```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
// $Id: B2TrackerSD.cc 87359 2014-12-01 16:04:27Z gcosmo $
//
/// \file B2TrackerSD.cc
/// \brief Implementation of the B2TrackerSD class
//refer:https://www.slac.stanford.edu/xorg/geant4/KISTI2019/HandsOn3/#ex1s6
#include "G4VSensitiveDetector.hh"
#include "micalCal1SD.hh" //
#include "G4HCofThisEvent.hh"
#include "G4Step.hh"
#include "G4ThreeVector.hh"
#include "G4SDManager.hh"
#include "G4ios.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4RunManager.hh"//..
#include "micalRunAction.hh"//..
#include "micalDetectorParameterDef.hh"
#include <iostream>
#include <fstream>
 #include "TRandom2.h"
using namespace std;


//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
//B1TrackerSD::B1TrackerSD(B1RunAction* runAction)
//: G4UserEventAction(),
  //fRunAction(runAction)

//{}
//////////////////////////////////////////////////

micalcal1SD::micalcal1SD(const G4String name)
  : G4VSensitiveDetector(name),
    cal1Collection(NULL),
    // SWidth(0),
    // fEnvelopeBox(0),
    numberInCell(20000),
    Counter(0),
    InCell(0)
{
  G4String HCname;
        paradef = micalDetectorParameterDef::AnPointer;
  collectionName.insert(HCname="cal1Collect");
  cal1SDMessenger = new micalcal1SDMessenger(this);
  pAnalysis = MultiSimAnalysis::AnPointer;
```



```
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

micalcal1SD::~micalcal1SD()
{
//delete fMessenger;
}
//.....................................................................

void micalcal1SD::Initialize(G4HCofThisEvent* hce)
{
  //cout<<"micalcal1SD::Initialize start"<<endl;
  //InCell =0;


  for(int op=0; op<3;op++) {
    partopscint[op] = paradef->partopscint[op];
  }

  AirGapScintTop= paradef->AirGapScintTop;


  for(int ji=0; ji<4;ji++) {
    for(int kl=0; kl<3; kl++) {
      Phys_TopScint_GPos[ji][kl]= paradef->Phys_TopScint_GPos[ji][kl];

    }//kl
  }//ji


  for(int jk=0; jk<3;jk++) {
    for(int mn=0; mn<3; mn++) {

      Phys_SideScint_R_GPos[jk][mn]= paradef->Phys_SideScint_R_GPos[jk][mn];
      Phys_SideScint_L_GPos[jk][mn]= paradef->Phys_SideScint_L_GPos[jk][mn];
      Phys_SideScint_D_GPos[jk][mn]= paradef->Phys_SideScint_D_GPos[jk][mn];
    }//mn
  }//jk

 // cout<<"parscint[0]"<<parscint[0]<<endl;
  // cout<<"parscint[1]"<<parscint[1]<<endl;
 //cout<<"parscint[2]"<<parscint[2]<<endl;
  //if (!fEnvelopeBox)
  //{
    //G4LogicalVolume* envLV
      //= G4LogicalVolumeStore::GetInstance()->GetVolume("logicScint_1cm");
    //if ( envLV ) fEnvelopeBox = dynamic_cast<G4Box*>(envLV->GetSolid());
  //}

  //if ( fEnvelopeBox ) {
    //pargas[0] = fEnvelopeBox->GetXHalfLength();
      //pargas[1] = fEnvelopeBox->GetYHalfLength();
    //pargas[2] = fEnvelopeBox->GetZHalfLength();
  //}
  //else  {
    //G4ExceptionDescription msg;
    //msg << "Envelope volume of box shape not found.\n";
    //msg << "Perhaps you have changed geometry.\n";
    //msg << "The gun will be place at the center.";
    //G4Exception("B1PrimaryGeneratorAction::GeneratePrimaries()",
     //"MyCode0002",JustWarning,msg);
  //}
  //cout<<pargas[0]<<endl;
   //cout<<pargas[1]<<endl;
    //cout<<pargas[2]<<endl;
//.....................................................................
.....................
