ACTS hackathon recap (Key4hep perspective)

Key4hep Tracking meeting

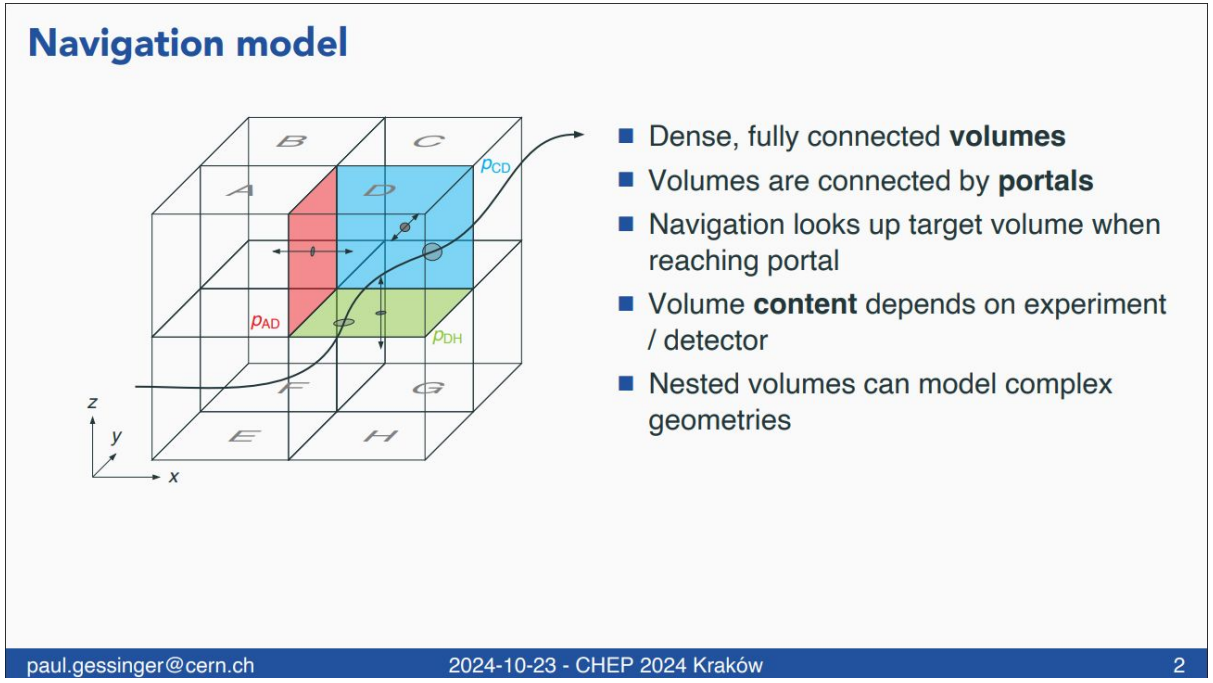
Thomas Madlener
Dec 12, 2025

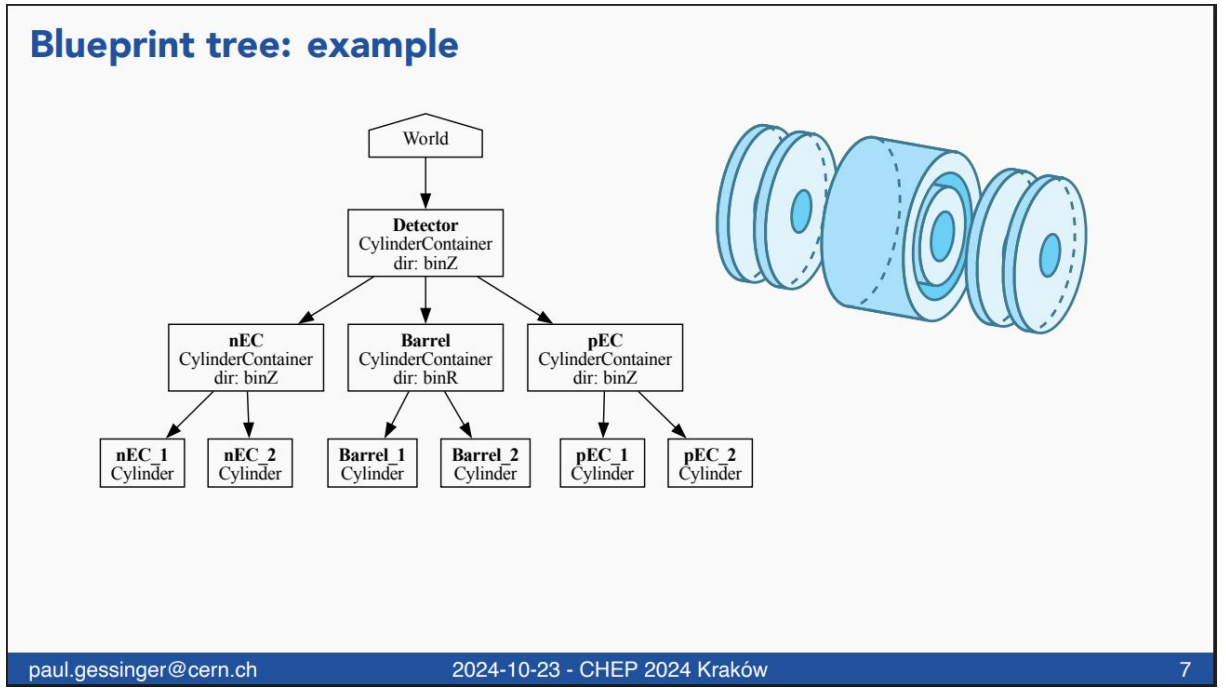
HELMHOLTZ

DESY.

(Very) quick Acts Gen3 Geo intro

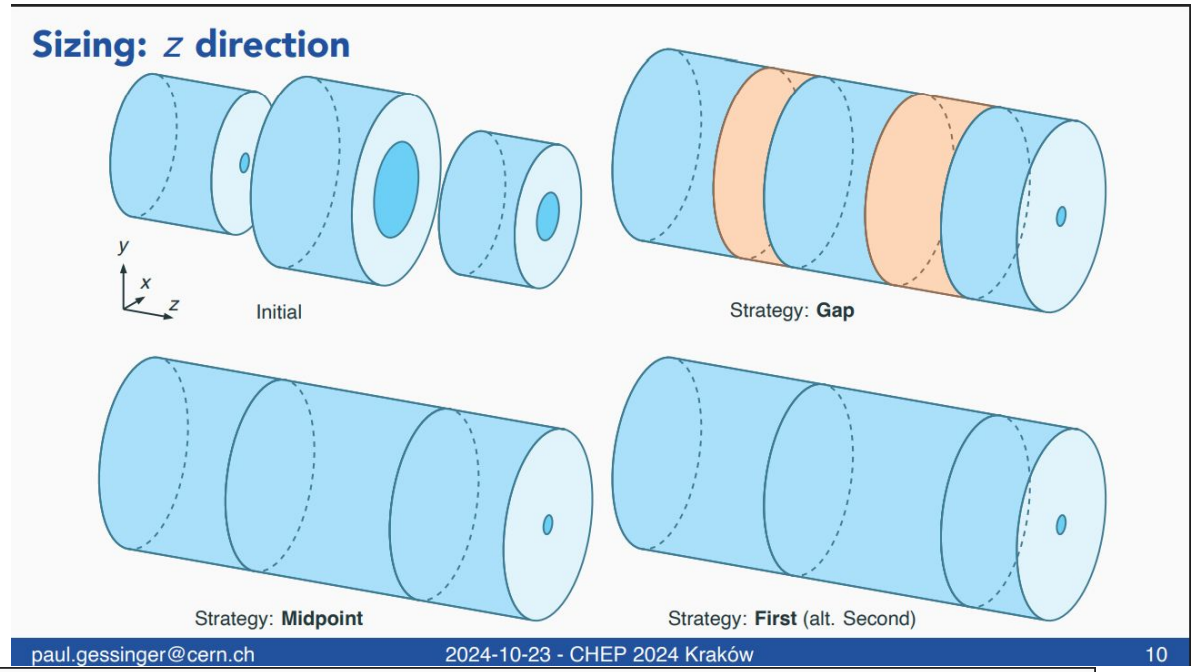
- See P. Gessingers talks at CHEP 2024 and at ACTS4NP 2025 workshop for more details
- Core objects
  - **Surfaces, Volumes, Portals**
- Central concept for geometry construction: **Blueprint tree**
  - Nodes can be subsystems, tracker layers, etc.
  - Build phases are propagated down, nodes consider children into consideration
- 3-phase construction
  - **Build**: Construct volume representation, compute sizing
  - **Connect**: Create and connect portals at volume boundaries
  - **Finalize**: Register portals with volumes, create acceleration structures





