Simulation Data Model and File Input/Output

Brett Viren

Physics Department

BROOKHAVEN NATIONAL LABORATORY

2nd Offline Tutorial 2008/12/10

Outline

Transient Data Model

DetSim Data Model

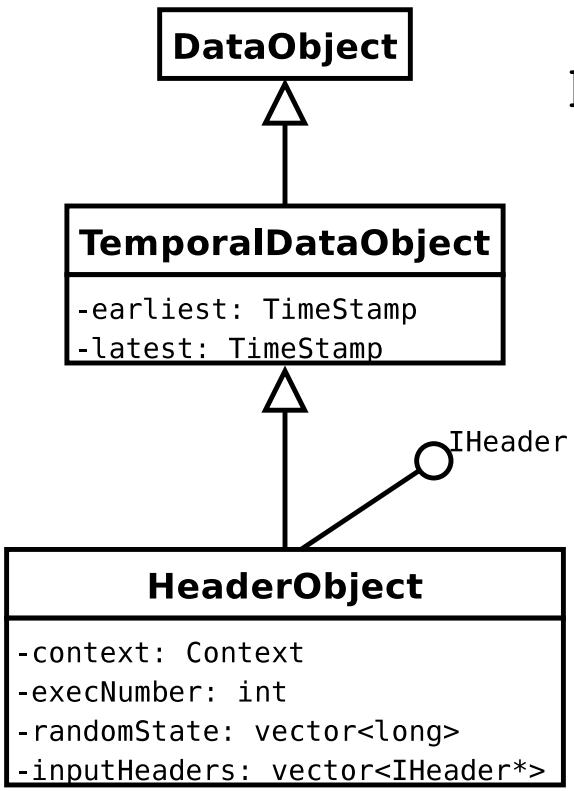
Persistent Data

Converters

Persistent File Format

Configuration

Transient Base Classes



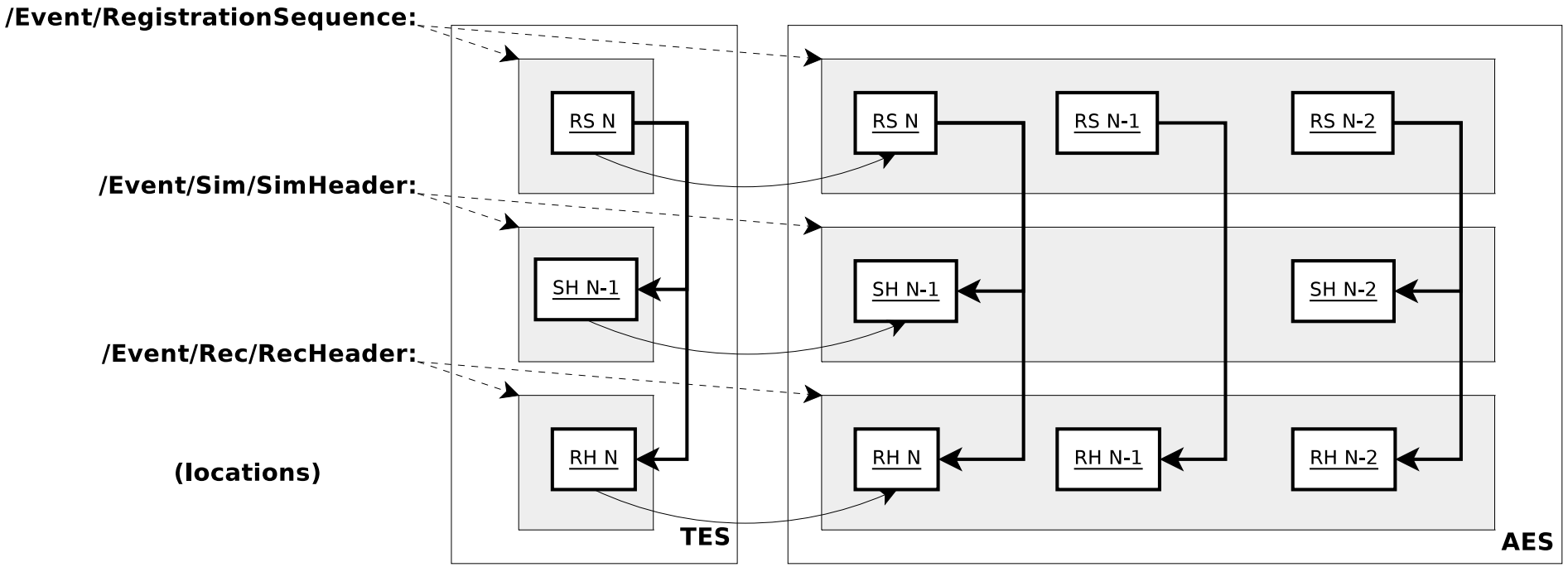
`TemporalDataObject`: used to implement analysis time window.

