Zastosowanie pakietu Geant4 w fizyce jądrowej Wykład 9

Aleksandra Fijałkowska

20 grudnia 2018

Propozycje projektów

Symulacja światła w pakiecie Geant4 obejmuję:

- produkcję światła – scyntylacja, promieniowanie Czerenkowa
- propagację światła
- kolekcję i rejestrację

Lista fizyczna

Fotony optyczne są reprezentowane przez klasę **G4OpticalPhoton**. Wykorzystanie fotonów, podobnie jak każdej innej cząstki w symulacji wymaga dodania ich definicji do tabeli cząstek (metoda **ConstructParticle**).

```
void PhysicsList::ConstructParticle()
{
G4OpticalPhoton::Definition();
}
```

Geant dysponuje trzema procesami, wktórzych fotony optyczne mogą być produkowane: scyntylacja (**G4Scintillation**), promieniowanie Czerenkowa (**G4Cerenkov**) oraz promieniowanie przejścia (**G4TransitionRadiation**). Aby doszło do powstania fotonów przynajmniej jeden z tych procesów musi być aktywowany (chyba, że fotony optyczne będą wysyłane na początku zdarzenia jako cząstki pierwotne), przykład:

```
void PhysicsList::ConstructProcess()
{
G4Scintillation* scintillationProcess  = new G4Scintillation();
  auto aParticleIterator=GetParticleIterator();
  aParticleIterator->reset();
  while( (*aParticleIterator)() )
  {
     G4ParticleDefinition* particle = theParticleIterator->value();
     G4ProcessManager* pmanager = particle->GetProcessManager();
     if (theScintillationProcess->IsApplicable(*particle))
     {
          pmanager->AddProcess(theScintillationProcess);
          pmanager->SetProcessOrdering(theScintillationProcess, idxPostStep);
      }
   }
}
```

Lista fizyczna cd.

Do tego wszystkiego należy dołączyć procesy, którym mogą ulegać fotony (do tej pory dołączyliśmy procesy, w których fotony optyczne powstają, a którym ulegają cząstki naładowane). Do procesów optycznych zaliczamy:

- Absorpcja
- Rozproszenie Rayleigh-a
- Procesy przejścia przez granicę dwóch ośrodków (dielektryk-dielektryk, dielektryk-metal, dielektryk-cialo czarne)
- Wavelength Shifting (WLS) – przesunięcie długości fali

Fragment metody ConstructProcess() dołączający procesy optyczne

```
theAbsorptionProcess          = new G4OpAbsorption();
theRayleighProcess            = new G4OpRayleigh();
theBoundaryProcess            = new G4OpBoundaryProcess();
theWLSProcess                 = new G4OpWLS();
if (particleName == "opticalphoton")
{
    pmanager->AddDiscreteProcess(theAbsorptionProcess);
    pmanager->AddDiscreteProcess(theRayleighProcess );
    pmanager->AddDiscreteProcess(theBoundaryProcess);
    pmanager->AddDiscreteProcess(theWLSProcess);
}
```

Lista fizyczna cd.

Dobra informacja jest taka, że jeśli cząstki i procesy fizyczne są dołączane w sposób, jaki zastosowaliśmy w naszych przykładowych kodach, czyli przez dziedziczenie po **G4VModularPhysicsList()** oraz wywołanie metody **RegisterPhysics** dla każdej z niezbędnych „typów" fizyki dołączenie procesów odpowiadających za powstanie i propogację fotonów optycznych jest bardzo łatwe. Wystarczy w konstruktorze klady **PhysicsList** „zarejestrować" klasę **G4OpticalPhysics** zawierającą wszystkie niezbędne procesy. Fragment metody **ConstructProcess()** dołączający procesy optyczne:

```
PhysicsList::PhysicsList() : G4VModularPhysicsList()
{
...
  RegisterPhysics( new GeneralPhysics("general") );
  // EM Physics
  RegisterPhysics( new EMPhysics("standard EM"));
  // Muon Physics
  RegisterPhysics( new MuonPhysics("muon"));
...
  // Optical Physics
  G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
  RegisterPhysics( opticalPhysics );
}
```

Właściwości materiałów

Geant nie posiada wewnętrznej bazy właśności optycznych różnych materiałów. Dotyczy to zarówno produkcji światła, jak i jego propagacji. Wszystkie parametry materiałów, w których mogą propagować się fotony muszą być określone przez użytkownika. Własności te określa się przez podanie odpowiednich tablic w funkcji energii fotonu. Przykładowe własności materiałów:

- współczynnik załamania
- długość absorpcji
- wydajność scyntylacji (można uwzględnić dwa komponenty – szybki wolny)
- stała czasowa scyntylacji (dwia komponenty)
- średni kąt rozproszenia Rayleigh-a

Właściwości materiałów, przykład

Przykład budowania materiału, który może emitować fotony optyczne (scyntylatora) wraz z określeniem jego własności optycznych

```
G4Material* MaterialsManager::GetBC408()
{
  //budowa materiału
  BC408 = new G4Material("BC408", density= 1.032*g/cm3, numberElements=2);
  BC408->AddElement(H, 138);//H:C ratio = 1.104
  BC408->AddElement(C, 125);

  //tablica do której wkładamy własnosci optyczne
  G4MaterialPropertiesTable* BC408LightProperties = new G4MaterialPropertiesTable();

  //scyntylacj, energie fotonów
  int scintEntries = 12;
  G4double photonEnergy[] = {2.38*eV, 2.48*eV, 2.58*eV, 2.69*eV,
                             2.75*eV, 2.82*eV, 2.92*eV, 2.95*eV,
                             3.02*eV, 3.10*eV, 3.26*eV, 3.44*eV};

  //liczba wyemitowanych fotonów na 1 MeV zdeponowanej energii
  BC408LightProperties->AddConstProperty("SCINTILLATIONYIELD", (10000.0)/MeV);

  //niepewność liczby wyemitowanych fotonów:
  //sigma = RESOLUTIONSCALE*sqrt(MeanNumberofPhotons)
  //można ten rozkład poszerzyć, jeśli się chce
  BC408LightProperties->AddConstProperty("RESOLUTIONSCALE", 1.0);

cdn.
```

Właściwości materiałów, przykład


```
cd.
    //prawdopodobieństwo emisji danej energii dla szybkiej składowej
  G4double scintilFast[] = { 0.02, 0.09, 0.20, 0.40, 0.55, 0.80,
                             1.00, 0.80, 0.50, 0.20, 0.08, 0.02 };
  BC408LightProperties->AddProperty("FASTCOMPONENT",photonEnergy,
                                    scintilFast, scintEntries)->SetSpline(true);

  //czas wyświecania szybkiej składowej
  BC408LightProperties->AddConstProperty("FASTTIMECONSTANT", 2.1*ns);

  //wolna składowa - załóżmy, ze jest taka sama
  G4double* scintilSlow = scintilFast;
  BC408LightProperties->AddProperty("SLOWCOMPONENT",photonEnergy,
                                    scintilSlow,scintEntries)->SetSpline(true);

  //ma jadnak dłuższy czas wyświecania
  BC408LightProperties->AddConstProperty("SLOWTIMECONSTANT", 10.*ns);

  //jaka część całej scyntylacji to szybka składowa:
  BC408LightProperties->AddConstProperty("YIELDRATIO",0.5);

cdn.
```

Właściwości materiałów, przykład

Pozostałe własności (nie związane z emisją światła)

```
cd.
  //współczynnik załamania
  G4double rIndex[] = { 1.58, 1.58, 1.58, 1.58, 1.58, 1.58,
                        1.58, 1.58, 1.58, 1.58, 1.58, 1.58 };

  BC408LightProperties->AddProperty("RINDEX", photonEnergy,
                                    rIndex, scintEntries);

  //długość absorpcji
  G4double absLength[] = { 3.8*m, 3.8*m, 3.8*m, 3.8*m, 3.8*m, 3.8*m,
                           3.8*m, 3.8*m, 3.8*m, 3.8*m, 3.8*m, 3.8*m};

  BC408LightProperties->AddProperty("ABSLENGTH", photonEnergy,
                                    absLength, scintEntries);

  //ma koniec dodajemy naszą tablicę do materiału
  BC408->SetMaterialPropertiesTable(BC408LightProperties);
  BC408LightProperties->DumpTable();

  //zwracamy wskaźnik do materiału
  return BC408;
}
```

