Kernel

I. Hrivnacova, IJCLab

Credits: M. Asai, SLAC, J. Apostolakis, CERN and others

Geant4 IN2P3 and ED PHENIICS Tutorial,
16 – 20 May 2022, IJCLab

Outline

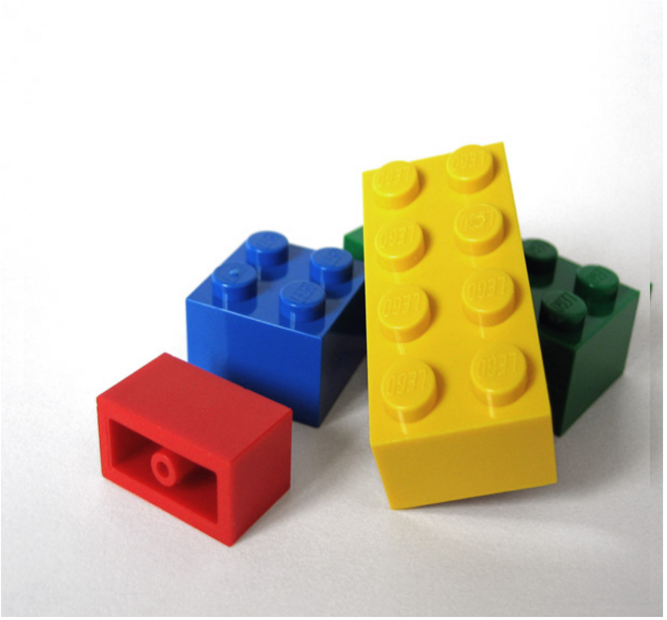
- How does it work ?
- Geant4 kernel classes
  - Run, event, track, step, classes to define particle
  - Tracking and processes
  - Application states
- User application classes

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

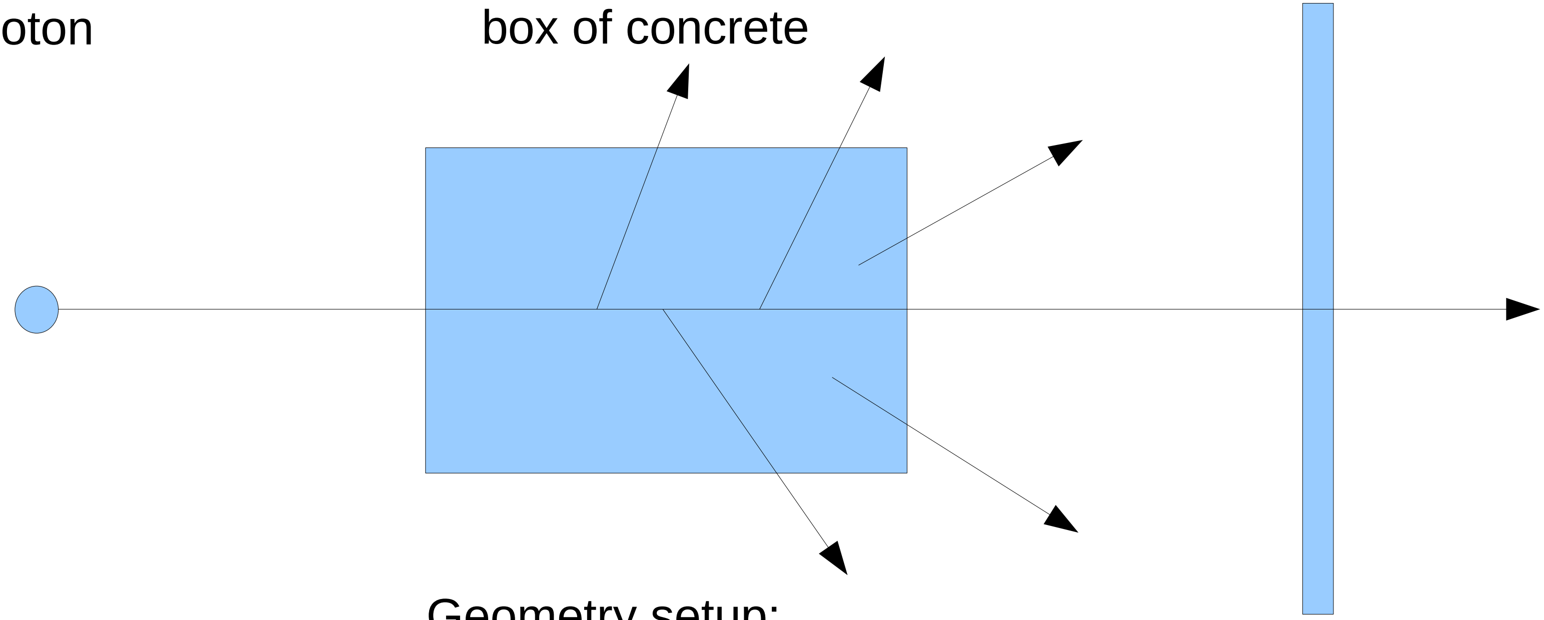
Geant4 and User Application (3)

