Sunnybrook Research Institute

Geant4 Cargo Scanning Monte-Carlo Simulation Project Report

Ruiheng Su

March, 06, 2020

Contents

1 Acknowledgements

This work was enabled by the availability of the course material for a Geant4 tutorial given to graduate students at *Lund University, Sweden*, from September 3 to September 7, 2018 [3], which I used as my primary resource in learning to construct Monte-Carlo simulation applications using the Geant4 tool kit.

Further, I was learning the C++ language while completing this project as a research assistant at *Odette Cancer Centre, Ontario, Canada.* I am thankful to my supervisor, *Dr. Geordi Pang* for this learning opportunity and exposure to the field of medical physics.

Figure 1: This is a photo of *Odette Cancer Centre, Ontario*, which is a part of *Sunnybrook Health Sciences Centre*, where I worked during my first co-op term as a second-year engineering physics undergraduate at the *University of British Columbia.* (Image Source: [2])

2 Introduction

2.1 The Geant4 Tool Kit

Geant4 is a general-purpose Monte-Carlo simulation tool kit for particle-matter interactions written in the C++ language. Geant4 is a tool kit in that it provides various concrete and abstract classes which require user implementation to be built into simulation applications.

While Geant4 was originally used in high energy and accelerator physics, it is applicable to medical and space science as well.

2.2 Project Description

This project is based on the course exercise description provided by the Geant4 tutorial which can found along with the course notes [3].

The context of this project involves the radiographic inspection of a cargo container, a technique used in international boarder agencies to detect special nuclear materials and other types of suspicious cargo.

This project will exercise the four fundamental aspects of a Geant4 application,

- Defining geometry: material, volume
- Defining physics: particles involved, physical processes/models, production thresholds
- Defining how an event starts: primary track generation
- Extracting useful information

by constructing a simulation application to produce images that result from the individual exposure of a shipping container geometry to uniform gamma-ray and neutron sources, as well as a muon source.

2.3 Source Code

In addition to the appendix, the source code for this project can be found on GitHub:

`https://github.com/sillyPhotons/G4_Cargo`

3 Simulation Construction

3.1 Geometry

The ability to describe the geometry is crucial to an accurate simulation. By implementing the `G4VDetectorConstruction` abstract class, the following objects were defined in a world volume made of air ("`G4_AIR`"), a predefined Geant4 material, with dimensions 20 m × 6 m × 6 m,

- A 10 m × 3 m × 3 m aluminium ("`G4_Al`") container with 5 mm thick walls, filled with air
- One 10 m × 3 m × 10 cm and two 10 m × 10 cm × 3 m detectors made of cesium iodide ("`G4_CESIUM_IODIDE`") for muon and gamma-ray detection
- A 10 m × 3 m × 10 cm detector made of scintillating material ("`G4_PLASTIC_SC_VINYLTOLUENE`") for neutron detection
- An 8 × 15 × 5 cuboid consisting of arranged gold ("`G4_Au`") bricks, spanning a box with dimensions 2 m × 2.25 m × 1.5 m
- Two wooden (user-defined) cylindrical barrels of outer radius 0.5 m, inner radius 0.45 m, height 1 m, completely filled with water ("`G4_WATER`").
- Three humanoid phantoms, made of skeletal muscle ("`G4_MUSCLE_SKELETAL_ICRP`")
- A ball made of uranium ("`G4_U`"), of radius 38 cm, placed within a 40 cm^3 cubic radiation shield made of lead ("`G4_Pb`") with 5 mm thick walls

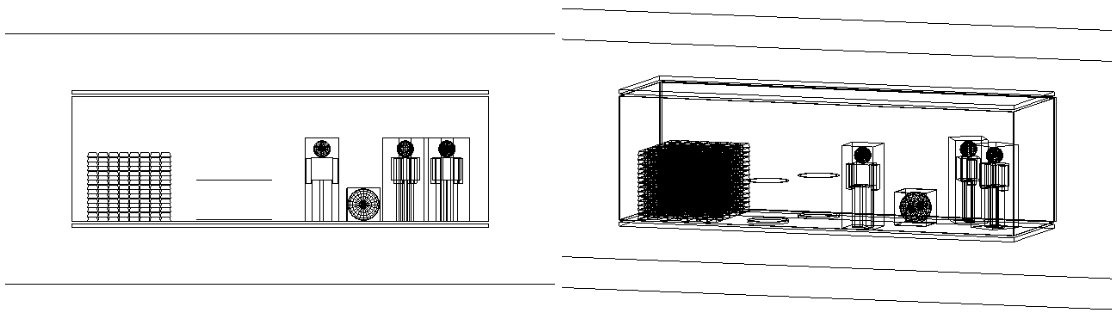


Figure 2: This figure shows a wireframe visualization of the simulation geometry as listed

above. There are two muon detector plates visible on the top and bottom surface of the cargo container, and the gamma/neutron detector is visible in the image on the right, positioned behind the container.

3.2 Particle Source

Each particle is generated using a `G4ParticleGun` object in the implementation of the `G4VUserPrimaryGeneratorAction` class.

To simulate a large area parallel beam, the particle gun uniformly samples, per event, a location on the shipping container surface, and generates a single particle with momentum orthogonal to the container surface at a location 3 m away.

In addition, the `G4GenericMessenger` is used in the a user-defined method, `DefineCommands()`, to implement a UI command which allows the user to change the simulation beam type to one of the following at run-time:

1. 5 MeV Gamma-ray parallel beam
2. 15 MeV Neutron parallel beam
3. 4 GeV Muon parallel beam

3.3 Physics

All particles and its associated processes must be defined in Geant4 through the implementation of the `G4VUserPhysicsList` abstract class. To implement a custom physics list requires the implementation of virtual methods which define all particles present in the simulation, the list of all physics processes to be used for each particle, and optionally, secondary production cut-off values which gives the opportunity to improve simulation efficiency.

However, an in depth understanding of the simulation setting is required to accurately implement a fully customized physics list and such lists can be difficult to maintain. For our purposes, we will use the prepackaged physics list `"FTFP_BERT_HP"`, which is a suitable choice due to its operating energy range and support for high precision neutron model for neutrons below 20 MeV.

4 Imaging with Gammas and Neutrons

4.1 Scoring

To produce an image, command based scorers were used to define a scoring mesh over the logical volume of the γ/n detector, with dimensions identical to the detector volume, and 250 bins along x, 75 bins along y and 1 bin along z.

However, to use the same detector logical volume for the simulation of both particles, a UI-command was defined using a `G4GenericMessenger` object to enable the configuration of the detector material between `G4_CESIUM_IODIDE` or `G4_PLASTIC_SC_VINYLTOLUENE` at runtime.

4.2 Running the Simulation

The use of macro files enables the definition of a sequential set of UI-commands to be executed before and after the `beamOn` command. Following is the commented `.mac` file to simulate with 500000 events using γ or n.

```
/run/initialize
/run/numberOfThreads 4

# choosing the particle type
/Cargo/gun/chooseGun <integer>
# choosing the material of the detector
/Cargo/Gamma_Neutron/chooseMat <integer>

# verbose settings
/control/verbose 0
/run/verbose 1
/event/verbose 0
/tracking/verbose 0

# creation of scoring mesh
/score/create/boxMesh gammaMesh
/score/mesh/boxSize 500. 150 5. cm
/score/mesh/nBin 250 75 1
/score/mesh/translate/xyz 0. 0. -160. cm
/score/quantity/energyDeposit edep eV
```

```
/score/close

# print the number of events passed
/run/printProgress 100
/run/beamOn 500000

# put simulation results into a csv file
/score/dumpQuantityToFile gammaMesh edep gammaMeshEdepNeutron.csv
```

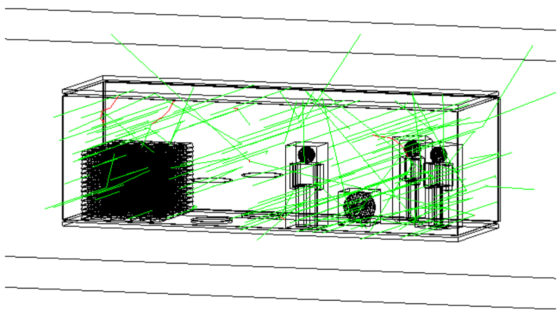


Figure 3: The figure above shows the the exposure of the shipping container to gamma-rays.

