Journal of Physics: Conference Series **331** (2011) 032035 doi:10.1088/1742-6596/331/3/032035

CATIA-GDML geometry builder

S Belogurov[1,2], Yu Berchun[2], A Chernogorov[1], P Malzacher[3], E Ovcharenko[1,2], A Semennikov[1]

[1] Institute for Theoretical and Experimental Physics,
B. Cheremushkinskaja 25, 117218, Russia, Moscow, belogurov@itep.ru
[2] Bauman Moscow State Technical University,
2-nd Baumanskaja str. 5,105005, Russia, Moscow, cad@bmstu.ru
[3] GSI Helmholtzzentrum für Schwerionenforschung GmbH,
Planckstr. 1, 64291, Germany, Darmstadt, P.Malzacher@gsi.de

Abstract. Due to conceptual difference between geometry descriptions in Computer-Aided Design (CAD) systems and particle transport Monte Carlo (MC) codes direct conversion of detector geometry in either direction is not feasible. An original set of tools has been developed for building a GEANT4/ROOT compatible geometry in the CATIA CAD system and exchanging it with mentioned MC packages using GDML file format. A Special structure of a CATIA product tree, a wide range of primitives, different types of multiple volume instantiation, and supporting macros have been implemented.

1. Introduction

Transfer of the detector geometry from Computer-Aided Design (CAD) systems to particle transport Monte Carlo (MC) codes like GEANT4 (G4) [1] and ROOT [2] is always an issue due to geometry description incompatibility.

Traditionally, the geometry for MC simulations is quite simple at the design and optimization stage. It reflects not a detailed, but a conceptual layout of the setup. When the design is finished, the detailed engineering drawings are used for manual creation of more precise geometry description. The last operation is very laborious and can hardly be repeated several times. However, for modern experiments working in severe radiation environment the design optimization process should be iterative.

In principle, it is possible to generate automatically a tessellated solid from any CAD model. The produced geometry can be imported then into G4/ROOT. Such an approach is realized e.g. in the FASTRAD package [3]. However, due to low simulation speed for big assemblies and complicated shapes, direct automated transfer via tessellated solids has limited applicability.

We are presenting a set of tools, which allows to facilitate significantly creation of a G4/ROOT-compatible geometry from the CAD system CATIA v.5. The geometry is exchanged via Geometry Description Markup Language (GDML) [4]. CATIA v.5 is widely used in big physical laboratories like CERN, GSI, LNGS, etc. The tools are being developed for upcoming upgrade of LHC experiments and for experiments at the future Facility for Antiproton and Ion Research (FAIR).

2. Geometry representation in CAD and GEANT4/ROOT

The difference between the geometry representations in the CAD systems and G4/ ROOT is twofold: in the description of solid bodies and in the hierarchy of assemblies. Beside very specific method of the geometry description and the product structure in G4/ROOT, there are some limitations from the side of computational resources. Preparing any geometry for MC simulations one has to keep in mind other two issues: required level of detail and simulation optimization

2.1. Geometry of a solid

In CAD systems solid bodies are built using a wide class of surfaces and curves in advanced boundary representation (BRep).

In G4/ROOT the Constructive Solid Geometry (CSG) is used. Fundamental building blocks for CSG are the so called primitives - 3D objects described by a minimal set of parameters necessary to define the shape and size of the solid.

Simple solids can be combined using Boolean operations. Creating such a new Boolean solid requires: two solids (operands), a Boolean operation*: union, intersection* or *subtraction*, optionally a transformation (*scale, rotation*, and *translation*) for the second solid.

An operand for a Boolean operation should be either primitive or another Boolean solid: the result of a previous Boolean operation.

2.2. Assembly hierarchy

In CAD software there is a minimal indivisible unit – a solid body (*part)* and a conceptually unlimited number of nested assemblies (*products*). Products and subproducts are only logical units – all the materials are assigned to solid bodies inside the part files or to parts themselves.

