Beautiful, simple and remote ROOT graphics and GUI

Bertrand Bellenot (CERN),
Olivier Couet (CERN),
Sergey Linev (GSI, Darmstadt),
Axel Naumann (CERN)
ROOT graphics

• More than 20 years of glory history!
  – big effort to keep it running

• From other side, lot of inherited problems:
  – missing support of X11 on Mac
  – GTK on 64-bit Windows
  – difficulty to configure and use OpenGL
    • especially on remote X11
  – no support of tablets / smartphones
  – custom TG... GUI classes
ROOT7 graphics

• **Goals:**
  – multithreading
  – remote displays
  – multiple views
  – portable

• **Web-based**
  – C++ server
  – JavaScript clients

• **Reuse existing components**
  – *THttpServer* for communication
  – *TBufferJSON* for I/O
  – *JavaScript ROOT* as code base for clients
THttpServer

• http access to running ROOT application
  – civetweb
  – fastcgi

• execution of commands and methods

• objects hierarchy inspection

• objects visualization with JSROOT
  – possibility for fully custom UI

• **websockets support**
  – bidirectional
  – binary data (when necessary)
  – fallback solution with long poll requests
TBufferJSON

• Converts any streamable object into JSON
• ROOT I/O remains fully on server side
• Support of custom streamers
• Optional array compression
• Now also reading of objects from JSON

• Significantly simplifies data exchange between C++ server and JavaScript-based clients
JavaScript ROOT

- ROOT objects display in web browsers
- Binary data reading, including TTree
- ROOT JSON format support
- User interface for the THttpServer

- Developed since 2012
  - [https://root.cern/js/](https://root.cern/js/)
  - [https://github.com/root-project/jsroot](https://github.com/root-project/jsroot)
https://root.cern/js/latest/examples.htm#tgraph_excl

\[ \tan \beta \]

- \( t \rightarrow bH^+, H^+ \rightarrow \tau V \)

- \( gb \rightarrow tH^+, H^+ \rightarrow \tau V \)

**ATLAS**

\[ \int L dt = 300 \text{ fb}^{-1} \]

- Maximal mixing

- **LEP 2000**

\[ gb \rightarrow tH^+, H^+ \rightarrow tb \]
https://root.cern/js/latest/examples.htm#tgeo_proj
New in JSROOT

• Full support of TTree reading
  – including TTree::Draw() functionality
• Full support of Node.js
  – reading binary ROOT files
  – creation of SVG images
  – npm install jsroot
• Support of more classes
  – TGraphPolar, TSpline, TPolyLine3D, ...
• Special painter classes for ROOT7
  – coexistence of v6 and v7 graphics

See poster #348
RWebWindow class

• Server-side entity in new ROOT7 window management
• Display window in web browser(s)
• Communicate with multiple clients
• Support of batch mode
using namespace ROOT::Experimental;

// create window instance
auto window = RWebWindowsManager::Instance()->CreateWindow();

// configure html page loaded when window shown
window->SetDefaultPage("file:Main.html");

// this is call-back, invoked when message received via websocket
window->SetDataCallBack([](unsigned connid, const std::string &arg) {
    printf("Get msg %s from %u\n", arg.c_str(), connid);
});

// configure predefined geometry
window->SetGeometry(300, 300);

// display window
window->Show();
RWebWindow - client side

<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>RWebWindow example</title>

<script src="/jsrootsys/scripts/JSRootCore.js" type="text/javascript"></script>

<script type="text/javascript">
function InitCustomUI(handle) {
    // assign receiver object - here dummy
    handle.SetReceiver({});
    // connect to the server
    handle.Connect();
    // start loading openui5 components
}

JSROOT.ConnectWebWindow({
    prereq: "openui5",
    callback: InitCustomUI
});
</script>
</head>
<body class="sapUiBody" id="content" role="application">
</body>
</html>
Client – server communication

• Bi-directional
  – websockets
  – long polling
  – cefQuery (CEF, local)
  – QWebEngineUrlSchemeHandler (Qt5, local)

• Text exchange in both directions
  – binary transfer only from server to client

• Asynchronous
  – server and client sends data independent from each other

• Flow control
  – credit-based, avoid oversubscription of communication channel
RCanvas class

• Central graphics class in ROOT7
• Uses RWebWindow to implement display

• Main idea – decouple data from drawing attributes
using namespace ROOT::Experimental;

// Create histograms
TAxisConfig xaxis(100, -10., 10.);
auto pHist = std::make_shared<ROOT::Experimental::TH1D>(xaxis);
auto pHist2 = std::make_shared<ROOT::Experimental::TH1D>(xaxis);

// fill random points
TRandom3 random;
Float_t px, py;
for(int n=0;n<10000;++n) {
    random.Rannor(px,py);
    pHist->Fill(px-2);
    pHist2->Fill(py+2);
}

// Create a canvas to be displayed.
auto canvas = RCanvas::Create("ROOT7 Canvas");
canvas->Draw(pHist)->SetLineColor(RColor::kRed);
canvas->Draw(pHist2)->SetLineColor(RColor::kBlue);
canvas->Show();
RCanvas screenshot

Graphics attributes editor
Data generations

C++

- **RCanvas**
  - **RHistDrawable**
    - shared_ptr<TH1D>
    - RLineAtt
    - RFillAtt
  - **RHistDrawable**
    - shared_ptr<TH1D>
    - RLineAtt
    - RFillAtt
  - **RLabel**
    - std::string
    - position
    - color

**JSON**

- **RPadDisplayItem**
  - **RHistDisplayItem**
    - TH1D*
    - RLineAtt*
    - RFillAtt*
  - **RHistDisplayItem**
    - TH1D*
    - RLineAtt*
    - RFillAtt*
  - **RLabelDisplayItem**
    - std::string
    - position
    - color

**JavaScript**

- **v7.RCanvasPainter**
  - **v7.RH1Painter**
    - drawing
    - update
    - interaction
  - **v7.RH1Painter**
    - drawing
    - update
    - interaction
  - **v7.RLabelPainter**
    - simple draw
    - simple update
    - interaction

- flat list of drawbles
- reference data object
- includes graphical attributes
- temporary data containers
- includes data for painting
- may not include original data
- reuse JSROOT as much as possible
- pure C++ painter will be supported through TVirtualX-like interface
Interactivity

• Client side
  – tooltips
  – bins highlight
  – object context menu
  – zooming

• Client -> Server -> Client
  – change color, line or fill attribute
    • modify request to server
    • server automatically updates all other clients
ROOT7 GUI

- ROOT7 needs not only graphics
  - RBrowser, RFitPanel, RGEditor, ...

- Library for buttons, checkbox, list, menu, ...
  - SAP OpenUI5 http://openui5.org/

- Fully supported in RWebWindow
  - any other library can be used
FitPanel

- Inspired from the existing FitPanel layout
- Reimplement with OpenUI5
- Prototype with v6 fitting
  – by Illiana Betsou
WebEve prototype (see next talk from Alja)
Conclusion

• Full relaunch of graphics and GUI classes
• Use web techniques
• Reuse already existing ROOT components
• Provide more flexibility in implementing user code