GEANT4 GEOMETRY FROM TEXT FILE version 1.0.

Introduction

Since release 9.2 of Geant4, it is also possible to import geometry setups based on a plain text description, according to a well defined syntax for identifying the different geometrical entities (solids, volumes, materials and volume attributes) with associated parameters. An example showing how to define a geometry in plain text format and import it in a Geant4 application is shown in `examples/extended/persistency/P03`. We describe here first the format of the text file description and then how to use it in a Geant4 example.

Description of format

The description of the geometry is based on tags. A tag is a word that appears at the first one in a line and sets what the line means, for example if you write `:SOLID myBox BOX 10. 10. 10.`, the first word `:SOLID` means that the line describe the solid shape and parameters.

There are no constraints on the order of the tags in the file, except some logical restrictions, e.g. a volume cannot be positioned or given attributes if it has not been defined (e.g. no ':PLACE', ':VIS', ':COLOUR', ':CHECK_OVERLAPS' tags before ':VOLU' tag).

We will explain in this section the tags used to describe the geometry, also explaining the meaning of each of the words that follow the tag, and an example of each tag. Tags can be given with any combination of upper case and lower case letters. Each tag has a fixed number of arguments, known by the parser; therefore you may write all arguments in a line or in several lines at your will. You will observe that the order of parameters is always the same as the order of the parameters in the corresponding Geant4 class (if it exists).

Elements and materials

Isotope

:ISOT

- Name
- Z
- N
- A

Example: `:ISOT Cl35 17 18 35.`

Element made of one unique isotope

:ELEM

- Name
- Symbol
- Z
- A

Example: `:ELEM Hydrogen H 1. 1.`

Element composed of several isotopes

:ELEM_FROM_ISOT

- Name
- Symbol
- Number of components

One line per isotope with

- isotope name
- fraction of number of atoms per volume

Example: `:ELEM_FROM_ISOT Chlorine Cl 2 Cl35 0.4 Cl36 0.6`

Material made of one element

:MATE

- Name
- Z
- A
- Density

Example: `:MATE Iron 26. 55.85 7.87`

Material made of a mixture of elements or materials

:MIXT

- Name
- Density
- Number of components

One line per material or element with

- material name
- proportion of material in the mixture

The components can be either all elements or all materials, but both types cannot appear in the same mixture.

There are three mixture tags, depending of the way the proportions are defined:

- *:MIXT_BY_WEIGHT*Proportions by weight fractions (This tag is equivalent to the ":MIXT" tag)
- *:MIXT_BY_NATOMS* Proportions by number of atoms
- *:MIXT_BY_VOLUME* Proportions by volume

The first tag can be used to build material mixtures out of elements or materials, but the second tag can only be used with elements as components and the last tag can only be used with material as components.

Example: `:MIXT Fiber_Lead 9.29 2 Lead 0.9778 Polystyrene 0.0222 :MIXT_BY_NATOMS CO2 1.8182E-3 2 C 1 O 2 :MIXT_BY_VOLUME H-CO2 (1.214E-03+1.8182E-3)/2. 2 Hydrogen 0.5 CO2 0.5`

Geant4 internal database of materials and elements

Geant4 provides a list of predefined materials, whose compositions correspond to the NIST definition. Among them you can find all single elementary materials, from Z = 1 (Hydrogen) to Z = 98 (Californium). You can use those materials when building a volume without the need to redefine them on your ASCII file. It is just enough that the material name you assign to a volume corresponds to the name of one of these predefined materials (they all start with “G4_”). The Geant4 materials have the mean excitation energy set explicitly, instead of allowing an automatic calculation from its components. You may override those materials if you want by redefining them in your ASCII file.

Also Geant4 provides the definition of all elements from Z = 1 (Hydrogen) to Z = 107 (Bohrium). Their names are the usual symbol in the periodic table of elements (no “G4_”). These elements take into account the isotope composition.