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

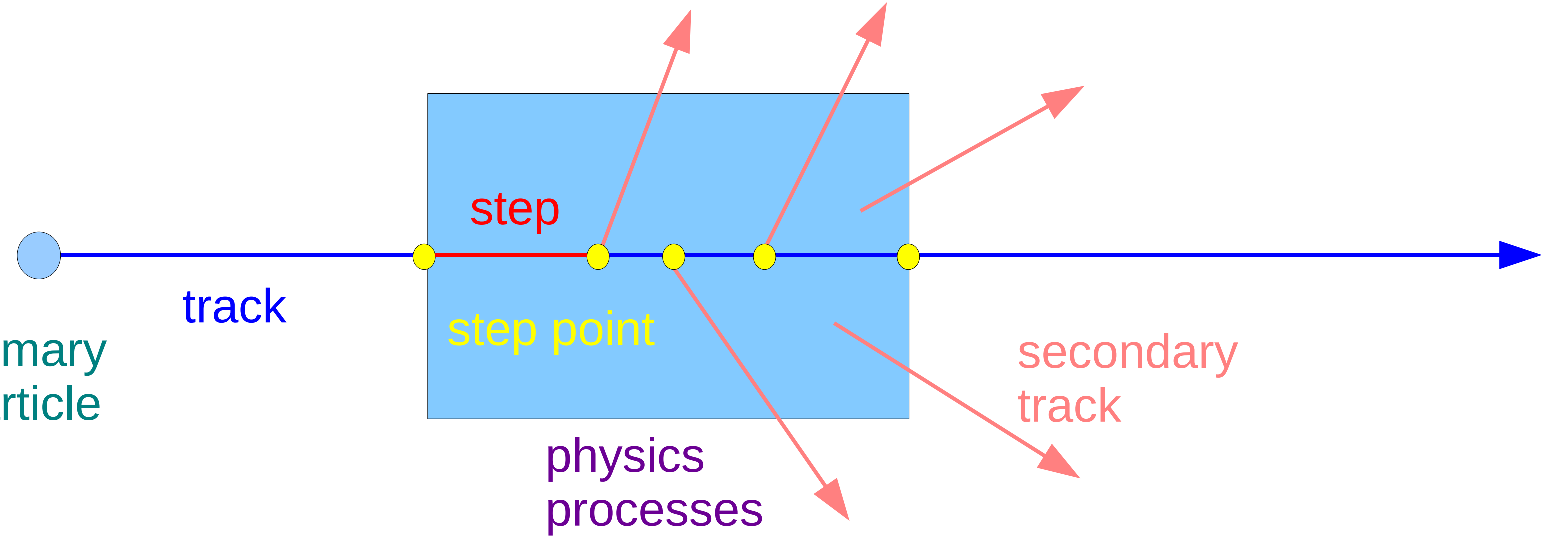
Geant4 and User Application (4)

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

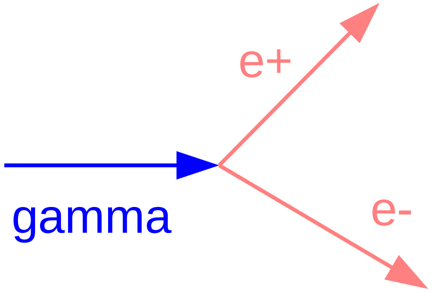
Geant4 then tracks the defined primary particles and let them interact with the materials present in geometry

