ROOT I/O in JavaScript

Bertrand Bellenot[1] and Sergey Linev[2]

[1]CERN, PH-SFT, Geneva, Switzerland
[2]GSI, CS-EE, Darmstadt, Germany

E-mail: bertrand.bellenot@cern.ch

Abstract. In order to be able to browse (inspect) ROOT files in a platform independent way, a JavaScript version of the ROOT I/O subsystem has been developed. This allows the content of ROOT files to be displayed in most available web browsers, without having to install ROOT or any other software on the server or on the client. This gives a direct access to ROOT files from any new device in a lightweight way. It is possible to display simple graphical objects such as histograms and graphs (TH1, TH2, TH3, TProfile, and TGraph). The rendering of 1D/2D histograms and graphs is done with an external JavaScript library (D3.js), and another library (Three.js) is used for 2D and 3D histograms. We will describe the techniques used to display the content of a ROOT file, with a rendering being now very close to the one provided by ROOT.

1. Introduction

ROOT [3] is used by almost all experiments throughout high energy and nuclear physics to write, read and analyze data. In order to allow browsing (inspecting) ROOT files on the web, a JavaScript version of the ROOT I/O subsystem has been developed, JSRootIO. As the I/O and communication mechanisms have already been presented [4], this document will focus on two major developments, namely the new graphic subsystem, where most of the recent developments took place, in particular for 3D visualization, and a new approach especially designed for online monitoring that shows how to monitor a running ROOT application.

2. The graphic subsystem

A JavaScript library, *D3.js* [5], is used to display two dimensional graphics, such as histograms and graphs. It is released under the BSD License [6]. Another JavaScript library, *Three.js* [7], is used to display three dimensional graphics, like 2D histograms in “LEGO” rendering mode, and 3D histograms. It is released under the MIT License [8].

The *D3.js* library is using the HTML, SVG [9] and CSS web standards, is very fast, and supports dynamic behaviour for interaction and animation. With the *Three.js* library one can create cameras, objects, lights, materials and more, and there is a choice of renderers, namely HTML5's canvas [10], WebGL [11] or SVG.

Figure 1 shows an example of different RooFit [12] plot styles in a canvas. It was created with one of the standard RooFit tutorial shipped with ROOT, *rf107_plotstyles.C*, then saved in a ROOT file, and displayed on the web using JSRootIO.