(Very) quick Acts Gen3 Geo intro

- Volume stacks
  - Arrangement of volumes along (local) axes
  - Cylinder volumes: z and r direction
  - Configurable **attachment** and **resize strategies**
  - Only expansion



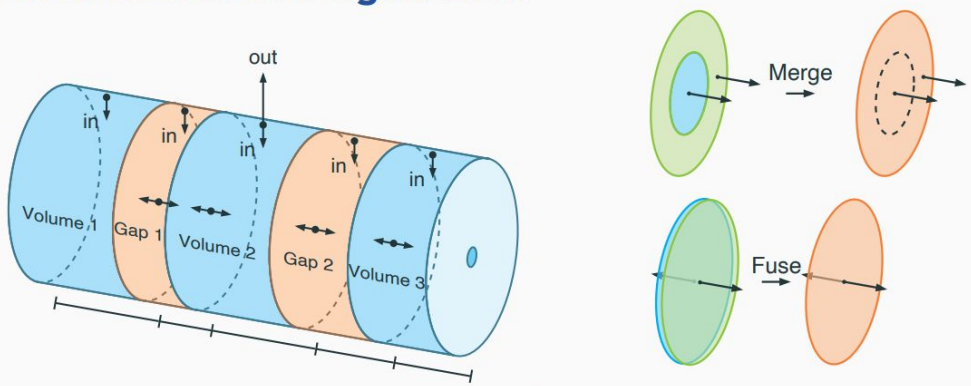
Building the blueprint tree ii

- Alternative: **convenience functions** on `BlueprintNode` for common node types
- Add corresponding node as child and return a reference

```
// Pre-constructed volume
StaticBlueprintNode& addStaticVolume(std::unique_ptr<TrackingVolume> volume);
// Cylinder r-stack or z-stack
CylinderContainerBlueprintNode& addCylinderContainer(const std::string& name,
                                                     AxisDirection direction);
// Cuboid x-, y- or z-stack
CuboidContainerBlueprintNode& addCuboidContainer(const std::string& name,
                                                 AxisDirection direction);
// Material designator wrapping a single node
MaterialDesignatorBlueprintNode& addMaterial(const std::string& name);
// Static volume that wraps around contained surfaces
LayerBlueprintNode& addLayer(const std::string& name);
// Configuration assign geometry identifiers
GeometryIdentifierBlueprintNode& withGeometryIdentifier();
```

paul.gessinger@cern.ch ACTS4NP Workshop @ Berkeley 2025 37

Portal construction and registration



- Portal construction from volume boundaries after sizing
- Nodes can take specific action (e.g. cylinder container)
- Portal accumulation strategy configurable, e.g. suitable for `detray`

paul.gessinger@cern.ch 2024-10-23 - CHEP 2024 Kraków 12

DD4hep to Acts Gen3 conversion

- Existing “do-it-all” plugin for DD4hep to Acts Gen1 geometry conversion
  - Geared towards ODD
  - Key differences in certain assumptions / conventions wrt Key4hep geometries in k4geo
  - No/low code solution but limited flexibility
- Plan for Gen3: Implement commonly useful helper functionality in Acts DD4hep plugin
  - Dedicated geometry building downstream
  - “Set of conversion functions” that deal with different sub detectors
  - User knowledge of geometry can be integrated into the Acts geometry building (e.g. by enriching the blueprint tree building with necessary information)
- Hackathon results:
  - Full conversion of ODD
  - General structure for the DD4hep plugin helper functionality in Acts and basic API (plus customization points)
  - Partial conversion of MAIA_v0 geometry (MuColl)
  - Identified some “good to know” things that we will encounter in the process
  - Upstream changes (yet to land in Acts): https://github.com/paulgessinger/acts/tree/feat/dd4hep-gen3
  - WIP integration k4ActsTracking (playground for MAIA_v0 conversion): https://github.com/key4hep/k4ActsTracking/pull/36

