Geant4 2005 Bordeaux

Concrete Primitive Sensitivity

T.Aso[1,2], M.Asai[2], A.Kimura[3]

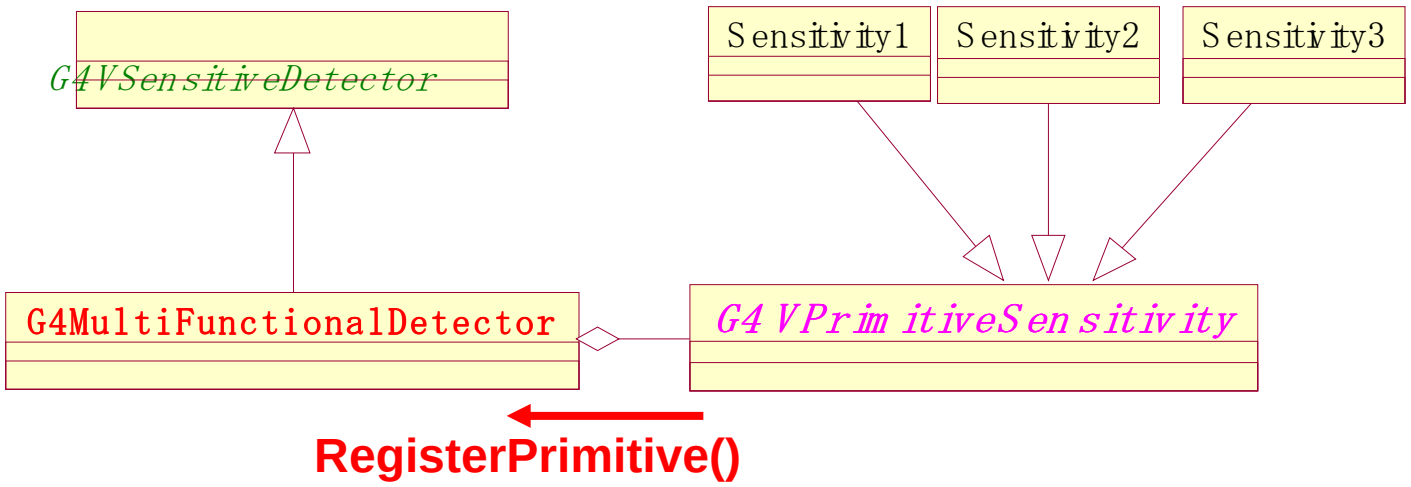
[1]Toyama National College of Maritime Technology
[2]SLAC,
[3]JST CREST

Motivation

- Up to the current version, Geant4 provides only an abstract base class (***G4VSensitiveDetector***) for the user to define his/her own detector sensitivity.
  - Various example detector classes are provided.
    - This is enough for **HEP experiments**, since their interest is mostly ***storing hits*** in their detectors in each individual event. Their detectors are quite different to each other and thus they anyway have to implement their own detector.
  - But it is not convenient for **space and medical applications**.
    - Their interest is mainly ***scoring dose/flux.*** This interest is quite common to all users.
    - Helper/Sample classes for convenient scoring is desirable.

Concrete Sensitivity Classes

- We introduce **G4MultiFunctionalDetector**, that is a concrete class derived from ***G4VSensitiveDetector***.
  - It allows the user to **register** concrete class objects of ***G4VPrimitiveSensitivity*** to define a scoring detector of his/her needs.
  - We also provide concrete primitive sensitivity classes such as **dose scoring**, **surface flux counting**, etc.



