IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 2, Ver. II (Mar.-Apr. 2017), PP 22-27
www.iosrjournals.org

Visual GDML, an open-source software for building and visualizing Geant4 geometry based on the GDML markup language

Jaafar EL Bakkali[1,2], Abderrahim Doudouh[1], Hamid Mansouri[2]
[1](Nuclear Medicine Department, Military Hospital Mohammed V, Rabat, Morocco)
[2](Radiotherapy Department, Military Hospital Mohammed V, Rabat, Morocco)

Abstract: *Background and Objective: An open-source software called VisualGDML which is dedicated for geometrical and material specifications using the GDML markup language has been developed for Monte Carlo Geant4 code. Methods: The software has been written using QT4 API and uses Qscintilla library. Results: The Visual GDML open-source software offers the following features: a) a syntax highlighting for the GDML code, that makes the code more flexible and readable. b) a periodic table for facilitating the specification of the relevant materials, this by auto-generating of the adequate GDML code for each chemical element selected by the user. c) an interactive visualization system that visualizes the GDML geometry built by the user through an Open GL Stored Qt session, letting user interactively moving, rotating and zooming their 3D geometry with the aid of a computer mouse. Conclusions: We believe that our open-source software can be a helpful tool for reducing the likelihood of the human error in writing GDML code*.

Keywords: *GDML; Geant4;* Qscintilla*; Xerces-C++; QT programming; VisualGDML.*

I. Introduction

The Geant4 Monte Carlo code [1] is a simulation toolkit for simulating the passage of particles through matter. It is used in a various fields of physics that include medical and space science, high energy and accelerator physics. It was written in C++ language and exploits object-oriented technology to achieve a high transparency. An in-depth knowledge of C++ programming language is one of essential requirements to optimally use the toolkit. In Geant4, the traditional manner to define materials and geometry of such experimental setup is achieved by implementing an hard-coding C++ code thanks to the corresponding APIs. Really, this is not a trivial task; the difficulty related to it seems most important when simulating complex geometry-related shapes as medical linear accelerator (Linac) device which is used in external radiotherapy treatment of cancer diseases. It should be emphasized that the precise implementation of both geometrical and material specifications of Linac device is crucial to achieve a high precision of dose distribution calculations [2], and a small perturbation in material density can introduce consequently high discrepancies between calculated and measured doses [3].

Due to it flexibility, the Geant4 code provides other approaches to code materials and geometries of a specific experimental setup. The first one is intents to define geometrical entities (solids, volumes, materials attributes, etc.) in plain text format called "Geant4Geometry From Text File". This method is used in some Geant4-based frameworks as GAMOS [4]. Whereas the second one, which is the subject of our paper, is an XML markup language called GDML [5] which allows the construction of user geometry and material data in an XML format, this markup language is also supported by the GNU ROOT software package [6]. The GDML language has been developed for physics simulation and analysis purposes, allows complex and precise geometry implementation for a given experimental setup, and it can be modified without having to recompile the Geant4 application and providing a simple reading and writing mechanism, as well as the extensibility feature. The are many research papers [7][8]9] which employ this approach, and point out the usability of the GDML code behind C++ code for crossing the difficulty that might be encountered during specification phase of a given experimental setup. In aim to accurate simulating the corresponding complex geometry and to minimize geometry-related errors that might be encountered when modeling it by hand using hard-coding C++ approach, some research papers [10][11] intent to perform this task by exporting three-dimensional CAD (compute-aided design) complex geometry into the Geant4 simulation. This solution consists of performing the following steps: a) exporting the setup design from the CAD software in a STEP file format which provides a portable representation of geometry within well defined tolerances. b) converting STEP file to GDML file by the means of a STEP-GDML converter software for example the Fastrad software [12]. c) adding manually additional GDML codes to take into account material compositions and density specifications of each individual component. d) importing GDML file into a Geant4 application thereby including the G4GDMLParser header file. This solution seems easy but it has some limitations, particularly the user is responsible to material

assignments of individual component by the means of the Fastrad software, probably a sort of likelihood of the human error can be cross user during material assignments phase.

