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

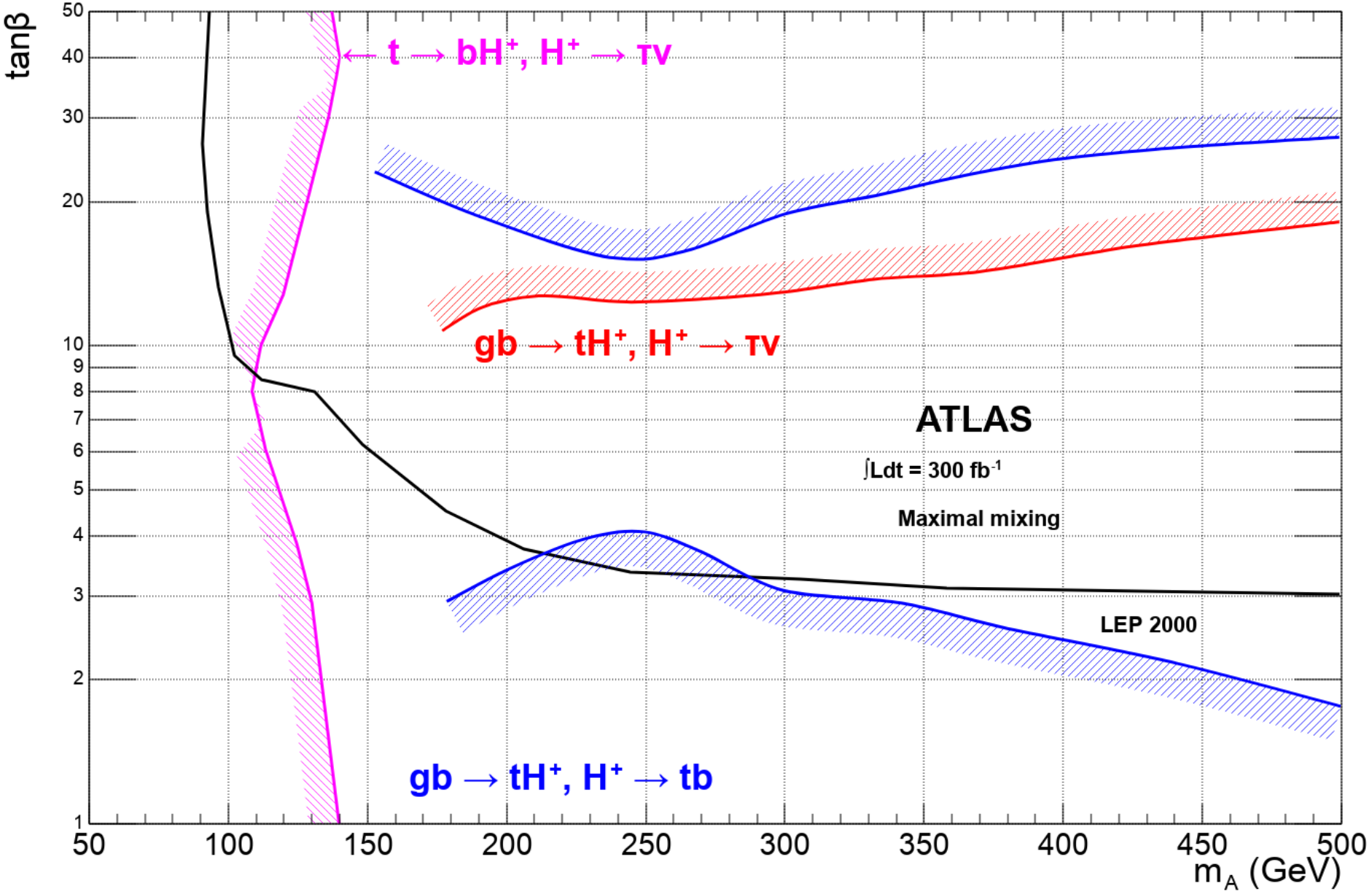
- Significantly simplifies data exchange between C++ server and JavaScript-based clients

JavaScript ROOT

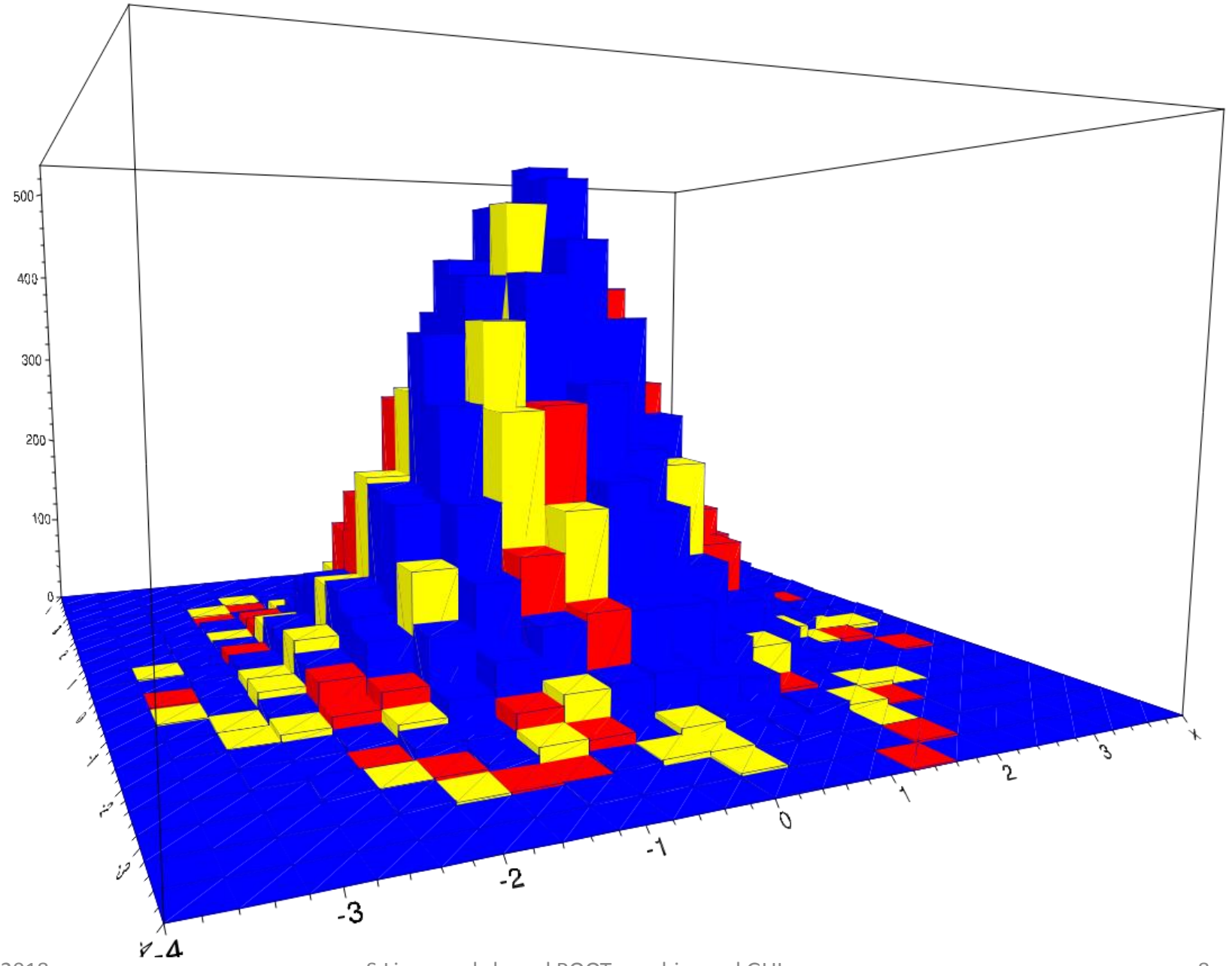
- ROOT objects display in web browsers
- Binary data reading, including TTree
- ROOT JSON format support
- User interface for the THttpServer

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

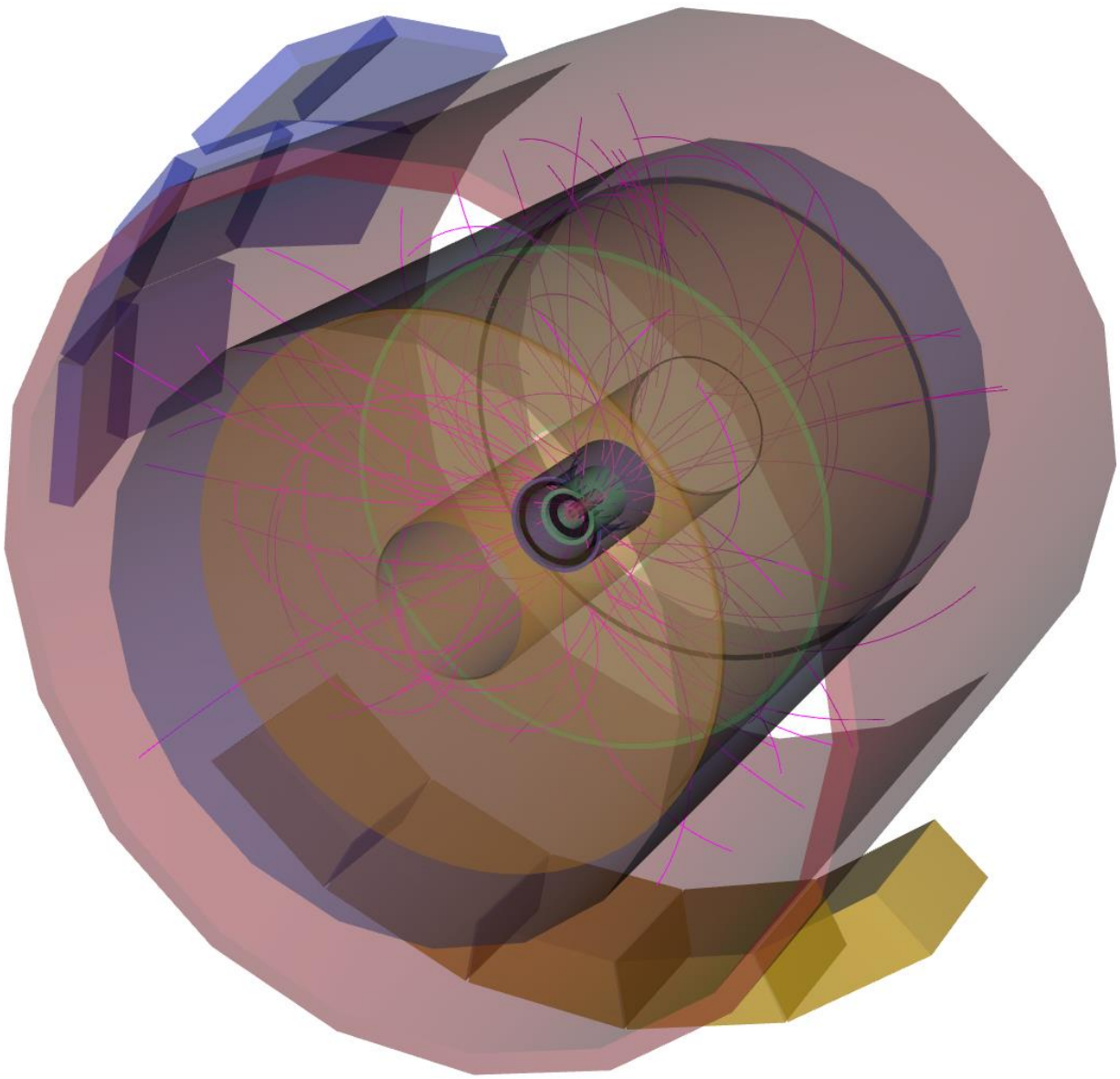
https://root.cern/js/latest/examples.htm#tgraph_excl

