Introduction to Geant4 with remoll

Prakash Gautam
photon@virginia.edu

Simulation Workshop
2023-05-26

Geant4

Toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science.

- Geant4 is a C++ Library to write application for particle physics simulation.
- The simulation is based on Monte Carlo(MC) method.
- Geant4 can simulate:
  - the passage of particle through matter.
  - interactions of particles with matter. (EM, weak and strong interactions etc.)
  - the production of secondary particles.
  - detector response
- Has visulaization features. Detector geometry, tracks, etc.

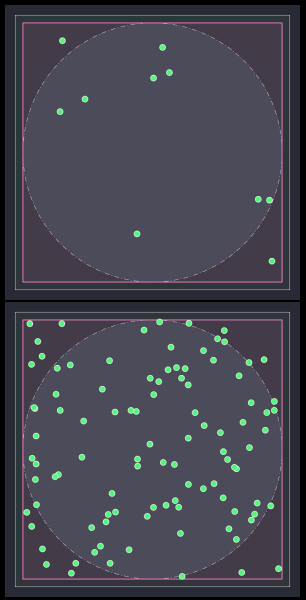
What is Monte Carlo Simulation?

Monte Carlo method is a statistical process.

$$P(C) = \frac{\#C}{\#S} = \frac{\text{Aref of Circle}}{\text{Area of Squre}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4} \implies \pi = 4 \times \frac{\#C}{\#A}$$

- Top plot: N $= 10$, $\pi \approx 4 \times \frac{8}{10} = 3.2$.
- Bottom plot: N $= 100$, $\pi \approx 4 \times \frac{79}{100} = 3.16$.
- More N: 1k: 3.108, 10k: 3.1485, 100k: 3.13624, 1M: 3.14353, 100M: 3.14151724

Higher statictics produce more precise result.

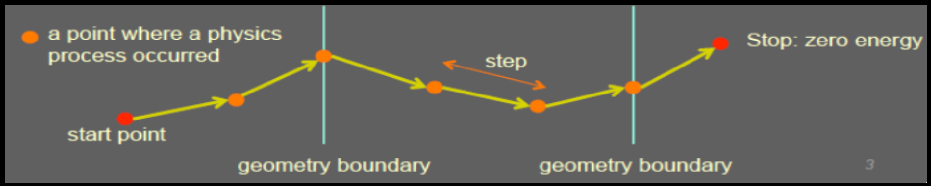


A typical Experiment

- **Setup experimental hardware**
  - Construct geometry.
  - Place detectors
- **Initialize run**
  - Particle type
  - Position
  - Momentum etc.
- **Setup Physics Environment**
  - Magnetic field
  - Electric field etc.
- **Measure/Record**
  - Record data.
  - Read the recorded data.
  - Process and analyze.

A typical Simulation

- Construct a geometry approximating the experiment hardware.
- Initialize particle(s). (beam, elastic etc.) with some initial parameter (position, momentum etc.)
- For each generated particle.
  - Propagate by a step-size according to the physics processes in the current geometry.
  - Generate secondary particles according to interactions.
- If the particle reaches a sensitive detector save the information.



DocDB: 732

- An **event** is the entire process from generation point to its stop point (either by limit or by end of geometry) including all secondary particles generated along the way.
- A **hit** is an interaction at the sensitive volume. Thus an event might have multiple hits.

Setting Up Simulation

- As any of c++ program a `main()` function marks beginning of simulation.

```
int main(int argc, char** argv)
{
    G4RunManager* run_manager = new G4RunManager();
    run_manager->SetUserInitialization(new MyDetectorConstruction()); // constructs detector
    run_manager->SetUserInitialization(new MyActionInitialization()); // sets up generator
    run_manager->SetUserInitialization(new MyPhysicsList()); // creates physics environment
    run_manager->Initialize();

    G4UIExecutive* ui = new G4UIExecutive(argc,argv);
    G4VisManager* vis_manager = new G4VisExecutive();
    vis_manager->Initialize();

    G4UImanager* uip = G4UImanager::GetUIpointer();
    uip->ApplyCommand("/vis/open OGL");
    uip->ApplyCommand("/vis/drawVolume");
    uip->ApplyCommand("/vis/viewer/sets autorefresh");
    uip->ApplyCommand("/vis/scene/add/trajectories smooth");
    ui->SessionStart();
    return 0;
}
```

- For `remoll` the main file is `remoll.cc`. That file closely follows this structure.

