Geant 4

Basic structure of the Geant4 Simulation Toolkit

http://cern.ch/geant4

The full set of lecture notes of this Geant4 Course is available at
http://www.ge.infn.it/geant4/events/nss2003/geant4course.html

Contents

- Basic concepts in Geant4
- Geant4 architecture
  - Category structure
  - System of units
  - Intercoms and G4cout
- User classes

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 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.
- 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.

Event in Geant4

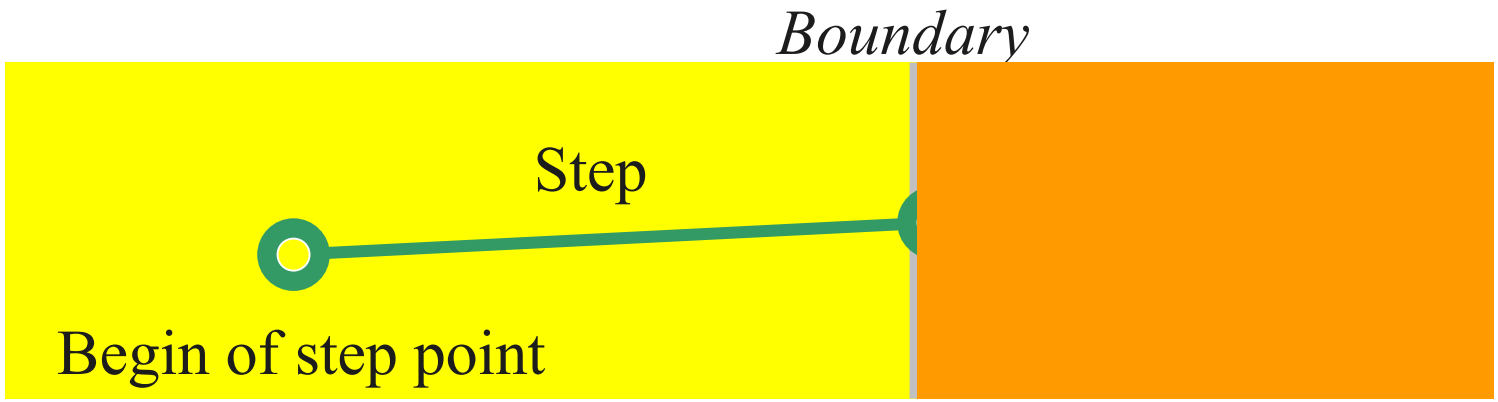
- At beginning of processing, an event contains primary particles. These primaries are pushed into a stack.
- When the stack becomes empty, processing of an event is over.
- G4Event class represents an event. It has following objects at the end of its processing.
  - List of primary vertexes and particles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

Track in Geant4

- Track is a snapshot of a particle.
  - It has only position and physical quantities of current instance.
- 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 "AtRest" additional process is required
  - the user decides to kill it
- No track object persists at the end of event.
  - For the record of track, use trajectory class objects.

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 two volumes, boundary processes such as transition radiation or refraction could be simulated.



