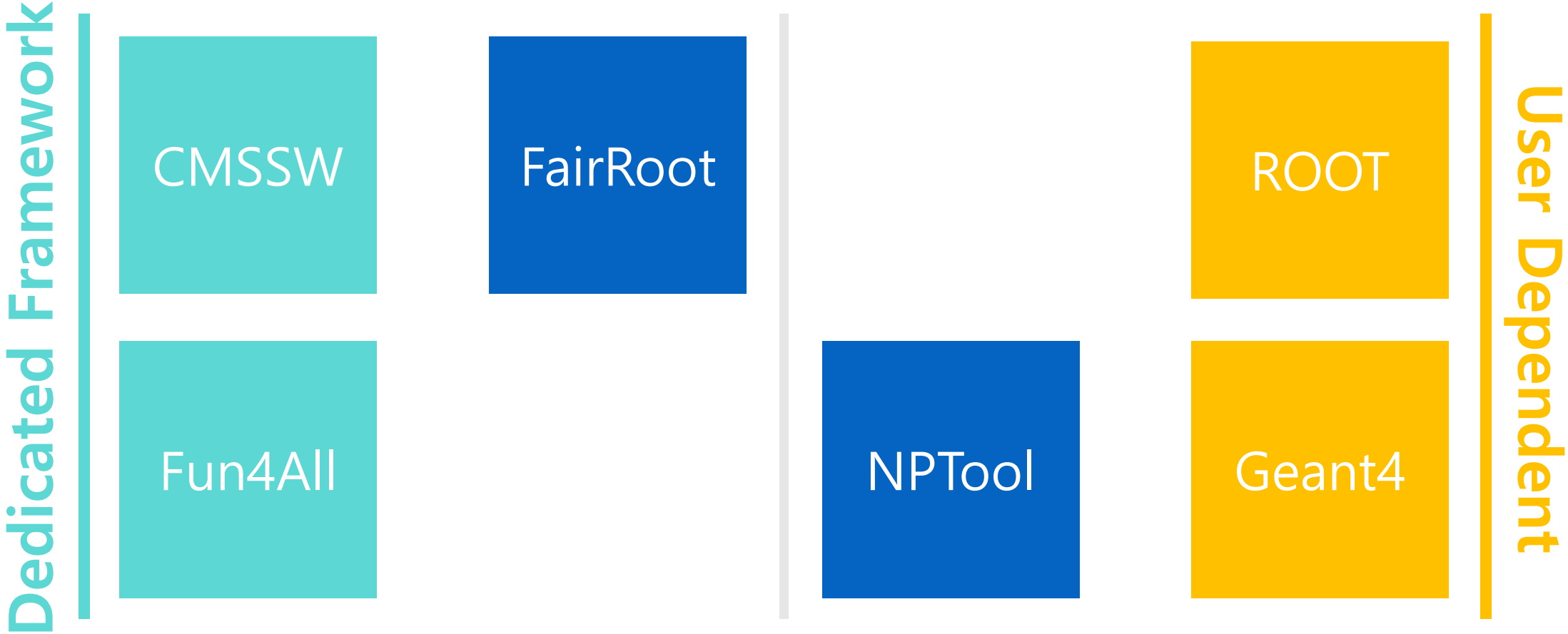
Analysis framework LILAK

Jung Woo Lee

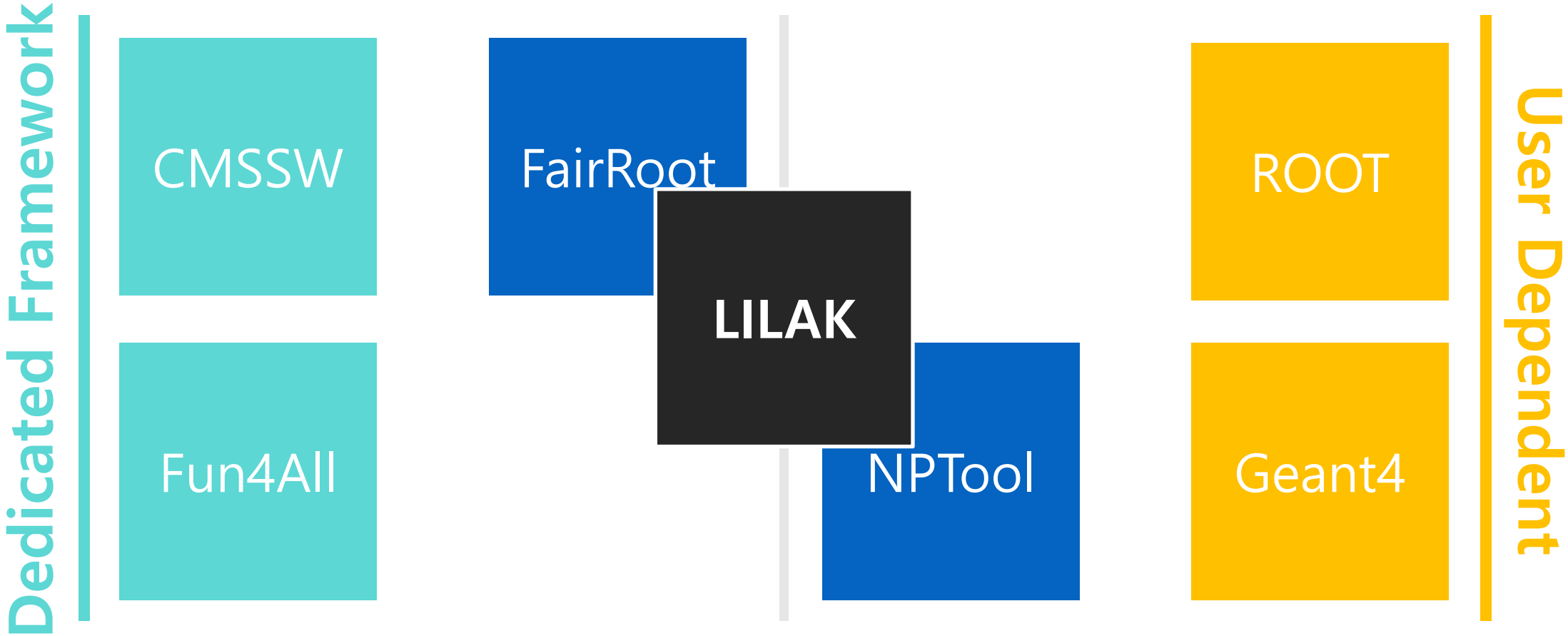
Tables of contents

- LILAK Design
- Tool
- Task
- Container
- LILAK Run
- Parameter Container
- MFM Converter
- Download
- Links

Analysis Frameworks



Analysis Frameworks



LILAK

Low and **I**ntermediate energy nuc**l**ear experiment **A**nalysis tool**k**it

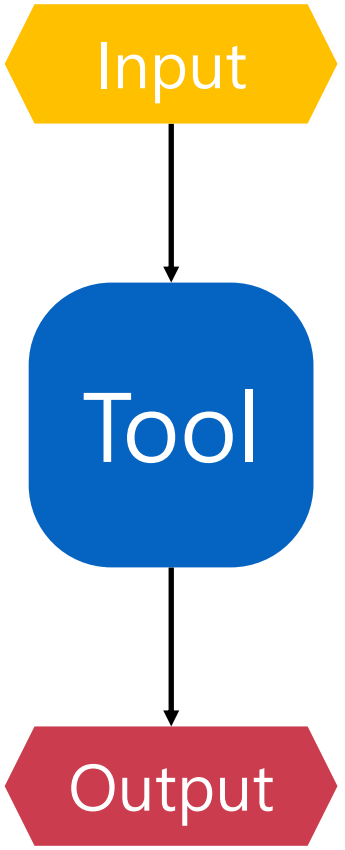
1. General analysis toolkit
2. Task base analysis.
3. Well defined data container.
4. Easy parameter handling.
5. No package dependence other than Git and ROOT (+Geant4).
6. Sharable analysis code.
7. TPC friendly software.
8. Version control using Git.
9. Documentation using Doxygen.

LILAK

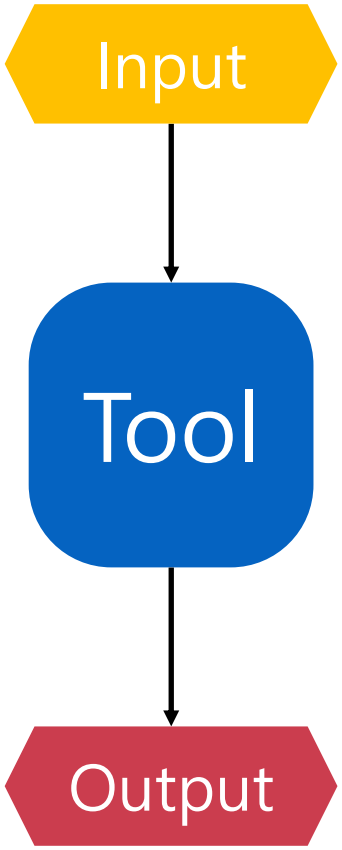
- Components
  - Tool
  - Task
  - Container
- Stear
  - Run manager (LKRun)
  - Parameter container (LKParameterContainer)
  - Logger (LKLogger)
  - User project
- Others
  - Geant4 binding