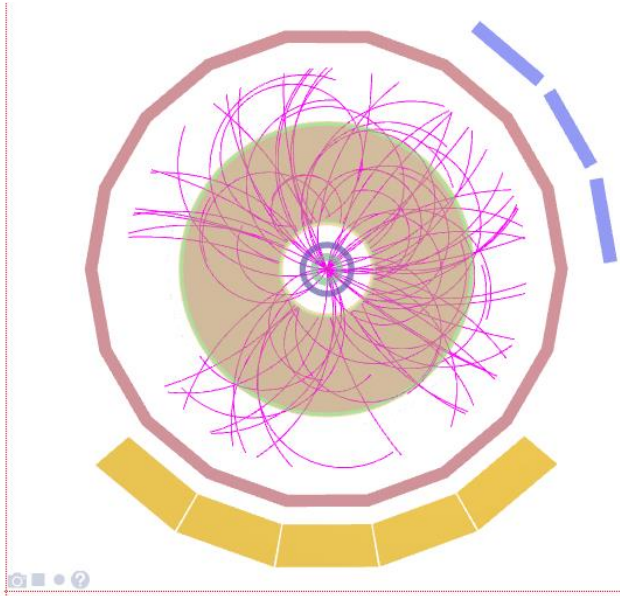


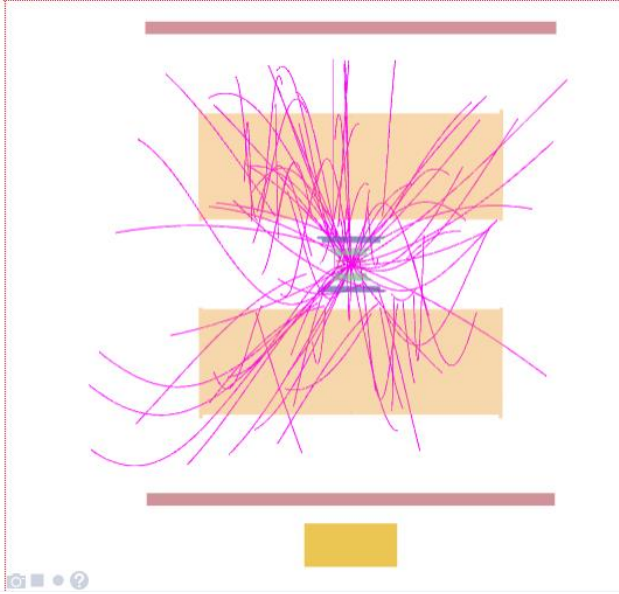
https://root.cern/js/latest/examples.htm#thstack_lego



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







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

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 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

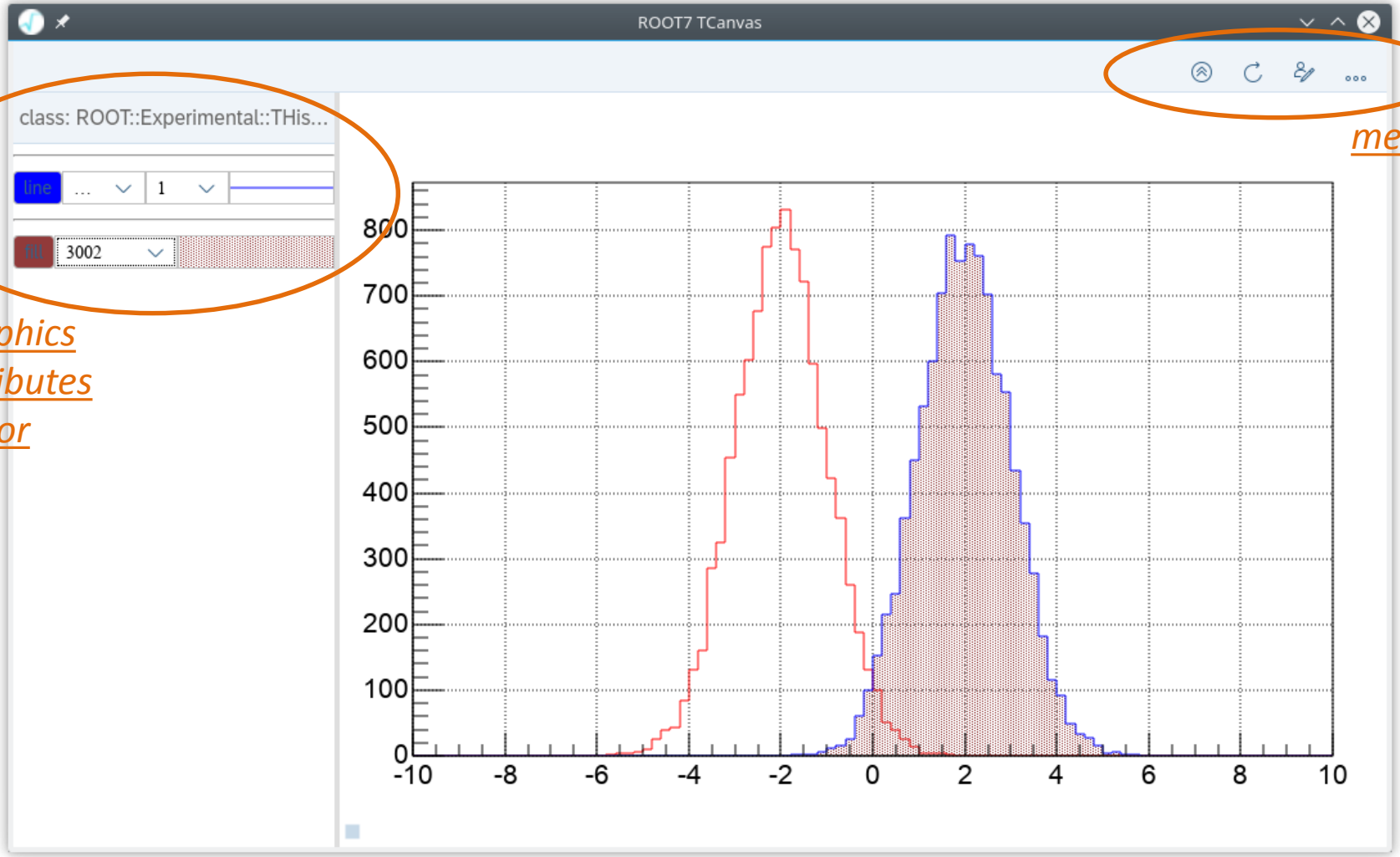
- Central graphics class in ROOT7
- Uses RWebWindow to implement display

