Queen's University, Belfast (UK)
23th January 2013

Particles, processes and production cuts

Geant 4 tutorial course

INFN Geant4

Outline

- Introduction
  - A mention to physics list
  - Required methods
- Particles
  - definition and construction
- Processes
  - The G4VProcess class
  - Handling multiple processes
- Production cuts

Introduction

Mandatory user classes in a Geant4:

- **G4VUserPrimaryGeneratorAction**
- **G4VUserDetectorConstruction**
- **G4VUserPhysicsList**

Particles, ***physics processes*** *and* ***cut-off parameters*** *to be used in the simulation must be defined in the* ***G4VUserPhysicsList*** *class*

Why a physics list?

- "*Physics is physics* – shouldn't Geant4 provide, as a default, a complete set of physics that everyone can use?"
- **NO**:
  - Software can only capture Physics through a modelling
    - No unique Physics modelling
      - Very much the case for hadronic physics
      - But also the electromagnetic physics
      - Existing models still evolve and new models are created
    - Some modellings are more suited to some energy ranges
      - Medical applications not interested in multi-GeV physics in general
      - HEP experiments not interested in effects due to atomic shell structure
  - computation speed is an issue
    - a user may want a less-detailed, but faster approximation

Why a physics list?

- For this reason Geant4 takes an atomistic, rather than an integral approach to physics
  - provide many physics components (processes) which are de-coupled from one another
  - user selects these components in custom-designed physics lists
- This physics environment is built by the user in a flexible way:
  - picking up the particles he wants
  - picking up the physics to assign to each particle
- User must have a good understanding of the physics required
  - omission of particles or physics could cause errors or poor simulation

User may also use some provided "ready-to-use" physics list

`G4VUserPhysicsList`: required methods

`ConstructParticle()`:

- choose the particles you need in your simulation, define all of them here

`ConstructProcess()` :

- for each particle, assign all the physics processes relevant to your simulation
  - What's a process ?
    - a class that defines how a particle should interact with matter, or decays
      - » it's where the physics is!

`SetCuts()` :

- set the range cuts for secondary production
  - What's a range cut ?
    - a threshold on particle production
      - » Particle unable to travel at least the range cut value are not produced

Particles: basic concepts

There are three levels of class to describe particles in Geant4:

- **`G4ParticleDefinition`**

  – define a particle

  aggregates information to characterize a particle's properties (name, mass, spin, etc…)

- **`G4VDynamicParticle`**

  – describe a particle interacting with materials

  aggregates information to describe the dynamic of particles (energy, momentum, polarization, etc…)

- **`G4VTrack`**

  – describe a particle travelling in space and time

  includes all the information for tracking in a detector simulation (position, step, current volume, track ID, parent ID, etc…)

Definition of a particle

Geant4 provides the **`G4ParticleDefinition`** definition class to represent a large number of elementary particles and nuclei, organized in six major categories:
lepton, meson, baryon, boson, shortlived and ion

- Each particle is represented by its own class, which is derived from **`G4ParticleDefinition`**
- Proprieties characterizing individual particles are "read only" and can not be changed directly

*User must define **all particles** type which are used in the application: not only **primary particles** but also all other particles which may appear as **secondaries** generated by the used physics processes*

Constructing particles

Due to the large number of particles can be necessary to define, this method sometimes can be not so comfortable