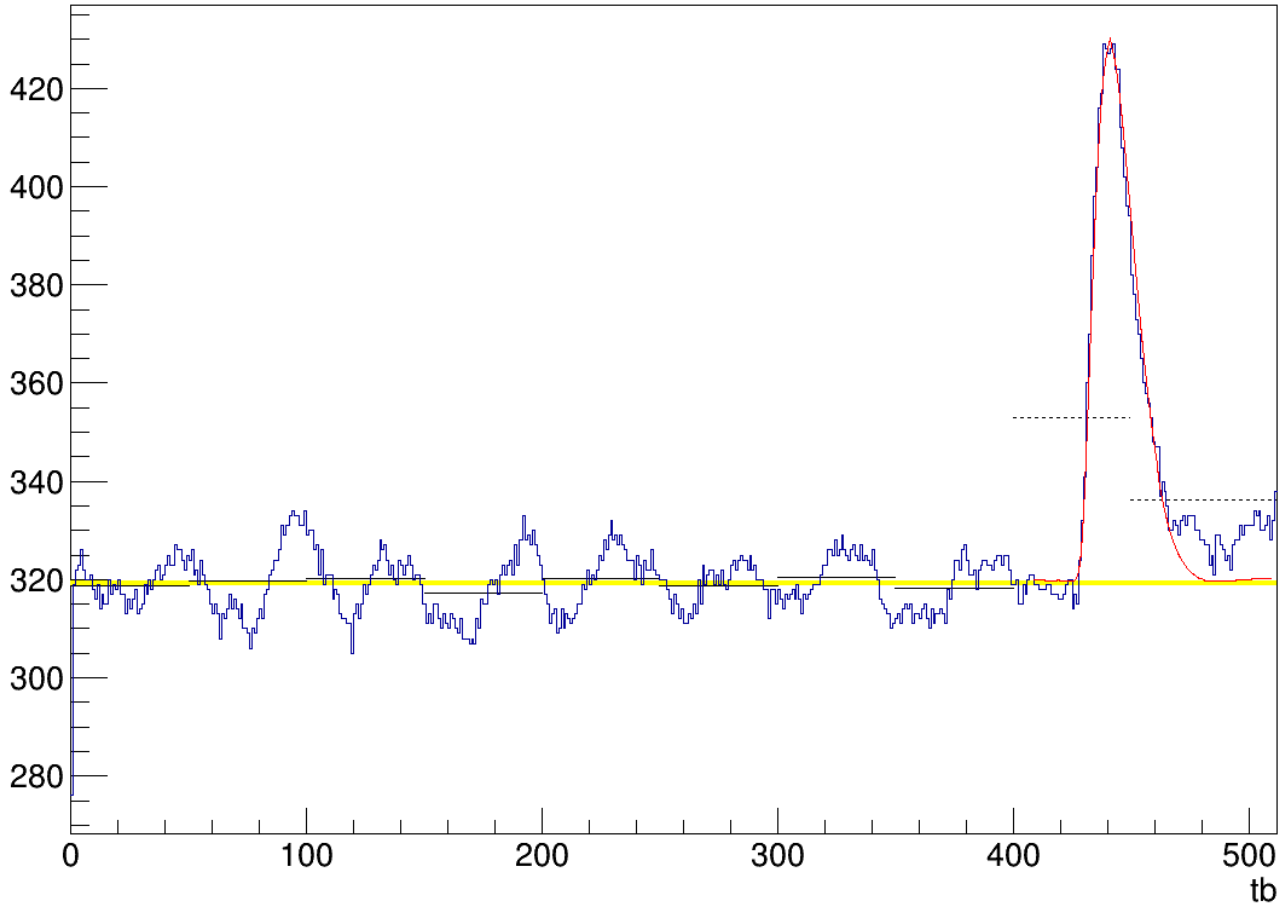
Tool

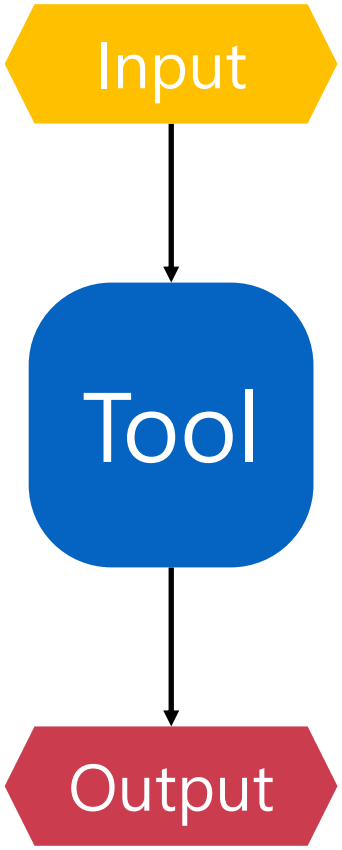
- Examples of tool:
  - Pulse Shape Analyzer
  - Hough Transform Tracker
  - Track Fitter
- Works separately outside the LKRun.
- Can be used repeatedly without exiting the program.
- Input and output of the data should be done manually.



```
void example_tool()
{
    auto run = new LKRun();
    run -> AddPar("config.mac");
    run -> AddInputFile("data.root");
    run -> Init();
    run -> GetEvent(8);
    auto chArray = run -> GetBranchA("RawData");
    auto channel = (GETChannel*) chArray -> At(0);
    auto buffer = channel -> GetWaveformY();

    auto ana = new LKChannelAnalyzer();
    ana -> SetPulse("pulse.root");
    ana -> Analyze(buffer);
    ana -> Draw();
}
```

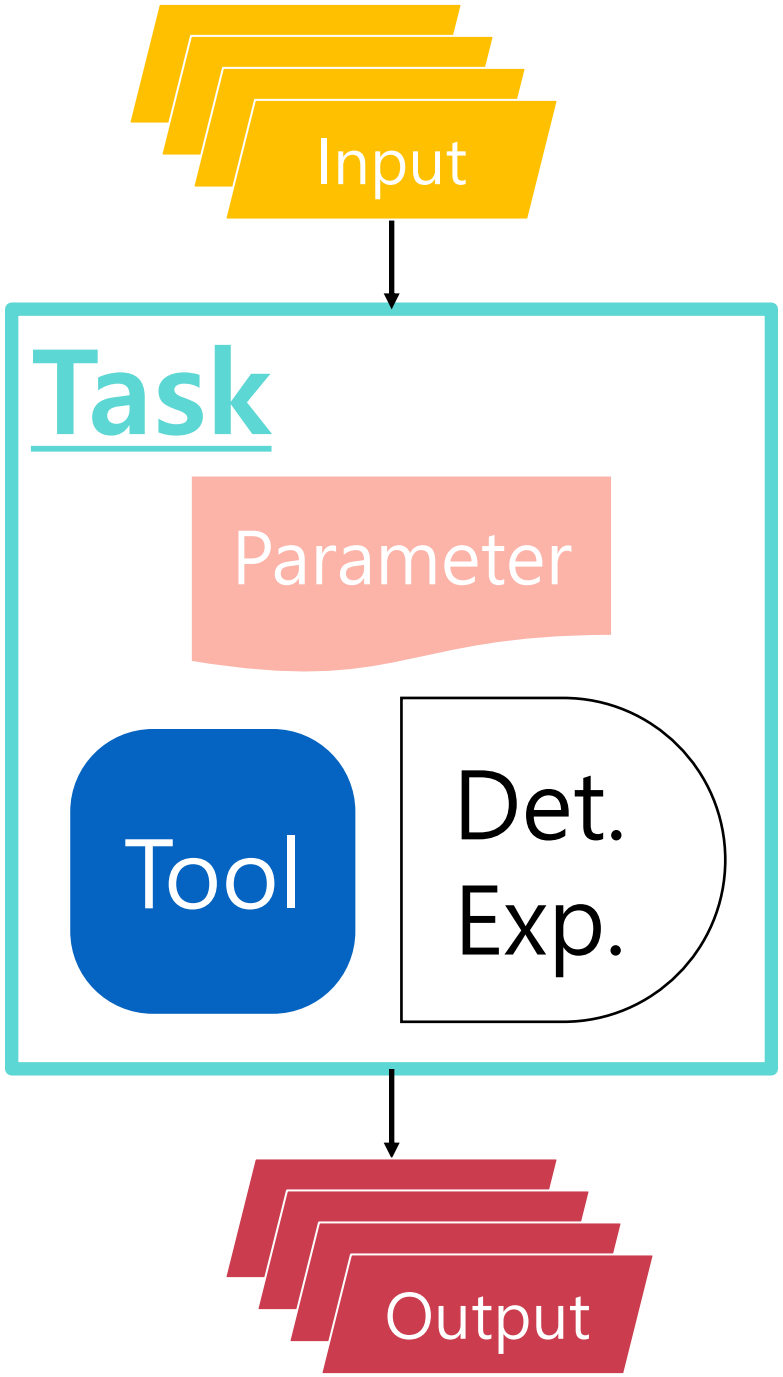



420
400
380
360
340
320
300
280
0
100
200
300
400
500
tb

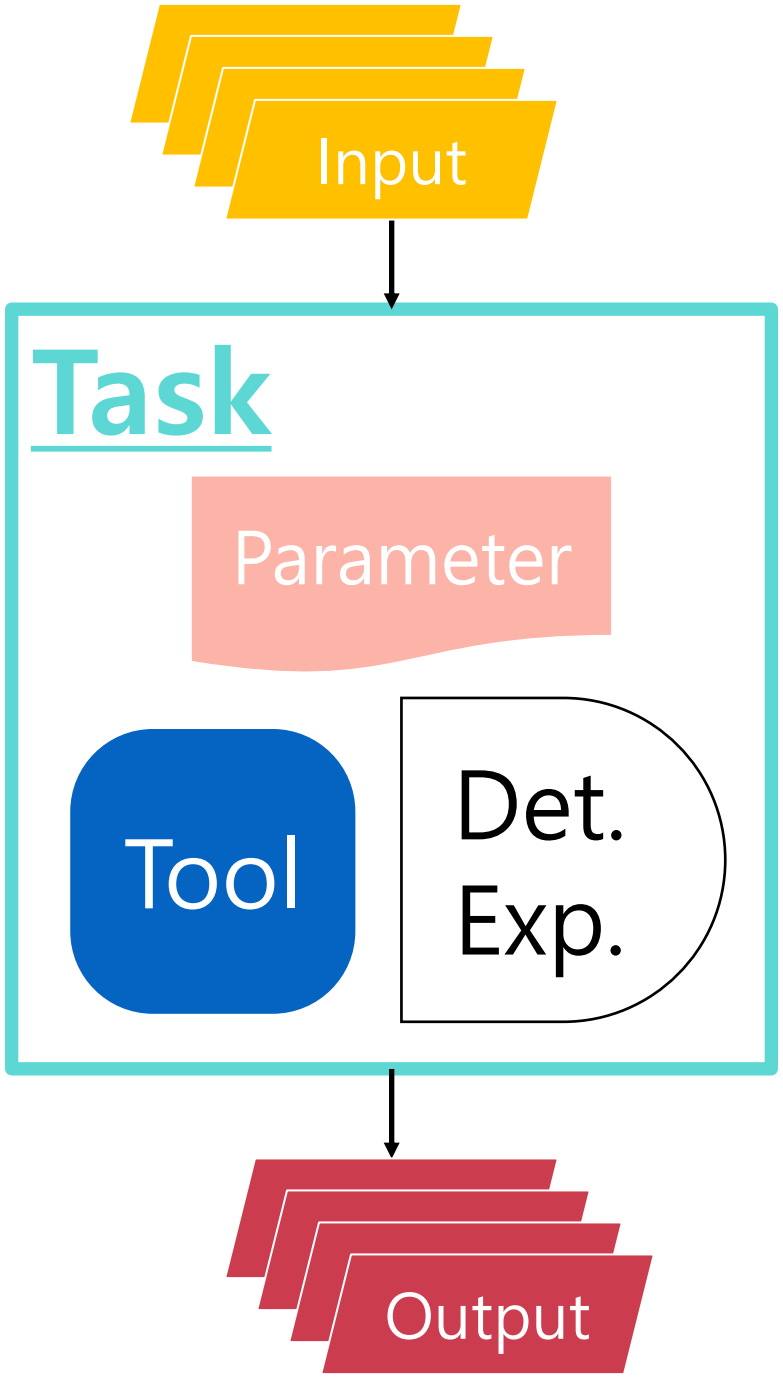

Input
Tool
Output

Task

- Examples of task:
  - Pulse Shape Analyze Task
  - Hough Transform Tracking Task
- List of tasks are executed in order.
- Executed for one triggered event at a time.
- Dedicated to experiment or detector.
- Input and output format is predefined inheriting TObject.



