ILD Phoenix Event Display

Creating an event display for ILD detectors, a way to add future detector designs and converting ILD geometry files to the right file format

Abigail Harrison
Hamburg, 05/09/2024

ILD Phoenix Event Display

01 The International Large Detector (ILD)

02 Event Displays

03 Conversion Scripts (XML to GlTF)

04 The ILD Phoenix Event Display

05 Comparison

The ILD at the International Linear Collider (ILC)

ILC

- The ILC is the linear collider planned to be built in Japan which collides electrons with positrons.

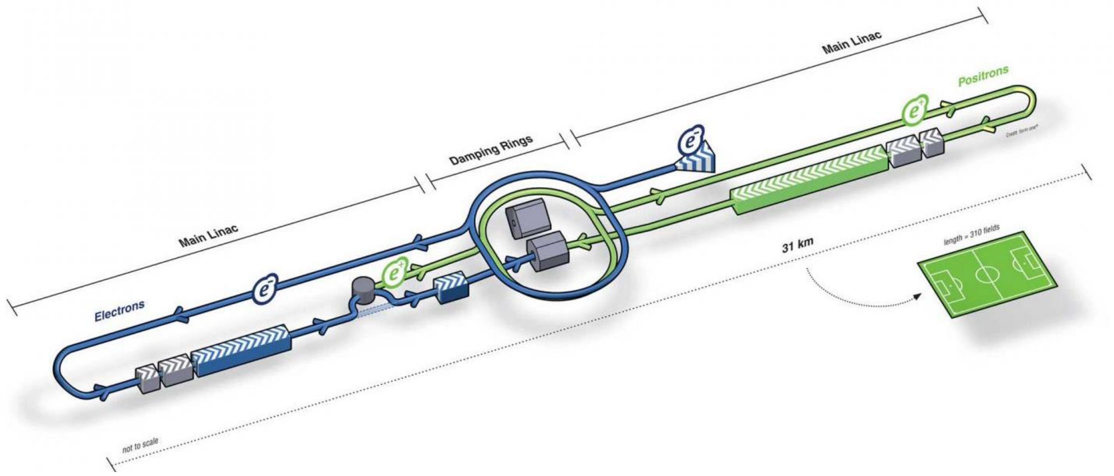


Figure 1: Diagram of the ILC.

ILD

- The ILD is a detector that will be used at the ILC.
- The ILD uses high granular calorimeters with excellent tracking which is needed for the particle flow reconstruction.

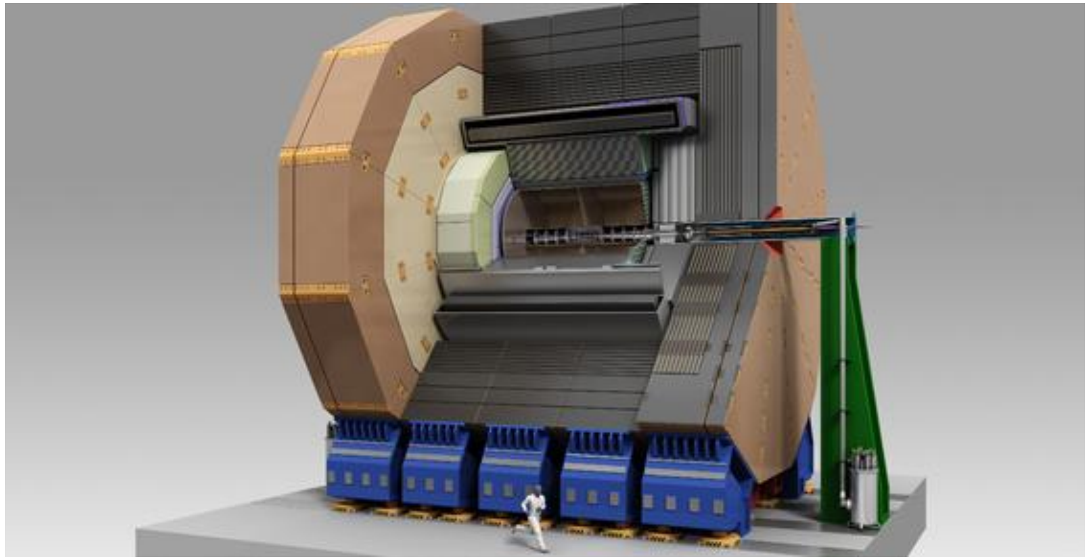


Figure 2: CAD Drawing of the ILD.

Event Displays

- Event displays allow for the visualization of subatomic particles propagation through matter.
- Some early examples of event displays include:
  - Cloud chambers
  - Bubble chambers
  - Spark chambers
- Modern event displays use digital data to display particle tracks and hits as well as the detector itself.

Figure 3: Spark Chamber.

