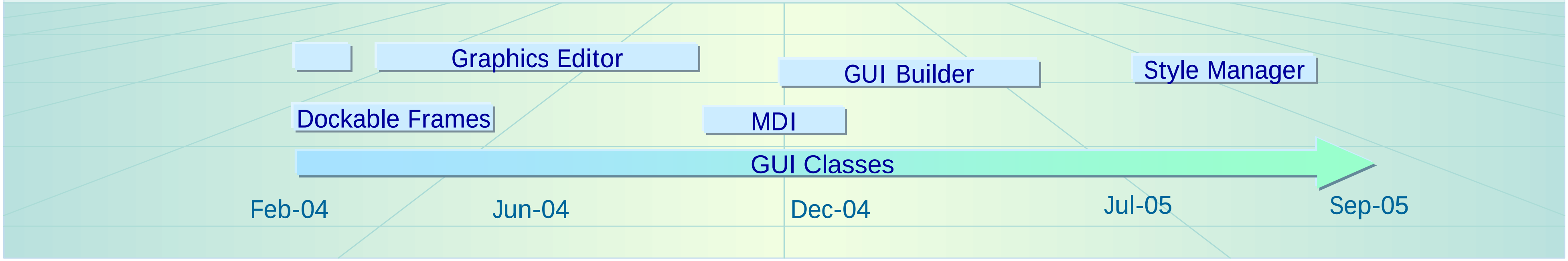
GUI Status and Development

Ilka Antcheva

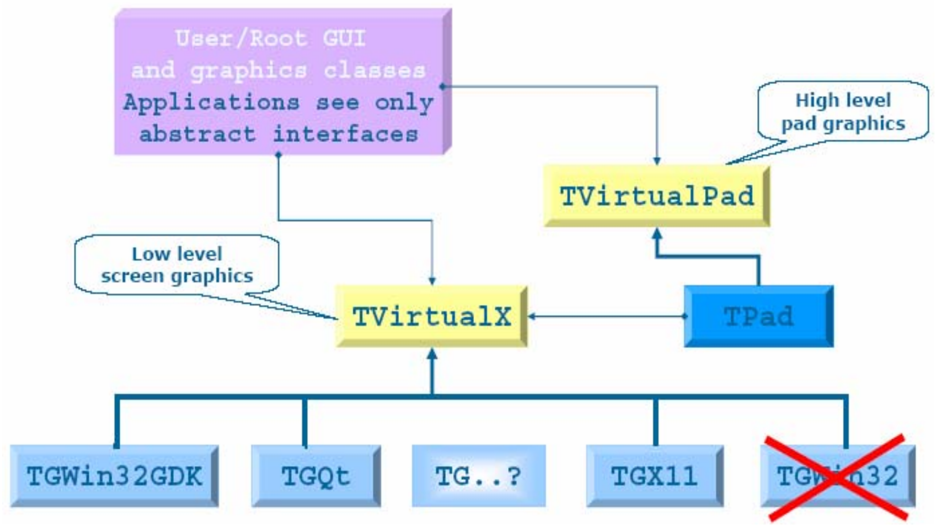
Overview

- Status
- GUI Classes
- Graphics Editor
- Style Manager
- GUI Builder
- Next Steps



Status (1)

- Cross-platform GUIs – consistent look everywhere
- All machine dependent low graphics calls abstracted via TVirtualX
  - X11
  - Win32GDK - solved problems with not thread safe gdk environment
  - Qt layer - standard ROOT "plug-in" shared library, allows to be turned on/off at run time with no changes of the user's code



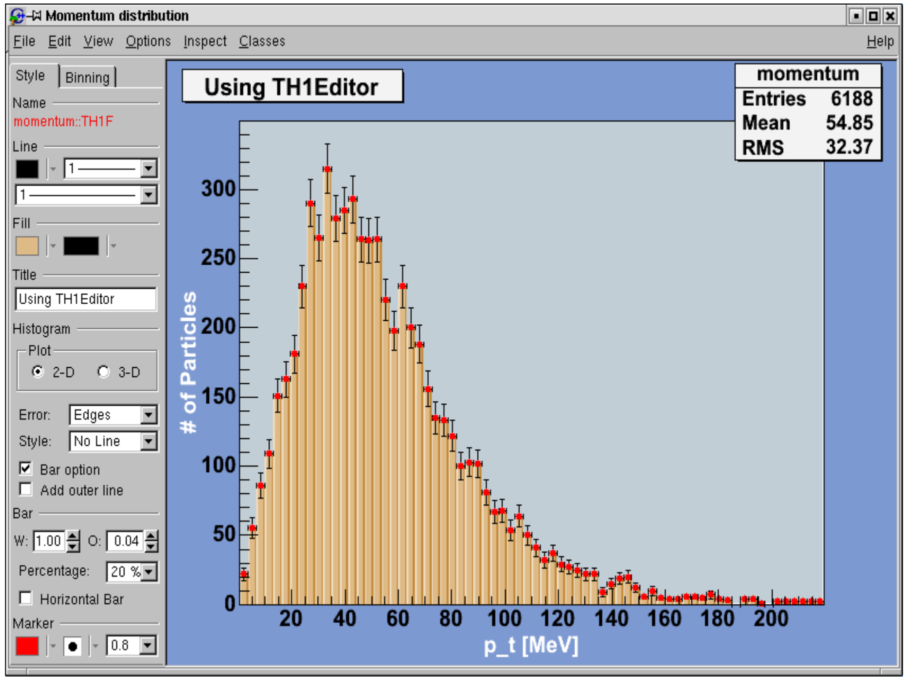
- The benefit of applications running on different platforms is obvious - it increases the program's robustness, makes their maintenance easier and improves the reusability of the code. No need to implement specific code for each platform.

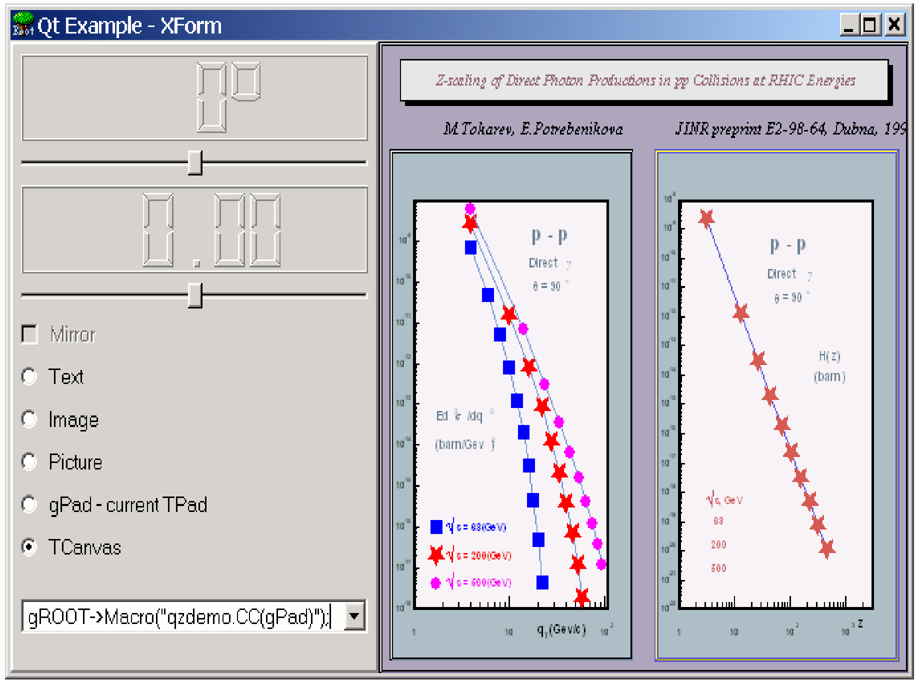
Status (2)

