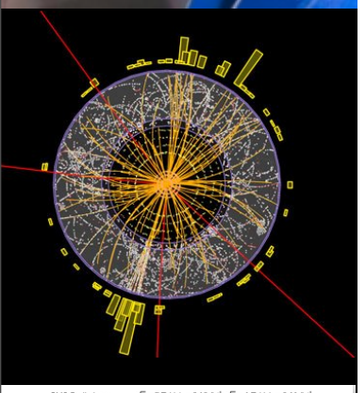
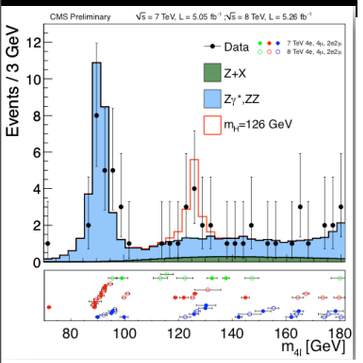
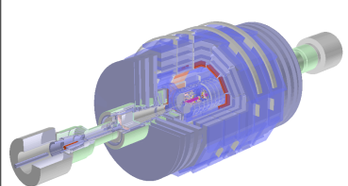






Native graphics on Mac OS X

ROOT on iOS and RootBrowser app

Timur Pocheptsov
root.cern.ch

Motivation

- X11.application (XQuartz) is not required anymore.
- No mysterious crashes with XQuartz (we can not fix XQuartz). But we can fix bugs in a ROOT's code.
- We have an access to the native modern graphics on Mac, not limited by outdated X11 (at the moment new features we have: transparency, anti-aliasing, gradient fills and shadows)
- We can mix native and ROOT's widgets in a ROOT GUI application.

Timeline

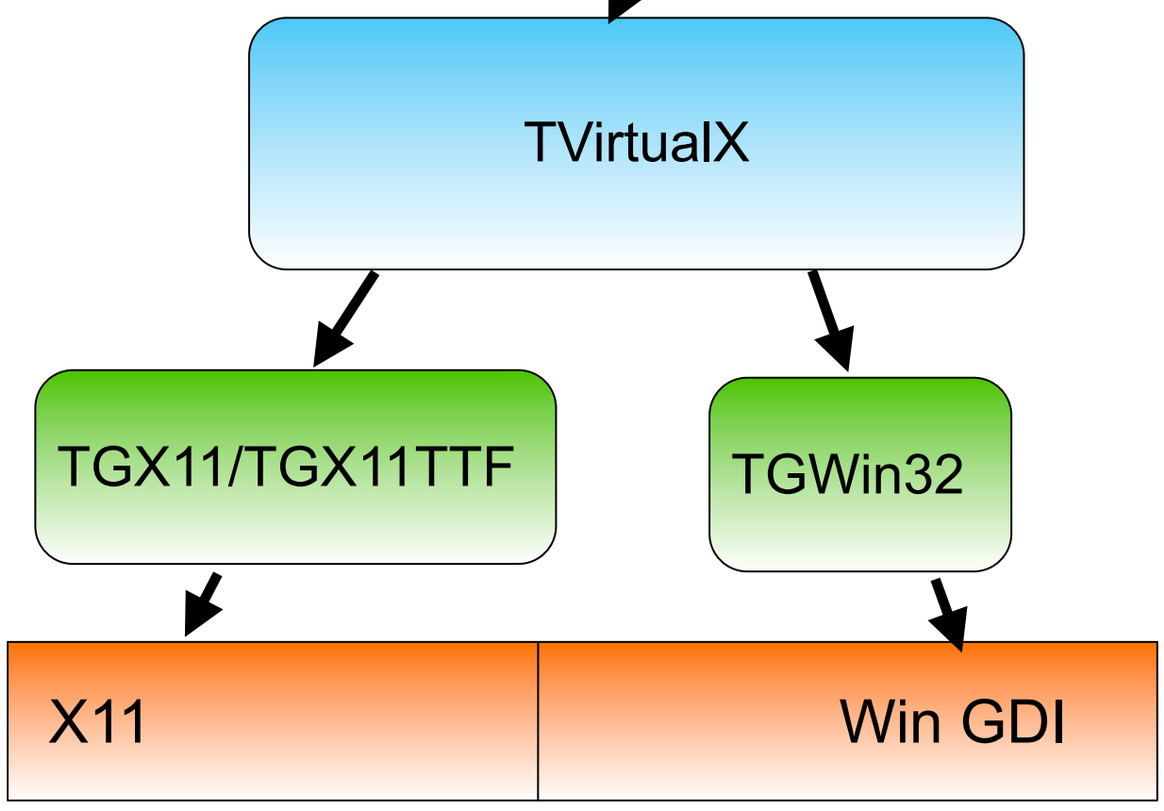
- Project started in December 2011
- The first ‘alpha-version’ (without OpenGL support) was available April 2012 (the first of 5.34 releases)
- May 2012 - OpenGL
- Became more stable and robust by September 2012
- Support for new Retina devices - October 2012
- pyROOT - February 2013.

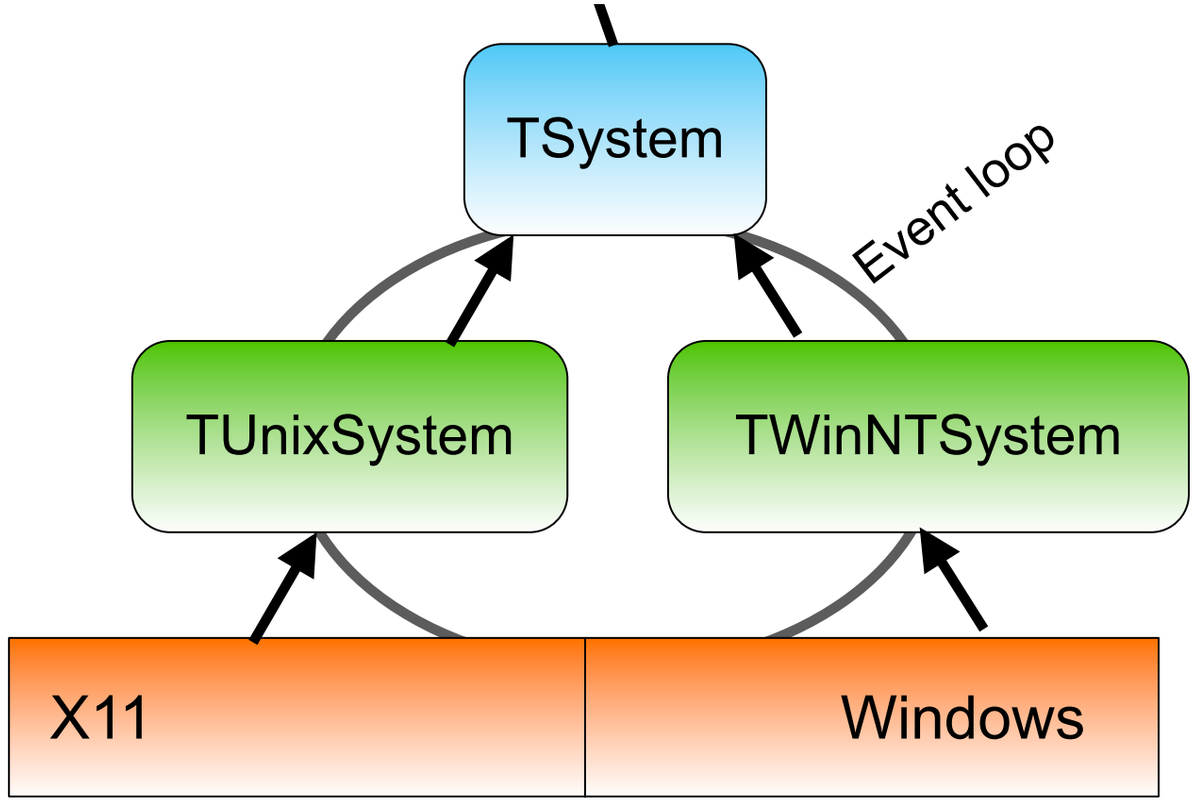
ROOT's graphics:

ROOT's GUI Framework
TGWindow, TGFrame, TGMainFrame, ... TGButton ...
TGSlider ... TGEtc.

