Glast Code Review: 24 April 2002

An Introduction to G4Generator

R.Giannitrapani

Contents

The work on G4Generator has been carried on by many people with direct contributions (code) and indirect ones (comments, critics, bugs finding etc.); among them (in no particular order)

Toby, Francesco, Tracy, Joanne, Alessandro, Leon, Marco, Sasha, Richard, Heather

These slides have been typeset in ConTEXt.

Introduction

- Lets start with a simple fact: the package `G4Generator` is not a finished, closed, we-are-all-happy-of-it piece of software
- It will become that (we hope) in the ***very near*** future
- To some extent it is already quite **stable** and **usable**
- Netherveless it is still in an evolving phase, due principally to the nature of the problem, i.e. merging together two evolving piece of software (a toolkit and a framework) that have been designed in two completely different ways: ***Geant4*** and **Gaudi**
- First of all I want to briefly introduce these main actors, their interaction problems and how we solved them
- Then I'll show how it works in some detail

Geant4

- It is a toolkit
  - a collection of abstract classes that the user has to concretely implement
  - a manager mechanism to register these classes
  - a central engine hidden from the user
- It has a lot of features and functionalities, more than a normal Monte Carlo library (graphics, persistency, data structure, geometry databasing etc etc)
- It is designed as a stand alone application
- It is quite well supported (two main releases in a year)
- Tune group has shown how ***Geant4*** can be used as a standalone application for ***GLAST*** simulations (Balloon)
- They have also contributed to physics validation (together with Francesco and Alessandro), showing that ***Geant4*** is reliable at least for what concern em processes

- The user must provide a certain number of classes derived by ***Geant4*** abstract ones; at least
  - A concrete implementation of a `G4VUserDetectorConstruction` class for geometry construction; in particular the implementation of the virtual method `G4VPhysicalVolume* Construct()` that must return a pointer to the world volume, i.e. the root of volumes hierarchy (note that also materials must be defined in this class)
  - A concrete implementation of a `G4VPhysicsList` class for physics processes management; in particular the implementation of the virtual methods `void ConstructParticle()`, `void ConstructProcess()` and `void SetCuts()` (as we will see we adopted a more modular implementation)
  - A concrete implementation of a `G4VUserPrimaryGeneratorAction` class for primary particles generation; in particular the implementation of the virtual method `void GeneratePrimaries (G4Event *)` that generates the primary particle in the simulation
- These classes are registered with a manager (`G4RunManager`) that runs the main event loop

Gaudi

- ⋆ It is a framework
- ⋆ Algorithms work on data objects in data stores, both in and out
- ⋆ Services provide functionalities useful to algorithms through abstract interfaces
- ⋆ The main event loop is controlled by the framework

