EIC Central Detector integration software suite

(aka EIC Toy Model)

Alexander Kiselev

BNL NPPS Group Meeting July 10 2020

In place of the introduction

- **By the end of this talk you may have natural questions like:**
  - There are packages A,B,C,… which can do *(almost)* what you needed; why re-inventing this wheel all over again?

    This one is hard, depends somewhat on the personal preferences

  - Even then, why choose such a weird ROOT-centric implementation?

    This one is easy (the tool was not meant to be a geometry manager :-)

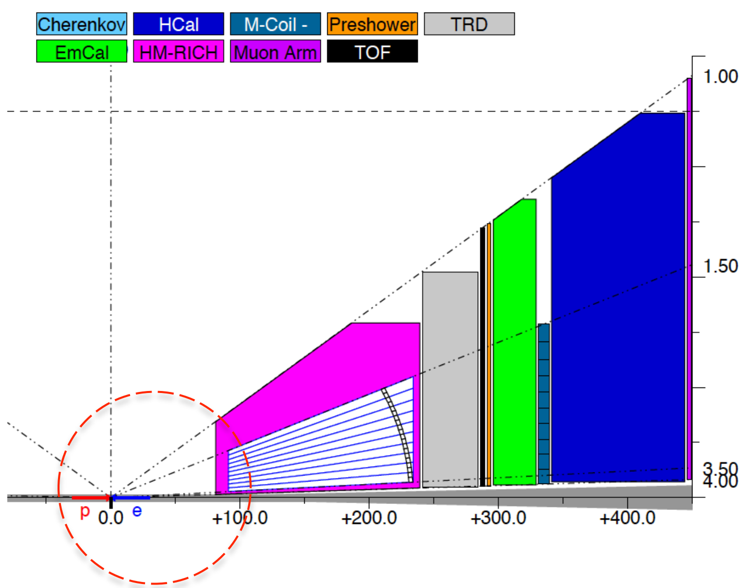
End of April: the starting point

Commitment #1: work on the EIC greenfield solenoid specs document

	"Status"	Minimal	Default	"Ideal"
Muon Detector	optional	0	5 cm	
Hadronic calorimeter	mandatory	105 cm	105 cm	~150 cm
Correction coils	optional	0	10 cm	0 ☺
e/m calorimeter	mandatory	35 cm	35 cm	>35 cm
Preshower	optional	0	5 cm	
Time of flight	optional	0	5 cm	
GEM-TRD	TRD functionality is optional	~15 cm	45 cm	~60 cm
High-momentum RICH	mandatory	~120 cm	165 cm	

Table 1 Forward endcap space allocation.

Given the obvious space limitations in the forward endcap it seems to be reasonable to push the calorimetry equipment towards the very end of the +4.5 m zone from the start.

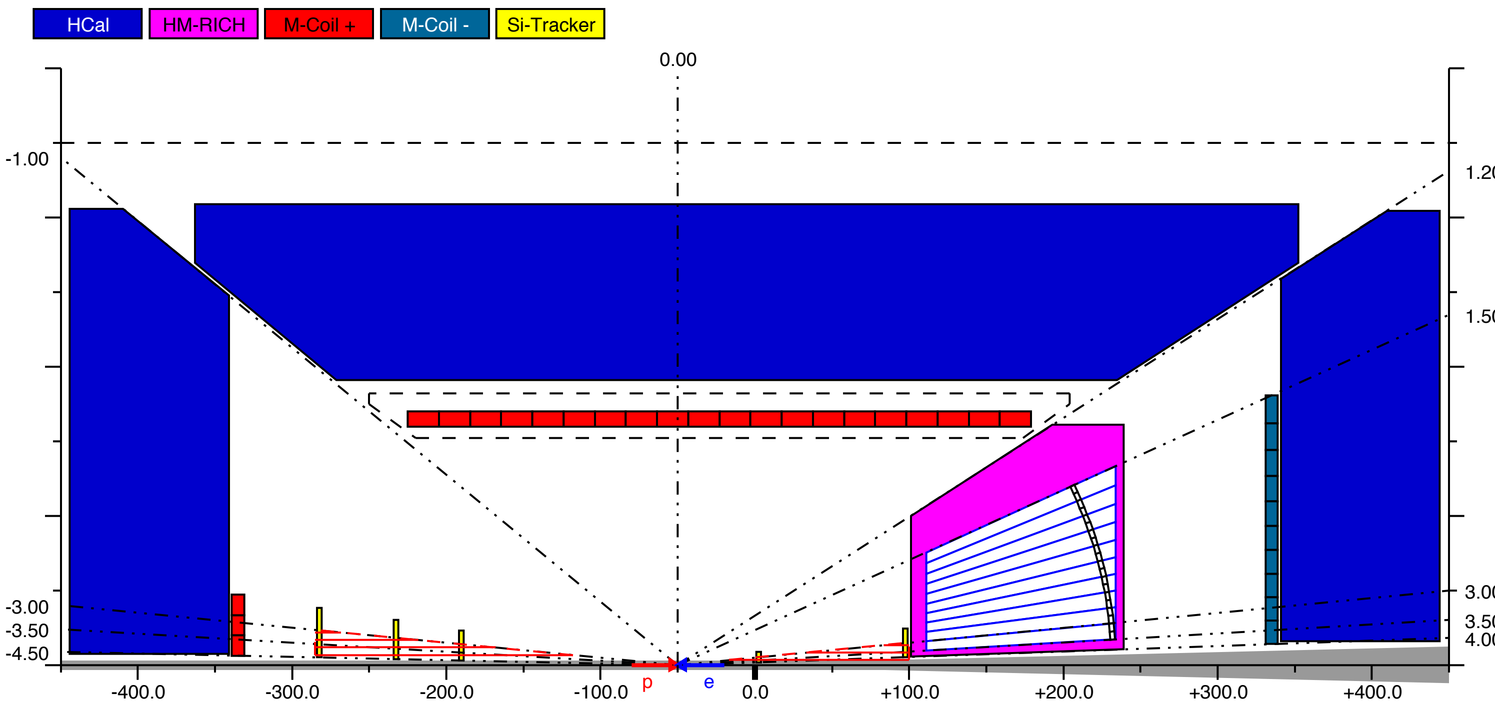


The detector composition, which includes all of the above-mentioned subsystems, with their respective **default** space allocations, requires ~3.75 m along the beam line direction, and has no real contingency included. In a "symmetric" configuration, shown in Figure 5, when the nominal IP is located in the center of the +/- 4.5 m region, provided by the accelerator layout, this would

A week of time and several hundred lines of ROOT / C++

The starting point, cont'd

The primary goal: provide a set of cartoons like this



- **Just be able to illustrate several key features of the detector layout:**
  - A definitive location of the flux return elements (hadronic calorimeters)
  - A supposed location of the gaseous RICH (projective field required)
  - An optimal location of the nominal IP (split space between two endcaps)
  - The range where a constant magnetic field is desirable

An attempt to connect some of the other dots

Commitment #2: EIC Yellow Report Central Detector integration WG

Escalate & fun4all; migration process	**Physics simulations & engineering design**
Tracker, PID & Calorimetry detectors in GEANT	**Ideal detectors & services / support**
1-st & 2-d IR	**$\|\eta\|$<4.5 & reality**
EIC detector & greenfield solenoid design	**Space available for the detectors & IR vacuum chamber**

- One can easily identify a number of places with a lack of sync at this early stage
- Some of them can seemingly be addressed in a more or less consistent way

Hack something together on top of the existing cartoon tool?

