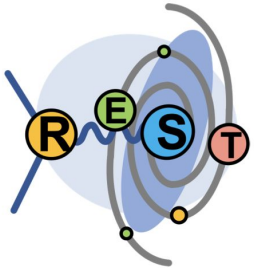


REST-for-Physics Framework

3.1 Raw-signal processing

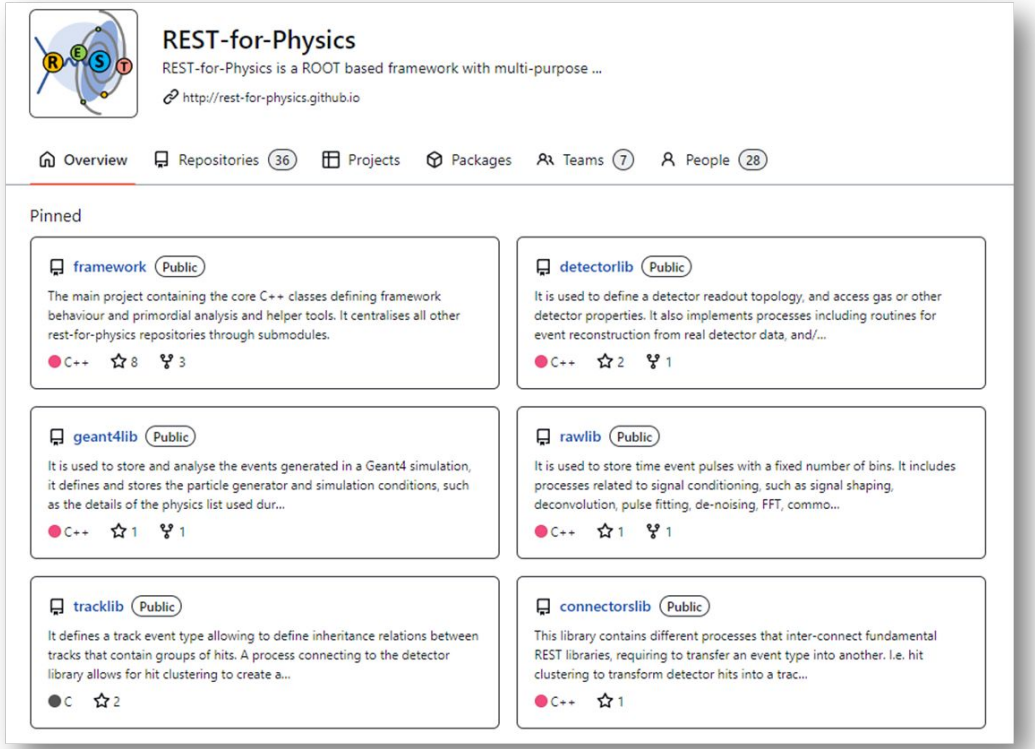
26.01.2023 - David Díez Ibáñez - daviddiez@unizar.es

Centro de Astropartículas y Física de Altas Energías
Universidad Zaragoza

- Overview of event data processing in REST-for-Physics
- Basic description of rawlib
- Signal data analysis
- Signal conditioning

The full REST-for-Physics project is splitted in different Github repositories

- **Main project**
  - Framework
- **Libraries** for montecarlo and detector data processing
  - Rawlib / Geant4lib
  - Detectorlib / Tracklib
  - Axionlib
  - Connectorslib
- **Packages** that exploit REST libraries
  - restG4
  - restSQL
  - ...



The full REST-for-Physics project is splitted in different Github repositories

- **Main project**
  - Framewo
- **Libraries** for mo
  data processing
  - Rawlib / C
  - Detectorli
  - Axionlib
  - Connecto
- **Packages** that e
  - restG4
  - restFileIn
  - ...

Naming convention:

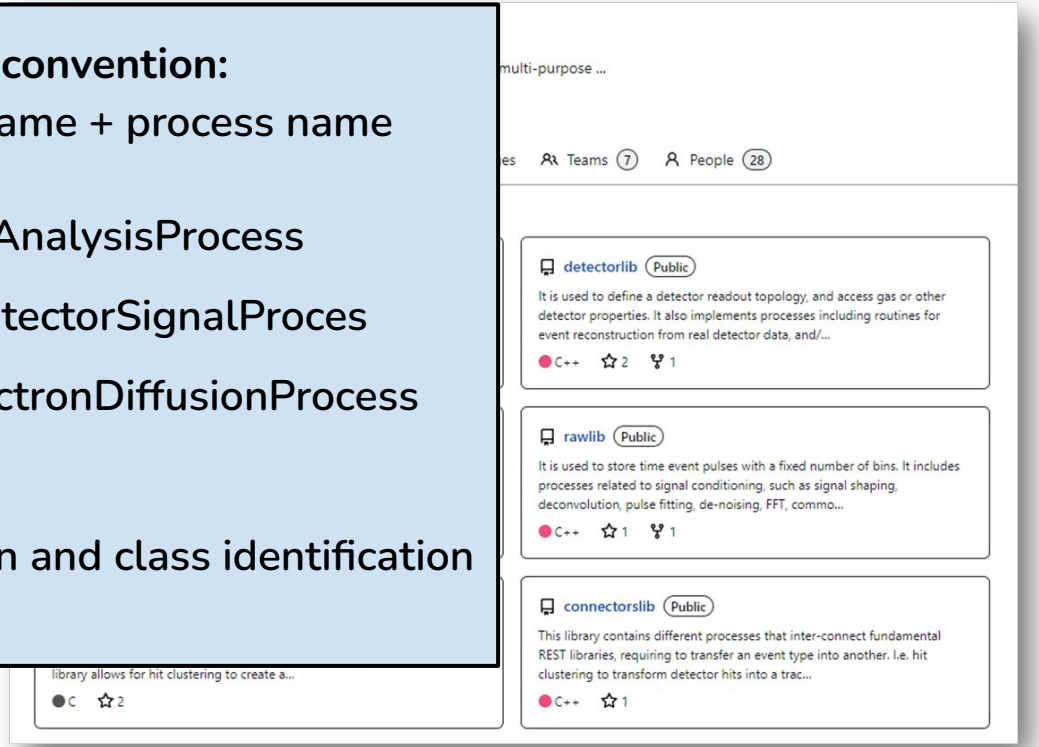
TRest + library name + process name

TRestAxionAnalysisProcess

TRestRawToDetectorSignalProces

TRestDetectorElectronDiffusionProcess

Helps autocompletion and class identification



TRestRun definition for event processing

```
<TRestRun name="CAST-Ar" title="CAST Ar data taking" verboseLevel="
    <parameter name="experiment" value="CAST"/>
    <parameter name="runNumber" value="preserve"/>
    <parameter name="runTag" value="preserve"/>
    <parameter name="runType" value="${RUN_TYPE}"/>
    <parameter name="runDescription" value=""/>
    <parameter name="user" value="${USER}"/>
    <parameter name="verboseLevel" value="0"/>
    <parameter name="overwrite" value="off" />

    <parameter name="inputFormat" value="${AQS_FILENAME_FORMAT}"/>
    <parameter name="outputFileName" value="R[fRunNumber]_[fParentRunNumber]_[fRunType]_[fRunTag]_[fRunUser]_[fVersion].root" />
    <parameter name="readOnly" value="false" />

    <TRestDetector name="detector" >
        <parameter name="detectorName" value="Micromegas" />
    </TRestDetector>

</TRestRun>
```

A reserved keyword. The runTag from the input file, or from other initialization method will be used.

