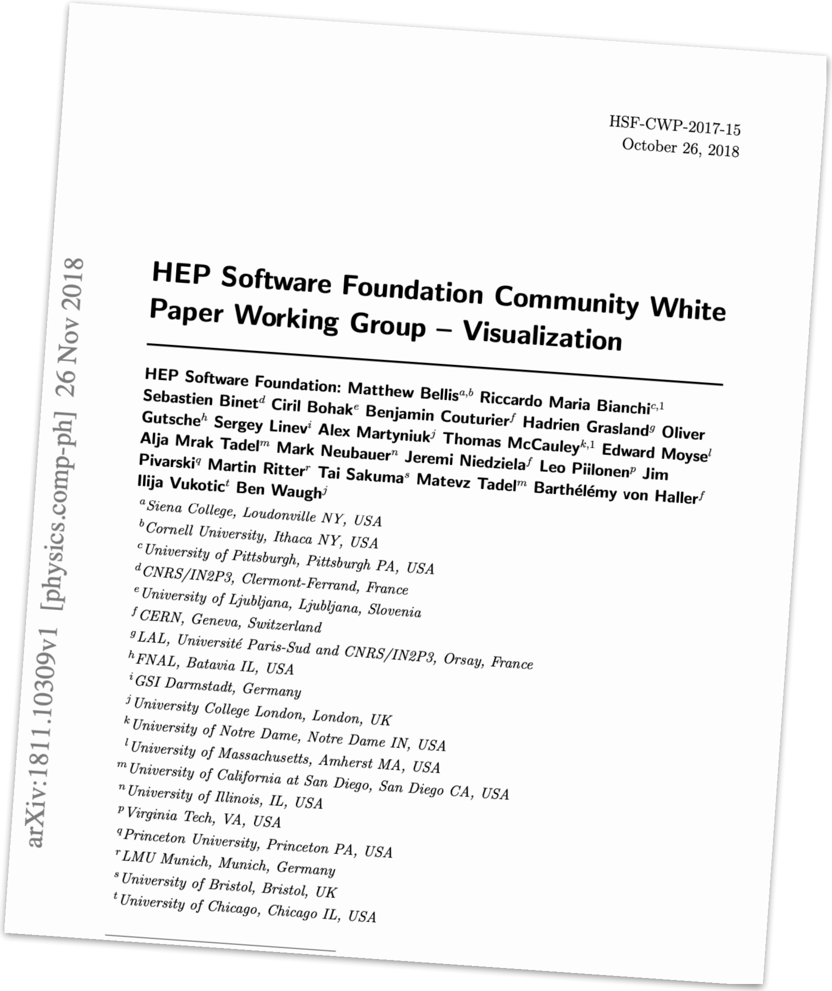
EDWARD MOYSE

THE PHOENIX EVENT DISPLAY FRAMEWORK

INTRODUCTION

- In 2017 the HSF visualisation white paper identified the desirability of having a common event format, and a common tool to visualise event data (and geometry)
    - Up until now, event displays have tended to be per-experiment
- Phoenix is an experiment agnostic display, supported by the HSF visualisation group:
    - Repository: https://github.com/HSF/phoenix
    - Demo: http://hepsoftwarefoundation.org/phoenix/
    - Runs entirely in the browser, so scalable and cheap to host
        - Uses industry standard, 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 officially used by ATLAS, FCC, LHCb, Belle-II (see documentation)



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

HSF-CWP-2017-15
October 26, 2018

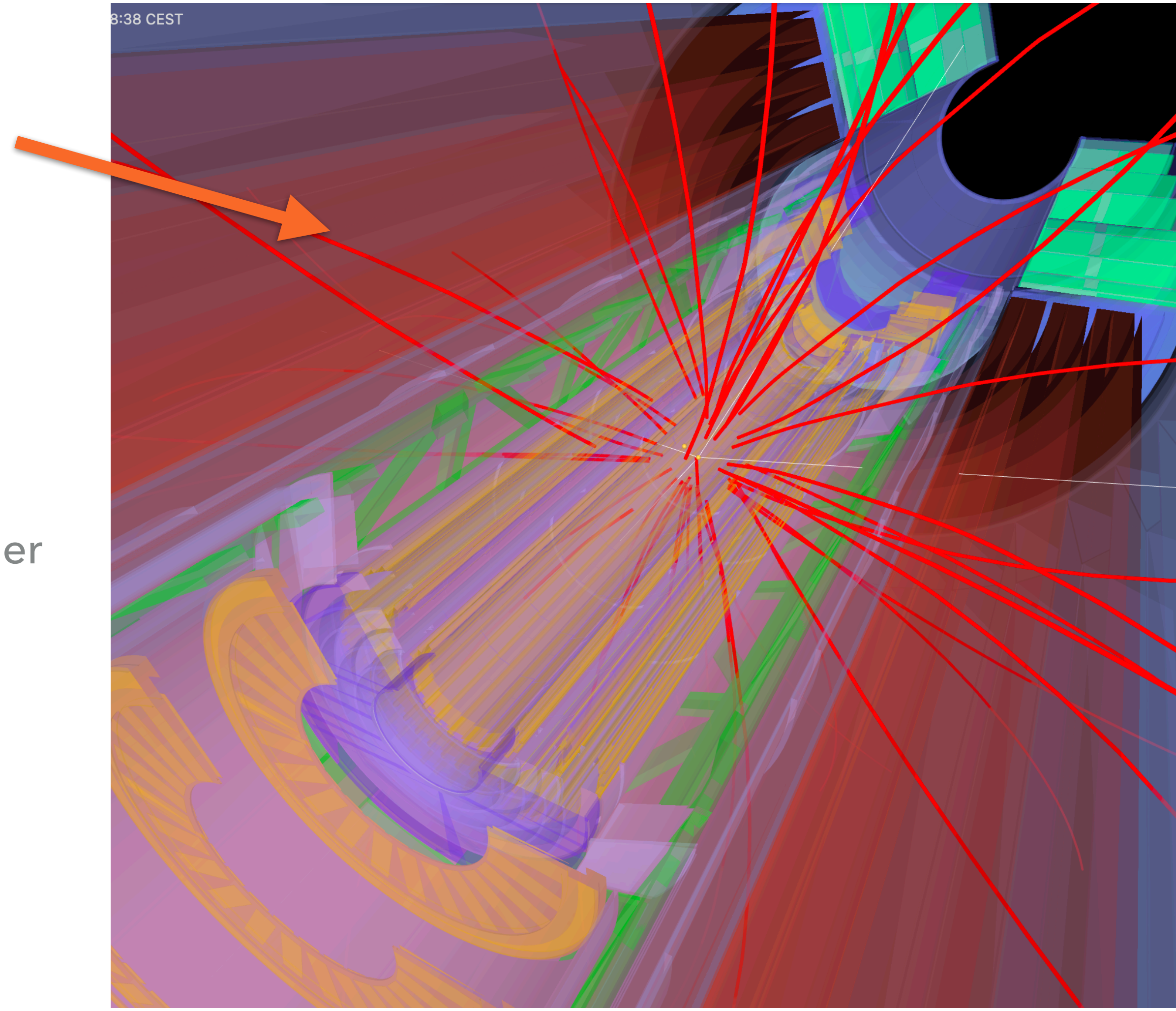
HEP Software Foundation Community White Paper Working Group – Visualization

