Theta13 detector
Geant4 based simulation package

Dmitry Onoprienko / onoprien@phys.ksu.edu
Kansas State University

April 30, 2004
Fermilab

Goals :

The simulation package should be :

- based on standard Geant4 functionality
- flexible and extendable
- interfaced to standard analysis tools

Limitations :

The simulation package is :

- only as good (and as fast) as Geant4 built-in physics
- needs Gd data – not a part of standard distribution
- inevitably generic – limited optimization

Give it a test drive :

www.phys.ksu.edu/~onoprien/theta13/geant

Version 0.1.01 (!!!) – limited functionality, no optimization.
Very little testing and validation has been done.

Simulation setup

Everything assembled in T13Setup class, also accessible through interactive commands (macro files can be used).

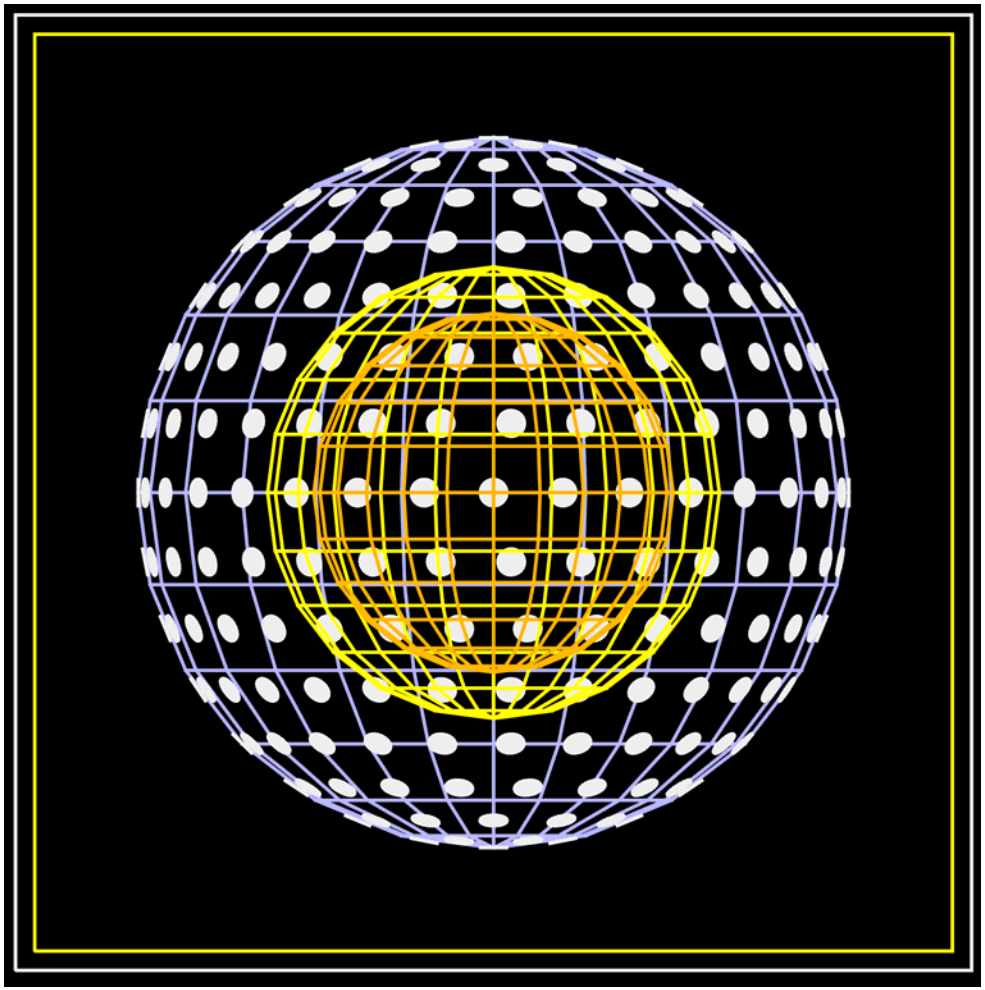
```
Idle> help
Command directory path : /
 Sub-directories :
 1) /control/   UI control commands.
 2) /units/   Available units.
 3) /geometry/   Geometry control commands.
 4) /tracking/   TrackingManager and SteppingManager control commands.
 5) /event/   EventManager control commands.
 6) /run/   Run control commands.
 7) /random/   Random number status control commands.
 8) /particle/   Particle control commands.
 9) /process/   Process Table control commands.
 10) /det/   Theta13: Detector geometry and materials control
 11) /gen/   Theta13: Primary particle generator control
 12) /read/   Theta13: Readout and Analysis control
 13) /visual/   Theta13: Visualization control
 14) /vis/   Visualization commands.
 15) /hits/   Sensitive detectors and Hits
 16) /gun/   Particle Gun control commands.

Type the number ( 0:end, -n:n level back ) : |
Command directory path : /det/

Theta13: Detector geometry and materials control

 Sub-directories :
 Commands :
 1) targetRadius * Target volume radius
 2) targetShellThickness * Target shell thickness
 3) catcherThickness * Gamma catcher thickness
 4) catcherShellThickness * Gamma catcher shell thickness
 5) bufferThickness * Buffer thickness
 6) bufferShellThickness * Buffer shell thickness
 7) pmtRequestedNumber * Requested number of PMTs looking at target volume
 8) pmtRadius * PMT radius
 9) vetoThickness * Veto volume thickness
 10) vetoShellThickness * PMT radius
 11) print * Print detector geometry and materials settings
```