We propose in this paper another solution to handle the complexity that may appear during geometrical and material assignment phases, this by using GDML markup language. It is about an open-source tool that will facilitate the creation and edition of GDML files and the large Geant4 user community asking for. The QT4 API has been used to write this tool. The main focus is to become simple the use of GDML markup language without requiring the user to have knowledge about it rules. To achieve this goal, technologies as QtCreator was used for the development of our open-source tool. The tool is intents to avoid potential mistakes that can be probably committed by users, for example, introducing wrongs materials attributes or uncorrected geometrical parameters when creating or editing a GDML file with any suitable ASCII text editor. Those unpredictable mistakes can change dramatically the output of a given simulation, then causing systematic errors in the experiment. With this issue, the open-source software described in this paper has been designed and implemented to support a number of important features. These features which will be described fully through this paper, are syntax highlighting for GDML code, interactive GUI table of chemical elements and real-time fast visualization of 3D model of a given expirement.

This work is aimed essentially at creation of a GDML data description generation and edition tool coupled to a GEANT4-based geometry visualization tool. We mention that, to our knowledge, this is the first open-source software that intents to facilitate the specification of both materials and geometries using GDML language through a very-friendly open -source software.

II. Methods

Visual GDML open-source software is written using QT cross-platform application framework. The QT4 is chosen because it seems a very powerful C++ framework for building a cross-platform GUI applications, it is based on idea of "write once, compile anywhere", which means that by simply recompiling the code, the executable will be run into any operating system where is built on top of it, includes Windows, Linux and Mac OS X operating systems.

In this work we use the Qt framework to develop an open-source software. The QtCreator has been used as an integrated development environment (IDE) for building our QT4 application. In order to provide a syntax highlighting feature for the GDML code, a very useful library called Qscintilla [13] has been used. To perform the 3D model visualization task, a tiny Geant4 application has been developed and integrated to the QT application. The concerned Geant4 application called "visualizing" accepts as input argument a GDML file and has the following C++ classes:

1. Primary Generator Action class which is used to control the generation of primary particles.
2. Physics List class which collects all data and physics processes needed for such Geant4 experiment.
3. Detector Construction class which allows one to read the geometry parameters and relevant materials of an experimental setup from a GDML file by the means of the G4GDMLParser class.
4. GDML Color Writer and GDML Color Reader classes which are used for making a custom reader and writer of a GDML code that handle the color extension.

The design of the "visualizing" Geant4 application is presented in the following figure as an UML class diagram.

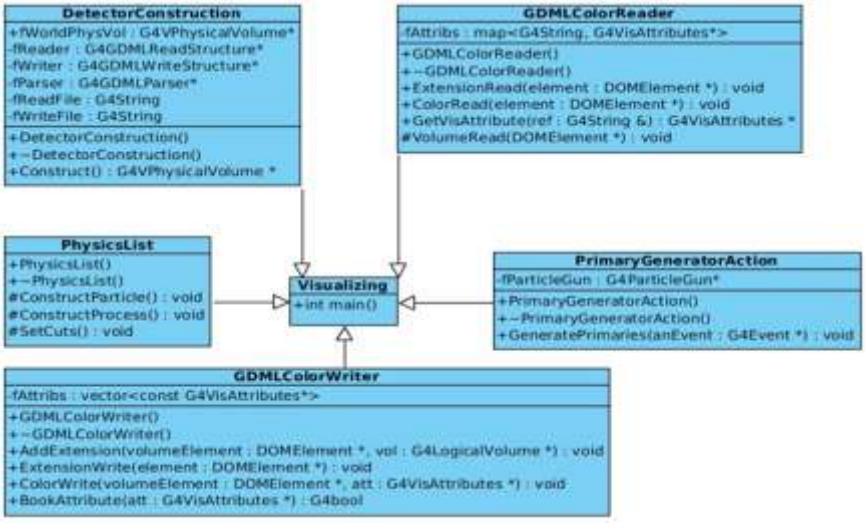

Fig 1 UML diagram generated from C++ code source of visualizing Geant4 application.

The connection between Geant4 and QT4 applications is done by the means of a Linux bash script. More details about this connection can be found in the following figure which explicates the architecture of the VisualGDML open-source software.

