(Hands-on)
Scoring and analysis

T. Aso, NIT-Toyama

Preface

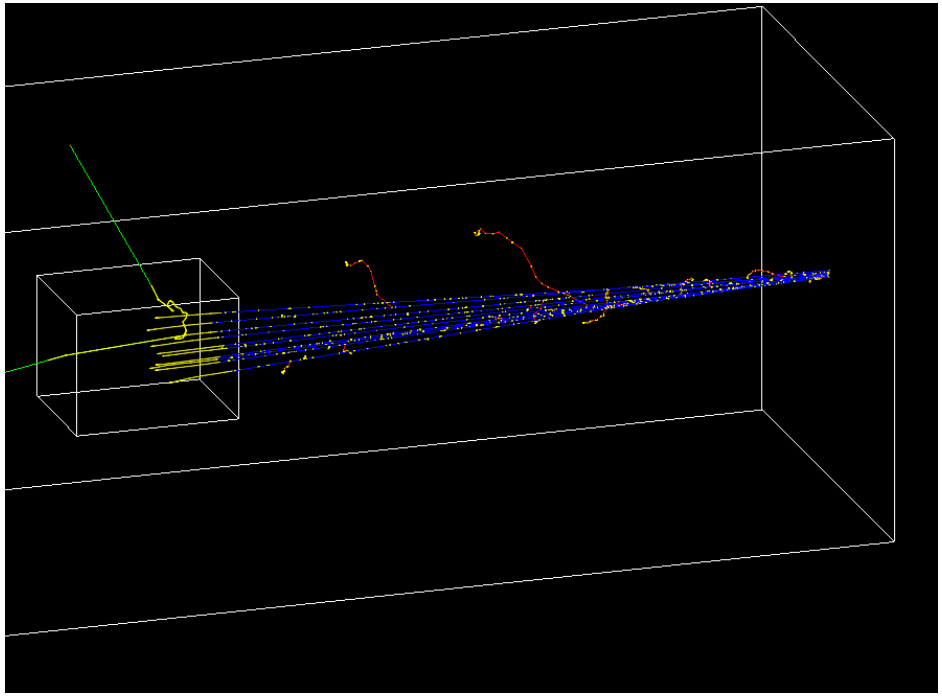
- In this Hands-on,
  - The sample code includes a concrete SD class and a concrete Hit class with the HitCollection.
    - GaletSD.hh/cc
    - GaletHit.hh/cc （GaletHitsCollection）
  - The G4AnalysisManager is used for analysis and output of scoring quantities.
- This hands-on aims to learn
  - how to utilize the Hit information (in GaletHit)
  - how to score as histograms and ntuples

