Physics List in Geant4

What is a Physics List (Is it really required ??)

Various Physics List Interface (usecase depends on the expertise)

- UserPhysicsList (Level1 : Write from scratch)
- ModularPhysicsList (Level 2 : More convenient to implement)
- Prepacked Physics List (Level 3 : Provided by Geant4 Toolkit)
  - Reference Physics List
  - How to extend them

What is a Physics List

Setting up the physics environment:

Particles and the associated physics processes

An object of a C++ class, which is responsible of defining following:

Particles used in simulation

Physics process associated with each particle

Is it really required ??

Yes, as it is one the 3 mandatory object that needs to be registered with RunManager

Eventually its just an interface to define physics of your application.

User is supposed to have a good idea of the physics behind the application

Removal of particles and process may lead to incomplete or unpredictable simulation results

Why not all the physics is included by default by Geant4

Will that be a good idea : Yes / No ? NO

There are different model that defines the same interaction.
Some are approximation and some are extremely precise.
Effect on computation time

You actually DON'T need all the particles

Eg. : Study of energy deposition by gamma radiation in NaI

One may not need optical photon and the associated process,
Unless you need to do its photon yield study
or PMT response study.

For these reasons Geant4 does not following integral physics approach rather it follows application based physics approach.

Application based physics approach

Provides
Independent Particles to be used
Independent physics components : Physics process

These components (processes) may be select in the custom physics list defined by user.

One important process that should always be there : Transportation

This must be assigned to all the **stable particles**

Depending upon the requirement one needs to chose different components.

Results in efficient simulation run.

Sometimes you may afford less accurate calculation, so you may get faster model for a particular interaction

Creating a Physics List

There are three ways

UserPhysicsList : Create from scratch using components (processes) and particles
available in Geant4

ModularPhysicsList : Again going to use existing components and particles but it provides
an easy to use interface.

Prepacked Physics List : Use the PhysicsList already existing in Geant4
(A good start point)

UserPhysicsList : Inheritance mechanism of C++ : G4VUserPhysicsList

Utilizes the Inheritance mechanism of C++

Base class : G4**V**UserPhysicsList

The most basic interface

User needs to specify all the particles that may be used generated during the lifetime of a run.

For each of these particles specify all the associated process
    Transportation needs to be attached to all the stable particles.

Not suitable for less experience users

But provides great flexibility.

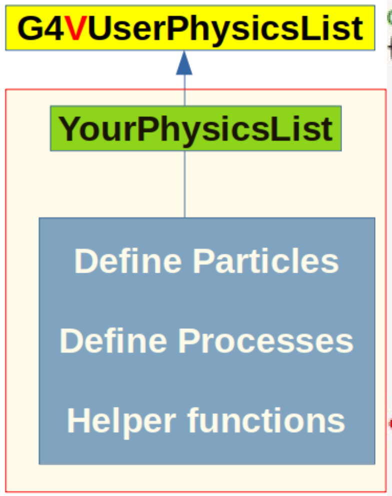
Hence recommended for advance users.

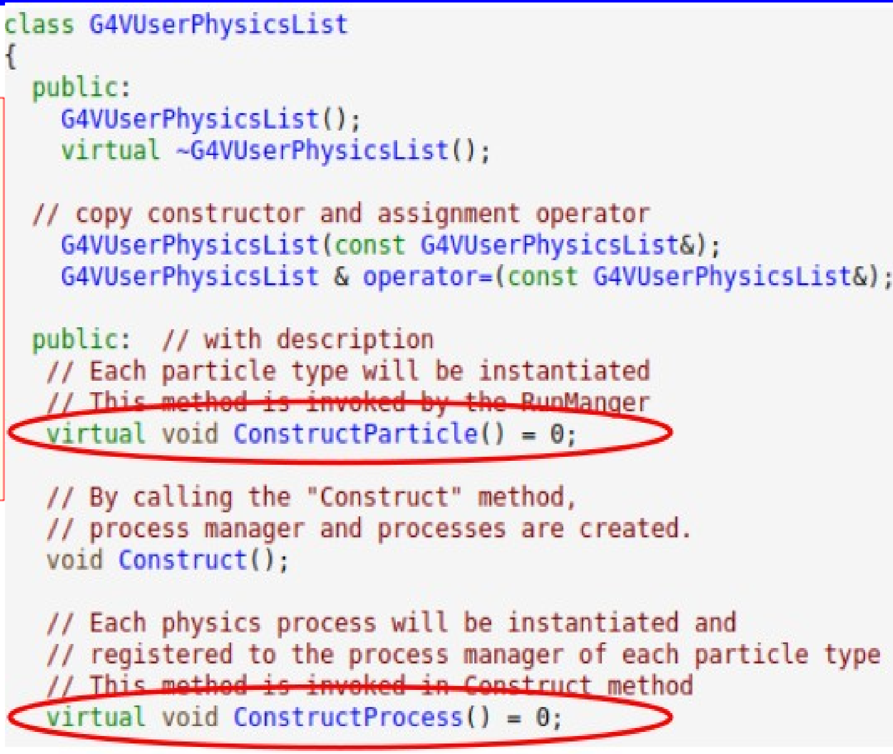
UserPhysicsList : Interface to define Physics List

