Kernel

I. Hrivnacova, IPN Orsay

Credits: M. Asai, SLAC and others

Geant4 ED PHENIICS Tutorial,
13 - 17 May 2019, Orsay

Outline

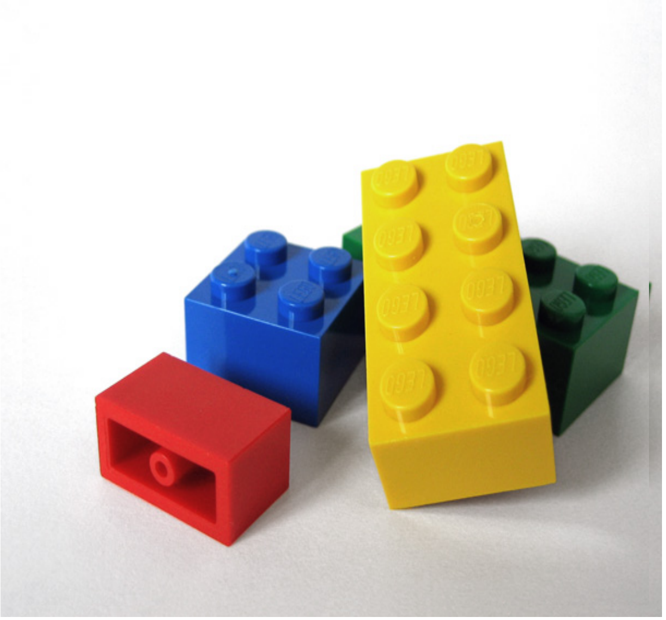
- How does it work ?
- Geant4 kernel classes
  - Run, event, track, step, classes to define particle
  - Tracking and processes
  - Application states
- User application classes

How Does It Work ?

Geant4 and User Application

- Geant4 provides building blocks (the bricks)

- Users have to assemble them to describe their scenario in their application program



Geant4 and User Application (2)

- Geant4 provides building blocks (bricks)

- Users have to assemble them to describe their scenario in their application program

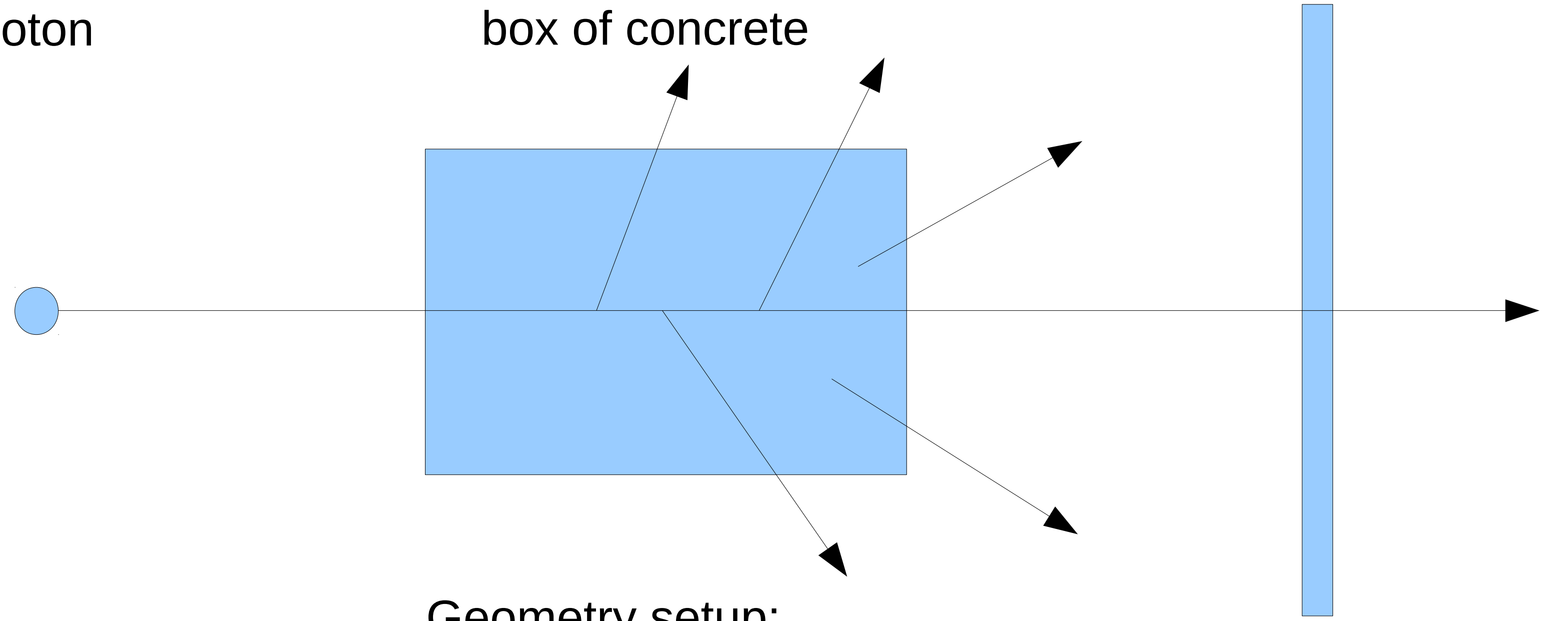
Geant4 and User Application (3)

User Application

scoring device
(detector)

proton

box of concrete



Primary generator

Geometry setup:
Volumes, materials, ...