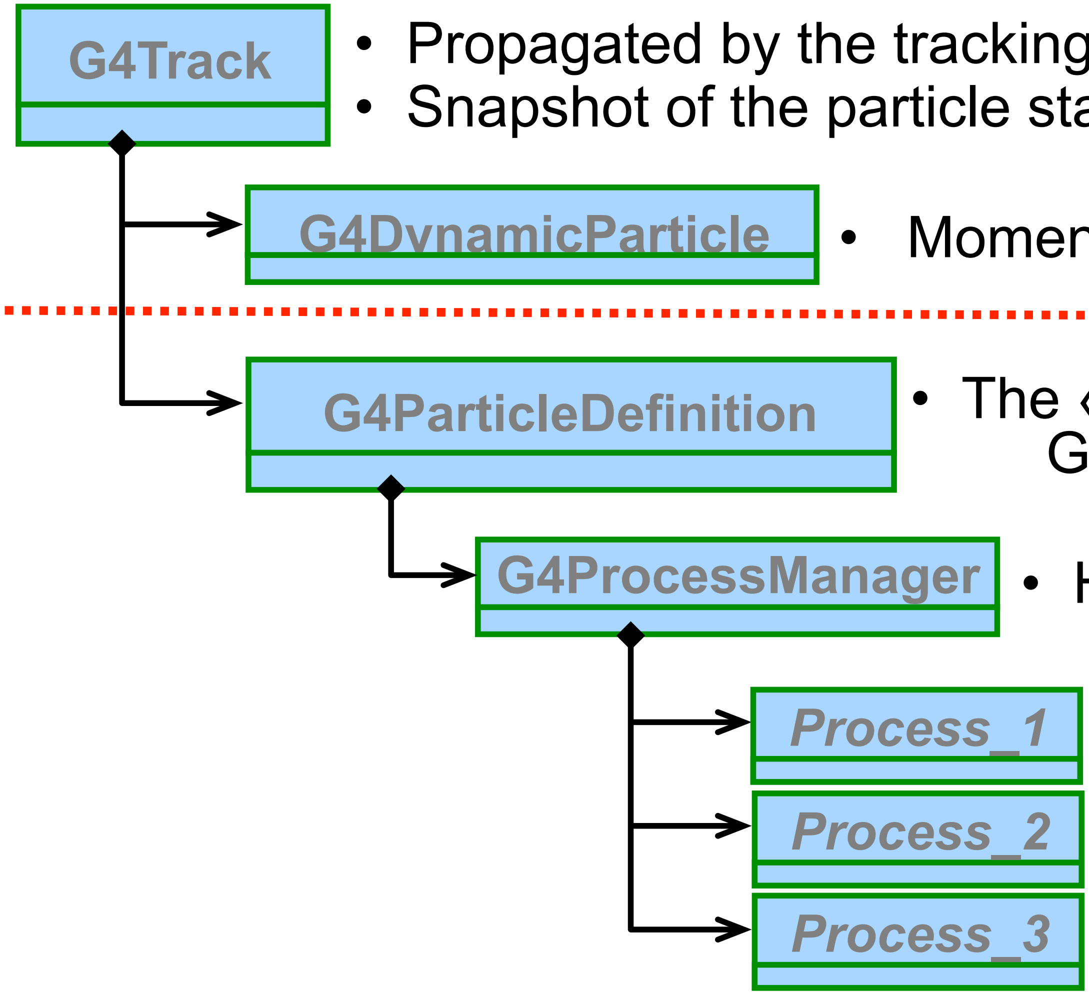
```
void MyPhysicsList::ConstructParticle ()
{
    G4Electron::ElectronDefinition();
    G4Proton::ProtonDefinition();
    G4Neutron::NeutronDefinition();
    G4Gamma::GammaDefinition();
    ....
}
```

It is possible to define **all** the particles belonging to a **Geant4 category:**

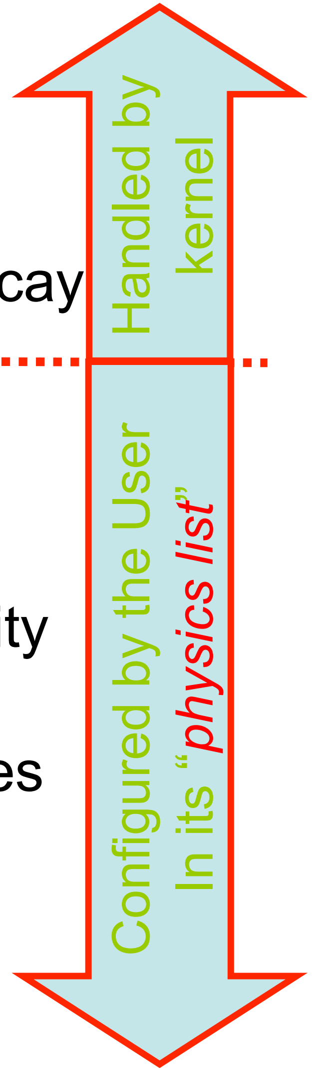
- `G4LeptonConstructor`
- `G4MesonContructor`
- `G4BarionConstructor`
- `G4BosonConstructor`
- `G4ShortlivedConstructor`
- `G4IonConstructor`

```
void
 MyPhysicsList::ConstructBaryons()
{
   // Construct all baryons
 G4BaryonConstructor pConstructor;
 pConstructor.ConstructParticle();
}
```

From particles to processes



- Propagated by the tracking,
- Snapshot of the particle state.
- Momentum, pre-assigned decay
- The « particle type »: G4Electron, ...
- Holds physics sensitivity
- … i.e. the processes



Processes

Physics processes describe how particles interact with materials

Geant4 provides seven major categories of processes:

- Electromagnetic
- Hadronic
- Decay
- Optical
- Photolepton_hadron
- Parameterization
- Transportation

A process does two things:

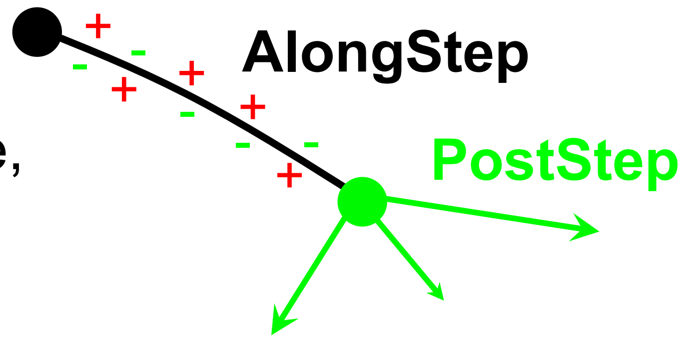
- decides when and where an interaction will occur
  - method: **`GetPhysicalInteractionLength()`** → *limit the step*
  - this requires a cross section
  - for the transportation process, the distance to the nearest object
- generates the final state of the interaction (changes momentum, generates secondaries, etc.)
  - method: **`DoIt()`**
  - this requires a model of the physics

G4Vprocess class

Physics processes are derived from the **G4VProcess** base class

- Abstract class defining the common interface of all processes in Geant4:
  - Used by all physics processes (also 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, occurring along the path of the particle, like ionisation;



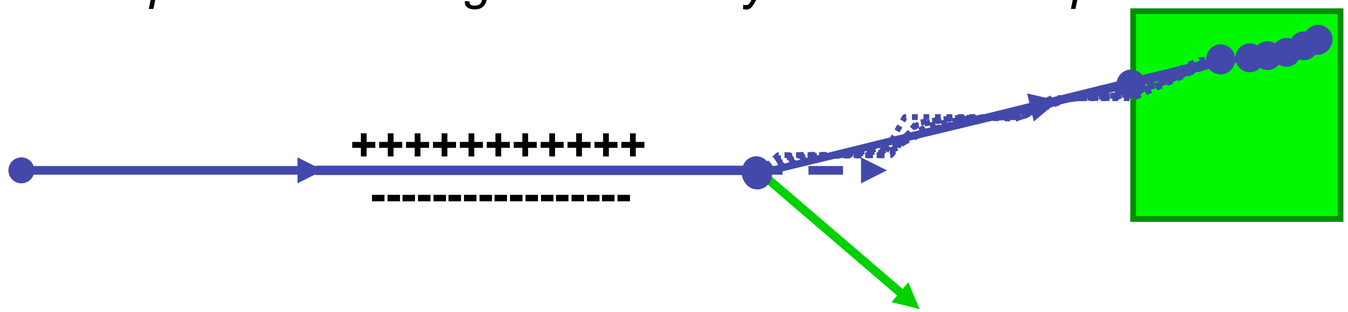
  - **PostStep** actions:
    - For describing point-like (inter)actions, like decay in flight, hadronic interactions …

A process can implement a combination of them (decay = AtRest + PostStep)

Handling multiple processes

- STAGE 1: a particle is shot and “transported”
- STAGE 2: all processes associated to the particle propose a geometrical step length (depends on process cross-section)
- STAGE 3: The process proposing the shortest step “wins” and the particle is moved to destination (if shorter than “Safety”)
- STAGE 4: All processes “along the step” are executed (e.g. ionization)
- STAGE 5: “post step” phase of the process that limited the step is executed New tracks are “pushed” to the stack
- STAGE 6: If E_{kin}=0 all “at rest” processes are executed; if particle is stable the track is killed. Else:
- STAGE 7: A new step starts and sequence repeats...

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.



Example processes

- Discrete process: Compton Scattering, hadronic inelastic, ...
  - step determined by cross section, interaction at end of step
    - PostStepGPIL(), PostStepDoIt()
- Continuous process: Cerenkov effect
  - photons created along step, roughly proportional to step length
    - AlongStepGPIL(), AlongStepDoIt()
- At rest process: mu- capture at rest
  - interaction at rest
    - AtRestGPIL(), AtRestDoIt()

pure

- Rest + discrete: positron annihilation, decay, ...
  - both in flight and at rest
- Continuous + discrete: ionization
  - energy loss is continuous
  - knock-on electrons (δ-ray) are discrete

