VRIJE
UNIVERSITEIT
BRUSSEL

SIMULATION OF THE SOLAR SYSTEM IN PYTHON

Course: Advanced Programming Concepts

SEPPE LAMPE & PETIO TODOROV
2019-2020

Professor: Bart Jansen

1. Introduction

To familiarise ourselves with Object Oriented Programming in Python, a simulation of the Solar System has been constructed. The work was carried out as follows. Both Petio and Seppe researched how to model the equations guiding the motion of objects in the Solar System separately before coming together on one final system. Petio explored translating some of the equations provided by NASA JPL for computing the positions of celestial objects in orbit but this system proved to be too complex. Seppe took a more straightforward approach of modelling the solar system by calculating the forces between each of the bodies in the system in order to simulate their motion. This is the final approach that was settled upon. Petio suggested to improve this model by changing the subclasses of CelestialBody to Star and Satellite and to change the SolarSystem class to a RevolvingSystem class which is also able to simulate any central body with its satellites.

2. Materials and Methods

The visualisation of the objects was performed via the VPython package. The project contains two different major type of classes and one important function. The function 'gforce' computes the gravitational force between two objects and is the basis for the simulation. This gravitational force function was mainly based on code made available online by 'wlane' on https://www.glowscript.org/#/user/wlane/folder/Let'sCodePhysics/program/Solar-System-1/edit.

The first major class is called 'CelestialBody' and has two subclasses (Star and Satellite). The parameters for creating a CelestialBody instance are mass, radius and colour. Star takes no additional arguments and has the default values set to match those of the Sun. Satellite takes 'parent', 'distanceToParent', 'orbitalPeriod', and 'inclination' as additional parameters. Parent is the body around which the satellite revolves, distanceToParent the average distance to its parent, orbitalPeriod the time (in Earth days) it takes to revolve around its parent and inclination is the inclination (in degrees) of its pathway in the equatorial plane.

The second class is 'RevolvingSystem' and handles the simulation of Satellites around their parents. It takes one argument i.e., centralBody. This is the body which is going to be at the centre of the simulation. The class has two main methods and some helper methods. The two main methods are 'addSatellite' and 'simulate'. The former takes one argument, a satellite. It first calls the recursive method 'checkParent', this method recursively checks whether the parent of the satellite is the centralBody, if so then the function returns True. Otherwise it checks whether the parent is a Star, if so then the function will return False. Otherwise 'checkParent' will be called on the parent body. If this method returns True then two other helper functions are called i.e., 'generatePos' and 'generateMomentum' which respectively calculate the start position and start momentum for the object. These methods first check whether the 'pos' and 'momentum' of the parent body has already been calculated. If not then these methods will first be called on the parent. Once the parent pos and momentum have been calculated then the pos and momentum of the body can be calculated. Lastly, the satellite is added to the 'self.objects' list.

The second main method of the RevolvingSystem class 'simulate' handles the visualisation and movement of the objects in the system. It takes one argument 'dt', which specifies the speed of the simulation in Earth days/second. Firstly, the method creates a canvas and then a VPython sphere object for every satellite in self.objects. It does this by looping over the self.objects list and calling the 'show' method on each of its elements, the show method regulates the creation of VPython sphere instances which are automatically stored by VPython in self.canvas.objects. The simulate method then calls the show method on the centralBody before it starts an endless loop. The loop keeps the simulation running until the user closes the browser window that contains the canvas or until the user stops it manually in Pycharm. A cycle of this loop contains two distinctive parts. In the first part the spheres in self.canvas.objects are traversed and the methods 'updateForce' and 'updateMomentum' are called on each sphere object. The updateForce method uses the previously mentioned gforce function to calculate the gravitational force between a certain body and every other body in self.canvas.objects except that body itself. The updateMomentum method then accordingly updates the momentum of the sphere based on the new gravitational force vector. After this has been done for each element the second part of the cycle is initiated where a last method is called i.e. 'updatePos'. This method updates the position of each sphere based on their momentum, mass and the value of dt. The cycle of the loop is now finished and it will start a new cycle. There is also a rate(100) call; rate is a function in VPython which is required for the animation of events in canvas. As explained in the following source:

https://www.glowscript.org/docs/VPythonDocs/rate.html, "if you place rate(50) inside a computational loop, the loop will execute a maximum of 50 times per second, even if the computer can run faster than this." In our program the rate(100) call affects the input parameter dt, which has units earth days per second. We have to convert dt into seconds before recalculating forces, momentum, and position. To convert dt to seconds you have to multiply it by 60*60*24 but because we have the rate(100) we do 100 computations per second and that's why the dt is not 60*60*24 but 60*60*24/100 = 36*24 seconds.

