Środowisko programistyczne GEANT4

Leszek Adamczyk

Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza

Wykłady w semestrze zimowym 2013/2014

Procesy fizyczne : G4UserPhysicsList

- Klasa wyprowadzona z klasy G4UserPhysicsList służy do definiowania:
  - listy możliwych cząstek wtórnych;
  - aktywnych procesów fizycznych oddziaływania cząstek z materią;
  - warunków na produkcje cząstek wtórnych;
- Użytkownik ma pełną swobodę w budowie otoczenia fizycznego projektu:
  - jakie cząstki maja być produkowane?
  - jakim oddziaływaniom mają podlegać?
  - jaka jest minimalna energia cząstek wtórnych?
- Użytkownik musi rozumieć wymagania fizyczne stawiane przed projektem:
  - pominięcie jakiś cząstek lub procesów może skutkować błędami i niepoprawną symulacją;
  - nadmiar procesów i/lub cząstek wtórnych może istotnie spowolnić symulacje;

Procesy fizyczne : G4UserPhysicsList

Fizyka jest jedna. GEANT4 powinien dostarczyć pełną listę cząstek/procesów którą wszyscy powinni używać w symulacji (?)

- takiej listy nie ma;
- istnieje wiele alternatywnych modeli i przybliżeń (głównie w oddziaływaniach hadronów)
- w wielu sytuacjach szybkość symulacji jest istotna, użytkownik może preferować mniej dokładne ale szybsze aplikacje
- żadna dziedzina zastosowań nie wymaga wszystkich dostępnych procesów i cząstek oferowanych przez GEANT4 (fizyka medyczna, przestrzeni kosmicznej, jądrowa, fizyka wysokich energii, ...)

Procesy fizyczne : G4UserPhysicsList

Z tego powodu w GEANT4 wybrano atomistyczne podejście do symulacji oddziaływań cząstek:

- dysponujemy szeregiem składników niezależnych jeden od drugiego z których tworzymy swoje własne środowisko fizyczne podobnie jak buduje się geometrię detektora.
- Wyjątki:
  - kilka procesów elektromagnetycznych musi być używanych wspólnie;
  - w przyszłych wersjach będzie dodana interferencja między różnymi procesami

Procesy fizyczne : G4UserPhysicsList

- Należy wyprowadzić z klasy bazowej G4VUserPhysicsList klasę pochodną implementując funkcje:
  - ConstructParticle()
    aktywuje wszystkie potrzebne cząstki;
  - ConstructProcess()
    aktywuje wymagane procesy ich oddziaływania z materią;
  - SetCuts()
    określa warunki produkcji cząstek wtórnych;

Cząstki

- GEANT zawiera definicje większości potrzebnych cząstek.
- Każda cząstka reprezentowana jest przez klasę wyprowadzoną z klasy G4ParticleDefinition
- leptony:

  $e,\ \mu,\ \tau,\ \nu_e,\ \nu_\mu,\ \nu_\tau$
- hadrony:
  - mezony (qq):
    $\pi, K, J/\psi, \eta, B, D$
  - bariony (qqq):
    $p,\ n,\ \Lambda,\ \Omega,\ \Sigma$
  - jony:
    jądra atomowe + elektrony
- bozony:

  foton (cząstka), foton optyczny, geantino
- inne cząstki krótkożyciowe:
  - kwarki, gluony;
  - stany wzbudzone: mezonów, barionów, jonów;
  - inne stany krótkożyciowe: mezony, bariony, jony

Cząstki

- Każdy rodzaj cząstki reprezentowany jest przez swoją własną klasę.
- Każda taka klasa ma TYLKO JEDNĄ instancje (static singleton)
- Przykład dla elektronu:
  - Klasa G4Electron definiuje elektron
  - Funkcja G4Electron::Definition()
    zwraca wskaźnik do jedynego obiektu klasy G4Electron
- Aktywacja cząstek:
  - Rodzaj cząstki jest obiektem typu static i jako taki jest automatycznie tworzony przed wykonaniem funkcji main()
  - Jednakże, użytkownik musi dokonać instancji odpowiedniej klasy gdziekolwiek w programie w przeciwnym przypadku dana cząstka nie będzie dostępna w symulacji.