- Main idea – decouple data from drawing attributes

Demo macro

```
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



menu

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

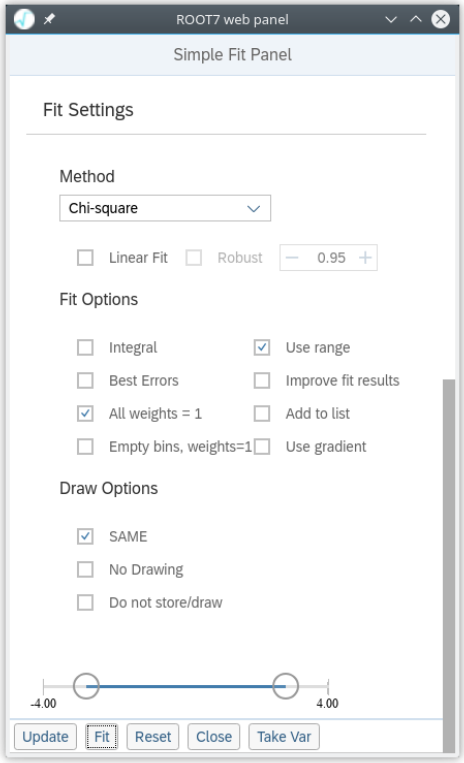
- Client side
  - tooltips
  - bins highlight
  - object context menu
  - zooming
- Client -> Server -> Client
  - change color, line or fill attribute
    - modify request to server
    - server automatically updates all other clients

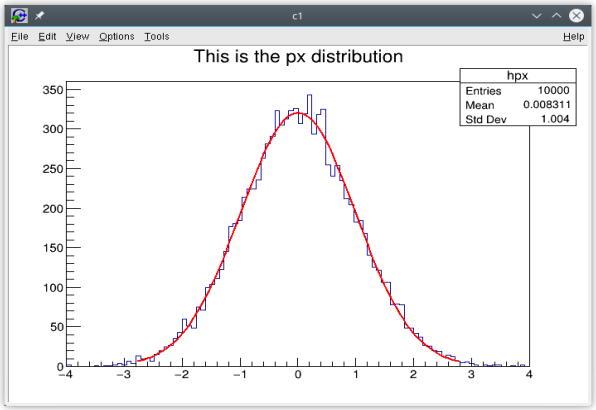
ROOT7 GUI

- 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

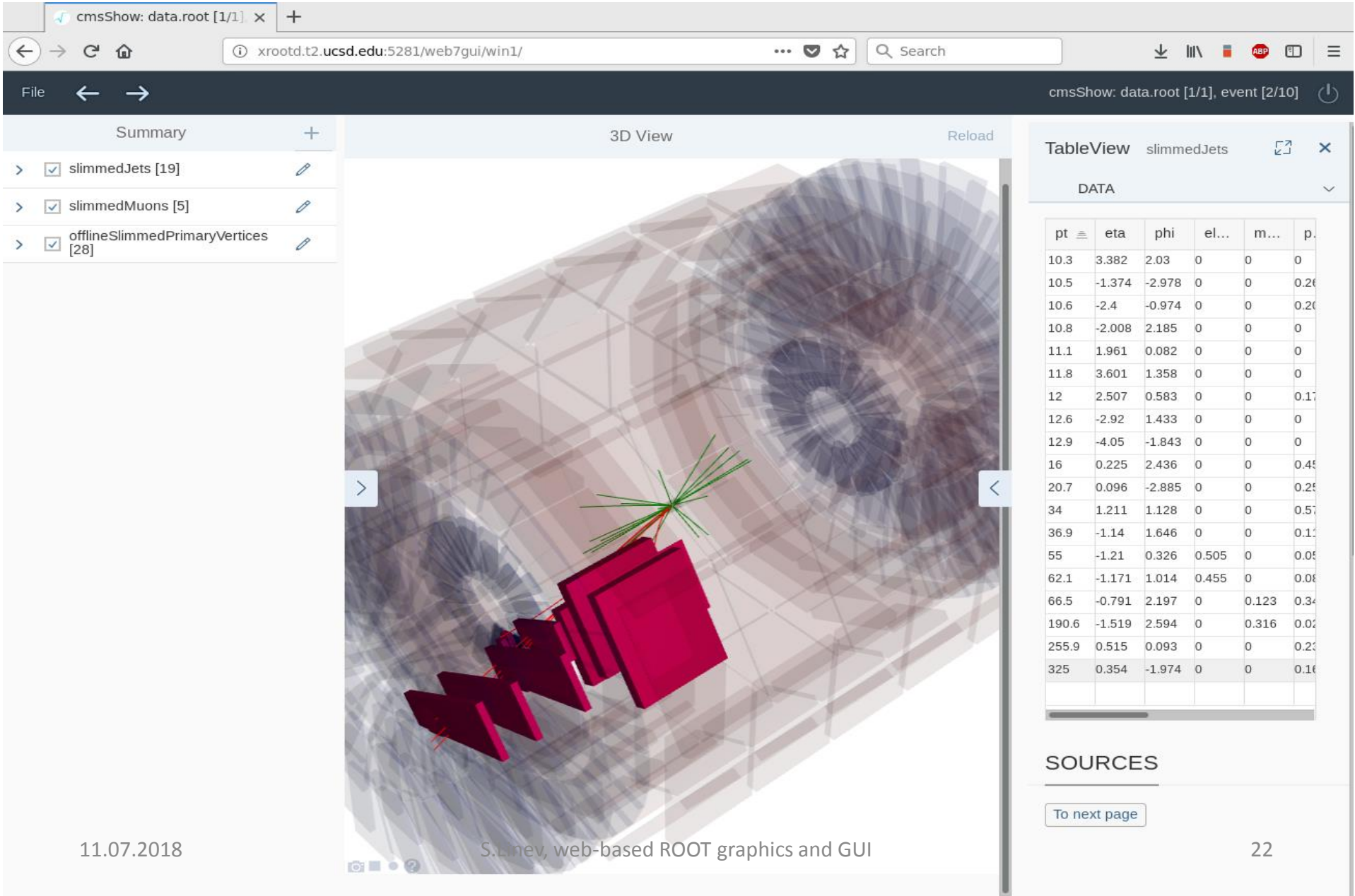
FitPanel

- Inspired from the existing FitPanel layout
- Reimplement with OpenUI5
- Prototype with v6 fitting
  - by Illiana Betsou





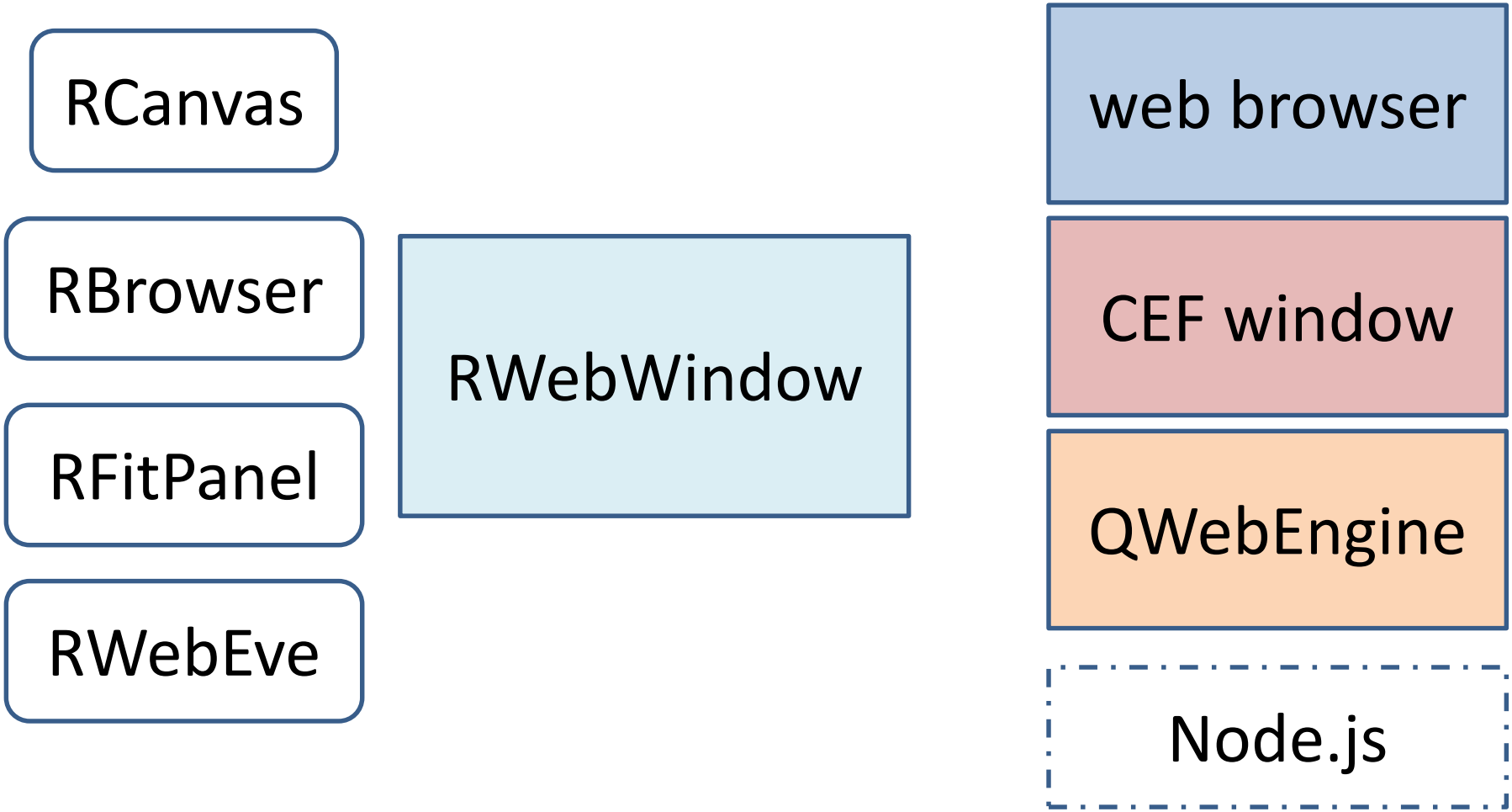
WebEve prototype (see next talk from Alja)



Conclusion

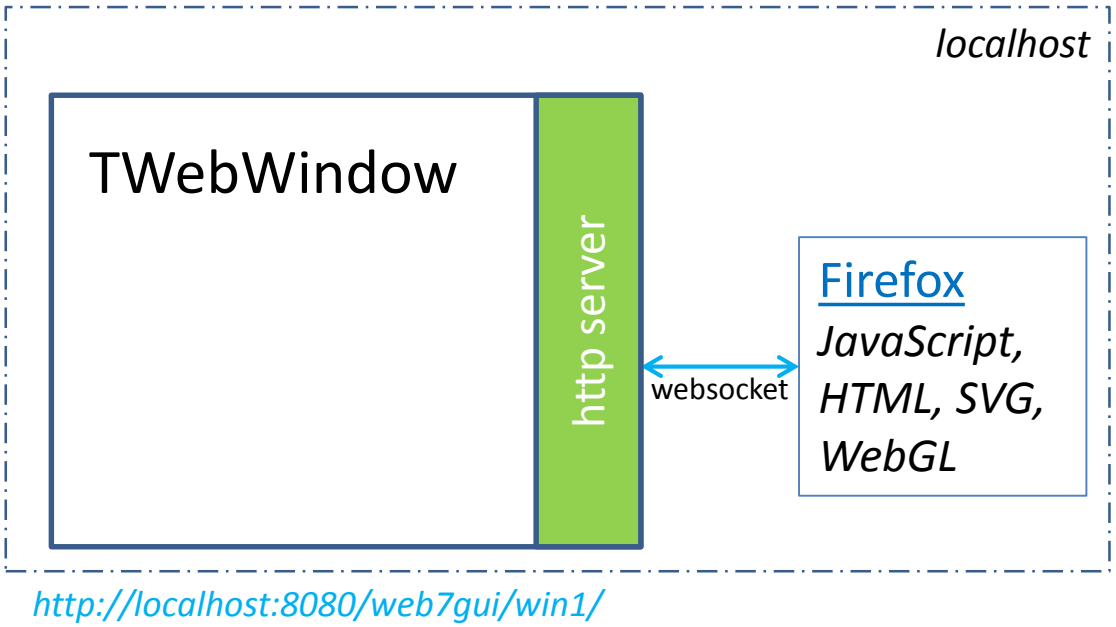
- Full relaunch of graphics and GUI classes
- Use web techniques
- Reuse already existing ROOT components
- Provide more flexibility in implementing user code

ROOT7 – coming soon



Backup slides

Local clients



- Run any browser on the some host
- Use THttpServer bound to loopback address
- Communication via websockets

Local clients

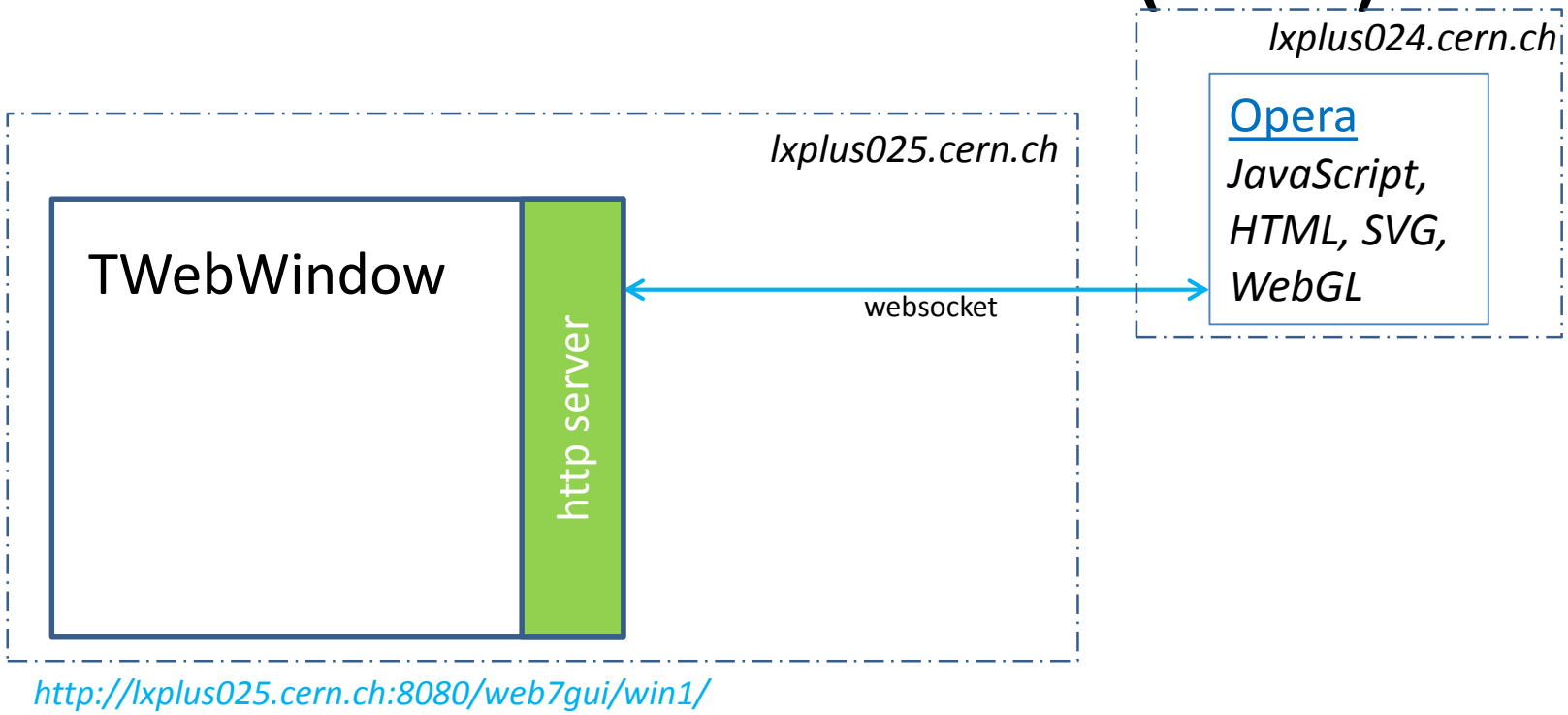
- Use Chromium Embedded Framework CEF
  see https://bitbucket.org/chromiumembedded/cef
- Create necessary window(s) directly in C++
- Communication via CEF API - no any sockets
- Qt5 with QWebEngine – also no any opened sockets

Local display

- Use Google Chrome sources to integrate browser directly into application
  - Chromium Embedded Framework
  - Qt5 WebEngine classes

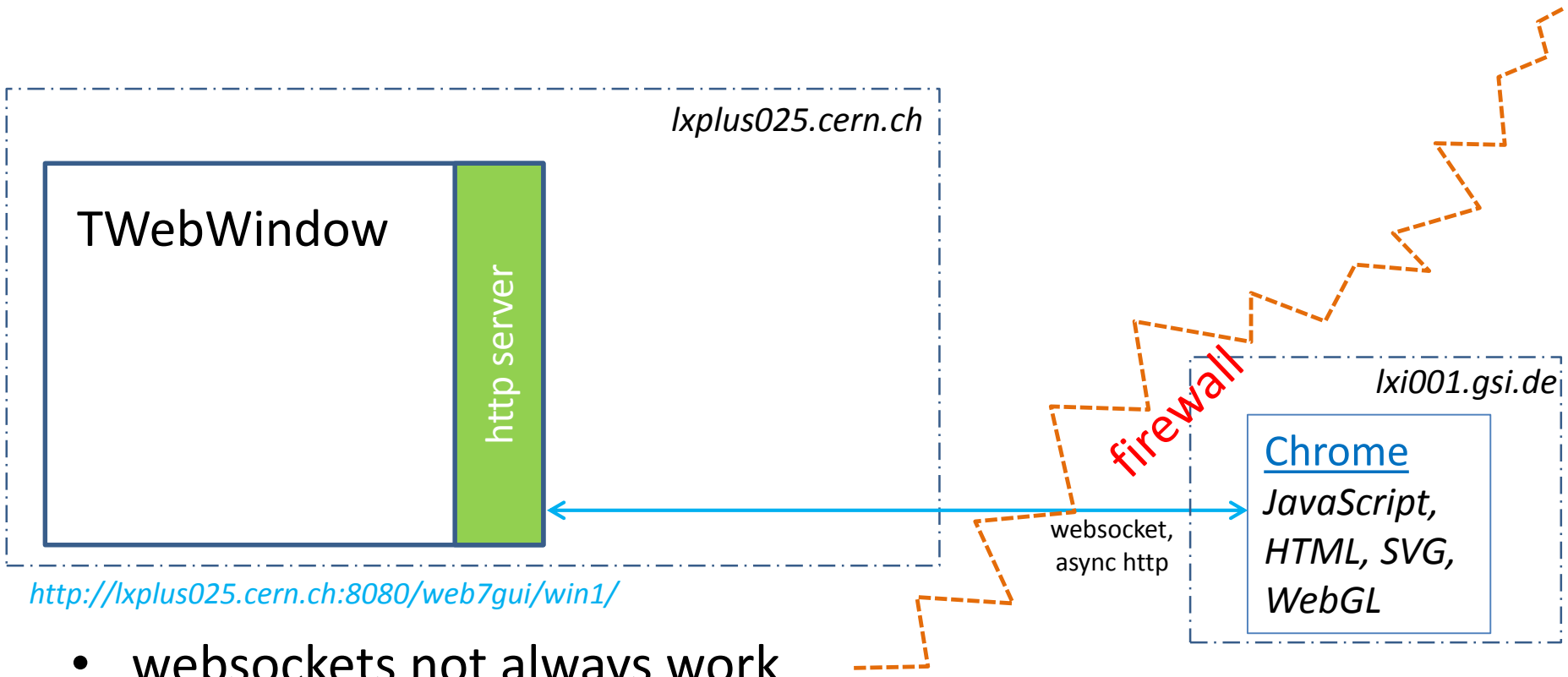
- No any opened http ports
  - all communication inside ROOT application

Remote clients (LAN)



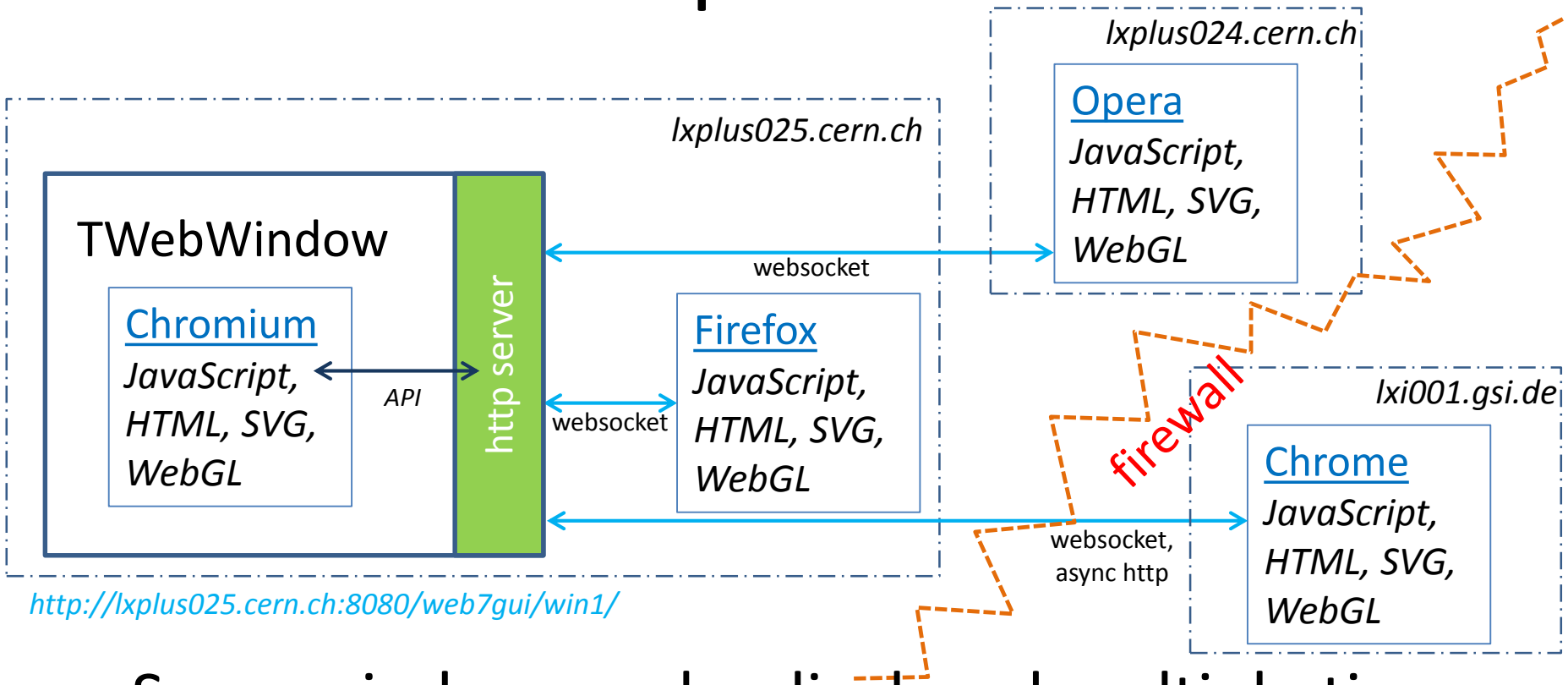
- No any difference with local clients
- Bound THttpServer with normal IP address
- Communication via websockets

Remote clients (WAN)



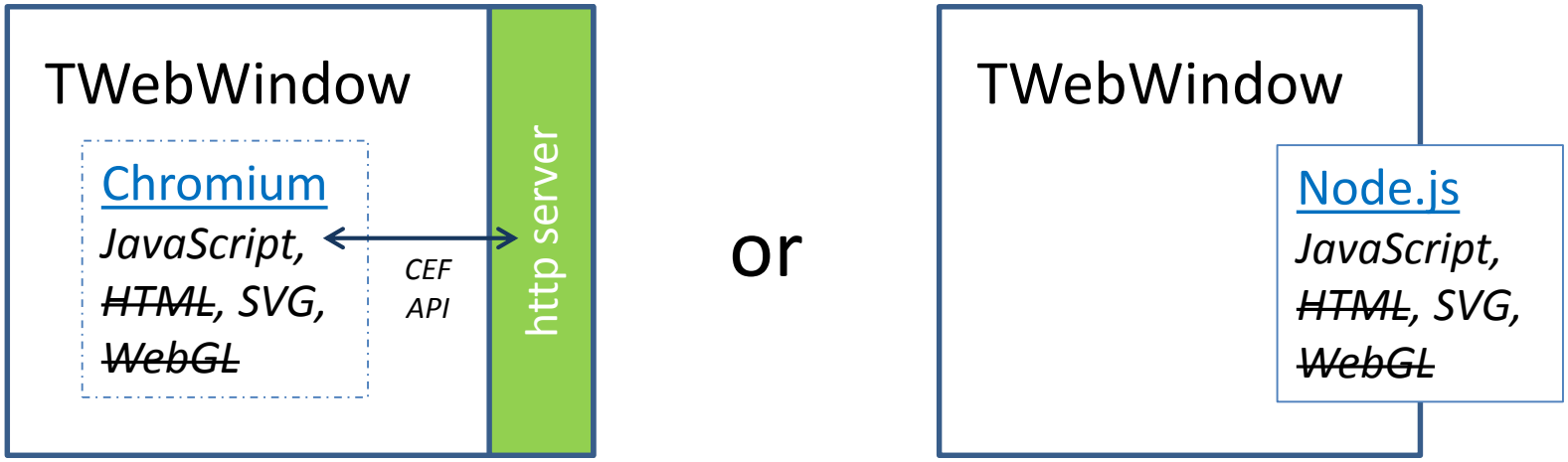
- websockets not always work
  - firewall/proxy limitation
  - not supported in fastcgi
  - automatic fall back to long polling (async http)

Multiple clients



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

Batch mode



- For image production
  - SVG, PS, PNG, GIF, ...
- No any window created
- No any socket opened
- Instead of WebGL use three.js SVG renderer

Batch mode

- Produce images without creating display
- Use same client and server code as for interactive display
- Supported by Chrome on Linux
  - partially by CEF and QWebEngine
  - workaround with Xvfb (X server emulation)
- Probably, use with Node.js

TBufferJSON example

- Writing:

```
auto ptr = new UserClass;
TString json = TBufferJSON::ToJSON(ptr);
```

transfer data to JavaScript client

```
var obj = JSROOT.parse(json);
```

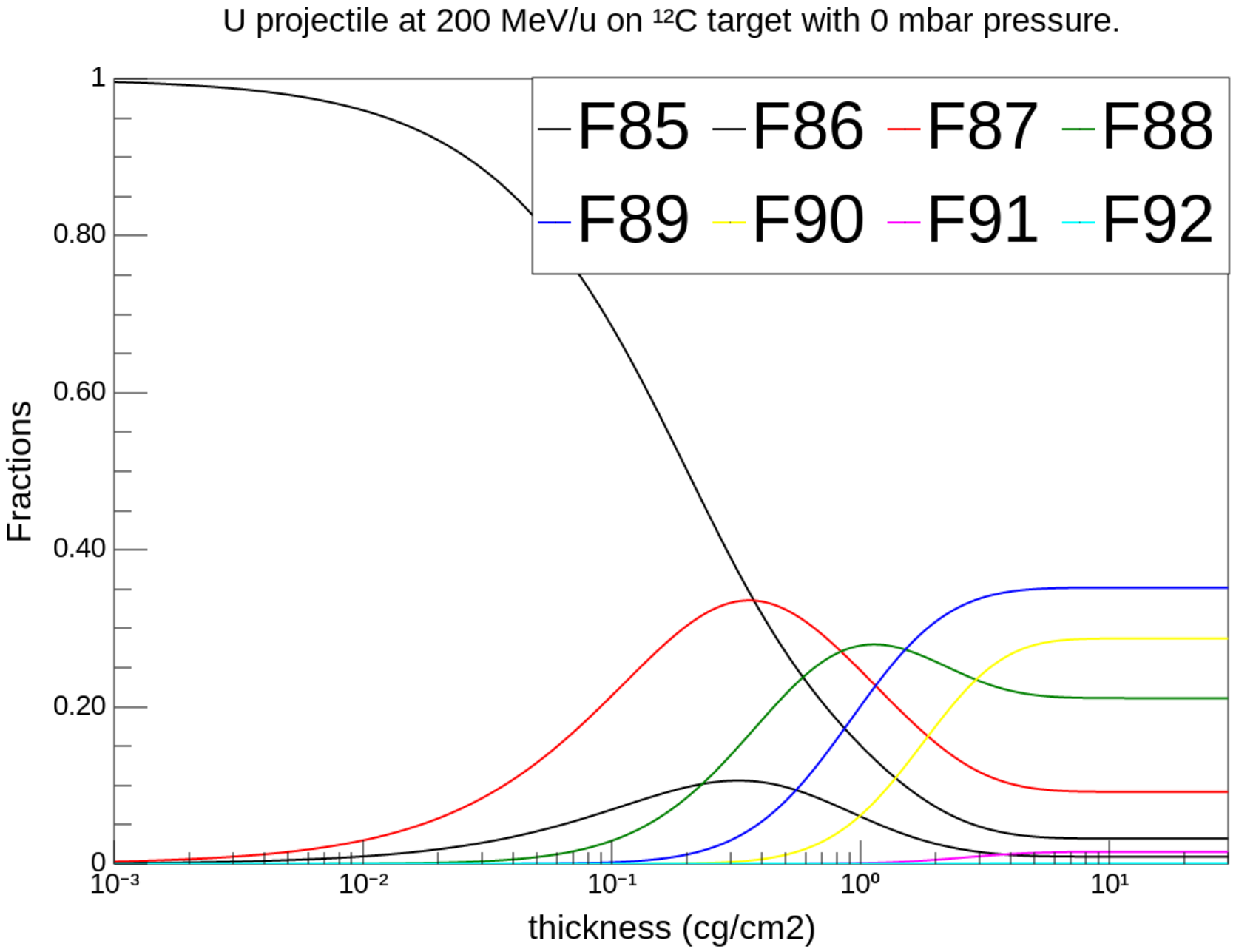
- Reading:

```
var json = JSROOT.toJSON(obj);
```

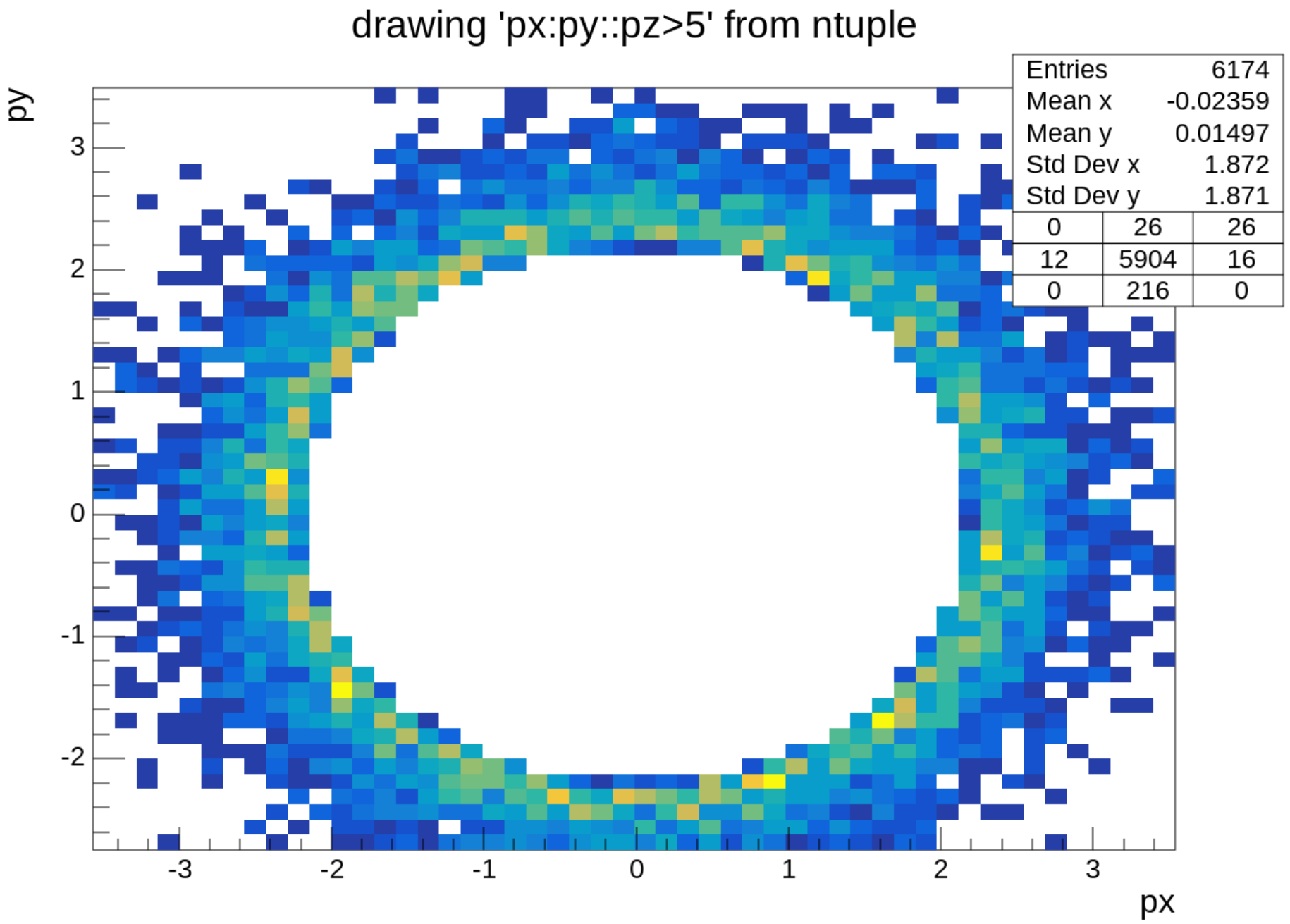
transfer data from JavaScript client

