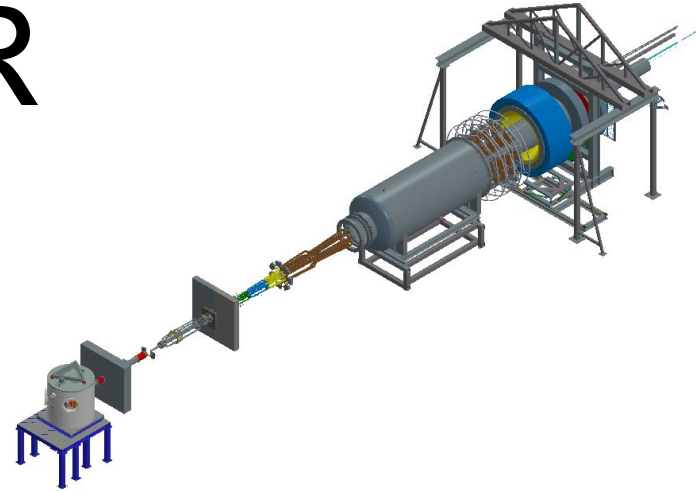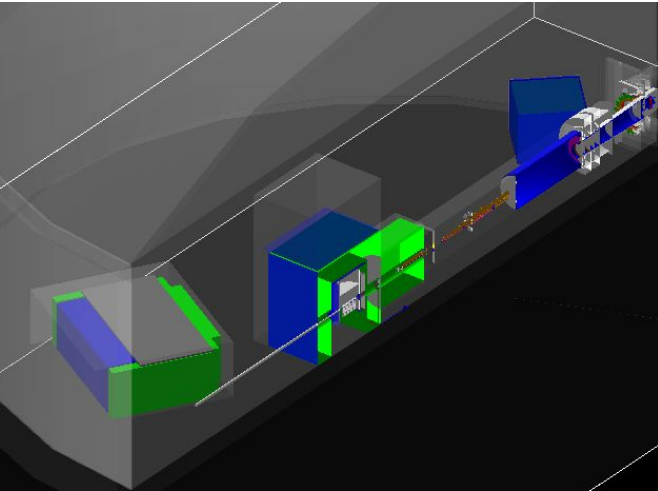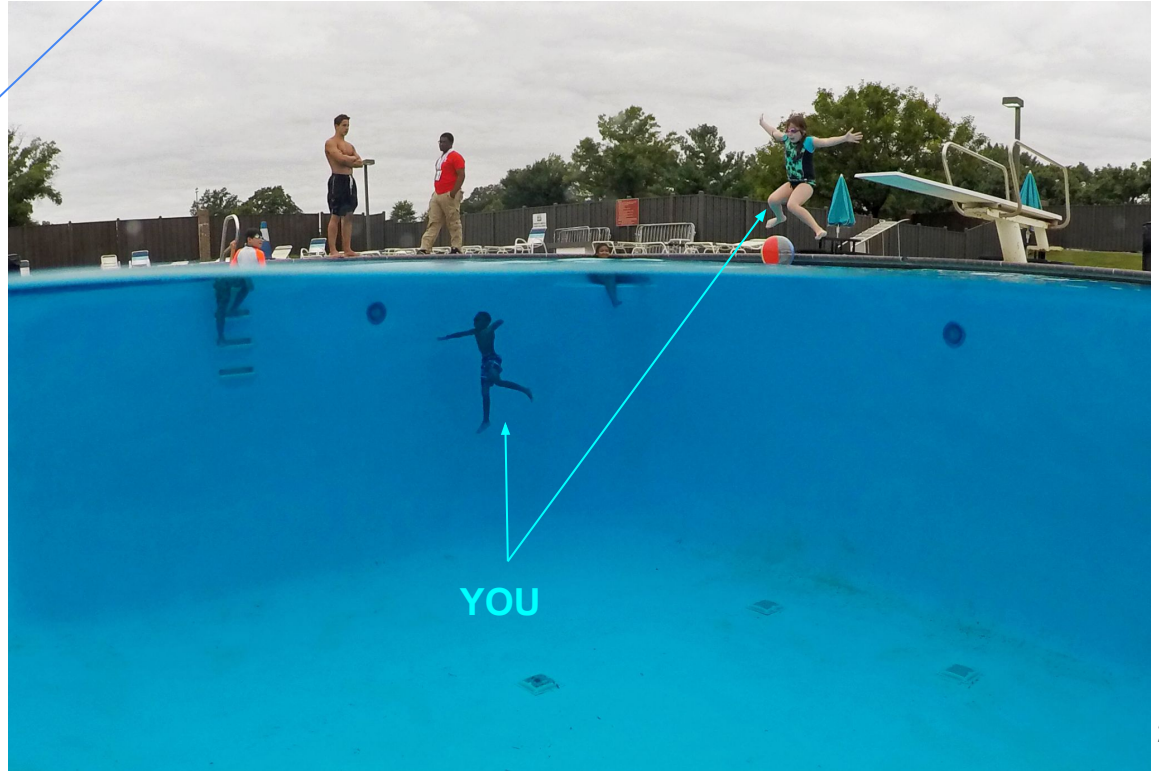# Geant4 Simulations for MOLLER

