Daya Bay Detector Model and Response Simulation

Brett Viren

Physics Department

Daya Bay Offline Workshop @ BNL, April 2008

Contents

1 Modeling Daya Bay Detectors and Their Environment
- Overview of Detector Description
- Detector Description Sections
  - Material Section
  - Geometry Section
  - Structure Section
  - Misc. Sections
- Detector Description Visualization Methods

2 Simulating Detector Response to Particles
- Overview of Detector Simulation
- Configuration
- Simulated Hit Data
- Examples
- Caveats, a.k.a. To Do List

Forms of Detector Description Information

The detector description is currently available in three forms:

XML Files: The source of (ideal) description is in the form of XML files following well defined DTD schema. The `XmlDetDesc` package contains these files.

TDS Objects: The full description is available as objects from the Gaudi Transient Detector Store (TDS). The `DetDesc` package provides these objects.

Geant4 Geometry: The TDS objects are convertible to Geant4 geometry objects for use by the detector simulation. The `GiGaCnv` package provides this conversion.

When necessary, an alignment database can be built to supply offsets to the TDS objects (and thus to Geant4 as well).

Detector Description Sections

The detector description is divided into 3 main sections:

Materials:	The makeup of all materials.
Geometry:	The full hierarchy of logical/physical volume containment.
Structure:	The parallel, subset hierarchy of important Detector Elements.

There are also the **Surface** and **Tabproperty** ("tabulated properties") sections for defining properties.

Detector Description XML Basics

Top level `dayabay.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDDB SYSTEM "DTD/structure.dtd">
<DDDB>
  <catalog name="dd">
    <catalogref href="materials/materials.xml#Materials"/>
    <catalogref href="geometry.xml#Geometry" />
    <catalogref href="structure.xml#Structure" />
  </catalog>
</DDDB>
```

- **DDDB** = "Detector Description DataBase", top level XML tag.
- Specify the **DTD** schema file for XML validation.
- **Catalogs** build up TDS paths under `/dd`
- Flexible **href** URL reference mechanism (for referencing `<catalog>`, `<logvol>`, `<detelem>`, `<tabproperty>`, `<surface>`, `<material>`, `<isotope>`, `<element>`)

XML Basics - Parameters

Hard coded numbers are a necessary evil - but put them in parameters!

```
<parameter name="MidSiteX" value="60910.93*m"/>
<parameter name="FarSiteRotZ" value="29.45*deg"/>
<parameter name="PoolIWSBevelSize"
           value="2*(sqrt(2)-1)*(PoolIWSThickness)"/>
<parameter name="ADadeHead" value="1.0*m"/>
<parameter name="PmtHemiFaceROC" value="131*mm"/>
```

- Values should always use units!
- Mathematical expression calculations are supported.
- Above is a misc. sampling. Each major element has self-contained self-contained, `parameters.xml` file.
- Parameter files are included in other XML files via the “external entity” mechanism $\longrightarrow$

External Entity Inclusion Method

An XML version of C++'s "#include", Pool/geometry.xml example:

```
<!DOCTYPE DDDB SYSTEM "../DTD/geometry.dtd" [
  <!ENTITY SiteParameters SYSTEM "../Sites/parameters.xml">
  <!ENTITY PoolParameters SYSTEM "parameters.xml">
 ]>
<DDDB>
&SiteParameters;
&PoolParameters;
...
```

- Must associate an entity name (eg. "PoolParameters") to a source, in this case a "SYSTEM" file.
- Wherever the entity name is later placed, the parser will expand it to the file contents.

File Organization Basics

XML files are organized.

- All are found under `XmlDetDesc/DDDB/`
- Subdirectory for each major grouping (`Sites`, `Pool`, `AD`, `AdPmts`, `PMT`, etc).
- Top level `geometry.xml` and `structure.xml` in each directory.

Some files are generated:

- For repetitive, algorithmic descriptions that can't be done in XML.
- Can use `XmlDetDescGen` Python module to assist.
- Basic AD and Pool structures generated (`XmlDetDescGen/AD,Pool/gen.py`)
- Don't mix generated subdirectories with hand-written ones.
- Generated XML includes hand-written XML via "href" or "external entity" mechanisms.

TDS Object Organization Basics

Each major section has a sub directory in the TDS under /dd

/dd/Materials/ :