MAIA_v0 conversion

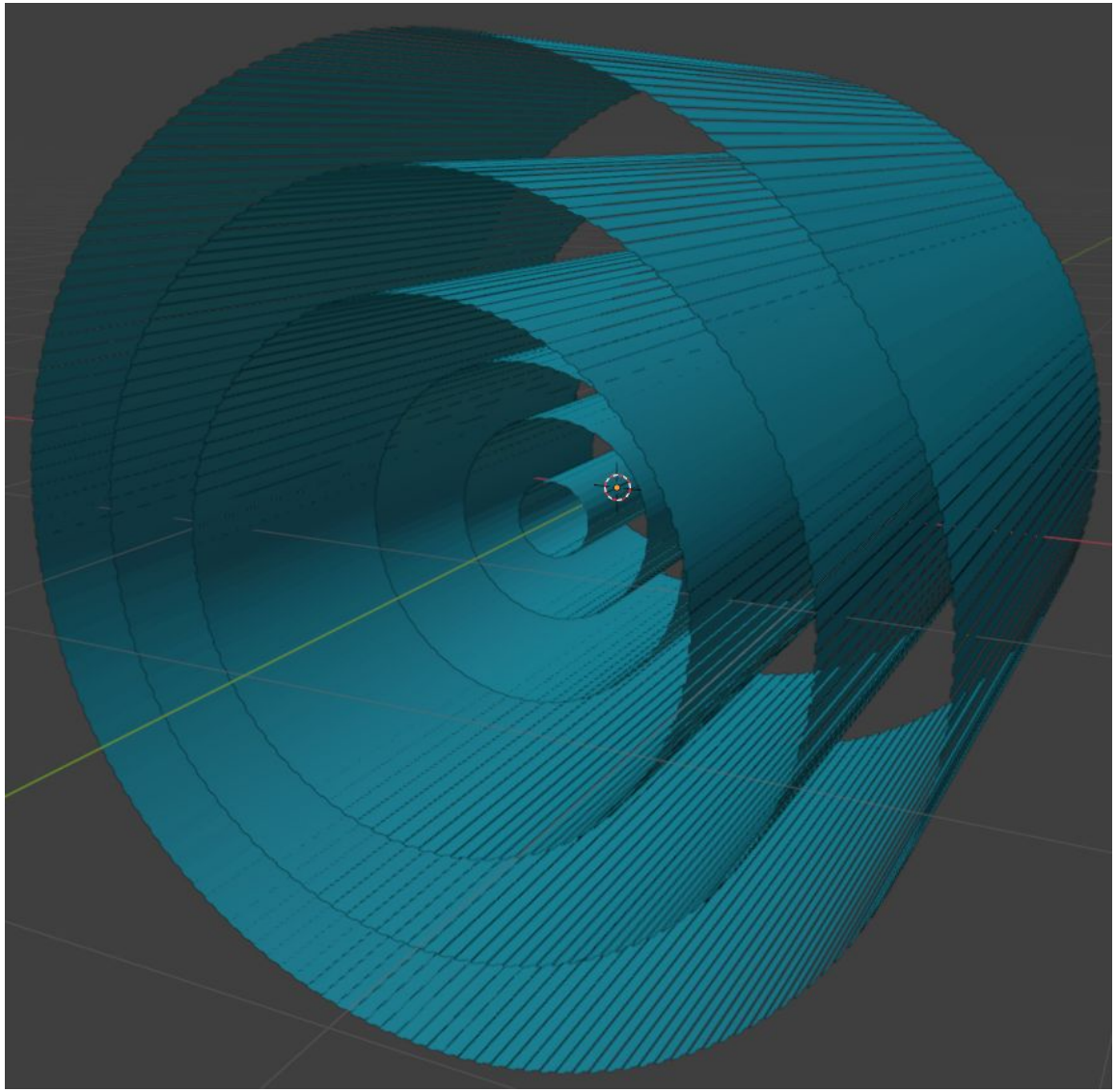
- Create a BlueprintBuilder
- Add a beampipe cylinder
  - geometry needs to reach r=0
- Use the BlueprintBuilder helper functions to walk the DD4hep geometry and match **sensitive** elements to build layers
- Add layers to the blueprint tree
- Construct geometry

