Physics II : processes

Paris Geant4 Tutorial

5 June 2007

Marc Verderi
Ecole Polytechnique - LLR

Introduction

- All processes in Geant4 derived from the same abstract interface: **`G4VProcess`**
- If you don’t have to setup physics lists, this tutorial is “cultural”.
- If you have to build physics a few points are ***critical***

- Present here the **`G4VProcess`** abstract interface
  - and the way processes are handled by the tracking

Contents

I. From **`G4Track`** to processes

II. The process interface

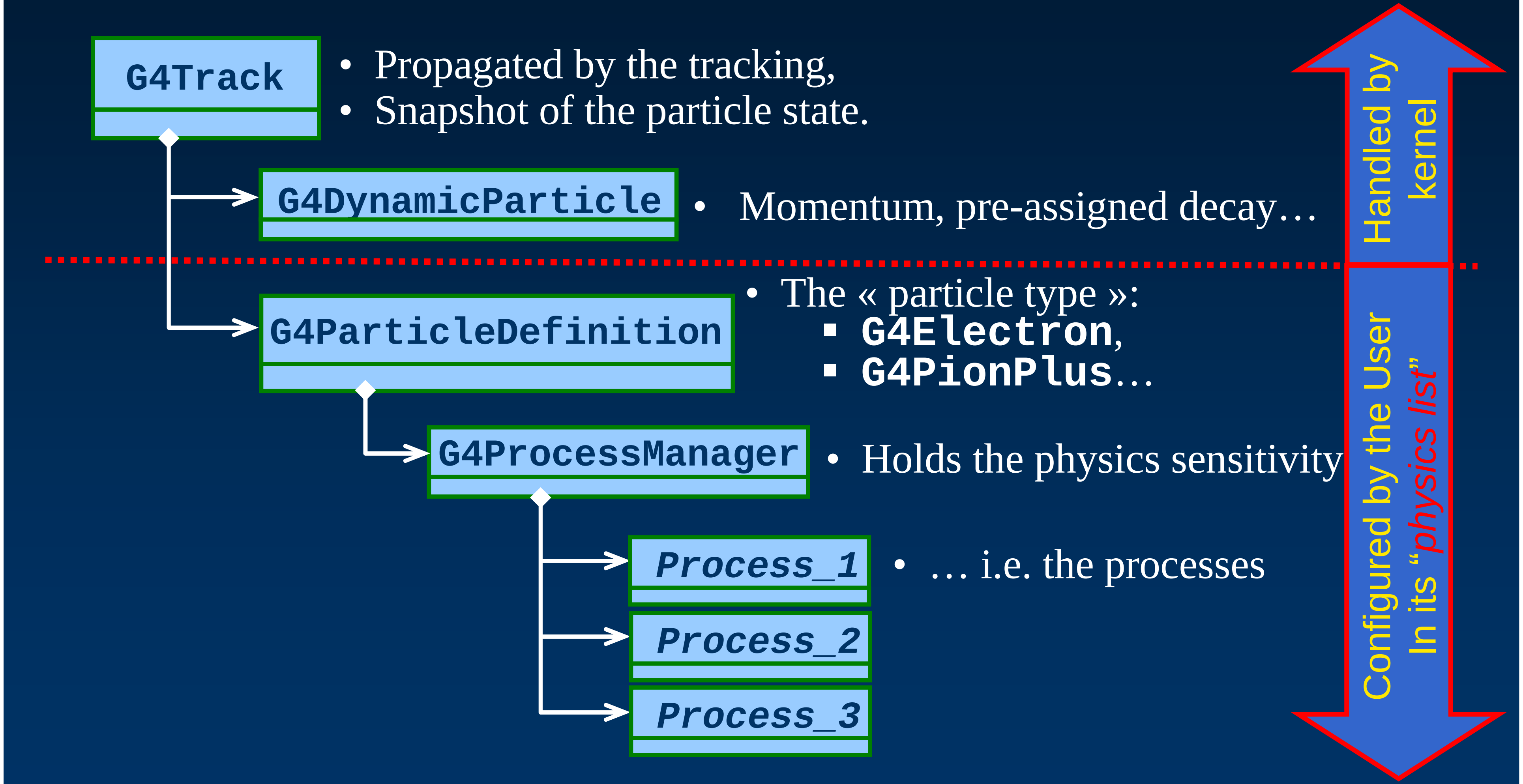
- **`G4VProcess`**
- How processes are used by the stepping

III. The production cuts

- Just illustration

I. From **`G4Track`** to processes

From **G4Track** to processes



G4Track

- Propagated by the tracking,
- Snapshot of the particle state.

