THE VIRTUAL MONTE CARLO: STATUS AND APPLICATIONS

R. Brun, F. Carminati, E. Futó, A. Gheata, P. Hristov, A. Morsch, CERN, Geneva Switzerland

A. Fassò, SLAC, Stanford, USA

M. Gheata, Institute of Space Science, Bucharest, Romania

I. Hřivnáčová, IPN, Orsay, France

Abstract

The current major detector simulation programs, i.e. GEANT3, GEANT4 and FLUKA have largely incompatible environments. This forces the physicists willing to make comparisons between the different transport Monte Carlos to develop entirely different programs. Moreover, migration from one program to the other is usually very expensive, in manpower and time, for an experiment offline environment, as it implies substantial changes in the detector description and response simulation code. To solve this problem, the ALICE Offline project has developed a virtual interface to these three programs allowing their seamless use without any change in the framework, the geometry description or the scoring code. Moreover, a new geometrical modeller has been developed in collaboration with the ROOT team, and successfully interfaced to the three programs. This allows the use of one description of the geometry, which can be used also during reconstruction and visualization. The paper will describe the present status and future plans for the Virtual Monte Carlo. It will also present the capabilities and performance of the geometrical modeller.

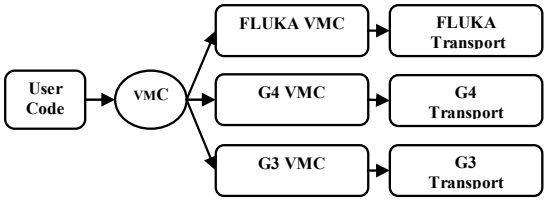

Fig. 1: The Virtual MC Concept

CONCEPT AND CURRENT DESIGN

The Virtual MC (VMC) has already been described in detail in Ref. [1]. Here we describe briefly the main concepts (Fig. 1) and progress since then. The VMC defines an abstract layer between the detector simulation user code and the transport code. In this way the user code for detector description, scoring and detector response simulation becomes independent from the specific MC and it is easy to use different transport codes, GEANT3 [2], GEANT4 [3], FLUKA [4] within the same simulation application.

The interface consists of four abstract classes that are part of the ROOT [5] framework (Fig. 2). *TVirtualMC* is the interface to the transport MC itself. *TVirtualMCApplication*, *TVirtualMCStack*, and *TVirtualDecayer* are used to make a concrete implementation of the *TVirtualMC* independent from the user specific simulation application. *TVirtualMCApplication* represents an abstraction of the run and event steering, *TVirtualMCStack* defines an interface to a user defined particle stack and *TVirtualMCDecayer* defines an interface to an external particle decayer.

Two concrete implementations, *TGeant3* and Geant4 VMC for GEANT3 and GEANT4, respectively have already been described in Ref. [1]. Since then, these implementations have been further tested and consolidated. On user demand *TVirtualMC* has been extended by a few new methods for user particle definitions and for user run abortion. Our main efforts have been concentrated on the implementation of the *TVirtualMC* realisation of the FLUKA transport code, *TFluka*. This work has been simplified by the introduction of an additional level of abstraction provided by the class *TVirtualMCGeometry*. It facilitates the interfacing of the MC transport code to an external geometry modeller for geometry description and navigation as discussed in the following section.

GEOMETRY AND VIRTUAL MC

Geometry is a critical issue when implementing the *TVirtualMC* interface. Mapping the *TVirtualMC* methods for defining the geometry to the native geometry definition and creation was a simple task only for GEANT3. For GEANT4 the G3toG4 toolkit [6] is used. In case of FLUKA it is practically impossible to convert to its native geometry that is based on a flat structure obtained by Boolean operations between basic shapes as

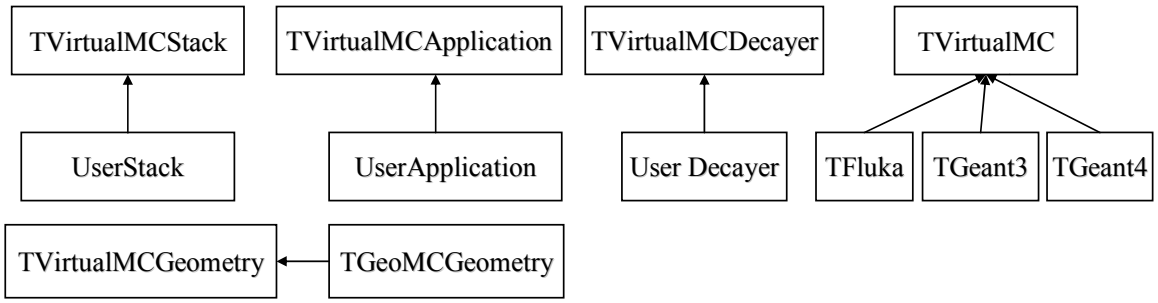

Fig. 2: The Virtual MC class design.

