New tools to build web-based graphics and user interfaces in ROOT

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

ROOT7 graphics and UI

- Goals:
  - portable
  - multiple views
  - remote displays
  - multithreading
- 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 arrays compression
- Now also reading of objects from JSON

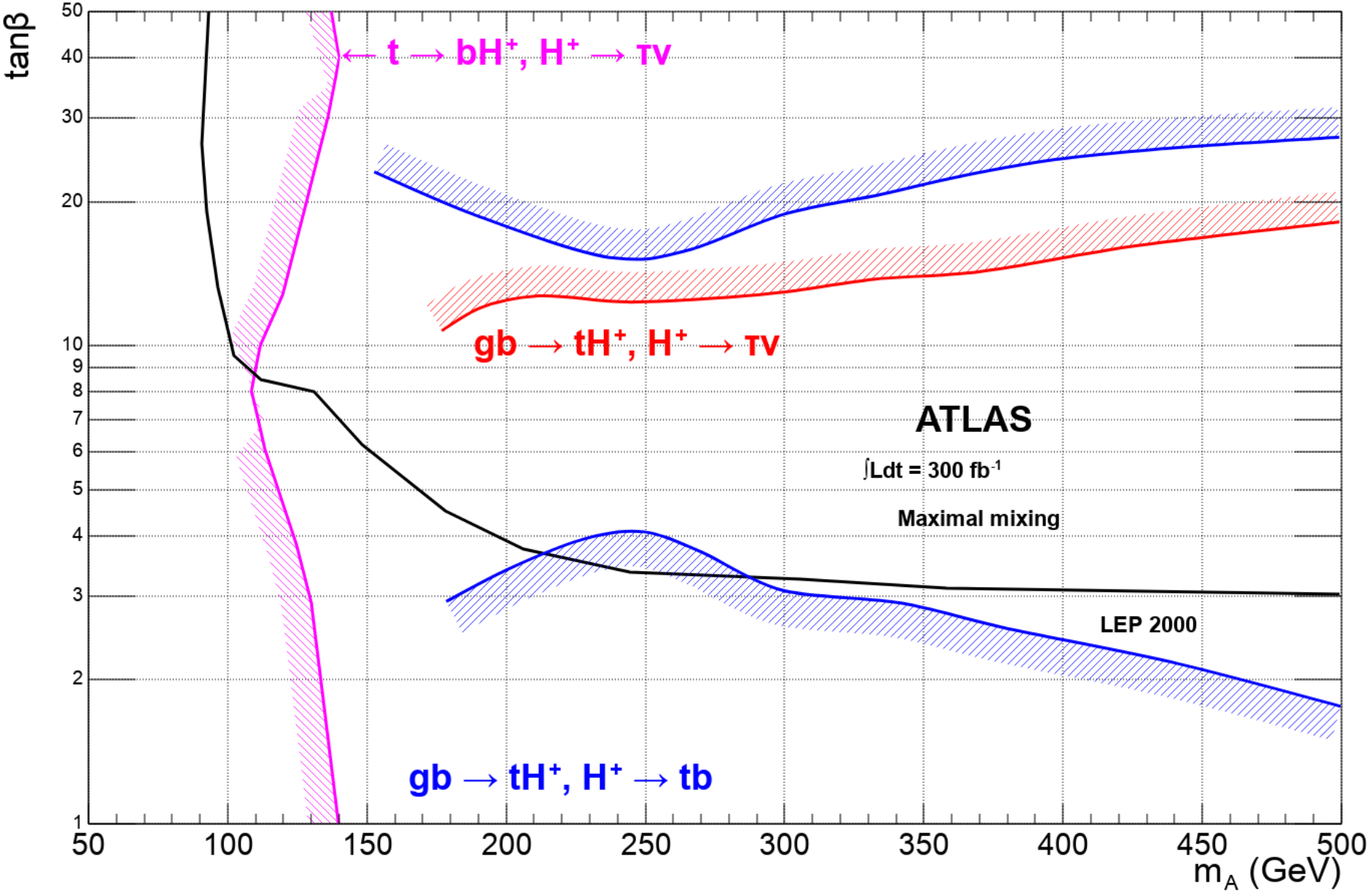
- Significantly simplifies data exchange between C++ server and JavaScript-based clients