G4DynamicParticle

- Momentum, pre-assigned decay…

G4ParticleDefinition

- The « particle type »:
  - **G4Electron**,
  - **G4PionPlus**…

G4ProcessManager

- Holds the physics sensitivity

Process_1

Process_2

Process_3

- … i.e. the processes

Handled by kernel

Configured by the User In its "*physics list*"

II. The process interface

Speak about:

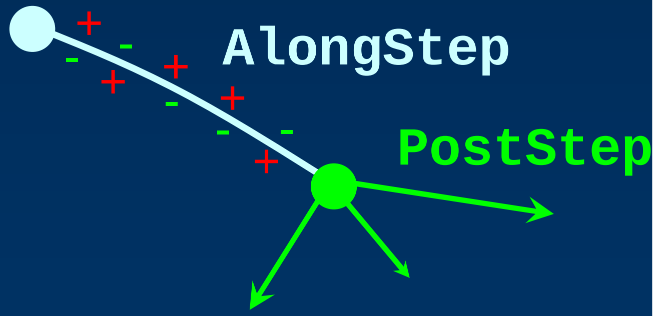
`G4VProcess`

The stepping

G4VProcess: 3 kind of actions (1)

- Abstract class defining the common interface of all processes in Geant4:
  - Used by all « physics » processes
  - but is also used by the transportation, etc…
  - Defined in **`source/processes/management`**
- Define three kinds of actions:
  - **`AtRest`** actions:

    - Decay, e$^+$ annihilation …
  - **`AlongStep`** actions:
    - To describe continuous (inter)actions, occuring along the path of the particle, like ionisation;



  - **`PostStep`** actions:
    - For describing point-like (inter)actions, like decay in flight, hard radiation…

`G4VProcess`: 3 kind of actions (2)

- A process can implement any combination of the three **`AtRest`**, **`AlongStep`** and **`PostStep`** actions:
  - eg: decay = **`AtRest`** + **`PostStep`**

- If you plan to implement your own process:
  - A set on intermediate classes exist implementing various combinations of actions:
    - For example:
      - **`G4VDiscreteProcess`**: only **`PostStep`** actions;
      - **`G4VContinuousDiscreteProcess`**: **`AlongStep + PostStep`** actions;
      - …

`G4VProcess`: action methods

- Each action defines two methods:
  - `GetPhysicalInteractionLength()`:
    - Used to ***limit the step***:
      - either because the process « triggers » an interaction, a decay
      - or any other reasons, like fraction of energy loss, geometry boundary, user's limit …
  - `DoIt()`:
    - Implements the ***actual action*** to be applied on the track;
    - And the related production of secondaries.

`G4VProcess` : actions summary

- The « action » methods are thus:
  - **`AtRestGetPhysicalInteractionLength(), AtRestDoIt();`**
  - **`AlongStepGetPhysicalInteractionLength(), AlongStepDoIt();`**
  - **`PostStepGetPhysicalInteractionLength(), PostStepDoIt();`**
- **`G4VProcess`** defines other methods:
  - **`G4bool IsApplicable(const G4ParticleDefinition &);`**
    - Used to check if a process can handle the given particle type
  - And methods called at the beginning and end of tracking of a particle, etc…

How the Stepping handles processes

- The stepping treats processes ***generically***:
  - The stepping does not know[*] what processes it is handling;
- The stepping makes the processes to:
  - Cooperate for **`AlongStep`** actions;
  - Compete for **`PostStep`** and **`AtRest`** actions;