opposed to the tree structure used by GEANT3 and GEANT4. For the first implementation of *TFluka* we used FLUGG [7] in order to be able to run FLUKA together with the GEANT4 geometry modeller. However, this was unsatisfying since the combination *TFluka*+FLUGG was completely unstable in tests with the ALICE detector simulation. This is just one of the symptoms of the fact that geometry modellers are deeply embedded within the MC frameworks.

For this reason the ALICE Off-line Project in collaboration with the ROOT team have proposed and implemented a multi-purpose geometry modeller for HEP, *TGeo* [5, 8], that is integrated within the VMC infrastructure. *TGeant3* has already been fully interfaced to *TGeo* and will enter production next year. Also *TFluka* uses *TGeo*. Testing of *TFluka*+*TGeo* is still ongoing. The interfacing with *TGeant4* turned out to be difficult so far due to the lack of abstraction of the GEANT4 navigation. However the situation has improved and a strategy has been defined for the complete integration of *TGeo* into the GEANT4 VMC implementation. We expect that the implementation will be finished in 2005. At present, the *TGeo* model is supported in GEANT4 VMC via the *roottog4* tool by converting directly from the *TGeo* model to the GEANT4 native geometry model. In addition, a generalized version of the converters [9] has been developed in order to facilitate the use of VMC for GEANT4 users by providing a tool for transition from the GEANT4 native geometry to *TGeo*.

STATUS OF GEANT3 and GEANT4 VMC

GEANT3 VMC

TGeant3 is used by ALICE in production. Further developments concentrated on the use with *TGeo* and testing. Comparisons between *TGeo* and GEANT3 native geometry on the hit level do not show any relevant differences. Step-by-step comparisons are ongoing. Especially transport close to boundaries needs further detailed testing and fine-tuning. Timing benchmark tests show that *TGeo* is in all tested cases faster then the native GEANT3 geometry modeller [8].

GEANT4 VMC

GEANT4 VMC is in maintenance, i.e. it is regularly tested with each new GEANT4 and ROOT releases and updated if needed. The main new features are the enabling of user defined physics lists, the consolidation of the *rootog4* converter (support for positioning with reflection and for composite shapes) and extension of *g4toxml* to the GDML scheme [10].

FLUKA VMC STATUS

The main components and communication lines for the *TFluka* implementation are shown in Fig. 3.

Initialisation and Configuration

In the initialisation phase the *TGeo* geometry is created and *TFluka* collects the user physics and transport configuration in objects of type *TFlukaCutsOption* and *TFlukaConfigOption*. Using this information *TFluka* writes a standard FLUKA configuration file that contains the user configuration, the material definitions and the material to region assignments. *TFluka* calls FLUKA only through its main steering routine *flukam*. This routine is called once for the FLUKA initialisation in which FLUKA reads the configuration file and then for every event. During the processing of events, FLUKA calls back *TFluka* through the FLUKA user routines. These have been written in C++ and have the task to pass intermediate information about the transport status from the argument list to *TFluka* and to delegate the FLUKA calls to corresponding methods in *TFluka, TVirtualMCApplication and TVirtualMCStack*.

Primary Particles

The routine responsible for fetching new primary particles and putting them on the FLUKA stack is *source*. Our version of *source* takes particles one by one from the *TVirtualMCStack*. It also checks whether there are any un-transported primaries on the stack. This can happen in the case particles have been put on the stack by the user. Finally *source* is responsible to call for each primary the following methods of *TVirtualMCApplication*: *BeginPrimary()* and *PreTrack()* at the beginning of transport and *FinishPrimary()* and *PostTrack()* at the end.

Stepping

User stepping is called through the FLUKA routines *mgdraw, bxdraw, usdraw* and *endraw* for normal steps, boundary crossings, interactions and energy depositions, respectively. In order to minimize the stack size FLUKA transport follows first the secondaries arising from interactions like *bremsstrahlung* or δ-electron production and resumes the transport of the mother particle after the daughters have been finished. The pausing and resuming of the mother particle is signalled to the stepping routine in order to guarantee that the user hit generation algorithms can handle this situation.