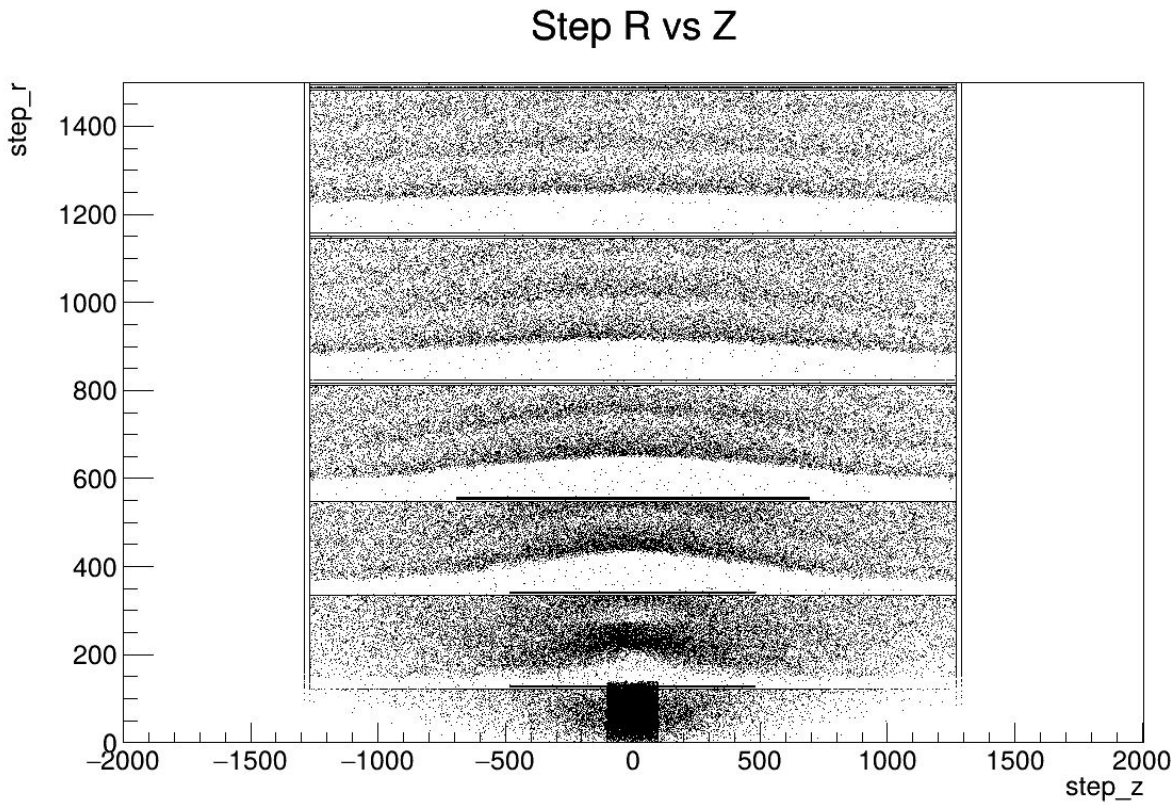
```
ActsPlugins::DD4hep::BlueprintBuilder builder{{
                                                .dd4hepDetector = m_geoSvc->getDetector(),
                                                .lengthScale    = Acts::UnitConstants::cm / dd4hep::cm,
                                            },
                                            gaudiLogger->cloneWithSuffix("|BlpBld")};
```

```
Blueprint::Config cfg;
// Padding around subvolumes of the world volume
cfg.envelope[AxisZ] = {20_mm, 20_mm};
cfg.envelope[AxisR] = {0_mm, 20_mm};
Blueprint root{cfg};

auto& outer = root.addCylinderContainer("MAIA_v0", AxisR);
outer.addStaticVolume(Acts::Transform3::Identity(),
                      std::make_unique<Acts::CylinderVolumeBounds>(0_mm, 10_mm, 1000_mm), "Beampipe");
// We want to pull the next volume in towards the beampipe to map material to
// the correct places in the end. We need to ensure that the enclosing
// cylinder contains the beampipe entirely.
outer.setAttachmentStrategy(Acts::VolumeAttachmentStrategy::First);
```

```
outer.addCylinderContainer("OuterTracker", AxisZ, [&](auto& outerTracker) {
  auto envelope = Acts::ExtentEnvelope{}.set(AxisZ, {5_mm, 5_mm}).set(AxisR, {5_mm, 5_mm});

  auto barrel = builder.layerHelper()
                    .barrel()
                    .setAxes("XYZ")
                    .setPattern("layer\\d")
                    .setContainer("OuterTrackerBarrel")
                    .setEnvelope(envelope)
                    .build();

  barrel->setAttachmentStrategy(Acts::VolumeAttachmentStrategy::First);
  outerTracker.addChild(barrel);
});

BlueprintOptions      options;
Acts::GeometryContext gctxt{};

debug() << "Constructing tracking geometry" << endmsg;
m_trackingGeo = root.construct(options, gctxt, *gaudiLogger->cloneWithSuffix("|Construct"));
```