```
/dd/Materials/Oxygen
/dd/Materials/Hydrogen
/dd/Materials/Water
/dd/Materials/WaterProperties/WaterAbsorptionLength
/dd/Materials/WaterProperties/WaterRefractionIndex
```

/dd/Geometry :

```
/dd/Geometry/Sites/lvNearSiteRock
/dd/Geometry/Pool/lvFarPoolIWS
/dd/Geometry/AD/lvOIL
/dd/Geometry/PMT/lvPmtHemiCathode
/dd/Geometry/AdPmts/lvAdPmtRing
```

/dd/Structure :

```
/dd/Structure/Sites/db-rock
/dd/Structure/Pool/far-iws
/dd/Structure/AD/la-ade2
```

Organization is by our own conventions.

Material Section

This section describes:

elements	A, Z, density
isotopes	A, Z, density, mass fractions
materials	density, temperature, pressure, component elements

Material XML

```
<element name="Oxygen" symbol="O"
            density="0.14300e-02*g/cm3" >
  <atom A="15.999*g/mole" Zeff="8.0000" />
</element>

<element name="Hydrogen" symbol="H"
            density="0.70800E-01*g/cm3">
  <atom A="1.00794*g/mole" Zeff="1.0"/>
</element>

<material name="Water" density="1.0000*g/cm3">
  <component name="Hydrogen" natoms="2"/>
  <component name="Oxygen" natoms="1"/>
  <tabprops
   address="/dd/Materials/WaterProperties/WaterAbsorptionLengt
  <tabprops
   address="/dd/Materials/WaterProperties/WaterRefractionIndex
</material>
```

Geometry Section

The Geometry section:

- Associates a shape and a material to a logical volume.
- Places "physical" daughter volumes inside mother.
- Assigns sensitive detector names to logical volumes.

Naming convention:

Logical Volumes "`lv`" prefix + ever more specific labels

- "`lvFarSiteRock`"
- "`lvAdPmtArray`"

Physical Volumes "`pv`" prefix + ever more specific labels

- "`pvLASiteRock`"
- "`pvNearADE2`"

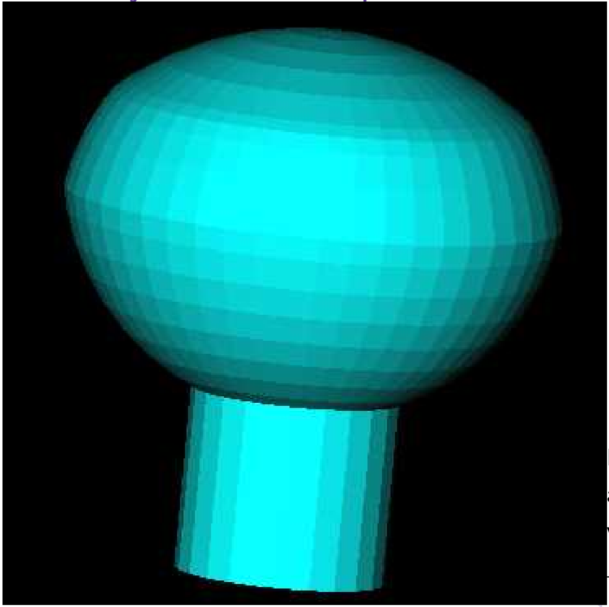
Geometry XML - Example 1

AD Oil cylinder logical volume with placed daughters: Outer Acrylic Vessel and PMT array.

