ROOT Graphical User Interface

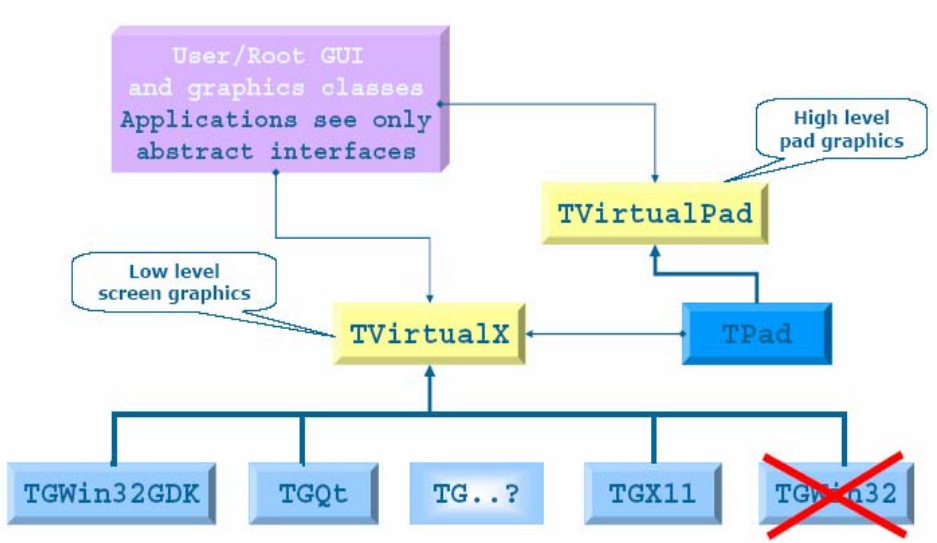
Ilka Antcheva, Bertrand Bellenot, Valeri Fine,
Valeriy Onuchin, Fons Rademakers

Overview

- Main Goals
- GUI Widgets
- Graphics Editor
- GUI Builder
- Tree Viewer
- Undo/Redo Tools

Main Goals

- Cross-platform GUIs – consistent look everywhere
- All machine dependent low graphics calls abstracted via TVirtualX
  - X11
  - Win32GDK
  - Qt layer - standard ROOT "plug-in" share library, allows to be turned on/off at run time with no changes of the user's code

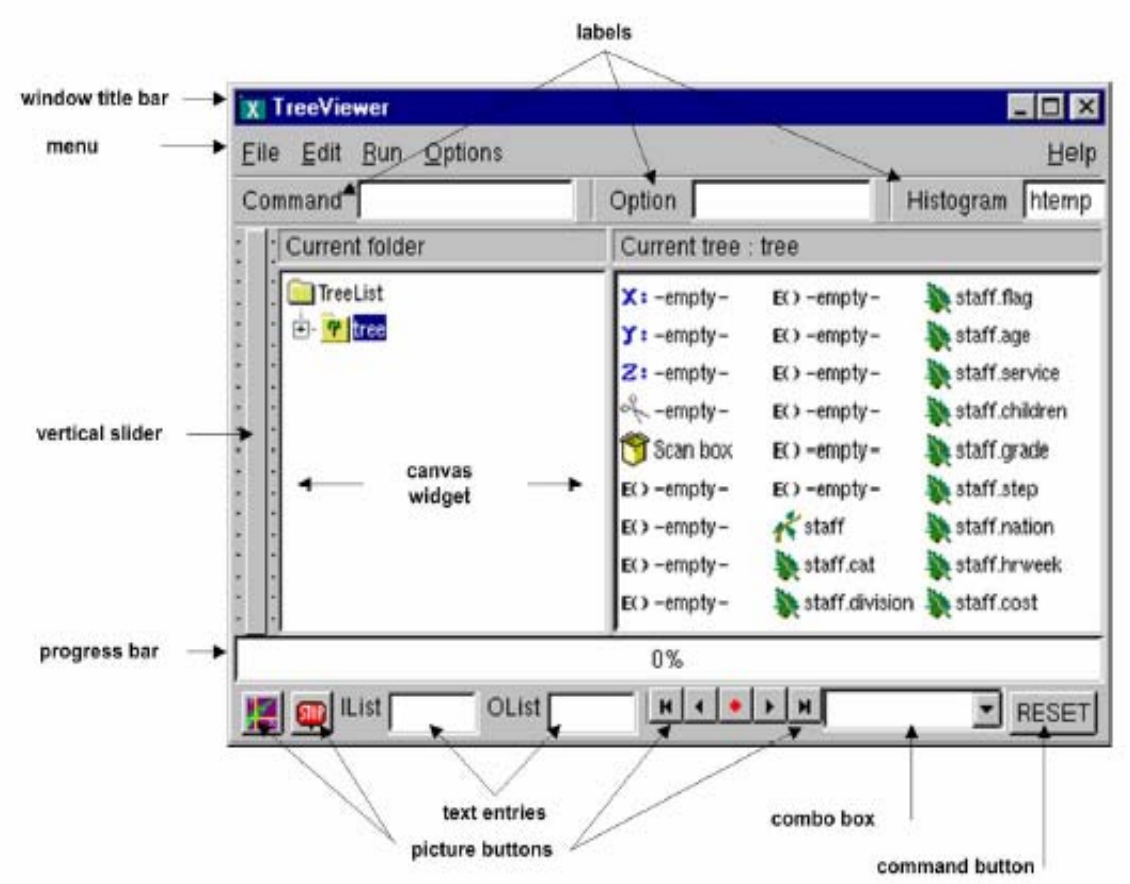


- Improve the GUI design and performance; modify and iterate as much as necessary
- Integrate all system components: software, documentation, help functions, tutorials

