Geant4 Code Reivew

Geant4 Code Review

Scintillation by incident particle in Geant4

```
if (scintillationByParticleType) {
   // The scintillation response is a function of the energy
   // deposited by particle types.

   // Get the definition of the current particle
   G4ParticleDefinition *pDef = aParticle->GetDefinition();
   G4MaterialPropertyVector *Scint_Yield_Vector = NULL;

   // Obtain the G4MaterialPropertyVectory containing the
   // scintillation light yield as a function of the deposited
   // energy for the current particle type

   // Protons
   if(pDef==G4Proton::ProtonDefinition())
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("PROTONSCINTILLATIONYIELD");

   // Deuterons
   else if(pDef==G4Deuteron::DeuteronDefinition())
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("DEUTERONSCINTILLATIONYIELD");

   // Tritons
   else if(pDef==G4Triton::TritonDefinition())
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("TRITONSCINTILLATIONYIELD");

   // Alphas
   else if(pDef==G4Alpha::AlphaDefinition())
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("ALPHASCINTILLATIONYIELD");

   // Ions (particles derived from G4VIon and G4Ions)
   // and recoil ions below tracking cut from neutrons after hElastic
   else if(pDef->GetParticleType()== "nucleus" ||
           pDef==G4Neutron::NeutronDefinition())
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("IONSCINTILLATIONYIELD");

   // Electrons (must also account for shell-binding energy
   // attributed to gamma from standard PhotoElectricEffect)
   else if(pDef==G4Electron::ElectronDefinition() ||
           pDef==G4Gamma::GammaDefinition())
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("ELECTRONSCINTILLATIONYIELD");

   // Default for particles not enumerated/listed above
   else
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("ELECTRONSCINTILLATIONYIELD");

   // If the user has not specified yields for (p,d,t,a,carbon)
   // then these unspecified particles will default to the
   // electron's scintillation yield
   if(!Scint_Yield_Vector){
     Scint_Yield_Vector = aMaterialPropertiesTable->
       GetProperty("ELECTRONSCINTILLATIONYIELD");
   }

   // Throw an exception if no scintillation yield is found
   if (!Scint_Yield_Vector) {
      G4ExceptionDescription ed;
      ed << "\nG4Scintillation::PostStepDoIt(): "
            << "Request for scintillation yield for energy deposit and particle type without correct entry in MaterialPropertiesTable\n"
            << "ScintillationByParticleType requires at minimum that ELECTRONSCINTILLATIONYIELD is set by the user\n"
```

Scintillation Yield
(언어마다 다르지만 일반적으로 NULL === false)

Particle type
: Alpha particle

각각의 particle에 대해 Scintillation yield가 지정되지 않았다면, electron에서의 scintillation yield와 동일하게 취급됨

```
            << G4endl;
      G4String comments = "Missing MaterialPropertiesTable entry - No correct entry in MaterialPropertiesTable";
      G4Exception("G4Scintillation::PostStepDoIt","Scint01",
                  FatalException,ed,comments);
      return G4VRestDiscreteProcess::PostStepDoIt(aTrack, aStep);
   }

   if (verboseLevel>1) {
      G4cout << "\n"
             << "Particle = " << pDef->GetParticleName() << "\n"
             << "Energy Dep. = " << TotalEnergyDeposit/MeV << "\n"
             << "Yield = "
             << Scint_Yield_Vector->Value(TotalEnergyDeposit)
             << "\n" << G4endl;
   }

   // Obtain the scintillation yield using the total energy
   // deposited by the particle in this step.

   // Units: [# scintillation photons]
   ScintillationYield = Scint_Yield_Vector->
                          Value(TotalEnergyDeposit);
} else {
   // The default linear scintillation process
   ScintillationYield = aMaterialPropertiesTable->
                          GetConstProperty("SCINTILLATIONYIELD");

   // Units: [# scintillation photons / MeV]
   ScintillationYield *= YieldFactor;
}

G4double ResolutionScale    = aMaterialPropertiesTable->
                          GetConstProperty("RESOLUTIONSCALE");
```

Scintillation yield가 설정되어 있지 않다면 예외 처리가 불가능하기 때문에 return이 아니라 ,throw를 써서 프로그램을 끝냄을 의미

Geant4 Code Review

Scintillator photons in Geant4

Scintillation photon의 수를 결정하는 방법

평균 photon의 수가 10보다 크면 Gaussian distribution으로 주어진다. 이 때, gaussian의 sigma 값은 resolution scale * scintillation photon의 수의 sqrt 값을 이용한다.

