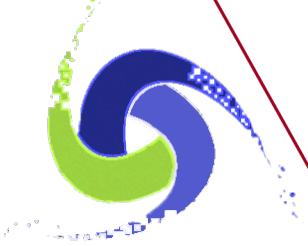


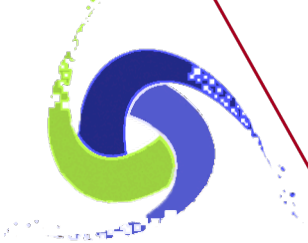
GEANT4
A SIMULATION TOOLKIT
Version 10.5

Scoring I

Makoto Asai (SLAC)
Geant4 Tutorial Course

Contents

- Introduction to sensitivity
- Command-based scoring
- Add a new scorer/filter



GEANT4

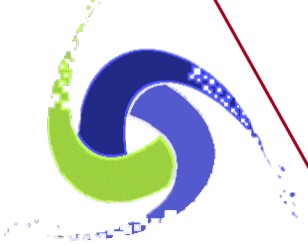
A SIMULATION TOOLKIT

Version 10.5

Retrieving information from Geant4

Extract useful information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”.
  - You have to do something to extract information useful to you.
- There are three ways:
  - Built-in scoring commands
    - Most commonly-used physics quantities are available.
  - Use scorers in the tracking volume
    - Create scores for each event
    - Create own Run class to accumulate scores
  - Assign G4VSensitiveDetector to a volume to generate “hit”.
    - Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary
- You may also use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
  - You have full access to almost all information
  - Straight-forward, but do-it-yourself



GEANT4

A SIMULATION TOOLKIT

Version 10.5

Command-based scoring

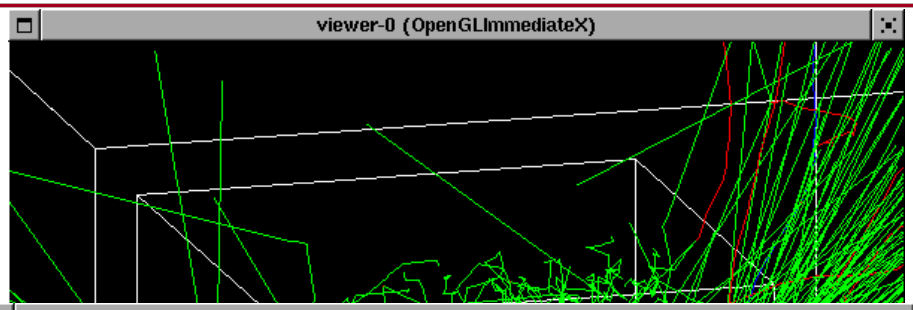
Command-based scoring

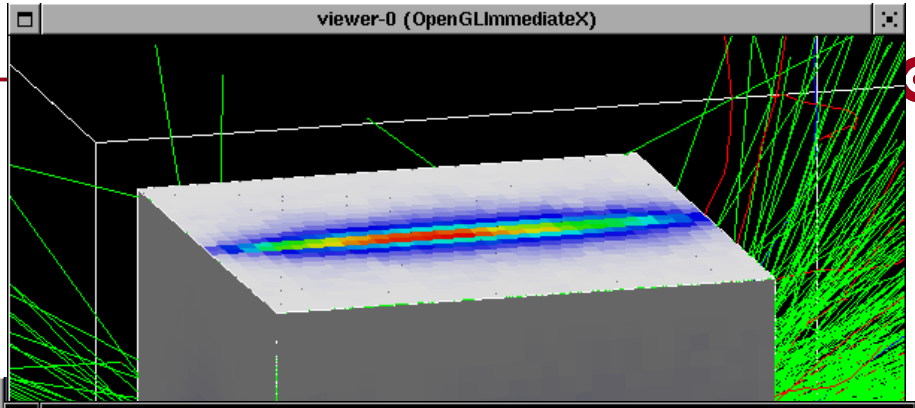
- Command-based scoring functionality offers the built-in scoring mesh and various scorers for commonly-used physics quantities such as dose, flux, etc.
  - Due to small performance overhead, it does not come by default.
- To use this functionality, access to the G4ScoringManager pointer after the instantiation of G4(MT)RunManager in your *main*().

