Journal of Physics: Conference Series **664** (2015) 072020 doi:10.1088/1742-6596/664/7/072020

Geant4 VMC 3.0

I Hřivnáčová

Institut de Physique Nucléaire (IPNO), Université Paris-Sud, CNRS-IN2P3, 91406 Orsay Cedex, France

E-mail: ivana@ipno.in2p3.fr

A Gheata

European Organization for Nuclear Research (CERN), CH-1211 Genève 23, Switzerland

E-mail: andrei.gheata@cern.ch

Abstract. Virtual Monte Carlo (VMC) [1] provides an abstract interface into Monte Carlo transport codes. A user VMC based application, independent from the specific Monte Carlo codes, can be then run with any of the supported simulation programs. Developed by the ALICE Offline Project and further included in ROOT [2], the interface and implementations have reached stability during the last decade and have become a foundation for other detector simulation frameworks, the FAIR facility experiments framework being among the first and largest.

Geant4 VMC [3], which provides the implementation of the VMC interface for Geant4 [4], is in continuous maintenance and development, driven by the evolution of Geant4 on one side and requirements from users on the other side. Besides the implementation of the VMC interface, Geant4 VMC also provides a set of examples that demonstrate the use of VMC to new users and also serve for testing purposes. Since major release 2.0, it includes the G4Root navigator package, which implements an interface that allows one to run a Geant4 simulation using a ROOT geometry.

The release of Geant4 version 10.00 with the integration of multithreading processing has triggered the development of the next major version of Geant4 VMC (version 3.0), which was released in November 2014. A beta version, available for user testing since March, has helped its consolidation and improvement. We will review the new capabilities introduced in this major version, in particular the integration of multithreading into the VMC design, its impact on the Geant4 VMC and G4Root packages, and the introduction of a new package, MTRoot, providing utility functions for ROOT parallel output in independent files with necessary additions for thread-safety. Migration of user applications to multithreading that preserves the ease of use of VMC will be also discussed. We will also report on the introduction of a new CMake [5] based build system, the migration to ROOT major release 6 and the improvement of the testing suites.

1. Introduction

Geant4 VMC [3] has been previously described in [6], [7] and in the context of ALICE in [8]. In this paper, we will report on the new developments and improvements included in its major version 3.0. First we will briefly present the history of Geant4 VMC development and previous major versions. The next sections will be then devoted to new capabilities introduced in this


Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

major version: the integration of multithreading into the VMC design, the introduction of a new CMake based build system, the migration to ROOT major release 6 and the other improvements.

2. Geant4 VMC History

VMC defines an abstract layer between a detector simulation user code and the Monte Carlo transport code (MC). In this way the user code is independent from any specific MC and can be used with different transport codes, such as GEANT 3.21 [9], Geant4 [4] or FLUKA [10], within the same simulation application. It was developed by the ALICE Offline Project and, after the complete removal of all dependencies from the experiment specific framework, it was included in ROOT.

The first version of the VMC interface was included in ROOT 3.03/05 in October 2002. The Geant4 VMC version series started with version 0.1 and continued up to version 0.5. The base Geant4 and ROOT versions for this release, as well as for the next Geant4 VMC major releases, are presented in Table 1.

Version	**v0.1**	**v1.0**	**v2.0**	**v3.0**
Date	Oct 2002	Jul 2003	Dec 2006	Nov 2014
New features	*First version*	*Geometry convertors*	*G4Root navigation*	*Multithreading CMake, ROOT 6*
ROOT	`3.03/09`	`3.05/06`	`5.14/00`	`5.34/23 and 6.02/01`
Geant4	`4.1`	`5.2`	`8.2`	`10.00.p03`
Last	v0.5	v1.9	v2.15	v3.1

Table 1. Overview of Geant4 VMC versions.

The major version 1.0 included geometry convertors which allowed to use geometry defined with new ROOT geometrical modeller, TGeo [11], in a VMC application. This 1.0 version was released in July 2003. The geometry convertors were gradually replaced with usage of an external tool, VGM [12] and removed from Geant4 VMC in version 1.8. The 1.x series continued up to version 1.9 for three years.

The next major version, 2.0, released in December 2006, included the G4Root package which implements Geant4 navigation that uses directly the TGeo geometry. G4Root was moved in Geant4 VMC from ROOT in order to group together Geant4 related packages based on ROOT and to facilitate their building for the user. In this version, the geometry definition in all examples was reimplemented with usage of TGeo geometry modeller. The previous code, based on GEANT3-style functions defined in *TVirtualMC* interface, was kept as an option in testing for assuring a backward compatibility. This 2.0 version started the series which continued up to version 2.15 for 8 years.