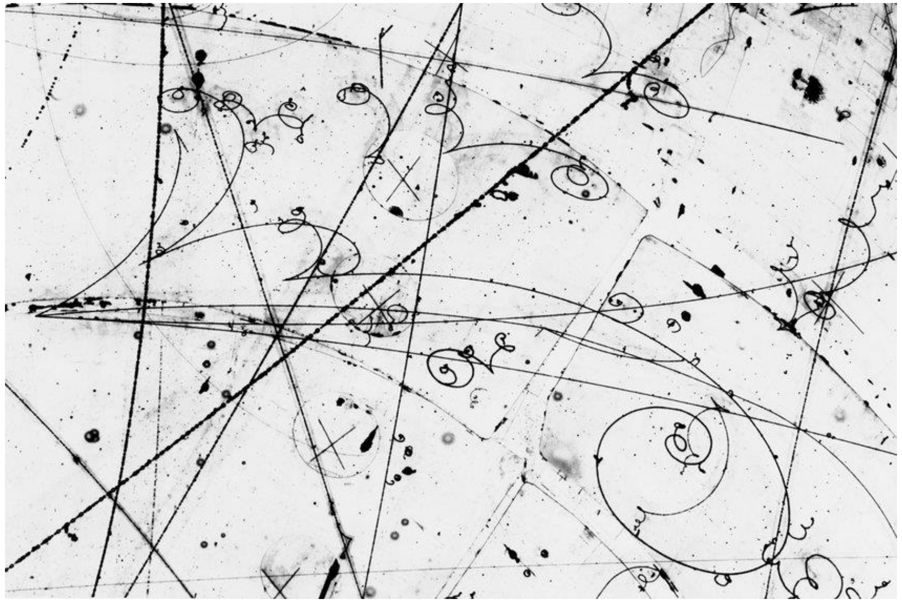


Figure 4: Bubble Chamber Tracks Image.

C Event Display (CED)

- One event display for the ILD, written in C that draws 3D images using OpenGL.
- The event display is still currently working but is quite old and therefore hard to maintain and not extendable.
- Using the CED also involves having the correct environment.

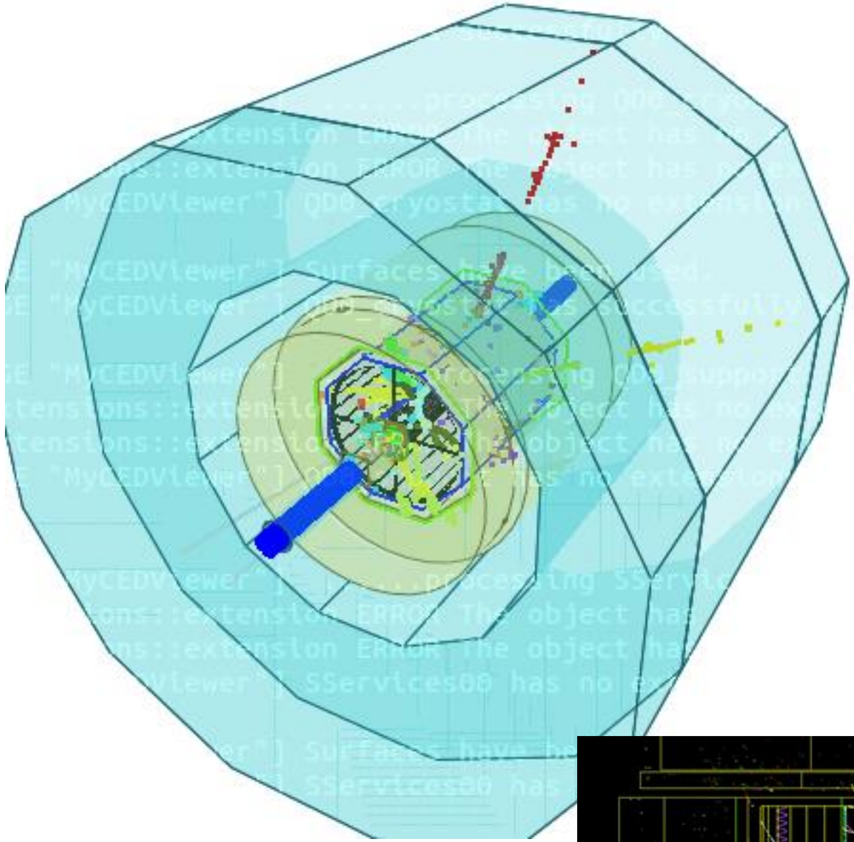


Figure 5: CED image of detector.

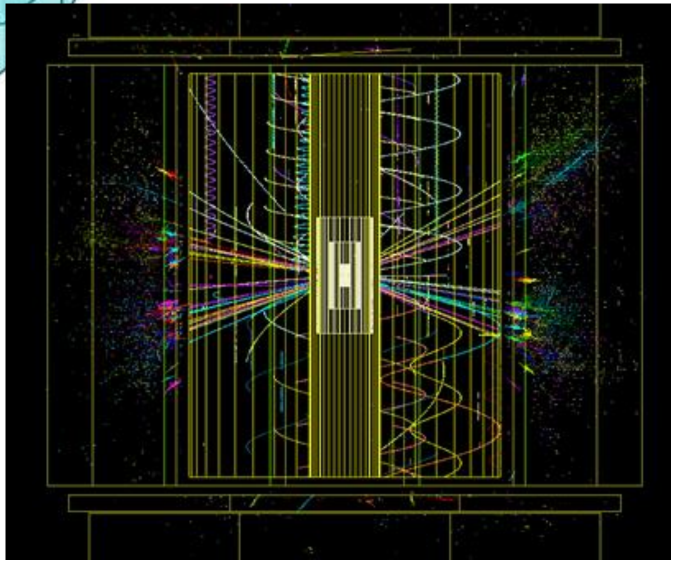


Figure 6: CED detector image with side view with wire mesh.

The Phoenix Event Display

Introduction

- The Phoenix Event Display was developed for ATLAS and there are also applications for the detectors at FCC and CMS.
- An event display using Typescript and three.js and Angular.
- The design principles for Phoenix state that the event display should:
  - Have good documentation.
  - Be easily accessible (available through a browser).
  - Avoid experiment-specific assumptions.