Window management, graphics, images, etc.

Events (mouse, keyboard, geometry changes, etc.)



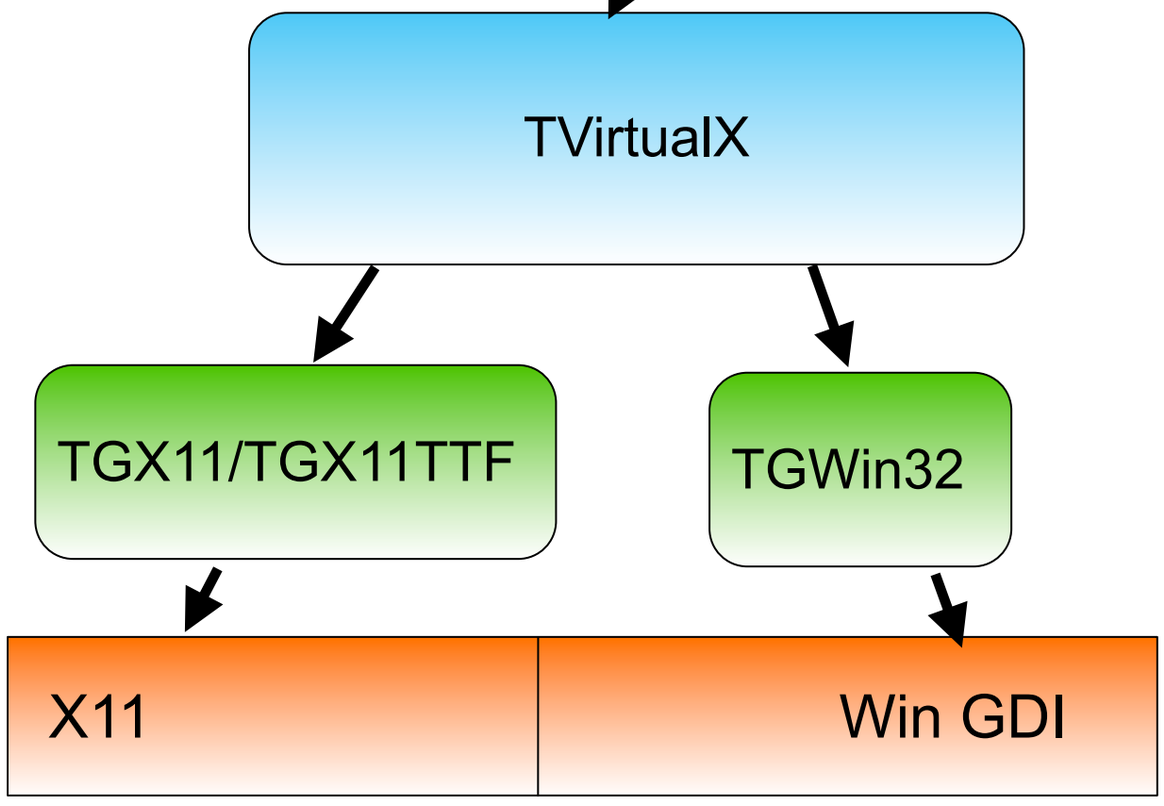


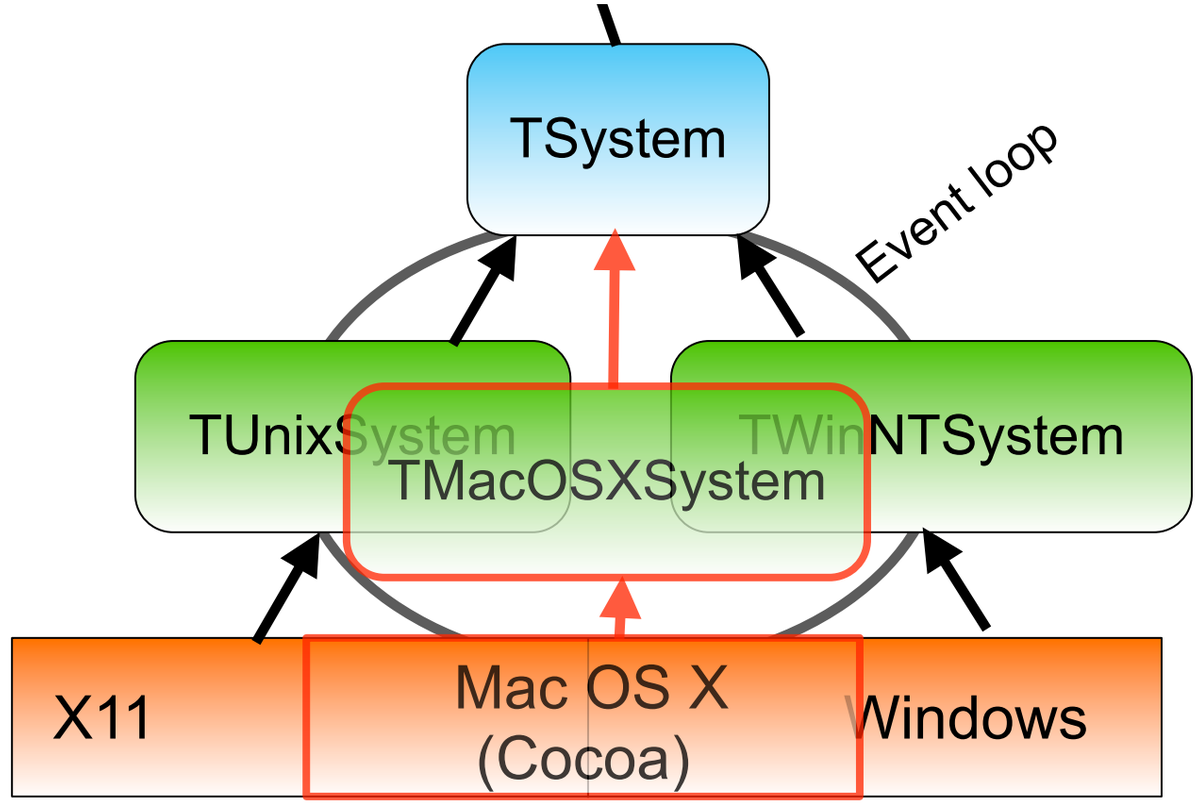
ROOT's graphics:

ROOT's GUI Framework
TGWindow, TGFrame, TGMainFrame, ... TGButton ...
TGSlider ... TGEtc.