Właściwości materiałów, przykład 2

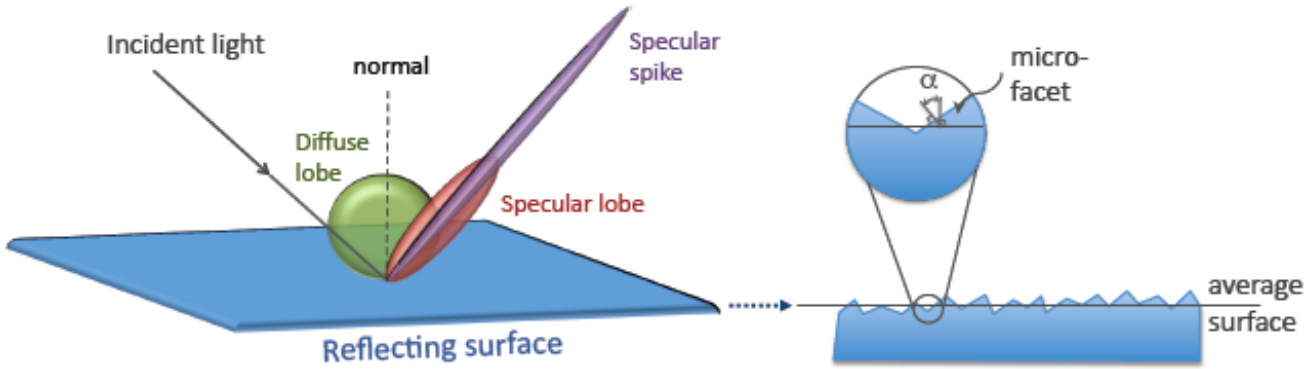
Dużo łatwiej to wygląda jeśli fotony mają tylko poruszać się w materiale, a nie być w nim emitowane:

```
G4Material* MaterialsManager::GetAir()
{
  air = new G4Material("Air", density= 1.29*mg/cm3, numberElements=2);
  air->AddElement(N, 70*perCent);
  air->AddElement(O, 30*perCent);

  G4MaterialPropertiesTable* airLightProperties = new G4MaterialPropertiesTable();
  //definiuję parametry materialu dla tych samych fotonów, ale nie muszę
  int scintEntries = 12;
  G4double energies[] = {2.38*eV, 2.48*eV, 2.58*eV, 2.69*eV,
                         2.75*eV, 2.82*eV, 2.92*eV, 2.95*eV,
                         3.02*eV, 3.10*eV, 3.26*eV, 3.44*eV};

  G4double rIndex[] = {1.00029, 1.00029, 1.00029, 1.00029,
                       1.00029, 1.00029, 1.00029, 1.00029,
                       1.00029, 1.00029, 1.00029, 1.00029};

  G4double absLength[] = {100.0*m, 100.0*m, 100.0*m, 100.0*m,
                          100.0*m, 100.0*m, 100.0*m, 100.0*m,
                          100.0*m, 100.0*m, 100.0*m, 100.0*m};

  airLightProperties->AddProperty("RINDEX", energies, rIndex, scintEntries);
  airLightProperties->AddProperty("ABSLENGTH", energies, absLength, scintEntries);
  air->SetMaterialPropertiesTable(airLightProperties);
  return air;
}
```

Wykończenie elementów

Wszystkie materiały, w których mogą się poruszać fotony muszą mieć ustalony współczynnik załamania i długość absorpcji.
Geant umożliwia także określenie rodzaj przejścia pomiędzy elementami geometrii (dielektryk-dielektryk, dielektryk-metal) oraz sposobu wykończenia ich powierzchni (gładkie/chropowate/pomalowane).
Dostępne parametry:

- Finish – polished, ground, groundfrontpainted, lista w G4OpticalSurface.hh
- Model – unified, glisur (z geant3)
- RINDEX – współczynnik załamania
- SPECULARLOBECONSTANT
- SPECULARSPIKECONSTANT
- REFLECTIVITY – odbicie
- EFFICIENCY – wydajność pochłaniania



http://wiki.opengatecollaboration.org/index.php/Users_Guide:Generating_and_tracking_optical_photons