- Earliest and latest time of its data

`HeaderObject`: bookkeeping:

- Context (reference TimeStamp, detector, sim flag)
- Execution count ("event" number)
- Random state before algorithm runs, in order to reproduce
- Other `HeaderObjects` used as input to make this one.

RegistrationSequence



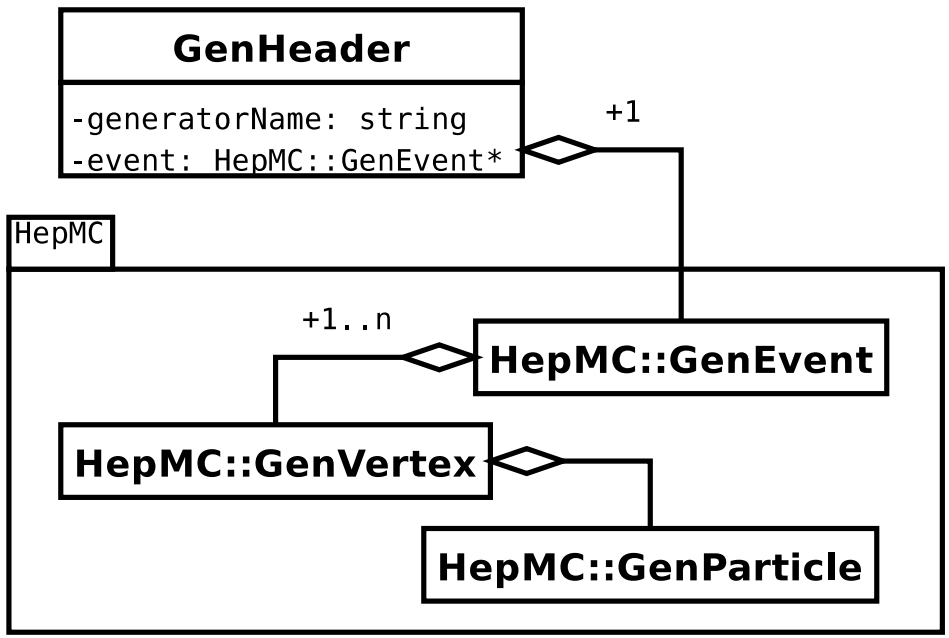
The RegistrationSequence (RS) captures the results of one execution cycle. When a new object is added to TES, TES registers it with RS and inserts it into AES collection at same location. Note missing `SimHeader` (SH) possibly due to SH_{n-2} being an IBD. (Some data not shown for simplicity).

Requirements for Transient Data Model Classes

Transient Data Model Classes:

- Should be designed in consultation with the offline group.
- Should use GaudiObjDesc XML for their definition.
- Must inherit from `TemporalDataObject` if they will participate in the time based analysis window.
- Must inherit from `HeaderObject` and satisfy persistency requirements if they will be persisted.
- Suggest using `DybAlgorithm` as base class for algorithms that produce Data Model objects.

