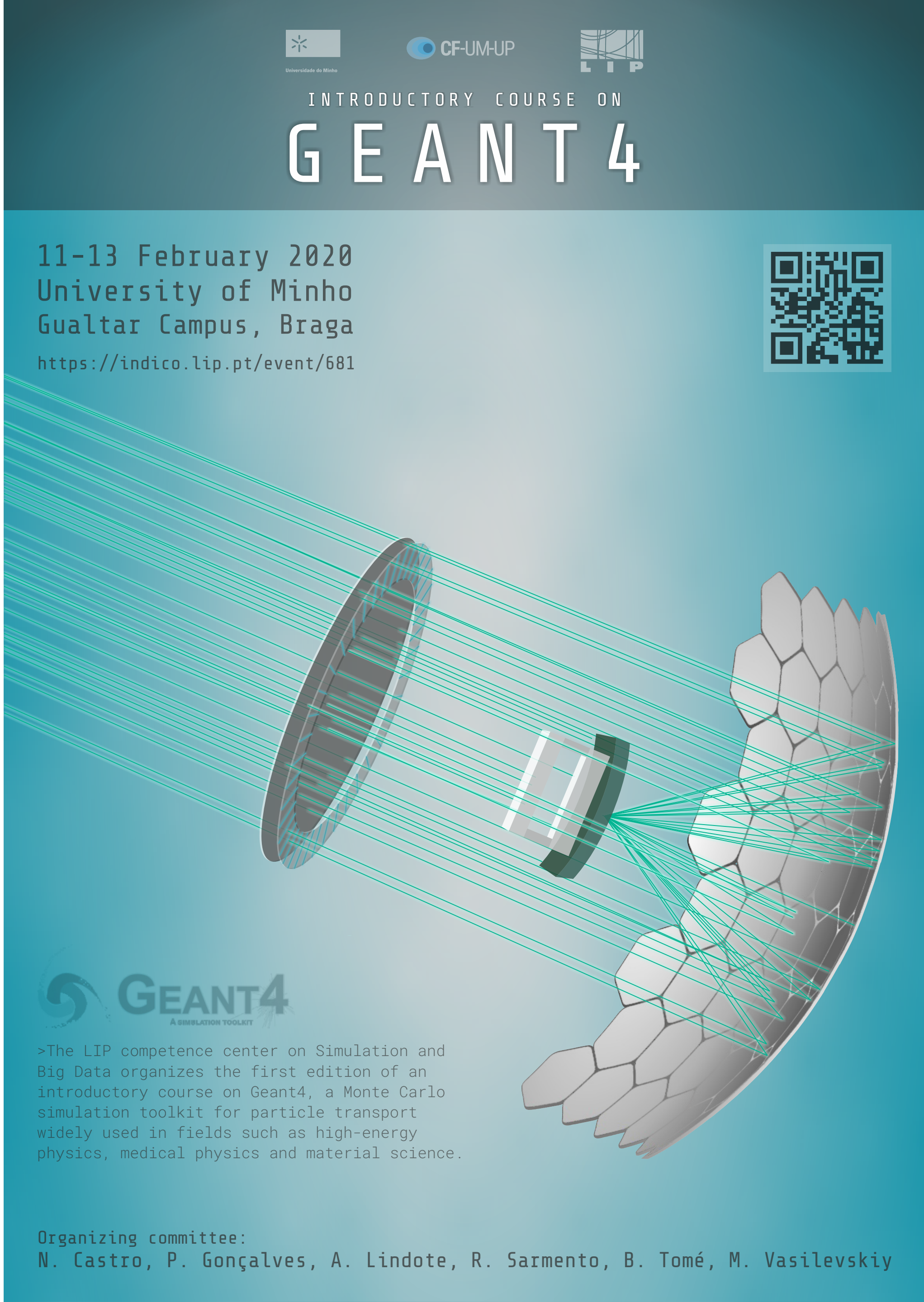
Physics Processes

Bernardo Tomé



Slides adapted from slides produced by :
Marc Verderi,Dennis Wright, Vladimir Ivantchenko, Mihaly Novak
http://cern.ch/geant4

