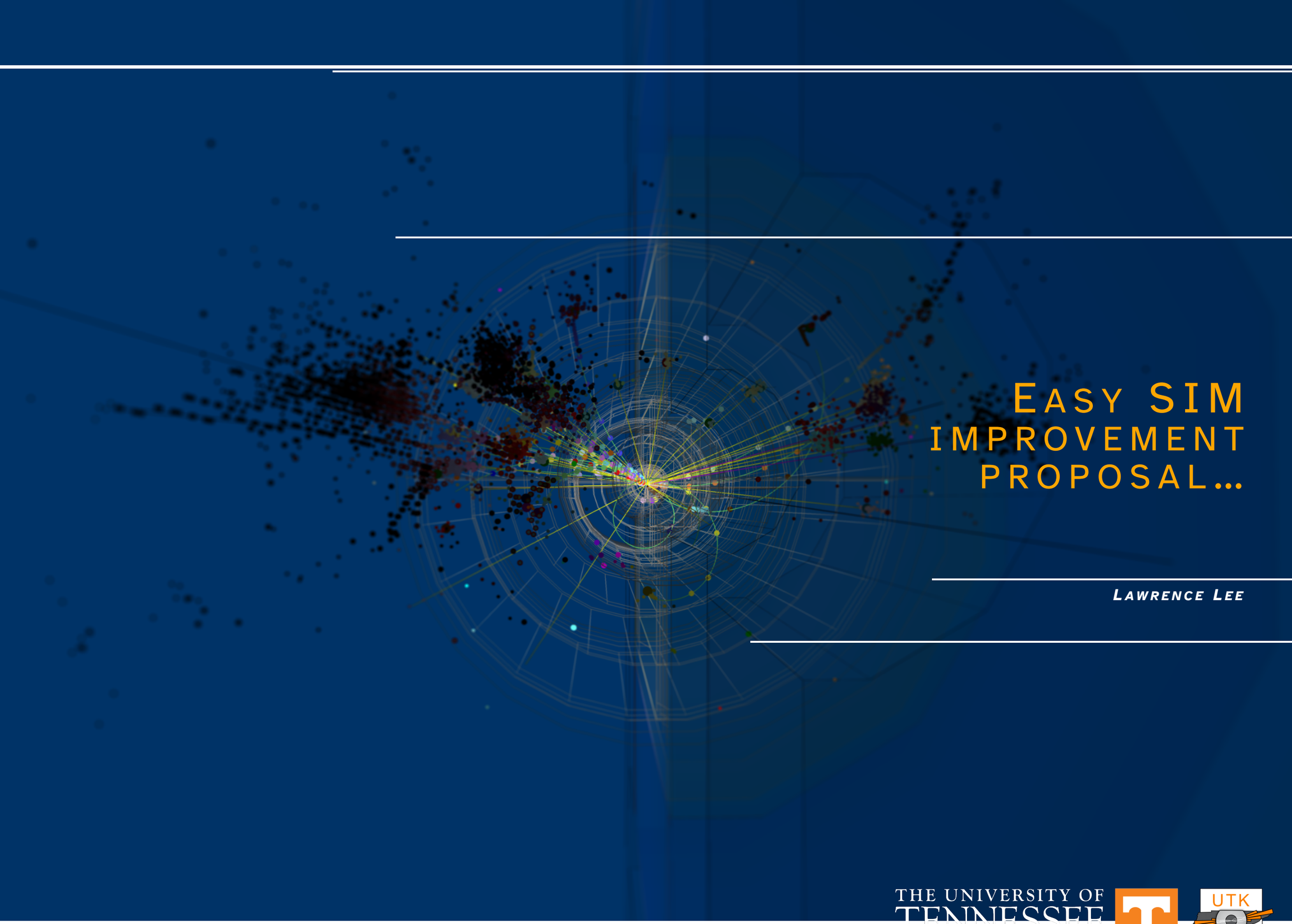


