TGeoCad: an Interface between ROOT and CAD Systems

C Luzzi [1] [2] **and F Carminati** [2]

[1] University of Ferrara, Via Giuseppe Saragat, 1, 44020 FE, Italy
[2] CERN, 1211 Geneva 23, Switzerland

E-mail: `cluzzi@cern.ch`

Abstract. In the simulation of High Energy Physics experiment a very high precision in the description of the detector geometry is essential to achieve the required performances.
The physicists in charge of Monte Carlo Simulation of the detector need to collaborate efficiently with the engineers working at the mechanical design of the detector. Often, this collaboration is made hard by the usage of different and incompatible software. ROOT is an object-oriented C++ framework used by physicists for storing, analyzing and simulating data produced by the high-energy physics experiments while CAD (Computer-Aided Design) software is used for mechanical design in the engineering field.
The necessity to improve the level of communication between physicists and engineers led to the implementation of an interface between the ROOT geometrical modeler used by the virtual Monte Carlo simulation software and the CAD systems.
In this paper we describe the design and implementation of the TGeoCad Interface that has been developed to enable the use of ROOT geometrical models in several CAD systems. To achieve this goal, the ROOT geometry description is converted into STEP file format (ISO 10303), which can be imported and used by many CAD systems.

1. Introduction

The incompatibility between the ROOT [1] geometrical modeler and the CAD systems comes from the different representation of the geometrical shapes.

In a CAD system, the shapes are treated as *boundary represented objects* (BREPS), a BREPS is defined by the description of its boundaries. Boundary representation models are based on two elements: topology and geometry. The topology gives information about the relationships between the main topological items, i.e. faces (bounded portion of a surface), edges (bounded piece of a curve) and vertices (points). The geometry describes the integration of these elements in space. Each solid is placed by a translation/rotation matrix in an assembly of many solids and the different assemblies form a complete hierarchy of geometrical objects.

On the other side, in ROOT the Constructive Solid Geometry method is adopted according to which the solids are defined directly as three-dimensional primitives and created out of a set of simple shapes like cuboids, cylinders, prisms, pyramids, spheres. Complex shapes are created by using boolean operation between simple shapes. Each shape can then be placed by translation/rotation matrices in assemblies. For more information about solid modelling see [2].

Another substantial difference between ROOT and CAD systems is the hierarchy of assemblies.

In a CAD system the basic unit is a solid part that can have an arbitrary number of nested assemblies.

A ROOT geometry is composed by a number of volumes. The largest one is used as a container for all other volumes in the detector geometry. Each volume created is placed inside another one, called the mother volume. A volume is composed by its shape and all its physical characteristics as the material of the volume, whether it contains any sensitive detector elements, the magnetic field, etc [3].

The main task of the TGeoCad interface is to analyse the ROOT shape representation, extract all the information about the solid and reproduce it in a BREPS format using the OpenCASCADE Technology [4] libraries which provide also the possibility to write BREPS assemblies in a STEP file.

2. The Open CASCADE Technology

The Open CASCADE Technology (OCCT) is an open source software development platform written in C++. It has been designed for rapid production of sophisticated domain-specific design applications.

OCCT allows to develop application dealing with two or three-dimensional (2D or 3D) geometric modeling in general-purpose or specialized CAD systems, manufacturing or analysis applications, simulation applications, or illustration tools.

The OCCT libraries are grouped into six modules, those used by the TGeoCad interface are:

- *Foundation Classes:* it is used for the description of elementary geometric shapes (points, vectors, lines, circles, conics, planes), for the positioning of geometries in space or on a plane by means of an axis or a coordinate system and for the definition of geometric transformations such as translation, rotation and symmetries;
- *Modeling Data:* it allows building pure topological data structures defining relationships between simple geometric entities and giving the possibility to assign to a shape an orientation and a location. Using the abstract topological data structures each shape can be divided into various topological components as shown in Fig.1:

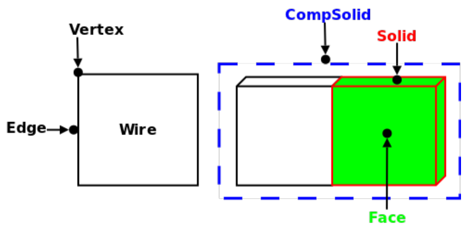


Figure 1. Topological data structures

- *Modeling Algorithm:* it provides geometric and topological algorithms for creating vertices, edges, faces and solids. It gives the possibility to directly build primitive objects such as boxes, wedges and rotational objects. Moreover, sweeping and Boolean operations are also provided.
- *Extended data exchange (XDE) module:* it allows writing applications based on the Open CASCADE Technology in the IGES and/or STEP (AP203, AP214) file formats.
- *Open Cascade Application Framework (OCAF):* it is a Rapid Application Development (RAD) framework used for specifying and organizing application data according to which data structure are organized by mean of a reference model. The reference can point to any

type of data in the application and it is implemented in the form of labels. Application data such as a shape itself or a shape location are attached to these labels as attributes. The OCAF application is used to manage an XDE document which is a container for the reference keys composed by a set of labels organized in a tree structure. Each label has a tag expressed as an integer value and it is identified by a string build by concatenation of tags from the root of the tree, e.g. [0:1:2] (see Fig.2).

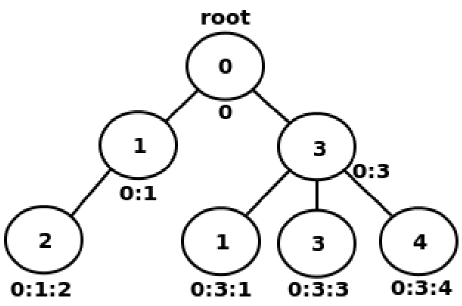


Figure 2. The labels tree

3. The STEP Standard

STEP stands for STandard for the Exchange of Product model data and it is the ISO 10303 standard intended to handle a wide range of product-related (electronic, electro-mechanical, mechanical, architectural, process plant, furniture, etc) data covering the entire life-cycle of a product (design, analysis, planning, manufacture).

The STEP format was designed to meet the need of having a common set of CAD/CAM tools within most companies using a neutral format for the file exchange in order to reduce the number of needed translators.

The components of the STEP standard are divided into several series of parts. Each part series contains one or more types of ISO 10303 parts. The OpenCascade technology gives the possibility to create files according to the STEP parts AP203 and AP214.

The AP 203 specifies the application protocol for configuration controlled three-dimensional design. It is related to the transfer of product shape models, assembly structure, and configuration control information such as part versioning and release status. Using AP 203 assemblies of positioned and oriented part models can be exchanged.

The AP 214 specifies core data for automotive mechanical design processes. It is an extension of AP203 and it also manages manufacturing tools, tolerance data, numerical control data, data for kinematic simulation, 2D drawings, and product data management information.

The STEP format is a neutral format well supported by almost any 3D CAD software. It provides a mechanism to describe product data independent from any particular system. For more information about the STEP standard see [5].

4. Details of the conversion

TGeoCad has different way to reproduce a ROOT shape in the OpenCASCADE platform:

- Shapes created step by step starting from points such as box, parallelepiped, trapezoid etc.
  - For example the ROOT TGeoTrd1 is translated creating edges from points, wire from edges, faces (planar surfaces) from wires, shells from faces and solid from shells;

- Shapes created using OCCT capabilities for solid primitives creation and boolean operations such as tube, cone, sphere, etc.
  - As shown in the Fig.3, the TGeoCone is created using the OCCT libraries which giving the height and radius information allow to create an inner and outer cones and to substract the inner cone from the outer cone.

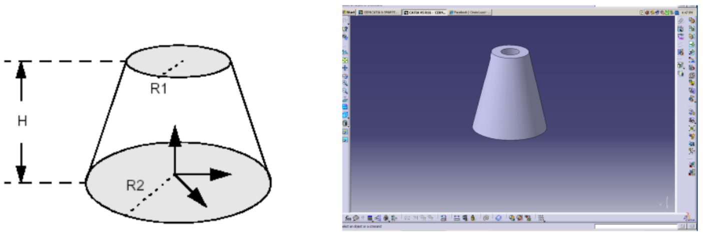


Figure 3. a. OCCT cone representation; b. A TGeoCone translated in a STEP file and loaded in CATIA.

- Shapes created by using modeling algorithm (extrusions, revolutions, lofts) applied to basic geometries such as the hyperboloid;
- TGeoCompositeShape created using OCCT Boolean operations between two or more shapes.