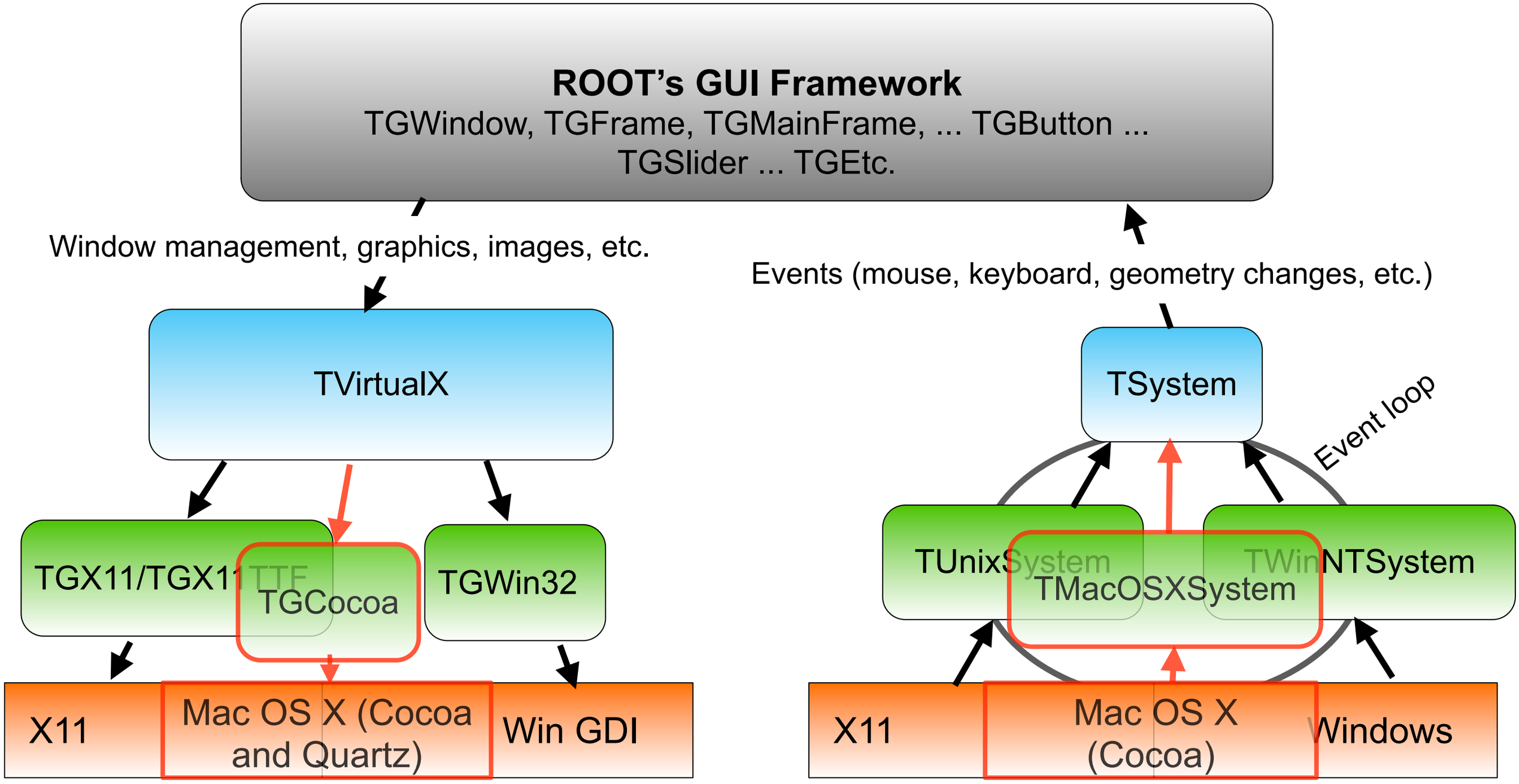
Window management, graphics, images, etc.

Events (mouse, keyboard, geometry changes, etc.)





ROOT's graphics:



Implementation: TVirtualX

- TVirtualX - abstract base class, includes functions for:
  - Window management
  - Pixmap and image management
  - Font management
  - Cursors
  - Drag and drop
  - OpenGL (window/context management)

  (Cocoa)
  - Graphical context management

  Emulation
  - Drawing (GUI)
  - Drawing (non-GUI - for TPad)

  (Quartz 2D)

Cocoa, Objective-C

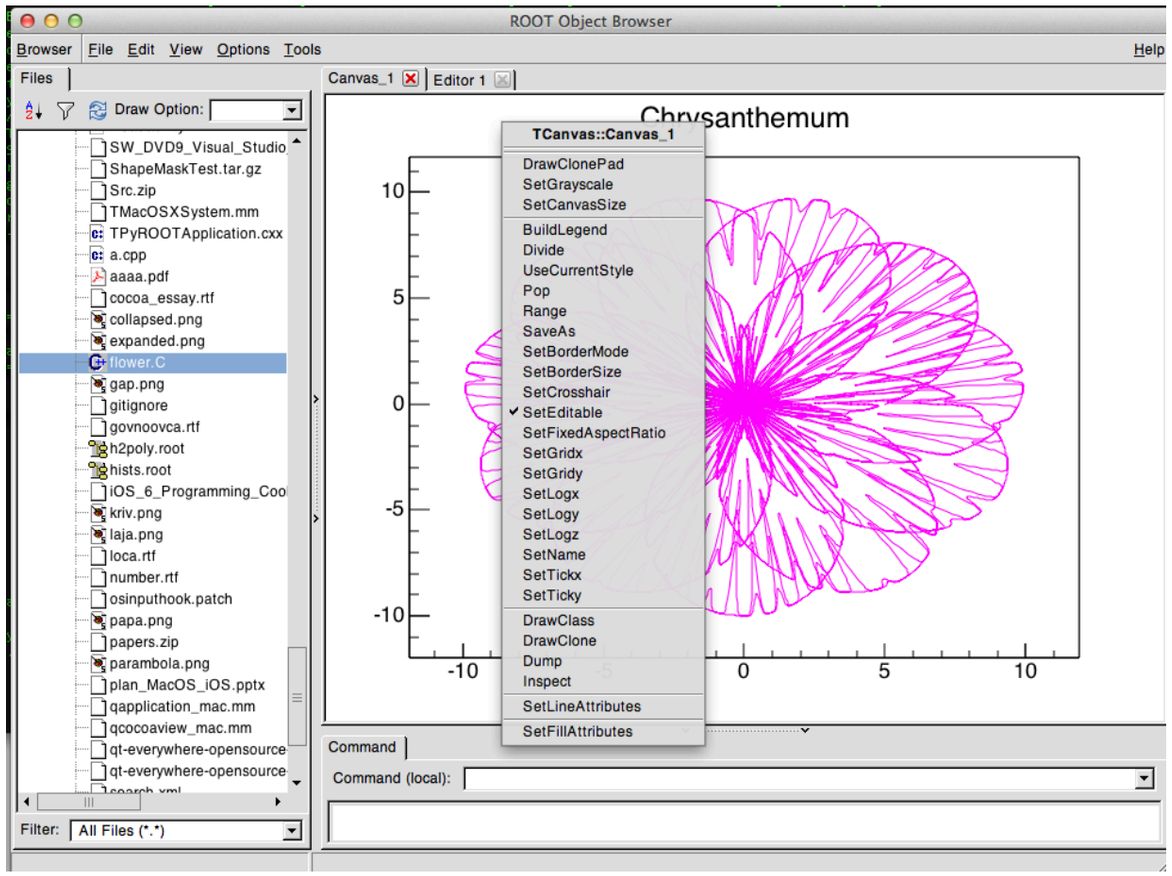
- **Cocoa**: object-oriented API, written in **Objective-C** programming language
- Descendant of the **NeXTSTEP** API ('NS' prefix in class names for 'NeXT STEP')
- Consists of:
  - **Foundation Kit** : containers, iterations, etc. - foundation.
  - **Application Kit** : GUI programming
  - **Core Data** : object persistence framework
- **Objective-C**:
  - Simple general purpose object-oriented programming language
  - Strict superset of the C programming language
  - Adds simple object-oriented features (classes, protocols, messaging etc.)
  - You can mix Objective-C and C++ : **Objective-C++**

