Go4 advanced features

J.Adamczewski-Musch, S.Linev

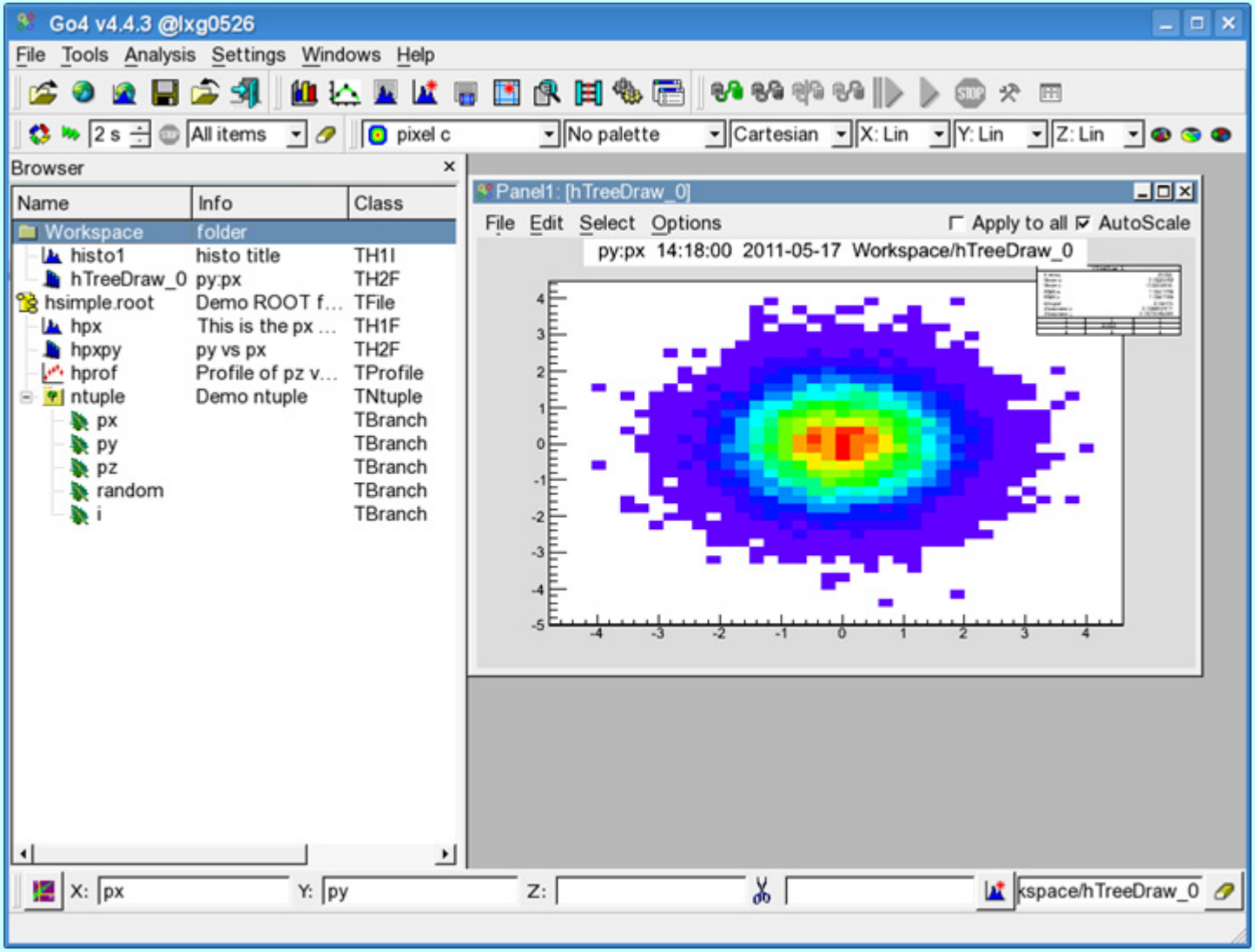
Overview

- Tree viewer
- Parameter editor
- Dynamic list editor
- Macros usage
- Fitting in go4

Go4 as ROOT files browser

- Go4 GUI can be used in offline as **ROOT files** browser
- To create example ROOT file:
  [shell] root -l $ROOTSYS/tutorials/hsimple.C –q
- Open produced file with go4:
  [shell] go4 hsimple.root
- Subdirectories and TTree brunches/leafs can be seen
- Display of any histogram by double click

