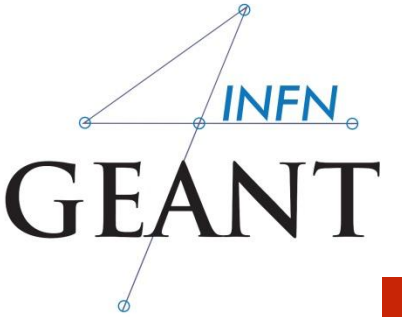


Interaction with the Geant4 Kernel-II

Davide Chiappara
University of Padova (UNIPD)
Istituto Nazionale di Fisica Nucleare (INFN)

Geometry - implementation basics

Implement a class inheriting from the abstract base class **G4VUserDetectorConstruction**:

```
class MyDetector : public G4VUserDetectorConstruction {
public:
    virtual G4VPhysicalVolume* Construct();      // required

    virtual void ConstructSDAndField();          // optional
    // ...
};
```

Create an instance in the main program:

```
MyDetector* detector = new MyDetector();
runManager->SetUserInitialization(detector);
```

Note: Split the implementation into more classes and methods! (good programming practice)

especially for complex geometries!

Note2: you should not delete the **MyDetector** instance! Run manager does that automatically.

Part I: Sensitive detectors

Sensitive Detector (SD)

A **logical volume** becomes **sensitive** if it has a pointer to a sensitive detector (**G4VSensitiveDetector**)

- A sensitive detector can be instantiated several times, where the instances are assigned to different logical volumes.
  - Note that SD objects must have unique detector names
  - A logical volume can **only** have **one SD object** attached (But you can implement your detector to have many functionalities)
- **Two possibilities** to make use of the SD functionality:
- Create your own sensitive detector (using class inheritance)
  - Highly customizable (see Kernel III)
- Geant4 built-in tools: **Primitive scorers**

Adding sensitivity to a logical volume

Create an **instance** of a **sensitive detector** and **register it** to the **SensitiveDetector Manager**

Assign the pointer of your SD to the logical volume of your detector geometry

Must be done in **ConstructSDandField( )** of the user geometry class

```
G4VSensitiveDetector* mySensitive
   = new MySensitiveDetector(SDname="/MyDetector");

G4SDManager* sdMan =G4SDManager::GetSDMpointer();
sdMan->AddNewDetector(mySensitive);

SetSensitiveDetector("LVname",mySensitive);
```

Create the instance

Register the SD

Name of the logical volume

Assign to logical volume

Adding sensitivity to a logical volume - variant

Create an **instance** of a **sensitive detector** and **register it** to the **SensitiveDetector Manager**

Assign the pointer of your SD to the logical volume of your detector geometry

Must be done in **ConstructSDandField( )** of the user geometry class

```
G4VSensitiveDetector* mySensitive
   = new MySensitiveDetector(SDname="/MyDetector");

G4SDManager* sdMan =G4SDManager::GetSDMpointer();
sdMan->AddNewDetector(mySensitive);

logicVol->SetSensitiveDetector(mySensitive);
```

Create the instance

Register the SD

Pointer to logical volume

Assign to logical volume

Part II: Native Geant4 scoring

Extract Useful information

Geant4 provides a number of **primitive scorers**, each one accumulating one physics quantity (e.g. total dose) for an event

This is alternative to the **customized** sensitive detectors (see lecture on *Kernel III*), which can be used with full flexibility to gain complete control

It is **convenient** to use primitive scorers instead of user-defined sensitive detectors when:

- you are not interested in recording each individual step, but **accumulating physical quantities** for an event or a run
- you have **not too many** scorers

G4MultiFunctionalDetector

G4MultiFunctionalDetector is a concrete class derived from G4VSensitiveDetector

It should be assigned to a logical volume as a kind of (ready-for-the-use) **sensitive detector**

It takes an arbitrary number of **G4VPrimitiveScorer** classes, to define the scoring quantities that you need

- Each **G4VPrimitiveScorer** accumulates one physics quantity for each physical volume
- E.g. ***G4PSDoseScorer*** (a concrete class of G4VPrimitiveScorer provided by Geant4) accumulates dose for each cell

By using this approach, no need to implement sensitive detector and hit classes!

