Exercise 1(a)

Create a Box of 25cm x 25cm x 50cm filled with liquid Argon

Take size of world box: 1m x1m x 3m.

Take Material of world box: Air

(Detector Construction Class)

Define Detetector Materials:

Air in World Volume

Liquid Ar in Box

```
void B4DetectorConstruction::DefineMaterials()
{
  // Liquid argon material
  G4double a;  // mass of a mole;
  G4double z;  // z=mean number of protons;
  G4double ncomponents, fractionmass;
  G4double density;
  new G4Material("liquidArgon", z=18., a= 39.95*g/mole, density= 1.390*g/cm3);
         // The argon by NIST Manager is a gas with a different density

//Air
a = 14.01*g/mole;
G4Element* elN  = new G4Element("Nitrogen","N", z= 7., a);

a = 16.00*g/mole;
G4Element* elO  = new G4Element("Oxygen"  ,"O", z= 8., a);

density = 1.290*mg/cm3;
G4Material* Air = new G4Material("Air", density, ncomponents=2);
Air->AddElement(elN, fractionmass=0.7);
Air->AddElement(elO, fractionmass=0.3);


  // Print materials
  G4cout << *(G4Material::GetMaterialTable()) << G4endl;
}
```

(Detector Construction Class)

Create Geometry

```
// Create World Volume
G4double world_x = 1.0*m;
G4double world_y = 1.0*m;
G4double world_z = 3.0*m;
'/This creates a box named ¡°World¡± with the extent from -1.0 to 1.0 meters in
X, -1.0 meters to +1.0 meters along the Y axis and from -3.0 to 3.0 meters in Z.
Note that the G4Box constructor takes as arguments the halves of the total box
size.
G4Box* WorldBox = new G4Box("World", world_x, world_y, world_z); // its name and
 size
    G4LogicalVolume* WorldLV
   = new G4LogicalVolume(
              WorldBox,          // its solid
              WorldMaterial,  // its material
              "World");          // its name

  G4VPhysicalVolume* WorldPV
   = new G4PVPlacement(
              0,                 // no rotation
              G4ThreeVector(),  // at (0,0,0)
              WorldLV,           // its logical volume

              "World",           // its name
              0,                 // its mother  volume
              false,             // no boolean operation
              0,                 // copy number
              fCheckOverlaps);  // checking overlaps
```

```
//========================Create a box of dimensions 25cm*25cm*50cm========
 G4double box_x = 25.0*cm;
 G4double box_y = 25.0*cm;
 G4double box_z = 50.0*cm;
 //This creates a box named ¡°box¡± with the extent from -25.0 to 25.0 cm in X,
-25cm to 25cm along the Y axis and from -50cm to 50cm in Z. Note that the G4Box
constructor takes as arguments the halves of the total box size.
4Box* smallBox = new G4Box("Box", box_x, box_y, box_z); // its name and size

 G4LogicalVolume* BoxLV
   = new G4LogicalVolume(
              smallBox,          // its solid
              BoxMaterial,  // its material
              "Box");          // its name



 G4PVPlacement* BoxPV = new G4PVPlacement(
              0,                 // no rotation
              G4ThreeVector(0,0,0),  // at (0,0,0)
              BoxLV,           // its logical volume
              "Box",    // its name
              WorldLV,           // its mother  volume
              false,             // no boolean operation
              0,                 // copy number
              fCheckOverlaps);  // checking overlaps
```

How to Run Exercise 1(a)

$ cd Exercise1

Our first step is to create a build directory in which build the example. We will create this alongside our example source directory as follows:

$ mkdir BOX-build

$ cd BOX-build

Now run CMake to generate the Makefiles needed to build example

$ kavita@kavita: ~/Geant4_BAW_2019/exercise/Exercise1/BOX-build$ **cmake -Dgeant4_DIR=/home/kavita/BAW_2019/geant4-install/lib/Geant4-4.10.05 <space> /home/kavita/Geant4_BAW_2019/exercise/Exercise1/BOX**

Note the Makefile and that all the scripts for running the example application we're about to build have been copied across.

