esipap
European School of Instrumentation
in Particle & Astroparticle Physics

esi
European Scientific Institute

Detector Simulation
Scoring

Witek Pokorski
Alberto Ribon
CERN
8-9.02.2021

Geant4 'SCORING'

- Retrieving information from Geant4 using scoring
- Command-based scoring

covered here

- Add a new scorer/filter to command-based scoring
- Define scorers in the tracking volume
- Accumulate scores for a run

not covered here

Extract useful information - reminder

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”.
  - You have to add a bit of code to extract information useful to you.
- There are three ways:
  - Assign G4VSensitiveDetector to a volume to generate “hit”.
    - Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary

    } Covered before
  - Built-in scoring commands
    - Most commonly-used physics quantities are available.

    } Covered here
  - Use scorers in the tracking volume
    - Create scores for each event
    - Create own Run class to accumulate scores

    } Not covered here
- You may also use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
  - You have full access to almost all information
  - Straight-forward, but do-it-yourself

  } Covered before

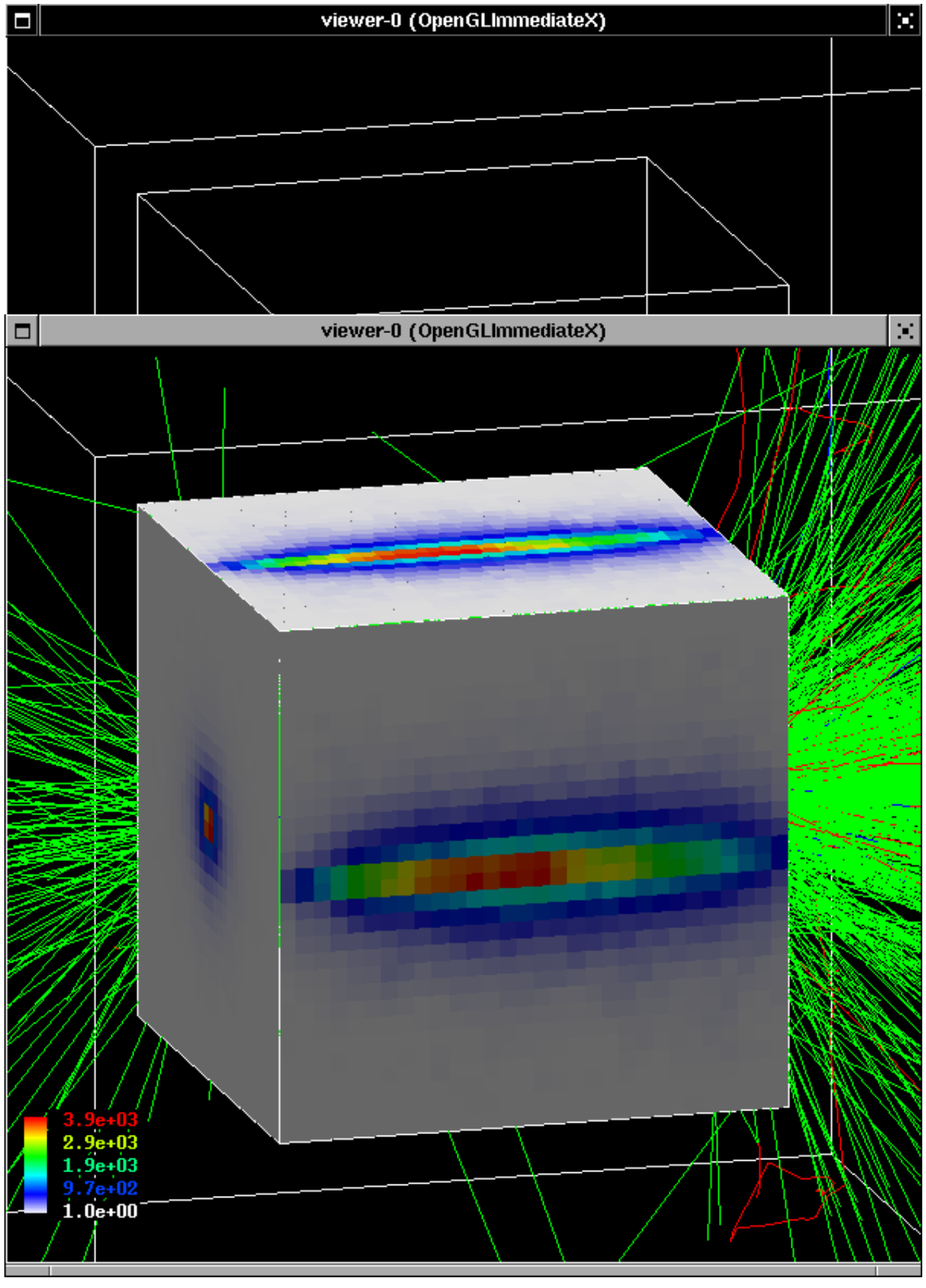
Command-based scoring

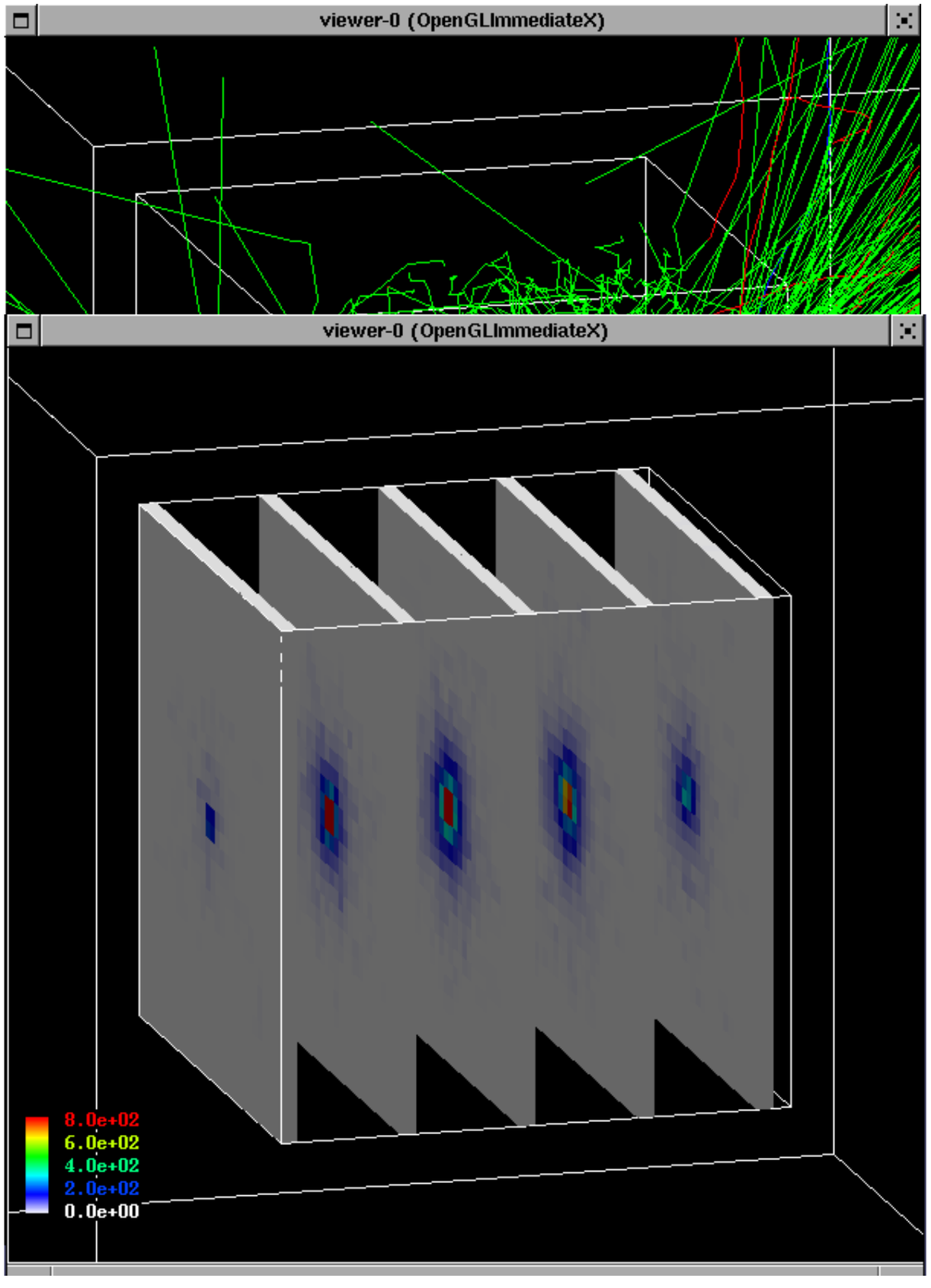
- Command-based scoring functionality offers the built-in scoring mesh and various scorers for commonly-used physics quantities such as dose, flux, etc.
- To use this functionality, access to the G4ScoringManager pointer after the instantiation of G4RunManager in your *main*().

```
#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 followings.
  1. Shape and name of the 3D scoring mesh. Currently, box is the only available shape.
