Geometries for Detector Construction in Geant4

Raman Sehgal (BARC)

Things to be discussed

1) Steps involved to create the detector geometries

2) Some of the complex geometries available in Geant4

3) Discussion of “Materials” in brief.

4) Geometry hierarchy in a detector setup.

5) How to import / export the geometry

6) Use of GDML

5) How to read CAD geometries.

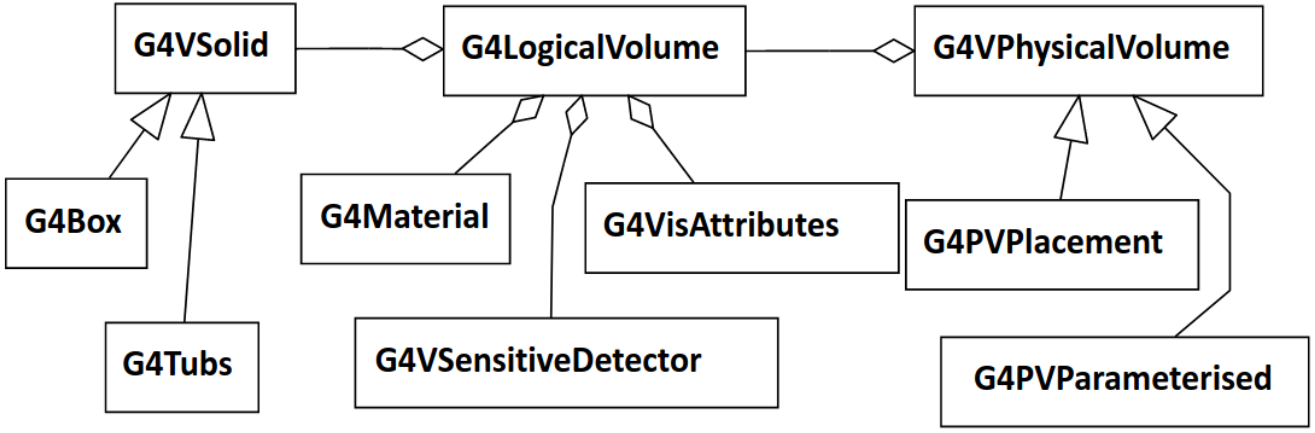
Software architecture of Detector geometry

Basically consist of three layers

1) Solid (Shape) : G4VSolid : Defines the shape and size of the geometry

2) Logical Volume : G4LogicalVolume : material, sensitivity, visualization attributes, physical placement of daughter volumes etc.

3) Physically place volume : G4VPhysicalVolume : defines position, rotatation, and mother volume



Various Shapes available in Geant4

CSG (Constructed Solid Geometry) solids

- G4Box, G4Tubs, G4Cons, G4Trd, G4Para, G4Trap, G4Torus, G4CutTubs, G4Orb, G4Sphere

Specific solids (CSG like)

- G4Polycone, G4Polyhedra, G4Hype, G4Ellipsoid, G4EllipticalTube, G4Tet, G4EllipticalCone, G4Hype, G4GenericPolycone, G4GenericTrap, G4Paraboloid

Tessellated solids

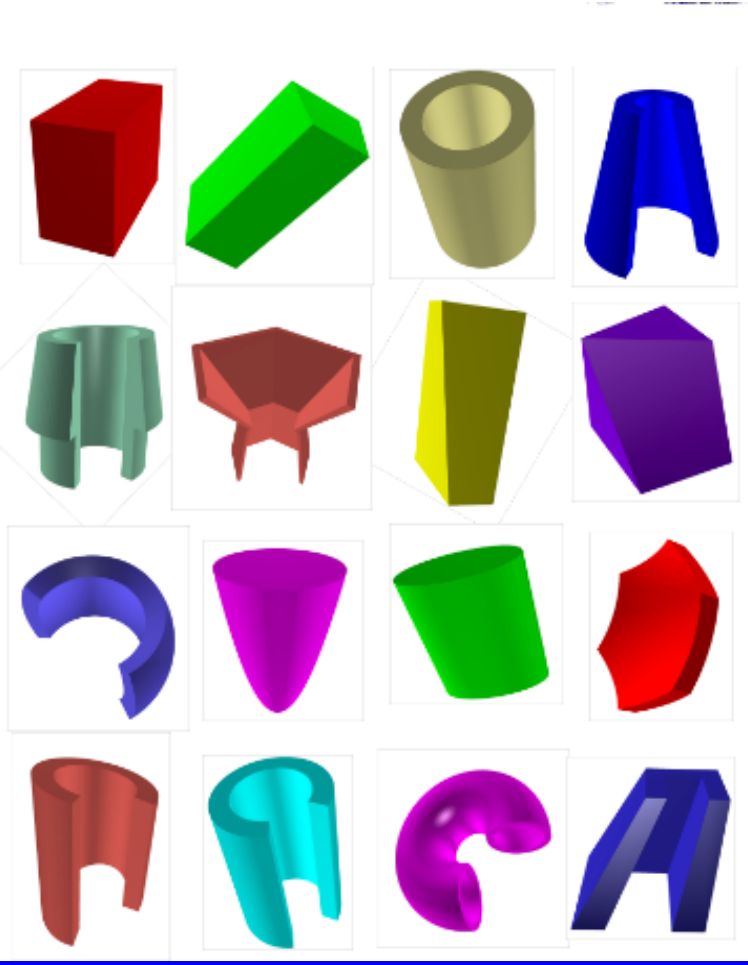
- G4TessellatedSolid, G4ExtrudedSolid

Boolean & scaled solids

- G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid, G4MultiUnion, G4ScaledSolid

