Monte Carlo 4

2006/12/25

Physics Process

- We have to specify all particles and physics processes which will be used in our simulation
- Edit PhysicsList.cc and PhysicsList.hh
- We must create our own class derived from G4VuserPhysicsList
  - ConstructParticle() : construction of particles
  - ConstructProcess(): construct processes and register them to particles
  - SetCuts() : setting a cut value in range to all particles

Particle definition

- Geant4 provides various types of particles used in simulations
  - Ordinary particles (electron, proton, gamma etc.)
  - Resonant particles with very shot life (vector mesons, delta baryons etc.)
  - Nuclei (deuteron, alpha, heavy ions etc.)
  - Quarks, di-quarks, and gluon
- There are 6 major particle categories
  - Lepton
  - Meson
  - Baryon
  - Boson
  - Shortlived
  - ion

G4ParticleDefinition

```
G4KaonMinus* G4KaonMinus::Definition()
{
  if (theInstance !=0) return theInstance;
  const G4String name = "kaon-";
  // search in particle table]
  G4ParticleTable* pTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* anInstance = pTable->FindParticle(name);
  if (anInstance ==0)
  {

   anInstance = new G4ParticleDefinition(
                name,    0.493677*GeV,   5.315e-14*MeV,    -1.*eplus,
                  0,             -1,             0,
                  1,             -1,             0,
              "meson",              0,            0,       -321,
                false,       12.3785*ns,          NULL,
                false,       "kaon");

 //create Decay Table
  G4DecayTable* table = new G4DecayTable();

 // create decay channels
  G4VDecayChannel** mode = new G4VDecayChannel*[6];
  // kaon- -> mu- + anti_nu_mu
  mode[0] = new G4PhaseSpaceDecayChannel("kaon-",0.6339,2,"mu-","anti_nu_mu");
  // kaon- -> pi- + pi0
  mode[1] = new G4PhaseSpaceDecayChannel("kaon-",0.2103,2,"pi-","pi0");
  // kaon- -> pi- + pi+ + pi-
  mode[2] = new G4PhaseSpaceDecayChannel("kaon-",0.0559,3,"pi-","pi+","pi-");
  // kaon- -> pi- + pi0 + pi0
  mode[3] = new G4PhaseSpaceDecayChannel("kaon-",0.01757,3,"pi-","pi0","pi0");
  // のこりは省略

  for (G4int index=0; index <6; index++ ) table->Insert(mode[index]);
  delete [] mode;

   anInstance->SetDecayTable(table);
  }
  theInstance = reinterpret_cast<G4KaonMinus*>(anInstance);
  return theInstance;
}
```

- Properties to characterize individual particles
  - Name
  - Mass
  - Width
  - Spin, parity
  - Decay mode (decay table)
- More than 100 types of particles are already provided by default.
- In normal application, users do not need to define particles for themselves.

Construct particles

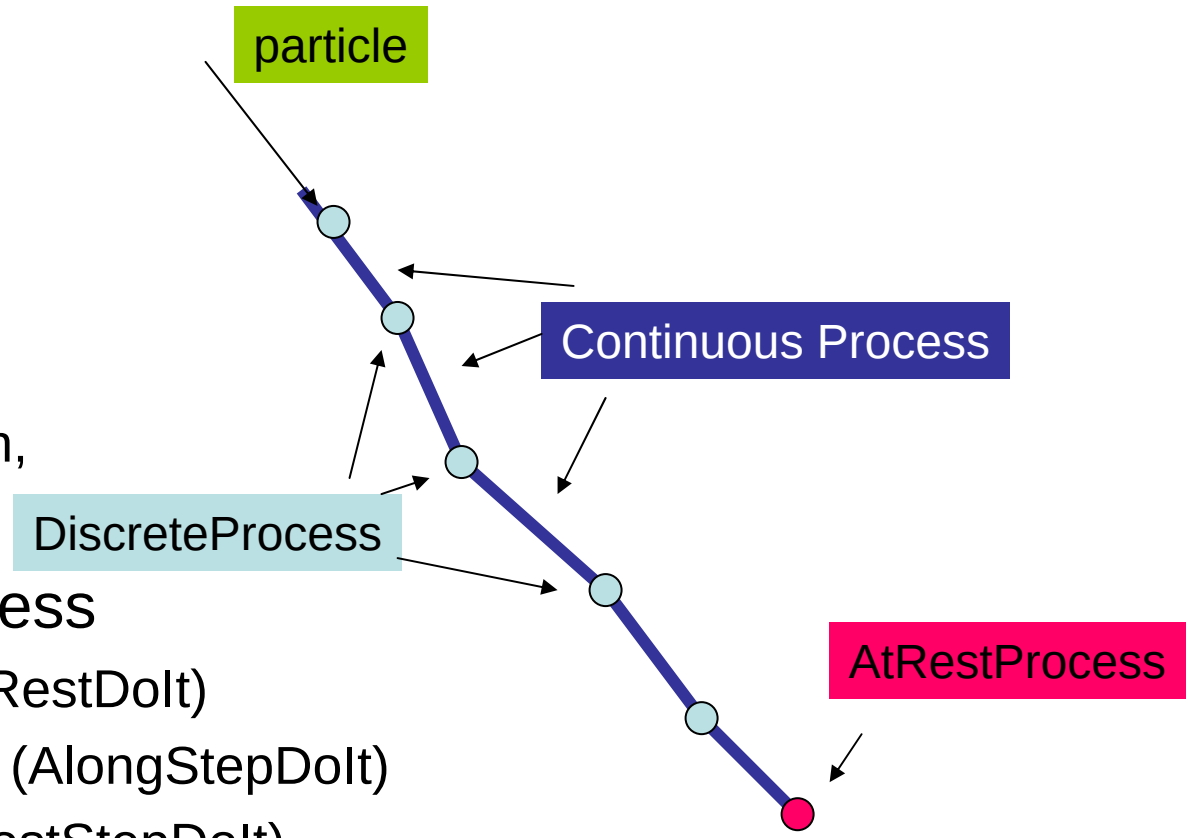
- In ConstructParticle() method, you should call Definition functions for all the particles you want. This ensures that objects of these particles will be created.

```
void ExN01PhysicsList::ConstructParticle()
{
  G4Geantino::GeantinoDefinition();
  G4Proton::ProtonDefinition();                Proton
  G4PionPlus::PionPlusDefinition();            π+
  G4PionMinus::PionMinusDefinition();          π-
  G4PionZero::PionZeroDefinition();            π0
}
```