Multi-Functional Detector

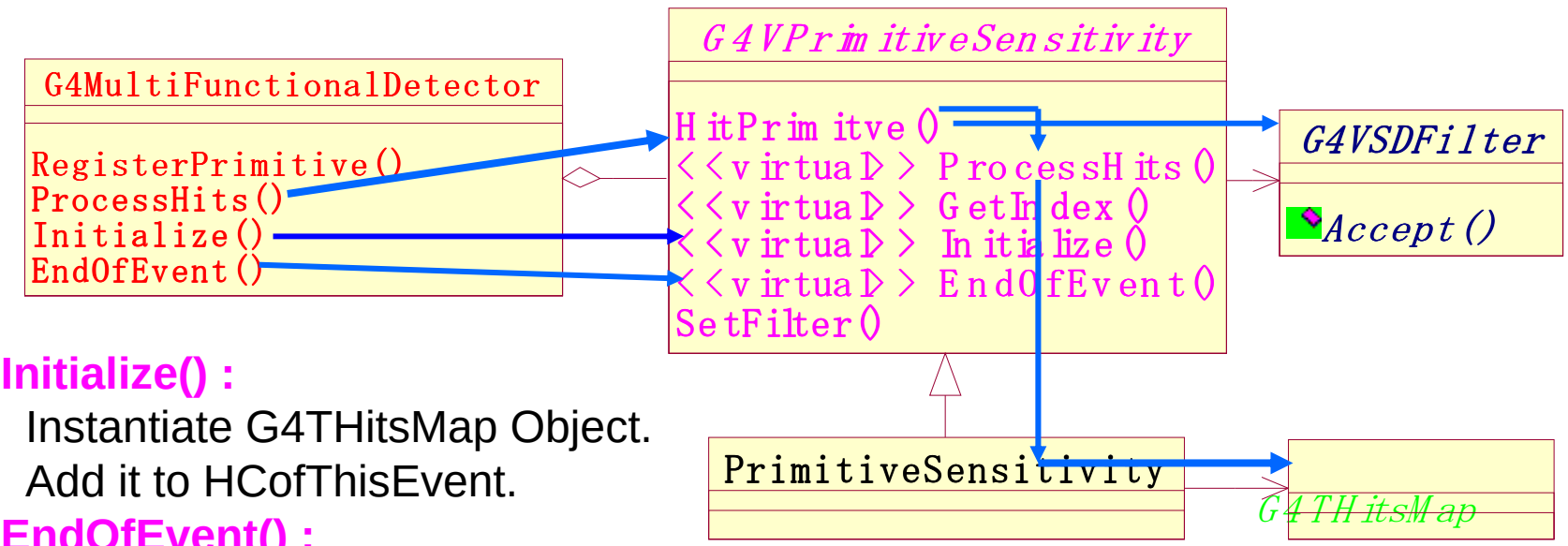
- The user can register ***one or more*** objects of ***G4VPrimitiveSensitivity*** into a G4MultiFunctionalDetector object, in order to obtain one or more physical quantities at an execution of simulation.

- G4MultiFunctionalDetector does:
  - Take care of the **collection name** of primitive sensitivity and notify to the collection name to G4SDManager, where the collection name is defined as
    collection name =
    <MultiFunctionalDetectorName>/<PrimitiveSensitivityName> **.**
  - Invoke methods of registered primitive sensitivities during a event.

Primitive Sensitivity -1-

- Each G4VPrimitiveSensitivity class is designed to *score one kind of quantity* and generate **one hits collection** per event.
  - By registering more than one objects of G4VPrimitiveSensitivity classes, G4MultiFunctionalDetector scores more than one kinds of physical quantities.
- The new template class, **G4THitsMap**, is introduced.
  - It does NOT mandate **G4VHit** concrete class to be stored, but for example **a simple double value** can be mapped with **a copy number of the geometry**.
  - It is more convenient for scoring purposes than currently provided G4THitsCollection.
- New class ***G4VSDFilter*** is introduced. It may be attached to *G4VSensitiveDetector* and/or ***G4VPrimitiveSensitivity*** to define which kinds of tracks are to be scored.

Schematic sequence



Initialize() :

Instantiate G4THitsMap Object.
Add it to HCofThisEvent.

EndOfEvent() :

HitPrimitives():

If a filter is available, the track is examined.
if the track is acceptable,

ProcessHits() :

Insert or add physical quantity into G4THitsMap object with associating index number which is obtained by **GetIndex()**.

GetIndex() return only single index number, so that it does not reply to the nested replication. In that case, the user needs to override this method on their primitive sensitivity as their needs.
Otherwise, using parameterised geometry is the easiest way.

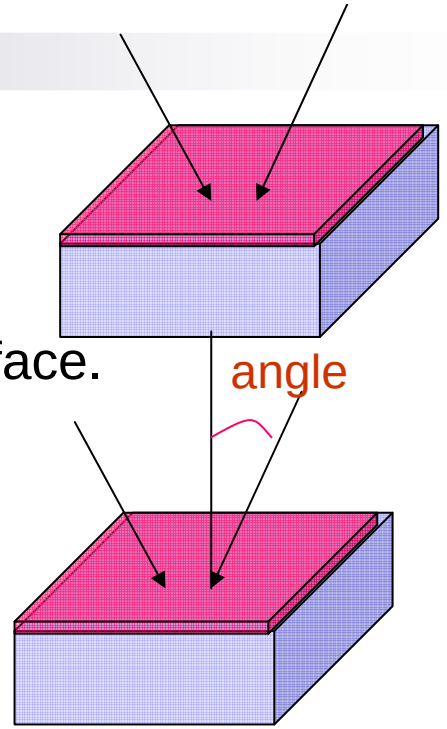
Primitive Sensitivity and Filter

We have already implemented these primitive sensitivities and filters.
The class names are still preliminary.

