New web based event data and geometry visualization for LHCb

A Pappas[1], B Couturier[2], S Ponce[3]

[1] National and Kapodistrian University of Athens, Department of Informatics and Telecommunications, Panepistimioupolis, Ilisia Athens, 157 84, Greece
[2] CERN, Esplanade des Particules 1, P.O. Box, 1211 Geneva 23 Switzerland

E-mail: andreas.pappas@cern.ch, ben.couturier@cern.ch, sebastien.ponce@cern.ch

Abstract. The LHCb detector is undergoing a comprehensive upgrade for data taking in the LHC's Run 3, which is scheduled to begin in 2022. The new Run 3 detector has a different, upgraded geometry and uses new tools for its description, namely DD4hep [2] and ROOT [12]. Besides, the visualization technologies have evolved quite a lot since Run 1, with the introduction of ubiquitous web based solutions or Augmented Reality (AR) for example. The LHCb collaboration has thus started the development of a new visualization solution, based on the Phoenix [6] framework, developed jointly by several experiments in the context of the HEP Software Foundation (HSF) [10]. We present here the architecture and implementation of this new solution, as well as the different contributions made to the Phoenix ecosystem. In particular we discuss a generic tool for exporting ROOT geometries to the visualization application, which can be used to display in a browser either the whole detector or subparts of it. Extensions to the Phoenix visualization primitives regarding calorimeters and performance improvements are also presented.

1. Introduction

This paper details the work behind the new solution for visualizing (LHCb) geometry and event data. It builds on latest client side 3D technologies such as three.js [11] and WebGL [13], as well as Augmented Reality (AR) to provide a smoother and exciting user experience for Physics visualizations. It also demonstrates the integration of the HSF [10] Phoenix [6] framework alongside an experiment software stack and the standard High Energy Physics frameworks like ROOT [12]. The tools developed have been kept generic and are thus reusable by other experiments even outside High Energy Physics (HEP).

2. Phoenix visualization framework and LHCb contributions

In 2017 the HSF visualisation white paper [3] identified the desirability of having a common event format, and a common tool to visualise event data and geometry. Up until now, event displays tended to be experiment specific and had to be installed alongside their dependencies when someone wanted to use them. Installations triggered the usual versioning and upgrade burdens, especially when some components were dependent on experiment reconstruction frameworks.

As browsers became more popular in recent years and their technology evolved drastically, 3D visualization in a browser became possible, thus making event display development simpler.

Visiting a URL is all you have to do to use the event display and no software installation is required at all. This is what the Phoenix event display framework offers.

The Phoenix framework is a TypeScript-based (browser) event display visualization framework, which uses the popular three.js [11] library for 3D and is supported by the HEP Software Foundation (HSF). It focuses on being experiment agnostic by design, with common tools (such as custom menus, controls, propagators) and the possibility to add experiment specific extensions.

2.1. LHCb web event display

As most experiments, the LHCb collaboration has an experiment specific event display, called Panoramix [7] which is still in use for run 1 and 2 data. Maintenance has become difficult as it uses old technologies. Thus the LHCb collaboration decided to develop a new event display and to base it on Phoenix.

Phoenix being experiment agnostic, LHCb could take benefit of its features and concentrate on upgrading the geometry for run 3. VErtex LOcator (VELO) hits in particular were easy to visualize using the visualization primitives available in Phoenix (`Figure 1`) on top of the new geometry. In other cases the set of visualization primitives had to be extended, in particular for calorimeter hits (`Figure 2`). These extensions have been kept generic to be reused by other experiments.

