```
//
// ********************************************************************
// * 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 "VetoSD.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"

VetoSD::VetoSD(G4String name):G4VSensitiveDetector(name){
  collectionName.insert("vetoCollection");
  useBirks=false;
  fMessenger = new VetoSDMessenger(this);

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

}

VetoSD::~VetoSD()
{;}

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

  VetoID.clear();

  verboseLevel = 0;
}

G4int VetoSD::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("VetoYDown"))   // -X
    detID= 1E3 + 4*1E2 + 1*1E1 + 1*1E0;
  if(!volumeID.compare("VetoYUp")) // +X
    detID= 1E3 + 4*1E2 + 1*1E1 + 2*1E0;
  if(!volumeID.compare("VetoXLeft"))  // -Y
    detID= 1E3 + 4*1E2 + 2*1E1 + 1*1E0;
  if(!volumeID.compare("VetoXRight"))  // +Y
    detID= 1E3 + 4*1E2 + 2*1E1 + 2*1E0;

  if(!volumeID.compare("VETOBottScintLayer")) // Z
    detID= 1E3 + 4*1E2 + 3*1E1 + 0*1E0;

  return detID;
}

G4bool VetoSD::ProcessHits(G4Step*aStep,G4TouchableHistory*){
  G4double edep = aStep->GetTotalEnergyDeposit();
  G4int tkID = aStep->GetTrack()->GetTrackID();
  if(verboseLevel>1) G4cout << "Next step edep(MeV) = " << edep/MeV << G4endl;
  if(edep==0.) return false;

  if(useBirks)
    edep *= BirksAttenuation(aStep);

//   G4VPhysicalVolume* physVol = aStep->GetPreStepPoint()->GetPhysicalVolume();
//   G4String volume = physVol->GetName();

  G4int detID;
  detID=GetDetID(aStep);


  if(VetoID.find(detID)==VetoID.end()){
    //    CaloHit* vetoHit = new CaloHit(volume);
    CaloHit* vetoHit = new CaloHit(detID);
    vetoHit->SetEdep(edep/MeV,tkID);
    G4int icell = VetoCollection->insert(vetoHit);
    VetoID[detID] = icell - 1;
    if(verboseLevel>0){
      G4cout << " New Calorimeter Hit on VetoID "
             << detID << G4endl;
    }
  }else{
    (*VetoCollection)[VetoID[detID]]->AddEdep(edep/MeV,tkID);
    if(verboseLevel>0){
      G4cout << " Energy added to VetoID "
             << detID << G4endl;
    }
  }
  return true;
}

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

void VetoSD::clear(){
}

void VetoSD::DrawAll(){
}

void VetoSD::PrintAll(){
}

G4double VetoSD::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();
    double rkb     = birk1scint/density;
    double c       = birk2scint*rkb*rkb;
    if (std::abs(charge) >= 2.) rkb /= birk3scint; // based on alpha particle da
ta
    weight = 1./(1.+rkb*dedx+c*dedx*dedx);
  }
  return weight;
}
```