With the Makefile available, we can now build by simply running make:

$ **make -jN(N is the no of core of your system)**

CMake generated Makefiles support parallel builds, so N can be set to the number of cores on your machine (e.g. on a dual core processor, you could set N to 2)

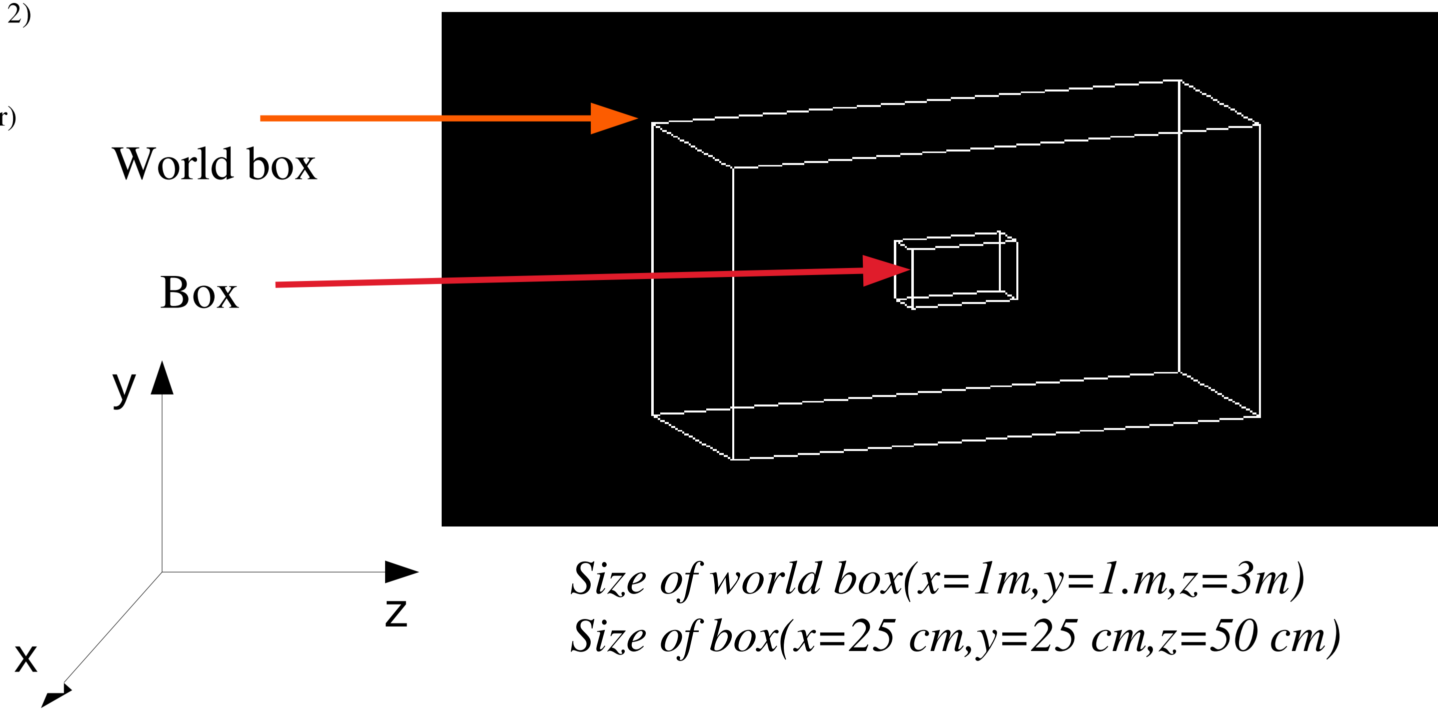
$ ls

This will generate the executable file (green color)

To execute this file, use command

$ **./exampleB4a**

Visualization



Size of world box(x=1m,y=1.m,z=3m)
Size of box(x=25 cm,y=25 cm,z=50 cm)

Exercise 1(b)

Rotate the inside box by 90 deg along X axis with respect to its mother volume.

In GEANT 4, the rotation matrix associated to a placed physical volume represents the rotation of the reference system of this volume with respect to its mother.
A rotation matrix is normally constructed as in CLHEP, by instantiating the identity matrix and then applying a rotation to it

---

Script *Rotation for Coordinate system

```
/*
//========Rotate inside box by 90 degree with respect to its mother volume

// Rotate  box 90 degree from X axis
G4RotationMatrix* boxRot = new G4RotationMatrix();
boxRot->rotateX(90.*deg);

new G4PVPlacement(
               boxRot,
               G4ThreeVector(0,0,0),  // at (0,0,0)
               BoxLV,          // its logical volume
               "Box",    // its name
               WorldLV,          // its mother  volume
               false,            // no boolean operation
               0,                // copy number
               fCheckOverlaps);  // checking overlaps
```

Visualize the Gemotry ?

Exercise 1(c)

Generate the electron beam of energy 50 MeV in extercise 1(a).

