Analysis

I. Hrivnacova, IJCLab Orsay

Geant4 IN2P3 and ED PHENIICS Tutorial,
16 – 20 May 2022, IJCLab

Outline

- Introduction
- Geant4 analysis tools
  - Using Geant4 analysis
  - Histograms, ntuples, analysis UI commands
  - Reader, batch and interactive graphics
- Using external analysis tools
  - ROOT, Plotting CSV

Geant4 Analysis Tools

Geant4 Analysis Tools

- Provides code to write **histograms** and **“flat ntuples**” in several formats:
  - **ROOT, XML AIDA format, CSV, HDF5**
- Based on g4tools from inlib/exlib developed by G. Barrand (LAL): http://inexlib.lal.in2p3.fr/
  - “Pure header code” - all code is inlined, can be installed, besides Geant4 supported platforms, also on iOS, Android
- Area of new developments and improvements: more features are added in each release
  - For example support for HDF5 output type in 10.4, support for multiple files output in 10.7

Using Geant4 Analysis

Basic steps:

1) Create G4AnalysisManager
2) Book (create) your histograms, ntuples
3) Open a file
4) Fill values in histograms, ntuples
5) Write & close file

1) Create Analysis Manager

- The analysis manager is created with the first call to G4AnalysisManager::Instance() function

RunAction.cc

```
#include "G4AnalysisManager.hh"

RunAction::RunAction()
{
  // Create analysis manager
  auto analysisManager = G4AnalysisManager::Instance();
  analysisManager->SetVerboseLevel(1);
  analysisManager->SetDefaultFileType("root");
}
```

If the default file type is set, the filenames can be provided without extension

2) Book (Create) Histograms

- Example of creating one-dimensional histograms

RunAction.cc

```
#include "G4AnalysisManager.hh"

RunAction::RunAction()
{
  // Create or get analysis manager
  // ...

  // Create histograms
  analysisManager->CreateH1("EDep","Energy deposit", 100, 0., 800*MeV);
  analysisManager->CreateH1("TLen","Track length",   100, 0., 100*mm);
}
```

3) Open a File

- Example of opening a file

RunAction.cc

```
#include "G4AnalysisManager.hh"

void RunAction::BeginOfRunAction(const G4Run* run)
{
  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // Open an output file
  analysisManager->OpenFile("MyFile");
}
```

The filename extension can be omitted as the default file type was set previously;
otherwise the full name "MyFile.root" should be provided

4) Fill Histograms

- Example of filling one-dimensional histograms

EventAction.cc

```
#include "G4AnalysisManager.hh"

void EventAction::EndOfEventAction(const G4Event* event)
{
  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // Fill histograms
  AnalysisManager->FillH1(0, fEdep);
  AnalysisManager->FillH1(1, fTrackLength);
}
```

5) Write & Close a File

- Writing & closing a file

RunAction.cc

```
#include "G4AnalysisManager.hh"

void RunAction::EndOfRunAction(const G4Run* run)
{
  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // Write and close the output file
  analysisManager->Write();
  analysisManager->CloseFile();
}
```

Using Geant4 Analysis

Basic steps:

1)	Create G4AnalysisManager	… in RunAction constructor
2)	Book (create) your histograms, ntuples	… in RunAction constructor
3)	Open a file	… in BeginOfRunAction()
4)	Fill values in histograms, ntuples	… anywhere (during event processing)
5)	Write & close file	… in EndOfRunAction()

Performing the steps in the suggested class&method is not obligatory, but it guarantees correct functioning in multi-threaded mode

Histograms

- 1D, 2D, 3D histograms and 1D, 2D profile histograms available
- Histogram identifiers
  - The histogram identifier (an integer value) is automatically generated when a histogram is created by the analysis manager CreateH1 function, and its value is returned from this function.
    - It is used eg. in FillH1(id, value)
  - The default start value 0 can be changed (eg. to 1) with the SetFirstH1Id(G4int) method.
  - The 1D, 2D and 3D histograms IDs are defined independently
- Histogram objects
  - It is also possible to access directly the histogram by the GetH1(G4int id) function.

```
G4cout << "Print histograms statistic \n" << G4endl;
G4cout << " EAbs : mean = " << analysisManager->GetH1(1)->mean()
       << " rms = " << analysisManager->GetH1(1)->rms() << G4endl;
```

