Detector Simulation

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

2^{nd} Offline Tutorial 2008/12/14

Outline

Lesson Objective.

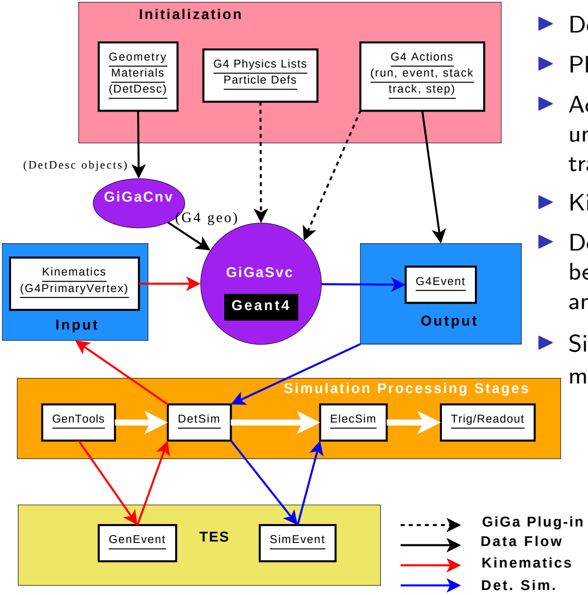
After this lesson you should understand the following things about the detector simulation:

- How it fits into the framework.
- What features it has and how to customize it.
- What data it produces.
- How to run the tutorial example and look at some simple output.

Detector Simulation Overview

- **Geant4** tracks individual particles through the a model of the detector.
- Geant4 is wrapped and run from **GiGaSvc**
- This allows integration with the **Gaudi** framework.
- Makes use of the **GiGaCnv** package to convert detector description to Geant4 geometry objects.
- Initial kinematics generated by the **GenTools** package.
- Produces **SimEvent** objects.
- Supports multiple processing models.

Interface to Geant4



- DetDesc → G4 geometry
- PhysList classes from G4dyb
- Action classes for unobservable statistics & trajectory recording
- Kine in, G4 data out
- DetSim algs interface between Kine & GiGa/G4 and TES
- Simple linear processing model shown as example.
  - Alternative processor algorithm for "15 minutes" style model.

Features: Physics Lists

The available Physics is broken up into a list of six elements (listed with the implementing class name):

General (DsPhysConsGeneral) constructs particles and decay processes.

Optical (DsPhysConsOptical) scintillation and Cherenkov processes.

E&M (DsPhysConsEM) Brehm, ionization, photo-electric, Compton, gamma conversion, scattering, pair production and other E&M processes. Note, this must be on for Optical to work.

Elctro Nuclear (DsPhysConsElectroNu) photo-nuclear, electro-nuclear and muon-nuclear processes.

Hadronic (DsPhysConsHadron) many hadronic processes.

Ionic (DsPhysConsIon) processes involving ions

Typically the last three need not be used for water pool or dry-run studies.

Feature: Selective Geometry

- Configuring the detector description (geometry) was just covered.
- How much of that geometry is loaded can be limited to just one or more of the three sites.
- Due to the optimized loading it is not yet clear if limiting this is useful.

Features: Particle Historian and Unobservable Statistics

DetSim makes use of the Historian package that provides:

Particle History
: that records how all, or a subset of all, particles traverse the detectors.

Unobservable Statistics
: that records MC truth information on an event by event basis.

Both use a sophisticated rule based configuration to allow precise control over how little or how much information is saved.

Features: Customization

DetSim is highly customizable:

Geometry
: described in a flexible XML based form

Geant4 user code
: (actions, new physics) can be added in a modular fashion that will not interfere with existing code.

Configuration
: allows modules to be turned on or off and can control how much and what type of output is generated.

Configuration: a default DetSim

The default configuration is done like:

```
import DetSim
ds = DetSim.Configure()
```

This has:

- Geometry for all three sites loaded
- All physics turned on, default cuts
- Inserts “push” processing mode algorithms
- No Historian nor Unboserver.

Configuration: Limiting Physics and Geometry

Any custom list can be used but two lists are predefined:

`physics_list_basic` first three, enough for most pool studies.

`physics_list_nuclear` last three, additionally needed for AD.

