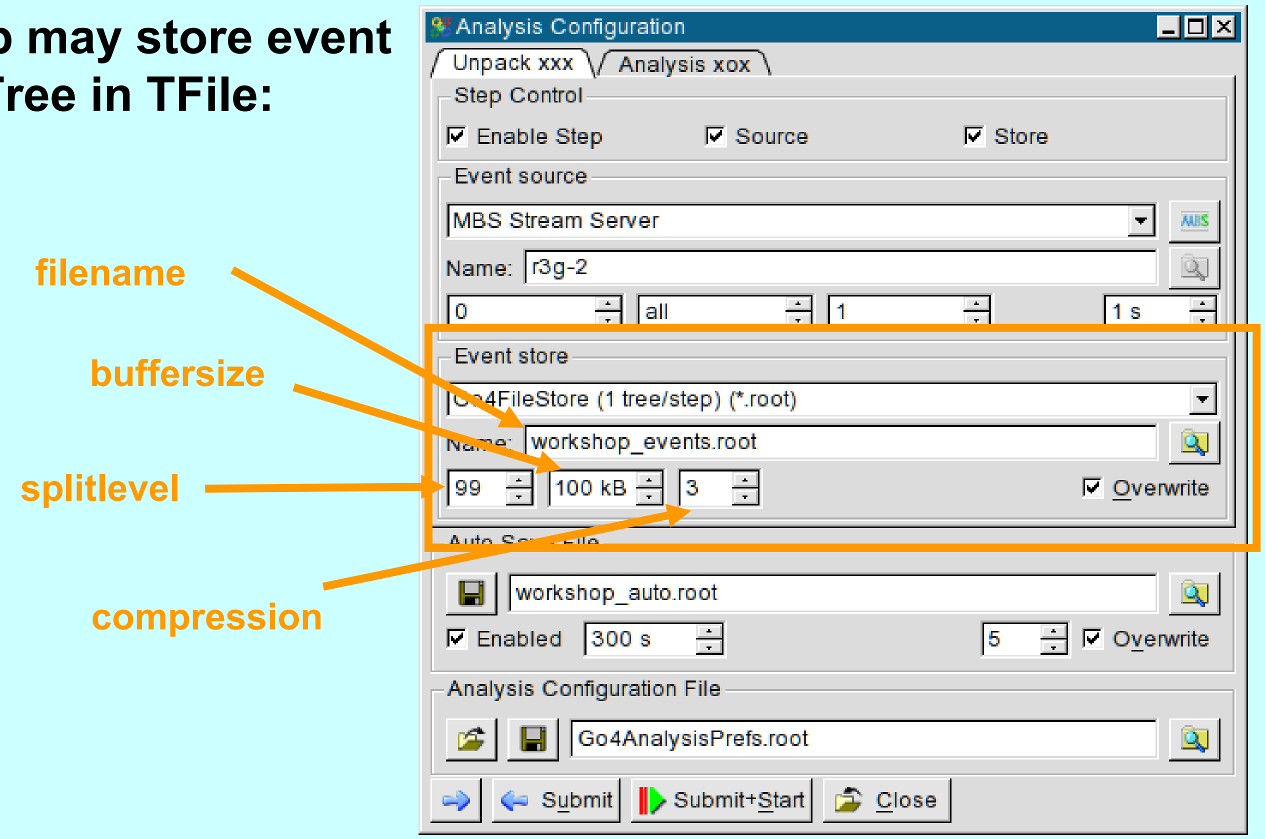
Go4 Trees in CINT

J.Adamczewski-Musch, H.G.Essel, S.Linev

Go4 Workshop 2010

Go4 event store as ROOT Tree

- Each step may store event as ROOT Tree in TFile:



Analysis Configuration

Unpack xxx | Analysis xox

Step Control: ☑ Enable Step ☑ Source ☑ Store

Event source: MBS Stream Server
Name: r3g-2
0 | all | 1 | 1 s

Event store: Go4FileStore (1 tree/step) (*.root)
Name: workshop_events.root
99 | 100 kB | 3 | ☑ Overwrite

Auto Save File: workshop_auto.root
☑ Enabled | 300 s | 5 | ☑ Overwrite

Analysis Configuration File: Go4AnalysisPrefs.root

Submit | Submit+Start | Close

- Tree file can be input for subsequent step in Go4
- Tree file can be inspected from ROOT TBrowser or Go4 browser
- Tree file can be analyzed by CINT macro

