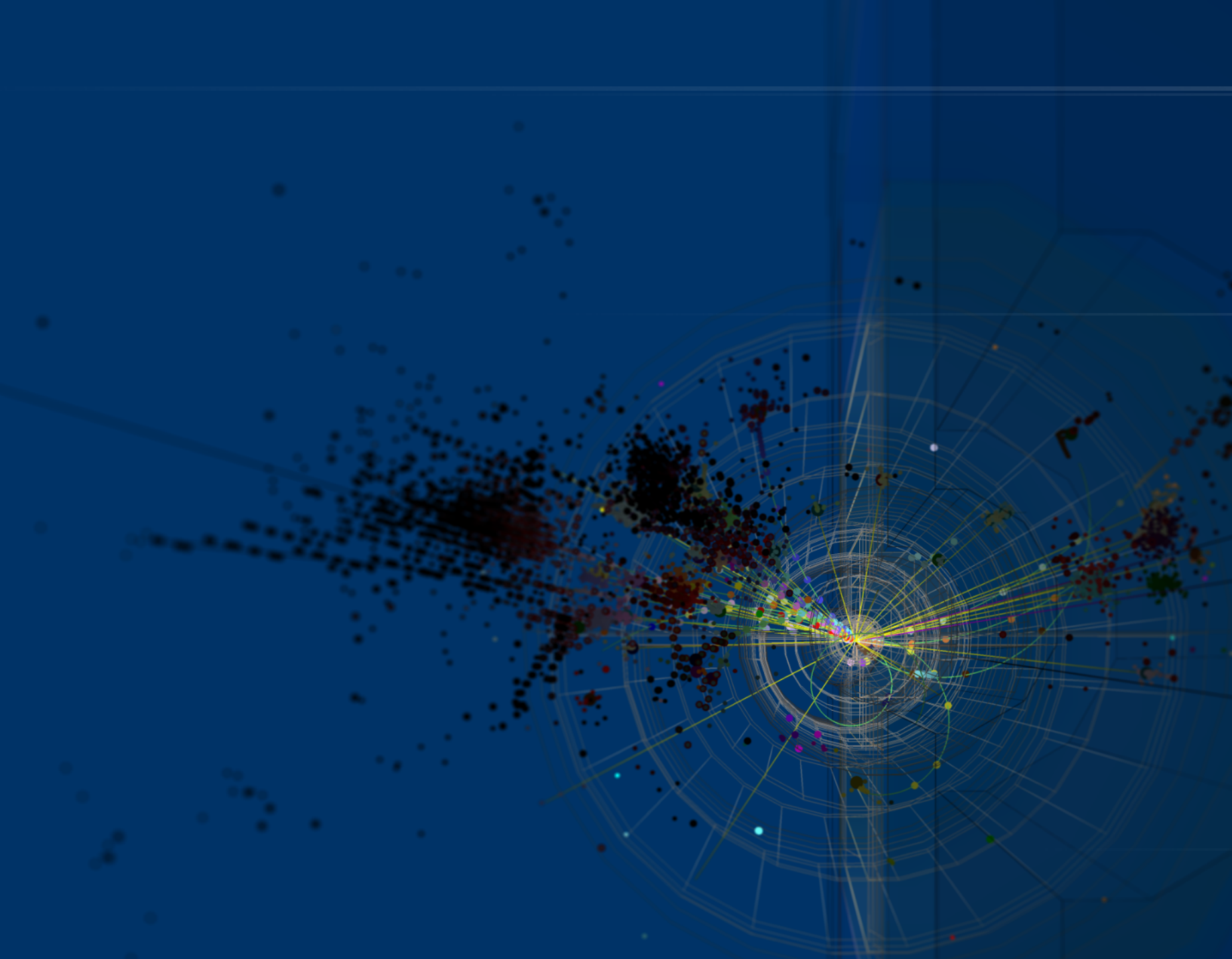


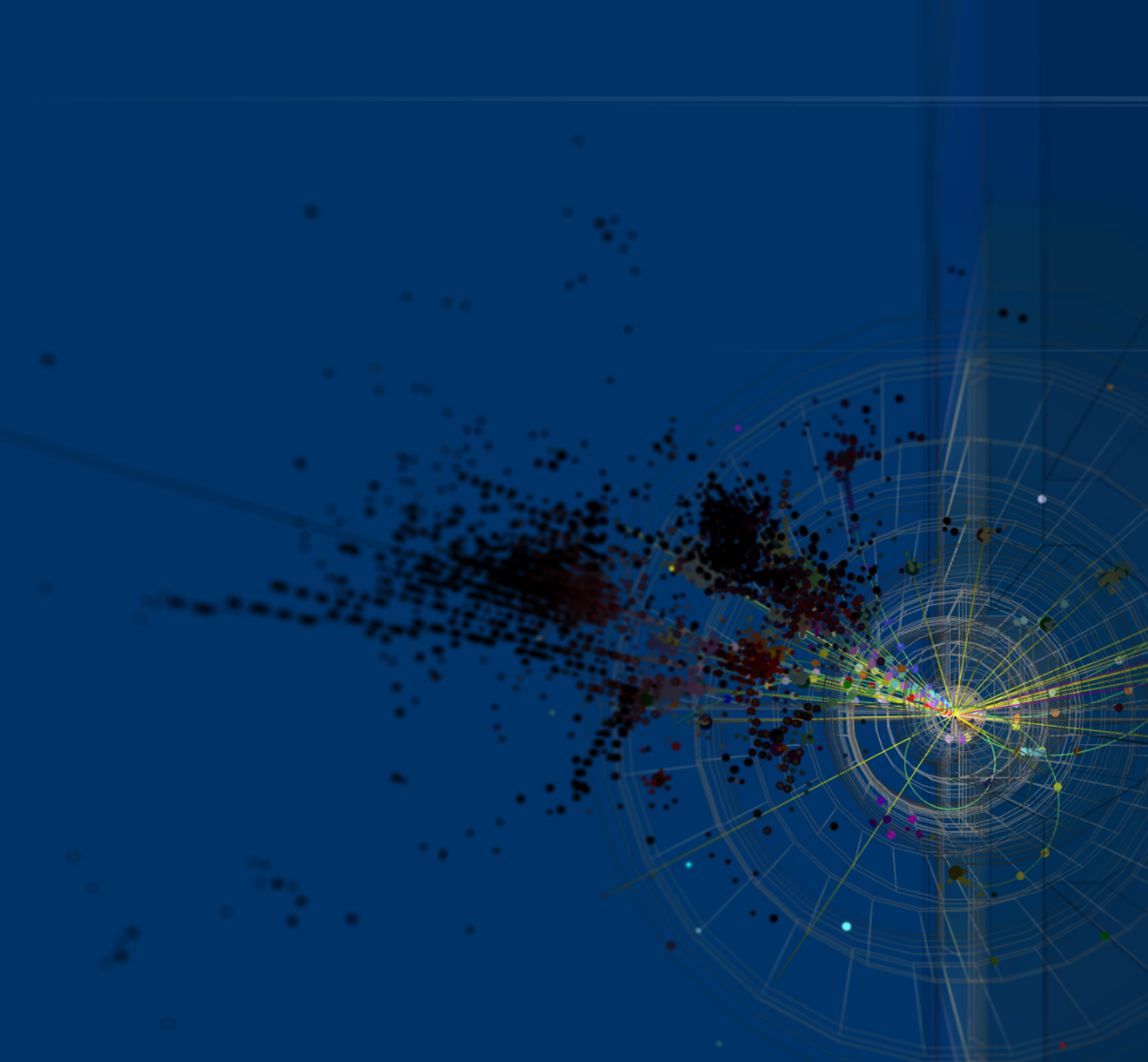
Revisiting Event Displays

Lawrence Lee

3 Aug 2023

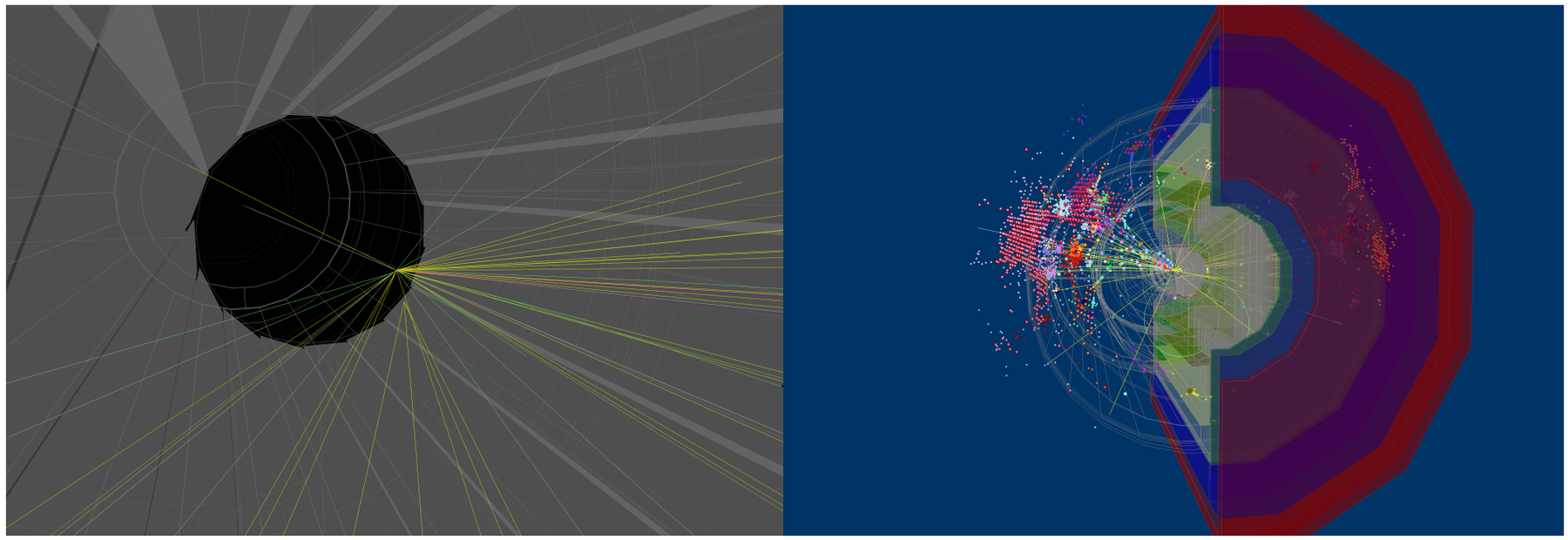
THE UNIVERSITY OF TENNESSEE KNOXVILLE

UTK CMS



- **ced2go** is the default event display program for our software stack (originally from ILC)
  - This background was made with ced2go (a few years ago)
- dd4hep also provides its own event display. **ddeve**
- We've discussed if it makes sense to develop an interface to **Phoenix**
  - (Web-based event display pioneered from the ATLAS side [Ed Moyse, et al]. Used a bit by LHCb as well.)
- Background: Federico was trying to get new magnet geometry to be reflected in ced2go. Wasn't working as expected even w/ some prodding.