The procedure of versioning was standardized during this period. A new Geant4 VMC version release follows every new Geant4 version, which happens once in a year. Then, a new release can also be triggered by new developments or by a change in the VMC interface in ROOT. A given Geant4 VMC version is tested with given versions of Geant4 and ROOT, which are noted in the Geant4 VMC release documentation. In general, it can be then used with the given version of ROOT or higher and with the given version of Geant4 including its further patches. The users are recommended to update their Geant4 installation with each Geant4 patch release,

even though the Geant4 patches released after the Geant4 VMC tag do not appear in the Geant4 VMC release documentation.

The last major version, 3.0, triggered by the release of Geant4 version 10.00 with the integration of multithreading processing, was released in November 2014 and it was followed by a consolidated version 3.1, based on the next Geant4 version, 10.01, in December 2014. The new capabilities introduced in this major version will be discussed in next sections.

3. Multithreading

The development of Geant4 VMC multithreading (MT) prototype started in the last quarter of 2011. We adopted the same approach as in Geant4 and the main task of the migration to multithreading processing was the replacement of all singleton objects in Geant4 VMC with singletons per thread. Modifications of the same type as described in the Geant4 documentation [13] were applied to Geant4 VMC classes.

Changes were required also on the level of the VMC interfaces as both *TVirtualMC* and *TVirtualMCApplication* are defined as singletons. A new function, *IsMT()*, was added in *TVirtualMC* in order to be able to query multithreading mode in user applications. A set of new functions was added to the *TVirtualMCApplication* interface class as presented in Table 2.

```
// required for running in MT
virtual TVirtualMCApplication* CloneForWorker() const;
// optional
virtual void InitForWorker() const;
virtual void BeginWorkerRun() const;
virtual void FinishWorkerRun() const;
virtual void Merge(TVirtualMCApplication* localMCApplication);
```

Table 2. New functions added in TVirtualMCApplication for multithreading.

The users are required to override *TVirtualMCApplication::CloneForWorker()*, while the implementation of the other functions is optional. These functions are then used to clone the application and its containing objects on thread workers. Creating these objects on worker threads is then triggered from the Geant4 VMC classes. Detailed instructions for migration of VMC applications to multithreading are provided on the VMC Web site together with other implementation details and useful tips.

A new set of classes taking care of locking critical ROOT operations in multithreading mode - as for example registering ROOT objects to ROOT trees - is introduced in a new MTRoot package. This package enhanced the ROOT input/output functionality previously provided by the Ex02RootManager class within the examples. MTRoot is independent from the Geant4 VMC interface, but it is used in VMC examples. It provides *TMCRootManager* and *TMCRootManagerMT*, the ROOT manager classes for VMC sequential and multithreaded applications respectively. Both these classes are derived from a common interface and are introduced in terms of a single implementation class in order to avoid code duplication, see Figure 1. In addition, a utility class *TMCAutoLock*, providing a locking mechanism, is also included in this package. It was extracted from the *G4AutoLock* class implementation in Geant4 for Linux and MAC OSX platforms.

The G4Root package is the interface allowing running a Geant4 simulation with a ROOT geometry. In spite of being provided within Geant4 VMC, it can be built and used independently

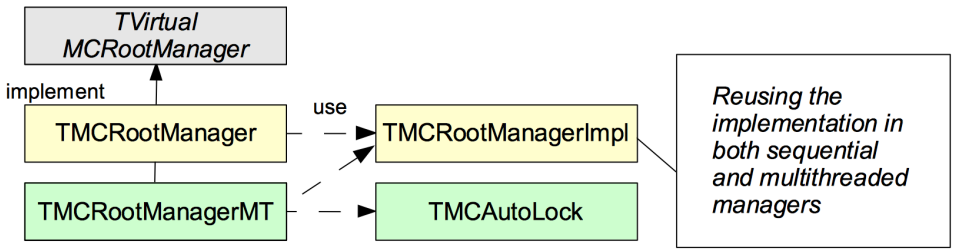

Figure 1. The design of MTRoot classes for the ROOT output.

from VMC. It implements a specialization of the *G4Navigator* class which uses directly the TGeo geometry. Its migration to multithreading mode has been also accomplished in version 3.0. The design had to be adapted to creating an instance of the navigator for each worker thread. The old G4Root test based on Geant4 novice example N06 has been replaced with a new test based on Geant4 extended example OpNovice. This example was introduced in Geant4 as a new version of the N06 example after a transition to a new set of examples for novice users, called basic, in the Geant4 9.5 release. An important fix in G4Root navigation in geometries using assembly volumes has been introduced in the first patch release, v3.0.p01. It addressed a problem of stuck particles getting killed, reported by ALICE .