ROOT and Qt (*see the talk about Qt & ROOT by Valeri Fine*)

- ROOT controls the event loop via TApplication::Run()
- Transformed QEvent into Event_t structure allows event piping

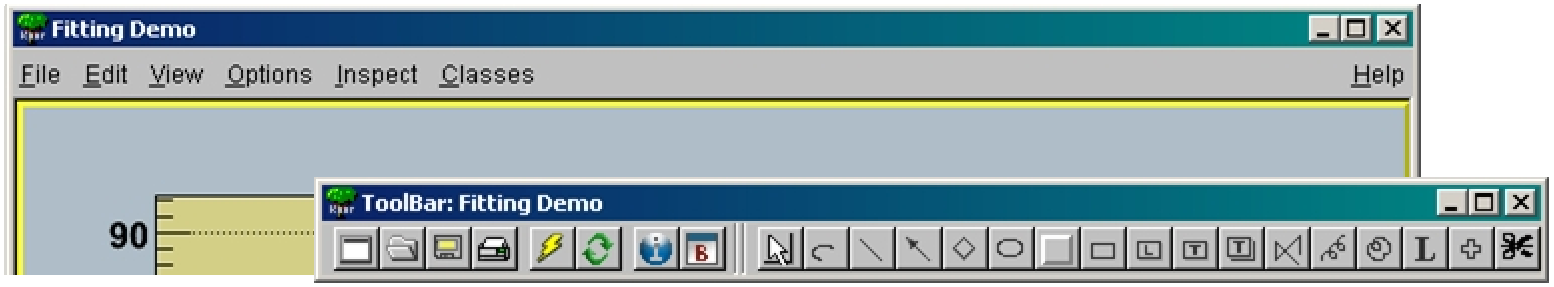
- Qt controls the event loop via QApplication::exec()
- TQtWidget class provides the embedded ROOT canvas



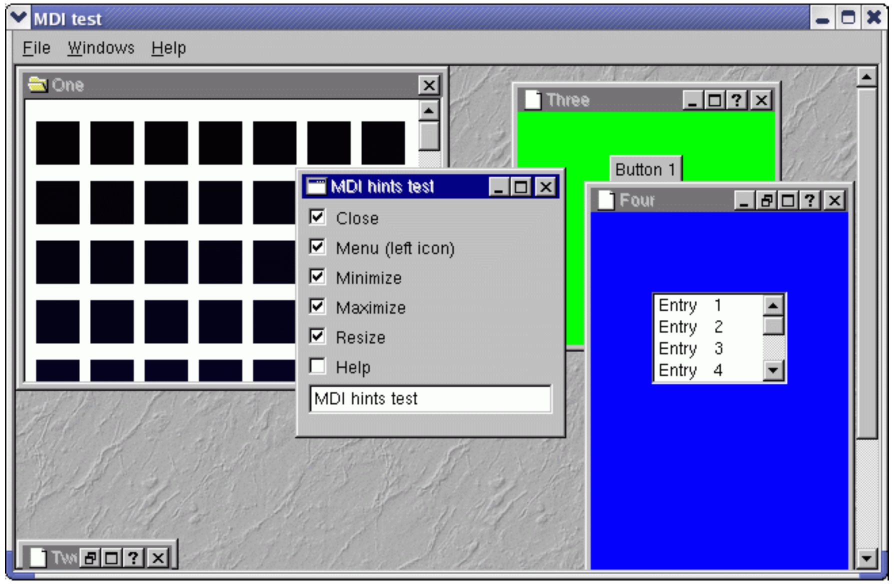

Qt Example - XForm
Mirror
Text
Image
Picture
gPad - current TPad
TCanvas
gROOT->Macro("qzdemo.CC(gPad)");
Z-scaling of Direct Photon Productions in pp Collisions at RHIC Energies
M.Tokarev, E.Potrebenikova
JINR preprint E2-98-64, Dubna, 199
p - p
Direct γ
θ = 90°
H(z)
(barn)
q⊥(GeV/c)
z

GUI Classes (1)

- TGDockableFrame widget - allows the undocking/docking of menus, tool or status bars, or the collapsing of these bars.



- MDI (Multiple Document Interface) widgets

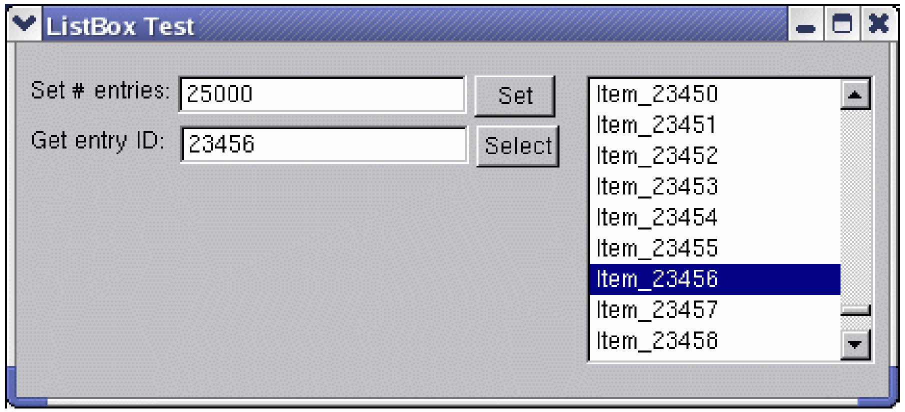


GUI Classes (2)

- Cleanup methods

```
TGCompositeFrame *fr = new TGCompositeFrame(this, 80, 20, kHorizontalFrame);
fr->AddFrame(new TGLabel(fr, "Size:“),
                   new TGLayoutHints(kLHintsLeft | kLHintsCenterY, 3, 0, 1, 1));
// all objects (frames and layout hints) must be unique
. . .
fr->Cleanup();
```

- ‘Pseudo-windows' concept allows to draw & scroll > 10 000 items
  - TGListView
  - TGListBox
  - TGListTree
  - TGComboBox


