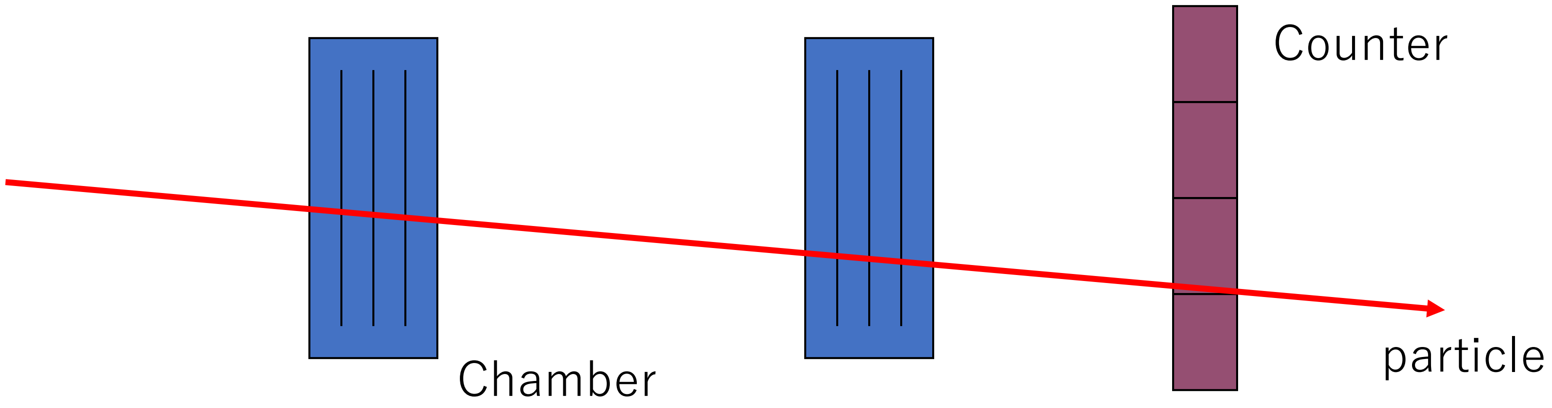
Monte Carlo ゼミ4

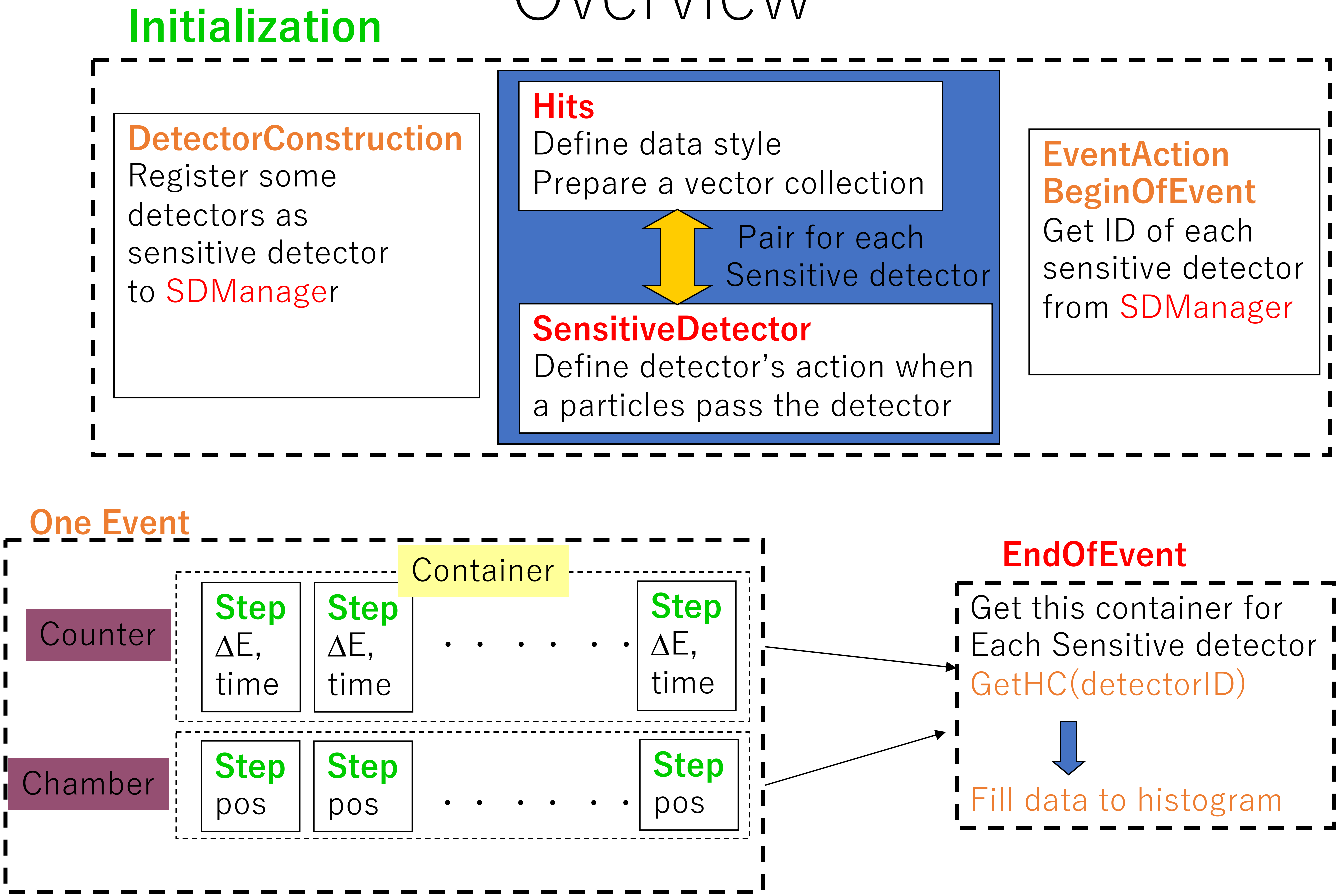
K. Miwa

Sensitive detector



- In a simulation, we want to know
  - Position and time of the particle at the detector
  - Momentum and energy of the particle at the detector
  - Energy deposition of the particle in the detector
- We register the detector as a sensitive detector

Overview


Initialization
DetectorConstruction
Register some detectors as sensitive detector to SDManager
Hits
Define data style
Prepare a vector collection
Pair for each Sensitive detector
SensitiveDetector
Define detector's action when a particles pass the detector
EventAction
BeginOfEvent
Get ID of each sensitive detector from SDManager
One Event
Container
Counter
Step ΔE, time
Step ΔE, time
Step ΔE, time
Chamber
Step pos
Step pos
Step pos
EndOfEvent
Get this container for Each Sensitive detector
GetHC(detectorID)
Fill data to histogram

Registration of Sensitive Detector

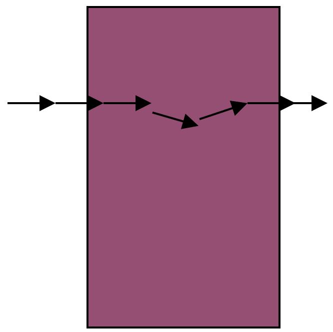
DetectorConstruction.cc

```
solidCalor = new G4Box("Calorimeter",        //its name
                 CalorThickness/2,CalorSizeYZ/2,CalorSizeYZ/2);

logicCalor = new G4LogicalVolume(solidCalor,  //its solid
                         Sci, //its material
                         "Calorimeter");      //its name

physiCalor = new G4PVPlacement(0,                   //no rotation
                       ……省略:)

//-------------Sensitive Detector
G4SDManager* SDman = G4SDManager::GetSDMpointer();
if(!counterSD)
  {
   counterSD = new ExN00CounterSD("counterSD",this);
   SDman->AddNewDetector( counterSD );
  }
logicCalor->SetSensitiveDetector(counterSD);
```

G4SDManager is the singleton manager class for sensitive detector.
We register a sensitive detector to G4SDManager.

