Framework: hands-on session

A. Sarti

Dipartimento di Scienze di Base e Applicate per l'Ingegneria

Sapienza Università di Roma

First exercise

- Let's practice how to access the information within shoe.
  - We're going to use the files in 'Full' folder. Both Tutorial/Full/12C_C_200shoe.root and Tutorial/Full/12C_C_200.root
  - Both files contain the very same information. The first one, in the 'shoe' format, the second one is a 'simple' tree with structures.
  - You'll need DecodeMC to 'process' the files.
    - **ATTENTION** When you run on '*shoe*' files remember that the FootGlobal.par file to be used (the one in config/12C_200, used when you specify the -exp 12C_200 flag) need to have the **EnableRootObject flag set to: y**
  - From the bin/Reconstruction/level0 folder launch: ../../bin/DecodeMC -in #pathto/12C_C_200shoe.root -out #yourfavouriteteststring.root -nev 1500 -exp 12C_200 -run 1 [to process 1500 events]
- Once you're done, open the #yourfavouriteteststring.root and play a little bit with the histograms and the ntuple.. try to plot histograms and branches…!

Second exercise

- Now that we know how to 'decode' the MC we can do something similar with data!
  - We're going to use the files in 'Data' folder. Both data_built.2211.physics_foot.daq.WD.1.dat and data_built.2211.physics_foot.daq.VTX.1.dat. The files contain the information from the 'timing' detectors (SC, TW) and the VTX detector respectively (we could not manage to 'sync' offline the data… this is a well known problem).
  - You'll need DecodeRaw to 'process' the files.
  - From the bin/Reconstruction/level0 folder launch: ../../bin/DecodeRaw -in #pathtodata/data_built.2211.physics_foot.daq.WD.1.dat -out #lookmom!it'sdata.root -nev 1000 -exp GSI -run 2211 [to process 1000 events]
- Once you're done, open the #lookmom!it'sdata.root and play a little bit with the histograms and the ntuple.. try to understand the main differences btw the info available for data and MC!

Third exercise

- Let's concentrate to one of the few detectors available both in data and MC… Eg: the TW! We will be playing with: TATWactNtuRaw
  - Add an histogram in the TW folder that will show the time differences at the two ends of a bar
  - 2D plots: plot the position along the bar as a function of the PositionId (crossing btw horizontal and vertical bars)
  - Add to the hit information the 'PositionId' and position along the bar information, so that it is stored into an ntuple, and produce, with root interactively a plot to be compared with the previous one
  - Add a cout and control its output using FootDebugLevel(XXX)
- The former exercises have to be done both on data and MC!!! Check the output and the differences!

3rd exercise hints

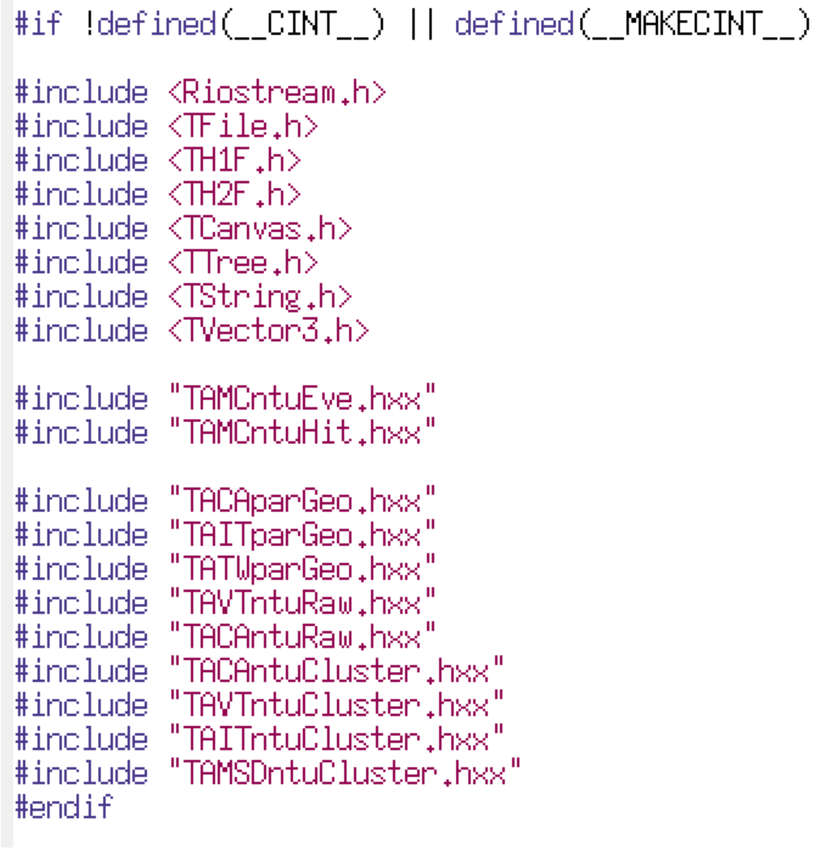
- To add an histogram:
  - remember that first of all you need to book it inside bookhistograms() method of the action
  - Inside Action() you need to
    - check the existence with ValidHistogram()
    - Fill it
- To add an information to a given branch
  - Go to the dataDsc class that implements the data object stored inside the tree
  - modify the class adding what you need (.hxx)
  - add the setters and the getters
  - initialise to meaningful values the info when you create the object pointer
  - fill the info inside the Action using the setters