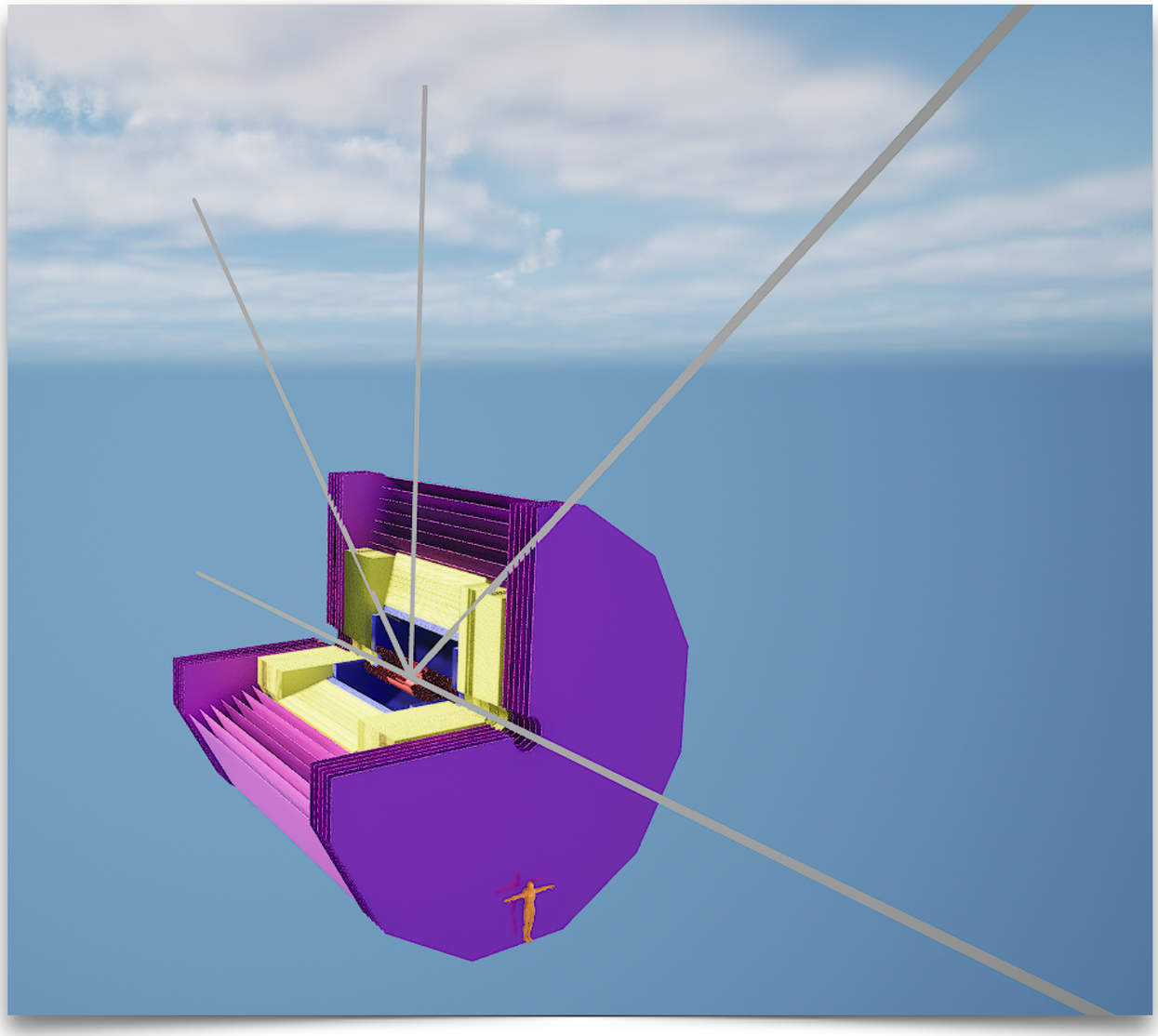
Easy SIM Improvement Proposal...

