An Introduction to ORSA

Pasquale Tricarico

Physics Department, Washington State University

Overview

- ORSA by samples: the *graphical application*
- ORSA by samples: the *numerical library*
- ORSA main features
- *What can I do with ORSA?*
- Incoming features: *where is ORSA going?*

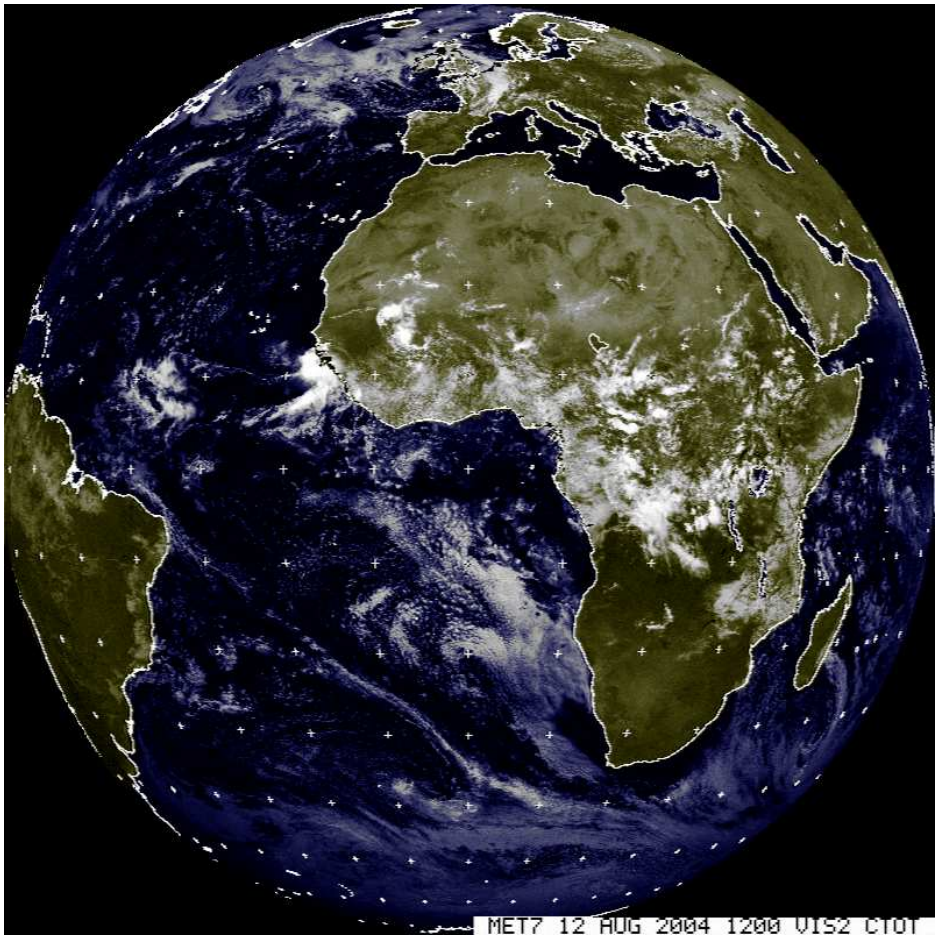
ORSA by samples: the *graphical application*

We start showing some pictures, representing the ORSA graphical application in action. The ORSA graphical application is called *xorsa* under Unix/Linux, *macorsa* under Mac OS X, and *winorsa* under Windows.

