GEANT4 Introductory Course

Instituto de Estudos Avançados – Instituto Tecnológico de Aeronáutica,
São José Dos Campos, Brazil
July 28th – August 1st, 2014

Scoring

Miguel A. Cortés-Giraldo

based on the original lecture by Sebastien Incerti (IN2P3)

Outline

- **What is scoring?**

- Three types of scoring:
  - User hooks
  - Sensitive Detectors
  - Command-based scoring

Scoring: Extraction of Useful Information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation "silently" by default.
  - **You have to add a bit of code to extract information useful to you**

- There are several ways:
  - Use **user hooks** (G4UserTrackingAction, G4UserSteppingAction, etc.):
    - You have full access to almost all information
    - Straight-forward, but do-it-yourself
  - Use **sensitive detectors:**
    - Assign G4VSensitiveDetector to a volume and optionally generate **"hits"**
    - Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary
  - **Built-in scoring commands:**
    - Most commonly-used physics quantities are available.
  - (Other less common alternatives)

Outline

- ✓ What is scoring?

- ➢ **Three types of scoring:**
  - ➢ **User hooks**
  - Sensitive Detectors
  - Command-based scoring

User Hooks

- **"Do it yourself" approach**
- In Geant4, you have full access to almost all information:
  - G4UserSteppingAction
  - G4UserTrackingAction
  - G4UserEventAction
  - G4UserRunAction
- This is **well adapted to small applications** & Geant4 examples.
- In **large applications**, where many data from many volumes need to be recorded, **this becomes too heavy:**
  - Crowded UserSteppingAction.
  - Need to subdivide problem, which Sensitive Detectors already do for you.

Principle

- In your **SteppingAction**, check that particle is in volume A and do what you want.

- Usually, your containers and histograms will be attributes of Track, Event or Run;
  - Therefore you will have to instantiate TrackingAction and/or EventAction and/or RunAction.
  - Pass their pointer to SteppingAction.

- This approach is illustrated in:
  - examples/novice N03, N06
  - extended/electromagnetic, optical, and many others ...

Geometrical Information - 1

- A G4Step object consists of **two points:**

```
G4StepPoint* point1 = step->GetPreStepPoint();
G4StepPoint* point2 = step->GetPostStepPoint();
```

- To get **their positions in the global coordinate system:**

```
G4ThreeVector pos1 = point1->GetPosition();
G4ThreeVector pos2 = point2->GetPosition();
```

- Hereafter we call **"current volume"** the volume where the step has just gone through.

- Geometrical information is available from **preStepPoint !**

Geometrical Information - 2

- **G4VTouchable** and its derivates keep these geometrical information:

```
G4TouchableHandle touch1 = point1->GetTouchableHandle();
```

- To get the **current physical volume:**

```
G4VPhysicalVolume* volume = touch1->GetVolume();
```

- To get its **name:**

```
G4String name = volume->GetName();
```

- To get **copy number:**

```
G4int copyNumber = touch1->GetCopyNumber();
```

- To get **logical volume:**

```
G4LogicalVolume* lVolume = volume->GetLogicalVolume();
```

Geometrical Information - 3

- To get **material** the following statements are equivalent:

```
G4Material* material = point1->GetMaterial();
G4Material* material = lVolume->GetMaterial();
```

- To get **region:**

```
G4Region* region = lVolume->GetRegion();
```

- To get **mother volume:**

```
G4VPhysicalVolume* mother = touch1->GetVolume(depth=1);
```

  - grandMother: depth=2 ...etc...

- To get **copy number of mother:**

```
G4int copyNumber = touch1->GetCopyNumber(depth=1);
```

  - grandMother: depth=2 ...etc...

Geometrical Information - 4

- To check that particle has just entered in the current volume, i.e. is at the first step in the volume; the **preStepPoint** is at boundary:

```
if (point1->GetStepStatus() == fGeomBoundary)
```

- To check that particle is leaving the current volume, i.e. is at the last step in the volume; the **postStepPoint** is at boundary:

```
if (point2->GetStepStatus() == fGeomBoundary)
```

- In the above situation, get touchable of the next volume:

```
G4TouchableHandle touch2 = point2->GetTouchableHandle();
```

- From **`touch2`**, all information on the next volume as above.

Get Information About Physics - 1

- Physics quantities are available from **step** (G4Step) or **track** (G4Track).

