



Geant4 general concept




Varisano Annagrazia

LNS-INFN

Geant4 simulation code: theory and practical session

X Seminar on Software for Nuclear, Subnuclear and Applied Physics

Basic concepts and capabilities of Geant4

- C++ language
- Object Oriented
- Open Source
- Released twice per year
- Supported and tested in different platforms:   
- It is a toolkit, i.e. a **collection of tools** the User can use for his/her simulation
- It does not provide a general setup configurations:
 - You **must** provide the **necessary information** to configure your simulation
 - You must choose the **Geant4 tools** to use

Main Geant4 capabilities

- Transportation of a particle 'step-by-step' taking into account all the possible interactions with materials and fields
- The transport ends if the particle
 - reaches zero kinetic energy
 - disappears in some interaction
 - reaches the end of the simulation volume
- Geant4 allows the User to access the transportation process to retrieve the simulation results (USER ACTIONS)
 - at the beginning and end of the transport
 - at the end of each step in transportation
 - if particle reaches a sensitive detector;
 - others

Geant4: main steps

What you MUST do:

- Describe your experimental set-up
- Provide the primary particles input to your simulation
- Decide which particles and physics models you want to use out of those available in Geant4 and the precision of your simulation (cuts to produce secondary particles)

Files composing a Geant4 app

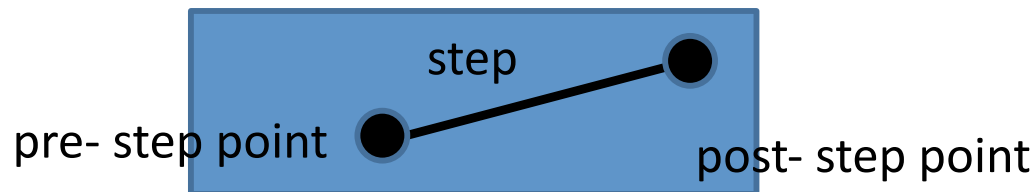
- Main() file
- Sources files (*.cc)
 - usually included in the /src folder
- Header files (*.hh)
 - usually included in the /include folder
- Three important files are mandatory(with the Main.cc)
 - The PrimarygeneratorAction (.cc and .hh)
 - The DetectorConstruction (.cc and .hh)
 - The PhysicsList (.cc and .hh)

Terminology (jargons)

- **Step** : has two points *pre-step point* and *post-step point*, and it gives us 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.

G4SteppingManager class manages processing a step, a step is represented by **G4Step** class.

G4UserSteppingAction is the optional user hook.



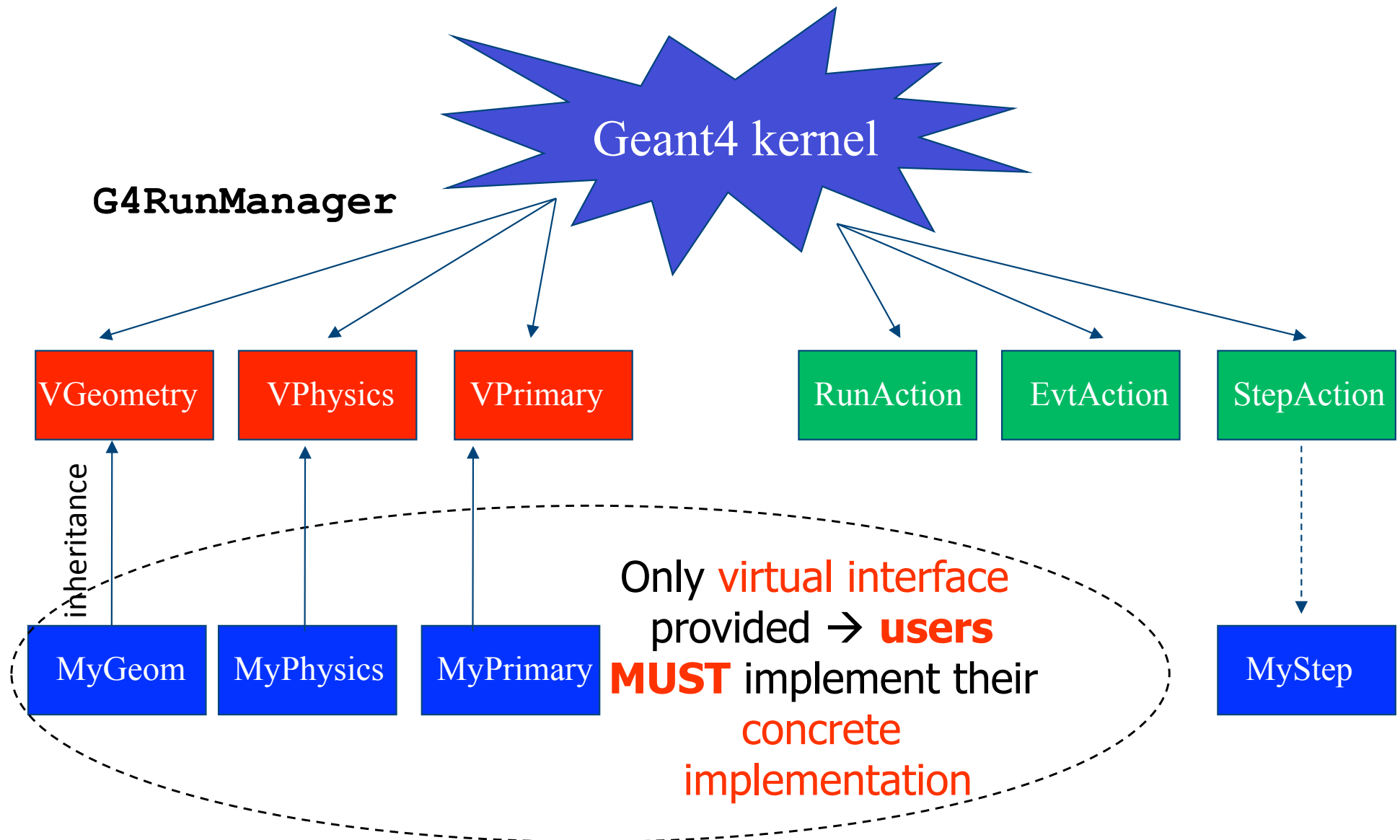
Terminology (jargons)

