

# Geant 4

## Particles and Processes

<http://cern.ch/geant4>

The full set of lecture notes of this Geant4 Course is available at  
<http://www.ge.infn.it/geant4/events/nss2004/geant4course.html>

# Physics

From the Minutes of LCB (LHCC Computing Board) meeting on 21/10/1997:

“It was noted that experiments have requirements for independent, alternative physics models. In Geant4 these models, differently from the concept of packages, allow the user to **understand** how the results are produced, and hence improve the **physics validation**. Geant4 is developed with a modular architecture and is the ideal framework where existing components are integrated and new models continue to be developed.”

# Physics: general features

- Ample variety of physics functionalities
- Uniform treatment of electromagnetic and hadronic processes
- Abstract interface to physics processes
  - Tracking independent from physics
- Distinction between processes and models
  - often multiple models for the same physics process (complementary/alternative)
- Open system
  - Users can easily create and use their own models
- Transparency (supported by encapsulation and polymorphism)
  - Calculation of cross-sections independent from the way they are accessed (data files, analytical formulae etc.)
  - Distinction between the calculation of cross sections and their use
  - Calculation of the final state independent from tracking
- Modular design, at a fine granularity, to expose the physics
- Explicit use of units throughout the code
- Public distribution of the code, from one reference repository worldwide

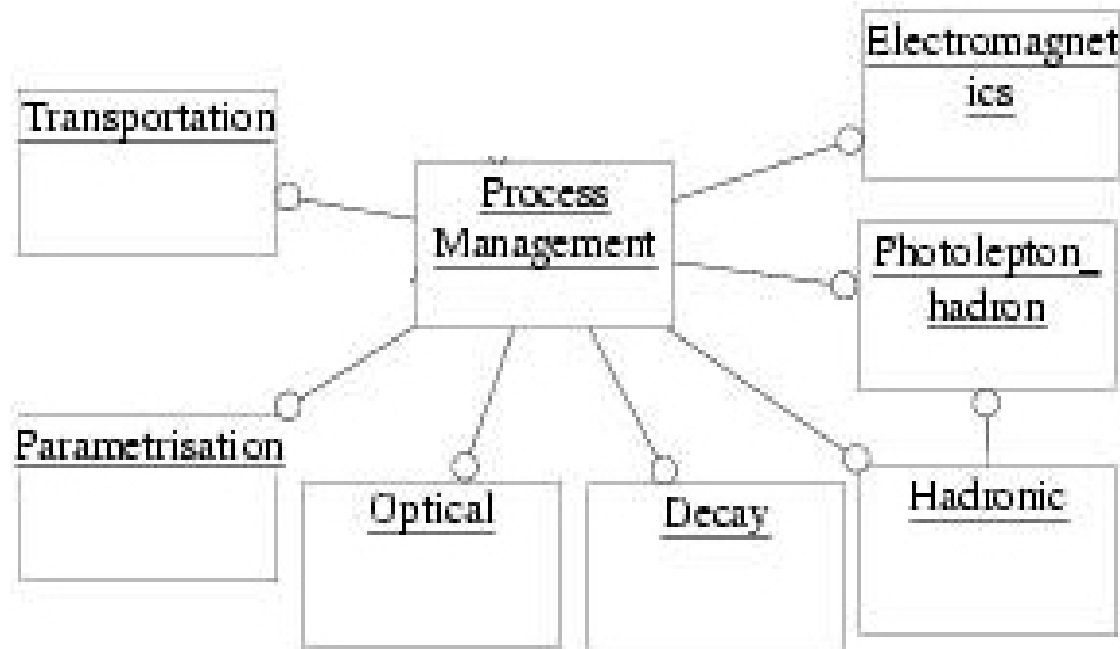
# Data libraries & Units

---

- Systematic collection and evaluation of experimental data from many sources worldwide
- Databases
  - ENDF/B, JENDL, FENDL, CENDL, ENSDF, JEF, BROND, EFF, MENDL, IRDF, SAID, EPDL, EEDL, EADL, SANDIA, ICRU etc.
- Collaborating distribution centres
  - NEA, LLNL, BNL, KEK, IAEA, IHEP, TRIUMF, FNAL, Helsinki, Durham, Japan etc.
- The use of evaluated data is important for the validation of physics results of the experiments
- Geant4 is independent from the system of units
  - all numerical quantities expressed with their units explicitly

