Final project on Geant4 course 2018

Katja Mankinen katja.mankinen@hep.lu.se

October 17, 2018

Contents

1 Introduction

The aim of this project was to set up a detector simulation in Geant4 and use different primary particles (neutrons, gammas, and muons) to detect various items inside an aluminium container.
I have re-used code from tutorials and examples B2, B4, B5.

2 Experimental setup

2.1 Geometry definition

Geometry is defined in `src/DetectorConstructor.cc`. The setup consists of an aluminium container filled with air, 3 detector planes and different items to be probed.

2.1.1 Detector material and position

There are in total four detector planes:

- material: CsI (for gamma, muon) and scintillator (neutron)
- size: 10 m long, 3 m high, 10 cm thick
- gamma and neutron detectors: located with a gap of 10 cm next to the container
- muon detectors: same gap but located on top and below the container

2.1.2 Item descriptions

In order to see the effect of both size and material of the items when shooting them with different particles, I placed the following items to the container using mostly `G4PVPlacement`:

- innocent nuclear weapon of pure uranium, simple `G4Orb`
- human made entirely of tissue (nist database: `G4_A-150_TISSUE`), constructed by combining `G4Orb` and `G4Tubs`
- tiny golden ring to test our detection resolution: `G4Tubs("ring", 1.6*cm, 1.8*cm, 1*cm, 0.0*deg, 360*deg);`
- three wooden boxes on top of each other to test parametrised volumes. A new class was created to parameterise the boxes: `woodBoxParameterisation()`
- two lead boxes, simply to see how well lead actually absorbs low-energy particles
- stainless steel trapezoid to test weird shapes: `G4Trd("steelTrd", 10.*cm, 20*cm, 40*cm, 10*cm, 60*cm);`

Materials were constructed using `G4NistManager`, for example
`G4Material* tissue = G4Material::GetMaterial("G4_A-150_TISSUE").`

- air = `G4Material::GetMaterial("G4_AIR");`
- scintillator = `G4Material::GetMaterial("G4_PLASTIC_SC_VINYLTOLUENE");`
- CsI = `G4Material::GetMaterial("G4_CESIUM_IODIDE");`
- stainless steel = `G4Material::GetMaterial("G4_STAINLESS-STEEL");`
- gold = `G4Material::GetMaterial("G4_Au");`
- wood = `G4Material::GetMaterial("G4_CELLULOSE_CELLOPHANE");`

The whole setup is visualized in Figure 1.

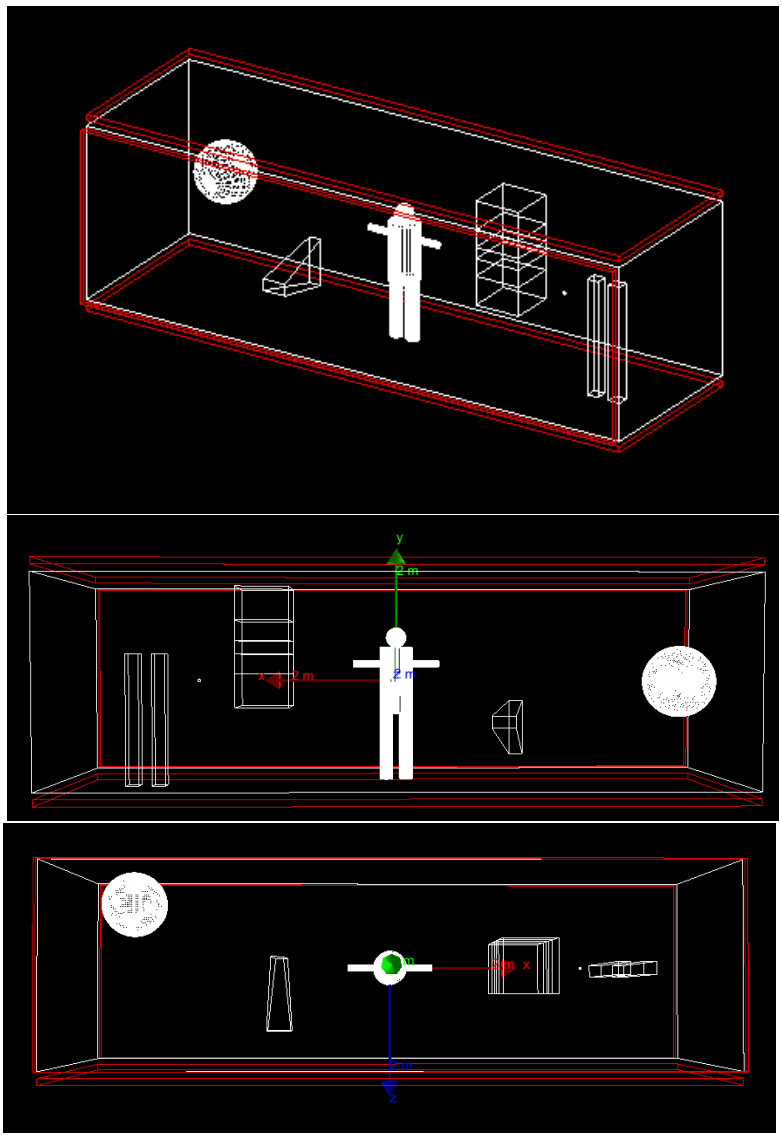

Figure 1: Different views of the detectors (red) and items (white) placed inside the container volume.

