ANALYSIS

Sebastien Incerti (CNRS, France)

based on material kindly provided by
I. Hrivnacova (CNRS) and A. Dotti (SLAC)

KIRAMS, Seoul – July 9-13, 2018

Contents

- Introduction on analysis

- Histograms

- Ntuples

- UI Commands for analysis

3 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

Status of g4analysis

☐ Analysis category in Geant4 since December 2011

- ☐ Before the analysis code in Geant4 examples used external tools (based on AIDA = Abstract Interfaces for Data Analysis) that had to be linked with Geant4 to produce histograms or ntuples

☐ Area of new developments and improvements: more features are added in each release

- ☐ Example from the latest release: better MPI (Message Passing Interface) support

☐ Based on g4tools from inlib/exlib developed by G. Barrand (LAL, France)

- ☐ http://inexlib.lal.in2p3.fr
- ☐ "Pure header code" - all code is inlined : can be installed on iOS, Android, UNIXes, Windows...

☐ Provides code to write histograms and "flat ntuples" in several formats:
ROOT, XML AIDA format, CSV for ntuples

Status of g4analysis

- It includes a manager (G4AnalysisManager)
  - Handles output file(s) creation
  - Owns and handles histograms and ntuples
- It provides
  - Uniform interface
    - Hides 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

7 | Histograms

Using Geant4 Analysis

- 3 basic steps
    1. - Create G4AnalysisManager

        - Book (create) your histograms, ntuples

        - Open a file
    2. Fill values in histograms, ntuples
    3. Write & close file

Using Geant4 Analysis

- The few basic steps in detail…

  1. Create G4AnalysisManager   in RunAction BeginOfRunAction()

     Open an output file   in RunAction BeginOfRunAction()

     Book (create) your histograms, ntuples   in RunAction BeginOfRunAction()

  2. Fill values in histograms, ntuples   anywhere during event processing,
     e.g. in EventAction EndOfEventAction()

  3. Write & close file   in RunAction EndOfRunAction()

  Note: performing the steps in the suggested classes & methods is not mandatory, but it guarantees correct functioning in multi-threaded mode

Selection of output type

- For simplicity of use, G4AnalysisManager provides a complete access to all interfaced methods for all output formats: ROOT, CSV, AIDA XML
  - though it is implemented via a more complex design
  - the real type is different for each output type:
    G4RootAnalysisManager, G4CsvAnalysisManager, G4XmlAnalysisManager
- The generic types are defined in dedicated header files for each output type:
  - g4root.hh, g4csv.hh, g4xml.hh
  - using **namespaces** and **typedefs**
- It is recommended to add the selected include in an extra header file MyAnalysis.hh and include this header file in all classes which use g4analysis

MyAnalysis.hh

```
#ifndef MyAnalysis_h
#define MyAnalysis_h 1

#include "g4root.hh"
//#include "g4csv.hh"
//#include "g4xml.hh"
#endif
```

1) Step 1: creation

MyRunAction.cc

```cpp
#include "MyAnalysis.hh"

void MyRunAction::BeginOfRunAction(const G4Run* run)
{
  // Create analysis manager
  G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
  analysisManager->SetVerboseLevel(1);

 // Open an output file
   analysisManager->OpenFile("MyApplication");

 // Create histograms
   analysisManager→CreateH1("Edep","Energy deposit", 100, 0., 800*MeV);
   analysisManager→CreateH1("Tlen","Track length", 100, 0., 100*mm);
}
```

2) Step 2: filling

☐ Example of filling 2 one-dimensional histograms

MyEventAction.cc

```cpp
#include "MyAnalysis.hh"

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

  // Fill histograms
  analysisManager→FillH1(0, fEdep);
  analysisManager→FillH1(1, fTrackLength);
}
```

3) Step 3: write & close

☐ Example of writing & closing the output file

MyRunAction.cc

```cpp
#include "MyAnalysis.hh"

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

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

More on histograms

- 1D, 2D, 3D histograms and 1D, 2D profile histograms available

- 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 at filling
  - The default start value 0 can be changed (eg. to 1) with: G4AnalysisManager::SetFirstHistoId(G4int)
    - The 1D, 2D and 3D histograms IDs are defined independently

- Histogram objects

  - It is also possible to access directly a histogram by G4AnalysisManager::GetH1(G4int id)
    The concrete histogram type is hidden behind a selected namespace

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

Histogram options

- 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)
    - When a histogram is defined with both unit and function, then the unit is applied first
  - Binning scheme : users can define a non-equidistant binning scheme (passing a 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 commands

Ntuples

16

1) Step 1: creation

☐ Example of creating an ntuple

MyRunAction.cc

```
void MyRunAction::BeginOfRunAction(const G4Run* run)
{
 // Create analysis manager
  G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
  analysisManager->SetVerboseLevel(1);

 // Open an output file
   analysisManager->OpenFile("MyApplication");

 // Creation of ntuple
  analysisManager->CreateNtuple("MyNtuple", "Edep and TrackLength");
  analysisManager->CreateNtupleDColumn("Eabs");
  analysisManager->CreateNtupleDColumn("Labs");
  analysisManager->FinishNtuple();
}
```

2) Step 2: filling

□ Example of filling an ntuple

MyEventAction.cc

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

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

3) Step 3: write & close

□ Example of writing & closing the output file

MyRunAction.cc

```cpp
#include "MyAnalysis.hh"

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

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

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(G4int) and G4AnalysisManager::SetFirstNtupleColumnId(G4int) methods.