Contents

I. Generalities on particles and processes

II. The process interface

III. The physics categories

IV. Physics Lists

V. Optical processes (afternoon)

Geant4 Physics

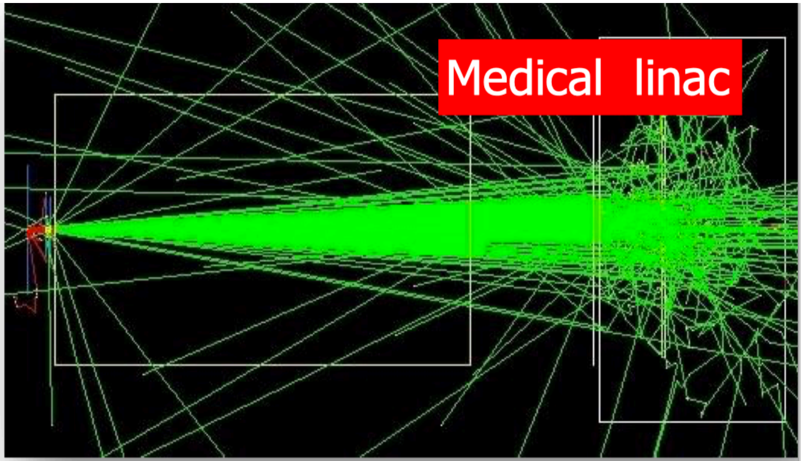
- Geant4 provides a wide variety of physics components for use in simulation
- Physics components are coded as processes
  - a process is a class which tells a particle what to do and how (how to travel, interact, decay, etc …)
  - user may write his own processes (derived from the Geant4 process abstract base class)
- Processes are grouped into
  - electromagnetic, hadronic, decay, parameterized and transportation

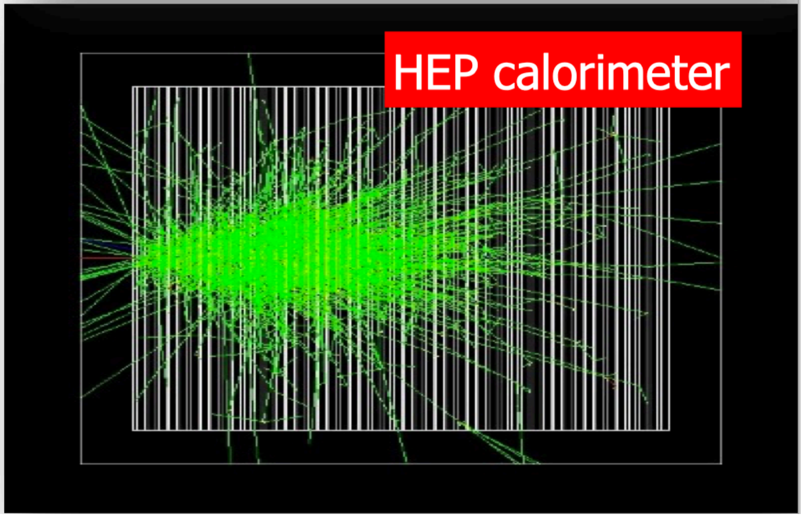
Geant4 Physics: Electromagnetic

- standard – complete set of (EM) processes covering charged particles and gammas
  - energy range 1 keV to ~PeV
- low energy – specialized routines for electrons, gammas, charged hadrons
  - more atomic shell structure details
  - some processes valid down to below keV
  - but some processes can be used only up to few GeV
- optical photon – only for long wavelength photons (x-rays, UV, visible)
  - processes for reflection/refraction, absorption, wavelength shifting, Rayleigh scattering
  - Special particle type : **G4OpticalPhoton**

Processes for Gamma and Electron

- **Photon processes**
  - γ conversion into e+e- pair
  - Compton scattering
  - Photoelectric effect
  - Rayleigh scattering
  - *Gamma-nuclear interaction in hadronic sub-package*
- **Electron and positron processes**
  - Ionisation
  - Coulomb scattering
  - Bremsstrahlung
  - Positron annihilation
  - Production of e+e- pairs
  - *Nuclear interaction in hadronic sub-package*
- Suitable for HEP & many other Geant4 applications with electron and gamma beams