JavaScript ROOT

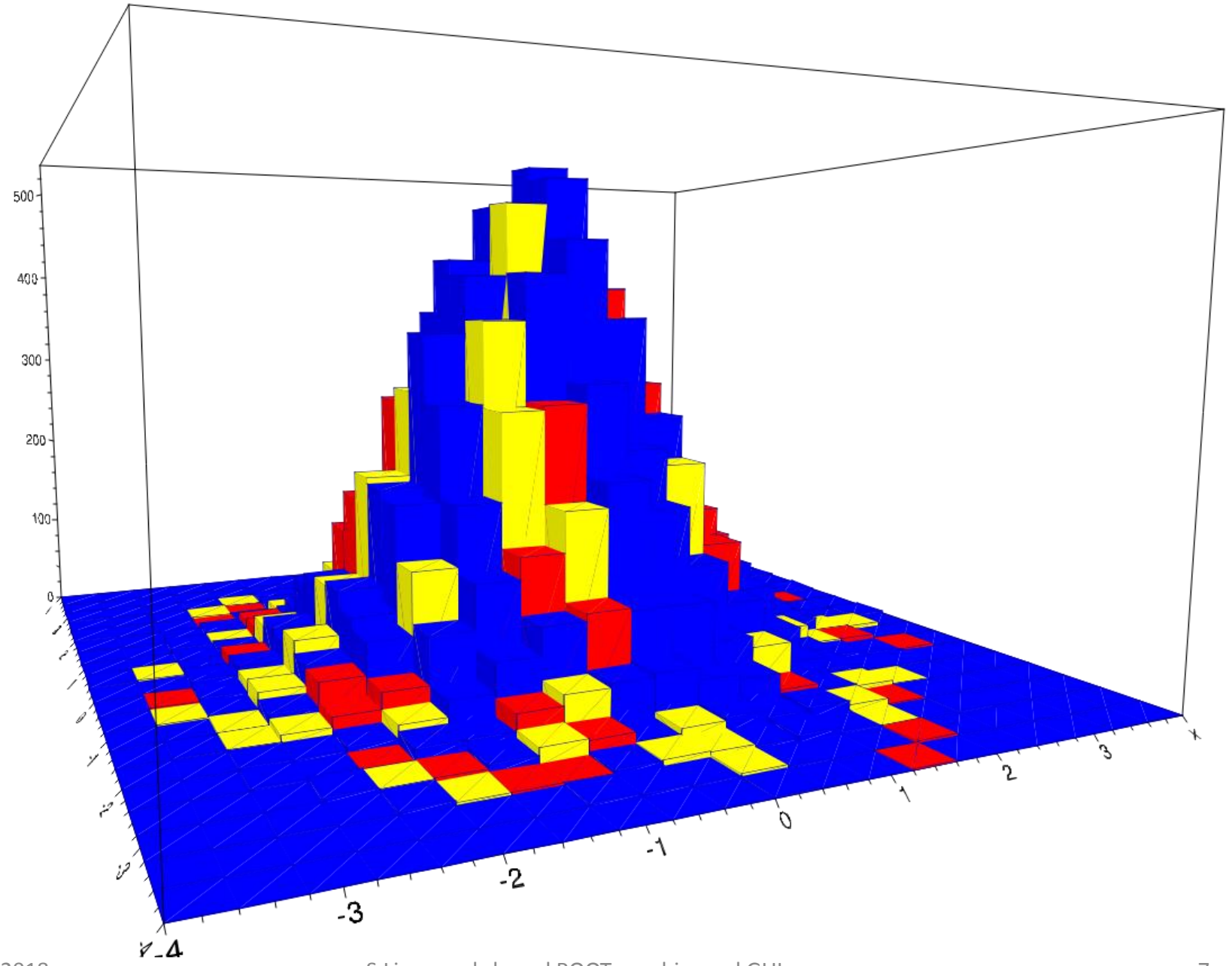
- ROOT objects display in web browsers
- Binary data reading, including TTree
- ROOT JSON format support
- User interface for the THttpServer

- Developed since 2012
  - https://root.cern/js/
  - https://github.com/root-project/jsroot

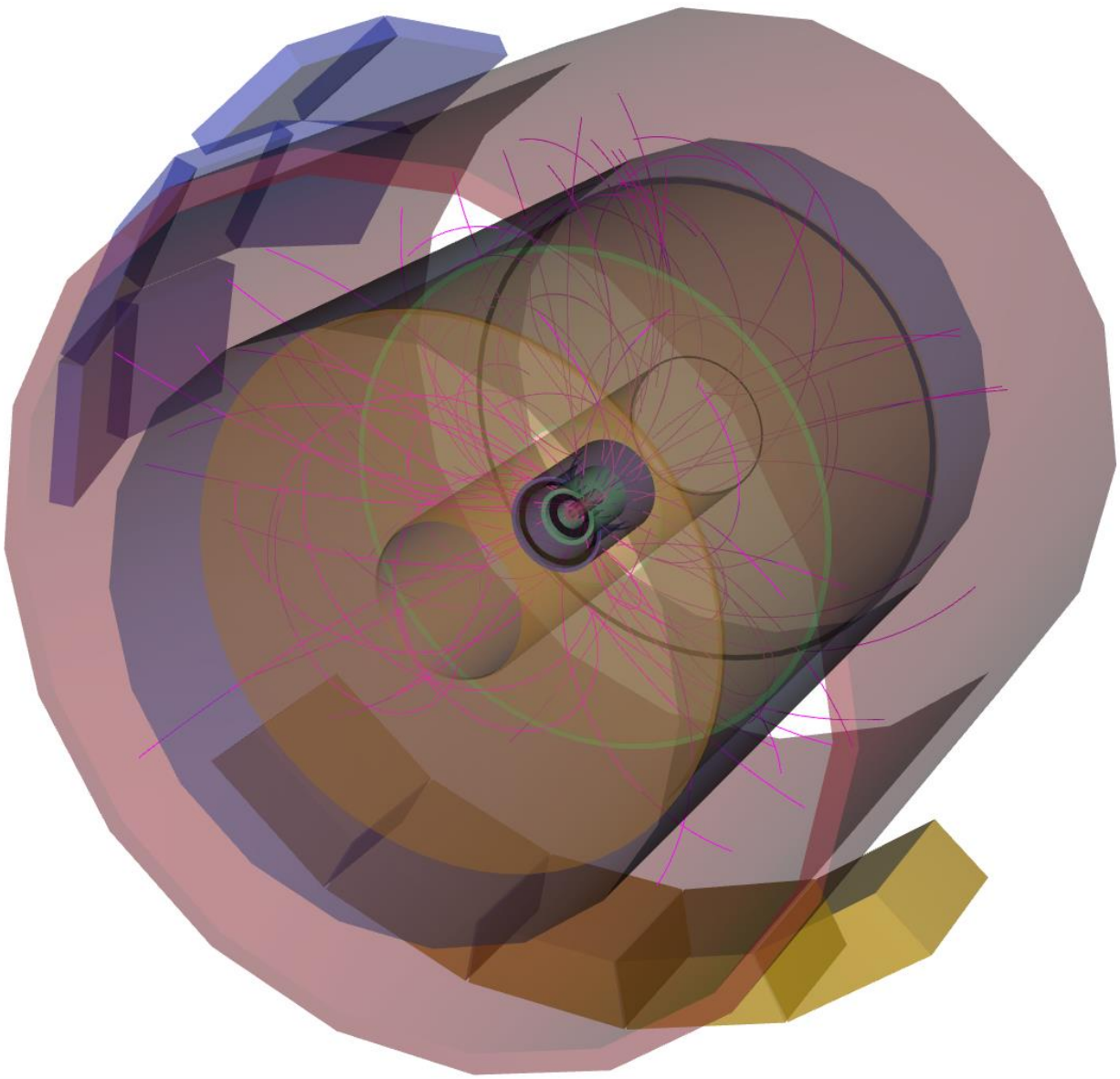
https://root.cern/js/latest/examples.htm#tgraph_excl

