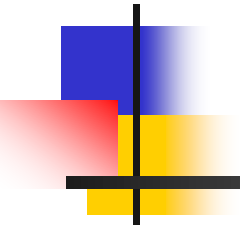
Interaction with the Geant4 kernel – part 2

Luciano Pandola

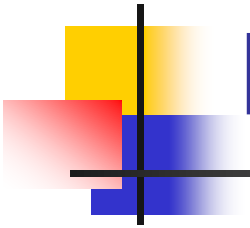
INFN – Laboratori Nazionali del Sud

Partially based on a presentation by G.A.P. Cirrone (INFN-LNS)

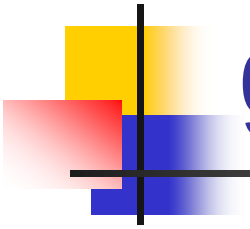


G4analysis tools

Native Geant4 analysis classes



- A basic analysis interface is available in Geant4 for histograms (1D and 2D) and ntuples
  - Make life easier because they are MT-compliant (no need to worry about the interference of threads)
- Unique interface to support different output formats
  - ROOT, AIDA XML, CSV and HBOOK
  - Code is the same, just change one line to switch from one to an other
- Everything done via the public analysis interface **`G4AnalysisManager`**
  - Singleton class: Instance()
  - **UI commands** available for creating histograms at run-time and setting their properties



g4analysis

- Selection of output format is hidden in a user-defined .hh file
- **All** the rest of the code **unchanged**
  - Unique interface

```
#ifndef MyAnalysis_h
#define MyAnalysis_h 1

#include "g4root.hh"
//#include "g4xml.hh"
//#include "g4csv.hh"  // can be used only with ntuples

#endif
```

Open file and book histograms

```
#include "MyAnalysis.hh"

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

  // Creating histograms
  man->CreateH1("h","Title", 100, 0., 800*MeV);
  man->CreateH1("hh","Title",100,0.,10*MeV);

 // Open an output file
  man->OpenFile("myoutput");
}
```

Start numbering of histograms from ID=1

ID=1

ID=2

Open output file

Fill histograms and close

```
#include "MyAnalysis.hh"
void MyEventAction::EndOfEventAction(const G4Run* aRun)
{
  G4AnalysisManager* man = G4AnalysisManager::Instance();
  man->FillH1(1, fEnergyAbs);
  man->FillH1(2, fEnergyGap);
}
void MyRunAction::EndOfRunAction(const G4Run* aRun)
{
  G4AnalysisManager* man = G4AnalysisManager::Instance();
  man->Write();
  man->CloseFile();
}
MyRunAction::~MyRunAction()
{
  delete G4AnalysisManager::Instance();
}
```

ID=1

ID=2

Histograms - 1

- Support linear and log scales and un-even bins
- **`CreateH2()`** for 2D histograms

```
G4int CreateH1(const G4String& name, const G4String& title,
               G4int nbins, G4double xmin, G4double xmax,
               const G4String& unitName = "none",
               const G4String& fcnName = "none",
               const G4String& binSchemeName = "linear");

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

Histograms - 2

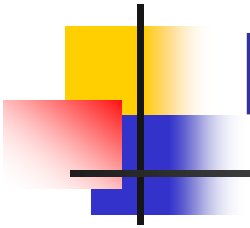
- Can change parameters of an existing histogram
- Can fill with a weight
- Methods to scale, retrieve, get rms and mean

```
G4bool SetH1Title(G4int id, const G4String& title);
G4bool SetH1XAxisTitle(G4int id, const G4String& title);
G4bool SetH1YAxisTitle(G4int id, const G4String& title);

G4bool FillH1(G4int id, G4double value, G4double weight =
1.0);

G4bool ScaleH1(G4int id, G4double factor);

G4int GetH1Id(const G4String& name, G4bool warn = true) const;
```

Histograms - 3

- UI support available, to change parameters (e.g. file name) at run-time

```
/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

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



Ntuples

- g4tool supports ntuples
  - Any number of ntuples, each with any number of columns
  - The content can be **int / float / double**
- For more complex tasks (e.g. full functionality of ROOT TTrees) have to link ROOT directly
- Similar strategy as for histograms. Access happens through the common interface **`G4AnalysisManager`**
  - Saved on the same output file with histograms

Book ntuples

```
#include "MyAnalysis.hh"
void MyRunAction::BeginOfRunAction(const G4Run* run)
{
  // Get analysis manager
  G4AnalysisManager* man = G4AnalysisManager::Instance();
  man-> SetFirstNtupleId(1);

  // Creating ntuple
  man->CreateNtuple("name", "Title");
  man->CreateNtupleDColumn("Eabs");
  man->CreateNtupleDColumn("Egap");
  man->FinishNtuple();

  man->CreateNtuple("name2","title2");
  man->CreateNtupleIColumn("ID");
  man->FinishNtuple();
}
```

Start numbering of ntuples from ID=1

ID=1

ID=2

Fill ntuples

- File handling and general clean-up as shown for histograms

```
#include "MyAnalysis.hh"
void MyEventAction::EndOfEventAction(const G4Run* aRun)
{
  G4AnalysisManager* man = G4AnalysisManager::Instance();
  man->FillNtupleDColumn(1, 0, fEnergyAbs);
  man->FillNtupleDColumn(1, 1, fEnergyGap);
  man->AddNtupleRow(1);

  man->FillNtupleIColumn(2, 0, fID);
  man->AddNtupleRow(2);

}
```

ID=1, columns 0, 1

ID=2, column 0