AST1100SolarSystemViewer

In the weeks to come you will be given some assignments asking you to calculate orbits, creating a rocket engine and sending a satellite into space. In order to achieve this you will need access some codes. There are 2 main codes; a python class called AST1100SolarSystemViewer and graphics applications called MCast and SSView. The class creates a solar system based on a personal random seed value that you will be given. The class contains all the information you need about the solar system to calculate orbits, launch a satellite and make videos as well as methods to check you calculations and your progress.

How to get your individual seed

The python module myseed.pyc, published on the course webpage, should be used to generate a 5-digit number based on your UiO username. In a terminal, find the folder where you downloaded myseed.pyc, and write the following:

```
> python myseed.pyc uiousername
```

The number generated is associated to your UiO username, and should be different for all students.

How to access the class AST1100SolarSystemViewer

The class is accessed by import from its code which is also called AST1100SolarSystemViewer using your designated seed. You need to have AST1100SolarSystemViewer.pyc placed in the same folder you are working in. This is an example of how to access the class.

```
from AST1100SolarSystemViewer import AST1100SolarSystemViewer
seed = 456
system = AST1100SolarSystemViewer(seed)
```

Using the commands above you will access the class. To get out the information you need, ie the mass of your sun, simply set a new variable called sunMass = system.starMass. The class contains information such as how many planets you have available in your system, their mass, their initial positions and much more. This is a list and example of which variables you will need and how to access them using the previous commands from above.

```
planetsRadius = system.radius      # Radiuses of planets, [km].
planetsMass = system.mass          # Mass of the planets, [solar masses].
planets_initial_x0 = system.x0     # Initial x-position of planets, [AU].
planets_initial_y0 = system.y0     # Initial y-position of planets, [AU].
planets_initial_vx0 = system.vx0   # Initial x-velocity of planets, [AU].
planets_initial_vy0 = system.vy0   # Initial y-velocity of planets, [AU].
rho0 = system.rho0                 # Atmospheric density at surface [kg/m^3].

# Data about the system.
G    = 4 * pi * pi                 # Gravitational constant in astronomical units.
numberOfPlanets = system.numberOfPlanets # Number of planets in the system.
starRadius = system.starRadius     # Radius of star, [km]
starMass = system.starMass         # Mass of the star, [solar masses].
starTemperature = system.temperature  # Surface temperature of the star, [K]
```

starMass and starRadius are floating point numbers, while numberOfPlanets is an integer. PlanetRadius and PlanetMass are arrays, such that planetMass[0] contains the mass of the first planet, planetMass[1] contains the mass of the second planet, and so on.the initial x-positions, y-positions, x-velocities and y-velocities are also arrays, such that the value of v0 in the x direction for planet 0 (the first planet) is planets_initial_x0[0], and so on. The velocity vector for planet numbered 3 will for example be (planets_initial_vx0[3], planets_initial_vy0[3]).

Available methods contained in AST1100SolarSystemViewer

As you start your assignments you will need to access some methods inside the class. To do this simply call the method you need with the necessary arguments. Below follows a list of the methods you will need and how to use them. Notice that the shape of the input arrays vary slightly between the methods, so keep calm and check if the order of your indices is correct if you get "ValueError: could not broadcast input array from shape (x) into shape (y)".

- system.orbitXml(planetPos, times) This method is for exercise 1B.7 and takes the arguments **planetPos** (position array for all your planets, in AU units. The shape has to be exactly [2, n_planets, n_times], where the first index signifies x or y coordinate, the second index corresponds to the planet number, and the third index is the time index) and **times** (your time array containing all the time values from t=t0 to t=tmax. The units are years). It returns an xml file which is stored in the same folder as you own program. This xml-file is supposed to be used with the SSView application as explained below.
- escapeVelMovie(deltaV, dt) This method takes the arguments **deltaV** (a list of velocities from zero to final velocity which you will create in assignment 1A.7) and **dt** (the time step you use in the same assignment). It also returns an xml file to be used with the MCAst application (as explained below) showing you a video from your rocket as you send it out into space.
- landingSat(satPos, times, planetChoice) This method is for exercise 1B.8 and takes the arguments **satPos** (position array for your satellite, in AU, relative to the planet. Shape is exactly [n_times, 2]), **times** (time array for satellite positions) and **planetChoice** (integer corresponding to the planet you wish to land on). It returns an xml file to use with the SSView application showing the trajectory of your satellite as you attempt to land it.
- dualStarXml(times, posPlanet, posStarm1, posStarm2) This method is for the 3-body problem in exercise 1B.7 and takes the arguments **times** (time array for your simulation), **posPlanet** (position array of the planet), **posStarm1** (positions array for star with mass 1) and **posStarm2** (position array of star with mass m2). All positions are in AU, and time in years. The shapes of each of the position arrays must be exactly [n_times, 2], with the first index being time and the second index being the x- and y-coordinates at that time It returns an xml file to use with the SSView application showing you the movement of the 3 bodies.