- Mostly want to document the recipe for my studies here.

ced2go



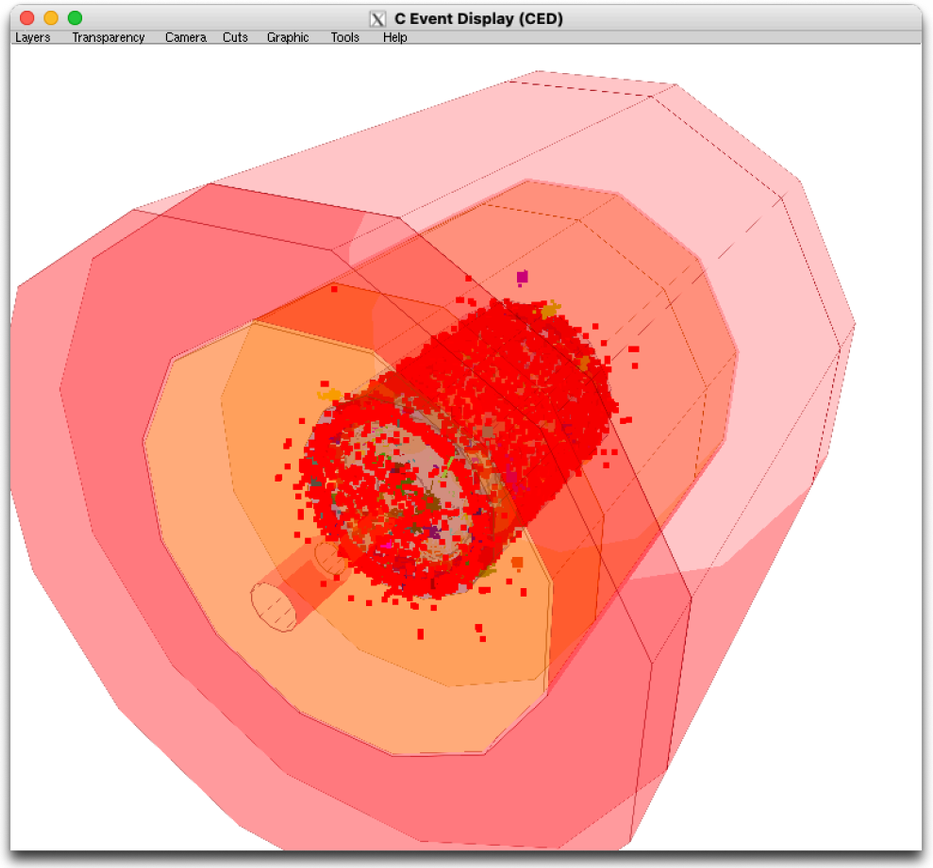
- Have been able to run CED event display from the docker image on my Mac (wasn't simple to figure out…)
  - Follow the instructions here using `socat` for establishing an X connection with your container
    - https://cntnr.io/running-guis-with-docker-on-mac-os-x-a14df6a76efc
    - This should allow you to open a TBrowser/xeyes/whatever
  - But that doesn't enable openGL over that connection yet. So you have to run
    - `> defaults write org.macosforge.xquartz.X11 enable_iglx -bool true`
    - And reboot.
    - Taken from http://atlas-vp1.web.cern.ch/atlas-vp1/home/documentation/how-to-run-vp1/tutorial/0-pre-requisites/#Mac_OS_X

- A few years ago, I put together a recipe for running ced2go from containers running the µC docker images (on macs — which is medium tricky)
  - Some of these things have changed like the location of the enable_iglx parameter.
  - **But this doesn't work on Apple Silicon.** There's a fundamental error in qemu (the layer that emulates x86 on arm) that Apple would need to fix. The OpenGL forwarding still works (glxgears runs) but ced2go fails to show actual geometry. (See https://github.com/docker/for-mac/issues/5123)
- I resorted to running ced2go on docker on local linux (on x86) machine.