  - In a similar way as for histogram ID

- The ntuple column ID is not specific to the column type: available column types:

  - integer (I), float (F), double (D), string (S)

  - std::vector of integer (I), float (F), double (D) types

21 UI commands

Analysis UI commands (1/3): options and output file handling

- ## General options

```
# Set verbose level
/analysis/verbose level
# Set activation option
/analysis/setActivation true|false
```

- ## Handling output files and general options

```
# Set name for the histograms and ntuple file
/analysis/setFileName name
# Set name for the histograms/ntuple directory
/analysis/setHistoDirName name
/analysis/setNtupleDirName name
```

UI commands (2/3)

☐ Commands to create or define 1D histogram

```
# Create 1D histogram
/analysis/h1/create name title [nbin min max] [unit] [fcn] [binscheme]

# Set histogram parameters
/analysis/h1/set id nbin min max [unit] [fcn] [binscheme]
```

☐ Example of a macro gammaSpectrum.mac in TestEm5 example (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
```

☐ Analogous commands are available for 2D and 3D histograms and 1D and 2D profiles

UI commands (3/3)

☐ For 1D histograms control

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

☐ The same sets of commands are available for 2D and 3D histograms and 1D and 2D profiles, under h2, h3, p1 and p2 directories

More: batch graphics (1/3)

- Since Version 10.2

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

```
// 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

More: output files (2/3)

Depending on selected file format, multiple output files can be produced

- ROOT
    - All histograms, profiles and ntuples are written in one file
- XML (AIDA)
    - The histograms and profiles are written in one file, and each ntuple is written in a separate file
- CSV (comma-separated values)
    - Each histogram, profile and ntuple are written in a separate file

File names are generated automatically
fileName[_objectName].ext where ext = xml, csv

More: plotting style (3/3)

- ☐ 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

- ☐ Set 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

ROOT

- A powerful analysis tool from CERN
  - Installed on your virtual machine
- Start session with root, then open a browser with:

root [0] new TBrowser

File generated
In Geant4
simulation

Selected H1
is automatically
drawn in the
canvas
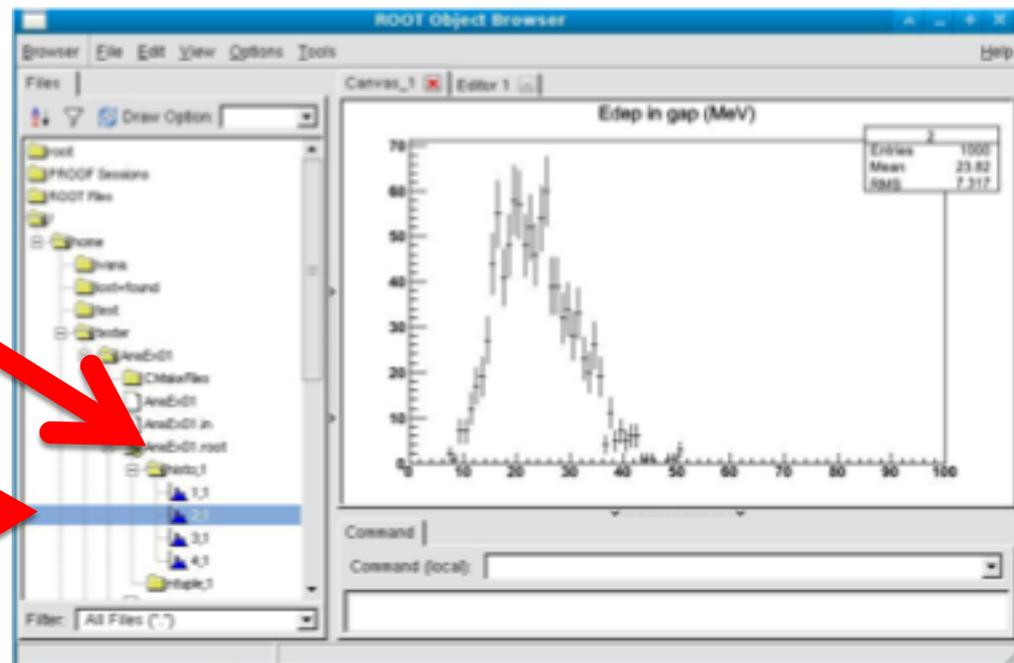
Geant4 examples

□ In examples/extended/analysis, 3 examples to demonstrate how to make histograms and ntuples

  ▫ AnaEx01 – use of Geant4 analysis tools

  ▫ AnaEx02 – use of ROOT classes, requires linking with Root libraries

  ▫ AnaEx03 – use of AIDA interface classes, requires linking with an AIDA compliant tool, eg. OpenScientist

□ http://geant4.web.cern.ch/geant4/UserDocumentation/Doxygen/examples_doc/html/Examples_analysis.html

Summary

- Geant4 provides a lightweight analysis tools as part of distribution

- Can handle histograms (1D, 2D, profiles) and ntuples

- Variety of UI commands

- Variety of output formats

- Compatible with MT

    - Histogram and ntuple merging

- 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