XIII SEMINAR ON SOFTWARE FOR NUCLEAR, SUBNUCLEAR AND APPLIED PHYSICS

Alghero, Italy
6 – 10 June 2016

Physics in Geant4: Particles, processes and cuts

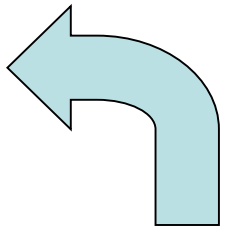
J. Pipek, F. Romano, ...

Geant 4 tutorial

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 implementation of physics that everyone can use?
- **NO**:
  - Physics has to described by models
    - No universal physics models
      - Very much the case for hadronic physics
      - But also the electromagnetic physics
      - Existing models still evolve and new models are created
    - Different models are more suited for different energy ranges
      - Medical applications not interested in multi-GeV physics in general
      - HEP experiments not interested in effects due to atomic shell structure
    - Testing of new hypothetical physics
  - computation speed is an issue
    - a user may want a less-detailed, but faster approximation

Physics list = user's responsibility

- Geant4 takes an *atomistic* approach to physics and provides:
  - huge variety of available particles
  - many physics independent components (processes)
  - different descriptions of these processes (models)

- Complete physics is then selected by user in a flexible way:
  - select particles
  - assign processes to these particles
  - *(customize these processes)*
  - *(implement new processes)*

- Good understanding of the physics required
  - omission of particles or physics could cause errors or poor simulation

Alternative: "ready-to-use" physics lists or constructors.

`G4VUserPhysicsList`: implementation

`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*

`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`**

  Particle static properties: name, mass, spin, PDG number, etc.

- **`G4DynamicParticle`**

  Particle dynamic state: energy, momentum, polarization, etc.

- **`G4Track`**

  Information for tracking in a detector simulation: position, step, current volume, track ID, parent ID, etc.

Definition of a particle

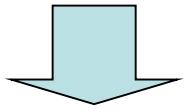
Geant4 provides **`G4ParticleDefinition`** daughter classes to represent a large number of elementary particles and nuclei, organized in six major categories:
leptons, ***mesons***, ***baryons***, ***bosons***, ***short-lived*** *and* ***ions***

- Each particle type is represented by its own class (e.g. **`G4Electron`**) which inherits from **`G4ParticleDefinition`**

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

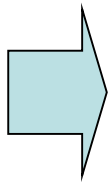
Due to the large number of particles can be necessary to instantiate, this method sometimes can be not so comfortable



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

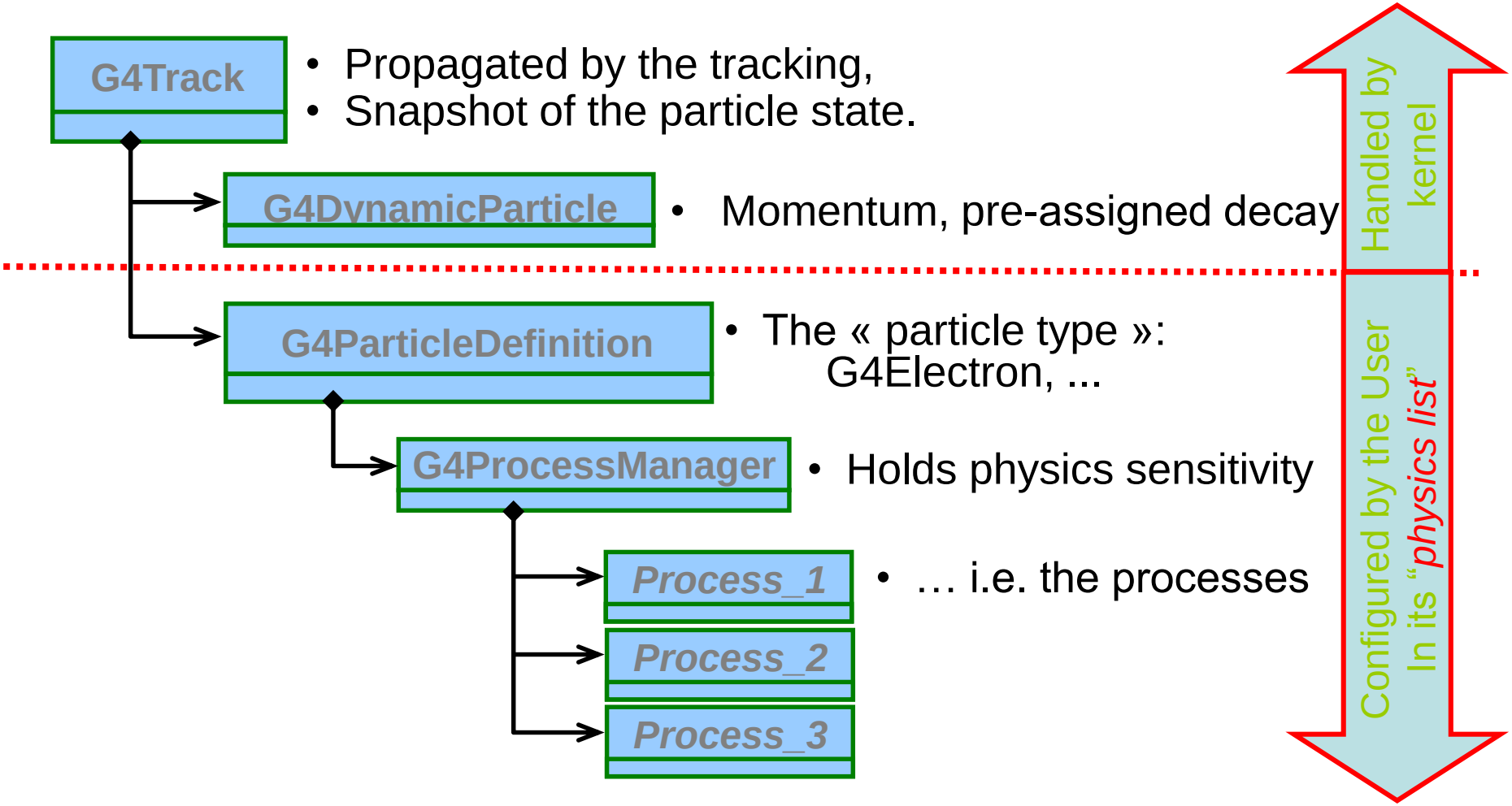
- **G4LeptonConstructor**
- **G4MesonContructor**
- **G4BarionConstructor**
- **G4BosonConstructor**
- **G4ShortlivedConstructor**
- **G4IonConstructor**