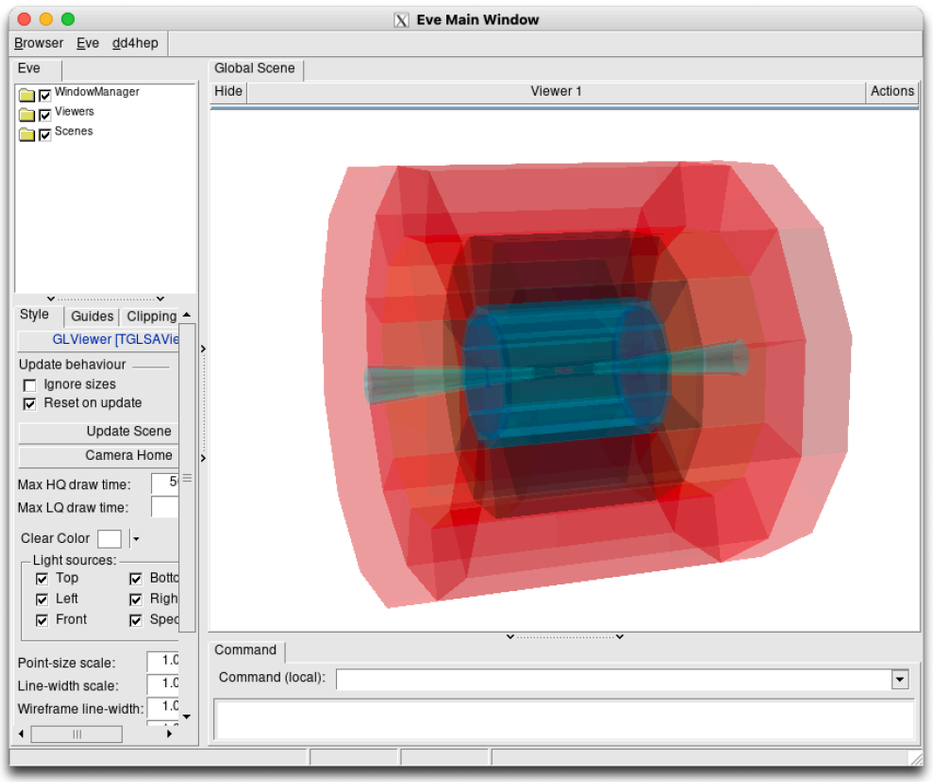
ced2go/ddeve

- Running ced2go on x86 docker host (ssh'ing in from my M1 Mac w/ X forwarding)
  - Starting docker image with display forwarded over ssh connection.
    - `SOCK=/tmp/.X11-unix; XAUTH=/tmp/.docker.xauth; xauth nlist $DISPLAY | sed -e 's/^..../ffff/' | xauth -f $XAUTH nmerge -; chmod 777 $XAUTH;`

      `docker run --rm -ti -v $PWD:$PWD -w $PWD -e DISPLAY=$DISPLAY -v $XSOCK:$XSOCK -v $XAUTH:$XAUTH -e XAUTHORITY=$XAUTH --net host infnpd/mucoll-ilc-framework:1.7-almalinux9`
  - Then just running ced2go handing it an slcio input and the top level geo XML
    - `export USER=root;`
      `ced2go -d detector-simulation/geometries/MuColl_10TeV_v0A/ MuColl_10TeV_v0A.xml gen_muonGun_reco_390.slcio`