Reading a TTree explicitely

```
TFile hfile(“workshop_events.root");
TTree* tr= dynamic_cast<TTree*>(hfile.Get(“UnpackXTree“));
if(tr==0){
             cerr << “error: did not find tree!”;
             return 1; // or may throw exception here...
      }
TObject* h1= new TH1F(“crate1-0“,“Crate 1 Ch 0“,2048,0,2047);
TXXXUnpackEvent* eve= new TXXXUnpackEvent;
tr->SetBranchAddress(“UnpackEvent”,&eve); // by branchname!
Int_t all=tr->GetEntries(); // number of events
for(Int_t i=0; i<all; ++i){
      tr->GetEntry(i); // read event #i into memory
      h1->Fill(eve->fiCrate1[0]);
      // do analysis on members of event class here!
      //...
}
```

Reading a TTree explicitely (cont.)

- **Event object at SetBranchAddress must match the structure used on writing the tree**

NOTE: Go4 classes are known in CINT by automatically generated `libGo4UserAnalysis.rootmap` in user directory

- **TTree::GetEntry will read event data from active branches into local event object**

- **TTree::MakeClass() generates sourcecode template for event reading from given TTree**

Eventclass needs not to be known here!

- **Explicit reading of events is not necessary for simple analysis, use TTree::Draw() feature (GUI: treeviewer)!**

TTree::Draw() examples

```
TTree* tr= .. // got from file
tr->Draw("fValue","fValue>100 && fValue<500");
        // fill default histogram htemp with fValue if
        // condition is true; draw htemp
tr->Draw("fX:fY >> hpxpy","","lego");
        // fill existing 2d histogram of name "hpxpy"
        // and display as "lego" plot
tr->Draw("fMatrix[][]/fValue >>+hmatrix","");
        // continue filling histogram hmatrix
        // with sum of all elements of matrix by fValue
tr->Draw(">>myeventlist","sqrt(fValue)>fMatrix[0][2]");
        // mark all events in tree that fulfill
        // the condition into TEventList "myeventlist"
```

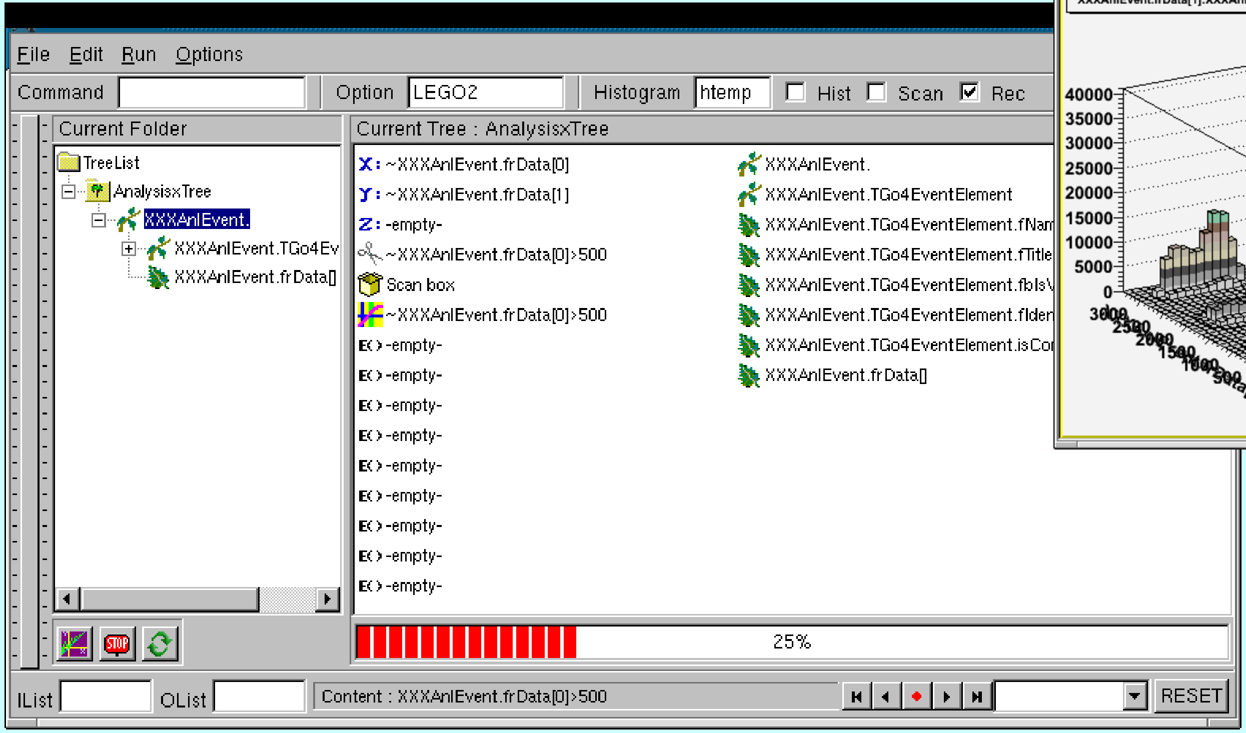
TTree::Draw() (cont.)