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*

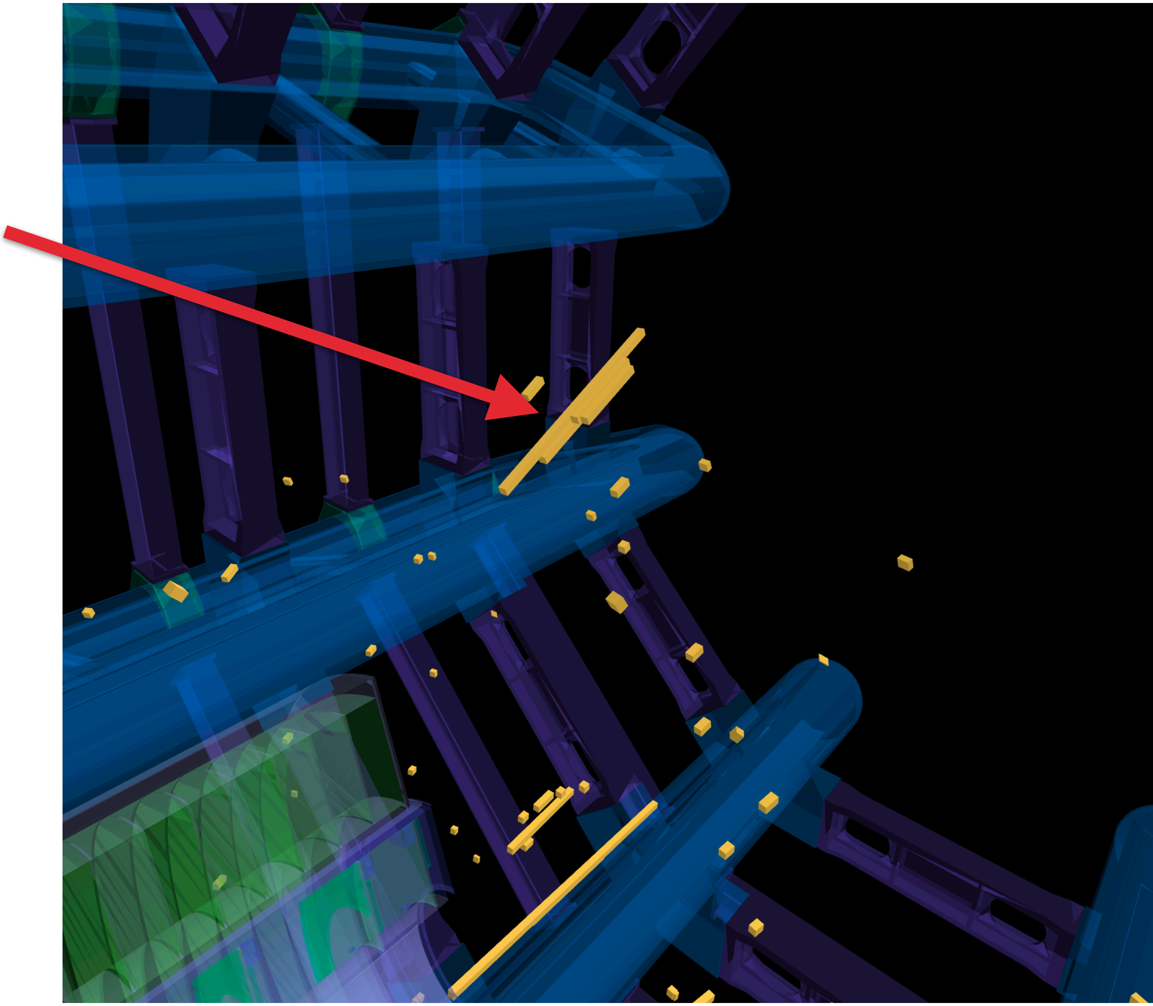
SUPPORTED PHYSICS OBJECTS

- **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**



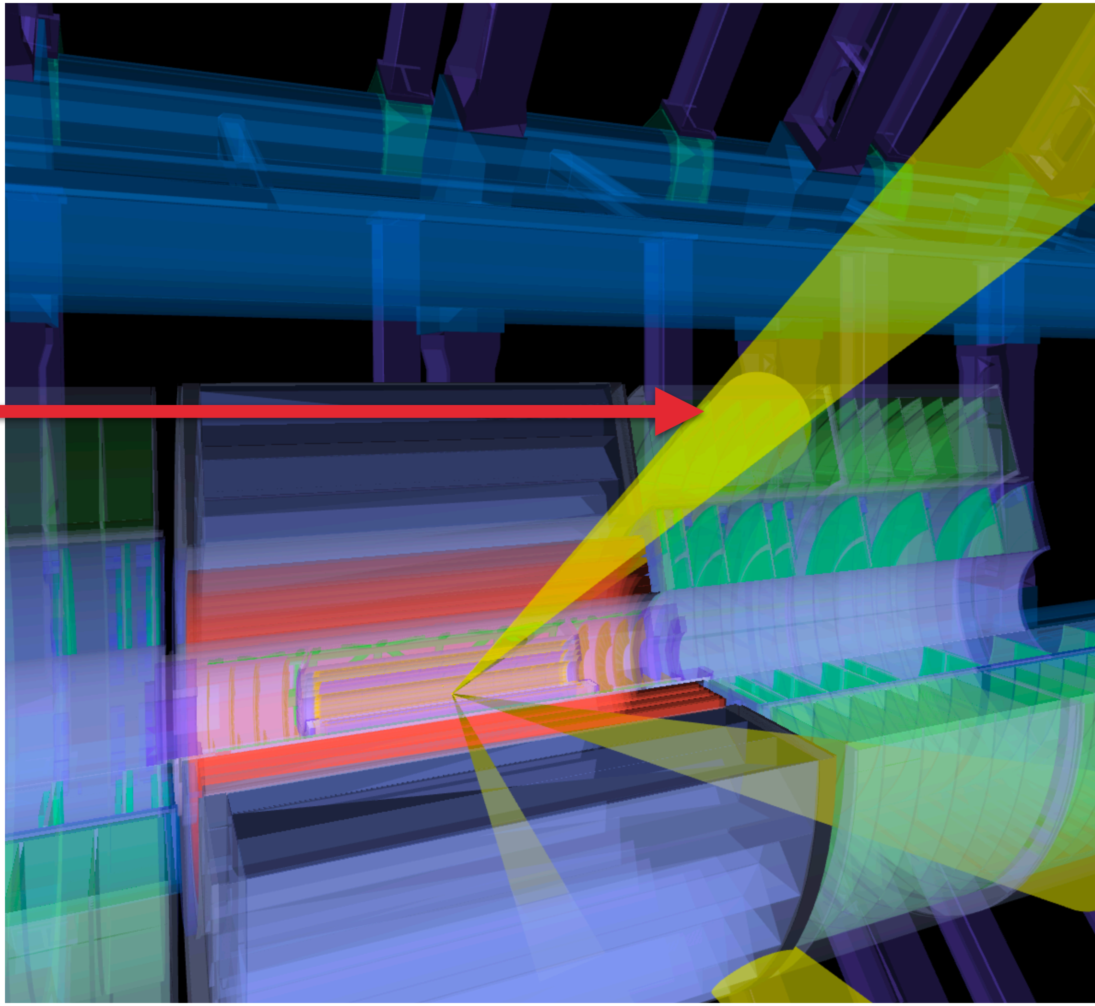
SUPPORTED PHYSICS OBJECTS

- **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**



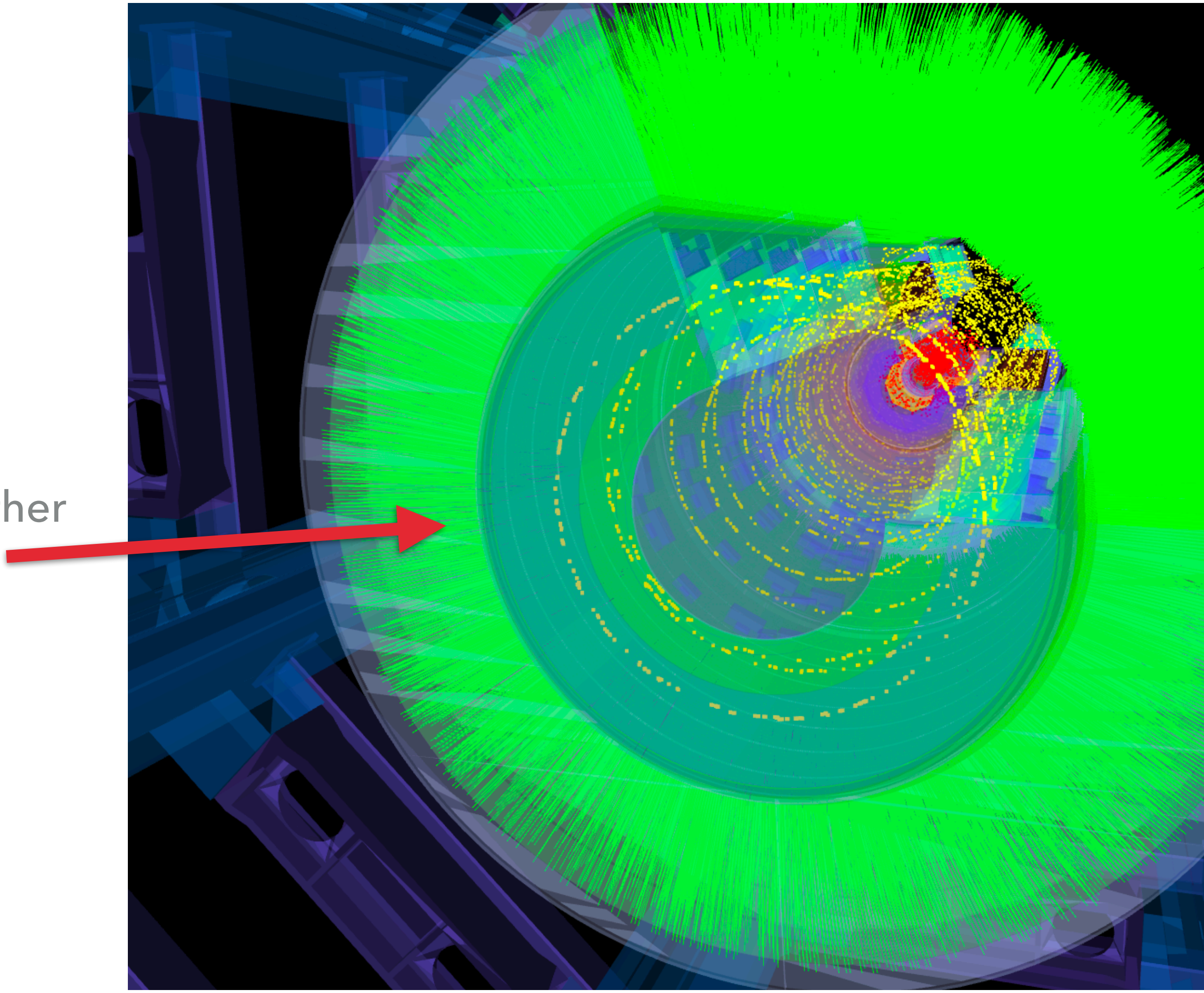
SUPPORTED PHYSICS OBJECTS

- **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**



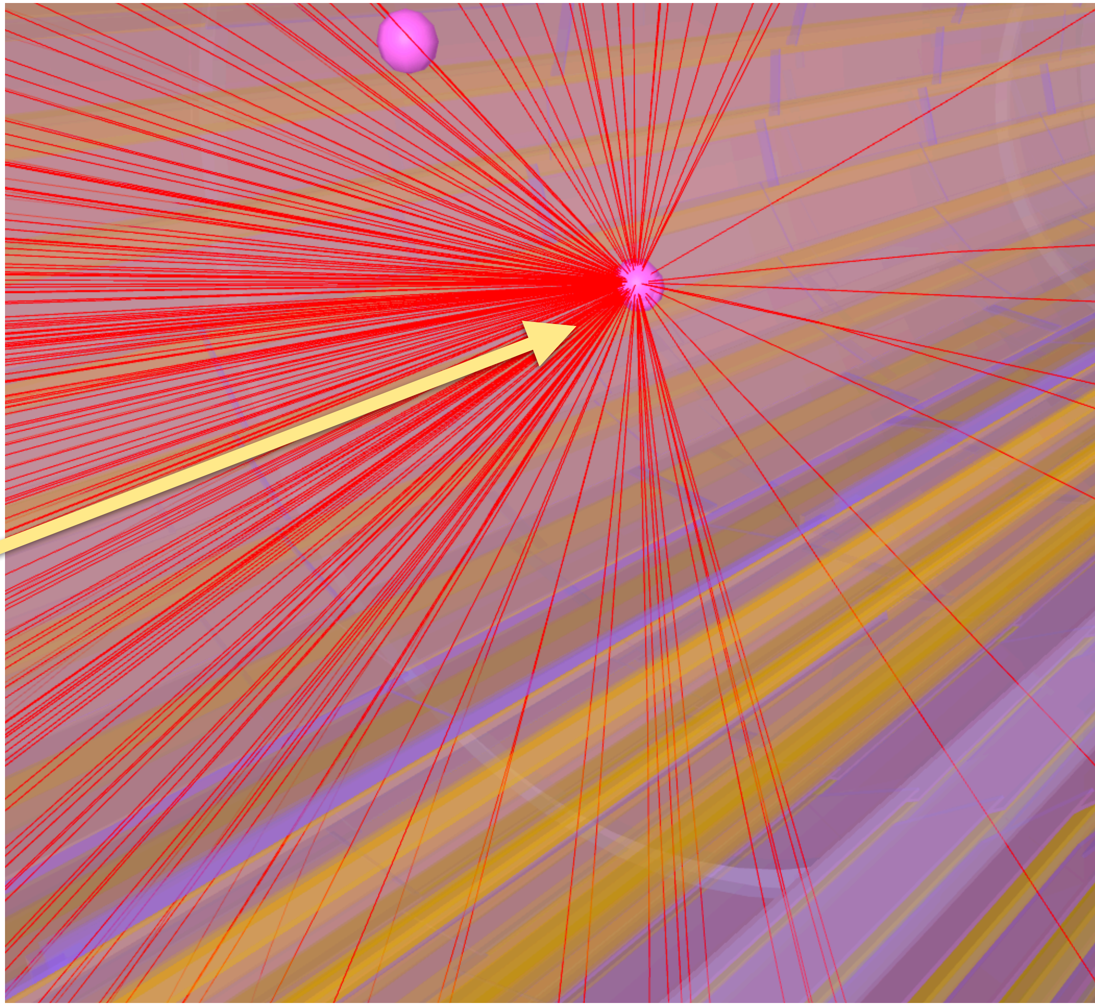
SUPPORTED PHYSICS OBJECTS

- **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**