Histogram Options

- The properties, additional to those defined in g4tools, can be added to histograms via G4AnalysisManager
  - Unit: if defined, all filled values are automatically converted to this defined unit
  - Function: if defined, the function is automatically executed on the filled values (can be log, log10, exp)
  - Binning scheme: users can define a non-equidistant binning scheme (passing a vector of bin edges)
  - ASCII option: if activated the histogram is also printed in an ASCII file when G4AnalysisManager::Write() function is called.
  - Activation: users can activate/inactivate selected histograms

Book (Create) an Ntuple

- An n-tuple is a sequence (or ordered list) of n elements (quantities), that may be of different types
- The sets are written in a file in a serial way: we often speak about **rows** and **columns**; row is one set recorded, a column is the set of all records of one quantity
- Example of creating an ntuple

RunAction.cc

```
RunAction::RunAction()
{
  // Create or get analysis manager
  // ...
  // Create ntuple
  analysisManager->CreateNtuple("MyNtuple", "Edep and TrackLength");
  analysisManager->CreateNtupleDColumn("Eabs");
  analysisManager->CreateNtupleDColumn("Labs");
  analysisManager->FinishNtuple();
}
```

Fill an Ntuple

- Example of filling an ntuple

EventAction.cc

```
void EventAction::EndOfEventAction(const G4Event* event)
{
  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // Fill ntuple
  analysisManager->FillNtupleDColumn(0, fEnergyAbs);
  analysisManager->FillNtupleDColumn(1, fTrackLAbs);
  analysisManager->AddNtupleRow();
}
```

Ntuples

- Ntuple and Ntuple Column Identifiers
  - Automatically generated when the ntuple or ntuple column is created by the CreateNtuple() or CreateNtupleTColumn() function and its value is returned from this function.
  - The default start value 0 can be again changed with the SetFirstNtupleId(G4int) and SetFirstNtupleColumnId(G4int) methods
- The ntuple column ID is not specific to the ntuple column type
- Available column types:
  - `integer` (I), `float` (F), `double` (D), `std::string` (S)
  - `std::vector` of these types

Analysis UI Commands

- Many settings can be performed with UI command
  - General options like the verbose level
  - Global and per object file and directory names
  - Histogram and profile properties per object
- Defined in `/analysis` directory
- See more details in the Analysis section in the Application Developer Guide

Generic Analysis Manager

- New since 10.7 and the default since 11.0, the analysis manager class is capable to mix output file formats
- Alias to G4AnalysisManager is defined in `G4AnalysisManager.hh`

G4AnalysisManager.hh

```
#ifndef G4AnalysisManager_h
#define G4AnalysisManager_h

#include "G4GenericAnalysisManager.hh"

using G4AnalysisManager = G4GenericAnalysisManager;

#endif
```

Generic Analysis Manager (2)

- Users can choose to write selected objects in a different file than the default one using the G4AnalysisManager functions:

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

- The setting can be also performed with UI commands
- While it is possible to mix output types for histogram and profiles objects, only *one output type is supported for ntuples*.

Output File(s)

- Depending on a selected file format, multiple output files can be produced
- ROOT, HDF5
  - All histograms, profiles and ntuples are written in one file
- XML
  - The histograms and profiles are written in one file
  - Each ntuple is written in a separate file
- CSV
  - Each histogram, profile and ntuple are written in a separate file
- File names are generated automatically:
  - fileName[_objectName].ext
    - where ext = xml, csv

Analysis Reader

- It allows to read in g4analysis objects from the files generated by the analysis manager during processing a Geant4 application.
- Available for each supported output format
  - There is no “Generic” analysis reader class allowing to mix reading types
  - The reader of output specific type must be included; e.g
    ```
    #include G4RootAnalysisReader.hh
    ```
- The histograms and profiles objects handled by the analysis reader are of the same type as those handled by analysis manager, the ntuple objects are of different types