# Processes

- Processes describe how particles interact with material or with a volume
- Three basic types
  - **At rest** process  
(*eg. decay at rest*)
  - **Continuous** process  
(*eg. ionisation*)
  - **Discrete** process  
(*eg. Compton scattering*)
- Transportation is a process
  - interacting with volume boundary
- A process which requires the shortest interaction length limits the step



# Outline

---

- What is tracked

*G4ParticleDefinition*  
*G4DynamicParticle*  
*G4Track*

- The process interface

*G4VProcess*  
*How processes are used in tracking*

- The production cuts

*Why production cuts are needed*  
*The cuts scheme in Geant4*

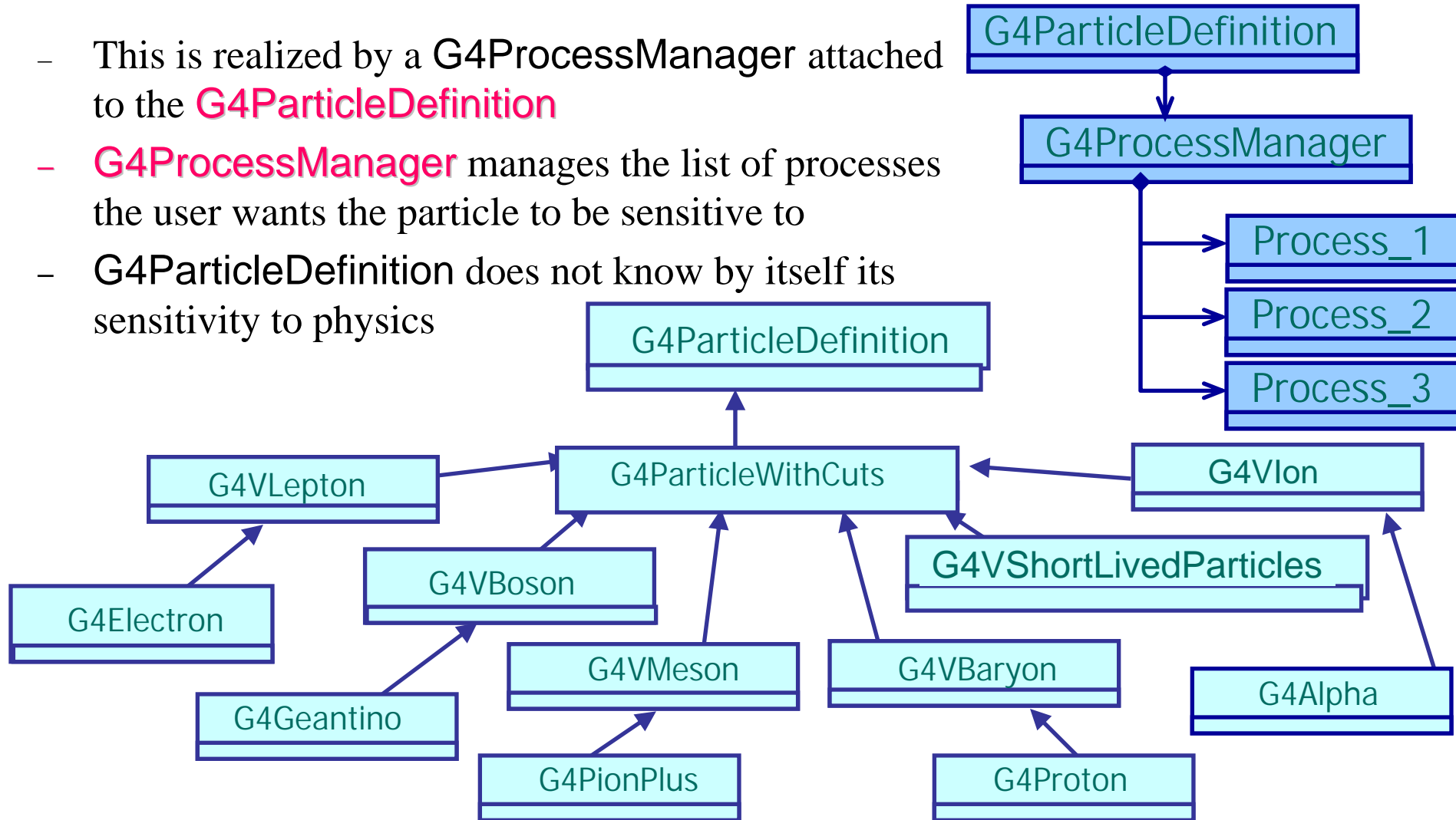
- Building the PhysicsLists

*G4VUserPhysicsList*  
*Concrete physics lists*

# G4ParticleDefinition

- intrinsic particle properties: *mass, width, spin, lifetime...*
- sensitivity to physics

