HMB Simulations with GEANT4 & Visualization in ROOT

Harsh Purwar

Jul 19th, 2021 – Coffee and Techniques (CaT)
Introduction to Geant4

- **Geant4** → **Geometry And Tracking**, a platform (toolkit) for simulation of the passage of particles through matter using Monte Carlo methods.
- Last stable release: ver. 10.6 10.7
- Developed in C++ by Geant4 Collaboration
- Documentation: [https://geant4.web.cern.ch/support/user_documentation](https://geant4.web.cern.ch/support/user_documentation)
- Provides a complete range of functionality including tracking, geometry, physics models and hits.
- The toolkit can be installed on Windows, Mac OS and various Linux distributions (although, only CentOS 7 is officially supported).
- Virtual machines with Geant4 pre-installed are also available: [https://geant4.cenbg.in2p3.fr/](https://geant4.cenbg.in2p3.fr/)
  Installation instructions: [https://indico.cern.ch/event/1014059/page/21901-geant4-virtual-machine](https://indico.cern.ch/event/1014059/page/21901-geant4-virtual-machine)
- Detailed Geant4 installation instructions including dependencies for Ubuntu 20.04 are on HMB GitHub.
- Geant4 article (from 2003): [http://doi.org/10.1016/S0168-9002(03)01368-8](http://doi.org/10.1016/S0168-9002(03)01368-8)
  Getting started tutorial: [https://indico.cern.ch/event/1014059/](https://indico.cern.ch/event/1014059/)
HMB Simplified Tracker Geometry

- Started with a very simple tracker geometry:
  - 75 simple (no cladding) optical fibres stacked together (double layers)
  - Two orthogonal double-layered trackers (for x & y co-ordinates)
  - HMB will have 2 such x/y trackers (not added here)

- Material: G4_SILICON_DIOXIDE (not correct!)
Particle source: 4 GeV muons ($\mu^+$, $\mu^-$)
Incident with a moving particle gun
GEANT4 Project File/Class Structure

- Run Manager — To manage the overall simulation, how detector is constructed, what physics to include, what needs to be calculated, etc.
  - Geometry – Detector Construction class
  - User Physics list class/Reference Physics list
  - **Primary Generator Action class** or macros – to define incident particles
  - User Run Action class
  - User Event Action class
  - User Stepping Action class
  - User Tracking Action class (missing!)
  - Visualizations, etc. – via macros

```cpp
RunManager
{
    DetectorConstruction{}
    PhysicsList{}
    RunAction
    {
        PrimaryGeneratorAction{}
        EventAction
        {
            SteppingAction{}
            TrackingAction{}
        }
    }
}
```
Simplified Geometry Construction Class

GitHub: geant4/test1/src/B1DetectorConstruction.cc

// World (HMB Box)
G4double world_hX = 121*cm, world_hY = 131*cm, world_hZ = 121*cm;
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
G4Box* solidWorld =
    new G4Box("World", 0.5*world_hX, 0.5*world_hY, 0.5*world_hZ);
G4LogicalVolume* logicWorld =
    new G4LogicalVolume(solidWorld, world_mat, "World");
G4PVPhysicalVolume* physWorld =
    new G4PVPlacement(0, G4ThreeVector(), logicWorld, "World", false, 0, checkOverlaps);

// Fiber
G4Material* fiber_mat = nist ->
FindOrBuildMaterial("G4_SILICON_DIOXIDE");
G4double iRad = 0.*cm, oRad = 0.1*cm, hz = 15.*cm;
G4double stAng = 0.*deg, spanAng = 360.*deg;
G4Tubs* fiberTube =
    new G4Tubs("Fiber", iRad, oRad, 0.5*hz, stAng, spanAng);

G4LogicalVolume* logicFiber =
    new G4LogicalVolume(fiberTube, fiber_mat, "Fiber");

G4double scint_sep = 11.2*cm; // Sep. b/w two pairs of a tracker
G4double fiberLayerDist = 0.438*cm; // vertical sep. b/w layers

// Bottom pair of the fiber set
G4double xoff = -4.725*cm;
for(int i=0; i<38; i++) {
    new G4PVPlacement(0, G4ThreeVector(xoff+i*0.25*cm, -scint_sep/2.0 + fiberLayerDist/2.0, 0*cm), logicFiber, "Fiber", logicWorld, false, i, checkOverlaps); }
int k=0;

for(int i=38; i<75; i++) {
    new G4PVPlacement(0, G4ThreeVector(xoff+0.125*cm + k*0.25*cm, -scint_sep/2.0-fiberLayerDist/2.0, 0*cm), logicFiber, "Fiber", logicWorld, false, i, checkOverlaps);
k++;
}

// Top pair of the fiber set

GDML
Geometry Description Markup Language
Source of Muons – CRY library

• Muon source in reality: Cosmic ray muons – more details in Gary’s talk: [HMB Cosmic Ray Overview](#)

• For HMB simulations, we are using the CRY library – Cosmic-RaY Shower Library

• Developed at Lawrence Liverpool National Laboratory ~2012

• Generates correlated cosmic-ray particle showers at different elevations for use as input to transport and detector simulation codes. Independent from GEANT4.