Tree viewer


Go4 v4.4.3 @lxg0526
File Tools Analysis Settings Windows Help
2 s
All items
pixel c
No palette
Cartesian
X: Lin
Y: Lin
Z: Lin
Browser
Name Info Class
Workspace folder
histo1 histo title TH1I
hTreeDraw_0 py:px TH2F
hsimple.root Demo ROOT f... TFile
hpx This is the px ... TH1F
hpxpy py vs px TH2F
hprof Profile of pz v... TProfile
ntuple Demo ntuple TNtuple
px TBranch
py TBranch
pz TBranch
random TBranch
i TBranch
Panel1: [hTreeDraw_0]
File Edit Select Options
Apply to all
AutoScale
py:px 14:18:00 2011-05-17 Workspace/hTreeDraw_0
X: px
Y: py
Z:
kspace/hTreeDraw_0

Tree viewer

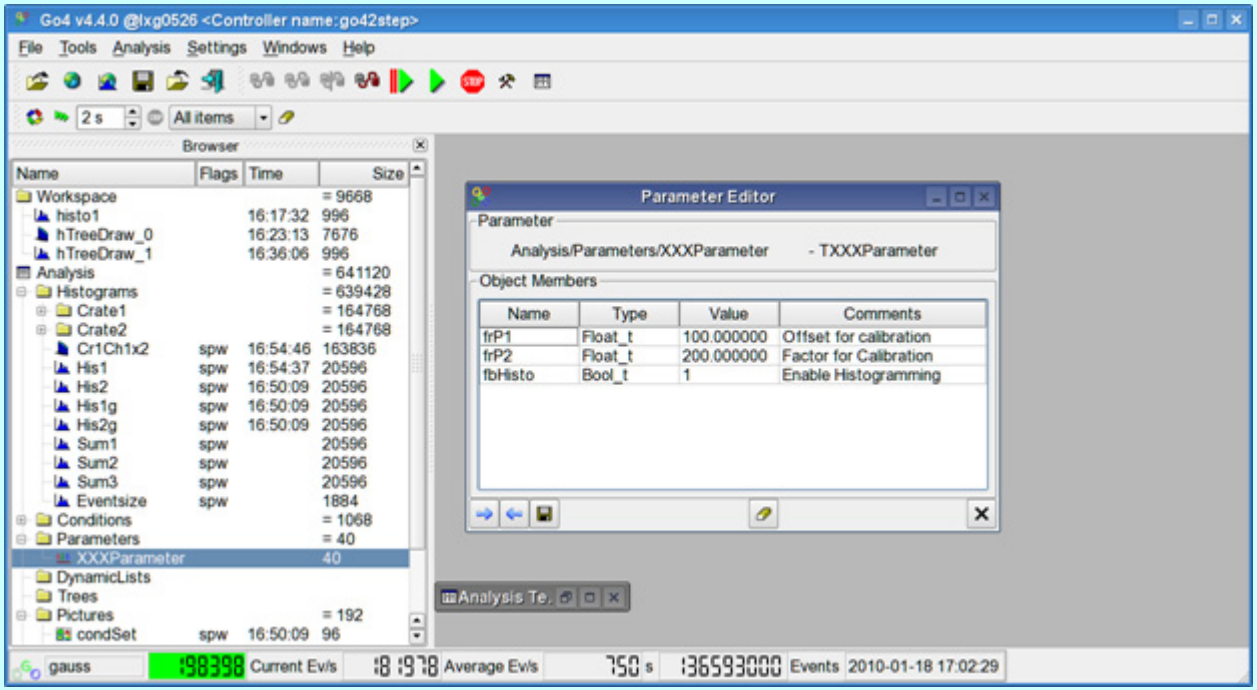
- Tree viewer activated by double-click on any tree leaf item
- 1-D, 2-D and 3-D tree draw is possible with cut condition
- Drag & drop of leafs names to viewer fields
- Automatic histogram creation or via special dialog

- Same in ROOT session:

```
shell$ root –l hsimple.root
root [1] new TBrowser;
root [2] “double click” leaf item in TBrowser
```

