Solar System Unit 3: Input and Output

Computer Modelling

Due: 16:00 Monday, Week 5, Semester 2

Aims

In this third unit of the project you will upgrade the input and output utilities of your code. You have three tasks:

- Make your code read a list of bodies to simulate from an input file.
- Make your code output a trajectory file in the XYZ format.
- Allow a user to configure the code from the command-line or spyder terminal instead of using fixed values of parameters.

1 Tasks

1.1 Input bodies list

In the template code the list of bodies to simulate is fixed (the Sun, Mercury, Earth, and the Moon). In this first task you will modify the code so that the bodies are read from a file instead. To start with you can fix the name of the file to be read in your code.

Two example files are available on Learn to show you the format you must read from. One, `mini_system.txt` contains exactly the same bodies as in the original template code, so if your .

The second example file, `solar_system.txt`, contains more solar system bodies. If you switch your code to use this file you should only have to change the file name used; it should need no other modifications (e.g. you should not need to also tell your code how many bodies to expect in the file).

1.2 Output XYZ file

In this second task you will add code to your file to save the complete trajectories of your bodies list in the XYZ format described in the background document. To start with you can fix the name of the XYZ file to be written in your code.

You can either do this during the integration loop, writing output at each step, or wait until after the integration loop and write it all at once. If you do the first case then the `__str__` method you wrote in the particle class should print out the right line.

If you have done this correctly then you can use the `plot_xyz.py` program like this from spyder, assuming that you called your xyz file `filename.xyz`:

```
%run plot_xyz  filename.xyz  -o animation.gif
```

There are other options to `plot_xyz.py` that you can use to help check things are working; you can read about them by doing:

```
%run plot_xyz  --help
```

1.3 Command-line interface

See the notes in the background information on command line interfaces. You should now use this feature to allow the user to set:

- The number of steps.
- The time step size.
- The name of the input bodies file.
- The name of the output XYZ file.
- (optionally) Names for any other output files like plots.

There are various ways you can do this. You could set them all on the command line, or have the user specify a single parameter file that contains all the other options. If you are feeling ambitious you could use the built-in python `argparse` module to allow default parameter values. See the documentation on the python website.

Generally, convenient approaches will get good marks, and fiddly or inconvenient ones will not. You also get marks for ensuring that if the users gives no input or the wrong input then they get some help printed out.

2 Submission

2.1 Submitting

Submit a zip file through Learn as explained in the background briefing. Successful submissions are emailed a receipt.

Always check your code runs one final time before submitting.

2.1.1 What to submit

Ensure you submit at least:

- the main python file.
- the particle3D file.
- any other python files needed to run the code - e.g. a basic functions file if you

kept it separate.

- A small readme text file explaining how the user should run the code.

2.2 Marking Scheme

This assignment counts for 10% of your total course mark, and is marked out of 20.

1. Input files are read correctly **5**
2. XYZ file is output in the correct format **5**
3. Command line interface is correct and efficient **5**
4. Code quality, layout, and commenting/docstrings **5**

Total: 20 marks.