Analysis Functionalities

20-23 June, 2016
Geant4 tutorial @ MIT
Whitaker Building – Room 56-154

A. Dotti
(adotti@slac.stanford.edu)
Acknowledgment and Notes

Material from Ivana Hrivnacova, IPN Orsay

• New module in Geant4 since Version 10.0
• Area of new developments and improvements: more features are added in each release
  • Example from latest release: better MPI support

Geant4 Examples available in:
<g4src>/examples/extended/analysis
Overview

• Analysis of simulation data in Geant4
• New histogramming and ntuples tool
• UI Commands for analysis
Introduction
Introduction

• Geant4 does not provide a complete analysis sub-system
  - Our user community is too heterogeneous
  - Each user group has its own requirements and a favorite tool (e.g. ROOT in HEP, what is yours?)

• Typical simulation output consists of: ntuples-like tables (row: event, column: quantity), histograms
Information for past-versions

Historically (up to 9.5) was discussed in Geant4 examples:

• Based on **AIDA** = Abstract Interfaces for Data Analysis

• List of AIDA compliant tools (linked in the Geant4 Guide for Application Developers):
  • JAS, iAIDA, Open Scientist Lab, rAIDA
  • Not all kept maintained, not all implement the AIDA interfaces completely. Not always easy to install & use

• Feedback and help: Geant4 user forum, Analysis category

• Still supported in Geant4 Version 9.6.

• From Geant4 Version 10.0: focus only on new
New Design

• New **analysis category** introduced
• Based on g4tools from inlib/exlib developed by G. Barrand (LAL):
  • “Pure header code” - all code is inlined (*a la* Boost)
    Can be installed on iOS, Android, UNIXes, Windows
• Provides code to write **histograms** and **“flat ntuples”**
  in several formats: ROOT, XML AIDA format, CSV
  - pure header: you do not need ROOT/AIDA compliant tool to produce output files (very useful for production, no need to install code on worker nodes)
  - But you need a tool to interact with output file
  - Version 10.2 has prototype of “plotter”: creates images of histograms from G4
g4analysis

It includes a manager (G4AnalysisManager):
• Handles output file(s) creation
• Owns and handles histograms and ntuples

It provides:

• **Uniform interface**
  • Hide the differences according to a selected technology (root, XML, HBOOK, CSV) from the user

• **Higher level management** of g4tools objects (file, histograms, ntuples)
  • Memory management
  • Access to histograms, ntuple columns via indexes

• Integration in the Geant4 framework
  • Interactive commands, Units

• It is thread-safe and provides **automatic summing of histograms**
Histogramming
Histogram creation

B4RunAction.cc

```cpp
#include "B4Analysis.hh"

void B4RunAction::BeginOfRunAction(const G4Run* run)
{
    // Get analysis manager
    G4AnalysisManager* man = G4AnalysisManager::Instance();
    // Open an output file
    man->OpenFile("exampleB4");
    // Create histogram(s)
    man->CreateH1("0", "Edep in absorber", 100, 0., 800*MeV);
    man->CreateH1("1", "Edep in gap", 100, 0., 100*MeV);
}

void B4RunAction::EndOfRunAction(const G4Run* aRun)
{
    G4AnalysisManager* man = G4AnalysisManager::Instance();
    man->Write();
    man->CloseFile();
}
```

B4Analysis.hh

```cpp
#ifndef B4Analysis_h
#define B4Analysis_h 1

#include "g4root.hh"
//#include "g4xml.hh"
//#include "g4hbook.hh"

#endif
```

Selection of the output format at a single place

Histogram IDs are attributed Automatically
More on Histograms

• Histogram identifiers:
  • The histogram ID is automatically generated when a histogram is created by `G4AnalysisManager::CreateH1()`, and its value is returned from this function.
  • **Note:** the histogram names have no relation to the histogram ID which is used in filling.
  • The default start value 0 can be changed with:
    `G4AnalysisManager::SetFirstHistId(G4int)`
    - The 1D and 2D histograms IDs are defined independently

• Histogram objects:
  • It is also possible to access directly the histogram by
    `G4AnalysisManager::GetH1(G4int id) / G4AnalysisManager::GetH1Id(G4String name)`

• **New (10.1): profiles**

```cpp
G4cout << "\n ---> print histograms statistic \n" << G4endl;
G4cout << " EAbs : mean = " << analysisManager->GetH1(1)->mean() << " rms = " << analysisManager->GetH1(1)->rms() << G4endl;
```
Advanced histogram functionalities

• Advanced options are available (via G4AnalysisManager):
  - **Unit**: if a histogram is defined with a unit, all filled values are automatically converted to it
  - **Function**: if a histogram is defined with an associated function, this is used to transform the filling
    - Currently available functions: log, log10, exp.
    - When a histogram is defined with both unit and function the unit is applied first.
  - **User-defined binning scheme** supported (passing vector of bin edges)
    - UI command only for lin/log scheme
  - ASCII option: if activated the histogram is also printed in an ASCII file when G4AnalysisManager::Write() function is called.
  - See /analysis/h1/set UI command
Analysis category UI commands: output file handling

• Handling of files and general options:

```
/analysis/setFileName name  # Set name for the histograms and ntuple file
/analysis/setHistoDirName name  # Set name for the histograms directory
/analysis/setNtupleDirName name  # Set name for the histograms directory
/analysis/setActivation true|false  # Set activation option
/analysis/verbose level  # Set verbose level
```
Histogram Plotting

Since Version 10.2: produce graphics of histograms directly from application

In code:
G4AnalysisManager* analysisManager = G4Analysis::Manager::Instance();
analysisManager->SetH1Plotting( id , true );
//etc for H2, H3, P1, P2

Or using UI commands:
/analysis/h1/setPlotting id true|false
/analysis/h1/setPlottingToAll true/false
#etc for H2, H3, P1, P2

Warning: when ui command /analysis/h1/setPlottingToAll is used all histograms defined up to that moment in application, will be plotted, not added to future histograms!