Processes for Hadrons

- Pure hadronic interactions up to TeV
  - elastic
  - inelastic
  - capture
  - fission
- Radioactive decay
  - at-rest and in-flight
- photo-nuclear interaction (~1 MeV up to 100 TeV)
- lepto-nuclear interaction (~100 MeV up to 100TeV)
  - e+ and e- induced nuclear reactions
  - muon induced nuclear reactions

Geant4 Physics: Decay, Parameterized and Transportation

- Decay processes include
  - weak decay (leptonic decays, semi-leptonic decays, radioactive decay of nuclei)
  - electromagnetic decay (π^0, Σ^0, etc.)
  - strong decay are not included here (part of hadronic models)
- Parameterized processes
  - e.g. electromagnetic showers propagated according to parameters averaged over many events
  - faster than detailed shower simulation
- Transportation process
  - special process responsible to propagate the particles through the geometry
  - need to be assigned to each particle

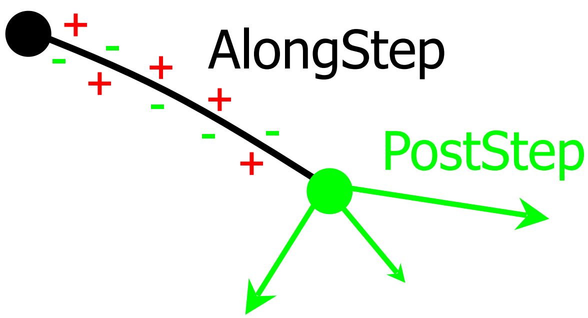
Physics Processes (1)

- All the work of particle decays and interactions is done by processes

- A process does two things:
  - decides when and where an interaction will occur
    - method: **`GetPhysicalInteractionLength()`**
    - this requires a **cross section or decay lifetime**
    - for the transportation process, the **distance to the nearest volume border** along the track is used
  - generates the final state of the interaction (changes momentum, position, generates secondaries, etc.)
    - method: **`DoIt()`**
    - this requires a **model of the physics**

Physics Processes (2)

- There are three flavors of processes:
  - well-located in space -> PostStep
  - distributed in space -> AlongStep
  - well-located in time -> AtRest



- A process may be a combination of all three of the above
  - in that case six methods must be implemented (GetPhysicalInteractionLength() and DoIt() for each action)
- “Shortcut” processes are defined which invoke only one
  - Discrete process (has only PostStep physics)
  - Continuous process (has only AlongStep physics)
  - AtRest process (has only AtRest physics)

Example Processes (1)

- Discrete process: Compton Scattering
  - step determined by cross section, interaction at end of step
    - PostStepGPIL()
    - PostStepDoIt()
- Continuous process: Cherenkov effect
  - photons created along step, # roughly proportional to step length
    - AlongStepGPIL()
    - AlongStepDoIt()
- At rest process: positron annihilation at rest
  - no displacement, time is the relevant variable
    - AtRestGPIL()
    - AtRestDoIt()
- These are examples of so-called “pure” processes