Turn On the Electron beam of energy 50MeV in the Exercise1(a).
(PrimaryGeneratorAction Class)

Generate beam (particle type,momentum Direction,Energy)	`// default particle kinematic` `//` `auto particleDefinition = G4ParticleTable::GetParticleTable()->FindParticle("e-");` `fParticleGun->SetParticleDefinition(particleDefinition);` `fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));` `fParticleGun->SetParticleEnergy(50.*MeV);`
Particle gun position	`// Set gun position` `fParticleGun ->SetParticlePosition(G4ThreeVector(0., 0., -1.5*m));` `fParticleGun->GeneratePrimaryVertex(anEvent);`

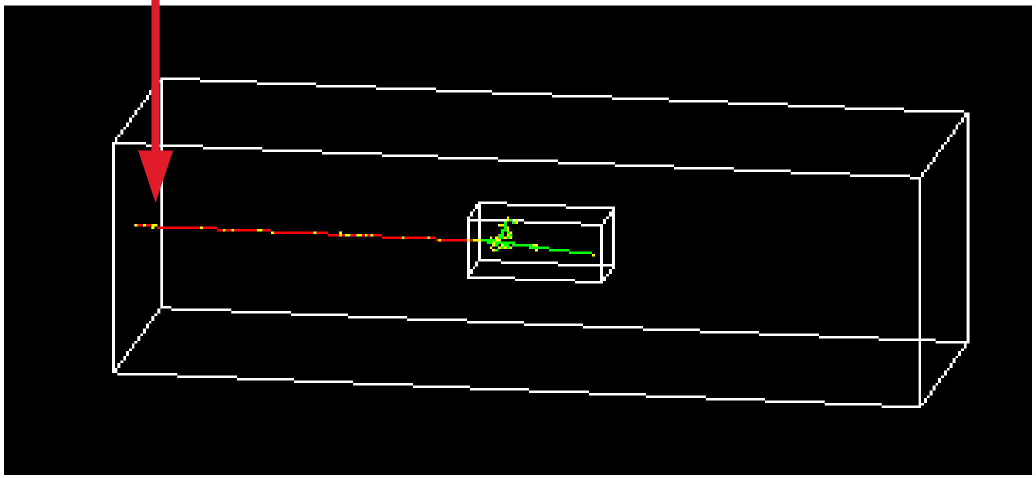
```
// default particle kinematic
//
auto particleDefinition
  = G4ParticleTable::GetParticleTable()->FindParticle("e-");
fParticleGun->SetParticleDefinition(particleDefinition);
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
fParticleGun->SetParticleEnergy(50.*MeV);
      // Set gun position
      fParticleGun
        ->SetParticlePosition(G4ThreeVector(0., 0., -1.5*m));

      fParticleGun->GeneratePrimaryVertex(anEvent);
```

Visualization

- **Turn On Beam-**

$ ./exampleB4a (name of executable file)
on viewer screen type
/run/beamOn 1 (no of events)

Incoming e- beam(Red colour)



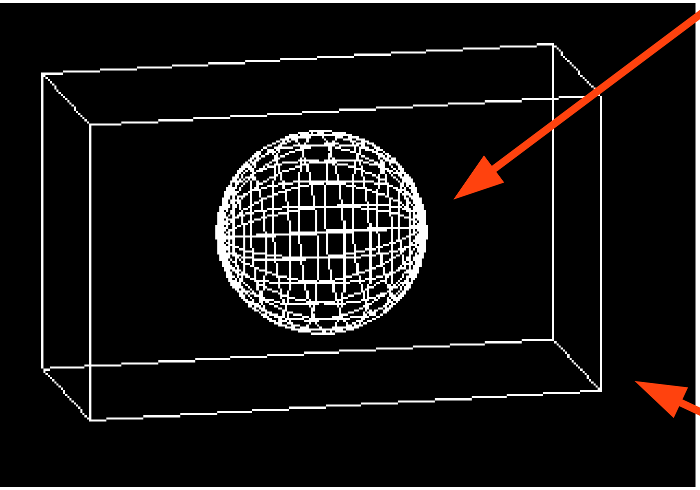
Exercise 2: Create a Sphere of inner radius of 70cm & outer radius 90cm, filled with liquid Argon. Take size of world box of 1m x 1m x 3m filled with Air.

Exercise 2: Create a Sphere

- **Define materials-** Take help from exercise 1
- **Create World** - Take help from exercise 1
- **Create Solid Sphere-**

```
/=========================Create a Sphere=============================
 G4double rmin = 70.*cm;
 G4double rmax = 90.*cm;
 G4double phmin = 0*deg;
 G4double phmax = 360*deg;
 G4double thmin = 0*deg;
 G4double thmax = 180*deg;

 G4Sphere* sphere = new G4Sphere("Sphere", rmin, rmax, phmin, phmax, thmin, thma
x);

uto SphereLV =
   new G4LogicalVolume(sphere,          //its solid
                       SphereMaterial,        //its material
                       "Sphere");            //its name
new G4PVPlacement(0,                       //no rotation
                   G4ThreeVector(),         //at (0,0,0)
                   SphereLV,            //its logical volume
                   "Sphere",              //its name
                  WorldLV,              //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  fCheckOverlaps);         // checking overlaps
```

Visualization



World Volume

Exercise 3 (a)

Create a Tube of inner radius of 30cm,outer radius of 70cm and half length in z of 100cm filled with liquid Argon.