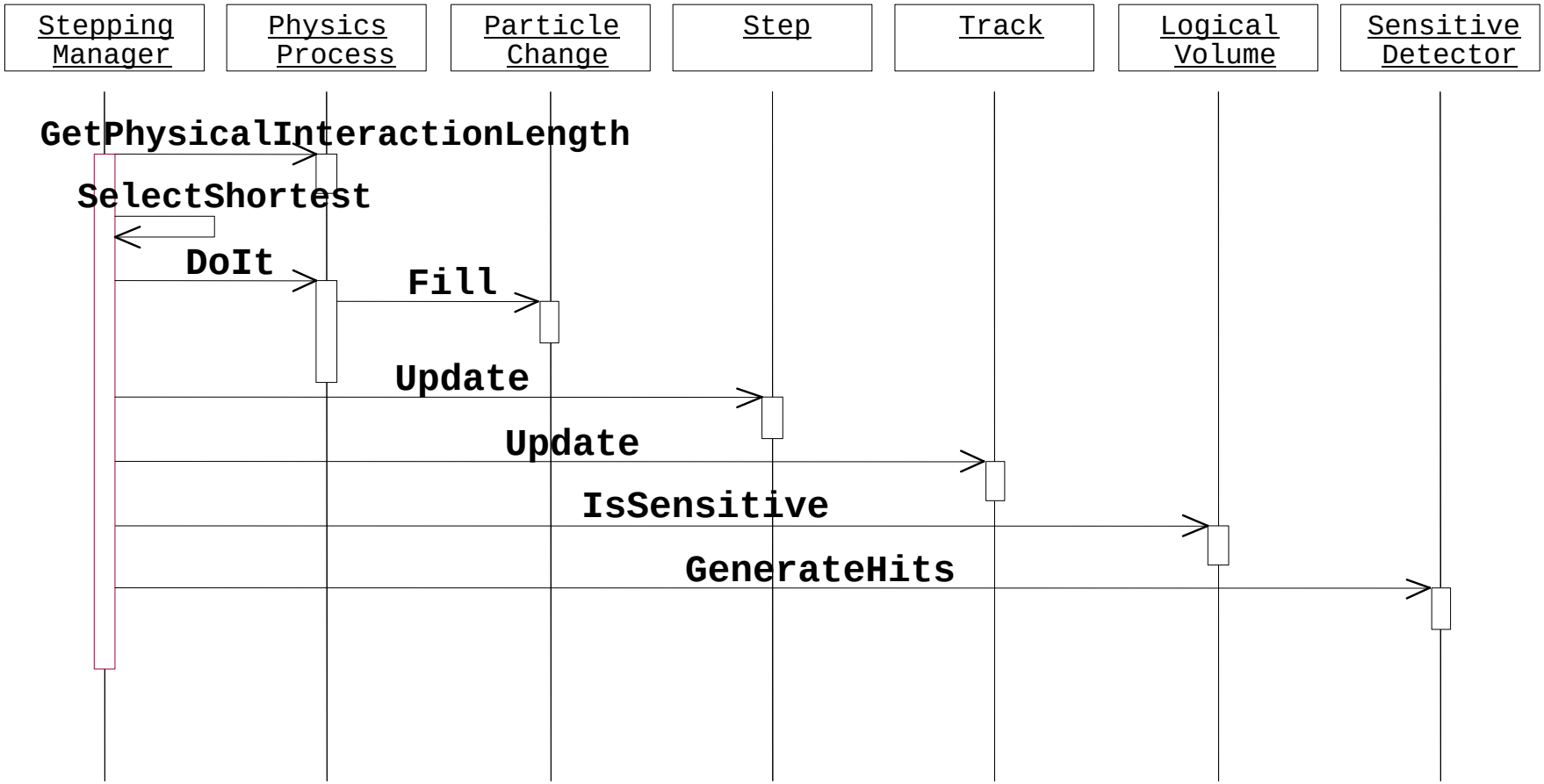
Tracking and processes

- Geant4 tracking is general.
  - It is independent of
    - the particle type
    - the physics processes involving to a particle
  - It gives the chance to all processes
    - To contribute to determining the step length
    - To contribute any possible changes in physical quantities of the track
    - To generate secondary particles
    - To suggest changes in the state of the track
      - e.g. to suspend, postpone or kill it.

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.
  - Shower parameterization process can take over from the ordinary transportation.
- 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.

How Geant4 runs (one step)