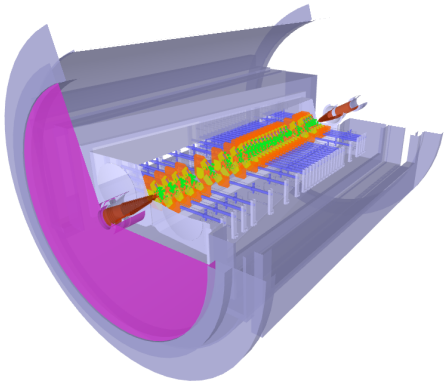

Figure 1. VErtex LOcator (VELO) hits and geometry Phoenix visualization

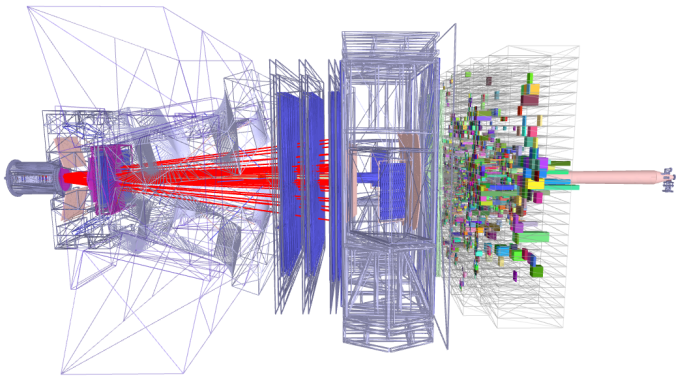

Figure 2. LHCb run3 geometry and event data Phoenix visualization

2.2. Extension of the visualization primitives inside Phoenix

In this subsection, we are going to detail the visualization extensions in Phoenix.

LHCb had to extend the set of visualization primitives of Phoenix in order to display its planar calorimeter hits. Thanks to the Phoenix documentation, this has been easy to do and could be kept generic for future reuse. This feature was immediately reused when the original authors of Phoenix themselves provided feedback, that they used the extension and they drew a planar type of visualization for the ATLAS experiment.

In the following snippet we show the core part of the code that was written to produce the visualization of the calo cells which do not point to the origin, i.e. planar calo cells (`Figure 2`).

The general principle is that the data to visualize is available in JSON format (as a dictionary of key value pairs), and a method needs to be provided with interface `any -> Object3D`, which receives the data, handles them and returns to the Phoenix loader the final 3D format to be visualized.

In the case of LHCb, the idea of the planar calo cell is to visualize the energy deposits as boxes perpendicular to the plane of the calorimeter, where the length of the box will be proportional to the amount of energy. The produced result can be seen in (`Figure 5`). Specifically the method contains the following steps: (`Figure 3`)

- It retrieves the data (`lines 2:5`).
- It creates the box representing the deposit (`lines 7:11`).
- It places the box in the back of the calorimeter and returns the final mesh (`lines 13:24`).

The code looks like this:

```
1  public static getPlanarCaloCell(caloCells: any): Object3D {
2    let position = caloCells.pos;
3    const length = caloCells.energy * 0.22;
4    const size = caloCells.cellSize;
5    const plane = caloCells.plane;
6
7    const geometry = new BoxBufferGeometry(size, size, length);
8    const outerBox = new Object3D();
9    const material = new MeshPhongMaterial({color: caloCells.color ?? ↩
         EVENT_DATA_TYPE_COLORS.PlanarCaloCells});
10   const box = new Mesh(geometry, material);
11   outerBox.add(box);
12
13   const boxPosition = new Vector3( ...position.slice(0,2),
14                                    ((plane[3]) + (length/2)) );
15   box.position.copy(boxPosition);
16   let qrot = new Quaternion();
17   qrot.setFromUnitVectors( new Vector3(0, 0, 1),
18                            new Vector3( ...plane.slice(0,3) ) );
19   outerBox.quaternion.copy(qrot);
20   outerBox.userData = Object.assign({}, caloCells);
21   outerBox.name = 'PlanarCaloCell';
22
23   return outerBox;
24 }
```

Figure 3. Creation of the planar calo cells code.