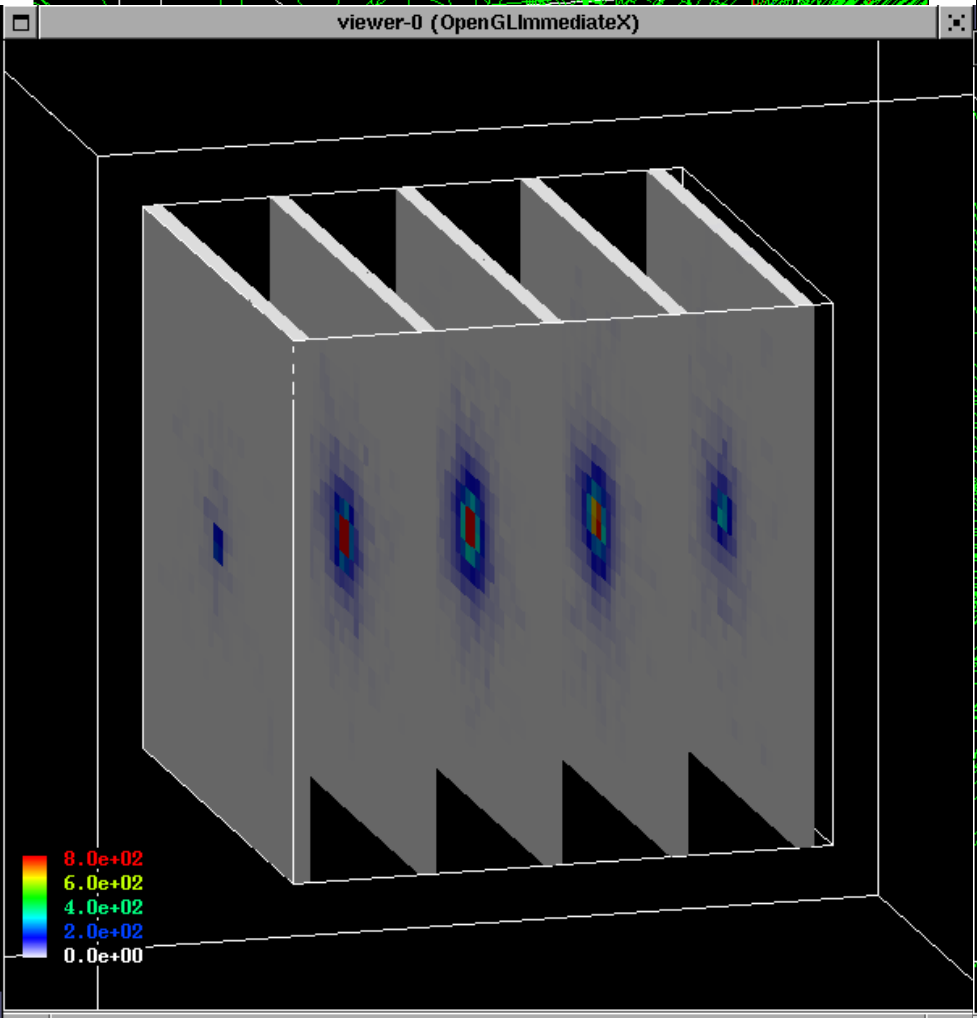
```
#include "G4ScoringManager.hh"
int main()
{
  G4RunManager* runManager = new G4MTRunManager;
  G4ScoringManager* scoringManager =
                                  G4ScoringManager::GetScoringManager();
   …
```

