Geant4: A Simulation toolkit

O. Stézowski and A. Cazes

With many thanks to the Geant4 community !!!!

The roadmap of the week

W1: installation / running a G4 application

Why?

W2: Primary generator, GPS, physics list

W3: Geometries !

W4: Sensitive detectors / user's actions

Do the one
you want to practice on

NOW, HOW does it really work ?

The user's application

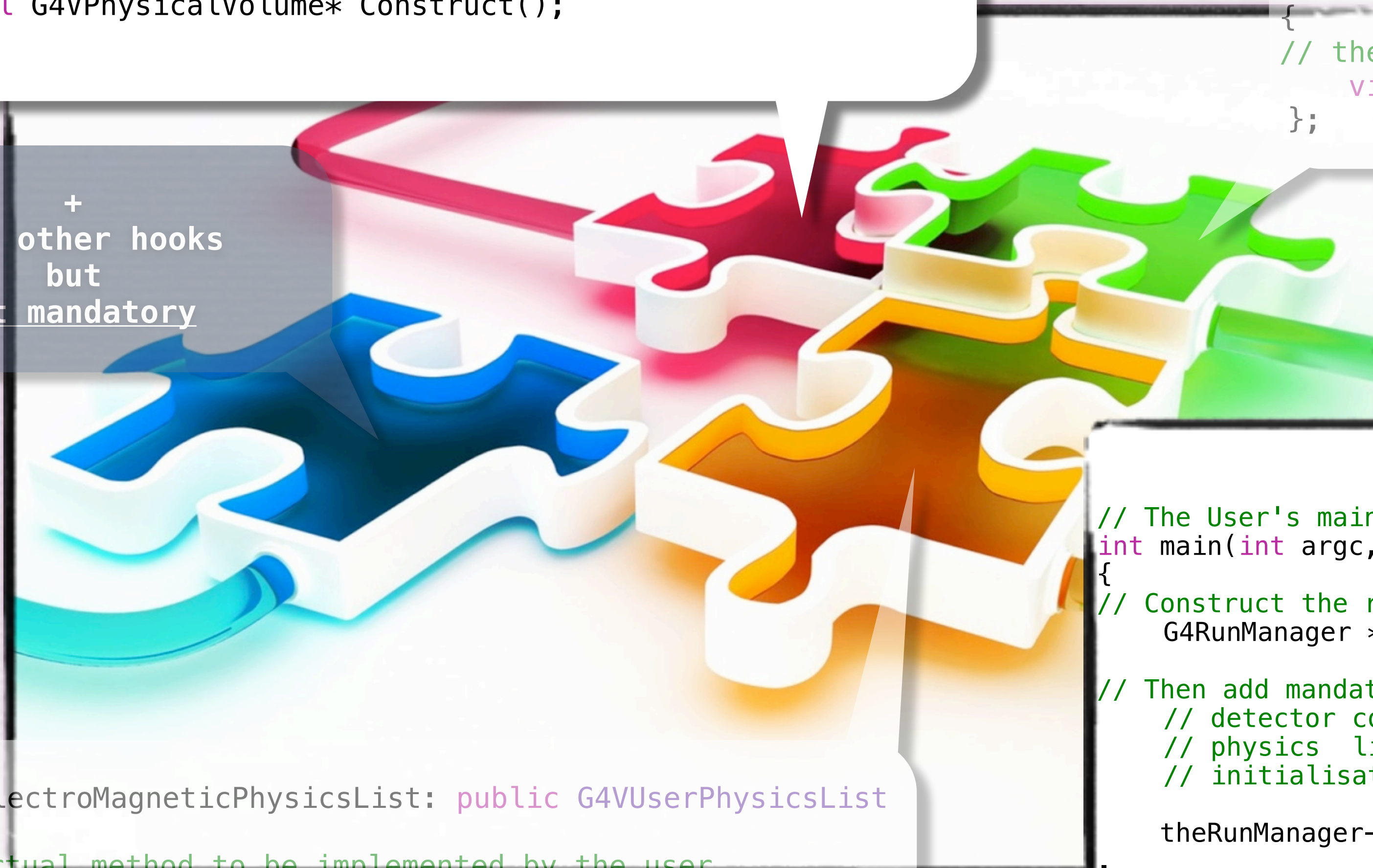
Building an application requires to put together 3 mandatory bricks*

the detector construction - the description of the physics - the primary generator

```
class ARedSphereConstruction : public G4VUserDetectorConstruction
{
// the virtual method to be implemented by the user
    virtual G4VPhysicalVolume* Construct();
};
```

```
class AGammaGun : public G4VUserPrimaryGeneratorAction
{
// the virtual method to be implemented by the user
    virtual void GeneratePrimaries(G4Event* anEvent);
 };
```

+
many other hooks
but
<u>not mandatory</u>



```
class AnElectroMagneticPhysicsList: public G4VUserPhysicsList
{
// the virtual method to be implemented by the user
void ConstructParticle();
// the virtual method to be implemented by the user
void ConstructProcess();
// the virtual method to be implemented by the user
void SetCuts();
};
```

```
// The User's main program to control / run simulations
int main(int argc, char** argv)
{
// Construct the run manager, necessary for G4 kernel to control everything
    G4RunManager *theRunManager = new G4RunManager();

// Then add mandatory initialization G4 classes provided by the USER
    // detector construction
    // physics  list
    // initialisation of the generator

    theRunManager->SetUserInitialization( new ARedSphereConstuction() );
.
.
    return 0;
}
```

W3: Geometries !

- Volumes - general aspects
- Definition of materials
- Definition shapes
- All bricks together
- Exportation / importation

The user's application

A detector geometry is made of a number of volumes

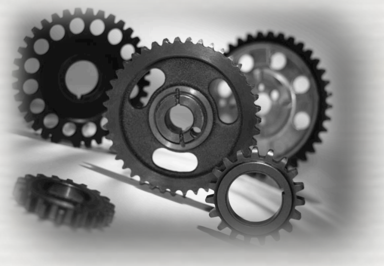
Requirements to write the method **Construct()** i.e. the full setup of the simulation

- Construct all necessary materials
- Define shapes/solids
- Define logical volumes
- Place volumes of your detector geometry
- Associate (magnetic) field to geometry *(optional)*
- Instantiate sensitive detectors/scorers, set them some logical volumes *(optional)*
- Define visualization attributes for the detector elements *(optional)*
- Define regions *(optional)*

Not covered in this lecture

see workshop #4

Not covered in this lecture



Geant4 defines two kind of volume

a **G4LogicalVolume** is used to keep the characteristics of a volume

a **G4VPhysicalVolume** is used to place (translation, rotation) a logical volume with respect to a mother volume.

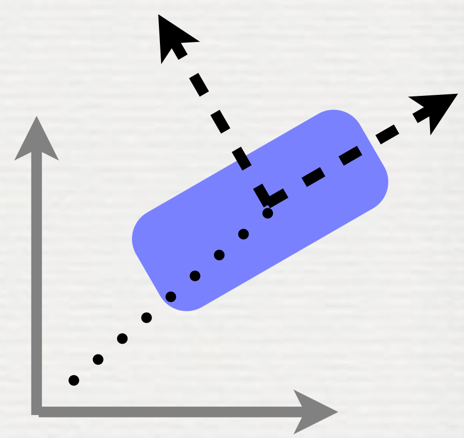
☛ There is a top volume which is called the World Volume !

G4LogicalVolume contains:

G4Material [composition]
G4VSolid [shape]
G4VisAttributes [color]
...
list of physical volumes
...

G4VPhysicalVolume contains:

G4ThreeVector T
G4RotationMatrix R
copy #

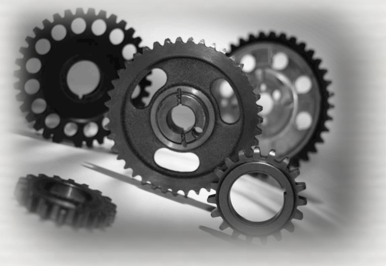


mother referential

The Construct method of **G4VUserDetectorConstruction** returns a **G4VPhysicalVolume**, the world

W3: Geometries !

- Volumes - general aspects
- **Definition of materials**
- Definition shapes
- All bricks together
- Exportation / importation



Construct all necessary materials

Different kinds of materials can be defined:

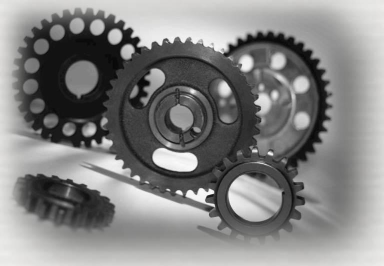
- isotopes ↔ **G4Isotope**
- elements ↔ **G4Element**
- molecules ↔ **G4Material**
- compounds and mixtures ↔ **G4Material**

➥ Attributes associated: temperature, pressure, state, density

- **G4Isotope** and **G4Element** describe *microscopic* properties of the *atoms*:
  ➥ Atomic number, number of nucleons, mass of a mole, shell energies, cross-sections per atoms ...
- **G4Material** describes the *macroscopic* properties of the *matter*:
  ➥temperature, pressure, state, density
  ➥Radiation length, absorption length, etc...