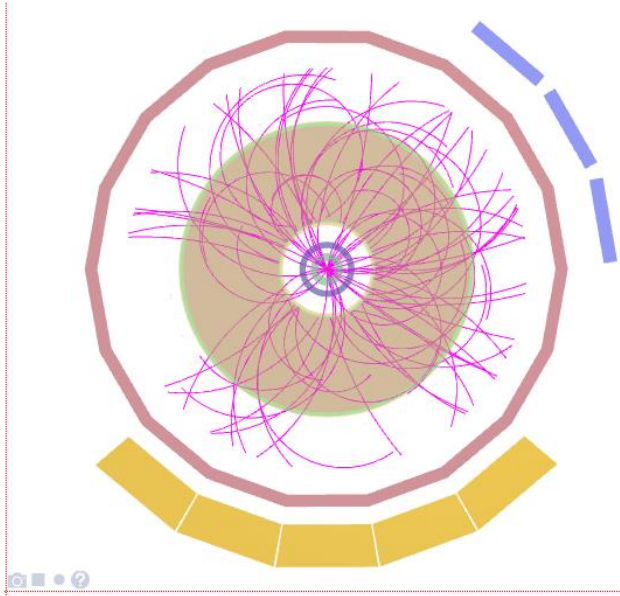


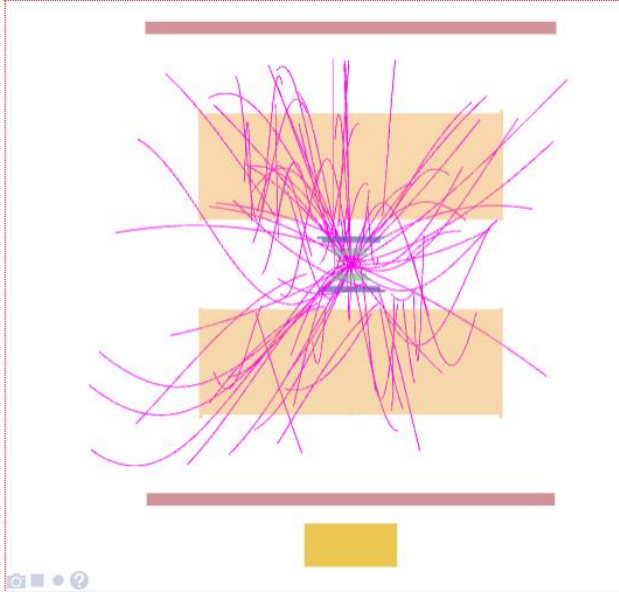
https://root.cern/js/latest/examples.htm#thstack_lego



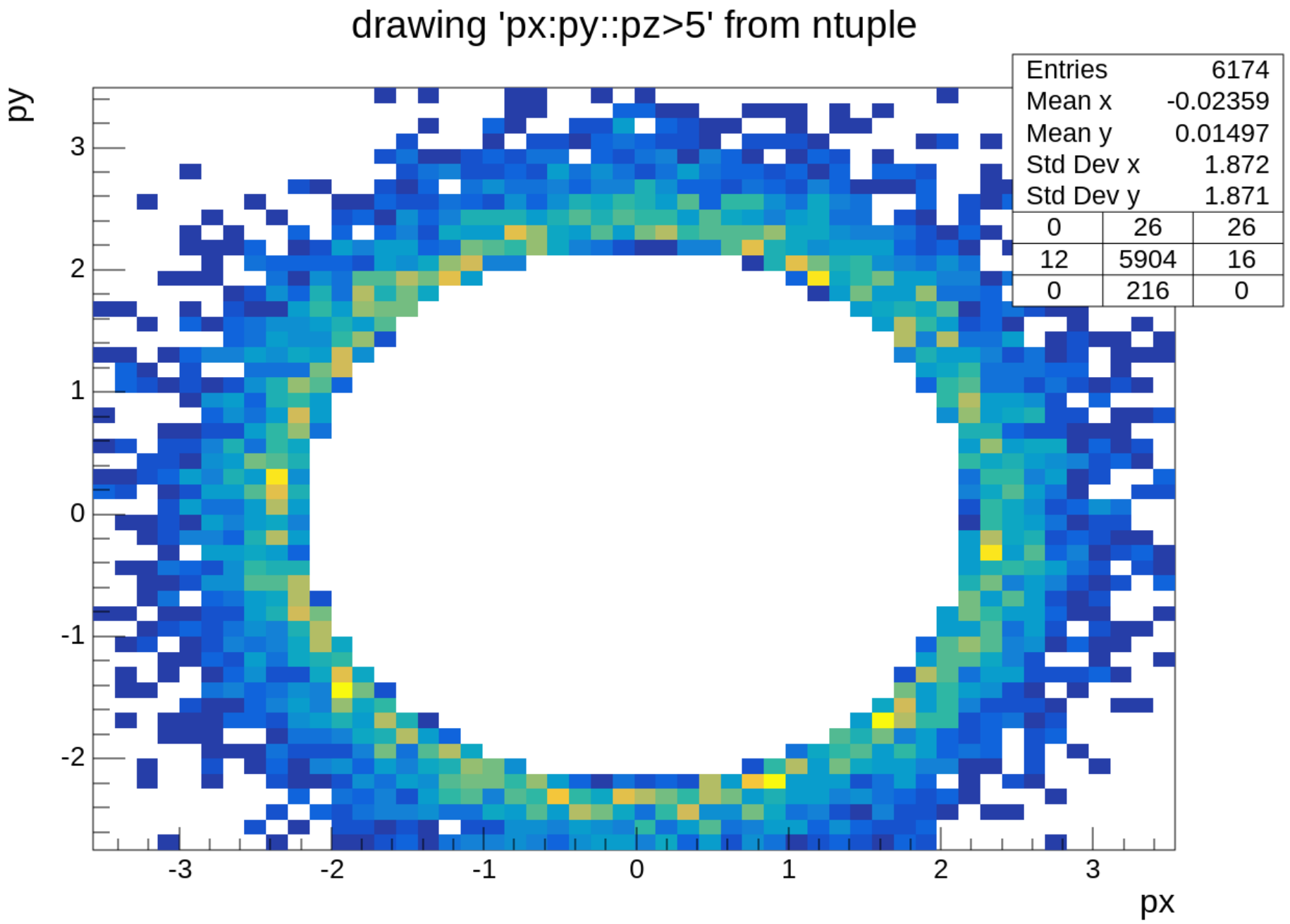
https://root.cern/js/latest/examples.htm#tgeo_proj