```

```
  // Create hits collection

  cal1Collection
    = new micalcal1HitsCollection(SensitiveDetectorName, collectionName[0]);

  //cout<<"SensitiveDetectorName" <<SensitiveDetectorName<<endl;
  //cout<<"collectionName[0]" <<collectionName[0]<<endl;
  // Add this collection in hce

  G4int hcID
    = G4SDManager::GetSDMpointer()->GetCollectionID(collectionName[0]);
  hce->AddHitsCollection( hcID, cal1Collection );

  paradef =  micalDetectorParameterDef::AnPointer;


  //cout<<"micalcal1SD::Initialize ends         "<<endl;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

G4bool micalcal1SD::ProcessHits(G4Step* aStep,
                                G4TouchableHistory*)  //called everytime when th
e particle hts the sensitive dete
{

  //cout<<endl<<"micalcal1SD::ProcessHits start"<<endl;

  G4double edep = aStep->GetTotalEnergyDeposit()/keV;

  G4TouchableHistory* theTouchable = (G4TouchableHistory*)( aStep->GetPreStepPoi
nt()->GetTouchable() );

  int pdgid = aStep->GetTrack()->GetDefinition()->GetPDGEncoding();
  int level = theTouchable->GetHistoryDepth();

  //cout<<"level="<<level<<endl; //level=4
  //for(int ij=0; ij<level+1; ij++) { //level=4

  //cout<<ij<<" volname "<<theTouchable->GetVolume(ij)->GetName()
  //<<" copyNo. "<<theTouchable->GetCopyNumber(ij) <<"edep "<< edep << "pid "<<p
dgid<< "nInLA:"<<theTouchable->GetCopyNumber(1)<<endl;
  //}

  G4ThreeVector parmom = aStep->GetTrack()->GetMomentum();

  double momentum= parmom.mag();
  double polang = parmom.theta();
  double aziang = parmom.phi();

  //cout<<"edep "<< edep<<endl;
  if (edep==0.)  return false;

  //We are simulating a  detector that will trigger only if some energy has been
 deposited (i.e. via ionization),
  //for example if a neutron passes through the detector (without making interac
tions) its passage should not be recorded.
  // Check the energy deposited in the step, if zero do not do anything.


  G4ThreeVector glbpos =0.5*(aStep->GetPreStepPoint()->GetPosition() + aStep->Ge
tPostStepPoint()->GetPosition()); //avg distance betwen the two

  G4ThreeVector tmp;

  tmp = (1/m)*glbpos;
  // tmp.y() = (1/m)*glbpos.y();
  // tmp.z() = (1/m)*glbpos.z();
```



```
  //cout<<"tmpx" << tmp.x() <<endl;
  //cout<<"tmpy" << tmp.y() <<endl;
  //cout<<"tmpz" << tmp.z() <<endl;

  G4int nScntStrp = theTouchable->GetCopyNumber(0);
  G4int nInLA = theTouchable->GetCopyNumber(1) ;
  G4String loc = theTouchable->GetVolume(1)->GetName();

  //cout<<"loc " << loc<<endl;
  //cout<<"nInLA " << nInLA<<endl;
  //cout<<"nScntStrp " << nScntStrp<<endl;


  //cout<<"check_Process hits_1"<<endl;
  G4String Cond_one = "physiTopScint_1cm";
  G4String Cond_two = "physiTopScint_2cm";
  G4String Cond_three = "physiSideScint_L";
  G4String Cond_four = "physiSideScint_R";
  G4String Cond_five ="physiSideScint_D";
  G4int loc_no=-1;


  if (loc == Cond_one || loc == Cond_two){
    loc_no = 0;
    //  PhyVolGlPos[0]= paradef->Phys_TopScint_GPos[nInLA][0];
    // PhyVolGlPos[2]= paradef->Phys_TopScint_GPos[nInLA][2];

  } else if (loc == Cond_three){

    loc_no = 1;
    // PhyVolGlPos[0]= paradef->Phys_SideScint_L_GPos[nInLA][0];
    //PhyVolGlPos[1]=
    // PhyVolGlPos[2]= paradef->Phys_SideScint_L_GPos[nInLA][2];
  } else if (loc == Cond_four){
    loc_no = 2;
    // PhyVolGlPos[0]= paradef->Phys_SideScint_R_GPos[nInLA][0];
    //PhyVolGlPos[1]=
    // PhyVolGlPos[2]= paradef->Phys_SideScint_R_GPos[nInLA][2];
  } else if (loc == Cond_five){
    loc_no = 3;
    // PhyVolGlPos[1]= paradef->Phys_SideScint_D_GPos[nInLA][1];
    //PhyVolGlPos[1]=
    // PhyVolGlPos[2]= paradef->Phys_SideScint_D_GPos[nInLA][2];
  }

  //cout<<"loc_no " << loc_no<<endl;

  //cout<<"check_Process hits_2"<<endl;


  G4double atime = aStep->GetPreStepPoint()->GetGlobalTime()/(ns);
  cout<<" -------"<<atime<<endl;
  G4int nInT = G4int(atime/125.);
  cout<<nInT<<endl;
  G4ThreeVector localpos = theTouchable->GetHistory()->GetTopTransform().Transfo
rmPoint(glbpos);
  // double tra_LPosy = localpos.y() + partopscint[2]; // shift of origin
 //To reduce the usage of memory.............................
