Physics Lists

Dennis Wright (SLAC)

Geant4 Tutorial
Bordeaux
3-5 November 2005

Outline

- Physics list introduction
- Step-by-step example for building a simple physics list
- Modular physics lists
- Best guess physics lists

Physics List Introduction

- Geant4 requires the user to decide:
  - which particles are required for a given application
  - which physics processes are to be assigned to each particle
  - what the secondary particle production cuts are (electromagnetic processes only)
- All this is done in the Physics List
- Must be invoked in the user's main() after detector construction and before generator action:

```
int main()  {
    G4RunManager* runMan = new G4RunManager;
    runMan->SetUserInitialization(new MyDetectorConstruction);
    runMan->SetUserInitialization(new MyPhysicsList);
    runMan->SetUserAction(new MyPrimaryGeneratorAction);
```

Physics List Class

- All physics lists are derived from G4VUserPhysicsList
- It has three methods the user must implement:
  - ConstructParticle()
  - ConstructProcess()
  - SetCuts()
- Other methods provided:
  - AddTransportation() - required or else particles go nowhere
  - DumpList() - print list of registered particles
  - DumpCutValues() - print list of range cuts for particles
  - ....

How to Build a Physics List – Step 1

- Choose the physics
  - which particles at which energies
  - what physics processes are important
- Our example: space radiation environment
  - cosmic rays: 85% protons, 14% alpha, + C, N, O; most effects in 100 MeV to 20 GeV range
  - Van Allen belts: electrons up to 10 MeV, protons ~10 – 100 MeV
  - solar particles: protons up to a few GeV
  - diffuse gamma background: 0.1 – 200 MeV
  - need electromagnetic, hadronic and photo-nuclear processes

How to Build a Physics List – Step 2

- Implement ConstructParticle method in your physics list
  - void MyPhysicsList::ConstructParticle()

```
{
  G4Gamma::GammaDefinition();
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
  G4Proton::ProtonDefinition();
  G4Neutron::NeutronDefinition();
  G4Alpha::AlphaDefinition();
  G4GenericIon::GenericIonDefintion();
}
```

- Or use the ConstructParticle method of the classes:
  - G4LeptonConstructor
  - G4MesonConstructor
  - G4BaryonConstructor

How to Build a Physics List – Step 3a

- Implement ConstructProcess method in your physics list
  - void MyPhysicsList::ConstructProcess()

```
{
    ConstructEM();
    ConstructHadronic();
    ConstructPhotoNuclear();
}
```

- Here we have divided the processes in our example into categories for convenience - now implement each one
- Before implementing ConstructEM(), decide which processes are needed:
  - gamma conversion, photo-electric effect, Compton scattering,
  - multiple Coulomb scattering, ionization, bremsstrahlung, positron annihilation

How to Build a Physics List – Step 3a

- Can use "standard" or "low-energy" processes
  - standard generally faster and cover higher energies
  - low energy more accurate at low incident energies where atomic shell effects are important
- For this example we choose "standard" processes
- Hence the work for ConstructEM() is already done
  - Look at novice example N03
  - Copy from ExN03PhysicsList::ConstructEM()
- See advanced examples for the use of low energy processes

How to Build a Physics List – Step 3b

- Now implement ConstructPhotoNuclear() method
- For hadronic and photo-nuclear reactions we not only need to choose processes, but also models
  - for photo-nuclear we choose G4PhotoNuclearProcess and G4GammaNuclearReactionModel
  - this was easy because there is only one photo-nuclear process available
  - also there is only one model available with which to implement this process for gamma energies below 200 MeV
  - we would also need to select a cross section data set, but this comes by default with the process (in most cases)

How to Build a Physics List – Step 3b

- physics list code for diffuse gamma background (photo-nuclear):
  - void MyPhysicsList::ConstructPhotoNuclear()

```
{
    G4ParticleDefinition* photon = G4Gamma::Gamma();
    G4ProcessManager* pman = photon->GetProcessManager();

    // Inelastic photon scattering

    G4PhotoNuclearProcess* process = new G4PhotoNuclearProcess;
    G4GammaNuclearReaction* model =
                                  new G4GammaNuclearReaction;
    process->RegisterMe(model);
    pman->AddDiscreteProcess(process);
}
```

How to Build a Physics List – Step 3c

- Now implement ConstructHadronic() method
- Now there are more process and models to choose from
  - need elastic and inelastic hadron scattering from nuclei
- For protons choose:
  - G4HadronElasticProcess with G4LElastic model
  - G4ProtonInelasticProcess with G4LEProtonInelastic model
- For alphas choose:
  - G4HadronElasticProcess with G4LElastic model
  - G4AlphaInelasticProcess with G4LEAlphaInelastic model
- All of the above have default cross sections

