Satellite project, AST 1100

Introduction and useful information

0.1 Project overview

In this project you are going to send a satellite from your home planet, in your own personal star system, to visit another planet in that system. To do this you will use much of the physics you learn in the course and put it into use when solving the different the various problems you will encounter.

The project will consist of 7 parts:

1. In the first part you are going to simulate a simplified rocket engine to help you calculate the amount of fuel your are going to need to boost the rocket to the required location.

2. In the second part you will calculate the orbits of all the planets in your star system and visualize the movement.

3. In the third part you will calculate the trajectory of the satellite from your home planet to your chosen destination. You will also try to find out what instructions to give to the satellite so that it can reach the planet that you want to visit.

   Bare in mind that the first 3 parts are only a preparation for part 4-7. It's in these later part where we actually send out the satellite and check if our previous parts work correctly.

4. In the fourth part you will write the software to allow the satellite to look around and orient itself to find out where it is and what it's velocity is.

5. In the fifth part you will send the satellite on its way. You will use what you made in part 3 to check and correct the calculations that you did in part two along the way to your destination.

6. Once you have reached the intended planet, part 6 will consist of going into a very low orbit above the planet and taking photographs of the surface. You will then choose a nice spot to land.

7. In the last part you will be launching the lander module from your satellite and try to land it safely on the surface of the planet.

0.2 Important information

Read this section carefully before starting your project:

- The first 4 parts, all though equally important as the next 3 as these lay the groundwork for the others, are your own simulations of how you think the rockets and the planet trajectories will be. When we actually link your work up to the main class AST1100SolarSystem at the last 3 part of the project you find probably find that your rocket is not in the place you thought it should be. Don't panic, this is quite normal and actually intended as the last 3 parts are devoted to locating and adjusting your trajectory so that you might safely land on your designated planet.

- Python is the language to use for all of the coding in this project. The module you will recieve, and need to use, is a python module, so you need pyhon, or some amount of cleverness, to succeed!

- For this project you will all recieve an integer that is your personal seed. This seed determines all the physical properties of the star system that you live in, so everyone gets their own unique star system. This means that you can work together and discuss the problems, but you all have to solve all the problems and write all the code yourselves.

- Although we try to make this project as realistic as possible, we also make alot of simplifying assumptions. These assumptions are also used in the python module "AST1100SolarSystem" that you will use and interact with, so that these assumtions will be built into the virtual world that your satellite is exploring. This is so that some of the problems become easier to solve, but this also mean that if you try to solve the problem in the "correct" way, e.g. by including forces that we specifically say that you should neglect, your results will be wrong. In other words, although it is generally good to be clever, don't try to be too clever!

- We will give you a set of exercises that you will have to do, these exercises will guide you through planning and executing your satellite trip. There is however nothing stopping you from playing around on your own. Using the tools that you will develop and the "AST1100SolarSystem" python module you can explore whatever part of the star system that you would like to see!

- Your resulting grade will mainly be based on a single scientific report you will hand in at the end of your project, where you describe everything you did in part 1-7. It is advised that you write notes for yourself so that you remember the details of what you did, even after several months.

- It is very possible that you will encounter bugs or inconsistencies in the module "AST1100SolarSystem". If you think this is the case, then contact the group teacher responsible for this project, and hopefully this can be sorted out.

0.3 What you need

Basically you need just a few things to start this project, you should find a link to all of them on the course webpage. Be sure to check back often, since bugs are constantly being fixed in both the python module and the 3D app.

- A full installation of Python 2.7, with some data science packages. If you are installing on your own computer (any operating system), Anaconda is recommended to get all the needed packages. Specifically the python module needs nympy, scipy and lxml. Plus some plotting package is recommended, like matplotlib.
- The AST1100SolarSystem.pyc module. This is a compiled python-file, which can be included like any other package.
- A personal seed. To generate this seed, use the myseed.pyc script, with your UiO username, like this:

```
terminal>python myseed.pyc johndoe
76945
```

- To get the visual effects in full 3D, you will also need to download AST1100 Mission Control (MCast).

0.4 The AST1100SolarSystem class

In order to find information about your own virtual star system you need to use the AST1100SolarSystem class. To use this class just put the "AST1100SolarSystem.pyc" file in the folder of your sourcecode and include it. To get your information you need to make an instance of this class. If your seed is e.g. 42:

```
from AST1100SolarSystem import AST1100SolarSystem
seed = 42
myStarSystem = AST1100SolarSystem(seed)
```

All the properties of your star system are given as variables in this class, so you can access them directly once you have made an instance:

```
massOfMyStar    = myStarSystem.starMass
radiusOfMyStar  = myStarSystem.starRadius
N               = myStarSystem.numberOfPlanets
```

