The Geant4 Virtual Monte Carlo

Ivana Hřivnáčová

Institut de Physique Nucleaire, 91406 Orsay Cedex, France

E-mail: `ivana@ipno.in2p3.fr`

Abstract. The Virtual Monte Carlo (VMC) [1] provides an abstract interface into the Monte Carlo transport codes: Geant3, Geant4 and Fluka. The user VMC based application, independent from the specific Monte Carlo codes, can then be run with all three simulation programs. The VMC has been developed by the ALICE Offline Project and since then it has been adopted in other experimental frameworks.

Since its first release in 2002, the implementation of the VMC for Geant4 (Geant4 VMC) [2] is in continuous maintenance and development, mostly driven by the requirements from new, non ALICE, users. In this presentation we will give an overview and the present status of this interface. We will report on new features, such as support for user defined Geant4 classes (the physics list, detector construction class) or support for Root TGeo geometry definition and G4Root navigation. We will also discuss the aspects specific to Geant4 and comment on the strong and weak points of the VMC approach.

1. Introduction

The VMC has already been described in detail in [1] and [3], here we only briefly summarize the main ideas. The VMC defines an abstract layer between the detector simulation user code and the transport Monte Carlo code (MC). In this way the user code is independent from a specific MC and it can be used with different transport codes, as Geant3 [4], Geant4 [5], FLUKA [6], within the same simulation application (Fig. 1).

In the VMC, we introduce on one side an interface to the transport MC itself, *TVirtualMC*, and on the other side an interface to the user application, *TVirtualMCApplication*, as schematically shown in Fig. 2. This allows decoupling the dependence between the user code and the concrete MC. In VMC, we also define *TVirtualMCStack*, an interface to a user defined particle stack. All these classes are available together with a few more utility classes in the *vmc* package in the ROOT framework [7].

The overview of the implementation of the VMC for Geant4, its present status and the new features will be described in this article.

2. Geant4 VMC as a Geant4 application

To use the Geant4 toolkit, users have to define their application based on the Geant4 user classes. There are three mandatory classes whose methods the user must override in order to implement a simulation. They require the user to define the detector geometry, specify the physics to be used, and describe how initial particles are to be generated. Then there are five optional user action classes whose methods the user may override in order to gain control of the simulation at various stages.

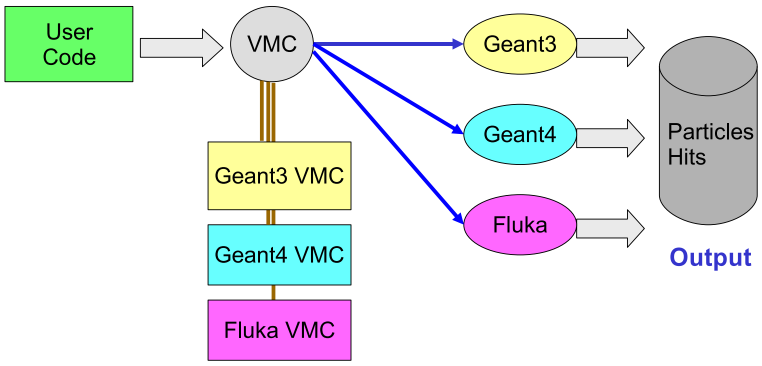

Figure 1. The VMC concept

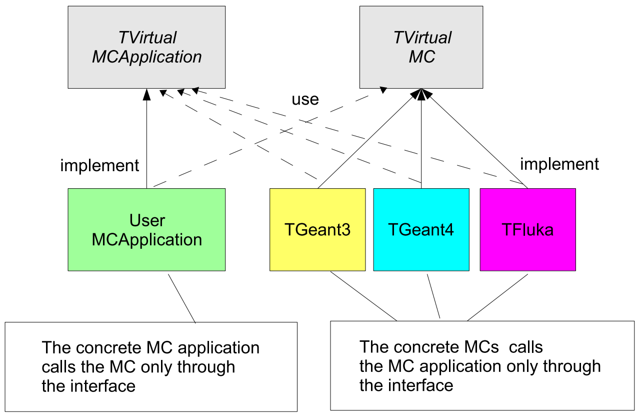
TVirtual MCApplication
TVirtual MC
use
implement
implement
User MCApplication
TGeant3
TGeant4
TFluka
The concrete MC application calls the MC only through the interface
The concrete MCs calls the MC application only through the interface

Figure 2. The VMC design

Besides implementing the functions defined in the *TVirtualMC* interface, already described in [3], the Geant4 VMC also fulfills the role of a Geant4 application. In the following sub-sections we will describe how the Geant4 mandatory and user action classes are implemented in VMC.