- When you produce K- in PrimaryGeneratorAction and add decay process, you have to also register decay products of K-
- If you add electromagnetic process, e-,e+,gamma are produced. Therefore you must also register these particles

Physical Processes

- Physics processes describe how particles interact with materials. Geant4 provides seven major categories of processes
  - Electromagnetic,
  - Hadronic,
  - Transportation,
  - Decay,
  - Optical,
  - Photolepton_hadron,
  - parameterisation.
- Three kinds of process
  - AtRestProcess (AtRestDoIt)
  - ContinuousProcess (AlongStepDoIt)
  - DiscreteProcess (PostStepDoIt)



Specifying Physics Processes

```
void ExN00PhysicsList::ConstructProcess()
{
  AddTransportation();
  ConstructEM();
}
void ExN00PhysicsList::ConstructEM()
{
  theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle =
                          theParticleIterator->value();
    G4ProcessManager* pmanager =
                          particle->GetProcessManager();
    G4String particleName =
                          particle->GetParticleName();

    if (particleName == "gamma") {
    // gamma
      pmanager->AddDiscreteProcess(
                            new G4GammaConversion());
      pmanager->AddDiscreteProcess(
                           new G4ComptonScattering());
      pmanager->AddDiscreteProcess(
                           new G4PhotoElectricEffect());
    }
}
```

- In the ConstructProcess() method, physics processes should be created and registered with each particle's instance of G4ProcessManager
- The AddTransportation() method is provided to register the G4Transportation class with all particles classes.
- The G4ProcessManager class contains a list of processes that a particle can undertake.
  - The order of invocation
  - Kind of DoIt method
    - AtRestProcess
    - ContinuousProcess
    - DiscreteProcess

Managing Processes

ConstructEM()の中

```
//electron
   G4VProcess* theeminusMultipleScattering = new G4MultipleScattering();
   G4VProcess* theeminusIonisation         = new G4eIonisation();
   G4VProcess* theeminusBremsstrahlung     = new G4eBremsstrahlung();
   //
   // add processes
   pmanager->AddProcess(theeminusMultipleScattering);
   pmanager->AddProcess(theeminusIonisation);
   pmanager->AddProcess(theeminusBremsstrahlung);
   //
   // set ordering for AlongStepDoIt
   pmanager->SetProcessOrdering(theeminusMultipleScattering, idxAlongStep,1);
   pmanager->SetProcessOrdering(theeminusIonisation,         idxAlongStep,2);
   pmanager->SetProcessOrdering(theeminusBremsstrahlung,     idxAlongStep,3);

   //
   // set ordering for PostStepDoIt
   pmanager->SetProcessOrdering(theeminusMultipleScattering, idxPostStep,1);
   pmanager->SetProcessOrdering(theeminusIonisation,         idxPostStep,2);
   pmanager->SetProcessOrdering(theeminusBremsstrahlung,     idxPostStep,3);
```

- In order to validate processes, processes should be registered with the particle's G4ProcessManager.
- Processes are added with AddProcess() method
- The order of invocation
- Kind of DoIt method
  - AtRestProcess
  - ContinuousProcess
  - DiscreteProcess

Decay process

```
#include "G4Decay.hh"

void ExN00PhysicsList::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);
      // set ordering for PostStepDoIt and AtRestDoIt
      pmanager ->SetProcessOrdering(theDecayProcess, idxPostStep);
      pmanager ->SetProcessOrdering(theDecayProcess, idxAtRest);
    }
  }
}
```