How to Build a Physics List – Step 3c

- physics list code for cosmic rays (hadronic):
  - void MyPhysicsList::ConstructHadronic()

```
{
   G4ParticleDefinition* proton = G4Proton::Proton();
   G4ProcessManager* pman = proton->GetProcessManager();

   // Elastic scattering

   G4HadronElasticProcess* eproc = new G4HadronElasticProcess;
   G4LElastic* emodel = new G4LElastic;
   eproc->RegisterMe(emodel);
   pman->AddDiscreteProcess(eproc);

   // Inelastic scattering

   G4ProtonInelasticProcess* iproc = new G4ProtonInelasticProcess;
```

How to Build a Physics List – Step 3c

- physics list code for cosmic rays (continued):
  - ```
    G4LEProtonInelastic* imodel = new G4LEProtonInelastic;
    iproc->RegisterMe(imodel);
    pman->AddDiscreteProcess(iproc);

    // alpha

    G4ParticleDefinition* alpha = G4Alpha::Alpha();
    G4ProcessManager* pman = alpha->GetProcessManager();

    // Elastic scattering. Same model as proton (G4LElastic)

    G4HadronElasticProcess* aproc = new G4HadronElasticProcess;
    aproc->RegisterMe(emodel);
    pman->DiscreteProcess(paroc);
    ```
    ```

How to Build a Physics List – Step 3c

- physics list code for cosmic rays (continued):
  - ```
    // Inelastic scattering. Not the same model as for proton.

    G4AlphaInelasticProcess* aiproc = new G4AlphaInelasticProcess;
    G4LEAlphaInelastic* aimodel = new G4LEAlphaInelastic;
    aiproc->RegisterMe(aimodel);
    pman->AddDiscreteProcess(aproc);
    }
    ```
  - exercise for the student:
    - extend physics list to include neutrons, pions, and kaons
    - continue to use LEP models for elastic and inelastic scattering
    - extend physics list to high energies by using HEP models
    ```

Modular Physics Lists

- As physics requirements become more realistic, the physics list gets much longer
  - it may be useful to break it up into smaller files
  - you may want to define subsets of physics processes which correspond to a given particle or type of interaction
  - you may want to switch on/off a set of processes
- Modular physics lists allow this
  - derive your physics list from class G4VModularPhysicsList
  - create "sub-physics lists" or modules by deriving from G4VPhysicsConstructor
  - register sub-physics list to main physics list:
    - RegisterPhysics(G4VPhysicsConstuctor* fPhysCons)

Organizing A Modular Physics List

- choose physics domains:
  - physics of protons, physics of gammas, etc.
- example:
  - ```
    MyModPhysList::MyModPhysList() : G4VModularPhysicsList()
    {
       defaultCutValue = 1.0*mm;

       RegisterPhysics(new GammaPhysics(“gamma”));
       RegisterPhysics(new LeptonPhysics(“lepton”));
       RegisterPhysics(new HadronPhysics(“hadron”));
       RegisterPhysics(new DecayPhysics(“decay”));
    }
    //Set cut values for gamma and lepton processes

    void MyModPhysList::SetCuts()
    {  SetCutsWithDefault();   }  // use default value above
    ```
    ```

Organizing A Modular Physics List

- now write the individual physics constructors
- sample header:
  - ```
    class LeptonPhysics : public G4VPhysicsConstructor
    {
       public:
         LeptonPhysics(const G4String& name = “lepton”);
         virtual ~LeptonPhysics();

         virtual void ConstructParticle();
         virtual ConstructProcess();
    }
    ```
- For each physics constructor, particles and processes are constructed just as in the non-modular case
- AddTransportation method called automatically in modular lists
    ```

Best Guess Physics Lists

- Geant4 provides a set of already-written physics lists which can be used for a number of applications
- These lists were developed as a “best guess” of the physics required for a given use case
  - application areas include high energy physics, medical, radiation protection
  - written as modular physics lists
- They are a good starting point, but the user should always validate a chosen list to make sure it does the right thing
- To use, first build the physics list libraries (parallel to Geant4 source directory), then invoke the physics list in your main():
  - runManager->SetUserInitialization(new prebuiltPhysList);

Summary

- Physics lists are where the user defines all the particles and processes required for a given application
- Users must take care to include all the important particles, processes, models and cross sections
- The user has three choices:
  - develop a “simple” physics list derived from G4VUserPhysicsList in which all particles and processes are defined
  - develop a modular physics list derived from G4VModularPhysicsList in which particle and process definition can be grouped according to a particular subset of the relevant physics
  - use the already-written physics lists provided along with the Geant4 source code