ROOT GUI Builder

Status & Plans

ROOT & External GUI World

MFC, FOX, Qt, PVSS…

Snapshot of the Future

Shaped Windows

Drag & Drop

Ilka Antcheva, Bertrand Bellenot, Valeriy Onuchin

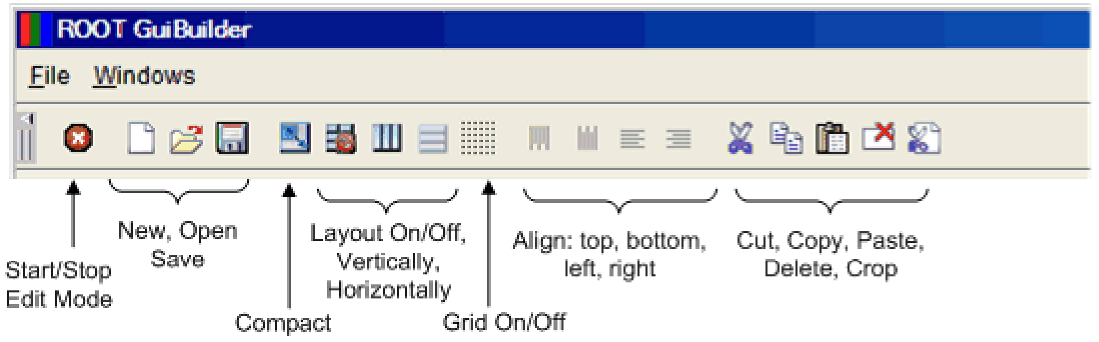
GUI Builder
on behalf of Valeriy Onuchin

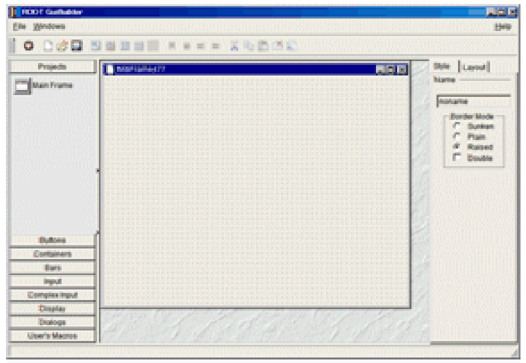
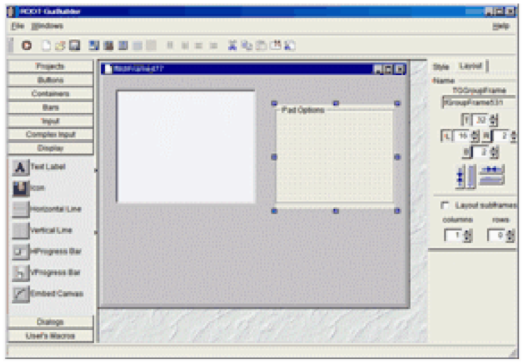
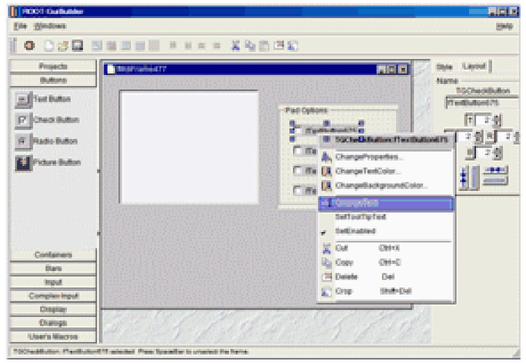
- The ROOT GUI builder provides graphical user-interface (GUI) tools for developing user interfaces based on the ROOT GUI classes. It includes over 30 advanced widgets and has as a goal to offer the complete set of features needed for developing the GUI projects. It gives developers an effective object-oriented development environment and accelerates the GUI development life cycles. The new version of the GUI builder has a new look and better appearance.