2.2 Physics list and action initialization

Only one physics list, `FTFP_BERT` was used as requested in the project instructions.
I created additional class ActionInitialization which is needed to instantiate the user action classes, in this case `PrimaryGeneratorAction`, `RunAction` and `EventAction`.

2.3 Primary generator

The primary generator action class uses the `G4ParticleGun`. The primary kinematics depends on the primary particle:

- neutron: 15 MeV, momentum to +z direction
- gamma: 5 MeV, momentum to +z direction
- muon: 4 GeV, momentum to -y direction

To have a particle shotgun, 10 particles at the same time with random positions along the 10×3 plane.

```
int nPrimaries = 10;

// uniform distribution for parallel beam in 10*m x 3*m:
// I want "particle shotgun": many primaries at the same time!

for(int i=0; i<nPrimaries; i++){
G4double x0 = 10*(0.5-G4UniformRand()); //uniform random number in [0,1]
G4double y0 = 3*(0.5-G4UniformRand());
G4double z0 = 0;

// z1 for muons: x0 same, y0 should not change
G4double z1 = 3*(0.5-G4UniformRand());

// n, gamma:
if (particle == fGamma || particle == fNeutron){
fParticleGun->SetParticlePosition(G4ThreeVector(x0*m,y0*m,-4.*m));
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));} //
if (particle == fMuon){
  fParticleGun->SetParticlePosition(G4ThreeVector(x0*m,4*m,z1*m));
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,-1.,0.));} // diff for
      muons

// particles have different energies depending on their type

G4double Ekin = 1*MeV;
if (particle == fGamma) Ekin = 5*MeV;
else if (particle == fNeutron) Ekin = 15*MeV;
else Ekin = 4*GeV; // for muons

fParticleGun->SetParticleEnergy(Ekin);
fParticleGun->GeneratePrimaryVertex(event);
}
```

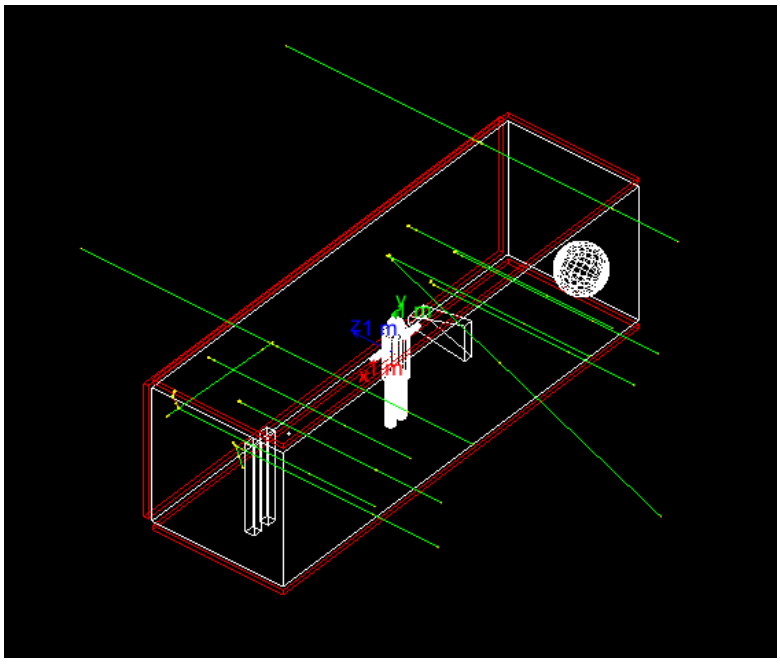

Figure 2: Example of running `/run/beamOn 1` which shoots 10 primaries, in this case 5 MeV gammas shown as green lines.

2.4 Run and events

Two additional source cores and corresponding headers were made: `RunAction` and `EventAction`.

In the `RunAction`, analysis ntuples are created by analysis manager, using ROOT. Ntuple stores only the location and time of the muon hit. In a similar manner, two histograms are created: one for each muon detector plane, storing the hit locations on the x-z plane. At the beginning of `RunAction`, analysis manager is instantiated and output files are opened. Correspondingly histograms and the ntuple are saved at `EndOfRunAction`.
`EventAction` is created to get the muon hit collection, to save hit locations into histograms and to calculate muon scattering angles and to save the location and angle information to an output text file. At the end, event diagnostics are also printed.