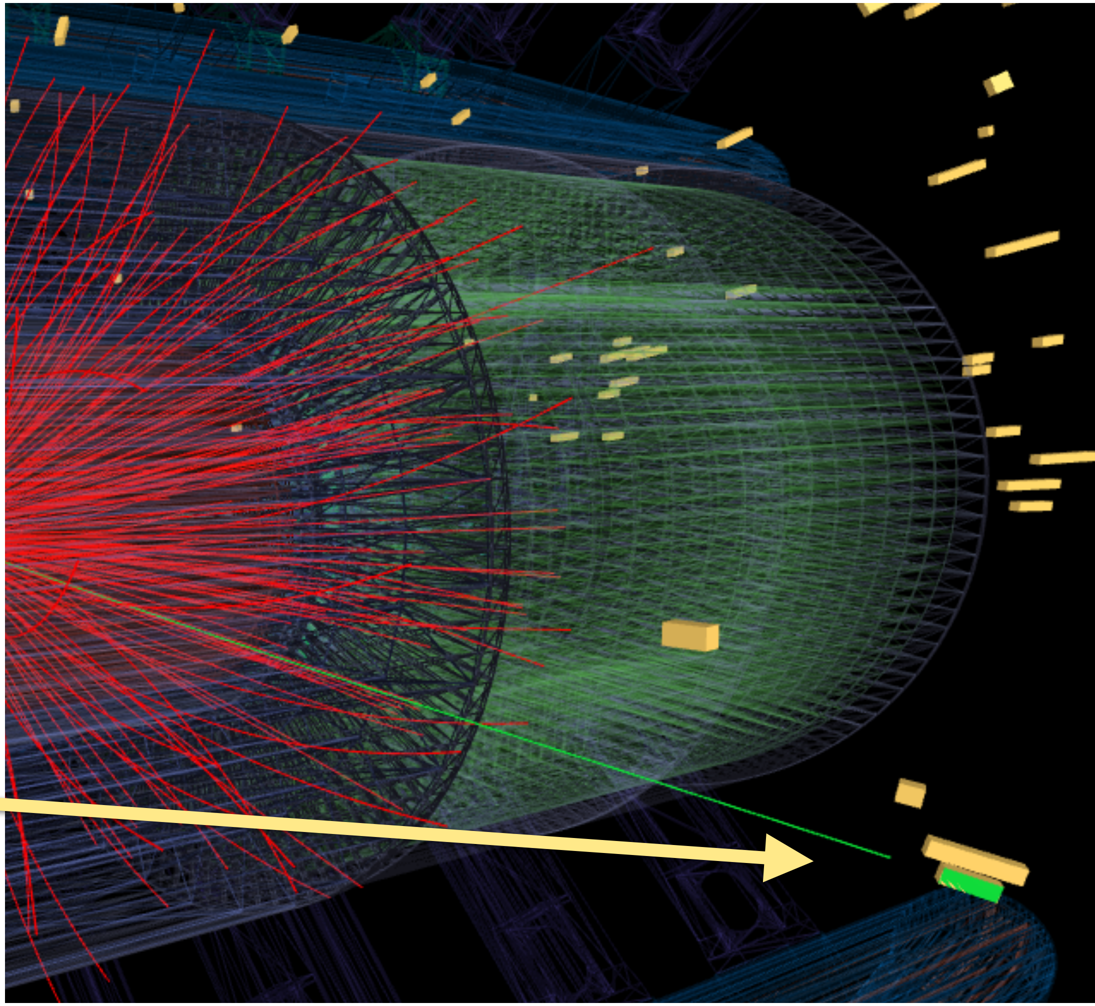
SUPPORTED PHYSICS OBJECTS

- **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**



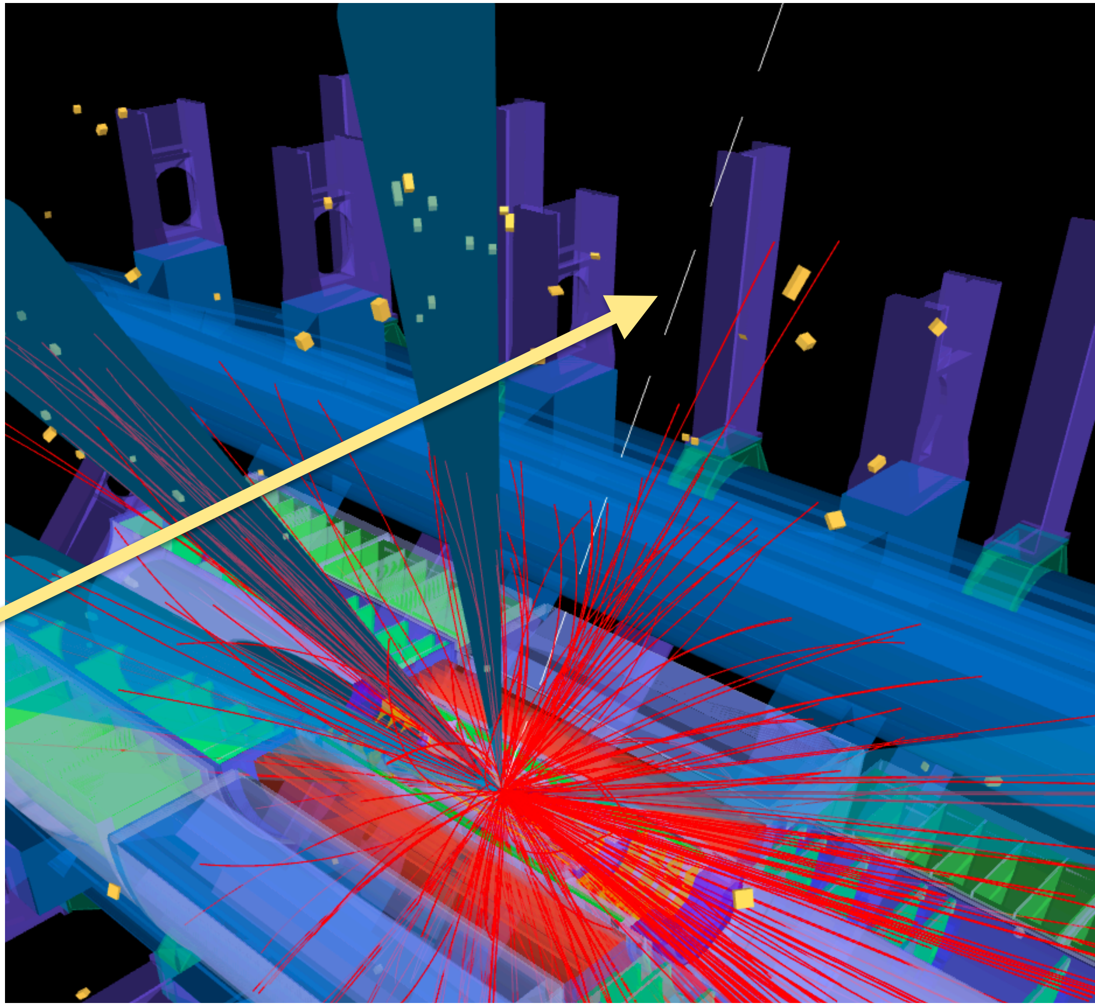
SUPPORTED PHYSICS OBJECTS

- **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**



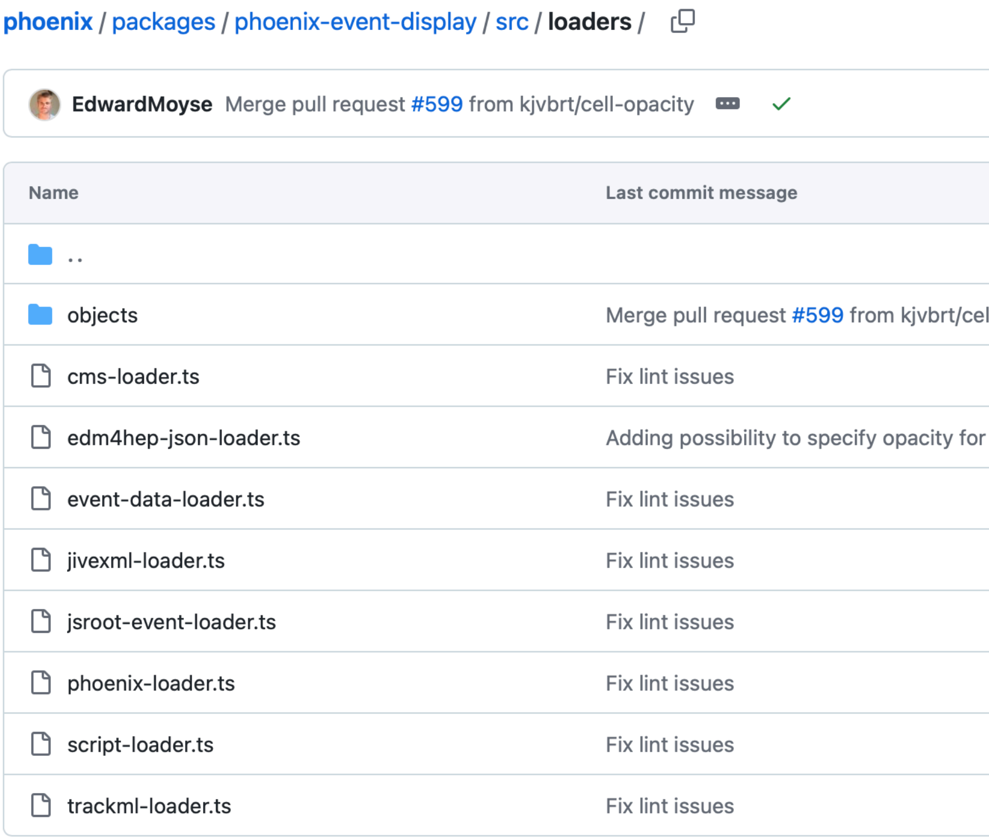
SUPPORTED PHYSICS OBJECTS

- **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 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 our internal format…
  - (More on this later)



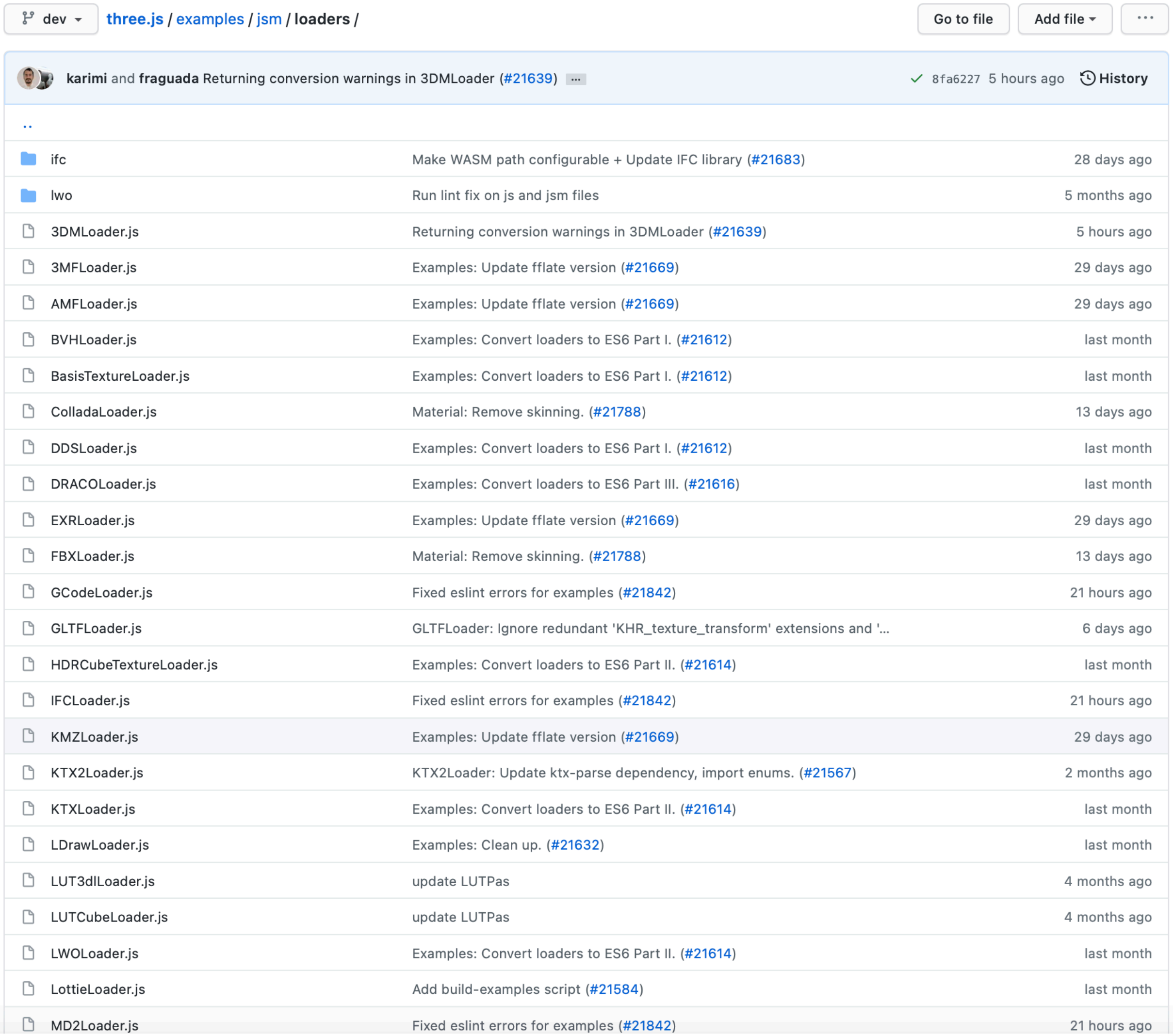
phoenix / packages / phoenix-event-display / src / loaders /

