ONTARIO TECH UNIVERSITY
FACULTY OF SCIENCE, COMPUTER SCIENCE

Project Group: 1 March 31, 2024

Simulation and Modeling Course Project: Simulation of our Local Solar System.

by

Sandeep Virk Ginthushan Kandasamy

Abstract:

This project aims to present a comprehensive Python-based simulation and visualization of the local Solar System to Earth. Its goal is to provide an educational tool for understanding the dynamics of the celestial neighborhood. This project offers a detailed representation of orbits, and planetary positions in realtime with usage of real data, whilst trying to mimic the appearance of the solar system to the best of its ability. Through this tool, we try to foster a deeper appreciation and understanding of the complexities and beauty of the Solar System.

1 Introduction

The Solar System, an intricate dance of various types of celestial bodies interacting with one another, have captivated humanity since time. Our understanding of this cosmic neighborhood has evolved from ancient mythologies, to more precise and mathematically accurate models. Within modern technology, computational simulations are the stapling tool in visualizing and simulating such vast and gigantic interacting elements. This project is an extension of that computational simulation, which leverages the power of Python, a versatile programming language with a diverse set of libraries ready to be used, to create a magnificent simulation of our local Solar System. Combining principles of celestial dynamics, and computational techniques such as integration, the aim is to provide immersive educational experiences through this simulation to students, educators, and enthusiasts.

Python, chosen for the vast availability of scientific libraries, serves as the optimal platform for developing this project. This project utilizes libraries such as Pygame-GUI, aiding to create paths and dynamic visualization of orbits, NumPy for efficient mathematical computation, PyGame being the graphical and interactive backbone of the simulation, and random library serving as the ideal library to generate the random attributes within the Solar System in appearances and celestial bodies. Further information on the usage of libraries will be provided later in the document.

2 Technical Implementation and Functionality

The core objective of the simulation as stated before is to simulate orbital mechanics of planets and other celestial bodies around the Sun. This provides as an educational tool being accurate and engaging. A lot of this was completed by using update rules for each celestial body and its physical properties. Each celestial body was divided into which class they can fall under, for example a moon could be classified as just another 'planet', or 'asteroid' rather than declaring an entire new moon class. All object classes are declared within planet.py file which consist of Planet, Asteroid, and AsteroidBelt classes.

2.1 Planet Class

The Planet class covers the majority of cosmic bodies, from planets themselves, to suns, and acts as a parent class for asteroids. This class includes class variables

such as: x,y position variables for visualization, radius variables for the size of the planet to draw which is represented as just a circle with respective radius. The planet class also includes colour and mass of each planet for diversity and realism, while maintaining the properties of the actual planets in the real world. To simulate the attraction of planets, Newton's law of universal gravitation was used to calculate the gravitational force between two bodies using the formula:

$$F = G\frac{m_1 m_2}{r^2}$$

Here, F denotes the gravitational force, G is the gravitational constant, m_1 and m_2 are the masses of the two bodies, and r is the distance between their centers. Once F has been found it is separated in its horizontal and vertical components, using trigonometry, and passed to the update_position method. In the update_position method, the planet's velocity and position are updated using the following formulas:

$$a_x = F_x/m$$

$$a_y = F_y/m$$

$$v_x \mathrel{+}= a_x \cdot \Delta t$$

$$v_y \mathrel{+}= a_y \cdot \Delta t$$

$$x \mathrel{+}= v_x \cdot \Delta t$$

$$y \mathrel{+}= v_y \cdot \Delta t$$

This iterative process is repeated to simulate the motion of celestial bodies within the system. The draw methods are similar to that within the course, which just uses the x and y coordinates of the instance and draws the pygame circle at the position.

2.2 Asteroid Class

The Asteroid class is a child class to the Planet Class. This is due to the fact that Asteroid shares a majority of initialization variables. The major difference is that asteroids are used primarily for appearance and appeal to resemble the real life

asteroid belts. It is extremely computationally expensive to calculate forces and update each of these asteroids with complex integration methods or update laws. The asteroid class would not only be used in terms of creating the asteroid belt, but also be re-used to create moons revolving around planets. It was attempted to create many 'asteroids' similar to using realistic mass and forces, however, it resulted in the application running poorly, and the simulation could not perform properly. Due to this, it was decided that the asteroid child class needs a newer update method. The update method was changed to just use simple orbits without the use of forces, which still use a legitimate radius of orbit, although are not updated based on forces and mass. In this way, the simulation saves a lot of computation, yet allows for the simulation to having an asteroid belt for the realistic look of the Solar System.

2.3 AsteroidBelt Class

The AsteroidBelt class is the extension and use of the Asteroid class. It simplifies the creation of asteroid belts in which there are two main ones around our Solar System. For this reason, this class is created as a helper function and provides an orbit parent for each asteroid and the belt in general. For example the main asteroid belt uses the Sun as its reference to orbit around, all instances of asteroids share this and are created within this class. This class also utilizes a random range in which asteroids are distributed at random throughout the belt at various incrementing angles, and this allows for a more scattered and realistic asteroid belt as well.

