

ATOM

0.2

Generated by Doxygen 1.9.8

Chapter 1

Purpose of this project.

The project title "ATOM" is abbreviation of APTS Task Organisation Machine. This project is based on [MLR1 DAQ Software](#) made by CERN ALICE group. The existing code, however, was written by Python code. So the code was re-made by C++ and ROOT programming language for shorter runtime and efficient memory management.

Chapter 2

Bug List

Class [TAnalyser](#)

Class [TClusterAnalyser](#)

Class [TClusterShape](#)

Member [TClusterShape::clusterMap](#) (const [TMatrix2D< int >](#) *clusterMatrix)

The canvases and histograms have same name with each other.

Member [TClusterShape::identifyShapes](#) ()

Member [TClusterShape::sortShapes](#) (bool descend=true)

Member [TClusterShape::TClusterShape](#) ()

Struct [TShapeInfo](#)

Chapter 3

Todo List

Class `TAAnalyser`

Add template for plots. Map, distribution, etc.

Member `TAAnalyser::~~TAAnalyser ()`

The desctructors are commented out. It should be set.

Class `TClusterAnalyser`

Add Legend about experiment setting

Make more plots about cluster information

Class `TClusterShape`

It can be modified if needed.

Member `TClusterShape::clusterMap (const TMatrix2D< int > *clusterMatrix)`

Avoiding same name problem.

Member `TClusterShape::identifyShapes ()`

Member `TClusterShape::sortShapes (bool descend=true)`

More strict criteria is needed. If they have same entry, then it cannot be sorted and saved by coming order.

Member `TClusterShape::TClusterShape ()`

Struct `TShapeInfo`

Add struct member if needed.

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AnalysisManager	??
Argument	??
ArgumentParser	??
Colour	??
ControlExperimentAnalysis	??
ControlExperimentComparison	??
CppConfigDictionary	??
CppConfigFile	??
DACtoADC	??
EventTuple	??
std::exception	
ConfigurableNoValue	??
CppConfigFileError	??
MergeFileOpen	??
MergeTreeOpen	??
G4UserEventAction	
EventAction	??
G4UserRunAction	
RunAction	??
G4UserSteppingAction	
SteppingAction	??
G4UserTrackingAction	
TrackingAction	??
G4VUserActionInitialization	
ActionInitialization	??
G4VUserDetectorConstruction	
DetectorConstruction	??
G4VUserPrimaryGeneratorAction	
PrimaryGeneratorAction	??
HelpMessage	??
Material	??
ProgressBar	??
Quantity	??
RunTuple	??
Solid	??

StepTuple	??
TAnalyser	??
TClusterAnalyser	??
TClusterShapeAnalyser	??
TGeantAnalyser	??
TCluster	??
TClusterDivideData	??
TClusterization	??
TClusterShape	??
TDecoder	??
TALPIDDecoder	??
TAPTSDDecoder	??
TDetector	??
TDisk	??
TEntrySimulation	??
TEvent	??
TALPIDEvent	??
TAPTSEvent	??
TExperimentData	??
TGraph	
TGraphUser	??
TGraphCompare	??
TInputRoot	??
TMatrix2D< numT >	??
TMatrix2D< int >	??
TMerge	??
TMergeExperimentROOT	??
TrackTuple	??
TShapeInfo	??
TThreshold	??
TThresholdAnalyser	??
TThresholdCompare	??
TTimer	??
Unit	??

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ActionInitialization	??
AnalysisManager	??
Argument	??
ArgumentParser	??
Colour	??
ConfigurableNoValue	??
ControlExperimentAnalysis	??
ControlExperimentComparison	??
CppConfigDictionary	??
CppConfigFile	??
CppConfigFileError	??
DACtoADC	??
DetectorConstruction	??
EventAction	??
EventTuple	??
HelpMessage	??
Material	??
MergeFileOpen	??
MergeTreeOpen	??
PrimaryGeneratorAction	??
ProgressBar	??
Quantity	??
RunAction	??
RunTuple	??
Solid	??
SteppingAction	??
StepTuple	??
TALPIDDecoder	??
TALPIDEEvent	??
TAnalyser	
For ROOT and config file when analysis	??
TAPTSDDecoder	??
TAPTSEvent	??
TCluster	??
TClusterAnalyser	
Communicating execute file for controlling cluster research	??

TClusterDivideData	??
TClusterization	
Class of tools for clusterizing events for single event	??
TClusterShape	
Class for extracting cluster shape information with same cluster size	??
TClusterShapeAnalyser	??
TDecoder	??
TDetector	??
TDisk	??
TEntrySimulation	??
TEvent	??
TExperimentData	??
TGeantAnalyser	??
TGraphCompare	??
TGraphUser	
The general tools for drawing TGraph Class with config file	??
TInputRoot	??
TMatrix2D< numT >	??
TMerge	??
TMergeExperimentROOT	??
TrackingAction	??
TrackTuple	??
TShapeInfo	
The information set stucture for clusters that having homeomorphism shape	??
TThreshold	??
TThresholdAnalyser	??
TThresholdCompare	??
TTimer	??
Unit	??

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

/home/ychoi/ATOM/entryCalculator_copy.cpp	??
/home/ychoi/ATOM/graph_draw.cpp	??
/home/ychoi/ATOM/alpide/analysis/inc/TAnalyser.h	
The class for controlling ROOT and config file when analysing	??
/home/ychoi/ATOM/alpide/analysis/inc/TFileFormat.h	??
/home/ychoi/ATOM/alpide/analysis/inc/TThresholdAnalyser.h	
The tools for threshold analysis	??
/home/ychoi/ATOM/alpide/analysis/src/TAnalyser.cpp	??
/home/ychoi/ATOM/alpide/analysis/src/TThresholdAnalyser.cpp	??
/home/ychoi/ATOM/alpide/cluster/inc/TCluster.h	
The TCluster class header. Cluster properties are defined	??
/home/ychoi/ATOM/alpide/cluster/inc/TClusterDivideData.h	??
/home/ychoi/ATOM/alpide/cluster/inc/TClusterization.h	
Class for clusterizing events on a chip	??
/home/ychoi/ATOM/alpide/cluster/inc/TClusterShape.h	
Tools for extracting cluster shape image	??
/home/ychoi/ATOM/alpide/cluster/inc/TExperimentData.h	??
/home/ychoi/ATOM/alpide/cluster/inc/TMatrix2D.h	??
/home/ychoi/ATOM/alpide/cluster/src/TCluster.cpp	??
/home/ychoi/ATOM/alpide/cluster/src/TClusterDivideData.cpp	??
/home/ychoi/ATOM/alpide/cluster/src/TClusterization.cpp	??
/home/ychoi/ATOM/alpide/cluster/src/TClusterOperator.cpp	??
/home/ychoi/ATOM/alpide/cluster/src/TClusterShape.cpp	??
/home/ychoi/ATOM/alpide/cluster/src/TExperimentData.cpp	??
/home/ychoi/ATOM/alpide/comparison/inc/TGraphCompare.h	??
/home/ychoi/ATOM/alpide/comparison/inc/TMerge.h	
Tools for integrate same configure but seperate file	??
/home/ychoi/ATOM/alpide/comparison/src/TGraphCompare.cpp	??
/home/ychoi/ATOM/alpide/comparison/src/TMerge.cpp	??
/home/ychoi/ATOM/alpide/daq/inc/TALPIDEDecoder.h	??
/home/ychoi/ATOM/alpide/daq/inc/TALPIDEEvent.h	??
/home/ychoi/ATOM/alpide/daq/src/TALPIDEDecoder.cpp	??
/home/ychoi/ATOM/alpide/daq/src/TALPIDEEvent.cpp	??
/home/ychoi/ATOM/alpide/threshold/inc/TThreshold.h	??
/home/ychoi/ATOM/alpide/threshold/inc/TThresholdCompare.h	??

/home/ychoi/ATOM/alpide/threshold/src/TThreshold.cpp	??
/home/ychoi/ATOM/alpide/threshold/src/TThresholdCompare.cpp	??
/home/ychoi/ATOM/aps/inc/TAPTSDecoder.h	??
/home/ychoi/ATOM/aps/inc/TAPTSEvent.h	??
/home/ychoi/ATOM/aps/src/TAPTSDecoder.cpp	??
/home/ychoi/ATOM/aps/src/TAPTSEvent.cpp	??
/home/ychoi/ATOM/chip/inc/TDecoder.h	??
/home/ychoi/ATOM/chip/inc/TEvent.h	??
/home/ychoi/ATOM/chip/src/TDecoder.cpp	??
/home/ychoi/ATOM/chip/src/TEvent.cpp	??
/home/ychoi/ATOM/drawing_tool/inc/TGraphUser.h	??
/home/ychoi/ATOM/drawing_tool/inc/TH1User.h	??
/home/ychoi/ATOM/drawing_tool/src/TGraphUser.cpp	??
/home/ychoi/ATOM/exe/alpide_dac.cpp	??
/home/ychoi/ATOM/exe/alpide_hitmap.cpp	??
/home/ychoi/ATOM/exe/CompareExperimentData.cpp	??
/home/ychoi/ATOM/exe/entrySimulation.cpp	??
/home/ychoi/ATOM/exe/ExperimentAnalysis.cpp	??
/home/ychoi/ATOM/exe/ExperimentAnalysis.hpp	??
/home/ychoi/ATOM/exe/GarfieldSimulation.cpp	??
/home/ychoi/ATOM/exe/Merge.cpp	??
/home/ychoi/ATOM/exe/simulation.cpp	??
/home/ychoi/ATOM/exe/SimulationAnalysis.cpp	??
/home/ychoi/ATOM/exe/ThresholdAnalysis.cpp	??
/home/ychoi/ATOM/geant4/analysis/inc/TGeantAnalyser.h	??
/home/ychoi/ATOM/geant4/analysis/src/TGeantAnalyser.cpp	??
/home/ychoi/ATOM/geant4/main/inc/ActionInitialization.h	??
/home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h	??
/home/ychoi/ATOM/geant4/main/inc/Colour.h	??
/home/ychoi/ATOM/geant4/main/inc/DetectorConstruction.h	??
/home/ychoi/ATOM/geant4/main/inc/EventAction.h	??
/home/ychoi/ATOM/geant4/main/inc/Material.h	??
/home/ychoi/ATOM/geant4/main/inc/PrimaryGeneratorAction.h	??
/home/ychoi/ATOM/geant4/main/inc/RunAction.h	??
/home/ychoi/ATOM/geant4/main/inc/Solid.h	??
/home/ychoi/ATOM/geant4/main/inc/SteppingAction.h	??
/home/ychoi/ATOM/geant4/main/inc/TrackingAction.h	??
/home/ychoi/ATOM/geant4/main/src/ActionInitialization.cpp	??
/home/ychoi/ATOM/geant4/main/src/AnalysisManager.cpp	??
/home/ychoi/ATOM/geant4/main/src/Colour.cpp	??
/home/ychoi/ATOM/geant4/main/src/DetectorConstruction.cpp	??
/home/ychoi/ATOM/geant4/main/src/EventAction.cpp	??
/home/ychoi/ATOM/geant4/main/src/Material.cpp	??
/home/ychoi/ATOM/geant4/main/src/PrimaryGeneratorAction.cpp	??
/home/ychoi/ATOM/geant4/main/src/RunAction.cpp	??
/home/ychoi/ATOM/geant4/main/src/Solid.cpp	??
/home/ychoi/ATOM/geant4/main/src/SteppingAction.cpp	??
/home/ychoi/ATOM/geant4/main/src/TrackingAction.cpp	??
/home/ychoi/ATOM/pycpp/config/inc/CppConfigDictionary.h	??
/home/ychoi/ATOM/pycpp/config/inc/CppConfigError.h	??
/home/ychoi/ATOM/pycpp/config/inc/CppConfigFile.h	??
/home/ychoi/ATOM/pycpp/config/src/CppConfigDictionary.cpp	??
/home/ychoi/ATOM/pycpp/config/src/CppConfigError.cpp	??
/home/ychoi/ATOM/pycpp/config/src/CppConfigFile.cpp	??
/home/ychoi/ATOM/pycpp/inc/cppargs.h	??
/home/ychoi/ATOM/pycpp/inc/cppTimer.h	??
/home/ychoi/ATOM/pycpp/inc/cpptqdm.h	??
/home/ychoi/ATOM/pycpp/inc/cppUnit.h	??

/home/ychoi/ATOM/pycpp/src/ cppargs.cpp	??
/home/ychoi/ATOM/pycpp/src/ cppTimer.cpp	??
/home/ychoi/ATOM/pycpp/src/ cpptqdm.cpp	??
/home/ychoi/ATOM/pycpp/src/ cppUnit.cpp	??
/home/ychoi/ATOM/simulation/inc/ TEntrySimulation.h	??
/home/ychoi/ATOM/simulation/src/ TEntrySimulation.cpp	??
/home/ychoi/ATOM/trashcan/ TClusterAnalyser.cpp	??
/home/ychoi/ATOM/trashcan/ TClusterAnalyser.h	
Control cluster analysis process and save plots	??
/home/ychoi/ATOM/trashcan/ TClusterShapeAnalyser.cpp	
Tools for analysing shape property of cluster	??
/home/ychoi/ATOM/trashcan/ TClusterShapeAnalyser.h	
Tools for analysing and drawing cluster shape	??

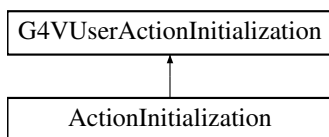
Chapter 7

Class Documentation

7.1 ActionInitialization Class Reference

```
#include <ActionInitialization.h>
```

Inheritance diagram for ActionInitialization:



Public Member Functions

- [ActionInitialization](#) ()
- virtual [~ActionInitialization](#) ()
- virtual void [Build](#) () const
- virtual void [BuildForMaster](#) () const

7.1.1 Detailed Description

Definition at line 16 of file [ActionInitialization.h](#).

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ActionInitialization()

```
ActionInitialization::ActionInitialization ( )
```

Definition at line 4 of file [ActionInitialization.cpp](#).

7.1.2.2 ~ActionInitialization()

ActionInitialization::~~ActionInitialization () [virtual]

Definition at line 8 of file [ActionInitialization.cpp](#).

7.1.3 Member Function Documentation

7.1.3.1 Build()

void ActionInitialization::Build () const [virtual]

Definition at line 12 of file [ActionInitialization.cpp](#).

7.1.3.2 BuildForMaster()

void ActionInitialization::BuildForMaster () const [virtual]

Definition at line 30 of file [ActionInitialization.cpp](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/ActionInitialization.h](#)
- [/home/ychoi/ATOM/geant4/main/src/ActionInitialization.cpp](#)

7.2 AnalysisManager Class Reference

```
#include <AnalysisManager.h>
```

Public Member Functions

- [AnalysisManager](#) ()
- [AnalysisManager](#) (std::string name)
- [~AnalysisManager](#) ()
- void [setEventID](#) (const int id)
- void [setTrackID](#) (const int id)
- int [getEventID](#) () const
- int [getTrackID](#) () const
- void [openBook](#) (std::string name)
- void [setRunTree](#) ()
- void [setEventTree](#) ()
- void [setTrackTree](#) ()
- void [setStepTree](#) ()
- void [RecordingRun](#) (const G4Run *run)
- void [RecordingEvent](#) (const G4Event *event)
- void [RecordingTrackStart](#) (const G4Track *track)
- void [RecordingTrackEnd](#) (const G4Track *track)
- void [RecordingStep](#) (const G4Step *step)
- void [closeBook](#) ()
- [RunTuple](#) & [getRunTuple](#) ()
- [EventTuple](#) & [getEventTuple](#) ()
- [TrackTuple](#) & [getTrackTuple](#) ()
- [StepTuple](#) & [getStepTuple](#) ()

Static Public Member Functions

- static [AnalysisManager](#) * [Instance](#) ()

Private Attributes

- int [runID](#) = 0
- int [eventID](#) = 0
- int [trackID](#) = 0
- int [stepID](#) = 0
- TFile * [mOutputFile](#)
- TTree * [mRunTree](#)
- TTree * [mEventTree](#)
- TTree * [mTrackTree](#)
- TTree * [mStepTree](#)
- [RunTuple](#) [runTuple](#)
- [EventTuple](#) [eventTuple](#)
- [TrackTuple](#) [trackTuple](#)
- [StepTuple](#) [stepTuple](#)

Static Private Attributes

- static [AnalysisManager](#) * [fInstance](#) = nullptr

7.2.1 Detailed Description

Definition at line 64 of file [AnalysisManager.h](#).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AnalysisManager() [1/2]

```
AnalysisManager::AnalysisManager ( )
```

Definition at line 6 of file [AnalysisManager.cpp](#).

7.2.2.2 AnalysisManager() [2/2]

```
AnalysisManager::AnalysisManager (
    std::string name )
```

Definition at line 11 of file [AnalysisManager.cpp](#).

7.2.2.3 ~AnalysisManager()

```
AnalysisManager::~AnalysisManager ( )
```

Definition at line 17 of file [AnalysisManager.cpp](#).

7.2.3 Member Function Documentation

7.2.3.1 closeBook()

```
void AnalysisManager::closeBook ( )
```

Definition at line 157 of file [AnalysisManager.cpp](#).

7.2.3.2 getEventID()

```
int AnalysisManager::getEventID ( ) const [inline]
```

Definition at line 92 of file [AnalysisManager.h](#).

7.2.3.3 getEventTuple()

```
EventTuple & AnalysisManager::getEventTuple ( ) [inline]
```

Definition at line 111 of file [AnalysisManager.h](#).

7.2.3.4 getRunTuple()

```
RunTuple & AnalysisManager::getRunTuple ( ) [inline]
```

Definition at line 110 of file [AnalysisManager.h](#).

7.2.3.5 getStepTuple()

```
StepTuple & AnalysisManager::getStepTuple ( ) [inline]
```

Definition at line 113 of file [AnalysisManager.h](#).

7.2.3.6 getTrackID()

```
int AnalysisManager::getTrackID ( ) const [inline]
```

Definition at line 93 of file [AnalysisManager.h](#).

7.2.3.7 getTrackTuple()

```
TrackTuple & AnalysisManager::getTrackTuple ( ) [inline]
```

Definition at line 112 of file [AnalysisManager.h](#).

7.2.3.8 Instance()

```
AnalysisManager * AnalysisManager::Instance ( ) [static]
```

Definition at line 21 of file [AnalysisManager.cpp](#).

7.2.3.9 openBook()

```
void AnalysisManager::openBook (
    std::string name )
```

Definition at line 28 of file [AnalysisManager.cpp](#).

7.2.3.10 RecordingEvent()

```
void AnalysisManager::RecordingEvent (
    const G4Event * event )
```

Definition at line 96 of file [AnalysisManager.cpp](#).

7.2.3.11 RecordingRun()

```
void AnalysisManager::RecordingRun (
    const G4Run * run )
```

Definition at line 88 of file [AnalysisManager.cpp](#).

7.2.3.12 RecordingStep()

```
void AnalysisManager::RecordingStep (
    const G4Step * step )
```

Definition at line 133 of file [AnalysisManager.cpp](#).

7.2.3.13 RecordingTrackEnd()

```
void AnalysisManager::RecordingTrackEnd (
    const G4Track * track )
```

Definition at line 125 of file [AnalysisManager.cpp](#).

7.2.3.14 RecordingTrackStart()

```
void AnalysisManager::RecordingTrackStart (
    const G4Track * track )
```

Definition at line 105 of file [AnalysisManager.cpp](#).

7.2.3.15 setEventID()

```
void AnalysisManager::setEventID (
    const int id ) [inline]
```

Definition at line 90 of file [AnalysisManager.h](#).

7.2.3.16 setEventTree()

```
void AnalysisManager::setEventTree ( )
```

Definition at line 42 of file [AnalysisManager.cpp](#).

7.2.3.17 setRunTree()

```
void AnalysisManager::setRunTree ( )
```

Definition at line 36 of file [AnalysisManager.cpp](#).

7.2.3.18 setStepTree()

```
void AnalysisManager::setStepTree ( )
```

Definition at line 70 of file [AnalysisManager.cpp](#).

7.2.3.19 setTrackID()

```
void AnalysisManager::setTrackID (
    const int id ) [inline]
```

Definition at line 91 of file [AnalysisManager.h](#).

7.2.3.20 setTrackTree()

```
void AnalysisManager::setTrackTree ( )
```

Definition at line 50 of file [AnalysisManager.cpp](#).

7.2.4 Member Data Documentation

7.2.4.1 eventID

```
int AnalysisManager::eventID = 0 [private]
```

Definition at line 67 of file [AnalysisManager.h](#).

7.2.4.2 eventTuple

```
EventTuple AnalysisManager::eventTuple [private]
```

Definition at line 80 of file [AnalysisManager.h](#).

7.2.4.3 fInstance

```
AnalysisManager * AnalysisManager::fInstance = nullptr [static], [private]
```

Definition at line 71 of file [AnalysisManager.h](#).

7.2.4.4 mEventTree

```
TTree* AnalysisManager::mEventTree [private]
```

Definition at line 75 of file [AnalysisManager.h](#).

7.2.4.5 mOutputFile

```
TFile* AnalysisManager::mOutputFile [private]
```

Definition at line 73 of file [AnalysisManager.h](#).

7.2.4.6 mRunTree

```
TTree* AnalysisManager::mRunTree [private]
```

Definition at line 74 of file [AnalysisManager.h](#).

7.2.4.7 mStepTree

```
TTree* AnalysisManager::mStepTree [private]
```

Definition at line 77 of file [AnalysisManager.h](#).

7.2.4.8 mTrackTree

```
TTree* AnalysisManager::mTrackTree [private]
```

Definition at line 76 of file [AnalysisManager.h](#).

7.2.4.9 runID

```
int AnalysisManager::runID = 0 [private]
```

Definition at line 66 of file [AnalysisManager.h](#).

7.2.4.10 runTuple

```
RunTuple AnalysisManager::runTuple [private]
```

Definition at line 79 of file [AnalysisManager.h](#).

7.2.4.11 stepID

```
int AnalysisManager::stepID = 0 [private]
```

Definition at line 69 of file [AnalysisManager.h](#).

7.2.4.12 stepTuple

```
StepTuple AnalysisManager::stepTuple [private]
```

Definition at line 82 of file [AnalysisManager.h](#).

7.2.4.13 trackID

```
int AnalysisManager::trackID = 0 [private]
```

Definition at line 68 of file [AnalysisManager.h](#).

7.2.4.14 trackTuple

```
TrackTuple AnalysisManager::trackTuple [private]
```

Definition at line 81 of file [AnalysisManager.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h](#)
- [/home/ychoi/ATOM/geant4/main/src/AnalysisManager.cpp](#)

7.3 Argument Class Reference

```
#include <cppargs.h>
```


Public Member Functions

- [Argument](#) (std::string str)
- void [setArgType](#) (ARGTYPES typ)
- void [setArgValue](#) (std::string str)
- void [setArgValueList](#) (std::vector< std::string > strList)
- void [replaceArgValueList](#) (std::vector< std::string > strList)
- void [setArgDomain](#) (std::string str)
- void [setArgDomain](#) (std::vector< std::string > strList)
- void [setArgName](#) (std::string str)
- void [setArgOpt](#) (std::string str)
- void [setArgMinorOpt](#) (std::string str)
- void [setArgMinorOpt](#) (std::vector< std::string > strList)
- void [setArgDenoteOut](#) (std::string str)
- void [setArgDenoteIn](#) (std::string str)
- void [setArgDescription](#) (std::string description)
- void [isArgMulti](#) ()
- void [notArgMulti](#) ()
- void [isArgConst](#) ()
- void [notArgConst](#) ()
- ARGTYPES [getArgType](#) () const
- std::string [getArgValue](#) (const int order) const
- std::vector< std::string > [getArgValueList](#) () const
- std::string [getArgDomain](#) (const int order) const
- std::vector< std::string > [getArgDomain](#) () const
- std::string [getArgName](#) () const
- std::string [getArgOpt](#) () const
- std::vector< std::string > [getArgMinorOpt](#) () const
- std::string [getArgDenoteOut](#) () const
- std::string [getArgDenoteIn](#) () const
- std::string [getArgDescription](#) () const
- bool [getArgMulti](#) () const
- bool [getArgConst](#) () const

Private Attributes

- ARGTYPES [_argType](#) = ARGTYPES::NONE
- std::vector< std::string > [_argValueList](#)
- std::vector< std::string > [_argDomain](#)
- std::string [_argName](#) = ""
- std::string [_argOpt](#) = ""
- std::vector< std::string > [_argMinorOpt](#)
- std::string [_argDenoteOut](#) = ""
- std::string [_argDenoteIn](#) = ""
- std::string [_description](#) = ""
- bool [_isArgMulti](#) = false
- bool [_isArgConst](#) = false

7.3.1 Detailed Description

Definition at line 45 of file [cppargs.h](#).

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Argument()

```
Argument::Argument (
    std::string str )
```

Definition at line 133 of file [cppargs.cpp](#).

7.3.3 Member Function Documentation

7.3.3.1 getArgConst()

```
bool Argument::getArgConst ( ) const
```

Definition at line 256 of file [cppargs.cpp](#).

7.3.3.2 getArgDenoteIn()

```
std::string Argument::getArgDenoteIn ( ) const
```

Definition at line 244 of file [cppargs.cpp](#).

7.3.3.3 getArgDenoteOut()

```
std::string Argument::getArgDenoteOut ( ) const
```

Definition at line 240 of file [cppargs.cpp](#).

7.3.3.4 getArgDescription()

```
std::string Argument::getArgDescription ( ) const
```

Definition at line 248 of file [cppargs.cpp](#).

7.3.3.5 getArgDomain() [1/2]

```
std::vector< std::string > Argument::getArgDomain ( ) const
```

Definition at line 224 of file [cppargs.cpp](#).

7.3.3.6 getArgDomain() [2/2]

```
std::string Argument::getArgDomain (
    const int order ) const
```

Definition at line 220 of file [cppargs.cpp](#).

7.3.3.7 getArgMinorOpt()

```
std::vector< std::string > Argument::getArgMinorOpt ( ) const
```

Definition at line 236 of file [cppargs.cpp](#).

7.3.3.8 getArgMulti()

```
bool Argument::getArgMulti ( ) const
```

Definition at line 252 of file [cppargs.cpp](#).

7.3.3.9 getArgName()

```
std::string Argument::getArgName ( ) const
```

Definition at line 228 of file [cppargs.cpp](#).

7.3.3.10 getArgOpt()

```
std::string Argument::getArgOpt ( ) const
```

Definition at line 232 of file [cppargs.cpp](#).

7.3.3.11 getArgType()

```
ARGTYPES Argument::getArgType ( ) const
```

Definition at line 208 of file [cppargs.cpp](#).

7.3.3.12 getArgValue()

```
std::string Argument::getArgValue (
    const int order ) const
```

Definition at line 212 of file [cppargs.cpp](#).

7.3.3.13 getArgValueList()

```
std::vector< std::string > Argument::getArgValueList ( ) const
```

Definition at line 216 of file [cppargs.cpp](#).

7.3.3.14 isArgConst()

```
void Argument::isArgConst ( )
```

Definition at line 200 of file [cppargs.cpp](#).

7.3.3.15 isArgMulti()

```
void Argument::isArgMulti ( )
```

Definition at line 192 of file [cppargs.cpp](#).

7.3.3.16 notArgConst()

```
void Argument::notArgConst ( )
```

Definition at line 204 of file [cppargs.cpp](#).

7.3.3.17 notArgMulti()

```
void Argument::notArgMulti ( )
```

Definition at line 196 of file [cppargs.cpp](#).

7.3.3.18 replaceArgValueList()

```
void Argument::replaceArgValueList (
    std::vector< std::string > strList )
```

Definition at line 148 of file [cppargs.cpp](#).

7.3.3.19 setArgDenoteIn()

```
void Argument::setArgDenoteIn (
    std::string str )
```

Definition at line 184 of file [cppargs.cpp](#).

7.3.3.20 setArgDenoteOut()

```
void Argument::setArgDenoteOut (
    std::string str )
```

Definition at line 180 of file [cppargs.cpp](#).

7.3.3.21 setArgDescription()

```
void Argument::setArgDescription (
    std::string description )
```

Definition at line 188 of file [cppargs.cpp](#).

7.3.3.22 setArgDomain() [1/2]

```
void Argument::setArgDomain (
    std::string str )
```

Definition at line 154 of file [cppargs.cpp](#).

7.3.3.23 setArgDomain() [2/2]

```
void Argument::setArgDomain (
    std::vector< std::string > strList )
```

Definition at line 158 of file [cppargs.cpp](#).

7.3.3.24 setArgMinorOpt() [1/2]

```
void Argument::setArgMinorOpt (
    std::string str )
```

Definition at line 171 of file [cppargs.cpp](#).

7.3.3.25 setArgMinorOpt() [2/2]

```
void Argument::setArgMinorOpt (
    std::vector< std::string > strList )
```

Definition at line 175 of file [cppargs.cpp](#).

7.3.3.26 setArgName()

```
void Argument::setArgName (
    std::string str )
```

Definition at line 163 of file [cppargs.cpp](#).

7.3.3.27 setArgOpt()

```
void Argument::setArgOpt (
    std::string str )
```

Definition at line 167 of file [cppargs.cpp](#).

7.3.3.28 setArgType()

```
void Argument::setArgType (
    ARGTYPES typ )
```

Definition at line 135 of file [cppargs.cpp](#).

7.3.3.29 setArgValue()

```
void Argument::setArgValue (
    std::string str )
```

Definition at line 139 of file [cppargs.cpp](#).

7.3.3.30 setArgValueList()

```
void Argument::setArgValueList (
    std::vector< std::string > strList )
```

Definition at line 143 of file [cppargs.cpp](#).

7.3.4 Member Data Documentation

7.3.4.1 _argDenoteIn

```
std::string Argument::_argDenoteIn = "" [private]
```

Definition at line 56 of file [cppargs.h](#).

7.3.4.2 _argDenoteOut

```
std::string Argument::_argDenoteOut = "" [private]
```

Definition at line 55 of file [cppargs.h](#).

7.3.4.3 _argDomain

```
std::vector<std::string> Argument::_argDomain [private]
```

Definition at line 49 of file [cppargs.h](#).

7.3.4.4 _argMinorOpt

```
std::vector<std::string> Argument::_argMinorOpt [private]
```

Definition at line 53 of file [cppargs.h](#).

7.3.4.5 `_argName`

```
std::string Argument::_argName = "" [private]
```

Definition at line 51 of file [cppargs.h](#).

7.3.4.6 `_argOpt`

```
std::string Argument::_argOpt = "" [private]
```

Definition at line 52 of file [cppargs.h](#).

7.3.4.7 `_argType`

```
ARGTYPES Argument::_argType = ARGTYPES::NONE [private]
```

Definition at line 47 of file [cppargs.h](#).

7.3.4.8 `_argValueList`

```
std::vector<std::string> Argument::_argValueList [private]
```

Definition at line 48 of file [cppargs.h](#).

7.3.4.9 `_description`

```
std::string Argument::_description = "" [private]
```

Definition at line 58 of file [cppargs.h](#).

7.3.4.10 `_isArgConst`

```
bool Argument::_isArgConst = false [private]
```

Definition at line 61 of file [cppargs.h](#).

7.3.4.11 `_isArgMulti`

```
bool Argument::_isArgMulti = false [private]
```

Definition at line 60 of file [cppargs.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cppargs.h](#)
- [/home/ychoi/ATOM/pycpp/src/cppargs.cpp](#)

7.4 ArgumentParser Class Reference

```
#include <cppargs.h>
```

Public Member Functions

- [ArgumentParser](#) (int _argc, char **_argv)
- [~ArgumentParser](#) ()
- [ArgumentParser](#) & [setDescription](#) (std::string _description)
- [ArgumentParser](#) & [add_argument](#) (const std::string &opts)
- [ArgumentParser](#) & [add_minor_argument](#) (const std::string &opts)
- [ArgumentParser](#) & [add_domain](#) (const std::vector< std::string > &opts)
- [ArgumentParser](#) & [dest](#) (const std::string &str)
- [ArgumentParser](#) & [metavar](#) (const std::string &str)
- [ArgumentParser](#) & [set_const](#) ()
- [ArgumentParser](#) & [nargs](#) ()
- [ArgumentParser](#) & [type](#) (std::string typeOpt)
- [ArgumentParser](#) & [help](#) (std::string message)
- [ArgumentParser](#) & [set_default](#) (std::string value)
- [ARGTYPES](#) [detType](#) (const std::string &str)
- void [add_finish](#) ()
- void [parse_args](#) ()
- template<typename T >
 T [get_value](#) (const std::string &valueName)
- template<> int [get_value](#) (const std::string &valueName)
- template<> double [get_value](#) (const std::string &valueName)
- template<> bool [get_value](#) (const std::string &valueName)

Private Attributes

- std::vector< std::string > [argv](#)
- std::string [prog](#) = ""
- std::string [description](#) = ""
- std::vector< [Argument](#) > [Pos_args](#)
- std::vector< [Argument](#) > [Opt_args](#)
- [Argument](#) * [args_temp](#)
- std::unordered_map< std::string, std::vector< std::string > > [argv_init](#)
- bool [needHelp](#) = false

7.4.1 Detailed Description

Definition at line 110 of file [cppargs.h](#).

7.4.2 Constructor & Destructor Documentation

7.4.2.1 ArgumentParser()

```
ArgumentParser::ArgumentParser (
    int _argc,
    char ** _argv )
```

Definition at line 261 of file [cppargs.cpp](#).

7.4.2.2 ~ArgumentParser()

```
ArgumentParser::~~ArgumentParser ( )
```

Definition at line 288 of file [cppargs.cpp](#).

7.4.3 Member Function Documentation

7.4.3.1 add_argument()

```
ArgumentParser & ArgumentParser::add_argument (
    const std::string & opts )
```

Definition at line 290 of file [cppargs.cpp](#).

7.4.3.2 add_domain()

```
ArgumentParser & ArgumentParser::add_domain (
    const std::vector< std::string > & opts )
```

Definition at line 310 of file [cppargs.cpp](#).

7.4.3.3 add_finish()

```
void ArgumentParser::add_finish ( )
```

Definition at line 378 of file [cppargs.cpp](#).

7.4.3.4 add_minor_argument()

```
ArgumentParser & ArgumentParser::add_minor_argument (
    const std::string & opts )
```

Definition at line 305 of file [cppargs.cpp](#).

7.4.3.5 dest()

```
ArgumentParser & ArgumentParser::dest (
    const std::string & str )
```

Definition at line 315 of file [cppargs.cpp](#).

7.4.3.6 detType()

```
ARGTYPES ArgumentParser::detType (
    const std::string & str )
```

Definition at line 363 of file [cppargs.cpp](#).

7.4.3.7 `get_value()` [1/4]

```
template<typename T >
T ArgumentParser::get_value (
    const std::string & valueName )
```

Definition at line 465 of file `cppargs.cpp`.

7.4.3.8 `get_value()` [2/4]

```
template<>
int ArgumentParser::get_value (
    const std::string & valueName )
```

Definition at line 469 of file `cppargs.cpp`.

7.4.3.9 `get_value()` [3/4]

```
template<>
double ArgumentParser::get_value (
    const std::string & valueName )
```

Definition at line 501 of file `cppargs.cpp`.

7.4.3.10 `get_value()` [4/4]

```
template<>
bool ArgumentParser::get_value (
    const std::string & valueName )
```

Definition at line 533 of file `cppargs.cpp`.

7.4.3.11 `help()`

```
ArgumentParser & ArgumentParser::help (
    std::string message )
```

Definition at line 353 of file `cppargs.cpp`.

7.4.3.12 `metavar()`

```
ArgumentParser & ArgumentParser::metavar (
    const std::string & str )
```

Definition at line 320 of file `cppargs.cpp`.

7.4.3.13 nargs()

[ArgumentParser](#) & [ArgumentParser::nargs](#) ()

Definition at line 330 of file [cppargs.cpp](#).

7.4.3.14 parse_args()

void [ArgumentParser::parse_args](#) ()

Definition at line 384 of file [cppargs.cpp](#).

7.4.3.15 set_const()

[ArgumentParser](#) & [ArgumentParser::set_const](#) ()

Definition at line 325 of file [cppargs.cpp](#).

7.4.3.16 set_default()

[ArgumentParser](#) & [ArgumentParser::set_default](#) (
std::string value)

Definition at line 358 of file [cppargs.cpp](#).

7.4.3.17 setDescription()

[ArgumentParser](#) & [ArgumentParser::setDescription](#) (
std::string _description) [inline]

Definition at line 124 of file [cppargs.h](#).

7.4.3.18 type()

[ArgumentParser](#) & [ArgumentParser::type](#) (
std::string typeOpt)

Definition at line 335 of file [cppargs.cpp](#).

7.4.4 Member Data Documentation

7.4.4.1 args_temp

[Argument*](#) [ArgumentParser::args_temp](#) [private]

Definition at line 117 of file [cppargs.h](#).

7.4.4.2 argv

```
std::vector<std::string> ArgumentParser::argv [private]
```

Definition at line 112 of file [cppargs.h](#).

7.4.4.3 argv_init

```
std::unordered_map<std::string, std::vector<std::string> > ArgumentParser::argv_init [private]
```

Definition at line 118 of file [cppargs.h](#).

7.4.4.4 description

```
std::string ArgumentParser::description = "" [private]
```

Definition at line 114 of file [cppargs.h](#).

7.4.4.5 needHelp

```
bool ArgumentParser::needHelp = false [private]
```

Definition at line 119 of file [cppargs.h](#).

7.4.4.6 Opt_args

```
std::vector<Argument> ArgumentParser::Opt_args [private]
```

Definition at line 116 of file [cppargs.h](#).

7.4.4.7 Pos_args

```
std::vector<Argument> ArgumentParser::Pos_args [private]
```

Definition at line 115 of file [cppargs.h](#).

7.4.4.8 prog

```
std::string ArgumentParser::prog = "" [private]
```

Definition at line 113 of file [cppargs.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cppargs.h](#)
- [/home/ychoi/ATOM/pycpp/src/cppargs.cpp](#)

7.5 Colour Class Reference

```
#include <Colour.h>
```

Public Member Functions

- [Colour](#) ()
- void [setStandColour](#) ()
- void [setScreenColour](#) ()
- void [setAlpideColour](#) ()
- void [setBoardColour](#) ()
- G4VisAttributes * [getStandColour](#) () const
- G4VisAttributes * [getScreenColour](#) () const
- G4VisAttributes * [getAlpideColour](#) () const
- G4VisAttributes * [getBoardColour](#) () const

Private Attributes

- G4VisAttributes * [standColour](#)
- G4VisAttributes * [screenColour](#)
- G4VisAttributes * [alpideColour](#)
- G4VisAttributes * [boardColour](#)

7.5.1 Detailed Description

Definition at line 10 of file [Colour.h](#).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Colour()

```
Colour::Colour ( )
```

Definition at line 4 of file [Colour.cpp](#).

7.5.3 Member Function Documentation

7.5.3.1 getAlpideColour()

```
G4VisAttributes * Colour::getAlpideColour ( ) const
```

Definition at line 43 of file [Colour.cpp](#).

7.5.3.2 getBoardColour()

```
G4VisAttributes * Colour::getBoardColour ( ) const
```

Definition at line 47 of file [Colour.cpp](#).

7.5.3.3 getScreenColour()

```
G4VisAttributes * Colour::getScreenColour ( ) const
```

Definition at line 39 of file [Colour.cpp](#).

7.5.3.4 getStandColour()

```
G4VisAttributes * Colour::getStandColour ( ) const
```

Definition at line 35 of file [Colour.cpp](#).

7.5.3.5 setAlpideColour()

```
void Colour::setAlpideColour ( )
```

Definition at line 23 of file [Colour.cpp](#).

7.5.3.6 setBoardColour()

```
void Colour::setBoardColour ( )
```

Definition at line 29 of file [Colour.cpp](#).

7.5.3.7 setScreenColour()

```
void Colour::setScreenColour ( )
```

Definition at line 17 of file [Colour.cpp](#).

7.5.3.8 setStandColour()

```
void Colour::setStandColour ( )
```

Definition at line 11 of file [Colour.cpp](#).

7.5.4 Member Data Documentation

7.5.4.1 alpideColour

```
G4VisAttributes* Colour::alpideColour [private]
```

Definition at line 14 of file [Colour.h](#).

7.5.4.2 boardColour

```
G4VisAttributes* Colour::boardColour [private]
```

Definition at line 15 of file [Colour.h](#).

7.5.4.3 screenColour

```
G4VisAttributes* Colour::screenColour [private]
```

Definition at line 13 of file [Colour.h](#).

7.5.4.4 standColour

```
G4VisAttributes* Colour::standColour [private]
```

Definition at line 12 of file [Colour.h](#).

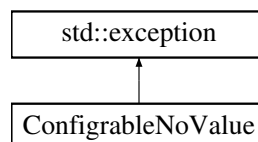
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/Colour.h](#)
- [/home/ychoi/ATOM/geant4/main/src/Colour.cpp](#)

7.6 ConfigurableNoValue Class Reference

```
#include <CppConfigError.h>
```

Inheritance diagram for ConfigurableNoValue:



Public Member Functions

- [ConfigurableNoValue](#) (std::string_view key, std::string_view configName)
- const char * [what](#) () const throw ()

Public Attributes

- std::string [message](#)

7.6.1 Detailed Description

Definition at line 24 of file [CppConfigError.h](#).

7.6.2 Constructor & Destructor Documentation

7.6.2.1 ConfigurableNoValue()

```
ConfigurableNoValue::ConfigurableNoValue (
    std::string_view key,
    std::string_view configName ) [inline]
```

Definition at line 28 of file [CppConfigError.h](#).

7.6.3 Member Function Documentation

7.6.3.1 what()

```
const char * ConfigurableNoValue::what ( ) const throw ( ) [inline]
```

Definition at line 31 of file [CppConfigError.h](#).

7.6.4 Member Data Documentation

7.6.4.1 message

```
std::string ConfigurableNoValue::message
```

Definition at line 26 of file [CppConfigError.h](#).

The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/pycpp/config/inc/CppConfigError.h](#)

7.7 ControlExperimentAnalysis Class Reference

```
#include <ExperimentAnalysis.hpp>
```

Public Member Functions

- [ControlExperimentAnalysis](#) (int argc, char **argv)
- [~ControlExperimentAnalysis](#) ()
- void [setConfig](#) ()
- void [openInputFile](#) ()
- void [setExpDataSet](#) ()
- void [doBasicAnalysis](#) ()
- void [doMasking](#) ()
- void [drawHitmap](#) ()
- void [clusterization](#) ()
- void [drawClustermapAndClustersize](#) ()
- void [doDivideBySize](#) ()
- void [drawClusterShapeInfos](#) ()
- std::vector< int > [getClusterSizeRange](#) (const [CppConfigDictionary](#) privateProperty)

Private Attributes

- [ArgumentParser](#) mParser
- [CppConfigFile](#) * mConfig
- [std::vector< std::string >](#) mTypeNameSet
- [std::unordered_map< std::string, CppConfigDictionary * >](#) mSubConfigSet
- [std::unordered_map< std::string, TExperimentData * >](#) mExpDataSet
- [std::string](#) mInputFilePath
- [TFile](#) * mInputFile = nullptr
- [TAnalyser](#) * mAnalyser
- [std::vector< int >](#) mClusterRange

7.7.1 Detailed Description

Definition at line 16 of file [ExperimentAnalysis.hpp](#).

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ControlExperimentAnalysis()

```
ControlExperimentAnalysis::ControlExperimentAnalysis (
    int argc,
    char ** argv )
```

Definition at line 45 of file [ExperimentAnalysis.hpp](#).

7.7.2.2 ~ControlExperimentAnalysis()

```
ControlExperimentAnalysis::~~ControlExperimentAnalysis ( )
```

Definition at line 51 of file [ExperimentAnalysis.hpp](#).

7.7.3 Member Function Documentation

7.7.3.1 clusterization()

```
void ControlExperimentAnalysis::clusterization ( )
```

Definition at line 129 of file [ExperimentAnalysis.hpp](#).

7.7.3.2 doBasicAnalysis()

```
void ControlExperimentAnalysis::doBasicAnalysis ( )
```

Definition at line 89 of file [ExperimentAnalysis.hpp](#).

7.7.3.3 doDivideBySize()

```
void ControlExperimentAnalysis::doDivideBySize ( )
```

Definition at line 153 of file [ExperimentAnalysis.hpp](#).

7.7.3.4 doMasking()

```
void ControlExperimentAnalysis::doMasking ( )
```

Definition at line 102 of file [ExperimentAnalysis.hpp](#).

7.7.3.5 drawClustermapAndClustersize()

```
void ControlExperimentAnalysis::drawClustermapAndClustersize ( )
```

Definition at line 137 of file [ExperimentAnalysis.hpp](#).

7.7.3.6 drawClusterShapeInfos()

```
void ControlExperimentAnalysis::drawClusterShapeInfos ( )
```

Definition at line 160 of file [ExperimentAnalysis.hpp](#).

7.7.3.7 drawHitmap()

```
void ControlExperimentAnalysis::drawHitmap ( )
```

Definition at line 111 of file [ExperimentAnalysis.hpp](#).

7.7.3.8 getClusterSizeRange()

```
std::vector< int > ControlExperimentAnalysis::getClusterSizeRange (
    const CppConfigDictionary privateProperty )
```

Definition at line 197 of file [ExperimentAnalysis.hpp](#).

7.7.3.9 openInputFile()

```
void ControlExperimentAnalysis::openInputFile ( )
```

Definition at line 79 of file [ExperimentAnalysis.hpp](#).

7.7.3.10 setConfig()

```
void ControlExperimentAnalysis::setConfig ( )
```

Definition at line 71 of file [ExperimentAnalysis.hpp](#).

7.7.3.11 setExpDataSet()

```
void ControlExperimentAnalysis::setExpDataSet ( )
```

Definition at line 83 of file [ExperimentAnalysis.hpp](#).

7.7.4 Member Data Documentation

7.7.4.1 mAnalyser

```
TAnalyser* ControlExperimentAnalysis::mAnalyser [private]
```

Definition at line 25 of file [ExperimentAnalysis.hpp](#).

7.7.4.2 mClusterRange

```
std::vector<int> ControlExperimentAnalysis::mClusterRange [private]
```

Definition at line 26 of file [ExperimentAnalysis.hpp](#).

7.7.4.3 mConfig

```
CppConfigFile* ControlExperimentAnalysis::mConfig [private]
```

Definition at line 18 of file [ExperimentAnalysis.hpp](#).

7.7.4.4 mExpDataSet

```
std::unordered_map<std::string, TExperimentData*> ControlExperimentAnalysis::mExpDataSet [private]
```

Definition at line 21 of file [ExperimentAnalysis.hpp](#).

7.7.4.5 mInputFile

```
TFile* ControlExperimentAnalysis::mInputFile = nullptr [private]
```

Definition at line 24 of file [ExperimentAnalysis.hpp](#).

7.7.4.6 mInputFilePath

```
std::string ControlExperimentAnalysis::mInputFilePath [private]
```

Definition at line 23 of file [ExperimentAnalysis.hpp](#).

7.7.4.7 mParser

```
ArgumentParser ControlExperimentAnalysis::mParser [private]
```

Definition at line 17 of file [ExperimentAnalysis.hpp](#).

7.7.4.8 mSubConfigSet

```
std::unordered_map<std::string, CppConfigDictionary*> ControlExperimentAnalysis::mSubConfigSet  
[private]
```

Definition at line 20 of file [ExperimentAnalysis.hpp](#).

7.7.4.9 mTypeNameSet

```
std::vector<std::string> ControlExperimentAnalysis::mTypeNameSet [private]
```

Definition at line 19 of file [ExperimentAnalysis.hpp](#).

The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/exe/ExperimentAnalysis.hpp](#)

7.8 ControlExperimentComparison Class Reference

Public Member Functions

- [ControlExperimentComparison](#) (int argc, char **argv)
- void [setConfig](#) ()
- void [initComparison](#) ()

Private Attributes

- [ArgumentParser](#) mParser
- [CppConfigFile](#) * mConfig
- std::vector< std::string > mFileSet
- std::vector< std::string > mTypeNameSet
- [TGraphCompare](#) * mCompare

7.8.1 Detailed Description

Definition at line 11 of file [CompareExperimentData.cpp](#).

7.8.2 Constructor & Destructor Documentation

7.8.2.1 ControlExperimentComparison()

```
ControlExperimentComparison::ControlExperimentComparison (
    int argc,
    char ** argv )
```

Definition at line 24 of file [CompareExperimentData.cpp](#).

7.8.3 Member Function Documentation

7.8.3.1 initComparison()

```
void ControlExperimentComparison::initComparison ( )
```

Definition at line 37 of file [CompareExperimentData.cpp](#).

7.8.3.2 setConfig()

```
void ControlExperimentComparison::setConfig ( )
```

Definition at line 30 of file [CompareExperimentData.cpp](#).

7.8.4 Member Data Documentation

7.8.4.1 mCompare

```
TGraphCompare* ControlExperimentComparison::mCompare [private]
```

Definition at line 17 of file [CompareExperimentData.cpp](#).

7.8.4.2 mConfig

```
CppConfigFile* ControlExperimentComparison::mConfig [private]
```

Definition at line 14 of file [CompareExperimentData.cpp](#).

7.8.4.3 mFileSet

```
std::vector<std::string> ControlExperimentComparison::mFileSet [private]
```

Definition at line 15 of file [CompareExperimentData.cpp](#).

7.8.4.4 mParser

[ArgumentParser](#) ControlExperimentComparison::mParser [private]

Definition at line 13 of file [CompareExperimentData.cpp](#).

7.8.4.5 mTypeNameSet

std::vector<std::string> ControlExperimentComparison::mTypeNameSet [private]

Definition at line 16 of file [CompareExperimentData.cpp](#).

The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/exe/CompareExperimentData.cpp](#)

7.9 CppConfigDictionary Class Reference

```
#include <CppConfigDictionary.h>
```

Public Member Functions

- [CppConfigDictionary](#) ()
- [CppConfigDictionary](#) (const [CppConfigDictionary](#) ©)
- [CppConfigDictionary](#) & operator= (const [CppConfigDictionary](#) ©)
- [CppConfigDictionary](#) ([CppConfigDictionary](#) &&move)
- [CppConfigDictionary](#) & operator= ([CppConfigDictionary](#) &&move)
- [CppConfigDictionary](#) (std::string_view configName)
- void [addDictionary](#) (std::string_view key, std::string_view value)
- void [addSubConfigDictionary](#) (const [CppConfigDictionary](#) &subConfigDictionary)
- const bool [hasKey](#) (std::string_view key) const
- const std::string & [find](#) (const std::string &key) const
- const [CppConfigDictionary](#) & [getSubConfig](#) (std::string_view key) const
- const std::vector< [CppConfigDictionary](#) > [getSubConfigSet](#) () const
- const std::vector< std::string > [getValueList](#) () const
- const std::vector< std::string > [getKeyList](#) () const
- std::unordered_map< std::string, std::string > [getDictionary](#) ()
- const std::unordered_map< std::string, [CppConfigDictionary](#) > [getSubConfigSetWithName](#) () const
- std::string_view [getConfigName](#) () const
- [CppConfigDictionary](#) & operator+ (const [CppConfigDictionary](#) ©)
- [CppConfigDictionary](#) & operator+= (const [CppConfigDictionary](#) ©)

Private Attributes

- std::string [mConfigName](#)
- std::unordered_map< std::string, std::string > [mDictionary](#)
- std::vector< [CppConfigDictionary](#) > [mSubConfigDictionary](#)

Friends

- `std::ostream & operator<< (std::ostream &os, const CppConfigDictionary ©)`

7.9.1 Detailed Description

Definition at line 12 of file [CppConfigDictionary.h](#).

7.9.2 Constructor & Destructor Documentation

7.9.2.1 CppConfigDictionary() [1/4]

```
CppConfigDictionary::CppConfigDictionary ( ) [default]
```

7.9.2.2 CppConfigDictionary() [2/4]

```
CppConfigDictionary::CppConfigDictionary (
    const CppConfigDictionary & copy )
```

Definition at line 7 of file [CppConfigDictionary.cpp](#).

7.9.2.3 CppConfigDictionary() [3/4]

```
CppConfigDictionary::CppConfigDictionary (
    CppConfigDictionary && move )
```

Definition at line 17 of file [CppConfigDictionary.cpp](#).

7.9.2.4 CppConfigDictionary() [4/4]

```
CppConfigDictionary::CppConfigDictionary (
    std::string_view configName )
```

Definition at line 5 of file [CppConfigDictionary.cpp](#).

7.9.3 Member Function Documentation

7.9.3.1 addDictionary()

```
void CppConfigDictionary::addDictionary (
    std::string_view key,
    std::string_view value )
```

Definition at line 32 of file [CppConfigDictionary.cpp](#).

7.9.3.2 addSubConfigDictionary()

```
void CppConfigDictionary::addSubConfigDictionary (
    const CppConfigDictionary & subConfigDictionary )
```

Definition at line 36 of file [CppConfigDictionary.cpp](#).

7.9.3.3 find()

```
const std::string & CppConfigDictionary::find (
    const std::string & key ) const
```

Definition at line 53 of file [CppConfigDictionary.cpp](#).

7.9.3.4 getConfigName()

```
std::string_view CppConfigDictionary::getConfigName ( ) const
```

Definition at line 69 of file [CppConfigDictionary.cpp](#).

7.9.3.5 getDictionary()

```
std::unordered_map< std::string, std::string > CppConfigDictionary::getDictionary ( )
```

Definition at line 106 of file [CppConfigDictionary.cpp](#).

7.9.3.6 getKeyList()

```
const std::vector< std::string > CppConfigDictionary::getKeyList ( ) const
```

Definition at line 91 of file [CppConfigDictionary.cpp](#).

7.9.3.7 getSubConfig()

```
const CppConfigDictionary & CppConfigDictionary::getSubConfig (
    std::string_view key ) const
```

Definition at line 110 of file [CppConfigDictionary.cpp](#).

7.9.3.8 getSubConfigSet()

```
const std::vector< CppConfigDictionary > CppConfigDictionary::getSubConfigSet ( ) const
```

Definition at line 157 of file [CppConfigDictionary.cpp](#).

7.9.3.9 getSubConfigSetWithName()

```
const std::unordered_map< std::string, CppConfigDictionary > CppConfigDictionary::getSubConfigSetWithName ( ) const
```

Definition at line 161 of file [CppConfigDictionary.cpp](#).

7.9.3.10 getValueList()

```
const std::vector< std::string > CppConfigDictionary::getValueList ( ) const
```

Definition at line 125 of file [CppConfigDictionary.cpp](#).

7.9.3.11 hasKey()

```
const bool CppConfigDictionary::hasKey (
    std::string_view key ) const
```

Definition at line 40 of file [CppConfigDictionary.cpp](#).

7.9.3.12 operator+()

```
CppConfigDictionary & CppConfigDictionary::operator+ (
    const CppConfigDictionary & copy )
```

Definition at line 135 of file [CppConfigDictionary.cpp](#).

7.9.3.13 operator+=()

```
CppConfigDictionary & CppConfigDictionary::operator+= (
    const CppConfigDictionary & copy )
```

Definition at line 146 of file [CppConfigDictionary.cpp](#).

7.9.3.14 operator=() [1/2]

```
CppConfigDictionary & CppConfigDictionary::operator= (
    const CppConfigDictionary & copy )
```

Definition at line 10 of file [CppConfigDictionary.cpp](#).

7.9.3.15 operator=() [2/2]

```
CppConfigDictionary & CppConfigDictionary::operator= (
    CppConfigDictionary && move )
```

Definition at line 20 of file [CppConfigDictionary.cpp](#).

7.9.4 Friends And Related Symbol Documentation

7.9.4.1 operator<<

```
std::ostream & operator<< (  
    std::ostream & os,  
    const CppConfigDictionary & copy ) [friend]
```

Definition at line 73 of file [CppConfigDictionary.cpp](#).

7.9.5 Member Data Documentation

7.9.5.1 mConfigName

```
std::string CppConfigDictionary::mConfigName [private]
```

Definition at line 13 of file [CppConfigDictionary.h](#).

7.9.5.2 mDictionary

```
std::unordered_map<std::string, std::string> CppConfigDictionary::mDictionary [private]
```

Definition at line 14 of file [CppConfigDictionary.h](#).

7.9.5.3 mSubConfigDictionary

```
std::vector<CppConfigDictionary> CppConfigDictionary::mSubConfigDictionary [private]
```

Definition at line 15 of file [CppConfigDictionary.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/config/inc/CppConfigDictionary.h](#)
- [/home/ychoi/ATOM/pycpp/config/src/CppConfigDictionary.cpp](#)

7.10 CppConfigFile Class Reference

```
#include <CppConfigFile.h>
```

Public Member Functions

- [CppConfigFile](#) ()
Construct a new [CppConfigFile::CppConfigFile](#) object.
- [CppConfigFile](#) (std::string_view configFile)
Construct a new [CppConfigFile::CppConfigFile](#) object.
- [CppConfigFile](#) (const [CppConfigFile](#) ©)
- [CppConfigFile](#) & operator= (const [CppConfigFile](#) ©)
- [CppConfigFile](#) ([CppConfigFile](#) &&move)
- [CppConfigFile](#) & operator= ([CppConfigFile](#) &&move)
- [~CppConfigFile](#) ()
- void [addConfig](#) (std::string_view configFile)
Add config dictionaries from config file.
- [CppConfigDictionary](#) [getConfigFromArray](#) (std::string_view key, const std::vector< std::string > &valueArray)
- void [addConfig](#) (std::string_view configTitle, const std::vector< std::string > &configArray)
- const std::vector< std::string > [getConfigurableNameList](#) () const
- const [CppConfigDictionary](#) [getConfig](#) (std::string_view configTitle) const
- const bool [hasConfig](#) (std::string_view configTitle) const

Private Attributes

- std::vector< [CppConfigDictionary](#) > mConfigs

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [CppConfigFile](#) ©)

7.10.1 Detailed Description

Definition at line 17 of file [CppConfigFile.h](#).

7.10.2 Constructor & Destructor Documentation

7.10.2.1 CppConfigFile() [1/4]

```
CppConfigFile::CppConfigFile ( ) [default]
```

Construct a new [CppConfigFile::CppConfigFile](#) object.

7.10.2.2 CppConfigFile() [2/4]

```
CppConfigFile::CppConfigFile (
    std::string_view configFile )
```

Construct a new [CppConfigFile::CppConfigFile](#) object.

Taking config file path and store config dictionaries.

Parameters

<i>configFile</i>	
-------------------	--

Definition at line 15 of file [CppConfigFile.cpp](#).

7.10.2.3 CppConfigFile() [3/4]

```
CppConfigFile::CppConfigFile (
    const CppConfigFile & copy )
```

Definition at line 26 of file [CppConfigFile.cpp](#).

7.10.2.4 CppConfigFile() [4/4]

```
CppConfigFile::CppConfigFile (
    CppConfigFile && move )
```

Definition at line 37 of file [CppConfigFile.cpp](#).

7.10.2.5 ~CppConfigFile()

```
CppConfigFile::~~CppConfigFile ( )
```

Definition at line 240 of file [CppConfigFile.cpp](#).

7.10.3 Member Function Documentation

7.10.3.1 addConfig() [1/2]

```
void CppConfigFile::addConfig (
    std::string_view configFile )
```

Add config dictionaries from config file.

Parameters

<i>configFile</i>	
-------------------	--

Definition at line 54 of file [CppConfigFile.cpp](#).

7.10.3.2 addConfig() [2/2]

```
void CppConfigFile::addConfig (
    std::string_view configTitle,
    const std::vector< std::string > & configArray )
```

Definition at line 123 of file [CppConfigFile.cpp](#).

7.10.3.3 getConfig()

```
const CppConfigDictionary CppConfigFile::getConfig (
    std::string_view configTitle ) const
```

Definition at line 211 of file [CppConfigFile.cpp](#).

7.10.3.4 getConfigFromArray()

```
CppConfigDictionary CppConfigFile::getConfigFromArray (
    std::string_view key,
    const std::vector< std::string > & valueArray )
```

Definition at line 128 of file [CppConfigFile.cpp](#).

7.10.3.5 getConfigurableNameList()

```
const std::vector< std::string > CppConfigFile::getConfigurableNameList ( ) const
```

Definition at line 202 of file [CppConfigFile.cpp](#).

7.10.3.6 hasConfig()

```
const bool CppConfigFile::hasConfig (
    std::string_view configTitle ) const
```

Definition at line 222 of file [CppConfigFile.cpp](#).

7.10.3.7 operator=() [1/2]

```
CppConfigFile & CppConfigFile::operator= (
    const CppConfigFile & copy )
```

Definition at line 31 of file [CppConfigFile.cpp](#).

7.10.3.8 operator=() [2/2]

```
CppConfigFile & CppConfigFile::operator= (
    CppConfigFile && move )
```

Definition at line 42 of file [CppConfigFile.cpp](#).

7.10.4 Friends And Related Symbol Documentation

7.10.4.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const CppConfigFile & copy ) [friend]
```

Definition at line 232 of file [CppConfigFile.cpp](#).

7.10.5 Member Data Documentation

7.10.5.1 mConfigs

```
std::vector<CppConfigDictionary> CppConfigFile::mConfigs [private]
```

Definition at line 19 of file [CppConfigFile.h](#).

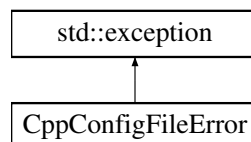
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/config/inc/CppConfigFile.h](#)
- [/home/ychoi/ATOM/pycpp/config/src/CppConfigFile.cpp](#)

7.11 CppConfigFileError Class Reference

```
#include <CppConfigError.h>
```

Inheritance diagram for CppConfigFileError:



Public Member Functions

- [CppConfigFileError](#) (std::string_view errorType, std::string_view parameter)
- const char * [what](#) () const throw ()

Public Attributes

- std::string [mMessage](#)

7.11.1 Detailed Description

Definition at line 7 of file [CppConfigError.h](#).

7.11.2 Constructor & Destructor Documentation

7.11.2.1 CppConfigFileError()

```
CppConfigFileError::CppConfigFileError (
    std::string_view errorType,
    std::string_view parameter ) [inline]
```

Definition at line 11 of file [CppConfigError.h](#).

7.11.3 Member Function Documentation

7.11.3.1 what()

```
const char * CppConfigFileError::what ( ) const throw ( ) [inline]
```

Definition at line 18 of file [CppConfigError.h](#).

7.11.4 Member Data Documentation

7.11.4.1 mMessage

```
std::string CppConfigFileError::mMessage
```

Definition at line 9 of file [CppConfigError.h](#).

The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/pycpp/config/inc/CppConfigError.h](#)

7.12 DACtoADC Class Reference

Public Member Functions

- [DACtoADC](#) (std::string kind)
- void [setDAC](#) (int dac)
- void [setADC](#) (int adc)
- void [setDAC](#) (std::vector< int > dac)
- void [setADC](#) (std::vector< int > adc)
- std::string [getKind](#) ()
- std::vector< Int_t > [getDAC](#) ()
- std::vector< Int_t > [getADC](#) ()

Private Attributes

- std::string [kind_](#)
- std::vector< Int_t > [dac_](#)
- std::vector< Int_t > [adc_](#)

7.12.1 Detailed Description

Definition at line 15 of file [alpide_dac.cpp](#).

7.12.2 Constructor & Destructor Documentation

7.12.2.1 DACtoADC()

```
DACtoADC::DACtoADC (
    std::string kind ) [inline]
```

Definition at line 21 of file [alpide_dac.cpp](#).

7.12.3 Member Function Documentation

7.12.3.1 getADC()

```
std::vector< Int_t > DACtoADC::getADC ( ) [inline]
```

Definition at line 42 of file [alpide_dac.cpp](#).

7.12.3.2 getDAC()

```
std::vector< Int_t > DACtoADC::getDAC ( ) [inline]
```

Definition at line 39 of file [alpide_dac.cpp](#).

7.12.3.3 getKind()

```
std::string DACtoADC::getKind ( ) [inline]
```

Definition at line 36 of file [alpide_dac.cpp](#).

7.12.3.4 setADC() [1/2]

```
void DACtoADC::setADC (
    int adc ) [inline]
```

Definition at line 25 of file [alpide_dac.cpp](#).

7.12.3.5 setADC() [2/2]

```
void DACtoADC::setADC (
    std::vector< int > adc ) [inline]
```

Definition at line 32 of file [alpide_dac.cpp](#).

7.12.3.6 setDAC() [1/2]

```
void DACToADC::setDAC (
    int dac ) [inline]
```

Definition at line 22 of file [alpide_dac.cpp](#).

7.12.3.7 setDAC() [2/2]

```
void DACToADC::setDAC (
    std::vector< int > dac ) [inline]
```

Definition at line 28 of file [alpide_dac.cpp](#).

7.12.4 Member Data Documentation**7.12.4.1 adc_**

```
std::vector<Int_t> DACToADC::adc_ [private]
```

Definition at line 19 of file [alpide_dac.cpp](#).

7.12.4.2 dac_

```
std::vector<Int_t> DACToADC::dac_ [private]
```

Definition at line 18 of file [alpide_dac.cpp](#).

7.12.4.3 kind_

```
std::string DACToADC::kind_ [private]
```

Definition at line 17 of file [alpide_dac.cpp](#).

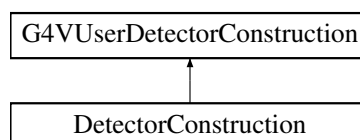
The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/exe/alpide_dac.cpp](#)

7.13 DetectorConstruction Class Reference

```
#include <DetectorConstruction.h>
```

Inheritance diagram for DetectorConstruction:



Public Member Functions

- [DetectorConstruction](#) ()
- virtual [~DetectorConstruction](#) ()
- virtual G4VPhysicalVolume * [Construct](#) ()
- G4VPhysicalVolume * [Construct](#) (G4String [standType](#), G4double distance)
- void [SetWorld](#) (G4double air_pressure)
- void [SetStand](#) (G4String [standType](#), G4double hallDiameter=0.)
- void [SetShield](#) (G4double shieldWidth)
- void [SetALPIDE](#) (G4String alptideType)
- void [SetCarrierBoard](#) ()
- G4LogicalVolume * [GetScoringStand](#) () const
- G4LogicalVolume * [GetScoringShield](#) () const
- G4LogicalVolume * [GetScoringALPIDEcircuit](#) () const
- G4LogicalVolume * [GetScoringALPIDEepitaxial](#) () const
- G4LogicalVolume * [GetScoringCarrierBoard](#) () const
- void [SetEnergy](#) (G4double energy)
- void [SetSourceType](#) (G4String type)
- void [SetStandType](#) (G4String type)
- void [SetDistance](#) (G4double distance)
- void [SetVacuum](#) (G4double vacuum)
- void [SetHallWidth](#) (G4double diameter)

Private Member Functions

- void [DefineCommands](#) ()

Private Attributes

- [Material](#) * [material](#) = nullptr
- [Colour](#) * [colour](#) = nullptr
- [Solid](#) * [solids](#) = nullptr
- G4VPhysicalVolume * [WorldPhysical](#) = nullptr
- G4VPhysicalVolume * [StandPhysical](#) = nullptr
- G4VPhysicalVolume * [ShieldPhysical](#) = nullptr
- G4VPhysicalVolume * [CarrierBoardPhysical](#) = nullptr
- G4LogicalVolume * [WorldLogical](#) = nullptr
- G4LogicalVolume * [StandLogical](#) = nullptr
- G4LogicalVolume * [ShieldLogical](#) = nullptr
- G4LogicalVolume * [ALPIDEcircuitLogical](#) = nullptr
- G4LogicalVolume * [ALPIDEepitaxialLogical](#) = nullptr
- G4AssemblyVolume * [ALPIDEassembly](#) = nullptr
- G4LogicalVolume * [CarrierBoardLogical](#) = nullptr
- [SourceType](#) [sourceType](#)
- [StandType](#) [standType](#) = [StandType::none](#)
- G4double [sEnergy](#) = 5.4
- G4double [sDistance](#) = 10.
- G4double [cDiameter](#) = 1.

7.13.1 Detailed Description

Definition at line 40 of file [DetectorConstruction.h](#).

7.13.2 Constructor & Destructor Documentation

7.13.2.1 DetectorConstruction()

```
DetectorConstruction::DetectorConstruction ( )
```

Definition at line 4 of file [DetectorConstruction.cpp](#).

7.13.2.2 ~DetectorConstruction()

```
DetectorConstruction::~~DetectorConstruction ( ) [virtual]
```

Definition at line 11 of file [DetectorConstruction.cpp](#).

7.13.3 Member Function Documentation

7.13.3.1 Construct() [1/2]

```
G4VPhysicalVolume * DetectorConstruction::Construct ( ) [virtual]
```

Definition at line 18 of file [DetectorConstruction.cpp](#).

7.13.3.2 Construct() [2/2]

```
G4VPhysicalVolume * DetectorConstruction::Construct (
    G4String standType,
    G4double distance )
```

Definition at line 22 of file [DetectorConstruction.cpp](#).

7.13.3.3 DefineCommands()

```
void DetectorConstruction::DefineCommands ( ) [private]
```

7.13.3.4 GetScoringALPIDEcircuit()

```
G4LogicalVolume * DetectorConstruction::GetScoringALPIDEcircuit ( ) const
```

Definition at line 129 of file [DetectorConstruction.cpp](#).

7.13.3.5 GetScoringALPIDEepitaxial()

```
G4LogicalVolume * DetectorConstruction::GetScoringALPIDEepitaxial ( ) const
```

Definition at line 133 of file [DetectorConstruction.cpp](#).

7.13.3.6 GetScoringCarrierBoard()

```
G4LogicalVolume * DetectorConstruction::GetScoringCarrierBoard ( ) const
```

Definition at line 137 of file [DetectorConstruction.cpp](#).

7.13.3.7 GetScoringShield()

```
G4LogicalVolume * DetectorConstruction::GetScoringShield ( ) const
```

Definition at line 125 of file [DetectorConstruction.cpp](#).

7.13.3.8 GetScoringStand()

```
G4LogicalVolume * DetectorConstruction::GetScoringStand ( ) const
```

Definition at line 121 of file [DetectorConstruction.cpp](#).

7.13.3.9 SetALPIDE()

```
void DetectorConstruction::SetALPIDE (
    G4String alpideType )
```

Definition at line 89 of file [DetectorConstruction.cpp](#).

7.13.3.10 SetCarrierBoard()

```
void DetectorConstruction::SetCarrierBoard ( )
```

Definition at line 115 of file [DetectorConstruction.cpp](#).

7.13.3.11 SetDistance()

```
void DetectorConstruction::SetDistance (
    G4double distance )
```

7.13.3.12 SetEnergy()

```
void DetectorConstruction::SetEnergy (
    G4double energy )
```

7.13.3.13 SetHallWidth()

```
void DetectorConstruction::SetHallWidth (
    G4double diameter )
```

7.13.3.14 SetShield()

```
void DetectorConstruction::SetShield (
    G4double shieldWidth )
```

Definition at line 82 of file [DetectorConstruction.cpp](#).

7.13.3.15 SetSourceType()

```
void DetectorConstruction::SetSourceType (
    G4String type )
```

7.13.3.16 SetStand()

```
void DetectorConstruction::SetStand (
    G4String standType,
    G4double hallDiameter = 0. )
```

Definition at line 63 of file [DetectorConstruction.cpp](#).

7.13.3.17 SetStandType()

```
void DetectorConstruction::SetStandType (
    G4String type )
```

7.13.3.18 SetVacuum()

```
void DetectorConstruction::SetVacuum (
    G4double vacuum )
```

7.13.3.19 SetWorld()

```
void DetectorConstruction::SetWorld (
    G4double air_pressure )
```

Definition at line 57 of file [DetectorConstruction.cpp](#).

7.13.4 Member Data Documentation

7.13.4.1 ALPIDEAssembly

```
G4AssemblyVolume* DetectorConstruction::ALPIDEAssembly = nullptr [private]
```

Definition at line 56 of file [DetectorConstruction.h](#).

7.13.4.2 ALPIDECircuitLogical

```
G4LogicalVolume* DetectorConstruction::ALPIDECircuitLogical = nullptr [private]
```

Definition at line 54 of file [DetectorConstruction.h](#).

7.13.4.3 ALPIDEEpitaxialLogical

```
G4LogicalVolume* DetectorConstruction::ALPIDEEpitaxialLogical = nullptr [private]
```

Definition at line 55 of file [DetectorConstruction.h](#).

7.13.4.4 CarrierBoardLogical

```
G4LogicalVolume* DetectorConstruction::CarrierBoardLogical = nullptr [private]
```

Definition at line 57 of file [DetectorConstruction.h](#).

7.13.4.5 CarrierBoardPhysical

```
G4VPhysicalVolume* DetectorConstruction::CarrierBoardPhysical = nullptr [private]
```

Definition at line 49 of file [DetectorConstruction.h](#).

7.13.4.6 cDiameter

```
G4double DetectorConstruction::cDiameter = 1. [private]
```

Definition at line 63 of file [DetectorConstruction.h](#).

7.13.4.7 colour

```
Colour* DetectorConstruction::colour = nullptr [private]
```

Definition at line 43 of file [DetectorConstruction.h](#).

7.13.4.8 material

```
Material* DetectorConstruction::material = nullptr [private]
```

Definition at line 42 of file [DetectorConstruction.h](#).

7.13.4.9 sDistance

```
G4double DetectorConstruction::sDistance = 10. [private]
```

Definition at line 62 of file [DetectorConstruction.h](#).

7.13.4.10 sEnergy

```
G4double DetectorConstruction::sEnergy = 5.4 [private]
```

Definition at line 61 of file [DetectorConstruction.h](#).

7.13.4.11 ShieldLogical

```
G4LogicalVolume* DetectorConstruction::ShieldLogical = nullptr [private]
```

Definition at line 53 of file [DetectorConstruction.h](#).

7.13.4.12 ShieldPhysical

```
G4VPhysicalVolume* DetectorConstruction::ShieldPhysical = nullptr [private]
```

Definition at line 48 of file [DetectorConstruction.h](#).

7.13.4.13 solids

```
Solid* DetectorConstruction::solids = nullptr [private]
```

Definition at line 44 of file [DetectorConstruction.h](#).

7.13.4.14 sourceType

```
SourceType DetectorConstruction::sourceType [private]
```

Definition at line 59 of file [DetectorConstruction.h](#).

7.13.4.15 StandLogical

```
G4LogicalVolume* DetectorConstruction::StandLogical = nullptr [private]
```

Definition at line 52 of file [DetectorConstruction.h](#).

7.13.4.16 StandPhysical

```
G4VPhysicalVolume* DetectorConstruction::StandPhysical = nullptr [private]
```

Definition at line 47 of file [DetectorConstruction.h](#).

7.13.4.17 standType

```
StandType DetectorConstruction::standType = StandType::none [private]
```

Definition at line 60 of file [DetectorConstruction.h](#).

7.13.4.18 WorldLogical

```
G4LogicalVolume* DetectorConstruction::WorldLogical = nullptr [private]
```

Definition at line 51 of file [DetectorConstruction.h](#).

7.13.4.19 WorldPhysical

```
G4VPhysicalVolume* DetectorConstruction::WorldPhysical = nullptr [private]
```

Definition at line 46 of file [DetectorConstruction.h](#).

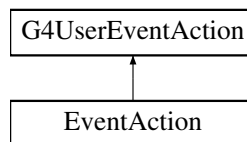
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/DetectorConstruction.h](#)
- [/home/ychoi/ATOM/geant4/main/src/DetectorConstruction.cpp](#)

7.14 EventAction Class Reference

```
#include <EventAction.h>
```

Inheritance diagram for EventAction:



Public Member Functions

- [EventAction](#) ([RunAction](#) *runAction)
- virtual [~EventAction](#) ()
- virtual void [BeginOfEventAction](#) (const G4Event *event)
- virtual void [EndOfEventAction](#) (const G4Event *event)

Private Attributes

- [G4int](#) iStartTrack

7.14.1 Detailed Description

Definition at line 18 of file [EventAction.h](#).

7.14.2 Constructor & Destructor Documentation

7.14.2.1 EventAction()

```
EventAction::EventAction (
    RunAction * runAction )
```

Definition at line 4 of file [EventAction.cpp](#).

7.14.2.2 ~EventAction()

```
EventAction::~EventAction ( ) [virtual]
```

Definition at line 8 of file [EventAction.cpp](#).

7.14.3 Member Function Documentation

7.14.3.1 BeginOfEventAction()

```
void EventAction::BeginOfEventAction (
    const G4Event * event ) [virtual]
```

Definition at line 12 of file [EventAction.cpp](#).

7.14.3.2 EndOfEventAction()

```
void EventAction::EndOfEventAction (
    const G4Event * event ) [virtual]
```

Definition at line 24 of file [EventAction.cpp](#).

7.14.4 Member Data Documentation

7.14.4.1 iStartTrack

```
G4int EventAction::iStartTrack [private]
```

Definition at line 20 of file [EventAction.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/EventAction.h](#)
- [/home/ychoi/ATOM/geant4/main/src/EventAction.cpp](#)

7.15 EventTuple Struct Reference

```
#include <AnalysisManager.h>
```

Public Attributes

- int [runID](#)
- int [eventGlobalID](#)
- int [eventLocalID](#)
- int [nTracks](#)

7.15.1 Detailed Description

Definition at line 29 of file [AnalysisManager.h](#).

7.15.2 Member Data Documentation

7.15.2.1 eventGlobalID

```
int EventTuple::eventGlobalID
```

Definition at line 31 of file [AnalysisManager.h](#).

7.15.2.2 eventLocalID

```
int EventTuple::eventLocalID
```

Definition at line 32 of file [AnalysisManager.h](#).

7.15.2.3 nTracks

```
int EventTuple::nTracks
```

Definition at line 33 of file [AnalysisManager.h](#).

7.15.2.4 runID

```
int EventTuple::runID
```

Definition at line 30 of file [AnalysisManager.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h](#)

7.16 HelpMessage Class Reference

```
#include <cppargs.h>
```

Public Member Functions

- [HelpMessage](#) (std::string prog, std::string description)
- void [print](#) (std::vector< [Argument](#) > &Pos_args, std::vector< [Argument](#) > &Opt_args)

Private Attributes

- std::string [usage](#)

7.16.1 Detailed Description

Definition at line 29 of file [cppargs.h](#).

7.16.2 Constructor & Destructor Documentation

7.16.2.1 HelpMessage()

```
HelpMessage::HelpMessage (
    std::string prog,
    std::string description )
```

Definition at line 3 of file [cppargs.cpp](#).

7.16.3 Member Function Documentation

7.16.3.1 print()

```
void HelpMessage::print (
    std::vector< Argument > & Pos_args,
    std::vector< Argument > & Opt_args )
```

Definition at line 7 of file [cppargs.cpp](#).

7.16.4 Member Data Documentation

7.16.4.1 usage

```
std::string HelpMessage::usage [private]
```

Definition at line 31 of file [cppargs.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cppargs.h](#)
- [/home/ychoi/ATOM/pycpp/src/cppargs.cpp](#)

7.17 Material Class Reference

```
#include <Material.h>
```

Public Member Functions

- [Material](#) ()
- G4Material * [getWorldMaterial](#) () const
- G4Material * [getStandMaterial](#) () const
- G4Material * [getScreenMaterial](#) () const
- G4Material * [getAlpideMaterial](#) () const
- G4Material * [getBoardMaterial](#) () const
- void [setElement](#) ()
- void [setWorldMaterial](#) (const double density=1.)
- void [setStandMaterial](#) ()
- void [setScreenMaterial](#) ()
- void [setAlpideMaterial](#) ()
- void [setBoardMaterial](#) ()

Private Attributes

- G4Material * [worldMaterial](#)
- G4Material * [standMaterial](#)
- G4Material * [screenMaterial](#)
- G4Material * [alpideMaterial](#)
- G4Material * [boardMaterial](#)
- G4Material * [epoxyResin](#)
- G4Material * [fibrousGlass](#)
- G4Element * [elSi](#)
- G4Element * [elN](#)
- G4Element * [elO](#)
- G4Element * [elAl](#)
- G4Element * [elFe](#)
- G4Element * [elCa](#)
- G4Element * [elMg](#)
- G4Element * [elNa](#)
- G4Element * [elTi](#)
- G4Element * [elC](#)
- G4Element * [elH](#)
- G4Element * [elBr](#)

7.17.1 Detailed Description

Definition at line 14 of file [Material.h](#).

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Material()

```
Material::Material ( )
```

Definition at line 4 of file [Material.cpp](#).

7.17.3 Member Function Documentation

7.17.3.1 `getAlpideMaterial()`

```
G4Material * Material::getAlpideMaterial ( ) const
```

Definition at line 112 of file [Material.cpp](#).

7.17.3.2 `getBoardMaterial()`

```
G4Material * Material::getBoardMaterial ( ) const
```

Definition at line 116 of file [Material.cpp](#).

7.17.3.3 `getScreenMaterial()`

```
G4Material * Material::getScreenMaterial ( ) const
```

Definition at line 108 of file [Material.cpp](#).

7.17.3.4 `getStandMaterial()`

```
G4Material * Material::getStandMaterial ( ) const
```

Definition at line 104 of file [Material.cpp](#).

7.17.3.5 `getWorldMaterial()`

```
G4Material * Material::getWorldMaterial ( ) const
```

Definition at line 100 of file [Material.cpp](#).

7.17.3.6 `setAlpideMaterial()`

```
void Material::setAlpideMaterial ( )
```

Definition at line 46 of file [Material.cpp](#).

7.17.3.7 `setBoardMaterial()`

```
void Material::setBoardMaterial ( )
```

Definition at line 50 of file [Material.cpp](#).

7.17.3.8 setElement()

```
void Material::setElement ( )
```

Definition at line 14 of file [Material.cpp](#).

7.17.3.9 setScreenMaterial()

```
void Material::setScreenMaterial ( )
```

Definition at line 42 of file [Material.cpp](#).

7.17.3.10 setStandMaterial()

```
void Material::setStandMaterial ( )
```

Definition at line 35 of file [Material.cpp](#).

7.17.3.11 setWorldMaterial()

```
void Material::setWorldMaterial (
    const double density = 1. )
```

Definition at line 29 of file [Material.cpp](#).

7.17.4 Member Data Documentation

7.17.4.1 alpineMaterial

```
G4Material* Material::alpineMaterial [private]
```

Definition at line 19 of file [Material.h](#).

7.17.4.2 boardMaterial

```
G4Material* Material::boardMaterial [private]
```

Definition at line 20 of file [Material.h](#).

7.17.4.3 elAl

```
G4Element* Material::elAl [private]
```

Definition at line 28 of file [Material.h](#).

7.17.4.4 elBr

```
G4Element* Material::elBr [private]
```

Definition at line 36 of file [Material.h](#).

7.17.4.5 elC

```
G4Element* Material::elC [private]
```

Definition at line 34 of file [Material.h](#).

7.17.4.6 elCa