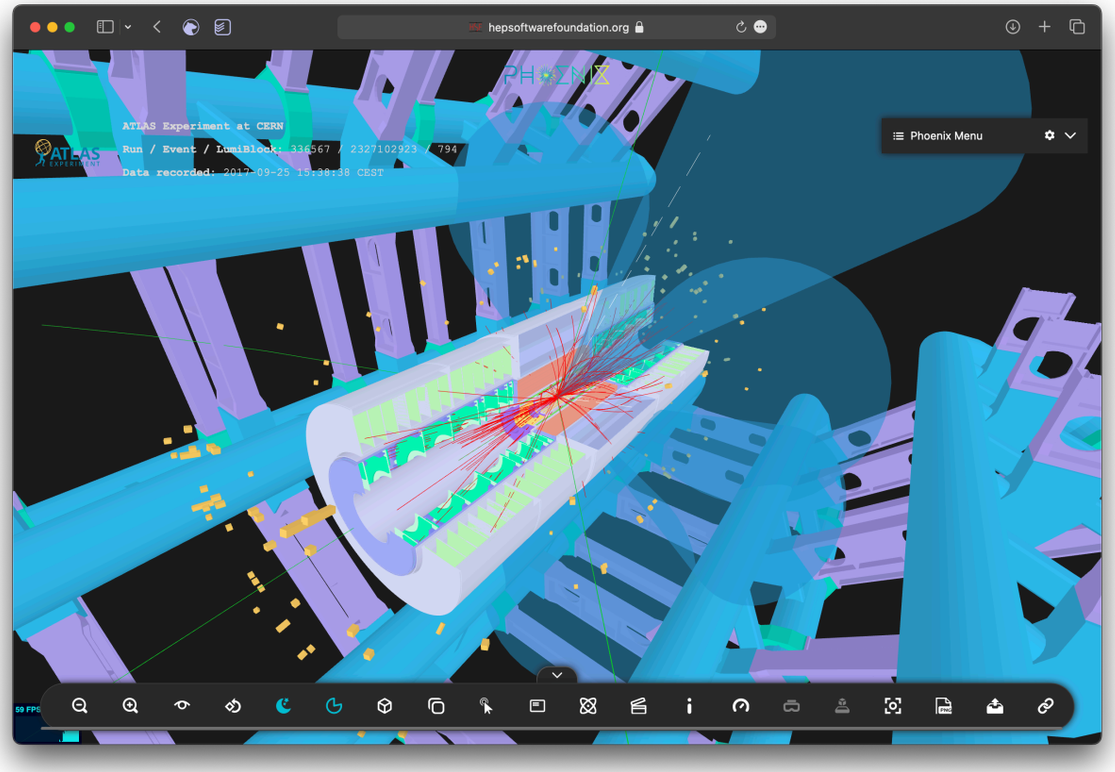
- But ran into same problem as Federico. ced2go only seems able to draw the envelope of each detector and not individual layer info.
- Switched to ddeve which is available in the same container. Same problem.
  - `ddeve -config detector-simulation/geometries/MuColl_10TeV_v0A/ MuColl_10TeV_v0A.xml`