https://root.cern/js/latest/examples.htm#ttree_cut



RWebWindow class

- Server-side entity in new ROOT7 window management

- Main functionality:
  - display window in web browser(s)
  - manage multiple connections with clients
  - data transfer to/from clients
  - support of batch (headless) mode

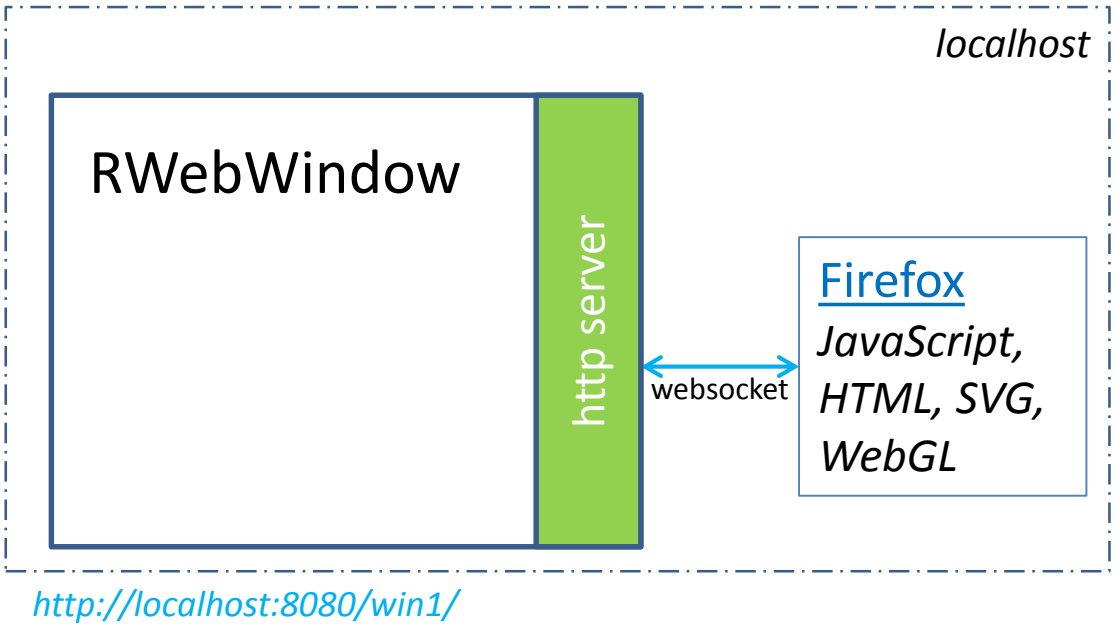
RWebWindow - server side

```
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();
```