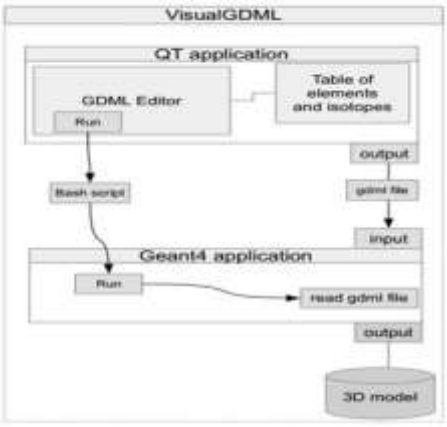

Fig 2 architecture of the Visual GDML open-source software.

This QT application has external dependencies to both Monte Carlo Geant4 code and Xerces-C++ packages. Several environment variables of the two above packages must be set in a linux script called "visualizing.sh".

III. Features of Visualgdml

Graphical editor for GDML language

The main window of Visual GDML application contains a set of graphical components including a text editor (with rich support for syntax highlighting) used to display the full content of a given GDML file. Moreover, a set of button components are placed to the right, left of the text editor. Each button allows to insert a specific GDML element into the text editor. These buttons are regrouped by category to easy it use, including solids category (contains box, sphere, tube, etc.), booleans category (contains boolean operation on CSG), definition category (contains position, constant, variable, etc.) and materials category(contains element, isotope, etc.). The Visual GDML has been designed for hiding the difficulty of writing of a lot of GDML code by hand, this is done through a very friendly GUI application; when a particular button is clicked, a GDML code is generated and placed inside text editor at the line where the cursor is positioned, this code contains everything between the GDML element's start and end tags, including sub-elements, text, attributes, as well as a mix of all of the above. The majority of the GDML code will be generated automatically, only the values of attributes belongs to the each GDML element must be set manually.

Fig 3 main window of the VisualGDML opensource software.

Syntax highlighting for GDML language

The syntax highlighting considered as a form of secondary notation and the highlights are not part of the text meaning. It is a very useful common utility some editors provide to display the code in a more readable way; various techniques for displaying code elements are implemented by many highlighters which do not reflect the semantics of the code, in most cases color and fonts styles are used to distinguish between different types of item. Displaying programming code in various colors and font styles will help the developer to comprehend its meaning. Many text editors can highlight code in various languages such Java, Delphi and C++. Markup code languages as HTML and XML have also the possibility to be highlighted. In this work, a port to QT4 of the Scintilla library called Qscintilla has been used as default syntax colorization library to highlighting the GDML markup language. The QScintilla plugin for Qt Designer has been used as text editor for our GUI application. This plugin allowing QScintilla component to be included in GUI designs similar to any other Qt widget.

Easy modeling materials by mean of a periodic table window

To facilitate the modeling of materials, Visual GDML provides a proposal window which represents a periodic table, given a flexible way to choose chemical elements and generates automatically the adequate GDML codes for them, also the mixture of elements can be created easly with the tool. Probably, this feature will accelerate the material specifications, since the majority of code needed to be implemented by hand will be generated automatically without users intervention. Only filling the data relative to the attributes of each generated GDML element is required. The periodic table of Visual GDML displays all chemical elements (elements 1 through 119) in the form of button widgets. Clicking on an element symbol will generate a GDML code for a given chemical element and it can be easy pasted inside the GDML editor.

Fig 4 a proposal window for generating GDML code for chemical elements.

Visualizing 3D model

The open-source software described in this paper allows users to view their 3D model described in GDML file through a tiny Monte Carlo Geant4 application which displays an OpenGLStoredQt session created by the G4OpenGLStoredSceneHandler class that shows a fast interactive visualization of 3D models, allowing users to translate, rotate and move their geometry, this simply are done by a computer mouse. The feature described in this section is very important to do verification of position and dimensions of each component of 3D model as well as the detection of overlaps regions between volumes. Here, it should be noted that, by default the visualization of a 3D model created from GDML file is supplied without colorization, just white color is

allowed, because the standards GDML schema doesn't include the required code for making geometry colorization, but fortunately, the extendibility nature of the GDML markup language easy allows one to extend the GDML schema and plug-in a color reader and writer to the system for handling the colorization extension. This extension called "Simple Extension Schema" (which is located in "extended/persistency/gdml/G03" Geant4 examples repository) has been used in our open-source software to handle the geometry colorization task.