Rakitha Beminiwattha
Louisiana Tech

What is the Goal of this Workshop
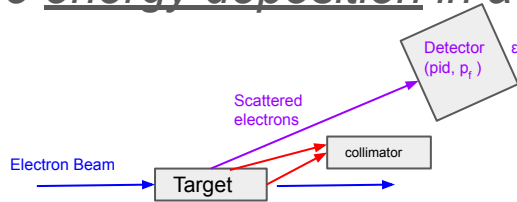
To not drown yourself

YOU

What is Monte Carlo (MC) Simulation

Evaluate Complicated <u>Integrals</u> such as:

- *Average <u>rate</u> of particles reaching a detector*
- *Average <u>energy deposition</u> in a detector*
- *Average <u>energy deposition</u> in a collimator or structural element*

<u>The integrals is defined as</u>

phase space of initial states (pid, p_0) with cross section (σ) multiplied by an acceptance function of the detector ϵ(pid, p_f)

3

What is our initial state (pid, p_0)?

1. Unbiased "Beam generator" beam electrons upstream of target
   a. Each event is a beam electron and **all events are equally weighted**
2. Biased "Physics generator": a selected physics processes with calculation of σ and A
   a. A weight is assigned to each event: integrated luminosity (L)✕ differential cross section (dσ/dΩ) ✕ phase space factor (ΔΩ)
   b. Variable `rate` in simulation output assign an event rate for the simulated beam current for the selected physics processes
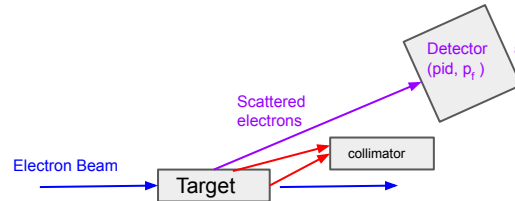
What is our acceptance function $\varepsilon(pid, p_f)$

$\varepsilon(pid, p_0) \equiv$ number of "hits" in a **sensitive detector volume** (an active area in the simulation to record), A "hits" could be:

1. Any charged particle in a specific quartz tile,
2. Electrons going through a virtual detector plane,
3. Photo-electrons generated a PMT photocathode,
4. Electromagnetic interactions in a magnet coil

Acceptance function depends on the what is our end goal of the simulation, for example:

1. Estimating the moller electrons reaching MOLLER detector area
2. Energy deposit in MOLLER primary collimator

"hits" contains particle identification (pid), kinematics, energy deposits, and etc.