For example, to generate the position array to be used in orbitXml, you can use the following skeleton:

```
T = #number of years
N = #total number of time steps
times = zeros(N)
# Fill out the times-array yourself here, using uniform time steps dt
pos_computed = zeros((2, system.numberOfPlanets, N))
for t_i in xrange(N): #for each time step...
    for p_no in xrange(numberOfPlanets): #for each planet...
        time = times[t_i] #the current time step
        pos_computed[0, p_no, t_i] = #calculate the x position of planet p_no
        pos_computed[1, p_no, t_i] = #calculate the y position of planet p_no

system.orbitXml(pos_computed, times) #Will generate the xml file
```

Using other programming languages (not recommended)

If you for some reason do not wish to use python for programming your solar system, you will probably not be able to generate xmls for using the 3D app. If you really want to proceed and make your own plots, you can print the information about your star system by calling the AST1100SolarSystemViewer.pyc file from your terminal with your seed as the first argument:

```
> python AST1100SolarSystemViewer.py 12345
```

MCast

MCast is a graphics program designed to read those xml files you will create in your assignments and to visualize them for you. To download the software MCast go to the following website: http://www.irio.co.uk/MCAst. When it's downloaded you will have to unzip it and place it at a desired location. The xml files you have created are placed in the same folder as your program. These xml files will have to be moved to the data folder in the MCast folder for MCast to be able to read them.

To start MCast (this is for visualisation of your satellite launch) or ssView (this is for visualisation of your planetary orbits), it should in most cases suffice to double-click the corresponding executable inside the MCast directory. You can also start MCast or ssview from the terminal. To do this, cd to the MCast folder and type ./mcast or ./ssview. If you're using Anaconda type "mlp ssview" or "mlp mcast". Here's an example on how to open the programs:

```
> cd MCast
> ./ssview
#to access the folder an to play Ssview

> cd MCast
> ./mcast
#to access the foler and to play MCast

> cd MCast
> mlp ssview
#same but in Anaconda

> cd MCast
> mlp mcast
#same but in Anaconda
```

A pop-up window will appear asking you to choose graphics setting. If you are unsure, simply leave the suggested settings and press Play! If you have too low resolution, it might be difficult to read the text properly, in this case, try to increase the resolution.

The program will now open (hopefully, if you're having problems contact your group teachers for help). You now have to load your xml file in the option "Load solar system". If you do not find your xml here, please check that you copied it into the "data" folder in MCast, and restart the app. After selecting your xml, use the buttons " > " or " >> " to play the file at normal or fast speed. In Ssview you can also click on the planets with your mouse (or select a planet number from the drop down menu) to gain information about them and to focus on them. To zoom, use the mouse wheel, to rotate your view, use the right mouse button (if you use touchpad, two fingers to the left/right to zoom, press the button combined with left/right movement to rotate view). Note also that you can turn on/off the planet names with the button on the lower left.

NOTE for SSView: If you have problems finding/seeing your objects, select the planet (when using a satellite you can also select the satellite) from the drop-down menu so that the focus is on the selected object and then zoom in/rotate view. Often if you loose the object from view, just select it again from the menu and zoom. When looking at the satellite landing, if you select the satellite the camera is moving with the satellite (as you have chosen to have it in focus) and it might be difficult to se what is happening. In this case it might be better to choose to focus on the planet (select it from the drop-down menu or simply click directly on the planet to get it in focus) instead. Try to switch between focusing on the satellite and the planet to find the best way to view the orbit of the satellite.

Which xml file runs in which program?

- The xml file from methods orbitXml and dualStarXml runs in the program SSview. Simply open ssview as discribed above and load the file.
- The xml file from methods escapeVelMovie and landingSat run in MCast. Simply open mcast as described above and load the file.