Muography Workshop

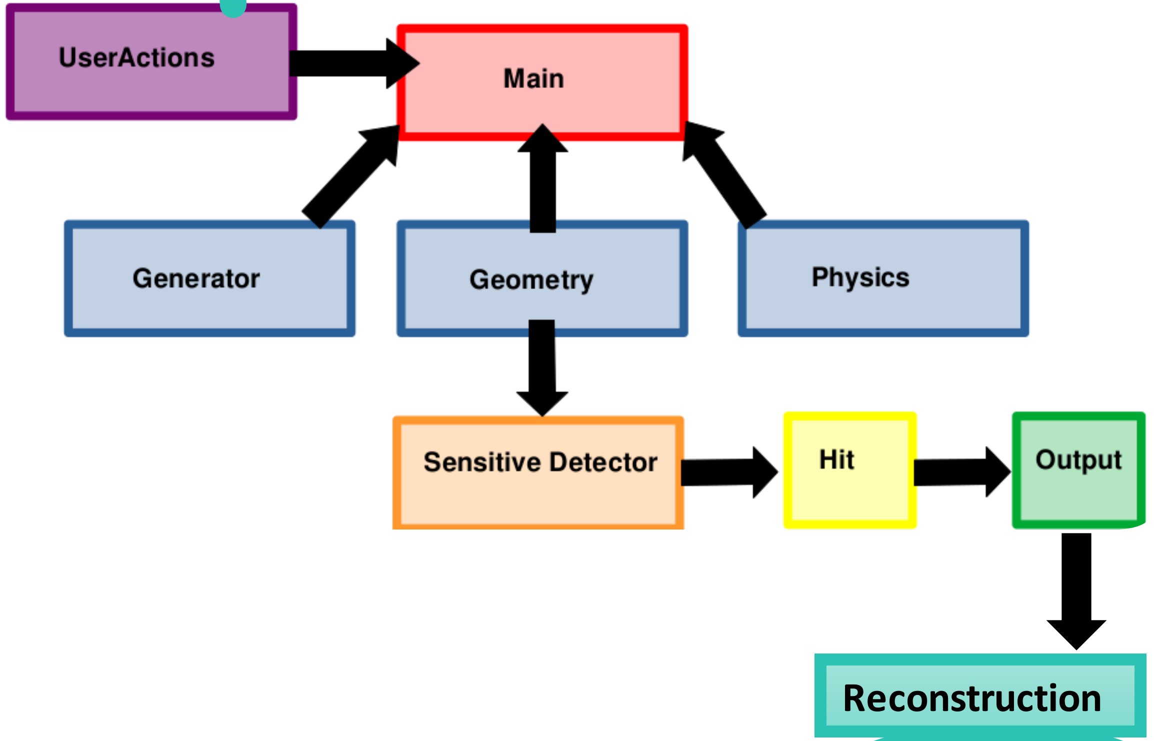
End-to-end simulation framework

Geant4: **GE**ometry **AN**d **T**racking

- C++ toolkit designed for High Energy Physics (HEP) detectors.
- It utilizes Monte Carlo methods to simulate the passage of particles through matter.
- The toolkit includes features for handling geometry, tracking, detector response, run management, visualization, and user interface.
- The simulation serves two main purposes: aiding in the design of detectors during the research and development (R&D) phase, and understanding the detector's response for physics studies.

Simulation Chain

- To create a virtual reality of particle-matter interactions, the toolkit models abstract behaviors of **the interactions**, **geometry**, and **materials**.
- It enables the **propagation** of elementary particles into the detector for simulation.
- The toolkit also describes the **sensitivity of the detector** for generating raw data during simulations



Geant4 Class :

Mandatory	Optional	Main program
<u>G4VUserDetectorConstruction:</u> • Initialize the **physical geometry** of the • simulation. • It necessarily introduces a virtual **Construct()** method which is invoked by the **Initialize()** method of the G4RunManager class included in the main file. • The **Construct()** method must return a pointer to **G4VPhysicalVolume** which represents the volume of simulation space	<u>G4UserRunAction :</u> • **BeginOfRunAction**(const G4Run*): define the data to represent graphically when running. • **EndOfRunAction**(const G4Run*): store the data measured in defined graphic formats.	In main() : 1. Explicitly instantiate **G4RunManager:** This object controls the running of the program and manages the event loop of an execution 2. The **SetUserInitialization()** method is used to instantiate the objects of classes derived from mandatory classes **G4VUserDectectorConstruction** and **G4VUserPhysicsList** 3. The **SetUserAction()** method is used to instantiate an object of the class derived from mandatory class **G4VUserPrimaryGeneratorAction** 4. The **initialize()** method allows to apply the geometric parameters and simulation physics 5. Finally we initialize the desired number of events and launch the execution by the method **beamOn**(int numberOfEvent)
<u>G4VUserPhysicsList:</u> • Constructing physical particles and processes. • The user must implement three virtual methods in their own concrete class derived from this class: • **G4VUserPhysicsList::ConstructParticle():** particle construction. • **G4VUserPhysicsList::ConstructPhysics():** build processes physical. • **G4VUserPhysicsList::SetCuts()**: set of energy values from cutoff in a range for all particles.	<u>G4UserEventAction :</u> • **BeginOfEventAction**(const G4Event*) : selects an event and data to be represented in histograms • **EndOfEventAction**(const G4Event*) : analyzes the event with graphs	
<u>G4VUserPrimaryGeneratorAction :</u> • For particle generation. • It introduces a method • virtual **GeneratePrimaries()** which is invoked by the class G4RunManager during execution	<u>G4UserSteppingAction :</u> • Collects the different information on a particle (energy, displacement vector etc.) at time dt	