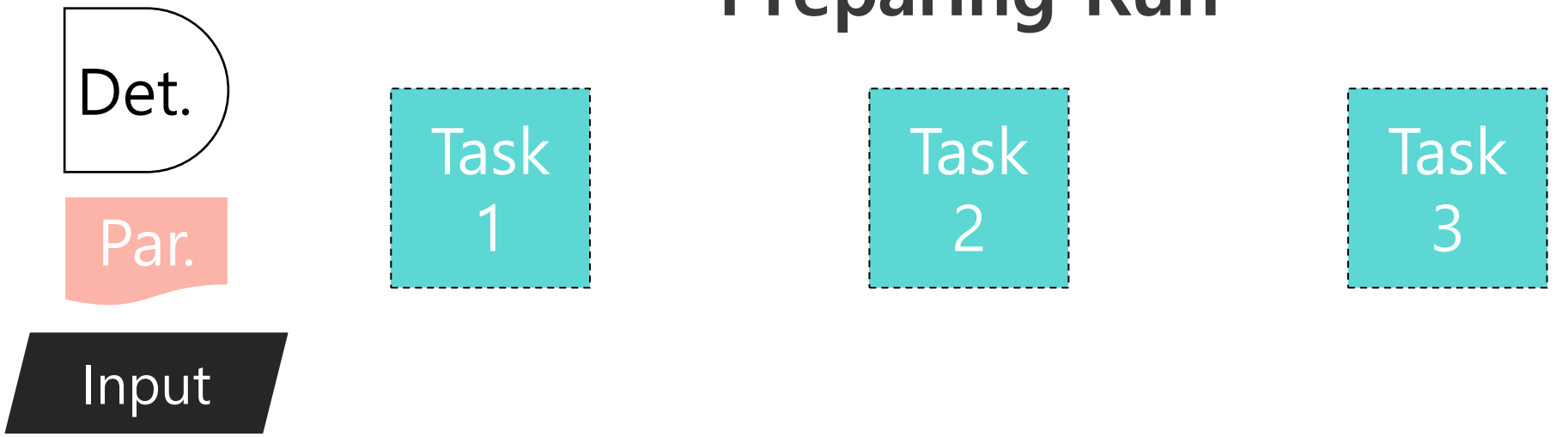
```
void example_task()
{
    auto run = new LKRun();
    run -> AddDetector(new MyDetector());
    run -> AddInputFile("data.root");
    run -> AddPar("config.mac");
    run -> Add(new LKPulseShapeAnalysisTask);
    run -> Add(new LTHelixTrackFindingTask);
    run -> Init();

    run -> Run();
}
```



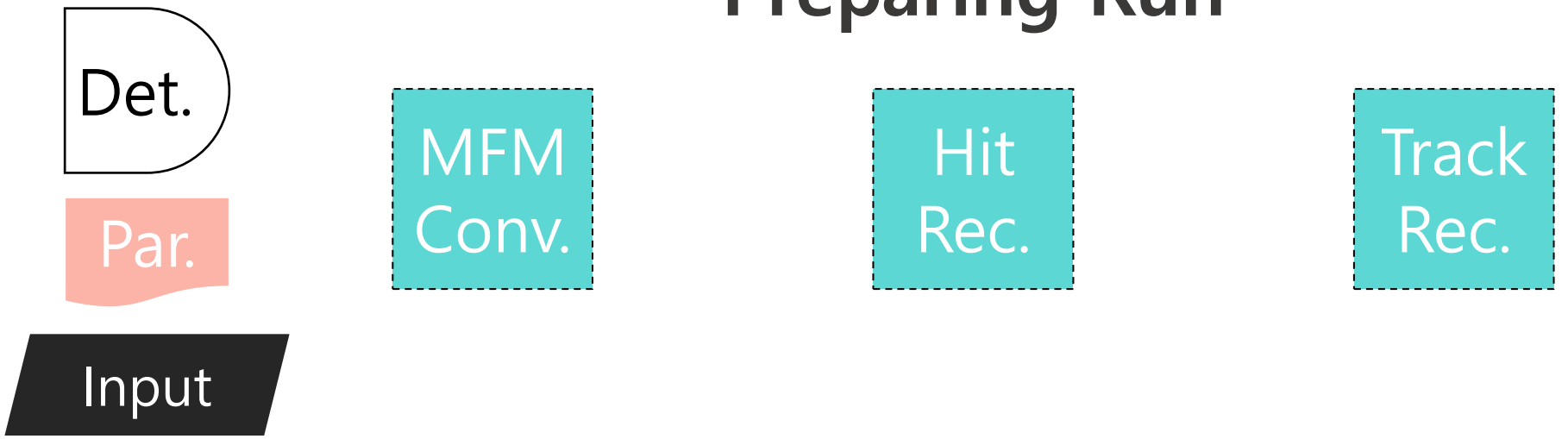
Container

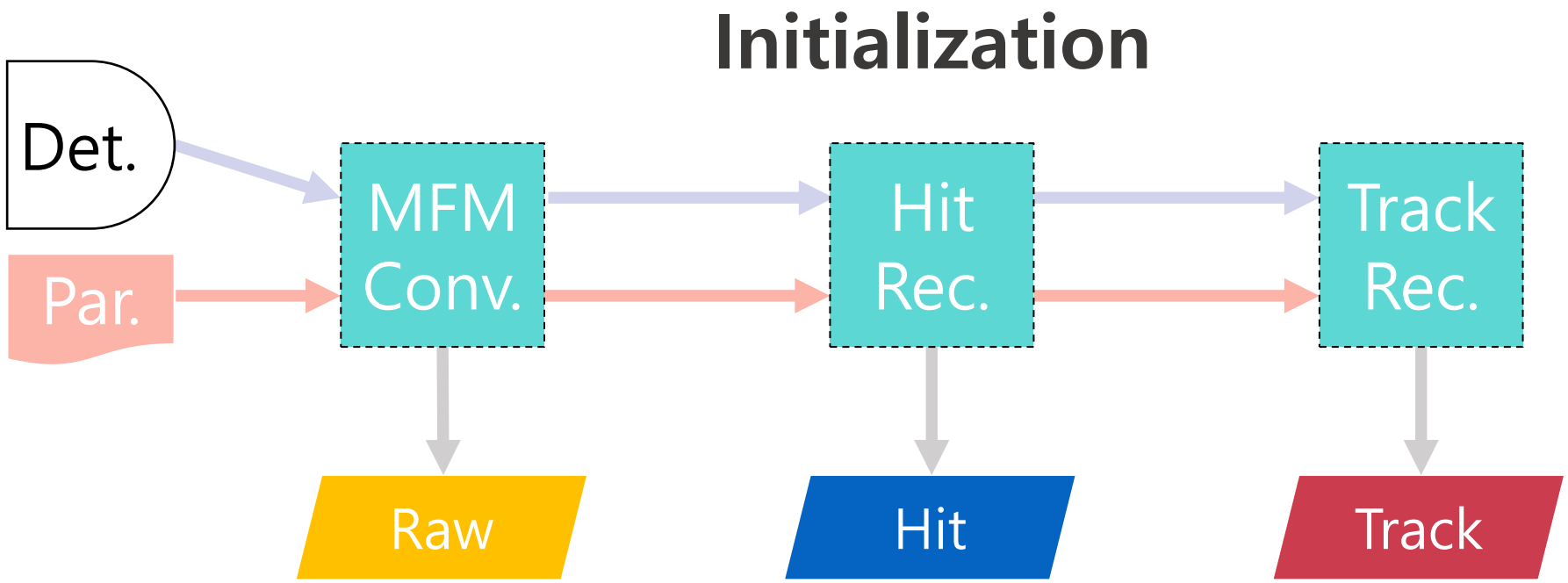
- Examples of container:
  - GETChannel
  - LKHit
  - LKLinearTrack
- Container is predefined TObject container.
- Creation of container is to prevent different variations of data explaining same type of data.