Apart from the elementary materials, many other are available, usually related to medical physics applications. The full list of materials and their composition can be found here (http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/apas08.html#sect.G4M

Solids

:SOLID

- solid name
- solid type name
- ... List of solid parameters

The meaning and order of the solid parameters is the same as in the corresponding Geant4 solid constructor. All the Geant4 CSG and "specific" solids are implemented. The list of solid types and the corresponding parameters is the
following (for better understanding of the solid parameters meaning we refer to the Geant4 documentation (https://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/ch04.html#sect.Geom

BOX: box

- X Half-length
- Y Half-length
- Z Half-length

TUBE: tube

- Inner radius
- Outer radius
- Half length in z

TUBS: tube section

- Inner radius
- Outer radius
- Half length in z
- Starting phi angle
- Delta angle of the segment

CONE: cone

- Inner radius at -fDz
- Inner radius at +fDz
- Outer radius at -fDz
- Outer radius at +fDz
- Half length in z (=fDz)

CONS: cone section

- Inner radius at -fDz
- Inner radius at +fDz
- Outer radius at -fDz
- Outer radius at +fDz
- Half length in z (=fDz)
- Starting angle of the segment
- Delta angle of the segment

TRD: trapezoid

- Half-length along x at the surface positioned at -dz
- Half-length along x at the surface positioned at +dz
- Half-length along y at the surface positioned at -dz
- Half-length along y at the surface positioned at +dz
- Half-length along z axis

PARA: paralellepiped

- Half-length along x at the surface positioned at -dz
- Half-length along x at the surface positioned at +dz
- Half-length along y at the surface positioned at -dz
- Angle formed by the y axis and by the plane joining the centre of the faces G4Parallel to the z-x plane at -dy and +dy
- Polar angle of the line joining the centres of the faces at -dz and +dz in z
- Azimuthal angle of the line joining the centres of the faces at -dz and +dz in z Half-length along y at the surface positioned at +dz

TRAP: generic trapezoid

- Half-length along the z-axis (=pDz)
- Polar angle of the line joining the centres of the faces at -/+pDz
- Azimuthal angle of the line joining the centre of the face at -pDzto the centre of the face at +pDz
- Half-length along y of the face at -pDz (=pDy1)
- Half-length along x of the side at y=-pDy1 of the face at -pDz
- Half-length along x of the side at y=+pDy1 of the face at -pDz
- Angle with respect to the y axis from the centre of the side at y=-pDy1 to the centre at y=+pDy1 of the face at -pDz
- Half-length along y of the face at +pDz (=pDy2)
- Half-length along x of the side at y=-pDy2 of the face at +pDz
- Half-length along x of the side at y=+pDy2 of the face at +pDz
- Angle with respect to the y axis from the centre of the side at y=-pDy2 to the centre at y=+pDy2 of the face at +pDz

or alternatively, if your trapezoid is a simpler one, you can use the parameters

- Length along z
- Length along y
- Length along x at the wider side
- Length along x at the narrower side

SPHERE: sphere

- Inner radius
- Outer radius
- Starting angle of the segment
- Delta angle of the segment
- Theta starting angle of the segment
- Theta delta angle of the segment

ORB: full solid sphere

- Outer radius

TORUS: torus

- Inside radius
- Outside radius
- Swept radius of torus
- Starting Phi angle `(fSPhi+fDPhi <= 2PI, fSPhi > -2PI)`
- Delta angle of the segment

POLYCONE: polycone

- Initial phi starting angle
- Total phi angle
- Number of z planes or Number of rz points

For each z plane:

- Position of z plane
- Tangent distance to outer surface
- Half-length along the z-axis

For each rz corner:

- R coordinate of these corners
- Z coordinate of these corners

The software will know which if the numbers refer to plane or rz points by looking at the number of parameters provided and comparing it with the number expected

Example:

:SOLID polyc POLYCONE 0 360 6
3 -2
3.5 -2

3.5 0.75
3.75 1
3.75 2
3 2

or equivalently

:SOLID polyc POLYCONE 0 360 4
-2 3 3.5
0.75 3 3.5
1. 3. 3.75
2. 3. 3.75

POLYHEDRA: polyhedra

- Initial phi starting angle
- Total phi angle
- Number of sides
- Number of z planes or Number of rz points

For each z plane:

- Position of z plane
- Tangent distance to outer surface
- Half-length along the z-axis

For each rz corner:

- R coordinate of these corners
- Z coordinate of these corners

The software will know which if the numbers refer to plane or rz points by looking at the number of parameters provided and comparing it with the number expected

Example:

:SOLID polyh POLYHEDRA 20. 180. 3 4
1900. 32.
1800. 30
1800. 0.
1900. 0.

or equivalently

```
:SOLID polyh POLYHEDRA 20. 180. 3 2
1800. 0. 30.
1900. 0. 32.
```

ELLIPTICAL_TUBE: elliptical tube

- Half length X
- Half length Y
- Half length Z

ELLIPSOID: ellipsoid

- Semiaxis in X
- Semiaxis in Y
- Semiaxis in Z
- Lower cut plane level, z
- Upper cut plane level, z

ELLIPTICAL_CONE: elliptical cone

- Semiaxis in X
- Semiaxis in Y
- Height of elliptical cone
- Upper cut plane level

HYPE: hyperbolic profile

- Inner radius
- Outer radius
- Inner stereo angle
- Outer stereo angle
- Half length in Z

TET: tetrahedron

- Anchor point
- Point 2
- Point 3
- Point 4
- Flag indicating degeneracy of points

TWISTED_BOX: box twisted along one axis

- Twist angle
- Half x length
- Half y length
- Half z length

TWISTED_TRAP: trapezoid twisted along one axis

- Twisted angle
- Half x length at y=-pDy
- Half x length at y=+pDy
- Half y length (=pDy1)
- Half z length (=pDz)
- Polar angle of the line joining the centres of the faces at -/+pDz
- Half y length at -pDz (=pDy2)
- Half x length at -pDz, y=-pDy1
- Half x length at -pDz, y=+pDy1
- Half y length at +pDz
- Half x length at +pDz, y=-pDy2
- Half x length at +pDz, y=+pDy2
- Angle with respect to the y axis from the centre of the side

TWISTED_TRD: twisted trapezoid with the x and y dimensions varying along z

- Half x length at the surface positioned at -pDz
- Half x length at the surface positioned at +pDz
- Half y length at the surface positioned at -pDz
- Half y length at the surface positioned at +pDz
- Half z length (=pDz)
- Twisted angle

TWISTED_TUBS: tube section twisted along its axis

- Twisted angle
- Inner radius at end-cap
- Outer radius at end-cap
- Half z length
- Phi angle of a segment

Boolean solids

The three types of Geant4 boolean solids are supported: union, subtraction and intersection. The same tag should be used as for normal solids, but putting as solid type the type of boolean operation. The parameters are

- Solid name
- Solid boolean operation (UNION/SUBTRACTION/INTERSECTION)
- First component solid name
- Second component solid name
- Name of relative rotation matrix
- Relative X position
- Relative Y position
- Relative Z position

Example: `:SOLID myunion UNION solid1 solid2 RM30 -11.8 12.5 0.`

Logical volumes

There are two ways to define a logical volume. You can build if from a previously declared solid associating a material to it

:VOLU

- Volume name
- Solid name
- Material name

Example: `:VOLU HALL HALL Air`

or you can skip the definition of the solid and in one unique line define the solid and the material (valid also for boolean solids). You should then use the same format as for the :SOLID tag, but adding as last word the material name

Example:

Instead of

:SOLID HALL BOX 5000. 5000. 20000.

:VOLU HALL HALL Air

use

`:VOLU HALL BOX 5000. 5000. 20000. Air`

Physical volumes

All the possible ways to place a volume in Geant4 are supported: a single placement, a parameterised one, a division, a replica and an assembly.

