Analysis Tools in Geant4

Ivana Hřivnáčová[1,*] and *Guy* Barrand[1,**]

[1]Laboratoire de Physique des 2 infinis Iréne Joliot-Curie (IJCLab), Université Paris-Sud, CNRS-IN2P3, Orsay, France

Abstract. The analysis category was introduced in Geant4 more than ten years ago (in 2011) with the aim to provide users with a lightweight analysis tool, available as part of the Geant4 installation without the need to link to an external analysis package. It helps capture statistical data in the form of histograms and n-tuples and store these in files in four various formats. It was already presented at CHEP multiple times, the last time five years ago. In this article we give an update on its evolution since then.

We will report on new functionalities: the connection of the analysis to visualization, flexibility in the selection of the output files and also saving data in multiple formats from the same simulation run, and new support for data object cycles in the latest version Geant4 11.1.

We will then present the evolution of its design including the major update in the past two years that allowed the introduction of a new Generic analysis manager. In particular, we will discuss the advantages of our design choice based on the so-called Non Virtual Interface pattern: the code robustness and stability in the context of the code evolution over more than ten years.

Finally, we will present the continuous code improvements using static code analysis and sanitizer tools.

1 Introduction

The analysis category was introduced in Geant4 [1] in the release 9.5 (2011) with the aim to provide users with a lightweight, easy to use, analysis tool, available as part of the Geant4 installation. It helps capture statistical data in the form of histograms and n-tuples and store these in files in four various formats. It was already presented at the CHEP conference multiple times [2] - [4]. In this article we will give an update on its evolution since Geant4 10.5 to 11.1. The implementation details can be found in the Geant4 Release notes [5] and the guidelines for users in the Geant4 User's Guide for Application Developers [6].

The analysis API was designed with the goal of maximum simplicity of its use. The key features of this design, listed below, allow the users to define and manipulate their analysis objects with a minimum number of lines of code or user interface commands:

- All user tasks are managed through a single manager class.
- Objects are accessed using integer indices.

*e-mail: ivana.hrivnacova@ijclab.in2p3.fr
**e-mail: guy.barrand@ijclab.in2p3.fr

Table 1. Evolution of principal features since the first version

First version - 2011	Current version - 2023
1,2 - dimensional histograms	1,2,3 - dimensional histograms; 1,2 - dimensional profiles
single n-tuple n-tuple columns of `int`, `float` and `double` types	no limitation on number of n-tuples + n-tuples columns of `string`; n-tuples columns of `vector` of all these types
single file output Csv, ROOT, Xml, HBOOK	multiple file output including multiple formats Csv, ROOT, Xml, HDF5

- After objects booking, users are required to perform a very limited number of manager functions: `OpenFile()`, `Write()`, and `CloseFile()`. The complexity of more advanced operations is then performed automatically behind these calls and gets completely hidden to users, such as, for example, the merging of histograms during the `Write()` call.

2 Basic Functionalities Overview

The Geant4 Analysis category has undergone a successive evolution in its basic functionalities since its first version in 2011, that is summarized in table 1. Initially the analysis tools provided 1-dimensional and 2-dimensional histograms and the ability to generate a single n-tuple. This n-tuple featured columns of `integer`, `float`, and `double` data types. These objects could be written and read in four formats: CVS, ROOT [7], XML, and the now-discontinued HBOOK, with the limitation of a single output file per application run.

In its current state, as of 2023, the analysis tools also provide 3-dimensional histograms, 1-dimensional and 2-dimensional profiles, n-tuples with columns of `string` types and of vectors of all the supported fundamental and `string` types. The restriction of a single n-tuple was removed shortly after the first version. Users can save their analysis data in multiple files and histograms and profiles also in files of multiple output formats. The support for HDF5 [8] output format, widely used in other domains of science than HEP, was added in 2018 [4].

Multi-threading and MPI

The analysis code has been adapted for multi-threading along with all other Geant4 categories in Geant4 major release 10.0 (2013). Automatic histogram merging with multi-threading (MT) debuted in the same version and was enhanced to MPI two years later. The ability to merge n-tuples on-the-fly, both under MT and MPI frameworks, utilizing the ROOT format, was achieved in the next two years [3]. Moreover, the flexibility of merging strategies was introduced, enabling users to choose between column-wise and row-wise merging.