A python script was written to read the `.csv` file, and uses the `imshow` function of the `matplotlib.pyplot` module to display the 2D histogram using a sequential color map.

4.3 Gamma

After a run of 500000 events, the following histogram was produced

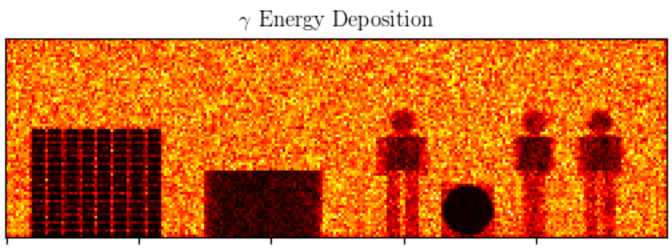


Figure 4: Above the is the resulting histogram of the gamma-ray energy deposition on the cesium iodide detector. It is clear that the brightness of the pixel is proportional to the energy

of the gamma-ray since the pixels have a brighter color overall when the gamma-ray trajectory does not interact with one of the objects in the container.

For a energy of 5 MeV, we know that the predominant interaction between γ and matter is Compton scattering. It is known that the Compton effect is mostly independent of the atomic number, as well as photon energy, and is mostly dependent on the density of the material in question. For this reason, the fainter appearance of the human phantom compared to the uranium orb and the gold blocks is justified.

4.4 Neutrons

After a run of 500000 events, the following histogram was produced

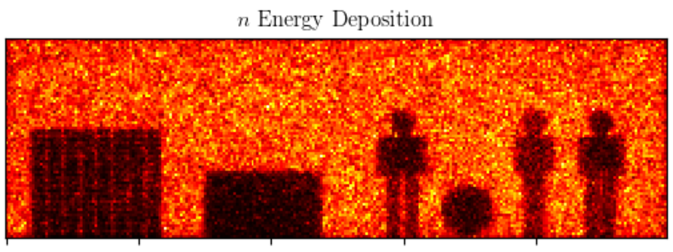


Figure 5: Above the is the resulting histogram of the neutron energy deposition in the vinyl-toluene scintillator detector.

Neutron interactions are strongly dependent on the energy of the neutron, and that it mainly goes through scattering or absorption. During the absorption interaction, the incoming neutron is absorbed by the nucleus of the interacting atom, and energy is released in the form of gamma-rays. The additional introduction of gamma-rays may be the cause of decreased sharpness of the images compared to the histogram resulting from γ radiation, which can be seen present in the gold cluster.

Further, note that in both gamma-ray and neutron simulations, the energy deposition in the detector is directly scored, which neglects any possible optical conversion loss or noise introduced in actual detectors such as direct or indirect conventional flat-panel imagers used in radiography. Then, the resolution of the image is only dependent on the ability of the material used for the simulated detector to absorb radiation. For example, an image similar to figure 5 can also be produced even when the material of the detector is changed to be completely made of lead.

5 Imaging with Muons

5.1 Method

The proposed method to reconstruct an image of the shipping container content using giga-voltage muons is by plotting a two-dimensional histogram, with its weight proportional to the muon scattering angle.

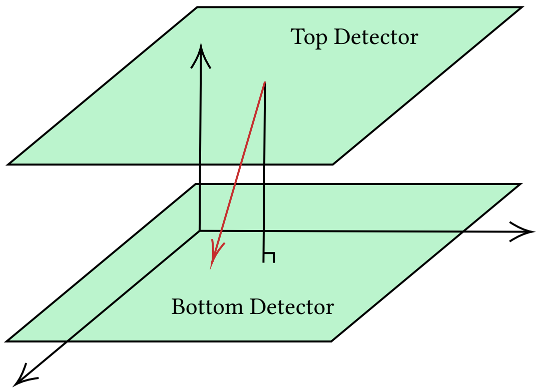


Figure 6: This figure shows the muon detector on the top and bottom of the cargo container as two planes, with the top point marking the interaction of a particular muon with the top detector, and the bottom point marking the interaction of that muon with the bottom detector, if its interaction with the container and its contents did not cause its trajectory to be scattered outside of detector surface. We define the scattering angle to be the acute angle between the red vector, and the black reference vector, which is normal to both detector planes.

5.1.1 Computing the Scattering Angle

In order to compute the scattering angle, the position of interaction of the muon with the first and second detector must be known. Let $\mathbf{i} = (a, b, c)$ be the coordinates of the first interaction, and $\mathbf{f} = (\alpha, \beta, \gamma)$ be the coordinates of the second interaction if it was detected. Then, the scattering angle can be found by finding the angle between the vectors $\mathbf{v} = (a, \beta, c) - \mathbf{i}$ and $\mathbf{u} = \mathbf{f} - \mathbf{i}$ from the definition of the inner product in Euclidean space on $\mathbb{R}^3$.

5.1.2 Recording First and Second Interaction

In order to record the position of the first and second interaction, a `MuonDetectorSD` object, after implementing the `G4VSensitiveDetector` class, is assigned to the top and bottom detector in the `ConstructSDandField()` method of the detector construction class.

The purpose assigning a sensitive detector is to construct `hit` objects, which contains information regarding a particular interaction, and is stored in the `G4HCofThisEvent` object of its associating `G4event`. Within the `ProcessHits()` method of the sensitive detector class, we use the condition

```
preStepPoint->GetStepStatus() == fGeomBoundary
```

to determine whether this particular step is the first interaction with the sensitive detector.

By also implementing the `G4VUserRunAction` class and the `G4VUserEventAction` class, the Geant4 analysis functionality can be used to create and write per event data to a 2D histogram, which was visualized using the root software.

5.2 Running the Simulation

To run the simulation using the 4 GeV muon parallel beam, the macro file is much simpler as it is not necessary to define a scoring mesh, and that the creation and closing of the generated histogram are automated by the implementation of the `G4VUserRunAction` class.

```
/run/initialize
/run/numberOfThreads 4
/Cargo/gun/chooseGun 3

/run/printProgress 100
/run/beamOn 500000
```

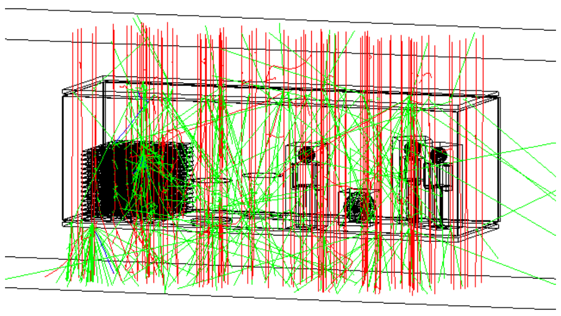
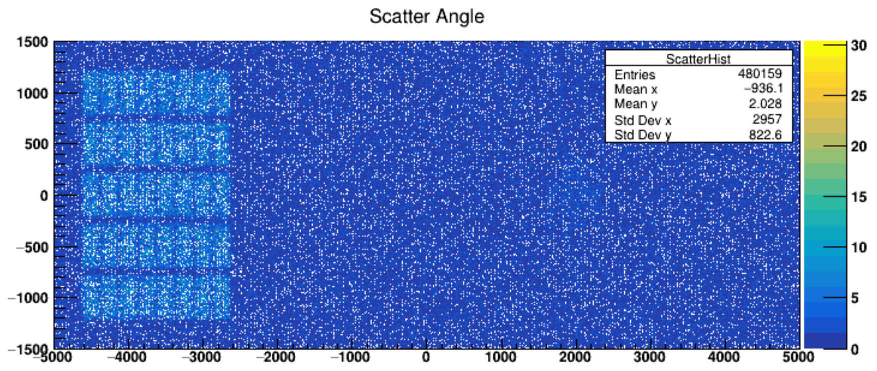


