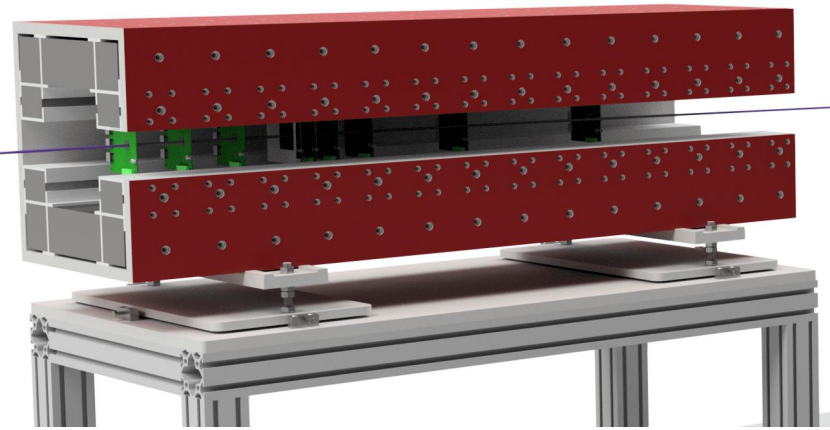
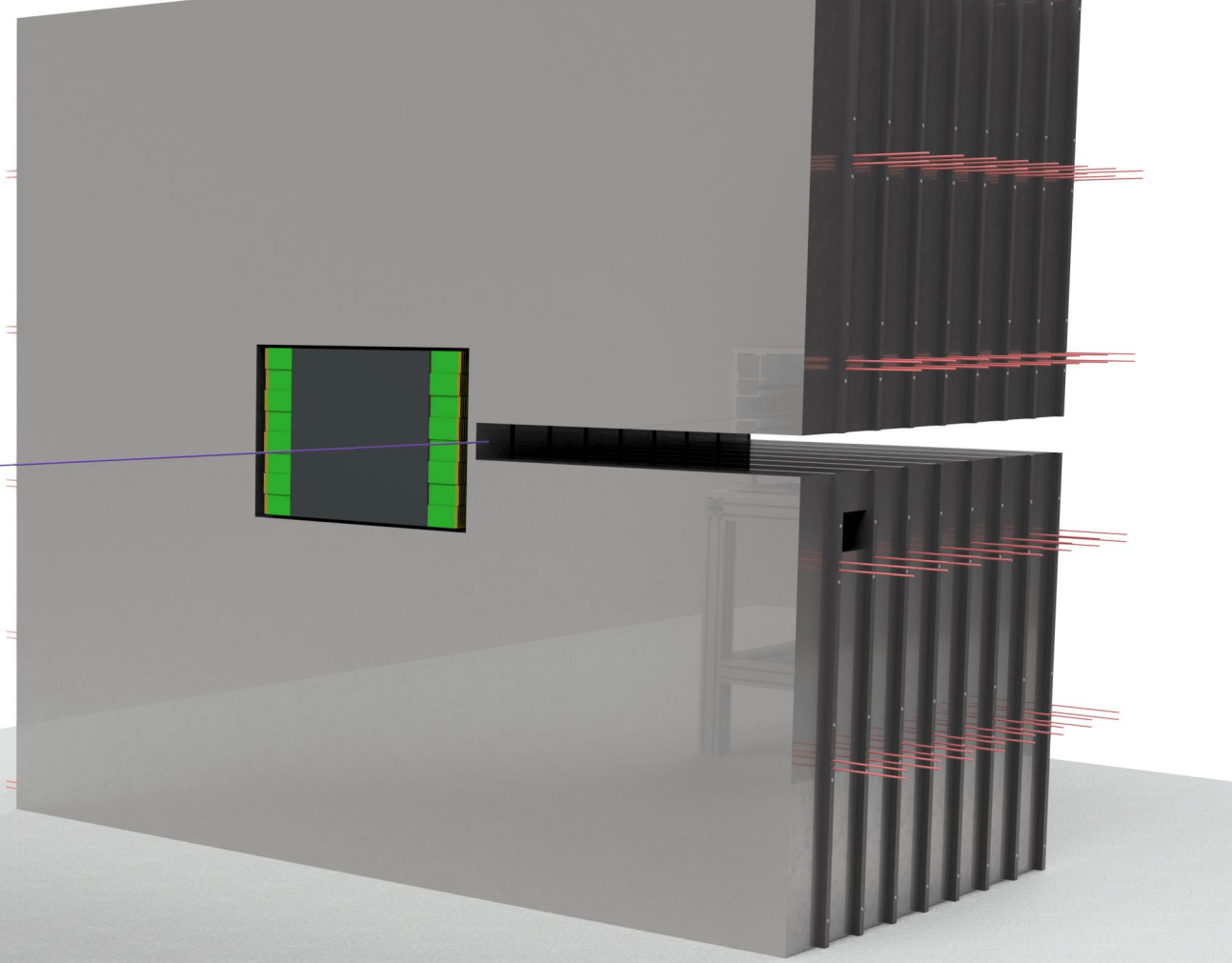
UNIVERSITÄT BONN