G4VUserPhysicsList : Defines the interface for Geant4 Physics List.

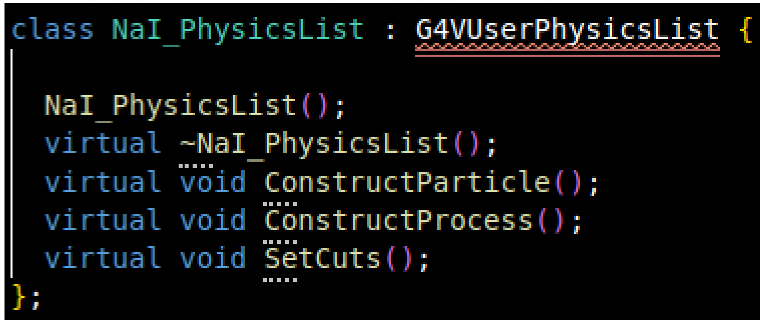
Recipe to implement you physics list

- Inherit the G4VUserPhysicsList in your physics list
- **Implement the 2 mandatory functions**
- **(Pure virtual functions in G4VPhysicsList)**
- **ConstructParticle()** responsible for creating all the particles required during simulation
- **ConstructProcess()** responsible for assigning the required processes with each particle type





```
class G4VUserPhysicsList
{
 public:
   G4VUserPhysicsList();
   virtual ~G4VUserPhysicsList();

 // copy constructor and assignment operator
   G4VUserPhysicsList(const G4VUserPhysicsList&);
   G4VUserPhysicsList & operator=(const G4VUserPhysicsList&);

 public:  // with description
  // Each particle type will be instantiated
  // This method is invoked by the RunManger
  virtual void ConstructParticle() = 0;

  // By calling the "Construct" method,
  // process manager and processes are created.
  void Construct();

  // Each physics process will be instantiated and
  // registered to the process manager of each particle type
  // This method is invoked in Construct method
  virtual void ConstructProcess() = 0;
```

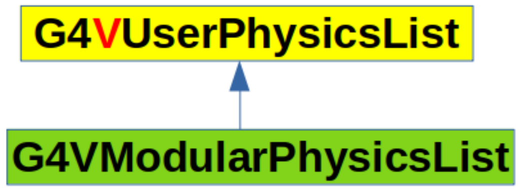


```
class NaI_PhysicsList : G4VUserPhysicsList {

  NaI_PhysicsList();
  virtual ~NaI_PhysicsList();
  virtual void ConstructParticle();
  virtual void ConstructProcess();
  virtual void SetCuts();
};
```

UserPhysicsList cont..

Add some snapshots

ModularPhysicsList : Inheritance mechanism of C++ : G4VModularPhysicsList



Utilizes the Inheritance mechanism of C++

Base class : G4VModularPhysicsList

G4VModularPhysicsList Class is inherited from G4VUserPhysics List

Different Processes related to particular physics are already attached to respective particles
Electromagnetic Physics: Attach all the particles which undergoes electromagnetic interactions to the corresponding process.

User don't have to worry for individual particles and the associated processes.

Provide more convenient way to create user define physics lists.