- To get the **process which has limited the current step:**

```
G4VProcess* aProcess = point2->GetProcessDefinedStep();
```

- **Current particle name:**

```
step->GetTrack()->GetDynamicParticle()->GetDefinition()
     ->GetParticleName()
```

- To get **energy deposition**, **step length**, **displacement** and **time of flight** spent by this step:

```
G4double eDeposit      = step->GetTotalEnergyDeposit();
G4double sLength       = step->GetStepLength();
G4ThreeVector displace = step->GetDeltaPosition();
G4double tof           = step->GetDeltaTime();
```

Get Information About Physics - 2

- To get **momentum**, **kinetic energy** and **global time** (time since the beginning of the event) of the track **after the completion of the current step:**

```
G4Track* track          = step->GetTrack();
G4ThreeVector momentum = track->GetMomentum();
G4double kinEnergy      = track->GetKineticEnergy();
G4double globalTime     = track->GetGlobalTime();
```

- **Additional remark:** To transform position from the global coordinate system to the local system of current volume, use **preStepPoint** transformation:

```
G4ThreeVector localPosi = touch1->GetHistory()
     ->GetTopTransform().TransformPoint(position);
```

And more…

- Similarly in TrackingAction one can access track information:

```
void MyTrackingAction::PostUserTrackingAction(const G4Track* track)
{
 G4double tracklen = track->GetTrackLength();
 G4double charge   = track->GetDefinition()->GetPDGCharge();
…
```

- See more in:
  - **$G4INSTALL/include/Geant4/G4Step.hh**
  - **$G4INSTALL/include/Geant4/G4Track.hh**
  - …

- You can retrieve easily quantities at each **step** and cumulate them over events or run using user accessors / recorders added to your EventAction and RunAction classes:

```
G4double dose = aStep->GetTotalEnergyDeposit()/MassTarget;
Run->AddDose(dose);
```

Outline

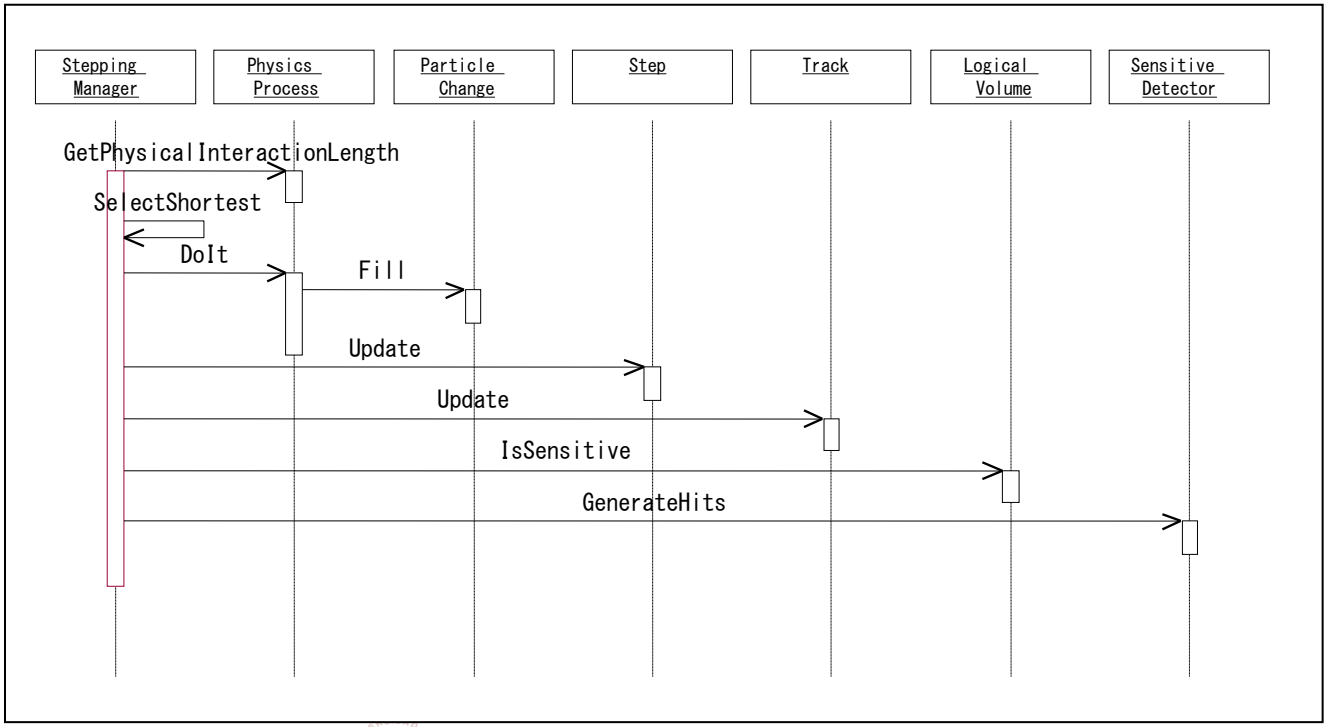
- ✓ What is scoring?
- ➢ **Three types of scoring:**
  - ✓ User hooks
  - ➢ **Sensitive Detectors**
  - ▪ Command-based scoring