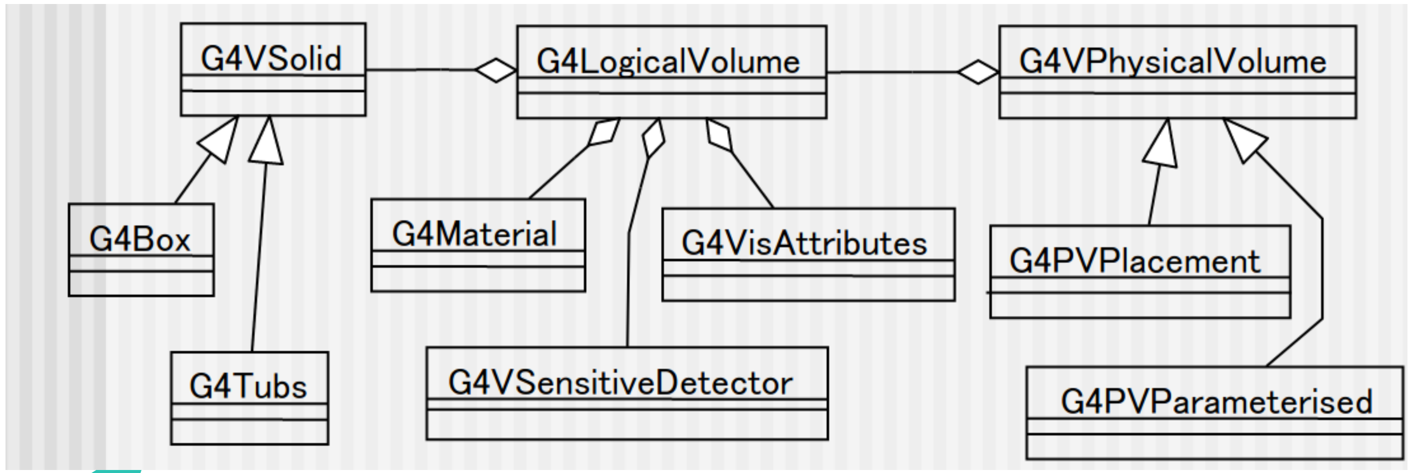
The implementation of a simulation under the GEANT4 workbench requires a set of classes that the user derives to create his own simulation codes

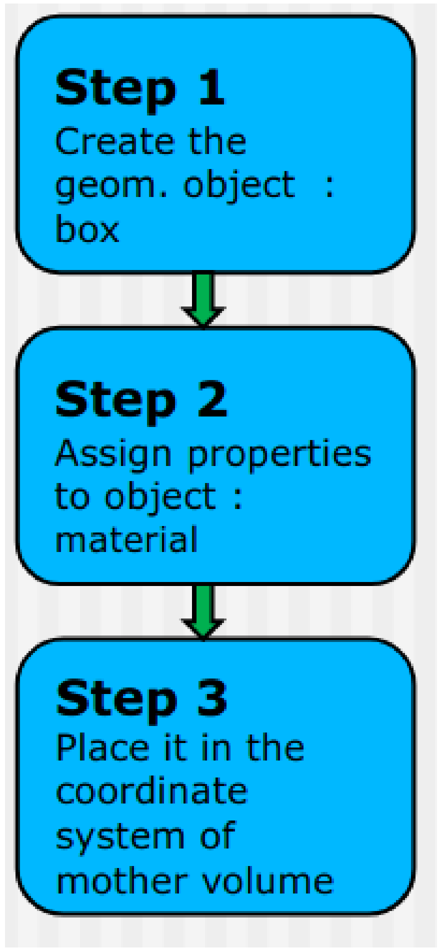
Basics:
Detector Geometry

How we create a volume?

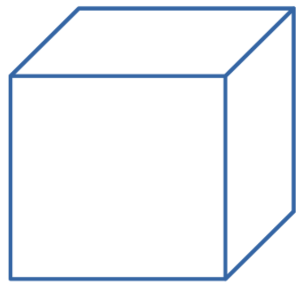
The detector geometry defined in your **G4VUserDetectorConstruction**:

1. G4VSolid: Create a solid (shape, dimension)
2. G4LogicalVolume: Create a logical volume (material, sensitivity, magnetic field, etc)
3. G4VPhysicalVolume: Create a physical volume (position, rotation, copy ...)




Step 1
Create the geom. object : box
Step 2
Assign properties to object : material
Step 3
Place it in the coordinate system of mother volume