Automatically attached Transportation to all the constructed particles.

User may add more helper functions for debugging or to get additional information about the physics process and the associated particles.

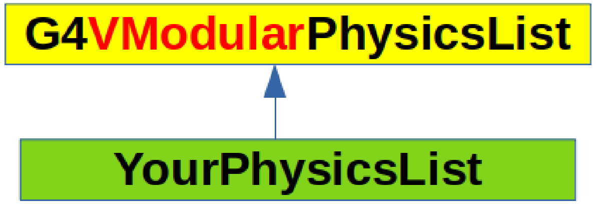
Defining a user defined physics list using G4VModularPhysicsList

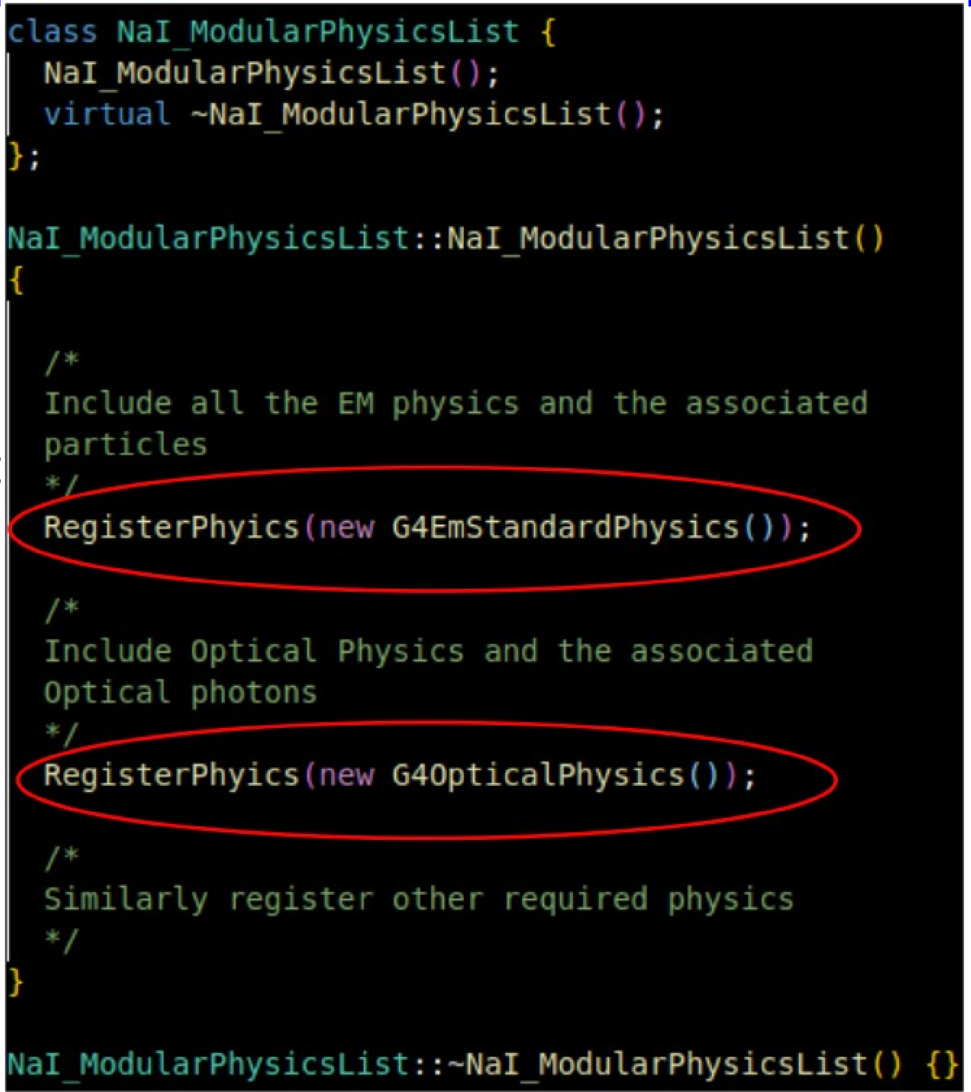
G4VModularPhysicsList : Defines the higher level interface for Geant4 Physics List.