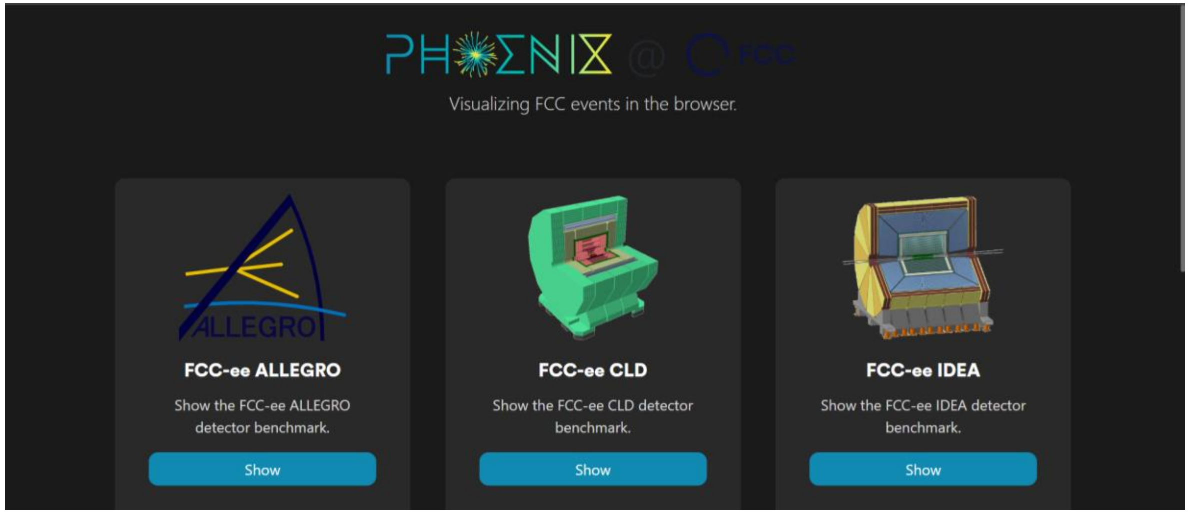


Figure 7: FCC Phoenix Event Display Main Page.

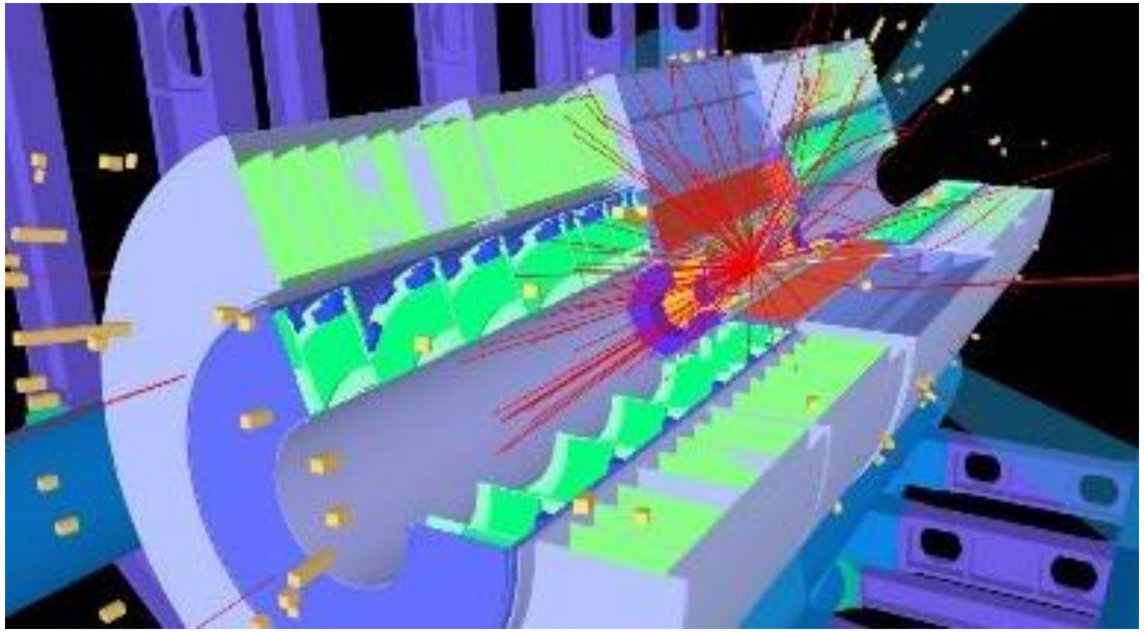


Figure 8: Example Image of a Detector using Phoenix.

The Phoenix Event Display

Features

- There are two menus in Phoenix: the Side Bar and the Bottom Menu.
- The Side bar allows users to:
  - Change appearance of the subdetector (colour, opacity).
  - Toggle subdetectors and event data.
  - Save different visualisation states.
- The bottom menu allows users to complete functions including:
  - Clip the detector (slice the detector at an angle to see the inside)
  - Add an event and look through the hits and tracks
  - Look at different views of the detector
  - Look at coordinates, lengths or object labelling