- This is realized by a **G4ProcessManager** attached to the **G4ParticleDefinition**
- **G4ProcessManager** manages the list of processes the user wants the particle to be sensitive to
- **G4ParticleDefinition** does not know by itself its sensitivity to physics



**G4ParticleDefinition** is the base class for defining concrete particles

# More about particle design

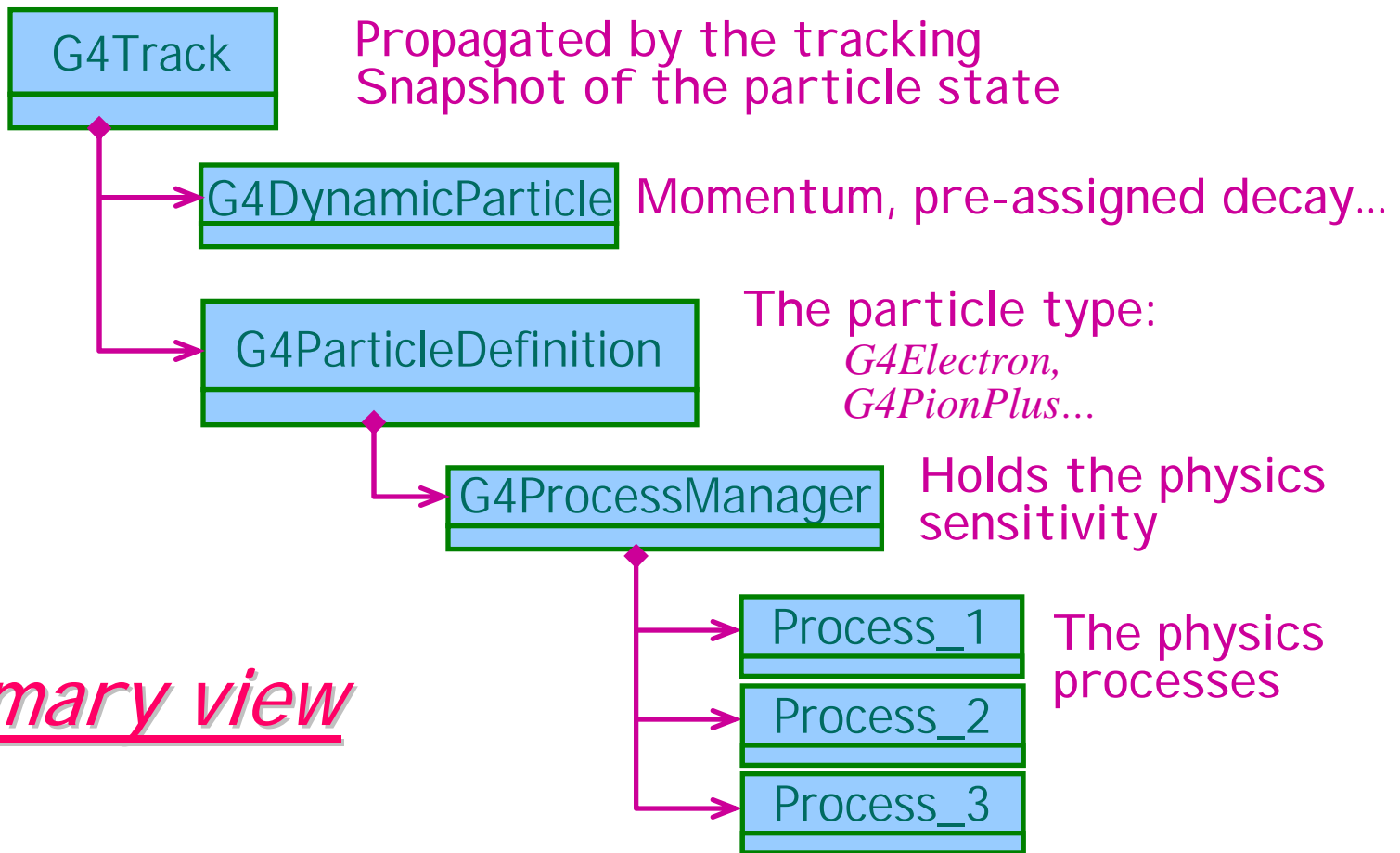
## G4DynamicParticle

- Describes the purely dynamic part (*i.e. no position, nor geometrical information...*) of the particle state:
  - momentum, energy, polarization
- Holds a G4ParticleDefinition pointer
- Retains eventual pre-assigned decay information
  - decay products
  - lifetime

