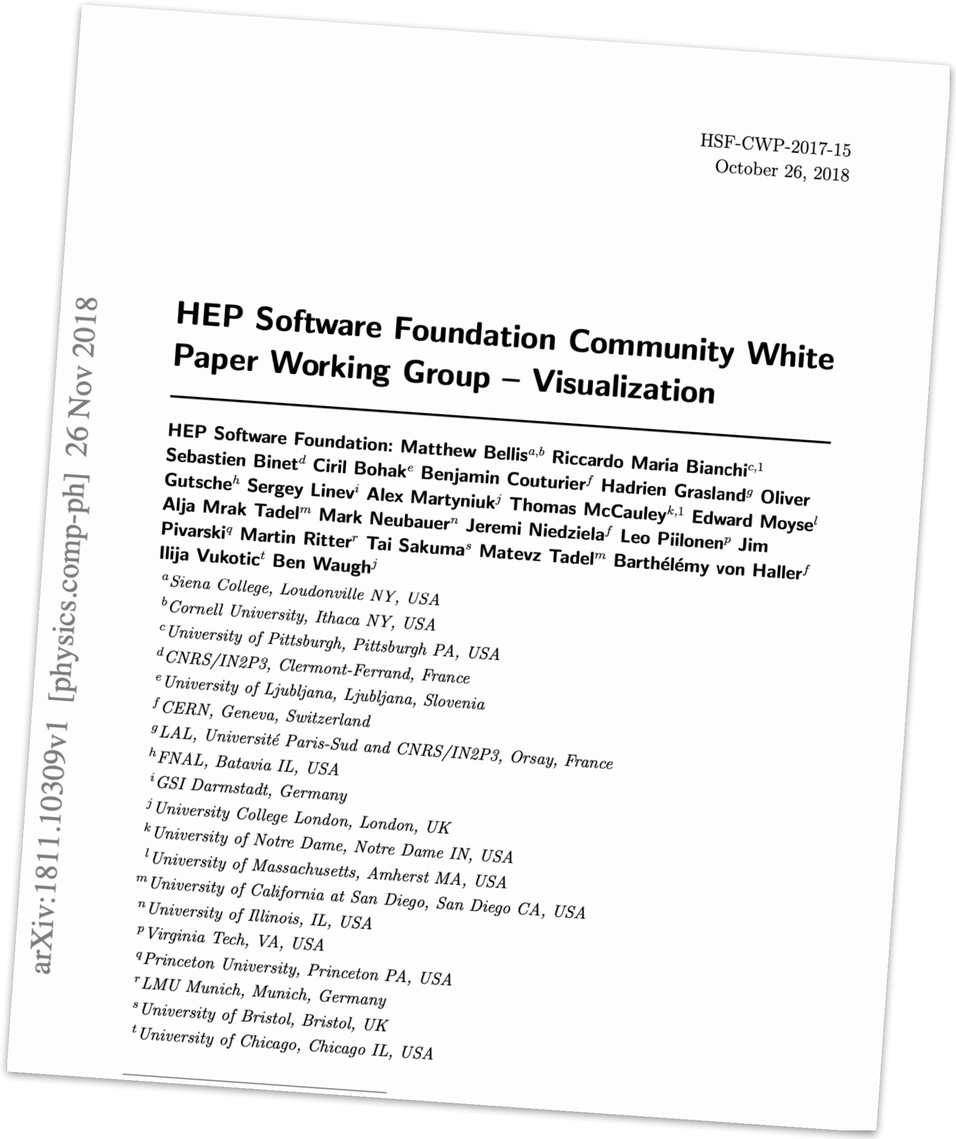
EDWARD MOYSE

THE PHOENIX EVENT DISPLAY

INTRODUCTION

- In 2017 the HSF visualisation white paper identified the need for a **common event format**, and a **common tool** to visualise event data (and geometry)
- Phoenix is an experiment agnostic display, supported by the HSF visualisation group:
  - Repository: https://github.com/HSF/phoenix
    - Open-source + Apache 2.0 license
  - *Runs entirely in the browser*, so no installation required for clients.
  - Scalable and cheap to host (plain website, minimal backend infrastructure)
  - Uses industry standards, such as three.js and angular, nodeJS, NPM (+ other libraries)
    - (Also a demo using reactjs)
- **Extensible by design**
  - Currently has built in support for LHCb, ATLAS, CMS, TrackML , EDM4HEP **geometry** and/or **event data**
  - Currently used by ATLAS, FCC, LHCb, Belle-II, and now EIC(?)
    - (see documentation)



HSF-CWP-2017-15
October 26, 2018

arXiv:1811.10309v1 [physics.comp-ph] 26 Nov 2018

HEP Software Foundation Community White Paper Working Group – Visualization

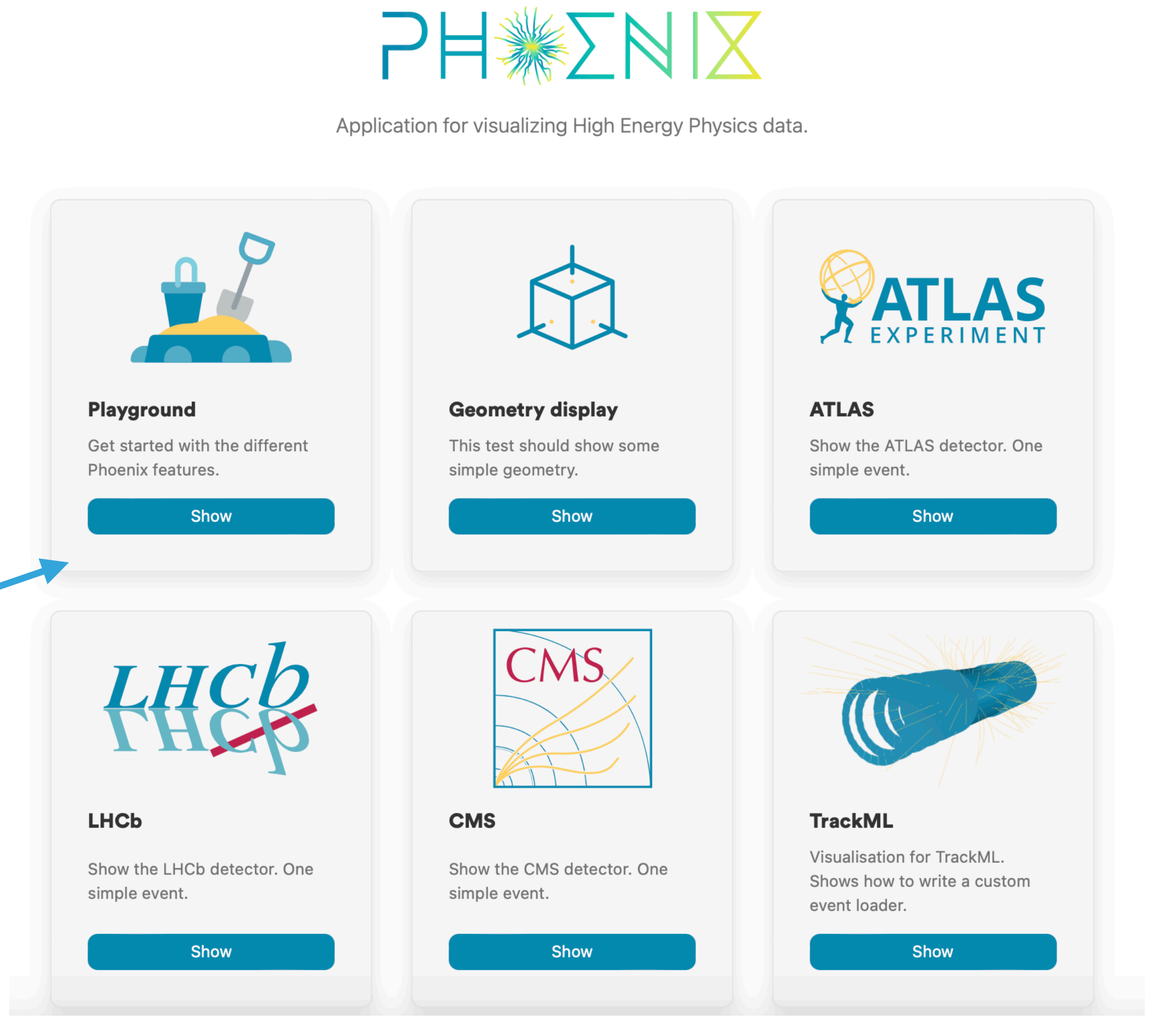
HEP Software Foundation: Matthew Bellis[a,b] Riccardo Maria Bianchi[c,1] Sebastien Binet[d] Ciril Bohak[e] Benjamin Couturier[f] Hadrien Grasland[g] Oliver Gutsche[h] Sergey Linev[i] Alex Martyniuk[j] Thomas McCauley[k,1] Edward Moyse[l] Alja Mrak Tadel[m] Mark Neubauer[n] Jeremi Niedziela[f] Leo Piilonen[p] Jim Pivarski[q] Martin Ritter[r] Tai Sakuma[s] Matevz Tadel[m] Barthélémy von Haller[f] Ilija Vukotic[t] Ben Waugh[j]

[a] Siena College, Loudonville NY, USA
[b] Cornell University, Ithaca NY, USA
[c] University of Pittsburgh, Pittsburgh PA, USA
[d] CNRS/IN2P3, Clermont-Ferrand, France
[e] University of Ljubljana, Ljubljana, Slovenia
[f] CERN, Geneva, Switzerland
[g] LAL, Université Paris-Sud and CNRS/IN2P3, Orsay, France
[h] FNAL, Batavia IL, USA
[i] GSI Darmstadt, Germany
[j] University College London, London, UK
[k] University of Notre Dame, Notre Dame IN, USA
[l] University of Massachusetts, Amherst MA, USA
[m] University of California at San Diego, San Diego CA, USA
[n] University of Illinois, IL, USA
[p] Virginia Tech, VA, USA
[q] Princeton University, Princeton PA, USA
[r] LMU Munich, Munich, Germany
[s] University of Bristol, Bristol, UK
[t] University of Chicago, Chicago IL, USA

SCOPE OF THIS TALK

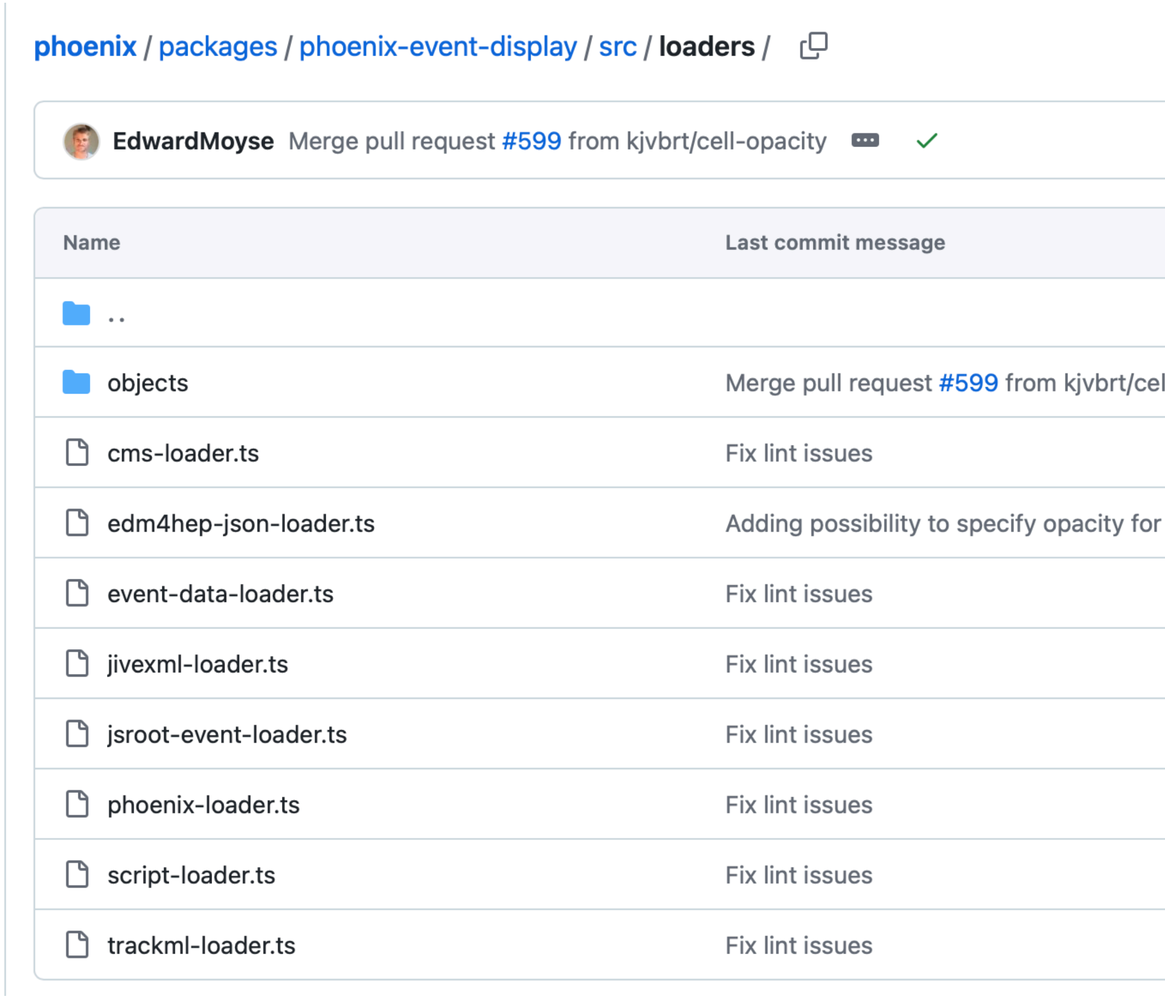
- Will divide talk into two broad sections:
  - *Some of the capabilities of Phoenix*
  - *Some examples of how it is used by the community*

- This talk will necessarily not be able to cover all details, please check out the demo:
  - https://hepsoftwarefoundation.org/phoenix
  - Splash screen shows various demonstration implementations
    - Shows flexibility and ensures that Phoenix developments continue to support a wide range of detectors



SUPPORTED EVENT DATA FORMATS

- Phoenix internally makes use of a JSON format to represent event data. The JSON format is designed to be human-readable, but still compact.
- We also provide “loaders” to convert from arbitrary formats to phoenix format…
  - ATLAS (JiveXML)
  - TrackML
  - EDM4Hep
  - …



SUPPORTED GEOMETRY

- Phoenix can display geometry stored in many standard formats:
  - Natively supported formats are OBJ, **glTF** , ROOT, json(gz)
    - Compressed glTF (glb) is recommendation
      - Preferred by **threejs**
      - Small file size
      - Phoenix can automatically populate the detector menu (see next slide) with the embedded hierarchy
        - (see the docs for more information)
    - However `threejs` supports a ***HUGE*** number of 3D formats, so any of these could easily be added
- We also have a workflow (described here) for how to convert from GDML to ROOT to glTF/glb
- **Aside**: the ACTS project can output OBJ format geometry, so easy to display in Phoenix

dev · three.js / examples / jsm / loaders / · Go to file · Add file · ···

karimi and fraguada Returning conversion warnings in 3DMLoader (#21639) ··· ✓ 8fa6227 5 hours ago History