• Provides all particles ($\mu, n, p, e, \gamma, \pi, k$) within a specified area as well as time of arrival and zenith angle* of secondary particles.

• Provides basic correlations between particles within the shower, latitude, and solar-cycle variations.

• Fast simulation based on precomputed input tables derived from full MCNPX simulations of primary cosmic rays (1 GeV to 100 TeV primary particles) on full atmosphere model.

• Function library callable from C, C++, Fortran, and now also Python.

*Zenith angle: $\cos^{-1}(-v_z)$

$v_z$ is the z-component of the particle velocity.
Generation of Particles with CRY Library

CRY: Cosmic-ray Shower Library (v1.7)

~/cry/test/testOut.cc setup.file 100000

Output file → shower.out:

<table>
<thead>
<tr>
<th>#</th>
<th>nEvent</th>
<th>nSecondary</th>
<th>KE</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>u</th>
<th>v</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1183</td>
<td>-0.35553</td>
<td>-0.17427</td>
<td>0</td>
<td>0.21387</td>
<td>0.5026</td>
<td>-0.83764</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>6618</td>
<td>-0.74499</td>
<td>-0.3886</td>
<td>0</td>
<td>-0.3633</td>
<td>0.04763</td>
<td>-0.93046</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>6580.5</td>
<td>0.68209</td>
<td>-0.18992</td>
<td>0</td>
<td>-0.27037</td>
<td>-0.38348</td>
<td>-0.88309</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>4150</td>
<td>0</td>
<td>-0.19935</td>
<td>-0.59347</td>
<td>0</td>
<td>0.22575</td>
<td>-0.32956</td>
</tr>
</tbody>
</table>

Altitude: 0 m, 2.1 km, 11.3 km
Generation of Particles with CRY Library

CRY: Cosmic-ray Shower Library (v1.7)

~/cry/test/testOut.cc

Output file → shower.out:

- returnNeutrons: 0
- returnProtons: 0
- returnGammas: 0
- returnElectrons: 0
- returnMuons: 1
- returnPions: 0
- returnKaons: 0

Date: 7-19-2021
Latitude: 21.3
Altitude: 0 m, 2.1 km, 11.3 km

Subbox Length: 1.5

Monday, July 19, 2021
### CRY Data Files

```c
#include <stdio.h>

int main() {
    // Primary binning
    double primary_bins[] = {
        1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000, 12000, 13000, 14000, 15000, 25118.9, 39818.7, 63095.7, 100000, 158490, 251189, 398170, 630957, 1000000,
        1.58490e+06, 2.51189e+06, 3.50170e+06, 6.05976e+06, 1.00000e+07, 1.58490e+07, 3.98170e+07, 6.30957e+07, 1.00000e+08
    };
    // Secondary binning
    double secondary_bins[] = {
        1.58490391924611e-09, 2.51188643159959e-09, 3.98170170553499e-09, 6.30957344488195e-09,
        1.0e-08, 1.58490391924611e-08, 2.51188643159959e-08, 3.98170170553499e-08, 6.30957344488195e-08,
        1.0e-07, 1.58490391924611e-07, 2.51188643159959e-07, 3.98170170553499e-07, 6.30957344488195e-07,
        1.0e-06, 1.58490391924611e-06, 2.51188643159959e-06, 3.98170170553499e-06, 6.30957344488195e-06,
        1.0e-05, 1.58490391924611e-05, 2.51188643159959e-05, 3.98170170553499e-05, 6.30957344488195e-05,
        0.0000000001, 0.000158490391924611, 0.000251188643159959, 0.000398170170553499, 0.000630957344488195,
        0.001, 0.00158490391924611, 0.00251188643159959, 0.00398170170553499, 0.00630957344488195,
        0.01, 0.0158490391924611, 0.0251188643159959, 0.0398170170553499, 0.0630957344488195,
        0.1, 0.158490391924611, 0.251188643159959, 0.398170170553499, 0.630957344488195,
        1.0, 1.58490391924611, 2.51188643159959, 3.98170170553499, 6.30957344488195,
        10, 10.58490391924611, 25.1188643159959, 39.8170170553499, 63.0957344488195,
        100, 150.8490391924611, 251.188643159959, 398.170170553499, 630.957344488195,
        1000, 1000.0000000001, 1508.490391924611, 2511.88643159959, 3981.70170553499, 6309.57344488195,
        10000, 10000.0000000001, 15084.90391924611, 25118.8643159959, 39817.0170553499, 63095.7344488195,
        100000, 100000.0000000001, 150849.0391924611, 251188.8643159959, 398170.170553499, 630957.344488195,
        1000000, 1000000.0000000001, 1508493.91924611, 2511888.864315996, 3981701.70553499, 6309573.444488196
    };
    return 0;
}
```
Organization of Particle PDFs in CRY Data File