2.1. Geometry definition

In a Geant4 application, the geometry definition is implemented in the detector construction class. The geometry definition in the VMC has evolved in two phases:

- In the first implementation, the VMC provided Geant3-like functions for building geometry. This facilitated the move to VMC for Geant3 users, however this interface is limited to the features available in Geant3.
  The support for Geant3-like VMC functions is in Geant4 VMC implemented using the G3toG4 package in Geant4, as already described in [1].
- A new approach came with the introduction in Root of its own geometrical modeller, TGeo [8], independent from existing simulation tools, which provides IO, visualization and verification tools. Since the validation of TGeo with Geant3 VMC, the old way of defining geometry via VMC functions is deprecated and new VMC users are encouraged to start from TGeo.
  The support for TGeo in Geant4 VMC is realized in two ways:
  – By geometry conversion from TGeo to Geant4.

It was first, in 2003, implemented with a specific roottog4 converter, provided with Geant4 VMC, which was later, in 2005, replaced with use of the Virtual Geometry Model [9], provided as a tool external to Geant4 VMC.

– By direct interaction between the TGeo navigator and the Geant4 simulation toolkit. By the end of 2006, Geant4 VMC was integrated with the G4Root package [10], which provides the interface between the TGeo navigation and Geant4 simulation toolkit.

Users have the possibility to choose between Geant4 native navigation and G4Root navigation, which are available in both cases to define geometry: via the VMC interface or via TGeo. The possible selections are:
VMCtoGeant4, *VMCtoRoot*, *RootToGeant4*, *Root*, *Geant4*,
where the first word refers to the geometry input and the second one to the navigator to be used. The last option is reserved for geometry defined via the Geant4 geometry model, which is supported in Geant4 VMC, too.

Once the options selected are specified in the user code, the necessary packages are automatically connected and the selected navigator is activated (Fig. 3).

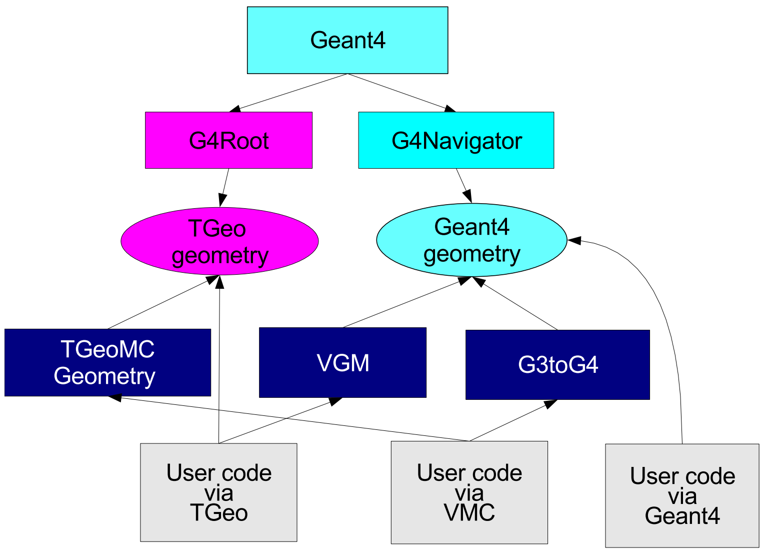

Figure 3. Geometry options

2.2. *Tracking media*

The tracking medium in VMC has the same meaning as in Geant3. It represents a set of tracking parameters associated to a material: sensitivity flag, parameters for magnetic field, maximum step, etc.

As Geant4 did not adopt the Geant3 concept of tracking media, in Geant4 VMC a specialization of the *G4UserLimits* class is used to hold the relevant information from user defined tracking media: the step limit, the vector of cut values and the vector of process controls. The user limits are then used by the special processes, described in the next chapter, to apply user defined values in tracking.

2.3. *Physics selection*

Geant4 does not have any default particle or physics process. Users have to define them explicitly in their application.