Download and Build the Hands-on example code

- Download
  - Galet-MedEx-02.tar.gz

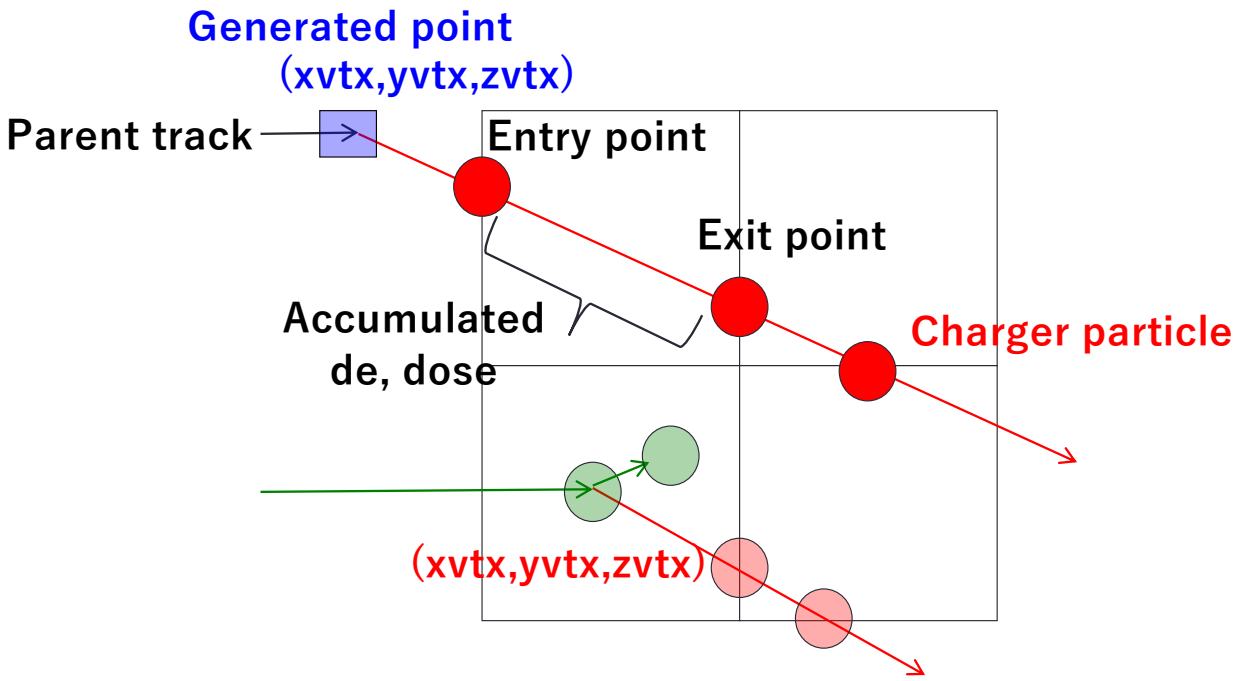
```
$ cd
$ tar xvf  ~/Downloads/Galet-MedEx-02.tar
  (or $ tar zxvf  Galet-MedEx-02.tar.gz )

$ mkdir Galet-MedEx-02-build
$ cmake ../Galet-MedEx-02
$ make
$ ./Galet
Session: /run/beamOn 10
```