ConstructParticle()

- Aby kod był przejrzysty instancji cząstek dokonujemy w metodzie G4VUserPhysicsList::ConstructParticle().
- Przykład: aktywacja elektronu i protonu w klasie MyPhysicsList:

```
#include "G4Electron.hh"
#include "G4Proton.hh"
#include "MyPhysicsList.hh"
-----------------------------------------------
class MyPhysicsList : public G4VUserPhysicsList
-----------------------------------------------

void MyPhysicsList::ConstructParticle()
{
  G4Electron::Definition();
  G4Proton::Definition();
}
```

Podobnie aktywujemy pozostałe potrzebne cząstki.

ConstructParticle()

- Istnieją klasy pomocnicze umożliwiające aktywacje wszystkich cząstek danego typu.
- Przykład: aktywacja leptonów za pomocą klasy G4LeptonConstructor

```
#include "G4LeptonConstructor.hh"
#include "MyPhysicsList.hh"
-------------------------------------------
class MyPhysicsList : public G4VUserPhysicsList
-------------------------------------------

void MyPhysicsList::ConstructParticle()
{
  G4LeptonConstructor leptons;
  leptons.ConstructParticle();
}
```

Podobnie aktywujemy pozostałe grupy cząstek:
G4BaryonConstructor, G4BosonConstructor, ...

Tablica aktywnych cząstek

- Klasa G4ParticleTable reprezentuje zbiór aktywnych cząstek.
- Klasa ta posiada metody umożliwiające przeszukiwanie listy aktywnych cząstek:
  - FindParticle (G4String name)
  - FindParticle (G4int PDGcode)
- G4ParticleTable jest static singleton a funkcja G4ParticleTable::GetParticleTable zwraca wskaźnik do tablicy cząstek

```
G4ParticleTable::GetParticleTable->FindParticle("electron");
```

Zwraca wskaźnik do elektronu jesli jest on w tablicy aktywnych cząstek cząstek.

Oddziaływania cząstek z materią

- GEANT4 oferuje szereg rodzajów oddziaływania promieniowania z materią.
- W żargonie GEANT'a nazywamy je procesami
- Process jest klasą która określa w jaki sposób cząstka oddziaływuje;
- Użytkownik może definiować własne klasy;
- Procesy w GEANT'cie ogólnie dzielimy na:
  - elektromagnetyczne
  - hadronowe
  - rozpadu
  - transportu
  - parametryzowane;

Procesy elektromagnetyczne

Procesy elektromagnetyczne (oddziaływanie fotonów i cząstek naładowanych)

- standard- energia > 1000 eV (opis na podstawie QED)
- low energy- energie > 250 eV (opis na podstawie danych dośw.)
- optical photon- fotony o dużej długości fali (opis na podstawie klasycznej ED)

Dostepne procesy standardowe:

- rozpraszanie Comptona
- konwersja fotonu
- efekt fotoelektryczny
- jonizacja
- promieniowanie hamowania
- rozpraszanie Coulomba
- anihilacja pary cząstka-antycząstka
- oddziaływanie spolaryzowanych elektronów i fotonów
- produkcja fotonów optycznych:
  promieniowanie: synchrotronowe, przejścia, Czerenkowa, scyntylacje

Dostepne procesy dla fotonów optycznych:
odbicie, załamanie, absorpcja, przesunięcie długości fali,
rozpraszanie Rayleigh'a

Procesy hadronowe

Procesy hadronowe (oddziaływania nieelektromagnetyczne w których biorą udział hadrony)

- oddziaływanie hadron-hadron (opis na podstawie QCD do energii TeV)
  - rozpraszanie elastyczne i nieelastyczne;
  - wychwyt
  - rozszczepienie
- rozpady silne (jądrowe, mezonów, barionów)
- reakcje fotojądrowe
- reakcje leptojądrowe

Proces rozpadu i procesy parametryzowane

- Procesy rozpadu obejmują:
  - rozpady słabe (leptonów, jąder atomowych)
  - rozpady elektromagnetyczne (np. π^0, ...)
  - rozpady silne należą do grupy procesów hadronowych
- Procesy parametryzowane (szybka symulacja):
  np. parametryzacja rozwoju kaskady elektromagnetycznej znacznie przyspiesza obliczenia.
- Szczegółowy opis wszystkich procesów:
  Phisics Reference Manual dostępny na stronie GEANT4