- **G4Material** is the only class used and visible to the toolkit:

➥ it is used by tracking, geometry and physics



Construct all necessary materials

Isotopes can be assembled into ...

G4Isotope (const G4String& name,
G4int z, /* atomic number */
G4int n, /* number of nucleons */
G4double a); /*mass of mole*/

... elements

G4element (const G4String& name,
const G4String& symbol, /*element symbol*/
G4int nIso); /*n. of isotopes*/

```
// Germanium isotopes
G4Isotope* Ge70 = new G4Isotope(name="Ge70", 32, 70, 69.9242*g/mole);
G4Isotope* Ge72 = new G4Isotope(name="Ge72", 32, 72, 71.9221*g/mole);
G4Isotope* Ge73 = new G4Isotope(name="Ge73", 32, 73, 72.9235*g/mole);
G4Isotope* Ge74 = new G4Isotope(name="Ge74", 32, 74, 73.9212*g/mole);
G4Isotope* Ge76 = new G4Isotope(name="Ge76", 32, 76, 75.9214*g/mole);
// germanium defined via its isotopes
G4Element* elGe = new G4Element(name="Germanium",symbol="Ge", 5);
elGe->AddIsotope(Ge70, 0.2123);
elGe->AddIsotope(Ge72, 0.2766);
elGe->AddIsotope(Ge73, 0.0773);
elGe->AddIsotope(Ge74, 0.3594);
elGe->AddIsotope(Ge76, 0.0744);
```

Fraction of atoms per volumes

Construct all necessary materials

... elements into materials ...

single element material

```
density = 2.7*g/cm3;
a = 26.98*g/mole;
G4Material *al = new G4Material(name="Aluminium",z=13.,a,density);
```

Example of materials filled with gas

```
density = 27.0*mg/cm3;
temperature = 325.*kelvin;
pressure = 50.*atmosphere;
G4Material *co2 =
      new G4Material(name="CarbonicGas",density,nel,
                     kStateGas,temperature, pressure);
co2->AddElement(c, natoms = 1);
co2->AddElement(o, natoms = 2);
```

```
atomicNumber = 1.;
massOfMole = 1.008*g/mole;
density = 1.e-25*g/cm3;
temperature = 2.73*kelvin;
pressure = 3.e-18*pascal;
G4Material *vacuum =
      new G4Material(name="interGalactic",
                     atomicNumber,massOfMolde,density,nel,
                     kStateGas,temperature, pressure);
```

absolute vacuum **does not** exist !
Gaz at very low pressure

composition of compound materials

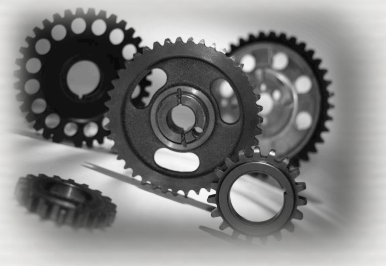
```
G4Element  *c = ...      // carbone element
G4Material *quartz = ... // quartz material
G4Material *water = ...  //  water material

density = 0.200*g/cm3;
nel = 3;
G4Material *aerogel = new G4Material(name="Aerogel",density,nel);
aerogel->AddMaterial(quartz, natoms = 1);
aerogel->AddMaterial(water, natoms = 1);
aerogel->AddElement(c, natoms = 1);
```

A material made of several elements (composition by number of atoms)

```
a=22.99*g/mole;
G4Element *na = new G4Element(name="Sodium",symbol="Na",z=11.,a);
a=126.90477*g/mole;
G4Element *i = new G4Element(name="Iodine",symbol="I",z=53.,a);

density = 3.67*g/cm3;
nel = 2;
G4Material *mix = new G4Material(name="NaI",density,nel);
mix->AddElement(na, natoms = 1);
mix->AddElement(i, natoms = 1);
```

A material made of several elements (composition by of mass)

```
a=14.01*g/mole;
G4Element *n = new G4Element(name="Nitrogen",symbol="N",z=7.,a);
a=16.00*g/mole
G4Element *o = new G4Element(name="Oxygen",symbol="O",z=8.,a);

density = 1.29*mg/cm3;
nel = 2;
G4Material *air = new G4Material(name="Air",density,nel);
mix->AddElement(n, 0.7);
mix->AddElement(o, 0.3);
```



Construct all necessary materials

Geant4 provides defaults based on the NIST database*

Z	A	m	error (%)	A_{eff}
14	Si	22	22.03453	(22)
		23	23.02552	(21)
		24	24.011546	(21)
		25	25.004107	(11)
		26	25.992330	(3)
		27	26.98670476	(17)
		28	27.9769265327	(20)
		29	28.97649472	(3)
		30	29.97377022	(5)
		31	30.97536327	(7)
		32	31.9741481	(23)
		33	32.978001	(17)
		34	33.978576	(15)
		35	34.984580	(40)
		36	35.98669	(11)
		37	36.99300	(13)
		38	37.99598	(29)
		39	39.00230	(43)
		40	40.00580	(54)
		41	41.01270	(64)
		42	42.01610	(75)

natural isotope compositions
more than 3000 isotope masses

Z	Name	density(g/cm^3)	I(eV)
1	G4_H	8.3748e-05	19.2
2	G4_He	0.000166322	41.8
3	G4_Li	0.534	40
4	G4_Be	1.848	63.7
5	G4_B	2.37	76
6	G4_C	2	81
7	G4_N	0.0011652	82
8	G4_O	0.00133151	95
9	G4_F	0.00158029	115
10	G4_Ne	0.000838505	137
11	G4_Na	0.971	149
12	G4_Mg	1.74	156
13	G4_Al	2.699	166
14	G4_Si	2.33	173
15	G4_P	2.2	173
16	G4_S	2	180
17	G4_Cl	0.00299473	174
18	G4_Ar	0.00166201	188

Many elements defined

Ncomp	Name		density(g/cm^3)	I(eV)
6	G4_A-150_TISSUE		1.127	65.1
	1	0.101327		
	6	0.7755		
	7	0.035057		
	8	0.0523159		
	9	0.017422		
	20	0.018378		
3	G4_ACETONE		0.7899	64.2
	1	0.104122		
	6	0.620405		
	8	0.275473		
2	G4_ACETYLENE		0.0010967	58.2
	1	0.077418		
	6	0.922582		
3	G4_ADENINE		1.35	71.4
	1	0.037294		
	6	0.44443		
	7	0.518276		

Many materials provided

C++

```
G4NistManager* man = G4NistManager::Instance();
G4Material *air = man->FindOrBuildMaterial("G4_AIR");
```

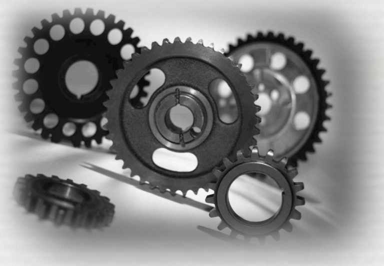
G4

```
/material/nist/printElement
/material/nist/listMaterials
```

* https://www.nist.gov/pml/atomic-spectra-database

W3: Geometries !

- Volumes - general aspects
- Definition of materials
- **Definition shapes**
- All bricks together
- Exportation / importation



G4VSolid to define the shape

All kind of shapes in G4 inherits from **G4VSolid**

It does not include the material

There are different ways to define a 3D shape

- CSG (Constructed Solid Geometry) solids

  G4Box, **G4Tubs**, **G4Cons**, **G4Trd**, ...

- Specific solids (CSG like)

  G4Polycone, **G4Polyhedra**, **G4Hype**, ...

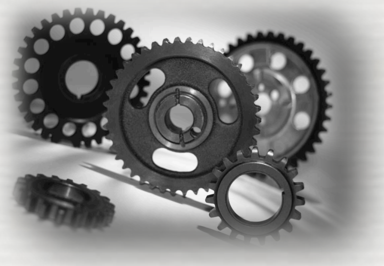
- BREP (Boundary REPresented) solids

  G4BREPSolidPolycone, **G4BSplineSurface**, ...

  Any order surface

- Boolean solids

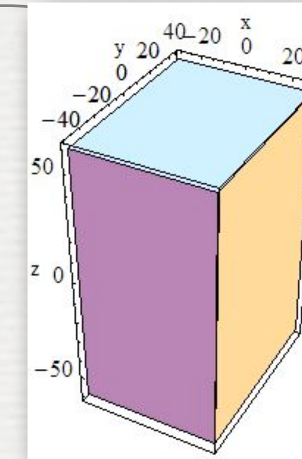
  G4UnionSolid, **G4SubtractionSolid**, ...



