The Virtual Geometry Model

Ivana Hřívnáčová
Institut de Physique Nucléaire, 91406 Orsay Cedex, France
E-mail: ivana@ipno.in2p3.fr

Brett Viren
Brookhaven National Lab, P.O. Box 5000, Upton, NY 11973-5000, USA
E-mail: bv@bnl.gov

Abstract. The Virtual Geometry Model (VGM) was introduced at CHEP in 2004 [1], where its concept, based on the abstract interfaces to geometry objects, has been presented. Since then, it has undergone a design evolution to pure abstract interfaces, it has been consolidated and completed with more advanced features. Currently it is used in Geant4 VMC for the support of TGeo geometry definition with Geant4 native geometry navigation and recently it has been used in the validation of the G4Root tool.

The implementation of the VGM for a concrete geometry model represents a small layer between the VGM and the particular native geometry. In addition to the implementations for Geant4 and Root TGeo geometry models, there is now added the third one for AGDD, which together with the existing XML exporter makes the VGM the most advanced tool for exchanging geometry formats providing 9 ways of conversions between Geant4, TGeo, AGDD and GDML models.

In this presentation we will give the overview and the present status of the tool, we will review the supported features and point to possible limits in converting geometry models.

1. Introduction
At present, there are several geometry modelers of interest for the physicists in HEP community:

- The geometry modeller [2] in Geant4 [3] - naturally utilized by many Geant4 users.
- The geometry modeller TGeo [4] in Root [5] - coming later, disconnected from any simulation tools. Used in the context of Virtual Monte Carlo [6].
- GDML (the geometry description markup language) [7] - an application-independent geometry description based on XML. Used mainly for providing persistence for Geant4 geometry model.
- AGDD (the Atlas geometry detector description) [8] - another geometry description based on XML. It was developed by the Atlas collaboration, however it was abandoned. It has been adopted and further developed in the STAR collaboration and recently adopted in the Daya Bay collaboration, in both for primary geometry implementation. Both GDML and AGDD formats are supported in the GraXML tool [9].
Each geometry model provides a set of tools for geometry verification and visualization. The VGM development aims to provide the user the possibility to use all the available tools regardless their primary geometry format.

2. Architecture

2.1. The VGM concept

The Virtual Geometry Model (VGM) has been developed as a generalization of the existing converters roottog4, g4toxml provided within Geant4 VMC [10], when new directions: g4toroot, roottoxml were asked for by users.

Instead of adding other one-way converters and multiplying the implementations, the abstract layer to geometry has been defined and the geometry models have been “mapped” to this generalized scheme. Once this is done, the geometry objects in these integrated models can be handled in the same way.

This new concept is demonstrated in Fig. 1

![Figure 1. The VGM concept](image)

2.2. Packages

All VGM packages are shown in Fig. 2 together with their dependencies on the external software.

Since the VGM first release, described already in detail at [1], the VGM design has been reviewed and the packages have been cleaned up from unnecessary dependencies. The VGM package now includes only the abstract interfaces and does not depend on any external package.
All common implementation has been moved from the VGM package in BaseVGM and the CLHEP dependent code in the utility package ClhepVGM.

The implementation of the VGM for a concrete geometry model represents a layer between the VGM and the particular native geometry. At present, the complete implementation is provided for the Geant4 [2], and the Root TGeo [4] geometry models, and there is now being added the third implementation for AGDD [8]. The dependence on the particular geometry models is hence restricted only to its VGM geometry specific package.

Besides the geometry model specific packages, the VGM provides also the XML exporter, which is included in its own package, XmlVGM and does not bring dependency on any external software.

With applying the new design the class name prefixes were replaced with namespaces. A unique namespace was introduced for each package, with the name equal to the name of the package.

2.3. Interfaces to geometry objects

The interfaces to geometry objects were defined with the intention to be suitable for a description of “geant-like” geometries with a hierarchical volume structure. The basic entities sufficient for such geometry description have been identified: solid, volume and placement to describe volumes hierarchies; and isotope, element, material and medium to describe material properties.

The solid and placement entities can have more specifications. In case of the solid it is reflected by introducing a specific interface for each solid type (box, tubs, cons, ...), while in the
placement case, both types introduced (simple placement and multiple placement) are described with a single interface.

The class diagram for the VGM implementation of the box object, see Fig. 3, demonstrates the mapping between the objects in the native geometry and the interfaces introduced in the VGM. The same approach has been applied to the other geometry entities. In case a common implementation of some functions (imposed by the interface) is wanted, an abstract base class providing this implementation is defined first in the BaseVGM package and the class in the VGM layer for a particular geometry model makes a specification of this abstract base class.