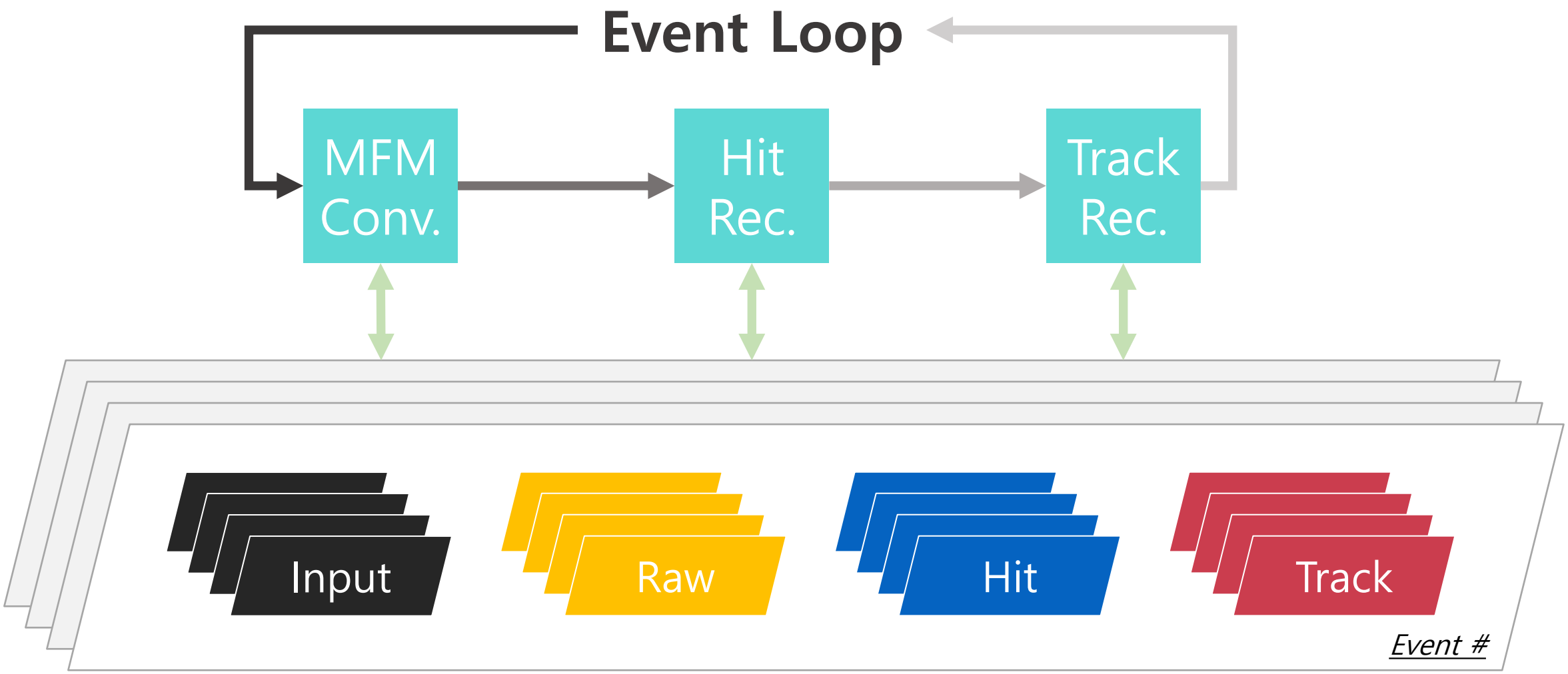
Preparing Run

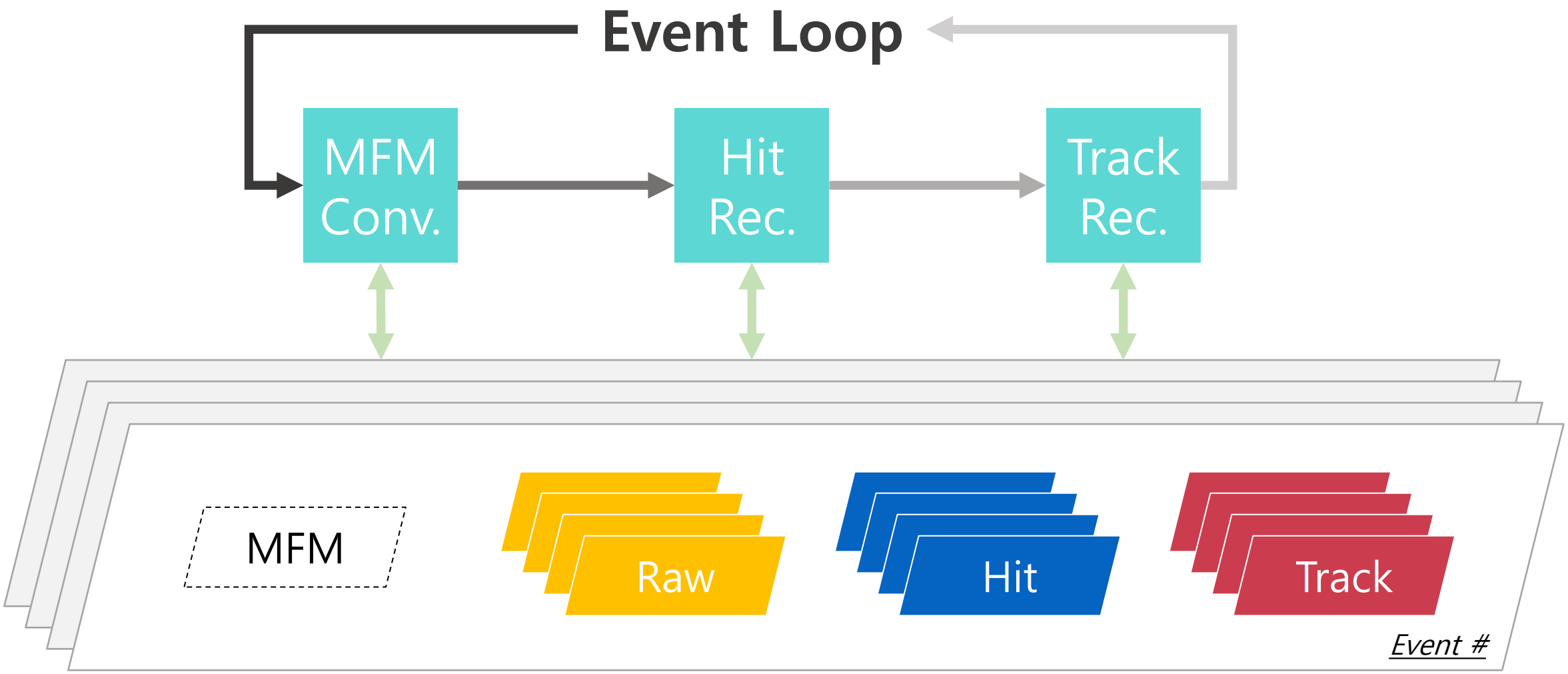


Preparing Run

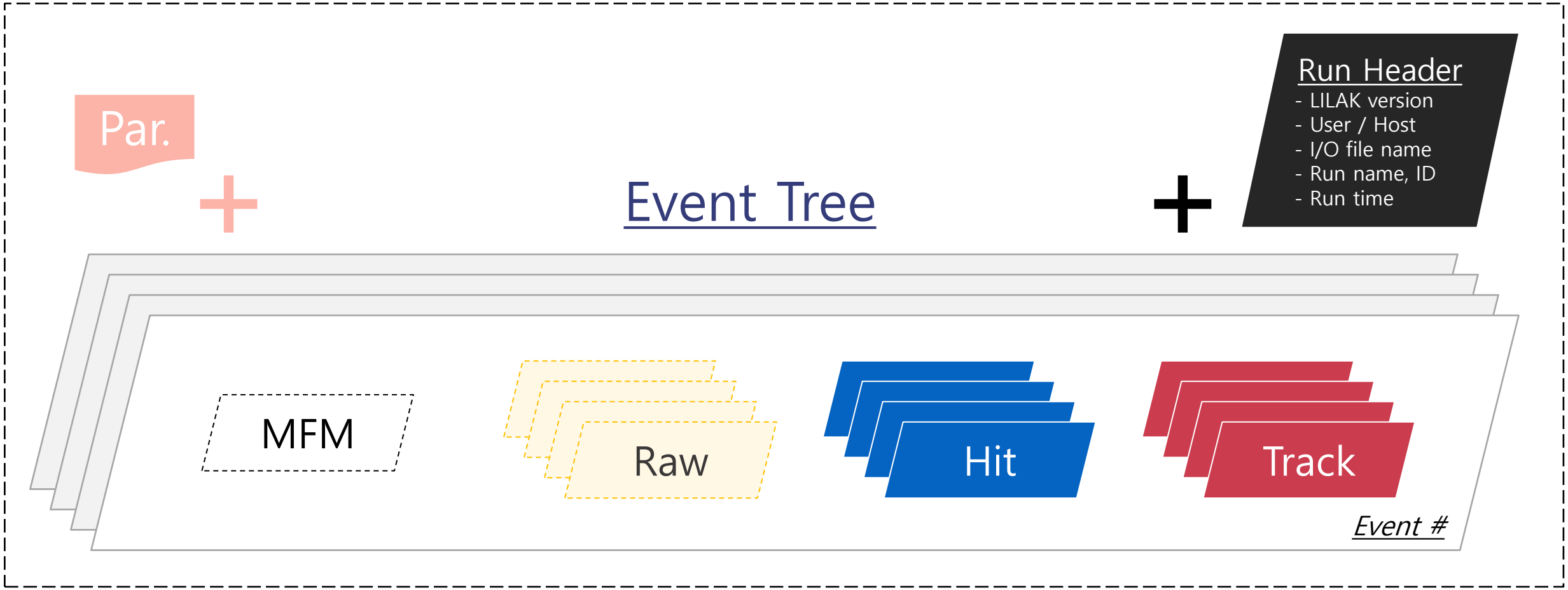



Initialization
Det.
Par.
MFM Conv.
Hit Rec.
Track Rec.
Raw
Hit
Track


Event Loop
MFM Conv.
Hit Rec.
Track Rec.
Input
Raw
Hit
Track
Event #


Event Loop
MFM Conv.
Hit Rec.
Track Rec.
MFM
Raw
Hit
Track
Event #

Output


Par.
+
Event Tree
+
Run Header
- LILAK version
- User / Host
- I/O file name
- Run name, ID
- Run time
MFM
Raw
Hit
Track
Event #


Par.
LKParameterContainer
Run Header
- LILAK version
- User / Host
- I/O file name
- Run name, ID
- Run time