G4VProcess

- Każdy proces fizyczny reprezentowany jest przez klasę wyprowadzoną z klasy G4VProcess
- Wszystkie procesy definiowane są poprzez implementacje dwóch metod:
  - GetPhysicalInteractionLenght()
    - określa kiedy i gdzie dojdzie do oddziaływania danego typu
    - wymaga znajomości przekroi czynnych, czasu życia, ...
  - DoIt()
    - określa stan końcowy po oddziaływaniu (zmienia pęd, generuje cząstki wtórne, ...)
    - wymaga modelu fizycznego

G4VProcess

- Każdy proces może być kombinacją trzech komponent wyszczególnionych ze względu na to kiedy lub gdzie dochodzi do oddziaływania:
  - AtRest zachodzi w określonym czasie;
  - AlongStep jest ”rozciągły” w czasie i przestrzeni
  - PostStep zachodzi w określonym miejscu;
- Najbardziej skomplikowane procesy wymagają zatem implementacji sześciu metod (GetPhysicalInteractionLength oraz DoIt dla każdej komponenty.
- Istnieje szereg klas bazowych wyprowadzonych z klasy G4VProcess w zależności od liczby i rodzaju komponent:
  - G4VRestProcess ma tylko komponentę AtRest np. anihilacja spoczywającego pozytonu
  - G4VContinuouesProcess ma tylko komponentę AlongStep np. promieniowanie Czerenkowa
  - G4VDiscreteProcess ma tylko komponentę PostStep np. rozpraszanie Comptona
  - G4VContinuouesDiscreteProcess ma obie komponenty AlongStep+PostStep np. promieniowanie hamowania

G4ProcessManager

- Dla każdego typu cząstki istnieje osobna instancja klasy G4ProcessManager która zawiera listę procesów dla danej cząstki oraz ich uporządkowanie.
- Metoda G4ProcessManager::AddProcess() dodaje do listy kolejny proces.
- Metoda G4ProcessManager::SetProcessOrdering() określa uporządkowanie procesów.
- Aktywne procesy można dodatkowo dezaktywować lub aktywować ponownie za pomocą metod InActivateProcess() oraz ActivateProcess().

G4VUserPhysicsList::ConstructProcess()

- Metoda ConstructProcess() służy do aktywacji procesów i ich rejestracji do ProcessManager'a
- Dla każdej cząstki zadeklarowanej w ConstructParticle(), użytkownik musi uzyskać wskaźnik do ProcessManager'a danej cząstki i za pomoca metody AddProcess() przekazać mu wskaźniki do procesów które chce aktywowwać.
- Transport (ruch cząstek w przestrzeni) jest traktowany jak proces fizyczny
- Proces transportu reprezentowany jest przez klasę G4Transportation i musi on być zarejestrowany do ProcessManager'a danej cząstki.
- Metoda AddTransportation() jest dziedziczona z klasy bazowej G4VUserPhysicsList i musi być wywołana w metodzie ConstructProcess()
- Metoda AddTransportation() aktywuje proces transportu wszystkich aktywnych cząstek oprócz krótkożyciowych.

MyPhysicsList::ConstructProcess()

src/MyPhysicsList.cc

```
void MyPhysicsList::ConstructProcess()
{
  AddTransportation();
  ConstructEM();
  ConstructGeneral();
}
```

include/MyPhysicsList.hh

```
class MyPhysicsList: public G4VUserPhysicsList
.....................................
 protected:

  // Construct particles and physics processes
  void ConstructParticle();
  void ConstructProcess();
  void SetCuts();
 private:
  // Helper methods
  void ConstructGeneral();
  void ConstructEM();
```

MyPhysicsList::ConstructEM()

src/MyPhysicsList.cc

```
void MyPhysicsList::ConstructEM()
{
theParticleIterator->reset();

while( (*theParticleIterator)() ){
 G4ParticleDefinition* particle = theParticleIterator->value();
 G4ProcessManager* pmanager = particle->GetProcessManager();
 G4String particleName = particle->GetParticleName();
 ..................................................
 } else if (particleName == "e-") {
 // Electron
 pmanager->AddProcess(new G4eMultipleScattering, -1, 1, 1);
 pmanager->AddProcess(new G4eIonisation,         -1, 2, 2);
 pmanager->AddProcess(new G4eBremsstrahlung,     -1, 3, 3);
```

```
    G4int AddProcess(
          G4VProcess *aProcess,
          G4int      ordAtRestDoIt = ordInActive,
          G4int      ordAlongSteptDoIt = ordInActive,
          G4int      ordPostStepDoIt = ordInActive );
```