ListBox Test

Set # entries: 25000 Set

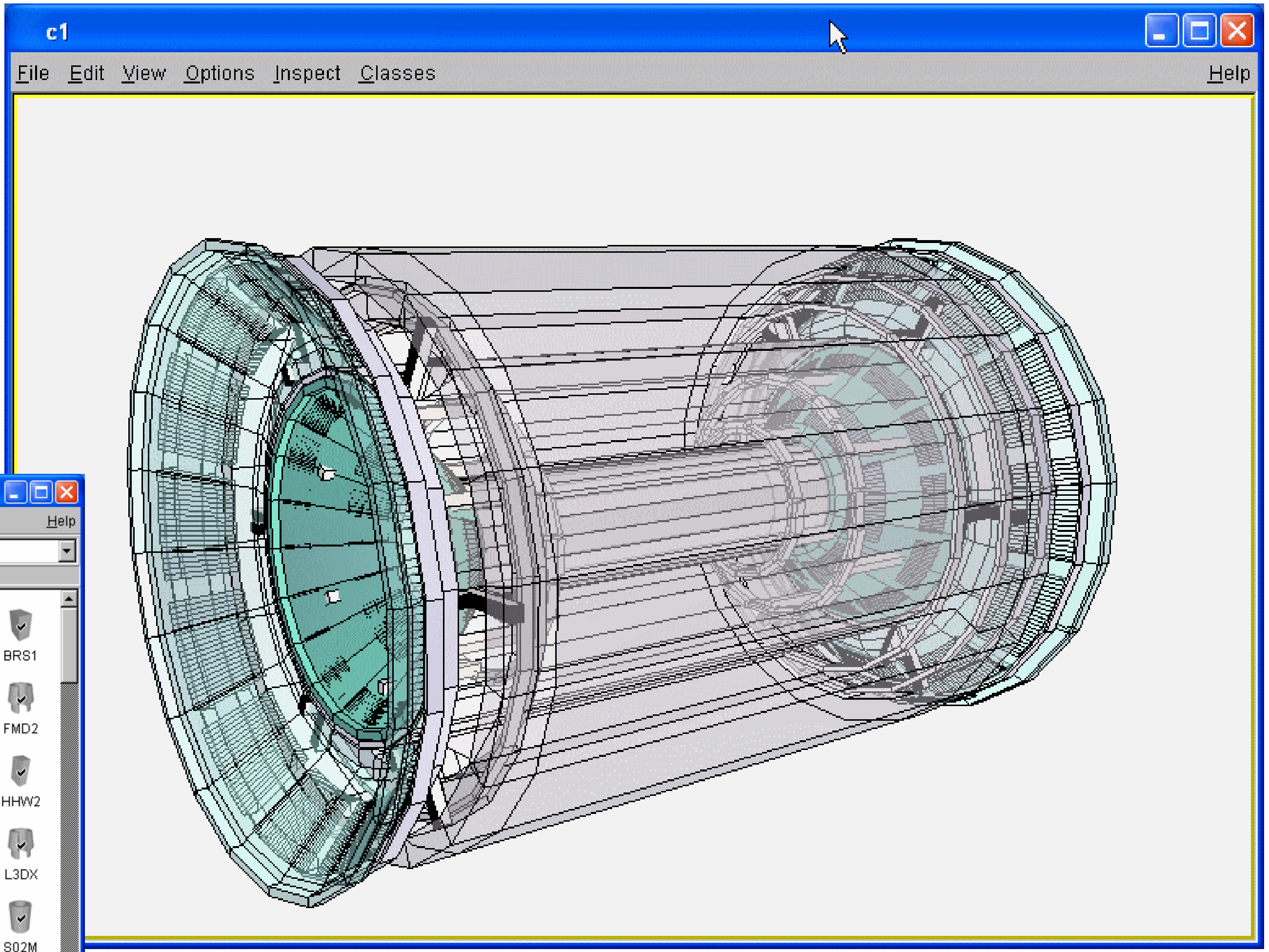
Get entry ID: 23456 Select

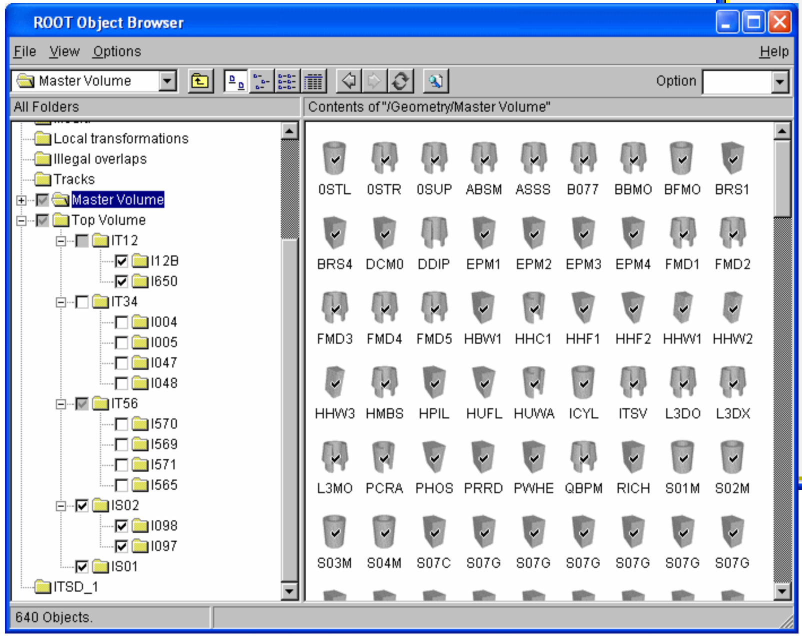
```
Item_23450
Item_23451
Item_23452
Item_23453
Item_23454
Item_23455
Item_23456
Item_23457
Item_23458
```

GUI Classes (3)

TGListTree

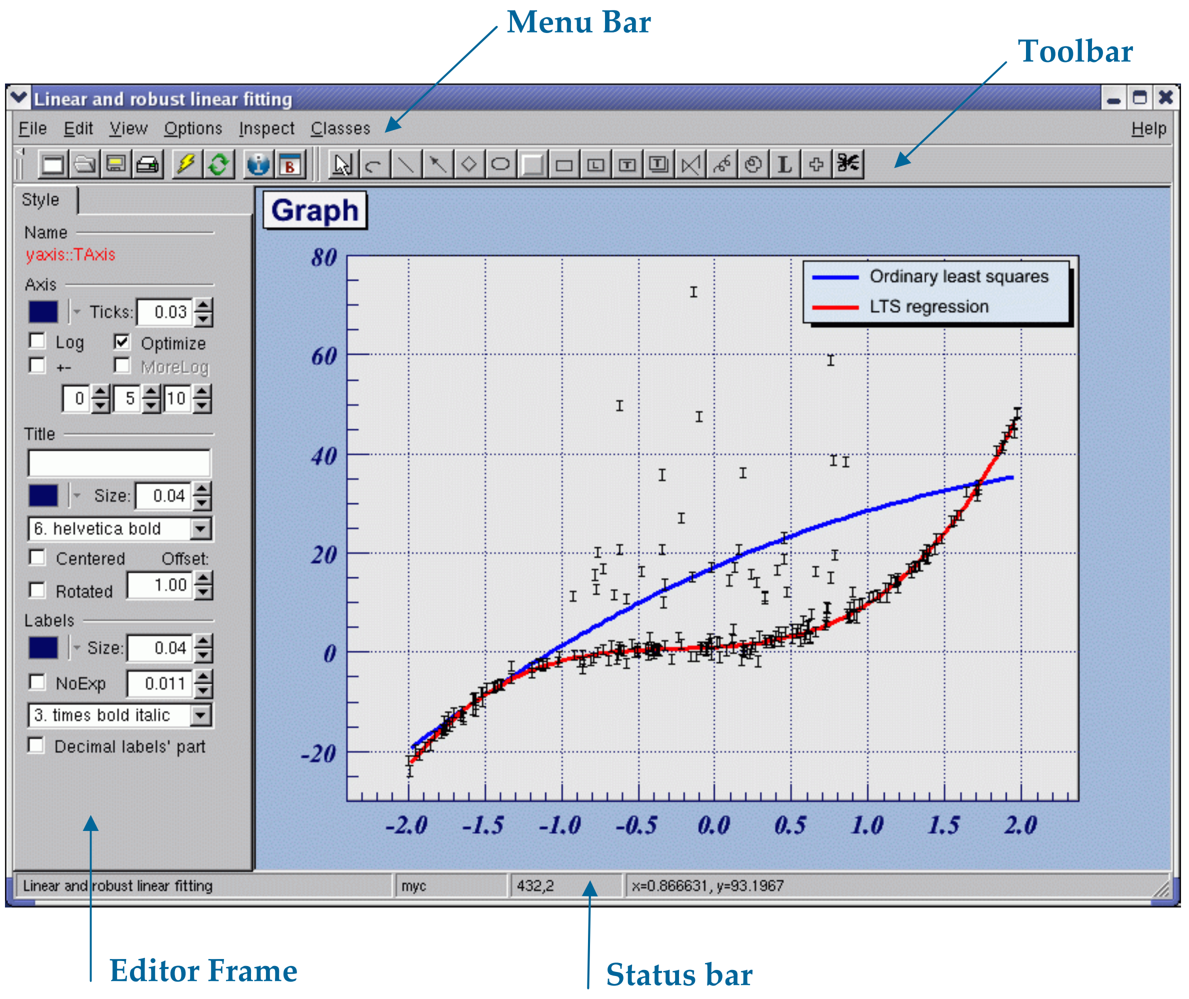
- checkboxes on the tree nodes turn on/off pieces of the tree hierarchy