The input filename created by the data acquisition might have a format with meaningful information. “inputFormat” allows to define the input filename format.

Output filename will be built using metadata members given between brackets []. Any metadata member is allowed. If the class name is not given, TRestRun is assumed.

Inside our TRestRun section we can define any metadata class that will be accessible inside the data processing chain.

Then we can decide which metadata members will be assigned with the values identified in the filename.

```
        <variable name="AQS_FILENAME_FORMAT" value="R[fRunNumber]_[fRunTag]_Vm_[TRestDetector::fAmplificationVoltage]_Vd_[TRestDetector::fDrift
Voltage]_Pr_[TRestDetector::fPressure]_Gain_[TRestDetector::fElectronicsGain]_Shape_[TRestDetector::fElectronicsShaping]_Clock_[TRestDetector::
fElectronicsClock]-[fParentRunNumber].aqs"/>
```

In order to create a processing chain we need to define the TRestProcessRunner

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

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

    <parameter name="inputEventStorage" value="False" />
    <parameter name="outputEventStorage" value="True" />

    <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>
```

Defines the range of events to be processed

Event storage. Seen already in session 2.3

First process definition

More process definitions within the *<addProcess key* follow in sequential order to define the data processing chain.

Process definitions

In-line metadata parameters definition

```
<addProcess type="TRestRawSignalChannelActivityProcess" name="rawChActivity" value="ON"
    lowThreshold="25" highThreshold="50" daqChannels="360" daqStartChannel="4320" daqEndChannel="4680"
    readoutChannels="250" readoutStartChannel="0" readoutEndChannel="250" />

<addProcess type="TRestRawSignalRecoverChannelsProcess" name="recoverChannels" value="OFF" >
    <parameter name="channelIds" value="{4541}" />
</addProcess>

<addProcess type="TRestRawVetoAnalysisProcess" name="veto" value="ON" vetoSignalId="4322">
    <parameter name="baseLineRange" value="(10,100)"/>
    <parameter name="range" value="(10,500)"/>
    <parameter name="observable" value="all"/>
    <parameter name="threshold" value="500"/>
    <parameter name="timeWindow" value="100,300"/>
</addProcess>

<!-- Signal analysis -->
<addProcess type="TRestRawSignalAnalysisProcess" name="sAna" value="ON" file="processes.rml"/>