## G4Track

- Defines the class of objects propagated by Geant4 tracking
- Represents a snapshot of the particle state
- Aggregates
  - a G4ParticleDefinition
  - a G4DynamicParticle
  - geometrical information:
    - *position, current volume ...*
  - track ID, parent ID;
  - process which created this G4Track
  - weight, used for event biasing





## Summary view

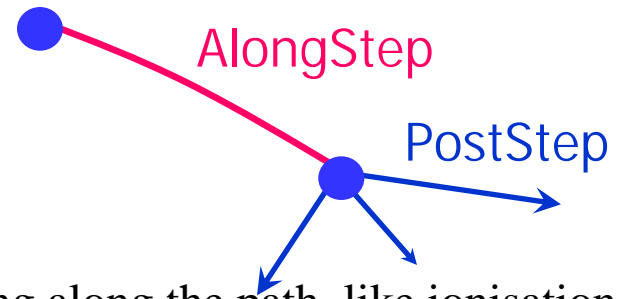
The classes involved in building the **PhysicsList** are:

- the G4ParticleDefinition concrete classes
- the G4ProcessManager
- the processes

# G4VProcess

Abstract class defining the common interface of all processes in Geant4

- Define three kinds of actions:
  - **AtRest** actions: decay, annihilation ...
  - **AlongStep** actions: continuous interactions occurring along the path, like ionisation
  - **PostStep** actions: point-like interactions, like decay in flight, hard radiation...
- A process can implement *any combination* of the three AtRest, AlongStep and PostStep actions: eg: decay = AtRest + PostStep
- Each action defines two methods:
  - **GetPhysicalInteractionLength()**  
used to limit the step size
    - *either because the process triggers an interaction or a decay*
    - *or in other cases, like fraction of energy loss, geometry boundary, user's limit...*
  - **DoIt()**
    - implements the actual action to be applied to the track
    - implements the related production of secondaries



# Processes, ProcessManager and Stepping

- G4ProcessManager retains three vectors of actions:
  - one for the **AtRest** methods of the particle
  - one for the **AlongStep** ones
  - one for the **PostStep** actions
  - *these are the vectors which the user sets up in the PhysicsList and which are used by the tracking*
- The stepping treats processes generically
  - it does not know which process it is handling
- The stepping lets the processes
  - cooperate for **AlongStep** actions
  - compete for **PostStep** and **AtRest** actions
- Processes emit also signals to require particular treatment:
  - **notForced**: normal case
  - **forced**: PostStepDoIt action applied anyway;
  - **conditionallyForced**: PostStepDoIt applied if AlongStep has limited the step

# Invocation sequence of processes: particle in flight

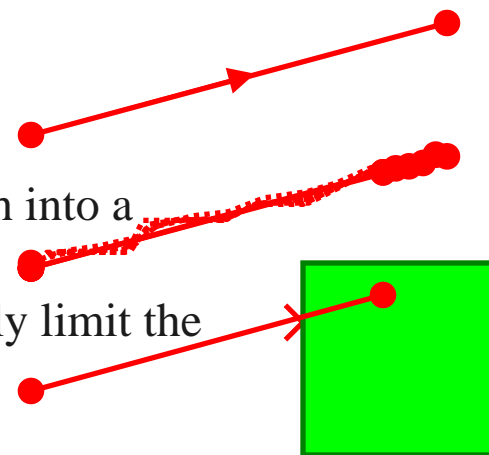
- At the beginning of the step, determine the **step length**
  - consider all processes attached to the current G4Track
  - define the step length as the smallest of the lengths among
    - all AlongStepGetPhysicalInteractionLength()
    - all PostStepGetPhysicalInteractionLength()
- Apply all **AlongStepDoIt()** actions **at once**
  - changes computed from particle state at the beginning of the step
  - accumulated in G4Step
  - then applied to G4Track, by G4Step
- Apply **PostStepDoIt()** action(s) **sequentially**, as long as the particle is alive
  - apply PostStepDoIt() of the process which proposed the smallest step length
  - apply *forced* and *conditionnally forced* actions