What is our acceptance function $\varepsilon(pid, p_f)$

$\varepsilon(pid, p_0) \equiv$ number of "hits" in a **sensitive detector volume** (an active area in the simulation to record), A "hits" could be:

1. Any charged particle in a specific quartz tile,
2. Electrons going through a virtual detector plane,
3. Photo-electrons generated a PMT photocathode,
4. Electromagnetic interactions in a magnet coil

Two approaches are available:

1. Store all "hits" (this uses the `hit` branch) and do the Monte Carlo sum with weighted by `rate` parameter in offline analysis
2. Integrate "hits" for each event during simulation, e.g. $\langle E \, dep \rangle = \sum E \, dep \cdot \varepsilon(pid, p_0)$, and follow up with sum of $\langle E \, dep \rangle$ in offline analysis (this uses the `sum` branch)

What is our acceptance function $\varepsilon(pid, p_f)$

Some practical reason why we need biased events:

- By design, $\varepsilon(pid, p_0)$ for MOLLER detectors is large for Møller scattering but Møller scattering cross section is very low compared to total cross section of beam electrons interactions with Hydrogen atoms (in the target)
- To study and optimize detectors we need lots of Møller events, How do we do that?
  - We could run lots of (I mean we need billions of events) "Beam generator" (rastered) beam electrons on target. Time consuming, inefficient
  - More efficient way is to use a biased "Physics generator" that only do Møller scattering and now each event must be weighted
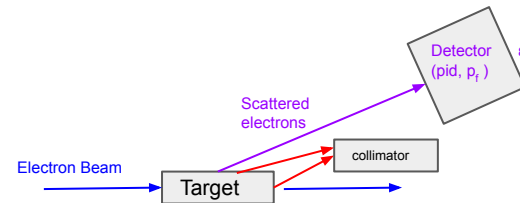
Geant4 Determines what happens with the initial state

One initial state (biased or unbiased) can result in multiple outcomes; which one is randomly decided?

- Physics Lists and Limits determines particle interactions and decays as particles propagates from the initial state
- Several predetermined options are available in geant4. Recommendations:●
  - **QGSP_BERT** (default)
  - **QGSP_BERT_HP** (for better shielding simulations)
- Production cuts: minimum required interaction length for new particles
- User Limits: minimum energy, maximum length, maximum time
  - We may set maximum propagation length to zero for some volumes, and call it "kryptonite". All tracks will stop immediately in these volumes.

Geant4: Definition of Geometry

How we define our simulation world within Geant4? There are three core geometry classes in geant4:

1. **Solids** = a primitive geometric shape (CSG = constructive solid geometry), such as a box, tube,..., polycone,..., up to boolean combinations of other solids with relative position and rotation
2. **Logical volumes** = a combination of a solid and a material (e.g. quartz box), local B-fields, sensitive detector, visibility attributes, GDML auxiliary attributes
3. **Physical volumes** = placement of a 'daughter' logical volume (with relative position and rotation) inside a 'mother' logical volume; single or replication
4. We use a markup language: **GDML** to define simulation world

We have a dedicated talk and hands-on session for GDML geometry

Geant4: Particle Propagation in Geometry

Particles are propagated as tiny **steps** in Geant4. When inside a **physics volume**, at every **step**, geant4 evaluates the following:

- Check if still inside the volume itself (i.e. distance to outside)
- Check if entering any new daughters (i.e. distance to inside)
- When exiting the volume, enter mother volume and any of its daughters

Geometry do's and don'ts:

- Reuse solids and logical volumes if possible
- Encapsulate complicated volumes in simple primitives
- Avoid large numbers of daughters

Geant4: Definition of Geometry

- Overlapping volumes are bad, avoid at any cost
- Overlap checks are done automatically (read warnings in simulation output)
- Shared faces are allowed
  - There is no need to add arbitrary small distances to avoid shared faces

Geant4: Randomness

- Monte Carlo method only work when events are statistically independent
- In computers we only have access to Pseudo-Random number generators
- Geant4 can sets a pseudo-random seed from the computer system, unless you specify the seed explicitly or load a full state explicitly
- Every event's state is saved in output ROOT tree so you can reproduce that exact event given exact same compiled software version

Geant4: Definition of a Hit

Endpoint of a step in a logical volume that is a sensitive detector is a hit within Geant4

It is important to note that:

- One track can leave multiple "geant4 hits" in a sensitive detector. E.g. using photo-electrons in a PMT to determine rate is going to overestimate actual rate by the average number of photoelectrons per event
- Multiple tracks can each leave "geant4 hits" in a sensitive detector. E.g. a showermax calorimeter would likely have many hits from many tracks

These issues must be accounted when estimating rates, energy deposits using "geant4 hit"

Geant4 (Remoll) Output Data Structure

The output ROOT file from remoll includes the geant4 hits in one list per event

- `units`: branch with common units
- `seed`: branch with random state information
- `rate`: variable for rate-weighted histograms
- `ev`: event-level quantities: asymmetry, beam energy, cross section
- `bm`: beam-level quantities: raster position, angles
- `part`: generated particles in initial state (two per event for Møller generator)
- `hit`: hit-level quantities: time, position, energy, pid, etc
- `sum`: summed hit-level quantities: E dep , $\langle x \rangle, \langle y \rangle, \langle z \rangle$

Geant4 System of Units

Geant4 uses a self-consistent system of units with a corresponding set of physical constants. Physical quantities in geant4 (remoll) are implicitly in these units. A subset of common units is stored in the output ROOT file

- Output quantities by dividing units: hit.x/mm, hit.p/GeV
- **Input** quantities with their units: 0.5*cm, 11*GeV

```
T->Draw("hit.p/MeV","")
```
to plot (output) histogram in units of MeV
```
T->Draw("hit.p/GeV","hit.p>1000*MeV")
```
to plot (output) histogram in units of GeV for momentum above 1000 MeV as a cut (input)

We have a dedicated talk and hands-on session for CERN ROOT

Optical Photons

- Geant4 can simulate scintillation and Cherenkov light but one charged particle can generate many 100s or 1000s of optical photons, so this is not enabled by default
- Standalone simulations of individual detectors are possible inside remoll framework to conduct optical photon simulations
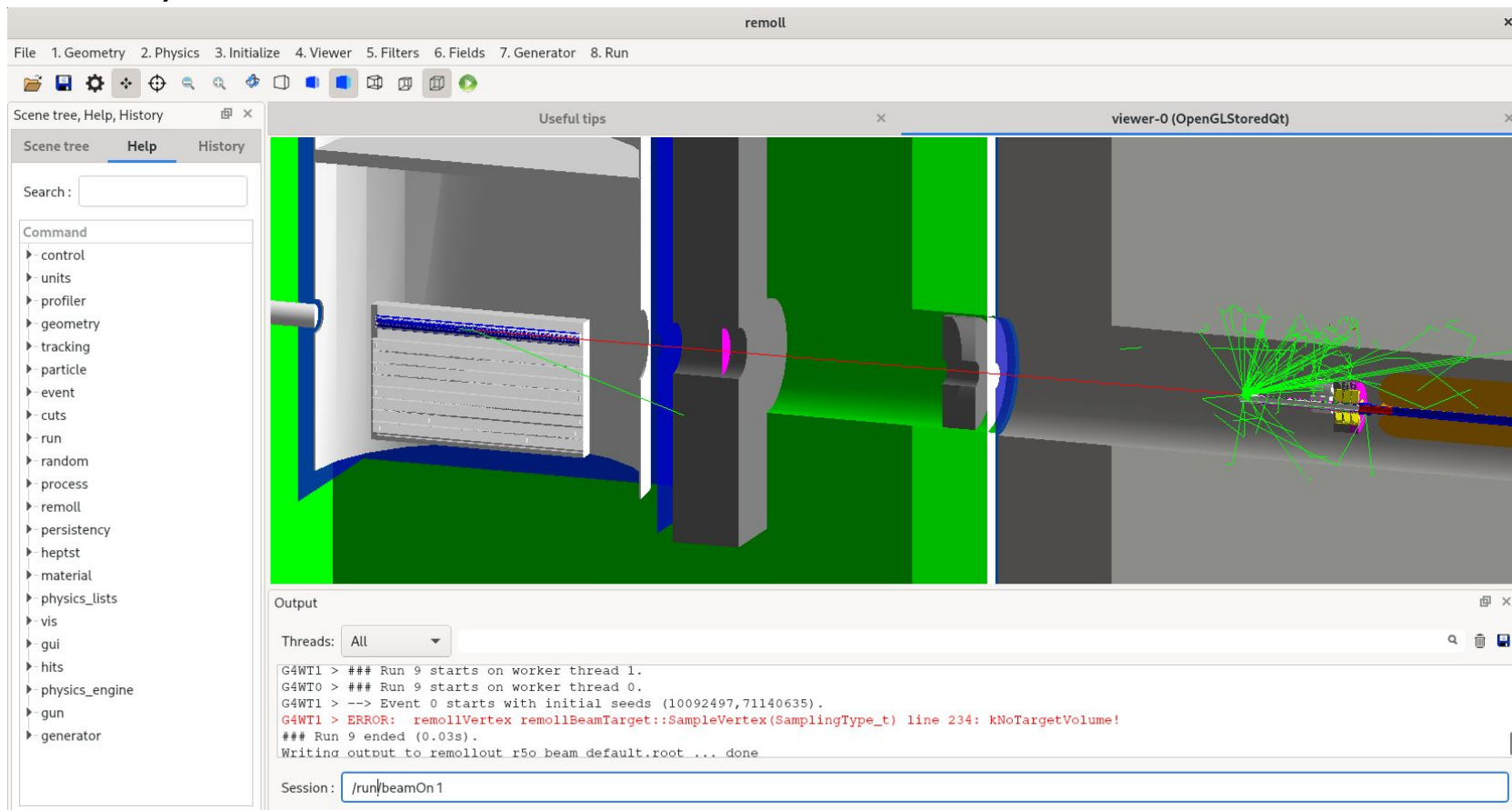
Remoll: MOLLER Geant4 Simulation framework