<addProcess type="TRestEventRateAnalysisProcess" name="rateAna" observables="all" value="ON"/>
```

Disabled process

In-section metadata parameters definition

External process definition

A **working example for the TREX-DM** data processing is provided inside the validation pipeline.

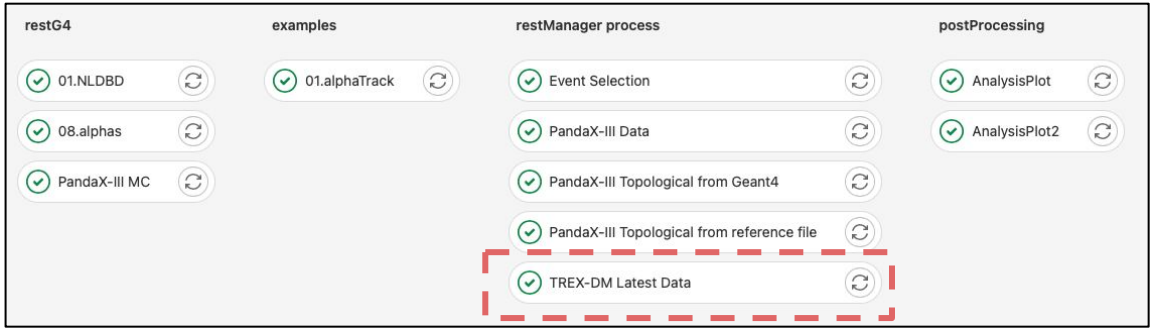
See directory pipeline/trexdm/

Pipelines are running at GitLab CERN instance and at GitHub workflow pipelines.

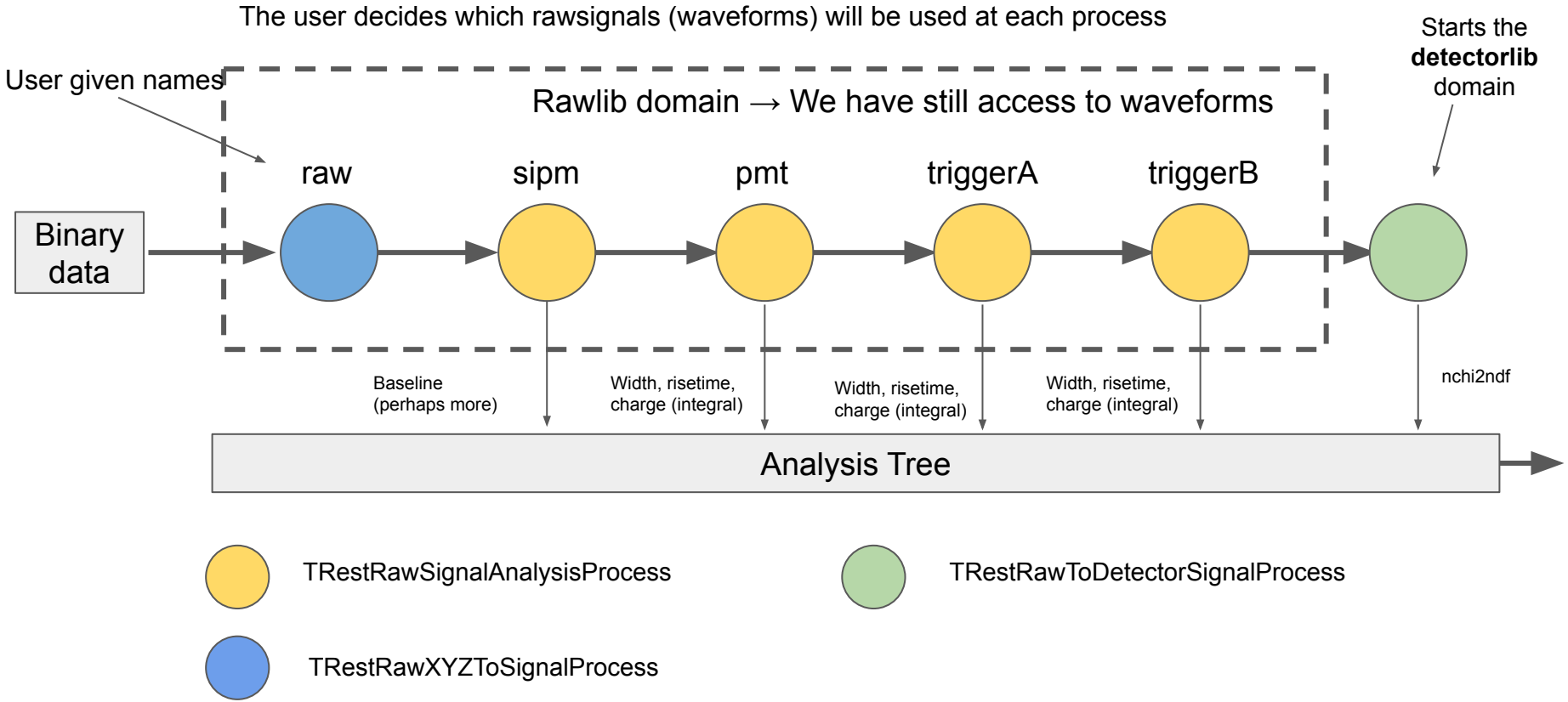
See implementations at .gitlab-ci.yml and .github/workflow.md

The pipeline guarantees that the TREX-DM pipeline keeps working properly and reproducing the same results, and that it does not break with future contributions to the REST-for-Physics code.

Only if pipeline succeeds we will be able to merge our new codes to the master branch repository.

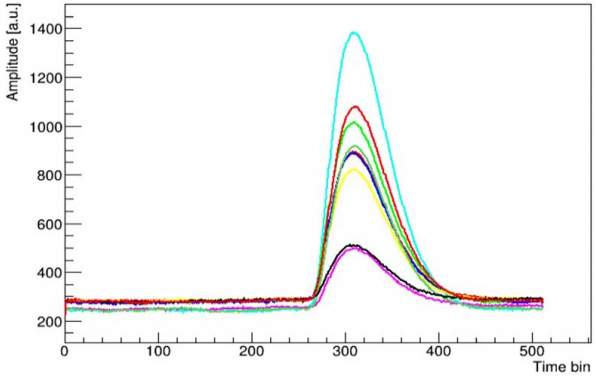


RawSignal in processing chain



Data class to store a set of pulses.

TRestRawSignal is not only storage, **it also implements methods** that operate on the waveforms, such as retrieving risetime or pulse amplitude.



TRestEvent
Int_t fId;
TTimeStamp fTimeStamp;
....

TRestRawSignalEvent
std::vector <TRestRawSignal> fSignal;

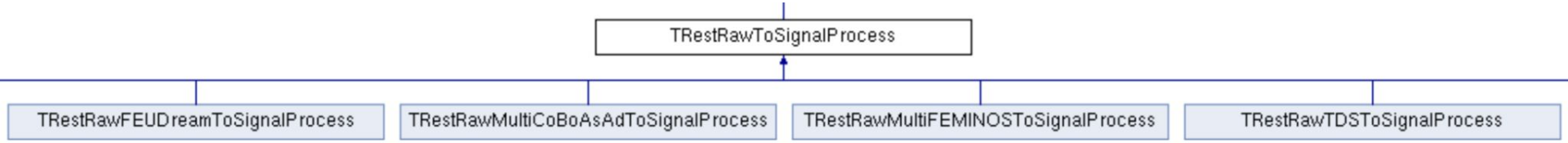
TRestRawSignal
Int_t fSignalId;
std::vector <Short_t> fSignalData;

A **TRestRawSignal** contains a given number of samples with fixed sampling time.

Values such as the sampling time is stored as a metadata member at a dedicated metadata class