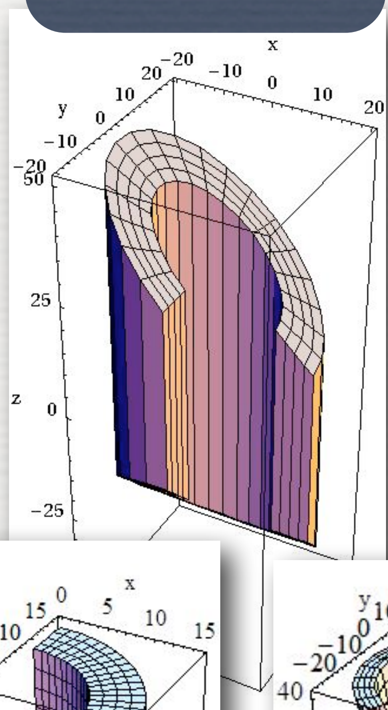
G4VSolid to define the shape

Constructed Solid Geometry (CSG) Solids

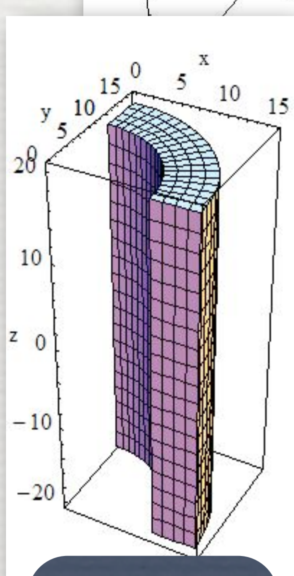
```
G4Box(const G4String &pname,    // name
             G4double half_x,    // X half size
             G4double half_y,    // Y half size
             G4double half_z);   // Z half size
```



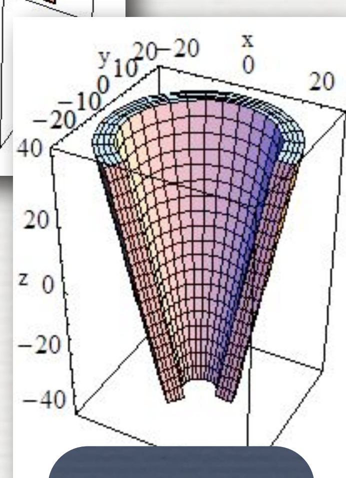
- ☛ sometimes center at 0 or not
- ☛ be careful for exportation ...
- ☛ sometimes several constructors



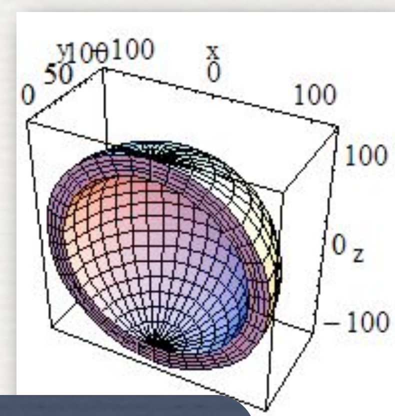
G4CutTub



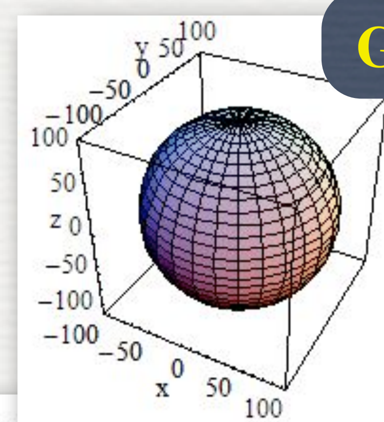
G4Tub



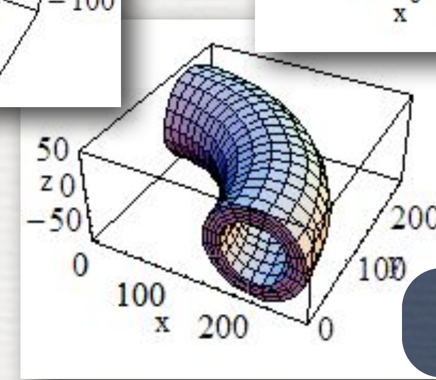
G4Cons



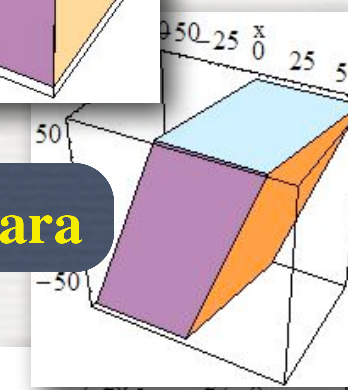
G4Sphere



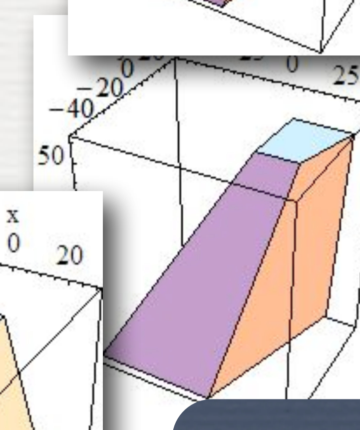
G4Orb



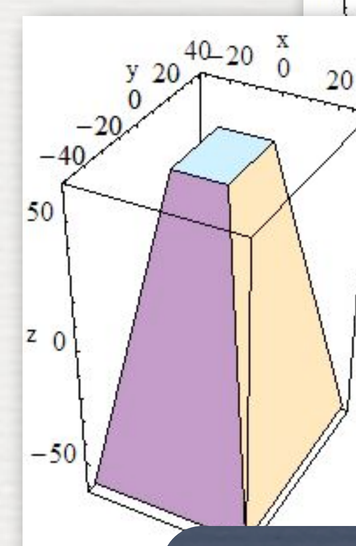
G4Torus



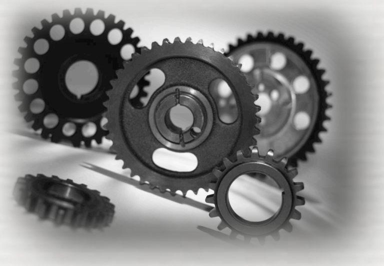
G4Para



G4Trap



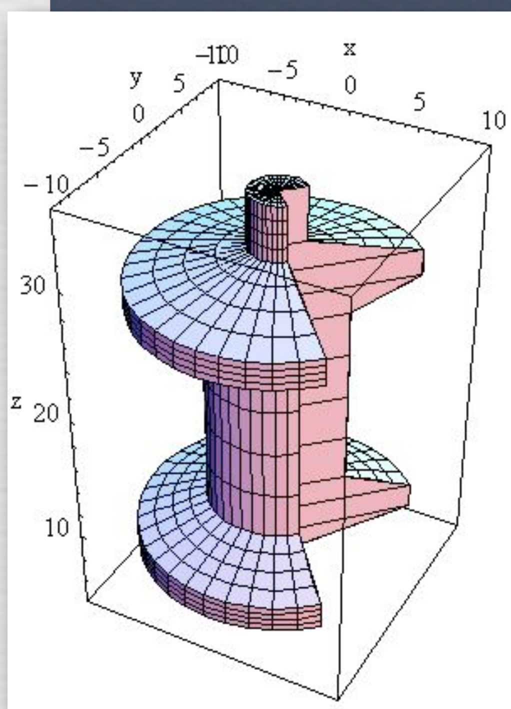
G4Trd



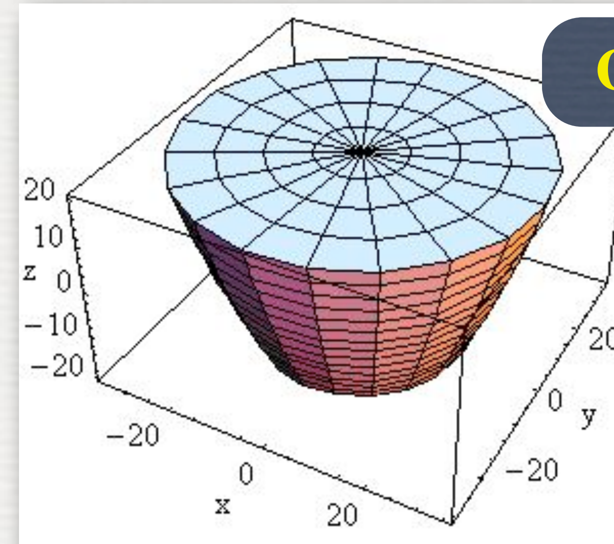
G4VSolid to define the shape

Constructed Solid Geometry (CSG) Solids

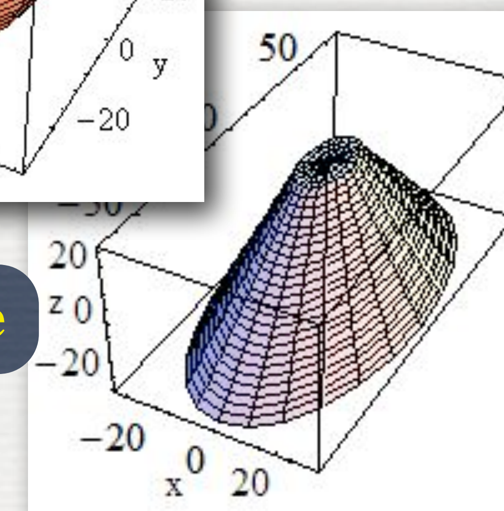
```
G4Polycone(const G4String& pName,
                G4double   phiStart,
                G4double   phiTotal,
                G4int      numZPlanes,
                const G4double   zPlane[],
                const G4double   rInner[],
                const G4double   rOuter[])
```