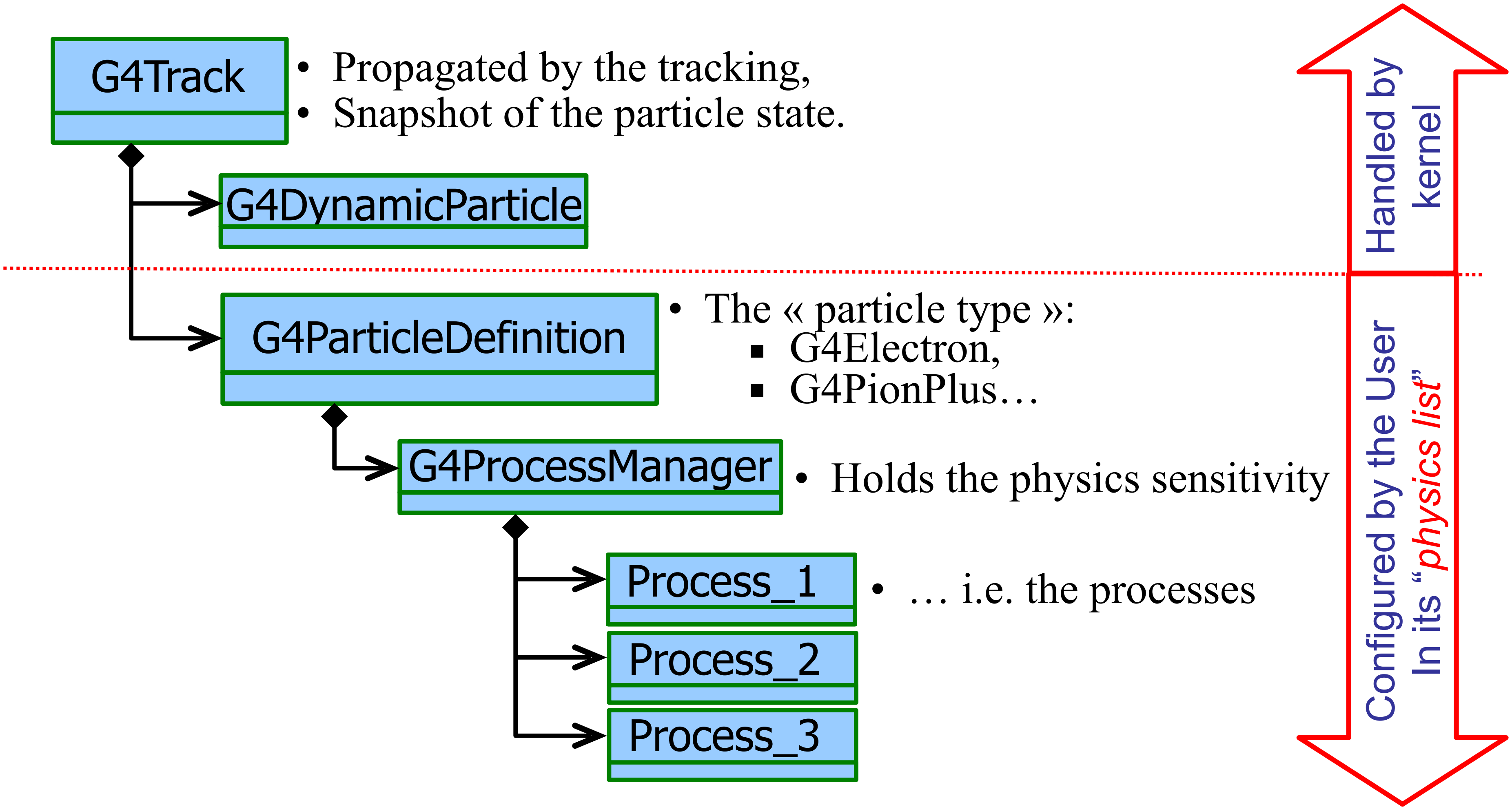
Example Processes (2)

- Continuous + discrete: ionization
  - energy loss is continuous
  - Moller/Bhabha scattering and knock-on electrons are discrete
- Continuous + discrete: bremsstrahlung
  - energy loss due to soft photons is continuous
  - hard photon emission is discrete
- In both cases, the production threshold separates the continuous and discrete parts of the process
- Multiple scattering is also continuous + discrete

Handling Multiple Processes

- Many processes (and therefore many interactions) can be assigned to the same particle
- How does Geant4 decide which interaction happens at any one time?
  - interaction length or decay length is sampled from each process
  - shortest one happens, unless
    - a volume boundary is encountered in less than the sampled length --> no physics interaction occurs (just simple transport).
  - the processes that were not chosen have their interaction lengths shortened by the distance travelled in the previous step
  - repeat the procedure

From G4Track to processes



Physics Lists

What is a Physics List?

- A class which collects all the particles, physics processes and production thresholds needed for your application
- It tells the run manager how and when to invoke physics

- It is a very flexible way to build a physics environment
  - user can pick the particles he wants
  - user can pick the physics to assign to each particle

- But, user must have a **good understanding of the physics required**
  - omission of particles or physics processes could cause errors or poor simulation

Why a Physics List?

- there are many different physics models and approximations
  - very much the case for hadronic physics
  - but also the case for electromagnetic physics
- computation speed is an issue
  - a user may want a less-detailed, but faster approximation
- no application requires all the physics and particles Geant4 has to offer
  - e.g., most medical applications do not need multi-GeV physics

Why a Physics List?