rawlib : In this library processes using TRestRawSignalEvent only (as input and output).

Exception: processes to read data from binaries. Those that inherit from TRestRawToSignalProcess



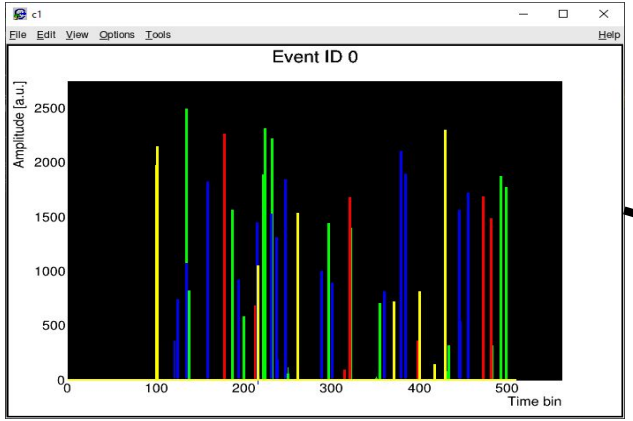
Event type: TRestRawSignalEvent.cxx

Analysis: TRestRawSignalAnalysisProcess.cxx

Manipulation: TRestRawSignalShapingProcess.cxx
TRestRawSignalAddNoiseProcess.cxx
TRestRawCommonNoiseReductionProcess.cxx
TRestRawSignalGeneralFitProcess.cxx

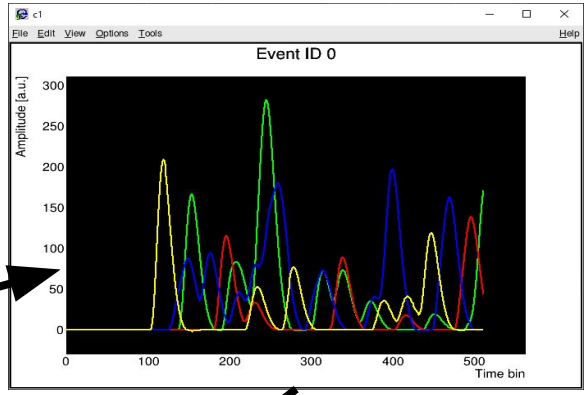
And many others...

RawSignal processes

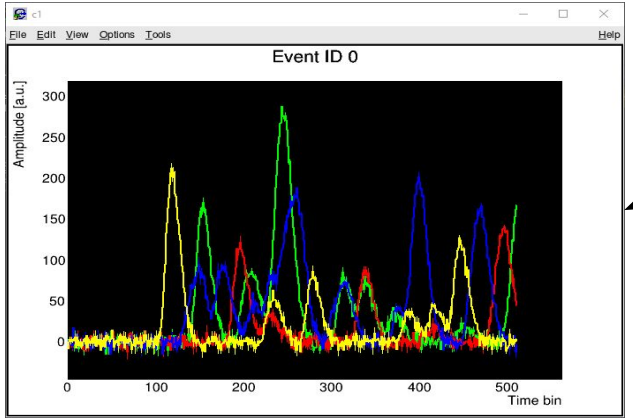


Simulation

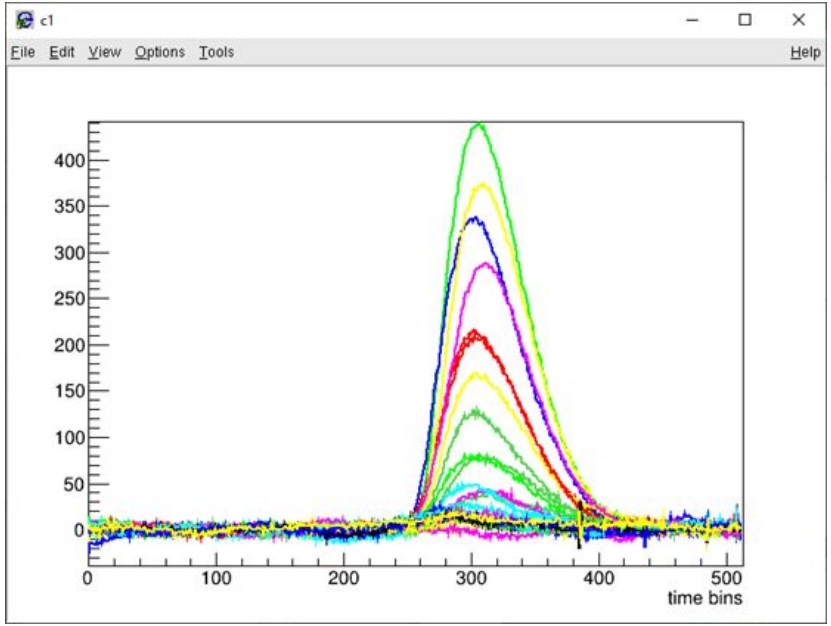
TRestRawSignal
Shaping
Process



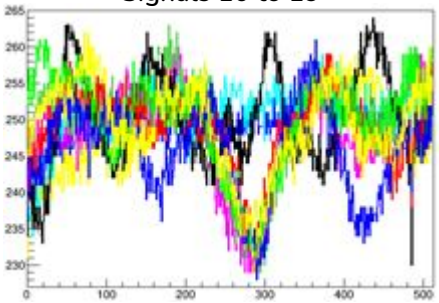
TRestRawSignal
AddNoise
Process



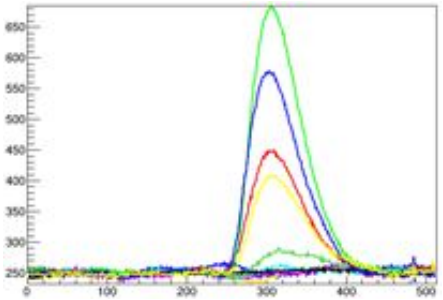
Add random noise to
emulate real data



Signals 10 to 19



Signals 50 to 59



DrawEvent() method of TRestRawSignalEvent

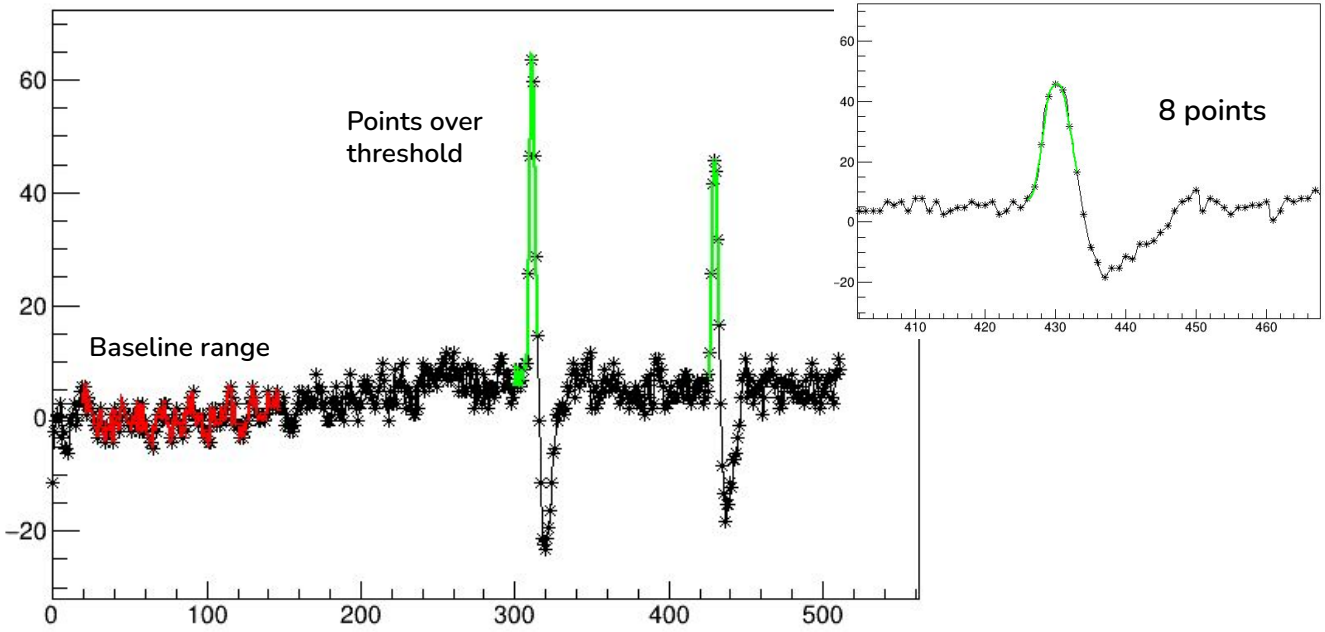
It allows to select signals by range or ids and apply “good signal” selection:

ev0->DrawEvent("0-10:onlyGoodSignals[3.5,1.5,7]:baseLineRange[20,150]:printIDs");

TRestRawSignal::InitializePointsOverThreshold is the key function.

Identifies signals with points over threshold using 3 parameters: *pointThreshold, signalThreshold* and *nPointsOver*

pointThreshold 2.2
signalThreshold 1.5
nPointsOver 7

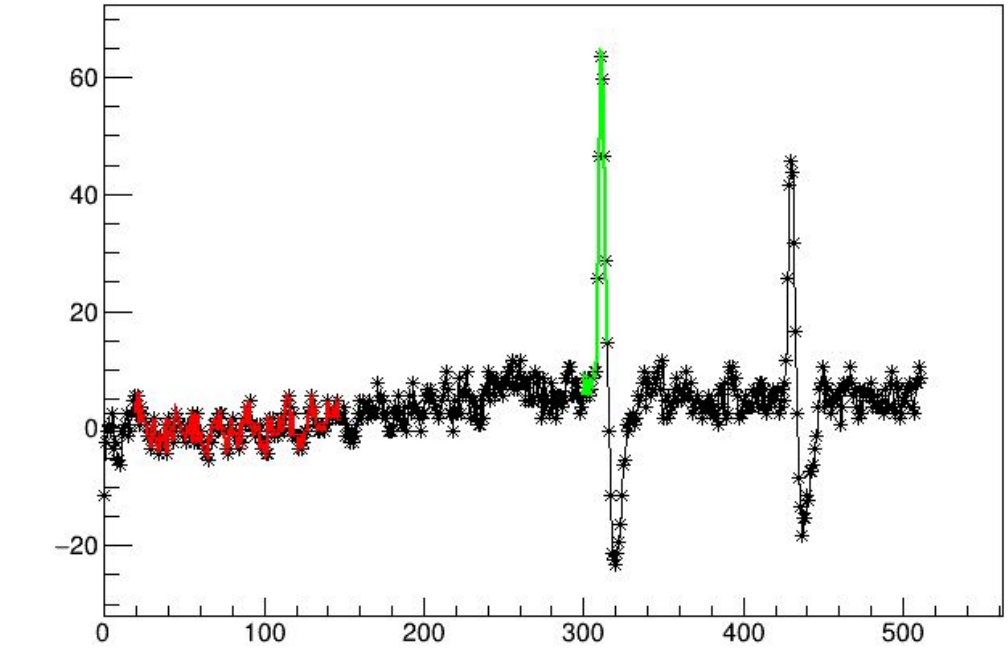