EdwardMoyse Merge pull request #599 from kjvbrt/cell-opacity ✓

Name	Last commit message
..	
objects	Merge pull request #599 from kjvbrt/cel
cms-loader.ts	Fix lint issues
edm4hep-json-loader.ts	Adding possibility to specify opacity for
event-data-loader.ts	Fix lint issues
jivexml-loader.ts	Fix lint issues
jsroot-event-loader.ts	Fix lint issues
phoenix-loader.ts	Fix lint issues
script-loader.ts	Fix lint issues
trackml-loader.ts	Fix lint issues

SUPPORTED GEOMETRY

- Phoenix can display geometry stored in many standard formats:
    - Natively supported formats are OBJ, **glTF** , ROOT, json(gz)
        - We recommend compressed glTF (glb) as it is the most compact, recommended by threejs, and Phoenix can automatically populate the detector menu with the embedded hierarchy (see our docs for more)
        - 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
- ACTS can output OBJ format geometry



three.js / examples / jsm / loaders /

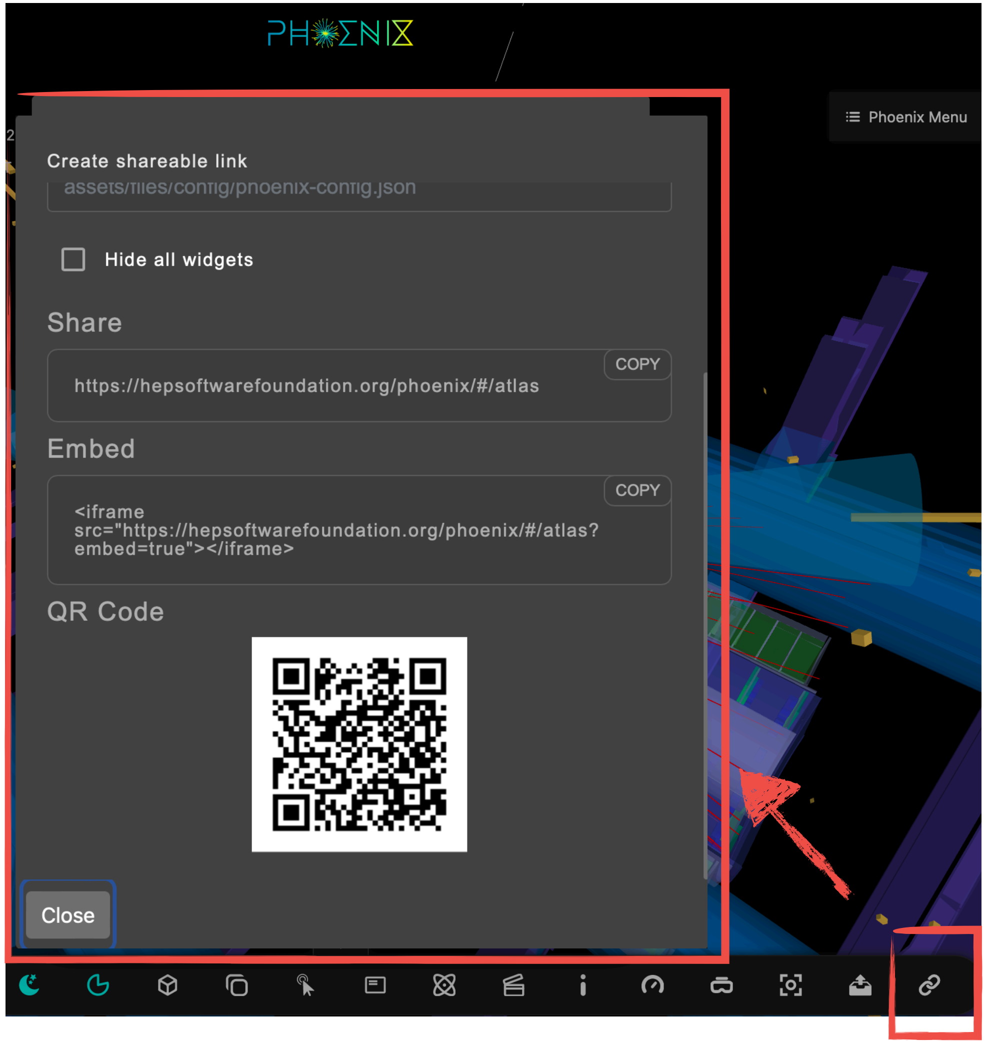
karimi and fraguada Returning conversion warnings in 3DMLoader (#21639) ✓ 8fa6227 5 hours ago

..		
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
LottieLoader.js	Add build-examples script (#21584)	last month
MD2Loader.js	Fixed eslint errors for examples (#21842)	21 hours ago

https://github.com/mrdoob/three.js/tree/dev/examples/jsm/loaders

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
- 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]()



Create shareable link

assets/files/config/phoenix-config.json

□ Hide all widgets

Share

https://hepsoftwarefoundation.org/phoenix/#/atlas

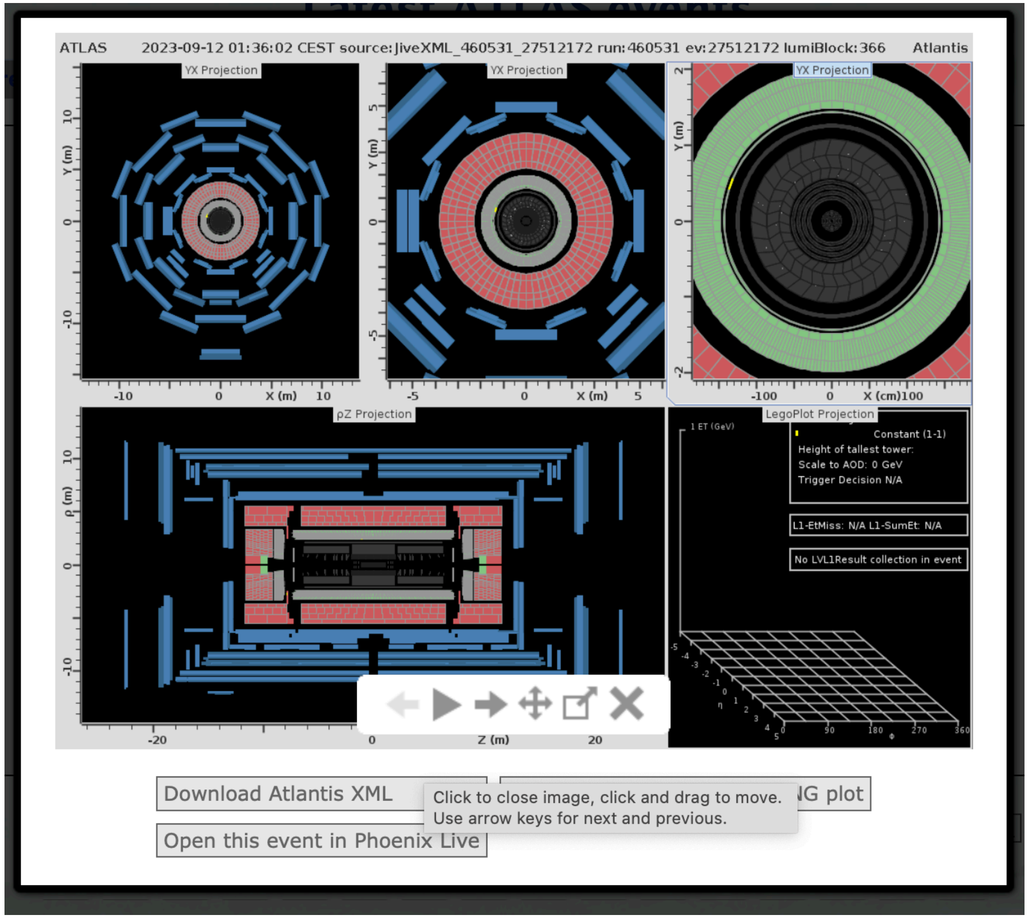
Embed

<iframe src="https://hepsoftwarefoundation.org/phoenix/#/atlas?embed=true"></iframe>

QR Code

“LIVE” STREAMING EVENTS

- ATLAS copies a small fraction of live events to a server
- From here, can open a view generated by Atlantis, or a link to PhoenixATLAS (the ATLAS-specific)
- The



ATLAS 2023-09-12 01:36:02 CEST source:JiveXML_460531_27512172 run:460531 ev:27512172 lumiBlock:366 Atlantis

Constant (1-1)
Height of tallest tower:
Scale to AOD: 0 GeV
Trigger Decision N/A

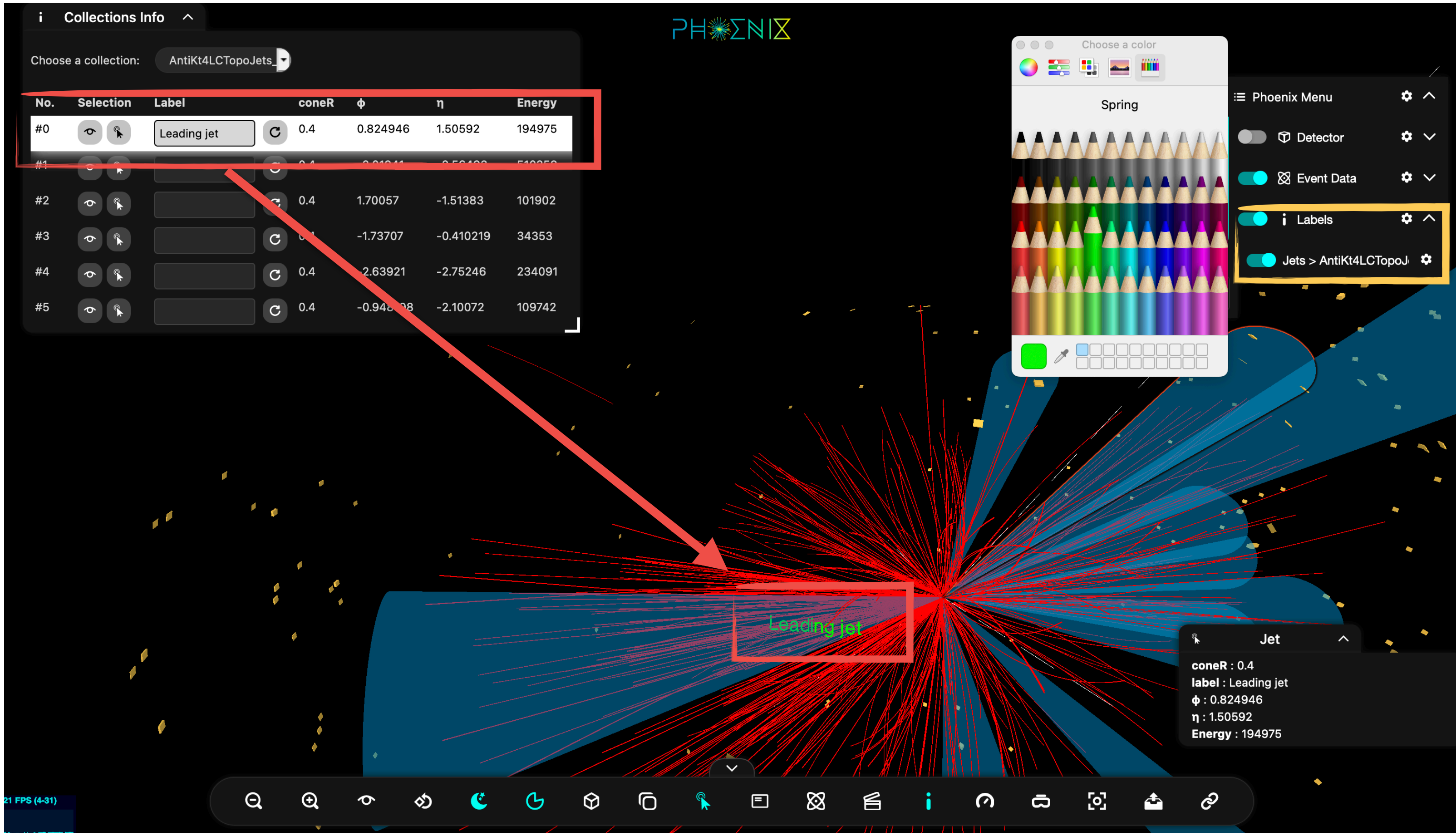
L1-EtMiss: N/A L1-SumEt: N/A

No LVL1Result collection in event

Click to close image, click and drag to move.
Use arrow keys for next and previous.