They can be set via the `physlist` argument, eg:

```
# some:
ds = DetSim.Configure(physlist=DetSim.physics_list_basic)
# all (default):
ds = DetSim.Configure(physlist=DetSim.physics_list_basic \
                             +DetSim.physics_list_nuclear)
```

Limiting geometry is done by with the `site` argument which is a comma separated string:

```
# Just far:
ds = DetSim.Configure(site="far")
# all (default):
ds = DetSim.Configure(site="far,dayabay,lingao")
```

Configuration: Particle Historian

The historian takes a set of track and vertex selection rules which are set like:

```
ts = "..."
vs = "..."
ds = DetSim.Configure(...)
ds.historian(trackSelection=ts, vertexSelection=vs)
```

Selections are logical statements using the rule language of the Historian package. For example, if you only care to record neutron tracks:

```
ts = "(pdg == 2112)"
vs = "(pdg == 2112)"
```

Configuration: Unobservable Statistics

The “unobserver” takes a list of statistics to collect. For each the statistic has a name, a variable and a rule saying governing when to update. The sum, sum^2 and count are kept for each statistic. It is configured like:

```
ds.unobserver(stats=[...])
```

Each element of the stat list is of the form: `[name, variable, rule]`. For example, the starting time, energy and PID could be defined like:

```
params = {
  'start' :"(start > 0)",
  'track1':"(id==1 and ProcessType==1)",
}
ds.unobserver(stat=[
  ["pdgId_Trk1","pdg","%(track1)s and %(start)s"%params],
  ["t_Trk1",    "t" , "%(track1)s and %(start)s"%params],
  ["e_Trk1",    "E",  "%(track1)s and %(start)s"%params],
])
```

Unique Detector IDs

`Conventions/Detectors.h` defines globally unique IDs for detector sensors (PMTs and RPCs).

- ID is a packed `int`
  - byte 1: unique `Site::Site_t` number
  - byte 2: unique `DetectorId::DetectorId_t` number
  - bytes 3-4: unique sensor (ie PMT/RPC) ID - detector specific packing.
- Unpacking performed by these classes (all in `DayaBay::`)
  - `Detector` access site/detector level values
  - `DetectorSensor` access sensor values generically
  - `AdPmtSensor` access sensor ID via AD ring# and column#
  - `PoolPmtSensor` access sensor ID via Pool specific addresses
  - `RpcSensor` access sensor ID via RPC specific addresses

  Last two are still being defined.
- We should use this for all PMT/RPC identifying.

SimEvent Data Objects

DetSim produces SimEvent data objects.

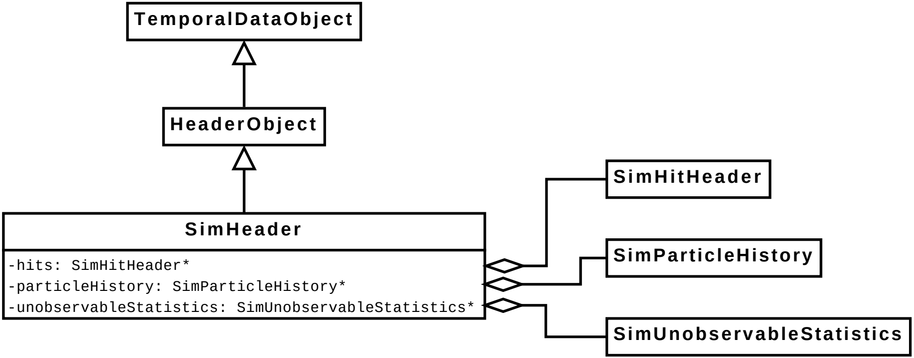
- In `DataModel/SimEvent` package.
- Default TES location: `/Event/Sim/SimHeader`
- Three "sub" header objects:

  `SimHitHeader` access to all the collections of hits.

  `SimParticleHistoryHeader` access to intermediate particle tracking information.

  `SimUnobservableStatisticsHeader` access to intermediate physics (eg. "total photons in water")

SimHeader - UML Class Diagram



SimHit data

SimHitHeader gives access to a SimHitCollection based on the site/detector unique ID.

SimHitCollection stores back pointer to the hit header and holds a vector of SimHit

SimHit expected hit quantities. Subclassed for hits in specific sensors:

- SimPmtHit for optical photons hitting PMTs
- SimRpcHit for particle hits on RPCs

SimHit data - continue

`SimHit` base class:

- **hc** pointer to parent hit collection
- **hitTime** `double`, hit time relative to primary vertex time
- **localPos** `Hep3Vector`, hit position in sensors local coord.
- **sensDetId** `int`, globally unique ID of sensor
- **weight** `float`, some weight