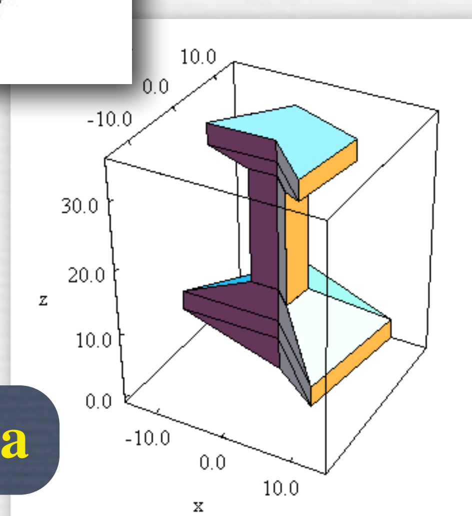
```
phiStart = 1/4*Pi, phiTotal = 3/2*Pi, numZPlanes = 9,

rInner = {0,0,0,0,0,0,0,0,0}, rOuter = {0,10,10,5,5,10,10,2,2},

z = {5,7,9,11,25,27,29,31,35}
```



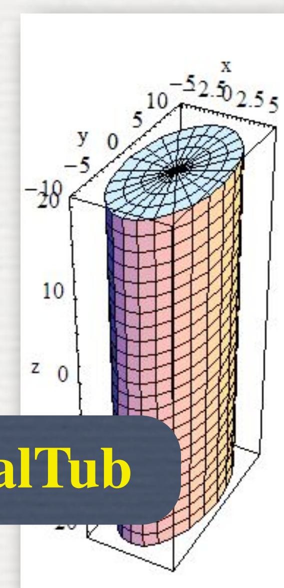
G4Paraboloi



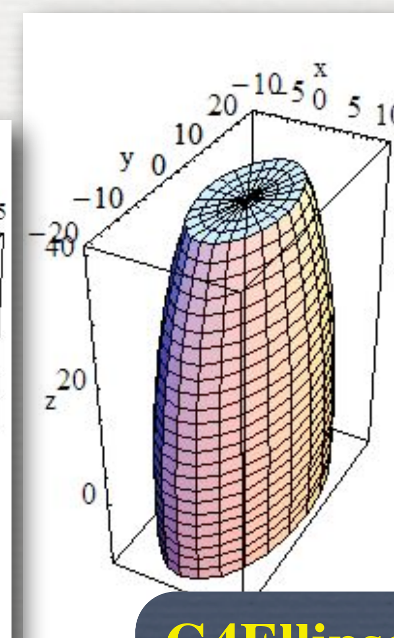
G4EllipticalCone



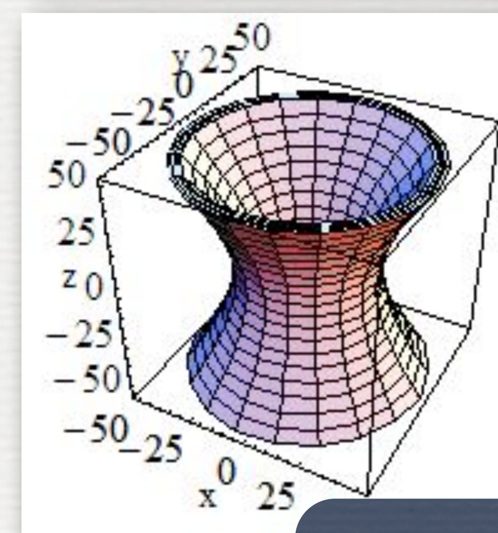
G4Polyhedra



G4EllipticalTub

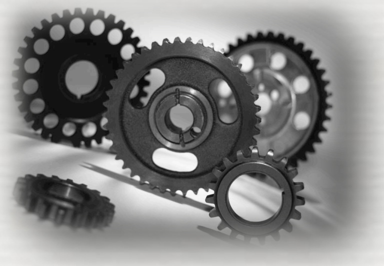


G4Ellipsoid



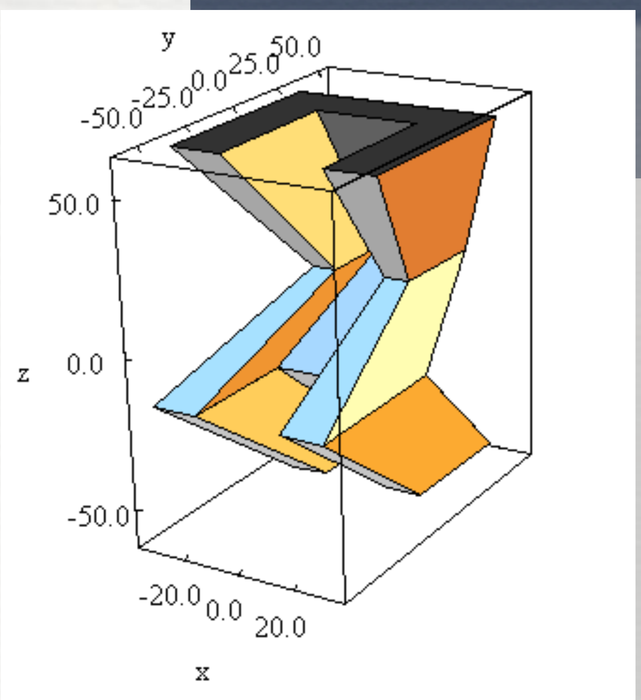
G4Hyp

G4VSolid to define the shape



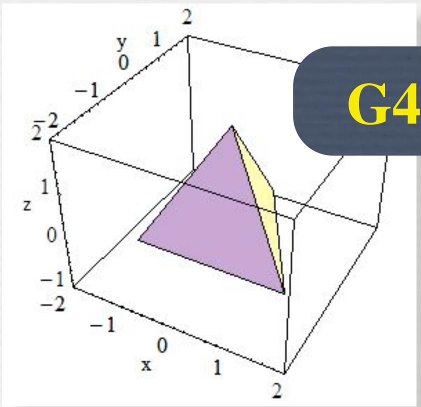
Constructed Solid Geometry (CSG) Solids

```
G4ExtrudedSolid(const G4String& pName,
                std::vector<G4TwoVector> polygon,
                std::vector<ZSection> zsections)
```

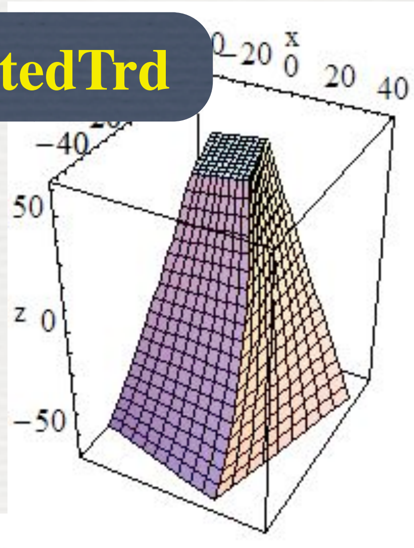


polygon={-30,-30},{-30,30},{30,30},{30,-30},{15,-30},{15,15},{-15,15},{-15,-30}

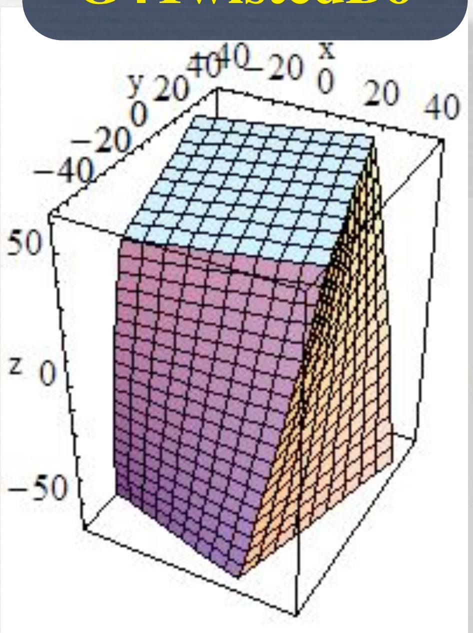
zsections=[-60,{0,30},0.8],[-15,{0,-30},1.],[10,{0,0},0.6],[60,{0,30},1.2]