Geant4 VMC provides a helper physics list class, *TG4ComposedPhysicsList* (Fig. 4), which allows to combine several *G4VUserPhysicsList* derived classes together. By default,

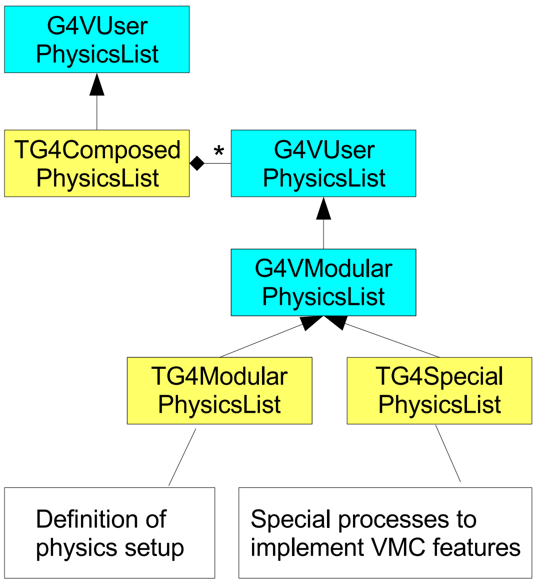

Figure 4. The Geant4 VMC default physics list

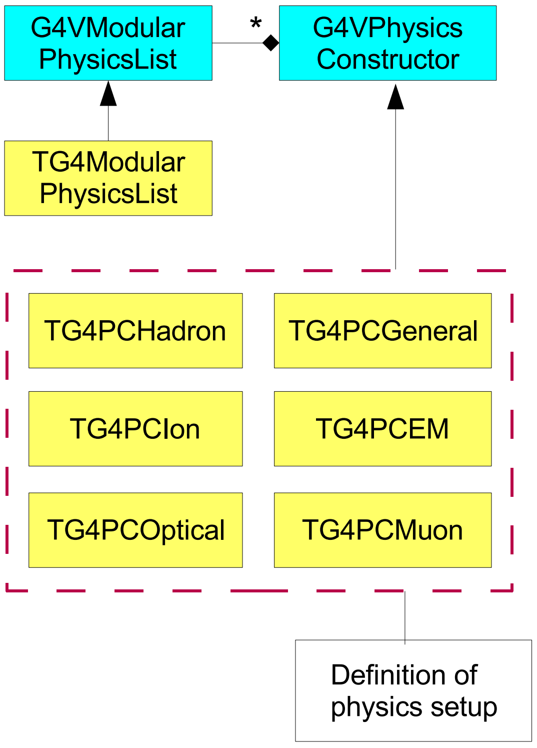

Figure 5. The Geant4 VMC modular physics list

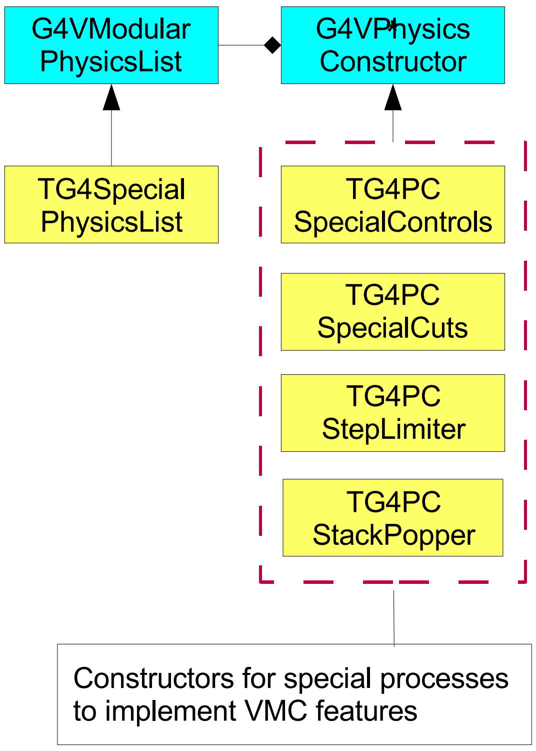

Figure 6. The Geant4 VMC special physics list

this composed physics list includes two physics lists: *TG4ModularPhysicsList*, implementing the physics setup, and *TG4SpecialPhysicsList*, implementing VMC specific features. Both implement the *G4VModularPhysicsList* class, which allows to decompose the physics setup into the *G4VPhysicsConstructor* objects.

The physics setup implemented in *TG4ModularPhysicsList* (Fig. 5), has been collected from

Geant4 novice examples and it is not tuned to any particular physics problem. That is why users are invited to implement their own physics list class. The special processes are re-usable in the user physics list.

The VMC defines its own identifiers associated with given physics processes. First, the processes are associated with Geant3-like flag names, which can be used by users for activation or inactivation of a selected process. Then, the *TMCProcess* codes are defined for getting the information during tracking via *TVirtualMC*.

If users want to use certain VMC functions, they have to fill in the maps for the process controls and for the process codes in the physics list. The mapping itself is already done in *TG4ModularPhysicsList*.

