Geant 4

Detector Response

Acknowledgements:

A. Lechner,

J. Apostolakis, M. Asai, G. Cosmo, A. Howard

Overview

- **Concepts**
  - Readout geometry
  - Sensitive detector
  - Hits
  - Digis
- **Details**
  - Hit class
  - Sensitive detector class
  - Hits Collection class and its use
- **Basic features**
  - Digitizer module and digit
  - Touchable

Basic concepts

Extracting information from the simulation

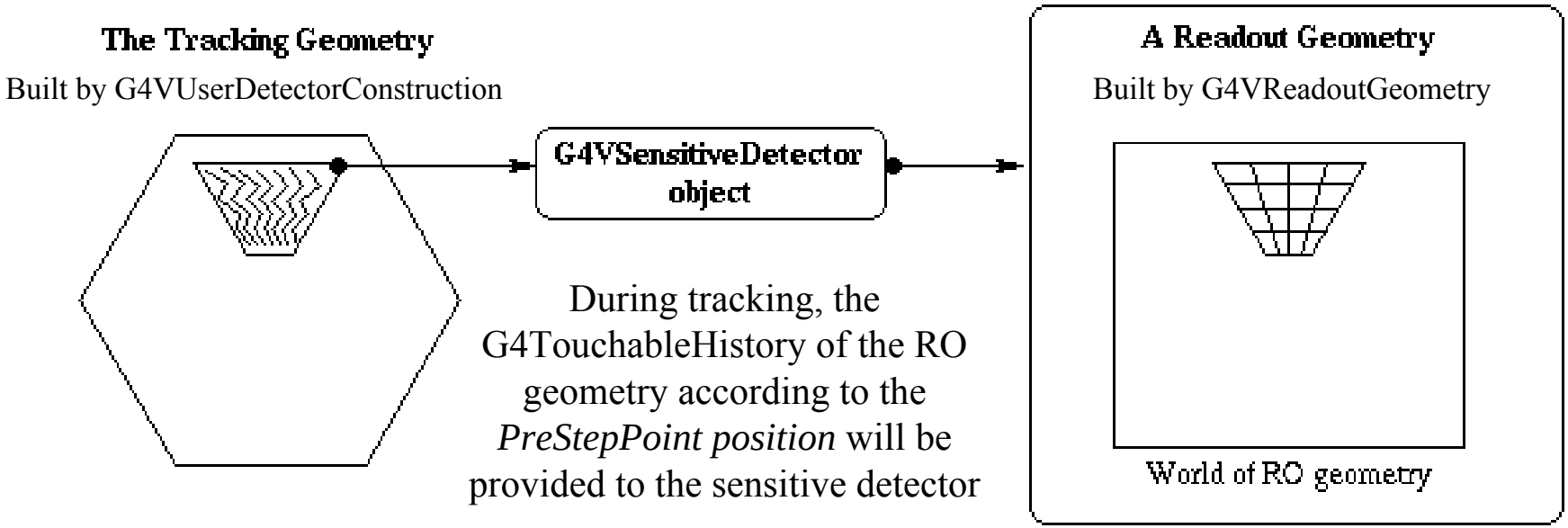
- Once the mandatory classes *(DetectorConstruction, PhysicsList, PrimaryGeneration)* are implemented, the Geant4 application does not yet include functionality to extract information produced in the simulation

- A user **must provide his/her own code**
  - **To extract information** relevant to the simulation application
  - **To describe the detector response**

- Geant4 concepts for such functionality are
  - **Sensitive Detector** (optionally with **Readout Geometry**)
  - **Hits** and **Hits Collections**
  - **Digis** and **Digis Collections**

Readout geometry

- Readout geometry is a **virtual** and **artificial** geometry which can be defined in parallel to the real detector geometry
- Tracks will be tracked in the “real” geometry, but the sensitive detector can have its own geometry for readout purposes
  - e.g. to find the cell the current hit belongs to
- The readout geometry is **optional**; on may have more than one
  - Each one should be associated to a sensitive detector
  - But a sensitive detector is not required to have a readout geometry
- **A step is not limited by** the boundary of **the readout geometry**



How to create a ReadOut Geometry

- Derive your own concrete class from the **G4VReadoutGeometry** abstract base class
- The geometry setup is done in the **same way** as for the “real” tracking geometry:
  - Create solids, logical and physical volumes
- However, the materials used in the RO geometry are **dummy materials**
  - i.e. they are not used
- A **sensitive detector** for the RO geometry must be defined but is not used
  - This means that you need to declare the sensitive parts of the RO geometry by setting a non-NULL sensitive detector pointer to the logical volume

Hit

- Hit is a **user-defined** class, which derives from the G4VHit base class

- You can store various kind of information by implementing your own concrete Hit class
- Typically one records quantities like:
  - Position and time of the step
  - Momentum and energy of the track
  - Energy deposit in the step
  - Geometrical information
  - etc.

Example of a Hit class

```
// header file: MyHit.hh

#include "G4VHit.hh"

class MyHit : public G4VHit {
public:
MyHit();
virtual ~MyHit();
...
inline void SetEnergyDeposit(double energy) { energyDeposit = energy; }
inline double GetEnergyDeposit() { return energyDeposit;}
... // more member functions

private:
G4double energyDeposit;
... // more data members
};
```

Sensitive Detector

- A logical volume becomes **sensitive** if it has a pointer to a **SensitiveDetector (SD)**
- A SensitiveDetector **can be instantiated several times**, where the instances are assigned to different logical volumes
  - 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 multiple functionality
- Two possibilities to make use of the SD functionality
  - **Create your own sensitive detector**
    - Highly customizable
  - **Use Geant4 built-in tools**
    - Primitive Scorers

Creating your own sensitive detector

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

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

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

Basic strategy to retrieve information - 1

- Assume that 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 setup
  - 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**

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.* cm, 1.* cm, 1.* cm);
G4LogicalVolume* boxLog =
        new G4LogicalVolume( boxSolid, materialSilicon, “aBoxLog“, 0, 0, 0);
G4VSensitiveDetector* sensitiveBox = new MySensitiveDetector(“MyDetector”);
G4SDManager* SDManager = G4SDManager::GetSDMPointer();
SDManager ->AddNewDetector(sensitiveBox);
boxLog ->SetSensitiveDetector(sensitiveBox);
```

Basic strategy to retrieve information - 2

- 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 should 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 etc.) to obtain a summary of the relevant event/run features

Using built-in scorers

- Alternatively, you can use a predefined sensitive detector **G4MultiFunctionalDetector** and primitive scorers:

	Concrete class
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 over an event)
- Many scorers are provided by Geant4
  - energy deposit, flux, ...

Basic strategy to retrieve information

- Assume that 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 **multifunctional detector** (G4MultiFunctionalDetector) to the ***logical volume*** of your detector geometry set-up
- **Register** instances of the required **primitive scorers** to your **multifunctional detector**
- Finally, **process** the content of **hit maps**