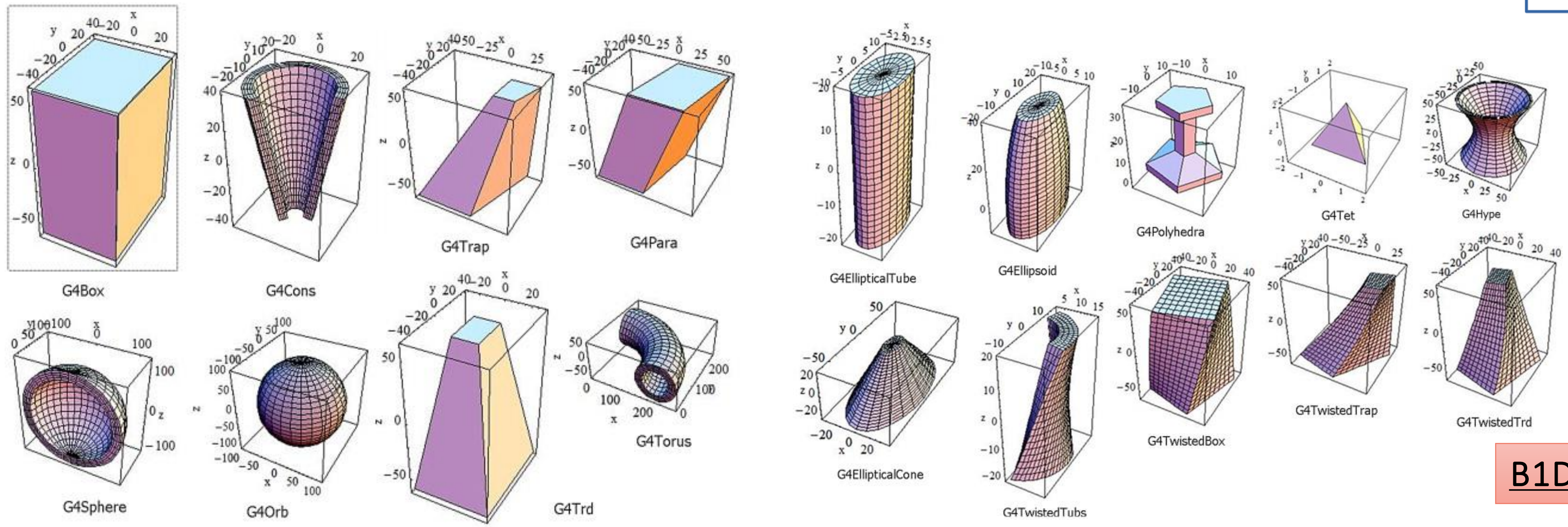
Shape and size



Step 1
Create the geom. object : box

A Solid can defined using **G4VSolid():**

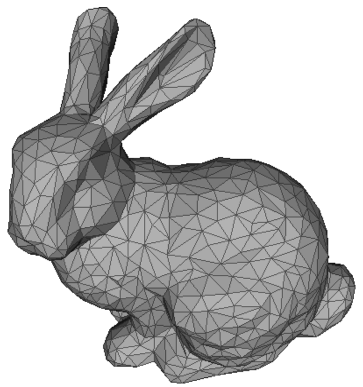
- The simple classes in Geant4 where we can specify the shape of the volume :



B1DetectorConstruction.cc

```
 //Example  box volume
 G4double dimXY = 2*m, dimZ = 2*m;
 G4Box* solidVolume= new G4Box("VolumeSol",                    //its name
                               0.5*dimXY, 0.5*dimXY, 0.5*dimZ);    //its size
```

- Importing CAD models as tessellated shapes using different tools : InStep, SALOME, CADMesh,B2G4..

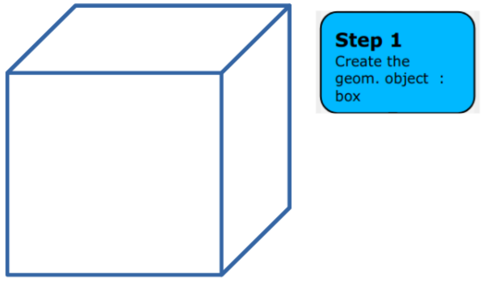


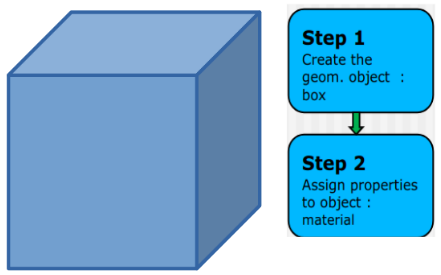
Logical volume and material

B1DetectorConstruction.cc

- The Logical Volume manages the information associated with detector elements represented by a given Solid and Material, independently from its physical position in the detector => Combine the solid shape with a material.

already defined   Need to be defined →





Step 1
Create the geom. object : box
Step 2
Assign properties to object : material

```
//Example LogicVolume
//Get nist material manager and access to lead material
 G4Material* lead_mat = nist->FindOrBuildMaterial("G4_Pb");
G4LogicalVolume* logicVol =
            new G4LogicalVolume(solidVolume,        //its solid
                lead_mat,          //its material
                "VolumeLog");         //its name
```

- Use pedefined materials in Geant4, ex : G4_Fe, G4_Al, G4_AIR....*https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Appendix/materialNames.html*

- Create custom made materials by specifying their properties: density, elements, mass fraction... For thet we use **G4Material** and **G4Element, ex :**

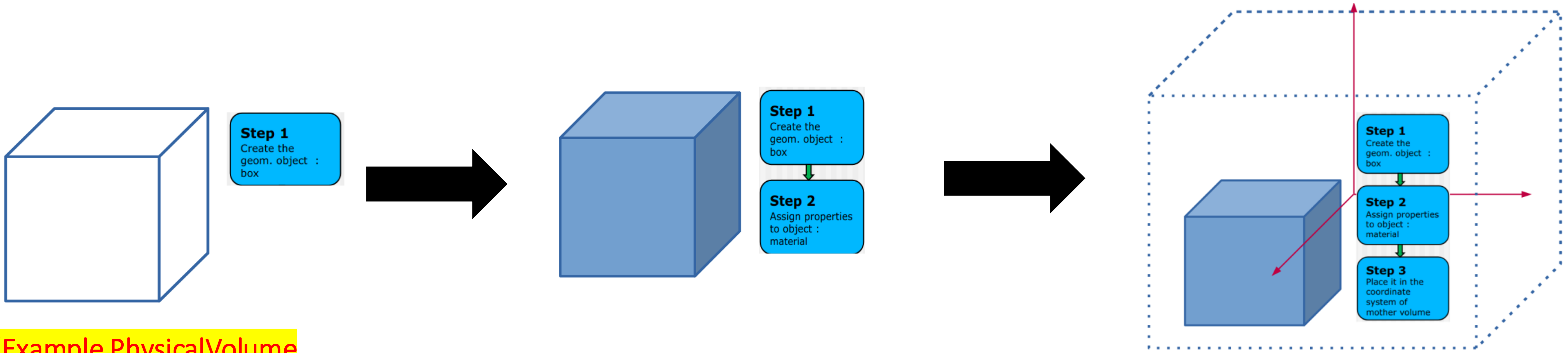
```
•//C2H2F4 freon
   density = 4.25*mg/cm3;
  G4Material* freon = new G4Material(name="freon",density,nel=3);
  freon->AddElement(elC,2);
  freon->AddElement(elH,2);
  freon->AddElement(elF,4);
```

N.B: in case you have a mixe of material you use AddMaterial(materialName,fractionmass)

- *Our sensitive detector SD, will be defined according to the logical volume*

Physical volume and position

- Position and Rotation: After creating the logical volume, you can set its position and rotation in the global coordinate system. This allows you to place the volume at specific locations within your detector setup.