```
root [10] ev0->DrawSignal(4777,"goodSignals[2.2,1.5,7]:baseLineRange[20,150]")
(TPad *) 0x55579dd31a80
```

TRestRawSignal::InitializePointsOverThreshold is the key function.

Identifies signals with points over threshold using 3 parameters: *pointThreshold, signalThreshold* and *nPointsOver*

pointThreshold 2.2
signalThreshold 1.5
nPointsOver 9




```
root [11] ev0->DrawSignal(4777,"goodSignals[2.2,1.5,9]:baseLineRange[20,150]")
(TPad *) 0x55579dd31a80
```

The process TRestRawSignalAnalysisProcess produces observables from TRestRawSignalEvent.

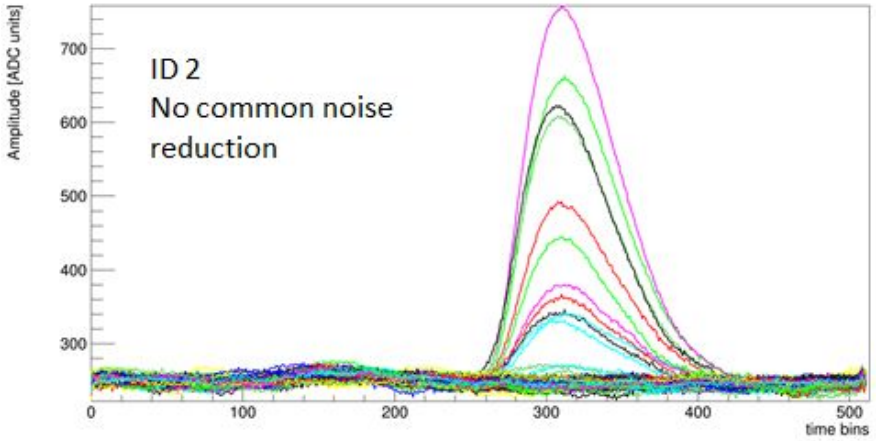
Some of them only take into account the selected points over threshold.

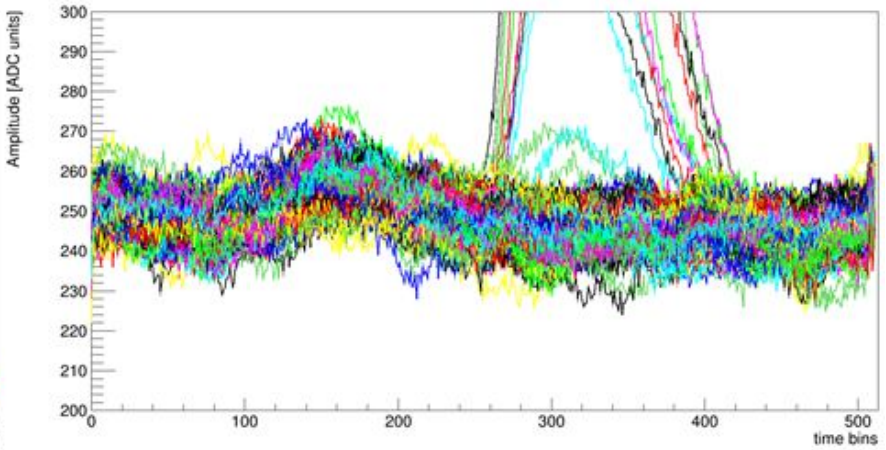
```
<!-- Signal analysis -->
<addProcess type="TRestRawSignalAnalysisProcess" name="sAna" title="" verboseLevel="info" observable="all">

    <!-- This parameter is used to define the baseline calculation -->
    <parameter name="baseLineRange" value="(${BL_MIN},${BL_MAX})" />

    <!-- This parameter is used to define the signal integration window -->
    <parameter name="integralRange" value="(${INT_MIN},${INT_MAX})" />