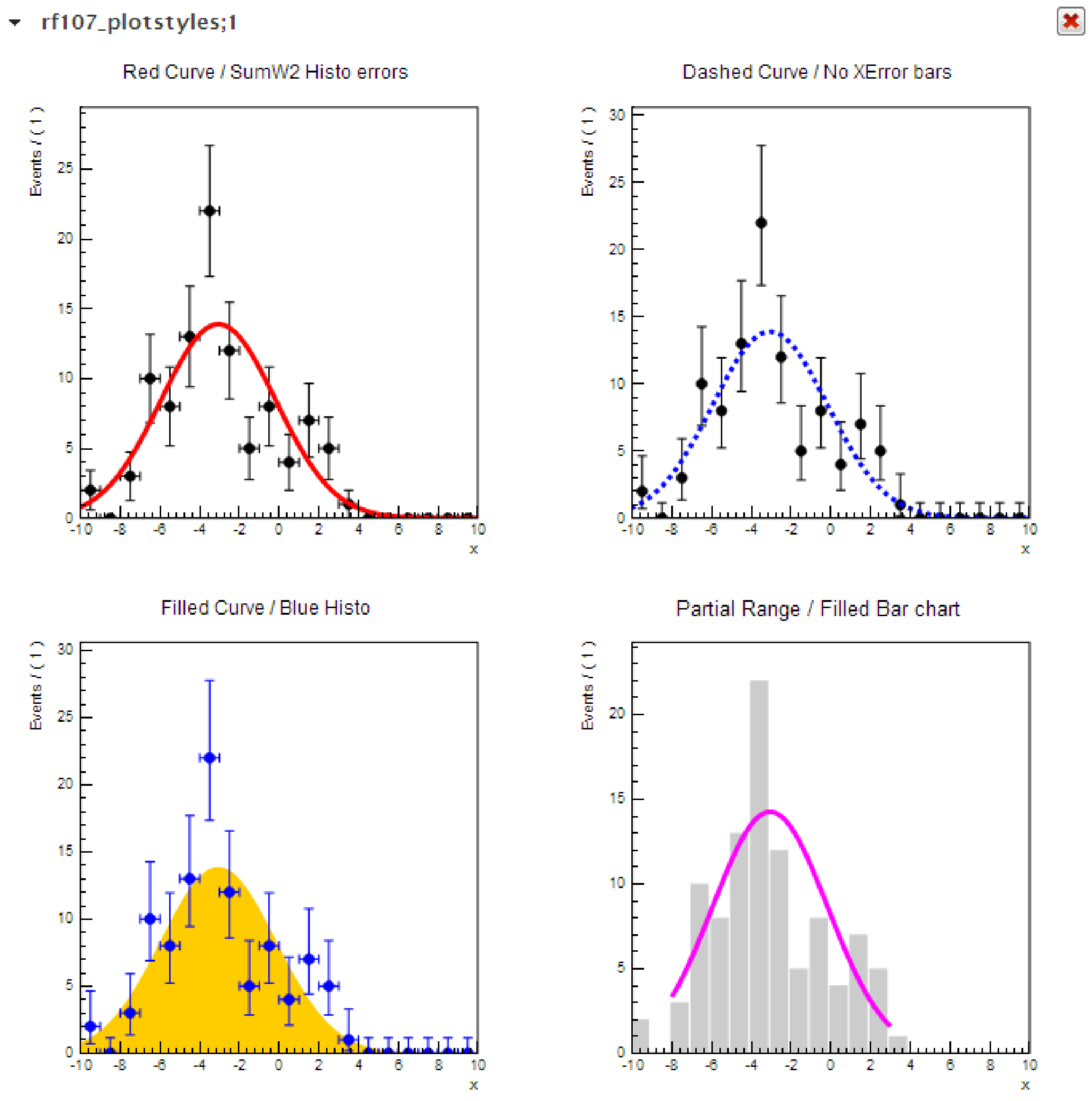

rf107_plotstyles;1
Red Curve / SumW2 Histo errors
Dashed Curve / No XError bars
Filled Curve / Blue Histo
Partial Range / Filled Bar chart
Events / (1)
x

Figure 1. Different RooFit plot styles.

Figures 2 and 3 show a canvas from the *zdemo.C* tutorial saved in a ROOT file. The first is drawn by ROOT and the second drawn by JSRootIO and *D3.js*. This demonstrates how both are very similar, with an identical layout, fonts, and colours, reaching one of the primary goals of the graphic subsystem. The only minor differences are special characters, such as Greek symbols, subscript and superscript characters. This issue should be addressed by adding full Latex support, using yet another JavaScript library, one candidate being MathJax [13].

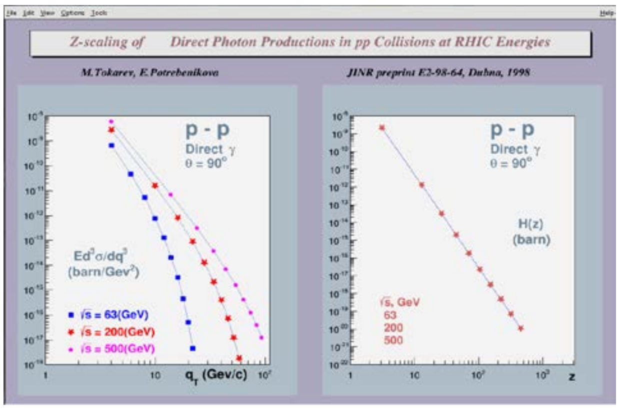

Z-scaling of Direct Photon Productions in pp Collisions at RHIC Energies
M.Tokarev, E.Potrebenikova
JINR preprint E2-98-64, Dubna, 1998
p - p
Direct γ
θ = 90°
Ed³σ/dq³ (barn/Gev²)
√s = 63(GeV)
√s = 200(GeV)
√s = 500(GeV)
q_T (Gev/c)
H(z) (barn)
√s, GeV
63
200
500
z

Figure 2. Original rendering, with ROOT.