3 Recent Features

In this section we will briefly describe the recent features added since 2021.

3.1 Batch and Interactive Graphics

The Geant4 visualization system was equipped to be able to do plotting, then to have a representation (a plot) of 1-dimensional or 2-dimensional histograms within a Geant4 visualization

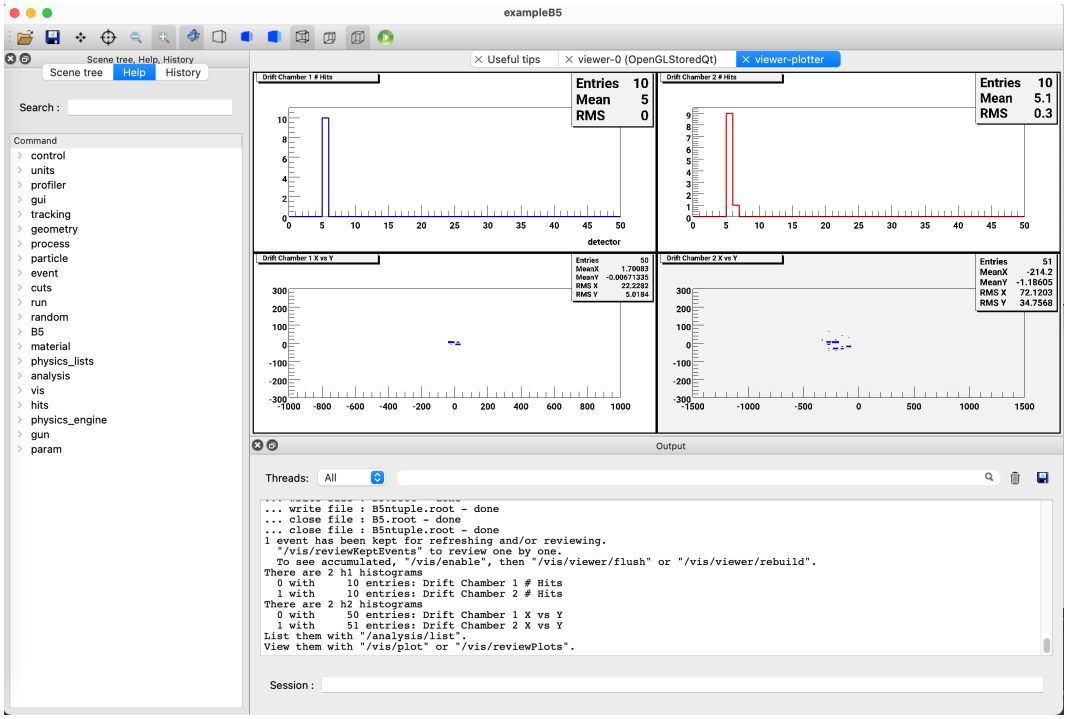

Figure 1. Example of histograms plots in the Geant4 graphical user interface.

viewer. This allows users to dynamically explore their data patterns. Currently, interactive plotting is exclusive to the new ToolsSG visualization driver.

The integration of histograms plots in the Geant4 graphical user interface is illustrated in figure 1. This new feature complements so-called batch plotting, already available since 2016. Users can activate plotting for selected histograms or profiles, the resulting plots will be then saved in a Postscript file without any drawing to the screen.

3.2 Multiple File Outputs

The Geant4 Analysis category has expanded its versatility by allowing users to specify separate output files for designated objects. The flexibility of mixing output types is granted for histogram and profile objects, while n-tuples are limited to a single output type. Users are required to provide file names with suitable extensions (`.csv`, `.hdf5`, `.root`, or `.xml`), unless a default file type is defined via a dedicated function. The corresponding user interface commands are also available, ensuring ease of use.

3.3 Object Cycles

Object cycles offer users the ability to write the same histogram, profile, or n-tuple to a file multiple times. This is achieved by the introduction of automated cycle numbering of all objects. While this functionality is naturally supported by the ROOT Input/Output (IO) system, for other output formats, the cycle number is appended to the object name by the analysis classes.