2.5 Detector response

I use a simple scoring macro to score energy deposits in gamma and neutron detectors. In addition, two sensitive muon detectors and a muon hit class are defined to be able to track particles: `MuonDetectorSD` and `MuonDetectorHit`, code in Appendix B. Each hit has a identifier and position information, and is created during tracking of a particle step by step. This hit is then inserted in a muon hits collection.

Example of a scoring macro `scoring.mac` to save energy deposits:

```
/run/initialize

# setting up visualization

/vis/open OGL 600x600-0+0
/vis/verbose errors
/vis/viewer/set/autoRefresh false

# define scoring mesh

/score/create/boxMesh boxMesh_1
/score/mesh/boxSize 5. 1.5 0.05 m
/score/mesh/translate/xyz 0 0 1.65 m
/score/mesh/nBin 100 30 30 # 200 60 60
/score/quantity/energyDeposit eDep MeV

/vis/disable
/run/verbose 2
/run/beamOn 2000
/vis/enable

#drawing projections
/score/drawProjection boxMesh_1 eDep

# Dump scores to a file

/score/dumpQuantityToFile boxMesh_1 eDep eDep.txt

/control/execute draw.mac
```

3 Results

For both gamma and neutron studies, 20000 primary particles (i.e. 2000 events) were shot. This number was chosen because more than 5000 events usually caused a segmentation fault, and even with smaller amount of primaries the energy deposits were visible.
Since muon studies were done in a batch mode, 100 000 muons (10 000 events with a particle shotgun) were shot.

3.1 Gamma

3.1.1 Energy deposits

Example of running `/run/beamOn 1` which shoots gamma rays is shown in Figure 2. For gammas, detector material is CsI. Energy deposits in the gamma detector plane are shown in Figure 3. We can easily see that uranium bomb (left-most sphere), cross-section of stainless steel trapezoid (second-left white spot), and both wood and lead boxes are visible - but the human in the middle is somewhat blurry.

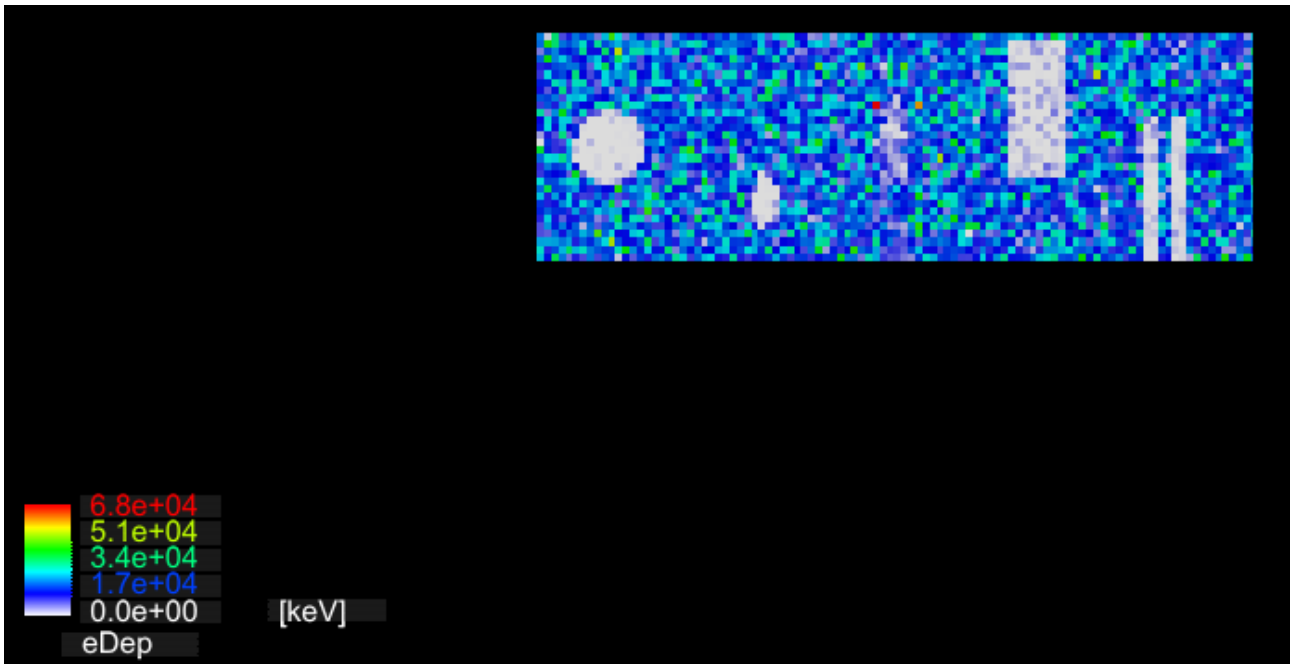

Figure 3: Energy deposits in the gamma detector

To figure out if the tiny golden ring and the human would get better visibility with smaller bin size, I doubled number of bins to eventually get 5 cm $\times$ 5 cm bins: `/score/mesh/nBin 200 60 60`. As we can see from Figure 4, the golden ring (located between the tower of wooden box and lead boxes) is still not visible. In addition, human visibility did not increase either - tissue is not dense enough.

