Basic Concepts and Structure of Geant4

Shigeyuki Tajima

Duke/NCCU and TUNL

Slides courtesy of SLAC GEANT4 Team
Special thanks to Makoto Asai
GEANT4 Terminology

- The following keywords are often used in GEANT4:
  - Run, Event, Track, Step ...
  - Processes: (At rest, Along step, post step)
  - Cut (Production threshold)
Run in Geant4

- As an analogy of the real experiment, a run of Geant4 starts with “Beam On”.
- Within a run, the user cannot change
  - detector setup
  - settings of physics processes
- Conceptually, a run is a collection of events which share the same detector and physics conditions.
  - A run consists of one event loop.
- At the beginning of a run, geometry is optimized for navigation and cross-section tables are calculated according to materials appear in the geometry and the cut-off values defined.
- **G4RunManager** class manages processing a run, a run is represented by
  **G4Run** class or a user-defined class derived from G4Run.
  - A run class may have a summary results of the run.
- **G4UserRunAction** is the optional user hook.
Event in Geant4

- An event is the basic unit of simulation in Geant4.
- At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack.
- A track is popped up from the stack one by one and “tracked”. Resulting secondary tracks are pushed into the stack.
  - This “tracking” lasts as long as the stack has a track.
- When the stack becomes empty, processing of one event is over.
- G4Event class represents an event. It has following objects at the end of its (successful) processing.
  - List of primary vertices and particles (as input)
  - Hits and Trajectory collections (as output)
- G4EventManager class manages processing an event. G4UserEventAction is the optional user hook.
Track in Geant4

- Track is a **snapshot** of a particle.
  - It has physical quantities of **current instance** only. It does not record previous quantities.
  - Step is a “delta” information to a track. Track is not a collection of steps. Instead, a track is being updated by steps.
- Track object is deleted when
  - it goes out of the world volume,
  - it disappears (by e.g. decay, inelastic scattering),
  - it goes down to zero kinetic energy and no “AtRest” additional process is required, or
  - the user decides to kill it artificially.
- **No track object persists at the end of event.**
  - For the record of tracks, use trajectory class objects.
- **G4TrackingManager** manages processing a track, a track is represented by **G4Track** class.
- **G4UserTrackingAction** is the optional user hook.
Run, Event and Tracks

- One Run consists of:
  
  Event #1 (track #1, #2, ... track #N₁)
  Event #2 (track #1, #2, ... track #N₂)
  ...
  ...
  Event #N (track #1, #2, ... track #N_N)
A Event and Tracks (example)

- Tracking order follows ‘last in first out’ rule:
  
  T1 -> T4 -> T3 -> T6 -> T7 -> T5 -> T8 -> T2
Step in Geant4

- 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 (and material). 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.
  - Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated.
- `G4SteppingManager` class manages processing a step, a step is represented by `G4Step` class.
- `G4UserSteppingAction` is the optional user hook.
Particle in Geant4

- Particle in general has the following three properties:
  - Particle position, geometrical info
    \[\Rightarrow\text{G4Track class}\] (representing a particle to be tracked)
  - Dynamic properties (momentum, energy, spin, etc)
    \[\Rightarrow\text{G4DynamicParticle class}\] (representing an individual particle)
  - Static properties (rest mass, charge, life time, etc)
    \[\Rightarrow\text{G4ParticleDefinition class}\]
  - All G4DynamicParticle objects of the same kind of particle share
    the same G4ParticleDefinition
Processes in Geant4

- In Geant4, particle transportation is a process as well, by which a particle interacts with geometrical volume boundaries and field of any kind.
  - Because of this, shower parameterization process can take over from the ordinary transportation without modifying the transportation process.
- Each particle has its own list of applicable processes. At each step, all processes listed are invoked to get proposed physical interaction lengths.
- The process which requires the shortest interaction length (in space-time) limits the step.
- Each process has one or combination of the following natures.
  - AtRest
    - e.g. muon decay at rest
  - AlongStep (a.k.a. continuous process)
    - e.g. Celenkov process
  - PostStep (a.k.a. discrete process)
    - e.g. decay on the fly
Cuts in Geant4

- A Cut in Geant4 is a **production threshold**.
  - Not tracking cut, which does not exist in Geant4 as default.
    - All tracks are traced down to zero kinetic energy.
    - It is applied **only** for physics processes that have infrared divergence
  - Much detail will be given at later talks on physics.
Extract useful information

- 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 two ways:
  - Use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
    - You have an access to almost all information
    - Straight-forward, but do-it-yourself
  - Use Geant4 scoring functionality
    - Assign G4VSensitiveDetector to a volume
    - Hits collection is automatically stored in G4Event object, and automatically accumulated if user-defined Run object is used.
    - Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary
GEANT4 Kernel

- Geant4 consists of 17 categories
- Each category is independently maintained by a working group
- Geant4 Kernel:
  - Handles run, event, track, step, hit, trajectory
  - Provides a frameworks of geometrical representation and physics process

GEANT4 Tutorial @ TUNL Jun 22-24, 2009

This slide based on a SLAC Geant4 tutorial slide
Unit system

- Internal unit system used in Geant4 is completely hidden not only from user’s code but also from Geant4 source code implementation.