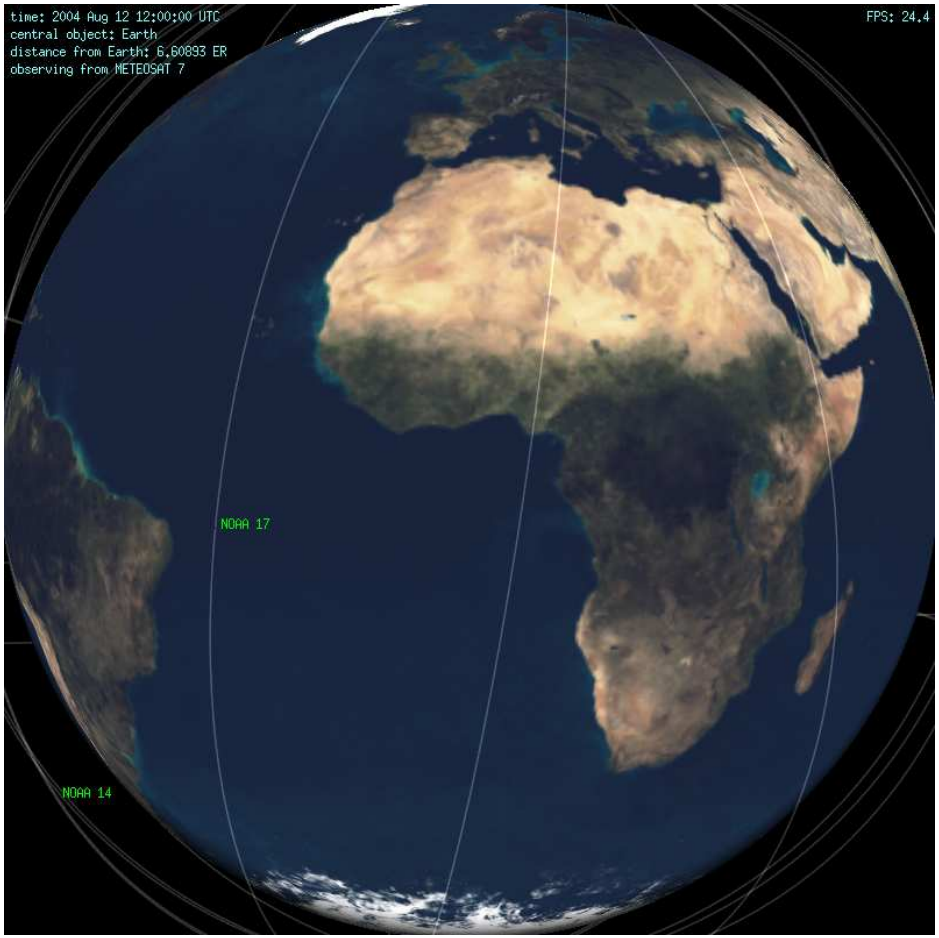
With the graphical application it is possible to **create** a physical system, **edit** bodies, perform and **visualize** the numerical integration, and use other **analysis tools**.

The working session can be saved on a single file, and after can be completely restored by just loading the file.

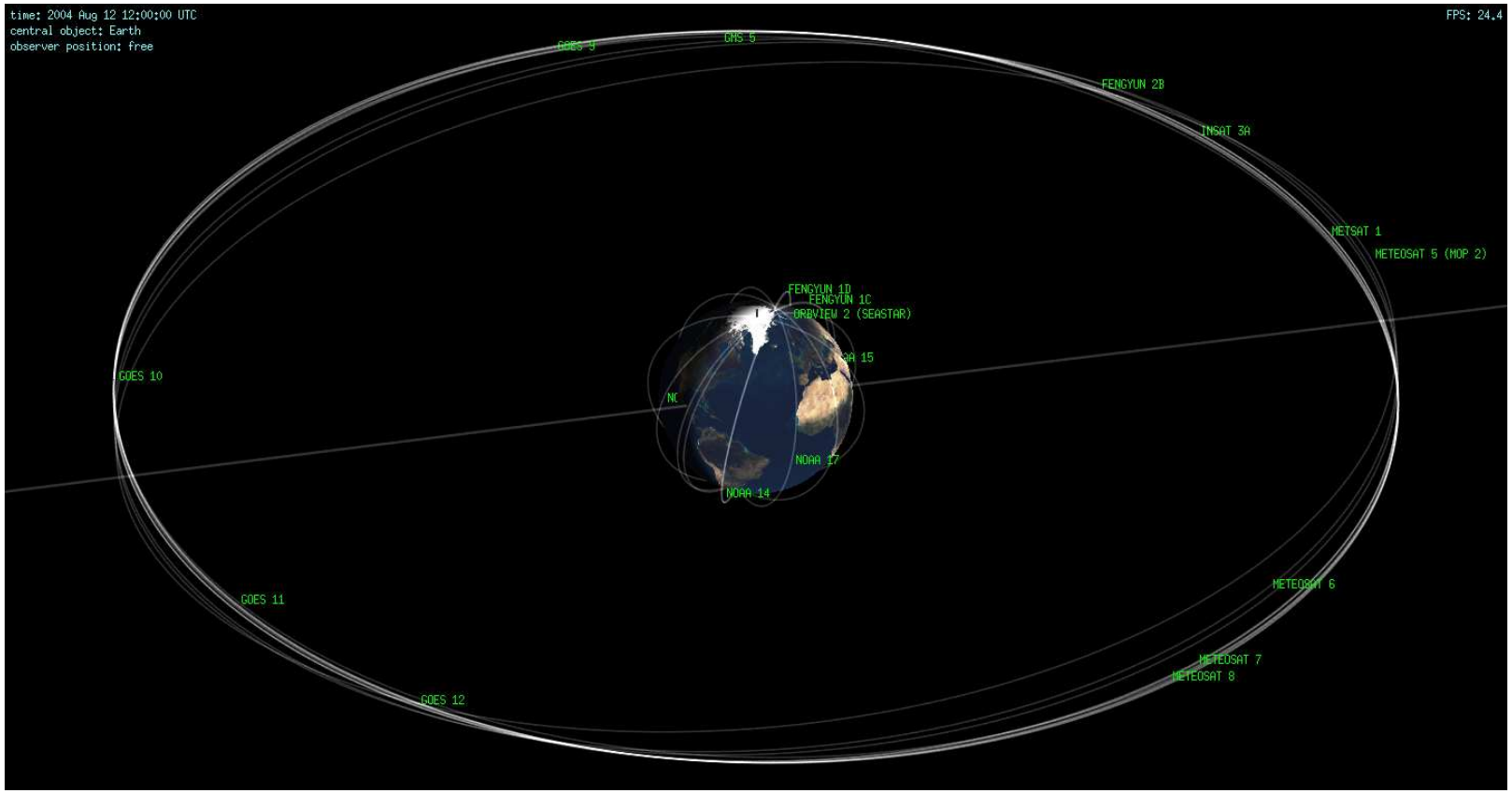
A Meteosat Image...