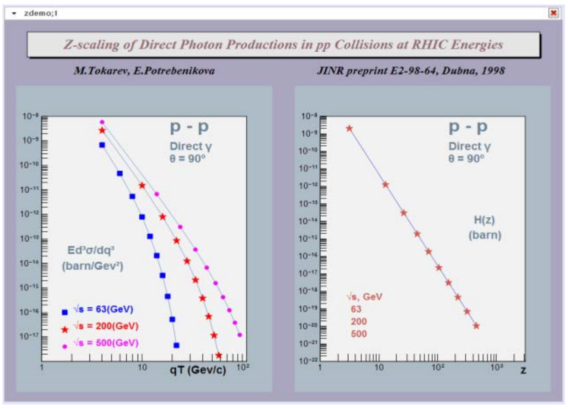
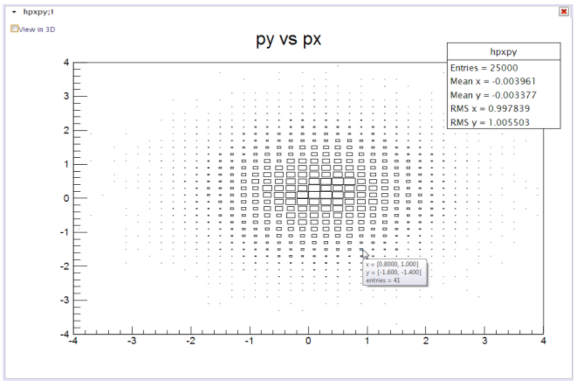

zdemo;1
Z-scaling of Direct Photon Productions in pp Collisions at RHIC Energies
M.Tokarev, E.Potrebenikova
JINR preprint E2-98-64, Dubna, 1998
p - p
Direct γ
θ = 90°
Ed³σ/dq³ (barn/Gev²)
√s = 63(GeV)
√s = 200(GeV)
√s = 500(GeV)
qT (Gev/c)
H(z) (barn)
√s, GeV
63
200
500
z

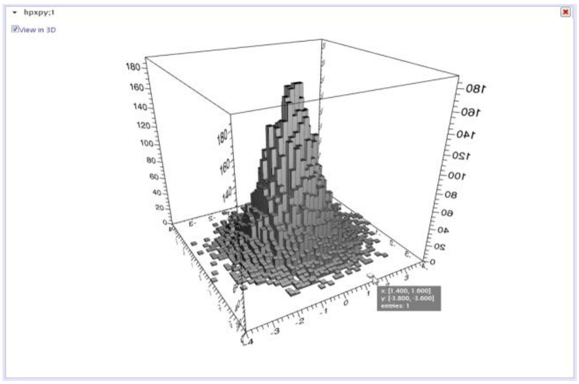
Figure 3. JavaScript rendering, in a browser.

Figures 4 and 5 show the same 2D histogram, once displayed in two dimensions with *D3.js*, and once displayed in three dimensions, with *Three.js*. The JavaScript code automatically detects if the web browser supports WebGL by querying the existence of a WebGL rendering context, and then will use it, or switch to an emulated mode, which is much slower, if there is no such support. When displayed in three dimensions, one can rotate the scene in any direction. These figures show also the "tooltips" indicating the bin position in its Cartesian coordinates, and the bin content (for example the number of entries). These tooltips automatically appear when the mouse cursor passes over a bin.



```
hpxpy
Entries = 25000
Mean x = -0.003961
Mean y = -0.003377
RMS x = 0.997839
RMS y = 1.005503
```

```
x = [0.800, 1.000]
y = [-1.600, -1.400]
entries = 41
```



```
x: [1.400, 1.600]
y: [-3.800, -3.600]
entries: 1
```

Figure 4. 2D histogram drawn with the BOX option (default).

Figure 5. Same histogram, displayed with the 3D option "LEGO".