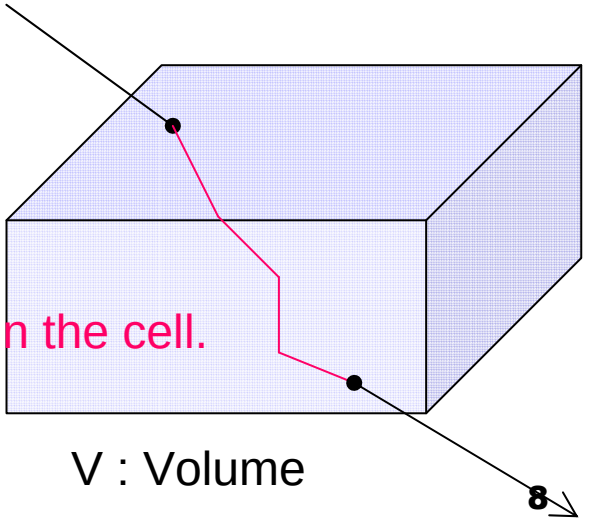
PS	**G4PSEnergyDeposit**	**Sum of deposited energy**
	G4PSDoseDeposit	**Sum of dose**
	G4PSTrackLength	**sum of step length**
	G4PSCellFlux	**TrackLength divided by volume**
	G4PSPassageCellTrackLength	**TrackLength only for through particle**
	G4PSPassageCellFlux	**CellFlux only for through particle**
	G4PSNofStep	**Number of steps**
	G4PSFlatSurfaceCurrent (G4Box)	**Number of particle crossing surface**
	G4PSFlatSurfaceFlux (G4Box)	**FlatSurfaceCurrent with incident angle collection**
	G4PSSphereSurfaceCurrent (G4Sphere)	
Filter	**G4SDParticleFilter**	**Filter by particle names**
	G4SDChargedFilter	**Filter for charged particles**
	G4SDNeutralFilter	**Filter for neutral particles**
	G4SDKineticEnergyFilter	**Filter for kinetic energy (low/high)**
	G4SDParticleWithEnergyFilter	**Combination of particle and kinetic energy**
	G4SDBoxSurfaceFilter	**Filter for surface**

Terminology

- SurfaceCurrent :
  - Count number of injecting particles at defined surface.
- SurfaceFlux :
  - Take into account injection angle of the particle.
  - Sum up 1/cos(angle) of injecting particles at defined surface



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



Accumulation of scoring in the Run.

- G4HCofThisEvent stores “HitsMap”, but it is deleted at the end of event.
- In order to avoid serious precision degradation, we have to pay attention the way for summing up.

- Current version of Geant4 has already involved the solution.
  - The “G4HCofThisEvent” object is obtained from G4SDmanager in arbitral manner from G4Event object.
  - The user can override G4Run class to involve HitsMap for accumulation during the run,
    and can sum up the HitsMap at G4Run::RecordEvent(G4Event*).
  - The user can launch user’s G4Run object from G4UserRunAction::GenerateRun().
  - The user’s G4Run object is accessible at EndOfRunAction(G4Run*) for output of the result.

Sample code - Assignments of sensitivity -

```
UserDetectorConstruction::Construct( ){   //  G4VUserDetectorConstruction
 - - - - - - - - - - sniff - - - - - - - -
  //
  // MultiFunctionalDetector
  //
  G4String mfdName;
  G4MultiFunctionalDetector* mfd = new G4MultiFunctionalDet                ux");
     G4SDManager::GetSDMpointer()->AddNewDetector(mf
     logical->SetSensitiveDetector(mfd);
  //
  // PrimitiveSensitivity
  //
  G4String psName
  G4PSFlatSurfaceFlux* gammaFlux =                    aceFlux(psName="gammaFlux") ;
     mfd->RegisterPrimitive(gamm
  //
  // Filter
   //
   G4SDParticleFilter* ga                 G4SDPartileFilter("gammaFilter");
       gammaFlt->Add("gan
       gammaFlux->SetFilter(                ctronFlt);

- - - - - - - - - - sniff - - - - - - - -
}
```

These Concrete Sensitivity Classes are provided.
The user is not forced to develop SensitiveDetector and Hit Classes.

