Particle definition, Process management

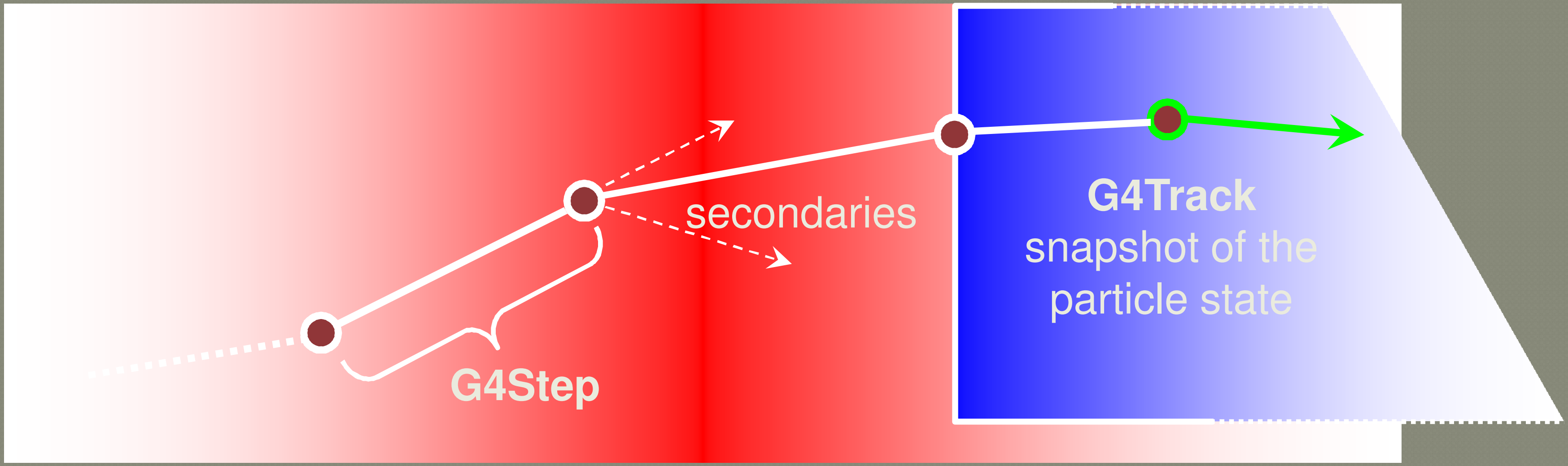
Geant4 tutorial
MC-PAD Network Training Event
28-30 January 2010
V. Ivanchenko
Adaptation of Marc Verderi original lecture

Geant4 interface to physics

- The **`G4ParticleDefinition`** interface
- The **`G4VProcess`** class process interface
- The **`G4ProcessManager`** class