![Class Diagram](image)

**Figure 3.** The class diagram of the box solid in the VGM

### 2.4. Abstract factories

The VGM abstract factories, `VGM::IFactory` and `VGM::IMaterialFactory`, provide functions for geometry construction, import and export. While the functions for geometry construction and import are specific to a geometry and have to be provided by the geometry specific layers, the export function could be implemented in a common way and is provided in the BaseVGM package (classes `BaseVGM::VFactory`, `BaseVGM::VMaterialFactory`).

The class diagram for the VGM factories is shown in Fig. 4.

### 3. Use of VGM

#### 3.1. Geometry conversions

To convert native geometry from one geometry model to another, the geometry has to be first imported in the VGM (the native geometry objects are mapped to the VGM interfaces) using
the concrete VGM factory for this geometry model, and then exported using the VGM factory for the other geometry model.

At present, the implemented VGM packages provide the following ways:

- AGDD -> Geant4, Root, GDML
- Geant4 -> AGDD, Root, GDML
- Root -> AGDD, Geant4, GDML

The GDML import is not available.

In Example 1 the conversion of geometry from Geant4 to Root is demonstrated.

3.2. Use of VGM factory and interfaces

The VGM can be used to define geometry independently from a concrete geometry model. The abstract VGM factories provide a complete set of create functions which can be used to define user geometry. The concrete geometry model will then be chosen with the instantiation of the concrete factory.

The VGM interfaces to geometry objects make also possible to define a geometry application, eg. for visualization, based on the VGM, which can be then run with all supported geometry models.
**Example 1.** Converting geometry from Geant4 to Root via the VGM

```cpp
#include "Geant4GM/volumes/Factory.h"
#include "RootGM/volumes/Factory.h"
#include "TGeoManager.h"

// Import Geant4 geometry to VGM
Geant4GM::Factory g4Factory;
g4Factory.Import(physiWorld);
// where physiWorld is of G4VPhysicalVolume* type

// Export VGM geometry to Root
Root::Factory rtFactory;
g4Factory.Export(&rtFactory);
gGeoManager->CloseGeometry();
return rtFactory.World();
// returns Root top volume of TGeoVolume* type
```

3.3. Export to XML
The VGM geometry can be exported to XML in the AGDD [8] or GDML [7] format. Complying with the XML schema is embedded in the VGM XML exporter code itself, no external XML parser is then needed.
This is demonstrated in Example 2.

**Example 2.** Exporting geometry from the VGM factory in XML (AGDD, GDML)

```cpp
#include "XmlVGM/AGDDExporter.h"
XmlVGM::AGDDExporter agddExporter(&theFactory);
agddExporter.GenerateXMLGeometry();
// Export geometry to AGDD

#include "XmlVGM/GDMLExporter.h"
XmlVGM::GDMLExporter gdmlExporter(&theFactory);
gdmlExporter.GenerateXMLGeometry();
// Export geometry to GDML
```

4. Testing
The same simple geometry setups are defined via AGDD, Geant4, Root and VGM to test different aspects of the VGM: Solids, Placements, Reflections, Boolean Solids and Assemblies (see Fig. 5, Fig. 6, Fig. 8, Fig. 7 and Fig. 9).

The test program can be then configured by a list of arguments to select the input geometry type, the selected geometry setup and the destination geometry model or XML output. It includes also options to run a test with Geant4 tracking and with G4Root navigation [11].
Figure 5. The geometry setup to test solids. It includes all solids supported in VGM. The bottom row (from left to right): box, elliptical tube, polycone, sphere, generic trapezoid, tube, extruded solid. The upper row: cone, parallelepiped, polyhedra, torus, trapezoid, cut tube, extruded solid with adjacent z-sections.

Figure 6. The geometry setup to test placements. A box with division including one more box layer is placed in a regular pattern.

Figure 7. The geometry setup to test Boolean solids. The boolean intersection, subtraction and union is applied to box and cone solids with the same displacement transformation.

Figure 8. The geometry setup to test reflections. All solids from the Solids test are placed once more with reflection.

Figure 9. The geometry setup to test assemblies. Taken from Root tutorials.

The testing procedure has been automated in three suite shell scripts, which execute the test program with all possible combinations of parameters and generate the output that can be compared with the reference output. The following test suites are available:

- Suite 1 - geometry import and export with verbosity output
• Suite 2 - XML export from all possible inputs
• Suite 3 - Geant4 tracking with Geant4 native navigation and G4Root navigation with verbose output. The GPS [12] source of geantinos tuned for each geometry setup with a fixed random number seed is used.

The test program can be also run with a user defined geometry and the comparison of the outputs from tracking with Geant4 native and G4Root navigation turned out to be very efficient for geometry debugging.