Step 1
Create the geom. object : box
Step 1
Create the geom. object : box
Step 2
Assign properties to object : material
Step 1
Create the geom. object : box
Step 2
Assign properties to object : material
Step 3
Place it in the coordinate system of mother volume

//Example PhysicalVolume

```
G4VPhysicalVolume* physVolume =  new G4PVPlacement(0,                  //no rotation
                G4ThreeVector(0,0,-5*cm),      //at (0,0,0)
                logicVol,          //its logical volume
                "WorldPhys",            //its name
                logicWorld,                  //its mother  volume
                false,             //no boolean operation
               0,                 //copy number
                checkOverlaps);       //overlaps checking
```

B1DetectorConstruction.cc

All the volume must be set it either in mother volume or World volume

Very important when we define our detector Pannel, since it give us an information about the ID

Basics:

A DETECTOR GEOMETRY IN GEANT4 IS MADE OF A NUMBER OF VOLUMES

THE BIGGEST VOLUME IS CALLED THE WORLD VOLUME CONTAIN ALL OTHER VOLUMES IN THE DETECTOR GEOMETRY

THE OTHER VOLUMES ARE CREATED AND PLACED INSIDE IT

THE MOST SIMPLE (AND EFFICIENT) SHAPE TO DESCRIBE THE WORLD IS A BOX

ATTENTION: THE WORLD SHOULD BE AT (0,0,0) AND IT CANNOT BE ROTATED

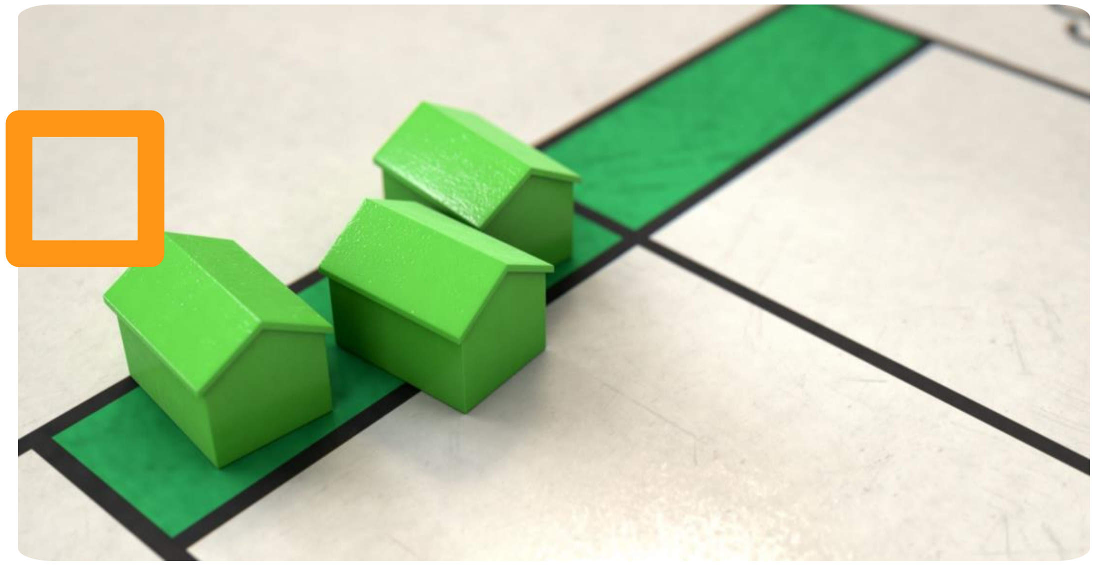
EACH VOLUME IS CREATED BY DESCRIBING ITS SHAPE AND ITS PHYSICAL CHARACTERISTICS, AND THEN PLACING IT INSIDE A CONTAINING VOLUME

```
//
// Mother Volume
//
G4Material* mother_mat = nist->FindOrBuildMaterial("G4_Galactic");
G4Box* solidMother =
  new G4Box("MotherB",                     //its name
      0.5*env_sizeXY, 0.5*env_sizeXY, 0.5*env_sizeZ); //its size

G4LogicalVolume* logicMother =
  new G4LogicalVolume(solidMother,           //its solid
                      mother_mat,            //its material
                      "MotherL");          //its name

new G4PVPlacement(0,                       //no rotation
                  G4ThreeVector(0,0,-2*cm),        //at (0,0,0)
                  logicMother,               //its logical volume
                  "MotherS",               //its name
                  logicWorld,               //its mother  volume
                  false,                    //no boolean operation
                  0,                        //copy number
                  checkOverlaps);           //overlaps checking
```

