IEEE Nuclear Science Symposium and Medical Imaging Conference

Short Course

Simulation Techniques Using Geant4

Maria Grazia Pia (*INFN Genova, Italy)*
MariaGrazia.Pia@ge.infn.it

Dresden, 18 October 2008

http://www.ge.infn.it/geant4/events/nss2008/geant4course.html

This course exploits training material developed by several Geant4 Collaboration members: thanks to all of them!

Geant4 kernel

System of Units

Run, Event, Track, Step, Trajectory

Unit system

- Geant4 has **no default units**
- Units must be specified through a **System Of Units**
  - CLHEP **`SystemOfUnits.h`**
  - It was originally developed in Geant4, then moved to CLHEP
  - Most commonly used units are available
  - The user can define new units
- The system of units defined in CLHEP is based on:
  - millimetre (**`mm`**)
  - nanosecond (**`ns`**)
  - MeV (**`MeV`**)
  - positron charge (**`eplus`**)
  - degree Kelvin (**`kelvin`**)
  - the amount of substance (**`mole`**)
  - luminous intensity (**`candela`**)
  - radian (**`radian`**), steradian (**`steradian`**)
- All other units are computed from the basic ones

Using the System of Units

- To **specify a variable with units**, one multiplies its numerical value by its proper unit

```
G4double width = 12.5*m;
G4double density = 2.7*g/cm3;
```

- To **retrieve a variable in a desired unit**, divide it by the selected unit

```
G4cout << dE / MeV << " (MeV)" << G4endl;
```

- If no unit is specified, the *internal* Geant4 units will be used
  - but **this is discouraged,** since it is prone to errors!

- Geant4 can choose the **most appropriate unit** to use in output
  - Just specify the type of data (**`Length`, `Time`, `Energy`**, etc…):

```
G4cout << G4BestUnit(StepSize, "Length");
```

`StepSize` will be printed in `km`, `m`, `mm`, `fermi`… depending on its value

Defining new units

- New units can be defined via **G4UnitDefinition**
  - **G4UnitDefinition (name,symbol,category,value)**
- Example (mass thickness):
  - `G4UnitDefinition (“grammpercm2”, “g/cm2”, “MassThickness”, g/cm2);`
  - The new category “MassThickness” will be registered in the kernel in **G4UnitsTable**
- To print the list of units:
  - From the code

    G4UnitDefinition::PrintUnitsTable();
  - At run-time, as UI command:

    `Idle> /units/list`

Run

- Conceptually, a **run** is a collection of events that share the same detector conditions
- Within a run, the user cannot change
  - detector geometry
  - settings of physics processes
- Similarly to a real experiment at an accelerator, a Geant4 run starts with “Beam On”

Event

- At beginning of processing, an event contains primary particles
  - These primaries are pushed into a stack and further processed
  - When the stack becomes empty, processing of an event is over

- **G4Event** class is responsible of an event

  It has the following objects at the end of event processing
  - List of **primary vertices** and **particles**
  - **Trajectory** collection *(optional)*
  - **Hits** collections
  - **Digits** collections *(optional)*

Track

- A **track** is a snapshot of a particle
  - A Track is not a collection of steps
- A track is deleted when
  - it goes out of the world volume
  - it disappears (e.g. decay)
  - it goes down to zero kinetic energy and no “at rest” additional process is required
  - the user decides to kill it

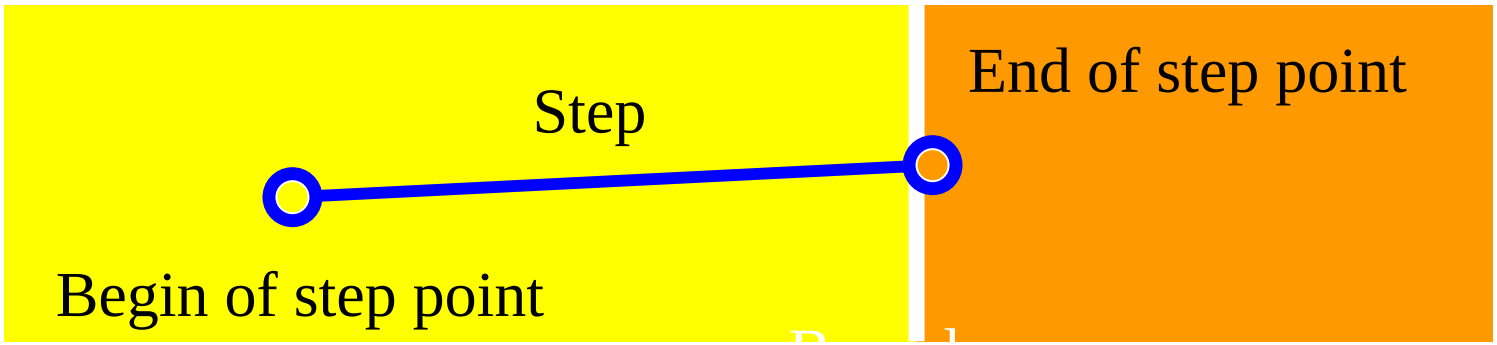
Track

A track consists of three layers of objects

- **G4Track**
  - Position, volume, track length, global ToF
  - ID of itself and its mother track
- **G4DynamicParticle**
  - Momentum, energy, local time, polarization
  - Decay channel
- **G4ParticleDefinition**
  - Mass, lifetime, charge, other physical quantities
  - Decay table
  - Shared by all G4DynamicParticle of same type

Step

- A **step** is associated to two points
- It carries information about what happened to a particle in the tracking step
  - energy loss along the step, time-of-flight spent in the step, etc.
- Each point is aware of the volume where it is
  - In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume



Trajectory

- A **trajectory** is a record of a track history
- It stores information about all steps done by the track as objects of **G4TrajectoryPoint** class

- The user can create his own trajectory class deriving from **G4VTrajectory** and **G4VTrajectoryPoint** base classes for storing any additional information useful to the simulation

- It is advised **not to store trajectories for secondary particles generated in a large scale simulation** (e.g. an electromagnetic shower) because of the memory consumption required