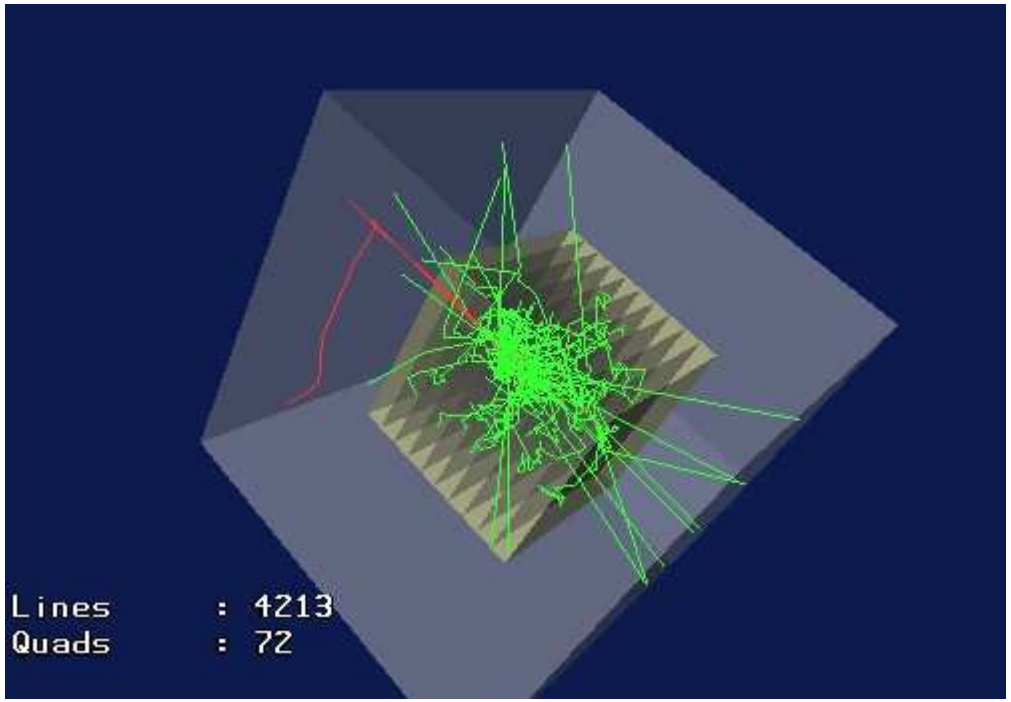
Sample code – Accumulating event score into run score -

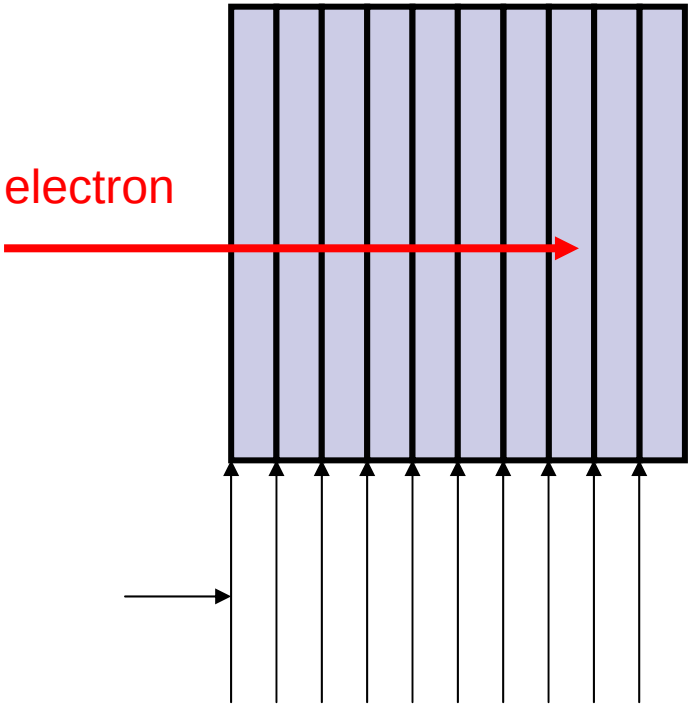
```
UserRun::UserRun(){     // Derived from G4Run.
  G4String sdName, psName;
  //
  //------  Initialize  HitsMap for accumulating run score. ---------
  fRunGFlux = new G4THisMap<G4double>(sdName="Flux",psName="gammaFlux");
  //
  //-------  Get collection ID for a event. ---------
  fCollIDGFlux = G4SDManager::GetSDMpointer()->GetCollectionID("Flux/gammaFlux");
}

UserRun::RecordEvent(const G4Event aEvent){   //--  Called at every end of events. ----
  numberOfEvent++;   // mandatory
  G4HCofThisEvent* HCE = aEvent->GetHCofThisEvent();
  if ( !HCE ) return;
  //
  // --  Get HitsMap of this event ---
  G4THitsMap<G4double>*  EvtGMapFlux =
                                        (G4THitsMap<G4double>*)(HCE->GetHC(fCollIDGFlux);

  //
  // ---- Sum up score of this event into the score of RUN. -----
  //
  fRunGFlux += EvtGMapFlux;
}
```