Single placement

:PLACE

- Volume name
- Copy number
- Parent volume name
- Name of rotation matrix
- X position
- Y position
- Z position

Example:

```
:VOLU yoke :TUBS  Iron   3  620. 820. 1270.
:PLACE  yoke  1   expHall  R00      0.0     0.0     370.
```

Parameterisation

The parameterisations supported are the placement of several copies of a volume along a line, in a circle and in a bidimensional grid (other types of parameterisation may be added at user request).

:PLACE_PARAM

- Volume name
- Copy number
- Parent volume name
- Parameterisation type
- Name of rotation matrix
- Number of copies
- Step (separation between copies)
- Offset
- Extra arguments (optional, depend on parameterisation type)

There are three types of linear parameterisations, along the three axis X,Y,Z (types: *LINEAR_X, LINEAR_Y, LINEAR_Z*) and a general one (type *LINEAR*) for which you have to add as extra arguments

the axis direction *DIR_X DIR_Y DIR_Z*. The offset for linear parameterisations represents the distance from the centre of the first copy to the point (0,0,0) along the line.

In the case of circle parameterisation, the circle is around the Z axis by default. If you want a circle around another axis you can provide as extra arguments the axis and, optionally, the position of the first copy.

There are three types of square parameterisation, in the planes XY,XZ,YZ (types: *SQUARE_XY, SQUARE_XZ, SQUARE_YZ*) and a general one (type *SQUARE*). For this bidimensional parameterisations you have to provide two copy numbers, two steps and, optionally, two offsets. For the general case, *SQUARE* the offset is not needed, but you have to add as extra arguments the two axis, that do not have to be orthogonal, and, optionally, the position of the first copy. In the case of this paramerisation type, you have to provide two number of copies, one for each axis.

Example:

:PLACE_PARAM mytube 1 subworld2 LINEAR_X RM0 5 20. 0.

:PLACE_PARAM mytube 1 subworld1 LINEAR RM0 5 20. 0. 1. 1. 1. -50. 0. 0.

:PLACE_PARAM mybox 0 mother CIRCLE RM0 30 0.209 1 150

:PLACE_PARAM mybox 1 subworld2 SQUARE_XZ RM0 5 5 20. 20.

:PLACE_PARAM mybox 1 subworld1 SQUARE RM0 5 8 20. 10. 0. 1. 1. 0. 1. 0.

Be aware that putting offset = 0 means that the first copy is placed at (0,0,0). This may be not what you want if, for example, you are filling a box with an square of small boxes using an square parameterisation: offset 0 will mean that all the copies are placed in the positive-positive quarter of the mother box.

Division

There are several ways to define a division in Geant4, by giving:

- the number of divisions (so that the width of each division will be automatically calculated)
- the division width (so that the number of divisions will be automatically calculated to fill as much of the mother as possible)
- both the number of divisions and the division width (this is especially designed for the case where the copies do not fully fill the mother)

To each of these types correspond a different tag

:DIV_WIDTH

- Volume name
- Parent volume name

- Material name
- Axis of division
- Division width
- Offset (not mandatory)

:DIV_NDIV

- Volume name
- Parent volume name
- Material name
- Axis of division
- Number of divisions
- Offset (not mandatory)

:DIV_NDIV_WIDTH

- Volume name
- Parent volume name
- Material name
- Axis of division
- Number of divisions
- Division width
- Offset (not mandatory)

Example:

:DIV_WIDTH mybox mother AIR Z 10.

:DIV_NDIV_WIDTH mytube mother copper PHI 12 10.*deg

Replica

To define a replica the following tag must be used:

:REPL

- Volume name
- Parent volume name
- Axis of division
- Number of divisions
- Division width

- Offset (not mandatory)

Example: `:REPL crystal Block X 10 5.*cm`

Remember, that different to the divisions, where the solid type and dimensions are calculated automatically be Geant4, in the case of replicas the volume name used must be the name of a previously defined volume. This solid is not really used for navigation but should have the correct type and dimensions for visualisation.