Twisted shapes

- G4TwistedBox, G4TwistedTrap, G4TwistedTrd, G4TwistedTubs



Concept of Half lengths

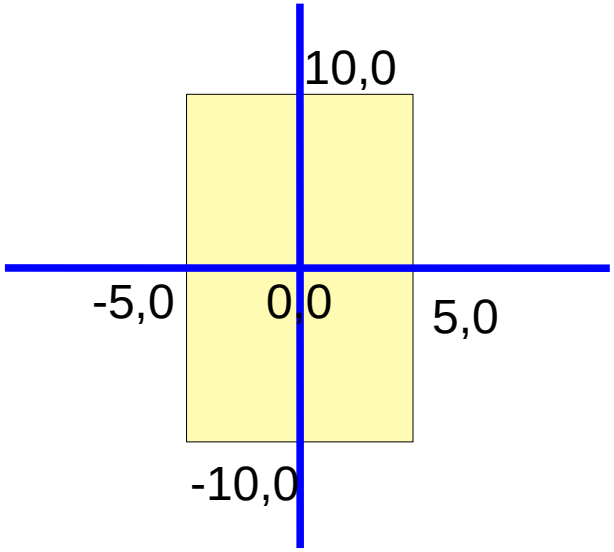
Geant4 geometry works on the concept of half lengths

Suppose we want to create the box of

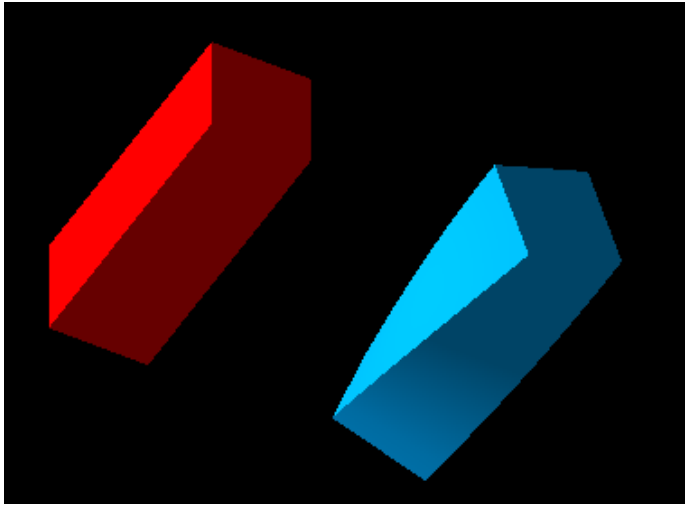
[10 cm X 20 cm X 30 cm] : Required dimension

[5 cm X 10 cm X 15 cm] : Specified Half length

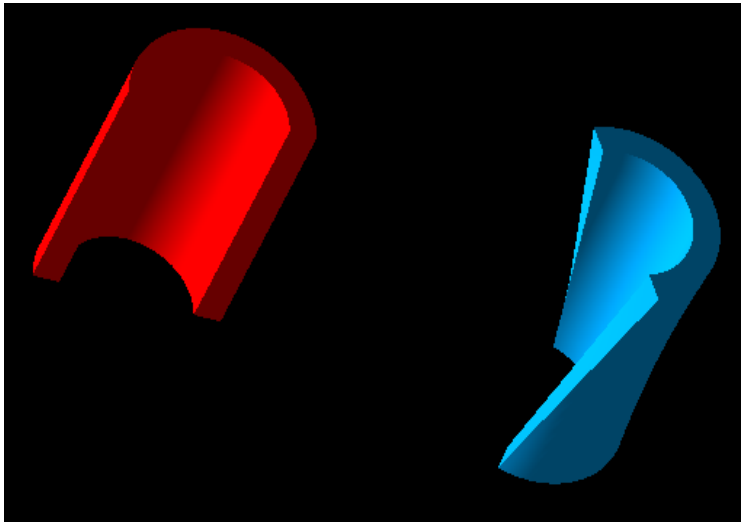
Same concept is applicable to all the geometries
Cone, Tube, etc.



Twisted Box and Twisted Tube



G4Box(“Box”,halfX,halfY,halfZ)
G4TwistedBox(“TwistedBox”,twist,halfX,halfY,halfZ)



G4Tubs(“Tube”,rmin,rmax,dz,sphi, dphi)
G4TwistedTubs(“TwistedTube”,twist,rmin,rmax,dz,dphi)

Cone and Polycone

Worth discussing as they are used in frequently in experiment setup.

Cone is same as tube but having different lower and upper radius.

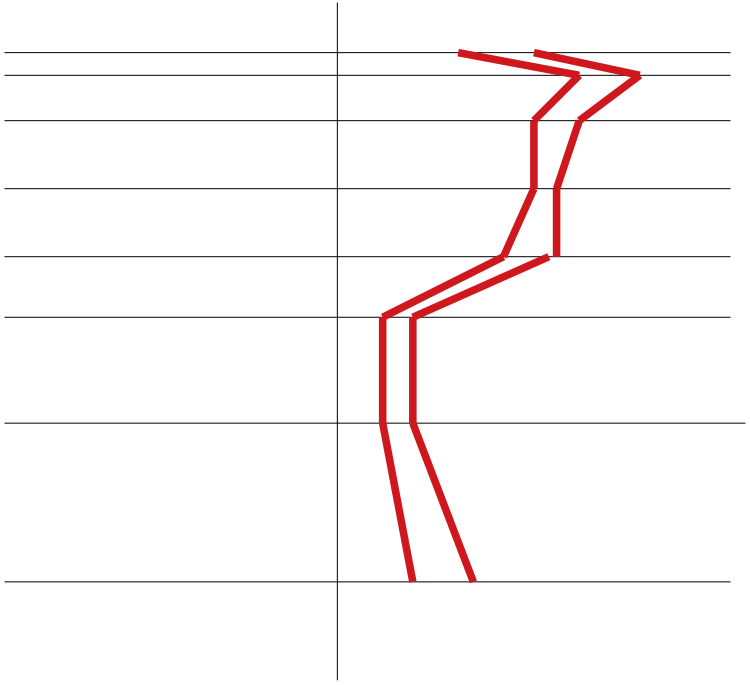
Polycone is something like connecting various cones and tubes one after the another.