Fourth exercise

- Now that you have both the output of MC and data processing, we can try to use a ROOT macro to plot something using directly the output of Decode*
  - Remember to set to ‘y’ the EnableTree flag inside FootGlobal.par!
- You need to prepare a macro that:
  - Configure the shoe objects that you want to read
  - Configure the geometry, calibration, configurations of the detector you want to explore
  - Open the output file and book the histogram you want to produce
  - Open the input files, start a loop
  - Get access the pointer of the dataDsc object you want to study
  - Use the shoe methods to fill the histograms.

An example

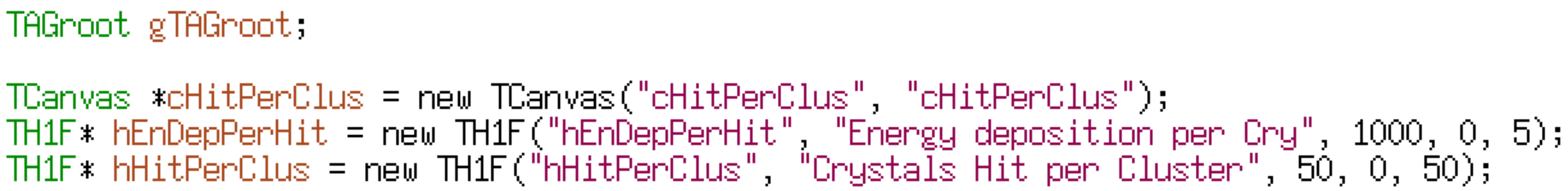
- If you want you can use SC or TW that are present both in data & MC or the Calo that is available only in MC
- 1st: add the needed includes.. (not all of them are shown! e.g. campaign manager…)