GUI Widgets (1)

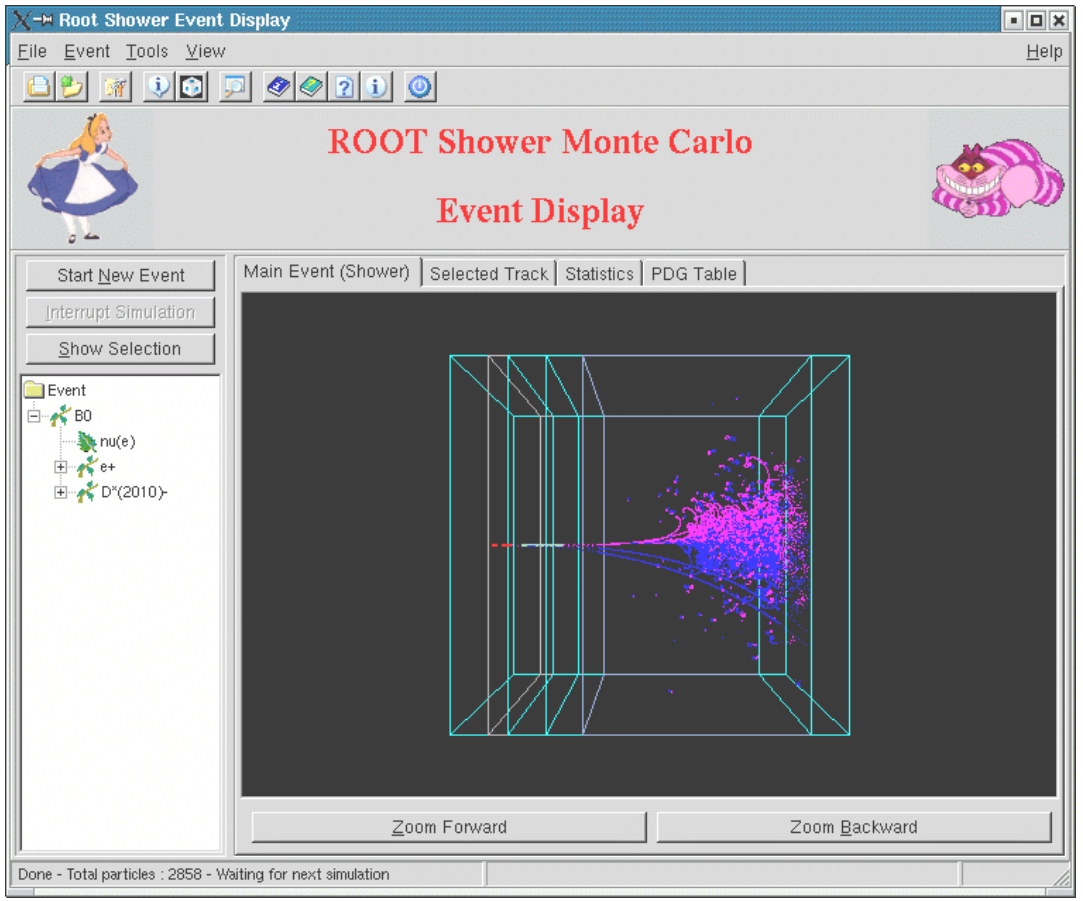
Current Status

- Based on the XClass library from Hector Peraza
- Provide standard components for application environment with windows 'look and feel'
- Object-oriented, event-driven programming model



GUI Widgets (2)

- Containers - use of ROOT container classes for fast object look up
- Any widgets can be extended via inheritance
- Layout managers
- Signals/slots communication
- Conventional model



GUI Widgets (3)

Next Steps

- Not finished GUI tasks
  - Keyboard navigation
  - Menu hot keys:

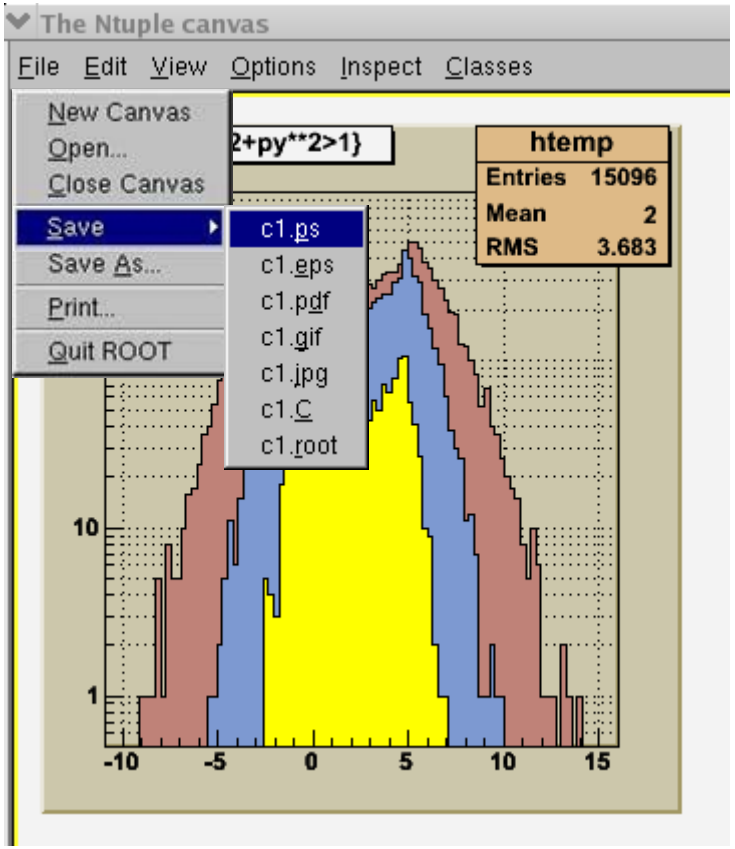
    Alt + F = pops up File menu

    S = Save, pops up a submenu

    P should create c1.ps
  - Dialogs:

    Enter = OK, Esc = Cancel
  - Combo boxes:

    Up/Down arrows, Home, End, PgUp, PgDn
  - List view improvements, etc.
- Cleanup tools

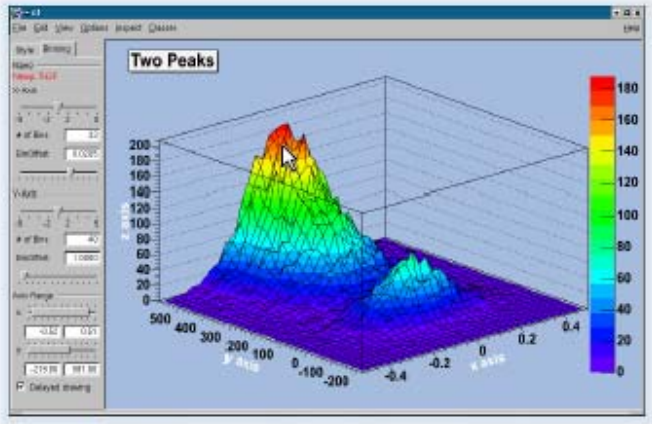


GUI Widgets (4)

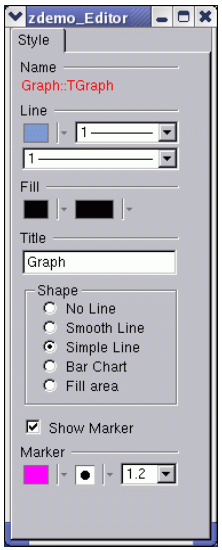
- Code optimization
  - Layout algorithms
  - GUI Dialogs
- Qt layer
  - Validation tests of interaction with ROOT GUI classes with the same ‘look and feel’
  - To solve problems with so-called "popup widgets", like menus, drop down combo boxes, etc. and actions that require full-scale X11 - like mouse pointer grabbing
- To complete the reference documentation of GUI classes

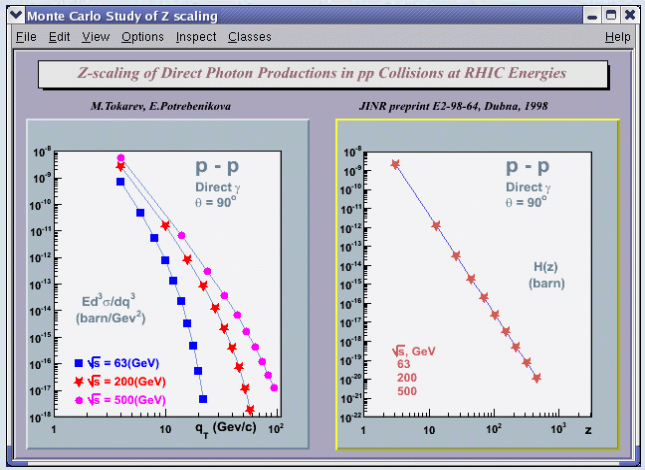
Graphics Editor (1)

- ROOT graphics editor can be:
  - Embedded – connected only with the canvas in the application window



  - Global – has own application window and can be connected to any created canvas (under development)