Assembly volumes

Assembly volumes are sets of logical volumes that are combined together, so that they act as if there were in a real mother, but without creation of the mother.

To define assembly volumes you have to define the relative rotations and positions of all the logical volumes

:VOLU_ASSEMBLY

- Volume name
- Number of logical volumes
- Axis of division
- Number of divisions
- Division width
- Offset (not mandatory)

One line per logical volume with

- Logical volume name
- Rotation matrix name
- position X
- position Y
- position Z

Then to place the assembly volume you can use:

:PLACE_ASSEMBLY

- Volume name
- Copy number
- Parent volume name
- Name of rotation matrix
- X position
- Y position
- Z position

Example:

```
:SOLID Crystal BOX 10 10 10
:SOLID Crystal2 BOX 5 5 5
:VOLU_ASSEMBLY CrystalSet 3
Crystal RM0 0. 0. 0.
Crystal RM1 0. 0. 20.
Crystal2 RM0 0. 20 -10

:PLACE_ASSEMBLY CrystalSet  1  expHall  R00   100.    0.    0.
```

Other tags

Rotation matrix

A rotation matrix is interpreted as the rotation that should be applied to the object in the reference system of its mother. It can be defined in three ways:

- a) By giving the three rotation angles around the X, Y and Z axis (in this order of rotations)
- b) By giving the polar and azimuthal angles of the X, Y and Z axis after the rotation is applied
- c) By giving the nine matrix elements of the rotation matrix: XX, XY, XZ, YX, YY, YZ, ZX, ZY, ZZ

The tag for the three cases is the same. The parser will know which case is meant by the number of parameters.

a) *:ROTM*

- Name
- Angle of rotation around global X axis
- Angle of rotation around global X axis
- Angle of rotation around global X axis

Example: `:ROTM R0 0. 0. 0.`

b)*:ROTM*

- Name
- Polar angle for axis X
- Azimuthal angle for axis X
- Polar angle for axis Y
- Azimuthal angle for axis Y
- Polar angle for axis Z

- Azimuthal angle for axis Z

Example: `:ROTM R0 90. 0. 90. 90. 0. 0.`

c)*:ROTM*

- Name
- 9 parameters defining the rotation matrix

Example: `:ROTM R0 1. 0. 0. 0. 1. 0. 0. 0. 1.`

Visibility

:VIS

- Volume name
- ON or TRUE, OFF or FALSE

Example: `:VIS yoke OFF`

By default the visibility of all volumes is set to ON

Colour and transparency

To define the colour of a volume

:COLOUR/:COLOR

- Volume name
- Red colour proportion
- Green colour proportion
- Blue colour proportion
- Transparency

The four parameters can take a value between 0 and 1. The transparency parameter is not mandatory.

Example: `:COLOUR NDC_chamber 0.2 0.4 0.1`

By default, the three colour proportions will be set to -1.

Check overlaps

Geant4 offers the possibility to check if a volume overlaps with other volumes. By default it is not set, but you can activate it with the commands

:CHECK_OVERLAPS

- Volume name
- ON or TRUE, OFF or FALSE

Example: `:CHECK_OVERLAPS NDC_chamber 0.2 0.4 0.1`

By default, the three colour proportions will be set to -1

Use of '*' in names

In the case of the *:VIS*, *:COLOUR* and *:CHECK_OVERLAPS* tags, you may use '*' to define the volume name. This '*' will be replaced by 'any name'. For example *Crys** means all the volume names starting by 'Crys', * means all volume names.

Use of parameters

You can also define a parameter for later use in any tag.

:P

- parameter name
- parameter value

You can then use the parameter as: '$' + parameter_name

Example:

```
:P InnerR 12.
:VOLU yoke :TUBS  Iron  3  $InnerR 820. 1270.
```

Units

Any value in a tag has a default unit that depends on the dimension of the value (automatically known by the parser). The default units are the following:

- length: mm
- angle: degrees
- density: g/cm3
- atomic mass: g/mole

The user can override the default value of a unit by indicating the unit of each value. This can be done adding at the end of the value the unit name (see CLHEP file *Units/SystemOfUnits.h*) preceded by a '*' character; e.g. 3*mm, 1.4*rad,...