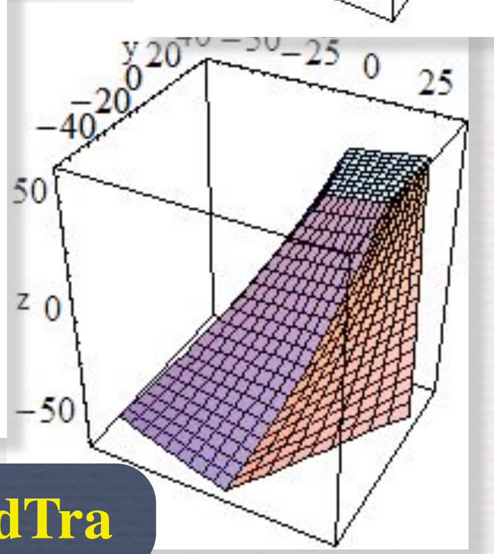
G4Tet



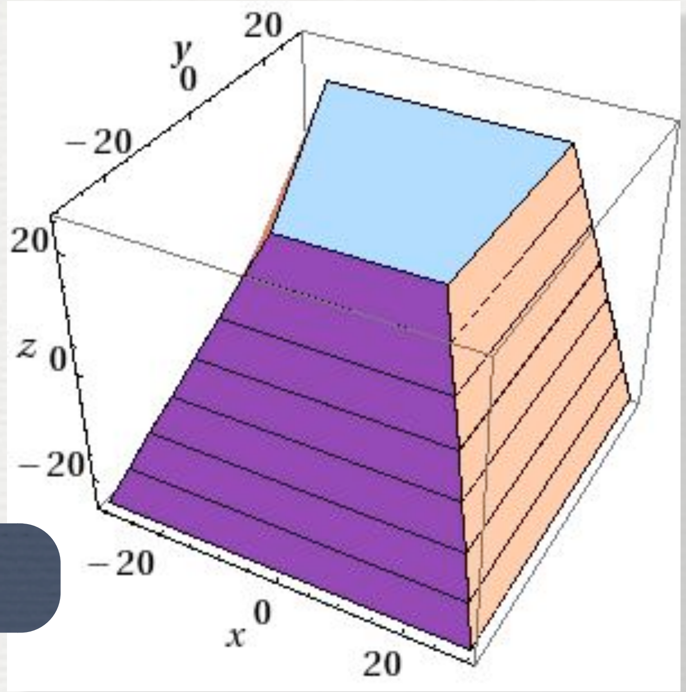
G4TwistedTrd



G4TwistedBox



G4TwistedTrap



G4GenericTrap

G4VSolid to define the shape

BREP (Boundary REPresented) Solids

- Listing all its surfaces specifies a solid

e.g. 6 planes for a cube

- Surfaces can be

planar, 2nd or higher order

☛ elementary BREPS

Splines, B-Splines, NURBS (Non-Uniform B-Splines)

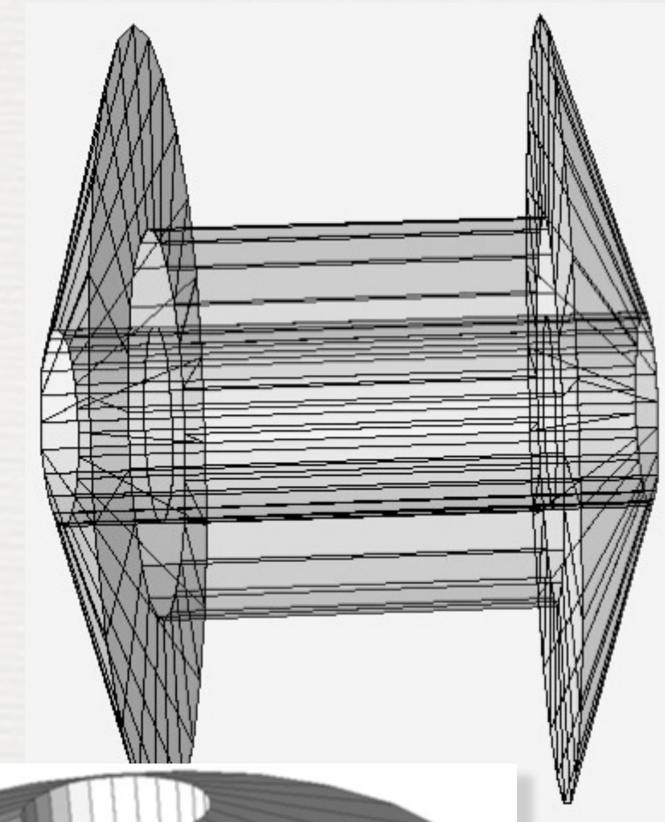
☛ advanced BREPS

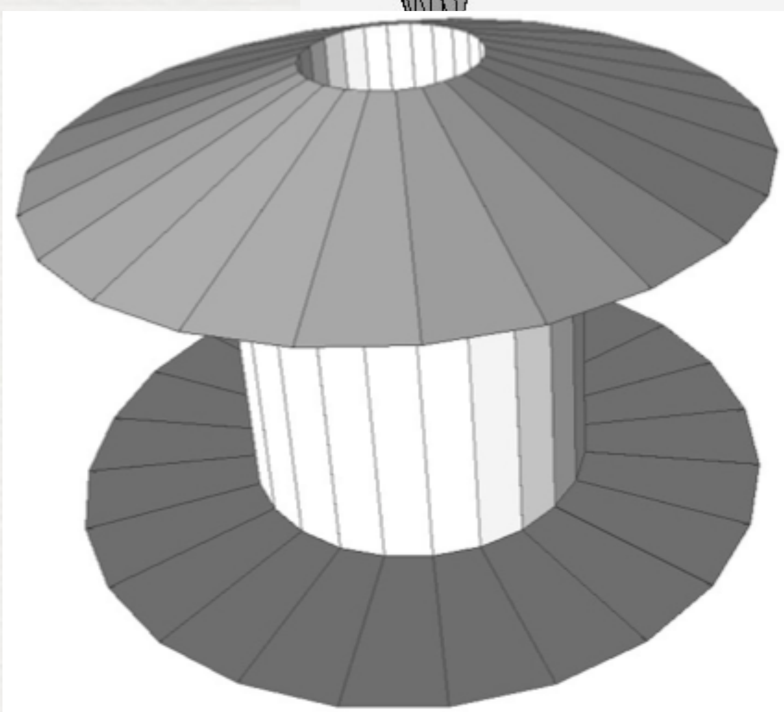
- Few elementary BREPS pre-defined

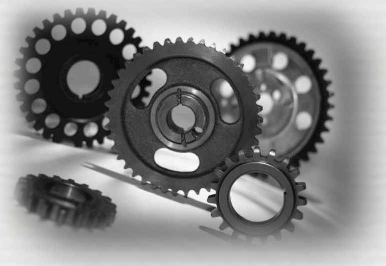
box, cons, tubs, sphere, torus, polycone, polyhedra

- Advanced BREPS built through CAD systems

```
G4BREPSolidPCone( const G4String& pName,
                  G4double  start_angle,
                  G4double  opening_angle,
                  G4int num_z_planes, //sections,
                  G4double  z_start,
                  const G4double  z_values[],
                  const G4double  RMIN[],
                  const G4double  RMAX[]  )
```







G4VSolid to define the shape

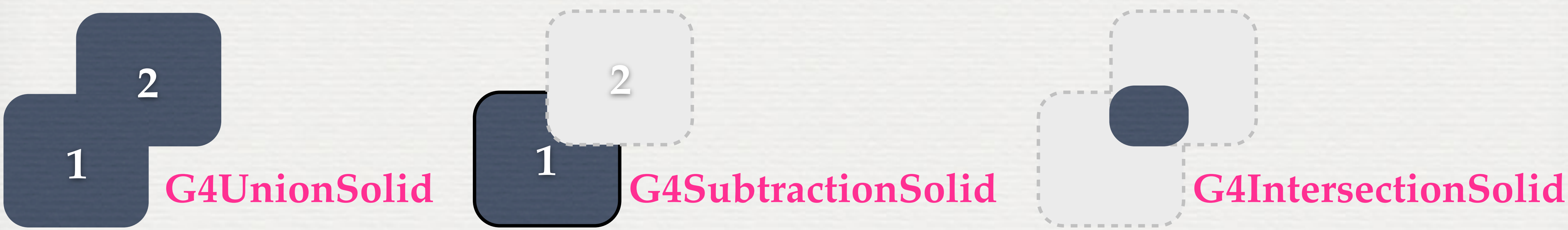
Boolean Solids

Solids can be combined using boolean operations:
G4UnionSolid, **G4SubtractionSolid**, **G4IntersectionSolid**

☛ Requirements: 2 solids, 1 boolean operation, and an (optional) transformation for the 2^{nd} solid
☛ 2^{nd} solid is positioned relative to the coordinate system of the 1^{st} solid
☛ Result of boolean operation becomes a solid. Thus the third solid can be combined to the resulting solid of first operation.

Solids to be combined can be either CSG or other Boolean solids.

Note: tracking cost for the navigation in a complex Boolean solid is proportional to the number of constituent CSG solids



G4VSolid to define the shape

Boolean Solids

```
G4VSolid* box = new G4Box(“Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder = new G4Tubs(“Cylinder”,0.,50.*cm,50.*cm,0.,2*M_PI*rad);
G4VSolid* union
     = new G4UnionSolid("Box+Cylinder", box, cylinder);


G4ThreeVector T(30.*cm,0.,0.);
G4VSolid* subtract
     = new G4SubtractionSolid("Box-Cylinder", box, cylinder,0, T);


G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4ThreeVector Tr(0.,0.,0.);
G4VSolid* intersect
     = new G4IntersectionSolid("Box&&Cylinder", box, cylinder, rm, Tr);
```

With all the possibilities proposed in Geant4 to build shapes
there are probably several ways to define a complex geometry

➥ be careful if you would like to export it ! [see gdml section]

W3: Geometries !