GaletHit

- Storing particle information in a volume



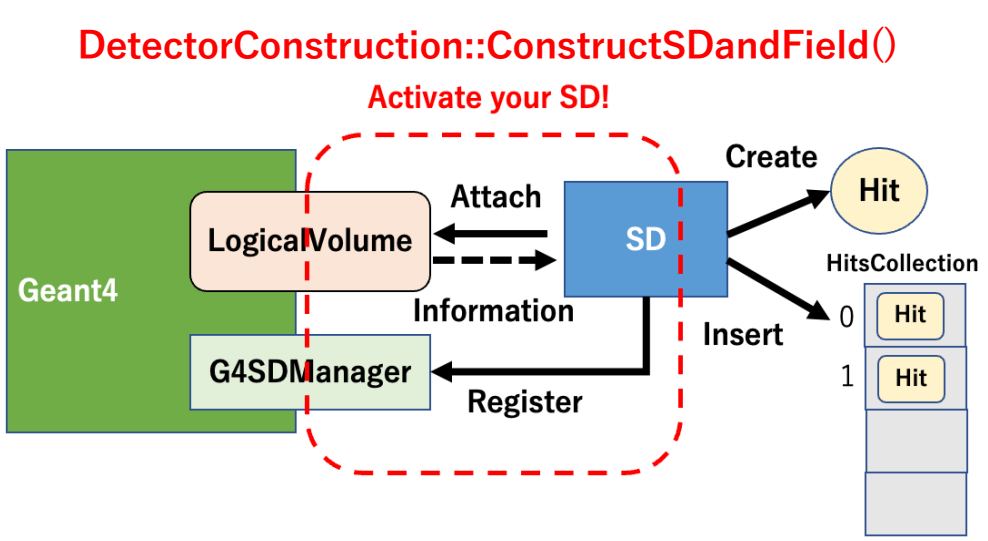
GaletHit

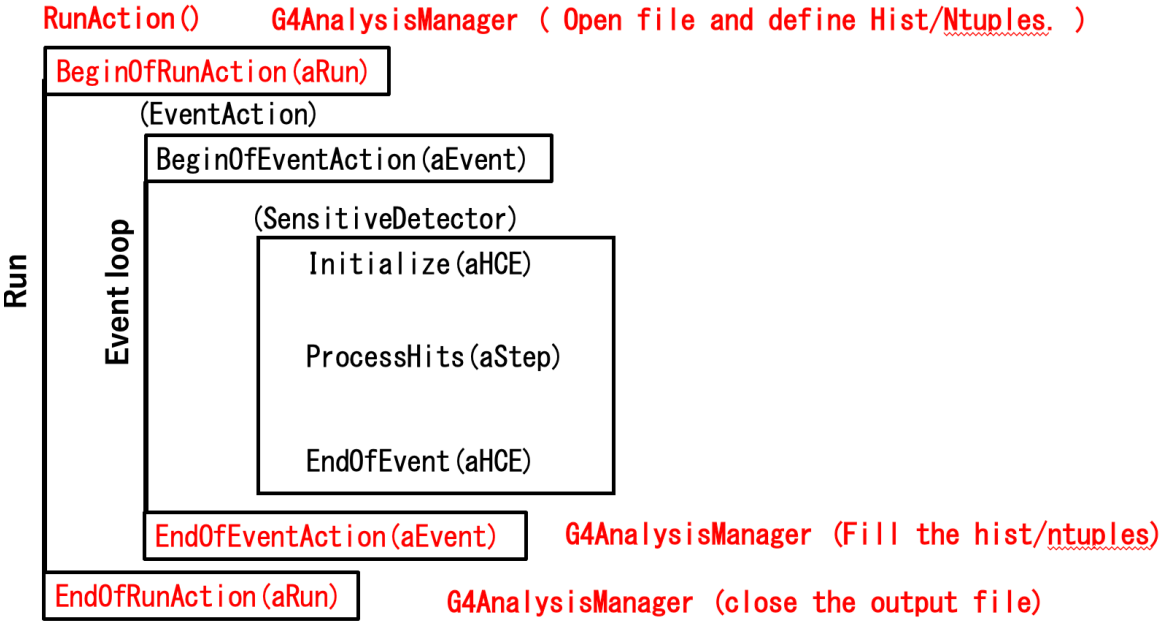
method	Description	method	Description
G4int GetPID()	Particle ID (PDG code)	G4double GetDose()	Dose
G4int GetAtomicNumber()	Z	G4double GetStepLength()	Step length
G4int GetAtomicMass()	A	G4int GetStepCount()	Step count
G4int GetTrackID()	Track ID	G4ThreeVector GetPrimaryVertex()	Primary vertex position
G4ThreeVector GetVertex()	Generated point	G4int GetParentTrackID()	Parent Track ID
G4int GetUnitXID() G4int GetUnitYID() G4int GetUnitZID()	Voxel ID	G4int GetParentPID()	Parent PID
G4ThreeVector GetEntryPoint() G4ThreeVector GetExitPoint()	Entry point Exit point		
G4ThreeVector GetMomentum()	Momentum at entry point		
G4double GetGlobalTime()	Global time at entry point		
G4double GetElectronicLET()	LET at entry point. [proton only]		
G4double GetEntryKinE() G4double GetExitKinE()	Kinetic energy at entry point Kinetic energy at exit point		
G4double GetEnergyDeposit()	Energy deposit		

PDG code
e- 11,
e+ -11,
proton 2212,
neutron 2112,
gamma 22

Ions
+-100ZZZAAAI

Overview, what we need to do.





Simulation and analysis conditions

- Scoring in Water phantom (DetectorConstruction)
  - Half length : 15 cm x 15 cm x 20 cm,
  - Segmentation: 150 , 150, 200 (2 mm cubic voxel)

- Proton beam with a gaussian beam field (gps.mac)
  - 5 mm gaussian spot at (0., 0., 1500) mm
  - 20 mrad divergence in x and y,
  - 150 MeV proton w/ 0.17 MeV gaussian energy fluctuation

Scoring

- Score
  Black lines are already implemented.
  Red lines are required to be implemented in this Hands-on.

  - 1D depth energy-deposit distribution (iz - Sum(de)/keV)
  - 1D depth dose distribution (iz - Sum(dose)/Gray for protons)
  - 1D depth dose distribution (iz - Sum(dose)/Gray for electrons)
  - 1D depth dose distribution (iz - Sum(dose)/Gray for others)

  - 2D Dose in x-y (Dose/Gray, x-y in mm)

  - Ntuple
    - eventID, ix,iy,iz, de
    - pid
    - stepL (step length)
    - LET (Electronic LET)

Outline

- The user needs to do following steps.