3. Results

Three different systems have been modelled, the solar system, the Earth with the Moon and Jupiter with its Galilean moons. The Earth and Jupiter systems were modelled without any difficulties (Fig. 1). The solar system has some obscurities e.g., if the model is run too fast (dt is too large) then the visualisation of the inner planets behaves non-optimally: some moons might even become dislocated from their trajectory (Figs. 2 & 3) and the moons are not clearly visible due to their close distance to other objects (Fig. 4).

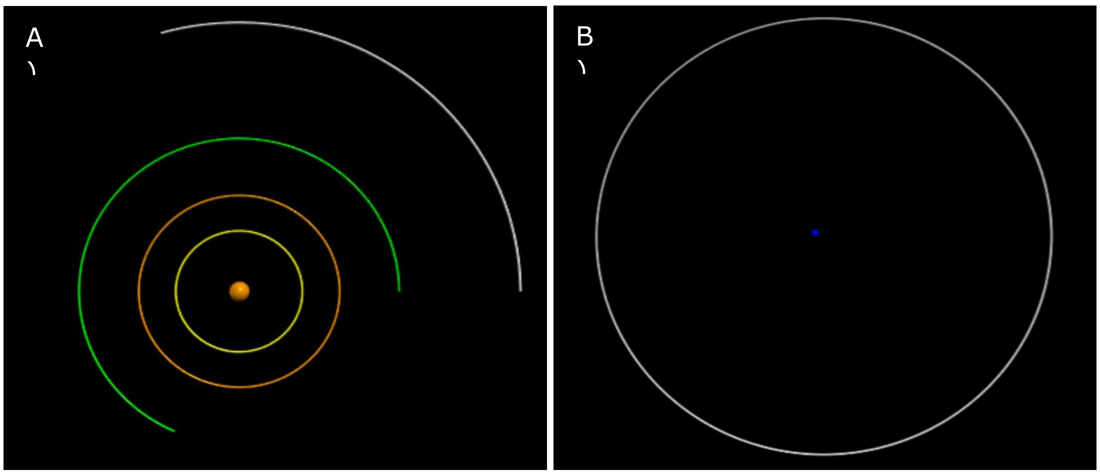

Figure 1: A) Jupiter with its four Galilean moons, from inner to outer orbit: Io, Europa, Ganymede & Callisto. B) The Earth with the Moon revolving around it.

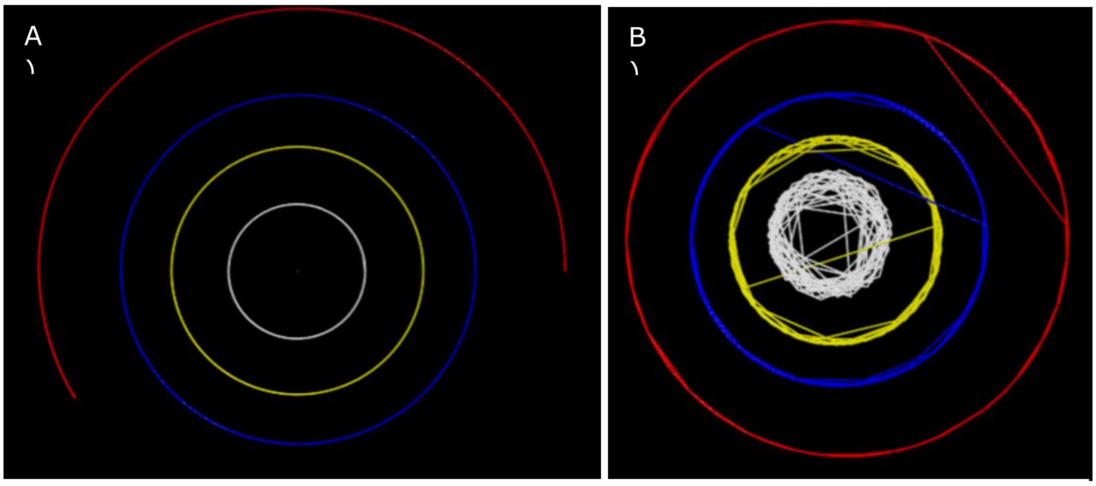

Figure 2: A) The simulation of the Solar System with a dt=10. B) The simulation of the Solar System with a dt=365.