Parameter editor

- Editing of local/remote TGo4Parameter objects



User parameter class

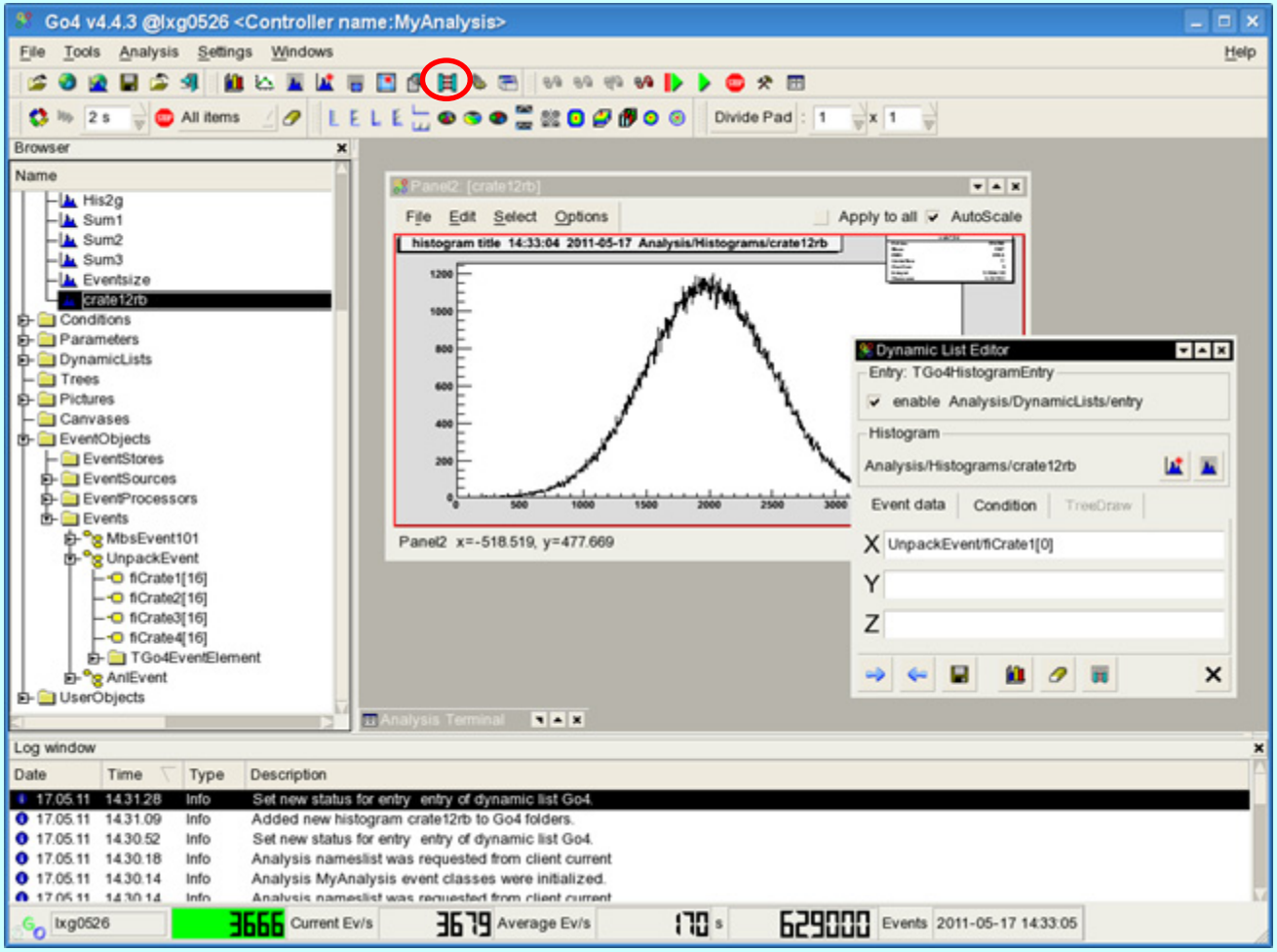
- User classes, derived from TGo4Parameter, could be used as coefficients table
- TGo4Parameter::UpdateFrom() method can be implemented for user control how values will be assigned in the analysis
- For simple cases UpdateFrom() implementation is no longer required

```
class TUserParameter : public TGo4Parameter {
   public:
      TUserParameter(const char* name = 0) :
          TGo4Parameter(name),
           fillRaw(kFALSE) {}

      Bool_t   fillRaw;  // control filling of raw histograms

   ClassDef(TUserParameter,1)
};
```