Improved Functionality

- New functionality added via menus and the tool bar.
- Any created user interface can be saved in a macro.
- Any ROOT GUI application can be switched to editable mode – already exiting user interface objects, container frames can be just drag-and-drop, move, resize, copy, paste, change re-parent, etc.
- Drag/Drop of selected frames.
- Copy/Paste/Cut of selected widget.
- Open/Save of the designed GUI in a macro.
- Easy layout settings and widget alignment.
- Improved keyboard navigation.
- Widget's attributes can be changed via dynamically created context menus or activated edit dialogs.
- Many new widgets were added to the widgets' palette (see slide #6)

Quick Start

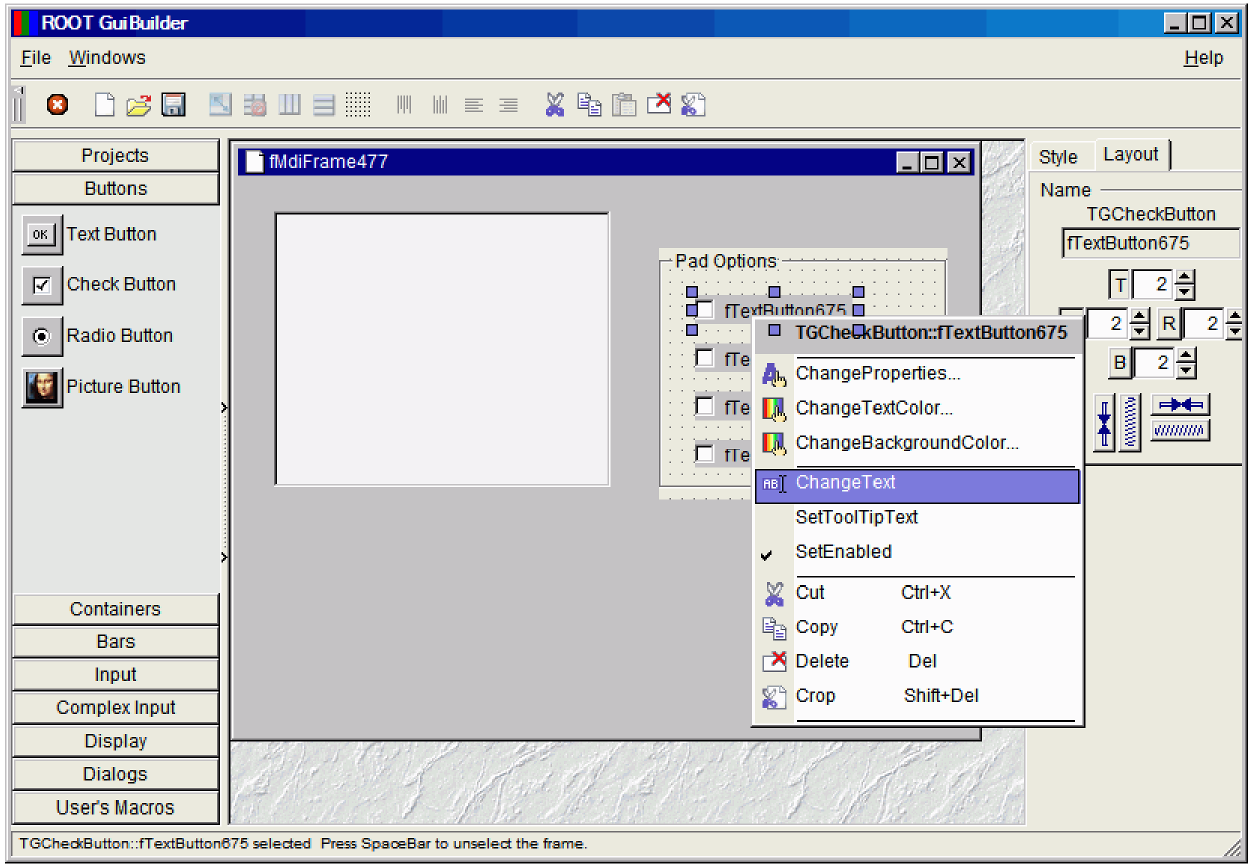

ROOT GuiBuilder
File Windows
Start/Stop Edit Mode
New, Open Save
Compact
Layout On/Off, Vertically, Horizontally
Grid On/Off
Align: top, bottom, left, right
Cut, Copy, Paste, Delete, Crop

- Steps of creating the interface:
- Click on Project button in the Widget's Palette (the first icon). By this, a window is created which will form the main window of your GUI.
- Turn the edit mode ON for that window.
- Add different widgets to your main frame: text entry, buttons, list box, etc.