4. Build System

The VMC packages used the build system based on configuration Make files distributed in ROOT specific directory, *root/etc/vmc*, including also platform specific configurations. Being disconnected from the ROOT build system, it required extra maintenance. Moving to CMake was a natural step after its introduction in Geant4 three years ago, facilitating both introducing the support for multithreading mode and the future maintenance of the system.

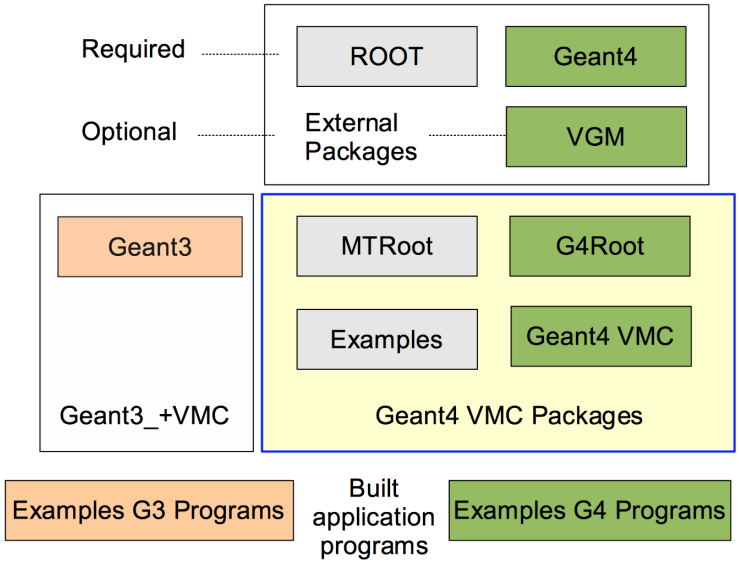

Figure 2. The Geant4 VMC packages.

Geant4 VMC, besides its own set of classes provided in directory *source* and built in *geant4vmc*

library, includes also G4Root, MTRoot and VMC examples packages, each with different dependencies, see Figure 2. All these packages are based on the ROOT framework. The G4Root and Geant4 VMC sources depend on Geant4, and Geant4 VMC source can optionally depend on VGM.

In order to make possible running VMC applications without dynamic loading of libraries, which causes a known performance penalty, becoming even more important with multithreading, the VMC examples can now be also linked statically with all libraries into standalone programs. For keeping maximum simplicity of the code, a fixed configuration is defined in the examples *main()* function and its more flexible version is provided in the tests. These examples application programs bring a direct dependence on both Monte Carlo libraries and VMC packages.

These dependencies are addressed by the new CMake based build system. This makes possible building all packages at once or building any of them individually. It also handles the use of optional packages. The available build options are summarised in Table 3. The configuration files for all included packages, PackageConfig.cmake, are generated and installed in the installation area. The packages can be then used directly in the client projects without a need to define their FindPackage.cmake files. "Find" configuration files for ROOT and Geant4 and some more utility files are also provided.

Geant4VMC_BUILD_G4Root	Build G4Root	ON
Geant4VMC_BUILD_MTRoot	Build MTRoot	ON
Geant4VMC_BUILD_Geant4VMC	Build Geant4VMC	ON
Geant4VMC_BUILD_EXAMPLES	Build VMC examples	ON
Geant4VMC_USE_G4Root	Build with G4Root	ON
Geant4VMC_USE_VGM	Build with VGM	OFF
Geant4VMC_USE_GEANT4_UI	Build with Geant4 UI drivers	ON
Geant4VMC_USE_GEANT4_VIS	Build with Geant4 Vis drivers	ON
Geant4VMC_USE_GEANT4_G3TOG4	Build with Geant4 G3toG4 library	OFF
Geant4VMC_INSTALL_EXAMPLES	Install examples	ON

Table 3. Overview of Geant4 VMC build options.