Geometry



Actual number of PMTs is calculated automatically to allow for symmetric placement

No detailed PMT geometry is currently simulated

Scintillator parameters roughly match those of CHOOZ

No interactive control over geometry at the moment – hope to fix this soon.

Physics

- Only Geant4 built-in physics. Which is great but:
    - No molecules.
    - No non-linear dependence between energy deposition and the amount of scintillation light proguced
- Default: “Standard” EM physics, QGSP_HP hadrons, optical physics.
- No spectral properties for optical photons – “effective” values are used for attenuation length, photocathode quantum efficiency, etc. No wavelength shifting in scintillator.
- Use Geant4.6.1 (not 4.6.0.xx) - at least if you use standard EM physics.

Primary particle generation

Control interactively, through macro file, or by editing *T13Setup.cc* (no error checking in this case).

```
class T13GeneratorModel {

  public:

    T13GeneratorModel(G4String modelName);
    T13GeneratorModel();
    virtual ~T13GeneratorModel();

  public :

    virtual G4String getName();

    virtual G4bool providesPosition();
    virtual G4bool providesDirection();
    virtual G4bool providesEnergy();

    virtual G4bool providesEnergyDistribution();

    virtual G4ThreeVector getPosition();
    virtual G4ThreeVector getDirection();
    virtual G4double getEnergy();

    virtual G4double getEnergyPDF(G4double energy);
    virtual G4double getMinEnergy();
    virtual G4double getMaxEnergy();

  protected:

    G4String name;

};
```

Accepts any particle known by Geant4. $\tilde{\nu}_e$ is automatically replaced by neutron-positron pair, taking into account inverse beta decay differential cross section.

Position, direction, energy – either given explicitly or supplied by a model.

Currently implemented models – “RandomUniform”, “NeutrinoSpectrum”. Easily extendable by subclassing *T13GeneratorModel*

Primary particle generation – Volume Mask

Allows to confine primary vertex generation to certain volumes, and control relative “per unit volume” rates.

Accepts any physical volume name :

“target” – target volume filled with Gd-loaded scintillator

"targetShell" - Acrylic vessel.

"catcher" - Gamma catcher filled with plain scintillator.

"catcherShell" - Acrylic vessel.

"buffer" - Buffer volume filled with non-scintillating mineral oil

"bufferShell" - Stainless steel vessel.