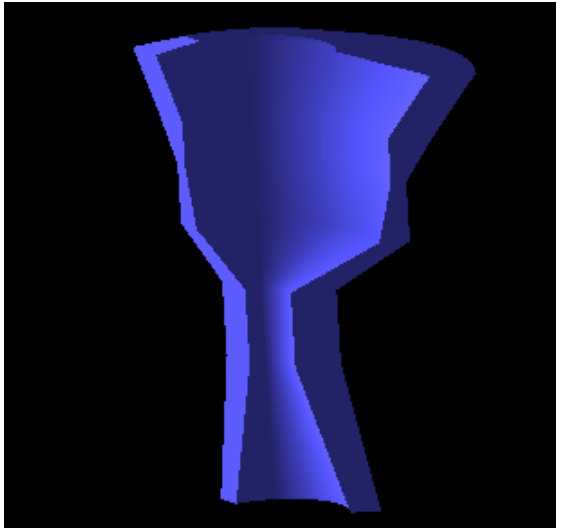
Different constructors exist

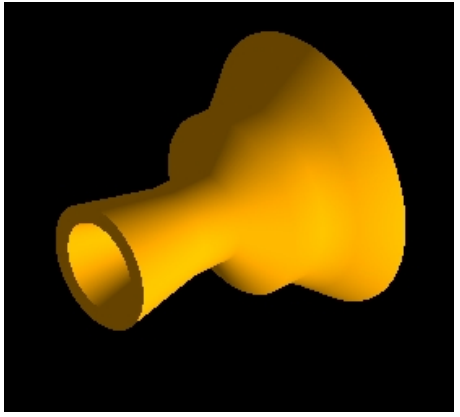
```
G4Polycone("MyPolycone", sPhi, dPhi, numZ, z,  rmin,  rmax);

double z[8]       = {-10., 0., 5., 8., 12., 15., 19, 21};
double rmin[8]  = {5., 2., 2., 8., 9, 9, 12., 6};
double rmax[8] = {7., 5., 5., 10., 10, 12, 15, 8};
G4Polycone("LeadBlock",0.,  2*M_PI,  8,  z,  rmin,  rmax);
```

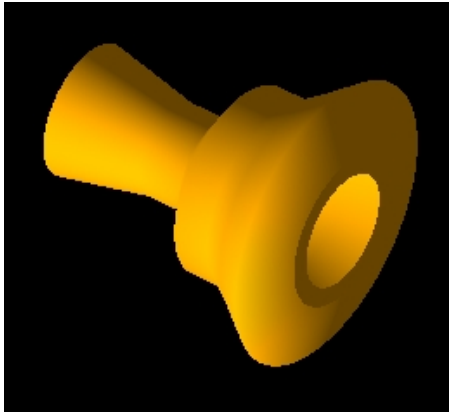
Polycone

```
double z[8]    = {-10., 0., 5., 8., 12., 15., 19, 21};
double rmin[8] = {5., 2., 2., 8.,   9,   9, 12., 6};
double rmax[8] = {7., 5., 5., 10., 10, 12, 15, 8};
G4Polycone("LeadBlock",0.,  2*M_PI,  8,  z,  rmin,  rmax);
```









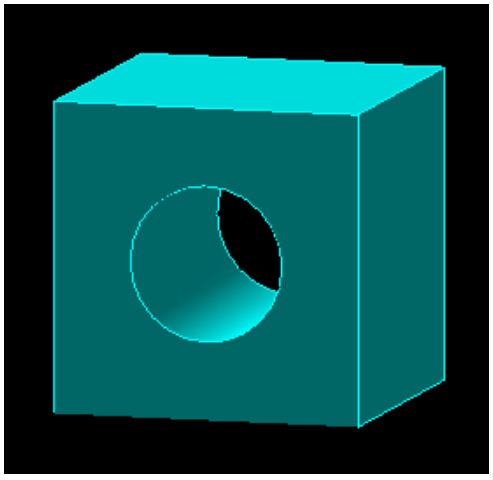
Boolean Operation

G4SubtractionSolid :
Subtraction of one shape from another.

```
G4SubtractionSolid(  const G4String&pName,
                     G4VSolid* pSolidA ,
                     G4VSolid* pSolidB   ) ;
```

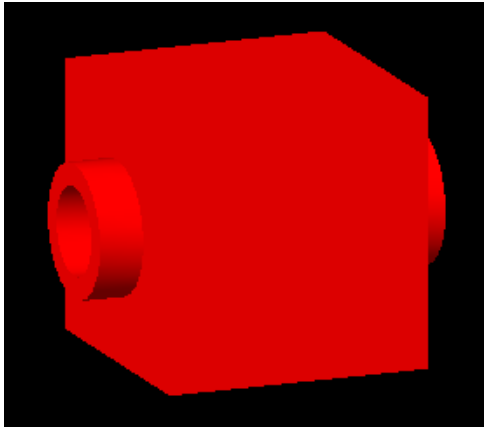
```
G4Box boxA("boxA",3*m,3*m,3*m);
G4Orb orb("orbB",4*m);
```

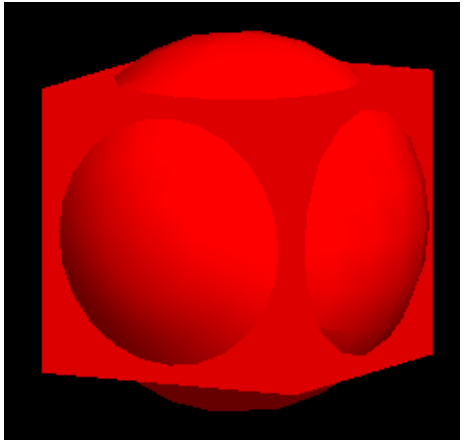
```
G4SubtractionSolid
subtracted("subtracted_boxes",&boxA,&orb);
```



G4UnionSolid:
Union of two shapes.

```
G4UnionSolid(  const G4String&pName,
               G4VSolid* pSolidA ,
               G4VSolid* pSolidB   ) ;
```





Defining Materials

Material can be define in two ways :

1) Using the exising NIST database provided by Geant4
   -- Contains a lot of material as elements, isotopes and compound.

   -- Need an object of NistManger class
      -- G4NistManager *nist = G4NistManager::Instance();
      -- G4Material *world_mat = nist->FindOrBuildMaterial("G4_AIR");
      (G4_Pb, G4_Al, G4_Mg, G4_Na .. etc.)
      (G4_BAKELLITE, G4_ANTHRACENE etc..)