- Geant4 is a framework and set of libraries
- To utilize it, we need to build our own software using libraries provided by Geant4 for simulation and CERN root for output
- Remoll is the software we have built over years for our many simulation purposes
- The code is kept in a version controlled repository
- We will do hands-on session to setup remoll and do a simple simulation project

Remoll: Visualization of Simulation Model (Hands-on session)

- The simulation geometries are implemented using GDML (see GDML overview talk)
- remoll has an interactive GUI that allows users to view 3D geometry
- We mainly use visualization to verify simulation geometry

Remoll: Visualization of Simulation Model (Hands-on session)

Remoll: How to run simulation (Hands-on session)

- Configurations for simulation is set using a text file called macro file
- This file set
  - Geometry we want to simulation
  - Set various setting for sensitive detectors that record "hit" information
  - Event generator: type, energy, initial kinematics, etc.
  - Physics list
  - How many events to simulate
  - Output ROOT file name
- Then we run (execute) the remoll code

Remoll: Output Data Structure

The output ROOT file from remoll includes the geant4 hits in one list per event

- `units`: branch with common units
- `seed`: branch with random state information
- `rate`: variable for rate-weighted histograms
- `ev`: event-level quantities: asymmetry, beam energy, cross section
- `bm`: beam-level quantities: raster position, angles
- `part`: generated particles in initial state (two per event for Møller generator)
- `hit`: hit-level quantities: time, position, energy, pid, etc
- `sum`: summed hit-level quantities: E dep , $\langle x\rangle, \langle y\rangle, \langle z\rangle$

Remoll: How to access the output data (Hands-on Session)

We can use ROOT data analysis framework to analyze simulation output

- Simplest way is to use the data structure browser provided by ROOT
- Command line tools such as "Draw" command
- Write your own scripts

