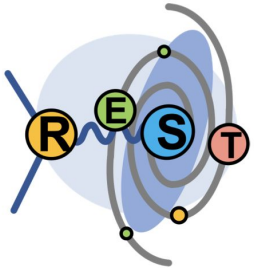


REST-for-Physics Framework

3.2 Detector event reconstruction

26.01.2023 - Cristina Margalejo Blasco - cmargalejo@unizar.es

Centro de Astropartículas y Física de Altas Energías
Universidad Zaragoza

- We will start this session with two relevant detector metadata descriptions: TRestDetectorGas (exercise 1) and TRestDetectorReadout (exercise 2).

- Entering in the domain of detectorlib.

- Hits detector processes (exercise 3).

- Generating simulated rawdata (exercise 4).

https://sultan.unizar.es/rest/classTRestDetectorGas.html

Requires to compile REST linking to Garfield libraries.
-DREST_GARFIELD=ON

A specific metadata class to generate and read gas files using Magboltz interface.

Sometime we use a centralized gas server: **https://sultan.unizar.es/gasFiles/gases.rml**. It contains pre-generated Magboltz gas files, produced using the TRestDetectorGas definition.

This metadata class allows to define an arbitrary gas mixture. It provides access to different gas properties, such as drift velocity, diffusion, townsend coefficients, etc. **TRestDetectorGas** allows to generate or read gas Magboltz files, from which the gas properties are extracted.

https://sultan.unizar.es/rest/classTRestDetectorGas.html

Requires to compile REST linking to Garfield libraries.
-DREST_GARFIELD=ON

```
//------- User : tfg204 ---- REST version : 2.2.14 --------------------------
<TRestGas name="Xenon-Ne 33Pct-Iso 0.5Pct 10-10E3Vcm" title="Xenon-Ne-Iso Mixture (33Pct Ne-0.5Pct Iso)">
  <parameter name="pressure" value="1"/>
  <parameter name="temperature" value="293.15"/>
  <parameter name="maxElectronEnergy" value="400"/>
  <parameter name="W_value" value="21.9"/>
  <parameter name="nCollisions" value="10"/>
  <eField Emin="10" Emax="1000" nodes="20"/>
  <gasComponent name="xe" fraction="0.665"/>
  <gasComponent name="ne" fraction="0.33"/>
  <gasComponent name="iC4H10" fraction="0.005"/>
</TRestGas>
```

root [0] TRestDetectorGas *gas = new TRestDetectorGas(**"server"**, "Xenon-Ne 33Pct-Iso 0.5Pct 10-10E3Vcm");
root [1] gas->SetPressure(1.) → Pressure : 1bar
root [2] gas->GetDriftVelocity(100.) → Field : 100V/cm
(double) 0.70544740 → Drift velocity : 0.7 mm/us
root [3] gas->GetTransversalDiffusion(100.)
(double) 0.073647210 → Transversal diffusion coefficient : 0.073 $cm^{-1/2}$

Time for exercise 1!

Accessing the gas properties through TRestDetectorGas

Once we are in the detectorlib domain we need to build a readout (see TRestDetectorReadout documentation) that identifies the waveform channel id with a physical detector readout channel.

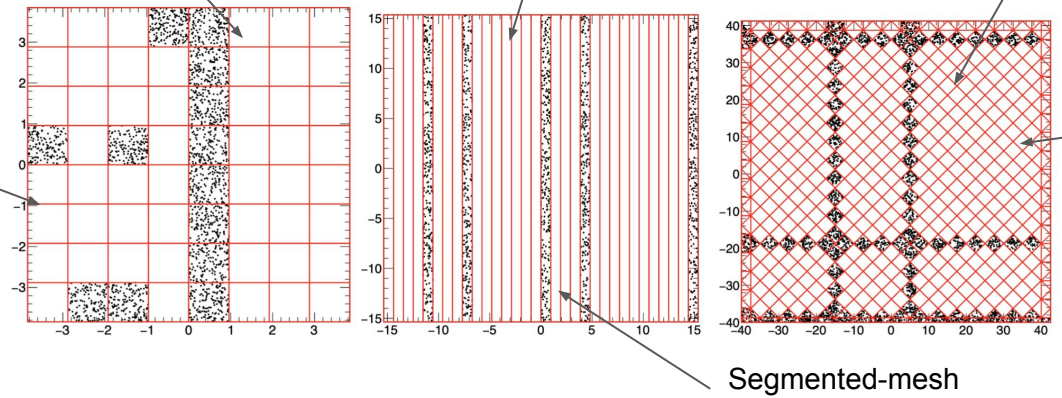
TRestDetectorReadout allows building any readout topology. See the basic-readouts examples repository.

LiquidO-like readout topology (each pixel one fiber)

One pixel per channel

Rectangular pixels

Channels with interconnected pixels



IAXO microbulk readout

Segmented-mesh microbulk

A readout is made of any number of **readout planes**, that contain any number of **readout modules**, that contain any number of **channels** that contain any number of **pixels**.

A **readout channel** is identified with the waveform. It has a channel id and a readout id → decoding.

IMPORTANT. There is a mapping that optimizes pixel identification!

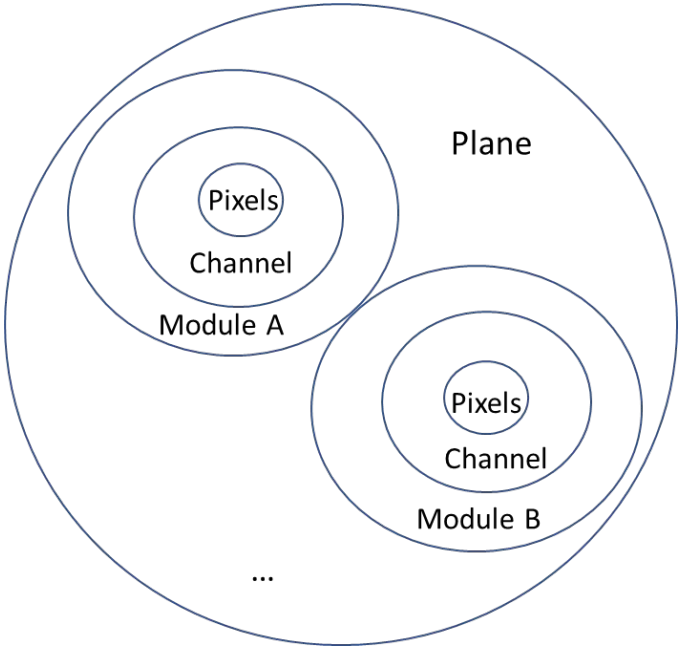
Defining a readout topology

The readout class defines the methods allowing to establish a relation between the hits inside the TPC and the signals read out by the electronics daq.