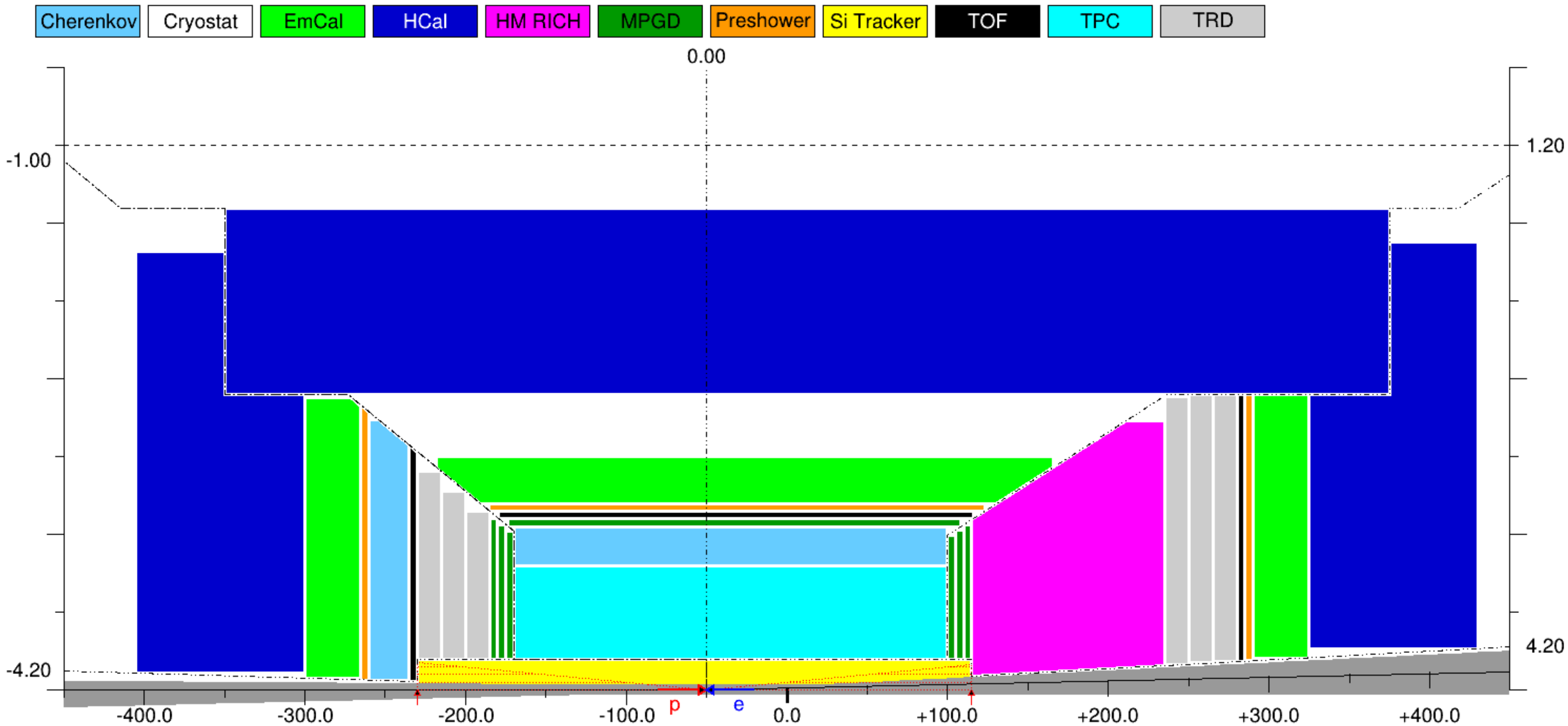
EIC Toy Model: overview

- **A tool to model & generate EIC Central Detector "templates" in a way:**
  - the new geometries (models) can be generated "quickly" …
  - … *by everybody*, and represented instantly in a WYSIWYG fashion
  - the sub-detector "container objects" are guaranteed to not overlap either with each other or with the IR vacuum chamber elements
  - technically they can be imported in GEANT frameworks in a consistent way and used as wrappers to the "real" sub-detectors
  - they can be exported in a CAD format to be used in the engineering design of the detector support structures and / or laying out services

- **Repository:** https://github.com/eic/EicToyModel
  - a README.md file ☺
  - example ROOT scripts
  - a standalone GEANT example
  - detailed API description

The workflow

Create a model → **Save it as a .root file** → **Import in GEANT**



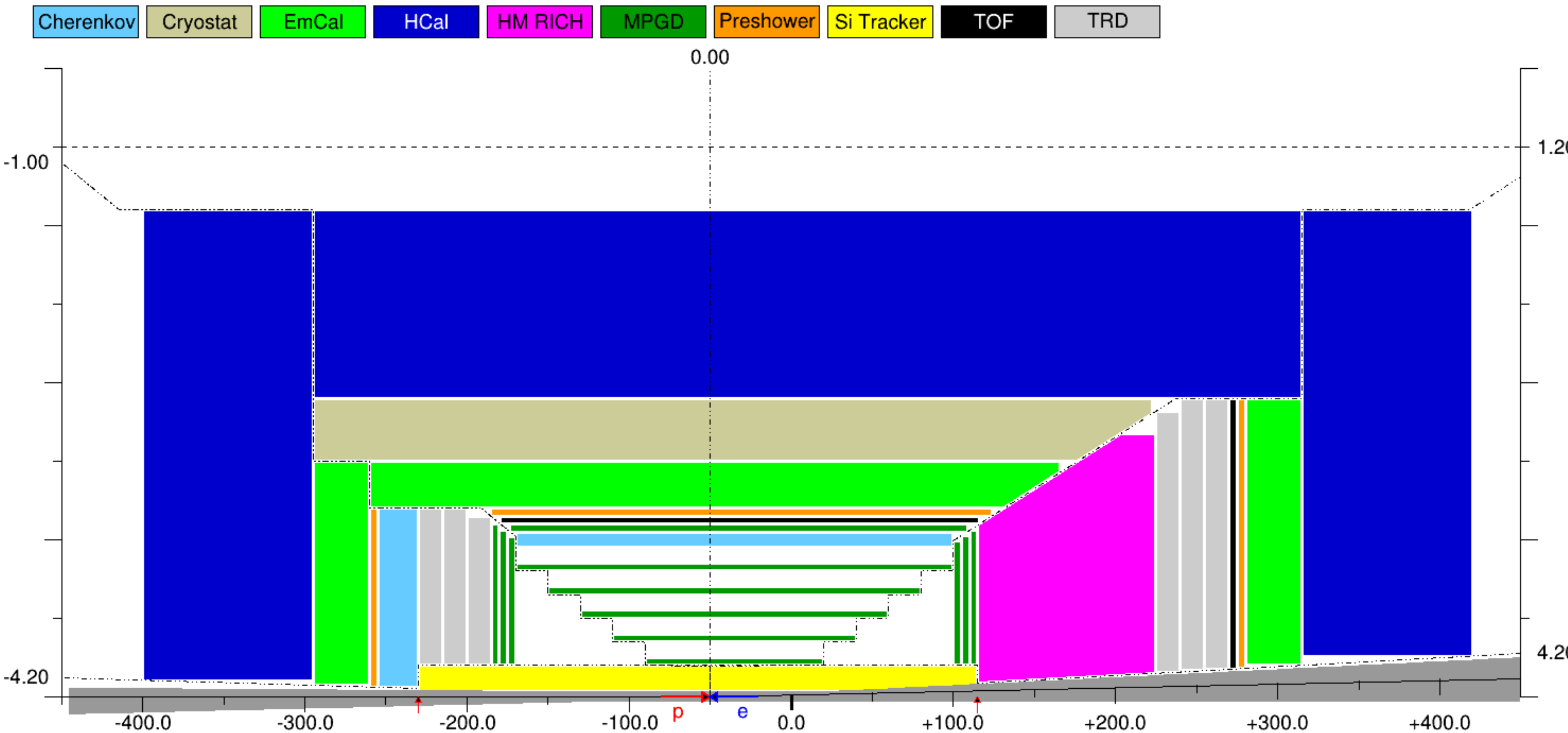
- https://github.com/eic/EicToyModel/scripts/example.C
- Minimal overhead to create a 2D scheme like this (ROOT scripting)
- Model can be saved, distributed and re-imported as a .root file
- GEANT application: import .root file and **create volumes on the fly**
  - Alternatively: export and import GDML file(s) *(can be implemented)*