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



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

From particles to 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

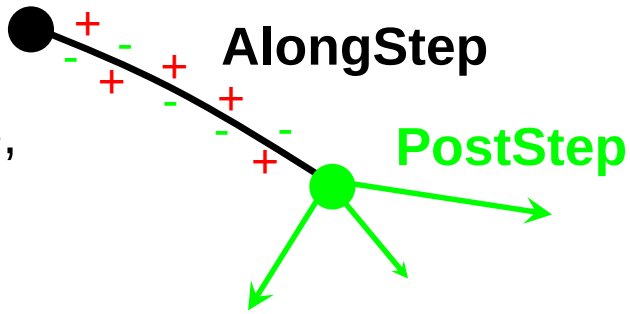
A process does two things:

1. *decides when and where an interaction will occur*
   - **`GetPhysicalInteractionLength...()`** → *limit the step*
   - this requires a cross section
   - for the transportation process, the distance to the nearest object
2. *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)

Example processes

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

Handling multiple processes

1 a particle is shot and “transported”

2 all processes associated to the particle propose a geometrical step length (depends on process cross-section)

3 The process proposing the shortest step “wins” and the particle is moved to destination (if shorter than “Safety”)

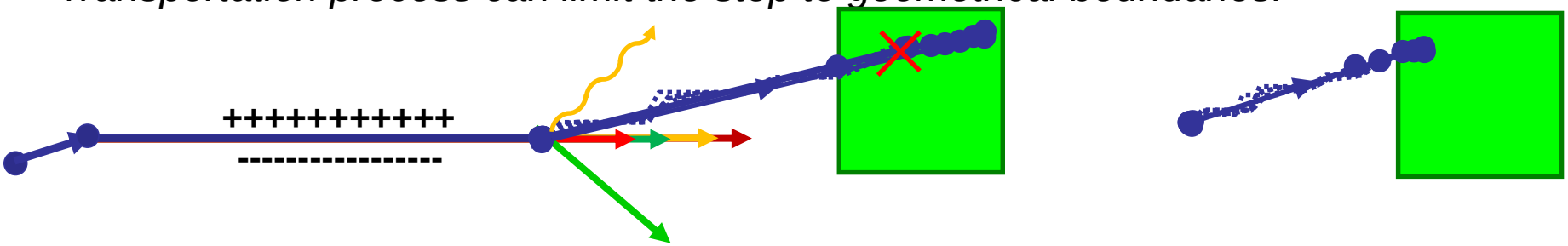
4 All processes along the step are executed (e.g. ionization)

5 post step phase of the process that limited the step is executed. New tracks are “pushed” to the stack