RWebWindow - client side

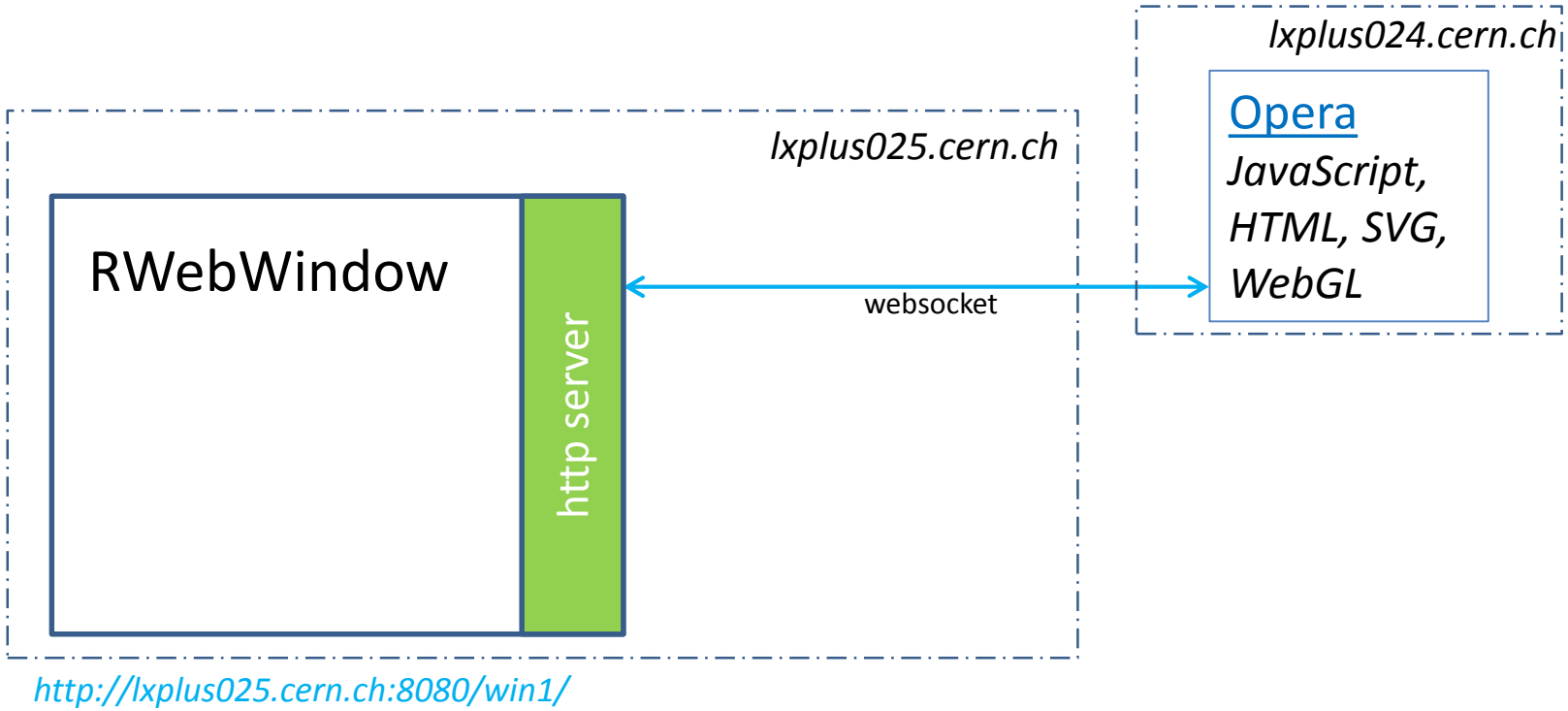
```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>RWebWindow example</title>

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

    <script type="text/javascript">
       function 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>
```

RWebWindow clients



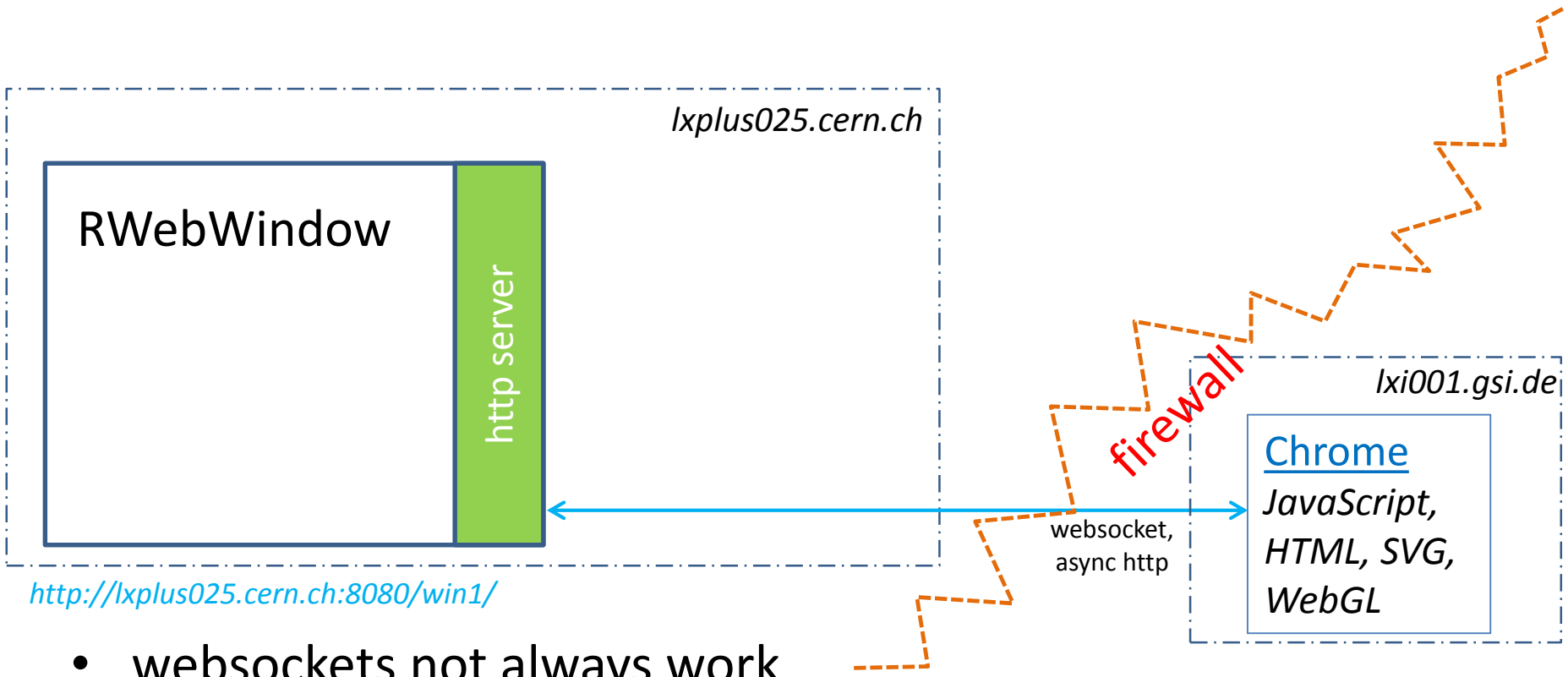
- Any modern web browser – typically on the same host
- THttpServer bound to loopback address
- Communication via websockets