Lawrence Lee

THE UNIVERSITY OF TENNESSEE KNOXVILLE

UTK CMS

Aside: Visualization



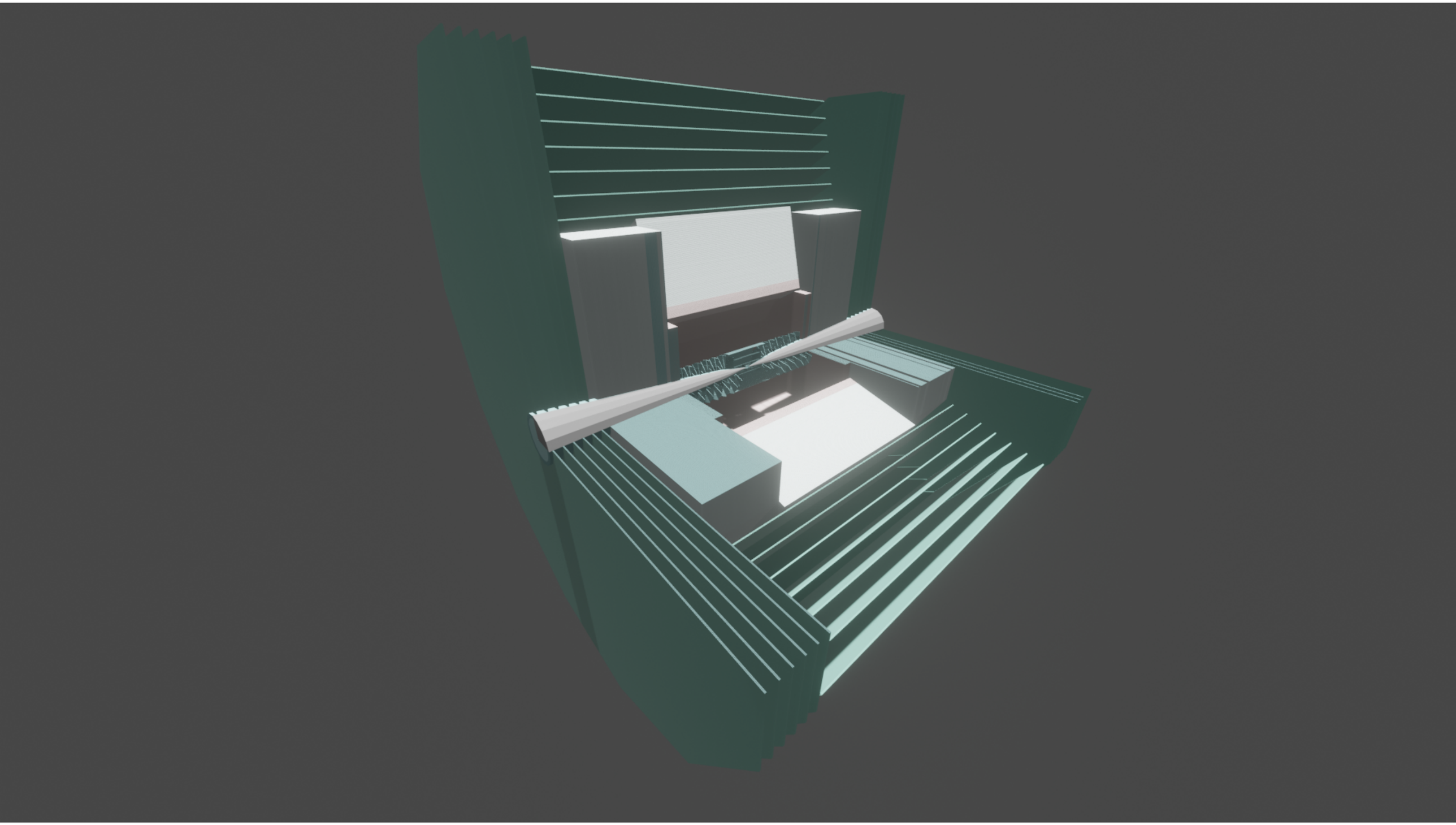
Charles also is figuring out how to add these annotations in Unreal Engine

Able to hack together a system that we can present on later.

Compact XML -> GDML -> Hacked pyG4ometry Scripts -> GLTF
-> Blender for phi cut away -> Unreal Engine