- **Event** : At beginning of processing, an event contains primary particles. These primaries and secondary produced 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

Terminology (jargons)

- **Run:** Conceptually, a run is a collection of events which share the same detector conditions.
- 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

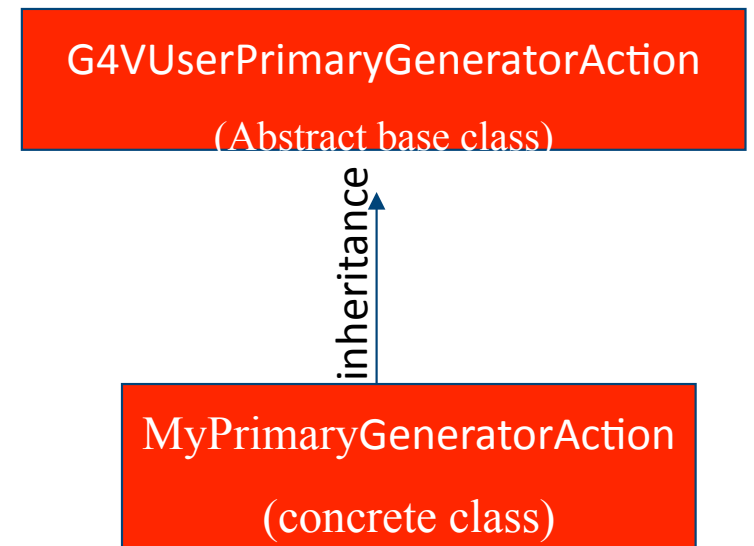
Interaction with the Geant4 kernel - 1



Interaction with the Geant4 kernel - 2

Two types of Geant4 classes:

- **Abstract base classes** (classes starting with G4V)
 - User derived concrete classes are **mandatory**
 - User to implement the purely virtual methods
- **User Hook classes** for user interaction
 - User derived classes are **optional**



User Classes

Initialisation classes

Invoked at the initialization

G4RunManager::SetUserInitialization()

- G4VUserDetectorConstruction
- G4VUserPhysicsList

Classes having name starting with **G4V** are **abstract classes** (containing purely virtual methods)

Mandatory classes in ANY Geant4 User Application

- G4VUserDetectorConstruction
describe the experimental set-up
- G4VUserPhysicsList
select the physics you want to activate
- G4VUserPrimaryGeneratorAction
generate primary events

Action classes

Invoked during the execution loop

G4RunManager::SetUserAction()

- G4VUserPrimaryGeneratorAction
- G4UserRunAction
- G4UserEventAction
- G4UserTrackingAction
- G4UserStackingAction
- G4UserSteppingAction