Recipe to implement you physics list

Inherit the G4VModularPhysicsList in your physics list

Register the desired physics in the constructor c
your physics list.





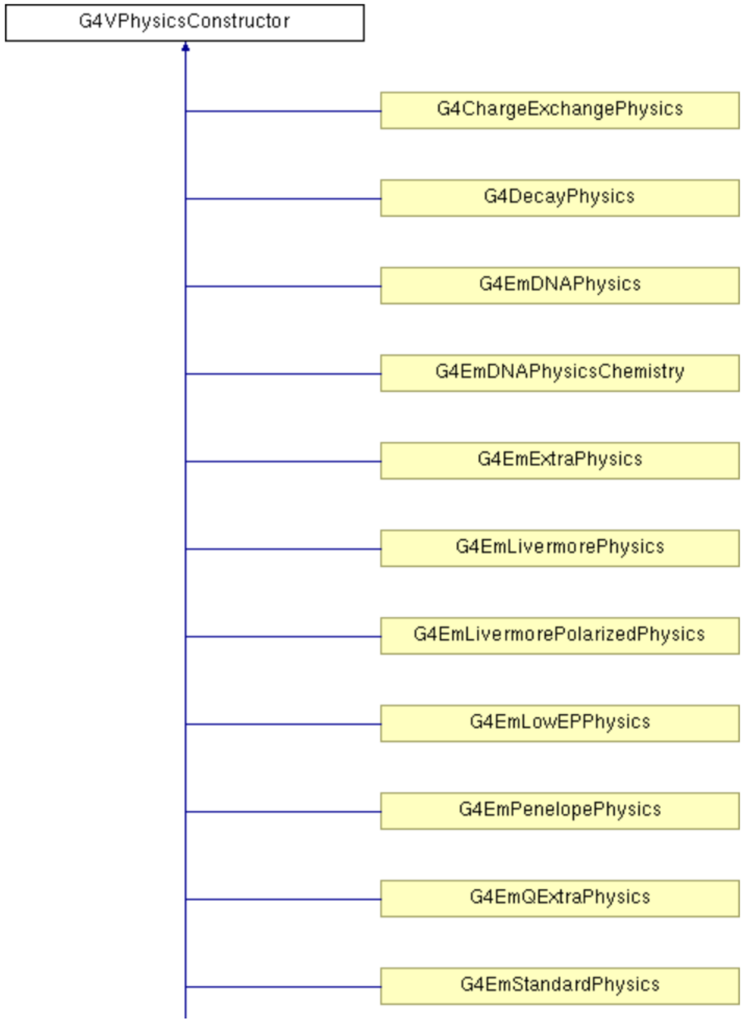
```
class NaI_ModularPhysicsList {
  NaI_ModularPhysicsList();
  virtual ~NaI_ModularPhysicsList();
};

NaI_ModularPhysicsList::NaI_ModularPhysicsList()
{

  /*
  Include all the EM physics and the associated
  particles
  */
  RegisterPhyics(new G4EmStandardPhysics());

  /*
  Include Optical Physics and the associated
  Optical photons
  */
  RegisterPhyics(new G4OpticalPhysics());

  /*
  Similarly register other required physics
  */
}

NaI_ModularPhysicsList::~NaI_ModularPhysicsList() {}
```

Existing Physics constructor required use ModularPhysicsList interface

What all Physics constructor are already defined.

https://apc.u-paris.fr/~franco/g4doxy/html/classG4VPhysicsConstructor.html

- **Some “standard” EM physics constructors:**
  - G4EmStandardPhysics – default
  - G4EmStandardPhysics_option1 - for HEP, fast but not precise settings
  - G4EmStandardPhysics_option2 - for HEP, experimental
  - G4EmStandardPhysics_option3 - for medical and space science applications
  - G4EmStandardPhysics_option4 - most accurate EM models and settings
- **Some hadronic physics constructors**
  - G4HadronElasticPhysics – default for hadron nuclear elastic for all hadrons
  - G4HadronElasticPhysicsHP – as above, but use HP for neutrons below 20 MeV
  - G4HadronPhysicsFTFP_BERT – hadron nucleus inelastic physics for all hadrons
  - G4IonPhysics – interactions of Ions
- **The complete list of constructors can be found in your toolkit:**
  - geant4/source/physics_lists/constructors/...
- **More information at:**
  - README files in geant4/source/physics_lists/constructors/..../README
  - http://cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html



Pre-Packaged PhysicsList : Inheritance NOT required

Pre-packaged physics list provides several advantages

Ready to use physics list : Just create an object of the existing pre-packaged class and inform the RunManager

Extremely easy to use, even for beginners.