LOHENGRIN MEETING

UPDATES: SIMUATION FRAMEWORK PERCEVAL

Cedric Breuning – 24.07.2025

PERCEVAL

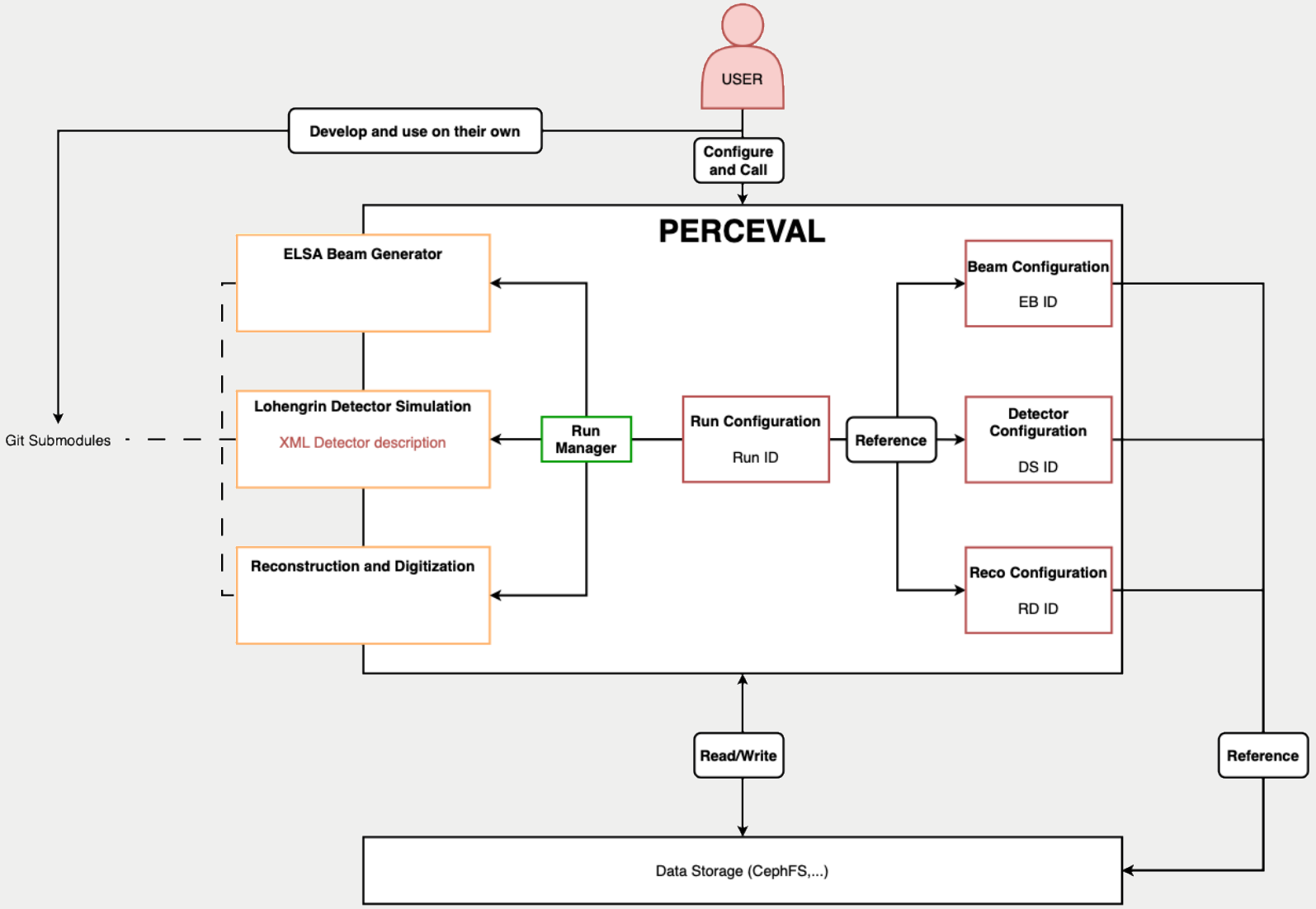
- Software framework for simulation of the experiment
  - DD4hep/key4hep
  - Some custom tools
  - Easy configuration of simulation parameters (detectors, …) + Documentation
  - A beautiful acronym. Current Workingtitle:

