Exploring server/web-client event display for CMS

Alja Mrak Tadel[1,*], *Matevz* Tadel[1,**], *Avi* Yagil[1], *Dmytro* Kovalskyi[2], and *Sergey* Linev[3]

[1]UC San Diego, La Jolla, CA, USA 92093
[2]MIT, Cambridge, MA, USA 02139
[3]GSI, Darmstadt, Germany 64291

Abstract. The divergence of windowing systems among modern Linux distributions and OSX is making the current mode of event display operations difficult to maintain. In order to continue to support the CMS experiment event display, Fireworks, we need to explore other options beyond the current distribution model of centrally built tarballs.
C++-server web-client event display is a promising direction that can maintain the full functionality of Fireworks, including operation from the full experiment framework. In addition, it brings new features like multi-user debugging and the possibility to implement more elaborate visualization of non-event data through remote access to independent services.
We have been exploring mainly in the direction of Fireworks-based C++ server and thin web-client user interface as it allows for a large degree of reuse of existing algorithms as well as for full access to CMS data formats and accompanying functions that are crucial for the correct physics interpretation of event data. This paper presents the basic architecture of the system, discusses the communication protocol between server and client, and shows existing prototypes that demonstrate the feasibility of advanced event display features.

1 Introduction

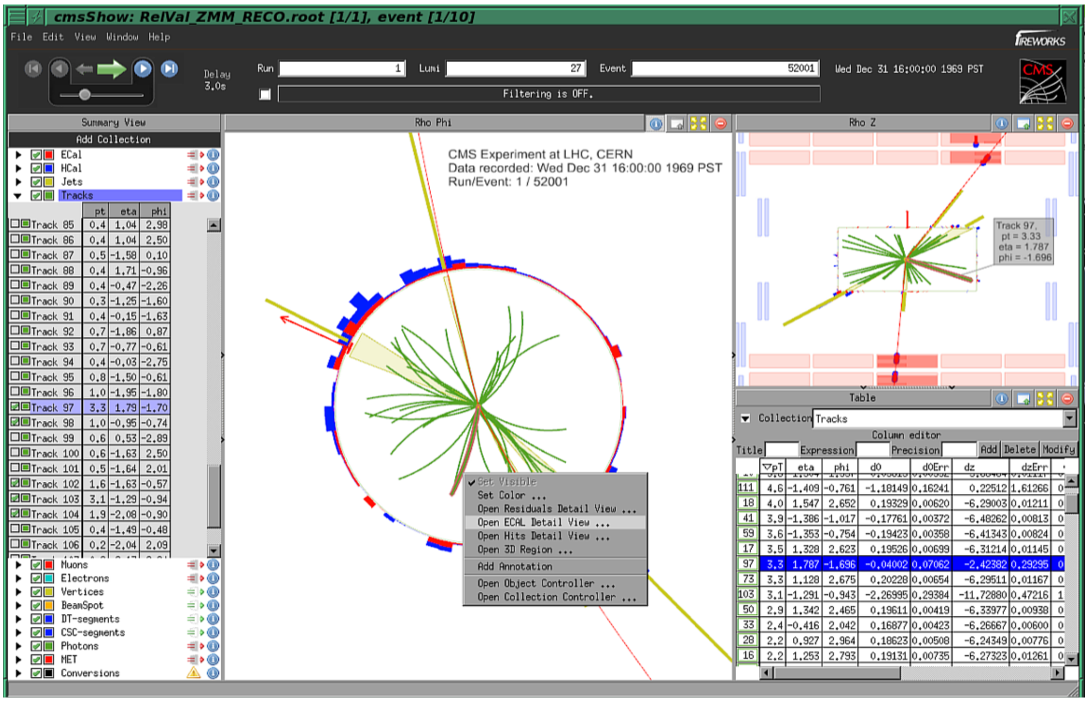
Fireworks is the official physics analyis oriented CMS event display since 2008. Its unique and powerful feature is its full interactivity with CMS Event Data Model (EDM)[1, 2]. From the very beginning, it has been intended to assist with physics analysis, trigger performance monitoring and evaluation, reconstruction algorithm development, and to facilitate development and debugging of detector geometry description. Every CMS physics collection can have a plugin that produces representation of physics objects in graphical views. The plugins are typically simple 100 line C++ sources that access EDM data and create new elements for a graphical scene. A screenshot of Fireworks application is shown in Figure 1.