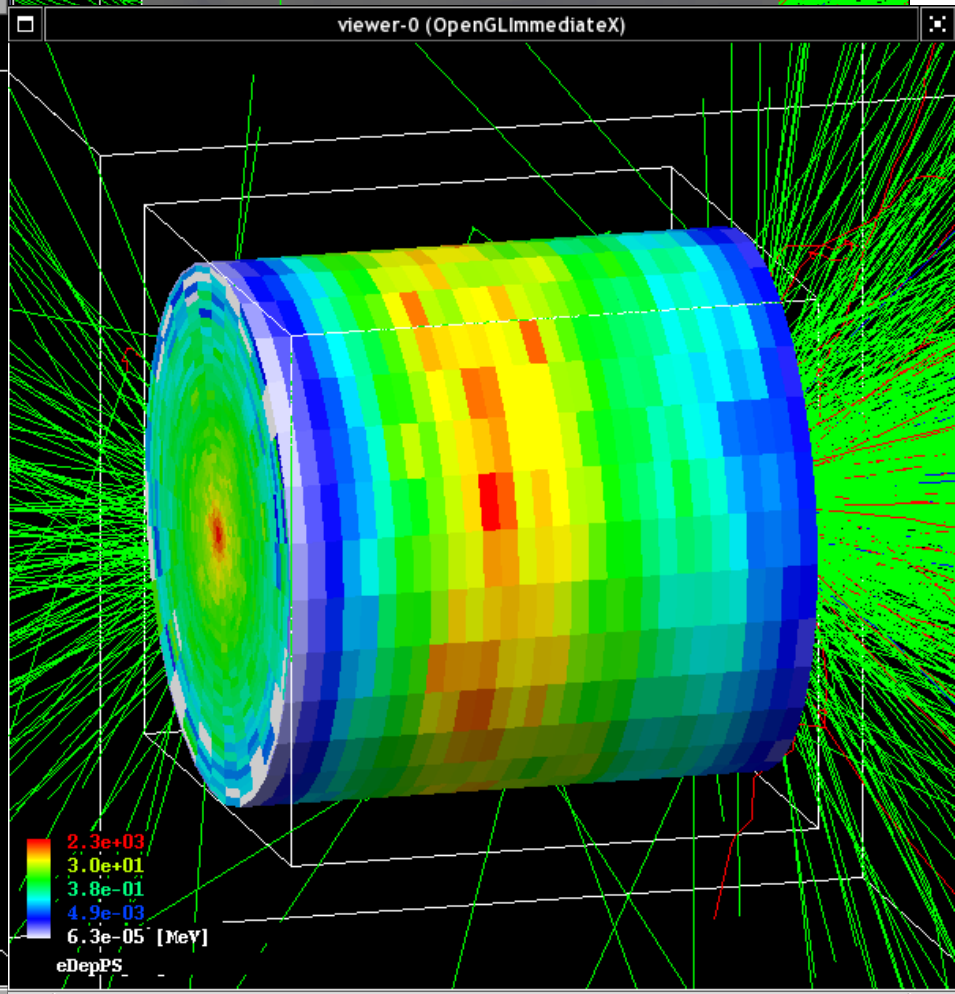
- All of the UI commands of this functionality are in /score/ directory.
- /examples/extended/runAndEvent/RE03