In G4/ROOT detector geometry is made of a number of volumes. The largest one is called the *World*. It contains all other volumes in the detector geometry. The other volumes are created and placed inside previous (mother) volumes, included in the *World* volume. Each volume is created by describing its shape and its physical characteristics, and then placing it inside a containing volume. To describe a volume's shape, the concept of a *solid* is used. To describe a volume's full properties, one uses a *logical volume*. It includes the geometrical properties of the solid, and adds physical characteristics: the material of the volume, whether it contains any sensitive detector elements, the magnetic field, etc. To define the position of the volume one should create a physical volume – a copy of the *logical volume* put inside a larger containing volume. Important is that unlike a *part* in the CAD systems, any logical volume may be a mother for placing smaller volumes inside. The *World* is a special *logical volume*, because it can not be positioned.

2.3. Required level of detail

A G4/ROOT model requires normally only details perceivable by the detectors. For a detector with poor position resolution, the description should be less accurate than for a more precise device. Choosing an appropriate level of detail, one can reduce significantly both required computational time and memory. E.g. threaded hole in a flange with a screw inside is equivalent for MC to a bulk piece of metal; for peripheral equipment, sometimes, just a simple solid filled with a correct mixture of materials is sufficient. For CAD models, instead, all the details are essential.

2.4. Simulation optimization

There are various tricks allowing to optimize simulations and reduce geometry description file. E.g. description of a sampling calorimeter [5] will be more compact and loading of the geometry faster if scintillator plates are inserted into the bulk lead mother volume instead of building a stack of alternating lead and plastic plates; putting several simple solids of the same material next to each other instead of using union helps to accelerate simulations, etc.

3. An approach to G4 geometry building from inside CATIA

The goal of the CATIA-GDML geometry builder is to make manual geometry transfer from CATIA to GEANT4 more convenient and fast than before. We employ powerful measurement, analysis, and design features of various CATIA workbenches to create a MC-compatible representation of an existing engineering assembly. Manual geometry building allows to have enough flexibility for creation of the geometry optimal for fast and efficient simulation.

The workflow includes the following steps. Apprehension of the CSG representation for an existing part; creation of auxiliary geometrical elements (planes, lines, point) in the existing model; measurement of parameters for the CSG solid; creation of CSG solid, assigning material and positioning it; export of GDML.

The tool is targeted on scientists who understand what geometry representation and level of detail are optimal for a given simulation task. For usage of the tool they should get familiar with only limited functionality of CATIA and our plugins.

3.1. Implementation of primitives and logical volumes

The following approach is realized in the geometry builder. For the most widely used G4/ROOT primitives we built parameterized User Defined Features (UDF) in CATIA. UDFs for G4Cons and G4Torus primitives are shown as an example in the figure 1.

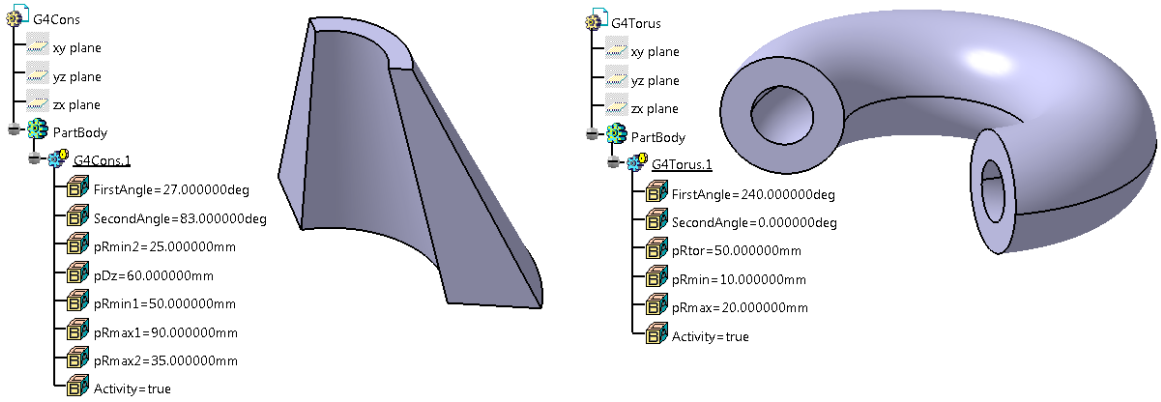

Figure 1. CATIA realization of G4Cons and G4Torus.

