A SIMULATION TOOLKIT

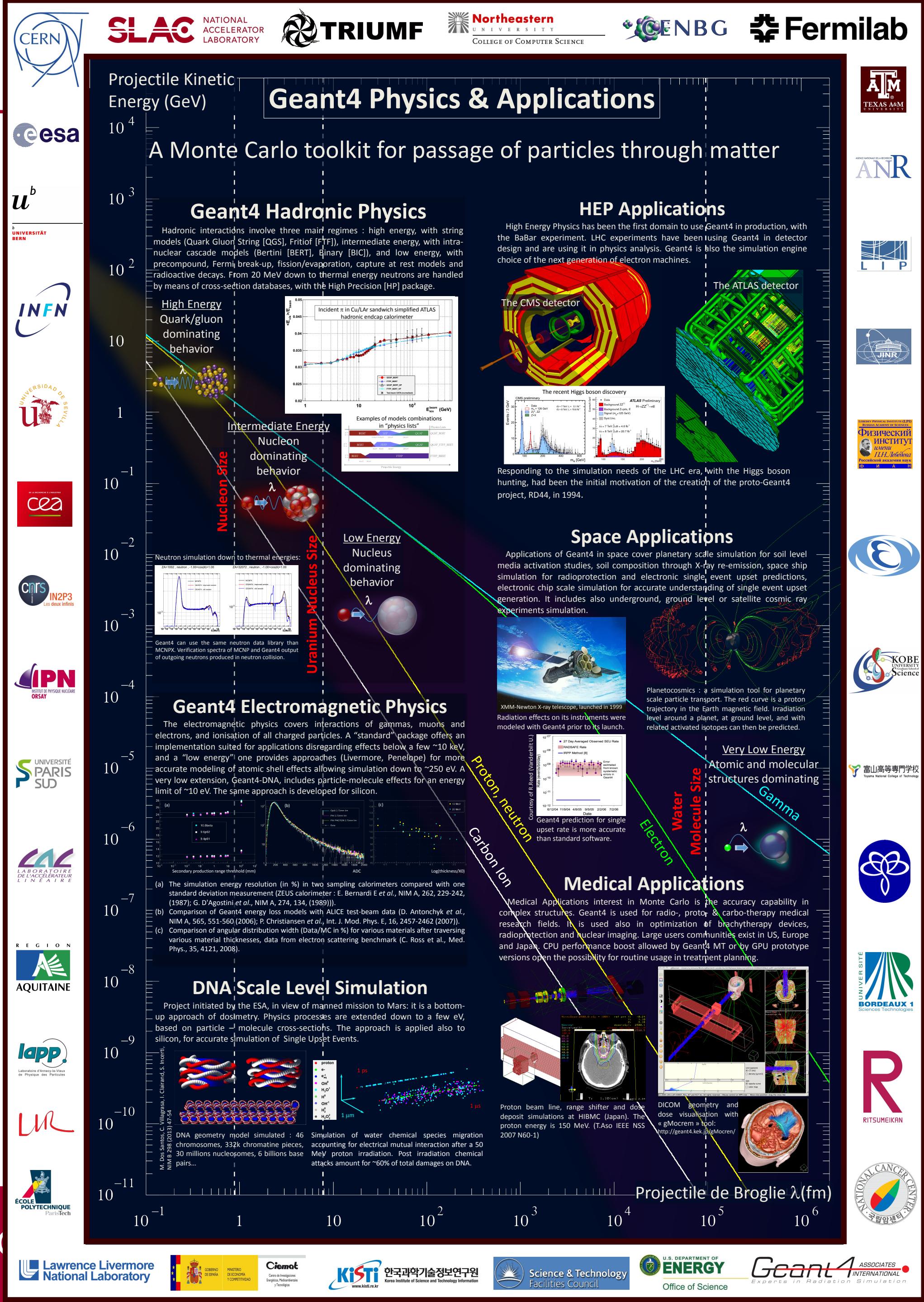
Version 10.7.p02

Geant4 Kernel
- part 2

Makoto Asai
SLAC National Accelerator Laboratory

Contents

- Selecting physics models
- Primary particle generator
- Scoring and sensitive detector


Projectile Kinetic Energy (GeV)
Projectile de Broglie λ(fm)
Nucleon Size
Uranium Nucleus Size
Water Molecule Size
Proton, neutron
Carbon Ion
Electron
Gamma
λ
Incident π in Cu/LAr sandwich simplified ATLAS hadronic endcap calorimeter
Examples of models combinations in "physics lists"
The CMS detector
The ATLAS detector
The recent Higgs boson discovery
XMM-Newton X-ray telescope, launched in 1999
Courtesy of R.Reed (Vanderbilt U.)
M. Dos Santos, C. Villagrasa, I. Clairand, S. Incerti, NIM B 298 (2013) 47-54
1 ps
1 µs
1 µm

Geant4 Physics & Applications

A Monte Carlo toolkit for passage of particles through matter

Geant4 Hadronic Physics

Hadronic interactions involve three main regimes : high energy, with string models (Quark Gluon String [QGS], Fritiof [FTF]), intermediate energy, with intra-nuclear cascade models (Bertini [BERT], Binary [BIC]), and low energy, with precompound, Fermi break-up, fission/evaporation, capture at rest models and radioactive decays. From 20 MeV down to thermal energy neutrons are handled by means of cross-section databases, with the High Precision [HP] package.

High Energy
Quark/gluon dominating behavior

Intermediate Energy
Nucleon dominating behavior

Low Energy
Nucleus dominating behavior

Neutron simulation down to thermal energies:

Geant4 can use the same neutron data library than MCNPX. Verification spectra of MCNP and Geant4 output of outgoing neutrons produced in neutron collision.

HEP Applications

High Energy Physics has been the first domain to use Geant4 in production, with the BaBar experiment. LHC experiments have been using Geant4 in detector design and are using it in physics analysis. Geant4 is also the simulation engine choice of the next generation of electron machines.

Responding to the simulation needs of the LHC era, with the Higgs boson hunting, had been the initial motivation of the creation of the proto-Geant4 project, RD44, in 1994.

Space Applications

Applications of Geant4 in space cover planetary scale simulation for soil level media activation studies, soil composition through X-ray re-emission, space ship simulation for radioprotection and electronic single event upset predictions, electronic chip scale simulation for accurate understanding of single event upset generation. It includes also underground, ground level or satellite cosmic ray experiments simulation.

XMM-Newton X-ray telescope, launched in 1999
Radiation effects on its instruments were modeled with Geant4 prior to its launch.