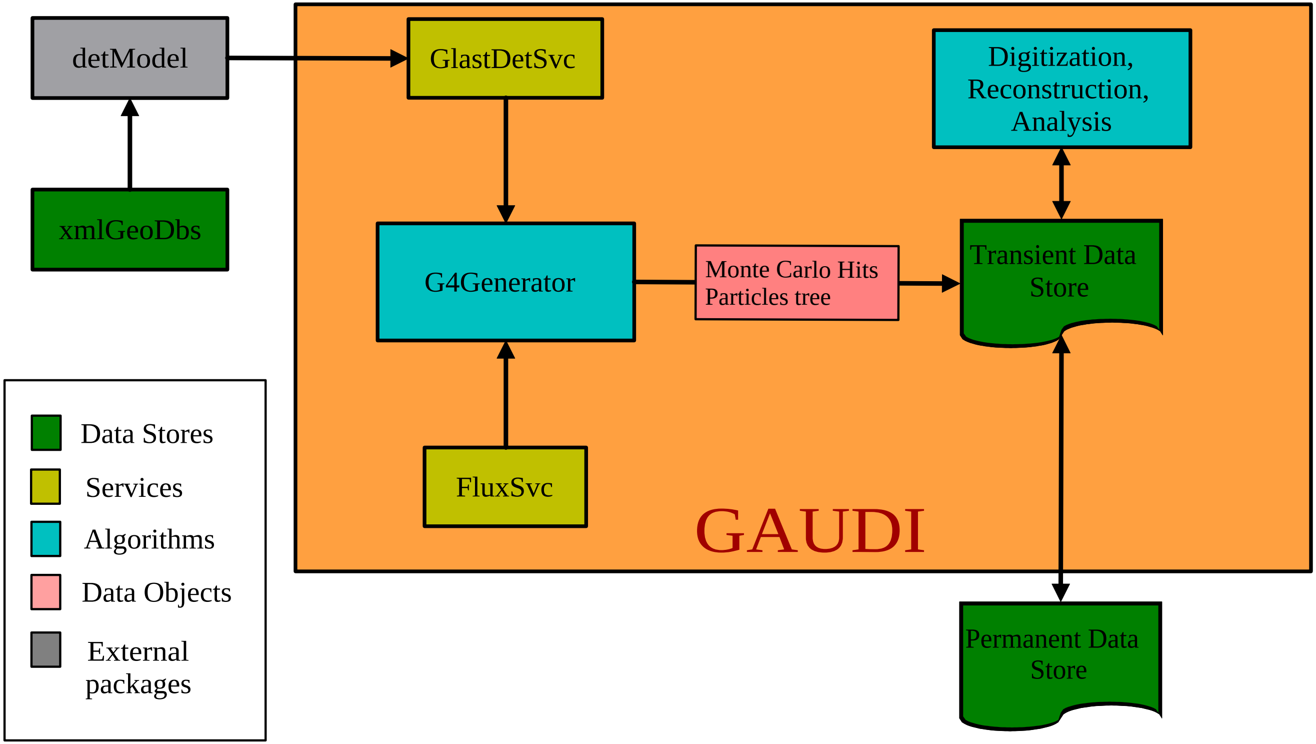
Early problems and solution

- Both ***Geant4*** and Gaudi want to manage the event loop
- ***Geant4*** provides a lot of functionalities that are already provided by Gaudi Services (like graphics, persistency, event data structure, geometry databasing, digitization); we don't need them
- ***Geant4*** is composed of many classes and it can be hard to find out what we need
- The main central engine of ***Geant4*** is managed by the `G4RunManager` that is not an abstract class of the toolkit

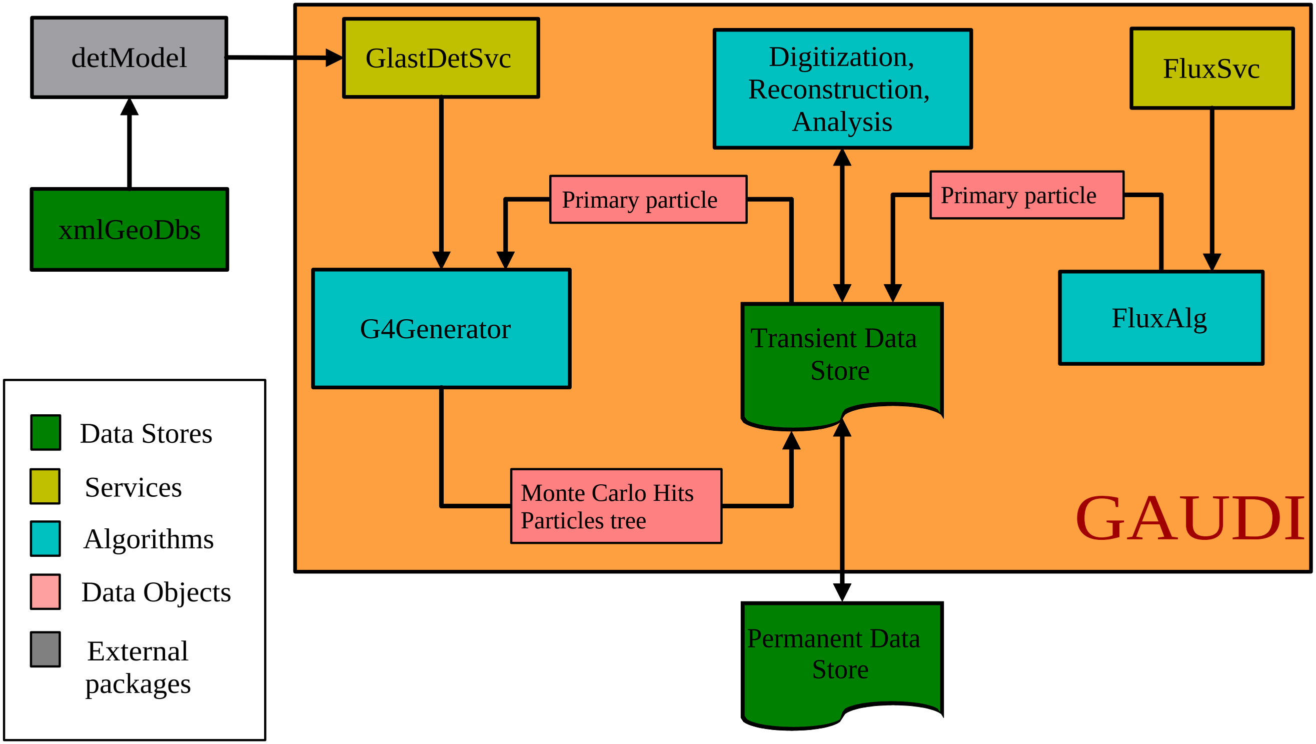
GLAST recipe:

- A local installed ***Geant4*** distribution (we are using v.3.2 binary distribution in the external libraries directory of ***GLAST***)
- A CMT package that wraps the installation of ***Geant4*** for makefiles generation and dependencies handling (it is called `Geant4`)
- A GAUDI algorithm, that is `G4Generator`, that uses ***Geant4*** to produce hits in the detectors and store them in the TDS, followed by other algorithms to produce digits and reconstructed quantities
- Provide a ***GLAST*** specific RunManager for that algorithm such that:
  - It uses only a subset of ***Geant4*** functionalities, the ones we need really
  - It leaves control of the event loop to Gaudi
- Can be dangerous for future ***Geant4*** compatibility: we will take care of it ...
- Some properties and parameters can be passed at run-time with a standard GAUDI **jobOptions** file

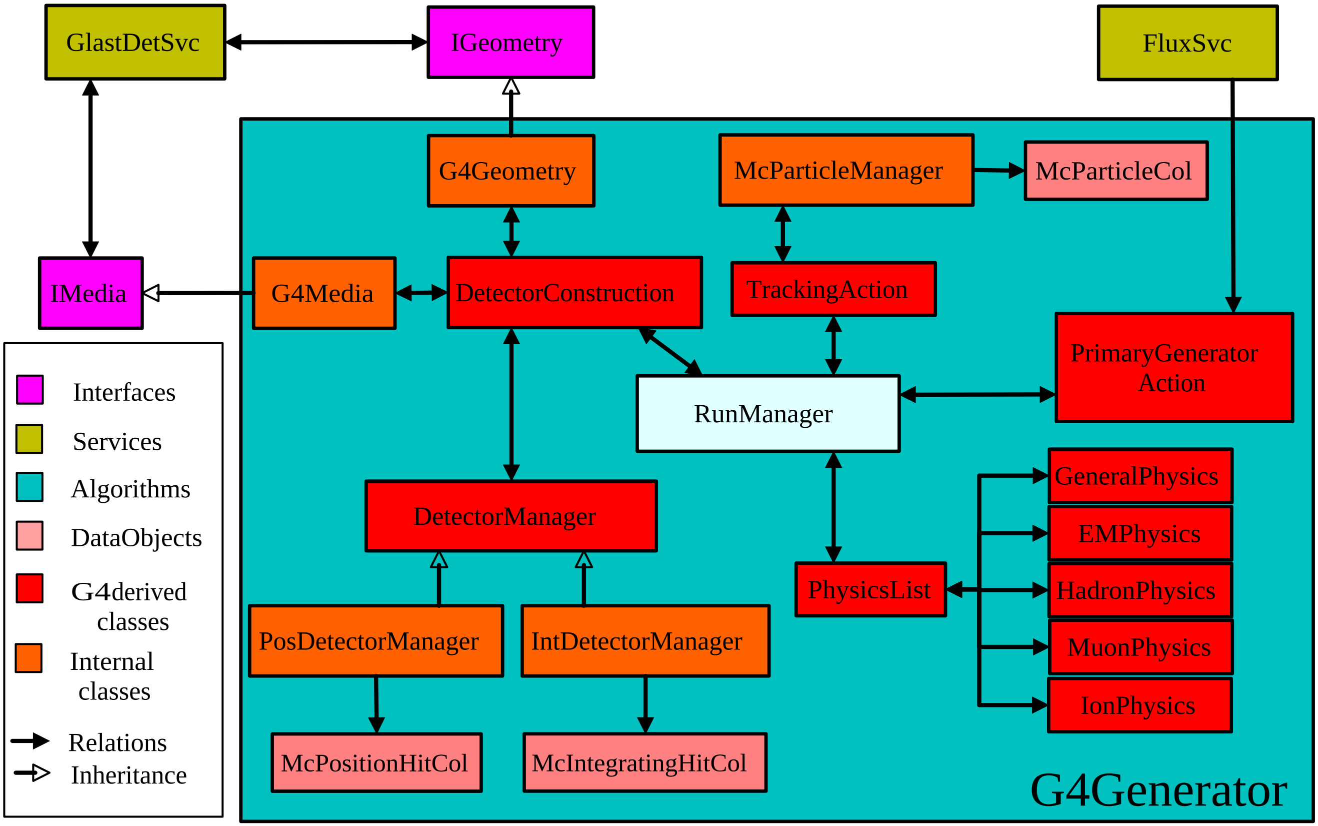
The external interactions of G4Generator