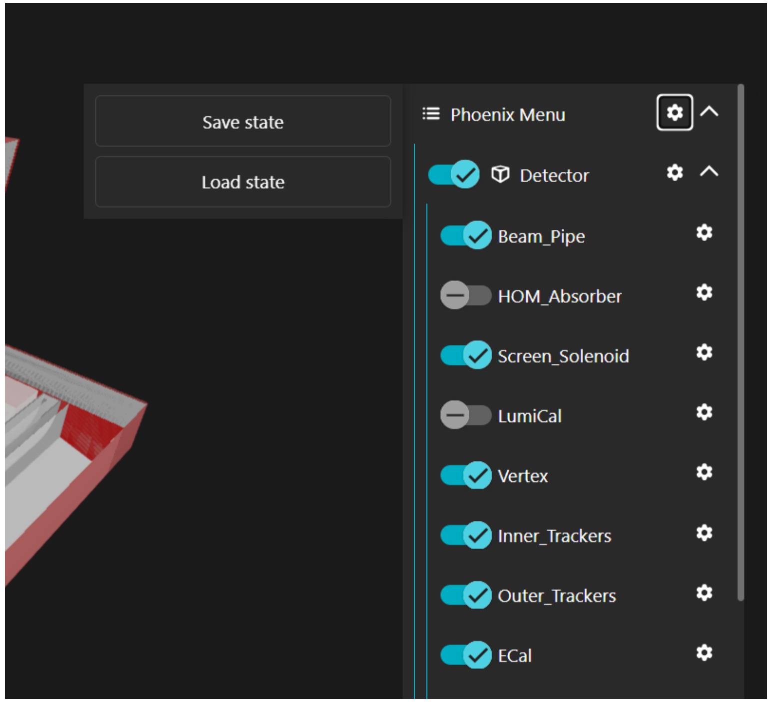
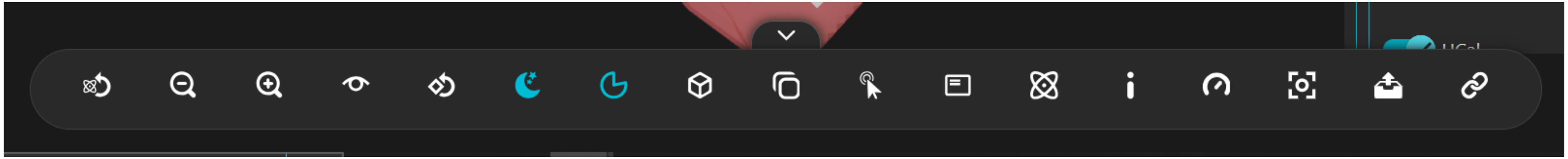


Figure 9: Phoenix Menu in Phoenix FCC.

Figure 10: Bottom Menu in Phoenix FCC.



Converting ILD Detector XML Files to GlTF

Introduction

- The ILD detector files are stored as XML files which are DD4hep geometries (they can be found at: https://github.com/key4hep/k4geo).
- For the Phoenix event display to read the detector files they need to be stored in the GlTF file which is a standard format for 3D scenes.

- This can be quite a lengthy procedure as it involves two steps (converting XML to ROOT and then ROOT to GlTF).
- Therefore, it was important to make a one-step automatic conversion script that would run each part of the conversion in sequence.
- An automatic configuration file generator was added to the one-step conversion, as the conversion of ROOT to GlTF requires a configuration file.

Converting ILD Detector XML Files to GlTF

Total Conversion

- This first conversion used DD4hep in the ROOT package in python to convert the DD4hep XML files into TGeo ROOT files.
- The user can set the number of layers that the ROOT file sees (‘visibility’).
- The second conversion from ROOT to GlTF, uses Javascript code from our FCCee colleagues (https://github.com/abbyxh/root2gltf.git).

```python
def root_convert(cfile, out_path, visibility):
    ## Converts an xml file to root file
    print('INFO: Converting following compact file(s):')
    print('       ' + cfile)

    ROOT.gSystem.Load('libDDCore')
    description = ROOT.dd4hep.Detector.getInstance()
    description.fromXML(cfile)

    ## Sets the number of layers visible in the root file
    ROOT.gGeoManager.SetVisLevel(visibility)
    ROOT.gGeoManager.SetVisOption(0)
    ## Sets an automatic root path if one isn't given
    root_path = determine_outpath(out_path, cfile, 'root')
    ROOT.gGeoManager.Export(root_path)

    return root_path
```

Figure 11: Code for converting file from xml to root.

My Contribution to the Conversion

Automatic Configuration File

- The conversion script provides an automatic configuration file that the user can either use or edit.
- The different variables that the user can determine in the configuration file:
  - What parts of the detector should be shown.
  - The opacity for each subdetector (set automatically at 0.8).
  - The colour (can either choose ‘ILD colouring’ or default colouring.

```
{
    "childrenToHide": ["SServices00"],
    "subParts": {
        "BeBeampipe": [
            [
                "BeBeampipe_(?!envelope)\\w+"
            ],
            0.6, [0,1,0]
        ],
        "BeamPipe": [
            [
                "BeamPipe_(?!envelope)\\w+"
            ],
            0.6, [0,1,0]
        ],
```

Figure 12: Example of edited configuration file.

```
"FTD": [
    ["(FTD_(?!envelope))\\w+|(FTD(?!_))\\w+",
    "FTDDisk_.*_negZ\\w+",
    "ftd_petal_negZ_.*_.*\\w+",
    "ftd_sensor_negZ_.*_.*_.*\\w+",
    "FTDDisk_.*_posZ\\w+",
    "ftd_petal_posZ_.*_.*\\w+",
    "ftd_sensor_posZ_.*_.*_.*\\w+",
    "FTD_support\\w+"],
    0.8,
    [0.39, 0.1, 0.57]
],
"HcalBarrel": [
    ["(HcalBarrel_(?!envelope))\\w+|(HcalBarrel(?!_))\\w+"],
    0.8,
    [0.76, 0.76, 0.19]
],
```

Figure 13: Example of automatic configuration file.

My Contribution to the Conversion

Adding ILD Colours

- Developed functionality to make colours configurable.
- User inputs ‘ild’ as a command line and the ILD colours are added.