Example how to create mother volume



Steps to make your geometry

1

Create your World geometry

2

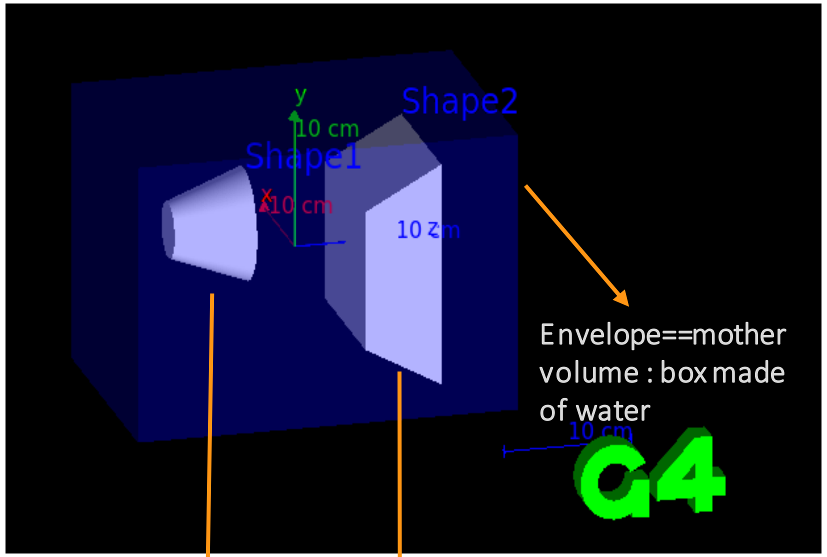
Create your mother geometry

3

Create your detector geometry

Play with B1DetectorConstruction: Exercice1

1. Activate your micromamba :micromamba activate geant-root
2. ./exampleB1



Shape1: Cons made of tissue

Shape2: Trds made of bone

B1DetectorConstruction.cc

1. Change the dimension of shape1: *shape1_rmaxa=4cm* ; *shape1_rmaxb=8cm*; *Save*
2. Now in terminal : make && ./exampleB1
3. Move shape1 by changing *pos1= G4ThreeVector(0*cm, 5*cm,-1*cm)*, *Save* and redo step 2.
4. Change the dimension of shape1 and make it bigger like *shape1_rmxa 6cm*, *shape1_rmxb=12 cm*, *Save* and redo step 2. what do you see in the output?
5. Change the dimension of your world and envelope by changing : G4double env_sizeXY = 20*cm, env_sizeZ = 30*cm; to a few of meters 2*m and 3*m respectively , *Save* and redo step 2.what do you see in the output?
6. Change the material: "*G4_A-150_TISSUE*" to lead "*G4_Pb*", *Save* and redo step 2.
7. Change shape2 to a box :

```
/*G4Trd* solidShape2 = new G4Trd("Shape2",                //its name
     0.5*shape2_dxa, 0.5*shape2_dxb, 0.5*shape2_dya, 0.5*shape2_dyb, 0.5*shape2_dz); //its
size
*/
 G4Box* solidShape2 = new G4Box("Shape",
                      0.5*shape2_dxa,0.5*shape2_dxb,0.5*shape2_dya);
```

, *Save* and redo step 2.

Exercice 2

1. Change dimension of the World from cm ->m (like 20m)
2. Create your mother volume: in B1 example : Envelope.
   1. Simply replace the material from G4_water->G4_AIR
   2. Change place in G4VPlacement : G4ThreeVector(0,0,0) to G4ThreeVector(0,0,-1*cm) (To make sure that all the geometry in the negative direction)
3. Shape1: detector and Shape2:Volume of Intersest
4. Shape1: 6 Boxes of dimensions (1m,1m,0.02m), using plastic scintillator material place it in your mother or envelope.
   1. One box "solidDetector " : **dimensions (1m,1m,0.02m)**
   2. One logic volume " logicDetector": **Plastic scintillator material : G4_PLASTIC_SC_VINYLTOLUENE**
   3. 6 placements => 6 positions: (0,0,z)
      1. G4double zPosSc0 = 0*cm;
      2. G4double zPosSc1 = - 20*cm;
      3. G4double zPosSc2 = - 40*cm;
      4. G4double zPosSc3 = - 80*cm;
      5. G4double zPosSc4 = - 100*cm;
      6. G4double zPosSc5 = - 120*cm;
      7. Define an array of z positions using G4double zPosSc[]={zPosSc0,zPosSc1,zPosSc2,zPosSc3,zPosSc4,zPosSc5}
      8. Loop over the Position and place the pannels in our mother world or envelope
      9. Give an ID for each pannel using copy number = i
      10. Change: fScoringVolume = logicDetector ;

B1DetectorConstruction.cc

Exercice 2

1. Change dimension of the World from cm ->m (like 20m)
2. Create your mother volume: in B1 example : Envelope.
   1. Simply replace the material from G4_water->G4_AIR
   2. Change place in G4VPlacement :G4ThreeVector(0,0,0) to G4ThreeVector(0,0,-1*cm) (To make sure that all the geometry in the negative direction)
3. Shape1: detector and Shape2:Volume of Intersest
4. Shape1: 6 Boxes of dimensions (1m,1m,0.02m), using plastic scintillator material place it in your mother or envelope.
   1. **G4Material* detector_mat = nist->FindOrBuildMaterial("G4_PLASTIC_SC_VINYLTOLUENE");**
   2. One box "SolidDetector or you can keep the name solidShape1":
      G4Box *solidDetector = new G4Box("solidDetector", 0.5*m, 0.5*m, 0.01*m);
   3. One logic volume "LogicDetector":
      G4LogicalVolume* logicDetector = new G4LogicalVolume(solidDetector, detector_mat, "logicDetector");
   4. 6 placements => 6 positions:
      1. G4double zPosSc0 = 0*cm;
      2. G4double zPosSc1 = - 20*cm;
      3. G4double zPosSc2 = - 40*cm;
      4. G4double zPosSc3 = - 80*cm;
      5. G4double zPosSc4 = - 100*cm;
      6. G4double zPosSc5 = - 120*cm;
      7. Define an array of z positions using G4double zPosSc[]={zPosSc0,zPosSc1,zPosSc2,zPosSc3,zPosSc4,zPosSc5}
      8. Loop over the Position and place the pannels in our mother world or envelope
      9. Give an ID for each pannel using copy number = i
      10. Change: fScoringVolume = logicDetector ;