2) Making your own material that can be defined using the various classes available
   -- Isotope : G4Isotope
   -- Element : G4Element
   -- Molecules : G4Material
   -- Compound and Mixture : G4Material

Defining Materials

G4Isotope and G4Element class defines the properties of the atom.

-- Atomic Number, number of nucleons, cross-section per atom

G4Material on the other hand defines the macroscopic properties

-- temperature, pressure, density

-- absorption length, radiation length etc.

Defining Materials

Let's start with a single-element material:

```
G4double density = 4.506*g/cm3;
G4double a = 47.867*g/mole;
G4Material* ti = new G4Material("pureTitanium", z=22, a, density);
```

Creating Elements

```
G4double a = 1.01*g/mole;
G4Element* elH = new G4Element("Hydrogen", "H", z=1, a);
a = 16.00*g/mole;
G4Element* elO =  new G4Element("Oxygen", "O", z=8, a);
```

Finally creating material from elements

```
G4double density = 1.0*g/cm3;
G4int ncomp = 2;
G4Material* H2O = new G4Material("Water", density, ncomp);
G4int nAtoms;
H2O->AddElement(elH, nAtoms=2);
H2O->AddElement(elO, nAtoms=1);
```

Getting Water from NISTManager object

```
G4Material *world_mat =
nist->FindOrBuildMaterial("G4_WATER");
```

Defining Compund Materials

Composition of compound materials

```
G4Element* elementH = ...;
G4Material* Air = ...;
G4Material* H2O = ...;

density = 1.200*g/cm3;

G4Material* myMaterial = new G4Material("MyNewMaterial",density,ncomponents=3);

MyMaterial->AddMaterial(H20,35*perCent);
myMaterial->AddMaterial(Air, 25*perCent);
myMaterial->AddElement(elementH, 40*perCent);
```

Creating Logical Volume and it Physical placement

```
G4LogicalVolume(G4VSolid* pSolid,
                G4Material* pMaterial,
                const G4String& name);
```

```
G4PVPlacement(G4RotationMatrix *pRot,
              const G4ThreeVector &tlate,
              G4LogicalVolume *pLogical,
              const G4String& pName,
              G4LogicalVolume *pMotherLogical,
              G4bool pMany,
              G4int  pCopyNo,
              G4bool pSurfChk=false);
```

```
G4Box box("test", 5*cm, 5*cm, 5*cm);
G4Material Al;
G4LogicalVolume *logicalBox = new
G4LogicalVolume(box, Al, "LogicalBox");
```

```
new G4PVPlacement(0,
                  G4ThreeVector(),
                  logicalBox,
                  "PhysicalVolume",
                   motherLogicalVol,
                   false,
                   0,
                   true);
```

Understanding the Geometry Hierarchy

Raw shapes never forms the part of geometry hierarchy.

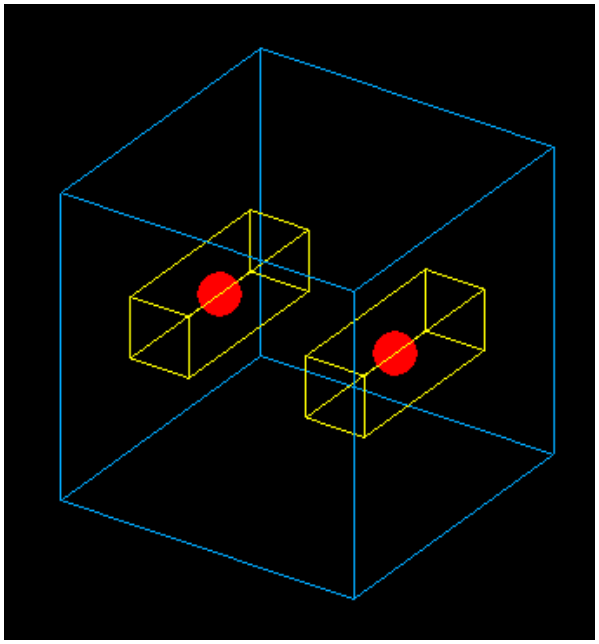
Physical placement is alway done for a logical volume.

The geometry hierarchy consist of Mother-Daughter relationship.

One Logical volume contains other Physical Volume daughter volumes

The mother logical volume forms the local coordinate system for all its daughter volumes.

If a mother volume is placed more than once, all its daughter volumes will be there in all physical volumes



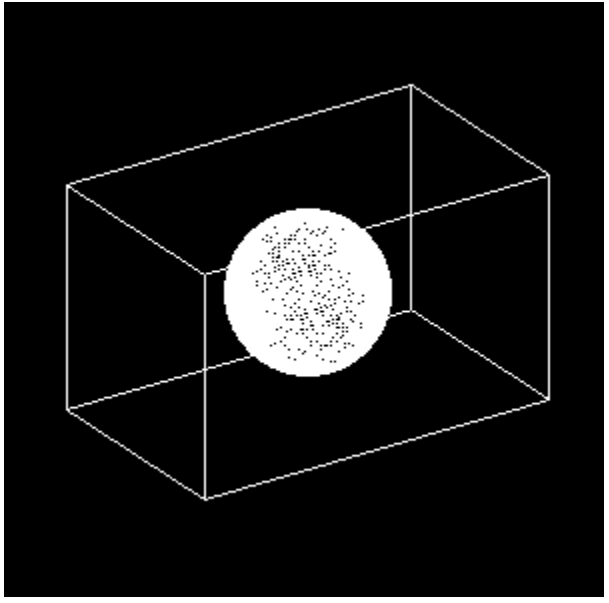
Only Exception : World Volume
Its a unique physical volume which contains all the other volume of your detector setup

World volume also forms the global coordinate system.