Sensitive Detector

Geant4

Users have first to define their experimental setup via Geant4 toolkit classes

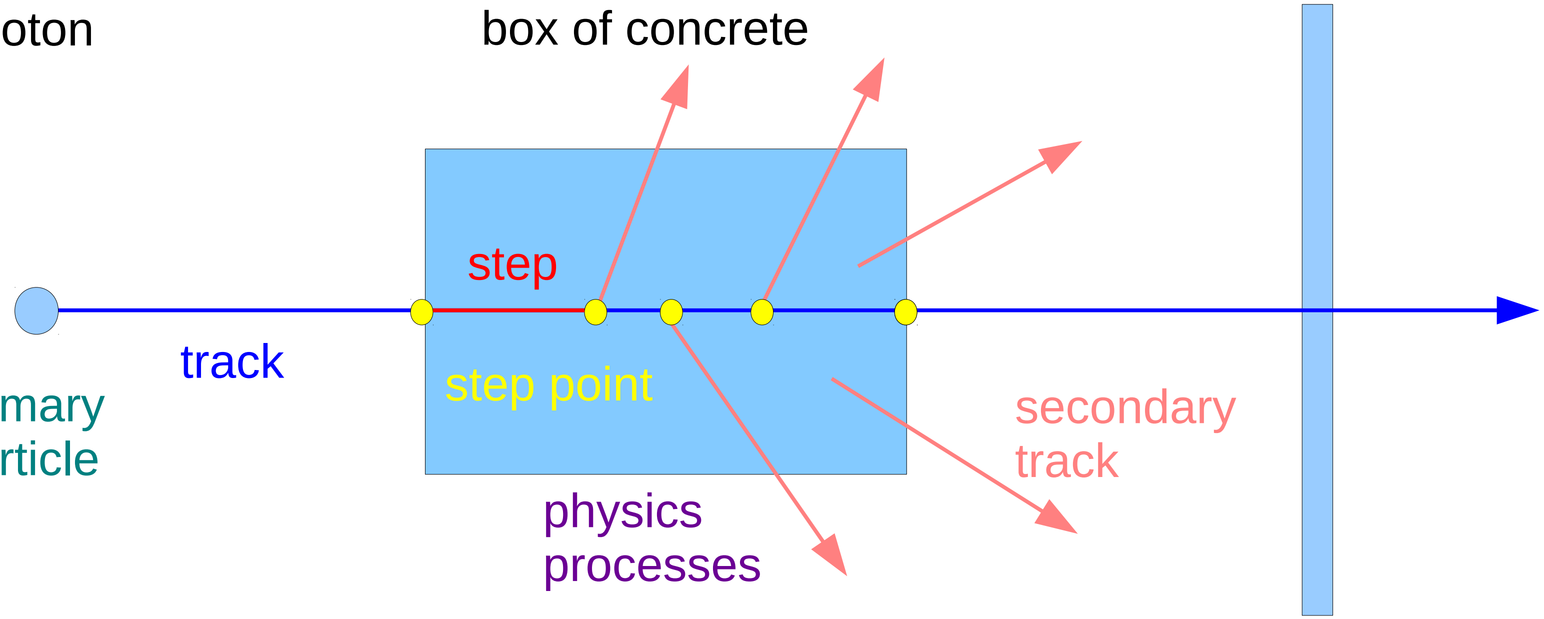
Geant4 and User Application (4)

User Application

scoring device
(detector)

proton

box of concrete



step

track

step point

primary
particle

secondary
track

physics
processes

Primary generator

Geometry setup:
Volumes, materials, ...