6 If E_{kin}=0 all at rest processes are executed; if particle is stable the track is killed. Else:

7 New step starts and sequence repeats...

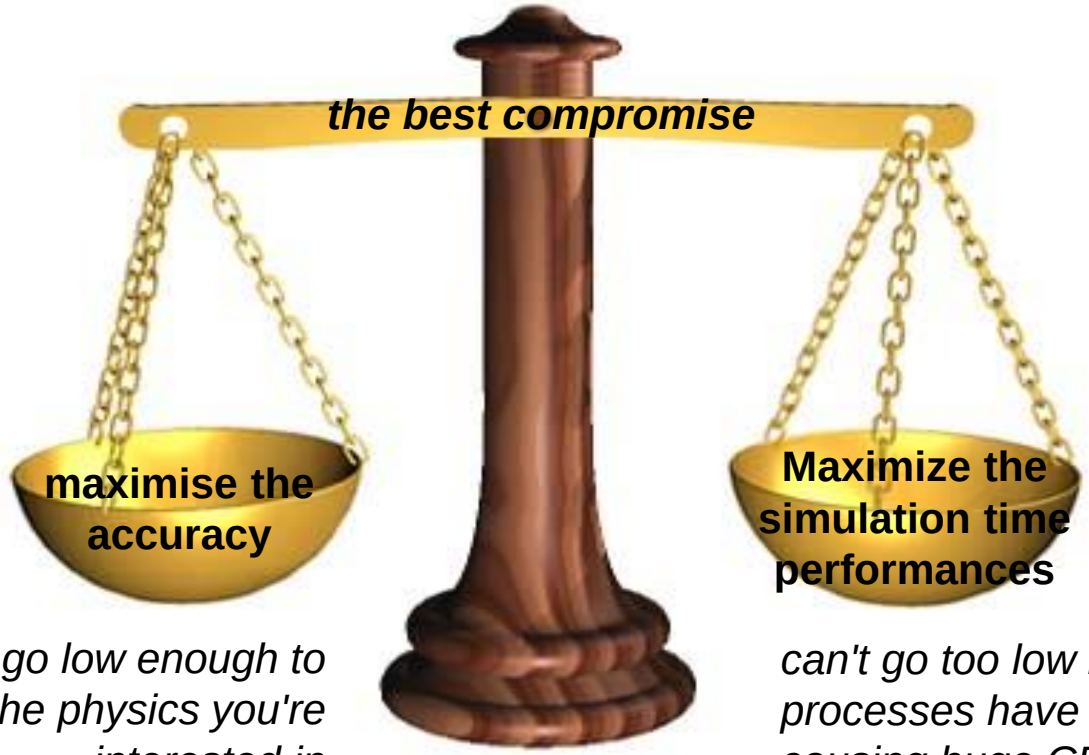
- *Processes return a “true path length”. The multiple scattering “virtually folds up” this true path length into a shorter ”geometrical” path length.*
- *Transportation process can limit the step to geometrical boundaries.*



Production thresholds: cut

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

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



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 p in Si is different from range of 10 keV e- in Si
- And a **material dependence**
  - e.g. 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*
  - *however, in principle you can implement this yourself*

- Only **production cuts** exist
  - i.e. cuts deciding whether a particle to be produced 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, δ-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) → energy balance is preserved

Production thresholds: cut

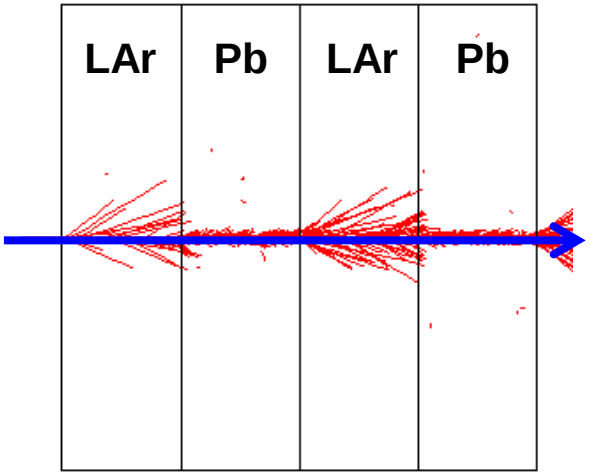
- Geant4 solution: impose a "range" production threshold
  - this threshold is a distance, not an energy
  - default = 1 mm

  Particles unable to travel at least the range cut value are not produced !