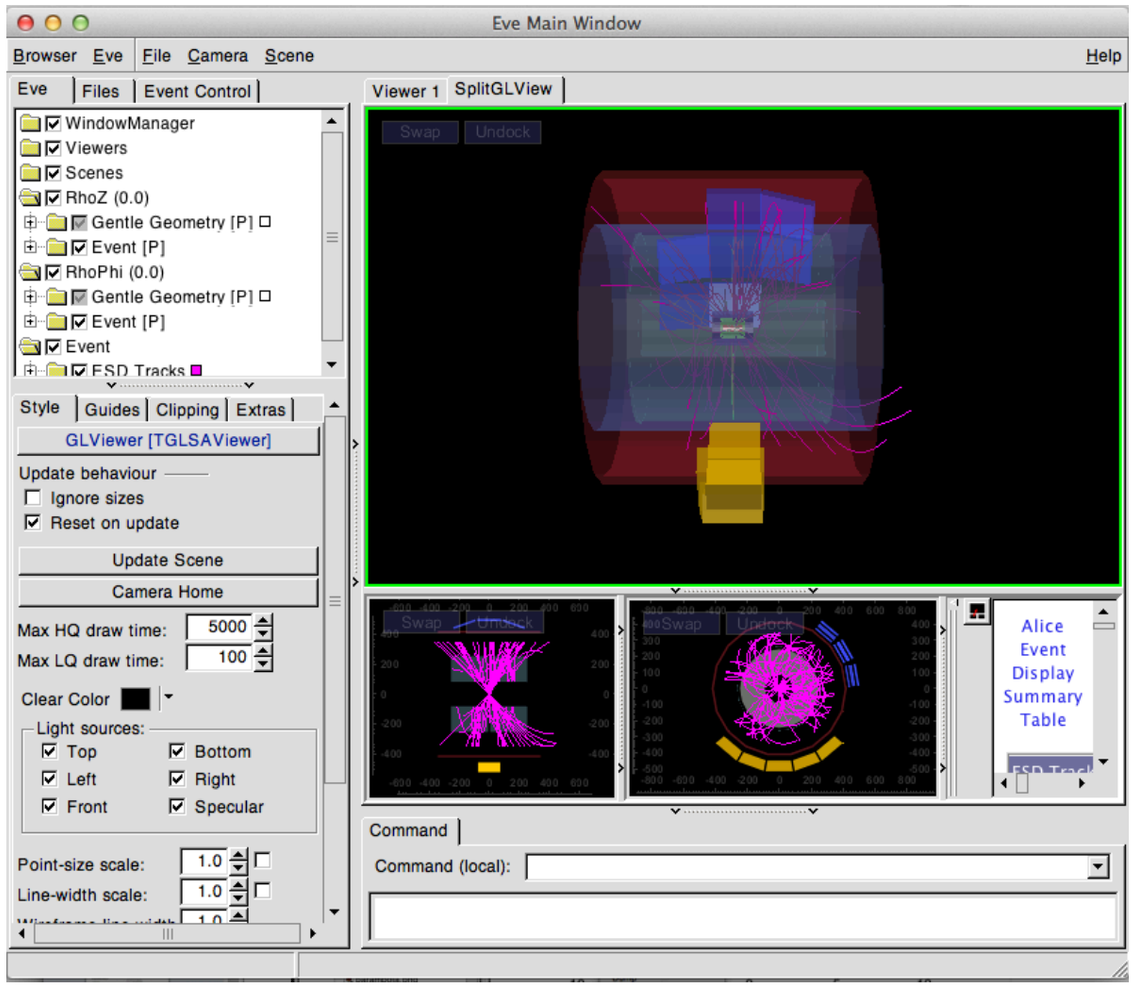
Quartz 2D

- Native API on Mac OS X for 2D graphics
- I use Quartz 2D to render:
  - Text
  - Lines
  - Paths (filled polygons and strokes)
  - Gradients
  - Shadows
- The same code draws on the screen, into the image, can also draw into PDF context.
- From wikipedia: "***Quartz 2D*** *is available to all Mac OS X and iOS application environments, and provides resolution-independent and device-independent rendering of bitmap graphics, text, and vectors both on-screen and in preparation for printing. ...*"

Implementation: TVirtualX

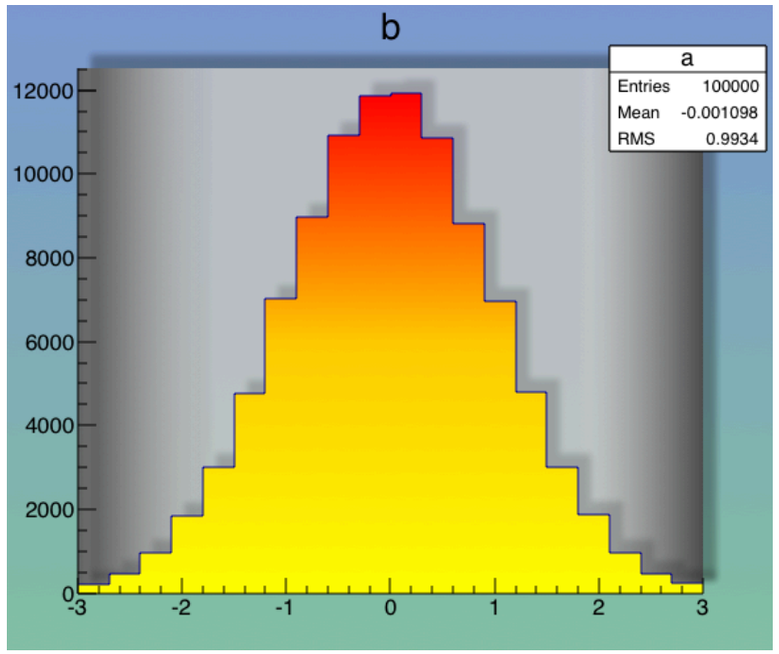
- $**ROOTSYS/graf2/cocoa** - new module introduced.
- Code is written in Objective-C++, contains C++ classes and functions + Objective-C protocols, classes.
- The “heart” of the module is the **TGCocoa class,** which inherits TVirtualX.
- ROOT is not a pure C/C++ framework anymore! ;)
- Examples: TBrowser, Eve:

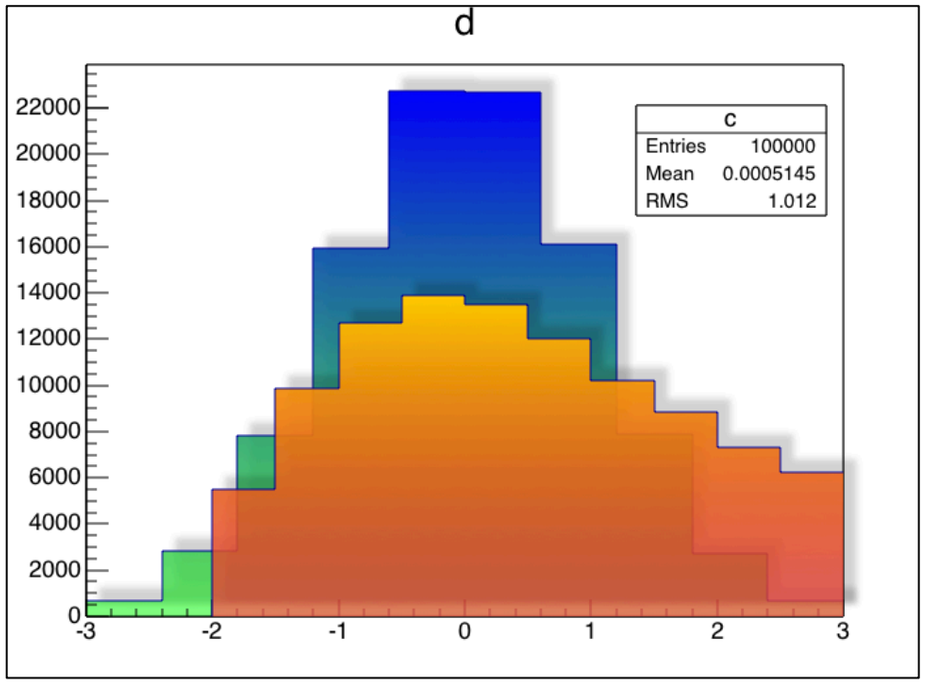


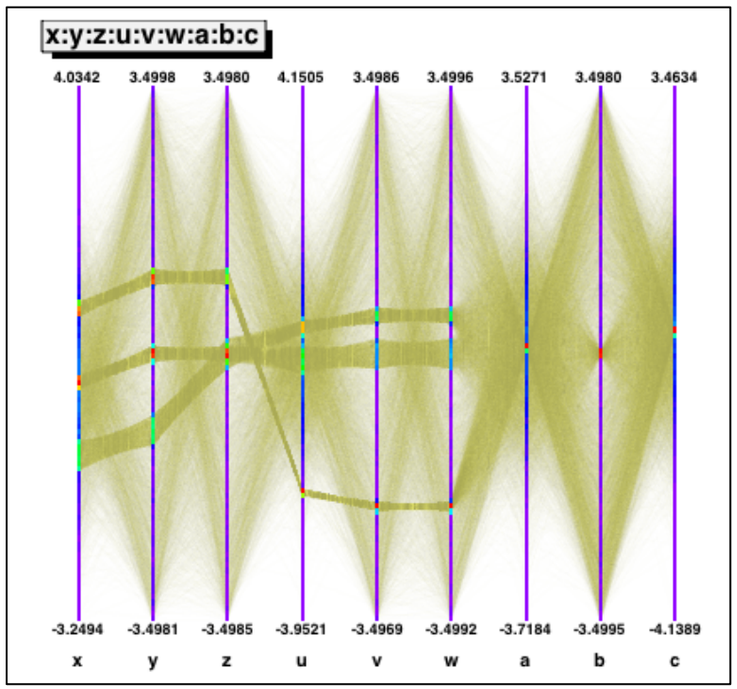


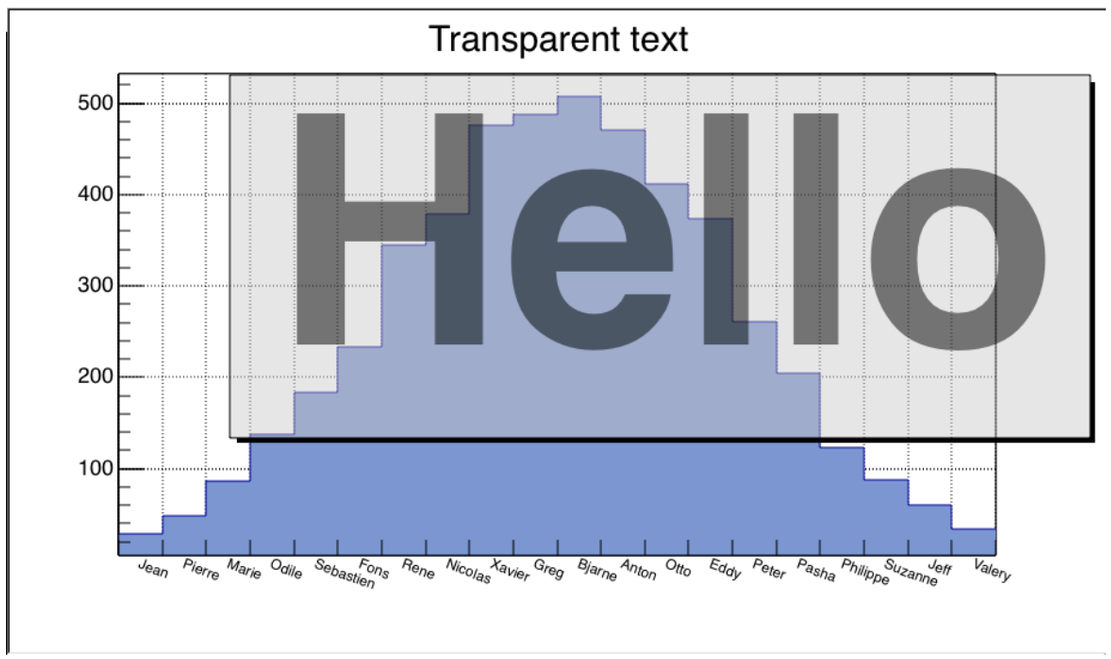
Implementation: TVirtualX (graphics, examples)

- Thanks to Quartz, we have transparency, anti-aliasing, gradients and shadows:



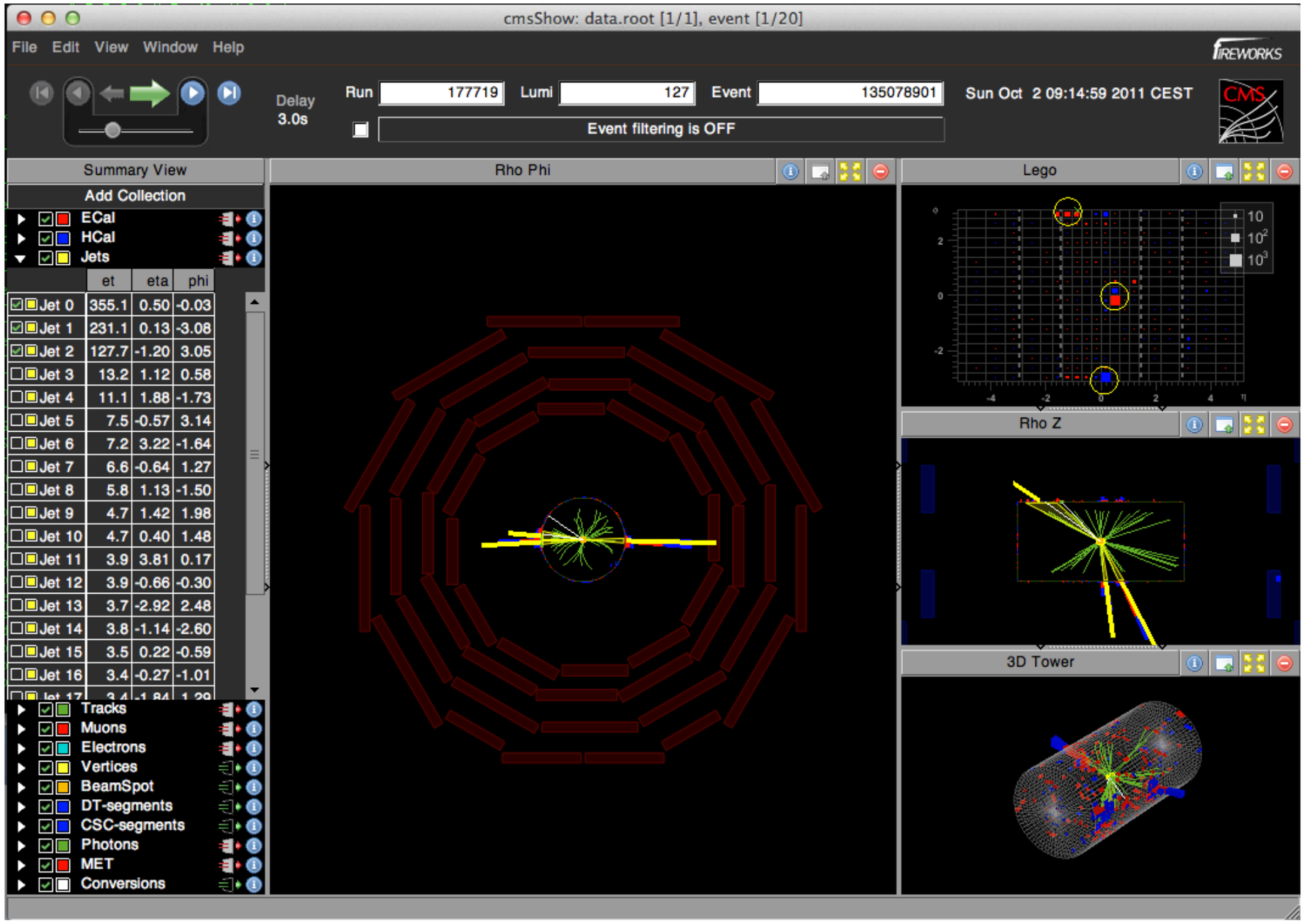






Implementation: add OpenGL

- OpenGL with Cocoa (CMS Fireworks event display, compiled by Matevz Tadel and Alja Mrak Tadel, using ROOT with Cocoa back-end):



Implementation: event loop, TMacOSXSystem

- Typical Cocoa-based application has its own event loop hidden inside a “black box” function - **`NSApplicationMain`**
- ROOT has its own event loop and knows nothing about Cocoa’s event loop.
- Cocoa has **NSApplication** class, using methods of this class you can (to some degree) emulate the same ‘native’ Cocoa’s event loop.
- This is the work done in **TMacOSXSystem** class (core/macosx module) - the new concrete TSystem’s implementation for Mac OS X and Cocoa: works like **TUnixSystem**.
- Note: we have only **one ROOT’s process** now, not two different applications, some things can happen faster than you expect - I do not have to synchronize with X11 server.