# Invocation sequence of processes: particle at rest

- If the particle is at rest, is stable and cannot annihilate, it is **killed** by tracking
  - more properly said: if a particle at rest has no *AtRest* actions defined, it is killed
- Otherwise determine the **lifetime**
  - Take the smallest time among all `AtRestGetPhysicalInteractionLength()`
  - Called *physical interaction length*, but it returns a time
- Apply the **AtRestDolt()** action of the process which returned the smallest time

# Processes ordering

- Ordering of following processes is critical:
  - 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 limit the step
- Other processes ordering usually do not matter



# Cuts in Geant4

- 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

*Why are production cuts needed ?*

- Some electromagnetic processes involve **infrared divergences**
  - this leads to an infinity [huge number] of smaller and smaller energy photons/electrons (*such as in Bremsstrahlung,  $\delta$ -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)

# Range vs. energy production cuts

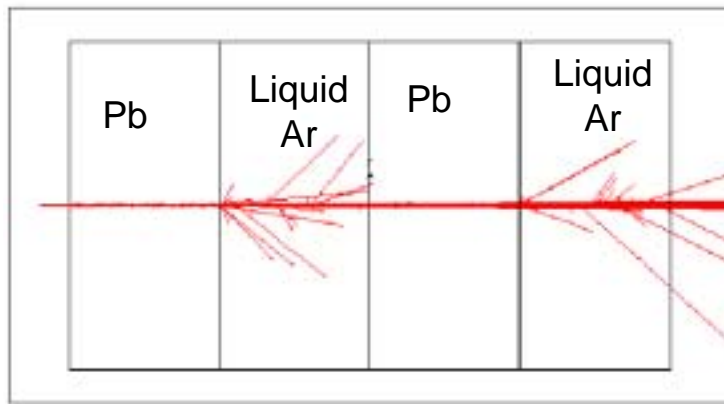
- The production of a secondary particle is relevant if it can generate visible effects in the detector
  - otherwise “*local energy deposit*”
- A **range cut** allows to easily define such visibility
  - “*I want to produce particles able to travel at least 1 mm*”
  - criterion which can be applied uniformly across the detector (whole or “region”)
- The same **energy cut** leads to very different ranges
  - for the same particle type, depending on the material
  - for the same material, depending on particle type
- The user specifies a unique range cut in the PhysicsList
  - this range cut is converted into energy cuts
  - each particle (*G4ParticleWithCut*) converts the range cut into an energy cut, for each material
  - processes then compute the cross-sections based on the energy cut



# Effect of production thresholds

## Geant 4

500 MeV incident proton



Threshold in range: 1.5 mm

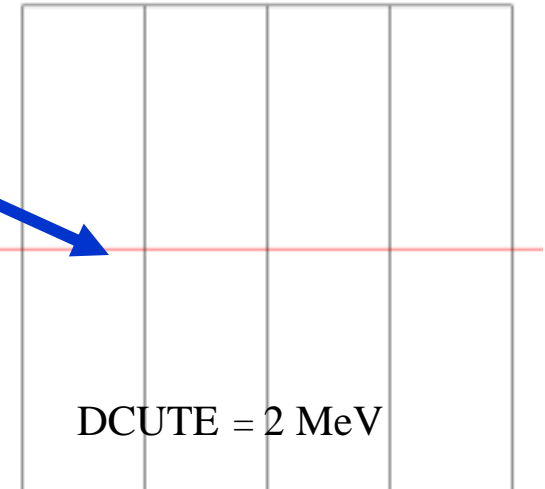
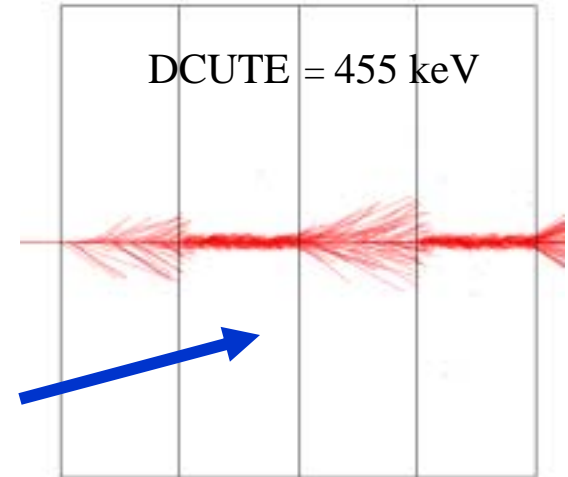