The introduction of object cycles facilitates user data organization by avoiding the necessity to create a new file with a different file name with each analysis manager `Write()` call.

An example of a Geant4 macro with two write cycles in one file, which is enabled by this new feature, is given in figure 2. The highlighted user interface commands for file operations (`openFile`, `write` and `closeFile`) used in this macro were introduced with the implementation of object cycles for higher flexibility.

```
/run/initialize
/analysis/openFile B5.root
#
/run/beamOn 30
/analysis/write
/analysis/reset
#
/run/beamOn 30
/analysis/write
/analysis/reset
#
/analysis/closeFile
```

Figure 2. An example of a Geant4 run macro with two write cycles.

4 Design Evolution

The Geant4 Analysis category's design has naturally evolved with the addition of new features and functionality. Each iteration improved code reuse as well as code maintainability.

4.1 First Design (2011)

The first design [2] was based on a simple inheritance. All interfaces were defined in the common base class, `G4VAnalysisManager`, and they were implemented in the output specific manager classes.

This facilitated uniformity and ease of use, as all managers could be accessed through a generic `G4AnalysisManager` type. Internally, this type was defined via a `typedef` through dedicated header files, `g4csv.hh`, `g4root.hh`, `g4xml.hh` and `g4hbook.hh`, and it pointed to one of four output type specific manager classes.

4.2 NVI Design Pattern (2013)

Design evolution took a significant step forward in 2013 (see also [2]) splitting the monolithic manager classes into smaller classes per analysis object type and with the incorporation of the so-called Non-Virtual Interface (NVI) pattern [9]. This pattern refined the structure by employing non-virtual public methods in the `G4VAnalysisManager` class, which in turn called protected, pure virtual functions in the friend component classes. These component classes were responsible for output-specific implementations.

This step greatly strengthened the stability of the public interface and allowed further developments without impacting user code.

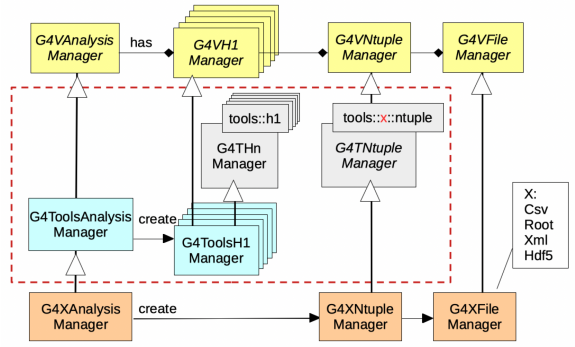

Figure 3. G4ToolsAnalysisManager design iteration

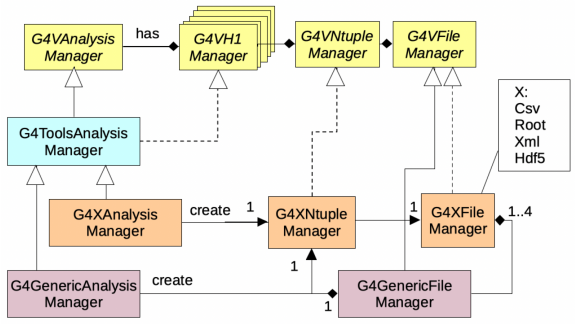

Figure 4. G4GenericAnalysisManager design iteration

4.3 Tools Analysis Manager (2017)

The introduction of the `G4ToolsAnalysisManager` class in 2017, see figure 3, added efficiency. This class is used for common handling of all histograms and profiles. Besides introducing this class, code duplication has also been mitigated through template integration, thereby improving the maintainability and clarity of the code. It's worth noting that these changes concerned only the analysis classes internals and did not affect the API seen by the users.

4.4 Generic Analysis Manager (2020)

After further reorganization of file and n-tuple managers, a generic analysis manager, `G4GenericAnalysisManager`, could be added, see figure 4. This class assumes the role of the primary analysis manager, capable of hosting multiple file managers and it superseded the output specific managers. This update permitted mixing output file formats for histograms and profiles within the same application run discussed in section 3.2. The single n-tuple file manager remained in order to preserve code simplicity and efficiency.