The conversion method is selected according to the type of each shape. When it is possible we use directly the OCCT libraries for solid primitives, sometimes combined with the OCCT modeling algorithms. When there is not a suitable OCCT library for the creation of a specific solid we create it starting from points.

5. Structure of the interface

Three main classes compose the TGeoCad interface:

- *TGeoToStep*: a TGeoObject based class that takes the TGeoManager pointer and creates the geometry step file;
- *TGeoToOCC*: implements all the methods shown in the Table n.1 to translate each ROOT shape in the corresponding OCCT shape;

Table 1. Methods implemented to traslate ROOT shapes in OCCT shapes.

ROOT	TGeoToOCC	ROOT	TGeoToOCC
TGeoBBox(..)	OCC_Box(..)	TGeoTrd2(..)	OCC_Trd(..)
TGeoSphere(..)	OCC_Sphere(..)	TGeoTubeSeg(..)	OCC_Tube(..)
TGeoArb8(..)	OCC_Arb8(..)	TGeoCtub(..)	OCC_Cuttub(..)
TGeoConeSeg(..)	OCC_Cones(..)	TGeoTube(..)	OCC_TubeSeg(..)
TGeoCone(..)	OCC_Cones(..)	TGeoPcon(..)	OCC_Pcon(..)
TGeoPara(..)	OCC_ParaTrap(..)	TGeoTorus(..)	OCC_Torus(..)
TGeoGtra(..)	OCC_ParaTrap(..)	TGeoPgon(..)	OCC_Pgon(..)
TGeoTrd1(..)	OCC_Trd(..)	TGeoEltu(..)	OCC_Eltu(..)
TGeoHype(..)	OCC_Hype(..)	TGeoXtru(..)	OCC_Xtru(..)
TGeoComposite(..)	OCC_Composite(..)		

- *TOCCToStep*: reproduces the ROOT geometry tree (mother-children relationship) in the XDE document and writes it to the STEP file:

– As first step an OCAF application is created in order to manage the XDE document, afterwards shapes are created and memorized on the XDE document without reporting any information about the relathionship between shapes.
As shown in Fig.4, starting from the top of the ROOT geometry tree the *OCCShapeCreation(..)* method translates each ROOT shape in the OCCT version using all the methods shown in the Table n.1.
For each translated shape a new label is created and memorized into the XDE document. The correspondence shape-label is also stored in a map of volumes and labels in order to keep memory of the shapes already translated.
Therefore, a check is done for each shape taken from the root file. If the shape-label correspondence is present in the map it means that the shape has already been translated then the label is copied from the map to the XDE document using the up-to-date location of the current root shape. If there is no correspondence present in the map, the shape needs to be translated.

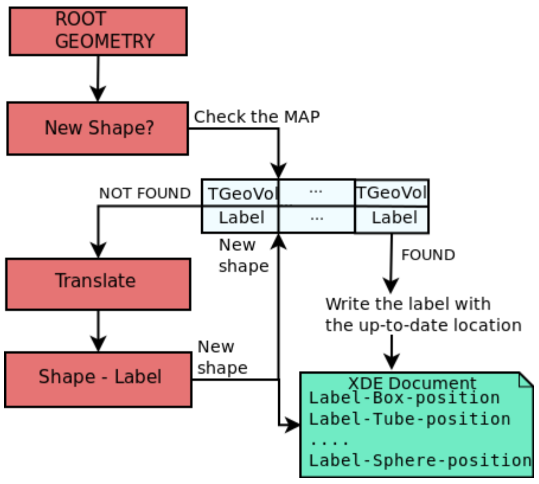


Figure 4. The OCCShapeCreation method

– As second step, Fig. 5 shows the *OCCTreeCreation(..)* method which starts to browse the geometry and for each node from the end to the top of the ROOT physical tree the mother and daughter label references are found in the map. In addition the daughter location matrix is obtained from the root file. These parameters are used to calculate the position of the daughter with respect to the mother. The connection between the daughter label and the mother label is then created, resulting in a new label, which is added to the XDE document.
The OCCT shapes are added, step by step, to an assembly entity where each of its