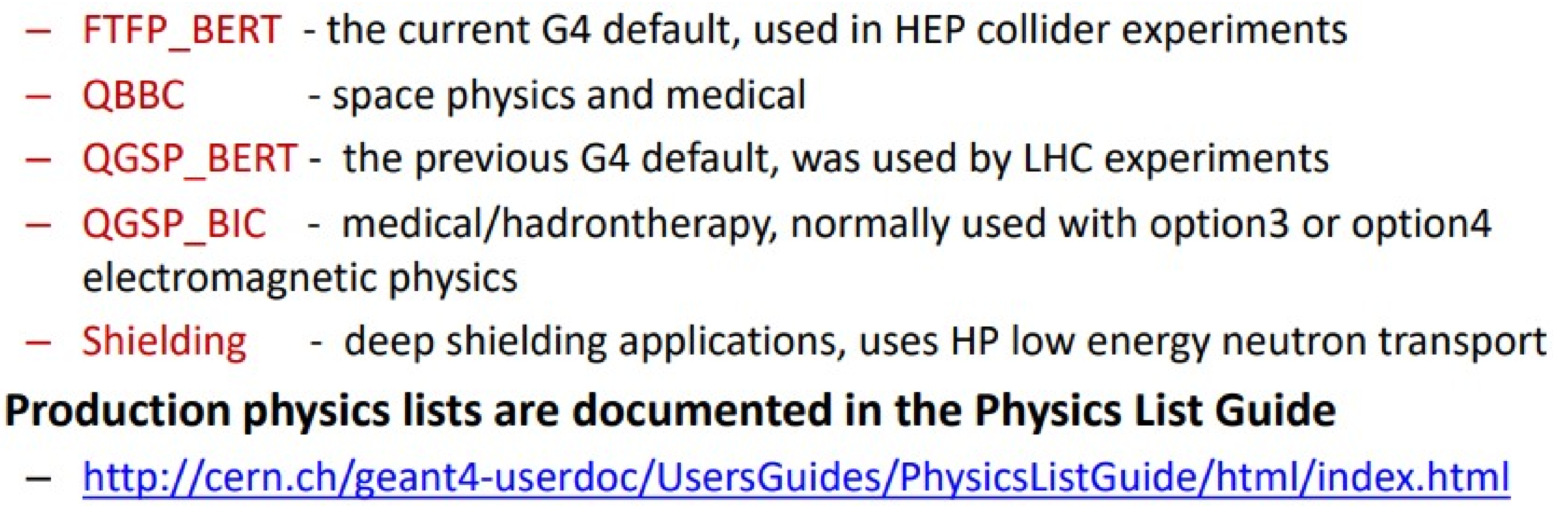
Created and maintained by experts, chances of error are extremely less.

Warning : Responsibility lies with the user. One has see carefully chose the physics list based on the his application.

Also the user is responsible to validate the chosen physics list for his application.

Pre-Packaged PhysicsList cont..

All the pre-packaged physics list are very well documented, and is used by large group of people