1. Activate SD by attaching to the target logical volume
   (**DetectorConstruction::SetSDandFied()**)
2. Create histograms and a ntuple (**RunAction::RunAction()**)
3. Calculate quantities from Hits and fill histograms and the ntuple
   (**EventAction::EndOfEventAction()**)
4. Rebuild and run the application
5. Check the results in the output file

1. Activate SD by attaching the target LogicalVolume

- Edit **DetecotrConstruction.cc**

  $ code ~/Galet-MedEx-02/src/DetectorConstruction.cc (!!g4user)

Check the parameters of Water Phantom

```
G4VPhysicalVolume* DetectorConstruction::DefineVolumes(){
… snippet …

 // WaterPhantom
  GaletPhantom* phantom = new GaletPhantom("phantomVoxelLV");
  phantom->SetWaterPhantom(15.*cm,15.*cm,20.*cm,150,150,200);
  //                       dxHalf,dyHalf,dzHalf,Nx,Ny,Nz
  phantom->SetVisOn(true);
  G4LogicalVolume* phantomLV = phantom->ConstructPhantom();
  G4RotationMatrix* rotation = new G4RotationMatrix();
  rotation->rotateY(180.*degree);
  new G4PVPlacement(rotation, G4ThreeVector(0,0,-20.0*cm),
                    phantomLV, "PhantomPV",
                    worldLV, false, 0, fCheckOverlaps);

  //
  // Always return the physical World
  //
  return worldPV;
}
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```

“phantomVoxelLV” is the logical volume name of voxel in this case.

1. Activate SD by attaching the target LogicalVolume

- Edit **DetecotrConstruction.cc**

Remove
/*
*/
and activate
these lines

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
void DetectorConstruction::ConstructSDandField(){
  G4SDManager::GetSDMpointer()->SetVerboseLevel(1);
  //
  // Sensitive detectors
  // (HE4)
  /*                                  SDname        HCName
  auto galetSD = new GaletSD("galetSD", "GaletHitsCollection",0);
  galetSD->SetDepth(1,2,0); // Geometric hierarchy of ix, iy, iz
  G4String lvName = "phantomVoxelLV";
  //
  G4SDManager::GetSDMpointer()->AddNewDetector(galetSD);   Register SD to SDManager
  SetSensitiveDetector(lvName,galetSD);                    Attach SD to the LogicalVolume by Name
  */
  //
  // Magnetic Field  will be here
  //


}
```

Don't forget to save the file:
Ctrl-s

2. Create histograms and a ntuple

- Edit **RunAction.cc**
  - vscode: [File] -> [Open file] -> RunAction.cc

```
RunAction::RunAction() : G4UserRunAction(){
… snippet
 auto analysisManager = G4AnalysisManager::Instance();
… snippet
 //
 // Book histograms, ntuple
 //
 // Creating 1-D histograms (HID starts from 100)
 G4int h1Id = 0;
 h1Id = analysisManager->CreateH1("H1_1","Depth-edep (iz)", 150, 0., 150);
 G4cout << " H1  Id " << h1Id<<G4endl;
 // (HE4)
 h1Id = analysisManager->CreateH1("H1_2","Depth-edep proton (iz)",200,0.,200);
 h1Id = analysisManager->CreateH1("H1_3","Depth-edep electron (iz)",200,0.,200);
 h1Id = analysisManager->CreateH1("H1_4","Depth-edep others (iz)",200,0.,200);

 // Creating 2-D histograms (HID starts from 200)
…cont'd
```

Define these new 1D-hist.