Hit

- A hit is a snapshot of the physical interaction of a track in the sensitive region.
- The information of
  - Position and time of the step
  - Momentum and energy of track
  - Energy deposition of the step
  - Geometrical information

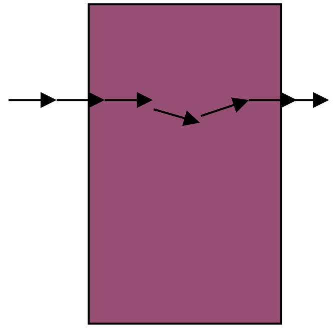


- G4VHit
  - G4VHit is an abstract base class which represent a hit
  - You have to inherit this base class and derive your own class
- G4THitsCollection
  - There are many hits in one event.
  - This is a vector collection which contain all hits in one event

```
class ExN00CounterHit : public G4VHit
{
 public:

   ExN00CounterHit();
  ~ExN00CounterHit();
     …省略…
 public:
   void AddAbs(G4double de, G4double dl) {EdepAbs += de; TrackLengthAbs += dl;};
   void SetTime(G4double time) {Time =time;};
   G4double GetEdepAbs()     { return EdepAbs; };
   G4double GetTrakAbs()     { return TrackLengthAbs; };
   G4double GetTime()     { return Time; };
 private:
   G4double EdepAbs, TrackLengthAbs;
   G4double Time;
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

typedef G4THitsCollection<ExN00CounterHit> ExN00CounterHitsCollection;

extern G4Allocator<ExN00CounterHit> ExN00CounterHitAllocator;
```

Sensitive Detector

- G4VSensitiveDetector is an abstract base class which represents a detector.The principal mandate of a sensitive detector is the construction of hit objects using information from steps along a particle track.
- The ProcessHits() method performs this task using G4Step objects as input



- ProcessHits() method
  - This method Is invoked when there is a step in the sensitive detector.
  - In this method, you can get information of the particle which pass the detector and energy deposit.
  - These values are passed to ***Hit class .
  - Then this Hit information are stored in a vector container.

```
G4bool ExN00CounterSD::ProcessHits(G4Step* aStep,G4TouchableHistory* ROhist)
{
  //G4cout << "####ExN00CalorimeterSD::ProcessHits " << detectorname  << G4endl;
  G4double edep = aStep->GetTotalEnergyDeposit();
  G4double time = aStep->GetTrack()->GetGlobalTime();

  G4double stepl = 0.;
  G4String particleName;
  particleName = aStep->GetTrack()->GetDefinition()->GetParticleName();
  if (aStep->GetTrack()->GetDefinition()->GetPDGCharge() != 0.)
      stepl = aStep->GetStepLength();

  if ((edep==0.)&&(stepl==0.)) return false;

  G4TouchableHistory* theTouchable
    = (G4TouchableHistory*)(aStep->GetPreStepPoint()->GetTouchable());

  if (HitID==-1)
    {
      ExN00CounterHit* calHit = new ExN00CounterHit();
      calHit->AddAbs(edep, stepl);
      calHit->SetTime(time);
      HitID = CalCollection->insert(calHit) - 1;
    }
  else
    {
      (*CalCollection)[HitID]->AddAbs(edep,stepl);
    }

  return true;
}
```