The special processes in *TG4SpecialPhysicsList* (Fig. 6) implement the following VMC features:

- **Special controls process**
  This process implements the activation and inactivation of selected processes via *TVirtualMC* using the VMC control flags. The flag names and the flag values have the same meaning as in Geant3.
  Below we give the declaration of the *TVirtualMC* function and an example of its use for the global activation of the Compton process:
  ```
  TVirtualMC::SetProcess(const char* flagName, Int_t flagValue);
  gMC->SetProcess("COMPT", 1);
  ```
- **Special cuts process**
  In Geant4, cuts are applied as cuts in range per particle and region. In VMC instead, there are introduced Geant3-like cuts in energy which are applied globally or per tracking medium. In Geant4 VMC, Geant3-like cuts are implemented by the special cuts process and user limits. These cuts are applied as tracking cuts, not as a threshold.
  Below we give the declaration of the *TVirtualMC* function and an example of its use for setting the global energy cut 1 MeV to *gamma* particle:
  ```
  TVirtualMC::SetCut(const char* cutName, Double_t cutValue);
  gMC->SetCut("CUTGAMA", 1e-03);
  ```
- **Step limiter process**
  The Geant4 *G4StepLimiter* process is used to limit the step if step limitation is selected by the user.
- **Stack popper process**
  This process implements adding user defined secondary particles and will be discussed in detail in the next section.

The special processes are not activated by default, they have to be activated by the user explicitly.

2.4. Primary generator and VMC Stack

The VMC provides the interface for the stack of particles, which has to be implemented by the user. Users can also re-use the stack implementations from the VMC examples. The particles in the VMC stack are of the Root *TParticle* type.

The primary particles are first filled by the user application in the user VMC stack as *TParticle* objects, then they are transformed by Geant4 VMC in Geant4 objects and passed to the Geant4 kernel. This sequence is shown in Fig. 7.

The secondary particles, created by Geant4 physics processes, are stored by Geant4 VMC in the user VMC stack. By default, storing happens at the beginning of tracking a secondary particle. Optionally, users can choose to store a particle already in the step of the parent track,

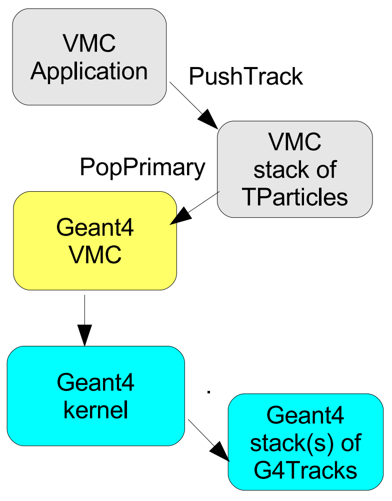

Figure 7. Stacking of primary particles

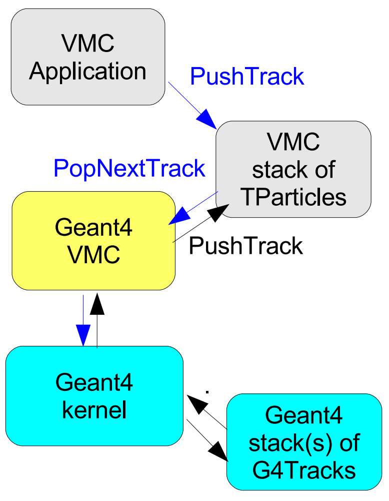

Figure 8. Stacking of secondary particles

just when the secondary particle is produced. The storing of secondary particles can be also inactivated.

In VMC, users also have the possibility of adding particles to the VMC stack during tracking. In this case, the particle added by the user is considered as a secondary particle of a current particle tracked. This feature is used by the ALICE collaboration to simulate the generation of feedback photons in an avalanche close to a multiplicative wire. In Geant4 VMC, this functionality has been implemented in the special *TG4StackPopper* process, which monitors the VMC stack, and if a new particle in the stack is detected, it is popped from the VMC stack and passed to the Geant4 tracking.

Stacking of secondary particles is shown in Fig. 8, where the processing with an activated stack popper process is highlighted in blue.

3. Geant4 VMC specific features

Though the goal of the VMC is to hide to the user the specific transport codes, it does not necessarily intend to prevent a user to access their facilities like various debug printing, geometry verification and visualization tools. Geant4 VMC has been implemented in a modular and easy extensible way and allows the user to re-use already existing Geant4 based code.