  Phenomenological **E**vent **R**econstruction and **C**reation for the **EVA**luation of **L**ohengrin
  - Look at it on GitLab

MODULARITY, FLEXIBILITY AND EASY TO USE

- Flexible detector usage with DD4hep
- Modularity:
  - Each Module (Beam, Detector, Reco) can be used on its own → Git Submodules
  - Modular detector system
  - Only run the necessary modules
- Easy to use: Only configure a couple YAML/XML files. No need to go deep into the code for the „average user“

SOFTWARE ARCHITECTURE



ELSA BEAM GENERATOR

Currently very basic prototype (aka. one python script)

config.yaml

```yaml
description: |
  This is the description of the configuration. You can dump information here the user need
  The description element has no influence on the generated sample

# The ID of the EBG run. Needs to be unique and the given output directory will be checked for this ID
ebg_id: "000000"

# Number of electrons to be generated
number_of_electrons: 10000

# Output directory and format.
# Available extensions: hepevt, HEPevt
# In the future: hepmc
output_dir: "/cephfs/user/cbreunin/data/ELSA-Beam"
output_format: "hepevt"

# For developing it can be useful to overwrite existing samples.
# USE WITH CAUTION
overwrite_existing_file: false

# Energy E in GeV. Defaults to 3.2
# Includes the electron mass E^2 = m^2 + p^2
electron_energy: 3.2

# frequency of the electrons in MHz
# Defaults to 100 MHz
electron_frequency: 100

# Should the number of electrons per event be smeared with a poisson distribution
# Defaults to false
# NOT IMPLEMENTED YET
smear_electrons_per_event: false

