

Mokka, main guidelines and future

Paulo Mora de Freitas

Laboratoire Leprince-Ringuet - Ecole polytechnique/IN2P3

Since 1999 we have developed Mokka [1], a Geant4 [2] full event simulation allowing a complete reconstruction and energy flow analysis to evaluate the detector designs for the Future Linear Collider (FLC). For the first time the next Mokka release includes code developed elsewhere and committed directly into the Mokka CVS repository in coordination with us. It's so time to recall some Mokka main guidelines which should (or not) be held for this new Mokka era.

1 Mokka main guidelines

Three major guidelines have guided us while developing Mokka:

1.1 Let's compare detector designs in a common framework !

The time being software simulation is the only way to compare performances of different detector designs currently in consideration for the FLC. To insure that different performances come only from the detector designs all detector models should be simulated in the same simulation framework. It means to simulate the same physics with the same simulation tool, to accept the same generator data files as input. Moreover, keeping the all in a common framework insures that the simulation output data is written in a common data format, so the users are able to apply almost the same reconstruction and analyze programs for all detector designs.

In Mokka this requirement is successfully implemented thanks to an external geometry database. "Geometry drivers", pieces of C++ code, act as "pluggins" to connect detector parts to the simulation framework. It insures that the Mokka Kernel doesn't depend anyway on the simulated geometry, so all detector proposals can share the same Geant4 machinery, the same input and output file formats and so on.

1.2 Keep it simple, stand alone as possible and available for all !

New approaches for what should be the best detector for the F.L.C. have to be considered. Several groups around the world are interested to contribute with the F.L.C. project and they need a common simulation tool working as a reference test bed for their new ideas. Doors should be kept wide open for all these groups. The common simulation framework has to be stand alone as

possible, available for all, just relying on free available software and standard building tools. To keep a native ASCII output format insures that it can be read anywhere. Open wide access to the simulation code has to be provided.

Following this guideline Mokka depends only on Geant4 and MySQL [3] free available softwares. Mokka is LCIO [6] compliant but it keeps always an ASCII native file format. Since 2003 the Mokka sources and documentation are available from the Mokka web site [4], everyone is able to download and build it just issuing the standard “gmake” command.

1.3 Keep trace !

To make the best choices is not enough if we cannot reproduce the simulations used to decide it. The time being the simulation for the F.L.C. detectors acts as experiments where impacting results have to be reproducibles anywhere else. For a software it means to keep trace of its development process via a tag police and full release notes. Geometry descriptions of released detector models have to be frozen, improvements should let to new models instead geometry modifications in an old one.

The Mokka code is kept in a cvs repository and all released tags can be checked out. Just one reference geometry database [5] is available over broad thanks to the MySQL client-server model. Local database copies can be useful while developing new detector models, but if significant results are achieved the new model has to be released in the reference database. Thanks to this centralized approach output data can refer to the model name, run log files can keep all the necessary information to reproduce a simulation run (detector model, Geant4 and Mokka release tag ids, line command parameters, etc).

2 What we learned for these five years

Along these five years several real use cases in detector studies for the F.L.C. let us to re-engineer Mokka a few number of times. We resume here the reasons for these revisions as it can be useful for future developments.

2.1 Geometry have to be shared

Simulation, reconstruction and analysis are independent applications which have to share a common detector geometry at run time to insure their coherence. For this reason the geometry layer of Mokka became a general propose “Common Geometry Access interface” (CGA) to make available the simulated detector geometry for reconstruction, analysis and visualization jobs. CGA

provides application interfaces (API) for F77, C, C++ and Java for programmers developing reconstruction and analysis code. CGA implements a few reconstruction utilities and it should be extended in the future, to provide access for information in the geometry database concerning specifically reconstruction and analysis tasks.

2.2 Data format have to be shared

Indeed the native ASCII file format insures that everyone in the world can read the Mokka output, of course it's not the best format to save disk space and to organize data. Moreover, the ASCII file format evolves with the software development so reconstruction and analysis applications depend on the Mokka release to run correctly. A better solution is to adopt a persistence model providing an abstract interface to access data. For Mokka it's done thanks to LCIO, which provides APIs for F77, C, C++ and Java.

2.3 New users ask for new functionalities and use cases !

The initial Mokka specification approach was to keep a clear separation between generation, simulation, reconstruction and analysis jobs. Recently new users requested to be able to plug analysis as user actions inside the simulation job. In the same way Mokka were specified to be controlled by command line parameters and new users asked to control simulation jobs via steering files. Both new user facilities are now implemented thanks to Frank Gaede (DESY) and it illustrates that new users ask for new functionalities in use cases, so the simulation framework architecture for the F.L.C has to be kept as open as possible.

2.4 We have to work together !

The big improvement thanks to the LCIO file format and the recent contributions from Frank Gaede (DESY) and Jeremy McCormick (NICADD) illustrates the power of joining efforts to develop a common simulation framework for the F.L.C. Moreover, only detector specialists are able to describe correctly and to verify the simulation results for the detector part they are concerned. So we have to improve working together by an informal developers committee and its organization around the CVS repository management, code standards, Geant4 - FLC users interface, to bring adequate level of detector descriptions to this common simulation tool.

3 Conclusion

”Improve framework” but ”Keep things simple”. ”Let’s compare detector designs in a common framework”, to do this ”We need adequate level of detector descriptions”. So... Let’s work together !!!

References

1. For a full description of Mokka and its status rapport see the previous LCWS04 talk “Geant4 simulation for the FLC detector models with Mokka” by Gabriel Musat. For an historical view, see also the one for the LCW2000 at <http://www-lc.fnal.gov/proceedings/proceedings.html#simulation/Freitas.ps>.
2. Geant4 Home Page, <http://wwwinfo.cern.ch/asd/geant4/geant4.html>.
3. A free of charge and almost standard SGBD server available for Lynux and Windows systems, see <http://www.mysql.com/>.
4. <http://polywww.in2p3.fr/geant4/TESLA/www/MOKKA/MOKKA.html>.
5. <http://polywww.in2p3.fr/geant4/TESLA/www/MOKKA/software/database/phpMyAdmin/index.php3>
6. LCIO, a persistency framework for future linear collider detector simulations, <http://www-it.desy.de/physics/projects/simsoft/lcio/>