ABSTRACT

We explain simulation tools in the JASMINE project (JASMINE simulator). The JASMINE project stands at the stage where its basic design will be determined in a few years. Then it is very important to simulate the data stream generated by astromanetic fields at JASMINE in order to support investigations of error budgets, sampling strategies, data compression, data analysis, scientific performances, etc. Of course, component simulations are needed, but total end-to-end simulations which include all components from observation target to satellite system or scientific outputs are also very important. We find that new software technologies, such as Object Oriented (OO) methodologies are ideal tools for the simulation system of JASMINE (the JASMINE simulator).

The simulation system should include all objects in JASMINE such as observation techniques, models of instruments and a design of satellite bus system, orbit, data transfer, data analysis, etc. in order to resolve all issues which can be expected beforehand and make it easy to cope with some unexpected problems which might occur during the mission of JASMINE. So, the JASMINE Simulator is designed as handling events such as photons from astromanctic objects, control signals for devices, disturbances for satellite attitude, by instruments such as mirrors and detectors, successively. The simulator is also applied to the technical demonstration "Nano-JASMINE".

Simulator Framework

Mission components → apply Observer Pattern. Telescope, Detector, OBC and etc. handle such events. These Event Handlers are implemented as both Observer and Observable. Event s (Photon, Control signal, Telescope, Detector, OBC, etc.) handle such events. These Event Handlers are implemented as both Observer and Observable.

Satellite components compute attitude in a complicated way. Mission components should know only interfaces of GetStatus class.

Observer Pattern

Observer, Observable, etc. are inherited from JSEventHandler class, which is a subclass of Façade class. GetEvent class is a single object class.

Mission Component(Observer)

Get Event

State changed

Catch event

Mission Component(Observer)

Core Framework

Collaboration diagram of mission component

UML collaboration diagram of the core framework. JASMINE components are placed within the chain. Events which the simulator treats are handed from the components at upstream to that at downstream.

Class diagram of mission component

UML class diagram of the framework. All JASMINE components are inherited from JSEventHandler class, which is a subclass of Observer and Observable.

Galaxy model

Wainscoat et al.(92), Cohen(94)

Solar system

Requirement of data compression

High compression rates, Enable with satellite CPU(80186 level @ 2006)

Solution: Combination of Karuhnen-Loeve transformation and Golomb-Rice code. About 60% compression rate is achieved. Above are first three principal components.

Observation method

Strategy of the methods are encapsulated into the class.

Single telescope with overlapping plate

Double telescope with great circle scan

etc.

System component simulation

Use Façade pattern. Also façade class is implemented as a singleton.

The requirement of attitude stability is very high. Use mission instrument as ‘star tracker’. For achieving sub pixel accuracy of position detection, Karuhnen-Loeve transformation is also effective.

Focal plane simulation

CCD model=composite of pixels

Exposure are simulated in photon level.

PSF shape, attitude disturbance and attitude estimation error, error caused by CTI≠0, CR events etc. are simulated.

Satellite altitude information in sub pixel accuracy are fed back to the satellite attitude control system.

Reference to terms with specific meaning within a object-oriented programming context:

class: encapsulates common behavior of a group of objects;
attribute: data member of a class;
method(s): function(s) which may be performed on instances of a class;
object: instance of a class;
abstract class: a class from which no instances may be created;
inheritance: a way to form new classes or objects using predefined objects or classes where new ones simply take over old ones’ implementations and characteristics.