The new visualization primitive requires a few more lines of code (in the Phoenix loader) to be activated in the Phoenix framework. The steps of the extra code are: (`Figure 4`)

- It checks if planar calo cells are in input and it firstly adds an energy cut to them (`lines 50:51`).
- It adds a slider to control the scale of the energy deposit (`lines 52:60`).
- It adds the cuts, the slider and the planar calo cells that were created in the previous code to the Phoenix loader (`lines 61:64`).

The final code looks like this:

```
50 if (eventData.PlanarCaloCells) {
51     const cuts = [ new Cut('energy', 0, 10000) ];
52     const addPlanarCaloCellsOptions = (
53         typeFolder: GUI,
54         typeFolderPM: PhoenixMenuNode
55     ) => {
```

```
56        const scalePlanarCaloCells = (value: number) => {
57          this.graphicsLibrary
58          .getSceneManager()
59          .scaleChildObjects('PlanarCaloCells', value / 100, 'z');
60      };
61      this.addCollection( cuts,
62          addPlanarCaloCellsOptions,
63          PhoenixObjects.getPlanarCaloCell );
64  }
```

Figure 4. Addition of the planar calo cells code to the Phoenix Loader

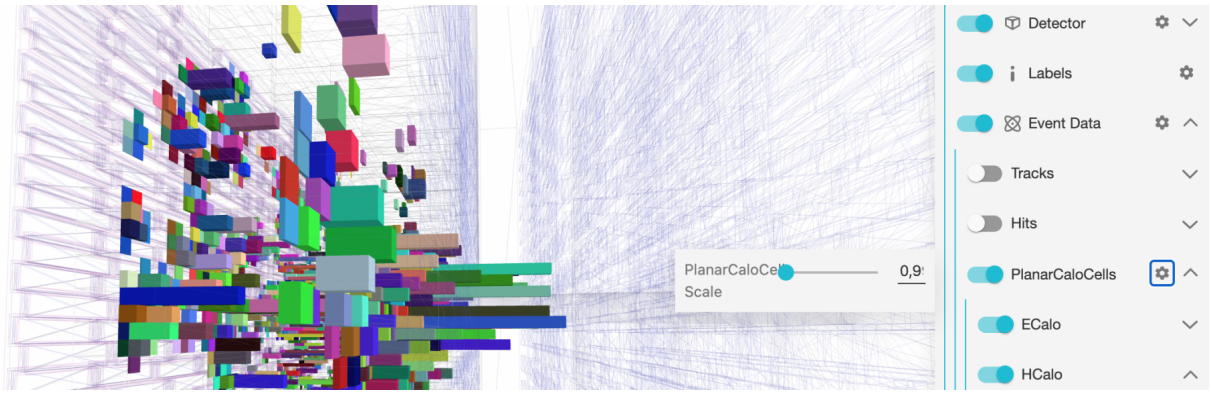

Figure 5. Planar calo cells visualization along with its scale and cut options.

3. Experiment agnostic import of GDML and ROOT geometry

It is common in HEP to describe geometry using ROOT and/or GDML file format. However, Phoenix does not understand GDML out of the box and so conversions need to be made in order to get a proper visualization of that format.

3.1. The ROOT framework

ROOT is an open-source data analysis framework used heavily in HEP. ROOT had already expanded its framework and its way of visualizing data and geometry to the web. In particular it provides a JavaScript API that was used inside Phoenix to understand ROOT geometry format and visualize it. It was thus a tool of choice to translate our LHCb run 3 geometry from the GDML format to Phoenix. The first step thus, was to use the `GDML -> ROOT` export functionality of the ROOT framework.

After the successful translation to ROOT format the geometry can be visualized inside Phoenix. However, the ability to show or hide as well as make transparent or wireframed specific sub parts of the geometry was missing. In other terms, when using ROOT geometry format, the whole geometry appears inside Phoenix as a single large unit. On top of that, large geometry descriptions, i.e. geometry described with large number of volumes, were causing performance issues with Phoenix and there was no way to limit the level of details.