MAIA_v0 Inner & Outer Tracker Barrel in Acts



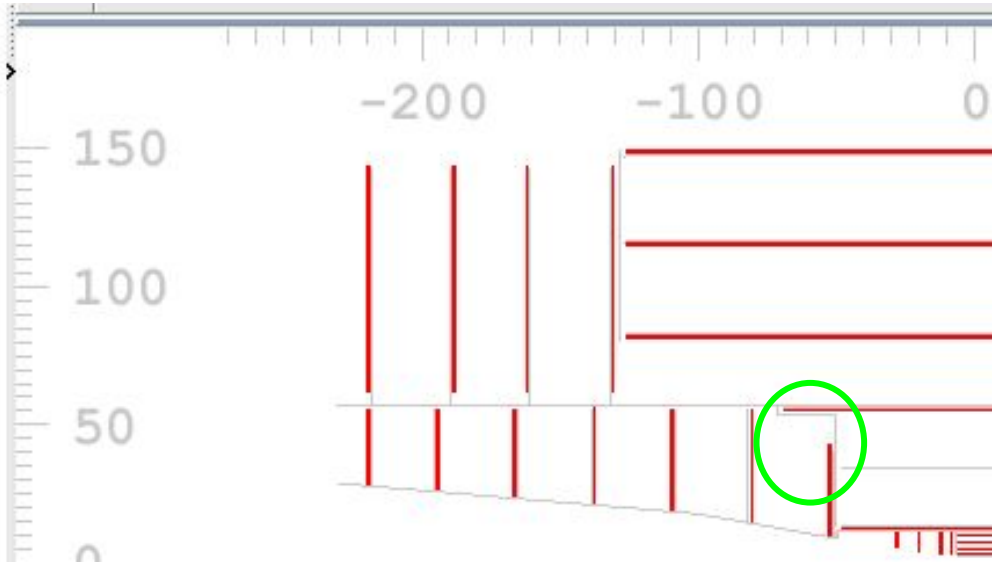


Simple Acts “particle gun” for checking navigation in geometry. Plot shows the step positions of the navigator when propagating a step through the geometry

Navigation through converted geometry works!