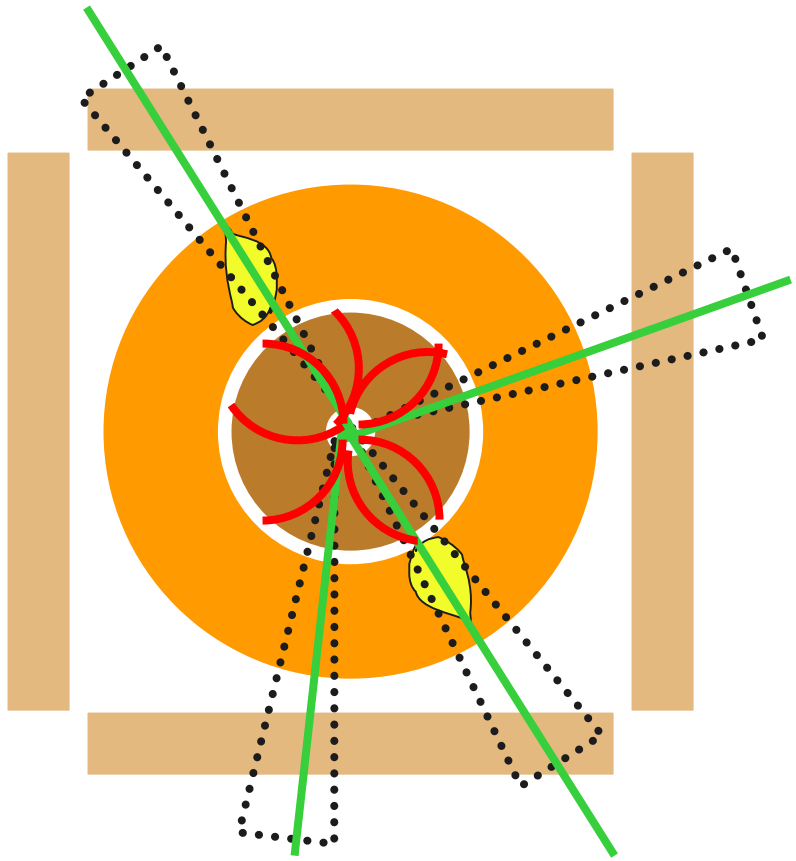


Cuts in Geant4

- A Cut in Geant4 is a production threshold.
  - Only for physics processes that have infra-red divergence
  - Not tracking cut, which does not exist in Geant4
- Energy threshold must be determined at which discrete energy loss is replaced by continuous loss
  - Old way:
    - Track primary until cut-off is reached, calculate continuous loss and dump it at that point, stop tracking primary
    - Create secondaries only above cut-off, or add to continuous loss of primary for less energetic secondaries
  - Geant4 way:
    - specify range (which is converted to energy for each material) at which continuous loss begins, track primary down to zero range
    - Create secondaries only above specified range, or add to continuous loss of primary for less energetic secondaries

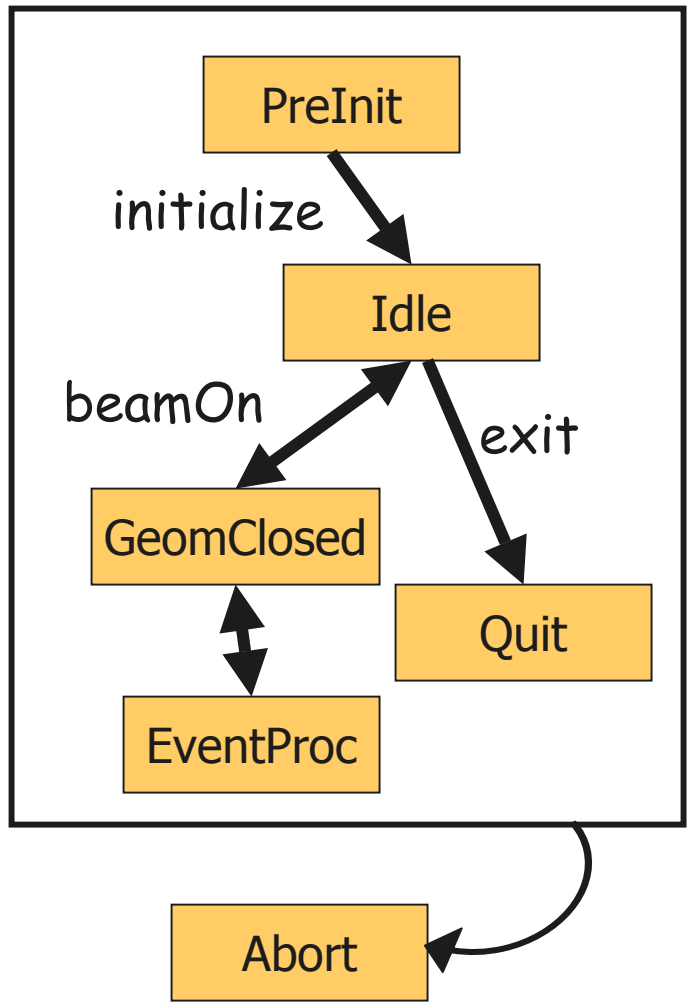
Stack

- G4Track is a class object, thus it is easy to treat suspending or postponing tracks. For example,
  - Suspend tracks at the entrance of calorimeter, i.e. simulate all tracks in tracking region before generating showers.
  - Suspend a “looper” track after certain time and postpone it to next event.
- Prioritized tracking without performance cost
- Well-thought prioritization/abortion of tracks/events makes entire simulation process much more efficient.