- Volumes - general aspects
- Definition of materials
- Definition shapes
- **All bricks together**
- Exportation / importation

How to define the World Volume

material

```
G4NistManager *man = G4NistManager::Instance();
G4PVPlacement *matWorld = man->FindOrBuildMaterial("G4_AIR");

// use a physical as a container to describe the detector
detWorld = new G4Box("BWorld",10.*m,10.*m,50.*m);
detlogicWorld = new G4LogicalVolume(detWorld, matWorld, "LWorld", 0, 0, 0);
```

Shape

➥ Logical world is a box made of air ... it is also hidden ...

```
detlogicWorld->SetVisAttributes(G4VisAttributes::Invisible); // hide the world

//  Must place the World Physical volume unrotated at (0,0,0).
thePhysWorld = new G4PVPlacement(0,                    // no rotation
                         G4ThreeVector(),        // no translation
                         detlogicWorld,          // its logical volume
                         "PWorld",               // its name
                         0,                      // its mother  volume
                         false,                  // no boolean operations
                         -1);                    // copy number
```

Place the World,
No mother,
No rotation
No translation

Adding daughter volumes to the World

- A volume is placed in its mother volume
  - ☛ Position, rotation of the daughter is described with respect to the local coordinate system of the mother
  - ☛The origin of the mother's local coordinate system is at the center of the mother volume
  - ☛ Daughter volumes cannot **protrude** from the mother volume, Daughter volumes cannot **overlap**
    - ➥ User's responsibility to check this, some tools are provided

➻ graphical widows [hepRApp, Qt]

➻ dedicated commands

/vis/ASCIITree/verbose 11
/vis/drawTree

/geometry/test/run or **geometry/test/grid_test**
check for overlapping regions based on a standard grid setup, limited to the first depth level
/geometry/test/recursive_test
applies the grid test to all depth levels (may require lots of CPU time!)

/geometry/test/line_test
to shoot a line along a specified direction and position

- The logical volume of mother knows the daughter volumes it contains
  - ☛ It is uniquely defined to be their mother volume

- One logical volume can be placed more than once. One or more volumes can be placed in a mother volume

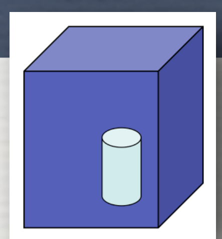
- The mother-daughter relationship is an information of **G4LogicalVolume**
  - ☛ If the mother volume is placed more than once then all daughters by definition appear in each placed physical volume

- The **world** volume must be a unique physical volume, it fully contains (with margin) all the other volumes
  - ☛ The world defines **the global coordinate system**, which origin is at the center of the world volume
  - ☛ Position of a track is given with respect to the global coordinate system

Adding daughter volumes to the World

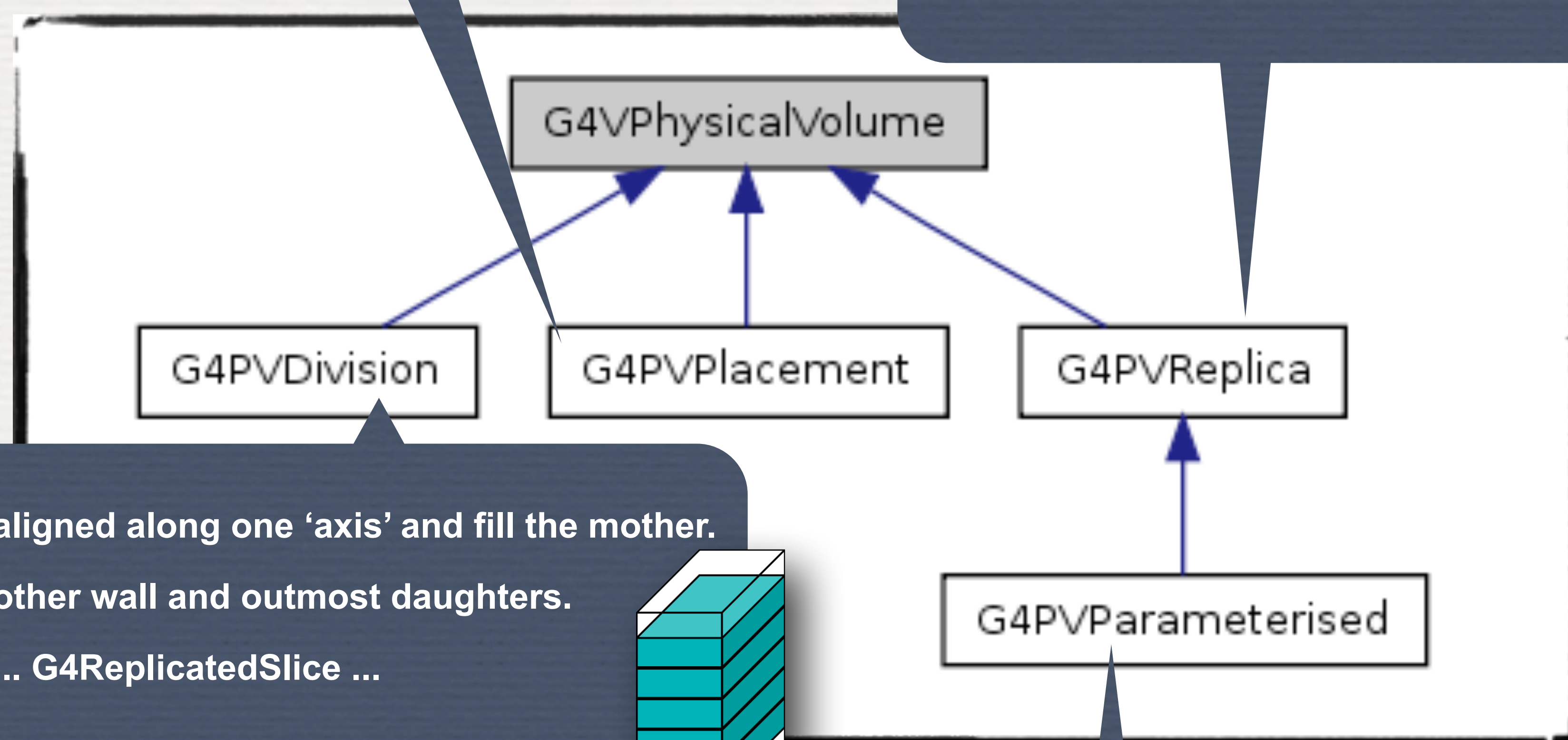
There are different ways to create physical (placed) volumes

A volume instance positioned once in its mother volume



Daughters of same shape are aligned along one 'axis'

Daughters fill the mother completely without gap in between.



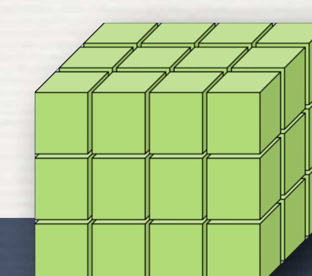
Daughters of same shape are aligned along one 'axis' and fill the mother.

There can be gaps between mother wall and outmost daughters.

No gap in between daughters ... G4ReplicatedSlice ...

+ **G4AssemblyVolume:**
  to make snapshot of a complex volume at given position,rotation
+ **G4ReflexionFactory:**
  a pair of volume, useful typically for end-cap calorimeter

Reduction of memory consumption



Currently: parameterization can be used only for volumes that either
- have no further daughters,
- are identical in size,shape (so that grand-daughters are safely fit inside)

Adding daughter volumes to the World

```
// Now add a blue cube to the world
 G4Box *asolidBox;
 G4LogicalVolume *alogicBox;
 G4VPhysicalVolume *aphysiBox;
 G4VisAttributes *visatt;

 asolidBox = new G4Box("BlueCube",Side/2.,Side/2.,Side/2.);
 alogicBox = new G4LogicalVolume(asolidBox, CubeMaterial, "LBlueCube", 0, 0, 0);

 // the cube is blue
 visatt = new G4VisAttributes( G4Colour(0.0, 0.0, 1.0) );
 visatt->SetVisibility(true);
 alogicBox->SetVisAttributes( visatt );

 aphysiBox = new G4PVPlacement(
              0,                                            // no rotation
      G4ThreeVector(X_Center, Y_Center, Z_Center), // at (X_Center,Y_Center,Z_Center)
          alogicBox,                                    // the blue cube logical volume
      "PBlueCube",                                  // the physical blue cube name
          logicWorld,                                   // its mother  volume
          false,                                        // no boolean operations
          0);                                           // copy number
```

user's limits

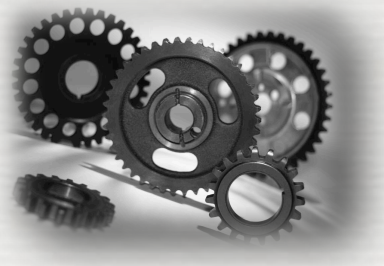
magnetic fields

last workshop #4

last workshop #4

W3: Geometries !