- Geant4 takes an *atomistic*, rather than an integral approach to physics description :
  - provide many physics components (processes) which are de-coupled from one another;
  - user selects these components in custom-designed physics lists in much the same way as a detector geometry is built;
  - tailor the physics to the simulation needs;

G4VUserPhysicsList

- All physics lists must **derive** from this class;
  - and then be registered with the run manager
  - one of the 3 mandatory classes

```
class YourPhysicsList: public G4VUserPhysicsList {
  public:
    // CTR
    YourPhysicsList();
    // DTR
    virtual ~YourPhysicsList();

    // pure virtual => needs to be implemented
    virtual void ConstructParticle();
    // pure virtual => needs to be implemented
    virtual void ConstructProcess();

    // virtual method
    virtual void SetCuts();
    ...
    ...
};
```

○ User must implement the 2 pure virtual methods : `ConstructParticle()` **and** `ConstructProcess()`; **`SetCuts()`** method is optional.

ConstructParticle()

- Interface method to define the list of particles to be used in the simulation

- Construct particles individually:

```
void YourPhysicsList::ConstructParticle() {
    G4Electron::Definition();
    G4Gamma::Definition();
    G4Proton::Definition();
    G4Neutron::Definition();
    // other particle definitions
    …
    …
}
```

- Construct particles by using helpers:

```
void YourPhysicsList::ConstructParticle() {
    // construct baryons
    G4BaryonConstructor baryonConstructor;
    baryonConstructor.ConstructParticle();
    // construct bosons
    G4BosonConstructor bosonConstructor;
    bosonConstructor.ConstructParticle();
    // more particle definitions
    …
    …
}
```

ConstructProcess()

- Interface method to define the list of physics processes to be used in the simulation for a given particle

```
void YourPhysicsList::ConstructProcess() {
    // method (provided by the G4VUserPhysicsList base class)
    // that assigns transportation process to all particles
    // defined in ConstructParticle()
    AddTransportation();
    // helper method might be defined by the user (for convenience)
    // to add electromagnetic physics processes
    ConstructEM();
    // helper method might be defined by the user
    // to add all other physics processes
    ConstructGeneral();
}
```

Construct Electromagnetic Physics

```
void YourPhysicsList::ConstructEM() {
  // get the physics list helper
  // it will be used to assign processes to particles
  G4PhysicsListHelper* ph = G4PhysicsListHelper::GetPhysicsListHelper();
  auto particleIterator   = GetParticleIterator();
  particleIterator->reset();
  // iterate over the list of particles constructed in ConstructParticle()
  while( (*particleIterator)() ) {
    // get the current particle definition
    G4ParticleDefinition* particleDef  = particleIterator->value();
    // if the current particle is the appropriate one => add EM processes
    if ( particleDef == G4Gamma::Definition() ) {
      // add physics processes to gamma particle here
      ph->RegisterProcess(new G4GammaConversion(),  particleDef);
      ...
      ...
    } else if ( particleDef == G4Electron::Definition() ) {
      // add physics processes to electron here
      ph->RegisterProcess(new G4eBremsstrahlung(), particleDef);
      ...
      ...
    } else if (...) {
      // do the same for all other particles like e+, mu+, mu-, etc.
      ...
    }
  }
}
```

Construct Decay Physics

```
void YourPhysicsList::ConstructGeneral() {
  // get the physics list helper
  // it will be used to assign processes to particles
  G4PhysicsListHelper* ph  = G4PhysicsListHelper::GetPhysicsListHelper();
  auto particleIterator    = GetParticleIterator();
  particleIterator->reset();
  // create processes that need to be assigned to particles
  // e.g. create decay process
  G4Decay* theDecayProcess = new G4Decay();
  ...
  ...
  // iterate over the list of particles constructed in ConstructParticle()
  while( (*particleIterator)() ) {
    // get the current particle definition
    G4ParticleDefinition* particleDef  = particleIterator->value();
    // if the process can be assigned to the current particle => do it!
    if ( theDecayProcess->IsApplicable( *particleDef ) ) {
      // add the physics processes to the particle
      ph->RegisterProcess(theDecayProcess, particleDef);
    }
    // other processes might be assigned to the current particle as well
    ...
    ...
  }
}
```

Modular Physics Lists

- A realistic physics list is likely to have many particles and physics processes;
- Such a list can become quite long, complicated and hard to maintain;
- Modular physics list provides a solution :
  - the interface is defined in `G4VModularPhysicsList`, derived from `G4VUserPhysicsList`
  - `AddTransportation()` automatically called for all registered particles
  - allows to use “physics modules” :
    - a given physics module handles consistently a well defined category of physics (e.g. electromagnetic physics, hadronic physics, decay, etc.)