- Each hard-coded number must be multiplied by its proper unit.
  
  \[
  \text{radius} = 10.0 \times \text{cm}; \\
  \text{kineticE} = 1.0 \times \text{GeV};
  \]

- To get a number, it must be divided by a proper unit.
  \[
  \text{G4cout} \ll \text{eDep} / \text{MeV} \ll "\ [\text{MeV}]" \ll \text{G4endl};
  \]

- Most of commonly used units are provided and user can add his/her own units.

- By this unit system, source code becomes more readable and importing / exporting physical quantities becomes straightforward.
  
  - For particular application, user can change the internal unit to suitable alternative unit without affecting to the result.
Partial List of Available Units in GEANT4

- g, kg, mg, ...
- mm, cm, m, km, angstrom, fermi, cm2, m3, barn, ...
- s, ms, ns ...
- degree, radian, steradian, rad, mrad ...
- watt, newton, joule, eV, keV, MeV, GeV...
- kilovolt, volt, megavolt, ohm ...
- ampere, milliampere, microampere, nanoampere...
- weber, tesla, gauss, kilogauss, henry, farad...
- hertz, kilohertz, megahertz ...
- perCent
- kelvin, mole...

- Complete list of units is given in G4SystemOfUnits.hh
Basic types in GEANT4

- For basic types, different compiles and platforms use different value ranges. To make a program portable, the following basic types are used in GEANT4:
  
  G4int
  G4long
  G4double
  G4bool
  G4complex
  G4String

- These consists of simple ‘typedefs’ to respective types in CLHEP (Computing Libraries for High Energy Physics), etc
G4cout, G4cerr

- **G4cout** and **G4cerr** are ostream objects defined by Geant4.
  - **G4endl** is also provided.
    ```cpp
    G4cout << "Hello Geant4!" << G4endl;
    ```
- Some GUIs are buffering output streams so that they display print-outs on another window or provide storing / editing functionality.
  - The user should not use std::cout, etc.
- The user should not use std::cin for input. Use user-defined commands provided by intercoms category in Geant4.
  - Ordinary file I/O is OK.
To use Geant4, you have to...

- Geant4 is a toolkit. You have to build an application.
- To make an application, you have to
  - Define your geometrical setup
    - Material, volume
  - Define physics to get involved
    - Particles, physics processes/models
    - Production thresholds
  - Define how an event starts
    - Primary track generation
  - Extract information useful to you
- You may also want to
  - Visualize geometry, trajectories and physics output
  - Utilize (Graphical) User Interface
  - Define your own UI commands
  - etc.
User classes

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

- Initialization classes
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList

- Action classes
  - Use G4RunManager::SetUserAction() to define.
  - Invoked during an event loop
    - G4VUserPrimaryGeneratorAction
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction

Note: classes written in **yellow** are mandatory.
The main program

- Geant4 does not provide the main().
- In your main(), you have to
  - Construct G4RunManager (or your derived class)
  - Set user mandatory classes to RunManager
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
    - G4VUserPrimaryGeneratorAction
- You can define VisManager, (G)UI session, optional user action classes, and/or your persistency manager in your main().
Three Mandatory Classes

- Three important classes that users must implement:
  - Define material and geometry
    ==> G4VUserDetectorConstruction class
  - Select appropriate particles and processes and define production threshold(s)
    ==> G4VUserPhysicsList class
  - Define the way of primary particle generation
    ==> G4VUserPrimaryGeneratorAction class
Describe your detector

- Derive your own concrete class from `G4VUserDetectorConstruction` abstract base class.
- In the virtual method `Construct()`,
  - Instantiate all necessary materials
  - Instantiate volumes of your detector geometry
  - Instantiate your sensitive detector classes and set them to the corresponding logical volumes
- Optionally you can define
  - Regions for any part of your detector
  - Visualization attributes (color, visibility, etc.) of your detector elements
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)
- Geant4 provides lots of utility classes/methods and examples.
  - "Educated guess" physics lists for defining hadronic processes for various use-cases.
Generate primary event

- Derive your concrete class from `G4VUserPrimaryGeneratorAction` abstract base class.
- Pass a `G4Event` object to one or more primary generator concrete class objects which generate primary vertices and primary particles.
- Geant4 provides several generators in addition to the `G4VPrimaryParticleGenerator` base class.
  - `G4ParticleGun`
  - `G4HEPEvtInterface`, `G4HepMCInterface`
    - Interface to `/hepevt/` common block or HepMC class
  - `G4GeneralParticleSource`
    - Define radioactivity
Summary

- GEANT4 is a toolkit which consists of 17 categories.
- Run consists of Events, Tracks, and Steps.
- Track is a ‘snapshot’ of a particle:
- Particle is tracked until zero kinetic energy (production threshold must be set)
- Users need to implement three mandatory classes for materials&geometry, selecting physics processes, and primary particle generation