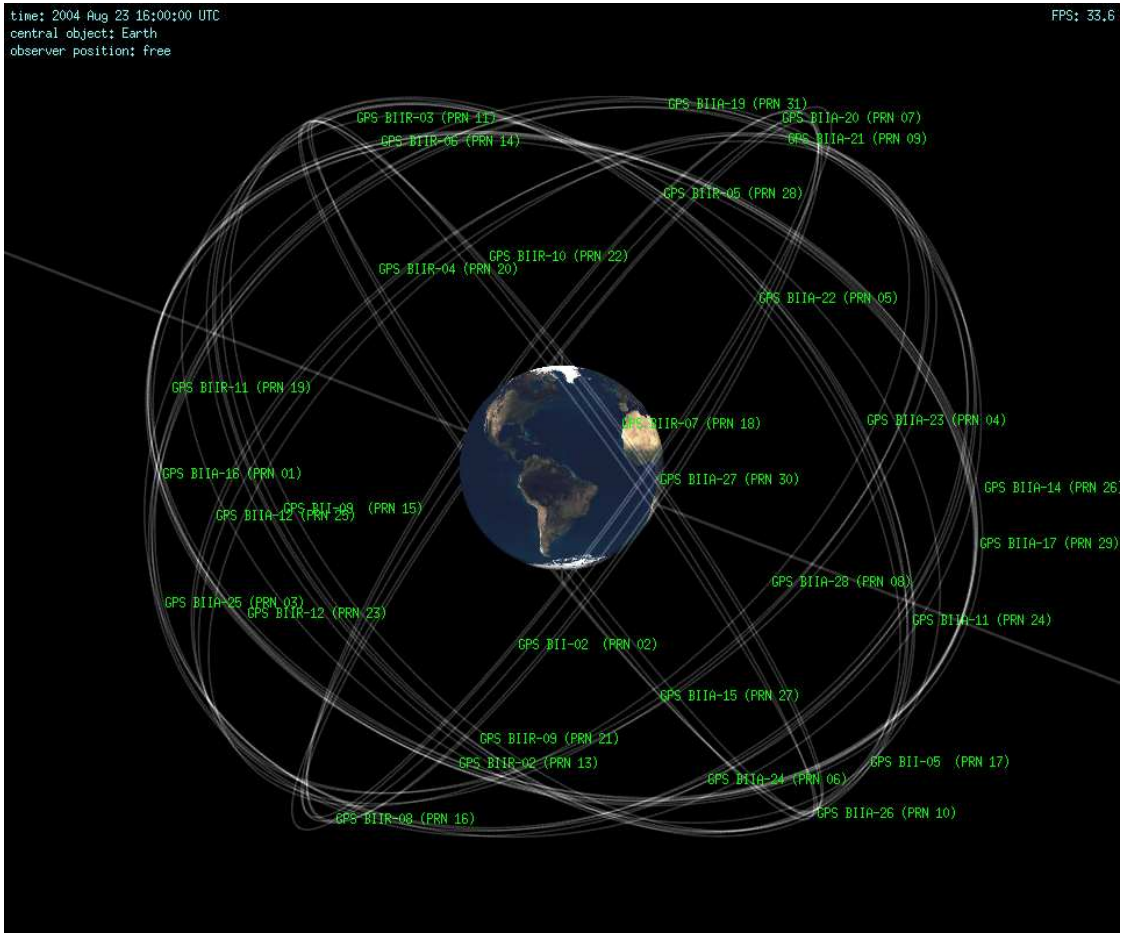
. . . and the ORSA Image *(with some low-orbit satellites)*