```
<logvol name="lvOIL" material="MineralOil">
  <!-- the shape: -->
  <tubs name="oil" sizeZ="ADoilHeight"
             outerRadius="ADoilRadius" />

  <!-- any physical daughter volumes: -->
  <physvol name="pvOAV"
          logvol="/dd/Geometry/AD/lvOAV"/>
  <physvol name="pvAdPmtArray"
          logvol="/dd/Geometry/AdPmts/lvAdPmtArray"/>
</logvol>
```

Geometry XML - Example 2



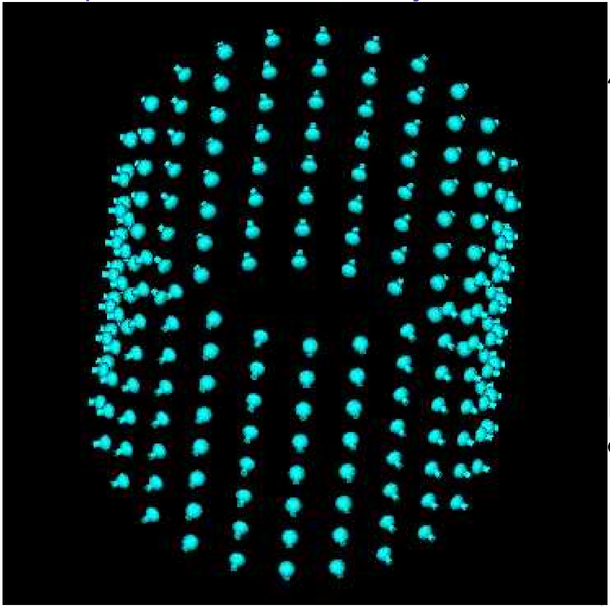
- Tak-Pui's model[a], union of three hemispheres
- Pyrex, photocathode and vacuum volumes
- Photocathode / sensitive detector connection given in XML
- Center @ face sphere's center

Note: polygons are just an artifact of this particular visualization program.

[a] DocDB #1596

```xml
<logvol name="lvPmtHemi" material="Pyrex">
  <union name="pmt-hemi">
    <intersection name="pmt-hemi-glass-bulb">
      <sphere name="pmt-hemi-face-glass"
              outerRadius="PmtHemiFaceROC"/>
      <sphere name="pmt-hemi-top-glass"
              outerRadius="PmtHemiBellyROC"/>
      <posXYZ z="PmtHemiFaceOff-PmtHemiBellyOff"/>
     <sphere name="pmt-hemi-bot-glass"
              outerRadius="PmtHemiBellyROC"/>
      <posXYZ z="PmtHemiFaceOff+PmtHemiBellyOff"/>
    </intersection>
    <tubs name="pmt-hemi-base"
          sizeZ="PmtHemiGlassBaseLength"
          outerRadius="PmtHemiGlassBaseRadius"/>
    <posXYZ z="-0.5*PmtHemiGlassBaseLength"/>
  </union>
  <physvol name="pvPmtHemiCathode"
          logvol="/dd/Geometry/PMT/lvPmtHemiCathode"/>
  <physvol name="pvPmtHemiVacuum"
          logvol="/dd/Geometry/PMT/lvPmtHemiVacuum"/>
</logvol>
```

Example 3: AD PMT Array



4 steps:

1. Position one PMT at bottom of tank
2. Copy to make ring of 24 PMTs
3. Copy ring 8 times
4. Rotate everything 1/2 angular period.

Result can be placed as one volume.

Building the AD PMT array with parameterized placement

```
<logvol name="lvAdPmtUnit">            <!-- step 1 -->
  <physvol name="pvAdPmtUnit" logvol="/dd/Geometry/PMT/lvPmtHemi">
    <posXYZ x="AdPmtRadialPos" z="-0.5*(AdPmtNrings-1)*AdPmtZsep"/>
    <rotXYZ rotY="-90*degree" />
  </physvol>
</logvol>

<logvol name="lvAdPmtRing">            <!-- step 2 -->
  <paramphysvol number="AdPmtNperRing">
    <physvol name="pvAdPmtInRing:1" logvol="/dd/Geometry/AdPmts/lvAdPmtUnit" />
    <posXYZ/>
    <rotXYZ rotZ="AdPmtAngularSep" />
  </paramphysvol>
</logvol>

<logvol name="lvAdPmtArrayZero">       <!-- step 3 -->
  <paramphysvol number="AdPmtNrings">
    <physvol name="pvAdPmtRingInCyl:1" logvol="/dd/Geometry/AdPmts/lvAdPmtRing"/>
    <posXYZ z="AdPmtZsep"/>
  </paramphysvol>
</logvol>