Sensitive Detector

Geant4

Geant4 then tracks the defined primary particles and let them interact with the materials present in geometry

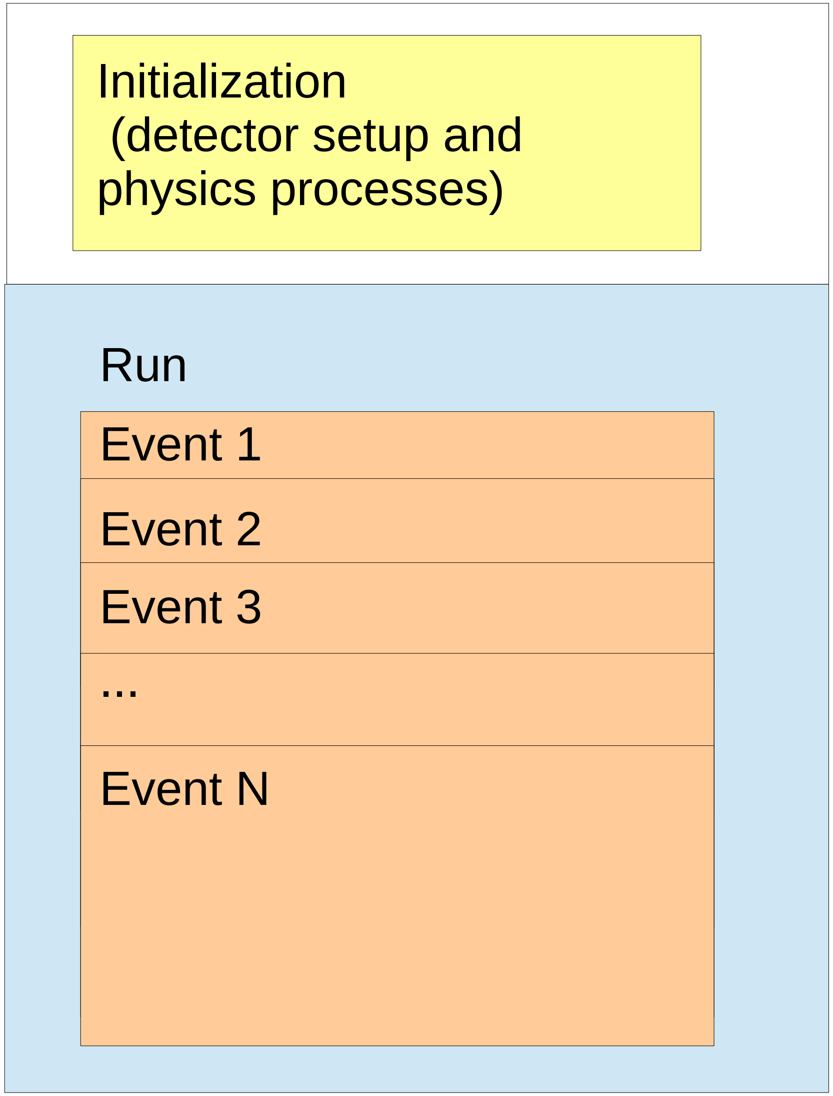
Geant4 Application

- User defines
  - Detector geometry, physics setup and primary particles in sets called (primary) events
- Geant4 kernel then loops over events
- In each event:
  - Loops over primaries
  - Each primary
    - Is tracked through the detector undergoing the registered physics processes
    - Which may create secondary particles (daughters)
  - It tracks also its daughters
  - Each track
    - Processed via steps

Geant4 Kernel Classes

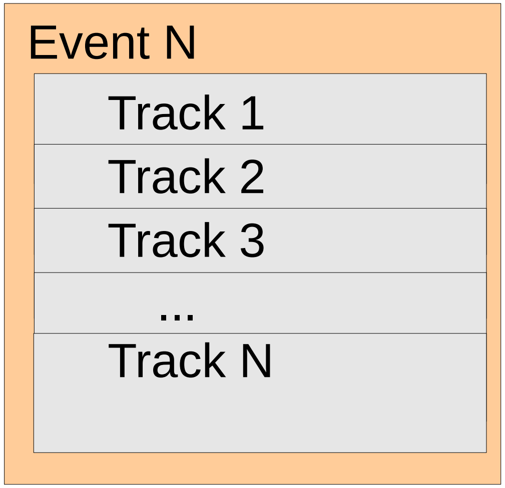
Geant4 Run

- As an analogy of the real experiment, a run of Geant4 starts with "Beam On"
- Conceptually, a run is a collection of events which share the same detector and physics conditions.
- A run consists of one event loop
- 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.