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

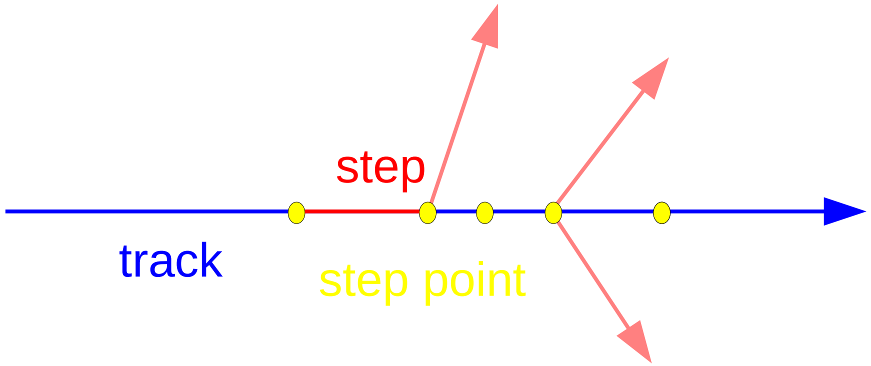
Track in Geant4



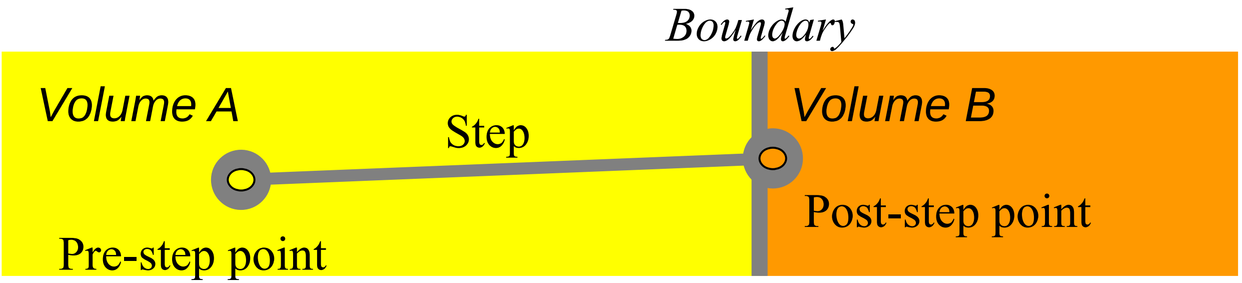
- Track is a snapshot of a particle.
  - It's physical quantities (i.e. energy,momentum, position ,time) represent the current 'instant' in the simulation. It does not record previous quantities.
  - Step is a "delta" information to a track. Track is not a collection of steps. Instead, a track is updated in a series steps.
- Classes:
  - G4TrackingManager manages processing a track
  - G4Track – represent a track.

- Each Track object **disappears** (is deleted) when it either
  - leaves the outermost ('world') volume,
  - disappears in an interaction (e.g. by decay or inelastic scattering),
  - it's kinetic energy becomes zero and it has no "AtRest" process, or
  - the user decides to kill it ('artificially').
- All tracks disappear. **None persist** at the end of event.
  - To record tracks, you must use objects of a trajectory class.

Step in Geant4



- **Step** has the **two points** and represents the “delta” information of a particle (energy loss over the step, time-of-flight during by the step, etc.).
- During simulation **Point** knows the volume(s) in which it belongs (& its material)
- If 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 such a **Step** knows materials of two volumes, boundary processes (such as light reflections or refractions) can be simulated.
- Classes: G4SteppingManager, G4Step, G4StepPoint



Event in Geant4

- **Event i**s the basic unit of simulation in Geant4.
- At its beginning primary tracks are generated (and pushed onto a stack).
- One ‘track’ at a time is popped from the stack and it is “tracked”
  - Any resulting secondary tracks are pushed back onto the stack.
  - This “tracking” lasts as long as the stack has a track.
- When the stack becomes empty, it’s the end of processing that event
- Classes:
  - An object of G4Event class represents an event. After its processing it contains few objects:
    - List of primary vertexes and particles (its input)
    - Hits and Trajectory collections (its output)
  - The G4EventManager class coordinates the processing of an event