What is a Sensitive Detector?

- A sensitive detector can be used to simulate the “read-out” of your detector:
  - It is a way to declare a geometric element **“sensitive” to the passage of particles.**
  - It gives the user a handle to collect quantities from these elements at stepping time
    - For example: energy deposited, position, time information…

Sensitive Detector

- A **G4VSensitiveDetector** object can be assigned to G4LogicalVolume.
- In case a step takes place in a logical volume that has a G4VSensitiveDetector object, this G4VSensitiveDetector is invoked **with the current G4Step object.**
  - You can implement your own sensitive detector classes…
  - or use scorer classes provided by Geant4.



Defining a Sensitive Detector

- Basic strategy in **src/DetectorConstruction.cc**


```
G4LogicalVolume* myLogCalor = ……;
G4VSensitiveDetector* pSensitivePart = new MyDetector("/mydet");
G4SDManager* SDMan = G4SDManager::GetSDMpointer();
SDMan->AddNewDetector(pSensitivePart);
myLogCalor->SetSensitiveDetector(pSensetivePart);
```

- Each detector object must have a unique name:
  - Different logical volumes can share one detector object.
  - More than one SD object can be made from the same SD class with different detector name.
  - One logical volume cannot have more than one detector objects. But, one detector object can generate more than one kinds of hits.
    - e.g. a double-sided silicon micro-strip detector can generate hits for each side separately.

Sensitive Detector Class

- A sensitive detector is a user-defined class that you need to derive from G4VSensitiveDetector

```
#include "G4VSensitiveDetector.hh"

class G4Step;

class MyDetector : public G4VSensitiveDetector
{
  public:
      MyDetector(G4String name);
      virtual ~MyDetector();

      virtual G4bool ProcessHits(G4Step*aStep,
                                 G4TouchableHistory*ROhist);
};
```

include

Base class

SD name in constructor

At each step

How to Collect Information

- At **stepping time**, Geant4 kernel checks for you if particle is in the sentitive detector.
  - If so, it gives you the control to **G4VSensitiveDetector::ProcessHits()**

- Do what you want in **ProcessHits()** using hooks:
  - See previous section on user hooks to collect information you need from step, track…

Outline

- ✓ What is scoring?
- ➢ **Three types of scoring:**
  - ✓ User hooks
  - ✓ Sensitive Detectors
  - ➢ **Command-based scoring**

Command-Based Scoring

- Command-based scoring functionality offers built-in scoring mesh and various scorers for commonly-used physics quantities such as dose, flux, etc.