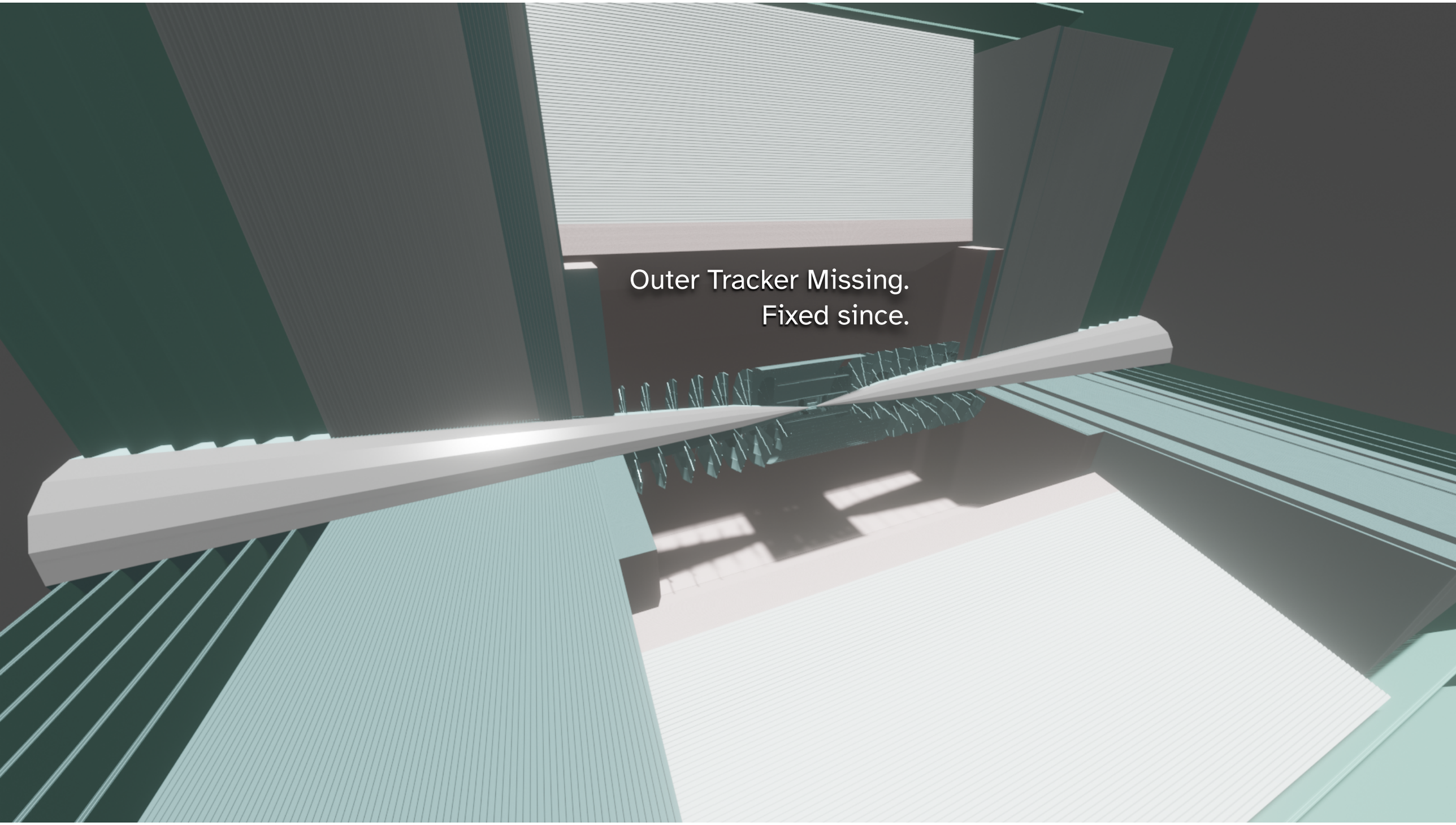
Aside: Visualization

Renderings direct from Blender



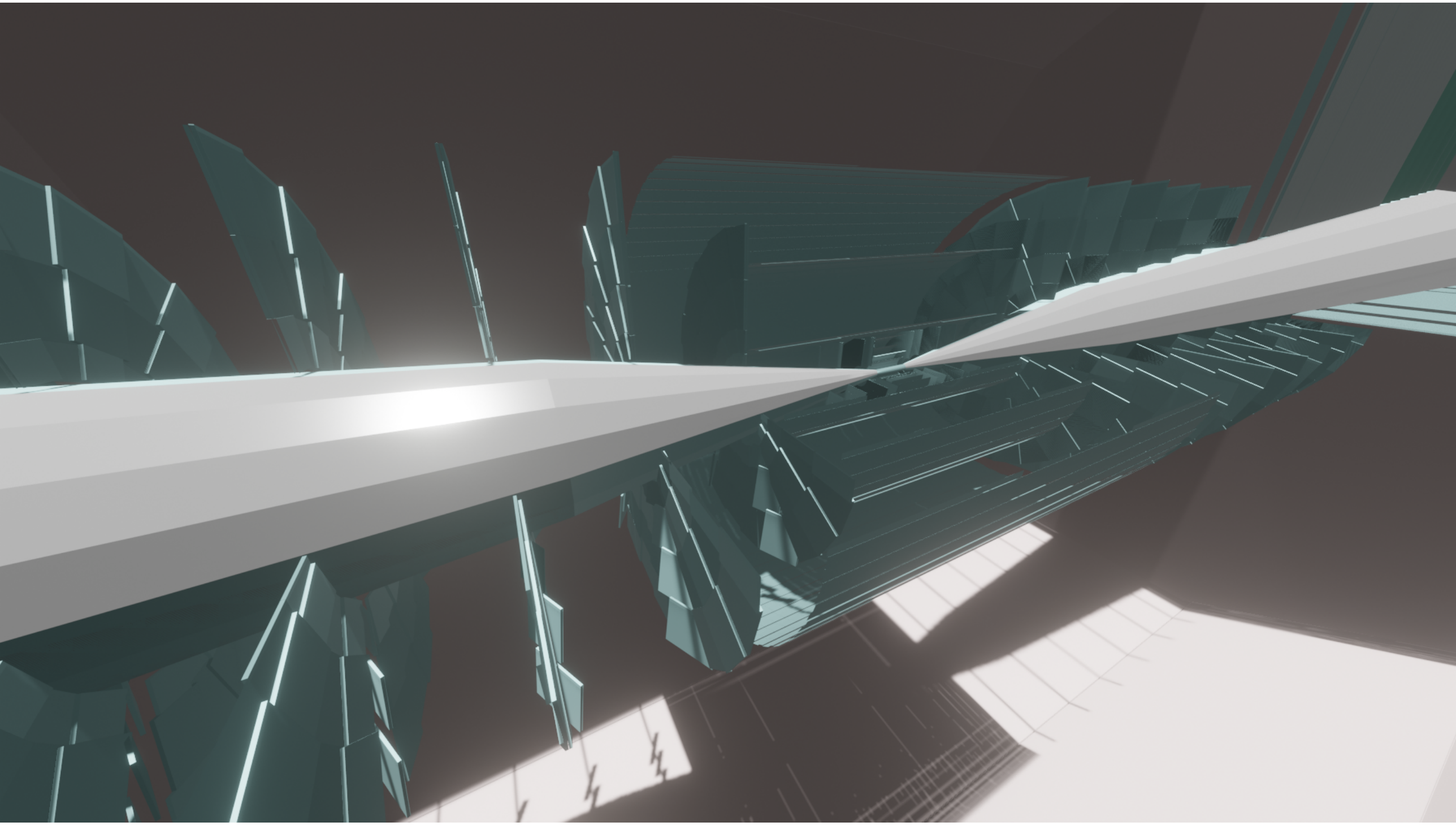
Aside: Visualization

Renderings direct from Blender



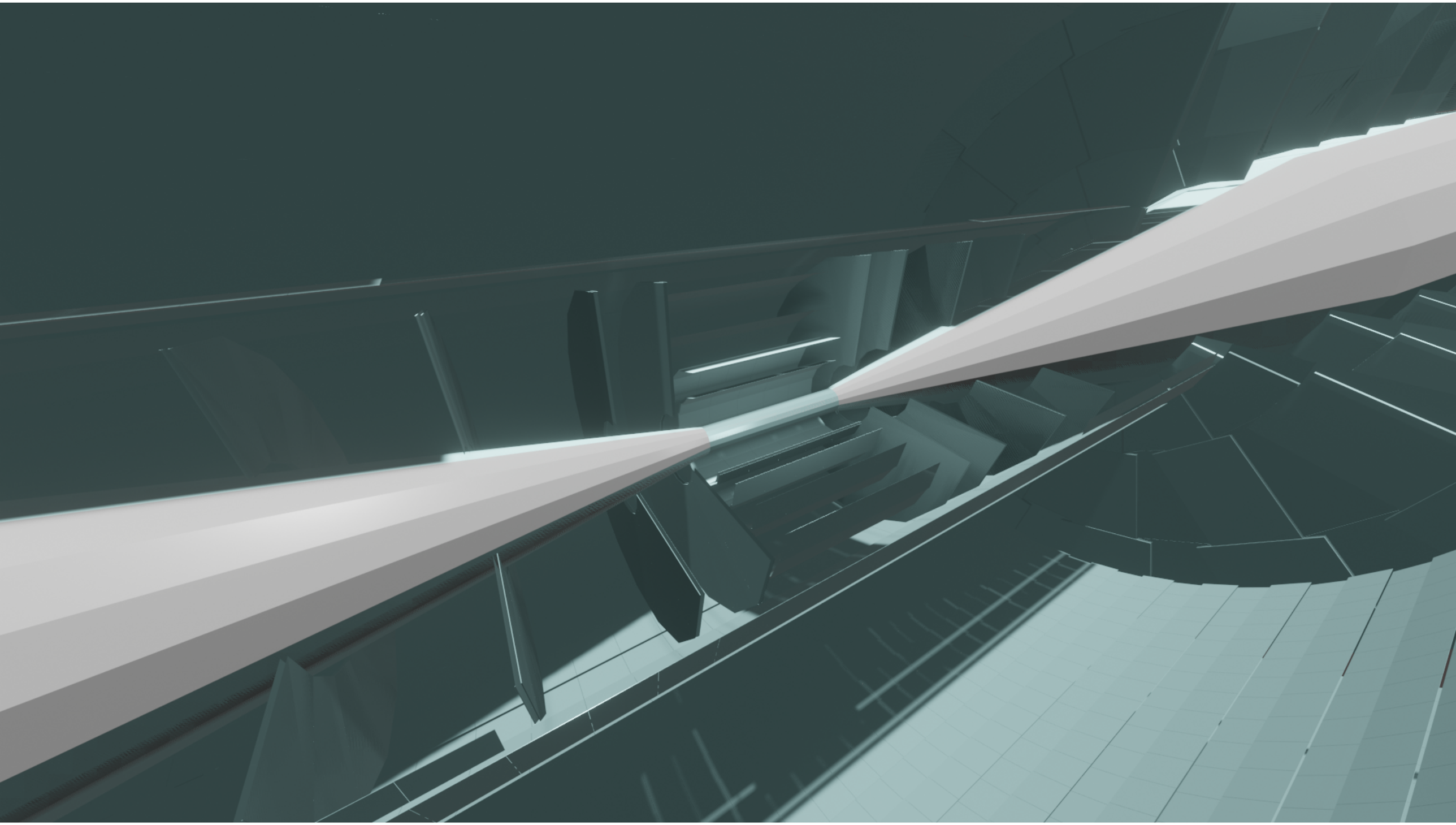
Aside: Visualization

Renderings direct from Blender



Aside: Visualization

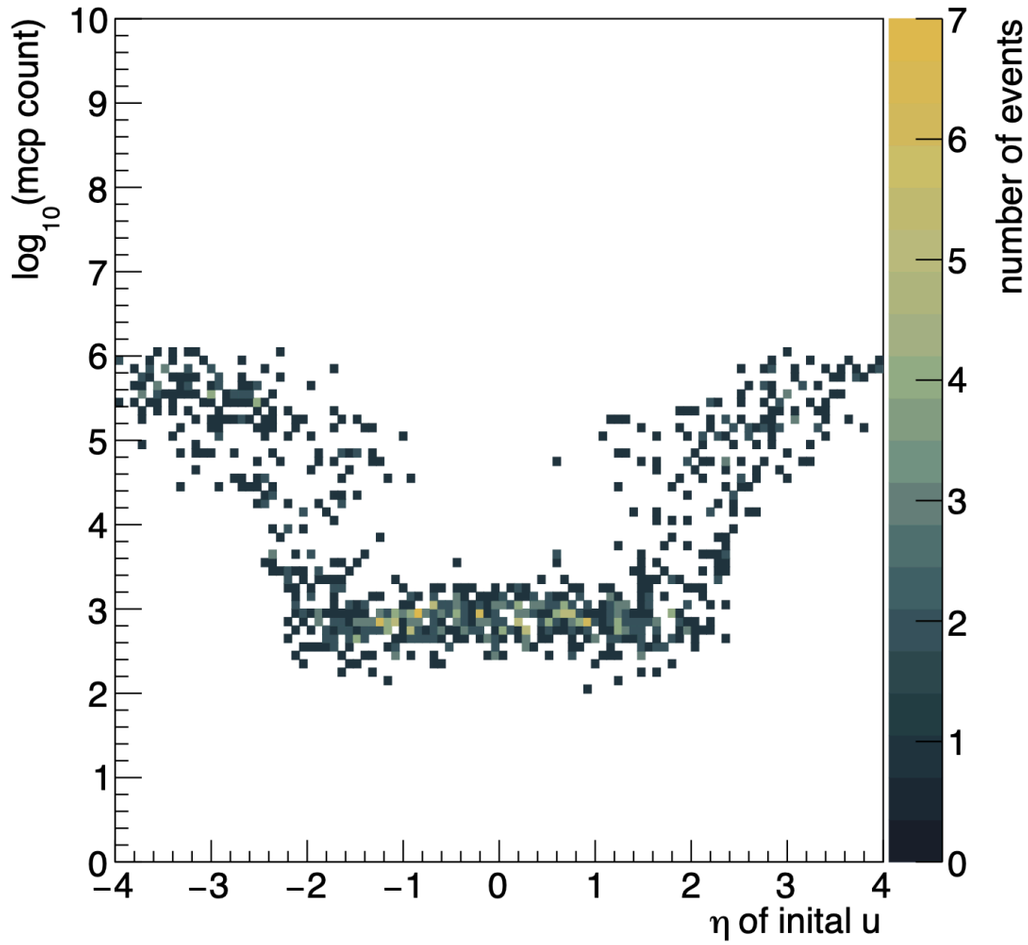
Renderings direct from Blender



Location of simulated particles

- n.b. Since this is about SIM, “Tracking” in these slides means G4 particle propagation. (Not to be confused with charged particle tracking.)
- Ben last month: When a jet is forward, the number of post-SIM particles is increased by 3 orders of magnitude
  - Question is — are we burning a LOT of CPU (fossil fuels 😔🌎) on G4 particle interactions with the nozzle?
- If that’s the cause of this, can we change G4 to stop tracking in the nozzles?
- Two paths:
  - XML Geometry Description
    - Just artificially change the material of the nozzle in the geometry handed to G4
