



# New Functionalities and Updates to Analysis Module in Release 11

I. Hrivnacova, G. Barrand,  
IJCLab Orsay (CNRS/IN2P3)

[Virtual] 26<sup>th</sup> Geant4 Collaboration Meeting,  
23 September 2021

# Outline

- Updates in g4tools
- Updates in analysis
  - Support for multiple output files
  - Migration to Generic analysis manager
  - Developments/Fixes in 11.0
  - Other Migration Items
- Work plan items

# *g4tools in G4-11*



- Have a MR to prune g4tools to have in it only what is really used in G4/analysis and now G4/vis/ToolsSG. Based on the #63 issue of Ben to ease the maintenance and integration of g4tools, see: <https://gitlab.cern.ch/geant4/geant4-dev/-/issues/63>
- (A lot of files (around 100) here for g4tools/tests, but these can stay in gbarrand/github/g4tools).
- ? Have a MR to have “toolx” namespaced code for code related to “externals” (now needed for X11, Windows, GL, png, jpeg). It should ease Ben-#63.

# *G4/vis/plotting in G4-11?*

- All is here to have interactive plotting.
- (We have already tools/sg offscreen plotting in G4/analysis).
- What is the right way to do it in G4? Through G4UI or G4/vis?
- I had (since 2018) a “G4UI way”, but it is quite not satisfactory.
- A G4/vis way would be better since the logic of attaching a windowing and a rendering system, creating viewers and scenes is already here.
- After all a plot is nothing more than the visualisation of an histogram. And then some “G4VSceneHandler::AddPrimitive(<histo>)” should be able to do it!
- Relationship with G4/analysis?
- Have to be discussed with John and Ivana...

## *As a reminder...*

- g4tools is an automatic extraction of some code found in the softinex/inlib,exlib and namespaced “g4tools” for an embedding in Geant4.
- Pure header code. Highly portable (including iOS, Android and now WebAssembly). Easily embeddable (no “config.h” or specific build tool in the way).
- Strongly OO. No implicit management. Layered.
- Thread safe (no writable statics).
- See <http://gbarrand.github.io>

# New Developments in 10.7

# Using Multiple Files in 10.7

- Users can choose the output file per object via new analysis manager functions:

```
void SetH1FileName(G4int id, const G4String& fileName);  
    //... etc for H2, H3, P1, P2  
void SetNtupleFileName(G4int id, const G4String& fileName);
```

- The file name can be provided without an extension, then the default file type must be set via

```
void SetDefaultFileType(const G4String& value);
```

- Multiple output types can be used for histograms and profiles, only one output type for ntuples
  - This limitation is going to be removed in 11.0
- The corresponding UI commands are also available

# G4GenericAnalysisManager

- Handling more files by analysis manager required a deeper redesign of the manager classes
  - New templated classes were introduced in for separation of the objects output from the analysis managers (presented in the analysis talk in the last Collaboration Meeting)
- New classes in the 'factory' sub-category:
- **G4GenericAnalysisManager** - takes the role of the top analysis manager class, it has
  - **G4GenericFileManger** handles output specific managers for file management (the file managers for all output types can coexist)
  - One output type dependent manager for ntuple management (in 10.7), coexisting types foreseen for 11.0

# New Instance Method in 10.7

```
#include "G4GenericAnalysisManager.hh"

using G4AnalysisManager = G4GenericAnalysisManager;

// Create/Get analysis manager
auto analysisManager = G4AnalysisManager::Instance();
```

NEW in 10.7  
basic/B5

G4AnalysisManager =  
G4GenericAnalysisManager

# Other Instance Methods

```
//#include "g4csv.hh" } #include "B4Analysis.hh"  
//#include "g4xml.hh"  
#include "g4root.hh"  
  
// Create analysis manager  
// The choice of analysis technology is done via selection  
// of a namespace in B4Analysis.hh  
auto analysisManager = G4AnalysisManager::Instance();  
G4cout << "Using " << analysisManager->GetType() << G4endl;
```

basic/B4

G4AnalysisManager =  
G4RootAnalysisManager

```
#include "g4analysis.hh"  
  
// Create the analysis manager using a new factory method.  
// The choice of analysis technology is done via the function  
// argument.  
auto analysisManager = G4Analysis::ManagerInstance("root");  
G4cout << "Using " << analysisManager->GetType() << G4endl;
```

SINCE 10.6

G4AnalysisManager =  
G4ToolsAnalysisManager  
... base class for  
G4XyzAnalysisManager

# New Developments For 11.0

# Migration to Generic Analysis Manager

- The output specific analysis manager classes (G4CsvAnalysisManager, etc.) are superseded by G4GenericAnalysisManager
  - They can be dropped in favor of the new generic manager
- Migration in two phases:
- 1. phase (in 11.0)
  - The helper include files will define G4GenericAnalysisManager as G4AnalysisManager type (more details on the next slides)
  - The specific analysis manager classes will be deprecated, but besides the deprecation warning can be used as before
- 2. phase (next year):
  - The specific analysis manager classes will be removed
  - G4GenericAnalysisManager will be renamed in G4AnalysisManager

# Migration to Generic Analysis Manager in 11.0 - 1

- A new header file [G4AnalysisManager.hh](#) is provided
  - Its use will be demonstrated in B4 example

```
#ifndef G4AnalysisManager_h
#define G4AnalysisManager_h

#include "G4GenericAnalysisManager.hh"

using G4AnalysisManager = G4GenericAnalysisManager;

#endif
```