Understanding Physical placement in the Geometry Hierarchy

Both Box and Orb are placed with respect to **world reference frame**

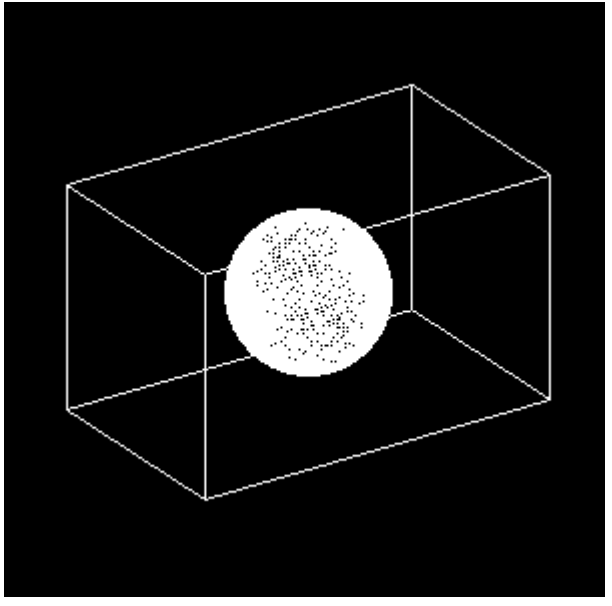
```
"/vis/list" to see available colours.
Checking overlaps for volume PhysicalLead ... OK!
Checking overlaps for volume PhysicalOrb ...
-------- WWWW ------- G4Exception-START -------- WWWW --
*** G4Exception : GeomVol1002
      issued by : G4PVPlacement::CheckOverlaps()
Overlap with volume already placed !
          Overlap is detected for volume PhysicalOrb:0
          with PhysicalLead:0 volume's
          local point (78.0177,-62.4168,4.16952), overla
pping by at least: 7.19823 cm
```



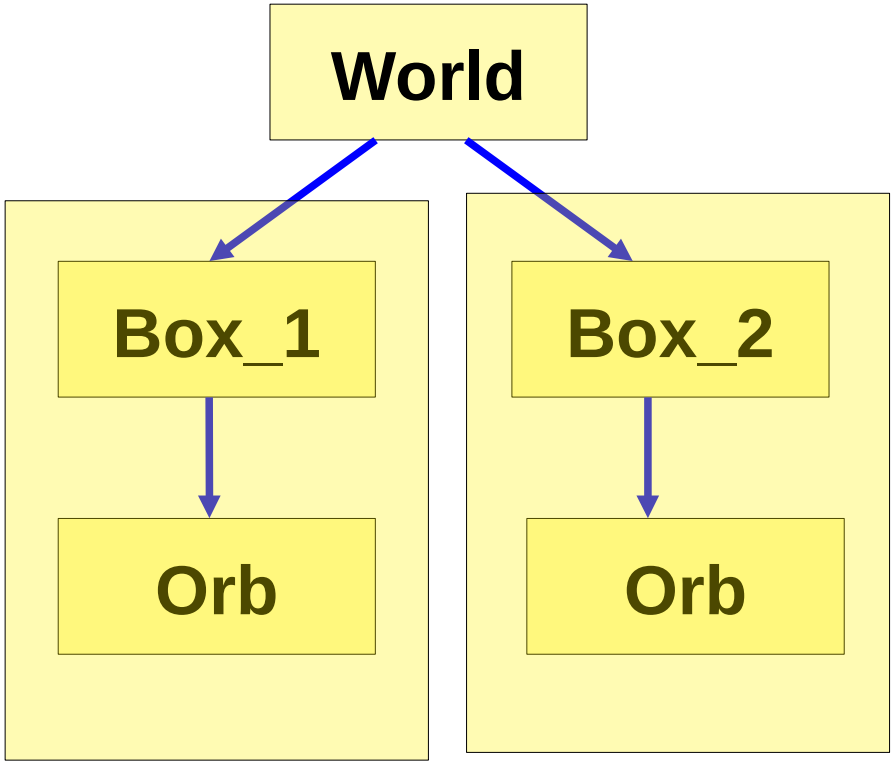
Box is placed with respect to **world reference frame**

Orb is place with respect to **box reference frame**

```
Some /vis commands (optionally) take a string to specify
 colour.
"/vis/list" to see available colours.
Checking overlaps for volume PhysicalLead ... OK!
Checking overlaps for volume PhysicalOrb ... OK!
G4GDML: Writing 'test.gdml'...
G4GDML: Writing definitions...
G4GDML: Writing materials...
G4GDML: Writing solids...
G4GDML: Writing structure...
G4GDML: Writing setup...
G4GDML: Writing surfaces...
G4GDML: Writing 'test.gdml' done !
```



Geometry Hierarchy Tree



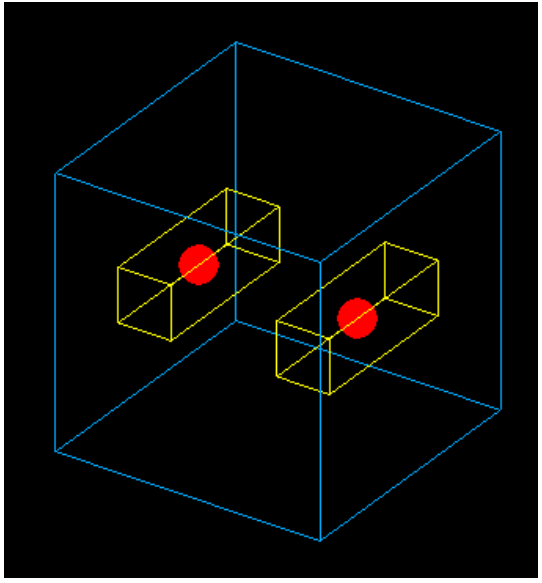
Complete hierarchy contains 5 shapes

But you had created only 3 shapes.

World, Box and Orb

Mother box contains Orb daughter

Multiple placement of mother box contains all the daugter volumes



Exporting Geometry

- If the same geometry setup needs to be used in multiple simulations or needs to be used by different people
- Geant supported geometry export format
- GDML (Graphics Description Markup language)
- A portable format, similar to XML.
- Can be read by standalone application
- Various XML reading libraries are present.
- Xerces-C is used by Geant4

Moving ahead

- Till now, whatever we understood

C++ {

  - Defining shapes, logical volume and their physical placement
  - Define material and attaching them to the shapes to convert them to logical volumes

Alternative way to achieve the same thing.