IV. Results And Discussion

In order that the reader may see an example of the use of the Visual GDML user-friendly software, we provide here an example of modeling of a very complex system. It is precisely about head of a specific linear accelerator (Linac) used in radiotherapy called Saturne 43. The simulated components include:

1. Invariant elements: primary collimator, titanium window, ionization chamber and aluminum plaque.
2. Elements which strongly depend on the selected irradiation energy, namely X-ray target.
3. Others that depend on the shape of the beam, namely removable jaws.

The figure 5 shows a 3D model of Linac Saturne 43 head handled by Open GL StoredQt viewer which allows real-time fast visualization and demonstration of 3D geometry, enabling users to get mouse control of rotation, translation and zoom also.

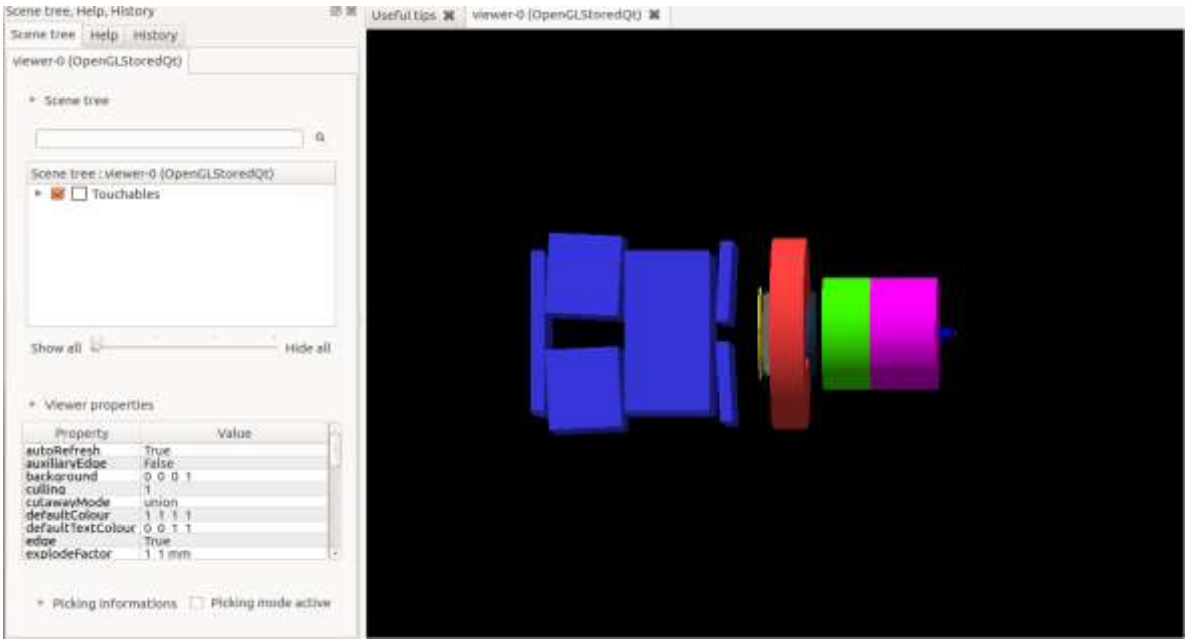

Fig 5 example of visualizing 3D model of medical accelerator from GDML file through Open GL StoredQt viewer.

With the aid of Visual GDML software the writing of GDML file is made in an easy and in a fast way without losing too much of time for writing each GDML tags by hand, since the majority of GDML code will be generated automatically. It important to note here, that from our experience, it seems much easier than manually coding the entire GDML for modeling relevant materials and geometries of a Saturne 43 Linac head when the Visual GDML is used for.