MyPhysicsList::ConstructEM()

Procesy o tylko jednej aktywnej komponencie można dodawać za pomocą metod pomocniczych AddRestProcess(), AddDiscreteProcess(); AddContinuousProcess()

src/MyPhysicsList.cc

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

MyPhysicsList::ConstructGeneral()

Czasami jeden (ten sam) proces aktywujemy dla kilku cząstek
src/MyPhysicsList.cc

```
void MyPhysicsList::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);
    }
  }
}
```

Gotowe klasy PhysicsList

- Zadanie napisania własnej (poprawnej) implementacji klasy PhysicsList okazuje się być trudne, szczególnie dla początkujących.
- Zaleca się wykorzystanie jako wersji początkowych gotowych implementacji z różnych dziedzin zastosowań.
- Lista rekomendowanych implementacji dostępna jest pod adresem:
  `geant4.cern.ch/support/proc_mod_catalog`
- Przykład dla oddziaływań hadronowych w modelu QGSP_BERT

```
G4VPhysicsConstructor* hadronList =
                new HadronPhysicsQGSP_BERT("hadron",true);
hadronList->ConstructProcess();
```

- Drugim dobrym źródłem początkowych wersji klas PhysicsList są przykłady zawarte w dystrybucji GEANT4 z szeregu dziedzin:
  ls $GEANTINSTALL/example/advanced

MyPhysicsList::SetCuts()

- Użytkownik musi określić warunki na produkcję cząstek wtórnych.
- Należy znaleźć kompromis pomiędzy:
  - potrzebą symulowania niskoenergetycznych cząstek w celu uzyskania wiarygodnych wyników
  - znacznym wydłużeniem czasu trwania symulacji
  - niektóre procesy charakteryzują się ”rozbieżnością w podczerwieni”, produkcją olbrzymiej ilości niskoenergetycznych cząstek wtórnych
- Tradycyjnym rozwiązaniem (nie w GEANT4) jest określenie progu energetycznego. Jeśli energia cząstki spadnie poniżej tego progu to transport cząstki jest przerywany a reszta jej energii jest deponowana punktowo.
- Takie rozwiązanie może powodować niedokładności określenia zasięgu cząstek wtórnych oraz rozkładu depozytów energii.
  - np. zasięg 10 keV fotonów w krzemie jest sto razy większy niż elektronów.
  - zasięg cząstki silnie zależy od materiału ośrodka.

MyPhysicsList::SetCuts()

- W GEANT4 zastosowano próg na produkcję cząstek wtórnych jako zasięg (domyślnie 1 mm) a nie energia;
- Jeśli cząstka nie ma już na tyle energii aby wyprodukować cząstki wtórne których zasięg jest większy od progu to:
  - cząstka traci swoją energię w procesie dyskretnym bez produkcji cząstek wtórnych
  - cząstka traci swoją energię w procesie ciągłym;
- Poprawnie (z dokładnością do zadanego progu) zlokalizowany zasięg cząstek wtórnych
- Jedna wartość progu dla wszystkich materiałów i/lub cząstek.

MyPhysicsList::SetCuts()

- Użytkownik musi sam znaleźć ”najlepszą” wartość progu na produkcję cząstek wtórnych.
- Zależy to od rozmiarów elementów aktywnych detektora oraz zasobów komputerowych (CPU) jakimi dysponujemy.
- Należy zawsze sprawdzić czułość wyników na wartość progu.
- Użytkownik musi określić próg dla fotonów, elektronów i pozytonów
- W pewnych warunkach możliwa jest produkcja cząstek wtórnych poniżej wymaganego progu:
  - jeśli wtórna cząstka ma szanse dotrzeć do obszaru aktywnego detektora.
  - jeśli masa cząstki wtórnej może być zamieniona na energię. Np. w konwersji fotonu na parę lektron-pozyton, pozyton jest zawsze generowany, nawet z zerową energią bo może później anihilować

MyPhysicsList::SetCuts()