Instead of doing a the detector construction at compile that (as done above),
Idea is to generate it at run time.
Reading a text file (XML)
Benefits : Allows to quickly recreate the full detector construction with very few lines of code

GDML : Graphics Description Markup Language

- XML formatted text file.
- It implements hierarcy of volumes in a detector setup as the tree of geometries.
- Allows to define the material, and place the volumes.
- Makes the detector construction portable, and independent of the remaining simulation code.

Benefits of using GDML

- Language independent
- Containing user defined tags.
- Can be processed by any library that can process XML.
- Provides hierarchal structure, and mother daughter relationship can be easily maintained.
- Hierarchical structure make its suitable for object oriented programming.

Overview of GDML : Various Components

The flow of a default GDML file follows:

1) Definitions

2) Materials

3) Solids

4) Structures

5) Setup

```
−<gdml xsi:noNamespaceSchemaLocation="http://service-
spi.web.cern.ch/service-spi/app/releases/GDML/schema
/gdml.xsd">
   <define/>
 +<materials></materials>
 +<solids></solids>
 +<structure></structure>
 +<setup name="Default" version="1.0"></setup>
 </gdml>
```

NOTE : Your internet browser is a very useful tool to have a look at the XML file.

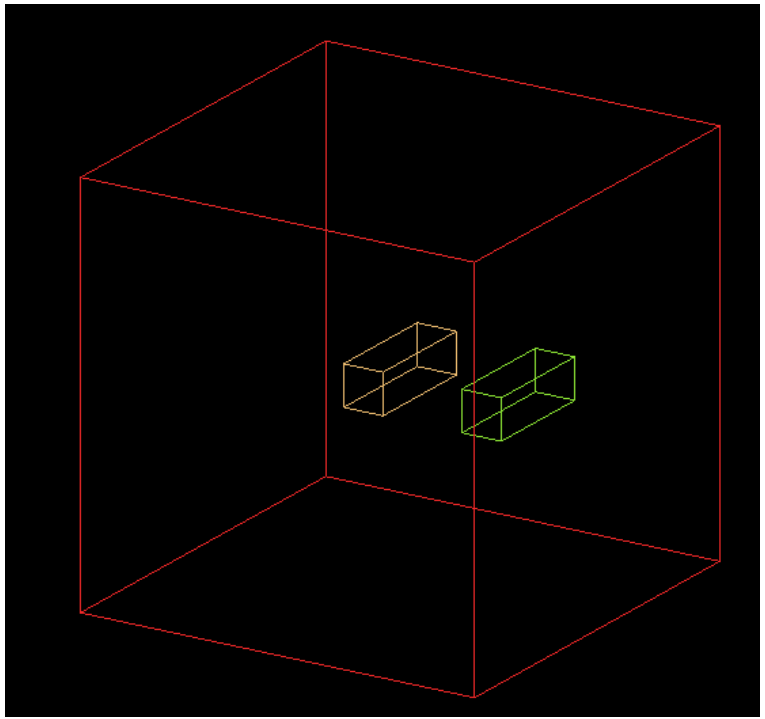
Various GDML Solids

GDML supports all the solids provided by Geant4

Box
Orb
Sphere
Cone
Tube
Parboloid
Ellipsoid
Polyhedro
Polycone
Torus

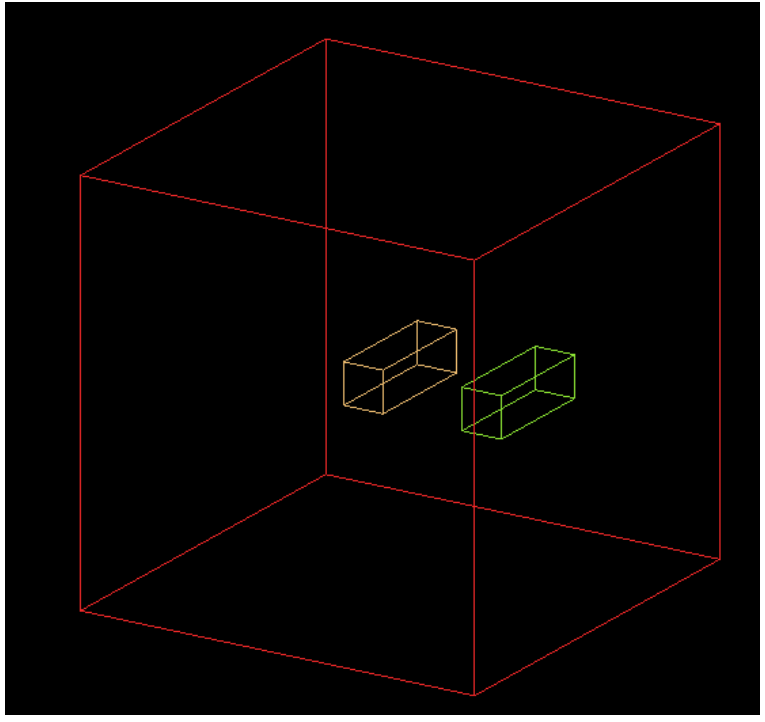
Trapezoid
Cut Tube
Segment of a Tube
Twisted tube
Extruded Solids
Tesellated Solids
Tetrahedrons
Twisted Generic Trapezoid
Twisted Box