ROOT Object Browser
File
View
Options
Help
Master Volume
Option
All Folders
Contents of "/Geometry/Master Volume"
Local transformations
Illegal overlaps
Tracks
Master Volume
Top Volume
IT12
I12B
I650
IT34
I004
I005
I047
I048
IT56
I570
I569
I571
I565
IS02
I098
I097
IS01
ITSD_1
0STL 0STR 0SUP ABSM ASSS B077 BBMO BFMO BRS1
BRS4 DCM0 DDIP EPM1 EPM2 EPM3 EPM4 FMD1 FMD2
FMD3 FMD4 FMD5 HBW1 HHC1 HHF1 HHF2 HHW1 HHW2
HHW3 HMBS HPIL HUFL HUWA ICYL ITSV L3DO L3DX
L3MO PCRA PHOS PRRD PWHE QBPM RICH S01M S02M
S03M S04M S07C S07G S07G S07G S07G S07G S07G
640 Objects.

GUI Classes (4)

- Canvas interface
  - Menus - restructured to better follow standard conventions; give access to new developed GUIs.
  - Tool bar is dockable and provides shortcuts for menu's and buttons for primitive drawing
  - Editor frame – provides GUIs for objects drawn in the canvas window

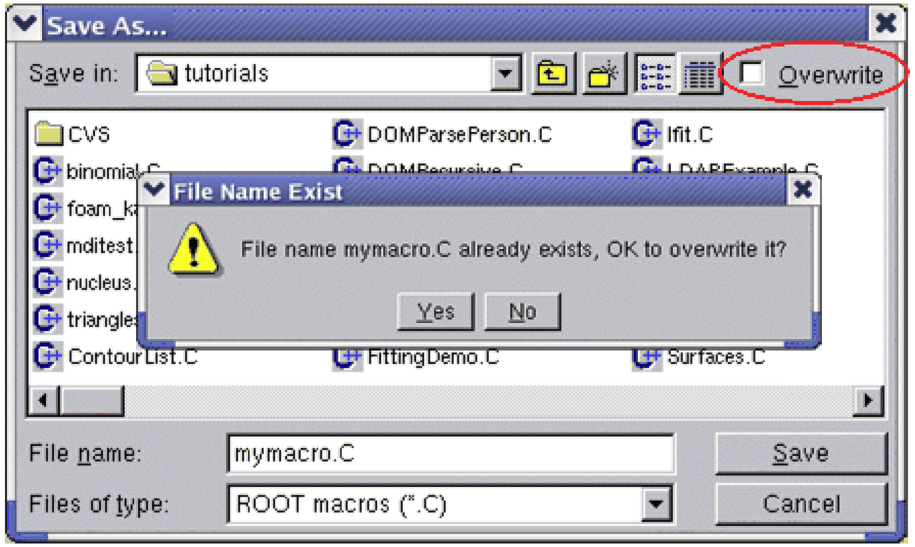


Style
Name
yaxis::TAxis
Axis
Ticks: 0.03
Log [✓ Optimize]
+- MoreLog
0 5 10
Title
Size: 0.04
6. helvetica bold
Centered Offset:
Rotated 1.00
Labels
Size: 0.04
NoExp 0.011
3. times bold italic
Decimal labels' part

Linear and robust linear fitting | myc | 432,2 | x=0.866631, y=93.1967

GUI Classes (5)

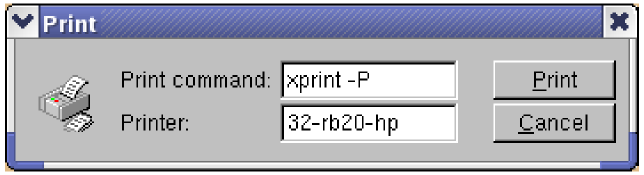
- SaveAs file dialog gives a choice for automatically overwriting existing files



File name mymacro.C already exists, OK to overwrite it?

File name: mymacro.C

Files of type: ROOT macros (*.C)



Print command: xprint -P

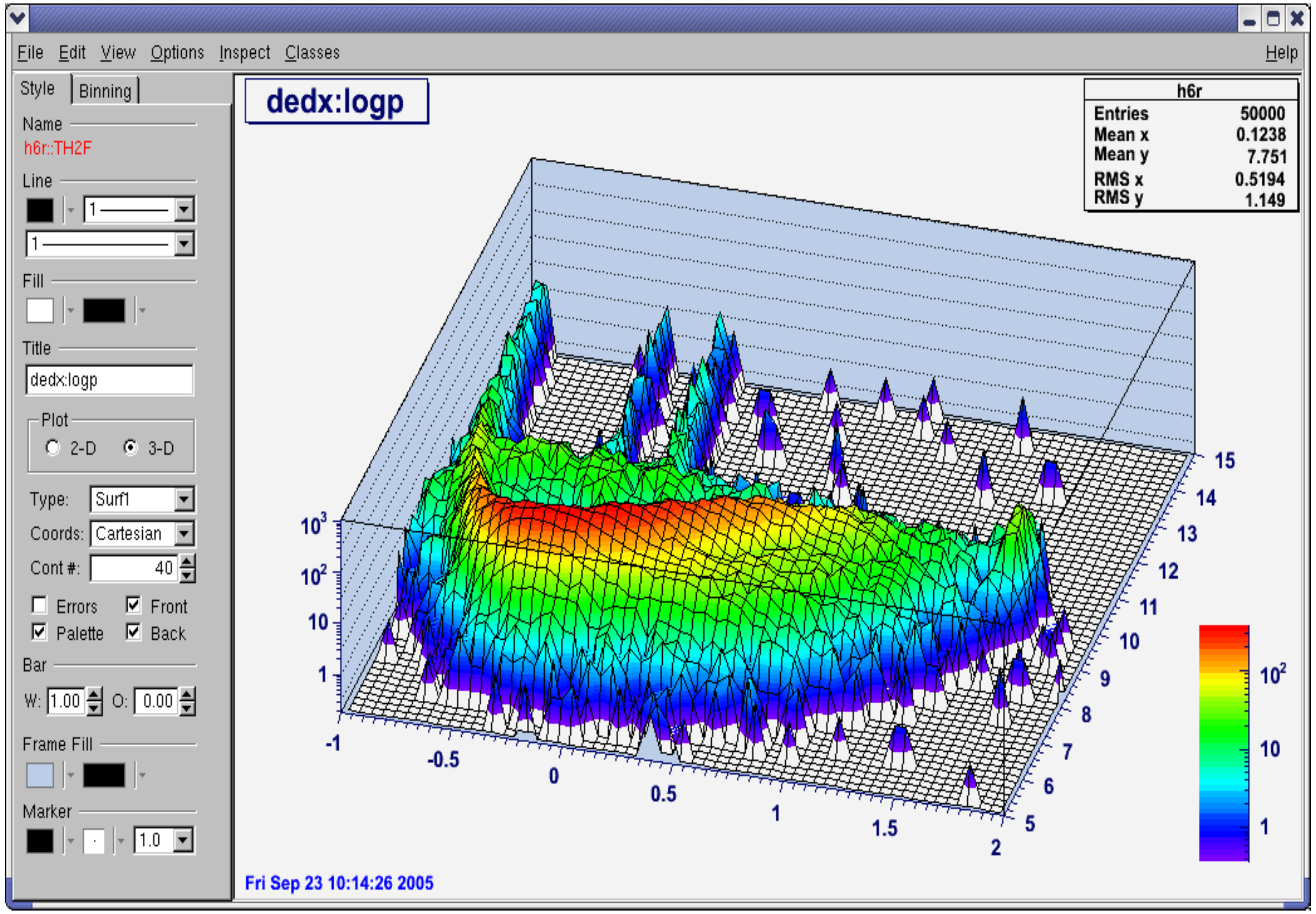
Printer: 32-rb20-hp

- Print command is enabled and pops-up a simple print dialog. Both parameters can be set via the new *Print.Command* and *Print.Printer* resources:

WinNT..Print.Command:*	*AcroRd32.exe*
Unix..Print.Command:*	*xprint -P%p %f*
Print.Printer:	*32-rb205-hp*
Print.Directory:	*.*

Graphics Editor (1)

- Object orientation of editor design
- Manage GUI complexity by object editors
- Presents the right GUI at the right time according to the selected object in the canvas
- Easy-to-use
- Capacity for growth



Name h6r::TH2F
Title: dedx:logp
Plot: ( ) 2-D [✓ 3-D]
Type: Surf1
Coords: Cartesian
Cont #: 40
☐ Errors ☑ Front
☑ Palette ☑ Back
W: 1.00 O: 0.00

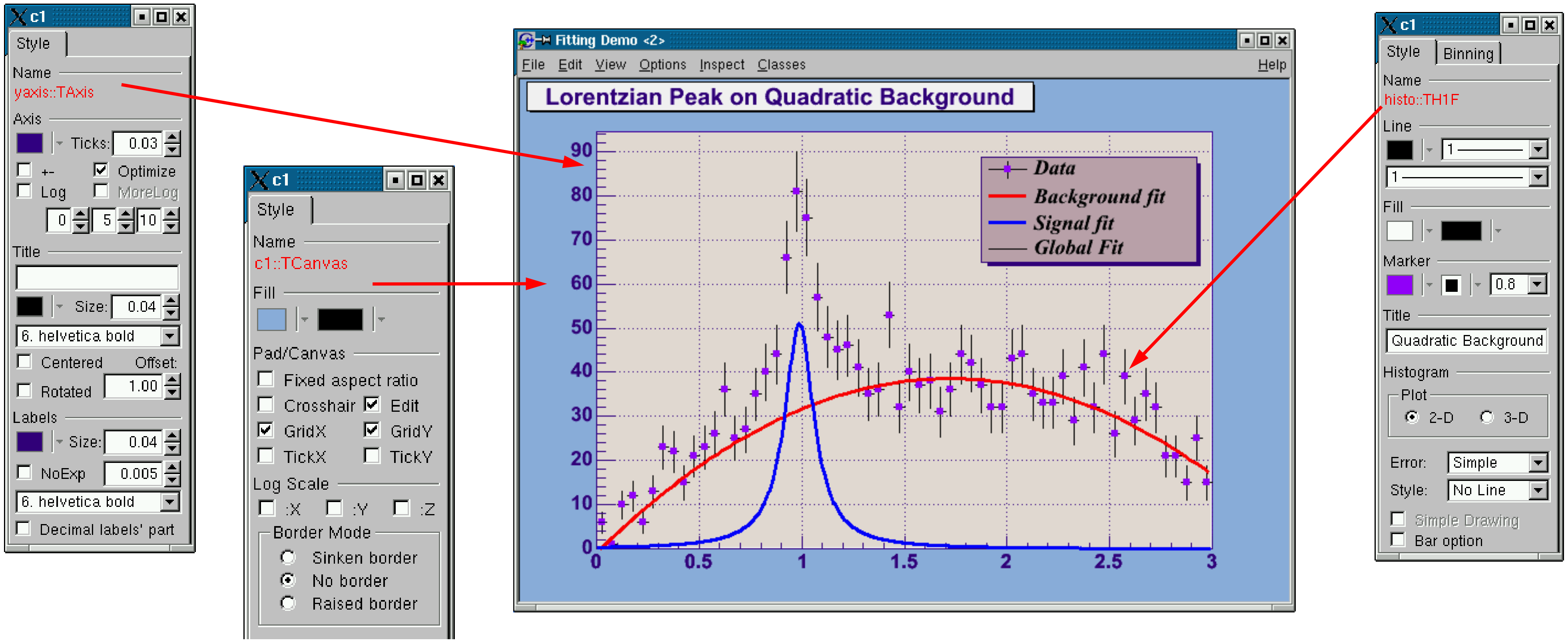
h6r	
Entries	50000
Mean x	0.1238
Mean y	7.751
RMS x	0.5194
RMS y	1.149

Fri Sep 23 10:14:26 2005

Graphics Editor (2)

Signals/Slots communication mechanism handles GUI actions:

- Canvas sends a signal identifying which object is selected
- Corresponding object editor is activated and ready for use



c1 – Style

Name: yaxis::TAxis

Axis: Ticks: 0.03; ☐ +-; ☑ Optimize; ☐ Log; ☐ MoreLog; 0 5 10

Title: Size: 0.04; 6. helvetica bold; ☐ Centered; Offset: 1.00; ☐ Rotated

Labels: Size: 0.04; ☐ NoExp 0.005; 6. helvetica bold; ☐ Decimal labels' part

c1 – Style

Name: c1::TCanvas

Fill

Pad/Canvas: ☐ Fixed aspect ratio; ☐ Crosshair; ☑ Edit; ☑ GridX; ☑ GridY; ☐ TickX; ☐ TickY

Log Scale: ☐ :X ☐ :Y ☐ :Z

Border Mode: Sinken border [✓ No border] Raised border

c1 – Style | Binning

Name: histo::TH1F

Line: 1; 1

Fill

Marker: 0.8

Title: Quadratic Background

Histogram – Plot: [✓ 2-D] 3-D

Error: Simple

Style: No Line

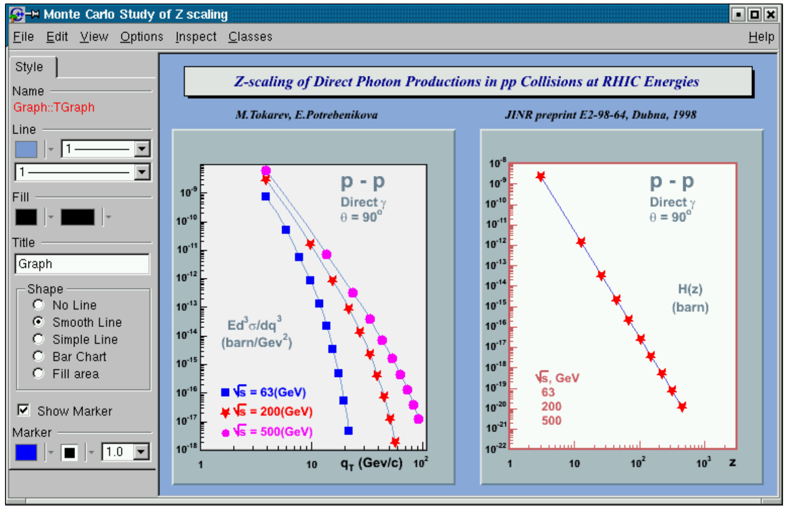
☐ Simple Drawing

☐ Bar option

Graphics Editor (3)