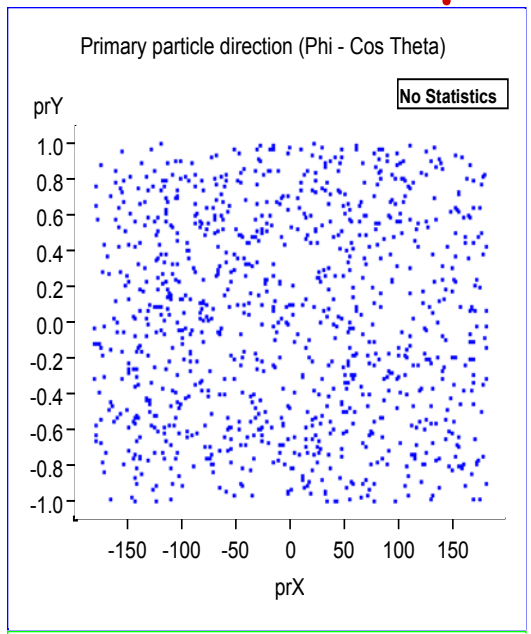
“veto” – water filled volume surrounding buffer shell

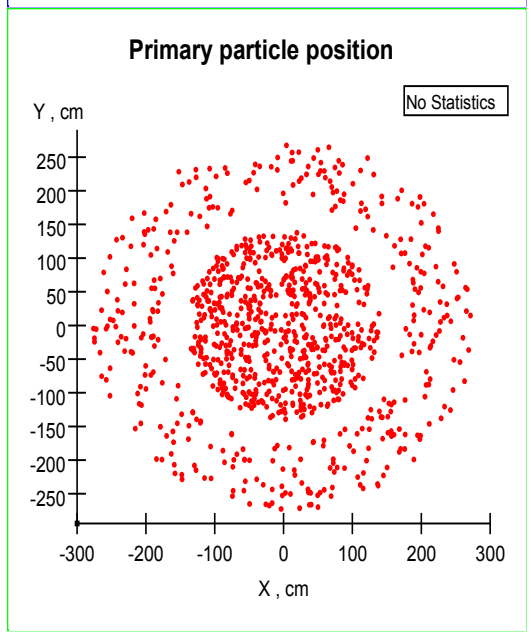
“vetoShell” – stainless steel

Primary particle generation – Example 1

```
$ theta13
……................
……................

Idle> /read/runName example_1
Idle> /read/plotterOn
Idle>
Idle> /gen/particle geantino
Idle> /gen/directionCode RandomUniform
Idle>
Idle> /gen/positionCode RandomUniform
Idle> /gen/vmReset
Idle> /gen/vmAddVolume target 4
Idle> /gen/vmAddVolume buffer 1
Idle> /gen/vmActive
Idle>
Idle> /gen/kineticEnergy 5 MeV
Idle>
Idle> /run/beamOn 1000
```