Planetocosmics : a simulation tool for planetary scale particle transport. The red curve is a proton trajectory in the Earth magnetic field. Irradiation level around a planet, at ground level, and with related activated isotopes can then be predicted.

Geant4 prediction for single upset rate is more accurate than standard software.

Very Low Energy
Atomic and molecular structures dominating

Geant4 Electromagnetic Physics

The electromagnetic physics covers interactions of gammas, muons and electrons, and ionisation of all charged particles. A "standard" package offers an implementation suited for applications disregarding effects below a few ~10 keV, and a "low energy" one provides approaches (Livermore, Penelope) for more accurate modeling of atomic shell effects allowing simulation down to ~250 eV. A very low extension, Geant4-DNA, includes particle-molecule effects for an energy limit of ~10 eV. The same approach is developed for silicon.

(a) The simulation energy resolution (in %) in two sampling calorimeters compared with one standard deviation measurement (ZEUS calorimeter : E. Bernardi E *et al.*, NIM A, 262, 229-242, (1987); G. D'Agostini *et al.*, NIM A, 274, 134, (1989))).
(b) Comparison of Geant4 energy loss models with ALICE test-beam data (D. Antonchyk *et al.*, NIM A, 565, 551-560 (2006); P. Christiansen *et al.*, Int. J. Mod. Phys. E, 16, 2457-2462 (2007)).
(c) Comparison of angular distribution width (Data/MC in %) for various materials after traversing various material thicknesses, data from electron scattering benchmark (C. Ross et al., Med. Phys., 35, 4121, 2008).

Medical Applications

Medical Applications interest in Monte Carlo is the accuracy capability in complex structures. Geant4 is used for radio-, proto- & carbo-therapy medical research fields. It is used also in optimization of brachytherapy devices, radioprotection and nuclear imaging. Large users communities exist in US, Europe and Japan. CPU performance boost allowed by Geant4 MT or by GPU prototype versions open the possibility for routine usage in treatment planning.

Proton beam line, range shifter and dose deposit simulations at HIBMC (Japan). The proton energy is 150 MeV. (T.Aso IEEE NSS 2007 N60-1)

DICOM geometry and dose visualisation with « gMocrem » tool: http://geant4.kek.jp/gMocren/

DNA Scale Level Simulation

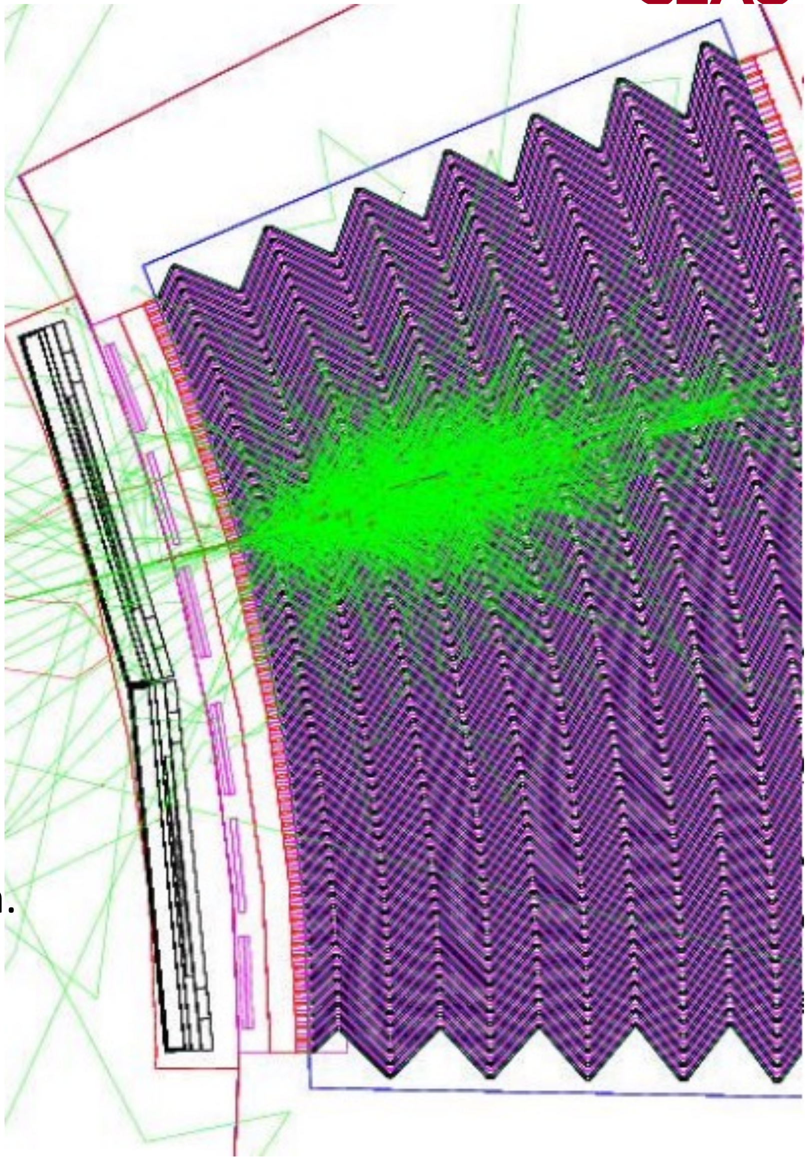
Project initiated by the ESA, in view of manned mission to Mars: it is a bottom-up approach of dosimetry. Physics processes are extended down to a few eV, based on particle – molecule cross-sections. The approach is applied also to silicon, for accurate simulation of Single Upset Events.

DNA geometry model simulated : 46 chromosomes, 332k chromatine pieces, 30 millions nucleosomes, 6 billions base pairs...

Simulation of water chemical species migration accounting for electrical mutual interaction after a 50 MeV proton irradiation. Post irradiation chemical attacks amount for ~60% of total damages on DNA.

Physics models in Geant4

- Geant4 offers
  - Electromagnetic processes
  - Hadronic and nuclear processes
  - Photon/lepton-hadron processes
  - Optical photon processes
  - Decay processes
  - Shower parameterization
  - Event biasing techniques
  - And you can plug-in more
- Geant4 provides sets of alternative physics models so that the user can freely choose appropriate models according to the type of his/her application.
  - For example, some models are more accurate than others at a sacrifice of speed.