평균 photon의 수가 10보다 작으면 Poisson distribution으로 주어진다.

```
G4int NumPhotons;

if (MeanNumberOfPhotons > 10.)
{
  G4double sigma = ResolutionScale * std::sqrt(MeanNumberOfPhotons);
  NumPhotons = G4int(G4RandGauss::shoot(MeanNumberOfPhotons,sigma)+0.5);
}
else
{
  NumPhotons = G4int(G4Poisson(MeanNumberOfPhotons));
}

if (NumPhotons <= 0)
{
   // return unchanged particle and no secondaries

   aParticleChange.SetNumberOfSecondaries(0);

   return G4VRestDiscreteProcess::PostStepDoIt(aTrack, aStep);
}

aParticleChange.SetNumberOfSecondaries(NumPhotons);

if (fTrackSecondariesFirst) {
   if (aTrack.GetTrackStatus() == fAlive )
          aParticleChange.ProposeTrackStatus(fSuspend);
}


G4int materialIndex = aMaterial->GetIndex();
```

Geant4 Code Review

Saturation(Quenching) in Geant4

```
// Birks law saturation:

//G4double constBirks = 0.0;

//constBirks = aMaterial->GetIonisation()->GetBirksConstant();

G4double MeanNumberOfPhotons;

// Birk's correction via emSaturation and specifying scintillation by
// by particle type are physically mutually exclusive

if (scintillationByParticleType)
   MeanNumberOfPhotons = ScintillationYield;
else if (emSaturation)
   MeanNumberOfPhotons = ScintillationYield*
                         (emSaturation->VisibleEnergyDeposition(&aStep));
else
   MeanNumberOfPhotons = ScintillationYield*TotalEnergyDeposit;
```

emSaturation이 정의 되었는지 확인

```
protected:

        void BuildThePhysicsTable();
        // It builds either the fast or slow scintillation integral table;
        // or both.

        // Class Data Members

        G4PhysicsTable* theSlowIntegralTable;
        G4PhysicsTable* theFastIntegralTable;

        G4bool fTrackSecondariesFirst;
        G4bool fFiniteRiseTime;

        G4double YieldFactor;

        G4double ExcitationRatio;

        G4bool scintillationByParticleType;

private:

        G4double single_exp(G4double t, G4double tau2);
        G4double bi_exp(G4double t, G4double tau1, G4double tau2);

        // emission time distribution when there is a finite rise time
        G4double sample_time(G4double tau1, G4double tau2);

        G4EmSaturation* emSaturation;
```

Saturation의 경우를 설명
(다음 페이지에 자세히 설명)

G4EmSaturation::G4EmSaturation ( )

Definition at line 58 of file G4EmSaturation.cc.

References G4NistManager::Instance().

```
{
  verbose = 1;
  manager = 0;

  curMaterial = 0;
  curBirks    = 0.0;
  curRatio    = 1.0;
  curChargeSq = 1.0;
  nMaterials  = 0;

  electron = 0;
  proton   = 0;
  nist     = G4NistManager::Instance();

  Initialise();
}
```

C++에서 Constructor가 정의가 되어 있을 경우, if 절에 들어가면 true로 간주.
Constructor가 NULL으로 정의 되어 있거나 제대로 정의 되어 있지 않은 경우, if 절에 들어가면 false로 간주.

Geant4 Code Review

Saturation(Quenching) in Geant4

Evis = energy deposit per step
Bfactor = Birks' constant

Correct evis by Birks' law for gamma ray
(pdg code 22 = gamma)

Nloss = non-ionizing energy loss(niel)
Eloss = ionizing energy = energy deposit - NIEL

Correct Ionizing Energy
(by Birks' law)

Non ionizing energy loss가
scintillation process에 기여하는
energy correction

Return corrected evis

```
G4double G4EmSaturation::VisibleEnergyDeposition ( const G4ParticleDefinition * ,
                                                    const G4MaterialCutsCouple * ,
                                                    G4double                   length,
                                                    G4double                   edepTotal,
                                                    G4double                   edepNIEL = 0.0
                                                  )
```

Definition at line 83 of file G4EmSaturation.cc.

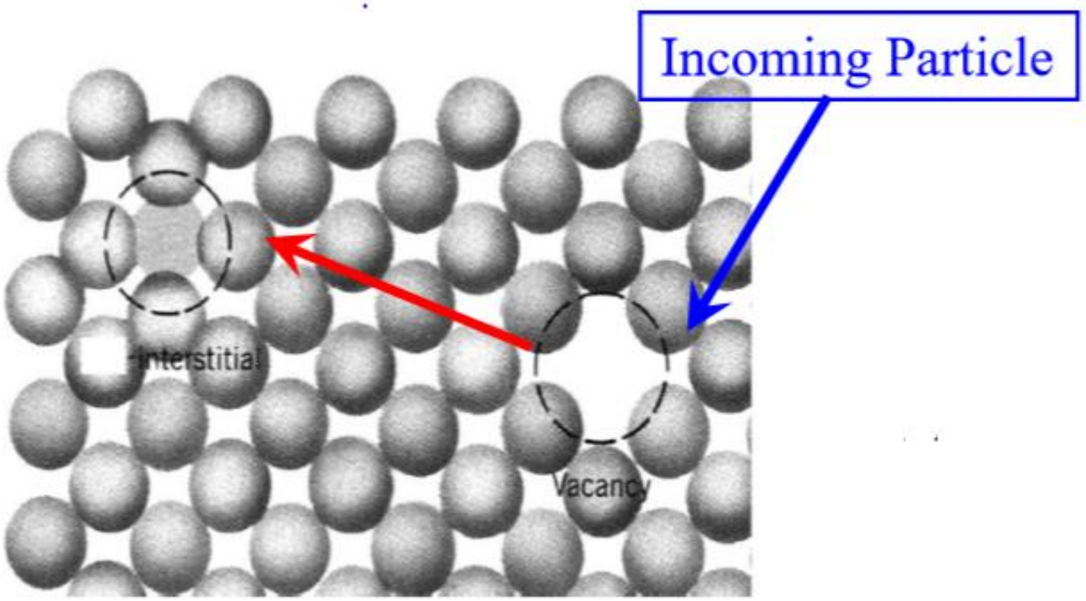
References G4MaterialCutsCouple::GetMaterial(), G4ParticleDefinition::GetPDGEncoding(), G4LossTableManager::GetRange(), and G4Proton::Proton().