```
G4Element* Material::elCa [private]
```

Definition at line 30 of file [Material.h](#).

7.17.4.7 elFe

```
G4Element* Material::elFe [private]
```

Definition at line 29 of file [Material.h](#).

7.17.4.8 elH

```
G4Element* Material::elH [private]
```

Definition at line 35 of file [Material.h](#).

7.17.4.9 elMg

```
G4Element* Material::elMg [private]
```

Definition at line 31 of file [Material.h](#).

7.17.4.10 elN

```
G4Element* Material::elN [private]
```

Definition at line 26 of file [Material.h](#).

7.17.4.11 elNa

```
G4Element* Material::elNa [private]
```

Definition at line 32 of file [Material.h](#).

7.17.4.12 eIO

```
G4Element* Material::eIO [private]
```

Definition at line 27 of file [Material.h](#).

7.17.4.13 eSi

```
G4Element* Material::eSi [private]
```

Definition at line 25 of file [Material.h](#).

7.17.4.14 eTi

```
G4Element* Material::eTi [private]
```

Definition at line 33 of file [Material.h](#).

7.17.4.15 epoxyResin

```
G4Material* Material::epoxyResin [private]
```

Definition at line 22 of file [Material.h](#).

7.17.4.16 fibrousGlass

```
G4Material* Material::fibrousGlass [private]
```

Definition at line 23 of file [Material.h](#).

7.17.4.17 screenMaterial

```
G4Material* Material::screenMaterial [private]
```

Definition at line 18 of file [Material.h](#).

7.17.4.18 standMaterial

```
G4Material* Material::standMaterial [private]
```

Definition at line 17 of file [Material.h](#).

7.17.4.19 worldMaterial

```
G4Material* Material::worldMaterial [private]
```

Definition at line 16 of file [Material.h](#).

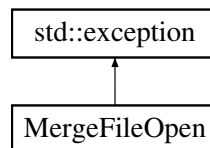
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/Material.h](#)
- [/home/ychoi/ATOM/geant4/main/src/Material.cpp](#)

7.18 MergeFileOpen Class Reference

```
#include <TMerge.h>
```

Inheritance diagram for MergeFileOpen:



Public Member Functions

- [MergeFileOpen](#) (std::string_view fileName)
- const char * [what](#) () const throw ()

Public Attributes

- std::string [message](#)

7.18.1 Detailed Description

Definition at line 28 of file [TMerge.h](#).

7.18.2 Constructor & Destructor Documentation

7.18.2.1 MergeFileOpen()

```
MergeFileOpen::MergeFileOpen (  
    std::string_view fileName ) [inline]
```

Definition at line 32 of file [TMerge.h](#).

7.18.3 Member Function Documentation

7.18.3.1 what()

```
const char * MergeFileOpen::what ( ) const throw ( )    [inline]
```

Definition at line 35 of file [TMerge.h](#).

7.18.4 Member Data Documentation

7.18.4.1 message

```
std::string MergeFileOpen::message
```

Definition at line 30 of file [TMerge.h](#).

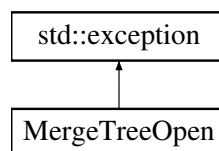
The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/alpide/comparison/inc/TMerge.h](#)

7.19 MergeTreeOpen Class Reference

```
#include <TMerge.h>
```

Inheritance diagram for MergeTreeOpen:



Public Member Functions

- [MergeTreeOpen](#) (std::string_view fileName)
- const char * [what](#) () const throw ()

Public Attributes

- std::string [message](#)

7.19.1 Detailed Description

Definition at line 40 of file [TMerge.h](#).

7.19.2 Constructor & Destructor Documentation

7.19.2.1 MergeTreeOpen()

```

MergeTreeOpen::MergeTreeOpen (
    std::string_view fileName ) [inline]

```

Definition at line 44 of file [TMerge.h](#).

7.19.3 Member Function Documentation

7.19.3.1 what()

```

const char * MergeTreeOpen::what ( ) const throw ( ) [inline]

```

Definition at line 47 of file [TMerge.h](#).

7.19.4 Member Data Documentation

7.19.4.1 message

```

std::string MergeTreeOpen::message

```

Definition at line 42 of file [TMerge.h](#).

The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/alpide/comparison/inc/TMerge.h](#)

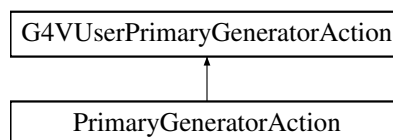
7.20 PrimaryGeneratorAction Class Reference

```

#include <PrimaryGeneratorAction.h>

```

Inheritance diagram for PrimaryGeneratorAction:



Public Member Functions

- [PrimaryGeneratorAction](#) ()
- virtual [~PrimaryGeneratorAction](#) ()
- void [setParticleGun](#) ()
- void [GeneratePrimaries](#) (G4Event *anEvent)
- const G4GeneralParticleSource * [GetParticleGun](#) () const

Private Attributes

- G4GeneralParticleSource * [fParticleGun](#)

7.20.1 Detailed Description

Definition at line 13 of file [PrimaryGeneratorAction.h](#).

7.20.2 Constructor & Destructor Documentation

7.20.2.1 PrimaryGeneratorAction()

```
PrimaryGeneratorAction::PrimaryGeneratorAction ( )
```

Definition at line 4 of file [PrimaryGeneratorAction.cpp](#).

7.20.2.2 ~PrimaryGeneratorAction()

```
PrimaryGeneratorAction::~~PrimaryGeneratorAction ( ) [virtual]
```

Definition at line 14 of file [PrimaryGeneratorAction.cpp](#).

7.20.3 Member Function Documentation

7.20.3.1 GeneratePrimaries()

```
void PrimaryGeneratorAction::GeneratePrimaries (
    G4Event * anEvent )
```

Definition at line 19 of file [PrimaryGeneratorAction.cpp](#).

7.20.3.2 GetParticleGun()

```
const G4GeneralParticleSource * PrimaryGeneratorAction::GetParticleGun ( ) const
```

Definition at line 23 of file [PrimaryGeneratorAction.cpp](#).

7.20.3.3 setParticleGun()

```
void PrimaryGeneratorAction::setParticleGun ( )
```

7.20.4 Member Data Documentation

7.20.4.1 fParticleGun

G4GeneralParticleSource* PrimaryGeneratorAction::fParticleGun [private]

Definition at line 15 of file [PrimaryGeneratorAction.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/PrimaryGeneratorAction.h](#)
- [/home/ychoi/ATOM/geant4/main/src/PrimaryGeneratorAction.cpp](#)

7.21 ProgressBar Class Reference

```
#include <cpptqdm.h>
```

Public Member Functions

- [ProgressBar](#) ()
- [ProgressBar](#) (int setSize)
- [~ProgressBar](#) ()
- void [getTerminalLength](#) ()
- void [printProgress](#) ()
- int [getSecond](#) (int num)
- int [getMinute](#) (int num)

Private Attributes

- std::chrono::system_clock::time_point [start_time](#)
- int [mTerminalWidth](#)
- int [mSetSize](#)
- std::chrono::system_clock::time_point [printPoint](#)
- int [called](#) = 0

7.21.1 Detailed Description

Definition at line 12 of file [cpptqdm.h](#).

7.21.2 Constructor & Destructor Documentation

7.21.2.1 ProgressBar() [1/2]

```
ProgressBar::ProgressBar ( )
```

Definition at line 10 of file [cpptqdm.cpp](#).

7.21.2.2 ProgressBar() [2/2]

```
ProgressBar::ProgressBar (
    int setSize )
```

Definition at line 3 of file [cpptqdm.cpp](#).

7.21.2.3 ~ProgressBar()

```
ProgressBar::~~ProgressBar ( )
```

Definition at line 16 of file [cpptqdm.cpp](#).

7.21.3 Member Function Documentation

7.21.3.1 getMinute()

```
int ProgressBar::getMinute (
    int num )
```

Definition at line 26 of file [cpptqdm.cpp](#).

7.21.3.2 getSecond()

```
int ProgressBar::getSecond (
    int num )
```

Definition at line 30 of file [cpptqdm.cpp](#).

7.21.3.3 getTerminalLength()

```
void ProgressBar::getTerminalLength ( )
```

Definition at line 20 of file [cpptqdm.cpp](#).

7.21.3.4 printProgress()

```
void ProgressBar::printProgress ( )
```

Definition at line 34 of file [cpptqdm.cpp](#).

7.21.4 Member Data Documentation

7.21.4.1 called

```
int ProgressBar::called = 0 [private]
```

Definition at line 18 of file [cpptqdm.h](#).

7.21.4.2 mSetSize

```
int ProgressBar::mSetSize [private]
```

Definition at line 16 of file [cpptqdm.h](#).

7.21.4.3 mTerminalWidth

```
int ProgressBar::mTerminalWidth [private]
```

Definition at line 15 of file [cpptqdm.h](#).

7.21.4.4 printPoint

```
std::chrono::system_clock::time_point ProgressBar::printPoint [private]
```

Definition at line 17 of file [cpptqdm.h](#).

7.21.4.5 start_time

```
std::chrono::system_clock::time_point ProgressBar::start_time [private]
```

Definition at line 14 of file [cpptqdm.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cpptqdm.h](#)
- [/home/ychoi/ATOM/pycpp/src/cpptqdm.cpp](#)

7.22 Quantity Class Reference

```
#include <cppUnit.h>
```

Public Member Functions

- [Quantity](#) ()=delete
- [Quantity](#) (std::string quantity)
- [Quantity](#) (double num, std::string unit)
- double [getNum](#) () const
- double [getNum](#) (std::string_view unit) const
- const std::string [getUnit](#) () const
- const std::string [getQuantity](#) () const
- const std::string [getQuantity](#) (std::string_view unit) const
- [Quantity](#) operator+ (const [Quantity](#) &ref) const
- [Quantity](#) operator- (const [Quantity](#) &ref) const
- [Quantity](#) operator* (const [Quantity](#) &ref) const
- [Quantity](#) operator/ (const [Quantity](#) &ref) const
- [Quantity](#) operator+= (const [Quantity](#) &ref)
- [Quantity](#) operator-= (const [Quantity](#) &ref)
- [Quantity](#) operator*= (const [Quantity](#) &ref)
- [Quantity](#) operator/= (const [Quantity](#) &ref)

Static Public Member Functions

- static void [setUserQuantity](#) ()

Private Attributes

- double [mNum](#)
- int [mDigit](#)
- [Unit](#) [mUnit](#)

Static Private Attributes

- static std::vector< std::tuple< std::string, std::string, [Unit](#) > > [userQuantity](#) = { }

Friends

- std::ostream & [operator<<](#) (std::ostream &os, [Quantity](#) &ref)
- std::ostream & [operator<<](#) (std::ostream &os, const [Quantity](#) &ref)

7.22.1 Detailed Description

Definition at line [52](#) of file [cppUnit.h](#).

7.22.2 Constructor & Destructor Documentation

7.22.2.1 [Quantity\(\)](#) [1/3]

```
Quantity::Quantity ( ) [delete]
```

7.22.2.2 [Quantity\(\)](#) [2/3]

```
Quantity::Quantity (
    std::string quantity )
```

Definition at line [283](#) of file [cppUnit.cpp](#).

7.22.2.3 [Quantity\(\)](#) [3/3]

```
Quantity::Quantity (
    double num,
    std::string unit )
```

Definition at line [300](#) of file [cppUnit.cpp](#).

7.22.3 Member Function Documentation

7.22.3.1 `getNum()` [1/2]

```
double Quantity::getNum ( ) const
```

Definition at line 306 of file `cppUnit.cpp`.

7.22.3.2 `getNum()` [2/2]

```
double Quantity::getNum (
    std::string_view unit ) const
```

Definition at line 310 of file `cppUnit.cpp`.

7.22.3.3 `getQuantity()` [1/2]

```
const std::string Quantity::getQuantity ( ) const
```

Definition at line 322 of file `cppUnit.cpp`.

7.22.3.4 `getQuantity()` [2/2]

```
const std::string Quantity::getQuantity (
    std::string_view unit ) const
```

Definition at line 326 of file `cppUnit.cpp`.

7.22.3.5 `getUnit()`

```
const std::string Quantity::getUnit ( ) const
```

Definition at line 318 of file `cppUnit.cpp`.

7.22.3.6 `operator*()`

```
Quantity Quantity::operator* (
    const Quantity & ref ) const
```

Definition at line 344 of file `cppUnit.cpp`.

7.22.3.7 `operator*=()`

```
Quantity Quantity::operator*= (
    const Quantity & ref )
```

Definition at line 376 of file `cppUnit.cpp`.

7.22.3.8 operator+()

```
Quantity Quantity::operator+ (
    const Quantity & ref ) const
```

Definition at line 334 of file `cppUnit.cpp`.

7.22.3.9 operator+=()

```
Quantity Quantity::operator+= (
    const Quantity & ref )
```

Definition at line 354 of file `cppUnit.cpp`.

7.22.3.10 operator-()

```
Quantity Quantity::operator- (
    const Quantity & ref ) const
```

Definition at line 339 of file `cppUnit.cpp`.

7.22.3.11 operator-=()

```
Quantity Quantity::operator-= (
    const Quantity & ref )
```

Definition at line 365 of file `cppUnit.cpp`.

7.22.3.12 operator/()

```
Quantity Quantity::operator/ (
    const Quantity & ref ) const
```

Definition at line 349 of file `cppUnit.cpp`.

7.22.3.13 operator/=()

```
Quantity Quantity::operator/= (
    const Quantity & ref )
```

Definition at line 384 of file `cppUnit.cpp`.

7.22.3.14 setUserQuantity()

```
void Quantity::setUserQuantity ( ) [static]
```

Definition at line 36 of file `cppUnit.cpp`.

7.22.4 Friends And Related Symbol Documentation

7.22.4.1 `operator<<` [1/2]

```
std::ostream & operator<< (  
    std::ostream & os,  
    const Quantity & ref ) [friend]
```

Definition at line 398 of file [cppUnit.cpp](#).

7.22.4.2 `operator<<` [2/2]

```
std::ostream & operator<< (  
    std::ostream & os,  
    Quantity & ref ) [friend]
```

Definition at line 393 of file [cppUnit.cpp](#).

7.22.5 Member Data Documentation

7.22.5.1 `mDigit`

```
int Quantity::mDigit [private]
```

Definition at line 55 of file [cppUnit.h](#).

7.22.5.2 `mNum`

```
double Quantity::mNum [private]
```

Definition at line 54 of file [cppUnit.h](#).

7.22.5.3 `mUnit`

```
Unit Quantity::mUnit [private]
```

Definition at line 56 of file [cppUnit.h](#).

7.22.5.4 `userQuantity`

```
std::vector< std::tuple< std::string, std::string, Unit > > Quantity::userQuantity = { }  
[static], [private]
```

Definition at line 9 of file [cppUnit.h](#).

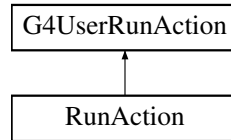
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cppUnit.h](#)
- [/home/ychoi/ATOM/pycpp/src/cppUnit.cpp](#)

7.23 RunAction Class Reference

```
#include <RunAction.h>
```

Inheritance diagram for RunAction:



Public Member Functions

- [RunAction](#) ()
- virtual [~RunAction](#) ()
- virtual void [BeginOfRunAction](#) (const G4Run *)
- virtual void [EndOfRunAction](#) (const G4Run *)

Public Attributes

- [AnalysisManager](#) * [fAnalysisManager](#)

7.23.1 Detailed Description

Definition at line 17 of file [RunAction.h](#).

7.23.2 Constructor & Destructor Documentation

7.23.2.1 RunAction()

```
RunAction::RunAction ( )
```

Definition at line 4 of file [RunAction.cpp](#).

7.23.2.2 ~RunAction()

```
RunAction::~~RunAction ( ) [virtual]
```

Definition at line 9 of file [RunAction.cpp](#).

7.23.3 Member Function Documentation

7.23.3.1 BeginOfRunAction()

```
void RunAction::BeginOfRunAction (
    const G4Run * run ) [virtual]
```

Definition at line 13 of file [RunAction.cpp](#).

7.23.3.2 EndOfRunAction()

```
void RunAction::EndOfRunAction (
    const G4Run * run ) [virtual]
```

Definition at line 18 of file [RunAction.cpp](#).

7.23.4 Member Data Documentation

7.23.4.1 fAnalysisManager

```
AnalysisManager* RunAction::fAnalysisManager
```

Definition at line 25 of file [RunAction.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/RunAction.h](#)
- [/home/ychoi/ATOM/geant4/main/src/RunAction.cpp](#)

7.24 RunTuple Struct Reference

```
#include <AnalysisManager.h>
```

Public Attributes

- int [runID](#)
- int [nEvents](#)

7.24.1 Detailed Description

Definition at line 24 of file [AnalysisManager.h](#).

7.24.2 Member Data Documentation

7.24.2.1 nEvents

```
int RunTuple::nEvents
```

Definition at line 26 of file [AnalysisManager.h](#).

7.24.2.2 runID

```
int RunTuple::runID
```

Definition at line 25 of file [AnalysisManager.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h](#)

7.25 Solid Class Reference

```
#include <Solid.h>
```

Public Member Functions

- [Solid](#) ()
- void [setAlphaStandSolid](#) ()
- void [setBetaStandSolid](#) ()
- void [setNewStandSolid](#) (double diameter)
- void [setScreenSolid](#) ()
- void [setAlpideCircuitSolid](#) ()
- void [setAlpideEpitaxialSolid](#) ()
- void [setBoardSolid](#) ()
- G4VSolid * [getAlphaStandSolid](#) () const
- G4VSolid * [getBetaStandSolid](#) () const
- G4VSolid * [getNewStandSolid](#) () const
- G4VSolid * [getScreenSolid](#) () const
- G4VSolid * [getAlpideCircuitSolid](#) () const
- G4VSolid * [getAlpideEpitaxialSolid](#) () const
- G4VSolid * [getBoardSolid](#) () const

Private Attributes

- G4VSolid * [alphaStandSolid](#)
- G4VSolid * [betaStandSolid](#)
- G4VSolid * [newStandSolid](#)
- G4VSolid * [screenSolid](#)
- G4VSolid * [alpideCircuitSolid](#)
- G4VSolid * [alpideEpitaxialSolid](#)
- G4VSolid * [boardSolid](#)

7.25.1 Detailed Description

Definition at line 15 of file [Solid.h](#).

7.25.2 Constructor & Destructor Documentation

7.25.2.1 Solid()

```
Solid::Solid ( )
```

Definition at line 6 of file [Solid.cpp](#).

7.25.3 Member Function Documentation

7.25.3.1 getAlphaStandSolid()

```
G4VSolid * Solid::getAlphaStandSolid ( ) const
```

Definition at line 63 of file [Solid.cpp](#).

7.25.3.2 getAlpideCircuitSolid()

```
G4VSolid * Solid::getAlpideCircuitSolid ( ) const
```

Definition at line 79 of file [Solid.cpp](#).

7.25.3.3 getAlpideEpitaxialSolid()

```
G4VSolid * Solid::getAlpideEpitaxialSolid ( ) const
```

Definition at line 83 of file [Solid.cpp](#).

7.25.3.4 getBetaStandSolid()

```
G4VSolid * Solid::getBetaStandSolid ( ) const
```

Definition at line 67 of file [Solid.cpp](#).

7.25.3.5 getBoardSolid()

```
G4VSolid * Solid::getBoardSolid ( ) const
```

Definition at line 87 of file [Solid.cpp](#).

7.25.3.6 getNewStandSolid()

```
G4VSolid * Solid::getNewStandSolid ( ) const
```

Definition at line 71 of file [Solid.cpp](#).

7.25.3.7 `getScreenSolid()`

```
G4VSolid * Solid::getScreenSolid ( ) const
```

Definition at line 75 of file [Solid.cpp](#).

7.25.3.8 `setAlphaStandSolid()`

```
void Solid::setAlphaStandSolid ( )
```

Definition at line 12 of file [Solid.cpp](#).

7.25.3.9 `setAlpideCircuitSolid()`

```
void Solid::setAlpideCircuitSolid ( )
```

Definition at line 49 of file [Solid.cpp](#).

7.25.3.10 `setAlpideEpitaxialSolid()`

```
void Solid::setAlpideEpitaxialSolid ( )
```

Definition at line 53 of file [Solid.cpp](#).

7.25.3.11 `setBetaStandSolid()`

```
void Solid::setBetaStandSolid ( )
```

Definition at line 21 of file [Solid.cpp](#).

7.25.3.12 `setBoardSolid()`

```
void Solid::setBoardSolid ( )
```

Definition at line 57 of file [Solid.cpp](#).

7.25.3.13 `setNewStandSolid()`

```
void Solid::setNewStandSolid (
    double diameter )
```

Definition at line 36 of file [Solid.cpp](#).

7.25.3.14 setScreenSolid()

```
void Solid::setScreenSolid ( )
```

Definition at line 45 of file [Solid.cpp](#).

7.25.4 Member Data Documentation

7.25.4.1 alphaStandSolid

```
G4VSolid* Solid::alphaStandSolid [private]
```

Definition at line 17 of file [Solid.h](#).

7.25.4.2 alpideCircuitSolid

```
G4VSolid* Solid::alpideCircuitSolid [private]
```

Definition at line 21 of file [Solid.h](#).

7.25.4.3 alpideEpitaxialSolid

```
G4VSolid* Solid::alpideEpitaxialSolid [private]
```

Definition at line 22 of file [Solid.h](#).

7.25.4.4 betaStandSolid

```
G4VSolid* Solid::betaStandSolid [private]
```

Definition at line 18 of file [Solid.h](#).

7.25.4.5 boardSolid

```
G4VSolid* Solid::boardSolid [private]
```

Definition at line 23 of file [Solid.h](#).

7.25.4.6 newStandSolid

```
G4VSolid* Solid::newStandSolid [private]
```

Definition at line 19 of file [Solid.h](#).

7.25.4.7 screenSolid

```
G4VSolid* Solid::screenSolid [private]
```

Definition at line 20 of file [Solid.h](#).

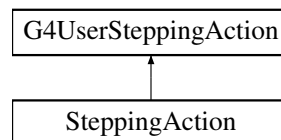
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/Solid.h](#)
- [/home/ychoi/ATOM/geant4/main/src/Solid.cpp](#)

7.26 SteppingAction Class Reference

```
#include <SteppingAction.h>
```

Inheritance diagram for SteppingAction:



Public Member Functions

- [SteppingAction](#) ([EventAction](#) *eventAction)
- virtual [~SteppingAction](#) ()
- virtual void [UserSteppingAction](#) (const G4Step *)

7.26.1 Detailed Description

Definition at line 15 of file [SteppingAction.h](#).

7.26.2 Constructor & Destructor Documentation

7.26.2.1 SteppingAction()

```
SteppingAction::SteppingAction (
    EventAction * eventAction )
```

Definition at line 4 of file [SteppingAction.cpp](#).

7.26.2.2 ~SteppingAction()

```
SteppingAction::~SteppingAction ( ) [virtual]
```

Definition at line 8 of file [SteppingAction.cpp](#).

7.26.3 Member Function Documentation

7.26.3.1 UserSteppingAction()

```
void SteppingAction::UserSteppingAction (
    const G4Step * step ) [virtual]
```

Definition at line 12 of file [SteppingAction.cpp](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/SteppingAction.h](#)
- [/home/ychoi/ATOM/geant4/main/src/SteppingAction.cpp](#)

7.27 StepTuple Struct Reference

```
#include <AnalysisManager.h>
```

Public Attributes

- int [trackGlobalID](#)
- int [stepGlobalID](#)
- std::string [volumeName](#)
- double [time](#)
- double [position](#) [3]
- double [kineticEnergy](#)
- double [momentum](#) [3]
- double [deltaEnergy](#)
- double [totalDepositEnergy](#)
- double [nonIonizingEnergyLoss](#)

7.27.1 Detailed Description

Definition at line 51 of file [AnalysisManager.h](#).

7.27.2 Member Data Documentation

7.27.2.1 deltaEnergy

```
double StepTuple::deltaEnergy
```

Definition at line 59 of file [AnalysisManager.h](#).

7.27.2.2 kineticEnergy

```
double StepTuple::kineticEnergy
```

Definition at line 57 of file [AnalysisManager.h](#).

7.27.2.3 momentum

```
double StepTuple::momentum[3]
```

Definition at line 58 of file [AnalysisManager.h](#).

7.27.2.4 nonIonizingEnergyLoss

```
double StepTuple::nonIonizingEnergyLoss
```

Definition at line 61 of file [AnalysisManager.h](#).

7.27.2.5 position

```
double StepTuple::position[3]
```

Definition at line 56 of file [AnalysisManager.h](#).

7.27.2.6 stepGlobalID

```
int StepTuple::stepGlobalID
```

Definition at line 53 of file [AnalysisManager.h](#).

7.27.2.7 time

```
double StepTuple::time
```

Definition at line 55 of file [AnalysisManager.h](#).

7.27.2.8 totalDepositEnergy

```
double StepTuple::totalDepositEnergy
```

Definition at line 60 of file [AnalysisManager.h](#).

7.27.2.9 trackGlobalID

```
int StepTuple::trackGlobalID
```

Definition at line 52 of file [AnalysisManager.h](#).

7.27.2.10 volumeName

```
std::string StepTuple::volumeName
```

Definition at line 54 of file [AnalysisManager.h](#).

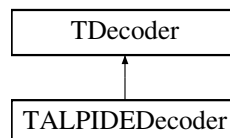
The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h](#)

7.28 TALPIDEDecoder Class Reference

```
#include <TALPIDEDecoder.h>
```

Inheritance diagram for TALPIDEDecoder:



Public Member Functions

- [TALPIDEDecoder](#) (const std::filesystem::path &binaryPath)
- [TALPIDEDecoder](#) (const std::string &binaryPath)
- void [decode](#) ()
- std::vector< std::unique_ptr< [TALPIDEEvent](#) > > [getData](#) ()

Public Member Functions inherited from [TDecoder](#)

- [TDecoder](#) (const std::filesystem::path &binaryPath)
- [TDecoder](#) (const std::string &binaryPath)
- void [readFile](#) ()
- int [getDataLength](#) ()
- std::vector< uint8_t > & [getBinaryData](#) ()

Private Member Functions

- void [inputEvent](#) ()
- bool [isDone](#) ()
- void [preTest](#) ()
- bool [strayTest](#) ()
- void [headerTest](#) ()
- long int [hex_to_dec](#) (const uint8_t *data, const int &digits)
- int [bitwise_and](#) (int v1, int v2)
- int [bitwise_or](#) (int v1, int v2)
- int [bitwise_xor](#) (int v1, int v2)

Private Attributes

- `std::vector< std::unique_ptr< TALPIDEEvent > > alpides`
- `int index_ = 0`

7.28.1 Detailed Description

Definition at line 18 of file [TALPIDEDecoder.h](#).

7.28.2 Constructor & Destructor Documentation

7.28.2.1 [TALPIDEDecoder\(\)](#) [1/2]

```
TALPIDEDecoder::TALPIDEDecoder (  
    const std::filesystem::path & binaryPath )
```

Definition at line 6 of file [TALPIDEDecoder.cpp](#).

7.28.2.2 [TALPIDEDecoder\(\)](#) [2/2]

```
TALPIDEDecoder::TALPIDEDecoder (  
    const std::string & binaryPath )
```

Definition at line 7 of file [TALPIDEDecoder.cpp](#).

7.28.3 Member Function Documentation

7.28.3.1 [bitwise_and\(\)](#)

```
int TALPIDEDecoder::bitwise_and (  
    int v1,  
    int v2 ) [private]
```

Definition at line 119 of file [TALPIDEDecoder.cpp](#).

7.28.3.2 [bitwise_or\(\)](#)

```
int TALPIDEDecoder::bitwise_or (  
    int v1,  
    int v2 ) [private]
```

Definition at line 132 of file [TALPIDEDecoder.cpp](#).

7.28.3.3 bitwise_xor()

```
int TALPIDEDecoder::bitwise_xor (
    int v1,
    int v2 ) [private]
```

Definition at line 145 of file [TALPIDEDecoder.cpp](#).

7.28.3.4 decode()

```
void TALPIDEDecoder::decode ( )
```

Definition at line 9 of file [TALPIDEDecoder.cpp](#).

7.28.3.5 getData()

```
std::vector< std::unique_ptr< TALPIDEEvent > > TALPIDEDecoder::getData ( )
```

Definition at line 16 of file [TALPIDEDecoder.cpp](#).

7.28.3.6 headerTest()

```
void TALPIDEDecoder::headerTest ( ) [private]
```

Definition at line 104 of file [TALPIDEDecoder.cpp](#).

7.28.3.7 hex_to_dec()

```
long int TALPIDEDecoder::hex_to_dec (
    const uint8_t * data,
    const int & digits ) [private]
```

Definition at line 111 of file [TALPIDEDecoder.cpp](#).

7.28.3.8 inputEvent()

```
void TALPIDEDecoder::inputEvent ( ) [private]
```

Definition at line 20 of file [TALPIDEDecoder.cpp](#).

7.28.3.9 isDone()

```
bool TALPIDEDecoder::isDone ( ) [private]
```

Definition at line 158 of file [TALPIDEDecoder.cpp](#).

7.28.3.10 preTest()

```
void TALPIDEDecoder::preTest ( ) [private]
```

Definition at line 88 of file [TALPIDEDecoder.cpp](#).

7.28.3.11 strayTest()

```
bool TALPIDEDecoder::strayTest ( ) [private]
```

Definition at line 95 of file [TALPIDEDecoder.cpp](#).

7.28.4 Member Data Documentation

7.28.4.1 alpides

```
std::vector<std::unique_ptr<TALPIDEEvent> > TALPIDEDecoder::alpides [private]
```

Definition at line 20 of file [TALPIDEDecoder.h](#).

7.28.4.2 index_

```
int TALPIDEDecoder::index_ = 0 [private]
```

Definition at line 21 of file [TALPIDEDecoder.h](#).

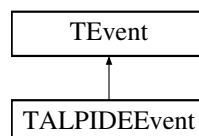
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/daq/inc/TALPIDEDecoder.h](#)
- [/home/ychoi/ATOM/alpide/daq/src/TALPIDEDecoder.cpp](#)

7.29 TALPIDEEvent Class Reference

```
#include <TALPIDEEvent.h>
```

Inheritance diagram for TALPIDEEvent:



Public Types

- enum { [kNotDeleted](#) = 0x02000000 }

Public Member Functions

- [TALPIDEEvent](#) ()
- [TALPIDEEvent](#) (const [TALPIDEEvent](#) ©)
- [TALPIDEEvent](#) & [operator=](#) (const [TALPIDEEvent](#) ©)
- [TALPIDEEvent](#) ([TALPIDEEvent](#) &&move)
- [TALPIDEEvent](#) & [operator=](#) ([TALPIDEEvent](#) &&move)
- void [setTime](#) (const long int time)
- const long int [getTime](#) () const
- const std::vector< std::pair< int, int > > & [getData](#) () const
- void [removePixel](#) (const std::pair< int, int > &coordinate)
- void [pushData](#) (const std::pair< int, int > &coordinate)
- void [removeDuplication](#) ()
- void [sortPixel](#) ()
- const int [getNData](#) () const
- bool [IsDestroyed](#) () const
- bool [TestBit](#) (unsigned int f) const

Public Member Functions inherited from [TEvent](#)

- [TEvent](#) ()
- virtual [~TEvent](#) ()
- void [setEvent](#) (const int event)
- const int [getEvent](#) () const

Private Attributes

- long int [iTime](#)
- std::vector< std::pair< int, int > > [data](#)
- unsigned int [fBits](#)

7.29.1 Detailed Description

Definition at line 14 of file [TALPIDEEvent.h](#).

7.29.2 Member Enumeration Documentation**7.29.2.1 anonymous enum**

anonymous enum

Enumerator

kNotDeleted	
-------------	--

Definition at line 43 of file [TALPIDEEvent.h](#).

7.29.3 Constructor & Destructor Documentation

7.29.3.1 TALPIDEEvent() [1/3]

```
TALPIDEEvent::TALPIDEEvent ( )
```

Definition at line 4 of file [TALPIDEEvent.cpp](#).

7.29.3.2 TALPIDEEvent() [2/3]

```
TALPIDEEvent::TALPIDEEvent (
    const TALPIDEEvent & copy )
```

Definition at line 6 of file [TALPIDEEvent.cpp](#).

7.29.3.3 TALPIDEEvent() [3/3]

```
TALPIDEEvent::TALPIDEEvent (
    TALPIDEEvent && move )
```

Definition at line 19 of file [TALPIDEEvent.cpp](#).

7.29.4 Member Function Documentation

7.29.4.1 getData()

```
const std::vector< std::pair< int, int > > & TALPIDEEvent::getData ( ) const
```

Definition at line 52 of file [TALPIDEEvent.cpp](#).

7.29.4.2 getNData()

```
const int TALPIDEEvent::getNData ( ) const
```

Definition at line 87 of file [TALPIDEEvent.cpp](#).

7.29.4.3 getTime()

```
const long int TALPIDEEvent::getTime ( ) const
```

Definition at line 48 of file [TALPIDEEvent.cpp](#).

7.29.4.4 IsDestructed()

```
bool TALPIDEEvent::IsDestructed ( ) const [inline]
```

Definition at line 46 of file [TALPIDEEvent.h](#).

7.29.4.5 operator=() [1/2]

```
TALPIDEEvent & TALPIDEEvent::operator= (
    const TALPIDEEvent & copy )
```

Definition at line 11 of file [TALPIDEEvent.cpp](#).

7.29.4.6 operator=() [2/2]

```
TALPIDEEvent & TALPIDEEvent::operator= (
    TALPIDEEvent && move )
```

Definition at line 28 of file [TALPIDEEvent.cpp](#).

7.29.4.7 pushData()

```
void TALPIDEEvent::pushData (
    const std::pair< int, int > & coordinate )
```

Definition at line 44 of file [TALPIDEEvent.cpp](#).

7.29.4.8 removeDuplication()

```
void TALPIDEEvent::removeDuplication ( )
```

Definition at line 60 of file [TALPIDEEvent.cpp](#).

7.29.4.9 removePixel()

```
void TALPIDEEvent::removePixel (
    const std::pair< int, int > & coordinate )
```

Definition at line 56 of file [TALPIDEEvent.cpp](#).

7.29.4.10 setTime()

```
void TALPIDEEvent::setTime (
    const long int time )
```

Definition at line 40 of file [TALPIDEEvent.cpp](#).

7.29.4.11 sortPixel()

```
void TALPIDEEvent::sortPixel ( )
```

Definition at line 78 of file [TALPIDEEvent.cpp](#).

7.29.4.12 TestBit()

```
bool TALPIDEEvent::TestBit (
    unsigned int f ) const [inline]
```

Definition at line 47 of file [TALPIDEEvent.h](#).

7.29.5 Member Data Documentation

7.29.5.1 data

```
std::vector<std::pair<int, int> > TALPIDEEvent::data [private]
```

Definition at line 17 of file [TALPIDEEvent.h](#).

7.29.5.2 fBits

```
unsigned int TALPIDEEvent::fBits [private]
```

Definition at line 41 of file [TALPIDEEvent.h](#).

7.29.5.3 iTime

```
long int TALPIDEEvent::iTime [private]
```

Definition at line 16 of file [TALPIDEEvent.h](#).

The documentation for this class was generated from the following files:

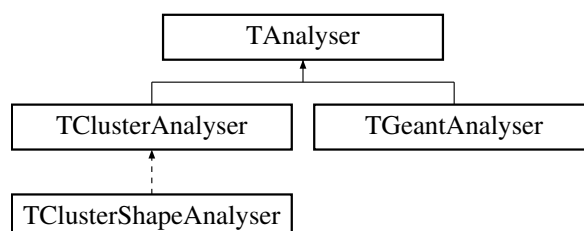
- [/home/ychoi/ATOM/alpide/daq/inc/TALPIDEEvent.h](#)
- [/home/ychoi/ATOM/alpide/daq/src/TALPIDEEvent.cpp](#)

7.30 TAnalyser Class Reference

For ROOT and config file when analysis.

```
#include <TAnalyser.h>
```

Inheritance diagram for TAnalyser:



Public Types

- enum { `kNotDeleted` = 0x02000000 }

Public Member Functions

- `TAnalyser` ()=default
- `TAnalyser` (TFile *inputFile, std::unordered_map< std::string, TExperimentData * > expData)
Construct a new TAnalyser::TAnalyser object.
- `~TAnalyser` ()
Destroy the TAnalyser::TAnalyser object.
- TTree * `openTree` (std::string treeName)
open trees
- void `storeEvents` (CppConfigDictionary settingConfig)
Extract event from TTree.
- void `doMasking` (int mMaskOver)
- void `openOutputGraphFile` (std::string_view fileName)
Open ROOT file to save results graph.
- void `openDirectory` (std::string_view typeName)
Set the new directory to store graph root file.
- void `setExpSettingLegend` (CppConfigDictionary settingConfig)
- void `doDivideBySize` (std::string_view typeName)
- TExperimentData * `getAnEventSet` (std::string_view typeName) const
- TH2D * `getHitPlot` (const CppConfigDictionary &config, const std::vector< TALPIDEEvent * > &events)
- void `saveHitmap` (std::string typeName, const CppConfigDictionary &config)
- TH2D * `getClusterPlot` (const CppConfigDictionary &config, const std::vector< TCluster * > &clusters)
Generalized function to draw clustermap.
- TH1D * `getClustersizePlot` (const CppConfigDictionary &config, const std::vector< TCluster * > &clusters)
- void `setClusterDataWithShape` (const std::vector< int > &clusterSizeRange)
- void `saveClustermap` (std::string typeName, const CppConfigDictionary &config)
- void `saveClustersize` (std::string typeName, const CppConfigDictionary &config)
- std::vector< int > `getClusterSizeRange` (const CppConfigDictionary &privateProperty)
- void `doShaping` (std::string_view typeName, const std::vector< int > &clusterSizeRange)
Store clusters to objects of TClusterShape for extracting shape informations.
- void `saveIndividualShapes` (std::string_view typeName, const CppConfigDictionary config)
Drawing individual shapes.
- void `saveSameSizeInfos` (std::string_view typeName, const CppConfigDictionary config)
Drawing information for classifying multi-cluster.
- void `saveSameSizeShapes` (std::string_view typeName, const CppConfigDictionary config)
- void `saveTotalShapes` (std::string_view typeName, const CppConfigDictionary config)
- void `saveSameSizeShapeEntry` (std::string_view typeName, const CppConfigDictionary config)
- void `saveTotalShapeEntry` (std::string_view typeName, const CppConfigDictionary config)
- bool `IsDestructed` () const
- bool `TestBit` (UInt_t f) const

Protected Attributes

- TFile * [mOutputFile](#) = nullptr
- bool [mIsOutputGraph](#) = false
- TPaveText * [mExpSettingLegend](#)
- std::unordered_map< std::string, TH2D * > [mHitmaps](#)
- std::unordered_map< std::string, TH2D * > [mClustermaps](#)
- std::unordered_map< std::string, TH1D * > [mClustersizes](#)
- std::unordered_map< std::string, std::unordered_map< int, std::vector< TCluster * > > > [mClusterDataWithShape](#)
- std::unordered_map< std::string, TDirectory * > [mDirectorySet](#)
- std::unordered_map< std::string, TExperimentData * > [mExpData](#)
- std::unordered_map< std::string, TClusterDivideData * > [mDivideData](#)
- std::unordered_map< std::string, std::vector< TClusterShape * > > [mClusterShapeSet](#)
- std::unordered_map< std::string, int > [mNTotalShapeSet](#)
- std::unordered_map< std::string, int > [mMaxModeSet](#)

Private Attributes

- TFile * [mInputFile](#) = nullptr
- TTree * [mTree](#)
- TInputRoot [mInput](#)
- UInt_t [fBits](#)

7.30.1 Detailed Description

For ROOT and config file when analysis.

It store ROOT file and Config file. It provide open and access such kind files. It is made for being mother class of Analysis class.

Warning

Bug

Todo Add template for plots. Map, distribution, etc.

Definition at line 79 of file [TAnalyser.h](#).

7.30.2 Member Enumeration Documentation

7.30.2.1 anonymous enum

anonymous enum

Enumerator

kNotDeleted	
-------------	--

Definition at line 141 of file [TAnalyser.h](#).

7.30.3 Constructor & Destructor Documentation

7.30.3.1 TAnalyser() [1/2]

```
TAnalyser::TAnalyser ( ) [default]
```

7.30.3.2 TAnalyser() [2/2]

```
TAnalyser::TAnalyser (
    TFile * inputFile,
    std::unordered_map< std::string, TExperimentData * > expData )
```

Construct a new [TAnalyser::TAnalyser](#) object.

It opens raw root file and 'hit' tree. And it matches branch name and address.

Parameters

<i>inputFile</i>	
<i>expData</i>	

Getting File path

It prints out the constructor message. It outputs the file name.

'hit' tree is opened. And the branches are matched to each variables.

It sets ignore level.

Definition at line 11 of file [TAnalyser.cpp](#).

7.30.3.3 ~TAnalyser()

```
TAnalyser::~TAnalyser ( )
```

Destroy the [TAnalyser::TAnalyser](#) object.

Todo The desctructors are commented out. It should be set.

Definition at line 31 of file [TAnalyser.cpp](#).

7.30.4 Member Function Documentation

7.30.4.1 doDivideBySize()

```
void TAnalyser::doDivideBySize (
    std::string_view typeName )
```

Definition at line 319 of file [TAnalyser.cpp](#).

7.30.4.2 doMasking()

```
void TAnalyser::doMasking (
    int mMaskOver )
```

Definition at line 177 of file [TAnalyser.cpp](#).

7.30.4.3 doShaping()

```
void TAnalyser::doShaping (
    std::string_view typeName,
    const std::vector< int > & clusterSizeRange )
```

Store clusters to objects of [TClusterShape](#) for extracting shape informations.

Parameters

<i>typeName</i>	
<i>clusterSizeRange</i>	

Definition at line 545 of file [TAnalyser.cpp](#).

7.30.4.4 getAnEventSet()

```
TExperimentData * TAnalyser::getAnEventSet (
    std::string_view typeName ) const
```

Definition at line 238 of file [TAnalyser.cpp](#).

7.30.4.5 getClusterPlot()

```
TH2D * TAnalyser::getClusterPlot (
    const CppConfigDictionary & config,
    const std::vector< TCluster * > & clusters )
```

Generalized function to draw clustermap.

Parameters

<i>config</i>	Draw configuration for map title, directory and filename
<i>clusters</i>	Dataset to draw

Returns

const TH2*

Definition at line 332 of file [TAnalyser.cpp](#).

7.30.4.6 getClustersizePlot()

```
TH1D * TAnalyser::getClustersizePlot (
    const CppConfigDictionary & config,
    const std::vector< TCluster * > & clusters )
```

Parameters

<i>config</i>	
<i>clusters</i>	

Returns

TH1D*

Definition at line 410 of file [TAnalyser.cpp](#).

7.30.4.7 getClusterSizeRange()

```
std::vector< int > TAnalyser::getClusterSizeRange (
    const CppConfigDictionary & privateProperty )
```

Definition at line 518 of file [TAnalyser.cpp](#).

7.30.4.8 getHitPlot()

```
TH2D * TAnalyser::getHitPlot (
    const CppConfigDictionary & config,
    const std::vector< TALPIDEEvent * > & events )
```

Definition at line 246 of file [TAnalyser.cpp](#).

7.30.4.9 IsDestructed()

```
bool TAnalyser::IsDestructed ( ) const [inline]
```

Definition at line 144 of file [TAnalyser.h](#).

7.30.4.10 openDirectory()

```
void TAnalyser::openDirectory (
    std::string_view typeName )
```

Set the new directory to store graph root file.

Parameters

<i>typeName</i>	
-----------------	--

Open graph root file

Make new directory

Save the directory into graph root file

Definition at line 98 of file [TAnalyser.cpp](#).

7.30.4.11 openOutputGraphFile()

```
void TAnalyser::openOutputGraphFile (
    std::string_view fileName )
```

Open ROOT file to save results graph.

Parameters

<i>fileName</i>	
-----------------	--

Open output file

Set control variable

Definition at line 88 of file [TAnalyser.cpp](#).

7.30.4.12 openTree()

```
TTree * TAnalyser::openTree (
    std::string treeName )
```

open trees

Parameters

<i>treeName</i>	
-----------------	--

Returns

TTree*

It determines the tree is belong to TFile. If it exists, then this function return TTree.

Date

05/08/2024

Definition at line 68 of file [TAnalyser.cpp](#).

7.30.4.13 saveClustermap()

```
void TAnalyser::saveClustermap (
    std::string typeName,
    const CppConfigDictionary & config )
```

Definition at line 484 of file [TAnalyser.cpp](#).

7.30.4.14 saveClustersize()

```
void TAnalyser::saveClustersize (
    std::string typeName,
    const CppConfigDictionary & config )
```

Definition at line 498 of file [TAnalyser.cpp](#).

7.30.4.15 saveHitmap()

```
void TAnalyser::saveHitmap (
    std::string typeName,
    const CppConfigDictionary & config )
```

Definition at line 305 of file [TAnalyser.cpp](#).

7.30.4.16 saveIndividualShapes()

```
void TAnalyser::saveIndividualShapes (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Drawing individual shapes.

Parameters

<i>typeName</i>	
<i>config</i>	

Definition at line 578 of file [TAnalyser.cpp](#).

7.30.4.17 saveSameSizeInfos()

```
void TAnalyser::saveSameSizeInfos (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Drawing information for classifying multi-cluster.

Parameters

<i>typeName</i>	
<i>config</i>	

Definition at line 732 of file [TAnalyser.cpp](#).

7.30.4.18 saveSameSizeShapeEntry()

```
void TAnalyser::saveSameSizeShapeEntry (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 1093 of file [TAnalyser.cpp](#).

7.30.4.19 saveSameSizeShapes()

```
void TAnalyser::saveSameSizeShapes (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 866 of file [TAnalyser.cpp](#).

7.30.4.20 saveTotalShapeEntry()

```
void TAnalyser::saveTotalShapeEntry (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 1133 of file [TAnalyser.cpp](#).

7.30.4.21 saveTotalShapes()

```
void TAnalyser::saveTotalShapes (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 981 of file [TAnalyser.cpp](#).

7.30.4.22 setClusterDataWithShape()

```
void TAnalyser::setClusterDataWithShape (
    const std::vector< int > & clusterSizeRange )
```

Definition at line 512 of file [TAnalyser.cpp](#).

7.30.4.23 setExpSettingLegend()

```
void TAnalyser::setExpSettingLegend (
    CppConfigDictionary settingConfig )
```

Definition at line 215 of file [TAnalyser.cpp](#).

7.30.4.24 storeEvents()

```
void TAnalyser::storeEvents (
    CppConfigDictionary config )
```

Extract event from TTree.

Parameters

<i>config</i>	
---------------	--

Array for 'TALPIDEEEvent' class. The events extracted from TTree will be saved here.

Variable for storing previous time.

Store initial event to 'TALPIDEEEvent' array.

Open progress bar

Definition at line 109 of file TAnalyser.cpp.

7.30.4.25 TestBit()

```
bool TAnalyser::TestBit (
    UInt_t f ) const [inline]
```

Definition at line 145 of file TAnalyser.h.

7.30.5 Member Data Documentation

7.30.5.1 fBits

```
UInt_t TAnalyser::fBits [private]
```

Definition at line 139 of file TAnalyser.h.

7.30.5.2 mClusterDataWithShape

```
std::unordered_map<std::string, std::unordered_map<int, std::vector<TCluster*> > > TAnalyser←
::mClusterDataWithShape [protected]
```

Definition at line 93 of file TAnalyser.h.

7.30.5.3 mClustermaps

```
std::unordered_map<std::string, TH2D*> TAnalyser::mClustermaps [protected]
```

Definition at line 91 of file TAnalyser.h.

7.30.5.4 mClusterShapeSet

```
std::unordered_map<std::string, std::vector<TClusterShape*> > TAnalyser::mClusterShapeSet
[protected]
```

Definition at line 97 of file TAnalyser.h.

7.30.5.5 mClustersizes

```
std::unordered_map<std::string, TH1D*> TAnalyser::mClustersizes [protected]
```

Definition at line 92 of file [TAnalyser.h](#).

7.30.5.6 mDirectorySet

```
std::unordered_map<std::string, TDirectory*> TAnalyser::mDirectorySet [protected]
```

Definition at line 94 of file [TAnalyser.h](#).

7.30.5.7 mDivideData

```
std::unordered_map<std::string, TClusterDivideData*> TAnalyser::mDivideData [protected]
```

Definition at line 96 of file [TAnalyser.h](#).

7.30.5.8 mExpData

```
std::unordered_map<std::string, TExperimentData*> TAnalyser::mExpData [protected]
```

Definition at line 95 of file [TAnalyser.h](#).

7.30.5.9 mExpSettingLegend

```
TPaveText* TAnalyser::mExpSettingLegend [protected]
```

Definition at line 89 of file [TAnalyser.h](#).

7.30.5.10 mHitmaps

```
std::unordered_map<std::string, TH2D*> TAnalyser::mHitmaps [protected]
```

Definition at line 90 of file [TAnalyser.h](#).

7.30.5.11 mInput

```
TInputRoot TAnalyser::mInput [private]
```

Definition at line 83 of file [TAnalyser.h](#).

7.30.5.12 mInputFile

```
TFile* TAnalyser::mInputFile = nullptr [private]
```

Input file with ROOT extension.

Definition at line 81 of file [TAnalyser.h](#).

7.30.5.13 mIsOutputGraph

```
bool TAnalyser::mIsOutputGraph = false [protected]
```

Definition at line 88 of file [TAnalyser.h](#).

7.30.5.14 mMaxModeSet

```
std::unordered_map<std::string, int> TAnalyser::mMaxModeSet [protected]
```

Definition at line 99 of file [TAnalyser.h](#).

7.30.5.15 mNTotalShapeSet

```
std::unordered_map<std::string, int> TAnalyser::mNTotalShapeSet [protected]
```

Definition at line 98 of file [TAnalyser.h](#).

7.30.5.16 mOutputFile

```
TFile* TAnalyser::mOutputFile = nullptr [protected]
```

Definition at line 87 of file [TAnalyser.h](#).

7.30.5.17 mTree

```
TTree* TAnalyser::mTree [private]
```

Definition at line 82 of file [TAnalyser.h](#).

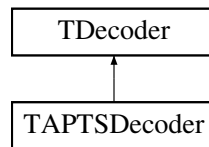
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/analysis/inc/TAnalyser.h](#)
- [/home/ychoi/ATOM/alpide/analysis/src/TAnalyser.cpp](#)

7.31 TAPTSDecoder Class Reference

```
#include <TAPTSDecoder.h>
```

Inheritance diagram for TAPTSDecoder:



Public Member Functions

- [TAPTSDecoder](#) (const std::filesystem::path &binaryPath)
- [TAPTSDecoder](#) (const std::string &binaryPath)
- void [decode](#) ()
- std::vector< [TAPTSEvent](#) * > [getData](#) ()

Public Member Functions inherited from [TDecoder](#)

- [TDecoder](#) (const std::filesystem::path &binaryPath)
- [TDecoder](#) (const std::string &binaryPath)
- void [readFile](#) ()
- int [getDataLength](#) ()
- std::vector< uint8_t > & [getBinaryData](#) ()

Private Member Functions

- void [inputEvent](#) ()
- void [preTest](#) ()
- void [rangeTest](#) ()
- void [headerTest](#) ()
- void [missingTest](#) ()
- int [getEventLength](#) ()
- void [postTest](#) ()
- bool [isDone](#) ()

Private Attributes

- std::vector< [TAPTSEvent](#) * > [aptss](#)
- bool [mux_](#) = false
- int [iEvent_](#) = 0
- std::array< int, 16 > [mapping](#)
- std::array< uint8_t, 4 > [expected_header](#) = {0xAA, 0xAA, 0xAA, 0xAA}
- std::array< uint8_t, 4 > [expected_footer](#) = {0xBB, 0xBB, 0xBB, 0xBB}
- int [nFrame_](#)
- int [index_](#) = 0
- int [lenEvent_](#) = 0

7.31.1 Detailed Description

Definition at line 17 of file [TAPTSDecoder.h](#).

7.31.2 Constructor & Destructor Documentation

7.31.2.1 TAPTSDecoder() [1/2]

```
TAPTSDecoder::TAPTSDecoder (
    const std::filesystem::path & binaryPath )
```

Definition at line 7 of file [TAPTSDecoder.cpp](#).

7.31.2.2 TAPTSDecoder() [2/2]

```
TAPTSDecoder::TAPTSDecoder (
    const std::string & binaryPath )
```

Definition at line 8 of file [TAPTSDecoder.cpp](#).

7.31.3 Member Function Documentation

7.31.3.1 decode()

```
void TAPTSDecoder::decode ( )
```

Definition at line 10 of file [TAPTSDecoder.cpp](#).

7.31.3.2 getData()

```
std::vector< TAPTSEvent * > TAPTSDecoder::getData ( )
```

Definition at line 20 of file [TAPTSDecoder.cpp](#).

7.31.3.3 getEventLength()

```
int TAPTSDecoder::getEventLength ( ) [private]
```

Definition at line 97 of file [TAPTSDecoder.cpp](#).

7.31.3.4 headerTest()

```
void TAPTSDecoder::headerTest ( ) [private]
```

Definition at line 85 of file [TAPTSDecoder.cpp](#).

7.31.3.5 inputEvent()

```
void TAPTSDecoder::inputEvent ( ) [private]
```

Definition at line 24 of file [TAPTSDecoder.cpp](#).

7.31.3.6 isDone()

```
bool TAPTSDecoder::isDone ( ) [private]
```

Definition at line 105 of file [TAPTSDecoder.cpp](#).

7.31.3.7 missingTest()

```
void TAPTSDecoder::missingTest ( ) [private]
```

Definition at line 91 of file [TAPTSDecoder.cpp](#).

7.31.3.8 postTest()

```
void TAPTSDecoder::postTest ( ) [private]
```

Definition at line 65 of file [TAPTSDecoder.cpp](#).

7.31.3.9 preTest()

```
void TAPTSDecoder::preTest ( ) [private]
```

Definition at line 71 of file [TAPTSDecoder.cpp](#).

7.31.3.10 rangeTest()

```
void TAPTSDecoder::rangeTest ( ) [private]
```

Definition at line 79 of file [TAPTSDecoder.cpp](#).

7.31.4 Member Data Documentation

7.31.4.1 aptss

```
std::vector<TAPTSEvent*> TAPTSDecoder::aptss [private]
```

Definition at line 19 of file [TAPTSDecoder.h](#).

7.31.4.2 expected_footer

```
std::array<uint8_t, 4> TAPTSDecoder::expected_footer = {0xBB, 0xBB, 0xBB, 0xBB} [private]
```

Definition at line 24 of file [TAPTSDecoder.h](#).

7.31.4.3 expected_header

```
std::array<uint8_t, 4> TAPTSDecoder::expected_header = {0xAA, 0xAA, 0xAA, 0xAA} [private]
```

Definition at line 23 of file [TAPTSDecoder.h](#).

7.31.4.4 iEvent_

```
int TAPTSDecoder::iEvent_ = 0 [private]
```

Definition at line 21 of file [TAPTSDecoder.h](#).

7.31.4.5 index_

```
int TAPTSDecoder::index_ = 0 [private]
```

Definition at line 27 of file [TAPTSDecoder.h](#).

7.31.4.6 lenEvent_

```
int TAPTSDecoder::lenEvent_ = 0 [private]
```

Definition at line 28 of file [TAPTSDecoder.h](#).

7.31.4.7 mapping

```
std::array<int, 16> TAPTSDecoder::mapping [private]
```

Definition at line 22 of file [TAPTSDecoder.h](#).

7.31.4.8 mux_

```
bool TAPTSDecoder::mux_ = false [private]
```

Definition at line 20 of file [TAPTSDecoder.h](#).

7.31.4.9 nFrame_

```
int TAPTSDecoder::nFrame_ [private]
```

Definition at line 26 of file [TAPTSDecoder.h](#).

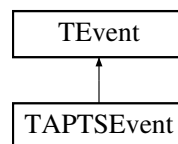
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/apts/inc/TAPTSDecoder.h](#)
- [/home/ychoi/ATOM/apts/src/TAPTSDecoder.cpp](#)

7.32 TAPTSEvent Class Reference

```
#include <TAPTSEvent.h>
```

Inheritance diagram for TAPTSEvent:



Public Member Functions

- [TAPTSEvent](#) ()
- void [setFrame](#) (int frame)
- int [getFrame](#) ()
- std::array< int, 16 > & [getData](#) ()

Public Member Functions inherited from [TEvent](#)

- [TEvent](#) ()
- virtual [~TEvent](#) ()
- void [setEvent](#) (const int event)
- const int [getEvent](#) () const

Private Attributes

- int [iFrame](#)
- std::array< int, 16 > [data](#) {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

7.32.1 Detailed Description

Definition at line 8 of file [TAPTSEvent.h](#).

7.32.2 Constructor & Destructor Documentation

7.32.2.1 TAPTSEvent()

```
TAPTSEvent::TAPTSEvent ( )
```

Definition at line 3 of file [TAPTSEvent.cpp](#).

7.32.3 Member Function Documentation

7.32.3.1 getData()

```
std::array< int, 16 > & TAPTSEvent::getData ( )
```

Definition at line 13 of file [TAPTSEvent.cpp](#).

7.32.3.2 getFrame()

```
int TAPTSEvent::getFrame ( )
```

Definition at line 9 of file [TAPTSEvent.cpp](#).

7.32.3.3 setFrame()

```
void TAPTSEvent::setFrame (
    int frame )
```

Definition at line 5 of file [TAPTSEvent.cpp](#).

7.32.4 Member Data Documentation

7.32.4.1 data

```
std::array<int, 16> TAPTSEvent::data {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} [private]
```

Definition at line 11 of file [TAPTSEvent.h](#).

7.32.4.2 iFrame

```
int TAPTSEvent::iFrame [private]
```

Definition at line 10 of file [TAPTSEvent.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/apts/inc/TAPTSEvent.h](#)
- [/home/ychoi/ATOM/apts/src/TAPTSEvent.cpp](#)

7.33 TCluster Class Reference

```
#include <TCluster.h>
```

Public Types

- enum { `kNotDeleted` = 0x02000000 }

Public Member Functions

- `TCluster` ()
- `TCluster` (int event, int time)
- `TCluster` (const `TCluster` ©)
- `TCluster` & `operator=` (const `TCluster` ©)
- `TCluster` (`TCluster` &&move)
- `TCluster` & `operator=` (`TCluster` &&move)
- `~TCluster` ()
- void `AddPixel` (const std::pair< int, int > &pixel)
- void `AddCluster` (const `TCluster` &cluster)
- bool `isNeighbour` (const std::pair< int, int > &pixel) const
- bool `isNeighbour` (const `TCluster` &cluster) const
- bool `isContain` (const std::pair< int, int > &pixel) const
- bool `isContain` (const `TCluster` &cluster) const
- const int `getDistance` (const std::pair< int, int > &pixel1, const std::pair< int, int > &pixel2) const
- void `calMembers` ()
- void `calMinMax` ()
- void `calCenter` ()
- void `calLongRadius` ()
- void `calSize` ()
- const std::pair< double, double > `getCenter` () const
- const double `getLongRadius` () const
- const int `getSize` () const
- void `setEvent` (const int event)
- void `setTimeStamp` (const int time)
- void `setMinX` (const int minX)
- void `setMinY` (const int minY)
- void `setMaxX` (const int maxX)
- void `setMaxY` (const int maxY)
- const std::vector< std::pair< int, int > > `getPixels` () const
- const int `getEvent` () const
- const int `getTimeStamp` () const
- const int `getMinX` () const
- const int `getMinY` () const
- const int `getMaxX` () const
- const int `getMaxY` () const
- bool `operator==` (const `TCluster` &cluster) const
- bool `operator!=` (const `TCluster` &cluster) const
- bool `IsDestructed` () const
- bool `TestBit` (unsigned int f) const

Private Attributes

- int [mEvent](#)
- int [mTime](#)
- std::vector< std::pair< int, int > > [mPixels](#)
- int [mMinX](#) = 1024
- int [mMinY](#) = 512
- int [mMaxX](#) = 0
- int [mMaxY](#) = 0
- std::pair< double, double > [center](#)
- double [mLongRadius](#)
- int [size](#) = 0
- unsigned int [fBits](#)

7.33.1 Detailed Description

Definition at line 22 of file [TCluster.h](#).

7.33.2 Member Enumeration Documentation**7.33.2.1 anonymous enum**

anonymous enum

Enumerator

kNotDeleted	
-----------------------------	--

Definition at line 97 of file [TCluster.h](#).

7.33.3 Constructor & Destructor Documentation**7.33.3.1 TCluster() [1/4]**

```
TCluster::TCluster ( )
```

Definition at line 4 of file [TCluster.cpp](#).

7.33.3.2 TCluster() [2/4]

```
TCluster::TCluster (
    int event,
    int time )
```

Definition at line 6 of file [TCluster.cpp](#).

7.33.3.3 TCluster() [3/4]

```
TCluster::TCluster (
    const TCluster & copy )
```

Definition at line 8 of file [TCluster.cpp](#).

7.33.3.4 TCluster() [4/4]

```
TCluster::TCluster (
    TCluster && move )
```

Definition at line 24 of file [TCluster.cpp](#).

7.33.3.5 ~TCluster()

```
TCluster::~TCluster ( )
```

Definition at line 37 of file [TCluster.cpp](#).

7.33.4 Member Function Documentation

7.33.4.1 AddCluster()

```
void TCluster::AddCluster (
    const TCluster & cluster )
```

Definition at line 44 of file [TCluster.cpp](#).

7.33.4.2 AddPixel()

```
void TCluster::AddPixel (
    const std::pair< int, int > & pixel )
```

Definition at line 39 of file [TCluster.cpp](#).

7.33.4.3 calCenter()

```
void TCluster::calCenter ( )
```

Definition at line 111 of file [TCluster.cpp](#).

7.33.4.4 calLongRadius()

```
void TCluster::calLongRadius ( )
```

Definition at line 126 of file [TCluster.cpp](#).

7.33.4.5 calMembers()

```
void TCluster::calMembers ( )
```

Definition at line 92 of file [TCluster.cpp](#).

7.33.4.6 calMinMax()

```
void TCluster::calMinMax ( )
```

Definition at line 102 of file [TCluster.cpp](#).

7.33.4.7 calSize()

```
void TCluster::calSize ( )
```

Definition at line 122 of file [TCluster.cpp](#).

7.33.4.8 getCenter()

```
const std::pair< double, double > TCluster::getCenter ( ) const
```

Definition at line 135 of file [TCluster.cpp](#).

7.33.4.9 getDistance()

```
const int TCluster::getDistance (
    const std::pair< int, int > & pixel1,
    const std::pair< int, int > & pixel2 ) const
```

Definition at line 88 of file [TCluster.cpp](#).

7.33.4.10 getEvent()

```
const int TCluster::getEvent ( ) const
```

Definition at line 157 of file [TCluster.cpp](#).

7.33.4.11 getLongRadius()

```
const double TCluster::getLongRadius ( ) const
```

Definition at line 143 of file [TCluster.cpp](#).

7.33.4.12 getMaxX()

```
const int TCluster::getMaxX ( ) const
```

Definition at line 161 of file [TCluster.cpp](#).

7.33.4.13 getMaxY()

```
const int TCluster::getMaxY ( ) const
```

Definition at line 162 of file [TCluster.cpp](#).

7.33.4.14 getMinX()

```
const int TCluster::getMinX ( ) const
```

Definition at line 159 of file [TCluster.cpp](#).

7.33.4.15 getMinY()

```
const int TCluster::getMinY ( ) const
```

Definition at line 160 of file [TCluster.cpp](#).

7.33.4.16 getPixels()

```
const std::vector< std::pair< int, int > > TCluster::getPixels ( ) const
```

Definition at line 156 of file [TCluster.cpp](#).

7.33.4.17 getSize()

```
const int TCluster::getSize ( ) const
```

Definition at line 139 of file [TCluster.cpp](#).

7.33.4.18 getTimeStamp()

```
const int TCluster::getTimeStamp ( ) const
```

Definition at line 158 of file [TCluster.cpp](#).

7.33.4.19 isContain() [1/2]

```
bool TCluster::isContain (
    const std::pair< int, int > & pixel ) const
```

Definition at line 71 of file [TCluster.cpp](#).

7.33.4.20 isContain() [2/2]

```
bool TCluster::isContain (
    const TCluster & cluster ) const
```

Definition at line 79 of file [TCluster.cpp](#).

7.33.4.21 IsDestructed()

```
bool TCluster::IsDestructed ( ) const [inline]
```

Definition at line 100 of file [TCluster.h](#).

7.33.4.22 isNeighbour() [1/2]

```
bool TCluster::isNeighbour (
    const std::pair< int, int > & pixel ) const
```

Definition at line 51 of file [TCluster.cpp](#).

7.33.4.23 isNeighbour() [2/2]

```
bool TCluster::isNeighbour (
    const TCluster & cluster ) const
```

Definition at line 62 of file [TCluster.cpp](#).

7.33.4.24 operator"!="()

```
bool TCluster::operator!= (
    const TCluster & cluster ) const
```

Definition at line 17 of file [TClusterOperator.cpp](#).

7.33.4.25 operator=() [1/2]

```
TCluster & TCluster::operator= (
    const TCluster & copy )
```

Definition at line 12 of file [TCluster.cpp](#).

7.33.4.26 operator=() [2/2]

```
TCluster & TCluster::operator= (
    TCluster && move )
```

Definition at line 26 of file [TCluster.cpp](#).

7.33.4.27 operator==()

```
bool TCluster::operator== (
    const TCluster & cluster ) const
```

Definition at line 3 of file [TClusterOperator.cpp](#).

7.33.4.28 setEvent()

```
void TCluster::setEvent (
    const int event )
```

Definition at line 148 of file [TCluster.cpp](#).

7.33.4.29 setMaxX()

```
void TCluster::setMaxX (
    const int maxX )
```

Definition at line 152 of file [TCluster.cpp](#).

7.33.4.30 setMaxY()

```
void TCluster::setMaxY (
    const int maxY )
```

Definition at line 153 of file [TCluster.cpp](#).

7.33.4.31 setMinX()

```
void TCluster::setMinX (
    const int minX )
```

Definition at line 150 of file [TCluster.cpp](#).

7.33.4.32 setMinY()

```
void TCluster::setMinY (
    const int minY )
```

Definition at line 151 of file [TCluster.cpp](#).

7.33.4.33 setTimeStamp()

```
void TCluster::setTimeStamp (
    const int time )
```

Definition at line 149 of file [TCluster.cpp](#).

7.33.4.34 TestBit()

```
bool TCluster::TestBit (
    unsigned int f ) const [inline]
```

Definition at line 101 of file [TCluster.h](#).

7.33.5 Member Data Documentation

7.33.5.1 center

```
std::pair<double, double> TCluster::center [private]
```

cluster centre value. The average of x and y values.

Definition at line 30 of file [TCluster.h](#).

7.33.5.2 fBits

```
unsigned int TCluster::fBits [private]
```

Definition at line 95 of file [TCluster.h](#).

7.33.5.3 mEvent

```
int TCluster::mEvent [private]
```

Number of event to which this cluster belongs

Definition at line 24 of file [TCluster.h](#).

7.33.5.4 mLongRadius

```
double TCluster::mLongRadius [private]
```

Definition at line 31 of file [TCluster.h](#).

7.33.5.5 mMaxX

```
int TCluster::mMaxX = 0 [private]
```

Definition at line 29 of file [TCluster.h](#).

7.33.5.6 mMaxY

```
int TCluster::mMaxY = 0 [private]
```

Minimum pixel x and y value in cluster

Definition at line 29 of file [TCluster.h](#).

7.33.5.7 mMinX

```
int TCluster::mMinX = 1024 [private]
```

Definition at line 28 of file [TCluster.h](#).

7.33.5.8 mMinY

```
int TCluster::mMinY = 512 [private]
```

Maximum pixel x and y value in cluster

Definition at line 28 of file [TCluster.h](#).

7.33.5.9 mPixels

```
std::vector<std::pair<int, int> > TCluster::mPixels [private]
```

The bundle of pixels which are composed to this cluster

Definition at line 26 of file [TCluster.h](#).

7.33.5.10 mTime

```
int TCluster::mTime [private]
```

Time stamp of event

Definition at line 25 of file [TCluster.h](#).

7.33.5.11 size

```
int TCluster::size = 0 [private]
```

The number of pixels in cluster

Definition at line 34 of file [TCluster.h](#).

The documentation for this class was generated from the following files:

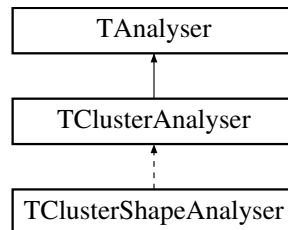
- [/home/ychoi/ATOM/alpide/cluster/inc/TCluster.h](#)
- [/home/ychoi/ATOM/alpide/cluster/src/TCluster.cpp](#)
- [/home/ychoi/ATOM/alpide/cluster/src/TClusterOperator.cpp](#)

7.34 TClusterAnalyser Class Reference

Communicating execute file for controlling cluster research.

```
#include <TClusterAnalyser.h>
```

Inheritance diagram for TClusterAnalyser:



Public Types

- enum { [kNotDeleted](#) = 0x02000000 }

Public Types inherited from [TAnalyser](#)

- enum { [kNotDeleted](#) = 0x02000000 }

Public Member Functions

- [TClusterAnalyser](#) ()=default
- [TClusterAnalyser](#) (const [TAnalyser](#) &analyser)
Construct a new [TClusterAnalyser::TClusterAnalyser](#) object.
- [TClusterAnalyser](#) (const [TClusterAnalyser](#) ©)
- [~TClusterAnalyser](#) ()
Destroy the [TClusterAnalyser::TClusterAnalyser](#) object.
- TH2D * [getClusterPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- TH1D * [getClustersizePlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- void [setClusterDataWithShape](#) (const std::vector< int > &clusterSizeRange)
- void [saveClustermap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveClustersize](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveHitmapByClustersize](#) (const [CppConfigDictionary](#) &config)
- bool [IsDestructed](#) () const
- bool [TestBit](#) (unsigned int f) const

Public Member Functions inherited from [TAnalyser](#)

- [TAnalyser](#) ()=default
- [TAnalyser](#) (TFile *inputFile, std::unordered_map< std::string, [TExperimentData](#) * > expData)
Construct a new [TAnalyser::TAnalyser](#) object.
- [~TAnalyser](#) ()
Destroy the [TAnalyser::TAnalyser](#) object.
- TTree * [openTree](#) (std::string treeName)
open trees
- void [storeEvents](#) ([CppConfigDictionary](#) settingConfig)
Extract event from TTree.
- void [doMasking](#) (int mMaskOver)
- void [openOutputGraphFile](#) (std::string_view fileName)
Open ROOT file to save results graph.
- void [openDirectory](#) (std::string_view typeName)
Set the new directory to store graph root file.
- void [setExpSettingLegend](#) ([CppConfigDictionary](#) settingConfig)
- void [doDivideBySize](#) (std::string_view typeName)
- [TExperimentData](#) * [getAnEventSet](#) (std::string_view typeName) const
- TH2D * [getHitPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TALPIDEEvent](#) * > &events)
- void [saveHitmap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- TH2D * [getClusterPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
Generalized function to draw clustermap.
- TH1D * [getClustersizePlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- void [setClusterDataWithShape](#) (const std::vector< int > &clusterSizeRange)
- void [saveClustermap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveClustersize](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- std::vector< int > [getClusterSizeRange](#) (const [CppConfigDictionary](#) &privateProperty)
- void [doShaping](#) (std::string_view typeName, const std::vector< int > &clusterSizeRange)
Store clusters to objects of [TClusterShape](#) for extracting shape informations.
- void [saveIndividualShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
Drawing individual shapes.
- void [saveSameSizeInfos](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
Drawing information for classifying multi-cluster.
- void [saveSameSizeShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveTotalShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveSameSizeShapeEntry](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveTotalShapeEntry](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- bool [IsDestroyed](#) () const
- bool [TestBit](#) (UInt_t f) const

Private Attributes

- unsigned int [fBits](#)

Additional Inherited Members

Protected Attributes inherited from [TAnalyser](#)

- TFile * [mOutputFile](#) = nullptr
- bool [mIsOutputGraph](#) = false
- TPaveText * [mExpSettingLegend](#)
- std::unordered_map< std::string, TH2D * > [mHitmaps](#)
- std::unordered_map< std::string, TH2D * > [mClustermaps](#)
- std::unordered_map< std::string, TH1D * > [mClustersizes](#)
- std::unordered_map< std::string, std::unordered_map< int, std::vector< [TCluster](#) * > > > [mClusterDataWithShape](#)
- std::unordered_map< std::string, TDirectory * > [mDirectorySet](#)
- std::unordered_map< std::string, [TExperimentData](#) * > [mExpData](#)
- std::unordered_map< std::string, [TClusterDivideData](#) * > [mDivideData](#)
- std::unordered_map< std::string, std::vector< [TClusterShape](#) * > > [mClusterShapeSet](#)
- std::unordered_map< std::string, int > [mNTotalShapeSet](#)
- std::unordered_map< std::string, int > [mMaxModeSet](#)

7.34.1 Detailed Description

Communicating execute file for controlling cluster research.

It takes ROOT file which is result of experiment. The analysis data about raw data, clusterized data and masked data are stored as data member. From these data, the plots for informations about hitmap, clustermap, size and shapes are drawn.

Warning

Bug

Todo Add Legend about experiment setting
Make more plots about cluster information

Definition at line 52 of file [TClusterAnalyser.h](#).

7.34.2 Member Enumeration Documentation

7.34.2.1 anonymous enum

anonymous enum

Enumerator

kNotDeleted	
-----------------------------	--

Definition at line 75 of file [TClusterAnalyser.h](#).

7.34.3 Constructor & Destructor Documentation

7.34.3.1 TClusterAnalyser() [1/3]

```
TClusterAnalyser::TClusterAnalyser ( ) [default]
```

7.34.3.2 TClusterAnalyser() [2/3]

```
TClusterAnalyser::TClusterAnalyser (
    const TAnalyser & analyser )
```

Construct a new [TClusterAnalyser::TClusterAnalyser](#) object.

Parameters

<i>analyser</i>	
-----------------	--

Definition at line 9 of file [TClusterAnalyser.cpp](#).

7.34.3.3 TClusterAnalyser() [3/3]

```
TClusterAnalyser::TClusterAnalyser (
    const TClusterAnalyser & copy )
```

Definition at line 13 of file [TClusterAnalyser.cpp](#).

7.34.3.4 ~TClusterAnalyser()

```
TClusterAnalyser::~TClusterAnalyser ( )
```

Destroy the [TClusterAnalyser::TClusterAnalyser](#) object.

Definition at line 20 of file [TClusterAnalyser.cpp](#).

7.34.4 Member Function Documentation

7.34.4.1 getClusterPlot()

```
TH2D * TClusterAnalyser::getClusterPlot (
    const CppConfigDictionary & config,
    const std::vector< TCluster * > & clusters )
```

7.34.4.2 getClustersizePlot()

```
TH1D * TClusterAnalyser::getClustersizePlot (
    const CppConfigDictionary & config,
    const std::vector< TCluster * > & clusters )
```

7.34.4.3 IsDestructed()

```
bool TClusterAnalyser::IsDestructed ( ) const [inline]
```

Definition at line 78 of file [TClusterAnalyser.h](#).

7.34.4.4 saveClustermap()

```
void TClusterAnalyser::saveClustermap (
    std::string typeName,
    const CppConfigDictionary & config )
```

7.34.4.5 saveClustersize()

```
void TClusterAnalyser::saveClustersize (
    std::string typeName,
    const CppConfigDictionary & config )
```

7.34.4.6 saveHitmapByClustersize()

```
void TClusterAnalyser::saveHitmapByClustersize (
    const CppConfigDictionary & config )
```

7.34.4.7 setClusterDataWithShape()

```
void TClusterAnalyser::setClusterDataWithShape (
    const std::vector< int > & clusterSizeRange )
```

7.34.4.8 TestBit()

```
bool TClusterAnalyser::TestBit (
    unsigned int f ) const [inline]
```

Definition at line 79 of file [TClusterAnalyser.h](#).

7.34.5 Member Data Documentation

7.34.5.1 fBits

```
unsigned int TClusterAnalyser::fBits [private]
```

Definition at line 73 of file [TClusterAnalyser.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/trashcan/TClusterAnalyser.h](#)
- [/home/ychoi/ATOM/trashcan/TClusterAnalyser.cpp](#)

7.35 TClusterDivideData Class Reference

```
#include <TClusterDivideData.h>
```

Public Member Functions

- [TClusterDivideData](#) (const std::vector< [TCluster](#) * > &clusters)
- [TClusterDivideData](#) (const [TClusterDivideData](#) ©)
- const std::vector< [TCluster](#) * > & [getClusterOfSize](#) (const int clusterSize)

Private Attributes

- std::unordered_map< int, std::vector< [TCluster](#) * > > [mClusterData](#)

7.35.1 Detailed Description

Definition at line 14 of file [TClusterDivideData.h](#).

7.35.2 Constructor & Destructor Documentation

7.35.2.1 TClusterDivideData() [1/2]

```
TClusterDivideData::TClusterDivideData (  
    const std::vector< TCluster * > & clusters )
```

Definition at line 4 of file [TClusterDivideData.cpp](#).

7.35.2.2 TClusterDivideData() [2/2]

```
TClusterDivideData::TClusterDivideData (  
    const TClusterDivideData & copy )
```

Definition at line 17 of file [TClusterDivideData.cpp](#).

7.35.3 Member Function Documentation

7.35.3.1 getClusterOfSize()

```
const std::vector< TCluster * > & TClusterDivideData::getClusterOfSize (  
    const int clusterSize )
```

Definition at line 20 of file [TClusterDivideData.cpp](#).

7.35.4 Member Data Documentation

7.35.4.1 mClusterData

```
std::unordered_map<int, std::vector<TCluster*> > TClusterDivideData::mClusterData [private]
```

Definition at line 16 of file [TClusterDivideData.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/cluster/inc/TClusterDivideData.h](#)
- [/home/ychoi/ATOM/alpide/cluster/src/TClusterDivideData.cpp](#)

7.36 TClusterization Class Reference

Class of tools for clusterizing events for single event.

```
#include <TClusterization.h>
```

Public Member Functions

- [TClusterization](#) ()=delete
- [TClusterization](#) (const std::vector< [TALPIDEEvent](#) * > &events)
- [~TClusterization](#) ()
- void [removeConsecutionPixels](#) ()
- void [addNewCluster](#) (const std::pair< int, int > &pixel, std::vector< [TCluster](#) * > &clusters, int iEvent, int iTime)
- void [removeIndependentCluster](#) (const std::pair< int, int > &pixel, std::vector< [TCluster](#) * > &clusters)
- bool [clusterLitmusTest](#) (const std::pair< int, int > &pixel, std::vector< [TCluster](#) * > &clusters)
- void [clusterize](#) ()
- const std::vector< [TCluster](#) * > & [getClusters](#) () const

Private Attributes

- std::vector< [TALPIDEEvent](#) * > [mEvents](#)
- std::vector< [TCluster](#) * > [mClusters](#)

7.36.1 Detailed Description

Class of tools for clusterizing events for single event.

It takes a role to clusterize an event.

Definition at line 32 of file [TClusterization.h](#).

7.36.2 Constructor & Destructor Documentation

7.36.2.1 TClusterization() [1/2]

```
TClusterization::TClusterization ( ) [delete]
```

7.36.2.2 TClusterization() [2/2]

```
TClusterization::TClusterization (
    const std::vector< TALPIDEEvent * > & events )
```

Definition at line 4 of file [TClusterization.cpp](#).

7.36.2.3 ~TClusterization()

```
TClusterization::~~TClusterization ( )
```

Definition at line 10 of file [TClusterization.cpp](#).

7.36.3 Member Function Documentation

7.36.3.1 addNewCluster()

```
void TClusterization::addNewCluster (
    const std::pair< int, int > & pixel,
    std::vector< TCluster * > & clusters,
    int iEvent,
    int iTime )
```

Definition at line 29 of file [TClusterization.cpp](#).

7.36.3.2 clusterize()

```
void TClusterization::clusterize ( )
```

Definition at line 70 of file [TClusterization.cpp](#).

7.36.3.3 clusterLitmusTest()

```
bool TClusterization::clusterLitmusTest (
    const std::pair< int, int > & pixel,
    std::vector< TCluster * > & clusters )
```

Definition at line 48 of file [TClusterization.cpp](#).

7.36.3.4 getClusters()

```
const std::vector< TCluster * > & TClusterization::getClusters ( ) const
```

Definition at line 104 of file [TClusterization.cpp](#).

7.36.3.5 removeConsecutionPixels()

```
void TClusterization::removeConsecutionPixels ( )
```

Definition at line 12 of file [TClusterization.cpp](#).

7.36.3.6 removeIndependentCluster()

```
void TClusterization::removeIndependentCluster (
    const std::pair< int, int > & pixel,
    std::vector< TCluster * > & clusters )
```

Definition at line 35 of file [TClusterization.cpp](#).

7.36.4 Member Data Documentation

7.36.4.1 mClusters

```
std::vector<TCluster*> TClusterization::mClusters [private]
```

Definition at line 34 of file [TClusterization.h](#).

7.36.4.2 mEvents

```
std::vector<TALPIDEEvent*> TClusterization::mEvents [private]
```

Definition at line 33 of file [TClusterization.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/cluster/inc/TClusterization.h](#)
- [/home/ychoi/ATOM/alpide/cluster/src/TClusterization.cpp](#)

7.37 TClusterShape Class Reference

Class for extracting cluster shape information with same cluster size.