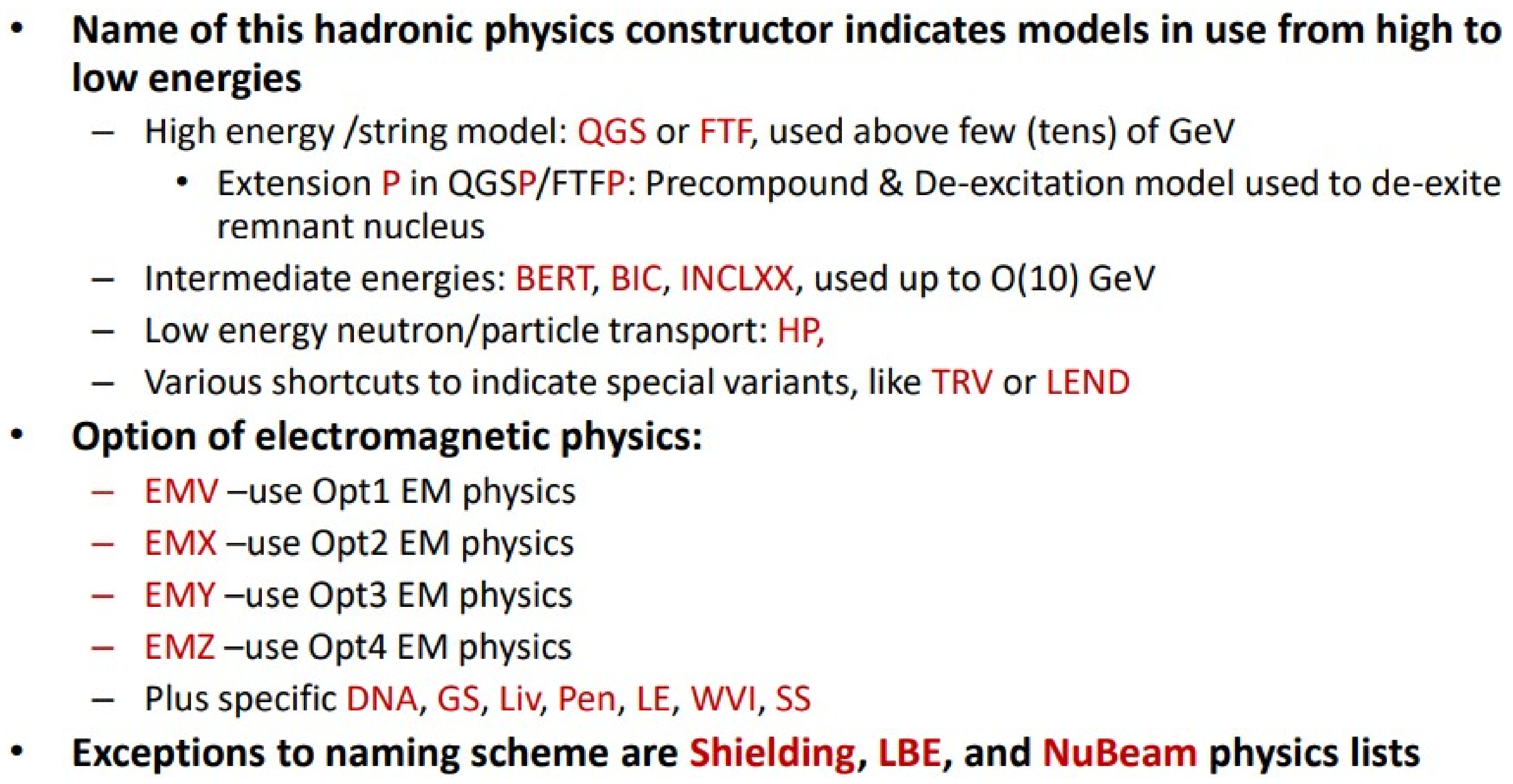
- FTFP_BERT - the current G4 default, used in HEP collider experiments
- QBBC - space physics and medical
- QGSP_BERT - the previous G4 default, was used by LHC experiments
- QGSP_BIC - medical/hadrontherapy, normally used with option3 or option4 electromagnetic physics
- Shielding - deep shielding applications, uses HP low energy neutron transport

Production physics lists are documented in the Physics List Guide

- http://cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

Naming conventions of various Pre-packaged Production Physics lists

Most of the physics lists follows name of Physics Constructor



- **Name of this hadronic physics constructor indicates models in use from high to low energies**
  - High energy /string model: QGS or FTF, used above few (tens) of GeV
    - Extension P in QGSP/FTFP: Precompound & De-excitation model used to de-exite remnant nucleus
  - Intermediate energies: BERT, BIC, INCLXX, used up to O(10) GeV
  - Low energy neutron/particle transport: HP,
  - Various shortcuts to indicate special variants, like TRV or LEND
- **Option of electromagnetic physics:**
  - EMV –use Opt1 EM physics
  - EMX –use Opt2 EM physics
  - EMY –use Opt3 EM physics
  - EMZ –use Opt4 EM physics
  - Plus specific DNA, GS, Liv, Pen, LE, WVI, SS
- **Exceptions to naming scheme are Shielding, LBE, and NuBeam physics lists**

Heading

Content

Conclusion

Physics list is one of the mandatory class used in Geant4 Simulation

To do a logical simulation all the required particles and process needs to be registered within the physics list.