The main() program

- Geant4 does **not** provide the **main()**
 - Geant4 is a toolkit!
 - The **main()** is part of the **user application**
- In his/her **main()**, the user **must**
 - construct **G4RunManager** (or his/her own derived class)
 - **notify** the **G4RunManager** **mandatory user classes** derived from
 - **G4VUserDetectorConstruction**
 - **G4VUserPhysicsList**
 - **G4VUserPrimaryGeneratorAction**
- in his/her **main()**, the user **may define** optional user action classes
- User mustn't forget to **delete** the **G4RunManager** at the end
- The user also has to take care of **retrieving** and **saving** the relevant **information** from the simulation (Geant4 will not do that by default)

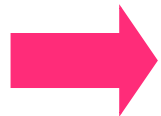
An example of main()

```
{  
    ...  
    // Construct the default run manager  
    G4RunManager* runManager = new G4RunManager;  
    // Set mandatory user initialization classes  
    MyDetectorConstruction* detector = new MyDetectorConstruction;  
    runManager->SetUserInitialization(detector);  
    MyPhysicsList* physicsList = new MyPhysicsList;  
    runManager->SetUserInitialization(myPhysicsList);  
    // Set mandatory user action classes  
    runManager->SetUserAction(new MyPrimaryGeneratorAction);  
    // Set optional user action classes  
    MyEventAction* eventAction = new MyEventAction();  
    runManager->SetUserAction(eventAction);  
    MyRunAction* runAction = new MyRunAction();  
    runManager->SetUserAction(runAction);  
    ...  
}
```

Select physics processes

- Geant4 **doesn't have** any **default** particles or processes
- Derive your own **concrete** class from the **G4VUserPhysicsList** abstract base class
 - define all necessary **particles**
 - define all necessary **processes** and assign them to proper particles
 - define γ/δ production **thresholds** (in terms of range)
- **Pure virtual** methods of **G4VUserPhysicsList**

ConstructParticles()
ConstructProcesses()
SetCuts()



must be implemented by the user in his/her concrete derived class

Optional user classes

- Five concrete base classes whose virtual member functions the user may override to gain control of the simulation at various stages

- G4UserRunAction
- G4UserEventAction
- G4UserTrackingAction
- G4UserStackingAction
- G4UserSteppingAction



e.g. actions to be done
at the beginning and
end of each event

- Each member function of the base classes has a dummy implementation (**not** purely virtual)
 - Empty implementation: does nothing
- Objects of user action classes must be **registered** with **G4RunManager**
`runManager->SetUserAction(new MyEventActionClass`

Methods of user classes - 1

G4UserRunAction

- `BeginOfRunAction(const G4Run*)` // book histos
- `EndOfRunAction(const G4Run*)` //store histos

G4UserEventAction

- `BeginOfEventAction(const G4Event*)` //initialize event
- `EndOfEventAction (const G4Event*)` // analyze event

G4UserTrackingAction

- `PreUserTrackingAction(const G4Track*)`
//decide to store/not store a given track
- `PostUserTrackingAction(const G4Track*)`

Methods of user classes - 2

G4UserSteppingAction

- `UserSteppingAction(const G4Step*)`

//kill, suspend, postpone the track, draw the step, ...

G4UserStackingAction

- `PrepareNewEvent()` *//reset priority control*

- `ClassifyNewTrack(const G4Track*)`

// Invoked when a new track is registered (e.g. kill, postpone)

- `NewStage()`

// Invoked when the Urgent stack becomes empty (re-classify, abort event)

Optional: select (G)UI

- In your **main()**, taking into account your computer environment, **instantiate a G4UIsession concrete/derived class** provided by Geant4 and invoke its **SessionStart()** method

```
mysession->SessionStart();
```

Optional: select visualization

- In your **main()**, taking into account your computer environment, **instantiate a G4VisExecutive** and invoke its **Initialize()** method
- Geant4 provides interfaces to various graphics drivers:
 - DAWN (*Fukui renderer*)
 - WIRED
 - RayTracer (*ray tracing by Geant4 tracking*)
 - OpenGL
 - OpenInventor
 - VRML
 - ...

General recipe for novice users

Experienced users may do much more, but the conceptual process is still the same...

- **Design** your application... requires some preliminar **thinking** (what is it supposed to do?)
- Create your **derived mandatory** user classes
 - MyDetectorConstruction
 - MyPhysicsList
 - MyPrimaryGeneratorAction
- Create **optionally** your **derived user action** classes
 - MyUserRunAction, MyUserEventAction, ...
- Create your **main()**
 - Instantiate **G4RunManager** or your own derived MyRunManager
 - **Notify** the RunManager of your mandatory and optional user classes
 - Optionally initialize your favourite User Interface and Visualization
- That's all!