What is under the hood

- **A small ROOT-based C++ library, with several interfaces:**
  - GEANT4: dynamic conversion of a 2D cartoon into G4 “container volumes”
  - OpenCascade: export to STEP format
  - VGM: IR vacuum chamber TGeo -> G4 conversion for a “boolean cut”
  - VGM: direct import of EicRoot-like models into GEANT *(experimental)*
  - BeastMagneticField: ASCII field map import *(forward compatible format)*
- **Custom simplified IR vacuum chamber implementation**
  - *(In theory)* it is parametric, so can be used to create e.g. a 50mrad layout
- **Limited set of interactive commands (IP shift, η range change, …)**

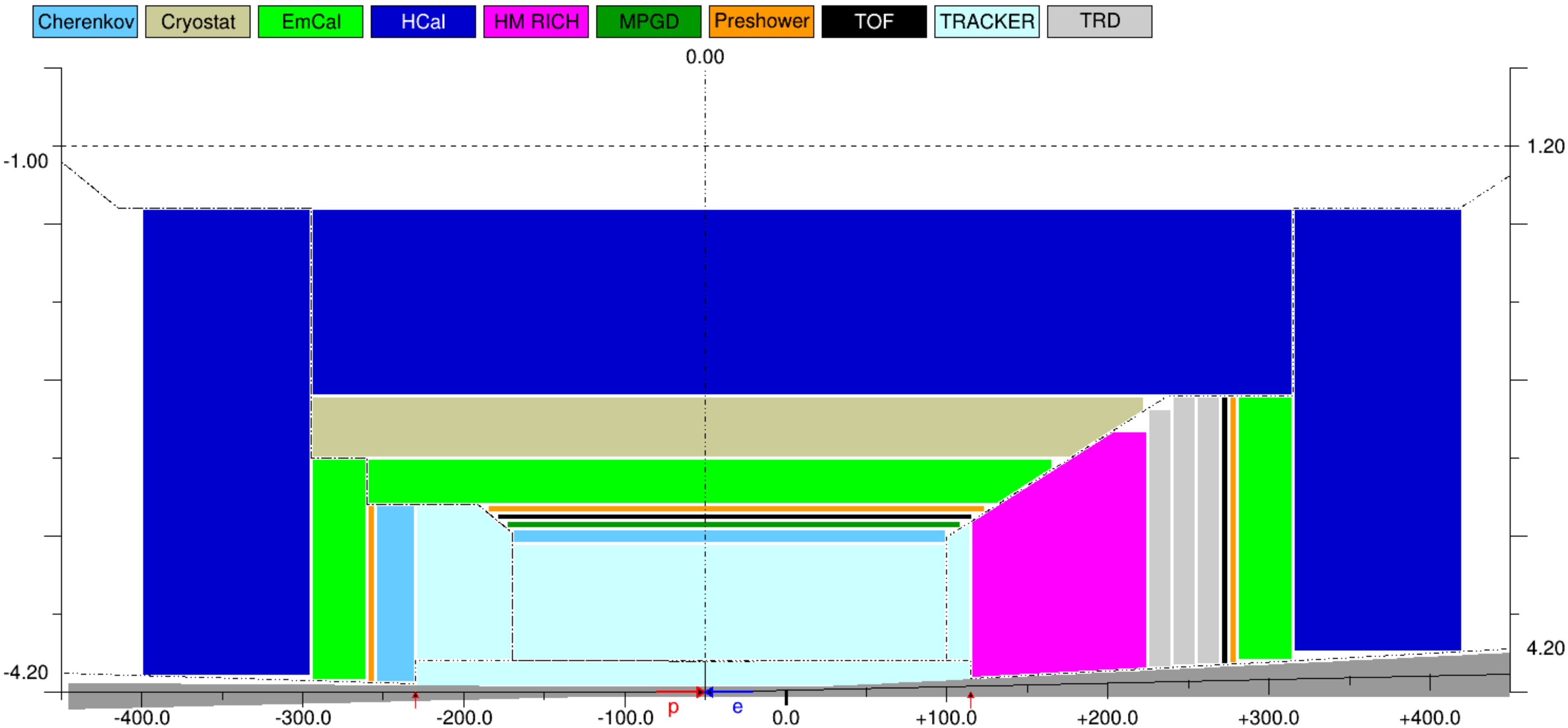
Library has to be installed locally
Supposed to run on Linux *(seemingly works under Mac OS as well)*

Integration volume granularity: tracker



- **Detector grouping is certainly possible**
  - Is it flexible enough?
  - As shown here: too detailed at this early stage?
  - https://github.com/eic/EicToyModel/blob/master/scripts/tracking.C

Integration volume granularity: tracker



- **Detector grouping is certainly possible**
  - Is it flexible enough?
  - Allocate larger volumes for PID / Tracking / Calorimetry, to start with?
  - https://github.com/eic/EicToyModel/blob/master/scripts/tracking.C

Limitations in the geometry description

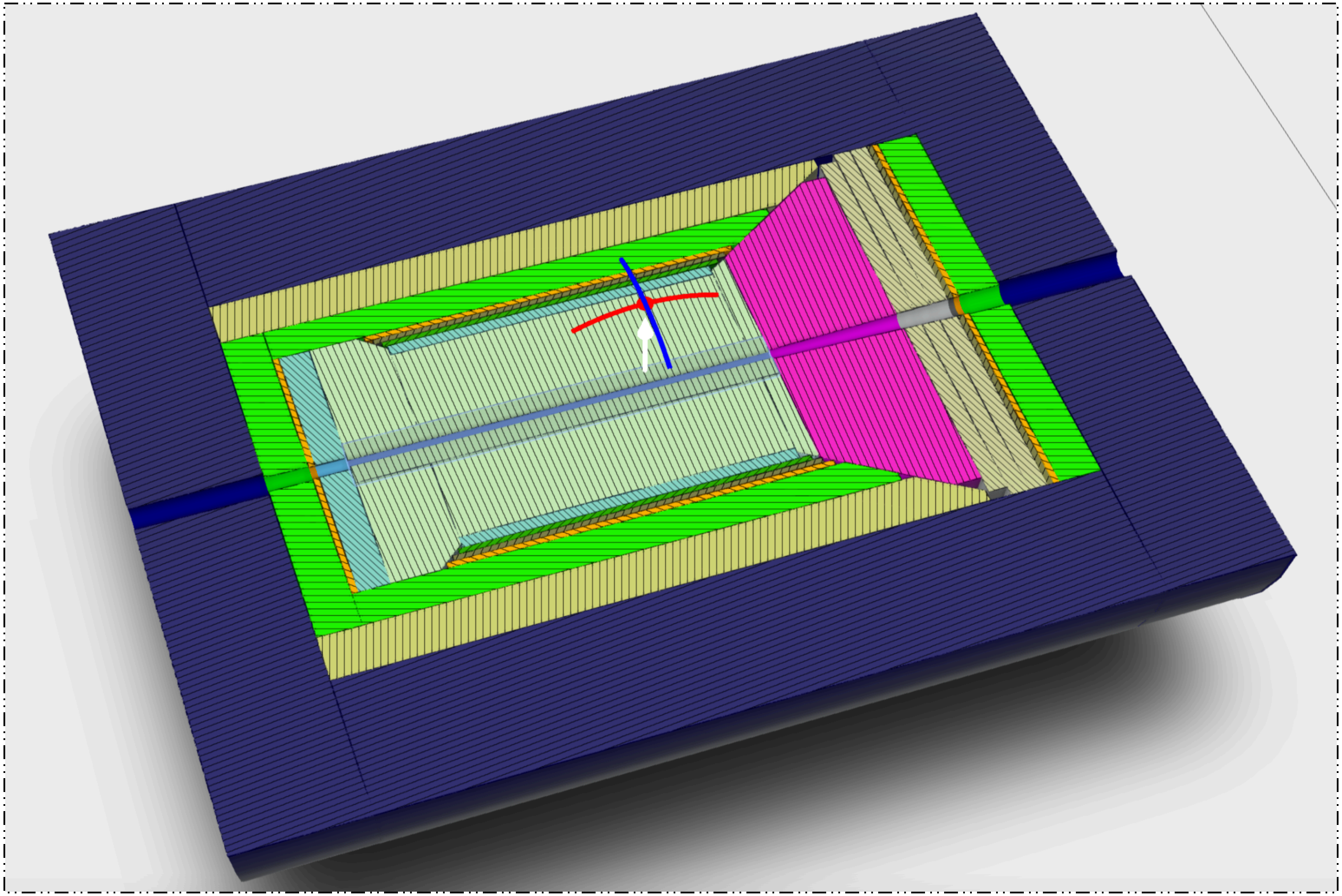
- **Four pre-defined detector "stacks": vertex, barrel, and two endcaps** …
- … **in a projective configuration (defined by the η ranges)**
- **Detector volumes in the endcap stacks are placed as strictly aligned objects with flat front and rear sides, one after the other**
  - ... although stack boundaries can be shaped up creatively, if needed