Constructing Geometry

- Using C++ API: Requires recompilation on code change.

```
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
auto solid_world = std::make_shared<G4Box>("solidworld",0.5*m,0.5*m,0.5*m);
std::shared_ptr<G4LogicalVolume> logic_world = std::make_shared<G4LogicalVolume>(solid_world.get(),world_mat,"logicworld");
auto phy_world = std::make_shared<G4PVPlacement>(0,G4ThreeVector(0,0,0),logic_world.get(),"physworld",0,false,0,true);
return phy_world;
```

- Using GDML: No need to recompile on geometry change.

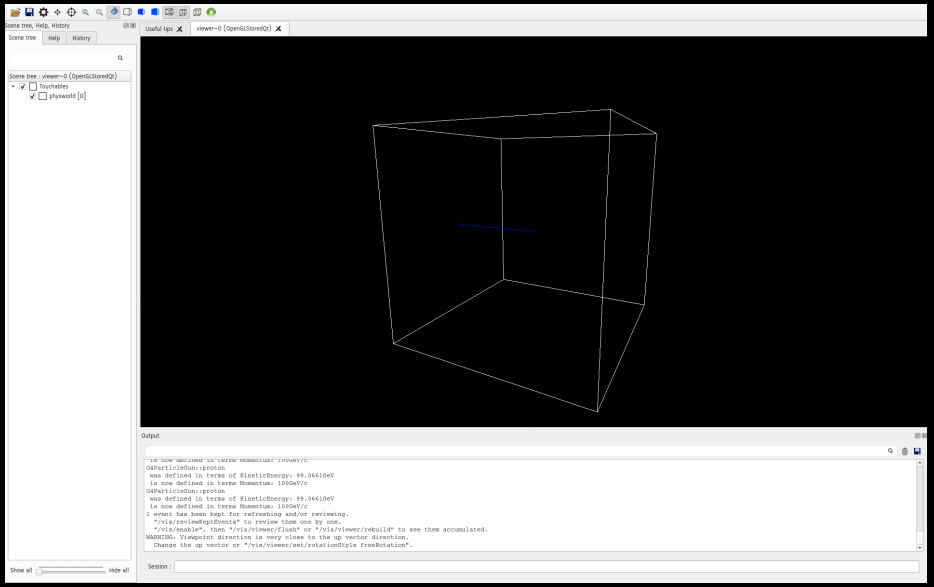
```
<box lunit="m" name="solidworld" x="0.5" y="0.5" z="0.5"/>

<volume name="logicworld">
    <materialref ref="G4_AIR"/>
    <solidref ref="solidworld"/>
    <auxiliary auxtype="Color" auxvalue="blue"/>
</volume>

<physvol name="physworld">
    <volumeref ref="logicworld"/>
    <position name="origin" x="0" y="0" z="0"/>
</physvol>
```

- Solid → Logical Volume → Physical Volume.

Visualizer



A typical Geant4 visualization GUI with a simple geometry.

Making Sensitive Detector

Once the physics processes are simulated, we need to record the information of that particle.

- We can make detector sensitive by setting parameter.

```
<volume name="logicworld">
    <materialref ref="G4_AIR"/>
    <solidref ref="solidworld"/>
    <auxiliary auxtype="Color" auxvalue="blue"/>
    <auxiliary auxtype="SensDet" auxvalue="collDet"/>
    <auxiliary auxtype="DetNo" auxvalue="170"/>
</volume>
```

- Each sensitive detector is identified by an unique id. `DetNo`.
- The recorded event information can be output in various ways.
- Usually `ROOT` is used save the information in a format called `.root` file.

Generators

- **Particle Generators**
- This provides the initial particle generation.

```
G4ParticleTable* particle_table = G4ParticleTable::GetParticleTable();
G4ParticleDefinition* proton = particle_table->FindParticle("proton");
G4ThreeVector position(0,0,0);
G4ThreeVector momentum(0.0,0.0,1.0);

auto proton_gun = new G4ParticleGun();
proton_gun->SetParticlePosition(position);
proton_gun->SetParticleMomentumDirection(momentum);
proton_gun->SetParticleMomentum(100.0*GeV);
proton_gun->SetParticleDefinition(proton);
proton_gun->GeneratePrimaryVertex(event);
```

- We have control over various parameters of generated particle, position, momentum, spin, charge etc.

Physics Processes aka PhysicsLists

- **Physics Processes**
- What Physics processes do we want to simulate, EM, optical photon, custom interactions.

```
RegisterReferencePhysList("QGSP_BERT");
```

QGSP_BERT

QGSP_BERT is a physics process used in the Geant4 toolkit for simulating the passage of particles through matter. It is a hybrid model that combines the Quark Gluon String (QGS) model for high-energy interactions with the Bertini cascade model for low-energy interactions. The QGSP model is based on the theory of quantum chromodynamics (QCD), while the Bertini cascade model is based on a semi-empirical approach.