EndOfEvent

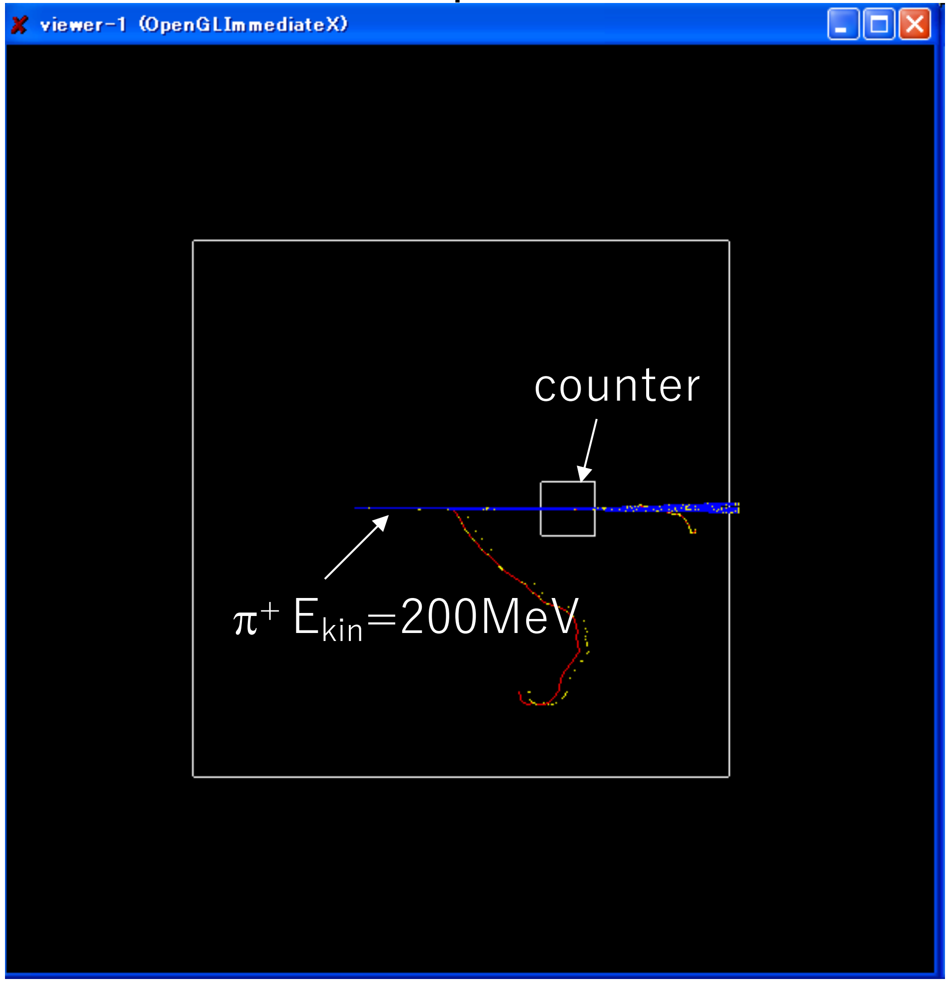
```
void ExN00EventAction::EndOfEventAction(const G4Event*
evt)
{
  G4int evtNb = evt->GetEventID();

  G4HCofThisEvent* HCE = evt->GetHCofThisEvent();
  ExN00CounterHitsCollection* CounterHC = 0;

  if (HCE) CounterHC = (ExN00CounterHitsCollection*)(HCE-
>GetHC(counterCollID));
  if (CounterHC) {
    G4int n_hit = CounterHC->entries();
    for (G4int i=0;i<n_hit;i++){
      double time  = (*CounterHC)[i]->GetTime();
      double de    = (*CounterHC)[i]->GetEdepAbs();
      G4cout << "Time : " << time <<", dE : " << de <<
G4endl;
      AnaRoot->FillEnergyDeposit(de);
      AnaRoot->FillTime(time);
    }
  }
```

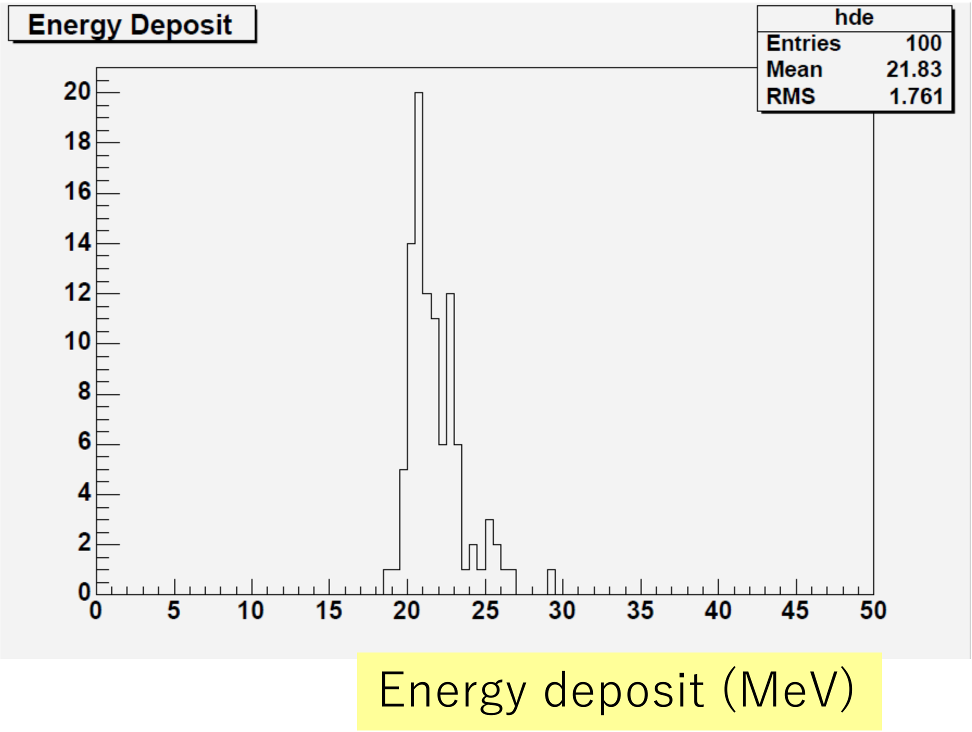
- You can get HitCollectoin of your sensitive detector for 1event at the EndOfEvent