- **Detector tags (like "EmCal") and respective colors are hardcoded …**
  - … though custom ones can be generated dynamically, if really needed

- **Exported objects are azimuthally symmetric Polycones, although …**
  - … with an asymmetric cutaway representing the IR vacuum chamber
- **Polyhedra export implemented, but can not be mixed with Polycones**

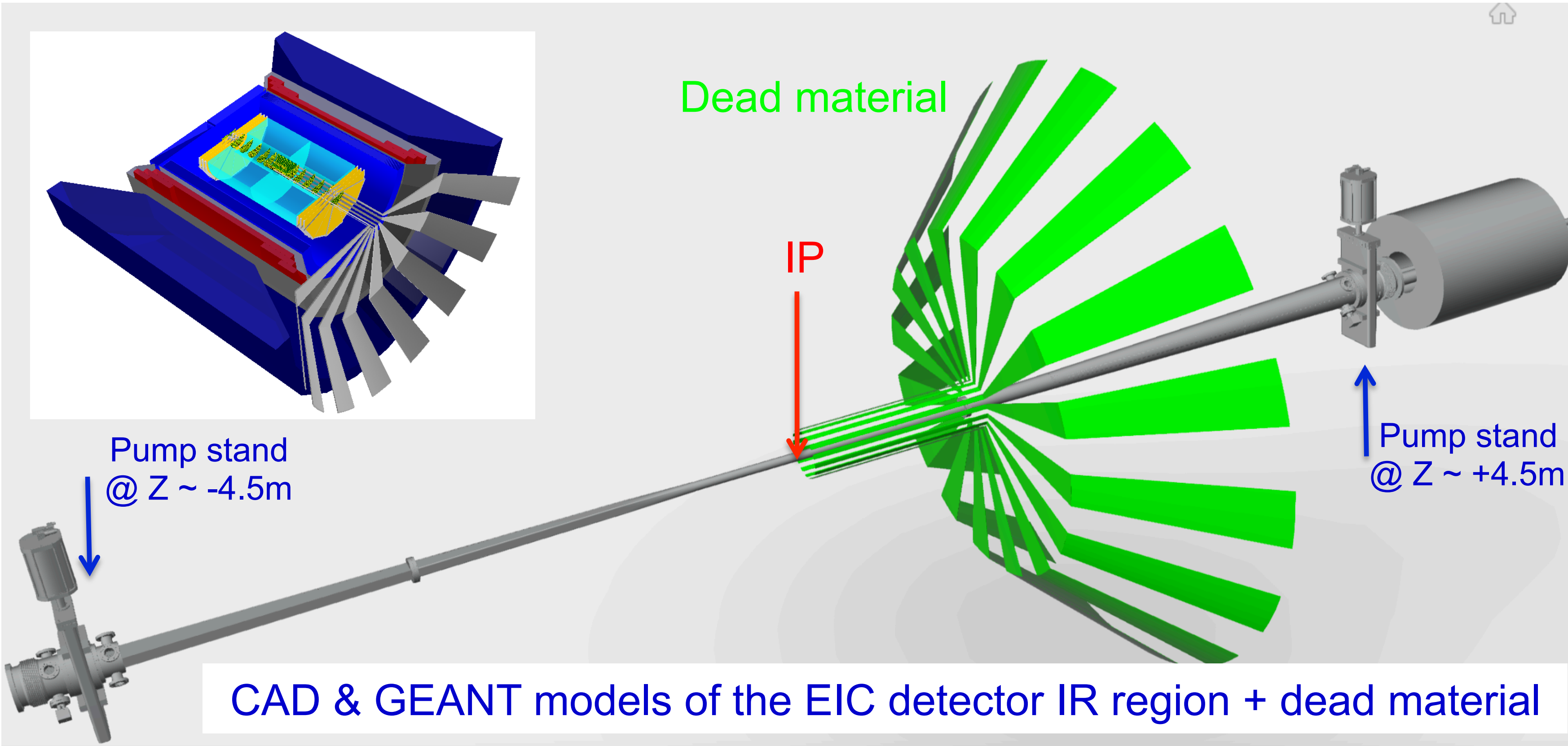
- **CAD export: presently without the vacuum chamber cutaway**

CAD interface (3D model in Autodesk viewer)



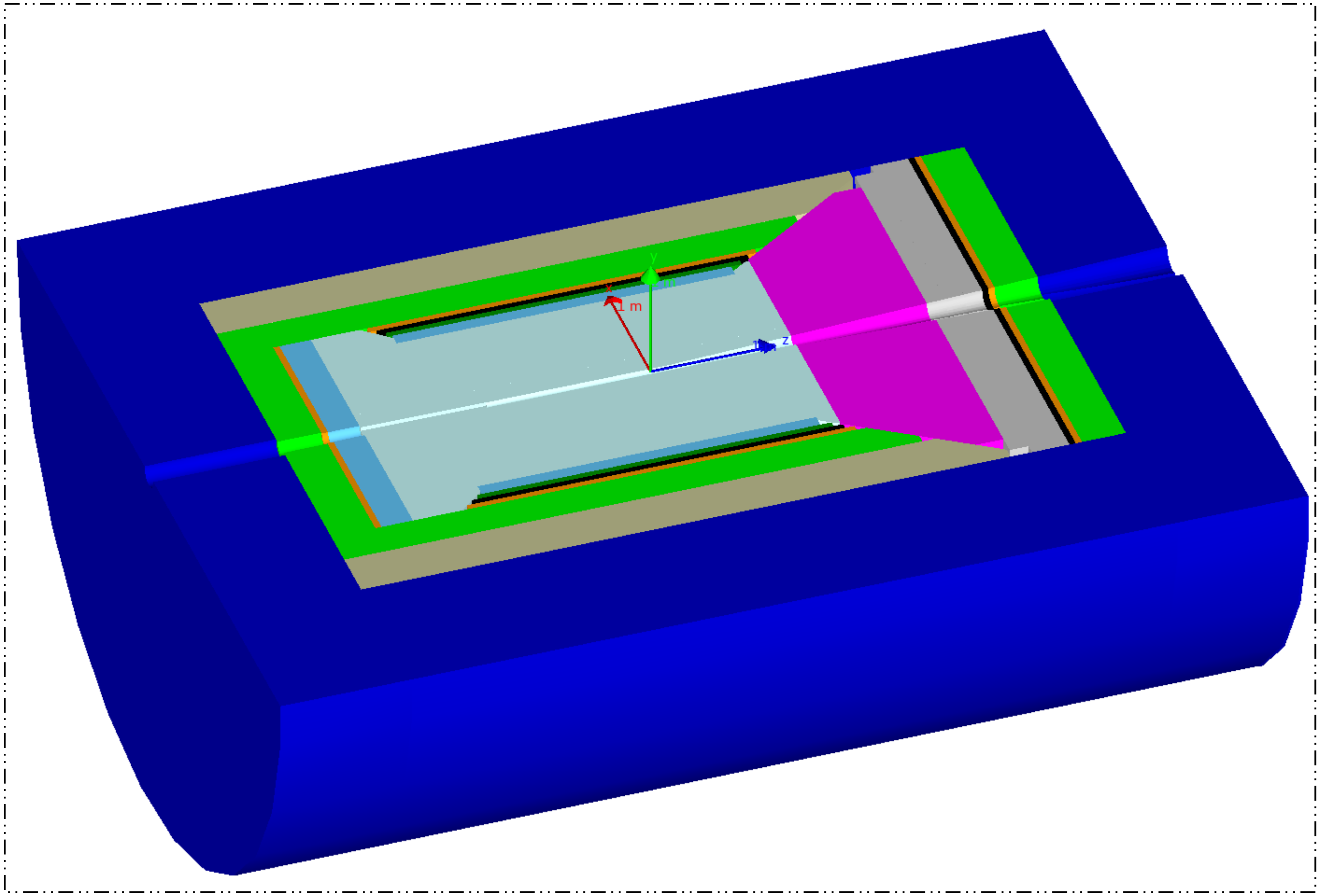
- GEANT picture will look identical
  - Services and support structure engineering design can start off the same configuration as used in GEANT for physics simulations

