Geant 4

Detector response – a brief overview

Anton Lechner, CERN

Acknowledgements:
Slides based on work by J. Apostolakis, G. Cosmo, M. Asai, A. Howard

http://cern.ch/geant4

Extracting information

- Once the mandatory classes (Geometry, Physics List, Primary Generator) are implemented, the Geant4 application does not yet include the functionality to extract and store information
  - A user must provide his/her own code to extract information relevant to the simulation application
- For retrieving application-specific information, Geant4 provides the following scoring functionality to users:
  - **Sensitive detector** (optional with **readout geometry**),
  - **Hits** and
  - **Hits collections**

Sensitive detector (SD)

- A logical volume becomes sensitive if it has a pointer to a sensitive detector
  - 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
  - Use Geant4 built-in tools: Primitive scorers

Adding sensitivity to a logical volume:

- Create an instance of a sensitive detector
- Register the sensitive detector to the SD manager
- Assign the pointer of your SD to the logical volume of your detector geometry

```
G4VSolid* boxSolid =  new G4Box( “aBoxSolid“, 1.0 * cm, 1.0 * cm, 1.0 * cm);

G4LogicalVolume* boxLog =
    new  G4LogicalVolume( boxSolid, matSilicon, “aBoxLog“, 0, 0, 0);

G4VSensitiveDetector* sensitiveBox =
                                    new MySensitiveDetector(“/MyDetector”);
G4SDManager* SDManager = G4SDManager::GetSDMPointer();
 SDManager -> AddNewDetector(sensitiveBox);
 boxLog -> SetSensitiveDetector(sensitiveBox);
```

User-defined SD (1/3)

- A powerful way of extracting information from the physics simulation is to define your own SD
- The ingredients of the scoring set-up are:

	Concrete Class	Base Class
Sensitive Detector	*MySensitiveDetector*	G4VSensitiveDetector
Readout Geometry	*MyReadoutGeometry* (optional)	G4VReadoutGeometry
Hit	*MyHit*	G4VHit
		Template Class
Hits Collection		G4THitsCollection<your hit class>

- Derive your own concrete classes from the base classes and customize them according to your needs

User-defined SD (2/3)

- Basic strategy to retrieve information:
  - Assume, you have already created the detector geometry
    - Shape and size (Solid) of your detector, Material
    - Logical volumes
    - Physical volumes
  - Implement a sensitive detector and assign an instance of it to the *logical volume* of your detector geometry set-up
    - Then this volume becomes sensitive
    - The sensitive detector will become “active” for each particle step, if the step starts inside this logical volume
  - Optionally: Implement a readout geometry and attach it to the sensitive detector

User-defined SD (3/3)

- Basic strategy to retrieve information (cont.)
  - Then, create hit objects in your sensitive detector using information from particle steps
    - Hit is a snapshot of the physical interaction of a track or an accumulation of interactions of tracks in the sensitive or interesting region of your detector: *You need to create hit class(es) according to your needs*
    - Use Touchable of the Readout geometry to retrieve geometrical information associated with hits
  - Store your hits in hit collections (hit collections are automatically associated to the G4Event object)
  - Finally, process the information associated with hits in user action classes (G4UserEventAction,G4UserRunAction) to obtain a summary of the event/run

Primitive Scorers (1/2)

- Alternatively, you can use a pre-defined sensitive detector (G4MultiFunctionalDetector) and primitive scorers:

	Concrete Class(es)
Sensitive Detector Primitive Scorers	G4MultiFunctionalDetector G4PSEnergyDeposit, G4PSTracklength, ...
	Template Class
Hits Collection	G4THitsMap<G4double>

- Each primitive scorer stores one physics quantity for each *physical volume* (accumulated for an event)
  - Many scorers are provided by Geant4 (energy deposit, flux, ...)

Primitive Scorers (2/2)

- Basic strategy to retrieve information:
  - Assume, you have already created the detector geometry
    - Shape and size (Solid) of your detector, Material
    - Logical volumes
    - Physical volumes
  - Assign an instance of the Geant4 multi-functional detector (G4MultiFunctionalDetector) to the *logical volume* of your detector geometry set-up
  - Register instances of the required primitive scorers to your the multi-functional detector
  - Finally, process the content of hit maps and store the information