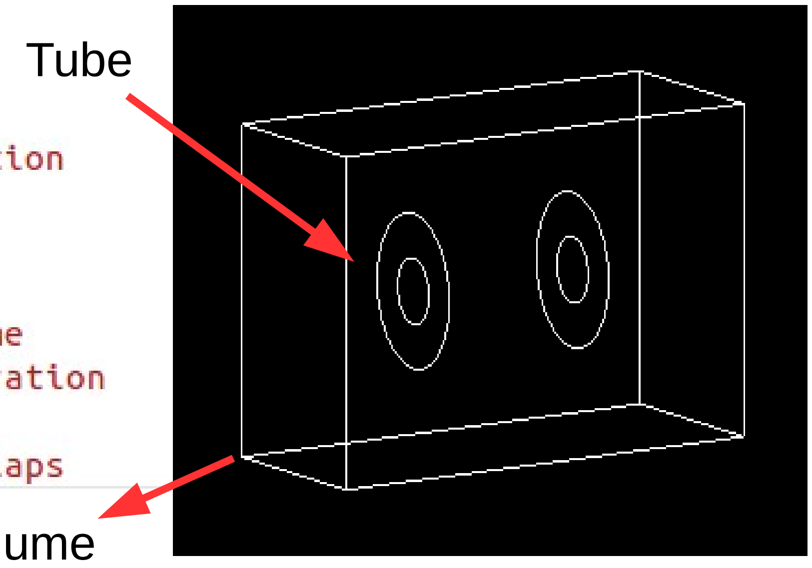
Take size of world in a world of 1m x1m x 3m filled with Air.

Ex. 3 (a) Construction of Tube (Detector Construction Class)

- **Define materials- Take the help from** exercise1 OR 2.
- **Create World – Take the help from** exercise 1 OR 2.
- **Create Tube-**

```
//==========================Create a Tube================================
  G4double rmin = 30.*cm;
  G4double rmax = 70.*cm;
  G4double zh = 100*cm;
  G4double thmin = 0*deg;
  G4double thmax = 360*deg;

  G4Tubs* TubS = new G4Tubs("Tube", rmin, rmax, zh, thmin, thmax);

  G4LogicalVolume* TubLV
    = new G4LogicalVolume(
                 TubS,      // its solid
                 TubeMaterial,  // its material
                 "Tube");    // its name

  G4PVPlacement* TubPV = new G4PVPlacement(0,       //no rotation
                   G4ThreeVector(),          //at (0,0,0)
                   TubLV,             //its logical volume
                   "Tube",                //its name
                   WorldLV,                //its mother  volume
                  false,                    //no boolean operation
                  0,                        //copy number
                  fCheckOverlaps);          // checking overlaps
```

Visualization



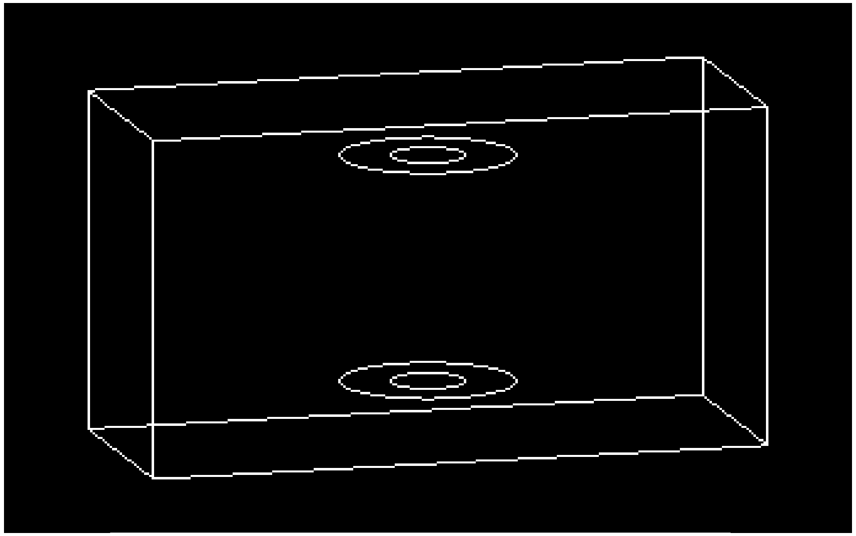
Exercise 3 (b)

Rotate the Tube by 90 degree from X-axis in Exercise 3(a).