`TTree::Draw(`*`expression`*`,`*`selection`*`,`*`option`*`)`

- **May fill histogram/graph from expression, or will mark matching events in a TEventList**
- **Expression may contain any combination of known branch names**
- **Expression may specify output histogram name and dimensions, or output eventlist**
- **Selection gives condition between branch values of one event; this must be true to execute expression**
- **Option may contain draw option for result histogram**
- **SEE ROOT DOC for complete list of features!**

ROOT treeviewer

TTree::Draw by click / drag and drop of tree leaves



File Edit Run Options

Command | Option LEGO2 | Histogram htemp | ☐ Hist ☐ Scan ☑ Rec

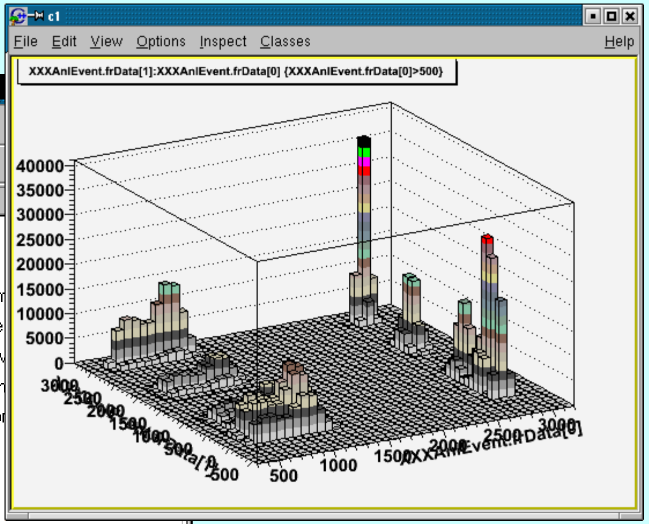
Current Folder
- TreeList
  - AnalysisxTree
    - XXXAnlEvent.
      - XXXAnlEvent.TGo4Ev
      - XXXAnlEvent.frData[]

Current Tree : AnalysisxTree

```
X: ~XXXAnlEvent.frData[0]         XXXAnlEvent.
Y: ~XXXAnlEvent.frData[1]         XXXAnlEvent.TGo4EventElement
Z: -empty-                        XXXAnlEvent.TGo4EventElement.fNam
~XXXAnlEvent.frData[0]>500        XXXAnlEvent.TGo4EventElement.fTitle
Scan box                          XXXAnlEvent.TGo4EventElement.fbIsV
~XXXAnlEvent.frData[0]>500        XXXAnlEvent.TGo4EventElement.fIden
E() -empty-                       XXXAnlEvent.TGo4EventElement.isCo
E() -empty-                       XXXAnlEvent.frData[]
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
E() -empty-
```