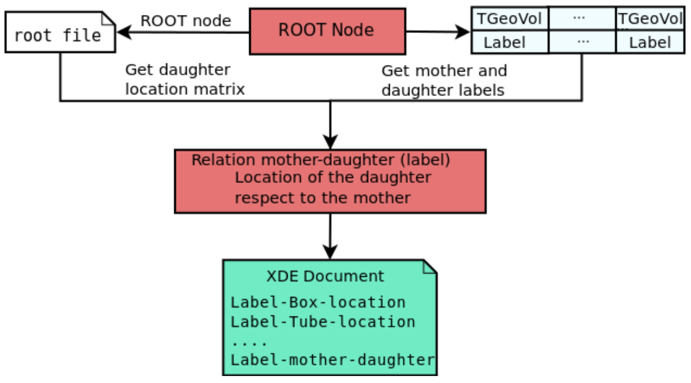
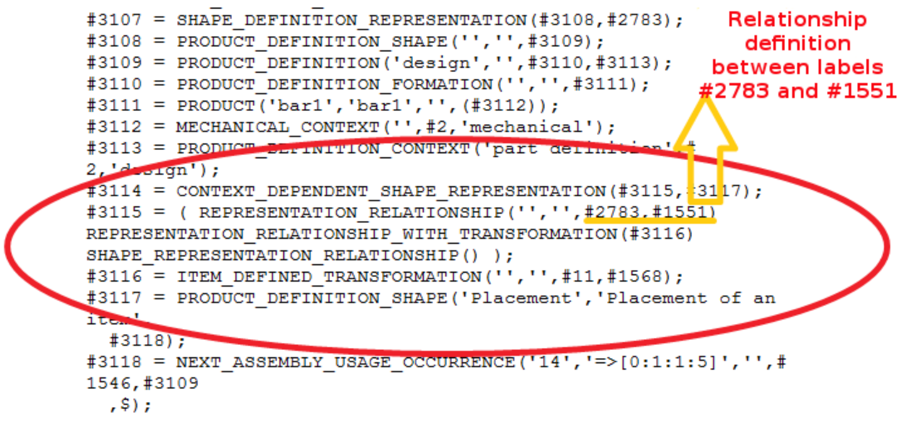

root file
ROOT node
ROOT Node
TGeoVol
Label
...
TGeoVol
Label
Get daughter location matrix
Get mother and daughter labels
Relation mother-daughter (label) Location of the daughter respect to the mother
XDE Document
Label-Box-location
Label-Tube-location
....
Label-mother-daughter

Figure 5. The OCCTreeCreation method

sub-shapes define a sub label therefore each node of the assembly refers to its subshapes.

– At last, the XDE document containing the whole geometry is written to the STEP file. An example of STEP file is shown in the Fig. 6



```
#3107 = SHAPE_DEFINITION_REPRESENTATION(#3108,#2783);
#3108 = PRODUCT_DEFINITION_SHAPE('','',#3109);
#3109 = PRODUCT_DEFINITION('design','',#3110,#3113);
#3110 = PRODUCT_DEFINITION_FORMATION('','',#3111);
#3111 = PRODUCT('bar1','bar1','',(#3112));
#3112 = MECHANICAL_CONTEXT('',#2,'mechanical');
#3113 = PRODUCT_DEFINITION_CONTEXT('part definition',#
2,'design');
#3114 = CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#3115,#3117);
#3115 = ( REPRESENTATION_RELATIONSHIP('','',#2783,#1551)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#3116)
SHAPE_REPRESENTATION_RELATIONSHIP() );
#3116 = ITEM_DEFINED_TRANSFORMATION('','',#11,#1568);
#3117 = PRODUCT_DEFINITION_SHAPE('Placement','Placement of an
item',
  #3118);
#3118 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('14','=>[0:1:1:5]','',#
1546,#3109
  ,$);
```

Figure 6. Relationship between two shapes reproduced in the STEP file.

Fig. 7 shows the whole architecture of the TGeoCad interface.

6. Technical details

In order to have the TGeoCad interface available in ROOT, OpenCascade must be installed and ROOT needs to be compiled using the configuration options:

```
./configure --enable-geocad;
--with-occ-incdir:  location of OpenCascade inc files
--with-occ-libdir:  location of OpenCascade lib files
```

The TGeoCad has to be used from ROOT by loading a geometry in memory, creating an object of the TGeoToStep class and calling the method CreateGeometry(). A STEP file called geometry.stp containing the geometry will be created on the current directory.

```
root[0] gSystem->Load("libGeoCad.so");
root[1] .x roottest.C
```