For a maximum simplicity the generic "Find" and "Use" configuration files are provided for building both examples libraries and programs. While the VMC application libraries are independent from specific Monte Carlo libraries, the VMC application program has to be linked also with selected Monte Carlo libraries including their VMC interface. That is why two sets of "Find" and "Use" files were defined: FindVMC, UseVMC and FindMC, UseMC. The former are used for building the VMC application library and the latter for the program. "Find" files are used to find all needed packages and "Use" files to set compiler definitions, includes and libraries according to selected configuration options. Besides being provided in Geant4 VMC, they are also available in Geant3_+VMC.

5. Other Improvements

5.1. ROOT 6

This Geant4 VMC version has been also migrated to ROOT 6. This required to adapt the examples macros to use the new ROOT interpreter, Cling. The simplest solution for a backward

interpreter incompatibility was a separation of macros for loading libraries from the single macros used to run VMC example. Minor fixes were also needed in the code to make it compiling against C++11 standard, mandatory with this ROOT version. The CMake configuration files were updated for changes in the generation of dictionaries with Cling. A generation of *rootmap* files, which enable library autoloading, was also added in this context.

The migration to ROOT 6 has not discontinued the build against ROOT 5 series and actually build against both these ROOT major versions is supported and tested.

5.2. Testing

VMC testing is based on shell scripts which run automatically 21 test configurations plus a special test for all Geant4 available physics lists, the list of which is updated after each Geant4 release. The test configurations are continuously enhanced while adding new features in Geant4 VMC. In addition to the existing test suite run from the standard ROOT session, in version 3.0 we added a second test suite which runs the same tests configurations from examples programs. The test suites perform tests with both GEANT3 and Geant4 simulation programs.

The test scripts were also improved to facilitate their possible inclusion in automated testing. Command line arguments were added to make possible to select testing with one simulation program only, either GEANT3 or Geant4, or to change the test build directory. Summary messages and return codes were also added to allow to evaluate the result of the tests in a client code.

5.3. User Support

The enhancements and features available in the new version are documented on the Geant4 VMC Web site integrated in ROOT Drupal. The web pages on "Installing Geant4 VMC" and "Installing and Running Examples" are completely replaced with new instructions and a new page dedicated to "Multithreading" was added.

With each new version, the source code documentation is automatically generated with the Doxygen [14] tool and the release notes are presented with a detailed description of the new developments and bug fixes, added in the dedicated "history" file on the Web site.

In October 2014, the VMC related bug reports were separated from the ROOT JIRA project in a new, JIRA VMC project.

6. Conclusions

Geant4 VMC version 3.0 providing multithreading support is available since November 2014. The interest in Geant4 multithreading was expressed by both ALICE and FAIR experiments: migration of the FairRoot framework to multithreading is in progress and there are ongoing ALICE tests with a multithreading prototype. This prototype is based on a VMC example with realistic ALICE geometry, magnetic field, primary event generator and simplified TPC detector response. Besides multithreading, this version includes also new build system based on CMake together with further improvements.

Acknowledgments

The authors would like to acknowledge Oliver Freyermuth from *Physics Institute of the University of Bonn* for testing development versions and his contribution to CMake build, examples test suites and ROOT 6 migration.

References

[1] http://root.cern.ch/drupal/content/vmc
Hřivnáčová I et al 2003 *Proc. of Computing in High Energy and Nuclear Physics* (La Jolla) pp THJT006
[2] http://root.cern.ch

[3] http://root.cern.ch/drupal/content/geant4-vmc
[4] Agostinelli S et al 2003 *Nucl. Instrum, and Methods* **A506** 250-303
Allison J et al 2006 *IEEE Transactions on Nuclear Science* **53 No. 1** 270-278
[5] K. Martin and B. Hoffman, *Mastering CMake: A Cross-Platform Build System* , Kitware Inc., 2003
[6] Hřivnáčová I 2008 *J. Phys: Conf. Series* **119** 032025
[7] Hřivnáčová I 2012 *J. Phys: Conf. Series* **396** 022024
[8] Hřivnáčová I et al 2011 *J. Phys: Conf. Series* **331** 032016
[9] Brun R et al 1985 *GEANT3 User Guide* (CERN Data Handling Division, DD/EE/84-1)
[10] Fasso A et al 2001 *Proc. of the MonteCarlo 2000 Conference* (Lisbon, Springer Verlag Berlin) 159-164 and 955-960.
[11] Brun R, Gheata A and Gheata M 2003 *Proc. of Computing in High Energy and Nuclear Physics* (La Jolla) pp THMT001
[12] Hřivnáčová I 2008 *J. Phys: Conf. Series* **119** 042016
[13] http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForToolkitDeveloper/html/ch02s14.html
[14] http://www.stack.nl/ dimitri/doxygen/index.html