Event in Geant4

- An event is the basic unit of simulation in Geant4.
- At beginning of processing, primary tracks are generated and 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.
- G4EventManager class manages processing an event
- G4Event class represents an event. At the end of its (successful) processing, it has:
  - List of primary vertices and particles (as input)
  - Hits and Trajectory collections (as output)



Event N

Track 1

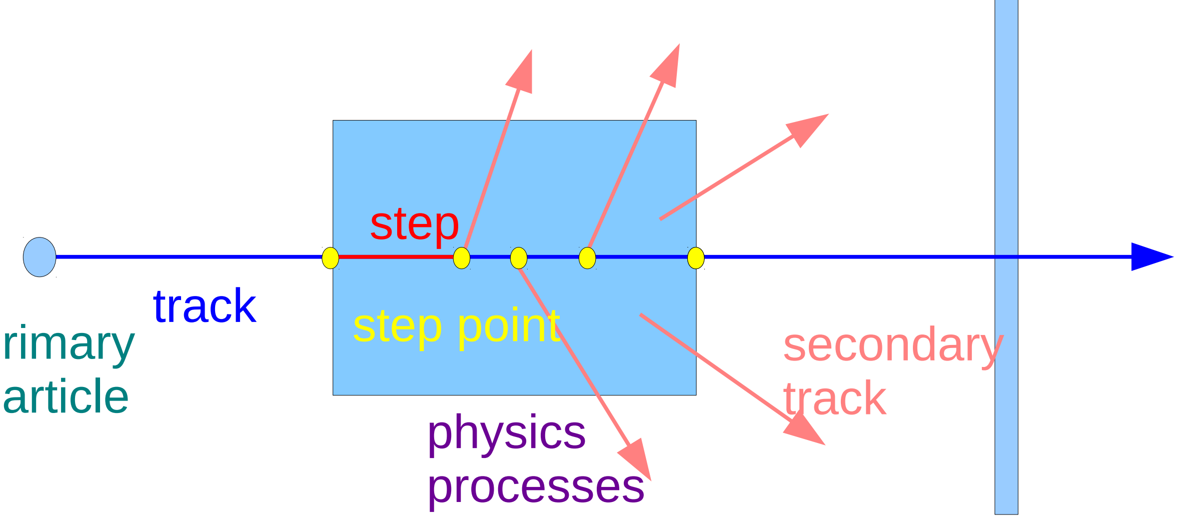
Track 2

Track 3

...

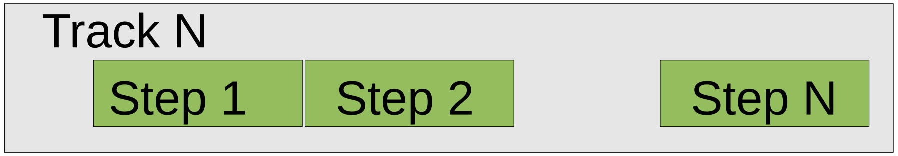
Track N

Track in Geant4

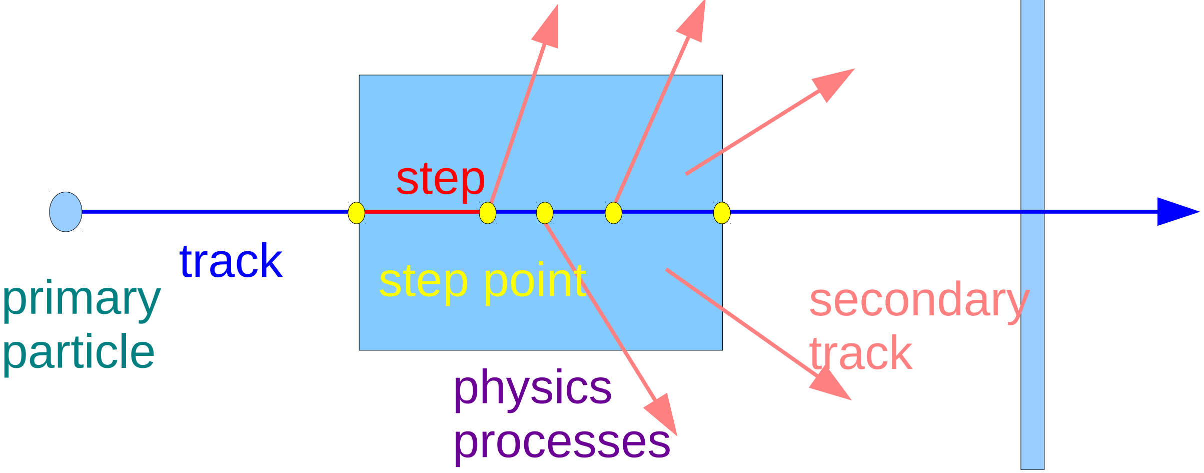


- Track is a snapshot of a particle.
  - It has physical quantities of the current instance only. It does not record previous quantities.
  - Step is a "delta" information to a track. Track is not a collection of steps. Instead, a track is being updated by steps.
- No track object persists at the end of event.
  - For the record of tracks, use trajectory class objects.