 // unsigned int ScntStrpid;   // 16 bits  declared in hh   15 14 13 12 11 10 9
8 7 6 5 4 3 2 1 0

  unsigned int IdSiPM;

  //

  IdSiPM = loc_no;     //occupies 2 bits
  //  cout<<IdSiPM      <<endl;
  IdSiPM<<=2;
  // cout<<IdSiPM       <<endl;
```

```
  IdSiPM +=nInLA;       //occupies 2 bits
  // cout<<IdSiPM       <<endl;
  IdSiPM<<=7;
  // cout<<IdSiPM       <<endl;
  IdSiPM +=nScntStrp;// 7 bits space
  // cout<<IdSiPM       <<endl;
  IdSiPM<<=2;
  for(int nSiPM=0;nSiPM<4;nSiPM++){
    //cout<<"Id "<<IdSiPM<<endl;
    // cout<<"check_Process hits_3"<<endl;
    int oldCellId = -1;

    //  cout<<"oldCellId "<<oldCellId<<endl;
    // cout<<"InCell "<<InCell<<endl;
    for (int ij=0; ij<InCell; ij++) {
      // cout<<"ij "<<ij<<endl;
      if (IdSiPM ==CellDetID[ij]) {
        oldCellId = ij;//cout<<"oldCellId"<<oldCellId<<endl;
      }
    }

    if (oldCellId ==-1 && InCell <numberInCell -1 ) {
      // cout<<"New Hit"<<Counter<<endl;
      Counter++;
      micalcal1Hit* newHit = new micalcal1Hit();
      // B1TrackerHit* newHit = new B1TrackerHit();

      newHit->SetHitId(IdSiPM);

      int pdgid = aStep->GetTrack()->GetDefinition()->GetPDGEncoding();
      //newHit->SetPos(aStep->GetPostStepPoint()->GetPosition());
      newHit->SetpdgId(pdgid);
      newHit->SetEdep(edep);
      newHit->SetTime(atime);
      newHit->SetPos(glbpos); // 0.5*(aStep->GetPreStepPoint()->GetPosition() +
aStep->GetPostStepPoint()->GetPosition()));
      //newHit->SetLocalXPos(localpos.x());
      //newHit->SetLocalYPos(localpos.y());
      newHit->SetLocalPos(localpos);
      //        newHit->SetSigTimDif();
      newHit->SetMom( aStep->GetTrack()->GetMomentum());

      InCell = cal1Collection->insert( newHit );
      //  cout<<"InCell "<<InCell<<endl;
      CellDetID[InCell-1] = IdSiPM;      //?
      //  cout<<"CellDetID[InCell-1] "<<CellDetID[InCell-1]<<endl;
      //.....................................................................
.............................

    }

    //cout<<"check_Process hits_4"<<endl;
    //cout<<"oldCellId "<<oldCellId<<endl;
    if (oldCellId >=0) {
      (*cal1Collection)[oldCellId]->AddEdep(edep);
      if (atime <(*cal1Collection)[oldCellId]->GetTime()) {

        (*cal1Collection)[oldCellId]->SetTime(atime);
      }
    }

    //cout<<"check_Process hits_5"<<endl;
    // cout<<"detid:"<<detid<<endl;
    //cout<<"micalcal1SD::ProcessHits ends "<<endl;
    IdSiPM++;

  }
  return true;
```



```
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......