Various interface exists to define you physics list

G4VUserPhysics : Provides the maximum level of flexibility but needs expertisze, hence \ can safely be used for simple experiment setup

G4VModularPhysicsList : Provides a convenient way to define you physics list, and is generally used for more complex physics problem.

Reference (pre-packaged) physics lists makes the life easier and is a good starting point for beginners.

Physics lists must be selected with extreme care to get the meaningful results from the simulations.

Assignment

You have to write the simulation code to simulate the interaction of 662 keV gammas with NaI crystal.

Specification of NaI Crystal : Cylinder of 2 inch diameter, 2 inch height

Energy resolution : 40 keV

Particle source : gammas of 662 keV

Physics List creation : (a) Pre-packaged
(b) ModularPhysicsList
(c) User Physics List.

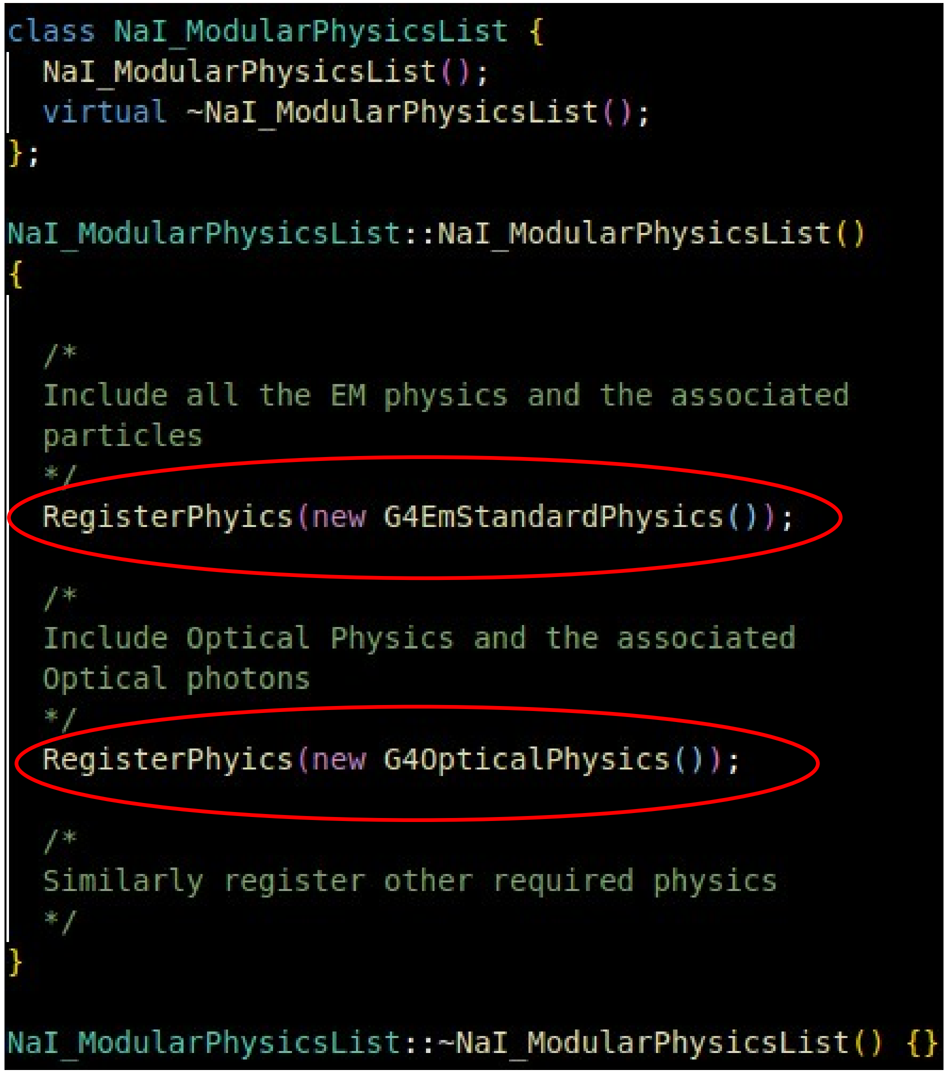
Final outcome : Should get a gaussian peak at 662 keV.
spectras from all the three types of physics lists should match