```
#include <TClusterShape.h>
```

Public Types

- enum { `kNotDeleted` = 0x02000000 }

Public Member Functions

- `TClusterShape` ()
Construct a new `TClusterShape::TClusterShape` object.
- `TClusterShape` (const int clusterSize, const std::vector< `TCluster` * > &clusters)
- `TClusterShape` (const std::vector< `TCluster` * > clusters)
- `~TClusterShape` ()
- void `identifyShapes` ()
Specificate the clusters according to their shapes.
- void `sortShapes` (bool descend=true)
Sorting shapes according to the entry.
- void `calClusterInfo` (`TShapeInfo` &shapeInfo, `TCluster` *cluster)
- `TMatrix2D`< int > * `clusterMatrix` (const `TCluster` *cluster)
- `TH2I` * `clusterMap` (const `TMatrix2D`< int > *clusterMatrix)
Drawing and saving shape image.
- const std::vector< `TShapeInfo` > & `getClusterShapeInfos` () const
Getter of `mClusterShapeInfos` member.
- void `setClusterSize` (const int ClusterSize)
Setter of `mClusterSize` member.
- const int `getClusterSize` () const
Getter of `mClusterSize` member.
- bool `IsDestructed` () const
- bool `TestBit` (unsigned int f) const

Private Attributes

- int `mClusterSize`
- std::vector< `TCluster` * > `mClusterOriginSet`
- std::vector< `TShapeInfo` > `mClusterShapeInfos`
- unsigned int `fBits`

7.37.1 Detailed Description

Class for extracting cluster shape information with same cluster size.

It collects clusters having same cluster size. And specificate the clusters according to their shapes. The image and other informations can be extract by `TShapeInfo` struct.

Warning

Bug

Todo It can be modified if needed.

Definition at line 58 of file `TClusterShape.h`.

7.37.2 Member Enumeration Documentation

7.37.2.1 anonymous enum

anonymous enum

Enumerator

kNotDeleted	
-------------	--

Definition at line 83 of file [TClusterShape.h](#).

7.37.3 Constructor & Destructor Documentation

7.37.3.1 TClusterShape() [1/3]

```
TClusterShape::TClusterShape ( )
```

Construct a new [TClusterShape::TClusterShape](#) object.

It takes cluster set and cluster size for analysing here. The role of this construct is extracting cluster from cluster set according to cluster size.

Parameters

<i>clusters</i>	
<i>clusterSize</i>	

Warning

Bug

Todo

See also

[TCluster::getSize\(\)](#)

Definition at line 17 of file [TClusterShape.cpp](#).

7.37.3.2 TClusterShape() [2/3]

```
TClusterShape::TClusterShape (
    const int clusterSize,
    const std::vector< TCluster * > & clusters )
```

Definition at line 19 of file [TClusterShape.cpp](#).

7.37.3.3 TClusterShape() [3/3]

```
TClusterShape::TClusterShape (
    const std::vector< TCluster * > clusters )
```

Definition at line 31 of file [TClusterShape.cpp](#).

7.37.3.4 ~TClusterShape()

```
TClusterShape::~TClusterShape ( )
```

Definition at line 35 of file [TClusterShape.cpp](#).

7.37.4 Member Function Documentation

7.37.4.1 calClusterInfo()

```
void TClusterShape::calClusterInfo (
    TShapeInfo & shapeInfo,
    TCluster * cluster )
```

Definition at line 164 of file [TClusterShape.cpp](#).

7.37.4.2 clusterMap()

```
TH2I * TClusterShape::clusterMap (
    const TMatrix2D< int > * clusterMatrix )
```

Drawing and saving shape image.

It visualizes shape informations form matrix form. TImage class is used for saving plots. The extra pixel on border of plot is drawn for readability.

Parameters

<i>clusterMatrix</i>	
----------------------	--

Returns

TImage*

Warning

Bug The canvases and histograms have same name with each other.

Todo Avoiding same name problem.

See also

Definition at line 134 of file [TClusterShape.cpp](#).

7.37.4.3 clusterMatrix()

```
TMatrix2D< int > * TClusterShape::clusterMatrix (
    const TCluster * cluster )
```

Definition at line 184 of file [TClusterShape.cpp](#).

7.37.4.4 getClusterShapeInfos()

```
const std::vector< TShapeInfo > & TClusterShape::getClusterShapeInfos ( ) const
```

Getter of mClusterShapeInfos member.

Returns

const std::vector<TShapeInfo>&

Definition at line 206 of file [TClusterShape.cpp](#).

7.37.4.5 getClusterSize()

```
const int TClusterShape::getClusterSize ( ) const
```

Getter of mClusterSize member.

Returns

const int

Definition at line 213 of file [TClusterShape.cpp](#).

7.37.4.6 identifyShapes()

```
void TClusterShape::identifyShapes ( )
```

Specify the clusters according to their shapes.

This function checks homeomorphism of cluster and store the shape informations. By the pixel structure, there are 8 types for homeomorphism.

Returns

void

Warning

Bug

Todo

See also

[TCluster::getShape\(\)](#), [TMatrix2D::hasHomeomorphism](#)

Definition at line 53 of file [TClusterShape.cpp](#).

7.37.4.7 IsDestructed()

```
bool TClusterShape::IsDestructed ( ) const [inline]
```

Definition at line 86 of file [TClusterShape.h](#).

7.37.4.8 setClusterSize()

```
void TClusterShape::setClusterSize (
    const int clusterSize )
```

Setter of `mClusterSize` member.

Parameters

<i>clusterSize</i>	
--------------------	--

Returns

void

Definition at line 199 of file [TClusterShape.cpp](#).

7.37.4.9 sortShapes()

```
void TClusterShape::sortShapes (
    bool descend = true )
```

Sorting shapes according to the entry.

It sorts cluster information structs with same shapes. The number of clusters having same shapes is saved in cluster information. It is used for sort criteria.

Parameters

<i>descend</i>	
----------------	--

Returns

void

Warning

Bug

Todo More strict criteria is needed. If they have same entry, then it cannot be sorted and saved by coming order.

See also

Definition at line 107 of file [TClusterShape.cpp](#).

7.37.4.10 TestBit()

```
bool TClusterShape::TestBit (
    unsigned int f ) const [inline]
```

Definition at line 87 of file [TClusterShape.h](#).

7.37.5 Member Data Documentation

7.37.5.1 fBits

```
unsigned int TClusterShape::fBits [private]
```

Definition at line 81 of file [TClusterShape.h](#).

7.37.5.2 mClusterOriginSet

```
std::vector<TCluster\*> TClusterShape::mClusterOriginSet [private]
```

Definition at line 61 of file [TClusterShape.h](#).

7.37.5.3 mClusterShapeInfos

```
std::vector<TShapeInfo> TClusterShape::mClusterShapeInfos [private]
```

The set of cluster shape informations

Definition at line 62 of file [TClusterShape.h](#).

7.37.5.4 mClusterSize

```
int TClusterShape::mClusterSize [private]
```

The cluster size of this shape

Definition at line 60 of file [TClusterShape.h](#).

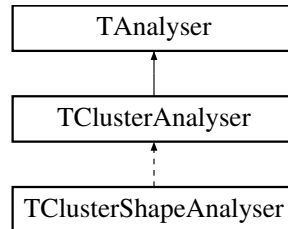
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/cluster/inc/TClusterShape.h](#)
- [/home/ychoi/ATOM/alpide/cluster/src/TClusterShape.cpp](#)

7.38 TClusterShapeAnalyser Class Reference

```
#include <TClusterShapeAnalyser.h>
```

Inheritance diagram for TClusterShapeAnalyser:



Public Types

- enum { `kNotDeleted` = 0x02000000 }

Public Member Functions

- `TClusterShapeAnalyser` (const `TClusterAnalyser` &analyser)
Construct a new `TClusterShapeAnalyser::TClusterShapeAnalyser` object.
- `~TClusterShapeAnalyser` ()
Destroy the `TClusterShapeAnalyser::TClusterShapeAnalyser` object.
- void `doShaping` (std::string_view typeName, const std::vector< int > &clusterSizeRange)
Store clusters to objects of `TClusterShape` for extracting shape informations.
- void `saveIndividualShapes` (std::string_view typeName, const `CppConfigDictionary` config)
Drawing individual shapes.
- void `saveSameSizeInfos` (std::string_view typeName, const `CppConfigDictionary` config)
Drawing information for classifying multi-cluster.
- void `saveSameSizeShapes` (std::string_view typeName, const `CppConfigDictionary` config)
- void `saveTotalShapes` (std::string_view typeName, const `CppConfigDictionary` config)
- void `saveSameSizeShapeEntry` (std::string_view typeName, const `CppConfigDictionary` config)
- void `saveTotalShapeEntry` (std::string_view typeName, const `CppConfigDictionary` config)
- bool `IsDestroyed` () const
- bool `TestBit` (unsigned int f) const

Private Attributes

- std::unordered_map< std::string, std::vector< `TClusterShape` * > > `mClusterShapeSet`
- std::unordered_map< std::string, int > `mNTotalShapeSet`
- std::unordered_map< std::string, int > `mMaxModeSet`
- unsigned int `fBits`

Additional Inherited Members

Protected Types inherited from `TClusterAnalyser`

- enum { `kNotDeleted` = 0x02000000 }

Protected Types inherited from [TAnalyser](#)

- enum { [kNotDeleted](#) = 0x02000000 }

Protected Member Functions inherited from [TClusterAnalyser](#)

- [TClusterAnalyser](#) ()=default
- [TClusterAnalyser](#) (const [TAnalyser](#) &analyser)
Construct a new [TClusterAnalyser::TClusterAnalyser](#) object.
- [TClusterAnalyser](#) (const [TClusterAnalyser](#) ©)
- [~TClusterAnalyser](#) ()
Destroy the [TClusterAnalyser::TClusterAnalyser](#) object.
- TH2D * [getClusterPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- TH1D * [getClustersizePlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- void [setClusterDataWithShape](#) (const std::vector< int > &clusterSizeRange)
- void [saveClustermap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveClustersize](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveHitmapByClustersize](#) (const [CppConfigDictionary](#) &config)
- bool [IsDestructed](#) () const
- bool [TestBit](#) (unsigned int f) const

Protected Member Functions inherited from [TAnalyser](#)

- [TAnalyser](#) ()=default
- [TAnalyser](#) (TFile *inputFile, std::unordered_map< std::string, [TExperimentData](#) * > expData)
Construct a new [TAnalyser::TAnalyser](#) object.
- [~TAnalyser](#) ()
Destroy the [TAnalyser::TAnalyser](#) object.
- TTree * [openTree](#) (std::string treeName)
open trees
- void [storeEvents](#) ([CppConfigDictionary](#) settingConfig)
Extract event from TTree.
- void [doMasking](#) (int mMaskOver)
- void [openOutputGraphFile](#) (std::string_view fileName)
Open ROOT file to save results graph.
- void [openDirectory](#) (std::string_view typeName)
Set the new directory to store graph root file.
- void [setExpSettingLegend](#) ([CppConfigDictionary](#) settingConfig)
- void [doDivideBySize](#) (std::string_view typeName)
- [TExperimentData](#) * [getAnEventSet](#) (std::string_view typeName) const
- TH2D * [getHitPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TALPIDEEvent](#) * > &events)
- void [saveHitmap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- TH2D * [getClusterPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
Generalized function to draw clustermap.
- TH1D * [getClustersizePlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- void [setClusterDataWithShape](#) (const std::vector< int > &clusterSizeRange)
- void [saveClustermap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveClustersize](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- std::vector< int > [getClusterSizeRange](#) (const [CppConfigDictionary](#) &privateProperty)
- void [doShaping](#) (std::string_view typeName, const std::vector< int > &clusterSizeRange)
Store clusters to objects of [TClusterShape](#) for extracting shape informations.

- void [saveIndividualShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
Drawing individual shapes.
- void [saveSameSizeInfos](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
Drawing information for classifying multi-cluster.
- void [saveSameSizeShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveTotalShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveSameSizeShapeEntry](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveTotalShapeEntry](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- bool [IsDestructed](#) () const
- bool [TestBit](#) (UInt_t f) const

Protected Attributes inherited from [TAnalyser](#)

- TFile * [mOutputFile](#) = nullptr
- bool [mIsOutputGraph](#) = false
- TPaveText * [mExpSettingLegend](#)
- std::unordered_map< std::string, TH2D * > [mHitmaps](#)
- std::unordered_map< std::string, TH2D * > [mClustermaps](#)
- std::unordered_map< std::string, TH1D * > [mClustersizes](#)
- std::unordered_map< std::string, std::unordered_map< int, std::vector< [TCluster](#) * > > > [mClusterDataWithShape](#)
- std::unordered_map< std::string, TDirectory * > [mDirectorySet](#)
- std::unordered_map< std::string, [TExperimentData](#) * > [mExpData](#)
- std::unordered_map< std::string, [TClusterDivideData](#) * > [mDivideData](#)
- std::unordered_map< std::string, std::vector< [TClusterShape](#) * > > [mClusterShapeSet](#)
- std::unordered_map< std::string, int > [mNTotalShapeSet](#)
- std::unordered_map< std::string, int > [mMaxModeSet](#)

7.38.1 Detailed Description

Definition at line 43 of file [TClusterShapeAnalyser.h](#).

7.38.2 Member Enumeration Documentation

7.38.2.1 anonymous enum

anonymous enum

Enumerator

kNotDeleted	
-----------------------------	--

Definition at line 62 of file [TClusterShapeAnalyser.h](#).

7.38.3 Constructor & Destructor Documentation

7.38.3.1 TClusterShapeAnalyser()

```
TClusterShapeAnalyser::TClusterShapeAnalyser (
    const TClusterAnalyser & analyser )
```

Construct a new [TClusterShapeAnalyser::TClusterShapeAnalyser](#) object.

Parameters

<i>analyser</i>	
-----------------	--

Definition at line 19 of file [TClusterShapeAnalyser.cpp](#).

7.38.3.2 ~TClusterShapeAnalyser()

```
TClusterShapeAnalyser::~~TClusterShapeAnalyser ( )
```

Destroy the [TClusterShapeAnalyser::TClusterShapeAnalyser](#) object.

Definition at line 28 of file [TClusterShapeAnalyser.cpp](#).

7.38.4 Member Function Documentation

7.38.4.1 doShaping()

```
void TClusterShapeAnalyser::doShaping (
    std::string_view typeName,
    const std::vector< int > & clusterSizeRange )
```

Store clusters to objects of [TClusterShape](#) for extracting shape informations.

Parameters

<i>typeName</i>	
<i>clusterSizeRange</i>	

Definition at line 47 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.2 IsDestroyed()

```
bool TClusterShapeAnalyser::IsDestroyed ( ) const [inline]
```

Definition at line 65 of file [TClusterShapeAnalyser.h](#).

7.38.4.3 saveIndividualShapes()

```
void TClusterShapeAnalyser::saveIndividualShapes (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Drawing individual shapes.

Parameters

<i>typeName</i>	
<i>config</i>	

Definition at line 80 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.4 saveSameSizeInfos()

```
void TClusterShapeAnalyser::saveSameSizeInfos (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Drawing information for classifying multi-cluster.

Parameters

<i>typeName</i>	
<i>config</i>	

Definition at line 234 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.5 saveSameSizeShapeEntry()

```
void TClusterShapeAnalyser::saveSameSizeShapeEntry (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 595 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.6 saveSameSizeShapes()

```
void TClusterShapeAnalyser::saveSameSizeShapes (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 368 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.7 saveTotalShapeEntry()

```
void TClusterShapeAnalyser::saveTotalShapeEntry (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 635 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.8 saveTotalShapes()

```
void TClusterShapeAnalyser::saveTotalShapes (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 483 of file [TClusterShapeAnalyser.cpp](#).

7.38.4.9 TestBit()

```
bool TClusterShapeAnalyser::TestBit (
    unsigned int f ) const [inline]
```

Definition at line 66 of file [TClusterShapeAnalyser.h](#).

7.38.5 Member Data Documentation

7.38.5.1 fBits

```
unsigned int TClusterShapeAnalyser::fBits [private]
```

Definition at line 60 of file [TClusterShapeAnalyser.h](#).

7.38.5.2 mClusterShapeSet

```
std::unordered_map<std::string, std::vector<TClusterShape*> > TClusterShapeAnalyser::mClusterShapeSet [private]
```

Definition at line 45 of file [TClusterShapeAnalyser.h](#).

7.38.5.3 mMaxModeSet

```
std::unordered_map<std::string, int> TClusterShapeAnalyser::mMaxModeSet [private]
```

Definition at line 47 of file [TClusterShapeAnalyser.h](#).

7.38.5.4 mNTotalShapeSet

```
std::unordered_map<std::string, int> TClusterShapeAnalyser::mNTotalShapeSet [private]
```

Definition at line 46 of file [TClusterShapeAnalyser.h](#).

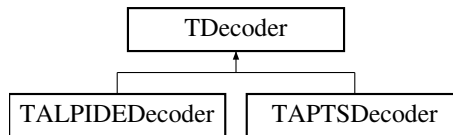
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/trashcan/TClusterShapeAnalyser.h](#)
- [/home/ychoi/ATOM/trashcan/TClusterShapeAnalyser.cpp](#)

7.39 TDecoder Class Reference

```
#include <TDecoder.h>
```

Inheritance diagram for TDecoder:



Public Member Functions

- [TDecoder](#) (const std::filesystem::path &binaryPath)
- [TDecoder](#) (const std::string &binaryPath)
- void [readFile](#) ()
- int [getDataLength](#) ()
- std::vector< uint8_t > & [getBinaryData](#) ()

Private Attributes

- std::ifstream [binaryFile_](#)
- int [dataLength_](#)
- std::vector< uint8_t > [binaryData_](#)

7.39.1 Detailed Description

Definition at line 13 of file [TDecoder.h](#).

7.39.2 Constructor & Destructor Documentation

7.39.2.1 TDecoder() [1/2]

```
TDecoder::TDecoder (
    const std::filesystem::path & binaryPath )
```

Definition at line 3 of file [TDecoder.cpp](#).

7.39.2.2 TDecoder() [2/2]

```
TDecoder::TDecoder (
    const std::string & binaryPath )
```

Definition at line 10 of file [TDecoder.cpp](#).

7.39.3 Member Function Documentation

7.39.3.1 getBinaryData()

```
std::vector< uint8_t > & TDecoder::getBinaryData ( )
```

Definition at line 29 of file [TDecoder.cpp](#).

7.39.3.2 getDataLength()

```
int TDecoder::getDataLength ( )
```

Definition at line 25 of file [TDecoder.cpp](#).

7.39.3.3 readFile()

```
void TDecoder::readFile ( )
```

Definition at line 17 of file [TDecoder.cpp](#).

7.39.4 Member Data Documentation

7.39.4.1 binaryData_

```
std::vector<uint8_t> TDecoder::binaryData_ [private]
```

Definition at line 17 of file [TDecoder.h](#).

7.39.4.2 binaryFile_

```
std::ifstream TDecoder::binaryFile_ [private]
```

Definition at line 15 of file [TDecoder.h](#).

7.39.4.3 dataLength_

```
int TDecoder::dataLength_ [private]
```

Definition at line 16 of file [TDecoder.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/chip/inc/TDecoder.h](#)
- [/home/ychoi/ATOM/chip/src/TDecoder.cpp](#)

7.40 TDetector Struct Reference

```
#include <TEntrySimulation.h>
```

Public Member Functions

- [bool isBelong](#) ([double x](#), [double y](#))

Public Attributes

- [double width](#)
- [double height](#)
- [double coordX](#)
- [double coordY](#)
- [double coordZ](#)

7.40.1 Detailed Description

Definition at line 23 of file [TEntrySimulation.h](#).

7.40.2 Member Function Documentation

7.40.2.1 isBelong()

```
bool TDetector::isBelong (  
    double x,  
    double y ) [inline]
```

Definition at line 26 of file [TEntrySimulation.h](#).

7.40.3 Member Data Documentation

7.40.3.1 coordX

```
double TDetector::coordX
```

Definition at line 25 of file [TEntrySimulation.h](#).

7.40.3.2 coordY

```
double TDetector::coordY
```

Definition at line 25 of file [TEntrySimulation.h](#).

7.40.3.3 coordZ

```
double TDetector::coordZ
```

Definition at line 25 of file [TEntrySimulation.h](#).

7.40.3.4 height

```
double TDetector::height
```

Definition at line 24 of file [TEntrySimulation.h](#).

7.40.3.5 width

```
double TDetector::width
```

Definition at line 24 of file [TEntrySimulation.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/simulation/inc/TEntrySimulation.h](#)

7.41 TDisk Struct Reference

```
#include <TEntrySimulation.h>
```

Public Member Functions

- bool [isBelong](#) (double x, double y)

Public Attributes

- double [radius](#)
- double [coordZ](#)

7.41.1 Detailed Description

Definition at line 11 of file [TEntrySimulation.h](#).

7.41.2 Member Function Documentation

7.41.2.1 isBelong()

```
bool TDisk::isBelong (  
    double x,  
    double y ) [inline]
```

Definition at line 14 of file [TEntrySimulation.h](#).

7.41.3 Member Data Documentation

7.41.3.1 coordZ

```
double TDisk::coordZ
```

Definition at line 13 of file [TEntrySimulation.h](#).

7.41.3.2 radius

```
double TDisk::radius
```

Definition at line 12 of file [TEntrySimulation.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/simulation/inc/TEntrySimulation.h](#)

7.42 TEntrySimulation Class Reference

```
#include <TEntrySimulation.h>
```

Public Member Functions

- void [setInitGeometry](#) (double diskRadius, double upperDiskCoordZ, double lowerDiskCoordZ, double detectorWidth, double detectorHeight, double detectorCoordZ)
- void [setSource](#) (double sourceRadius)
- void [setCollimator](#) (double diskRadius, double upperDiskCoordZ, double lowerDiskCoordZ)
- void [setDetector](#) (double detectorWidth, double detectorHeight, double detectorCoordX, double detectorCoordY, double detectorCoordZ)
- double [doCount](#) ()

Private Attributes

- [TDisk](#) source
- [TDisk](#) upperDisk
- [TDisk](#) lowerDisk
- [TDetector](#) detector

7.42.1 Detailed Description

Definition at line 35 of file [TEntrySimulation.h](#).

7.42.2 Member Function Documentation

7.42.2.1 doCount()

```
double TEntrySimulation::doCount ( )
```

Definition at line 33 of file [TEntrySimulation.cpp](#).

7.42.2.2 setCollimator()

```
void TEntrySimulation::setCollimator (
    double diskRadius,
    double upperDiskCoordZ,
    double lowerDiskCoordZ )
```

Definition at line 18 of file [TEntrySimulation.cpp](#).

7.42.2.3 setDetector()

```
void TEntrySimulation::setDetector (
    double detectorWidth,
    double detectorHeight,
    double detectorCoordX,
    double detectorCoordY,
    double detectorCoordZ )
```

Definition at line 25 of file [TEntrySimulation.cpp](#).

7.42.2.4 setInitGeometry()

```
void TEntrySimulation::setInitGeometry (
    double diskRadius,
    double upperDiskCoordZ,
    double lowerDiskCoordZ,
    double detectorWidth,
    double detectorHeight,
    double detectorCoordZ )
```

Definition at line 3 of file [TEntrySimulation.cpp](#).

7.42.2.5 setSource()

```
void TEntrySimulation::setSource (
    double sourceRadius )
```

Definition at line 13 of file [TEntrySimulation.cpp](#).

7.42.3 Member Data Documentation

7.42.3.1 detector

`TDetector TEntrySimulation::detector [private]`

Definition at line 40 of file [TEntrySimulation.h](#).

7.42.3.2 lowerDisk

`TDisk TEntrySimulation::lowerDisk [private]`

Definition at line 39 of file [TEntrySimulation.h](#).

7.42.3.3 source

`TDisk TEntrySimulation::source [private]`

Definition at line 37 of file [TEntrySimulation.h](#).

7.42.3.4 upperDisk

`TDisk TEntrySimulation::upperDisk [private]`

Definition at line 38 of file [TEntrySimulation.h](#).

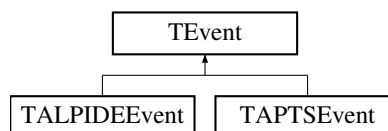
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/simulation/inc/TEntrySimulation.h](#)
- [/home/ychoi/ATOM/simulation/src/TEntrySimulation.cpp](#)

7.43 TEvent Class Reference

```
#include <TEvent.h>
```

Inheritance diagram for TEvent:



Public Member Functions

- [TEvent](#) ()
- virtual [~TEvent](#) ()
- void [setEvent](#) (const int event)
- const int [getEvent](#) () const

Private Attributes

- int [iEvent](#)

7.43.1 Detailed Description

Definition at line 4 of file [TEvent.h](#).

7.43.2 Constructor & Destructor Documentation

7.43.2.1 TEvent()

```
TEvent::TEvent ( )
```

Definition at line 3 of file [TEvent.cpp](#).

7.43.2.2 ~TEvent()

```
TEvent::~~TEvent ( ) [virtual]
```

Definition at line 5 of file [TEvent.cpp](#).

7.43.3 Member Function Documentation

7.43.3.1 getEvent()

```
const int TEvent::getEvent ( ) const
```

Definition at line 11 of file [TEvent.cpp](#).

7.43.3.2 setEvent()

```
void TEvent::setEvent (
    const int event )
```

Definition at line 7 of file [TEvent.cpp](#).

7.43.4 Member Data Documentation

7.43.4.1 iEvent

```
int TEvent::iEvent [private]
```

Definition at line 6 of file [TEvent.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/chip/inc/TEvent.h](#)
- [/home/ychoi/ATOM/chip/src/TEvent.cpp](#)

7.44 TExperimentData Class Reference

```
#include <TExperimentData.h>
```

Public Types

- enum { [kNotDeleted](#) = 0x02000000 }

Public Member Functions

- [TExperimentData](#) ()
- [TExperimentData](#) (const [TExperimentData](#) ©)
- [TExperimentData](#) & [operator=](#) (const [TExperimentData](#) ©)
- [~TExperimentData](#) ()
- void [setEvents](#) (const std::vector< [TALPIDEEvent](#) * > &events)
- void [setClusters](#) (const std::vector< [TCluster](#) * > &clusters)
- const std::vector< [TALPIDEEvent](#) * > [getEvents](#) () const
- const std::vector< [TCluster](#) * > [getClusters](#) () const
- bool [IsDestructed](#) () const
- bool [TestBit](#) (unsigned int f) const

Private Attributes

- std::vector< [TALPIDEEvent](#) * > [mEvents](#)
- std::vector< [TCluster](#) * > [mClusters](#)
- unsigned int [fBits](#)

7.44.1 Detailed Description

Definition at line 25 of file [TExperimentData.h](#).

7.44.2 Member Enumeration Documentation

7.44.2.1 anonymous enum

anonymous enum

Enumerator

kNotDeleted	
-----------------------------	--

Definition at line 56 of file [TExperimentData.h](#).

7.44.3 Constructor & Destructor Documentation

7.44.3.1 TExperimentData() [1/2]

```
TExperimentData::TExperimentData ( )
```

Definition at line 4 of file [TExperimentData.cpp](#).

7.44.3.2 TExperimentData() [2/2]

```
TExperimentData::TExperimentData (
    const TExperimentData & copy )
```

Definition at line 6 of file [TExperimentData.cpp](#).

7.44.3.3 ~TExperimentData()

```
TExperimentData::~~TExperimentData ( )
```

Definition at line 24 of file [TExperimentData.cpp](#).

7.44.4 Member Function Documentation

7.44.4.1 getClusters()

```
const std::vector< TCluster * > TExperimentData::getClusters ( ) const
```

Definition at line 92 of file [TExperimentData.cpp](#).

7.44.4.2 getEvents()

```
const std::vector< TALPIDEEvent * > TExperimentData::getEvents ( ) const
```

Definition at line 88 of file [TExperimentData.cpp](#).

7.44.4.3 IsDestructed()

```
bool TExperimentData::IsDestructed ( ) const [inline]
```

Definition at line 59 of file [TExperimentData.h](#).

7.44.4.4 operator=()

```
TExperimentData & TExperimentData::operator= (
    const TExperimentData & copy )
```

Definition at line 12 of file [TExperimentData.cpp](#).

7.44.4.5 setClusters()

```
void TExperimentData::setClusters (
    const std::vector< TCluster * > & clusters )
```

Definition at line 68 of file [TExperimentData.cpp](#).

7.44.4.6 setEvents()

```
void TExperimentData::setEvents (
    const std::vector< TALPIDEEEvent * > & events )
```

Definition at line 64 of file [TExperimentData.cpp](#).

7.44.4.7 TestBit()

```
bool TExperimentData::TestBit (
    unsigned int f ) const [inline]
```

Definition at line 60 of file [TExperimentData.h](#).

7.44.5 Member Data Documentation

7.44.5.1 fBits

```
unsigned int TExperimentData::fBits [private]
```

Definition at line 54 of file [TExperimentData.h](#).

7.44.5.2 mClusters

```
std::vector<TCluster*> TExperimentData::mClusters [private]
```

Definition at line 28 of file [TExperimentData.h](#).

7.44.5.3 mEvents

```
std::vector<TALPIDEEEvent*> TExperimentData::mEvents [private]
```

Definition at line 27 of file [TExperimentData.h](#).

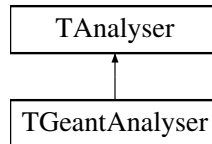
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/cluster/inc/TExperimentData.h](#)
- [/home/ychoi/ATOM/alpide/cluster/src/TExperimentData.cpp](#)

7.45 TGeantAnalyser Class Reference

```
#include <TGeantAnalyser.h>
```

Inheritance diagram for TGeantAnalyser:



Public Member Functions

- [TGeantAnalyser](#) ()=delete

Public Member Functions inherited from [TAnalyser](#)

- [TAnalyser](#) ()=default
- [TAnalyser](#) (TFile *inputFile, std::unordered_map< std::string, [TExperimentData](#) * > expData)
Construct a new [TAnalyser::TAnalyser](#) object.
- [~TAnalyser](#) ()
Destroy the [TAnalyser::TAnalyser](#) object.
- TTree * [openTree](#) (std::string treeName)
open trees
- void [storeEvents](#) ([CppConfigDictionary](#) settingConfig)
Extract event from TTree.
- void [doMasking](#) (int mMaskOver)
- void [openOutputGraphFile](#) (std::string_view fileName)
Open ROOT file to save results graph.
- void [openDirectory](#) (std::string_view typeName)
Set the new directory to store graph root file.
- void [setExpSettingLegend](#) ([CppConfigDictionary](#) settingConfig)
- void [doDivideBySize](#) (std::string_view typeName)
- [TExperimentData](#) * [getAnEventSet](#) (std::string_view typeName) const
- TH2D * [getHitPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TALPIDEEvent](#) * > &events)
- void [saveHitmap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- TH2D * [getClusterPlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
Generalized function to draw clustermap.
- TH1D * [getClustersizePlot](#) (const [CppConfigDictionary](#) &config, const std::vector< [TCluster](#) * > &clusters)
- void [setClusterDataWithShape](#) (const std::vector< int > &clusterSizeRange)
- void [saveClustermap](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- void [saveClustersize](#) (std::string typeName, const [CppConfigDictionary](#) &config)
- std::vector< int > [getClusterSizeRange](#) (const [CppConfigDictionary](#) &privateProperty)
- void [doShaping](#) (std::string_view typeName, const std::vector< int > &clusterSizeRange)
Store clusters to objects of [TClusterShape](#) for extracting shape informations.
- void [saveIndividualShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
Drawing individual shapes.
- void [saveSameSizeInfos](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
Drawing information for classifying multi-cluster.
- void [saveSameSizeShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveTotalShapes](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveSameSizeShapeEntry](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- void [saveTotalShapeEntry](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- bool [IsDestructed](#) () const
- bool [TestBit](#) (UInt_t f) const

Additional Inherited Members

Public Types inherited from [TAnalyser](#)

- enum { [kNotDeleted](#) = 0x02000000 }

Protected Attributes inherited from [TAnalyser](#)

- TFile * [mOutputFile](#) = nullptr
- bool [mIsOutputGraph](#) = false
- TPaveText * [mExpSettingLegend](#)
- std::unordered_map< std::string, TH2D * > [mHitmaps](#)
- std::unordered_map< std::string, TH2D * > [mClustermaps](#)
- std::unordered_map< std::string, TH1D * > [mClustersizes](#)
- std::unordered_map< std::string, std::unordered_map< int, std::vector< [TCluster](#) * > > > [mClusterDataWithShape](#)
- std::unordered_map< std::string, TDirectory * > [mDirectorySet](#)
- std::unordered_map< std::string, [TExperimentData](#) * > [mExpData](#)
- std::unordered_map< std::string, [TClusterDivideData](#) * > [mDivideData](#)
- std::unordered_map< std::string, std::vector< [TClusterShape](#) * > > [mClusterShapeSet](#)
- std::unordered_map< std::string, int > [mNTotalShapeSet](#)
- std::unordered_map< std::string, int > [mMaxModeSet](#)

7.45.1 Detailed Description

Definition at line 6 of file [TGeantAnalyser.h](#).

7.45.2 Constructor & Destructor Documentation

7.45.2.1 TGeantAnalyser()

```
TGeantAnalyser::TGeantAnalyser ( ) [delete]
```

The documentation for this class was generated from the following file:

- [/home/ychoi/ATOM/geant4/analysis/inc/TGeantAnalyser.h](#)

7.46 TGraphCompare Class Reference

```
#include <TGraphCompare.h>
```

Public Member Functions

- [TGraphCompare](#) (const std::vector< std::string > &graphFilePath)
- void [TCompareClusterSize](#) (std::string_view typeName, const [CppConfigDictionary](#) config)
- TH1D * [getClustersizeHistogram](#) (std::string_view pathInRoot, std::string operation)

Private Attributes

- `std::unordered_map< std::string, TFile * >` [mGraphFileSet](#)
- `std::vector< TH1D * >` [mClusterSizeSet](#)
- `std::vector< std::string >` [mGraphInfoSet](#)

7.46.1 Detailed Description

Definition at line 28 of file [TGraphCompare.h](#).

7.46.2 Constructor & Destructor Documentation**7.46.2.1 TGraphCompare()**

```
TGraphCompare::TGraphCompare (
    const std::vector< std::string > & graphFilePath )
```

Definition at line 81 of file [TGraphCompare.cpp](#).

7.46.3 Member Function Documentation**7.46.3.1 getClustersizeHistogram()**

```
TH1D * TGraphCompare::getClustersizeHistogram (
    std::string_view pathInRoot,
    std::string operation )
```

Definition at line 4 of file [TGraphCompare.cpp](#).

7.46.3.2 TCompareClusterSize()

```
void TGraphCompare::TCompareClusterSize (
    std::string_view typeName,
    const CppConfigDictionary config )
```

Definition at line 89 of file [TGraphCompare.cpp](#).

7.46.4 Member Data Documentation**7.46.4.1 mClusterSizeSet**

```
std::vector<TH1D*> TGraphCompare::mClusterSizeSet [private]
```

Definition at line 31 of file [TGraphCompare.h](#).

7.46.4.2 mGraphFileSet

```
std::unordered_map<std::string, TFile*> TGraphCompare::mGraphFileSet [private]
```

Definition at line 30 of file [TGraphCompare.h](#).

7.46.4.3 mGraphInfoSet

```
std::vector<std::string> TGraphCompare::mGraphInfoSet [private]
```

Definition at line 32 of file [TGraphCompare.h](#).

The documentation for this class was generated from the following files:

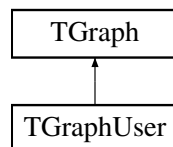
- [/home/ychoi/ATOM/alpide/comparison/inc/TGraphCompare.h](#)
- [/home/ychoi/ATOM/alpide/comparison/src/TGraphCompare.cpp](#)

7.47 TGraphUser Class Reference

The general tools for drawing TGraph Class with config file.

```
#include <TGraphUser.h>
```

Inheritance diagram for TGraphUser:



Public Member Functions

- [TGraphUser](#) (const [CppConfigDictionary](#) &config)
Construct a new [TGraphUser::TGraphUser](#) object.
- void [AddGraph](#) (TGraph *graph)
- void [AddGraph](#) (TGraph *graph, const [CppConfigDictionary](#) &graphConfig)
- void [Save](#) (const std::filesystem::path &outputPath=".")

Private Member Functions

- void [setCanvas](#) (const [CppConfigDictionary](#) &config)
Set Canvas Attributes.
- void [setLegend](#) (const [CppConfigDictionary](#) &config)
Set Legend Attributes.

Private Attributes

- TCanvas * [mCanvas](#)
- TLegend * [mLegend](#)
- TMultiGraph * [mMultiGraph](#)
- TString [savePath](#)
- std::string [mFileName](#)
- TString [title](#)
- TString [x_title](#)
- TString [y_title](#)
- std::vector< std::tuple< TGraph *, Style_t, Size_t, Style_t, Size_t > > [graphSetAndAttribute](#)
- std::array< Float_t, 4 > [canvasMargins](#)
- std::array< Float_t, 4 > [canvasOffsets](#)
- Style_t [mUniversalMarkerStyle](#)
- Size_t [mUniversalMarkerSize](#)
- Style_t [mUniversalLineStyle](#)
- Size_t [mUniversalLineWidth](#)
- TString [mLegendTitle](#)
- std::array< Float_t, 4 > [mLegendPoints](#)

7.47.1 Detailed Description

The general tools for drawing TGraph Class with config file.

Definition at line 17 of file [TGraphUser.h](#).

7.47.2 Constructor & Destructor Documentation

7.47.2.1 TGraphUser()

```
TGraphUser::TGraphUser (
    const CppConfigDictionary & config )
```

Construct a new [TGraphUser::TGraphUser](#) object.

Parameters

<i>config</i>	
---------------	--

Definition at line 10 of file [TGraphUser.cpp](#).

7.47.3 Member Function Documentation

7.47.3.1 AddGraph() [1/2]

```
void TGraphUser::AddGraph (
    TGraph * graph )
```

Definition at line 33 of file [TGraphUser.cpp](#).

7.47.3.2 AddGraph() [2/2]

```
void TGraphUser::AddGraph (
    TGraph * graph,
    const CppConfigDictionary & graphConfig )
```

Definition at line 38 of file [TGraphUser.cpp](#).

7.47.3.3 Save()

```
void TGraphUser::Save (
    const std::filesystem::path & outputPath = "." )
```

Definition at line 17 of file [TGraphUser.cpp](#).

7.47.3.4 setCanvas()

```
void TGraphUser::setCanvas (
    const CppConfigDictionary & config ) [private]
```

Set Canvas Attributes.

The following attributes are adjusted.

- Main title, x-axis and y-axis titles.
- Margin between canvas border and plot border

Parameters

<i>config</i>	
---------------	--

Definition at line 68 of file [TGraphUser.cpp](#).

7.47.3.5 setLegend()

```
void TGraphUser::setLegend (
    const CppConfigDictionary & config ) [private]
```

Set Legend Attributes.

Definition at line 118 of file [TGraphUser.cpp](#).

7.47.4 Member Data Documentation

7.47.4.1 canvasMargins

```
std::array<Float_t, 4> TGraphUser::canvasMargins [private]
```

Definition at line 29 of file [TGraphUser.h](#).

7.47.4.2 canvasOffsets

```
std::array<Float_t, 4> TGraphUser::canvasOffsets [private]
```

Definition at line 30 of file [TGraphUser.h](#).

7.47.4.3 graphSetAndAttribute

```
std::vector<std::tuple<TGraph*, Style_t, Size_t, Style_t, Size_t> > TGraphUser::graphSetAndAttribute [private]
```

Definition at line 27 of file [TGraphUser.h](#).

7.47.4.4 mCanvas

```
TCanvas* TGraphUser::mCanvas [private]
```

Definition at line 19 of file [TGraphUser.h](#).

7.47.4.5 mFileName

```
std::string TGraphUser::mFileName [private]
```

Definition at line 24 of file [TGraphUser.h](#).

7.47.4.6 mLegend

```
TLegend* TGraphUser::mLegend [private]
```

Definition at line 20 of file [TGraphUser.h](#).

7.47.4.7 mLegendPoints

```
std::array<Float_t, 4> TGraphUser::mLegendPoints [private]
```

Definition at line 38 of file [TGraphUser.h](#).

7.47.4.8 mLegendTitle

```
TString TGraphUser::mLegendTitle [private]
```

Definition at line 37 of file [TGraphUser.h](#).

7.47.4.9 mMultiGraph

```
TMultiGraph* TGraphUser::mMultiGraph [private]
```

Definition at line 21 of file [TGraphUser.h](#).

7.47.4.10 mUniversalLineStyle

```
Style_t TGraphUser::mUniversalLineStyle [private]
```

Definition at line 34 of file [TGraphUser.h](#).

7.47.4.11 mUniversalLineWidth

```
Size_t TGraphUser::mUniversalLineWidth [private]
```

Definition at line 35 of file [TGraphUser.h](#).

7.47.4.12 mUniversalMarkerSize

```
Size_t TGraphUser::mUniversalMarkerSize [private]
```

Definition at line 33 of file [TGraphUser.h](#).

7.47.4.13 mUniversalMarkerStyle

```
Style_t TGraphUser::mUniversalMarkerStyle [private]
```

Definition at line 32 of file [TGraphUser.h](#).

7.47.4.14 savePath

```
TString TGraphUser::savePath [private]
```

Definition at line 23 of file [TGraphUser.h](#).

7.47.4.15 title

```
TString TGraphUser::title [private]
```

Definition at line 26 of file [TGraphUser.h](#).

7.47.4.16 x_title

```
TString TGraphUser::x_title [private]
```

Definition at line 26 of file [TGraphUser.h](#).

7.47.4.17 y_title

```
TString TGraphUser::y_title [private]
```

Definition at line 26 of file [TGraphUser.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/drawing_tool/inc/TGraphUser.h](#)
- [/home/ychoi/ATOM/drawing_tool/src/TGraphUser.cpp](#)

7.48 TInputRoot Struct Reference

```
#include <TFileFormat.h>
```

Public Attributes

- [UChar_t](#) chipid
- [UInt_t](#) timeStamp
- [UShort_t](#) x
- [UShort_t](#) y

7.48.1 Detailed Description

Definition at line 8 of file [TFileFormat.h](#).

7.48.2 Member Data Documentation

7.48.2.1 chipid

```
UChar\_t TInputRoot::chipid
```

Definition at line 9 of file [TFileFormat.h](#).

7.48.2.2 timeStamp

```
UInt\_t TInputRoot::timeStamp
```

Definition at line 10 of file [TFileFormat.h](#).

7.48.2.3 x

```
UShort\_t TInputRoot::x
```

Definition at line 11 of file [TFileFormat.h](#).

7.48.2.4 y

```
UShort_t TInputRoot::y
```

Definition at line 12 of file [TFileFormat.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/alpide/analysis/inc/TFileFormat.h](#)

7.49 TMatrix2D< numT > Class Template Reference

```
#include <TMatrix2D.h>
```

Public Member Functions

- [TMatrix2D](#) ()
- [TMatrix2D](#) (int nRow, int nColumn)
- [TMatrix2D](#) (const std::vector< std::vector< numT > > &matrix)
- void [setMatrix](#) (const std::vector< std::vector< numT > > &matrix)
- void [setElement](#) (const int iRow, const int iColumn, const numT element)
- const numT [getElement](#) (const int iRow, const int iColumn) const
- const int [getNRow](#) () const
- const int [getNColumn](#) () const
- const std::pair< int, int > [getDimension](#) () const
- template<typename numT2 >
bool [isSameDimension](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [isSame](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasXSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasYSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasXYSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasRSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasRXSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasRYSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasRXYSymmetry](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
bool [hasHomeomorphism](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
[TMatrix2D](#) & [operator+=](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
[TMatrix2D](#) & [operator+](#) (const [TMatrix2D](#)< numT2 > &matrix)
- template<typename numT2 >
[TMatrix2D](#) & [operator-=](#) (const [TMatrix2D](#)< numT2 > &matrix)

- `template<typename numT2 >`
`TMatrix2D & operator-` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D & operator*==` (numT2 scalar)
- `template<typename numT2 >`
`TMatrix2D & operator*` (numT2 scalar)
- `template<typename numT2 >`
`TMatrix2D & operator*==` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D & operator*` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator+=` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator+` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator-=` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator-` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator*==` (const numT2 scalar)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator*` (const numT2 scalar)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator*==` (const `TMatrix2D< numT2 >` &matrix)
- `template<typename numT2 >`
`TMatrix2D< numT > & operator*` (const `TMatrix2D< numT2 >` &matrix)

Private Attributes

- `std::vector< std::vector< numT > >` `mMatrix`
- `int` `mNRow`
- `int` `mNColumn`

Friends

- `std::ostream & operator<<` (std::ostream &os, const `TMatrix2D< numT >` &matrix)

7.49.1 Detailed Description

`template<typename numT>`
`class TMatrix2D< numT >`

Definition at line 13 of file `TMatrix2D.h`.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 TMatrix2D() [1/3]

`template<typename numT >`
`TMatrix2D< numT >::TMatrix2D ()`

Definition at line 72 of file `TMatrix2D.h`.

7.49.2.2 TMatrix2D() [2/3]

```
template<typename numT >
TMatrix2D< numT >::TMatrix2D (
    int nRow,
    int nColumn )
```

Definition at line 75 of file [TMatrix2D.h](#).

7.49.2.3 TMatrix2D() [3/3]

```
template<typename numT >
TMatrix2D< numT >::TMatrix2D (
    const std::vector< std::vector< numT > > & matrix )
```

Definition at line 86 of file [TMatrix2D.h](#).

7.49.3 Member Function Documentation

7.49.3.1 getDimension()

```
template<typename numT >
const std::pair< int, int > TMatrix2D< numT >::getDimension ( ) const
```

Definition at line 128 of file [TMatrix2D.h](#).

7.49.3.2 getElement()

```
template<typename numT >
const numT TMatrix2D< numT >::getElement (
    const int iRow,
    const int iColumn ) const
```

Definition at line 106 of file [TMatrix2D.h](#).

7.49.3.3 getNColumn()

```
template<typename numT >
const int TMatrix2D< numT >::getNColumn ( ) const
```

Definition at line 123 of file [TMatrix2D.h](#).

7.49.3.4 getNRow()

```
template<typename numT >
const int TMatrix2D< numT >::getNRow ( ) const
```

Definition at line 118 of file [TMatrix2D.h](#).

7.49.3.5 hasHomeomorphism()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasHomeomorphism (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 248 of file [TMatrix2D.h](#).

7.49.3.6 hasRSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasRSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 196 of file [TMatrix2D.h](#).

7.49.3.7 hasRXSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasRXSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 209 of file [TMatrix2D.h](#).

7.49.3.8 hasRXYSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasRXYSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 235 of file [TMatrix2D.h](#).

7.49.3.9 hasRYSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasRYSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 222 of file [TMatrix2D.h](#).

7.49.3.10 hasXSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasXSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 170 of file [TMatrix2D.h](#).

7.49.3.11 hasXYSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasXYSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 183 of file [TMatrix2D.h](#).

7.49.3.12 hasYSymmetry()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::hasYSymmetry (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 157 of file [TMatrix2D.h](#).

7.49.3.13 isSame()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::isSame (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 144 of file [TMatrix2D.h](#).

7.49.3.14 isSameDimension()

```
template<typename numT >
template<typename numT2 >
bool TMatrix2D< numT >::isSameDimension (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 134 of file [TMatrix2D.h](#).

7.49.3.15 operator*() [1/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator* (
    const numT2 scalar )
```

Definition at line 387 of file [TMatrix2D.h](#).

7.49.3.16 operator*() [2/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator* (
    const TMatrix2D< numT2 > & matrix )
```

7.49.3.17 operator*() [3/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator* (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 419 of file [TMatrix2D.h](#).

7.49.3.18 operator*() [4/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator* (
    numT2 scalar )
```

7.49.3.19 operator*=() [1/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator*= (
    const numT2 scalar )
```

Definition at line 376 of file [TMatrix2D.h](#).

7.49.3.20 operator*=() [2/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator*= (
    const TMatrix2D< numT2 > & matrix )
```

7.49.3.21 operator*=() [3/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator*= (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 398 of file [TMatrix2D.h](#).

7.49.3.22 operator*=() [4/4]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator*= (
    numT2 scalar )
```

7.49.3.23 operator+() [1/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator+ (
    const TMatrix2D< numT2 > & matrix )
```

7.49.3.24 operator+() [2/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator+ (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 328 of file [TMatrix2D.h](#).

7.49.3.25 operator+=() [1/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator+= (
    const TMatrix2D< numT2 > & matrix )
```

7.49.3.26 operator+=() [2/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator+= (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 312 of file [TMatrix2D.h](#).

7.49.3.27 operator-() [1/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator- (
    const TMatrix2D< numT2 > & matrix )
```

7.49.3.28 operator-() [2/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator- (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 360 of file [TMatrix2D.h](#).

7.49.3.29 operator-=() [1/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D & TMatrix2D< numT >::operator-= (
    const TMatrix2D< numT2 > & matrix )
```

7.49.3.30 operator-=() [2/2]

```
template<typename numT >
template<typename numT2 >
TMatrix2D< numT > & TMatrix2D< numT >::operator-= (
    const TMatrix2D< numT2 > & matrix )
```

Definition at line 344 of file [TMatrix2D.h](#).

7.49.3.31 setElement()

```
template<typename numT >
void TMatrix2D< numT >::setElement (
    const int iRow,
    const int iColumn,
    const numT element )
```

Definition at line 101 of file [TMatrix2D.h](#).

7.49.3.32 setMatrix()

```
template<typename numT >
void TMatrix2D< numT >::setMatrix (
    const std::vector< std::vector< numT > > & matrix )
```

Definition at line 93 of file [TMatrix2D.h](#).

7.49.4 Friends And Related Symbol Documentation

7.49.4.1 operator<<

```
template<typename numT >
std::ostream & operator<< (
    std::ostream & os,
    const TMatrix2D< numT > & matrix ) [friend]
```

Definition at line 299 of file [TMatrix2D.h](#).

7.49.5 Member Data Documentation

7.49.5.1 mMatrix

```
template<typename numT >
std::vector<std::vector<numT> > TMatrix2D< numT >::mMatrix [private]
```

Definition at line 15 of file [TMatrix2D.h](#).

7.49.5.2 mNColumn

```
template<typename numT >
int TMatrix2D< numT >::mNColumn [private]
```

Definition at line 16 of file [TMatrix2D.h](#).

7.49.5.3 mNRow

```
template<typename numT >
int TMatrix2D< numT >::mNRow [private]
```

Definition at line 16 of file [TMatrix2D.h](#).

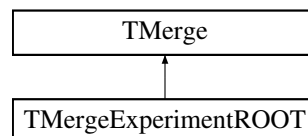
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/cluster/inc/TClusterShape.h](#)
- [/home/ychoi/ATOM/alpide/cluster/inc/TMatrix2D.h](#)

7.50 TMerge Class Reference

```
#include <TMerge.h>
```

Inheritance diagram for TMerge:



Public Member Functions

- [TMerge](#) (std::string_view outputFileName, const std::vector< std::string > &inputFileNames)

Protected Attributes

- std::string [mOutputFileName](#)
- std::vector< std::string > [mInputFileNames](#)

7.50.1 Detailed Description

Definition at line 52 of file [TMerge.h](#).

7.50.2 Constructor & Destructor Documentation

7.50.2.1 TMerge()

```
TMerge::TMerge (
    std::string_view outputFileName,
    const std::vector< std::string > & inputFileNames )
```

Definition at line 4 of file [TMerge.cpp](#).

7.50.3 Member Data Documentation

7.50.3.1 mInputFileNames

```
std::vector<std::string> TMerge::mInputFileNames [protected]
```

Definition at line 55 of file [TMerge.h](#).

7.50.3.2 mOutputFileName

```
std::string TMerge::mOutputFileName [protected]
```

Definition at line 54 of file [TMerge.h](#).

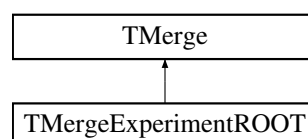
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/comparison/inc/TMerge.h](#)
- [/home/ychoi/ATOM/alpide/comparison/src/TMerge.cpp](#)

7.51 TMergeExperimentROOT Class Reference

```
#include <TMerge.h>
```

Inheritance diagram for TMergeExperimentROOT:



Public Member Functions

- [TMergeExperimentROOT](#) (std::string_view outputFileName, const std::vector< std::string > &inputFileNames)
- void [mergeFile](#) ()
- [~TMergeExperimentROOT](#) ()

Public Member Functions inherited from [TMerge](#)

- [TMerge](#) (std::string_view outputFileName, const std::vector< std::string > &inputFileNames)

Private Attributes

- TFile * [mOutputFile](#)
- [TInputRoot](#) [mInputValue](#)
- [TInputRoot](#) [mOutputValue](#)

Additional Inherited Members

Protected Attributes inherited from [TMerge](#)

- std::string [mOutputFileName](#)
- std::vector< std::string > [mInputFileNames](#)

7.51.1 Detailed Description

Definition at line 60 of file [TMerge.h](#).

7.51.2 Constructor & Destructor Documentation

7.51.2.1 [TMergeExperimentROOT\(\)](#)

```
TMergeExperimentROOT::TMergeExperimentROOT (
    std::string_view outputFileName,
    const std::vector< std::string > & inputFileNames )
```

Definition at line 8 of file [TMerge.cpp](#).

7.51.2.2 [~TMergeExperimentROOT\(\)](#)

```
TMergeExperimentROOT::~~TMergeExperimentROOT ( )
```

Definition at line 58 of file [TMerge.cpp](#).

7.51.3 Member Function Documentation

7.51.3.1 mergeFile()

```
void TMergeExperimentROOT::mergeFile ( )
```

Definition at line 12 of file [TMerge.cpp](#).

7.51.4 Member Data Documentation

7.51.4.1 mInputValue

```
TInputRoot TMergeExperimentROOT::mInputValue [private]
```

Definition at line 63 of file [TMerge.h](#).

7.51.4.2 mOutputFile

```
TFile* TMergeExperimentROOT::mOutputFile [private]
```

Definition at line 62 of file [TMerge.h](#).

7.51.4.3 mOutputValue

```
TInputRoot TMergeExperimentROOT::mOutputValue [private]
```

Definition at line 64 of file [TMerge.h](#).

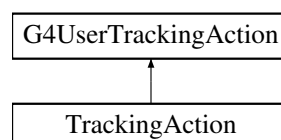
The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/comparison/inc/TMerge.h](#)
- [/home/ychoi/ATOM/alpide/comparison/src/TMerge.cpp](#)

7.52 TrackingAction Class Reference

```
#include <TrackingAction.h>
```

Inheritance diagram for TrackingAction:



Public Member Functions

- [TrackingAction](#) ()
- [~TrackingAction](#) ()
- virtual void [PreUserTrackingAction](#) (const G4Track *)
- virtual void [PostUserTrackingAction](#) (const G4Track *)

7.52.1 Detailed Description

Definition at line 11 of file [TrackingAction.h](#).

7.52.2 Constructor & Destructor Documentation

7.52.2.1 TrackingAction()

```
TrackingAction::TrackingAction ( )
```

Definition at line 4 of file [TrackingAction.cpp](#).

7.52.2.2 ~TrackingAction()

```
TrackingAction::~~TrackingAction ( )
```

Definition at line 7 of file [TrackingAction.cpp](#).

7.52.3 Member Function Documentation

7.52.3.1 PostUserTrackingAction()

```
void TrackingAction::PostUserTrackingAction (
    const G4Track * track ) [virtual]
```

Definition at line 16 of file [TrackingAction.cpp](#).

7.52.3.2 PreUserTrackingAction()

```
void TrackingAction::PreUserTrackingAction (
    const G4Track * track ) [virtual]
```

Definition at line 11 of file [TrackingAction.cpp](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/geant4/main/inc/TrackingAction.h](#)
- [/home/ychoi/ATOM/geant4/main/src/TrackingAction.cpp](#)

7.53 TrackTuple Struct Reference

```
#include <AnalysisManager.h>
```

Public Attributes

- int [eventGlobalID](#)
- int [trackGlobalID](#)
- int [trackLocalID](#)
- int [trackParentLocalID](#)
- std::string [particleName](#)
- std::string [processName](#)
- std::string [volumeName](#)
- double [genTime](#)
- double [genPosition](#) [3]
- double [genKineticEnergy](#)
- double [genMomentum](#) [3]
- int [nSteps](#)

7.53.1 Detailed Description

Definition at line 36 of file [AnalysisManager.h](#).

7.53.2 Member Data Documentation

7.53.2.1 eventGlobalID

```
int TrackTuple::eventGlobalID
```

Definition at line 37 of file [AnalysisManager.h](#).

7.53.2.2 genKineticEnergy

```
double TrackTuple::genKineticEnergy
```

Definition at line 46 of file [AnalysisManager.h](#).

7.53.2.3 genMomentum

```
double TrackTuple::genMomentum[3]
```

Definition at line 47 of file [AnalysisManager.h](#).

7.53.2.4 genPosition

```
double TrackTuple::genPosition[3]
```

Definition at line 45 of file [AnalysisManager.h](#).

7.53.2.5 genTime

```
double TrackTuple::genTime
```

Definition at line 44 of file [AnalysisManager.h](#).

7.53.2.6 nSteps

```
int TrackTuple::nSteps
```

Definition at line 48 of file [AnalysisManager.h](#).

7.53.2.7 particleName

```
std::string TrackTuple::particleName
```

Definition at line 41 of file [AnalysisManager.h](#).

7.53.2.8 processName

```
std::string TrackTuple::processName
```

Definition at line 42 of file [AnalysisManager.h](#).

7.53.2.9 trackGlobalID

```
int TrackTuple::trackGlobalID
```

Definition at line 38 of file [AnalysisManager.h](#).

7.53.2.10 trackLocalID

```
int TrackTuple::trackLocalID
```

Definition at line 39 of file [AnalysisManager.h](#).

7.53.2.11 trackParentLocalID

```
int TrackTuple::trackParentLocalID
```

Definition at line 40 of file [AnalysisManager.h](#).

7.53.2.12 volumeName

```
std::string TrackTuple::volumeName
```

Definition at line 43 of file [AnalysisManager.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h](#)

7.54 TShapeInfo Struct Reference

The information set structure for clusters that having homeomorphism shape.

```
#include <TClusterShape.h>
```

Public Attributes

- [TCluster](#) * [mPresidentCluster](#)
- [TMatrix2D](#)< int > * [mClusterMatrix](#)
- [TH2I](#) * [mClusterMap](#)
- int [iShape](#)
- int [mEntry](#)
- int [mShortBinN](#)
- int [mLongBinN](#)

7.54.1 Detailed Description

The information set structure for clusters that having homeomorphism shape.

It stores on cluster for extracting basic cluster informations. And cluster image and the number of homeomorphism clusters are saved.

Warning

Bug

Todo Add struct member if needed.

Definition at line 38 of file [TClusterShape.h](#).

7.54.2 Member Data Documentation

7.54.2.1 iShape

```
int TShapeInfo::iShape
```

Definition at line 42 of file [TClusterShape.h](#).

7.54.2.2 mClusterMap

```
TH2I* TShapeInfo::mClusterMap
```

Definition at line 41 of file [TClusterShape.h](#).

7.54.2.3 mClusterMatrix

```
TMatrix2D<int>* TShapeInfo::mClusterMatrix
```

Definition at line 40 of file [TClusterShape.h](#).

7.54.2.4 mEntry

```
int TShapeInfo::mEntry
```

Definition at line 43 of file [TClusterShape.h](#).

7.54.2.5 mLongBinN

```
int TShapeInfo::mLongBinN
```

Definition at line 45 of file [TClusterShape.h](#).

7.54.2.6 mPresidentCluster

```
TCluster* TShapeInfo::mPresidentCluster
```

Definition at line 39 of file [TClusterShape.h](#).

7.54.2.7 mShortBinN

```
int TShapeInfo::mShortBinN
```

Definition at line 44 of file [TClusterShape.h](#).

The documentation for this struct was generated from the following file:

- [/home/ychoi/ATOM/alpide/cluster/inc/TClusterShape.h](#)

7.55 TThreshold Class Reference

```
#include <TThreshold.h>
```

Public Member Functions

- [TThreshold](#) ()
- [TThreshold](#) (int x, int y)
- [TThreshold](#) (const std::array< int, 2 > &coordinate)
- [TThreshold](#) (int x, int y, const std::array< int, 50 > &dacs)
- [TThreshold](#) (const std::array< int, 2 > &coodrdinate, const std::array< int, 50 > &dacs)
- [TThreshold](#) (int x, int y, std::array< int, 50 > &&dacs)
- [TThreshold](#) (const std::array< int, 2 > &coodrdinate, std::array< int, 50 > &&dacs)
- [TThreshold](#) (const [TThreshold](#) ©)
- [TThreshold](#) & [operator=](#) (const [TThreshold](#) ©)
- [TThreshold](#) ([TThreshold](#) &&move)
- [TThreshold](#) & [operator=](#) ([TThreshold](#) &&move)
- [~TThreshold](#) ()
- [ThrCondition](#) [calculateThreshold](#) ()
- void [savePlot](#) ()
- const double [getX](#) () const
- const double [getY](#) () const
- const double [getThreshold](#) () const
- const double [getError](#) () const
- const double [getQualityFactor](#) () const
- const [ThrCondition](#) [getCondition](#) () const

Private Attributes

- int [mX](#)
- int [mY](#)
- std::array< int, 50 > [mDacs](#)
- double [mThr](#)
- double [mErr](#)
- double [mQualityFactor](#)
- [ThrCondition](#) [mCondition](#)
- std::unique_ptr< TGraph > [thresholdGraph](#)
- std::unique_ptr< TF1 > [fitFunction](#)

7.55.1 Detailed Description

Definition at line 24 of file [TThreshold.h](#).

7.55.2 Constructor & Destructor Documentation**7.55.2.1 TThreshold() [1/9]**

```
TThreshold::TThreshold ( )
```

Definition at line 4 of file [TThreshold.cpp](#).

7.55.2.2 TThreshold() [2/9]

```
TThreshold::TThreshold (
    int x,
    int y )
```

Definition at line 6 of file [TThreshold.cpp](#).

7.55.2.3 TThreshold() [3/9]

```
TThreshold::TThreshold (
    const std::array< int, 2 > & coordinate )
```

Definition at line 8 of file [TThreshold.cpp](#).

7.55.2.4 TThreshold() [4/9]

```
TThreshold::TThreshold (
    int x,
    int y,
    const std::array< int, 50 > & dacs )
```

Definition at line 10 of file [TThreshold.cpp](#).

7.55.2.5 TThreshold() [5/9]

```
TThreshold::TThreshold (
    const std::array< int, 2 > & coordrdinate,
    const std::array< int, 50 > & dacs )
```

Definition at line 15 of file [TThreshold.cpp](#).

7.55.2.6 TThreshold() [6/9]

```
TThreshold::TThreshold (
    int x,
    int y,
    std::array< int, 50 > && dacs )
```

Definition at line 19 of file [TThreshold.cpp](#).

7.55.2.7 TThreshold() [7/9]

```
TThreshold::TThreshold (
    const std::array< int, 2 > & coordrdinate,
    std::array< int, 50 > && dacs )
```

Definition at line 23 of file [TThreshold.cpp](#).

7.55.2.8 TThreshold() [8/9]

```
TThreshold::TThreshold (
    const TThreshold & copy )
```

Definition at line 27 of file [TThreshold.cpp](#).

7.55.2.9 TThreshold() [9/9]

```
TThreshold::TThreshold (
    TThreshold && move )
```

Definition at line 40 of file [TThreshold.cpp](#).

7.55.2.10 ~TThreshold()

```
TThreshold::~~TThreshold ( )
```

Definition at line 53 of file [TThreshold.cpp](#).

7.55.3 Member Function Documentation

7.55.3.1 calculateThreshold()

```
ThrCondition TThreshold::calculateThreshold ( )
```

Definition at line 55 of file [TThreshold.cpp](#).

7.55.3.2 getCondition()

```
const ThrCondition TThreshold::getCondition ( ) const
```

Definition at line 122 of file [TThreshold.cpp](#).

7.55.3.3 getError()

```
const double TThreshold::getError ( ) const
```

Definition at line 114 of file [TThreshold.cpp](#).

7.55.3.4 getQualityFactor()

```
const double TThreshold::getQualityFactor ( ) const
```

Definition at line 118 of file [TThreshold.cpp](#).

7.55.3.5 `getThreshold()`

```
const double TThreshold::getThreshold ( ) const
```

Definition at line 110 of file [TThreshold.cpp](#).

7.55.3.6 `getX()`

```
const double TThreshold::getX ( ) const
```

Definition at line 102 of file [TThreshold.cpp](#).

7.55.3.7 `getY()`

```
const double TThreshold::getY ( ) const
```

Definition at line 106 of file [TThreshold.cpp](#).

7.55.3.8 `operator=()` [1/2]

```
TThreshold & TThreshold::operator= (
    const TThreshold & copy )
```

Definition at line 31 of file [TThreshold.cpp](#).

7.55.3.9 `operator=()` [2/2]

```
TThreshold & TThreshold::operator= (
    TThreshold && move )
```

Definition at line 44 of file [TThreshold.cpp](#).

7.55.3.10 `savePlot()`

```
void TThreshold::savePlot ( )
```

Definition at line 95 of file [TThreshold.cpp](#).

7.55.4 Member Data Documentation

7.55.4.1 `fitFunction`

```
std::unique_ptr<TF1> TThreshold::fitFunction [private]
```

Definition at line 36 of file [TThreshold.h](#).

7.55.4.2 mCondition

`ThrCondition TThreshold::mCondition [private]`

Definition at line 33 of file [TThreshold.h](#).

7.55.4.3 mDacs

`std::array<int, 50> TThreshold::mDacs [private]`

Definition at line 28 of file [TThreshold.h](#).

7.55.4.4 mErr

`double TThreshold::mErr [private]`

Definition at line 30 of file [TThreshold.h](#).

7.55.4.5 mQualityFactor

`double TThreshold::mQualityFactor [private]`

Definition at line 31 of file [TThreshold.h](#).

7.55.4.6 mThr

`double TThreshold::mThr [private]`

Definition at line 29 of file [TThreshold.h](#).

7.55.4.7 mX

`int TThreshold::mX [private]`

Definition at line 26 of file [TThreshold.h](#).

7.55.4.8 mY

`int TThreshold::mY [private]`

Definition at line 27 of file [TThreshold.h](#).

7.55.4.9 thresholdGraph

```
std::unique_ptr<TGraph> TThreshold::thresholdGraph [private]
```

Definition at line 35 of file [TThreshold.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/threshold/inc/TThreshold.h](#)
- [/home/ychoi/ATOM/alpide/threshold/src/TThreshold.cpp](#)

7.56 TThresholdAnalyser Class Reference

```
#include <TThresholdAnalyser.h>
```

Public Member Functions

- [TThresholdAnalyser \(\)](#)
- [TThresholdAnalyser \(std::ifstream &file\)](#)
- [~TThresholdAnalyser \(\)](#)
- void [openFile](#) (std::ifstream &file)
- void [refineData](#) ()
- void [saveThresholdDistribution](#) (std::string_view title) const
- void [saveErrorDistribution](#) (std::string_view title) const
- void [saveThresholdmap](#) (std::string_view title) const
- void [saveQualityDistribution](#) (std::string_view title) const

Private Attributes

- std::ifstream [mFile](#)
- int [mVcasn](#)
- int [mlthr](#)
- std::vector< [TThreshold](#) * > [mThresholds](#)
- TH1 * [mThresholdDistribution](#)
- TH1 * [mErrorDistribution](#)
- TH2 * [mThresholdmap](#)
- TH1 * [mChi2NdfDistribution](#)

7.56.1 Detailed Description

Definition at line 36 of file [TThresholdAnalyser.h](#).

7.56.2 Constructor & Destructor Documentation

7.56.2.1 TThresholdAnalyser() [1/2]

```
TThresholdAnalyser::TThresholdAnalyser ( )
```

Fitting quality distribution (Chi2 / Ndof)

Definition at line 5 of file [TThresholdAnalyser.cpp](#).

7.56.2.2 TThresholdAnalyser() [2/2]

```
TThresholdAnalyser::TThresholdAnalyser (
    std::ifstream & file )
```

Definition at line 7 of file [TThresholdAnalyser.cpp](#).

7.56.2.3 ~TThresholdAnalyser()

```
TThresholdAnalyser::~~TThresholdAnalyser ( )
```

Definition at line 11 of file [TThresholdAnalyser.cpp](#).

7.56.3 Member Function Documentation

7.56.3.1 openFile()

```
void TThresholdAnalyser::openFile (
    std::ifstream & file )
```

Definition at line 15 of file [TThresholdAnalyser.cpp](#).

7.56.3.2 refineData()

```
void TThresholdAnalyser::refineData ( )
```

Definition at line 43 of file [TThresholdAnalyser.cpp](#).

7.56.3.3 saveErrorDistribution()

```
void TThresholdAnalyser::saveErrorDistribution (
    std::string_view title ) const
```

Definition at line 83 of file [TThresholdAnalyser.cpp](#).

7.56.3.4 saveQualityDistribution()

```
void TThresholdAnalyser::saveQualityDistribution (
    std::string_view title ) const
```

Definition at line 99 of file [TThresholdAnalyser.cpp](#).

7.56.3.5 saveThresholdDistribution()

```
void TThresholdAnalyser::saveThresholdDistribution (
    std::string_view title ) const
```

Definition at line 77 of file [TThresholdAnalyser.cpp](#).

7.56.3.6 saveThresholdmap()

```
void TThresholdAnalyser::saveThresholdmap (
    std::string_view title ) const
```

Definition at line 89 of file [TThresholdAnalyser.cpp](#).

7.56.4 Member Data Documentation

7.56.4.1 mChi2NdfDistribution

```
TH1* TThresholdAnalyser::mChi2NdfDistribution [private]
```

Definition at line 46 of file [TThresholdAnalyser.h](#).

7.56.4.2 mErrorDistribution

```
TH1* TThresholdAnalyser::mErrorDistribution [private]
```

Error value distribution plot

Definition at line 44 of file [TThresholdAnalyser.h](#).

7.56.4.3 mFile

```
std::ifstream TThresholdAnalyser::mFile [private]
```

Definition at line 38 of file [TThresholdAnalyser.h](#).

7.56.4.4 mlthr

```
int TThresholdAnalyser::mIthr [private]
```

vcasn and ithr value. They are key values for determine threshold

Definition at line 39 of file [TThresholdAnalyser.h](#).

7.56.4.5 mThresholdDistribution

```
TH1* TThresholdAnalyser::mThresholdDistribution [private]
```

Threshold value distribution plot

Definition at line 43 of file [TThresholdAnalyser.h](#).

7.56.4.6 mThresholdmap

```
TH2* TThresholdAnalyser::mThresholdmap [private]
```

Thresholdmap

Definition at line 45 of file [TThresholdAnalyser.h](#).

7.56.4.7 mThresholds

```
std::vector<TThreshold*> TThresholdAnalyser::mThresholds [private]
```

The array in which the informations about threshold of ALPIDE are stored

Definition at line 41 of file [TThresholdAnalyser.h](#).

7.56.4.8 mVcasn

```
int TThresholdAnalyser::mVcasn [private]
```

Dat file

Definition at line 39 of file [TThresholdAnalyser.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/analysis/inc/TThresholdAnalyser.h](#)
- [/home/ychoi/ATOM/alpide/analysis/src/TThresholdAnalyser.cpp](#)

7.57 TThresholdCompare Class Reference

```
#include <TThresholdCompare.h>
```

Public Member Functions

- [TThresholdCompare](#) ()
- [TThresholdCompare](#) (std::vector< std::unique_ptr< [TThreshold](#) * > > thresholds)

Private Attributes

- std::vector< std::unique_ptr< [TThreshold](#) * > > [thresholds_](#)

7.57.1 Detailed Description

Definition at line 15 of file [TThresholdCompare.h](#).

7.57.2 Constructor & Destructor Documentation

7.57.2.1 TThresholdCompare() [1/2]

```
TThresholdCompare::TThresholdCompare ( )
```

Definition at line 4 of file [TThresholdCompare.cpp](#).

7.57.2.2 TThresholdCompare() [2/2]

```
TThresholdCompare::TThresholdCompare (
    std::vector< std::unique_ptr< TThreshold * > > thresholds )
```

Definition at line 5 of file [TThresholdCompare.cpp](#).

7.57.3 Member Data Documentation

7.57.3.1 thresholds_

```
std::vector<std::unique_ptr<TThreshold*> > TThresholdCompare::thresholds_ [private]
```

Definition at line 17 of file [TThresholdCompare.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/alpide/threshold/inc/TThresholdCompare.h](#)
- [/home/ychoi/ATOM/alpide/threshold/src/TThresholdCompare.cpp](#)

7.58 TTimer Class Reference

```
#include <cppTimer.h>
```

Public Member Functions

- [TTimer](#) ()
- void [Measure](#) ()
- void [EndProgram](#) ()

Private Attributes

- clock_t [start](#)
- clock_t [finish](#)
- double [duration](#)

7.58.1 Detailed Description

Definition at line 7 of file [cppTimer.h](#).

7.58.2 Constructor & Destructor Documentation

7.58.2.1 TTimer()

```
TTimer::TTimer ( )
```

Definition at line 3 of file [cppTimer.cpp](#).

7.58.3 Member Function Documentation

7.58.3.1 EndProgram()

```
void TTimer::EndProgram ( )
```

Definition at line 13 of file [cppTimer.cpp](#).

7.58.3.2 Measure()

```
void TTimer::Measure ( )
```

Definition at line 7 of file [cppTimer.cpp](#).

7.58.4 Member Data Documentation

7.58.4.1 duration

```
double TTimer::duration [private]
```

Definition at line 10 of file [cppTimer.h](#).

7.58.4.2 finish

```
clock_t TTimer::finish [private]
```

Definition at line 9 of file [cppTimer.h](#).

7.58.4.3 start

```
clock_t TTimer::start [private]
```

Definition at line 9 of file [cppTimer.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cppTimer.h](#)
- [/home/ychoi/ATOM/pycpp/src/cppTimer.cpp](#)

7.59 Unit Class Reference

```
#include <cppUnit.h>
```

Public Member Functions

- [Unit](#) ()
- [Unit](#) (std::string_view unit)
- void [setUnit](#) (std::string_view unit)
- bool [operator==](#) (const [Unit](#) &ref) const
- bool [operator!=](#) (const [Unit](#) &ref) const
- [Unit operator*](#) (const [Unit](#) &ref) const
- [Unit operator*="](#) (const [Unit](#) &ref)
- [Unit operator/](#) (const [Unit](#) &ref) const
- [Unit operator/=](#) (const [Unit](#) &ref)
- const int [getDigit](#) () const
- const std::array< int, 7 > & [getUnitCount](#) () const
- const std::vector< std::string > [getNonBasicUnits](#) () const
- const std::string [getUnit](#) () const

Static Public Member Functions

- static void [setUserUnit](#) (std::string_view newUnit, std::string_view newSiUnit)

Private Member Functions

- bool [seperate](#) (std::vector< std::string > &store, std::string_view unit)
- bool [vanishSlash](#) (std::vector< std::string > &store)
- bool [removePrefix](#) (std::vector< std::string > &store)
- void [setUnitCount](#) (std::string unit, int power)
- void [plusCount](#) (std::array< int, 7 > nums)

Private Attributes

- int [mDigit](#)
- std::array< int, 7 > [unitCount](#) = {0, 0, 0, 0, 0, 0, 0}
- std::vector< std::string > [nonBasicUnit](#)

Static Private Attributes

- static std::vector< std::pair< std::string, std::array< int, 7 > > > [userUnits](#) = { }
- static std::vector< std::string > [exceptPrefixList](#) = {"m", "mol", "cd"}
- static bool [isUserUnit](#) = false

Friends

- std::ostream & [operator<<](#) (std::ostream &os, [Unit](#) &ref)
- std::ostream & [operator<<](#) (std::ostream &os, const [Unit](#) &ref)

7.59.1 Detailed Description

Definition at line 16 of file [cppUnit.h](#).

7.59.2 Constructor & Destructor Documentation

7.59.2.1 Unit() [1/2]

```
Unit::Unit ( )
```

Definition at line 3 of file [cppUnit.cpp](#).

7.59.2.2 Unit() [2/2]

```
Unit::Unit (
    std::string_view unit )
```

Definition at line 5 of file [cppUnit.cpp](#).

7.59.3 Member Function Documentation

7.59.3.1 getDigit()

```
const int Unit::getDigit ( ) const
```

Definition at line 255 of file [cppUnit.cpp](#).

7.59.3.2 getNonBasicUnits()

```
const std::vector< std::string > Unit::getNonBasicUnits ( ) const
```

Definition at line 263 of file [cppUnit.cpp](#).

7.59.3.3 `getUnit()`

```
const std::string Unit::getUnit ( ) const
```

Definition at line 267 of file [cppUnit.cpp](#).

7.59.3.4 `getUnitCount()`

```
const std::array< int, 7 > & Unit::getUnitCount ( ) const
```

Definition at line 259 of file [cppUnit.cpp](#).

7.59.3.5 `operator"!=()`

```
bool Unit::operator!= (
    const Unit & ref ) const
```

Definition at line 183 of file [cppUnit.cpp](#).

7.59.3.6 `operator*()`

```
Unit Unit::operator* (
    const Unit & ref ) const
```

Definition at line 187 of file [cppUnit.cpp](#).

7.59.3.7 `operator*=()`

```
Unit Unit::operator*= (
    const Unit & ref )
```

Definition at line 193 of file [cppUnit.cpp](#).

7.59.3.8 `operator/()`

```
Unit Unit::operator/ (
    const Unit & ref ) const
```

Definition at line 206 of file [cppUnit.cpp](#).

7.59.3.9 `operator/=()`

```
Unit Unit::operator/= (
    const Unit & ref )
```

Definition at line 212 of file [cppUnit.cpp](#).

7.59.3.10 operator==()

```
bool Unit::operator== (
    const Unit & ref ) const
```

Definition at line 179 of file [cppUnit.cpp](#).

7.59.3.11 plusCount()

```
void Unit::plusCount (
    std::array< int, 7 > nums ) [private]
```

Definition at line 169 of file [cppUnit.cpp](#).

7.59.3.12 removePrefix()

```
bool Unit::removePrefix (
    std::vector< std::string > & store ) [private]
```

Definition at line 122 of file [cppUnit.cpp](#).

7.59.3.13 seperate()

```
bool Unit::seperate (
    std::vector< std::string > & store,
    std::string_view unit ) [private]
```

Definition at line 64 of file [cppUnit.cpp](#).

7.59.3.14 setUnit()

```
void Unit::setUnit (
    std::string_view unit )
```

Definition at line 50 of file [cppUnit.cpp](#).

7.59.3.15 setUnitCount()

```
void Unit::setUnitCount (
    std::string unit,
    int power ) [private]
```

Definition at line 143 of file [cppUnit.cpp](#).

7.59.3.16 setUserUnit()

```
void Unit::setUserUnit (
    std::string_view newUnit,
    std::string_view newSiUnit ) [static]
```

Definition at line 14 of file [cppUnit.cpp](#).

7.59.3.17 vanishSlash()

```
bool Unit::vanishSlash (
    std::vector< std::string > & store ) [private]
```

Definition at line 102 of file [cppUnit.cpp](#).

7.59.4 Friends And Related Symbol Documentation

7.59.4.1 operator<< [1/2]

```
std::ostream & operator<< (
    std::ostream & os,
    const Unit & ref ) [friend]
```

Definition at line 240 of file [cppUnit.cpp](#).

7.59.4.2 operator<< [2/2]

```
std::ostream & operator<< (
    std::ostream & os,
    Unit & ref ) [friend]
```

Definition at line 225 of file [cppUnit.cpp](#).

7.59.5 Member Data Documentation

7.59.5.1 exceptPrefixList

```
std::vector< std::string > Unit::exceptPrefixList = {"m", "mol", "cd"} [static], [private]
```

Definition at line 11 of file [cppUnit.h](#).

7.59.5.2 isUserUnit

```
bool Unit::isUserUnit = false [static], [private]
```

Definition at line 23 of file [cppUnit.h](#).

7.59.5.3 mDigit

```
int Unit::mDigit [private]
```

Definition at line 18 of file [cppUnit.h](#).

7.59.5.4 nonBasicUnit

```
std::vector<std::string> Unit::nonBasicUnit [private]
```

Definition at line 20 of file [cppUnit.h](#).

7.59.5.5 unitCount

```
std::array<int, 7> Unit::unitCount = {0, 0, 0, 0, 0, 0, 0} [private]
```

Definition at line 19 of file [cppUnit.h](#).

7.59.5.6 userUnits

```
std::vector< std::pair< std::string, std::array< int, 7 > > > Unit::userUnits = { } [static],  
[private]
```

Definition at line 10 of file [cppUnit.h](#).

The documentation for this class was generated from the following files:

- [/home/ychoi/ATOM/pycpp/inc/cppUnit.h](#)
- [/home/ychoi/ATOM/pycpp/src/cppUnit.cpp](#)

Chapter 8

File Documentation

8.1 /home/ychoi/ATOM/alpide/analysis/inc/TAnalyser.h File Reference

The class for controlling ROOT and config file when analysing.

```
#include <string>
#include <filesystem>
#include <unordered_map>
#include <vector>
#include "TFileFormat.h"
```

Classes

- class [TAnalyser](#)
For ROOT and config file when analysis.

Typedefs

- typedef unsigned int [UInt_t](#)

8.1.1 Detailed Description

The class for controlling ROOT and config file when analysing.

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

13-04-2024

Copyright

Copyright (c) 2024

Definition in file [TAnalyser.h](#).

8.1.2 Typedef Documentation

8.1.2.1 UInt_t

typedef unsigned int [UInt_t](#)

Definition at line 67 of file [TAnalyser.h](#).