Generator Level Data



- Simple, just holds a `HepMC::GenEvent`
- Default location: `/Event/Gen/GenHeader`.

Transient Data Model

DetSim Data Model

Persistent Data

Converters

Persistent File Format

Configuration

SimEvent Data Objects

DetSim produces SimEvent data objects.

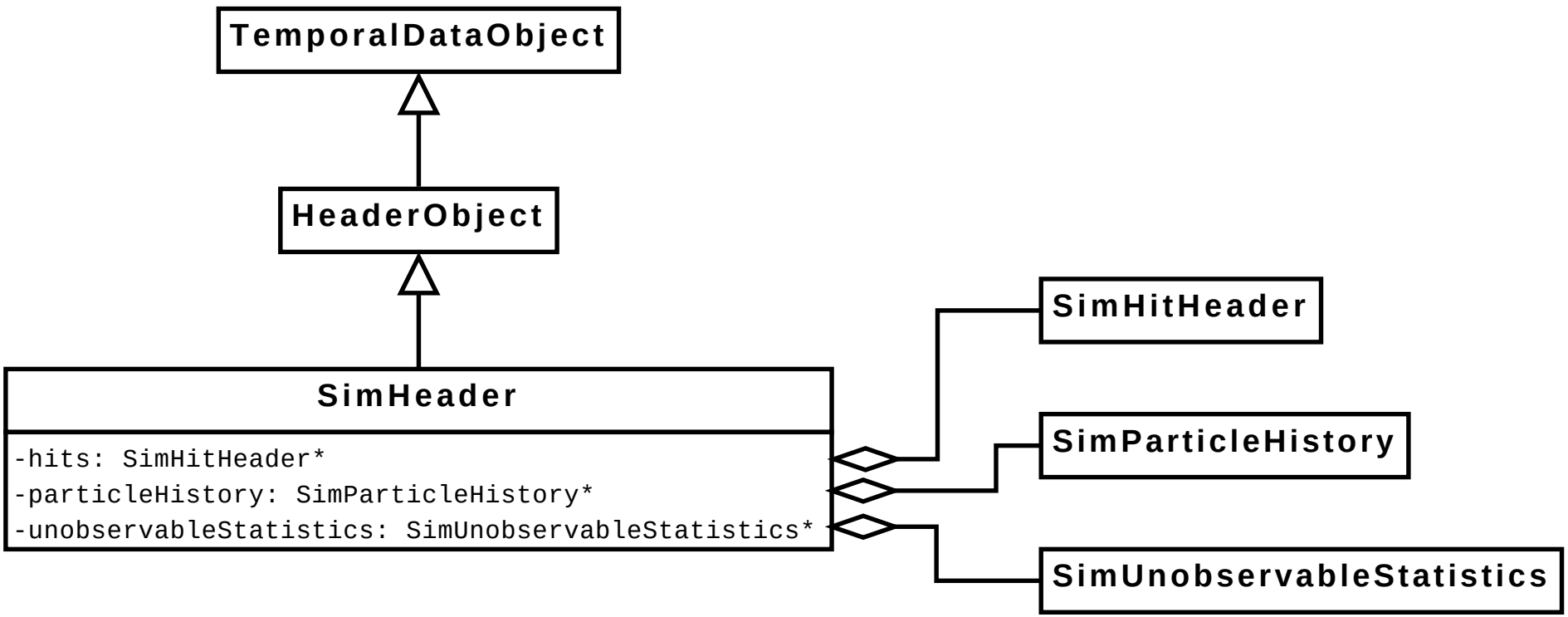
- In `DataModel/SimEvent` package.
- Default TES location: `/Event/Sim/SimHeader`
- Three "sub" header objects:

  `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

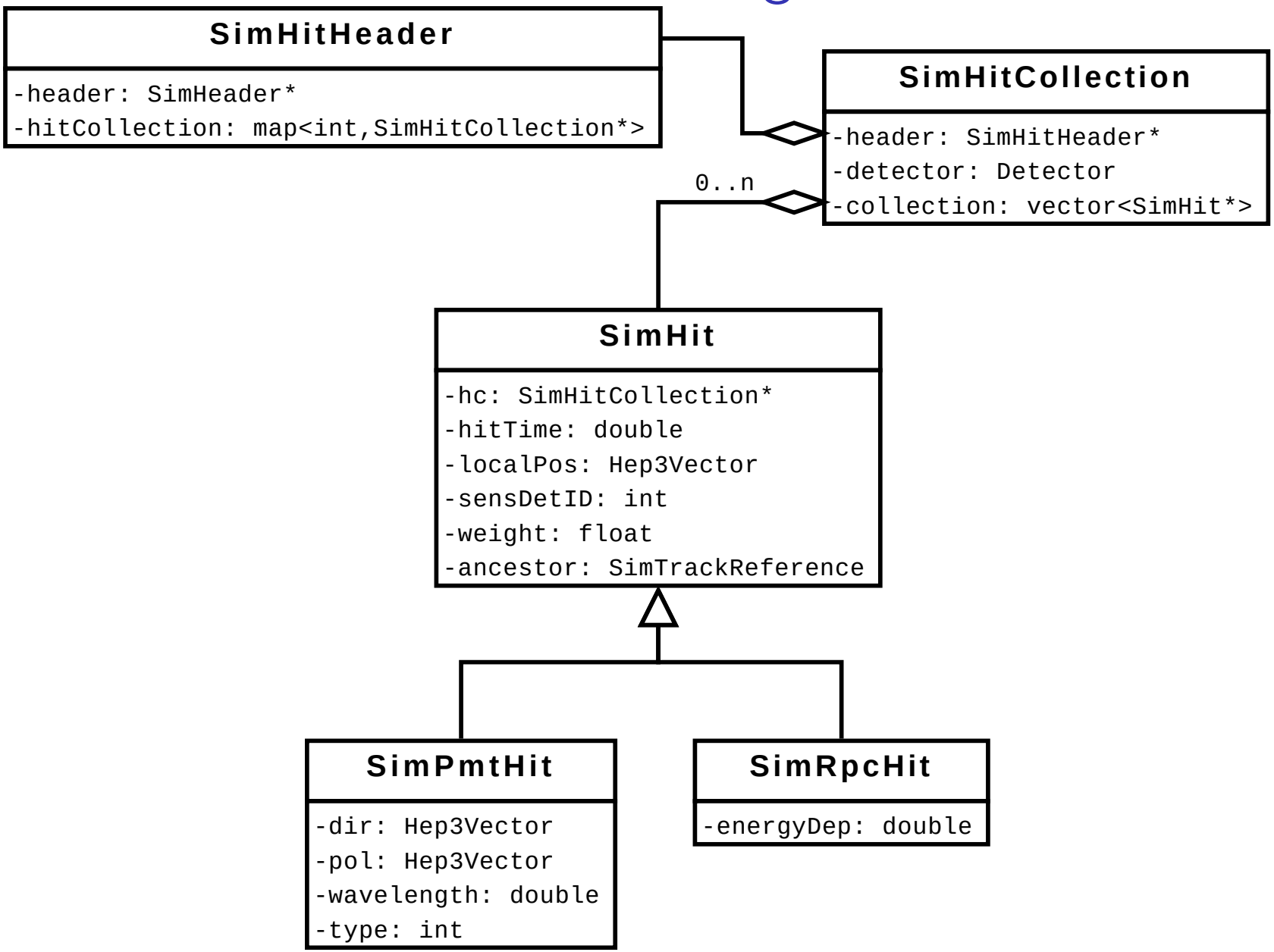
`SimPmtHit` subclass:

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

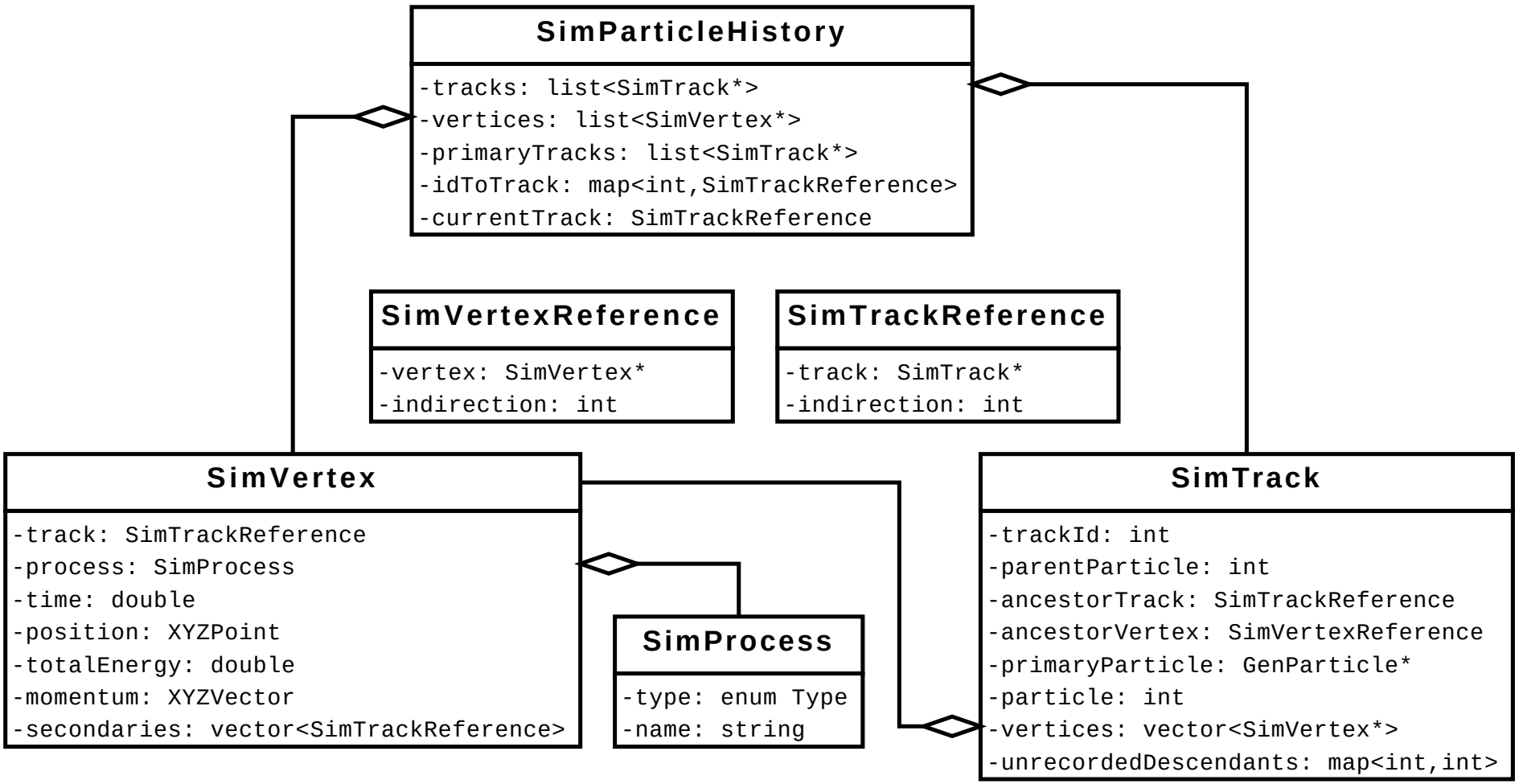
SimHitHeader - UML Class Diagram



SimParticleHistory data

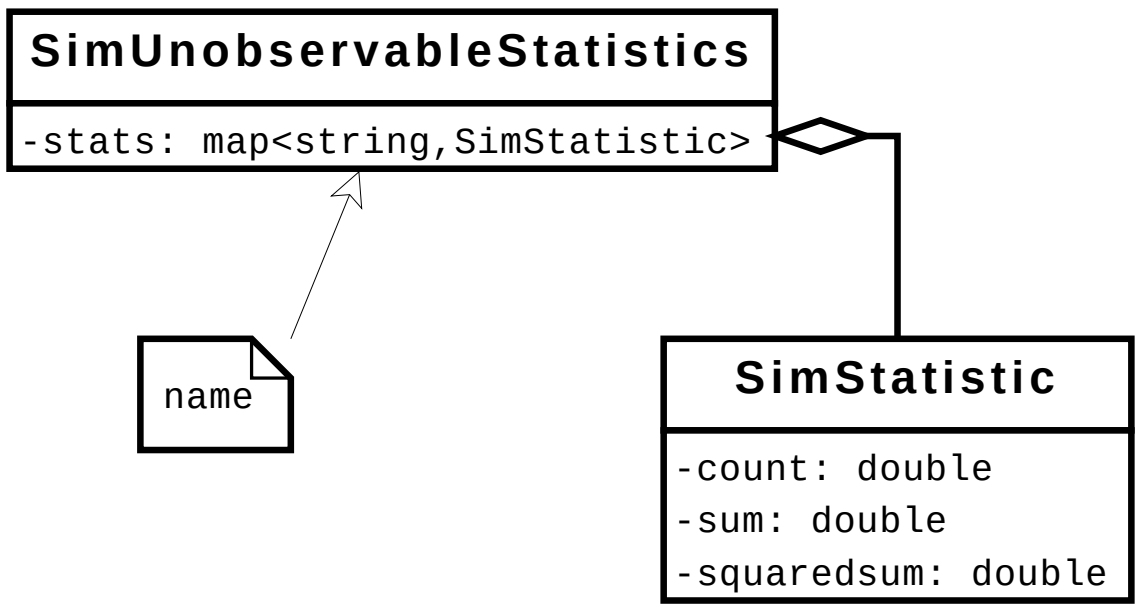
- Records all tracks and vertices passing user's rules.
- Vertex is where/when "something" happened on a track.
- Track can have multiple vertices.
- Track references an "ancestor" track and vertex, either direct parent or through some number of unsaved ancestors.
- Mapping from G4 ID to track

SimParticleHistory - UML Class Diagram



SimUnobservableStatistics data

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



Transient Data Model

DetSim Data Model

Persistent Data

Converters

Persistent File Format

Configuration

I/O Streams

Streams:

- An abstract serial sequence of one type of data.
  - Hides all ROOT I/O stuff.
  - Maps a location in the TES/AES to a TDirectory & TTree in a file
- Streams are logically independent from one another.
  - I/O of many streams may come from / go to one or many files.
- Some file level control:
  - Output may be broken into different files (limit file size, implement sub-runs).
  - Input may be taken from multiple, sequential files (logical concatenation)

RootIO Subproject

`RootIOSvc`:

- Provides Streams and ROOT file management classes.
- A default event selector that assumes all streams are synchronized at the TTree entry level.
- The controlling conversion service `RootIOCnvSvc`.
- Base class for all persistent classes and converters.

`Per*` Packages:

- Suite of `Per*` packages with current persistent classes and converters[1].

`RootIOTest`

- Unit tests and temporary DybStoreAlg (TES driven I/O)

[1] This should probably move to `DybIO`!

