ROOT Graphical User Interface

User Interface Classes

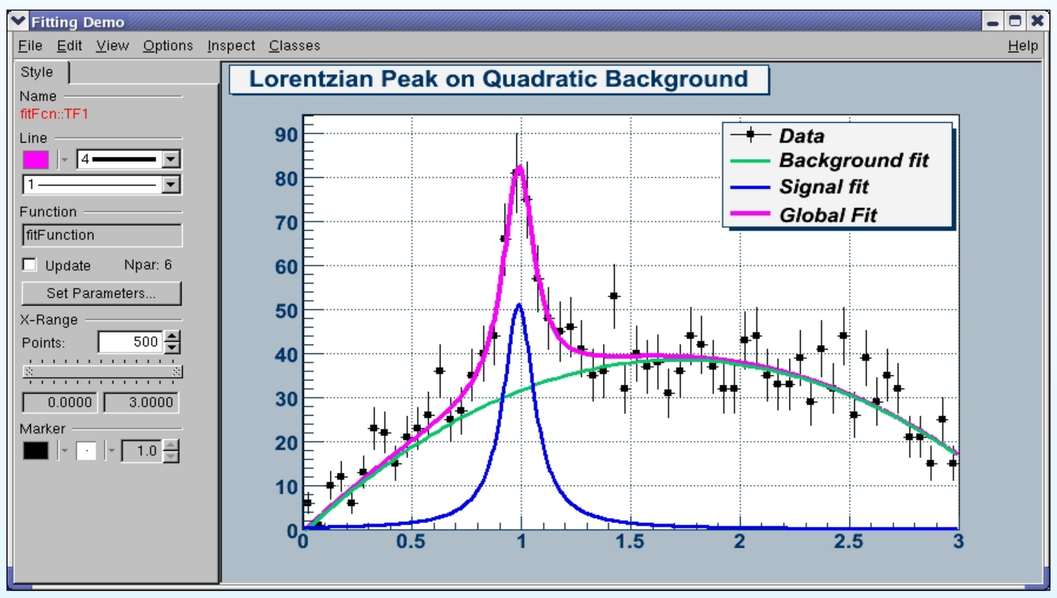
The ROOT GUI classes provide a rich and complete set of widgets allowing the construction of modern looking graphical user interfaces.

Like everything else in ROOT the GUI classes are fully cross platform and provide the same look and feel on either X11, Win32 or Mac OS X.

Complex GUI's can easily be constructed using a GUI builder, which allows widgets to be dragged and dropped into frames.

The GUI and the ROOT graphics classes are fully integrated and it is simple to embed a scientific data display into a GUI .

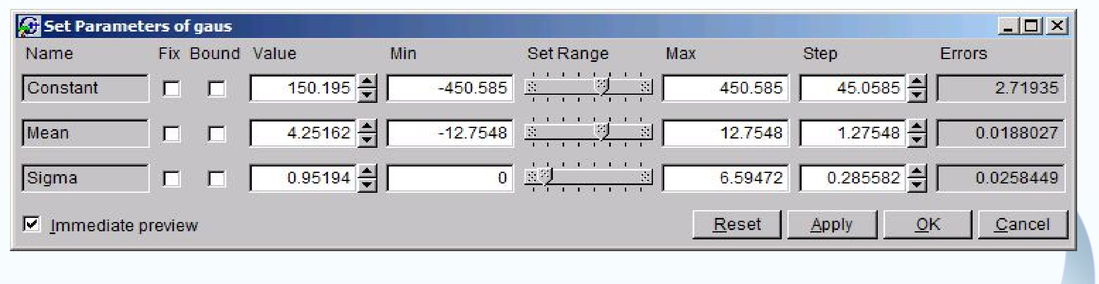
ROOT comes with many examples of high level GUI's like the browser, tree viewer, fit panel, etc.