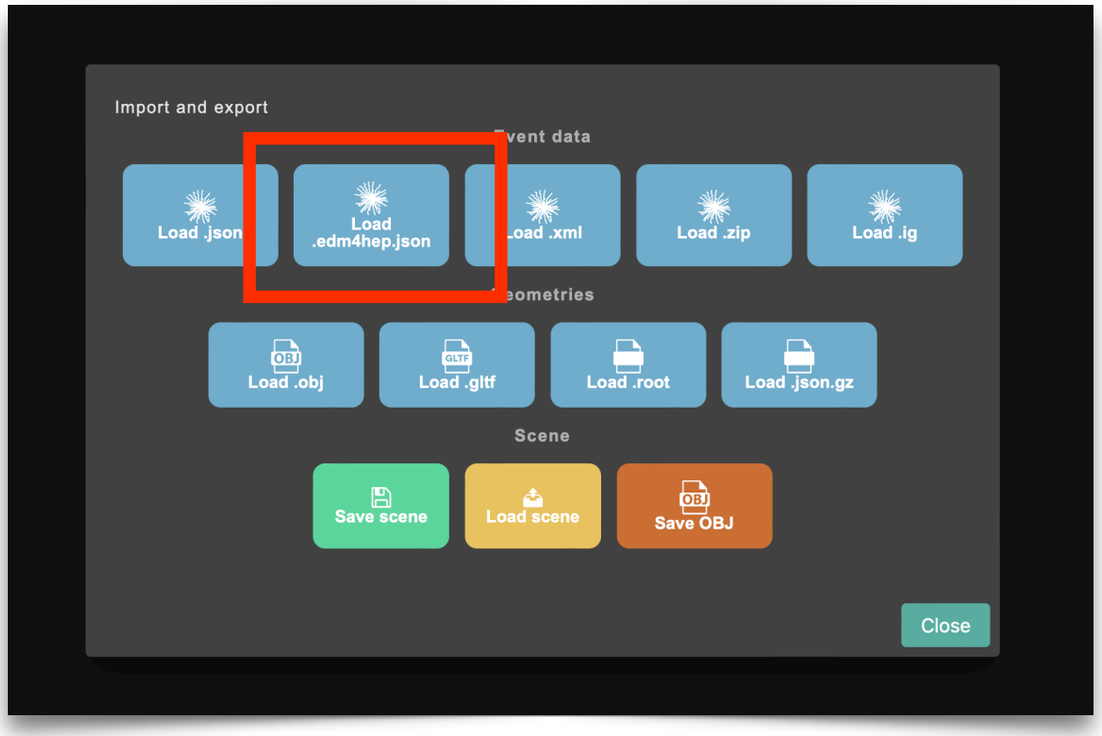


Phoenix

- Playing with Phoenix.
  - Meant to be flexible. Multiple geo input formats. Multiple event data formats.
  - Runs in browser. No need to install anything. Can just go to https://hepsoftwarefoundation.org/phoenix/#/atlas and play around, upload new inputs, whatever.
    - If we get something we like, we can even ask them nicely if they can put it on their main page (and can feedback tools to the ee community)
- Event data input. EDM4HEP supports JSON output that is now supported by Phoenix (only on in some of the examples, but it exists)
  - So should be (hopefully) straightforward to get our event info in!
- The slightly trickier part is the geometry

ATLAS Example

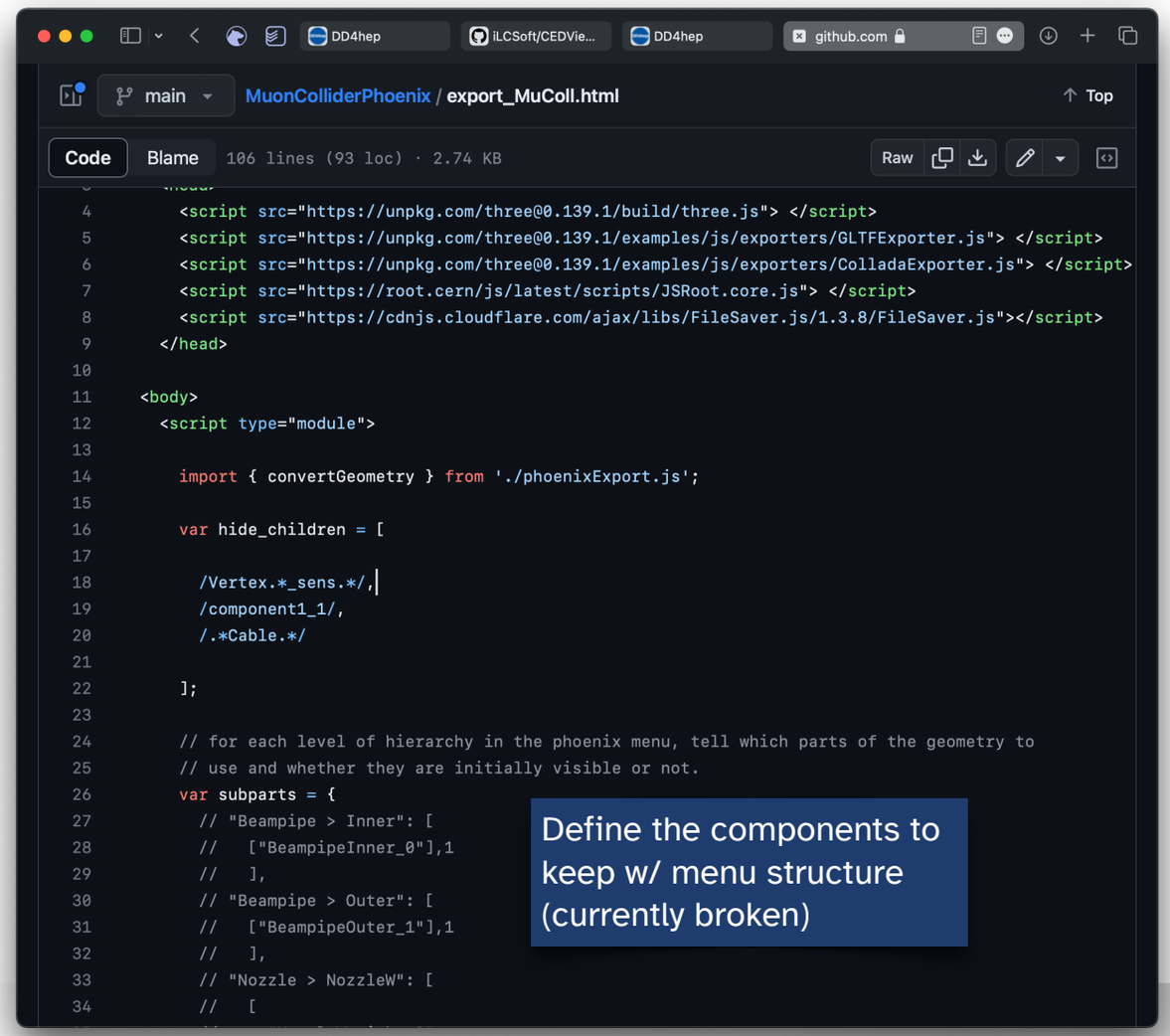




Edm4hep interface exists since latest version (May, v2.14.1)