In Geant4 major release 11.0, the generic analysis manager became the default `G4AnalysisManager` type via a new dedicated header file, `G4AnalysisManager.hh`. This new header replaced the output specific headers (`g4csv.hh`, `g4root.hh`, `g4xml.hh` and `g4hbook.hh`), as well as the previously introduced alternative factory methods, that could then be removed. The output specific managers can be still used by directly including their class headers.

4.5 User Interface Stability

The design choice, which involved adopting a non-virtual public interface within the `G4VAnalysisManager` class, proved to be a nice success in terms of maintaining user code stability. Over the past decade, this design has significantly minimized the need for user code migration and required transitions in user code have been remarkably few.

Below we list these cases that required migration efforts:

- Version 10.3: Discontinuation of the support for the HBOOK output format.
- Versions 10.4, 10.5, 10.6: These releases introduced changes to n-tuple merging settings, incorporating additional or modified arguments to specify merging modes.
- Version 11.0 (major Geant4 release):
  - The output format specific headers and earlier introduced alternative factory methods were removed in favour of the new `G4AnalysisManager.hh` header file.
  - Migration to `G4ThreadLocalSingleton` in all analysis manager and reader classes. As the singleton instances are now being deleted by the Geant4 kernel, their explicit deletion in client code had to be removed.

5 Code Quality and Modernization

In the Geant4 Analysis category, we have always focused on code quality and modernization in line with contemporary software development standards.

This process is facilitated by the Geant4 software infrastructure including continuous integration testing, the Coverity static analysis tool and also its evolving Coding guidelines following new C++ standards with updates for C++11 and C++17.

Besides regular testing in many examples, there are four test codes dedicated to the analysis category:

- `test03` covers testing of all object types with most of the configuration options including generic and all output specific managers.
- `test32` is dedicated to testing multiple file outputs.
- `test320` is a new test, that will be released in 11.2, for testing object cycles and, also new in 11.2, deleting selected objects.
- `test08` covers the `accummulables` sub-category, the functions of which are not discussed in this article.

The analysis category developments were aligned with C++ standard upgrades along with the other Geant4 categories. Since 2015, it was the C++11 upgrades, incorporating constructs like `auto`, `nullptr`, `shared` and `unique` pointers, `using` directives, and `deleted` constructors. Subsequently, since 2021, the C++17 features like structured binding or `filesystem` were progressively introduced.

The code modernization was significantly facilitated by the application of the Clang-tidy [10] tool, in accordance with Geant4's coding guidelines, namely by applying checks from the `performance-`, `modernize-`, and `readability-` families.

6 Conclusion

The Geant4 analysis tools have been available to users for over a decade. It started with a simple set of classes providing a limited set of functionalities and it has been continuously enhanced with new features almost every year.

In this article, we gave an overview of its recent developments. Integrating histogram plotting into the Geant4 visualization system allows users to view their results directly in the Geant4 graphical user interface without the need to open an external analysis tool. New flexibility in file output, including mixing file types within the same simulation, and support for object cycles met user demand in Geant4 User Forum.

We also discussed the evolution of its design and briefly presented the initial design and our motivations for three major iterations and their benefits. We particularly emphasized the central role of our design choices in facilitating the evolution of the tool with minimum impact on the user code.

References

[1] Agostinelli S et al, Nucl. Instrum, and Methods **A506**, 250-303 (2003)
Allison J et al, IEEE Transactions on Nuclear Science **53 No. 1**, 270-278 (2006)
Allison J et al, Nuclear Instruments and Methods in Physics Research **A835**, 186-225 (2016)

[2] Hrivnacova I, J. Phys.: Conf. Ser. **513**, 022014 (2014)

[3] Hrivnacova I, Barrand G, J. Phys.: Conf. Ser. **898** 042018 (2017)

[4] Hrivnacova I, Barrand G, EPJ Web Conf., **214** 02009 (2019)

[5] https://geant4.web.cern.ch/download/all

[6] https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/index.html

[7] http://root.cern.ch

[8] https://portal.hdfgroup.org

[9] Sutter H, Alexandrescu A, *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices* (Boston, MA: Addison-Wesley, 2005) p.68

[10] https://clang.llvm.org/extra/clang-tidy/