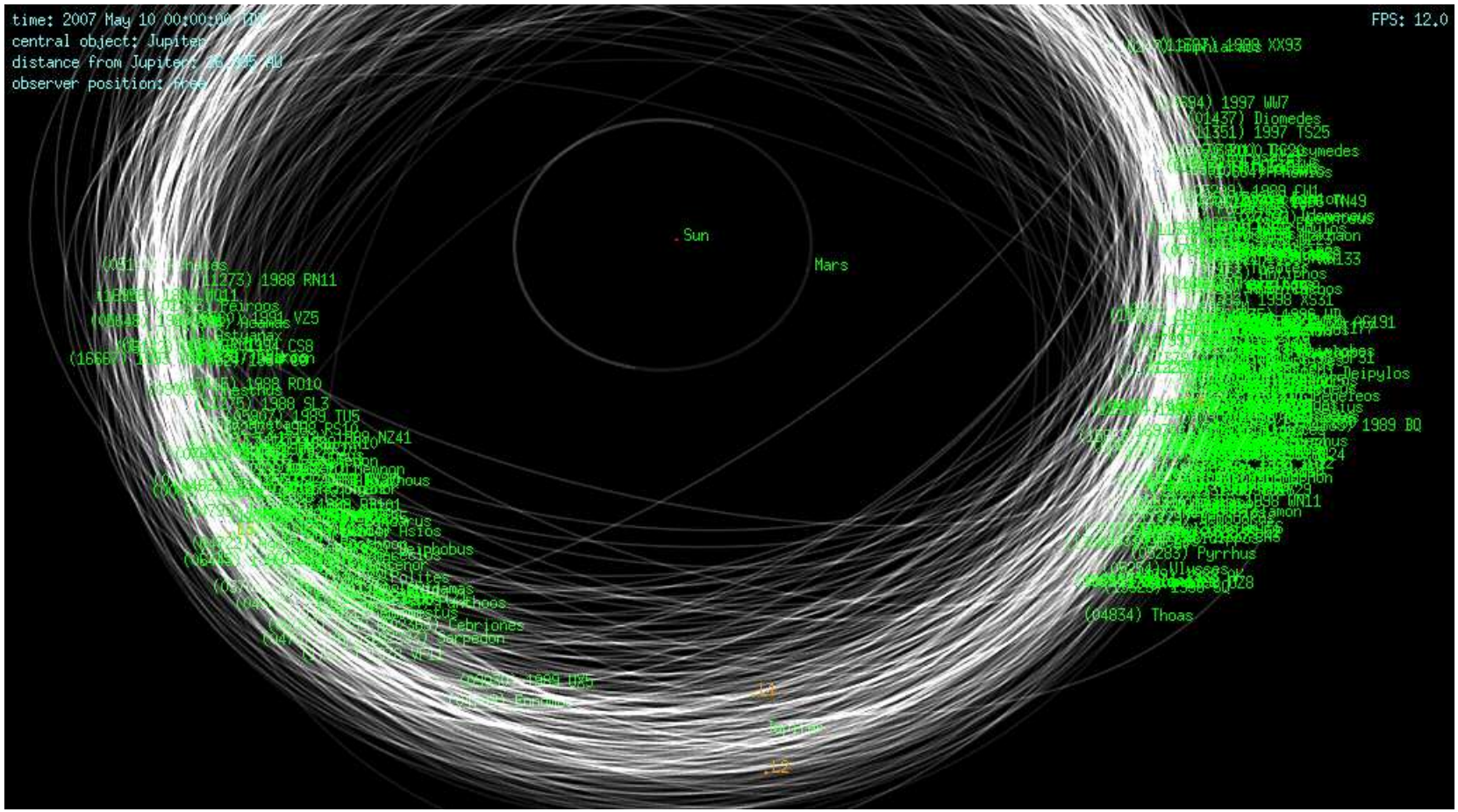
More Weather Satellites



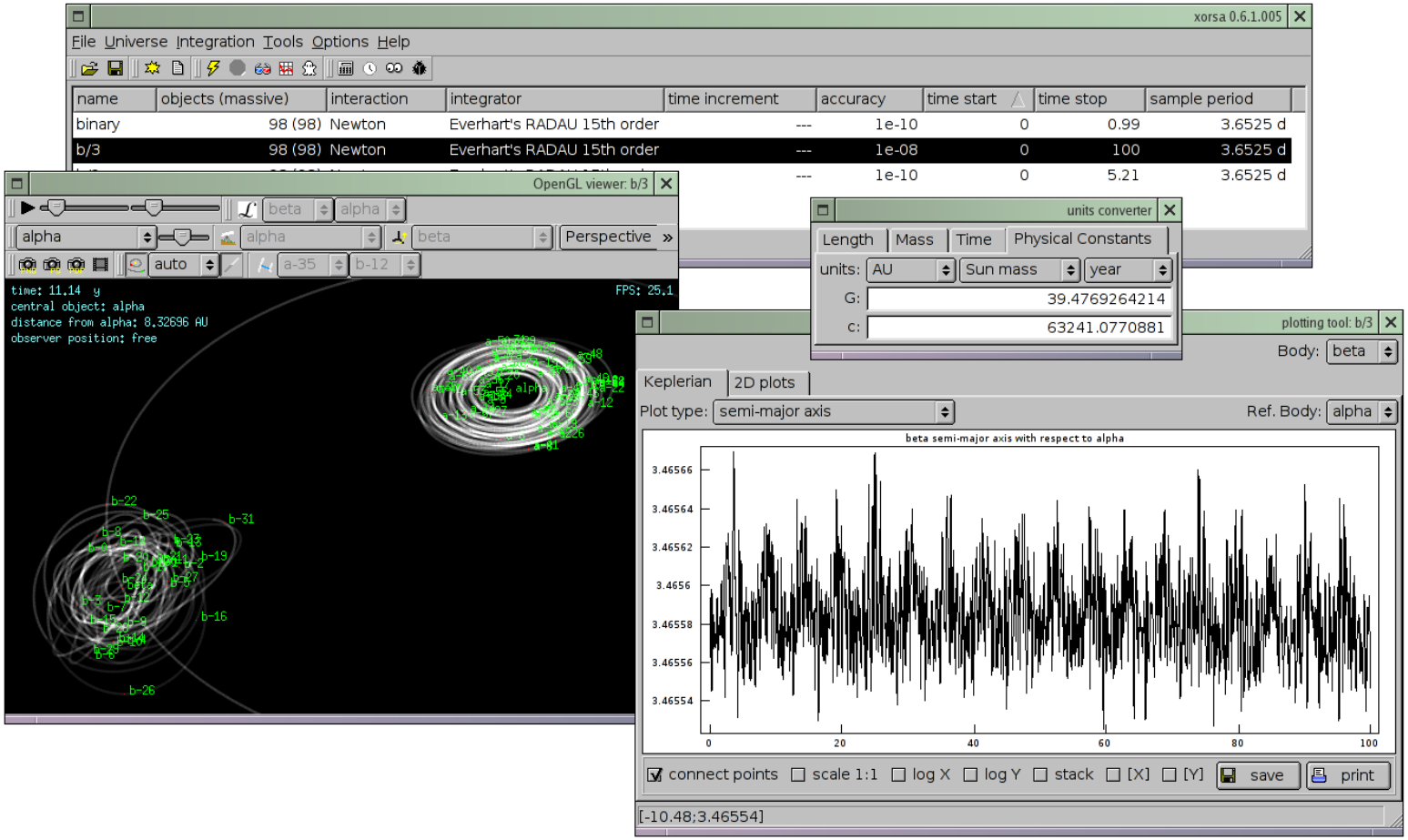
The GPS Constellation



Jupiter Trojans



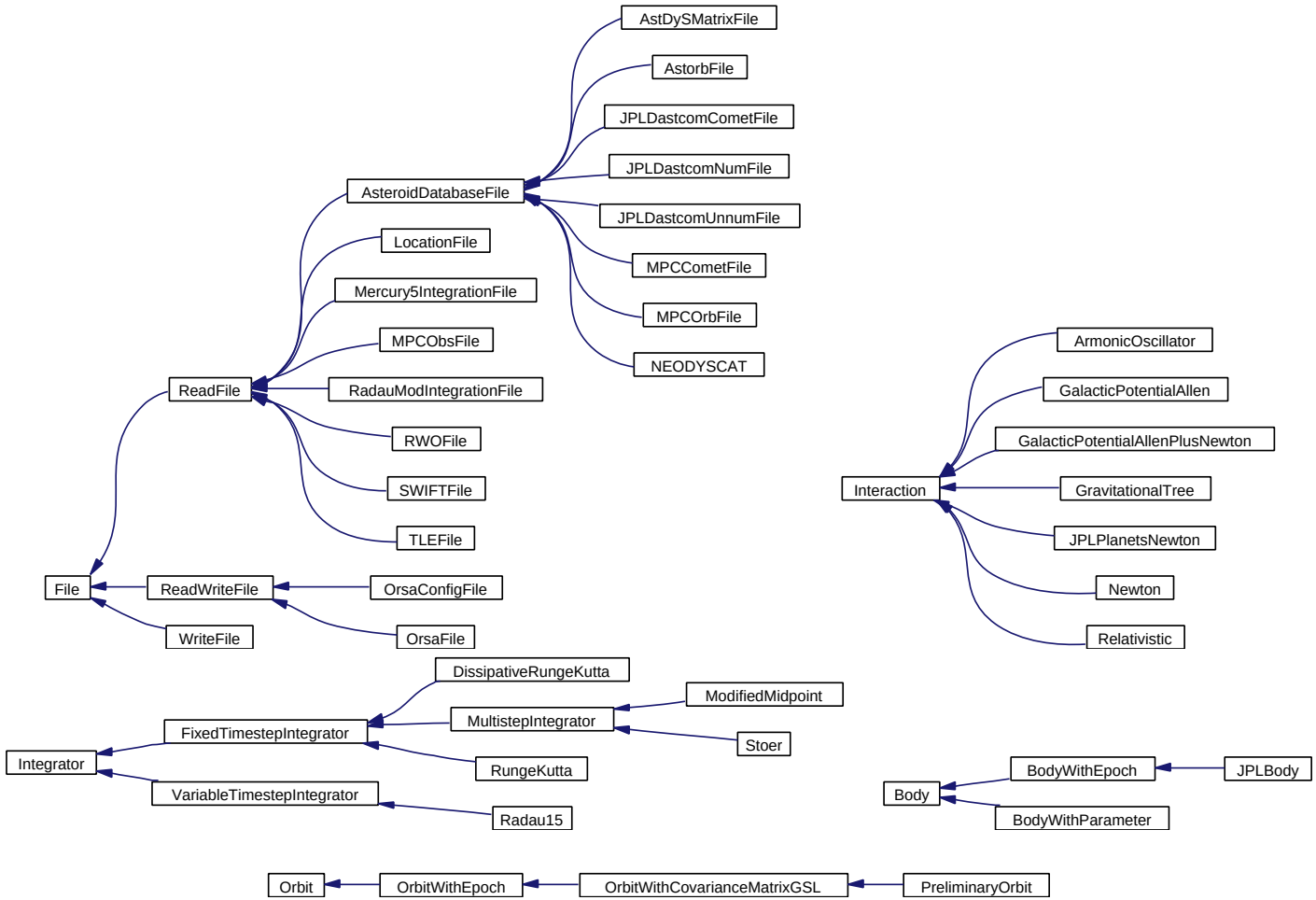
Sample ORSA session: simulation of a binary system



ORSA by samples: the *numerical library*

- The **core** of ORSA is the *liborsa* numerical library, written using the C++ programming language; any external application can link to this library;
- This library defines an API and all the fundamental objects: *Units*, *Vector*, *Body*, *Interaction*, *Orbit*, *Integrator*, *Evolution*, *Universe*;
- This library can use other numerical libraries like GSL, FFTW and GINAC for more numerical algorithms;
- Initial support for parallel computing, using the MPI interface.

