Modern Particle Physics Experiments 2021/22

B. Badełek, G. Grzelak, K. Grzelak,
A. Kalinowski, M. Konecki, M. Kuich,
L. Piotrowski, P. Podlaski, A.F. Żarnecki

PART 4: Data analysis

Marcin Konecki
marcin.konecki@fuw.edu.pl, pok. 4.09

Session 11: Simulation of detector response – geant4

about GEANT 4

- Geant4 - toolkit for simulation of passage of particle through matter
- Project started 1994 in order to replace FORTRAN based GEANT 3;
  ~2000 in production; ~2004 ATLAS,CMS,LHCb experiments start using Geant4.
- Geant4 is a C++ toolkit:
  - there is no main program (to be defined by user)
  - each simulation problem requires different configuration (geometry, particles, physics,…) to be defind by user
  - provides necessary components in form of interfaces (called actions in Geant4)
- Official Web page: https://geant4.web.cern.ch
  from there access to documentation (via →UserSupport) and (for impatient) directly to →Download
- today's introduction (we will create s simple application) based on:
  - YouTube tutorial (1-8 of 17): Geant4 Tutorial by "Physics Matters" (Mustafa Schmidt).
  - official geant4 tutorials for 2021,2022 available from: https://geant4.web.cern.ch/support/training
- goal of this session: starting point for your MSc/PhD application / better understand what your "fat" collaboration reconstruction program is doing

step 0 - let's start

Connect to zcobl3 with X forwarding on (ssh –X)

Create an exercise directory

```
$ mkdir exercise; cd exercise
```

To initialize Geant4 on zcobl3 (for bash):

```
$ . ~konec/geant4/geant4-v11.0.1-install/share/Geant4-11.0.1/geant4make/geant4make.sh
```

Prepare makefile for Geant4 B1 example, compile, run:

```
$ cmake ~konec/geant4/geant4-v11.0.1-install/share/Geant4-11.0.1/examples/basic/B1
$ make
$ ./exampleB1
```

enjoy, click green button to generate event
(or type in session window `/run/beamOn 1`), red button to exit

```
$ rm –rf *
```

step 1: main.cc

```
#include <iostream>
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VisManager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"

int main(int argc, char **argv)
{
  G4RunManager *runManager = new G4RunManager();
  //runManager->Initialize();
  G4UIExecutive *ui = new G4UIExecutive(argc,argv);

  G4VisManager *visManager = new G4VisExecutive();
  visManager->Initialize();

  G4UImanager *UImanager = G4UImanager::GetUIpointer();

  ui->SessionStart();

  delete runManager;
  delete visManager;
  delete ui;
  return 0;
}
```

- (create, configure, compile and) run:

```
cp -r ~konec/geant4/exercise/step1/* .
cd build
cmake ..
make
./exercise
```

- `G4RunManager` is mandatory. Responsible for control flow. `Initialize()` needed after adding detector physics and action initialization
- `G4VisManager`, `G4UIExecutive`, `G4UImanager` – for visualization and interactive UI
- program starts but no functionality

step2: creating geometry

```
cp ~konec/geant4/exercise/step2/include/MyDetectorConstruction.h ../include/
```

see virtual method Construct – purpose provide geometrical setup

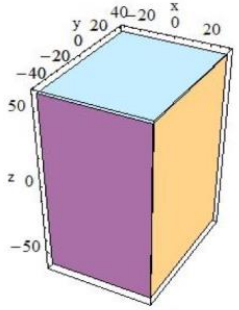
```
cp ~konec/geant4/exercise/step2/src/MyDetectorConstruction.cc  ../src/
```