1.1 Event Visualization Environment EVE as a base

After CMS reimplemented its software framework and data formats in 2007 a need arose also for a new event display application. At that time ROOT's Event Visualization Environment (EVE)[3] was chosen as the base. EVE was fully integrated with ROOT and it also provided most of the components required for any physics event display, enumerated below.

[*]e-mail: amraktadel@ucsd.edu
[**]e-mail: mtadel@ucsd.edu

1. Track propagation:
   - magnetic field representation;
   - track propagation through a set of points with given track parameters.
2. Projected views supporting nonlinear projections:
   - nonlinear scaling;
   - runtime distortion;
   - overlay guides like coordinate axes and energy scale legends.
3. A wide range of elements used to represent physics objects in 2D and 3D scenes: tracks, points, digit sets, calorimters, jets, etc.
4. Window management: it was possible to easily create OpenGL views, undock them, or rearrange them in complex layouts.



CMS Experiment at LHC, CERN
Data recorded: Wed Dec 31 16:00:00 1969 PST
Run/Event: 1 / 52001

	pt	eta	phi
Track 85	0.4	1.04	2.98
Track 86	0.4	1.04	2.50
Track 87	0.5	-1.58	0.10
Track 88	0.4	1.71	-0.96
Track 89	0.4	-0.47	-2.26
Track 90	0.3	-1.25	-1.60
Track 91	0.4	-0.15	-1.63
Track 92	0.7	-1.86	0.87
Track 93	0.7	-0.77	-0.61
Track 94	0.4	-0.03	-2.75
Track 95	0.8	-1.50	-0.61
Track 96	1.0	-1.95	-1.80
Track 97	3.3	1.79	-1.70
Track 98	1.0	-0.95	-0.74
Track 99	0.6	0.53	-2.89
Track 100	0.6	-1.63	2.50
Track 101	0.5	-1.64	2.01
Track 102	1.6	-1.63	-0.57
Track 103	3.1	-1.29	-0.94
Track 104	1.9	-2.08	-0.90
Track 105	0.4	-1.49	-0.48
Track 106	0.2	-2.04	2.09

	▽pT	eta	phi	d0	d0Err	dz	dzErr	
111	4.6	-1.409	-0.761	-1.18149	0.16241	0.22512	1.61266	0
18	4.0	1.547	2.652	0.19329	0.00620	-6.29003	0.01211	0
41	3.9	-1.386	-1.017	-0.17761	0.00372	-6.48262	0.00813	0
59	3.6	-1.353	-0.754	-0.19423	0.00358	-6.41343	0.00824	0
17	3.5	1.328	2.623	0.19526	0.00699	-6.31214	0.01145	0
97	3.3	1.787	-1.696	-0.04002	0.07062	-2.42382	0.29295	0
73	3.3	1.128	2.675	0.20228	0.00654	-6.29511	0.01167	0
103	3.1	-1.291	-0.943	-2.26995	0.29384	-11.72880	0.47216	1
50	2.9	1.342	2.465	0.19611	0.00419	-6.33977	0.00938	0
33	2.4	-0.416	2.042	0.16877	0.00423	-6.26667	0.00600	0
28	2.2	0.927	2.964	0.18623	0.00508	-6.24349	0.00776	0
16	2.2	1.253	2.793	0.19131	0.00735	-6.27323	0.01261	0

Figure 1. Screenshot of Fireworks event display with table views, event navigation controls, and popup menus.