8.2 TAnalyser.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef __TANALYSER__
00013 #define __TANALYSER__
00014
00015 #ifdef __TANALYSER_HEADERS__
00016 #include <iostream>
00017
00018 #include "TFile.h"
00019 #include "TTree.h"
00020 #include "TBranch.h"
00021 #include "TH1D.h"
00022 #include "TH1I.h"
00023 #include "TH2D.h"
00024 #include "TError.h"
00025 #include "TPaveText.h"
00026 #include "TDirectory.h"
00027 #include "TCanvas.h"
00028 #include "TLine.h"
00029 #include "TGraph.h"
00030 #include "TFl.h"
00031 #include "TLegend.h"
00032
00033 #include "cpptqdm.h"
00034 #include "CppConfigFile.h"
00035
00036 #include "TExperimentData.h"
00037 #include "TMatrix2D.h"
00038 #include "TALPIDEEvent.h"
00039 #include "TCluster.h"
00040 #include "TClusterDivideData.h"
00041 #include "TClusterShape.h"
00042 #endif
00043
00044 #include<string>
00045 #include<filesystem>
00046 #include<unordered_map>
00047 #include<vector>
00048 #include "TFileFormat.h"
00049
00050 class TFile;
00051 class TTree;
00052 class TH2D;
00053 class TH1D;
00054 class TPaveText;
00055
00056 class Configurable;
00057
00058 // struct TInputRoot;
00059 class TALPIDEEvent;
00060 class TExperimentData;
00061 class TClusterDivideData;
00062 class TClusterShape;
00063 class TDirectory;
00064
00065 class CppConfigDictionary;
00066 class TCluster;
00067 typedef unsigned int UInt_t;
00068
00079 class TAnalyser {
00080 private:
00081     TFile* mInputFile = nullptr;
00082     TTree* mTree;
00083     TInputRoot mInput;
00084
00085
00086 protected:

```

```

00087     TFile* mOutputFile = nullptr;
00088     bool mIsOutputGraph = false;
00089     TPaveText* mExpSettingLegend;
00090     std::unordered_map<std::string, TH2D*> mHitmaps;
00091     std::unordered_map<std::string, TH2D*> mClustermaps;
00092     std::unordered_map<std::string, TH1D*> mClustersizes;
00093     std::unordered_map<std::string, std::unordered_map<int, std::vector<TCluster*>>
mClusterDataWithShape;
00094     std::unordered_map<std::string, TDirectory*> mDirectorySet;
00095     std::unordered_map<std::string, TExperimentData*> mExpData;
00096     std::unordered_map<std::string, TClusterDivideData*> mDivideData;
00097     std::unordered_map<std::string, std::vector<TClusterShape*> mClusterShapeSet;
00098     std::unordered_map<std::string, int> mNTotalShapeSet;
00099     std::unordered_map<std::string, int> mMaxModeSet;
00100 public:
00101     TAnalyser() = default;
00102     TAnalyser(TFile* inputFile, std::unordered_map<std::string, TExperimentData*> expData);
00103     ~TAnalyser();
00104
00105     TTree* openTree(std::string treeName);
00106     void storeEvents(CppConfigDictionary settingConfig);
00107     void doMasking(int mMaskOver);
00108     void openOutputGraphFile(std::string_view fileName);
00109     void openDirectory(std::string_view typeName);
00110
00111     // void setSavePath(const std::filesystem::path& savePath);
00112     void setExpSettingLegend(CppConfigDictionary settingConfig);
00113
00114     void doDivideBySize(std::string_view typeName);
00115
00116     TExperimentData* getAnEventSet(std::string_view typeName) const;
00117
00118     TH2D* getHitPlot(const CppConfigDictionary& config, const std::vector<TALPIDEEEvent*>& events);
00119
00120     void saveHitmap(std::string typeName, const CppConfigDictionary& config);
00121
00122     TH2D* getClusterPlot(const CppConfigDictionary& config, const std::vector<TCluster*>& clusters);
00123     TH1D* getClustersizePlot(const CppConfigDictionary& config, const std::vector<TCluster*>&
clusters);
00124     void setClusterDataWithShape(const std::vector<int>& clusterSizeRange);
00125
00126     void saveClustermap(std::string typeName, const CppConfigDictionary& config);
00127     void saveClustersize(std::string typeName, const CppConfigDictionary& config);
00128     std::vector<int> getClusterSizeRange(const CppConfigDictionary& privateProperty);
00129
00130     void doShaping(std::string_view typeName, const std::vector<int>& clusterSizeRange);
00131     void saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config);
00132     void saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config);
00133     void saveSameSizeShapes(std::string_view typeName, const CppConfigDictionary config);
00134     void saveTotalShapes(std::string_view typeName, const CppConfigDictionary config);
00135     void saveSameSizeShapeEntry(std::string_view typeName, const CppConfigDictionary config);
00136     void saveTotalShapeEntry(std::string_view typeName, const CppConfigDictionary config);
00137
00138 private:
00139     UInt_t fBits;
00140 public:
00141     enum {
00142         kNotDeleted = 0x02000000
00143     };
00144     bool IsDestructed() const { return !TestBit(kNotDeleted); }
00145     bool TestBit(UInt_t f) const { return (bool) ((fBits & f) != 0); }
00146 };
00147
00148 #endif

```

8.3 /home/ychoi/ATOM/alpide/analysis/inc/TFileFormat.h File Reference

Classes

- struct [TInputRoot](#)

Typedefs

- typedef unsigned char [UChar_t](#)
- typedef unsigned int [UInt_t](#)
- typedef unsigned short [UShort_t](#)

8.3.1 Typedef Documentation

8.3.1.1 UChar_t

```
typedef unsigned char UChar_t
```

Definition at line 4 of file [TFileFormat.h](#).

8.3.1.2 UInt_t

```
typedef unsigned int UInt_t
```

Definition at line 5 of file [TFileFormat.h](#).

8.3.1.3 UShort_t

```
typedef unsigned short UShort_t
```

Definition at line 6 of file [TFileFormat.h](#).

8.4 TFileFormat.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TFILEFORMAT__
00002 #define __TFILEFORMAT__
00003
00004 typedef unsigned char UChar_t;
00005 typedef unsigned int UInt_t;
00006 typedef unsigned short UShort_t;
00007
00008 struct TInputRoot {
00009     UChar_t chipid;
00010     UInt_t timeStamp;
00011     UShort_t x;
00012     UShort_t y;
00013 };
00014
00015 #endif
```

8.5 /home/ychoi/ATOM/alpide/analysis/inc/TThresholdAnalyser.h File Reference

The tools for threshold analysis.

```
#include <vector>
#include <fstream>
```

Classes

- class [TThresholdAnalyser](#)

8.5.1 Detailed Description

The tools for threshold analysis.

Author

Yongjun Choi (yochoi@cern.ch)

Version

0.1

Date

2024-04-13

Copyright

Copyright (c) 2024

Definition in file [TThresholdAnalyser.h](#).

8.6 TThresholdAnalyser.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef __TTHRESHOLDANALYSER__
00012 #define __TTHRESHOLDANALYSER__
00013
00014 #ifdef __TTHRESHOLDANALYSER_HEADER__
00015 #include <iostream>
00016 #include <sstream>
00017
00018 #include "RtypesCore.h"
00019 #include "TH1D.h"
00020 #include "TH2D.h"
00021 #include "TCanvas.h"
00022
00023 #include "cpptqdm.h"
00024
00025 #include "TThreshold.h"
00026 #endif
00027
00028 #include<vector>
00029 #include<fstream>
00030 // #include "TThreshold.h"
00031 // #include "cpptqdm.h"
00032 class TThreshold;
00033 class TH1;
00034 class TH2;
00035
00036 class TThresholdAnalyser {
00037 private:
00038     std::ifstream mFile;
00039     int mVcasn, mIthr;
00041     std::vector<TThreshold*> mThresholds;
00043     TH1* mThresholdDistribution;
00044     TH1* mErrorDistribution;
00045     TH2* mThresholdmap;
00046     TH1* mChi2NdfDistribution;
00047 public:
00048     TThresholdAnalyser();
00049     TThresholdAnalyser(std::ifstream& file);
00050     ~TThresholdAnalyser();
00051
00052     void openFile(std::ifstream& file);
00053
00054     void refineData();
00055
00056     void saveThresholdDistribution(std::string_view title) const;
00057     void saveErrorDistribution(std::string_view title) const;
00058     void saveThresholdmap(std::string_view title) const;
00059     void saveQualityDistribution(std::string_view title) const;
00060 };
00061
00062 #endif
```

8.7 /home/ychoi/ATOM/alpide/analysis/src/TAnalyser.cpp File Reference

```
#include "TAnalyser.h"
```

Macros

- `#define __TANALYSER_HEADERS__`

Functions

- `int calNIncludePixel` (const `TMatrix2D`< int > *matrix)
- `double calRatioOfRadius` (const `TMatrix2D`< int > *matrix)

8.7.1 Macro Definition Documentation

8.7.1.1 __TANALYSER_HEADERS__

```
#define __TANALYSER_HEADERS__
```

Definition at line 1 of file [TAnalyser.cpp](#).

8.7.2 Function Documentation

8.7.2.1 calNIncludePixel()

```
int calNIncludePixel (  
    const TMatrix2D< int > * matrix )
```

Definition at line 659 of file [TAnalyser.cpp](#).

8.7.2.2 calRatioOfRadius()

```
double calRatioOfRadius (  
    const TMatrix2D< int > * matrix )
```

Definition at line 691 of file [TAnalyser.cpp](#).

8.8 TAnalyser.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TANALYSER_HEADERS__
00002
00003 #include "TAnalyser.h"
00004
00011 TAnalyser::TAnalyser(TFile* inputFile, std::unordered_map<std::string, TExperimentData*> expData) :
    mInputFile(inputFile), mExpData(expData), fBits(kNotDeleted) {
00012     std::filesystem::path inputPath = mInputFile->GetName();
00013     std::clog << "TAnalyser object for \033[1;32m" << inputPath.stem() << "\033[0m is armed" << std::endl;
00018     mTree = openTree("hit");
00019     mTree->SetBranchAddresses("ChipID", &mInput.chipid);
00020     mTree->SetBranchAddresses("TimeStamp", &mInput.timeStamp);
00021     mTree->SetBranchAddresses("X", &mInput.x);
00022     mTree->SetBranchAddresses("Y", &mInput.y);
00023
00024     gErrorIgnoreLevel = kWarning;
00025 }
00026
00031 TAnalyser::~TAnalyser() {
00032     if ( mOutputFile != nullptr && !mOutputFile->IsDestructed() ) {
00033         mOutputFile->Close();
00034         mOutputFile = nullptr;
00035     }
00036     // for ( const auto& pair : mHitmaps ) {
00037     //     if ( !pair.second->IsDestructed() ) {
00038     //         delete pair.second;
00039     //     }
00040     // }
00041     // for ( const auto& pair : mExpData ) {
00042     //     if ( !pair.second->IsDestructed() ) {
00043     //         delete pair.second;
00044     //     }
00045     // }
00046     // if ( mInputFile != nullptr && !mInputFile->IsDestructed() ) {
00047     //     mInputFile->Close();
00048     //     delete mInputFile;
00049     //     mInputFile = nullptr;
00050     // }
00051     // if ( mTree != nullptr && !mTree->IsDestructed() ) {
00052     //     delete mTree;
00053     //     mTree = nullptr;
00054     // }
00055     // if ( mExpSettingLegend != nullptr && !mExpSettingLegend->IsDestructed() ) {
00056     //     delete mExpSettingLegend;
00057     //     mExpSettingLegend = nullptr;
00058     // }
00059     std::clog << "TAnalyser object is terminated" << std::endl;
00060 }
00061
00068 TTree* TAnalyser::openTree(std::string treeName) {
00073     try {
00074         if ( mInputFile->Get(static_cast<TString>(treeName)) == nullptr ) throw(treeName);
00075         else if ( std::string(mInputFile->Get(static_cast<TString>(treeName))>->ClassName()) != "TTree"
00076 ) throw(treeName);
00077         return static_cast<TTree*>(mInputFile->Get(static_cast<TString>(treeName)));
00078     } catch ( std::string treeName ) {
00079         std::cerr << "There are no " << treeName << "TTree in this TFile" << std::endl;
00080         exit(1);
00081     }
00082
00088 void TAnalyser::openOutputGraphFile(std::string_view fileName) {
00089     mOutputFile = new TFile(static_cast<TString>(fileName), "RECREATE");
00090     mIsOutputGraph = true;
00091 }
00092
00098 void TAnalyser::openDirectory(std::string_view typeName) {
00099     mOutputFile->cd();
00100     TDirectory* directory = mOutputFile->mkdir(static_cast<TString>(typeName));
00101     mDirectorySet.insert_or_assign(std::string(typeName), directory);
00102 }
00103
00109 void TAnalyser::storeEvents(CppConfigDictionary config) {
00110     std::clog << "Extracting Events..." << std::endl;
00111
00112     std::vector<TALPIDEEvent*> tempEvents;
00113     UInt_t preTime = 0;
00116     tempEvents.push_back(new TALPIDEEvent());
00117     tempEvents.back()->setEvent(0);
00118     tempEvents.back()->setTime(static_cast<long int>(0));
00119
00120     ProgressBar* pbar = new ProgressBar(mTree->GetEntries());
00122     TH1I* timeStampHist = new TH1I("TSHist", "", 50001, 0, 50000);

```

```

00123
00124     for ( int entry = 0; entry < mTree->GetEntries(); entry++ ) {
00125         pbar->printProgress();
00126         mTree->GetEntry(entry);
00127         if ( mInput.timeStamp == preTime ) {
00128             tempEvents.back()->pushData({mInput.x, mInput.y});
00129         } else {
00130             tempEvents.back()->removeDuplication();
00131             tempEvents.back()->sortPixel();
00132             if ( config.hasKey("time_stamp_cut") ) {
00133                 if ( tempEvents.back()->getNData() > stoi(config.find("time_stamp_cut")) ) {
00134                     tempEvents.pop_back();
00135                 }
00136             }
00137             tempEvents.push_back(new TALPIDEEvent());
00138             tempEvents.back()->setEvent(mInput.timeStamp);
00139             tempEvents.back()->setTime(mInput.timeStamp);
00140             tempEvents.back()->pushData({mInput.x, mInput.y});
00141             preTime = mInput.timeStamp;
00142         }
00143     }
00144     tempEvents.back()->removeDuplication();
00145     tempEvents.back()->sortPixel();
00146
00147     for ( auto& event : tempEvents ) {
00148         if ( event->getNData() > 200000 ) std::cout << event->getNData();
00149         timeStampHist->SetBinContent(event->getEvent(), event->getNData());
00150     }
00151
00152     TCanvas* can = new TCanvas("TCanvas", "", 2000, 1000);
00153     timeStampHist->Draw();
00154     std::filesystem::path filePath(config.find("output_path"));
00155     std::filesystem::create_directories(filePath);
00156     std::filesystem::path file = filePath;
00157     file /= "timeStamp.png";
00158     can->SaveAs(static_cast<TString>(std::string(file)));
00159
00160     TH1D* timeStampHist2 = new TH1D("TSHist2", "", 1000, 0, 1000);
00161     for ( int i = 0; i < 50000; i++ ) {
00162         timeStampHist2->Fill(timeStampHist->GetBinContent(i));
00163     }
00164     TCanvas* can2 = new TCanvas("TCanvas2", "", 2000, 1000);
00165     timeStampHist2->Draw();
00166     std::filesystem::path filePath2(config.find("output_path"));
00167     std::filesystem::create_directories(filePath2);
00168     std::filesystem::path file2 = filePath2;
00169     file2 /= "timeStampHist.png";
00170     can2->SetLogy();
00171     can2->SaveAs(static_cast<TString>(std::string(file2)));
00172     delete pbar;
00173     std::clog << "The " << tempEvents.size() << " of events is extracted from " << mTree->GetEntries() << "
stamps." << std::endl;
00174     mExpData.find("Basic")->second->setEvents(std::move(tempEvents));
00175 }
00176
00177 void TAnalyser::doMasking(int mMaskOver) {
00178     TMatrix2D<int> matrix(1025, 513);
00179     for ( int x = 0; x < 1025; x++ ) {
00180         for ( int y = 0; y < 513; y++ ) {
00181             if ( mHitmaps.find("Basic")->second->GetBinContent(x, y) > mMaskOver ) {
00182                 matrix.setElement(x - 1, y - 1, 1);
00183             }
00184         }
00185     }
00186
00187     std::vector<TALPIDEEvent*> tempEvents;
00188     std::vector<TALPIDEEvent*> noiseEvents;
00189
00190     for ( const TALPIDEEvent* event : mExpData.find("Basic")->second->getEvents() ) {
00191         tempEvents.push_back(new TALPIDEEvent());
00192         tempEvents.back()->setEvent(event->getEvent());
00193         tempEvents.back()->setTime(event->getTime());
00194         noiseEvents.push_back(new TALPIDEEvent());
00195         noiseEvents.back()->setEvent(event->getEvent());
00196         noiseEvents.back()->setTime(event->getTime());
00197         for ( const std::pair<int, int>& pixel : event->getData() ) {
00198             if ( matrix.getElement(pixel.first, pixel.second) != 1 ) {
00199                 tempEvents.back()->pushData(std::move(pixel));
00200             } else {
00201                 noiseEvents.back()->pushData(std::move(pixel));
00202             }
00203         }
00204     }
00205
00206     TExperimentData* maskedData = new TExperimentData();
00207     maskedData->setEvents(std::move(tempEvents));
00208     mExpData.insert_or_assign("Masked", maskedData);

```

```

00209
00210     TExperimentData* noisePixelData = new TExperimentData();
00211     noisePixelData->setEvents(std::move(noiseEvents));
00212     mExpData.insert_or_assign("NoisePixel", noisePixelData);
00213 }
00214
00215 void TAnalyser::setExpSettingLegend(CppConfigDictionary settingConfig) {
00216     mExpSettingLegend = new TPaveText(.78, .1, .981, .65, "NDC");
00217     if ( settingConfig.hasKey("NTRG") ) {
00218         mExpSettingLegend->AddText(static_cast<TString>("NTRG= " + settingConfig.find("NTRG")));
00219     }
00220     mExpSettingLegend->AddText(static_cast<TString>("NEVT= " +
std::to_string(mExpData.find("Basic")->second->getEvents().size())));
00221     if ( settingConfig.hasKey("THRESHOLD") ) {
00222         mExpSettingLegend->AddText(static_cast<TString>("Threshold= " +
settingConfig.find("THRESHOLD")));
00223     }
00224     if ( settingConfig.hasKey("COL_LENGTH") ) {
00225         mExpSettingLegend->AddText(static_cast<TString>("Collimator Length = " +
settingConfig.find("COL_LENGTH")));
00226     }
00227     if ( settingConfig.hasKey("COL_DIAMETER") ) {
00228         mExpSettingLegend->AddText(static_cast<TString>("Collimator Diameter = " +
settingConfig.find("COL_DIAMETER")));
00229     }
00230     if ( settingConfig.hasKey("AIR_PRESSURE") ) {
00231         mExpSettingLegend->AddText(static_cast<TString>("Air Pressure = " +
settingConfig.find("AIR_PRESSURE")));
00232     }
00233
00234     mExpSettingLegend->SetTextAlign(11);
00235     mExpSettingLegend->SetFont(42);
00236 }
00237
00238 TExperimentData* TAnalyser::getAnEventSet(std::string_view typeName) const {
00239     if ( mExpData.count(std::string(typeName)) ) {
00240         return mExpData.find(std::string(typeName))->second;
00241     } else {
00242         return new TExperimentData();
00243     }
00244 }
00245
00246 TH2D* TAnalyser::getHitPlot(const CppConfigDictionary& config, const std::vector<TALPIDEEvent*>&
events) {
00247     // Static variable for numbering
00248     static int iHMap = 0;
00249     // Allocate a 2d histogram.
00250     std::string plotTitle = "";
00251     if ( config.hasKey("title") ) {
00252         plotTitle += config.find("title");
00253     }
00254     if ( config.hasKey("x_title") ) {
00255         plotTitle += "; " + config.find("x_title");
00256     } else {
00257         plotTitle += "; ";
00258     }
00259     if ( config.hasKey("y_title") ) {
00260         plotTitle += "; " + config.find("y_title");
00261     } else {
00262         plotTitle += "; ";
00263     }
00264
00265     TH2D* map = new TH2D(Form("hmap%d", iHMap), static_cast<TString>(plotTitle), 1024, 0, 1024, 512,
0, 512);
00266     // Fill data
00267     for ( const TALPIDEEvent* event : events ) {
00268         for ( const std::pair<int, int>& pixel : event->getData() ) {
00269             map->Fill(pixel.first, pixel.second);
00270         }
00271     }
00272     // Canvas setting
00273     TCanvas* canvas = new TCanvas(Form("hmapCan%d", iHMap), "", 2500, 1000);
00274     canvas->SetMargin(.07, .35, .12, .08);
00275     map->GetXaxis()->SetTitleOffset(1.4);
00276     map->GetXaxis()->SetLabelOffset(0.003);
00277     // Find directory for saving clustermap. If it doesn't exist, then make the directory with mother
directories.
00278     std::filesystem::path filePath(config.find("output_path"));
00279     filePath /= config.find("subdirectory");
00280     std::filesystem::create_directories(filePath);
00281     // Draw plot with options
00282     if ( config.hasKey("options") ) {
00283         for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
00284             map->Draw(static_cast<TString>(optionName));
00285             mExpSettingLegend->Draw("SAME");
00286             std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
00287             if ( config.hasKey("extension") ) {

```

```

00288         file.replace_extension(config.find("extension"));
00289     } else {
00290         file.replace_extension("png");
00291     }
00292     canvas->SaveAs(static_cast<TString>(file));
00293 }
00294 } else {
00295     map->Draw();
00296     mExpSettingLegend->Draw("SAME");
00297     canvas->SaveAs(static_cast<TString>(filePath));
00298 }
00299 // Delete canvas;
00300 delete canvas;
00301 iHMap++;
00302 return map;
00303 }
00304
00305 void TAnalyser::saveHitmap(std::string typeName, const CppConfigDictionary& config) {
00306     std::clog << "Generating \033[1;32m" << typeName << " Hitmap\033[1;0m..." << std::endl;
00307     if ( !mHitmaps.count(typeName) ) {
00308         mHitmaps.insert_or_assign(typeName, getHitPlot(config,
00309 mExpData.find(typeName)->second->getEvents()));
00310         if ( mIsOutputGraph ) {
00311             mDirectorySet.find(std::string(typeName))->second->cd();
00312             mHitmaps.find(std::string(typeName))->second->Write("hitmap");
00313             mOutputFile->cd();
00314         } else {
00315             getHitPlot(config, mExpData.find(typeName)->second->getEvents());
00316         }
00317     }
00318 }
00319 void TAnalyser::doDivideBySize(std::string_view typeName) {
00320     TClusterDivideData* clusterDivideData = new
00321 TClusterDivideData(mExpData.find(std::string(typeName))->second->getClusters());
00322     mDivideData.insert_or_assign(std::string(typeName), clusterDivideData);
00323 }
00324
00325 TH2D* TAnalyser::getClusterPlot(const CppConfigDictionary& config, const std::vector<TCluster*>&
clusters) {
00326     // Static variable for numbering
00327     static int iMap = 0;
00328     // Allocate a 2d histogram.
00329     std::string plotTitle = "";
00330     if ( config.hasKey("title") ) {
00331         plotTitle += config.find("title");
00332     }
00333     if ( config.hasKey("x_title") ) {
00334         plotTitle += "; " + config.find("x_title");
00335     } else {
00336         plotTitle += "; ";
00337     }
00338     if ( config.hasKey("y_title") ) {
00339         plotTitle += "; " + config.find("y_title");
00340     } else {
00341         plotTitle += "; ";
00342     }
00343     TH2D* map = new TH2D(Form("map%d", iMap), static_cast<TString>(plotTitle), 1024, 0, 1024, 512, 0,
512);
00344     // Fill data
00345     std::vector<int> interestSizeSet = getClusterSizeRange(config);
00346     for ( const TCluster* cluster : clusters ) {
00347         if ( !interestSizeSet.empty() ) {
00348             if ( std::find(interestSizeSet.begin(), interestSizeSet.end(), cluster->getSize()) !=
interestSizeSet.end() ) {
00349                 map->Fill(cluster->getCenter().first, cluster->getCenter().second);
00350             }
00351         } else {
00352             map->Fill(cluster->getCenter().first, cluster->getCenter().second);
00353         }
00354     }
00355     // Canvas setting
00356     TCanvas* canvas = new TCanvas(Form("mapCan%d", iMap), "", 2500, 1000);
00357     canvas->SetMargin(.07, .35, .12, .08);
00358     map->GetXaxis()->SetTitleOffset(1.4);
00359     map->GetXaxis()->SetLabelOffset(0.003);
00360     // Find directory for saving clustermap. If it doesn't exist, then make the directory with mother
directories.
00361     std::filesystem::path filePath(config.find("output_path"));
00362     filePath /= config.find("subdirectory");
00363     std::filesystem::create_directories(filePath);
00364
00365     // Draw plot with options
00366     if ( config.hasKey("options") ) {

```

```

00376         for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
00377             map->Draw(static_cast<TString>(optionName));
00378             mExpSettingLegend->Draw("SAME");
00379             std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
00380             if ( config.hasKey("extension") ) {
00381                 file.replace_extension(config.find("extension"));
00382             } else {
00383                 file.replace_extension("png");
00384             }
00385             canvas->SaveAs(static_cast<TString>(file));
00386         }
00387     } else {
00388         map->Draw();
00389         mExpSettingLegend->Draw("SAME");
00390         std::filesystem::path file = filePath / (config.find("filename"));
00391         if ( config.hasKey("extension") ) {
00392             file.replace_extension(config.find("extension"));
00393         } else {
00394             file.replace_extension("png");
00395         }
00396         canvas->SaveAs(static_cast<TString>(file));
00397     }
00398     // Delete canvas;
00399     delete canvas;
00400     iMap++;
00401     return map;
00402 }
00410 TH1D* TAnalyser::getClustersizePlot(const CppConfigDictionary& config, const std::vector<TCluster*>&
clusters) {
00411     static int iDistribution = 0;
00412     TString distName = Form("distribution%d", iDistribution);
00413     TString distTitle = "";
00414     distTitle += config.hasKey("title") ? config.find("title") : "";
00415     distTitle += config.hasKey("x_title") ? " ; " + config.find("x_title") : ";";
00416     distTitle += config.hasKey("y_title") ? " ; " + config.find("y_title") : ";";
00417     Int_t nBins = 0;
00418     Float_t maxBin = 0;
00419     Float_t minBin = 0;
00420     if ( config.hasKey("distribution_info") ) {
00421         nBins = config.getSubConfig("distribution_info").hasKey("nbins") ?
stoi(config.getSubConfig("distribution_info").find("nbins")) : 80;
00422         maxBin = config.getSubConfig("distribution_info").hasKey("max") ?
stoi(config.getSubConfig("distribution_info").find("max")) + .5 : 80.5;
00423         minBin = config.getSubConfig("distribution_info").hasKey("min") ?
stoi(config.getSubConfig("distribution_info").find("min")) - .5 : .5;
00424     } else {
00425         nBins = 80;
00426         maxBin = 80.5;
00427         minBin = 0.5;
00428     }
00429     TH1D* distribution = new TH1D(distName, distTitle, nBins, minBin, maxBin);
00430     for ( const TCluster* cluster : clusters ) {
00431         distribution->Fill(cluster->getSize()); // Fill clustersize to clustersize distribution
00432     }
00433     TCanvas* canvas = new TCanvas(Form("distributionCan%d", iDistribution), "", 2500, 1000);
00434     canvas->SetMargin(.07, .28, .12, .08);
00435     distribution->GetXaxis()->SetTitleOffset(1.4);
00436     distribution->GetXaxis()->SetLabelOffset(0.003);
00437
00438     std::filesystem::path filePath(config.find("output_path"));
00439     filePath /= config.find("subdirectory");
00440     std::filesystem::create_directories(filePath);
00441
00442     if ( config.hasKey("options") ) {
00443         for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
00444             if ( optionName == "logy" ) {
00445                 distribution->Draw();
00446                 mExpSettingLegend->Draw("SAME");
00447                 canvas->SetLogy();
00448                 std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
00449                 if ( config.hasKey("extension") ) {
00450                     file.replace_extension(config.find("extension"));
00451                 } else {
00452                     file.replace_extension("png");
00453                 }
00454                 canvas->SaveAs(static_cast<TString>(file));
00455             } else if ( optionName == "basic" ) {
00456                 distribution->Draw();
00457                 mExpSettingLegend->Draw("SAME");
00458                 std::filesystem::path file = filePath / config.find("filename");
00459                 if ( config.hasKey("extension") ) {
00460                     file.replace_extension(config.find("extension"));
00461                 } else {
00462                     file.replace_extension("png");
00463                 }
00464                 canvas->SetLogy(0);
00465                 canvas->SaveAs(static_cast<TString>(file));

```

```

00466         }
00467     }
00468     } else {
00469         distribution->Draw();
00470         mExpSettingLegend->Draw("SAME");
00471         std::filesystem::path file = filePath / (config.find("filename"));
00472         if ( config.hasKey("extension") ) {
00473             file.replace_extension(config.find("extension"));
00474         } else {
00475             file.replace_extension("png");
00476         }
00477         canvas->SaveAs(static_cast<TString>(file));
00478     }
00479     delete canvas;
00480     iDistribution++;
00481     return distribution;
00482 }
00483
00484 void TAnalyser::saveClustermap(std::string typeName, const CppConfigDictionary& config) {
00485     std::clog << "Generating \033[1;32mClustermap\033[1;0m..." << std::endl;
00486     if ( !mClustermaps.count(typeName) ) {
00487         mClustermaps.insert_or_assign(typeName, getClusterPlot(config,
00488             mExpData.find(typeName)->second->getClusters()));
00489         if ( mIsOutputGraph ) {
00490             mDirectorySet.find(std::string(typeName))->second->cd();
00491             mClustermaps.find(std::string(typeName))->second->Write("clustermap");
00492             mOutputFile->cd();
00493         } else {
00494             getClusterPlot(config, mExpData.find(typeName)->second->getClusters());
00495         }
00496     }
00497
00498 void TAnalyser::saveClustersize(std::string typeName, const CppConfigDictionary& config) {
00499     std::clog << "Generating \033[1;32mCluster Size Distribution\033[1;0m..." << std::endl;
00500     if ( !mClustersizes.count(typeName) ) {
00501         mClustersizes.insert_or_assign(typeName, getClustersizePlot(config,
00502             mExpData.find(typeName)->second->getClusters()));
00503         if ( mIsOutputGraph ) {
00504             mDirectorySet.find(std::string(typeName))->second->cd();
00505             mClustersizes.find(std::string(typeName))->second->Write("clustersize");
00506             mOutputFile->cd();
00507         } else {
00508             getClustersizePlot(config, mExpData.find(typeName)->second->getClusters());
00509         }
00510     }
00511
00512 void TAnalyser::setClusterDataWithShape(const std::vector<int>& clusterSizeRange) {
00513     for ( const int clusterSize : clusterSizeRange ) {
00514         std::vector<TCluster*> clustersWithShape;
00515     }
00516 }
00517
00518 std::vector<int> TAnalyser::getClusterSizeRange(const CppConfigDictionary& privateProperty) {
00519     std::vector<int> clusterSizeRange;
00520     if ( privateProperty.hasKey("interest_size") ) {
00521         for ( const std::string& rangeStr :
00522             privateProperty.getSubConfig("interest_size").getValueList() ) {
00523             if ( rangeStr.find('.') != std::string::npos ) {
00524                 for ( int i = stoi(rangeStr.substr(0, rangeStr.find('.'))); i <
00525                     stoi(rangeStr.substr(rangeStr.find('.') + 3)) + 1; i++ ) {
00526                     clusterSizeRange.push_back(i);
00527                 }
00528             } else {
00529                 clusterSizeRange.push_back(stoi(rangeStr));
00530             }
00531         }
00532     } else {
00533         for ( int i = 0; i < 100; i++ ) {
00534             clusterSizeRange.push_back(i);
00535         }
00536     }
00537     return clusterSizeRange;
00538 }
00539
00540 void TAnalyser::doShaping(std::string_view typeName, const std::vector<int>& clusterSizeRange) {
00541     // The number of total shape and the maximum number of shapes of each cluster sizes.
00542     int nTotalShape = 0;
00543     int maxMode = 0;
00544     // The array of objects of TClusterShape class.
00545     std::vector<TClusterShape*> clusterShapes;
00546     // This stores and extracts clusters shape informations.
00547     for ( const int clusterSize : clusterSizeRange ) {
00548         if ( mClustersizes.find(std::string(typeName))->second->GetBinContent(clusterSize) != 0 ) {
00549             // This gets cluster bunch from TClusterDivideData object which classifies clusters by

```



```

    their size.
00555     const std::vector<TCluster*> divideData =
mDivideData.find(std::string(typeName))->second->getClusterOfSize(clusterSize);
00556     // This inserts cluster size information and clusters to TClusterShape objects.
00557     TClusterShape* clusterShape = new TClusterShape(clusterSize, divideData);
00558     // This extracts shape informations.
00559     clusterShape->identifyShapes();
00560     // This sorts shapes according to its size of long axis.
00561     clusterShape->sortShapes();
00562     //
00563     clusterShapes.push_back(clusterShape);
00564     nTotalShape += clusterShape->getClusterShapeInfos().size();
00565     maxMode = std::max(maxMode,
static_cast<int>(clusterShape->getClusterShapeInfos().size()));
00566 }
00567 }
00568 mClusterShapeSet.insert_or_assign(std::string(typeName), clusterShapes);
00569 mNTotalShapeSet.insert_or_assign(std::string(typeName), nTotalShape);
00570 mMaxModeSet.insert_or_assign(std::string(typeName), maxMode);
00571 }
00572 void TAnalyser::saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config) {
00573     // Print log message
00574     std::clog << "Generating " << "\033[1;32m" << "individual shapes" << "\033[1;0m" << "..." << std::endl;
00575
00576     // Get the total number of shapes for progress bar.
00577     int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;
00578
00579     // Get save path from config file.
00580     std::filesystem::path filePath(config.find("output_path"));
00581     filePath /= config.find("subdirectory");
00582     std::filesystem::create_directories(filePath);
00583
00584     // Init progress bar
00585     ProgressBar pBar(nTotalShape);
00586
00587     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00588         int clusterSize = clusterShape->getClusterSize(); // The cluster size of shape set.
00589         int iClusterShape = 0; // The i-th cluster shape
00590         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00591             pBar.printProgress(); // Print progress bar.
00592
00593             int shapeWidth = shapeInfo.mClusterMatrix->getNRow(); // Shape width of matrix
00594             int shapeHeight = shapeInfo.mClusterMatrix->getNColumn(); // Shape height of matrix
00595
00596             // Initialize canvas
00597             TCanvas* canvas = new TCanvas(Form("%d_%d.png", clusterSize, iClusterShape), "",
(shapeWidth + 2) * 500, (shapeHeight + 2) * 500);
00598             // Set canvas margin
00599             canvas->SetMargin(0., 0., 0., 0.);
00600             // Draw shape on canvas
00601             shapeInfo.mClusterMap->Draw();
00602             // Fill colour
00603             shapeInfo.mClusterMap->SetDrawOption("COLZ");
00604             // Remove histogram axis
00605             shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
00606             shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
00607
00608             // Fill line for separating pixels
00609             TLine* line = new TLine();
00610             line->SetLineColorAlpha(kRed, 6. / 8);
00611             for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
00612                 for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
00613                     if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
00614                         double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
00615                         double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
00616                         double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
00617                         double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
00618                         line->DrawLine(xlow, ylow, xup, ylow); // Bottom
00619                         line->DrawLine(xlow, yup, xup, yup); // Top
00620                         line->DrawLine(xlow, ylow, xlow, yup); // Left
00621                         line->DrawLine(xup, ylow, xup, yup); // Right
00622                     }
00623                 }
00624             }
00625
00626             // Set title of shape
00627             TText* title = new TText(.5, 1. - .5 / (shapeHeight + 2), Form("%d-th Cluster Shape of
Cluster Size %d", iClusterShape, clusterSize));
00628             title->SetTextAlign(23);
00629             title->SetNDC();
00630             title->Draw();
00631
00632             // Save shape figure
00633             std::filesystem::path file = filePath / (config.find("filename") + "_" +
std::to_string(clusterSize) + "_" + std::to_string(iClusterShape));
00634             if ( config.hasKey("extension") ) {
00635                 file.replace_extension(config.find("extension"));
00636             }
00637         }
00638     }
00639 }

```

```

00642         } else {
00643             file.replace_extension("png");
00644         }
00645         canvas->SaveAs(static_cast<TString>(file));
00646
00647         iClusterShape++;
00648
00649         delete title;
00650         title = nullptr;
00651         delete line;
00652         line = nullptr;
00653         delete canvas;
00654         canvas = nullptr;
00655     }
00656 }
00657 }
00658
00659 int calNIncludePixel(const TMatrix2D<int>* matrix) {
00660     int centreX = 0, centreY = 0;
00661     int clusterSize = 0;
00662     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00663         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00664             if ( matrix->getElement(pixelX, pixelY) == 1 ) {
00665                 centreX += pixelX * 2;
00666                 centreY += pixelY * 2;
00667                 clusterSize++;
00668             }
00669         }
00670     }
00671     int radiusSquare = 0;
00672     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00673         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00674             if ( matrix->getElement(pixelX, pixelY) == 1 ) {
00675                 int distance = pow(abs(2 * pixelX * clusterSize - centreX) + clusterSize, 2) +
pow(abs(2 * pixelY * clusterSize - centreY) + clusterSize, 2);
00676                 radiusSquare = std::max(radiusSquare, distance);
00677             }
00678         }
00679     }
00680     int count = 0;
00681     for ( int x = floor((centreX - sqrt(radiusSquare)) / (2 * clusterSize)); x < ceil((centreX +
sqrt(radiusSquare)) / (2 * clusterSize)) + 1; x++ ) {
00682         for ( int y = floor((centreY - sqrt(radiusSquare)) / (2 * clusterSize)); y < ceil((centreY +
sqrt(radiusSquare)) / (2 * clusterSize)) + 1; y++ ) {
00683             if ( static_cast<int>(std::pow(abs(2 * x * clusterSize - centreX) + clusterSize, 2)) +
static_cast<int>(std::pow(abs(2 * y * clusterSize - centreY) + clusterSize, 2)) <= radiusSquare ) {
00684                 count++;
00685             }
00686         }
00687     }
00688     return count;
00689 }
00690
00691 double calRatioOfRadius(const TMatrix2D<int>* matrix) {
00692     int centreX = 0, centreY = 0;
00693     int clusterSize = 0;
00694     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00695         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00696             if ( matrix->getElement(pixelX, pixelY) == 1 ) {
00697                 centreX += pixelX * 2;
00698                 centreY += pixelY * 2;
00699                 clusterSize++;
00700             }
00701         }
00702     }
00703     int longRadiusSquare = 0;
00704     int shortRadiusSquare = 100000000;
00705     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00706         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00707             bool hasValue = matrix->getElement(pixelX, pixelY) == 1;
00708             bool isWorldBorder = pixelX == 0 || pixelY == 0 || pixelX == matrix->getNRow() - 1 ||
pixelY == matrix->getNColumn() - 1;
00709             bool isBorder = isWorldBorder ? true : matrix->getElement(pixelX - 1, pixelY) == 0 ||
matrix->getElement(pixelX, pixelY - 1) == 0 || matrix->getElement(pixelX + 1, pixelY) == 0 ||
matrix->getElement(pixelX, pixelY + 1) == 0;
00710             if ( hasValue ) {
00711                 int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY *
clusterSize - centreY, 2);
00712                 longRadiusSquare = std::max(longRadiusSquare, distance);
00713             }
00714             if ( (hasValue && isWorldBorder) || (hasValue && !isWorldBorder && isBorder) ) {
00715                 int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY *
clusterSize - centreY, 2);
00716                 shortRadiusSquare = std::min(shortRadiusSquare, distance);
00717             }
00718         }
00719     }

```

```

00720     if ( matrix->getNRow() == 1 || matrix->getNColumn() == 1 ) {
00721         shortRadiusSquare = 0;
00722     }
00723     return sqrt(static_cast<double>(shortRadiusSquare) / longRadiusSquare);
00724 }
00725
00732 void TAnalyser::saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config) {
00733     // Print out a log
00734     std::cout << "Generating " << "\033[1;32m" << "Informations of shapes with same size" << "\033[1;0m" <<
    "... " << std::endl;
00735
00736     // Setting output file path
00737     std::filesystem::path filePath(config.find("output_path"));
00738     filePath /= config.find("subdirectory");
00739     // The creation of directories of output path
00740     std::filesystem::create_directories(filePath);
00741     // Call configs. First element is config name and second one is config values.
00742     std::unordered_map<std::string, CppConfigDictionary> plotConfigList =
    config.getSubConfig("ratio_distribution").getSubConfigSetWithName();
00743     // Call histograms
00744     std::unordered_map<std::string, TH1D*> distributionSet;
00745     // Set the information of histograms from config file
00746     for ( const auto& plotConfig : plotConfigList ) {
00747         // Plot name is config name. It isn't value of "name" key.
00748         std::string plotName = plotConfig.second.find("name");
00749         // Set plot title and x, y label.
00750         TString plotTitle = plotConfig.second.hasKey("title") ? plotConfig.second.find("title") : "";
00751         TString plotXTitle = plotConfig.second.hasKey("x_title") ? plotConfig.second.find("x_title") :
    "";
00752         TString plotYTitle = plotConfig.second.hasKey("y_title") ? plotConfig.second.find("y_title") :
    "";
00753         // Set number of bin and min and max of x-direction range.
00754         Int_t plotNBin = plotConfig.second.hasKey("n_bin") ? stoi(plotConfig.second.find("n_bin")) :
    100;
00755         Float_t plotXMin = plotConfig.second.hasKey("x_min") ? stof(plotConfig.second.find("x_min")) :
    0.;
00756         Float_t plotXMax = plotConfig.second.hasKey("x_max") ? stof(plotConfig.second.find("x_max")) :
    1.;
00757
00758         // Add histograms to map.
00759         distributionSet.insert_or_assign(plotConfig.first, new TH1D(static_cast<TString>(plotName),
    plotTitle + "; " + plotXTitle + "; " + plotYTitle, plotNBin, plotXMin, plotXMax));
00760     }
00761
00762     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
00763     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00764         // Get cluster size.
00765         int clusterSize = clusterShape->getClusterSize();
00766         if ( clusterSize < stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_min"))
    || clusterSize > stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_max")) ) {
00767             continue;
00768         }
00769         // Initialize canvas.
00770         TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
00771         // int binNum = clusterShape->getClusterShapeInfos().size();
00772         TGraph* areaRatioGraph = new TGraph();
00773         TGraph* radiusRatioGraph = new TGraph();
00774
00775         int iShape = 0;
00776         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00777             // pBar.printProgress();
00778             for ( const auto& plotConfig : plotConfigList ) {
00779                 if ( plotConfig.second.find("name") == "area_ratio" ) {
00780                     distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) /
    calNIncludePixel(shapeInfo.mClusterMatrix));
00781                 }
00782                 if ( plotConfig.second.find("name") == "area_ratio_with_entry" ) {
00783                     distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) /
    calNIncludePixel(shapeInfo.mClusterMatrix), shapeInfo.mEntry);
00784                 }
00785                 if ( plotConfig.second.find("name") == "radius_ratio" ) {
00786                     distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix));
00787                 }
00788                 if ( plotConfig.second.find("name") == "radius_ratio_with_entry" ) {
00789                     distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix),
    shapeInfo.mEntry);
00790                 }
00791             }
00792
00793             areaRatioGraph->SetPoint(iShape, iShape, static_cast<double>(clusterSize) /
    calNIncludePixel(shapeInfo.mClusterMatrix));
00794             radiusRatioGraph->SetPoint(iShape, iShape, calRatioOfRadius(shapeInfo.mClusterMatrix));
00795             // std::cout << clusterSize << "\t" << iShape << "\t" << static_cast<double>(clusterSize) /

```

```

        calNIncludePixel(shapeInfo.mClusterMatrix) << "\t" << calRatioOfRadius(shapeInfo.mClusterMatrix) << "\t"
        << shapeInfo.mEntry << std::endl;
00796         iShape++;
00797     }
00798     TF1* line1 = new TF1("line1", "0.8", 0, iShape);
00799     TF1* line2 = new TF1("line2", "0.5", 0, iShape);
00800
00801     areaRatioGraph->SetTitle(Form("Informations for cluster shapes in cluster size %d",
clusterSize));
00802     areaRatioGraph->GetXaxis()->SetTitle("i-th cluster shape");
00803     areaRatioGraph->GetXaxis()->SetTitleOffset(1.4);
00804     areaRatioGraph->GetXaxis()->SetLabelOffset(0.003);
00805     areaRatioGraph->GetYaxis()->SetTitle("Ratio");
00806     areaRatioGraph->GetYaxis()->SetTitleOffset(0.7);
00807     areaRatioGraph->SetMarkerStyle(45);
00808     areaRatioGraph->SetMarkerSize(4);
00809     areaRatioGraph->SetMarkerColor(kRed);
00810     areaRatioGraph->SetMaximum(1);
00811     areaRatioGraph->SetMinimum(0);
00812     areaRatioGraph->Draw("AP");
00813     radiusRatioGraph->SetMarkerStyle(41);
00814     radiusRatioGraph->SetMarkerSize(4);
00815     radiusRatioGraph->SetMarkerColor(kBlue);
00816     radiusRatioGraph->Draw("P");
00817     mExpSettingLegend->Draw("SAME");
00818     line1->SetLineColor(kPink + 2);
00819     // line1->Draw("SAME");
00820     line2->SetLineColor(kCyan - 7);
00821     // line2->Draw("SAME");
00822     TLegend* legend = new TLegend(.78, .7, .98, .92);
00823     legend->SetHeader("Ratios", "c");
00824     legend->AddEntry(areaRatioGraph, "Pixels / Full pixels", "p");
00825     legend->AddEntry(radiusRatioGraph, "Short radius / Long radius", "p");
00826     legend->Draw();
00827     canvas->SetMargin(.07, .28, .12, .08);
00828     std::filesystem::path file = filePath;
00829
00830     file /= config.getSubConfig("graphs").find("filename") + "_" + std::to_string(clusterSize);
00831     file.replace_extension(config.find("extension"));
00832
00833     canvas->SaveAs(static_cast<TString>(file));
00834
00835     delete line1;
00836     delete line2;
00837     delete areaRatioGraph;
00838     delete radiusRatioGraph;
00839     delete canvas;
00840 }
00841
00842 for ( const auto& plotConfig : plotConfigList ) {
00843     TString canvasName = "can" + plotConfig.first;
00844     Int_t canvasWidth = plotConfig.second.hasKey("canvas_width") ?
stoi(plotConfig.second.find("canvas_width")) : 500;
00845     Int_t canvasHeight = plotConfig.second.hasKey("canvas_height") ?
stoi(plotConfig.second.find("canvas_height")) : 500;
00846     TCanvas* canvas = new TCanvas(canvasName, "", canvasWidth, canvasHeight);
00847
00848     distributionSet.find(plotConfig.first)->second->SetStats(0);
00849
00850     Int_t y_min = plotConfig.second.hasKey("y_min") ? stoi(plotConfig.second.find("y_min")) : 0;
00851     Int_t y_max = plotConfig.second.hasKey("y_max") ? stoi(plotConfig.second.find("y_max")) :
1000;
00852     distributionSet.find(plotConfig.first)->second->SetMinimum(y_min);
00853     distributionSet.find(plotConfig.first)->second->SetMaximum(y_max);
00854     distributionSet.find(plotConfig.first)->second->Draw();
00855
00856     std::filesystem::path file = filePath;
00857     file /= plotConfig.second.find("filename");
00858     file.replace_extension(config.find("extension"));
00859     canvas->SaveAs(static_cast<TString>(file));
00860
00861     delete canvas;
00862     canvas = nullptr;
00863 }
00864 }
00865
00866 void TAnalyser::saveSameSizeShapes(std::string_view typeName, const CppConfigDictionary config) {
00867     std::cout << "Generating " << "\033[1;32m" << "Shapes with same sized" << "\033[1;0m" << "..." <<
std::endl;
00868
00869     std::filesystem::path filePath(config.find("output_path"));
00870     if ( config.hasKey("subdirectory") ) {
00871         filePath /= config.find("subdirectory");
00872     }
00873     std::filesystem::create_directories(filePath);
00874
00875     int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;

```

```

00876
00877     const int nominalWidth = 200;
00878     const int nominalHeader = 100;
00879     const int nPlotInRow = 10;
00880     int plotsWidth = nPlotInRow * nominalWidth;
00881
00882     ProgressBar pBar(nTotalShape);
00883     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))>second ) {
00884         int clusterSize = clusterShape->getClusterSize();
00885         int nClusterShape = clusterShape->getClusterShapeInfos().size();
00886
00887         int plotsHeight = (floor((nClusterShape - 1.) / 10.) + 1.) * nominalWidth;
00888
00889         TCanvas* canvas = new TCanvas(Form("shapes%d", clusterSize), "", plotsWidth, plotsHeight +
nominalHeader);
00890
00891         int iClusterShape = 0;
00892
00893         std::vector<TPad*> padSet;
00894         std::vector<TLine*> lineSet;
00895         std::vector<TText*> textSet;
00896
00897         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00898             pBar.printProgress();
00899
00900             int width = shapeInfo.mClusterMatrix->getNRow() + 2;
00901             int height = shapeInfo.mClusterMatrix->getNColumn() + 2;
00902
00903             double padCenterX = (1. / nPlotInRow) * (.5 + (iClusterShape % 10));
00904             double padCenterY = (static_cast<double>(plotsHeight) / static_cast<double>(plotsHeight +
nominalHeader)) / ((floor((nClusterShape - 1) / static_cast<double>(nPlotInRow))) + 1.) * .5 * (2. *
(floor((nClusterShape - 1) / static_cast<double>(nPlotInRow)) - floor(iClusterShape /
static_cast<double>(nPlotInRow))) + 1.);
00905
00906             double padWidth = (1. / (2. * nPlotInRow));
00907             double padHeight = (static_cast<double>(plotsHeight) / (plotsHeight + nominalHeader)) /
(2. * ((floor(static_cast<double>(nClusterShape) / nPlotInRow) + 1.)));
00908
00909             TPad* pad = new TPad(Form("pad%d", iClusterShape), "", padCenterX - padWidth, padCenterY -
padHeight, padCenterX + padWidth, padCenterY + padHeight);
00910             pad->SetMargin(0., 0., .5 * (1. - static_cast<double>(height) / width), .5 * (1. -
static_cast<double>(height) / width));
00911             pad->Draw();
00912             pad->cd();
00913             shapeInfo.mClusterMap->SetDrawOption("COL");
00914             shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
00915             shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
00916             shapeInfo.mClusterMap->Draw("col");
00917             pad->SetFrameLineWidth(0);
00918             padSet.push_back(pad);
00919
00920             TLine* line = new TLine();
00921             line->SetLineColorAlpha(kRed, 6. / 8);
00922             for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
00923                 for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
00924                     if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
00925                         double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
00926                         double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
00927                         double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
00928                         double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
00929                         line->DrawLine(xlow, ylow, xup, ylow); // Bottom
00930                         line->DrawLine(xlow, yup, xup, yup); // Top
00931                         line->DrawLine(xlow, ylow, xlow, yup); // Left
00932                         line->DrawLine(xup, ylow, xup, yup); // Right
00933                     }
00934                 }
00935             }
00936             lineSet.push_back(line);
00937
00938             TText* numberingText = new TText(padCenterX, padCenterY, Form("%d", iClusterShape));
00939             numberingText->SetNDC();
00940             numberingText->SetTextAlign(22);
00941             numberingText->SetTextSize(.3 * nominalWidth / (plotsHeight + nominalHeader));
00942             numberingText->SetTextColorAlpha(kBlack, 5. / 8.);
00943
00944             canvas->cd();
00945             numberingText->Draw();
00946             textSet.push_back(numberingText);
00947             iClusterShape++;
00948         }
00949         TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight),
Form("Cluster shapes with cluster size %d", clusterSize));
00950         titleText->SetTextSize(.5 * nominalHeader / (plotsHeight + nominalHeader));
00951         titleText->SetTextAlign(22);
00952         titleText->SetNDC();
00953         titleText->Draw();
00954         std::filesystem::path file = filePath / (config.find("filename") + "_" +

```

```

        std::to_string(clusterSize));
00955         file.replace_extension(config.find("extension"));
00956         canvas->SaveAs(static_cast<TString>(file));
00957
00958         delete titleText;
00959         titleText = nullptr;
00960
00961         for ( TPad* pad : padSet ) {
00962             delete pad;
00963             pad = nullptr;
00964         }
00965
00966         for ( TLine* line : lineSet ) {
00967             delete line;
00968             line = nullptr;
00969         }
00970
00971         for ( TText* text : textSet ) {
00972             delete text;
00973             text = nullptr;
00974         }
00975
00976         delete canvas;
00977         canvas = nullptr;
00978     }
00979 }
00980
00981 void TAnalyser::saveTotalShapes(std::string_view typeName, const CppConfigDictionary config) {
00982     // Print log
00983     std::cout << "Generating \033[1;32mTotal shapes\033[1;0m..." << std::endl;
00984
00985     std::filesystem::path filePath(config.find("output_path"));
00986     if ( config.hasKey("subdirectory") ) {
00987         filePath /= config.find("subdirectory");
00988     }
00989     std::filesystem::create_directories(filePath);
00990
00991     int nWidth = mMaxModeSet.find(std::string(typeName))->second;
00992     int nHeight = mClusterShapeSet.find(std::string(typeName))->second.size();
00993
00994     int nominalWidth = 100;
00995     int nominalHeader = 100;
00996
00997     int plotsWidth = nWidth * nominalWidth;
00998     int plotsHeight = nHeight * nominalWidth;
00999     TCanvas* canvas = new TCanvas("tShape", "cluster shape", plotsWidth + nominalHeader, plotsHeight +
nominalHeader);
01000     int iClusterSize = 0;
01001     std::vector<TPad*> padSet;
01002     std::vector<TLine*> lineSet;
01003     std::vector<TText*> textSet;
01004
01005     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
01006     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
01007         int clusterSize = clusterShape->getClusterSize();
01008         int iClusterShape = 0;
01009         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
01010             pBar.printProgress();
01011             int width = shapeInfo.mClusterMap->GetNbinsX();
01012             int height = shapeInfo.mClusterMap->GetNbinsY();
01013
01014             TPad* pad = new TPad(Form("pad%d_%d", iClusterShape, iClusterShape), "pad", (double)
nominalHeader / (plotsWidth + nominalHeader) + ((double) plotsWidth / (plotsWidth + nominalHeader)) *
((double) iClusterShape / nWidth), ((double) plotsHeight / (plotsHeight + nominalHeader)) * ((double)
nHeight - iClusterShape - 1) / nHeight, (double) nominalHeader / (plotsWidth + nominalHeader) +
((double) plotsWidth / (plotsWidth + nominalHeader)) * (double) (iClusterShape + 1) / nWidth,
((double) plotsHeight / (plotsHeight + nominalHeader)) * ((double) nHeight - iClusterShape) /
nHeight, -1, 1);
01015             padSet.push_back(pad);
01016             pad->Draw();
01017             pad->cd();
01018             pad->SetMargin(0., 0., .5 * (1. - (double) height / width), .5 * (1. - (double) height /
width));
01019
01020             shapeInfo.mClusterMap->Draw();
01021             shapeInfo.mClusterMap->SetDrawOption("COL");
01022             shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
01023             shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
01024             TLine* line = new TLine();
01025             lineSet.push_back(line);
01026             line->SetLineColorAlpha(kRed, 6. / 8);
01027             for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
01028                 for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
01029                     if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
01030                         double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
01031                         double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
01032                         double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);

```

```

01033         double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
01034         line->DrawLine(xlow, ylow, xup, ylow); // Bottom
01035         line->DrawLine(xlow, yup, xup, yup); // Top
01036         line->DrawLine(xlow, ylow, xlow, yup); // Left
01037         line->DrawLine(xup, ylow, xup, yup); // Right
01038     }
01039     }
01040     }
01041     pad->SetFrameLineWidth(0);
01042     canvas->cd();
01043     TText* numberingText = new TText((double) nominalHeader / (plotsWidth + nominalHeader) +
((double) plotsWidth / (plotsWidth + nominalHeader)) * ((double) iClusterShape + .5) / nWidth),
((double) plotsHeight / (plotsHeight + nominalHeader)) * ((double) nHeight - iClusterSize) /
nHeight), Form("%d", shapeInfo.mEntry));
01044     textSet.push_back(numberingText);
01045     numberingText->SetNDC();
01046     numberingText->SetTextAlign(22);
01047     numberingText->SetTextSize(.4 * nominalWidth / (plotsHeight + nominalHeader));
01048     numberingText->SetTextColor(kBlack);
01049     numberingText->Draw();
01050     iClusterShape++;
01051 }
01052 TText* sizeText = new TText((double) nominalHeader * .5 / (nominalHeader + plotsWidth),
((double) plotsHeight / (plotsHeight + nominalHeader)) * ((double) nHeight - iClusterSize - .5) /
nHeight), Form("%d", clusterSize));
01053 textSet.push_back(sizeText);
01054 sizeText->SetNDC();
01055 sizeText->SetTextAlign(22);
01056 sizeText->SetTextSize(.6 * nominalWidth / (plotsHeight + nominalHeader));
01057 sizeText->SetTextColor(kBlack);
01058 sizeText->Draw();
01059 iClusterSize++;
01060 }
01061 TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight),
static_cast<TString>("Total Cluster Shapes"));
01062 titleText->SetTextSize(.8 * nominalHeader / (plotsHeight + nominalHeader));
01063 titleText->SetTextAlign(22);
01064 titleText->SetNDC();
01065 titleText->Draw();
01066
01067 std::filesystem::path file = filePath / config.find("filename");
01068 file.replace_extension(config.find("extension"));
01069 canvas->SaveAs(static_cast<TString>(file));
01070
01071 delete titleText;
01072 titleText = nullptr;
01073
01074 for ( TPad* pad : padSet ) {
01075     delete pad;
01076     pad = nullptr;
01077 }
01078
01079 for ( TLine* line : lineSet ) {
01080     delete line;
01081     line = nullptr;
01082 }
01083
01084 for ( TText* text : textSet ) {
01085     delete text;
01086     text = nullptr;
01087 }
01088
01089 delete canvas;
01090 canvas = nullptr;
01091 }
01092
01093 void TAnalyser::saveSameSizeShapeEntry(std::string_view typeName, const CppConfigDictionary config) {
01094     std::cout << "Generating \033[1;32mEntry of shapes with same size\033[1;0m..." << std::endl;
01095
01096     std::filesystem::path filePath(config.find("output_path"));
01097     if ( config.HasKey("subdirectory") ) {
01098         filePath /= config.find("subdirectory");
01099     }
01100     std::filesystem::create_directories(filePath);
01101
01102     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
01103     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
01104         int clusterSize = clusterShape->getClusterSize();
01105         TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
01106         int binNum = clusterShape->getClusterShapeInfos().size();
01107         TH1D* distribution = new TH1D(Form("shapeEntry%d", clusterSize),
Form(static_cast<TString>("Shape Entry with cluster size %d"), clusterSize), binNum, -.5, binNum -
.5);
01108         int iShape = 0;
01109         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
01110             pBar.printProgress();
01111             distribution->Fill(iShape, shapeInfo.mEntry);

```



```

01112         iShape++;
01113     }
01114     distribution->GetXaxis()->SetNdivisions(10, 10, 0, true);
01115     distribution->GetXaxis()->SetTitle("i-th cluster size");
01116     distribution->GetXaxis()->SetTitleOffset(1.4);
01117     distribution->GetXaxis()->SetLabelOffset(0.003);
01118     distribution->GetYaxis()->SetTitle("Entry");
01119     distribution->GetYaxis()->SetTitleOffset(0.7);
01120     distribution->Draw("HIST");
01121     mExpSettingLegend->Draw("SAME");
01122     canvas->SetMargin(.07, .28, .12, .08);
01123
01124     std::filesystem::path file = filePath / (config.find("filename") + "_" +
std::to_string(clusterSize));
01125     file.replace_extension(config.find("extension"));
01126     canvas->SaveAs(static_cast<TString>(file));
01127
01128     delete distribution;
01129     delete canvas;
01130 }
01131 }
01132
01133 void TAnalyser::saveTotalShapeEntry(std::string_view typeName, const CppConfigDictionary config) {
01134     std::cout << "Generating \033[1;32mEntry of total shapes\033[1;0m..." << std::endl;
01135
01136     std::filesystem::path filePath(config.find("output_path"));
01137     if ( config.hasKey("subdirectory") ) {
01138         filePath /= config.find("subdirectory");
01139     }
01140     std::filesystem::create_directories(filePath);
01141
01142     int nXbin = 0;
01143     int nYbin = 0;
01144     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
01145         nYbin = std::max(nYbin, static_cast<int>(clusterShape->getClusterShapeInfos().size()));
01146         nXbin = std::max(nXbin, clusterShape->getClusterSize());
01147     }
01148     TCanvas* canvas = new TCanvas("shapeEntryTotal", "shape entry", 2500, 1000);
01149     TH2D* distribution = new TH2D("shapeEntryTotal", static_cast<TString>(config.find("plot_titles")),
nXbin, -nXbin - .5, -.5, nYbin, -nYbin + .5, .5);
01150     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
01151     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
01152         int clusterSize = clusterShape->getClusterSize();
01153         int iShape = 0;
01154         for ( const TShapeInfo shapeInfo : clusterShape->getClusterShapeInfos() ) {
01155             pBar.printProgress();
01156             distribution->Fill(-clusterSize, -iShape, shapeInfo.mEntry);
01157             iShape++;
01158         }
01159     }
01160     for ( int iXbin = 1; iXbin < distribution->GetNbinsX(); ++iXbin ) {
01161         distribution->GetXaxis()->SetBinLabel(iXbin, Form("%g", floor(-1 *
distribution->GetXaxis()->GetBinLowEdge(iXbin))));
01162     }
01163
01164     for ( int iYbin = 1; iYbin < distribution->GetNbinsY(); ++iYbin ) {
01165         distribution->GetYaxis()->SetBinLabel(iYbin, Form("%g", floor(-1 *
distribution->GetYaxis()->GetBinLowEdge(iYbin))));
01166     }
01167
01168     distribution->SetStats(0);
01169     if ( config.hasKey("options") ) {
01170         for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
01171             distribution->Draw(static_cast<TString>(optionName));
01172             canvas->SetPhi(10);
01173             canvas->SetTheta(25);
01174             canvas->SetLogz();
01175             std::filesystem::path file = filePath;
01176             file /= (config.find("filename") + "_" + optionName);
01177             file.replace_extension(config.find("extension"));
01178             canvas->SaveAs(static_cast<TString>(file));
01179         }
01180     } else {
01181         distribution->Draw();
01182         std::filesystem::path file = filePath;
01183         file /= config.find("filename");
01184         file.replace_extension(config.find("extension"));
01185         canvas->SaveAs(static_cast<TString>(file));
01186     }
01187
01188     delete distribution;
01189     delete canvas;
01190 }

```


8.9 /home/ychoi/ATOM/alpide/analysis/src/TThresholdAnalyser.cpp File Reference

```
#include "TThresholdAnalyser.h"
```

Macros

- `#define __TTHRESHOLDANALYSER_HEADER__`

8.9.1 Macro Definition Documentation

8.9.1.1 __TTHRESHOLDANALYSER_HEADER__

```
#define __TTHRESHOLDANALYSER_HEADER__
```

Definition at line 1 of file [TThresholdAnalyser.cpp](#).

8.10 TThresholdAnalyser.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TTHRESHOLDANALYSER_HEADER__
00002
00003 #include "TThresholdAnalyser.h"
00004
00005 TThresholdAnalyser::TThresholdAnalyser() { }
00006
00007 TThresholdAnalyser::TThresholdAnalyser(std::ifstream& file) {
00008     openFile(file);
00009 }
00010
00011 TThresholdAnalyser::~TThresholdAnalyser() {
00012     std::cout << "The threshold analyser is terminated" << std::endl;
00013 }
00014
00015 void TThresholdAnalyser::openFile(std::ifstream& file) {
00016     std::string line;
00017     std::array<Int_t, 50> dacs;
00018     std::streampos originalPos = file.tellg();
00019     int numLines = 0;
00020     char ch;
00021     while ( file.get(ch) ) {
00022         if ( ch == '\n' ) {
00023             ++numLines;
00024         }
00025     }
00026     file.clear();
00027     file.seekg(originalPos);
00028     ProgressBar bar(numLines + 1);
00029     while ( file ) {
00030         bar.printProgress();
00031         getline(file, line);
00032         std::stringstream values(line);
00033         int x, y, iDac, dac;
00034         values >> y >> x >> iDac >> dac;
00035         dacs[iDac] = dac;
00036         if ( iDac == 49 ) {
00037             TThreshold* temp(new TThreshold(x, y, dacs));
00038             mThresholds.push_back(temp);
00039         }
00040     }
00041 }
00042
00043 void TThresholdAnalyser::refineData() {
00044     mThresholdDistribution = new TH1D("thrDist", "Threshold Distribution", 500, 0, 50);
00045     mErrorDistribution = new TH1D("errDist", "Error Distribution", 300, 0, 30);
```

```

00046     mThresholdmap = new TH2D("thrmap", "Threshold map", 1024, -0.5, 1023.5, 512, -0.5, 511.5);
00047     mChi2NdfDistribution = new TH1D("chi2", "Chi2 and NDoF", 200, 0, 200);
00048     int nHigh = 0;
00049     int nLow = 0;
00050     int nUndefined = 0;
00051     for ( const TThreshold* threshold : mThresholds ) {
00052         if ( threshold->getCondition() == ThrCondition::good ) {
00053             mThresholdDistribution->Fill(threshold->getThreshold());
00054             mErrorDistribution->Fill(threshold->getError());
00055             mThresholdmap->SetBinContent(threshold->getX(), threshold->getY(),
threshold->getThreshold());
00056             mChi2NdfDistribution->Fill(threshold->getQualityFactor());
00057         } else if ( threshold->getCondition() == ThrCondition::bad_too_high ) {
00058             nHigh++;
00059             mThresholdmap->SetBinContent(threshold->getX(), threshold->getY(), 50);
00060         } else if ( threshold->getCondition() == ThrCondition::bad_too_low ) {
00061             nLow++;
00062             mThresholdmap->SetBinContent(threshold->getX(), threshold->getY(), 0);
00063         } else if ( threshold->getCondition() == ThrCondition::bad_undefine ) {
00064             nUndefined++;
00065         }
00066     }
00067     std::cout << mThresholdDistribution->GetMean() << std::endl;
00068     std::cout << mThresholdDistribution->GetStdDev() << std::endl;
00069     std::cout << mErrorDistribution->GetMean() << std::endl;
00070     std::cout << mErrorDistribution->GetStdDev() << std::endl;
00071     std::cout << mThresholdDistribution->GetEntries() << std::endl;
00072     std::cout << nLow << std::endl;
00073     std::cout << nHigh << std::endl;
00074     std::cout << nUndefined << std::endl;
00075 }
00076
00077 void TThresholdAnalyser::saveThresholdDistribution(std::string_view title) const {
00078     TCanvas* can = new TCanvas("thrDist", "Threshold Distribution; Threshold [$500 \times e^{-}$];
Entry", 1000, 500);
00079     mThresholdDistribution->Draw();
00080     can->SaveAs(static_cast<const TString>(title));
00081 }
00082
00083 void TThresholdAnalyser::saveErrorDistribution(std::string_view title) const {
00084     TCanvas* can = new TCanvas("errDist", "Error Distribution;Threshold [$e^{-}$]; Entry", 1000, 500);
00085     mErrorDistribution->Draw();
00086     can->SaveAs(static_cast<const TString>(title));
00087 }
00088
00089 void TThresholdAnalyser::saveThresholdmap(std::string_view title) const {
00090     TCanvas* can(new TCanvas("thrmap", "", 1000, 500));
00091     mThresholdmap->SetTitle("Threshold Distribution; rows; coloums");
00092     mThresholdmap->SetMinimum(0);
00093     mThresholdmap->SetMaximum(50);
00094     mThresholdmap->SetStats(0);
00095     mThresholdmap->Draw("COLZ0");
00096     can->SaveAs(static_cast<const TString>(title));
00097 }
00098
00099 void TThresholdAnalyser::saveQualityDistribution(std::string_view title) const {
00100     TCanvas* can = new TCanvas("chi2", "Fit quality Distribution; Chi2/NDoF; entry", 1000, 500);
00101     mChi2NdfDistribution->Draw();
00102     can->SaveAs(static_cast<const TString>(title));
00103 }

```

8.11 /home/ychoi/ATOM/alpide/cluster/inc/TCluster.h File Reference

The [TCluster](#) class header. Cluster properties are defined.

```

#include <vector>
#include <algorithm>
#include <cmath>

```

Classes

- class [TCluster](#)

8.11.1 Detailed Description

The [TCluster](#) class header. Cluster properties are defined.

Author

Yongjun Choi (yochoi@cern.ch)

Version

0.1

Date

2024-04-06

Copyright

Copyright (c) 2024

Definition in file [TCluster.h](#).

8.12 TCluster.h

[Go to the documentation of this file.](#)

```
00001
00012 #ifndef __TCLUSTER__
00013 #define __TCLUSTER__
00014
00015 #ifdef __TCLUSTER_HEADER__
00016 #endif
00017
00018 #include<vector>
00019 #include<algorithm>
00020 #include<cmath>
00021
00022 class TCluster {
00023 private:
00024     int mEvent;
00025     int mTime;
00026     std::vector<std::pair<int, int>> mPixels;
00028     int mMinX = 1024, mMinY = 512;
00029     int mMaxX = 0, mMaxY = 0;
00030     std::pair<double, double> center;
00031     double mLongRadius;
00032     // std::pair<double, double> stdevInAxis; /**< standard deviation in x and y direction */
00033     // double stdev; /**< Root of mean of squared distance from cluster centre */
00034     int size = 0;
00035     // TMatrix2D<int> shape; /**< The shape matrix for analyse cluster shape */
00036
00037 public:
00038     TCluster();
00039     TCluster(int event, int time);
00040     TCluster(const TCluster& copy);
00041     TCluster& operator=(const TCluster& copy);
00042     TCluster(TCluster&& move);
00043     TCluster& operator=(TCluster&& move);
00044     ~TCluster();
00045
00046     // Add Function
00047     void AddPixel(const std::pair<int, int>& pixel);
00048     void AddCluster(const TCluster& cluster);
00049
00050     // Compare functions
00051     bool isNeighbour(const std::pair<int, int>& pixel) const;
00052     bool isNeighbour(const TCluster& cluster) const;
```

```

00053     bool isContain(const std::pair<int, int>& pixel) const;
00054     bool isContain(const TCluster& cluster) const;
00055     const int getDistance(const std::pair<int, int>& pixel1, const std::pair<int, int>& pixel2) const;
00056
00057     void calMembers();
00058     void calMinMax();
00059     void calCenter();
00060     void calLongRadius();
00061     // void calStdevInAxis();
00062     // void calStdev();
00063     void calSize();
00064     // void calShape();
00065
00066     // Getter
00067     const std::pair<double, double> getCenter() const;
00068     const double getLongRadius() const;
00069     // const std::pair<double, double> getStdevInAxis() const;
00070     // const double getStdev() const;
00071     const int getSize() const;
00072     // const TMatrix2D<int>& getShape() const;
00073
00074     // Setter for member
00075     void setEvent(const int event);
00076     void setTimeStamp(const int time);
00077     void setMinX(const int minX);
00078     void setMinY(const int minY);
00079     void setMaxX(const int maxX);
00080     void setMaxY(const int maxY);
00081
00082     // Getter for member
00083     const std::vector<std::pair<int, int>> getPixels() const;
00084     const int getEvent() const;
00085     const int getTimeStamp() const;
00086     const int getMinX() const;
00087     const int getMinY() const;
00088     const int getMaxX() const;
00089     const int getMaxY() const;
00090
00091     bool operator==(const TCluster& cluster) const;
00092     bool operator!=(const TCluster& cluster) const;
00093
00094 private:
00095     unsigned int fBits;
00096 public:
00097     enum {
00098         kNotDeleted = 0x02000000
00099     };
00100     bool IsDestroyed() const { return !TestBit(kNotDeleted); }
00101     bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
00102 };
00103
00104 #endif

```

8.13 /home/ychoi/ATOM/alpide/cluster/inc/TClusterDivideData.h File Reference

```

#include <unordered_map>
#include <vector>

```

Classes

- class [TClusterDivideData](#)

8.14 TClusterDivideData.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TCLUSTERDIVIDEDATA__
00002 #define __TCLUSTERDIVIDEDATA__

```

```

00003
00004 #ifndef __TCLUSTERDIVIDEDATA_HEADER__
00005 #include <iostream>
00006 #include "TCluster.h"
00007 #endif
00008
00009 #include<unordered_map>
00010 #include<vector>
00011
00012 class TCluster;
00013
00014 class TClusterDivideData {
00015 private:
00016     std::unordered_map<int, std::vector<TCluster*>> mClusterData;
00017 public:
00018     TClusterDivideData(const std::vector<TCluster*>& clusters);
00019     TClusterDivideData(const TClusterDivideData& copy);
00020     const std::vector<TCluster*>& getClusterOfSize(const int clusterSize);
00021 };
00022
00023 #endif

```

8.15 /home/ychoi/ATOM/alpide/cluster/inc/TClusterization.h File Reference

Class for clusterizing events on a chip.

```
#include <vector>
```

Classes

- class [TClusterization](#)
Class of tools for clusterizing events for single event.

8.15.1 Detailed Description

Class for clusterizing events on a chip.

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

13-04-2024

Copyright

Copyright (c) 2024

Definition in file [TClusterization.h](#).

8.16 TClusterization.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef __TCLUSTERIZATION__
00013 #define __TCLUSTERIZATION__
00014
00015 #ifdef __TCLUSTERIZATION_HEADER__
00016 #include "cpptqdm.h"
00017
00018 #include "TALPIDEEvent.h"
00019 #include "TCluster.h"
00020 #endif
00021
00022 #include <vector>
00023
00024 class TALPIDEEvent;
00025 class TCluster;
00026
00032 class TClusterization {
00033     std::vector<TALPIDEEvent*> mEvents;
00034     std::vector<TCluster*> mClusters;
00035 public:
00036     TClusterization() = delete;
00037     TClusterization(const std::vector<TALPIDEEvent*>& events);
00038     ~TClusterization();
00039
00040     void removeConsecutionPixels(); // Remove later pixels if it fired continuously.
00041
00042     void addNewCluster(const std::pair<int, int>& pixel, std::vector<TCluster*>& clusters, int iEvent,
00043 int iTime);
00044
00045     void removeIndependentCluster(const std::pair<int, int>& pixel, std::vector<TCluster*>& clusters);
00046
00047     bool clusterLitmusTest(const std::pair<int, int>& pixel, std::vector<TCluster*>& clusters);
00048     void clusterize();
00049     const std::vector<TCluster*>& getClusters() const;
00050 };
00051 #endif

```

8.17 /home/ychoi/ATOM/alpide/cluster/inc/TClusterShape.h File Reference

Tools for extracting cluster shape image.

```

#include <vector>
#include <unordered_map>

```

Classes

- struct [TShapeInfo](#)
The information set structure for clusters that having homeomorphism shape.
- class [TClusterShape](#)
Class for extracting cluster shape information with same cluster size.

8.17.1 Detailed Description

Tools for extracting cluster shape image.

AuthorYongjun Choi (yochoi@cern.ch)**Version**

0.1

Date

16-04-2024

Copyright

Copyright (c) 2024

Definition in file [TClusterShape.h](#).

8.18 TClusterShape.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef __TCLUSTERSHAPE__
00013 #define __TCLUSTERSHAPE__
00014
00015 #ifdef __TCLUSTERSHAPE_HEADER__
00016 #include "TCluster.h"
00017 #include "TH2I.h"
00018 #include "TMatrix2D.h"
00019 #endif
00020
00021 #include<vector>
00022 #include<unordered_map>
00023
00024 class TH2I;
00025
00026 class TCluster;
00027 template<typename T> class TMatrix2D;
00028
00038 struct TShapeInfo {
00039     TCluster* mPresidentCluster;
00040     TMatrix2D<int>* mClusterMatrix;
00041     TH2I* mClusterMap;
00042     int iShape;
00043     int mEntry;
00044     int mShortBinN;
00045     int mLongBinN;
00046 };
00047
00058 class TClusterShape {
00059 private:
00060     int mClusterSize;
00061     std::vector<TCluster*> mClusterOriginSet;
00062     std::vector<TShapeInfo> mClusterShapeInfos;
00063 public:
00064     TClusterShape();
00065     TClusterShape(const int clusterSize, const std::vector<TCluster*>& clusters);
00066     TClusterShape(const std::vector<TCluster*> clusters);
00067     // TClusterShape(const std::vector<TCluster*> clusters, const int clusterSize);
00068     ~TClusterShape();
00069
00070     void identifyShapes();
00071     void sortShapes(bool descend = true);
00072     void calClusterInfo(TShapeInfo& shapeInfo, TCluster* cluster);
00073     TMatrix2D<int>* clusterMatrix(const TCluster* cluster);
00074     TH2I* clusterMap(const TMatrix2D<int>* clusterMatrix);
00075
00076     const std::vector<TShapeInfo>& getClusterShapeInfos() const;
00077
00078     void setClusterSize(const int ClusterSize);

```

```
00079     const int getClusterSize() const;
00080 private:
00081     unsigned int fBits;
00082 public:
00083     enum {
00084         kNotDeleted = 0x02000000
00085     };
00086     bool IsDestructed() const { return !TestBit(kNotDeleted); }
00087     bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
00088 };
00089
00090 #endif
```

8.19 /home/ychoi/ATOM/alpide/cluster/inc/TExperimentData.h File Reference

```
#include <vector>
```

Classes

- class [TExperimentData](#)

8.19.1 Detailed Description

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

16-04-2024

Copyright

Copyright (c) 2024

Definition in file [TExperimentData.h](#).