A full readout description is composed by at least one readout plane, (TRestDetectorReadoutPlane), where we can place any number of readout modules (TRestDetectorReadoutModule). A readout module is composed by readout channels (TRestDetectorReadoutChannel) which describe the basic active detection area, which can take any complex shape by combining primitive geometry elements (TRestDetectorReadoutPixel).

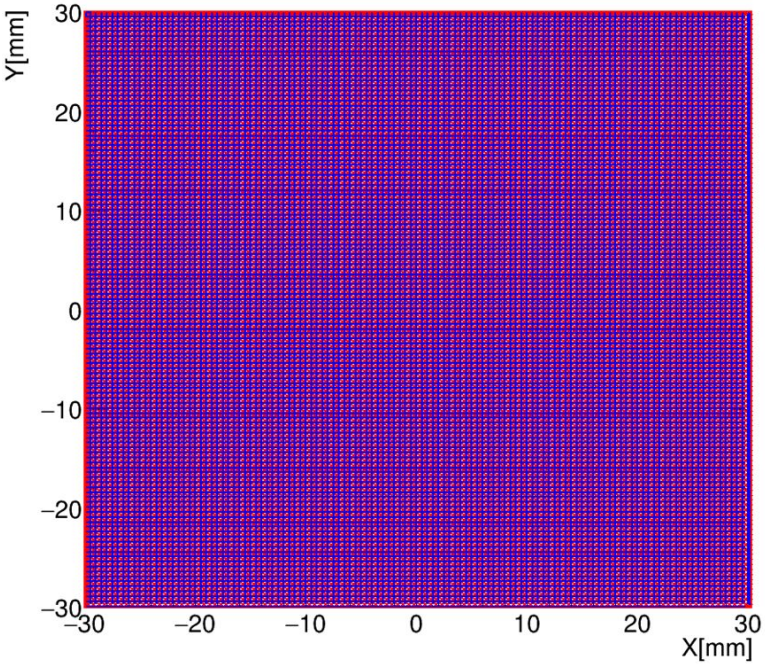
REST processes such as TRestDetectorSignalToHitsProcess use the TRestDetectorReadout class to transform the spatial coordinates into raw signal data or vice versa.

Once TRestDetectorReadout has been initialized, we can find the corresponding xy-position for a given channel inside a module contained in a readout plane.



Documentation:
https://sultan.unizar.es/rest/class/TRestDetectorReadout.html

MM detector defined with REST



https://arxiv.org/pdf/1512.09034.pdf

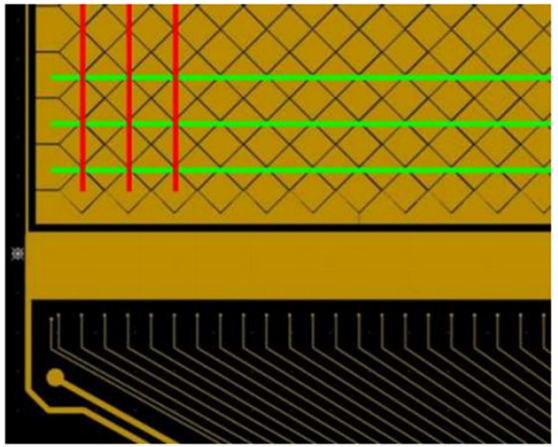
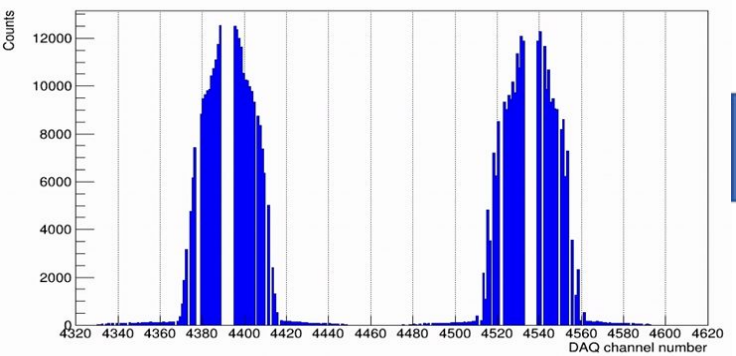


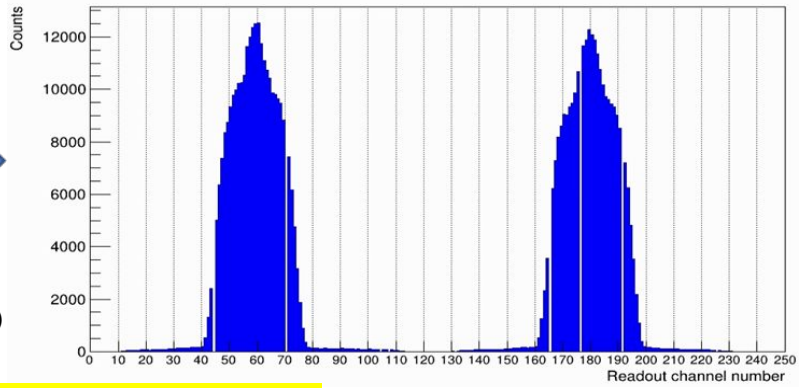
Figure 4. A detail of one of the micromegas readouts being manufactured for the PandaX-III experiment. This picture corresponding to a 3 mm pitch stripped readout. Red vertical lines and green horizontal lines are drawn to show the different pixels interconnectivity, which are readout through the same electronic channel.