Ex. 3(b): Rotate the Tube by 90 degree from X-axis in Exercise 3(a). (DetectorConstruction Class)

In Geant4, the rotation matrix associated to a placed physical volume represents the rotation of the reference system of this volume with respect to its mother.
A rotation matrix is normally constructed as in CLHEP, by instantiating the identity matrix and then applying a rotation to it

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

auto TubPV
      = new G4PVPlacement(
             TubeRot,
            G4ThreeVector(),  // at (0,0,0)
            TubLV,            // its logical volume
            "Tube",      // its name
            worldLV,          // its mother  volume
            false,            // no boolean operatic
            0,                // copy number
            fCheckOverlaps);  // checking overlaps
```



Tube rotated by 90 deg from X-axis

Exercise-4

Design a calorimeter of 10 layers:
Each layer consists one absorber layer + one gap layer

Size of absorber layer is 5cm x 5cm x 5mm
Material of absorber layer is lead

Size of gap layer is 5cm x 5cm x 2.5mm,
Material of gap layer is liquid argon,

Take World Box size of 6cm x 6cm x 18cm,
Take Material of World box Galactic,

Ex. 4: Design of Calorimeter
(DetectorConstruction class)

Geometry :

- The calorimeter is a box made of a given number of layers. A layer consists of an absorber plate and of a detection gap.
- The absorber plate contains lead (Pb) and the gap contains liquid Argon .
- The layer is then replicated using **G4PVReplica** Class**.**

```
             |<----layer 0---------->|<----layer 1---------->|<----layer 2---------->|
             |                       |                       |                       |
             ==========================================================================
             ||                |       ||                |       ||                |       ||
             ||                |       ||                |       ||                |       ||
    beam     ||    absorber    |  gap  ||    absorber    |  gap  ||    absorber    |  gap  ||
  ======>    ||                |       ||                |       ||                |       ||
             ||                |       ||                |       ||                |       ||
             ==========================================================================
```

Pb **liquid Argon**

Ex. 4: Calorimeter Geometry (DetectorConstructionClass)

Create Geometry

```
//
// World
//
auto worldS
  = new G4Box("World",           // its name
           worldSizeXY/2, worldSizeXY/2, worldSizeZ/2); // its size

auto worldLV
  = new G4LogicalVolume(
           worldS,               // its solid
           defaultMaterial,      // its material
           "World");             // its name

auto worldPV
  = new G4PVPlacement(
           0,                    // no rotation
           G4ThreeVector(),      // at (0,0,0)
           worldLV,              // its logical volume
           "World",              // its name
           0,                    // its mother  volume
           false,                // no boolean operation
           0,                    // copy number
           fCheckOverlaps);      // checking overlaps
```

```
G4VPhysicalVolume* B4DetectorConstruction::DefineVolumes()
{
  // Geometry parameters
  G4int nofLayers = 10;
  G4double absoThickness = 10.*mm;
  G4double gapThickness =  5.*mm;
  G4double calorSizeXY  = 10.*cm;

  auto layerThickness = absoThickness + gapThickness;
  auto calorThickness = nofLayers * layerThickness;
  auto worldSizeXY = 1.2 * calorSizeXY;
  auto worldSizeZ  = 1.2 * calorThickness;

  // Get materials
  auto defaultMaterial = G4Material::GetMaterial("Galactic");
  auto absorberMaterial = G4Material::GetMaterial("G4_Pb");
  auto gapMaterial = G4Material::GetMaterial("liquidArgon");

  }
```

Absorber

```
//
// Absorber
//
auto absorberS
  = new G4Box("Abso",                    // its name
               calorSizeXY/2, calorSizeXY/2, absoThickness/2); // its size

auto absorberLV
  = new G4LogicalVolume(
               absorberS,                // its solid
               absorberMaterial,         // its material
               "Abso");                  // its name

fAbsorberPV
  = new G4PVPlacement(
               0,                        // no rotation
               G4ThreeVector(0., 0., -gapThickness/2), // its position
               absorberLV,               // its logical volume
               "Abso",                   // its name
               layerLV,                  // its mother  volume
               false,                    // no boolean operation
               0,                        // copy number
               fCheckOverlaps);          // checking overlaps
```

Gap