Template example

- dhcp2576:/home/sks/user/miwa/MonteCarlo/geant4/N00
- New Added files
  - ExN00CounterHit.cc, .hh define Hit class and HitCollection
  - ExN00CounterSD.cc, .hh defile SensitiveDetector's behavior
  - ExN00RunAction.cc, .hh open root file at the BeginOfRun and close root file at the EndOfRun
  - ExN00AnaRoot.cc, .hh defile root files and histogram
- Modified files
  - ExN00DetectorConstruction.cc, .hh
  - ExN00EventAction.cc, .hh
  - exampleN00.cc

Energy deposit at Counter (10cm thick plastic scintillator)



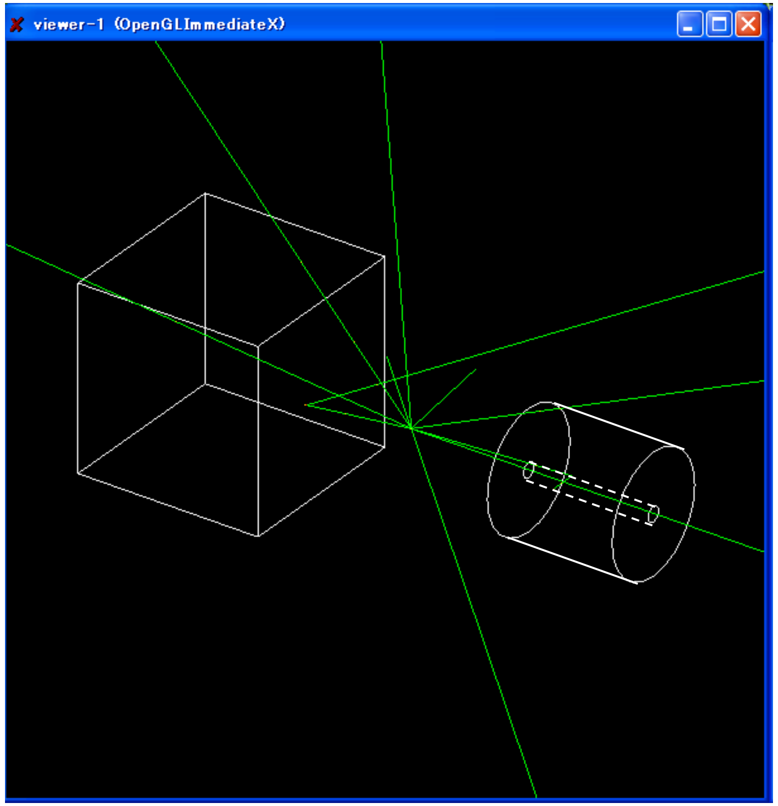


In this program, a root file is created for each run. (run(run#).root)