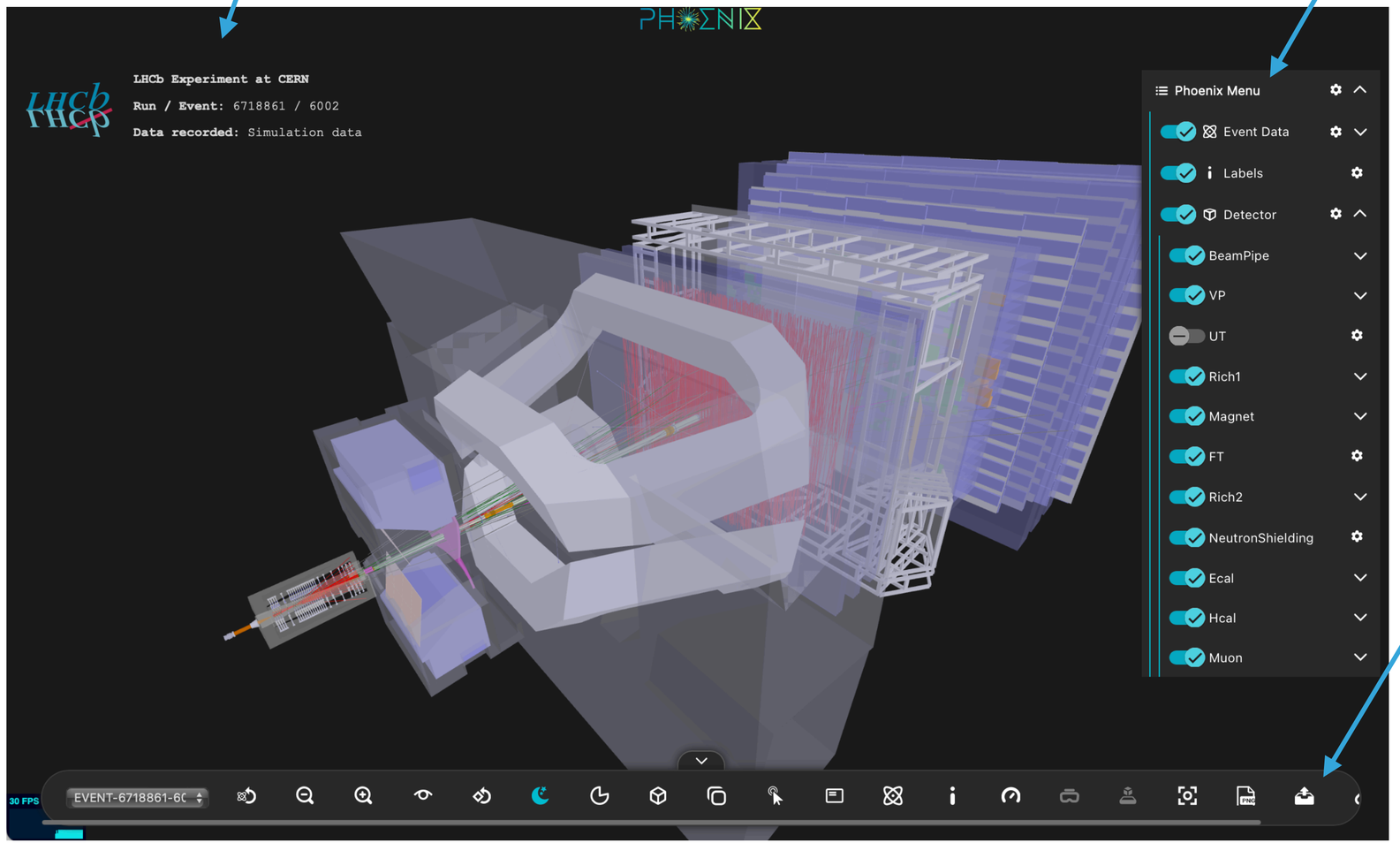
..		
ifc	Make WASM path configurable + Update IFC library (#21683)	28 days ago
lwo	Run lint fix on js and jsm files	5 months ago
3DMLoader.js	Returning conversion warnings in 3DMLoader (#21639)	5 hours ago
3MFLoader.js	Examples: Update fflate version (#21669)	29 days ago
AMFLoader.js	Examples: Update fflate version (#21669)	29 days ago
BVHLoader.js	Examples: Convert loaders to ES6 Part I. (#21612)	last month
BasisTextureLoader.js	Examples: Convert loaders to ES6 Part I. (#21612)	last month
ColladaLoader.js	Material: Remove skinning. (#21788)	13 days ago
DDSLoader.js	Examples: Convert loaders to ES6 Part I. (#21612)	last month
DRACOLoader.js	Examples: Convert loaders to ES6 Part III. (#21616)	last month
EXRLoader.js	Examples: Update fflate version (#21669)	29 days ago
FBXLoader.js	Material: Remove skinning. (#21788)	13 days ago
GCodeLoader.js	Fixed eslint errors for examples (#21842)	21 hours ago
GLTFLoader.js	GLTFLoader: Ignore redundant 'KHR_texture_transform' extensions and '...	6 days ago
HDRCubeTextureLoader.js	Examples: Convert loaders to ES6 Part II. (#21614)	last month
IFCLoader.js	Fixed eslint errors for examples (#21842)	21 hours ago
KMZLoader.js	Examples: Update fflate version (#21669)	29 days ago
KTX2Loader.js	KTX2Loader: Update ktx-parse dependency, import enums. (#21567)	2 months ago
KTXLoader.js	Examples: Convert loaders to ES6 Part II. (#21614)	last month
LDrawLoader.js	Examples: Clean up. (#21632)	last month
LUT3dlLoader.js	update LUTPas	4 months ago
LUTCubeLoader.js	update LUTPas	4 months ago
LWOLoader.js	Examples: Convert loaders to ES6 Part II. (#21614)	last month
https://github.com/mrdoob/three.js/tree/dev/examples/jsm/loaders		onth
MD2Loader.js	Fixed eslint errors for examples (#21842)	21 hours ago

INTERFACE – A TYPICAL PHOENIX SETUP

EXPERIMENT LOGO + EVENT INFORMATION

PHOENIX MENU

ICON BAR



Responsible for determining what event data or geometry is visible

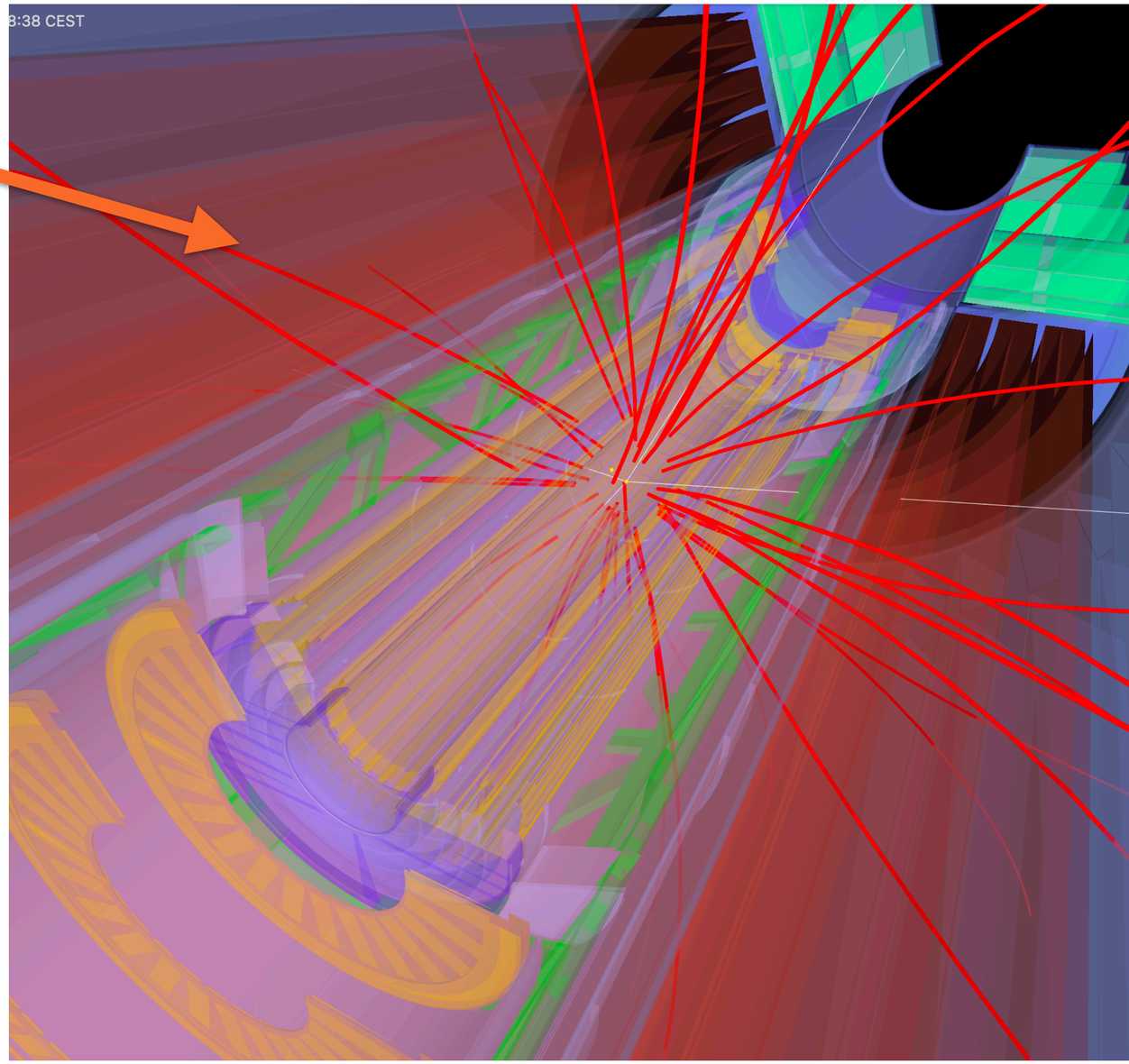
- Can apply filters (“cuts”) to event data
- Can control event data / geometry appearance
- Save/load configuration

Responsible for interactions

- Can open dialogs for more controls + ways to navigate the scene
- Animations, zooms, performance mode, VR
- Screenshots, loading saving events, geometry

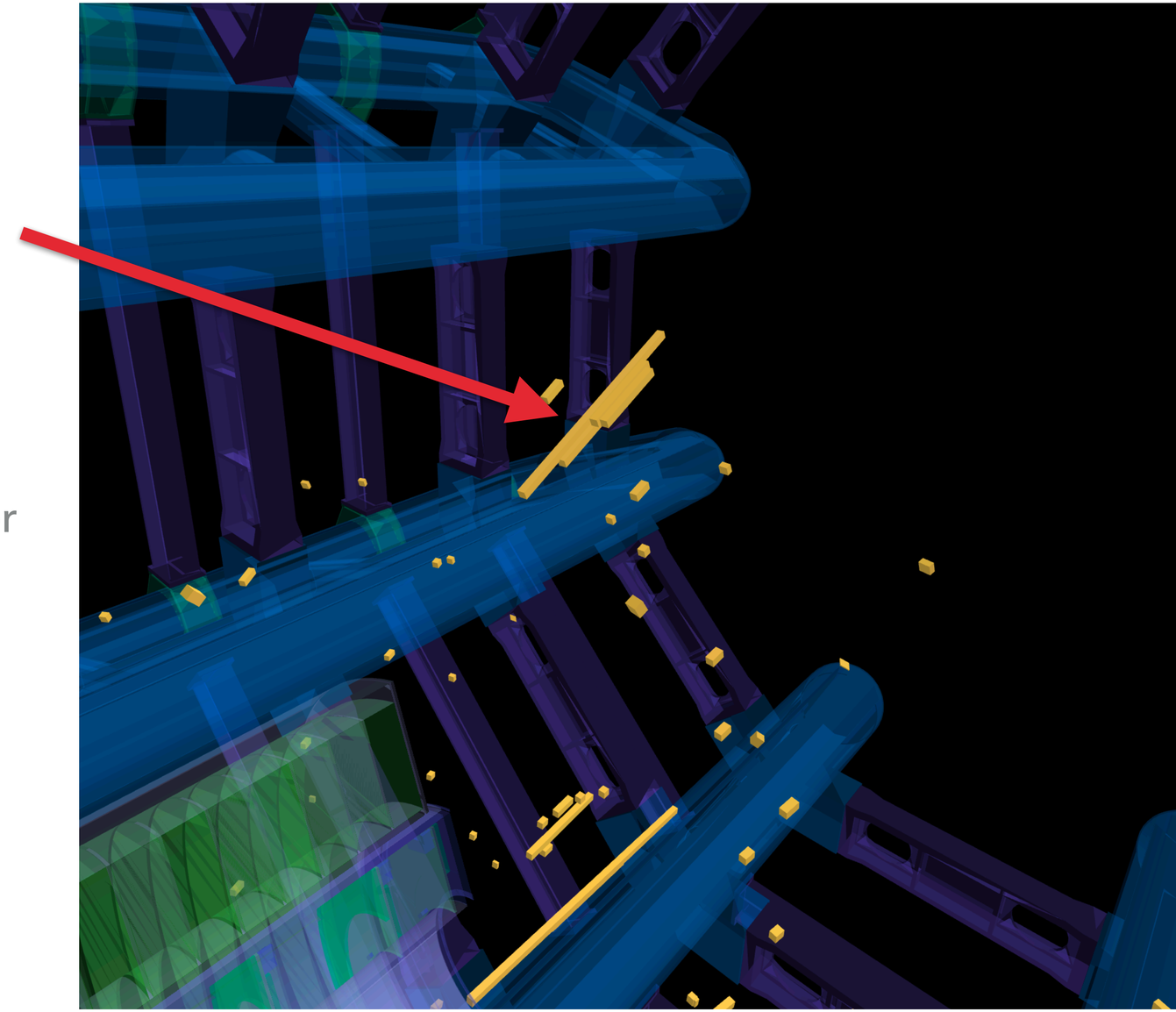
SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and ‘Clusters’)
- **Missing energy**



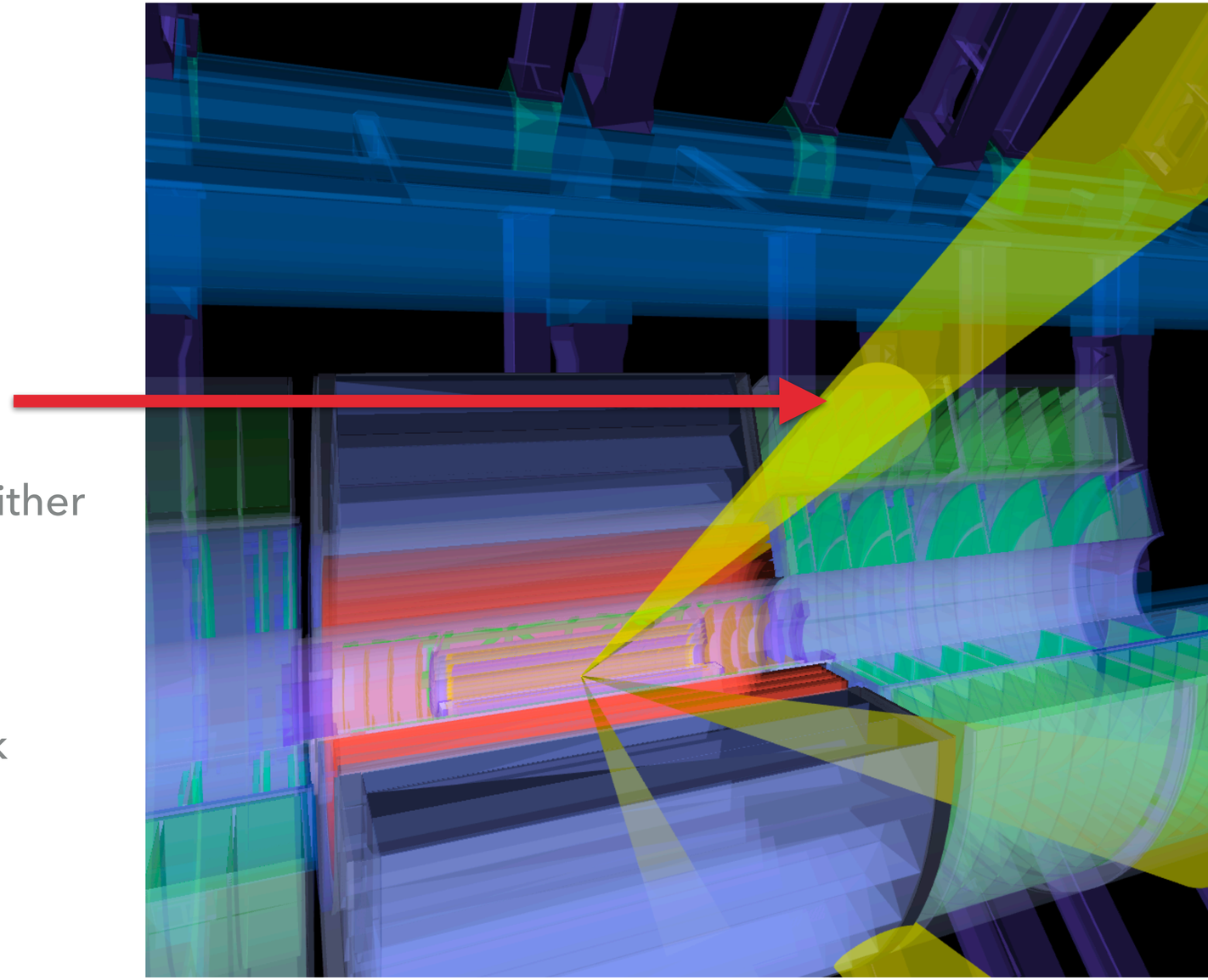
SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and ‘Clusters’)
- **Missing energy**