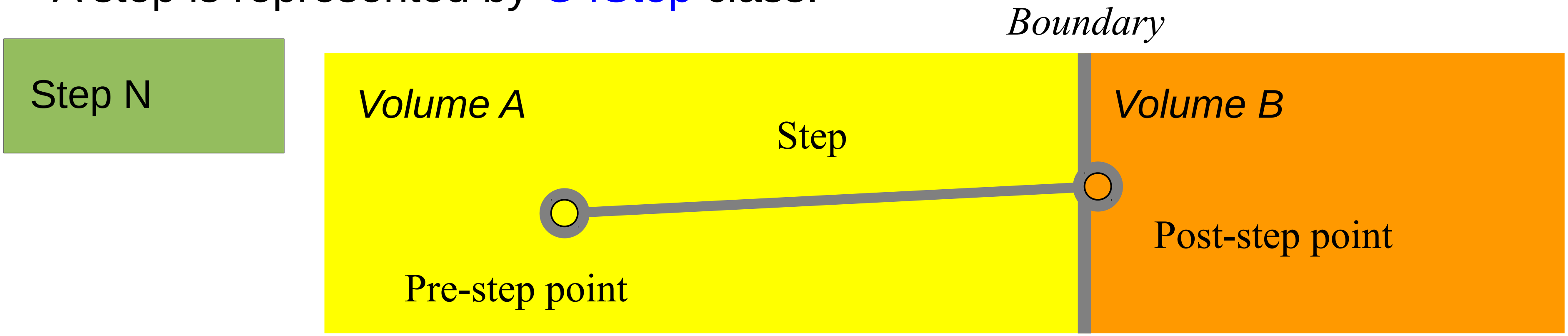
- G4TrackingManager manages processing a track
- A track is represented by G4Track class.