- Volumes - general aspects
- Definition of materials
- Definition shapes
- All bricks together
- **Exportation / importation**



GDML: exportation / importation

- Geometries can be saved in XML (gdml) files
- XML is widely used in computer applications since:
  - ➥ it is human readable (html like)
  - ➥ it is structured, with ways to check the schema is correct
  - ➥ the schema is defined consistently using xml language!
  - ➥ GDML* is an extension for 3D geometries

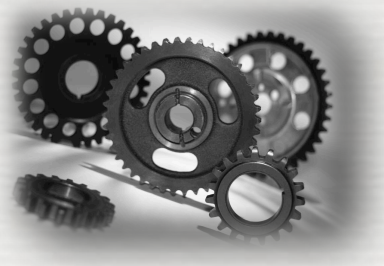
☛ It is a format to exchange geometries between framework

☛ BUT it could also be used to define new geometries (without C++ knowledge)

human readable !

☛ GDML is also the bridge to import CAD files ...

* Geometry Description Markup Language



GDML: exportation / importation

define gdml schema

```
<materials>
  <isotope N="138" Z="57" name="La138">
    <atom unit="g/mole" value="137.907"/>
  </isotope>
  <isotope N="139" Z="57" name="La139">
    <atom unit="g/mole" value="138.906"/>
  </isotope>
  <element name="Lanthanum">
    <fraction n="0.0009" ref="La138"/>
    <fraction n="0.9991" ref="La139"/>
  </element>
```

```
F-8" standalone="no" ?>
rg/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation
).cern.ch/service-spi/app/releases/GDML/schema/gdml.xsd">
```

Shape

```
 x="50.8" y="50.8" z="50.8"/>
="50.8" y="50.8" z="152.4"/>
V_2-bare" x="200" y="200" z="1000"/>
```

LogicalVolume

PhysicalVolume

Translation, rotation if any

the top volume, the World !

```
  <structure>
    <volume name="PW:0">
      <materialref ref="LaBr3"/>
      <solidref ref="LaBr3"/>
    </volume>
    <volume name="PW:1">
      <materialref ref="NaI"/>
      <solidref ref="NaI"/>
    </volume>
    <volume name="ParisPW_2-bare">
      <materialref ref="Air"/>
      <solidref ref="ParisPW_2-bare"/>
      <physvol name="PW:0">
        <volumeref ref="PW:0"/>
        <position name="PW:0_pos" unit="mm" x="0" y="0" z="25.9"/>
      </physvol>
      <physvol name="PW:1">
        <volumeref ref="PW:1"/>
        <position name="PW:1_pos" unit="mm" x="0" y="0" z="127.5"/>
      </physvol>
    </volume>
  </structure>

  <setup name="Default" version="1.0">
    <world ref="ParisPW_2-bare"/>
  </setup>

</gdml>
```

http://lcgapp.cern.ch/project/simu/framework/GDML/doc/GDMLmanual.pdf

GDML: exportation / importation

export the world from G4 into a gdml file

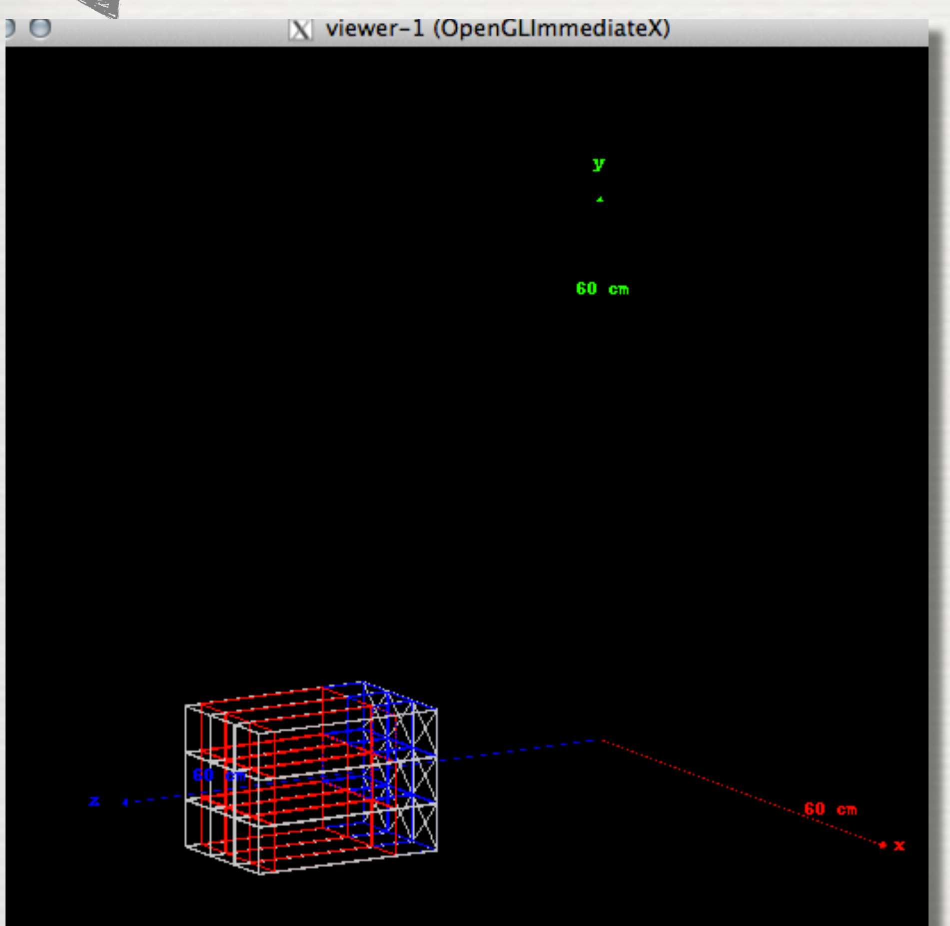
```
#include "G4GDMLParser.hh"

...

G4VPhysicalVolume *world; // this is the world
...

G4GDMLParser parser;
// write
parser.Write("myGDML.gdml",world,false);
```

import the world into G4 from a gdml file

```
#include "G4GDMLParser.hh"

...

G4VPhysicalVolume *world; // this is the world
G4GDMLParser parser;
parser.Read("myGDML.gdml",false);

world = parser.GetWorldVolume();
```

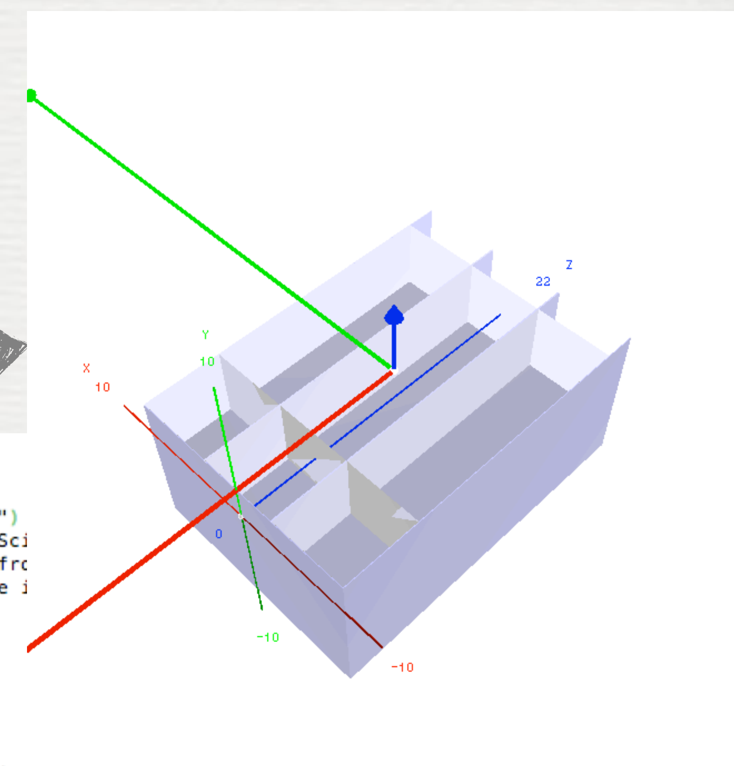
It requires Geant4 GDML module !