User classes

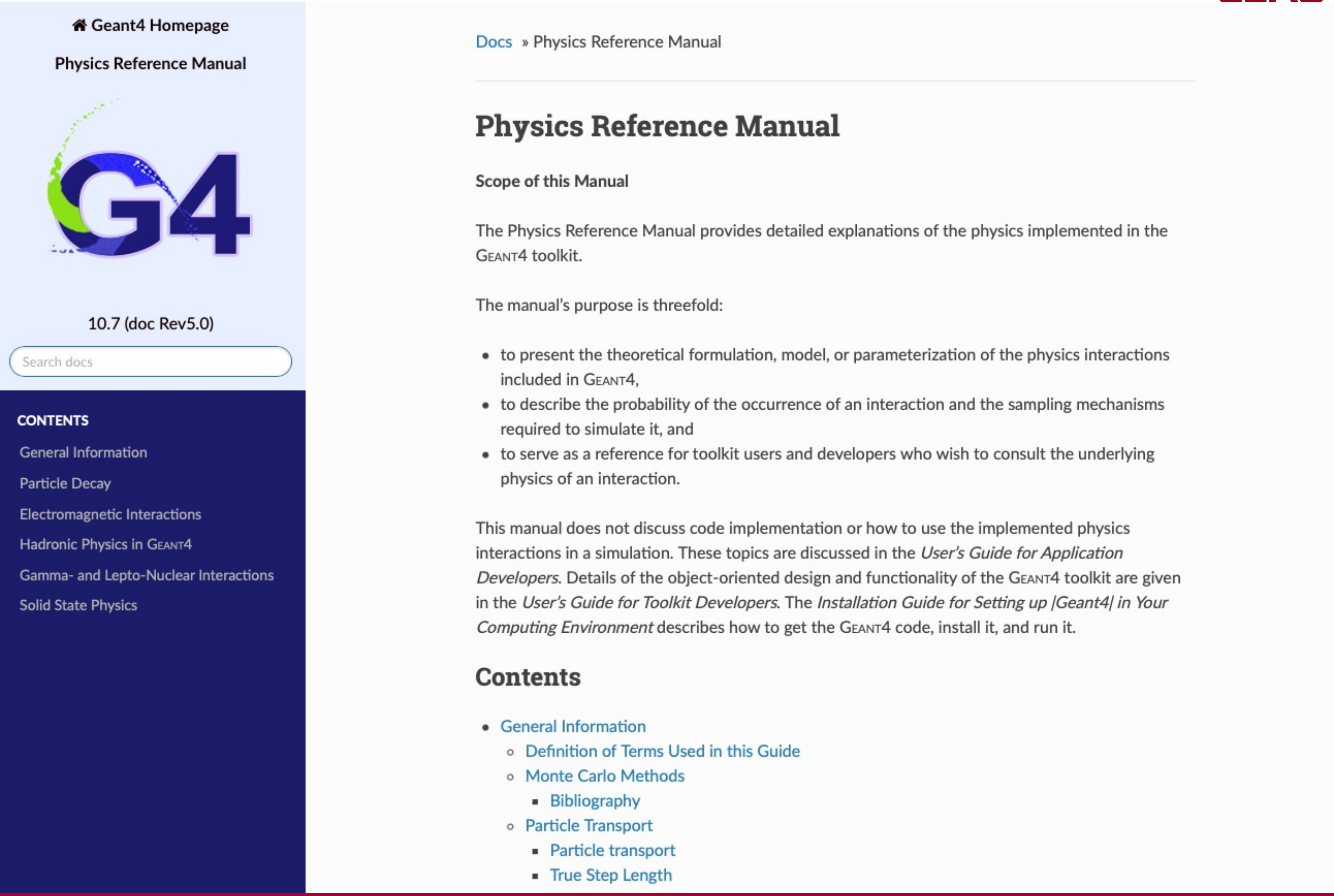
- main()
  - Geant4 does not provide *main().*

  Note : classes written in red are mandatory.
- Initialization classes
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList ←
    - G4VUserActionInitialization
- Action classes
  - Instantiated in G4VUserActionInitialization.
  - Invoked during an event loop
    - G4VUserPrimaryGeneratorAction
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction

Select physics processes

- Geant4 does not have any default particles or processes.
  - Even for the particle transportation, you have to define it explicitly.
- Derive your own concrete class from G4VUserPhysicsList abstract base class.
  - Define all necessary particles
  - Define all necessary processes and assign them to proper particles
  - Define cut-off ranges applied to the world (and each region)
- Primarily, the user's task is choosing a "pre-packaged" physics list, that combines physics processes and models that are relevant to a typical application use-cases.
  - If "pre-packaged" physics lists do not meet your needs, you may add or alternate some processes/models.
  - If you are brave enough, you may implement your physics list.

https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsReferenceManual/html/index.html



Geant4 Homepage

Physics Reference Manual

10.7 (doc Rev5.0)

Search docs

Docs » Physics Reference Manual

Physics Reference Manual

Scope of this Manual

The Physics Reference Manual provides detailed explanations of the physics implemented in the GEANT4 toolkit.