Overview of types of studies we do using remoll

- Radiation shielding optimization and dose estimates
- Collimator design (energy deposit estimates)
- Background estimation (one and two bounce photon background)
- Study effects of ferrous materials in the experiment apparatus on main detectors
- Acceptance studies (envelope and etc.)
- Detector development (cerenkov and scintillation light simulations)
- Above list is a limited set of examples for studies we have been doing
- These studies can be categorized into primary and secondary simulations
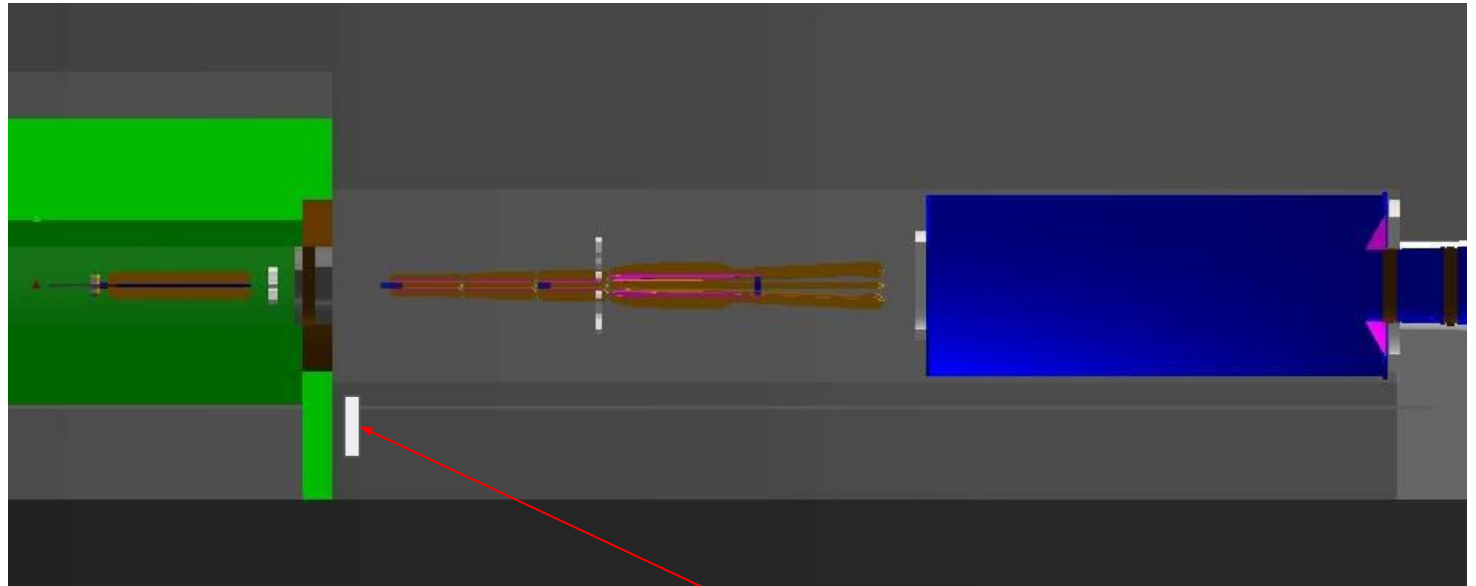
Primary Simulations