Some ~~issues encountered~~ general lessons learned

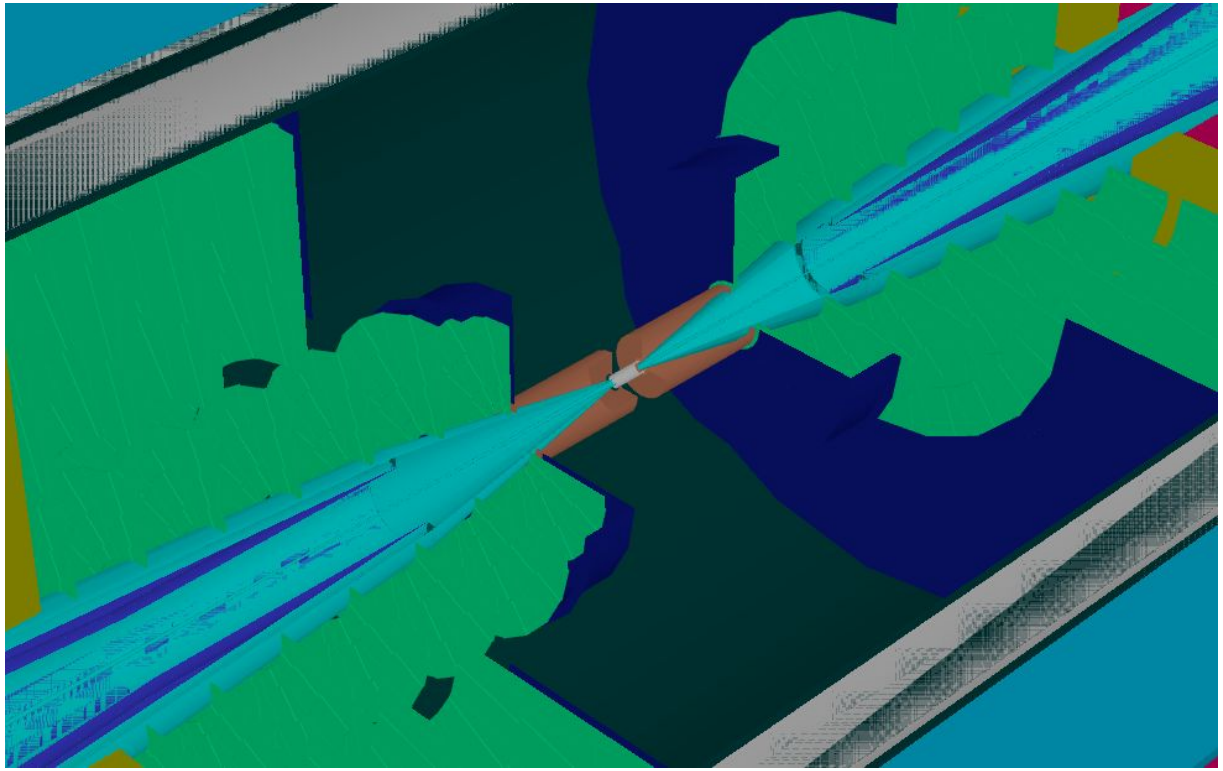
- TrackerEndcap_o2_v06 does not create a layer DetElement in k4geo
  - Makes auto detection in Acts plugin not work
  - Fix in k4geo#549 (assumed in place for the rest)
- Conversion of InnerTrackerEndcap does currently not yield meaningful sensitive surfaces in Acts
  - To be investigated / understood
- InnerTrackerEndcap protrudes into cylinder envelope of InnerTrackerBarrel
  - Naively adding Barrel and Endcap together results in overlap
  - Solution: Stack Barrel and Endcap layers into blueprint tree overlap free
    - e.g. 2 Barrel layers first, then first endcap layer, then final Barrel layer than rest of the endcap layers, …




```
ActsGeoSvc|Cons...VERBOSE Sorting by volume z position
ActsGeoSvc|Cons...VERBOSE Checking for overlaps and attaching volumes in z
ActsGeoSvc|Cons...VERBOSE Checking overlap between
ActsGeoSvc|Cons...VERBOSE   -  z: [  -697.300 <-     0.000 ->   697.300 ]
ActsGeoSvc|Cons...VERBOSE   -  z: [   517.802 <-  1358.852 ->  2199.902 ]
ActsGeoSvc|Cons...  ERROR  -> Overlap in z
ActsGeoSvc          FATAL in sysInitialize(): standard std::exception is caught
ActsGeoSvc          ERROR Volumes overlap in z
```