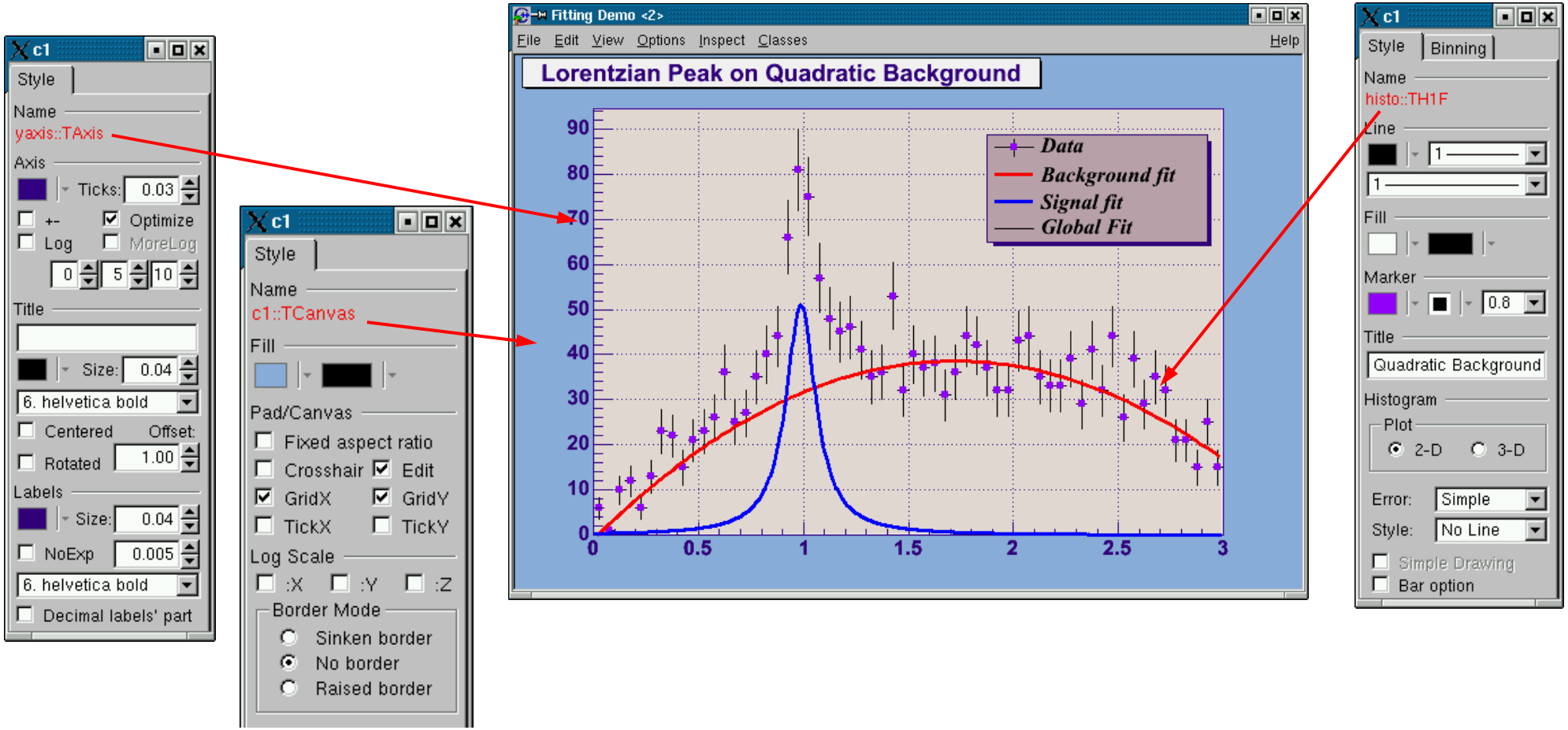


Graphics Editor (2)

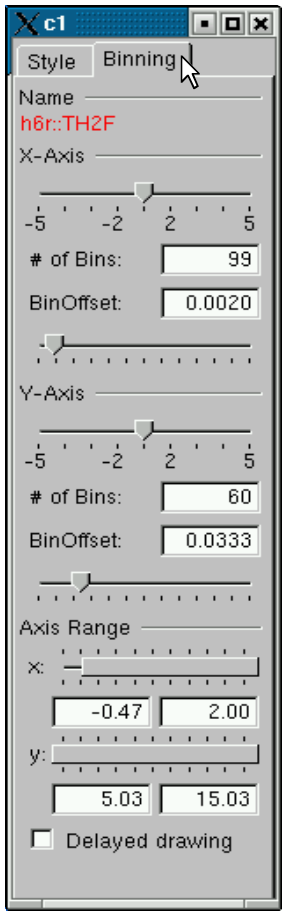
- Modular – it loads the corresponding object editor xxxEditor according to the selected object xxx in the canvas respecting the class inheritance.
- Can be extended easily by any user-defined object editor - this makes GUI design easier and adaptive to the users' profiles
- This way the GUI complexity is reduced by hiding some interface elements and revealing them when necessary.
- Rules to follow:
  - Derive in the code the object editor from the base class TGedFrame
  - Correct naming
  - Register the object editor in the list TClass::fClassEditors

