Analysis

L.G. Sarmiento Pico
Lund University, Sweden
(Some slides/inspiration from Maurizio Ungaro)

12th International Geant4 Tutorial in Korea, Pohang 2025.

Starting Point

- The default Geant4 simulation flow demands from you to set up
  - the geometry,
  - the physics list, and
  - the primary generator.
- Then Geant4 transports particles and... that's it, no output!

Bash - exampleB1 execution:

```
$ source <geant4>/install/bin/geant4.sh
$ cp -r <geant4>/examples/basic/B1 .
$ cd B1
$ cmake .
$ make -j 12 -f Makefile
$ ./exampleB1 run1.mac
**************************************************************
Geant4 version Name: geant4-11-03-00 [MT] (16-February-2024)
... output ...
**************************************************************
$ ls
CMakeCache.txt CMakeLists.txt History README exampleB1
exampleB1.in include     run1.mac src CMakeFiles GNUmakefile Makefile cmake_install.cmake exampleB1.cc exampleB1.out
init_vis.mac run2.mac vis.mac
```

No output file, only some info to screen

g4analysis

- Like most MC codes, in Geant4 the user is responsible for scoring AND/OR storing the quantities of interest
- The Geant4 analysis category **g4analysis** (geant4/source/analysis/) provides a unique interface:
  - to write **histograms** and **n-tuples** (the so-called primitive types), and
  - to specify the storing format
    - ROOT
    - HDF5[1]
    - XML
    - CSV
- Good choice to support the large and heterogeneous user community that adopts different tools (Python, ROOT in HEP, ...) to run the analysis *a posteriori*, or '*online*' using ToolsSG (Discussed later in this document)
- Probably a combination of both is a good idea,

  online: to quickly check that you are running the right settings
  offline: to *proper* analyse the data

[1] Geant4 must have been installed using -DGEANT4_USE_HDF5=ON

g4analysis

- **g4analysis** is available in `Geant4` since December 2011
  - It is an active area of development with **new features added at every release**
  - It ensures input/output (I/O) thread-safe capability in multi-threaded simulations (*take a second to appreciate how cool is that...*)
- How to use it? Via an interface
  - Users access the **g4analysis** tools via the `G4AnalysisManager` class

Users can still link their applications against external libraries to handle I/O
typically `ROOT::TH*`/`ROOT::TTree` for small applications and Event Data Models for large applications.
Be careful if running in Multi-thread mode

G4AnalysisManager

- Basic features:
  - `G4AnalysisManager` is a **singleton**
    - users must access it with static method `::Instance()`
  - handles output file(s) (re)creation
  - owns and handles histograms and n-tuples
- It provides:
  - uniform user interface to the different formats (`ROOT`, `HDF5`, `XML`, `CSV`), i.e. the user is not required to know how to handle any of them,
  - with high-level memory management and access to low-level objects (e.g. n-tuples columns of variable size)
- Fully integrated in the `Geant4` framework
  - it is accessible via user commands and
  - uses `Geant4` units

Using G4AnalysisManager (1/3) : Preparation/Set up

Create the manager, book histos/n-tuples and open a file, to be done at the *begin of a run* (::BeginOfRunAction() in your application). **The format is picked by the extension of the file**

RunAction.cc (excerpt)

```
#include "G4AnalysisManager.hh"
void RunAction::BeginOfRunAction(const G4Run* run) {
  auto analysisManager = G4AnalysisManager::Instance(); //get the singleton
  std::string runnumber = std::to_string( run->GetRunID() );  // fileName can be a fixed G4String
  G4String fileName = "Run" + runnumber + ".root";      //select root format
  //G4String fileName = "Run" + runnumber + ".xml"; //or select  xml format
  //G4String fileName = "Run" + runnumber + ".csv"; //or select  csv format
  //G4String fileName = "Run" + runnumber + ".hdf5";//or select hdf5 format -> requires hdf5 installation
  analysisManager->OpenFile(fileName);

  // When you declare more than one, remember their order
  //Create histogram(s)              Name   Title                      nbins   xmin    xmax      unit
  analysisManager->CreateH1("Edep","Energy deposit",100,0.*MeV,10*GeV, "MeV");

  // When you declare more than one, remember their order
  // Create ntuple(s)
  analysisManager->CreateNtuple("Ntuple", "Ntuple name");
  analysisManager->CreateNtupleDColumn("Energy");  // Data name
  analysisManager->FinishNtuple();
  }
```

Using G4AnalysisManager (2/3) : Fill values

Fill values in histos and n-tuples, likely at the end of each event (`EventAction::EndOfEventAction()`). They are identified **in creation order starting with 0**.