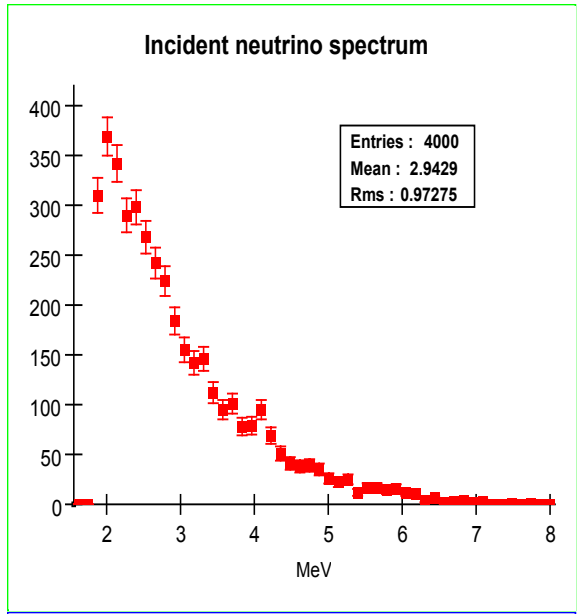


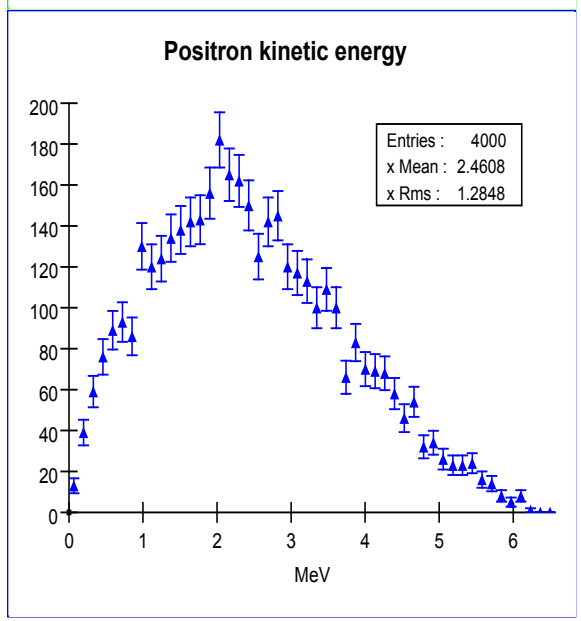


Primary particle generation – Example 2

```
$ theta13
……................
……................

Idle> /read/runName example_2
Idle> /read/plotterOn
Idle>
Idle> /gen/particle anti_nu_e
Idle> /gen/directionCode RandomUniform
Idle> /gen/position 12. 10. -20. cm
Idle>
Idle> /gen/energyCode NeutrinoSpectrum
Idle>
Idle> /run/beamOn 1000
```





Detector response

Dummy implementation now – need to write something meaningful, hopefully with better understanding of what studies the package is going to be used for…

By default, keep trajectories for primary particles, with an option of storing trajectories for everything except optical photons.

Hit Collections (time-sliced) :

- PMT
- Buffer shell
- Veto PMT
- Veto shell

No realistic PMT response at the moment.

Convenience class for user analysis / data storage- T13UserAction :

```
class T13UserAction : public T13UserActionBase {

  public:  // --- Constructors & Destructors :  --------

    T13UserAction();
    virtual ~T13UserAction();

  public:  // --- Implementing T13UserActionBase :  ----

    virtual void beginRun();
    virtual void beginEvent();
    virtual void endEvent();
    virtual void endRun();

  //  --- Add your stuff here :  -----------------------
```

Analysis / Persistency

Set G4USE_ANALYSIS to activate AIDA based analysis interface.

AIDA - Abstract Interfaces for Data Analysis (http://aida.freehep.org/)

JAIDA – Simple Java implementation (http://java.freehep.org/jaida/index.html)
(or use OpenScientist, Anaphe, etc.)

AIDAJNI – bridge for calling JAIDA from C++
(http://java.freehep.org/aidajni/index.html)

/read/plotterOn true command activates real time plotter

/read/runName command sets the name for AIDA file (compressed XML) to be saved at the end of the run.

JAS3 – interactive analysis (http://jas.freehep.org/jas3/index.html)

