Geant4: A Simulation toolkit

O. Stézowski and M. Pinto

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

NOW, HOW does it really work?

- W1: 3:00, Monday
- W2: 3:00, Tuesday
- W3: 4:30, Wednesday
- W4: 3:00, Thursday
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

```cpp
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);
};

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

+ many other hooks
but not mandatory

// 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 ARedSphereConstruction() );
    .
    .
    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]
  - ...

- G4VPhysicalVolume contains:
  - G4ThreeVector T
  - G4RotationMatrix R
  - copy #

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

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 \( \rightarrow \) G4Isotope
- elements \( \rightarrow \) G4Element
- molecules \( \rightarrow \) G4Material
- compounds and mixtures \( \rightarrow \) G4Material

\( \Rightarrow \) 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*/

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,massOfMole,density,nel,
kStateGas,temperature, pressure);

**composition of compound materials**

G4Element *c = ...  // carbone element
G4Element *na = new G4Element(name="Sodium",symbol="Na",z=11.,a);
G4Element *i = new G4Element(name="Iodine",symbol="I",z=53.,a);

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

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

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);

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

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);

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,massOfMole,density,nel,
kStateGas,temperature, pressure);

**absolute vacuum does not exist !**

Gaz at very low pressure

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);
G4Element *i = new G4Element(name="Iodine",symbol="I",z=53.,a);

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);

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

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);
G4Element *o = new G4Element(name="Oxygen",symbol="O",z=8.,a);

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);

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

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

Many elements defined

<table>
<thead>
<tr>
<th>Z</th>
<th>Name</th>
<th>density (g/cm³)</th>
<th>I (eV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G4_H</td>
<td>8.3740e-05</td>
<td>19.2</td>
</tr>
<tr>
<td>2</td>
<td>G4_He</td>
<td>0.000166322</td>
<td>41.8</td>
</tr>
<tr>
<td>3</td>
<td>G4_Li</td>
<td>0.534</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>G4_Be</td>
<td>1.848</td>
<td>63.7</td>
</tr>
<tr>
<td>5</td>
<td>G4_B</td>
<td>2.37</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>G4_C</td>
<td>2</td>
<td>81</td>
</tr>
<tr>
<td>7</td>
<td>G4_N</td>
<td>0.0011652</td>
<td>82</td>
</tr>
<tr>
<td>8</td>
<td>G4_O</td>
<td>0.00133151</td>
<td>95</td>
</tr>
<tr>
<td>9</td>
<td>G4_F</td>
<td>0.00158029</td>
<td>115</td>
</tr>
<tr>
<td>10</td>
<td>G4_Ne</td>
<td>0.000838505</td>
<td>137</td>
</tr>
<tr>
<td>11</td>
<td>G4_Na</td>
<td>0.971</td>
<td>149</td>
</tr>
<tr>
<td>12</td>
<td>G4_Mg</td>
<td>1.74</td>
<td>156</td>
</tr>
<tr>
<td>13</td>
<td>G4_Al</td>
<td>2.699</td>
<td>166</td>
</tr>
<tr>
<td>14</td>
<td>G4_Si</td>
<td>2.33</td>
<td>173</td>
</tr>
<tr>
<td>15</td>
<td>G4_P</td>
<td>2.2</td>
<td>173</td>
</tr>
<tr>
<td>16</td>
<td>G4_S</td>
<td>2</td>
<td>180</td>
</tr>
<tr>
<td>17</td>
<td>G4_Cl</td>
<td>0.00299473</td>
<td>174</td>
</tr>
<tr>
<td>18</td>
<td>G4_Ar</td>
<td>0.00166201</td>
<td>188</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ncomp</th>
<th>Name</th>
<th>density (g/cm³)</th>
<th>I (eV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>G4_A-150_TISSUE</td>
<td>1.127</td>
<td>65.1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.101327</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.7755</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0.035057</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0.0523159</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>0.017422</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>0.018378</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>G4_ACETONE</td>
<td>0.7899</td>
<td>64.2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.104122</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.620405</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0.275473</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>G4_ACETYLENE</td>
<td>0.0010967</td>
<td>58.2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.077418</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.922582</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>G4_ADENINE</td>
<td>1.35</td>
<td>71.4</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0.037294</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.44443</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0.518276</td>
<td></td>
</tr>
</tbody>
</table>

* http://physics.nist.gov/PhysRefData

natural isotope compositions
more than 3000 isotope masses

C++

G4

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

/material/nist/printElement
/material/nist/listMaterials
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**
  - Parameters: `const G4String &pname`, `G4double half_x`, `G4double half_y`, `G4double half_z`
  - Description: Sometimes center at 0 or not.
  - Advice: Be careful for exportation ...
  - Notes: Sometimes several constructors.

Additional Solid Functions:
- **G4Tubs**
- **G4Cons**
- **G4Sphere**
- **G4Torus**
- **G4Orb**
- **G4Para**
- **G4Trd**
- **G4Trap**
- **G4CutTubs**

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}

G4VSolid to define the shape

**Constructed Solid Geometry (CSG) Solids**

- **G4ExtrudedSolid**
  - `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]
G4VSolid to define the shape

BREP (Boundary REPReented) 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 2nd solid
  - 2nd solid is positioned relative to the coordinate system of the 1st 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

---

**Links**

G4VSolid to define the shape

Boolean Solids

```cpp
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
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);

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
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 [hepRAp, Qt]
      ✴ dedicated commands

• 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
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’ and 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
+ G4ReflectionFactory:
  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)
// 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,
    G4ThreeVector(X_Center, Y_Center, Z_Center),
    alogicBox,
    "PBlueCube",
    logicWorld,
    false,
    0);

// no rotation
// at (X_Center,Y_Center,Z_Center)
// the blue cube logical volume
// the physical blue cube name
// its mother volume
// no boolean operations
// copy number
W3: Geometries!

- Volumes - general aspects
- Definition of materials
- Definition shapes
- All bricks together
- 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)

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

* Geometry Description Markup Language

⇒ see section 4.1.2.4. Tessellated Solids
GDML: exportation / importation

Define GDML schema:

- Shape: LogicalVolume
- PhysicalVolume
- Translation, rotation if any

The top volume, the World!

GDML: exportation / importation

GDML: exportation / importation

```
<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="0.009" ref="La138"/>
    <fraction n="0.991" ref="La139"/>
  </element>
</materials>

<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: exportation / importation

Define GDML schema

Shape

LogicalVolume

PhysicalVolume

Translation, rotation if any

The top volume, the World!

GDML: exportation / importation

Export the world from G4 into a gdml file

```cpp
#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

```cpp
#include "G4GDMLParser.hh"
...
G4VPhysicalVolume *world; // this is the world
G4GDMLParser parser;
parser.Read("myGDML.gdml", false);
world = parser.GetWorldVolume();
```

ROOT reads GDML files only if gdml module is compiled

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

It requires Geant4 GDML module!
TODO List

• 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

In G4, using QT

In root, after gdml exportation
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!