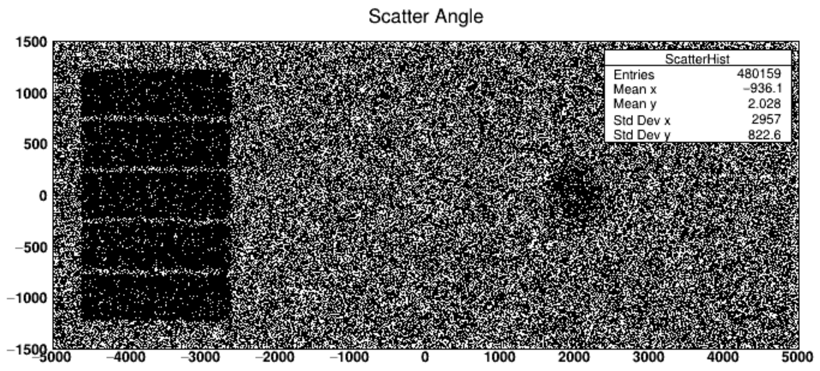
Figure 7: This figure shows the interaction of 100 muons with the shipping container. We can see that the red muons trajectories produce numerous gamma-rays trajectories in green.

5.3 Muons

After a run of 500000 events, the following histograms were produced



(a)



(b)

Figure 8 (a/b): This figure shows the histograms of x, y coordinates of the first muon interaction weighted by its scattering angle using two visualization schemes. As we can see from the legend, the number of entries in the histogram is 480159, which is approximately 96 percent of the total number of events generated. Further, only the uranium orb and the blocks of gold were visible in the histogram.

As we can see from figure 8a, while most scattering angles are less than 5 percent, greater scattering angles are seen in the area of the gold blocks due to its high density and stacked structure. Thus, it can be reasoned that muons tend to interact with denser materials, and materials made of elements of a greater atomic number since only the gold blocks and the uranium orb were visible in histogram.

It is known that muons observed on earth are primarily produced by primary cosmic rays in the upper atmosphere and arrives at sea level at a flux of 1 muon per minute per cm^2 [1]. Since the detector spans an incident area of 300000 cm^2, it would require approximately 1 minute and 40 seconds replicate the number of events used in the simulation, disregarding the complication that not all muons arrive incident to the detector, and potential for other background radiation sources.

6 Conslusion

This project successfully demonstrated a basic application of the Geant4 tool kit by exercising the four fundamental aspects of a Geant4 simulation by implementing the basic contain and detector geometry, using a prepackage physics list, defining the particle source, scoring γ/n energy deposition, batch calculation of muon scattering angle, and visualization of extracted data using 2D histograms.

Further, this work shows that even with a moderate number of γ, n, μ, it is possible to generate an intelligible image of a cargo container. However, it is noted that since the energy deposition of the gamma-rays were directly scored, the image produced is idealized as it neglects any possible imaging artifacts. Further, is clear that the muon imaging is impractical in cargo scanning since it is only able to image dense materials, and requires more time and effort for reconstruction, compared to a radiographic image.

For future projects, a scanning beam may be used to mimic an actual cargo scanning gantry, analytical techniques may be applied to evaluate the safety of such equipment on containers that may contain living creatures, and more realistic model of the atmospheric muon source may be investigated.

7 Appendix

The appendix includes commented source code for the simulation. This code can also be found on GitHub as mentioned in 2.3.

7.1 main()

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoImaging.cc
/// \brief Main program of the Cargo example

#include "CargoDetectorConstruction.hh"
#include "CargoActionInitialization.hh"
```

```
#ifdef G4MULTITHREADED
#include "G4MTRunManager.hh"
#else
#include "G4RunManager.hh"
#endif

#include "G4UImanager.hh"
#include "FTFP_BERT_HP.hh"
#include "G4ScoringManager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"

#include "Randomize.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

int main(int argc, char **argv)
{
  // Detect interactive mode (if no arguments) and define UI session
  //
  G4UIExecutive *ui = 0;
  if (argc == 1)
  {
    ui = new G4UIExecutive(argc, argv);
  }

  // Choose the Random engine
  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager
  //
#ifdef G4MULTITHREADED
  G4MTRunManager *runManager = new G4MTRunManager;
#else
  G4RunManager *runManager = new G4RunManager;
#endif

  // Set mandatory initialization classes
  //
  // Detector construction
  runManager->SetUserInitialization(new CargoDetectorConstruction());

  // Physics list
  G4VModularPhysicsList *physicsList = new FTFP_BERT_HP;
  physicsList->SetVerboseLevel(1);
  runManager->SetUserInitialization(physicsList);

  // User action initialization
  runManager->SetUserInitialization(new CargoActionInitialization());

  G4ScoringManager* scManager = G4ScoringManager::GetScoringManager();

  // Initialize visualization
  //
  G4VisManager *visManager = new G4VisExecutive;
  // G4VisExecutive can take a verbosity argument - see /vis/verbose guidance.
  // G4VisManager* visManager = new G4VisExecutive("Quiet");
  visManager->Initialize();

  // Get the pointer to the User Interface manager
  G4UImanager *UImanager = G4UImanager::GetUIpointer();

  // Process macro or start UI session
  //
  if (!ui)
  {
    // batch mode
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command + fileName);
  }
  else
  {
    // interactive mode
    UImanager->ApplyCommand("/control/execute init_vis.mac");
    ui->SessionStart();
    delete ui;
  }

  // Job termination
  // Free the store: user actions, physics_list and detector_description are
  // owned and deleted by the run manager, so they should not be deleted
  // in the main() program !

  delete visManager;
  delete runManager;
}
```

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo.....
```

7.2 CargoActionInitialization

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoActionInitialization.hh
/// \brief Definition of the CargoActionInitialization class

#ifndef CargoActionInitialization_h
#define CargoActionInitialization_h 1

#include "G4VUserActionInitialization.hh"

/// Action initialization class.

class CargoActionInitialization : public G4VUserActionInitialization
{
  public:
    CargoActionInitialization();
    virtual ~CargoActionInitialization();

    virtual void BuildForMaster() const;
    virtual void Build() const;
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif
```

7.3 CargoPrimaryGeneratorAction

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoPrimaryGeneratorAction.hh
/// \brief Definition of the CargoPrimaryGeneratorAction class
```

```
#ifndef CargoPrimaryGeneratorAction_h
#define CargoPrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4ParticleGun.hh"
#include "globals.hh"
#include "G4GenericMessenger.hh"

class G4ParticleGun;
class G4Event;
class G4Box;

/// The primary generator action class with particle gun.
///
/// The default kinematic is a 6 MeV gamma, randomly distribued
/// in front of the phantom across 80% of the (X,Y) phantom size.

class CargoPrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
public:
  CargoPrimaryGeneratorAction();
  virtual ~CargoPrimaryGeneratorAction();

  // method from the base class
  virtual void GeneratePrimaries(G4Event *);
  void gGun();
  void nGun();
  void mGun();

  // method to access particle gun
  const G4ParticleGun *GetParticleGun() const { return fParticleGun; }

private:
  G4ParticleGun *fParticleGun;
  G4double field_x, field_y, field_z;
  G4int gun_num;

  G4GenericMessenger *fMessenger;
  void DefineCommands();
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif
```