B1DetectorConstruction.cc

Exercice 2

- Step 4.8 && 4.9

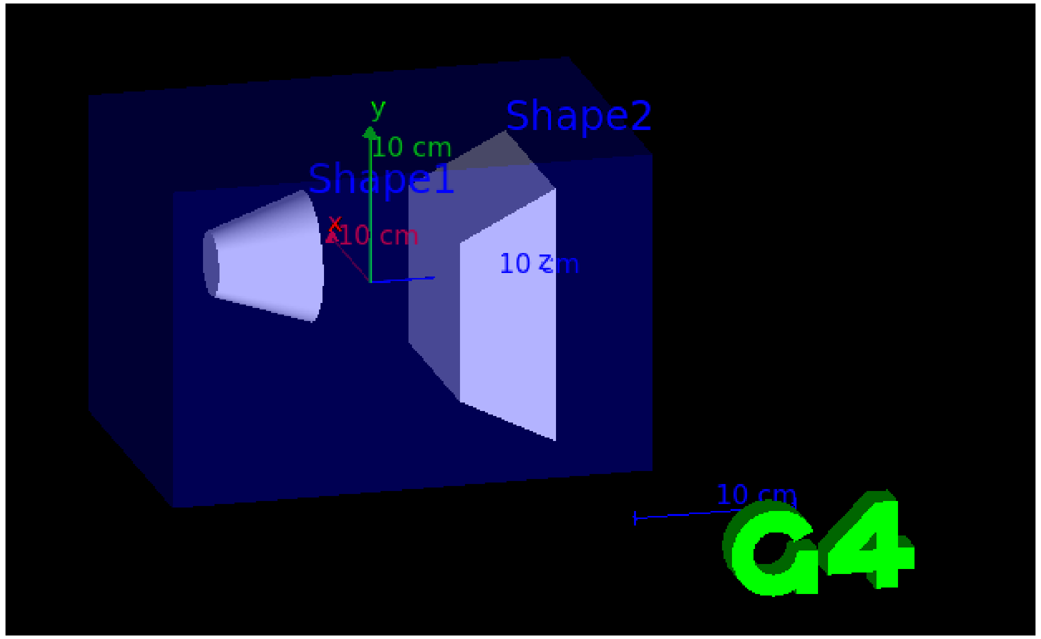
```
//Copy number
 G4int CopyNo=0;
 //Just for naming purpose
 std::stringstream PannelName;
 for (G4int iSc=0;iSc<6;iSc++){
        //give the CopyNo like an identity for each plane
          CopyNo = iSc ; //=> 0 1 2 3 4 5 for planes
           PannelName << iSc;
           new G4PVPlacement(0,              //no rotation
                        G4ThreeVector(0,0,zPosSc[iSc]),  //its position
                        logicDetector,     //its logical volume
                        "Pannel" + PannelName.str(), //its name
                         logicEnv         //its mother
                         false,           //no boulean operat
                         CopyNo);              //copy number
        }
```

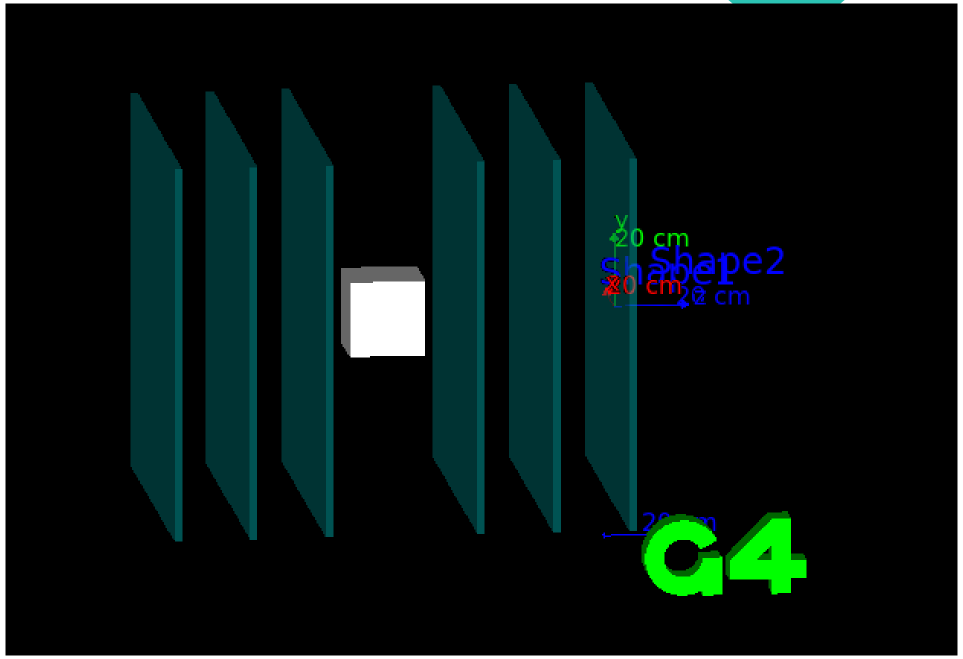
B1DetectorConstruction.cc

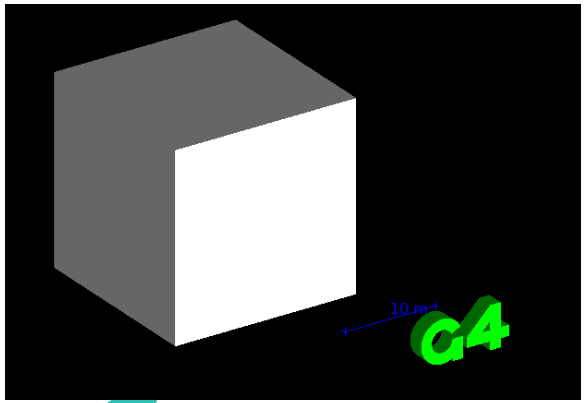
5. Shape 2: Box of Lead "G4_Pb" with dimension (20x20x20cm^3) located @ (0,0,-60*cm)

6. When the code is ready: go to the terminal:

   1. make
   2. Activate micromamba : micromamba activate geant-root
   3. ./exampleB1







In case you see your geometry like that : please do the following :