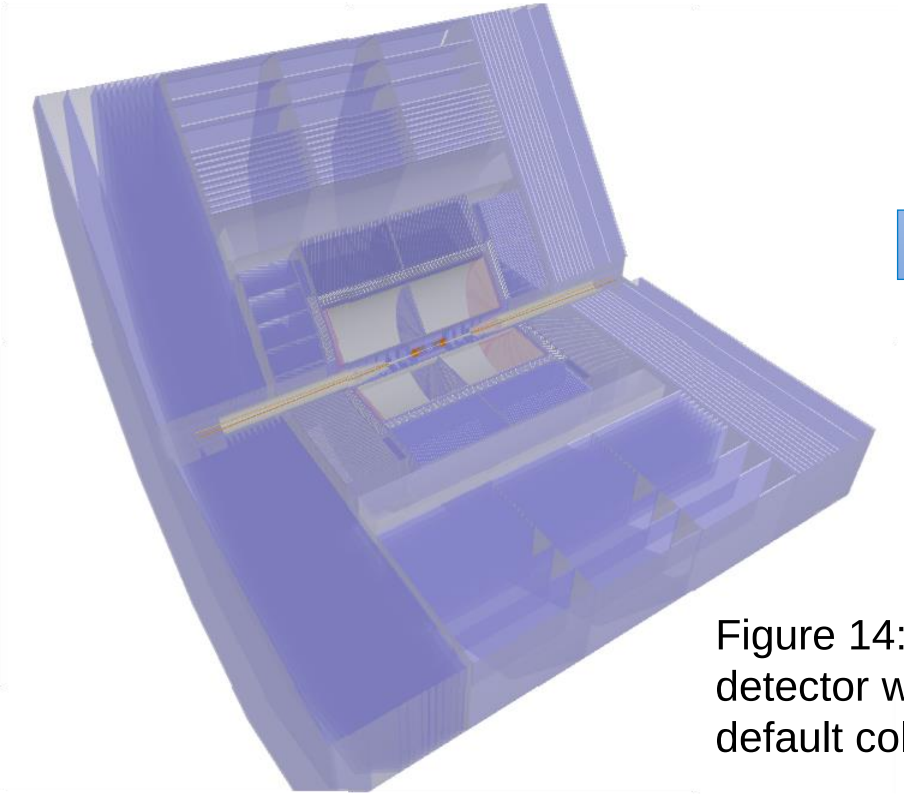


Figure 14: ILD detector with the default colouring.

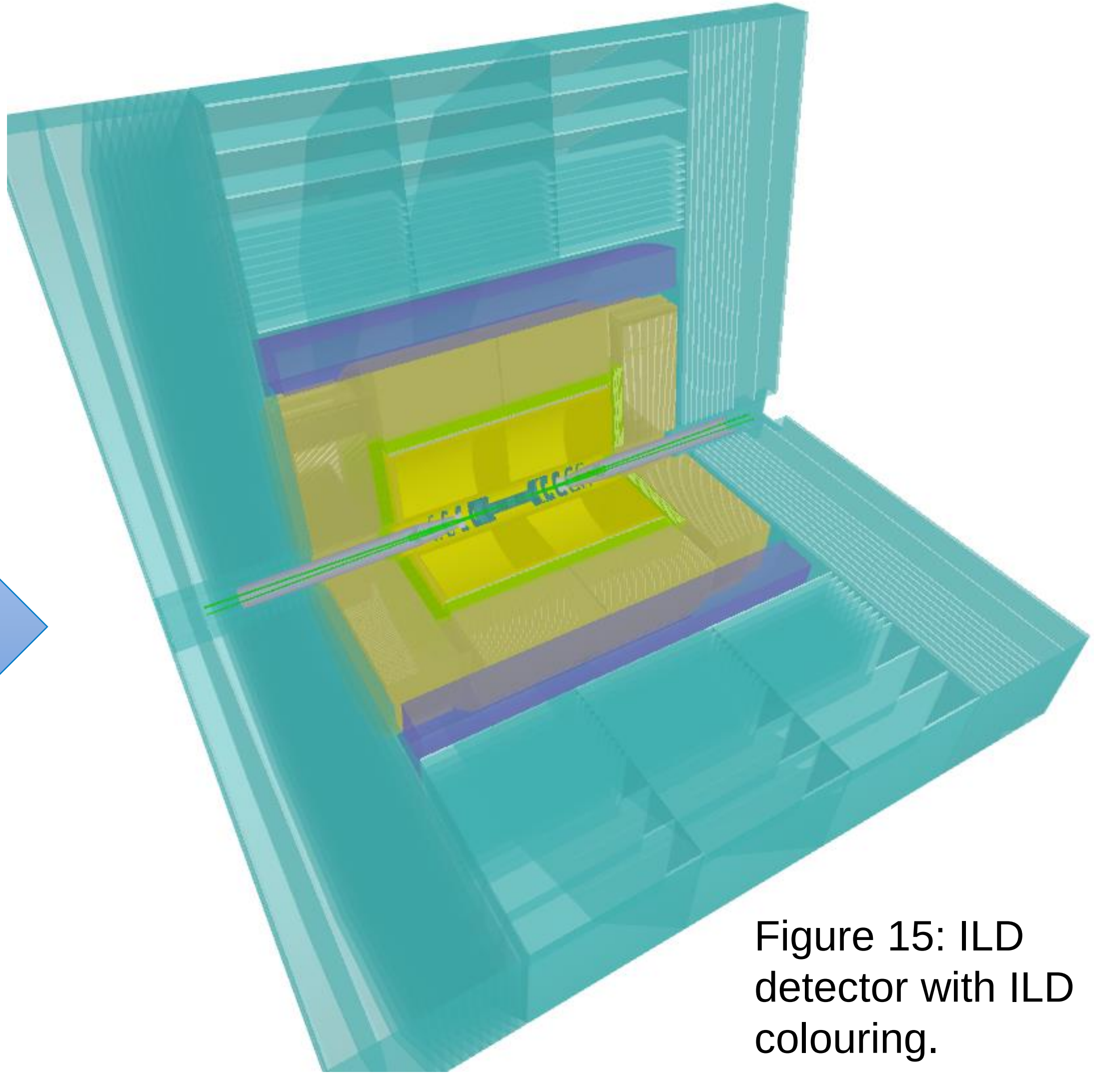


Figure 15: ILD detector with ILD colouring.