Some ~~issues encountered~~ general lessons learned

- Beampipe is a simple cylinder in ODD
- k4geo bempipes usually complex composite shapes
- For MuColl additionally the tungsten nozzle
- For geometry conversion for now only take a cylinder around the beam axis
  - Should probably work for the foreseeable future
  - Main concern: Material mapping to the correct final Acts surfaces
  - TODO: Try to take (approximately) appropriate dimensions from DD4hep (must fit inside the first actual tracker layer)
  - Longer term(?) Auto build e.g. nozzles as stack of cylinders with increasing size for more appropriate material mapping



Some ~~issues encountered~~ general lessons learned

- VertexBarrel has odd number of modules
  - Automated cylinder sizing of Acts pulls the resulting volume off the z-axis
  - Can be fixed by disabling the use of center of gravity for sizing
- Still some overlap in r
  - Needs to be investigated / understood

```
outer.addCylinderContainer("Vertex", AxisZ, [&](auto& vertex) {
  auto envelope = Acts::ExtentEnvelope{}.set(AxisZ, {5_mm, 5_mm}).set(AxisR, {5_mm, 5_mm});

  auto barrel = builder.layerHelper()
                       .barrel()
                       .setAxes("XYZ")
                       .setPattern("layer_\\d")
                       .setContainer("VertexBarrel")
                       .setEnvelope(envelope)
                       .customize([&](const dd4hep::DetElement&, auto& layer) {
                         // Force the Barrel onto the z-axis by not using the
                         // center of gravity for auto-sizing. We do this because
                         // the VertexBarrel has an odd number of modules, which
                         // shifts them off-axis when using CoG
                         layer.setUseCenterOfGravity(false, false, true);
                       })
                       .build();

  barrel->setAttachmentStrategy(Acts::VolumeAttachmentStrategy::First);

  vertex.addChild(barrel);
});
```

```
ActsGeoSvc|Cons...VERBOSE Checking for overlaps and attaching volumes in r
ActsGeoSvc|Cons...VERBOSE Checking overlap between
ActsGeoSvc|Cons...VERBOSE   -  r: [    25.140 <->    36.364 ]
ActsGeoSvc|Cons...VERBOSE   -  r: [    27.190 <->    38.342 ]
ActsGeoSvc|Cons...   ERROR  -> Overlap in r
ActsGeoSvc           FATAL in sysInitialize(): standard std::exception is caught
ActsGeoSvc           ERROR Volumes overlap in r
ServiceManager       ERROR Unable to initialize Service: ActsGeoSvc
```

(Attempts at) tracking with the new geometry

- Can build CellID -> Surface mapping on-the fly
  - Replace the whole GeometryIdMapper

```
m_trackingGeo->visitSurfaces([&](const Acts::Surface* surface) {
  const auto& actsDetElem =
      dynamic_cast<const ActsPlugins::DD4hepDetectorElement&>(*surface->associatedDetectorElement());
  const auto& detElem = actsDetElem.sourceElement();
  const auto& [existing, inserted] = m_cellIDToSurface.emplace(detElem.volumeID(), surface);
  if (!inserted) {
    error() << fmt::format(
                  "The Acts surface {} pointing to dd4hep DetElement with cellID {} is already registered in the "
                  "map for Acts surface {}",
                  surface->geometryId(), detElem.volumeID(), existing->second->geometryId())
            << endmsg;
  }
```

```
for (const auto& hit : trackerHitCollection) {
  verbose() << fmt::format("Converting hit {} to Acts. (global) position = {}, cellID = {}", hit.id(),
                           hit.getPosition(), hit.getCellID())
            << endmsg;
  const auto* surface = geoIdMap.at(hit.getCellID());
```

(Attempts at) tracking with the new geometry

- Tried to do Tracking using only Inner/OuterTrackerBarrel with MAIA_v0
- Not working, because
  - Seed SpacePoint selection doesn’t select any of the hits
  - Even when passing all hits to seed finding no seeds found
- All of this will probably need a re-tuning after the new geometry is in place
  - Not sure if it will be necessary, but CellID -> Surface mapping might be different with Gen3 geometry

Next steps

- Implement a simple KalmanFilter and use single particle gun to see if track fit is possible
- Investigate conversion issues in InnerTrackerEndcap and Vertex
- Convert more of the MAIA_v0 geometry