ROOT graphics editor can be:

- Embedded – connected only with the canvas in the application window



Monte Carlo Study of Z scaling

Style

Name: Graph::TGraph

Line

Fill

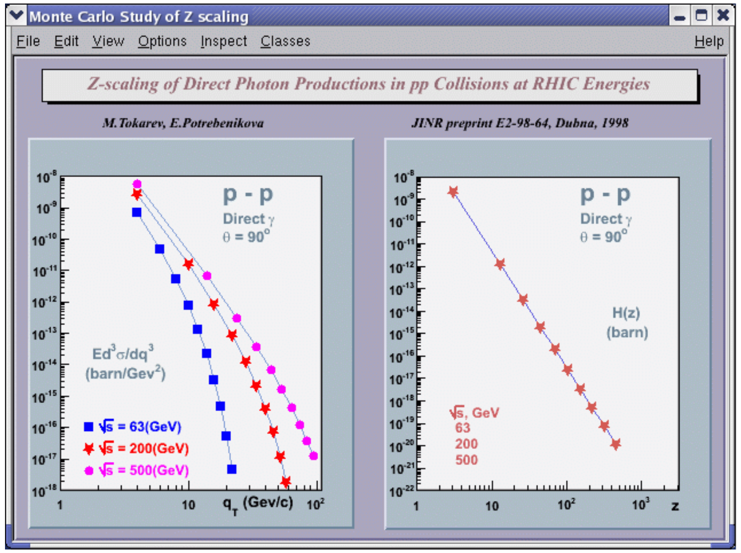
Title: Graph

Shape
- No Line
- [✓ Smooth Line]
- Simple Line
- Bar Chart
- Fill area

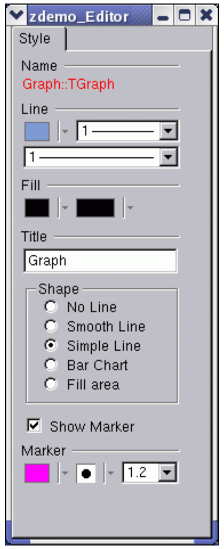
[✓ Show Marker]

Marker: 1.0

Z-scaling of Direct Photon Productions in pp Collisions at RHIC Energies
M.Tokarev, E.Potrebenikova
JINR preprint E2-98-64, Dubna, 1998
p - p
Direct γ
θ = 90°
Ed³σ/dq³ (barn/Gev²)
√s = 63(GeV)
√s = 200(GeV)
√s = 500(GeV)
q_T (Gev/c)
H(z) (barn)
√s, GeV
63
200
500
z



Monte Carlo Study of Z scaling



zdemo_Editor

Style

Name: Graph::TGraph

Line

Fill

Title: Graph

Shape
- No Line
- Smooth Line
- [✓ Simple Line]
- Bar Chart
- Fill area

[✓ Show Marker]

Marker: 1.2

- Global – has own application window and can be connected to any created canvas in a ROOT session.

Graphics Editor (4)

- Modular – it loads the corresponding object editor **objEditor** according to the selected object **obj** in the canvas respecting the class inheritance.

TArrow	TAttMarker	TCurlyArc	TH1	TPad
TAttFill	TAttText	TCurlyLine	TH2	TPaveStats
TAttLine	TAxis	TFrame	TGraph	. . .

- Algorithm:

  Search for a class name objEditor (correct naming is important).

  Check that this class derives TGedFrame (the editor base class).

  Make an instance of the object editor using TROOT::ProcessLine method.

  Scan all base classes for corresponding object editors.

Graphics Editor (5)

- Can be extended easily by any user-defined object editor - this makes GUI design easier and adaptive to the users' profiles.
- Rules to follow:

  Derive in the code from the base editor class TGedFrame.

  Correct naming convention: the name of the object editor should be the object class name + 'Editor'.

  Register the new object editor in the list TClass::fClassEditors at the end of its constructor.

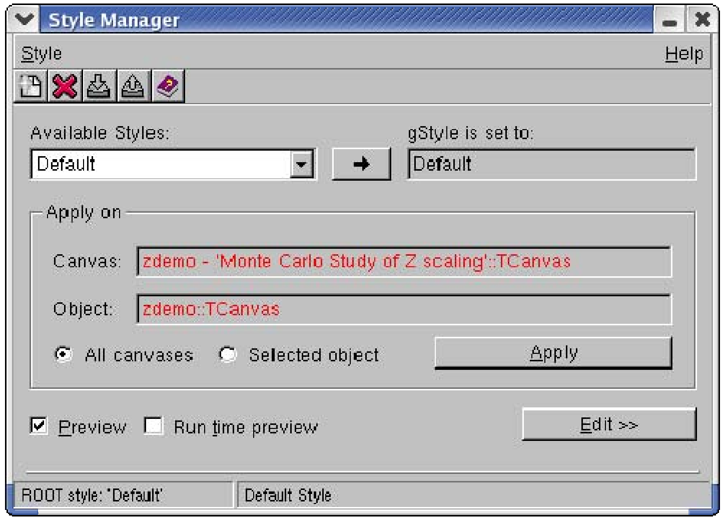
  Use signals/slots communication mechanism for event processing.

  Implement SetModel method to set GUI widgets according to the object's attributes.

  Implement all necessary slots & connect them to appropriate widget signals.

Style Manager (1)

- Top level interface
  - Manage a collection of TStyle objects
  - Create a new style
  - Delete a selected style
  - Import from a canvas / a C++ macro
  - Export to a C++ macro
  - Apply on all canvases or a selected object
  - Activate the style editor
- Preview window
  - Show the predicted results
  - On line update or by request
  - Placed in front of the selected canvas
- Style Editor



Available Styles: Default

gStyle is set to: Default

Apply on

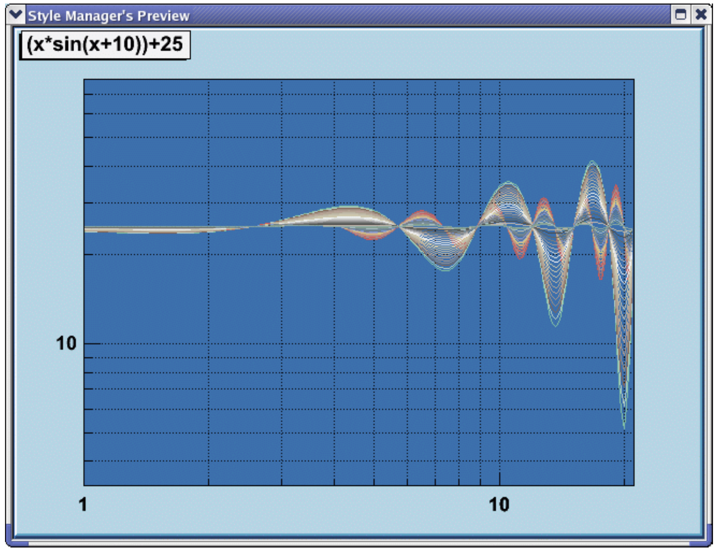
Canvas: zdemo - 'Monte Carlo Study of Z scaling'::TCanvas