The ILD Phoenix Event Display

- The ILD Phoenix event display uses EDM4hep JSON files as input for event data.
- The ILD event display can be viewed on github pages using the link below:
  - https://ilcsoft.github.io/Phoenix-ILD/

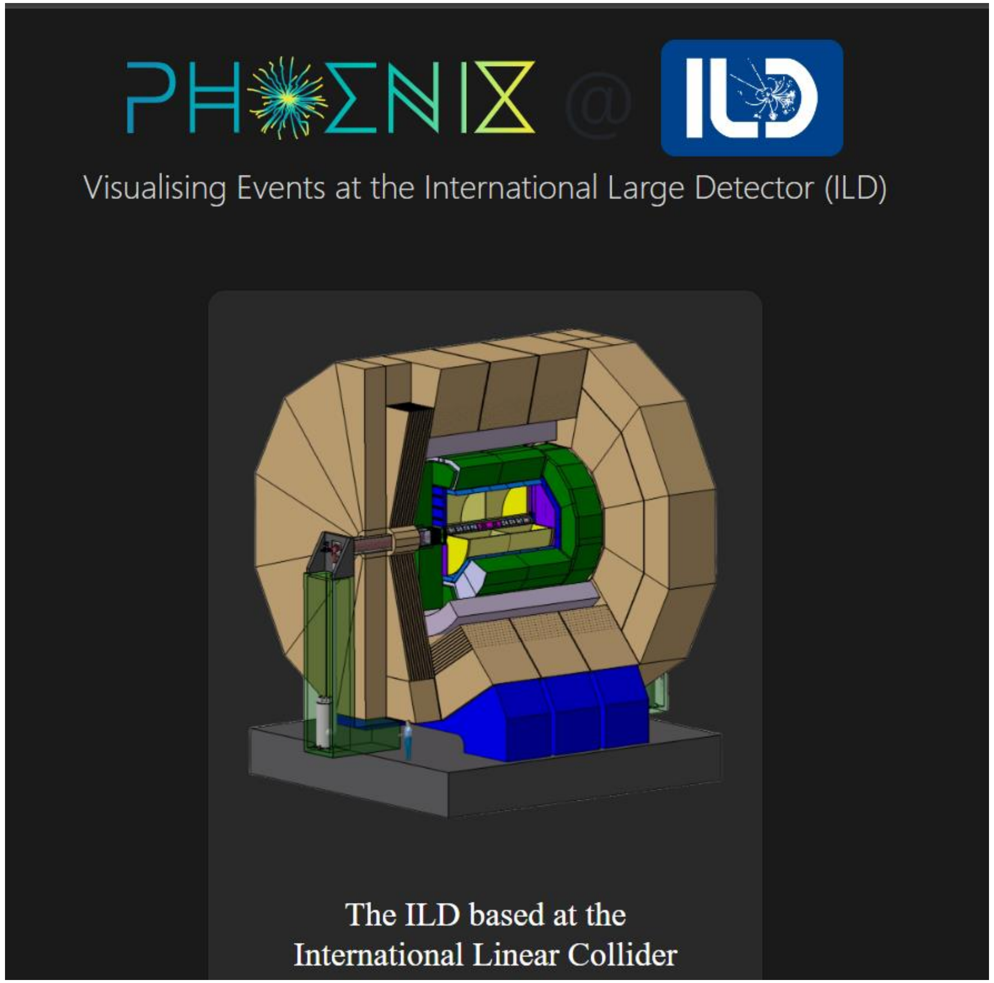


Figure 16: ILD Phoenix Event Display Main Page.