Implementation: event handling (dispatching)

- ROOT's GUI requires X11-like events: implementation should work like it's a **real X-server**.
- X11 has quite sophisticated event model
- And **Cocoa has a different event model** (some events are simpler, some - more complex, some events do not exist in Cocoa).
- **Cocoa events has to be 'translated' to X11-like events**, if it's possible, also many artificial X11 events are generated.
- Event dispatching/processing is quite complicated and after Cocoa's window receive some simple event (e.g. 'mouse button press'), twisted logic is required to mimic X-server's event dispatching :)

Implementation: drag and drop, copy and paste operations

- Similar to the Windows version of ROOT:
  - you can drag an object from a Finder's window into a ROOT's window (if it supports D&D)
  - you can drag objects between ROOT's windows (if they support D&D)
  - copy and paste operations in a ROOT's text editor (TGTextView, TGTextEditor)
- **Implementation** is a mix of:
  - **Pure emulation** (X11's **'properties'** and **'selections'**)
  - **Native API** (Cocoa).

Implementation: Retina displays

- Summer 2012: new MacBooks with Retina displays were introduced.
- It's possible to enable non-standard "HD-resolutions" on non-retina device and test your app (**Quartz Debug** tool).
- In General, TGCocoa was "Retina-ready". All GUI parts were ok (clear and crisp text). Only a few modifications were done:
  - Canvases/pads require larger pixmaps.
  - OpenGL - almost literally one call:
    - `[self  setWantsBestResolutionOpenGLSurface : YES];// somewhere inside ROOTOpenGLView class`
  - Some coordinate-translation functions had to be replaced.
- Matevz Tadel updated **graf3d/gl** and **graf3d/eve** to use larger viewports on Retina MacBooks (he also added a really nice multisampling setting into system.rootrc).

Implementation: pyROOT

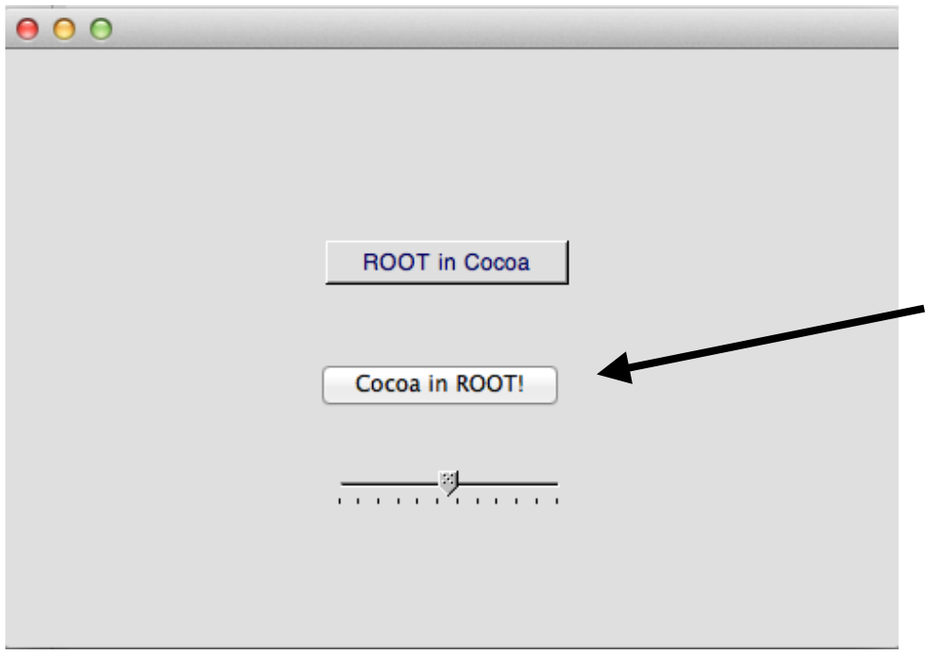
- Did not work at the beginning.
- Fixed in February 2013 (many thanks to Wim for help!):
  - Required simple modification in TMacOSXSystem (now DispatchOneEvent functions is even more TUnixSystem-like).
  - Simple modification in TGCocoa (ROOT's process manipulations/transformations)
  - Modifications in pyROOT (done by Wim).

How-to install

- Available in the 5.34 release, ROOT must be configured with **--enable-cocoa** option.
- **Is default in the trunk (on Mac OS X 10.8)**
- You can still switch back to the **good old X11** version - configure ROOT with **--disable-cocoa** option.
- Cocoa-specific tutorials are in **$ROOTSYS/tutorials/cocoa**

Conclusions, the status and future

- In general, it's a **complete TVirtualX implementation** (minus several very specific functions, which are never called from our GUI).
- It must be seriously tested - your feedback is welcome.
- **The next step is to try to use multi-threaded rendering**. Thread management is done by Cocoa automatically. The tough part is on the ROOT's side, though - gVirtualX->DoSomething() - global variables, object with a complex (shared) state.
- Now it's possible to mix native controls and ROOT's GUI



NSButton (Cocoa's control) inside TGMainFrame

ROOT on iOS

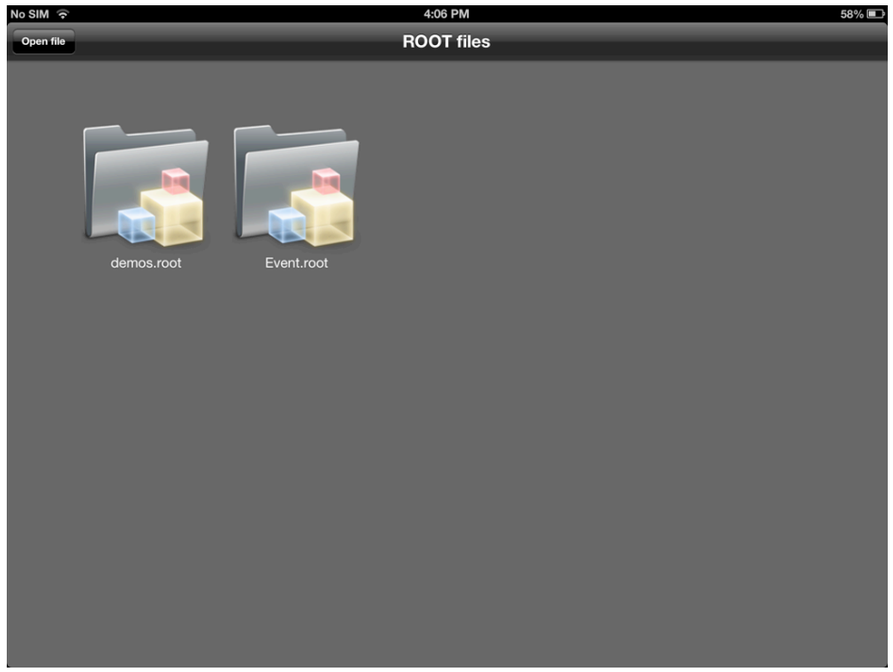
- Starting from 2010 ROOT can be compiled for iOS (but not the trunk version at the moment):
  - configure with **iossim/ios** as a platform.
- In principle, your iOS application has access to all ROOT's classes.
- But there is no ROOT's interactive session on iOS, well, this can also be done, but still:
- iOS has a very different GUI, ROOT's GUI can not be ported to iOS.
- ROOT's non-GUI graphics (2D/3D) can be ported and we have iOS specific version of TCanvas/TPad (== 2D-graphics), implementation is based on the **Quartz 2D API**, the same API as we use on Mac OS X.