Parameter editor

- Basic data types, array of basic data types, TString and TGo4Fitter are supported for editing
- Array contents can be expanded / shrink
- Comments in class declaration are visible in parameter editor
- Class library **is not required** for parameter editing (only for file I/O)
- Parameter can be updated in analysis or stored / restored in file

Dynamic list editor



Dynamic list editor

- Browser displays analysis event structure and opened tree in analysis
- These data updated once per event and therefore can be used for histogramming
- Two alternatives for dynamic histogramming:
  - over any element(s) in events structures, including condition
  - TTree::Draw() operation

Macros usage with go4

- ROOT provides C-like scripting language CINT
- Advantage – no need to recompile before usage, very flexible
- Any go4 class (parameter, event, fitter) is available after ". go4login"
- To compile macro with ACLiC:

  [root] gSystem->AddIncludePath("-I$GO4SYS/include");

  [root] .x your_macro.C++

ROOT macros in user analysis

- Access to all object via **go4** – pointer on TGo4Analysis object
- Often use for configuration – macro can be changed without need for recompilation of whole analysis

```
void setup() {
   TUserParameter* par = (TUserParameter*) go4->GetParameter("UserPar");
   if (par==0) return;

   par->fillRaw = kTRUE;

   cout << "Setup done" << endl;
}
```

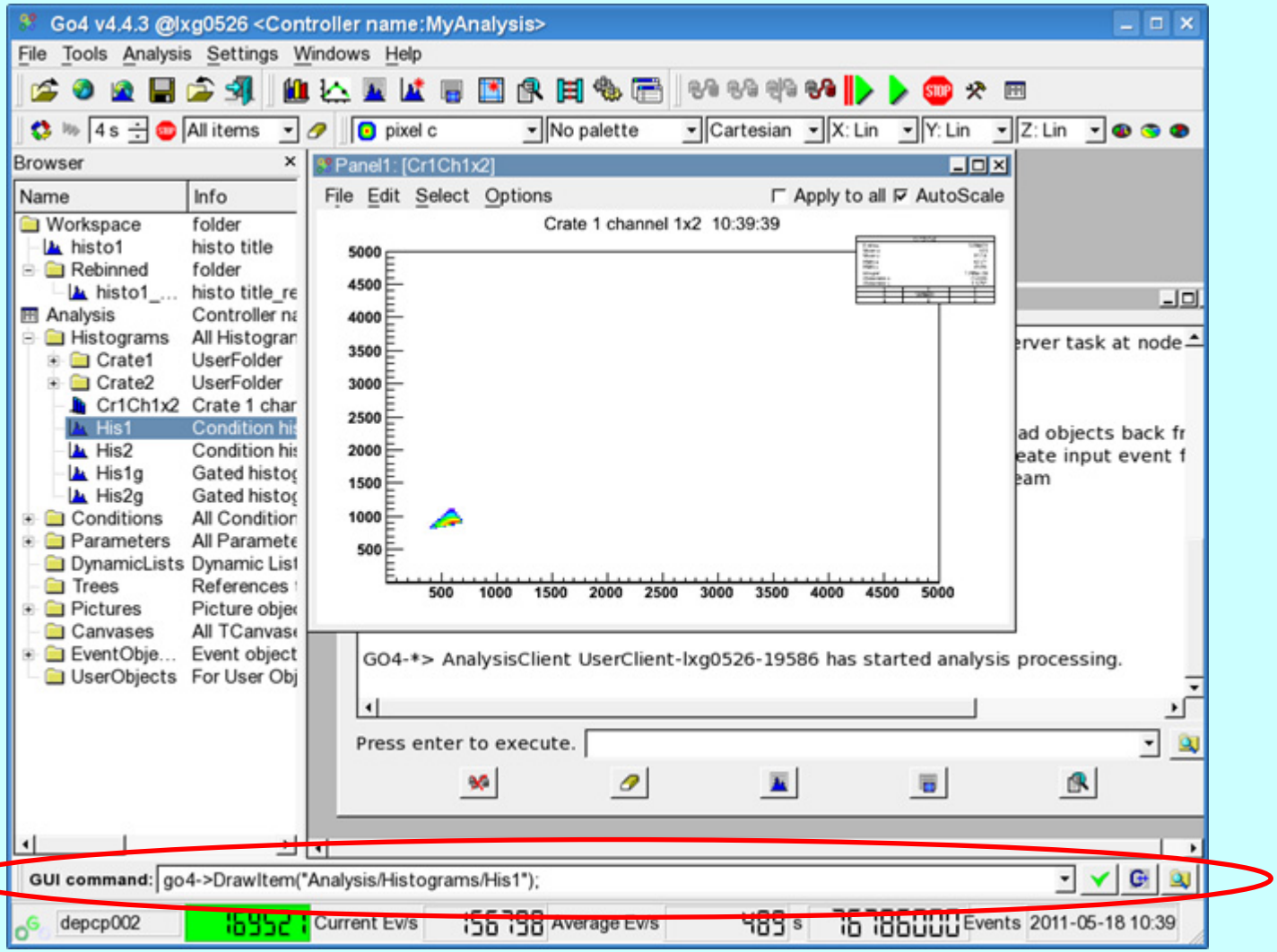
- Somewhere in processor constructor:

```
...
gROOT->ProcessLine(".x setup.C");
...
```