Figure 6 shows a 3D histogram, which has once again been rendered with *Three.js*, using WebGL technology. In this example, the bin positions are described by three-dimensional Cartesian coordinates, and their size corresponds to their content. One can see one bin being highlighted with the mouse cursor, displaying a tooltip with this information. As for the 2D histogram drawn with the "LEGO" option, one can manipulate the 3D histogram, by rotating the whole scene in any direction.

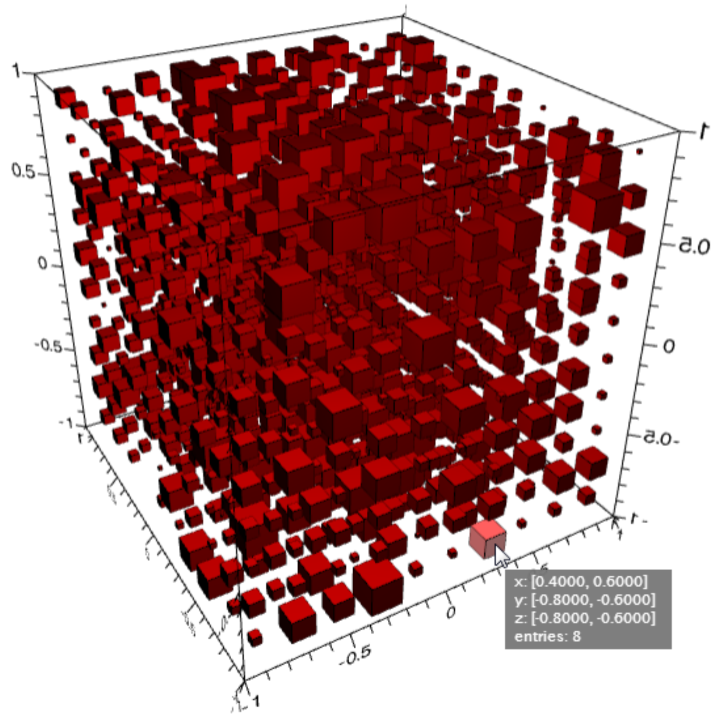


Figure 6. 3D Histogram (TH3) displayed with Three.js and WebGL.

3. Online monitoring with JSRootIO

The functionality of JSRootIO was extended to the online case, where object data are retrieved from running a ROOT-based application. This provides new possibilities for monitoring and control of live applications – a typical example could be online analysis. A special http server was introduced which, if requested, streams objects and delivers data to the browser. The main advantage of such an approach is that remote access to running applications from any computing device with normal web browser is gained.

Such an http server was implemented within the DABC [14] framework, using the Mongoose [15] web server (at the moment the server works only on Linux). The server can be attached to any ROOT-based code, such as a CINT script or a compiled executable. The server gets access to objects such as histograms, canvases, trees, folders or opened files through the *gROOT* global variable. Objects selected by a user's request are then streamed and delivered to the browser in a zipped format. The implementation is thread-safe, while the streaming of objects is performed in the main process, when no user code is executed.

Another new functionality is the possibility to enable live update of objects displayed in the browser. Several other new features were provided, such as context menu handling or statistics box update. Figure 7 shows the objects hierarchy, produced with the *hsimple.C* macro from ROOT tutorials. Several histograms are displayed and the context menu is activated for the 2D histogram. The user can activate monitoring (regular update) of displayed objects. With a minimal effort, the web page layout can be adjusted to the user's needs.

Using the powerful JSRootIO graphics, the DABC framework can provide a generic web-based user interface to the different DAQ, slow-control and analysis systems together.