```
//
// Gap
//
auto gapS
  = new G4Box("Gap",                     // its name
               calorSizeXY/2, calorSizeXY/2, gapThickness/2); // its size

auto gapLV
  = new G4LogicalVolume(
               gapS,                     // its solid
               gapMaterial,              // its material
               "Gap");                   // its name

fGapPV
  = new G4PVPlacement(
               0,                        // no rotation
               G4ThreeVector(0., 0., absoThickness/2), // its position
               gapLV,                    // its logical volume
               "Gap",                    // its name
               layerLV,                  // its mother  volume
               false,                    // no boolean operation
               0,                        // copy number
               fCheckOverlaps);          // checking overlaps
```

Calorimeter

```
//
// Calorimeter
//
auto calorimeterS
  = new G4Box("Calorimeter",      // its name
              calorSizeXY/2, calorSizeXY/2, calorThickness/2); // its s

auto calorLV
  = new G4LogicalVolume(
              calorimeterS,     // its solid
              defaultMaterial,  // its material
              "Calorimeter");   // its name

new G4PVPlacement(
              0,                // no rotation
              G4ThreeVector(),  // at (0,0,0)
              calorLV,          // its logical volume
              "Calorimeter",    // its name
              worldLV,          // its mother  volume
              false,            // no boolean operation
              0,                // copy number
              fCheckOverlaps);  // checking overlaps

//
```

layer

```
//
// Layer
//
auto layerS
  = new G4Box("Layer",           // its name
              calorSizeXY/2, calorSizeXY/2, layerThickness/2); // its s

auto layerLV
  = new G4LogicalVolume(
              layerS,           // its solid
              defaultMaterial,  // its material
              "Layer");         // its name

new G4PVReplica(
              "Layer",          // its name
              layerLV,          // its logical volume
              calorLV,          // its mot

              kZAxis,           // axis of replication
              nofLayers,        // number of replica
              layerThickness);  // witdth of replica

//
```

Absorber

```
//
// Absorber
//
auto absorberS
  = new G4Box("Abso",                    // its name
              calorSizeXY/2, calorSizeXY/2, absoThickness/2); // its size

auto absorberLV
  = new G4LogicalVolume(
              absorberS,          // its solid
              absorberMaterial,   // its material
              "Abso");            // its name

fAbsorberPV
  = new G4PVPlacement(
              0,                  // no rotation
              G4ThreeVector(0., 0., -gapThickness/2), // its position
              absorberLV,         // its logical volume
              "Abso",             // its name
              layerLV,            // its mother  volume
              false,              // no boolean operation
              0,                  // copy number
              fCheckOverlaps);    // checking overlaps
```

Gap

```
//
// Gap
//
auto gapS
  = new G4Box("Gap",                     // its name
              calorSizeXY/2, calorSizeXY/2, gapThickness/2); // its size

auto gapLV
  = new G4LogicalVolume(
              gapS,               // its solid
              gapMaterial,        // its material
              "Gap");             // its name

fGapPV
  = new G4PVPlacement(
              0,                  // no rotation
              G4ThreeVector(0., 0., absoThickness/2), // its position
              gapLV,              // its logical volume
              "Gap",              // its name
              layerLV,            // its mother  volume
              false,              // no boolean operation
              0,                  // copy number
              fCheckOverlaps);    // checking overlaps
```

How to run Exercise 4 (Calorimeter)

kavita@kavita:~/Geant4_BAW_2019/exercise/calorimeter$ mkdir B4a-build
cd B4a-build/

$cmake -DGeant4_DIR=/home/kavita/BAW_2019/geant4-install/lib/Geant4-4.10.05 /home/kavita/Geant4_BAW_2019/exercise/calorimeter/B4a

$make -j 4

$./exampleB4a

$./example -m run2.mac

for macro

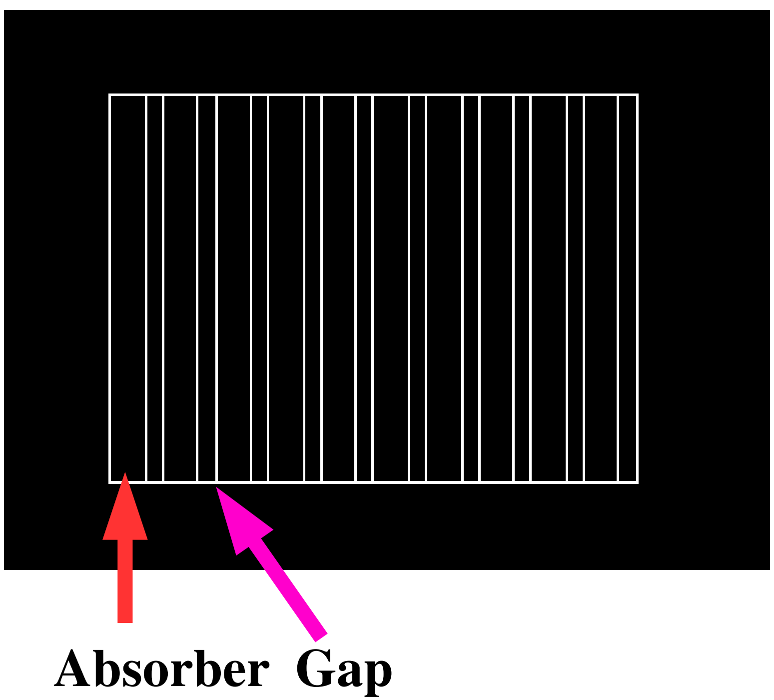
Run time output