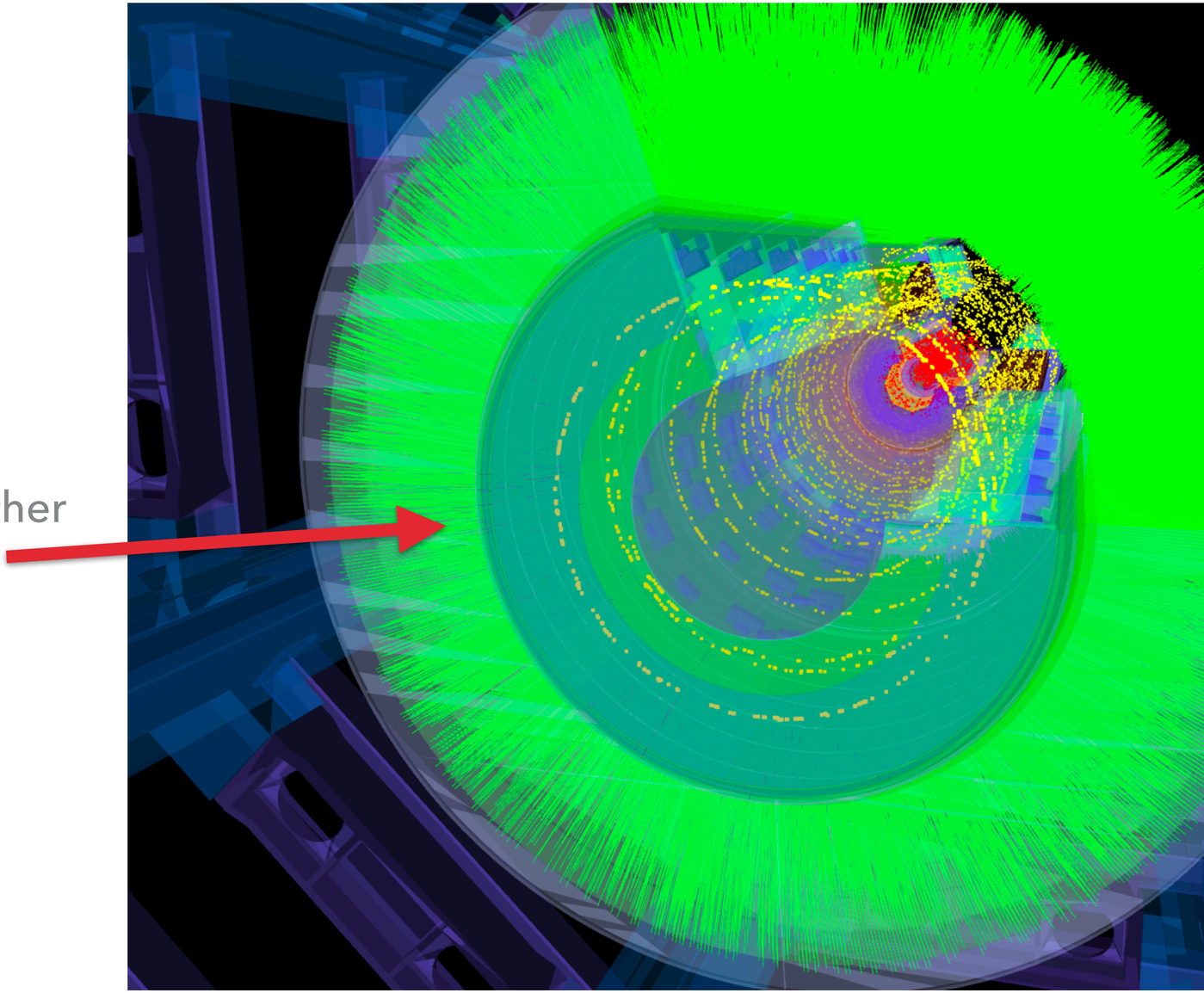
SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and ‘Clusters’)
- **Missing energy**



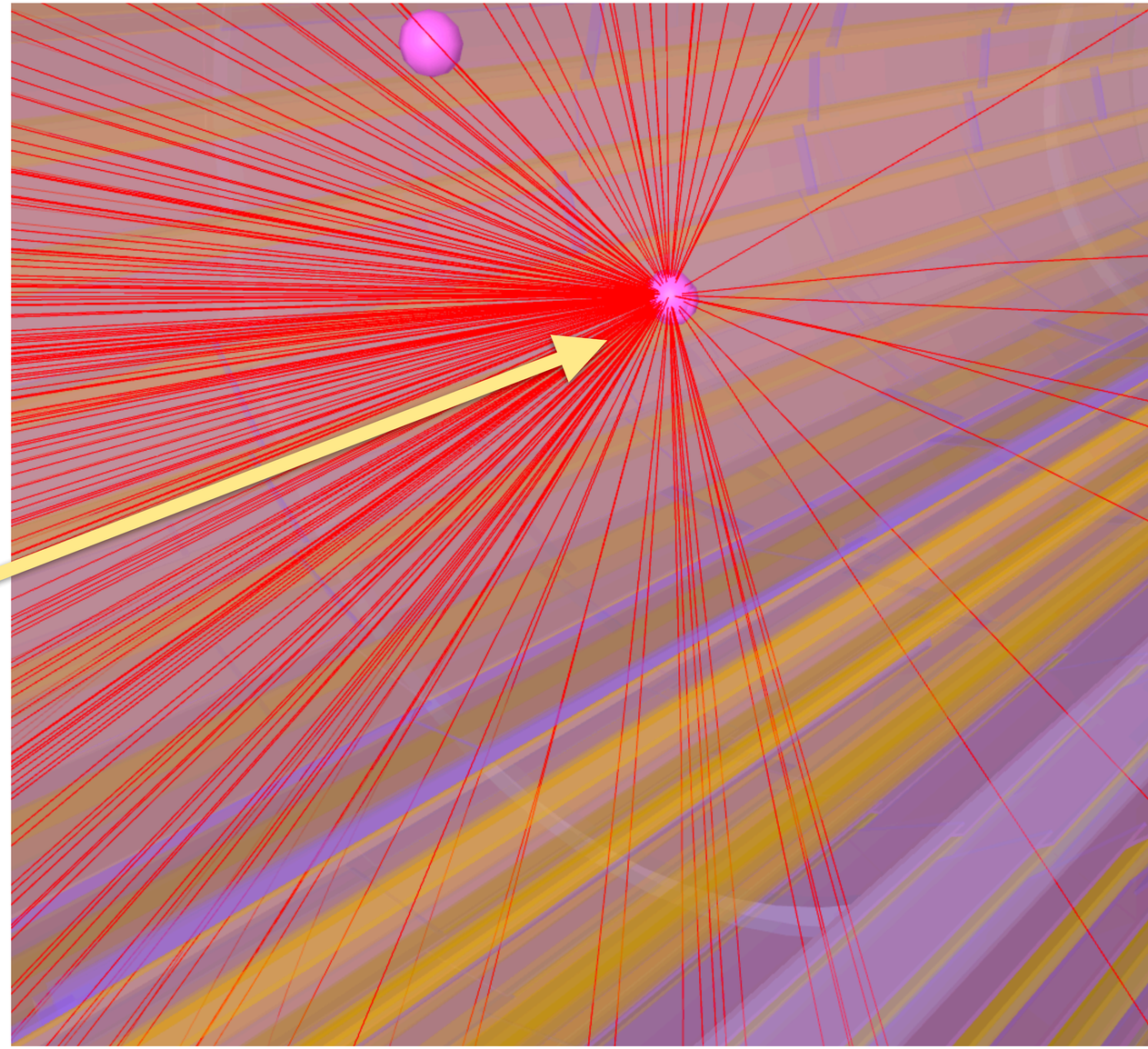
SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and ‘Clusters’)
- **Missing energy**



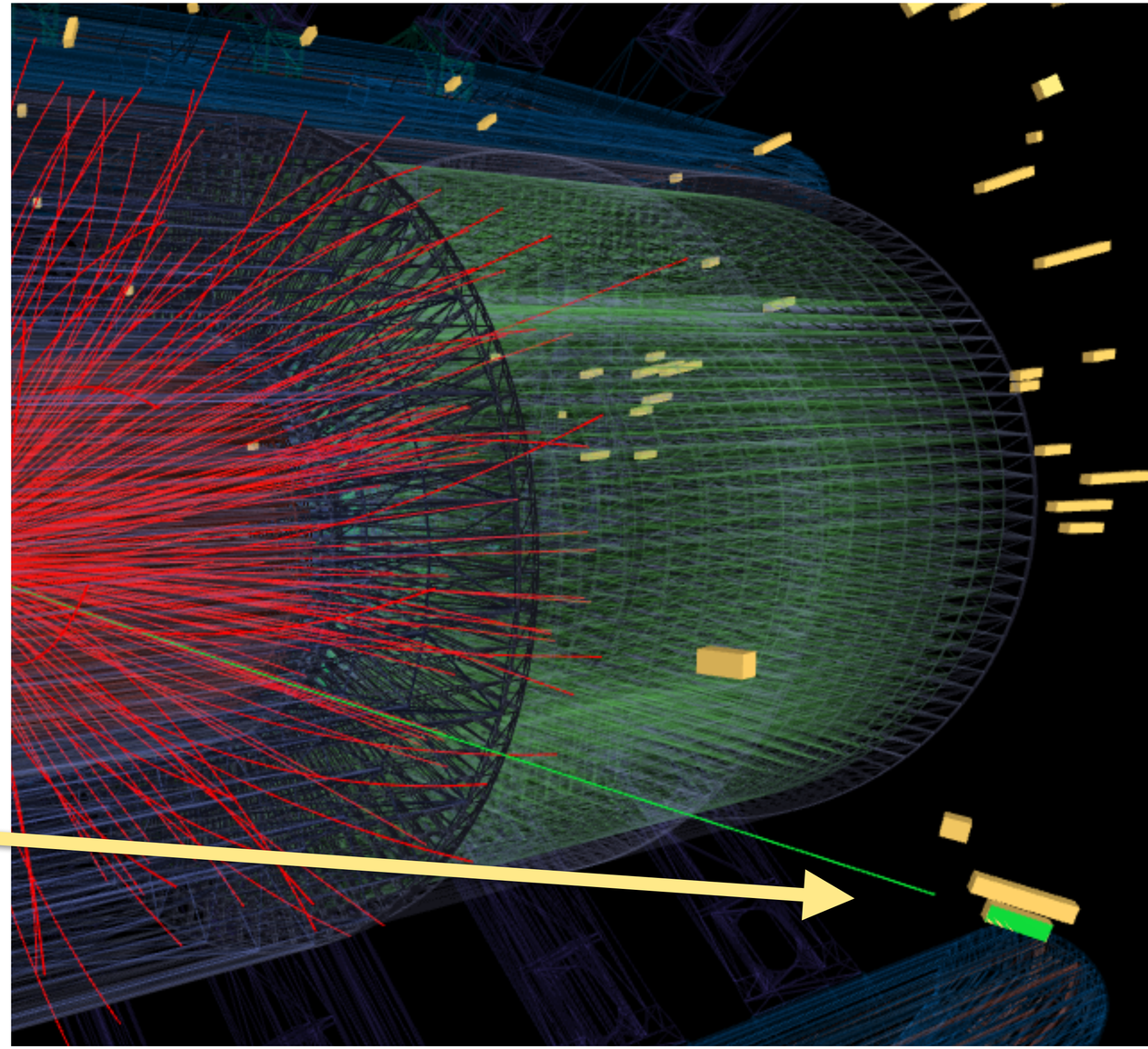
SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and ‘Clusters’)
- **Missing energy**



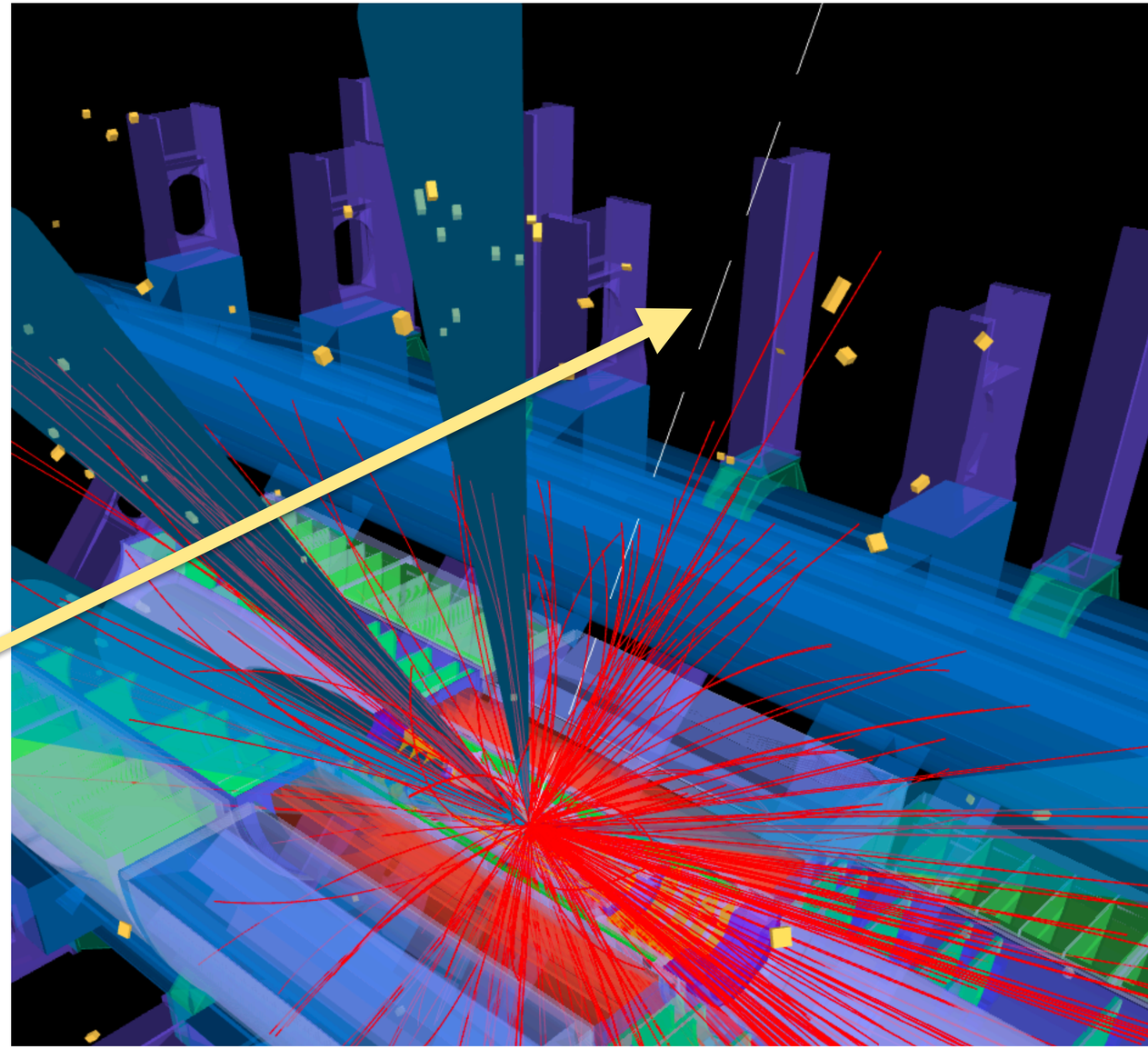
SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and ‘Clusters’)
- **Missing energy**



SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity within the detector
- **Hits** - individual measurements, which can either be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and 'Clusters')
- **Missing energy**

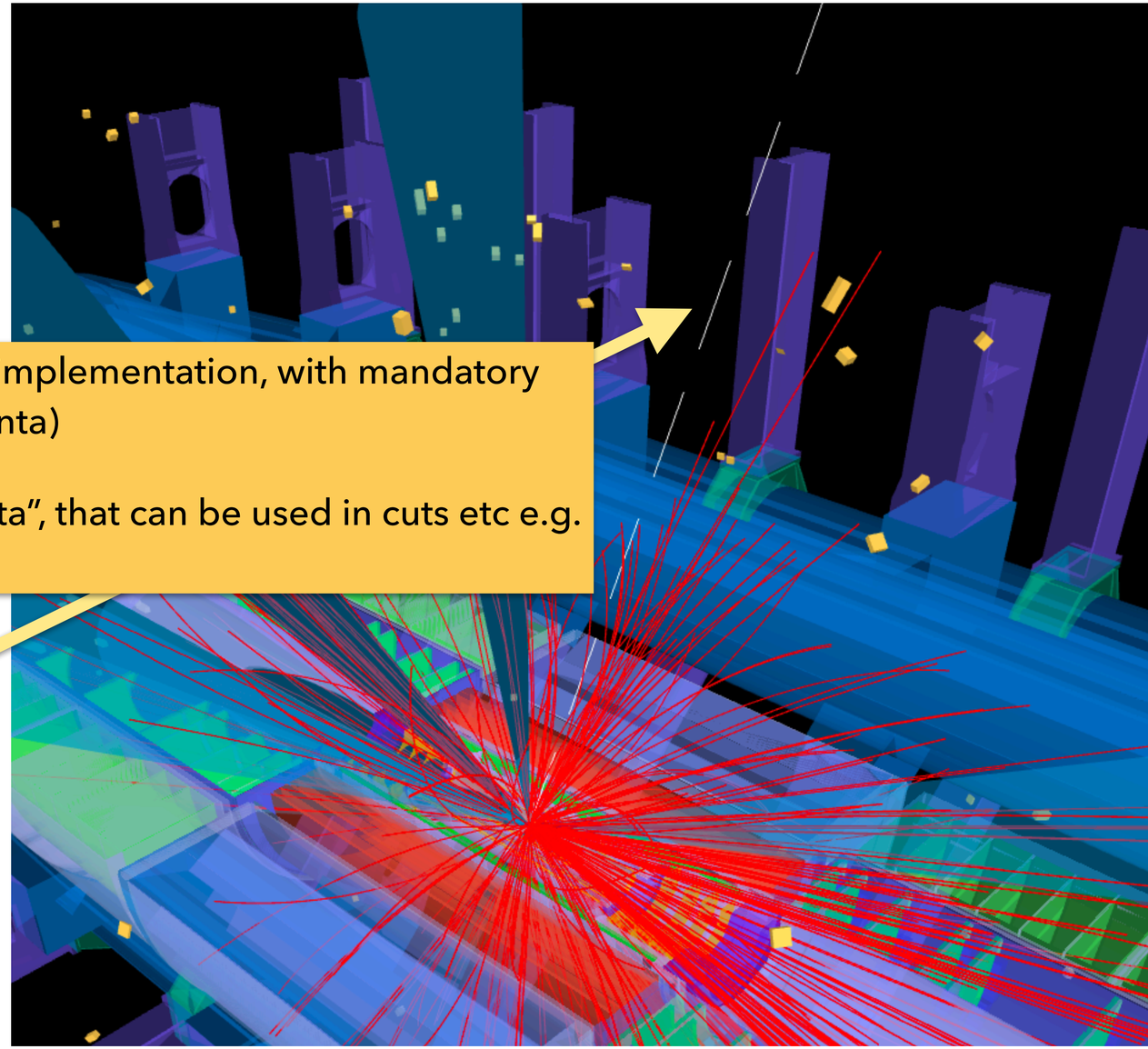


SUPPORTED PHYSICS OBJECTS

- **Tracks** - the trajectory of a charged particle (usually in a magnetic field)
- **Calorimeter cells** - deposits of energy in a calorimeter (planar and cylindrical are supported).
- **Jets** - cones of activity v
- **Hits** - individual measur
  be points or lines
- **Vertices** - optionally linked to tracks
- **Compound objects** (e.g. 'Muons', which link 'Tracks' and 'Clusters')
- **Missing energy**

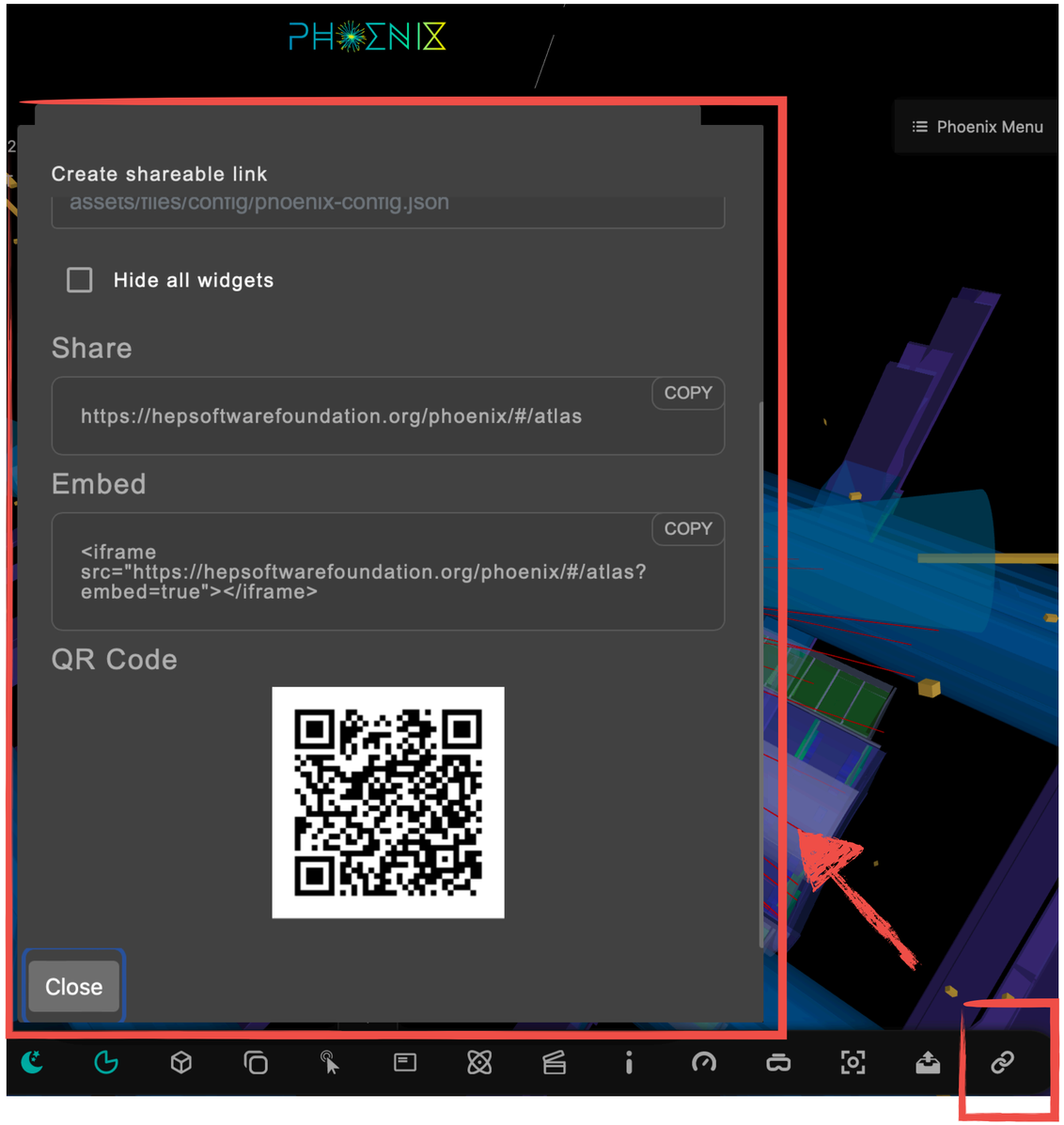
All these objects have a standard implementation, with mandatory information (e.g. positions, momenta)

All can be extended with "user data", that can be used in cuts etc e.g. time of flight information



SHAREABLE URLS

- Clicking on the **link button** in the menu bar opens a dialog which provides you with a shareable link
  - For example, for **outreach**, you can give a URL which opens Phoenix with a predefined event and configuration
    - Allows you to frame the physics and geometry you want to show
  - Can also generate a QR code, for e.g. posters (see next slide)
- Also get an embeddable link, optionally with limited GUI
  - Useful for e.g. Physics briefing - instead of a static event display, you have a rotating, animated (and interactive) one
  - See for example, [Heavyweight champions: a search for new heavy W′ bosons with the ATLAS detector]



QR CODE IN A POSTER

New Web Based Event Data and Geometry Visualization for LHCb

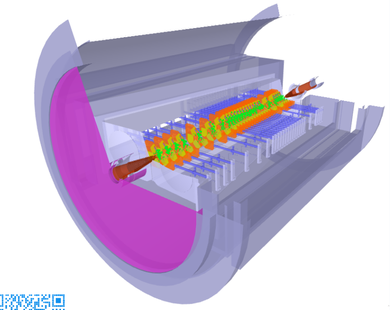
Andreas Pappas[1] Ben Couturier[2] Sebastien Ponce[2]
[1]National & Kapodistrian University of Athens, [2]CERN
andreas.pappas@cern.ch, ben.couturier@cern.ch, sebastien.ponce@cern.ch

Key Features

- Generic & reusable by the community
- Ubiquitous web-based solution built on the Phoenix framework
- Generic translation tool from GDML & ROOT to Phoenix format
- Usability in outreach & data tracking
- Collaboration between High Energy Physics (HEP) experiments, Phoenix & ROOT projects

Experiment Agnostic Import of ROOT Geometry

- Easily convert your GDML and ROOT geometry files into PHOENIX format



- Allows usage of GDML and ROOT geometry descriptions in a generic browser based solution

Fig. 1: A closer look on the VErtex LOcator (VELO)

Collaborative Contributions

- LHCb joins the HEP Software Foundation (HSF) common effort on event displays, providing state of the art, web based solutions
- Provided a generic converter for standard geometry, making Phoenix the perfect generic solution for any High Energy Physics (HEP) experiment and others
- The geometry converter is integrating the different ROOT framework tools, notably in GDML to ROOT and three.js conversion
- **Kudos** to the ROOT team for their amazing framework

Conclusion

By putting work in common effort with the other communities, all our tools are kept generic and reusable with ease by every High Energy Physics (HEP) experiment and not only.
By making use of latest web technologies such as three.js, WebGL & Angular, room is kept as well for improvements. For example on the Phoenix side a Virtual Reality Visualization implementation is about to come.
On the LHCb side, more event data will be added, notably muon, UT & FT hits. Also, the web display will move to production for all LHCb users and be integrated with the online environment.
Details can be found in the paper itself, but feel free to ask questions and open room for discussions!
Thank you!

PHOENIX Visualization Framework

- Phoenix framework is experiment agnostic and supported by HEP Software Foundation (HSF)
- Used by several experiments including ATLAS & LHCb
- **Kudos** to the Phoenix team for their amazing framework and support

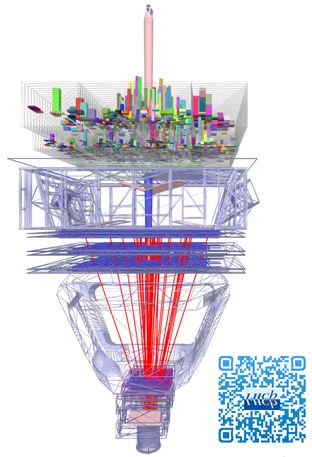


Fig. 2: Phoenix Visualization of an LHCb event (Wireframed)

- Easily extensible, allowing to add and visualize any event data with ease
- Extended the set of visualization primitives, notably for calorimeter hits
- Improved performance of LHCb geometry visualization
- Allows to visualize HEP experiments in Augmented Reality

References

1. The LHCb Web Event Data & Geometry Display (https://lhcb-web-display.app.cern.ch/)
2. The Phoenix Event Display Framework vCHEP 2021 (https://indico.cern.ch/event/948465/contributions/4323946/)
3. HEP Software Foundation Community White Paper Working Group — Visualization (https://arxiv.org/abs/1811.10309)
4. Framework TDR for the LHCb Upgrade, CERN-LHCC-2012-007 ; LHCb-TDR-12 (https://cds.cern.ch/record/1443882)
5. Upgrade Software and Computing Technical Design Report, CERN-LHCC-2018-007 ; LHCB-TDR-017 (https://cds.cern.ch/record/2310827)
6. HEP Software Foundation (https://hepsoftwarefoundation.org/)
7. G. Barrand, PANORAMIX, proceedings of 14th International Conference on Computing in High-Energy and Nuclear Physics (2005) (http://cds.cern.ch/record/688747/files/CERN-2005-002-V1.pdf?version=2)
8. ROOT (https://root.cern/)
9. DD4hep (https://dd4hep.web.cern.ch/dd4hep/)
10. Three.js (https://threejs.org/)
11. WebGL (https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API)

QR code link

ACAT 2021 link

EMBEDDED PHOENIX IN A PHYSICS BRIEFING

- Not a jet all the way: is dark matter hiding in plain sight?

The search for semi-visible jets very challenging, as its event signature can also arise due to mis-measured jets in the detector. How did ATLAS researchers overcome this challenge?

This novel result sets the first limits on this specific semi-visible-jet production scenario, shown in Figure 2 as functions of both the mediator mass and the invisible fraction. The search is more sensitive at intermediate values of the invisible fraction, and excludes mediator masses up to 2.7 TeV. Researchers were also able to report the number of data events observed corresponding to the event selection requirements. This sets important groundwork for future searches for dark matter, enabling physicists to build semi-visible-jet models that take into account the existing constraints on this signature.

There's still a lot left to explore! ATLAS researchers plan to systematically study all possible signatures from the strong dark sector landscape, which may include uncovered signatures like the one considered in this search. As the ATLAS experiment continues to grow its mammoth dataset, it will provide new opportunities for exploration and new options to extend the search for semi-visible jets.