G4VModularPhysicsList

```
class YourModularPhysicsList : public G4VModularPhysicsList {
  public:
    // CTR
    YourModularPhysicsList();
    ...
};

// CTR implementation
YourModularPhysicsList::YourModularPhysicsList()
: G4VModularPhysicsList() {
  // set default cut value (optional)
  defaultCutValue = 0.7*CLHEP::mm;
  // use pre-defined physics constructors
  // e.g. register standard EM physics using the pre-defined constructor
  // (includes constructions of all EM processes as well as the
  // corresponding particles)
  RegisterPhysics( new G4EmStandardPhysics() );
  // user might create their own constructor and register it
  // e.g. all physics processes having to do with protons (see below)
  RegisterPhysics( new YourProtonPhysics()   );
  // add more constructors to complete the physics
  ...
}
```

Modular Physics Lists : Standard EM Constructors

- **Some “standard” EM physics constructors**
  - G4EmStandardPhysics - default, used by ATLAS
  - G4EmStandardPhysics_option1 - for HEP, fast but not precise for sampling calorimeters, used by CMS
  - G4EmStandardPhysics_option2 - for HEP, fast but not precise for sampling calorimeters, used by LHCb
  - G4EmStandardPhysics_option3 - for medical and space science applications
  - G4EmStandardPhysics_option4 - most accurate EM models and settings
- **Many experimental, low-energy and DNA physics**
  - G4EmStandardPhysicsSS – used single scattering instead of multiple
- **G4EmExtraPhysics**
  - gamma, electro-nuclear, G4SynchrotronRadiation, rare EM processes
- **G4OpticalPhysics**

Check here for a detailed description of the available EM physics constructors
or :
http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/electromagnetic/index.html

Modular Physics Lists : Decay Physics Constructors

- **G4DecayPhysics**
  - main constructor for decay physics
  - defines standard list of particles
  - Included in all reference physics lists
- **G4RadioactiveDecayPhysics**
  - Defines radioactive decay of isotopes
  - Enable extra physics needed for radioactive decay
  - Should be registered by user

A realistic physics list can be found in basic example B3 :

» a modular physics list that includes “standard” EM physics and decay physics by using built in physics constructors;

» serves as a good starting point to construct your own physics list;

» add any other physics according to your needs

Reference Physics Lists

- **Adding hadronic physics is even more involved :**
  - for any hadronic process, the user might chose from several “models”;
  - choosing the most appropriate model for a given application requires significant experience
- **Pre-packaged physics lists:**
  - in order to help the users, the Geant4 toolkit provides pre-packaged physics lists according to some reference use cases;
  - these are “ready-to-use”, **complete physics lists** constructed by the experts
  - each pre-packaged reference physics list includes different combinations of EM and hadronic physics;
  - ~20 reference physics lists

The complete list of pre-packaged reference physics lists can be found here

or :

geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/reference_PL/index.html

How to use a Reference Physics List (RPL)

Example of FTFP_BERT :

In your main program:

```
#include "FTFP_BERT.hh"
...
int main( int argc, char** argv ) {
...
G4VModularPhysicsList* physicsList = new FTFP_BERT;
runManager->SetUserInitialization( physicsList );
…
}
```

Adding extra physics to a RPL

Adding radioactive decay :

```
#include "G4RadioactiveDecayPhysics.hh"
...
int main( int argc, char** argv ) {
...
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->RegisterPhysics( new G4RadioactiveDecayPhysics );
runManager->SetUserInitialization( physicsList );
…
}
```

Adding optical photon physics :

```
#include "G4OpticalPhysics.hh"
...
int main( int argc, char** argv ) {
...
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->RegisterPhysics( new G4OpticalPhysics );
runManager->SetUserInitialization( physicsList );
…
}
```

END