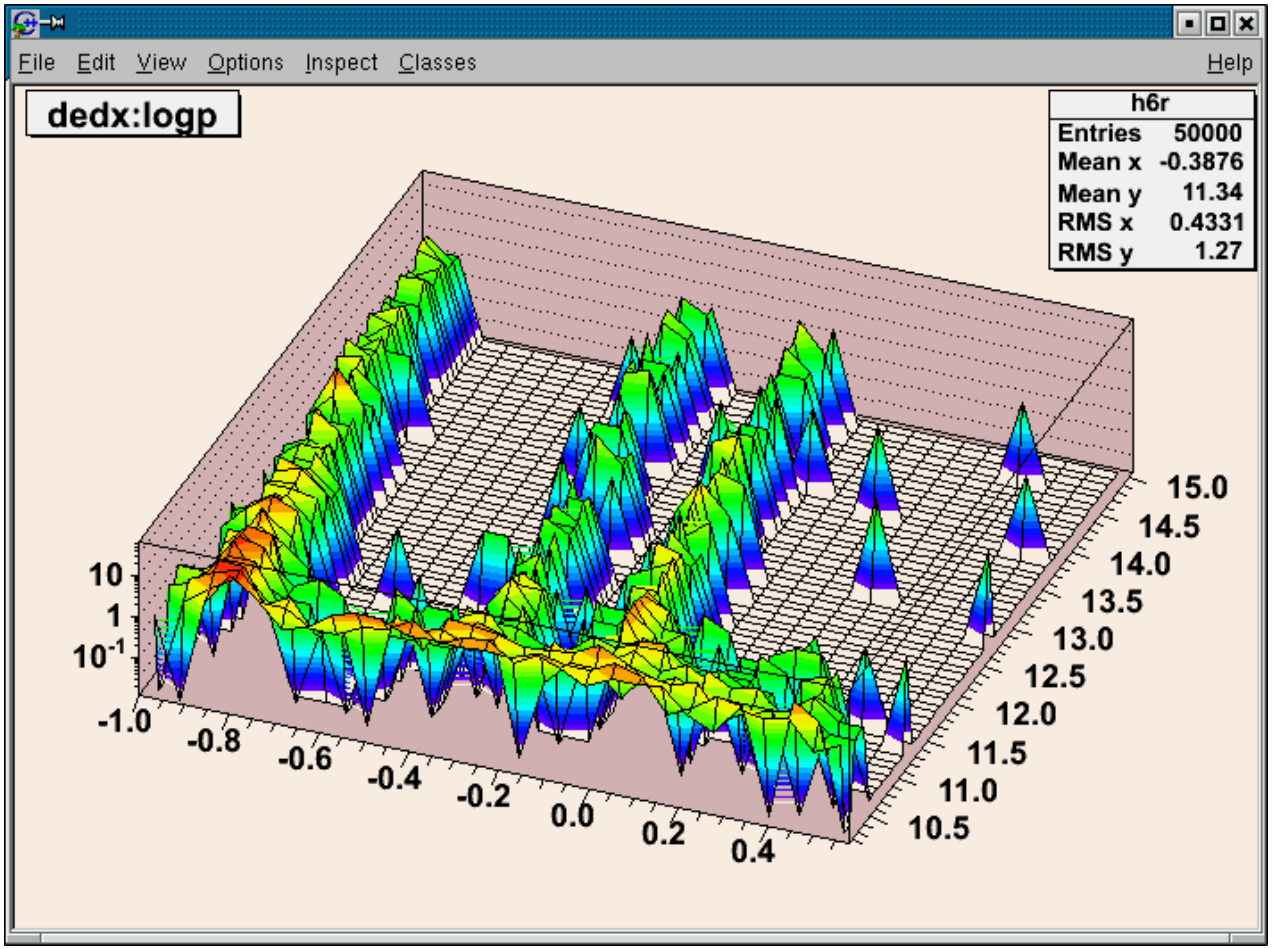
Graphics Editor (3)

- Different object editors



Graphics Editor (4)

