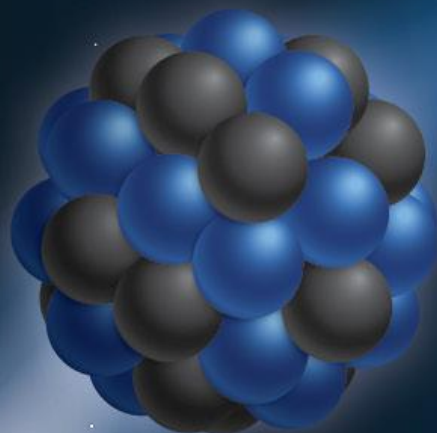


GEANT4

A SIMULATION TOOLKIT

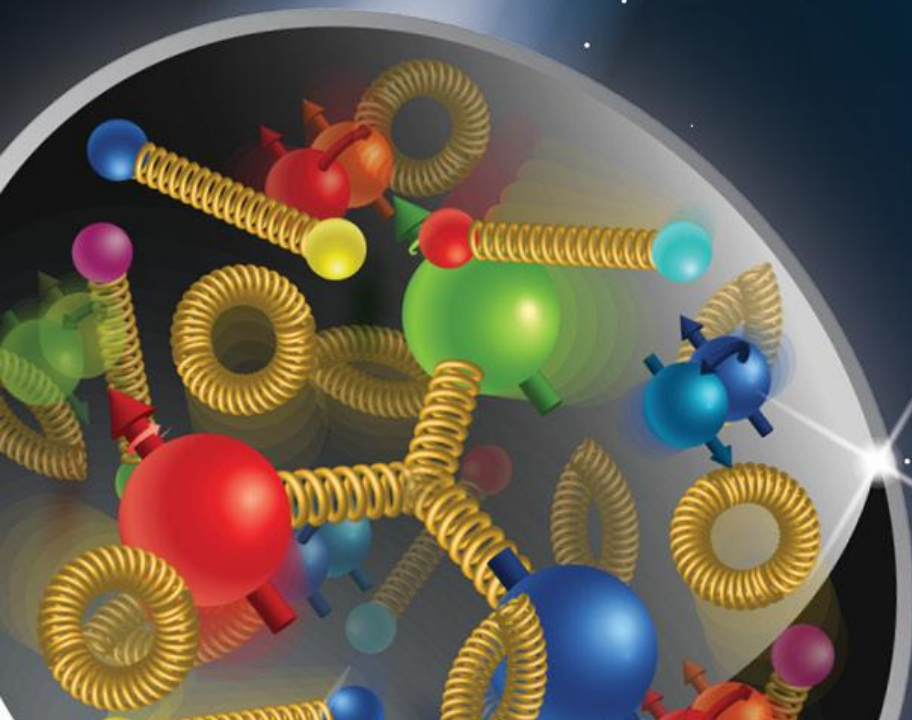
Version 11.3-p02



Scoring I

Makoto Asai (Jefferson Lab)

Geant4 Tutorial Course



 **Jefferson Lab**



U.S. DEPARTMENT OF
ENERGY

Office of
Science



Contents



- Retrieving information from Geant4
- Command-based scoring
 - Mesh scorer
 - Scorer in real world volume
 - Probe scorer



Contents



- Retrieving information from Geant4
- Command-based scoring
 - Mesh scorer
 - Scorer in real world volume
 - Probe scorer



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

This talk

Command-based scoring

- Command-based scoring functionality offers the various built-in 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 and /examples/advanced/gorad are good examples

Three types of command-based scorers

1) Scoring mesh

- Define 3-D mesh (box or cylinder)
- The mesh may overlap with real-world volumes
- Assign arbitrary number of **primitive scorers** to mesh cell

2) Assigning scorers to a real-world logical volume

- Declare a real-world **logical** volume as a detector
- Assign arbitrary number of primitive scorers to the detector
- If the volume is placed more than once, assigned scorers individually score for each physical volume

3) Scoring probe

- A probe is a small virtual cube that is located at arbitrary position. It may overlap with real-world volumes.
- Assign arbitrary number of primitive scorers to the probe
- If probe is placed more than once, assigned scorers individually score for each probe.

