

Geometry implementation for the FCC

Alvaro Tolosa-Delgado (CERN)

SFT Group Meeting

May 6st, 2024



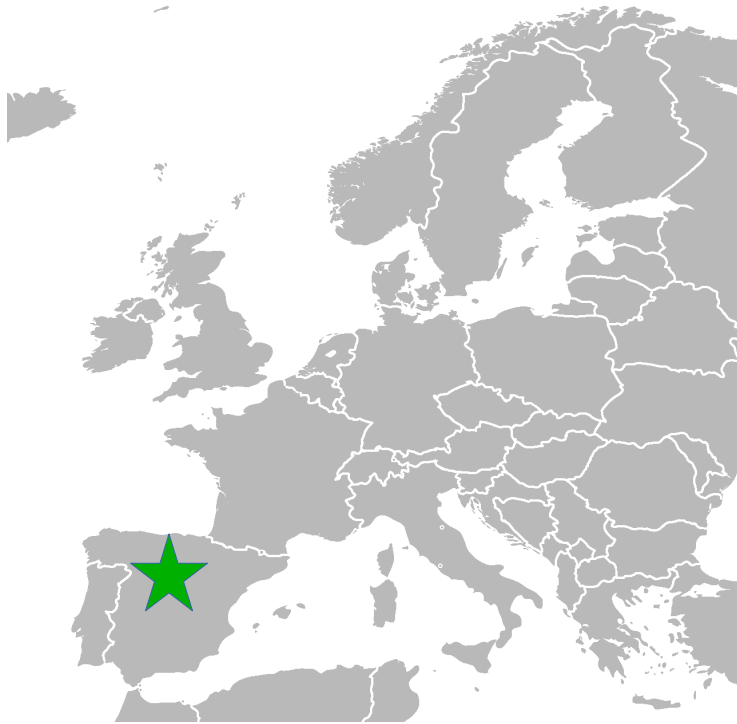
**FUTURE
CIRCULAR
COLLIDER**



About me...



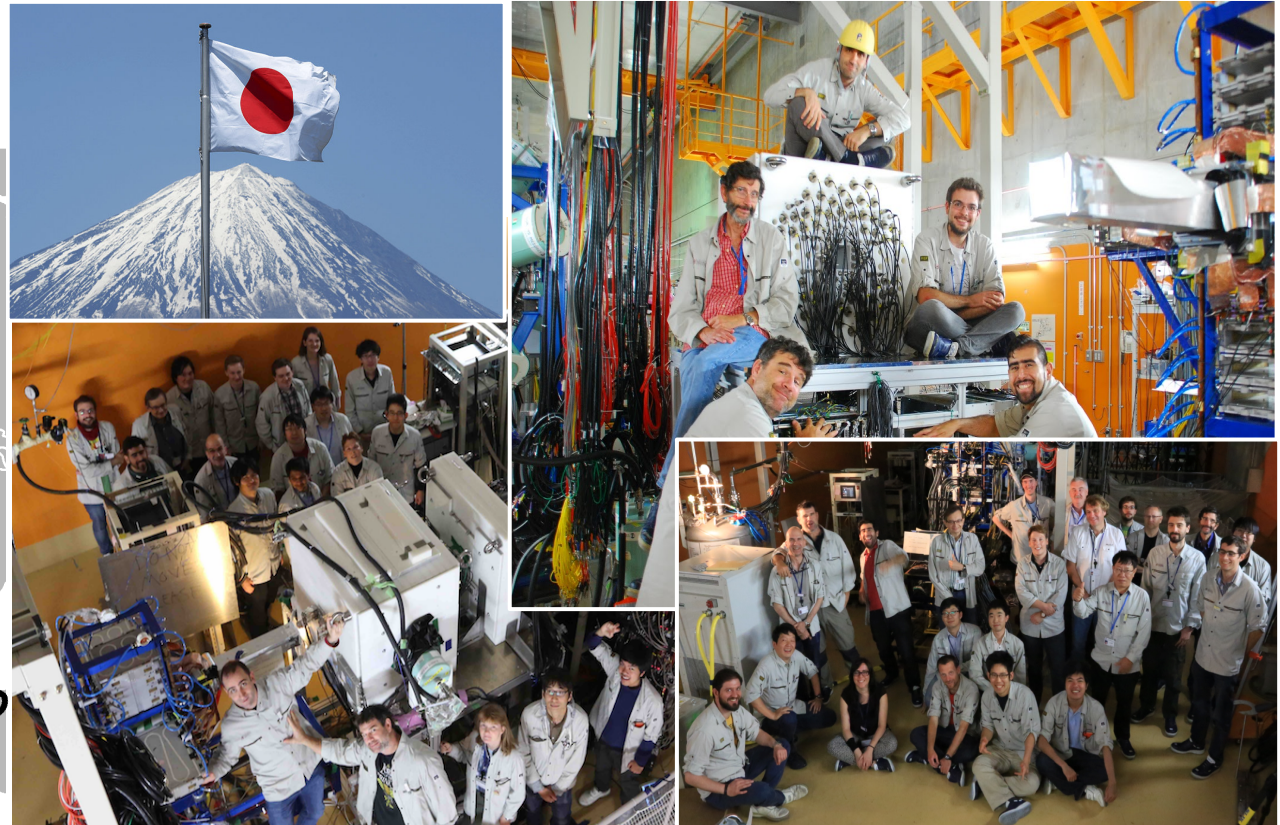
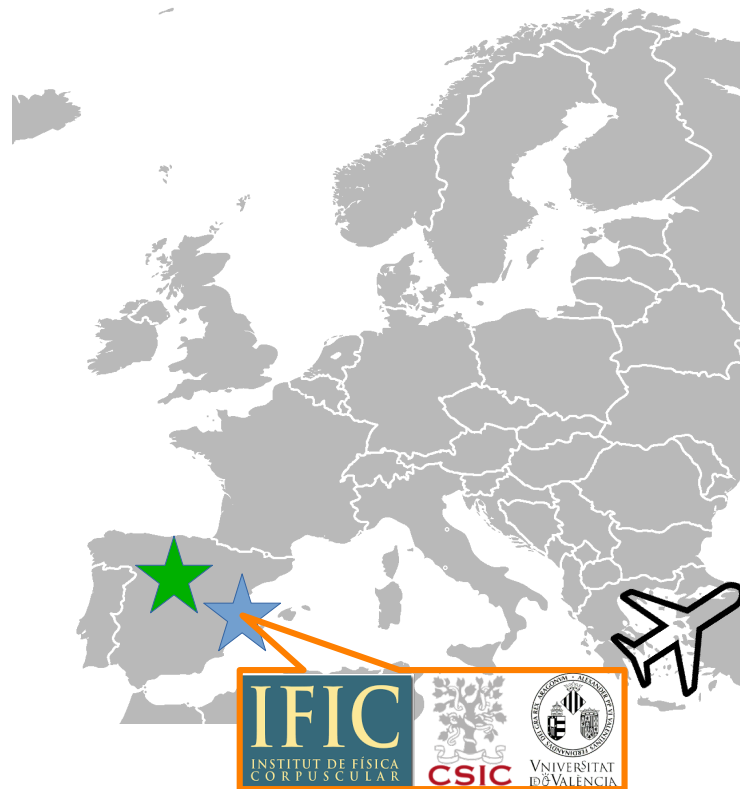
★ 5 yr physics deg. (Valladolid/Zaragoza) + Master in nuclear physics (Madrid)