Command-based scorers









Define a scoring mesh

- To define a scoring mesh, the user has to specify the followings.
  1. Shape and name of the 3D scoring mesh.
     - Currently, box and cylinder are available.
  2. Size of the scoring mesh.
     - Mesh size must be specified as "half width" similar to the arguments of G4Box / G4Tubs.
  3. Number of bins for each axes.
     - Note that too many bins causes immense memory consumption.
  4. Specify 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
/score/mesh/translate/xyz 0. 0. 100. cm
```

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

Scoring quantities

- A mesh may have arbitrary number of scorers. Each scorer scores one physics quantity.
  - 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.

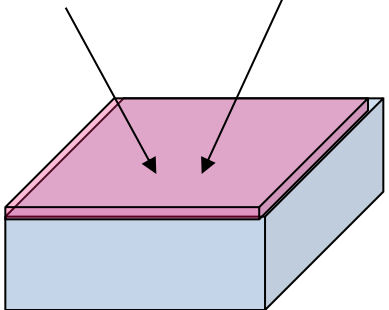
/score/quantity/xxxxx <scorer_name> <unit>

List of provided primitive scorers

- Concrete Primitive Scorers (See Application Developers Guide 4.4.6)
  - Track length
    - G4PSTrackLength, G4PSPassageTrackLength
  - Deposited energy
    - G4PSEnergyDepsit, G4PSDoseDeposit, G4PSChargeDeposit
  - Current/Flux
    - G4PSFlatSurfaceCurrent, G4PSSphereSurfaceCurrent,G4PSPassageCurrent, G4PSFlatSurfaceFlux, G4PSCellFlux, G4PSPassageCellFlux
  - Others
    - G4PSMinKinEAtGeneration, G4PSNofSecondary, G4PSNofStep

SurfaceCurrent :

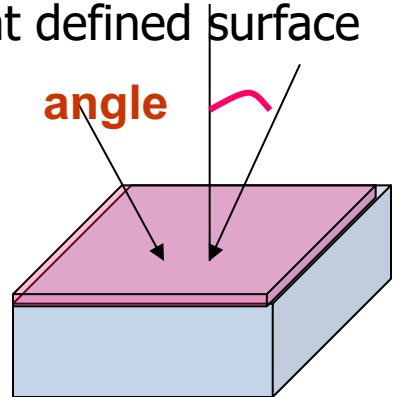
Count number of injecting particles at defined surface.



SurfaceFlux :

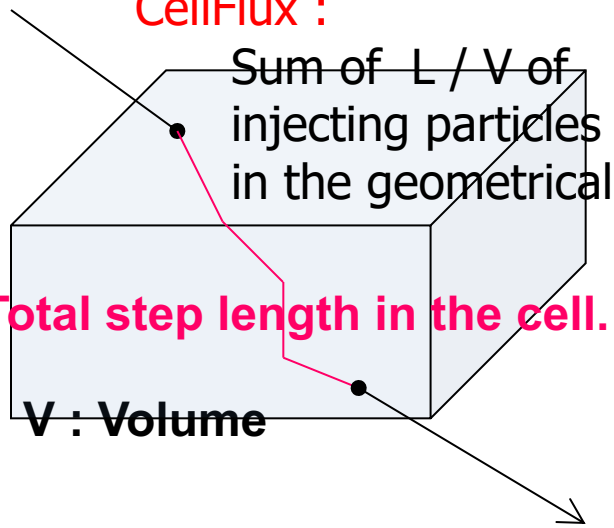
Sum up 1/cos(angle) of injecting particles at defined surface

angle



CellFlux :

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



L : Total step length in the cell.

V : Volume

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/filter/ParticleWithKineticEnergy
    <fname> <eLow> <eHigh> <unit> <p1> … <pn>

/score/quantity/energyDeposit eDep MeV
/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+

Same primitive scorers with different filters may be defined.

/score/close **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.

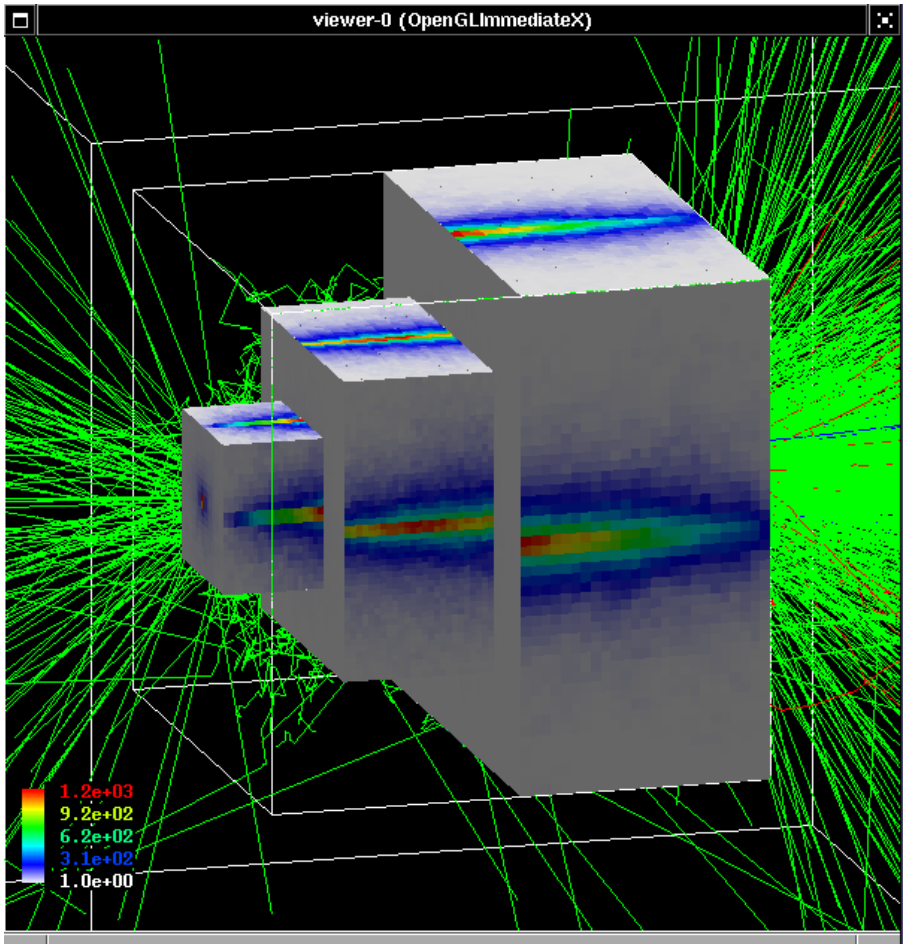
Energy spectrum?

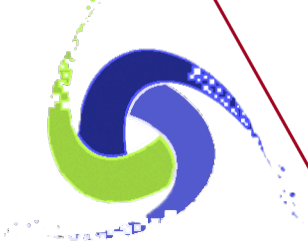
- One of most frequently asked questions is “How to get energy spectrum?”.
- Create arbitrary number of flux scorers of same kind combined with particle and kinetic energy filters.

