Geant 4

Kernel

Author: *Makoto Asai*

Run

- As an analogy of the real experiment, a run of Geant4 starts with “Beam On”.
- Within a run, the user cannot change
  - detector geometry
  - settings of physics processes

  ---> detector is inaccessible during a run
- Conceptually, a run is a collection of events which share the same detector conditions.

Event

- At beginning of processing, an event contains primary particles. These primaries are pushed into a stack.
- When the stack becomes empty, processing of an event is over.
- G4Event class represents an event. It has following objects at the end of its processing.
  - List of primary vertexes and particles
  - Trajectory collection (optional)
  - Hits collections
  - Digits collections (optional)

Track

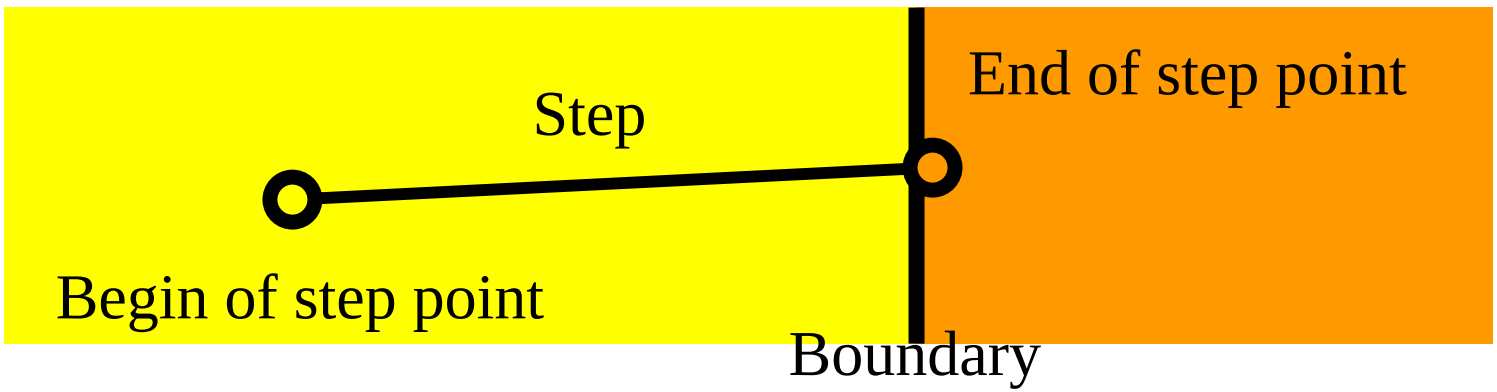
- Track is a snapshot of a particle.
- Step is a “delta” information to a track.
  - Track is not a collection of steps.
- Track is deleted when
  - it goes out of the world volume
  - it disappears (e.g. decay)
  - it goes down to zero kinetic energy and no “at rest” additional process is required
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

Track

- A track is made of three layers of class objects.
  - G4Track
    - Position, volume, track length, global ToF
    - ID of itself and mother track
  - G4DynamicParticle
    - Momentum, energy, local time, polarization
    - Pre-fixed decay channel
  - G4ParticleDefinition
    - Shared by all G4DynamicParticle of same type
    - Mass, lifetime, charge, other physical quantities
    - Decay table

Step

- Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- Each point knows the volume. 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

- Trajectory is a record of a track history. It stores some information of all steps done by the track as objects of G4TrajectoryPoint class.
- It is advised not to store trajectories for secondary particles generated in a shower because of the memory consumption.
- The user can create his own trajectory class deriving from G4VTrajectory and G4VTrajectoryPoint base classes for storing any additional information useful to the simulation.

Geant 4

System of Units

Gabriele Cosmo, CERN/IT

Unit system

- Geant4 has no default unit. To give a number, unit must be "multiplied" to the number.
  - for example :

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

  - If no unit is specified, the *internal* G4 unit will be used, but this is discouraged !
  - Almost all commonly used units are available.
  - The user can define new units.
  - Refer to CLHEP: `SystemOfUnits.h`
- Divide a variable by a unit you want to get.

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

HEP system of Units

- System of units are defined in CLHEP, 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.
- In output, Geant4 can choose the most appropriate unit to use. Just specify the *category* for the data (`Length`, `Time`, `Energy`, etc…):

```
G4cout << G4BestUnit(StepSize, “Length”);
```

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

Defining new units

- New units can be defined directly as constants, or (suggested way) 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`