All the pieces of GDML



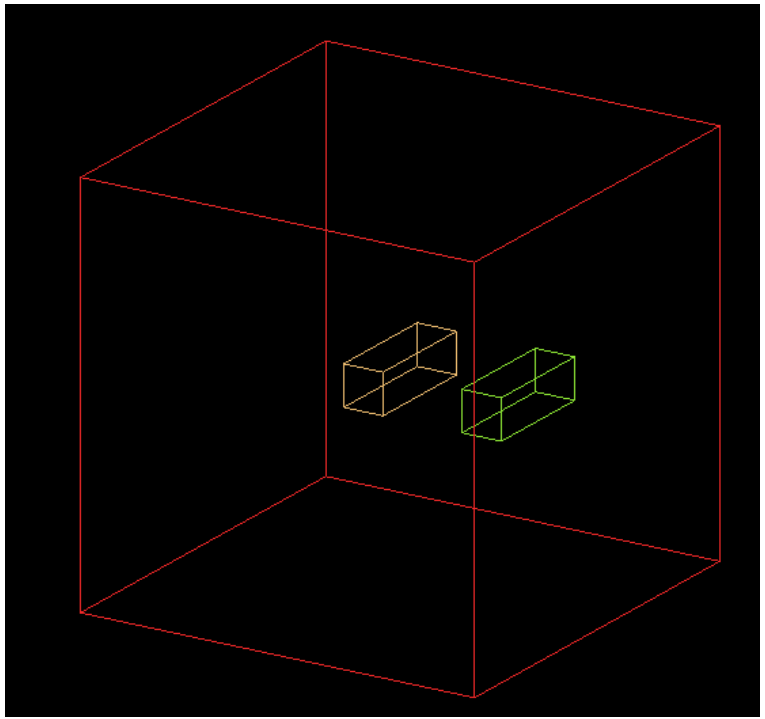
All the pieces of GDML

<solids> tag of GDML



```
−<solids>
    <box lunit="mm" name="LeadBlock" x="100" y="100" z="300"/>
    <box lunit="mm" name="AluminiiBlock" x="100" y="100" z="300"/>
    <box lunit="mm" name="WorldBlock" x="1000" y="1000" z="1000"/>
  </solids>
```

<materials> tag of GDML



```
−<materials>
  −<isotope N="204" Z="82" name="Pb204">
      <atom unit="g/mole" value="203.973"/>
    </isotope>
  −<isotope N="206" Z="82" name="Pb206">
      <atom unit="g/mole" value="205.974"/>
    </isotope>
  −<isotope N="207" Z="82" name="Pb207">
      <atom unit="g/mole" value="206.976"/>
    </isotope>
  −<isotope N="208" Z="82" name="Pb208">
      <atom unit="g/mole" value="207.977"/>
    </isotope>
  −<element name="Pb">
      <fraction n="0.014" ref="Pb204"/>
      <fraction n="0.241" ref="Pb206"/>
      <fraction n="0.221" ref="Pb207"/>
      <fraction n="0.524" ref="Pb208"/>
    </element>
  −<material name="G4_Pb" state="solid">
      <T unit="K" value="293.15"/>
      <MEE unit="eV" value="823"/>
      <D unit="g/cm3" value="11.35"/>
      <fraction n="1" ref="Pb"/>
    </material>
```

<structure> tag of GDML

Structure tag actually defines how different components of detector setup are arranged.
(Actually shows the mother-daughter relatioship)

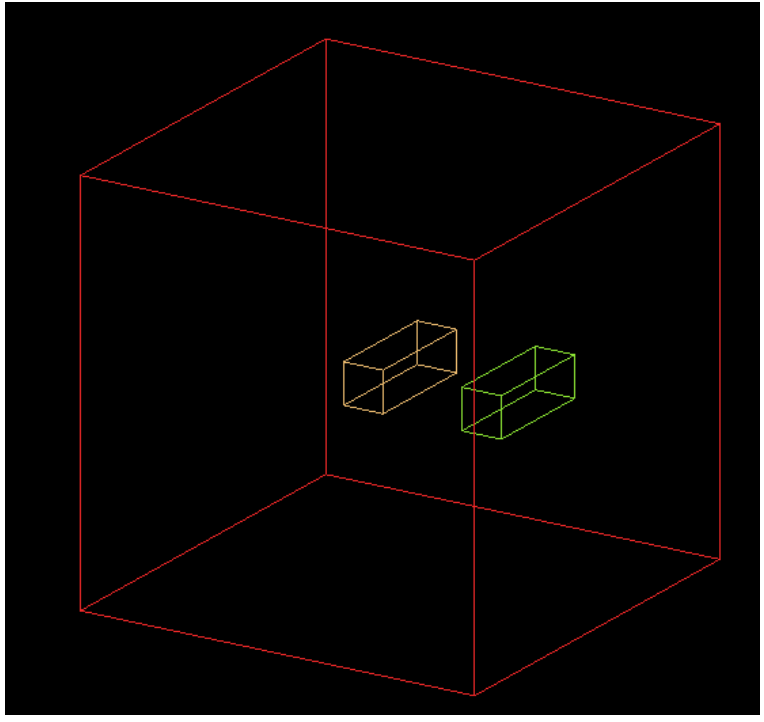
Both logical and physical volumes are defined in one structure

It show the hierarchy of detector components.

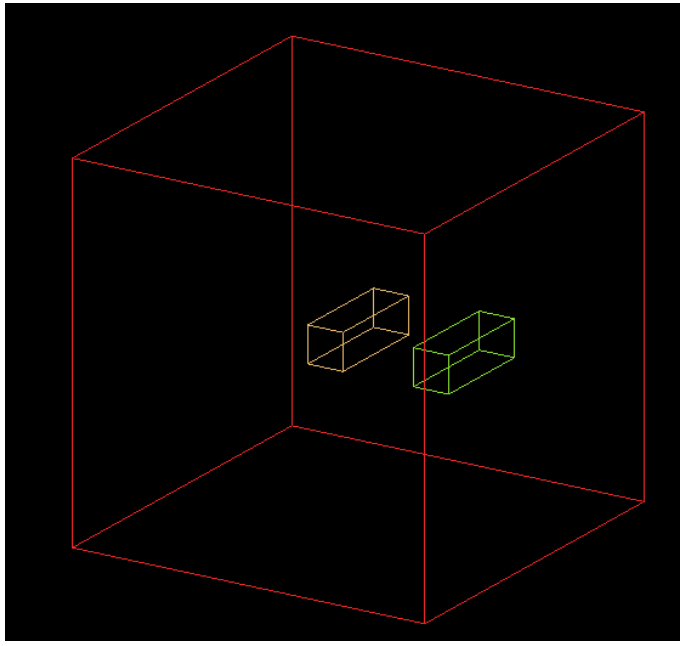
It consist of sequence of volumes tags, that define your logical volumes.

Each volume tag, keeps a pointer to the associated solid and the material.

