EPJ Web of Conferences **245**, 02005 (2020) https://doi.org/10.1051/epjconf/202024502005
CHEP 2019

New Developments in the VMC Project

Ivana Hřivnáčová[1,*] and *Benedikt* Volkel[2,3]

[1]Université Paris-Saclay, CNRS/IN2P3, IJCLab, 91405 Orsay, France
[2]European Organization for Nuclear Research (CERN), Geneva, Switzerland
[3]Ruprecht-Karls-Universitaet Heidelberg, Germany

Abstract. Virtual Monte Carlo (VMC) provides a unified interface to different detector simulation transport engines such as GEANT3 and GEANT4. Since recently all the VMC packages (the VMC core library, also included in ROOT, and the GEANT3 and GEANT4 VMC) are distributed via the VMC Project GitHub organization. In addition to these VMC related packages, the VMC project also includes the Virtual Geometry Model (VGM), which is optionally used in GEANT4 VMC for conversion between GEANT4 and ROOT TGeo geometry models.
In this contribution we will present the new organization of the VMC project at GitHub and new developments in the VMC interfaces and the VMC packages. We will cover the introduction of the sensitive detector interface in the VMC core and both GEANT3 and GEANT4 VMC and the new GEANT4-related developments.
GEANT4 VMC 3.0 with the integration of multithreading processing was presented at CHEP in 2015. In this presentation we will report on new features included since this version: the improved support for magnetic fields, the integration of fast simulation, Garfield physics, GEANT4 transition radiation and monopole physics. Five new VMC examples demonstrating these new features, and serving also for tests, will be also discussed. Finally we will mention the work towards the code quality and improvements in testing, documentation and automated code formatting.

1 Introduction

Virtual Monte Carlo (VMC) [1] 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 [2], GEANT4 [3] or FLUKA [4], within the same simulation application. VMC 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 [5].

The first version of the VMC interface was included in ROOT 3.03/05 in October 2002, and the packages with the implementation of the VMC interface for GEANT3 and GEANT4 were distributed at the ROOT web site. In 2004, a new geometrical modeller, TGeo [6], was developed in a collaboration between the ALICE and ROOT teams, and successfully interfaced to the three existing programs. In GEANT4 VMC, users can choose either to

*e-mail: ivana@ipno.in2p3.fr

convert the TGeo geometry into GEANT4 native geometry using Virtual Geometry Model (VGM) [7], or to use G4Root navigator which implements the GEANT4 navigation that uses directly the TGeo geometry.

After the integration with the ROOT geometrical modeller, GEANT3 VMC became stable as the development of GEANT3 was stopped a long time ago. GEANT4 VMC and FLUKA VMC are in continuous maintenance and development, driven by the evolution of the Monte Carlo codes on one side and requirements from users on the other side. The interface for FLUKA VMC has been discontinued in 2010 and restricted to the ALICE Collaboration usage, and will not be discussed in this paper.

In past two years the VMC packages were gradually moved from ROOT into a stand-alone project in GitHub [8]. This transition will be described in section 2. In sections 3 and 4 we will report on new features included in the VMC core and GEANT4 VMC since version 3.1 respectively. Finally, in section 5 we will briefly report on the work towards the code quality.

The introduction of the multiple engine framework in the VMC packages and new developments in VGM are covered in two separate contributions to this conference [9], [10].

2 VMC Project on GitHub

Since its first version, VMC was distributed with the ROOT system. It consisted of the `vmc` core package, included directly in ROOT, and the `geant3` and `geant4_vmc` packages, distributed with the ROOT CVS (and later SVN and Git) server.

To facilitate maintenance and distribution, `geant4_vmc` included also other packages: the VMC examples since its first version, the G4Root navigator, added in 2006, and MTRoot, providing the classes for Root IO management, which was introduced with multi-threading in 2013.

In 2017, `geant3` and `geant4_vmc` packages were moved into GitHub, under a new organization, `vmc-project`. They were then followed by the Virtual Geometry Model (VGM) package, a geometry conversion tool used in GEANT4 VMC for the support of external geometry definition previously distributed via SourceForge.

