# Geant4 — an Introduction
# Lecture — 3

Experimental High Energy Particle and Astroparticle Physics
February 19, 2026

Sunanda Banerjee

# What we Learnt so far

- Workflow of a Geant4 Application
- Terminologies used within Geant4
- Modelling of a detector in an application
  - Materials
  - Logical and Physical Volumes
- Extraction of Useful information
  - Sensitive detectors and Hits
  - Step points and Touchables
- Physics Models Geant4 provides
  - Electromangentic
  - Hadronic
- Physics List
- Some example codes to be used for making an application

# Today's Menu

- Some examples
  - Addition of new particles
  - Addition of physics process to make one's own physics list

# Particles

- Geant4 provides a list of well established particles. One can access the minimal set of particles by using

  #include “G4EmBuilder.hh”

  G4EmBuilder::ConstructMinimalEmSet();

- To simulate new particles which are not yet there in standard list, there is a provision to add new particles:

```
G4ParticleDefinition(aName,          // G4String
                     mass,           // G4double
                     width,          // G4double
                     charge,         // G4double
                     iSpin,          // G4int
                     iParity,        // G4int
                     iConjugation,   // G4int
                     iIsospin,       // G4int
                     iIsospin3,      // G4int
                     gParity,        // G4int
                     pType,          // G4String
                     lepton,         // G4int
                     baryon,         // G4int
                     encoding,       // G4int
                     stable,         // G4bool
                     lifetime,       // G4double
                     decaytable);    // G4DecayTable*
```

# Process

- Processesare to be added for all the particles separately. Use ConstructProcess()  to add the processes. Some hint is here:

```
void MyPhysicsList::ConstructProcess() {

  G4PhysicsListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
  G4LossTableManager* man = G4LossTableManager::Instance();

  // muon & hadron bremsstrahlung and pair production
  G4MuBremsstrahlung* mub = nullptr;
  G4MuPairProduction* mup = nullptr;
  G4hBremsstrahlung* pib = nullptr;
  G4hPairProduction* pip = nullptr;
  G4hBremsstrahlung* kb = nullptr;
  G4hPairProduction* kp = nullptr;
  G4hBremsstrahlung* pb = nullptr;
  G4hPairProduction* pp = nullptr;

  // muon & hadron multiple scattering
  G4MuMultipleScattering* mumsc = nullptr;
  G4hMultipleScattering* pimsc = nullptr;
  G4hMultipleScattering* kmsc = nullptr;
  G4hMultipleScattering* pmsc = nullptr;
  G4hMultipleScattering* hmsc = nullptr;

  // muon and hadron single scattering
  G4CoulombScattering* muss = nullptr;
  G4CoulombScattering* piss = nullptr;
  G4CoulombScattering* kss = nullptr;

  // high energy limit for e+- scattering models and bremsstrahlung
  G4double highEnergyLimit = 100 * CLHEP::MeV;
  G4ParticleTable* table = G4ParticleTable::GetParticleTable();
  EmParticleList emList;
```

# Types of Processes

- Loop over the list of particles and add suitable processes for each type

```
for (const auto& particleName : emList.PartNames()) {
  G4ParticleDefinition* particle = table->FindParticle(particleName);

  if (particleName == "gamma") {
    G4PhotoElectricEffect* pee = new G4PhotoElectricEffect();

    if (G4EmParameters::Instance()->GeneralProcessActive()) {
      G4GammaGeneralProcess* sp = new G4GammaGeneralProcess();
      sp->AddEmProcess(pee);
      sp->AddEmProcess(new G4ComptonScattering());
      sp->AddEmProcess(new G4GammaConversion());
      man->SetGammaGeneralProcess(sp);
      ph->RegisterProcess(sp, particle);
    } else {
      ph->RegisterProcess(pee, particle);
      ph->RegisterProcess(new G4ComptonScattering(), particle);
      ph->RegisterProcess(new G4GammaConversion(), particle);
    }
```

For Photons

# Types of Processes

```
} else if (particleName == "e-") {
   G4eIonisation* eioni = new G4eIonisation();

   G4eMultipleScattering* msc = new G4eMultipleScattering;
   G4UrbanMscModel* msc1 = new G4UrbanMscModel();
   G4WentzelVIModel* msc2 = new G4WentzelVIModel();
   msc1->SetHighEnergyLimit(highEnergyLimit);
   msc2->SetLowEnergyLimit(highEnergyLimit);
   msc->SetEmModel(msc1);
   msc->SetEmModel(msc2);
```

```
G4eCoulombScatteringModel* ssm = new G4eCoulombScatteringModel();
G4CoulombScattering* ss = new G4CoulombScattering();
ss->SetEmModel(ssm);
ss->SetMinKinEnergy(highEnergyLimit);
ssm->SetLowEnergyLimit(highEnergyLimit);
ssm->SetActivationLowEnergyLimit(highEnergyLimit);

ph->RegisterProcess(msc, particle);
ph->RegisterProcess(eioni, particle);
ph->RegisterProcess(new G4eBremsstrahlung(), particle);
ph->RegisterProcess(ss, particle);
```

For Electrons

# What about Ions

- Get a generic ion and then add specific processes

```
particle = G4GenericIon::GenericIon();
G4ionIonisation* ionIoni = new G4ionIonisation();
ionIoni->SetEmModel(new
G4IonParametrisedLossModel());
ph->RegisterProcess(hmsc, particle);
ph->RegisterProcess(ionIoni, particle);
```