- Only one production threshold cut is *uniformly* set
- Production threshold is *internally converted* to an energy threshold, depending on **particle type** and **material**

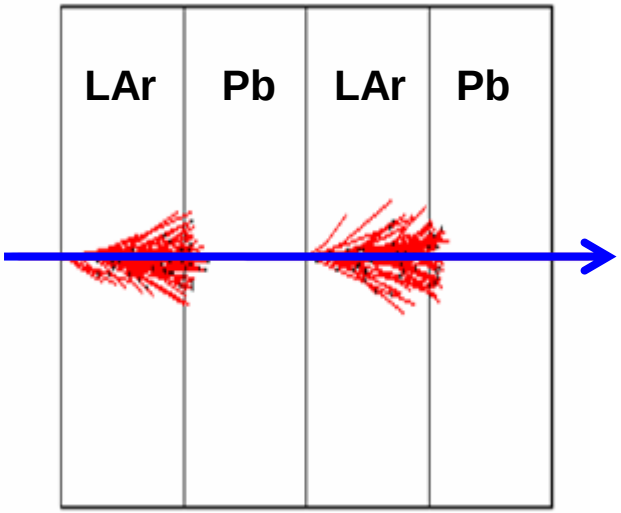
- When primary no longer has enough energy to produce secondaries which travel at least 1 mm, two things happen:
  - discrete energy loss stops (no more secondaries produced)
  - the primary is tracked down to zero energy using continuous energy loss

Production thresholds: cut

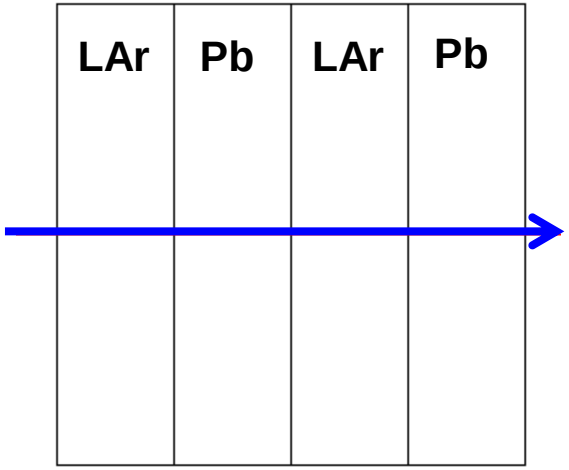


Cut = 455 keV
(range in LAr = 1.5 mm)

500 MeV p in LAr-Pb sampling calorimeter



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

- Complex detector may contain many different sub-detectors involving
  - finely segmented volumes
  - very sensitive materials
  - large, undivided volumes
  - inert materials
- The same cut may not be appropriate for all of these
  - user can define regions (indepent of geometry hierarchy tree) and assign different cuts for each region

- Warning: it is very difficult topic and requires experience!

G4StepLimiter

- Sometimes, you want to set a maximum step length.
- Why?
  - you want to see the exact track of the particle
  - you don‘t trust the chord finder for your magnetic field
- How?
  - Include **`G4StepLimiter`** process in your physics list
  - Set “user limits” for the *logical volumes* or *regions* of interest: **`SetUserLimits()`**

```
logVol->SetUserLimits(new G4UserLimits(1.0 * mm));
```

Example: Put it together

```
void StandardPhysics::ConstructParticle()
{
        // We are interested in gamma, electrons and possibly positrons
        G4Electron::ElectronDefinition();
        G4Positron::PositronDefinition();
        G4Gamma::GammaDefinition();
}

void StandardPhysics::ConstructProcess()
{
        // Transportation is necessary
        AddTransportation();

        // Electrons
        G4ProcessManager *elManager = G4Electron::ElectronDefinition()->GetProcessManager();
        elManager->AddProcess(new G4eMultipleScattering, -1, 1, 1);
        elManager->AddProcess(new G4eIonisation, -1, 2, 2);
        elManager->AddProcess(new G4eBremsstrahlung, -1, -1, 3);
        elManager->AddDiscreteProcess(new G4StepLimiter);

        // Positrons
        G4ProcessManager *posManager = G4Positron::PositronDefinition()->GetProcessManager();
        posManager->AddProcess(new G4eMultipleScattering, -1, 1, 1);
        posManager->AddProcess(new G4eIonisation, -1, 2, 2);
        posManager->AddProcess(new G4eBremsstrahlung, -1, -1, 3);
        posManager->AddProcess(new G4eplusAnnihilation, 0, -1, 4);
        posManager->AddDiscreteProcess(new G4StepLimiter);

        // Gamma
        G4ProcessManager *phManager = G4Gamma::GammaDefinition()->GetProcessManager();
        phManager->AddDiscreteProcess(new G4ComptonScattering);
        phManager->AddDiscreteProcess(new G4PhotoElectricEffect);
        phManager->AddDiscreteProcess(new G4GammaConversion);

        // TODO: Introduce Rayleigh scattering. It has large cross-section than Pair production
}

void StandardPhysics::SetCuts()
{
        // TODO: Create a messenger for this
        defaultCutValue = 0.03 * mm;
        SetCutsWithDefault();
}
```

Conclusion

- Geant4 description of physics is very flexible
  - many particles
  - many processes
  - many models
  - many physics lists

- ...more to come in the next lecture

End of process, let's cut the tasks!