LABELS

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



Collections Info

Choose a collection: AntiKt4LCTopoJets_

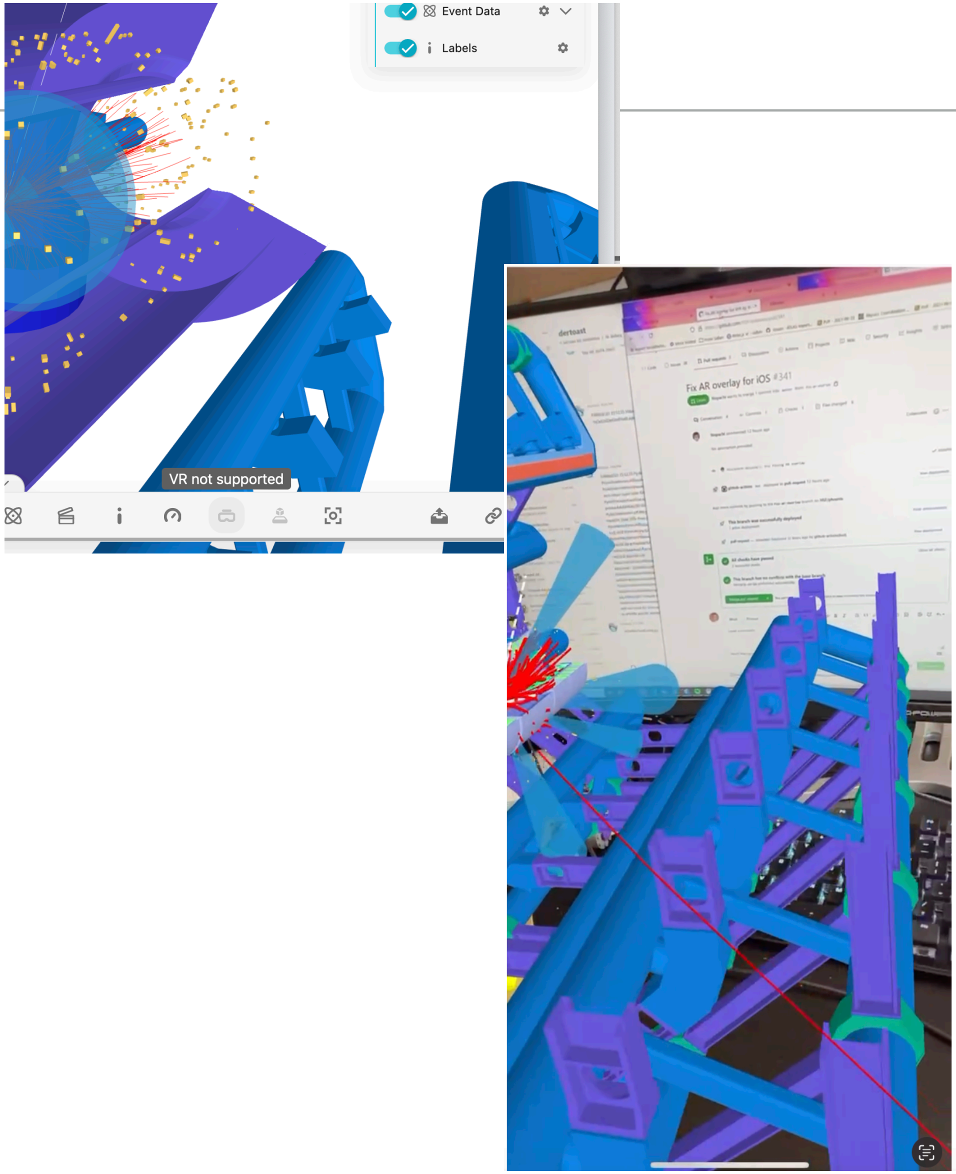
No.	Selection	Label	coneR	ϕ	η	Energy
#0		Leading jet	0.4	0.824946	1.50592	194975
#1			0.4	[illegible]	[illegible]	[illegible]
#2			0.4	1.70057	-1.51383	101902
#3			0.4	-1.73707	-0.410219	34353
#4			0.4	-2.63921	-2.75246	234091
#5			0.4	-0.948[illegible]08	-2.10072	109742

Jet

coneR : 0.4
label : Leading jet
ϕ : 0.824946
η : 1.50592
Energy : 194975

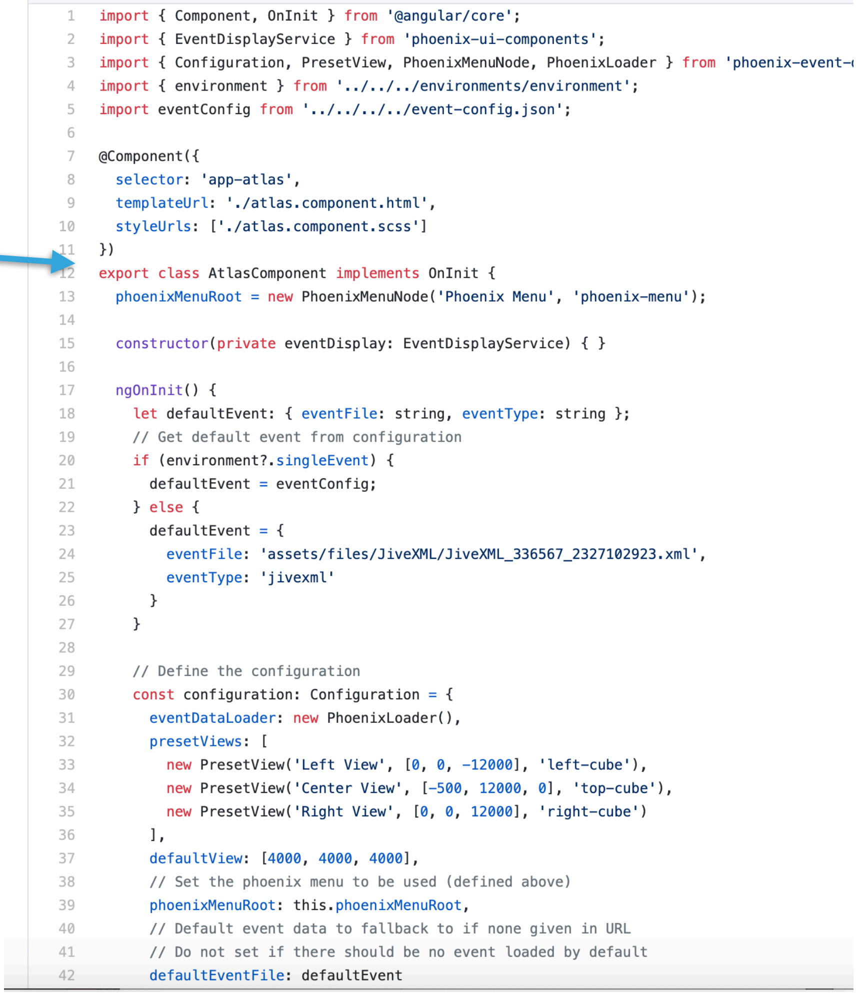
VR/AR

- 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



EXTENSIBILITY: ADDING A NEW DETECTOR

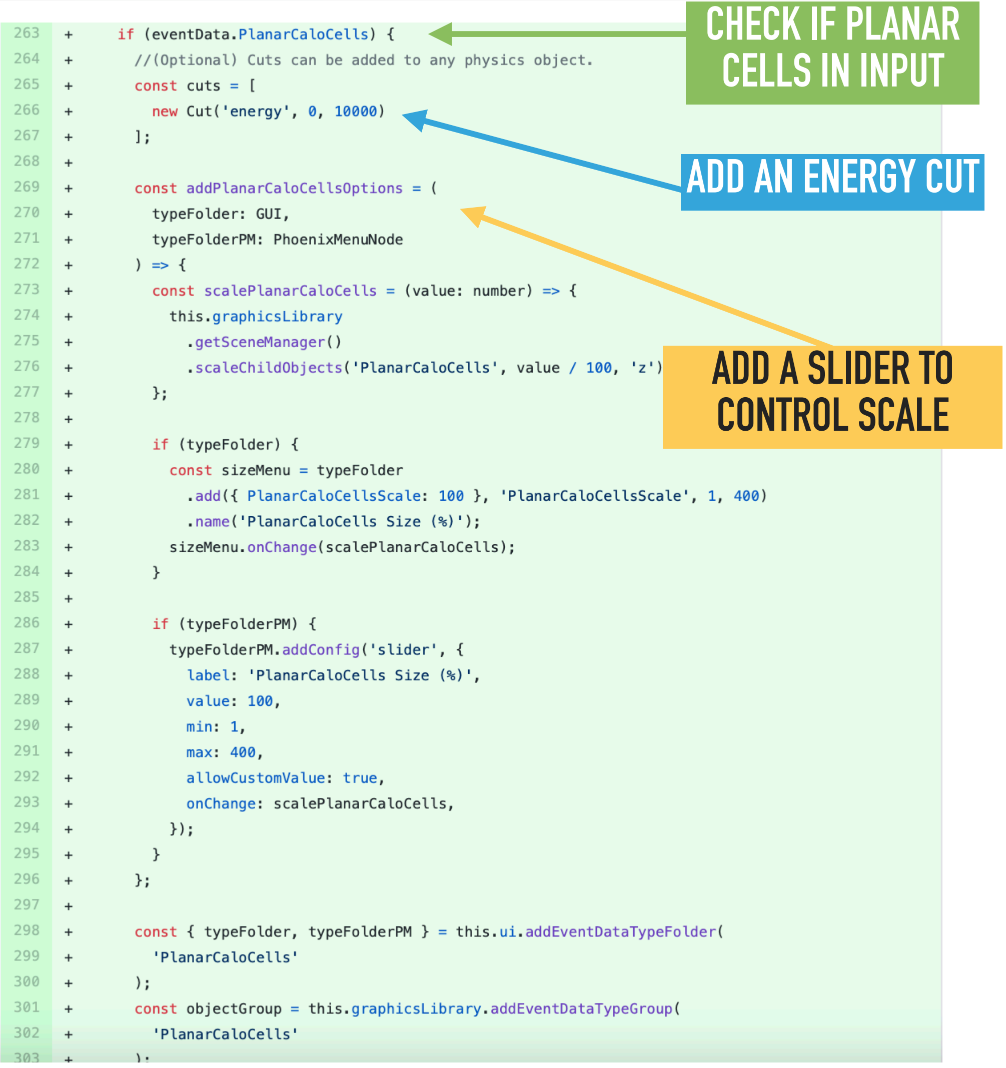
- **How would you add a new detector?**
- You basically need to add **two** files
  - `experiment.component.html` file (defines the 'view')
  - `experiment.component.ts` the experiment specific **implementation** i.e. file contains e.g.
    - The default configuration and event,
    - Loaders required (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: 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