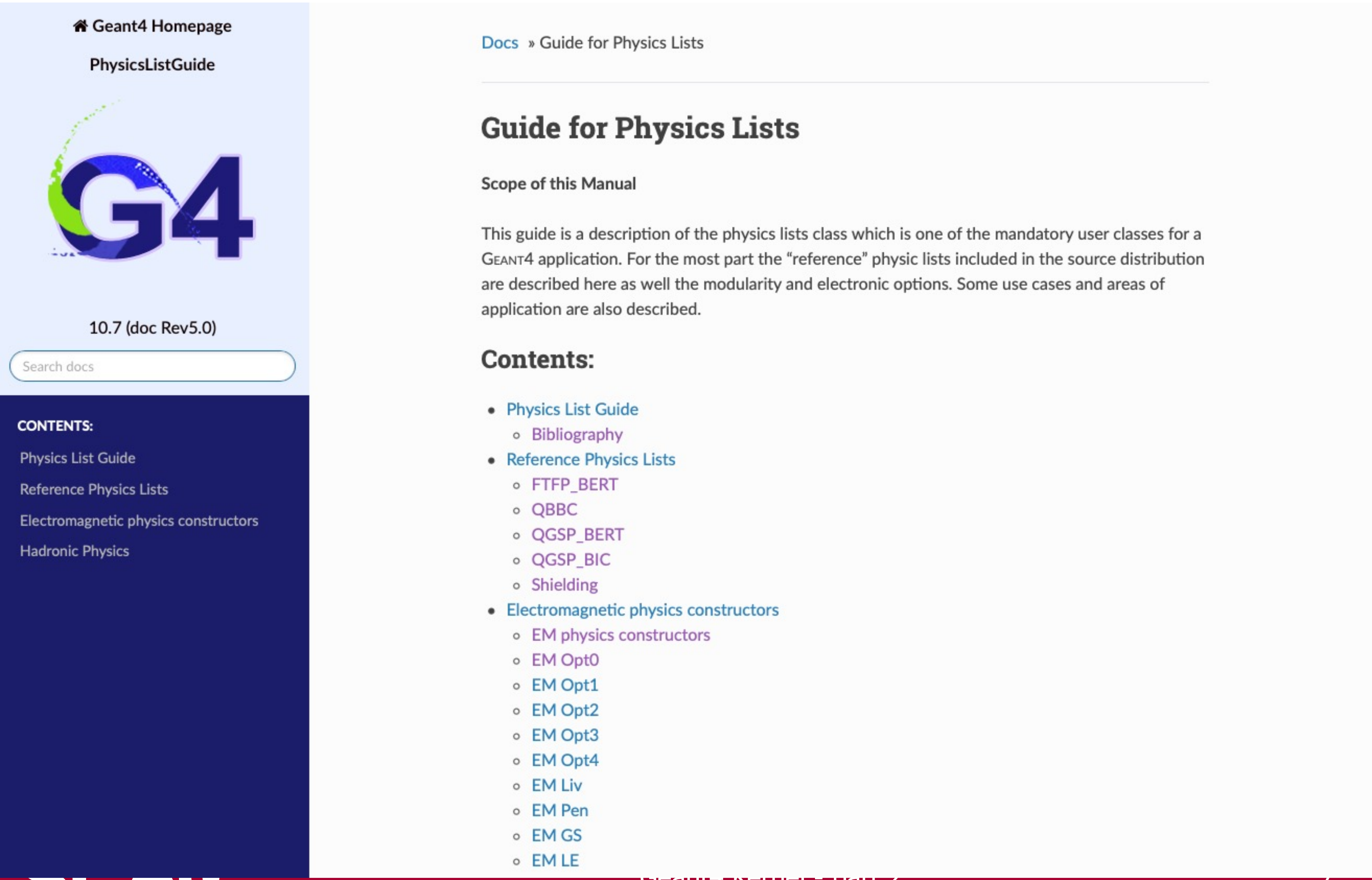
The manual's purpose is threefold:

- to present the theoretical formulation, model, or parameterization of the physics interactions included in GEANT4,
- to describe the probability of the occurrence of an interaction and the sampling mechanisms required to simulate it, and
- to serve as a reference for toolkit users and developers who wish to consult the underlying physics of an interaction.

This manual does not discuss code implementation or how to use the implemented physics interactions in a simulation. These topics are discussed in the *User's Guide for Application Developers*. Details of the object-oriented design and functionality of the GEANT4 toolkit are given in the *User's Guide for Toolkit Developers*. The *Installation Guide for Setting up |Geant4| in Your Computing Environment* describes how to get the GEANT4 code, install it, and run it.

Contents

https://geant4-userdoc.web.cern.ch/UsersGuides/PhysicsListGuide/html/index.html



Guide for Physics Lists

Scope of this Manual

This guide is a description of the physics lists class which is one of the mandatory user classes for a GEANT4 application. For the most part the "reference" physic lists included in the source distribution are described here as well the modularity and electronic options. Some use cases and areas of application are also described.

Contents:

Most-recommended physics lists

- FTFP_BERT
  - Recommended for most of the use-cases
  - “Reference” to be used as the starting point
- QBBC
  - Recommended for medical and space engineering use-cases
- Shielding
  - Recommended for radiation shielding and deep-underground experiments

```
#include ”QBBC.hh”
int main(int argc,char** argv)
{
  auto* runManager =
    G4RunManagerFactory::CreateRunManager(G4RunManagerType::Default);

  runManager->SetUserInitialization(new B1DetectorConstruction());

  runManager->SetUserInitialization(new QBBC);
```

EM physics options

- EM Option 0 is by default used by all physics lists recommended in the previous slide.
- EM Option 1 (_EMV)
  - Faster than Option 0 with relatively low accuracy
- EM Option 4 (_EMZ)
  - Most accurate, with additional CPU cost
- EM GS (_GS)
  - Same as Option 0 except Goutsmit-Sounderson multiple-scattering mode (more accurate)
- EM SS (_SS)
  - No multiple-scattering, only single-scattering is used.
  - Only for low-energy applications
- EM Liv (_LIV)
  - Livermore EM physics models are used where applicable
- EM DNA (_DNA)
  - For DNA physics/chemistry

Applying EM physics options

- Use G4PhysListFactory and specify the EM option name appended to your choice of the reference physics list.

```
#include ”G4PhysListFactory.hh”
int main(int argc,char** argv)
{
  auto* runManager = new G4MTRunManager();

  runManager->SetUserInitialization(new B1DetectorConstruction());

  G4PhysListFactory factory;
  auto* physList = factory.GetReferencePhysicsList(”FTFP_BERT_EMZ”);
  runManager->SetUserInitialization(physList);
```

Adding further extensions

- Further extensions can be added to the physics list/
  - Optical photon generation (e.g. Cherenkov)
  - Optical photon transport processes
  - Artificial track killer (e.g. neutron killer)
  - Event biasing

```
#include ”G4PhysListFactory.hh”
#include ”G4NeutronTrackingCut.hh”
int main(int argc,char** argv)
{
  auto* runManager = new G4MTRunManager();
  runManager->SetUserInitialization(new B1DetectorConstruction());

  G4PhysListFactory factory;
  auto* physList = factory.GetReferencePhysicsList(”FTFP_BERT_EMZ”);
  auto* neutronKiller = new G4NeutronTrackingCut();
  neutronKiller->SetTimeLimit(100.*CLHEP::s);
  physList->RegisterPhysics(neutronKiller);
  runManager->SetUserInitialization(physList);
```