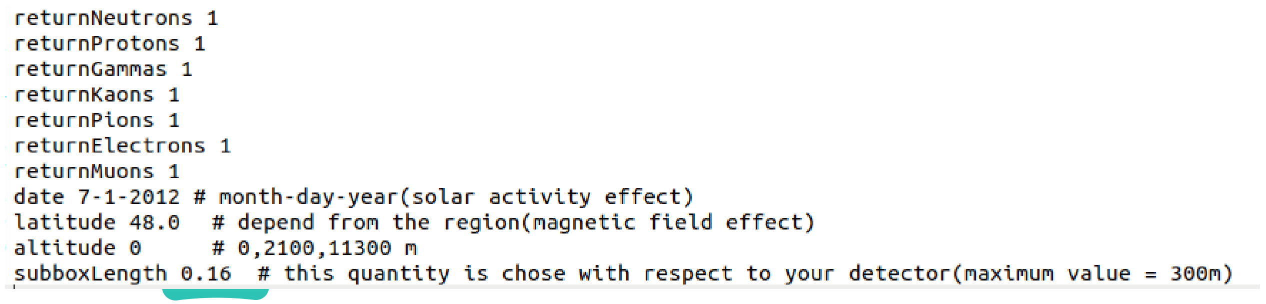
```
#include "G4VisAttributes.hh"
                    .
                    .
  logicEnv->SetVisAttributes(G4VisAttributes::GetInvisible());
  logicWorld->SetVisAttributes(G4VisAttributes::GetInvisible());

  //always return the physical World
  //
  return physWorld;
}
```

Generator

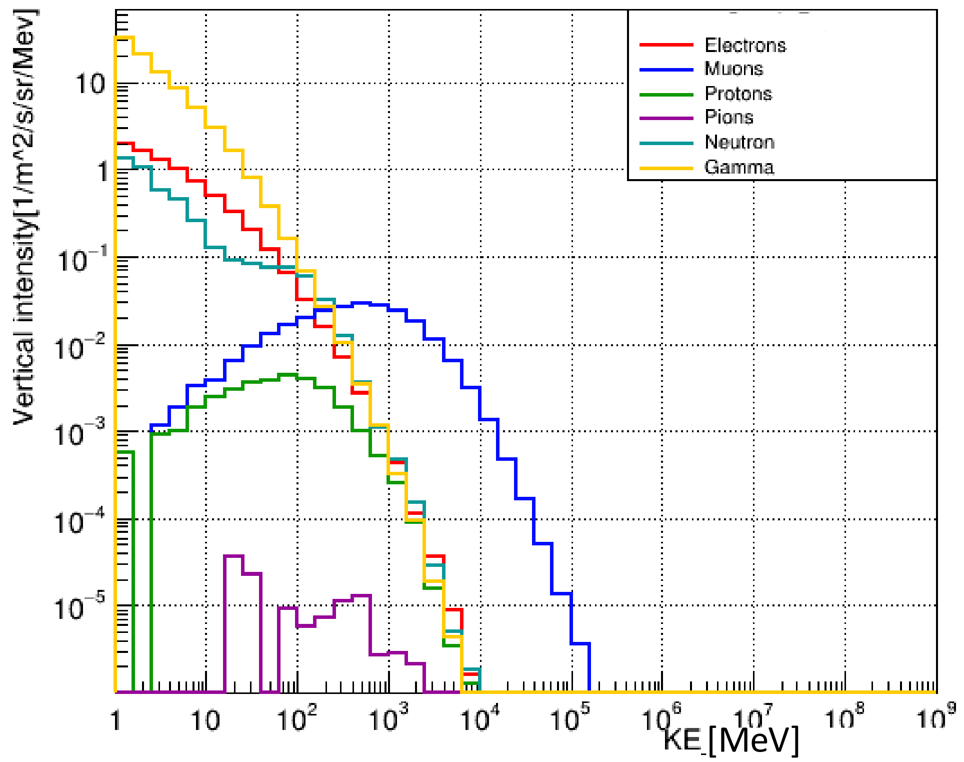
CRY: **Co**smic **RaY** Shower Library

- A Monte Carlo parametric simulation:
- The flux and the kinematics of all **secondaries** (μ, n, p, e, γ, π, K) tabulated from MCNPX*, assuming showers from protons (1 GeV-100 TeV)
- Take into acccount geomagnetic effects on the cosmic flux depending on the **time** (solar cycle), **latitude** and **altitude** (provide 3 options : 0, 2100, 11300 m)
- Limited to flat surface (with surface = **subboxLength**^{2} [m^2])



```
returnNeutrons 1
returnProtons 1
returnGammas 1
returnKaons 1
returnPions 1
returnElectrons 1
returnMuons 1
date 7-1-2012 # month-day-year(solar activity effect)
latitude 48.0   # depend from the region(magnetic field effect)
altitude 0      # 0,2100,11300 m
subboxLength 0.16  # this quantity is chose with respect to your detector(maximum value = 300m)
```

