```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//

#include "CalorimeterSD.hh"
#include "CaloHit.hh"
#include "G4VPhysicalVolume.hh"
#include "G4LogicalVolume.hh"
#include "G4Track.hh"
#include "G4Step.hh"
#include "G4ParticleDefinition.hh"
#include "G4VTouchable.hh"
#include "G4TouchableHistory.hh"
#include "G4SystemOfUnits.hh"
#include "G4ios.hh"

CalorimeterSD::CalorimeterSD(G4String name):G4VSensitiveDetector(name){
  collectionName.insert("caloCollection");
  useBirks=false;

  birk1scint=0.0052*(g/(MeV*cm2));
  birk2scint=0.142;
  birk3scint=1.75;

  birk1crystal=0.03333*(g/(MeV*cm2));
  birk2crystal=0.;
  birk3crystal=1.;

  fMessenger = new CalorimeterSDMessenger(this);

}

CalorimeterSD::~CalorimeterSD()
{;}

void CalorimeterSD::Initialize(G4HCofThisEvent*){
  CaloCollection = new CaloHitsCollection(SensitiveDetectorName,collectionName[0
]);

  LayerID.clear();
  verboseLevel = 0;
}


G4int CalorimeterSD::GetDetID(G4Step*aStep){

  G4int layer2Up = aStep->GetPreStepPoint()->GetTouchable()->GetCopyNumber(2);
```



```
  G4int layerUp = aStep->GetPreStepPoint()->GetTouchable()->GetCopyNumber(1);
  G4int layerVol = aStep->GetPreStepPoint()->GetTouchable()->GetCopyNumber();
  G4VPhysicalVolume* physVol = aStep->GetPreStepPoint()->GetPhysicalVolume();
  G4String volumeID = physVol->GetName();
  G4int detID = -1000;
  if(!volumeID.compare("S1ScintillatorM"))
    detID= 1E3 + 3*1E2 + 1*1E1 + (layerVol+1)*1E0;
  if(!volumeID.compare("S1ScintillatorP"))
    detID= 1E3 + 3*1E2 + 2*1E1 + (layerVol+1)*1E0;
  if(!volumeID.compare("ActiveLayerScint"))
    detID= 1E3 + 2*1E2 + (layerUp+2);
  if(!volumeID.compare("ActiveLastLayerScint"))
    detID= 1E3 + 2*1E2 + 1;
  if(!volumeID.compare("ActiveBlockCrystal"))
    detID= 1E3 + 1*1E2 + (layer2Up+1)*1E1 + (layerUp+1)*1E0;
  return detID;
}

G4bool CalorimeterSD::ProcessHits(G4Step*aStep,G4TouchableHistory*){
  G4double edep = aStep->GetTotalEnergyDeposit();
  G4int tkID = aStep->GetTrack()->GetTrackID();
  // if(verboseLevel>1) G4cout << "Calo step edep(MeV) = " << edep/MeV <<" ; giv
en by Track = "<<tkID<< G4endl;
  if(edep==0.) return false;
  if(useBirks)
    edep*=BirksAttenuation(aStep);
  //  G4VPhysicalVolume* physVol = aStep->GetPreStepPoint()->GetPhysicalVolume()
;

  //  std::stringstream ss;
//   if(layerUp==layerVol||layerUp>layerVol){
//     ss << layerUp;
//   }else{
//     ss << layerVol;
//   }
  //G4String volumeID = physVol->GetName()+ss.str();
  //G4String volumeID = physVol->GetName();


  G4int detID;
  detID=GetDetID(aStep);

   //  if(verboseLevel>1) G4cout << "Calo step on Volume = " << volumeID << G4en
dl;
   if(verboseLevel>1) G4cout << "Calo step on Volume = "<< detID << G4endl;

   if(LayerID.find(detID)==LayerID.end()){
    //    CaloHit* calHit = new CaloHit(volumeID);
    CaloHit* calHit = new CaloHit(detID);
    calHit->SetEdep(edep/MeV,tkID);
    G4int icell = CaloCollection->insert(calHit);
    LayerID[detID] = icell - 1;

     if(verboseLevel>1){
       G4cout << " New  Hit on Calo Layer "
          << detID <<" with deposited energy = "<<edep/MeV<< G4endl;
     }
     }else{
     (*CaloCollection)[LayerID[detID]]->AddEdep(edep/MeV,tkID);


    if(verboseLevel>1){
      G4cout << " Energy added to Calo Layer "
             << detID <<" adding this energy deposit = "<<edep/MeV<< G4endl;
    }
 }
  return true;
}
```

```
void CalorimeterSD::EndOfEvent(G4HCofThisEvent*HCE){
  static G4int HCID = -1;
  if(HCID<0)
    HCID = GetCollectionID(0);
  HCE->AddHitsCollection( HCID, CaloCollection );
}

void CalorimeterSD::clear(){
}

void CalorimeterSD::DrawAll(){
}

void CalorimeterSD::PrintAll(){
}

G4double CalorimeterSD::BirksAttenuation(const G4Step* aStep){
  double weight = 1.;
  double charge = aStep->GetPreStepPoint()->GetCharge();
  if (charge != 0. && aStep->GetStepLength() > 0){
    G4Material* mat = aStep->GetPreStepPoint()->GetMaterial();
    double density = mat->GetDensity();
    double dedx    = aStep->GetTotalEnergyDeposit()/aStep->GetStepLength();
    if(mat->GetName()=="Scintillator"||mat->GetName()=="Polystyrene"){
      double rkb     = birk1scint/density;
      double c       = birk2scint*rkb*rkb;
      if (std::abs(charge) >= 2.) rkb /= birk3scint; // based on alpha particle
data
      weight = 1./(1.+rkb*dedx+c*dedx*dedx);
    }else{
      double rkb     = birk1crystal/density;
      double c       = birk2crystal*rkb*rkb;
      if (std::abs(charge) >= 2.) rkb /= birk3crystal; // based on alpha particl
e data
      weight = 1./(1.+rkb*dedx+c*dedx*dedx);
    }
  }
  return weight;
}
```