GDML Manual 1.4

What is GDML?

GDML stands for geometry description markup language. Its purpose is to store Geant4 setups. It is XML based so there is no need to use any special software to modify GDML files.

What is new?

In Geant4 9.1 GDML became a fully integrated part of the system. The previous implementation of GDML, known as GDML 2.10.0, was an external package.

Usage

Building Geant4 with GDML support is only a matter of setting an environment variable now. The setup script will offer it as an option. In your code you only need to include a single file:

```
#include <G4GDMLParser.hh>
```

Now you can instantiate the class which is capable of both reading and writing GDML files.

```
G4GDMLParser parser;
```

If you wish to read a GDML setup into geant4, use the following function with the filename specified as an argument.

```
parser.Read(„input.gdml”);
```

After a succesful reading use the function below to get a pointer to the world/top volume. The returned volume is a physical volume, so it can be directly used in your detector construction class.

```
G4VPhysicalVolume *pvWorld = parser.GetWorldVolume();
```

If you wish to write out your Geant4 setup created in C++, use the following function where you should specify two arguments: the name of the intended output file and a pointer to the world volume, what should be a logical volume.

```
parser.Write(„output.gdml”(G4LogicalVolume*),lvWorld);
```

In the „geant4/examples/extended/gdml” directory you can find an example demonstrating both the reader and the writer functionality.

A simple GDML example

A GDML file has five main sections within the root element: define, materials, solids, structure and setup. The root element itself is named gdml.

```
<gdml>
  </define>
  <materials>
    <material name=”Air” Z=”1.0”>
      <D value=”1e-25”/>
      <atom value=”1.00794”/>
    </material>
    <material name=”Aliminium” Z=”13.0”>
      <D value=”2.7”/>
      <atom value=”26.98”/>
    </material>
  </materials>
  <solids>
    <box name=”TheBox” x=”100” y=”100” z=”100” lunit=”mm”/>
    <box name=”WorldBox” x=”500” y=”500” z=”500” lunit=”mm”/>
  </solids>
  <structure>
     <volume name=”lvBox”>
       <materialref ref=”Aluminium”/>
       <solidref ref=”TheBox”/>
     </volume>
    <volume name=”TOP”>
      <materialref ref=”Air”/>
      <solidref ref=”WorldBox”/>
      <physvol>
         <volumeref ref=”lvBox”>
       </physvol>
    </volume>
  </stucture>
  <setup name=”Default” version=”1.0”>
    <world ref=”TOP”/>
  </setup>
</gdml>
```

Definitions

All the defined items should have a unique name.

constant

Once a constant is defined, it can be used in any expression. The name is mandatory and the value is zero by default. A constant has no unit.

```
<constant name=”Pi”  value=”3.1415”/>
<constant name=”TwoPi”  value=”2.0*Pi”/>
```

variable

Once a constant is defined, it can be used in any expression and loop. The name is mandatory and the value is zero by default. A variable has no unit.

```
<variable name=”x”  value=”125.0”/>
```

position

Once a position is defined, it can be referenced where a position is expected. The name is mandatory, the coordinates are zeros and the unit is „mm” by default.

```
<position name=”pos”  x=”25.0”  y=”50.0”  z=”75.0”  unit=”mm”/>
<position name=”TranslateX”  x=”100.0”  unit=”cm”/>
```

Constants can be used as well:

```
<constant name=”size” value=”25.0”/>
<position name=”pos”  x=”size”  y=”2.0*size”  z=”3.0*size”  unit=”mm”/>
```

rotation

Once a rotation is defined, it can be referenced where a rotation is expected. The name is mandatory, the rotations are zeros and the unit is in radian by default.

```
<rotation name=”rot”  x=”30.0”  y=”45.0”  z=”60.0”  unit=”deg”/>
<rotation name=”RotateZ”  z=”30”  unit=”deg”/>
```

Constants can be used as well:

```
<constant name=”Pi”  value=”3.1415”/>
<rotation name=”rot”  x=”Pi/6.0”  y=”Pi/4.0”  z=”Pi/3.0”  unit=”rad”/>
```

scale

Once a scale is defined, it can be referenced where a scale is expected. The name is mandatory and every scale factor is one by default. Scale has no unit.

```
<scale name=”scl”  x=”2.0”  y=”4.0”  z=”8.0” />
<scale name=”StrectchY”  y=”2.0”/>
```

Constants can be used as well:

```
<constant name=”ratio” value=”2.0”/>
<scale name=”scl”  x=”ratio”  y=”2.0*ratio”  z=”4.0*ratio” />
```

Scale can be used to implement reflection:

<scale name=”planar_reflection” x=”-1” y=”+1” z=”+1” />
<scale name=”axial_reflection” x=”-1” y=”-1” z=”+1” />
<scale name=”point_reflection” x=”-1” y=”-1” z=”-1” />

Matrix

Materials

Solids

The geometry of a particular setup is described with a composition of various solids. All of them should be defined in the solids section. In GDML the following solids are available:

box

Definition of an axis aligned box. The origin is at the center of the box.

x,y,z: dimensions of the box

<box name=”TheBox” x=”100” y=”200” z=”300” lunit=”mm”/>

cone

Definition of a cone segment aligned to the Z-axis. The origin is at halfway between the base and the top of the cone. The inner radius always should be less than it’s respective outter radius.

rmin1: inner radius at the base
rmax1: outter radius at the base
rmin2: inner radius at the top
rmax2: outter radius at the top
z: distance between the base and the top
startphi: start angle of cone segment
endphi: end angle of cone segment