EventAction.cc (excerpt)

```
#include "G4AnalysisManager.hh"
void EventAction::EndOfEventAction(const G4Event*){
  auto analysisManager = G4AnalysisManager::Instance();  // Get singleton

  //Fill histogram(s)
  analysisManager->FillH1(0, Edep); // Fill value as MeV because you declared it

  //Fill ntuple(s)
  analysisManager->FillNtupleDColumn(0, Edep / MeV);   // Store/Dump value as MeV (You are responsible for the conve

  analysisManager->AddNtupleRow();
}
```

Using G4AnalysisManager (2/3) : Closing the File

Write and **close** file, at the end of each run (`RunAction::EndOfRunAction()`)[2]

RunAction.cc (excerpt)

```
#include "G4AnalysisManager.hh"
void RunAction::EndOfRunAction(const G4Run*) {
  auto analysisManager = G4AnalysisManager::Instance();
  // Write and close file
  analysisManager->Write();
  analysisManager->CloseFile();
}
```

[2] The user is free to open/fill/close the files ANYWHERE in their application but this is probably where you want to do it

Output: ROOT

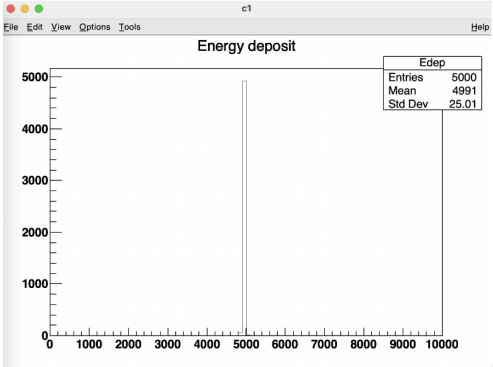
RunAction.cc

```
G4String fileName = "Run" + runnumber + ".root";
```

Bash - after execution

```
$ root -l Run0.root
root [0]
Attaching file Run0.root as _file0...
(TFile *) 0x1258cc990
root [1] .ls
TFile**         Run0.root
 TFile*         Run0.root
  KEY: TTree    Ntuple;1  Ntuple
  KEY: TH1D     Edep;1    Energy deposit
root [2] Edep->Draw("histo")
root [3] Ntuple->Scan()

************************
*    Row   * Energy.En *
************************
*        0 * 4986.2465
*        1 * 4988.8028
*        2 *      5000
*        3 * 4996.4155
```



Output: AIDA XML & CSV

RunAction.cc

```
G4String fileName = "Run" + runnumber + ".xml";
G4String fileName = "Run" + runnumber + ".csv";
```

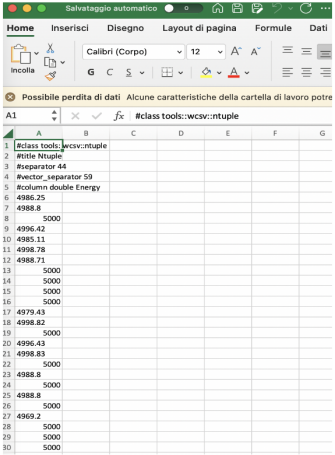
Bash - after execution

```
$ open Run0_nt_Ntuple.csv
$ open Run0.xml
```

Run0.xml — Edited

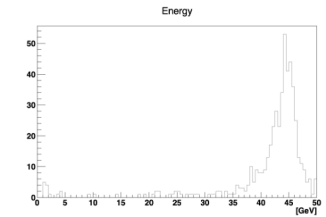
Run0 › histogram1d "Energy deposit"