Structure actually correponds to the your physical detector



Structures cont...



```
−<structure>
  −<volume name="LogicalLeadBlock">
      <materialref ref="G4_Pb"/>
      <solidref ref="LeadBlock"/>
    </volume>
  −<volume name="LogicalAlBlock">
      <materialref ref="G4_Al"/>
      <solidref ref="AluminiiBlock"/>
    </volume>
  −<volume name="World">
      <materialref ref="G4_Galactic"/>
      <solidref ref="WorldBlock"/>
    −<physvol name="Physical_PB_Block">
        <volumeref ref="LogicalLeadBlock"/>
      </physvol>
    −<physvol name="Physical_Al_Block">
        <volumeref ref="LogicalAlBlock"/>
        <position name="Physical_Al_Block_pos" unit="mm" x="300" y="0" z="0"/>
      </physvol>
    </volume>
  </structure>
```

Finally the <setup> tab

Setup contains the pointer to you world volume.

While creating a detector setup using gdml as an input file, we need to return a pointer to world volume from the **“Construct”** function of DetectorConstruction file.

It is possible to define multiple geometry setup, and chosing different volumes as world volume.

Moreover, we can actually split this geometry description in multiple file, which allows more granularity, and ease of maintainance.

```
−<setup name="Default" version="1.0">
    <world ref="World"/>
  </setup>
```

Exporting/ Importing GDML into Geant4

The GDML files can be imported directly into Geant4, in the detector construction class

Required class : **G4GDMLParser : (#include <G4GDMLParser.hh>)**

As usual this class contains various functions :
We will focus on Write and Read

An object of **“G4GDMLParseR”** class is required.

G4GDMLParser myGDMLParser;

To export a full detector construction written in C++
myGDMLParser.Write(“geom.gdml”,pointerToPhyWorld);

To import a full detector setup, just do
myGDMLParser.Read(“geom.gdml);

Finally return the pointer to physical world volume to Geant.
return MyGDMLParser.GetWorldVolume()

What about CAD geometries

What if I want to include the CAD drawing of my detector setup available me, and I would like to use that directly in Geant instead of writing C++ code for the same.

Practically possible, but there are few problems.

How do we go about ??

We need three things : 1) Solid 2) Material 3) Placement

Difference in the tolerances required for machining and tolerance required to do particle transport.

Need a common format understandable by both (CAD and GEANT)

Some output format provided by various CAD programs are STEP and STL

Problem with these is that some of these does not contain material information.

Do not export the individual primitive geometries and there parameter, rather export the CAD drawing as a sequence of triangles and their normal, know as the tesellations of the geometries.

Solution for CAD Geometries

As we have discussed, Geant4 can import GDML files

Converting STEP/STL file to GDML is a potential solution.

Needs some third party tools

Various converters are available like:

- FastRad
- FreeCAD
- CADMesh

None of them is perfect.

But still they can be used to import CAD geometries, along with some manual intervention.

Acceptable from visualization point of view, but may contain some mismatches, which may results in some stuck tracks in Geant4 navigation.

```
solid Default
  facet normal 0.000000e+00 0.000000e+00 1.000000e+00
    outer loop
      vertex -1.501741e+01 5.467951e+00 2.488822e+01
      vertex -1.501741e+01 3.843249e+00 2.488822e+01
      vertex -1.477975e+01 5.467951e+00 2.488822e+01
    endloop
  endfacet
  facet normal 0.000000e+00 0.000000e+00 1.000000e+00
    outer loop
      vertex -1.477975e+01 5.467951e+00 2.488822e+01
      vertex -1.501741e+01 3.843249e+00 2.488822e+01
      vertex -1.477975e+01 3.843249e+00 2.488822e+01
    endloop
  endfacet
  facet normal 0.000000e+00 0.000000e+00 1.000000e+00
    outer loop
      vertex -6.764211e+00 1.188298e-01 2.488822e+01
      vertex -6.764211e+00 1.782447e+00 2.488822e+01
      vertex -7.001871e+00 1.188298e-01 2.488822e+01
    endloop
  endfacet
endsolid Default
```

Using CADMesh to import STL file

CADMesh is a Header only package.

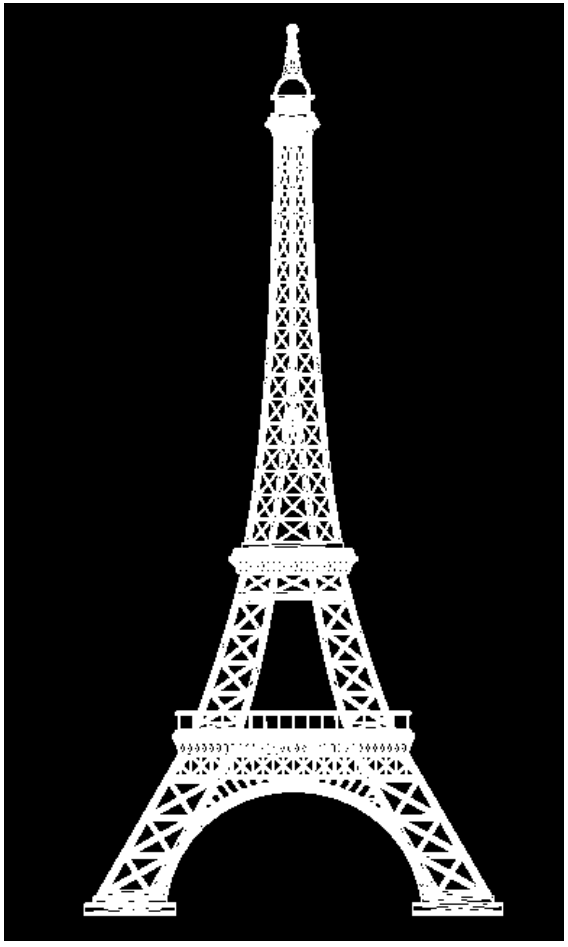
Can be downloaded from
https://github.com/christopherpoole/cadmesh

Copy the CADMesh.hh file in the include directory of you simulation code.

Whole mesh can be imported using following lines of code.

```
auto mesh = CADMesh::TessellatedMesh::FromSTL("eiffel.stl");
std::vector<G4VSolid*> solids = mesh->GetSolids();
```

Since the .stl does not contain the material information, Hence, one has to create the logical volume on his own.



Thank You for your attention

Any doubts ???