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoPrimaryGeneratorAction.cc
/// \brief Implementation of the CargoPrimaryGeneratorAction class

#include "CargoPrimaryGeneratorAction.hh"

#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4Box.hh"
#include "G4RunManager.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4SystemOfUnits.hh"
#include "Randomize.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoPrimaryGeneratorAction::CargoPrimaryGeneratorAction()
  : G4VUserPrimaryGeneratorAction(),
    fParticleGun(0),
    field_x(10. * m),
    field_y(3. * m),
    field_z(3. * m),
    gun_num(1),
    fMessenger(0)
{
  DefineCommands();

  G4int n_particle = 1;
  fParticleGun = new G4ParticleGun(n_particle);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoPrimaryGeneratorAction::~CargoPrimaryGeneratorAction()
{
  delete fParticleGun;
  delete fMessenger;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoPrimaryGeneratorAction::GeneratePrimaries(G4Event *anEvent)
{
  switch (gun_num)
  {
  case 1:
    gGun();
    break;
  case 2:
    nGun();
    break;
  case 3:
    mGun();
    break;
  default:
    gGun();
    break;
  }

  fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoPrimaryGeneratorAction::DefineCommands()
{
  fMessenger = new G4GenericMessenger(this,
                    "/Cargo/gun/",
                    "Primary Generator Control");

  G4GenericMessenger::Command &chooseGun
    = fMessenger->DeclareProperty("chooseGun",
                    gun_num,
                "Choose Gun: 1: Gamma, 2: Neutron, 3: Muon");

  chooseGun.SetParameterName("num", true);
  chooseGun.SetRange("num>=1 && num <=3");
  chooseGun.SetDefaultValue("1");
}

void CargoPrimaryGeneratorAction::gGun()
{
  G4ParticleTable *particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition *particle = particleTable->FindParticle("gamma");

  fParticleGun->SetParticleDefinition(particle);
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0., 0., -1.));
  fParticleGun->SetParticleEnergy(5. * MeV);

  G4double particle_x = field_x * G4UniformRand() - 0.5 * field_x,
       particle_y = field_y * G4UniformRand() - 0.5 * field_y,
       particle_z = field_z;

  G4ThreeVector particle_pos = G4ThreeVector(particle_x, particle_y, particle_z);
  fParticleGun->SetParticlePosition(particle_pos);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoPrimaryGeneratorAction::nGun()
{
  G4ParticleTable *particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition *particle = particleTable->FindParticle("neutron");

  fParticleGun->SetParticleDefinition(particle);
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0., 0., -1.));
  fParticleGun->SetParticleEnergy(15. * MeV);

  G4double particle_x = field_x * G4UniformRand() - 0.5 * field_x,
      particle_y = field_y * G4UniformRand() - 0.5 * field_y,
      particle_z = field_z;

  G4ThreeVector particle_pos
    = G4ThreeVector(particle_x, particle_y, particle_z);
  fParticleGun->SetParticlePosition(particle_pos);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoPrimaryGeneratorAction::mGun()
{
  G4ParticleTable *particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  G4ParticleDefinition *particle
    = particleTable->FindParticle(particleName = "mu-");

  fParticleGun->SetParticleDefinition(particle);
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0., -1, 0.));
  fParticleGun->SetParticleEnergy(4. * GeV);

  G4double particle_x = field_x * G4UniformRand() - 0.5 * field_x,
      particle_y = field_y,
      particle_z = field_z * G4UniformRand() - 0.5 * field_z;

  G4ThreeVector particle_pos
    = G4ThreeVector(particle_x, particle_y, particle_z);
  fParticleGun->SetParticlePosition(particle_pos);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

7.4 CargoDetectorConstruction

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoDetectorConstruction.hh
/// \brief Definition of the CargoDetectorConstruction class

#ifndef CargoDetectorConstruction_h
#define CargoDetectorConstruction_h 1

#include "G4VUserDetectorConstruction.hh"
#include "globals.hh"
#include "G4GenericMessenger.hh"

class G4VPhysicalVolume;
class G4LogicalVolume;

/// Detector construction class to define materials and geometry.

class CargoDetectorConstruction : public G4VUserDetectorConstruction
{
public:
  CargoDetectorConstruction();
  virtual ~CargoDetectorConstruction();
  virtual void ConstructSDandField();
  virtual G4VPhysicalVolume *Construct();

  void DefineCommands();
  void ConstructMaterials();
private:
  G4bool checkOverlaps;
  G4GenericMessenger *fMessenger;
  G4int gnDet_num;

  G4LogicalVolume *flogicalMuonDetBot;
  G4LogicalVolume *flogicalMuonDetTop;
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif

//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoDetectorConstruction.cc
/// \brief Implementation of the CargoDetectorConstruction class

#include "CargoDetectorConstruction.hh"
```

```
#include "G4RunManager.hh"
#include "G4NistManager.hh"
#include "G4Box.hh"
#include "G4Cons.hh"
#include "G4Orb.hh"
#include "G4Sphere.hh"
#include "G4Trd.hh"
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4AssemblyVolume.hh"
#include "G4SystemOfUnits.hh"
#include "G4SDManager.hh"
#include "G4VSensitiveDetector.hh"
#include "G4GenericMessenger.hh"
#include "MuonDetectorSD.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoDetectorConstruction::CargoDetectorConstruction()
  : G4VUserDetectorConstruction(),
    checkOverlaps(false),
    fMessenger(0),
    gnDet_num(0),
    flogicalMuonDetBot(0),
    flogicalMuonDetTop(0)
{
  DefineCommands();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoDetectorConstruction::~CargoDetectorConstruction()
{
  delete fMessenger;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoDetectorConstruction::ConstructSDandField()
{
  G4SDManager *SDman = G4SDManager::GetSDMpointer();

  G4VSensitiveDetector *muonDetTop = new MuonDetectorSD("/muonDetTop");
  SDman->AddNewDetector(muonDetTop);
  flogicalMuonDetTop->SetSensitiveDetector(muonDetTop);

  G4VSensitiveDetector *muonDetBot = new MuonDetectorSD("/muonDetBot");
  SDman->AddNewDetector(muonDetBot);
  flogicalMuonDetBot->SetSensitiveDetector(muonDetBot);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

G4VPhysicalVolume *CargoDetectorConstruction::Construct()
{
  ConstructMaterials();

  G4Material *world_mat = G4Material::GetMaterial("G4_AIR");
  G4Material *container_mat = G4Material::GetMaterial("G4_Al");

  // material for gamma or neutron detection
  G4Material *gnDet_mat;
  switch (gnDet_num)
  {
  case 0:
    gnDet_mat = G4Material::GetMaterial("G4_CESIUM_IODIDE");
    break;
  case 1:
    gnDet_mat = G4Material::GetMaterial("G4_PLASTIC_SC_VINYLTOLUENE");
    break;
  default:
    gnDet_mat = G4Material::GetMaterial("G4_CESIUM_IODIDE");
    break;
  }

  G4Material *mDet_mat = G4Material::GetMaterial("G4_CESIUM_IODIDE");
  G4Material *gold_mat = G4Material::GetMaterial("G4_Au");
  G4Material *barrel_mat = G4Material::GetMaterial("wood");
  G4Material *barrelContent_mat = G4Material::GetMaterial("G4_WATER");
  G4Material *man_mat = G4Material::GetMaterial("G4_MUSCLE_SKELETAL_ICRP");
  G4Material *bomb_mat = G4Material::GetMaterial("G4_U");
  G4Material *lead_mat = G4Material::GetMaterial("G4_Pb");

  const G4double world_hx = 10. * m,
                 world_hy = 3. * m,
                 world_hz = 3. * m;
```