<logvol name="lvAdPmtArray">           <!-- step 4 -->
  <physvol name="pvAdPmtArray" logvol="/dd/Geometry/AdPmts/lvAdPmtArrayZero">
    <posXYZ/>
    <rotXYZ rotZ="0.5*AdPmtAngularSep"/>
  </physvol>
</logvol>
```

Fully parameter driven, no hard coded values!

Set initial copy numbers, eg “pvAdPmtInRing:1”.

Structure Section

The Structure section defines a hierarchy of "**Detector Elements**". They point out important **physical** volumes.

The hierarchy is built from:

- DetElem's logical volume name (TDS path under `/dd/Geometry`)
- Name of supporting DetElem (under `/dd/Structure`)
- Physical volume **trail** ("`npath`") starting from supporting DetElem's logical volume.

Structure used for:

- local volume coordinates for vertex and direction GenTools.
- specifying top level volumes for simulation.
- absolutely locating some volume.
- applying alignment offsets.

Structure XML Example

From `Pool/structure.xml`, defining Far Inner Water Shield (`far-iws`):

```
<catalog name="Pool"> <!-- /dd/Structure/Pool -->
  ...
  <detelem name="far-iws">
    <geometryinfo lvname="/dd/Geometry/Pool/lvFarPoolIWS"
                  npath="pvFarPoolIWS"
                  support="/dd/Structure/Pool/far-curtain" />
  </detelem>
```

From `AD/structure.xml`, defining Far AD Envelope #1 (`far-ade1`):

```
<catalog name="AD"> <!-- /dd/Structure/AD -->
  ...
  <detelem name="far-ade1">
    <geometryinfo lvname="/dd/Geometry/AD/lvADE"
                  npath="pvFarADE1"
                  support="/dd/Structure/Pool/far-iws" />
  </detelem>
```

Using Structure Objects in C++, Example

The GenTools position generator picks a point local to some physical volume and must convert it to global coordinates as this is what Geant4 requires.

```
Gaudi::XYZPoint local_point, global_point;
DetectorElement* detelem;
IGeometryInfo* gi;

gi = detelem->geometry();
global_point = gi->toGlobal(local_point);
```

See GenTools/GtPositioner class for real example.

Attaching UserParameters to Detector Elements

Arbitrary parameters can be attached to Detector Elements.