Parameter Container

- LKParameterContainer, list of LKParameter.
- User can create configuration file as list of [name] [value].
- Store parameter as string type.
- Get types: bool, int, double, long, TString, vector<*>.
- Feature
  - Call another parameter file.
  - Parameter referencing within text file.
  - Basic operations (+, -, *, /, TMath, Color_t)
  - Array
  - Group
  - Comment
  - Conditional parameter
  - Special parameters: {lilak_data}, {lilak_common}, etc.

```
# par.mac

*LKRun/RunName  lilak 1 sim   # this is comment
*LKRun/DataPath {lilak_data}  # path/to/lilak/data

title        common parameter definition
dimension    70
length       {dimension}+30 # = 100
color        kRed+1         # = 633

pulseFile    {lilak_common}/pulseReference.root  # path/to/lilak/common/

persistency/  # group
    hit      false   # this parameter will be defined as "persistency/hit false"
    track    true    # this parameter will be defined as "persistency/track true"
```

```
void example_par()
{
    auto par = new LKParameterContainer();
    par -> AddFile("par.mac");
    par -> Print();
}
```

```
[ParameterContainer::AddFile] info> Adding parameter file /home/ejungwoo/lilak/macros_example/par.mac

[ParameterContainer::Print] info> Parameter Container ParameterContainer
   0. *LKRun/RunName       lilak 1 sim           # this is comment
   1. *LKRun/DataPath      /home/ejungwoo/lilak/data/      # lilak/data
   2. title                common parameter definitions
   3. dimension            70
   4. length               100    # = 100
   5. color                633    # = 633
   6. pulseFile            /home/ejungwoo/lilak/common//pulseReference.root    # lilak/common/...
   7. persistency/hit      false       # this parameter will be defined as "persistency/hit false"
   8. persistency/track    true   # this parameter will be defined as "persistency/track true"
[ParameterContainer::Print] info> End of Parameter Container ParameterContainer
```

```cpp
void example_run_header()
{
    auto file = new TFile("data.root");
    file -> Get("RunHeader") -> Print();
}
```

```
[RunHeader::Print] info> Parameter Container RunHeader
   0. MainP_Version        master.123.4bcbafc
   1. LILAK_Version        master.123.4bcbafc
   2. LILAK_HostName       CENS-ALPHA-00
   3. LILAK_UserName       cens-alpha-00
   4. LILAK_Path           /home/cens-alpha-00/lilak
   5. NumInputFiles        0
   6. InputFile
   7. OutputFile           data/lamps_0000.0.all.root
   8. RunName              lamps
   9. RunID                0
  10. RunTag               0.all
  11. start_time           1703049146
  12. start_ymd            2023.12.20
  13. start_hms            14:12:26
  14. end_time             1703049153
  15. end_ymd              2023.12.20
  16. end_hms              14:12:33
  17. run_time             54007
  18. run_time_s           0days 0h 0m 7s
[RunHeader::Print] info> End of Parameter Container RunHeader
```

Logger

- Create and export log file
  - `#include “LKLogger.h”`
  - `lk_logger(“my.log”);`

- Type of loggers
  - `lk_cout << “No header” << endl;`
  - `lk_info << “informative” << endl;`
  - `lk_warning << “warning” << endl;`
  - `lk_strong << “strong message” << endl;`
  - `lk_error << “error message” << endl;`
  - `lk_debug << “This is debug logger” << endl;`
    - `+21  /home/ejungwoo/lilak/macros_example/example_tool.C # This is debug logger`

Geant4 simulation

- LILAK Geant4 classes
  - LKG4RunManager
  - LKG4RunMessenger
  - LKMCEventGenerator
  - LKPrimaryGeneratorAction
  - LKEventAction
  - LKStackingAction
  - LKSteppingAction
  - LKTrackingAction

- Geometry should be built by user add following to collect data.
  - `(LKG4RunManager *) G4RunManager::GetRunManager() -> SetSensitiveDetector(solid_volume)`

MFM Converter

- Task: LKMFMConversionTask. (Event Trigger)
- Tool: LKFrameBuilder

```
#include "LKLogger.h"

void run_all()
{
    lk_logger("data/run_all.log");

    auto run = new LKRun();
    run -> SetRunName("texat",801,"all");
    run -> SetDataPath("data");
    run -> AddPar("config_all.mac");
    run -> AddDetector(new TexAT2());

    run -> SetEventTrigger(new LKMFMConversionTask());
    run -> Add(new TTEventPreviewTask());
    run -> Add(new TTPulseAnalysisTask());
    run -> Add(new TTHTTrackingTask());

    run -> SetEventCountForMessage(200);
    run -> Init();
    run -> Run();
}
```

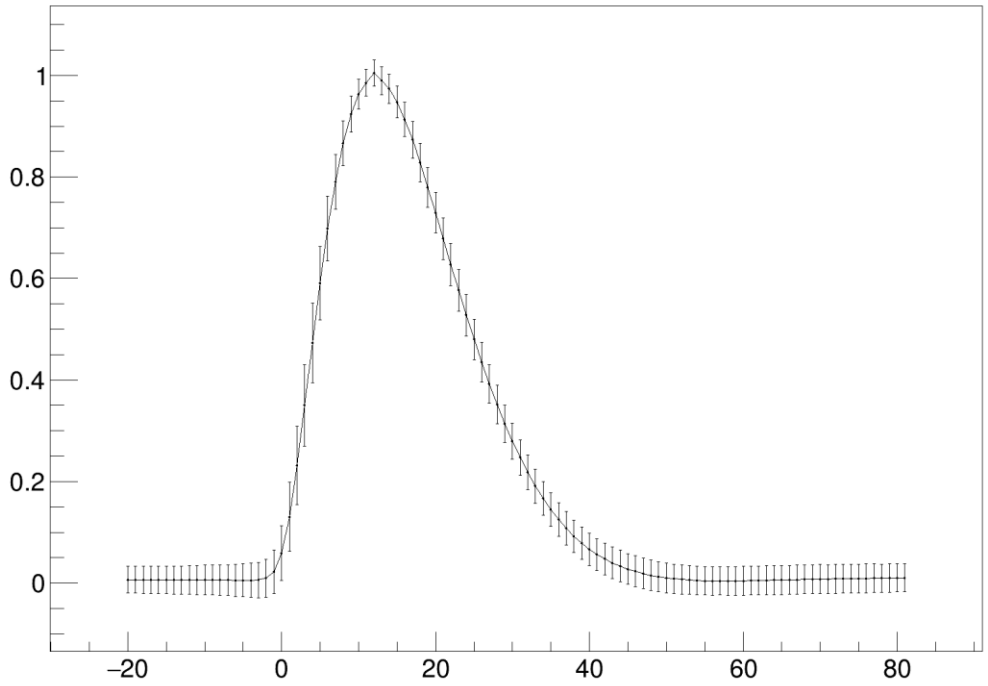
- mfm::FrameBuilder

1. Read file continuously
2. Build frames using internal method.
3. When event frame appears, processFrame() method is called.
4. Build channels and tell LKRun event is created.

Pulse Extraction

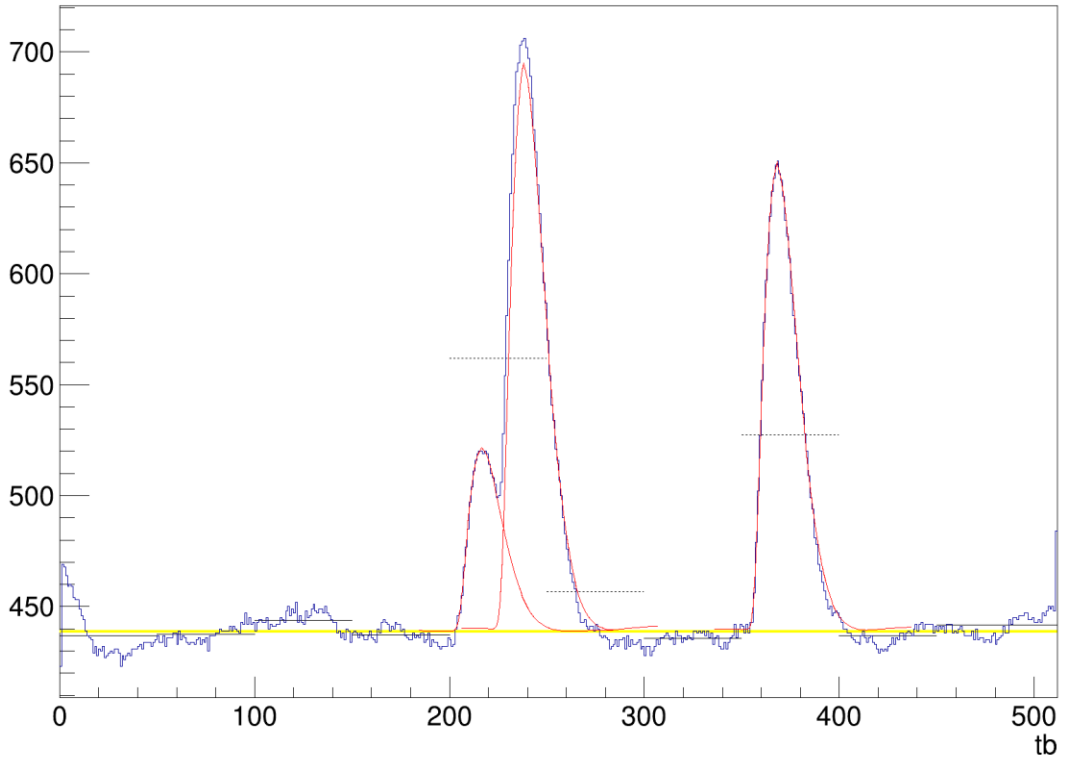
- Task: LKPulseExtractionTask
- Tool: LKPulseAnalyzer

```
> root pulseReference.root .
root [1] .ls
TFile**         pulseReference.root
 TFile*         pulseReference.root
  KEY: TGraphErrors     pulse;1
  KEY: TGraph    error;1
  KEY: TGraph    error0;1
  KEY: TParameter<int>  numAnaChannels;1
  KEY: TParameter<int>  threshold;1
  KEY: TParameter<int>  yMin;1
  KEY: TParameter<int>  yMax;1
  KEY: TParameter<int>  xMin;1
  KEY: TParameter<int>  xMax;1
  KEY: TParameter<double>       FWHM;1
  KEY: TParameter<double>       ratio;1
  KEY: TParameter<double>       width;1
  KEY: TParameter<double>       widthLeading;1
  KEY: TParameter<double>       widthTrailing;1
  KEY: TParameter<int>  pulseRefTbMin;1
  KEY: TParameter<int>  pulseRefTbMax;1
  KEY: TParameter<double>       backGroundLevel;1
  KEY: TParameter<double>       backGroundError;1
  KEY: TParameter<double>       fluctuationLevel;
  KEY: LKParameterContainer     RunHeader;1
root [2]
```