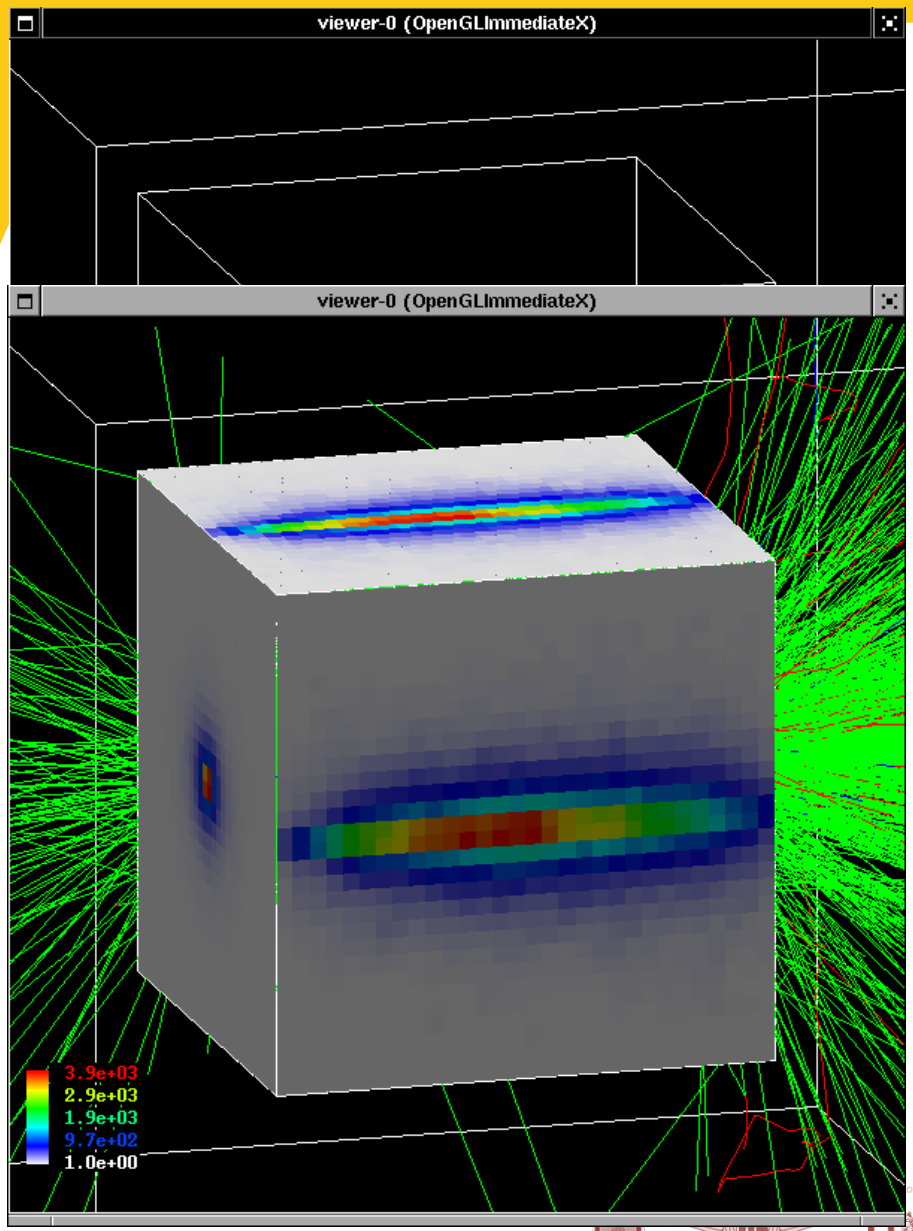
- To use this functionality, access to the G4ScoringManager pointer after the instantiation of G4RunManager in your main()