Finally, in 2019 the `vmc` core package was separated from the ROOT source into a new stand-alone `vmc` package in the GitHub `vmc-project` organization. The motivation for this step was a gain in flexibility and faster workflow for new developments of multiple engine mode. The `vmc` package in ROOT is deprecated since version 6.18 and will be removed in the future.

The overview of the VMC packages is shown in Figure 1. The packages external dependencies are highlighted with colors.

2.1 Documentation

The VMC documentation was migrated from the ROOT Drupal system into the static site generator Hugo [11], exploiting the Learn [12] theme. The documentation source pages were adapted for Hugo by choosing the Markdown format with TOML front matter format. The Hugo web site source was then migrated to the new `vmc-documentation` GitHub package in the `vmc-project` organization and a deployment via GitHub pages into new web site [13] was introduced.

3 New developments in VMC

After a long stability period, the VMC interfaces were enhanced in the past year with new functions for user defined sensitive detectors and classes for new multiple engine mode.

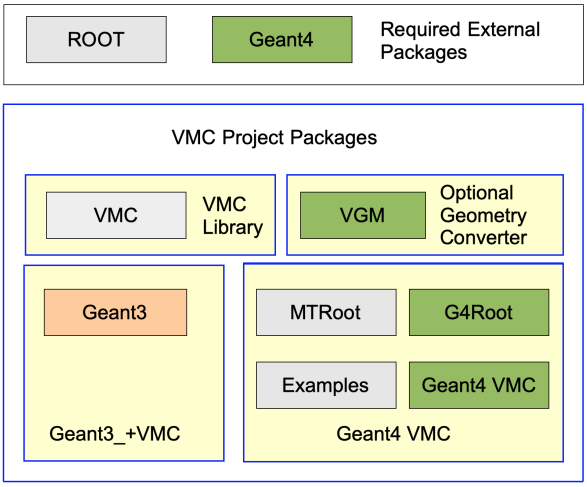

Figure 1. VMC Project Packages Overview

The support for user defined sensitive detectors was implemented as per request of the FAIR Collaboration. A new interface to a user sensitive detector, *TVirtualMCSensitiveDetector*, has been added and the support for this new feature was implemented in `geant3` and `geant4_vmc`.

The user sensitive detector objects should be associated with selected volumes using the new *TVirtualMC* and *TVirtualMCApplication* functions. Users can also choose whether scoring should be performed exclusively in a new way, via sensitive detectors, or via both new and old ways.

Recently, VMC was enhanced with a new multiple engine mode. Users have the possibility to mix multiple engines within the simulation of one event. Depending on user conditions, the simulation is effectively split among the chosen engines profiting from each of their advantages or specific capabilities. This development is presented in a separate contribution to the CHEP 2019 conference [9].

To demonstrate the usage of these new features, the VMC example `E03` was split into three variants:

- `E03a` - the original version of the example demonstrating scoring via sensitive volumes and *MCApplication::Stepping*
- `E03b` - new variant demonstrating scoring via sensitive detectors derived from the new interface
- `E03c` - new variant demonstrating sharing event simulation among multiple engines

4 New developments in GEANT4 VMC

In this section, we will report on new developments and improvements in GEANT4 VMC included in version 3.1 and beyond.

4.1 New features in the support for magnetic fields

The user magnetic field in VMC is defined by the *TVirtualMagField* interface.

Since GEANT4 VMC version 3.2, it is possible to define local magnetic fields. A local field is defined in the same way as a global field, but it is associated with a selected volume or volumes. The local magnetic field is also applied to all daughter volumes and it is possible to combine a global field with one or more local fields. It should be noted that local fields are supported only with GEANT4 and an equivalent global magnetic field has to be provided for GEANT3 simulation.

The default magnetic field equation integration method and default accuracy parameters, defined in GEANT4, can be customized with a set of dedicated commands, defined in GEANT4 VMC, both for global and local fields. It is also possible to provide a user defined magnetic field equation of motion and/or its integrator. These objects should be instantiated in a GEANT4 based detector construction class.