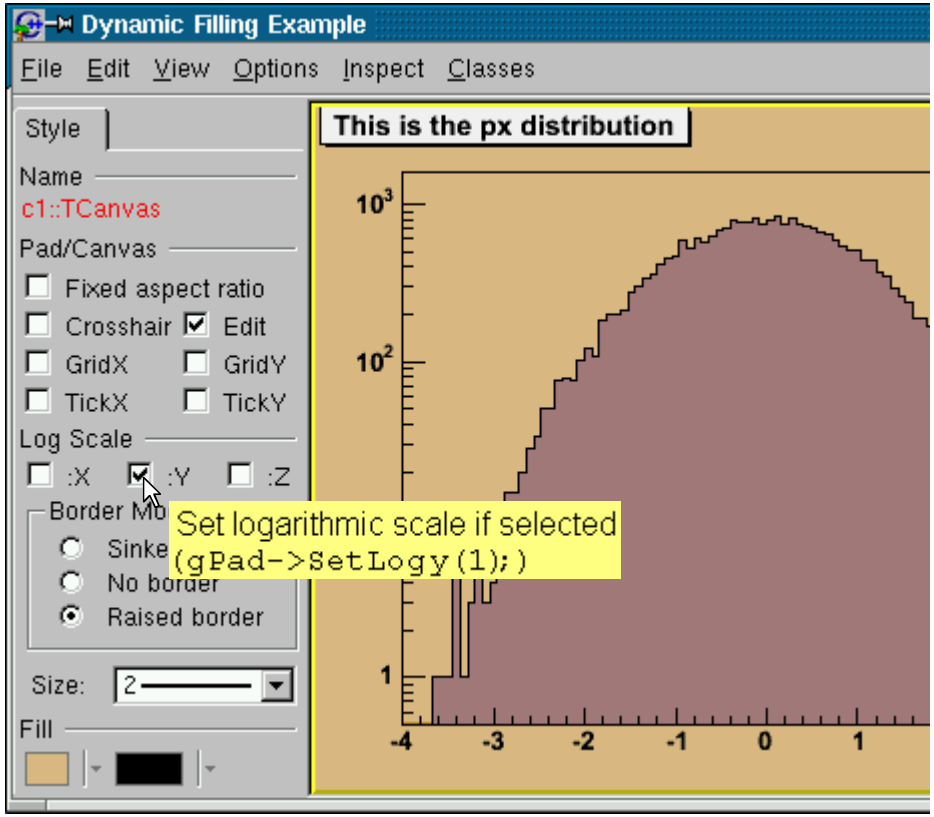




Graphics Editor (5)

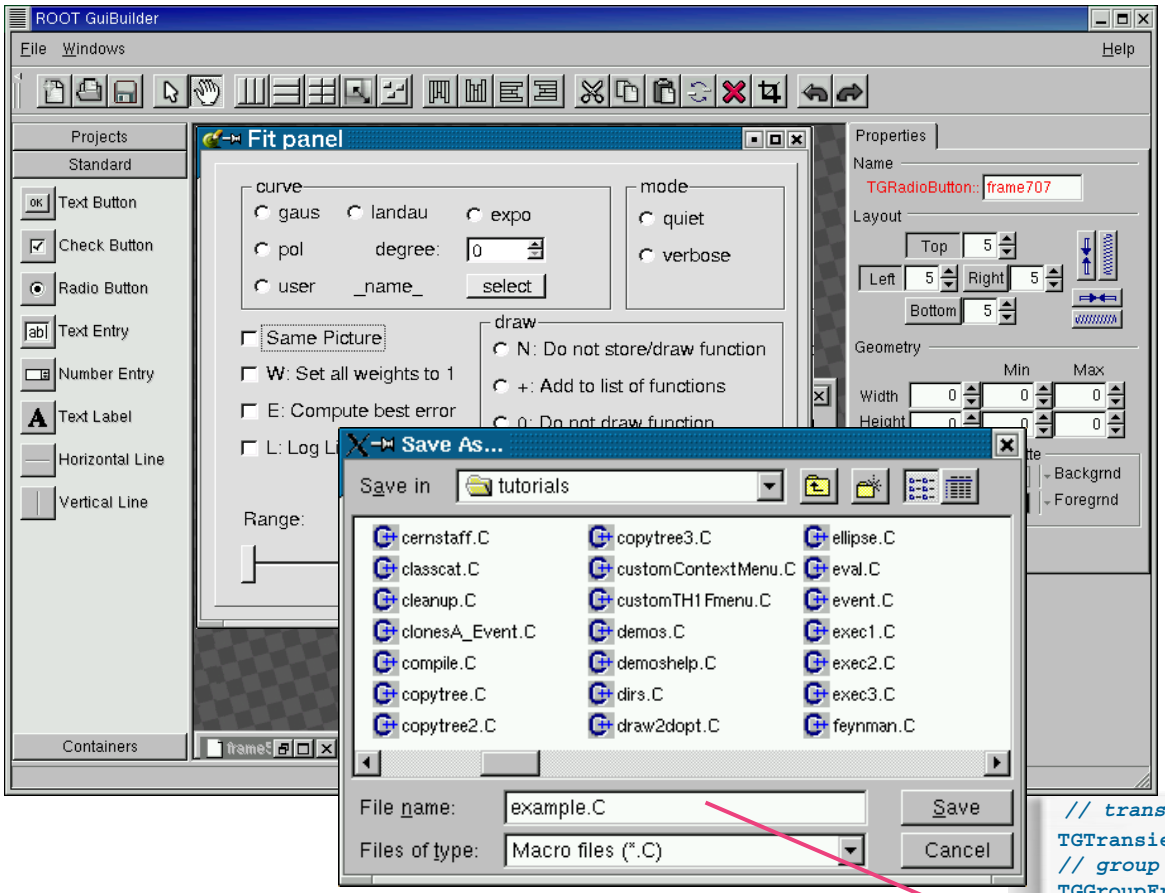
Next Steps

- To include ROOT commands in tool tips of the GUI widgets
- Help
- Global graphics editor
  - To show the related canvas title
  - Close button
- Hide/Show objects in a canvas
- New object editors development
- Style manager – summer student project
- Fit Panel GUI




```
root[9] gPad->SetLogy(1);
```

