Physics I: Physics List

Shigeyuki Tajima
Duke/NCCU and TUNL

Slides courtesy of SLAC GEANT4 Team
Special thanks to Dennis Wright
Reminder

There are three mandatory classes that users must implement.

- G4VUserDetectorConstruction
- G4VUserPhysicsList
- G4VUserPrimaryGeneratorAction
Outline

• Introduction
  ▪ What is a physics list and why do we need one?

• The G4VUserPhysicsList class
  ▪ What you need to begin

• Modular physics lists
  ▪ A more sophisticated way to go

• Pre-packaged physics lists
  ▪ Leave the driving to us
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 could cause errors or poor simulation
Why Do We Need a Physics List?

• Physics is physics – shouldn't Geant4 provide, as a default, a complete set of physics that everyone can use?

• No:
  - 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 want multi-GeV physics
Why Do We Need a Physics List?

- For this reason Geant4 takes an atomistic, rather than an integral approach to physics
  - 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
- Exceptions:
  - a few electromagnetic processes must be used together
  - future processes involving interference of electromagnetic and strong interactions may require coupling as well
Physics Processes Provided by Geant4

- **EM physics**
  - “standard” processes valid from ~ 1 keV to ~ PeV
  - “low-energy” valid from 250 eV to ~ PeV
  - optical photons

- **Weak physics**
  - decay of subatomic particles
  - radioactive decay of nuclei

- **Hadronic physics**
  - pure hadronic processes valid from 0 to ~100 TeV
  - $\gamma$, $\mu$-nuclear valid from 10 MeV to ~TeV

- **Parameterized or “fast simulation” physics**
G4VUserPhysicsList

- All physics lists must derive from this class
  - and then be registered with the run manager

- In our example:
  ```
  class BeamTestPhysicsList: public G4VUserPhysicsList
  {
    public:
      BeamTestPhysicsList();
      ~BeamTestPhysicsList();

    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();
  };
  ```

- User must implement the methods ConstructParticle, ConstructProcess and SetCuts
G4VUserPhysicsList: Required Methods

- **ConstructParticle()** - choose the particles you need in your simulation and define all of them here

- **ConstructProcess()** - for each particle, assign all the physics processes important in your simulation
  - What's a process?
  - => a class that defines how a particle should interact with matter (it's where the physics is!)
  - more on this later

- **SetCuts()** - set the range cuts for secondary production
  - What's a range cut?
  - => essentially a low energy limit on particle production
  - more on this later

see the next several slides for details about these methods
ConstructParticle()

void BeamTestPhysicsList::ConstructParticle()
{
    G4BaryonConstructor* baryonConstructor = new
        G4BaryonConstructor();
    baryonConstructor->ConstructParticle();
    delete baryonConstructor;

    G4BosonConstructor* bosonConstructor = new
        G4BosonConstructor();
    bosonConstructor->ConstructParticle();
    delete bosonConstructor;
    ....
    ....
}
void BeamTestPhysicsList::ConstructParticle()
{
    G4Electron::ElectronDefinition();
    G4Proton::ProtonDefinition();
    G4Neutron::NeutronDefinition();
    G4Gamma::GammaDefinition();
    ....
    ....
}
ConstructProcess()

```cpp
void BeamTestPhysicsList::ConstructProcess()
{
    AddTransportation();
    // method provided by G4VUserPhysicsList
    // assigned transportation process to all particles
    // defined in ConstructParticle()

    ConstructEM();
    // method may be defined by user (for convenience)
    // put electromagnetic physics here

    ConstructGeneral();
    // method may be defined by user (for convenience)
}
```

see next slides for ConstructEM and ConstructGeneral
ConstructEM()

```c++
void BeamTestPhysicsList::ConstructEM()
{
    theParticleIterator->reset();
    while( (*theParticleIterator)() ) {
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager =
            particle->GetProcessManager();
        G4String particleName = particle->GetParticleName();

        if (particleName == "gamma") {
            pmanager->AddDiscreteProcess(new
                G4GammaConversion());
            ...
        }
    }
    ...
}
```