Users can also activate utilising the cached magnetic field inspired by the *G4CachedMagneticField* class in GEANT4. To reduce the number of field evaluations, recalculating of the field value is avoided inside a sphere of a given radius *ConstDistance* from the previously evaluated location.

Since GEANT4 VMC version 3.5, it is possible to propagate the zero magnetic field defined in VMC tracking media into GEANT4 geometry. A local zero magnetic field is set to the logical volumes associated with the tracking medium with the *ifield* parameter of zero value.

4.2 Integration of specific physics models

In this section we will present the integration of various specific GEANT4 physics models, most of which were added on the ALICE Collaboration requirements in order to improve the simulation performance in terms of physics results and speed or for further physics studies.

4.2.1 Specific physics models per regions

In this sub-section we will discuss the integration of specific physics models which can be applied in selected geometry regions.

Dedicated user interface (UI) commands were implemented in GEANT4 VMC for user selection of regions, particles and specific physics models. As the VMC application is independent from GEANT4, this selection cannot be defined in the application classes like in GEANT4 applications.

A set of classes was introduced to handle the specific models configurations: *TG4ModelConfiguration* to hold the information of the association of the special physics model with the regions and particles where the model should be applied, *TG4ModelConfigurationManager* to hold a collection of model configurations and *TG4ModelConfigurationMessenger* to define UI commands. Each new instance of the model configuration manager class creates a new command directory with the commands applied to the associated particular model, see Table 1.

`/mcPhysics/physicsName/setModel modelName`
`/mcPhysics/physicsName/setParticles particleName1 particleName2 ...`
`/mcPhysics/physicsName/setRegions regionName1 regionName2 ...`
where physicsName = fastSimulation, emModel, biasing

Table 1. User interface commands to define the special physics models.

At present, such configuration is available for the following special models:

- `emModel` - for activating the electromagnetic extra physics models: PAI, PAIPhotonModel and SpecialUrbanMscModel (the GEANT4 UrbanMsc model adapted for needs of the ALICE electromagnetic calorimeter by the GEANT4 Collaboration)
- `fastSimulation` - for fast simulation models: Gflash (provided) or user defined model
- `biasing` - for setting special physics models via biasing technique.

The electromagnetic extra physics models and fast simulation models are applied by the means of the physics builders (classes derived from *G4VPhysicsConstructor*), defined in GEANT4 VMC, which are invoked when a corresponding model configuration is selected and configured by the user.

The biasing techniques are applied by the *TG4BiasingManger* class which creates the instances of the biasing operator and operation and applies them to selected volumes. The classes for biasing operator and operation were developed by the GEANT4 Collaboration on the ALICE experiment requirement to make possible to use a different combination of hadronic models, e.g. FTFP + INCLXX instead of the default FTFP + BERT, for the final-state generation in selected regions. They are also available in the new GEANT4 extended example `Hadr08` in the `hadronic` category, introduced in GEANT4 version 10.6.

The fast simulation model framework was developed in order to enable the integration of the GEANT4 Gflash fast simulation and, later, also the Garfield [14] physics. This integration is demonstrated in new VMC examples `Glash` and `ExGarfield` (see also [15]). Besides these specific cases the framework allows also user defined fast simulation models.

Usage of the electromagnetic extra physics models and biasing technique is demonstrated and tested in the VMC example `E03`.

4.2.2 Transition radiation

The GEANT4 transition radiation physics is not available in the GEANT4 physics lists classes and its inclusion in the user application, requiring also a definition of radiator properties, is demonstrated in a dedicated GEANT4 example. That's why its use in GEANT4 VMC was not straightforward and required its integration.

A new set of commands was introduced to define the radiator properties, see Table 2, which are stored in a collection of the *TG4RadiatorDescription* objects in *TG4GeometryManager*. The GEANT4 transition radiation process, using these radiator parameters, is then created in the *TG4TransitionRadiationPhysics* builder according to a selected model (`xtrModel`).

```
/mcDet/setNewRadiator volumeName xtrModel foilNumber
/mcDet/setRadiatorLayer materialName thickness [fluctuation]
/mcDet/setRadiatorStrawTube gasMaterialName wallThickness gassThickness
```

Table 2. User interface commands to define radiator properties.