TRestDetectorSignalChannelActivityProcess.cxx

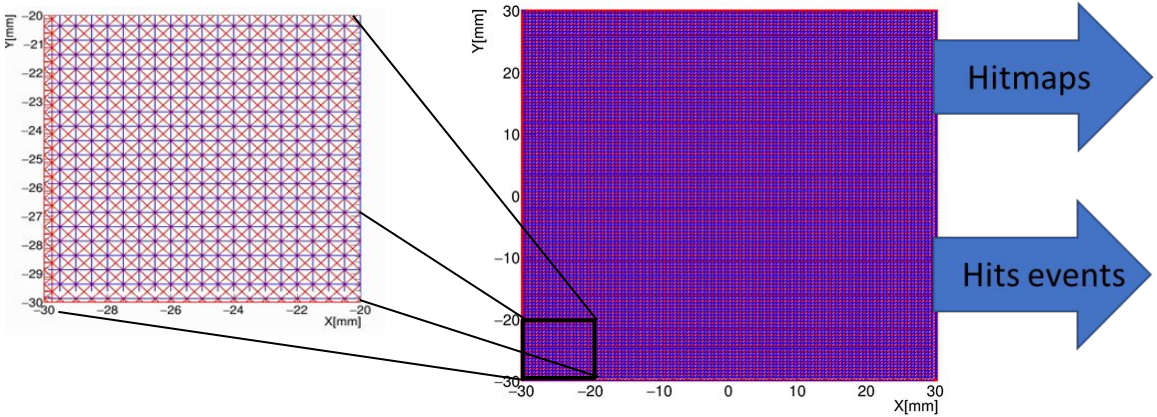


Decoding file

(DAQ to readout channel number correspondence)