```
G4Box *solidWorld = new G4Box("World", world_hx, world_hy, world_hz);

G4LogicalVolume *logicalWorld = new G4LogicalVolume(solidWorld,
                          world_mat,
                          "World");

G4VPhysicalVolume *physWorld = new G4PVPlacement(0,
                         G4ThreeVector(),
                         logicalWorld,
                         "World",
                         0,
                         false,
                         0,
                         checkOverlaps);

G4double container_hx = 5. * m,
     container_hy = 1.5 * m,
     container_hz = 1.5 * m;

G4Box *solidContainer = new G4Box("solidContainer",
                  container_hx,
                  container_hy,
                  container_hz);

G4LogicalVolume *logicalContainer = new G4LogicalVolume(solidContainer,
                            container_mat,
                            "logicalContainer");

new G4PVPlacement(0,
          G4ThreeVector(),
          logicalContainer,
          "Container",
          logicalWorld,
          false,
          0,
          checkOverlaps);

const G4double air_hx = container_hx - 5. * mm,
         air_hy = container_hy - 5. * mm,
         air_hz = container_hz - 5. * mm;

G4Box *solidAir = new G4Box("solidAir", air_hx, air_hy, air_hz);

G4LogicalVolume *logicalAir = new G4LogicalVolume(solidAir,
                          world_mat,
                          "logicalAir");

new G4PVPlacement(0,
          G4ThreeVector(),
          logicalAir,
          "AirInContainer",
          logicalContainer,
          false,
          0,
          checkOverlaps);

const G4double gammaNeutron_hx = 5.0 * m,
         gammaNeutron_hy = 1.5 * m,
         gammaNeutron_hz = 5.0 * cm;

G4Box *solidGammaNeutronDet = new G4Box("solidGammaNeutronDet",
                    gammaNeutron_hx,
                    gammaNeutron_hy,
                    gammaNeutron_hz);

G4LogicalVolume *logicalGammaNeutronDet
  = new G4LogicalVolume(solidGammaNeutronDet,
              lead_mat,
                "logicalGammaNeutronDet");

G4ThreeVector gammaNeutronDetPos
  = G4ThreeVector(0, 0, -1 * (container_hz + 10.0 * cm));

new G4PVPlacement(0,
          gammaNeutronDetPos,
          logicalGammaNeutronDet,
          "GammaNeutronDetector",
          logicalWorld,
          false,
          0,
          checkOverlaps);

const G4double muon_hx = 5.0 * m,
         muon_hy = 5.0 * cm,
         muon_hz = 1.5 * m;

G4Box *solidMuonDet = new G4Box("solidMuonDet",
                muon_hx,
                muon_hy,
                muon_hz);

flogicalMuonDetBot = new G4LogicalVolume(solidMuonDet,
                     mDet_mat,
                     "logicalMuonDetBot");
flogicalMuonDetTop = new G4LogicalVolume(solidMuonDet,
                     mDet_mat,
                     "logicalMuonDetTot");

G4ThreeVector muonDetPos1
  = G4ThreeVector(0, -1 * (container_hy + 10.0 * cm), 0);

new G4PVPlacement(0,
          muonDetPos1,
          flogicalMuonDetBot,
          "MuonDetectorBot",
          logicalWorld,
          false,
          0,
          checkOverlaps);

G4ThreeVector muonDetPos2 = G4ThreeVector(0, (container_hy + 10.0 * cm), 0);

new G4PVPlacement(0,
          muonDetPos2,
          flogicalMuonDetTop,
          "MuonDetectorTop",
          logicalWorld,
          false,
          1,
          checkOverlaps);

G4Trd *solidGold = new G4Trd("solidGold",
               125.0 * mm,
               100.0 * mm,
               225.0 * mm,
               200.0 * mm,
               50.0 * mm);

G4LogicalVolume *logicalGold = new G4LogicalVolume(solidGold,
                           gold_mat,
                           "logicalGold");
```

```
G4RotationMatrix *gold_rot = new G4RotationMatrix();
gold_rot->rotateX(90 * deg);

G4int gold_count = 0;
for (int k = 0; k < 5; k += 1)
{
  for (int j = 0; j < 15; j += 1)
  {
    for (int i = 0; i < 8; i += 1)
    {
      G4double gold_xpos = -4.5 * m + i * (250 * mm),
           gold_ypos = -1.5 * m + 55.0 * mm + j * (110 * mm),
           gold_zpos = 1.0 * m - k * (500 * mm);

      new G4PVPlacement(gold_rot,
        G4ThreeVector(gold_xpos, gold_ypos, gold_zpos),
                logicalGold,
                "gold",
                logicalAir,
                false,
                gold_count,
                checkOverlaps);

      gold_count += 1;
    }
  }
}

G4double barrel_outer_radius = 0.5 * m,
     barrel_half_length = 0.5 * m;

G4Tubs *solidBarrel = new G4Tubs("solidBarrel",
                 0.,
                 barrel_outer_radius,
                 barrel_half_length,
                 0 * deg,
                 360 * deg);

G4LogicalVolume *logicalBarrel = new G4LogicalVolume(solidBarrel,
                           barrel_mat,
                           "logicalBarrel");

G4Tubs *solidBarrelContent = new G4Tubs("solidBarrelContent",
                    barrel_outer_radius - 5 * cm,
                    0.5 * m,
                    barrel_half_length - 2.5 * cm,
                    0 * deg,
                    360 * deg);

G4LogicalVolume *logicalBarrelContent
  = new G4LogicalVolume(solidBarrelContent,
              barrelContent_mat,
              "logicalBarrelContent");

new G4PVPlacement(0,
          G4ThreeVector(0, 0, 2.5 * cm),
          logicalBarrelContent,
          "physicalBarrelContent",
          logicalBarrel,
          0,
          checkOverlaps);

G4RotationMatrix *barrrel_rot = new G4RotationMatrix();
barrrel_rot->rotateX(90 * deg);
new G4PVPlacement(barrrel_rot,
          G4ThreeVector(-1.5 * m, -1 * m, 0.7 * m),
          logicalBarrel,
          "physicalBarrel1",
          logicalAir,
          0,
          checkOverlaps);
new G4PVPlacement(barrrel_rot,
          G4ThreeVector(-0.7 * m, -1 * m, -0.7 * m),
          logicalBarrel,
          "physicalBarrel2",
          logicalAir,
          0,
          checkOverlaps);

G4double manAir_hx = 0.4 * m,
     manAir_hy = 1. * m,
     manAir_hz = 0.4 * m;

G4Box *solidManAir
  = new G4Box("solidManAir", manAir_hx, manAir_hy, manAir_hz);

G4LogicalVolume *logicalManAir = new G4LogicalVolume(solidManAir,
                           world_mat,
                          "logicalManAir");

new G4PVPlacement(0,
          G4ThreeVector(1 * m, -0.5 * m, 1. * m),
          logicalManAir,
          "physicalManAir",
          logicalAir,
          0,
          checkOverlaps);

G4RotationMatrix *man_rot = new G4RotationMatrix();
man_rot->rotateY(30 * deg);
new G4PVPlacement(man_rot,
          G4ThreeVector(4 * m, -0.5 * m, 0. * m),
          logicalManAir,
          "physicalManAir",
          logicalAir,
          0,
          checkOverlaps);

G4RotationMatrix *man_rot2 = new G4RotationMatrix();
man_rot2->rotateY(60 * deg);
new G4PVPlacement(man_rot2,
          G4ThreeVector(3 * m, -0.5 * m, -1. * m),
          logicalManAir,
          "physicalManAir",
          logicalAir,
          0,
          checkOverlaps);