2 Motivation for change

It has been more than 10 years since EVE framework has been developed. ROOT GUI and OpenGL components are even older and have by now really started to show their age. Furthermore, there is an increasing number of issues with the system level support for local and remote OpenGL. OpenGL API is being deprecated in favor of Vulkan (supported on

Android, Linux, and Windows). Apple has even announced OpenGL deprecation in OSX-10.4 in favor of proprietary graphics API Metal.

Another motivation to implement web-based event display was the fact that ROOT is moving to a web-based graphical interface. This is part of the ROOT project modernization branded as ROOT-7. While the full release is planned for the future, existing features are shipped with normal ROOT releases, enabling an easy start for web-based EVE module. An important component for this development is an embeddable http server that also supports usage of web sockets. The client side of web-based ROOT interface is named JSROOT[4, 5]. Its features include:

- mechanism to instantiate client windows and subwindows;
- integration of OpenUI5 GUI toolkit;
- simple structure to construct GL scene and navigate a camera;
- support drawing TGeo geometrical shapes.

3 Web-based Event Visualization Environment EVE-7

3.1 Modernization approach

Most of the functionality existing in EVE module can be simply reused on the server side. All elements representing physics objects are created on the server side. Magnetic field representation, track propagation system, and non-linear projections are reused without change. The life cycle management of EVE elements, views, and scenes stays basically the same as in the old implementation but requires some extensions to support server-client object synchronization.

The client side of EVE naturally extends JSROOT (a client side of ROOT-7) framework. In current prototype, we added a highlight / selection system across different views that also supports compound objects. At this stage, we have not yet implemented the window management functionality available in EVE. To test scene updates we have implemented graphical editors for EVE elements where each element type has a minimalistic definition of widget type and stubs required for sending method invocation requests to the server side.

3.2 Server-client communication

The server can send content to the client in text or binary format. Rendering information of scene elements is sent in binary format. This includes vertices, polygons indices, normals, and transformation matrices. All other data is sent in text format using `nlohmann::json` package[6].

3.3 Client connect

When a new client connects to a server all the scenes are streamed and sent to the client. This is a limitation of the prototype: when window management is implemented, the client will receive only the scenes from the viewers it subscribes to. For each scene a core text (typically this is nlohman::json object dumped into a flat string) information is sent first and then the binary.

3.4 Scene changes

In the current prototype element editors are auto generated based on their class type (example of REveJetCone editor is shown in figure 2). When a value inside a widget is changed by the user a *method invocation request* (MIR) is sent to the server. Below is a fragment of the javascript code:

```
sendMethodInvocationRequest : function(value, event)
{
  var mir = event.getSource().data("myData").srv + "(" + value + ")";
  var obj = { "mir"         : mir,
              "fElementId" : this.editorElement.fElementId,
              "class"       :  this.editorElement._typename  };
  this.mgr.handle.Send(JSON.stringify(obj));
}
```

The server invokes the method requested by the client and collects element changes into change information buffers held by each scene. Here is the list of currently supported element changes and the package content sent to the client in each case:

visibility: element ID, element and children visibility flags.

color: element ID, color and transparency attributes.

complete rebuild: element ID, core data and render data data.

Element additions and removals are also part of scene change updates. The new elements are streamed in the same way as when the client first connects to a server (core data as JSON, render data in binary format). When elements are removed only element indices are sent.