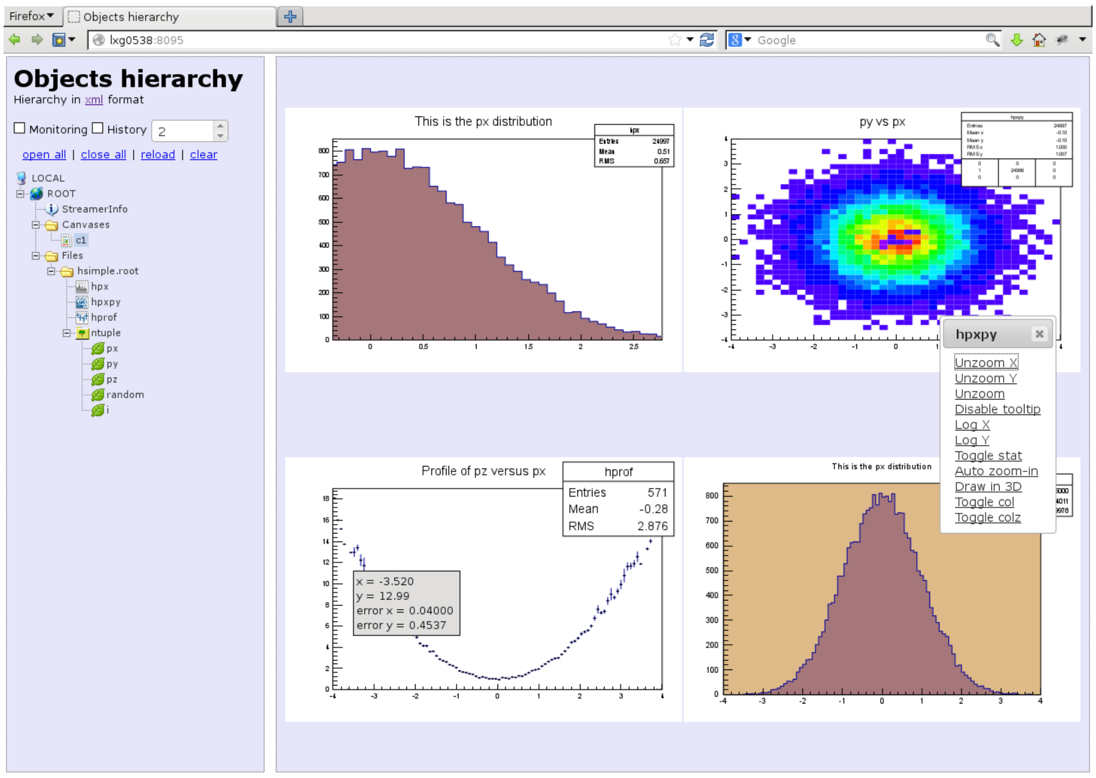


Figure 7. Objects display from running the *hsimple.C* macro.

4. Summary and Outlook

The JSRootIO library allows reading and displaying ROOT objects in an efficient way, and its rendering is now very close to the original one provided by ROOT. Using DABC-based web server, objects from any running ROOT application can be displayed remotely.

There are still some missing features, such as the Latex rendering using a JavaScript library, and these should be implemented in a near future. And since the JSRootIO library doesn't depend on any ROOT specific release, the development can proceed independently, and new features can be added in a transparent way for users.

References

[3] R. Brun and al., "ROOT — A C++ framework for petabyte data storage, statistical analysis and visualization", Computer Physics Communications; Anniversary Issue; Volume 180, Issue 12, December 2009, Pages 2499-2512
[4] B Bellenot 2012 J. Phys.: Conf. Ser. 396 052011
[5] D3.js, a JavaScript library for manipulating documents based on data, http://d3js.org/
[6] The BSD 3-Clause License, http://opensource.org/licenses/BSD-3-Clause
[7] Three.js, a JavaScript 3D library, http://threejs.org/
[8] The MIT License (MIT), http://opensource.org/licenses/MIT
[9] Scalable Vector Graphics (SVG), http://www.w3.org/TR/SVG
[10] HTML5's canvas, http://www.w3.org/wiki/HTML/Elements/canvas
[11] WebGL, OpenGL ES 2.0 for the Web, http://www.khronos.org/webgl
[12] RooFit, a toolkit for data modeling, http://root.cern.ch/drupal/content/roofit
[13] MathJax, a JavaScript display engine for mathematics, http://www.mathjax.org
[14] The DABC framework, http://dabc.gsi.de
[15] The Mongoose project, https://github.com/cesanta/mongoose