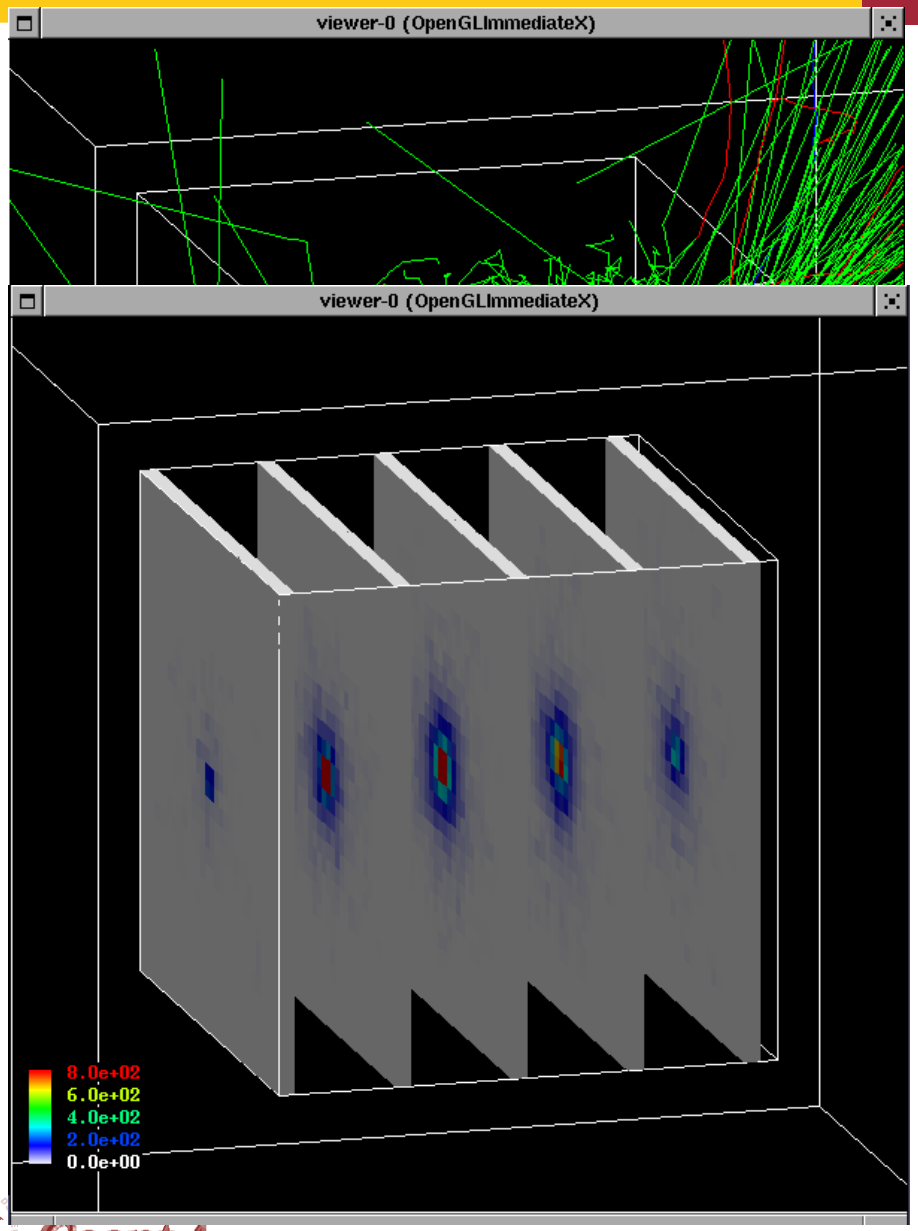
```
#include "G4ScoringManager.hh"
int main()
{
  G4RunManager* runManager = new G4RunManager;
  G4ScoringManager* scoringManager =
      G4ScoringManager::GetScoringManager();
  …
```

- All of the UI commands of this functionality is in **/score/** directory.

- **/examples/extended/runAndEvent/RE03**

/example/extended/runAndEvent/RE03





Define a Scoring Mesh

- To define a scoring mesh, the user has to specify the following:
  - **Shape** and **name** of the 3D scoring mesh.
    - Box, cylindrical mesh
  - **Size** of the scoring mesh. Mesh size must be specified as "half width" similar to the arguments of G4Box.
  - **Number of bins for each axes.** Note that too many bins causes immense memory consumption.
  - Optionally, position and rotation of the mesh. If not specified, the mesh is positioned at the center of the world volume without rotation.

```
# define scoring mesh
/score/create/boxMesh boxMesh_1
/score/mesh/boxSize 100. 100. 100. cm
/score/mesh/nBin 30 30 30
```

- The mesh geometry can be **completely independent** from the real material geometry.

Scoring Quantities

- A **mesh** may have **arbitrary number of scorers.** Each scorer scores one physics quantity (xxxxx).
  - **energyDeposit** * Energy deposit scorer.
  - **cellCharge** * Cell charge scorer.
  - **cellFlux** * Cell flux scorer.
  - **passageCellFlux** * Passage cell flux scorer
  - **doseDeposit** * Dose deposit scorer.
  - **nOfStep** * Number of step scorer.
  - **nOfSecondary** * Number of secondary scorer.
  - **trackLength** * Track length scorer.
  - **passageCellCurrent** * Passage cell current scorer.
  - **passageTrackLength** * Passage track length scorer.
  - **flatSurfaceCurrent** * Flat surface current Scorer.
  - **flatSurfaceFlux** * Flat surface flux scorer.
  - **nOfCollision** * Number of collision scorer.
  - **population** * Population scorer.
  - **nOfTrack** * Number of track scorer.
  - **nOfTerminatedTrack** * Number of terminated tracks scorer.

```
/score/quantity/xxxxx <scorer_name>
```

Filter

- Each scorer may take a **filter:**
  - **charged** * Charged particle filter.
  - **neutral** * Neutral particle filter.
  - **kineticEnergy** * Kinetic energy filter.

    `/score/filter/kineticEnergy <fname> <eLow> <eHigh> <unit>`
  - **particle** * Particle filter.