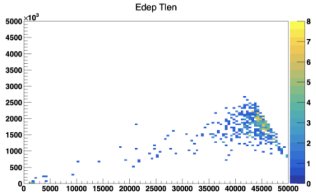
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE aida SYSTEM "http://aida.freehep.org/schemas/3.2.1/aida.dtd">
<aida version="3.2.1">
  <implementation package="tools" version="5.6.0"/>
  <histogram1d path="/" name="Edep" title="Energy deposit">
    <annotation>
      <item key="axis_x.title" value=""/>
    </annotation>
    <axis direction="x" numberOfBins="100" min="0" max="10000"/>
    <statistics entries="5000">
      <statistic direction="x" mean="4991.163651377722089819144" rms="25.006461587641538812931697"/>
    </statistics>
    <data1d>
      <bin1d binNum="44" entries="1" height="1" error="1" weightedMean="4451.5204783110584685346 11"/>
      <bin1d binNum="45" entries="3" height="3" error="1.7320508075688771931766044"
          weightedMean="4576.961659167744073783979" weightedRms="22.328554700306483482562364"/>
      <bin1d binNum="46" entries="2" height="2" error="1.4142135623730951454746222"
          weightedMean="4621.2469482664855604652566" weightedRms="15.712966417564876664414446"/>
      <bin1d binNum="47" entries="6" height="6" error="2.4494897427831778813356324"
          weightedMean="4756.0596685582031568628744" weightedRms="25.498484724997929617984487"/>
      <bin1d binNum="48" entries="55" height="55" error="7.4161984870956629833926855"
          weightedMean="4866.7247667115707372431644" weightedRms="28.309935867661234709657957"/>
      <bin1d binNum="49" entries="4921" height="4921" error="70.149839629182338285318111"
          weightedMean="4993.3267016718873349220924" weightedRms="12.664306018189466577452607"/>
      <bin1d binNum="50" entries="12" height="12" error="3.4641016151377543863532088"
          weightedMean="5002.2032023686423262669239" weightedRms="0.57316533276943542931703575"/>
    </data1d>
</aida>
```

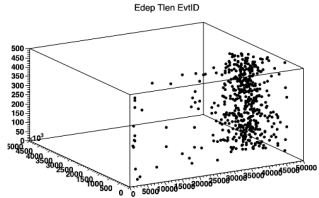


Output: Histogram

- It is possible to create
  - 1D (`CreateH1()`),
  - 2D (`CreateH2()`),
  - 3D (`CreateH3()`) histograms as well as
  - 1D (`CreateP1()`) and
  - 2D (`CreateP2()`) profile histograms
- **IDENTIFIERS**: each histo ID is automatically generated when the histo is created and its value returned by the creating function (*histo names are not related to IDs and cannot be used for filling*)
  - **Default start value is 0**, it can be changed with `G4AnalysisManager::SetFirstHistoId(G4int)`
  - IDs for histograms `H*`, profiles `P*` and n-tuples run independently







Output: Histogram

- It is possible to access histogram objects (and use their properties/functions) via the `G4AnalysisManager`

RunAction.cc - accessing histogram(s) mean and rms

```
void RunAction::EndOfRunAction(const G4Run* /*run*/) {

  auto analysisManager = G4AnalysisManager::Instance();

  G4cout << G4endl << " ----> print histograms statistic ";
  if(isMaster) {
    G4cout << "for the entire run " << G4endl << G4endl;
  } else {
    G4cout << "for the local thread " << G4endl << G4endl;
  }

  G4cout << " Edep : mean = "
         << G4BestUnit(analysisManager->GetH1(0)->mean(), "Energy")
         << " rms = "
         << G4BestUnit(analysisManager->GetH1(0)->rms(),  "Energy")
         << G4endl;

}
```

Terminal - Geant4 execution on 2 threads

```
************************************************************
 Geant4 version Name: geant4-11-01 [MT]   (9-December-2022)
  << in Multi-threaded mode >>
     Copyright  : Geant4 Collaboration
     References : NIM A 506 (2003), 250-303
                : IEEE-TNS 53 (2006), 270-278
                : NIM A 835 (2016), 186-225
                  WWW : http://geant4.org/
************************************************************



// some output


G4WT0 >  ----> print histograms statistic for the local thread
G4WT0 >  Edep : mean = 41.117 MeV rms = 9.2968 MeV
G4WT1 >  ----> print histograms statistic for the local thread
G4WT1 >  Edep : mean = 41.906 MeV rms = 7.36116 MeV
----> print histograms statistic for the entire run
Edep : mean = 41.5036 MeV rms = 8.41347 MeV
```

Histograms - More Details (link)

Unit if defined, filled values are automatically converted to it

```
analysisManager->CreateH1("Tlen","Tracks length",100,0.*km,5.*km,"km")
```

Function if defined, the function is automatically executed on the filled value

- `log` ($\log_e$),
- `log10`,
- `exp`

(unit value conversion precedes the function)

Binning scheme default is linear, possible to set it to logarithm

- `lin`
- `log` ($\log_{10}$)

```
AnalysisManager->CreateH1("Tlen","Tracks length",100,0.1*km,5.*km, "none", "none", "log")
```

User-defined binning scheme alternative constructor available to set an arbitrary binning scheme using a vector of bin edges

```
G4int CreateH1(const G4String& name, const G4String& title,
               const std::vector<G4double>& edges,          // any distribution is fine
               const G4String& unitName = "none",
               const G4String& fcnName = "none");
```

Output - Ntuples

- **IDENTIFIERS**: similar to histograms, ntuple and ntuple column IDs are **automatically generated** and returned by `G4AnalysisManager->CreateNtuple()` and `G4AnalysisManager->CreateNtupleXColumn()`
- IDs **must be used to fill** columns if more than one ntuple is created (i.e. "0" is implied)

