Journal of Physics:
Conference Series

Geant4 application in a Web browser

To cite this article: Laurent Garnier and (On behalf ofthe Geant4 Collaboration) 2014 *J. Phys.: Conf. Ser.* **513** 062016

Journal of Physics: Conference Series **513** (2014) 062016 doi:10.1088/1742-6596/513/6/062016

Geant4 application in a Web browser

Laurent Garnier
On behalf of the Geant4 Collaboration

Laboratoire de l'accélérateur linaire (LAL), Université Paris-Sud, CNRS-IN2P3,
91898 ORSAY Cedex, France
E-Mail : garnier@lal.in2p3.fr

Abstract. Geant4 is a toolkit for the simulation of the passage of particles through matter. The Geant4 visualization system supports many drivers including OpenGL[1], OpenInventor, HepRep[2], DAWN[3], VRML, RayTracer, gMocren[4] and ASCIITree, with diverse and complementary functionalities. Web applications have an increasing role in our work, and thanks to emerging frameworks such as Wt [5], building a web application on top of a C++ application without rewriting all the code can be done. Because the Geant4 toolkit's visualization and user interface modules are well decoupled from the rest of Geant4, it is straightforward to adapt these modules to render in a web application instead of a computer's native window manager. The API of the Wt framework closely matches that of Qt [6], our experience in building Qt driver will benefit for Wt driver. Porting a Geant4 application to a web application is easy, and with minimal effort, Geant4 users can replicate this process to share their own Geant4 applications in a web browser.

1. A complete visualization system

The Geant4 Visualization System [7] is a multi-driver graphics system designed to serve the Geant4 Simulation Toolkit. It is aimed at the visualisation of Geant4 data, primarily detector descriptions and simulated particle trajectories and hits. It can handle a variety of graphical technologies simultaneously and interchangeably, allowing the user to choose the visual representation most appropriate to their requirements. For example:

- very quick response in surveying successive events;
- high-quality output for presentation and documentation;
- flexible camera control for debugging detector geometry and physics;
- selection of visualizable objects;
- interactive picking of graphical objects for attribute editing or feedback to the associated data;
- highlighting incorrect intersections of physical volumes;
- co-working with a graphical user interface;

Because it is very difficult to respond to all of these requirements with only one built-in visualizer, the Graphics Interface was developed to support multiple graphics drivers over several complementary

graphics technologies. Geant4 may use a graphics library directly, communicate with an independent process via pipe or socket, or simply write an intermediate file for a separate viewer. The current distribution of Geant4 contains, at the latest count, 14 drivers of various sorts. They are listed in figure 1 along with their capabilities. Those that need external libraries or packages may only be activated at the compilation step if the corresponding external system is installed. In addition, the user may extend this list by implementing his/her own driver to the specification of the abstract interface.

Driver	Categorie	Variant	Picking	Hight Quality print	Interactive	Browse geom. Hearichie	Direct access to G4	Make movies	Web
OpenGL	Graphic	X							
		Xm							
		Qt							
		Win32							
OpenInventor	Graphic								
HepRepFile	File-writing								
RayTracer	File-writing								
Dawn	File-writing								
VRML	File-writing								
gMocren	File-writing								
ASCII Tree	File-writing								

Figure 1. Driver list and their capabilities.

A user may draw to the basic abstract interfaces, either in C++ code or, more usually, via visualization commands through a user interface, and expect it to be rendered in one of a number of different ways: to a computer screen (graphic drivers) or to a file for subsequent browsing (file-writing drivers).

The workhorse of the Geant4 Visualisation System is the set of OpenGL drivers. Recent work [7] has focused on our implementation of OpenGL within the context of the popular platform-independent GUI toolkit, Qt. The Geant4 Qt visualization/GUI solution allows the user to rotate and zoom the view, to toggle visibility of various viewed objects, and to interrogate detailed information about geometry, trajectories and hits. The same interface allows the user to control the overall simulation, even customizing the GUI with the user's own action buttons. Other recent work on Geant4's OpenGL implementations has improved speed of rendering and exploited OpenGL's native ability to generate high quality PostScript renderings. For a wide variety of graphics drivers, Geant4 now offers improved view annotation features such as text, arrows, rulers, axes, date stamps, logos, etc.

2. Geant4 web-based driver: Wt driver

Recent developments now allow the user to run OpenGL inside their web browser. Thanks to this key feature, building a new web-based driver (called Wt driver) is possible. It will render graphics directly into a web browser and will be platform independent.

The web provides a resource for running complex applications with the user requiring nothing more than a web browser. The Geant4 Wt driver will provide a means for all users to run simulations and interact with a Geant4 application without installing Geant4 itself. The interaction will be done by an identical web page for each application whatever the user platform.

As there is nothing to install on the user computer in order to run this web based driver, it is an easy way of running tutorials, learning Geant4 and teaching physics without any software installation problems.

Providing a web page with the full set of Geant4 examples could be useful for users in order to demonstrate each example without the need to install Geant4 libraries on his computer.