Editing

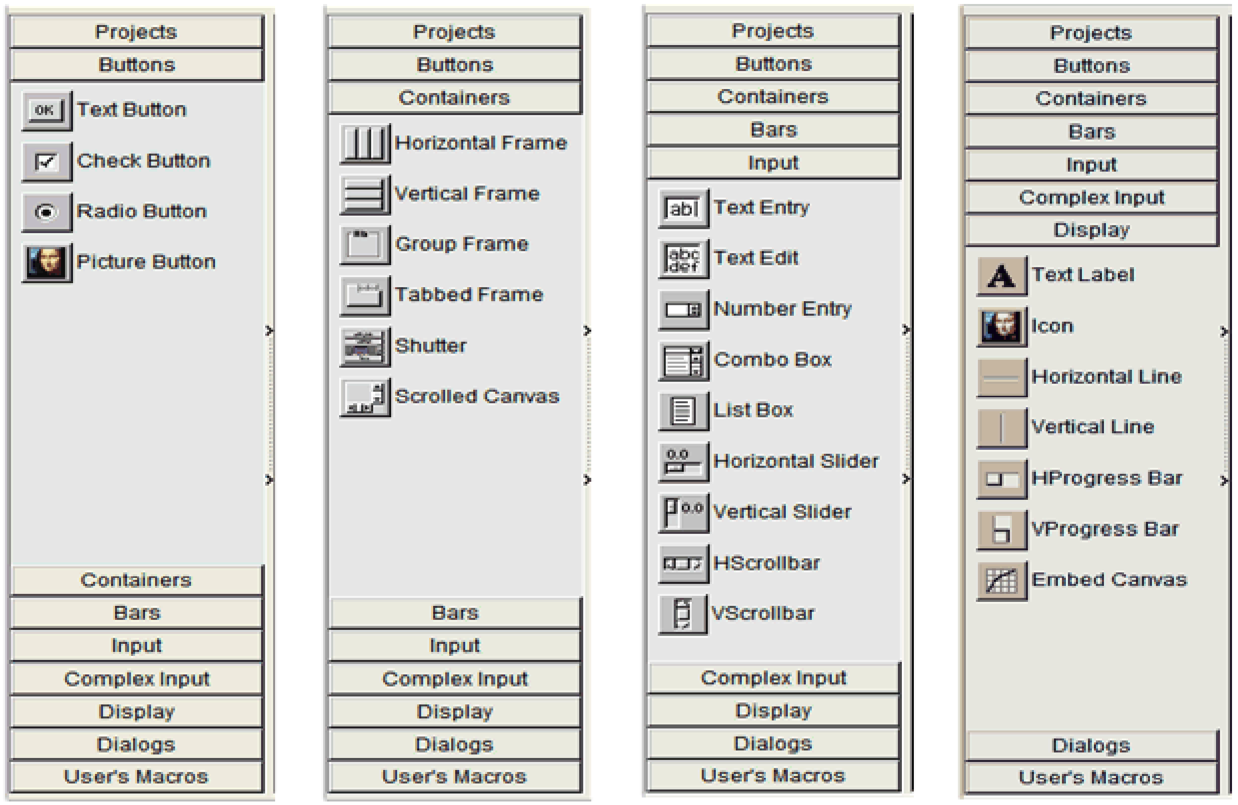
- The user interface elements can be selected, positioned, and grouped, laid out in the main application frame. According to the selected widget, a dynamically created context menu provides detailed control of widget attributes' settings.



TGCheckButton::fTextButton675 selected Press SpaceBar to unselect the frame.