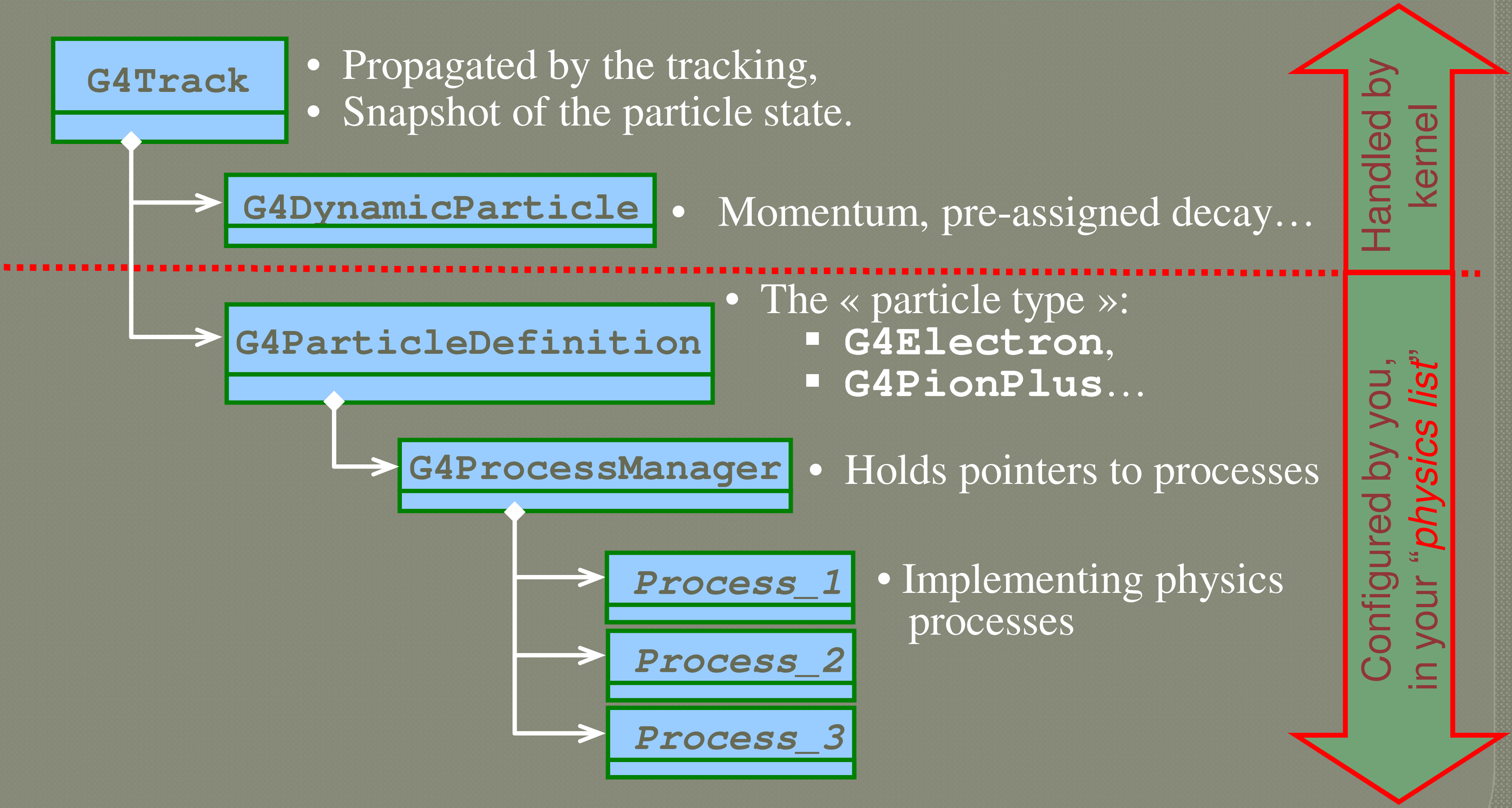
Geant4 tracking

- **G4Track** is the object "pushed" step by step by the tracking :



- Moving by one step is the responsibility of the "stepping"
  - Which is the core engine of the "tracking" machinery
- These moves/steps have to be physically meaningful
  - And the stepping invokes physics to realize them
- This physics is attached to the **G4Track**, let's see how.

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 pointers to processes

Process_1

Process_2

Process_3

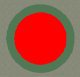
- Implementing physics processes

Handled by kernel

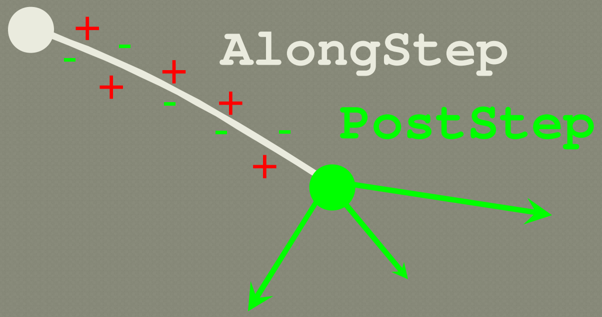
Configured by you, in your *“physics list”*

G4VProcess: 3 kind of actions

- Abstract class defining the common interface of all processes in Geant4:
  - Used by all processes
    - including transportation, etc…
  - Defined in `source/processes/management`
- Three kinds of actions:
  - **AtRest** actions:



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



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

G4VProcess : actions summary

- The virtual « action » methods are following:
  - `AtRestGetPhysicalInteractionLength(), AtRestDoIt();`
  - `AlongStepGetPhysicalInteractionLength(), AlongStepDoIt();`
  - `PostStepGetPhysicalInteractionLength(), PostStepDoIt();`

- Other important virtual method:
  - `G4bool IsApplicable(const G4ParticleDefinition &);`
    - Used to check if a process can handle the given particle type
    - It is called by the kernel when you set up your physics list

G4VProcess: extensions

- 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

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 you have to set up in your "physics list"
  - These vectors are used by the tracking.
- Note that the ordering of processes provided by/to the G4ProcessManager vectors is relevant and used by the stepping
  - There are few critical points you should be aware of
    - Multiple scattering can shift end point of a step and step length
    - Scintillation, Cerenkov and some other processes assuming that step and energy deposition at the step are defined

Adding a process in physics list

- Get the process manager of the particle:

```
G4ProcessManager* pmanager = particle->GetProcessManager();
```

- Add the process:

```
  pmanager->AddProcess(new G4eIonisation,         -1, 2, 2);
```

  - The indices provided are these of the ordering in the `DoIt()` vectors
  - Which is by default **reverse** of the ordering of the `GetPhysicalInteractionLength()` one ! ☹
    - Index in `AtRestDoIt()` vector
    - Index in `AlongStepDoIt()` vector
    - Index in `PostStepDoIt()` vector
  - There are more utility methods to add a process, but above one is probably the most clear

About process ordering

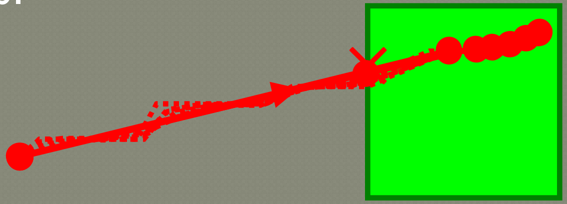
- **The most strong rule for multiple-scattering and transportation.**
- In your physics list, you should **always** have, for the ordering of the `AlongGetPhysicalInteractionLength(…)` methods:
  - – Transportation last
    - For all particles
  - – Multiple scattering second last
    - For charged particles only
      - assuming n processes

        [n-2] …

        [n-1] multiple scattering

        [n] transportation

- Why ?
  - Processes return a « true path length »;
  - The multiple scattering folds up this length into a ***shorter*** « geometrical » path length;
  - Based on this new length, the transportation can geometrically limits the step.



Displaying processes and particles

- When you application has started and when the run manager has been initialized, you can:

- Check the physics processes attached and their ordering:
  - `/particle/select e-`
  - `/particle/processes/dump`

- Check what particles exist:
  - `/particle/list`
- Check a particle property:
  - `/particle/select e-`
  - `/particle/property/dump`
- Please type “help” to get the full set of commands