Support, services, detector frames: *TODO list*



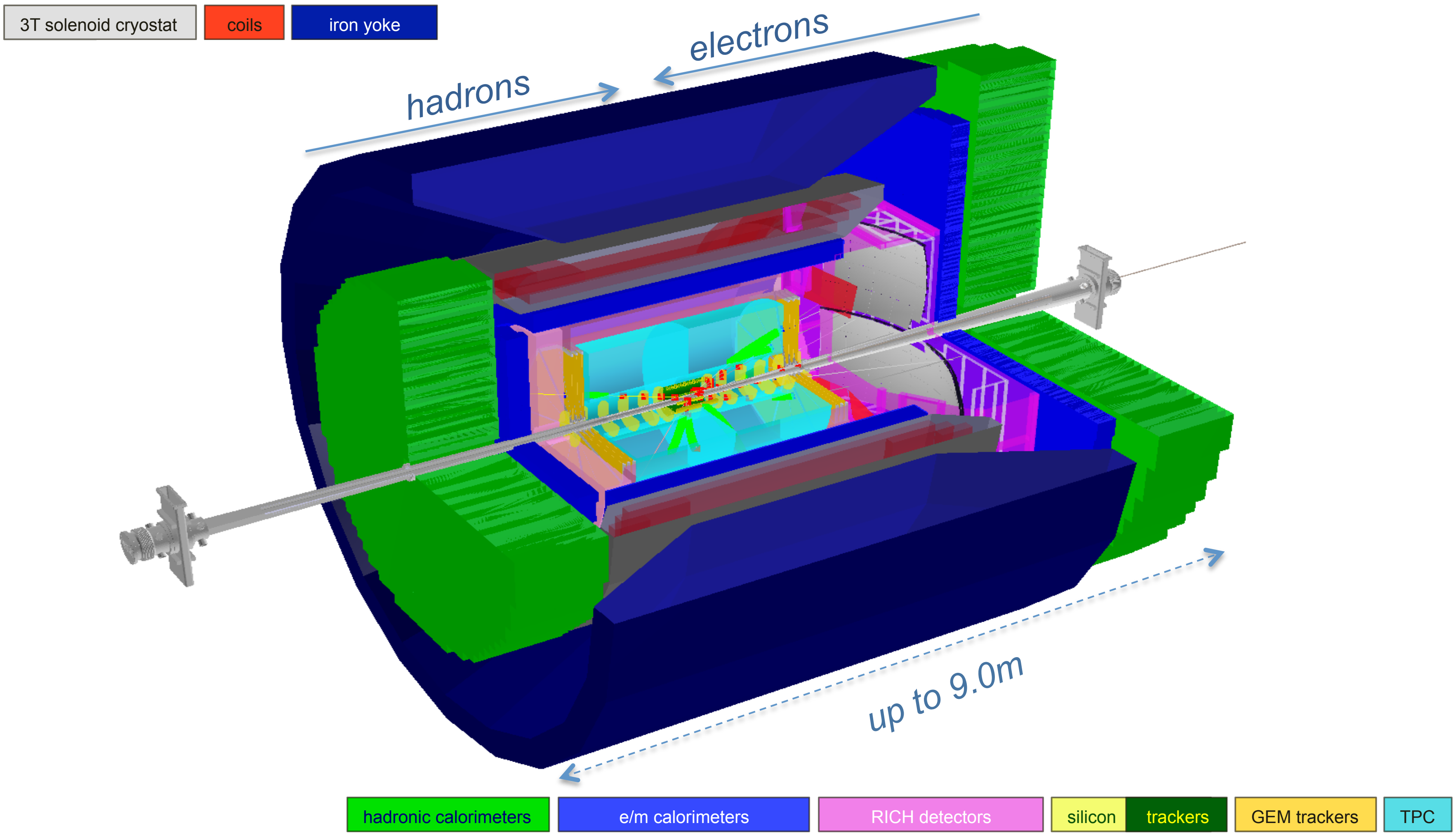
- Support structure:
  - Generic part (outside of the integration volumes): engineering effort
  - Matching detector-specific part (inside the integration volumes ?)
- Services: should be configurable, accumulating from / to “inner” detectors
- Detector frames: should naturally come together with the active volumes

GEANT interface (Qt event display)



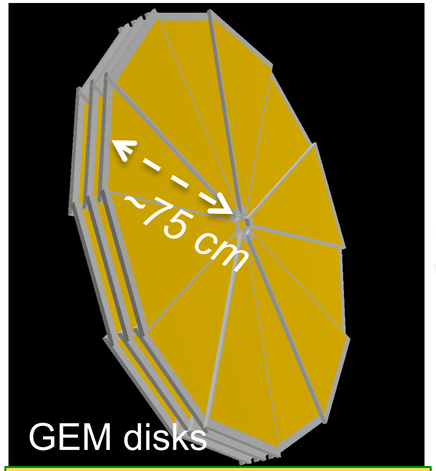
- Volumes are currently generated on the fly *(is GDML step really needed?)*
- Once imported, the layout will look the same **in all G4 applications**

Compare: BeAST EicRoot implementation

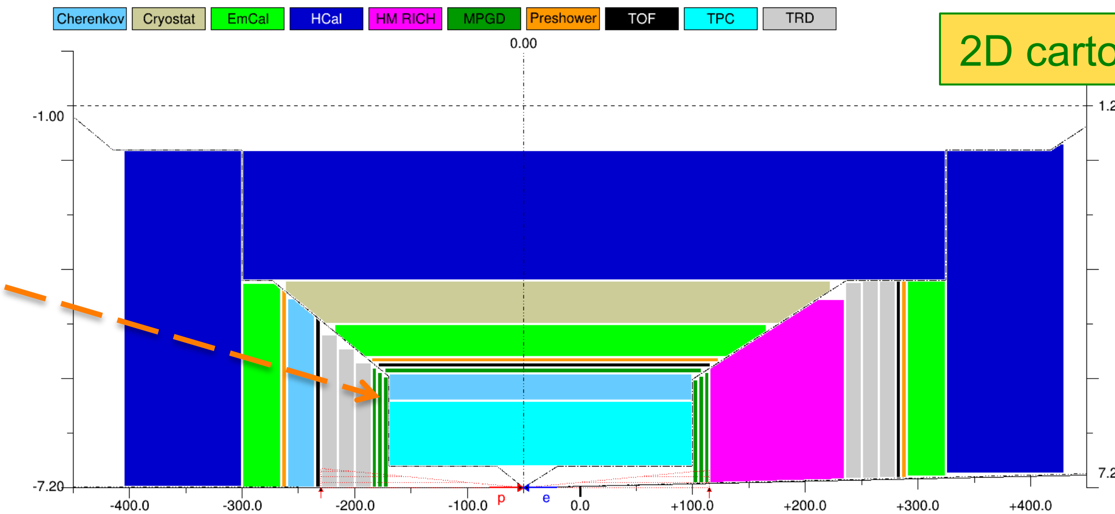


- Comment#1: some of the volumes here (PID) *are also air balloons*
- Comment#2: one can seemingly reuse TGeo objects in the new scheme