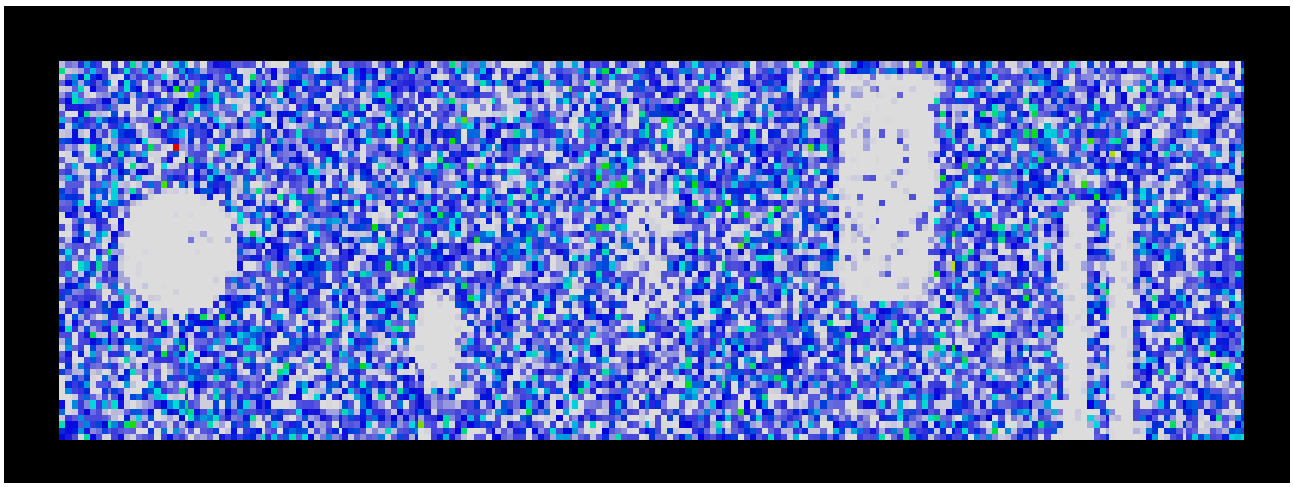

Figure 4: Energy deposits in the gamma detector, 5 cm $\times$ 5 cm bins

3.2 Neutron

Almost identically to the gamma case, 15 MeV neutrons were used as primaries to the scintillator detector. Event example is shown in Figure 5.