About me...

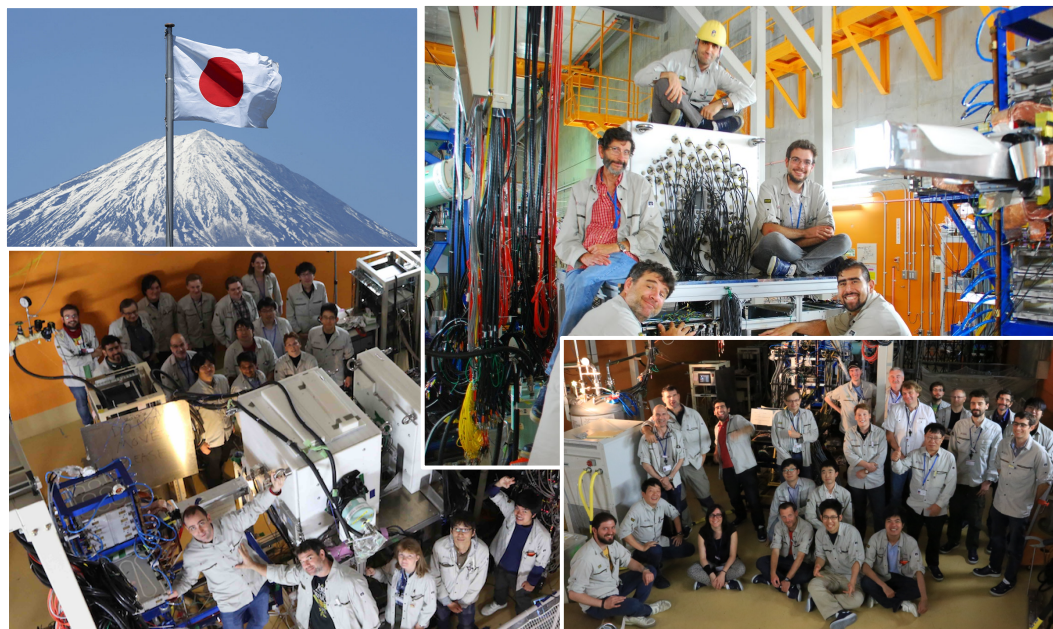
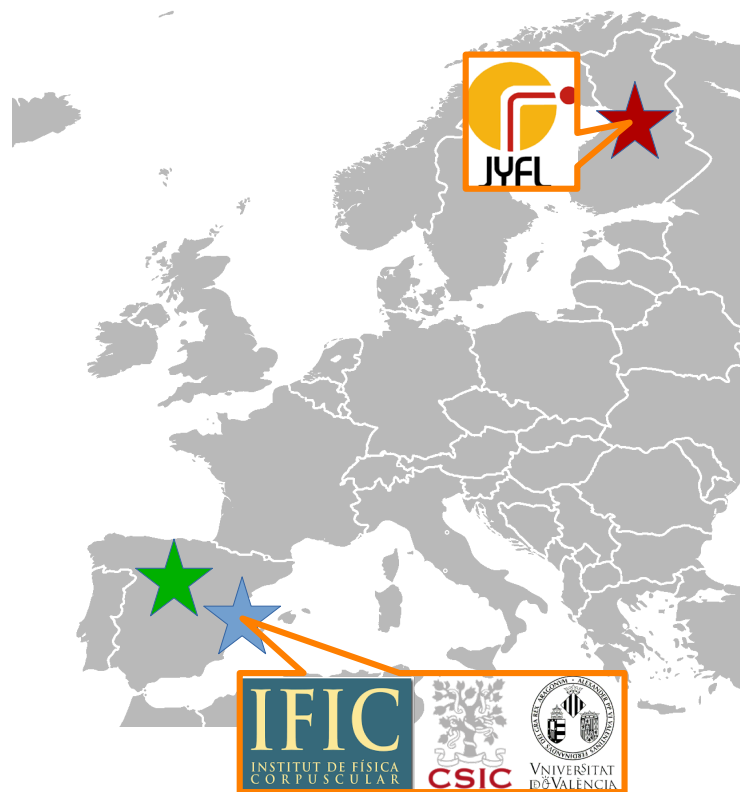


- ★ 5 yr physics deg. (Valladolid/Zaragoza) + Master in nuclear physics (Madrid)
- ★ PhD thesis about beta-delayed neutron emission within the BRIKEN project