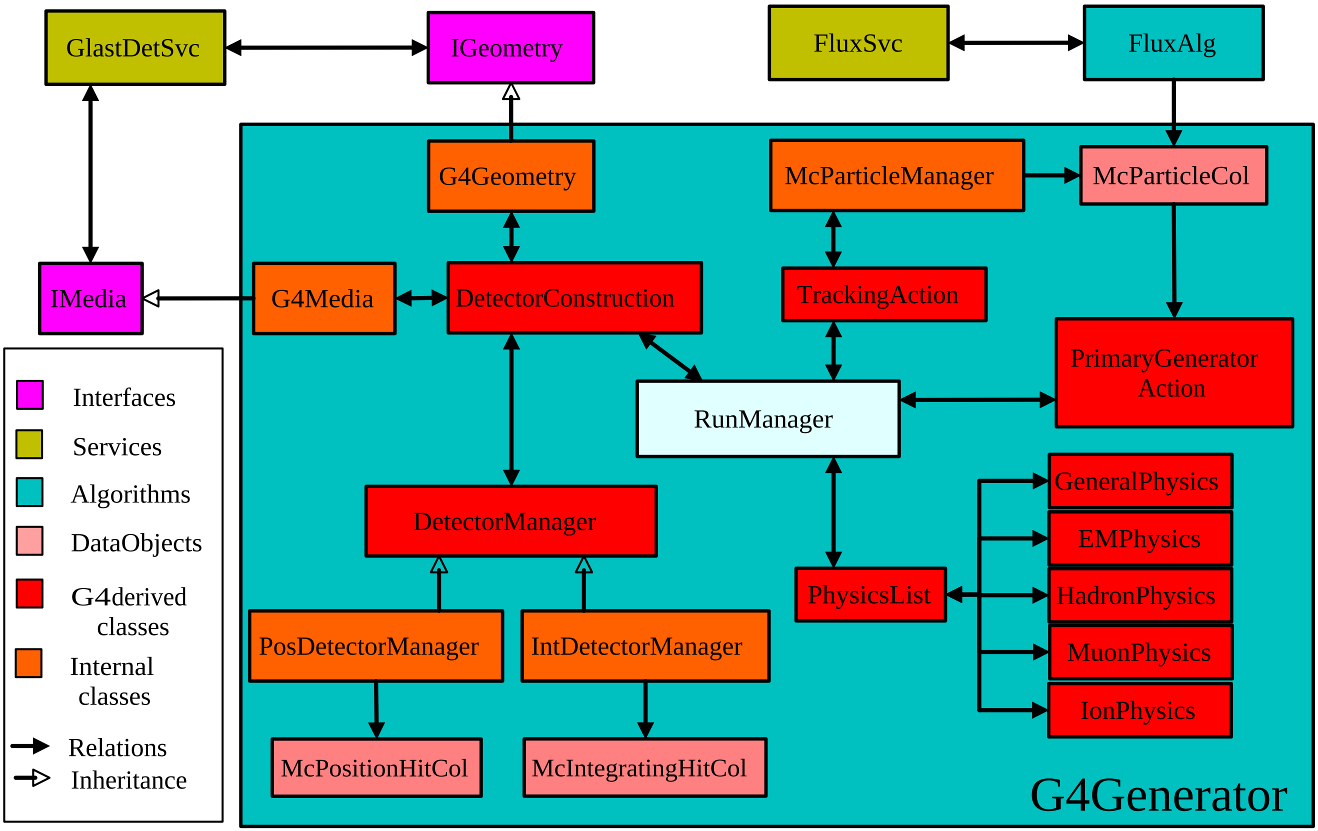
The external interactions of G4Generator (new design)



The internal structure of G4Generator



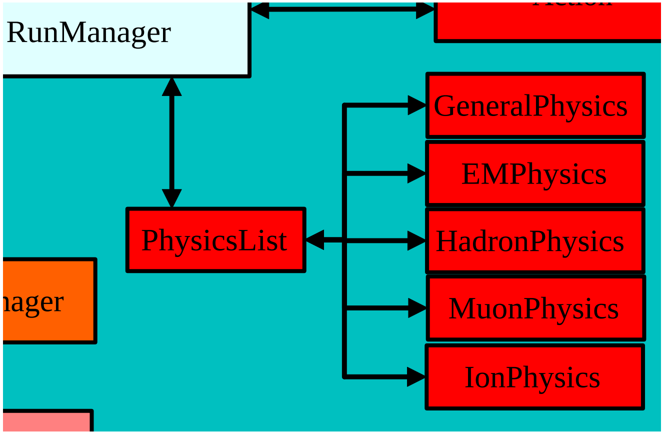
The internal structure of G4Generator (new design)



To summarize (and take a breath)