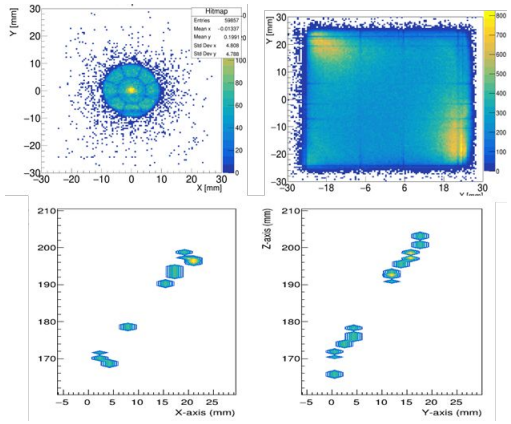


TRestDetectorSignalToHitsProcess.cxx



Hitmaps

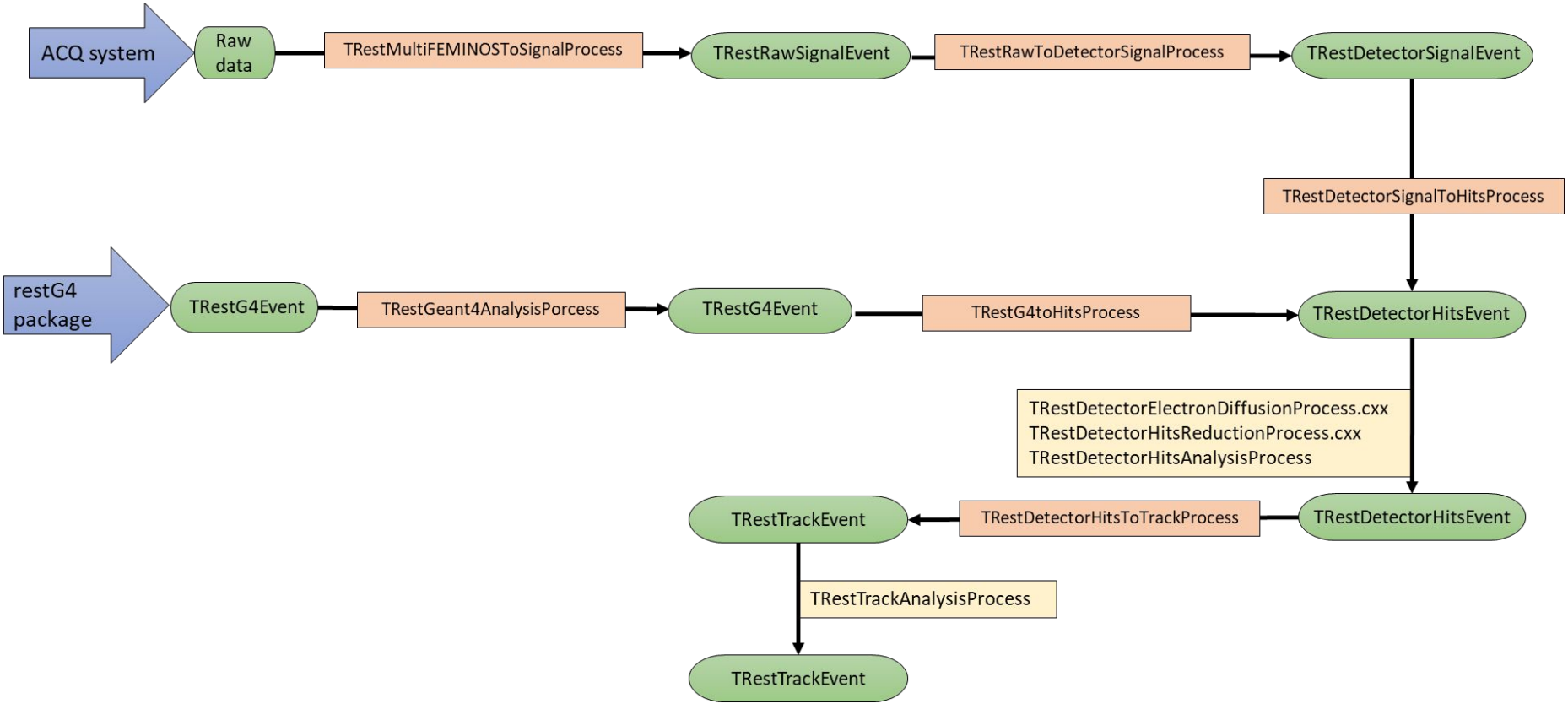
Hits events



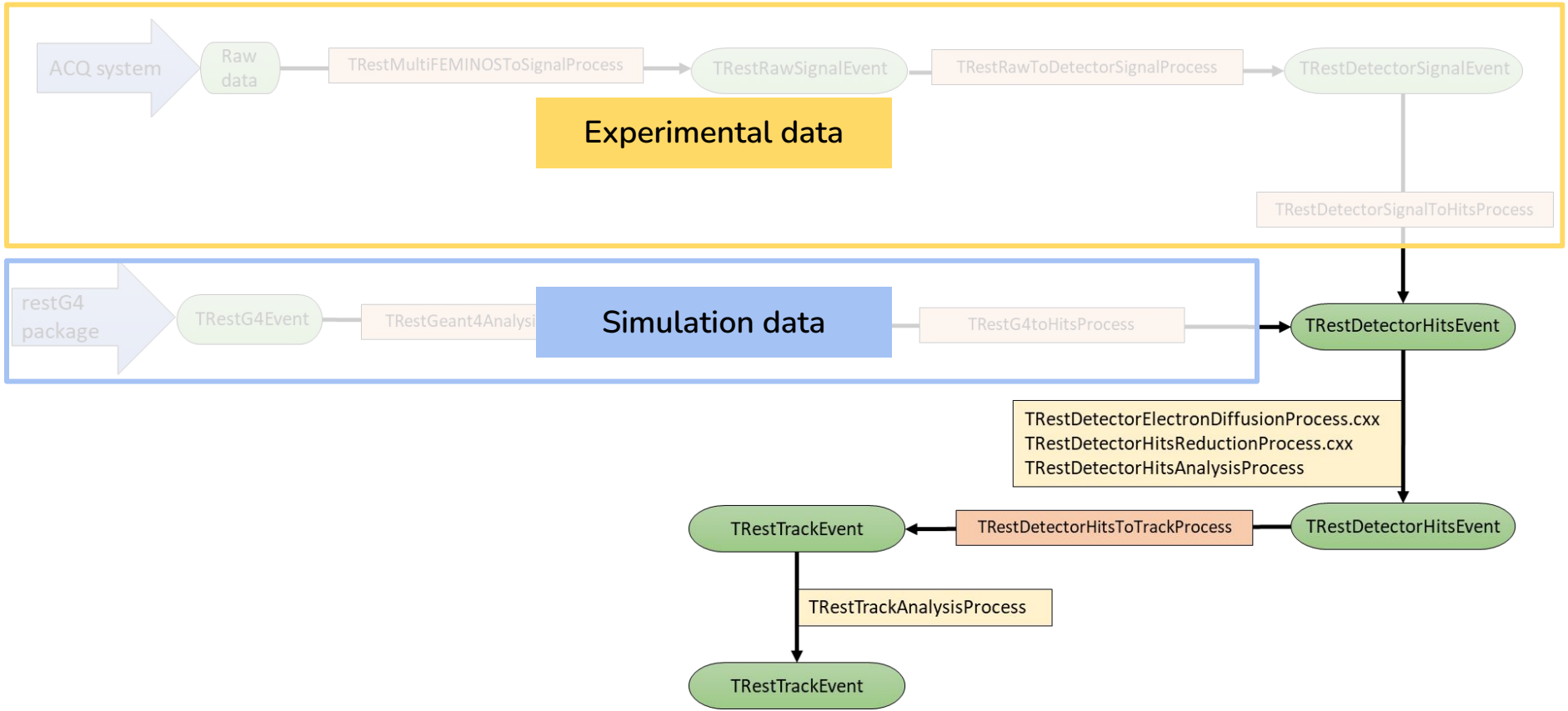
Time for exercise 2!