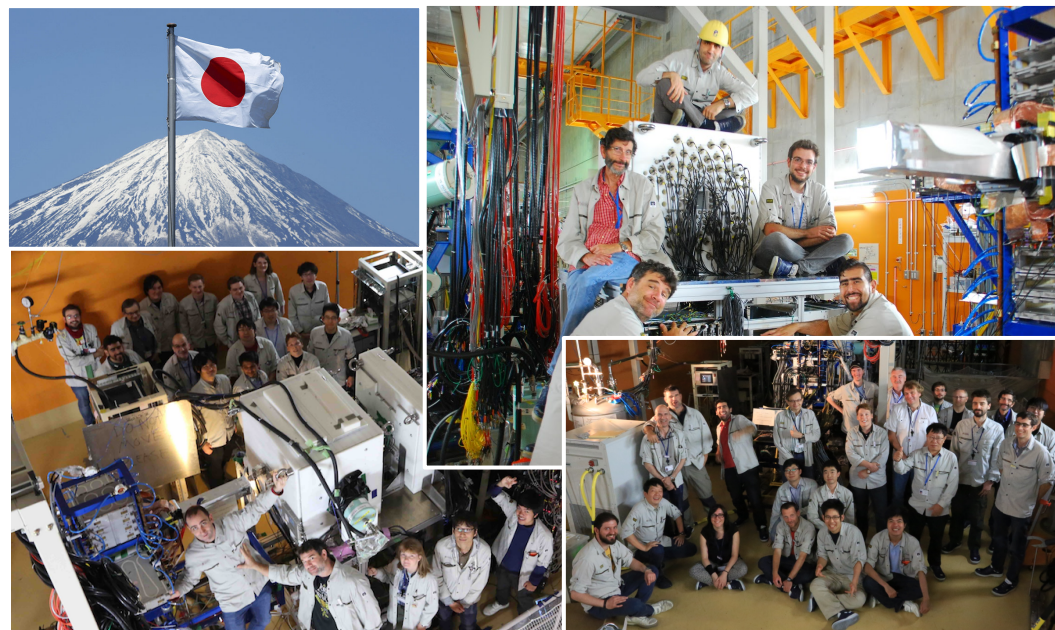
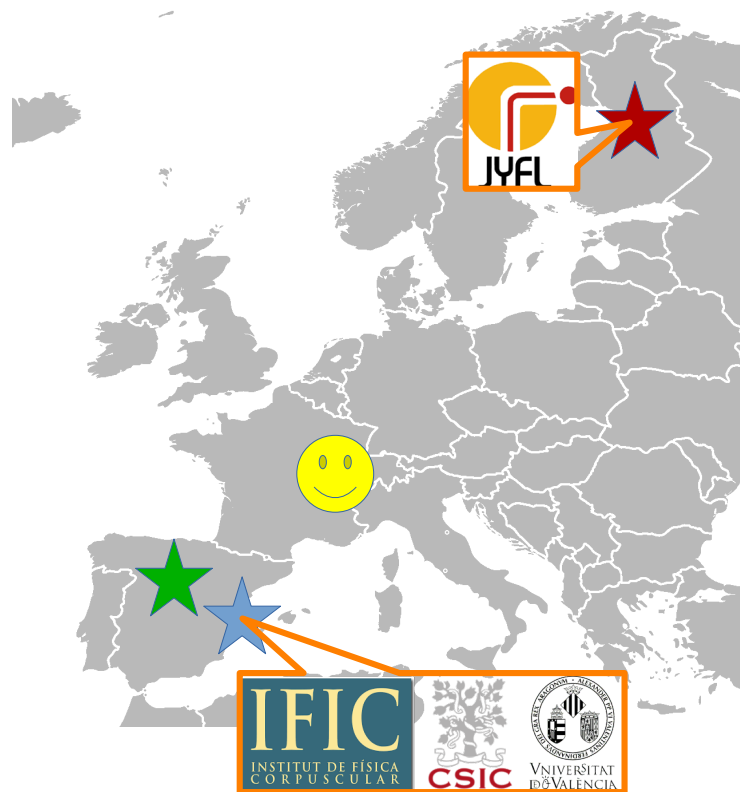
About me...

- ★ 5 yr physics deg. (Valladolid/Zaragoza) + Master in nuclear physics (Madrid)
- ★ PhD thesis about beta-delayed neutron emission within the BRIKEN project
- ★ Postdoc in JYFL accelerator laboratory as part of MARA separator group



About me...

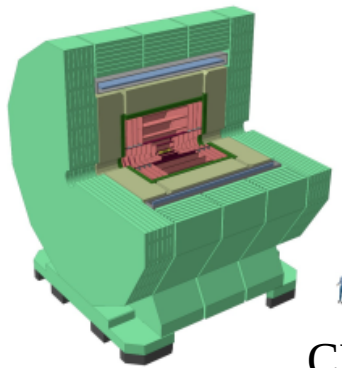
- ★ 5 yr physics deg. (Valladolid/Zaragoza) + Master in nuclear physics (Madrid)
- ★ PhD thesis about beta-delayed neutron emission within the BRIKEN project
- ★ Postdoc in JYFL accelerator laboratory as part of MARA separator group



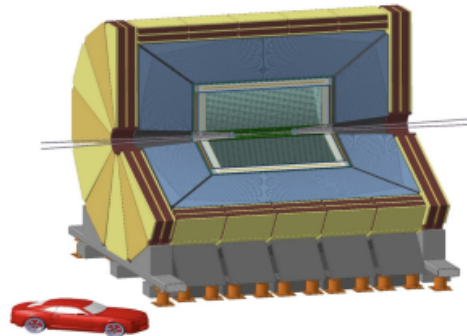
Thanks you for the warm welcome, mentoring and advice!!