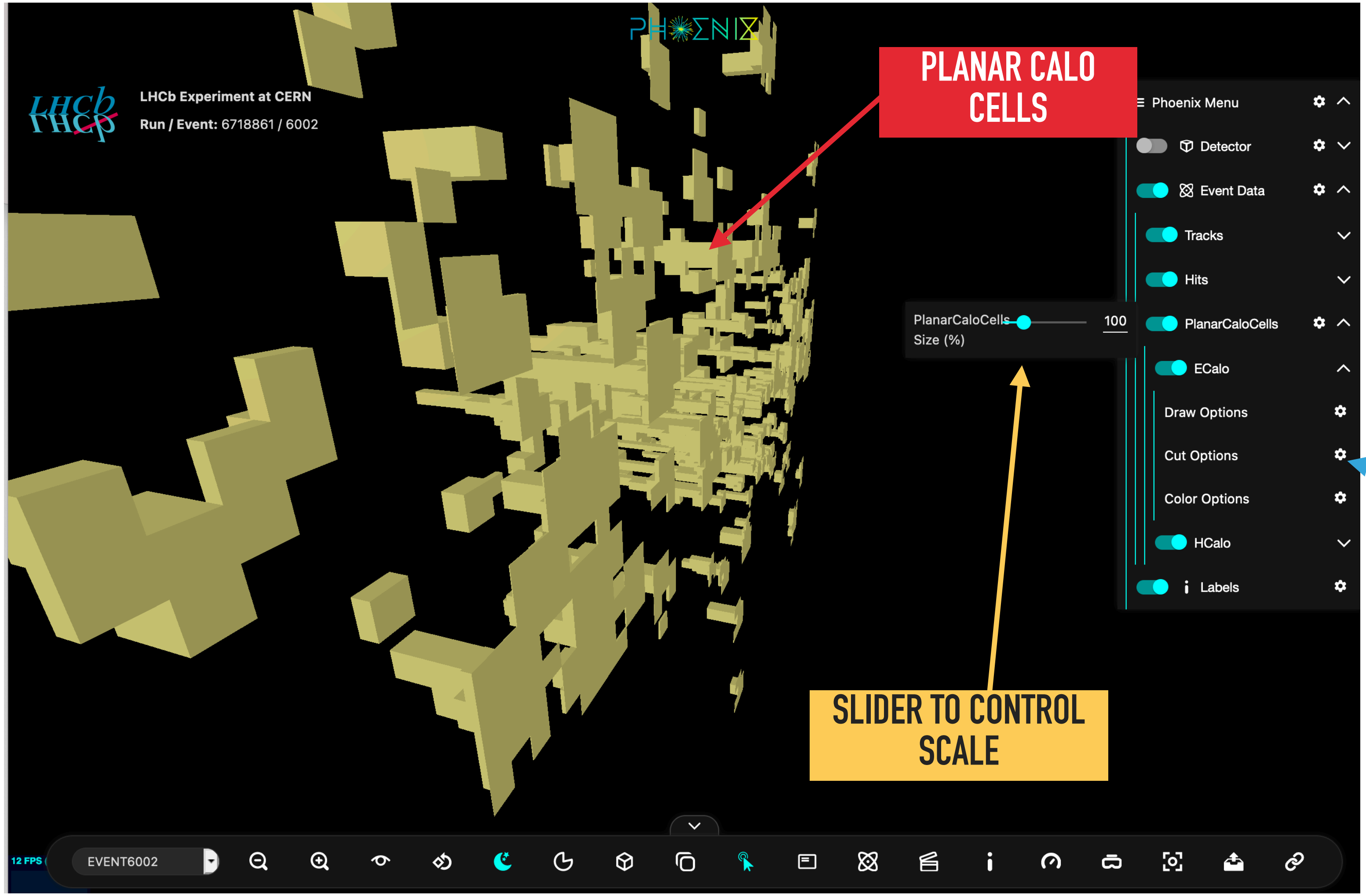
```
+      if (eventData.PlanarCaloCells) {
+        //(Optional) Cuts can be added to any physics object.
+        const cuts = [
+          new Cut('energy', 0, 10000)
+        ];
+
+        const addPlanarCaloCellsOptions = (
+          typeFolder: GUI,
+          typeFolderPM: PhoenixMenuNode
+        ) => {
+          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

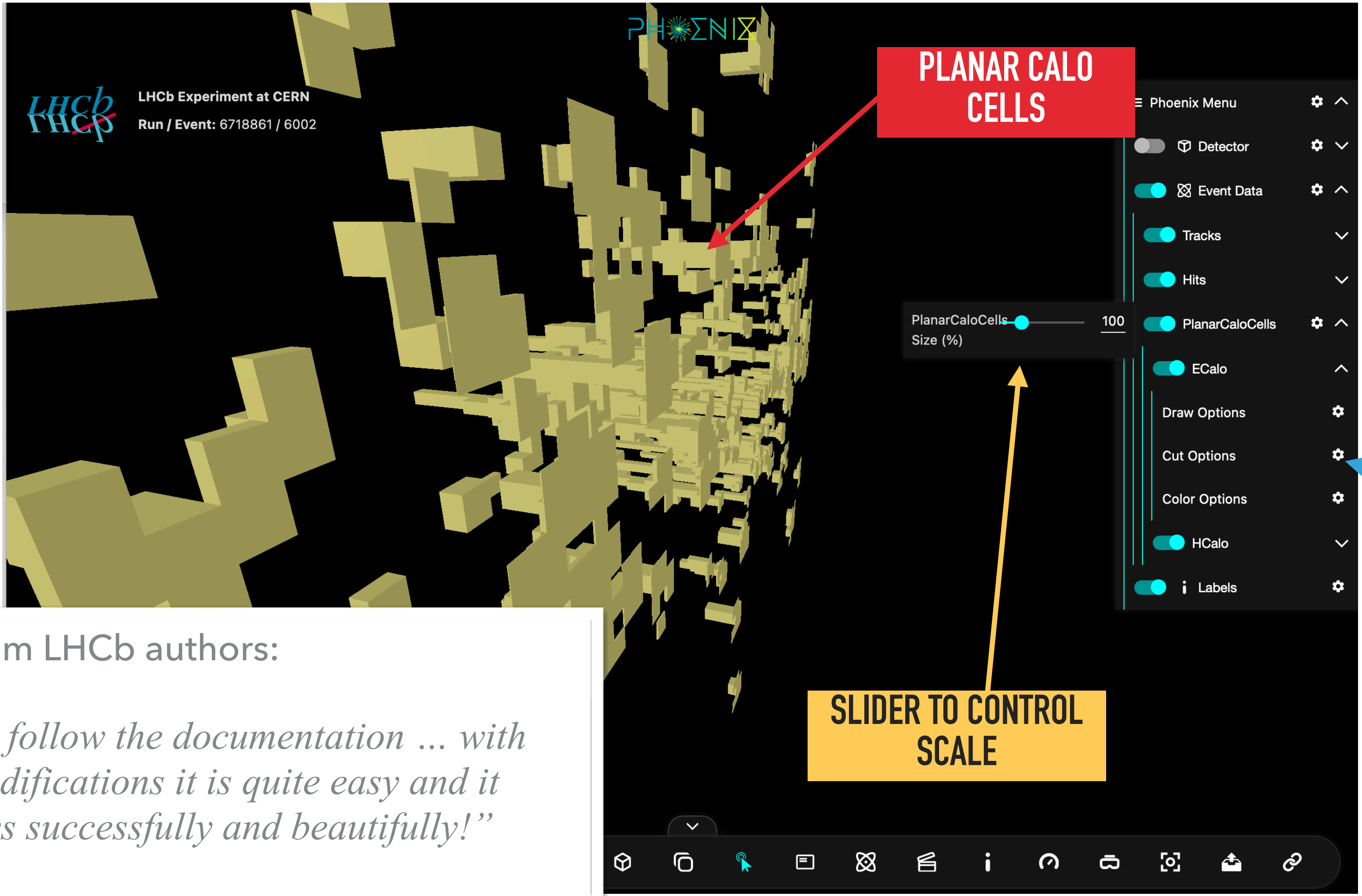


PLANAR CALO CELLS

SLIDER TO CONTROL SCALE

ENERGY CUT

ADDING A NEW PHYSICS OBJECT: RESULT



LHCb Experiment at CERN

Run / Event: 6718861 / 6002

PlanarCaloCells Size (%) 100

PLANAR CALO CELLS

SLIDER TO CONTROL SCALE

ENERGY CUT

Feedback from LHCb authors:

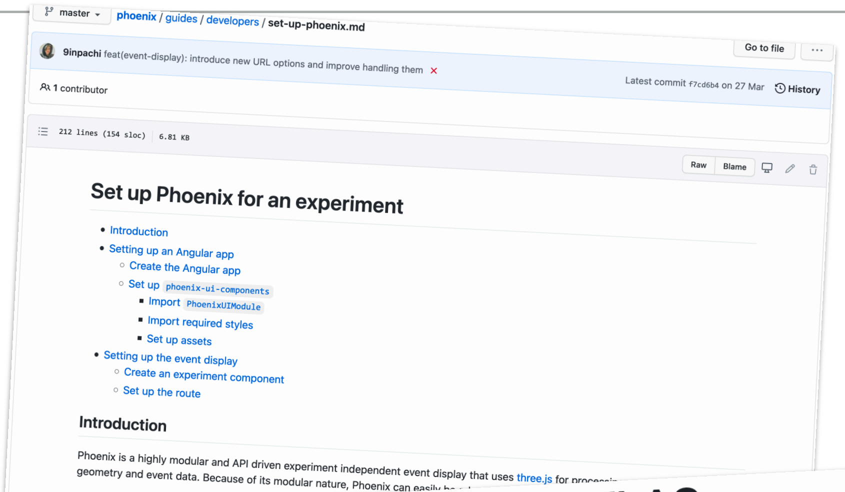
- *"if you follow the documentation … with few modifications it is quite easy and it displays successfully and beautifully!"*
- *"Surprisingly easy to add new objects"*

EXTENSIBILITY: ROLL YOUR OWN VERSION

- Of course, you can use Phoenix as part of an entirely independent application i.e.
  - Your own repository, your own default configuration etc
  - No Phoenix home screen with demos for other experiments
- Just install phoenix, following the detailed instructions

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

- Example: PhoenixATLAS



master / phoenix / guides / developers / set-up-phoenix.md

9inpachi feat(event-display): introduce new URL options and improve handling them ×

1 contributor

Latest commit f7cd6b4 on 27 Mar History

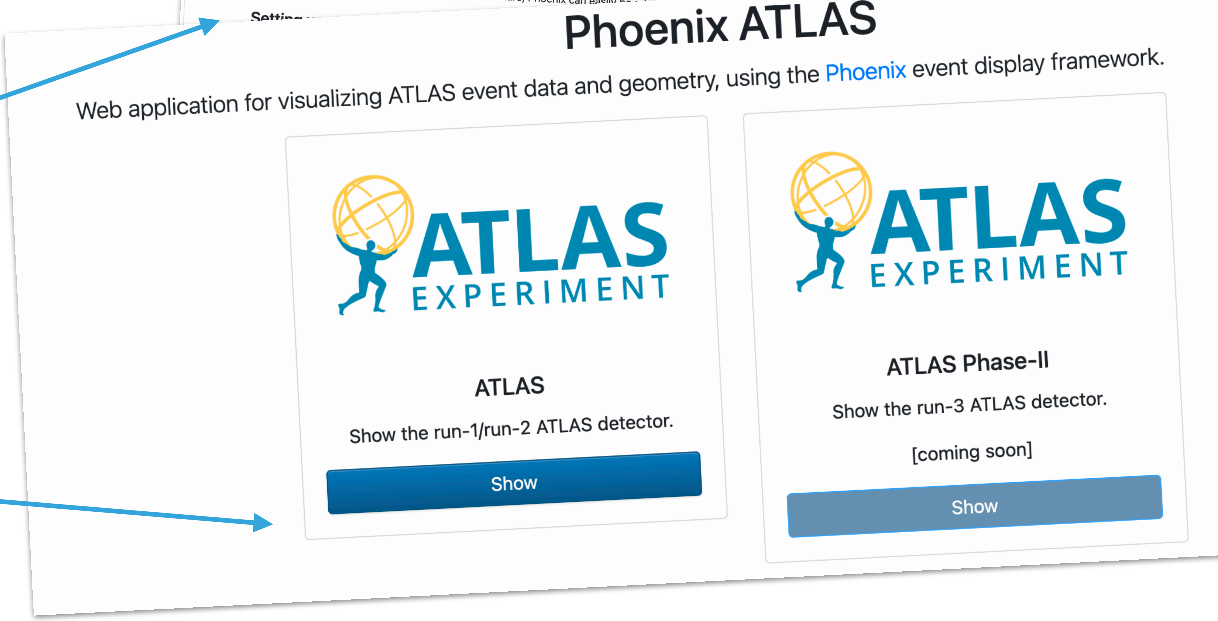
212 lines (154 sloc) | 6.81 KB

Set up Phoenix for an experiment

- Introduction
- Setting up an Angular app
  - Create the Angular app
  - Set up `phoenix-ui-components`
    - Import `PhoenixUIModule`
    - Import required styles
    - Set up assets
- Setting up the event display
  - Create an experiment component
  - Set up the route

Introduction

Phoenix is a highly modular and API driven experiment independent event display that uses three.js for processin
geometry and event data. Because of its modular nature, Phoenix can easily be



Phoenix ATLAS

Web application for visualizing ATLAS event data and geometry, using the Phoenix event display framework.

ATLAS

Show the run-1/run-2 ATLAS detector.

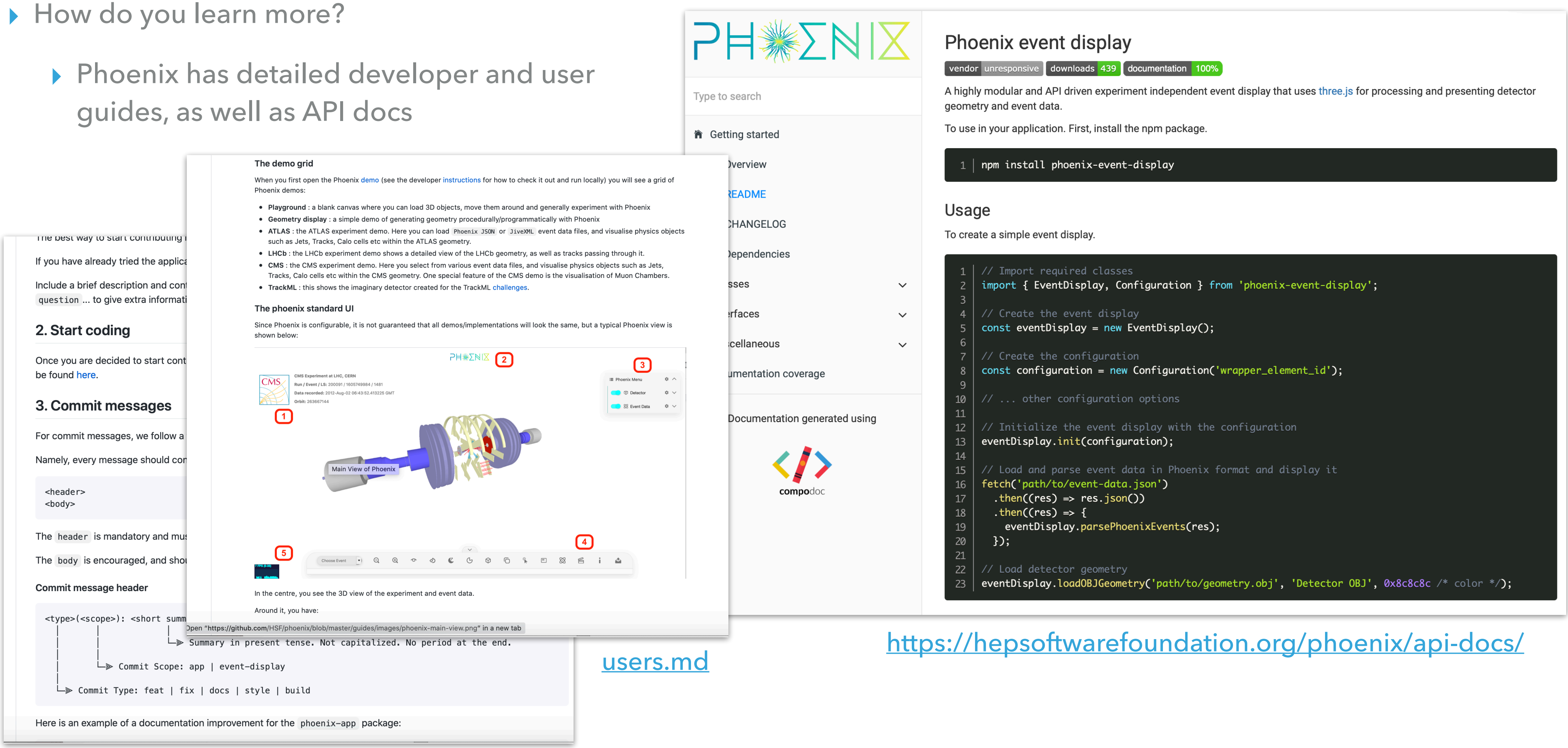
ATLAS Phase-II

Show the run-3 ATLAS detector.

[coming soon]

DOCUMENTATION

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



The best way to start contributing i

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 con