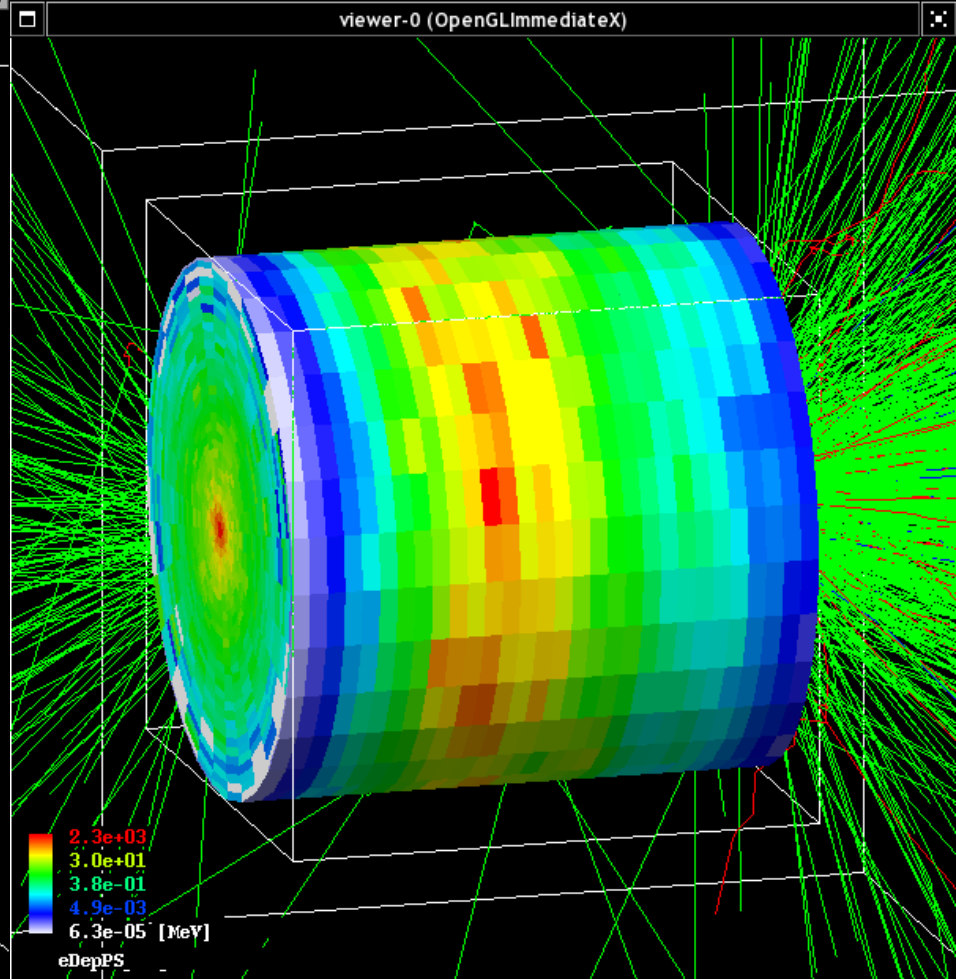
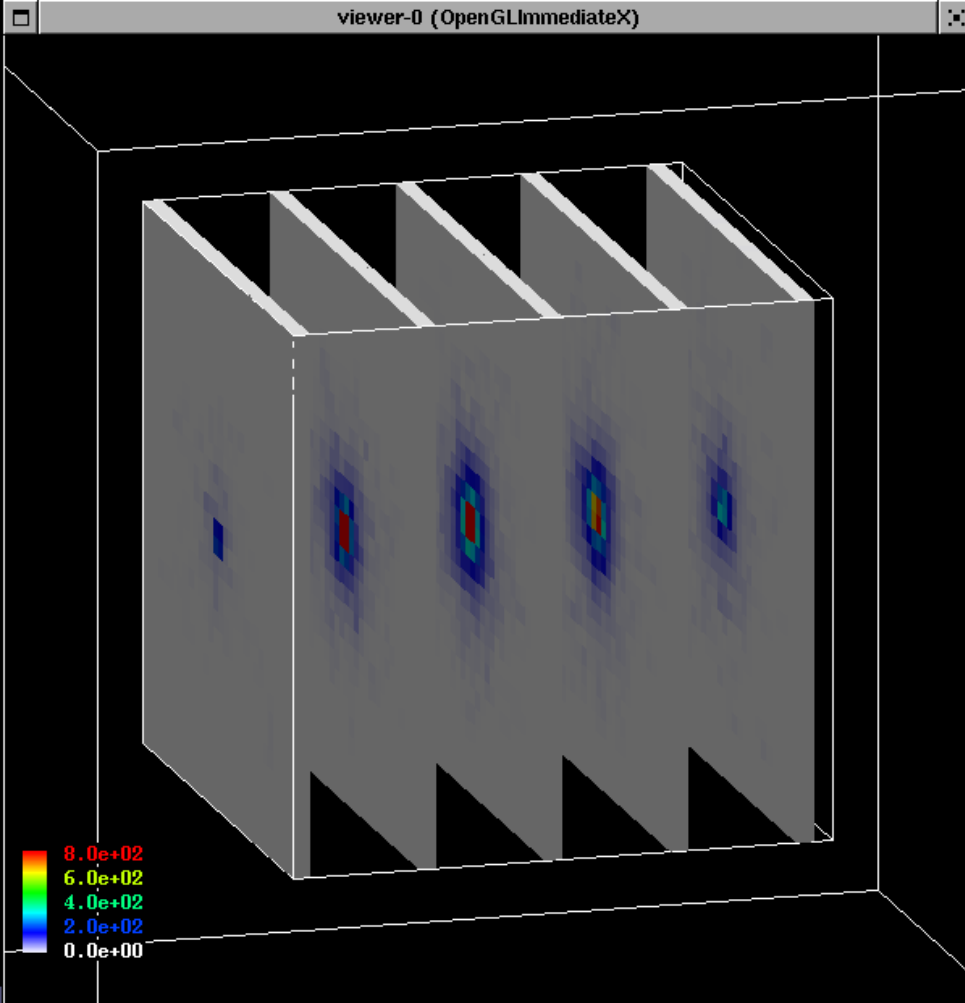
