A browser-based event display for the CMS Experiment at the LHC using WebGL

To cite this article: T McCauley 2017 J. Phys.: Conf. Ser. 898 072030

View the article online for updates and enhancements.
A browser-based event display for the CMS Experiment at the LHC using WebGL

T McCauley

1 University of Notre Dame, Notre Dame, IN, USA
E-mail: thomas.mccauley@cern.ch

Abstract. Modern web browsers are powerful and sophisticated applications that support an ever-wider range of uses. One such use is rendering high-quality, GPU-accelerated, interactive 2D and 3D graphics in an HTML canvas. This can be done via WebGL, a JavaScript API based on OpenGL ES. Applications delivered via the browser have several distinct benefits for the developer and user. For example, they can be implemented using well-known and well-developed technologies, while distribution and use via a browser allows for rapid prototyping and deployment and ease of installation. In addition, delivery of applications via the browser allows for easy use on mobile, touch-enabled devices such as phones and tablets.

iSpy WebGL is an application for visualization of events detected and reconstructed by the CMS Experiment at the Large Hadron Collider at CERN. The first event display developed for an LHC experiment to use WebGL, iSpy WebGL is a client-side application written in JavaScript, HTML, and CSS and uses the WebGL API three.js. iSpy WebGL is used for monitoring of CMS detector performance, for production of images and animations of CMS collisions events for the public, as a virtual reality application using Google Cardboard, and as a tool available for public education and outreach such as in the CERN Open Data Portal and the CMS masterclasses. We describe here its design, development, and usage as well as future plans.

1. Introduction

CMS (Compact Muon Solenoid) [1] is a general-purpose particle physics experiment at the Large Hadron Collider (LHC) at CERN. It has a broad physics program including studies of QCD, electroweak, top, B, forward, and heavy-ion physics as well as searches for supersymmetry and physics beyond the Standard Model. During LHC Run 1, from 2010 to early 2013, CMS collected proton-proton collision data at center-of-mass energies of up to 8 TeV as well as data from proton-lead and lead-lead collisions. The culmination of Run 1 was the discovery of the Higgs boson by CMS [2] and the ATLAS experiment [3].

At the beginning of 2015 Run 2 began and the LHC resumed operation after a two-year shutdown for improvements and repairs to the accelerator and to the experiments. 2016 saw the LHC deliver 41 fb$^{-1}$ of proton-proton collision data at $\sqrt{s} = 13$ TeV as well as proton-lead collisions to CMS. As of this writing at the beginning of 2017, Run 2 has paused for an extended technical stop. The CMS collaboration continues study of the Standard Model of particle physics and what may lie beyond at this high-energy frontier.

In particle physics experiments such as CMS, applications known as event displays visualize collision events in 2D and 3D and are valuable tools that find many uses. These include validation...
of detector geometries, development of event reconstruction algorithms, visual inspection of reconstructed events, and production of images for communication of physics results to the public.

We describe in this paper how iSpy WebGL fulfills some of these use-cases, beginning with the requirements, proceeding with a description of its features, and ending with future plans for development.

2. Requirements and Development

One approach (probably the most common) for development of event displays is to build the application “from scratch” using a 3D graphics library such as OpenGL [4] and a graphical-user-interface (GUI) toolkit. The original iSpy application [5] was just such an application, written in C++ which used OpenGL (via the Coin3D API [6]) and Qt4 [7]. This application was distributed for use on the desktop as a fully-bound executable, binding up multiple dependencies for different versions of several operating systems. iSpy was originally used for monitoring detector status and producing high-quality public images of collisions.

A browser-based application [8] with a look-and-feel similar to the original iSpy was developed for use in educational programs such as the QuarkNet [9] CMS masterclasses [10] where students conduct simplified analyses of CMS data using visualizations of events. In this use-case, where older computers or operating systems may be used, restrictive computing environments may exist, or connection to the Internet could be poor or non-existent, a browser-based application is ideal.

Both applications were used concurrently to satisfy two use-cases: the desktop application for production of high-quality event images for publicizing CMS results and the browser-based application for use by the public, particularly students. Given the time and effort required to develop and maintain two separate applications, the decision was made to combine the functionality of both applications into one. This is made possible by the development of WebGL, a “cross-platform, royalty-free web standard for a low-level 3D graphics API based on OpenGL ES 2.0, exposed through the HTML5 Canvas element as Document Object Model interfaces” [11]. WebGL allows for GPU-accelerated graphics in the browser. In iSpy WebGL an open-source JavaScript API to WebGL called three.js [12] is used.

iSpy WebGL therefore combines the high-quality graphics of the desktop application with release, distribution, and use via the browser, supporting both use-cases. Additional benefits include a smaller code base, fewer dependencies, and rapid prototyping.