```
UserClass *ptr = nullptr;
TBufferJSON::FromJSON(ptr, json);
```

https://root.cern/js/latest/examples.htm#tf1_ff2



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



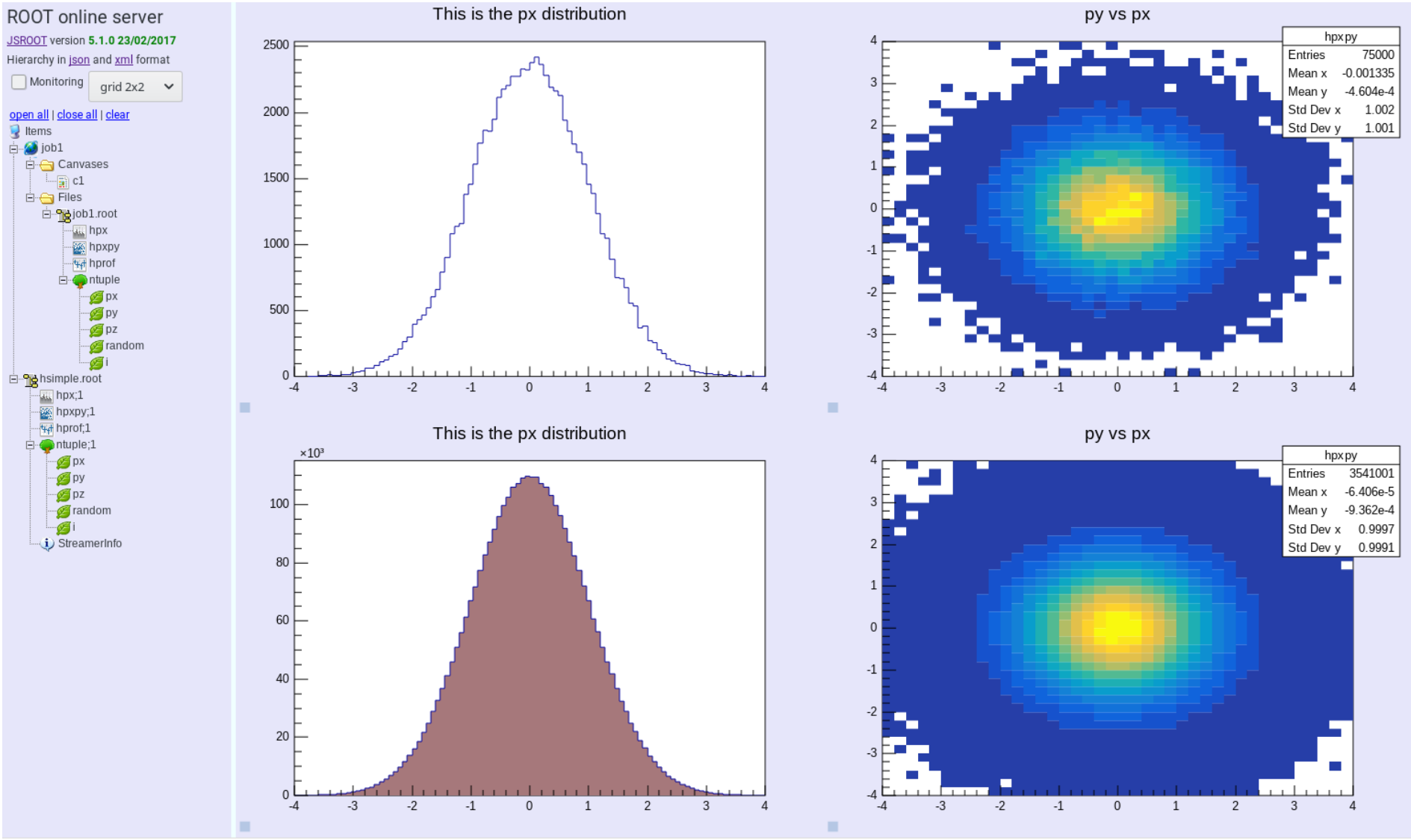
Simple API

- Few lines of JavaScript code to insert JSROOT graphics on any HTML page

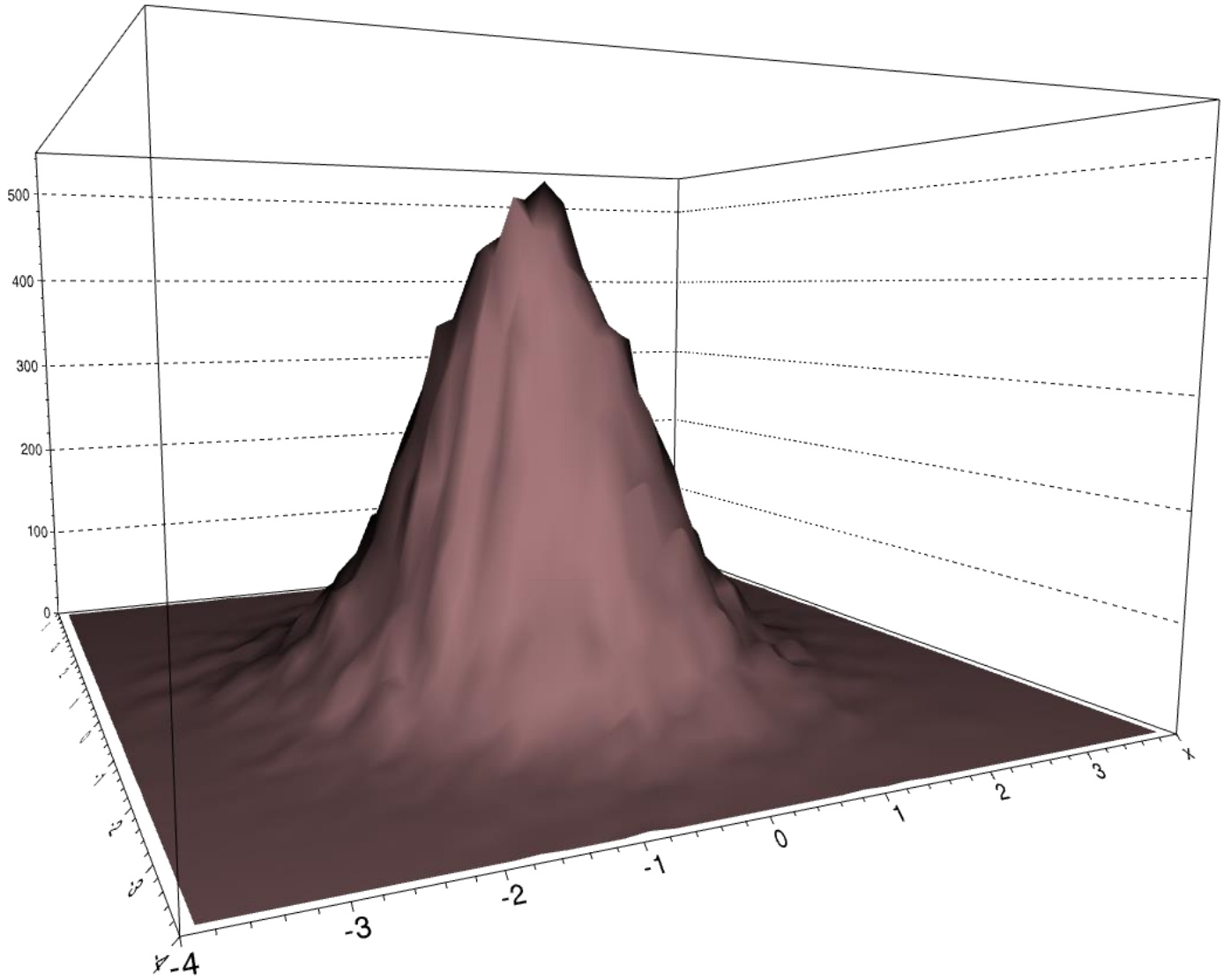
```
<script src="https://root.cern/js/latest/scripts/JSRootCore.min.js"
        type="text/javascript"></script>

<script type='text/javascript'>
    JSROOT.NewHttpRequest("hpx.json", 'object', function(obj) {
       JSROOT.draw("drawing", obj, "hist");
    }).send();
</script>