```
<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:

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:

PHOENIX
2
CMS
CMS Experiment at LHC, CERN
Run / Event / LS: 200091 / 1605749984 / 1481
Data recorded: 2012-Aug-02 06:43:52.413225 GMT
Orbit: 263667144
1
3
Phoenix Menu
Detector
Event Data
Main View of Phoenix
5
4
Choose Event

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

Phoenix event display

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 */);
```

users.md

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

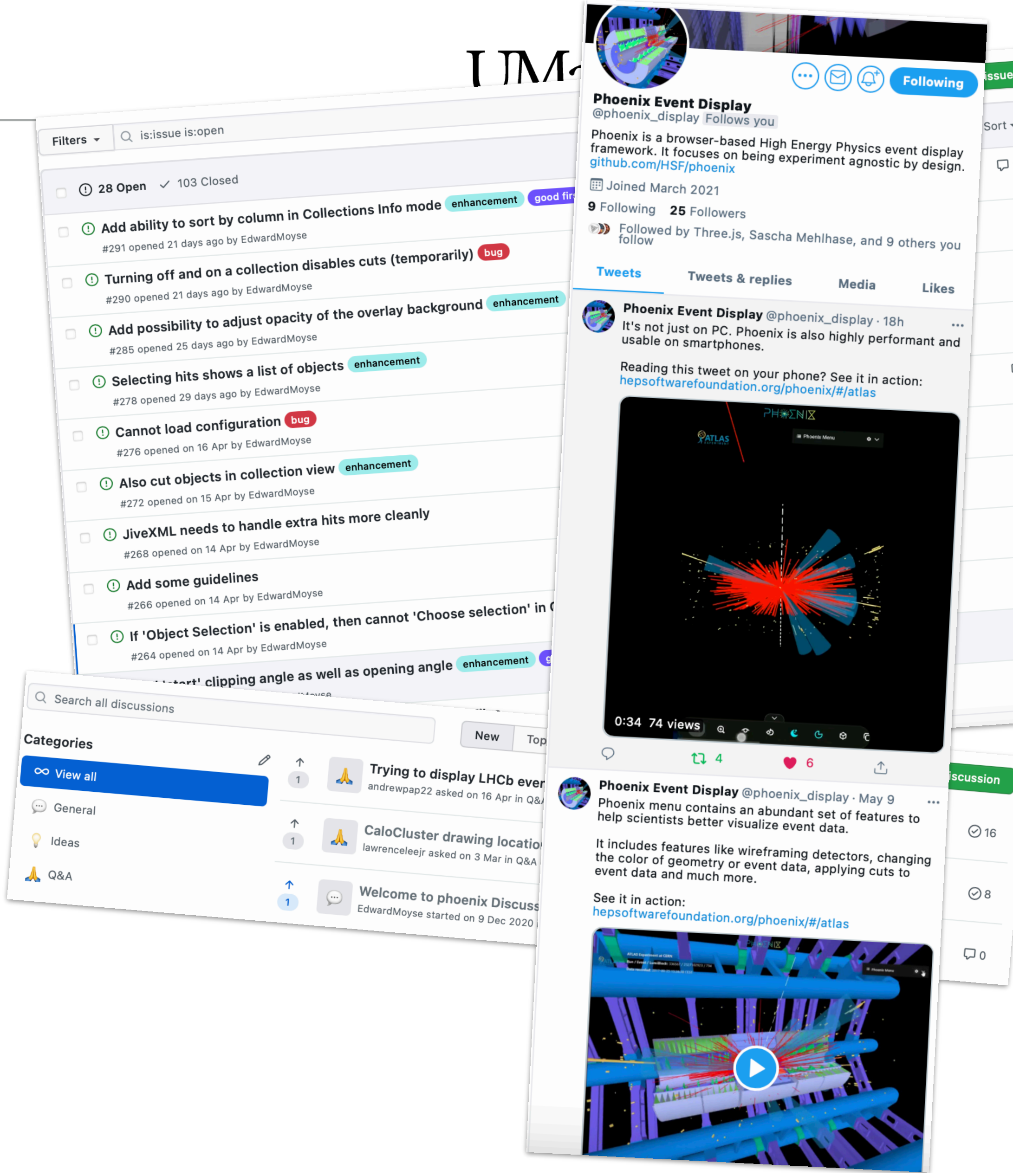
CONTRIBUTING.md

HOW DOES PHOENIX MATCH THE EPIC REQUIREMENTS?

- I went through the requirements document, and mostly it all seems fine. I had a few comments…
- Section 1
  - *Subsystem-Specific Troubleshooting* - not sure I understand what this means?
- Section 2
  - *Streaming readout* - ditto?
  - *Automated tools compatible & Batch mode graphics* - we do not yet have a batch mode
  - *Security* - we use industry standard tools such as threejs, angular, node etc
  - *Visualization Capabilities* - showing active detector elements can be shown, but this is not trivial and needs improvements.
  - *Remote data sources* - we can load data from local directories (can be network mounted) on server, or via URL. Is this sufficient?

CONCLUSION

- **Very brief overview of Phoenix**
  - Didn't have time to cover many features, such as the integrated RK propagator, object collection cuts etc etc
- 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



Filters ▾ is:issue is:open

28 Open ✓ 103 Closed

Add ability to sort by column in Collections Info mode enhancement good fir
#291 opened 21 days ago by EdwardMoyse

Turning off and on a collection disables cuts (temporarily) bug
#290 opened 21 days ago by EdwardMoyse

Add possibility to adjust opacity of the overlay background enhancement
#285 opened 25 days ago by EdwardMoyse

Selecting hits shows a list of objects enhancement
#278 opened 29 days ago by EdwardMoyse

Cannot load configuration bug
#276 opened on 16 Apr by EdwardMoyse

Also cut objects in collection view enhancement
#272 opened on 15 Apr by EdwardMoyse

JiveXML needs to handle extra hits more cleanly
#268 opened on 14 Apr by EdwardMoyse

Add some guidelines
#266 opened on 14 Apr by EdwardMoyse

If 'Object Selection' is enabled, then cannot 'Choose selection' in
#264 opened on 14 Apr by EdwardMoyse

'start' clipping angle as well as opening angle enhancement

Search all discussions

New Top

Categories

- View all
- General
- Ideas
- Q&A

1 Trying to display LHCb ever
andrewpap22 asked on 16 Apr in Q&A

1 CaloCluster drawing locatio
lawrenceleejr asked on 3 Mar in Q&A

1 Welcome to phoenix Discuss
EdwardMoyse started on 9 Dec 2020

16 8 0

Following

Phoenix Event Display
@phoenix_display Follows you

Phoenix is a browser-based High Energy Physics event display framework. It focuses on being experiment agnostic by design.
github.com/HSF/phoenix

Joined March 2021

9 Following **25** Followers

Followed by Three.js, Sascha Mehlhase, and 9 others you follow

Tweets · Tweets & replies · Media · Likes

Phoenix Event Display @phoenix_display · 18h
It's not just on PC. Phoenix is also highly performant and usable on smartphones.

Reading this tweet on your phone? See it in action:
hepsoftwarefoundation.org/phoenix/#/atlas

0:34 74 views

4 6

Phoenix Event Display @phoenix_display · May 9
Phoenix menu contains an abundant set of features to help scientists better visualize event data.

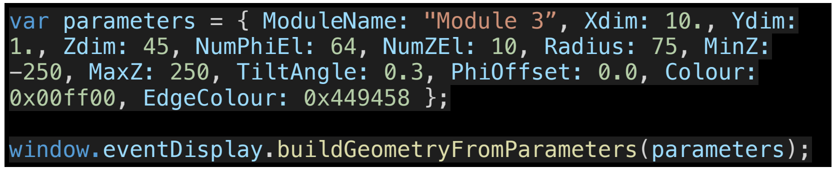
It includes features like wireframing detectors, changing the color of geometry or event data, applying cuts to event data and much more.

See it in action:
hepsoftwarefoundation.org/phoenix/#/atlas

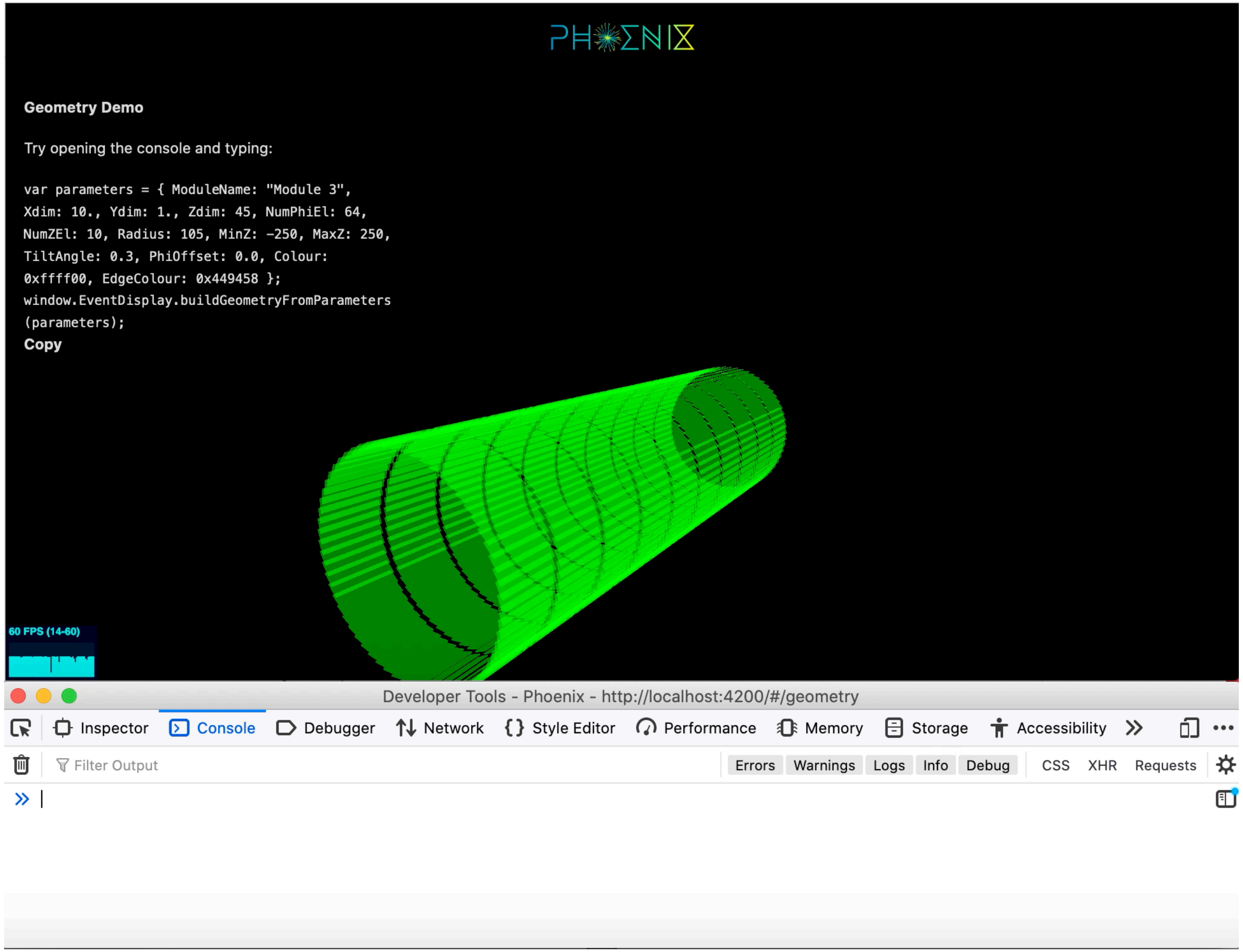
BACKUP

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);
```