A new VMC example, called `TR`, is provided to demonstrate and test the usage of transition radiation with GEANT4 VMC. This example implements the same simulation setup as the GEANT4 extended `TestEm10` example in the `electromagnetic` category.

4.2.3 Monopole

The integration of the GEANT4 extended `monopole` example in the `exoticphysics` category was requested for physics studies by the ALICE Collaboration. A set of classes describ-

ing the monopole "physics" (monopole particle, field setup, equation of motion, transportation, physics builder and its messenger) was extracted from GEANT4 in a GEANT4 VMC new sub-category, `physics_monopole`. The existing GEANT4 VMC classes were extended to allow a selection of the monopole parameters and instantiation of the monopole physics.

A new VMC example, `Monopole`, is provided to demonstrate and test usage of a monopole transportation with GEANT4 VMC. This example implements the same simulation setup as the example in GEANT4.

5 Code quality

5.1 Test suites

VMC testing is based on shell scripts which automatically run 61 test configurations plus a special test for all GEANT4 available physics lists, the list of which is updated after each GEANT4 release. Two test suites, one running from the standard ROOT session and the other one running the same tests configurations from examples programs, perform tests with both GEANT3 and GEANT4 simulation programs.

The test suite scripts were restructured with the use of functions to facilitate further maintenance and addition of new tests. The output was also improved to give a better overview of passed and failed tests including the concept of multiple test cases (the same example is tested multiple times with different configuration macros) for each example.

The test suites were also enhanced with new command line arguments that allow to run only selected examples, to turn on an additional verbosity and to print the differences of the obtained outputs and the reference test outputs, which are provided with the source code and updated with each release.

5.2 Uniform code style

The usage of clang-format tool [16] was introduced to automatically format the code in order to obtain a consistent code style from all contributing developers. A configuration file with a customized style and a script for automatically applying the tool were introduced in the `vmc` and `geant4_vmc` repositories.

6 Conclusion

VMC has been in production since a long time. In this paper we discussed the new organization of the packages in GitHub and we have presented the new features in the interface and new developments in GEANT4 VMC since last four years.

References

[1] Hřivnáčová I et al, 2003, *Proceedings of the CHEP2003 conference* pp THJT006

[2] Brun R and Rademakers F, 1985, *GEANT3 User Guide* (CERN Data Handling Division, DD/EE/84-1)

[3] 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
Allison J et al, 2016, *Nuclear Instruments and Methods in Physics Research* **A 835** 186-225

[4] Ferrari A et al, *FLUKA: A multi-particle transport code* (CERN, Geneva, 2005) Report Number CERN-2005-010; INFN-TC-2005-11; SLAC-R-773
Böhlen T T et al, 2014, *Nuclear Data Sheets* **120**, 211–214
[5] Brun R et al, 1997, *Nucl. Inst. & Meth. in Phys. Res. A* **389**, 81-86
http://root.cern.ch
[6] Brun R, Gheata A et al, 2003, *Proceedings of the CHEP2003 conference* pp THMT001
[7] Hřivnáčová I, 2008, *J. Phys: Conf. Series* **119** 042016
[8] https://github.com
[9] Volkel B et al, Using multiple engines in the Virtual Monte Carlo package. *Proceedings of the CHEP2019 conference J. Phys.: Conf. Series*
[10] Hřivnáčová I, The Virtual Geometry Model. *Proceedings of the CHEP2019 conference J. Phys.: Conf. Series*
[11] https://gohugo.io
[12] https://learn.netlify.com
[13] https://vmc-project.github.io/
[14] http://garfield.web.cern.ch/garfield/
http://garfieldpp.web.cern.ch/garfieldpp/
[15] Pfeiffer D et al, 2019, *Nucl. Instrum. and Methods* **A935**, 121-134
[16] https://clang.llvm.org/docs/ClangFormat.html