A SIMULATION TOOLKIT

Version 10.7.p02

Primary vertex and Primary particle

NATIONAL ACCELERATOR LABORATORY

U.S. DEPARTMENT OF ENERGY

Office of Science

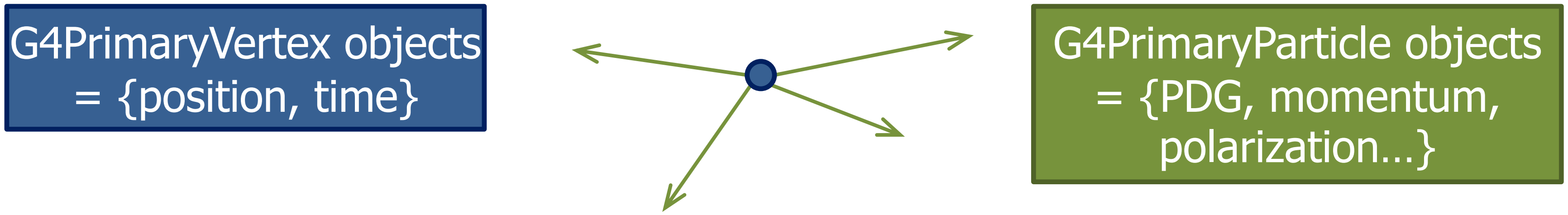
User classes

- main()
  - Geant4 does not provide *main().*

  Note : classes written in red are mandatory.
- Initialization classes
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
    - G4VUserActionInitialization
- Action classes
  - Instantiated in G4VUserActionInitialization.
  - Invoked during an event loop
    - G4VUserPrimaryGeneratorAction ←
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction

Primary vertex and primary particle

- Primary particle means particle with which you start an event.
  - E.g. particles made by the primary p-p collision, an alpha particle emitted from radioactive material, a gamma-ray from treatment head, etc.
  - Then Geant4 tracks these primary particles in your geometry with physics interactions and generates secondaries, detector responses and/or scores.
- Primary vertex has position and time. Primary particle has a particle ID, momentum and optionally polarization. One or more primary particles may be associated with a primary vertex. One event may have one or more primary vertices.


G4PrimaryVertex objects = {position, time}
G4PrimaryParticle objects = {PDG, momentum, polarization…}

- Generation of primary vertex/particle is one of the user-mandatory tasks. G4VUserPrimaryGeneratorAction is the abstract base class to control the generation.
  - Actual generation should be delegated to G4VPrimaryGenerator class. Several concrete implementations, e.g. G4ParticleGun, G4GeneralParticleSource, are provided.

G4VUserPrimaryGeneratorAction

- This class is one of mandatory user classes to control the generation of primaries.
  - This class itself should NOT generate primaries but invoke **GeneratePrimaryVertex()** method of primary generator(s) to make primaries.
- Constructor
  - Instantiate primary generator(s)
  - Set default values to it(them)
- GeneratePrimaries() method
  - Invoked at the beginning of each event.
  - Randomize particle-by-particle value(s)
  - Set these values to primary generator(s)
    - Never use hard-coded UI commands
  - Invoke **GeneratePrimaryVertex()** method of primary generator(s)
- Your concrete class of G4VUserPrimaryGeneratorAction must be instantiated in the Build() method of your G4VUserActionInitialization

G4VUserPrimaryGeneratorAction

Constructor :
Invoked only once

```
MyPrimaryGeneratorAction::MyPrimaryGeneratorAction()
{
    G4int n_particle = 1;
    fparticleGun  = new G4ParticleGun(n_particle);

    // default particle kinematic
    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
    G4ParticleDefinition* particle = particleTable->FindParticle("gamma");
    fparticleGun->SetParticleDefinition(particle);
    fparticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
    fparticleGun->SetParticleEnergy(100.*MeV);
    fparticleGun->SetParticlePosition(G4ThreeVector(0.,0.,-50*cm));
}
```

Invoked once
per each event

```
void MyPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
    fparticleGun->SetParticleMomentum(G4RandomDirection());
    fparticleGun->GeneratePrimaryVertex(anEvent);
}
```

A SIMULATION TOOLKIT

Version 10.7.p02

Primary vertex and primary particle

NATIONAL ACCELERATOR LABORATORY

U.S. DEPARTMENT OF ENERGY

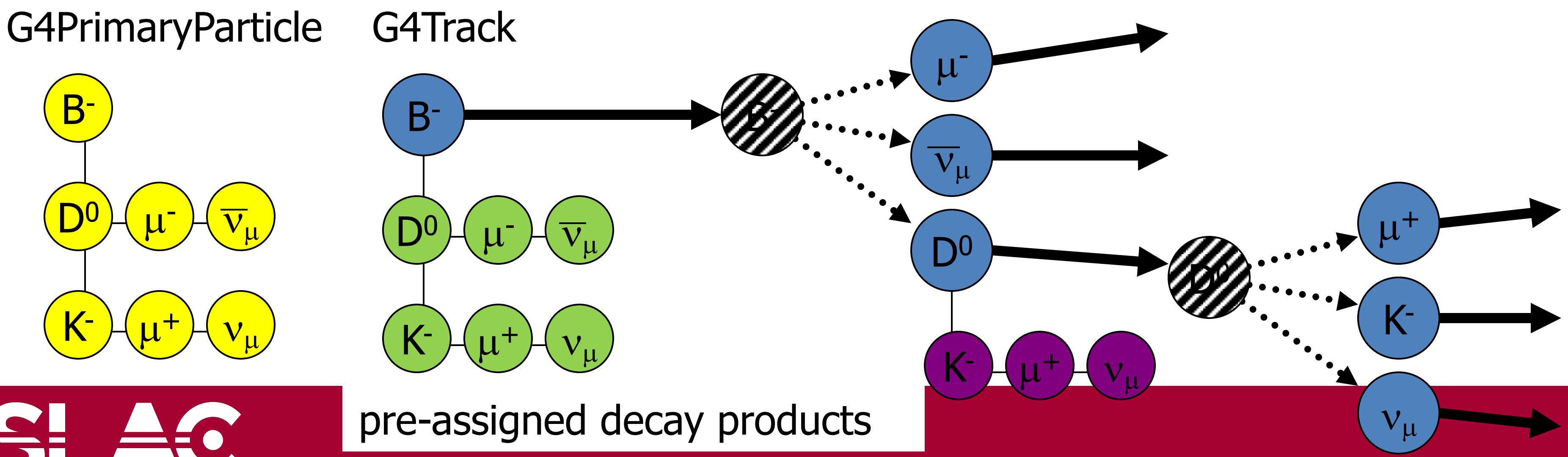
Office of Science

Primary vertices and particles

- Primary vertices and primary particles are stored in G4Event in advance to processing an event.
  - G4PrimaryVertex and G4PrimaryParticle classes
    - These classes don't have any dependency to G4ParticleDefinition nor G4Track.
  - Capability of bookkeeping decay chains
    - Primary particles may not necessarily be particles which can be tracked by Geant4.
- Geant4 provides some concrete implementations of G4VPrimaryGenerator.
  - G4ParticleGun
  - G4GeneralParticleSource
  - G4HEPEvtInterface, G4HEPMCInterface

Pre-assigned decay products

- Physics generator can assign a decay channel for each individual particle separately.
  - Decay chain can be "pre-assigned". Without pre-assigned decay channel. Geant4 naively looks up the decay table.
  - G4PromaryParticle::SetDaughter(G4PrimaryParticle*)
- A parent particle in the form of G4Track object travels in the detector, bringing "pre-assigned" decay daughters as objects of G4DynamicParticle.
  - When the parent track comes to the decay point, pre-assigned daughters become to secondary tracks, instead of randomly selecting a decay channel defined to the particle type. Decay time of the parent can be pre-assigned as well.