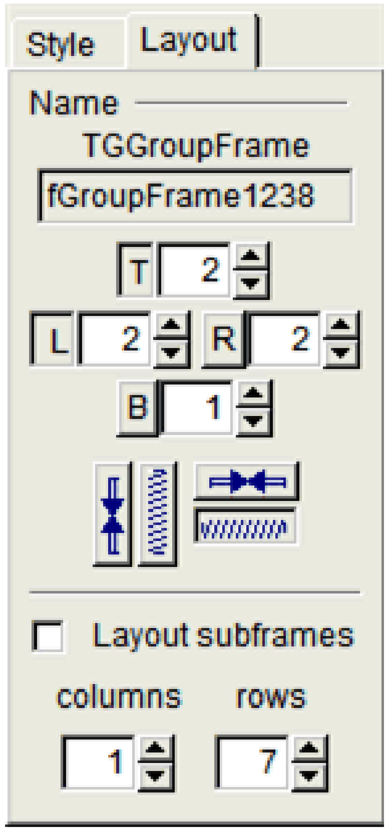
ROOT
An Object-Oriented Data Analysis Framework

Snapshot of Available Widgets



Layout

- The frame layout can be changed by using Property editor on the right.
- Once the frame is successfully created, the designed GUI can be saved in a macro.
- The generated macro can be opened in the GUI builder and additional widgets can be added to the previously saved main frame. For that you do not need to edit the code using an external text editor - just create your new design and save it again in a macro. This macro can be run separately in a ROOT session.


Style | Layout

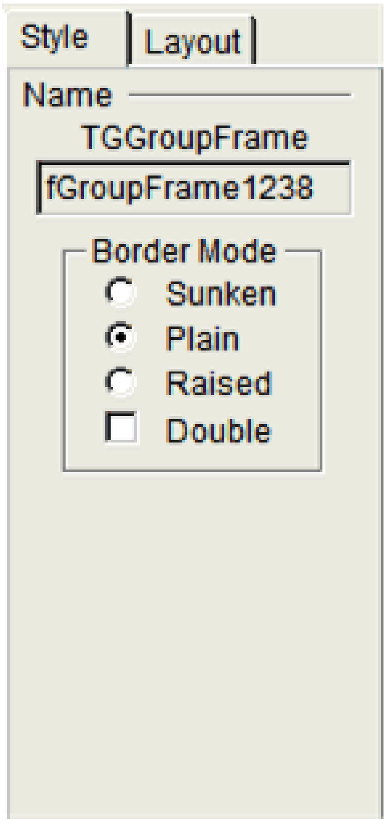
Name: TGGroupFrame

fGroupFrame1238

T: 2
L: 2
R: 2
B: 1

[ ] Layout subframes

columns: 1
rows: 7


Style | Layout

Name: TGGroupFrame

fGroupFrame1238

Border Mode
Sunken [✓ Plain] Raised
[ ] Double

Summary

- One major feature, allowing a rapid development, is the possibility to save the actual GUI in a macro, to execute it in a ROOT session, reopen it (or continue editing) in the GUI Builder, and so on. This allows to have an instantaneous view of the GUI and its behavior, while editing it.
- Another powerful feature is the possibility to edit macros generated by pressing CTRL+S in any running GUI application, thanks to SavePrimitive, a very useful method implemented in every GUI class (thanks to Ilka Antcheva).

Next Steps

- **Validation tests** and **robustness** of the **GUI builder functionality**
- Documentation – Help and HowTo
- Complete the GUI builder with the rest of the ROOT widgets
- Undo/Redo
- Adding ToolTips
- Property editors development
- Improvements of the GUI builder ‘look and feel’

ROOT & External GUI World

MFC, FOX, Qt, PVSS…

ROOT +

Microsoft Visual Studio.net

FOX TOOLKIT

PVSS II by ETM Solutions for process control engineering

Features

- Allows to embed a TCanvas into any external application or any toolkit, using only a few lines of code, as soon as it is possible to:
    - Obtain a Window ID (XID on X11, HWND on Win32).
    - Create a timer to handle Root events.
    - Forward (i.e. mouse) events to the Canvas