RWebWindow clients - LAN



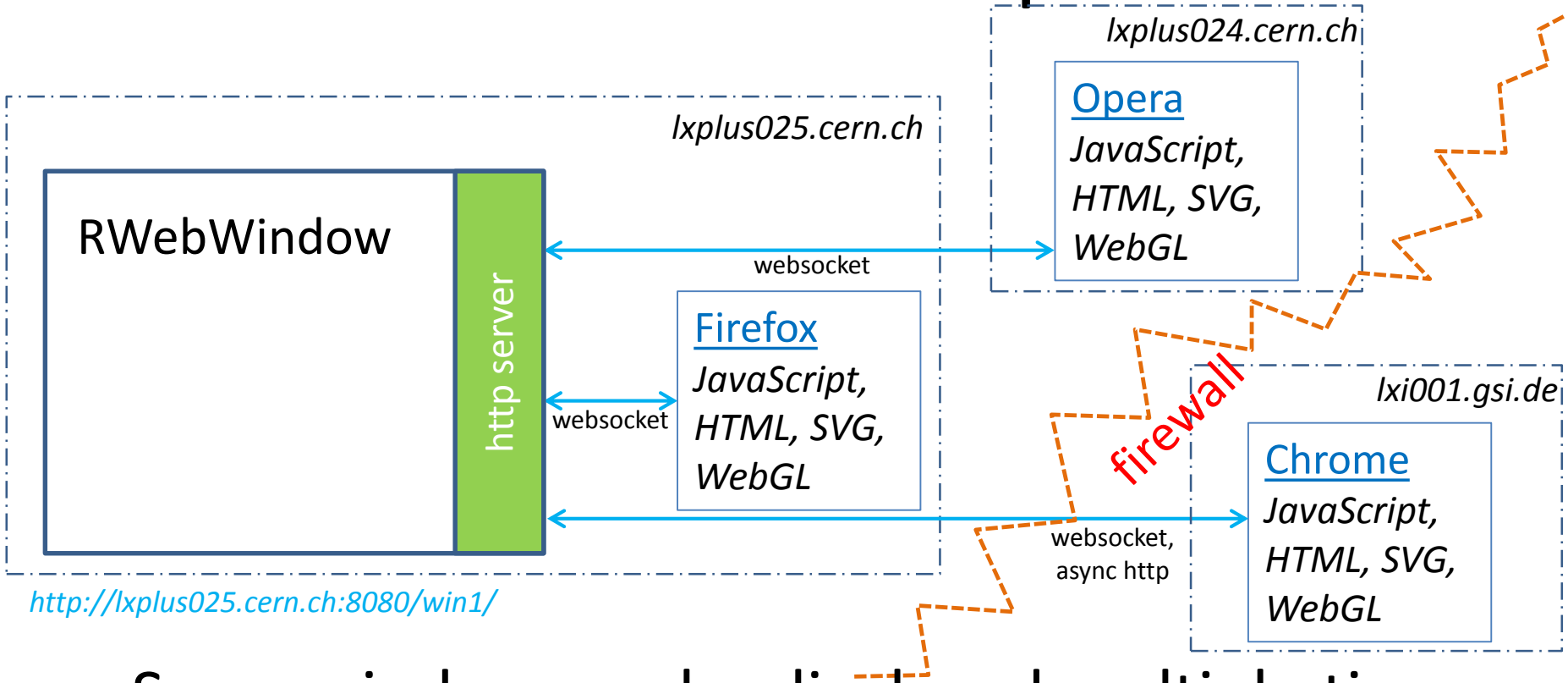
- No any difference with local clients
- Bound THttpServer with normal IP address
- Communication via websockets

RWebWindow clients - WAN



- websockets not always work
  - firewall/proxy limitation
  - not supported in fastcgi
  - automatic fall back to long polling (async http)

RWebWindow - multiple clients



- Same window can be displayed multiple times
- Separate connection for each client

Special displays

- Use Chromium Embedded Framework CEF
  see https://bitbucket.org/chromiumembedded/cef
- Create necessary window(s) directly in C++
- Communication via CEF API - no any http/websockets
- Qt5 with QWebEngine – also no any networking

- Made as optional plugins, fully transparent

Batch mode

- Produce images without creating display
- Reuse client and server code as is

- Headless Google Chrome
  - including WebGL support on Linux!
- Headless Mozilla Firefox
  - no WebGL
- Potentially, Node.js
  - with “canvas” module, required extra system libs