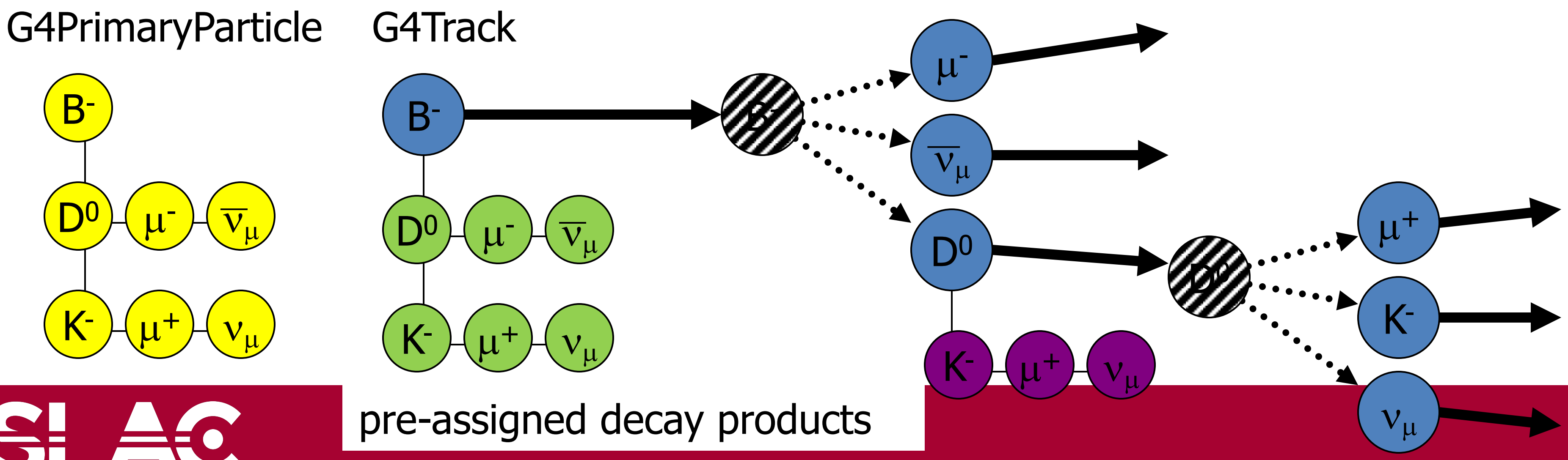


Pre-assigned decay products

```
auto* primVertex = new G4PrimaryVertex(x0,y0,z0,t0);
auto* b_minus = new G4PrimaryParticle(pdg,px,py,pz);
primVertex->SetPrimary(b_minus);
auto* d_zero = new G4PrimaryParticle(pdg,px,py,pz);
auto* mu_minus = new G4PrimaryParticle(pdg,px,py,pz);
auto* munu_bar = new G4PrimaryParticle(pdg,px,py,pz);
b_minus->SetDaughter(d_zero);
b_minus->SetDaughter(mu_minus);
b_minus->SetDaughter(munu_bar);
auto* K_minus = new G4PrimaryParticle(pdg,px,py,pz);
auto* mu_plus = new G4PrimaryParticle(pdg,px,py,pz);
auto* munu = new G4PrimaryParticle(pdg,px,py,pz);
d_zero->SetDaughter(K_minus);
d_zero->SetDaughter(mu_plus);
d_zero->SetDaughter(munu);
g4event->SetPrimaryVertex(primVertex);
```

Note:

- Momenta of daughters can be given in arbitrary flame. Daughters are Lorentz-boosted at the mother's decay point.
- If a parent particle is not a particle known to Geant4 (e.g. W-boson), it is ignored, and daughters are directly converted into G4Track objects.



A SIMULATION TOOLKIT

Version 10.7.p02

Built-in primary particle generators

G4ParticleGun

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation
- It shoots one primary particle of a certain energy from a certain point at a certain time to a certain direction.
  - Various set methods are available
  - Intercoms commands are also available for setting initial values
- One of most frequently asked questions is :

  I want "particle shotgun", "particle machinegun", etc.

- Instead of implementing such a fancy weapon, in your implementation of UserPrimaryGeneratorAction, you can
  - Shoot random numbers in arbitrary distribution
  - Use set methods of G4ParticleGun
  - Use G4ParticleGun as many times as you want
  - Use any other primary generators as many times as you want to make overlapping events

What to do and where to do

- In the constructor of your UserPrimaryGeneratorAction
  - Instantiate G4ParticleGun
  - Set default values by set methods of G4ParticleGun
    - Particle type, kinetic energy, position and direction
- In your macro file or from your interactive terminal session
  - Set values for a run
    - Particle type, kinetic energy, position and direction
- In the GeneratePrimaries() method of your UserPrimaryGeneratorAction
  - Shoot random number(s) and prepare track-by-track or event-by-event values
    - Kinetic energy, position and direction
  - Use set methods of G4ParticleGun to set such values
  - Then invoke GeneratePrimaryVertex() method of G4ParticleGun
  - If you need more than one primary tracks per event, loop over randomization and GeneratePrimaryVertex().

- examples/basic/B5/src/B5PrimaryGeneratorAction.cc is a good example to start with.

G4VUserPrimaryGeneratorAction

```
void T01PrimaryGeneratorAction::
         GeneratePrimaries(G4Event* anEvent)
{ G4ParticleDefinition* particle;
  G4int i = (int)(5.*G4UniformRand());
  switch(i)
  { case 0: particle = positron; break; ... }
  particleGun->SetParticleDefinition(particle);
  G4double pp =
    momentum+(G4UniformRand()-0.5)*sigmaMomentum;
  G4double mass = particle->GetPDGMass();
  G4double Ekin = sqrt(pp*pp+mass*mass)-mass;
  particleGun->SetParticleEnergy(Ekin);
  G4double angle = (G4UniformRand()-0.5)*sigmaAngle;
  particleGun->SetParticleMomentumDirection
           (G4ThreeVector(sin(angle),0.,cos(angle)));
  particleGun->GeneratePrimaryVertex(anEvent);
}
```

- You can repeat this for generating more than one primary particles.