8.20 TExperimentData.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef __TEXPERIMENTDATA__
00013 #define __TEXPERIMENTDATA__
00014
00015 #ifdef __TEXPERIMENTDATA_HEADER__
00016 #include "TALPIDEEvent.h"
00017 #include "TCluster.h"
00018 #endif
00019
00020 #include<vector>
00021
00022 class TALPIDEEvent;
00023 class TCluster;
00024
00025 class TExperimentData {
00026 private:
00027     std::vector<TALPIDEEvent*> mEvents;
00028     std::vector<TCluster*> mClusters;
00029     // std::vector<TALPIDEEvent*> mMaskedEvents;
00030     // std::vector<TCluster*> mMaskedClusters;
00031     // std::vector<TALPIDEEvent*> mNoiseEvents;
00032     // std::vector<TCluster*> mNoiseClusters;
00033
00034 public:
00035     TExperimentData();
00036     TExperimentData(const TExperimentData& copy);
00037     TExperimentData& operator=(const TExperimentData& copy);
00038     ~TExperimentData();
00039
00040     void setEvents(const std::vector<TALPIDEEvent*>& events);
00041     void setClusters(const std::vector<TCluster*>& clusters);
00042     // void setMaskedEvents(const std::vector<TALPIDEEvent*>&& maskedEvents);
00043     // void setMaskedClusters(const std::vector<TCluster*>&& maskedClusters);
00044     // void setNoiseEvents(const std::vector<TALPIDEEvent*>&& noiseEvents);
00045     // void setNoiseClusters(const std::vector<TCluster*>&& noiseClusters);
00046     const std::vector<TALPIDEEvent*> getEvents() const;
00047     const std::vector<TCluster*> getClusters() const;
00048     // const std::vector<TALPIDEEvent*> getMaskedEvents() const;
00049     // const std::vector<TCluster*> getMaskedClusters() const;
00050     // const std::vector<TALPIDEEvent*> getNoiseEvents() const;
00051     // const std::vector<TCluster*> getNoiseClusters() const;
00052
00053 private:
00054     unsigned int fBits;
00055 public:
00056     enum {
00057         kNotDeleted = 0x02000000
00058     };
00059     bool IsDestructed() const { return !TestBit(kNotDeleted); }
00060     bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
00061 };
00062
00063 #endif

```

8.21 /home/ychoi/ATOM/alpide/cluster/inc/TMatrix2D.h File Reference

```
#include <iostream>
```

Classes

- class [TMatrix2D< numT >](#)

Functions

- template<typename numT >
std::ostream & [operator<<](#) (std::ostream &os, const [TMatrix2D< numT >](#) &matrix)

8.21.1 Function Documentation

8.21.1.1 operator<<()

```
template<typename numT >
std::ostream & operator<< (
    std::ostream & os,
    const TMatrix2D< numT > & matrix )
```

Definition at line 299 of file [TMatrix2D.h](#).

8.22 TMatrix2D.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TMATRIX2D__
00002 #define __TMATRIX2D__
00003
00004 #include <iostream>
00005
00006 template <typename numT>
00007 class TMatrix2D;
00008
00009 template<typename numT>
00010 std::ostream& operator<< (std::ostream& os, const TMatrix2D<numT>& matrix);
00011
00012 template <typename numT>
00013 class TMatrix2D {
00014 private:
00015     std::vector<std::vector<numT>> mMatrix;
00016     int mNRow, mNColumn;
00017 public:
00018     TMatrix2D();
00019     TMatrix2D(int nRow, int nColumn);
00020     TMatrix2D(const std::vector<std::vector<numT>>& matrix);
00021
00022     void setMatrix(const std::vector<std::vector<numT>>& matrix);
00023     void setElement(const int iRow, const int iColumn, const numT element);
00024     const numT getElement(const int iRow, const int iColumn) const;
00025     const int getNRow() const;
00026     const int getNColumn() const;
00027     const std::pair<int, int> getDimension() const;
00028
00029     template<typename numT2>
00030     bool isSameDimension(const TMatrix2D<numT2>& matrix);
00031     template<typename numT2>
00032     bool isSame(const TMatrix2D<numT2>& matrix);
00033     template<typename numT2>
00034     bool hasXSymmetry(const TMatrix2D<numT2>& matrix);
00035     template<typename numT2>
00036     bool hasYSymmetry(const TMatrix2D<numT2>& matrix);
00037     template<typename numT2>
00038     bool hasXYSymmetry(const TMatrix2D<numT2>& matrix);
00039     template<typename numT2>
00040     bool hasRSymmetry(const TMatrix2D<numT2>& matrix);
00041     template<typename numT2>
00042     bool hasRXSymmetry(const TMatrix2D<numT2>& matrix);
00043     template<typename numT2>
00044     bool hasRYSymmetry(const TMatrix2D<numT2>& matrix);
00045     template<typename numT2>
00046     bool hasRXYSymmetry(const TMatrix2D<numT2>& matrix);
00047     template<typename numT2>
00048     bool hasHomeomorphism(const TMatrix2D<numT2>& matrix);
00049
00050     friend std::ostream& operator<<> (std::ostream& os, const TMatrix2D<numT>& matrix);
00051     template<typename numT2>
00052     TMatrix2D& operator+=(const TMatrix2D<numT2>& matrix);
00053     template<typename numT2>
00054     TMatrix2D& operator+(const TMatrix2D<numT2>& matrix);
00055     template<typename numT2>
00056     TMatrix2D& operator-=(const TMatrix2D<numT2>& matrix);
00057     template<typename numT2>
00058     TMatrix2D& operator-(const TMatrix2D<numT2>& matrix);
00059     template<typename numT2>
00060     TMatrix2D& operator*=(numT2 scalar);
00061     template<typename numT2>
```

```

00062     TMatrix2D& operator*(numT2 scalar);
00063     template<typename numT2>
00064     TMatrix2D& operator+=(const TMatrix2D<numT2>& matrix);
00065     template<typename numT2>
00066     TMatrix2D& operator*(const TMatrix2D<numT2>& matrix);
00067 };
00068
00069
00070
/*-----
00071 template<typename numT>
00072 TMatrix2D<numT>::TMatrix2D() { }
00073
00074 template<typename numT>
00075 TMatrix2D<numT>::TMatrix2D(int nRow, int nColumn) : mNRow(nRow), mNColumn(nColumn) {
00076     for ( int iRow = 0; iRow < nRow; iRow++ ) {
00077         std::vector<numT> temp;
00078         for ( int iColumn = 0; iColumn < nColumn; iColumn++ ) {
00079             temp.push_back(0);
00080         }
00081         mMatrix.push_back(temp);
00082     }
00083 }
00084
00085 template<typename numT>
00086 TMatrix2D<numT>::TMatrix2D(const std::vector<std::vector<numT>>& matrix) {
00087     mNRow = matrix.size();
00088     mNColumn = matrix[0].size();
00089     mMatrix.assign(matrix.begin(), matrix.end());
00090 }
00091
00092 template<typename numT>
00093 void TMatrix2D<numT>::setMatrix(const std::vector<std::vector<numT>>& matrix) {
00094     mNRow = matrix.size();
00095     mNColumn = matrix[0].size();
00096     mMatrix.clear();
00097     mMatrix.assign(matrix.begin(), matrix.end());
00098 }
00099
00100 template<typename numT>
00101 void TMatrix2D<numT>::setElement(const int iRow, const int iColumn, const numT element) {
00102     mMatrix[iRow][iColumn] = element;
00103 }
00104
00105 template<typename numT>
00106 const numT TMatrix2D<numT>::getElement(const int iRow, const int iColumn) const {
00107     try {
00108         if ( iRow < 0 || iRow > mNRow ) throw iRow;
00109         if ( iColumn < 0 || iColumn > mNColumn ) throw iColumn;
00110         return mMatrix[iRow][iColumn];
00111     } catch ( int iRow ) {
00112         std::cerr << "The row index should be positive and smaller than " << mNRow << ". Current: " << iRow
00113         << std::endl;
00114         return mMatrix[0][0];
00115     }
00116 }
00117
00118 template<typename numT>
00119 const int TMatrix2D<numT>::getNRow() const {
00120     return mNRow;
00121 }
00122
00123 template<typename numT>
00124 const int TMatrix2D<numT>::getNColumn() const {
00125     return mNColumn;
00126 }
00127
00128 template<typename numT>
00129 const std::pair<int, int> TMatrix2D<numT>::getDimension() const {
00130     return {mNRow, mNColumn};
00131 }
00132
00133 template<typename numT>
00134 template<typename numT2>
00135 bool TMatrix2D<numT>::isSameDimension(const TMatrix2D<numT2>& matrix) {
00136     if ( mNRow == matrix.getNRow() && mNColumn == matrix.getNColumn() ) {
00137         return true;
00138     } else {
00139         return false;
00140     }
00141 }
00142
00143 template<typename numT>
00144 template<typename numT2>
00145 bool TMatrix2D<numT>::isSame(const TMatrix2D<numT2>& matrix) {
00146     for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00147         for ( int iRow = 0; iRow < mNRow; iRow++ ) {

```

```

00147         if ( mMatrix[iRow][iColumn] != matrix.getElement(iRow, iColumn) ) {
00148             return false;
00149         }
00150     }
00151 }
00152 return true;
00153 }
00154
00155 template<typename numT>
00156 template<typename numT2>
00157 bool TMatrix2D<numT>::hasYSymmetry(const TMatrix2D<numT2>& matrix) {
00158     for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00159         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00160             if ( mMatrix[iRow][iColumn] != matrix.getElement(mNRow - iRow - 1, iColumn) ) {
00161                 return false;
00162             }
00163         }
00164     }
00165     return true;
00166 }
00167
00168 template<typename numT>
00169 template<typename numT2>
00170 bool TMatrix2D<numT>::hasXSymmetry(const TMatrix2D<numT2>& matrix) {
00171     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00172         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00173             if ( mMatrix[iRow][iColumn] != matrix.getElement(iRow, mNColumn - iColumn - 1) ) {
00174                 return false;
00175             }
00176         }
00177     }
00178     return true;
00179 }
00180
00181 template<typename numT>
00182 template<typename numT2>
00183 bool TMatrix2D<numT>::hasXYSymmetry(const TMatrix2D<numT2>& matrix) {
00184     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00185         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00186             if ( mMatrix[iRow][iColumn] != matrix.getElement(mNRow - iRow - 1, mNColumn - iColumn - 1) ) {
00187                 return false;
00188             }
00189         }
00190     }
00191     return true;
00192 }
00193
00194 template<typename numT>
00195 template<typename numT2>
00196 bool TMatrix2D<numT>::hasRSymmetry(const TMatrix2D<numT2>& matrix) {
00197     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00198         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00199             if ( mMatrix[iRow][iColumn] != matrix.getElement(mNColumn - iColumn - 1, iRow) ) {
00200                 return false;
00201             }
00202         }
00203     }
00204     return true;
00205 }
00206
00207 template<typename numT>
00208 template<typename numT2>
00209 bool TMatrix2D<numT>::hasRXSymmetry(const TMatrix2D<numT2>& matrix) {
00210     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00211         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00212             if ( mMatrix[iRow][iColumn] != matrix.getElement(iColumn, iRow) ) {
00213                 return false;
00214             }
00215         }
00216     }
00217     return true;
00218 }
00219
00220 template<typename numT>
00221 template<typename numT2>
00222 bool TMatrix2D<numT>::hasRYSymmetry(const TMatrix2D<numT2>& matrix) {
00223     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00224         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00225             if ( mMatrix[iRow][iColumn] != matrix.getElement(mNColumn - iColumn - 1, mNRow - iRow - 1) ) {
00226                 return false;
00227             }
00228         }
00229     }
00230     return true;
00231 }

```

```

00232
00233 template<typename numT>
00234 template<typename numT2>
00235 bool TMatrix2D<numT>::hasRXYSymmetry(const TMatrix2D<numT2>& matrix) {
00236     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00237         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00238             if ( mMatrix[iRow][iColumn] != matrix.getElement(iColumn, mNRow - iRow - 1) ) {
00239                 return false;
00240             }
00241         }
00242     }
00243     return true;
00244 }
00245
00246 template<typename numT>
00247 template<typename numT2>
00248 bool TMatrix2D<numT>::hasHomeomorphism(const TMatrix2D<numT2>& matrix) {
00249     if ( mNRow == mNColumn && mNRow == matrix.getNRow() && mNColumn == matrix.getNColumn() ) {
00250         if ( isSame(matrix) ) {
00251             return true;
00252         } else if ( hasXSymmetry(matrix) ) {
00253             return true;
00254         } else if ( hasYSymmetry(matrix) ) {
00255             return true;
00256         } else if ( hasXYSymmetry(matrix) ) {
00257             return true;
00258         } else if ( hasRSymmetry(matrix) ) {
00259             return true;
00260         } else if ( hasRXSymmetry(matrix) ) {
00261             return true;
00262         } else if ( hasRYSymmetry(matrix) ) {
00263             return true;
00264         } else if ( hasRXYSymmetry(matrix) ) {
00265             return true;
00266         } else {
00267             return false;
00268         }
00269     } else if ( mNRow == matrix.getNRow() && mNColumn == matrix.getNColumn() ) {
00270         if ( isSame(matrix) ) {
00271             return true;
00272         } else if ( hasXSymmetry(matrix) ) {
00273             return true;
00274         } else if ( hasYSymmetry(matrix) ) {
00275             return true;
00276         } else if ( hasXYSymmetry(matrix) ) {
00277             return true;
00278         } else {
00279             return false;
00280         }
00281     } else if ( mNRow == matrix.getNColumn() && mNColumn == matrix.getNRow() ) {
00282         if ( hasRSymmetry(matrix) ) {
00283             return true;
00284         } else if ( hasRXSymmetry(matrix) ) {
00285             return true;
00286         } else if ( hasRYSymmetry(matrix) ) {
00287             return true;
00288         } else if ( hasRXYSymmetry(matrix) ) {
00289             return true;
00290         } else {
00291             return false;
00292         }
00293     } else {
00294         return false;
00295     }
00296 }
00297
00298 template<typename numT>
00299 std::ostream& operator<<(std::ostream& os, const TMatrix2D<numT>& matrix) {
00300     for ( const std::vector<numT>& aRow : matrix.mMatrix ) {
00301         for ( const numT& element : aRow ) {
00302             os << element << " ";
00303         }
00304         os << std::endl;
00305     }
00306     os << "    Dimension: " << matrix.mNRow << "X" << matrix.mNColumn;
00307     return os;
00308 }
00309
00310 template<typename numT>
00311 template<typename numT2>
00312 TMatrix2D<numT>& TMatrix2D<numT>::operator+=(const TMatrix2D<numT2>& matrix) {
00313     try {
00314         if ( !isSameDimension(matrix) ) throw matrix;
00315         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00316             for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00317                 mMatrix[iRow][iColumn] += matrix.getElement(iRow, iColumn);
00318             }

```

```

00319     }
00320     } catch ( TMatrix2D<numT> matrix ) {
00321         std::cerr << "The dimension should be same for plus operator. (" << mNRow << "X" << mNColumn << "
!= " << matrix.mNRow << "X" << matrix.mNColumn << ")" << std::endl;
00322     }
00323     return *this;
00324 }
00325
00326 template<typename numT>
00327 template<typename numT2>
00328 TMatrix2D<numT>& TMatrix2D<numT>::operator+(const TMatrix2D<numT2>& matrix) {
00329     try {
00330         if ( !isSameDimension(matrix) ) throw matrix;
00331         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00332             for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00333                 mMatrix[iRow][iColumn] += matrix.getElement(iRow, iColumn);
00334             }
00335         }
00336     } catch ( TMatrix2D<numT> matrix ) {
00337         std::cerr << "The dimension should be same for plus operator. (" << mNRow << "X" << mNColumn << "
!= " << matrix.mNRow << "X" << matrix.mNColumn << ")" << std::endl;
00338     }
00339     return *this;
00340 }
00341
00342 template<typename numT>
00343 template<typename numT2>
00344 TMatrix2D<numT>& TMatrix2D<numT>::operator-=(const TMatrix2D<numT2>& matrix) {
00345     try {
00346         if ( !isSameDimension(matrix) ) throw matrix;
00347         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00348             for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00349                 mMatrix[iRow][iColumn] -= matrix.getElement(iRow, iColumn);
00350             }
00351         }
00352     } catch ( TMatrix2D<numT> matrix ) {
00353         std::cerr << "The dimension should be same for minus operator. (" << mNRow << "X" << mNColumn << "
!= " << matrix.mNRow << "X" << matrix.mNColumn << ")" << std::endl;
00354     }
00355     return *this;
00356 }
00357
00358 template<typename numT>
00359 template<typename numT2>
00360 TMatrix2D<numT>& TMatrix2D<numT>::operator-=(const TMatrix2D<numT2>& matrix) {
00361     try {
00362         if ( !isSameDimension(matrix) ) throw matrix;
00363         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00364             for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00365                 mMatrix[iRow][iColumn] -= matrix.getElement(iRow, iColumn);
00366             }
00367         }
00368     } catch ( TMatrix2D<numT> matrix ) {
00369         std::cerr << "The dimension should be same for minus operator. (" << mNRow << "X" << mNColumn << "
!= " << matrix.mNRow << "X" << matrix.mNColumn << ")" << std::endl;
00370     }
00371     return *this;
00372 }
00373
00374 template<typename numT>
00375 template<typename numT2>
00376 TMatrix2D<numT>& TMatrix2D<numT>::operator*=(const numT2 scalar) {
00377     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00378         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00379             mMatrix[iRow][iColumn] *= scalar;
00380         }
00381     }
00382     return *this;
00383 }
00384
00385 template<typename numT>
00386 template<typename numT2>
00387 TMatrix2D<numT>& TMatrix2D<numT>::operator*(const numT2 scalar) {
00388     for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00389         for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
00390             mMatrix[iRow][iColumn] *= scalar;
00391         }
00392     }
00393     return *this;
00394 }
00395
00396 template<typename numT>
00397 template<typename numT2>
00398 TMatrix2D<numT>& TMatrix2D<numT>::operator*=(const TMatrix2D<numT2>& matrix) {
00399     try {
00400         if ( mNRow != matrix.getNColumn() ) throw matrix;
00401

```

```

00402         TMatrix2D<numT> newMatrix(mNRow, matrix.getNColumn());
00403         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00404             for ( int iColumn = 0; iColumn < matrix.getNColumn(); iColumn++ ) {
00405                 for ( int iOperand = 0; iOperand < mNColumn; iOperand++ ) {
00406                     newMatrix.setElement(iRow, iColumn, newMatrix.getElement(iRow, iColumn) +
mMatrix[iRow][iOperand] * matrix.getElement(iOperand, iColumn));
00407                 }
00408             }
00409         }
00410         *this = newMatrix;
00411     } catch ( TMatrix2D<numT> matrix ) {
00412         std::cerr << "The row dimension should be same with column dimension of operand for multiple
operator. (" << mNRow << "X" << mNColumn << " != " << matrix.mNRow << "X" << matrix.mNColumn << ")" <<
std::endl;
00413     }
00414     return *this;
00415 }
00416
00417 template<typename numT>
00418 template<typename numT2>
00419 TMatrix2D<numT>& TMatrix2D<numT>::operator*(const TMatrix2D<numT2>& matrix) {
00420     try {
00421         if ( mNRow != matrix.getNColumn() ) throw matrix;
00422
00423         TMatrix2D<numT> newMatrix(mNRow, matrix.getNColumn());
00424         for ( int iRow = 0; iRow < mNRow; iRow++ ) {
00425             for ( int iColumn = 0; iColumn < matrix.getNColumn(); iColumn++ ) {
00426                 for ( int iOperand = 0; iOperand < mNColumn; iOperand++ ) {
00427                     newMatrix.setElement(iRow, iColumn, newMatrix.getElement(iRow, iColumn) +
mMatrix[iRow][iOperand] * matrix.getElement(iOperand, iColumn));
00428                 }
00429             }
00430         }
00431         *this = newMatrix;
00432     } catch ( TMatrix2D<numT> matrix ) {
00433         std::cerr << "The row dimension should be same with column dimension of operand for multiple
operator. (" << mNRow << "X" << mNColumn << " != " << matrix.mNRow << "X" << matrix.mNColumn << ")" <<
std::endl;
00434     }
00435     return *this;
00436 }
00437
00438
00439
00440
00441 #endif

```

8.23 /home/ychoi/ATOM/alpide/cluster/src/TCluster.cpp File Reference

```
#include "TCluster.h"
```

Macros

- `#define __TCLUSTER_HEADER__`

8.23.1 Macro Definition Documentation

8.23.1.1 __TCLUSTER_HEADER__

```
#define __TCLUSTER_HEADER__
```

Definition at line 1 of file [TCluster.cpp](#).

8.24 TCluster.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TCLUSTER_HEADER__
00002 #include "TCluster.h"
00003
00004 TCluster::TCluster() : fBits(kNotDeleted) { };
00005
00006 TCluster::TCluster(int event, int time) : mEvent(event), mTime(time), fBits(kNotDeleted) { }
00007
00008 TCluster::TCluster(const TCluster& copy) : mEvent(copy.mEvent), mTime(copy.mTime), mMinX(copy.mMinX),
    mMinY(copy.mMinY), mMaxX(copy.mMaxX), mMaxY(copy.mMaxY), fBits(kNotDeleted) {
00009     mPixels.assign(copy.mPixels.begin(), copy.mPixels.end());
00010 }
00011
00012 TCluster& TCluster::operator=(const TCluster& copy) {
00013     mEvent = copy.mEvent;
00014     mTime = copy.mTime;
00015     mPixels.assign(copy.mPixels.begin(), copy.mPixels.end());
00016     mMinX = copy.mMinX;
00017     mMinY = copy.mMinY;
00018     mMaxX = copy.mMaxX;
00019     mMaxY = copy.mMaxY;
00020
00021     return *this;
00022 }
00023
00024 TCluster::TCluster(TCluster&& move) : mEvent(move.mEvent), mTime(move.mTime),
    mPixels(std::move(move.mPixels)), mMinX(move.mMinX), mMinY(move.mMinY), mMaxX(move.mMaxX),
    mMaxY(move.mMaxY) { }
00025
00026 TCluster& TCluster::operator=(TCluster&& move) {
00027     mEvent = move.mEvent;
00028     mTime = move.mTime;
00029     mPixels = std::move(move.mPixels);
00030     mMinX = move.mMinX;
00031     mMinY = move.mMinY;
00032     mMaxX = move.mMaxX;
00033     mMaxY = move.mMaxY;
00034     return *this;
00035 }
00036
00037 TCluster::~TCluster() { }
00038
00039 void TCluster::AddPixel(const std::pair<int, int>& pixel) {
00040     mPixels.push_back(pixel);
00041     mMaxY = std::max(mMaxY, pixel.second);
00042 }
00043
00044 void TCluster::AddCluster(const TCluster& cluster) {
00045     for (const std::pair<int, int>& pixel : cluster.getPixels()) {
00046         mPixels.push_back(pixel);
00047     }
00048     mMaxY = std::max(mMaxY, cluster.getMaxY());
00049 }
00050
00051 bool TCluster::isNeighbour(const std::pair<int, int>& pixel) const {
00052     bool neighbour = false;
00053     for (std::pair<int, int> mPixel : mPixels) {
00054         if (getDistance(pixel, mPixel) <= 1) {
00055             neighbour = true;
00056             break;
00057         }
00058     }
00059     return neighbour;
00060 }
00061
00062 bool TCluster::isNeighbour(const TCluster& cluster) const {
00063     bool neighbour = false;
00064     for (std::pair<int, int> comparePixel : cluster.getPixels()) {
00065         neighbour = isNeighbour(comparePixel);
00066         if (neighbour) break;
00067     }
00068     return neighbour;
00069 }
00070
00071 bool TCluster::isContain(const std::pair<int, int>& pixel) const {
00072     bool neighbour = false;
00073     if (find(mPixels.begin(), mPixels.end(), pixel) != mPixels.end()) {
00074         neighbour = true;
00075     }
00076     return neighbour;
00077 }
00078
00079 bool TCluster::isContain(const TCluster& cluster) const {

```



```

00080     bool neighbour = false;
00081     for ( std::pair<int, int> comparePixel : cluster.getPixels() ) {
00082         neighbour = isContain(comparePixel);
00083         if ( neighbour ) break;
00084     }
00085     return neighbour;
00086 }
00087
00088 const int TCluster::getDistance(const std::pair<int, int>& pixel1, const std::pair<int, int>& pixel2)
const {
00089     return abs(pixel1.first - pixel2.first) + abs(pixel1.second - pixel2.second);
00090 }
00091
00092 void TCluster::calMembers() {
00093     calMinMax();
00094     calCenter();
00095     calSize();
00096     calLongRadius();
00097     // calShape();
00098     // calStdevInAxis();
00099     // calStdev();
00100 }
00101
00102 void TCluster::calMinMax() {
00103     for ( std::pair<int, int> pixel : mPixels ) {
00104         mMinX = std::min(pixel.first, mMinX);
00105         mMinY = std::min(pixel.second, mMinY);
00106         mMaxX = std::max(pixel.first, mMaxX);
00107         mMaxY = std::max(pixel.second, mMaxY);
00108     }
00109 }
00110
00111 void TCluster::calCenter() {
00112     double x = 0., y = 0.;
00113     for ( std::pair<int, int> pixel : mPixels ) {
00114         x += pixel.first;
00115         y += pixel.second;
00116     }
00117     x = (double) x / mPixels.size();
00118     y = (double) y / mPixels.size();
00119     center = {x, y};
00120 }
00121
00122 void TCluster::calSize() {
00123     size = mPixels.size();
00124 }
00125
00126 void TCluster::calLongRadius() {
00127     double maxDistance = 0.;
00128     for ( const std::pair<int, int>& pixel : mPixels ) {
00129         double distance = pow(abs(pixel.first - center.first) + .5, 2) + pow(abs(pixel.second -
center.second) + .5, 2);
00130         maxDistance = std::max(maxDistance, distance);
00131     }
00132     mLongRadius = sqrt(maxDistance);
00133 }
00134
00135 const std::pair<double, double> TCluster::getCenter() const {
00136     return center;
00137 }
00138
00139 const int TCluster::getSize() const {
00140     return size;
00141 }
00142
00143 const double TCluster::getLongRadius() const {
00144     return mLongRadius;
00145 }
00146
00147 // Setter for member
00148 void TCluster::setEvent(const int event) { mEvent = event; }
00149 void TCluster::setTimeStamp(const int time) { mTime = time; }
00150 void TCluster::setMinX(const int minX) { mMinX = minX; }
00151 void TCluster::setMinY(const int minY) { mMinY = minY; }
00152 void TCluster::setMaxX(const int maxX) { mMaxX = maxX; }
00153 void TCluster::setMaxY(const int maxY) { mMaxY = maxY; }
00154
00155 // Getter for member
00156 const std::vector<std::pair<int, int>> TCluster::getPixels() const { return mPixels; }
00157 const int TCluster::getEvent() const { return mEvent; }
00158 const int TCluster::getTimeStamp() const { return mTime; }
00159 const int TCluster::getMinX() const { return mMinX; }
00160 const int TCluster::getMinY() const { return mMinY; }
00161 const int TCluster::getMaxX() const { return mMaxX; }
00162 const int TCluster::getMaxY() const { return mMaxY; }
00163

```

8.25 /home/ychoi/ATOM/alpide/cluster/src/TClusterDivideData.cpp File Reference

```
#include "TClusterDivideData.h"
```

Macros

- `#define __TCLUSTERDIVIDEDATA_HEADER__`

8.25.1 Macro Definition Documentation

8.25.1.1 __TCLUSTERDIVIDEDATA_HEADER__

```
#define __TCLUSTERDIVIDEDATA_HEADER__
```

Definition at line 1 of file [TClusterDivideData.cpp](#).

8.26 TClusterDivideData.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TCLUSTERDIVIDEDATA_HEADER__
00002 #include "TClusterDivideData.h"
00003
00004 TClusterDivideData::TClusterDivideData(const std::vector<TCluster*>& clusters) {
00005     for ( TCluster* cluster : clusters ) {
00006         int clusterSize = cluster->getSize();
00007         if ( mClusterData.count(clusterSize) ) {
00008             mClusterData.find(clusterSize)->second.push_back(cluster);
00009         } else {
00010             std::vector<TCluster*> set;
00011             set.push_back(cluster);
00012             mClusterData.insert_or_assign(clusterSize, set);
00013         }
00014     }
00015 }
00016
00017 TClusterDivideData::TClusterDivideData(const TClusterDivideData& copy) :
00018     mClusterData(copy.mClusterData) { }
00019
00020 const std::vector<TCluster*>& TClusterDivideData::getClusterOfSize(const int clusterSize) {
00021     try {
00022         if ( !mClusterData.count(clusterSize) ) throw clusterSize;
00023         return mClusterData.find(clusterSize)->second;
00024     } catch ( int clusterSize ) {
00025         std::cerr << "There are no clusters having cluster Size " + std::to_string(clusterSize) <<
00026             std::endl;
00027         std::vector<TCluster*>* empty;
00028         return *empty;
00029     }
00030 }
```

8.27 /home/ychoi/ATOM/alpide/cluster/src/TClusterization.cpp File Reference

```
#include "TClusterization.h"
```

Macros

- `#define __TCLUSTERIZATION_HEADER__`

8.27.1 Macro Definition Documentation

8.27.1.1 __TCLUSTERIZATION_HEADER__

```
#define __TCLUSTERIZATION_HEADER__
```

Definition at line 1 of file [TClusterization.cpp](#).

8.28 TClusterization.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TCLUSTERIZATION_HEADER__
00002 #include "TClusterization.h"
00003
00004 TClusterization::TClusterization(const std::vector<TALPIDEvent*>& events) {
00005     for ( auto& event : events ) {
00006         mEvents.push_back(event);
00007     }
00008 }
00009
00010 TClusterization::~TClusterization() { }
00011
00012 void TClusterization::removeConsecutionPixels() {
00013     std::vector<std::pair<int, int>> preDeletedPixel;
00014     std::vector<std::pair<int, int>> currentDeletedPixel;
00015     for ( int iEvent = 1; iEvent < mEvents.size(); iEvent++ ) {
00016         for ( int iPixel = mEvents[iEvent]->getData().size() - 1; iPixel > -1; iPixel-- ) {
00017
00018             if ( std::find(mEvents[iEvent - 1]->getData().begin(), mEvents[iEvent -
00019 1]->getData().end(), mEvents[iEvent]->getData()[iPixel]) != mEvents[iEvent - 1]->getData().end() ||
00020 (!preDeletedPixel.empty() && find(preDeletedPixel.begin(), preDeletedPixel.end(),
00021 mEvents[iEvent]->getData()[iPixel]) != preDeletedPixel.end() ) ) {
00022                 currentDeletedPixel.push_back(mEvents[iEvent]->getData()[iPixel]);
00023                 mEvents[iEvent]->removePixel(mEvents[iEvent]->getData()[iPixel]);
00024             }
00025         }
00026         preDeletedPixel.clear();
00027         preDeletedPixel = std::move(currentDeletedPixel);
00028         currentDeletedPixel.clear();
00029     }
00030 }
00031 void TClusterization::addNewCluster(const std::pair<int, int>& pixel, std::vector<TCluster*>&
00032 clusters, int iEvent, int iTime) {
00033     TCluster* cluster = new TCluster(iEvent, iTime);
00034     cluster->AddPixel(pixel);
00035     clusters.push_back(cluster);
00036 }
00037
00038 void TClusterization::removeIndependentCluster(const std::pair<int, int>& pixel,
00039 std::vector<TCluster*>& clusters) {
00040     if ( clusters[0]->getMaxY() + 2 < pixel.second ) {
00041         int cut = 0;
00042         for ( const TCluster* cluster : clusters ) {
00043             if ( cluster->getMaxY() + 2 < pixel.second ) {
00044                 cut++;
00045             } else break;
00046         }
00047         mClusters.insert(mClusters.end(), clusters.begin(), clusters.begin() + cut);
00048         clusters.erase(clusters.begin(), clusters.begin() + cut);
00049     }
00050 }
00051
00052 bool TClusterization::clusterLitmusTest(const std::pair<int, int>& pixel, std::vector<TCluster*>&
00053 clusters) {
00054     bool clusterExist = false;
00055     int iCluster = 0;
00056     int iOrigin = 0;
00057     for ( TCluster* clustered : clusters ) {
```

```

00053         if ( clustered->isNeighbour(pixel) ) {
00054             if ( clusterExist == false ) {
00055                 clustered->AddPixel(pixel);
00056                 clusterExist = true;
00057                 iOrigin = iCluster;
00058             } else {
00059                 clusters[iOrigin]->AddCluster(*clustered);
00060                 clusters.erase(clusters.begin() + iCluster);
00061                 break;
00062             }
00063         }
00064         iCluster++;
00065     }
00066     return clusterExist;
00067 }
00068 }
00069
00070 void TClusterization::clusterize() {
00071     std::clog << "Generate clusters from events..." << std::endl;
00072
00073     ProgressBar* pBar = new ProgressBar(mEvents.size());
00074
00075     for ( TALPIDEEvent*& event : mEvents ) {
00076         pBar->printProgress();
00077         std::vector<TCluster*> clusterCandidate;
00078         bool isFirst = true;
00079         for ( const std::pair<int, int>& pixel : event->getData() ) {
00080             if ( isFirst ) {
00081                 addNewCluster(pixel, clusterCandidate, event->getEvent(), event->getTime());
00082                 isFirst = false;
00083                 continue;
00084             }
00085             removeIndependentCluster(pixel, clusterCandidate);
00086             bool clusterExist = clusterLitmusTest(pixel, clusterCandidate);
00087             if ( !clusterExist ) {
00088                 addNewCluster(pixel, clusterCandidate, event->getEvent(), event->getTime());
00089             } // Make new cluster if no relation with any cluster candidates.
00090         }
00091         if ( !clusterCandidate.empty() ) {
00092             mClusters.insert(mClusters.end(), clusterCandidate.begin(), clusterCandidate.end());
00093             clusterCandidate.clear();
00094         } // Move remained cluster candidates to cluster
00095     }
00096     delete pBar;
00097     pBar = nullptr;
00098     std::clog << "Total " << mClusters.size() << " clusters are extracted from " << mEvents.size() << "
events." << std::endl << std::endl;
00099     for ( TCluster* cluster : mClusters ) {
00100         cluster->calMembers();
00101     }
00102 }
00103
00104 const std::vector<TCluster*>& TClusterization::getClusters() const {
00105     return mClusters;
00106 }

```

8.29 /home/ychoi/ATOM/alpide/cluster/src/TClusterOperator.cpp File Reference

```
#include "TCluster.h"
```

8.30 TClusterOperator.cpp

[Go to the documentation of this file.](#)

```

00001 #include "TCluster.h"
00002
00003 bool TCluster::operator==(const TCluster& cluster) const {
00004     int aX = mMaxX - mMinX;
00005     int aY = mMaxY - mMinY;
00006     int bX = cluster.mMaxX - cluster.mMinX;
00007     int bY = cluster.mMaxY - cluster.mMinY;
00008     if ( mPixels.size() != cluster.mPixels.size() ) {

```

```

00009         return false;
00010     }
00011     if ( !(((aX == bX) && (aY == bY)) || ((aX == bY) && (aY == bX))) ) {
00012         return false;
00013     }
00014     return true;
00015 }
00016
00017 bool TCluster::operator!=(const TCluster& cluster) const {
00018     int aX = mMaxX - mMinX;
00019     int aY = mMaxY - mMinY;
00020     int bX = cluster.mMaxX - cluster.mMinX;
00021     int bY = cluster.mMaxY - cluster.mMinY;
00022
00023     if ( mPixels.size() != cluster.mPixels.size() ) {
00024         return true;
00025     }
00026
00027     if ( !(((aX == bX) && (aY == bY)) || ((aX == bY) && (aY == bX))) ) {
00028         return true;
00029     }
00030     return true;
00031 }

```

8.31 /home/ychoi/ATOM/alpide/cluster/src/TClusterShape.cpp File Reference

```
#include "TClusterShape.h"
```

Macros

- `#define __TCLUSTERSHAPE_HEADER__`

8.31.1 Macro Definition Documentation

8.31.1.1 __TCLUSTERSHAPE_HEADER__

```
#define __TCLUSTERSHAPE_HEADER__
```

Definition at line 1 of file [TClusterShape.cpp](#).

8.32 TClusterShape.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TCLUSTERSHAPE_HEADER__
00002 #include "TClusterShape.h"
00003
00004
00017 TClusterShape::TClusterShape() : fBits(kNotDeleted) { }
00018
00019 TClusterShape::TClusterShape(const int clusterSize, const std::vector<TCluster*>& clusters) :
00020     mClusterSize(clusterSize), fBits(kNotDeleted) {
00021     mClusterOriginSet.assign(clusters.begin(), clusters.end());
00022 }
00023 // TClusterShape::TClusterShape(const std::vector<TCluster*> clusters, const int clusterSize) :
00024     mClusterSize(clusterSize) {
00025     // for ( TCluster* cluster : clusters ) {
00026     //     if ( cluster->getSize() == clusterSize ) {
00027     //         mClusterWithN.push_back(cluster);
00028     //     }
00029 }

```

```

00028 // // }
00029 // }
00030
00031 TClusterShape::TClusterShape(const std::vector<TCluster*> clusters) {
00032     // mClusterWithN.assign(clusters.begin(), clusters.end());
00033 }
00034
00035 TClusterShape::~TClusterShape() {
00036     for ( TShapeInfo shapeInfo : mClusterShapeInfos ) {
00037         delete shapeInfo.mClusterMap;
00038         delete shapeInfo.mClusterMatrix;
00039     }
00040 }
00041 }
00042
00053 void TClusterShape::identifyShapes() {
00054     bool isFirst = true;
00055     int iShape = 0;
00056     for ( TCluster* cluster : mClusterOriginSet ) {
00057         if ( isFirst ) {
00058             TShapeInfo shapeInfo;
00059             shapeInfo.mPresidentCluster = cluster;
00060             shapeInfo.mEntry = 1;
00061             shapeInfo.iShape = iShape;
00062             iShape++;
00063             calClusterInfo(shapeInfo, cluster);
00064             shapeInfo.mClusterMap = clusterMap(shapeInfo.mClusterMatrix);
00065             mClusterShapeInfos.push_back(shapeInfo);
00066             std::vector<TCluster*> newClusterSet;
00067             isFirst = false;
00068             continue;
00069         }
00070         TMatrix2D<int>* checkingCluster = clusterMatrix(cluster);
00071         bool isHomoomorphismExist = false;
00072         for ( TShapeInfo& shapeInfo : mClusterShapeInfos ) {
00073             TMatrix2D<int>* comparedCluster = shapeInfo.mClusterMatrix;
00074             if ( comparedCluster->hasHomeomorphism(*checkingCluster) ) {
00075                 shapeInfo.mEntry++;
00076                 isHomoomorphismExist = true;
00077                 // mClusterSameSizeSet.find(shapeInfo.iShape)->second.push_back(cluster);
00078                 break;
00079             }
00080         }
00081         if ( !isHomoomorphismExist ) {
00082             TShapeInfo shapeInfo;
00083             shapeInfo.mPresidentCluster = cluster;
00084             shapeInfo.mEntry = 1;
00085             shapeInfo.iShape = iShape;
00086             iShape++;
00087             calClusterInfo(shapeInfo, cluster);
00088             shapeInfo.mClusterMap = clusterMap(shapeInfo.mClusterMatrix);
00089             mClusterShapeInfos.push_back(shapeInfo);
00090             std::vector<TCluster*> newClusterSet;
00091         }
00092     }
00093 }
00094 }
00107 void TClusterShape::sortShapes(bool descend) {
00108     std::sort(mClusterShapeInfos.begin(), mClusterShapeInfos.end(),
00109         [ ](TShapeInfo& a, TShapeInfo& b) {
00110             if ( a.mEntry != b.mEntry ) {
00111                 return a.mEntry > b.mEntry;
00112             } else if ( a.mClusterMatrix->getNRow() != b.mClusterMatrix->getNRow() ) {
00113                 return a.mClusterMatrix->getNRow() < b.mClusterMatrix->getNRow();
00114             } else {
00115                 return a.mClusterMatrix->getNColumn() < b.mClusterMatrix->getNColumn();
00116             }
00117         }
00118     );
00119 }
00120
00134 TH2I* TClusterShape::clusterMap(const TMatrix2D<int>* clusterMatrix) {
00135     static int numbering;
00136     int nRow = clusterMatrix->getNRow();
00137     int nColumn = clusterMatrix->getNColumn();
00138
00139     TH2I* map = new TH2I(Form("_%d", numbering), "", nRow + 2, 0, nRow + 2, nColumn + 2, 0, nColumn +
00140 2);
00141     for ( int iRow = 0; iRow < nRow; iRow++ ) {
00142         for ( int iColumn = 0; iColumn < nColumn; iColumn++ ) {
00143             if ( clusterMatrix->getElement(iRow, iColumn) == 1 ) {
00144                 map->Fill(iRow + 1, iColumn + 1);
00145             }
00146         }
00147     }
00148     map->GetXaxis()->SetNdivisions(nRow + 2, 0, 0, true);

```

```

00149     for ( int i = 1; i <= map->GetNbinsX(); ++i ) {
00150         map->GetXaxis()->SetBinLabel(i, "");
00151     }
00152     for ( int i = 1; i <= map->GetNbinsY(); ++i ) {
00153         map->GetYaxis()->SetBinLabel(i, "");
00154     }
00155     map->GetYaxis()->SetNdivisions(nColumn + 2, 0, 0, true);
00156     map->GetZaxis()->SetNdivisions(0, 0, 0, true);
00157     map->SetStats(0);
00158
00159     numbering++;
00160     return map;
00161 }
00162
00163
00164 void TClusterShape::calClusterInfo(TShapeInfo& shapeInfo, TCluster* cluster) {
00165     int xBinN = cluster->getMaxX() - cluster->getMinX();
00166     int yBinN = cluster->getMaxY() - cluster->getMinY();
00167     bool longHeight = yBinN > xBinN ? true : false;
00168     shapeInfo.mLongBinN = longHeight ? yBinN : xBinN;
00169     shapeInfo.mShortBinN = longHeight ? xBinN : yBinN;
00170
00171     shapeInfo.mClusterMatrix = new TMatrix2D<int>(shapeInfo.mLongBinN + 1, shapeInfo.mShortBinN + 1);
00172     if ( longHeight ) {
00173         for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
00174             shapeInfo.mClusterMatrix->setElement(pixel.second - cluster->getMinY(), pixel.first -
cluster->getMinX(), 1);
00175         }
00176     } else {
00177         for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
00178             shapeInfo.mClusterMatrix->setElement(pixel.first - cluster->getMinX(), pixel.second -
cluster->getMinY(), 1);
00179         }
00180     }
00181 }
00182 }
00183
00184 TMatrix2D<int>* TClusterShape::clusterMatrix(const TCluster* cluster) {
00185     int xBinN = cluster->getMaxX() - cluster->getMinX();
00186     int yBinN = cluster->getMaxY() - cluster->getMinY();
00187     TMatrix2D<int>* matrix = new TMatrix2D<int>(xBinN + 1, yBinN + 1);
00188     for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
00189         matrix->setElement(pixel.first - cluster->getMinX(), pixel.second - cluster->getMinY(), 1);
00190     }
00191     return matrix;
00192 }
00193
00199 void TClusterShape::setClusterSize(const int clusterSize) {
00200     mClusterSize = clusterSize;
00201 }
00206 const std::vector<TShapeInfo>& TClusterShape::getClusterShapeInfos() const {
00207     return mClusterShapeInfos;
00208 }
00213 const int TClusterShape::getClusterSize() const {
00214     return mClusterSize;
00215 }

```

8.33 /home/ychoi/ATOM/alpide/cluster/src/TExperimentData.cpp File Reference

```

#include "TExperimentData.h"
#include <iostream>

```

Macros

- `#define __TEXPERIMENTDATA_HEADER__`

8.33.1 Macro Definition Documentation

8.33.1.1 `__TEXPERIMENTDATA_HEADER__`

```
#define __TEXPERIMENTDATA_HEADER__
```

Definition at line 1 of file [TExperimentData.cpp](#).

8.34 TExperimentData.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TEXPERIMENTDATA_HEADER__
00002 #include "TExperimentData.h"
00003 #include<iostream>
00004 TExperimentData::TExperimentData() : fBits(kNotDeleted) { }
00005
00006 TExperimentData::TExperimentData(const TExperimentData& copy) : mEvents(copy.mEvents),
    mClusters(copy.mClusters) {
00007     std::cout << "mEvents" << mEvents.size() << std::endl;
00008     std::cout << "mClusters" << mClusters.size() << std::endl;
00009 }
00010
00011
00012 TExperimentData& TExperimentData::operator=(const TExperimentData& copy) {
00013     mEvents.clear();
00014     mEvents.assign(copy.mEvents.begin(), copy.mEvents.end());
00015
00016     mClusters.clear();
00017     mClusters.assign(copy.mClusters.begin(), copy.mClusters.end());
00018
00019     fBits = copy.fBits;
00020     return *this;
00021 }
00022
00023
00024 TExperimentData::~TExperimentData() {
00025     for ( TALPIDEvent* event : mEvents ) {
00026         if ( event != nullptr && !event->IsDestructed() ) {
00027             delete event;
00028             event = nullptr;
00029         }
00030     }
00031     for ( TCluster* cluster : mClusters ) {
00032         if ( cluster != nullptr && !cluster->IsDestructed() ) {
00033             delete cluster;
00034             cluster = nullptr;
00035         }
00036     }
00037     // for ( TALPIDEvent* maskedEvent : mMaskedEvents ) {
00038     //     if ( maskedEvent != nullptr && !maskedEvent->IsDestructed() ) {
00039     //         delete maskedEvent;
00040     //         maskedEvent = nullptr;
00041     //     }
00042     // }
00043     // for ( TCluster* maskedCluster : mMaskedClusters ) {
00044     //     if ( maskedCluster != nullptr && !maskedCluster->IsDestructed() ) {
00045     //         delete maskedCluster;
00046     //         maskedCluster = nullptr;
00047     //     }
00048     // }
00049     // for ( TALPIDEvent* noiseEvent : mNoiseEvents ) {
00050     //     if ( noiseEvent != nullptr && !noiseEvent->IsDestructed() ) {
00051     //         delete noiseEvent;
00052     //         noiseEvent = nullptr;
00053     //     }
00054     // }
00055     // for ( TCluster* noiseCluster : mNoiseClusters ) {
00056     //     if ( noiseCluster != nullptr && !noiseCluster->IsDestructed() ) {
00057     //         delete noiseCluster;
00058     //         noiseCluster = nullptr;
00059     //     }
00060     // }
00061 }
00062
00063
00064 void TExperimentData::setEvents(const std::vector<TALPIDEvent*>& events) {
00065     mEvents = events;
00066 }
00067
00068 void TExperimentData::setClusters(const std::vector<TCluster*>& clusters) {
00069     mClusters = clusters;
00070 }
00071
00072 // void TExperimentData::setMaskedEvents(const std::vector<TALPIDEvent*>&& maskedEvents) {
00073 //     mMaskedEvents = maskedEvents;
00074 // }
00075
00076 // void TExperimentData::setMaskedClusters(const std::vector<TCluster*>&& maskedClusters) {
00077 //     mMaskedClusters = maskedClusters;
00078 // }
00079
00080 // void TExperimentData::setNoiseEvents(const std::vector<TALPIDEvent*>&& noiseEvents) {
00081 //     mNoiseEvents = noiseEvents;

```



```

00082 // }
00083
00084 // void TExperimentData::setNoiseClusters(const std::vector<TCluster*>&& noiseClusters) {
00085 //     mNoiseClusters = noiseClusters;
00086 // }
00087
00088 const std::vector<TALPIDEEvent*> TExperimentData::getEvents() const {
00089     return mEvents;
00090 }
00091
00092 const std::vector<TCluster*> TExperimentData::getClusters() const {
00093     return mClusters;
00094 }
00095
00096 // const std::vector<TALPIDEEvent*> TExperimentData::getMaskedEvents() const {
00097 //     return mMaskedEvents;
00098 // }
00099
00100 // const std::vector<TCluster*> TExperimentData::getMaskedClusters() const {
00101 //     return mMaskedClusters;
00102 // }
00103
00104 // const std::vector<TALPIDEEvent*> TExperimentData::getNoiseEvents() const {
00105 //     return mNoiseEvents;
00106 // }
00107
00108 // const std::vector<TCluster*> TExperimentData::getNoiseClusters() const {
00109 //     return mNoiseClusters;
00110 // }

```

8.35 /home/ychoi/ATOM/alpide/comparison/inc/TGraphCompare.h File Reference

```

#include <unordered_map>
#include <vector>
#include <filesystem>

```

Classes

- class [TGraphCompare](#)

8.36 TGraphCompare.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TGRAPHCOMPARE__
00002 #define __TGRAPHCOMPARE__
00003
00004 #ifdef __TGRAPHCOMPARE_HEADER__
00005 #include <iostream>
00006 #include<sstream>
00007 #include "Rtypes.h"
00008 #include "TFile.h"
00009 #include "TCanvas.h"
00010 #include "TH1D.h"
00011 #include "TH2D.h"
00012 #include "TString.h"
00013 #include "TColor.h"
00014 #include "TLegend.h"
00015 #include "CppConfigFile.h"
00016 #include "TGraphAsymmErrors.h"
00017 #endif
00018
00019 #include <unordered_map>
00020 #include <vector>
00021 #include<filesystem>
00022
00023 class TFile;
00024 class TH1D;

```

```

00025
00026 class Configurable;
00027
00028 class TGraphCompare {
00029 private:
00030     std::unordered_map<std::string, TFile*> mGraphFileSet;
00031     std::vector<TH1D*> mClusterSizeSet;
00032     std::vector<std::string> mGraphInfoSet;
00033 public:
00034     TGraphCompare(const std::vector<std::string>& graphFilePath);
00035     void TCompareClusterSize(std::string_view typeName, const CppConfigDictionary config);
00036     // void TCompareClusterSizeRatio(std::string_view typeName, const Configurable* config);
00037     // void TCompareClusterSize3D(std::string_view typeName, const Configurable* config);
00038     TH1D* getClustersizeHistogram(std::string_view pathInRoot, std::string operation);
00039
00040 };
00041
00042 #endif
00043
00044 // struct TGraphObjects {
00045 //     std::string name;
00046 //     std::string operation;
00047 //     std::string legend;
00048 //     double weighting;
00049 //     bool operator < (const TGraphObjects& other) const { return true; }
00050 //     bool operator==(const TGraphObjects& other) const { return name == other.name; }
00051 //     TH1D* getClustersizeHistogram(std::string_view pathInRoot, const std::unordered_map<std::string,
00052 //                                     TFile*>& graphFileSet);
00052 // };

```

8.37 /home/ychoi/ATOM/alpide/comparison/inc/TMerge.h File Reference

Tools for integrate same configure but seperate file.

```

#include <vector>
#include <string>
#include "TFileFormat.h"

```

Classes

- class [MergeFileOpen](#)
- class [MergeTreeOpen](#)
- class [TMerge](#)
- class [TMergeExperimentROOT](#)

8.37.1 Detailed Description

Tools for integrate same configure but seperate file.

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

17-04-2024

Copyright

Copyright (c) 2024

Definition in file [TMerge.h](#).

8.38 TMerge.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef __TMERGE__
00013 #define __TMERGE__
00014
00015 #ifdef __TMERGE_HEADER__
00016 #include "TFile.h"
00017 #include "TTree.h"
00018 #include "cpptqdm.h"
00019 #endif
00020
00021 #include <vector>
00022 #include <string>
00023
00024 #include "TFileFormat.h"
00025
00026 class TFile;
00027
00028 class MergeFileOpen : public std::exception {
00029 public:
00030     std::string message;
00031 public:
00032     MergeFileOpen(std::string_view fileName) {
00033         message = static_cast<std::string>(fileName) + " cannot open";
00034     }
00035     const char* what() const throw() {
00036         return message.c_str();
00037     }
00038 };
00039
00040 class MergeTreeOpen : public std::exception {
00041 public:
00042     std::string message;
00043 public:
00044     MergeTreeOpen(std::string_view fileName) {
00045         message = "Tree of " + static_cast<std::string>(fileName) + " cannot be accessed";
00046     }
00047     const char* what() const throw() {
00048         return message.c_str();
00049     }
00050 };
00051
00052 class TMerge {
00053 protected:
00054     std::string mOutputFileName;
00055     std::vector<std::string> mInputFileNames;
00056 public:
00057     TMerge(std::string_view outputFileName, const std::vector<std::string>& inputFileNames);
00058 };
00059
00060 class TMergeExperimentROOT : public TMerge {
00061 private:
00062     TFile* mOutputFile;
00063     TInputRoot mInputValue;
00064     TInputRoot mOutputValue;
00065 public:
00066     TMergeExperimentROOT(std::string_view outputFileName, const std::vector<std::string>&
inputFileNames);
00067     void mergeFile();
00068     ~TMergeExperimentROOT();
00069 };
00070
00071 #endif

```

8.39 /home/ychoi/ATOM/alpide/comparison/src/TGraphCompare.cpp File Reference

```
#include "TGraphCompare.h"
```

Macros

- `#define __TGRAPHCOMPARE_HEADER__`

Functions

- int [getColor](#) (const std::string colorName)
- int [getStyle](#) (const int index)

8.39.1 Macro Definition Documentation

8.39.1.1 `__TGRAPHCOMPARE_HEADER__`

```
#define __TGRAPHCOMPARE_HEADER__
```

Definition at line 1 of file [TGraphCompare.cpp](#).

8.39.2 Function Documentation

8.39.2.1 `getColor()`

```
int getColor (
    const std::string colorName )
```

Definition at line 49 of file [TGraphCompare.cpp](#).

8.39.2.2 `getStyle()`

```
int getStyle (
    const int index )
```

Definition at line 65 of file [TGraphCompare.cpp](#).

8.40 TGraphCompare.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TGRAPHCOMPARE_HEADER__
00002 #include "TGraphCompare.h"
00003
00004 TH1D* TGraphCompare::getClustersizeHistogram(std::string_view pathInRoot, std::string operation) {
00005     std::vector<std::string> stringSet;
00006
00007     size_t start = 0, end = 0;
00008     operation.erase(remove(operation.begin(), operation.end(), ' '), operation.end());
00009     while ( (end = operation.find_first_of("+*/", start)) != std::string::npos ) {
00010         if ( end != start ) {
00011             stringSet.push_back(operation.substr(start, end - start));
00012         }
00013         stringSet.push_back(std::string(1, operation[end]));
00014         start = end + 1;
00015     }
00016     if ( start < operation.length() ) {
00017         stringSet.push_back(operation.substr(start));
00018     }
00019
00020     TH1D* hist1 = (TH1D*)
00021     mGraphFileSet.find(stringSet[0])->second->Get(static_cast<TString>(pathInRoot));
00022     TH1D* newHist = new TH1D(*hist1);
00023     if ( stringSet.size() == 3 ) {
00024         TH1D* hist2 = (TH1D*)
00025         mGraphFileSet.find(stringSet[2])->second->Get(static_cast<TString>(pathInRoot));
00026         if ( stringSet[1] == "+" ) {
```

```

00025         newHist->Add(hist2);
00026         for ( int i = 0; i < 80; i++ ) {
00027             newHist->SetBinError(i, sqrt(hist1->GetBinContent(i) + hist2->GetBinContent(i)));
00028         }
00029     } else if ( stringSet[1] == "-" ) {
00030         newHist->Add(hist2, -1);
00031         for ( int i = 0; i < 80; i++ ) {
00032             newHist->SetBinError(i, sqrt(hist1->GetBinContent(i) + hist2->GetBinContent(i)));
00033         }
00034     } else if ( stringSet[1] == "*" ) {
00035         newHist->Multiply(hist2);
00036         for ( int i = 0; i < 80; i++ ) {
00037             newHist->SetBinError(i, sqrt(hist1->GetBinContent(i) * hist2->GetBinContent(i) *
(hist1->GetBinContent(i) + hist2->GetBinContent(i))));
00038         }
00039     } else if ( stringSet[1] == "/" ) {
00040         newHist->Divide(hist2);
00041         for ( int i = 0; i < 80; i++ ) {
00042             newHist->SetBinError(i, sqrt(pow(hist1->GetBinContent(i) / hist2->GetBinContent(i), 4)
+ 1));
00043         }
00044     }
00045 }
00046 return newHist;
00047 }
00048
00049 int getColor(const std::string colorName) {
00050     if ( colorName == "kRed" ) {
00051         return kRed;
00052     } else if ( colorName == "kMagenta" ) {
00053         return kMagenta;
00054     } else if ( colorName == "kBlue" ) {
00055         return kBlue;
00056     } else if ( colorName == "kCyan" ) {
00057         return kCyan + 2;
00058     } else if ( colorName == "kOrange" ) {
00059         return kOrange;
00060     } else {
00061         return kWhite;
00062     }
00063 }
00064
00065 int getStyle(const int index) {
00066     if ( index == 0 ) {
00067         return 1;
00068     } else if ( index == 1 ) {
00069         return 8;
00070     } else if ( index == 2 ) {
00071         return 10;
00072     } else if ( index == 3 ) {
00073         return 6;
00074     } else if ( index == 4 ) {
00075         return 9;
00076     } else {
00077         return 0;
00078     }
00079 }
00080
00081 TGraphCompare::TGraphCompare(const std::vector<std::string>& graphFilePath) {
00082     for ( std::string_view filePath : graphFilePath ) {
00083         std::filesystem::path path(filePath);
00084         TFile* file = new TFile(static_cast<TString>(path));
00085         mGraphFileSet.insert_or_assign(std::string(path.stem()), file);
00086     }
00087 }
00088
00089 void TGraphCompare::TCompareClusterSize(std::string_view typeName, const CppConfigDictionary config) {
00090     TString canvasName = "can_" + config.find("name");
00091     Int_t canvasWidth = stoi(config.getSubConfig("canvas").find("width"));
00092     Int_t canvasHeight = stoi(config.getSubConfig("canvas").find("height"));
00093     TCanvas* canvas = new TCanvas(canvasName, "", canvasWidth, canvasHeight);
00094
00095     TString legendTitle = config.getSubConfig("legend").hasKey("title") ?
config.getSubConfig("legend").find("title") : "";
00096     Float_t legendXmin = config.getSubConfig("legend").hasKey("x_min") ?
stof(config.getSubConfig("legend").find("x_min")) : 0.7;
00097     Float_t legendXmax = config.getSubConfig("legend").hasKey("x_max") ?
stof(config.getSubConfig("legend").find("x_max")) : 0.7;
00098     Float_t legendYmin = config.getSubConfig("legend").hasKey("y_min") ?
stof(config.getSubConfig("legend").find("y_min")) : 0.9;
00099     Float_t legendYmax = config.getSubConfig("legend").hasKey("y_max") ?
stof(config.getSubConfig("legend").find("y_max")) : 0.9;
00100     Float_t legendDivide = config.getSubConfig("legend").hasKey("divide") ?
stoi(config.getSubConfig("legend").find("divide")) : 1;
00101     TLegend* legend = new TLegend(legendXmin, legendYmin, legendXmax, legendYmax);
00102     legend->SetHeader(legendTitle, "c");
00103     legend->SetNCColumns(legendDivide);

```

```

00104     std::map<std::string, CppConfigDictionary> plotConfigList;
00105     for ( const auto& pair : config.getSubConfig("plots").getSubConfigSetWithName() ) {
00106         plotConfigList.insert_or_assign(pair.first, pair.second);
00107     }
00108     std::map<std::string, TH1D*> distributionSet;
00109
00110     std::string rootPath = config.find("root_path");
00111     for ( const auto& plotConfig : plotConfigList ) {
00112         distributionSet.insert_or_assign(plotConfig.first, getClustersizeHistogram(rootPath,
plotConfig.second.find("histogram")));
00113     }
00114
00115     bool isFirst;
00116     for ( const auto& distribution : distributionSet ) {
00117         if ( isFirst ) {
00118             isFirst = false;
00119         }
00120
00121         double nEntry = 0;
00122         int csMin = stoi(config.find("cluster_size_of_interest_min"));
00123         int csMax = stoi(config.find("cluster_size_of_interest_max"));
00124
00125         for ( int clusterSize = 1; clusterSize < 81; clusterSize++ ) {
00126             if ( clusterSize < csMin || clusterSize > csMax ) {
00127                 distribution.second->SetBinContent(clusterSize, 0);
00128             }
00129         }
00130
00131         if ( config.hasKey("normalization") && (config.find("normalization") == "true") ) {
00132             distribution.second->Scale(1. / nEntry);
00133         } else if ( plotConfigList.find(distribution.first)->second.hasKey("scale") ) {
00134             Float_t scaleFactor = stof(plotConfigList.find(distribution.first)->second.find("scale"));
00135             distribution.second->Scale(scaleFactor);
00136         }
00137
00138         for ( int clusterSize = csMin; clusterSize < csMax + 1; clusterSize++ ) {
00139             nEntry += distribution.second->GetBinContent(clusterSize);
00140         }
00141         TString legendTitle = plotConfigList.find(distribution.first)->second.find("legend_title");
00142         TString strEntry = Form("%.2f", round(nEntry * 100) / 100.);
00143         if ( plotConfigList.find(distribution.first)->second.hasKey("scale_error_max") ) {
00144             TString strEntryPlus = Form("%.2f", abs(round(nEntry *
(stof(plotConfigList.find(distribution.first)->second.find("scale_error_max")) - 1) * 100) / 100.));
00145             TString strEntryMinus = Form("%.2f", abs(round(nEntry * (1 -
stof(plotConfigList.find(distribution.first)->second.find("scale_error_min")) * 100) / 100.));
00146             legend->AddEntry(distribution.second, legendTitle + "(" + strEntry + "+" + strEntryPlus +
"--" + strEntryMinus + " in " + std::to_string(csMin) + " ~ " + std::to_string(csMax) + ")", "l");
00147         } else {
00148             Float_t scaleFactor = stof(plotConfigList.find(distribution.first)->second.find("scale"));
00149             TString strError = Form("%.2f", sqrt(nEntry * scaleFactor));
00150             legend->AddEntry(distribution.second, legendTitle + "(" + strEntry + char(0x00B1) +
strError + " in " + std::to_string(csMin) + " ~ " + std::to_string(csMax) + ")", "l");
00151         }
00152
00153         TString title = config.find("title");
00154         TString xTitle = config.find("x_title");
00155         TString yTitle = config.find("y_title");
00156
00157         distribution.second->SetMinimum(stof(config.find("y_min")));
00158         distribution.second->SetMaximum(stof(config.find("y_max")));
00159         distribution.second->GetXaxis()->SetRangeUser(csMin - .5, csMax + .5);
00160         distribution.second->SetTitle(title + "; " + xTitle + "; " + yTitle);
00161
00162         distribution.second->SetLineColor(getColor(plotConfigList.find(distribution.first)->second.find("line_color")));
00163
00164         Float_t lineWidth = config.hasKey("line_width") ? stof(config.find("line_width")) : 1.;
00165         distribution.second->SetLineWidth(lineWidth);
00166         Float_t lineStyle = plotConfigList.find(distribution.first)->second.hasKey("line_style") ?
stof(plotConfigList.find(distribution.first)->second.find("line_style")) : 1.;
00167         distribution.second->SetLineStyle(lineStyle);
00168
00169         distribution.second->SetStats(0);
00170         distribution.second->Draw("SAME HISTE");
00171         if ( plotConfigList.find(distribution.first)->second.hasKey("scale_error_max") ) {
00172             TGraphAsymmErrors* systemGraph = new TGraphAsymmErrors();
00173             for ( int clusterSize = csMin; clusterSize < csMax + 1; clusterSize++ ) {
00174                 systemGraph->SetPoint(clusterSize, clusterSize,
distribution.second->GetBinContent(clusterSize));
00175                 systemGraph->SetPointError(clusterSize, .5, .5,
distribution.second->GetBinContent(clusterSize) * (1 -
stof(plotConfigList.find(distribution.first)->second.find("scale_error_min"))),
distribution.second->GetBinContent(clusterSize) *
(stof(plotConfigList.find(distribution.first)->second.find("scale_error_max")) - 1));
00176             }
00177             Float_t alpha = stof(config.find("error_box_alpha"));
00178

```

```

00179     systemGraph->SetFillColorAlpha(getColor(plotConfigList.find(distribution.first)->second.find("line_color")),
00180                                     alpha);
00181     }
00182     }
00183     legend->Draw();
00184     if ( config.HasKey("logy") && (config.find("logy") == "true") ) {
00185         canvas->SetLogy();
00186     }
00187     std::filesystem::path outputPath(config.find("output_path"));
00188     std::filesystem::create_directories(outputPath);
00189     outputPath /= config.find("name");
00190     outputPath.replace_extension(config.find("extension"));
00191     canvas->SaveAs(static_cast<TString>(outputPath));
00192 }
00193 }
00194 }

```

8.41 /home/ychoi/ATOM/alpide/comparison/src/TMerge.cpp File Reference

```
#include "TMerge.h"
```

Macros

- `#define __TMERGE_HEADER__`

8.41.1 Macro Definition Documentation

8.41.1.1 __TMERGE_HEADER__

```
#define __TMERGE_HEADER__
```

Definition at line 1 of file [TMerge.cpp](#).

8.42 TMerge.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TMERGE_HEADER__
00002 #include "TMerge.h"
00003
00004 TMerge::TMerge(std::string_view outputFileName, const std::vector<std::string>& inputFileNames) :
00005     mOutputFileName(outputFileName) {
00006     mInputFileNames.assign(inputFileNames.begin(), inputFileNames.end());
00007 }
00008 TMergeExperimentROOT::TMergeExperimentROOT(std::string_view outputFileName, const
00009     std::vector<std::string>& inputFileNames) : TMerge(outputFileName, inputFileNames) {
00010     mOutputFile = new TFile(static_cast<TString>(outputFileName), "RECREATE");
00011 }
00012 void TMergeExperimentROOT::mergeFile() {
00013     try {
00014         TTree* outputTree = new TTree("hit", "hit");
00015         outputTree->Branch("ChipID", &mOutputValue.chipid, "id/b");
00016         outputTree->Branch("TimeStamp", &mOutputValue.timeStamp, "t/i");
00017         outputTree->Branch("X", &mOutputValue.x, "x/s");
00018         outputTree->Branch("Y", &mOutputValue.y, "y/s");
00019
00020         int preTime = 0;

```

```

00021     for ( std::string_view inputFileName : mInputFileNames ) {
00022         TFile* inputFile = new TFile(static_cast<TString>(inputFileName), "READ");
00023         if ( !inputFile->IsOpen() ) {
00024             throw MergeFileOpen(inputFileName);
00025         } else if ( inputFile->Get("hit") == nullptr ) {
00026             throw MergeTreeOpen(inputFileName);
00027         }
00028         std::clog << inputFileName << std::endl;
00029         TTree* inputTree = static_cast<TTree*>(inputFile->Get("hit"));
00030         inputTree->SetBranchAddress("ChipID", &mInputValue.chipid);
00031         inputTree->SetBranchAddress("TimeStamp", &mInputValue.timeStamp);
00032         inputTree->SetBranchAddress("X", &mInputValue.x);
00033         inputTree->SetBranchAddress("Y", &mInputValue.y);
00034         ProgressBar* pbar = new ProgressBar(inputTree->GetEntries());
00035         for ( int entry = 0; entry < inputTree->GetEntries(); entry++ ) {
00036             pbar->printProgress();
00037             inputTree->GetEntry(entry);
00038             mOutputValue.chipid = mInputValue.chipid;
00039             mOutputValue.timeStamp = mInputValue.timeStamp + preTime;
00040             mOutputValue.x = mInputValue.x;
00041             mOutputValue.y = mInputValue.y;
00042             outputTree->Fill();
00043         }
00044         delete pbar;
00045         preTime = mOutputValue.timeStamp + 1;
00046         inputFile->Close();
00047     }
00048     mOutputFile->Write();
00049 } catch ( MergeFileOpen error ) {
00050     std::cerr << error.what() << std::endl;
00051     exit(1);
00052 } catch ( MergeTreeOpen error ) {
00053     std::cerr << error.what() << std::endl;
00054     exit(1);
00055 }
00056 }
00057
00058 TMergeExperimentROOT::~TMergeExperimentROOT() {
00059     mOutputFile->Close();
00060 }

```

8.43 /home/ychoi/ATOM/alpide/daq/inc/TALPIDEDecoder.h File Reference

```

#include <vector>
#include <filesystem>
#include <cmath>
#include "TDecoder.h"

```

Classes

- class [TALPIDEDecoder](#)

8.44 TALPIDEDecoder.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TALPIDEDECODER__
00002 #define __TALPIDEDECODER__
00003
00004 #ifdef __TALPIDEDECODER_HEADER__
00005 #include<array>
00006
00007 #include "TALPIDEEvent.h"
00008 #endif
00009
00010 #include<vector>
00011 #include<filesystem>

```



```

00012 #include<cmath>
00013
00014 #include "TDecoder.h"
00015
00016 class TALPIDEEvent;
00017
00018 class TALPIDEDecoder : public TDecoder {
00019 private:
00020     std::vector<std::unique_ptr<TALPIDEEvent>> alpides;
00021     int index_ = 0;
00022
00023 public:
00024     TALPIDEDecoder(const std::filesystem::path& binaryPath);
00025     TALPIDEDecoder(const std::string& binaryPath);
00026     void decode();
00027     std::vector<std::unique_ptr<TALPIDEEvent>> getData();
00028 private:
00029     void inputEvent();
00030     bool isDone();
00031     void preTest();
00032     bool strayTest();
00033     void headerTest();
00034     long int hex_to_dec(const uint8_t* data, const int& digits);
00035     int bitwise_and(int v1, int v2);
00036     int bitwise_or(int v1, int v2);
00037     int bitwise_xor(int v1, int v2);
00038 };
00039 #endif

```

8.45 /home/ychoi/ATOM/alpide/daq/inc/TALPIDEEvent.h File Reference

```

#include <utility>
#include <vector>
#include <algorithm>
#include "TEvent.h"

```

Classes

- class [TALPIDEEvent](#)

8.46 TALPIDEEvent.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TALPIDEEVENT__
00002 #define __TALPIDEEVENT__
00003
00004 #ifdef __TALPIDEEVENT_HEADER__
00005 #include <unordered_set>
00006 #endif
00007
00008 #include <utility>
00009 #include <vector>
00010 #include<algorithm>
00011
00012 #include "TEvent.h"
00013
00014 class TALPIDEEvent : public TEvent {
00015 private:
00016     long int iTime;
00017     std::vector<std::pair<int, int>> data;
00018
00019 public:
00020     TALPIDEEvent();
00021     TALPIDEEvent(const TALPIDEEvent& copy);
00022     TALPIDEEvent& operator=(const TALPIDEEvent& copy);
00023     TALPIDEEvent(TALPIDEEvent&& move);
00024     TALPIDEEvent& operator=(TALPIDEEvent&& move);
00025

```

```

00026     // Setter
00027     void setTime(const long int time);
00028
00029     // Getter
00030     const long int getTime() const;
00031     const std::vector<std::pair<int, int>& getData() const;
00032
00033     void removePixel(const std::pair<int, int>& coordinate);
00034     void pushData(const std::pair<int, int>& coordinate);
00035     void removeDuplication();
00036     void sortPixel();
00037
00038     const int getNData() const;
00039
00040 private:
00041     unsigned int fBits;
00042 public:
00043     enum {
00044         kNotDeleted = 0x02000000
00045     };
00046     bool IsDestructed() const { return !TestBit(kNotDeleted); }
00047     bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
00048 };
00049
00050 #endif

```

8.47 /home/ychoi/ATOM/alpide/daq/src/TALPIDEDecoder.cpp File Reference

```
#include "TALPIDEDecoder.h"
```

Macros

- #define [__TALPIDEDECODER_HEADER__](#)

Variables

- const std::array< uint8_t, 16 > [stray_sequence](#) {0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}

8.47.1 Macro Definition Documentation

8.47.1.1 [__TALPIDEDECODER_HEADER__](#)

```
#define __TALPIDEDECODER_HEADER__
```

Definition at line 1 of file [TALPIDEDecoder.cpp](#).

8.47.2 Variable Documentation

8.47.2.1 [stray_sequence](#)

```
const std::array<uint8_t, 16> stray_sequence {0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00, 0x00, 0x00,
0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
```

Definition at line 4 of file [TALPIDEDecoder.cpp](#).

8.48 TALPIDEDecoder.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TALPIDEDECODER_HEADER__
00002 #include "TALPIDEDecoder.h"
00003
00004 const std::array<uint8_t, 16> stray_sequence{0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00, 0x00, 0x00, 0x02,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
00005
00006 TALPIDEDecoder::TALPIDEDecoder(const std::filesystem::path& binaryPath) : TDecoder(binaryPath) { }
00007 TALPIDEDecoder::TALPIDEDecoder(const std::string& binaryPath) : TDecoder(binaryPath) { }
00008
00009 void TALPIDEDecoder::decode() {
00010     preTest();
00011     while ( !isDone() ) {
00012         inputEvent();
00013     }
00014 }
00015
00016 std::vector<std::unique_ptr<TALPIDEEvent> > TALPIDEDecoder::getData() {
00017     return std::move(alpides);
00018 }
00019
00020 void TALPIDEDecoder::inputEvent() {
00021     int numEvent = hex_to_dec(&*getBinaryData().begin() + index_ + 4, 4);
00022     long int time = hex_to_dec(&*getBinaryData().begin() + index_ + 8, 8);
00023     std::unique_ptr<TALPIDEEvent> alpide(new TALPIDEEvent());
00024     alpide->setEvent(numEvent);
00025     alpide->setTime(time);
00026     index_ += 4 * 4;
00027
00028     if ( bitwise_and(getBinaryData()[index_], 0xF0) == 0xE0 ) { // chip has empty frame
00029         if ( (getBinaryData()[index_ + 2] != 0xFF) || getBinaryData()[index_ + 3] != 0xFF ) {
00030             std::cerr << "Assertion Error at empty frame test" << std::endl;
00031             throw std::bad_exception();
00032         }
00033         index_ += 4;
00034     } else if ( bitwise_and(getBinaryData()[index_], 0xF0) == 0xA0 ) {
00035         index_ += 2;
00036         int n = getBinaryData().size();
00037         int reg = 0;
00038         while ( index_ < n ) {
00039             uint8_t data0 = getBinaryData()[index_];
00040             if ( bitwise_and(data0, 0xC0) == 0x00 ) { // data long
00041                 int d = bitwise_or(bitwise_or((reg * pow(2, 14)), bitwise_and(data0, 0x3F) * pow(2,
00042 8)), getBinaryData()[index_ + 1]);
00043                 int x = bitwise_or(bitwise_and((int) (d / pow(2, 9)), 0x3FE),
00044 bitwise_and(bitwise_xor(d, (int) (d / pow(2, 1))), 0x1));
00045                 int y = bitwise_and((int) d / pow(2, 1), 0x1FF);
00046                 alpide->pushData({x, y});
00047                 uint8_t data2 = getBinaryData()[index_ + 2];
00048                 d += 1;
00049                 while ( data2 ) {
00050                     if ( data2 & 1 ) {
00051                         int x = bitwise_or(bitwise_and((int) (d / pow(2, 9)), 0x3FE),
00052 bitwise_and(bitwise_xor(d, (int) (d / pow(2, 1))), 0x1));
00053                         int y = bitwise_and((int) d / pow(2, 1), 0x1FF);
00054                         alpide->pushData({x, y});
00055                     }
00056                     data2 = (int) (data2 / pow(2, 1));
00057                     d += 1;
00058                 }
00059                 index_ += 3;
00060             } else if ( bitwise_and(data0, 0xC0) == 0x40 ) {
00061                 int d = bitwise_or(bitwise_or((reg * pow(2, 14)), bitwise_and(data0, 0x3F) * pow(2,
00062 8)), getBinaryData()[index_ + 1]);
00063                 int x = bitwise_or(bitwise_and((int) (d / pow(2, 9)), 0x3FE),
00064 bitwise_and(bitwise_xor(d, (int) (d / pow(2, 1))), 0x1));
00065                 int y = bitwise_and((int) d / pow(2, 1), 0x1FF);
00066                 alpide->pushData({x, y});
00067                 index_ += 2;
00068             } else if ( bitwise_and(data0, 0xE0) == 0xC0 ) {
00069                 reg = bitwise_and(data0, 0x1F);
00070                 index_ += 1;
00071             } else if ( bitwise_and(data0, 0xF0) == 0xB0 ) {
00072                 index_ += 1;
00073                 index_ = static_cast<int> ((index_ + 3) / 4 * 4);
00074                 break;
00075             } else if ( data0 == 0xFF ) {
00076                 index_ += 1;
00077             } else {
00078                 std::cout << "DEBUG: data[i-10,i+10].:";
00079                 for ( int j = 0; j < 20; j++ ) {
00080                     std::cout << getBinaryData()[index_ + j - 10];
00081                 }
00082             }
00083         }
00084     }
00085 }

```

```

00077         std::cout << "ValueError: invalid literal" << std::endl;
00078         throw std::bad_exception();
00079     }
00080 }
00081 }
00082 index_ += 4;
00083 alpides.push_back(std::move(alpide));
00084 }
00085
00086
00087
00088 void TALPIDEDecoder::preTest() {
00089     if ( index_ == 0 && strayTest() ) {
00090         index_ += 16;
00091     }
00092     headerTest();
00093 }
00094
00095 bool TALPIDEDecoder::strayTest() {
00096     if ( std::equal(getBinaryData().begin(), getBinaryData().begin() + 16, stray_sequence.begin()) ) {
00097         std::cout << "Decoder: stray sequence 0x AA AA AA AA 00 00 00 00 02 00 00 00 00 00 00 00 found.
Skipping 16 bytes.";
00098         return true;
00099     } else {
00100         return false;
00101     }
00102 }
00103
00104 void TALPIDEDecoder::headerTest() {
00105     if ( !std::equal(getBinaryData().begin(), getBinaryData().begin() + 4, stray_sequence.begin()) ) {
00106         std::cerr << "Assertion Error in header, not 0x AA AA AA AA" << std::endl;
00107         throw std::bad_exception();
00108     }
00109 }
00110
00111 long int TALPIDEDecoder::hex_to_dec(const uint8_t* data, const int& digits) {
00112     long int val = 0;
00113     for ( int digit = 0; digit < digits; digit++ ) {
00114         val += data[digit] * pow(256, digit);
00115     }
00116     return val;
00117 }
00118
00119 int TALPIDEDecoder::bitwise_and(int v1, int v2) {
00120     int result = 0;
00121     uint8_t digit = 0;
00122     while ( true ) {
00123         result += (v1 % 2 * v2 % 2) * pow(2, digit);
00124         v1 = v1 / 2;
00125         v2 = v2 / 2;
00126         digit++;
00127         if ( v1 == 0 || v2 == 0 ) break;
00128     }
00129     return result;
00130 }
00131
00132 int TALPIDEDecoder::bitwise_or(int v1, int v2) {
00133     int result = 0;
00134     uint8_t digit = 0;
00135     while ( true ) {
00136         result += ((v1 % 2 - 1) * (v2 % 2 - 1) - 1) * pow(2, digit);
00137         v1 = v1 / 2;
00138         v2 = v2 / 2;
00139         digit++;
00140         if ( v1 == 0 && v2 == 0 ) break;
00141     }
00142     return result;
00143 }
00144
00145 int TALPIDEDecoder::bitwise_xor(int v1, int v2) {
00146     int result = 0;
00147     uint8_t digit = 0;
00148     while ( true ) {
00149         result += (v1 % 2 + v2 % 2) % 2 * pow(2, digit);
00150         v1 = v1 / 2;
00151         v2 = v2 / 2;
00152         digit++;
00153         if ( v1 == 0 && v2 == 0 ) break;
00154     }
00155     return result;
00156 }
00157
00158 bool TALPIDEDecoder::isDone() {
00159     if ( index_ >= getDataLength() ) return true;
00160     else return false;
00161 }

```

8.49 /home/ychoi/ATOM/alpide/daq/src/TALPIDEEvent.cpp File Reference

```
#include "TALPIDEEvent.h"
```

Macros

- `#define __TALPIDEEVENT_HEADER__`

8.49.1 Macro Definition Documentation

8.49.1.1 __TALPIDEEVENT_HEADER__

```
#define __TALPIDEEVENT_HEADER__
```

Definition at line 1 of file [TALPIDEEvent.cpp](#).

8.50 TALPIDEEvent.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TALPIDEEVENT_HEADER__
00002 #include "TALPIDEEvent.h"
00003
00004 TALPIDEEvent::TALPIDEEvent() : TEvent(), fBits(kNotDeleted) { }
00005
00006 TALPIDEEvent::TALPIDEEvent(const TALPIDEEvent& copy) : iTime(copy.iTime), fBits(copy.fBits) {
00007     setEvent(copy.getEvent());
00008     data.assign(copy.data.begin(), copy.data.end());
00009 }
00010
00011 TALPIDEEvent& TALPIDEEvent::operator=(const TALPIDEEvent& copy) {
00012     setEvent(copy.getEvent());
00013     fBits = copy.fBits;
00014     iTime = copy.iTime;
00015     data.assign(copy.data.begin(), copy.data.end());
00016     return *this;
00017 }
00018
00019 TALPIDEEvent::TALPIDEEvent(TALPIDEEvent&& move) : iTime(move.iTime), fBits(move.fBits) {
00020     setEvent(move.getEvent());
00021     data.assign(move.data.begin(), move.data.end());
00022     move.fBits = 0;
00023     move.setEvent(0);
00024     move.iTime = 0;
00025     move.data.clear();
00026 }
00027
00028 TALPIDEEvent& TALPIDEEvent::operator=(TALPIDEEvent&& move) {
00029     setEvent(move.getEvent());
00030     fBits = move.fBits;
00031     iTime = move.iTime;
00032     data.assign(move.data.begin(), move.data.end());
00033     move.setEvent(0);
00034     move.fBits = 0;
00035     move.iTime = 0;
00036     move.data.clear();
00037     return *this;
00038 }
00039
00040 void TALPIDEEvent::setTime(const long int time) {
00041     iTime = time;
00042 }
00043
00044 void TALPIDEEvent::pushData(const std::pair<int, int>& coordinate) {
00045     data.push_back(std::move(coordinate));
```

```

00046 }
00047
00048 const long int TALPIDEEvent::getTime() const {
00049     return iTime;
00050 }
00051
00052 const std::vector<std::pair<int, int>& TALPIDEEvent::getData() const {
00053     return data;
00054 }
00055
00056 void TALPIDEEvent::removePixel(const std::pair<int, int>& coordinate) {
00057     data.erase(std::find(data.begin(), data.end(), coordinate));
00058 }
00059
00060 void TALPIDEEvent::removeDuplication() {
00061     std::vector<std::pair<int, int>& uniqueData;
00062
00063     auto arrayHash = [ ](const std::pair<int, int>& arr) {
00064         return std::hash<int>() (arr.first) ^ (std::hash<int>() (arr.second) « 1);
00065     };
00066
00067     std::unordered_set<std::pair<int, int>, decltype(arrayHash)> uniqueSet(data.size() * 2,
arrayHash);
00068
00069     for ( const auto& arr : data ) {
00070         if ( uniqueSet.find(arr) == uniqueSet.end() ) {
00071             uniqueData.push_back(arr);
00072             uniqueSet.insert(arr);
00073         }
00074     }
00075     data = std::move(uniqueData);
00076 }
00077
00078 void TALPIDEEvent::sortPixel() {
00079     std::sort(data.begin(), data.end(), [ ](const std::pair<int, int>& a, const std::pair<int, int>&
b) {
00080         if ( a.second != b.second ) {
00081             return a.second < b.second; // Sort by first element in ascending order
00082         }
00083         return a.first < b.first; // If the first elements are equal, sort by second element in
ascending order
00084     });
00085 }
00086
00087 const int TALPIDEEvent::getNData() const {
00088     return data.size();
00089 }

```

8.51 /home/ychoi/ATOM/alpide/threshold/inc/TThreshold.h File Reference

```

#include <array>
#include <algorithm>

```

Classes

- class [TThreshold](#)

Enumerations

- enum [ThrCondition](#) { [good](#), [bad_too_low](#), [bad_too_high](#), [bad_undefine](#) }

8.51.1 Enumeration Type Documentation

8.51.1.1 ThrCondition

```
enum ThrCondition
```

Enumerator

good	
bad_too_low	
bad_too_high	
bad_undefine	

Definition at line 17 of file [TThreshold.h](#).

8.52 TThreshold.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TTHRESHOLD__
00002 #define __TTHRESHOLD__
00003
00004 #ifdef __TTHRESHOLD_HEADER__
00005 #include<numeric>
00006 #include "TCanvas.h"
00007 #include "TGraph.h"
00008 #include "TFl.h"
00009 #endif
00010
00011 #include<array>
00012 #include<algorithm>
00013
00014 class TGraph;
00015 class TFl;
00016
00017 enum ThrCondition {
00018     good,
00019     bad_too_low,
00020     bad_too_high,
00021     bad_undefine
00022 };
00023
00024 class TThreshold {
00025 private:
00026     int mX;
00027     int mY;
00028     std::array<int, 50> mDacs;
00029     double mThr;
00030     double mErr;
00031     double mQualityFactor;
00032
00033     ThrCondition mCondition;
00034
00035     std::unique_ptr<TGraph> thresholdGraph;
00036     std::unique_ptr<TFl> fitFunction;
00037 public:
00038     //Constructor
00039     TThreshold();
00040     TThreshold(int x, int y);
00041     TThreshold(const std::array<int, 2>& coordinate);
00042     TThreshold(int x, int y, const std::array<int, 50>& dacs);
00043     TThreshold(const std::array<int, 2>& coordrdinate, const std::array<int, 50>& dacs);
00044     TThreshold(int x, int y, std::array<int, 50>&& dacs);
00045     TThreshold(const std::array<int, 2>& coordrdinate, std::array<int, 50>&& dacs);
00046     //Copy Constructor
00047     TThreshold(const TThreshold& copy);
00048     //Copy Assignment
00049     TThreshold& operator=(const TThreshold& copy);
00050     //Move Constructor
00051     TThreshold(TThreshold&& move);
00052     //Move Assignment
00053     TThreshold& operator=(TThreshold&& move);
00054     //Destructor
00055     ~TThreshold();
00056
00057     ThrCondition calculateThreshold();
00058     void savePlot();
00059
00060     const double getX() const;
00061     const double getY() const;
00062     const double getThreshold() const;
00063     const double getError() const;

```

```

00064     const double getQualityFactor() const;
00065
00066     const ThrCondition getCondition() const;
00067 };
00068
00069 #endif

```

8.53 /home/ychoi/ATOM/alpide/threshold/inc/TThresholdCompare.h File Reference

```

#include <vector>
#include <memory>

```

Classes

- class [TThresholdCompare](#)

8.54 TThresholdCompare.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TTHRESHOLDCOMPARE__
00002 #define __TTHRESHOLDCOMPARE__
00003
00004 #ifdef __TTHRESHOLDCOMPARE_HEADER__
00005 #endif
00006
00007 #include <vector>
00008 #include <memory>
00009
00010 class TThreshold;
00011
00012 // #include "Headers.h"
00013 // #include "TThreshold.h"
00014
00015 class TThresholdCompare {
00016 private:
00017     std::vector<std::unique_ptr<TThreshold*>> thresholds_;
00018 public:
00019     TThresholdCompare();
00020     TThresholdCompare(std::vector<std::unique_ptr<TThreshold*>> thresholds);
00021
00022 };
00023
00024
00025 #endif

```

8.55 /home/ychoi/ATOM/alpide/threshold/src/TThreshold.cpp File Reference

```

#include "TThreshold.h"

```

Macros

- #define [__TTHRESHOLD_HEADER__](#)

8.55.1 Macro Definition Documentation

8.55.1.1 __TTHRESHOLD_HEADER__

```
#define __TTHRESHOLD_HEADER__
```

Definition at line 1 of file [TThreshold.cpp](#).

8.56 TThreshold.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TTHRESHOLD_HEADER__
00002 #include "TThreshold.h"
00003
00004 TThreshold::TThreshold() { }
00005
00006 TThreshold::TThreshold(int x, int y) : mX(x), mY(y) { }
00007
00008 TThreshold::TThreshold(const std::array<int, 2>& coordinate) : mX(coordinate[0]), mY(coordinate[1]) {
00009 }
00010 TThreshold::TThreshold(int x, int y, const std::array<int, 50>& dacs) : mX(x), mY(y) {
00011     std::copy(std::begin(dacs), std::end(dacs), std::begin(mDacs));
00012     mCondition = calculateThreshold();
00013 }
00014
00015 TThreshold::TThreshold(const std::array<int, 2>& coordinate, const std::array<int, 50>& dacs) :
00016     mX(coordinate[0]), mY(coordinate[1]) {
00017     std::copy(std::begin(dacs), std::end(dacs), std::begin(mDacs));
00018 }
00019 TThreshold::TThreshold(int x, int y, std::array<int, 50>&& dacs) : mX(x), mY(y) {
00020     std::move(std::begin(dacs), std::end(dacs), std::begin(mDacs));
00021 }
00022
00023 TThreshold::TThreshold(const std::array<int, 2>& coordinate, std::array<int, 50>&& dacs) :
00024     mX(coordinate[0]), mY(coordinate[1]) {
00025     std::move(std::begin(dacs), std::end(dacs), std::begin(mDacs));
00026 }
00027 TThreshold::TThreshold(const TThreshold& copy) : mX(copy.mX), mY(copy.mY), mThr(copy.mThr),
00028     mErr(copy.mErr) {
00029     std::copy(std::begin(copy.mDacs), std::end(copy.mDacs), std::begin(mDacs));
00030 }
00031 TThreshold& TThreshold::operator=(const TThreshold& copy) {
00032     mX = copy.mX;
00033     mY = copy.mY;
00034     std::copy(std::begin(copy.mDacs), std::end(copy.mDacs), std::begin(mDacs));
00035     mThr = copy.mThr;
00036     mErr = copy.mErr;
00037     return *this;
00038 }
00039
00040 TThreshold::TThreshold(TThreshold&& move) : mX(move.mX), mY(move.mY), mThr(move.mThr), mErr(move.mErr)
00041 {
00042     std::move(std::begin(move.mDacs), std::end(move.mDacs), std::begin(mDacs));
00043 }
00044 TThreshold& TThreshold::operator=(TThreshold&& move) {
00045     mX = move.mX;
00046     mY = move.mY;
00047     std::move(std::begin(move.mDacs), std::end(move.mDacs), std::begin(mDacs));
00048     mThr = move.mThr;
00049     mErr = move.mErr;
00050     return *this;
00051 }
00052
00053 TThreshold::~TThreshold() { }
00054
00055 ThrCondition TThreshold::calculateThreshold() {
00056     if ( *std::begin(mDacs) > 25 ) {
00057         mThr = 0;
00058         mErr = 0;
00059         return ThrCondition::bad_too_low;
00060     } else if ( *(std::end(mDacs) - 1) < 25 ) {
00061         mThr = 100;
```

```

00062         mErr = 100;
00063         return ThrCondition::bad_too_high;
00064     } else {
00065         std::array<int, 50> adcs;
00066         std::iota(std::begin(adcs), std::end(adcs), 1);
00067         thresholdGraph.reset(new TGraph(50, std::begin(adcs), std::begin(mDacs)));
00068         fitFunction.reset(new TF1("fitFunction", "[0]*TMath::Erf((x-[1])/[2])+[3]", 0., 50.));
00069         bool quality = false;
00070         int count = 0;
00071         while ( !quality ) {
00072             fitFunction->SetParameters(20, 10 * count + 10, 10, 25);
00073             thresholdGraph->Fit("fitFunction", "q");
00074             mThr = fitFunction->GetParameter(1);
00075             mErr = fitFunction->GetParameter(2);
00076             mQualityFactor = fitFunction->GetChisquare() / fitFunction->GetNDF();
00077             if ( mQualityFactor < 100. ) {
00078                 quality = true;
00079             }
00080             if ( count > 4 ) {
00081                 mThr = -1;
00082                 mErr = -1;
00083                 return ThrCondition::bad_undefine;
00084             }
00085             count++;
00086         }
00087         if ( mThr > 0 && mThr < 50 ) {
00088             return ThrCondition::good;
00089         } else {
00090             return ThrCondition::bad_undefine;
00091         }
00092     }
00093 }
00094
00095 void TThreshold::savePlot() {
00096     std::unique_ptr<TCanvas> can(new TCanvas("can", "can", 500, 500));
00097     thresholdGraph->SetTitle(static_cast<TString>("Threshold Graph at " + std::to_string(mX) + ", " +
00098     std::to_string(mY) + "; ADC[$500 \times e^{-}$]; DAC[# of Fire]"));
00099     thresholdGraph->Draw();
00100     // can->SaveAs(static_cast<TString>("data/" + std::to_string(mX) + "_" + std::to_string(mY) +
00101     ".png"));
00102 }
00103
00102 const double TThreshold::getX() const {
00103     return mX;
00104 }
00105
00106 const double TThreshold::getY() const {
00107     return mY;
00108 }
00109
00110 const double TThreshold::getThreshold() const {
00111     return mThr;
00112 }
00113
00114 const double TThreshold::getError() const {
00115     return mErr;
00116 }
00117
00118 const double TThreshold::getQualityFactor() const {
00119     return mQualityFactor;
00120 }
00121
00122 const ThrCondition TThreshold::getCondition() const {
00123     return mCondition;
00124 }

```

8.57 /home/ychoi/ATOM/alpide/threshold/src/TThresholdCompare.cpp

File Reference

```
#include "TThresholdCompare.h"
```

Macros

- `#define __TTHRESHOLDCOMPARE_HEADER__`

8.57.1 Macro Definition Documentation

8.57.1.1 __TTHRESHOLDCOMPARE_HEADER__

```
#define __TTHRESHOLDCOMPARE_HEADER__
```

Definition at line 1 of file [TThresholdCompare.cpp](#).