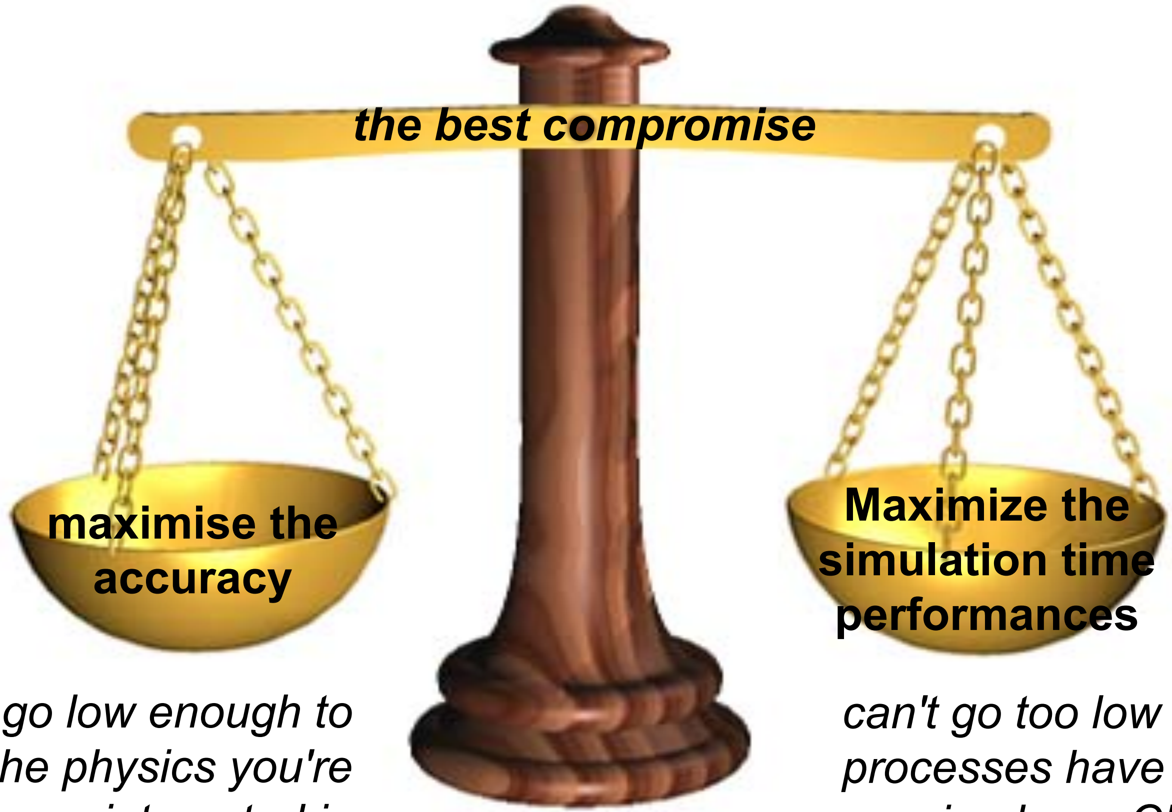
combined

Production thresholds: cut

Each simulation developer must answer the question:
how low can you go?

- should I produce (and track) everything or consider thresholds?

This is a balancing act:



need to go low enough to get the physics you're interested in

can't go too low because some processes have infrared divergence causing huge CPU time

Production thresholds: cut

- The traditional Monte Carlo solution is to impose an absolute cutoff in energy:
  - particles are stopped when this energy is reached
  - remaining energy is dumped at that point
- But, such a cut may cause **imprecise stopping location** and deposition of energy
- There is also a **particle dependence**
  - range of 10 keV γ in Si is different from range of 10 keV e- in Si is a few microns
- And a **material dependence**
  - suppose you have a detector made of alternating sheets of Pb and plastic scintillator
  - if the cutoff is OK for Pb, it will likely be wrong for the scintillator which does the actual energy deposition measurement

Production thresholds: cut

- In Geant4 there are no tracking cuts
  - *particles are tracked down to a zero range/kinetic energy*
- Only **production cuts** exist
  - i.e. cuts allowing a particle to be born or not
  - Applied to: ***gamma***, ***electron***, ***positron***, ***proton***
- *Why are production cuts needed ?*