Object: zdemo::TCanvas

[✓ All canvases] Selected object

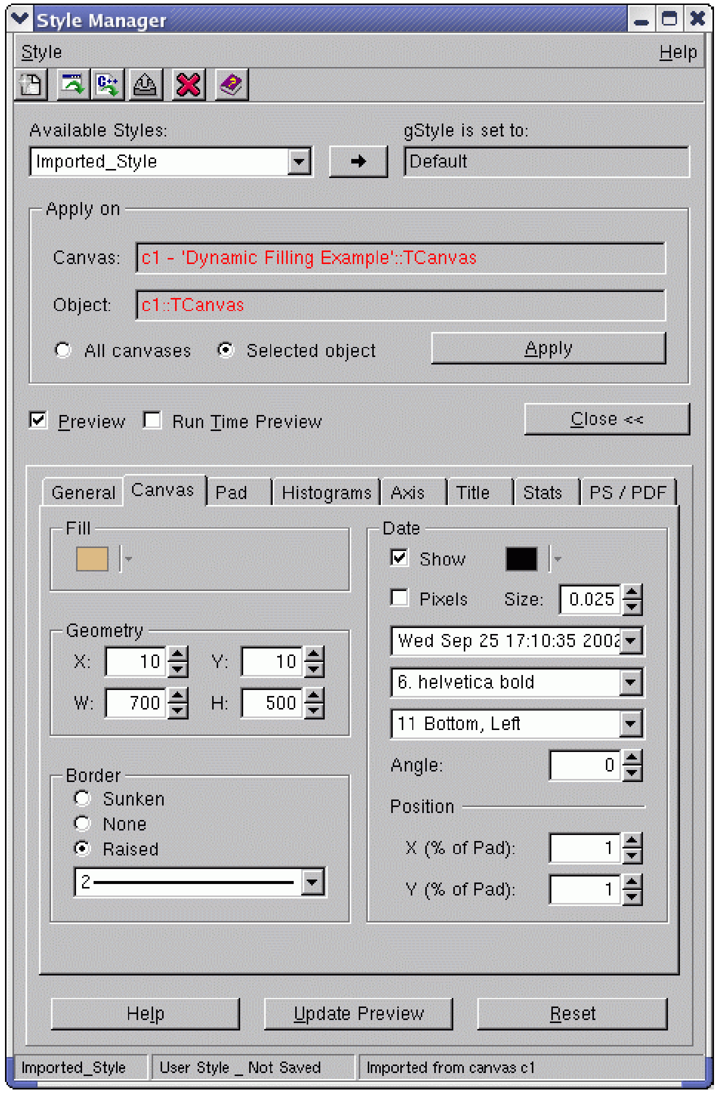
[✓ Preview] Run time preview

ROOT style: 'Default' | Default Style



Style Manager (2)

- To edit the selected TStyle object
  - Every data member can be edited
  - Protect users from errors– they can go back to a previous saved state easily
  - Update the Preview by request
  - Help
- Only information relative to the current task is presented; other GUI parts are hidden.
- Full and continuous feedback on the action result.
- GUI elements are grouped according to the task flow.



Available Styles: Imported_Style
gStyle is set to: Default

Apply on
Canvas: c1 - 'Dynamic Filling Example'::TCanvas
Object: c1::TCanvas
All canvases [✓ Selected object]

[✓ Preview] Run Time Preview

Fill

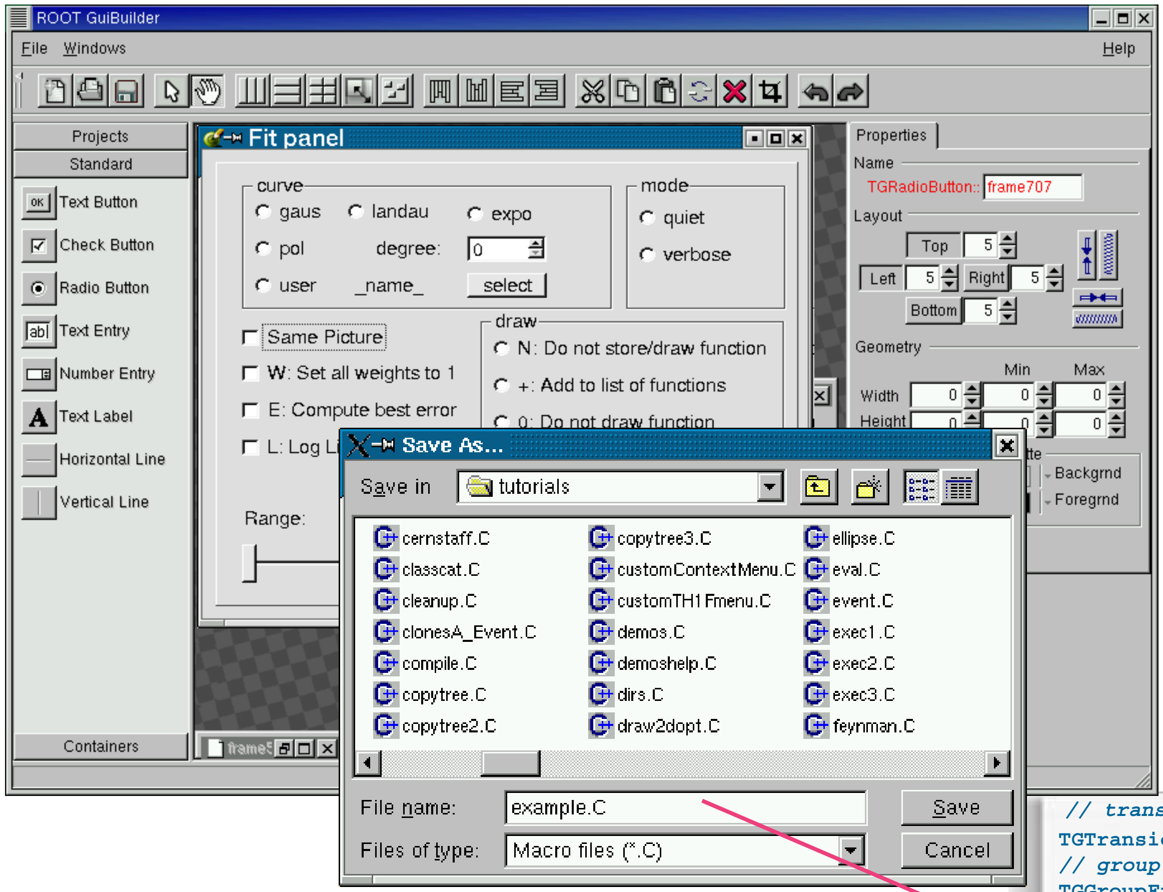
Geometry
X: 10 Y: 10
W: 700 H: 500

Border
Sunken
None
[✓ Raised]
2

Date
[✓ Show]
Pixels Size: 0.025
Wed Sep 25 17:10:35 2002
6. helvetica bold
11 Bottom, Left
Angle: 0
Position
X (% of Pad): 1
Y (% of Pad): 1

Imported_Style | User Style _ Not Saved | Imported from canvas c1

GUI Builder (1)