<div id="drawing" style="width:800px; height:600px"></div>
```

- Examples https://root.cern/js/latest/api.htm

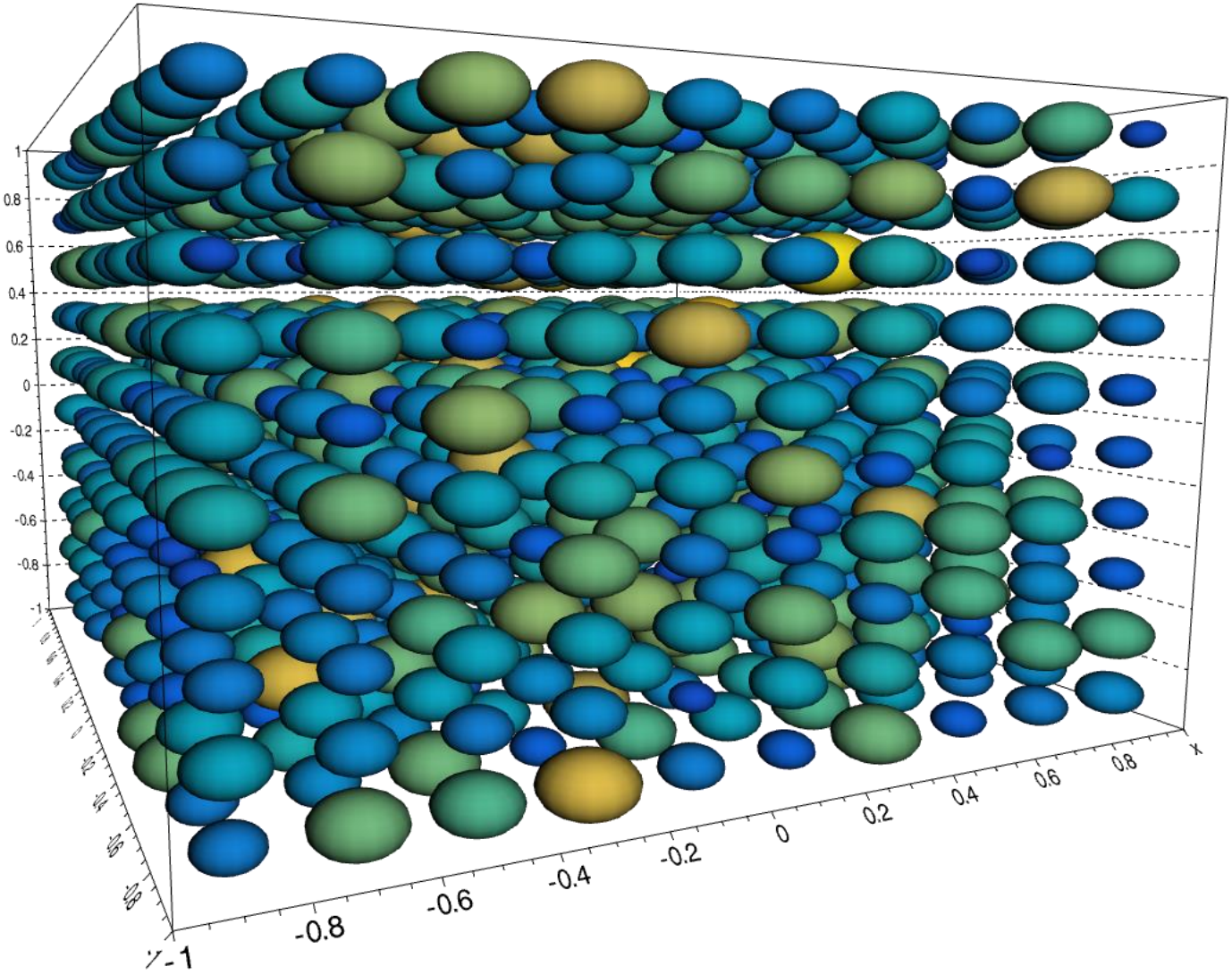
https://root.cern/js/latest/api.htm#httpserver_c_read_file



https://root.cern/js/latest/examples.htm#th2_surf4



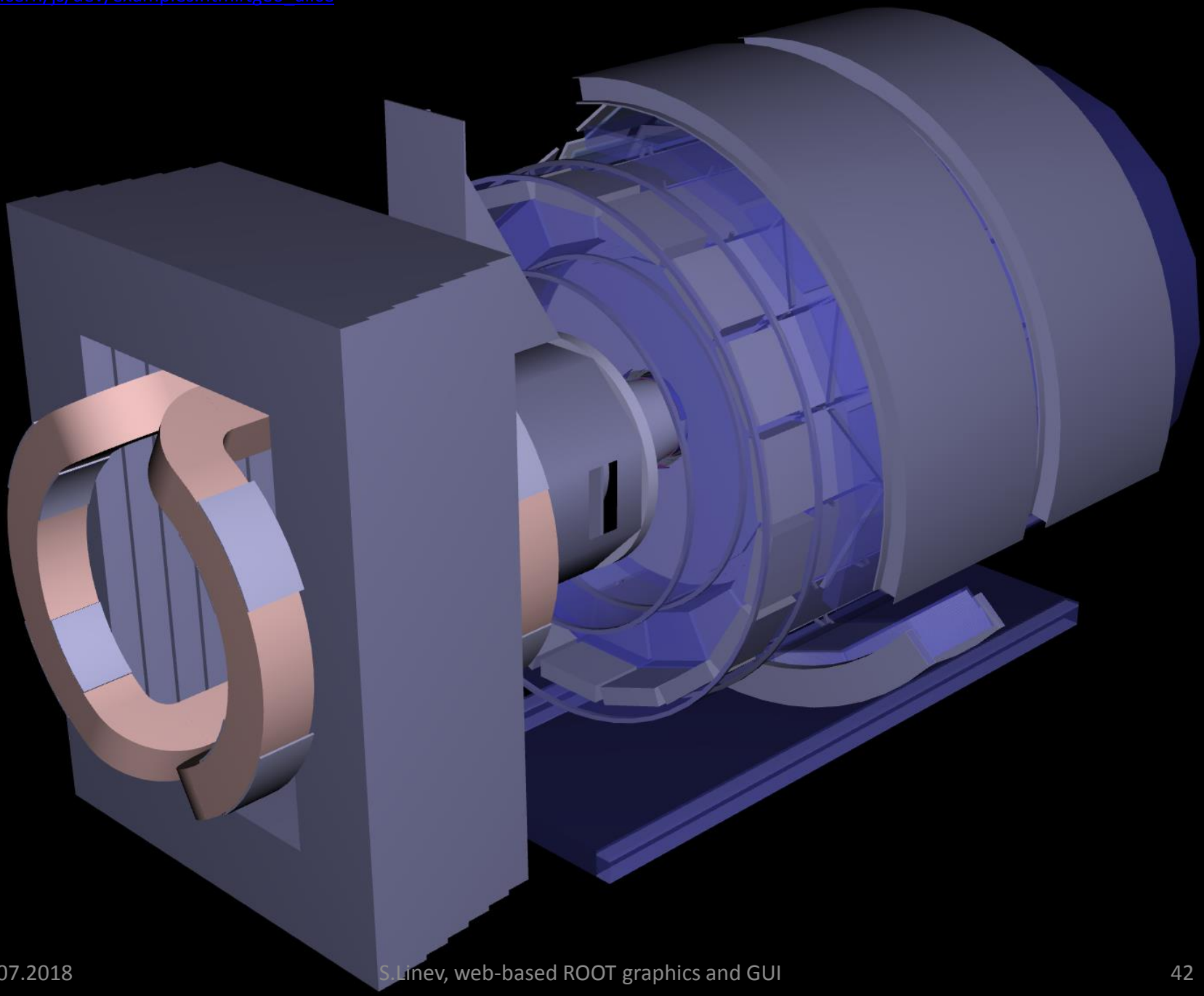
https://root.cern/js/latest/examples.htm#th3_glbox2



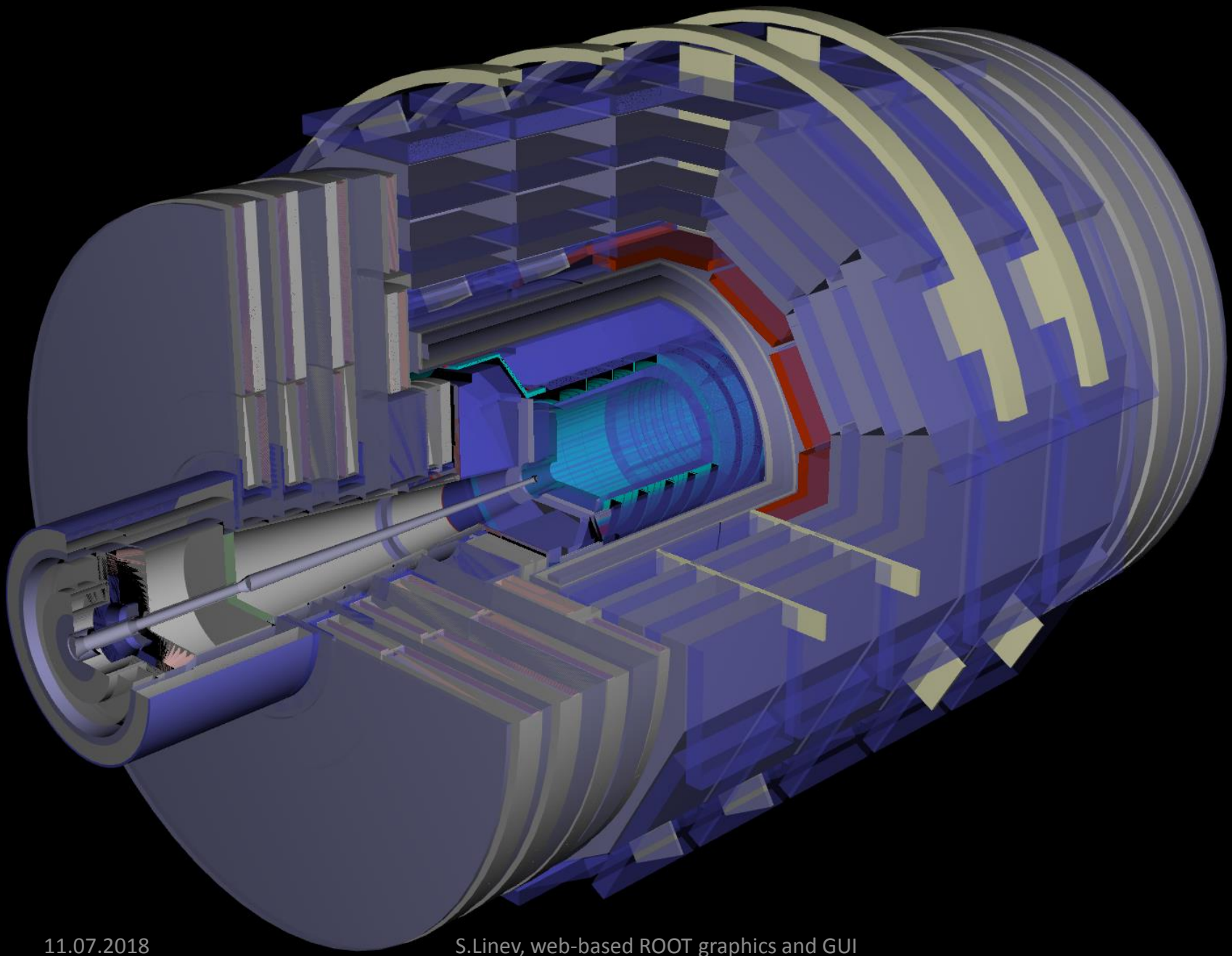
Geometry display

- three.js was there – just use it
- All kind of ROOT TGeo classes
- All kinds of shapes
  - composite with enhanced ThreeCSG.js
- THREE.BufferGeometry
- Interactive:
  - rotation and zoom
  - volumes highlight and tooltip
  - context menu
  - clip panels

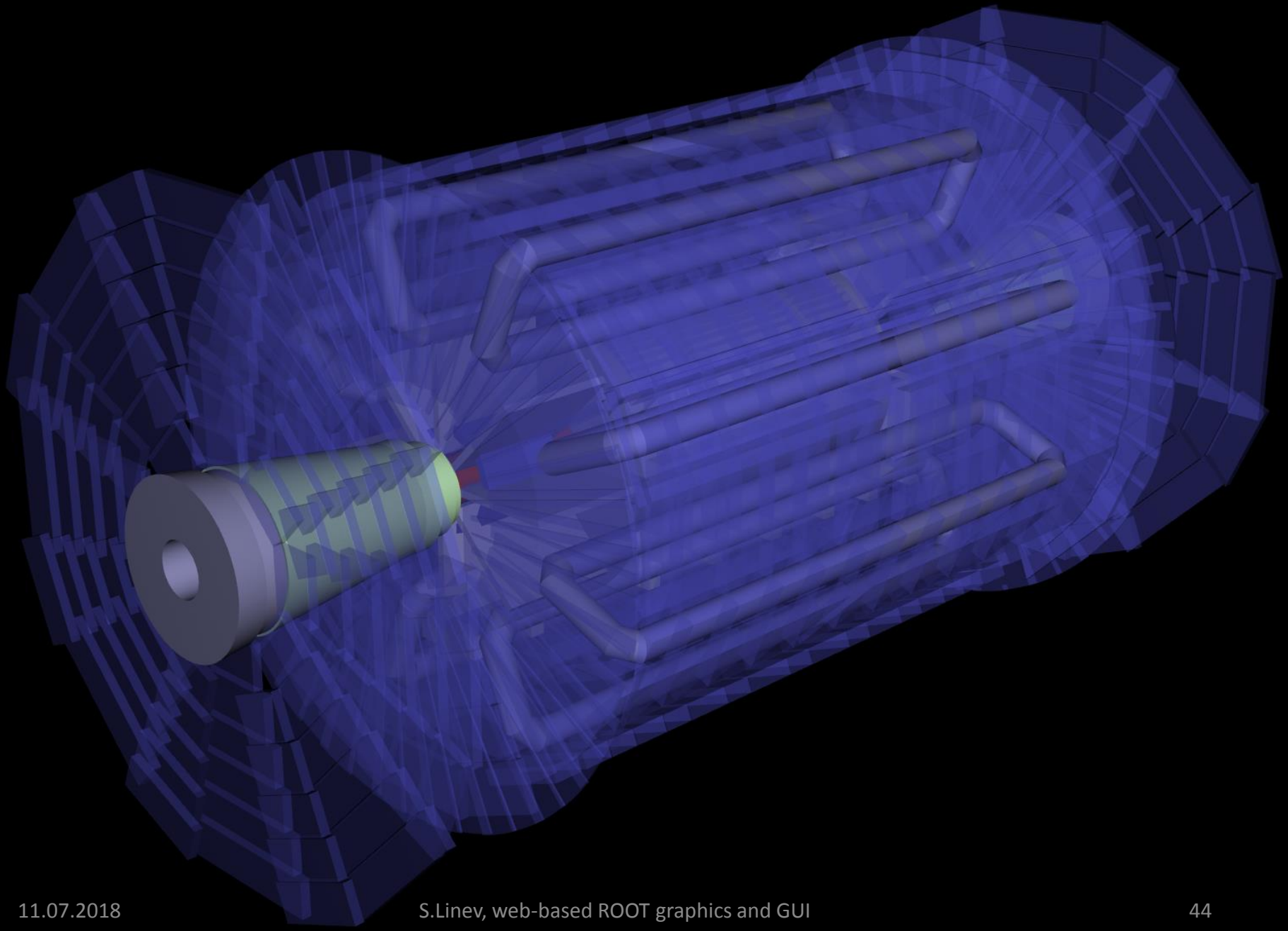
https://root.cern/js/dev/examples.htm#tgeo_alice



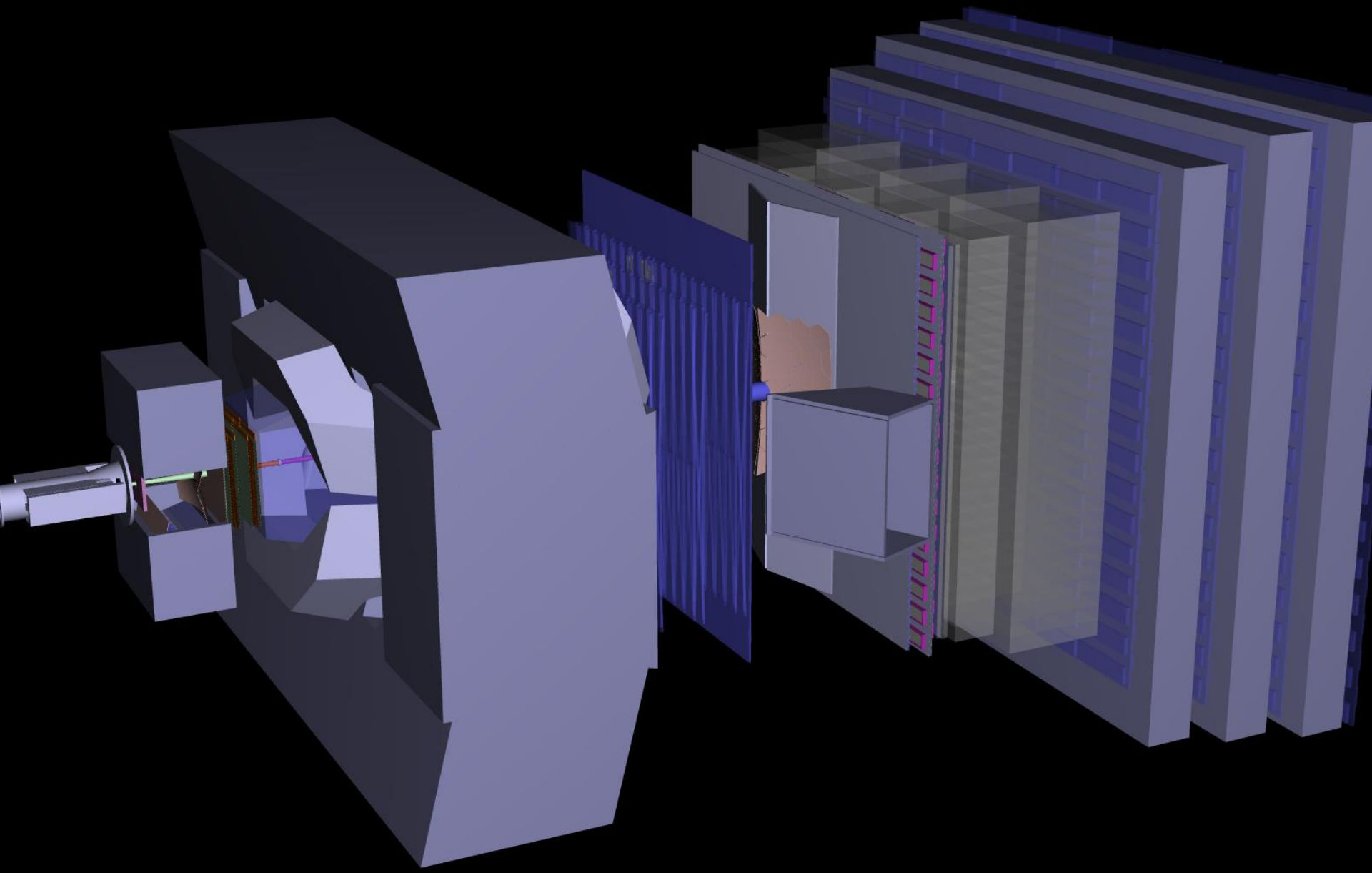
https://root.cern/js/dev/examples.htm#tgeo_cms



https://root.cern/js/dev/examples.htm#tgeo_atlas



https://root.cern/js/dev/examples.htm#tgeo_lhcb



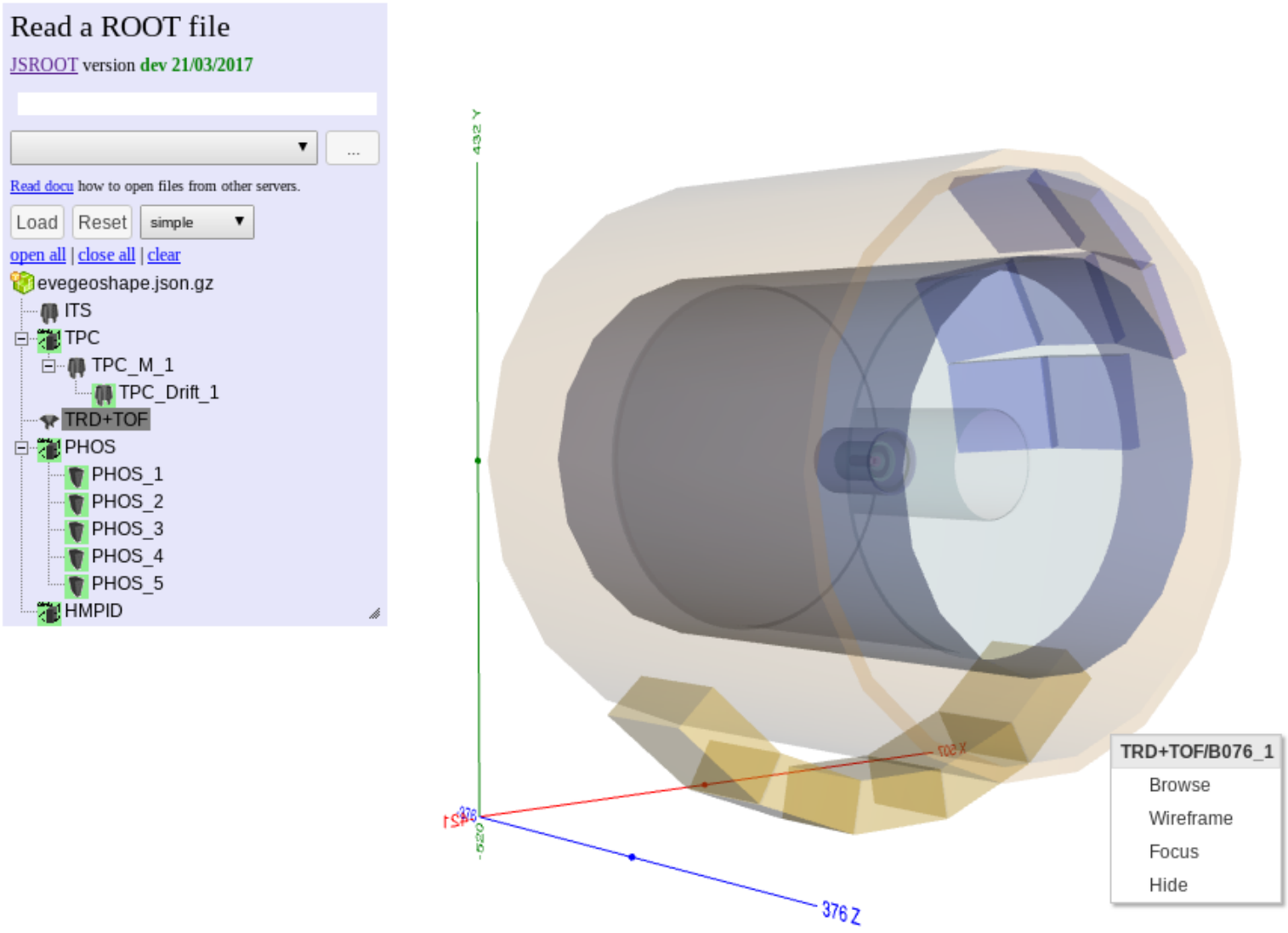
Behind the scene

- Select most significant visible volumes
  - try limit model by 10^5 faces
  - take into account view frustum (optionally)
  - create compact model descriptor
  - takes less then 0.5s for large geometries

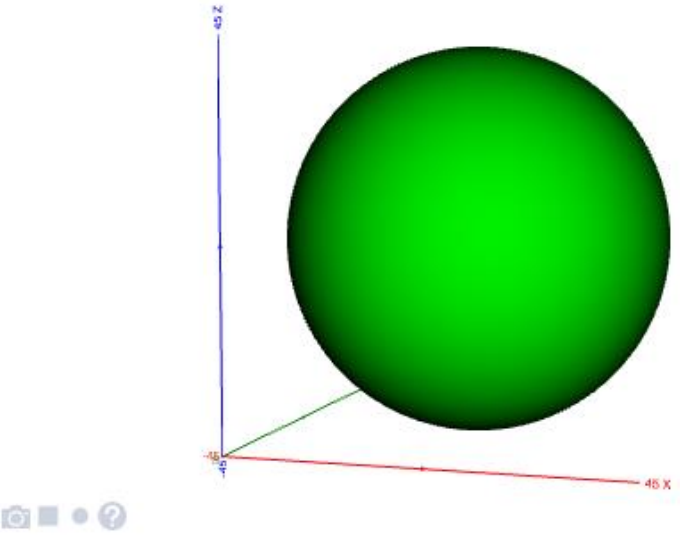
- Build three.js model
  - reproduce TGeo hierarchy in THREE.Object3D
  - create BufferGeometry for each unique shape
    - reuse build geometries when possible
  - can be offload to HTML5 Worker