Keep in mind

h1Id = 100

h1Id = 101
h1Id = 102
h1Id = 103

2. Create histograms and a ntuple

- Edit **RunAction.cc**

```
RunAction::RunAction() : G4UserRunAction(){
… snippet

  // Creating ntuple  (NtupleId starts from 1000)
  G4int ntupleId = analysisManager->CreateNtuple("Galet", "Galet");
  analysisManager->CreateNtupleIColumn(ntupleId,"evno");
  analysisManager->CreateNtupleIColumn(ntupleId,"ix");
  analysisManager->CreateNtupleIColumn(ntupleId,"iy");
  analysisManager->CreateNtupleIColumn(ntupleId,"iz");
  analysisManager->CreateNtupleFColumn(ntupleId,"de");
  analysisManager->CreateNtupleIColumn(ntupleId,"pid");
  analysisManager->CreateNtupleFColumn(ntupleId,"stepL");
  analysisManager->CreateNtupleFColumn(ntupleId,"LET");
  analysisManager->FinishNtuple(ntupleId);
  G4cout << " Ntuple Id " << ntupleId<<G4endl;
  //
}
```

Keep in mind

ntuple ID = 1000

column ID = 0

Add three values in the ntuple

column ID = 5
column ID = 6
column ID = 7

Don't forget to save the file:
Ctrl-s

3. Calculate quantities from Hits and fill histograms and the ntuple

- Edit EventAction.cc

```
void EventAction::EndOfEventAction(const G4Event* event){
  // Print per event (modulo n)
  //
  //  UI:  /run/printProgress  <n>
  G4int eventID = event->GetEventID();
  G4int printModulo = G4RunManager::GetRunManager()->GetPrintProgress();
  if ( ( printModulo > 0 ) && ( eventID % printModulo == 0 ) ) {
    G4cout << "---> End of event: " << eventID << G4endl;
  }
  // Get hits collections IDs (only once)
  if ( fHCID == -1 ) {
    fHCID
      = G4SDManager::GetSDMpointer()->GetCollectionID("galetSD/GaletHitsCollection");
    G4cout << " fHCID = " << fHCID << G4endl;
  }
  //
  // Skip following code by return if no hits collection
  if ( fHCID < 0 ) return;
  // Get hits collections
  auto HC = GetHitsCollection(fHCID, event);
  if ( !HC ) return;
…cont'd
```

"Sdname" + "/" +"HCName"

3. Calculate quantities from Hits and fill histograms and the ntuple

- Edit **EventAction.cc**

```
void EventAction::EndOfEventAction(const G4Event* event){
…snippet
  //
  // get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // Fill histgrams. Loop over for all of the Hits.
  for ( size_t  i = 0; i < HC->entries() ; i++){
    GaletHit* Hit = (*HC)[i];
    //
    G4double  de  = Hit->GetEnergyDeposit()/keV;
    G4double  dose= Hit->GetDose()/gray;
    const G4ThreeVector& pos = Hit->GetEntryPoint()/mm;
    G4double  x = pos.x();
    G4double  y = pos.y();
    G4double  z = pos.z();
    G4int    ix = Hit->GetUnitXID();
    G4int    iy = Hit->GetUnitYID();
    G4int    iz = Hit->GetUnitZID();
    G4int  pid  = Hit->GetPID();
    G4int  stepL= Hit->GetStepLength()/mm;
    G4double let = Hit->GetElectronicLET()/(keV/mm);
    //
…cont'd
```

Add three lines to get values from the Hit

3. Calculate quantities from Hits and fill histograms and the ntuple

- Edit EventAction.cc

Add these lines to fill the 1D Histograms

```
void EventAction::EndOfEventAction(const G4Event* event){
…snippet
  // Fill histgrams. Loop over for all of the Hits.
  for ( size_t i = 0; i < HC->entries() ; i++){
    GaletHit* Hit = (*HC)[i];
    //
    … snippet
    //
    // fill 1D hist
    analysisManager->FillH1(100,iz,de); // Depth-de
    if ( pid == 2212 ) {
       analysisManager->FillH1(101,iz,de);
    }else if ( pid == 11 || pid == -11 ) {
       analysisManager->FillH1(102,iz,de);
    }else{
       analysisManager->FillH1(103,iz,de);
    }
…cont'd
```

2212 proton
11 electron
-11 positron
22 gamma
+-100ZZZAAAI Ions

hid must be identical to the hist. definition.