Input setup file

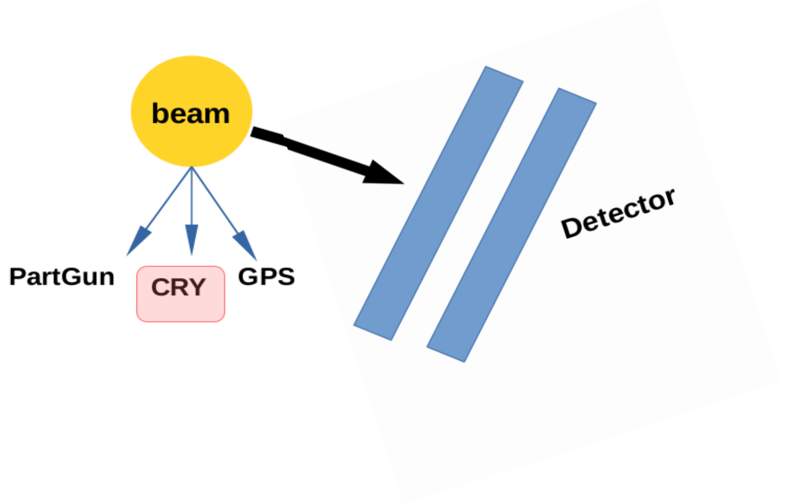


Secondaries flux at sea level

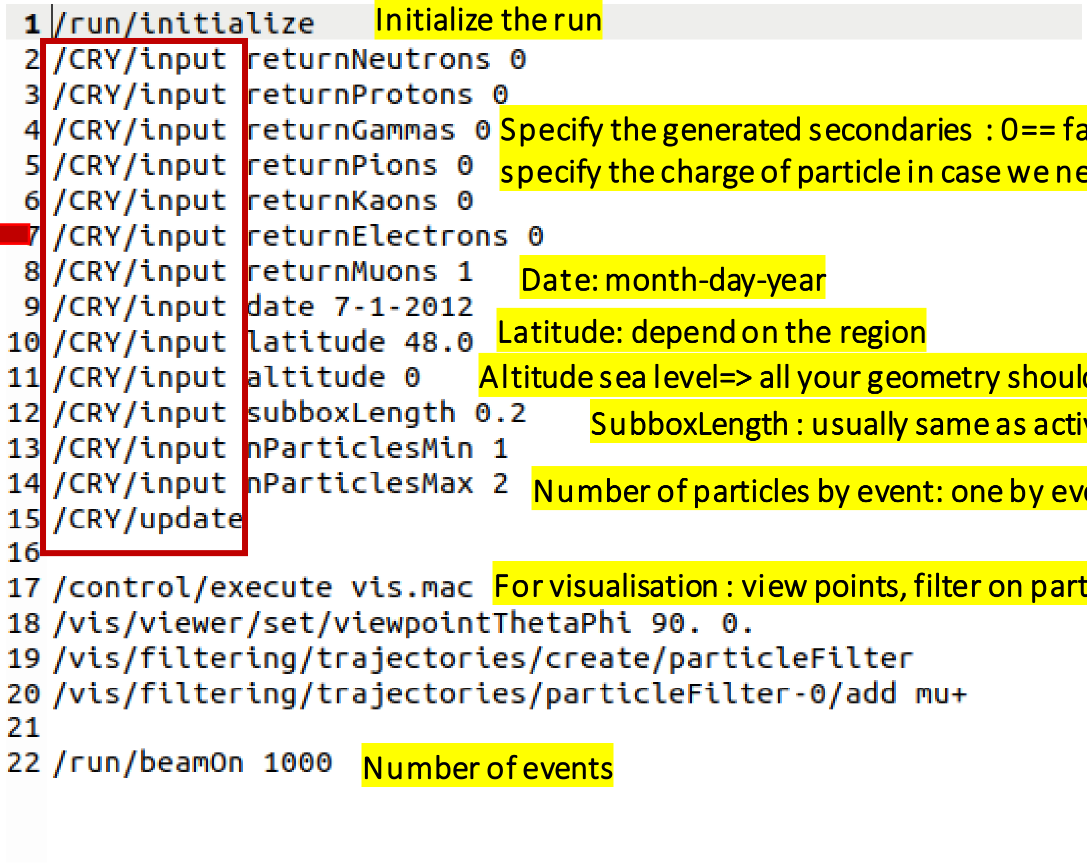
(*)Monte Carlo N-Particle eXtended: is a widely used computer code for simulating the transport of particles, such as neutrons, photons, electrons, and other charged particles, through various materials and complex geometries.

Generator : PrimaryGeneratorAction

- <u>PartGun:</u>
  - G4ParticleGun is a generator provided by Geant4.
  - This class generates primary particle(s) with a given momentum and position.
- <u>GPS:</u>
  - The G4GeneralParticleSource (GPS) is part of the Geant4 toolkit for Monte-Carlo, high-energy particle transport, GPS allows the user to control the following characteristics of primary particles:
    - Spatial sampling (2D or 3D surfaces such as discs, spheres, and boxes)
    - Angular distribution (unidirectional, isotropic, cosine-law, beam or arbitrary..)
    - Spectrum (linear, exponential, power-law, Gaussian,..)
- <u>CRY :</u>
  - Real flux generator (we need to link it to our Geant4 code):
    - Shape: flat surface
    - Energy: real spectrum of energy
    - Momentum direction: zenith angle [0°,90°] and flat azimuthal angle [0°,180°]
    - Created geometry should be all the time in the negative Z direction
- <u>Other generator:</u> EcoMug and CORSIKA



Generator : cmd.file



```
/run/initialize
/CRY/input returnNeutrons 0
/CRY/input returnProtons 0
/CRY/input returnGammas 0
/CRY/input returnPions 0
/CRY/input returnKaons 0
/CRY/input returnElectrons 0
/CRY/input returnMuons 1
/CRY/input date 7-1-2012
/CRY/input latitude 48.0
/CRY/input altitude 0
/CRY/input subboxLength 0.2
/CRY/input nParticlesMin 1
/CRY/input nParticlesMax 2
/CRY/update

/control/execute vis.mac
/vis/viewer/set/viewpointThetaPhi 90. 0.
/vis/filtering/trajectories/create/particleFilter
/vis/filtering/trajectories/particleFilter-0/add mu+

/run/beamOn 1000
```

Initialize the run

Specify the generated secondaries : 0== false && 1== true, in this case only muon are generated (we can specify the charge of particle in case we need only positive or only negative)

Defined already in PrimaryGeneratorMessenger.cc

Date: month-day-year

Latitude: depend on the region

Altitude sea level=> all your geometry should be below zero (Z<0)

SubboxLength : usually same as active area of detector

Number of particles by event: one by event

For visualisation : view points, filter on particle : in this case we see only muon+ trajectories

Number of events