# If the number of elecetrons per event is smear, we need an average number of electrons per event
# This value is the average number of electrons per event with the frequency defined above
# defaults to 1
average_number_of_electrons_per_event: 1

# beam spread in x and y in mm
# Defaults to 0
vertex_x_smear: 0
vertex_y_smear: 0

# Starting position in z in mm, needs to be adapted depending on your geometry in DD4HEP
# Defaults to 0
vertex_z: 0
```

```
python ebg.py <path to config>
```


```
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000000000
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000100000
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000000000
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000100000
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000000000
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000100000
1
1 11 0 0 0 0 0.0000000000 0.0000000000 3.1999999592 3.2000000000 0.0005109989 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```


UNIVERSITÄT BONN

LOHENGRIN DETECTOR SIMULATION WITH DD4HEP

Detector Description

XML Detector Description (can be changed between runs)

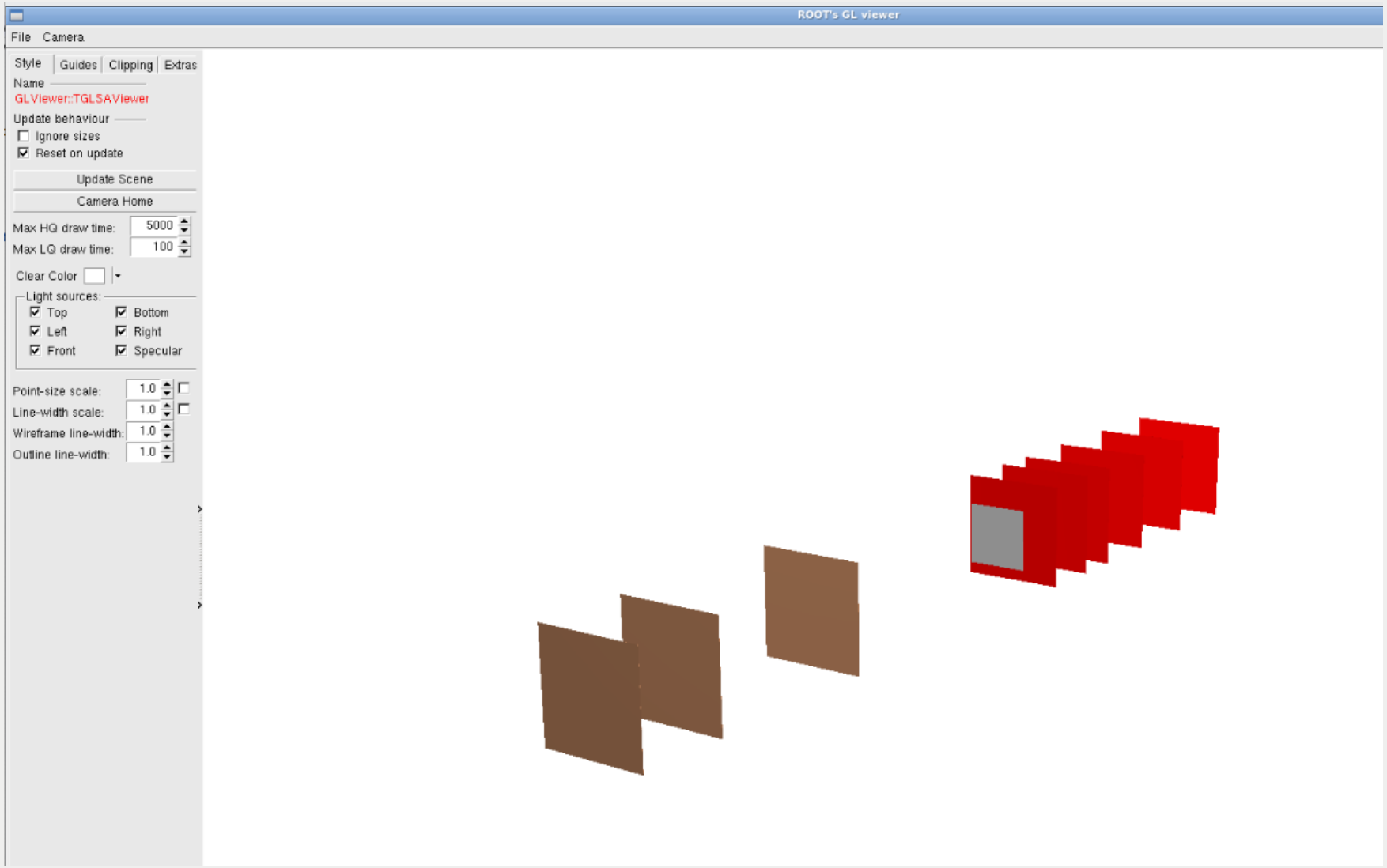
```xml
<lccdd>

    <comment> Detector definitions </comment>
    <detectors>
        <detector name="TagTracker" type="tagtracker" readout="TagTrackerHits" id="TagTrackerID">

            <comment> Definition of a pixel sensor; Currently only one Module is allowed! </comment>