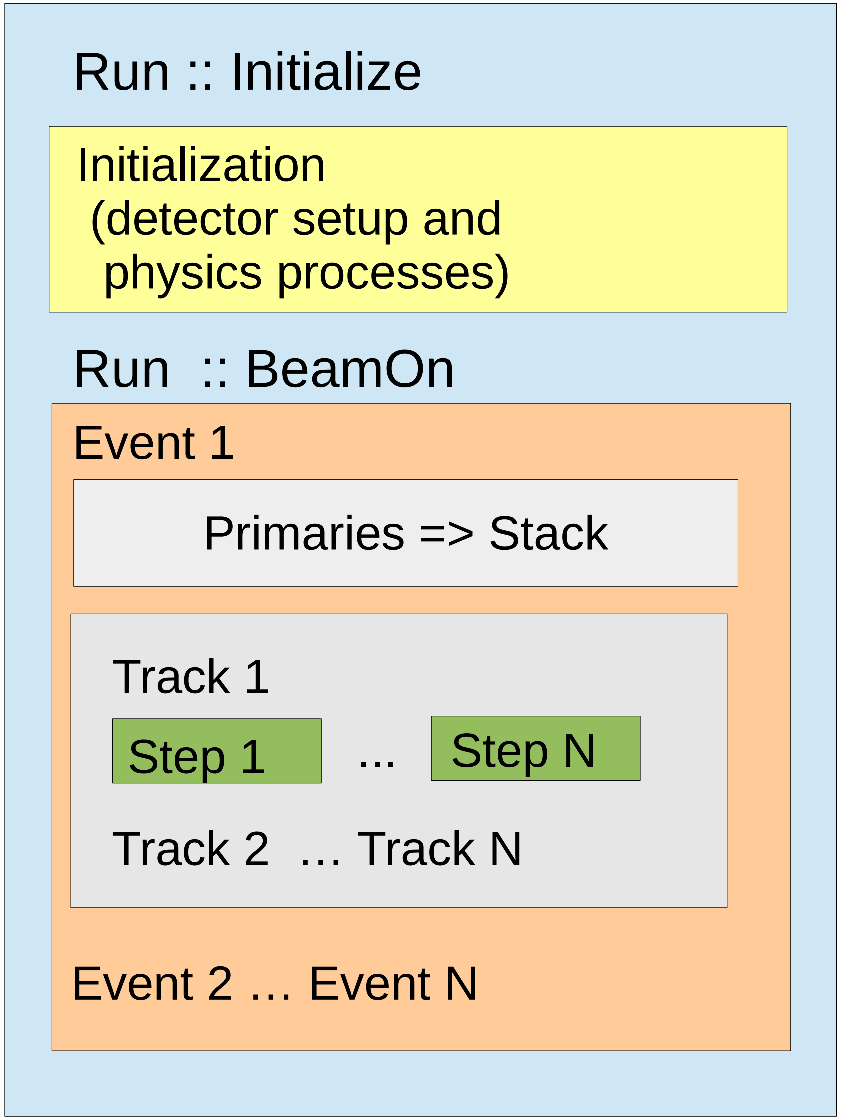
Run in Geant4

- **Run** consists of a configuration and a set of events
- By definition before starting a run, the user must already define the
  - detector setup, source and settings of physics processes
  - and you *must not change these* until the run has ended.

Classes:

- **Run** is represented by a G4Run object (or a user-defined class derived from G4Run.)
  - In analogy with experiments, you start a simulation by calling G4Run "Beam On".
  - Typically a run consists of one event loop. (Events are treated one after another.)
  - At the start of a run the geometry structures and physics configurations are prepared
    - the geometry is optimized for navigation,
    - cross-section tables are calculated for the setup's materials.
- The G4RunManager class organizes a run,
  - You will interact with G4RunManager to give it your setup, source, …

Geant4 Loops



A simulation job starts with 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.;

Particle in Geant4

- A particle in Geant4 is represented by three layers of classes.
- G4Track
  - Position, geometrical information, etc.
  - This is a class representing a particle being tracked.
- G4DynamicParticle
  - "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
  - Each G4Track object has its own and unique G4DynamicParticle object.
  - Each object of class represents an individual particle (i.e. one electron.)
- G4ParticleDefinition
  - "Static" properties of a particle, such as charge, mass, life time, decay channels, etc.
  - The list of processes involving to the particle
- All G4DynamicParticle objects of same kind of particle share the same G4ParticleDefinition.

Tracking and Processes

- The Geant4 tracking ‘loop’ is general.
  - It is independent of the particle type,
  - It obtains the list the applicable physics processes from each particle (type)
  - It gives the chance to each process in turn:
    - 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)
- This generality has strengths (adaptability) and costs (performance.)

