iSpy WebGL: a browser-based event display for CMS using WebGL

Tom McCauley
University of Notre Dame, USA
thomas.mccauley@cern.ch
@tpmccauley
http://cern.ch/ispy-webgl

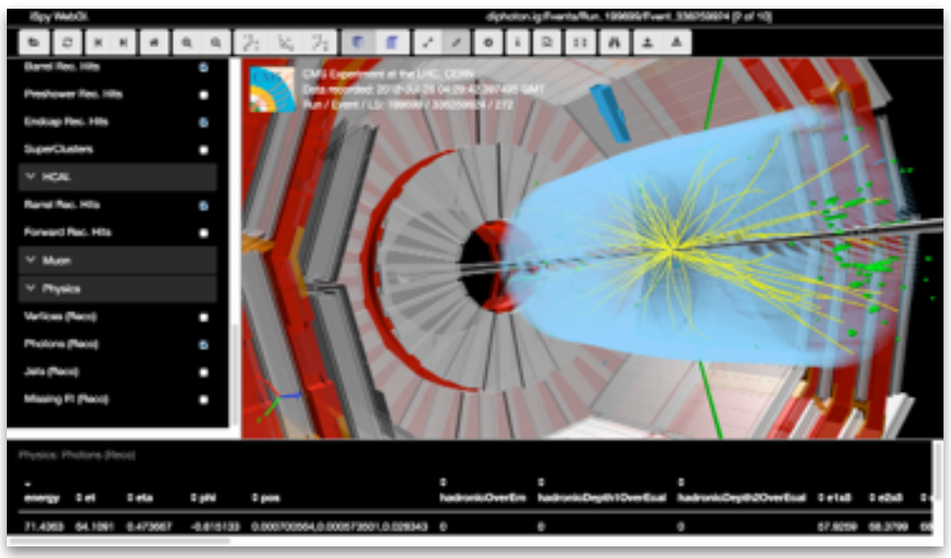
23-26 Jan 2017

Use-cases/requirements: an easy-to-use and easy-to-distribute event display that can produce high-quality event images and animations (mostly public-facing) and be used by the public (*e.g.* students)
Solution (partly): create an application for use in the browser using JavaScript, HTML, CSS, and WebGL

"WebGL is 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": https://www.khronos.org/webgl/

iSpy WebGL
http://cern.ch/ispy-webgl
https://github.com/cms-outreach/ispy-webgl

- 1st release: Dec 2014
- purely client-side
- uses three.js JavaScript WebGL library/API
- input format: JSON converted/derived from CMS ROOT format



The good/bad (of WebGL and using the browser):

- WebGL graphics ✔
- Accessibility: application is distributed, used, and updated via a browser ✔
- Fast development and release cycle ✔
- Possibility to run offline is not precluded ✔
- Support for mobile devices and touch screens comes "for free" using the browser ✔
- three.js: large and active open-source developer community ✔
- Very large and complicated geometries (and events?) can be burdensome to run in the browser (but browsers and support are always getting better) ✘
- Sometimes: JavaScript, ugh ✘
- Requirement that is difficult to satisfy (certainly for a client-side application): direct access to experiment's data model (but this is not necessarily a requirement for all use-cases; iSpy WebGL doesn't require it and uses an intermediate data format) ✘

Desiderata and the future (for WebGL applications):

- At least 3 LHC experiments have WebGL applications using three.js
- Perhaps an experiment-agnostic HEP library/API (a hep.js?) with three.js at its core?
- For example, common things that would be nice to have (and not have to reinvent for every application): support for 3D view, projected 2D views, lego view, table view, physics objects (tracks, jets, calohits), configuration of views, ...,WebVR?