- The physics processes can be altered.

```
auto hadronPhysics = new G4HadronPhysicsQGSP_BERT();
hadronPhysics->SetStepLimit(100, G4Proton::Proton());
```

Interaction with Messenger

- Messenger class allows us to interact with simulation through a configuration file.

```
//...
double somevariable = 0.0;
//...
G4GenericMessenger fMessenger{ this, "/category", "some category"};
//...
fMessenger.DeclareMethod("key", &set_value_of_key, "set value of key");
//...
void set_value_of_key(double somevalue){
   somevariable = some_other_function(somevalue);
};
```

- In a configuration file we can then set:

```
/category/key 10.0 # content of mac file
```

- They usually have extension `.mac`.

Macros and setting parameters

- By default the Geant4 library comes with lots of macros commands.
- They can be used to set the parameters of simulation.

```
/run/initialize

/run/verbose 1

/run/beamOn 1000
```

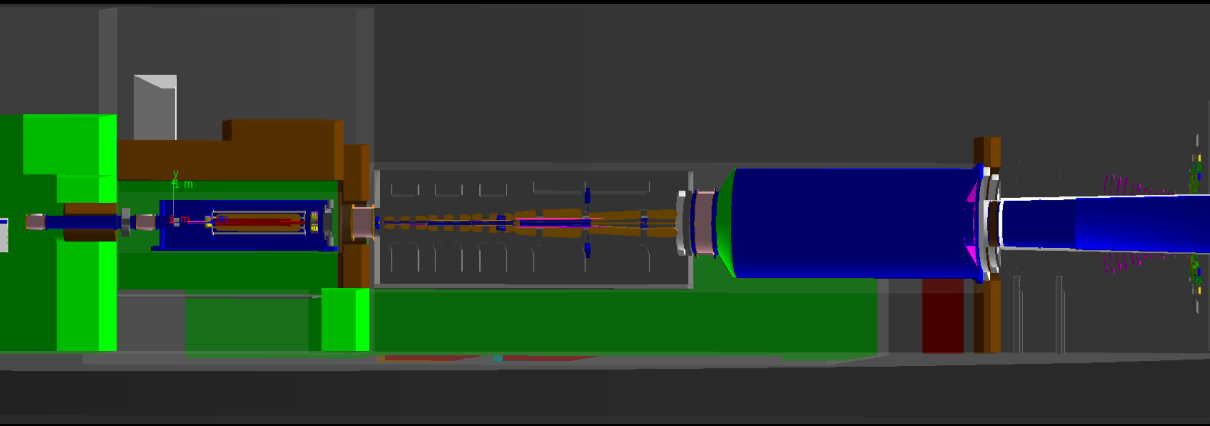
- https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Control/AllResources/Control/UIcommands/_.html has the list of Geant4 default macro commands.

remoll

- `remoll` is an simulation tool used to simulate Moller experiment.
- It builds on top of Geant4 library.
- Has various physics generators specific to Moller experiment.
  - `beam` : `remollgenBeam.cc`
  - `moller` : `remollGenMoller.cc`
  - `elastic` : `remollGenpElastic.hh`
- Compiles to an executable called `remoll`.
- The code is publicly available at `https://github.com/JeffersonLab/remoll.git`

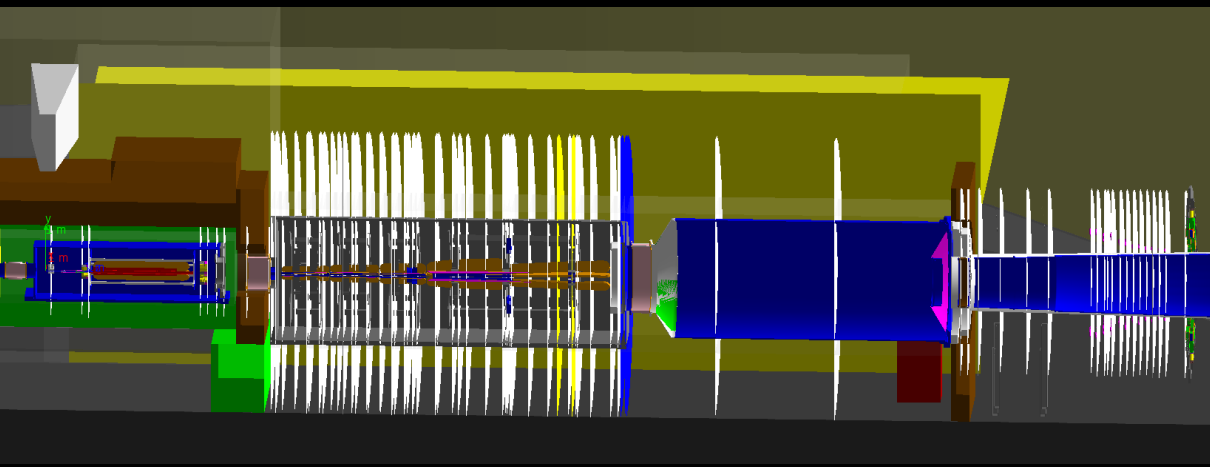
It is a result of combined effort of many people over several years.

remoll geometry



- The geometry is defined using GDML files.