Examples: I need to know particle flux along the spectrometer, I place sensitive detectors along the spectrometer
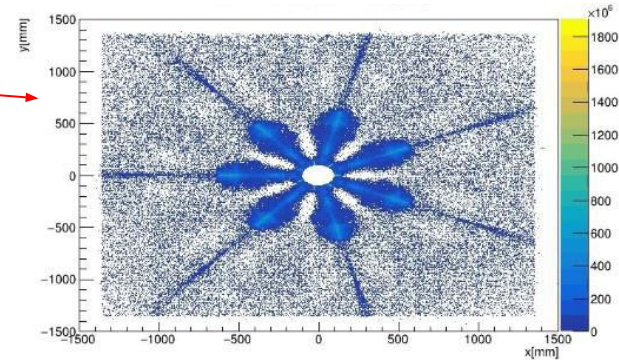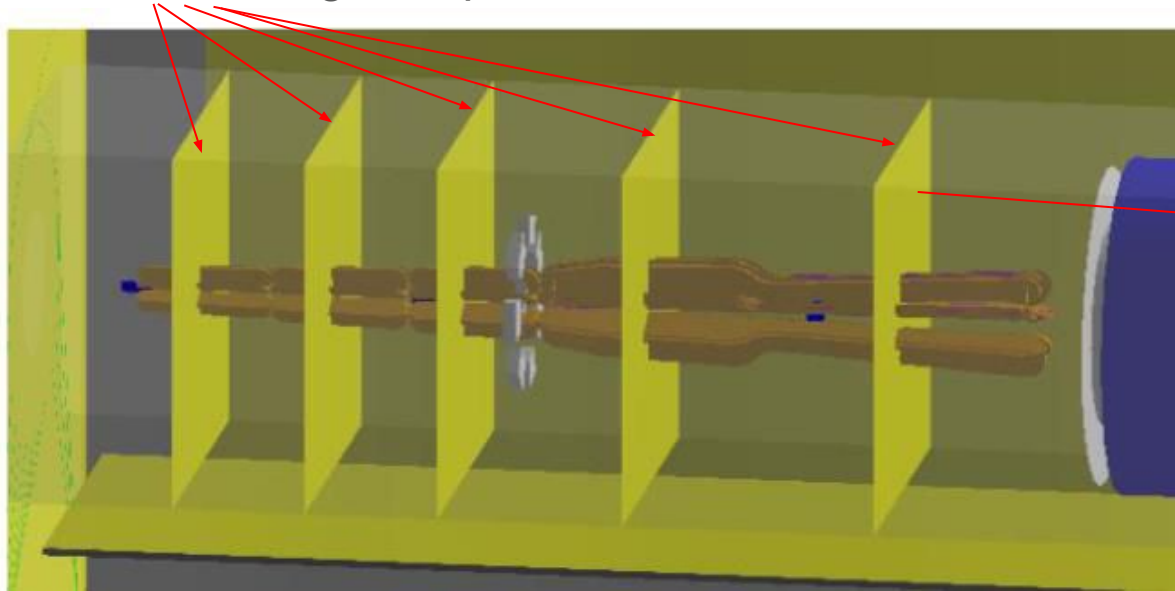
Root Plot of hit distribution recorded in the sensitive detector. The color is the weight factor `rate`

24

Primary Simulations

Example: I need to know what is the total flux of particles reaching this location when 11 GeV beam is incident on target: Place a sensitive detector to record hits and simulate beam on target using complete geometry of the experiment

Place a sensitive detector to record hits.

What is a Secondary Simulation

- An example: What if I want to know the total ionizing dose if I place an electronic module?
  - Then we first do primary simulation to get particle flux at a sensitive detector placed at the location where we want to keep the electronic module
  - Create a standalone geometry with a silicon volume to mimic electronic module
  - We do secondary simulation using the hits at the sensitive detector as initial state and incident them on silicon volume
- Another example: Optical photons on scinitillation detectors placed in the acceptance of MOLLER apparatus
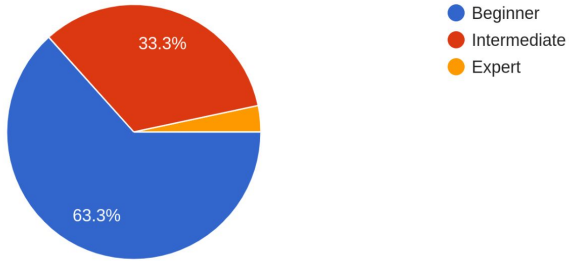
Summary

Following topics will be included in hands-on session

1. How to setup Geant4 simulation (remoll) for MOLLER
2. How to run Geant4 simulation (remoll) for MOLLER
3. GDML introduction
4. Visualize geometry
5. Simple simulation project to run a simulation
6. Access output using ROOT commands
7. How to utilize computing clusters available at JLab and compute canada

Skill Levels of Participants

Simulation Skill Level

30 responses

- ● Beginner
- ● Intermediate
- ● Expert

33.3%

63.3%

Occupation

31 responses

- ● Undergraduate Student
- ● Graduate Student
- ● Post-Doctaral Fellow
- ● Faculty

12.9%

61.3%

22.6%