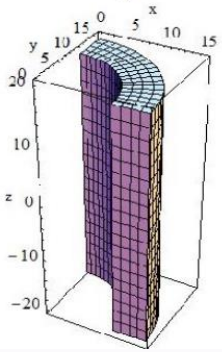
```
G4VPhysicalVolume *MyDetectorConstruction::Construct() {

 G4NistManager *nist = G4NistManager::Instance();
 G4Material *worldMat = nist->FindOrBuildMaterial("G4_AIR");

 G4Box * solidWorld = new G4Box("solidWorld",0.5*m,0.5*m,0.5*m);
 G4LogicalVolume *logicWorld = new G4LogicalVolume(solidWorld, worldMat,"logicWorld");
 G4VPhysicalVolume *physWorld = new G4PVPlacement(0, G4ThreeVector(0.,0.,0.),
                                                  logicWorld, "physWorld", 0, false, 0, true);

 return physWorld;
```

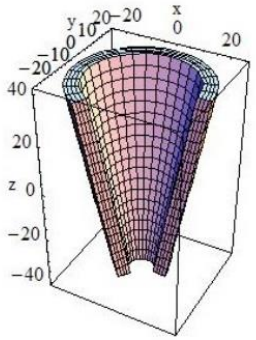
modify main example.cc (include proper header):

```
G4RunManager *runManager = new G4RunManager();
 runManager->SetUserInitialization(new MyDetectorConstruction());
 //runManager->Initialize();
```

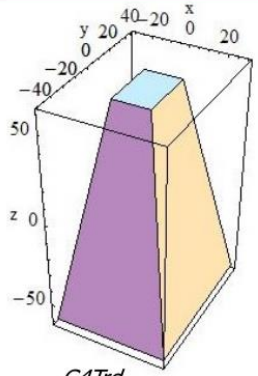
example shapes



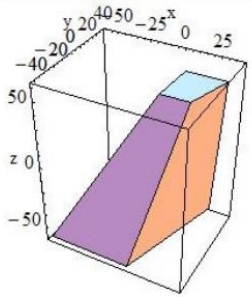




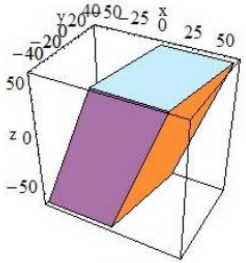
G4Cons



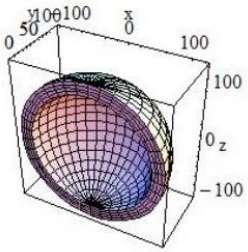
G4Trd



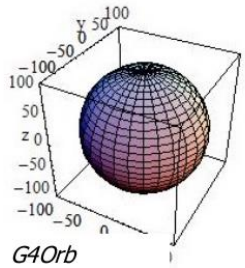
G4Trap



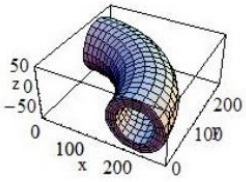
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)



G4Torus

Consult the Geant4 Application Developers Guide for all available shapes

step3: Physics list

- Physics List is an object that is responsible to:
  - specify all the particles that will be used in the simulation application
  - together with the list of physics processes assigned to them
- Provides a very flexible way to set up the physics environment
  - the user can chose, specify the particles that they want to be used
  - the user can chose the physics to assign to each particle

• BUT, the user must have a good understanding of the physics required to describe properly the given problem:
  - omission of relevant particles and/or physics interactions could lead to poor modelling results !!!

- in main add (add proper header):

```
runManager->SetUserInitialization(new MyDetectorConstruction());
G4PhysListFactory physListFactory;
G4String plName = "FTFP_BERT";
G4VModularPhysicsList *pList = physListFactory.GetReferencePhysList(plName);
runManager->SetUserInitialization(pList);
```

"FTFP_BERT“ is Geant4 default, but not good for 10TeV+ energies.