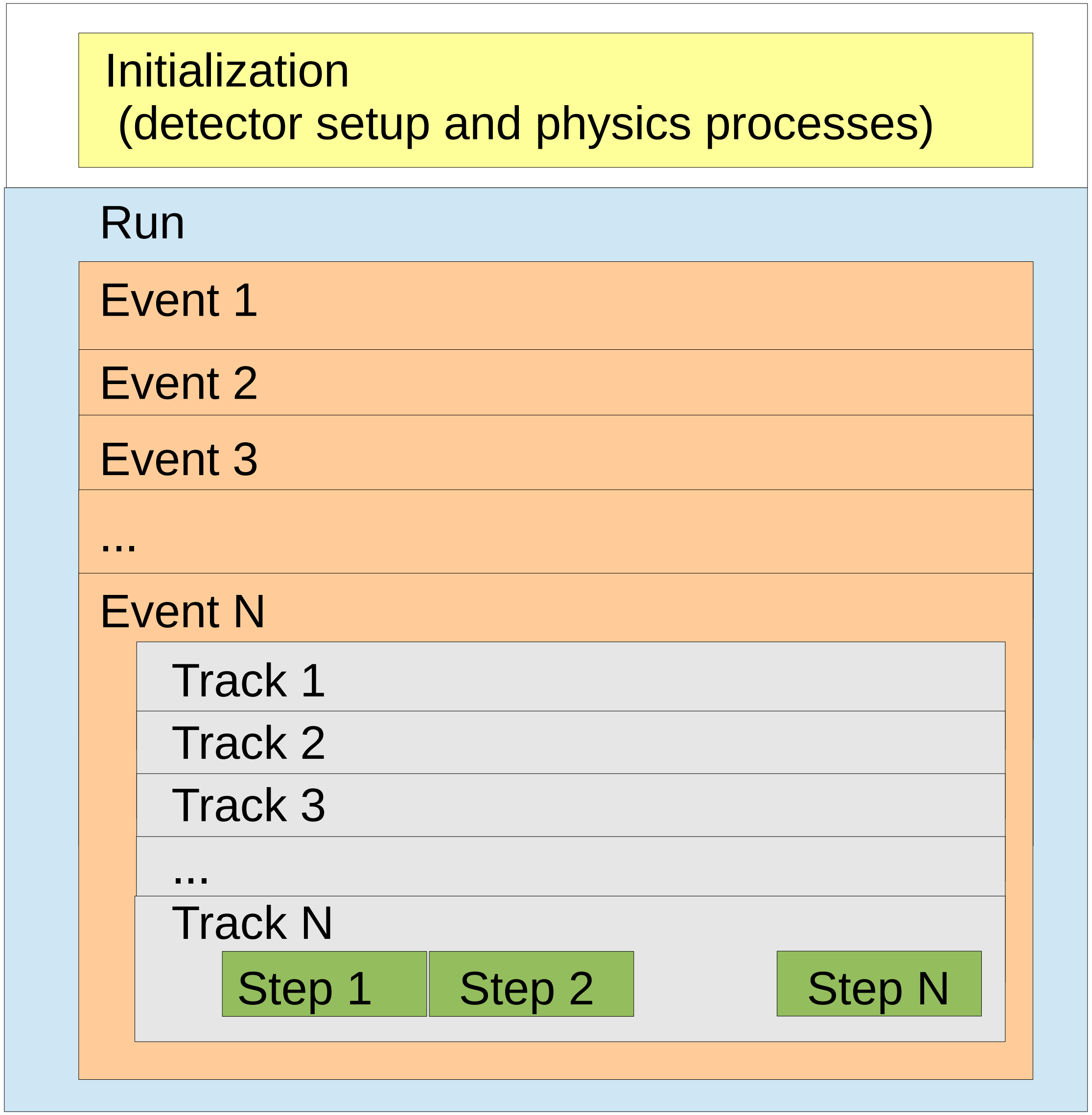
Step in Geant4



- 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.



Geant4 Run



Initialization
(detector setup and physics processes)

Run

Event 1

Event 2

Event 3

...

Event N

Track 1

Track 2

Track 3

...

Track N

Step 1 Step 2 Step N

Geant4 Loops

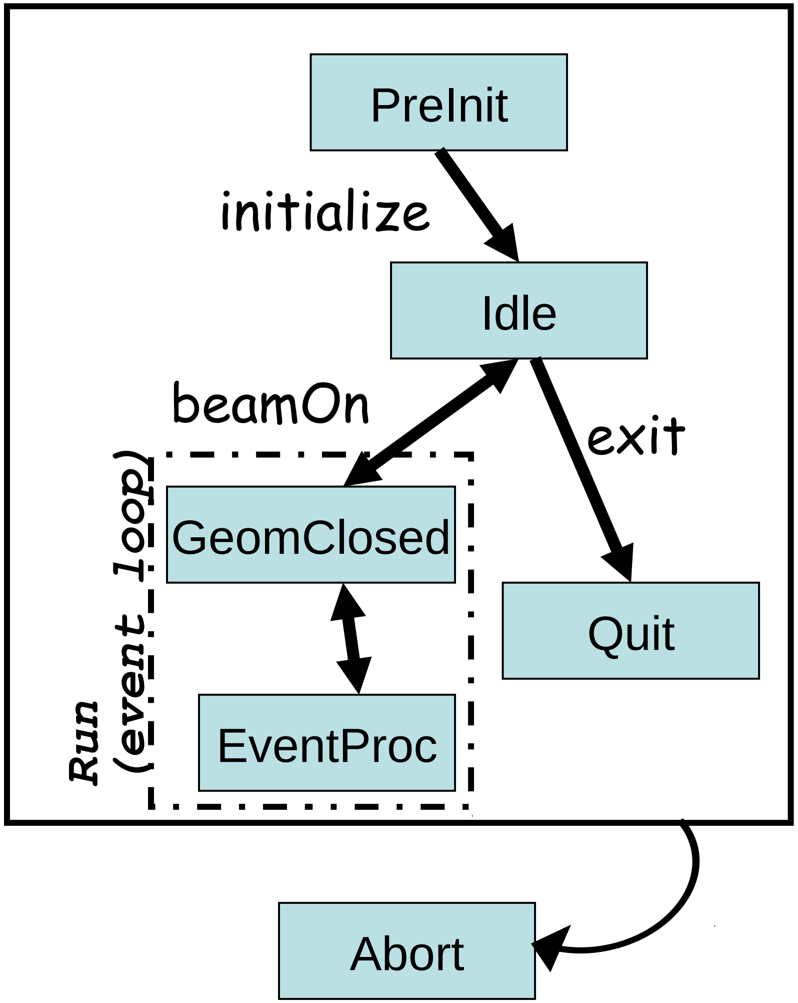
Geant4 works as a set of nested loops:

- **A job**
  - Geant4 kernel initialization;
  - Then one or several runs are launched;
  - **A run (G4Run):**
    - Physics and detector construction;
    - Then loop on events;
    - **An Event (G4Event):**
      - Generation of primary particles;
      - Then loop for tracking of these particles and all subsequent secondary particles
      - **A particle tracking (G4Track)**:
        - Loop on steps, propagating a particle object, up to the point this particle “dies or leave the detector “world”
        - **A step (G4Step))**
          - Loop on physics processes that apply to the current track to apply physics interactions,
          - generate secondary particles, compute energy deposit in the step, etc.;

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:
  - Ready to start a run
- G4State_GeomClosed

  Geometry is optimized and 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



User Application Classes

User Application (1)

- Geant4 is a toolkit. You have to build an application.
- You have to define:
  - Your geometrical setup (materials, volumes)
  - Physics to get involved (particles, physics processes/models), production thresholds
  - 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

User Application (2)

This is done in user application built in means of

- **main() program**

- **User initialization classes (mandatory**) derived from Geant4 base classes:

Detector	G4VUserDetectorConstruction
Primary generator	G4VPrimaryGeneratorAction,
Physics	G4VUserPhysicsList

- **User action classes (optional)** derived from

Run action	G4UserRunAction
Event action	G4UserEventAction
Tracking action	G4UserTrackingAction
Stepping action	G4UserSteppingAction
Stacking action	G4UserStackingAction

Describe Your Detector

- To describe your detector you have to derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- Implement the virtual method Construct(), where you
  - Instantiate all necessary materials
  - Instantiate volumes of your detector geometry
- Optionally, implement the virtual method ConstructSDandField(), where you
  - Instantiate your sensitive detector classes and set them to the corresponding logical volumes
  - Instantiate magnetic (or other) field
- Optionally you can define
  - Regions for any part of your detector
  - Visualization attributes (color, visibility, etc.) of your detector elements

Generate Primary Event

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Implement GeneratePrimaries(G4Event*) virtual function
  - 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

Select Physics Processes

- Geant4 does not have any default particles or processes however it provides a rich set of the physics lists for various use-cases
  - You can just instantiate the most suitable one for your application
- If none of these lists suites your needs you can cook your own one
  - Derive your own concrete class from G4VUserPhysicsList abstract base class.
  - Define all necessary particles in ConstructParticle() virtual function
  - Define all necessary processes and assign them to proper particles in ConstructProcess() virtual function
    - Even for the particle transportation, you have to define it explicitly.
  - Define cut-off ranges applied to the world (and each region)
  - Geant4 provides lots of utility classes/methods and examples.

Optional User Action Classes

- Optionally, you can implement some/all user action classes, methods of which are invoked during event processing at the beginning and the end of each run, event, track and step:
  - G4UserSteppingAction, G4UserTrackingAction, G4UserEventAction, G4UserRunAction, G4UserStackingAction
- A stacking action class provides a possibility to customize the default Geant4 stacking mechanism
- The action classes must be constructed and set to G4RunManager in UserActionInitialization
- The action classes methods are then called by Geant4 kernel in an appropriate phase of event processing