```
/score/quantity/flatSurfaceFlux  flux0
/score/filter/particleWithKineticEnergy range0 10. 20. MeV e-
/score/quantity/flatSurfaceFlux  flux1
/score/filter/particleWithKineticEnergy range1 20. 30. MeV e-
/score/quantity/flatSurfaceFlux  flux2
/score/filter/particleWithKineticEnergy range2 30. 40. MeV e-
/score/quantity/flatSurfaceFlux  flux3
/score/filter/particleWithKineticEnergy range3 40. 50. MeV e-
```

More than one scoring meshes

- 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





GEANT4

A SIMULATION TOOLKIT

Version 10.5

Add a new scorer/filter to command-based scorers

Scorer base class

- G4VPrimitiveScorer is the abstract base of all scorer classes.
- To make your own scorer you have to implement at least:
  - Constructor
  - Initialize()
    - Initialize G4THitsMap<G4double> map object
  - ProcessHits()
    - Get the physics quantity you want from G4Step, etc. and fill the map
  - Clear()
  - GetIndex()
    - Convert three copy numbers into an index of the map
- G4PSEnergyDeposit3D could be a good example.
- Create your own messenger class to define /score/quantity/<your_quantity> command.
  - Refer to G4ScorerQuantityMessengerQCmd class.

Creating your own scorer

- Though we provide most commonly-used scorers, you may want to create your own.
  - If you believe your requirement is quite common, just let us know, so that we will add a new scorer.
- G4VPrimitiveScorer is the abstract base class.

```
class G4VPrimitiveScorer
{
 public:
   G4VPrimitiveScorer(G4String name, G4int depth=0);
   virtual ~G4VPrimitiveScorer();
 protected:
   virtual G4bool ProcessHits(G4Step*,
                              G4TouchableHistory*) = 0;
   virtual G4int GetIndex(G4Step*);
 public:
   virtual void Initialize(G4HCofThisEvent*);
   virtual void EndOfEvent(G4HCofThisEvent*);
   virtual void clear();
       …
};
```

- Methods written in red will be discussed at “Scoring 2” talk.

Filter class

- G4VSDFilter
  - Abstract base class which you can use to make your own filter

```
class G4VSDFilter
{
  public:
      G4VSDFilter(G4String name);
      virtual ~G4VSDFilter();
  public:
      virtual G4bool Accept(const G4Step*) const = 0;
…
```

- Create your own messenger class to define /score/filter/<your_filter> command.
  - Refer to G4ScorerQuantityMessenger class.