GUI Builder (1)



- GUI Builder greatly simplifies the process of designing GUIs based on ROOT widgets' classes.
- Using *Ctrl+S* or *SaveAs* dialog, users can generate C++ code in a macro that can be edited and executed via the CINT interpreter:

  root[0] .x example.C

```
// transient frame
TGTransientFrame *frame2 = new TGTransientFrame(gClient->GetRoot(),760,590);
// group frame
TGGroupFrame *frame3 = new TGGroupFrame(frame2,"curve");
TGRadioButton *frame4 = new TGRadioButton(frame3,"gaus",10);
frame3->AddFrame(frame4);
```

```
frame2->SetWindowName("Fit Panel");
frame2->MapSubwindows();
frame2->Resize(frame2->GetDefaultSize());
frame2->MapWindow();
}
```

GUI Builder (2)

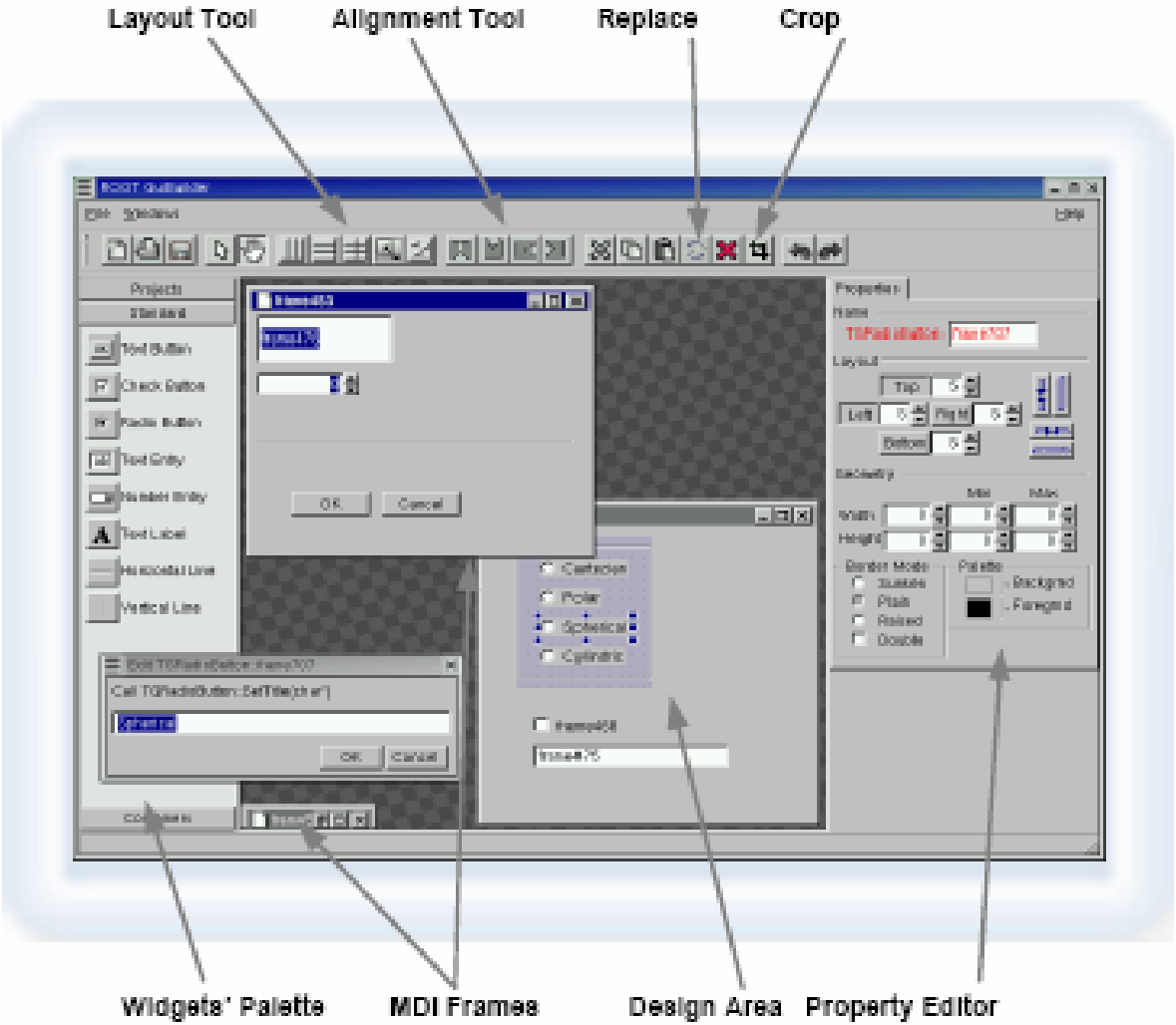
Current status

- Tests and validation of the current version
  - Layout a GUI quickly by dragging widgets, setting layout managers, changing options in the right-click context menus.
  - Final design can be saved as a C++ macro