Other available processes:
- G4PhotoElectricEffect,
- G4ComptonScattering,
ConstructGeneral()

void BeamTestPhysicsList::ConstructGeneral()
{

    // Add decay process
    G4Decay* theDecayProcess = new G4Decay();
    theParticleIterator->reset();
    while( (*theParticleIterator)() ) {
        G4ParticleDefinition* particle = theParticleIterator->value();
        G4ProcessManager* pmanager = particle->GetProcessManager();
        if (theDecayProcess->IsApplicable(*particle) ) {
            pmanager->AddProcess(theDecayProcess);
            pmanager->SetProcessOrdering(theDecayProcess, idxPostStep);
            pmanager->SetProcessOrdering(theDecayProcess, idxAtRest); } } }
void BeamTestPhysicsList::SetCuts()
{
    defaultCutValue = 1.0*mm;
    SetCutValue(defaultCutValue, "gamma");
    SetCutValue(defaultCutValue, "e-" );
    SetCutValue(defaultCutValue, "e+");
    //
    // These are all the production cut values you need to set
    // - not required for any other particle
}

G4VModularPhysicsList

• The physics list in our example is relatively simple

• A realistic physics list is likely to have many more physics processes
  ▪ such a list can become quite long, complicated and hard to maintain
  ▪ try a modular physics list instead

• Features of G4VModularPhysicsList
  ▪ derived from G4VUserPhysicsList
  ▪ AddTransportation() automatically called for all registered particles
  ▪ Allows you to define “physics modules”: EM physics, hadronic physics, optical physics, etc.
A Simple G4VModularPhysicsList

- Constructor:
  ```cpp
  MyModPhysList::MyModPhysList() : G4VModularPhysicsList()
  {
    defaultCutValue = 1.0*mm;
    RegisterPhysics( new ProtonPhysics() );  // all physics processes having to do with protons
    RegisterPhysics( new ElectronPhysics() );
    RegisterPhysics( new DecayPhysics() );   // physics of unstable particles
  }
  ```

- Set Cuts:
  ```cpp
  void MyModPhysList::SetCuts()
  {
    SetCutsWithDefault() ;
  }
  ```
Physics Constructors

- Allow you to group particle and process construction according to physics domains

```cpp
class ProtonPhysics : public G4VPhysicsConstructor {
    public:
        ProtonPhysics(const G4String& name = "proton");
        virtual ~ProtonPhysics();

        virtual void ConstructParticle();
        // easy – only one particle to build in this case

        virtual void ConstructProcess();
        // put here all the processes a proton can have
}
```
Pre-packaged Physics Lists (1)

- Our example deals mainly with electromagnetic physics

- A complete and realistic EM physics list can be found in novice example N03
  - good starting point
  - add to it according to your needs

- Adding hadronic physics is more involved
  - for any one hadronic process, user may choose from several hadronic models to choose from
  - choosing the right models for your application requires care
  - to make things easier, hadronic physics lists are now provided according to some use cases

Slide from SLAC Geant4 tutorial course in ‘06
Pre-packaged Physics Lists (2)

- Referred to as “hadronic physics lists” but include electromagnetic physics from example N03

- Can be found on the Geant4 web page at

- Caveats:
  - these lists are provided as a “best guess” of the physics needed in a given case
  - The user is responsible for validating the physics for his own application and adding (or subtracting) the appropriate physics
    - “Trust, but verify.”
  - they are intended as starting points or templates
Summary

• All the particles, physics processes, and production cuts needed for an application must go into a physics list

• Two kinds of physics list classes are available for users to derive from
  - G4VUserPhysicsList – for relatively simple physics lists
  - G4VModularPhysicsList – for detailed physics lists

• Some pre-packaged physics lists are provided by Geant4 as starting points for users
  - electromagnetic physics lists
  - hadronic physics lists

• Care is required by user in choosing the right physics to use