8.58 TThresholdCompare.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TTHRESHOLDCOMPARE_HEADER__
00002 #include "TThresholdCompare.h"
00003
00004 TThresholdCompare::TThresholdCompare() { }
00005 TThresholdCompare::TThresholdCompare(std::vector<std::unique_ptr<TThreshold*> thresholds) {
00006     thresholds_ = std::move(thresholds);
00007 }
00008
```

8.59 /home/ychoi/ATOM/apts/inc/TAPTSDecoder.h File Reference

```
#include <vector>
#include <array>
#include <string>
#include <cmath>
#include "TDecoder.h"
```

Classes

- class [TAPTSDecoder](#)

8.60 TAPTSDecoder.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TAPTSDECODER__
00002 #define __TAPTSDECODER__
00003
00004 #ifdef __TAPTSDECODER_HEADER__
00005 #include "TAPTSEvent.h"
00006 #endif
00007
00008 #include<vector>
00009 #include<array>
00010 #include<string>
00011 #include<cmath>
00012
00013 #include "TDecoder.h"
00014
00015 class TAPTSEvent;
00016
00017 class TAPTSDecoder : public TDecoder {
00018 private:
00019     std::vector<TAPTSEvent*> aptss;
00020     bool mux_ = false;
00021     int iEvent_ = 0;
00022     std::array<int, 16> mapping;
```

```

00023     std::array<uint8_t, 4> expected_header = {0xAA, 0xAA, 0xAA, 0xAA};
00024     std::array<uint8_t, 4> expected_footer = {0xBB, 0xBB, 0xBB, 0xBB};
00025
00026     int nFrame_;
00027     int index_ = 0;
00028     int lenEvent_ = 0;
00029
00030 public:
00031     TAPTSDecoder(const std::filesystem::path& binaryPath);
00032     TAPTSDecoder(const std::string& binaryPath);
00033     void decode();
00034     std::vector<TAPTSEvent*> getData();
00035 private:
00036     void inputEvent();
00037     void preTest(); // do range, header, missing test and calculate data length of Event.
00038     void rangeTest(); // Test whether the index number is overflowed or not.
00039     void headerTest(); // Test whether an event is normal or not. The first 4 num should be 0xAA.
00040     void missingTest(); // Test whether the data is overflowed or not.
00041     int getEventLength(); // Calculate the data length of an event.
00042     void postTest();
00043     bool isDone();
00044 };
00045 #endif

```

8.61 /home/ychoi/ATOM/apts/inc/TAPTSEvent.h File Reference

```

#include "TEvent.h"
#include <array>

```

Classes

- class [TAPTSEvent](#)

8.62 TAPTSEvent.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TAPTSEVENT__
00002 #define __TAPTSEVENT__
00003
00004 #include "TEvent.h"
00005
00006 #include <array>
00007
00008 class TAPTSEvent : public TEvent {
00009 private:
00010     int iFrame;
00011     std::array<int, 16> data{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
00012
00013 public:
00014     TAPTSEvent();
00015
00016     // Setter
00017     void setFrame(int frame);
00018
00019     // Getter
00020     int getFrame();
00021     std::array<int, 16>& getData();
00022 };
00023
00024 #endif

```

8.63 /home/ychoi/ATOM/apts/src/TAPTSDecoder.cpp File Reference

```

#include "TAPTSDecoder.h"

```

Macros

- `#define __TAPTSDECODER_HEADER__`

Variables

- `const int APTS_PIXEL_ADC_MAPPING[16] = {2, 1, 5, 0, 4, 8, 9, 12, 13, 14, 10, 15, 11, 7, 6, 3}`
- `const int APTS_MUX_PIXEL_ADC_MAPPING[16] = {3, 2, 1, 0, 4, 8, 9, 5, 12, 13, 14, 15, 10, 11, 6, 7}`

8.63.1 Macro Definition Documentation

8.63.1.1 __TAPTSDECODER_HEADER__

```
#define __TAPTSDECODER_HEADER__
```

Definition at line 1 of file [TAPTSDecoder.cpp](#).

8.63.2 Variable Documentation

8.63.2.1 APTS_MUX_PIXEL_ADC_MAPPING

```
const int APTS_MUX_PIXEL_ADC_MAPPING[16] = {3, 2, 1, 0, 4, 8, 9, 5, 12, 13, 14, 15, 10, 11, 6, 7}
```

Definition at line 5 of file [TAPTSDecoder.cpp](#).

8.63.2.2 APTS_PIXEL_ADC_MAPPING

```
const int APTS_PIXEL_ADC_MAPPING[16] = {2, 1, 5, 0, 4, 8, 9, 12, 13, 14, 10, 15, 11, 7, 6, 3}
```

Definition at line 4 of file [TAPTSDecoder.cpp](#).

8.64 TAPTSDecoder.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TAPTSDECODER_HEADER__
00002 #include "TAPTSDecoder.h"
00003
00004 const int APTS_PIXEL_ADC_MAPPING[16] = {2, 1, 5, 0, 4, 8, 9, 12, 13, 14, 10, 15, 11, 7, 6, 3};
00005 const int APTS_MUX_PIXEL_ADC_MAPPING[16] = {3, 2, 1, 0, 4, 8, 9, 5, 12, 13, 14, 15, 10, 11, 6, 7};
00006
00007 TAPTSDecoder::TAPTSDecoder(const std::filesystem::path& binaryPath) : TDecoder(binaryPath) { }
00008 TAPTSDecoder::TAPTSDecoder(const std::string& binaryPath) : TDecoder(binaryPath) { }
00009
00010 void TAPTSDecoder::decode() {
00011     mux_ ? std::copy(std::begin(APTS_MUX_PIXEL_ADC_MAPPING), std::end(APTS_MUX_PIXEL_ADC_MAPPING),
00012                     mapping.begin())
00013         : std::copy(std::begin(APTS_PIXEL_ADC_MAPPING), std::end(APTS_PIXEL_ADC_MAPPING),
00014                     mapping.begin());
00015     // Set mapping array.
00016     while ( !isDone() ) {
00017         inputEvent();
00018     }
00019 }
```

```

00019
00020 std::vector<TAPTSEvent*> TAPTSDecoder::getData() {
00021     return aptss;
00022 }
00023
00024 void TAPTSDecoder::inputEvent() {
00025     preTest();
00026
00027     std::vector<uint8_t> eventData(getBinaryData().begin() + index_, getBinaryData().begin() + index_
+ lenEvent_);
00028     // A part of binary data.
00029
00030     nFrame_ = lenEvent_ / 40;
00031     // Set number of frame.
00032
00033     for ( int iFrame = 0; iFrame < nFrame_; iFrame++ ) {
00034         TAPTSEvent* apt = new TAPTSEvent();
00035         apt->setEvent(iEvent_);
00036         for ( int j = 0; j < 16; j++ ) {
00037             int share1 = eventData[iFrame * 40 + 2 * j];
00038             int share2 = eventData[iFrame * 40 + 2 * j + 1];
00039             int rest1 = 0;
00040             int rest2 = 0;
00041             for ( int digit = 0; digit < 8; digit++ ) {
00042                 rest1 = share1 % 2;
00043                 rest2 = share2 % 2;
00044                 apt->getData()[mapping[digit]] += rest1 * pow(2, 15 - j);
00045                 apt->getData()[mapping[digit + 8]] += rest2 * pow(2, 15 - j);
00046                 share1 = ceil(share1 / 2);
00047                 share2 = ceil(share2 / 2);
00048             }
00049         }
00050         apt->setFrame(iFrame);
00051
00052         int trailer = eventData[iFrame * 40 + 39] + eventData[iFrame * 40 + 38] * 256;
00053         if ( iFrame < nFrame_ - 1 ) {
00054             if ( trailer != 0xFEFE ) std::cout << "Unexpected frame trailer 0x" << std::hex << trailer <<
" for frame " << iFrame + 1 << " out of " << std::dec << nFrame_ << std::endl;
00055         } else {
00056             if ( trailer != 0xAEAE ) std::cout << "Unexpected event trailer 0x" << std::hex << trailer <<
" for frame " << iFrame + 1 << " out of " << std::dec << nFrame_ << std::endl;
00057         }
00058         aptss.push_back(apt);
00059     }
00060     index_ += lenEvent_;
00061     postTest();
00062     iEvent_++;
00063 }
00064
00065 void TAPTSDecoder::postTest() {
00066     if ( index_ + 12 <= getDataLength() && std::equal(getBinaryData().begin() + index_,
getBinaryData().begin() + index_ + 3, expected_footer.begin()) ) {
00067         index_ += 16;
00068     }
00069 }
00070
00071 void TAPTSDecoder::preTest() {
00072     rangeTest();
00073     headerTest();
00074     lenEvent_ = getEventLength();
00075     index_ += 8;
00076     missingTest();
00077 }
00078
00079 void TAPTSDecoder::rangeTest() {
00080     if ( !(index_ + 8 <= getDataLength()) ) {
00081         std::cerr << "Ev " << iEvent_ << ": Unexpected data length(" << getDataLength() - 1 << " < 8)" <<
std::endl;
00082     }
00083 }
00084
00085 void TAPTSDecoder::headerTest() {
00086     if ( !std::equal(getBinaryData().begin() + index_, getBinaryData().begin() + index_ + 3,
expected_header.begin()) ) {
00087         std::cout << "Ev " << iEvent_ << std::hex << int(getBinaryData()[index_]) << "-" <<
int(getBinaryData()[index_ + 1]) << "-" << int(getBinaryData()[index_ + 2]) << "-" <<
int(getBinaryData()[index_ + 3]) << std::endl;
00088     }
00089 }
00090
00091 void TAPTSDecoder::missingTest() {
00092     if ( !(index_ + lenEvent_ <= getDataLength()) ) {
00093         std::cerr << "Ev " << iEvent_ << ": Missing data (" << index_ << " " << lenEvent_ << " " <<
getDataLength() << ")" << std::endl;
00094     }
00095 }
00096

```

```

00097 int TAPTSEvent::getEventLength() {
00098     int lenEvent_ = 0;
00099     for ( int i = 0; i < 4; i++ ) {
00100         lenEvent_ += getBinaryData()[index_ + 4 + i] * pow(256, i);
00101     }
00102     return lenEvent_;
00103 }
00104
00105 bool TAPTSEvent::isDone() {
00106     if ( index_ >= getDataLength() ) return true;
00107     else return false;
00108 }

```

8.65 /home/ychoi/ATOM/apts/src/TAPTSEvent.cpp File Reference

```
#include "TAPTSEvent.h"
```

8.66 TAPTSEvent.cpp

[Go to the documentation of this file.](#)

```

00001 #include "TAPTSEvent.h"
00002
00003 TAPTSEvent::TAPTSEvent() : TEvent() { }
00004
00005 void TAPTSEvent::setFrame(int frame) {
00006     iFrame = frame;
00007 }
00008
00009 int TAPTSEvent::getFrame() {
00010     return iFrame;
00011 }
00012
00013 std::array<int, 16>& TAPTSEvent::getData() {
00014     return data;
00015 }

```

8.67 /home/ychoi/ATOM/chip/inc/TDecoder.h File Reference

```

#include <iostream>
#include <fstream>
#include <vector>
#include <filesystem>
#include <string>

```

Classes

- [class TDecoder](#)

8.68 TDecoder.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TDECODER__
00002 #define __TDECODER__
00003
00004 #ifdef __TDECODER__
00005 #include<iostream>
00006 #endif
00007
00008 #include<fstream>
00009 #include<vector>
00010 #include<filesystem>
00011 #include<string>
00012
00013 class TDecoder {
00014 private:
00015     std::ifstream binaryFile_;
00016     int dataLength_;
00017     std::vector<uint8_t> binaryData_;
00018 public:
00019     //Constructor
00020     TDecoder(const std::filesystem::path& binaryPath);
00021     TDecoder(const std::string& binaryPath);
00022
00023     void readFile();
00024
00025     int getDataLength();
00026     std::vector<uint8_t>& getBinaryData();
00027 };
00028
00029 #endif
```

8.69 /home/ychoi/ATOM/chip/inc/TEvent.h File Reference

Classes

- class [TEvent](#)

8.70 TEvent.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TEVENT__
00002 #define __TEVENT__
00003
00004 class TEvent {
00005 private:
00006     int iEvent;
00007
00008 public:
00009     TEvent();
00010     virtual ~TEvent();
00011     void setEvent(const int event);
00012     const int getEvent() const;
00013 };
00014
00015 #endif
```

8.71 /home/ychoi/ATOM/chip/src/TDecoder.cpp File Reference

```
#include "TDecoder.h"
```


8.72 TDecoder.cpp

[Go to the documentation of this file.](#)

```
00001 #include "TDecoder.h"
00002
00003 TDecoder::TDecoder(const std::filesystem::path& path) {
00004     if ( !std::filesystem::exists(path) ) {
00005         std::cerr << "File " << path << " cannot be found." << std::endl;
00006     }
00007     binaryFile_ = std::ifstream(path, std::ios::binary);
00008 }
00009
00010 TDecoder::TDecoder(const std::string& path) {
00011     if ( !std::filesystem::exists(path) ) {
00012         std::cerr << "File " << path << " cannot be found." << std::endl;
00013     }
00014     binaryFile_ = std::ifstream(path, std::ios::binary);
00015 }
00016
00017 void TDecoder::readFile() {
00018     char buf[sizeof(uint8_t)];
00019     while ( binaryFile_.read(buf, sizeof(buf)) ) {
00020         binaryData_.push_back(static_cast<uint8_t>(*buf));
00021     }
00022     dataLength_ = binaryData_.size();
00023 }
00024
00025 int TDecoder::getDataLength() {
00026     return dataLength_;
00027 }
00028
00029 std::vector<uint8_t>& TDecoder::getBinaryData() {
00030     return binaryData_;
00031 }
```

8.73 /home/ychoi/ATOM/chip/src/TEvent.cpp File Reference

```
#include "TEvent.h"
```

8.74 TEvent.cpp

[Go to the documentation of this file.](#)

```
00001 #include "TEvent.h"
00002
00003 TEvent::TEvent() { }
00004
00005 TEvent::~TEvent() { }
00006
00007 void TEvent::setEvent(const int event) {
00008     iEvent = event;
00009 }
00010
00011 const int TEvent::getEvent() const {
00012     return iEvent;
00013 }
```

8.75 /home/ychoi/ATOM/drawing_tool/inc/TGraphUser.h File Reference

```
#include "CppConfigFile.h"
#include "TCanvas.h"
#include "TGraph.h"
#include "TGraphErrors.h"
#include "TLegend.h"
#include "TAxis.h"
#include "TMultiGraph.h"
#include <filesystem>
```

Classes

- class [TGraphUser](#)

The general tools for drawing TGraph Class with config file.

8.76 TGraphUser.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TGRAPHUSER__
00002 #define __TGRAPHUSER__
00003
00004 #include "CppConfigFile.h"
00005 #include "TCanvas.h"
00006 #include "TGraph.h"
00007 #include "TGraphErrors.h"
00008 #include "TLegend.h"
00009 #include "TAxis.h"
00010 #include "TMultiGraph.h"
00011 #include<filesystem>
00012
00017 class TGraphUser : public TGraph {
00018 private:
00019     TCanvas* mCanvas; // Canvas for graph
00020     TLegend* mLegend; // Legend
00021     TMultiGraph* mMultiGraph; // Multi-graph
00022 private:
00023     TString savePath; // Save path and file name of graph
00024     std::string mFileName;
00025
00026     TString title, x_title, y_title; // Graph title, x-axis and y-axis titles
00027     std::vector<std::tuple<TGraph*, Style_t, Size_t, Style_t, Size_t> > graphSetAndAttribute; // Graph,
marker style, marker size, line style, line size
00028
00029     std::array<Float_t, 4> canvasMargins; // Left, Right, Bottom, Top
00030     std::array<Float_t, 4> canvasOffsets;
00031
00032     Style_t mUniversalMarkerStyle;
00033     Size_t mUniversalMarkerSize;
00034     Style_t mUniversalLineStyle;
00035     Size_t mUniversalLineWidth;
00036
00037     TString mLegendTitle; // Title of legend
00038     std::array<Float_t, 4> mLegendPoints; // The corner points of legend
00039 public:
00040     TGraphUser(const CppConfigDictionary& config);
00041     void AddGraph(TGraph* graph);
00042     void AddGraph(TGraph* graph, const CppConfigDictionary& graphConfig);
00043     void Save(const std::filesystem::path& outputPath = ".");
00044 private:
00045     void setCanvas(const CppConfigDictionary& config);
00046     void setLegend(const CppConfigDictionary& config);
00047
00048 };
00049
00050
00051 #endif
```

8.77 /home/ychoi/ATOM/drawing_tool/inc/TH1User.h File Reference

8.78 TH1User.h

[Go to the documentation of this file.](#)

8.79 /home/ychoi/ATOM/drawing_tool/src/TGraphUser.cpp File Reference

```
#include "TGraphUser.h"
```

Variables

- `std::unordered_map<std::string, Color_t> colourSet = {"red", kRed}, {"green", kGreen + 2}, {"blue", kBlue}, {"magenta", kMagenta}, {"cyan", kCyan}, {"orange", kOrange + 1}, {"yellow", kYellow + 1}`

8.79.1 Variable Documentation

8.79.1.1 colourSet

```
std::unordered_map<std::string, Color_t> colourSet = {"red", kRed}, {"green", kGreen + 2}, {"blue", kBlue}, {"magenta", kMagenta}, {"cyan", kCyan}, {"orange", kOrange + 1}, {"yellow", kYellow + 1}}
```

Definition at line 3 of file [TGraphUser.cpp](#).

8.80 TGraphUser.cpp

[Go to the documentation of this file.](#)

```
00001 #include "TGraphUser.h"
00002
00003 std::unordered_map<std::string, Color_t> colourSet = {"red", kRed}, {"green", kGreen + 2}, {"blue",
kBlue}, {"magenta", kMagenta}, {"cyan", kCyan}, {"orange", kOrange + 1}, {"yellow", kYellow + 1}};
00004
00010 TGraphUser::TGraphUser(const CppConfigDictionary& config) {
00011     mFileName = config.find("file_name");
00012     mMultiGraph = new TMultiGraph();
00013     setCanvas(config);
00014     setLegend(config);
00015 }
00016
00017 void TGraphUser::Save(const std::filesystem::path& outputPath) {
00018     mCanvas->cd();
00019     mMultiGraph->SetTitle(title + ";" + x_title + ";" + y_title);
00020     mMultiGraph->GetXaxis()->SetLabelOffset(canvasOffsets[0]);
00021     mMultiGraph->GetYaxis()->SetLabelOffset(canvasOffsets[1]);
00022     mMultiGraph->GetXaxis()->SetTitleOffset(canvasOffsets[2]);
00023     mMultiGraph->GetYaxis()->SetTitleOffset(canvasOffsets[3]);
00024     mMultiGraph->Draw("AL");
00025     mCanvas->SetMargin(canvasMargins[0], canvasMargins[1], canvasMargins[2], canvasMargins[3]);
00026     mLegend->Draw();
00027     std::filesystem::path output = outputPath;
00028     output /= mFileName;
00029     savePath = output;
00030     mCanvas->SaveAs(savePath);
00031 }
00032
00033 void TGraphUser::AddGraph(TGraph* graph) {
00034     mLegend->AddEntry(graph, "Graph", "lp");
00035     mMultiGraph->Add(graph);
00036 }
00037
00038 void TGraphUser::AddGraph(TGraph* graph, const CppConfigDictionary& graphConfig) {
00039     TString legendLabel = graphConfig.hasKey("legend_label") ? graphConfig.find("legend_label") :
"graph";
00040
00041     Style_t lineStyle = graphConfig.hasKey("line_style") ? stof(graphConfig.find("line_style")) :
mUniversalLineStyle;
00042     Size_t lineWidth = graphConfig.hasKey("line_width") ? stof(graphConfig.find("line_width")) :
mUniversalLineWidth;
00043     Color_t lineColour = graphConfig.hasKey("line_colour") ?
colourSet.find(graphConfig.find("line_colour"))->second : 0;
00044     graph->SetLineStyle(lineStyle);
00045     graph->SetLineWidth(lineWidth);
00046     graph->SetLineColor(lineColour);
00047
00048     Style_t markerStyle = graphConfig.hasKey("marker_style") ? stof(graphConfig.find("marker_style"))
: mUniversalMarkerStyle;
00049     Size_t markerSize = graphConfig.hasKey("marker_size") ? stof(graphConfig.find("marker_size")) :
mUniversalMarkerSize;
00050     Color_t markerColour = graphConfig.hasKey("marker_colour") ?
colourSet.find(graphConfig.find("marker_colour"))->second : 0;
```

```

00051     graph->SetMarkerStyle(markerStyle);
00052     graph->SetMarkerSize(markerSize);
00053     graph->SetMarkerColor(markerColour);
00054
00055     TString graphOption = graphConfig.hasKey("graph_option") ? graphConfig.find("graph_option") :
    "pl";
00056     mLegend->AddEntry(graph, legendLabel, graphOption);
00057     mMultiGraph->Add(graph, graphOption);
00058 }
00059
00060
00068 void TGraphUser::setCanvas(const CppConfigDictionary& config) {
00069     Int_t canvasRow = config.hasKey("canvas_row") ? stoi(config.find("canvas_row")) : 1000;
00070     Int_t canvasWidth = config.hasKey("canvas_width") ? stoi(config.find("canvas_width")) : 1000;
00071
00072     mCanvas = new TCanvas();
00073     mCanvas->SetCanvasSize(canvasRow, canvasWidth);
00074
00075     // Set Canvas Titles
00076     if ( config.hasKey("titles") ) {
00077         CppConfigDictionary titles = config.getSubConfig("titles");
00078         title = titles.hasKey("title") ? titles.find("title") : "";
00079         x_title = titles.hasKey("x_title") ? titles.find("x_title") : "";
00080         y_title = titles.hasKey("y_title") ? titles.find("y_title") : "";
00081     }
00082
00083     // Set Canvas Margin
00084     if ( config.hasKey("canvas_margin") ) {
00085         CppConfigDictionary canvasMargin = config.getSubConfig("canvas_margin");
00086         Float_t marginLeft = canvasMargin.hasKey("left") ? stod(canvasMargin.find("left")) : .1;
00087         Float_t marginRight = canvasMargin.hasKey("right") ? stod(canvasMargin.find("right")) : .1;
00088         Float_t marginTop = canvasMargin.hasKey("top") ? stod(canvasMargin.find("top")) : .1;
00089         Float_t marginBottom = canvasMargin.hasKey("bottom") ? stod(canvasMargin.find("bottom")) : .1;
00090         canvasMargins = {marginLeft, marginRight, marginBottom, marginTop};
00091     } else {
00092         canvasMargins = {.1, .1, .1, .1};
00093     }
00094
00095     // Set Canvas Offsets
00096     if ( config.hasKey("offsets") ) {
00097         CppConfigDictionary canvasOffset = config.getSubConfig("offsets");
00098         Float_t xLabelOffset = canvasOffset.hasKey("x_label") ? stod(canvasOffset.find("x_label")) :
    .01;
00099         Float_t yLabelOffset = canvasOffset.hasKey("y_label") ? stod(canvasOffset.find("y_label")) :
    .01;
00100         Float_t xTitleOffset = canvasOffset.hasKey("x_title") ? stod(canvasOffset.find("x_title")) :
    1.;
00101         Float_t yTitleOffset = canvasOffset.hasKey("y_title") ? stod(canvasOffset.find("y_title")) :
    1.;
00102         canvasOffsets = {xLabelOffset, yLabelOffset, xTitleOffset, yTitleOffset};
00103     } else {
00104         canvasOffsets = {.01, .01, 1., 1.};
00105     }
00106
00107     mUniversalMarkerStyle = config.hasKey("marker_style") ? stoi(config.find("marker_style")) : 1;
00108     mUniversalMarkerSize = config.hasKey("marker_size") ? stoi(config.find("marker_size")) : 1;
00109     mUniversalLineStyle = config.hasKey("line_style") ? stoi(config.find("line_style")) : 1;
00110     mUniversalLineWidth = config.hasKey("line_width") ? stoi(config.find("line_width")) : 1;
00111 }
00112
00118 void TGraphUser::setLegend(const CppConfigDictionary& config) {
00119     if ( config.hasKey("legend") ) {
00120         CppConfigDictionary legendConfig = config.getSubConfig("legend");
00121         mLegendTitle = legendConfig.hasKey("title") ? legendConfig.find("title") : "";
00122         Float_t xMin = legendConfig.hasKey("x_min") ? stof(legendConfig.find("x_min")) : 0.7;
00123         Float_t yMin = legendConfig.hasKey("y_min") ? stof(legendConfig.find("y_min")) : 0.7;
00124         Float_t xMax = legendConfig.hasKey("x_max") ? stof(legendConfig.find("x_max")) : 0.9;
00125         Float_t yMax = legendConfig.hasKey("y_max") ? stof(legendConfig.find("y_max")) : 0.9;
00126         mLegendPoints = {xMin, yMin, xMax, yMax};
00127     } else {
00128         mLegendTitle = "Legend";
00129         mLegendPoints = {0.7, 0.7, 0.9, 0.9};
00130     }
00131     mLegend = new TLegend(mLegendPoints[0], mLegendPoints[1], mLegendPoints[2], mLegendPoints[3]);
00132     mLegend->SetHeader(mLegendTitle, "C");
00133 }

```

8.81 /home/ychoi/ATOM/entryCalculator_copy.cpp File Reference

```

#include <iostream>
#include "TMath.h"

```

```
#include "TF1.h"
#include "TF2.h"
#include "TH1.h"
#include "TH2.h"
#include "Math/AdaptiveIntegratorMultiDim.h"
```

Functions

- void [entryCalculator_copy](#) ()

8.81.1 Function Documentation

8.81.1.1 entryCalculator_copy()

```
void entryCalculator_copy ( )
```

Definition at line 11 of file [entryCalculator_copy.cpp](#).

8.82 /home/ychoi/ATOM/entryCalculator_copy.cpp

[Go to the documentation of this file.](#)

```
00001 #include <iostream>
00002 #include "TMath.h"
00003 #include "TF1.h"
00004 #include "TF2.h"
00005 #include "TH1.h"
00006 #include "TH2.h"
00007 #include "Math/AdaptiveIntegratorMultiDim.h"
00008
00009
00010
00011 void entryCalculator_copy() {
00012     Double_t s_R; // source Active radius
00013     Double_t c_L; // collimater
00014     Double_t c_R; // collimator
00015     Double_t s_A; // source Activity
00016     Double_t R; // r
00017     Double_t pi = TMath::Pi();
00018     Int_t nbins = 100;
00019
00020     TH1D* histogram = new TH1D("histo", "histo", nbins, 0, 2.5);
00021
00022     for ( Int_t i = 0; i < nbins + 1; ++i ) {
00023         //[0] = c_L
00024         //[1] = c_R
00025         // c_R= 0~2.5mm , c_L=10mm
00026         s_R = 2.5; // mm
00027         c_L = 10.0; // mm
00028         c_R = i * 2.5 / nbins; // mm
00029         s_A = 4174; // mm
00030         if ( s_R > c_R )
00031             R = c_R;
00032         else
00033             R = s_R;
00034
00035         // cout<<R<<endl;
00036
00037
00038
00039         //x=alpha or r //y=phi
00040         TF2* func1 = new TF2("func1",
00041             "(1/(4*pi))*2*pi*x*(1-([0]/(sqrt(pow([0],2)+pow(sqrt(pow([1],2)-pow(x*sin(y),2))-x*cos(y),2)))))", 0,
00042             R, 0, 2 * pi);
00041         func1->SetParameter(0, c_L);
00042         func1->SetParameter(1, c_R);
00043     }
```

```

00044
00045 // func1->Draw("COL");
00046 // func1->Draw("Surf");
00047 // func1->Draw("same");
00048 //EPS = Epsilon
00049 //EPS default = 1.e-9
00050
00051 Double_t integral = func1->Integral(0, R, 0, 2 * pi);
00052 // cout<< integral <<endl;
00053
00054
00055 Double_t EffSolAngle = integral * 4 * pi;
00056 cout << "Radius of collimator = " << c_R << endl;
00057 cout << "Effective Solid Angle = " << EffSolAngle << endl;
00058 // Double_t integralError = func1->IntegralError(0,2.5,0,2*pi);
00059 // cout<< integralError <<endl;
00060 //??
00061
00062 Int_t binNumberFactor = nbinx / 2.5;
00063 histogram->SetBinContent(c_R * binNumberFactor, EffSolAngle);
00064
00065 }
00066
00067
00068
00069 histogram->Draw("");
00070 TF1* quadratic = new TF1("quadratic", "[0]*pow(x,2)+[1]", 0, 2.5);
00071 histogram->Fit(quadratic);
00072 histogram->Draw("SAME");
00073 // // histogram->Draw("ALP");
00074 // TF1 *exponential = new TF1("exponential", "exp([0]*x+[1]", 0, 2.5);
00075 // histogram->Fit(exponential);
00076 // histogram->Draw("SAME");
00077
00078
00079
00080
00081 // // activity
00082 // Double_t activityDensity = s_A/(pi*pow(s_R,2));
00083 // // cout << "ActivityDensity" << endl;
00084 // // cout << activityDensity << endl;
00085 // Double_t TheoreticalValue=integral*activityDensity;
00086 // cout << "Effective Activity" <<endl;
00087 // cout << TheoreticalValue << endl;
00088
00089 // //10min Entry
00090 // Double_t entry_10min = TheoreticalValue*600;
00091 // cout<< "10min Entry" <<endl;
00092 // cout<<entry_10min<<endl;
00093
00094
00095 // // (rho-6)/rho
00096 // TF2 *func2 = new
TF2 ("func2", "2*pi*x*(1-(10/(sqrt(pow(10,2)+pow(sqrt(pow(3,2)-pow(x*sin(y),2))-x*cos(y),2))))", 0, 2.5, 0, 2*pi);
00097 // cout<<func2->Integral(0,2.5,0,2*pi)<<endl;
00098 // func2->Draw("Surf");
00099
00100 }

```

8.83 /home/ychoi/ATOM/exe/alpide_dac.cpp File Reference

```

#include <iostream>
#include <fstream>
#include <filesystem>
#include "TFile.h"
#include "TCanvas.h"
#include "TGraph.h"
#include "TLegend.h"
#include "TMultiGraph.h"
#include "TStyle.h"
#include "TMarker.h"
#include "cppargs.h"

```

Classes

- class [DACtoADC](#)

Functions

- [ArgumentParser](#) [add_parser](#) (int argc, char **argv)
- void [file_open](#) (std::ifstream &inputfile, [ArgumentParser](#) parser)
- void [input_data](#) (std::vector< [DACtoADC](#) > &data, std::ifstream &inputfile)
- void [get_graph](#) (std::vector< TGraph > &graphs, const std::vector< [DACtoADC](#) > &datas)
- void [get_multi_graph](#) (TMultiGraph &mg, std::vector< TGraph > &graphs)
- void [set_legend](#) (TLegend &legend, std::vector< TGraph > &graphs)
- void [make_root_file](#) (TMultiGraph &mg, std::vector< TGraph > &graphs, TLegend &legend)
- int [main](#) (int argc, char **argv)

8.83.1 Function Documentation

8.83.1.1 [add_parser\(\)](#)

```
ArgumentParser add_parser (  
    int argc,  
    char ** argv )
```

Definition at line 47 of file [alpide_dac.cpp](#).

8.83.1.2 [file_open\(\)](#)

```
void file_open (  
    std::ifstream & inputfile,  
    ArgumentParser parser )
```

Definition at line 56 of file [alpide_dac.cpp](#).

8.83.1.3 [get_graph\(\)](#)

```
void get_graph (  
    std::vector< TGraph > & graphs,  
    const std::vector< DACtoADC > & datas )
```

Definition at line 94 of file [alpide_dac.cpp](#).

8.83.1.4 [get_multi_graph\(\)](#)

```
void get_multi_graph (  
    TMultiGraph & mg,  
    std::vector< TGraph > & graphs )
```

Definition at line 115 of file [alpide_dac.cpp](#).

8.83.1.5 input_data()

```
void input_data (
    std::vector< DACToADC > & data,
    std::ifstream & inputfile )
```

Definition at line 64 of file [alpide_dac.cpp](#).

8.83.1.6 main()

```
int main (
    int argc,
    char ** argv )
```

Definition at line 149 of file [alpide_dac.cpp](#).

8.83.1.7 make_root_file()

```
void make_root_file (
    TMultiGraph & mg,
    std::vector< TGraph > & graphs,
    TLegend & legend )
```

Definition at line 131 of file [alpide_dac.cpp](#).

8.83.1.8 set_legend()

```
void set_legend (
    TLegend & legend,
    std::vector< TGraph > & graphs )
```

Definition at line 121 of file [alpide_dac.cpp](#).

8.84 alpide_dac.cpp

[Go to the documentation of this file.](#)

```
00001 #include <iostream>
00002 #include <fstream>
00003 #include <filesystem>
00004
00005 #include "TFile.h"
00006 #include "TCanvas.h"
00007 #include "TGraph.h"
00008 #include "TLegend.h"
00009 #include "TMultiGraph.h"
00010 #include "TStyle.h"
00011 #include "TMarker.h"
00012
00013 #include "cppargs.h"
00014
00015 class DACToADC {
00016 private:
00017     std::string kind_;
00018     std::vector<Int_t> dac_;
00019     std::vector<Int_t> adc_;
00020 public:
00021     DACToADC(std::string kind) : kind_(kind) {}
```



```

00022     void setDAC(int dac) {
00023         dac_.push_back(dac);
00024     }
00025     void setADC(int adc) {
00026         adc_.push_back(adc);
00027     }
00028     void setDAC(std::vector<int> dac) {
00029         dac_.reserve(dac_.size() + dac.size());
00030         dac_.insert(dac_.end(), dac.begin(), dac.end());
00031     }
00032     void setADC(std::vector<int> adc) {
00033         adc_.reserve(adc_.size() + adc.size());
00034         adc_.insert(adc_.end(), adc.begin(), adc.end());
00035     }
00036     std::string getKind() {
00037         return kind_;
00038     }
00039     std::vector<Int_t> getDAC() {
00040         return dac_;
00041     }
00042     std::vector<Int_t> getADC() {
00043         return adc_;
00044     }
00045 };
00046
00047 ArgumentParser add_parser(int argc, char** argv) {
00048     ArgumentParser parser = ArgumentParser(argc,argv).setDescription("Show DAC scan results");
00049     parser.add_argument("filename").help("Name of the file to analysis").add_finish();
00050     parser.add_argument("--path").help("Output plots path").set_default(".").add_finish();
00051     parser.add_argument("--no-fit").help("Do fit?").set_default("true").add_finish();
00052     parser.parse_args();
00053     return parser;
00054 }
00055
00056 void file_open(std::ifstream& inputfile, ArgumentParser parser) {
00057     std::filesystem::path inputfilepath(parser.get_value<std::string>("filename"));
00058     if (!std::filesystem::exists(inputfilepath)) {
00059         std::cerr << "File not found: " << inputfilepath << std::endl;
00060     }
00061     inputfile = std::ifstream(inputfilepath, std::ios::binary);
00062 }
00063
00064 void input_data(std::vector<DACtoADC>& data, std::ifstream& inputfile) {
00065     std::string line;
00066
00067     std::vector<int> dacs;
00068     std::vector<int> adcs;
00069     std::string old = "";
00070     std::string kind;
00071     std::string dac;
00072     std::string adc;
00073
00074     while (getline(inputfile,line)) {
00075         std::istringstream iss(line);
00076         std::getline(iss, kind, '\t');
00077         if (old != kind) {
00078             DACtoADC temp(old);
00079             temp.setDAC(dacs);
00080             temp.setADC(adcs);
00081             data.push_back(temp);
00082             old = kind;
00083             dacs.clear();
00084             adcs.clear();
00085         }
00086         std::getline(iss, dac, '\t');
00087         dacs.push_back(stoi(dac));
00088         std::getline(iss, adc, '\t');
00089         adcs.push_back(stoi(adc));
00090     }
00091     data.erase(data.begin());
00092 }
00093
00094 void get_graph(std::vector<TGraph>& graphs, const std::vector<DACtoADC>& datas) {
00095     for (DACtoADC data : datas) {
00096         graphs.push_back(TGraph((data.getADC().size()),data.getDAC().data(),data.getADC().data()));
00097         graphs.back().SetName((TString) data.getKind());
00098         graphs.back().SetTitle("Scan of " + (TString) data.getKind() + " DAC; DAC Setting; ADC");
00099     }
00100     int i = 0;
00101     for (TGraph& graph : graphs) {
00102         if (i < 9) {
00103             graph.SetMarkerStyle(20);
00104             graph.SetMarkerSize(.5);
00105             graph.SetMarkerColor(i+1);
00106         } else {
00107             graph.SetMarkerStyle(22);
00108             graph.SetMarkerSize(.5);

```

```

00109         graph.SetMarkerColor(i-7);
00110     }
00111     i++;
00112 }
00113 }
00114
00115 void get_multi_graph(TMGraph& mg, std::vector<TGraph>& graphs) {
00116     for (TGraph& graph : graphs) {
00117         mg.Add(&graph);
00118     }
00119 }
00120
00121 void set_legend(TLegend& legend, std::vector<TGraph>& graphs){
00122     legend.SetNColumns(2);
00123
00124     TMarker mark;
00125     for (TGraph graph : graphs) {
00126         graph.SetMarkerSize(1.5);
00127         legend.AddEntry(graph.Clone(),graph.GetName(),"p");
00128     }
00129 }
00130
00131 void make_root_file(TMGraph& mg, std::vector<TGraph>& graphs, TLegend& legend) {
00132     TFile rootFile("data/value.root","RECREATE");
00133     rootFile.mkdir("DAC");
00134     rootFile.cd("DAC");
00135     mg.SetName("Total");
00136     mg.SetTitle("Summary plot; DAC Setting; ADC");
00137     TCanvas mcan("Total","Merge Canvas", 1200,1000);
00138     mg.Draw("AP");
00139     legend.Draw();
00140     mcan.SetLeftMargin(0.12);
00141     mcan.Write();
00142     mg.Write();
00143     for (TGraph& graph : graphs) {
00144         graph.Write();
00145     }
00146     rootFile.Close();
00147 }
00148
00149 int main(int argc, char** argv) {
00150     ArgumentParser parser = add_parser(argc,argv);
00151
00152     std::ifstream inputfile;
00153     file_open(inputfile, parser);
00154
00155     std::vector<DACtoADC> data;
00156     input_data(data, inputfile);
00157
00158     std::vector<TGraph> graphs;
00159     get_graph(graphs,data);
00160
00161     inputfile.close();
00162
00163     TLegend legend(0.12,0.65,0.45,0.9);
00164     set_legend(legend,graphs);
00165
00166     TMGraph mg;
00167     get_multi_graph(mg,graphs);
00168
00169     make_root_file(mg, graphs, legend);
00170     return 0;
00171 }

```

8.85 /home/ychoi/ATOM/exe/alpide_hitmap.cpp File Reference

```

#include <iostream>
#include <filesystem>
#include <fstream>
#include <numeric>
#include <cmath>
#include "TCanvas.h"
#include "TH2.h"
#include "TStyle.h"
#include "TText.h"
#include "TPaveText.h"

```

```
#include "cppargs.h"
#include "alpide_decoder.h"
#include "pitch_clock.h"
```

Functions

- [ArgumentParser set_parse](#) (int argc, char **argv)
- `std::ifstream` [open_file](#) ([ArgumentParser](#) parser)
- `std::vector< uint8_t >` [read_file](#) (`std::ifstream` &outfile, int &nev)
- `std::vector< ALPIDE::TEvent >` [decode_event](#) (`std::vector< uint8_t >` &values)
- `TCanvas *` [draw_hitmap](#) (`std::vector< ALPIDE::TEvent >` &events)
- void [save_canvas](#) ([ArgumentParser](#) &parser, `TCanvas *`&c1)
- `TCanvas *` [draw_histogram](#) (`std::vector< ALPIDE::TEvent >` &events)
- int [main](#) (int argc, char **argv)

8.85.1 Function Documentation

8.85.1.1 `decode_event()`

```
std::vector< ALPIDE::TEvent > decode_event (
    std::vector< uint8_t > & values )
```

Definition at line 51 of file [alpide_hitmap.cpp](#).

8.85.1.2 `draw_histogram()`

```
TCanvas * draw_histogram (
    std::vector< ALPIDE::TEvent > & events )
```

Definition at line 132 of file [alpide_hitmap.cpp](#).

8.85.1.3 `draw_hitmap()`

```
TCanvas * draw_hitmap (
    std::vector< ALPIDE::TEvent > & events )
```

Definition at line 61 of file [alpide_hitmap.cpp](#).

8.85.1.4 `main()`

```
int main (
    int argc,
    char ** argv )
```

Definition at line 155 of file [alpide_hitmap.cpp](#).

8.85.1.5 open_file()

```
std::ifstream open_file (
    ArgumentParser parser )
```

Definition at line 28 of file [alpide_hitmap.cpp](#).

8.85.1.6 read_file()

```
std::vector< uint8_t > read_file (
    std::ifstream & outfile,
    int & nev )
```

Definition at line 39 of file [alpide_hitmap.cpp](#).

8.85.1.7 save_canvas()

```
void save_canvas (
    ArgumentParser & parser,
    TCanvas *& c1 )
```

Definition at line 123 of file [alpide_hitmap.cpp](#).

8.85.1.8 set_parse()

```
ArgumentParser set_parse (
    int argc,
    char ** argv )
```

Definition at line 16 of file [alpide_hitmap.cpp](#).

8.86 alpide_hitmap.cpp

[Go to the documentation of this file.](#)

```
00001 #include <iostream>
00002 #include <filesystem>
00003 #include <fstream>
00004 #include <numeric>
00005 #include <cmath>
00006
00007 #include "TCanvas.h"
00008 #include "TH2.h"
00009 #include "TStyle.h"
00010 #include "TText.h"
00011 #include "TPaveText.h"
00012 #include "cppargs.h"
00013 #include "alpide_decoder.h"
00014 #include "pitch_clock.h"
00015
00016 ArgumentParser set_parse(int argc, char** argv) {
00017     ArgumentParser parser = ArgumentParser(argc,argv).setDescription("Draw hitmap and calculate
00018     hitrate of an ALPIDE");
00019     parser.add_argument("rawdata").metavar("FILENAME").help("raw data file to be
00020     processed").add_finish();
00021     parser.add_argument("--bins").add_minor_argument("-b").type("int").add_domain({"1", "2", "4", "8",
00022     "16", "32"}).help("bin size").set_default("1").add_finish();
00023     parser.add_argument("--max").add_minor_argument("-M").type("int").help("color scale
00024     limit").add_finish();
```

```

00021     parser.add_argument("--path").help("Output plots path").set_default(".").add_finish();
00022     parser.add_argument("--dump-raw-hits").help("Dump hit pixel addresses for each event to
file").set_default("false").add_finish();
00023     parser.add_argument("--dump-acc-hits").help("Dump hit pixel addresses sorted by frequency to
file").set_default("false").add_finish();
00024     parser.parse_args();
00025     return parser;
00026 }
00027
00028 std::ifstream open_file(ArgumentParser parser) {
00029     std::filesystem::path outfilepath(parser.get_value<std::string>("rawdata"));
00030     std::string outfilename = outfilepath.stem();
00031
00032     if (!std::filesystem::exists(outfilepath)) {
00033         std::cerr << "File not found: " << outfilepath << std::endl;
00034     }
00035     std::ifstream outfile(outfilepath, std::ios::binary);
00036     return outfile;
00037 }
00038
00039 std::vector<uint8_t> read_file(std::ifstream& outfile, int& nev) {
00040     char buf[sizeof(uint8_t)];
00041     std::vector<uint8_t> values;
00042
00043     while (outfile.read(buf, sizeof(buf))) {
00044         values.push_back(0);
00045         memcpy(&values[nev], buf, sizeof(values[nev]));
00046         nev++;
00047     }
00048     return values;
00049 }
00050
00051 std::vector<ALPIDE::TEvent> decode_event(std::vector<uint8_t>& values) {
00052     int i = 0;
00053     std::vector<ALPIDE::TEvent> events;
00054     ALPIDE::TDecoder* decoder = new ALPIDE::TDecoder();
00055     while (i < values.size()) {
00056         events.push_back(decoder->alpide_decode_event(values, i));
00057     }
00058     return events;
00059 }
00060
00061 TCanvas* draw_hitmap(std::vector<ALPIDE::TEvent>& events) {
00062     TCanvas* c1 = new TCanvas("c1", "c1", 2000, 775);
00063     TH2I* hm = new TH2I("hm", "Hitmap; Column; Row", 1024, 0, 1024, 512, 0, 512);
00064     for (ALPIDE::TEvent event : events) {
00065         for (std::array<int, 2> coord : event.getCoords()) {
00066             hm->Fill(coord[0], coord[1]);
00067         }
00068     }
00069
00070     hm->Draw("colz");
00071     hm->GetXaxis()->SetNdivisions(4, 4, 4, kFALSE);
00072     hm->GetXaxis()->SetTickLength(-0.035);
00073     hm->GetXaxis()->SetLabelOffset(0.03);
00074     hm->GetXaxis()->CenterTitle();
00075     hm->GetXaxis()->SetTitleOffset(1.4);
00076
00077     hm->GetYaxis()->SetNdivisions(2, 4, 4, kFALSE);
00078     hm->GetYaxis()->SetTickLength(-0.018);
00079     hm->GetYaxis()->SetLabelOffset(0.017);
00080     hm->GetYaxis()->CenterTitle();
00081     hm->GetYaxis()->SetTitleOffset(0.9);
00082
00083     hm->GetZaxis()->SetMaxDigits(2);
00084     hm->GetZaxis()->SetTitle("dE/d[Row]d[Column]");
00085     hm->GetZaxis()->SetTitleOffset(0.6);
00086
00087     gStyle->SetTitleX(0.38);
00088     gStyle->SetTitleY(0.98);
00089     c1->SetTopMargin(0.085);
00090     c1->SetBottomMargin(0.115);
00091     c1->SetLeftMargin(0.065);
00092     c1->SetRightMargin(0.315);
00093
00094     std::vector<int> hitrate={0};
00095     for (int i = 0; i < 1024; i++) {
00096         for (int j = 0; j < 1024; j++) {
00097             hitrate.push_back(hm->GetBinContent(i, j));
00098         }
00099     }
00100
00101     sort(hitrate.rbegin(), hitrate.rend());
00102     TPaveText *pt = new TPaveText(1170, 0, 1515, 256);
00103     double total = (events.size()*1024.*512.);
00104     pt->AddText("Hit rate      0 masked:");
00105     TString str = Form("      %.2e per pixel per event", accumulate(hitrate.begin(), hitrate.end(),

```

```

00106         0)/total);
00107         pt->AddText(str);
00108         pt->AddText("Hit rate 10 masked:");
00109         str = Form(" %.2e per pixel per event",accumulate(hirate.begin()+10, hirate.end(), 0)/total);
00110         pt->AddText(str);
00111         pt->AddText("Hit rate 100 masked:");
00112         str = Form(" %.2e per pixel per event",accumulate(hirate.begin()+100, hirate.end(),
00113         0)/total);
00114         pt->AddText(str);
00115         pt->AddText("Hit rate 1000 masked:");
00116         str = Form(" %.2e per pixel per event",accumulate(hirate.begin()+1000, hirate.end(),
00117         0)/total);
00118         pt->AddText(str);
00119         c1->Update();
00120         pt->SetTextAlign(12);
00121         pt->Draw("SAME");
00122         return c1;
00123 }
00124 void save_canvas(ArgumentParser& parser, TCanvas*& c1) {
00125     std::filesystem::path outfilepath(parser.get_value<std::string>("rawdata"));
00126     std::string outfilename = outfilepath.stem();
00127     std::filesystem::path outputfile = parser.get_value<std::string>("path");
00128     outputfile.append(outfilename+".pdf");
00129     std::cout << outputfile << std::endl;
00130     c1->SaveAs((TString) outputfile);
00131 }
00132 TCanvas* draw_histogram(std::vector<ALPIDE::TEvent>& events) {
00133     gStyle->SetTitleX(0.5);
00134     gStyle->SetTitleY(1);
00135     TCanvas* c2 = new TCanvas("c2","c2",2000,1000);
00136     c2->Divide(2,1);
00137     c2->cd(1);
00138     TH1I* eventHist = new TH1I("h1","eventHist",150,0,events.back().getEventNum()+1);
00139     std::cout << (long int) events[0].getTime() << std::endl;
00140     TH1I* timeHist = new TH1I("h2","timeHist",150,events[0].getTime(),events.back().getTime());
00141     for (ALPIDE::TEvent event : events) {
00142         for (std::array<int,2> coord : event.getCoords()) {
00143             eventHist->Fill(event.getEventNum());
00144             timeHist->Fill(event.getTime());
00145         }
00146     }
00147     eventHist->Draw();
00148     c2->cd(2);
00149     timeHist->Draw();
00150
00151     c2->SaveAs("data/histogram.png");
00152
00153     return c2;
00154 }
00155 int main(int argc, char** argv) {
00156     TClock* clock = new TClock();
00157     std::cout << "ALPIDE Decoding..." << std::endl;
00158
00159     ArgumentParser parser = set_parse(argc,argv);
00160
00161     std::ifstream outfile = open_file(parser);
00162
00163     int nev = 0;
00164     std::vector<uint8_t> values = read_file(outfile,nev);
00165
00166     std::vector<ALPIDE::TEvent> events = decode_event(values);
00167
00168     TCanvas* c1 = draw_hitmap(events);
00169
00170     save_canvas(parser, c1);
00171
00172     draw_histogram(events);
00173
00174     clock->EndProgram();
00175     return 0;
00176 }

```

8.87 /home/ychoi/ATOM/exe/CompareExperimentData.cpp File Reference

```

#include <iostream>
#include <vector>
#include <string>

```

```
#include <filesystem>
#include "cppargs.h"
#include "CppConfigFile.h"
#include "TGraphCompare.h"
```

Classes

- class [ControlExperimentComparison](#)

Functions

- int [main](#) (int argc, char **argv)

8.87.1 Function Documentation

8.87.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

Definition at line 65 of file [CompareExperimentData.cpp](#).

8.88 CompareExperimentData.cpp

[Go to the documentation of this file.](#)

```
00001 #include <iostream>
00002 #include <vector>
00003 #include <string>
00004 #include <filesystem>
00005
00006 #include "cppargs.h"
00007 #include "CppConfigFile.h"
00008
00009 #include "TGraphCompare.h"
00010
00011 class ControlExperimentComparison {
00012 private:
00013     ArgumentParser mParser;
00014     CppConfigFile* mConfig;
00015     std::vector<std::string> mFileSet;
00016     std::vector<std::string> mTypeNameSet;
00017     TGraphCompare* mCompare;
00018 public:
00019     ControlExperimentComparison(int argc, char** argv);
00020     void setConfig();
00021     void initComparison();
00022 };
00023
00024 ControlExperimentComparison::ControlExperimentComparison(int argc, char** argv) : mParser(argc, argv)
00025 {
00026     mParser.setDescription("Draw plots for analysis data");
00027     mParser.add_argument("config").help("Config file").set_default("default").add_finish();
00028     mParser.parse_args();
00029
00030 void ControlExperimentComparison::setConfig() {
00031     std::string configPath = mParser.get_value<std::string>("config");
00032
00033     mConfig = new CppConfigFile(configPath);
00034     mTypeNameSet = mConfig->getConfig("File").getSubConfig("type_name").getValueList();
00035 }
```

```

00036
00037 void ControlExperimentComparison::initComparison() {
00038     std::filesystem::path inputPath = mConfig->getConfig("FileList").find("input_path");
00039     std::vector<std::string> inputNameSet =
00040         mConfig->getConfig("FileList").getSubConfig("input_file_name").getValueList();
00041     for ( std::string_view inputName : inputNameSet ) {
00042         mFileSet.push_back(std::string((inputPath / inputName).replace_extension(".root")));
00043     }
00044     mCompare = new TGraphCompare(mFileSet);
00045     // mCompare->setObject(mConfig->getConfig("DrawingObject"));
00046     for ( const std::string& typeName : mTypeNameSet ) {
00047         CppConfigDictionary typeConfig = mConfig->getConfig(typeName);
00048         if ( typeConfig.hasKey("clustersize") ) {
00049             CppConfigDictionary shapeConfig = typeConfig.getSubConfig("clustersize");
00050             shapeConfig += mConfig->getConfig("SharedProperty");
00051             mCompare->TCompareClusterSize(typeName, shapeConfig);
00052         }
00053         // if ( mConfig->hasConfig("clustersize_ratio") ) {
00054         //     CppConfigDictionary shapeConfig = mConfig->getConfig("clustersize_ratio");
00055         //     shapeConfig += mConfig->getConfig("SharedProperty");
00056         //     mCompare->TCompareClusterSizeRatio(typeName, shapeConfig);
00057         // }
00058         // if ( mConfig->hasConfig("clustersize_3D") ) {
00059         //     CppConfigDictionary shapeConfig = mConfig->getConfig("clustersize_3D");
00060         //     shapeConfig += mConfig->getConfig("SharedProperty");
00061         //     mCompare->TCompareClusterSize3D(typeName, shapeConfig);
00062         // }
00063     }
00064
00065 int main(int argc, char** argv) {
00066     ControlExperimentComparison controller(argc, argv);
00067     controller.setConfig();
00068     controller.initComparison();
00069 }

```

8.89 /home/ychoi/ATOM/exe/entrySimulation.cpp File Reference

```

#include "TEntrySimulation.h"
#include "TGraph.h"
#include "TCanvas.h"
#include "TMath.h"
#include "TF1.h"
#include "TColor.h"
#include "TLegend.h"
#include "TMultiGraph.h"
#include "cppargs.h"
#include "CppConfigFile.h"
#include "TGraphErrors.h"
#include "TGraphUser.h"

```

Functions

- [ArgumentParser set_parse](#) (int argc, char **argv)
- int [main](#) (int argc, char **argv)

8.89.1 Function Documentation

8.89.1.1 main()

```

int main (
    int argc,
    char ** argv )

```

Definition at line 22 of file [entrySimulation.cpp](#).

8.89.1.2 set_parse()

```
ArgumentParser set_parse (
    int argc,
    char ** argv )
```

Definition at line 15 of file [entrySimulation.cpp](#).

8.90 entrySimulation.cpp

[Go to the documentation of this file.](#)

```
00001 #include "TEntrySimulation.h"
00002
00003 #include "TGraph.h"
00004 #include "TCanvas.h"
00005 #include "TMath.h"
00006 #include "TFl.h"
00007 #include "TColor.h"
00008 #include "TLegend.h"
00009 #include "TMultiGraph.h"
00010 #include "cppargs.h"
00011 #include "CppConfigFile.h"
00012 #include "TGraphErrors.h"
00013 #include "TGraphUser.h"
00014
00015 ArgumentParser set_parse(int argc, char** argv) {
00016     ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Simulation for entry
research");
00017     parser.add_argument("config").help("Config file for controlling
simulation").set_default("").add_finish();
00018     parser.parse_args();
00019     return parser;
00020 }
00021
00022 int main(int argc, char** argv) {
00023     ArgumentParser parser = set_parse(argc, argv);
00024     std::string configPath = parser.get_value<std::string>("config");
00025     CppConfigFile config(configPath);
00026
00027     TEntrySimulation simulation;
00028     // Set config dictionary
00029     CppConfigDictionary fileConfig = config.getConfig("File");
00030     CppConfigDictionary environmentConfig = config.getConfig("Environment");
00031     CppConfigDictionary graphConfig = config.getConfig("Graph");
00032
00033     // Set source information
00034     double sourceRadius = 2.5;
00035     if ( environmentConfig.hasKey("source_radius") ) {
00036         sourceRadius = stod(environmentConfig.find("source_radius"));
00037     }
00038     simulation.setSource(sourceRadius);
00039
00040     // Set ALPIDE information
00041     double detectorX = 30.;
00042     double detectorY = 15.;
00043     double S2CDistance = .5;
00044     double C2DDistance = 2.;
00045     double collimatorLength = 1.;
00046     double detectorCoordX = 0.;
00047     double detectorCoordY = 0.;
00048     if ( environmentConfig.hasKey("detector_x") ) {
00049         detectorX = stod(environmentConfig.find("detector_x"));
00050     }
00051     if ( environmentConfig.hasKey("detector_y") ) {
00052         detectorY = stod(environmentConfig.find("detector_y"));
00053     }
00054     if ( environmentConfig.hasKey("source_to_collimator") ) {
00055         S2CDistance = stod(environmentConfig.find("source_to_collimator"));
00056     }
00057     if ( environmentConfig.hasKey("collimator_to_detector") ) {
00058         C2DDistance = stod(environmentConfig.find("collimator_to_detector"));
00059     }
00060     if ( environmentConfig.hasKey("collimator_length") ) {
00061         collimatorLength = stod(environmentConfig.find("collimator_length"));
00062     }
00063     if ( environmentConfig.hasKey("detector_coord_x") ) {
00064         detectorCoordX = stod(environmentConfig.find("detector_coord_x"));
```

```

00065     }
00066     if ( environmentConfig.hasKey("detector_coord_y") ) {
00067         detectorCoordX = stod(environmentConfig.find("detector_coord_y"));
00068     }
00069     simulation.setDetector(detectorX, detectorY, detectorCoordX, detectorCoordY, -(S2CDistance +
C2DDistance + collimatorLength));
00070
00071     // Set Collimator
00072     double collimatorRadius = 1.5;
00073     if ( environmentConfig.hasKey("collimator_radius") ) {
00074         collimatorRadius = stod(environmentConfig.find("collimator_radius"));
00075     }
00076     simulation.setCollimator(collimatorRadius, -S2CDistance, -(S2CDistance + collimatorLength));
00077
00078     // TGraph* graph = new TGraph();
00079     std::vector<std::pair<std::string, std::vector<std::array<double, 2>>> ratio;
00080     double divideCriteria = 0.;
00081     if ( graphConfig.find("type") == "comprehensive_point" ) {
00082         int nGraph = graphConfig.getSubConfig("graphs").getSubConfigSet().size();
00083         for ( int iGraph = 0; iGraph < nGraph; iGraph++ ) {
00084             std::vector<std::array<double, 2>> tempRatio;
00085             double length =
00086             stod(graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].find("length"));
00087             std::string widthStr =
graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].find("width");
00088             double min = stod(widthStr.substr(0, widthStr.find(':')));
00089             widthStr = widthStr.substr(widthStr.find(':') + 1);
00090             double max = stod(widthStr.substr(0, widthStr.find(':')));
00091             widthStr = widthStr.substr(widthStr.find(':') + 1);
00092             double gap = stod(widthStr.substr(0, widthStr.find(':')));
00093             for ( double width = min; width < max; width += gap ) {
00094                 simulation.setDetector(detectorX, detectorY, detectorCoordX, detectorCoordY,
-(S2CDistance + C2DDistance + length));
00095                 simulation.setCollimator(sqrt(width / TMath::Pi()), -S2CDistance, -(S2CDistance +
length));
00096                 double entryRatio = simulation.doCount();
00097                 tempRatio.push_back({width, (entryRatio / 2)});
00098                 if ( width < 35.1 && width > 34.9 && length < 1.1 && length > 0.9 ) {
00099                     divideCriteria = (entryRatio / 2);
00100                 }
00101                 std::cout << length << "\t" << width << "\t" << entryRatio / 2 << std::endl;
00102             }
00103             ratio.push_back({std::string(graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].getConfigName()),
tempRatio});
00104         }
00105         std::vector<TGraph*> entryRatioGraph;
00106
00107         TGraphUser* graph1 = new TGraphUser(graphConfig.getSubConfig("entryRatio"));
00108         for ( const std::pair<std::string, std::vector<std::array<double, 2>>>&ratioSet : ratio ) {
00109             entryRatioGraph.push_back(new TGraph());
00110             for ( const std::array<double, 2>&point : ratioSet.second ) {
00111                 entryRatioGraph.back()->AddPoint(point[0], point[1] / divideCriteria);
00112             }
00113             int nGraph = graphConfig.getSubConfig("graphs").getSubConfigSet().size();
00114             for ( int iGraph = 0; iGraph < nGraph; iGraph++ ) {
00115                 if ( graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].getConfigName() ==
ratioSet.first ) {
00116                     graph1->AddGraph(entryRatioGraph.back(),
graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph]);
00117                 }
00118             }
00119         }
00120
00121         for ( const CppConfigDictionary& expData : graphConfig.getSubConfig("exp_data").getSubConfigSet()
) {
00122             TGraphErrors* eGraph = new TGraphErrors();
00123             int i = 0;
00124             for ( const CppConfigDictionary& point : expData.getSubConfig("point").getSubConfigSet() ) {
00125                 Double_t width = stod(point.find("width"));
00126                 Double_t width_error = stod(point.find("width_error"));
00127                 Double_t value = stod(point.find("value"));
00128                 Double_t error = stod(point.find("error"));
00129                 eGraph->SetPoint(i, width, value);
00130                 eGraph->SetPointError(i, width_error, error);
00131                 i++;
00132             }
00133             graph1->AddGraph(eGraph, expData);
00134         }
00135
00136         // Set output path and create directories if it isn't.
00137         std::filesystem::path outputPath = fileConfig.find("output_path");
00138         std::filesystem::create_directories(outputPath);
00139
00140         graph1->Save(outputPath);

```

```

00142
00143     return 0;
00144 }

```

8.91 /home/ychoi/ATOM/exe/ExperimentAnalysis.cpp File Reference

```
#include "ExperimentAnalysis.hpp"
```

Functions

- `int main (int argc, char **argv)`

8.91.1 Function Documentation

8.91.1.1 main()

```

int main (
    int argc,
    char ** argv )

```

Definition at line 3 of file [ExperimentAnalysis.cpp](#).

8.92 ExperimentAnalysis.cpp

[Go to the documentation of this file.](#)

```

00001 #include "ExperimentAnalysis.hpp"
00002
00003 int main(int argc, char** argv) {
00004     ControlExperimentAnalysis controller(argc, argv);
00005     controller.setConfig();
00006     controller.openInputFile();
00007     controller.setExpDataSet();
00008     controller.doBasicAnalysis();
00009     controller.drawHitmap();
00010     controller.clusterization();
00011     controller.drawClustermapAndClustersize();
00012     // controller.doDivideBySize();
00013     // controller.drawClusterShapeInfos();
00014
00015     // clusterAnalyser.saveHitmapByClustersize(*config->getConfig("Clustermap"));
00016
00017     return 0;
00018 }

```

8.93 /home/ychoi/ATOM/exe/ExperimentAnalysis.hpp File Reference

```

#include <iostream>
#include <fstream>
#include <filesystem>
#include "CppConfigFile.h"
#include "TAnalyser.h"
#include "TExperimentData.h"
#include "cppargs.h"
#include "TClusterDivideData.h"
#include "TClusterization.h"
#include "TCanvas.h"
#include "TFile.h"

```

Classes

- class [ControlExperimentAnalysis](#)

8.94 ExperimentAnalysis.hpp

[Go to the documentation of this file.](#)

```

00001 #include <iostream>
00002 #include <fstream>
00003 #include <filesystem>
00004
00005 // #include "TEvent.h"
00006 #include "CppConfigFile.h"
00007 #include "TAnalyser.h"
00008 #include "TExperimentData.h"
00009 #include "cppargs.h"
00010 #include "TClusterDivideData.h"
00011 #include "TClusterization.h"
00012
00013 #include "TCanvas.h"
00014 #include "TFile.h"
00015
00016 class ControlExperimentAnalysis {
00017     ArgumentParser mParser;
00018     CppConfigFile* mConfig;
00019     std::vector<std::string> mTypeNameSet;
00020     std::unordered_map<std::string, CppConfigDictionary*> mSubConfigSet;
00021     std::unordered_map<std::string, TExperimentData*> mExpDataSet;
00022
00023     std::string mInputFilePath;
00024     TFile* mInputFile = nullptr;
00025     TAnalyser* mAnalyser;
00026     std::vector<int> mClusterRange;
00027
00028 public:
00029     ControlExperimentAnalysis(int argc, char** argv);
00030     ~ControlExperimentAnalysis();
00031     void setConfig();
00032     void openInputFile();
00033     void setExpDataSet();
00034     void doBasicAnalysis();
00035     void doMasking();
00036     void drawHitmap();
00037     void clusterization();
00038     void drawClustermapAndClustersize();
00039     void doDivideBySize();
00040     void drawClusterShapeInfos();
00041
00042     std::vector<int> getClusterSizeRange(const CppConfigDictionary privateProperty);
00043 };
00044
00045 ControlExperimentAnalysis::ControlExperimentAnalysis(int argc, char** argv) : mParser(argc, argv) {
00046     mParser.setDescription("Draw plots for analysis data");
00047     mParser.add_argument("config").help("Config file").set_default("default").add_finish();
00048     mParser.parse_args();
00049 }
00050
00051 ControlExperimentAnalysis::~ControlExperimentAnalysis() {
00052     for ( auto& key : mSubConfigSet ) {
00053         delete key.second;
00054         key.second = nullptr;
00055     }
00056     for ( auto& key : mExpDataSet ) {
00057         delete key.second;
00058         key.second = nullptr;
00059     }
00060     if ( mInputFile != nullptr && !mInputFile->IsDestructed() ) {
00061         mInputFile->Close();
00062         delete mInputFile;
00063         mInputFile = nullptr;
00064     }
00065     if ( mAnalyser != nullptr && !mAnalyser->IsDestructed() ) {
00066         delete mAnalyser;
00067         mAnalyser = nullptr;
00068     }
00069 }
00070
00071 void ControlExperimentAnalysis::setConfig() {
00072     std::string configPath = mParser.get_value<std::string>("config");
00073     mConfig = new CppConfigFile(configPath);

```

```

00074
00075     mTypeNameSet = mConfig->getConfig("File").getSubConfig("type_name").getValueList();
00076     mInputFilePath = mConfig->getConfig("File").find("input_file");
00077 }
00078
00079 void ControlExperimentAnalysis::openInputFile() {
00080     mInputFile = new TFile(static_cast<TString>(mInputFilePath), "READ");
00081 }
00082
00083 void ControlExperimentAnalysis::setExpDataSet() {
00084     for ( const std::string& typeName : mTypeNameSet ) {
00085         mExpDataSet.insert_or_assign(typeName, new TExperimentData());
00086     }
00087 }
00088
00089 void ControlExperimentAnalysis::doBasicAnalysis() {
00090     mAnalyser = new TAnalyser(mInputFile, mExpDataSet);
00091     mConfig->getConfig("Masking").hasKey("time_stamp_cut");
00092     mAnalyser->storeEvents(mConfig->getConfig("SharedProperty"));
00093     mAnalyser->setExpSettingLegend(mConfig->getConfig("ExperimentSetting"));
00094     if ( mConfig->getConfig("File").hasKey("output_graph") ) {
00095         mAnalyser->openOutputGraphFile(mConfig->getConfig("File").find("output_graph"));
00096         for ( std::string_view typeName : mTypeNameSet ) {
00097             mAnalyser->openDirectory(typeName);
00098         }
00099     }
00100 }
00101
00102 void ControlExperimentAnalysis::doMasking() {
00103     mAnalyser->doMasking(stoi(mConfig->getConfig("Masking").find("cut")));
00104     for ( std::string_view typeName : mTypeNameSet ) {
00105         if ( typeName != "Basic" ) {
00106             mExpDataSet.insert_or_assign(std::string(typeName), mAnalyser->getAnEventSet(typeName));
00107         }
00108     }
00109 }
00110
00111 void ControlExperimentAnalysis::drawHitmap() {
00112     if ( mConfig->getConfig("Basic").hasKey("hitmap") ) {
00113         CppConfigDictionary hitmapConfig = mConfig->getConfig("Basic").getSubConfig("hitmap");
00114         hitmapConfig += mConfig->getConfig("SharedProperty");
00115         mAnalyser->saveHitmap("Basic", hitmapConfig);
00116     }
00117     doMasking();
00118     for ( const std::string& typeName : mTypeNameSet ) {
00119         if ( typeName != "Basic" ) {
00120             if ( mConfig->getConfig(typeName).hasKey("hitmap") ) {
00121                 CppConfigDictionary hitmapConfig =
00122                 mConfig->getConfig(typeName).getSubConfig("hitmap");
00123                 hitmapConfig += mConfig->getConfig("SharedProperty");
00124                 mAnalyser->saveHitmap(typeName, hitmapConfig);
00125             }
00126         }
00127     }
00128 }
00129 void ControlExperimentAnalysis::clusterization() {
00130     for ( const std::string& typeName : mTypeNameSet ) {
00131         TClusterization clusterization(mExpDataSet.find(std::string(typeName))->second->getEvents());
00132         clusterization.clusterize();
00133         mExpDataSet.find(std::string(typeName))->second->setClusters(clusterization.getClusters());
00134     }
00135 }
00136
00137 void ControlExperimentAnalysis::drawClustermapAndClustersize() {
00138     for ( const std::string& typeName : mTypeNameSet ) {
00139         if ( mConfig->getConfig(typeName).hasKey("clustermap") ) {
00140             CppConfigDictionary hitmapConfig =
00141             mConfig->getConfig(typeName).getSubConfig("clustermap");
00142             hitmapConfig += mConfig->getConfig("SharedProperty");
00143             mAnalyser->saveClustermap(typeName, hitmapConfig);
00144         }
00145         if ( mConfig->getConfig(typeName).hasKey("clustersize") ) {
00146             CppConfigDictionary hitmapConfig =
00147             mConfig->getConfig(typeName).getSubConfig("clustersize");
00148             hitmapConfig += mConfig->getConfig("SharedProperty");
00149             mAnalyser->saveClustersize(typeName, hitmapConfig);
00150         }
00151     }
00152 }
00153 void ControlExperimentAnalysis::doDivideBySize() {
00154     mClusterRange = getClusterSizeRange(mConfig->getConfig("ShapeCut"));
00155     for ( std::string_view typeName : mTypeNameSet ) {
00156         mAnalyser->doDivideBySize(typeName);
00157     }

```

```

00158 }
00159
00160 void ControlExperimentAnalysis::drawClusterShapeInfos() {
00161     for ( const std::string& typeName : mTypeNameSet ) {
00162         mAnalyser->doShaping(typeName, mClusterRange);
00163         if ( mConfig->getConfig(typeName).hasKey("shape_individual") ) {
00164             CppConfigDictionary shapeConfig =
00165                 mConfig->getConfig(typeName).getSubConfig("shape_individual");
00166             shapeConfig += mConfig->getConfig("SharedProperty");
00167             mAnalyser->saveIndividualShapes(typeName, shapeConfig);
00168         }
00169         if ( mConfig->getConfig(typeName).hasKey("shape_same_size") ) {
00170             CppConfigDictionary shapeConfig =
00171                 mConfig->getConfig(typeName).getSubConfig("shape_same_size");
00172             shapeConfig += mConfig->getConfig("SharedProperty");
00173             mAnalyser->saveSameSizeShapes(typeName, shapeConfig);
00174         }
00175         if ( mConfig->getConfig(typeName).hasKey("shape_total") ) {
00176             CppConfigDictionary shapeConfig =
00177                 mConfig->getConfig(typeName).getSubConfig("shape_total");
00178             shapeConfig += mConfig->getConfig("SharedProperty");
00179             mAnalyser->saveTotalShapes(typeName, shapeConfig);
00180         }
00181         if ( mConfig->getConfig(typeName).hasKey("shape_same_size_entry") ) {
00182             CppConfigDictionary shapeConfig =
00183                 mConfig->getConfig(typeName).getSubConfig("shape_same_size_entry");
00184             shapeConfig += mConfig->getConfig("SharedProperty");
00185             mAnalyser->saveSameSizeShapeEntry(typeName, shapeConfig);
00186         }
00187         if ( mConfig->getConfig(typeName).hasKey("shape_total_entry") ) {
00188             CppConfigDictionary shapeConfig =
00189                 mConfig->getConfig(typeName).getSubConfig("shape_total_entry");
00190             shapeConfig += mConfig->getConfig("SharedProperty");
00191             mAnalyser->saveTotalShapeEntry(typeName, shapeConfig);
00192         }
00193         if ( mConfig->getConfig(typeName).hasKey("shape_info") ) {
00194             CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_info");
00195             shapeConfig += mConfig->getConfig("SharedProperty");
00196             mAnalyser->saveSameSizeInfos(typeName, shapeConfig);
00197         }
00198     }
00199 }
00200
00201 std::vector<int> ControlExperimentAnalysis::getClusterSizeRange(const CppConfigDictionary
00202     privateProperty) {
00203     std::vector<int> clusterSizeRange;
00204     if ( privateProperty.hasKey("cluster_size_oi") ) {
00205         for ( const std::string& rangeStr :
00206             privateProperty.getSubConfig("cluster_size_oi").getValueList() ) {
00207             if ( rangeStr.find('.') != std::string::npos ) {
00208                 for ( int i = stoi(rangeStr.substr(0, rangeStr.find('.'))); i <
00209                     stoi(rangeStr.substr(rangeStr.find('.') + 3)) + 1; i++ ) {
00210                     clusterSizeRange.push_back(i);
00211                 }
00212             } else {
00213                 clusterSizeRange.push_back(stoi(rangeStr));
00214             }
00215         }
00216     } else {
00217         for ( int i = 0; i < 100; i++ ) {
00218             clusterSizeRange.push_back(i);
00219         }
00220     }
00221     return clusterSizeRange;
00222 }

```

8.95 /home/ychoi/ATOM/exe/GarfieldSimulation.cpp File Reference

```

#include <iostream>
#include <fstream>
#include <cmath>
#include <TCanvas.h>
#include <TRoot.h>
#include <TSystem.h>
#include <TApplication.h>
#include <TH1D.h>

```

```

#include "Garfield/MediumSilicon.hh"
#include "Garfield/ComponentConstant.hh"
#include "Garfield/ComponentUser.hh"
#include "Garfield/ComponentAnalyticField.hh"
#include "Garfield/Sensor.hh"
#include "Garfield/TrackHeed.hh"
#include "Garfield/AvalancheMC.hh"
#include "Garfield/ViewDrift.hh"
#include "Garfield/ViewField.hh"
#include "Garfield/ViewCell.hh"
#include "Garfield/Plotting.hh"
#include "Garfield/FundamentalConstants.hh"
#include "Garfield/Random.hh"
#include "Garfield/ComponentTcad2d.hh"

```

Functions

- `int main (int argc, char *argv[])`

8.95.1 Function Documentation

8.95.1.1 main()

```

int main (
    int argc,
    char * argv[ ] )

```

Definition at line 31 of file [GarfieldSimulation.cpp](#).

8.96 GarfieldSimulation.cpp

[Go to the documentation of this file.](#)

```

00001 #include <iostream>
00002 #include <fstream>
00003 #include <cmath>
00004
00005 #include <TCanvas.h>
00006 #include <TROOT.h>
00007 #include <TSystem.h>
00008 #include <TApplication.h>
00009 #include <TH1D.h>
00010
00011 #include "Garfield/MediumSilicon.hh"
00012 #include "Garfield/ComponentConstant.hh"
00013 #include "Garfield/ComponentUser.hh"
00014 #include "Garfield/ComponentAnalyticField.hh"
00015 #include "Garfield/Sensor.hh"
00016 #include "Garfield/TrackHeed.hh"
00017 #include "Garfield/AvalancheMC.hh"
00018
00019 #include "Garfield/ViewDrift.hh"
00020 #include "Garfield/ViewField.hh"
00021 #include "Garfield/ViewCell.hh"
00022 #include "Garfield/Plotting.hh"
00023
00024 #include "Garfield/FundamentalConstants.hh"
00025 #include "Garfield/Random.hh"
00026
00027 #include "Garfield/ComponentTcad2d.hh"
00028
00029 using namespace Garfield;

```

```

00030
00031 int main(int argc, char * argv[]) {
00032
00033     TApplication app("app", &argc, argv);
00034
00035     ComponentTcad2d tcad;
00036     tcad.Initialise("data/pixel_des.grd", "data/pixel_des.dat");
00037
00038     ComponentAnalyticField field;
00039     ViewCell view;
00040     view.SetComponent(&field);
00041     view.Plot2d();
00042
00043     // // Define the medium.
00044     // MediumSilicon si;
00045     // si.SetTemperature(293.);
00046
00047     // // Make a plot of the drift velocities.
00048     // plottingEngine.SetDefaultStyle();
00049
00050     // // Sensor thickness [cm]
00051     // constexpr double d = 100.e-4;
00052     // constexpr double w = 280.e-4;
00053
00054     // // Make a component with constant drift field and weighting field.
00055     // // Bias voltage [V]
00056     // constexpr double vbias = -0.;
00057     // ComponentConstant uniformField;
00058     // uniformField.SetArea(-2 * d, 0., -2 * d, 2 * d, d, 2 * d);
00059     // uniformField.SetMedium(&si);
00060     // uniformField.SetElectricField(0, vbias / d, 0);
00061     // uniformField.SetWeightingField(0, -1. / d, 0, "pad");
00062
00063     // // Depletion voltage [V]
00064     // constexpr double vdep = .7;
00065     // // Make a component with linear drift field.
00066     // auto eLinear = [d,vbias,vdep](const double /*x*/, const double y,
00067     //                               const double /*z*/,
00068     //                               double& ex, double& ey, double& ez) {
00069     //     ex = ez = 0.;
00070     //     ey = (vbias - vdep) / d + 2 * y * vdep / (d * d);
00071     // };
00072     // ComponentUser linearField;
00073     // linearField.SetArea(-2 * d, 0., -2 * d, 2 * d, d, 2 * d);
00074     // linearField.SetMedium(&si);
00075     // linearField.SetElectricField(eLinear);
00076     // // std::string efield = "ey = " + std::to_string((vbias - vdep) / d) +
00077     // // " + 2 * y * " + std::to_string(vdep / (d * d));
00078     // // linearField.SetElectricField(efield);
00079
00080     // // Make a component with analytic weighting field for a strip or pixel.
00081     // constexpr double pitch = 28.e-4;
00082     // constexpr double halfpitch = 0.5 * pitch;
00083     // ComponentAnalyticField wField;
00084     // wField.SetMedium(&si);
00085     // wField.AddPlaneY(0, vbias, "back");
00086     // wField.AddPlaneY(d, 0, "front");
00087     // wField.AddStripOnPlaneY('z', d, -halfpitch, halfpitch, "strip");
00088     // wField.AddPixelOnPlaneY(d, -halfpitch, halfpitch,
00089     //                          -halfpitch, halfpitch, "pixel");
00090
00091     // // Create a sensor.
00092     // Sensor sensor;
00093     // sensor.AddComponent(&linearField);
00094     // const std::string label = "strip";
00095     // sensor.AddElectrode(&wField, label);
00096
00097     // // Plot the drift field if requested.
00098     // constexpr bool plotField = true;
00099     // if (plotField) {
00100     //     ViewField* fieldView = new ViewField();
00101     //     fieldView->SetSensor(&sensor);
00102     //     fieldView->SetArea(-0.5 * d, 0, 0.5 * d, d);
00103     //     fieldView->PlotContour("ey");
00104     // }
00105     // // Plot the weighting potential if requested.
00106     // constexpr bool plotWeightingField = true;
00107     // if (plotWeightingField) {
00108     //     ViewField* wfieldView = new ViewField();
00109     //     wfieldView->SetComponent(&wField);
00110     //     wfieldView->SetArea(-0.5 * d, 0, 0.5 * d, d);
00111     //     wfieldView->PlotContourWeightingField("strip", "v");
00112     // }
00113
00114     // // Set the time bins.
00115     // const unsigned int nTimeBins = 1000;
00116     // const double tmin = 0.;

```