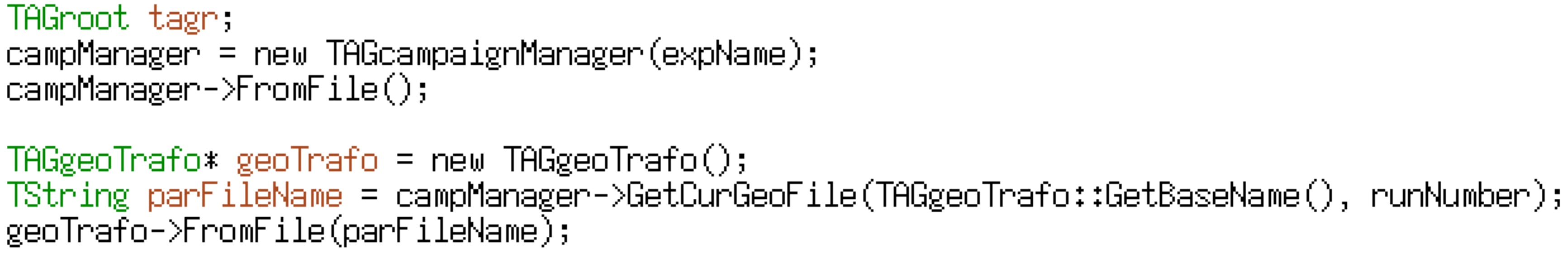
```
#if !defined(__CINT__) || defined(__MAKECINT__)

#include <Riostream.h>
#include <TFile.h>
#include <TH1F.h>
#include <TH2F.h>
#include <TCanvas.h>
#include <TTree.h>
#include <TString.h>
#include <TVector3.h>

#include "TAMCntuEve.hxx"
#include "TAMCntuHit.hxx"

#include "TACAparGeo.hxx"
#include "TAITparGeo.hxx"
#include "TATWparGeo.hxx"
#include "TAVTntuRaw.hxx"
#include "TACAntuRaw.hxx"
#include "TACAntuCluster.hxx"
#include "TAVTntuCluster.hxx"
#include "TAITntuCluster.hxx"
#include "TAMSDntuCluster.hxx"
#endif
```

An example

- If you want you can use SC or TW that are present both in data & MC or the Calo that is available only in MC
- 1st: add the needed includes (not all of them are shown! e.g. campaign manager…)
- 2nd: create taproot object to handle the root file, create the histograms for the output, create the objects and pre-configure the geometry…



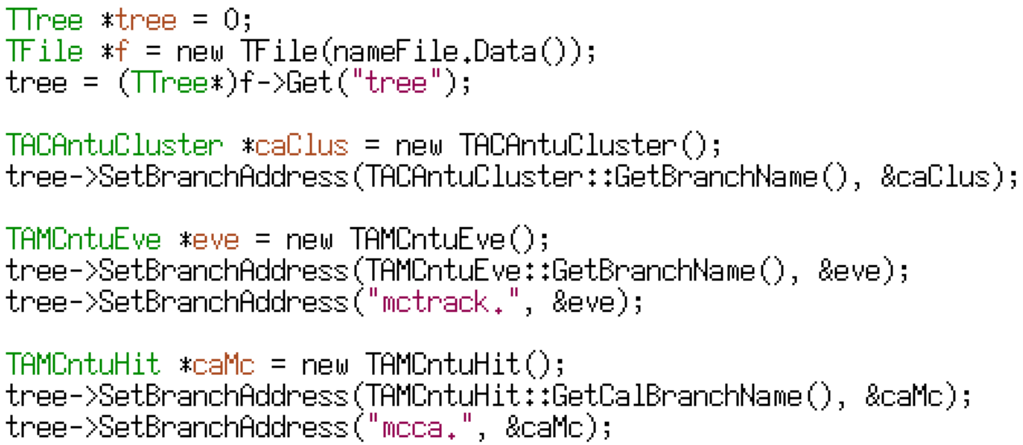
```
TAGroot gTAGroot;

TCanvas *cHitPerClus = new TCanvas("cHitPerClus", "cHitPerClus");
TH1F* hEnDepPerHit = new TH1F("hEnDepPerHit", "Energy deposition per Cry", 1000, 0, 5);
TH1F* hHitPerClus = new TH1F("hHitPerClus", "Crystals Hit per Cluster", 50, 0, 50);
```



```
TAGroot tagr;
campManager = new TAGcampaignManager(expName);
campManager->FromFile();

TAGgeoTrafo* geoTrafo = new TAGgeoTrafo();
TString parFileName = campManager->GetCurGeoFile(TAGgeoTrafo::GetBaseName(), runNumber);
geoTrafo->FromFile(parFileName);
```