Detectors for FCC

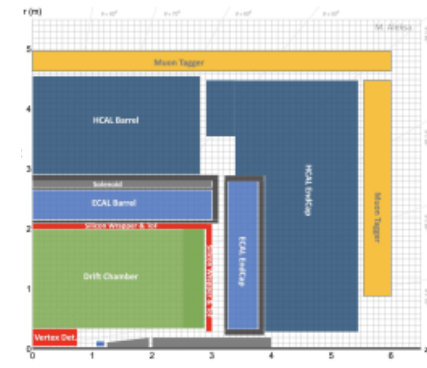
- FCC detector concepts are made up as combinations of different subdetectors
- These combinations are not fixed yet, the subdetectors may be interchangeable for dedicated studies



CLD



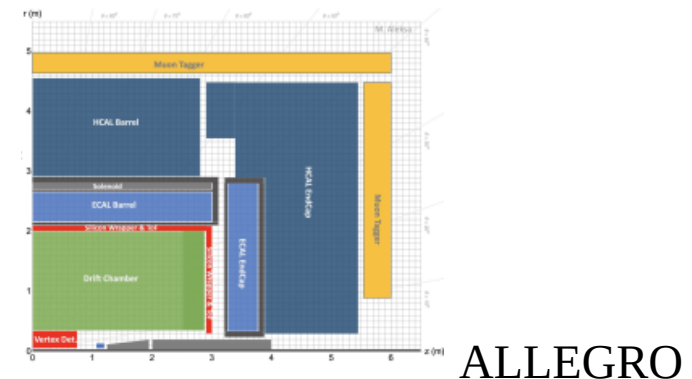
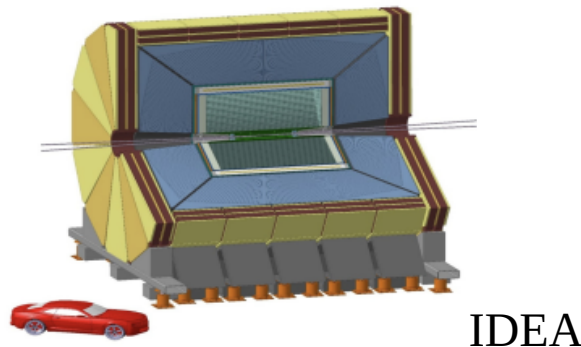
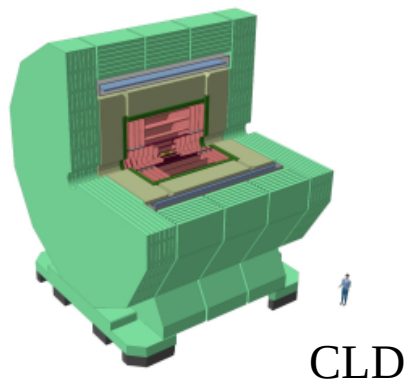
IDEA



ALLEGRO

Detectors for FCC

- FCC detector concepts are made up as combinations of different subdetectors
- These combinations are not fixed yet, the subdetectors may be interchangeable for dedicated studies
- Full simulation is needed to have reliable performance estimation of the different detector concepts and to evaluate how each subdetector will integrate with the others (e.g. the effect of the additional material budget)
- DD4hep was chosen as framework to describe the detectors for FCC, and me to work on it





Detectors for FCC. DD4hep

- DD4hep is used for detector description, physics simulation and event reconstruction
- It provides a modular, well structured and flexible way of describing the detector geometry, materials, visualization, and some data needed by Geant4 simulation (readout, segmentation, magnetic field, regions and limits)

Detectors for FCC. DD4hep

- DD4hep is used for detector description, physics simulation and event reconstruction
- It provides a modular, well structured and flexible way of describing the detector geometry, materials, visualization, and some data needed by Geant4 simulation (readout, segmentation, magnetic field, regions and limits)
- Typically subdetectors are described in a dedicated XML file, and the full detector is made up by referencing the subdetector files of interest

Detectors for FCC. DD4hep

- DD4hep is used for detector description, physics simulation and event reconstruction
- It provides a modular, well structured and flexible way of describing the detector geometry, materials, visualization, and some data needed by Geant4 simulation (readout, segmentation, magnetic field, regions and limits)
- Typically subdetectors are described in a dedicated XML file, and the full detector is made up by referencing the subdetector files of interest
- ddsim is an interface to run the physics simulation based on Geant4. Some Geant4 hooks may be used via DD4hep plugin mechanism (e.g., sensitive detector action)

Detectors for FCC. DD4hep

- DD4hep is used for detector description, physics simulation and event reconstruction
- It provides a modular, well structured and flexible way of describing the detector geometry, materials, visualization, and some data needed by Geant4 simulation (readout, segmentation, magnetic field, regions and limits)
- Typically subdetectors are described in a dedicated XML file, and the full detector is made up by referencing the subdetector files of interest
- ddsim is an interface to run the physics simulation based on Geant4. Some Geant4 hooks may be used via DD4hep plugin mechanism (e.g., sensitive detector action)
- A useful feature of DD4hep (DDRec) are the so-called data extensions: custom C++ classes attached to a subdetector, intended to store information about the geometry, to assist or replace the geometry navigation during the reconstruction

Detectors for FCC. DD4hep

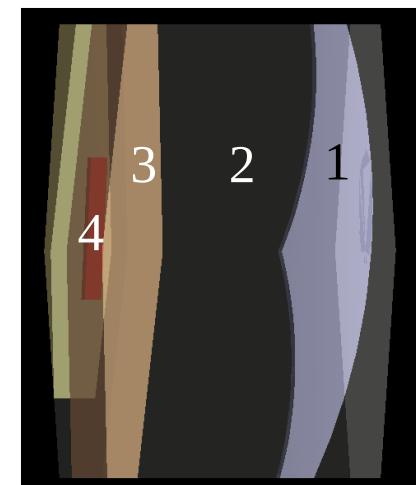
- DD4hep is used for detector description, physics simulation and event reconstruction
- It provides a modular, well structured and flexible way of describing the detector geometry, materials, visualization, and some data needed by Geant4 simulation (readout, segmentation, magnetic field, regions and limits)
- Typically subdetectors are described in a dedicated XML file, and the full detector is made up by referencing the subdetector files of interest
- ddsim is an interface to run the physics simulation based on Geant4. Some Geant4 hooks may be used via DD4hep plugin mechanism (e.g., sensitive detector action)
- A useful feature of DD4hep (DDRec) are the so-called data extensions: custom C++ classes attached to a subdetector, intended to store information about the geometry, to assist or replace the geometry navigation during the reconstruction
- DD4hep provides dedicated tools based on ROOT/Geant4 for visualization, material scan, geometry scan, material budget estimation...