    <!-- These parameters define the integral with threshold.
        threshold : number of baseline noise sigmas to consider a point for integration.
        pointsOverThreshold : Number of consecutive points over threshold to be considered for integration.
        minPeakAmplitude : Minimum peak signal amplitude to be considered at all.
    -->
    <parameter name="pointThreshold" value="${POINT_TH}" />
    <parameter name="pointsOverThreshold" value="${NPOINTS}" />
    <parameter name="signalThreshold" value="${SGNL_TH}" />
</addProcess>
```

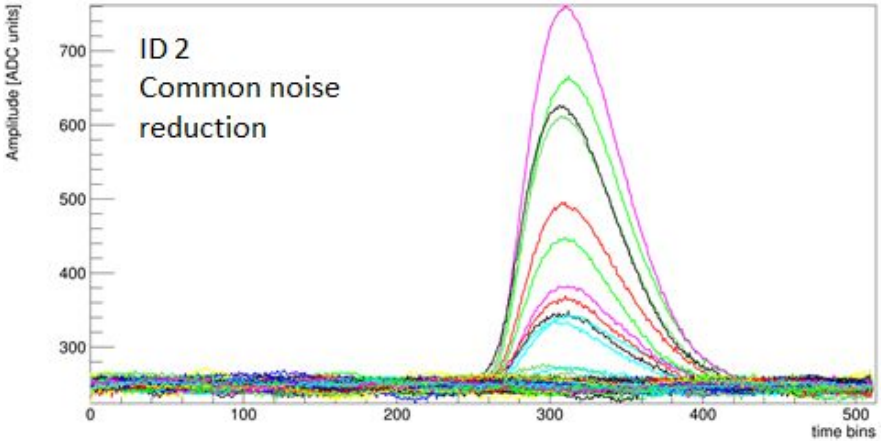
RawSignal Analysis

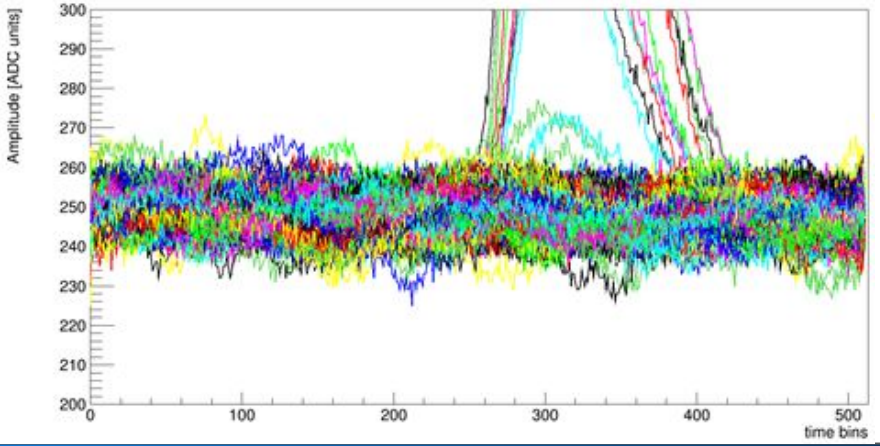
```
root [2] run0->PrintObservables()
Entry : 1
> Event ID : 3451923
> Event Time : 1664952185.90144
> Event Tag :
----------------------------------------
Observable : rawAna_pointsoverthres_map                 Value : {[120:65],[123:68],[180:0],[181:75
Observable : rawAna_risetime_map                        Value : {[120:18],[123:20],[180:0],[181:23
Observable : rawAna_peak_time_map                       Value : {[120:212],[123:213],[180:216],[18
Observable : rawAna_baseline_map                        Value : {[120:264.638],[123:257],[180:263.
Observable : rawAna_baselinesigma_map                   Value : {[120:28.9341],[123:35.6832],[180:
Observable : rawAna_max_amplitude_map                   Value : {[120:529.362],[123:754],[180:379.
Observable : rawAna_thr_integral_map                    Value : {[120:21815.5],[123:31511],[180:0]
Observable : rawAna_SaturatedChannelID                  Value : {}
Observable : rawAna_BaseLineMean                        Value : 261.038
Observable : rawAna_BaseLineSigmaMean                   Value : 20.0425
Observable : rawAna_TimeBinsLength                      Value : 512
Observable : rawAna_NumberOfSignals                     Value : 10
Observable : rawAna_NumberOfGoodSignals                 Value : 5
Observable : rawAna_FullIntegral                        Value : 290788
Observable : rawAna_ThresholdIntegral                   Value : 269656
Observable : rawAna_RiseSlopeAvg                        Value : 20509.8
Observable : rawAna_SlopeIntegral                       Value : 102549
Observable : rawAna_RateOfChangeAvg                     Value : 0.2
Observable : rawAna_RiseTimeAvg                         Value : 22
Observable : rawAna_TripleMaxIntegral                   Value : 18888
Observable : rawAna_IntegralBalance                     Value : 0.0377064
Observable : rawAna_AmplitudeIntegralRatio              Value : 42.7526
Observable : rawAna_MinPeakAmplitude                    Value : 529.362
Observable : rawAna_MaxPeakAmplitude                    Value : 2000.96
Observable : rawAna_PeakAmplitudeIntegral               Value : 6307.35