We developed a data structure of the CATPart file which allows for implementation of the G4/ROOT volume hierarchy i.e. concepts of logical, physical, mother volumes, and materials. The file structure contains a *Partbody* which is a CSG solid with a material from *G4Material* catalog assigned – this stands for *logical volume*. A publication of the *Partbody* keeps neither geometry parameterization nor material. It serves for insertion into bigger (mother) *logical volume*. Publications of smaller volumes from other files can be inserted into CATPart file as additional bodies with positioning; it is an analogue of *physical volume*. See e.g. trees in the figures 2-4.

3.2. Implementation of physical volumes and multiple instantiation

We developed a set of macros enabling positioning and replication of the G4/ROOT-like CAD volumes in the corresponding mother volumes. A multilevel G4/ROOT-like assembly can be built in the following way. First for each volume a *CATPart* file with parameterized solid in the *Partbody*, assigned material, and publication of the *Partbody* should be created. Then all those files should be loaded as components into a *CATProduct* file. The last step is to run inserter or replication macros for placing volumes into their mothers. Positioning template will be generated automatically. The procedure is illustrated in the figures 2-5. Note bright and pale icons in the trees corresponding to shown and hidden objects.

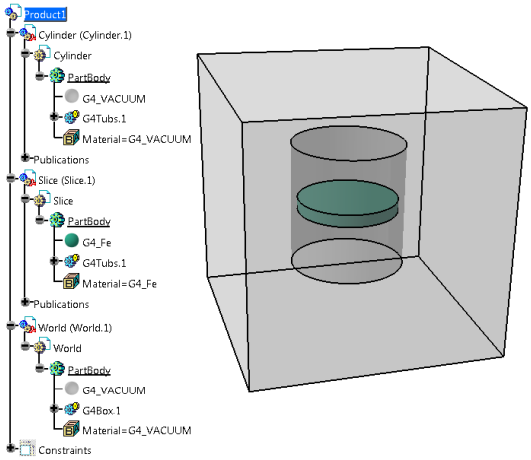

Figure 2. Three logical volumes in a product

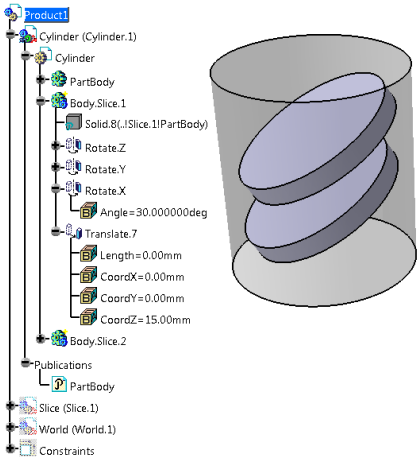

Figure 3. Inserting and positioning two slices in the cylinder

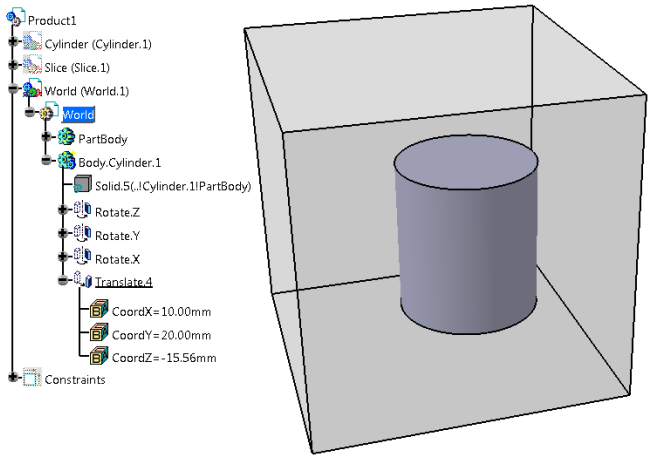

Figure 4. Positioning the cylinder in the *World*

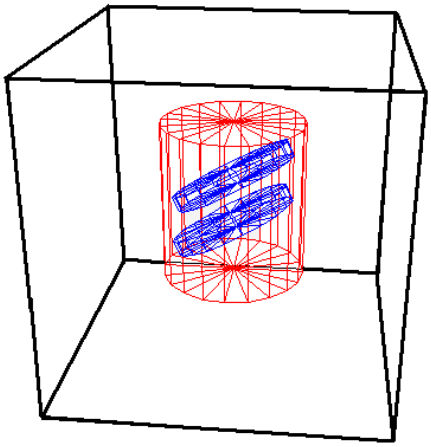

Figure 5. Exported geometry read by ROOT

We have implemented 3 methods of multiple instantiation of volumes.