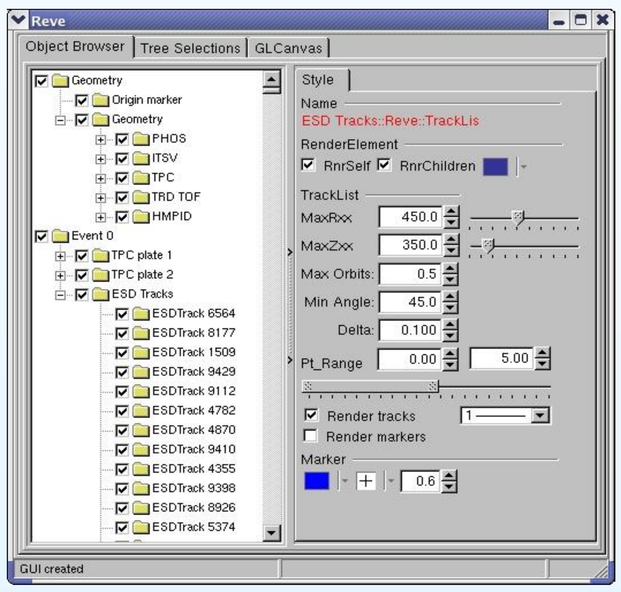
Signals / Slots

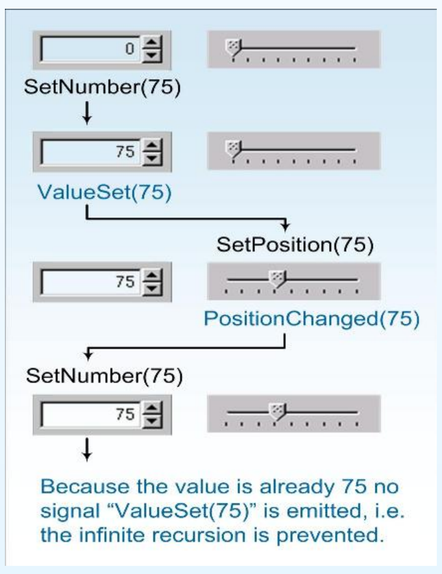
Using the signal/slot communication mechanism GUI elements can be easily connected to any number of action (slot) methods.

Signal/slots are integrated into the ROOT core and heavily use CINT to connect the signals to the slots and to call the slot methods when signals are issued.



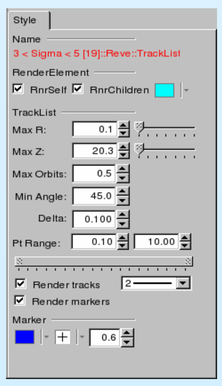
On interaction, widgets send out various signals. Any public object method can be connected to these signals.





Fast Prototyping

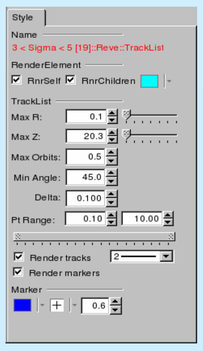
Like all classes in ROOT the GUI classes are fully scriptable allowing for fast prototyping via the embedded CINT C++ interpreter. In addition any GUI can be saved as C++ macro by typing ctrl-s when the mouse is over a GUI window. As macros can be stored in ROOT files one can envisage to store the GUI with the data:




```
root[] TMacro m("myApplication")
root[] m.ReadFile("myApplication.C")
root[] m.Exec()
root[] TFile f("myFile.root","recreate")
root[] m.Write()
root[] hpxpy.Write()
```

Executing the saved macro restores the complete application. Users can use the application to perform data analysis with the stored data in the same file.

```
root[] TFile f("myFile.root")
root[] f.ls()
TFile**         myFile.root
  TFile*          myFile.root
   KEY: TMacro    myApplication;1
   KEY: TH2F      hpxpy;1 py vs px
root[] TMacro *d = f.Get("myApplication")
root[] d.Exec()
```