Detectors for FCC: ARC

- The Array of Rich Cells (ARC) was designed by R. Forty, G. Wilkinson and M. Tat, to provide PID at high momentum with little additional material budget (5% X_0)
- The detector geometry, material description (including optical properties) and sensor readout is fully implemented in DD4hep framework

Detectors for FCC: ARC

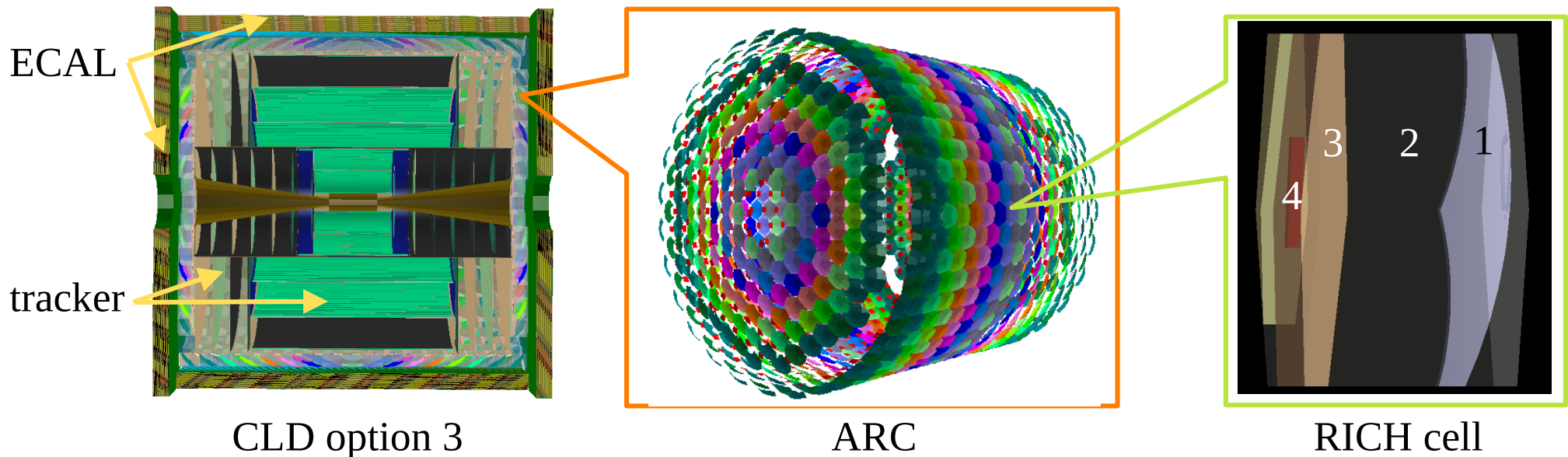
- The Array of Rich Cells (ARC) was designed by R. Forty, G. Wilkinson and M. Tat, to provide PID at high momentum with little additional material budget (5% X_0)
- The detector geometry, material description (including optical properties) and sensor readout is fully implemented in DD4hep framework
- Each RICH cell contains an spherical mirror (1) which focus the light produced in the two Cerenkov radiators (2,3) into a light sensor (4)



RICH cell

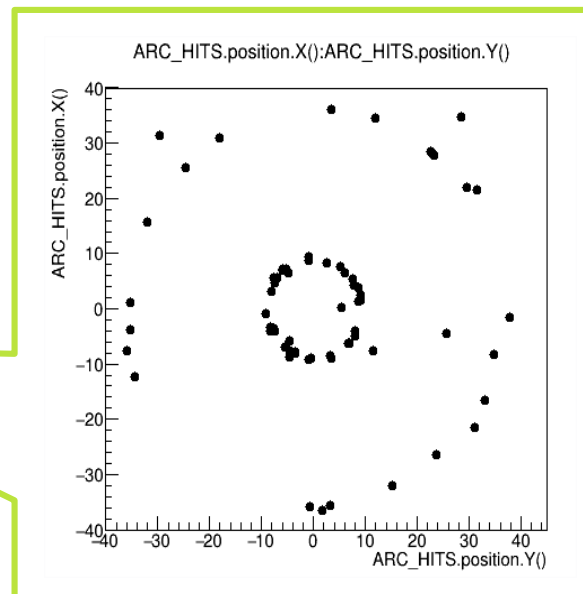
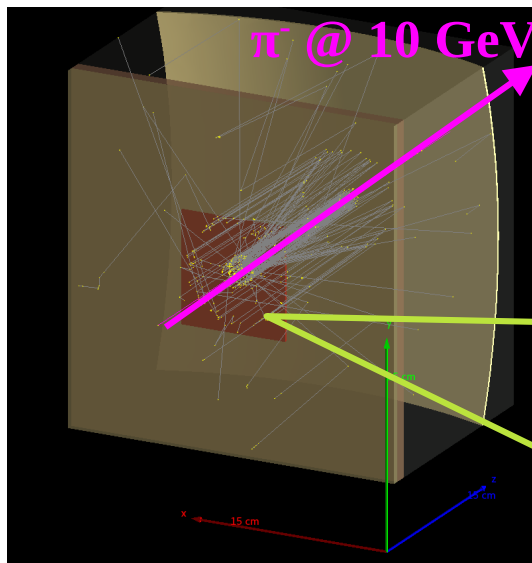
Detectors for FCC: ARC