```

00117 // const double tmax = 10.;
00118 // const double tstep = (tmax - tmin) / nTimeBins;
00119 // sensor.SetTimeWindow(tmin, tstep, nTimeBins);
00120
00121 // // Set up Heed.
00122 // TrackHeed track(&sensor);
00123 // // Set the particle type and momentum [eV/c].
00124 // track.SetParticle("pion");
00125 // track.SetMomentum(180.e9);
00126
00127 // // Simulate electron/hole drift lines using MC integration.
00128 // AvalancheMC drift(&sensor);
00129 // // Use steps of 1 micron.
00130 // drift.SetDistanceSteps(1.e-4);
00131
00132 // // Plot the signal if requested.
00133 // constexpr bool plotSignal = true;
00134 // TCanvas* cSignal = nullptr;
00135 // if (plotSignal) {
00136 //     cSignal = new TCanvas("cSignal", "", 600, 600);
00137 // }
00138
00139 // constexpr bool plotDrift = true;
00140 // ViewDrift* driftView = nullptr;
00141 // TCanvas* cDrift = nullptr;
00142 // if (plotDrift) {
00143 //     cDrift = new TCanvas("cDrift", "", 600, 600);
00144 //     driftView = new ViewDrift();
00145 //     driftView->SetArea(-0.5 * d, 0, -0.5 * d, 0.5 * d, d, 0.5 * d);
00146 //     driftView->SetCanvas(cDrift);
00147 //     track.EnablePlotting(driftView);
00148 // }
00149 // // Flag to randomise the position of the track.
00150 // constexpr bool smearx = true;
00151 // constexpr unsigned int nEvents = 10;
00152 // // Flag to save the signal to a file.
00153 // constexpr bool writeSignal = true;
00154 // for (unsigned int i = 0; i < nEvents; ++i) {
00155 //     if (plotDrift) driftView->Clear();
00156 //     // Reset the signal.
00157 //     sensor.ClearSignal();
00158 //     if (i % 10 == 0) std::cout << i << "/" << nEvents << "\n";
00159 //     // Simulate a charged-particle track.
00160 //     double xt = 0.;
00161 //     if (smearx) xt = -0.5 * pitch + RndmUniform() * pitch;
00162 //     track.NewTrack(xt, 0, 0, 0, 0, 1, 0);
00163 //     // Retrieve the clusters along the track.
00164 //     for (const auto& cluster : track.GetClusters()) {
00165 //         // Loop over the electrons in the cluster.
00166 //         for (const auto& electron : cluster.electrons) {
00167 //             // Simulate the electron and hole drift lines.
00168 //             if (plotDrift) {
00169 //                 drift.DisablePlotting();
00170 //                 if (RndmUniform() < 0.01) drift.EnablePlotting(driftView);
00171 //             }
00172 //             drift.DriftElectron(electron.x, electron.y, electron.z, electron.t);
00173 //         }
00174 //     }
00175 //     if (plotSignal) {
00176 //         sensor.PlotSignal(label, cSignal);
00177 //         cSignal->Update();
00178 //         gSystem->ProcessEvents();
00179 //     }
00180 //     if (plotDrift) {
00181 //         constexpr bool twod = true;
00182 //         driftView->Plot(twod);
00183 //         cDrift->Update();
00184 //         gSystem->ProcessEvents();
00185 //     }
00186 //     // Save the induced current signal to a file.
00187 //     if (writeSignal) {
00188 //         char filename[50];
00189 //         sprintf(filename, "signal_%05d.txt", i);
00190 //         std::ofstream outfile;
00191 //         outfile.open(filename, std::ios::out);
00192 //         for (unsigned int j = 0; j < nTimeBins; ++j) {
00193 //             const double t = (j + 0.5) * tstep;
00194 //             const double f = sensor.GetSignal(label, j);
00195 //             const double fe = sensor.GetElectronSignal(label, j);
00196 //             const double fh = sensor.GetIonSignal(label, j);
00197 //             outfile << t << " " << f << " " << fe << " " << fh << "\n";
00198 //         }
00199 //         outfile.close();
00200 //     }
00201 // }
00202
00203 // if (plotSignal || plotDrift ||

```

```
00204     //      plotField || plotWeightingField) {  
00205     //      app.Run();  
00206     //  }  
00207 }
```

8.97 /home/ychoi/ATOM/exe/Merge.cpp File Reference

```
#include <iostream>  
#include <vector>  
#include <string>  
#include "cppargs.h"  
#include "CppConfigFile.h"  
#include "TMerge.h"
```

Functions

- [ArgumentParser](#) [set_parse](#) (int argc, char **argv)
- int [main](#) (int argc, char **argv)

8.97.1 Function Documentation

8.97.1.1 main()

```
int main (  
    int argc,  
    char ** argv )
```

Definition at line 16 of file [Merge.cpp](#).

8.97.1.2 set_parse()

```
ArgumentParser set_parse (  
    int argc,  
    char ** argv )
```

Definition at line 9 of file [Merge.cpp](#).

8.98 Merge.cpp

[Go to the documentation of this file.](#)

```
00001 #include <iostream>
00002 #include <vector>
00003 #include <string>
00004
00005 #include "cppargs.h"
00006 #include "CppConfigFile.h"
00007 #include "TMerge.h"
00008
00009 ArgumentParser set_parse(int argc, char** argv) {
00010     ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw plots for analysis data");
00011     parser.add_argument("config").help("Config file").set_default("default").add_finish();
00012     parser.parse_args();
00013     return parser;
00014 }
00015
00016 int main(int argc, char** argv) {
00017     ArgumentParser parser = set_parse(argc, argv);
00018     CppConfigFile* config = new CppConfigFile(parser.get_value<std::string>("config"));
00019
00020     TMergeExperimentROOT* merge = new
00021     TMergeExperimentROOT(config->getConfig("Merge").find("output_file"),
00022     config->getConfig("Merge").getSubConfig("input_files").getValueList());
00021
00022     merge->mergeFile();
00023
00024     delete config;
00025     delete merge;
00026     return 0;
00027 }
```

8.99 /home/ychoi/ATOM/exe/simulation.cpp File Reference

```
#include "Randomize.hh"
#include "G4RunManager.hh"
#include "QBBC.hh"
#include "G4SystemOfUnits.hh"
#include "G4ParticleTable.hh"
#include "G4SingleParticleSource.hh"
#include "G4GeneralParticleSource.hh"
#include "DetectorConstruction.h"
#include "ActionInitialization.h"
#include "AnalysisManager.h"
#include "CppConfigFile.h"
#include "cppargs.h"
#include "cppUnit.h"
#include "cppTimer.h"
```

Functions

- [ArgumentParser set_parse](#) (int argc, char **argv)
- void [DetectorConstruct](#) ([DetectorConstruction](#) *detectorConstructor, const [CppConfigDictionary](#) &config)
- void [SetParticleSource](#) ([G4SingleParticleSource](#) *particleGun, const [CppConfigDictionary](#) &config)
- int [main](#) (int argc, char **argv)

8.99.1 Function Documentation

8.99.1.1 DetectorConstruct()

```
void DetectorConstruct (
    DetectorConstruction * detectorConstructor,
    const CppConfigDictionary & config )
```

Definition at line 26 of file [simulation.cpp](#).

8.99.1.2 main()

```
int main (
    int argc,
    char ** argv )
```

Definition at line 63 of file [simulation.cpp](#).

8.99.1.3 set_parse()

```
ArgumentParser set_parse (
    int argc,
    char ** argv )
```

Definition at line 19 of file [simulation.cpp](#).

8.99.1.4 SetParticleSource()

```
void SetParticleSource (
    G4SingleParticleSource * particleGun,
    const CppConfigDictionary & config )
```

Definition at line 46 of file [simulation.cpp](#).

8.100 simulation.cpp

[Go to the documentation of this file.](#)

```
00001 #include "Randomize.hh"
00002 #include "G4RunManager.hh"
00003 #include "QBEC.hh"
00004
00005 #include "G4SystemOfUnits.hh"
00006 #include "G4ParticleTable.hh"
00007 #include "G4SingleParticleSource.hh"
00008 #include "G4GeneralParticleSource.hh"
00009
00010 #include "DetectorConstruction.h"
00011 #include "ActionInitialization.h"
00012 #include "AnalysisManager.h"
00013
00014 #include "CppConfigFile.h"
00015 #include "cppargs.h"
00016 #include "cppUnit.h"
00017 #include "cppTimer.h"
00018
00019 ArgumentParser set_parse(int argc, char** argv) {
```

```

00020     ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw plots for analysis data");
00021     parser.add_argument("--config").add_minor_argument("-c").help("Config file for controlling
simulation").set_default("").add_finish();
00022     parser.parse_args();
00023     return parser;
00024 }
00025
00026 void DetectorConstruct(DetectorConstruction* detectorConstructor, const CppConfigDictionary& config) {
00027     double airPressure = Quantity(config.find("air_pressure")).getNum("bar");
00028     detectorConstructor->SetWorld(airPressure);
00029
00030     double hallDiameter = Quantity(config.find("hall_diameter")).getNum("mm");
00031     std::string standType = config.find("stand_type");
00032     detectorConstructor->SetStand(standType, hallDiameter);
00033
00034     double shieldWidth = Quantity(config.find("shield_width")).getNum("mm");
00035     if ( !(shieldWidth < 0.001) ) {
00036         detectorConstructor->SetShield(shieldWidth);
00037     }
00038
00039     std::string alpideType = config.find("alpide_type");
00040     detectorConstructor->SetALPIDE(alpideType);
00041
00042     double distance = Quantity(config.find("distance")).getNum("mm");
00043     detectorConstructor->Construct(standType, distance);
00044 }
00045
00046 void SetParticleSource(G4SingleParticleSource* particleGun, const CppConfigDictionary& config) {
00047     G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
00048     std::string particleType = config.find("particle_type");
00049     G4ParticleDefinition* particleDefinition = particleTable->FindParticle(particleType);
00050     particleGun->SetParticleDefinition(particleDefinition); // /gps/particle alpha
00051     particleGun->GetPosDist()->SetPosDisType("Plane"); // /gps/pos/type Plane
00052     particleGun->GetPosDist()->SetPosDisShape("Circle"); // /gps/pos/shape
00053     particleGun->GetPosDist()->SetRadius(2.5 * mm); // /gps/pos/radius 2.5 * mm
00054     particleGun->GetPosDist()->SetPosRot1(G4ThreeVector(0., 0., 1.)); // /gps/pos/rot1 0. 0. 1.
00055     particleGun->GetPosDist()->SetPosRot2(G4ThreeVector(1., 0., 0.)); // /gps/pos/rot2 1. 0. 0.
00056     particleGun->GetAngDist()->SetAngDisType("iso"); // /gps/ang/type iso
00057     particleGun->GetEneDist()->SetEnergyDisType("Mono");
00058     particleGun->GetEneDist()->ApplyEnergyWeight(false); // /gps/ene/type Mono
00059     double distance = Quantity(config.find("distance")).getNum("mm");
00060     particleGun->GetPosDist()->SetCentreCoords(G4ThreeVector(0., distance, 0.)); // /gps/pos/centre 0.
2. 0.
00061 }
00062
00063 int main(int argc, char** argv) {
00064     TTimer timer;
00065     // Get config file using argument parser.
00066     ArgumentParser parser = set_parse(argc, argv);
00067     CppConfigFile config(parser.get_value<std::string>("config"));
00068     Quantity::setUserQuantity();
00069
00070     // Set random engine
00071     CLHEP::RanecuEngine* RandomEngine = new CLHEP::RanecuEngine;
00072     G4Random::setTheEngine(RandomEngine);
00073
00074     // Define Run Manager
00075     G4RunManager* runManager = new G4RunManager;
00076
00077     // Analysis Manager
00078     AnalysisManager* analysisManager = new
AnalysisManager(config.getConfig("Analysis").find("output_file"));
00079
00080     // Geometry construction
00081     DetectorConstruction* detectorConstructor = new DetectorConstruction();
00082     DetectorConstruct(detectorConstructor, config.getConfig("Geometry"));
00083     runManager->SetUserInitialization(detectorConstructor);
00084
00085     // Set physics
00086     G4VModularPhysicsList* QBBCList = new QBBC(0);
00087     G4VUserPhysicsList* physicsList = QBBCList;
00088     physicsList->SetVerboseLevel(0);
00089     runManager->SetUserInitialization(physicsList);
00090
00091     // Set initial action
00092     G4VUserActionInitialization* actionInitialiator = new ActionInitialization();
00093     runManager->SetUserInitialization(actionInitialiator);
00094
00095     // Set Particle Source
00096     G4GeneralParticleSourceData* GPSData = G4GeneralParticleSourceData::Instance();
00097     G4SingleParticleSource* particleGun = GPSData->GetCurrentSource();
00098     SetParticleSource(particleGun, config.getConfig("Source"));
00099
00100     runManager->Initialize();
00101
00102     std::ifstream distFile(config.getConfig("Energy").find("alpha_energy_distribution"));
00103     std::string line;

```

```

00104     getline(distFile, line);
00105     while ( getline(distFile, line) ) {
00106         std::stringstream sstr(line, ',');
00107         std::string energyStr;
00108         std::string intensityStr;
00109         std::getline(sstr, energyStr, ',');
00110         std::getline(sstr, intensityStr, ',');
00111         std::getline(sstr, intensityStr, ',');
00112         if ( intensityStr == "" ) {
00113             continue;
00114         }
00115         double energy = stoi(energyStr) * 0.001;
00116         int intensity = static_cast<int>(round(stod(intensityStr) * 2000));
00117         particleGun->GetEneDist()->SetMonoEnergy(energy * MeV);
00118         runManager->BeamOn(intensity);
00119     }
00120     analysisManager->closeBook();
00121
00122     delete analysisManager;
00123     delete runManager;
00124     delete RandomEngine;
00125     timer.EndProgram();
00126     return 0;
00127 }

```

8.101 /home/ychoi/ATOM/exe/SimulationAnalysis.cpp File Reference

```

#include <filesystem>
#include "TGeantAnalyser.h"
#include "cppargs.h"

```

Functions

- [ArgumentParser set_parse](#) (int argc, char **argv)
- int [main](#) (int argc, char **argv)

8.101.1 Function Documentation

8.101.1.1 main()

```

int main (
    int argc,
    char ** argv )

```

Definition at line 14 of file [SimulationAnalysis.cpp](#).

8.101.1.2 set_parse()

```

ArgumentParser set_parse (
    int argc,
    char ** argv )

```

Definition at line 6 of file [SimulationAnalysis.cpp](#).

8.102 SimulationAnalysis.cpp

[Go to the documentation of this file.](#)

```
00001 #include <filesystem>
00002
00003 #include "TGeantAnalyser.h"
00004 #include "cppargs.h"
00005
00006 ArgumentParser set_parse(int argc, char** argv) {
00007     ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw plots for simulation");
00008     parser.add_argument("simulFile").metavar("FILENAME").help("simulation file").add_finish();
00009     parser.add_argument("--output").help("output path").set_default("default").add_finish();
00010     parser.parse_args();
00011     return parser;
00012 }
00013
00014 int main(int argc, char** argv) {
00015     // ArgumentParser parser = set_parse(argc, argv);
00016     // std::filesystem::path input(parser.get_value<std::string>("simulFile"));
00017     // std::filesystem::path path;
00018
00019     // if ( parser.get_value<std::string>("output") == "default" ) {
00020     //     path = std::filesystem::absolute(input.parent_path());
00021     // } else {
00022     //     path = parser.get_value<std::string>("output");
00023     // }
00024
00025     // std::filesystem::path stem = input.stem();
00026     // TFile* file = new TFile(static_cast<TString>((input.replace_extension(".root")).string()),
00027     "READ");
00028     // path.append(stem.string());
00029     // std::filesystem::create_directory(path);
00030     // TGeantAnalyser drawer(std::move(file));
00031     // drawer.refineData();
00032     // std::filesystem::path energyLossFile = "energy_loss_" + stem.string();
00033     // drawer.saveEnergyLossDistribution((path / energyLossFile.replace_extension(".png")).string());
00034     // std::filesystem::path clustermapFile = "clustermap_" + stem.string();
00035     // drawer.saveClustermap((path / clustermapFile.replace_extension(".png")).string());
00036     // std::filesystem::path doubleClusterFile = "Doubled_cluster_frequencymap_" + stem.string();
00037     // drawer.saveDoubleClusterFrequencymap((path /
00038     doubleClusterFile.replace_extension(".png")).string());
00039
00040     return 0;
00041 }
```

8.103 /home/ychoi/ATOM/exe/ThresholdAnalysis.cpp File Reference

```
#include <iostream>
#include <string>
#include <filesystem>
#include "TThresholdAnalyser.h"
#include "TString.h"
#include "cppargs.h"
```

Functions

- [ArgumentParser set_parse](#) (int argc, char **argv)
- int [main](#) (int argc, char **argv)

8.103.1 Function Documentation

8.103.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

Definition at line 19 of file [ThresholdAnalysis.cpp](#).

8.103.1.2 set_parse()

```
ArgumentParser set_parse (
    int argc,
    char ** argv )
```

Definition at line 9 of file [ThresholdAnalysis.cpp](#).

8.104 ThresholdAnalysis.cpp

[Go to the documentation of this file.](#)

```
00001 #include <iostream>
00002 #include <string>
00003 #include <filesystem>
00004
00005 #include "TThresholdAnalyser.h"
00006 #include "TString.h"
00007 #include "cppargs.h"
00008
00009 ArgumentParser set_parse(int argc, char** argv) {
00010     ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw threshold information of
an ALPIDE");
00011     parser.add_argument("thrFile").metavar("FILENAME").help("threshold file to be
processed").add_finish();
00012     parser.add_argument("--output").help("output path").set_default("default").help("output
path").add_finish();
00013     parser.add_argument("--drawing_plots").nargs().set_default("default").add_domain({"threshold_distribution",
"error_distribution", "thresholded_map", "quality_factor"}).help("Choose the kind of plots to
draw").add_finish();
00014     parser.parse_args();
00015     return parser;
00016 }
00017
00018
00019 int main(int argc, char** argv) {
00020     ArgumentParser parser = set_parse(argc, argv);
00021
00022     std::filesystem::path input(parser.get_value<std::string>("thrFile"));
00023     std::filesystem::path path;
00024
00025     if ( parser.get_value<std::string>("output") == "default" ) {
00026         path = std::filesystem::absolute(input.parent_path());
00027     } else {
00028         path = parser.get_value<std::string>("output");
00029     }
00030
00031     std::filesystem::path stem = input.stem();
00032
00033     std::ifstream file(input.replace_extension(".dat"));
00034     TThresholdAnalyser analyser(file);
00035     analyser.refineData();
00036
00037     std::vector<std::string> plotList = parser.get_value<std::vector<std::string>("drawing_plots");
00038     if ( find(plotList.begin(), plotList.end(), "threshold_distribution") != plotList.end() ) {
00039         std::filesystem::create_directory(path / stem);
00040         analyser.saveThresholdDistribution(static_cast<TString>(path / stem /
"Threshold_distribution.png"));
00041     }
00042     if ( find(plotList.begin(), plotList.end(), "error_distribution") != plotList.end() ) {
00043         std::filesystem::create_directory(path / stem);
00044         analyser.saveErrorDistribution(static_cast<TString>(path / stem / "Error_distribution.png"));
00045     }
00046     if ( find(plotList.begin(), plotList.end(), "threshold_map") != plotList.end() ) {
00047         std::filesystem::create_directory(path / stem);
00048         analyser.saveThresholdmap(static_cast<TString>(path / stem / "Thresholdmap.png"));
00049     }
00050     if ( find(plotList.begin(), plotList.end(), "quality_factor") != plotList.end() ) {
00051         std::filesystem::create_directory(path / stem);
00052         analyser.saveQualityDistribution(static_cast<TString>(path / stem /
"Quality_distribution.png"));
00053     }
00054     return 0;
00055 }
```


8.105 /home/ychoi/ATOM/geant4/analysis/inc/TGeantAnalyser.h File Reference

```
#include "TAnalyser.h"
```

Classes

- class [TGeantAnalyser](#)

8.106 TGeantAnalyser.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TGEANTANALYSER__
00002 #define __TGEANTANALYSER__
00003
00004 #include "TAnalyser.h"
00005
00006 class TGeantAnalyser : public TAnalyser {
00007 private:
00008
00009     // std::unique_ptr<TTree> tSource;
00010     // std::unique_ptr<TTree> tTrack;
00011
00012     // std::unique_ptr<TH1D> eLossHist[6];
00013     // std::unique_ptr<TH1D> eDep;
00014     // std::unique_ptr<TH2D> clusterMap;
00015     // std::unique_ptr<TH2D> angleMap;
00016     // std::unique_ptr<TH2D> doubleClusterMap;
00017     // double clusterDensity;
00018 public:
00019     TGeantAnalyser() = delete;
00020     // TGeantAnalyser(TString inputFile);
00021     // TGeantAnalyser(TFile* inputFile);
00022     // ~TGeantAnalyser();
00023     // void refineData();
00024     // void saveEnergyLossDistribution(std::string_view output_file_Name);
00025     // void saveTotalDepositEnergyInALPIDE(std::string_view output_file_Name);
00026     // void saveClustermap(std::string_view output_file_Name);
00027     // void saveAnglemap(std::string_view output_file_Name);
00028     // void saveDoubleClusterFrequencymap(std::string_view output_file_Name);
00029     // const double getClusterDensity() const;
00030 };
00031
00032 #endif
```

8.107 /home/ychoi/ATOM/geant4/analysis/src/TGeantAnalyser.cpp File Reference

```
#include "TGeantAnalyser.h"
```

8.108 TGeantAnalyser.cpp

[Go to the documentation of this file.](#)

```
00001 #include "TGeantAnalyser.h"
00002
00003 // TGeantAnalyser::TGeantAnalyser(TString inputFile) : TAnalyser(inputFile) {}
00004 // TGeantAnalyser::TGeantAnalyser(TFile* inputFile) : TAnalyser(inputFile) {}
00005
00006 // TGeantAnalyser::~TGeantAnalyser() {}
00007
00008 // void TGeantAnalyser::refineData() {
00009 //     Float_t Eloss[6] = {1,1,1,1,1,1};
00010 //     std::array<Double_t, 3> position;
00011 //     bool pass[6] = {false, false, false, false, false, false};
00012 //     std::array<Double_t, 2> angle;
00013 //     Double_t minX=3., minZ=3., maxX=-3., maxZ=-3.;
00014 //     Int_t cntParticle = 0;
00015
00016 //     tTrack->SetBranchAddress("posX",&position[0]);
00017 //     tTrack->SetBranchAddress("posY",&position[1]);
00018 //     tTrack->SetBranchAddress("posZ",&position[2]);
00019 //     tTrack->SetBranchAddress("Eloss",&Eloss);
00020 //     tTrack->SetBranchAddress("Passed",&pass);
00021 //     tTrack->SetBranchAddress("dirTheta",&angle[0]);
00022 //     tTrack->SetBranchAddress("dirPhi",&angle[1]);
00023
00024 //     for (int entry = 0; entry < tTrack->GetEntries(); entry++) {
00025 //         tTrack->GetEntry(entry);
00026 //         if ((pass[3] || pass[4]) && TMath::Abs(position[0]) < 15. && TMath::Abs(position[2]) < 6.5)
00027 //         {
00028 //             clusterMap->Fill(position[0], position[2]);
00029 //             doubleClusterMap->Fill(position[0], position[2]);
00030 //             angleMap->Fill(angle[0], angle[1]);
00031 //             minX=TMath::Min(minX, position[0]);
00032 //             minZ=TMath::Min(minZ, position[2]);
00033 //             maxX=TMath::Max(maxX, position[0]);
00034 //             maxZ=TMath::Max(maxZ, position[2]);
00035 //             cntParticle++;
00036 //         }
00037 //         eLossHist[0]->Fill(Eloss[0]);
00038 //         eLossHist[1]->Fill(Eloss[1]);
00039 //         eLossHist[2]->Fill(Eloss[2]);
00040 //         eLossHist[3]->Fill(Eloss[3]);
00041 //         eLossHist[4]->Fill(Eloss[4]);
00042 //         eLossHist[5]->Fill(Eloss[5]);
00043 //         if (pass[3] && pass[4]) {
00044 //             eDep->Fill(Eloss[3]+Eloss[4]);
00045 //         }
00046 //         Double_t area = TMath::Pi() * pow((maxX - minX) + (maxZ - minZ) )/4,2);
00047
00048 //         clusterDensity = (0.00025 * 4300 * cntParticle / area) / tTrack->GetEntries();
00049 //         Double_t probability = (pow(30./32.,2)-TMath::Pi()*(0.09991))/pow(30./32.,2);
00050 //         Double_t density = 0.;
00051 //         for (int x = 0; x < 32; x++) {
00052 //             for (int y = 0; y < 16; y++) {
00053 //                 density = (0.0025 * 4300 * doubleClusterMap->GetBinContent(x,y) / pow(30./32.,2)) /
00054 //                 tTrack->GetEntries();
00055 //                 doubleClusterMap->SetBinContent(x,y,1-pow(probability, density));
00056 //                 if (doubleClusterMap->GetBinContent(x,y) > 1.) {
00057 //                     std::cout << density << std::endl;
00058 //                 }
00059 //             }
00060 //         }
00061
00062 // void TGeantAnalyser::saveEnergyLossDistribution(std::string_view title) {
00063 //     for (std::unique_ptr<TH1D>& hist : eLossHist) {
00064 //         std::unique_ptr<TCanvas> canvase(new TCanvas("can","can",1610,1000));
00065 //         hist->SetBinContent(1,0);
00066 //         hist->Draw();
00067 //         canvase->SaveAs(static_cast<TString>(title));
00068 //     }
00069 // }
00070
00071 // void TGeantAnalyser::saveTotalDepositEnergyInALPIDE(std::string_view title) {
00072 //     std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
00073 //     eDep->Draw();
00074 //     can->SaveAs(static_cast<const TString>(title));
00075 // }
00076
00077 // void TGeantAnalyser::saveClustermap(std::string_view title) {
00078 //     std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
00079 //     clusterMap->Draw();
00080 //     can->SaveAs(static_cast<const TString>(title));
00081 // }
```

```

00081 // }
00082
00083 // void TGeantAnalyser::saveAnglemap(std::string_view title) {
00084 //     std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
00085 //     angleMap->Draw();
00086 //     can->SaveAs(static_cast<const TString>(title));
00087 // }
00088
00089 // void TGeantAnalyser::saveDoubleClusterFrequencyMap(std::string_view title) {
00090 //     std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
00091 //     doubleClusterMap->SetMaximum(1.);
00092 //     doubleClusterMap->SetStats(0);
00093 //     doubleClusterMap->Draw("COLZ");
00094 //     can->SaveAs(static_cast<const TString>(title));
00095 // }
00096
00097 // const double TGeantAnalyser::getClusterDensity() const {
00098 //     return clusterDensity;
00099 // }

```

8.109 /home/ychoi/ATOM/geant4/main/inc/ActionInitialization.h File Reference

```
#include "G4VUserActionInitialization.hh"
```

Classes

- class [ActionInitialization](#)

8.110 ActionInitialization.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __ACTIONINITIALIZATION__
00002 #define __ACTIONINITIALIZATION__
00003
00004 #ifdef __ACTIONINITIALIZATION_HEADER__
00005 #include "G4ios.hh"
00006 #include "RunAction.h"
00007 #include "EventAction.h"
00008 #include "SteppingAction.h"
00009 #include "TrackingAction.h"
00010 #include "PrimaryGeneratorAction.h"
00011 #endif
00012
00013 #include "G4VUserActionInitialization.hh"
00014
00015
00016 class ActionInitialization : public G4VUserActionInitialization {
00017 public:
00018     ActionInitialization();
00019     virtual ~ActionInitialization();
00020
00021     virtual void Build() const;
00022     virtual void BuildForMaster() const;
00023 };
00024
00025 #endif

```

8.111 /home/ychoi/ATOM/geant4/main/inc/AnalysisManager.h File Reference

```
#include <string>
```

Classes

- struct [RunTuple](#)
- struct [EventTuple](#)
- struct [TrackTuple](#)
- struct [StepTuple](#)
- class [AnalysisManager](#)

8.112 AnalysisManager.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __ANALYSISMANAGER__
00002 #define __ANALYSISMANAGER__
00003
00004 #ifdef __ANALYSISMANAGER_HEADER__
00005 #include "G4Run.hh"
00006 #include "G4Event.hh"
00007 #include "G4Track.hh"
00008 #include "G4Step.hh"
00009 #include "G4RunManager.hh"
00010 #include "TFile.h"
00011 #include "TTree.h"
00012 #endif
00013
00014 #include <string>
00015
00016 class TFile;
00017 class TTree;
00018
00019 class G4Run;
00020 class G4Event;
00021 class G4Track;
00022 class G4Step;
00023
00024 struct RunTuple {
00025     int runID;
00026     int nEvents;
00027 };
00028
00029 struct EventTuple {
00030     int runID;
00031     int eventGlobalID;
00032     int eventLocalID;
00033     int nTracks;
00034 };
00035
00036 struct TrackTuple {
00037     int eventGlobalID;
00038     int trackGlobalID;
00039     int trackLocalID;
00040     int trackParentLocalID;
00041     std::string particleName;
00042     std::string processName;
00043     std::string volumeName;
00044     double genTime;
00045     double genPosition[3];
00046     double genKineticEnergy;
00047     double genMomentum[3];
00048     int nSteps;
00049 };
00050
00051 struct StepTuple {
00052     int trackGlobalID;
00053     int stepGlobalID;
00054     std::string volumeName;
00055     double time;
00056     double position[3];
00057     double kineticEnergy;
00058     double momentum[3];
00059     double deltaEnergy;
00060     double totalDepositEnergy;
00061     double nonIonizingEnergyLoss;
00062 };
00063
00064 class AnalysisManager {
00065 private:
00066     int runID = 0;
00067     int eventID = 0;

```

```

00068     int trackID = 0;
00069     int stepID = 0;
00070
00071     static AnalysisManager* fInstance;
00072
00073     TFile* mOutputFile;
00074     TTree* mRunTree;
00075     TTree* mEventTree;
00076     TTree* mTrackTree;
00077     TTree* mStepTree;
00078
00079     RunTuple runTuple;
00080     EventTuple eventTuple;
00081     TrackTuple trackTuple;
00082     StepTuple stepTuple;
00083 public:
00084     AnalysisManager();
00085     AnalysisManager(std::string name);
00086     ~AnalysisManager();
00087
00088     static AnalysisManager* Instance();
00089
00090     void setEventID(const int id) { eventID = id; }
00091     void setTrackID(const int id) { trackID = id; }
00092     int getEventID() const { return eventID; }
00093     int getTrackID() const { return trackID; }
00094
00095     void openBook(std::string name);
00096
00097     void setRunTree();
00098     void setEventTree();
00099     void setTrackTree();
00100     void setStepTree();
00101
00102     void RecordingRun(const G4Run* run);
00103     void RecordingEvent(const G4Event* event);
00104     void RecordingTrackStart(const G4Track* track);
00105     void RecordingTrackEnd(const G4Track* track);
00106     void RecordingStep(const G4Step* step);
00107
00108     void closeBook();
00109
00110     RunTuple& getRunTuple() { return runTuple; }
00111     EventTuple& getEventTuple() { return eventTuple; }
00112     TrackTuple& getTrackTuple() { return trackTuple; }
00113     StepTuple& getStepTuple() { return stepTuple; }
00114 };
00115
00116 #endif

```

8.113 /home/ychoi/ATOM/geant4/main/inc/Colour.h File Reference

Classes

- class [Colour](#)

8.114 Colour.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __COLOUR__
00002 #define __COLOUR__
00003
00004 #ifdef __COLOUR_HEADER__
00005 #include "G4VisAttributes.hh"
00006 #endif
00007
00008 class G4VisAttributes;
00009
00010 class Colour {
00011 private:
00012     G4VisAttributes* standColour;
00013     G4VisAttributes* screenColour;
00014     G4VisAttributes* alpineColour;
00015     G4VisAttributes* boardColour;

```

```

00016 public:
00017     Colour();
00018
00019     void setStandColour();
00020     void setScreenColour();
00021     void setAlpideColour();
00022     void setBoardColour();
00023
00024     G4VisAttributes* getStandColour() const;
00025     G4VisAttributes* getScreenColour() const;
00026     G4VisAttributes* getAlpideColour() const;
00027     G4VisAttributes* getBoardColour() const;
00028 };
00029
00030 #endif

```

8.115 /home/ychoi/ATOM/geant4/main/inc/DetectorConstruction.h File Reference

```
#include "G4VUserDetectorConstruction.hh"
```

Classes

- class [DetectorConstruction](#)

Enumerations

- enum [SourceType](#) { [alpha](#) , [beta](#) , [photon](#) }
- enum [StandType](#) { [alpha_no_screen](#) , [alpha_screen](#) , [beta_no_screen](#) , [beta_screen](#) , [alpha_new_no_screen](#) , [alpha_new_screen](#) , [none](#) }

8.115.1 Enumeration Type Documentation

8.115.1.1 SourceType

```
enum SourceType
```

Enumerator

alpha	
beta	
photon	

Definition at line 24 of file [DetectorConstruction.h](#).

8.115.1.2 StandType

```
enum StandType
```

Enumerator

alpha_no_screen	
alpha_screen	
beta_no_screen	
beta_screen	
alpha_new_no_screen	
alpha_new_screen	
none	

Definition at line 30 of file [DetectorConstruction.h](#).

8.116 DetectorConstruction.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __DETECTORCONSTRUCTION__
00002 #define __DETECTORCONSTRUCTION__
00003
00004 #ifdef __DETECTORCONSTRUCTION_HEADER__
00005 #include "Material.h"
00006 #include "Colour.h"
00007 #include "Solid.h"
00008 #include "G4Box.hh"
00009 #include "G4VPhysicalVolume.hh"
00010 #include "G4LogicalVolume.hh"
00011 #include "G4AssemblyVolume.hh"
00012 #include "G4UserLimits.hh"
00013 #include "G4PVPlacement.hh"
00014 #include "G4SystemOfUnits.hh"
00015 #endif
00016
00017 #include "G4VUserDetectorConstruction.hh"
00018
00019 class Material;
00020 class Colour;
00021 class Solid;
00022 class G4AssemblyVolume;
00023
00024 enum SourceType {
00025     alpha,
00026     beta,
00027     photon
00028 };
00029
00030 enum StandType {
00031     alpha_no_screen,
00032     alpha_screen,
00033     beta_no_screen,
00034     beta_screen,
00035     alpha_new_no_screen,
00036     alpha_new_screen,
00037     none
00038 };
00039
00040 class DetectorConstruction : public G4VUserDetectorConstruction {
00041 private:
00042     Material* material = nullptr;
00043     Colour* colour = nullptr;
00044     Solid* solids = nullptr;
00045
00046     G4VPhysicalVolume* WorldPhysical = nullptr;
00047     G4VPhysicalVolume* StandPhysical = nullptr;
00048     G4VPhysicalVolume* ShieldPhysical = nullptr;
00049     G4VPhysicalVolume* CarrierBoardPhysical = nullptr;
00050
00051     G4LogicalVolume* WorldLogical = nullptr;
00052     G4LogicalVolume* StandLogical = nullptr;
00053     G4LogicalVolume* ShieldLogical = nullptr;
00054     G4LogicalVolume* ALPIDEcircuitLogical = nullptr;
00055     G4LogicalVolume* ALPIDEpitaxialLogical = nullptr;
00056     G4AssemblyVolume* ALPIDEAssembly = nullptr;
00057     G4LogicalVolume* CarrierBoardLogical = nullptr;
00058

```

```

00059     SourceType sourceType;
00060     StandType standType = StandType::none;
00061     G4double sEnergy = 5.4;
00062     G4double sDistance = 10.;
00063     G4double cDiameter = 1.;
00064 public:
00065     DetectorConstruction();
00066     virtual ~DetectorConstruction();
00067     virtual G4VPhysicalVolume* Construct();
00068     G4VPhysicalVolume* Construct(G4String standType, G4double distance);
00069
00070     // Setter
00071     void SetWorld(G4double air_pressure);
00072     void SetStand(G4String standType, G4double hallDiameter = 0.);
00073     void SetShield(G4double shieldWidth);
00074     void SetALPIDE(G4String alpideType);
00075     void SetCarrierBoard();
00076
00077     // Getter
00078     G4LogicalVolume* GetScoringStand() const;
00079     G4LogicalVolume* GetScoringShield() const;
00080     G4LogicalVolume* GetScoringALPIDEcircuit() const;
00081     G4LogicalVolume* GetScoringALPIDEpitaxial() const;
00082     G4LogicalVolume* GetScoringCarrierBoard() const;
00083
00084     void SetEnergy(G4double energy);
00085     void SetSourceType(G4String type);
00086     void SetStandType(G4String type);
00087     void SetDistance(G4double distance);
00088     void SetVacuum(G4double vacuum);
00089     void SetHallWidth(G4double diameter);
00090 private:
00091     void DefineCommands();
00092 };
00093
00094 #endif

```

8.117 /home/ychoi/ATOM/geant4/main/inc/EventAction.h File Reference

```
#include "G4UserEventAction.hh"
```

Classes

- class [EventAction](#)

Typedefs

- typedef int [G4int](#)

8.117.1 Typedef Documentation

8.117.1.1 G4int

```
typedef int G4int
```

Definition at line 16 of file [EventAction.h](#).

8.118 EventAction.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __EVENTACTION__
00002 #define __EVENTACTION__
00003
00004 #ifdef __EVENTACTION_HEADER__
00005 #include "G4Event.hh"
00006 #include "AnalysisManager.h"
00007 #include "RunAction.h"
00008 #include "G4GeneralParticleSourceData.hh"
00009 #include "G4SingleParticleSource.hh"
00010 #endif
00011
00012 #include "G4UserEventAction.hh"
00013
00014 class RunAction;
00015 class G4Event;
00016 typedef int G4int;
00017
00018 class EventAction : public G4UserEventAction {
00019 private:
00020     G4int iStartTrack;
00021
00022 public:
00023     EventAction(RunAction* runAction);
00024     virtual ~EventAction();
00025
00026     virtual void BeginOfEventAction(const G4Event* event);
00027     virtual void EndOfEventAction(const G4Event* event);
00028 };
00029
00030 #endif

```

8.119 /home/ychoi/ATOM/geant4/main/inc/Material.h File Reference

Classes

- class [Material](#)

8.120 Material.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __MATERIAL__
00002 #define __MATERIAL__
00003
00004 #ifdef __MATERIAL_HEADER__
00005 #include "G4Element.hh"
00006 #include "G4Material.hh"
00007 // #include "G4NistManager.hh"
00008 #include "G4SystemOfUnits.hh"
00009 #endif
00010
00011 class G4Material;
00012 class G4Element;
00013
00014 class Material {
00015 private:
00016     G4Material* worldMaterial;
00017     G4Material* standMaterial;
00018     G4Material* screenMaterial;
00019     G4Material* alpideMaterial;
00020     G4Material* boardMaterial;
00021
00022     G4Material* epoxyResin;
00023     G4Material* fibrousGlass;
00024
00025     G4Element* elSi;
00026     G4Element* elN;
00027     G4Element* elO;
00028     G4Element* elAl;
00029     G4Element* elFe;

```

```

00030     G4Element* elCa;
00031     G4Element* elMg;
00032     G4Element* elNa;
00033     G4Element* elTi;
00034     G4Element* elC;
00035     G4Element* elH;
00036     G4Element* elBr;
00037 public:
00038     Material();
00039
00040     G4Material* getWorldMaterial() const;
00041     G4Material* getStandMaterial() const;
00042     G4Material* getScreenMaterial() const;
00043     G4Material* getAlpideMaterial() const;
00044     G4Material* getBoardMaterial() const;
00045
00046     void setElement();
00047     void setWorldMaterial(const double density = 1.);
00048     void setStandMaterial();
00049     void setScreenMaterial();
00050     void setAlpideMaterial();
00051     void setBoardMaterial();
00052 };
00053
00054 #endif

```

8.121 /home/ychoi/ATOM/geant4/main/inc/PrimaryGeneratorAction.h File Reference

```
#include "G4VUserPrimaryGeneratorAction.hh"
```

Classes

- class [PrimaryGeneratorAction](#)

8.122 PrimaryGeneratorAction.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __PRIMARYGENERATORACTION__
00002 #define __PRIMARYGENERATORACTION__
00003
00004 #ifdef __PRIMARYGENERATORACTION_HEADER__
00005 #include "G4ios.hh"
00006 #include "G4GeneralParticleSource.hh"
00007 #endif
00008
00009 #include "G4VUserPrimaryGeneratorAction.hh"
00010
00011 class G4GeneralParticleSource;
00012
00013 class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction {
00014 private:
00015     G4GeneralParticleSource* fParticleGun;
00016 public:
00017     PrimaryGeneratorAction();
00018     virtual ~PrimaryGeneratorAction();
00019
00020     void setParticleGun();
00021
00022     void GeneratePrimaries(G4Event* anEvent);
00023
00024     const G4GeneralParticleSource* GetParticleGun() const;
00025 };
00026
00027 #endif

```

8.123 /home/ychoi/ATOM/geant4/main/inc/RunAction.h File Reference

```
#include "G4UserRunAction.hh"
```

Classes

- class [RunAction](#)

8.124 RunAction.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __RUNACTION__
00002 #define __RUNACTION__
00003
00004 #ifdef __RUNACTION_HEADER__
00005 #include "G4ios.hh"
00006 #include "G4Run.hh"
00007 #include "AnalysisManager.h"
00008 #endif
00009
00010 #include "G4UserRunAction.hh"
00011 // #include "G4Run.hh"
00012
00013 // #include "PrimaryGeneratorAction.h"
00014 // #include "AnalysisManager.h"
00015 class AnalysisManager;
00016
00017 class RunAction : public G4UserRunAction {
00018 public:
00019     RunAction();
00020     virtual ~RunAction();
00021
00022     virtual void BeginOfRunAction(const G4Run*);
00023     virtual void EndOfRunAction(const G4Run*);
00024
00025     AnalysisManager* fAnalysisManager;
00026 };
00027
00028 #endif
```

8.125 /home/ychoi/ATOM/geant4/main/inc/Solid.h File Reference

Classes

- class [Solid](#)

8.126 Solid.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __SOLID__
00002 #define __SOLID__
00003
00004 #ifdef __SOLID_HEADER__
00005 #include "G4Box.hh"
00006 #include "G4Tubs.hh"
00007 #include "G4SubtractionSolid.hh"
00008 #include "G4UnionSolid.hh"
00009 #include "G4SystemOfUnits.hh"
00010 #endif
00011
00012 class G4VSolid;
```

```

00013
00014
00015 class Solid {
00016 private:
00017     G4VSolid* alphaStandSolid;
00018     G4VSolid* betaStandSolid;
00019     G4VSolid* newStandSolid;
00020     G4VSolid* screenSolid;
00021     G4VSolid* alpideCircuitSolid;
00022     G4VSolid* alpideEpitaxialSolid;
00023     G4VSolid* boardSolid;
00024 public:
00025     Solid();
00026
00027     void setAlphaStandSolid();
00028     void setBetaStandSolid();
00029     void setNewStandSolid(double diameter);
00030     void setScreenSolid();
00031     void setAlpideCircuitSolid();
00032     void setAlpideEpitaxialSolid();
00033     void setBoardSolid();
00034
00035     G4VSolid* getAlphaStandSolid() const;
00036     G4VSolid* getBetaStandSolid() const;
00037     G4VSolid* getNewStandSolid() const;
00038     G4VSolid* getScreenSolid() const;
00039     G4VSolid* getAlpideCircuitSolid() const;
00040     G4VSolid* getAlpideEpitaxialSolid() const;
00041     G4VSolid* getBoardSolid() const;
00042 };
00043
00044 #endif

```

8.127 /home/ychoi/ATOM/geant4/main/inc/SteppingAction.h File Reference

```
#include "G4UserSteppingAction.hh"
```

Classes

- class [SteppingAction](#)

8.128 SteppingAction.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __STEPPINGACTION__
00002 #define __STEPPINGACTION__
00003
00004 #ifdef __STEPPINGACTION_HEADER__
00005 #include "G4ios.hh"
00006 #include "EventAction.h"
00007 #include "AnalysisManager.h"
00008 #endif
00009
00010 #include "G4UserSteppingAction.hh"
00011
00012 class EventAction;
00013 class G4Step;
00014
00015 class SteppingAction : public G4UserSteppingAction {
00016 public:
00017     SteppingAction(EventAction* eventAction);
00018     virtual ~SteppingAction();
00019
00020     virtual void UserSteppingAction(const G4Step*);
00021 };
00022
00023 #endif

```

8.129 /home/ychoi/ATOM/geant4/main/inc/TrackingAction.h File Reference

```
#include "G4UserTrackingAction.hh"
```

Classes

- class [TrackingAction](#)

8.130 TrackingAction.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __TRACKINGACTION__
00002 #define __TRACKINGACTION__
00003
00004 #ifdef __TRACKINGACTION_HEADER__
00005 #include "G4ios.hh"
00006 #include "AnalysisManager.h"
00007 #endif
00008
00009 #include "G4UserTrackingAction.hh"
00010
00011 class TrackingAction : public G4UserTrackingAction {
00012 public:
00013     TrackingAction();
00014     ~TrackingAction();
00015
00016     virtual void PreUserTrackingAction(const G4Track*);
00017     virtual void PostUserTrackingAction(const G4Track*);
00018 };
00019
00020 #endif
```

8.131 /home/ychoi/ATOM/geant4/main/src/ActionInitialization.cpp File Reference

```
#include "ActionInitialization.h"
```

Macros

- #define [__ACTIONINITIALIZATION_HEADER__](#)

8.131.1 Macro Definition Documentation

8.131.1.1 [__ACTIONINITIALIZATION_HEADER__](#)

```
#define __ACTIONINITIALIZATION_HEADER__
```

Definition at line 1 of file [ActionInitialization.cpp](#).

8.132 ActionInitialization.cpp

[Go to the documentation of this file.](#)

```
00001 #define __ACTIONINITIALIZATION_HEADER__
00002 #include "ActionInitialization.h"
00003
00004 ActionInitialization::ActionInitialization() : G4VUserActionInitialization() {
00005     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " is armed" << G4endl;
00006 }
00007
00008 ActionInitialization::~ActionInitialization() {
00009     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " is terminated" << G4endl;
00010 }
00011
00012 void ActionInitialization::Build() const {
00013     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " boots" << "\033[1;36m" << " Primary
Generator Action" << "\033[0m" << G4endl;
00014     SetUserAction(new PrimaryGeneratorAction);
00015     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " boots" << "\033[1;36m" << " Run Action"
<< "\033[0m" << G4endl;
00016     RunAction* runAction = new RunAction;
00017     SetUserAction(runAction);
00018
00019     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " boots" << "\033[1;36m" << " Event
Action" << "\033[0m" << G4endl;
00020     EventAction* eventAction = new EventAction(runAction);
00021     SetUserAction(eventAction);
00022
00023     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " boots" << "\033[1;36m" << " Stepping
Action" << "\033[0m" << G4endl;
00024     SetUserAction(new SteppingAction(eventAction));
00025
00026     G4cout << "\033[1;35m" << "ActionInitializer" << "\033[0m" << " boots" << "\033[1;36m" << " Tracking
Action" << "\033[0m" << G4endl;
00027     SetUserAction(new TrackingAction);
00028 }
00029
00030 void ActionInitialization::BuildForMaster() const {
00031     G4cout << "ActionInitialization::BuildForMaster()" << G4endl;
00032     RunAction* runAction = new RunAction;
00033     SetUserAction(runAction);
00034 }
```

8.133 /home/ychoi/ATOM/geant4/main/src/AnalysisManager.cpp File Reference

```
#include "AnalysisManager.h"
```

Macros

- `#define __ANALYSISMANAGER_HEADER__`

8.133.1 Macro Definition Documentation

8.133.1.1 __ANALYSISMANAGER_HEADER__

```
#define __ANALYSISMANAGER_HEADER__
```

Definition at line 1 of file [AnalysisManager.cpp](#).

8.134 AnalysisManager.cpp

[Go to the documentation of this file.](#)

```

00001 #define __ANALYSISMANAGER_HEADER__
00002 #include "AnalysisManager.h"
00003
00004 AnalysisManager* AnalysisManager::fInstance = nullptr;
00005
00006 AnalysisManager::AnalysisManager() {
00007     G4cout << "\033[1;35m" << "AnalysisManger" << "\033[0m" << " is armed" << G4endl;
00008     fInstance = this;
00009 }
00010
00011 AnalysisManager::AnalysisManager(std::string name) {
00012     G4cout << "\033[1;35m" << "AnalysisManger" << "\033[0m" << " is armed" << G4endl;
00013     fInstance = this;
00014     openBook(name);
00015 }
00016
00017 AnalysisManager::~AnalysisManager() {
00018     G4cout << "\033[1;35m" << "AnalysisManager" << "\033[0m" << " is terminated" << G4endl;
00019 }
00020
00021 AnalysisManager* AnalysisManager::Instance() {
00022     if ( fInstance == 0 ) {
00023         fInstance = new AnalysisManager();
00024     }
00025     return fInstance;
00026 }
00027
00028 void AnalysisManager::openBook(std::string name) {
00029     mOutputFile = new TFile(static_cast<TString>(name), "RECREATE");
00030     setRunTree();
00031     setEventTree();
00032     setTrackTree();
00033     setStepTree();
00034 }
00035
00036 void AnalysisManager::setRunTree() {
00037     mRunTree = new TTree("runTree", "Tree for recording run informations");
00038     mRunTree->Branch("Run ID", &runTuple.runID, "runID/I");
00039     mRunTree->Branch("Number of Event", &runTuple.nEvents, "nEvent/I");
00040 }
00041
00042 void AnalysisManager::setEventTree() {
00043     mEventTree = new TTree("eventTree", "Tree for recording event informations");
00044     mEventTree->Branch("Event Global ID", &eventTuple.eventGlobalID, "eventGlobalID/I");
00045     mEventTree->Branch("Event Local ID", &eventTuple.eventLocalID, "eventLocalID/I");
00046     mEventTree->Branch("Referenced Run ID", &eventTuple.runID, "runID/I");
00047     mEventTree->Branch("Number of Track", &eventTuple.nTracks, "nTrack/I");
00048 }
00049
00050 void AnalysisManager::setTrackTree() {
00051     mTrackTree = new TTree("trackTree", "Tree for recording track informations");
00052     mTrackTree->Branch("Track Local ID", &trackTuple.trackLocalID, "trackLocalID/I");
00053     mTrackTree->Branch("Track Global ID", &trackTuple.trackGlobalID, "trackGlobalID/I");
00054     mTrackTree->Branch("Referenced Event Global ID", &trackTuple.eventGlobalID, "eventGlobalID/I");
00055     mTrackTree->Branch("Parent Local ID", &trackTuple.trackParentLocalID, "trackParentLocalID/I");
00056     mTrackTree->Branch("Number of Steps", &trackTuple.nSteps, "nSteps/I");
00057     mTrackTree->Branch("Particle Name", &trackTuple.particleName, "particleName");
00058     mTrackTree->Branch("Process Name", &trackTuple.processName, "processName");
00059     mTrackTree->Branch("Generating Volume Name", &trackTuple.volumeName, "volumeName");
00060     mTrackTree->Branch("Generating Time", &trackTuple.genTime, "genTime/D");
00061     mTrackTree->Branch("Generating Position X", &trackTuple.genPosition[0], "genPositionX/D");
00062     mTrackTree->Branch("Generating Position Y", &trackTuple.genPosition[1], "genPositionY/D");
00063     mTrackTree->Branch("Generating Position Z", &trackTuple.genPosition[2], "genPositionZ/D");
00064     mTrackTree->Branch("Generating Kinetic Energy", &trackTuple.genKineteicEnergy,
"genKineteicEnergy/D");
00065     mTrackTree->Branch("Generating Momentum X", &trackTuple.genMomentum[0], "genMomentumX/D");
00066     mTrackTree->Branch("Generating Momentum Y", &trackTuple.genMomentum[1], "genMomentumY/D");
00067     mTrackTree->Branch("Generating Momentum Z", &trackTuple.genMomentum[2], "genMomentumZ/D");
00068 }
00069
00070 void AnalysisManager::setStepTree() {
00071     mStepTree = new TTree("stepTree", "Tree for recording step information");
00072     mStepTree->Branch("Step Global ID", &stepTuple.stepGlobalID, "stepGlobalID/I");
00073     mStepTree->Branch("Referenced Track Global ID", &stepTuple.trackGlobalID, "trackGlobalID/I");
00074     mStepTree->Branch("Volume Name", &stepTuple.volumeName, "volumeName");
00075     mStepTree->Branch("Time", &stepTuple.time, "time/D");
00076     mStepTree->Branch("Position X", &stepTuple.position[0], "time/D");
00077     mStepTree->Branch("Position Y", &stepTuple.position[1], "time/D");
00078     mStepTree->Branch("Position Z", &stepTuple.position[2], "time/D");
00079     mStepTree->Branch("Kinetic Energy", &stepTuple.kineticEnergy, "time/D");
00080     mStepTree->Branch("Momentum X", &stepTuple.momentum[0], "time/D");
00081     mStepTree->Branch("Momentum Y", &stepTuple.momentum[1], "time/D");

```

```

00082     mStepTree->Branch("Momentum Z", &stepTuple.momentum[2], "time/D");
00083     mStepTree->Branch("Delta Energy", &stepTuple.deltaEnergy, "time/D");
00084     mStepTree->Branch("Total Deposit Energy", &stepTuple.totalDepositEnergy, "time/D");
00085     mStepTree->Branch("Non Ionizing Deposit Energy", &stepTuple.nonIonizingEnergyLoss, "time/D");
00086 }
00087
00088 void AnalysisManager::RecordingRun(const G4Run* run) {
00089     runID = run->GetRunID();
00090     runTuple.runID = runID;
00091     runTuple.nEvents = run->GetNumberOfEvent();
00092     mRunTree->Fill();
00093     mRunTree->AutoSave();
00094 }
00095
00096 void AnalysisManager::RecordingEvent(const G4Event* event) {
00097     eventTuple.eventGlobalID = eventID;
00098     eventTuple.eventLocalID = event->GetEventID();
00099     eventTuple.runID = runID;
00100     mEventTree->Fill();
00101     mEventTree->AutoSave();
00102     eventID++;
00103 }
00104
00105 void AnalysisManager::RecordingTrackStart(const G4Track* track) {
00106     trackTuple.eventGlobalID = eventID;
00107     trackTuple.trackGlobalID = trackID;
00108     if ( track->GetTrackID() != 1 ) {
00109         trackTuple.trackParentLocalID = track->GetParentID();
00110         trackTuple.processName = track->GetCreatorProcess()->GetProcessName();
00111     }
00112     trackTuple.particleName = track->GetParticleDefinition()->GetParticleName();
00113     trackTuple.volumeName = track->GetVolume()->GetName();
00114     trackTuple.genTime = track->GetGlobalTime();
00115     trackTuple.genPosition[0] = track->GetPosition()[0];
00116     trackTuple.genPosition[1] = track->GetPosition()[1];
00117     trackTuple.genPosition[2] = track->GetPosition()[2];
00118     trackTuple.genKineticEnergy = track->GetKineticEnergy();
00119     trackTuple.genMomentum[0] = track->GetMomentum()[0];
00120     trackTuple.genMomentum[1] = track->GetMomentum()[1];
00121     trackTuple.genMomentum[2] = track->GetMomentum()[2];
00122     trackID++;
00123 }
00124
00125 void AnalysisManager::RecordingTrackEnd(const G4Track* track) {
00126     trackTuple.nSteps = track->GetCurrentStepNumber();
00127     mTrackTree->Fill();
00128     mTrackTree->AutoSave();
00129     trackTuple.trackParentLocalID = 0;
00130     trackTuple.processName = "";
00131 }
00132
00133 void AnalysisManager::RecordingStep(const G4Step* step) {
00134     stepTuple.stepGlobalID = stepID;
00135     stepTuple.trackGlobalID = trackID;
00136     stepTuple.position[0] = step->GetPostStepPoint()->GetPosition()[0];
00137     stepTuple.position[1] = step->GetPostStepPoint()->GetPosition()[1];
00138     stepTuple.position[2] = step->GetPostStepPoint()->GetPosition()[2];
00139     if ( (abs(stepTuple.position[0]) < 149) && (abs(stepTuple.position[1]) < 149) &&
00140         (abs(stepTuple.position[2]) < 74)) ) {
00141         stepTuple.volumeName = step->GetPostStepPoint()->GetPhysicalVolume()->GetName();
00142     } else {
00143         stepTuple.volumeName = "World";
00144     }
00145     stepTuple.time = step->GetPostStepPoint()->GetLocalTime();
00146     stepTuple.kineticEnergy = step->GetPostStepPoint()->GetKineticEnergy();
00147     stepTuple.momentum[0] = step->GetPostStepPoint()->GetMomentum()[0];
00148     stepTuple.momentum[1] = step->GetPostStepPoint()->GetMomentum()[1];
00149     stepTuple.momentum[2] = step->GetPostStepPoint()->GetMomentum()[2];
00150     stepTuple.totalDepositEnergy = step->GetTotalEnergyDeposit();
00151     stepTuple.nonIonizingEnergyLoss = step->GetNonIonizingEnergyDeposit();
00152     mStepTree->Fill();
00153     mStepTree->AutoSave();
00154     stepID++;
00155 }
00156
00157 void AnalysisManager::closeBook() {
00158     mOutputFile->cd();
00159     mRunTree->Write();
00160     mEventTree->Write();
00161     mTrackTree->Write();
00162     mStepTree->Write();
00163     mOutputFile->Close();
00164 }
00165

```


8.135 /home/ychoi/ATOM/geant4/main/src/Colour.cpp File Reference

```
#include "Colour.h"
```

Macros

- `#define __COLOUR_HEADER__`

8.135.1 Macro Definition Documentation

8.135.1.1 __COLOUR_HEADER__

```
#define __COLOUR_HEADER__
```

Definition at line 1 of file [Colour.cpp](#).

8.136 Colour.cpp

[Go to the documentation of this file.](#)

```
00001 #define __COLOUR_HEADER__
00002 #include "Colour.h"
00003
00004 Colour::Colour() {
00005     setStandColour();
00006     setScreenColour();
00007     setAlpideColour();
00008     setBoardColour();
00009 }
00010
00011 void Colour::setStandColour() {
00012     standColour = new G4VisAttributes(G4Colour(1., 1., 1.));
00013     standColour->SetVisibility(true);
00014     standColour->SetForceSolid(true);
00015 }
00016
00017 void Colour::setScreenColour() {
00018     screenColour = new G4VisAttributes(G4Colour(211 / 255., 211 / 255., 211 / 255.));
00019     screenColour->SetVisibility(true);
00020     screenColour->SetForceSolid(true);
00021 }
00022
00023 void Colour::setAlpideColour() {
00024     alpideColour = new G4VisAttributes(G4Colour(1., 1., 0.));
00025     alpideColour->SetVisibility(true);
00026     alpideColour->SetForceSolid(true);
00027 }
00028
00029 void Colour::setBoardColour() {
00030     boardColour = new G4VisAttributes(G4Colour(0 / 255., 100 / 255., 0 / 255.));
00031     boardColour->SetVisibility(true);
00032     boardColour->SetForceSolid(true);
00033 }
00034
00035 G4VisAttributes* Colour::getStandColour() const {
00036     return standColour;
00037 }
00038
00039 G4VisAttributes* Colour::getScreenColour() const {
00040     return screenColour;
00041 }
00042
00043 G4VisAttributes* Colour::getAlpideColour() const {
00044     return alpideColour;
00045 }
00046
00047 G4VisAttributes* Colour::getBoardColour() const {
00048     return boardColour;
00049 }
```

8.137 /home/ychoi/ATOM/geant4/main/src/DetectorConstruction.cpp File Reference

```
#include "DetectorConstruction.h"
```

Macros

- `#define __DETECTORCONSTRUCTION_HEADER__`

8.137.1 Macro Definition Documentation

8.137.1.1 __DETECTORCONSTRUCTION_HEADER__

```
#define __DETECTORCONSTRUCTION_HEADER__
```

Definition at line 1 of file [DetectorConstruction.cpp](#).

8.138 DetectorConstruction.cpp

[Go to the documentation of this file.](#)

```
00001 #define __DETECTORCONSTRUCTION_HEADER__
00002 #include "DetectorConstruction.h"
00003
00004 DetectorConstruction::DetectorConstruction() : G4VUserDetectorConstruction() {
00005     G4cout << "\033[1;35m" << "Detector Constructor" << "\033[0m" << " is armed" << G4endl;
00006     material = new Material();
00007     colour = new Colour();
00008     solids = new Solid();
00009 }
00010
00011 DetectorConstruction::~DetectorConstruction() {
00012     delete material;
00013     delete colour;
00014     delete solids;
00015     G4cout << "\033[1;35m" << "Detector Constructor" << "\033[0m" << " is terminated" << G4endl;
00016 }
00017
00018 G4VPhysicalVolume* DetectorConstruction::Construct() {
00019     return WorldPhysical;
00020 }
00021
00022 G4VPhysicalVolume* DetectorConstruction::Construct(G4String standType, G4double distance) {
00023     G4bool checkOverlaps = true;
00024     if ( WorldLogical != nullptr ) {
00025         WorldPhysical = new G4PVPlacement(0, G4ThreeVector(), WorldLogical, "World", 0, false, 0,
00026     checkOverlaps);
00027     }
00028     if ( StandLogical != nullptr ) {
00029         if ( standType == "type1" ) {
00030             StandPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, (distance - 1.) * mm, 0. *
00031 mm), StandLogical, "StandLogical", WorldLogical, false, 0, checkOverlaps);
00032         } else if ( standType == "type2" ) {
00033             StandPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, (distance - 52.5) * mm, 0. *
00034 mm), StandLogical, "StandLogical", WorldLogical, false, 0, checkOverlaps);
00035         } else if ( standType == "type3" ) {
00036             StandPhysical->SetTranslation(G4ThreeVector(0. * mm, (distance + 2) * mm, -1.25 * mm));
00037         }
00038     }
00039     if ( ShieldLogical != nullptr ) {
00040         if ( standType == "type1" ) {
00041             ShieldPhysical->SetTranslation(G4ThreeVector(0. * mm, (distance - 1. - 4. - 0.5 * 0.108) *
00042 mm, 0. * mm));
00043         } else if ( standType == "type2" ) {
```

```

00042         ShieldPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, (distance - 56. - 0.5 *
0.108) * mm, 0. * mm), ShieldLogical, "ShieldLogical", WorldLogical, false, 0, checkOverlaps);
00043     } else if ( standType == "type3" ) {
00044         ShieldPhysical->SetTranslation(G4ThreeVector(0. * mm, (distance - 1. - .5 * 0.108) * mm,
-1.25 * mm));
00045     }
00046 }
00047
00048 if ( CarrierBoardLogical != nullptr ) {
00049     G4Transform3D t3d = G4Transform3D();
00050     ALPIDEAssembly->MakeImprint(WorldLogical, t3d);
00051     CarrierBoardPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, -1.0 * mm * 0.5 + -100 * um
* 0.5, -2.8 * mm), CarrierBoardLogical, "CarrierBoard", WorldLogical, false, 0, !checkOverlaps);
00052 }
00053
00054 return WorldPhysical;
00055 }
00056
00057 void DetectorConstruction::SetWorld(G4double air_pressure) {
00058     material->setWorldMaterial(air_pressure);
00059     G4Box* solidWorld = new G4Box("World", .5 * 300. * mm, .5 * 300. * mm, .5 * 150. * mm);
00060     WorldLogical = new G4LogicalVolume(solidWorld, material->getWorldMaterial(), "World");
00061 }
00062
00063 void DetectorConstruction::SetStand(G4String standType, G4double hallDiameter) {
00064     G4VSolid* solid;
00065     if ( standType == "type1" ) {
00066         solids->setAlphaStandSolid();
00067         solid = solids->getAlphaStandSolid();
00068     } else if ( standType == "type2" ) {
00069         solids->setBetaStandSolid();
00070         solid = solids->getBetaStandSolid();
00071     } else if ( standType == "type3" ) {
00072         solids->setNewStandSolid(hallDiameter);
00073         solid = solids->getNewStandSolid();
00074     } else {
00075         std::cerr << "\033[1;31m" << "Invalid stand type" << "\033[0m" << std::endl;
00076         exit(0);
00077     }
00078     StandLogical = new G4LogicalVolume(solid, material->getStandMaterial(), "StandLogical");
00079     StandLogical->SetVisAttributes(colour->getStandColour());
00080 }
00081
00082 void DetectorConstruction::SetShield(G4double shieldWidth) {
00083     solids->setScreenSolid();
00084     G4VSolid* solid = solids->getScreenSolid();
00085     ShieldLogical = new G4LogicalVolume(solid, material->getScreenMaterial(), "ShieldLogical");
00086     ShieldLogical->SetVisAttributes(colour->getScreenColour());
00087 }
00088
00089 void DetectorConstruction::SetALPIDE(G4String alpideType) {
00090     if ( alpideType == "insensitive" ) {
00091         G4VSolid* ALPIDECircuitSolid = solids->getAlpideCircuitSolid();
00092         G4VSolid* ALPIDEEpitaxialSolid = solids->getAlpideEpitaxialSolid();
00093
00094         ALPIDECircuitLogical = new G4LogicalVolume(ALPIDECircuitSolid, material->getAlpideMaterial(),
"ALPIDECircuitLogical");
00095         ALPIDECircuitLogical->SetVisAttributes(colour->getAlpideColour());
00096         ALPIDECircuitLogical->SetUserLimits(new G4UserLimits(1 * um, 1 * um));
00097         ALPIDEEpitaxialLogical = new G4LogicalVolume(ALPIDEEpitaxialSolid,
material->getAlpideMaterial(), "ALPIDEEpitaxialLogical");
00098         ALPIDEEpitaxialLogical->SetVisAttributes(colour->getAlpideColour());
00099         ALPIDEEpitaxialLogical->SetUserLimits(new G4UserLimits(1 * um, 1 * um));
00100
00101         ALPIDEAssembly = new G4AssemblyVolume();
00102         G4RotationMatrix* Ra = new G4RotationMatrix(0.0 * deg, 0.0 * deg, 0.0 * deg);
00103         G4ThreeVector circuitVector = G4ThreeVector(0, (50.0 - 11.0 * 0.5) * um, 0);
00104         ALPIDEAssembly->AddPlacedVolume(ALPIDECircuitLogical, circuitVector, Ra);
00105         G4ThreeVector epitaxialVector = G4ThreeVector(0, -(11.0) * 0.5 * um, 0);
00106         ALPIDEAssembly->AddPlacedVolume(ALPIDEEpitaxialLogical, epitaxialVector, Ra);
00107
00108         SetCarrierBoard();
00109     } else {
00110         G4cerr << "\033[1;31m" << "Invalid ALPIDE type" << "\033[0m" << G4endl;
00111         exit(0);
00112     }
00113 }
00114
00115 void DetectorConstruction::SetCarrierBoard() {
00116     G4VSolid* solid = solids->getBoardSolid();
00117     CarrierBoardLogical = new G4LogicalVolume(solid, material->getBoardMaterial(),
"CarrierBoardLogical");
00118     CarrierBoardLogical->SetVisAttributes(colour->getBoardColour());
00119 }
00120
00121 G4LogicalVolume* DetectorConstruction::GetScoringStand() const {
00122     return StandLogical;

```

```

00123 }
00124
00125 G4LogicalVolume* DetectorConstruction::GetScoringShield() const {
00126     return ShieldLogical;
00127 }
00128
00129 G4LogicalVolume* DetectorConstruction::GetScoringALPIDEcircuit() const {
00130     return ALPIDEcircuitLogical;
00131 }
00132
00133 G4LogicalVolume* DetectorConstruction::GetScoringALPIDEpitaxial() const {
00134     return ALPIDEpitaxialLogical;
00135 }
00136
00137 G4LogicalVolume* DetectorConstruction::GetScoringCarrierBoard() const {
00138     return CarrierBoardLogical;
00139 }

```

8.139 /home/ychoi/ATOM/geant4/main/src/EventAction.cpp File Reference

```
#include "EventAction.h"
```

Macros

- #define [__EVENTACTION_HEADER__](#)

8.139.1 Macro Definition Documentation

8.139.1.1 [__EVENTACTION_HEADER__](#)

```
#define __EVENTACTION_HEADER__
```

Definition at line 1 of file [EventAction.cpp](#).

8.140 EventAction.cpp

[Go to the documentation of this file.](#)

```

00001 #define __EVENTACTION_HEADER__
00002 #include "EventAction.h"
00003
00004 EventAction::EventAction(RunAction* runAction) : G4UserEventAction() {
00005     G4cout << "\033[1;36m" << "Event Action" << "\033[0m" << " is armed" << G4endl;
00006 }
00007
00008 EventAction::~EventAction() {
00009     G4cout << "\033[1;36m" << "Event Action" << "\033[0m" << " is terminated" << G4endl;
00010 }
00011
00012 void EventAction::BeginOfEventAction(const G4Event* event) {
00013     if ( event->GetEventID() == 0 ) {
00014         G4GeneralParticleSourceData* GPSData = G4GeneralParticleSourceData::Instance();
00015         G4SingleParticleSource* particleGun = GPSData->GetCurrentSource();
00016         G4float energy = particleGun->GetParticleEnergy();
00017         G4cout << " of Events with " << "\033[1;32m" << energy << "\033[0m" << " MeV Energy." << G4endl;
00018     }
00019     AnalysisManager* analysisManager = AnalysisManager::Instance();
00020     iStartTrack = analysisManager->getTrackID();
00021     // G4RunManager* runManager;
00022 }
00023
00024 void EventAction::EndOfEventAction(const G4Event* event) {
00025     AnalysisManager* analysisManager = AnalysisManager::Instance();
00026     analysisManager->getEventTuple().nTracks = analysisManager->getTrackID() - iStartTrack;
00027     analysisManager->RecordingEvent(event);
00028 }

```

8.141 /home/ychoi/ATOM/geant4/main/src/Material.cpp File Reference

```
#include "Material.h"
```

Macros

- `#define __MATERIAL_HEADER__`

8.141.1 Macro Definition Documentation

8.141.1.1 __MATERIAL_HEADER__

```
#define __MATERIAL_HEADER__
```

Definition at line 1 of file [Material.cpp](#).

8.142 Material.cpp

[Go to the documentation of this file.](#)

```
00001 #define __MATERIAL_HEADER__
00002 #include "Material.h"
00003
00004 Material::Material() {
00005     setElement();
00006
00007     setWorldMaterial();
00008     setStandMaterial();
00009     setScreenMaterial();
00010     setAlpideMaterial();
00011     setBoardMaterial();
00012 }
00013
00014 void Material::setElement() {
00015     elSi = new G4Element("Silicon", "Si", 14, 28.085 * g / mole);
00016     elN = new G4Element("Nitrogen", "N", 7., 14.007 * g / mole);
00017     elO = new G4Element("Oxygen", "O", 8, 15.999 * g / mole);
00018     elAl = new G4Element("Aluminium", "Al", 13, 26.982 * g / mole);
00019     elFe = new G4Element("Iron", "Fe", 26, 55.845 * g / mole);
00020     elCa = new G4Element("Calcium", "Ca", 20, 40.078 * g / mole);
00021     elMg = new G4Element("Magnesium", "Mg", 12, 24.305 * g / mole);
00022     elNa = new G4Element("Natrium", "Na", 11, 22.990 * g / mole);
00023     elTi = new G4Element("Titanium", "Ti", 22, 47.867 * g / mole);
00024     elC = new G4Element("Carbon", "C", 6, 12.011 * g / mole);
00025     elH = new G4Element("Hydrogen", "H", 1, 1.008 * g / mole);
00026     elBr = new G4Element("Bromine", "Br", 35, 79.904 * g / mole);
00027 }
00028
00029 void Material::setWorldMaterial(const double pressure) {
00030     worldMaterial = new G4Material("worldMaterial", pressure * 1.2929e-03 * g / cm3, 2);
00031     worldMaterial->AddElement(elN, .7);
00032     worldMaterial->AddElement(elO, .3);
00033 }
00034
00035 void Material::setStandMaterial() {
00036     standMaterial = new G4Material("PLA", 0.3 * 1.210 * g / cm3, 3);
00037     standMaterial->AddElement(elC, 3);
00038     standMaterial->AddElement(elH, 4);
00039     standMaterial->AddElement(elO, 2);
00040 }
00041
00042 void Material::setScreenMaterial() {
00043     screenMaterial = new G4Material("Aluminium", 13, 26.982 * g / mole, 2.7 * g / cm3);
00044 }
00045
00046 void Material::setAlpideMaterial() {
00047     alpideMaterial = new G4Material("Silicon", 14, 28.085 * g / mole, 2.33 * g / cm3);
```

```

00048 }
00049
00050 void Material::setBoardMaterial() {
00051     fibrousGlass = new G4Material("fibrousGlass", 2.74351 * g / cm3, 7);
00052
00053     G4Material* SiO2 = new G4Material("SiO2", 2.20 * g / cm3, 2);
00054     SiO2->AddElement(elSi, 1);
00055     SiO2->AddElement(elO, 2);
00056
00057     G4Material* Al2O3 = new G4Material("Al2O3", 3.97 * g / cm3, 2);
00058     Al2O3->AddElement(elAl, 2);
00059     Al2O3->AddElement(elO, 3);
00060
00061     G4Material* Fe2O3 = new G4Material("Fe2O3", 5.24 * g / cm3, 2);
00062     Fe2O3->AddElement(elFe, 2);
00063     Fe2O3->AddElement(elO, 3);
00064
00065     G4Material* CaO = new G4Material("CaO", 3.35 * g / cm3, 2);
00066     CaO->AddElement(elCa, 1);
00067     CaO->AddElement(elO, 1);
00068
00069     G4Material* MgO = new G4Material("MgO", 3.58 * g / cm3, 2);
00070     MgO->AddElement(elMg, 1);
00071     MgO->AddElement(elO, 1);
00072
00073     G4Material* Na2O = new G4Material("Na2O", 2.27 * g / cm3, 2);
00074     Na2O->AddElement(elNa, 2);
00075     Na2O->AddElement(elO, 1);
00076
00077     G4Material* TiO2 = new G4Material("TiO2", 4.24 * g / cm3, 2);
00078     TiO2->AddElement(elTi, 1);
00079     TiO2->AddElement(elO, 2);
00080
00081     fibrousGlass->AddMaterial(SiO2, 60.0 * perCent);
00082     fibrousGlass->AddMaterial(Al2O3, 11.8 * perCent);
00083     fibrousGlass->AddMaterial(Fe2O3, 0.1 * perCent);
00084     fibrousGlass->AddMaterial(CaO, 22.4 * perCent);
00085     fibrousGlass->AddMaterial(MgO, 3.4 * perCent);
00086     fibrousGlass->AddMaterial(Na2O, 1.0 * perCent);
00087     fibrousGlass->AddMaterial(TiO2, 1.3 * perCent);
00088
00089     epoxyResin = new G4Material("epoxyResin", 1.1250 * g / cm3, 4);
00090     epoxyResin->AddElement(elC, 38);
00091     epoxyResin->AddElement(elH, 40);
00092     epoxyResin->AddElement(elO, 6);
00093     epoxyResin->AddElement(elBr, 4);
00094
00095     boardMaterial = new G4Material("Fr4", 1.98281 * mg / cm3, 2);
00096     boardMaterial->AddMaterial(fibrousGlass, 53 * perCent);
00097     boardMaterial->AddMaterial(epoxyResin, 47 * perCent);
00098 }
00099
00100 G4Material* Material::getWorldMaterial() const {
00101     return worldMaterial;
00102 }
00103
00104 G4Material* Material::getStandMaterial() const {
00105     return standMaterial;
00106 }
00107
00108 G4Material* Material::getScreenMaterial() const {
00109     return screenMaterial;
00110 }
00111
00112 G4Material* Material::getAlpideMaterial() const {
00113     return alpideMaterial;
00114 }
00115
00116 G4Material* Material::getBoardMaterial() const {
00117     return boardMaterial;
00118 }

```

8.143 /home/ychoi/ATOM/geant4/main/src/PrimaryGeneratorAction.cpp

File Reference

```
#include "PrimaryGeneratorAction.h"
```

Macros

- [#define __PRIMARYGENERATORACTION_HEADER__](#)

8.143.1 Macro Definition Documentation**8.143.1.1 __PRIMARYGENERATORACTION_HEADER__**

```
#define __PRIMARYGENERATORACTION_HEADER__
```

Definition at line 1 of file [PrimaryGeneratorAction.cpp](#).

8.144 PrimaryGeneratorAction.cpp

[Go to the documentation of this file.](#)

```
00001 #define __PRIMARYGENERATORACTION_HEADER__
00002 #include "PrimaryGeneratorAction.h"
00003
00004 PrimaryGeneratorAction::PrimaryGeneratorAction() : G4VUserPrimaryGeneratorAction(), fParticleGun(0) {
00005     G4cout << "\033[1;36m" << "Primary Generator Action" << "\033[0m" << " is armed" << G4endl;
00006     // G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
00007     // G4String particleName;
00008
00009     fParticleGun = new G4GeneralParticleSource();
00010     // fParticleGun->SetParticlePosition(G4ThreeVector(0., +14.0 * mm, 0.));
00011     // fParticleGun->SetParticleDefinition(particleTable->FindParticle(particleName = "alpha"));
00012 }
00013
00014 PrimaryGeneratorAction::~PrimaryGeneratorAction() {
00015     G4cout << "\033[1;36m" << "Primary Generator Action" << "\033[0m" << " is terminated" << G4endl;
00016     delete fParticleGun;
00017 }
00018
00019 void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent) {
00020     fParticleGun->GeneratePrimaryVertex(anEvent);
00021 }
00022
00023 const G4GeneralParticleSource* PrimaryGeneratorAction::GetParticleGun() const {
00024     return fParticleGun;
00025 }
```

8.145 /home/ychoi/ATOM/geant4/main/src/RunAction.cpp File Reference

```
#include "RunAction.h"
```

Macros

- [#define __RUNACTION_HEADER__](#)

8.145.1 Macro Definition Documentation**8.145.1.1 __RUNACTION_HEADER__**

```
#define __RUNACTION_HEADER__
```

Definition at line 1 of file [RunAction.cpp](#).

8.146 RunAction.cpp

[Go to the documentation of this file.](#)

```
00001 #define __RUNACTION_HEADER__
00002 #include "RunAction.h"
00003
00004 RunAction::RunAction() : G4UserRunAction() {
00005     G4cout << "\033[1;36m" << "Run Action" << "\033[0m" << " is armed" << G4endl;
00006     fAnalysisManager = AnalysisManager::Instance();
00007 }
00008
00009 RunAction::~RunAction() {
00010     G4cout << "\033[1;36m" << "Run Action" << "\033[0m" << " is terminated" << G4endl;
00011 }
00012
00013 void RunAction::BeginOfRunAction(const G4Run* run) {
00014     G4int nRun = run->GetNumberOfEventToBeProcessed();
00015     G4cout << "\033[1;32m" << "Run #" << run->GetRunID() << "\033[0m" << ">" << "\033[1;32m" << nRun <<
00016         "\033[0m";
00017 }
00018
00019 void RunAction::EndOfRunAction(const G4Run* run) {
00019     AnalysisManager* analysisManager = AnalysisManager::Instance();
00020     analysisManager->RecordingRun(run);
00021 }
```

8.147 /home/ychoi/ATOM/geant4/main/src/Solid.cpp File Reference

```
#include "Solid.h"
```

Macros

- `#define __SOLID_HEADER__`

8.147.1 Macro Definition Documentation

8.147.1.1 __SOLID_HEADER__

```
#define __SOLID_HEADER__
```

Definition at line 1 of file [Solid.cpp](#).

8.148 Solid.cpp

[Go to the documentation of this file.](#)

```
00001 #define __SOLID_HEADER__
00002 #include "Solid.h"
00003
00004
00005
00006 Solid::Solid() {
00007     setAlpideCircuitSolid();
00008     setAlpideEpitaxialSolid();
00009     setBoardSolid();
00010 }
00011
00012 void Solid::setAlphaStandSolid() {
00013     G4VSolid* solidBody = new G4Box("solidBody", .5 * 180. * mm, .5 * 8. * mm, .5 * 28.5 * mm);
00014     G4VSolid* solidSourceHolder = new G4Tubs("solidSourceHolder", 0., 6.5 * mm, 3.0 * mm, 0., 360. *
00015         deg);
```