How it Works

- In the application constructor or in main(), create a TApplication:

```
gMyRootApp = new TApplication("My ROOT Application", &argc, argv);
gMyRootApp->SetReturnFromRun(true);
```

- Create a timer to process Root events :

```
void MyWindow::OnRefreshTimer() {
    gApplication->StartIdleing();
    gSystem->InnerLoop();
    gApplication->StopIdleing();
}
```

- Get the id of the window where you want to embed the TCanvas:

```
void MyWindow::Create()  {
    int wid = gVirtualX->AddWindow((ULong_t)getid(), getWidth(), getHeight());
    fCanvas = new TCanvas("fCanvas", getWidth(), getHeight(), wid);
}
```

- Forward messages to the Canvas. i.e:

```
void MyWindow::OnPaint() {
    if (fCanvas) fCanvas->Update();
}
void MyWindow::OnLMouseDown() {
    if (fCanvas) fCanvas->HandleInput(kButton1Down, ev->x, ev->y);
}
```

Example with MFC (1)

- Application and Timer Creation:

```
CMFCRootApp::CMFCRootApp()
{
    int argc = 1;
    char *argv[] = { "MFCRootApp.exe", NULL };
    gMFCRootApp = new TApplication("MFC ROOT Application", &argc, argv);
    gMFCRootApp->SetReturnFromRun(true);
}

VOID CALLBACK MyTimerProc(HWND hwnd, UINT message, UINT idTimer, DWORD dwTime)
{
    gApplication->StartIdleing();
    gSystem->InnerLoop();
    gApplication->StopIdleing();
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
      return -1;
…
   SetTimer(1, 20, (TIMERPROC) MyTimerProc); // timer callback
   return 0;
}
```

Example with MFC (2)

- Canvas creation and event forwarding:

```
void CChildView::OnPaint()
{
   CPaintDC dc(this); // device context for painting

   RECT rect;
   if (fCanvas == 0) {
      GetWindowRect(&rect);
      int width  = rect.right-rect.left;
      int height = rect.bottom-rect.top;
      int wid = gVirtualX->AddWindow((ULong_t)m_hWnd, width, height);
      fCanvas = new TCanvas("fCanvas", width, height, wid);
   } else fCanvas->Update();
}

void CChildView::OnLButtonUp(UINT nFlags, CPoint point)
{
   if (fCanvas)
      fCanvas->HandleInput(kButton1Up, point.x, point.y);
   CWnd::OnLButtonUp(nFlags, point);
}
```