Some ORSA numerical C++ classes and their hierarchy



Sample C++ code: a numerical integration

```
#include <orsa_universe.h>
#include <orsa_file.h>

using namespace orsa;

int main() {
  // Universe constructor: units (length, mass, time), type
  new Universe(AU, MSUN, YEAR, Simulated);

  // Body constructor: name, mass, position, velocity
  Body star("star", FromUnits(1.4, MSUN), Vector(0,0,0), Vector(0,0,0));
  Body planet("planet", FromUnits(0.8, MEARTH), Vector(1,0,0), Vector(0,6.2,0.1));
  Frame f(0.0); f.push_back(star); f.push_back(planet); // the system at t=0.0

  Interaction * itr = new Newton;
  Integrator * itg = new Radau15; itg->accuracy = 1.0e-9; itg->timestep = 0.01;
  Evolution * evol = new Evolution; evol->integrator = itg; evol->interaction = itr;
  evol->sample_period = 0.1; evol->push_back(f); universe->push_back(evol);
  evol->Integrate(100.0); // perform numerical integration, 100.0 years
  OrsaFile of; of.SetFileName("sample.of"); of.Write(); of.Close(); // save on file
  return 0;
}
```

ORSA Current Main Features

- ORSA can simulate either the **Solar System**, providing automatically a coherent environment and all the needed physical constants, or any other planetary system, i.e. an **Extra-Solar System**;
- Numerical library: object-oriented accuracy-minded C++ library, continuously improving API allows the development of ORSA-based third party applications, General Relativistic corrections to the Newton interaction, different numerical integrators;
- Graphical library: fully featured graphical application, based on the Qt library, with 2D and 3D (OpenGL) rendering windows, object editing tools in Cartesian and Keplerian coordinates, numerical integration management, *all-in-one-file* save and restore of simulations, export of simulations to ascii tables.

ORSA Current Main Features *(continued)*

- Objects import: Solar System planets position from JPL ephemeris files; asteroids and comets from MPC, JPL, Lowell, AstDyS, and NEODyS database files; artificial satellites from TLE files;
- Portability: library and applications available for **Linux**, **Mac OS X** and **Windows**;
- Source code freely available, GPL license.

What can be done using ORSA?

- **Solar System** dynamics: any object in the Solar System can be simulated dynamically, using accurate numerical algorithms and the best available initial conditions for all the known bodies;
- Simulated systems: a generic system can be simulated, using any units system or interaction, and without constraints on the number of massive and massless objects;
- Education: ORSA is a scientific grade tool, but can be used for demonstrations and also as a teaching tool to effectively experiment the gravitational dynamics;
- 3*rd*-party applications can use the ORSA numerical library to perform numerical integrations;
- ORSA is free software so can be freely modified and enhanced to fit any needs, and the user's enhancements can be contributed back to the main ORSA distribution.

ORSA Incoming Features

The list of features we are working on includes:

- orbit determination from astrometic observations, with determination of the covariance matrix and impact probability;
- natural satellites: organize published data into a database;
- more numerical integration algorithms, some symplectic;
- space probes: thrust scheduling, mission analysis and optimization;
- planetary accretion, objects with variable mass, inelastic impacts with possible fragmentation;
- ***Something is still missing? Let us know!*** Feature requests are always very welcome, and drive the ORSA development.

Contact us!

We want to hear from you! Feature requests are particularly important to shape the future releases of ORSA, matching the developers' work with the users needs.

ORSA website: http://orsa.sourceforge.net

ORSA author and maintainer: Pasquale Tricarico <tricaric@wsu.edu>

This introduction is based on the ORSA public version 0.6.1 and on the development version 0.6.1.x, and is available for download on the ORSA documents page: http://orsa.sourceforge.net/docs.html