Summing up is simply expressed by "+=" operator.

UserRun object should be launched from G4UserRunAction::GenerateRun()

Example -1-





Water phantom 40x40x40 cm3
Sliced into 10 segments.
Primitive sensitivities are attached to the segment, with particle filter for selecting only “gamma”.

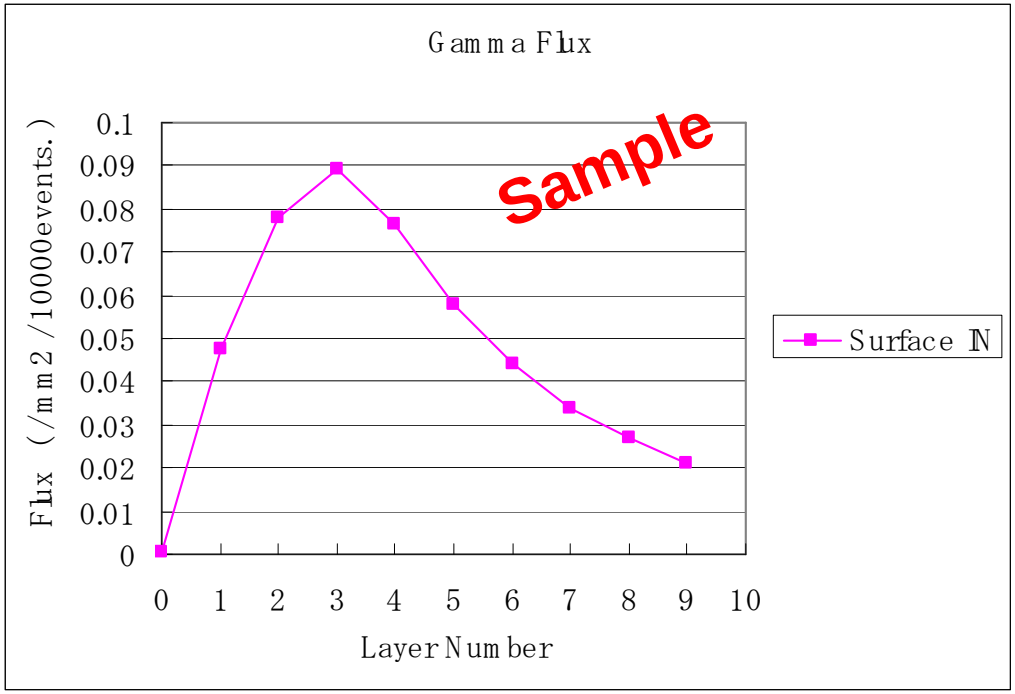
100 MeV electrons. 10000events.

Example -2-

Output of collection.

```
PrimitiveSenstivity RUN PhantomSD,FluxIn
 Number of entries 10
  copy no.: 0  Run Value : 0.00047317284
  copy no.: 1  Run Value : 0.047674118
  copy no.: 2  Run Value : 0.077936007
  copy no.: 3  Run Value : 0.089121806
  copy no.: 4  Run Value : 0.076487671
  copy no.: 5  Run Value : 0.057928149
  copy no.: 6  Run Value : 0.044228167
  copy no.: 7  Run Value : 0.033625537
  copy no.: 8  Run Value : 0.026960407
  copy no.: 9  Run Value : 0.020930302
```

Only for sample implementation.



Summary

- We introduced new functionality “Concrete sensitivity classes” to Geant4.
  - **We were not forced to implement scoring sensitive detectors.**
- This new functionality would be beneficial to realize convenient scoring in space and medical application.
  - Concrete sensitivity classes help convenient scoring.