- Primary cosmic ray proton energies (33 primary bins)
- 85 secondary bins (sBins)

Increasing sBins require interpolation of:
- Energy distribution
- Time distribution
- $\cos \theta$ distribution
- Charge distribution

For all secondary particles

---

<table>
<thead>
<tr>
<th>$E_p$ (MeV)</th>
<th>1500</th>
<th>2500</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.14 x 10^7</td>
<td>8.15 x 10^7</td>
</tr>
</tbody>
</table>

---

Monday, July 19, 2021
Muon Energy Distribution
(After interpolation of all distributions in CRY data files)

- Linear interpolation was done using MATLAB
- Only CRY data file for zero altitude was modified – modification straightforward for other altitudes
- Known bug: Run crashes if neutrons are requested, works well for all other particles
- May be a bug in the interpolated data – under investigation

1M muons
4x more points in KE dist.
CRY muons $\rightarrow$ GEANT4 simulation

subboxLength = 0.5 m

subboxLength = 2.0 m
Primary Generator Action/Messenger Classes

- CRY Library is integrated with GEANT4 simulation with the help of these two classes
- Reads CRY settings from an input file or from macro
- Reads particle distribution data from CRY data file
- Generate particles based on these distributions
- Supply each particle’s energy, momentum, direction, time info, etc. to the ParticleGun class object
- Each generated particle constitutes an event and EventAction class is called.

For later:
- Do we need to revise the definition of an event? Lost/scattered $\mu$ or $\gamma$, should they be counted as an event?
- Perhaps, only consider particles that enter the HMB box as an event?
Additions/Improvements to HMB Geometry
...since test1

- Scintillating fiber geometry and material
- Added 2 sets of Trackers (4 double layer trackers)
- Added G4_CONCRETE ceilings & roof to simulate correct KE dist. of incident muons
- Added 2 layers of Sodium Iodide (NaI) calorimeters
- Re-checked sizes & distances between each element

Yet to do:
- Add Aluminum housing on calorimeters
- Re-bars inside concrete ceiling and roof
- Correct material for the HMB box (G4_Air)
- Add more elements inside HMB box
Any questions, so far…

✓ Introduction to Geant4
✓ “Hello world” version of the G4 HMB simulation
✓ How to build Geometry in G4, define material, properties, etc.
✓ GDML file format → a compatible format – G4, ROOT, DD4hep, event display?
✓ Data from improved CRY Library
✓ How to integrate CRY with G4
✓ HMB simulation geometry updates

Next: G4 Physics…
GEANT4 Physics

• GEANT4 provides a wide variety of physics components for use in simulation
• Physics components are coded as processes:
  • a process is a class which tells a particle how to interact
  • GEANT4 provides many of these
  • users may write their own, but it must be derived from a Geant4 process
• Processes are classified as: electromagnetic, hadronic, decay, parameterized or transportation

• Electromagnetic (EM) standard
  • complete set of processes covering charged particles & gammas (1 keV to ~PeV)
  • Low energy (250 eV to ~GeV)
  • Long wavelength optical photons (x-rays, UV, visible)

• Hadronic
  • Pure hadronic (0 to ~TeV)
    • Elastic, inelastic, capture, fission
  • Radioactive decay – at rest and in-flight
  • lepto-nuclear (~10 MeV to ~TeV) – μ induced nuclear reactions

• Decay – weak & electromagnetic decays
• Transportation – only process to move particles through the geometry
Physics Lists in GEANT4

- **Physics List**: A class which collects physics processes and production thresholds for all particles needed for your application.
- It tells the run manager how and when to invoke what physics.
- For most applications, no need to write your own physics list from scratch.
- GEANT4 provides many reference physics lists tailored to specific use cases – regularly tested and validated by GEANT4.
- Commonly used reference physics lists: FTFP_BERT, QGSP_BIC, etc.
- For HEP calorimetry, tracker, and air shower simulations: FTFP_BERT is the recommended physics list.
- It is possible to adapt the reference list to our simulation: change thresholds, apply selection cuts, include/exclude certain physics processes, etc.