Arithmetic Expressions

For any value you want to define in a tag you can use the most common mathematical expressions instead of directly writing the figures, e.g:

3-sin(8.2/3.5)
(3+4)*(7-log(3))

You can also use parameters in the expressions, e.g:

```
7.2*$RADIUS-$X_LENGTH/1.5
```

If you use a regular expression, remember that there can only be a unit in the whole expression, and it must be at the end.

The regular expressions used include (their meaning is evident): *+, -, *, /, sin, cos ,tan ,asin ,acos ,atan ,atan2 ,sinh ,cosh ,tanh ,sqrt ,exp ,log ,log10 ,pow*.

Including other files

You can nest several files by using the *#include'* directive anywhere in your geometry files.

Example: `#include mygeom2.txt`

Examples of code

The example `examples/extended/persistency/P03` shows how to use the geometry from text in your application. It also covers the case of associating a sensitive detector to one of the volumes defined in the text geometry, the case of mixing C++ and text geometry definitions and the case of defining new tags in the text format so that regions and cuts by region can be defined in the text file. And it provides an example on how to write a geometry text file from the in-memory Geant4 geometry.

Simple example

To read your geometry from a text file you have to write the following lines in your user detector class

```
#include "ExTGDetectorConstruction.hh"
#include "G4tgbVolumeMgr.hh"
#include "G4tgrMessenger.hh"

// ---------------------------------------------------------------------------
ExTGDetectorConstruction::ExTGDetectorConstruction()
{
  messenger = new G4tgrMessenger;
}

// ---------------------------------------------------------------------------
ExTGDetectorConstruction::~ExTGDetectorConstruction()
```

```
{
  delete messenger;
}

// ------------------------------------------------------------------------
G4VPhysicalVolume* ExTGDetectorConstruction::Construct()
{
  //-----------------------------------------------
  // Define one or several text files containing the geometry description
  //-----------------------------------------------
  G4String filename = "g4geom.txt";
  G4tgbVolumeMgr* volmgr = G4tgbVolumeMgr::GetInstance();
  volmgr->AddTextFile(filename);

  //-----------------------------------------------
  // Read the text files and construct the GEANT4 geometry
  //-----------------------------------------------
  G4VPhysicalVolume* physiWorld = volmgr->ReadAndConstructDetector();

  return physiWorld;
}
```

Then you can define your geometry in a file named *g4geom.txt*. You can see several examples of geometry text files in the mentioned examples.

Associating a sensitive detector to a volume defined in the geometry text file

When you use the above code to read your geometry from a text file, you can still associate a sensitive detector to one or more of the volumes defined in your text file. To do so you can get a pointer to the logical volume by looking for it by its name, like in the following examples:

```
G4LogicalVolume * logicChamber =
    G4tgbVolumeMgr::GetInstance()->FindG4LogVol("Chamber",0);
```

Once you have the logical volume pointer, you can associate a sensitive detector to it. See the following code, extracted from the class *ExTGDetectorConstructionWithSD* in the P03 example:

```
  G4SDManager* SDman = G4SDManager::GetSDMpointer();

  G4String trackerChamberSDname = "ExTextGeom/TrackerChamberSD";
  ExTGTrackerSD* aTrackerSD = new ExTGTrackerSD( trackerChamberSDname );
  SDman->AddNewDetector( aTrackerSD );
  G4LogicalVolume * logicChamber =
    G4tgbVolumeMgr::GetInstance()->FindG4LogVol("Chamber",0);
  if(logicChamber)
  {
    logicChamber->SetSensitiveDetector( aTrackerSD );
  }
  else
```

```
  {
    G4Exception("ExTGDetectorConstructionWithSD::Construct()",
                "InvalidGeometry", JustWarning,
                "Volume does not exists in geometry: Chamber.");
  }
```