G4GeneralParticleSource

- A concrete implementation of G4VPrimaryGenerator
  - Suitable especially to space applications

```
MyPrimaryGeneratorAction::
          MyPrimaryGeneratorAction()
{ generator = new G4GeneralParticleSource; }
void MyPrimaryGeneratorAction::
          GeneratePrimaries(G4Event* anEvent)
{ generator->GeneratePrimaryVertex(anEvent); }
```

- Detailed description

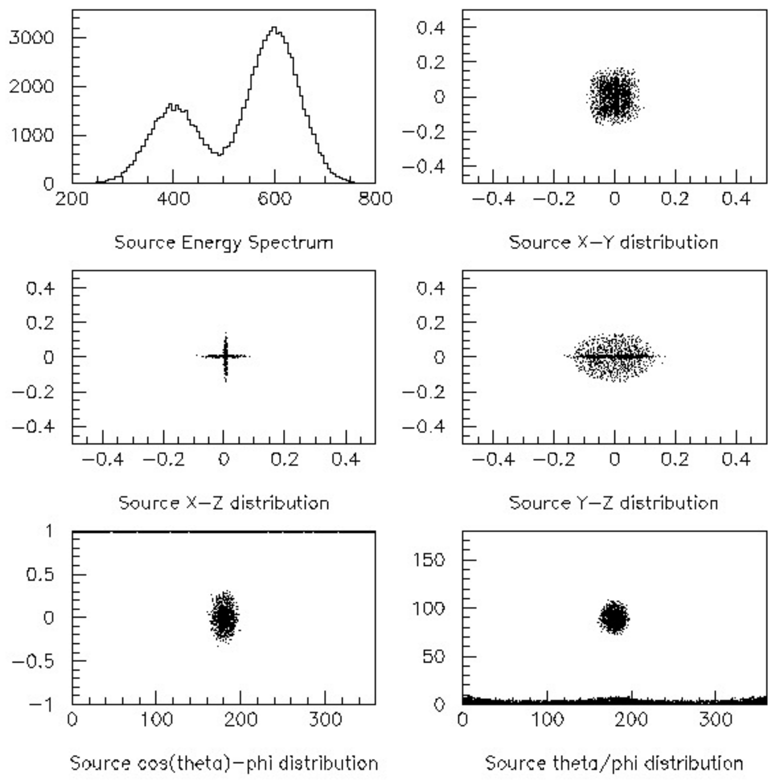
  [Section 2.7 of Application Developer's Guide]

Example commands of General Particle Source

```
# two beams in a generator
#
# beam #1
# default intensity is 1 now change to 5.
/gps/source/intensity 5.
#
/gps/particle proton
/gps/pos/type Beam
#
# the incident surface is in the y-z plane
/gps/pos/rot1 0 1 0
/gps/pos/rot2 0 0 1
#
# the beam spot is centered at the origin and is of
# 1d gaussian shape with a 1 mm central plateau
/gps/pos/shape Circle
/gps/pos/centre  0. 0. 0. mm
/gps/pos/radius 1. mm
/gps/pos/sigma_r .2 mm
#
# the beam is travelling along the X_axis with
# 5 degrees dispersion
/gps/ang/rot1 0 0 1
/gps/ang/rot2 0 1 0
/gps/ang/type beam1d
/gps/ang/sigma_r 5. deg
#
# the beam energy is in gaussian profile
# centered at 400 MeV
/gps/ene/type Gauss
/gps/ene/mono 400 MeV
/gps/ene/sigma 50. MeV
```

(macro continuation…)

```
# beam #2
# 2x the instensity of beam #1
/gps/source/add 10.
#
# this is a electron beam
/gps/particle e-
/gps/pos/type Beam
# it beam spot is of 2d gaussian profile
# with a 1x2 mm2 central plateau
# it is in the x-y plane centred at the orgin
/gps/pos/centre  0. 0. 0. mm
/gps/pos/halfx 0.5 mm
/gps/pos/halfy 1. mm
/gps/pos/sigma_x 0.1 mm
# the spread in y direction is stronger
/gps/pos/sigma_y 0.2 mm
#
#the beam is travelling along -Z_axis
/gps/ang/type beam2d
/gps/ang/sigma_x 2. deg
/gps/ang/sigma_y 1. deg
# gaussian energy profile
/gps/ene/type Gauss
/gps/ene/mono 600 MeV
/gps/ene/sigma 50. MeV
```



Interfaces to HEPEvt and HepMC

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation
- G4HEPEvtInterface
  - Suitable to /HEPEVT/ common block, which many of (FORTRAN) HEP physics generators are compliant to.
  - ASCII file input
- G4HepMCInterface
  - An interface to HepMC class, which a few new (C++) HEP physics generators are compliant to.
  - ASCII file input or direct linking to a generator through HepMC.

Version 10.7.p02

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. Be careful of thread-safety!!