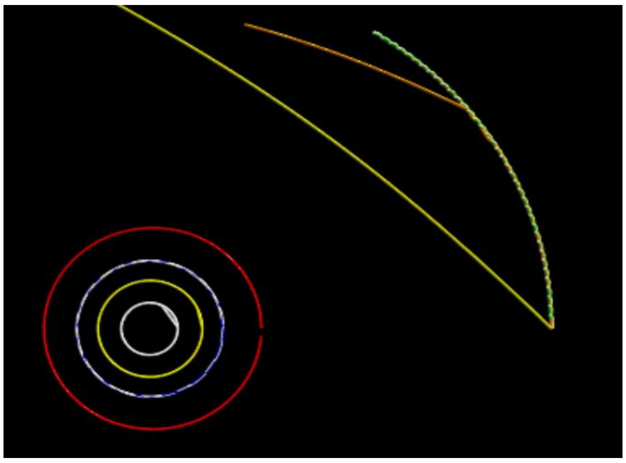

Figure 3: The dislocation of Io (yellow) and Europa (orange) from their respective trajectories around Jupiter. This simulation was run at a dt of 50.

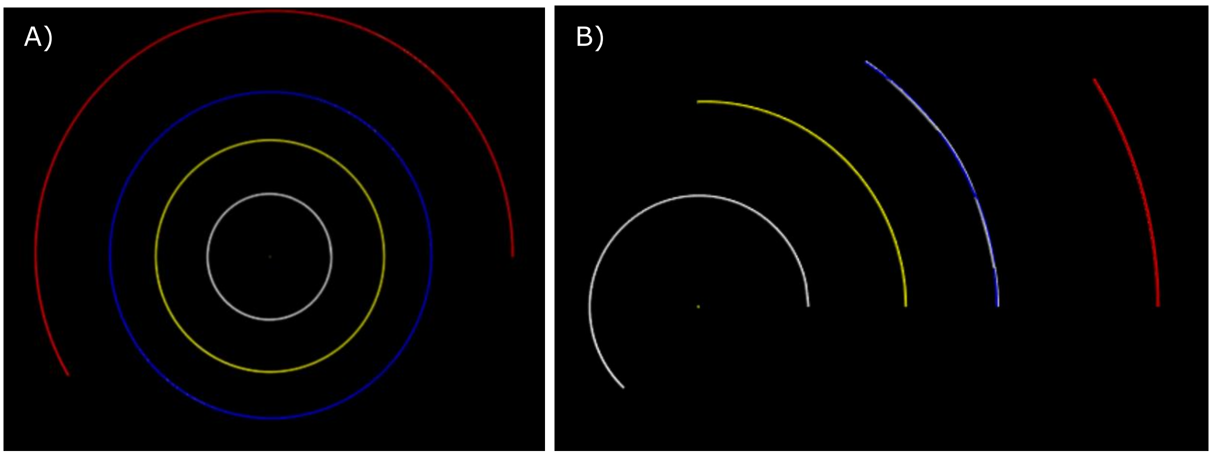

Figure 4: A) The simulation of the inner planets of the Solar System, without the Moon. B) The simulation of the inner planets of the Solar System, with the Moon included. It is clear that the colour of the Earth's trajectory changes from blue to white periodically but it is impossible to distinguish the two pathways.

4. Discussion

As displayed in Figures 1-4 the model is able to simulate the Solar System or any other real system where objects revolve around a central body. However a few issues arise, mainly when the simulation is run at very high speeds. Most significant is the escape of moons, depicted in Figure 3. A correlation was found between the orbital period of these escaping moons and their tendency to escape. The lower the orbital period of an object the more likely it is to escape if the simulation is run at higher speeds. For example at a dt of 50 Io revolves ~30 times per second around Jupiter. We believe that this escape might be due to small rounding errors resulting in a significant error in the physical calculations if the model is run too fast. As an increase in speed results in larger jump in space every time step, small errors otherwise corrected by sequential calculations of the gravitational force and momentum might now exceed a certain threshold.