```
<detelem name="la-oil2">
  ...
  <userParameter name="vendorName" comment="Oil Producer">
    Johnson and Johnson
  </userParameter>
  <userParameter name="priceHistory" type="USD">
    2004  30.0
    2005  40.0
    2006  65.0
    2007  70.0
    2008 100.0
  </userParameter>
```

- User code can retrieve these parameters from C++ object.
- Not currently used by us.

Misc. Sections

Two sections used to define optical properties for detector simulation are:

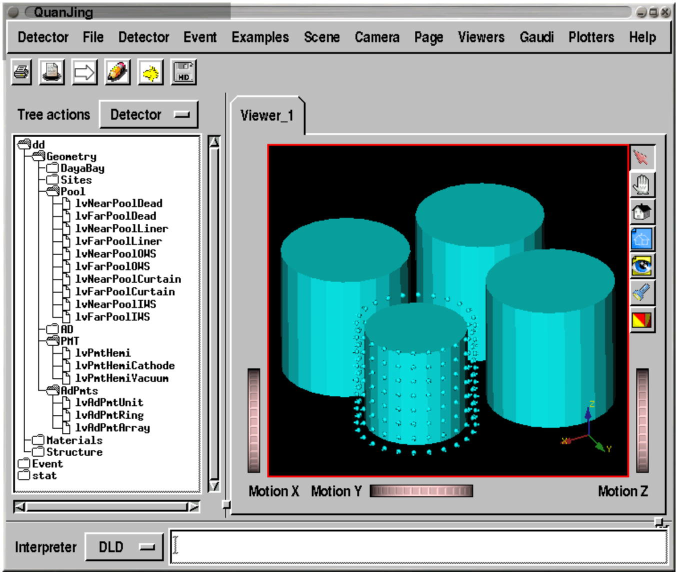
Surface	defines **skin** and **boundary** volume properties.
Tabproperty	defines tabulated values and are used to define volumetric material properties (eg ABSLENGTH, RINDEX).

Both are convertible to Geant4 equivalent.

Description Description Visualization

- Text based dumpers:
  `XmlDetDescChecks/python/dump_xmldetdesc.py`
- “X-ray” of densities:
  `XmlDetDescChecks/python/check_xmldetdesc.py`
- OpenInventor-based 3D viewer (next slide):
  `DetDescVis/python/quanjing.py`
- Geant4-based, still somewhat experimental:
  `DetSim/python/visdet.py`

Direct visualization: `quanjing.py`



- Inspect TDS (and TES)
- Rotate, pan, zoom
- Interrogate volumes
- Delete outer to see inner volumes
- Transparency & color effects
- GUI built from simple XML
- Easy to plug in user code
- Can call C++ or Python from interpreter
- Display event data

QuanJing - work needed.

From easiest to hardest

1. Originally stolen from LHCb's Panoramix
   → Need to rip out XML defining LHCb-specific menu entries
2. Need to add any useful Daya Bay specific functionality, eg:
   - Set different material colors
   - Add quick buttons to common volumes, scenes
3. Can use for 3D event display, eg:
   - PMTs sized/colored by time/charge
   - Highlight hit RPC pads.
   - Show reconstructed tracks and vertices.
4. Display simulated particle history information.

Anyone interested in one or more of these?

QuanJing Demo.

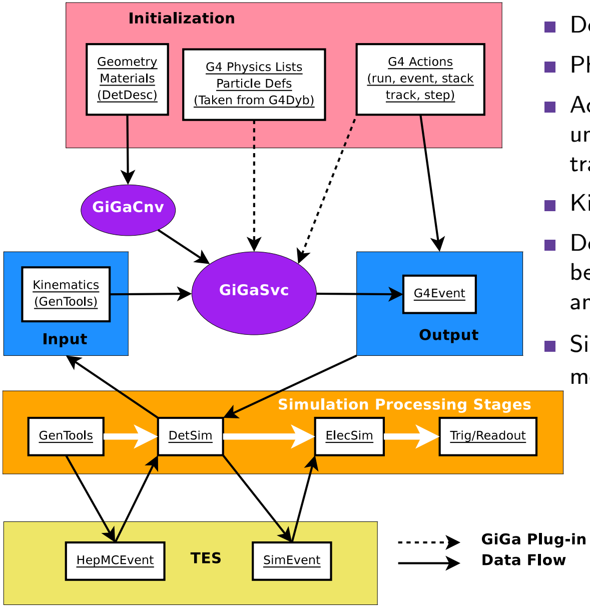
1 Modeling Daya Bay Detectors and Their Environment
- Overview of Detector Description
- Detector Description Sections
  - Material Section
  - Geometry Section
  - Structure Section
  - Misc. Sections
- Detector Description Visualization Methods

2 Simulating Detector Response to Particles
- Overview of Detector Simulation
- Configuration
- Simulated Hit Data
- Examples
- Caveats, a.k.a. To Do List

Detector Simulation Overview

- Monte Carlo integration method using **Geant4** to track individual particles.
- Runs in the **Gaudi** framework.
- Uses the **GiGa** package to organize Geant4 user code
- Uses 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


Initialization
Geometry Materials (DetDesc)
G4 Physics Lists Particle Defs (Taken from G4Dyb)
G4 Actions (run, event, stack track, step)
GiGaCnv
GiGaSvc
Kinematics (GenTools)
Input
G4Event
Output
Simulation Processing Stages
GenTools
DetSim
ElecSim
Trig/Readout
HepMCEvent
TES
SimEvent
GiGa Plug-in
Data Flow

- 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.

Configuring GiGa - basics

All example code derived from `DetSim/python/adgun.py` test script.

```python
from GaudiPython import AppMgr
app = AppMgr()
giga = app.service("GiGa")

app.TopAlg += [ "GaudiSequencer/SimSeq" ]
simseq = app.algorithm("SimSeq")
```

- Get Python handle on GiGa service for later.
- Create **sequencer algorithm** to hold all detector simulation related algorithms.

Configuring GiGa - Physics Lists.

GiGa lets you configure what physics to turn on or off and what energy cutoffs to use.