Practice

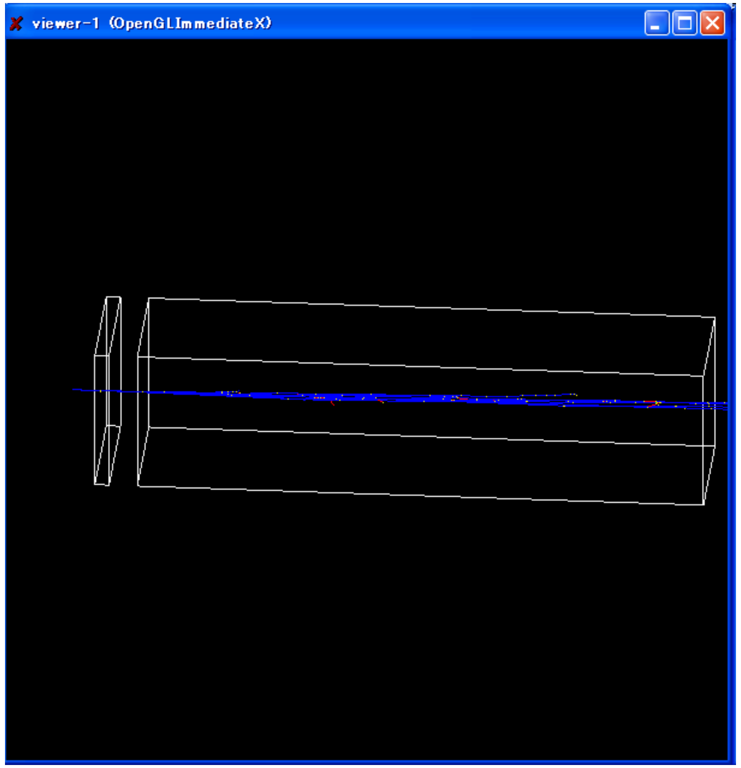
- 今の条件(pi+, E_{kin}=200MeV, 10cm thick Scintillator)
  - Geantでenergy lossのヒストを作る。
  - Bethe-Blochで計算した値と比較する。
- カウンターの材質を鉛にする。このときに上の条件で行うとそれぞれどうなるか？Pi+は鉛の中で止まるかどうか？
- E_{kin}=200MeVのpi+をscintillatorで止めるには厚さはどれくらい必要か？Geantでシンチの厚さを変えながら調べてみる。
- シンチでpi+を止めたとき、止まった場所の分布はどうなっているか？このときは簡単のためdecay processはなしにしましょう。

Practice1:
Interaction of photon with materials



- Detector
  - Plastic scintillator
    - Position (x, y, z)=(-10cm, 0, 0)
    - Volume 10.cm×10.cm ×10.cm
    - Density 1.032 g/cm^3
  - Ge crystal
    - Position (x, y, z)=(10cm, 0, 0)
    - Volume R_{min}=0.4cm, R_{max}=3.25cm, Z/2=3.47cm
- Physics
  - Transportation
  - EMProcess
  - Decay process
- Primary Generation
  - Generate γ ray from (0, 0, 0) isotropically in 3D space
  - $E\gamma$=1MeV
- Sensitive Detector
  - Plastic scintillator, Ge crystal
  - Measure energy deposit.
- Show energy deposit spectra of each detector.
- Compare the ratio of Compton scattering and photo electric effect ($\sigma_{Compton} \propto Z$, $\sigma_{photo} \propto Z^5$)

Practice2:
ΔE,E counter (particle identification)



- When you measure ΔE and total kinetic energy, you can identify the particle
  - ΔE depends on β
  - $E_{Kin} = ½ mv^2$
- Detector
  - Thin counter (Plastic scintillator)
    - Position (x, y, z)=(2.5cm, 0, 0)
    - Volume 10cm(W) × 10cm(H) × 1cm(T)
  - Thick counter (Plastic scintillator)
    - Position (x, y, z)=(25cm, 0, 0)
    - Volume 10cm(W) × 10cm(H) × 40cm(T)
- Physics process (Do Not include decay)
  - Transportation
  - EMProcess
- Primary Generation
  - Generate π+,K+,p in order
  - Ekin 20MeV~400MeV
  - Beam direction (1,0,0)
- Sensitive Detector
  - Thin counter
  - Thick counter
- Make a scattering plot (2-dim histogram) between ΔE (thin counter) and total E (thick counter)