Przykład, materiał otaczający scyntylator

```
G4double energy [] = {2.38*eV, 2.48*eV, 2.58*eV, 2.69*eV,
                      2.75*eV, 2.82*eV, 2.92*eV, 2.95*eV,
                      3.02*eV, 3.10*eV, 3.26*eV, 3.44*eV};
int entries = 12;
//nie pochałania fotonów (efficiency)
G4double efficiency[] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                          0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
//100% odbicie
G4double reflectivity[] = { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                            1.0, 1.0, 1.0, 1.0, 1.0, 1.0};

//sposob odbijania
G4double secularSpike[] = { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
                            1.0, 1.0, 1.0, 1.0, 1.0, 1.0};

G4MaterialPropertiesTable* reflFoilMat = new G4MaterialPropertiesTable();
reflFoilMat->AddProperty("EFFICIENCY",  energy, efficiency, entries);
reflFoilMat->AddProperty("REFLECTIVITY", energy, reflectivity, entries);
reflFoilMat->AddProperty("SPECULARSPIKECONSTANT", energy, secularSpike, entries);

//tworzenie powierzchmi optycznej:
G4OpticalSurface* reflectFoilOpSurf = new G4OpticalSurface("reflectFoilOpSurf");
reflectFoilOpSurf->SetType( dielectric_metal );
reflectFoilOpSurf->SetFinish( polished );
reflectFoilOpSurf->SetModel( unified );
reflectFoilOpSurf->SetMaterialPropertiesTable(reflFoilMatTable);

//przypisanie własności do zmiennej logicznej wrappingLogic
new G4LogicalSkinSurface( "reflectFoilSkin", wrappingLogic, reflectFoilOpSurf);
```

Przykład, fotokatoda fotopowielacza

```
G4double energy [] = {2.38*eV, 2.48*eV, 2.58*eV, 2.69*eV,
                      2.75*eV, 2.82*eV, 2.92*eV, 2.95*eV,
                      3.02*eV, 3.10*eV, 3.26*eV, 3.44*eV};
int entries = 12;
G4double efficiency[]={1., 1., 1., 1., 1., 1.,
                   1., 1., 1., 1., 1., 1.};
G4double reflectivity[]={0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
                     0.2, 0.2, 0.2, 0.2, 0.2, 0.2};

G4MaterialPropertiesTable* photocathMatTable = new G4MaterialPropertiesTable();
photocathMatTable->AddProperty("EFFICIENCY", energy,efficiency, entries);
photocathMatTable->AddProperty("REFLECTIVITY", energy,reflectivity, entries);

G4OpticalSurface* photocathOpSurf = new G4OpticalSurface("photocathOpSurf");
photocathOpSurf->SetModel( unified );
photocathOpSurf->SetFinish( polished );
photocathOpSurf->SetType( dielectric_metal );
photocathOpSurf->SetMaterialPropertiesTable(photocathMatTable);
new G4LogicalSkinSurface( "photocathOptSkin", photocathLogic, photocathOpSurf);
```

Detekcja fotonów

Detekcja fotonów optycznych w kodzie Geant4 jest specyficzna.
Po każdym kroku, w którym fotony przechodzą przez granicę dwóch ośrodków mają nadany status, np. Transmission, TotalInternalReflection, Absorption, Detection itp. Jeśli status fotonu jest „Detection", to uznajemy, że został on „zmierzony". Uzyskanie informacji o statusie odbywa się w metodzie void SteppingAction::UserSteppingAction(const G4Step* theStep)

Detekcja fotonów cd.

```
void SteppingAction::UserSteppingAction(const G4Step* theStep)
{
  G4OpBoundaryProcessStatus boundaryStatus=Undefined;
  static G4ThreadLocal G4OpBoundaryProcess* boundary=NULL;
  if(!boundary)
  {
    G4ProcessManager* pm
      = theStep->GetTrack()->GetDefinition()->GetProcessManager();
    G4int nprocesses = pm->GetProcessListLength();
    G4ProcessVector* pv = pm->GetProcessList();
    for( int i=0;i<nprocesses;i++){
      if((*pv)[i]->GetProcessName()=="OpBoundary"){
        boundary = (G4OpBoundaryProcess*)(*pv)[i];
        break;
      }
    }
  }
  G4ParticleDefinition* particleType = theTrack->GetDefinition();
  if(particleType==G4OpticalPhoton::OpticalPhotonDefinition())
  {
    boundaryStatus=boundary->GetStatus();
    if(thePostPoint->GetStepStatus()==fGeomBoundary)
    {
      if (boundaryStatus == Detection)
      {
         //foton został zmierzony
      }
    }
  }
```

Detekcja fotonów cd.

Można oczywiście śledzić fotony „klasycznie” patrząc na początek i koniec kroku.