Generating a detector readout using a TRestDetectorReadout

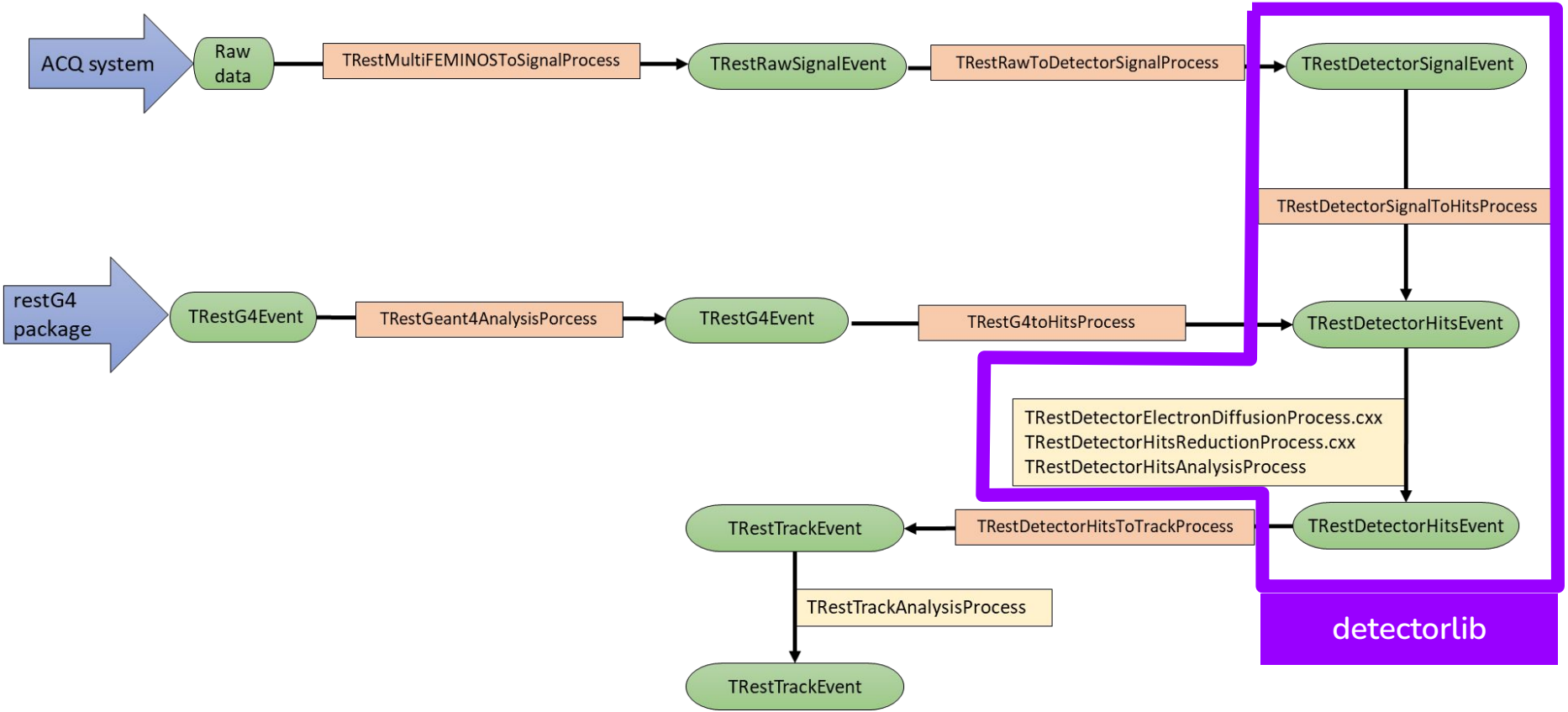
Data processing chain



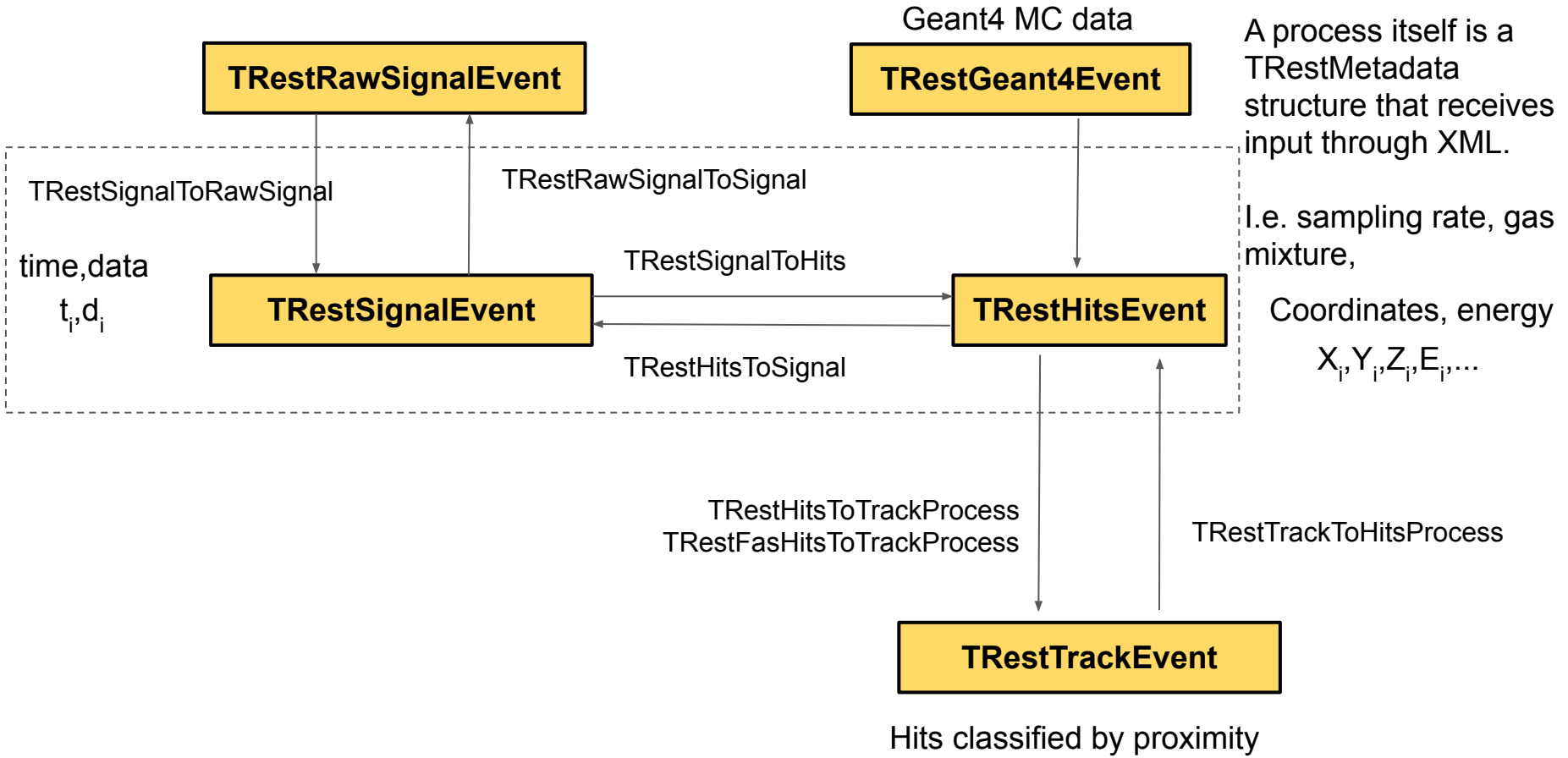
Data processing chain

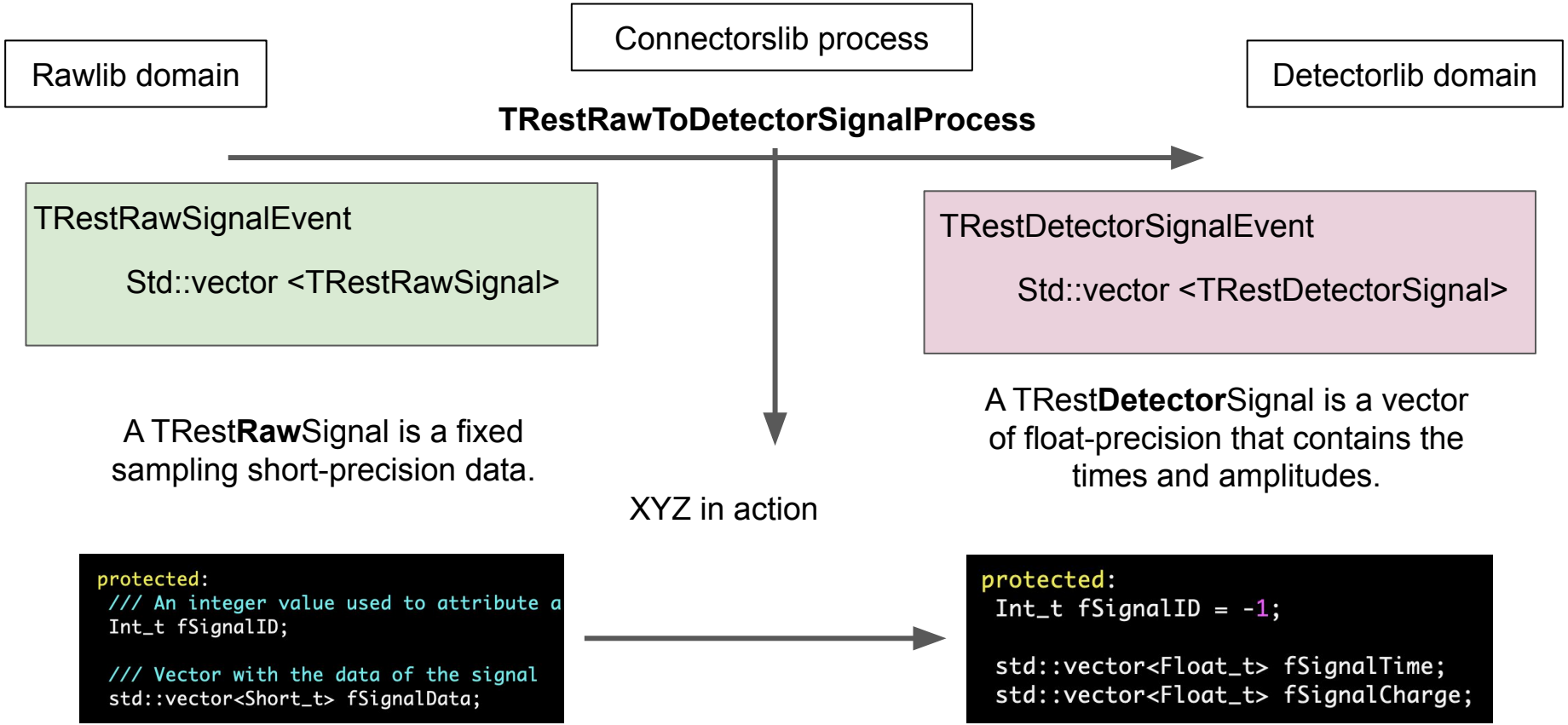


Data processing chain



Event data types and process interconnectivity




Rawlib domain
Connectorslib process
Detectorlib domain
TRestRawToDetectorSignalProcess
TRestRawSignalEvent
Std::vector <TRestRawSignal>
TRestDetectorSignalEvent
Std::vector <TRestDetectorSignal>
A TRestRawSignal is a fixed sampling short-precision data.
XYZ in action
A TRestDetectorSignal is a vector of float-precision that contains the times and amplitudes.
protected:
/// An integer value used to attribute a
Int_t fSignalID;
/// Vector with the data of the signal
std::vector<Short_t> fSignalData;
protected:
Int_t fSignalID = -1;
std::vector<Float_t> fSignalTime;
std::vector<Float_t> fSignalCharge;

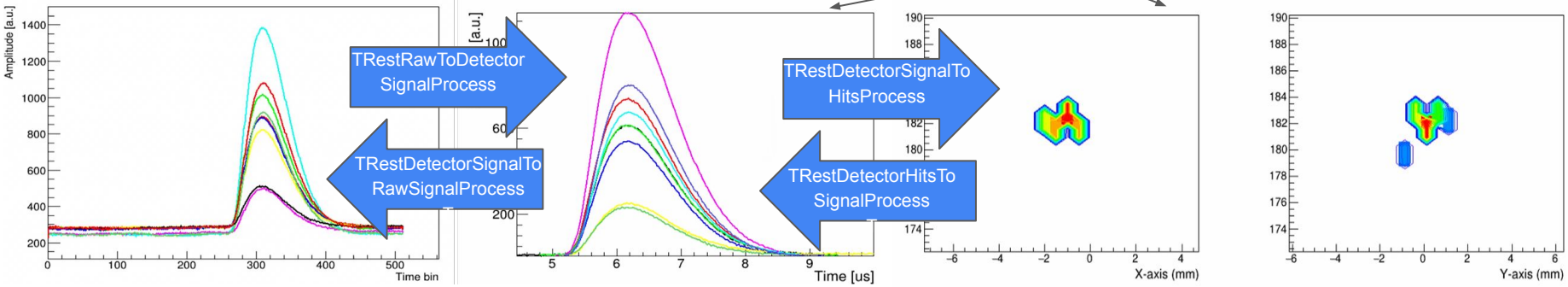