Pulse Shape Analysis

- Task: LKPulseShapeAnalysisTask
- Tool: LKChannelAnalyzer
- Input: Extracted pulse root file

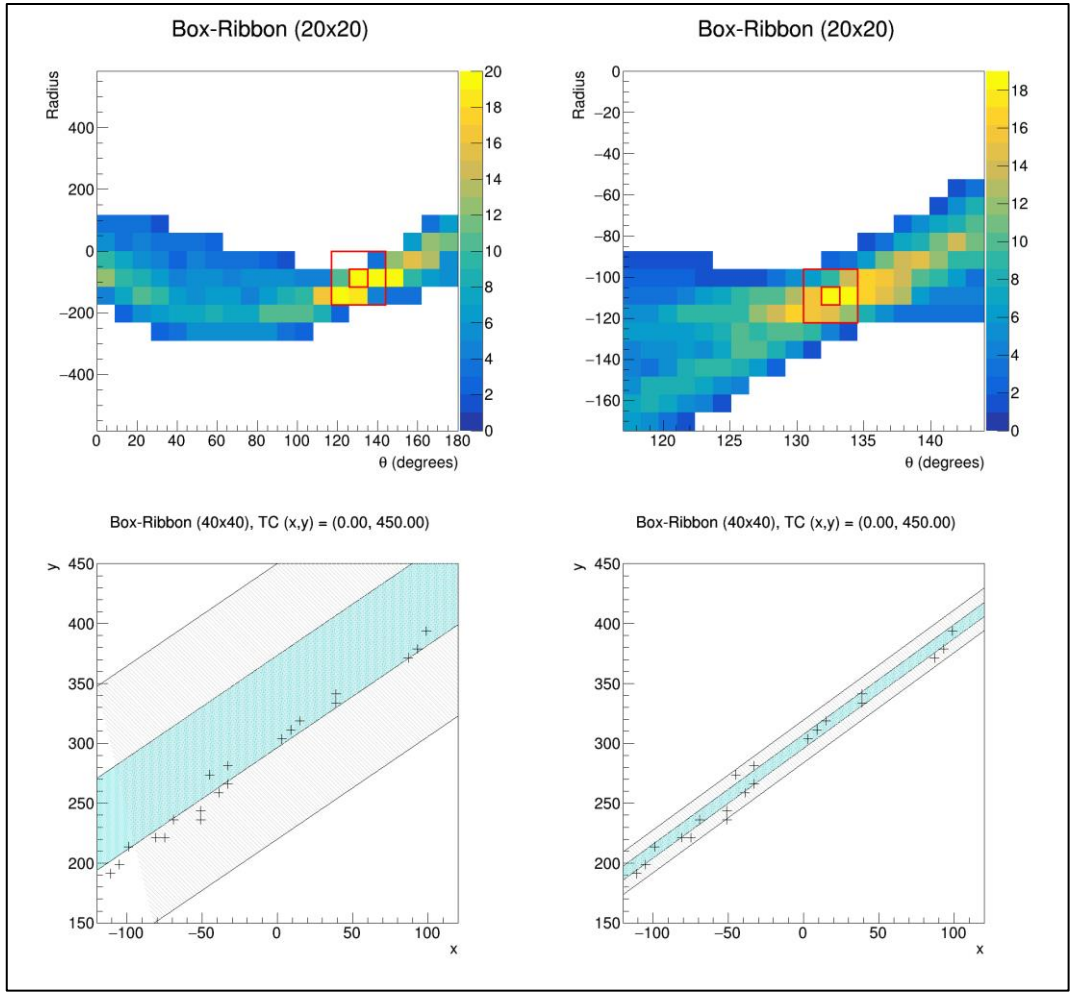


Hough Transform Tracker

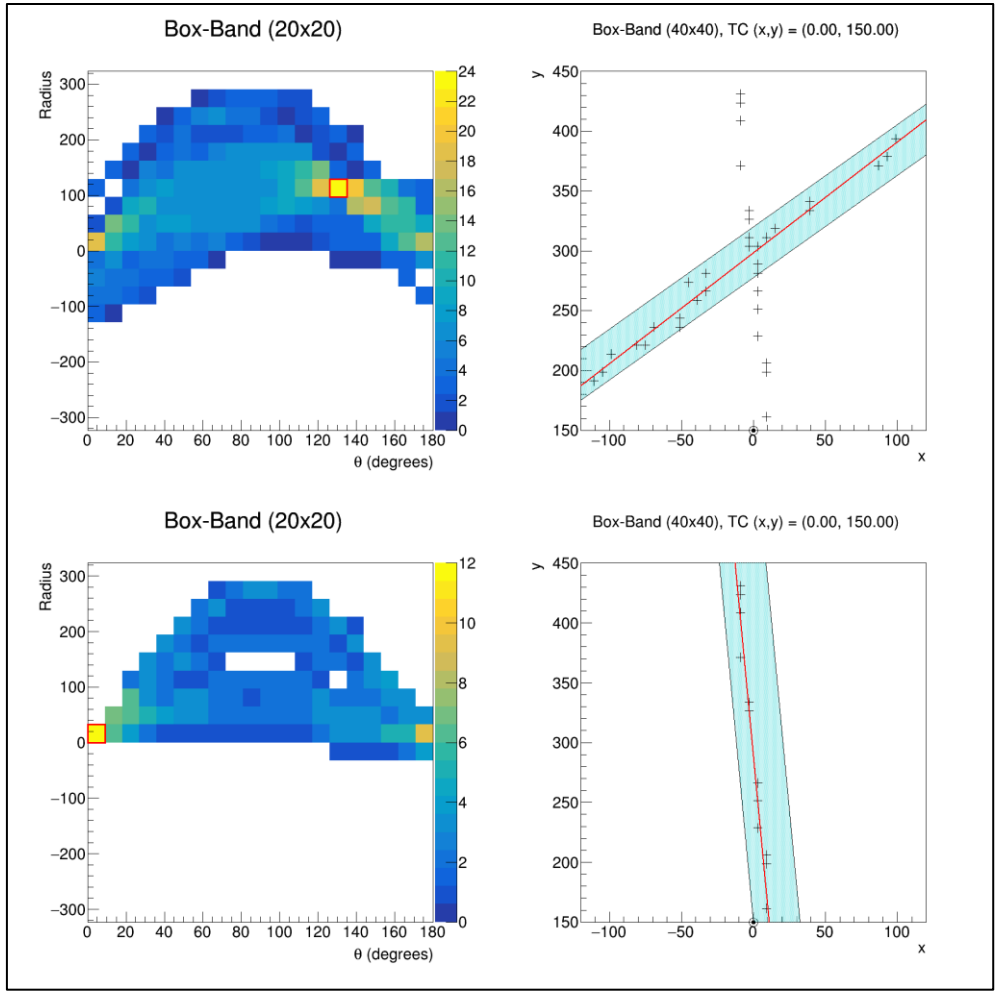
- Task: TTHTTrackingTask
- Tool: LKHTLineTracker

```
void example_ht()
{
     auto tracker = new LKHTLineTracker();
     tracker -> SetTransformCenter(TVector3(0,0,0));
     tracker -> SetImageSpaceRange(120, -150, 150, 120, 0, 500);
     tracker -> SetParamSpaceBins(numBinsR, numBinsT);
     for (...) {
         tracker -> AddImagePoint(x, xerror, y, yerror, weight);
     }
     tracker -> Transform();
     auto paramPoint = tracker -> FindNextMaximumParamPoint();
     track = tracker -> FitTrackWithParamPoint(paramPoint);
}
```

Zoom In



Multi-track fit



Helix Track Finding

- Task: TTHTTrackingTask
- Tool: LKHTLineTracker

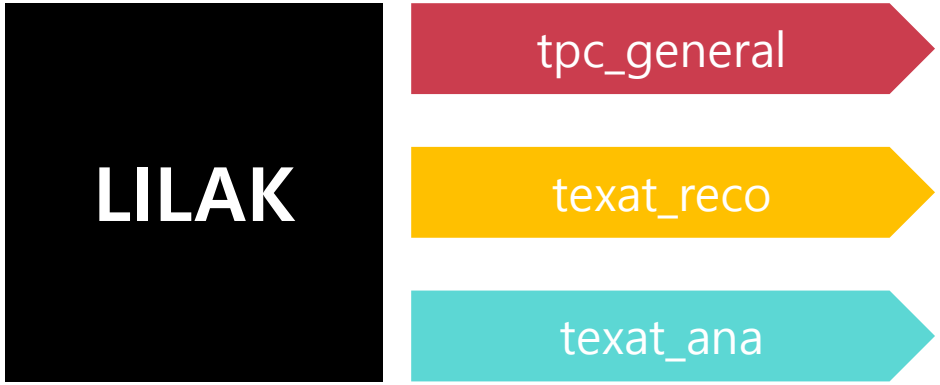
1. InitArray
2. NewTrack
3. RemoveTrack
4. **InitTrack**
5. **InitTrackAddHit**
6. **Continuum**
7. **ContinuumAddHit**
8. **Extrapolation**
9. **ExtrapolationAddHit**
10. **Confirmation**
11. **FinalizeTrack**
12. NextPhase
13. EndEvent