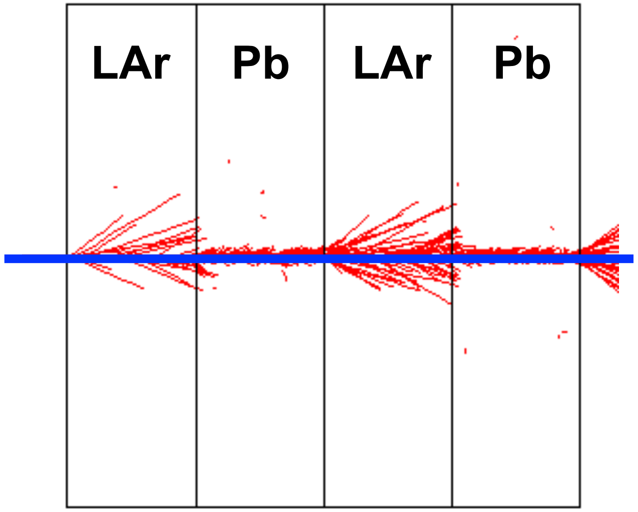
  Some electromagnetic processes involve infrared divergences
  - this leads to a huge number of smaller and smaller energy photons/ electrons (such as in Bremsstrahlung, d-ray production)
  - production cuts limit this production to particles above the threshold
  - the remaining, divergent part is treated as a continuous effect (i.e. AlongStep action)

Production thresholds: cut

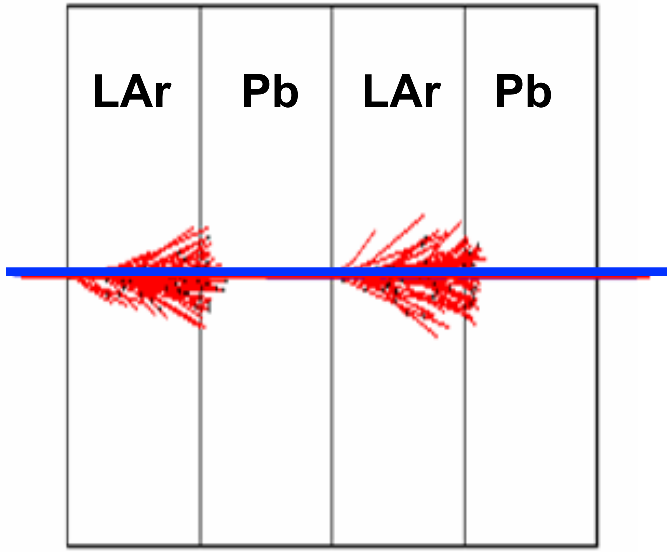
- Geant4 solution: impose a “range” production threshold
  - this threshold is a distance, not an energy
  - default = 1 mm
  - the primary particle loses energy by producing secondary electrons or gammas
  - if primary no longer has enough energy to produce secondaries which travel at least 1mm, two things happen:
    - discrete energy loss ceases (no more secondaries produced)
    - the primary is tracked down to zero energy using continuous energy loss
- Stopping location is therefore correct
- Only one value of production threshold distance is needed for all materials because it corresponds to different energies depending on material.

Production thresholds: cut

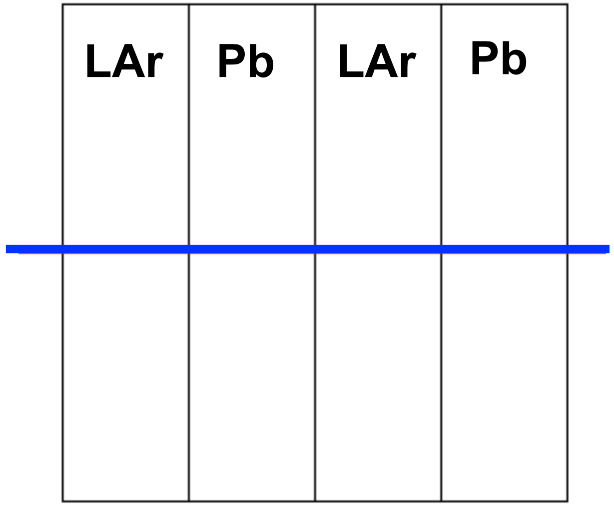
500 MeV p in LAr-Pb sampling calorimeter



Cut = 455 keV
(range in LAr = 1.5 mm)



Production range =1.5 mm



Cut = 2 MeV

Threshold in range: 1.5 mm

455 keV electron energy in liquid Ar

2 MeV electron energy in Pb

Cuts per region

- In a complex detector there may be many different types of sub-detectors involving
  - finely segmented volumes
  - very sensitive materials
  - large, undivided volumes
  - inert materials
- The same value of the secondary production threshold may not be appropriate for all of these
  - user must define regions of similar sensitivity and granularity and assign a different set of production thresholds (cuts) for each
- Warning: this feature is for users who are
  - simulating complex detectors
  - experienced at simulating EM showers in matter

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"
- 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

Thanks for your attention