However, ease of distribution does not come entirely from using the browser. This is made more possible by the choice of input format. The format used is the ig format, which is essentially a zip file containing one or more json (JavaScript Object Notation) files which themselves contain the necessary information for rendering of the event and of the detector elements. The information is extracted from the CMS event format with CMSSW [13, 14, 15] C++ code [16] and converted to the ig format. The dependence on CMSSW is minimized and the format is human-readable, portable, and easily parsed by multiple programming languages. The simple structure of an ig file and some of the json contents can be seen in Figure 1.

3. Features

iSpy WebGL is written in JavaScript, HTML, and CSS and is a purely client-side application for use in a web browser. The latest stable version is distributed at http://cern.ch/ispy-webgl. It was first released in December 2014 and the fourth and latest release was April 2016. It is open-source and code management, releases, and issue tracking are handled via GitHub [17]. By default the CMS geometry available is the so-called reconstruction geometry, which includes the “sensitive” parts of the CMS detector e.g. silicon strips in the tracker, crystals in the
Figure 1. Above: an example of the simple structure of an unzipped ig file. Below: an example of the json contents in an event file.

Figure 2. View of iSpy WebGL with an event loaded. Along with the event the reconstruction geometry of the electromagnetic calorimeter barrel can be seen along with an imported beam pipe geometry. The table view is below the 3D view containing event and geometry rendering, the tree view is to the left of the 3D view, and the toolbar is along the top.

electromagnetic calorimeter, layers in the hadronic calorimeters, and chambers in the muon system.

The application interface is partitioned into four main sections: (1) the 3D view where the WebGL graphics are exposed via a HTML5 canvas, (2) the table view, (3) the tree view, and (4) the toolbar. A screenshot of the application with an event loaded can be seen in Figure 2. iSpy WebGL uses the bootstrap.js front-end framework [18] which allows for responsive layout: the application scales automatically to fit the device viewport from a phone up to a large desktop monitor or touch screen. Touch events are enabled for manipulation of the 3D view on touch-
enabled devices such as phones, tablets, and touch screens. The standard touch events are supported: one-finger move for rotation, two-finger pinch and expand for zoom in and out, and three-finger move for panning.

The 3D view by default uses the three.js WebGLRenderer when WebGL is available. If either WebGL is not on the user’s machine or it is not supported by the user’s browser, then a slower canvas-based rendering is used. In the display one is also able to switch to a SVG (scaleable vector graphic) renderer in order to view and export vector graphics.

The tree view displays the main object collections available in the detector and event in a collapsable tree and visibility is controlled via a toggle. The contents of a particular object collection are displayed in a table view which is sortable by column. Selection of a particular object in the table view is correlated to its corresponding graphical representation in the 3D view: a particular object selected in the table will change color in the 3D view. Likewise, using ray tracing one can pick an object in the 3D view and its properties will be highlighted in the table view. Clicking on the object in the 3D view will display its information in a popup dialog.

Most controls in the toolbar correspond to standard event display tools: iterate forwards and backwards through events, return to a home (i.e. start) view, zoom in/out, view along the XYZ axes, and switch between orthographic and perspective views. Through the “Open File” dialogue one can load several default files available online. Using the HTML5 File API [19] one can also load files from the user’s local machine. One can also export the 3D view to a raster format such as png via the “Print Image to File” button. Configurable settings via the toolbar include switching back and forth from the default black background to white and setting of the maximum frame rate.

Several more detailed features are described in the following subsections.

3.1. Import/export

The simulation geometry of CMS includes all detector elements as well as support structure. This detailed geometry can be read into SketchUp [20] [21] and exported to various 3D file formats. Import and export of multiple formats are supported by three.js and iSpy WebGL for now just supports import and export of obj format. The exported geometries from SketchUp can therefore be loaded into iSpy WebGL and an example image of a CMS event along with a section of the detailed geometry can be seen in Figure 3.
3.2. Animation
The “Start/Stop Animation” button begins a default animation sequence where two shapes representing the colliding bunches arrive at the middle of the empty detector from opposite ends of the beam axis, event objects appear, and the camera zooms to the center of the detector and then rotates around the inside. Currently there is no capture of the animation to a video format via the browser (at least not yet available in production) but rather through screen capture tools such as QuickTime. A typical animation of a collision can be found on YouTube [22].

3.3. Stereo view
A Google Cardboard viewer [23] is an inexpensive virtual reality headset. The simplest version is made literally from cardboard. A suitable smartphone is inserted into the viewer and a stereoscopic view is achieved with the two lenses in the viewer combined with appropriately-rendered content in the phone, either a native or browser-based application. An example of a viewer with an inserted phone can be seen in Figure 4.

With iSpy WebGL running on a suitable smartphone and a Google Cardboard viewer one can view a collision event in stereo. With an event loaded in the application, one presses the “Stereo View” button, inserts the phone into the viewer, and views. A screenshot of the stereo view is shown in Figure 5. The view changes with the device orientation and the rendered scene slowly pans towards the viewer.