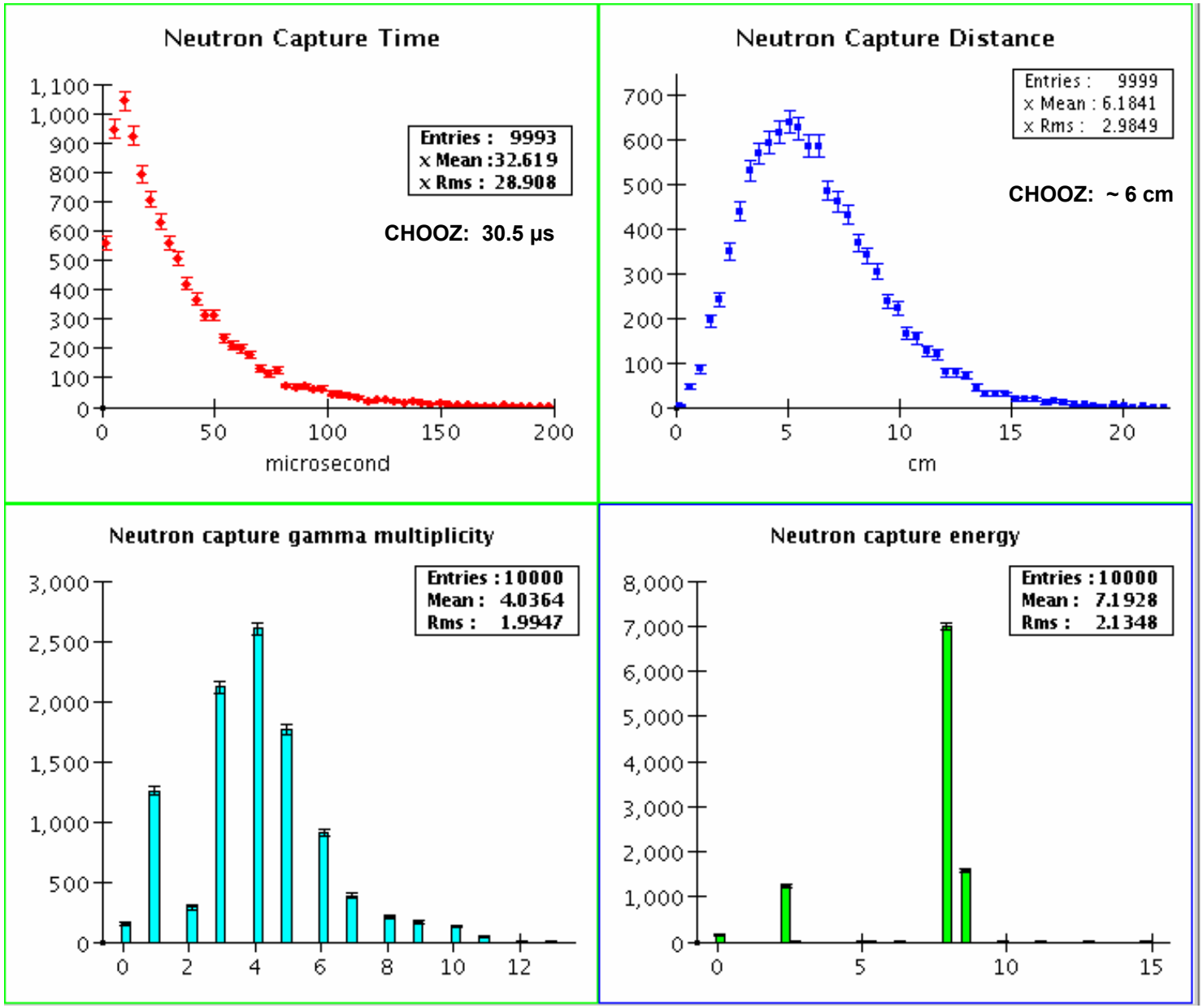
No persistency mechanism beyond that – do we need it ?

… that's what's available.

Now does it work ?

Can we trust the results ?

Neutron Capture



Capture on Gd : 85.4 %

CHOOZ : 84.1 %

More on reliability of results – or "A few words about Geant4"

(Almost) complete physics toolkit for neutrino experiment simulation – huge progress compared to Geant3.

Having it (almost) all in C++ does pay off.

Actively supported.

But …

Work in progress – bugs, untested / incomplete functionality.

No fixed maintained version – no distinction between bug fix releases and new functionality / new implementation releases.

Just starting to validate new releases with respect to earlier ones.

→ At the moment, maintaining and validating Geant4 based software – even simple simulation code like this one – requires considerable time investment.