Track Reconstruction

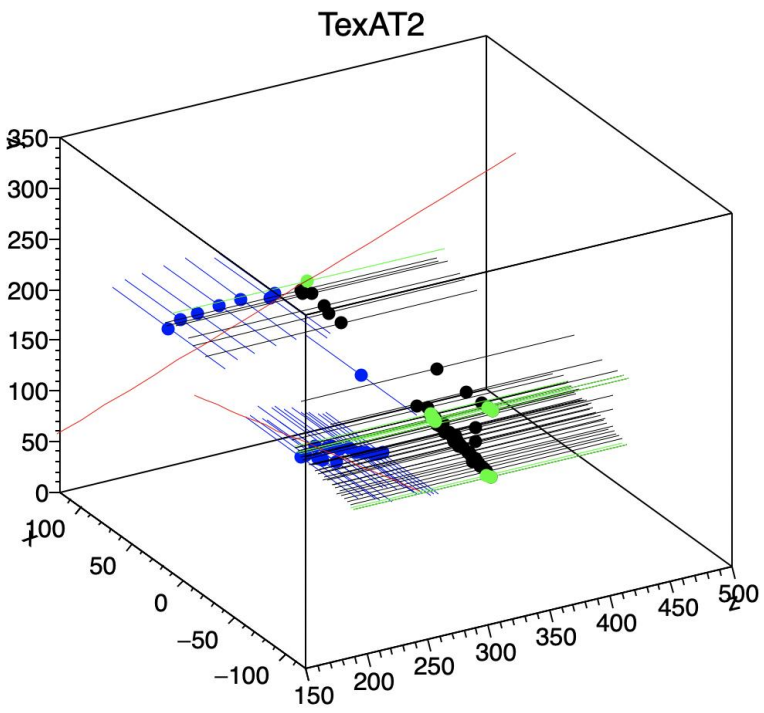
- Track finding
  - Hough transform
  - Track follower (bound with track fitter)
  - RANSAC (RANdom SAmple Consensus)
- Track fitting (direction / momentum)
  - Least chi-square fit
  - Riemann transform (indirect least chi-square method)
  - Hough transform
- Fine fitting (GENFIT, RAVE)

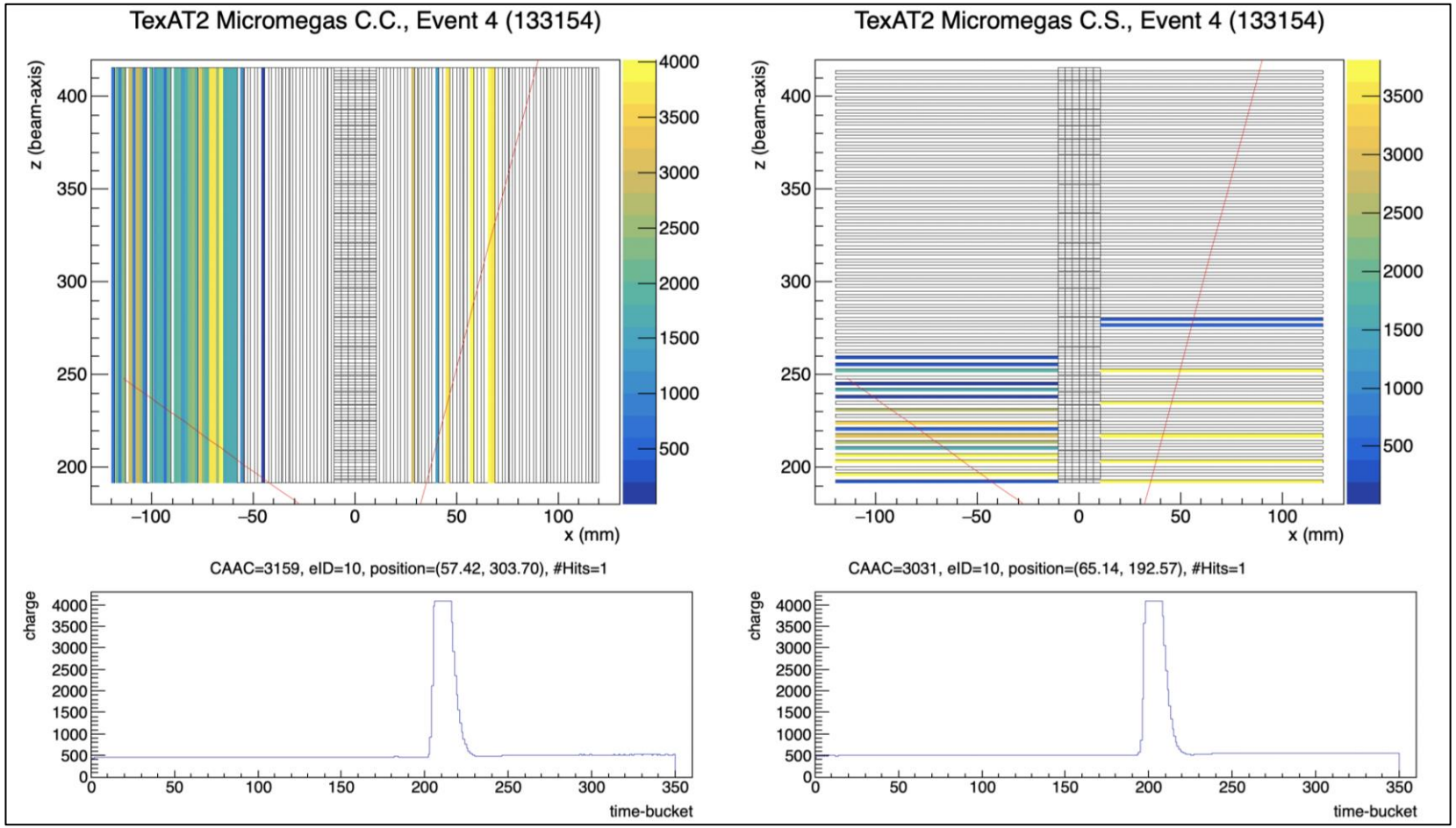
Project

- LILAK main source only contain general TPC classes.
- Detector / experiment dependent classes and macros should be written down separately.