Heading

Content

Heading

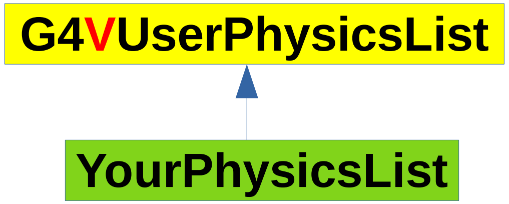
Content

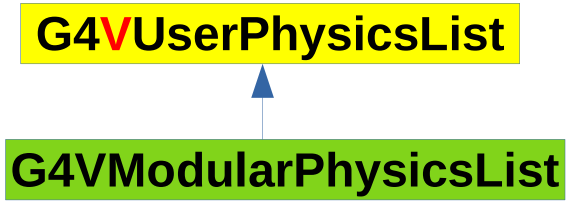


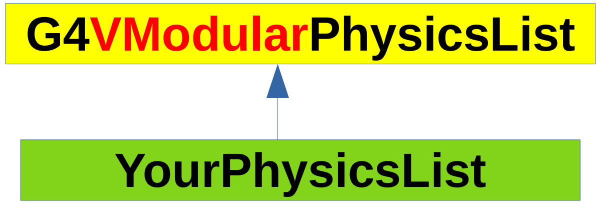
```
class NaI_ModularPhysicsList {
  NaI_ModularPhysicsList();
  virtual ~NaI_ModularPhysicsList();
};

NaI_ModularPhysicsList::NaI_ModularPhysicsList()
{

  /*
  Include all the EM physics and the associated
  particles
  */
  RegisterPhyics(new G4EmStandardPhysics());

  /*
  Include Optical Physics and the associated
  Optical photons
  */
  RegisterPhyics(new G4OpticalPhysics());

  /*
  Similarly register other required physics
  */
}

NaI_ModularPhysicsList::~NaI_ModularPhysicsList() {}
```

Heading

Content



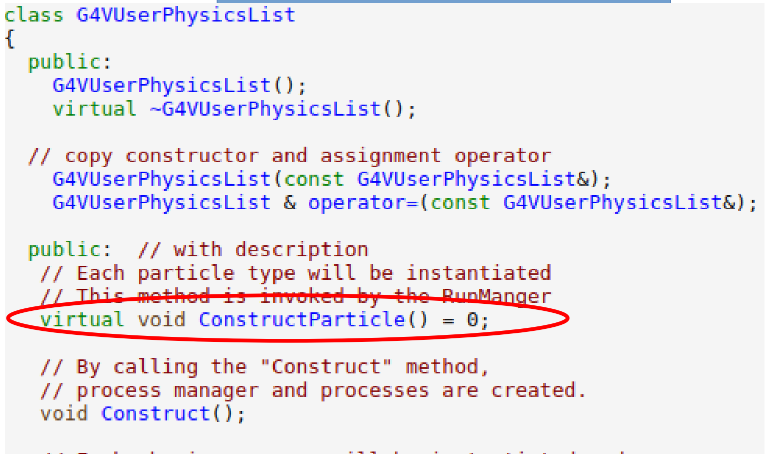




Define Particles

Define Processes

Helper functions



```
class G4VUserPhysicsList
{
  public:
    G4VUserPhysicsList();
    virtual ~G4VUserPhysicsList();

  // copy constructor and assignment operator
    G4VUserPhysicsList(const G4VUserPhysicsList&);
    G4VUserPhysicsList & operator=(const G4VUserPhysicsList&);

  public:  // with description
   // Each particle type will be instantiated
   // This method is invoked by the RunManger
   virtual void ConstructParticle() = 0;

   // By calling the "Construct" method,
   // process manager and processes are created.
   void Construct();
```