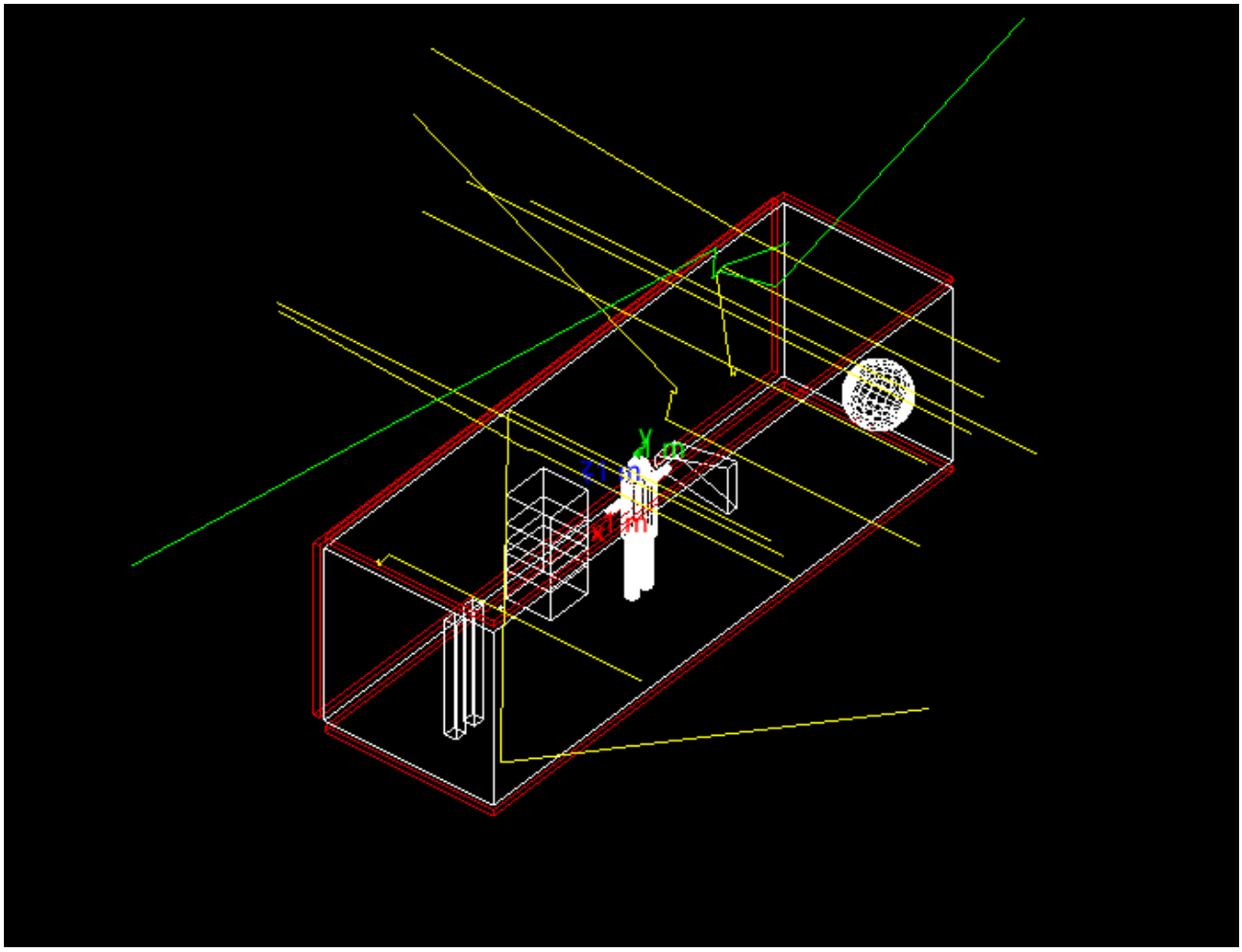

Figure 5: Example of running `/run/beamOn 1` which shoots 10 neutrons (yellow lines).

3.2.1 Energy deposits

As seen in Figure 6, all items except the tiny golden ring are visible in the neutron detector. Even human has absorbed energy and is visible in the middle of the figure.

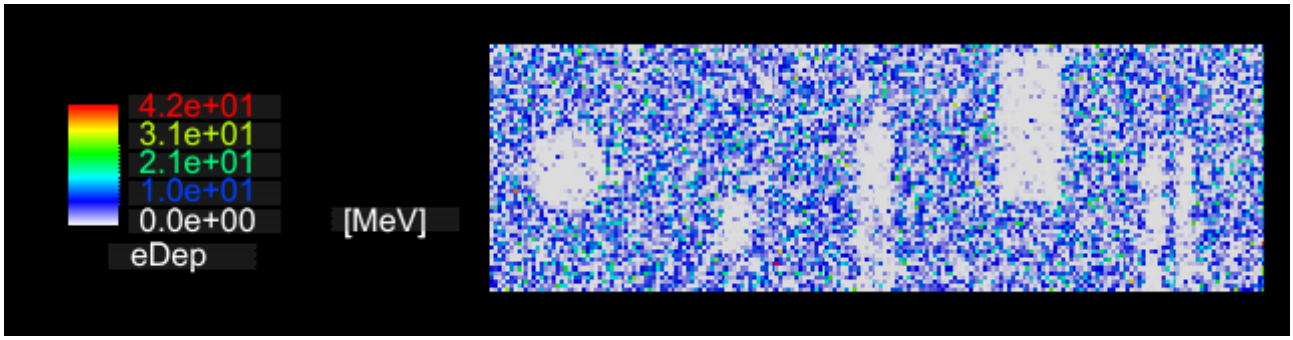

Figure 6: Energy deposits in the neutron detector

3.3 Muon

Muons are shot from top to bottom and the muon detectors of CsI are located on top and bottom of the container. The detectors were treated as sensitive detectors. For that reason no scoring macro was needed. I saved the location of hits on a x-z plane and calculated the scattering angle of the muon. For plotting the scattering angle, I wrote a very simple external ROOT script.
Event example is shown in Figure 7.