    `/score/filter/particle <fname> <p1> … <pn>`
  - **particleWithKineticEnergy** * Particle with kinetic energy filter.

```
/score/quantity/energyDeposit   eDep
/score/quantity/nOfStep   nOfStepGamma
/score/filter/particle   gammaFilter   gamma
/score/quantity/nOfStep   nOfStepEMinus
/score/filter/particle   eMinusFilter   e-
/score/quantity/nOfStep   nOfStepEPlus
/score/filter/particle   ePlusFilter   e+
/score/close
```

Same primitive scorers with different filters may be defined.

Close the mesh when defining scorers is done.

Drawing a Score

- **Projection:**

```
/score/drawProjection <mesh_name> <scorer_name> <color_map>
```

- **Slice:**

```
/score/drawColumn <mesh_name> <scorer_name> <plane> <column>
   <color_map>
```

- **Color map:**
  - By default, linear and log-scale color maps are available.
  - Minimum and maximum values can be defined by `/score/colorMap/setMinMax` command. Otherwise, min and max values are taken from the current score.

Write Scores to a File

- **Single score:**

```
/score/dumpQuantityToFile <mesh_name> <scorer_name> <file_name>
```

- **All scores:**

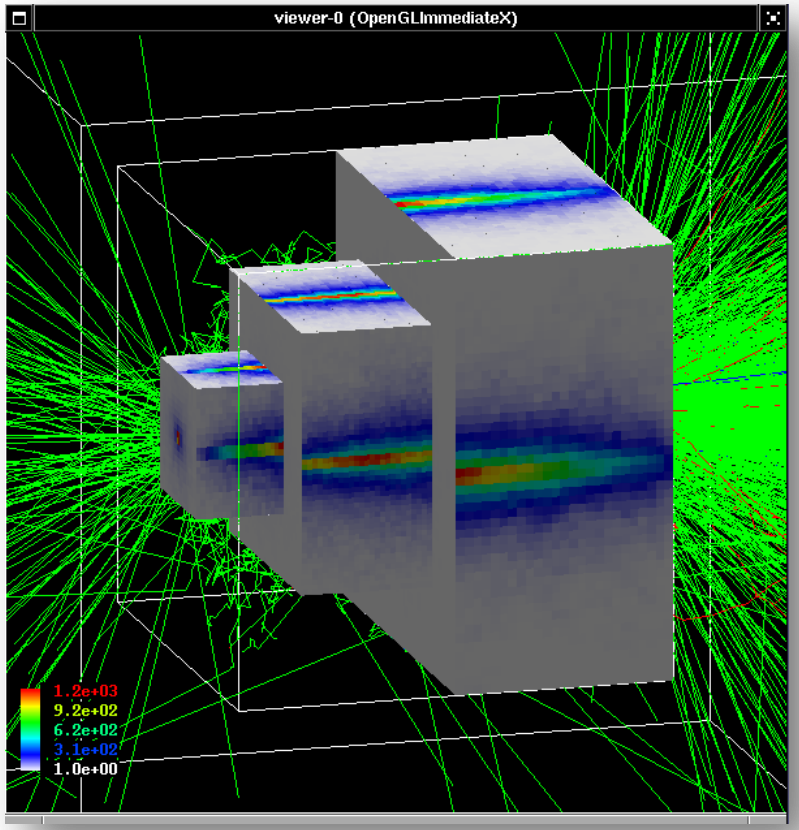
```
/score/dumpAllQuantitiesToFile <mesh_name> <file_name>
```

- By default, values are written in CSV.

- By creating a concrete class derived from G4VScoreWriter base class, the user can define his own file format.
  - Example in **/examples/extended/runAndEvent/RE03**
  - User's score writer class should be registered to G4ScoringManager.

More than One Scoring Mesh

- You may define **more than one** scoring mesh.
  - And, you may define **arbitrary number of primitive scorers** to each scoring mesh.
- Mesh volumes **may overlap** with other meshes and/or with mass geometry.
- A **step** is **limited on any boundary.**
- Please be cautious of too many meshes, too granular meshes and/or too many primitive scorers.
  - Memory consumption
  - Computing speed



Summary

- Geant4 already equipped for scoring.

- Several methods:
  - Use of user hooks at different stages.
    (step, track, event, run,…)
    - Methods for the retrieval of Physics quantities

  - Sensitive detectors & hit collections.

  - Built-in commands for scoring.
    - A rich variety of physics quantities

Thanks for your attention