5. Examples
As the test program, being written for the purpose of testing all aspects, is rather complex, five simple examples demonstrating use of the VGM for converting native geometries are provided. The summary of the use cases demonstrated in these examples is given in Table 3.

| Use case | Geometry source                                    |
|----------|----------------------------------------------------|
| E1       | G4 -> Root                                         |
| E2       | Root -> G4                                         |
| E3       | G4 -> XML                                          |
| E4       | Root -> XML                                        |
| E5       | AGDD -> Root                                       |

| Use case | Geometry source                                    |
|----------|----------------------------------------------------|
| E1       | Geant4 novice example N03                          |
| E2       | Root file with geometry generated in E1            |
| E3       | Geant4 novice example N03                          |
| E4       | Root tutorial macro rootgeom.C                    |
| E5       | AGDD test file                                     |

6. Building systems
The VGM provides three independent building systems which makes it easy to include it in various frameworks:

• The first building system is based on GNU makefiles [13], adapted from Geant4 [3].
• CMT [14] - The VGM has been ported to CMT thanks to Laurent Garnier, LAL, who takes care of maintenance of the CMT requirements files and as well as of testing on MacOS platform.
• Autoconf [15] - the autoconf building system was introduced with adding the AgddGM package.

7. Present status
7.1. Features supported in VGM
The VGM supports most of advanced features present in both Geant4 and Root geometry models:

• Large number of solids - all CSG solids and number of specific solids in Geant4 and their counterparts in Root
• Boolean solids (Geant4) and composite shapes (Root)
• Reflected solids (Geant4) and positioning with reflection (Root)
• Multiple placements - replicas, divisions (Geant4), divisions (Root)
• Assemblies (Root)

Not supported:
• “Exotic” solids - solids that have no counterpart in the other geometry model or recently added solids not yet required by any user
• Parameterised volumes (Geant4)
• Positions with the “MANY” option (Root)

7.2. Present status of packages
• Geant4GM, RootGM - Both VGM implementations for Geant4 and Root geometry models are complete, all VGM interfaces are implemented.
• AgddGM - The implementation started by the end of 2006. The interfaces to the geometry objects and the factory import function are implemented, the factory create functions, used in export, still have to be developed. It should be noted that the implementation is based on the AGDD v6, what was decided by Daya Bay collaboration, while the AGDD exporter is based on the newer, AGDD v7 version.
• XmlVGM - XML exporter
  – GDML - All solids are supported, missing export of reflections and multiple placements.
  – AGDD - All features are implemented, only some solids are not supported in AGDD.

The current version of the VGM tool is available at [16].

8. Conclusions
The VGM introduces a general approach for access to geometries of specific geometry models (Geant4, Root TGeo, AGDD, GDML) and for their conversion. This gives a possibility for a user of one specific package to use the tools supported by other packages. It also provides a gateway for a geometry based application independent from a concrete geometry model.

The VGM allows the user first to define the geometry independently from a concrete geometry model, and then to choose the concrete geometry model at run time, though this use case was not the primary goal of this tool.

The VGM is used in Geant4 VMC [10] to support user geometry defined via Root geometrical modeller with Geant4 native navigation. It has been also used in G4Root [11] validation on a subset of the ALICE geometry.

References
[1] Hrivnáková I 2004 Proc. of Computing in High Energy and Nuclear Physics (Interlaken) Id 387
[2] Cosmo G. et al 2006 Proc. of Computing in High Energy and Nuclear Physics (Mumbai) Id 257
[3] Agostinelli S et al 2003 Nucl. Instrum. and Methods A506 250-303
[4] Brun R, Gheata A and Gheata M 2003 Proc. of Computing in High Energy and Nuclear Physics (La Jolla) Id THMT001
[5] http://root.cern.ch
[6] Hrivnáková I et al 2003 Proc. of Computing in High Energy and Nuclear Physics (La Jolla) Id THJT006
[7] Chytracek R 2001 Proc. of Computing in High Energy and Nuclear Physics (Beijing) Id 8-009
[8] Arnault Ch et al 2001 Proc. of Computing in High Energy and Nuclear Physics (Beijing) Id 8-001
[9] Hrivnácková J 2003 Proc. of Computing in High Energy and Nuclear Physics (La Jolla) Id THJT009
[10] Hrivnáčková I 2007 Computing in High Energy and Nuclear Physics Id 126
[11] Gheata A and Gheata M 2007 Computing in High Energy and Nuclear Physics Id 38
[12] http://reat.space.qinetiq.com/gps/
[13] http://www.gnu.org/software/make/
[14] http://www.cmtsite.org/
[15] http://www.gnu.org/software/autoconf/
[16] http://ivana.home.cern.ch/ivana/VGM.html