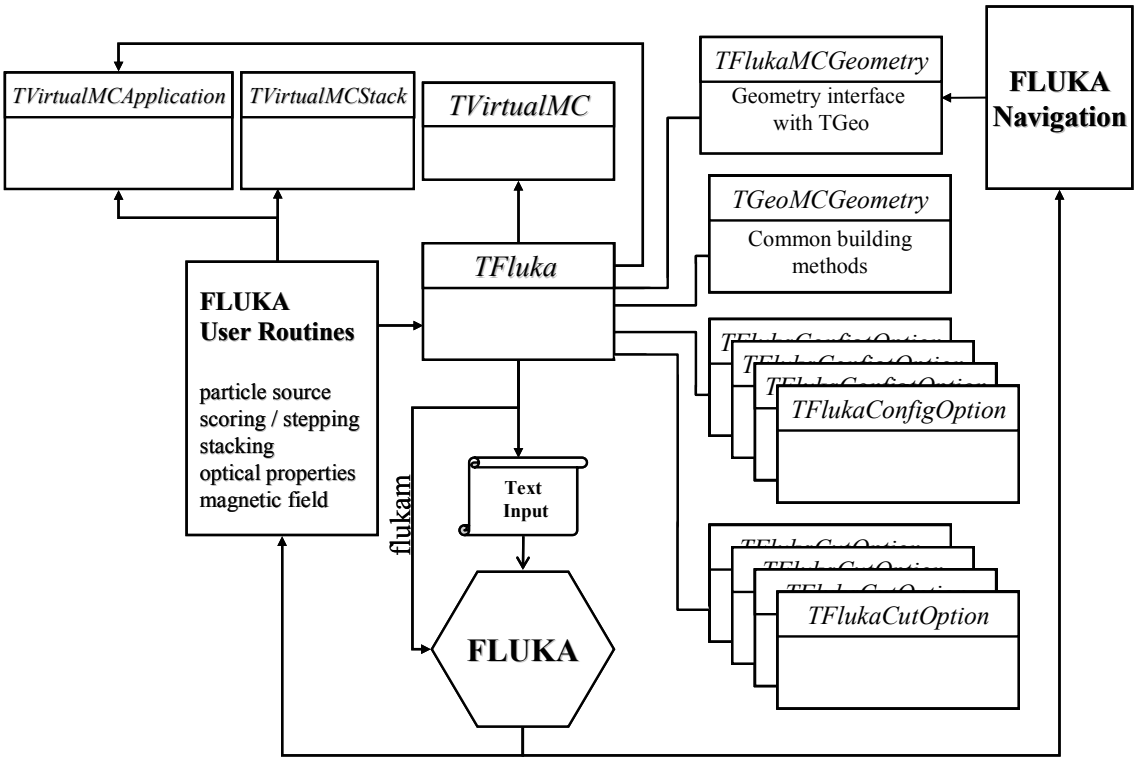

Figure 3*: TFluka* Implementation

Stacking

TVirtualMCStack does not replace the native FLUKA stack but acts as a monitor. The FLUKA routines *stupre* and *stuprf* that have the responsibility to copy secondaries from the stacks for electromagnetic and hadronic interactions, respectively, to the main stack, are used to copy particles to the VMC stack. A special case is Cerenkov photon production. Cerenkov photons are recorded at production in the FLUKA routine *crnkvp*.

Geometry

An important milestone in the development of the *TFluka* implementation has been the replacement of the FLUGG interface to the GEANT4 geometry modeller by an interface to *TGeo*. Tests with the ALICE geometry have shown that FLUGG+GEANT4 is unstable and suffers frequent crashes. The interface to *TGeo* has been implemented through the class *TFlukaMCGeometry*. This class wraps the FLUKA geometry navigation routines that delegate their tasks to corresponding *TGeo* methods. *TFluka* itself delegates all geometry related tasks to *TFlukaMCGeometry*, for example the writing of the material to region assignments for the FLUKA input.

PEMF Data Files

For the simulation of electromagnetic processes FLUKA needs a data file created by the PEMF pre-processor program. This file contains pre-processed data for all materials assigned to regions. Since the development to automate this procedure inside FLUKA is still ongoing we use *TFlukaMCGeometry* to prepare automatically and on demand the PEMF input files and run the pre-processor.