$\left.\begin{matrix}\\ \\ \end{matrix}\right\}$ This talk

Version 10.7.p02

Command-based scoring

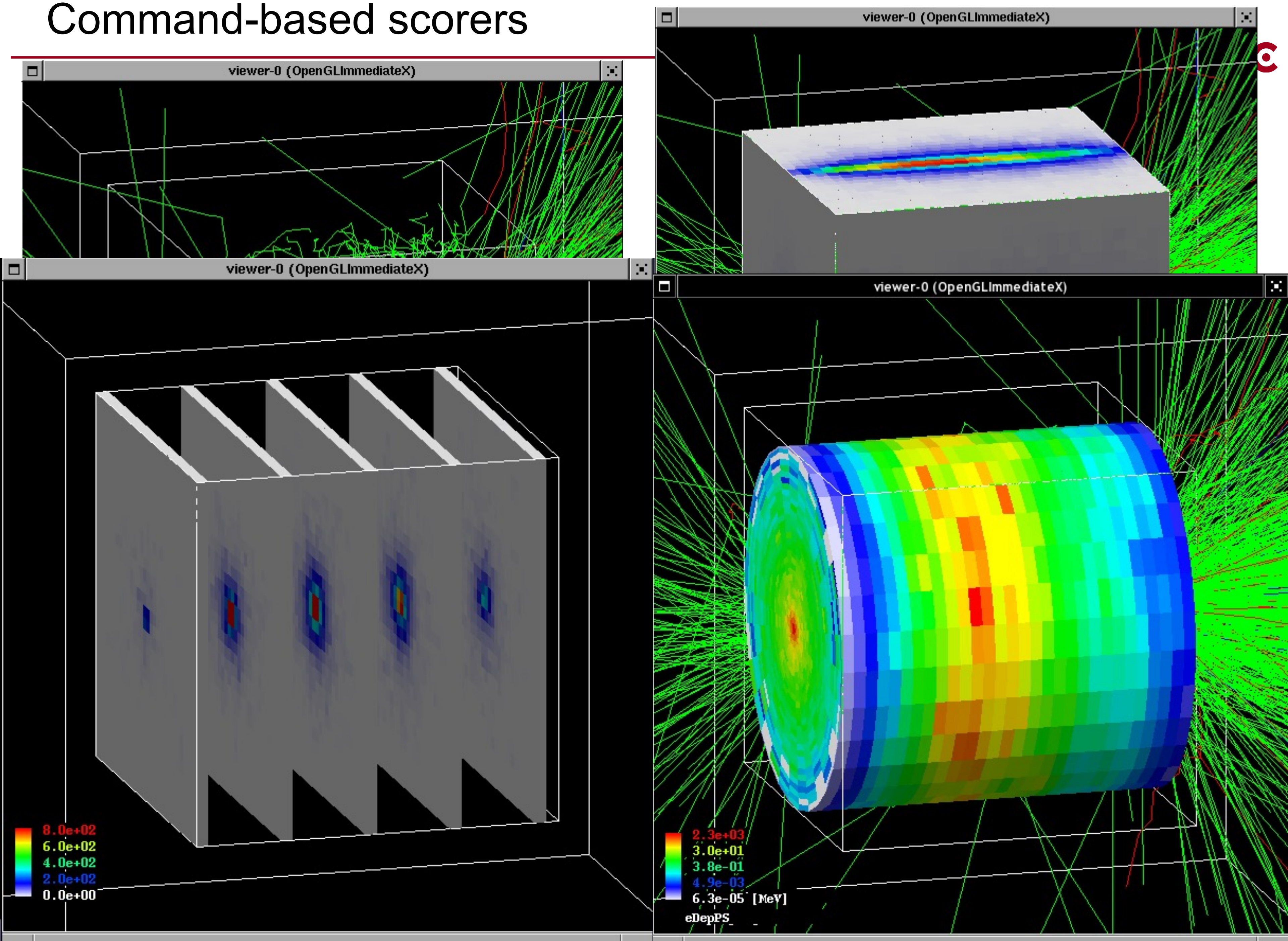
Command-based scoring

- 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()
{
  auto* runManager = new G4MTRunManager;
  auto* scoringManager = G4ScoringManager::GetScoringManager();
   …
```

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

Command-based scorers


viewer-0 (OpenGLImmediateX)
viewer-0 (OpenGLImmediateX)
viewer-0 (OpenGLImmediateX)
viewer-0 (OpenGLImmediateX)
8.0e+02
6.0e+02
4.0e+02
2.0e+02
0.0e+00
2.3e+03
3.0e+01
3.8e-01
4.9e-03
6.3e-05 [MeV]
eDepPS

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>
```

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>

- 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.

Scoring probe

New in v10.7

- This functionality allows the user to locate scoring "probes" at arbitrary locations.
  - A "probe" is a virtual cube, to which any Geant4 primitive scorers could be assigned.
  - This is an alternative to a scoring mesh, where cells of three-dimensional mesh are touching each other.

    /score/create/probe *<probeName> <halfWidth> <unit>*

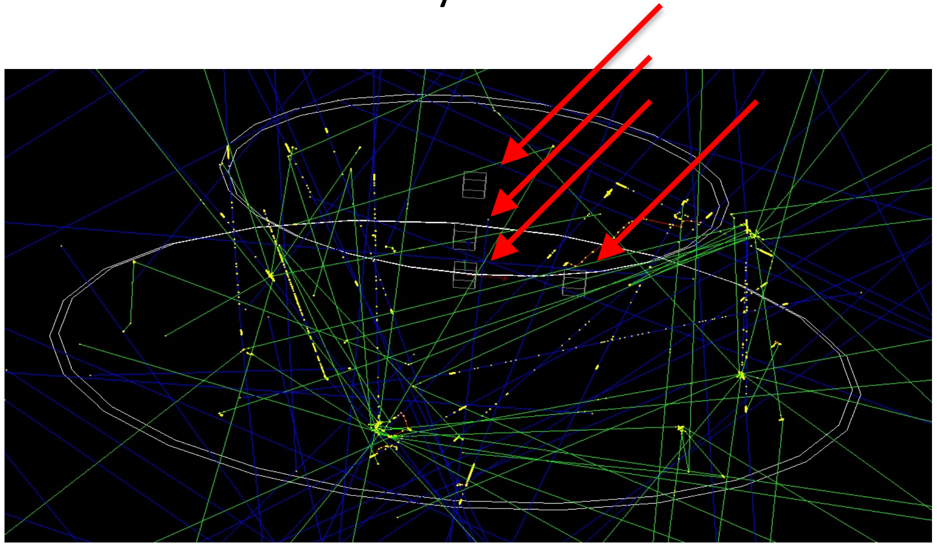
    /score/probe/locate *<x> <y> <z> <unit>*
- The user can locate an arbitrary number of probes by repeating */score/probe/locate* commands, but all of these probes are the same shape.
  - Given these probes are located in an artificial "parallel world", probes may overlap to the volumes defined in the mass geometry, as long as probes themselves are not overlapping to each other or protruding from the world volume.
- 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.
  - Please note that this overwriting material obviously affects to the simulation results, so the size and number of probes should be reasonably small to avoid significant side effects.

Scoring probe

New in v10.7

- Once a probe is defined, the user can associate arbitrary number of primitive scorers and filters like the ordinary scoring mesh.
  - All probes have the same types scorers but score individually.

```
/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/doseDeposit dose mGy
/score/quantity/volumeFlux volFlx
/score/quantity/volumeFlux protonFlux
/score/filter/particle protonFilter proton
/score/close
```



Assigning scorers to a logical volume in the real world

New in v10.7

- Alternative to the scoring probe discussed in the previous section, the user may define primitive scorers to a logical volume in the mass geometry.
  - This real-world volume scorer may take any */score/quantity/* commands.

```
/score/create/realWorldLogVol <logVolName> <level>
```

- where <*logVolName*> is the name of the logical volume that appears in the mass geometry, and it is also used as the name of the probe.
- If more than one physical volumes share the same logical volume, scores are filled individually with the copy number of the physical volume as the index.
  - If <*level*> is set, the copy number of the <*level*>-th higher ancestor in the geometrical hierarchy is used as the index.

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.