```
--> Event 99994 starts.
---> End of event: 99994
   Absorber: total energy: 46.9238 MeV       total track length: 3.53162 cm
        Gap: total energy: 1.46783 MeV       total track length: 8.02112 mm
--> Event 99995 starts.
---> End of event: 99995
   Absorber: total energy: 49.1339 MeV       total track length: 3.65163 cm
        Gap: total energy: 160.133 keV       total track length: 319.648 um
--> Event 99996 starts.
---> End of event: 99996
   Absorber: total energy: 39.1611 MeV       total track length: 2.73209 cm
        Gap: total energy: 4.04338 MeV       total track length: 2.17381 cm
--> Event 99997 starts.
---> End of event: 99997
   Absorber: total energy:  44.778 MeV       total track length: 3.29914 cm
        Gap: total energy: 5.22204 MeV       total track length: 2.86229 cm
--> Event 99998 starts.
---> End of event: 99998
   Absorber: total energy: 47.0693 MeV       total track length: 3.32753 cm
        Gap: total energy:  702.49 keV       total track length: 2.80209 mm
--> Event 99999 starts.
---> End of event: 99999
   Absorber: total energy:  45.769 MeV       total track length: 3.40049 cm
        Gap: total energy: 216.044 keV       total track length: 478.405 um