G4VPrimitiveScorer

Primitive scorers (classes derived from G4VPrimitiveScorer) have to be registered to the G4MultiFunctionalDetector

->RegisterPrimitive( ),

->RemovePrimitive( )

They are designed to score **one kind of quantity** (surface flux, total dose) and to generate **one hit collection** per event

automatically **named** as

<MultiFunctionalDetectorName>/<PrimitiveScorerName>

hit collections can be retrieved in the EventAction or RunAction (as those generated by sensitive detectors)

do not share the same primitive scorer object among multiple G4MultiFunctionalDetector objects (results may mix up!)

Create as many instances of the scorer as needed

For Example…

```
MyDetectorConstruction::ConstructSDandField()
{
   G4MultiFunctionalDetector* myScorer = new
         G4MultiFunctionalDetector("myCellScorer");     // Instantiate detector

    myCellLog->SetSensitiveDetector(myScorer);          // Attach to volume

    G4VPrimitiveScorer* totalSurfFlux = new
         G4PSFlatSurfaceFlux("TotalSurfFlux");          // Create scorer
    myScorer->RegisterPrimitive(totalSurfFlux);         // Attach it

    G4VPrimitiveScorer* totalDose = new
         G4PSDoseDeposit("TotalDose");                  // Create scorer
    myScorer->RegisterPrimitive(totalDose);             // Attach it

}
```

Some useful Primitive Scorers

Concrete Primitive Scorers (taken from ***Application Developers Guide 4.4.5***)

Track length

G4PSTrackLength, G4PSPassageTrackLength

Deposited energy

G4PSEnergyDepsit, G4PSDoseDeposit

Current/Flux

G4PSFlatSurfaceCurrent,
G4PSSphereSurfaceCurrent,G4PSPassageCurrent, G4PSFlatSurfaceFlux,
G4PSCellFlux, G4PSPassageCellFlux

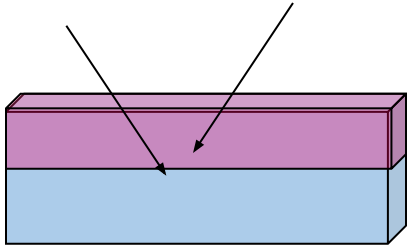
Others

G4PSMinKinEAtGeneration, G4PSNofSecondary, G4PSNofStep,
G4PSCellCharge

A Closer Look at some scorers

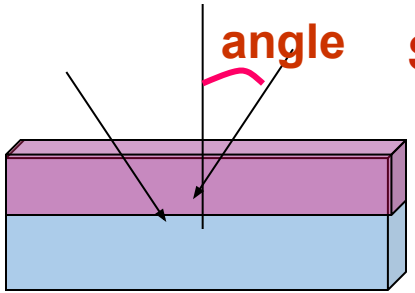
SurfaceCurrent :

Count number of injecting particles at defined surface.



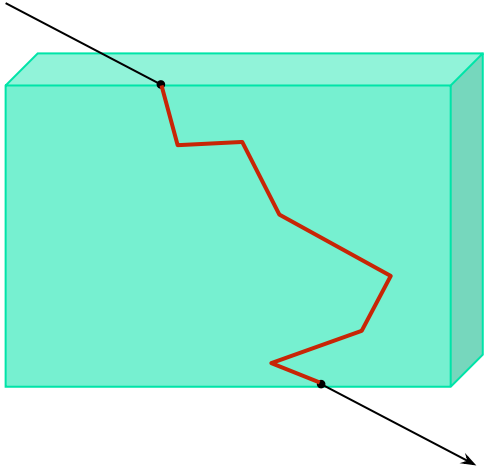
CellFlux :

Sum of L / V of injecting particles in the geometrical cell.