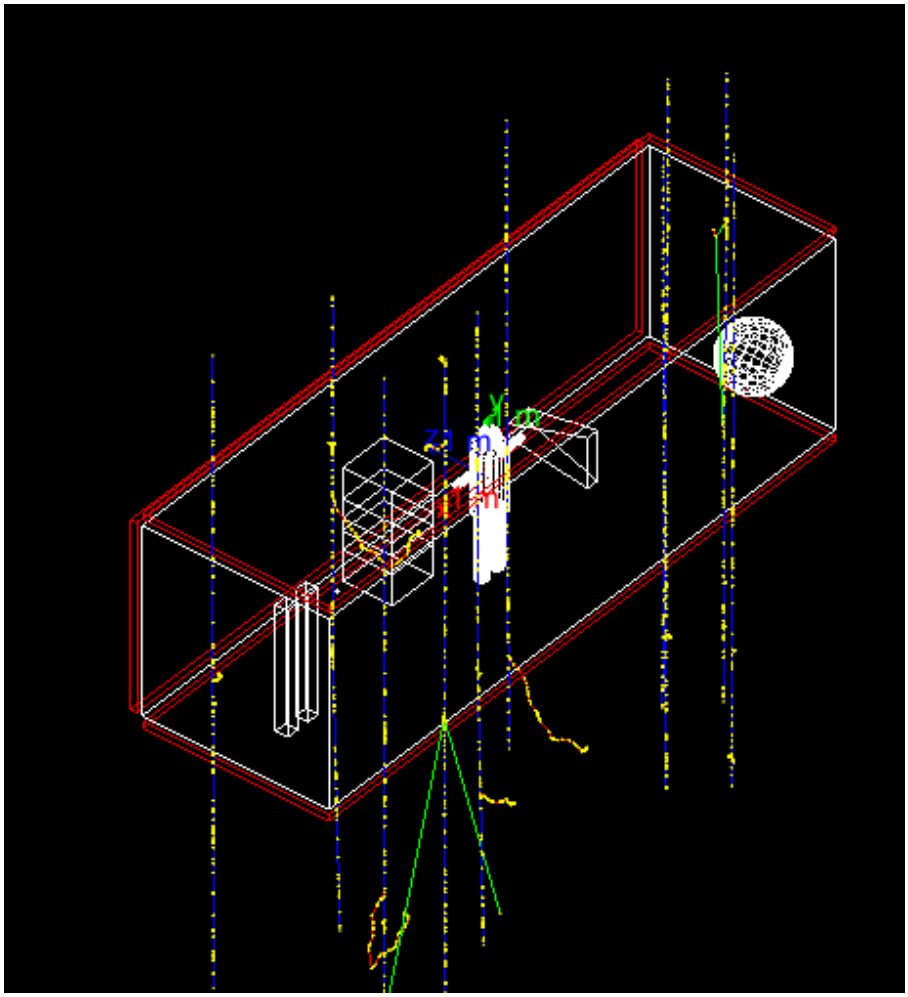

Figure 7: Example of running `/run/beamOn 1` which shoots 10 muons (blue lines).

3.3.1 Hit positions

Hit positions in the upper muon detector plane are shown in Figure 8. We can see that hits are uniformly distributed on the upper plane - which is exactly what we wanted from our particle gun.

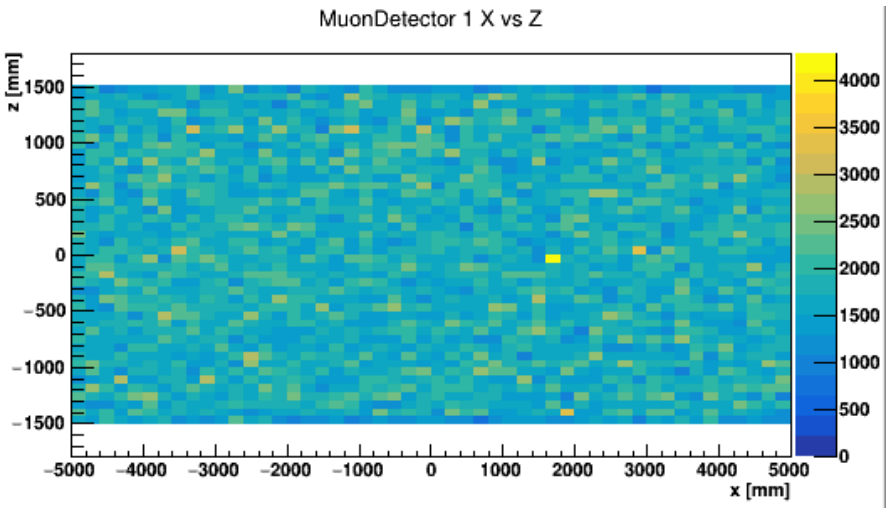

Figure 8: Hit position histogram on the x-z plane in the upper muon detector.

3.3.2 Deflection angle

Deflection angle θ is defined as shown in the schematic Figure 9.

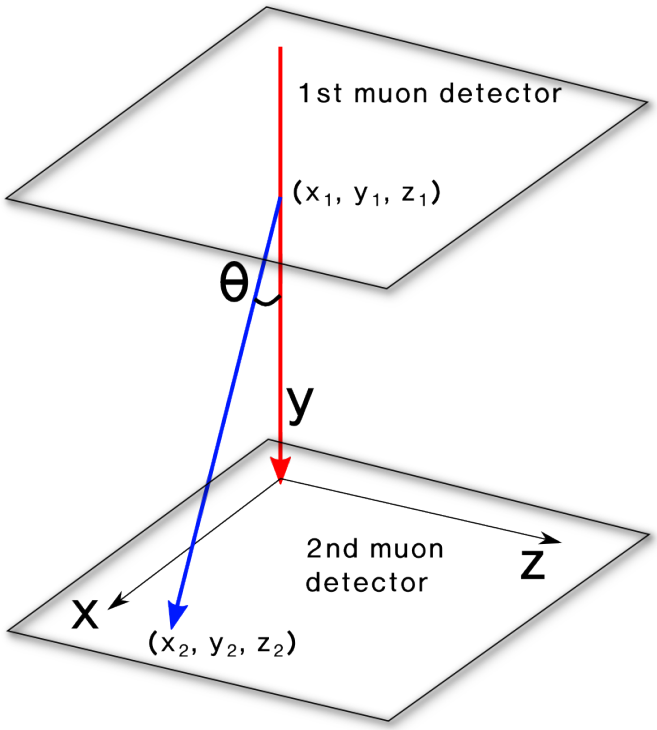

Figure 9: Definition for the deflection angle θ. Red line represents the original muon and blue one the scattered muon. The separation between two detectors is y, the long side of the container is the x direction and the short side is the z direction.