- The Array of Rich Cells (ARC) was designed by R. Forty, G. Wilkinson and M. Tat, to provide PID at high momentum with little additional material budget (5% X_0)
- The detector geometry, material description (including optical properties) and sensor readout is fully implemented in DD4hep framework
- Each RICH cell contains an spherical mirror (1) which focus the light produced in the two Cerenkov radiators (2,3) into a light sensor (4)
- The **ARC** consist of an large array of **RICH cells** placed as in the picture below (only mirrors and sensors are visible for simplicity)



Detectors for FCC. ARC geometry

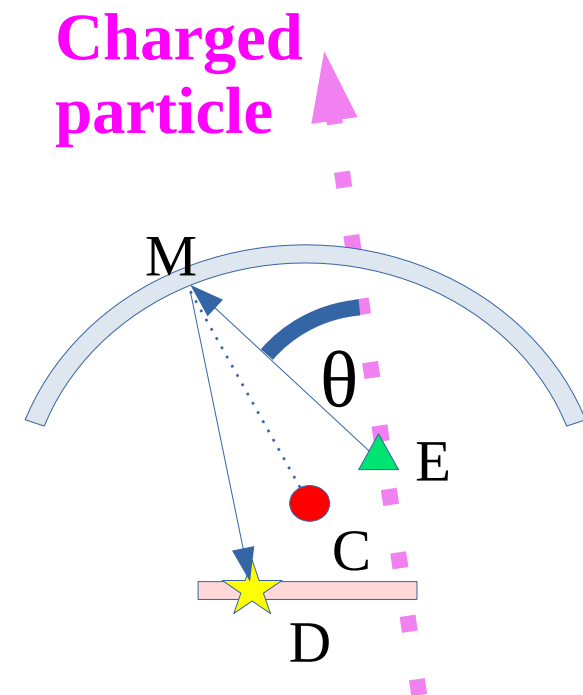
- Implementing this detector was challenging by different reasons:
 - ✓ Complex geometry: 5 parameters fix the position of mirror and sensor inside each cell.
 - ✓ The shape of the cell require a tessellated solid shape, buggy in VecGeom. During the adventure of debugging I learnt a lot about DD4hep, Geant4 and worthy experiences...
 - ✓ Optical properties of the Cherenkov radiators are critical for simulating the performance of the detector, understanding and calculating them is not trivial
 - ✓ CLD outer tracker geometry had to be modified to fit the ARC
- But the support I received was great and **I am grateful to all the people that helped**



After implementing the geometry, we have to develop an algorithm that works out the PID out of the ARC hits and the reconstructed track

Detectors for FCC. ARC reconstruction

- There are several reconstruction algorithms to extract the PID
- These algorithms work on the parameter space of angles theta/phi with respect to the trajectory of the particle crossing the RICH cell
- An inverse ray-tracing calculation is used to convert the hit positions of the Cherenkov photons in the ARC sensor into these theta/phi angles
- The position and properties of the elements of the RICH cells are needed for this step
- DD4hep allow to retrieve the needed information by quickly navigating the Detector Element tree (see next slide)



Example showing how to retrieve geometrical properties of a mirror

```
int ncell = 1, reflected = 0, nphi = 1; // based on hit cellID information

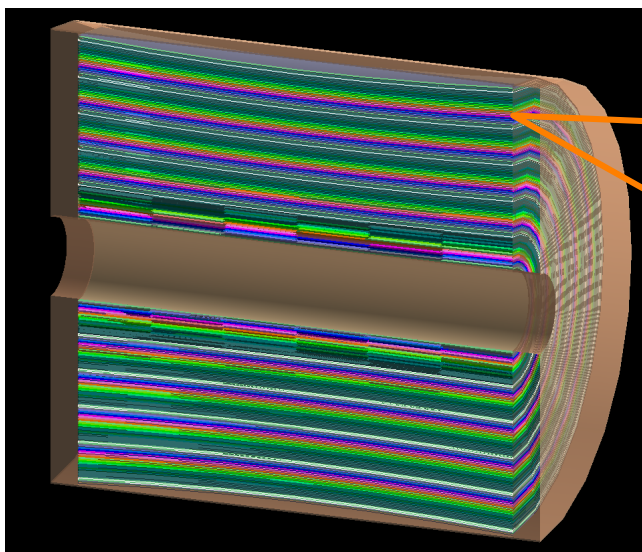
// retrieve pointer to the mirror shape
auto barrelDE = description->detector("ARCBARREL");
auto cellDE_name = Form("ARCBARREL_cell%d_ref%d_phi%dDE", ncell, reflected, nphi);
auto cellDE = barrelDE.child( cellDE_name );
auto mirrorDE_name = Form("ARCBARREL_mirror%d_ref%d_phi%dDE", ncell, reflected, nphi);
auto mirrorDE = cellDE.child( mirrorDE_name );
auto mirrorSolid = mirrorDE.solid();
auto m = (TGeoCompositeShape*)mirrorSolid.access();

// Retrieve the center of the sphere
//-- this is just the matrix for building the intersection, translation
auto matrix_boolean = m->GetBoolNode()->GetRightMatrix();
//-- matrix for placing the cell
auto matrix_cellvol = cellDE.nominal().worldTransformation();
double local_coord [3] = {0.,0.,0.};
double global_coord[3] = {0.,0.,0.};
(matrix_cellvol**matrix_boolean).LocalToMaster(local_coord, global_coord);
// global_coord contains now the XYZ global coordinates of the center of the mirror

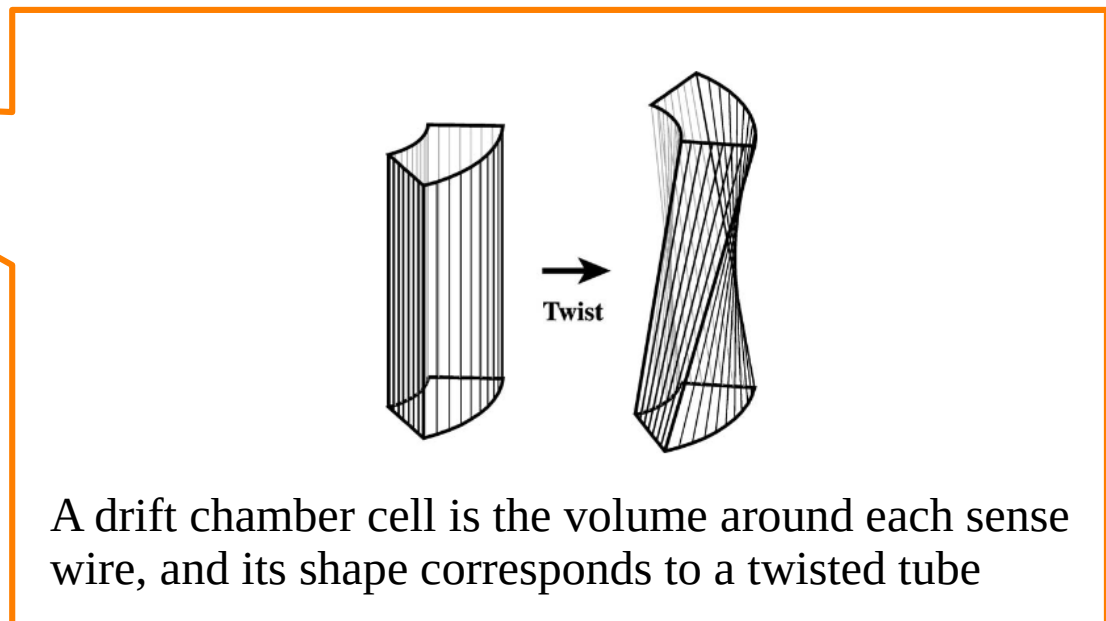
// Retrieve the inner radius of the spherical mirror
TGeoSphere * spherical_mirror_shape = (TGeoSphere*)m->GetBoolNode()->GetRightShape()
double mirror_r_min = spherical_mirror_shape->GetRmin();
```