ROOT macros in the GUI

- Manipulate objects in the GUI
- Number of useful macros provided in $GO4SYS/macros
- Hotstart file is just ROOT script for GUI

ROOT macros in the GUI



ROOT macros in the GUI

- In the GUI **go4** is pointer on TGo4Script class, which provides access to main GUI functionality

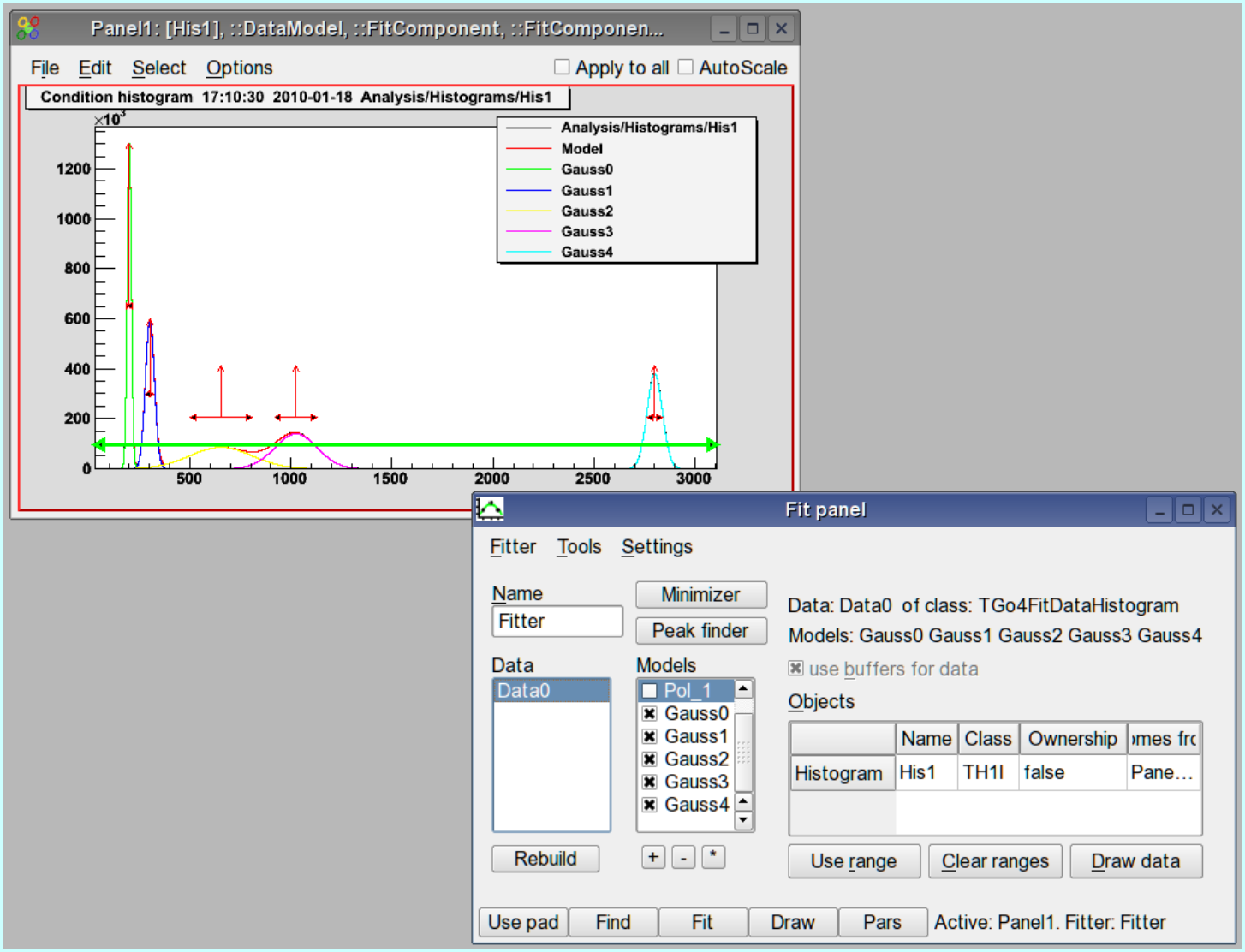
- Example of simple commands:

  - go4->MonitorItem("Analysis/Histograms/His1");
  - go4->DrawItem("Analysis/Histograms/His1");
  - go4->StartMonitoring(4);
  - go4->StopAnalysis();

Fitting in Go4

- Fit panel
- TGo4Fitter class
- Use of TGo4Fitter in macro
- Reuse of fitter in GUI / Analysis

Fit panel


Panel1: [His1], ::DataModel, ::FitComponent, ::FitComponen...
File Edit Select Options
Apply to all
AutoScale
Condition histogram 17:10:30 2010-01-18 Analysis/Histograms/His1
×10³
1200
1000
800
600
400
200
0
500
1000
1500
2000
2500
3000
Analysis/Histograms/His1
Model
Gauss0
Gauss1
Gauss2
Gauss3
Gauss4
Fit panel
Fitter Tools Settings
Name
Fitter
Minimizer
Peak finder
Data
Data0
Models
Pol_1
Gauss0
Gauss1
Gauss2
Gauss3
Gauss4
Rebuild
+
-
*
Data: Data0 of class: TGo4FitDataHistogram
Models: Gauss0 Gauss1 Gauss2 Gauss3 Gauss4
use buffers for data
Objects
Name
Class
Ownership
Histogram
His1
TH1I
false
Use range
Clear ranges
Draw data
Use pad
Find
Fit
Draw
Pars
Active: Panel1. Fitter: Fitter

Main features of fit panel

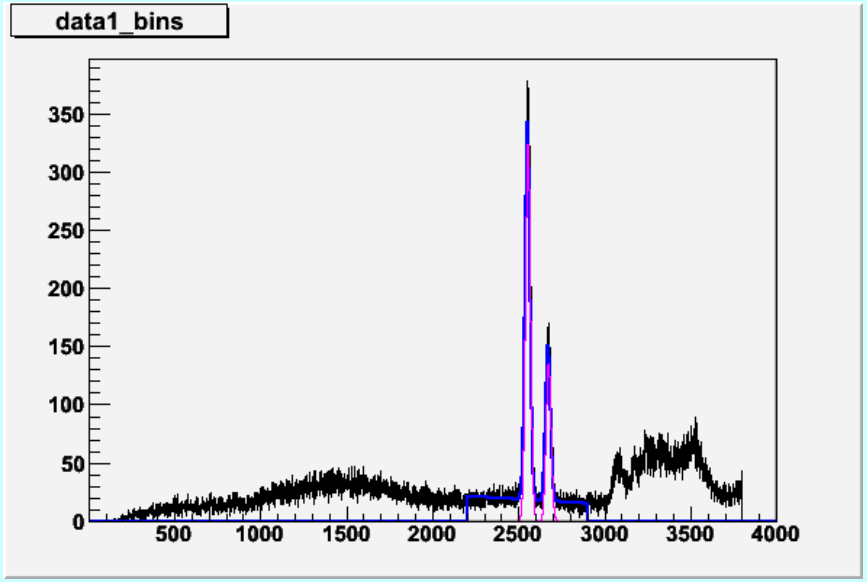
- Fitting of histograms / graphs for any kind of model
- Peak finder
- Manual change of model components / fit parameters
- Fitting and parameters view, parameters output
- Fit panel menu commands
- Different display modes:
  - show only model
  - show model components
  - use different panel for drawing
- Store fitter in file