Mixing C++- and text-file geometries

Geometries built from text files can be mixed with geometries built with C++ code in a seamless way. In the above section it was shown how to acces a logical volume built from a text file inside your C++ code. You can access in a similar way a physical volume:

```
GVPhysicalVolume * physChamber =
    G4tgbVolumeMgr::GetInstance()->FindG4PhysVol("Chamber",0);
```

Pointers to material, elements and isotopes can be obtained by name in the following way

```
G4Material* mate =
    G4tgbMaterialMgr::GetInstance()->FindBuiltG4Material("Water");

G4Element* elem =
    G4tgbMaterialMgr::GetInstance()->FindBuiltG4Elementl("Sodium");

G4Isotope* isot =
    G4tgbMaterialMgr::GetInstance()->FindBuiltG4Isotope("F18");
```

If you want to place a volume defined in your C++ code inside a volume read from the text file, you can simply get the G4LogicalVolume pointer of the text volume and place the C++ volume inside it. See the following code, extracted from the class *ExTGDetectorConstructionWithCpp* in the P03 example:

```
  //-------------------------------------------------
  // Build another volume and place it in the text geometry
  //-------------------------------------------------
  G4Box* solid = new G4Box("mybox",2.,2.,3.);

  G4Material* water = G4NistManager::Instance()->FindOrBuildMaterial("G4_Fe");

  G4LogicalVolume* logicVol = new G4LogicalVolume(solid,water,"mybox",0,0,0);

  //----- Place the volume in the world of the text geometry
  G4LogicalVolume* textWorldVol = physiWorld->GetLogicalVolume();

  // NOTE: if you want to place your full text geometry in the C++ geometry,
  //       you should use this G4LogicalVolume and place it in a C++ logical
  //       volume
  new G4PVPlacement(0,               // no rotation
                    G4ThreeVector(0.,-20.,0),  // at (x,y,z)
                    logicVol,        // its logical volume
                    "myBox1",        // its name
                    textWorldVol,    // its mother  volume
```

```
                  false,           // no boolean operations
                  0);              // copy number

  //----- Place the volume inside a volume of the text geometry
  G4LogicalVolume* textSphereVol =
    G4tgbVolumeMgr::GetInstance()->FindG4LogVol("sphere",1);
    new G4PVPlacement(0,           // no rotation
                  G4ThreeVector(0.,0.,0),  // at (x,y,z)
                  logicVol,        // its logical volume
                  "myBox2",        // its name
                  textSphereVol,   // its mother  volume
                  false,           // no boolean operations
                  1);              // copy number
```

Defining new tags in the geometry file format

The geometry file format can be extended by adding new tags. The P03 example illustrates how to add two new tags: to define a G4Region and to set the production cuts to a G4Region.

The first thing to do is to define your own class inheriting from *G4tgrLineProcessor*. In the method *ProcessLine* you should invoke the method *G4tgrLineProcessor::ProcessLine* and if it returns 0 (that is when a tag is not found) define what to do with your new tags. See the example at *ExTGRCLineProcessor*:

```
G4bool ExTGRCLineProcessor::ProcessLine( const std::vector<G4String> wl )
{

  G4bool iret = G4tgrLineProcessor::ProcessLine( wl );

  G4String wl0 = wl[0];
  for( size_t ii = 0; ii < wl0.length(); ii++ )
  {
    wl0[ii] = toupper( wl0[ii] );
  }

  if( !iret )
  {
    //------------------------------ parameter number
    if( wl0 == ":REGION" )
    {
      std::vector<G4String>::const_iterator ite = wl.begin()+1;
      std::vector<G4String> wlc;
      for( ; ite != wl.end(); ite++ )   //loop skipping the first one
      {
        wlc.push_back( *ite );
      }
      //      wlc = wlc.erase( wlc.begin() );
      ExTGRCRegionCutsMgr::GetInstance()->AddRegionData( wlc );
      iret = 1;

    }
```

```
    else if( wl0 == ":CUTS" )
    {
      std::vector<G4String>::const_iterator ite = wl.begin()+1;
      std::vector<G4String> wlc;
      for( ; ite != wl.end(); ite++ )   //loop skipping the first one
      {
        wlc.push_back( *ite );
      }
      ExTGRCRegionCutsMgr::GetInstance()->AddRegionCuts( wlc );
      iret = 1;
    }
    else
    {
      iret = 0;
    }
  }

  return iret;
}
```