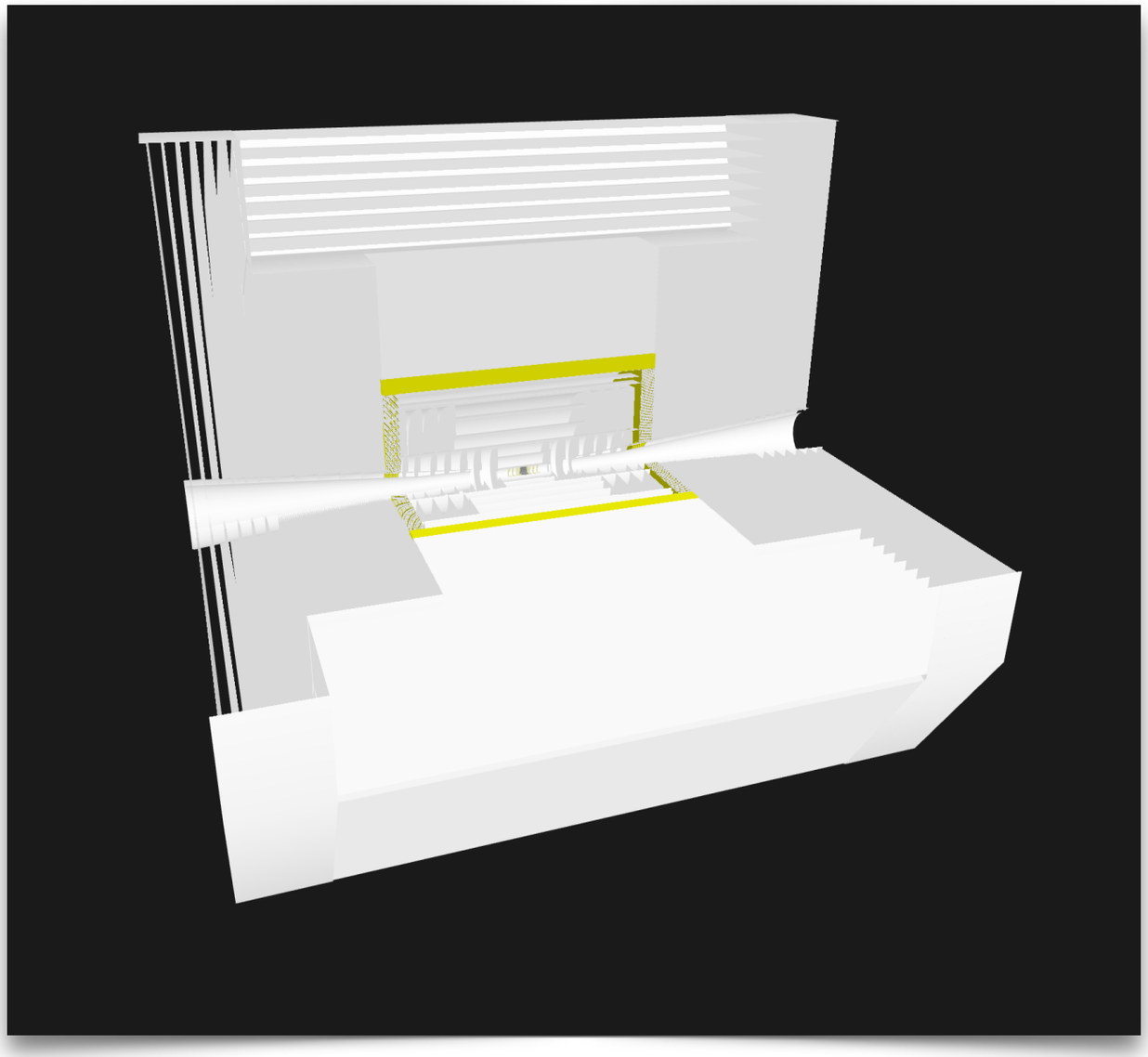
Phoenix

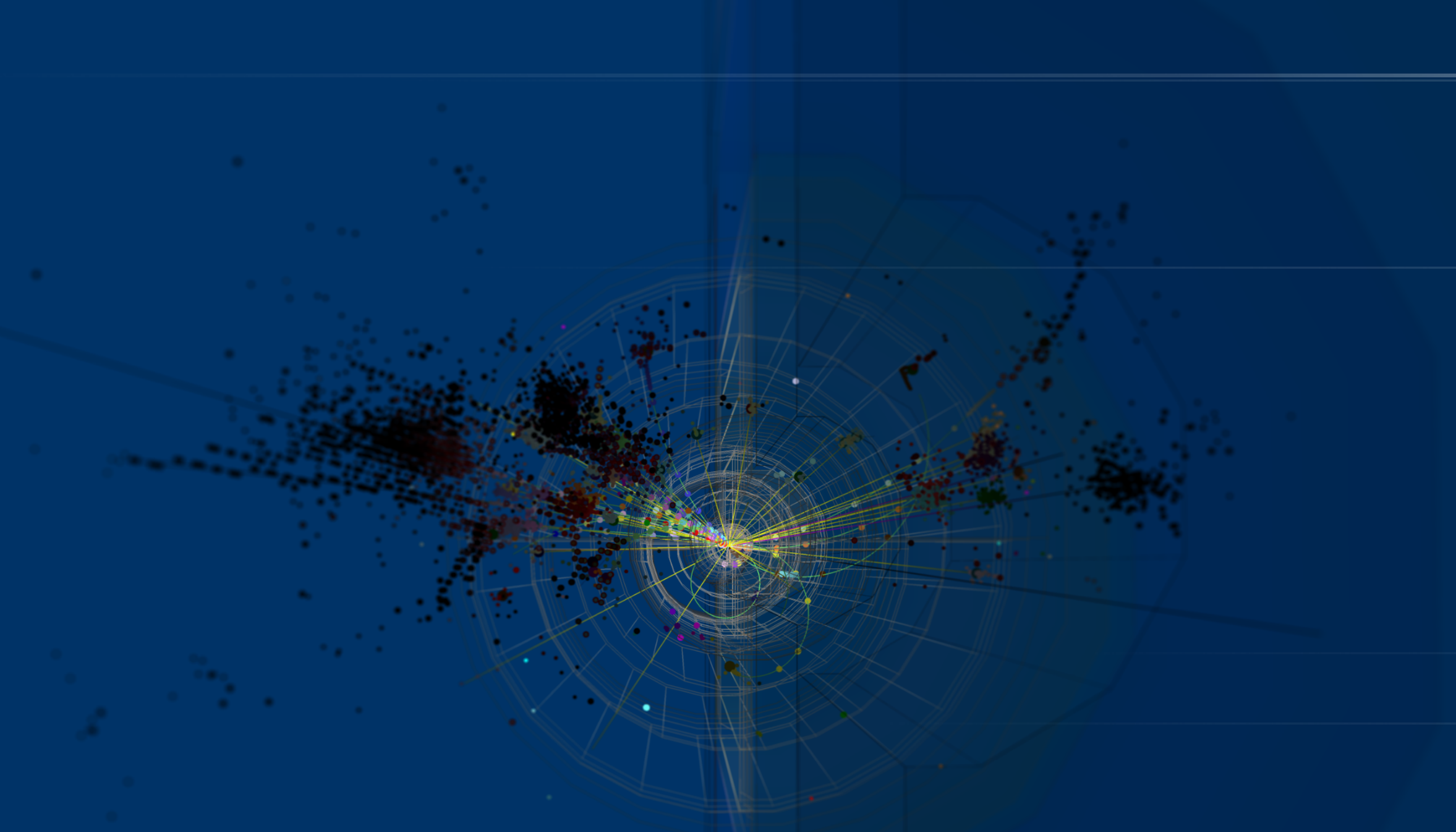
- Phoenix can read in many geo formats. I'm playing w/:
  - XML -> ROOT TGeo -> GLTF
- First stage is straightforward. dd4hep provides a tool:
  - `geoConverter -compact2tgeo -input detector-simulation/geometries/MuColl_10TeV_v0A/MuColl_10TeV_v0A.xml -output MuColl_10TeV_v0A.root`
- Second stage more difficult but Phoenix has some tools and examples documented here
  - Uses js converter (using THREE.js tools). Runs in browser
  - Make an HTML file that picks which TGeo objects you want to convert and how to structure them.
- https://github.com/lawrenceleejr/MuonColliderPhoenix
- A little buggy still and needed some changes to the js
  - Hoping to rewrite this converter since it's a little contrived and doesn't respect visibility or color properties from the input
  - (If somebody with more time is interested in working on this, I'm happy to support)



Phoenix

- Running this chain, I get a GLTF that I can import and play with.
- Issues to be worked on:
  - Length units need to be defined or converted
    - (Our detector comes out super tiny w.r.t. ATLAS. Probably a cm vs mm issue.)
  - *Some* of the XML/TGeo visibility/color properties don't seem to be respected
    - But I think I see why. The rewrite I want to do to the TGeo -> GLTF converter may fix this.
  - Runs a bit slow on my desktop with this level of detail. Need to figure out what detail level we want by default.
  - Converter works on TNamed names. But many of the names are the same (e.g. ∃ "slice0_0" objects in every bit of the muon and calo systems)
    - Either need to figure out how to differentiate these in the js, or give more unique names to objects in the XML
  - Probably other stuff I forgot...
- Because of the nature of this processing step, there is, in principle, complete customization possible.
  - (Just might require a little elbow grease.)





That’s all.