<table>
<thead>
<tr>
<th>FTF: Fritiof string model</th>
<th>QGS: Quark Gluon String model</th>
<th>BERT: Bertini-style cascade</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCL: Liege cascade</td>
<td>BIC: Binary Cascade</td>
<td>HP: High Precision neutron model</td>
</tr>
<tr>
<td>P: G4Precompund model used for de-excitation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ref.: [https://geant4.web.cern.ch/node/155](https://geant4.web.cern.ch/node/155)
How GEANT4 decides what processes occur & when?

Stepping Action class

- SteppingAction class is responsible for this
- Interaction lengths for all included physics processes are sampled
- Requires:
  - cross sections (from CRY Library for HMB)
  - decay lifetimes (pre-defined, I think!)
- Parameters like energy deposited, interaction point, etc. can be calculated on a boundary crossing from within this class

- Step 1:
  - all lengths sampled
  - Compton occurs
- Step 2:
  - Compton re-sampled
  - boundary is crossed
- Step 3:
  - Compton occurs again
  - new boundary found
- Step 4:
  - Compton re-sampled
  - pair production occurs
Introduction to ROOT

• **ROOT** is a framework for data processing. It enables statistically sound scientific analyses and visualization of large amounts of data.

• Last stable release: v6.24/02

• Developed at CERN by several contributors

• Documentation: [https://root.cern/manual/](https://root.cern/manual/)

• Extensively used in High Energy Physics for data processing and visualization.

• Works out of the box on Windows, Mac OS and various Linux distributions.

• ROOT installation instructions for Ubuntu 20.04 are on HMB GitHub. Other details: [https://root.cern/install/](https://root.cern/install/)

• Comes with an incredible C++ interpreter, ideal for fast prototyping. Integrates super-smoothly with Python, can also be used in a Jupyter notebook!
HMB Data Output → ROOT File Structure

- **HMB**
  - Event ID \( (eventID) \) [1, 2, 3,...]
  - If is a Hit \( (isHit) \) [0, 1]
  - Muon initial energy
  - Muon initial momentum
  - Particle energy \( (hitMuonEnergyMeV) \) [MeV] → Before hitting the calorimeters
  - Hit momentum \( (hitMomentum) \) [MeV/c] → Momentum just before hitting the calorimeters
  - Charge of the Particle \( (charge) \) [e]
  - Total energy deposited \( (eDep_J) \) [Joule]
  - Non-ionizing energy deposited \( (NIeDep_J) \) [Joule]
  - Hit region \( (hitRegion) \) [1, 2, or 3] → hit on top layer, bottom layer, or hit on both layers
  - Time since the last hit \( (lapTime) \) [second]
  - Hit position \( (posX, posY, posZ) \) [mm]

```cpp
root -l hmbData.root
HMB->Draw("eDep_J>>(100,1,200)", "isHit==1 && hitRegion==3")
```

Tracks
Using FTFP_BERT Reference Physics List

No treatment of optical photons

10M particle simulation \( (y = 50 \text{ cm}) \):

- Out of 10M incident Muons, 14227 Muons hit NaI slabs – 0.14%
- Avg. Muon incident rate: 2172 hits per hour (~36 hits/min on NaI slab)
- Total energy deposited: \( 411.96 \pm 4.94 \text{ kJ} \) in \(~6.5\) hours assuming above hit rate (~17.5 J/s or ~1.0 kJ/min)

\[
\begin{align*}
\text{Color codes:} \\
\mu^+ & : \text{blue} \\
\mu^- & : \text{red} \\
\gamma & : \text{green} \\
e^+ & : \text{yellow} \\
e^- & : \text{orange} \\
\text{Others: grey}
\end{align*}
\]
More plots...

GitHub: geant4/plotTools/plot.cc
Geometry Visualization in ROOT

- Import HMB GEANT4 geometry in ROOT and then layover the Muon tracks

- Export GEANT4 geometry to a GDML file using GDMLParser included in GEANT4

```cpp
auto detC = new B1DetectorConstruction();
G4VPhysicalVolume* phyVol = detC->Construct();
G4GDMLParser fParser;
fParser.SetOutputFileOverwrite(true);
fParser.Write("hmbGeo.gdml", phyVol);
runManager->SetUserInitialization(detC);
```

- Import geometry back in ROOT for visualization

```cpp
TGeoManager *geom = TGeoManager::Import("hmbGeo.gdml")
geom->GetTopVolume()->Draw("ogl")
```

- Working on exporting GEANT4 tracks in ROOT compatible format.
Yet to Implement (To do list)

- Add UserTrackingAction and SensitiveDetector classes to get complete muon tracks
- Write event-wise track information to ROOT
- Check simulation compatibility with DD4hep (particle source, multi-threading)
- Improve geometry & add more elements
- Improve physics: apply selection cuts, etc.
- More plots and visualizations
Mahalo for your time and attention.

Questions...?

- Harsh Purwar
Email: purwar@hawaii.edu