The deflection angle histogram was plotted on the x-z plane and can be seen in Figure 10. Here the angle on the z-axis is naturally not the value of angle itself but the weight of the particular hit. One can notice how the uranium bomb and wooden box tower are most visible features due to their size, while lead boxes, stainless steel trapezoid and human are a bit more dim.

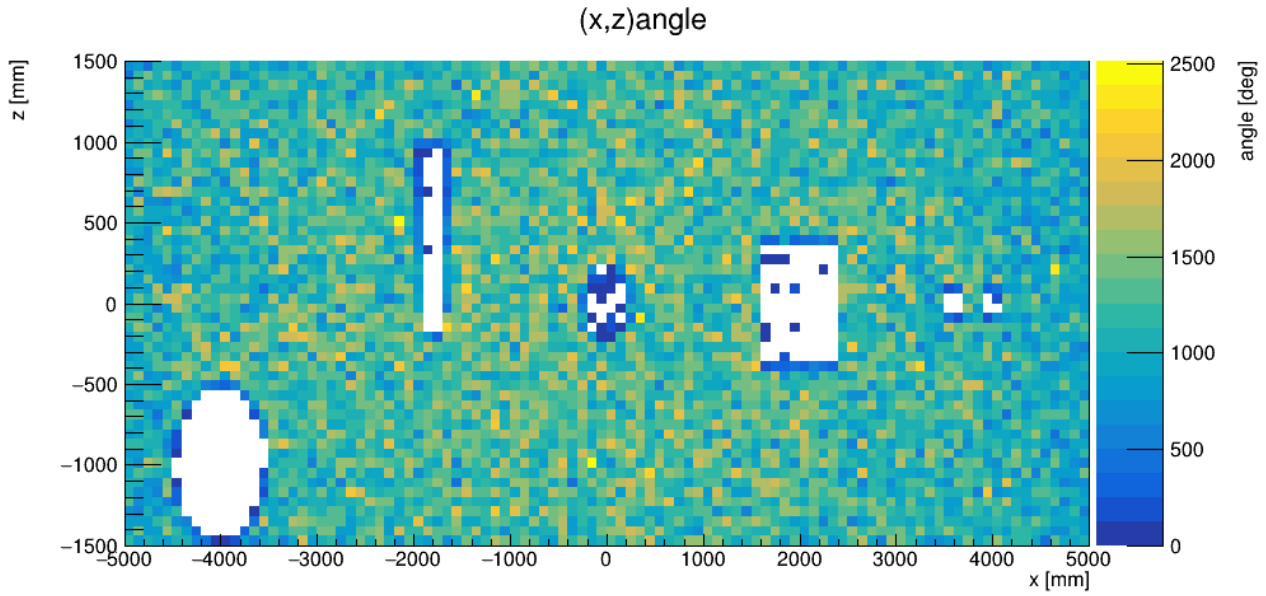

Figure 10: Muon deflection angle on the x-z plane

4 Summary and conclusions

In this project, a simulated aluminium container was filled with various items and then bombarded by different primary particles. All detector and primary particle types were able to see different items except the smallest one - golden ring - which may be explained by too large bin size. Golden rings are thus safe to smuggle through security checks!

Based on different amount of statistics (2000 neutron and gamma events vs 10 000 muon events) and the fact of having two detectors instead of one, it is not entirely fair to state that muon detectors perform the best. Gamma detector was however worse than neutron in terms of detecting people. Probably in the real world we should think about cost, computing time, processing time, radiation safety, etc etc before making any decision on which of these three detectors should be bought by the Homeland security.

In this project, I used relatively low statistics - shooting more primaries would not drastically increase the computing time but would probably give much more clearer images. It would be nice to compare even more detector materials with larger statistics, and more importantly the effect of different physics lists to understand the physics processes better. I was actually a bit surprised how there seemed to be no huge difference between wooden (or cellulose) and lead boxes - I would have doubted that lead absorbed much more. All in all I think this was a funny exercise to do, although openGL+ROOT combination was giving some hard time and quite some segmentation faults with visualization.