  - Create a custom stepping action class in G4
    - Would be great but would require some more development and playing

Slides from Ben Johnson [Aug 31]



G4 Geometry Hacking

- The 10 TeV Detector description XML (in the “compact” form from dd)
  - Can change nozzle material from Tungsten to custom material with tiny nuclear interaction length (10cm -> micron or something) so that particles come to a stop quickly and stop being [G4-]“tracked”
  - Or change the density? A number of handles to play with. NIL seems like a great start.

- Would only be done on the XML handed to G4 so it won’t affect the BIB simulation in any way
- Doesn’t need any new code, can implement something today.

Nozzle_10deg_v0.xml

```
14        <detectors>
15            <comment>Nozzle</comment>
16
17            <detector name="NozzleW_right" type="DD4hep_PolyconeSupport" vis="NozzleWVis"
18                <comment>Internal part of the nozzle: Tungsten</comment>
19                <material name="Tungsten"/>
20                <zplane z="Nozzle_zmin" rmin="1*cm" rmax="1*cm" />
21                <zplane z="15*cm" rmin="0.6*cm" rmax="2.59223*cm" />
22                <zplane z="Nozzle_kink_z" rmin="0.3*cm" rmax="17.63*cm" />
23                <zplane z="Nozzle_kink_z" rmin="0.3*cm" rmax="15*cm" />
24                <zplane z="200*cm" rmin="0.596*cm" rmax="17*cm" />
25                <zplane z="600*cm" rmin="1.78*cm" rmax="50*cm" />
26            </detector>
```

```
112       <material name="TungstenDens24">
113           <D value="17.8" unit="g/cm3"/>
114           <fraction n="0.93" ref="W"/>
115           <fraction n="0.061" ref="Ni"/>
116           <fraction n="0.009" ref="Fe"/>
117       </material>
118
119       <material formula="W" name="Tungsten" state="solid" >
120           <RL type="X0" unit="cm" value="0.350418" />
121           <NIL type="lambda" unit="cm" value="10.3057" />
122           <D type="density" unit="g/cm3" value="19.3" />
123           <composite n="1" ref="W" />
124       </material>
```

materials.xml

Custom Stepping Actions in G4

- G4 doesn’t have a “BLACKHOLE” (perfect absorber) material the way FLUKA does
- But in dd, we can assign custom stepping actions to particular volumes
  - Can set SIM.action.mapActions[’NozzleW_right’] = CustomNozzleAction
  - Where we create CustomNozzleAction of class G4UserSteppingAction
    - Dd4hep has a plugin structure that could make this not so bad
    - This we could customize however we wanted. We could allow normal simulation into a particular depth into the nozzle, and then kill the tracking. Could be cool to not affect reflections, e.g. but this skin depth would have to be tuned.
      - (For the custom material approach, this could be tuned similarly.)
- This is just a little less trivial to figure out, but probably not more than a post-doc-week or so.