V. Conclusion

This user-friendly GUI application is intended to facilitate the utilization of GDML markup language and has shown to speed up development of Geant4 user application, specially in it materials and geometries modeling part. This way could not only save time and manpower in implementation of geometry, also assure accurate implementation of simulated experiment. For using this software, it is not necessary to consult the GDML documentation since Visual GDML will generate automatically the required GDML code. Finally, the code source of this software has been hosted on GitHub website and it can be downloaded freely from: https://github.com/EL-Bakkali-Jaafar/VisualGDML

It should be noted here, that this user-friendly GUI application is now a part of a wider-range project entitled G4Linac [14], which is a Geant4-based framework focused on modeling medical linear accelerators and it has been used for accurate modeling all relevant materials and geometries of a 6 MV Varian Linac [15].

References

[1]. S. Agostinelli, J. Allison , K. Amako , J. Apostolakis, H. Araujo, P. Arce et al., Geant4 - A Simulation Toolkit, Nuclear Instruments and Methods A 506 (2003) 250-303.

[2]. N. Tyagi, J.M. Moran, D.W Litzenberg, A.F Bielajew, B.A Fraass, I.J Chetty, Experimental verification of a Monte Carlo-based MLC simulation model for IMRT dose calculation, Journal of Medical Physics 34 (2007) 651-63.

[3]. Chibani and M.A CM, The discrepancies between Monte Carlo dose calculations and measurements for the 18 MV Varian photon beam. Journal of Medical Physics 34 (2007) 1206-16.

[4]. P. Arce, J.I. Lagares, L. Harkness, D. Perez-Astudillo, M. Canadas, P. Rato and al., Gamos: A framework to do Geant4 simulations in different physics fields with an user-friendly interface, Journal of Nuclear Instruments and Methods in Physics Research Section A 735 (2014) 304–313.

[5]. R. Chytracek, J. Mccormick, W. Pokorski, G. Santin, Geometry Description Markup Language for Physics Simulation and Analysis Applications, IEEE Transactions on Nuclear Science 53 (2006) 2892-2896.

[6]. I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun , N. Buncic and al. ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization, Journal of Computer Physics Communications 180 (2009), Pages 2499–2512

[7]. J. McCormick, 2005, Full Detector Simulation using SLIC and LCDD, J. eConf C050318 (2005) 1003 SLAC-PUB-11418, LCWS-2005-1003.
Y. Liang, B. Zhu, Z. You, K. Liu, H. Ye, G. Xu and al., A uniform geometry description for simulation, reconstruction and visualization in the BESIII experiment, Journal of Nuclear Instruments and Methods in Physics Research A 603 (2009) 325–327.

[8]. M. Ruan, V. Boudry, G. Musat, D. Jeans, J. Pande, Druid, event display for the linear collider, physics.ins-det arXiv:1303.3759 (2013).

[9]. M. Constantin, D.E Constantin, P.J Keall, A. Narula, M. Svatos, J.Perl, Linking computer-aided design (CAD) to Geant4-based Monte Carlo simulations for precise implementation of complex treatment head geometries, Journal of Physics in Medicine and Biology 55 (2010) 211–220.

[10]. C.M. Poole, I. Cornelius, J.V. Trapp, C.M.Langton, A CAD Interface for GEANT4, Journal of Australasian Physical & Engineering Sciences in Medicine 35 (2012) 329-334.

[11]. T. Beutier, E. Delage, M. Wouts, O. Serres and P. -F. Peyrard. Fastrad new tool for radiation prediction. Proceedings of the 7th European Conference onRadiation and Its Effects on Components and Systems, RADECS 2003, Noordwijk, The Netherlands, (2004) 181- 183

[12]. QScintilla - a Port to Qt v4 and Qt v5 of Scintilla. Online available : http://pyqt.sourceforge.net/Docs/QScintilla2/

[13]. J. EL Bakkali. G4Linac: a Geant4-based application for modeling medical linear accelerators used in radiotherapy (2016). Available online: https://github.com/EL-Bakkali-Jaafar/G4Linac.

[14]. J. EL Bakkali and T. EL Bardouni. Validation of Monte Carlo Geant4 code for a 6 MV Varian linac, Journal of King Saud University – Science (In Press).