Batch Graphics

- Users can activate plotting of selected histograms and profiles using G4AnalysisManager functions:

```
// Activate plotting of 1D histogram
analysisManager->SetH1Plotting(id, true);
   // etc for H2, H3, P1, P2
```

- Or via UI command

```
/analysis/h1/setPlotting id true|false
/analysis/h1/setPlottingToAll  true|false
   ## etc for h2, h3, p1, p2
```

- The selected objects will be plotted in a single postscript file with the page size fixed to A4 format

Plotting Style

- Set plotting style

```
/analysis/plot/setStyle  styleName
```

  - ROOT_default (default), hippodraw: high resolution fonts
  - inlib_default: low resolution fonts
  - High resolution fonts are available only if Geant4 libraries are built with the support for Freetype font rendering

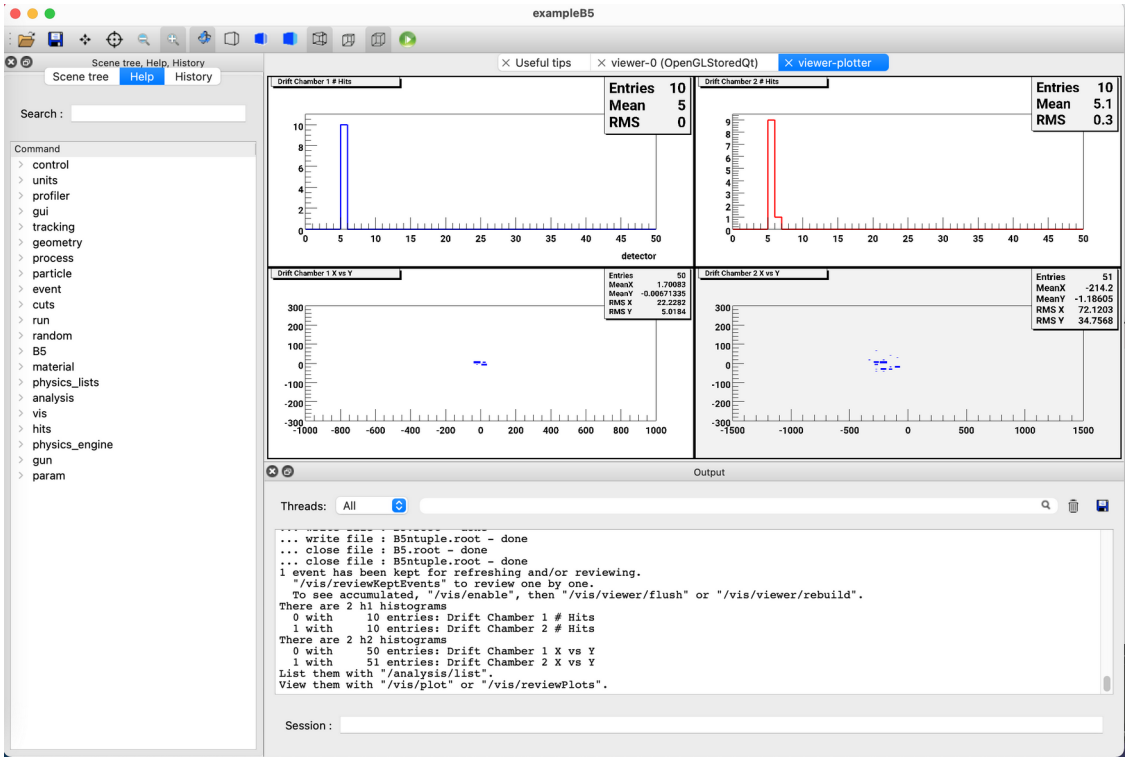
- The page layout

```
/analysis/plot/setLayout  columns rows
```

  - The number columns and the number of rows in a page.
  - The maximum number of plots is limited according to selected style.