- compile

step4: particle gun

```
cp ~konec/geant4/exercise/step4/include/MyActionInitialization.h ../include/
cp ~konec/geant4/exercise/step4/include/MyPrimaryGenerator.h ../include/
cp ~konec/geant4/exercise/step4/src/MyActionInitialization.cc  ../src/
cp ~konec/geant4/exercise/step4/src/MyPrimaryGenerator.cc  ../src/
```

see virtual method Build in MyActionInitialization and GeneratePrimaries in MyPrimaryGenerator

```
G4ParticleTable * particleTable = G4ParticleTable::GetParticleTable();
G4String beamPart="e-";
G4ParticleDefinition *particle=particleTable->FindParticle(beamPart);
G4ThreeVector pos(0., 0., 0.);
G4ThreeVector mom(0.,0.,1.);
fParticleGun->SetParticlePosition(pos);
fParticleGun->SetParticleMomentumDirection(mom);
fParticleGun->SetParticleMomentum(10.*GeV);
fParticleGun->SetParticleDefinition(particle);
fParticleGun->GeneratePrimaryVertex(ev);
```

fix main:

```
runManager->SetUserInitialization(new MyActionInitialization());
runManager->Initialize();
...
UImanager->ApplyCommand("/vis/open OGL");
UImanager->ApplyCommand("/vis/viewer/set/viewpointVector 1 1 1");
UImanager->ApplyCommand("/vis/drawVolume");
UImanager->ApplyCommand("/vis/viewea/set/autoRefresh true");
UImanager->ApplyCommand("/vis/scene/add/trajectories smooth");
ui->SessionStart();
```

step 5: more materials and volumes

```
cp ~konec/geant4/exercise/step5/include/MyDetectorConstruction.h ../include/
cp ~konec/geant4/exercise/step5/src/MyDetectorConstruction.cc  ../src/
```

check how materials are defined, example:

```
G4Material *Aerogel = new G4Material("Aerogel",0.200*g/cm3,3);
Aerogel->AddMaterial(SiO2, 62.5*perCent);
Aerogel->AddMaterial(H2O, 37.4*perCent);
Aerogel->AddElement(C, 0.1*perCent);
```

placement of detector in mother volume, multiple volumes placement:

```
 G4Box * solidPlate= new G4Box("solidPlate",0.4*m,0.4*m,0.05*m);
 G4LogicalVolume *logicPlate = new G4LogicalVolume(solidPlate, Pb, "logicPlate");
 G4VPhysicalVolume *physPlate= new G4PVPlacement(0, G4ThreeVector(0.,0.,0.25*m),
   logicPlate, "physPlate", logicWorld, false, 0, true);

  int ndiv=10;
  G4Box *solidDet = new G4Box("solidDet",0.5/ndiv*m, 0.5/ndiv*m, 0.01*m);
  logicDet = new G4LogicalVolume(solidDet, worldMat, "logicDetector");

  for (G4int i=0; i<ndiv; i++) {
    for (G4int j=0; j<ndiv; j++) {
      G4VPhysicalVolume *physDet = new G4PVPlacement(0,G4ThreeVector(-0.5*m+(i+0.5)*m/ndiv, -
0.5*m+(j+0.5)*m/ndiv, 0.4*m),logicDet, "physDet",logicWorld, false, j+i*ndiv, true);
    }
}
```

compile, run, check geometry

step 6: sensitive detectors & hits

```
cp ~konec/geant4/exercise/step6/include/MySensitiveDetector.h ../include/
cp ~konec/geant4/exercise/step6/src/MySensitiveDetector.cc  ../src/
```

uncomment lines related to `MySensitiveDetector` in `MyDetectorConstruction`

```
G4bool MySensitiveDetector::ProcessHits(G4Step *aStep, G4TouchableHistory
*ROhist){
  G4Track *track = aStep->GetTrack();
  track->SetTrackStatus(fStopAndKill);
  G4StepPoint *preStepPoint = aStep->GetPreStepPoint();
  G4StepPoint *postStepPoint = aStep->GetPostStepPoint();

  G4ThreeVector position = preStepPoint->GetPosition();
//  G4cout << "secondary position: "<<position<< G4endl;
  const G4VTouchable *touchable = aStep->GetPreStepPoint()->GetTouchable();
  G4int copuNo = touchable->GetCopyNumber();
  G4VPhysicalVolume *physVol = touchable->GetVolume();
  G4ThreeVector posDetector = physVol->GetTranslation();
  G4cout << "Det number :"<<copuNo<<" and position: "<<posDetector << G4endl;

  return true;
}
```

compile, run. Finally fix main to run simulation with no gui (runManager->BeamOn(10);).