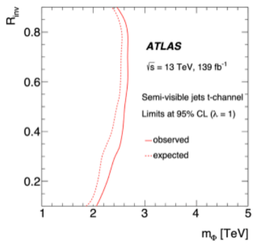
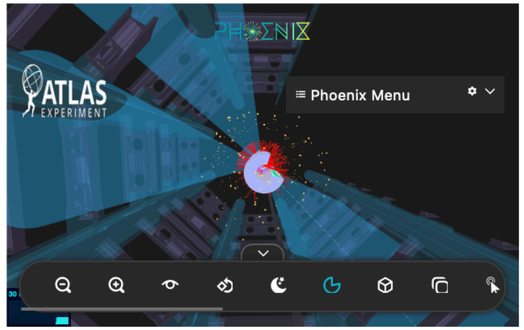


Figure 2: The expected and observed exclusion contours for semi-visible jets signal as a function of mediator mass on the x-axis, and the invisible fraction on the y-axis. Mediator masses on the left hand side of the red solid line are excluded for the given invisible fraction. (Image: ATLAS Collaboration/CERN)

About the event display: A representative semi-visible-jet signal-like event collected in the 2016 ATLAS dataset. The event has four jets (white shaded areas) and missing energy direction aligned along the sub-leading jet direction, as shown by the red line. Yellow bars correspond to the energy depositions in the calorimeters and the charged particles are shown as cyan lines. (Image: ATLAS Collaboration/CERN)

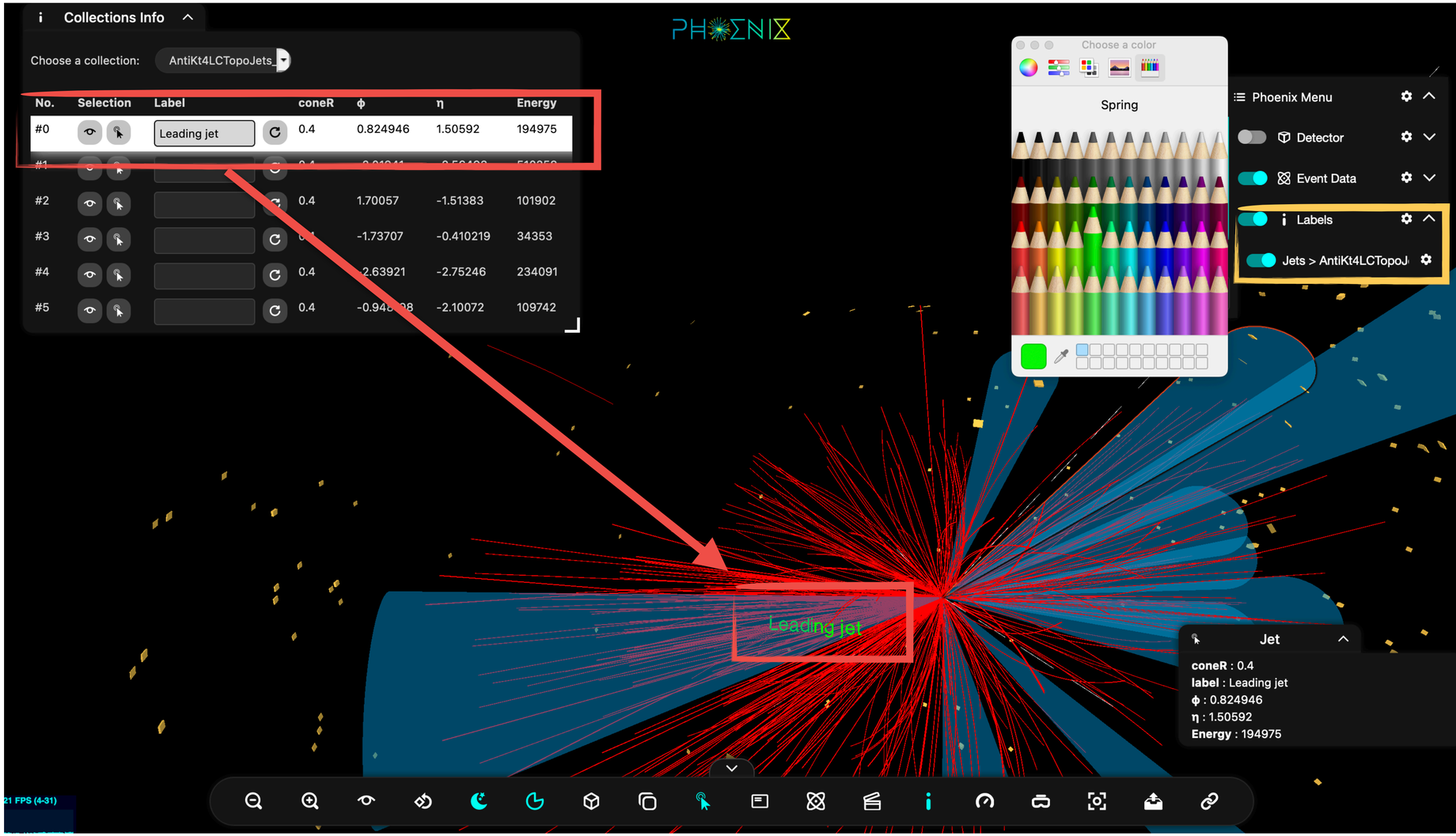
Explore the interactive event display



Dynamic view of the semi-visible-jet signal-like event. The jets are

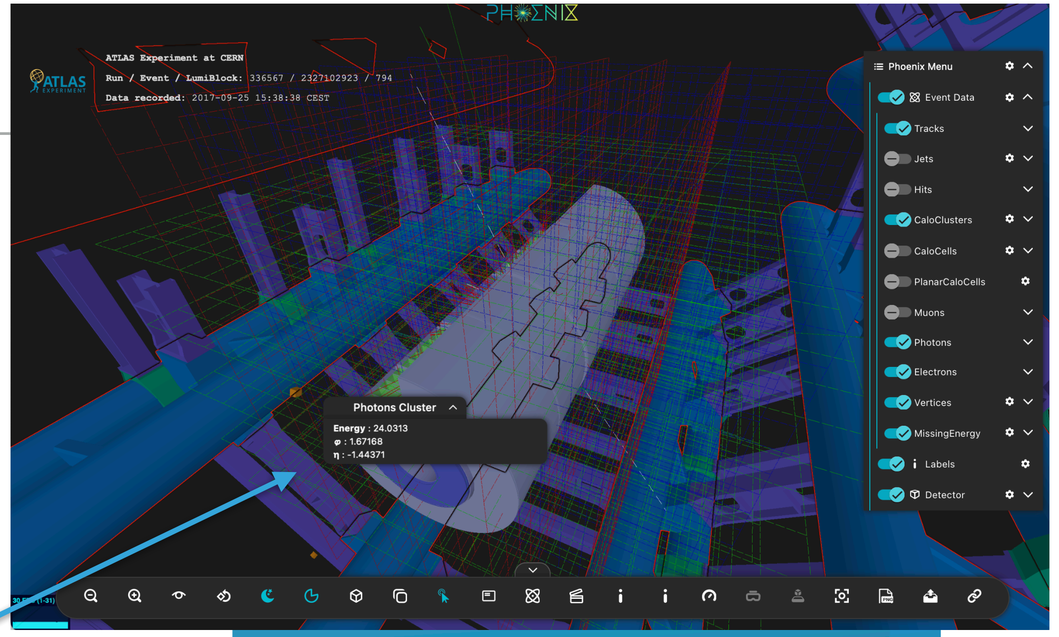
LABELS

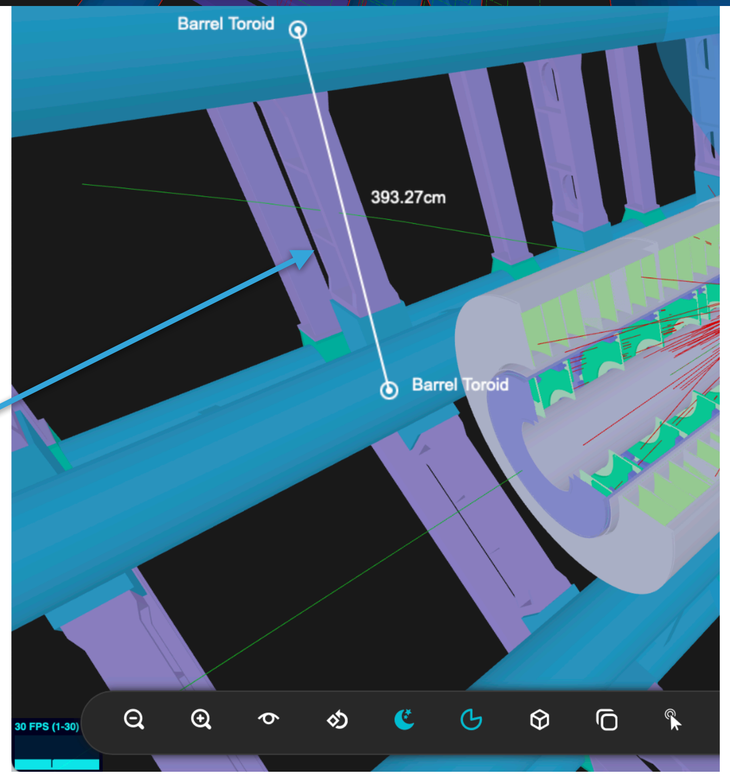
- Physics objects can be given **labels**:
  - Added in collection dialog
  - Dedicated entry in menu, to turn off/on, change colour etc



OTHER FEATURES

- Many more features I do not have time to go into:
  - All physics objects can be filtered (with “cuts”), geometry can be sliced
  - Can select objects in collections browser, and jump to them in 3D
  - Objects can be selected, and information about them displayed
  - Overlay view to e.g. show zoomed view and full view on one event display
  - Rudimentary VR/AR mode
  - 2023 GSOC project was about improving navigation in 3D:
    - Many ways to measure, orient yourself in the scene
- Please check the documentation to learn more.





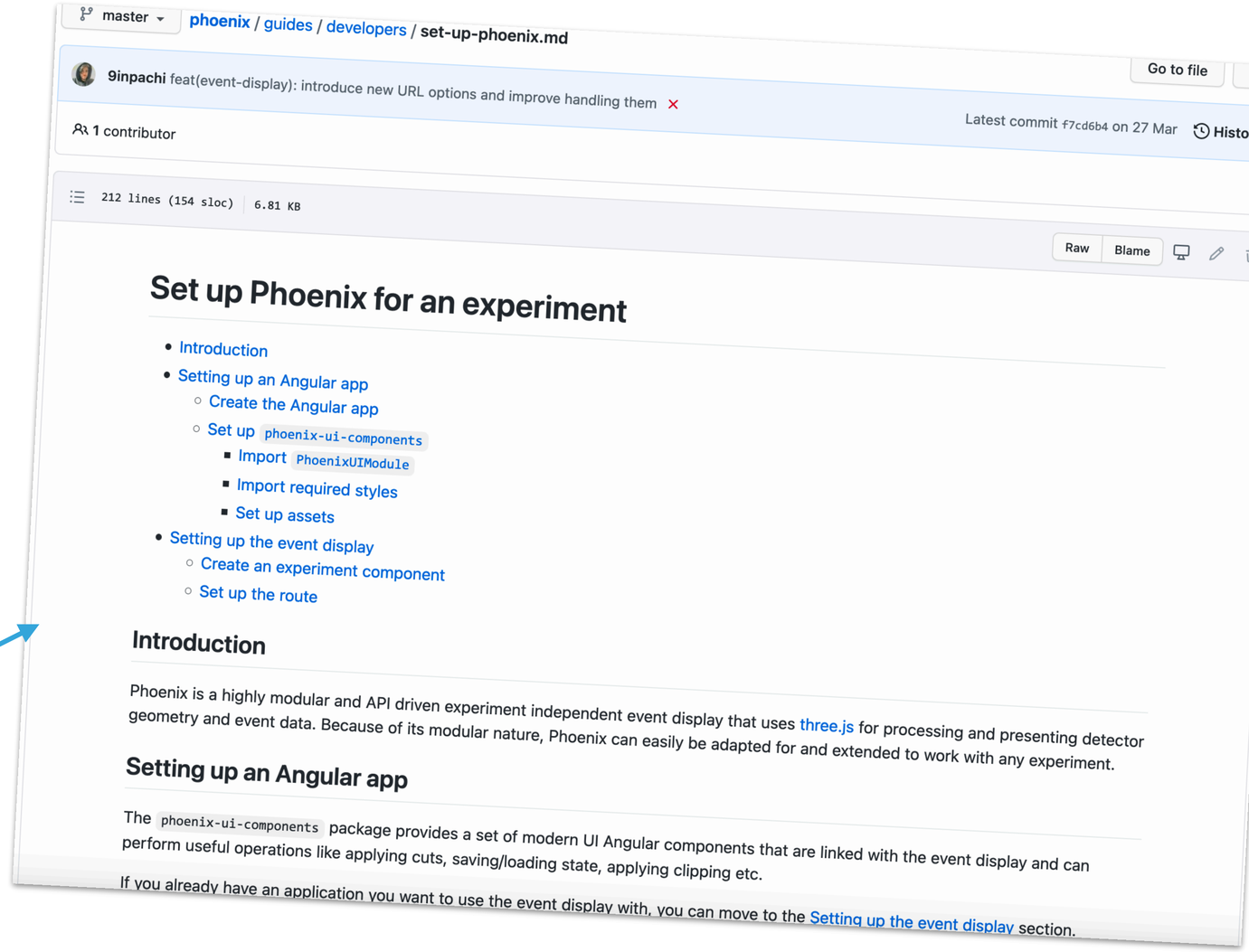
- *Some examples of how it is used by the community…*

EXAMPLES OF PHOENIX

- Can just fork and install Phoenix as-is, but many experiments use Phoenix as part of an *entirely independent application* i.e.
  - Your own repository, your own default configuration etc
  - Phoenix loaded as libraries
  - This way you can e.g. remove the Phoenix splash screen with demos for other experiments, make experiment-specific modifications
- We have detailed instructions, but main step is:

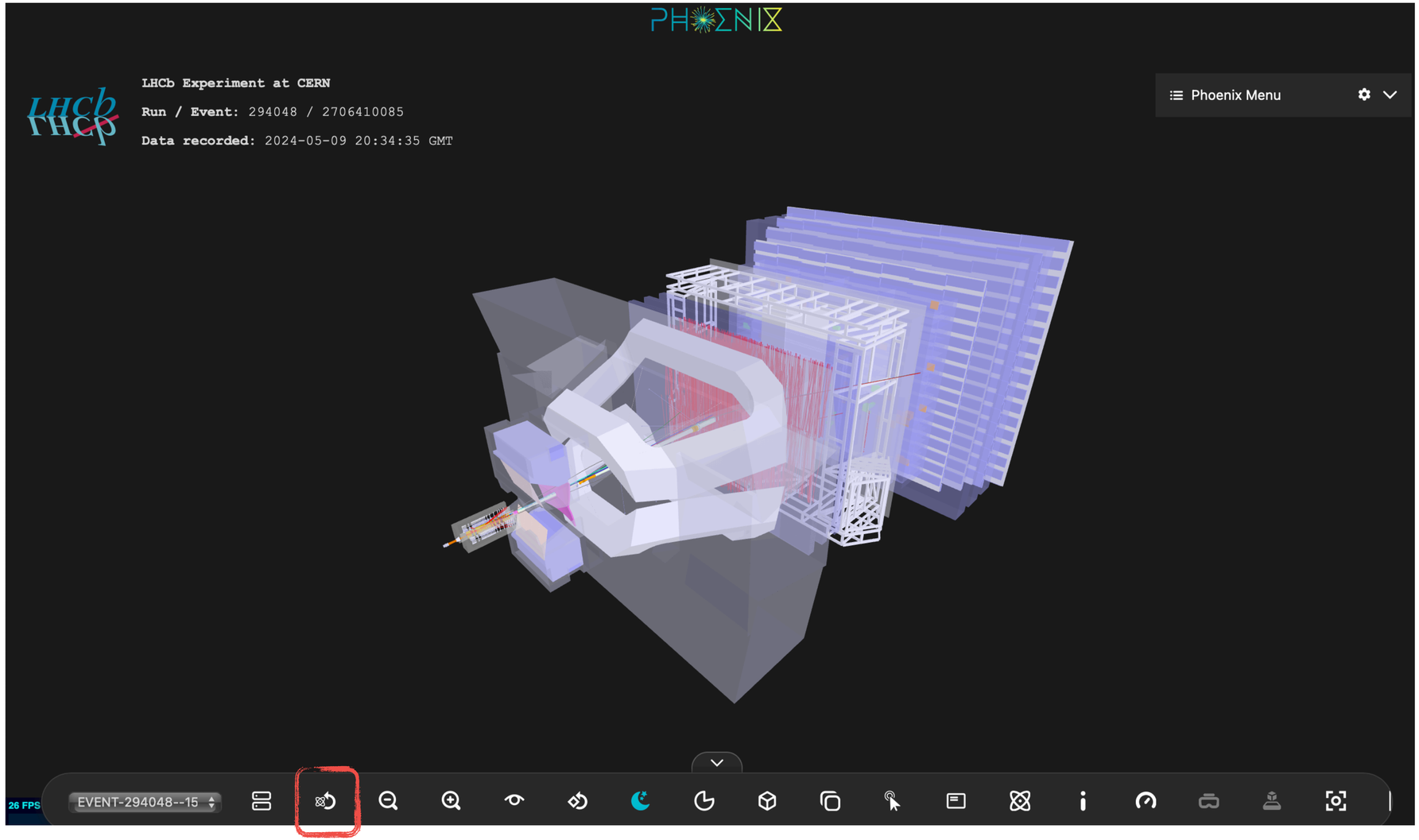
```
npm install phoenix-ui-components
npm install phoenix-event-display
```