3. Calculate quantities from Hits and fill histograms and the ntuple

```
void EventAction::EndOfEventAction(const G4Event* event){
…snippet
  // Fill histgrams. Loop over for all of the Hits.
  for ( size_t i = 0; i < HC->entries() ; i++){
    GaletHit* Hit = (*HC)[i];
    //
    … snippet
       // fill 2D
       if ( iz == 0 ) analysisManager->FillH2(200,x,y,dose);
       // fill ntuple                          ntupleId,columnId,value
       analysisManager->FillNtupleIColumn(1000,0, eventID);
       analysisManager->FillNtupleIColumn(1000,1, ix);
       analysisManager->FillNtupleIColumn(1000,2, iy);
       analysisManager->FillNtupleIColumn(1000,3, iz);
       analysisManager->FillNtupleFColumn(1000,4, de);
       analysisManager->FillNtupleIColumn(1000,5, pid);
       analysisManager->FillNtupleFColumn(1000,6, stepL);
       analysisManager->FillNtupleFColumn(1000,7, let);
       analysisManager->AddNtupleRow(1000);
    }  // loop end of HC-
>entries()
}
```

Add these lines to fill the 1D Histograms

Don't forget to save the file: Ctrl-s

4. Rebuild and run the application

- Ensure saving all modified files
- then,

  $ make
- Test run

  $ ./Galet

  Session: /rub/beamOn 10

  Session: exit
- Run in batch mode

  $ cat run0.mac <= Check commands in run0.mac

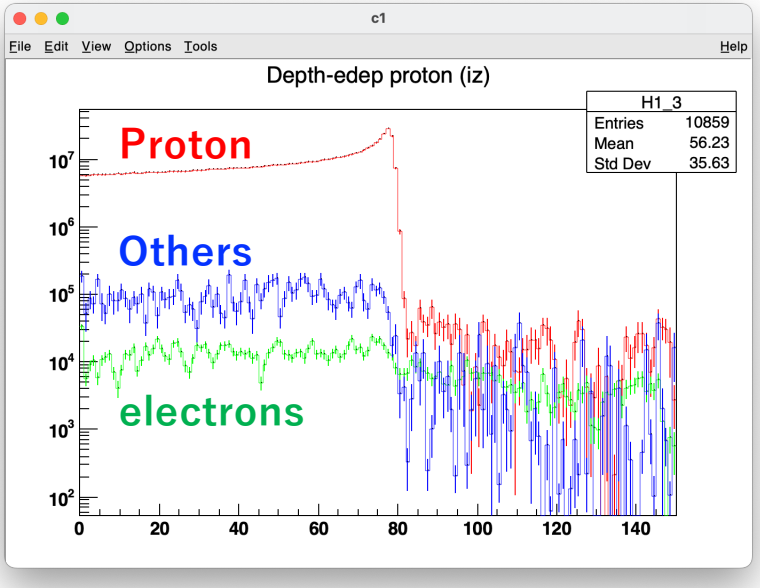
  $ ./Galet -m run0.mac

run0.mac

```
#
## Mandatory for initialization
/control/execute init.mac
#
## Start simulation
/run/beamOn 5000
```

5. Check the results in the output file

```
$ root  Galet.root
root[] .ls
   TFile** Galet.root
    TFile* Galet.root
     KEY: TTree Galet;1 Galet
     KEY: TH1D H1_1;1 Depth-edep (iz)
     KEY: TH1D H1_2;1 Depth-edep proton (iz)
     KEY: TH1D H1_3;1 Depth-edep electron (iz)
     KEY: TH1D H1_4;1 Depth-edep others (iz)
     KEY: TH2D H2_1;1 x-y dose at iz=0
root[] H1_2->SetLineColor(2); H1_2->Draw("h")
root[] c1->SetLogy()
root[] H1_3->SetLineColor(3); H1_3->Draw("hsame");
root[] H1_4->SetLineColor(4); H1_4->Draw("hsame");
```