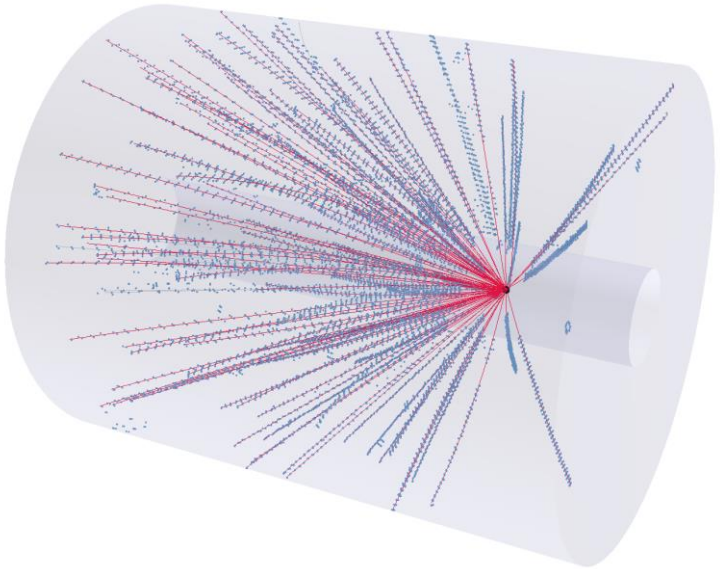


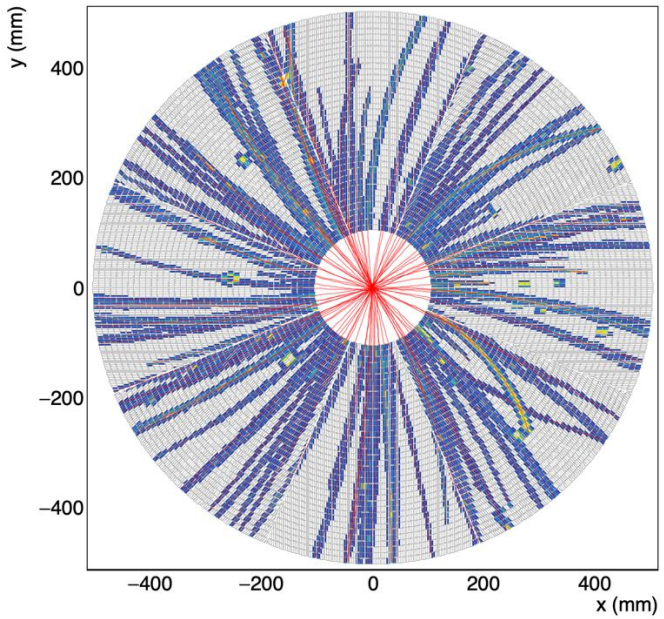
TexAT

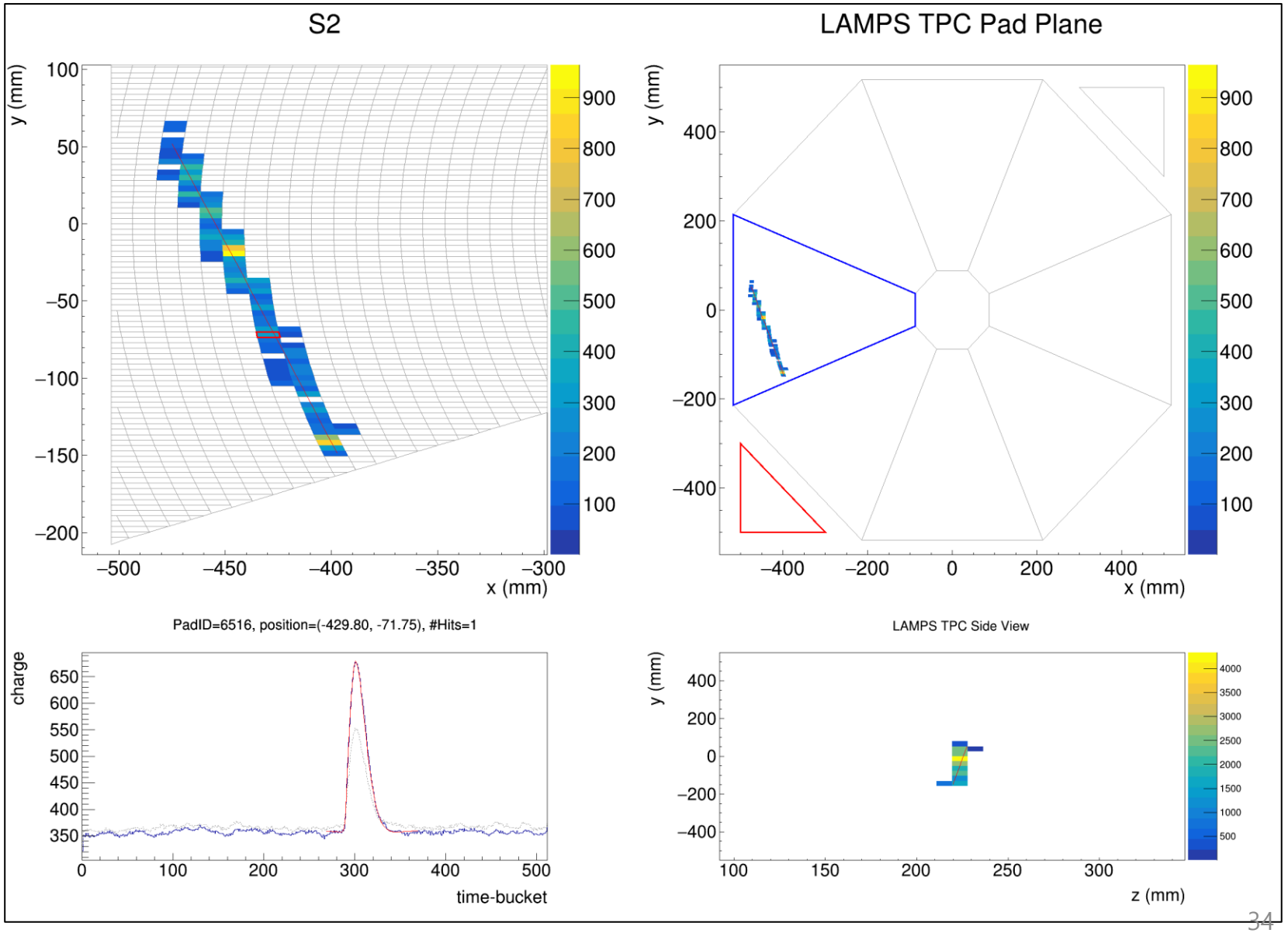

TexAT2


TexAT2 Micromegas C.C., Event 4 (133154)
TexAT2 Micromegas C.S., Event 4 (133154)
z (beam-axis)
x (mm)
CAAC=3159, eID=10, position=(57.42, 303.70), #Hits=1
CAAC=3031, eID=10, position=(65.14, 192.57), #Hits=1
charge
time-bucket

LAMPS TPC

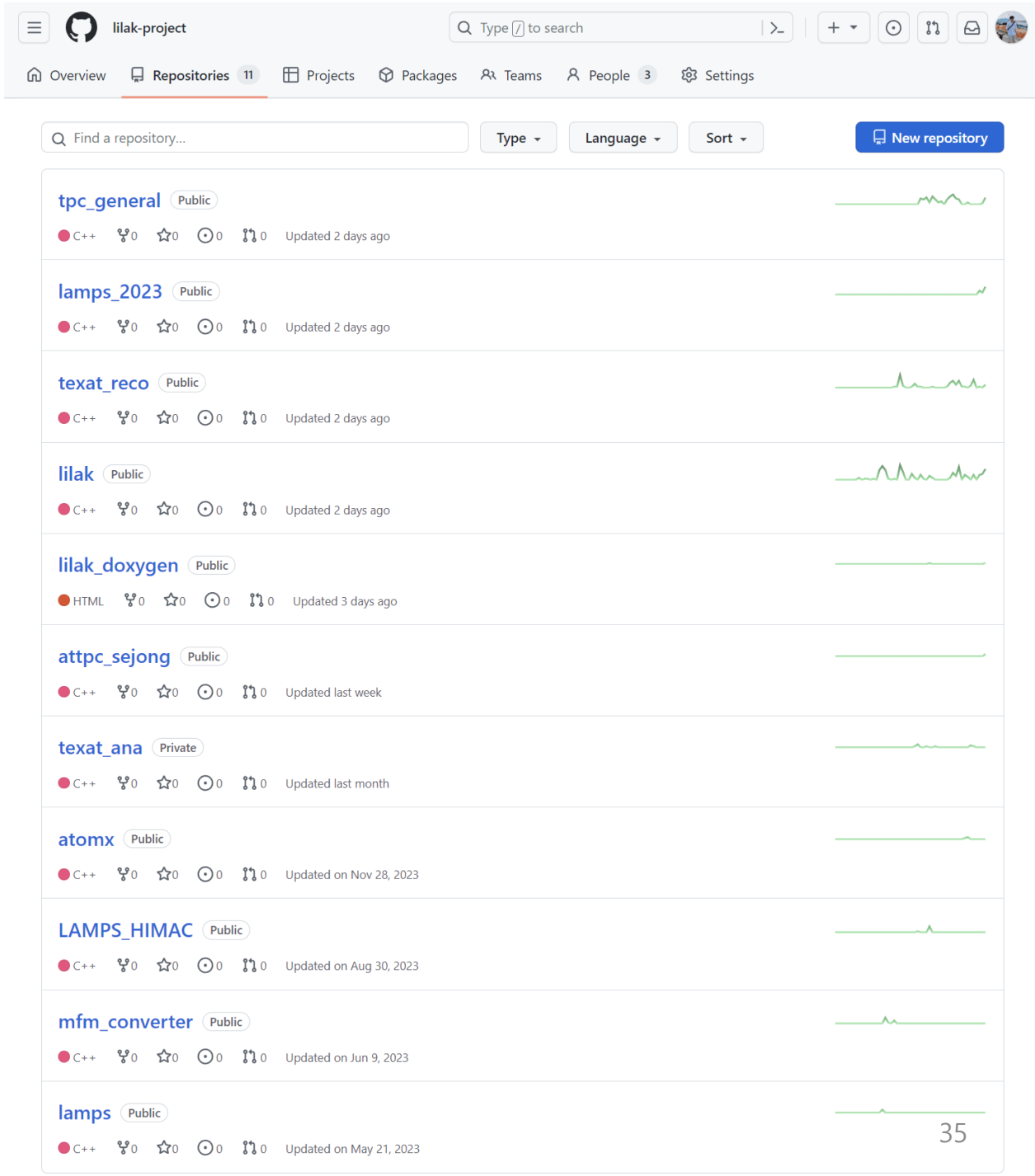



y (mm)
400
200
0
−200
−400
−400
−200
0
200
400
x (mm)


S2
y (mm)
100
50
0
−50
−100
−150
−200
−500
−450
−400
−350
−300
x (mm)
900
800
700
600
500
400
300
200
100
LAMPS TPC Pad Plane
y (mm)
400
200
0
−200
−400
−400
−200
0
200
400
x (mm)
PadID=6516, position=(-429.80, -71.75), #Hits=1
charge
650
600
550
500
450
400
350
0
100
200
300
400
500
time-bucket
LAMPS TPC Side View
y (mm)
400
200
0
−200
−400
100
150
200
250
300
z (mm)
4000
3500
3000
2500
2000
1500
1000
500

Download

- https://github.com/lilak-project



Creating project

- create_project.py

```
lilak> ./create_project.py

== 1 ========================================================================
## Creating new project

   Enter project name: myproject

   Project name is myproject

== 2 ========================================================================
## Creating project myproject

   LILAK   path is  /home/ejungwoo/llback
   Current path is  /home/ejungwoo/lilak
   Project path is  /home/ejungwoo/lilak/myproject


== 3 ========================================================================
## Creating sub-directories

   You can create dummy classes using scripts in macros/dummy_class_writer/

== 4 ========================================================================
## Creating macros

== 5 ========================================================================
## Creating container
   Enter class name to create container class <[name]/Enter>:

== 6 ========================================================================
## Creating detector
   Enter class name to create detector class <[name]/Enter>:

== 7 ========================================================================
## Creating common

== 8 ========================================================================
## Creating tool
   Enter class name to create tool class <[name]/Enter>:

== 9 ========================================================================
## Creating task
   Enter class name to create task class <[name]/Enter>:

== 10 =======================================================================
## Creating geant4
   Enter class name to create Geant4 Detector-Construction class <[name]/Enter>:

== 11 =======================================================================
## What now?

   It's all set! Now you might want to

   1. Setup git for the project
   2. Create and write up the classes.
   3. Compile the project, by running configure.py
   4. See https://github.com/lilak-project/lilak/wiki for more information
```

Install

- configure.py

```
./configure.py

== 1 ================================================================================
## LILAK configuration macro

   LILAK_PATH is
   /home/ejungwoo/lilak

== 2 ================================================================================
## Loading configuration from /home/ejungwoo/lilak/log/build_options.cmake

   ACTIVATE_EVE = False
   BUILD_GEANT4_SIM = False
   BUILD_MFM_CONVERTER = False
   BUILD_JSONCPP = False

   >> Main: lilak
   No Projects

Use above options? <Enter/0>: 0

== 3 ================================================================================
## Build options

    1) ACTIVATE_EVE
    2) BUILD_GEANT4_SIM
    3) BUILD_MFM_CONVERTER
    4) BUILD_JSONCPP

Type option number(s) to Add. Type <Enter> if non:

Selected option(s):

    --

== 4 ================================================================================
## Add Project

    1) texat_reco
    2) lamps_2023
    3) tpc_general
    4) texat_ana

Type project directory number(s) to Add. Type <Enter> if non: 123

Selected option(s):

    1) texat_reco
    2) lamps_2023
    3) tpc_general

== 5 ================================================================================
## Select main project

   >> Main: lilak
   1) Project: texat_reco
   2) Project: lamps_2023
   3) Project: tpc_general

Select index (or name) to set as main project. Type <Enter> to set main as lilak: 2
Main project is lamps_2023
```

Document

- Doxygen

LILAK master.139.138c03d

https://lilak-project.github.io/lilak_doxygen/

Main Page | Related Pages | Namespaces | Classes | Files

LILAK

Low- and Intermediate-energy nucLear experiment Analysis toolKit

How to build

Run `cofigure.py` and follow the instructions.

```
./configure.py
```

How to create project

Run `create_project.py` and follow the instructions.

```
./create_project.py
```

How to unlink lilak

Remove or comment out (#) Rint.Logon line as below in ~/.rootrc:

```
#Rint.Logon: /home/ejungwoo/lilak/macros/rootlogon.C
```

or run root with `-n` option when you run root to execute without logon and logoff macros.

```
root -n
```

Generated by doxygen 1.9.7

- LILAK
  - https://lilak-project.github.io/lilak_doxygen/
  - https://github.com/lilak-project
- CMSSW
  - https://github.com/cms-sw
  - https://cms-sw.github.io/
- Fun4All
  - https://www.phenix.bnl.gov/software/fun4all.html
  - https://www.phenix.bnl.gov/assets/fun4all/Fun4Spin.pdf
- FairRoot
  - https://fairroot.gsi.de/
- NPTool
  - https://nptool.in2p3.fr/