Web-based graphics and GUI

Sergey Linev
Content

• RCanvas
• RBrowser
• REve

• RFitPanel
• RGeomViewer

• reusing CHEP talk
RCanvas

- Full redesign of TCanvas class
- No gPad!
  - threads safety
- RDrawable
  - graphical primitive
  - attributes
  - reference data object
- RPadBase
  - maintain list of primitives

```cpp
#include "ROOT/RCanvas.hxx"
#include "ROOT/RHistDrawable.hxx"

using namespace ROOT::Experimental;

void draw_rh1()
{
    RAxisConfig xaxis(25, 0., 10.);
    auto pHist = std::make_shared<RH1D>(xaxis);
    pHist->Fill(5);

    auto canvas = RCanvas::Create("Canvas Title");
    auto draw1 = canvas->Draw(pHist);
    draw1->AttrLine().SetColor(RColor::kRed).SetWidth(2);

    canvas->Show();
}
```

$ROOTSYS/tutorials/v7/draw_rh1.cxx$
RCanvas

• Separate data (e.g. histogram) from view attributes

• Data can be shared via std::shared_ptr
  – provide I/O support, but only inside RCanvas

• Any attribute is optional
  – value can be provided with CSS-like syntax
  – default values provided in attribute classes

• Exactly same code for visual and batch mode
RCanvas plans

- Basic classes: RDrawable, RPadBase, RAttrBase, RColor, ...
  - review API, ~2 months
- RStyle, CSS parsing
  - ~1 month
- RHistDrawable (C++ & JS)
  - ~2 months
- RGraphDrawable (C++ & JS)
  - ~1 month
- RPalette, RFrame, RLegend, RLatex, other primitives
  - ~3 months
RCanvas testing

• Two kinds of tests are feasible:
  – batch jobs producing SVG images
    • either using headless browser mode (ROOT batch mode)
    • or run JavaScript code in Node.js
      – like https://github.com/linev/jsroot-test
    • ~1 month
  – interactivity tests
    • using tools like WebDriver
    • ~3 months
    • do not require deep knowledge of ROOT internals
      – good job for summer student
RBrowser

- Browse:
  - file system
  - ROOT files
  - TTree

- Display objects with:
  - RCanvas (ROOT7)
  - TWebCanvas (ROOT6)

- Edit text files
  - openui5 code editor

- View images

- Scalable hierarchy browser
  - load only visible items

- RBrowsable adapter classes:
  - object management
  - iterators over sub-elements
  - support old TObject::Browse(TBrowser*)
  - custom client info

root $ROOTSYS/tutorials/v7/browser.cxx
RBrowser plans

• RBrowsable
  – revise API
  – implement adapters for major ROOT classes
  – ~2 months

• Provide RFileDialog
  – to be used in different components

• Overall work estimation: ~4 months
Eve7 and FireworksWeb

Online event display

- Hierarchy browser
- 3D views
- Projection views
- Table views
- Multiple clients
- Offline mode

https://linev.github.io/eve7/
REve plans

• Mainly work of Matevz and Alja, final CHEP slide:

Development plan

- **Short term** - clean up existing code:
  - ROOT 6.20 release in December (EVE-7 is in ROOT since v6.16, Jan. 2019)
  - FireworksWeb technology preview release by the end of 2019 (FWLite based tarball)

- **Plan for 2020:**
  - EVE-7 ready as replacement for EVE
    - Most functionality supported, including physics collection / item handling.
  - FireworksWeb functional for Run3
    - Support CMS physics data-analysis & event scanning / trigger studies

- **Beyond 2020:**
  - EVE-7: optimization & beautification, and user support!
  - FireworksWeb - advanced functionality:
    - Running from full CMSSW framework & editing of CMS algorithm parameters
    - CMS geometry browser
    - Optimization for Heavy Ion runs
RGeomViewer

- Reuse eve7 and JSROOT code
- Browse hierarchy
- Search nodes
- Individual volume display
- Transparency
- Wireframes
- Offline mode

root $ROOTSYS/tutorials/eve7/viewer.cxx
RGeomViewer plans

• Beta quality:
  – most functionalities are there
  – need more testing/debugging

• To do:
  – integrate with RBrowser

• Work estimation:
  – ~2 months
RFitPanel

- Access fit functionality via web widget
  - very similar to original TFitPanel
  - use ROOT6 data classes for fitting
  - improve usability

- Example of model/view separation
  - model is C++ class
  - converted to/from JSON
  - used as is for view configuration