An example

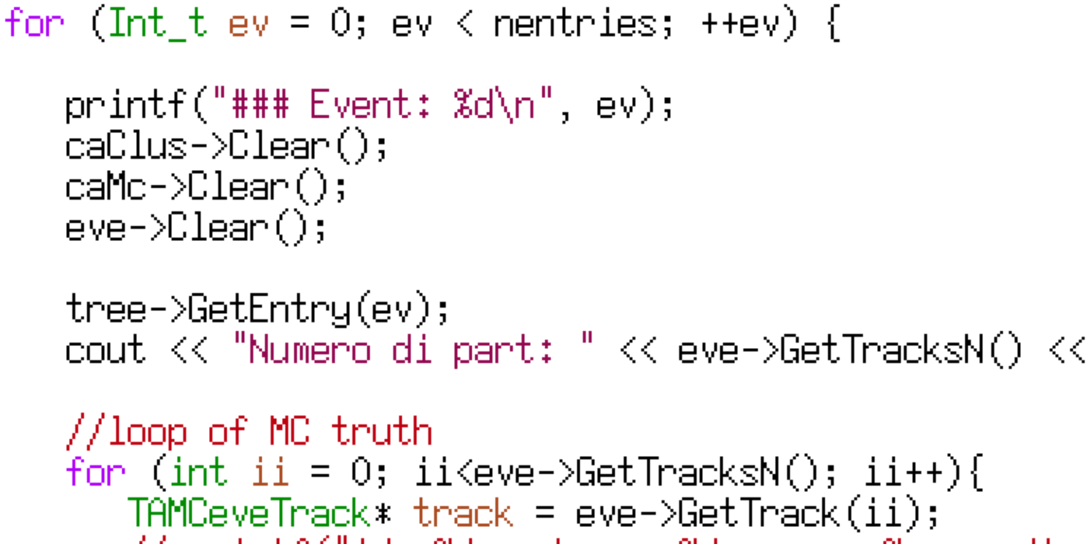
- ➡ If you want you can use SC or TW that are present both in data & MC or the Calo that is available only in MC
- ➡ 1st: add the needed includes (not all of them are shown! e.g. campaign manager…)
- ➡ 2nd: create taproot object to handle the root file, create the histograms for the output, create the objects and pre-configure the geometry…
- ➡ 3rd: Open the file and attach the branches to the shoe objects



```
TTree *tree = 0;
TFile *f = new TFile(nameFile.Data());
tree = (TTree*)f->Get("tree");

TACAntuCluster *caClus = new TACAntuCluster();
tree->SetBranchAddress(TACAntuCluster::GetBranchName(), &caClus);

TAMCntuEve *eve = new TAMCntuEve();
tree->SetBranchAddress(TAMCntuEve::GetBranchName(), &eve);
tree->SetBranchAddress("mctrack.", &eve);

TAMCntuHit *caMc = new TAMCntuHit();
tree->SetBranchAddress(TAMCntuHit::GetCalBranchName(), &caMc);
tree->SetBranchAddress("mcca.", &caMc);
```

An example

- ➡ If you want you can use SC or TW that are present both in data & MC or the Calo that is available only in MC
- ➡ 1st: add the needed includes (not all of them are shown! e.g. campaign manager…)
- ➡ 2nd: create taproot object to handle the root file, create the histograms for the output, create the objects and pre-configure the geometry…
- ➡ 3rd: Open the file and attach the branches to the shoe objects
- ➡ 4th: have fun!



```
for (Int_t ev = 0; ev < nentries; ++ev) {

   printf("### Event: %d\n", ev);
   caClus->Clear();
   caMc->Clear();
   eve->Clear();

   tree->GetEntry(ev);
   cout << "Numero di part: " << eve->GetTracksN() <<

   //loop of MC truth
   for (int ii = 0; ii<eve->GetTracksN(); ii++){
      TAMCeveTrack* track = eve->GetTrack(ii);
```

Last exercise :)

- Issue a git pull command :)
  - Last night we managed to fix a couple of things here and there.. so now the situation is the following:
  - You have changed shoe wrt the Master branch, but you want to update the branch to get the latest changes! how can you do that?
- Updating with git when you have done few changes:
  - from inside shoe:
    - issue a git pull command. The command should fail if you have changes that have not yet been committed.
    - to proceed you need to 'stash' your changes [https://git-scm.com/docs/git-stash]: git stash
    - then proceed with the pull: git pull (this time should work)
    - to reapply your changes you can use: git stash pop
  - Then you recompile and proceed happily to redo one of the previous exercises