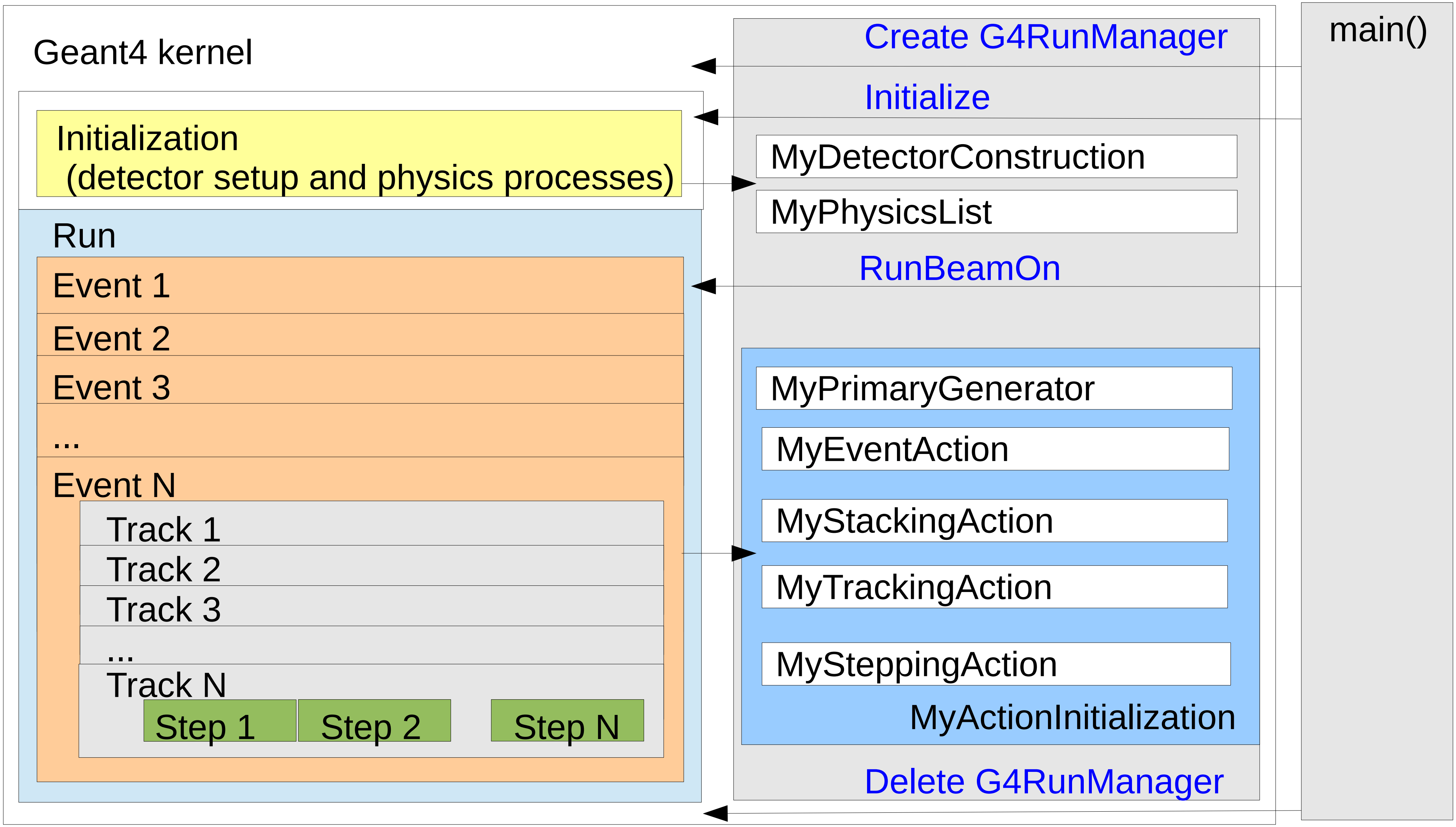
User Action Initialization

- The user initialization and action classes which are **called during event processing** can be defined all together in the user action initialization class derived from G4VUserActionInitialization abstract base class.
  - Note that use of this class is mandatory for multithreading processing
- Implement the virtual method Build(), where you
  - Instantiate all initialization and action classes called during event processing

main()

- Geant4 does not provide main().
- In your main(), you have to
  - Construct G4RunManager or its derived class (yours, MT)
  - Define your initialization classes: MyDetectorConstruction and MyPhysicsList and set them to G4RunManager
  - Define your primary generator class (MyPrimaryGenerator) using your MyActionInitialization class and set it to G4RunManager
- You can also
  - Define optional user action classes and set them to G4RunManager
  - Define Geant4 visualization and (G)UI session via G4VisExecutive and G4UIExecutive and/or your persistency manager
    - This part will be explained in the lectures on Visualization/UI

User Application and Geant4 Kernel



Geant4 kernel

Initialization
(detector setup and physics processes)

Run

Event 1

Event 2

Event 3

...

Event N

Track 1

Track 2

Track 3

...

Track N

Step 1

Step 2

Step N

main()

Create G4RunManager

Initialize

MyDetectorConstruction

MyPhysicsList

RunBeamOn

MyPrimaryGenerator

MyEventAction

MyStackingAction

MyTrackingAction

MySteppingAction

MyActionInitialization

Delete G4RunManager

exampleED.cc

Example of main() - part 1

```
#include "EDDetectorConstruction.hh"
#include "EDActionInitialization.hh"

#include "G4RunManager.hh"
#include "FTFP_BERT.hh"

int main(int argc,char** argv)
{
  // Create User Interface and enter in interactive session (1)

  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;

  // Detector construction
  runManager->SetUserInitialization(new EDDetectorConstruction());

  // Physics list
  G4VModularPhysicsList* physicsList = new FTFP_BERT;
  runManager->SetUserInitialization(physicsList);

  // User action initialization
  runManager->SetUserInitialization(new EDActionInitialization());

  // Create User Interface and enter in interactive session (2)
}
```

Action Initialization

EDActionInitialization.hh

```
#include "G4VUserActionInitialization.hh"

/// Action initialization class.
class EDActionInitialization : public G4VUserActionInitialization
{
  public:
    EDActionInitialization();
    virtual ~EDActionInitialization();

    virtual void Build() const;
};
```

EDActionInitialization.cc

```
#include "EDActionInitialization.hh"
#include "EDPrimaryGeneratorAction.hh"
#include "EDEventAction.hh"

EDActionInitialization::EDActionInitialization()
 : G4VUserActionInitialization()
{}

void EDActionInitialization::Build() const
{
  SetUserAction(new EDPrimaryGeneratorAction);
  SetUserAction(new EDEventAction);
}
```

Summary

- Geant4 kernel (“bricks”);
  - Manager classes: taking care of each steering run and each phase of event loop, G4RunManager as the top conductor
  - Classes to hold the information during event procession: G4Run, G4Event, G4Track and G4Step
  - Geant4 performs in six application states
- User application ('marvel”)
  - Users have to define their application writing their application program consisting of a main() function and their application classes derived from Geant4 base classes

In Next Lectures

- Define material and geometry
  - Geometry lectures
- Define the way of primary particle generation
  - Primary particles lecture
- Select appropriate particles and processes and define production threshold(s)
  - Physics lectures
- Define the way to extract useful information from Geant4
  - Scoring lectures