```
void EventAction::EndOfEventAction(const G4Event* event){
  auto analysisManager = G4AnalysisManager::Instance();
  //Fill ntuple 0 and 1
  analysisManager->FillNtupleDColumn(0, 0, Edep); //ntuple ID, column ID, value
  analysisManager->FillNtupleDColumn(0, 1, TrackL);
  analysisManager->AddNtupleRow(0);
  analysisManager->FillNtupleDColumn(1, 0, Edep/2.);
  analysisManager->FillNtupleDColumn(1, 1, TrackL/2.);
  analysisManager->AddNtupleRow(1);
}
```

- The following types can be saved in columns:
  - `int` (`I - IColumn`),
  - `float` (`F - ...`),
  - `double` (`D`),
  - `string` (`S`),
  - `std::vector` of `int` (`I`), `float` (`F`), or `double` (`D`).

User Interface to g4analysis

Several UI commands are available to interact with the `G4AnalysisManager` in your macro file

- General options and handling histos (similar commands available for H2, H3, P1 and P2)

```
/analysis/verbose 1                # Set verbose level
/analysis/h1/set 0 100 0. 50. GeV # set H1 nbins, min, max
```

- Handling output file

```
/analysis/setFileName name          # set output file name
/analysis/setHistoDirName histos    # store hist in directory
/analysis/setNtupleDirName ntuples # store ntuple in directory
```

- Example of a macro gammaSpectrum.mac in TestEm5 example
  - examples/extended/electromagnetic/TestEm5/gammaSpectrum.mac

User Interface to g4analysis

Several UI commands are available to interact with the `G4AnalysisManager` in your macro file

- Histograms control (similar commands available for H2, H3, P1 and P2)

```
/analysis/h1/setAscii id true|false # activate printing h1 on ASCII file
/analysis/h1/setTitle id title # Set title for the 1D histogram
/analysis/h1/setXaxis id title # Set x-axis, y-axis title for the 1D histogram
/analysis/h1/setYaxis id title
```

- Batch graphics (similar commands available for H2, H3, P1 and P2)

```
/analysis/h1/setPlotting id true|false          # Automatic plotting of h1
/analysis/h1/setPlottingToAll true|false
```

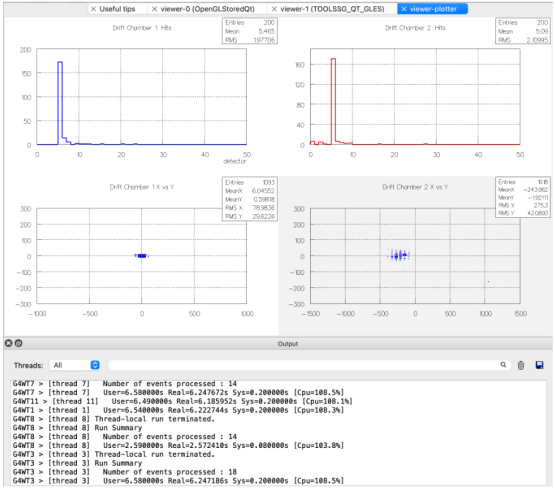
also available as plain C++: `analysisManager->SetH1Plotting(id, true);`

- The selected objects will be automatically plotted in a single (`.ps`) file with page size fixed to A4 format

ToolsSG Driver - **“online”** check

- Binds to various renderer: `OpenGL`, `X11`, `Xt`, `Windows`, `Qt5`, `Qt6`
- Allows some plotting
- Used in `G4/analysis` for batch plotting
- See `plotter.mac` in example `B5` (`examples/basic/B5`)

```
/vis/open TSGQt
/control/execute plotter.mac
/run/beamOn 200
```



Summary

The Geant4 toolkit provides a *lightweight* analysis tool

- It handles -for you- the histograms and ntuples creation and storage in ROOT, (AIDA) XML, CSV and HDF5 format
- It is fully integrated in the framework (units, UI commands)
- It is specifically designed to deliver thread-safe solutions in multi-threaded simulations

All basic and extended examples, as well as several advanced examples, use the G4AnalysisManager

- Examples that address data persistence in Geant4 (examples/extended/analysis)
  - AnaEx01 demonstrates how to use **g4analysis** tools for ntuples and histograms creation
  - AnaEx02 achieves same results by linking ROOT to a Geant4 application
  - AnaEx03 use of AIDA interface classes, requires linking with an AIDA compliant tool, eg. OpenScientist

Much more information in the Geant4 Analysis Documentation (link)