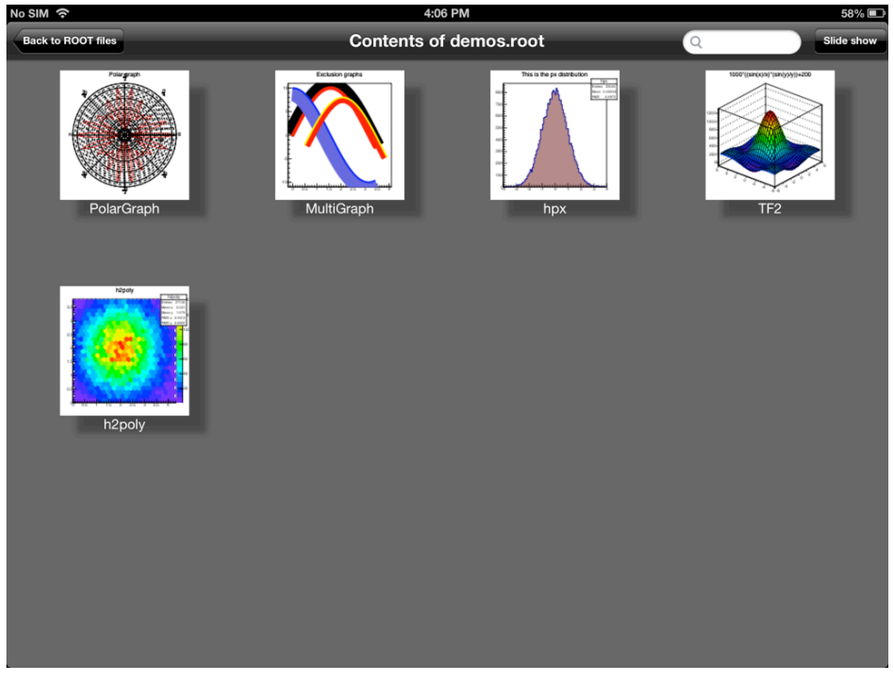
ROOT on iOS: RootBrowser app

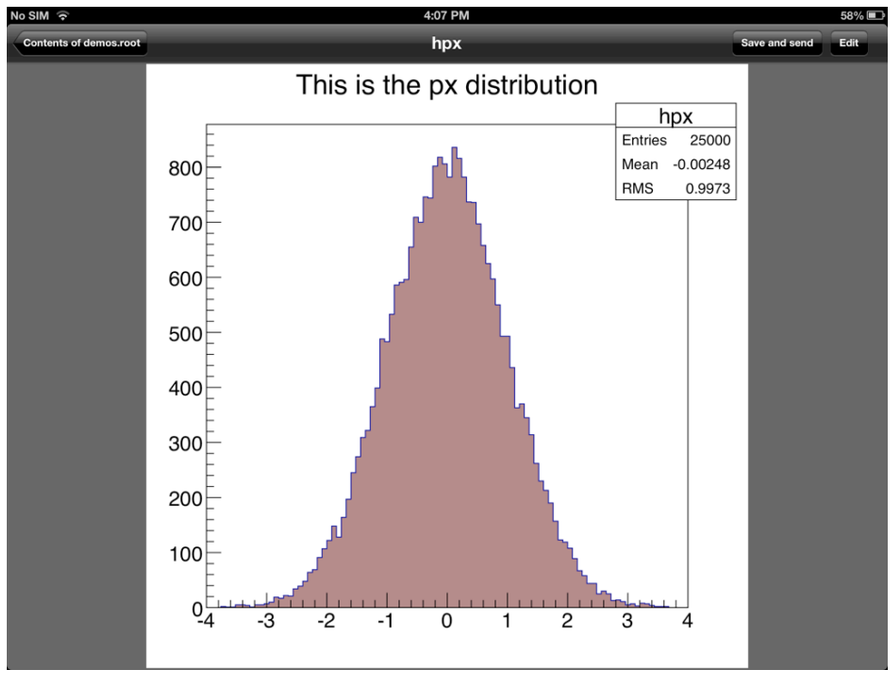
- Simple ROOT based application for iOS (iPad devices only)
- Simplified “version of TBrowser + TCanvas”.
- Uses TWebFile.
- Can browse ROOT files’ content and visualize TH1/ TH2/TGraph/TMultiGraph/TF1/TF2 objects (easy to add other ROOT’s objects).
- Has a simplified version of “graphical editor” from TCanvas.
- GUI is done with UIKit (Cocoa Touch, iOS-specific counterpart for AppKit).