3.2. glTF geometry description

To fix the above situations another geometry description format has been chosen, namely glTF [9]. glTF is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by engines and applications. glTF minimizes the size of 3D assets, and the runtime

processing needed to unpack and use them. It is natively used by Phoenix, e.g. inside the ATLAS experiment. The ROOT geometry format of LHCb was thus translated to glTF using the JavaScript API of the ROOT framework. In a second step a graphical user interface (GUI) web tool was built on top of the ROOT JavaScript API to automate the translation. A simple visit to the URL will upload the ROOT file and translate it to glTF. The glTF file can be split into multiple ones with a few modifications to the code which can be done with ease, by following the documentation of the LHCb geometry translation tool. All the parts then can be visualized inside Phoenix with complete functionality.

The LHCb geometry translation tool is open source and documented [1]. It is kept experiment independent and can translate any ROOT geometry or data chart description to glTF format and visualize it inside Phoenix. Positive feedback has already been received from Fermilab as well as from the FCC experiment who successfully translated and used their new geometry descriptions with our web tool.

4. Conclusion

LHCb has decided to replace its old event visualization application (Panoramix) with a new web based one for run 3. It has adopted the Phoenix framework, a common effort under the HEP software foundation (HSF) hat and all the tools are kept experiment agnostic and reusable. Phoenix uses latest technologies including, three.js, WebGL and Angular [8] and allows to provide a modern user experience, evolving with new technologies. As an example, a virtual reality visualization experience for HEP is being implemented in Phoenix.

On the LHCb side, more event data will be added and visualized, notably muon detector, Upstream Tracker (UT) and SciFi Tracker (FT) hits as well as reconstructed tracks in order to achieve a complete run 3 event display. Future work also includes the possibility of splitting the glTF geometry description into multiple parts, using the new GUI web geometry translation tool, so that the user does not have to edit the code himself, as well as integration with the online environment and the move to production for all LHCb users.

5. References

[1] Pappas A. The lhcb web event data & geometry display, 2021. [Online]. Available: https://lhcb-web-display.app.cern.ch/.

[2] AIDASoft. Detector description toolkit for high energy physics, 2020. [Online]. Available: https://dd4hep.web.cern.ch/.

[3] Binet S Bohak C Couturier B Grasland H Gutsche O Linev S Martyniuk A McCauley T Bellis M, Bianchi M et al. Hep software foundation community white paper working group – visualization. *arXiv:1811.10309 [physics.comp-ph]*, 1:1–37, 2018.

[4] LHCb collaboration. Framework tdr for the lhcb upgrade : technical design report. *CERN-LHCC-2012-007 ; LHCb-TDR*, 12:1–62, 2012.

[5] LHCb collaboration. Upgrade software and computing. *CERN-LHCC-2018-007 ; LHCB-TDR*, 17:1–102, 2018.

[6] Moyse E. Phoenix event display framework vchep, 2021. [Online]. Available: https://indico.cern.ch/event/948465/contributions/4323946/.

[7] Barrand G. Panoramix. In *Vol1 Proceedings of Computing in High Energy and Nuclear Physics 2004*, pages 365–369, Congress Centre, Interlaken, Switzerland, 2005.

[8] Google. The modern web developer's platform, 2016. [Online]. Available: https://angular.io/.

[9] Khronos group. gltf runtime 3d asset delivery, 2015. [Online]. Available: https://www.khronos.org/gltf/.

[10] Bianchi M. Hep software foundation, 2016. [Online]. Available: https://hepsoftwarefoundation.org/.

[11] mrdoob. Threejs javascript 3d library, 2010. [Online]. Available: https://threejs.org/.

[12] Brun R. Root data analysis framework, 1995. [Online]. Available: https://root.cern/.

[13] Khronos WebGL working group. Web graphics library, 2011. [Online]. Available: https://www.khronos.org/webgl/.