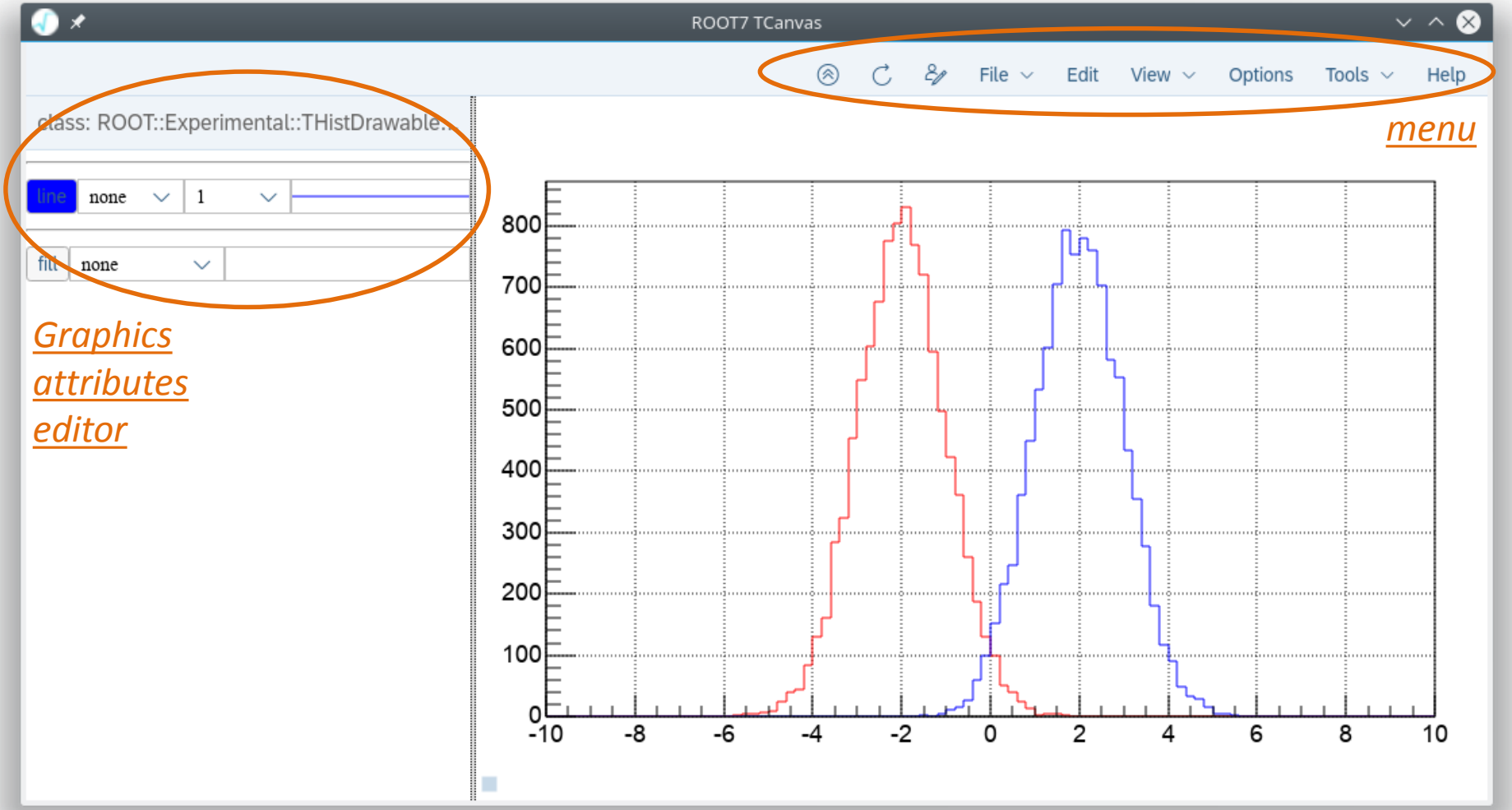
Multi threading

- Default:
  - all RWebWindows runs in main thread
- Optional:
  - special thread for THttpServer
  - dedicated thread for every RWebWindow
  - dedicated thread(s) for clients communication