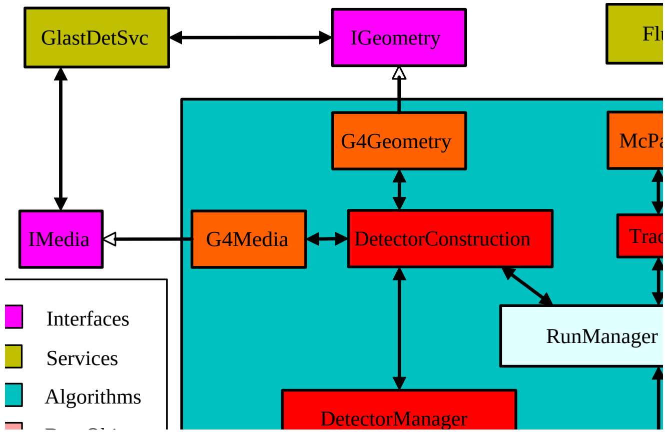
- G4Generator is a GAUDI algorithm
- It produces collections of data objects in the TDS
  - A vector of McPositionHit in /Event/MC/PositionHitsCol
  - A vector of McIntegratingHit in /Event/MC/IntegratingHitsCol
  - A vector of McParticle in /Event/MC/McParticleCol
- It builds the geometry with the help of GlastDetSvc
- It retrieves primary particles from the FluxSvc (directly or indirectly)
- Let's see some details

Physics



- The physics design from Francesco is now in `G4Generator`
- `PhysicsList` is the main class that is registered to the RunManager and hinerits from `G4VModularPhysicsList`
- This class uses 5 other classes to instantiate both particles and physics processes belonging to different physics domains
- For this, each class implements the two methods `ConstructParticle` and `ConstructProcess`

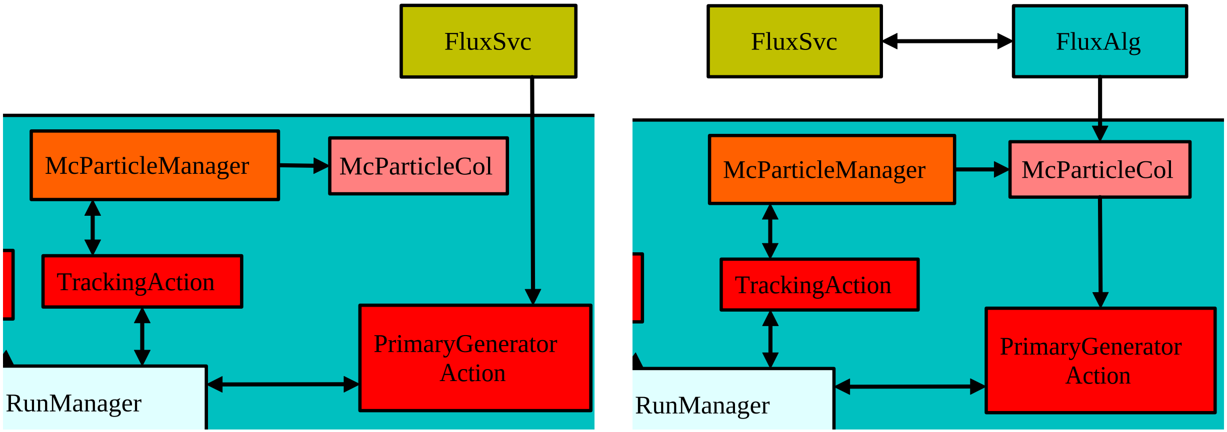
Geometry



- DetectorConstruction is derived from G4VDetectorConstruction and implements the virtual method Construct that returns a pointer to the world volume, i.e. to the root of the volumes hierarchy

- To build such a hierarchy it uses GlastDetSvc with the help of two classes
  - G4Geometry implements the abstract interface IGeometry; it is responsible for navigating the detModel hierarchy and building the ***Geant4*** geometry
  - G4Media implements the abstract interface IMedia; it is responsible for building the ***Geant4*** materials table from the detModel model.

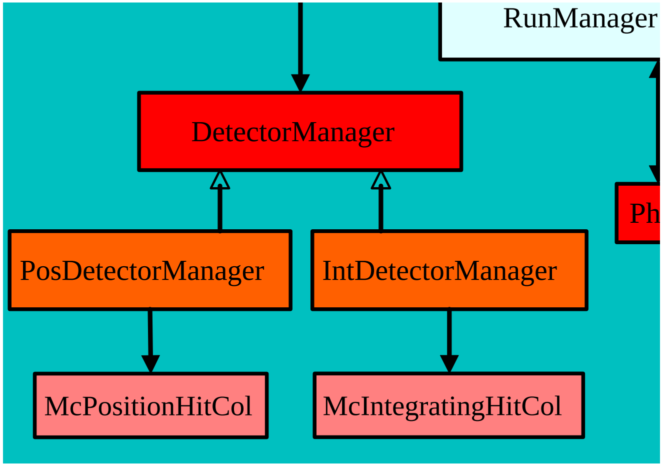
Particles



We are in the middle of a transition to a new design: `G4Generator` will not be related directly to FluxSvc, but will retrieve the primary particle as the root of an McParticle tree in the TDS