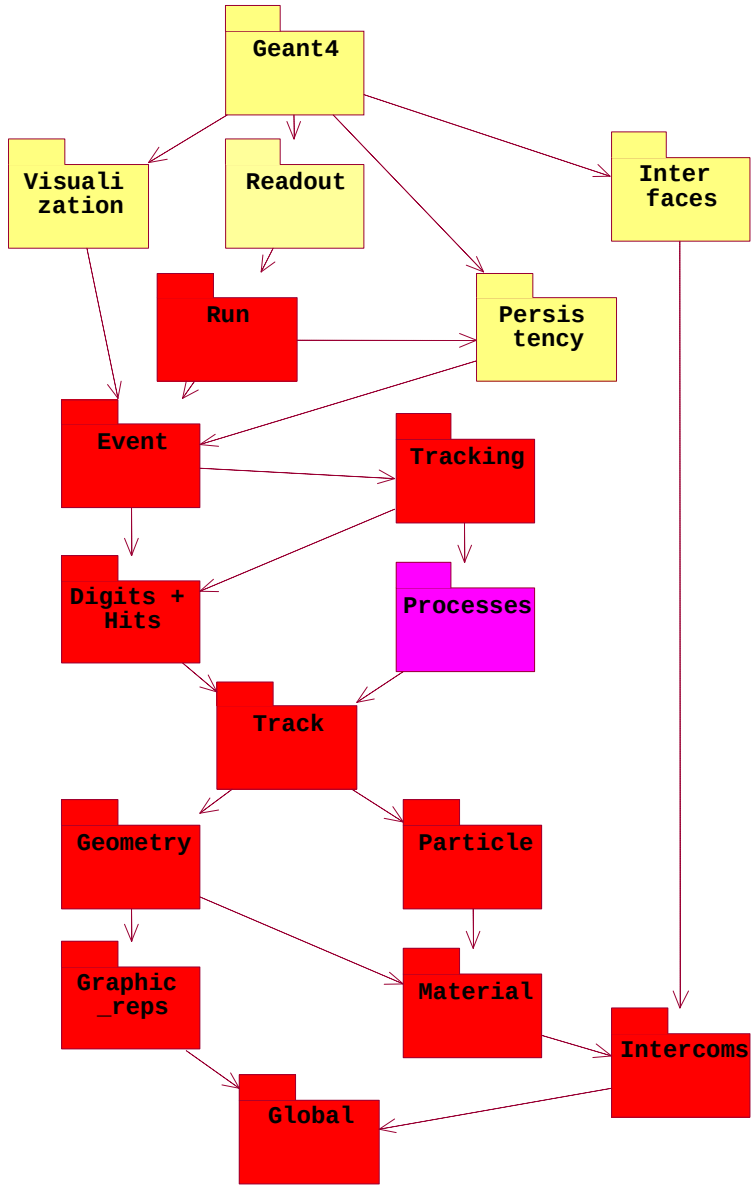
Geant4 as a state machine

- Geant4 has six application states.
  - G4State_PreInit
    - Material, Geometry, Particle and/or Physics Process need to be initialized/defined
  - G4State_Idle
    - All necessary initializations are done
    - Ready to start a run
    - Or, ready to modify geometry/physics to proceed to the next run
  - G4State_GeomClosed
    - Geometry is optimized
    - Cross-section tables are updated
    - Ready to process an event
  - G4State_EventProc
    - An event is processing
  - G4State_Quit
    - (Normal) termination
  - G4State_Abort
    - A fatal exception occurred and program is aborting



Geant4 kernel

- Geant4 consists of 17 categories.
  - Independently developed and maintained by WG(s) responsible to each (sub-)category.
  - Interfaces between categories are maintained by the global architecture WG.
- Geant4 Kernel
  - Handles run, event, track, step, hit, trajectory.
  - Provides framework for, or interfaces to
    - physics processes
    - Visualization drivers
    - (G)UI
    - Persistency mechanism
    - Histogramming
    - User's framework



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.

```
radius = 10.0 * cm;
kineticE = 1.0 * GeV;
```

- To get a number, it must be divided by a proper unit.

```
G4cout << eDep / MeV << " [MeV]" << G4endl;
```

- Most of commonly used units are provided and user can add his/her own units.
- By this unit system, importing / exporting physical quantities becomes straightforward and source code becomes more readable.
  - For particular application, user can change the internal unit with suitable precision without affecting to the result.