            <module material="Silicon" name="PixelSensor" vis="PixelSensorVisTag" sensitive="true">
                <dimensions x="PixelSensorDimensionX" y="PixelSensorDimensionY" z="PixelSensorDimensionZ"/>
            </module>

            <comment> Definitions of the layers </comment>

            <layer id="1" module="PixelSensor" z="1810*mm"/>
            <layer id="2" module="PixelSensor" z="1840*mm"/>
            <layer id="3" module="PixelSensor" z="1900*mm"/>
        </detector>
    </detectors>

    <comment> The Readout and Cell ID. For documentation of the latter see the documentation/wiki of the detector simulation </comment>
    <readouts>
        <readout name="TagTrackerHits">
            <segmentation type="CartesianGridXY" grid_size_x="PixelPitch" grid_size_y="PixelPitch" />
            <id>system:3,layer:8,x:-16,y:-16</id>
        </readout>
    </readouts>


</lccdd>
```

+

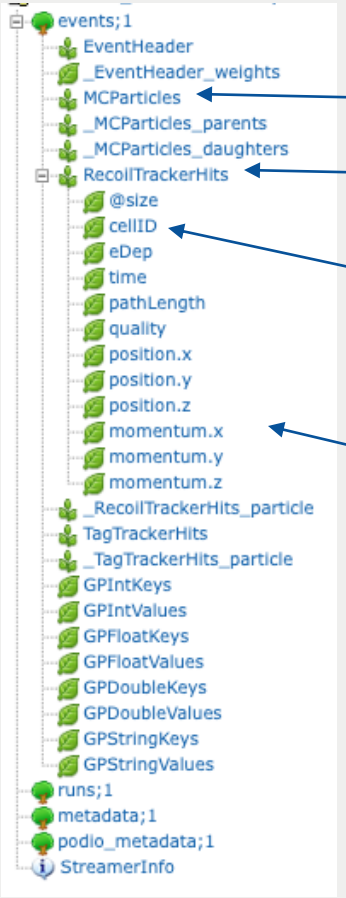
C++ Builder for each defined detector (needs to be recompiled when changed)

```cpp
// Go through all layers and create and place the volumes
for(int i = 0; i < numberOfLayers; i++) {

    // For some unknown reason it does not work, if you call this inside the Volume constructor
    string moduleMaterial = pixelSensorModule.attr<string>(_U(material));

    // Create the Module Volume
    Volume moduleLayerVolume(
        _toString(layerIDs[i], "moduleVolumeLayer%d"),
        Box(pixelSensorDimensionX / 2., pixelSensorDimensionY / 2., pixelSensorDimensionZ / 2.),
        description.material(moduleMaterial)
    );

    // Vizualisation of the modules
    moduleLayerVolume.setVisAttributes(description.visAttributes(pixelSensorModule.visStr()));

    // Place the layer (module volume) inside the tracker
    // Position: Go to the start of the tag tracker volume, calculate the offset from the layer positions and then add half
    PlacedVolume placedLayer = tagTrackerVolume.placeVolume(
        moduleLayerVolume,
        1,
        Position(0., 0., -tagTrackerLength / 2. + (layerPositions[i] - tagTrackerZStart) + pixelSensorDimensionZ / 2.)
    );

    // Set the Cell IDs for the later reconstruction and digitization
    placedLayer.addPhysVolID("system", detectorID).addPhysVolID("layer", layerIDs[i]);

    // Make the layer sensitive
    moduleLayerVolume.setSensitiveDetector(sens);