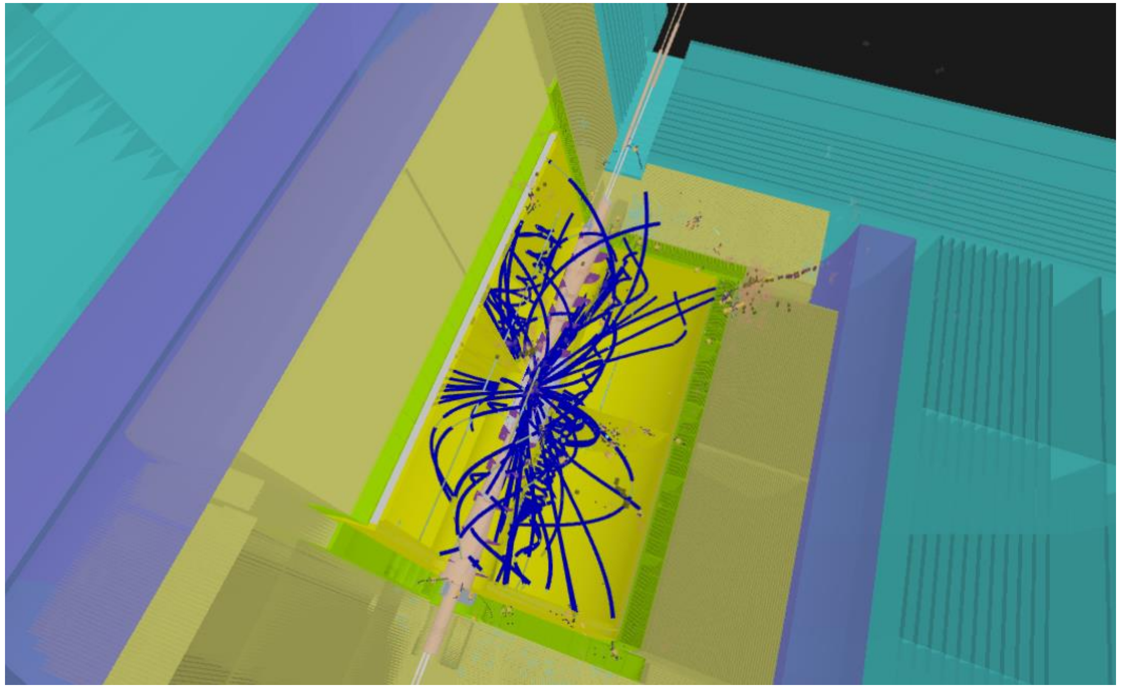


Figure 17: ILD Phoenix Event Display.

Adding a New Detector

- A new detector can be added using the command in the utils folder:

```
add_detector_component.py -d ... -n ...
```

- Where ‘d’ is the location of the GITF file and ‘n’ is the name of the detector.
- This method uses jinja templates to add a new detector Typescript, HTML and CSS file while adding a button to the drop-down menu.
- The app is then rebuilt and deployed again to github pages using a github action.

```
{%- for f in folders %}
import { {{ f }}Component } from './{{ f }}/{{ f }}.component';
{%- endfor %}

let routes: Routes = [
    {%- for f in folders %}
    { path: '{{ f }}', component: {{ f }}Component },
    {%- endfor %}
    { path: '', component: MainComponent }
]
```

Figure 18: Example of jinja template.

```
  - name: build app
    run: ng build --output-path ild-phoenix-app --base-href /Phoenix-ILD/

  - name: Upload artifact
    uses: actions/upload-pages-artifact@v3
    with:
      path: ild-phoenix-app
```

```
- name: update build file
  run: |
    tar -xvf doc_artifact/artifact.tar --directory ${GITHUB_WORKSPACE}
    rm -r doc_artifact

- name: commit and push
  env:
    GITHUB_TOKEN: ${{ secrets.PHOENIX_ILD }}
  run: |
    git config --global user.email ${GITHUB_ACTOR_ID}+${GITHUB_ACTOR}@users.noreply.github.com
    git config --global user.name ${GITHUB_ACTOR}
    git add .
    git diff-index --quiet HEAD 2>&1 > /dev/null || git commit -m "Update new detector files for Phoenix-ILD" && git push
```

Figure 19: Example of github action code.