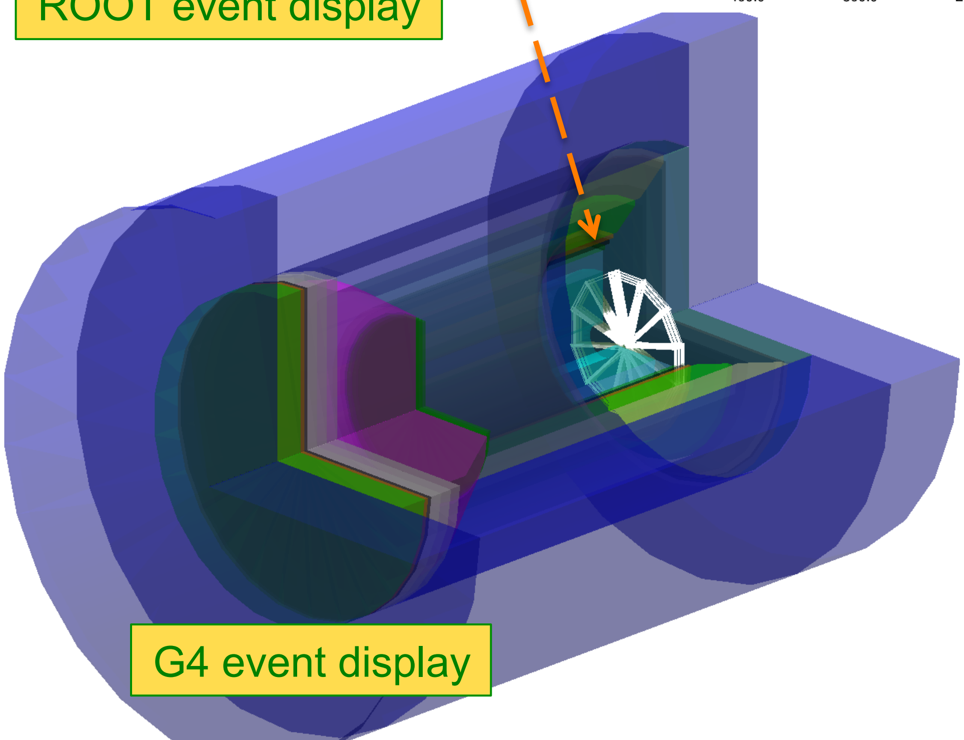
Optional EicRoot geometry import



ROOT event display



2D cartoon

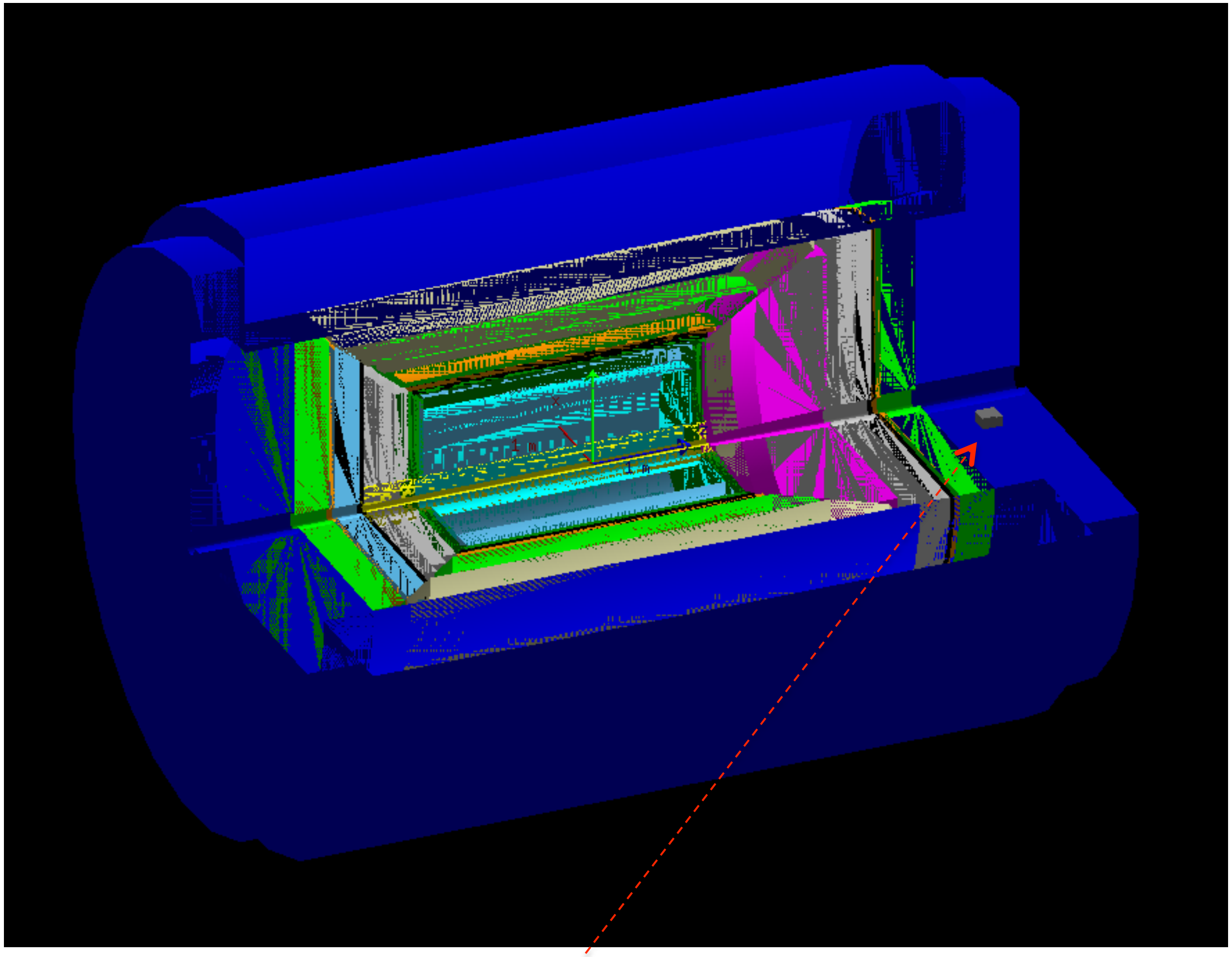


G4 event display

- Yet *experimental,* but seems to work, as expected
- Possible other candidates: MM barrel, *silicon vertex*, calorimetry
- Material information merging from different files may be an issue