Optical Properties

FLUKA does not manage and store detailed user data on material optical properties but delegates this task to user routines, *abscff, dffcff, queffc, rflct, rfrndx,* for respectively, absorption coefficient, diffusion coefficient, quantum efficiency, reflectivity, and refraction index. An elegant way to manage optical properties in FLUKA VMC is to attach for those materials that have user defined optical properties objects of type *TFlukaCerenkov* to the *TGeo* class *TGeoMaterial* (Fig. 4).

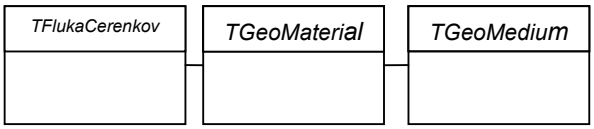

Fig. 4: Storing material optical properties.

Validation

Validations of *TFluka+TGeo* within the AliRoot simulation framework and within a specialized test-suite implemented as a *TVirtualApplication* have started. Within AliRoot we performed first comparisons on the hit level between *TFluka* and *TGeant3*. This allows us to test that the program is technically working correctly. Very stringent testing of the *TFluka+TGeo* geometry has started on point-by-point comparisons between *TFluka* and FLUKA at boundary crossings. As shown in Fig. 5 first results with a vacuum-filled geometry are very satisfying as we find differences on the level of the floating-point machine accuracy. More tests will be necessary before the geometry can be considered validated and they are ongoing.

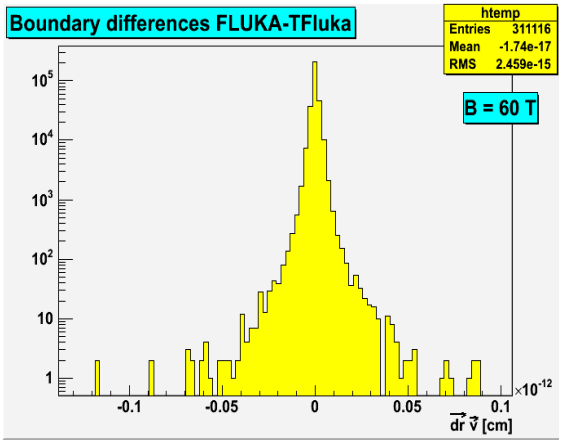

Fig.5: Exemplary result of a point-by-point comparison between *TFluka+TGeo* and FLUKA boundary crossings. $\vec{v}$ is the particle direction and $d\vec{r}$ the difference between the FLUKA and *TFluka* crossing points**.**

CONCLUSIONS

Implementations of the Virtual MC for the three main transport codes GEANT3, GEANT4 and FLUKA have been almost completed. *TGeant3* is used in production by the ALICE collaboration. The GEANT4 VMC implementation is complete, but ALICE plans to implement a GEANT4 navigation layer based completely on the *TGeo* modeller in order to use the *TGeo* modeller in an uniform way with all three VMC implementations. Most of our effort went into the implementation of *TFluka* and in particular in its interfacing with *TGeo*. *TFluka* is already used within the AliRoot framework in which it will undergo stringent testing.

REFERENCES

[1] I. Hřivnáčová *et al.*, Proceedings of Computing in High Energy and Nuclear Physics, La Jolla (2003).

[2] R. Brun *et al.*, GEANT3 User Guide, CERN Data Handling Division, DD/EE/84-1 (1985).

[3] S. Agostinelli et al., Nucl. Instrum, and Methods **A506** (2003), 250-303.

[4] A. Fassò *et al.*, Proc. of the MonteCarlo 2000 Conference, Lisbon, Springer Verlag Berlin (2001) p. 159-164 and p. 955-960.

[5] http://root.cern.ch/root/doc/RootDoc.html.

[6] I. Hřivnáčová, "*GEANT4 in the AliRoot framework*", Proc. of Computing in High Energy and Nuclear Physics 2001, Science Press Beijing New York, p. 534.

[7] M. Campanella *et al.*, ATLAS Internal Note, ATL-SOFT 98-039 (1998).

[8] A. Gheata, M. Gheata, and R. Brun, Proceedings of Computing in High Energy and Nuclear Physics, La Jolla (2003).

[9] I. Hřivnáčová, "*The Virtual Geometry Model*", these proceedings.

[10] R. Chytracek, "*The Geometry Description Markup Language*", CHEP'01, Beijing, September 2001, 8-009. http://gdml.web.cern.ch/gdml/.