Detectors for FCC: Drift chamber

- The drift-chamber is a very transparent ($1\% X_0$) tracker developed for the IDEA detector
- A drift chamber is a gaseous detector working on proportional mode, capable of resolving spatial coordinates of an ionizing event by measuring the electron drift time.
- This detector is characterized by around ~ 60000 sense wires distributed among 112 layers
- The sense wires in each layer define an hyperboloidal surface

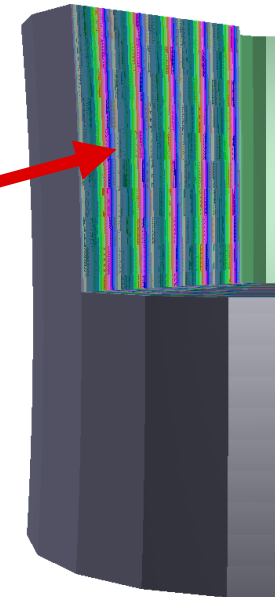


Sagittal plane of the drift chamber
Each color represents a layer







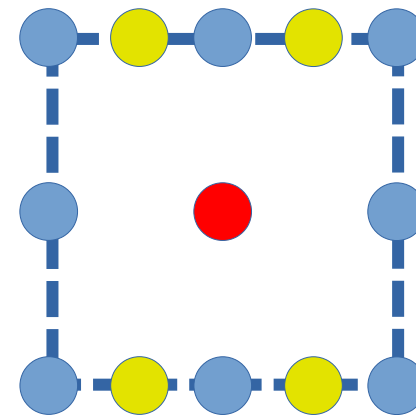
Detectors for FCC. Drift chamber geometry

- Cylindrical wall made of carbon fiber
- Cylindrical volume made of gas mix (He-Isobutane)
- Hyperboloidal layers (x112) made of gas mix
 - Cells are twisted tubes (twisted tube results from layer segmentation in phi, keeping the twist angle), made of gas mix. These cells are the sensitive volumes!
 - Field (x5) and sense (x1) wires inside each cell

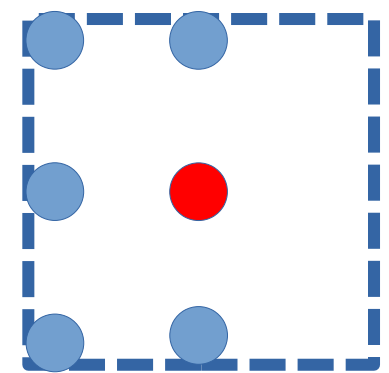


Distribution of wires is slightly modified to fit completely inside the cell:

-  Cross section of cell (twisted tube)
-  Field wire
-  Sense wire
-  Field wire of next/prev. layer cell



Original idea



Implementation

Detectors for FCC. Drift chamber geometry

- Few global parameters:
 - ✓ Inner/Outermost radius and z-length parameters are placed in a dedicated compact file **DetDimensions.xml**
 - ✓ Parameters related to internal parts of the detectors placed in subdetector compact file **DriftChamber_o1_v02.xml**

```

<!-- %%%%%%%%%          Central Drift Chamber Basic Parameters          %%%%%%%%% -->
<!-- %%%%%%%%%          based on the geometry version "IDEA231026"      %%%%%%%%% -->
<define>
<!-- Drift Chamber parameters      -->
<!-- Gas and vessel geometry parameters moved to DetDimensions file  -->
<!-- <constant name="DCH_inner_cyl_R_total"          value=" 349 * mm " /> -->
<!-- <constant name="DCH_outer_cyl_R_total"          value=" 2001 * mm " /> -->
<!-- <constant name="DCH_half_length_total"          value=" 2250 * mm " /> -->

<!-- Gas (active volume) geometry      -->
<constant name="DCH_gas_inner_cyl_R"      value=" 350 * mm " />
<constant name="DCH_gas_outer_cyl_R"      value=" 2000 * mm " />
<constant name="DCH_gas_Lhalf"            value=" 2000 * mm " />

<!-- Vessel cylinder surrounds the gas cylinder in radius and z      -->
<constant name="DCH_vessel_thickness"      value=" DCH_gas_inner_cyl_R - DCH_inner_cyl_R_total"

<!-- Service components fill the space from z=2m to z=2.25m      -->
<constant name="DCH_services_zmin"        value="DCH_gas_Lhalf + DCH_vessel_thickness" />
<constant name="DCH_services_zmax"        value="DCH_half_length_total" />

<!-- Position of guard wires      -->