3. Wt library

The Wt library is a C++ library for developing web applications. It will be straightforward to use with Geant4 visualization and interface libraries as they are in written in the same programming language. It provides an extensive set of widgets, which work regardless of JavaScript availability (but benefit from JavaScript availability) and a single specification for both client- and server-side validation and event handling. Wt optionally uses XHTML and CSS for layout and decoration. It will generate standards compliant HTML or XHTML code and will take care of the responsibility of writing secure and browser-portable web applications for us.

Developing a web application with the Wt framework is fundamentally identical to developing an application for any other graphical user interface system, such as Windows or X11. Most importantly, the entire application is written in only one compiled language (in Geant4's case, C++), from which the necessary HTML, Javascript, CGI, and AJAX code are generated. The responsibility for writing browser-portable web applications is thus transferred to the Wt library implementation.

A further benefit of Wt is that it will maximally use JavaScript and AJAX when available in the user's browser, but automatically revert to plain HTML otherwise. An application developed with Wt is thus technology-agnostic.

4. Geant4 Wt driver implementation

Thanks to the Wt framework, a Geant4 application developer does not need to write any HTML5, Javascript, or CSS; they write their interface in pure C++. The framework handles all User Interface interactions. The Wt driver pattern will be inspired by the Qt driver[8], moreover the syntax of the Wt framework is similar to the Qt framework syntax. The Wt visualization driver will be based on OpenGL.

4.1. OpenGL

OpenGL is available in web browsers thanks to the HTML5 canvas element (fully supported by all current versions of browsers [9]). The Geant4 Wt driver will use this element to render OpenGL inside the user web browser. To be able to render in OpenGL on different platforms and browsers, a standard called WebGL[10] has been adopted. It is based on OpenGL ES 2.0 and provides an API for 3D graphics inside a web browser. Inside the Geant4 visualization library, OpenGL commands are based on OpenGL 1.1. In order to use this Geant4 OpenGL in our Wt driver, all OpenGL commands will have to migrate first to OpenGL 2 on which OpenGL ES 2.0 is a subset (see figure 2).

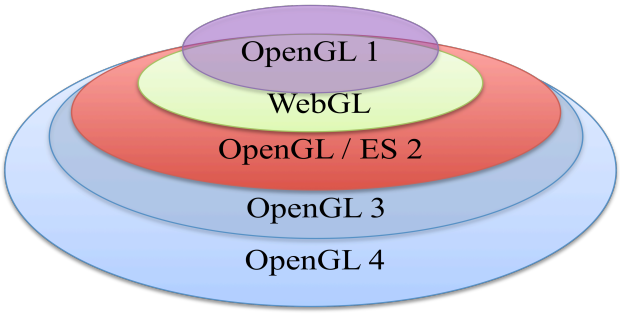

Figure 2. OpenGL version evolutions.

This migration will be a complex process as the interface in OpenGL ES2.0 is a radical departure from that in OpenGL2. The change will require not just minor recoding but significant restructuring in the Geant4 OpenGL driver (see part 5.)

4.2. Changes in Geant4 Visualization and Interface library part

The Geant4 Graphical User Interface (GUI) is loosely coupled to other Geant4 libraries, thus the only changes required in Geant4 to support a Wt driver are located in Geant4's Interfaces and Visualization libraries.

Wt driver will be mainly based on Qt driver, with changes due to the Wt framework. The similarities in the Qt/Wt interfaces allow an easy translation, as shown in this example of a specific driver function:

Wt driver C++ version	**Qt driver C++ version**

Wt::WWidget* G4UIWt::CreateHistoryTBWidget() {
  fHistoryTBWidget = new Wt::WPanel();
 	
  fHistoryTBWidget->setCollapsed(true);
  fHistoryTBWidget->setCollapsible(true);
 	
  fHistoryTBTableList = new Wt::WSelectionBox();
  fHistoryTBTableList->setSelectionMode
(Wt::SingleSelection);
  fHistoryTBTableList->changed().connect
  (this,&G4UIWt::CommandHistoryCallback);
 	
  fHistoryTBWidget->setCentralWidget
(fHistoryTBTableList);
  return fHistoryTBWidget;
}
	QWidget* G4UIQt::CreateHistoryTBWidget() {
	  fHistoryTBWidget = new QWidget();
	
	  QVBoxLayout *layoutHistoryTB = new QVBoxLayout();
	  fHistoryTBTableList = new QListWidget();
	  fHistoryTBTableList->setSelectionMode
	  (QAbstractItemView::SingleSelection);
	  connect(fHistoryTBTableList,
	  SIGNAL(itemSelectionChanged()),
	  SLOT(CommandHistoryCallback()));
	  fHistoryTBTableList->installEventFilter(this);
	
	  layoutHistoryTB->addWidget(fHistoryTBTableList);
	
	  fHistoryTBWidget->setLayout(layoutHistoryTB);
	  return fHistoryTBWidget;
	}

4.3. Main program

In other Geant4 user interfaces each application has a main event loop, however the Geant4 Wt driver will create a web server instance which will have one main event loop for one user. For every new session, which corresponds to a new user accessing the web application, the library will create a new Geant4 application object.