The connectors library

https://github.com/rest-for-physics/connectorslib

This library contains different processes that inter-connect fundamental REST libraries, requiring to transfer an event type into another. I.e. hit clustering to transform detector hits into a track event, or raw signal to be transformed into a detector event. It also may contain other complex processes that require to use 2 libraries simultaneously.

rawlib

detectorlib

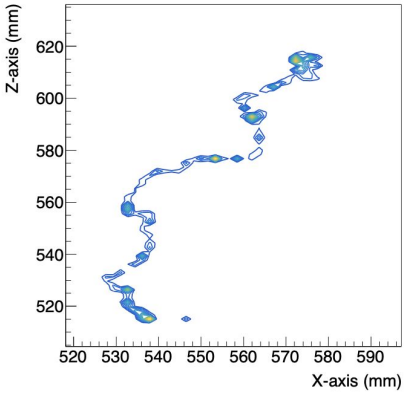


TRestRawSignalEvent

TRestDetectorSignalEvent

TRestDetectorHitsEvent

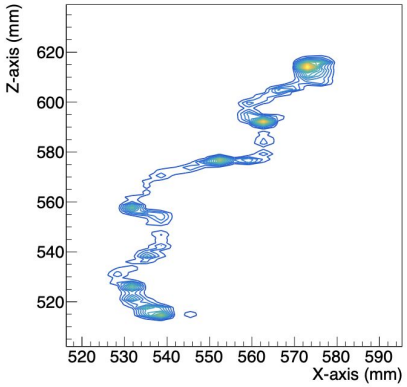
TRestGeant4Event



(a)

Simulated event

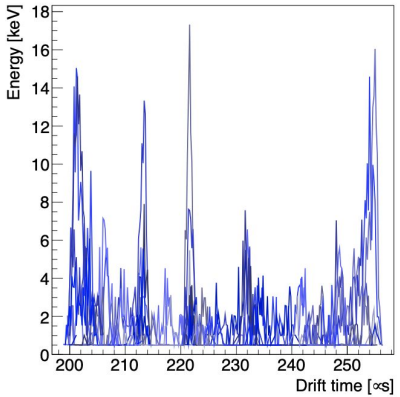
TRestDetectorHitsEvent



(b)

Appearance of the event after diffusion

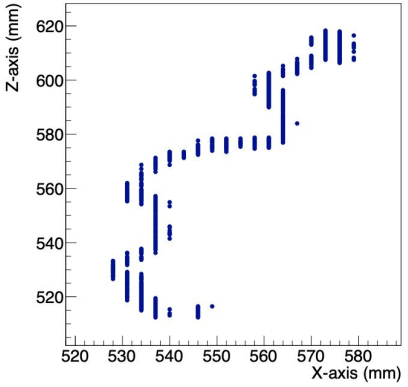
TRestDetectorSignalEvent



(c)

Reconstructed signal event

TRestDetectorHitsIEvent



(d)

Reconstructed hits event. The effect of the readout are clearly seen.