A Code: main()

```
#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"

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

```
#include "G4UImanager.hh"
#include "FTFP_BERT.hh"
#include "G4StepLimiterPhysics.hh"

#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"

#include "G4ScoringManager.hh"

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

int main(int argc,char** argv)
{
  //Detect interactive mode (if no argument) and define UI session
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) { //No commands line argument
    //Let G4UIExecutive guess what is the best available UI
    ui = new G4UIExecutive(argc,argv);
  }

  // Construct the default run manager
  // Note that if we have built G4 with support for Multi-threading we set it here
#ifdef G4MULTITHREADED
    G4MTRunManager* runManager = new G4MTRunManager;
    //Set the default number of threads to be the number of available cores of the
        machine
    //If not specified use 2 threads
    runManager->SetNumberOfThreads( G4Threading::G4GetNumberOfCores() );
#else
    G4RunManager* runManager = new G4RunManager;
#endif

    // Activate UI-command base scorer
    G4ScoringManager * scManager = G4ScoringManager::GetScoringManager();
    scManager->SetVerboseLevel(2);

    // Mandatory user initialization classes

    //====================
    //The Geometry
    runManager->SetUserInitialization(new DetectorConstruction);

    //====================
    //The Physics
    G4VModularPhysicsList* physicsList = new FTFP_BERT;
    physicsList->RegisterPhysics(new G4StepLimiterPhysics());
    runManager->SetUserInitialization(physicsList);

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

    // Visualization manager construction
    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();

    if (argc>1) {
        // execute an argument macro file if exist
        G4String command = "/control/execute ";
        G4String fileName = argv[1];
```

```
        UImanager->ApplyCommand(command+fileName);
    }
    else {
      //We have visualization, initialize defaults: look in init_vis.mac macro
      UImanager->ApplyCommand("/control/execute init_vis.mac");
      if (ui->IsGUI() ) {
         UImanager->ApplyCommand("/control/execute gui.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;

    return 0;
}

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

B Code: MuonDetectorHit

```
# This is more or less identical to the Basic Example 5 where I basically replaced drift
    chambers with muon detectors

#include <fstream>
#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"

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

G4ThreadLocal G4Allocator<MuonDetectorHit>* MuonDetectorHitAllocator;

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

MuonDetectorHit::MuonDetectorHit()
: G4VHit(),
  fLayerID(-1), fTime(0.), fLocalPos(0), fWorldPos(0)
{}

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

MuonDetectorHit::MuonDetectorHit(G4int layerID)
: G4VHit(),
  fLayerID(layerID), fTime(0.), fLocalPos(0), fWorldPos(0)
{}

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

```
MuonDetectorHit::~MuonDetectorHit()
{}

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

MuonDetectorHit::MuonDetectorHit(const MuonDetectorHit &right)
: G4VHit(),
  fLayerID(right.fLayerID),
  fTime(right.fTime),
  fLocalPos(right.fLocalPos),
  fWorldPos(right.fWorldPos)
{}

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

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

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

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

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

void MuonDetectorHit::Draw()
{
  auto visManager = G4VVisManager::GetConcreteInstance();
  if (! visManager) return;

  G4Circle circle(fWorldPos);
  circle.SetScreenSize(2);
  circle.SetFillStyle(G4Circle::filled);
  G4Colour colour(1.,1.,0.);
  G4VisAttributes attribs(colour);
  circle.SetVisAttributes(attribs);
  visManager->Draw(circle);
}

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

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

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

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

      (*store)["Time"]
        = G4AttDef("Time","Time","Physics","G4BestUnit","G4double");

      (*store)["Pos"]
```

```
        = G4AttDef("Pos", "Position", "Physics","G4BestUnit","G4ThreeVector");
  }

  return store;
}

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

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

  values
    ->push_back(G4AttValue("HitType","MuonDetectorHit",""));
  values
    ->push_back(G4AttValue("ID",G4UIcommand::ConvertToString(fLayerID),""));
  values
    ->push_back(G4AttValue("Time",G4BestUnit(fTime,"Time"),""));
  values
    ->push_back(G4AttValue("Pos",G4BestUnit(fWorldPos,"Length"),""));

  return values;
}

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

void MuonDetectorHit::Print()
{
  //G4cout << "Muon hit time " << fTime/ns
  //<< " (nsec), (x,y,z): " << fWorldPos.x()
  //<< " " << fWorldPos.y() << " " << fWorldPos.z() << G4endl;
}

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

C List of files

Macros:

- muonrun.mac: to run muon studies
- draw.mac: drawing tests
- vis.mac: visualization of events
- scoring.mac: scoring energy deposits in neutron and gamma detectors

Source codes:

- src/ActionInitialization.cc
- src/EventAction.cc
- src/PrimaryGeneratorAction.cc
- src/DetectorConstruction.cc
- src/MuonDetectorHit.cc
- src/RunAction.cc
- src/MuonDetectorSD.cc
- src/woodBoxParameterisation.cc

- main.cc

Header files:

- include/ActionInitialization.hh
- include/EventAction.hh
- include/PrimaryGeneratorAction.hh
- include/Analysis.hh
- include/MuonDetectorHit.hh
- include/RunAction.hh
- include/DetectorConstruction.hh
- include/MuonDetectorSD.hh
- include/woodBoxParameterisation.hh