Observable : rawAna_MinEventValue                       Value : -85
Observable : rawAna_AmplitudeRatio                      Value : 3.15216
Observable : rawAna_MaxPeakTime                         Value : 216
Observable : rawAna_MinPeakTime                         Value : 212
Observable : rawAna_MaxPeakTimeDelay                    Value : 4
Observable : rawAna_AveragePeakTime                     Value : 214.2
Observable : TREXsides_tagId                            Value : 1
root [3]
```

Modifying RawSignalEvents: Common Noise Reduction



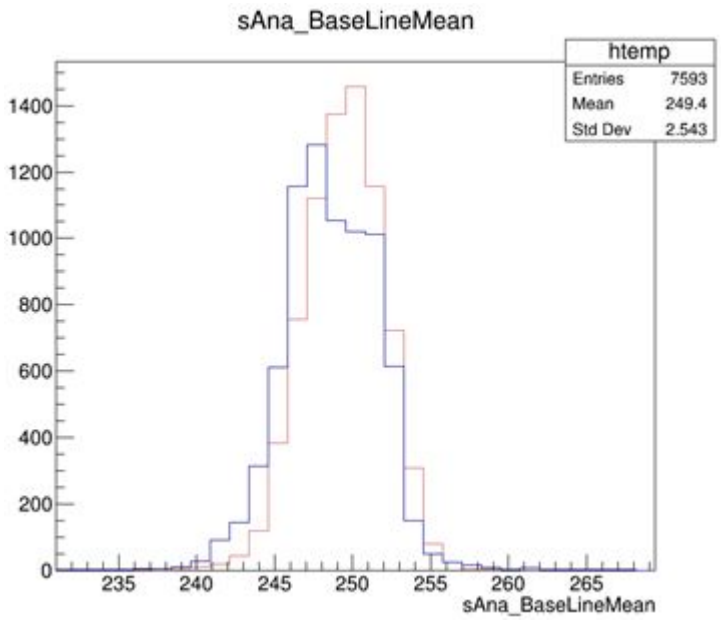


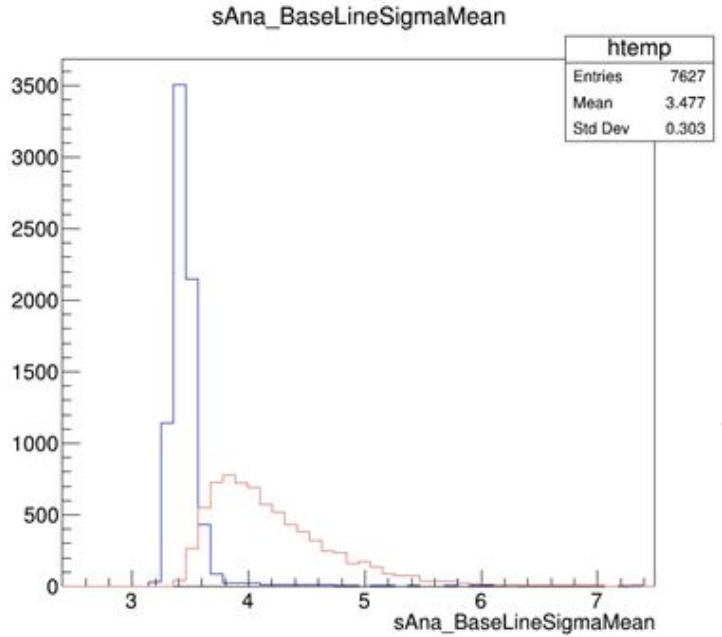




TRestRawCommonNoiseReductionProcess sustracts median value per bin to each signal.

Reduces the variance of the baseline without changing the baseline value.





Blue: common noise reduction

Red: without common noise reduction

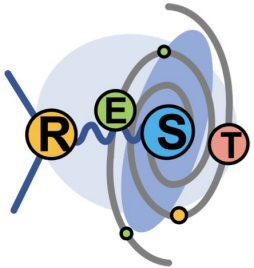
Not only for RawSignal events but for any kind of event

TRestEventSelectionProcess -> Apply conditions to select events

TRestEventRateAnalysisProcess -> Rate and other time observables

TRestSummaryProcess -> Save as metadata parameters from observable distributions

TRestMySQLToAnalysisProcess -> Add observables to AnalysisTree from database



REST-for-Physics Framework

Time for exercises!

26.01.2023 - David Díez Ibáñez - daviddiez@unizar.es

Centro de Astropartículas y Física de Altas Energías
Universidad Zaragoza

1. Binary file:

   R10513_Calibration_15min_Vm_340_Vd_113_Pr_1.4_Gain_0x1_Shape_0xD_Clock_0x02-000.aqs

2. Processing file: cast.rml
3. Open: R10513_00000_BasicRaw_Calibration_15min_${USER}_2.3.15.root
4. Print metadata: md_detector->PrintMetadata()

1. **Binary file:** R01208_Ar2Iso_Background14h_14Vetos_IccubFEC-000.aqs
2. **Processing file:** veto.rml
3. **Open:** R01208_quickData.root
4. **Print metadata:** run0->GetAnalysisTree()->PrintObservables()
5. Save ids: ids.txt

Exercise 3. Re-processing events on a given event selection

1. Binary file:

   R01208_Ar2Iso_Background14h_14Vetos_IccubFEC-000.aqs

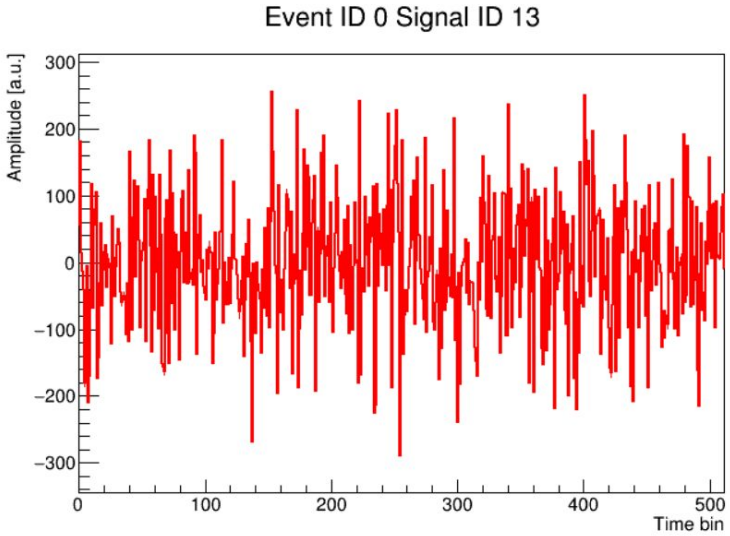
2. EventSelectionProcess in vetoOnSelection.rml
3. Check IDs of output events.

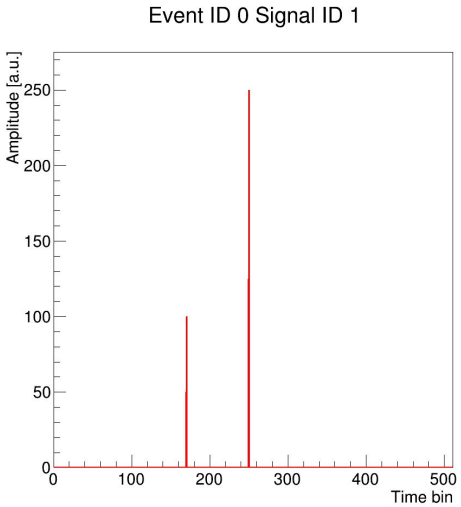
4.1 - Adding signal noise

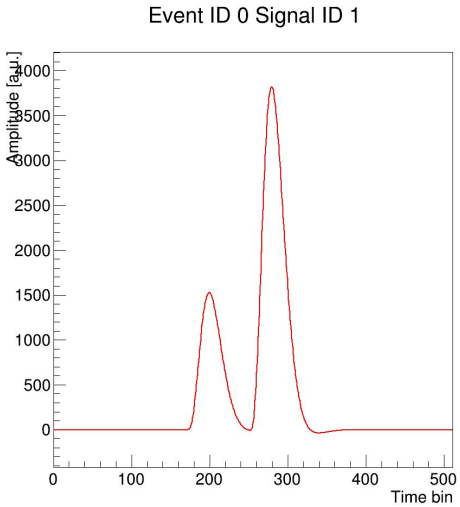
Write a python script to generate a noise signal.

4.2 - Signal convolution

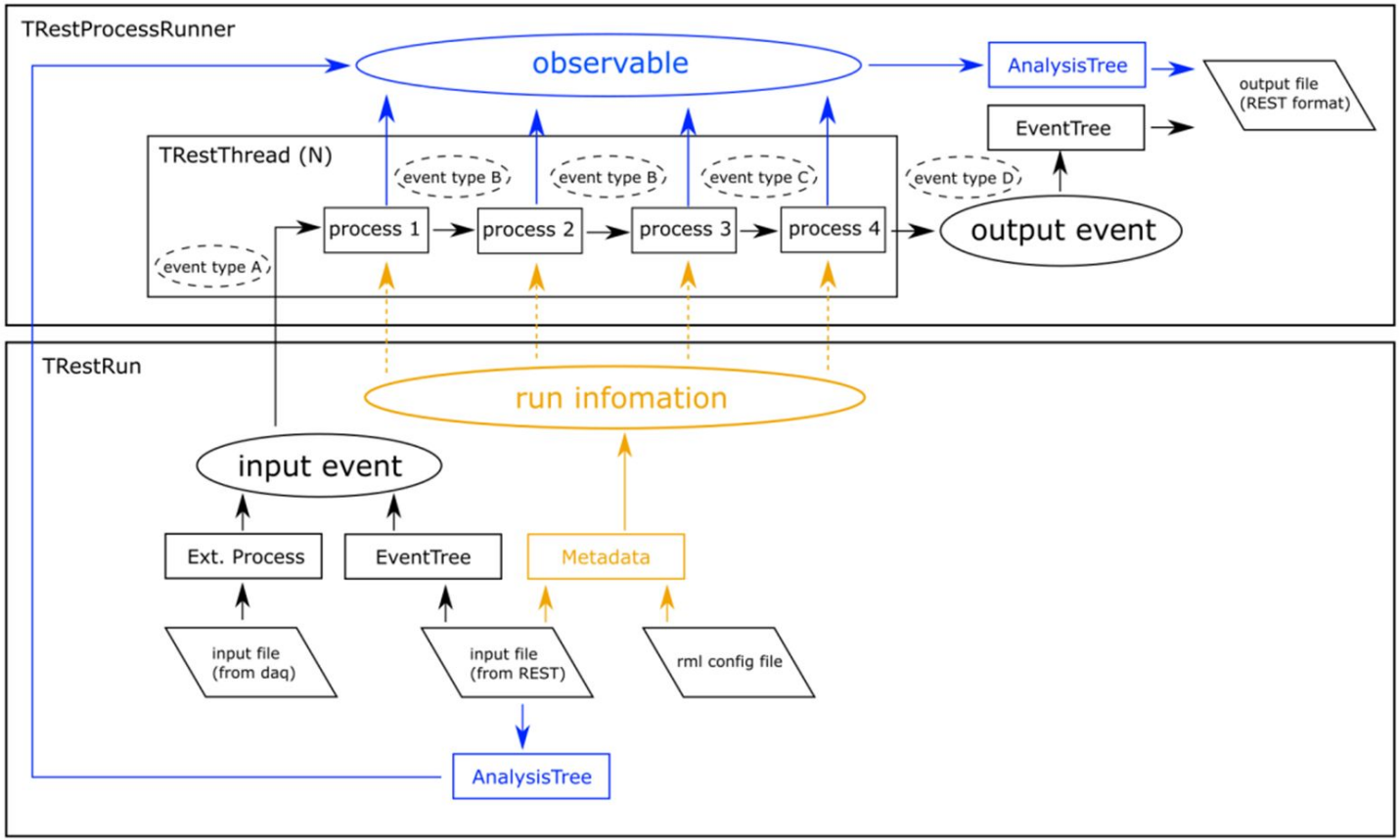
Write a C++ macro to convolute a pulse.



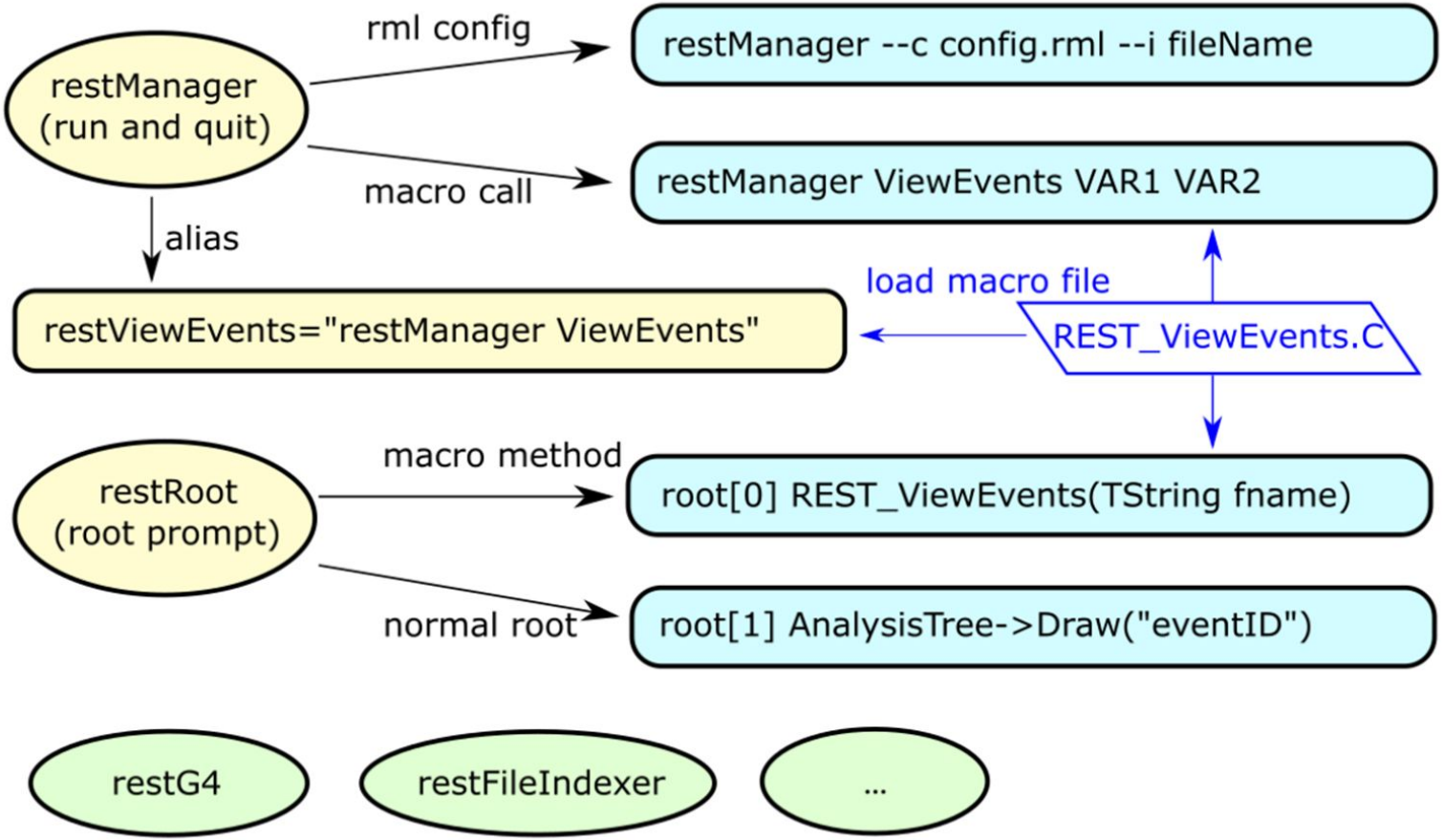




REST-for-Physics workflow in a processing event chain



Different ways of invoking/using REST-for-Physics



Some examples of using REST commands

(1) You can also call REST packages without Python bindings (using !)