455 keV electron energy in liquid Ar  
2 MeV electron energy in Pb

In Geant3

one must set the cut for delta-rays (DCUTE) either to the Liquid Argon value, thus producing many small unnecessary  $\delta$ -rays in Pb,

or to the Pb value, thus killing the  $\delta$ -rays production everywhere



# Violations of the production threshold

- In some cases particles are produced even if they are **below the production threshold**
- This is intended to let the processes *do the best they can*
- It happens typically for
  - decays
  - positron production:
    - in order to simulate the resulting photons from the annihilation
  - hadronic processes:
    - since no infrared divergences affect the cross-sections
- *Note these are not “hard-coded” exceptions, but a sophisticated, generic mechanism of the tracking*

# G4VUserPhysicsList

---

- It is one of the mandatory user classes (*abstract class*)
- Pure virtual methods
  - ConstructParticles()
  - ConstructProcesses()
  - SetCuts()

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

# Electromagnetic physics

energy loss

- Multiple scattering
- Bremsstrahlung
- Ionisation
- Annihilation
- Photoelectric effect
- Compton scattering
- Rayleigh effect
- $\gamma$  conversion
- $e^+e^-$  pair production
- Synchrotron radiation
- Transition radiation
- Cherenkov
- Refraction
- Reflection
- Absorption
- Scintillation
- Fluorescence
- Auger

- electrons and positrons
- $\gamma$ , X-ray and optical photons
- muons
- charged hadrons
- ions

Comparable to Geant3 already in the  $\alpha$  release (1997)

Further extensions (*facilitated by the OO technology*)

## • High energy extensions

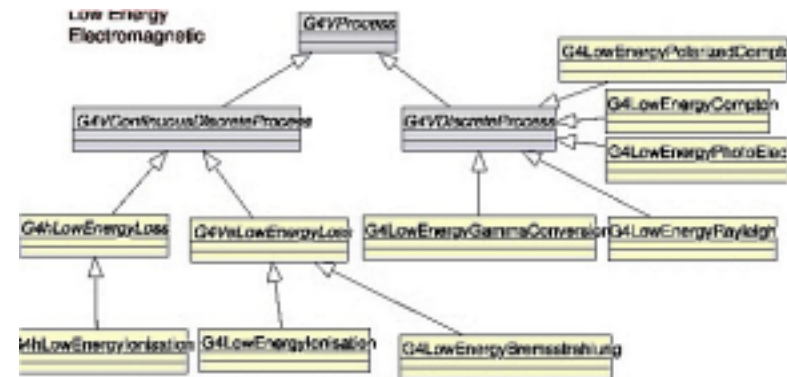
- needed for LHC experiments, cosmic ray experiments...

## • Low energy extensions

- fundamental for space and medical applications, dark matter and  $\nu$  experiments, antimatter spectroscopy etc.

## • Alternative models for the same process

*All obeying to the same abstract Process interface* → transparent to tracking



# Hadronic physics

- Completely different approach w.r.t. the past (Geant3)
  - native
  - transparent
  - no longer interface to external packages
  - clear separation between data and their use in algorithms
- Cross section data sets
  - transparent and interchangeable
- Final state calculation
  - models by particle, energy, material
- Ample variety of models
  - the most complete hadronic simulation kit on the market
  - Alternative/complementary models
  - it is possible to mix-and-match, with fine granularity
  - data-driven, parameterised and theoretical models
- Consequences for the users
  - no more confined to the black box of one package
  - the user has control on the physics used in the simulation, which contributes to the validation of experiment's results

# Summary

---

- **Transparency** and **modularity** are the key characteristics of Geant4 physics
- Ample variety of processes and models
  - Openness to extension and evolution thanks to the OO technology
- The PhysicsList exposes, **deliberately**, the user to the **choice** of physics (*particles + processes*) relevant to his/her application
  - This is a critical task, but guided by the framework
  - Examples can be used as starting point
- Physics processes and models are documented in Geant4 Physics Reference Manual