- *(Backup slides have some detailed examples of creating a loader etc)*

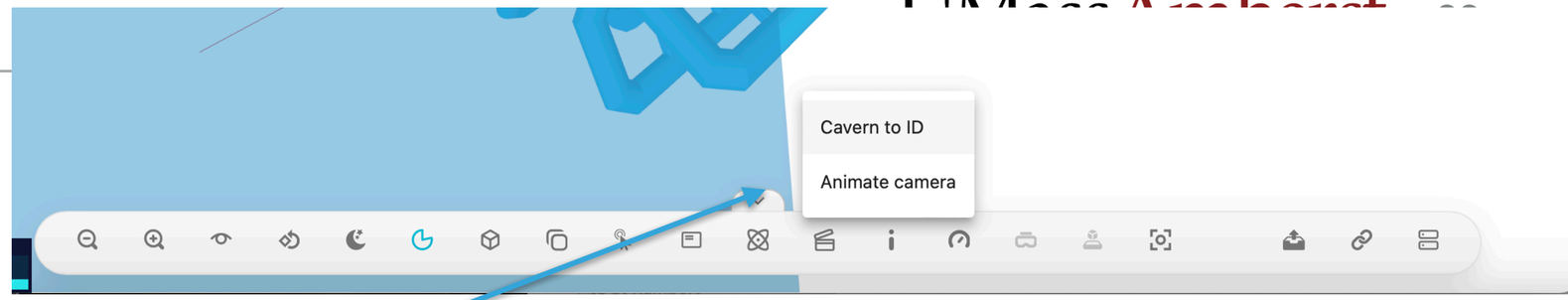


https://lhcb-eventdisplay.web.cern.ch/

- LHCb opted to remove the splash screen:
  - Clicking on the link takes you directly to a view similar to this
  - Menu tweaked to suit LHCb needs
    - E.g. cycle between events

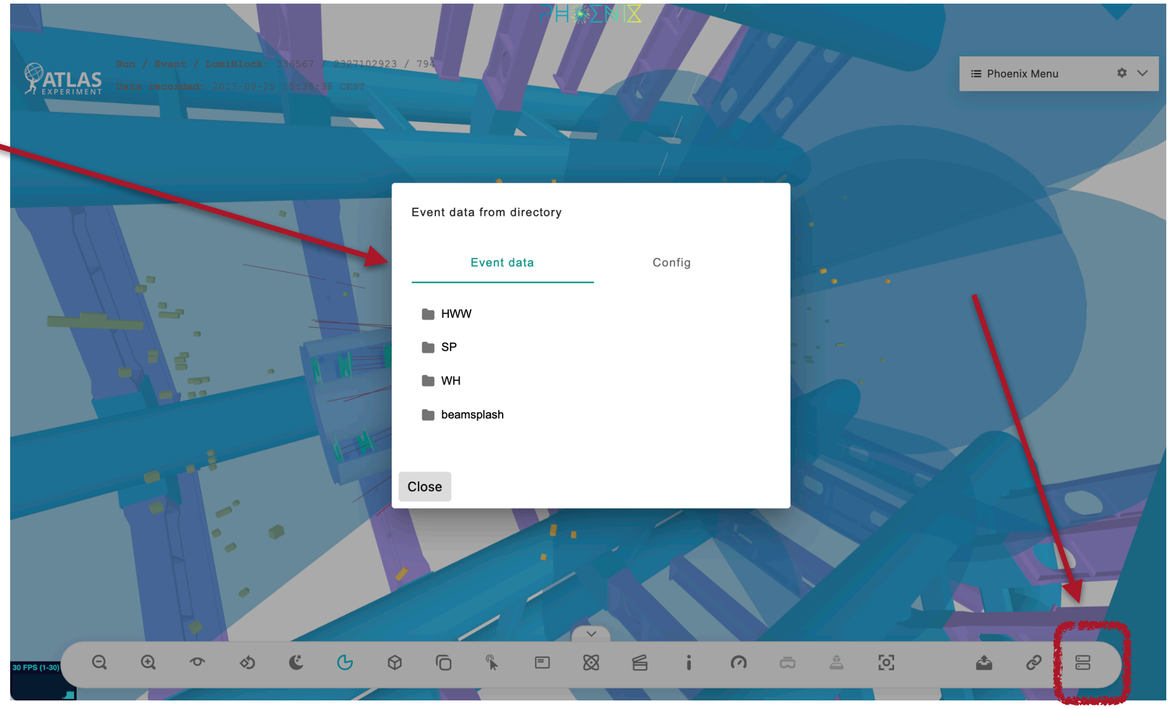


PHOENIXATLAS



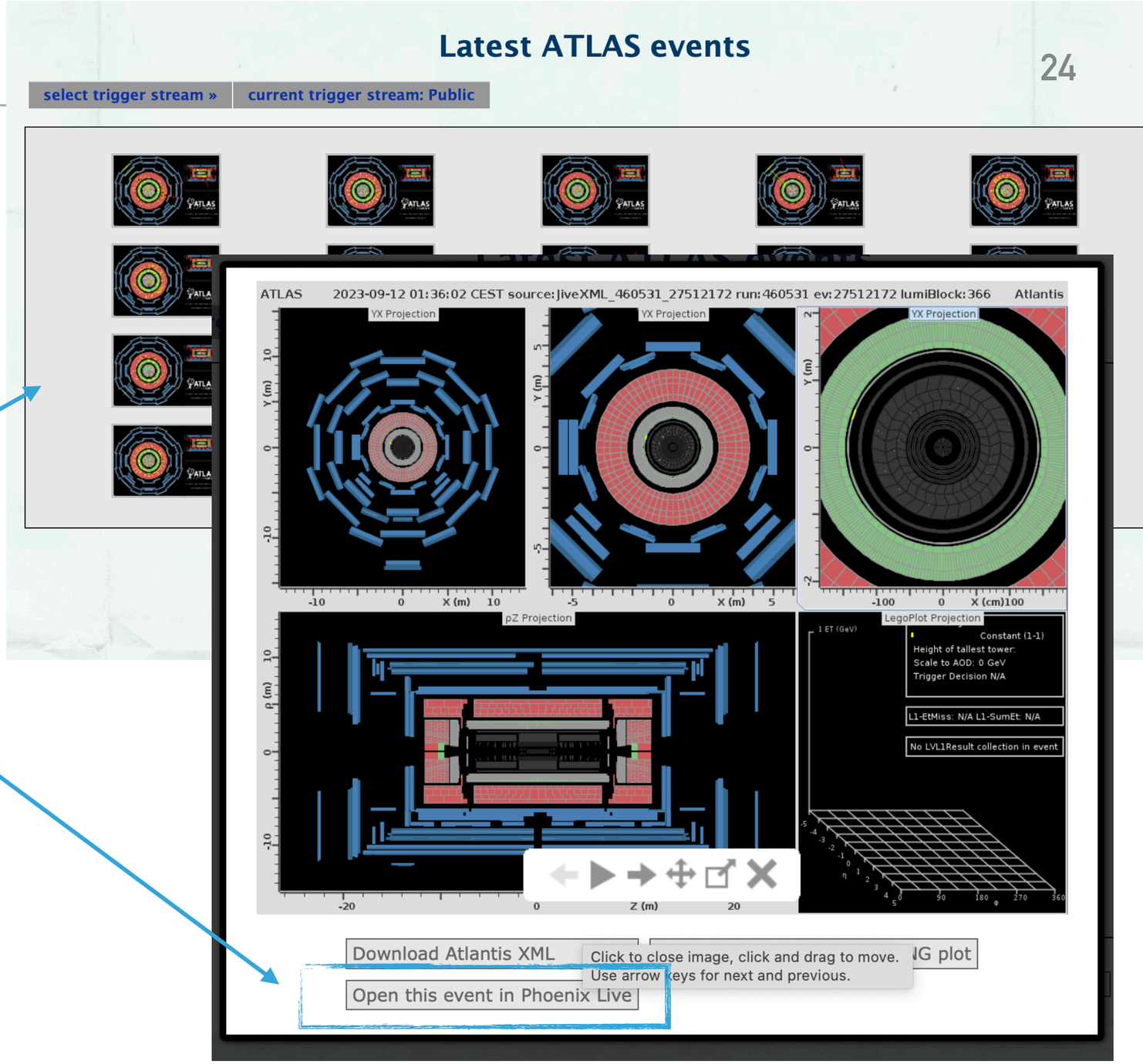
https://phoenixatlas.web.cern.ch/PhoenixATLAS/

- **Some key features:**
  - No splash screen
    - Directly start with example event, and animation
  - Pre-defined animations, to allow for inserting phoenix animations in predefined “circuit around the ring”
  - Have browsable directory of example events
  - Can pass geometry version in URL e.g.
    - https://phoenixatlas.web.cern.ch/PhoenixATLAS/?geom=run4Full
  - Also possible to force light or dark mode, by the same mechanism
- Git repo if you want to see how this is implemented in practice

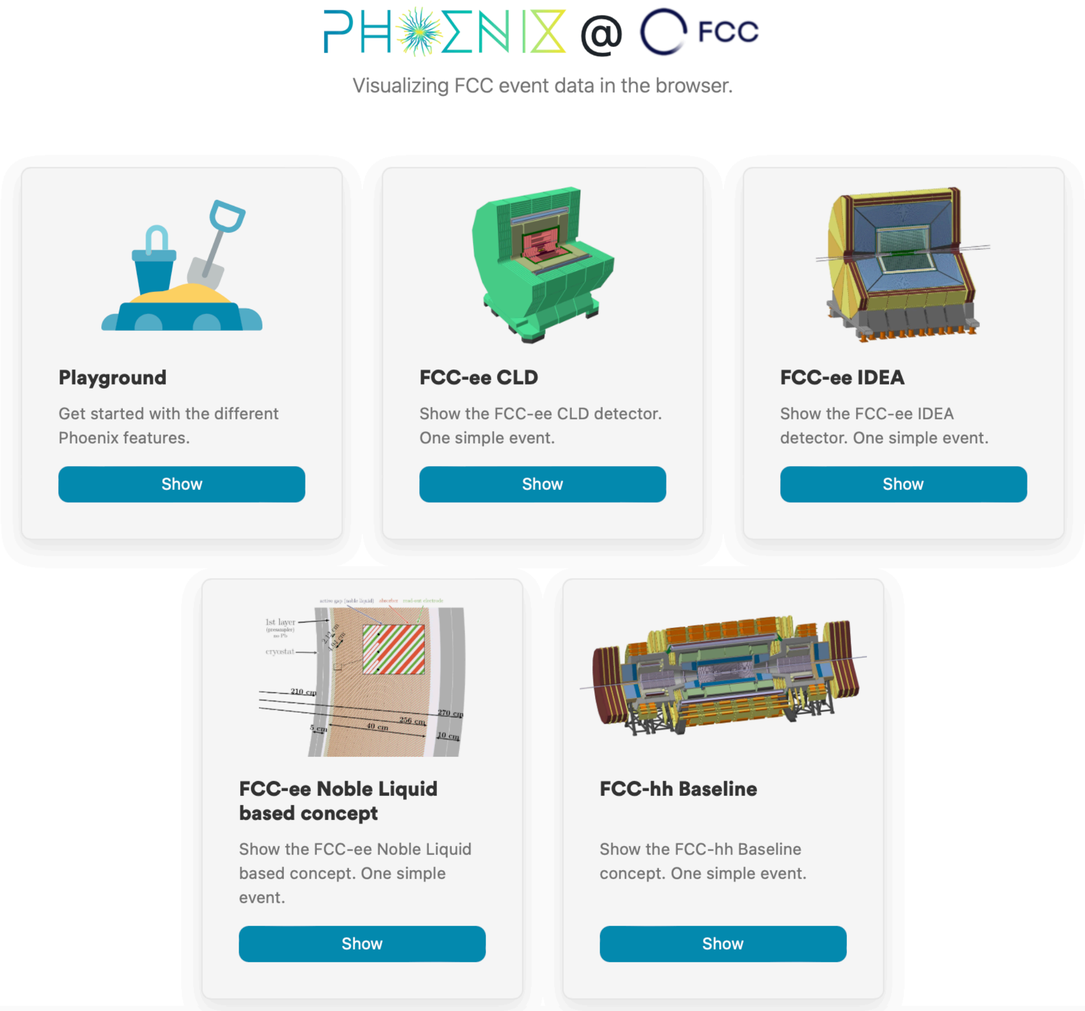


“LIVE” STREAMING EVENTS

- ATLAS copies a small fraction of live events to a server
  - Thumbnails are generated with Atlantis (the ATLAS 2D event display)
- From here, can open a link to PhoenixATLAS (the ATLAS Phoenix implementation)
- Several streams available, most limited to ATLAS members, but one is public:
  - https://atlas-live.cern.ch/latest



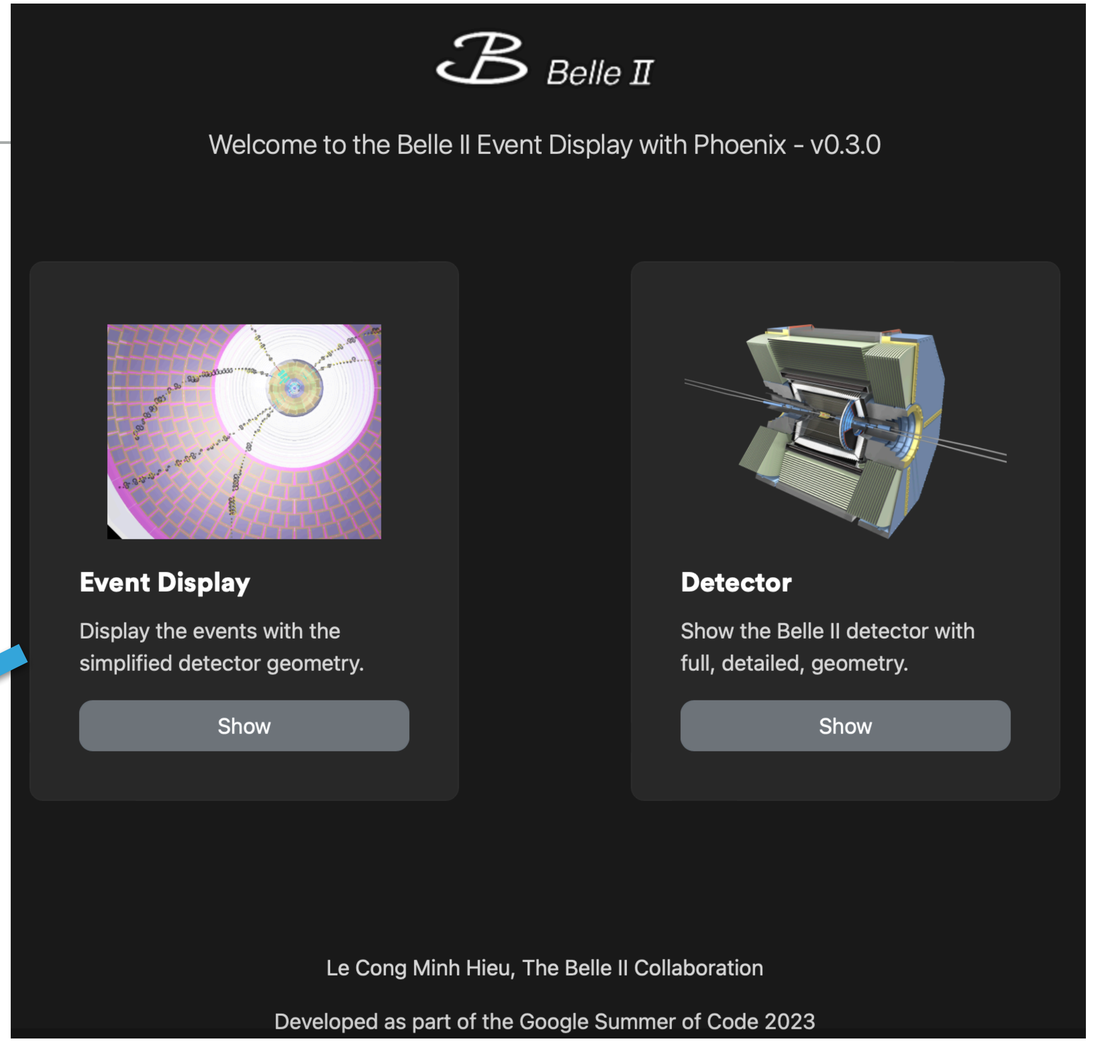
- FCC took a different approach, kept splash screen with multiple possible visualisations available:
  - FCC-ee CLD
  - FCC-ee IDEA
  - FCC-ee Noble Liquid concept
  - FCC-hh Baseline



https://fccsw.web.cern.ch/fccsw/phoenix/

BELLE-II

- Again opted for a splash screen with two choices: event display, or detailed detector
- Developed last year, by GSOC student