G4cout, G4cerr

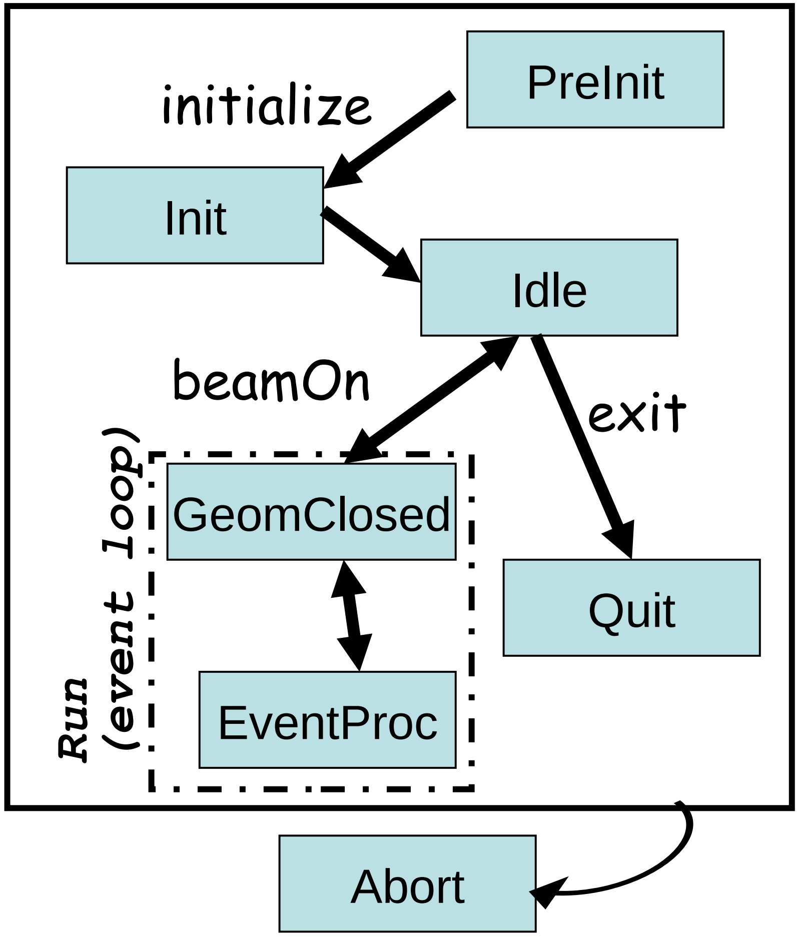
- G4cout and G4cerr are `ostream` objects defined by Geant4.
  - G4endl is also provided.

```
G4cout << ”Hello Geant4!” << G4endl;
```

- Some GUIs buffer these output streams to display print-out in another window or provide storing / editing functionality.
  - The user is asked to avoid using `std::cout` and `std::cerr`.
- We recommend also that the user also avoids using the ‘raw’ `std::cin` for input.
  - Instead we suggest to use the G4 user-defined commands which tie into the Geant4 User Interface system (provided by the intercoms category).
- You can use ‘ordinary’ file I/O – Geant4 will not interfere with it.

Geant4 as a State Machine

- Geant4 has 7 application states
  - Some methods in Geant4 are available for only a certain state(s)



- G4State_PreInit
  - Initial condition
- G4State_Init
  - During initialization
- 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

- 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

- Geant4 does not provide a main().
- In your main(), you have to
  - Construct G4RunManager
  - Set user mandatory initialization classes to RunManager
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
    - G4VUserActionInitialization
- You can define VisManager, (G)UI session, optional user action classes, and/or your persistency manager in your main().