Next steps

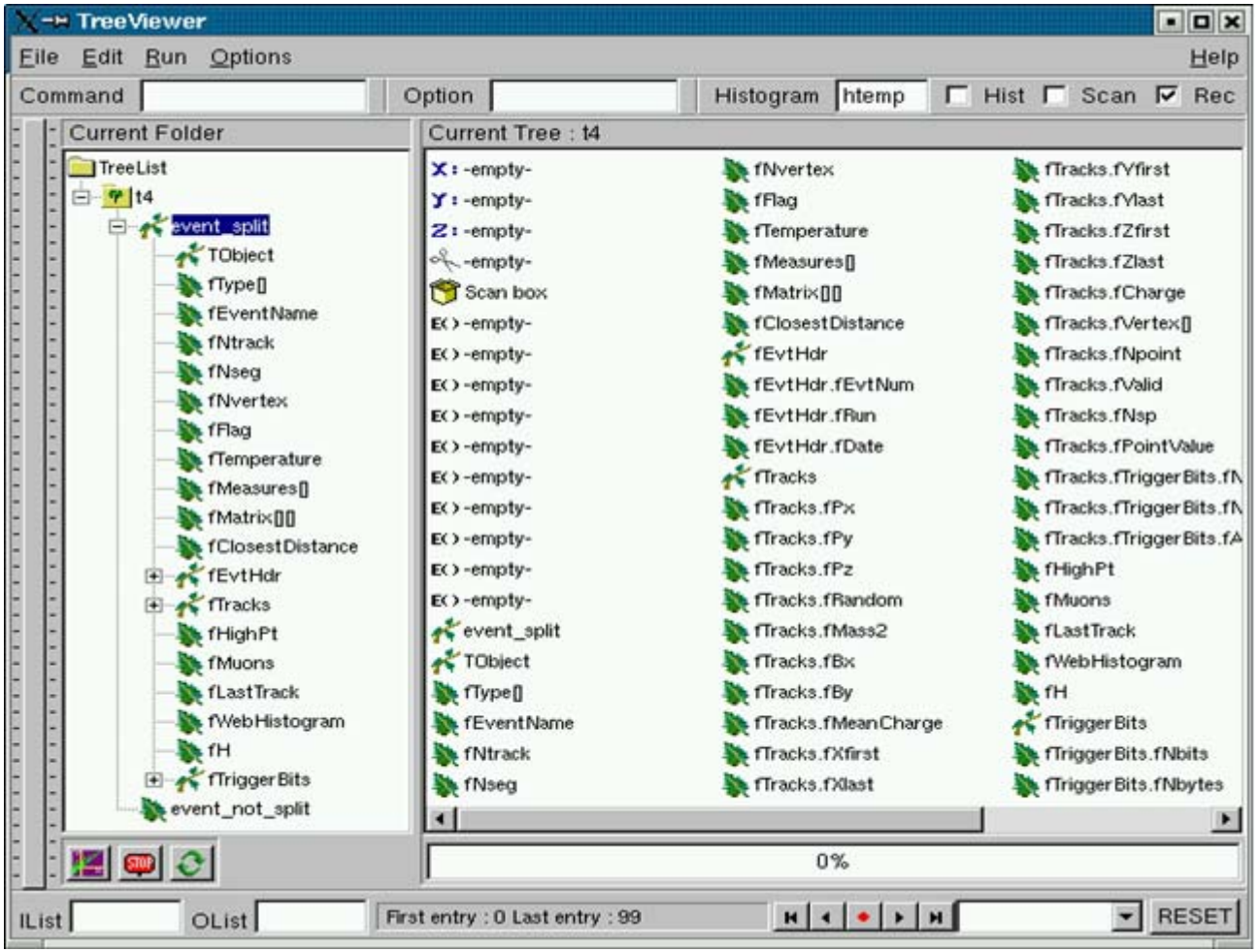
- To complete the GUI widgets' palette with combo/list boxes, double sliders, list view, list tree, shutters, button group, etc.
- To develop tools for signals/ slots mechanism of communication.
- To provide examples for several basic types of GUIs (as tutorials)

GUI Builder (3)



Tree Viewer

- Improvements of the Tree Viewer GUI



Undo/Redo Tools

- Allow users to recover from mistakes - very important part of GUI that will provide:
  - A stack of states/actions to go back
  - Confirmation of destructive actions: overwrite, delete, etc.
- Main idea: all editing in an application is done by creating instances of so-called command objects
- Tests and validation of already implemented classes:
  - TQCommand – each command knows how to undo its changes to bring the edited object back to its previous state.
  - TQCommandHistory
  - TQUndoManager – recorder of undo and redo operations; it is the command history list which can be traversed backwards and upwards performing undo/redo operations.