



RAT-PAC

(RAT is an Analysis Tool... Plus Additional Codes)

Andy Mastbaum

University of Chicago



THEIA Workshop
DESY Hamburg
23 March 2017

- What** A software package for both simulation and analysis with an emphasis on optical detectors
- How** Build on standards: GEANT4, customized GLG4Sim, and ROOT. C++ and occasional Python.
- Why** Wanted a framework atop Geant4 to handle event processing, data management, and make detector definition easier
- Who** Originally developed for Braidwood, now there are custom versions under development by SNO+, MiniCLEAN, DEAP, a few others. **RAT-PAC** is an open source version being developed by THEIA, WATCHMAN, ANNIE.

Code: <https://github.com/rat-pac/rat-pac>
Docs and Tutorials: <https://rat.readthedocs.io>

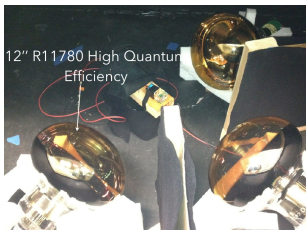
Goals

- ▶ Support current and future detector R&D by allowing quick changes to optics, simulation of test stands, etc.
- ▶ Generate realistic Monte Carlo samples of signals, backgrounds, and calibration sources.
- ▶ Provide a simple event-level analysis framework for MC and detector data, suitable for individual and production use.

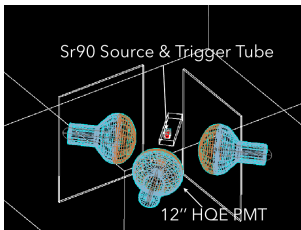
Principles

- ▶ “AMARA: As Microphysical As Reasonably Achievable.”
- ▶ Track each and every photon through a detailed detector geometry (photon thinning optional).
- ▶ Make simulation details like the full geometry model available to event analysis code like fitters.
- ▶ Stay generic to support different experiments with different geometries, DAQs, data structures, etc.
- ▶ Minimize hard-coded parameters to allow simple run-time configuration changes (less recompiling).

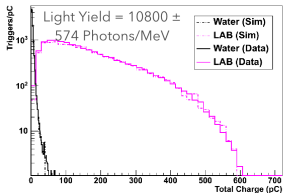
SNO+ Case Study¹



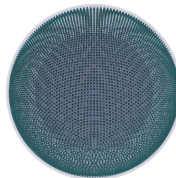
(a) Perform measurement



(b) Simulate in RAT



(c) Adjust parameter



(d) Simulate full detector

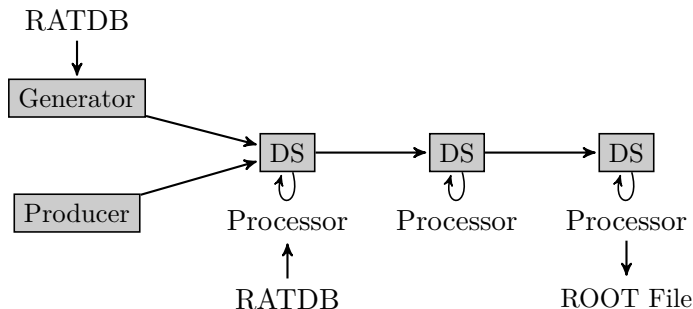
¹S. Grullon, Penn

The Simulation Process

1. Propagation of primary particles (Geant4)
2. Production of optical photons
 - ▶ Add Geant, GLG4Sim, custom processes
 - ▶ GLG4Scint, G4Scint, BNL WLS (L. Bignell, BNL)
3. Propagation of optical photons (absorption, reemission, scattering, boundary processes)
 - ▶ A component-scaling optical model is currently being developed by Richie Bonventre (Berkeley)
 - ▶ GPU-accelerated photon propagation in Chroma² is a possibility being investigated
4. Photon detection (photon tracking within the PMT geometry, hit charge and time model)
5. Digitization, triggering, and readout

²<http://chroma.bitbucket.org>

The Simulation Process



An event data structure is populated by a generator or producer, and operated on by processors.