}
```

DETECTOR VISUALIZATION



OUTPUT EDM4HEP ROOT FILE



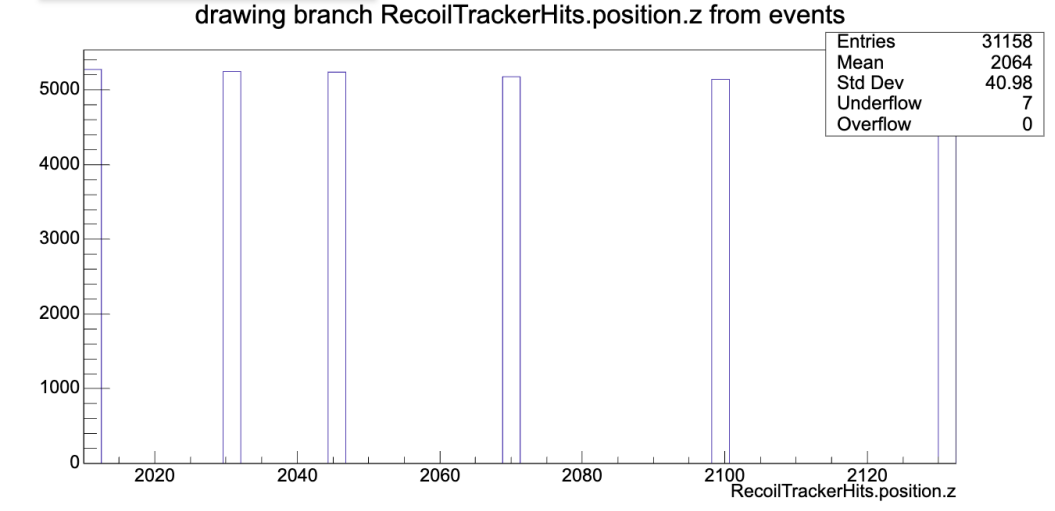
„Truth“ Information

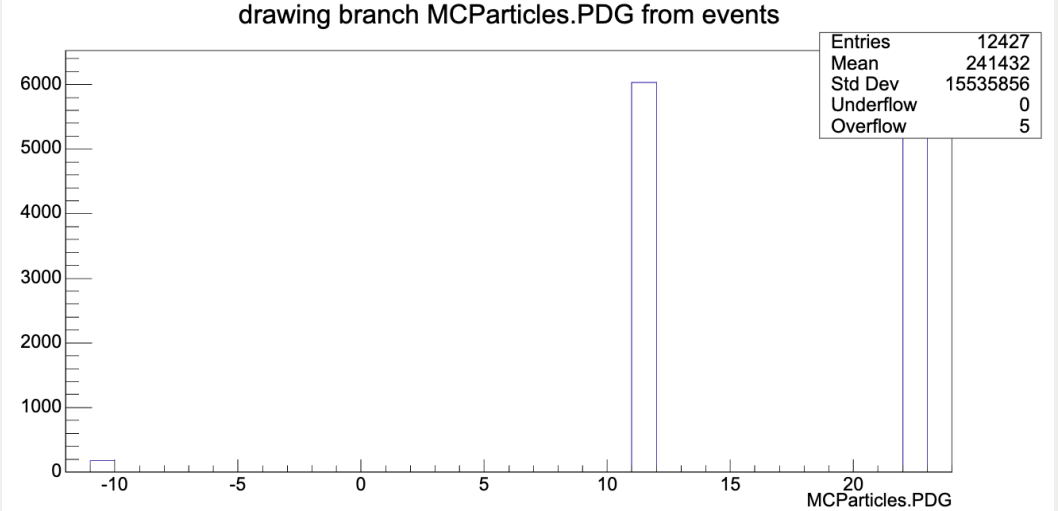
Hits for each defined sensitive detector

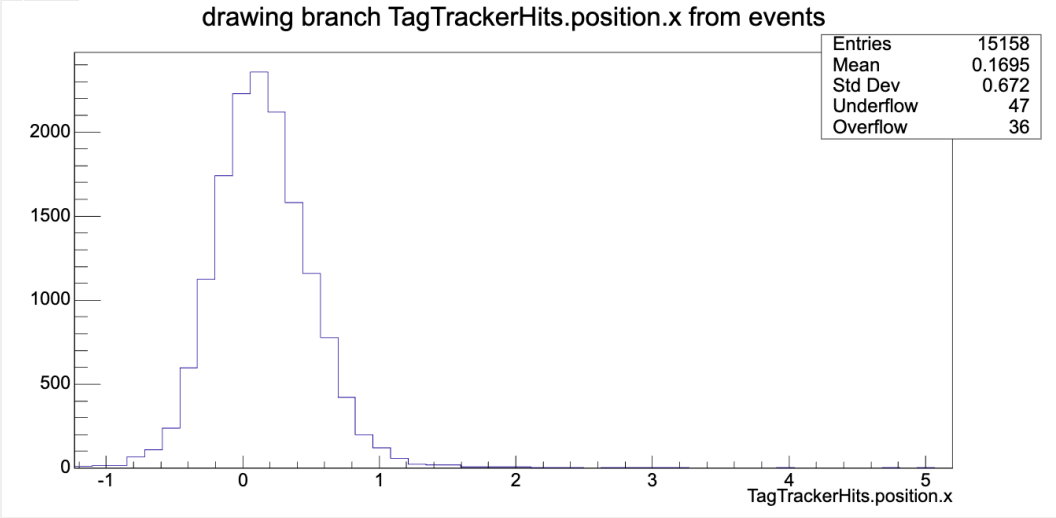
Cell ID defined in XML → ID which detector, which layer, which pixel → use for digitization, smearing,…

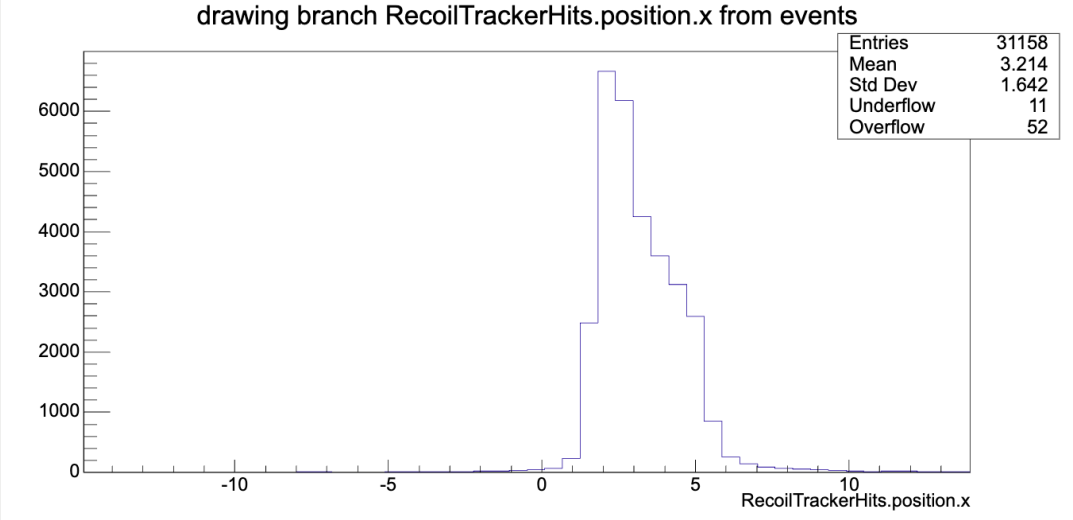
Information of the hits/energy depositions

SOME DATA


drawing branch RecoilTrackerHits.position.z from events
Entries 31158
Mean 2064
Std Dev 40.98
Underflow 7
Overflow 0
5000
4000
3000
2000
1000
0
2020
2040
2060
2080
2100
2120
RecoilTrackerHits.position.z


drawing branch MCParticles.PDG from events
Entries 12427
Mean 241432
Std Dev 15535856
Underflow 0
Overflow 5
6000
5000
4000
3000
2000
1000
0
-10
-5
0
5
10
15
20
MCParticles.PDG


drawing branch TagTrackerHits.position.x from events
Entries 15158
Mean 0.1695
Std Dev 0.672
Underflow 47
Overflow 36
2000
1500
1000
500
0
-1
0
1
2
3
4
5
TagTrackerHits.position.x


drawing branch RecoilTrackerHits.position.x from events
Entries 31158
Mean 3.214
Std Dev 1.642
Underflow 11
Overflow 52
6000
5000
4000
3000
2000
1000
0
-10
-5
0
5
10
RecoilTrackerHits.position.x