New ROOT graphics language

Iliana Betsou  National Technical University of Athens
Serguei Linev  GSI Darmstadt
Bertrand Bellenot  CERN
Olivier Couet  CERN
WebEve

https://linev.github.io/eve7/
Why new graphics?

- ROOT6 GUI is very “OS-specific”
- OpenGL + remote X11
- X11 will not be supported on MacOS
- Single threaded
- Long-term maintenance problems
- Lack of new technologies
Objectives of the new version

- Graphics can be reinvented based on 20 years of experience
- Based on widely used technologies (SVG/WebGL) supported by a very large community

- What we want from the new version:
  - Portable
  - Remote displays
  - Multiple views
  - Multi threaded
  - Remote/web GUI
  - Support any future new platform (as soon as a web browser is available)
What is ROOT7

- ROOT7 is a local application which uses web technologies
- ROOT7 is able to run in the Web Browser
How it’s working (1/2)

Client/Server Architecture

Runs directly in a browser
- Client side → JavaScript

Components:
- **JavaScript ROOT** → code base for clients
- **THttpServer** → communication
- **TBufferJSON** → I/O
- **OpenUI5** → JavaScript Library

**ROOT**
- Server side → C++
JavaScript ROOT

- ROOT objects displayed in web browsers
- Binary data reading, including TTree
- Uses ROOT JSON format
- It is used by the ROOT jupyter interface
- 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)
THttpServer

- Gives HTTP access to running ROOT application
- Execution of commands and methods
- Objects visualization with JSROOT
- Websockets support

bidirectional

binary data
TBufferJSON

- Converts any streamable object to JSON
- ROOT I/O remains fully on the server side
- Support of custom streamers
- Optional arrays compression
- Reading objects from JSON

Simplifies data exchange between C++ server side and JavaScript-based clients
OpenUI5 is the new technology we have used to design the layout of the new ROOT widgets. It is a JavaScript UI library consisting of a really large number of UI controls.
RWebWindow Class (1/2)

- Server-side entity in new ROOT window management

- Main functionality:
  - Display window in web browser(s)
  - Manage multiple connections with the clients
  - Data transfer to/from clients
  - Support of batch (headless) 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 from client
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();
Advantages:
- Independent from any local graphics backend
- Allows remote display on all kind of devices
- For local display, the JavaScript rendering might be performed in a local canvas

Implementation of such system already exists for ROOT6 → JSROOT
- The initial goal of JSROOT was to read/browse objects in ROOT files and display them in a web browser using JavaScript
- Then the displayed objects could be manipulated in the web browser (zoomed, scaled, etc...)
Concepts of ROOT7 (1/2)

- **Pad:**
  - Base entities containing the list of graphics objects to be drawn
  - Implemented in the `RPad` C++ class

- **Canvas:**
  - A window’s topmost pad.
  - Implemented in the `RCanvas` C++ class.

- **Drawable:**
  - Something which can be drawn on a pad.
  - Implemented in the `RDrawable` C++ class
  - Each drawable entity has a `GetDrawable` method
Concepts of ROOT7 (2/2)

Canvas
- Pad1
  - Drawable 1-1
  - Drawable 1-2
  - Drawable 1-3
  - ...
- Pad2
  - Drawable 2-1
  - Drawable 2-2
  - Drawable 2-3
  - ...

C++

Painting
HTTP Server

Web Browser
Drawable’s painters are implemented here.

JavaScript

I. Betsou, "New ROOT graphics language", ACAT 2019
Two main kinds of batch output images:

- **Vector graphics output** (PDF, SVG, Latex) → implemented by native ROOT classes (ROOT6)

- **Bitmap output** (PNG, JPEG, etc) → implemented natively on top of libAfterImage (ROOT6)

**GOAL in ROOT7**: Not rely on any native ROOT library

- **Headless mode** (Chrome/Firefox): the complete HTML/JavaScript/SVG code works without screen display
- Use this mode **to generate** SVG/PNG/JPEG/PDF images
- **MathJax.js** → for Latex (we are also working for a simpler solution)
JSROOT examples (1/2)

Red Curve / SumW2 Histo errors

Dashed Curve / No XError bars

Filled Curve / Blue Histo

Partial Range / Filled Bar chart
JSROOT examples (2/2)
FitPanel Example
Conclusion

- Graphics based on new technologies (SVG/HTML/WebGL)
- Client/Server model with C++ on the server side and JavaScript on the client side
- Batch output do not rely on any native ROOT Libraries with the use of headless mode
- Basic graphics classes have been implemented
Future Plans

- Authentication and authorization
  - Support OAuth technology in the future
- Implementation of main ROOT widgets
  - RFitPanel, RBrowser, RWebEve
- Painters on the JavaScript side for all high level objects
- Generation of batch graphics images
  - Investigate a solution for TeX
- Testing

I. Betsou, "New ROOT graphics language", ACAT 2019
Thank you for your attention!!
More Stuff!!

HTML RWebWindow Class
ROOT7 GUI

- ROOT7 need more than just graphics
  - RBrowser, RFitPanel

- Library for buttons, checkboxes, list, menu,…
  - SAP OpenUI5: https://openui5.org

- OpenUI5 fully supported in RWebWindow
  - Any other library can be used
How it’s working (2/2)

ROOT is running as usual and generates the graphics display list by drawing objects → **C++ side**

The graphics display list is sent to a client (Web Browser/libChromium) → **JavaScript side**
Multithreading

- Default:
  - All RWebWindow runs in main thread

- Optional:
  - Special thread for THttpServer
  - Dedicated thread for every RWebWindow
  - Dedicated thread(s) for clients communication

- Need to care about I/O thread safety
JSROOT examples (3/3)

drawing `px:py::pz>5` from ntuple

<table>
<thead>
<tr>
<th>Entries</th>
<th>6174</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean x</td>
<td>-0.02359</td>
</tr>
<tr>
<td>Mean y</td>
<td>0.01497</td>
</tr>
<tr>
<td>Std Dev x</td>
<td>1.872</td>
</tr>
<tr>
<td>Std Dev y</td>
<td>1.871</td>
</tr>
<tr>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>12</td>
<td>5904</td>
</tr>
<tr>
<td>0</td>
<td>216</td>
</tr>
</tbody>
</table>
RWebWindow Class(3/3)

<!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 InitUI(handle) {
            // full access to OpenUI functionality
        }

        JSROOT.ConnectWebWindow({
           prereq: "openui5",
           receiver: {
              OnWebsocketOpened: function(conn) {},
              OnWebsocketClosed: function(conn) {},
              OnWebsocketMsg: function(conn, data, len) {}
           },
           callback: InitUI });
      </script>
   </head>
   <body class="sapUiBody" id="content" role="application">
   </body>
</html>
WebEve
FitPanel Example