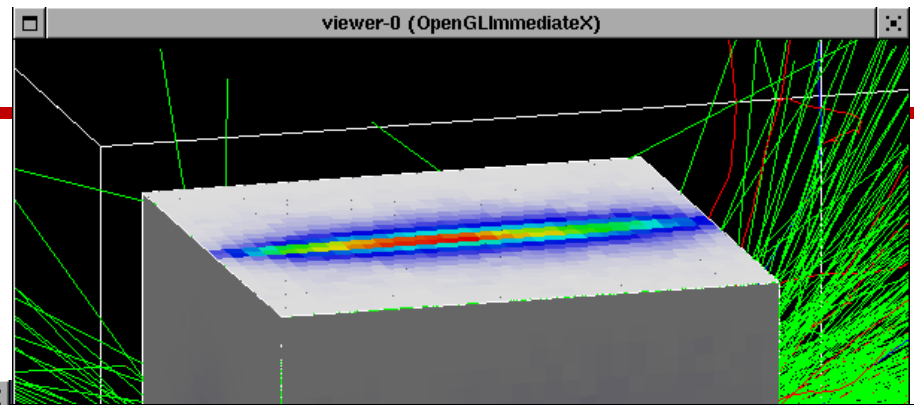
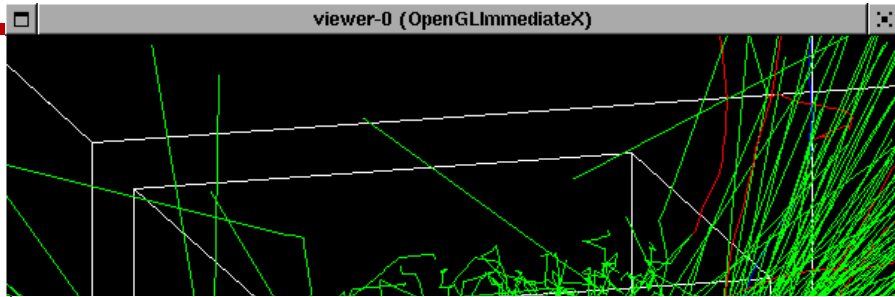
Contents