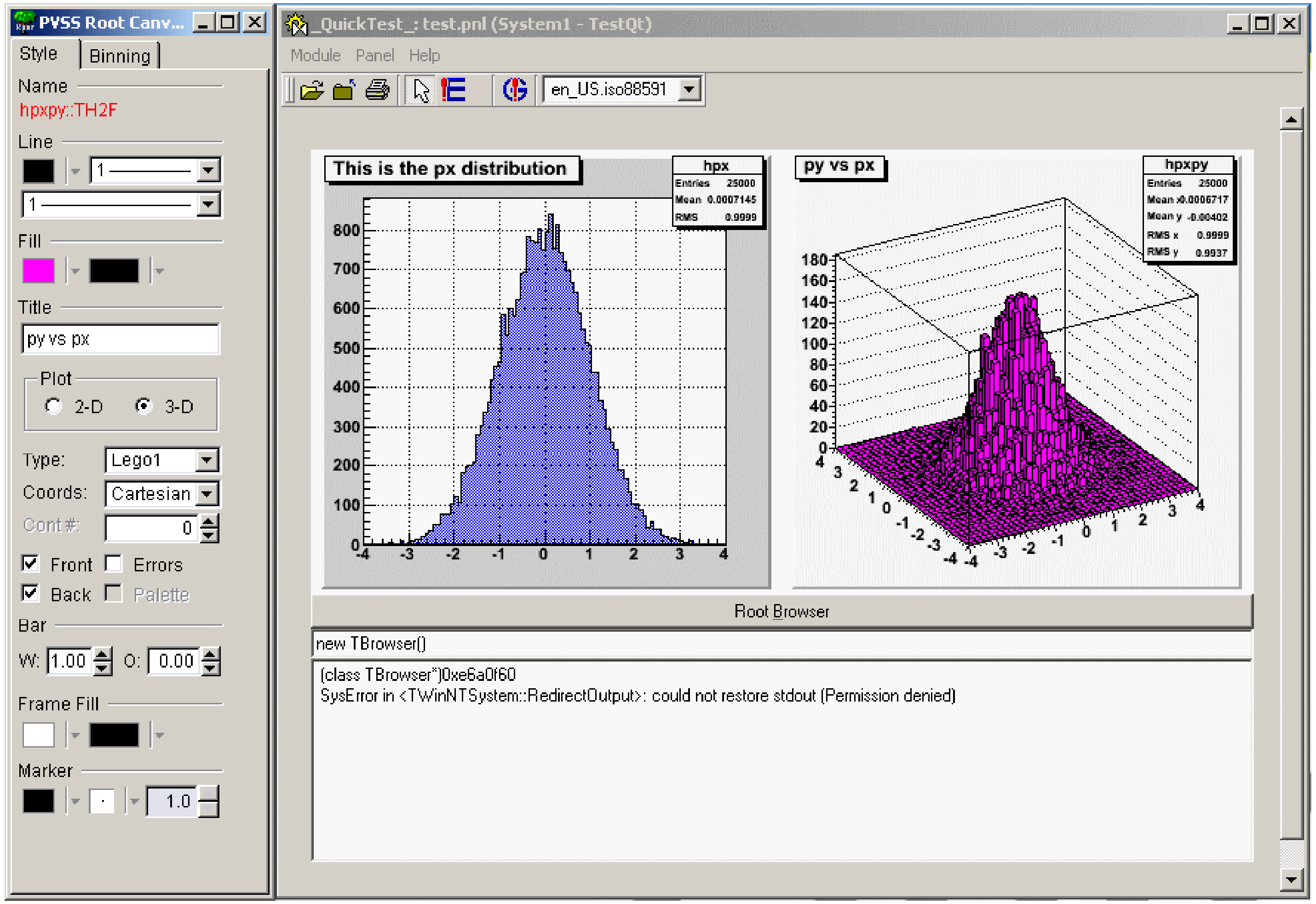
ROOT and PVSS

ROOT & PVSS

Screenshot

(in collaboration

with Piotr Golonka)



Name
hpxpy::TH2F

Line
1
1

Fill

Title
py vs px

Plot
2-D [✓ 3-D]

Type: Lego1
Coords: Cartesian
Cont #: 0

[✓ Front] Errors
[✓ Back] Palette

Bar
W: 1.00 O: 0.00

Frame Fill

Marker
1.0

en_US.iso88591

hpx: Entries 25000, Mean 0.0007145, RMS 0.9999

hpxpy: Entries 25000, Mean x 0.0006717, Mean y -0.00402, RMS x 0.9999, RMS y 0.9937

```
new TBrowser()
```

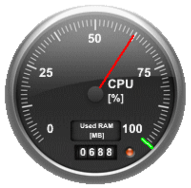
```
(class TBrowser*)0xe6a0f60
SysError in <TWinNTSystem::RedirectOutput>: could not restore stdout (Permission denied)
```

ROOT
TH2F Editor

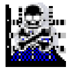
PVSS Panel

Snapshot of the Future

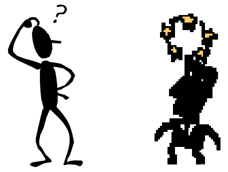
Custom shapes...



Drag...



And Drop...