Now is also a good time to mention that global variables were used to represent specific units. The entirety of the project uses Astronomical Units, where 1 AU = Distance from Sun to the Earth. Respective to this unit, all other units are represented and distances are calculated so the code doesn't look as crowded. The time step unit is also included, where each time step is now equal to 1 day rather than calculating them in seconds to speed up the simulation, however this is modular and can change the speed of the simulation. The scale is respective to the astronomical units. Realistic units and unit conversions are used in order to make the simulation follow their real life physical properties.

3 PyGame Utilization

At the heart of the project is Pygame, which facilitates the rendering of the Solar System in real time. Pygame is particularly suited for creating interactive applications and easily use a canvas. In this simulation, it is no different. It uses a window, and it is interactive by allowing users to pan using the arrow keys to different sections of the solar system. It also allows for zooming in and out to show certain planets, versus not showing others when you zoom far enough in perspective. The interactivity and the versatility in changing time steps, allows for easy and quick changes for the learning experience. It also features the ability with sliders and toggles to engage and disengage orbital paths, distances from the sun, and speed of the simulation. This aids in the quick interactive interface to improve user experience. PyGameGUI, an extension of the Pygame library is used in order to create this, alongside drawing the orbital paths for each planet.

4 Random Utilization

To introduce elements of unpredictability and enhance the simulations realism, the random library must be used. Pythons random library does a great job at producing random numbers and it had to be used especially in the creation of the asteroid belt. Creating known planets is easy enough, but creating asteroids must be done randomly as there simply is not enough data, and computationally expensive to instantiate each asteroid with its respective mass, shape, velocities, and distance from the sun. For this reason, asteroids are randomly given a distance from the sun within a certain range, which gives us an orbiting ring of asteroids or moons with ranged bands.

5 Conclusions

In conclusion, this Solar System simulation project marks a significant advancement in making the wonders of our universe more accessible and engaging. Through this development of an interactive simulation, we have provided an application and simulation where users can explore the complexities and realistic properties of celestial bodies as well as their orbits. The incorporation of being able to change the time-step and speed up or slow down the simulation, allows users to understand at their own pace. The success of this project lies in its ability to transform ab-

stract astronomical concepts which use large numbers, into experiences that are tangible.

Some lessons learned in improving projects like this could include the ability to integrate numerous bodies. Integration and solving equations and forces or even using update laws for a large number of celestial bodies is extremely computationally expensive, and it does not always warrant the need for implementation in terms of using integration techniques or update laws. Instead, a different approach was used to grasp the idea of certain aspects of the Solar System. Not all realistic simulations are easy to simulate, and some are too large or too numerous to be able to perfectly simulate. This was a great lesson, and difficult problem to overcome within this project.

Looking forward on this project, it is a strong foundation that can be built upon for further innovation. Future iterations could explore different angles of viewing, or deepen the interactions. Another example could include improving the graphical qualities, and allowing closer imaging for a more realistic view on the celestial bodies. The simulation and project also incorporates the opportunity for new discovery, and is easily modified to include such new discoveries without difficulty.

Ultimately, this Solar System simulation project blends technology, computation, programming and education, and provides an educational and unique experience to anyone interested in learning about the Solar System. It champions the idea that with curiosity as our guide, the vastness of space is not beyond our reach, but rather a canvas for our imagination.

6 References

- Content Editors, S. (n.d.). Solar system exploration - NASA science. NASA. https://science.nasa.gov/solar-system
- Gruppetta, S. (2023, November 12). Simulating orbiting planets in a solar system using Python. The Python Coding Book. https://shorturl.at/itAGU
- He, C. (2020, February 7). Simulating the solar system with under 100 lines of Python Code. Medium. https://shorturl.at/bexU3

- Ruscica, T. (2022, February 20). Planet simulation in Python - tutorial. [Video]. YouTube. https://www.youtube.com/watch?v=WTLPmUHTPqo&ab_channel=TechWithTim
- Ruscica, T. (2023, October 15). Intermediate Python tutorial — Gravitational slingshot simulation. [Video]. YouTube. https://www.youtube.com/watch?v=HTfwhmHVpqM

A Appendix

A.1 Mercury

Data for Mercury was sourced from the following link:
https://nssdc.gsfc.nasa.gov/planetary/factsheet/mercuryfact.html

A.2 Venus

Data for Venus was sourced from the following link:
https://nssdc.gsfc.nasa.gov/planetary/factsheet/venusfact.html

A.3 Earth

Data for Earth was sourced from the following link:
https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

A.4 Mars

Data for Mars was sourced from the following link:
https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html

A.5 Jupiter

Data for Jupiter was sourced from the following link:
https://nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html

A.6 Saturn

Data for Saturn was sourced from the following link:
`https://nssdc.gsfc.nasa.gov/planetary/factsheet/saturnfact.html`

A.7 Uranus

Data for Uranus was sourced from the following link:
`https://nssdc.gsfc.nasa.gov/planetary/factsheet/uranusfact.html`

A.8 Neptune

Data for Neptune was sourced from the following link:
`https://nssdc.gsfc.nasa.gov/planetary/factsheet/neptunefact.html`