The main features of the stereo view were developed during the Mozilla Science Lab Global Sprint over 4-5 June 2015 [24]. Over these two days many open-source open-science projects were organized by the Mozilla Science Lab [25] and carried out at various locations, often with virtual participation. One project hosted at CERN was development of the stereo view for iSpy WebGL.

4. Usage
iSpy WebGL has been used for production of public images and animations of collision events in Run 2 of the LHC. These media publicized CMS physics results and milestones such as the beginning of proton-proton collisions at $\sqrt{s} = 13$ TeV. Example images on the CERN Document Server may be found at [26].

Figure 4. A Google Cardboard viewer with a phone inserted. Image credit: [23].
In the QuarkNet CMS masterclasses iSpy WebGL is used in the simplified analyses conducted by the students. As part of the International Masterclasses organized by the International Particle Physics Outreach Group [27] held 11 February to 23 March 2016 approximately three thousand students from over forty countries used the event display [28]. Further masterclasses in 2016 where held in other countries such as Rwanda and Ethiopia [29].

In 2014 as part of its open-access policy CMS began to release a large fraction of its reconstructed collision data into the public domain. Two large releases have occurred. The first, in November 2014 was half of the 2010 proton-proton collision data at $\sqrt{s} = 7$ TeV, equivalent to an integrated luminosity of tens of pb$^{-1}$. The second, in April 2016, was half of 2011 proton-proton collision data at $\sqrt{s} = 7$ TeV, equivalent to around 2.5 fb$^{-1}$. Each dataset is divided into several smaller datasets corresponding to specific trigger paths.

These data were made available via the CERN Open Data Portal (CODP) [30] based on the digital library software Invenio [31]. The CODP contains several online applications. One application is a version of iSpy WebGL with which a sample of events from each smaller, selected dataset is available for visualization at [32]. The latest release of the CODP was 18 April 2016. Since the release the peak number of visits to the event display from distinct IP addresses in a day was around forty-six thousand, settling down to around one hundred per day by Jun 2016. From April to June 2016 there were around sixty-six thousand visits to the event display in total, which accounted for about thirty percent of the total number of visits to the CODP [33]. In the CODP there are several records of event display files in ig format. A version of iSpy WebGL works as an Invenio viewer plugin for the ig format [34], an example of which can be seen in Figure 6.

The cathode strip chamber (CSC) muon detector group of CMS uses the display for detailed technical examination of individual collision events to better understand event kinematics and detector behavior, and to explore muon track segment reconstruction algorithms.

Finally, for the general public there has been production of high-resolution animations for public exhibits and use. Two examples include for the “Uncertainty” exhibition at the Alyce de Roulet Williamson Gallery, Pasadena, California [36] and a 3D film for visitor viewing at CMS Point 5 in Cessy, France.
5. Conclusions and future plans

Since its first release in 2014 iSpy WebGL has been used by CMS to produce public images and animations, for detector monitoring, and for public uses such as masterclasses and in the CERN Open Data Portal. The combination of high-quality graphics enabled by WebGL in a browser-based application has enabled the application to be used in many environments by many different types of users from experts to the general public.

Future plans include an improved interface for style configuration, addition of dedicated RΦ and RZ views, improved configuration and export of animations, and additional support for import and export of 3D formats. Usual maintenance includes keeping up with evolving versions of CMSSW and CMS geometry as well as versions of three.js. Contributions have been made to three.js over 2016, specifically improvements to the obj format exporter. An importer for the Geometry Description Markup Language (GDML) [37] is under development and when completed will hopefully be included in future releases of three.js.

The new and evolving WebVR API [38] can expand the possibilities for virtual reality in the browser. This also will be worth investigating.

Acknowledgments

Thanks to fellow developers of the original iSpy application and of the ig file format G Alverson, G Eulisse, S Muzzafar, I Osbourne, L Taylor, and L Tuura. Thanks to M Hategan and P Nguyen for development of the original browser-based application and to previous iSpy WebGL
contributors L Barnard, C Logren, and M Saunby. Thanks to Invenio team for assistance in developing for the CERN Open Data Portal. Support from the CMS Collaboration and QuarkNet is gratefully acknowledged.

References
[12] https://threejs.org
[19] https://www.w3.org/TR/FileAPI
[22] https://www.youtube.com/watch?v=htwyxc4kgAs
[23] https://vr.google.com/cardboard
[25] https://science.mozilla.org
[26] https://cds.cern.ch/record/2151078
[27] http://physicsmasterclasses.org
[28] K Cecire and U Bilow, private communication
[33] T Simko, Invenio team, private communication
[34] https://github.com/inveniosoftware/invenio-previewer-isy
[38] https://w3c.github.io/webvr