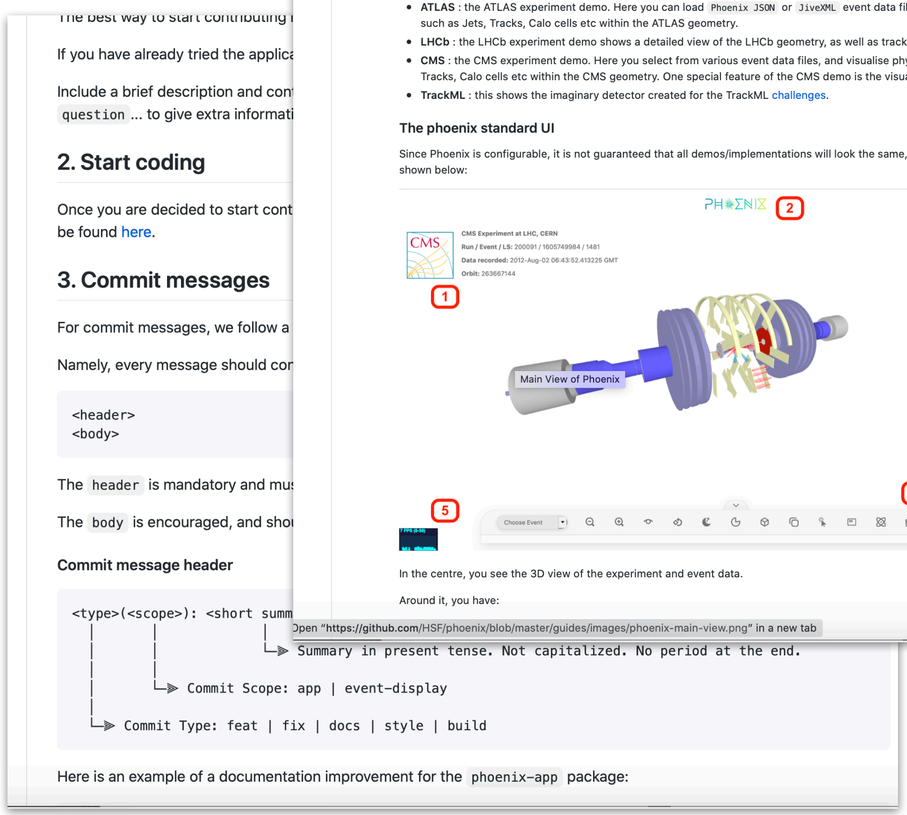


https://display.belle2.org/

- *Final thoughts*

DOCUMENTATION

- How do you learn more?
  - Phoenix has detailed developer and user guides, as well as API docs



The best way to start contributing

If you have already tried the applica

Include a brief description and cont
`question` ... to give extra informati

2. Start coding

Once you are decided to start cont
be found here.

3. Commit messages

For commit messages, we follow a

Namely, every message should cor

```
<header>
<body>
```

The `header` is mandatory and mus

The `body` is encouraged, and shou

Commit message header

```
<type>(<scope>): <short summ
  |       |             |
  |       |             └─⫸ Summary in present tense. Not capitalized. No period at the end.
  |       |
  |       └─⫸ Commit Scope: app | event-display
  |
  └─⫸ Commit Type: feat | fix | docs | style | build
```

Here is an example of a documentation improvement for the `phoenix-app` package:

CONTRIBUTING.md

The demo grid

When you first open the Phoenix demo (see the developer instructions for how to check it out and run locally) you will see a grid of Phoenix demos:

- **Playground** : a blank canvas where you can load 3D objects, move them around and generally experiment with Phoenix
- **Geometry display** : a simple demo of generating geometry procedurally/programmatically with Phoenix
- **ATLAS** : the ATLAS experiment demo. Here you can load `Phoenix JSON` or `JiveXML` event data files, and visualise physics objects such as Jets, Tracks, Calo cells etc within the ATLAS geometry.
- **LHCb** : the LHCb experiment demo shows a detailed view of the LHCb geometry, as well as tracks passing through it.
- **CMS** : the CMS experiment demo. Here you select from various event data files, and visualise physics objects such as Jets, Tracks, Calo cells etc within the CMS geometry. One special feature of the CMS demo is the visualisation of Muon Chambers.
- **TrackML** : this shows the imaginary detector created for the TrackML challenges.

The phoenix standard UI

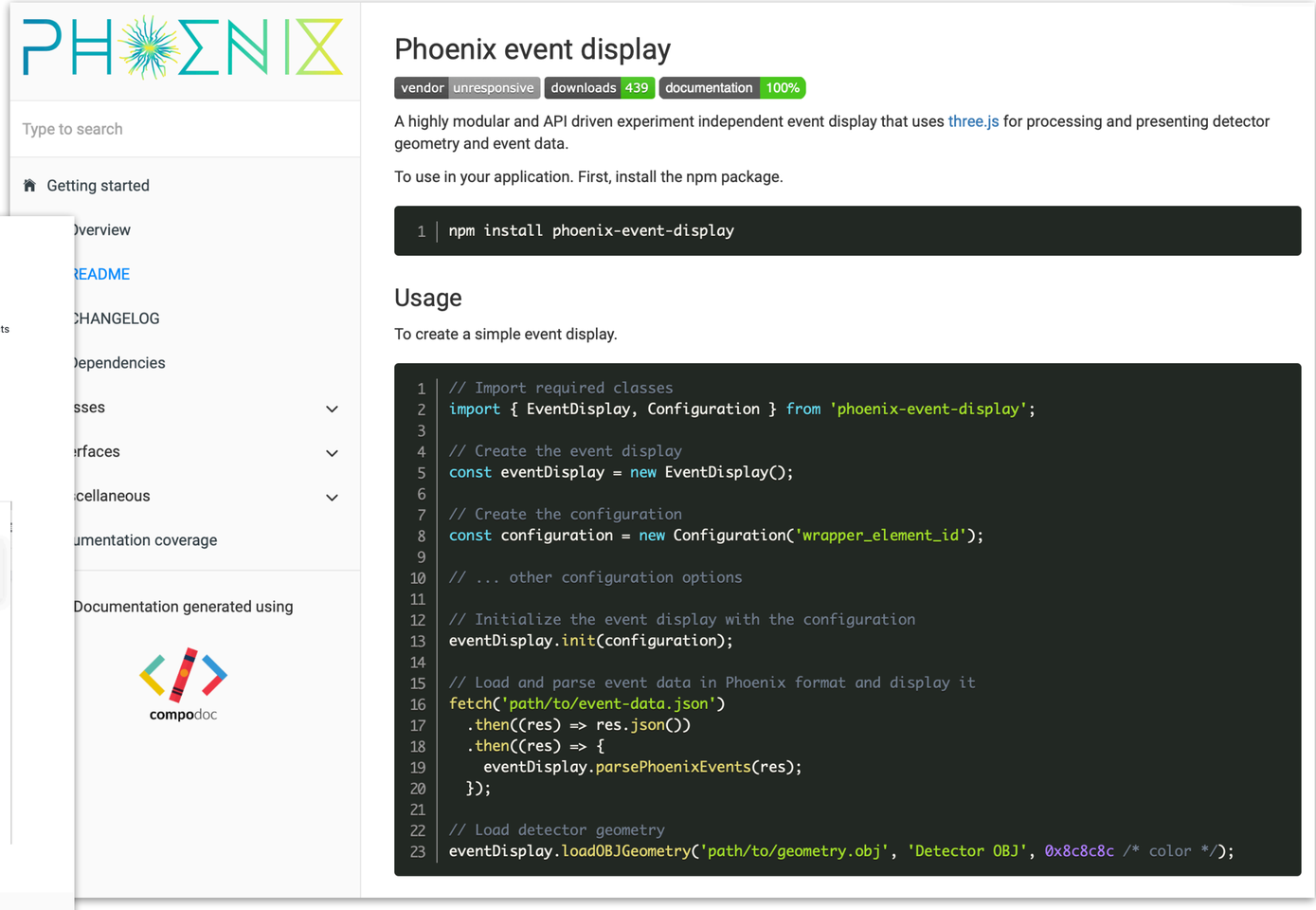
Since Phoenix is configurable, it is not guaranteed that all demos/implementations will look the same, but a typical Phoenix view is shown below:

In the centre, you see the 3D view of the experiment and event data.

Around it, you have:

Open "https://github.com/HSF/phoenix/blob/master/guides/images/phoenix-main-view.png" in a new tab

users.md



PHOENIX

Type to search

Getting started

Overview

README

CHANGELOG

Dependencies

sses

rfaces

cellaneous

umentation coverage

Documentation generated using

compodoc

Phoenix event display

vendor unresponsive | downloads 439 | documentation 100%

A highly modular and API driven experiment independent event display that uses three.js for processing and presenting detector geometry and event data.

To use in your application. First, install the npm package.

```
npm install phoenix-event-display
```

Usage

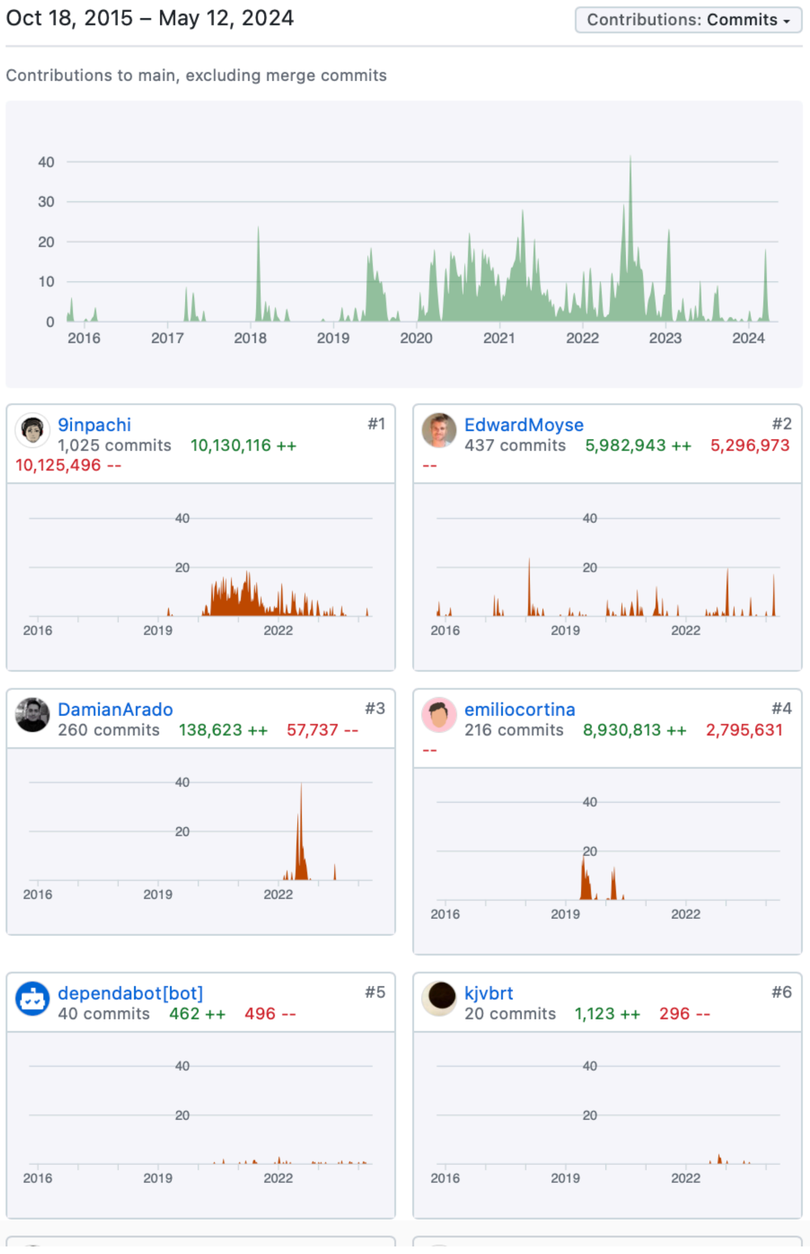
To create a simple event display.

```
// Import required classes
import { EventDisplay, Configuration } from 'phoenix-event-display';

// Create the event display
const eventDisplay = new EventDisplay();

// Create the configuration
const configuration = new Configuration('wrapper_element_id');

// ... other configuration options

// Initialize the event display with the configuration
eventDisplay.init(configuration);

// Load and parse event data in Phoenix format and display it
fetch('path/to/event-data.json')
  .then((res) => res.json())
  .then((res) => {
    eventDisplay.parsePhoenixEvents(res);
  });

// Load detector geometry
eventDisplay.loadOBJGeometry('path/to/geometry.obj', 'Detector OBJ', 0x8c8c8c /* color */);
```

https://hepsoftwarefoundation.org/phoenix/api-docs/

PERSONPOWER

- **Phoenix has no funding. We rely entirely on volunteer effort**
  - Help is always welcomed!
- **Several Google Summer of Code students (thanks to HSF!)**
  - 2019: Emilio Cortina Labra
  - 2020: Fawad Ali
  - 2022: Mohammad Humayun Khan + Hieu Le Cong Minh
  - 2023: Somya Bansal
- Most contributions come from physicists working on experiments using Phoenix (e.g. EIC are making a commitment to help)
  - However Phoenix is a fully open-sourced project, and so sometimes we get help from the general public



CONCLUSION

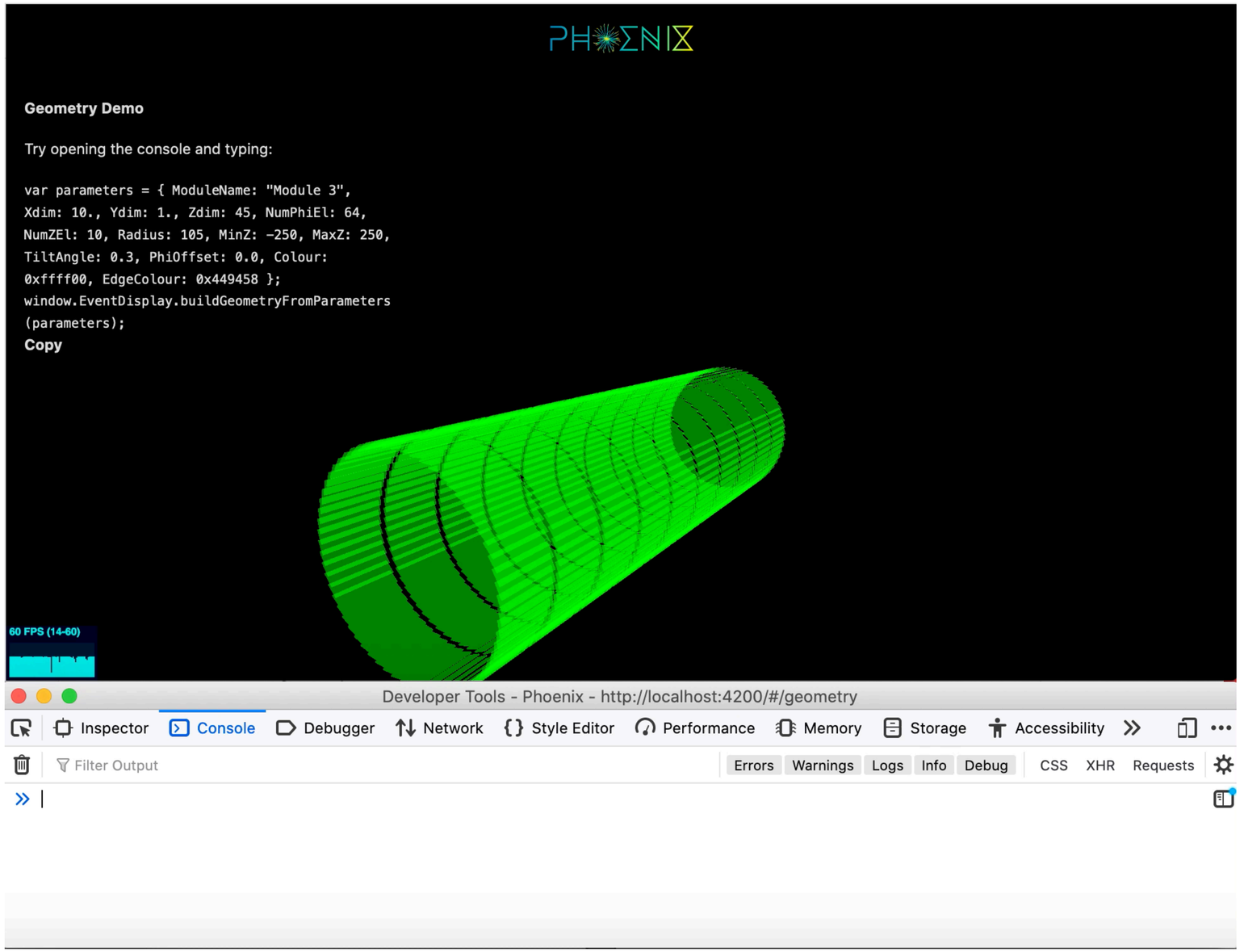
- **Very brief overview of Phoenix**
  - Many details were not covered (my apologies)
  - Phoenix is in active use by several experiments, and already has many powerful features
  - To add more, person power would be much appreciated - we rely on volunteers!
  - An example of the benefits of cross experiment collaboration
- If you are interested in using Phoenix, *or contributing*, please contact us:
  - Via **github** issues: [link] or discussions: [link]
  - Or on our **mailing list**: phoenix-event-display@cern.ch