Referenced by G4Scintillation::PostStepDoIt(), and VisibleEnergyDeposition().

```
{
  if(edep <= 0.0) { return 0.0; }

  G4double evis = edep;
  G4double bfactor = FindBirksCoefficient(couple->GetMaterial());

  if(bfactor > 0.0) {

    G4int pdgCode = p->GetPDGEncoding();
    // atomic relaxations for gamma incident
    if(22 == pdgCode) {
      evis /= (1.0 + bfactor*edep/manager->GetRange(electron,edep,couple));

      // energy loss
    } else {

      // protections
      G4double nloss = niel;
      if(nloss < 0.0) nloss = 0.0;
      G4double eloss = edep - nloss;

      // neutrons
      if(2112 == pdgCode || eloss < 0.0 || length <= 0.0) {
        nloss = edep;
        eloss = 0.0;
      }

      // continues energy loss
      if(eloss > 0.0) { eloss /= (1.0 + bfactor*eloss/length); }

      // non-ionizing energy loss
      if(nloss > 0.0) {
        if(!proton) { proton = G4Proton::Proton(); }
        G4double escaled = nloss*curRatio;
        G4double range = manager->GetRange(proton,escaled,couple)/curChargeSq;
        nloss /= (1.0 + bfactor*nloss/range);
      }

      evis = eloss + nloss;
    }
  }

  return evis;
}
```

Neutron

Non-ionizing Energy Loss(NIEL)



- 일반적으로 scintillation process를 일으키는 Energy(Bethe-Bloch formula)는 ionizing energy로, 전자와 상호작용하여 ionizing 시키는 에너지에 해당한다.
- 하지만 이와 다른 방식으로 energy를 deposit 할 수 있다.
- 들어온 입자가 원자를 이동시켜 Vaccancy와 interstitial defect를 만드는 방법인데, 이 때의 에너지를 Non-ionizing Energy loss라고 한다.
- 이 때의 에너지 역시 scintillation을 발생시킬 수 있다.

Conclusion

- Geant4에는 Birks' law의 Saturation(Quenching) Effect가 포함되어 있다.
- Scintillation Process를 발생시키는 energy deposit에는 ionzing energy deposit과 non-ionizing energy deposit 두 가지 종류가 있고, 모두 Geant4 G4Scintillation package에 포함되어 있다.
- Optical Simulation에서 더 정밀한 시뮬레이션을 위해서는, 사용하는 입자를 명확히 한 뒤, 해당 입자에 대한 scintillation yield를 각각 넣어줘야 한다.

Geant4 Code Review

G4EmSaturation 중
FindBirksCoefficient 함수의 Source Code

Mass fraction for calculating energy loss

Z = Atomic Number
W = Z^2 * atomic density(= the number of atom per unit volume)

```
// compute mean mass ratio
curRatio = 0.0;
curChargeSq = 0.0;
G4double norm = 0.0;
const G4ElementVector* theElementVector = mat->GetElementVector();
const G4double* theAtomNumDensityVector = mat->GetVecNbOfAtomsPerVolume();
size_t nelm = mat->GetNumberOfElements();
for (size_t i=0; i<nelm; ++i) {
  const G4Element* elm = (*theElementVector)[i];
  G4double Z = elm->GetZ();
  G4double w = Z*Z*theAtomNumDensityVector[i];
  curRatio += w/nist->GetAtomicMassAmu(G4int(Z));
  curChargeSq = Z*Z*w;
  norm += w;
}
curRatio *= proton_mass_c2/norm;
curChargeSq /= norm;

// store results
matPointers.push_back(mat);
matNames.push_back(name);
massFactors.push_back(curRatio);
effCharges.push_back(curChargeSq);
nMaterials++;
if(curBirks > 0.0 && verbose > 0) {
  G4cout << "### G4EmSaturation::FindBirksCoefficient Birks coefficient for "
         << name << "  " << curBirks*MeV/mm << " mm/MeV" << G4endl;
}
return curBirks;
```