Geometry Demo

Try opening the console and typing:

```
var parameters = { ModuleName: "Module 3",
Xdim: 10., Ydim: 1., Zdim: 45, NumPhiEl: 64,
NumZEl: 10, Radius: 105, MinZ: -250, MaxZ: 250,
TiltAngle: 0.3, PhiOffset: 0.0, Colour:
0xffff00, EdgeColour: 0x449458 };
window.EventDisplay.buildGeometryFromParameters
(parameters);
```

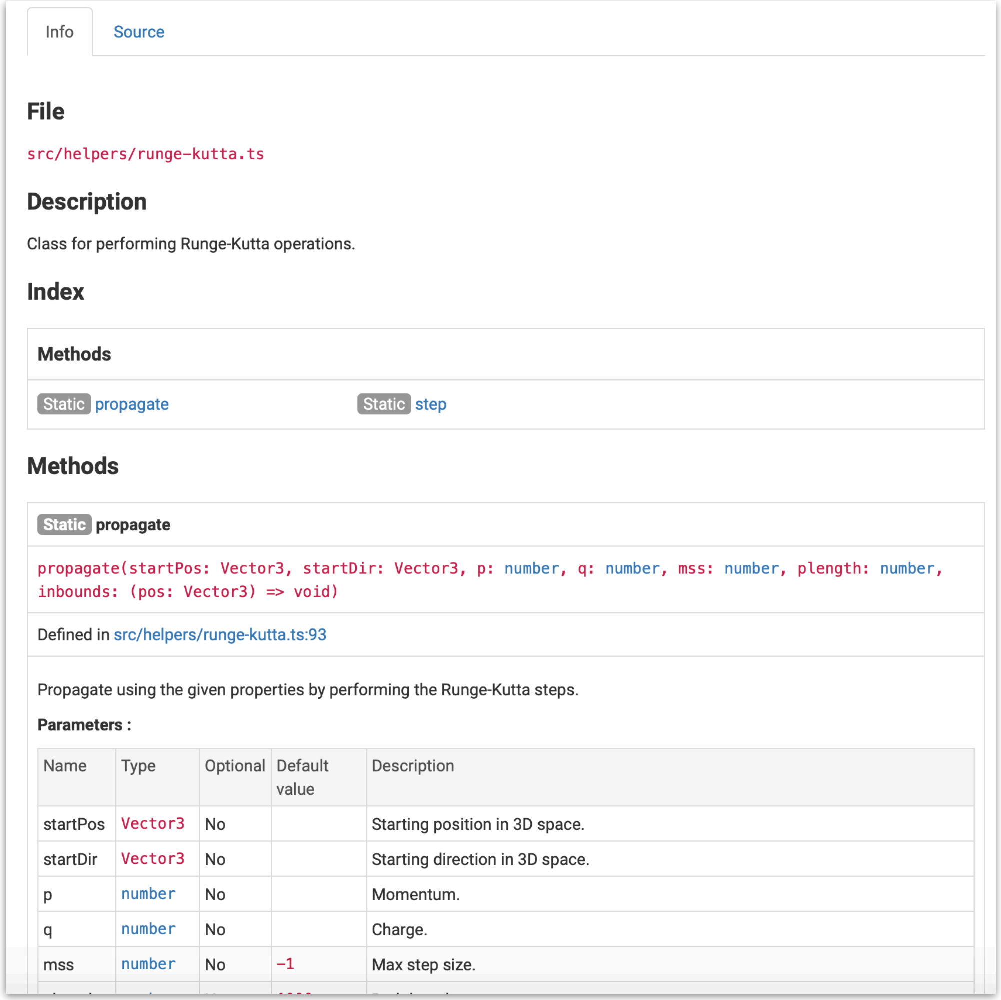
Copy

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!



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 :

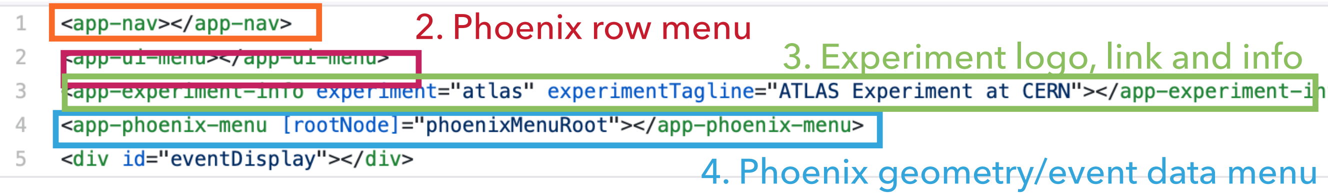
Name	Type	Optional	Default value	Description
startPos	`Vector3`	No		Starting position in 3D space.
startDir	`Vector3`	No		Starting direction in 3D space.
p	`number`	No		Momentum.
q	`number`	No		Charge.
mss	`number`	No	`-1`	Max step size.

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

EXTENSIBILITY

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

1. Link back to main Phoenix page

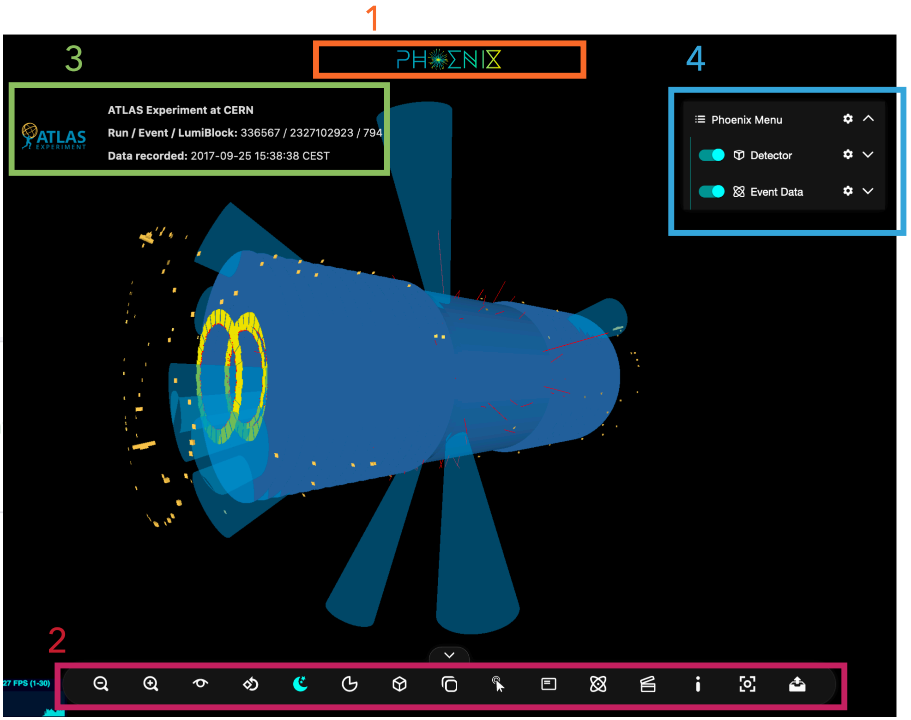


```
<app-nav></app-nav>
<app-ui-menu></app-ui-menu>
<app-experiment-info experiment="atlas" experimentTagline="ATLAS Experiment at CERN"></app-experiment-in
<app-phoenix-menu [rootNode]="phoenixMenuRoot"></app-phoenix-menu>
<div id="eventDisplay"></div>
```

2. Phoenix row menu

3. Experiment logo, link and info

4. Phoenix geometry/event data menu



ATLAS Experiment at CERN

Run / Event / LumiBlock: 336567 / 2327102923 / 794

Data recorded: 2017-09-25 15:38:38 CEST

atlas.component.html