From *Topological background discrimination in the PandaX-III neutrinoless double beta decay experiment*
https://arxiv.org/abs/1903.03979

Time for exercises!

3: Diffusing and smearing of punctual detector energy deposits .

4: Transforming our detector hits into a rawsignal (electronics acquisition-like data).

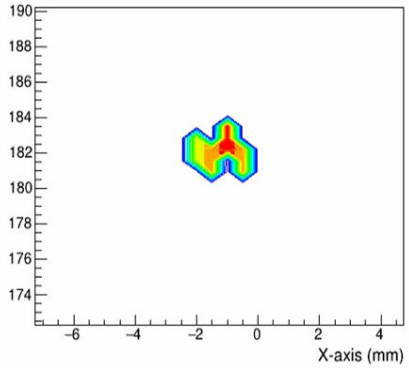
The connectors library

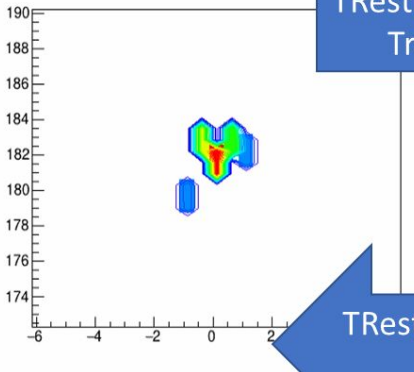
https://github.com/rest-for-physics/connectorslib

DOI 10.5281/zenodo.7495016 | user guide | user API | user forum

This library contains different processes that inter-connect fundamental REST libraries, requiring to transfer an event type into another. I.e. hit clustering to transform detector hits into a track event, or raw signal to be transformed into a detector event. It also may contain other complex processes that require to use 2 libraries simultaneously.