3.2.1. Replica

This type corresponds to GEANT4 *Replica* or ROOT *Division*s. A user has to create 2 volumes: a mother volume, that will be divided, and a "slice", that will be put into the mother volume several times. Replicated slices must fill the mother volume with no gaps. Linear and circular replications are available via CATIA "*Pattern*" feature. Geometry of slices has certain limitations. The tool is able to calculate parameters of either mother or slice.

3.2.2. Array

In the case of array "Pattern" feature is also applied to the body, but the difference between Replica and Array is that when converting to GDML not <*replicavol*>, but a list of simple physical volumes will be created. This method is useful when geometrical restrictions on the slice shape can not be observed or gaps can't be avoided. For this and the next method a special macro "Array" has been created.

3.2.3. SimplePlacementArray

This method is designed for a case when a subarray should be replicated but a volume containing a subarray can not be built without violation of the G4/ROOT rules. The method puts all the children

from subarrays directly into common mother. In CATIA the container for subarray plays the role of some reference frame, in which it is comfortable to position the children.
All the methods are illustrated by the figure 6.

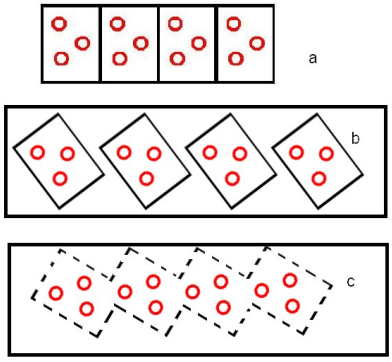

Figure 6. Different methods of multiple instantiation: a) *Replica*; b) *Array*; c) *SimplePlacementArray*.

Building a G4/ROOT representation of given CAD parts and products, the user can choose the best way of a solid description and the assembly hierarchy from point of view of the simulation performance.
At the end the entire model tree is analyzed and exported into GDML. An example of geometry transferred via GDML into GEANT is shown in the figure 5.
In addition, we found that the opposite direction which makes the design loop closed is also requested by the community. Following that demand we implemented also inverse operation i.e. import of GDML files into CATIA. This direction is useful for shifting from physical justification of a conceptual model to engineering design.

4. Example: crystal array for the calorimeter of R3B experiment at FAIR.

The unparameterized CAD model of the crystal array of the R3B [6] calorimeter was transferred to CATIA via STEP [7] format. The calorimeter can be considered as a circular array of 40 vertical slices. Each box is a carbon fiber composite (CFC) housing with 4 crystals. There are 6 types of boxes and 12 types of crystals. The G4/ROOT-like models of boxes (*Fiber_box_barrel*)* with crystals were built using our tool. Asterisks stand for various types of boxes and crystals. A vacuum trapezoid *Slice_container* filled with 24 *Fiber_box_barrel**-es of different types is used for representing the vertical slice. A circular pattern made of the slices represents entire crystal array. G4/ROOT-like representation of a slice of CFC boxes and entire crystal are shown in the figure 7. All the slices are written then into GDML file as single placements because they do not have a shape suitable for *G4Replica*.