Shaped Windows

- The nonrectangular window shape extension adds nonrectangular windows to the system. This allows for making shaped (partially transparent) windows.
- Already working since ROOT version 5.11.04

Very simple to implement, only a few steps needed:

- First, get the picture used as shape and background:

```
fBgnd = fClient->GetPicture("pictures/Main.xpm");
```

- Then call the magic function:

```
gVirtualX->ShapeCombineMask(fId, 0, 0, fBgnd->GetMask());
```

- Set the background picture:

```
SetBackgroundPixmap(fBgnd->GetPicture());
```

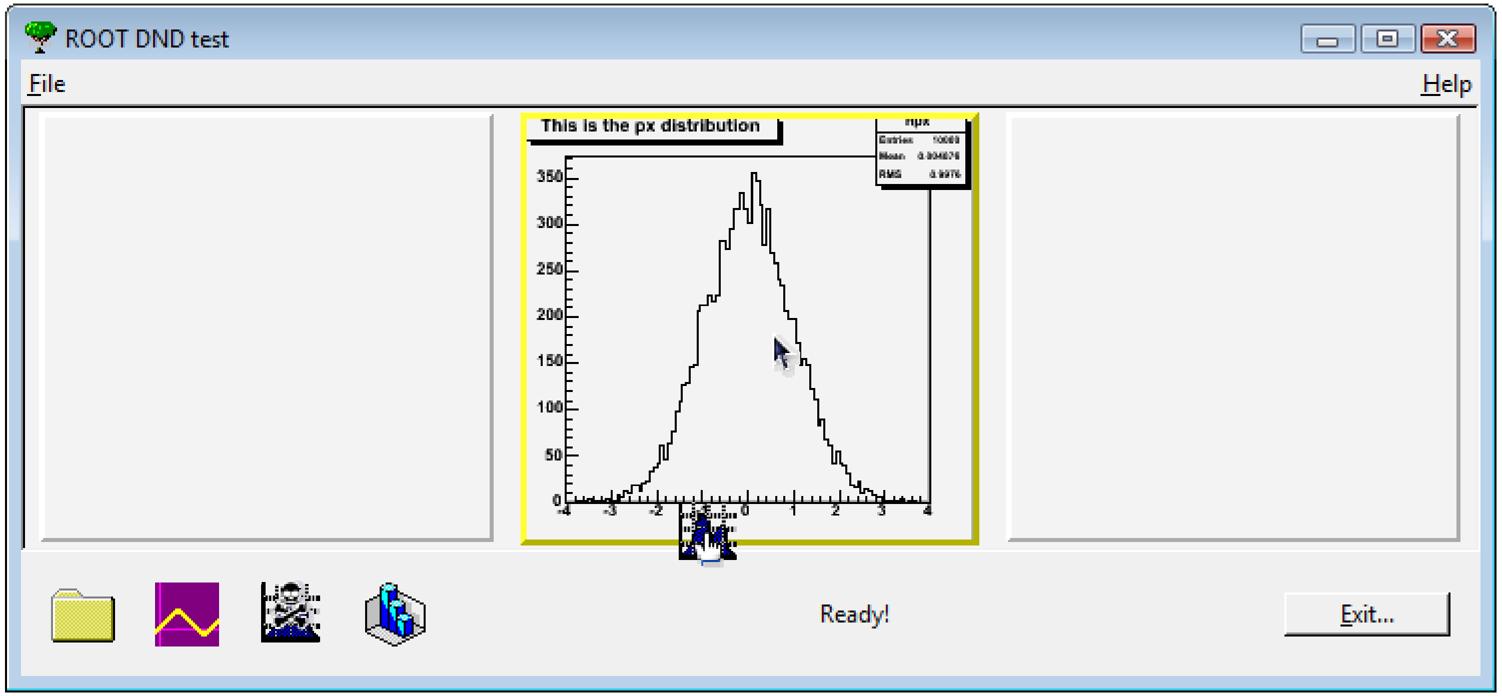
- Finally, as the picture cannot be resized, fix the size of the window:

```
SetWMSizeHints(fBgnd->GetWidth(), fBgnd->GetHeight(), fBgnd->GetWidth(), fBgnd->GetHeight(), 1, 1);
```

Drag & Drop

Features: Allows to drag and drop any ROOT object (i.e. from TBrowser to TCanvas)

First prototype is working on Windows (within the same ROOT application) and on Linux (also between different ROOT applications)


ROOT DND test
File
Help
This is the px distribution
hpx
Entries 10000
Mean 0.004076
RMS 0.9976
Ready!
Exit...