TGo4Fitter class

- Central API class for fitting in go4
- Keeps all components to perform a fit
- Advanced features:
  - simultaneous fit of several histograms
  - fitting model via:
    - TFormula
    - sample histogram
    - function from shared library
  - flexible range settings
  - two-dimensional histograms fit
  - most are available via “Expert” mode in Fit panel
- See Fit tutorial on Go4 webpage
- Many examples in $GO4SYS/Go4FitExamples

Example1 from Go4FitExamples

```
// create fitter, select fit function and add standard actions list
   TGo4Fitter fitter("Fitter", TGo4Fitter::ff_ML_Poisson, kTRUE);

// add histogram to fitter, which should be fitted
   fitter.AddH1("data1", GetHistogram("hDeg120_P_c"), kTRUE, 2200., 2900.);

// create polynom of first order
   fitter.AddPolynomX("data1", "Pol", 1);

// create two gaussians
   fitter.AddGauss1("data1", "Gauss1", 2553., 15.);
   fitter.AddGauss1("data1", "Gauss2", 2672., 15.);

// execute all actions
   fitter.DoActions();

// draw data, full model and two gaussains
   fitter.Draw("#data1,Gauss1,Gauss2");
```

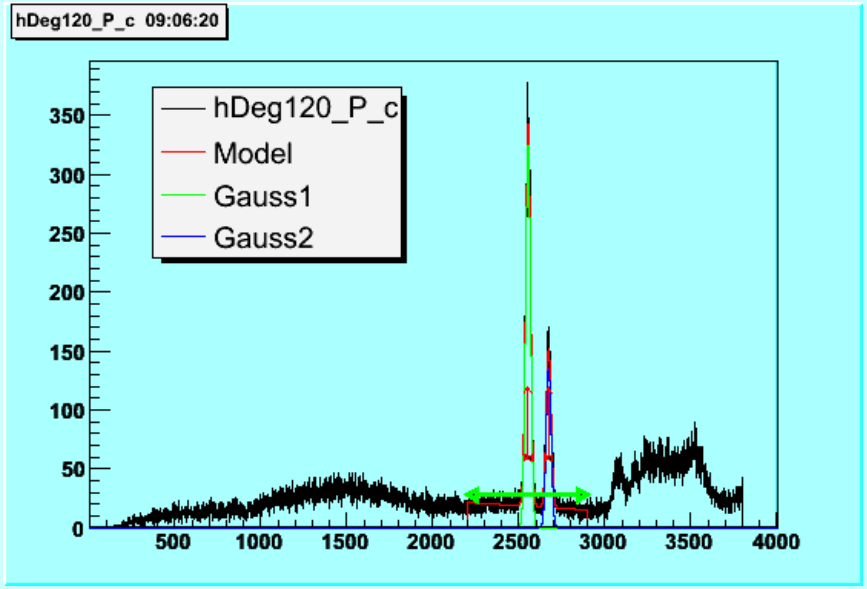


Reuse of fitter in GUI

- Store fitter in macro:

```
TFile* f = TFile::Open("fitter.root","recreate");
fitter.ClearObjects(0, kFALSE); // do not store histos with fitter
fitter.Write();
delete f;
```

- Load file with fitter in Go4 browser
- Drop fitter on histogram to be fitted
- Press **Fit** button



Reuse of fitter in macros

- Do fitting in fit panel
- Store fitter in file (via browser)
- Load fitter in macro and assign histograms:

```
// read fitter from file
TFile* f = TFile::Open("fitter2.root");
TGo4Fitter* fitter = (TGo4Fitter*) f->Get("Fitter");
delete f;
if (fitter==0) return;

// set histogram in fitter
fitter->SetObject("data1", GetHistogram("hDeg120_P_c"), kTRUE);

// make fitting
fitter->DoActions();

// just draw fitter
fitter->Draw("#data1,Gauss1,Gauss2");
```

- Repeat fitting as many time as necessary