Parallel geometry



- There are virtual planes along and perpendicular to the beam direction.
- These do not participate in physics processes.
- They record the hits as they reach plane and act as if it was not there physicswise.
- Overcoming the "measurement problem".

Running Simulation

- For some generator, start some number of events.
- Propagate the particle through the geometry.
- Record hits in sensitive detector.
- Write out the hit information at the sensitive detector to an outputfile.

To run a simulation

```
~/sim/dir$ remoll primary_sim.mac
```

```
/remoll/setgeofile geometry/mollerMother.gdml
/remoll/parallel/setfile geometry/mollerParallel.gdml
/remoll/physlist/parallel/enable
/run/initialize

/remoll/printgeometry true
/remoll/addfield V2DSg.9.75cm.parallel.txt
/remoll/addfield V2U.1a.50cm.parallel.txt

/remoll/evgen/set beam

/remoll/SD/disable_all
/remoll/SD/enable 911
/remoll/SD/detect lowenergyneutral 911
/remoll/SD/detect secondaries 911
/remoll/SD/detect boundaryhits 911

/remoll/filename output-filename.root

/run/beamOn 100000
```

Secondary Simulation

If we want to do a more detailed simulation of events at particular component, then we run so called "secondary simulation"

- First run a simulation ("primary").
- Select only those hits that would hit the "component" in question and write to so called "skim" files.
  The event statistics might be limited.
- Set those event information as new generator of particles, and run a new simulation.

```
/remoll/setgeofile geometry/mollerMother.gdml
/remoll/parallel/setfile geometry/mollerParallel.gdml
/remoll/physlist/parallel/enable
/run/initialize

/remoll/printgeometry true
/remoll/addfield V2DSg.9.75cm.parallel.txt
/remoll/addfield V2U.1a.50cm.parallel.txt

/remoll/evgen/set external
/remoll/evgen/external/file primary-skim-file.root
/remoll/evgen/external/detid 911
/remoll/evgen/external/zOffset 0.001

/remoll/SD/disable_all
/remoll/SD/enable 911
/remoll/SD/detect lowenergyneutral 911
/remoll/SD/detect secondaries 911
/remoll/SD/detect boundaryhits 911

/remoll/filename secondary-output-filename.root

/run/beamOn 100000
```

Remoll data structure

The events are saved in a root file with tree named "T". For each **event**, (we can think of this as a row)

- Any number particles generated by the generator is saved in the **part** branch. So part branch is `std::vector<remollEventParticle_t>;`
  - Each generated particle get unique track id, starting at 1 for first primary track.
- Depending on generator, each generated event might not be equally likely. The **rate** branch holds the rate corresponding to the likelihood of such event in that generator process.
- If the propagated particles reach a sensitive detector, it is stored in **hit** branch.
  - There are (most likely) multiple hit per event. So hit branch is `std::vector<remollGenericDetectorHit_t>;`

rate	part	hit
2.03	[p1, p2, ...]	[h1,h2,h3, ...]
4.03	[p1, p2, ...]	[h1,h2,h3, ...]
1.03	[p1, p2, ...]	[h1,h2,]

Remoll data structure(contd)

Besides these, there are other branches in the tree.

- **units**: branch with common units
- **seed**: branch with random state information
- **ev**: event-level quantities: asymmetry, beam energy, cross section
- **bm**: beam-level quantities: raster position, angles
- **sum**: summed hit-level quantities: $E_{dep}, \langle x \rangle, \langle y \rangle, \langle z \rangle$

Data Analysis

- Once we have the root file with the output we can do various analysis

```
~/output/dir$ reroot output-filename.root
```

- Draw a histogram of photon hit distribution at detector 28.

```
T->Draw("hit.y:hit.x","hit.det == 28 && hit.pid == 22","colz");
```

More on this in the Hands-On session.

Summary

- Geant4 is a simulation toolkit used to do Monte Carlo (MC) simulation.
- It provides tools to not only simulate physics processes but also visualize them.
- `remoll` is a simulation framework written in C++ with Geant4.
- we use `remoll` to simulate Moller experiment.

Backup