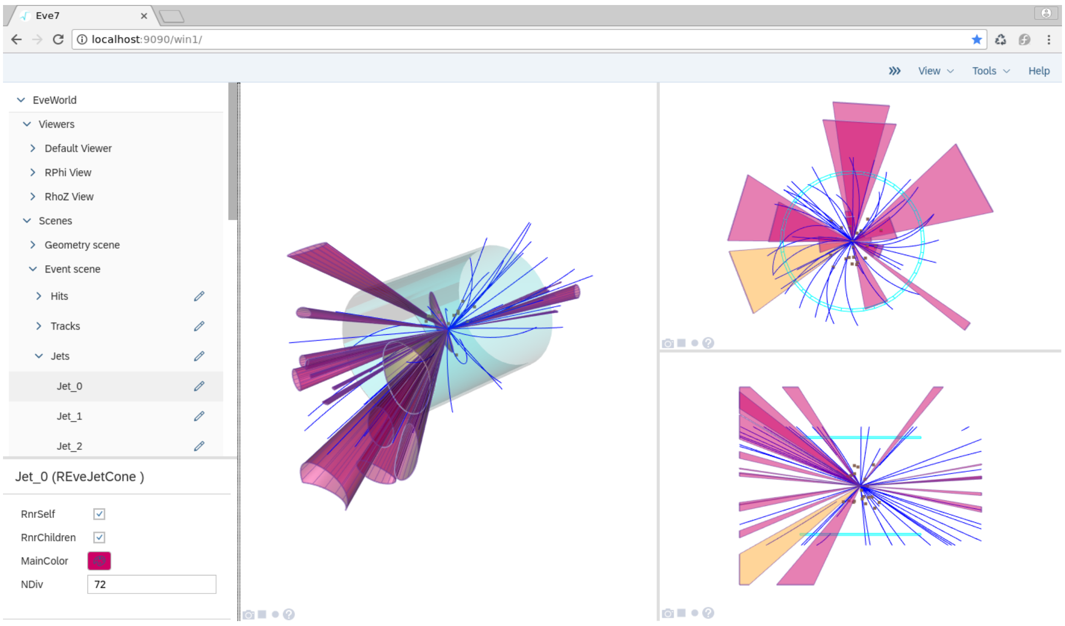


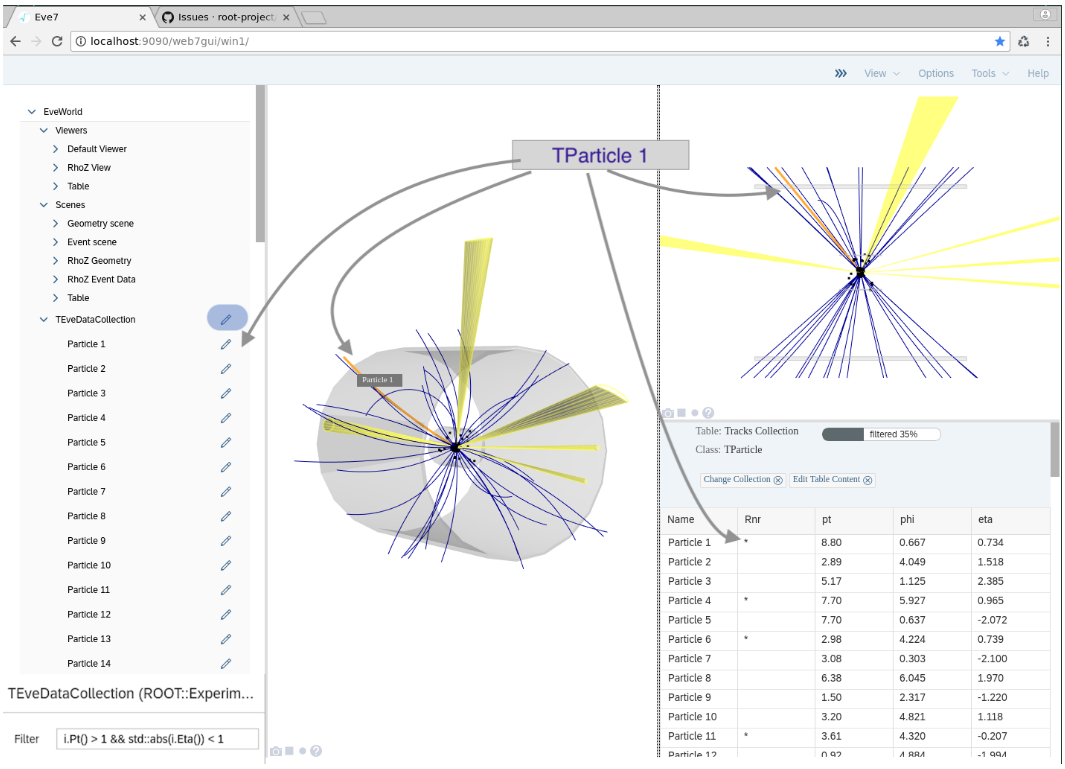
Figure 2. An example of three graphical views with an element editor in the bottom left corner.