The model is able to correctly simulate the inclination of a satellite as shown in Figure 5.

A smaller issue is the distortion of the pathway depicted in Figure 2. This distortion is most likely caused by the fact that the trails are drawn between two consecutive points. If between two time steps the object changes its position considerably then this will be visible as a straight line between its current and previous position (e.g., the pattern of Mercury in white in Fig. 2a). Furthermore it does seem that at certain moments the processor gets interrupted and does not draw one or more lines. If the object is moving slowly then this will be quite well hidden, but if the object is moving quite fast along its pathway than this can be seen quite clearly as well (e.g., the blue and red straight line in Fig. 2b).

As mentioned in the intermediate report the force and momentum of the central body are not updated. This is because the central body shows a small, overall drift in space if the model is run for a long enough time (Fig. 6). This does not have any real consequences since the satellites keep rotating around the actual object and not its original position, as it should be. However, for visual aspects we let a tracer fill the path where each planet has been. In normal circumstances these are ~circles and they stay that way. But with the beforementioned drift, the centres of these circles (the central body's position) start to slightly move, resulting in non-overlapping paths for the planets (Fig. 6). If these traces would be removed then the drift would frankly not be noticeable. However, as explained further below, the use of these traces can be useful. Moreover, research has led us to discover that actually the Suns position in our solar system changes slightly with time but not exactly as ours does (Fig. 7; Jose, 1965). But the intricacy and complexity of our solar system is impossible to model with just one law of physics and trying to model everything perfectly would take us to theoretical physics far beyond the scope of this project. Therefore, we feel it justified to manually fix this small problem, by simply not calculating the force the satellites have on the central body. These forces have a much smaller effect on the central body because its mass is usually much greater than the mass of its satellites.

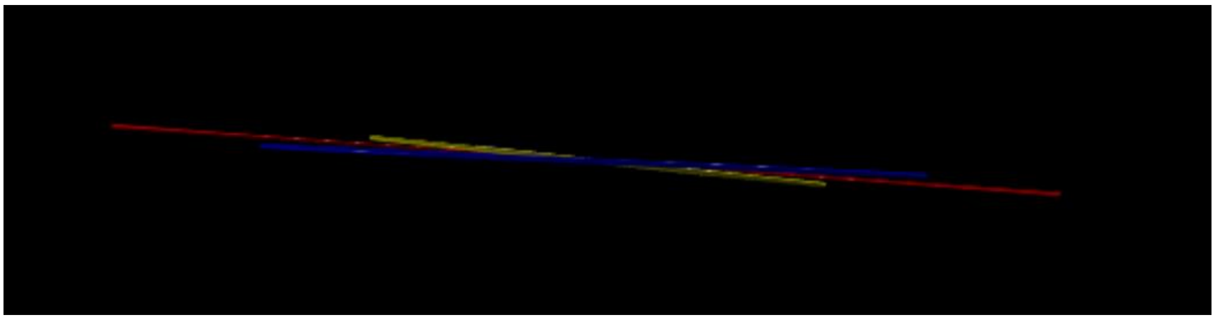

Figure 5: A 'side-view' of the trajectories of the inner planets of the Solar System. It is clear that they all have a similar yet distinctive inclination which is visualised well in the simulation.