  - ~3s for typical LHC geometry

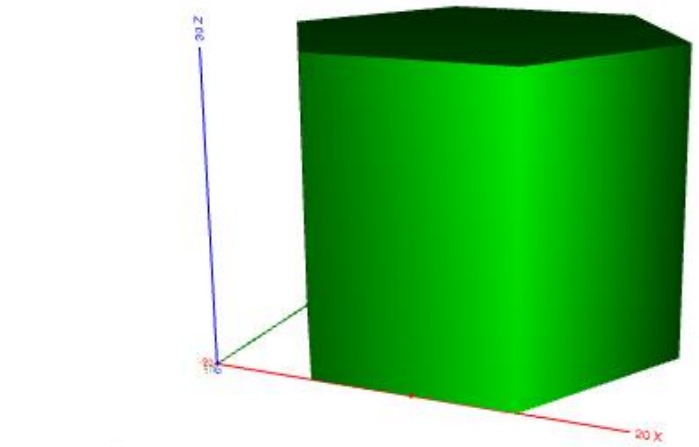
- three.js model can be used without JSROOT display
  - https://root.cern/js/dev/api.htm#custom_html_geometry

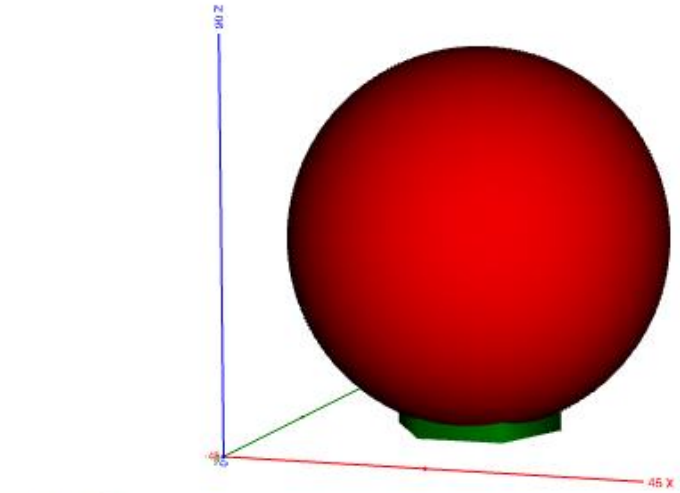
https://root.cern/js/dev/examples.htm#tgeo_eve

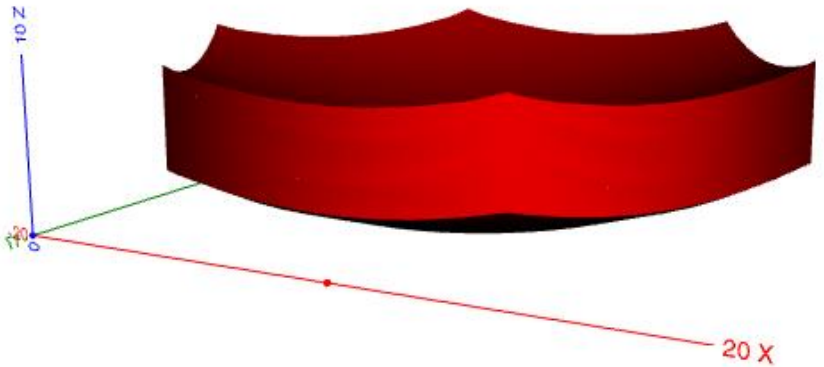


https://root.cern/js/dev/examples.htm#tgeo_comp

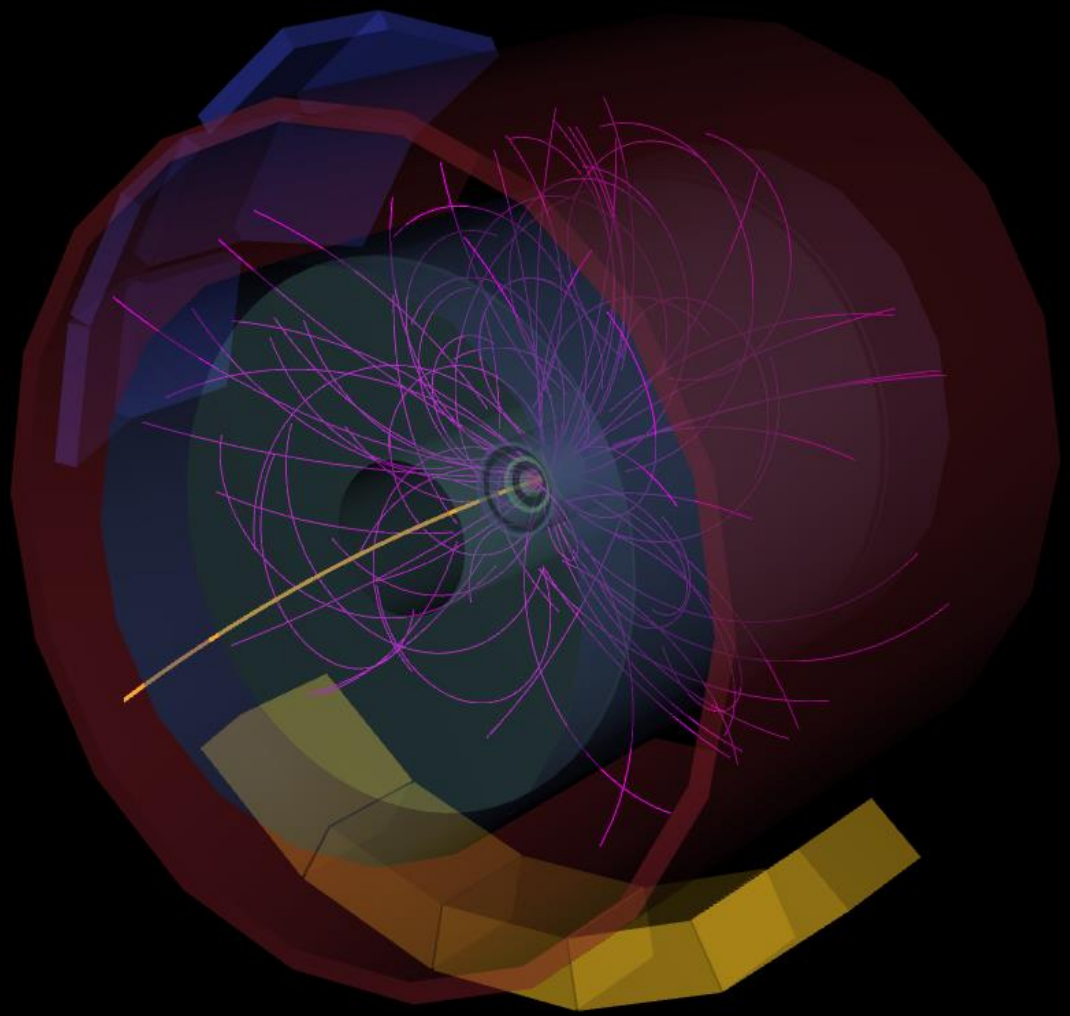


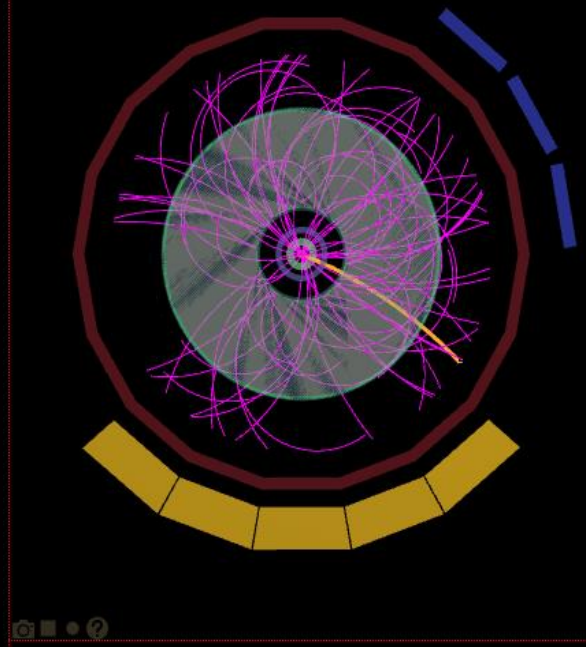


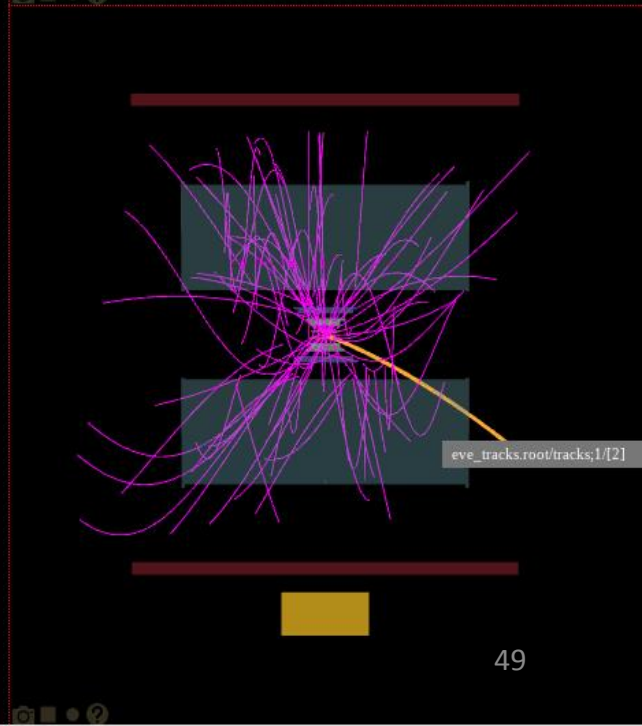




https://root.cern/js/latest/examples.htm#tgeo_proj

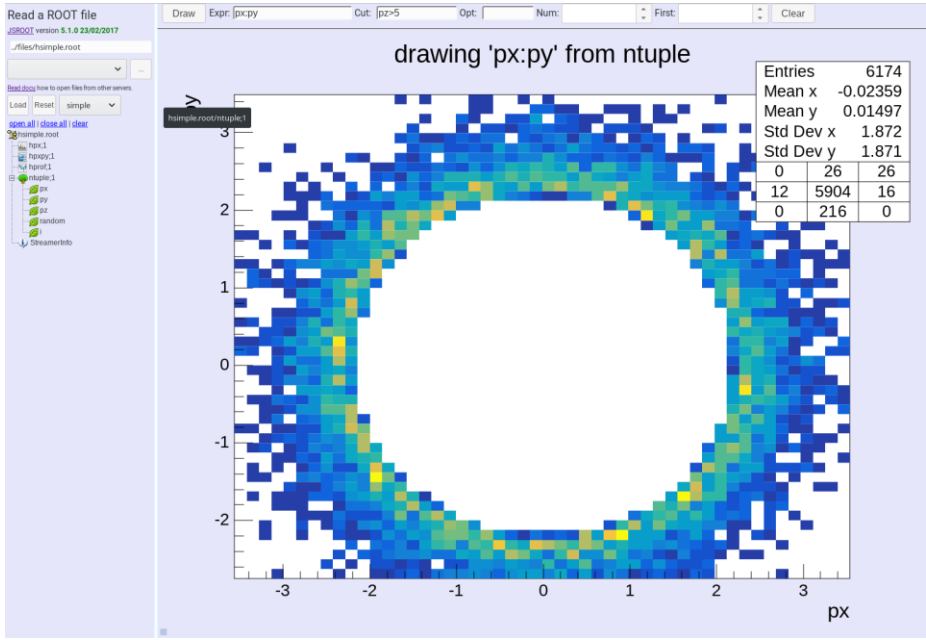






TTree::Draw

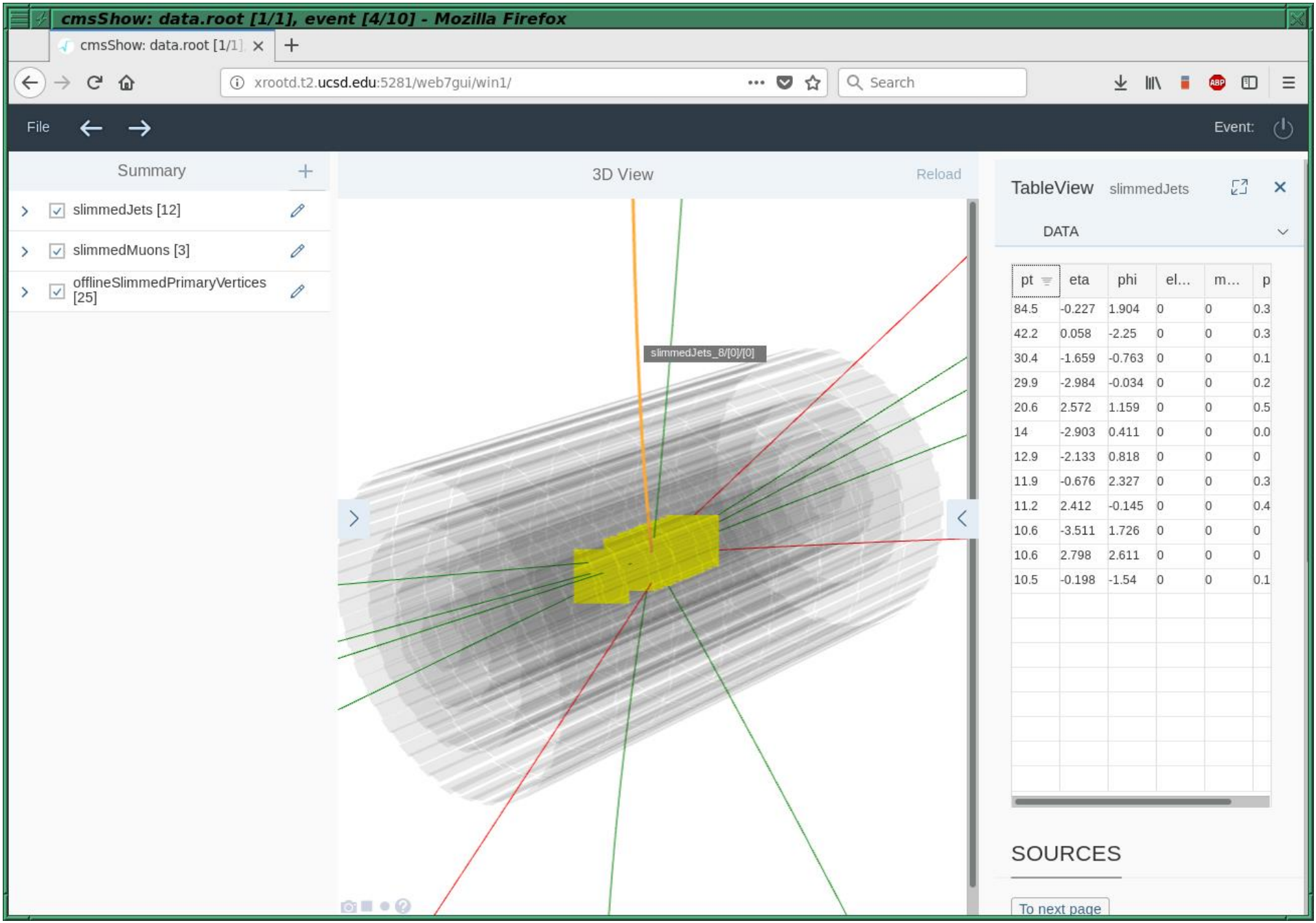
- Access to data directly from the browser
  - no need for MakeClass or MakeSelector
- All branches types are supported
  - including splitted STL containers
  - and old TBranchObject
- Fast – multi-range HTTP requests
- Direct dump of branch data
- Complex TTree::Draw syntax
  - expressions
  - cut condition
  - arrays indexes
  - Math functions
  - Class functions
  - histogram parameters
- TSelector-like user API



ROOT7 WebEve* for CMS

- CMS event display is a based on ROOT. The software will continue to depend on ROOT and therefore ROOT7 in the future
- Intent to transform existing event display where current CMS code provides EVE objects and write a web client for user interaction
- Initial exploration was done by passing over TEveElements for 3D scene and tables. The demo has limited functionality: drawing only hits, geo, and tracks.