Overview of User Classes

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

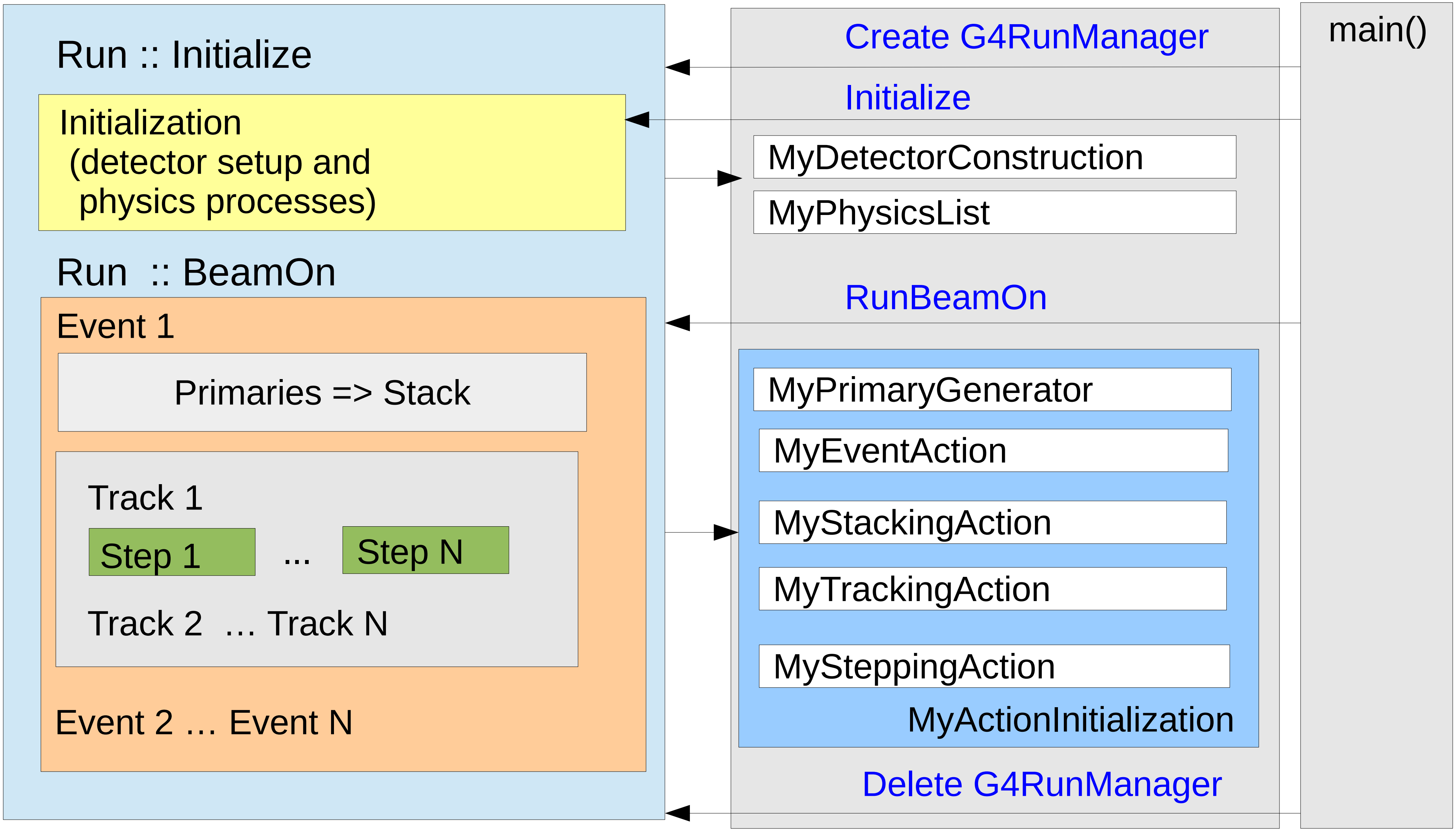
Detector	G4VUserDetectorConstruction
Primary generator	G4VPrimaryGeneratorAction,
Physics	G4VUserPhysicsList

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

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

The action classes methods are then called by Geant4 kernel in an appropriate phase of event processing

Geant4 Kernel & User Application



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 the 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()
  - C++: the function main is called at the program startup, leaving main() ends the program
- 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

exampleED.cc

Example of main() - part 1

```
#include "DetectorConstruction.hh"
#include "ActionInitialization.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 ED::DetectorConstruction());

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

  // User action initialization
  runManager->SetUserInitialization(new ED::ActionInitialization());

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

ActionInitialization.hh

```
#include "G4VUserActionInitialization.hh"

namespace ED
{
class ActionInitialization : public G4VUserActionInitialization
{
  public:
    ActionInitialization();
    virtual ~ActionInitialization();

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

ActionInitialization.cc

```
#include "ActionInitialization.hh"
#include "PrimaryGeneratorAction.hh"
#include "EventAction.hh"

namespace ED
{
ActionInitialization::ActionInitialization()
{}

void ActionInitialization::Build() const
{
  SetUserAction(new PrimaryGeneratorAction);
  SetUserAction(new EventAction);
}
}
```

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