 ----> print histograms statistic for the entire run

 EAbs : mean = 45.736 MeV rms = 3.8752 MeV
 EGap : mean = 1.6187 MeV rms = 2.01281 MeV
 LAbs : mean = 3.31009 cm  rms = 3.02617 mm
 LGap : mean = 7.95 mm  rms = 1.3516 cm
... write Root file : B4.root - done
... close Root file : B4.root - done
Graphics systems deleted.
Visualization Manager deleting...
kavita@kavita:~/Geant4_BAW_2019/exercise/calorimeter/B4a-build$
```

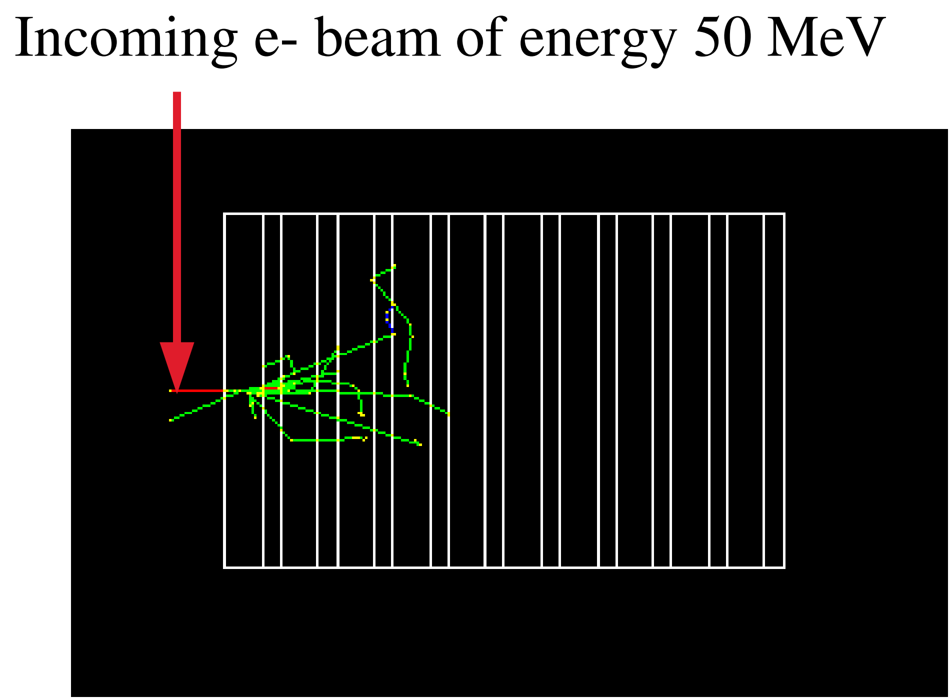
Root Output file

Visualization of Geometry (Calorimeter)

Beam off



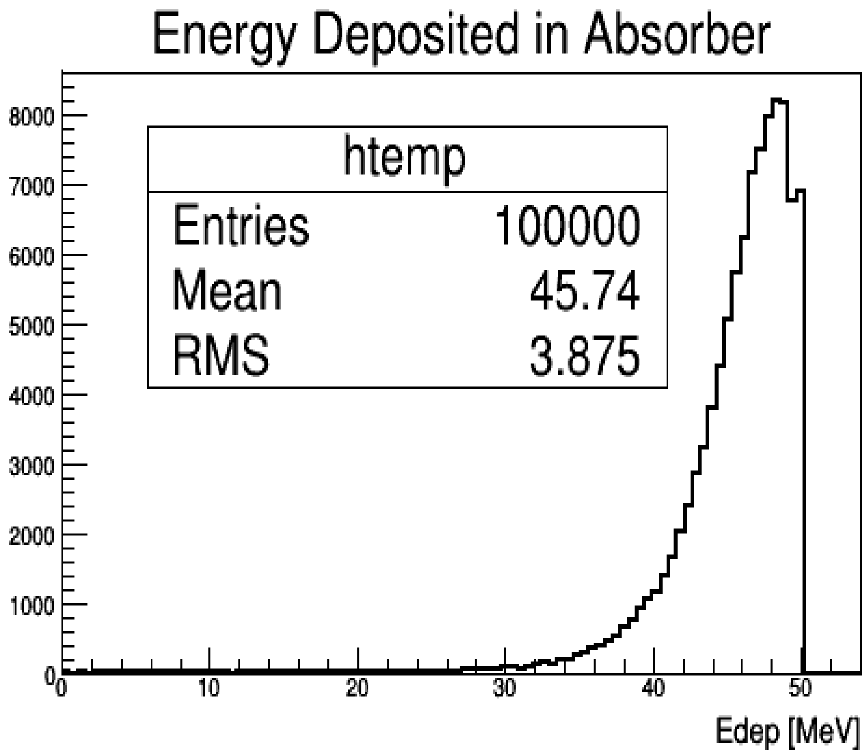
Beam on



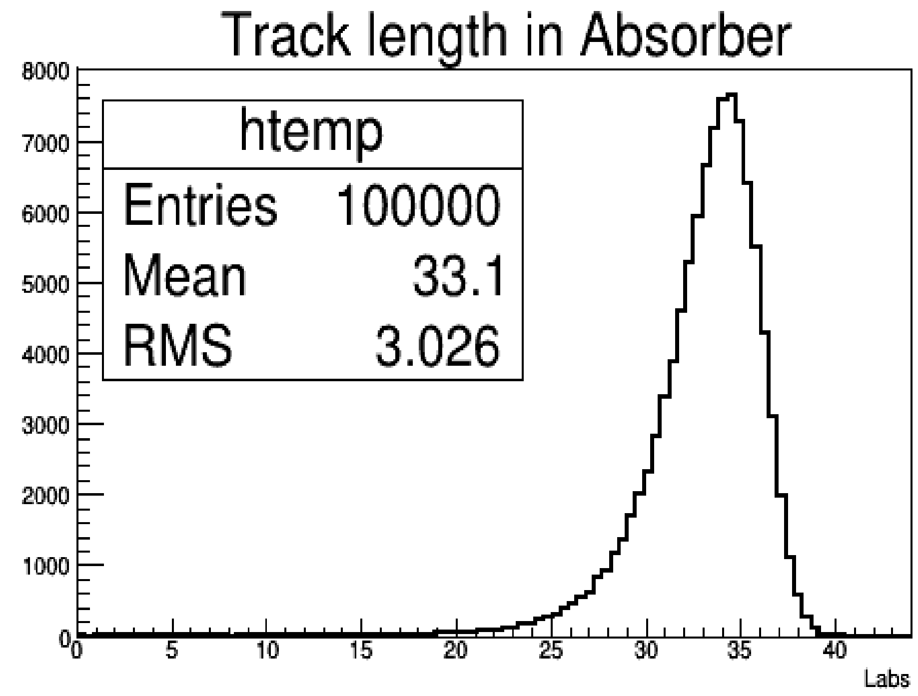
Energy Deposited and Track length (ROOT Output)

(Stepping Action Class and Event Action Class)

Stepping Action Class provide Energy Dep.
And track length on each step



Event Action Class stores the Edep and track
Length event by event at each step in Ntuples
& histograms



Ex. 5: Construct a Detector Setup consisting of two hodoscope, two drift chambers, Electromagnetic Calorimeter, Hadron Calorimeter