OBERON - OBliquity and Energy balance Run On Nbody systems
Version 1.1

Duncan Forgan (`github/dh4gan`)

September 28, 2018

NOTE: OBERON is developed for use on Unix systems (e.g. Linux or Mac). I can't guarantee support for Windows, and your mileage may vary.

Contents

1 Getting OBERON

OBERON is hosted on github at `https://github.com/dh4gan/oberon`. You can either directly download from the link or use `git`. Open a terminal and at the command line, type:

```
> git clone https://github.com/dh4gan/oberon.git
```

2 Compiling OBERON

The source code is in the `/src` folder, which contains a `Makefile`. You will require g++ (and gmake or cmake) to compile the code. With the appropriate software installed, type 'make' to compile the executable `./oberon`.

3 Running OBERON

With the code compiled, type

```
> ./oberon <parameterfile>
```

or

```
>./oberon
```

Which will then prompt you to enter a parameter file name.

4 Code Design

This code is object-oriented, with three base classes:

- The base class `Body` encapsulates the properties of each body in the simulation. The base class stores positions, velocities, masses and radii, and including methods for calculating gravitational forces and computing orbits. The derived `Star`, `Planet` and `World` classes contain methods for calculating luminosity and climate/spin evolution.
- The `System` class takes a vector of `Body` pointers, and controls method calls for the evolution of the collection of `Body` objects under gravity, and `World` objects' climate evolution.
- The `parFile` class handles input data to the simulation, which is read from a parameter file.

When the code is run, any `World` object in the simulation will have its climate computed under the assumption that it is Earthlike, with each World receiving its own set of output files.

5 Input Options

OBERON requires a single input parameter file which contains the essential information required to run the simulation. The parameter file system is quite flexible - the ordering of the options in the file is relatively unimportant, provided that all the global options are at the top of the file before individual body data.

Each variable is read in by checking the first string on each line. Please ensure that the keywords as given below are the first strings on each line, with no whitespace at the beginning, otherwise they will be ignored.

The read in system is robust to '–' and '#' symbols, so do feel free to use these as spacers between data points, or indeed to "comment out" a row and replace it with another input.

5.1 Global Options

- `ParType 'Positional/Orbital'` The format of data for entry (either Cartesian co-ordinates for position and velocity or orbital elements)
- `NBodyOutput (string)` The filename for the N Body data file
- `SnapshotTime (double)` The time interval between data dumps and snapshots
- `NGridPoints: (integer)` The number of grid points used by the latitudinal energy balance model
- `MaximumTime: (double)` The maximum runtime of the simulation
- `SystemName: (string)` A name for the simulation
- `Number Bodies: (integer)` The number of bodies in the system
- `Restart: 'T/F'` is the simulation a restart? (Default: False)
- `FullOutput 'T/F'` True: Output full snapshots as well as log files with surface averaged values for each body, False: log files only (Default: False)
- `TidalHeating 'T/F'` Compute the tidal heating of the Worlds by the object they orbit? (Default:False)
- `PlanetaryIllumination 'T/F'` Are 'Planet' objects also luminosity sources? (Default: False)
- `ObliquityEvolution 'T/F'` Allow the spin parameters of 'World' objects to change? (Default: False)
- `CarbonateSilicateCycle 'T/F'` Model the effects of the carbonate-silicate cycle? (Default: False)

5.2 Body Options (must be specified for each body)

- `BodyName: (string)` Name for the `Body` object
- `BodyType: 'Star/Planet/World'`
- `Mass: (double)` Body mass in solar masses
- `Radius: (double)` Body radius in solar radii

For Positional Files:

- `Position: (double) (double) (double)` Cartesian position vector (astronomical units, AU)
- `Velocity: (double) (double) (double)` Cartesian velocity vector (2π *AU/year)

For Orbital Files:

- `SemiMajorAxis: (double)` Semimajor Axis (AU)
- `Eccentricity: (double)` Eccentricity
- `Inclination: (double)` Inclination (radians)
- `LongAscend: (double)` Longitude of the Ascending Node (radians)
- `Periapsis: (double)` Argument of Periapsis (radians)
- `MeanAnomaly: (double)` Mean Anomaly (radians)
- `OrbitCentre: (integer)` Where is the initial orbit focus? -1 = (0,0,0), 0=system centre of mass, 1,2,3... = Body 1,2,3)

5.2.1 Planet Options

- `Albedo` - Albedo of the Planet (for Planetary Illumination calculations)

5.2.2 Star Options

- `Luminosity (double)` Bolometric Luminosity (solar luminosity)
- `SpectralType ('F/G/K/M')` Stellar spectral type (for Carbonate Silicate Cycle)

5.2.3 World Options

- `RotationPeriod (double)` World rotation period in days
- `Obliquity (double)` initial obliquity in degrees
- `WinterSolstice (double)` orbital longitude of the winter solstice (degrees)
- `OceanFraction (double,` $[0.0 \rightarrow 1.0]$`)` Fraction of the world's surface that is ocean
- `InitialTemperature (double)` Initial surface temperature of the world (at all latitudes)
- `IceMeltingOn 'T/F'` Is latent heat of melting for ice accounted for in climate calculation? True or False (Default: False)
- `OutgassingRate` Rate of CO2 outgassing (in Earth units, for Carbonate Silicate Cycle)
- `SeaFloorWeatheringRate` The seafloor weathering rate at Earth's PCO2, relative to Earth (Carbonate Silicate Cycle)
- `BetaCO2` The relationship between outgassing and PCO2 (power law index, Carbonate Silicate Cycle)
- `BetaCO2` The relationship between seafloor weathering and PCO2 (power law index, Carbonate Silicate Cycle)

6 Outputs

- `<WorldName>.<number>` - a snapshot of `<WorldName>`'s latitudinal climate properties
- `<WorldName>.log` - a log file for `<WorldName>` tracking globally averaged climate properties and position/orbital properties
- `<WorldName>.lat` - a file containing the temperature as a function of latitude at each time for `<WorldName>`

An N Body file is also produced (name specified by the user), which is more suited to plotting the entire simulation's evolution (Star and Planet objects included).

Python 2.7 scripts for plotting these datafiles can be found in 'plot/'