- `McParticleManager` is a singleton that manages insertion, retrieval and saving of the `McParticle` collection in the TDS; this is used by the DetectorManager classes to associate `McParticle` to hit objects
- `TrackingAction` is derived from `G4UserTrackingAction` and is used to add new `McParticle` when they are created by ***Geant4***

Hits



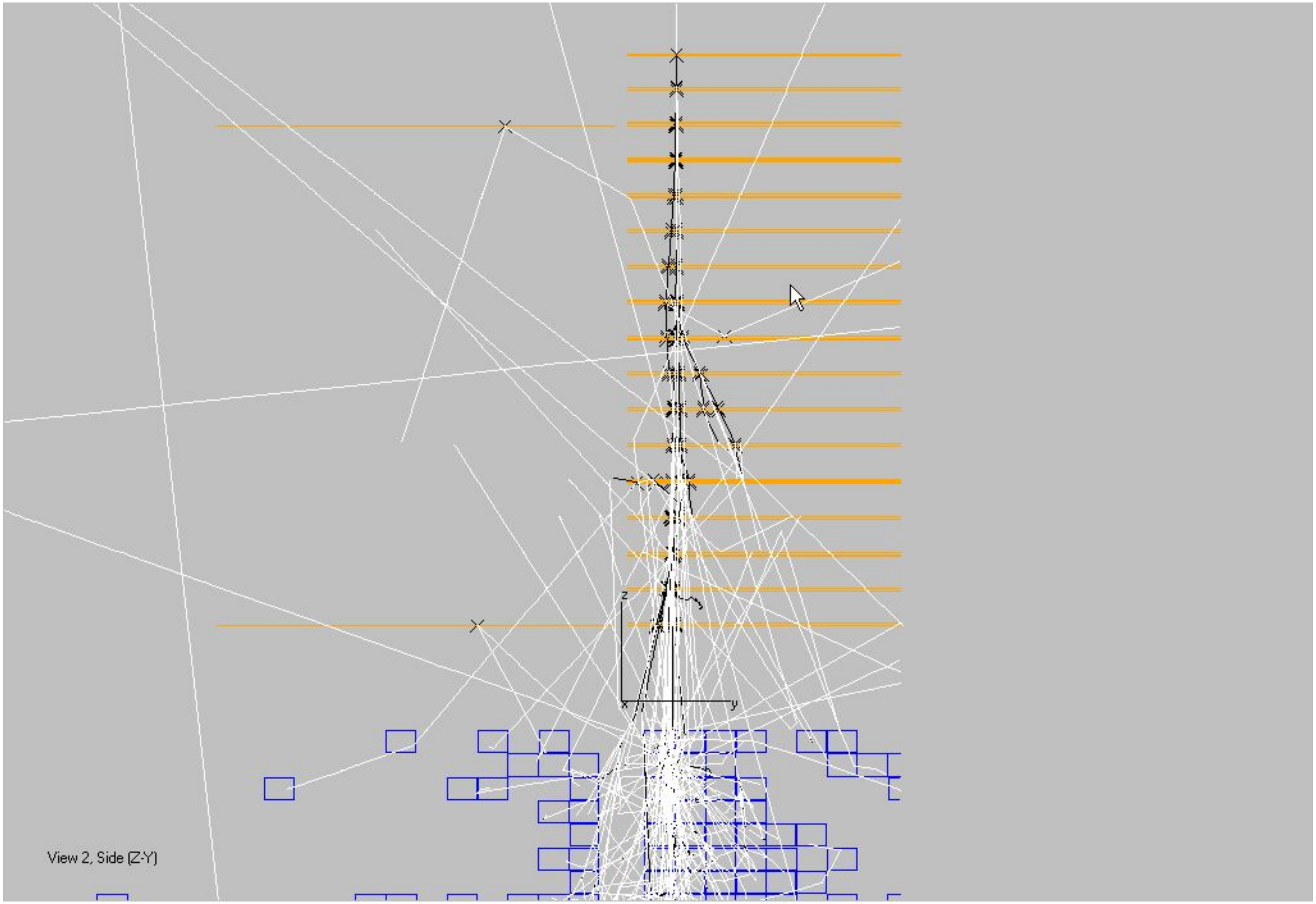
- `DetectorManager` derives from a `G4VSensitiveDetector`; it is automatically called by ***Geant4*** every time an hit occurs in a volume registered with this class (in the `DetectorConstruction`)
- ***Geant4*** calls the method `ProcessHits`; in our case this method is NOT implemented in the `DetectorManager` so that it is also an abstract class