- Closed exploration event display within CMS and move to eve7 development to define protocol to communicate changes between server and client, implement synchronized selection state between multiple views, and rendering of complex physics objects like calorimeters and jets

**Alja Mrak-Tadel and Matevz Tadel*


cmsShow: data.root [1/1], event [4/10] - Mozilla Firefox
cmsShow: data.root [1/1]
xrootd.t2.ucsd.edu:5281/web7gui/win1/
Search
File
Event:
Summary
slimmedJets [12]
slimmedMuons [3]
offlineSlimmedPrimaryVertices [25]
3D View
Reload
slimmedJets_8/[0]/[0]
TableView slimmedJets
DATA
pt eta phi el... m... p
84.5 -0.227 1.904 0 0 0.3
42.2 0.058 -2.25 0 0 0.3
30.4 -1.659 -0.763 0 0 0.1
29.9 -2.984 -0.034 0 0 0.2
20.6 2.572 1.159 0 0 0.5
14 -2.903 0.411 0 0 0.0
12.9 -2.133 0.818 0 0 0
11.9 -0.676 2.327 0 0 0.3
11.2 2.412 -0.145 0 0 0.4
10.6 -3.511 1.726 0 0 0
10.6 2.798 2.611 0 0 0
10.5 -0.198 -1.54 0 0 0.1
SOURCES
To next page

Behind the scene

- Drawable
  - flat list in RCanvas
  - reference data object
  - includes graphical attributes
- Display item
  - temporary data container send to client
  - includes data for painting
  - may or may not include original data
- Painter
  - JavaScript code on client side, reuse JSROOT as much as possible
  - pure C++ painter will be supported through TVirtualX-like interface
- Version control
  - account changes in drawables
  - resend only changed items to client
  - let minimize traffic to client