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

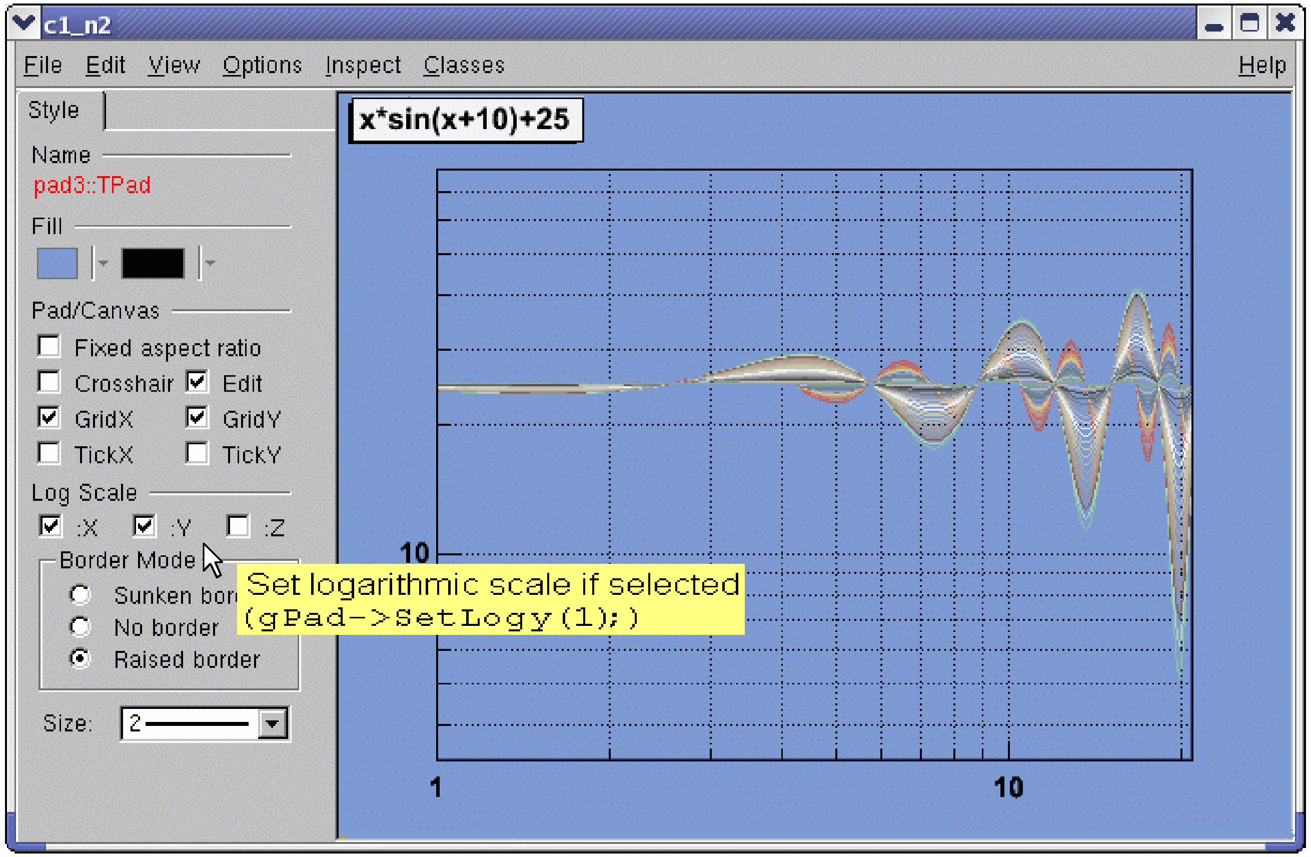
- To complete the GUI widget palette with combo/list boxes, double sliders, list view, list tree, shutters, button group, etc.
- To develop tools for signals/slots communication mechanism.
- To provide examples for several basic types of GUIs (as tutorials)

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: to create instances of so-called command objects for all editing actions.
- 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.

Next Steps (1)

- New object editors
- Undo/Redo tools
- Fit Panel
- New GUI widgets
- GUI Builder
- ROOT commands in tool tips
- Help
- GUI Tutorials
- Documentation

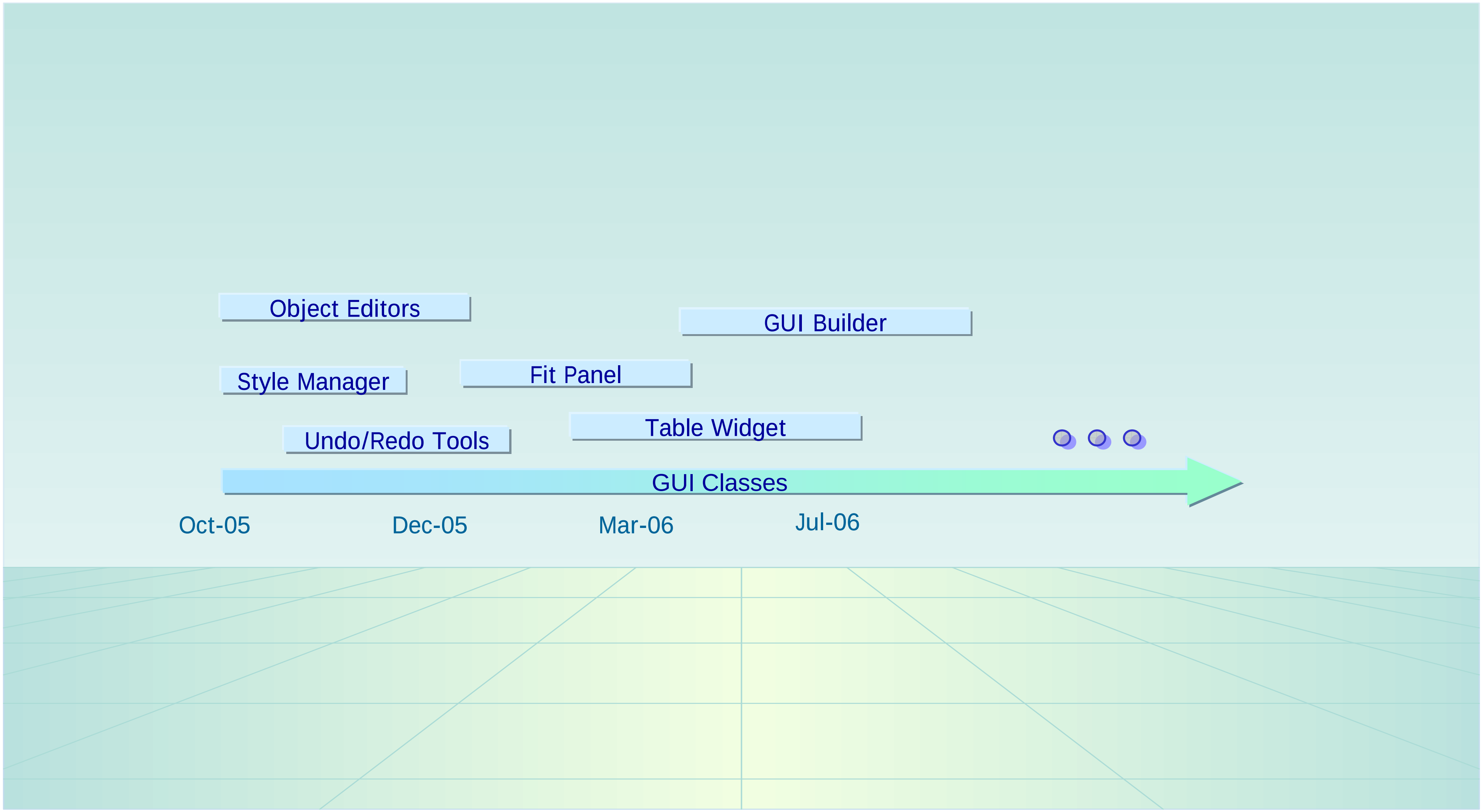


Name
pad3::TPad
Fill
Pad/Canvas
☐ Fixed aspect ratio
☐ Crosshair ☑ Edit
☑ GridX ☑ GridY
☐ TickX ☐ TickY
Log Scale
☑ :X ☑ :Y ☐ :Z
Border Mode
○ Sunken bor
○ No border
● Raised border
Size: 2

Set logarithmic scale if selected
(gPad->SetLogy(1);)

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

Next Steps (2)


Object Editors
GUI Builder
Style Manager
Fit Panel
Table Widget
Undo/Redo Tools
GUI Classes
Oct-05
Dec-05
Mar-06
Jul-06

Thank you!