SurfaceFlux :

Sum up 1/cos(angle) of injecting particles at defined surface



G4VSDFilter

A **G4VSDFilter** can be attached to **G4VPrimitiveScorer** to define which kind of tracks have to be scored (e.g. one wants to know surface flux of protons only)

G4SDChargeFilter (accepts only charged particles)

G4SDNeutralFilter (accepts only neutral particles)

G4SDKineticEnergyFilter (accepts tracks in a defined range of kinetic energy)

G4SDParticleFilter (accepts tracks of a given particle type)

G4VSDFilter (base class to create user-customized filters)

For Example…

```
MyDetectorConstruction::ConstructSDandField()
{
     G4VPrimitiveScorer* protonSurfFlux
          = new G4PSFlatSurfaceFlux("pSurfFlux");   // Create primitive scorer
     G4VSDFilter* protonFilter = new
          G4SDParticleFilter("protonFilter");       // Create the particle filter
     protonFilter->Add("proton");                   // Add "protons" to the filter

     protonSurfFlux->SetFilter(protonFilter);       // Register the filter to scorer

     myScorer->RegisterPrimitive(protonSurfFlux);   // Register scorer to MFD
}
```

How to retrieve information - part 1

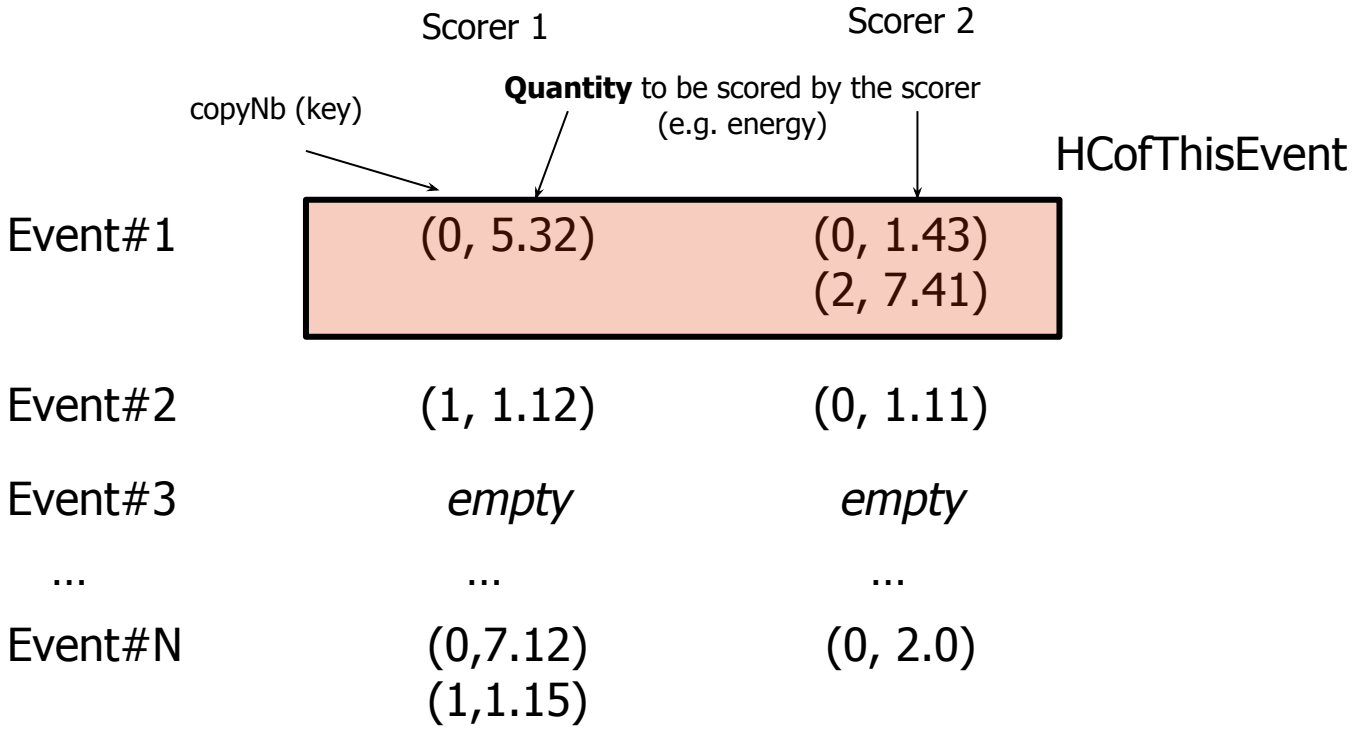
At the end of the day, one wants to retrieve the information from the scorers