Daya Bay File I/O Policy

The `DybIO` package builds on `RootIO` to apply Daya Bay specific I/O policy. Default policy is:

- On input, the state of the TES/AES should be identical to what it was when the original data was output to file.
- Gaudi's "nextEvent" is interpreted as "read/write next RegistrationSequence and then read/write all its registered objects".

Implementation classes:

`DybEvtSelector` implementing default input policy.

`DybStorageSvc` implementing default output policy.

- Both support the (non-mixing) "push" and "pull" processing models.
- Mixing-push requires a new event selector (in progress: Xingtao).

Requirements for Persistent Data Model Classes

Persistent classes:

- Must inherit from `PerHeaderObject`.
- Live in a package named `Per` + (transient DataModel package name).
- Any reference-by-pointer in the transient class must be stored as index-into-array
- Owned-by-pointer can remain unchanged
- Should leave data members public and not name them with any prefix (facilitates access in raw-ROOT session)

Transient Data Model

DetSim Data Model

Persistent Data

Converters

Persistent File Format

Configuration

Converters between Transient & Persistent objects

Converters:

- Responsible for copying information between the two representations.
- Hand written, and not always trivial.
- `RootIOTypedCnv` base class provides some automation.
- Registered with Gaudi:
  - Invoked automatically
  - Looked up by "technology ID" (ie, ROOT) and class ID

Requirements for Converters

Converters:

- Inherit from `RootIOTypedCnv`
- Implement `PerToTran()` and `TranToPer()` methods.
- May implement `fillRepRefs()` and `fillObjRefs()` to satisfy intra-object references.
- Copy information between representations, including those in base classes (can use on `HeaderObjectCnv` methods).

Some care is needed:

- Convert reference-by-pointer to index-into-array.
- Accept changes in persistent and transient model.