G4Box *solidManLeg = new G4Box("solidManLeg", 0.1 * m, 0.5 * m, 0.1 * m);

G4LogicalVolume *logicalManLeg = new G4LogicalVolume(solidManLeg,
                          man_mat,
                          "logicalManLeg");

new G4PVPlacement(0, G4ThreeVector(-0.14 * m, -0.5 * m, 0. * m),
          logicalManLeg, "physicalManLegLeft", logicalManAir,
          false, 0, checkOverlaps);
new G4PVPlacement(0, G4ThreeVector(0.14 * m, -0.5 * m, 0. * m),
          logicalManLeg, "physicalManLegRight", logicalManAir,
          false, 0, checkOverlaps);

G4Box *solidManBody
```

```
  = new G4Box("solidManBody", 0.25 * m, 0.27 * m, 0.25 * m);

G4LogicalVolume *logicalManBody = new G4LogicalVolume(solidManBody,
                            man_mat,
                            "logicalManBody");

new G4PVPlacement(0, G4ThreeVector(0, 0.27 * m, 0),
          logicalManBody, "physicalManBody", logicalManAir,
          false, 0, checkOverlaps);

G4Orb *solidManHead = new G4Orb("solidManHead", 0.2 * m);

G4LogicalVolume *logicalManHead = new G4LogicalVolume(solidManHead,
                            man_mat,
                            "logicalManHead");

new G4PVPlacement(0, G4ThreeVector(0, 0.74 * m, 0),
          logicalManHead, "physicalManHead", logicalManAir,
          false, 0, checkOverlaps);

G4Box *solidManArm = new G4Box("solidManArm", 0.07 * m, 0.3 * m, 0.1 * m);

G4LogicalVolume *logicalManArm = new G4LogicalVolume(solidManArm,
                           man_mat,
                           "logicalManArm");

new G4PVPlacement(0, G4ThreeVector(-0.32 * m, 0.22 * m, 0),
          logicalManArm, "physicalManArm1", logicalManAir,
          false, 0, checkOverlaps);

new G4PVPlacement(0, G4ThreeVector(0.32 * m, 0.22 * m, 0),
          logicalManArm, "physicalManArm2", logicalManAir,
          false, 0, checkOverlaps);

G4double shield_hx = 40. * cm,
     shield_hy = 40. * cm,
     shield_hz = 40. * cm;

G4Box *solidShield
   = new G4Box("solidShield", shield_hx, shield_hy, shield_hz);

G4LogicalVolume *logicalShield = new G4LogicalVolume(solidShield,
                           lead_mat,
                           "logicalShield");
  new G4PVPlacement(0,
            G4ThreeVector(2.0*m, -(air_hy - shield_hy), 0.*m),
            logicalShield, "physicalShield", logicalAir,
            false, 0, checkOverlaps);

  G4Box *solidShieldAir
    = new G4Box("solidShieldAir",
          shield_hx - 0.5*cm,
          shield_hz - 0.5*cm,
          shield_hz - 0.5*cm);

  G4LogicalVolume *logicalShieldAir
    = new G4LogicalVolume(solidShieldAir,
                world_mat,
                "solidShieldAir");

  new G4PVPlacement(0,
            G4ThreeVector(),
            logicalShieldAir, "physicalShieldAir", logicalShield,
            false, 0, checkOverlaps);

  G4Orb *solidBomb = new G4Orb("solidBomb", shield_hx - 2.*cm);

  G4LogicalVolume *logicalBomb
    = new G4LogicalVolume(solidBomb,
                bomb_mat,
                "logicalBomb");

  new G4PVPlacement(0,
            G4ThreeVector(),
            logicalBomb, "physicalBomb", logicalShieldAir,
            false, 0, checkOverlaps);

  return physWorld;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoDetectorConstruction::ConstructMaterials()
{
  G4String name, symbol;
  G4double density;
  G4int ncomp, z;
  G4NistManager *nistMan = G4NistManager::Instance();

  density = 1.01 * g / mole;
  G4Element *elH
    = new G4Element(name = "Hydrogen", symbol = "H", z = 1., density);
  density = 12.01 * g / mole;
  G4Element *elC
    = new G4Element(name = "Carbon", symbol = "C", z = 6., density);
  density = 16.00 * g / mole;
  G4Element *elO
    = new G4Element(name = "Oxygen", symbol = "O", z = 8., density);

  // Container Material
  nistMan->FindOrBuildMaterial("G4_AIR");
  nistMan->FindOrBuildMaterial("G4_Al");

  nistMan->FindOrBuildMaterial("G4_WATER");
  nistMan->FindOrBuildMaterial("G4_Au");
  nistMan->FindOrBuildMaterial("G4_MUSCLE_SKELETAL_ICRP");
  nistMan->FindOrBuildMaterial("G4_Pb");
  nistMan->FindOrBuildMaterial("G4_U");

  G4Material *wood
    = new G4Material(name = "wood", density = 0.9 * g / cm3, ncomp = 3);
  wood->AddElement(elH, 4);
  wood->AddElement(elO, 1);
  wood->AddElement(elC, 2);

  // Detector Material
  nistMan->FindOrBuildMaterial("G4_CESIUM_IODIDE");
  nistMan->FindOrBuildMaterial("G4_PLASTIC_SC_VINYLTOLUENE");

  G4cout << G4endl << "The materials defined are : " << G4endl << G4endl;
  G4cout << *(G4Material::GetMaterialTable()) << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoDetectorConstruction::DefineCommands()
{
  fMessenger
    = new G4GenericMessenger(this,
                 "/Cargo/Gamma_Neutron/",
                  "Gamma/Neutron Detector Material Control");

  G4GenericMessenger::Command &chooseGun
    = fMessenger->DeclareProperty("chooseMat",
                    gnDet_num,
              "Choose particle for detection: 0: Gamma, 1: Neutron");

  chooseGun.SetParameterName("num", true);
  chooseGun.SetRange("num>=0 && num <=1");
  chooseGun.SetDefaultValue("0");
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

7.5 MuonDetectorSD

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file MuonDetectorSD.hh
/// \brief Definition of the MuonDetectorSD class
```

```
#ifndef MuonDetectorSD_h
#define MuonDetectorSD_h 1

#include "G4VSensitiveDetector.hh"

#include "MuonDetectorHit.hh"

class G4Step;
class G4HCofThisEvent;
class G4TouchableHistory;

class MuonDetectorSD : public G4VSensitiveDetector
{
public:
  MuonDetectorSD(G4String name);
  virtual ~MuonDetectorSD();

  virtual void Initialize(G4HCofThisEvent *HCE);
  virtual G4bool ProcessHits(G4Step *aStep, G4TouchableHistory *ROhist);
  virtual void   EndOfEvent(G4HCofThisEvent* hitCollection);

private:
  MuonDetectorHitsCollection *fHitsCollection;
  G4int fHCID;
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif

//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file MuonDetectorSD.cc
/// \brief Implementation of the MuonDetector class

#include "MuonDetectorSD.hh"
#include "MuonDetectorHit.hh"
#include "G4HCofThisEvent.hh"
#include "G4TouchableHistory.hh"
#include "G4Track.hh"
#include "G4Step.hh"
#include "G4SDManager.hh"
#include "G4ios.hh"
#include "G4MuonMinus.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

MuonDetectorSD::MuonDetectorSD(G4String name)
    : G4VSensitiveDetector(name), fHitsCollection(0), fHCID(-1)
{
  collectionName.insert("MuonDetectorColl");
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

MuonDetectorSD::~MuonDetectorSD()
{
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void MuonDetectorSD::Initialize(G4HCofThisEvent *hce)
{
  fHitsCollection
```