Próg można ustawić dla wszystkich cząstek taki sam za pomocą metody SetCutsWithDefault() (UI /run/setCut)

```
void MyPhysicsList::SetCuts()
{
  SetCutsWithDefault();
}
```

lub dla poszczególnych cząstek

```
void MyPhysicsList::SetCuts()
{
 SetCutValue(defaultCutValue, "gamma");
 SetCutValue(defaultCutValue, "e-");
 SetCutValue(defaultCutValue, "e+");
}
```

wartość domyślną można zmienić w konstruktorze listy procesów fizycznych

```
MyPhysicsList::MyPhysicsList()
  :G4VUserPhysicsList()
{
  defaultCutValue = 1.0*mm;
}
```

Fotony optyczne

- Fotony jako obiekty klasy G4Gamma nie podlegają prawom optyki klasycznej nawet jeśli ich długość fali jest znacznie większa niż odległości między atomami;
- Naturę klasycznej fali em. posiadają fotony optyczne;
- Są to obiekty klasy G4OpticalPhoton, które dodajemy do tablicy cząstek;

```
#include "G4OpticalPhoton.hh"
-----------------------------------------------
class MyPhysicsList : public G4VUserPhysicsList
-----------------------------------------------
void MyPhysicsList::ConstructParticle()
{
  G4OpticalPhoton::Definition();
}
```

- GEANT4 dostarcza informacji o polaryzacji fotonów optycznych ale nie o pełnej fazie, zatem nie symuluje np. interferencji

```
G4DynamicParticle::SetPolarization(ux,uy,uz);
```

Fotony optyczne

- Fotony optyczne produkowane są przez cząstki naładowane w trzech procesach:
  - promieniowanie Czerenkowa - klasa G4Cerenkov
  - scyntylacje - klasa G4Scintillation
  - promieniwanie przejścia - G4TransitionRadiation
- Procesy te muszą być aktywowane np. scyntylacje

```
void MyPhysicsList::ConstructGeneral(){
  // Add Production of optical photons;
  G4Scintillation*  theScintillationProcess  = new G4Scintillation();

  theParticleIterator->reset();

  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* pmanager = particle->GetProcessManager();
-----------------------------------------------------------------------
    if (theScintillationProcess->IsApplicable(*particle)) {
      pmanager->AddProcess(theScintillationProcess);
      pmanager->SetProcessOrdering(theScintillationProcess, idxPostStep);
    }
```

- Uwaga: te procesy produkują fotony optyczne łamiąc zasadę zachowania energii

Fotony optyczne

- Fotony optyczne oddziałują z materią poprzez:
  - odbicie i załamanie na granicy ośrodków - klasa G4OpBoundaryProcess
  - absorpcje - klasa G4OpAbsorption
  - rozpraszanie Rayleigh'a G4OpRayleigh
  - przesunięcie długości fali (absorbcja i reemisja) - klasa G4OpWLS
- Procesy te muszą być aktywowane dla fotonów optycznych:

```
 theAbsorptionProcess          = new G4OpAbsorption();
 theRayleighProcess            = new G4OpRayleigh();
 theBoundaryProcess            = new G4OpBoundaryProcess();
 theWLSProcess                 = new G4OpWLS();


------------------------------------------------------------------
 if (particleName == "opticalphoton") {
  // Optical photons
  pmanager->AddDiscreteProcess(theAbsorptionProcess);
  pmanager->AddDiscreteProcess(theRayleighProcess );
  pmanager->AddDiscreteProcess(theBoundaryProcess);
  pmanager->AddDiscreteProcess(theWLSProcess);
```

Fotony optyczne

- Kluczową sprawą w implementacji procesów produkcji i oddziaływania fotonów optycznych są optyczne własności materiałów;
- Własności te definiuje się w postaci tablicy własności tworzonej dla każdego wymaganego materiału;
- Własności są funkcją energii fotonu optycznego.
- Można zdefiniować następujące własności optyczne materiałów:
  - współczynnik załamania;
  - długość absorpcji;
  - wydajność scyntylacji (komponenta szybka i wolna);
  - stała czasowa scyntylacji (komponenta szybka i wolna)
  - widma absorpcji i reemisji dla materiałow WLS
  - opóźnienie czasowe dla materiałów WLS
  - średni kąt rozproszenia (Rayleigh)
- Można zdefiniować następujące własności optyczne granicy między ośrodkami:
  - rodzaj przejścia (dielektryk-dielektryk, dielektryk-metal)
  - sposób wykończenia (gładkie/chropowate/pomalowane/...)