Session 9 : Exercise

Code for download: session9_start.tar.gz

Start code: session8_start with updated geometry, more collections for the calorimeter, and added ScreenSD. Bounds for calorimeter histogram have been changed also. Prints for hit have been removed, in order to run statistics without being annoyed by long outputs.

Exercise 9a (physics):

In the start code, a new volume has been added a new volume, a thin screen, far after the calorimeter. For this reason, the world volume has been made larger than in the previous exercices, as can be seen in the detector construction:

```
// Make world larger for this example:
hx += 5*m;
hy += 5*m;
hz += 5*m;
```

This screen will be used for counting particles that exit from the calorimeter. It will be made sensitive, but we will not create hits and hits collections, but just store in the ntuple the quantities we are interested in.

To this screen logical volume, attach a sensitive detector that you will have to complete.
In the ProcessHits() of this sensitive detector, you will fill a Root ntuple:

```
// Store hit in the ntuple
G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
analysisManager->FillNtupleIColumn(fNtupleId, 0, ID);
analysisManager->FillNtupleIColumn(fNtupleId, 1, pdgCode);
analysisManager->FillNtupleDColumn(fNtupleId, 2, Ekin/MeV);
analysisManager->FillNtupleDColumn(fNtupleId, 3, localPosition.x()/cm);
analysisManager->FillNtupleDColumn(fNtupleId, 4, localPosition.y()/cm);
analysisManager->FillNtupleDColumn(fNtupleId, 5, time/ns);
analysisManager->AddNtupleRow(fNtupleId);
```

where ID is the track ID, pdgCode its PDG code, Ekin the kinetic energy, localPosition is the G4ThreeVector of local coordinate in the frame of the screen (see EDChamberSD to see how to get these coordinates, for example), time is the hit time, and creatorModelID and ID telling which model has created this track. Note that this model is internal to a physics process, and this

ID is defined only in a few cases (too sad…).
The creator model can be obtained as:
G4int creatorModelID = track->GetCreatorModelID();
The full list of model ID is printed at the beginning of the job
(and indeces change depending on the physics list).

We will run quite statistics (10000 protons) several times and here the
MT is quite useful.
Given each thread produces its own ntuple file, a useful Root command
is the « chain »:

```
root> TChain myChain("Screen");
root> myChain.Add("ED_t0.root");
root> myChain.Add("ED_t1.root");
root> myChain.Add("ED_t2.root");
root> myChain.Add("ED_t3.root");
```

then you can use this chain myChain is if it was a ntuple like Screen, eg:

```
root> myChain.Draw("Ekin");
```

to see Ekin for a given particle, eg. proton:

```
root> myChain.Draw("Ekin","PdgCode==2212");
```

Exercise 9b (physics):

Change of neutron behaviour with FTFP_BERT versus FTFP_BERT_HP:

When the job starts, there is a long print out of the physics list
configuration. For the hadronics, this starts with the lines:

```
  ======================================================================
                    HADRONIC PROCESSES SUMMARY (verbose level 1)
  ...
  ...
```

Study the differences between these prints for neutrons when using
FTFP_BERT and FTFP_BERT but with the high precision neutron physics.
In particular, pay attention the differences for energies below 20 MeV.

Run 10k protons with FTFP_BERT and 10k protons FTFP_BERT_HP.
Compare for neutrons (pdgCode == 2112) the difference in of the
Ekin and time spectra of ntuple « Screen ».

NB : the long prints at the beginning of the job contain lots of
information. As for HP versus non HP, you may look at the differences
when using -say- FTFP_BERT versus FTFP_BERT_PEN, the print starting with:

```
  phot:   for  gamma    SubType= 12  BuildTable= 0
        LambdaPrime table from 200 keV to 10 TeV in 154 bins
  ...
  ...
```

Note in particular the differences about Rayl, Deexcitation.

Exercise 9c (geometry persistency):

- Implement the newly added command-line option [-g gdmlFile] in the main program: when the GDML file is provided export the geometry in GDML.
  Hint: The world volume can be accessed in main in this way:

```
G4LogicalVolume* world
= G4TransportationManager::GetTransportationManager()
->GetNavigatorForTracking()->GetWorldVolume()->GetLogicalVolume());
```

- The export GDML geometry can be then imported in ROOT:

```
root [0] TGeoManager::Import("geometry.gdml");
```

  the following commands allow to view the geometry in OpenGL ROOT visualization:

```
root [1] gGeoManager->SetVisLevel(10);
root [2] gGeoManager->GetTopVolume()->SetVisContainers(kTRUE);
root [3] gGeoManager->GetTopVolume()->Draw("ogl");
```

Solution: session9_solution.tar.gz