By default plot style is very simple but works out of the box: from old PAW style
Alternative you can configure Geant4 with FreeType support and choose alternative drawing styles (ROOT, hippodraw)
See: UserDocumentation chapter 9.2.5

Warning: plotting many histograms in current G4 version is very slow and file size is very large. Will be improved in the future
• Basic UI commands for 1D histograms

/analysis/h1/create name title [nbin min max] [unit] [fcn] [binscheme] #Create 1D histo
/analysis/h1/set id nbin min max [unit] [fcn] [fcn] [binscheme] #Change parameters

• 2D commands

/analysis/h2/create
    name title [nxbin xmin xmax xunit xfcn nybin ymin ymax yunit yfcn] # Create 2D histogram
/analysis/h2/set
    id nbin xmin xmax xunit xfcn nybin ymin ymax yunit yfcn # Set parameters

• Change histogram properties via macro
  (examples/extended/electromagnetic/TestEm5/gammaSpectrum.mac)

/analysis/setFileName gammaSpectrum
/analysis/h1/set 3 200 0.01 10 MeV #gamma: energy at vertex
/analysis/h1/set 5 200 0.01 10 MeV log10 #gamma: energy at vertex (log10)
/analysis/h1/set 20 200 0 6 MeV #gamma: energy at exit
/analysis/h1/set 40 200 0 6 MeV #gamma: energy at back
Advanced UI commands

• For 1D histograms:

```
/analysis/h1/setAscii id true|false     # Print 1D histogram on ASCII file
/analysis/h1/setTitle id title          # Set title for the 1D histogram
/analysis/h1/setXaxis id title          # Set x-axis title for the 1D histogram
/analysis/h1/setYaxis id title          # Set y-axis title for the 1D histogram
/analysis/h1/setActivation id true|false # Set activation for the 1D histogram
/analysis/h1/setActivationToAll true|false # Set activation to all 1D histograms
```

• Similar commands for 2D histograms and profiles: “/analysis/h2/....”
MT support

- Multi threading support for histograms:
  - Each thread owns its own copy of a given histograms
  - At the end of the run histograms are “merged” into single one
  - Single file with merged histograms will be produced

![Thread 1](histogram_thread1.png)

![Thread 2](histogram_thread2.png)

![Merge](merge.png)

Output
Ntuples creation

B4RunAction.cc

```cpp
#include "B4Analysis.hh"

void B4RunAction::BeginOfRunAction(const G4Run* run)
{
    // Get analysis manager
    G4AnalysisManager* man = G4AnalysisManager::Instance();

    // Open an output file
    man->OpenFile("exampleB4");

    // Create ntuple
    man->CreateNtuple("B", "Edep and TrackL");
    man->CreateNtupleDColumn("Eabs");
    man->CreateNtupleDColumn("Egap");
    man->FinishNtuple();
}
```

B4EventAction.cc

```cpp
#include "B4Analysis.hh"

void B4EventAction::EndOfEventAction(const G4Run* aRun)
{
    G4AnalysisManager* man = G4AnalysisManager::Instance();
    man->FillNtupleDColumn(0, fEnergyAbs);
    man->FillNtupleDColumn(1, fEnergyGap);
    man->AddNtupleRow();
}
```
More on Ntuples

• Ntuple and Ntuple Column Identifiers
• Automatically generated when the ntuple or ntuple column is created by `G4AnalysisManager::CreateNtuple()` or `G4AnalysisManager::CreateNtupleTColumn()` and its value is returned from this function.
• The default start value 0 can be changed with the `G4AnalysisManager::SetFirstNtupleID(int)` method.
• The ntuple column ID is not specific to the column type
Supported file formats

ROOT:
• Can be view/processed with ROOT: http://root.cern.ch; which is able to process also the HBOOK and CSV formats

XML (AIDA):
• JAS, iAIDA, Open Scientist Lab, rAIDA – see more details in the Appendix 2, in Geant4 Application Developer's Guide And “new” inlib/exlib: http://inexlib.lal.in2p3.fr/

CSV (comma-separated values):
• The simplest possible output format, can be analyzed by many tools Gnuplot, Excel, OpenOffice, ROOT
Additional notes

• Multi-threading note:
  • Each thread owns copy of ntuple
  • **Not** merged at the end of the run (concatenate in analysis!)
  • Use TChain in ROOT
  • as simple as: “cat *_nt_* > ntuple_merged.csv”
• Each thread will write out a separate file:
  fileName[ntupleName]_tD.[xml |root|hbook] (where D = thread Identifier 0,1,2...)
Exercise

Hands On 4 (Thursday Morning):

Exercise 2: Introduce a simple NTuple and few histograms