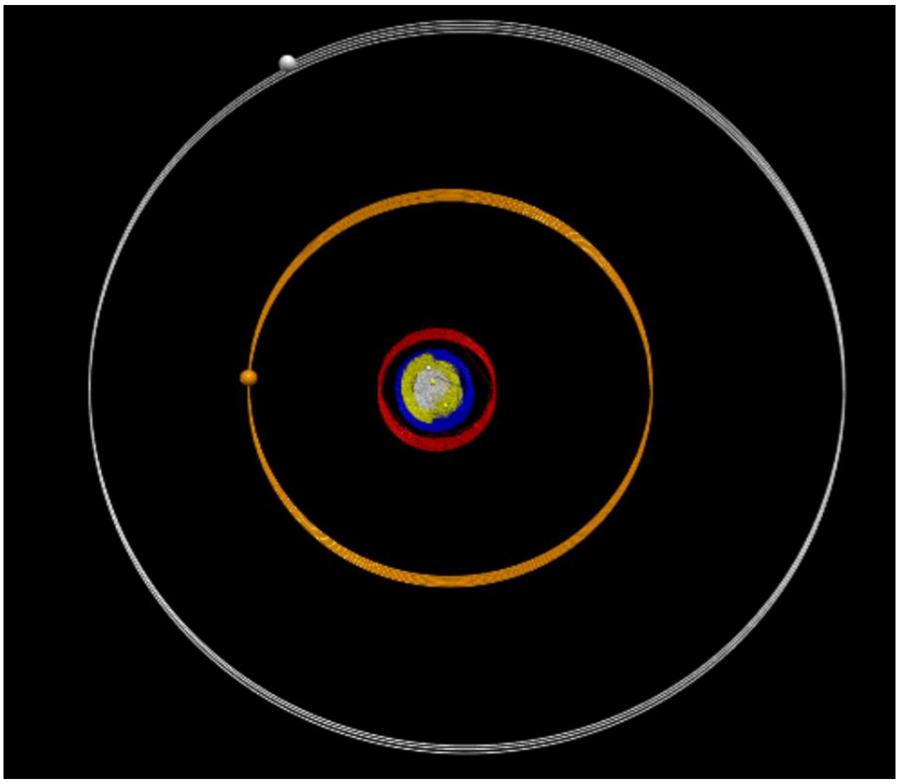

Figure 6: The drift of the Sun is visible by looking at the trails of the outer shown planets (Jupiter in orange and Saturn in white). At this time ~100 years had passed. Note that the trails of Venus and Mercury should be disregarded as this model was run at a relative high speed for those planets (~2 and 4 rotations/second respectively) and as a result our laptops sometimes draw skip some time steps.

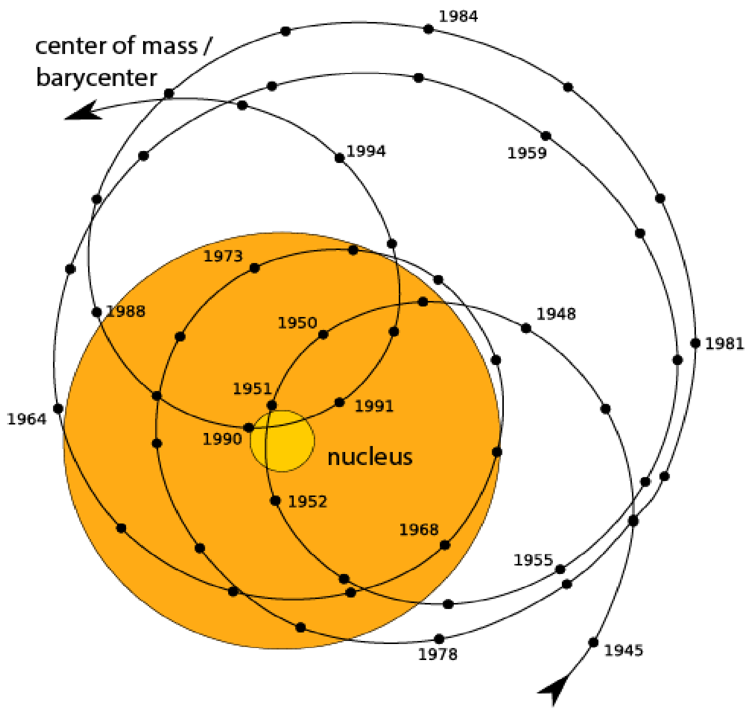

Figure 7: The drift of the Sun around the barycentre, the center point of all the mass in the Solar System (derived from https://www.education.com/science-fair/article/barycenter-balancing-point/).

Lastly, there is one thing still missing from the project which is a GUI. Although the project goal was to familiarise ourselves with object oriented programming, we feel that it would have made the project look much more professional.

5. Conclusions

The model is able to simulate the motion of objects around a central body in a representative manner. However, running the model at high speeds affects the objects with a small orbital period in several negative manners. It is therefore advised to either run the simulation at a mediocre speed ($dt < 15$) or to disable satellites with a low orbital period if high speed simulations are desired. A positive addition to the project which can be implemented in the future would be a graphical user interface.

Bibliography

Jose, P. D. (1965). Sun's motion and sunspots. *The Astronomical Journal*, *70*(3), 193-200.