An Example Job

1. Load any non-default DB tables, override parameters
2. Define processor chain
3. Define vertex generators
4. Run (rat macro.mac)

A macro:

```
/rat/db/set DETECTOR experiment "SNO"  
/rat/db/set DETECTOR geo_file "SNO/sno.geo"  
/rat/db/set GEO[target] material "wbls"  
/run/initialize  
/rat/proc frontend  
/rat/proc trigger  
/rat/proc calibrate  
/rat/proc awesomeFitter  
/rat/proc outroot  
/rat/procset file "some_electrons.root"  
/generator/add combo gun:fill:poisson  
/generator/vtx/set e- 0 0 0 1.0  
/generator/pos/set target  
/generator/rate/set 1  
/rat/run/start 100
```

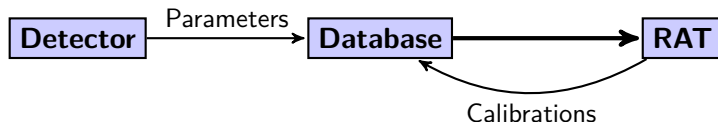
Now, let's look in more detail...

RATDB Database

Store and retrieve RAT parameters and detector constants, which can be set at run time.

```
/rat/db/set DETECTOR experiment "SNO"  
/rat/db/set DETECTOR geo_file "SNO/sno.geo"  
/rat/db/set GEO[target] material "wbls"
```

- ▶ Local text files
 - ▶ JSON format
 - ▶ Easy to use for studies or offline analysis
- ▶ CouchDB server support
 - ▶ Utilities to set up and push tables to server included



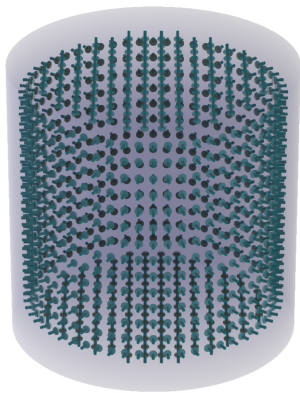
RATDB Database: Geometry

- ▶ Procedurally-defined nested solids – in between GDML and C++
- ▶ JSON format
- ▶ Shape builders defined in C++ (many included)

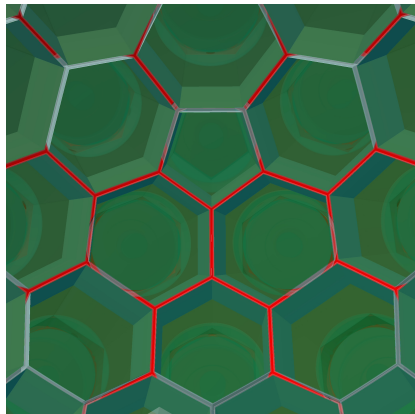
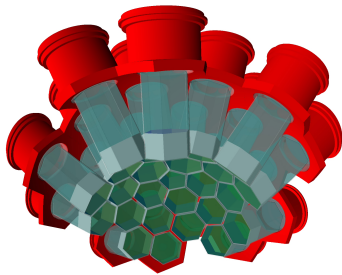
Several examples included in RAT-PAC

Simple: A Cylindrical Detector

- ▶ Build a library of parts defined in C++ to assemble detectors
- ▶ Place parts in a JSON geometry definition file
- ▶ This cylinder took ~ 30 lines, no C++, no recompiling
- ▶ Rapidly get new detectors (and new people!) up and running
- ▶ Quick and easy tweaks for testing



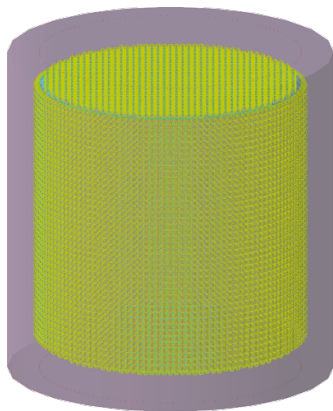
Excruciatingly Detailed: MiniCLEAN



T.Caldwell (Penn), J.Nikkel (RHUL), MiniCLEAN

Geometry

THEIA



THEIA_PARAMS.ratdb

```
{  
  name: "THEIA_PARAMS",  
  photocathode_coverage: 0.90,  
  veto_coverage: 0.00,  
  fiducial_diameter: 40000.0,  
  fiducial_height: 40000.0,  
  fiducial_buffer: 2000.0,  
  wall_thickness: 10.0,  
  veto_buffer: 2000.0,  
  veto_pmt_offset: 700.0,  
}
```

Parametric detector building
developed by Ben Land
(Berkeley)

RATDB Database: Materials, Optics, Etc.

Material composition and optical properties are also defined in database tables (which are also just JSON text files).

- ▶ Material and optical property tables map onto `G4MaterialPropertyVectors`
- ▶ Loaded at run time, so you can make changes without recompiling or from a macro command.
- ▶ PMT shape, timing, charge spectrum, and many other simulation aspects are in the DB.
- ▶ The goal is to have everything for which it makes sense run-time configurable.