- True also for the customized hits collection (**Kernel III**)

Each scorer creates a **hit collection**, which is attached to the **G4Event** object

- Can be retrieved and read at the end of the event, using an integer ID
- Hits collections mapped as **G4THitsMap<G4double>*** so can loop on the individual entries
- Operator **+=** provided which automatically sums up all hits (no need to loop manually)

How to retrieve information - part 2



How to retrieve information – recipe

```
//needed only once: Get the collection ID from the collection name.
G4int collID = G4SDManager::GetSDMpointer()
  ->GetCollectionID("myCellScorer/TotalSurfFlux");

// Get all HC available in this event.
G4HCofThisEvent* HCE = event->GetHCofThisEvent();
```

	Scorer 1	Scorer 2
Event#1	(0, 5.32)	(0, 1.43) (2, 7.41)

How to retrieve information – recipe

```
//needed only once: Get the collection ID from the collection name.
G4int collID = G4SDManager::GetSDMpointer()
  ->GetCollectionID("myCellScorer/TotalSurfFlux");

// Get all HC available in this event.
G4HCofThisEvent* HCE = event->GetHCofThisEvent();

// Get HC with the given ID (need a cast because of C++ polimorphism)
G4THitsMap<G4double>* evtMap =
    static_cast<G4THitsMap<G4double>*>
    (HCE->GetHC(collID));
```

	Scorer 1	Scorer 2
Event#1	(0, 5.32)	(0, 1.43) (2, 7.41)

How to retrieve information – recipe

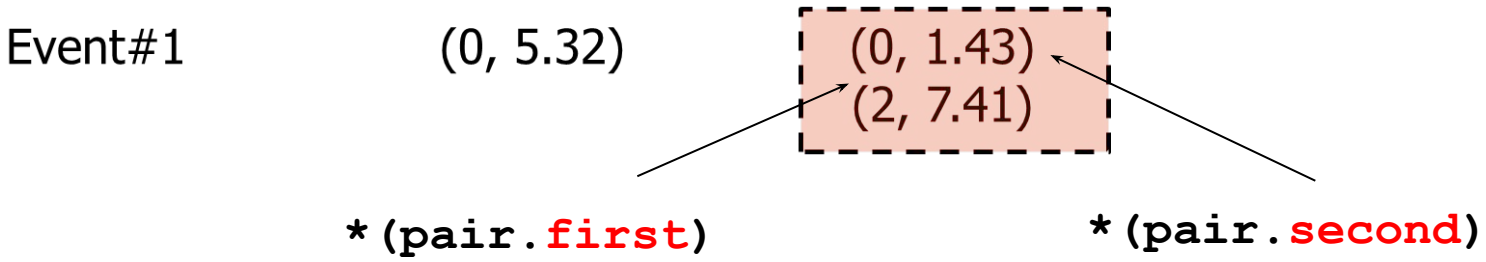
```
//needed only once: Get the collection ID from the collection name.
G4int collID = G4SDManager::GetSDMpointer()
  ->GetCollectionID("myCellScorer/TotalSurfFlux");

// Get all HC available in this event.
G4HCofThisEvent* HCE = event->GetHCofThisEvent();

// Get HC with the given ID (need a cast because of C++ polimorphism)
G4THitsMap<G4double>* evtMap =
    static_cast<G4THitsMap<G4double>*>
    (HCE->GetHC(collID));

// Loop over the entries of the HC. key is copy number
for (auto pair : *(evtMap->GetMap())) {
    G4double flux = *(pair.second);
    G4int copyNb  = *(pair.first);
}
```

How to retrieve information – recipe




```
for (auto pair : *(evtMap->GetMap())) {
    G4double flux = *(pair.second);
    G4int copyNb  = *(pair.first);
}
```

Loop1: copyNb = 0, value = 1.43
Loop2: copyNb = 2, value = 7.41

Hands-on session

Task4

Task4c: Native scoring

Task4d: (Optional) Multi-threading

Task5 (Optional)

Very similar to 4c, but on medical physics

http://geant4.lns.infn.it/pavia2023/task4

http://geant4.lns.infn.it/pavia2023/task5