Interactive Graphics

- Since version 11, the Geant4 visualization system is equipped to be able to do plotting, then to have a representation (a plot) of 1D or 2D histograms within a Geant4 visualization viewer.
- Currently only new **ToolsSG** visualization driver has this feature
- Demonstrated in the basic B5 example `plotter.mac` macro



Resetting Data

- By default the histogram and ntuple data are reset at the CloseFile() call at the end of run action
- But for interactive plotting we need to keep to have these data available – resetting can be postponed to the begin of the next run

```
void RunAction::BeginOfRunAction(const G4Run* run) {
  // ...
  analysisManager->Reset();
  analysisManager->OpenFile("MyFile");
}

void RunAction::EndOfRunAction(const G4Run* run) {
  // ...
  AnalysisManager->Write();
  analysisManager->CloseFile(false);
}
```

Analysis of Generated Files
With External Tools

Plotting ROOT files
... with ROOT

ROOT
https://root.cern.ch/

ROOT is a powerful analysis tools which provides

- histogramming and graphing to view and analyze distributions and functions
- curve fitting (regression analysis) and minimization of functionals, statistics tools used for data analysis,
- matrix algebra, four-vector computations, standard mathematical functions, multivariate data analysis, e.g. using neural networks,
- persistence and serialization of objects, which can cope with changes in class definitions of persistent data, creating files in various graphics formats, like PostScript, PNG, SVG
- 3D visualizations (geometry), image manipulation, used, for instance, to analyze astronomical pictures
- access to distributed data (in the context of the Grid), distributed computing, to parallelize data analyses, access to databases,
- ...