```
    = new MuonDetectorHitsCollection(SensitiveDetectorName, collectionName[0]);

  if (fHCID < 0)
  {
    fHCID = G4SDManager::GetSDMpointer()->GetCollectionID(fHitsCollection);
  }
  hce->AddHitsCollection(fHCID, fHitsCollection);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

G4bool MuonDetectorSD::ProcessHits(G4Step *step, G4TouchableHistory *)
{
  if (G4MuonMinus::MuonMinusDefinition() == step->GetTrack()->GetDefinition())
  {
    G4String volume_name
      = step->GetPreStepPoint()->GetPhysicalVolume()->GetName();

    G4StepPoint *preStepPoint = step->GetPreStepPoint();
    G4TouchableHistory *touchable
      = (G4TouchableHistory *)(step->GetPreStepPoint()->GetTouchable());

    G4VPhysicalVolume *motherPhysical = touchable->GetVolume(1);
    G4int copyNo = motherPhysical->GetCopyNo();

    MuonDetectorHit *hit = new MuonDetectorHit(copyNo);
    G4ThreeVector worldPos, localPos;

    if (preStepPoint->GetStepStatus() == fGeomBoundary)
    {
      worldPos = preStepPoint->GetPosition();

      localPos = touchable->GetHistory()
                     ->GetTopTransform()
                     .TransformPoint(worldPos);
      hit->SetWorldPos(worldPos);
      hit->SetLocalPos(localPos);
      fHitsCollection->insert(hit);
    }
  }

  return true;
}
```

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void MuonDetectorSD::EndOfEvent(G4HCofThisEvent *)
{
  if (verboseLevel > 1)
  {
    G4int nofHits = fHitsCollection->entries();
    G4cout << G4endl
           << "Hits Collection ID[" << fHCID
           << "]: in this event there are " << nofHits
           << " hits in the tracker chambers: " << G4endl;
    for (G4int i = 0; i < nofHits; i++)
      (*fHitsCollection)[i]->Print();
  }
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

7.6 MuonDetectorHit

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file MuonDetectorHit.hh
/// \brief Definition of the MuonDetectorHit class

#ifndef MuonDetectorHit_h
#define MuonDetectorHit_h 1

#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
#include "G4LogicalVolume.hh"
#include "G4Transform3D.hh"
#include "G4RotationMatrix.hh"

class G4AttDef;
class G4AttValue;

/// Drift chamber hit
///
/// It records:
/// - the layer ID
/// - the particle time
/// - the particle local and global positions

class MuonDetectorHit : public G4VHit
{
public:
  MuonDetectorHit();
  MuonDetectorHit(G4int i);
  MuonDetectorHit(const MuonDetectorHit &right);
  virtual ~MuonDetectorHit();

  const MuonDetectorHit &operator=(const MuonDetectorHit &right);
  int operator==(const MuonDetectorHit &right) const;

  inline void *operator new(size_t);
  inline void operator delete(void *aHit);

  virtual void Draw();
  virtual const std::map<G4String, G4AttDef> *GetAttDefs() const;
  virtual std::vector<G4AttValue> *CreateAttValues() const;
  virtual void Print();
  void SetLocalPos(G4ThreeVector xyz) { fLocalPos = xyz; }
  G4ThreeVector GetLocalPos() const { return fLocalPos; }

  void SetWorldPos(G4ThreeVector xyz) { fWorldPos = xyz; }
  G4ThreeVector GetWorldPos() const { return fWorldPos; }

private:
  G4ThreeVector fLocalPos;
  G4ThreeVector fWorldPos;
  G4int copyno;
};

typedef G4THitsCollection<MuonDetectorHit> MuonDetectorHitsCollection;

extern G4ThreadLocal G4Allocator<MuonDetectorHit> *MuonDetectorHitAllocator;

inline void *MuonDetectorHit::operator new(size_t)
{
  if (!MuonDetectorHitAllocator)
    MuonDetectorHitAllocator = new G4Allocator<MuonDetectorHit>;
  return (void *)MuonDetectorHitAllocator->MallocSingle();
}

inline void MuonDetectorHit::operator delete(void *aHit)
{
  MuonDetectorHitAllocator->FreeSingle((MuonDetectorHit *)aHit);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif

//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file MuonDetectorHit.cc
/// \brief Implementation of the MuonDetectorHit class

#include "MuonDetectorHit.hh"

#include "G4VVisManager.hh"
#include "G4VisAttributes.hh"
#include "G4Circle.hh"
#include "G4Colour.hh"
#include "G4AttDefStore.hh"
#include "G4AttDef.hh"
#include "G4AttValue.hh"
#include "G4UIcommand.hh"
#include "G4UnitsTable.hh"
#include "G4SystemOfUnits.hh"
#include "G4ios.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

G4ThreadLocal G4Allocator<MuonDetectorHit> *MuonDetectorHitAllocator = 0;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

MuonDetectorHit::MuonDetectorHit()
  : G4VHit(),
    fLocalPos(0),
    fWorldPos(0)
{
}
```

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

MuonDetectorHit::~MuonDetectorHit()
{
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

MuonDetectorHit::MuonDetectorHit(const MuonDetectorHit &right)
  : G4VHit()
{
  fWorldPos = right.fWorldPos;
  fLocalPos = right.fLocalPos;
}

MuonDetectorHit::MuonDetectorHit(G4int i)
  : G4VHit(),
    fLocalPos(0),
    fWorldPos(0),
    copyno(i)
{
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

const MuonDetectorHit &MuonDetectorHit::operator=(const MuonDetectorHit &right)
{
  fWorldPos = right.fWorldPos;
  fLocalPos = right.fLocalPos;
  return *this;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

int MuonDetectorHit::operator==(const MuonDetectorHit & /*right*/) const
{
  return 0;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void MuonDetectorHit::Draw()
{
  G4VVisManager *pVVisManager = G4VVisManager::GetConcreteInstance();
  if (pVVisManager)
  {
    G4Circle circle(fWorldPos);
    circle.SetScreenSize(2);
    circle.SetFillStyle(G4Circle::filled);
    G4Colour colour(1., 1., 0.);
    G4VisAttributes attribs(colour);
    circle.SetVisAttributes(attribs);
    pVVisManager->Draw(circle);
  }
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

const std::map<G4String, G4AttDef> *MuonDetectorHit::GetAttDefs() const
{
  G4bool isNew;
  std::map<G4String, G4AttDef> *store
    = G4AttDefStore::GetInstance("MuonDetectorHit", isNew);

  if (isNew)
  {
    (*store)["HitType"]
      = G4AttDef("HitType", "Hit Type", "Physics", "", "G4String");

    (*store)["ID"] = G4AttDef("ID", "ID", "Physics", "", "G4int");

    (*store)["Pos"]
    = G4AttDef("Pos", "Position", "Physics", "G4BestUnit", "G4ThreeVector");
  }
  return store;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

std::vector<G4AttValue> *MuonDetectorHit::CreateAttValues() const
{
  std::vector<G4AttValue> *values = new std::vector<G4AttValue>;

  values
    ->push_back(G4AttValue("HitType", "MuonDetectorHit", ""));
  values
    ->push_back(G4AttValue("Pos", G4BestUnit(fWorldPos, "Length"), ""));
  return values;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void MuonDetectorHit::Print()
{
  G4cout << "World (x,z) " << fWorldPos.x()/cm << "[cm]" << ", "
  << fWorldPos.z()/cm << "[cm]" << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

7.7 CargoRunAction

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoRunAction.hh
/// \brief Definition of the CargoRunAction class
```

```
#ifndef CargoRunAction_h
#define CargoRunAction_h 1

#include "G4UserRunAction.hh"
#include "globals.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

class G4Run;