- Retrieving information from Geant4
- Command-based scoring
 - Mesh scorer
 - Scorer in real world volume
 - Probe scorer




extended/runAndEvent/RE03



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
```

 **half width !**

- The mesh geometry can be completely independent to the real material geometry. It may overlap with any volumes in the real-world volumes.
 - Must be fully contained inside the world volume.

Scoring quantities

- A mesh may have arbitrary number of scorers. Each scorer scores one physics quantity.
 - energyDeposit * Energy deposit scorer.
 - cellCharge * Cell charge scorer.
 - volumeFlux * General-purpose flux 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.

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+

/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>`

- Available only for box or cylindrical mesh.

- Color map

- By default, linear and log-scale color maps are available.

- Minimum and maximum values can be defined by

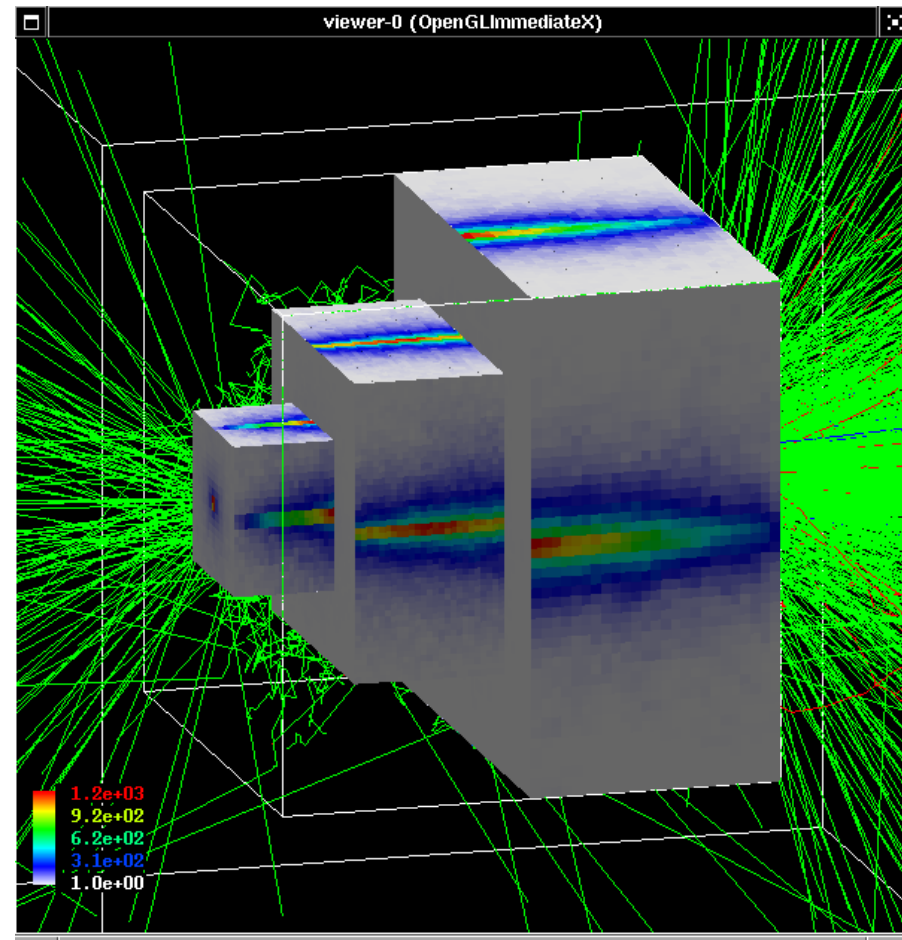
`/score/colorMap/setMinMax` command. Otherwise, min and max values are taken from the current score.

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 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