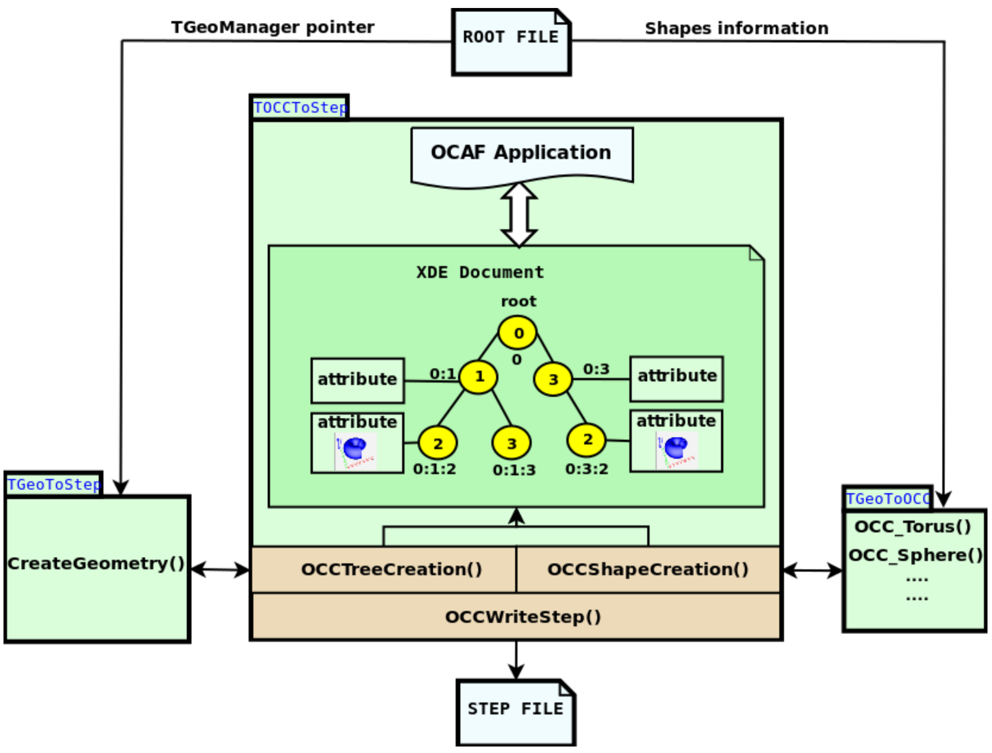


Figure 7. The TGeoCad architecture

```
root[2] TGeoToStep *myStep = new TGeoToStep (gGeoManager);
root[3] myStep->CreateGeometry();
```

7. Conclusion

The table 2 reports the number of volumes and nodes of the ROOT geometry, the dimensions of the ROOT files and of the related STEP files, as well as the execution time of each conversion.

Table 2. Information related to the conversion of several geometries.

Nr. Volumes	Nr. Nodes	ROOT file	STEP File	Execution time
13	485	5.3 KB	143.1 KB	0.18 s
90	184	19.7 KB	1.4 MB	792.53 s
645	646	36 KB	10.3 MB	10.27 s
936	936	13.1 KB	15.6 MB	16.22 s
7143	29046966	328.2 KB	13.9 MB	6303.58 s

The execution time is not directly proportional to the number of volumes or to the dimension of the root file. We can assert that, in the majority of the cases, the execution time depends on the number of boolean operations required to reproduce the geometry. These boolean operations are time consuming in OpenCascade.
The dimension of the STEP file depends on the size of the ROOT file and on how many times a same node is repeated in the geometry. Every time that a node is repeated in the ROOT geometry, the implementation of the shape does not take place but only the reference to the object is re-written in the file by modifying the location.

Acknowledgements
We would like to thank Prof. Eleonora Luppi, Dr. Latchezar Betev, Dr. Rene Brun, Dr. Andrei Gheata and the whole ROOT team for their support and advices throughout this work.

References
[1] *An Object-Oriented Data Analysis Framework* http://root.cern.ch
[2] Freeman W. H. & Co. 1988 *Introduction to Solid Modeling* New York NY USA ISBN:0-88175-108-1
[3] *The ROOT geometry package* http://root.cern.ch/download/doc/18Geometry.pdf
[4] *The OpenCASCADE Technology* http://www.opencascade.org/
[5] *ISO 10303* http://www.iso.org