3.5 Physics collections in EVE-7

Original EVE package has no support for management and display of experiment-specific physics collections. EVE objects were always just a visual representation of physics objects. During development of Fireworks we saw that access to *physics collections* and their contained *physics objects* are essential to the ability of the event display to offer the user full interactivity with the experiment data model. For this reason we have decided to implement this functionality in EVE-7 from the very beginning, relying on capabilitis of ROOT Cling and C++ lambda expressions. Some immediate benefits are:

- the ability to provide filtering of physics objects on the level of physics collections, and
- the support for table views with arbitrary, user specified expressions for each column.

A collection of TParticle objects shown in all supported views is presented in figure 3.



Filter: i.Pt() > 1 && std::abs(i.Eta()) < 1

Table: Tracks Collection
Class: TParticle
filtered 35%

Name	Rnr	pt	phi	eta
Particle 1	*	8.80	0.667	0.734
Particle 2		2.89	4.049	1.518
Particle 3		5.17	1.125	2.385
Particle 4	*	7.70	5.927	0.965
Particle 5		7.70	0.637	-2.072
Particle 6	*	2.98	4.224	0.739
Particle 7		3.08	0.303	-2.100
Particle 8		6.38	6.045	1.970
Particle 9		1.50	2.317	-1.220
Particle 10		3.20	4.821	1.118
Particle 11	*	3.61	4.320	-0.207
Particle 12		0.92	4.884	-1.994

Figure 3. A screenshot of a TParticle collection. One can see representations of the collection in summary view, 3D view, $\rho - z$ projection, and table view.

3.6 Dynamic tables

Example of an EVE-7 table is shown in figure 4. It is possible to edit the table properties at runtime, including changing of the assigned physics collection. The column contents can be modified by editing the C++ expression applied to every object in the collection. New columns can also be created from the GUI.