EIC frameworks: *fun4all* event display

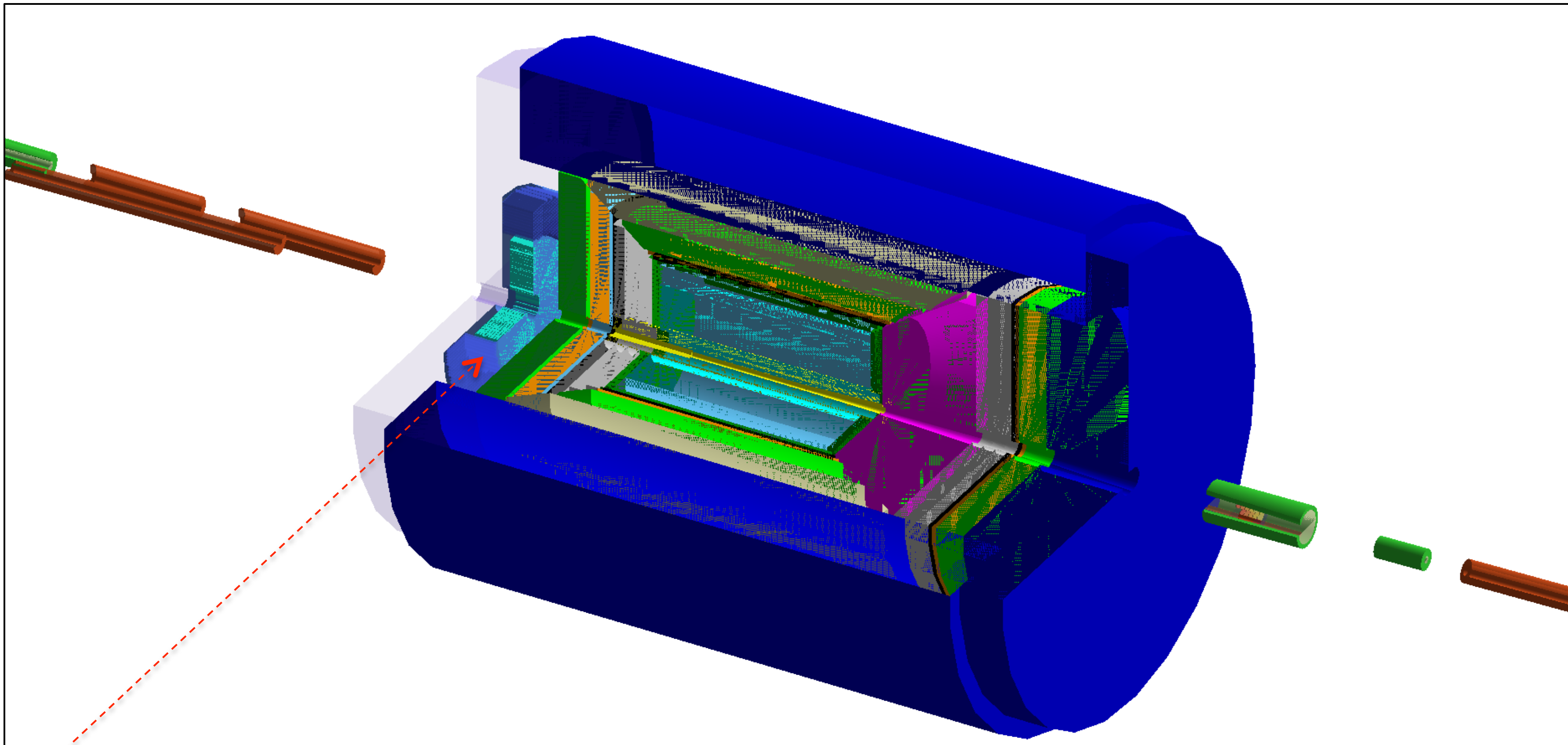
- **RACF: ROOT 6.16.00, GEANT 10.2.2**
  - Shown here: integration in one of the fun4all example codes



This object is a “template G4Box” placed into one of the container volumes

EIC frameworks: *escalate* event display

- **Latest escalate Docker container: ROOT 6.20.04, GEANT 10.6.1**
  - Shown here: direct integration into g4e JLEIC source code



This object is a “native” JLEIC EmCal placed into the respective container volume

Coding overhead in GEANT

Excerpt from a modified working calorimetry code:

```
// Construct the integration volumes geometry, internally;
TFile fin(argv[1]);
dynamic_cast<EicToyModel *>(fin.Get("EicToyModel"));
eic->Construct();
// Populate G4 world by these volumes;
eic->PlaceG4Volumes(expHall_phys);

// Place "MyHCal" tower matrix into the integration volume bubble instead of the world;
new G4PVPlacement(0, G4ThreeVector(0, 0, zOffset), myhcal_log, "MyHCal", expHall_log,       false, 0);
auto hcal_bubble_log = eic->fwd()->get("HCal")->GetG4Volume()->GetLogicalVolume();
new G4PVPlacement(0, G4ThreeVector(0, 0,       0), myhcal_log, "MyHCal", hcal_bubble_log, false, 0);
```

(The line beginning `new G4PVPlacement(0, G4ThreeVector(0, 0, zOffset)` is struck through on the slide.)

This part should be taken care of by the framework

- Immediate migration is not mandatory for everybody
  - Integration bubbles can be imported into a framework one by one
- Bubble size (and location) can be polled *(trivial; implemented partly)*
  - Parametric detectors can be implemented in a proper way
- If the community prefers to use GDML files instead, so be it (consistency?)

Coding overhead in GEANT: *escalate* case

```
void JLeicDetectorConstruction::Create_ce_Endcap(JLeicDetectorConfig::ce_Endcap_Config cfg)
{
  // ...

  // Import ROOT file with an "EicToyModel" singleton class instance;
  auto eic = EicToyModel::Import("example.root");

   // Construct the integration volumes geometry, internally;
  eic->Construct();

  // Place them as G4 volumes into the IR world volume all at once;
  eic->PlaceG4Volumes(World_Phys);

  // Get pointer to a particular G4VPhysicalVolume;
  ce_ENDCAP_GVol_Phys = eic->bck()->get("HCal")->GetG4Volume();

  // ...
}
```

- All in all: the overhead is seemingly very small
- Step by step details are communicated to the framework developers

Magnetic field map interface

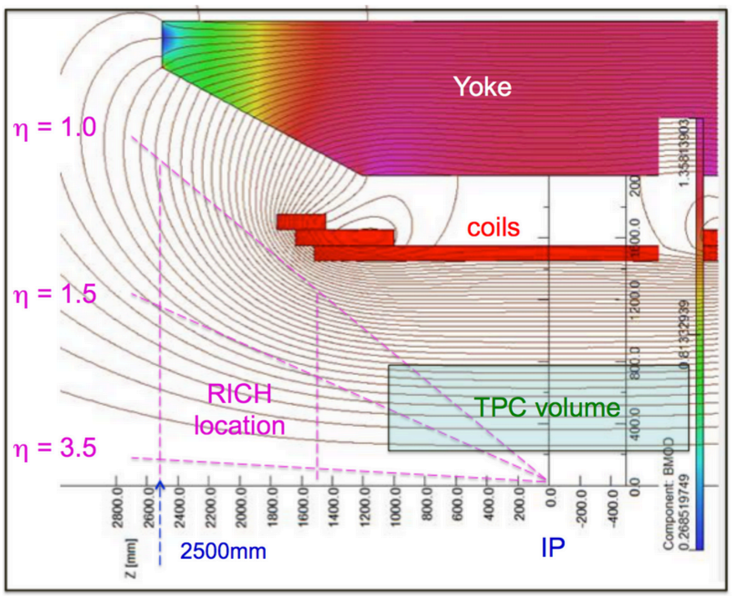
https://eic.github.io/software/beast_magnetic_field.html

Computing | Software | Working Group | Information | About

BeAST solenoid magnetic field map

The repository contains an ASCII file with the field map, a C++ class to handle it and a GDML model

https://github.com/eic/BeastMagneticField



Open solenoid design (no field clamps)

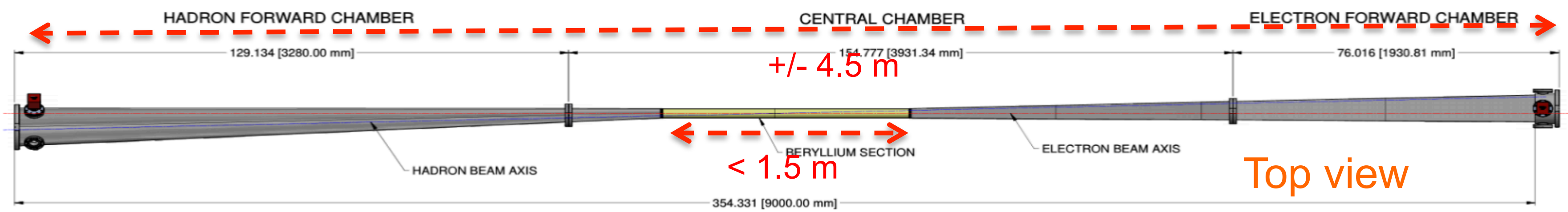
Homogeneous (less than than 4% variation) 3T central field in the TPC volume