[*] almost: some exception for transportation

- Particular treatments are also possible on process request, which can ask to be
  - **`forced`**:
    - **`PostStepDoIt()`** action applied anyway;
      - e.g. transportation to update **`G4Track`** geom. info
  - **`conditionallyForced`**:
    - **`PostStepDoIt()`** applied if **`AlongStep`** has limited the step;
  - etc …

Stepping Invokation Sequence of Processes for a particle travelling

1. At the beginning of the step, determine the step length:
   - Consider all processes attached to the current **`G4Track`**;
   - Define the step length as the smallest of the lengths among:
     - All **`AlongStepGetPhysicalInteractionLenght()`**
     - All **`PostStepGetPhysicalInteractionLength()`**
2. Apply ***all*** **`AlongStepDoIt()`** actions, « at once »:
   - Changes computed from particle state at the beginning of the step;
   - Accumulated in the **`G4Step`**;
   - Then applied to the **`G4Track`**, from the **`G4Step`**.
3. Apply **`PostStepDoIt()`** action(s) « sequentially », as long as the particle is alive:
   - Apply **`PostStepDoIt()`** of process which limited the step (if any);
   - And apply « forced » and « conditionnally forced » actions.

Stepping Invokation Sequence of Processes for a Particle at Rest

1. If the particle is at rest, *is stable and can't annihilate*, it is killed by the tracking:
   - More properly said: if a particle at rest has no « **`AtRest`** » actions defined, it is killed.
2. Otherwise determine the lifetime:
   - Take the smallest time among:
     - All **`AtRestGetPhysicalInteractionLenght()`**
       - Called « physical interation lenght » but returns a time;
3. Apply **`AtRestDoIt()`** action of process which returned the smallest time.

G4VProcess & G4ProcessManager

- **`G4ProcessManager`** maintains three vectors of actions:
  - One for the **`AtRest`** methods of the particle;
  - One for the **`AlongStep`** ones;
  - And one for the **`PostStep`** actions.
- These are these vectors the user sets up in the "physics list" and which are used by the tracking.
- Note that the process ordering provided by/to the **`G4ProcessManager`** vectors ***IS*** relevant.

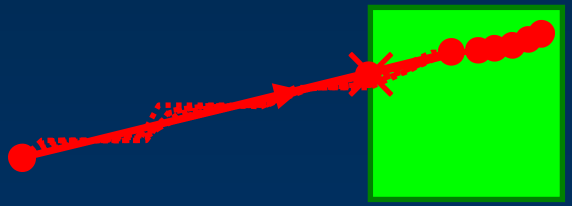
A word about processes ordering

- **The ordering of processes matters !**
- Ordering of following processes is critical for a few of them:
  - Assuming n processes, the ordering of the AlongGetPhysicalInteractionLength() of the last processes should be:

    [n-2] …

    [n-1] multiple scattering

    [n] transportation
- Why ?
  - Processes return a « true path length »;
  - The multiple scattering « virtually folds up » this true path length into a ***shorter*** « geometrical » path length;
  - Based on this new length, the transportation can geometrically limits the step.
- Other processes ordering usually does not matter.



- Show processes for:
  - Electron
  - Positron
  - Gamma
- Show physics list

III. The production cuts;

Makoto and Michel will give details later

Just illustration here

Conclusion/summary

- All processes share the same interface, **`G4VProcess`**:
  - This allows Geant4 to treat processes generically:
  - Three types of actions are defined:
    - **`AtRest`** (compete), **`AlongStep`** (cooperate), **`PostStep`** (compete)
    - Each action define a “**`GetPhysicalInterationLenght()`**” and a “**`DoIt()`**” method
- Processes are attached to the particle by its **`G4ProcessManager`**
  - This is the way the particle acquires its sensitivity to physics
  - This **`G4ProcessManager`** is set up in the “physics list”
    - Please be careful of the multiple scattering and transportation ordering
- Some processes require “cuts”, i.e. “production threshold”:
  - to be defined to absorb infrared divergences into a continuous energy loss contribution
  - That needs to be tuned by the user for its particular application
- One range cut can be specified per region