Intercoms

- “Intercoms” category handles the framework mechanism of defining and delivering commands.
  - Exportable to any other application
    - Independent to other Geant4 categories
  - Strong type and range checking
    - Range description by C++ syntax

      ```
      aCmd->SetRange(“x>0. && y>0.”);
      ```
  - Dynamic command definition / activation
  - Commands can be hard-coded or issued by (G)UI.
- Macro file
  - Recursive variable definition
  - Loop

G4cout, G4cerr

- G4cout and G4cerr are *ostream* objects defined by Geant4.
  - G4endl is also provided.
- 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.

User classes

- Initialization classes
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
- Action classes
  - Invoked during an event loop
    - G4VUserPrimaryGeneratorAction
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction
- main()
  - Geant4 does not provide *main()*.

Note : classes written in Red are mandatory.

Describe your detector

- Derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- In the virtual method *Construct*(),
  - Construct all necessary materials
  - Construct volumes of your detector geometry
  - Construct your sensitive detector classes and set them to the detector volumes
- Optionally you can define
  - Regions for any part of your detector
  - Visualization attributes 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

Optional user action classes

- All user action classes, methods of which are invoked during “Beam On”, must be constructed in the user’s *main*() and must be set to the RunManager.
- G4UserRunAction
  - BeginOfRunAction(const G4Run*)
    - Define histograms
  - EndOfRunAction(const G4Run*)
    - Store histograms
- G4UserEventAction
  - BeginOfEventAction(const G4Event*)
    - Event selection
    - Define histograms
  - EndOfEventAction(const G4Event*)
    - Analyze the event
- G4UserStackingAction
  - PrepareNewEvent()
    - Reset priority control
  - ClassifyNewTrack(const G4Track*)
    - Invoked every time a new track is pushed
    - Classify a new track -- priority control
      - Urgent, Waiting, PostponeToNextEvent, Kill
  - NewStage()
    - Invoked when the Urgent stack becomes empty
    - Change the classification criteria
    - Event filtering (Event abortion)
- G4UserTrackingAction
  - PreUserTrackingAction(const G4Track*)
    - Decide trajectory should be stored or not
    - Create user-defined trajectory
  - PostUserTrackingAction(const G4Track*)
- G4UserSteppingAction
  - UserSteppingAction(const G4Step*)
    - Kill / suspend / postpone the track
    - Draw the step (for a track not to be stored by a trajectory)

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()*.

Select (G)UI

- In your *main*(), according to your computer environments, construct a G4UIsession concrete class provided by Geant4 and invoke its *sessionStart*() method.
- Geant4 provides
  - G4UIterminal -- C- and TC-shell like character terminal
  - G4GAG -- Tcl/Tk or Java PVM based GUI
  - G4Wo -- Opacs
  - G4JAG -- Interface to JAS (Java Analysis Studio)
  - G4UIBatch -- Batch job with macro file

Visualization

- Derive your own concrete class from G4VVisManager according to your computer environments.
- Geant4 provides interfaces to graphics drivers
  - DAWN -- Fukui renderer
  - WIRED
  - RayTracer -- Ray tracing by Geant4 tracking
  - OPACS
  - OpenGL
  - OpenInventor
  - VRML

Environment variables

- You need to set following environment variables to compile, link and run Geant4-based simulation.
  - Mandatory variables
    - G4SYSTEM – OS (e.g. Linux-g++)
    - G4INSTALL – base directory of Geant4
    - G4WORKDIR – your temporary work space
    - CLHEP_BASE_DIR – base directory of CLHEP
  - Variables for physics processes in case corresponding processes are used
    - G4LEVELGAMMADATA - photon evaporation
    - G4LEDATA - cross-sections for Low-E EM module
    - G4RADIOACTIVEDATA - radioactive decay
    - NeutronHPCrossSections - neutron cross-section
  - Additional variables for GUI/Vis/Analysis