detectorlib

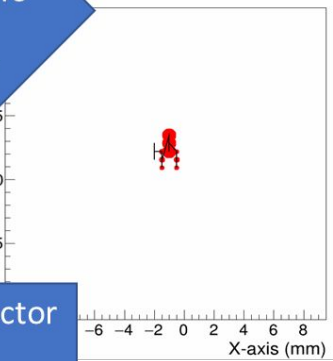


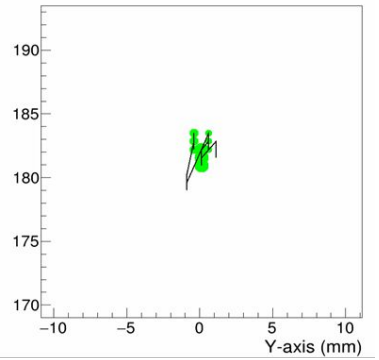


TRestDetectorHitsToTrackProcess

TRestTrackToDetectorHitsProcess

tracklib



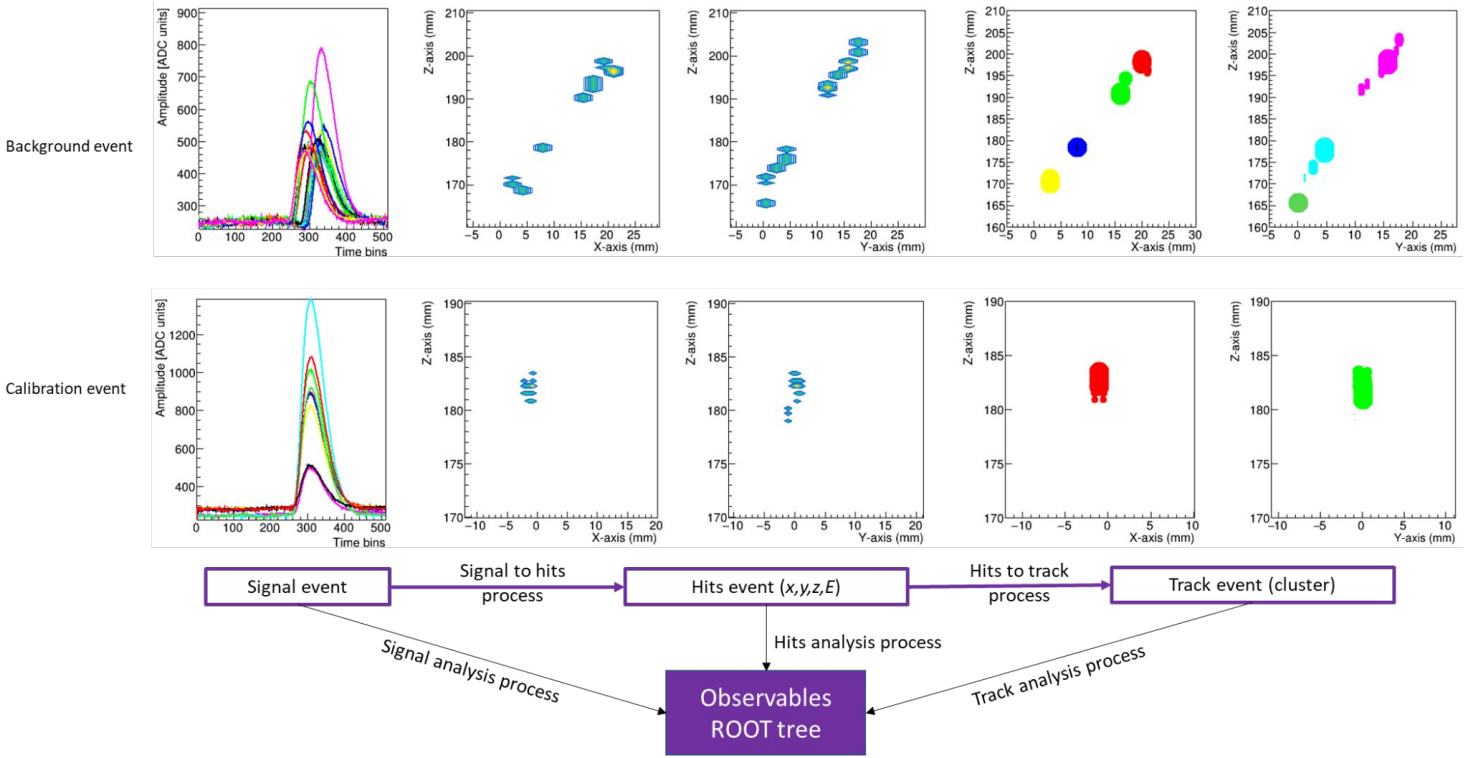


Hits event

Track event

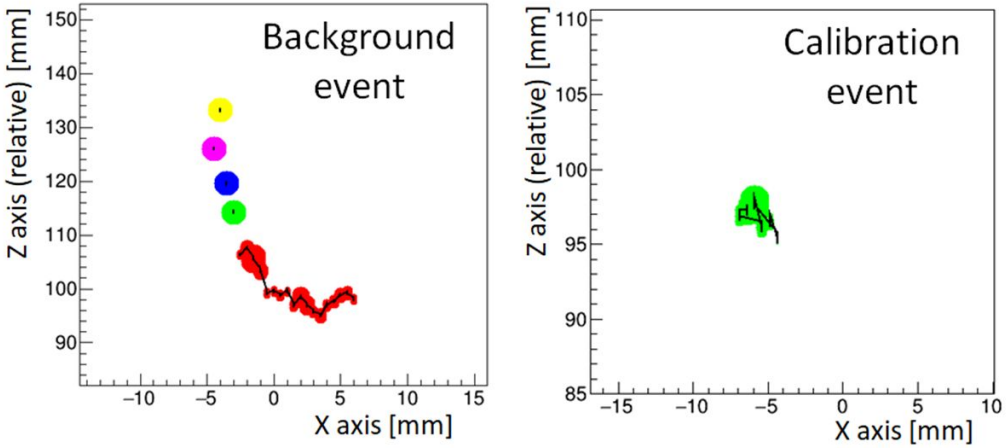
An event changes its type as it goes through the different processes.

The figure shows how both a calibration and a background signal event evolve along the processing chain, and how the ROOT tree is populated with observables computed at different stages of the processing chain.



The computed observables are used to define selection algorithms. Three main types of cuts are applied:

- Energy cuts: e.g. (1,10) keV.
- Fiducial cut: to select the size of the spot (e.g. 10 mm²).
- Topological cuts: event size and shape in the XY plane and in the Z direction.



Backup slides

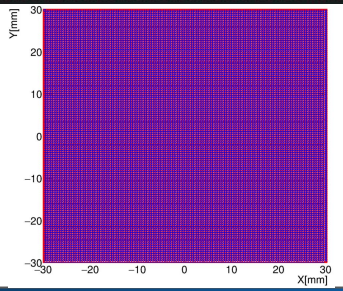
And how is all this implemented in practice?

The analysis is handled by RML configuration files where one can define:

- the variables common to all processes
- the list of processes to be executed
- the parameters/cuts for each process

And a readout root file specifying the characteristics of the detector and the readout.

```
TRestDetectorReadout content
Config file : readouts_IAXOD0.rml
Name : iaxo_readout
Title : IAXO-D0 readout 0.5 mm-Pitch 120+120 channels
REST Version : 2.3.4
REST Official release: No
Clean state: No
REST Commit : 02c4aa04
REST Library version : 1.0
Number of readout planes : 1
Decoding was defined : YES
-- Readout plane : 0
-- Position : X = 0 mm, Y : 0 mm, Z : -15 mm
-- Vector : X = 0 mm, Y : 0 mm, Z : 1 mm
-- Cathode Position : X = 0 mm, Y : 0 mm, Z : 15 mm
-- Total drift distance : 30 mm
-- Charge collection : 1
-- Total modules : 1
-- Total channels : 240
```




```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<TRestManager>

    <!-- Definition of global variables-->
    <globals file="globals.xml"/>

    <!-- Definition of TRestRun -->
    <TRestRun file="run.xml" />

    <TRestProcessRunner name="RawSignals" title="Raw processing and analysis" >

        <parameter name="firstEntry" value="0" />
        <parameter name="lastEntry" value="0" />
        <parameter name="eventsToProcess" value="1000" />

        <!-- Initial data processing: raw signal generation -->
        <addProcess type="TRestRawMultiFEMINOSToSignalProcess" name="virtualDAQ" value="ON" verboseLevel="silent">
            <parameter name="pedScript" value="ped"/>
            <parameter name="runScript" value="run"/>
            <parameter name="electronics" value="singleFeminos"/>
            <parameter name="fileFormat" value="SJTU" />
        </addProcess>
        <addProcess type="TRestRawSignalChannelActivityProcess" name="rawChActivity" value="ON" verboseLevel="info" file="processes.rml"/>
        <addProcess type="TRestRawVetoAnalysisProcess" name="veto" value="ON" vetoSignalId="4322" verboseLevel="info" file="processes.rml"/>

        <!-- Signal analysis -->
        <addProcess type="TRestRawSignalAnalysisProcess" name="sAna" value="ON" file="processes.rml"/>
        <addProcess type="TRestRawSignalViewerProcess" name="rsViewer" value="OFF" file="processes.rml"/>
        <addProcess type="TRestRawToDetectorSignalProcess" name="zSS" value="ON" file="processes.rml"/>
        <addProcess type="TRestDetectorSignalChannelActivityProcess" name="chActivity" value="ON" file="processes.rml"/>

        <!-- Signal to track reconstruction and track analysis -->
        <addProcess type="TRestDetectorSignalToHitsProcess" name="signalToHits" value="ON" verboseLevel="silent" file="processes.rml"/>
        <addProcess type="TRestDetectorHitsAnalysisProcess" name="hitsAna" value="ON" file="processes.rml"/>
        <addProcess type="TRestEventRateAnalysisProcess" name="rateAna" value="ON">
            <parameter name="observable" value="all"/>
        </addProcess>
        <addProcess type="TRestDetectorHitsAnalysisProcess" name="hitsAnaLite" value="OFF" file="processes.rml" />
        <addProcess type="TRestDetectorHitsToTrackProcess" name="hitsToTrack" value="ON" verboseLevel="silent" file="processes.rml"/>
        <addProcess type="TRestTrackAnalysisProcess" name="tckAna" value="ON" file="processes.rml"/>

    </TRestProcessRunner>

    <addTask type="processEvents" value="ON" />

</TRestManager>
```