BACKUP

WALKTHROUGH: PLAYGROUND AND GEOMETRY

- In **Geometry** [link], you can open the javascript console in your browser and programmatically add a very simple detector
  - e.g.

```
var parameters = { ModuleName: "Module 3", Xdim: 10., Ydim:
1., Zdim: 45, NumPhiEl: 64, NumZEl: 10, Radius: 75, MinZ:
-250, MaxZ: 250, TiltAngle: 0.3, PhiOffset: 0.0, Colour:
0x00ff00, EdgeColour: 0x449458 };

window.eventDisplay.buildGeometryFromParameters(parameters);
```

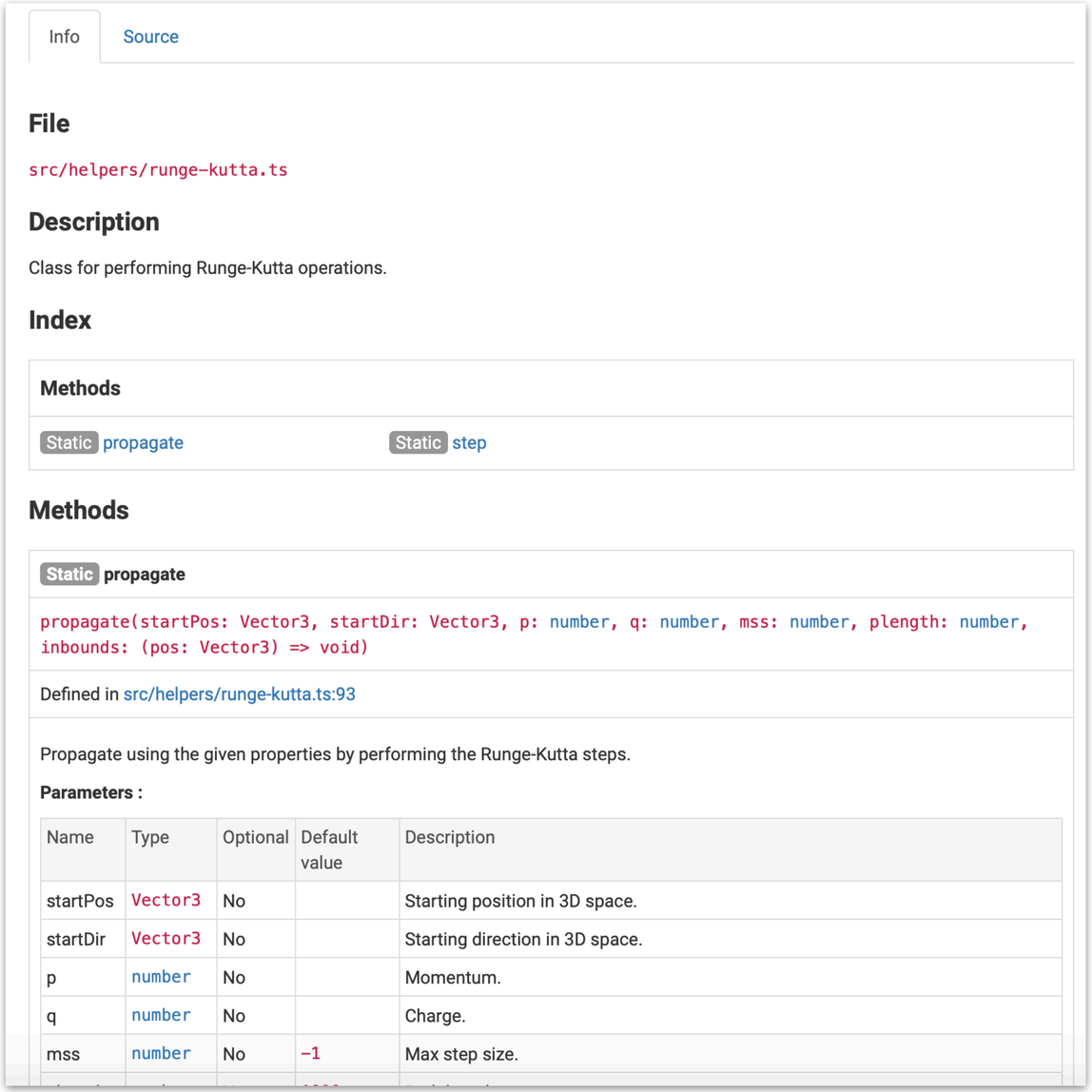


DESIGN CONCEPTS

- In order to support as many experiments as possible, some key goals:
  - **Permissive licence and open source** (Apache 2.0 Licence)
  - **Use industry standards**
  - **Simple standard format for Event Data**
  - **Good documentation**
  - **Don’t make experiment specific assumptions**
  - Make Phoenix **configurable, extendable and modular**

MENUS AND HELPERS

- Phoenix provides lots of functionality to help developers
  - e.g Phoenix has its own menu system phoenix-ui-components
- Phoenix also has many classes to help render physics data e.g.
  - Many experiments only store limited numbers of track parameters, so cannot draw a complete curve
  - Phoenix provides a **RungeKutta** propagator
  - You just need to supply the magnetic field!



Info Source

File

src/helpers/runge-kutta.ts

Description

Class for performing Runge-Kutta operations.

Index

Methods

Static propagate Static step

Methods

Static propagate

```
propagate(startPos: Vector3, startDir: Vector3, p: number, q: number, mss: number, plength: number,
inbounds: (pos: Vector3) => void)
```

Defined in src/helpers/runge-kutta.ts:93

Propagate using the given properties by performing the Runge-Kutta steps.

Parameters :

Name	Type	Optional	Default value	Description
startPos	Vector3	No		Starting position in 3D space.
startDir	Vector3	No		Starting direction in 3D space.
p	number	No		Momentum.
q	number	No		Charge.
mss	number	No	−1	Max step size.

https://hepsoftwarefoundation.org/phoenix/api-docs/classes/RungeKutta.html

EXTENSIBILITY: ADDING A NEW DETECTOR

- **How would you add a new detector?**
- You basically need to add **two** files
  - `experiment.component.html` file (defines the 'view')
  - `experiment.component.ts` the experiment specific **implementation** i.e. file contains e.g.
    - The default configuration and event,
    - Loaders required (if you need to convert from another event data format to Phoenix format)
    - Geometry etc
- And that is it!
  - *Less than a day of work to add a new detector*
- See the documentation for more information
  - e.g. How to write your own event data loader

```
import { Component, OnInit } from '@angular/core';
import { EventDisplayService } from 'phoenix-ui-components';
import { Configuration, PresetView, PhoenixMenuNode, PhoenixLoader } from 'phoenix-event-
import { environment } from '../../../environments/environment';
import eventConfig from '../../../../event-config.json';

@Component({
  selector: 'app-atlas',
  templateUrl: './atlas.component.html',
  styleUrls: ['./atlas.component.scss']
})
export class AtlasComponent implements OnInit {
  phoenixMenuRoot = new PhoenixMenuNode('Phoenix Menu', 'phoenix-menu');

  constructor(private eventDisplay: EventDisplayService) { }

  ngOnInit() {
    let defaultEvent: { eventFile: string, eventType: string };
    // Get default event from configuration
    if (environment?.singleEvent) {
      defaultEvent = eventConfig;
    } else {
      defaultEvent = {
        eventFile: 'assets/files/JiveXML/JiveXML_336567_2327102923.xml',
        eventType: 'jivexml'
      }
    }

    // Define the configuration
    const configuration: Configuration = {
      eventDataLoader: new PhoenixLoader(),
      presetViews: [
        new PresetView('Left View', [0, 0, -12000], 'left-cube'),
        new PresetView('Center View', [-500, 12000, 0], 'top-cube'),
        new PresetView('Right View', [0, 0, 12000], 'right-cube')
      ],
      defaultView: [4000, 4000, 4000],
      // Set the phoenix menu to be used (defined above)
      phoenixMenuRoot: this.phoenixMenuRoot,
      // Default event data to fallback to if none given in URL
      // Do not set if there should be no event loaded by default
      defaultEventFile: defaultEvent
```

EXTENSIBILITY

- The `experiment.component.html` file, specifies what is used in the view …

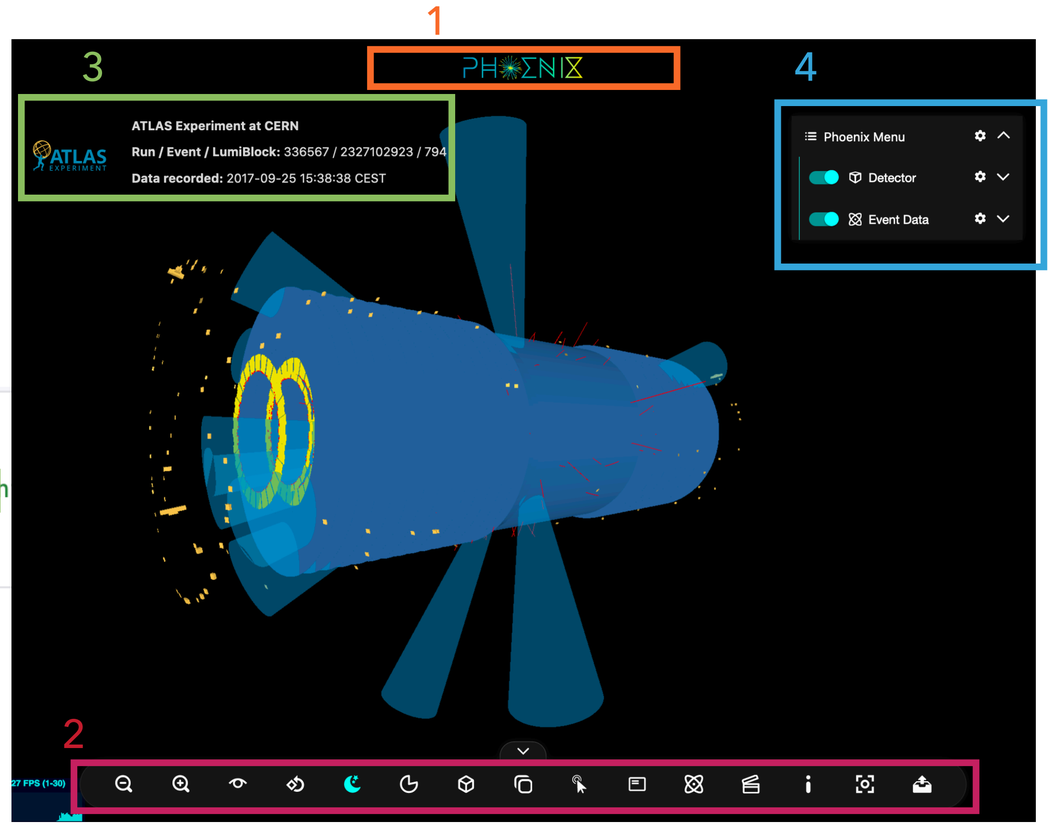
1. Link back to main Phoenix page

```
1  <app-nav></app-nav>
2  <app-ui-menu></app-ui-menu>
3  <app-experiment-info experiment="atlas" experimentTagline="ATLAS Experiment at CERN"></app-experiment-in
4  <app-phoenix-menu [rootNode]="phoenixMenuRoot"></app-phoenix-menu>
5  <div id="eventDisplay"></div>
```

2. Phoenix row menu

3. Experiment logo, link and info

4. Phoenix geometry/event data menu



atlas.component.html

EXTENSIBILITY: ADDING A NEW PHYSICS OBJECT

- **An example:** LHCb authors wanted to add CaloCells which do not point to the origin i.e. PlanarCaloCells
  - Have a look at PR 299 for details (and the documentation)
  - But, main steps were :
    - Add a `getPlanarCaloCell` function to `phoenix-objects.ts` (which draws the cells)
    - Call this from `phoenix-loader.ts`
      - And also add relevant cuts/filters, GUI options

```
+    if (eventData.PlanarCaloCells) {
+      //(Optional) Cuts can be added to any physics object.
+      const cuts = [
+        new Cut('energy', 0, 10000)
+      ];
+
+      const addPlanarCaloCellsOptions = (
+        typeFolder: GUI,
+        typeFolderPM: PhoenixMenuNode
+      ) => {
+        const scalePlanarCaloCells = (value: number) => {
+          this.graphicsLibrary
+            .getSceneManager()
+            .scaleChildObjects('PlanarCaloCells', value / 100, 'z')
+        };
+
+        if (typeFolder) {
+          const sizeMenu = typeFolder
+            .add({ PlanarCaloCellsScale: 100 }, 'PlanarCaloCellsScale', 1, 400)
+            .name('PlanarCaloCells Size (%)');
+          sizeMenu.onChange(scalePlanarCaloCells);
+        }
+
+        if (typeFolderPM) {
+          typeFolderPM.addConfig('slider', {
+            label: 'PlanarCaloCells Size (%)',
+            value: 100,
+            min: 1,
+            max: 400,
+            allowCustomValue: true,
+            onChange: scalePlanarCaloCells,
+          });
+        }
+      };
+
+      const { typeFolder, typeFolderPM } = this.ui.addEventDataTypeFolder(
+        'PlanarCaloCells'
+      );
+      const objectGroup = this.graphicsLibrary.addEventDataTypeGroup(
+        'PlanarCaloCells'
+      );
```

CHECK IF PLANAR CELLS IN INPUT

ADD AN ENERGY CUT

ADD A SLIDER TO CONTROL SCALE

ADDING A NEW PHYSICS OBJECT: RESULT

ADDING A NEW PHYSICS OBJECT: RESULT

Feedback from LHCb authors:

- *"if you follow the documentation … with few modifications it is quite easy and it displays successfully and beautifully!"*
- *"Surprisingly easy to add new objects"*

VR/AR

- Rudimentary support for VR/AR
  - AR works on Android, VR works in Quest 2 etc, see Twitter post for example video
  - No menu support in AR/XR so much functionality not available
    - Ticket 558
- Depends on browser (notably, Safari on iOS does not work any more)
  - VisionPro will support WebXR, so maybe it will FINALLY come to iOS (but I would not bet on it)
- In short, this works, but not on all devices and is currently quite limited