# Migration to Generic Analysis Manager in 11.0 - 2

- The applications using `"g4root.hh"`, etc. headers will get `G4GenericAnalysisManager` defined as `G4AnalysisManager`
  - No migration needed for code using file names with extension, the default output type `"root"`, etc. will have to be set in the applications using file names without extension
  - Can be also dropped and replaced with new `"G4AnalysisManager.hh"` include
- The applications using the `"g4analysis.hh"` header will get `G4GenericAnalysisManager` defined as `G4AnalysisManager`
  - The default output type is set according to the passed parameter
- Nothing will change for applications using already `G4GenericAnalysisManager` (example B5)
- The applications including `G4RootAnalysisManager.hh` directly will get `G4RootAnalysisManager` with a deprecation warning

# Instance Methods in 11.0 - 1

```
#include "G4AnalysisManager.hh"

// Create/Get analysis manager
auto analysisManager = G4AnalysisManager::Instance();

// Open an output file
G4String fileName = "B4.root";
// G4String fileName = "B4.csv";
// G4String fileName = "B4.hdf5";
// G4String fileName = "B4.xml";
analysisManager->OpenFile(fileName);
G4cout << "Using " << analysisManager->GetType() << G4endl;
```

NEW in 11.0  
basic/B4

G4AnalysisManager =  
G4GenericAnalysisManager

Implicit Using defined  
in G4AnalysisManger.hh

Output file name is provided  
with the extension

# Instance Methods in 11.0 - 2

```
#include "G4AnalysisManager.hh"

// Create/Get analysis manager
auto analysisManager = G4AnalysisManager::Instance();
analysisManager->SetDefaultFileType("root");
// analysisManager->SetDefaultFileType("csv");
// analysisManager->SetDefaultFileType("hdf5");
// analysisManager->SetDefaultFileType("xml");
G4cout << "Using " << analysisManager->GetType() << G4endl;

// Open an output file
G4String fileName = "B4";
analysisManager->OpenFile(fileName);
```

## NEW in 11.0

G4AnalysisManager =  
G4GenericAnalysisManager

Implicit Using defined  
in G4AnalysisManger.hh

File extension can be  
omitted if default file type  
is set explicitly

# Instance Methods in 11.0 - 3

```
#include "G4GenericAnalysisManager.hh"

using G4AnalysisManager = G4GenericAnalysisManager;

// Create/Get analysis manager
auto analysisManager = G4AnalysisManager::Instance();
```

As in 10.7  
basic/B5

Explicit Using in the  
user code

Output file name is provided  
with the extension

# Instance Methods in 11.0 - 4

```
#include "g4analysis.hh"

// Create the analysis manager using a new factory method.
// The choice of analysis technology is done via the function
// argument.
auto analysisManager = G4Analysis::ManagerInstance("root");
G4cout << "Using " << analysisManager->GetType() << G4endl;
```

## Deprecated in 11.0

G4AnalysisManager =  
G4GenericAnalysisManager

The default file type set  
in the factory function

File extension can be  
omitted

# Instance Methods in 11.0 - 5

```
#include "G4RootAnalysisManager.hh"

// Create/Get analysis manager
auto analysisManager = G4RootAnalysisManager::Instance();

// Open an output file
G4String fileName = "B4";
analysisManager->OpenFile(fileName);
```

Deprecated in 11.0

G4RootAnalysisManager

File extension can be omitted

# Developments/Fixes in 11.0 - 1

- Replaced the explicit `Write[Hn | Pn]` functions/ `Read[Hn | Pn]` functions in the output specific analysis managers/readers by a common implementation
  - The same code is now used for writing histograms/profiles by specific analysis managers and the generic analysis manager
- Added **support for ntuple columns of string vectors** via new public functions in `G4VAnalysisManager` and `G4VAnalysisReader`: and `G4VAnalysisReader`
- Simplified code issuing warnings and verbosity messages
  - **Reduced exception classification codes**
    - `Analysis_F001`: Fatal exceptions - all fatal exception
    - `Analysis_W001`: Warnings - all warnings

# Developments/Fixes in 11.0 - 2

- Code clean-up:
  - More C++11,14,17, clang-tidy modernize and performance fixes, consistent naming
- Bug Fixes:
  - Fixed the problem with merging ntuples with columns of vector type reported in G4 Forum
  - Fixed ntuple indexing when `FinishNtuple()` is called in different order than `CreateNtuple`. (Problem report [#2335](#))
  - Fixed compilation errors occurring with c++20 enabled (by G. Cosmo)

# Other Migration Items

- Migration to `G4ThreadLocalSingleton`: in all specific and generic analysis manager and reader classes.
  - After this migration the singleton instances are deleted by the Geant4 kernel, that's why their **deleting has to be removed from the client code**.
- Renamed `G4VAnalysisReader::GetNtuple(const G4String&)` in `ReadNtuple` to avoid ambiguity with `GetNtuple(G4int)`
  - `ReadNtuple()` function retrieves an ntuple for reading (via `GetNtupleRow()` calls), while `GetNtuple(G4int)` gives an access to an ntuple already read
  - Used in the `xray_fluorescence` advanced example

# Remaining Work Plan Items

- Additional flexibility in resetting/deleting histograms
  - Still considering for 11.0
- Review support for writing same histogram/profile on file several times
  - Will be moved to next year