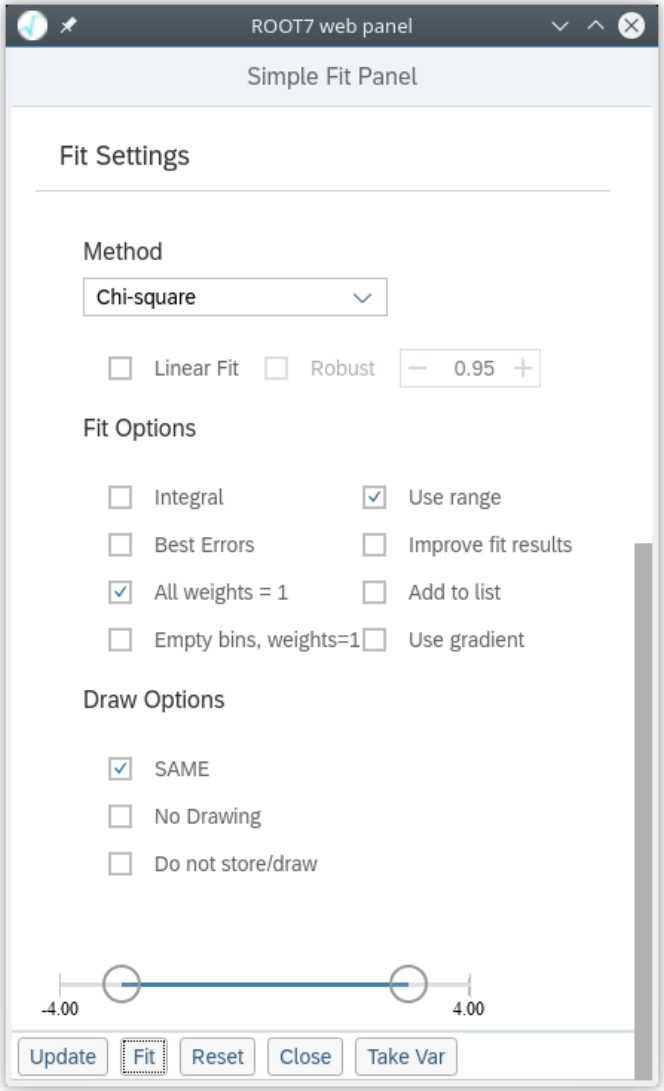
ROOT7 GUI

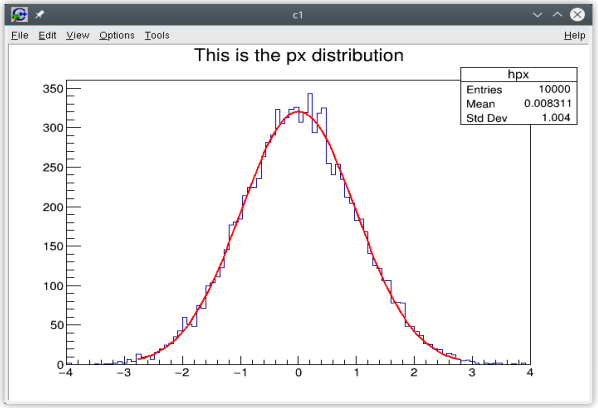
- ROOT7 needs not only graphics
  - RBrowser, RFitPanel, RGEditor, ...
- Library for buttons, checkbox, list, menu, ...
  - SAP OpenUI5 https://openui5.org/
- OpenUI fully supported in RWebWindow
  - any other library can be used

RCanvas – see Olivier talk

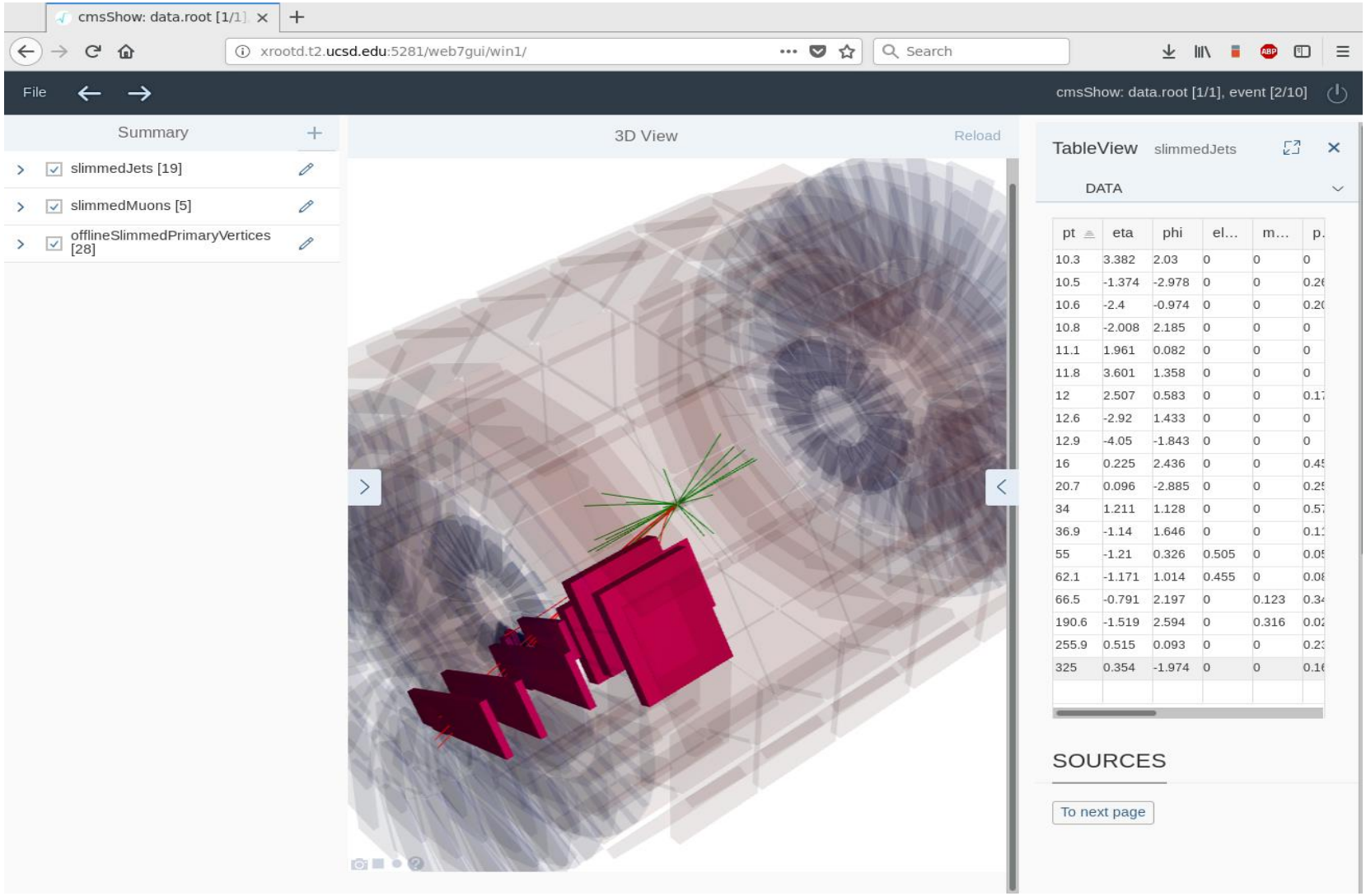


RFitPanel – see Iliana talk





WebEve* prototype



**Alja Mrak-Tadel and Matevz Tadel for CMS*

Plans and ideas

- Authentication and authorization
  - very simple method supported by civetweb
  - support OAuth technology in the future
- RWebWindow offline mode
  - display clients HTML without running ROOT