     - Cylindrical mesh also available as a beta-release.
  2. Size of the scoring mesh. Mesh size must be specified as "half width" similar to the arguments of G4Box.
  3. Number of bins for each axes. Note that too many bins causes immense memory consumption.
  4. 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 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/quantitly/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.

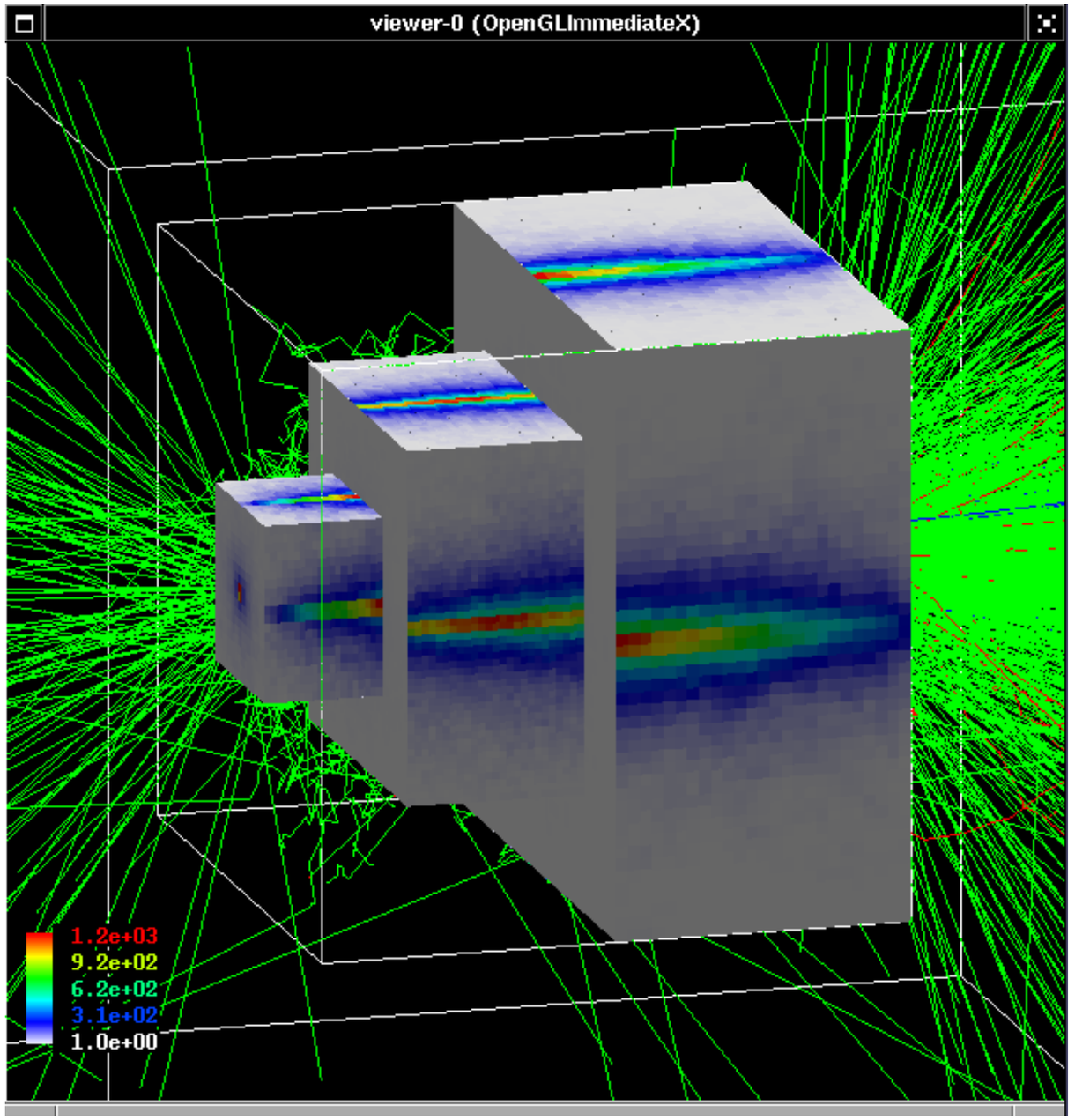
Write scores to a file

- 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 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



Summary

- Sensitive Detectors create ‘hits’
- User action classes allow user to control simulation or get information and results
  - Action classes for event generation, run, event, track, and step
- Ready-to-use scoring can be used to calculate different quantities (flux, etc)