Contents



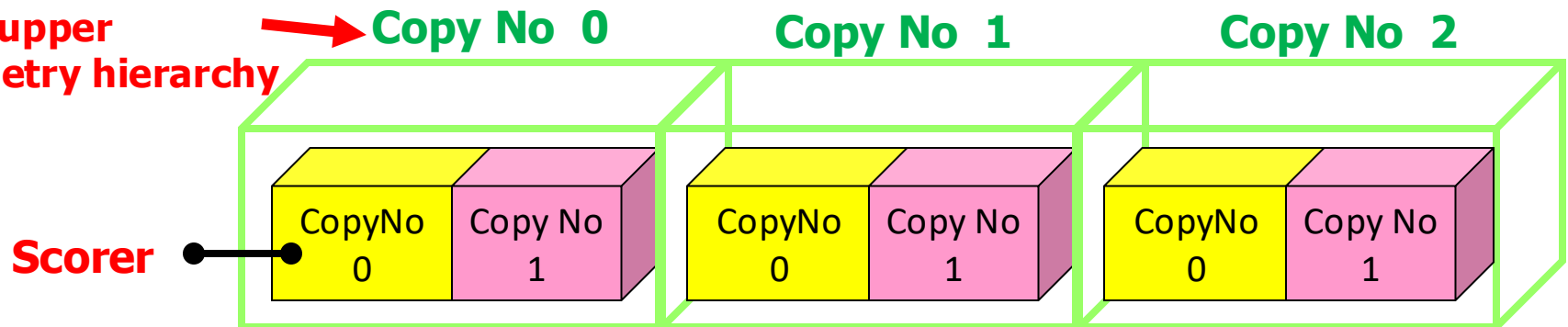
- Retrieving information from Geant4
- Command-based scoring
 - Mesh scorer
 - Scorer in real world volume
 - Probe scorer



Define scorer to a tracking volume

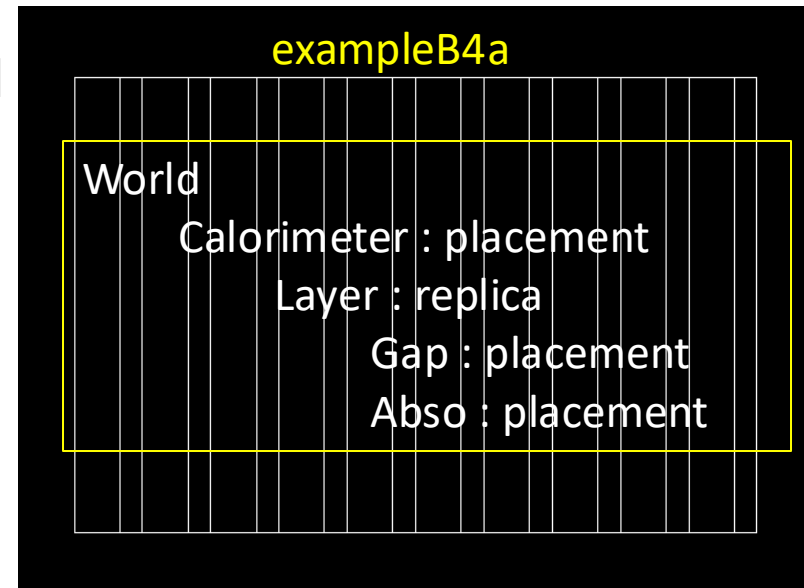
- Define a logical volume as a detector.
`/score/create/realWorldLogVol <LV_name> <anc_lvL>`
- One can define arbitrary scoring quantities and filters.
 - Same recipe as scoring mesh.
 - Scores are automatically merged over worker threads and written to a file.
 - Drawing is not yet supported.
- All physical volumes that share the same `<LV_name>` have the same primitive scorers but score separately.
 - Copy number of the physical volume is the index.
 - If the physical volume is placed only once, but its (grand-)mother volume is replicated, use the `<anc_lvL>` parameter to indicate the ancestor level where the copy number should be taken.

**Index to be taken
from upper
geometry hierarchy**



Command-based real-world scorer

- Do not use this
`/score/create/realWorldLogVol` command to a mother logical volume.
 - For example of this exampleB4, “Layer” is fully filled with “Gap” and “Abso” daughter volumes. You won’t see any energy deposition in “Layer” volume.