- `PosDetectorManager` is a concrete implementation of the `DetectorManager`; it is used to manage volumes in which hits must be saved as `McPositionHit` (mainly the silicon TKR planes)
- `IntDetectorManager` is a concrete implementation of the `DetectorManager`; it is used to manage volumes in which hits must be saved as `McIntegratingHit` (mainly the ACD tiles and the CAL cells)

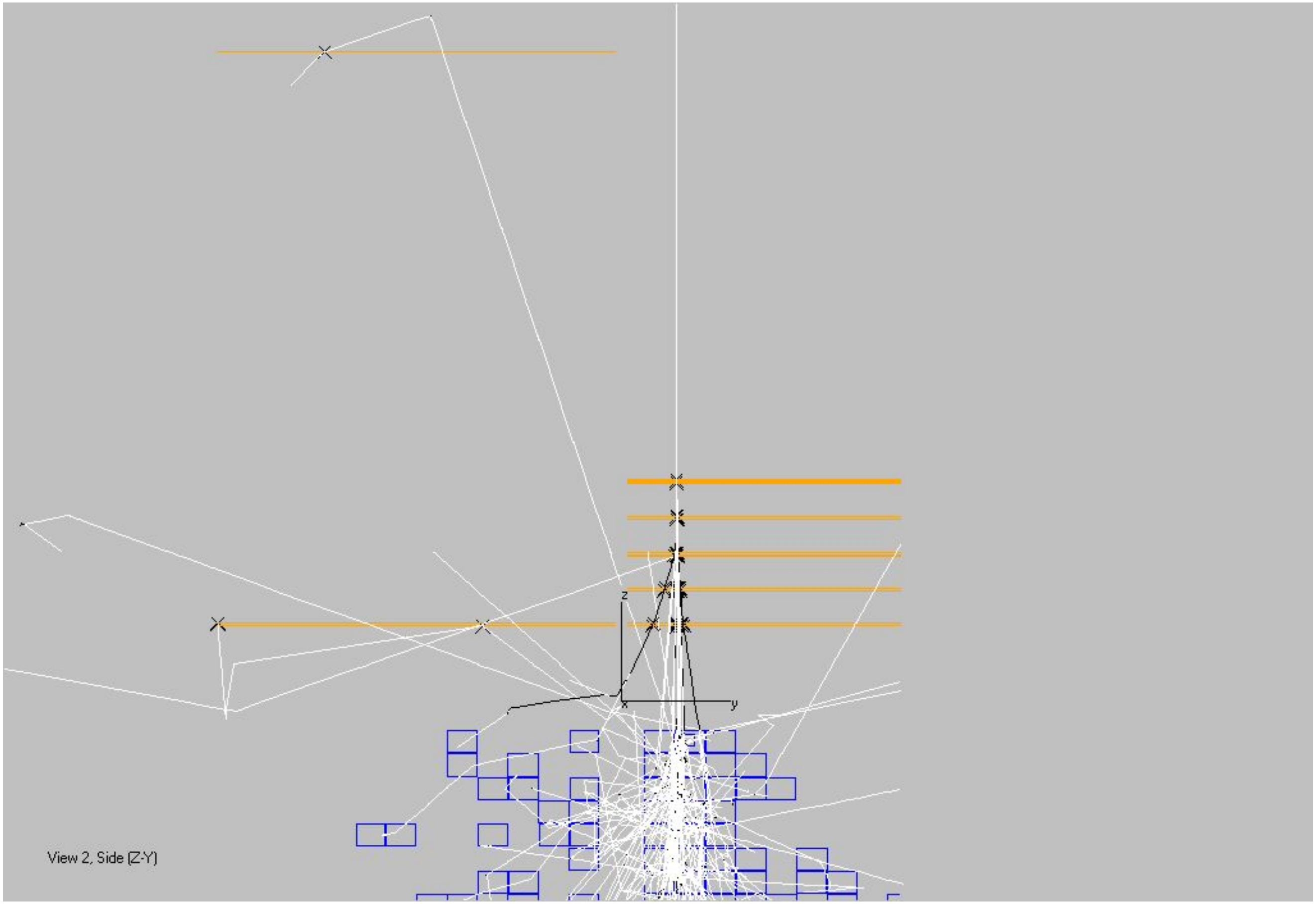
User guide

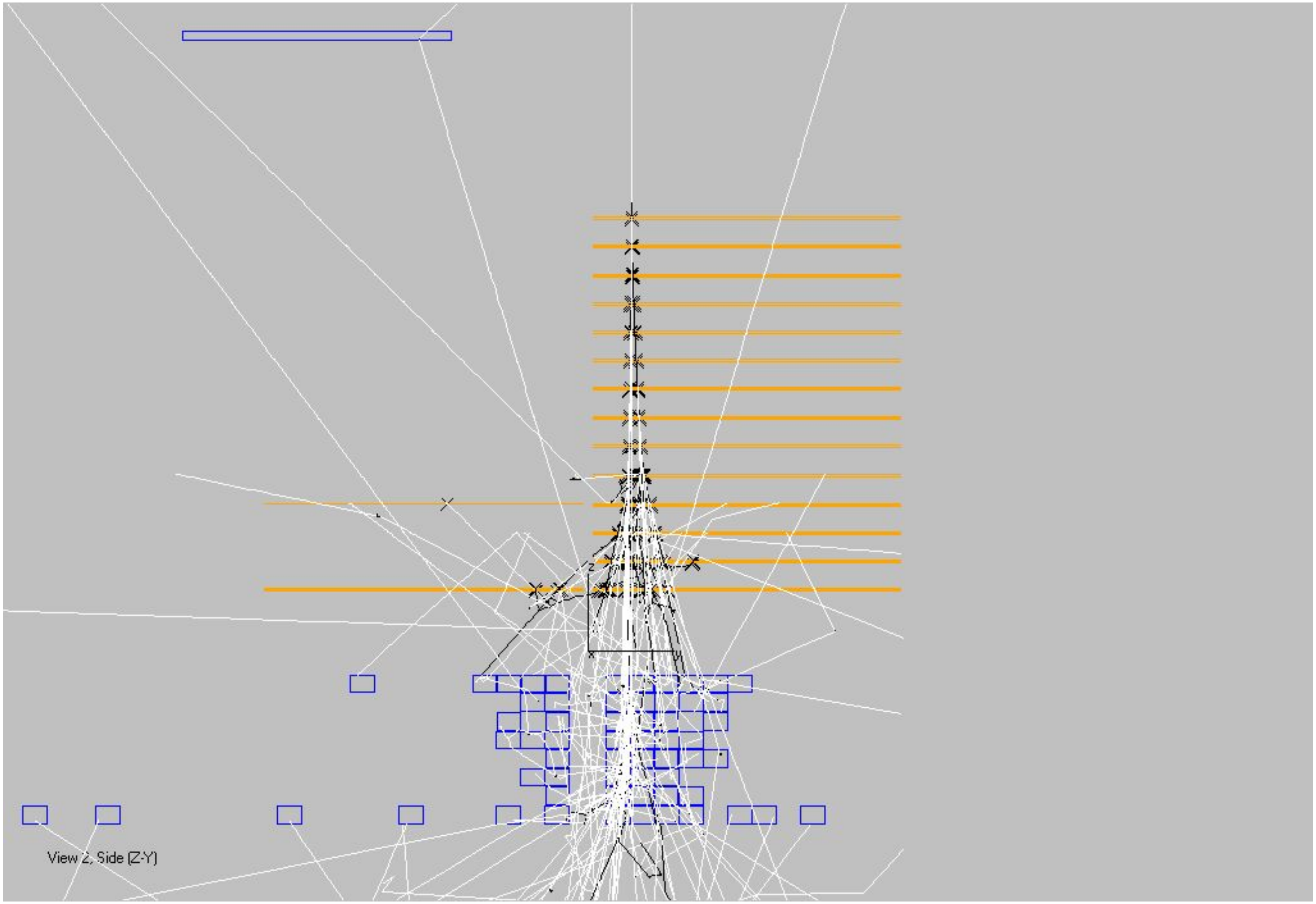
The user of G4Generator can set the following properties in the `jobObtions.txt` file

- `source_name` to set the source in the `FluxSvc`. This will be obsolete in the new design; this property will belong to the `FluxAlg`
- `UIcommands` to set an array of ***Geant4*** commands to pass to the `RunManager`; these can be used to activate verbosity, trajectories storing or other G4 relevant activities.
- `geometryMode` to set the level of details of the geometry from the xml files. For the range of possible values see the `xmlGeoDbs` package



View 2, Side (Z-Y)


View 2, Side (Z-Y)
z
x
y


View 2, Side (Z-Y)

Conclusions

- So it is possible to run ***Geant4*** simulations from Gaudi (yippee)
- It seems also to be possible to do it in a well integrated way with other ***GLAST*** specific packages (**FluxSvc**, **GuiSvc**, **GlastEvent**, **xmlGeoDbs** and **detModel**) and the **TDS**
- Although some more iterations are needed (see next slides on status) `G4Generator` seems to be now usable
- The ***Geant4*** toolkit is now quite stable and it should be not a big issue to step to new versions (but the use of a customized `RunManager` means careful evaluation)
- Last words will come from the users of `G4Generator`: is it usable? Stable? Easy? Fast enough? Fun?