Processors

Do something with the event data structure.

```
/rat/proc frontend  
/rat/proc trigger  
/rat/proc calibrate  
/rat/proc awesomeFitter  
/rat/proclast outroot  
/rat/procset file "some_electrons.root"
```

I/O ROOT files, network, other formats

DS Manipulation Pruning unneeded data to reduce file size

DAQ Front end, digitization, triggering, DAQ (i.e. more simulation)

Calibration Load constants from DB, apply calibrations

Analysis Reconstruction, analysis cuts, etc.

- ▶ Process each triggered event or see associated sets of triggered events (usually one MC event)
- ▶ Can be written in C++ or Python

Processors

A Few Examples:

outroot Write data structure to a ROOT file

outnet Send event to a remote RAT process

simpledaq A trivial DAQ/trigger simulation where all photoelectrons trigger a PMT hit

prune Trim off unwanted parts of data structure

fprompt Calculate fraction of hits in a prompt time window

calibratePMT Load and apply PMT calibration constants from the database (SNO+)

shellFit GPU-accelerated vertex fitter (MiniCLEAN)

Processors

Event Reconstruction

Currently Available:

Centroid Basic charge centroid position fitter

PathFitter SNO *xyzt* fitter for Cherenkov, ported to C++ by Ben Land (Berkeley)

Coming Soon:

Bonsai SuperK³ low-energy position + energy fit, being implemented by Marc Bergevin (LLNL)

This Session: **FitQun** (Mike Wilking), **3D topological reconstruction** (Björn Wonsak)

See Also: Slides from the October 2016 Mainz FroST meeting

³Michael Smy, ICRC07 30th International Cosmic Ray Conference

Vertex Generators

Make primary particles based on parameters

Top-level generators have ultimate control, may factor out into **vertex**, **position**, and **time** generators.

```
/generator/add combo gun:fill:poisson  
/generator/vtx/set e- 0 0 0 1.0  
/generator/pos/set target  
/generator/rate/set 1
```

Top-level Generators

combo V + P + T

decaychain Nuclear decay chains

coincidence Pile-up

solar Solar ν s

...

Vertex Generators

gun Single-particle gun

es Elastic scattering

pbomb Photon bomb

spectrum Energy spectrum

...

Vertex Generators

Position Generators

- point** All in one place
- fill** Fill a volume
- paint** Uniform on volume surface
- fillshell** Spherical shell within a volume
- line** Uniform on line segment

Time Generators

- uniform** Events separated by $t = 1/rate$
- poisson** Event rate is Poisson with $\mu = rate$

Producers

Instead of simulating, fill in the data structure from an external source

- ▶ Read in events from disk (saved MC or real detector data)
- ▶ Listen to events incoming from the network
- ▶ Unpack from other formats into the ROOT data structure
- ▶ Read GENIE vertices using **genie2rat** tool (R. Bonventre, Berkeley)

For example, **InROOTProducer** reads in a RAT ROOT file:

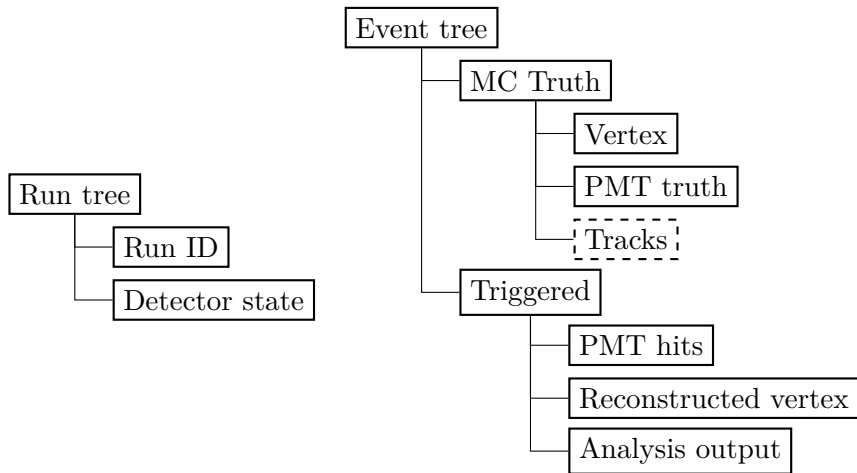
```
/rat/inroot/read filename.root
```

Processors don't care where events come from

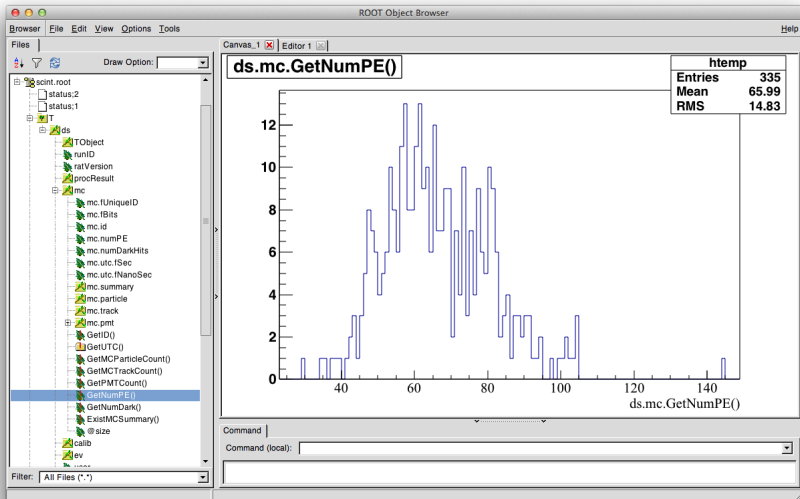
Hold all the information about the events

- ▶ ROOT TTree: store data in TObjects
 - ▶ Customized for experiments' needs
 - ▶ Physics-motivated hierarchy
- ▶ Use data in your standalone C++ or Python (PyROOT) code with `libRATEvent`
- ▶ Guideline: Use separate branches for MC, event-level data
 - ▶ Keep analysis processors from using MC truth
 - ▶ Perhaps not all MC events cause triggers
- ▶ Two trees: Event-level and run-level data

Data Structure Overview







```
root [0] T->Draw("mc.particle.ke");  
root [1] T->Draw("ev.pmt.charge", "ev.centroid.pos.Mag()<1000");
```



Testing with **rattest**

A high-level testing framework to make sure the physics doesn't change unintentionally

- ▶ Built-in functional testing framework eliminates boilerplate
 - ▶ You write a short macro and a ROOT script to produce histograms, **rattest** KS tests against a standard result
- ▶ CxxTest unit testing framework also included
- ▶ There is a web application (developed by MiniCLEAN and SNO+) to run full suite of tests on each commit and present results in a web interface

RAT Production Build Tests				
ID		Description	Tasks	Status
f6958320d85a5d233afbfb33e6ccf5506f2a1548		Merge pull request #975 from BenLand100/...	56	<div><div></div></div>
25ed4980890f9d5079a843b269884262c107b240		Merge pull request #973 from mjmottram/c...	28	<div><div></div></div>
daabb4f7656f4793bde0f27a1757d968e0d9f353		Merge pull request #974 from mjmottram/r...	28	<div><div></div></div>
b47fb13a8055e05a933fc154f9eeb97efb014a76		Merge pull request #966 from straitm/dox...	28	<div><div></div></div>

Getting the Code

Dependencies:

- ▶ Geant4 (10.01)
- ▶ ROOT 5
- ▶ SCons
- ▶ *Optional:* bzip2, cURL (for advanced database features)

Getting the code (public version):

- ▶ <https://github.com/rat-pac/rat-pac>

Compiling:

- ▶ `$./configure && source env.sh && scons`

Running:

- ▶ `$ rat mac/electron_demo_cylinder.mac`

Summary

RAT provides a quick “batteries included” path to a flexible, highly-detailed simulation for optical detectors

Microphysical Detailed microphysical simulation

Flexible Lightweight framework easily adapted to new experiments

Robust Built-in functional testing framework

ROOTful Automatic generation of ROOT dictionary and library for data structure

Scalable Supports a CouchDB database for detector/calibration constants

Community In use by several current and upcoming experiments

Documentation User guide at rat.rtdf.org

Questions?

(Future questions → mastbaum@hep.upenn.edu)

Development Model

- ▶ Originally developed for Braidwood
- ▶ Independent forks by MiniCLEAN, SNO+, DEAP, others
- ▶ Mid-2014: Open source RAT-PAC (MIT-style license)
- ▶ “PAC” refers to major components being ported in from BACCARAT and others (ongoing)
- ▶ Managed by an interest group which currently includes THEIA, WATCHMAN, and ANNIE.
- ▶ The code is available for anyone to use. It’s also possible to join in development; interested parties should contact Gabriel Orebi Gann (gorebigann@lbl.gov).

<https://github.com/rat-pac/rat-pac>