5. Plans for further development

Further developments of the Geometry Builder are foreseen in the following directions:
- Enhancement of the set of implemented primitives
- Improvement of the G4Materials catalog in CATIA
- Implementation of checkers for CSG tree structure and volume overlaps
- Adaptation of the CATIA Digital Mockup (DMU) optimizer for automatic fit of parameterized CSG models to existing parts
- Case study and best practice elaboration

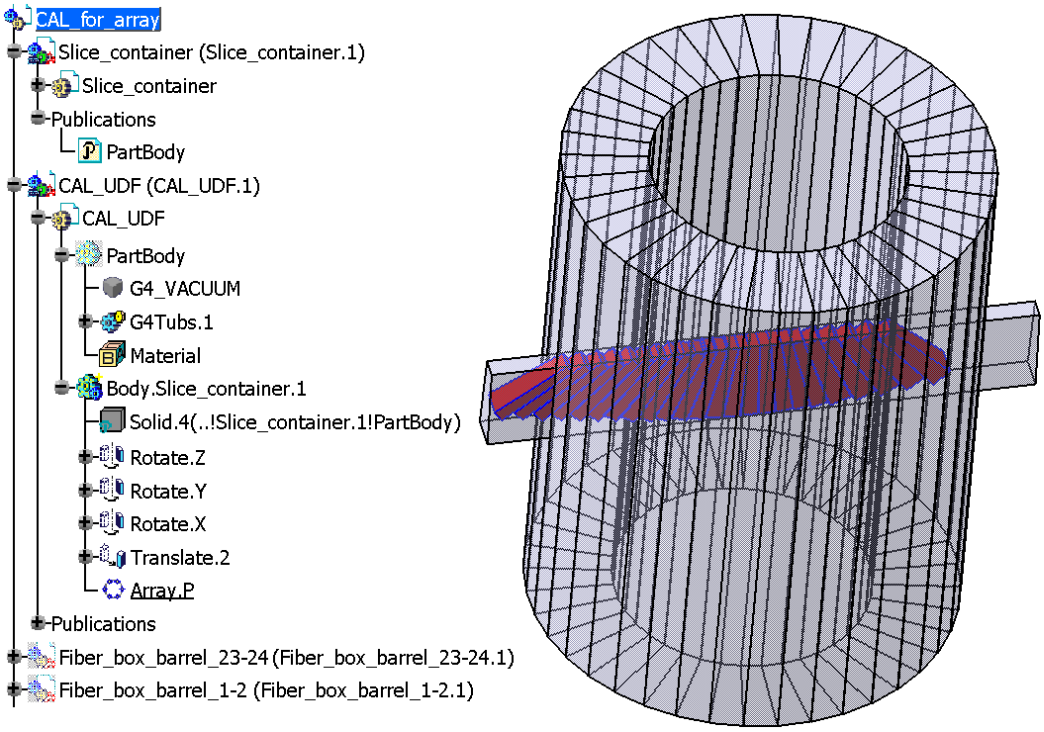

Figure 7. G4/ROOT-like representation of a slice of CFC boxes and entire crystal array

6. Conclusions

A set of tools for building simulation-optimized GEANT-like geometry from inside CATIA CAD system is developed. The geometry building process is facilitated significantly with respect to traditional approach. The geometry can be exchanged with Physical Monte Carlo via GDML files in both directions. The tools allow iterative optimization of complex systems in harsh radiation environment. The tools are developed for the upgrade of LHC experiments and for the experiments at future FAIR complex. The tools can be useful also for space and medical applications.

7. References

[1] Allison J et al. 2006 Geant4 developments and applications *IEEE Trans. on Nucl. Sci.* **53** 270-8
[2] Naumann A, Offermann E, Onuchin V, Panacek S, Rademakers F, Russo P and Tadel M 2009 ROOT – a C++ framework for petabyte data storage, statistical analysis and visualization *Computer Physics Communications* **180** 2499-512
[3] http://www.fastrad.net/
[4] Chytracek R, McCormick J, Pokorski W and Santin G 2006 Geometry description markup language for physics simulation and analysis applications *IEEE Trans. Nucl. Sci.* **53** 2892-6
[5] Machefert F and Martens A 2010 Overview of the LHCb calorimeters *Nucl. Instrum. Meth.* **617** 40-4
[6] http://www-win.gsi.de/r3b/Documents.htm
[7] http://www.iso.org/iso/iso_cafe_step.htm

Acknowledgements

The authors are grateful to Drs. F. Carminati, R. Brun, A. Gheata (CERN), and D. Bertini (GSI) for stimulating interest and discussions and to A. Markin (BMSTU) for his contribution into the project. The work was supported by INTAS grant 06-1000012-8778 and by FAIR-Russia Research Center sponsored by Rosatom..