The planet properties are numpy arrays, so you need to index them with a number. The number of your home planet is 0, otherwise the numbers are not in any particular order:

```
massOfHomePlanet    = myStarSystem.mass[0]
radiusOfHomePlanet  = myStarSystem.radius[0]
```

Here is a complete list of the different properties:

```
# Data about the star system.
numberOfPlanets # Number of planets in the system.
temperature     # Surface temperature of the star, [K].
starRadius      # Radius of star, [km]
starMass        # Mass of the star, [solar masses].

# Data about planets, lists with index.
a          # Semi-major axes of planets, [AU].
e          # Eccentricity of planets.
radius     # Radiuses of planets, [km].
omega      # Initial angle of planets' orbits.
psi        # Angle of semi-major axis for the planets.
mass       # Mass of the planets, [solar masses].
period     # Rotational period of planets, [earth days].
x0         # Initial x-position of planets, [AU].
y0         # Initial y-position of planets, [AU].
vx0        # Initial x-velocity of planets, [AU/Yr].
vx0        # Initial y-velocity of planets, [AU/Yr].
rho0       # Atmosphere density at surface, [kg/m^3].
```

You will recieve more information about how to use different parts of this class as you will need it.

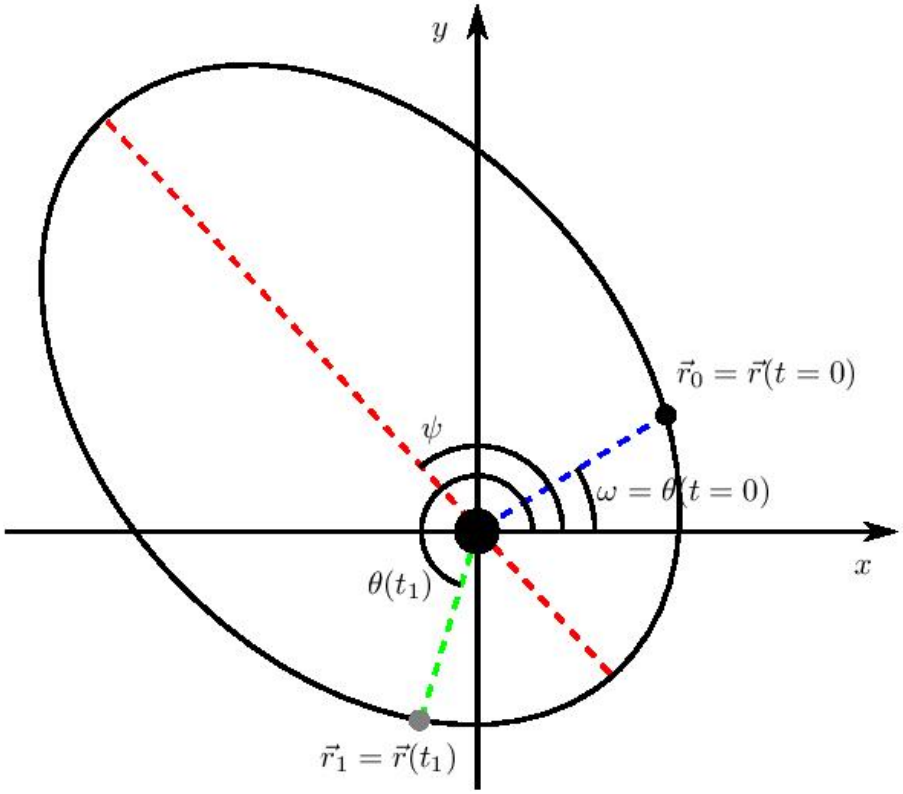

Figure 1: Figure to explain the geometry of the orbit. The figure shows the whole orbit of the planet, as well as the position at two different times, $t = 0$ and $t = t_1$.

0.5 Specifications of the satellite and lander

Here is some technical information about the satellite and the lander that you will need later:

	Satellite	Lander
Mass [kg]	1100	90
Cross-sectional area [m^2]	15	6.2 (241 w/pc)

Note that the lander has a parachute (pc) that it can deploy during landing.

0.6 The first exercise

This part only concerns those of you planning on using your own computer to solve parts of or the entire project.

The university computers run Python version 2.7.3, and the data about your star system relies heavily on drawing random numbers using the Python `random` module. You should check that your own Python version outputs the same sequence of random numbers as one of the university computers, given an identical integer seed. This can be done with the `testRandomNumberGenerator` function. So the first thing you should do is to run the following simple script from a directory in which the `AST1100SolarSystem` module is saved:

```
from AST1100SolarSystem import *

system = AST1100SolarSystem(1)
system.testRandomNumberGenerator()
```

This will print to terminal whether the test passed or failed. If it passes, you may simply move on. If the test fails, it most likely means you are running a version > 3.2 Python. In order to fix the discrepancy between the generators, be sure that you have python version 2.7 installed. If the test still does not pass, you should contact the group teacher responsible for this project and try to sort it out.