- Display fit results in TCanvas
  - x11 or web-based

root --web $ROOTSYS/tutorials/v7/fitpanel6.cxx

18.12.2019  S.Linev, web-based graphics and GUI
RFitPanel plans

• Beta quality, needs to be tested

• Work estimation:
  – ~0.5 month to finalize all small issues
RWebWindow

• Gateway to web-based displays in ROOT

• Launch web browser(s)
  – special support for Chrome and Firefox
  – headless mode, used for batch mode

• Local web displays
  – Chromium Embedded Framework **CEF**
  – Qt5 **QWebEngine** – also chrome-based

• Communication via websockets

• Openui5 support
  – any other GUI framework can be used

• Offline support
  – client code can be used without running ROOT
RWebWindow – batch mode

• Already working:
  – Google Chrome, but requires http
  – Mozilla Firefox, also requires http

• Can be implemented:
  – Node.js - based solution
    • text/SVG/WebGL rendering is not trivial
    • use external tools like ImageMagic?
  – CEF
    • uses X11, can be replaced by other implementation
    • requires custom compilation, extremely large and introduces many dependencies

• Main unresolved issue for 2020
• Work estimation: ~3-6 months
RWebWindow plans

• Connection sharing between widgets
  – prototyped now with RFileDialog in RBrowser
  – ~1 month

• Integration with JupyterLab
  – reuse web services
  – ~2 months

• MDI inside single web-browser page (optional)
  – handle many different widgets (canvases, browsers, fitpanel) in same browser window
  – example: https://root.cern/js/latest/api.htm#url_syntax_flexible_layout
  – use same connection, same context, do not reload JS again
  – ~3 months
Overall remarks

• Huge amount of work to complete
  – difficult promote to users before

• That is our priority list?
  – RBrowser (~4 months)
  – RCanvas (~9 months)
  – rock-solid batch mode (3+ months)
  – the rest (6+ months)
Backup slides
TWebCanvas

- Show TCanvas in browser
  - web-based TCanvasImp
- Reuse JSROOT code
- Limited support of TVirtualX
  - custom Paint() may work

root --web $ROOTSYS/tutorials/tree/parallelcoord.C

root --web $ROOTSYS/tutorials/hsimple.C
TWebCanvas with Qt5

- go4 v6.0
  - developed since 1999
  - [http://go4.gsi.de](http://go4.gsi.de)
- Qt5-based GUI
  - QtROOT for ROOT graphics
  - since a while not working on Mac (missing x11 support)
- Solution:
  - embed TWebCanvas in QWebEngine
  - provide support for custom go4 classes
- Same code for:
  - Linux/Mac/Windows
- Any ROOT web widget can be embed in Qt5:
  - `root --web=qt5 ...`
#include "ROOT/RCanvas.hxx"
#include "ROOT/RText.hxx"
#include "ROOT/RLine.hxx"

using namespace ROOT::Experimental;

void lineStyle() {
    auto canvas = RCanvas::Create("Canvas Title");
    double num = 0.3;
    for (int i=10; i>0; i--){
        num = num + 0.05;
        canvas->Draw<RText>(std::to_string(i))->SetPos({.3_normal, 1_normal*num}).AttrText().SetSize(13).SetAlign(32).SetFont(52);
        canvas->Draw<RLine>()->SetP1({.32_normal,1_normal*num}).SetP2({.8_normal, 1_normal*num}).AttrLine()->SetStyle(i);
    }
    canvas->Show();
}
#include "ROOT/RCanvas.hxx"
#include "ROOT/RText.hxx"
#include "ROOT/RLine.hxx"

using namespace ROOT::Experimental;

void lineStyle() {
    auto canvas = RCanvas::Create("Canvas Title");
    double num = 0.3;
    for (int i=10; i>0; i--){
        num = num + 0.05;
        auto text = canvas->Draw<RText>(std::to_string(i));
        text->SetPos(.3_normal, 1_normal*num);
        auto &atext = text->AttrText();
        atext.SetSize(13);
        atext.SetAlign(32);
        atext.SetFont(52);

        auto line = canvas->Draw<RLine>();
        line->SetP1(.32_normal, 1_normal*num);
        line->SetP2(.8_normal, 1_normal*num);
        auto &aline = line->AttrLine();
        aline.SetStyle(i);
    }
    canvas->Show();
}
Client

JavaScript

Controller → JSONModel

View

WebWindowHandle

Server

C++

Application code → Model

TBufferJSON

RWebWindow

THttpServer

websocket

OpenUI5

JSROOT