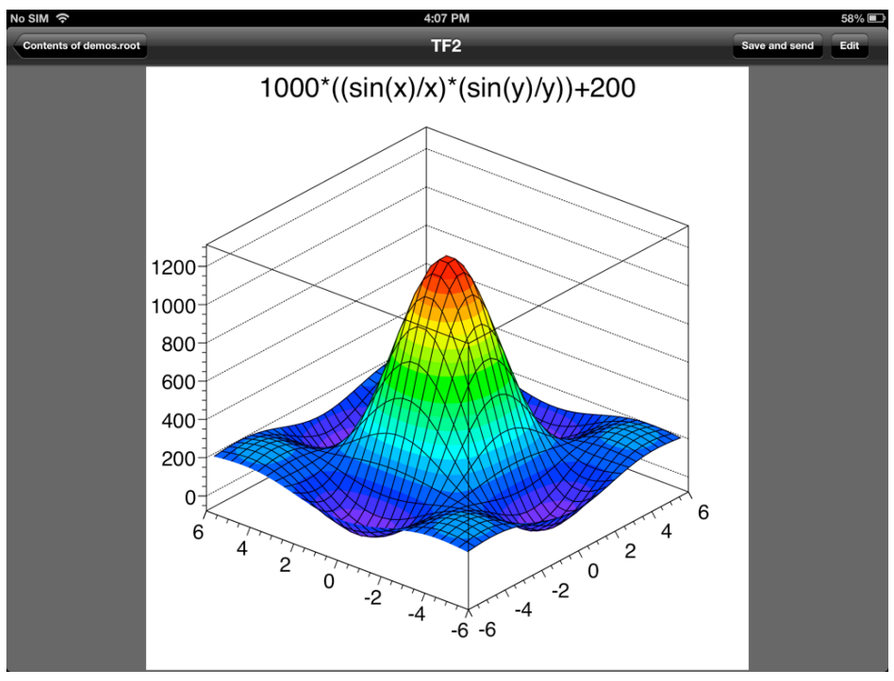
ROOTBrowser app

- Example: browsing file contents

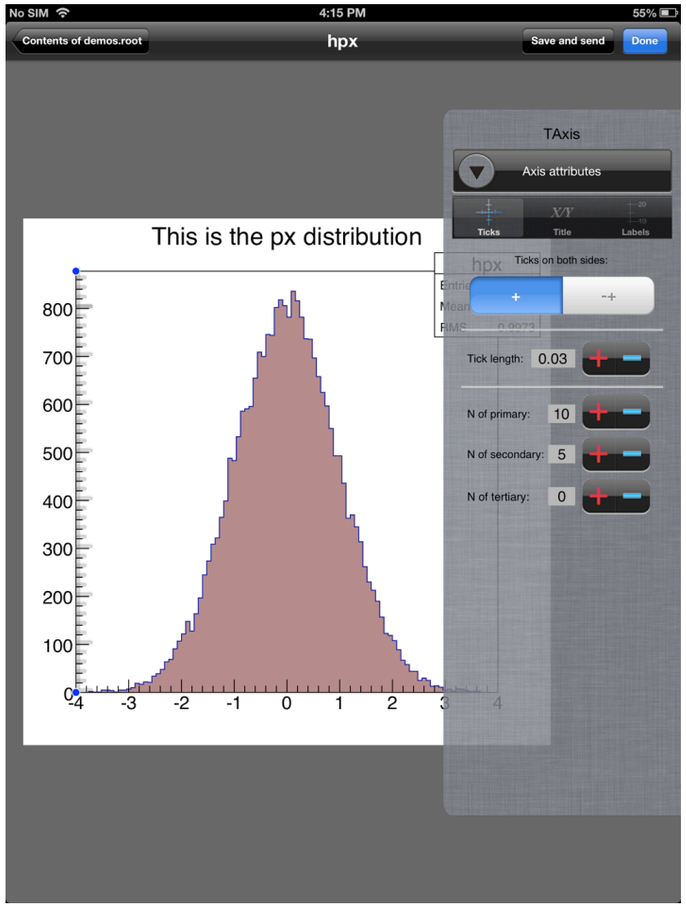


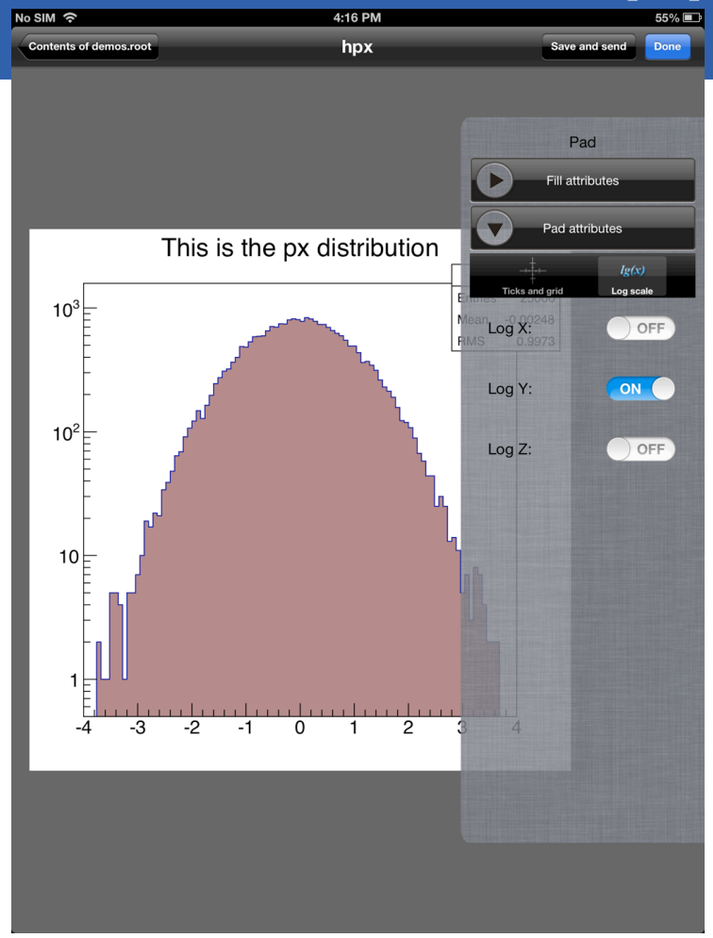


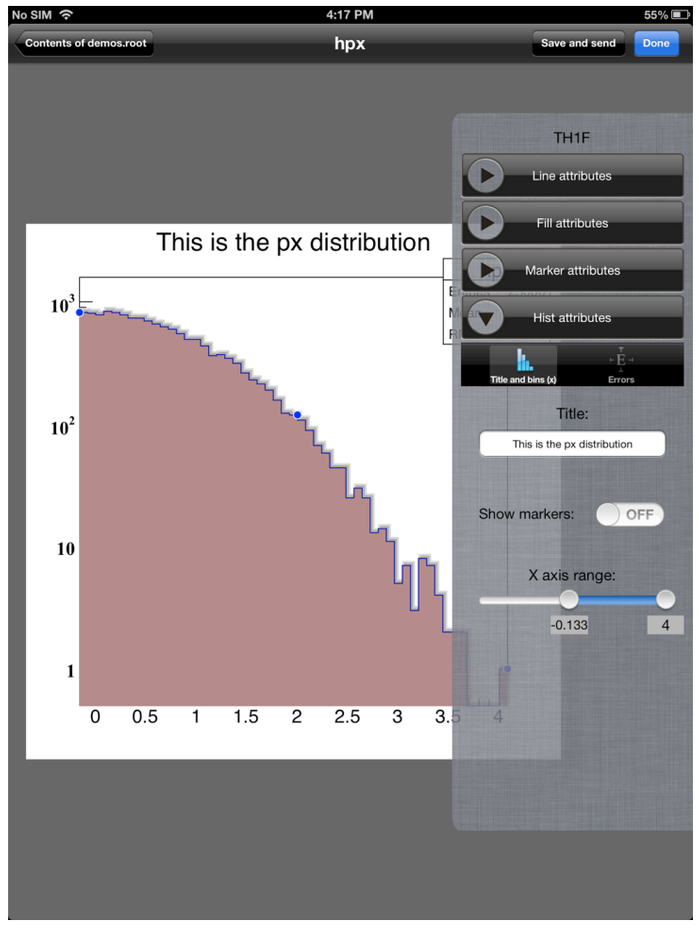


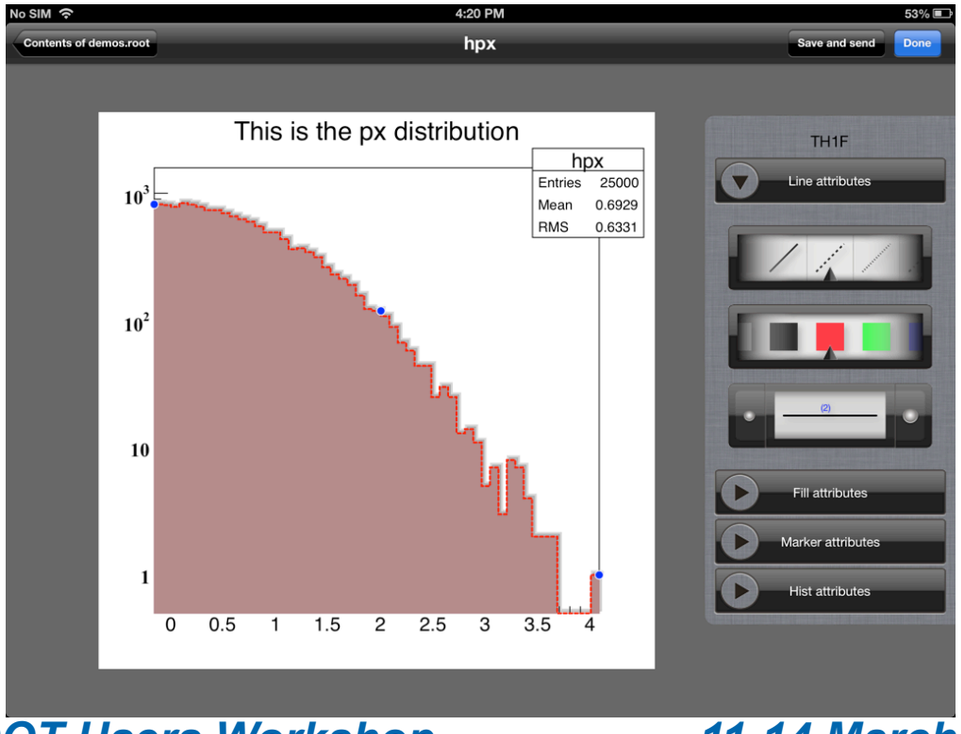


ROOTBrowser app: editor









Demos?

Thank you!