Attributes (colors, sensitivity ...) are not saved in gdml files ... there are way to pass the information



ROOT reads GDML files
only if
gdml module is compiled

```
CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TGeoManager::Import("DetectorFactory/Scintillators/CParisPW_2.gdml")
Info in <TGeoManager::Import>: Reading geometry from file: DetectorFactory/Sci
Info in <TGeoManager::TGeoManager>: Geometry GDMLImport, Geometry imported fro
Info in <TGeoManager::SetTopVolume>: Top volume is CParisPW_2. Master volume i
Info in <TGeoNavigator::BuildCache>: --- Maximum geometry depth set to 100
Info in <TGeoManager::CheckGeometry>: Fixing runtime shapes...
Info in <TGeoManager::CheckGeometry>: ...Nothing to fix
Info in <TGeoManager::CloseGeometry>: Counting nodes...
Info in <TGeoManager::Voxelize>: Voxelizing...
Info in <TGeoManager::CloseGeometry>: Building cache...
Info in <TGeoManager::CountLevels>: max level = 1, max placements = 27
Info in <TGeoManager::CloseGeometry>: 28 nodes/ 4 volume UID's in Geometry imp
Info in <TGeoManager::CloseGeometry>: ----------------modeler ready----------------
(class TGeoManager*)0x7f99e1869e00
root [1] gGeoManager->GetTopVolume()->Draw("ogl")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```



The user's application

<u>TODO List</u>

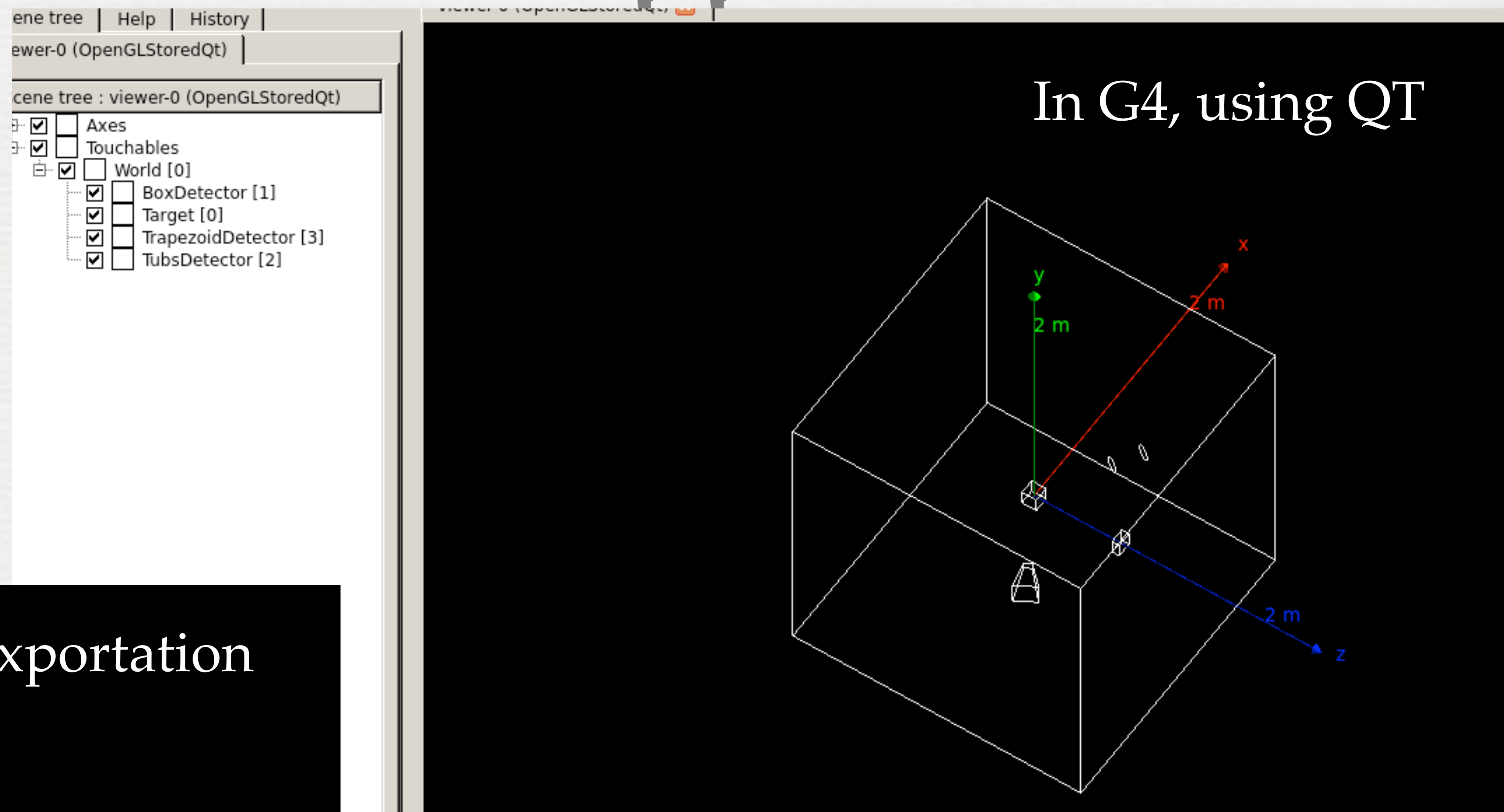
- Build the following setup:

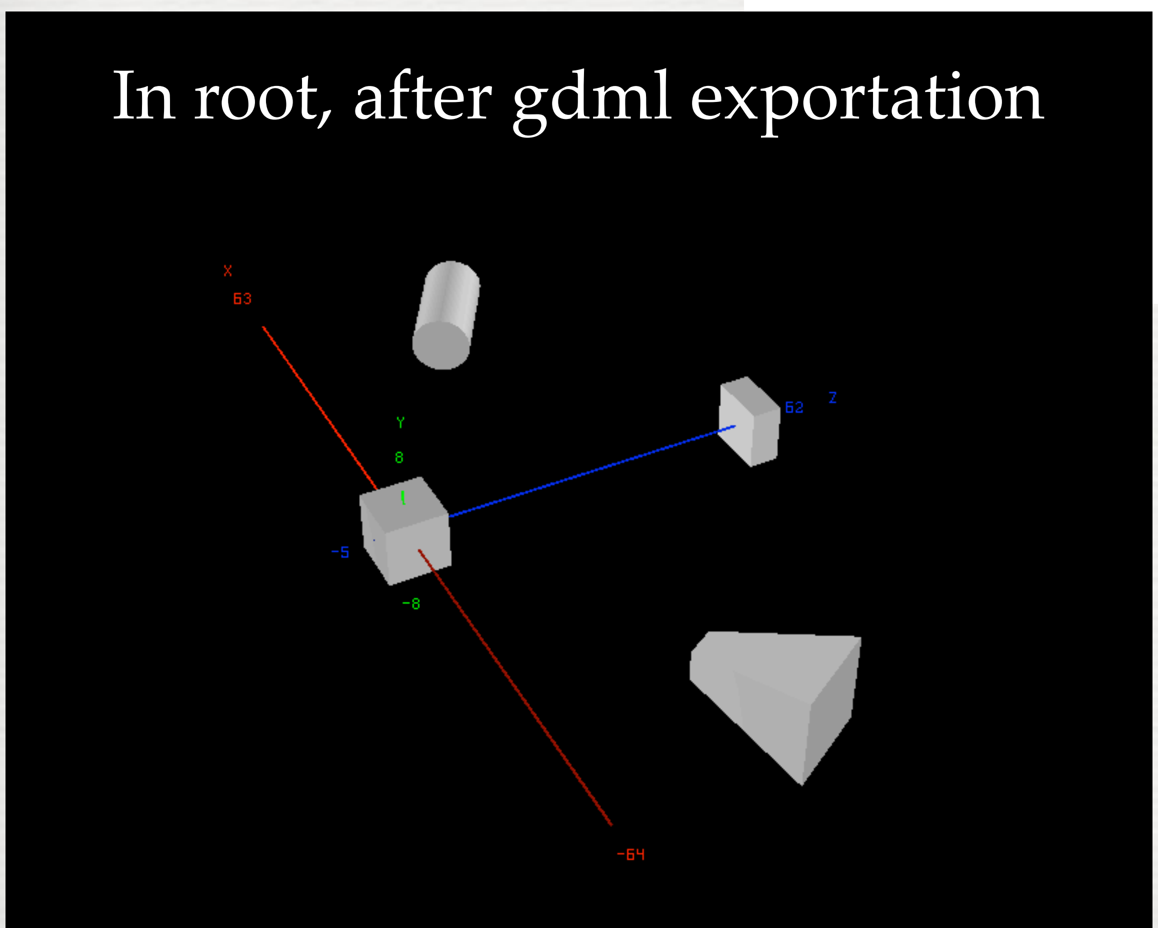
  The World is composed of air.
  The setup is a target composed of lead, placed at center of the World:
  target [box 10cm cube]
  And three detectors composed of BGO, placed at 60 cm from the target:
  a box [10 cm square, 5 cm depth] in the beam direction
  a tube detector 5cm radius, 10cm long rotated by 60 degrees with respect to the beam direction
  a trapezoid detector, face 5cm2, back 15cm2 depth 10cm rotated by -60 degrees with respect to the beam direction

- Modify the main program to save the geometry in a .gdml file
  load the geometry in root and check it

- Built your own detector !

The user's application





Conclusions of W3

We have seen:

- how to build a geometry
  - ➥ from isotopes to materials
  - ➥ from shapes by logical volumes to physical volumes
- how to use check the geometry validity
  - ➥ command line
  - ➥ using Graphical tools including export / import
- More information could be added to geometries
  - ➥ one can make some sensitive
  - ➥ copy number is important

} see last workshop !