void micalcal1SD::EndOfEvent(G4HCofThisEvent*)
{
  double sigspeed = paradef->sigspeed;//16.3cm/ns
  //    cout<<"micalcal1SD::EndOfEvent START "<<endl;
  InCell = 0;
        G4float ScintHitGPos[3];

        //int ijk=0;
        G4float PhyVolGlPos[3];
        //int countsy=0;
  //micalRunAction* fRunAction = micalRunAction::AnPointer; //# to store data in
 the same tree created in runaction


  // if ( verboseLevel>1 ) {
  G4int nofHits = cal1Collection->entries();
  //G4cout << G4endl
  //<< "-------->Hits Collection: in this event there are " << nofHits
  //<< " hits in the tracker chambers: " << G4endl;
  //for ( G4int i=0; i<nofHits; i++ ) (*cal1Collection)[i]->Print();
  cout<<"nofHits::"<<nofHits<<endl;
  pAnalysis->CMV_nsimhit = cal1Collection->entries();
  //cout<<"check-1"<<endl;
  for (int ij=0; ij<cal1Collection->entries(); ij++) {
    // cout<<"ij"<<ij<<endl;
    unsigned int SiPMId = (*cal1Collection)[ij]->GetHitId();
    G4ThreeVector posvec = (*cal1Collection)[ij]->GetPos();  // get glb poistion
    int pdgid = (*cal1Collection)[ij]->GetpdgId();
    double atime = (*cal1Collection)[ij]->GetTime();          //detid is stored
in Sethitid
    double Edep = (*cal1Collection)[ij]->GetEdep();
    G4ThreeVector localpos = (*cal1Collection)[ij]->GetLocalPos();


    cout<<"ij "<<ij<<" Edep "<<Edep<<" pdgid "<<pdgid<<" posvec "<<posvec<<" SiPMId "<<Si
PMId <<endl;
    //cout<<"check1"<<endl;

    pAnalysis->CMV_detid[ij] = SiPMId;
    pAnalysis->CMV_simpdgid[ij] =  pdgid;
    pAnalysis->CMV_simtime[ij] = atime;
    pAnalysis->CMV_simenr[ij] = Edep;
    pAnalysis->CMV_simposx[ij] = posvec.x();
    pAnalysis->CMV_simposy[ij] = posvec.y();
    pAnalysis->CMV_simposz[ij] = posvec.z();


    int LayerNo, loc_no, ScntStrpNo, SiPMNo;
    //cout<<"ScntStrpid"<<ScntStrpid<<endl;

    SiPMNo = SiPMId%4; //2^2
    SiPMId>>=2;
    ScntStrpNo= SiPMId%128; //2^7
    // cout<<"ScntStrpNo"<<ScntStrpNo<<endl;
    SiPMId>>=7;
    LayerNo = SiPMId%4;
    // cout<<"LayerNo"<<LayerNo<<endl;
    SiPMId>>=2;
    loc_no = SiPMId;
    //cout<<"Loc_No"<<loc_no<<endl;

    pAnalysis->CMV_digiSiPMNo[ij] =SiPMNo;
    pAnalysis->CMV_digiScntStrpNo[ij] = ScntStrpNo;
    pAnalysis->CMV_digiLayerNo[ij] =  LayerNo;
    pAnalysis->CMV_digiLocNo[ij] =      loc_no;
```

```
    cout<<"local position "<<localpos<<endl;
    cout<<" SiPMNo "<<SiPMNo <<" ScntStrpNo "<<ScntStrpNo <<" LayerNo "<<LayerNo <<" L
oc_No "<<loc_no<<endl;

    double sigpos;
    if(loc_no==3) {
      sigpos=localpos.x();
      // cout<<"signal position "<<sigpos<<endl;
    } else {
      sigpos = localpos.y();
      // cout<<"signal position "<<sigpos<<endl;
    }

    double sigtim = atime +  (partopscint[1]+ sigpos)/sigspeed + gRandom->Gaus(0