# Register all such Lists

```
// EM Physics
RegisterPhysics(new G4EmStandardPhysics(verbosity));

// Synchroton Radiation & GN Physics
G4EmExtraPhysics* gn = new G4EmExtraPhysics(verbosity);
RegisterPhysics(gn);

// Decays
this->RegisterPhysics(new G4DecayPhysics(verbosity));

// Hadron Elastic scattering
RegisterPhysics(new G4HadronElasticPhysics(verbosity));

// Hadron Physics
RegisterPhysics(new G4HadronPhysicsFTF_BIC(verbosity));

// Stopping Physics
RegisterPhysics(new G4StoppingPhysics(verbosity));

// Ion Physics
RegisterPhysics(new G4IonPhysics(verbosity));

// Neutron tracking cut
RegisterPhysics(new G4NeutronTrackingCut(verbosity));
```

# A Second Tutorial

# Second Tutorial

Goal: Study a detector of a combined calorimeter: ECAL + HCAL

- Detector Summary:
  - Electromagnetic Calorimeter (ECAL)
    - 7x7 Matrix of CeF3 crystals
      - 2.6cm x 2.6 cm x 43 cm
  - Hadron Calorimeter (HCAL)
    - Alternate layers (20) of brass absorbers and plastic scintillators
      - Brass absorbers of lateral dimension 80 cm × 80 cm
        - Front 10 layers 50 mm thick
        - Back 10 layers 56 mm thick
      - Plastic scintillators of size 80 cm × 80 cm x 10 mm
        - Each scintillator layeris divided into 8 x 8 cells of size 10 cm x 10 cm



S. Banerjee, P. Das, A. Roy

# Material Description

- Elements:

| Element | Z | A (g/mole) | Density |
|---|---|---|---|
| Ar | 18 | 39.948 | 1.662 mg/cm3 |
| C | 6 | 12.011 | 2.265 g/cm3 |
| Ce | 58 | 140.12 | 6.637 g/cm3 |
| Cu | 29 | 63.546 | 8.96 g/cm3 |
| F | 9 | 18.998 | 1.108 g/cm3 |
| H | 1 | 1.00794 | 0.708 mg/cm3 |
| N | 7 | 14.007 | 0.808 mg/cm3 |
| O | 8 | 15.998 | 1.335 mg/cm3 |
| Si | 14 | 28.005 | 2.33 g/cm3 |
| Zn | 30 | 65.39 | 7.112 g/cm3 |

- Materials

| Name | Density | Content | Fraction |
|---|---|---|---|
| Air | 1.214 mg/cm3 | N | 0.75 |
| | | O | 0.24 |
| | | Ar | 0.01 |
| Brass | 8.53 g/cm3 | Cu | 0.70 |
| | | Zn | 0.30 |
| CeF3 | 6.16 g /cm3 | Ce | 0.71085068 |
| | | F | 0.28914232 |
| Plastic | 1.032 g/cm3 | C | 0.91512109 |
| | | H | 0.08487891 |

# Hit and Sensitive Detector

- Hit class will contain:
  - ID of CeF3 crystal [x, y position: (100*cell IDY + cell IDX)] or of the Scintillator Cell [x, y, z positions (layer*10000 + 100*cell IDY + cell IDX).]
  - Energy deposit (all particles)
  - Time of the first hit

- Sensitive Detector
  - Attach to the logical volumes corresponding to the CeF3 crystal or the Scintillator cell, first calculate the cell ID:
    - For CeF3 crystal (combine row & column numbers), and for scintillator cell (combine cell position along x, y axes and layer number)
  - Then search, within the hit collection class of either type, a hit with the same ID
    - If found and the hit-time is within 1 ns of the stored hit time, add energy deposit to the corresponding hit energy
    - Otherwise, store cell ID, time, and energy deposit as a single hit

# Event Action and Primary Generator Action

- Event Action:
  - At the beginning of the event define a ROOT tree where o/p is saved
  - At the end of each event, fill the tree with the information from the hit collection
  - In the destructor, close the ROOT file

- Primary Generator Action:
  - Create a primary vertex at the left most edge of the world volume (must be just inside it)
  - Create a particle of given type going along +z direction with a predefined energy

# Creation of Event samples

- Generate 1000 events with 20 GeV negatively charged muons varying the primary vertex positions along x,y to which matches with the centers of each cell and save the outputs as ROOT tree

- Generate six samples each with 1000 events of negative pions for energies of 5, 10, 20, 50, 100 and 200 GeV with the vertex facing the middle of the detector.

# Final Analysis

- Always select hits which satisfy a timing cut (~50 ns)
- Analyze the muon sample to determine what a minimizing ionizing particle deposits energy in each layer of scintillator or in each crystal
- Use the MPV (most probable value) for each cell to have relative calibration of each cell in ECAL or HCAL
- Now use the 50 GeV pion run and see number of MIPs in each cell (separately for ECAL and HCAL). Scale total MIPs to 10 GeV to get overall energy scale factor for each type of hit energy
- Analyze all pion samples to get # of MIPS and convert to energies using the scale factor obtained from the previous step. From the energy distribution, compute most probable value and its mean dispersion to measure linearity as well as resolution.
- Separate events where shower starts in ECAL or HCAL. Check if one can improve resolution by treating the two samples separately.
- Also separate energy deposits due to electrons/positrons (which can produce Cerenkov as well as scintillation signals) or due to charged hadrons (which can generate signals only due to scintillation) and see if suitably combining the two components one can get back lineaarity in response and improve energy resolution.