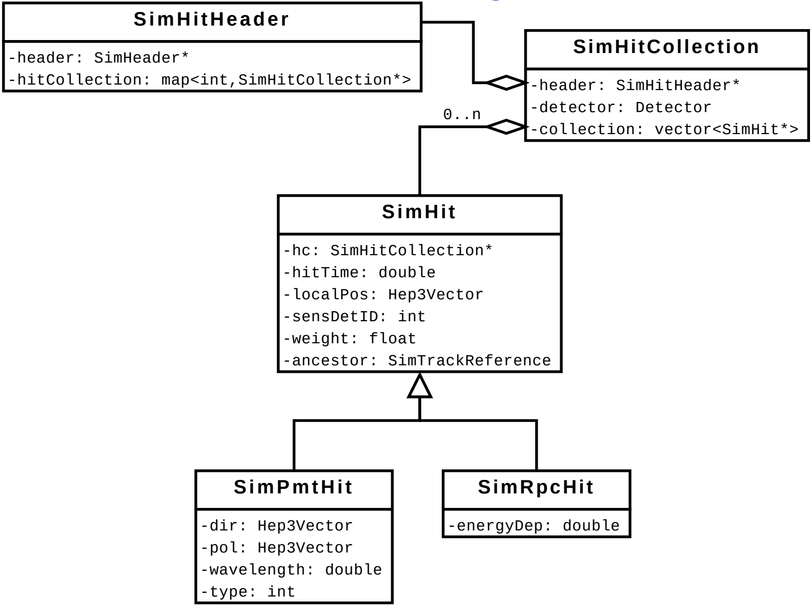
`SimPmtHit` subclass:

- **parent** pointer to particle that produced photon that hit PMT
- **dir** `Hep3Vector`, photon direction in local PMT coord
- **pol** `Hep3Vector`, photon polarization in local PMT coord
- **wavelength** `double`, photon wavelength
- **type** `int`, some hit type code(?)

`SimRpcHit` subclass:

- **particle** pointer to particle that hit RPC
- **energyDep** `double`, energy deposition of hit

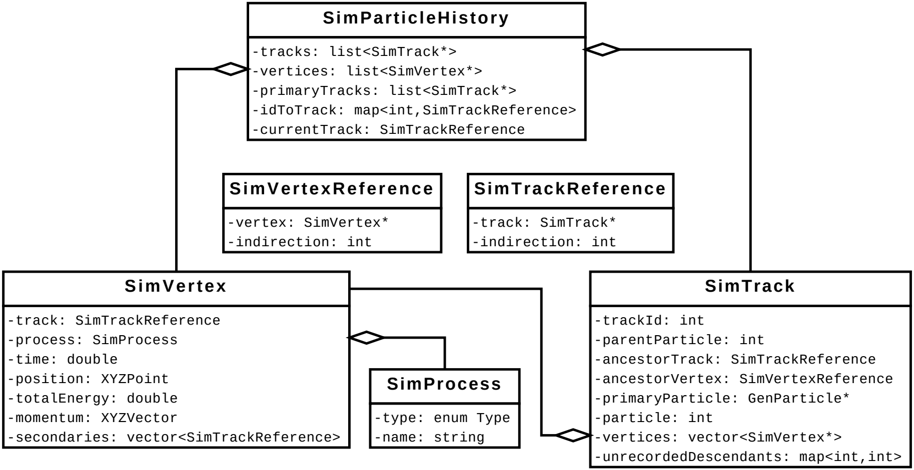
SimHitHeader - UML Class Diagram



SimParticleHistory data

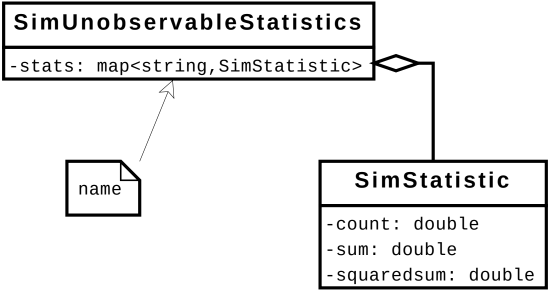
- Records all tracks and vertices passing user's rules.
- Vertex is where/when "something" happened on a track.
- Track can have multiple vertices.
- Track references an "ancestor" track and vertex, either direct parent or through some number of unsaved ancestors.
- Mapping from G4 ID to track

SimParticleHistory - UML Class Diagram



SimUnobservableStatistics data

- A single map from a statistic's name to the count, sum and squared sum that was collected.



Examples

Example algorithm to histogram some simple hit quantities:

- `tutorial/Simulation/SimHistsExample/src/SimHists.cc`

JOS to drive GenTools, DetSim and histograms:

- `tutorial/Simulation/SimHistsExample/python/simhists.py`

JOM to do the same:

- `nuwa.py -n 3 SimHistsExample.Tutorial`
- `root -l simhists.root`