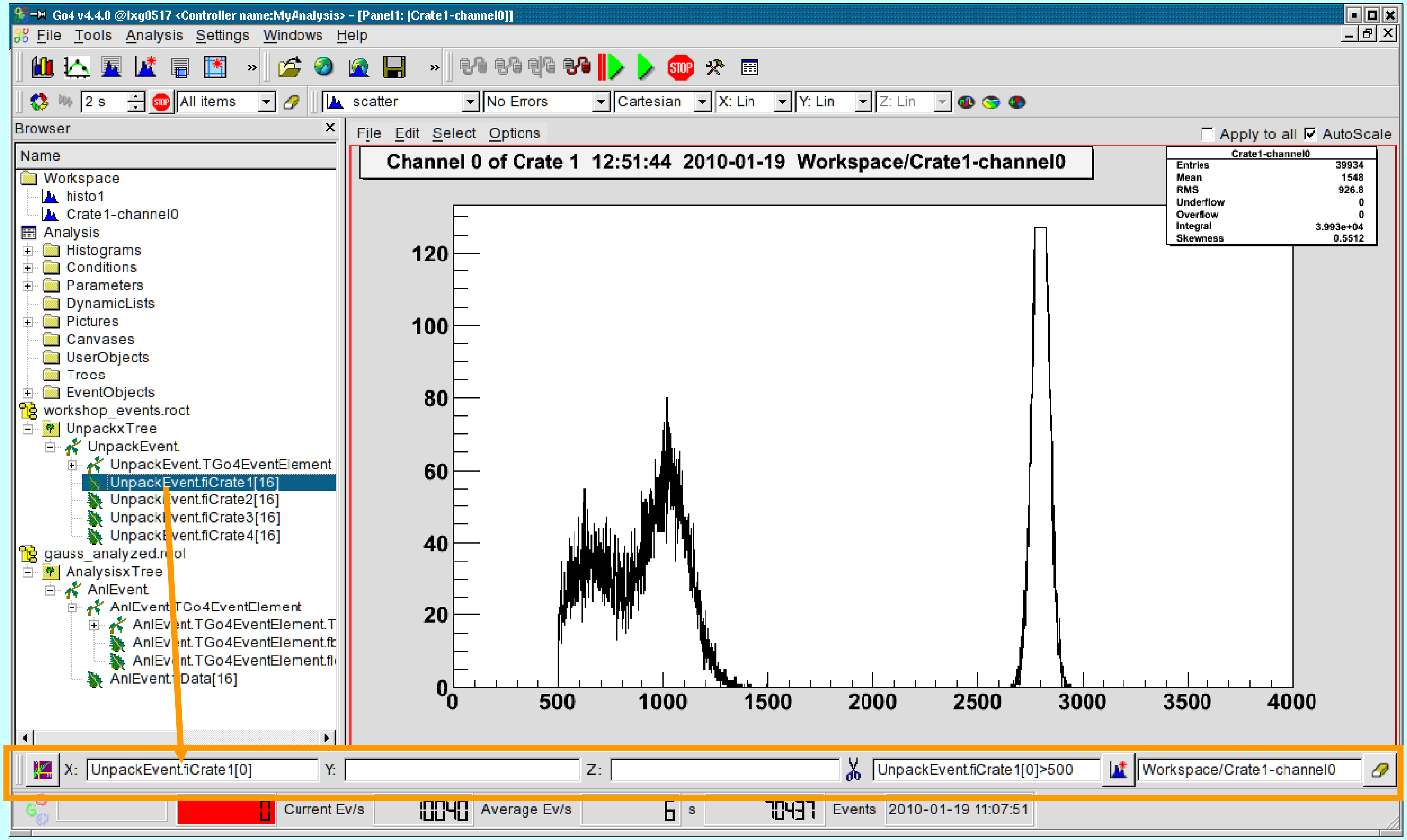
25%

IList | OList | Content : XXXAnlEvent.frData[0]>500 | RESET



From TBrowser: rmb menu on tree icon in file -> „StartViewer“

Go4 treeviewer



TTree::MakeClass()

```
TFile hfile(“workshop_events.root");
TTree* tr= dynamic_cast<TTree*>(hfile.Get(“UnpackXTree“));
if(tr!=0) tr->MakeClass(“MyAnalysis”);
```

workshop: `MakeClassExample.C` takes any tree name!

- **Generates code sceleton for analysis of any TTree**
  (files MyAnalysis.h, MyAnalysis.C)
- **Tree is analyzed by generated class MyAnalysis:**
  - **members contain each branch/leaf found in tree**
  - **constructor initializes tree/chain from file(s)**
  - **Init(TTree*) sets branch addresses to members**
  - **Show(int num) dumps entry #num**
  - **Loop() – here user can put own analysis code**