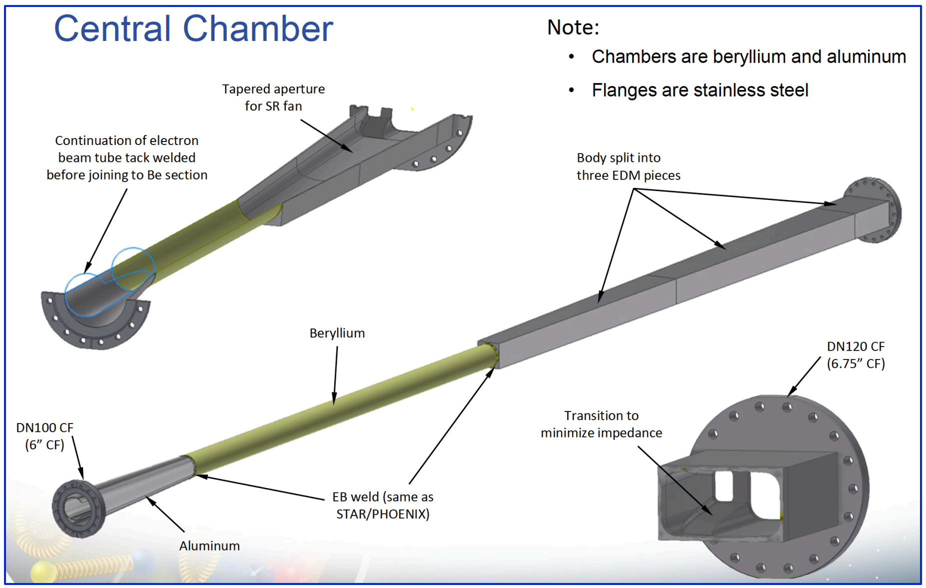
Fringe field is tuned in order and minimize charge particle bending in the forward gaseous RICH volume (less than 1mrad RMS for 10 GeV/c particles up to 25 degree polar angles)

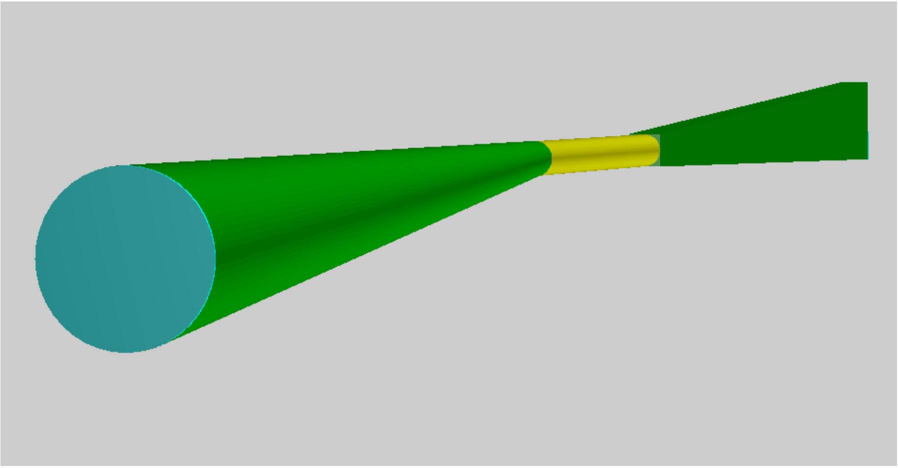
Field map originally produced by a collection of Open Source tools (Elmer, Netgen, ROOT)

- Currently only BeAST field map import implemented; *ePHENIX coming soon*
- Interface is forward compatible with the greenfield solenoid maps (?)

IR vacuum chamber description



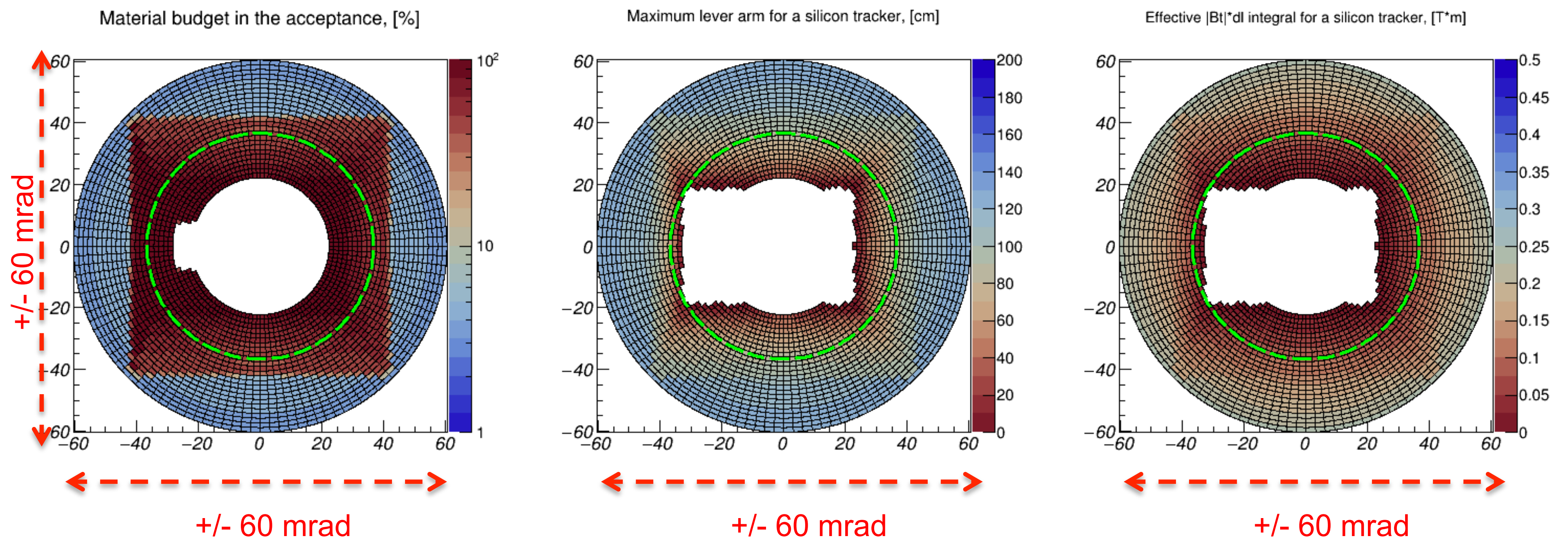




CAD drawing and ROOT TGeo implementation

- Coded in TGeo, exportable as GDML
- Exported to G4 representation (through VGM), used for a boolean cut
- Kind of parametric *(and as such suitable for the 2-d IR description)*
- *Only the essential part (the outer shell in particular) is implemented*

B*dl integral and material scan evaluation



- Material budget: direct use of the vacuum chamber TGeo implementation
- Estimate of the maximum lever arm available for the silicon tracker:
  - Account for the vacuum chamber shape: consider a 3D point where a particle with a given $\{\theta,\phi\}$ would exit the vacuum chamber (starting point) …
  - … and account for the configurable markers, indicating at which max distance from the IP the last silicon tracker station can be installed (end point)
- B_T*dl integral estimate: same idea + BeastMagneticField interface
- Primary vertex smearing implemented (this part is trivial of course)

Documentation

https://github.com/eic/EicToyModel/ Search

README.md

Perhaps almost too detailed already

EIC Toy Model (ETM)

EicToyModel (ETM) is a C++ ROOT-based software suite for EIC Central Detector configuration purposes.

Content:

Introduction

The main objective of this software was providing the EIC community users with a tool, which would allow them to easily (whatever easy means) create possible EIC detector configuration *templates* (namely, the self-consistent collections of 3D sub-detector intergation volumes) and make use of them in the GEANT simulation environment.

Next steps

- Finish the TODO list items (services, EicRoot detector model import, etc.)

- Next week: a discussion at the EIC Users Group meeting
- Afterwards: a second round of presentations for the YR Detector WGs
- At some point: a tutorial?

- Critical: a clean implementation in the EIC frameworks