/// Run action class

class CargoRunAction : public G4UserRunAction
{
public:
  CargoRunAction();
  virtual ~CargoRunAction();

  virtual void BeginOfRunAction(const G4Run *run);
  virtual void EndOfRunAction(const G4Run *run);
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoRunAction.cc
/// \brief Implementation of the CargoRunAction class

#include "CargoRunAction.hh"
#include "G4RunManager.hh"
#include "CargoAnalysis.hh"
#include "G4SystemOfUnits.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoRunAction::CargoRunAction()
    : G4UserRunAction()
{
  G4AnalysisManager *analysisManager = G4AnalysisManager::Instance();
  G4cout << "Using " << analysisManager->GetType() << G4endl;
  analysisManager->SetVerboseLevel(1);
  analysisManager->SetFileName("Cargo");
  // Creating 2D histograms
  analysisManager
      ->CreateH2("ScatterHist", "Scatter Angle",
                 1000, -500. * cm, 500. * cm, 300, -150. * cm, 150. * cm);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoRunAction::~CargoRunAction()
{
  delete G4AnalysisManager::Instance();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoRunAction::BeginOfRunAction(const G4Run *)
{
  auto analysisManager = G4AnalysisManager::Instance();
  analysisManager->OpenFile();
}
```

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoRunAction::EndOfRunAction(const G4Run *)
{
  auto analysisManager = G4AnalysisManager::Instance();
  analysisManager->Write();
  analysisManager->CloseFile();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

7.8 CargoEventAction

```
//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoEventAction.hh
/// \brief Definition of the CargoEventAction class

#ifndef CargoEventAction_h
```

```
#define CargoEventAction_h 1

#include "G4UserEventAction.hh"
#include "globals.hh"

class CargoEventAction : public G4UserEventAction
{
public:
  CargoEventAction();
  virtual ~CargoEventAction();

  virtual void BeginOfEventAction(const G4Event *);
  virtual void EndOfEventAction(const G4Event *);

private:
  G4int HCIDTop;
  G4int HCIDBot;
};

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#endif

//
// ********************************************************************
// * 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.          *
// ********************************************************************
//
/// \file CargoEventAction.cc
/// \brief Implementation of the CargoEventAction class

#include "CargoEventAction.hh"
#include "MuonDetectorHit.hh"
#include "CargoAnalysis.hh"
#include "G4Event.hh"
#include "G4RunManager.hh"
#include "G4EventManager.hh"
#include "G4HCofThisEvent.hh"
#include "G4VHitsCollection.hh"
#include "G4SDManager.hh"
#include "G4SystemOfUnits.hh"
#include "G4ios.hh"

namespace
{
// Utility function which finds a hit collection with the given Id
// and print warnings if not found
G4VHitsCollection *GetHC(const G4Event *event, G4int collId)
{
  auto hce = event->GetHCofThisEvent();
  if (!hce)
  {
    G4ExceptionDescription msg;
    msg << "No hits collection of this event found." << G4endl;
    G4Exception("CargoEventAction::EndOfEventAction()",
          "CargoCode001", JustWarning, msg);
    return nullptr;
  }

  auto hc = hce->GetHC(collId);
  if (!hc)
  {
    G4ExceptionDescription msg;
    msg << "Hits collection " << collId << " of this event not found."
      << G4endl;
    G4Exception("CargoEventAction::EndOfEventAction()",
          "CargoCode001", JustWarning, msg);
  }
  return hc;
}

} // namespace

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoEventAction::CargoEventAction()
  : G4UserEventAction(),
    HCIDTop(-1),
    HCIDBot(-1)
{
  G4RunManager::GetRunManager()->SetPrintProgress(1);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

CargoEventAction::~CargoEventAction()
{
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoEventAction::BeginOfEventAction(const G4Event *)
{
  // Find hit collections and histogram Ids by names (just once)
  // and save them in the data members of this class
  if (HCIDTop == -1 || HCIDBot == -1)
  {
    G4SDManager *sdManager = G4SDManager::GetSDMpointer();
    HCIDTop = sdManager->GetCollectionID("muonDetTop/MuonDetectorColl");
    HCIDBot = sdManager->GetCollectionID("muonDetBot/MuonDetectorColl");
  }
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void CargoEventAction::EndOfEventAction(const G4Event *event)
{
  // Get analysis manager
  G4AnalysisManager *analysisManager = G4AnalysisManager::Instance();

  G4HCofThisEvent *hce = event->GetHCofThisEvent();
  if (!hce)
```

```
{
  G4ExceptionDescription msg;
  msg << "No hits collection of this event found.\n";
  G4Exception("Run::RecordEvent()",
        "Code001", JustWarning, msg);
  return;
}

const MuonDetectorHitsCollection *HCTop =
  static_cast<const MuonDetectorHitsCollection *>(hce->GetHC(HCIDTop));

const MuonDetectorHitsCollection *HCBot =
  static_cast<const MuonDetectorHitsCollection *>(hce->GetHC(HCIDBot));

if (!HCBot || !HCTop)
{

  G4ExceptionDescription msg;
  msg << "Some of hits collections of this event not found.\n";
  G4Exception("Run::RecordEvent()",
        "Code001", JustWarning, msg);
  return;
}

G4int top_hits = HCTop->entries();
G4int bot_hits = HCBot->entries();

if (top_hits > 0 && bot_hits > 0)
{
  MuonDetectorHit *firstHit = (*HCTop)[0];
  MuonDetectorHit *secondHit = (*HCBot)[0];

  G4ThreeVector worldPos_1 = firstHit->GetWorldPos();
  G4ThreeVector worldPos_2 = secondHit->GetWorldPos();

  G4ThreeVector ref_vec = G4ThreeVector(0,
                      worldPos_2.getY(),
                      0);

  G4ThreeVector dis_vec
    = G4ThreeVector(worldPos_2.getX() - worldPos_1.getX(),
          worldPos_2.getY() - worldPos_1.getY(),
          worldPos_2.getZ() - worldPos_1.getZ());
    G4double weight
      = acos(ref_vec.dot(dis_vec) / (ref_vec.mag() * dis_vec.mag()))
        * (180. / M_PI);

    G4int hist_id = analysisManager->GetFirstH2Id();

    analysisManager
      ->FillH2(hist_id, worldPos_1.getX(), worldPos_1.getZ(), weight);
  }
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

7.9 plot.py

```
"""
This script takes the generated .csv 2D histogram files from the
gamma/neutron simulations and plots a color map
"""

import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import csv
from mpl_toolkits.mplot3d import Axes3D

mpl.rcParams["font.family"] = "FreeSerif"
plt.rc("text", usetex=True)

x = []
y = []
z =[]

with open("gammaMeshEdepNeutron.csv", "r") as f:
    reader = csv.reader(f, delimiter=',')
    count = 0
    for row in reader:
        if (count > 2):
            x.append(row[0])
            y.append(row[1])
            z.append(float(row[3]))
        count+= 1
```

```
x_array = np.array(x)
y_array = np.array(y)
z_array = np.array(z)

# 75 is related to bin settings in the scoring mesh
Z = np.reshape(z, (-1, 75))
Z = np.rot90(Z)

ax = plt.axes()
ax.imshow(Z, cmap = "hot")
ax.yaxis.set_major_locator(plt.NullLocator())
ax.xaxis.set_major_formatter(plt.NullFormatter())
plt.title("$\gamma$ Energy Deposition")
plt.show()
```

Bibliography

[1] *Atmospheric Muons*. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/Particles/muonatm.html. (accessed: March, 09, 2020).

[2] *Figure 1, Odette Cancer Centre*. URL: https://mapio.net/images-p/20502863.jpg. (accessed: March, 7, 2020).

[3] *Geant4 Tutorial*. URL: https://indico.lucas.lu.se/event/932/other-view?view=standard. (accessed: February, 11, 2020).