5. Prototype and ongoing improvements

The prototype for the Wt driver has been tested (see figure 3 and figure 4). Basic 3d functions like zoom, rotate, display volume and traces are fully functional, and interaction with the Geant4 kernel is implemented using a command line interface in the web application.

Regarding the web interface, some improvements are needed before it will be a fully functional driver. Some extended graphics components such as scene tree browser, help widget, and history have to be implemented.

Regarding visualization, migration from OpenGL to WebGL/OpenGL ES 2 is not complete, as some OpenGL functions such as glVertex, glBegin, glEnd, glMatrix, display lists have not yet been migrated and other functions deprecated since OpenGL 3 (and not available in WebGL) have to be replaced by using OpenGL Vertex Buffer Object (VBO) instead. Moreover, as OpenGL VBO may be faster than using OpenGL display lists, this migration to VBO will benefit all Geant4 OpenGL drivers such as Qt, Xm and Open Inventor. Regarding multiple users on a single Wt application server, a way to manage file access on the server has to be developed in order to deliver pictures or other files produced by the application.

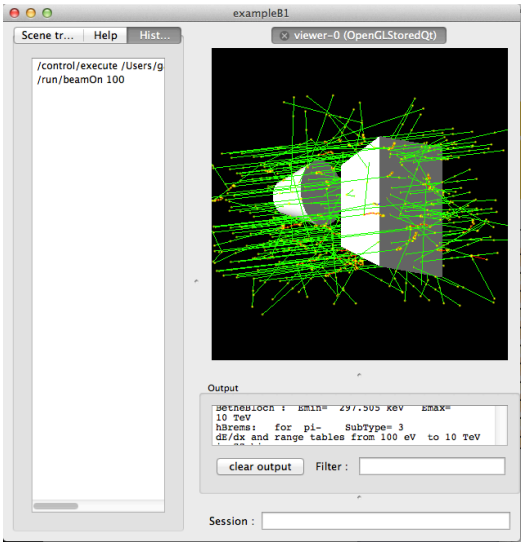

Figure 3. ExampleB1 on a Mac 10.8.5 with Qt driver.

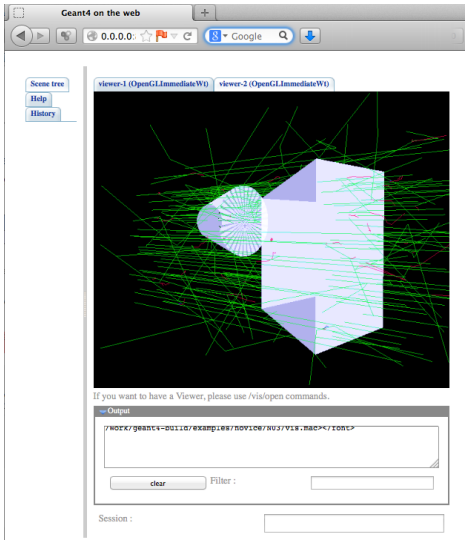

Figure 4. ExampleB1 on a Mac 10.8.5 in Firefox Browser with Wt driver.

6. Summary

A prototype of a Wt driver for Geant4 has been tested and basic functions are already working. All functions of the Qt driver are not available at the moment but will be integrated soon. Implementing the prototype needs not only the design of a new driver, but also some migration in the OpenGL part of the visualization library. This migration is almost finished now and will allow users to have better OpenGL performance in all graphics drivers.

7. References

[1] OpenGL web page (2013): http://www.opengl.org
[2] Joseph Perl (2004), *HepRep: a Generic Interface Definition for HEP Event Display Representables*. Retrieved from http://www.slac.stanford.edu/~perl/heprep
[3] S. Tanaka et al., (1997) *DAWN for Geant4 visualization, Proceedings of the CHEP '97 Conference,* Berlin
[4] A. Kimura, S. Tanaka, T. Sasaki., (2006) *A visualization tool for Geant4-based medical physics applications*. Int. J. of Computer Assisted Radiology and Surgery; Vol. 1; 462-463.
[5] Wt library web page (2013). Retrieved from http://www.webtoolkit.eu/wt
[6] Qt driver user guide in Geant4 library (2012). Retrieved from http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/ch08s03.html#sect.VisDrv.Qt
[7] Allison, J et al., (2013). *The Geant4 visualization system – A multi-driver graphics system*. International Journal of Modeling, Simulation, and Scientific Computing; Vol. 4, Suppl. 1
[8] Qt driver tutorial (2013). Retrieved from http://geant4.in2p3.fr/spip.php?article60&lang=en
[9] HTML5 canvas (2013). Retrieved from http://en.wikipedia.org/wiki/Canvas_element - Support
[10] WebGL web page (2013). Retrieved from http://www.khronos.org/webgl/