<cone name=”TheCone” rmin1=”1.0” rmax1=”2.0” rmin2=”2.0” rmax2=”4.0” z=”4.0” startphi=”0.0” endphi=”180.0” aunit=”deg” lunit=”mm”/>

ellipsoid

Definition of an axis-aligned ellipsoid. The origin is at the center. The clipping planes are perpendicular to the Z-axis and their position is measured along the Z-axis. The position of the first clipping plane must be less than the position of the second clipping plane. The example illustrates a clipped ellipsoid.

ax,by,cz: radiuses
zcut1: position of first clipping plane
zcut2: position of second clipping plane

<ellipsoid name=”TheEllipsoid” ax=”1.0” by=”2.0” cz=”3.0” zcut1=”-1.5” zcut2=”1.5” lunit=”mm”/>

eltube

Definition of a tube with elliptical base. The origin is at the center.

dx,dy: radiuses of the base ellipsoid
dz: half length of the tube

<eltube name=”TheEltube” dx=”1.0” dy=”2.0” dz=”4.0” lunit=”mm”/>

hype

Definition of a tube with hyperbolic profile.

rmin: inner radius

rmax: outter radius
inst: inner stereo angle
outst: outter stereo angle
z: (half???) length

<hype name="TheHype" rmin="1.0" rmax="4.0" inst="" inst="" outst="" aunit="deg" lunit="mm" />

orb

Definition of a sphere.

r: radius

<sphere name="TheSphere" r="4.0" lunit="mm"/>

para

Definition of a paralelepipedon.

x,y,z: dimensions of the initial box.
alpha: alpha
theta: theta
phi: phi

<para name="ThePara" x="10" y="20" z="30" alpha="" theta="" phi=""/>

polycone

polyhedra

Definition of polyhedra with sections.

startphi: start angle of segment
deltaphi: size of angle-segment
numsides: number of sides of the polyhedra

<polyhedra name="ThePolyhedra" startphi="0" deltaphi="180" numsides="8" aunit="deg" lunit="mm">
<zplane/>
<zplane/>
<zplane/>
</polyhedra>

reflectedSolid

Definition of a general transformation for a solid. In the previous version of GDML it was the only way of using scaling transformation or reflection. In this version it is recommended using the scale transformation instead (Please refer to section …).

solid: the name of the solid to be transformed
sx,sy,sz: scaling factors
rx,ry,rz: rotational angles
dx,dy,dz: translation

<reflectedSolid name="TheReflected" solid="The" sx="-1" sy="1" sz="1" rx="0" ry="0" rz="0" dx="0" dy="0" dz="0"/>

sphere

Definition of a sphere.segment

rmin:	inner radius
rmax:	outter radius
startphi:	start
deltaphi:	delta
starttheta:	start
deltatheta:	delta

```
<sphere name=”TheSphere” rmin=”1.0” rmax=”2.0” startphi=”0” deltaphi=”270”
starttheta=”0” deltatheta=”90” aunit=”deg” lunit=”mm”/>
```

tessellated

Definition of a tessellated solid. A solid can be formed using either triangular or quadrangular faces. At first, all the vertices should be defined int he define section.

```
<define>
  <position name=”base0” x=”+20” y=”+20” z=”0” unit=”mm”/>
  <position name=”base1” x=”+20” y=”-20” z=”0” unit=”mm”/>
  <position name=”base2” x=”-20” y=”-20” z=”0” unit=”mm”/>
  <position name=”base3” x=”-20” y=”+20” z=”0” unit=”mm”/>
  <position name=”top” x=”0” y=”0” z=”20” unit=”mm”/>
</define>
```

Once the vertices are defined, they can be referenced in the tessellated solid. Obviously the triangular face has three attributes and the quadrangular face hase four attributes.

```
<solids>
  <tessellated name=”TheTessellated”>
    <triangular vertex1=”base0” vertex2=”base1” vertex3=”top”/>
    <triangular vertex1=”base1” vertex2=”base2” vertex3=”top”/>
    <triangular vertex1=”base2” vertex2=”base3” vertex3=”top”/>
    <triangular vertex1=”base3” vertex2=”base0” vertex3=”top”/>
    <quadrangular vertex1=”base3” vertex2=”base2” vertex3=”base1”
vertex4=”base0”/>
  <tessellated>
</solids>
```

tet

Definition of a tetrahedron. A tetrahedron is formed using four vertices. At first all four vertices should be defined in the define section:

```
<define>
  <position name=”base0” x=”0” y=”0” z=”0” unit=”mm”/>
  <position name=”base1” x=”8” y=”0” z=”0” unit=”mm”/>
  <position name=”base2” x=”0” y=”8” z=”0” unit=”mm”/>
  <position name=”top” x=”0” y=”0” z=”8” unit=”mm”/>
</define>
```

Once the vertices are defined, they can be referenced in the tetrahedron:

```
<solids>
  <tet name=”TheTet” vertex1=”base0” vertex2=”base1” vertex3=”base2”
vertex4=”top”/>
</solids>
```

torus

Definition of a torus segment.

rmin:	inner radius
rmax:	outter radius
rtor:	radius of the torus
startphi:	start angle of torus segment
deltaphi:	size of angle segment

```
<torus name=”TheTorus” rmin=”1.0” rmax=”2.0” rtor=”5.0” startphi=”0”
endphi=”90” aunit=”deg” lunit=”rad”/>
```

trap

Definition of a general trapezoid.

trd

tube

Definition of a tube segment.

twistedbox

twistedtrap

twistedtrd

twistedtubs

Structure

In this section we put together all the defined solids, materials etc.