You have also to define your own class inheriting from *G4tgbDetectorBuilder* and in the *ReadDetector* method set your line processor as the one to be used:

```
const G4tgrVolume* ExTGRCDetectorBuilder::ReadDetector()
{
  //------------------- construct geometry
  tlproc = new ExTGRCLineProcessor;
  G4tgrFileReader* tfr = G4tgrFileReader::GetInstance();
  tfr->SetLineProcessor( tlproc );
  tfr->ReadFiles();

  //---------- find top G4tgrVolume
  G4tgrVolumeMgr* tgrVolmgr = G4tgrVolumeMgr::GetInstance();
  const G4tgrVolume* tgrVoltop = tgrVolmgr->GetTopVolume();

  return tgrVoltop;
}
```

In the *BuildDetector* method of this class you should build the corresponding Geant4 objects (they could not be built in the line processor method, because the building of Geant4 objects is not triggered until the *BuildDetector* method is invoked, and therefore it would not be possible to associate into G4Regions the G4LogicalVolume objects):

```
//-----------------------------------------------------------------------
G4VPhysicalVolume* ExTGRCDetectorBuilder::
ConstructDetector( const G4tgrVolume* tgrVoltop)
{
  G4VPhysicalVolume* topPV =
    G4tgbDetectorBuilder::ConstructDetector( tgrVoltop );

  //--- Create regions
  ExTGRCRegionCutsMgr::GetInstance()->BuildRegions();
```

```
  //--- Set cuts to regions
  ExTGRCRegionCutsMgr::GetInstance()->BuildProductionCuts();

  return topPV;
}
```

Finally in your detector construction class you should set your detector builder as the one to be used. See the example at *ExTGDetectorConstructionWithCuts*

```
G4VPhysicalVolume* ExTGDetectorConstructionWithCuts::Construct()
{
  //-------------------------------------------------
  // Define one or several text files containing the geometry description
  //-------------------------------------------------
  G4String filename = "g4geom_cutsPerRegion.txt";
  G4tgbVolumeMgr* volmgr = G4tgbVolumeMgr::GetInstance();
  volmgr->AddTextFile(filename);

  //-------------------------------------------------
  // Use your own detector builder, that will invoke your own line processor
  //-------------------------------------------------
  ExTGRCDetectorBuilder* gtb = new ExTGRCDetectorBuilder;
  volmgr->SetDetectorBuilder( gtb );

  const G4tgrVolume* tgrVoltop = gtb->ReadDetector();
  G4VPhysicalVolume* physiWorld = gtb->ConstructDetector(tgrVoltop);

  //  G4VPhysicalVolume* physiWorld = volmgr->ReadAndConstructDetector();

  return physiWorld;
}
```

Known bugs

There are some known bugs introduced in release geant4.9.2:

- The phiDelta/phiTotal of solids G4Tubs, G4Cons, G4Sphere, G4Torus, G4Polycone, G4Polyhedra, G4TwistedTubs,G4BREPSolidPCone, G4BREPSolidPolyhedra and G4BREPSolidOpenPCone and thetaDelta of G4Sphere is always set to 360 deg, ignoring the user value

Please take the following **file** (http://geant4.cern.ch/collaboration/working_groups/geometry/docs/textgeom/G4tgbVolume.cc), copy it into your geant4 distribution, at source/persistency/ascii/src/G4tgbVolume.cc and recompile.