Comparison Between CED and Phoenix- Images

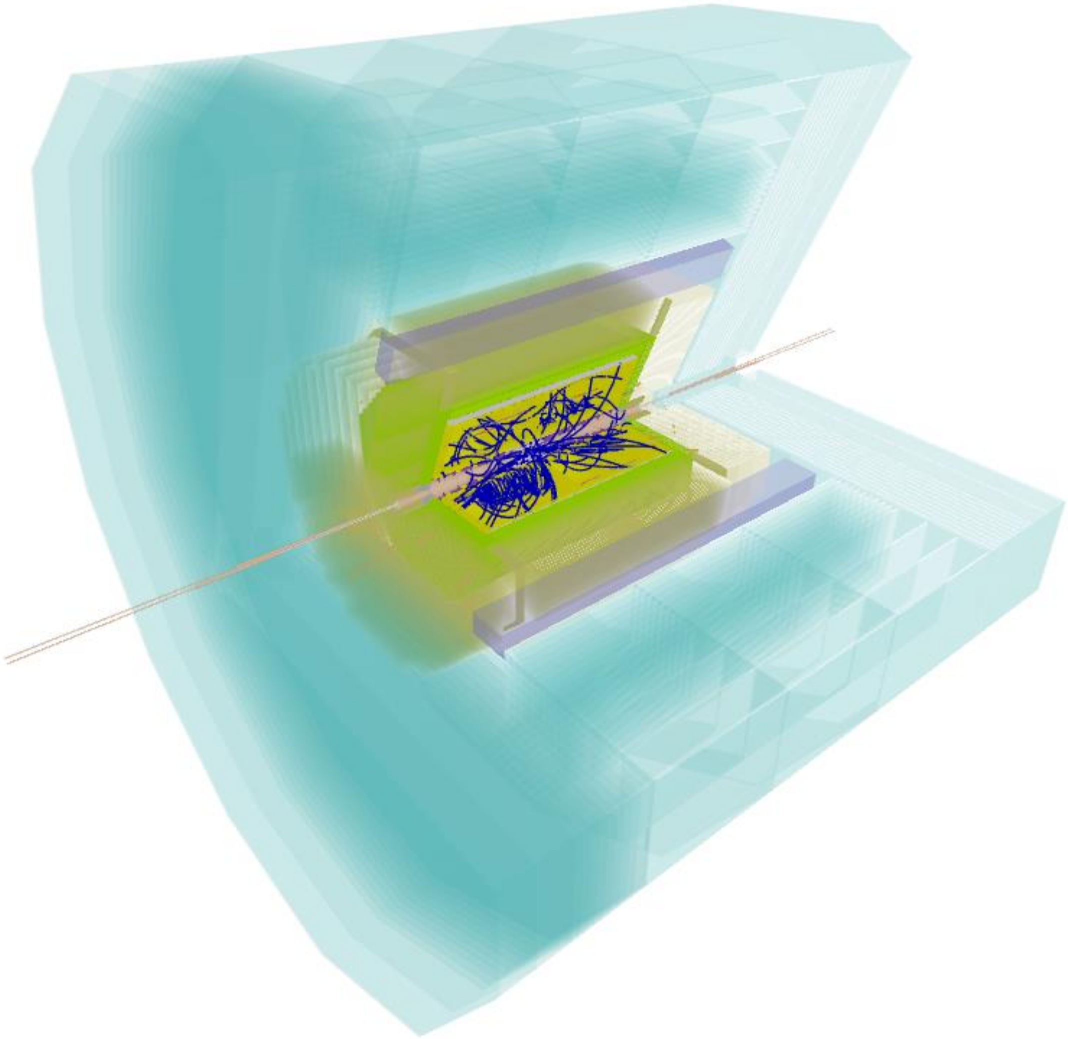


Figure 20: ILD in the Phoenix Event Display.

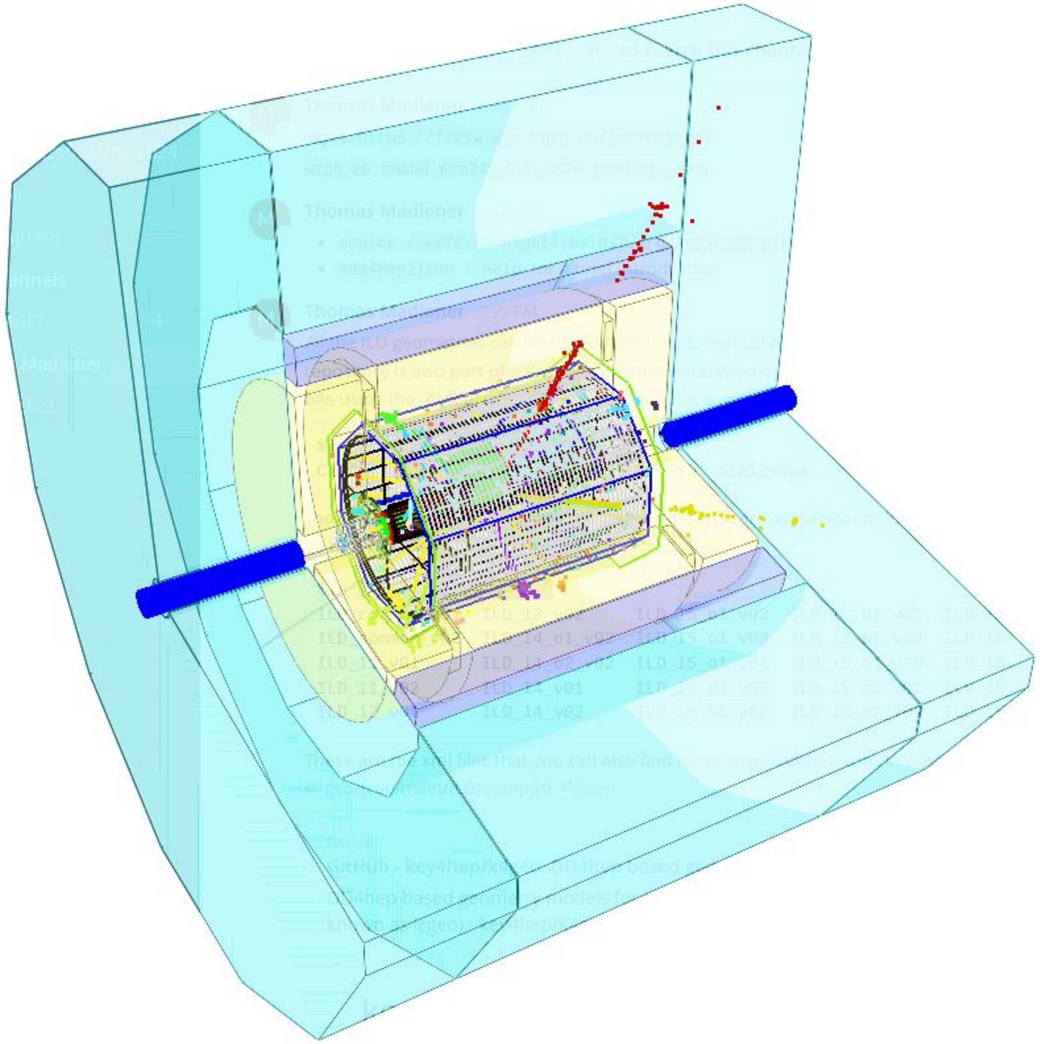


Figure 21: ILD in the CED.

Comparison Between CED and Phoenix

- Both Phoenix and the CED work very similarly.
- The CED event display however, had a few extra features compared to Phoenix:
  - Geometry fading at long distances.
  - Z-axis cuts.
  - Different views (side, front or fisheye view).
  - Background colours.
  - Keyboard shortcuts.
- The main difference was the way in which both could be used: Phoenix can be easily used in the browser whereas the CED needed more setting up time and the correct installations on the PC.

Conclusion

- Built the ILD Phoenix Event Display and deployed on github.
- Created extra scripts to help the usability of the Phoenix Event Display:
  - The XML to GlTF one-step conversion.
  - Automatic configuration file generator.
  - Adding a new detector and redeploying the app.

Thank you

Contact

Deutsches Elektronen-
Synchrotron DESY

www.desy.de

Abigail Harrison
FTX Group
abbyrharrison@gmail.com
Linkedin: www.linkedin.com/in/abby-harrison-79357918a