3.1. Run configuration

The run configuration class, *TG4RunConfiguration*, see Fig. 9, takes care of creating all Geant4 user defined mandatory and action classes that will be initialized and managed by the Geant4 kernel. This class can be extended in a user application by inheritance; this gives a user the possibility to override the Geant4 VMC default behavior or extend it for each Geant4 user defined class.

The use cases for a user defined physics list, detector construction and primary event generation are demonstrated and tested in the VMC example E03, .

3.2. User Interface

The Geant4 VMC brings together two toolkits: Geant4 and Root, both with their own user interface, with quite different aspects.

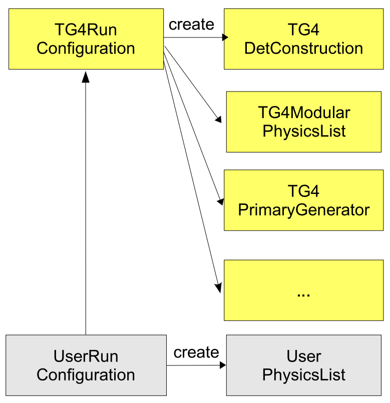

Figure 9. The run configuration class

In Geant4, the user interface consists of implemented commands associated to selected Geant4 or user objects. The commands are sensitive to the application state, and they perform strong checking of user input.

In Root, the user interface is based on the CINT interpreter. All Root and user objects, for which the dictionary exists, are accessible in the user interface. Any object public function can be called at any time.

In Geant4 VMC, both Geant4 and Root user interfaces are available for the user. User can switch between the two shells and it is also possible to call a foreign command or a foreign macro in both shells. However it is not possible to access Geant4 objects from the Root shell, as their dictionaries are not produced. A set of Geant4 commands associated with the objects defined in Geant4 VMC is implemented. To make their Geant4 VMC relation apparent, all these commands start with the prefix *mc*.

4. VMC examples and test suite

4.1. Examples

In order to demonstrate the use of VMC, four Geant4 novice examples (N01, N02, N03, N06) were rewritten as VMC applications. In all examples, the geometry is defined using the Root TGeo modeller and, alternatively for testing purpose, using the VMC functions. In the E03 example, user defined run configuration with physics list, geometry and primary generator defined directly via Geant4 is demonstrated.

4.2. Test suite

In the test suite script, all examples are executed with all options, the output is saved in files and can be compared with the reference outputs stored in CVS.

5. The VMC Approach

The greatest advantage of the VMC approach is the possibility to run the VMC application with different transport codes with minimum effort and so compare immediately the results.

As the VMC is integrated in the Root framework, it naturally provides the user the possibility to define the simulation application entirely in this framework: the application is run from the Root program and can be steered by a Root macro.

On the other hand, the user who opted for the VMC approach should be aware of some of its limitations. The Geant4 toolkit is constantly upgraded with new features, implemented by Geant4 collaboration developpers on request of its growing number of users . Naturally, not all these features can be immediately interfaced in the VMC, some of them being difficult to fulfill with other MC implementations.

As the VMC represents an additional layer between the user application and Geant4, it brings some overhead in performance. This depends on the user application, but in average it can be of the order of 10%.

6. Conclusions

Geant4 VMC is in production since 2002. In this paper we gave an overview of the tool and of its new features such as the support for user defined Geant4 classes, the support for Root TGeo geometry definition and navigation, or the user defined secondary particles.

The Geant4 VMC is under test by several collaborations who adopted the VMC code: ALICE, CBM, PANDA, Opera, MINOS.

References

[1] Hřivnáčová I et al 2003 *Proc. of Computing in High Energy and Nuclear Physics* (La Jolla) Id THJT006
[2] http://root.cern.ch/root/vmc/Geant4VMC.html
[3] Carminati et al 2004 *Proc. of Computing in High Energy and Nuclear Physics* (Interlaken) Id 433
[4] Brun R et al 1985 *GEANT3 User Guide* (CERN Data Handling Division, DD/EE/84-1)
[5] Agostinelli S et al 2003 *Nucl. Instrum, and Methods* **A506** 250-303
[6] Fasso A et al 2001 *Proc. of the MonteCarlo 2000 Conference* (Lisbon, Springer Verlag Berlin) 159-164 and 955-960.
[7] http://root.cern.ch
[8] Brun R, Gheata A and Gheata M 2003 *Proc. of Computing in High Energy and Nuclear Physics* (La Jolla) Id THMT001
[9] Hřivnáčová I 2004 *Proc. of Computing in High Energy and Nuclear Physics* (Interlaken) Id 387
[10] Gheata A and Gheata M 2007 *Computing in High Energy and Nuclear Physics* Id 38