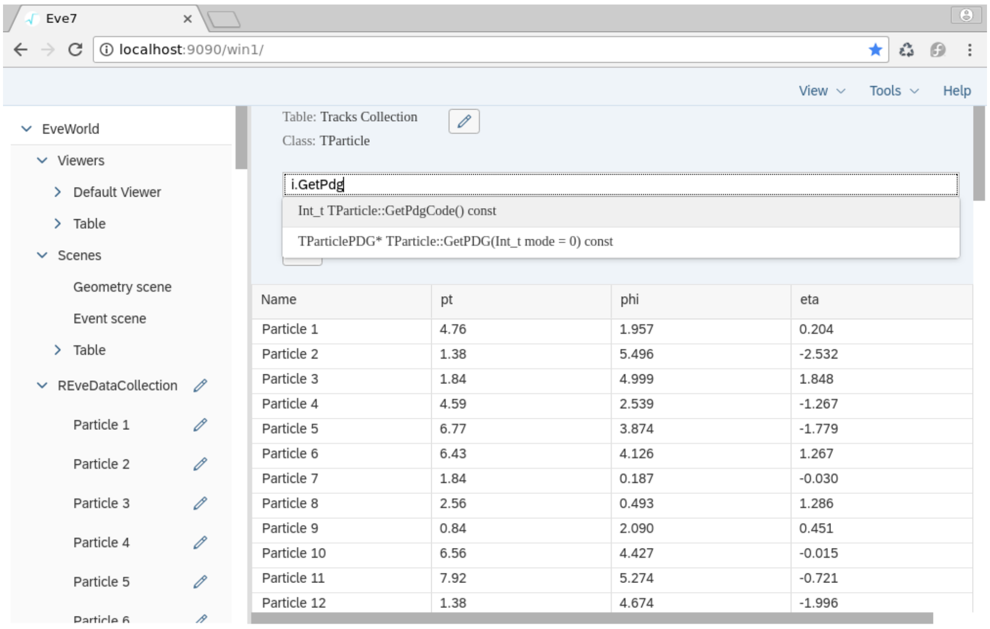


Table: Tracks Collection
Class: TParticle

i.GetPdg

Int_t TParticle::GetPdgCode() const

TParticlePDG* TParticle::GetPDG(Int_t mode = 0) const

Name	pt	phi	eta
Particle 1	4.76	1.957	0.204
Particle 2	1.38	5.496	-2.532
Particle 3	1.84	4.999	1.848
Particle 4	4.59	2.539	-1.267
Particle 5	6.77	3.874	-1.779
Particle 6	6.43	4.126	1.267
Particle 7	1.84	0.187	-0.030
Particle 8	2.56	0.493	1.286
Particle 9	0.84	2.090	0.451
Particle 10	6.56	4.427	-0.015
Particle 11	7.92	5.274	-0.721
Particle 12	1.38	4.674	-1.996

Figure 4. A screenshot of a TParticle collection table. One can add new content using suggestions supporting tab and drop-down menu completion.

4 Migrating Fireworks from EVE to Web EVE

Fireworks will continue to keep its primary role as a physics analysis oriented event display. From EDM event data the same minimally changed plugins will construct EVE elements and register them to EVE scenes for display and further processing, e.g., automatic transformation for projected views, or propagation of tracks through a set of given markers. Removing the TVirtualX dependency will be the first effort in the migration. The OpenUI5 [7] controls will be used instead of the ROOT GUI toolkit to access and control event navigation, filtering and physics collection content.

Some parts of Fireworks will be replaced with functionalities now available in EVE-7 and OpenUI5. For example, physics collection implementation is already existing in EVE-7 and custom TVirtualX implementation of tables will be replaced by OpenUI5 tables.

5 Conclusion

The exploratory work presented in this paper has reassured us that web-based event display is an attainable solution for the next decade. CMS is committed to supporting development of EVE-7 and modernization of Fireworks as it remains the official physics analysis oriented event display application for Run-3 of the LHC and for the High Luminosity LHC.

The proposed work plan is to evolve EVE-7 to a level where the majority of basic features are supported and then port a subset of Fireworks to EVE-7. Afterwards, as more advanced features are being implemented, development of EVE-7 and Firworks-Web can proceed in lock-step, especially for visualization of digits, calorimeters, and particle flow objects. In parallel, required optimizations will be implemented on server and client side as as well as in the server-client communication protocol.

References

[1] L. A. T. Bauerdick *et al.*, *"Event display for the visualization of CMS events,"* J. Phys. Conf. Ser. **331**, 072039 (2011).

[2] D. Kovalskyi *et al.*, *"Fireworks: A physics event display for CMS,"* J. Phys. Conf. Ser. **219**, 032014 (2010).

[3] M. Tadel, *"EVE: Event visualization environment of the ROOT framework,"* PoS ACAT **08**, 103 (2008).

[4] Sergey Linev, JSROOT[software], Development release 2012. http://root.cern.ch/js

[5] B. Bellenot and S. Linev, *"JavaScript ROOT,"* J. Phys. Conf. Ser. **664**, no. 6, 062033 (2015).

[6] Niels Lohman, json[software], Master release 2018. https://github.com/nlohmann/json

[7] Peter Muessing, OpenUI5 [software], Deveopment release 2018. https://openui5.org