```
!restG4 --help
```

```
restG4 requires at least one parameter, the rml configuration file (-c is optional)

example: restG4 example.rml

there are other convenient optional parameters that override the ones in the rml file:
        -h or --help | show usage (this text)
        -c example.rml | specify RML file (same as calling restG4 example.rml)
        -g geometry.gdml | specify geometry file
        -i | set interactive mode (default=false)
        -s | set serial mode (no multithreading) (default=true)
        -t nThreads | set the number of threads, also enables multithreading
```

(2) Let's run a simulation with restG4!

```
!restG4 simulations/simulation.rml
```

(3) You can see config file contents via console or

```
!cat simulations/simulation.rml
```

(4) To see ROOT file contents:

```
filename = "restG4_CosmicMuons_run00001.root"

file = ROOT.TFile(filename)

file.ls()
```

```
TFile**         restG4_CosmicMuons_run00001.root
 TFile*         restG4_CosmicMuons_run00001.root
  KEY: TRestAnalysisTree        AnalysisTree;3  AnalysisTree
  KEY: TTree    EventTree;3     TRestGeant4EventTree
  KEY: TRestRun DemoRun;3       A Demo Run
  KEY: TRestGeant4Metadata      restG4 run;2    Cosmic Muons
  KEY: TRestGeant4PhysicsLists  default;2       Physics List implementation.
```

(5) To access simulation event information:

```
run = ROOT.TRestRun(filename)

run.Print()

print(f"This run has {run.GetEntries()} entries")

event = ROOT.TRestGeant4Event()

run.SetInputEvent(event)

run.GetEntry(0)

event.PrintEvent()
```