SIM Steering File

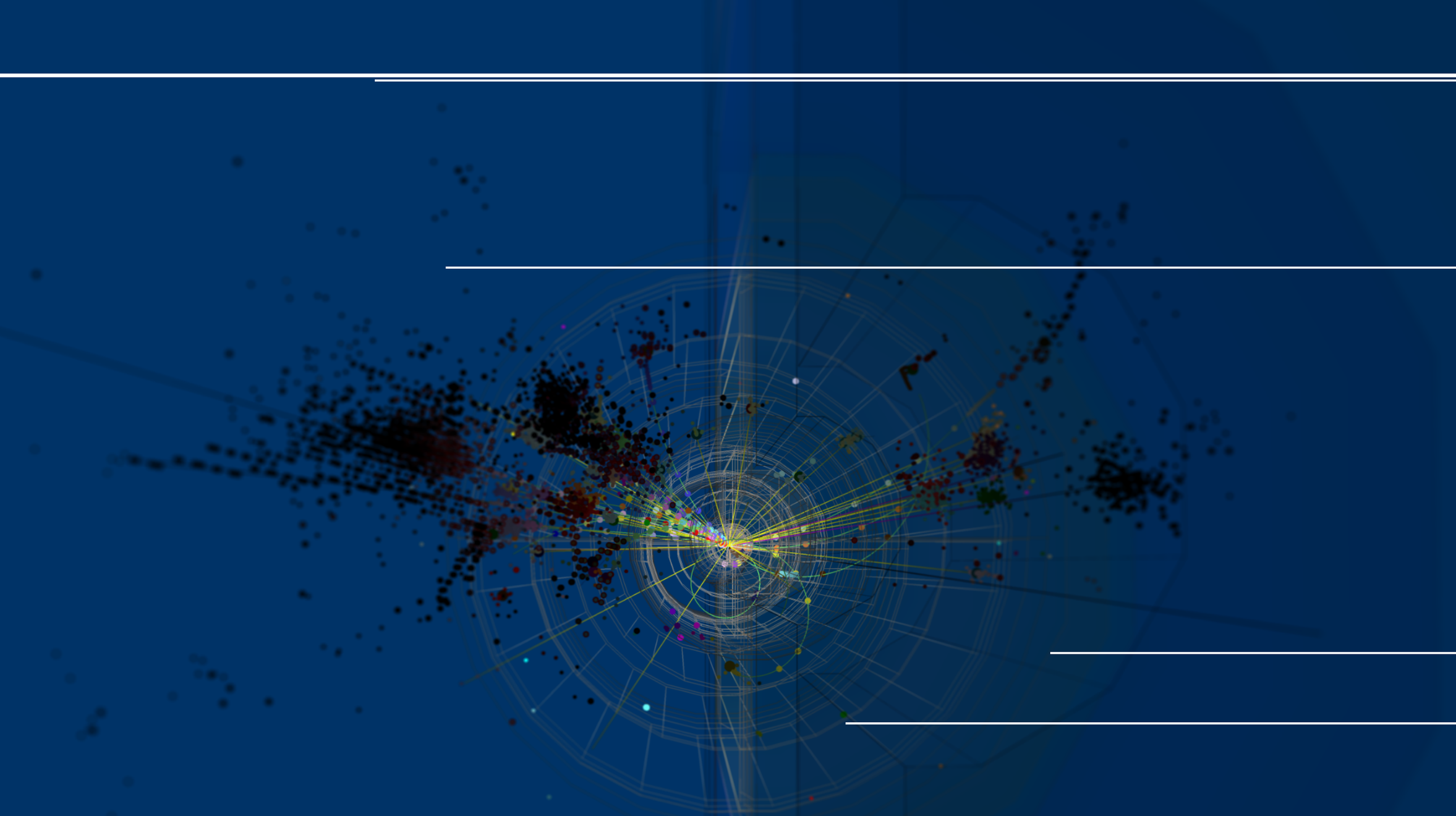
```
43    ##############################################################################
44    ## Action holding sensitive detector actions
45    ##   The default tracker and calorimeter actions can be set with
46    ##
47    ##   >>> SIM = DD4hepSimulation()
48    ##   >>> SIM.action.tracker = "Geant4TrackerAction"
49    ##   >>> SIM.action.calo    = "Geant4CalorimeterAction"
50    ##
51    ##   for specific subdetectors specific sensitive detectors can be set based on pattern matching
52    ##
53    ##   >>> SIM = DD4hepSimulation()
54    ##   >>> SIM.action.mapActions['tpc'] = "TPCSDAction"
```

Proposal

- Try handing SIM step a new geometry XML where the nozzles (potentially just the tungsten portions?) are made of a custom material with a tiny nuclear interaction length
- Study the distribution of detector digits throughout the detector with and without this change
  - If the distribution is affected, play with the new material parameters distributions are consistent

- If we're unable to get them to match, then might need to build a custom stepper action

- **Will hopefully give significant speed up in SIM step and smaller output files (also 😍🌎)**



That’s all.