```

Detectors for FCC. Drift chamber geometry

- We call the detector constructor inside the dedicated XML file as follows:

```

<detectors>
  <detector
    id="DetID_DCH"
    name="DCH_o2"
    type="DriftChamber_o1_v02_T"
    readout="DCHCollection"
    region="DCH_region" } Needed for PAI model
    limits="DCH_limits" } Useful when developing
    printExcelTable="True"
  >
  <!-- /detectors/detector/vessel -->
  <vessel
    material="CarbonFibStr"
    vis="dch_vessel_vis"
  >
</vessel>
  <!-- /detectors/detector/gas -->
  <gas
    material="GasHe_90Isob_10"
    vis="dch_gas_vis"
  >
</gas>
  <!-- /detectors/detector/wires -->
  <wires
    vis="dch_no_vis_nodaughters" } Make wires invisible for fast visualization
    buildSenseWires="True" } Construction of wires can be switch off
    buildFieldWires="True"
    SWire_thickness = "DCH_SWire_thickness" } Default is too small to be seen by eye,
    FSideWire_thickness = "DCH_FSideWire_thickness" } make them bigger for visualization
    FCentralWire_thickness="DCH_FCentralWire_thickness"
    SWire_material = "W" } Definition of material per type of wire
    FSideWire_material = "Al"
    FCentralWire_material="Al"
  >
</wires>
</detector>
</detectors>

```

Detectors for FCC. Drift chamber data extension

- A data extension is a custom C++ class that may be attached to a subdetector
- Ancillary class DCH_info was developed to store global parameters and parametrization of layers, cells and wires of a generic drift chamber and some functionality regarding the parametrization of a hyperboloidal (twisted) geometry
- The information in the table below is stored as an object DCH_info, which is created when building the geometry and attached to the detector element
- This info can be accessed any time, faster than reverse-engineer the geometry!

```
Global parameters of DCH:
Gas, half length/mm = 2000
Gas, radius in/mm = 350
Gas, radius out/mm = 2000
Guard, radius in(z=0)/mm = 354
Guard, radius out(z=L/2)/mm = 1987.5

Twist angle (2*alpha) / deg = 30

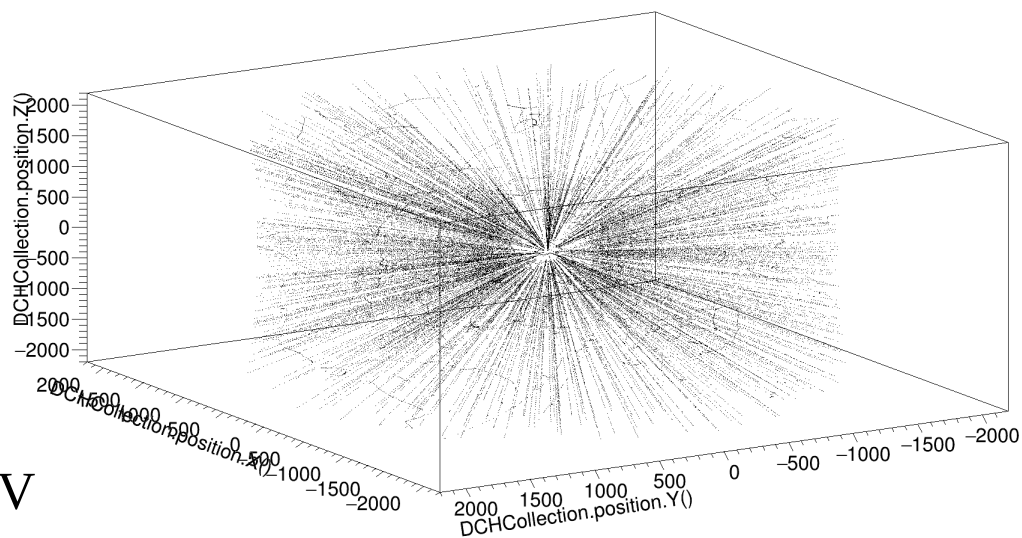
N superlayers = 14
N layers per superlayer = 8
N layers = 112

N cells layer1 = 192
N cells increment per superlayer = 48
N cells per sector = 24

Layer parameters of DCH:
layer  nwires  height_z0/mm  width_z0/mm  radius_fdw_z0/mm  radius_sw_z0/mm  radius_fuw_z0/mm
1      384      11.8464 11.8464 356.077 362      367.923 2.7766  5.92321 374.77  4004.71
2      384      12.2405 12.2405 367.923 374.043 380.164 -2.86882 6.12027 387.238 4005.02
```

Detectors for FCC. Drift chamber geometry

- Implementing this detector was relatively easy for different reasons:
 - ✓ The use of the proper shape (twisted tubes, hyperboloids) made the implementation of a complex geometry easy
 - ✓ The main reasons it was a straightforward work were:
 - Existence of a scientific paper describing the math and behavior of the twisted tube. **Please, publish papers** about technical developments, it is useful!
 - Clear and systematic prescription of the geometry
 - Great help from software and detector developers



Hits by 1k muons @ 10GeV

- I am enjoying working for software and FCC
 - ✓ mainly because of the people!
 - ✓ Incredible resources at CERN (e.g., recording of courses)
- Working on detector simulations is stimulating, and make me learn about physics, geometry, and general software (good practices, how to handle big projects, etc)
- I am willing to keep pushing the full simulation of FCC detectors
- I will keep learning about HEP and software
 - ✓ I am grateful to all the people that has guided me with remarkable patience
- Happy to contribute to the Geant4, DD4hep and ROOT projects (user forums, github, bugzilla, hackatons, meetings, and sometimes even pull requests)
- I am also guide at CERN, please let us know if someone is interested in a visit!