Transient Data Model

DetSim Data Model

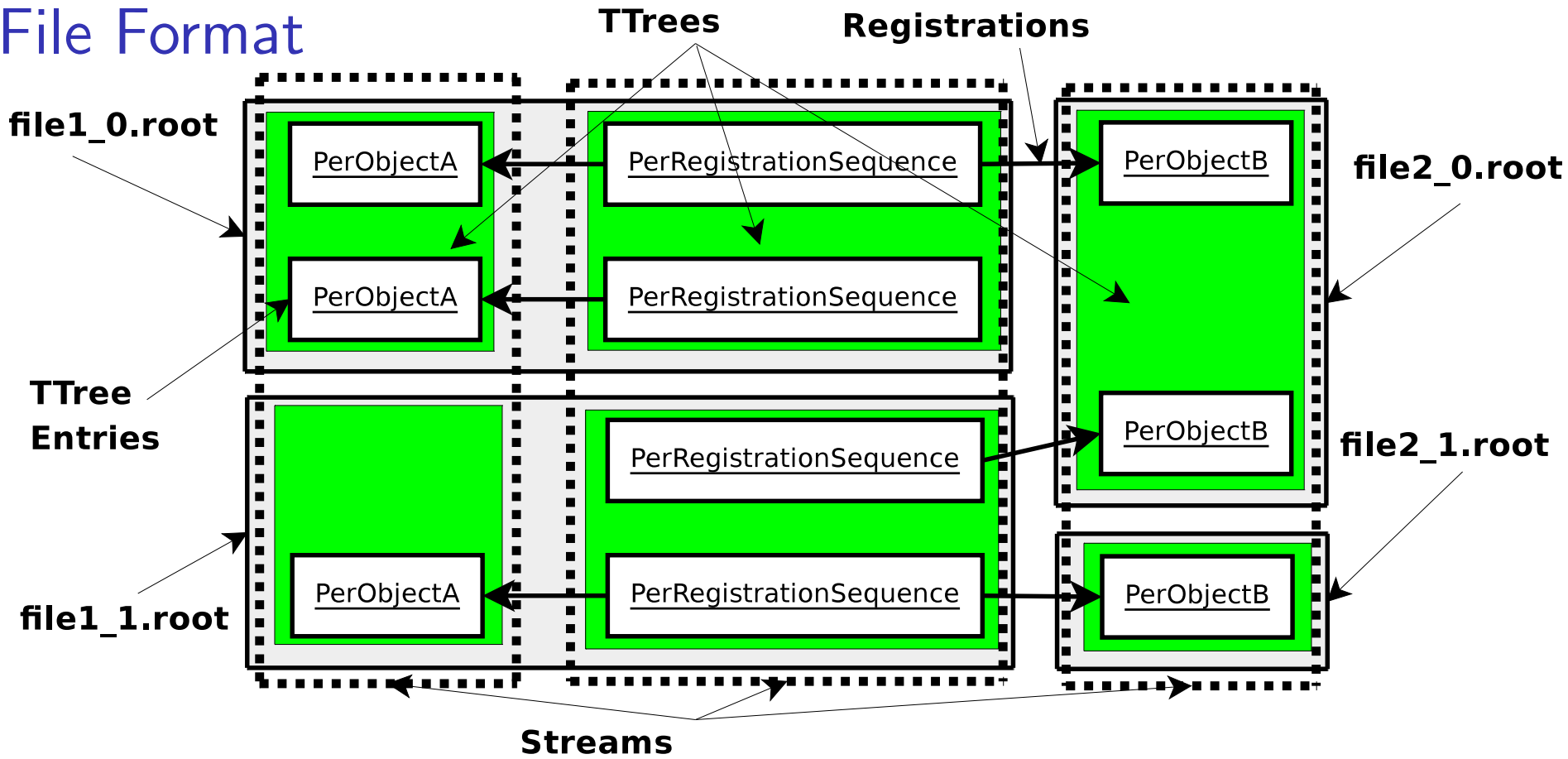
Persistent Data

Converters

Persistent File Format

Configuration

File Format



- ROOT TFiles with one or more TTree objects, one per stream.
- A stream's TTree can span sequential TFiles.
- Different streams can be associated with different TFiles.
- The RegistrationSequence stream/file is critical as it is the key by which data is written and read back.

Transient Data Model

DetSim Data Model

Persistent Data

Converters

Persistent File Format

Configuration

File I/O Configuration, simple

Configuration is handled by `nuwa.py`.

```
# Single output file:
nuwa.py [...] -o output.root
# One or more, sequential input files:
nuwa.py [...] input1.root input2.root input3.root ...
```

This almost always what you want to do.

File I/O Configuration, advanced

For experts, it is possible to spread streams across files. For example, maybe we separate the `RegistrationSequence` stream from all others.

```
# Save RegistrationSequence stream to ``rs.root'',
# all others to ``bare.root''
nuwa.py [...] -o bare.root \
          -O '{"/Event/RegistrationSequence": "rs.root"}'
# Read same back in:
nuwa.py [...] bare.root \
          -I '{"/Event/RegistrationSequence": "rs.root"}'
# or:
nuwa.py [...] -I '{"/Event/RegistrationSequence": "rs.root", \
                   "default": "bare.root"}'
```

- Remember, one must keep the RegistrationSequence stream around in order to read back any files that lack it!
- A location of "default" indicates that this stream supplies/accepts all streams not explicitly mapped.

Still to do

- Output is currently triggered at end of TES execution cycle (BV)
  - Can not currently support data reduction.
  - Data needs to be output as it is being trimmed from the AES.
  - Solution understood, but slightly non-trivial.
- Not yet support for persisting inter-execution-cycle references (BV)
  - Problem for the "pull" method and in the "push" method where one needs to isolate individual Readouts that come from a single event.
  - Solution mostly understood, but will need slight changes in how the `RegistrationSequence` works.
- Supporting "pull" input requires modifying existing pull algorithms (BV & Zhe).
- Mixing+push: requires special event selector to replace `DybEvtSelector` (XingTao).