5. Check the results in the output file

22 gamma
2112 neutron
10020 Z = 1, A = 2
80160 Z= 8, A = 16

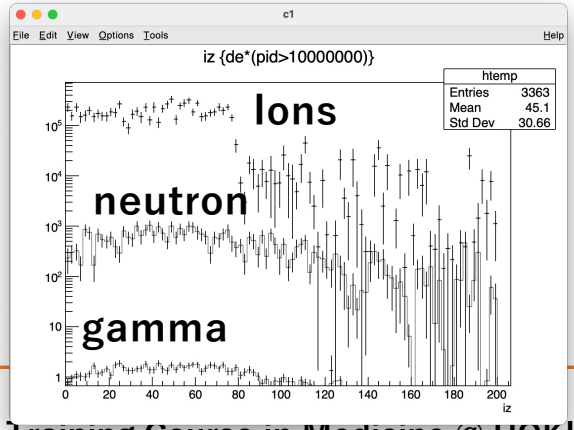
- What kind of species does contribute to the dose (Others)?
- Let's check PID in the ntuple

```
root[] Galet->Scan("pid%10000000","!(pid==2212||abs(pid)==11)")
```

*	71 *	2112 *
*	72 *	2112 *
*	73 *	2112 *
*	74 *	2112 *
*	75 *	2112 *
*	76 *	2112 *
*	77 *	2112 *
*	78 *	2112 *
*	79 *	10020 *
*	80 *	22 *
*	81 *	22 *
*	94 *	80160 *
*	933 *	80160 *
*	953 *	20040 *
*	954 *	20040 *
*	955 *	20040 *
*	963 *	80160 *
*	967 *	2112 *
*	968 *	2112 *
*	1062 *	80160 *
*	1076 *	22 *
*	1077 *	22 *
*	1084 *	70140 *
*	1115 *	60120 *
*	1116 *	20040 *

- How is the contribution from gamma, neutron, and ions

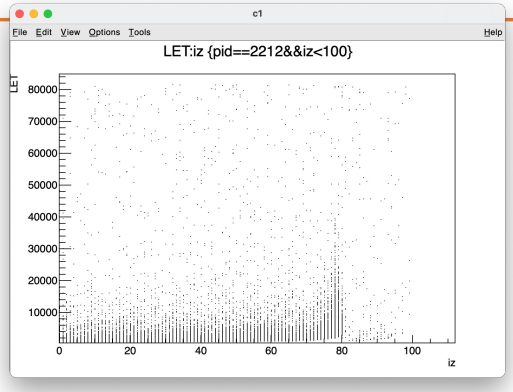
```
root [] Galet->Draw("iz","de*(pid>10000000)")
root [] Galet->Draw("iz","de*(pid==2112)","hsame")
root [] Galet->Draw("iz","de*(pid==22)","hsame")
```



5. Check the results in the output file

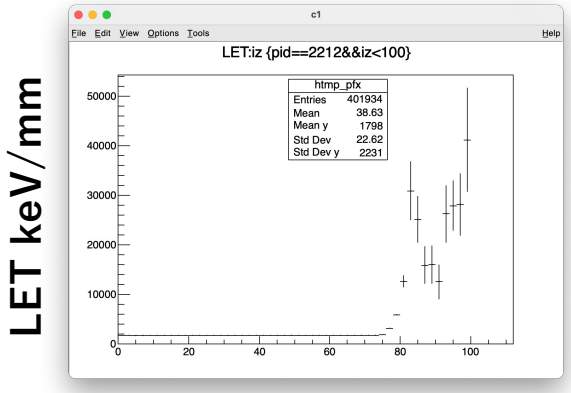
- **Check LET of proton in ntuple**

```
root[] c1->SetLogy(0)        <= to linear scale
root[] Galet->Draw("iz:LET","pid==2212&&iz<100")
```



- **make a x-profile**

```
root[] Galet->Draw("iz:LET >> htmp","pid==2212&&iz<100")
root[] htmp->ProfileX()
root[] htmp_pfx->Draw()
```



LET keV/mm

iz/2mm

Summary

- Using GaletSD and GaletHit, you can:
  - Attach the GaletSD to the target LogicalVolume
    - We attached to the 3D segmented voxel phantom
    - But it can be used to the other type of volumes
  - Define histograms and ntuples with the G4AnalysisManager
    - Fill those by getting information of hits.
    - 1D or 2D histograms are good for obtaining distributions
    - Ntuple is good for offline analysis