```
/score/create/realWorldLogVol Gap 1  
/score/quantity/energyDeposit eDep MeV  
/score/quantity/trackLength sLen mm  
/score/filter/charged cFilter  
/score/create/realWorldLogVol Abso 1  
/score/quantity/energyDeposit eDep MeV  
/score/quantity/trackLength sLen mm  
/score/filter/charged cFilter  
/score/close
```

If this is not set, given “Gap” and “Abso” are placed with copy number 0, energy deposition and track length are accumulated for all layers.

Contents

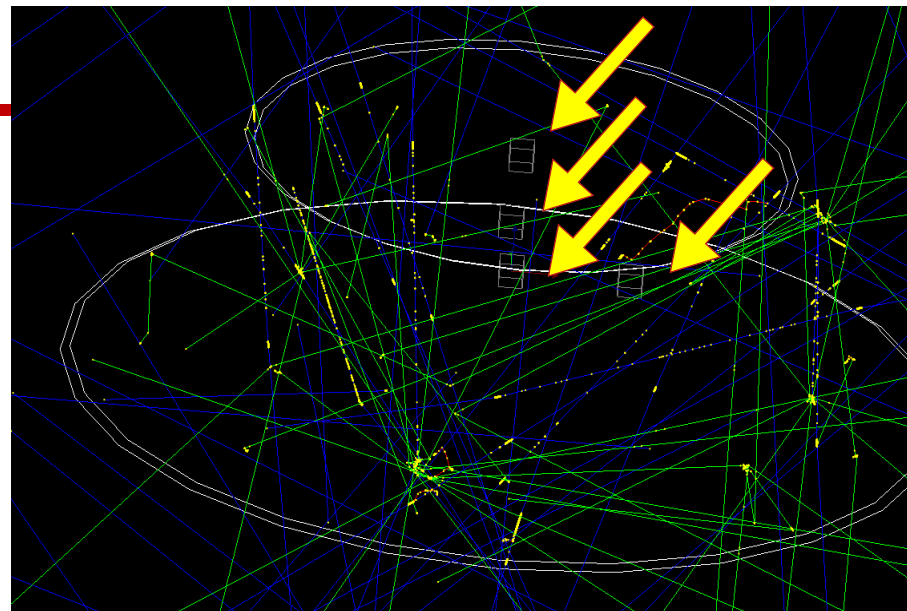


- Retrieving information from Geant4
- Command-based scoring
 - Mesh scorer
 - Scorer in real world volume
 - Probe scorer



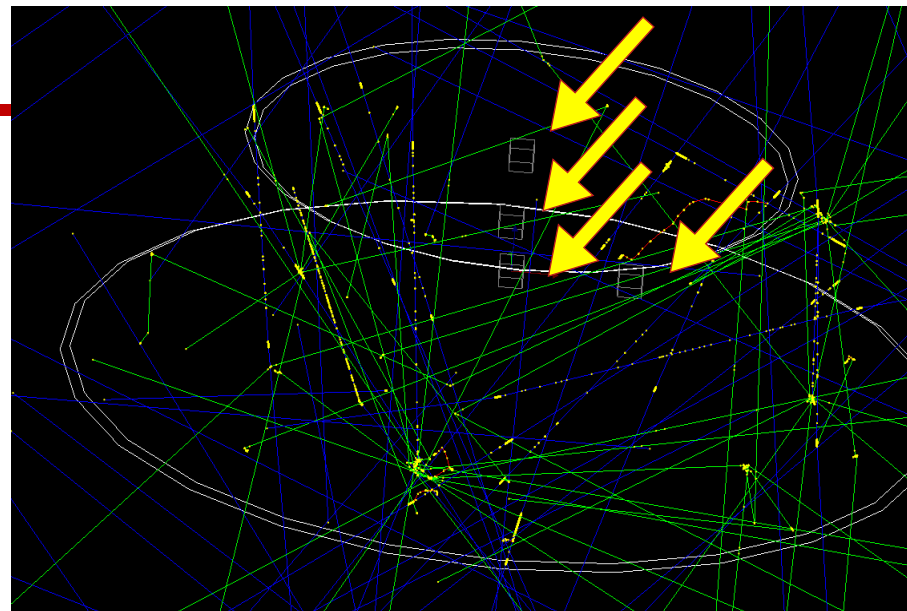
Command-based probe scorer

- User may locate scoring “probes” at arbitrary locations. A “probe” is a virtual cube, to which any primitive scorers could be assigned.
 - Once a probe is defined, user can associate arbitrary number of primitive scorers and filters like the conventional scoring mesh.
- Given these probes are located in an artificial “parallel world”, probes may overlap to the volumes defined in the mass geometry.
 - Probes cannot overlap to each other.
- If probes are located more than once, all probes have the same scorers but score individually.



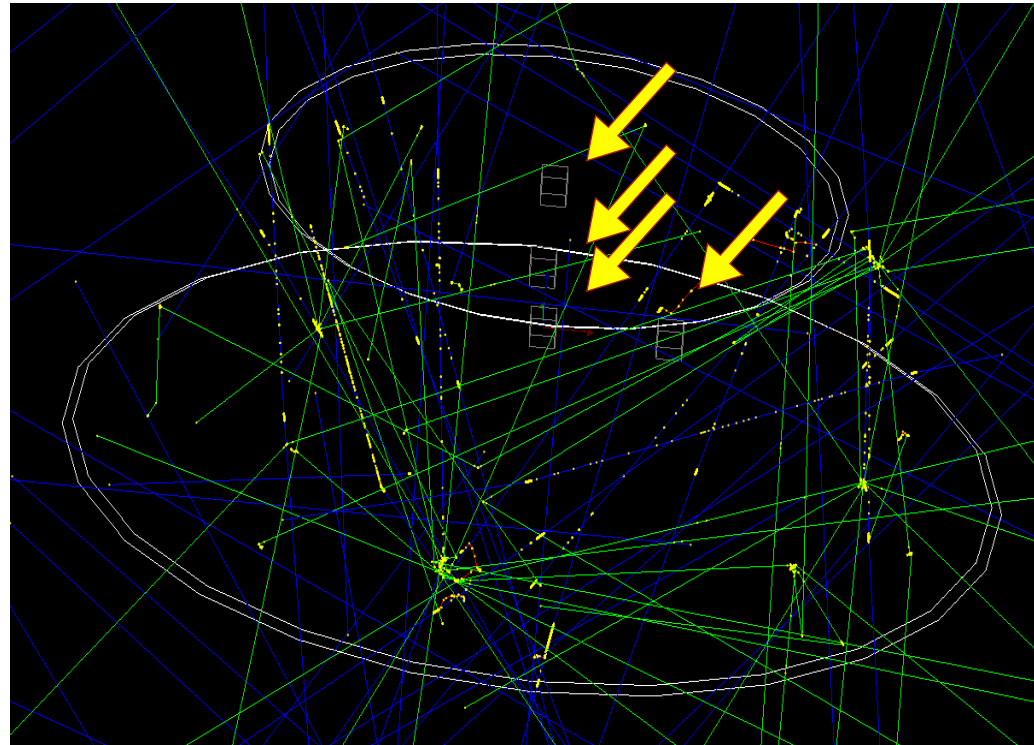
Command-based probe scorer

- In addition, the user may optionally set a material to the probe. Once a material is set to the probe, it overwrites the material(s) defined in the mass geometry when a track enters the probe cube.
 - Because of this overwriting, physics quantities that depend on material or density, e.g. energy deposition or dose, would be measured accordingly to the specified material.
 - This material overwriting affects to the physics results. Make the probe small to avoid large artificial side-effects.



Scoring probe