Viewing ROOT Files

- Start a ROOT session

```
$> root
```

- Open a ROOT browser in the ROOT interactive shell

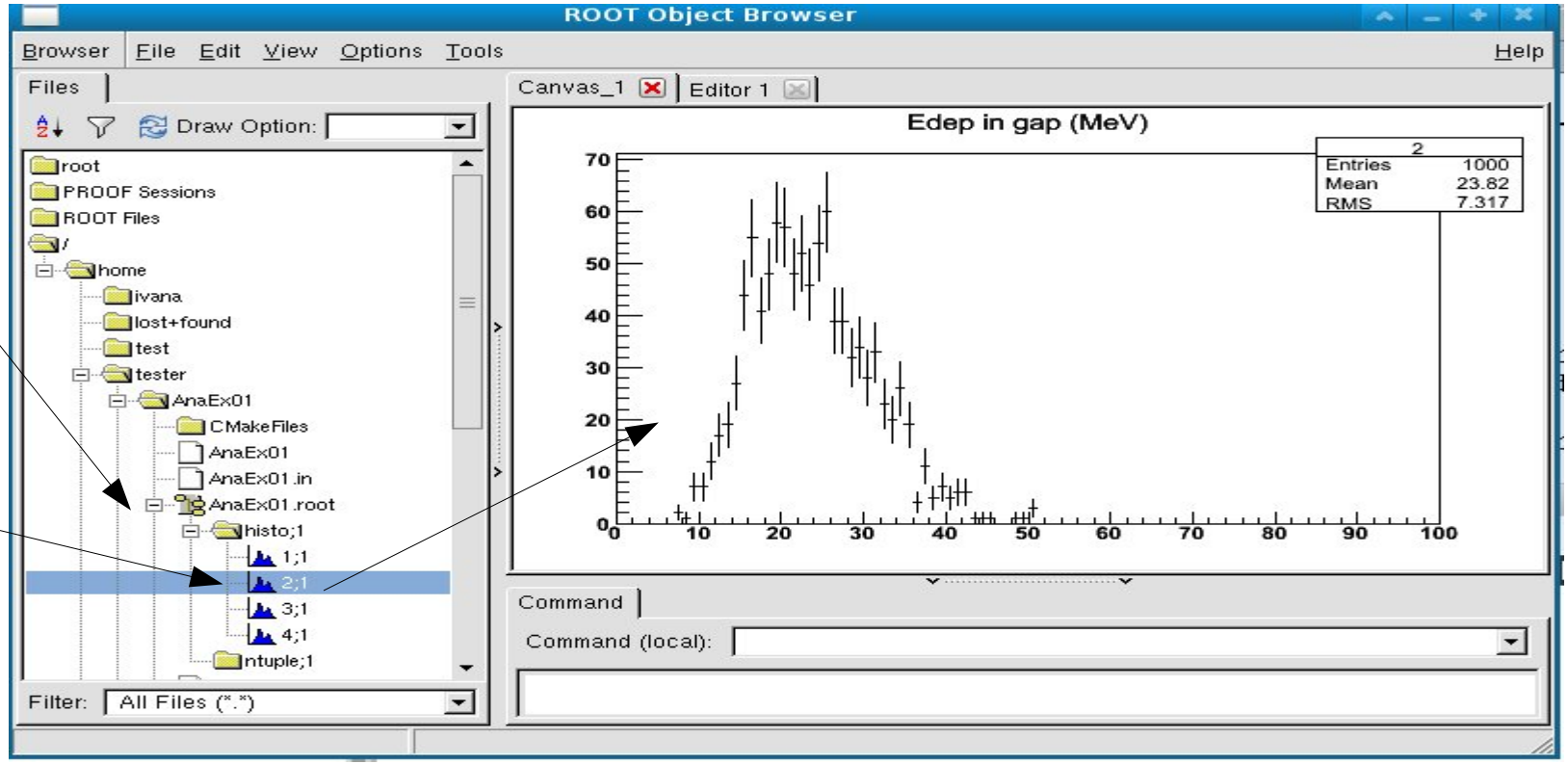
```
root [0] TBrowser b;
```

- See the ROOT documentation
  - How to edit histogram properties
  - How to open Fit panel
  - How to write ROOT macros

Viewing ROOT Files (2)

File generated
In Geant4
simulation

Selected H1
is automatically
drawn in the
canvas



Using ROOT in a Geant4 Application

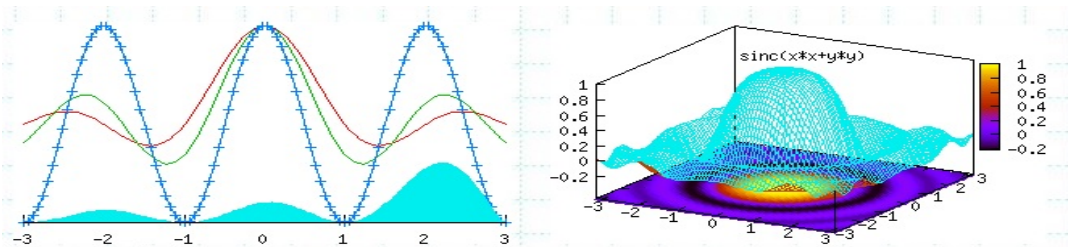
- Extended examples analysis/AnaEx01, 02 demonstrate how to make histograms and ntuples with g4analysis (AnaEx01) and ROOT (AnaEx02)
  - https://geant4-userdoc.web.cern.ch/Doxygen/examples_doc/html/Examples_analysis.html (link)
- The same scenario is implemented in shared classes
- All histogram/ntuples manipulations are located in one class: HistoManager, implementation of which is different in each example.
- The CMake configuration file demonstrates how to link Geant4 application with ROOT

Analysis of Generated Files
With External Tools

Plotting CSV Files
GNUplot, Excel, Open[Libre]Office

Plotting CSV Files

- Gnuplot is a portable command-line driven interactive data and function plotting utility
  - http://www.gnuplot.info/
- Excel:
  - The .cvs file can be imported as a text file and then processed as the data in spreadsheet
- Open[Libre]Office
  - The .csv file can be open from the “File” menu as “Text CSV” file
  - http://www.openoffice.org/
  - http://www.libreoffice.org



Examples of plots from Gnuplot
Thanks to J. Perl, M. Kelsey (SLAC)

Summary

- Geant4 provides a lightweight analysis tools as part of distribution
- The Geant4 analysis is now used in all basic, extended and most of advanced examples
- Users can also choose to use an external package and link their application against its libraries