

ORSA

Orbit Reconstruction, Simulation and Analysis

<http://orsa.sourceforge.net>

Pasquale Tricarico
Physics Department, Padova University, Italy
tricarico@pd.infn.it

Introduction

Orbit Reconstruction, Simulation, and Analysis (ORSA) is a framework for Celestial Mechanics investigations. The main goals of the project are the implementation of state of the art orbit integration algorithms, with concerns on accuracy and performance, and the development of analysis tools. Command line and graphical interfaces are also provided.

Overview

ORSA is a *Free Software* project developed under *Linux* and written in *C++*. The main purposes of the ORSA project are:

- create a *common infrastructure* among the existing Celestial Mechanics programs;
- implement all the *state of the art* orbit integration algorithms, with concerns on **accuracy**, **performance** and interaction options;
- develop command line and graphical applications;
- support for *High Throughput Computing*, *LSF* and *GRID*;
- use an object oriented programming language (*C++*), *CVS* and auto-configuration tools to achieve a good design, rapid development and maintainability;

- keep the ORSA project under **GPL**.

The main components of the project are:

- **liborsa** the main numerical library, where all the base classes like *Units*, *Vector*, *Body*, *Interaction*, *Orbit*, *Integrator*, *Frame*, *Evolution* and *Universe* are implemented;
- **libxorsa** the graphical library, which implements all the components of the Graphical User Interface;
- **xorsa** the main graphical application, which provides most of the functionalities of both **liborsa** and **libxorsa**.

The project is developed using freely available libraries, like *FFTW*, *Qt*, *readline* and the *GNU Scientific Library*.

Basic concepts

We briefly describe here the concepts ORSA is based on. Most of them are represented as *C++* classes, but this is just an implementation detail for programming-minded readers.

The **Universe** is the first class created by a program using ORSA, and keeps all the informations regarding interactions, units, bodies, evolutions and reference systems. Like other classes, the role of **Universe** is very intuitive and can be guessed by the class name. There are two type of universes: *Real* and *Simulated*. The *Real Universe* is of choice for investigations in the Solar System: the planets and asteroids can be easily imported from ephemeris files, and the time is in years, months and days. The *Simulated Universe* is the most general one, and allows the user to set all the bodies from scratch; in this case the time starts from zero and cannot be converted into a date.

The **Vector** class represents a 3D vector, and implements all the basic operations. The **Body** class contains all the properties of a body, like mass, name, position and velocity. The **Units** class provides a coherent implementation of physical quantities and constants. The base units used in the internal computations are defined at the creation of the **Universe**.

The **Date** class implements a reliable and standards-compliant representation of the time, with support for *Timescales* (*UTC*, *UT*, *UT1*, *TAI*, *TDT*, *ET*, *TT*, *GPS*). The **Orbit** class provides a representation of an orbit, and implements a set of methods for orbit determination tasks.

liborsa

The **liborsa** library is the basic library for all the ORSA applications. It implements all the basic physical classes described before, and also implements all the file import/export methods, the configurations-related classes, the implementation of the **Analysis** classes like the *Frequency Map Analysis*, *Lyapunov* and *Mean Motion Resonances*, and many other tasks.

This library uses other numerical libraries to perform part of the computations, like the *FFTW* library of the Fast Fourier Transforms, the *GNU Scientific Library* for minimization problems, and makes use of the *C++* Standard Library for containers and algorithms. The **liborsa** library provides support for *High Throughput Computing* (like *Condor*), *LSF* and *GRID*.

libxorsa

This is the graphical library, used only by the graphical applications. It implements more than fifty different widgets, like object browsers, editors and generators, plotting widgets, units and date conversion tools, 3D OpenGL viewers, and many others. This library makes a heavy use of the **liborsa** library, where all the data structures and all the basic methods are defined.

The **libxorsa** library is based on the *Trolltech's Qt* library, which allows, among the other benefits, to write multi-thread applications. This means that it is possible to perform a numerical integration while observing the same integration in a 3D viewer, or start more than one integration from the same application.

xorsa

This is the main application of the ORSA framework. With **xorsa** it is possible to perform all the operations allowed by the ORSA framework, as the base libraries and **xorsa** are developed concurrently.

Tutorials

Together with the libraries and the main application, we have developed some tutorials, which allow to learn quickly all the basic functionalities of the framework. They are usually very short programs focusing on a particular task, like basic system numerical integration, file import, analysis tasks and others.

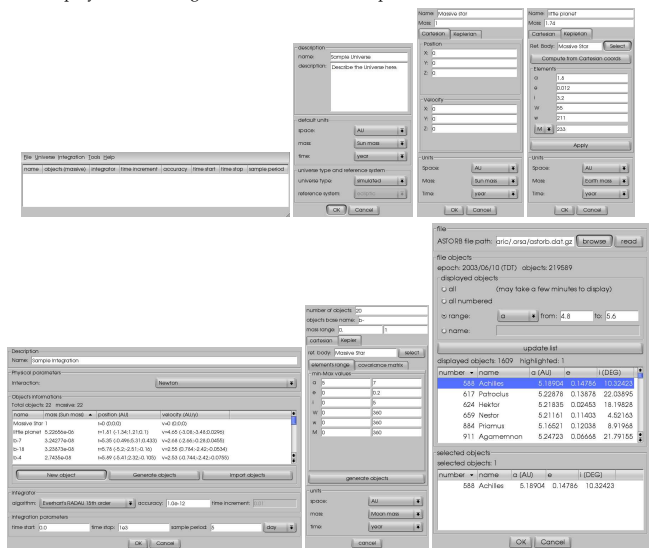
Sample session

We describe here an **xorsa** sample session.

- When the **xorsa** program is started, the main window appears. The main table, now empty, contains informations about all the integrations made, like the integration name, objects, integration algorithm, start and stop time and others.
- The first step consists in the creation of a new **Universe**. Each **Universe** has a name and a description, and when a **Universe** is created all the physical quantities for internal computations are set. Another important option is the selection of the type of **Universe**: *Simulated* for a system as generic as possible, *Real* for a system as close as possible to the Solar System. For this sample session we select *simulated*.
- Once the **Universe** is created, we can start a new integration. For each integration we can set name, an interaction, a set of bodies, and select the integration algorithm, time ranges and sampling period.
- When the integration is finished, but also while the integration is being performed, it is possible to plot the osculating elements of each object, 2D projections and also view the whole system in a 3D OpenGL viewer, with animation and zoom capabilities, rotating and intertial reference system, and other options.
- It is also possible to analyze the integration applying the *Frequency Map Analysis*.
- The whole session can be saved as a single file containing the whole **Universe**. This file is automatically compressed for a minimal disk usage. Once saved, the session can be later restored for further investigations, or sent to a colleague.

Gallery

Here we display some of widgets used in the **xorsa** sample session.



Conclusions

ORSA is an evolving project whose goal is to provide a useful frame for dynamical studies. ORSA is *Free Software*: feel free to use and modify it. ORSA can be downloaded from <http://orsa.sourceforge.net>.