```
    modularPL = app.property("GiGa.GiGaPhysListModular")
    modularPL.CutForElectron = 100*units.micrometer
    modularPL.CutForPositron = 100*units.micrometer
    modularPL.CutForGamma = 1*units.millimeter
    modularPL.PhysicsConstructors = [
        "DsPhysConsGeneral",
        "DsPhysConsOptical"
        ]
    giga.PhysicsList = "GiGaPhysListModular"
```

- Set tracking cuts, physics and output level
- The physics constructors shown are taken directly from G4dyb's `dywPhysicsList`.
- The rest (EM, ElectroNu, Had, Ion) still need to be copied over. Volunteer?

Configuring GiGa - Geometry

```
    gggeo = app.service("GiGaGeo")
    gggeo.XsizeOfWorldVolume = 2.4*units.kilometer
    gggeo.YsizeOfWorldVolume = 2.4*units.kilometer
    gggeo.ZsizeOfWorldVolume = 2.4*units.kilometer

    simseq.Members = [ "GiGaInputStream/GGInStream" ]
    ggin = app.algorithm("GGInStream")
    ggin.ExecuteOnce = True
    ggin.ConversionSvcName = "GiGaGeo"
    ggin.DataProviderSvcName = "DetectorDataSvc"
    ggin.StreamItems = [ "/dd/Structure/Sites/la-rock", ]
```

- Configure world size big enough to hold experiment sites + reactors
- Setup DetDesc→G4 conversion and which top level physical volumes to include. Could also include everything with the very top level `/dd/Structure/DayaBay`.

Configuring GiGa - Add Some Actions

GiGa does for Geant4 Actions what Gaudi does for user Algorithms.
No longer need to hard-code actions as in G4dyb.

```
    event_ac_cmds = app.property("GiGa.GiGaEventActionCommand")
    event_ac_cmds.BeginOfEventCommands = [
        "/control/verbose 2",
        "/run/verbose 1",
        "/event/verbose 2",
        "/tracking/verbose 2",
        "/geometry/navigator/verbose 2"
        ]
    giga.EventAction = "GiGaEventActionCommand"
```

- This uses an action that applies Geant4 macro commands.
- You can also write and add your own Action classes.
- You can add multiple, separate Actions by using a `GiGa*ActionSequence`.
- We anticipate some standard Actions to handle collecting **particle histories** and **unobservable statistics** (Nathaniel).

Configuring DetSim - I/O Marshalling

This shows how to implement a simple processing model. It does not handle multiple event types properly ordered in time.[1]

```
simseq.Members += [ "DsPushKine/PushKine",
                    "DsPullEvent/PullEvent" ]
input = app.algorithm("PushKine")
input.Converter = "HepMCtoG4"
input.Location = "/Event/Gen/HepMCEvents"
output = app.algorithm("PullEvent")
output.Location = "/Event/Sim/SimHeader"
```

Employ two algorithms:

1. `DsPushKine` converts `HepMC::GenEvents` produced by **GenTools** and feeds resulting `G4PrimaryVertex` to GiGa.
2. `DsPullEvent` retrieves resulting `G4Event` from GiGa, converts to **SimEvent** data and stores it in the TES.

Each can set input/output TES locations. Defaults are shown.

[1]See Zhe's "15 minute" talk later.

SimEvent Data Objects

DetSim (and soon SimuAlg) 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")

- Only `SimHitHeader` and related hit data defined now.
- Data formats still changing.

Interlude - 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.

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

Examples

Example algorithm to histogram some simple hit quantities:

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

Script to drive GenTools, DetSim and histograms:

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

Caveats and Work Still To Do

- PMT: QE=100%. Variation over face and PMT-to-PMT variations not yet supported.
- No RPCs nor AD reflectors
- Need rest of physics lists and optical properties copied from G4dyb.
- Need SimHits data model finalized.
- Particle histories and unobservable statistics just started (Nathaniel).
- Improve volume→ID lookup code for for sensors to be less dependent on fine geometry details. (note added: this was solved during the workshop)
- Validation against SimuAlg/G4Dyb!!!
- Add geometry details that go beyond what are in G4dyb.
- Documentation: user and internals.