```
/score/create/probe Probes 5. cm  
/score/probe/material G4_WATER  
/score/probe/locate 0. 0. 0. cm  
/score/probe/locate 25. 0. 0. cm  
/score/probe/locate 0. 25. 0. cm  
/score/probe/locate 0. 0. 25. cm  
/score/quantity/energyDeposit eDep MeV  
/score/quantity/doseDeposit dose mGy  
/score/quantity/volumeFlux volFlx  
/score/quantity/volumeFlux protonFlux  
/score/filter/particle protonFilter proton  
/score/close
```



Note: To visualize the probes defined in a parallel world, the following command is required.

```
/vis/drawVolume worlds
```

1-D histogram directly filled by a primitive scorer

- Through a newly introduced interface class (G4TScoreHistFiller) a primitive scorer can directly fill a 1-D histogram defined by G4Analysis.
 - Track-by-track or step-by-step filling allows command-based histogram such as energy spectrum.
- G4TScoreHistFiller template class must be instantiated in the user's code with his/her choice of analysis data format.

```
#include "G4AnalysisManager.hh"
#include "G4TScoreHistFiller.hh"
auto analysisManager = G4AnalysisManager::Instance();
analysisManager->SetDefaultFileType("root");
auto histFiller = new G4TScoreHistFiller<G4AnalysisManager>;
```

- Primitive scorer must be defined in advance to setting a histogram.
 - Direct filling works only with “volumeFlux” scorer.
- Histogram must be defined through /analysis/h1/create command in advance to setting it to a primitive scorer.
- This functionality is available only for primitive scorers defined in **real-world scorer or probe scorer**.
 - Not available for box or cylindrical mesh scorer due to memory consumption concern.

1-D histogram directly filled by a primitive scorer

```
/score/create/probe Probes 5. cm  
/score/probe/locate 0. 0. 0. cm  
/score/quantity/volumeFlux volFlux  
/score/quantity/volumeFlux protonFlux  
/score/filter/particle protonFilter proton  
/score/close  
/analysis/h1/create volFlux Probes_volFlux  
100 0.01 2000. MeV ! log
```

```
/score/fill1D 1 Probes volFlux  
/analysis/h1/create protonFlux Probes_protonFlux  
100 0.01 2000. MeV ! log
```

```
/score/fill1D 2 Probes protonFlux
```

N.B. If probe is placed more than once, *fill1D* command should be called to each *copyNo*.

```
/score/fill1D 1 Probes volFlux 0
```