,1)*ns;
    double  sigtimdash = atime + (partopscint[1]- sigpos)/sigspeed + gRandom->Ga
us(0,1)*ns ;

    if(SiPMNo ==0 || SiPMNo==1) {
      cout<<atime<<" signal time "<<sigtim/ns<<endl;
      pAnalysis->CMV_digisigtim[ij]= sigtim;
    } else {
      cout<<atime<<" signal time "<<sigtimdash/ns<<endl;
      pAnalysis->CMV_digisigtim[ij]=sigtimdash; //GMA it was an error
    }

    if ( loc_no == 0){
      PhyVolGlPos[0]= paradef->Phys_TopScint_GPos[LayerNo][0];
      PhyVolGlPos[1]=paradef->Phys_TopScint_GPos[LayerNo][1];
      PhyVolGlPos[2]= paradef->Phys_TopScint_GPos[LayerNo][2];
    } else if (loc_no == 1){
      PhyVolGlPos[0]= paradef->Phys_SideScint_L_GPos[LayerNo][0];
      PhyVolGlPos[1]=paradef->Phys_SideScint_L_GPos[LayerNo][1];
      PhyVolGlPos[2]= paradef->Phys_SideScint_L_GPos[LayerNo][2];
    } else if (loc_no == 2){
      PhyVolGlPos[0]= paradef->Phys_SideScint_R_GPos[LayerNo][0];
      PhyVolGlPos[1]=paradef->Phys_SideScint_R_GPos[LayerNo][1];
      PhyVolGlPos[2]= paradef->Phys_SideScint_R_GPos[LayerNo][2];
    } else if (loc_no == 3){
      PhyVolGlPos[1]= paradef->Phys_SideScint_D_GPos[LayerNo][1];
      PhyVolGlPos[0]=paradef->Phys_SideScint_D_GPos[LayerNo][0];
      PhyVolGlPos[2]= paradef->Phys_SideScint_D_GPos[LayerNo][2];
    }

    //cout<<"loc_no " << loc_no<<endl;


    // cout<<" ---------"<<PhyVolGlPos[0]<<endl;

    if(loc_no == 0){

      ScintHitGPos[0] = PhyVolGlPos[0] - 0.5*((88*2*partopscint[0])+(89*AirGapSc
intTop)) +( ScntStrpNo+1)*(AirGapScintTop)+(2*ScntStrpNo+1)*partopscint[0];
      ScintHitGPos[1] =PhyVolGlPos[1];
      ScintHitGPos[2] = PhyVolGlPos[2];
      // pAnalysis->ScntStrpXPos[LayerNo].push_back(ScintHitGPos[0]);
      // pAnalysis->ScntStrpNo[LayerNo].push_back(ScntStrpNo);

      cout<<" ---------"<<ScintHitGPos[0]<<endl;
    } else if(loc_no == 1 ||loc_no == 2 ){
      ScintHitGPos[0] = PhyVolGlPos[0] ;
      ScintHitGPos[1]= PhyVolGlPos[1];
      ScintHitGPos[2] = PhyVolGlPos[2]-0.5*((40*2*partopscint[0])+(41*AirGapScin
tTop))+( ScntStrpNo+1)*(AirGapScintTop)+(2*ScntStrpNo+1)*partopscint[0];

    }else if(loc_no == 3){
      ScintHitGPos[0]=PhyVolGlPos[0];
      ScintHitGPos[1] = PhyVolGlPos[1] ;
```



```
      ScintHitGPos[2] = PhyVolGlPos[2]-0.5*((40*2*partopscint[0])+(41*AirGapScin
tTop))+( ScntStrpNo+1)*(AirGapScintTop)+(2*ScntStrpNo+1)*partopscint[0];

    }

    pAnalysis->CMV_digiposx[ij] =       ScintHitGPos[0];
    pAnalysis->CMV_digiposy[ij] =       ScintHitGPos[1];
    pAnalysis->CMV_digiposz[ij] =       ScintHitGPos[2];
  }//for ij

//cout<<"micalcal1SD::EndOfEvent ends "<<endl;

}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo......
```