for later on designing the algorithmic approaches in the next step (4.3.4).

4.3.4. Algorithmic design.
As the mathematical foundations are known from 4.3.3, this step is about the way of computing, i.e. the algorithm. As computers are usually resource and power limited, this step is very important. There are several factors to consider when designing an algorithm for a computational model, such as recursive and iterative approaches.
In general, iterative approaches are more favourable, as they usually have a lower space complexity while providing the same results as recursive methods or at least provide a similar degree of accuracy.
Designing the algorithm also implies to simplify the mathematical model where computationally needed to reduce the overall space and time complexity of the program.

4.3.5. Guideline for computational models.
With all of these steps in mind, it is possible to reduce them to easy questions as a guideline:

1) What do we see? Which behaviour is observable?
2) What do we want to do? What is the problem we try to solve and which properties does the model need?
3) What formulas describe the behaviour of our model?
4) How do we implement the model in terms of algorithmic approach?

4.4. Assessment (0/150 words)

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

5. Transforming physics and observations to mathematics

5.1. Requirements (121/150 words)

This section requires to execute steps 1-3 from 4.3.5 in order of finding a computational model for the solar system. It will be required to provide subsections for every step of the guide, similar to the sections of 4. These sections will contain the scientific findings and important points of the model.
Not only the scientific findings of this section will play a central role for this section, but also how they correlate to the steps found in 4.3.5 and *why* the results found, are important. Based on this, the report will deliver the most crucial formulas of the model in this part, which are a necessity in 6. Also this section will provide some general simplifications, such as negligible factors.

5.2. Design (148/300 words)

As section 4 already contains a detailed look at computational models and computational modelling, this section will use the results of section 4 and combine it with the formulas describing the solar system. This will then provide a mathematical/computational model of the solar system, for which the goal is to model algorithmic approaches to emulate the behaviour of the solar system. This modelling will be step by step according to the guideline found in 4 and will illustrate the use of it on this example.

Additionally, next to the chapters used in the preceding section, this section will use the following sections of Chapter 1 of [Garrido]:

- 1.4: As this section is a more detailed description of Computational modelling with more details on the development process, which are not needed in 4.
- 1.8: Section 1.8 provides a software development approach to the problem of computational models.

5.3. Production (0/400 words)

5.4. Assessment (0/150 words)

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

6. Simplifications for ease of computations

6.1. Requirements (64/150 words)

This section will contain simplifications to the formulas found in 5.
It will also be required to take certain observations into account for simplifying the formulas. It will be required to be clear and well explained why these simplifications can and have to be made. This will include a critical computer scientific standpoint as well as a critical standpoint from the point of physics.

6.2. Design (95/300 words)

This section will take a look at the scale of the numbers used in the model developed in 5 and point out floating point and precision issues. Furthermore it will simplify some equations for ease of computation and also simplify some behaviour into constants.
Specifically, this section will use Chapter 1 of [Garrido]. Section 1.11 of this chapter is especially important, as it provides the necessary information regarding floating point precision, errors and accuracy.
To find issues in precision, the sections 4 and 5 will be used to show the errors or complex calculation made.

6.3. Production (0/400 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.

6.4. Assessment (0/150 words)

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

7. The solar system

7.1. Requirements (0/150 words)

In this section it is required to find explanations and pseudocode for the implementation of the solar system. This section will focus on the implementation of the "stable orbit script" (SOS) which is mainly responsible for the elliptical orbits in this system. It will also be required to look at the results from 5 and 6 as they provide the *theoretical* aspects of this part. With these as the main points covered, it will also be necessary to take a look at the game objects and the addition of a script to such an object. This will include a detailed glimpse at Unitys scripting API.

7.2. Design (0/300 words)

7.3. Production (0/400 words)

7.4. Assessment (0/150 words)

8. Movement in three dimensions and time control

8.1. Requirements (76/150 words)

This section will be mostly about the camera, camera controller and the time controller scripts of the Unity project. It will focus on the coding aspects as well as some mathematical background from Linear Algebra, such as Vectors and point transformations. Since this section is not based on physics, it does not rely on results from 5 and 6. Instead, it will illustrate based on pseudocode the algorithmic approach of movement in three dimensional space.

8.2. Design (0/300 words)

8.3. Production (0/400 words)

8.4. Assessment (0/150 words)

9. Satellite and gravity

9.1. Requirements (79/150 words)

For this section it is required to provide the most important explanations on how Gravity and the satellite object are implemented in the editor. This includes (Pseudo-)Code snippets for the most important functions. Also section 8 has to be taken into account, as also the satellite object has to move in three dimensions.
Most importantly, this section takes results from 5 and 6 to provide an answer to the question: *How is satellite movement implemented in this program?*

9.2. Design (0/300 words)

9.3. Production (0/400 words)

9.4. Assessment (0/150 words)

Acknowledgment

The author would like to thank his tutor for helping and guidance in the project and the Hochschule Trier for the Computer Science Summer Camps 2019 and 2021 which guided him to Computer Science and simulations.

10. Conclusion

The conclusion goes here.

11. 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.

[Garrido] Jose M. Garrido Introduction to Computational Models with Python *Chapman and Hall/CRC*, 2016.

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

12. Appendix

All images and additional material go there.

12.1. Source Code