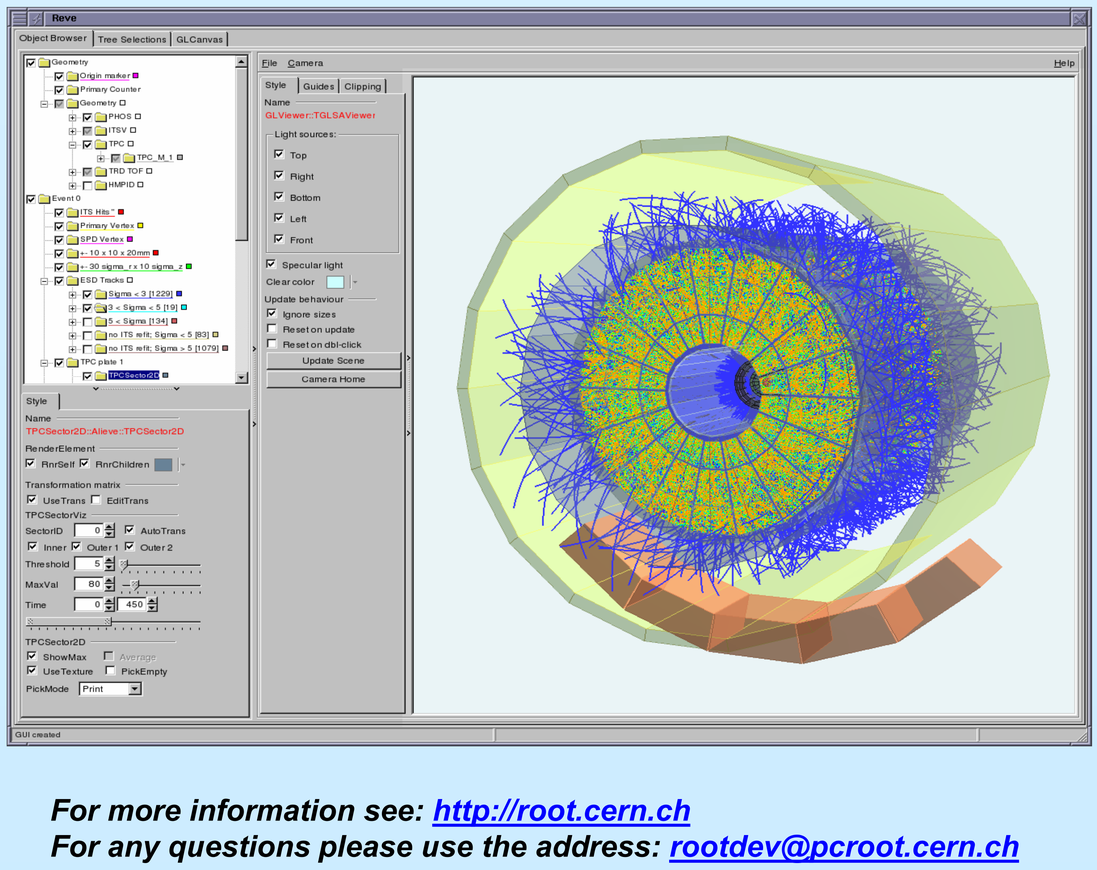
All necessary libraries are loaded on demand via the plug-in manager.

Examples

The ALICE Event Visualization Environment (AliEVE) is based on ROOT and its GUI, 2D/3D graphics classes. A small application kernel provides for registration and management of visualization objects. CINT scripts are used as an extensible mechanism for data extraction, selection and processing as well as for steering of event-related tasks.

AliEVE is used for event visualization in offline and high-level trigger frameworks.

The event below is a simulated peripheral lead-lead collision at 5.5 TeV/nucleon with 2600 reconstructed tracks (pT>100 MeV, |eta| < 1.5)



For more information see: http://root.cern.ch
For any questions please use the address: rootdev@pcroot.cern.ch