```

00015     G4VSolid* solidCollimater = new G4Tubs("solidCollimater", 0., 3.0 * mm, 8.0 * mm, 0., 360.0 *
deg);
00016
00017     G4VSolid* subtraction1 = new G4SubtractionSolid("subtraction1", solidBody, solidSourceHolder, new
G4RotationMatrix(0. * deg, 90. * deg, 90. * deg), G4ThreeVector(0.0 * mm, 4.0 * mm, 0.0 * mm));
00018     alphaStandSolid = new G4SubtractionSolid("alphaStandSolid", subtraction1, solidCollimater, new
G4RotationMatrix(0. * deg, 90. * deg, 90. * deg), G4ThreeVector(0.0 * mm, 4.0 * mm, 0.0 * mm));
00019 }
00020
00021 void Solid::setBetaStandSolid() {
00022     G4VSolid* solidBody1 = new G4Box("solidBody1", 70. * mm * 0.5, 7.0 * mm * 0.5, 70. * mm * 0.5);
00023     G4VSolid* solidBody1Hall1 = new G4Tubs("solidBody1Hall1", 0., 15. * mm * 0.5, 6. * mm * 0.5, 0.,
360.0 * deg);
00024     G4VSolid* solidBody1Hall2 = new G4Tubs("solidBody1Hall2", 0., 13. * mm * 0.5, 7. * mm * 0.5, 0.,
360.0 * deg);
00025
00026     G4VSolid* solidBody2 = new G4Tubs("solidBody2", 2. * mm * 0.5, 30. * mm * 0.5, 59. * mm * 0.5, 0.,
360.0 * deg);
00027     G4VSolid* solidBody2Hall = new G4Tubs("solidBody2Hall", 0., 14. * mm * 0.5, 10.2 * mm * 0.5, 0.,
360.0 * deg);
00028
00029     G4VSolid* subtraction = new G4SubtractionSolid("subtraction", solidBody1, solidBody1Hall1, new
G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., .5 * mm, 0.));
00030     G4VSolid* solidStand1 = new G4SubtractionSolid("solidStand1", subtraction, solidBody1Hall2, new
G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., 0. * mm, 0.));
00031     G4VSolid* solidStand2 = new G4SubtractionSolid("solidStand2", solidBody2, solidBody2Hall, new
G4RotationMatrix(0. * deg, 0. * deg, 0. * deg), G4ThreeVector(0., 0., -24.6 * mm));
00032
00033     betaStandSolid = new G4UnionSolid("betaStandSolid", solidStand1, solidStand2, new
G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., 33. * mm, 0.));
00034 }
00035
00036 void Solid::setNewStandSolid(double diameter) {
00037     G4VSolid* solidBody = new G4Box("solidBody", 15. * mm * 0.5, 6. * mm * 0.5, 17.5 * mm * 0.5);
00038     G4VSolid* solidCollimatedHall = new G4Tubs("solidCollimatedHall", 0., diameter * mm * 0.5, 1.1 *
mm * 0.5, 0., 360.0 * deg);
00039     G4VSolid* solidHolderHall = new G4Tubs("solidHolderHall", 0., 13 * mm * 0.5, 5.2 * mm * 0.5, 0.,
360.0 * deg);
00040
00041     G4VSolid* subtraction = new G4SubtractionSolid("subtraction", solidBody, solidCollimatedHall, new
G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., -2.5 * mm, 1.25 * mm));
00042     newStandSolid = new G4SubtractionSolid("newStandSolid", subtraction, solidHolderHall, new
G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., .59 * mm, 1.25 * mm));
00043 }
00044
00045 void Solid::setScreenSolid() {
00046     screenSolid = new G4Box("screenSolid", 30. * mm * 0.5, 0.108 * mm * 0.5, 30. * mm * 0.5);
00047 }
00048
00049 void Solid::setAlpideCircuitSolid() {
00050     alpideCircuitSolid = new G4Box("ALPIDEcircuitSolid", 30.0 * mm * 0.5, 11.0 * um * 0.5, 15.0 * mm *
0.5);
00051 }
00052
00053 void Solid::setAlpideEpitaxialSolid() {
00054     alpideEpitaxialSolid = new G4Box("ALPIDEepitaxialSolid", 30.0 * mm * 0.5, (50.0 - 11.0) * um *
0.5, 15.0 * mm * 0.5);
00055 }
00056
00057 void Solid::setBoardSolid() {
00058     G4Box* solidBody = new G4Box("solidBody", 70. * mm * 0.5, 1.0 * mm * 0.5, 70. * mm * 0.5);
00059     G4Box* solidEmpty = new G4Box("solidEmpty", 31. * mm * 0.5, 3 * mm, 12.8 * mm * 0.5);
00060     boardSolid = new G4SubtractionSolid("CarrierBoardSolid", solidBody, solidEmpty, G4Translate3D(0.,
0., 2.7 * mm));
00061 }
00062
00063 G4VSolid* Solid::getAlphaStandSolid() const {
00064     return alphaStandSolid;
00065 }
00066
00067 G4VSolid* Solid::getBetaStandSolid() const {
00068     return betaStandSolid;
00069 }
00070
00071 G4VSolid* Solid::getNewStandSolid() const {
00072     return newStandSolid;
00073 }
00074
00075 G4VSolid* Solid::getScreenSolid() const {
00076     return screenSolid;
00077 }
00078
00079 G4VSolid* Solid::getAlpideCircuitSolid() const {
00080     return alpideCircuitSolid;
00081 }
00082
00083 G4VSolid* Solid::getAlpideEpitaxialSolid() const {

```

```

00084     return alptideEpitaxialSolid;
00085 }
00086
00087 G4VSolid* Solid::getBoardSolid() const {
00088     return boardSolid;
00089 }

```

8.149 /home/ychoi/ATOM/geant4/main/src/SteppingAction.cpp File Reference

```
#include "SteppingAction.h"
```

Macros

- [#define __STEPPINGACTION_HEADER__](#)

8.149.1 Macro Definition Documentation

8.149.1.1 __STEPPINGACTION_HEADER__

```
#define __STEPPINGACTION_HEADER__
```

Definition at line 1 of file [SteppingAction.cpp](#).

8.150 SteppingAction.cpp

[Go to the documentation of this file.](#)

```

00001 #define __STEPPINGACTION_HEADER__
00002 #include "SteppingAction.h"
00003
00004 SteppingAction::SteppingAction(EventAction* eventAction) : G4UserSteppingAction() {
00005     G4cout << "\033[1;36m" << "Stepping Action" << "\033[0m" << " is armed" << G4endl;
00006 }
00007
00008 SteppingAction::~SteppingAction() {
00009     G4cout << "\033[1;36m" << "Stepping Action" << "\033[0m" << " is terminated" << G4endl;
00010 }
00011
00012 void SteppingAction::UserSteppingAction(const G4Step* step) {
00013     AnalysisManager* analysisManager = AnalysisManager::Instance();
00014     analysisManager->RecordingStep(step);
00015 }

```

8.151 /home/ychoi/ATOM/geant4/main/src/TrackingAction.cpp File Reference

```
#include "TrackingAction.h"
```

Macros

- `#define __TRACKINGACTION_HEADER__`

8.151.1 Macro Definition Documentation**8.151.1.1 __TRACKINGACTION_HEADER__**

```
#define __TRACKINGACTION_HEADER__
```

Definition at line 1 of file [TrackingAction.cpp](#).

8.152 TrackingAction.cpp

[Go to the documentation of this file.](#)

```
00001 #define __TRACKINGACTION_HEADER__
00002 #include "TrackingAction.h"
00003
00004 TrackingAction::TrackingAction() : G4UserTrackingAction() {
00005     G4cout << "\033[1;36m" << "Tracking Action" << "\033[0m" << " is armed" << G4endl;
00006 }
00007 TrackingAction::~TrackingAction() {
00008     G4cout << "\033[1;36m" << "Tracking Action" << "\033[0m" << " is terminated" << G4endl;
00009 }
00010
00011 void TrackingAction::PreUserTrackingAction(const G4Track* track) {
00012     AnalysisManager* analysisManager = AnalysisManager::Instance();
00013     analysisManager->RecordingTrackStart(track);
00014 }
00015
00016 void TrackingAction::PostUserTrackingAction(const G4Track* track) {
00017     AnalysisManager* analysisManager = AnalysisManager::Instance();
00018     analysisManager->RecordingTrackEnd(track);
00019 }
```

8.153 /home/ychoi/ATOM/graph_draw.cpp File Reference

```
#include "TCanvas.h"
#include "TAxis.h"
#include "TGraphErrors.h"
#include "TF1.h"
#include "TPaveText.h"
#include "TLatex.h"
#include <utility>
```

Functions

- `std::pair< double, int >` [getEffectiveValueAndDigit](#) (double value)
- `TString` [EffectiveExpression](#) (const double value, const double error)
- `void` [graph_draw](#) ()

8.153.1 Function Documentation

8.153.1.1 EffectiveExpression()

```
TString EffectiveExpression (
    const double value,
    const double error )
```

Definition at line 19 of file [graph_draw.cpp](#).

8.153.1.2 getEffectiveValueAndDigit()

```
std::pair< double, int > getEffectiveValueAndDigit (
    double value )
```

Definition at line 9 of file [graph_draw.cpp](#).

8.153.1.3 graph_draw()

```
void graph_draw ( )
```

Definition at line 31 of file [graph_draw.cpp](#).

8.154 /home/ychoi/ATOM/graph_draw.cpp

[Go to the documentation of this file.](#)

```
00001 #include "TCanvas.h"
00002 #include "TAxis.h"
00003 #include "TGraphErrors.h"
00004 #include "TF1.h"
00005 #include "TPaveText.h"
00006 #include "TLatex.h"
00007 #include<utility>
00008
00009 std::pair<double, int> getEffectiveValueAndDigit(double value) {
00010     // std::cout << round(func->GetParError(0) * pow(10., -floor(log10(func->GetParError(0)))) /
00011     pow(10., -floor(log10(func->GetParError(0)))) << std::endl;
00012     int digit = -floor(log10(value));
00013     if ( floor(value * pow(10, digit)) == 1 ) {
00014         return {round(value * pow(10, digit + 1)) / pow(10, digit + 1), digit + 1};
00015     } else {
00016         return {floor(value * pow(10, digit)) / pow(10, digit), digit};
00017     }
00018 }
00019 TString EffectiveExpression(const double value, const double error) {
00020     std::pair<double, int> ErrorAndDigit = getEffectiveValueAndDigit(error);
00021     double correctedError = ErrorAndDigit.first;
00022     int digit = ErrorAndDigit.second;
00023     double correctedValue = round(value * pow(10, digit)) / pow(10, digit);
00024     int correctedDigit = floor(log10(correctedValue));
00025     correctedValue /= pow(10, correctedDigit);
00026     correctedError /= pow(10, correctedDigit);
00027
00028     return Form("%. " + TString(std::to_string(correctedDigit + digit)) + "f #pm %.15g) #times
00029     10^{%d}", correctedValue, correctedError, correctedDigit);
00030 }
00031 void graph_draw() {
00032     // Collimator hole
00033     std::vector<std::vector<double>>values = {{1450, 1270, 1170, 1253, 1130},
00034     {19630, 19810, 19740, 19960},
00035     {93800, 88310, 89400, 90050, 88950, 87260},
00036     {196300, 198700, 204200, 197900, 202200, 193030},
```

```

00037         {666500, 708000, 717200, 712500, 741279, 746400},
00038         {2660000, 2500000, 2516000, 2536000, 2510000},
00039         {2522000, 2518000, 2530000, 2555000},
00040         {2620000, 2533000, 2550000, 2566000, 2532000, 2480000},
00041         {2510000, 2517000, 2590000, 2514000, 2571000, 2450000},
00042         {2510000, 2640000, 2680000, 2650000, 2740000, 2730000},
00043         {0.0708, 0.066, 0.0604, 0.0642, 0.0585},
00044         {1.011, 1.022, 1.013, 1.015},
00045         {4.65, 4.53, 4.55, 4.56, 4.56, 4.57},
00046         {10.16, 10.25, 10.24, 10.22, 10.21, 10.24},
00047         {34.6, 34.8, 34.8, 34.9, 35.1, 35.5}};
00048
00049     std::vector<std::vector<double>>errors = {{30, 20, 20, 18, 20},
00050         {30, 30, 40, 30},
00051         {300, 170, 60, 80, 40, 80},
00052         {200, 300, 200, 300, 400, 130},
00053         {300, 300, 400, 700, 600, 500},
00054         {40000, 20000, 13000, 14000, 30000},
00055         {8000, 11000, 10000, 14000, 30000},
00056         {40000, 7000, 8000, 10000, 8000, 40000},
00057         {20000, 14000, 30000, 7000, 12000, 40000},
00058         {70000, 40000, 50000, 30000, 30000, 40000},
00059         {0.0019, 0.0012, 0.0011, 0.0011, 0.0013},
00060         {0.007, 0.008, 0.007, 0.008},
00061         {0.08, 0.03, 0.03, 0.03, 0.03, 0.08},
00062         {0.1, 0.09, 0.13, 0.07, 0.08, 0.18},
00063         {0.9, 0.6, 0.7, 0.4, 0.4, 0.6}}};
00064
00065     std::vector<TString> titleSet = {
00066         "#phi 1 collimator; i-th photo; Pixel of collimator hole in photo",
00067         "#phi 2 collimator; i-th photo; Pixel of collimator hole in photo",
00068         "#phi 3 collimator; i-th photo; Pixel of collimator hole in photo",
00069         "#phi 4 collimator; i-th photo; Pixel of collimator hole in photo",
00070         "#phi 7 collimator; i-th photo; Pixel of collimator hole in photo",
00071         "#phi 1 collimator; i-th photo; Pixel of collimator case in photo",
00072         "#phi 2 collimator; i-th photo; Pixel of collimator case in photo",
00073         "#phi 3 collimator; i-th photo; Pixel of collimator case in photo",
00074         "#phi 4 collimator; i-th photo; Pixel of collimator case in photo",
00075         "#phi 7 collimator; i-th photo; Pixel of collimator case in photo",
00076         "#phi 1 collimator; i-th photo; Area of collimator hole",
00077         "#phi 2 collimator; i-th photo; Area of collimator hole",
00078         "#phi 3 collimator; i-th photo; Area of collimator hole",
00079         "#phi 4 collimator; i-th photo; Area of collimator hole",
00080         "#phi 7 collimator; i-th photo; Area of collimator hole"};
00081
00082     std::vector<TString> pathSet = {
00083         "build/output/hole_size/phi1_collimator_hole_pixels.pdf",
00084         "build/output/hole_size/phi2_collimator_hole_pixels.pdf",
00085         "build/output/hole_size/phi3_collimator_hole_pixels.pdf",
00086         "build/output/hole_size/phi4_collimator_hole_pixels.pdf",
00087         "build/output/hole_size/phi7_collimator_hole_pixels.pdf",
00088         "build/output/hole_size/phi1_collimator_case_pixels.pdf",
00089         "build/output/hole_size/phi2_collimator_case_pixels.pdf",
00090         "build/output/hole_size/phi3_collimator_case_pixels.pdf",
00091         "build/output/hole_size/phi4_collimator_case_pixels.pdf",
00092         "build/output/hole_size/phi7_collimator_case_pixels.pdf",
00093         "build/output/hole_size/phi1_collimator_area.pdf",
00094         "build/output/hole_size/phi2_collimator_area.pdf",
00095         "build/output/hole_size/phi3_collimator_area.pdf",
00096         "build/output/hole_size/phi4_collimator_area.pdf",
00097         "build/output/hole_size/phi7_collimator_area.pdf"
00098     };
00099
00100     for ( int iPlot = 0; iPlot < values.size(); iPlot++ ) {
00101         TCanvas* canvas = new TCanvas("can", "", 600, 600);
00102         TGraphErrors* graph = new TGraphErrors();
00103
00104         double maxValue = 0.;
00105         double minValue = 10E+12;
00106
00107         for ( int i = 0; i < values[iPlot].size(); i++ ) {
00108             maxValue = maxValue < values[iPlot][i] + errors[iPlot][i] ? values[iPlot][i] +
00109 errors[iPlot][i] : maxValue;
00110             minValue = minValue > values[iPlot][i] - errors[iPlot][i] ? values[iPlot][i] -
00111 errors[iPlot][i] : minValue;
00112             graph->SetPoint(i, i, values[iPlot][i]);
00113             graph->SetPointError(i, 0, errors[iPlot][i]);
00114         }
00115
00116         graph->SetTitle(titleSet[iPlot]);
00117         graph->SetMarkerStyle(8);
00118         graph->GetXaxis()->SetNdivisions(values[iPlot].size() + 1, 0, 0, kTRUE);
00119         graph->GetXaxis()->SetLimits(-1, values[iPlot].size());
00120         graph->SetMaximum(maxValue + (maxValue - minValue) * .5);
00121         graph->SetMinimum(minValue - (maxValue - minValue) * .1);
00122         graph->Draw("AP");
00123     }

```

```

00122         TF1* func = new TF1("func", "[0]", 0, values[iPlot].size() - 1);
00123         graph->Fit(func, "R");
00124         func->Draw("SAME");
00125
00126         TPaveText* legend = new TPaveText(.3, .7, .9, .9, "NDC");
00127         legend->AddText("Fit parameter");
00128         legend->AddText("Value: " + EffectiveExpression(func->GetParameter(0), func->GetParError(0)));
00129         legend->AddText(Form("Chi2/NDoF: %f", sqrt(func->GetChisquare() / func->GetNDF())));
00130         legend->Draw();
00131
00132         canvas->SetLeftMargin(0.15);
00133         canvas->SaveAs(pathSet[iPlot]);
00134
00135         delete func;
00136         func = nullptr;
00137         delete graph;
00138         graph = nullptr;
00139         delete canvas;
00140         canvas = nullptr;
00141     }
00142
00143     TGraphErrors* graph2 = new TGraphErrors();
00144     graph2->SetPoint(0, 0.0631, 0.0001415);
00145     graph2->SetPointError(0, 0.0005, 0.0000006);
00146     graph2->SetPoint(1, 1.015, 0.00708);
00147     graph2->SetPointError(1, 0.003, 0.00003);
00148     graph2->SetPoint(2, 4.554, 0.0525);
00149     graph2->SetPointError(2, 0.014, 0.0003);
00150     graph2->SetPoint(3, 10.22, 0.1404);
00151     graph2->SetPointError(3, 0.03, 0.0005);
00152     graph2->SetPoint(4, 35.0, 0.3343);
00153     graph2->SetPointError(4, 0.2, 0.0005);
00154
00155     TCanvas* canvas2 = new TCanvas("can2", "", 600, 600);
00156     graph2->SetTitle("Experiment vs Simulation; Collimator hole width [mm^{2}]; Experiment /
Simulation");
00157     graph2->SetMarkerStyle(8);
00158     graph2->SetMarkerSize(.1);
00159     graph2->SetMinimum(0);
00160     graph2->SetMaximum(.4);
00161     graph2->Draw("AP");
00162     canvas2->SetLeftMargin(.12);
00163     // canvas2->SetLogx();
00164     // canvas2->SetLogy();
00165     canvas2->SaveAs("build/output/hole_size/compare_exp_simul.pdf");
00166 }

```

8.155 /home/ychoi/ATOM/pycpp/config/inc/CppConfigDictionary.h File Reference

```

#include <iostream>
#include <string>
#include <sstream>
#include <unordered_map>
#include <vector>
#include "CppConfigError.h"

```

Classes

- class [CppConfigDictionary](#)

8.156 CppConfigDictionary.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __CPPCONFIGDICTIONARY__
00002 #define __CPPCONFIGDICTIONARY__

```

```

00003
00004 #include <iostream>
00005 #include <string>
00006 #include <sstream>
00007 #include <unordered_map>
00008 #include <vector>
00009
00010 #include "CppConfigError.h"
00011
00012 class CppConfigDictionary {
00013     std::string mConfigName;
00014     std::unordered_map<std::string, std::string> mDictionary;
00015     std::vector<CppConfigDictionary> mSubConfigDictionary;
00016 public:
00017     CppConfigDictionary();
00018     CppConfigDictionary(const CppConfigDictionary& copy);
00019     CppConfigDictionary& operator=(const CppConfigDictionary& copy);
00020     CppConfigDictionary(CppConfigDictionary&& move);
00021     CppConfigDictionary& operator=(CppConfigDictionary&& move);
00022
00023     CppConfigDictionary(std::string_view configName);
00024     void addDictionary(std::string_view key, std::string_view value);
00025     void addSubConfigDictionary(const CppConfigDictionary& subConfigDictionary);
00026
00027     const bool hasKey(std::string_view key) const;
00028     const std::string& find(const std::string& key) const;
00029     const CppConfigDictionary& getSubConfig(std::string_view key) const;
00030     const std::vector<CppConfigDictionary> getSubConfigSet() const;
00031     const std::vector<std::string> getValueList() const;
00032     const std::vector<std::string> getKeyList() const;
00033     std::unordered_map<std::string, std::string> getDictionary();
00034     const std::unordered_map<std::string, CppConfigDictionary> getSubConfigSetWithName() const;
00035
00036     std::string_view getConfigName() const;
00037
00038     friend std::ostream& operator<<(std::ostream& os, const CppConfigDictionary& copy);
00039     CppConfigDictionary& operator+(const CppConfigDictionary& copy);
00040     CppConfigDictionary& operator+=(const CppConfigDictionary& copy);
00041 };
00042
00043
00044 #endif

```

8.157 /home/ychoi/ATOM/pycpp/config/inc/CppConfigError.h File Reference

```

#include <exception>
#include <string>

```

Classes

- class [CppConfigFileError](#)
- class [ConfigurableNoValue](#)

8.158 CppConfigError.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __CPPCONFIGERROR__
00002 #define __CPPCONFIGERROR__
00003
00004 #include<exception>
00005 #include<string>
00006
00007 class CppConfigFileError : public std::exception {
00008 public:
00009     std::string mMessage;
00010 public:

```

```

00011     CppConfigFileError(std::string_view errorType, std::string_view parameter) {
00012         if ( errorType == "FileExist" ) {
00013             mMessage = "The Config File \033[1;32m" + std::string(parameter) + "\033[1;0m doesn't
exist ";
00014         } else if ( errorType == "ConfigDictionaryExist" ) {
00015             mMessage = "The Config Dictionary \033[1;32m" + std::string(parameter) + "\033[1;0m
doesn't exist";
00016         }
00017     }
00018     const char* what() const throw() {
00019         return mMessage.c_str();
00020     }
00021 };
00022
00023
00024 class ConfigurableNoValue : public std::exception {
00025 public:
00026     std::string message;
00027 public:
00028     ConfigurableNoValue(std::string_view key, std::string_view configName) {
00029         message = static_cast<std::string>(key) + " cannot be found in list of values of " +
static_cast<std::string>(configName);
00030     }
00031     const char* what() const throw() {
00032         return message.c_str();
00033     }
00034 };
00035
00036 #endif

```

8.159 /home/ychoi/ATOM/pycpp/config/inc/CppConfigFile.h File Reference

```

#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <unordered_map>
#include <fstream>
#include <filesystem>
#include <algorithm>
#include "CppConfigDictionary.h"
#include "CppConfigError.h"

```

Classes

- class [CppConfigFile](#)

8.160 CppConfigFile.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __CPPCONFIG__
00002 #define __CPPCONFIG__
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <sstream>
00007 #include <vector>
00008 #include <unordered_map>
00009 #include <fstream>
00010 #include <filesystem>
00011 #include <algorithm>
00012 #include <fstream>
00013

```



```

00014 #include "CppConfigDictionary.h"
00015 #include "CppConfigError.h"
00016
00017 class CppConfigFile {
00018 private:
00019     std::vector<CppConfigDictionary> mConfigs;
00020 public:
00021     CppConfigFile();
00022     CppConfigFile(std::string_view configFile);
00023     CppConfigFile(const CppConfigFile& copy);
00024     CppConfigFile& operator=(const CppConfigFile& copy);
00025     CppConfigFile(CppConfigFile&& move);
00026     CppConfigFile& operator=(CppConfigFile&& move);
00027     ~CppConfigFile();
00028     void addConfig(std::string_view configFile);
00029     CppConfigDictionary getConfigFromArray(std::string_view key, const std::vector<std::string>&
valueArray);
00030     void addConfig(std::string_view configTitle, const std::vector<std::string>& configArray);
00031     const std::vector<std::string> getConfigurableNameList() const;
00032     const CppConfigDictionary getConfig(std::string_view configTitle) const;
00033     const bool hasConfig(std::string_view configTitle) const;
00034     friend std::ostream& operator<<(std::ostream& os, const CppConfigFile& copy);
00035 };
00036
00037
00038
00039 #endif

```

8.161 /home/ychoi/ATOM/pycpp/config/src/CppConfigDictionary.cpp File Reference

```
#include "CppConfigDictionary.h"
```

Functions

- std::ostream & [operator<<](#) (std::ostream &os, const [CppConfigDictionary](#) ©)

8.161.1 Function Documentation

8.161.1.1 [operator<<\(\)](#)

```
std::ostream & operator<< (
    std::ostream & os,
    const CppConfigDictionary & copy )
```

Definition at line 73 of file [CppConfigDictionary.cpp](#).

8.162 CppConfigDictionary.cpp

[Go to the documentation of this file.](#)

```

00001 #include "CppConfigDictionary.h"
00002
00003 CppConfigDictionary::CppConfigDictionary() = default;
00004
00005 CppConfigDictionary::CppConfigDictionary(std::string_view configName) : mConfigName(configName) { }
00006
00007 CppConfigDictionary::CppConfigDictionary(const CppConfigDictionary& copy) :
    mConfigName(copy.mConfigName), mDictionary(copy.mDictionary),
    mSubConfigDictionary(copy.mSubConfigDictionary) {
00008

```

```

00009 }
00010 CppConfigDictionary& CppConfigDictionary::operator=(const CppConfigDictionary& copy) {
00011     mConfigName = copy.mConfigName;
00012     mDictionary = copy.mDictionary;
00013     mSubConfigDictionary = copy.mSubConfigDictionary;
00014
00015     return *this;
00016 }
00017 CppConfigDictionary::CppConfigDictionary(CppConfigDictionary&& move) : mConfigName(move.mConfigName),
    mDictionary(move.mDictionary), mSubConfigDictionary(move.mSubConfigDictionary) {
00018
00019 }
00020 CppConfigDictionary& CppConfigDictionary::operator=(CppConfigDictionary&& move) {
00021     mConfigName = move.mConfigName;
00022     mDictionary = move.mDictionary;
00023     mSubConfigDictionary = move.mSubConfigDictionary;
00024
00025     mDictionary.clear();
00026     mSubConfigDictionary.clear();
00027
00028     return *this;
00029 }
00030
00031
00032 void CppConfigDictionary::addDictionary(std::string_view key, std::string_view value) {
00033     mDictionary.insert_or_assign(std::string(key), std::string(value));
00034 }
00035
00036 void CppConfigDictionary::addSubConfigDictionary(const CppConfigDictionary& subConfigDictionary) {
00037     mSubConfigDictionary.push_back(subConfigDictionary);
00038 }
00039
00040 const bool CppConfigDictionary::hasKey(std::string_view key) const {
00041     for ( const auto& dict : mSubConfigDictionary ) {
00042         if ( dict.getConfigName() == key ) {
00043             return true;
00044         }
00045     }
00046     if ( mDictionary.count(std::string(key)) ) {
00047         return true;
00048     } else {
00049         return false;
00050     }
00051 }
00052
00053 const std::string& CppConfigDictionary::find(const std::string& key) const {
00054     if ( key.find('/') == std::string::npos ) {
00055         if ( mDictionary.count(key) == 1 ) {
00056             return mDictionary.find(key)->second;
00057         } else {
00058             return "";
00059         }
00060     } else {
00061         if ( hasKey(key.substr(0, key.find('/')).substr(0, key.find('/'))) ) {
00062             return getSubConfig(key.substr(0, key.find('/')).find(key.substr(key.find('/') + 1)));
00063         } else {
00064             return "";
00065         }
00066     }
00067 }
00068
00069 std::string_view CppConfigDictionary::getConfigName() const {
00070     return mConfigName;
00071 }
00072
00073 std::ostream& operator<<(std::ostream& os, const CppConfigDictionary& copy) {
00074     if ( copy.getConfigName().length() < 8 ) {
00075         os << copy.getConfigName() << "\t\t";
00076     } else if ( copy.getConfigName().length() < 16 ) {
00077         os << copy.getConfigName() << "\t";
00078     } else {
00079         os << copy.getConfigName() << std::endl << "\t\t";
00080     }
00081     for ( auto& dict : copy.mDictionary ) {
00082         os << "\033[1;47m" << dict.first << "\033[1;0m" << ", ";
00083     }
00084     for ( auto& subConfig : copy.mSubConfigDictionary ) {
00085         os << "\033[1;42m" << subConfig.getConfigName() << "\033[1;0m" << ", ";
00086     }
00087     os << std::endl;
00088     return os;
00089 }
00090
00091 const std::vector<std::string> CppConfigDictionary::getKeyList() const {
00092     std::vector<std::string> returnArray;
00093
00094     for ( const auto& pair : mDictionary ) {

```

```

00095         returnArray.push_back(pair.first);
00096     }
00097
00098     for ( const CppConfigDictionary& subConfigDictionary : mSubConfigDictionary ) {
00099         for ( std::string_view subConfigKey : subConfigDictionary.getKeyList() ) {
00100             returnArray.push_back(std::string(subConfigDictionary.getConfigName()) + "/" +
std::string(subConfigKey));
00101         }
00102     }
00103     return returnArray;
00104 }
00105
00106 std::unordered_map<std::string, std::string> CppConfigDictionary::getDictionary() {
00107     return mDictionary;
00108 }
00109
00110 const CppConfigDictionary& CppConfigDictionary::getSubConfig(std::string_view key) const {
00111     try {
00112         bool isExist = false;
00113         for ( const CppConfigDictionary& subConfigDictionary : mSubConfigDictionary ) {
00114             if ( subConfigDictionary.getConfigName() == key ) {
00115                 isExist = true;
00116                 return subConfigDictionary;
00117             }
00118         }
00119         if ( !isExist ) { throw CppConfigFileError("ConfigDictionaryExist", key); }
00120     } catch ( CppConfigFileError error ) {
00121         std::cerr << error.what() << std::endl;
00122     }
00123 }
00124
00125 const std::vector<std::string> CppConfigDictionary::getValueList() const {
00126     std::vector<std::string> returnArray;
00127
00128     for ( const auto& pair : mDictionary ) {
00129         returnArray.push_back(pair.second);
00130     }
00131
00132     return returnArray;
00133 }
00134
00135 CppConfigDictionary& CppConfigDictionary::operator+(const CppConfigDictionary& copy) {
00136     for ( const auto& pair : copy.mDictionary ) {
00137         mDictionary.insert_or_assign(pair.first, pair.second);
00138     }
00139     for ( const CppConfigDictionary& solo : copy.mSubConfigDictionary ) {
00140         mSubConfigDictionary.push_back(solo);
00141     }
00142
00143     return *this;
00144 }
00145
00146 CppConfigDictionary& CppConfigDictionary::operator+=(const CppConfigDictionary& copy) {
00147     for ( const auto& pair : copy.mDictionary ) {
00148         mDictionary.insert_or_assign(pair.first, pair.second);
00149     }
00150     for ( const CppConfigDictionary& solo : copy.mSubConfigDictionary ) {
00151         mSubConfigDictionary.push_back(solo);
00152     }
00153
00154     return *this;
00155 }
00156
00157 const std::vector<CppConfigDictionary> CppConfigDictionary::getSubConfigSet() const {
00158     return mSubConfigDictionary;
00159 }
00160
00161 const std::unordered_map<std::string, CppConfigDictionary>
CppConfigDictionary::getSubConfigSetWithName() const {
00162     std::unordered_map<std::string, CppConfigDictionary> returnMap;
00163     for ( const CppConfigDictionary& configDictionary : mSubConfigDictionary ) {
00164         returnMap.insert_or_assign(std::string(configDictionary.getConfigName()), configDictionary);
00165     }
00166     return returnMap;
00167 }

```

8.163 /home/ychoi/ATOM/pycpp/config/src/CppConfigError.cpp File Reference

```
#include "CppConfigError.h"
```

8.164 CppConfigError.cpp

[Go to the documentation of this file.](#)

```
00001 #include "CppConfigError.h"
```

8.165 /home/ychoi/ATOM/pycpp/config/src/CppConfigFile.cpp File Reference

```
#include "CppConfigFile.h"
```

Functions

- `std::ostream & operator<< (std::ostream &os, const CppConfigFile ©)`

8.165.1 Function Documentation

8.165.1.1 operator<<()

```
std::ostream & operator<< (
    std::ostream & os,
    const CppConfigFile & copy )
```

Definition at line 232 of file [CppConfigFile.cpp](#).

8.166 CppConfigFile.cpp

[Go to the documentation of this file.](#)

```
00001 #include "CppConfigFile.h"
00002
00007 CppConfigFile::CppConfigFile() = default;
00008
00015 CppConfigFile::CppConfigFile(std::string_view configFile) {
00016     // Find config file. If the file doesn't exist, then it omits error message.
00017     try {
00018         if ( !std::filesystem::exists(configFile) ) { throw CppConfigFileError("FileExist",
00019             configFile); }
00019         addConfig(configFile);
00020     } catch ( CppConfigFileError error ) {
00021         error.what();
00022         exit(1);
00023     }
00024 }
00025
00026 CppConfigFile::CppConfigFile(const CppConfigFile& copy) {
00027     for ( const CppConfigDictionary& dict : copy.mConfigs ) {
00028         mConfigs.push_back(dict);
00029     }
00030 }
00031 CppConfigFile& CppConfigFile::operator=(const CppConfigFile& copy) {
00032     for ( const CppConfigDictionary& dict : copy.mConfigs ) {
00033         mConfigs.push_back(dict);
00034     }
00035     return *this;
00036 }
00037 CppConfigFile::CppConfigFile(CppConfigFile&& copy) {
00038     for ( const CppConfigDictionary& dict : copy.mConfigs ) {
00039         mConfigs.push_back(dict);
00040     }
```

```

00041 }
00042 CppConfigFile& CppConfigFile::operator=(CppConfigFile&& copy) {
00043     for ( const CppConfigDictionary& dict : copy.mConfigs ) {
00044         mConfigs.push_back(dict);
00045     }
00046     return *this;
00047 }
00048
00054 void CppConfigFile::addConfig(std::string_view configFile) {
00055     // Open config file
00056     std::ifstream conf{std::string(configFile)};
00057
00058     // The reading line
00059     std::string line;
00060     // The key
00061     std::string key;
00062     // The value is stored as std::string form
00063     std::string value;
00064
00065     // Read config file line by line
00066     std::vector<std::string> valueArray;
00067     bool isFirst = true;
00068     while ( getline(conf, line) ) {
00069         // Remove comment out part
00070         std::string result;
00071         bool escapeNext = false;
00072
00073         for ( char c : line ) {
00074             if ( escapeNext ) {
00075                 if ( c == '#' ) {
00076                     result += '#';
00077                 } else {
00078                     result += '\\';
00079                     result += c;
00080                 }
00081                 escapeNext = false;
00082             } else {
00083                 if ( c == '\\' ) {
00084                     escapeNext = true;
00085                 } else {
00086                     result += c;
00087                 }
00088             }
00089         }
00090         if ( escapeNext ) {
00091             result += '\\'; // Add trailing backslash if input ends with a backslash
00092         }
00093         line = result;
00094         line.erase(remove(line.begin(), line.end(), '\\t'), line.end());
00095         if ( line[0] == '#' ) {
00096             line = "";
00097         }
00098         if ( line.find('=') != std::string::npos ) {
00099             line.erase(remove(line.begin(), line.begin() + line.find('='), ' '), line.begin() +
line.find('='));
00100         } else {
00101             line.erase(remove(line.begin(), line.end(), ' '), line.end());
00102         }
00103         if ( line == "" ) { // Pass the blank line
00104             continue;
00105         } else if ( line.find('[') != std::string::npos && (line[line.find('[') - 1] != '\\') ) { //
The line started with '[' is considered as Configurable name
00106             if ( !isFirst ) {
00107                 mConfigs.push_back(getConfigFromArray(key, valueArray));
00108                 valueArray.clear();
00109             } else {
00110                 isFirst = false;
00111             }
00112             key = line.substr(line.find('[') + 1, line.find(']') - 1);
00113             // Make new configurable
00114         } else {
00115             valueArray.push_back(line);
00116         }
00117     }
00118     if ( valueArray.size() != 0 ) {
00119         mConfigs.push_back(getConfigFromArray(key, valueArray));
00120     }
00121 }
00122
00123 void CppConfigFile::addConfig(std::string_view configTitle, const std::vector<std::string>&
configArray) {
00124     mConfigs.push_back(getConfigFromArray(configTitle, configArray));
00125 }
00126
00127
00128 CppConfigDictionary CppConfigFile::getConfigFromArray(std::string_view motherKey, const
std::vector<std::string>& valueArray) {

```

```

00129     std::string key, value;
00130     CppConfigDictionary returnConfigDictionary(motherKey);
00131     int numBra = 0;
00132     for ( int lineNum = 0; lineNum < valueArray.size(); lineNum++ ) {
00133         std::string line = valueArray[lineNum];
00134         if ( (line.find('=') != std::string::npos && line.find('{') == std::string::npos) && numBra ==
0 ) {
00135             line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() +
line.find('=') + 2);
00136             line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() +
line.find('=') + 2);
00137
00138             key = line.substr(0, line.find('='));
00139             value = line.substr(line.find('=') + 1);
00140             while ( true ) {
00141                 if ( value[0] == '\t' || value[0] == ' ' ) {
00142                     value.erase(value.begin());
00143                 } else {
00144                     break;
00145                 }
00146             }
00147             returnConfigDictionary.addDictionary(key, value);
00148             key = "";
00149             value = "";
00150         } else if ( (line.find('=') != std::string::npos) && (line.find('{') != std::string::npos) &&
(line[line.find('{') - 1] != '\\') ) {
00151             line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() +
line.find('=') + 2);
00152             line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() +
line.find('=') + 2);
00153             key = line.substr(0, line.find('='));
00154             std::vector<std::string> tempValueArray;
00155             int numSubBra = 0;
00156             lineNum++;
00157             while ( true ) {
00158                 line = valueArray[lineNum];
00159                 if ( (line.find('{') != std::string::npos) && (line[line.find('{') - 1] != '\\') ) {
00160                     line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'),
line.begin() + line.find('=') + 2);
00161                     line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '),
line.begin() + line.find('=') + 2);
00162                     tempValueArray.push_back(line);
00163                     numSubBra++;
00164                 } else if ( (line.find('{') != std::string::npos) && (line[line.find('{') - 1] !=
'\\') ) {
00165                     if ( numSubBra != 0 ) {
00166                         line.erase(remove(line.begin(), line.end(), '\t'), line.end());
00167                         line.erase(remove(line.begin(), line.end(), ' '), line.end());
00168                         tempValueArray.push_back(line);
00169                     } else {
00170                         returnConfigDictionary.addSubConfigDictionary(getConfigFromArray(key,
tempValueArray));
00171                         break;
00172                     }
00173                     numSubBra--;
00174                 } else {
00175                     line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'),
line.begin() + line.find('=') + 2);
00176                     line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '),
line.begin() + line.find('=') + 2);
00177                     tempValueArray.push_back(line);
00178                 }
00179                 lineNum++;
00180             }
00181         } else if ( (line.find('=') != std::string::npos) && (line.find('{') != std::string::npos) &&
(line[line.find('{') - 1] == '\\') ) {
00182             line.erase(remove(line.begin(), line.end(), '\\'), line.end());
00183             line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() +
line.find('=') + 2);
00184             line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() +
line.find('=') + 2);
00185             key = line.substr(0, line.find('='));
00186             value = line.substr(line.find('=') + 1);
00187             while ( true ) {
00188                 if ( value[0] == '\t' || value[0] == ' ' ) {
00189                     value.erase(value.begin());
00190                 } else {
00191                     break;
00192                 }
00193             }
00194             returnConfigDictionary.addDictionary(key, value);
00195             key = "";
00196             value = "";
00197         }
00198     }
00199     return returnConfigDictionary;
00200 }

```

```

00201
00202 const std::vector<std::string> CppConfigFile::getConfigurableNameList() const {
00203     std::vector<std::string> configurableNameList;
00204     for ( const CppConfigDictionary& config : mConfigs ) {
00205         configurableNameList.push_back(std::string(config.getConfigName()));
00206     }
00207     return configurableNameList;
00208 }
00209
00210
00211 const CppConfigDictionary CppConfigFile::getConfig(std::string_view configTitle) const {
00212     CppConfigDictionary rConfig;
00213     for ( auto& config : mConfigs ) {
00214         if ( config.getConfigName() == configTitle ) {
00215             rConfig = config;
00216             break;
00217         }
00218     }
00219     return rConfig;
00220 }
00221
00222 const bool CppConfigFile::hasConfig(std::string_view configTitle) const {
00223     bool hasValue = false;
00224     for ( auto& config : mConfigs ) {
00225         if ( config.getConfigName() == configTitle ) {
00226             hasValue = true;
00227         }
00228     }
00229     return hasValue;
00230 }
00231
00232 std::ostream& operator<<(std::ostream& os, const CppConfigFile& copy) {
00233     std::cout << "Config list (white: keys, green: subConfig)" << std::endl;
00234     for ( auto& config : copy.mConfigs ) {
00235         os << config;
00236     }
00237     return os;
00238 }
00239
00240 CppConfigFile::~CppConfigFile() { }

```

8.167 /home/ychoi/ATOM/pycpp/inc/cppargs.h File Reference

```

#include <iostream>
#include <algorithm>
#include <vector>
#include <cstdint>
#include <typeinfo>
#include <unordered_map>
#include <sstream>

```

Classes

- class [HelpMessage](#)
- class [Argument](#)
- class [ArgumentParser](#)

Enumerations

- enum [ActionType](#) {
ACTION_NONE , ACTION_STORE , ACTION_STORE_CONST , ACTION_STORE_TRUE ,
ACTION_STORE_FALSE , ACTION_APPEND , ACTION_APPEND_CONST , ACTION_COUNT ,
ACTION_HELP , ACTION_VERSION , ACTION_PARSERS , ACTION_EXTEND }
- enum [ARGTYPES](#) {
INT , DOUBLE , BOOL , STRING ,
NONE }

8.167.1 Enumeration Type Documentation

8.167.1.1 ActionType

enum [ActionType](#)

Enumerator

ACTION_NONE	
ACTION_STORE	
ACTION_STORE_CONST	
ACTION_STORE_TRUE	
ACTION_STORE_FALSE	
ACTION_APPEND	
ACTION_APPEND_CONST	
ACTION_COUNT	
ACTION_HELP	
ACTION_VERSION	
ACTION_PARSERS	
ACTION_EXTEND	

Definition at line 12 of file [cppargs.h](#).

8.167.1.2 ARGTYPES

enum [ARGTYPES](#)

Enumerator

INT	
DOUBLE	
BOOL	
STRING	
NONE	

Definition at line 37 of file [cppargs.h](#).

8.168 cppargs.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __CPPARGS_H__
00002 #define __CPPARGS_H__
00003
00004 #include <iostream>
00005 #include <algorithm>
00006 #include <vector>
00007 #include <cstdarg>
00008 #include <typeinfo>
00009 #include <unordered_map>
00010 #include <sstream>
00011
00012 enum ActionType {
```



```

00013     ACTION_NONE,
00014     ACTION_STORE,
00015     ACTION_STORE_CONST,
00016     ACTION_STORE_TRUE,
00017     ACTION_STORE_FALSE,
00018     ACTION_APPEND,
00019     ACTION_APPEND_CONST,
00020     ACTION_COUNT,
00021     ACTION_HELP,
00022     ACTION_VERSION,
00023     ACTION_PARSERS,
00024     ACTION_EXTEND
00025 };
00026
00027 class Argument;
00028
00029 class HelpMessage {
00030 private:
00031     std::string usage;
00032 public:
00033     HelpMessage(std::string prog, std::string description);
00034     void print(std::vector<Argument>& Pos_args, std::vector<Argument>& Opt_args);
00035 };
00036
00037 enum ARGYPES {
00038     INT,
00039     DOUBLE,
00040     BOOL,
00041     STRING,
00042     NONE
00043 };
00044
00045 class Argument {
00046 private:
00047     ARGYPES _argType = ARGYPES::NONE;
00048     std::vector<std::string> _argValueList;
00049     std::vector<std::string> _argDomain;
00050
00051     std::string _argName = "";
00052     std::string _argOpt = "";
00053     std::vector<std::string> _argMinorOpt;
00054
00055     std::string _argDenoteOut = "";
00056     std::string _argDenoteIn = "";
00057
00058     std::string _description = "";
00059
00060     bool _isArgMulti = false;
00061     bool _isArgConst = false;
00062 public:
00063     Argument(std::string str);
00064
00065     // Setter
00066     void setArgType(ARGYPES typ);
00067
00068     void setArgValue(std::string str);
00069     void setArgValueList(std::vector<std::string> strList);
00070     void replaceArgValueList(std::vector<std::string> strList);
00071     void setArgDomain(std::string str);
00072     void setArgDomain(std::vector<std::string> strList);
00073
00074     void setArgName(std::string str);
00075     void setArgOpt(std::string str);
00076     void setArgMinorOpt(std::string str);
00077     void setArgMinorOpt(std::vector<std::string> strList);
00078
00079     void setArgDenoteOut(std::string str);
00080     void setArgDenoteIn(std::string str);
00081
00082     void setArgDescription(std::string description);
00083
00084     void isArgMulti();
00085     void notArgMulti();
00086     void isArgConst();
00087     void notArgConst();
00088
00089     // Getter
00090     ARGYPES getArgType() const;
00091     std::string getArgValue(const int order) const;
00092     std::vector<std::string> getArgValueList() const;
00093     std::string getArgDomain(const int order) const;
00094     std::vector<std::string> getArgDomain() const;
00095
00096     std::string getArgName() const;
00097     std::string getArgOpt() const;
00098     std::vector<std::string> getArgMinorOpt() const;

```

```

00100
00101     std::string getArgDenoteOut() const;
00102     std::string getArgDenoteIn() const;
00103
00104     std::string getArgDescription() const;
00105
00106     bool getArgMulti() const;
00107     bool getArgConst() const;
00108 };
00109
00110 class ArgumentParser {
00111 private:
00112     std::vector<std::string> argv;
00113     std::string prog = "";
00114     std::string description = "";
00115     std::vector<Argument> Pos_args;
00116     std::vector<Argument> Opt_args;
00117     Argument* args_temp;
00118     std::unordered_map<std::string, std::vector<std::string> argv_init;
00119     bool needHelp = false;
00120 public:
00121     ArgumentParser(int _argc, char** _argv);
00122     ~ArgumentParser();
00123
00124     ArgumentParser& setDescription(std::string _description) { description = _description; return
    *this; }
00125     ArgumentParser& add_argument(const std::string& opts);
00126     ArgumentParser& add_minor_argument(const std::string& opts);
00127     ArgumentParser& add_domain(const std::vector<std::string>& opts);
00128     ArgumentParser& dest(const std::string& str);
00129     ArgumentParser& metavar(const std::string& str);
00130     ArgumentParser& set_const();
00131     ArgumentParser& nargs();
00132     ArgumentParser& type(std::string typeOpt); // You can set the type in int, double, bool, string.
00133     ArgumentParser& help(std::string message);
00134     ArgumentParser& set_default(std::string value);
00135
00136     ARGTYPES detType(const std::string& str);
00137     void add_finish();
00138     void parse_args();
00139     template<typename T>
00140     T get_value(const std::string& valueName);
00141 };
00142
00143 #endif

```

8.169 /home/ychoi/ATOM/pycpp/inc/cppTimer.h File Reference

```

#include <iostream>
#include <ctime>

```

Classes

- class [TTimer](#)

8.170 cppTimer.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __CPPTIMER__
00002 #define __CPPTIMER__
00003
00004 #include <iostream>
00005 #include <ctime>
00006
00007 class TTimer {
00008 private:
00009     clock_t start, finish;
00010     double duration;
00011 public:
00012     TTimer();
00013     void Measure();
00014     void EndProgram();
00015 };
00016
00017 #endif

```

8.171 /home/ychoi/ATOM/pycpp/inc/cpptqdm.h File Reference

```
#include <chrono>
#include <iostream>
#include <sys/ioctl.h>
#include <unistd.h>
#include <vector>
#include <iomanip>
#include <math.h>
```

Classes

- class [ProgressBar](#)

8.172 cpptqdm.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __CPPTQDM__
00002 #define __CPPTQDM__
00003
00004 #include <chrono>
00005 #include <iostream>
00006 #include <sys/ioctl.h>
00007 #include <unistd.h>
00008 #include <vector>
00009 #include <iomanip>
00010 #include <math.h>
00011
00012 class ProgressBar {
00013 private:
00014     std::chrono::system_clock::time_point start_time;
00015     int mTerminalWidth;
00016     int mSetSize;
00017     std::chrono::system_clock::time_point printPoint;
00018     int called = 0;
00019
00020
00021 public:
00022     ProgressBar();
00023     ProgressBar(int setSize);
00024     ~ProgressBar();
00025
00026     void getTerminalLength();
00027
00028     void printProgress();
00029     int getSecond(int num);
00030     int getMinute(int num);
00031 };
00032
00033
00034 #endif
```

8.173 /home/ychoi/ATOM/pycpp/inc/cppUnit.h File Reference

```
#include <string>
#include <iostream>
#include <array>
#include <vector>
#include <cmath>
#include <unordered_map>
#include <algorithm>
#include <fstream>
#include <sstream>
```

Classes

- class [Unit](#)
- class [Quantity](#)

Variables

- `const std::unordered_map< char, int > prefix = {{'a', -18}, {'f', -15}, {'p', -12}, {'n', -9}, {'u', -6}, {'m', -3}, {'c', -2}, {'k', 3}, {'M', 6}, {'G', 9}, {'T', 12}, {'P', 15}}`

8.173.1 Variable Documentation

8.173.1.1 [prefix](#)

```
const std::unordered_map<char, int> prefix = {{'a', -18}, {'f', -15}, {'p', -12}, {'n', -9},
{'u', -6}, {'m', -3}, {'c', -2}, {'k', 3}, {'M', 6}, {'G', 9}, {'T', 12}, {'P', 15}}
```

Definition at line 14 of file [cppUnit.h](#).

8.174 [cppUnit.h](#)

[Go to the documentation of this file.](#)

```
00001 #ifndef __CPPUNIT__
00002 #define __CPPUNIT__
00003
00004 #include <string>
00005 #include <iostream>
00006 #include <array>
00007 #include <vector>
00008 #include <cmath>
00009 #include <unordered_map>
00010 #include <algorithm>
00011 #include <fstream>
00012 #include <sstream>
00013
00014 const std::unordered_map<char, int> prefix = {{'a', -18}, {'f', -15}, {'p', -12}, {'n', -9}, {'u',
-6}, {'m', -3}, {'c', -2}, {'k', 3}, {'M', 6}, {'G', 9}, {'T', 12}, {'P', 15}};
00015
00016 class Unit {
00017 private:
00018     int mDigit;
00019     std::array<int, 7> unitCount = {0, 0, 0, 0, 0, 0, 0}; // 0: meter, 1: kilogram, 2: sec, 3: ampare,
4: kelvin, 5: mol, 6: candela
00020     std::vector<std::string> nonBasicUnit;
00021     static std::vector<std::pair<std::string, std::array<int, 7>>> userUnits;
00022     static std::vector<std::string> exceptPrefixList;
00023     static bool isUserUnit;
00024 public:
00025     Unit();
00026     Unit(std::string_view unit);
00027
00028     static void setUserUnit(std::string_view newUnit, std::string_view newSiUnit);
00029     void setUnit(std::string_view unit);
00030
00031     bool operator==(const Unit& ref) const;
00032     bool operator!=(const Unit& ref) const;
00033     Unit operator*(const Unit& ref) const;
00034     Unit operator*=(const Unit& ref);
00035     Unit operator/(const Unit& ref) const;
00036     Unit operator/=(const Unit& ref);
00037     friend std::ostream& operator<<(std::ostream& os, Unit& ref);
00038     friend std::ostream& operator<<(std::ostream& os, const Unit& ref);
00039
00040     const int getDigit() const;
00041     const std::array<int, 7>& getUnitCount() const;
00042     const std::vector<std::string> getNonBasicUnits() const;
00043     const std::string getUnit() const;
```

```

00044 private:
00045     bool seperate(std::vector<std::string>& store, std::string_view unit);
00046     bool vanishSlash(std::vector<std::string>& store);
00047     bool removePrefix(std::vector<std::string>& store);
00048     void setUnitCount(std::string unit, int power);
00049     void plusCount(std::array<int, 7> nums);
00050 };
00051
00052 class Quantity {
00053 private:
00054     double mNum;
00055     int mDigit;
00056     Unit mUnit;
00057     static std::vector<std::tuple<std::string, std::string, Unit>> userQuantity;
00058 public:
00059     static void setUserQuantity(); // There are user unit txt file should be located in same
    directory.
00060     Quantity() = delete;
00061     Quantity(std::string quantity);
00062     Quantity(double num, std::string unit);
00063
00064     double getNum() const;
00065     double getNum(std::string_view unit) const;
00066     const std::string getUnit() const;
00067     const std::string getQuantity() const;
00068     const std::string getQuantity(std::string_view unit) const;
00069
00070     Quantity operator+(const Quantity& ref) const;
00071     Quantity operator-(const Quantity& ref) const;
00072     Quantity operator*(const Quantity& ref) const;
00073     Quantity operator/(const Quantity& ref) const;
00074     Quantity operator+=(const Quantity& ref);
00075     Quantity operator-=(const Quantity& ref);
00076     Quantity operator*=(const Quantity& ref);
00077     Quantity operator/=(const Quantity& ref);
00078
00079     friend std::ostream& operator<<(std::ostream& os, Quantity& ref);
00080     friend std::ostream& operator<<(std::ostream& os, const Quantity& ref);
00081 };
00082
00083 #endif

```

8.175 /home/ychoi/ATOM/pycpp/src/cppargs.cpp File Reference

```
#include "cppargs.h"
```

Functions

- `template<> std::string ArgumentParser::get_value< std::string > (const std::string &valueName)`
- `template<> std::vector< int > ArgumentParser::get_value< std::vector< int > > (const std::string &valueName)`
- `template<> std::vector< double > ArgumentParser::get_value< std::vector< double > > (const std::string &valueName)`
- `template<> std::vector< bool > ArgumentParser::get_value< std::vector< bool > > (const std::string &valueName)`
- `template<> std::vector< std::string > ArgumentParser::get_value< std::vector< std::string > > (const std::string &valueName)`

8.175.1 Function Documentation

8.175.1.1 ArgumentParser::get_value< std::string >()

```

template<>
std::string ArgumentParser::get_value< std::string > (
    const std::string & valueName )

```

Definition at line 564 of file `cppargs.cpp`.

8.175.1.2 `ArgumentParser::get_value< std::vector< bool > >()`

```
template<>
std::vector< bool > ArgumentParser::get\_value< std::vector< bool > > (
    const std::string & valueName )
```

Definition at line 564 of file [cppargs.cpp](#).

8.175.1.3 `ArgumentParser::get_value< std::vector< double > >()`

```
template<>
std::vector< double > ArgumentParser::get\_value< std::vector< double > > (
    const std::string & valueName )
```

Definition at line 564 of file [cppargs.cpp](#).

8.175.1.4 `ArgumentParser::get_value< std::vector< int > >()`

```
template<>
std::vector< int > ArgumentParser::get\_value< std::vector< int > > (
    const std::string & valueName )
```

Definition at line 564 of file [cppargs.cpp](#).

8.175.1.5 `ArgumentParser::get_value< std::vector< std::string > >()`

```
template<>
std::vector< std::string > ArgumentParser::get\_value< std::vector< std::string > > (
    const std::string & valueName )
```

Definition at line 564 of file [cppargs.cpp](#).

8.176 `cppargs.cpp`

[Go to the documentation of this file.](#)

```
00001 #include "cppargs.h"
00002 // HelpMessage class
00003 HelpMessage::HelpMessage(std::string prog, std::string description) {
00004     std::cout << description << std::endl << std::endl;
00005 }
00006
00007 void HelpMessage::print(std::vector<Argument>& Pos_args, std::vector<Argument>& Opt_args) {
00008     int sentence_size = 0;
00009     if ( Pos_args.size() != 0 ) {
00010         std::cout << "positional arguments:" << std::endl;
00011         for ( Argument& arg : Pos_args ) {
00012             std::cout << " " << arg.getArgDenoteOut();
00013             sentence_size += arg.getArgDenoteOut().size() + 2;
00014             if ( arg.getArgDomain().size() != 0 ) {
00015                 int rest = 0;
00016                 std::cout << " {";
00017                 sentence_size += 2;
00018                 for ( std::string& domain : arg.getArgDomain() ) {
00019                     if ( rest < arg.getArgDomain().size() - 1 ) {
00020                         std::cout << domain << ",";
00021                         sentence_size += domain.size() + 1;
00022                     } else {
00023                         std::cout << domain;
```

```

00024         sentence_size += domain.size();
00025     }
00026     rest++;
00027 }
00028 std::cout << " ";
00029 sentence_size += 1;
00030 }
00031 if ( arg.getArgMinorOpt().size() != 0 ) {
00032     for ( std::string& minorOpt : arg.getArgMinorOpt() ) {
00033         std::cout << " " << minorOpt;
00034         sentence_size += 2;
00035     }
00036     if ( arg.getArgDomain().size() != 0 ) {
00037         int rest = 0;
00038         std::cout << " {";
00039         sentence_size += 2;
00040         for ( std::string& domain : arg.getArgDomain() ) {
00041             if ( rest < arg.getArgDomain().size() - 1 ) {
00042                 std::cout << domain << ",";
00043                 sentence_size += domain.size() + 1;
00044             } else {
00045                 std::cout << domain;
00046                 sentence_size += domain.size();
00047             }
00048             rest++;
00049         }
00050         std::cout << " ";
00051         sentence_size += 1;
00052     }
00053 }
00054 if ( sentence_size < 9 ) {
00055     std::cout << "\t\t\t";
00056 } else if ( sentence_size < 17 ) {
00057     std::cout << "\t\t";
00058 } else if ( sentence_size < 25 ) {
00059     std::cout << "\t";
00060 } else {
00061     std::cout << std::endl << "\t\t\t";
00062 }
00063 std::cout << arg.getArgDescription();
00064 std::cout << std::endl;
00065 sentence_size = 0;
00066 }
00067 }
00068 std::cout << std::endl;
00069 std::cout << "optional arguments:" << std::endl;
00070 std::cout << " -h, --help\t\t\tshow this help message and exit" << std::endl;
00071 if ( Opt_args.size() != 0 ) {
00072     for ( Argument& arg : Opt_args ) {
00073         std::cout << " " << arg.getArgDenoteOut();
00074         sentence_size += 2 + arg.getArgDenoteOut().size();
00075         if ( arg.getArgDomain().size() != 0 ) {
00076             int rest = 0;
00077             std::cout << " {";
00078             sentence_size += 2;
00079             for ( std::string& domain : arg.getArgDomain() ) {
00080                 if ( rest < arg.getArgDomain().size() - 1 ) {
00081                     std::cout << domain << ",";
00082                     sentence_size += domain.size() + 1;
00083                 } else {
00084                     std::cout << domain;
00085                     sentence_size += domain.size();
00086                 }
00087                 rest++;
00088             }
00089             std::cout << " ";
00090             sentence_size += 1;
00091         }
00092         if ( arg.getArgMinorOpt().size() != 0 ) {
00093             for ( std::string minorOpt : arg.getArgMinorOpt() ) {
00094                 std::cout << " " << minorOpt;
00095                 sentence_size += 2;
00096             }
00097             if ( arg.getArgDomain().size() != 0 ) {
00098                 int rest = 0;
00099                 std::cout << " {";
00100                 sentence_size += 2;
00101                 for ( std::string& domain : arg.getArgDomain() ) {
00102                     if ( rest < arg.getArgDomain().size() - 1 ) {
00103                         std::cout << domain << ",";
00104                         sentence_size += domain.size() + 1;
00105                     } else {
00106                         std::cout << domain;
00107                         sentence_size += domain.size();
00108                     }
00109                     rest++;
00110                 }

```

```

00111         rest = 0;
00112         std::cout << " ";
00113         sentence_size += 1;
00114     }
00115 }
00116 if ( sentence_size < 8 ) {
00117     std::cout << "\t\t\t";
00118 } else if ( sentence_size < 16 ) {
00119     std::cout << "\t\t";
00120 } else if ( sentence_size < 24 ) {
00121     std::cout << "\t";
00122 } else {
00123     std::cout << std::endl << "\t\t\t";
00124 }
00125 std::cout << arg.getArgDescription();
00126 std::cout << std::endl;
00127 sentence_size = 0;
00128 }
00129 }
00130 }
00131
00132 // Argument class
00133 Argument::Argument(std::string str) : _argOpt(str) { }
00134
00135 void Argument::setArgType(ARGTYPES typ) {
00136     _argType = typ;
00137 }
00138
00139 void Argument::setArgValue(std::string str) {
00140     _argValueList.push_back(str);
00141 }
00142
00143 void Argument::setArgValueList(std::vector<std::string> strList) {
00144     _argValueList.reserve(_argValueList.size() + strList.size());
00145     _argValueList.insert(_argValueList.end(), strList.begin(), strList.end());
00146 }
00147
00148 void Argument::replaceArgValueList(std::vector<std::string> strList) {
00149     _argValueList.clear();
00150     _argValueList.reserve(strList.size());
00151     _argValueList.assign(strList.begin(), strList.end());
00152 }
00153
00154 void Argument::setArgDomain(std::string str) {
00155     _argDomain.push_back(str);
00156 }
00157
00158 void Argument::setArgDomain(std::vector<std::string> strList) {
00159     _argDomain.reserve(_argDomain.size() + strList.size());
00160     _argDomain.insert(_argDomain.end(), strList.begin(), strList.end());
00161 }
00162
00163 void Argument::setArgName(std::string str) {
00164     _argName = str;
00165 }
00166
00167 void Argument::setArgOpt(std::string str) {
00168     _argOpt = str;
00169 }
00170
00171 void Argument::setArgMinorOpt(std::string str) {
00172     _argMinorOpt.push_back(str);
00173 }
00174
00175 void Argument::setArgMinorOpt(std::vector<std::string> strList) {
00176     _argMinorOpt.reserve(_argMinorOpt.size() + strList.size());
00177     _argMinorOpt.insert(_argMinorOpt.end(), strList.begin(), strList.end());
00178 }
00179
00180 void Argument::setArgDenoteOut(std::string str) {
00181     _argDenoteOut = str;
00182 }
00183
00184 void Argument::setArgDenoteIn(std::string str) {
00185     _argDenoteIn = str;
00186 }
00187
00188 void Argument::setArgDescription(std::string description) {
00189     _description = description;
00190 }
00191
00192 void Argument::isArgMulti() {
00193     _isArgMulti = true;
00194 }
00195
00196 void Argument::notArgMulti() {
00197     _isArgMulti = false;

```



```

00198 }
00199
00200 void Argument::isArgConst() {
00201     _isArgConst = true;
00202 }
00203
00204 void Argument::notArgConst() {
00205     _isArgConst = false;
00206 }
00207
00208 ARGYPES Argument::getArgType() const {
00209     return _argType;
00210 }
00211
00212 std::string Argument::getArgValue(const int order) const {
00213     return _argValueList[order];
00214 }
00215
00216 std::vector<std::string> Argument::getArgValueList() const {
00217     return _argValueList;
00218 }
00219
00220 std::string Argument::getArgDomain(const int order) const {
00221     return _argDomain[order];
00222 }
00223
00224 std::vector<std::string> Argument::getArgDomain() const {
00225     return _argDomain;
00226 }
00227
00228 std::string Argument::getArgName() const {
00229     return _argName;
00230 }
00231
00232 std::string Argument::getArgOpt() const {
00233     return _argOpt;
00234 }
00235
00236 std::vector<std::string> Argument::getArgMinorOpt() const {
00237     return _argMinorOpt;
00238 }
00239
00240 std::string Argument::getArgDenoteOut() const {
00241     return _argDenoteOut;
00242 }
00243
00244 std::string Argument::getArgDenoteIn() const {
00245     return _argDenoteIn;
00246 }
00247
00248 std::string Argument::getArgDescription() const {
00249     return _description;
00250 }
00251
00252 bool Argument::getArgMulti() const {
00253     return _isArgMulti;
00254 }
00255
00256 bool Argument::getArgConst() const {
00257     return _isArgConst;
00258 }
00259
00260 // ArgumentParser class
00261 ArgumentParser::ArgumentParser(int _argc, char** _argv) {
00262     for ( int arg = 0; arg < _argc; arg++ ) {
00263         argv.push_back(_argv[arg]);
00264     }
00265     std::string key = "Positional";
00266     std::vector<std::string> temp_value;
00267
00268     bool onlyPositional = true;
00269
00270     for ( std::string& arg : argv ) {
00271         if ( arg == "--help" || arg == "-h" ) {
00272             needHelp = true;
00273         }
00274         if ( arg[0] != '-' && arg[0] != '.' ) {
00275             temp_value.push_back(arg);
00276         }
00277         if ( arg[0] == '-' ) {
00278             argv_init.insert({key, temp_value});
00279             key = arg;
00280             temp_value.clear();
00281         }
00282         if ( arg == argv.back() && onlyPositional ) {
00283             argv_init.insert({key, temp_value});
00284         }

```

```

00285     }
00286 }
00287
00288 ArgumentParser::~ArgumentParser() { }
00289
00290 ArgumentParser& ArgumentParser::add_argument(const std::string& opts) {
00291     args_temp = new Argument(opts);
00292     std::string name;
00293     for ( int i = 0; i < opts.length(); i++ ) {
00294         if ( opts[i] != '-' ) {
00295             name = opts.substr(i, opts.length());
00296             break;
00297         }
00298     }
00299     args_temp->setArgName(name);
00300     args_temp->setArgDenoteIn(name);
00301     args_temp->setArgDenoteOut(opts);
00302     return *this;
00303 }
00304
00305 ArgumentParser& ArgumentParser::add_minor_argument(const std::string& opts) {
00306     args_temp->setArgMinorOpt(opts);
00307     return *this;
00308 }
00309
00310 ArgumentParser& ArgumentParser::add_domain(const std::vector<std::string>& opts) {
00311     args_temp->setArgDomain(opts);
00312     return *this;
00313 }
00314
00315 ArgumentParser& ArgumentParser::dest(const std::string& str) {
00316     args_temp->setArgDenoteIn(str);
00317     return *this;
00318 }
00319
00320 ArgumentParser& ArgumentParser::metavar(const std::string& str) {
00321     args_temp->setArgDenoteOut(str);
00322     return *this;
00323 }
00324
00325 ArgumentParser& ArgumentParser::set_const() {
00326     args_temp->isArgConst();
00327     return *this;
00328 }
00329
00330 ArgumentParser& ArgumentParser::nargs() {
00331     args_temp->isArgMulti();
00332     return *this;
00333 }
00334
00335 ArgumentParser& ArgumentParser::type(std::string typeOpt) {
00336     ARGYPES argType;
00337     if ( typeOpt == "int" ) {
00338         argType = ARGYPES::INT;
00339     } else if ( typeOpt == "double" ) {
00340         argType = ARGYPES::DOUBLE;
00341     } else if ( typeOpt == "bool" ) {
00342         argType = ARGYPES::BOOL;
00343     } else if ( typeOpt == "string" ) {
00344         argType = ARGYPES::STRING;
00345     } else {
00346         argType = ARGYPES::NONE;
00347         std::cout << "NO TYPE WARNING (SET TO ARBITRARY TYPE)" << std::endl;
00348     }
00349     args_temp->setArgType(argType);
00350     return *this;
00351 }
00352
00353 ArgumentParser& ArgumentParser::help(std::string message) {
00354     args_temp->setArgDescription(message);
00355     return *this;
00356 }
00357
00358 ArgumentParser& ArgumentParser::set_default(std::string value) {
00359     args_temp->setArgValue(value);
00360     return *this;
00361 }
00362
00363 ARGYPES ArgumentParser::detType(const std::string& str) {
00364     std::istringstream iss(str);
00365     int intVal;
00366     float floatVal;
00367     if ( iss >> intVal && iss.eof() ) {
00368         return ARGYPES::INT;
00369     } else if ( iss.clear(), iss.seekg(0), iss >> floatVal && iss.eof() ) {
00370         return ARGYPES::DOUBLE;
00371     } else if ( str == "true" || str == "false" ) {

```

```

00372         return ARGTYPES::BOOL;
00373     } else {
00374         return ARGTYPES::STRING;
00375     }
00376 }
00377
00378 void ArgumentParser::add_finish() {
00379     Argument* args_add = args_temp;
00380     args_add->getArgOpt()[0] == '-' ? Opt_args.push_back(*args_add) : Pos_args.push_back(*args_add);
00381     args_temp = nullptr;
00382 }
00383
00384 void ArgumentParser::parse_args() {
00385     if ( needHelp ) {
00386         HelpMessage* message = new HelpMessage(prog, description);
00387         message->print(Pos_args, Opt_args);
00388         exit(0);
00389     }
00390
00391     if ( Pos_args.size() > argv_init["Positional"].size() ) {
00392         std::cout << "ERROR: Not enough required arguments." << std::endl;
00393         exit(0);
00394     }
00395
00396     int numOfMulti = 0;
00397     for ( Argument& pos : Pos_args ) {
00398         if ( pos.getArgMulti() ) {
00399             numOfMulti++;
00400         }
00401     }
00402     if ( numOfMulti > 2 || (numOfMulti == 1 && !Pos_args.back().getArgMulti()) ) {
00403         std::cout << "Cannot classify that the arguments are belong to group" << std::endl;
00404         exit(0);
00405     }
00406
00407     int numArg = 0;
00408     for ( Argument& opt : Pos_args ) {
00409         if ( numArg < Pos_args.size() - 1 ) {
00410             opt.replaceArgValueList({(argv_init.find("Positional")->second)[0]});
00411             argv_init.find("Positional")->second.erase(argv_init.find("Positional")->second.begin());
00412         } else {
00413             opt.replaceArgValueList(argv_init.find("Positional")->second);
00414         }
00415         numArg++;
00416     }
00417
00418     for ( Argument& opt : Pos_args ) {
00419         if ( opt.getArgType() == ARGTYPES::NONE ) {
00420             opt.setArgType(detType(opt.getArgValueList()[0]));
00421         } else {
00422             if ( opt.getArgValueList().size() != 0 && opt.getArgType() !=
detType(opt.getArgValueList()[0]) ) {
00423                 std::cout << "Not proper type variable" << std::endl;
00424                 exit(0);
00425             }
00426         }
00427         if ( !opt.getArgMulti() && opt.getArgValueList().size() > 1 ) {
00428             std::cout << "Cannot enter more than 2 variables into this variable" << std::endl;
00429             exit(0);
00430         }
00431     }
00432
00433     for ( std::pair<std::string, std::vector<std::string> argv : argv_init ) {
00434         if ( argv.first == "Positional" ) continue;
00435         bool isArguExist = false;
00436         for ( Argument& opt : Opt_args ) {
00437             if ( opt.getArgOpt() == argv.first ) {
00438                 isArguExist = true;
00439                 opt.replaceArgValueList(argv.second);
00440                 break;
00441             }
00442         }
00443         if ( !isArguExist ) {
00444             std::cout << "Undefined argument" << std::endl;
00445             exit(0);
00446         }
00447     }
00448     for ( Argument& opt : Opt_args ) {
00449         if ( opt.getArgType() == ARGTYPES::NONE ) {
00450             opt.setArgType(detType(opt.getArgValueList()[0]));
00451         } else {
00452             if ( opt.getArgValueList().size() != 0 && opt.getArgType() !=
detType(opt.getArgValueList()[0]) ) {
00453                 std::cout << "Not proper type variable" << std::endl;
00454                 exit(0);
00455             }
00456         }
00457     }

```

```

00457         if ( !opt.getArgMulti() && opt.getArgValueList().size() > 1 ) {
00458             std::cout << "Cannot enter more than 2 variables into this variable" << std::endl;
00459             exit(0);
00460         }
00461     }
00462 }
00463
00464 template<typename T>
00465 T ArgumentParser::get_value(const std::string& valueName) {
00466 }
00467
00468 template<>
00469 int ArgumentParser::get_value<int>(const std::string& valueName) {
00470     bool isexist = false;
00471     ARGTYPES returnType = ARGTYPES::NONE;
00472     int returnValue;
00473     for ( Argument pos : Pos_args ) {
00474         if ( pos.getArgName() == valueName ) {
00475             returnValue = stoi(pos.getArgValueList()[0]);
00476             returnType = pos.getArgType();
00477             isexist = true;
00478             break;
00479         }
00480     }
00481     for ( Argument opt : Opt_args ) {
00482         if ( opt.getArgName() == valueName ) {
00483             returnValue = stoi(opt.getArgValueList()[0]);
00484             returnType = opt.getArgType();
00485             isexist = true;
00486             break;
00487         }
00488     }
00489     if ( !isexist ) {
00490         std::cout << "No proper argument" << std::endl;
00491         exit(0);
00492     }
00493     if ( returnType != ARGTYPES::INT ) {
00494         std::cout << "Not proper type" << std::endl;
00495         exit(0);
00496     }
00497     return returnValue;
00498 }
00499
00500 template<>
00501 double ArgumentParser::get_value<double>(const std::string& valueName) {
00502     bool isexist = false;
00503     ARGTYPES returnType = ARGTYPES::NONE;
00504     double returnValue;
00505     for ( Argument pos : Pos_args ) {
00506         if ( pos.getArgName() == valueName ) {
00507             returnValue = stod(pos.getArgValueList()[0]);
00508             returnType = pos.getArgType();
00509             isexist = true;
00510             break;
00511         }
00512     }
00513     for ( Argument opt : Opt_args ) {
00514         if ( opt.getArgName() == valueName ) {
00515             returnValue = stod(opt.getArgValueList()[0]);
00516             returnType = opt.getArgType();
00517             isexist = true;
00518             break;
00519         }
00520     }
00521     if ( !isexist ) {
00522         std::cout << "No proper argument" << std::endl;
00523         exit(0);
00524     }
00525     if ( returnType != ARGTYPES::DOUBLE ) {
00526         std::cout << "Not proper type" << std::endl;
00527         exit(0);
00528     }
00529     return returnValue;
00530 }
00531
00532 template<>
00533 bool ArgumentParser::get_value<bool>(const std::string& valueName) {
00534     bool isexist = false;
00535     ARGTYPES returnType = ARGTYPES::NONE;
00536     bool returnValue;
00537     for ( Argument pos : Pos_args ) {
00538         if ( pos.getArgName() == valueName ) {
00539             returnValue = pos.getArgValueList()[0] == "true" ? true : false;
00540             returnType = pos.getArgType();
00541             isexist = true;
00542             break;
00543         }

```

```

00544     }
00545     for ( Argument opt : Opt_args ) {
00546         if ( opt.getArgName() == valueName ) {
00547             returnValue = opt.getArgValueList()[0] == "true" ? true : false;
00548             returnType = opt.getArgType();
00549             isexist = true;
00550             break;
00551         }
00552     }
00553     if ( !isexist ) {
00554         std::cout << "No proper argument" << std::endl;
00555         exit(0);
00556     }
00557     if ( returnType != ARGYPES::BOOL ) {
00558         std::cout << "Not proper type" << std::endl;
00559         exit(0);
00560     }
00561     return returnValue;
00562 }
00563 template<>
00564 std::string ArgumentParser::get_value<std::string>(const std::string& valueName) {
00565     bool isexist = false;
00566     ARGYPES returnType = ARGYPES::NONE;
00567     std::string returnValue;
00568     for ( Argument pos : Pos_args ) {
00569         if ( pos.getArgName() == valueName ) {
00570             returnValue = pos.getArgValueList()[0];
00571             returnType = pos.getArgType();
00572             isexist = true;
00573             break;
00574         }
00575     }
00576     for ( Argument opt : Opt_args ) {
00577         if ( opt.getArgName() == valueName ) {
00578             returnValue = opt.getArgValueList()[0];
00579             returnType = opt.getArgType();
00580             isexist = true;
00581             break;
00582         }
00583     }
00584     if ( !isexist ) {
00585         std::cout << "No proper argument" << std::endl;
00586         exit(0);
00587     }
00588     if ( returnType != ARGYPES::STRING ) {
00589         std::cout << "Not proper type" << std::endl;
00590         exit(0);
00591     }
00592     return returnValue;
00593 }
00594
00595 template<>
00596 std::vector<int> ArgumentParser::get_value<std::vector<int>>(const std::string& valueName) {
00597     bool isexist = false;
00598     ARGYPES returnType = ARGYPES::NONE;
00599     std::vector<int> returnValue;
00600     for ( Argument pos : Pos_args ) {
00601         if ( pos.getArgName() == valueName ) {
00602             for ( std::string val : pos.getArgValueList() ) {
00603                 returnValue.push_back(stoi(val));
00604             }
00605             returnType = pos.getArgType();
00606             isexist = true;
00607             break;
00608         }
00609     }
00610     for ( Argument opt : Opt_args ) {
00611         if ( opt.getArgName() == valueName ) {
00612             for ( std::string val : opt.getArgValueList() ) {
00613                 returnValue.push_back(stoi(val));
00614             }
00615             returnType = opt.getArgType();
00616             isexist = true;
00617             break;
00618         }
00619     }
00620     if ( !isexist ) {
00621         std::cout << "No proper argument" << std::endl;
00622         exit(0);
00623     }
00624     if ( returnType != ARGYPES::INT ) {
00625         std::cout << "Not proper type" << std::endl;
00626         exit(0);
00627     }
00628
00629     return returnValue;
00630 }

```

```

00631
00632 template<>
00633 std::vector<double> ArgumentParser::get_value<std::vector<double>>(const std::string& valueName) {
00634     bool isexist = false;
00635     ARGYPES returnType = ARGYPES::NONE;
00636     std::vector<double> returnValue;
00637     for ( Argument pos : Pos_args ) {
00638         if ( pos.getArgName() == valueName ) {
00639             for ( std::string val : pos.getArgValueList() ) {
00640                 returnValue.push_back(stod(val));
00641             }
00642             returnType = pos.getArgType();
00643             isexist = true;
00644             break;
00645         }
00646     }
00647     for ( Argument opt : Opt_args ) {
00648         if ( opt.getArgName() == valueName ) {
00649             for ( std::string val : opt.getArgValueList() ) {
00650                 returnValue.push_back(stod(val));
00651             }
00652             returnType = opt.getArgType();
00653             isexist = true;
00654             break;
00655         }
00656     }
00657     if ( !isexist ) {
00658         std::cout << "No proper argument" << std::endl;
00659         exit(0);
00660     }
00661     if ( returnType != ARGYPES::DOUBLE ) {
00662         std::cout << "Not proper type" << std::endl;
00663         exit(0);
00664     }
00665     return returnValue;
00666 }
00667
00668 template<>
00669 std::vector<bool> ArgumentParser::get_value<std::vector<bool>>(const std::string& valueName) {
00670     bool isexist = false;
00671     ARGYPES returnType = ARGYPES::NONE;
00672     std::vector<bool> returnValue;
00673     for ( Argument pos : Pos_args ) {
00674         if ( pos.getArgName() == valueName ) {
00675             std::transform(pos.getArgValueList().begin(), pos.getArgValueList().end(),
00676                 std::back_inserter(returnValue), [ ](const std::string& str) { return str == "true" ? true : false;
00677             });
00678             returnType = pos.getArgType();
00679             isexist = true;
00680             break;
00681         }
00682     }
00683     for ( Argument opt : Opt_args ) {
00684         if ( opt.getArgName() == valueName ) {
00685             std::transform(opt.getArgValueList().begin(), opt.getArgValueList().end(),
00686                 std::back_inserter(returnValue), [ ](const std::string& str) { return str == "true" ? true : false;
00687             });
00688             returnType = opt.getArgType();
00689             isexist = true;
00690             break;
00691         }
00692     }
00693     if ( !isexist ) {
00694         std::cout << "No proper argument" << std::endl;
00695         exit(0);
00696     }
00697     if ( returnType != ARGYPES::BOOL ) {
00698         std::cout << "Not proper type" << std::endl;
00699         exit(0);
00700     }
00701     return returnValue;
00702 }
00703
00704 template<>
00705 std::vector<std::string> ArgumentParser::get_value<std::vector<std::string>>(const std::string&
00706     valueName) {
00707     bool isexist = false;
00708     ARGYPES returnType = ARGYPES::NONE;
00709     std::vector<std::string> returnValue;
00710     for ( Argument pos : Pos_args ) {
00711         if ( pos.getArgName() == valueName ) {
00712             returnValue = pos.getArgValueList();
00713             returnType = pos.getArgType();
00714             isexist = true;
00715             break;
00716         }
00717     }
00718     for ( Argument opt : Opt_args ) {

```

```

00713         if ( opt.getArgName() == valueName ) {
00714             returnValue = opt.getArgValueList();
00715             returnType = opt.getArgType();
00716             isexist = true;
00717             break;
00718         }
00719     }
00720     if ( !isexist ) {
00721         std::cout << "No proper argument" << std::endl;
00722         exit(0);
00723     }
00724     if ( returnType != ARGYPES::STRING ) {
00725         std::cout << "Not proper type" << std::endl;
00726         exit(0);
00727     }
00728     return returnValue;
00729 }

```

8.177 /home/ychoi/ATOM/pycpp/src/cppTimer.cpp File Reference

```
#include "cppTimer.h"
```

8.178 cppTimer.cpp

[Go to the documentation of this file.](#)

```

00001 #include "cppTimer.h"
00002
00003 TTimer::TTimer() {
00004     start = clock();
00005 }
00006
00007 void TTimer::Measure() {
00008     finish = clock();
00009     duration = (double)(finish-start) / CLOCKS_PER_SEC;
00010     std::cout << "Time from the start flows " << duration << "s" << std::endl;
00011 }
00012
00013 void TTimer::EndProgram() {
00014     finish = clock();
00015     duration = (double)(finish-start) / CLOCKS_PER_SEC;
00016     std::cout << "Total run time of this program is " << duration << "s" << std::endl;
00017     std::cout << "Good bye!" << std::endl;
00018 }

```

8.179 /home/ychoi/ATOM/pycpp/src/cpptqdm.cpp File Reference

```
#include "cpptqdm.h"
```

8.180 cpptqdm.cpp

[Go to the documentation of this file.](#)

```

00001 #include "cpptqdm.h"
00002
00003 ProgressBar::ProgressBar(int setSize) : mSetSize(setSize) {
00004     getTerminalLength();
00005     start_time = std::chrono::system_clock::now();
00006     printPoint = start_time;
00007     std::cout << "\x1b[?251";
00008 }
00009

```

```

00010 ProgressBar::ProgressBar() {
00011     getTerminalLength();
00012     start_time = std::chrono::system_clock::now();
00013     printPoint = start_time;
00014 }
00015
00016 ProgressBar::~ProgressBar() {
00017     std::cout << "\x1b[?25h" << std::endl;
00018 }
00019
00020 void ProgressBar::getTerminalLength() {
00021     struct winsize w;
00022     ioctl(STDOUT_FILENO, TIOCGWINSZ, &w);
00023     mTerminalWidth = w.ws_col;
00024 }
00025
00026 int ProgressBar::getMinute(int num) {
00027     return (num % 3600) / 60;
00028 }
00029
00030 int ProgressBar::getSecond(int num) {
00031     return num % 60;
00032 }
00033
00034 void ProgressBar::printProgress() {
00035     called++;
00036     std::chrono::system_clock::time_point now = std::chrono::system_clock::now();
00037     if ( std::chrono::duration_cast<std::chrono::milliseconds>(now - printPoint).count() > 1 || called
== mSetSize || called == 1 ) {
00038         double percent = (double) called / mSetSize * 100;
00039         double duration = std::chrono::duration_cast<std::chrono::milliseconds>(now -
start_time).count();
00040         double speed = 1000. * called / duration;
00041         int left = (mSetSize - called) / speed;
00042         int iBar = mTerminalWidth - (32 + 2 * (floor(log10(mSetSize)) + 1) + (getMinute(duration *
0.001) > 99 ? floor(log10(getMinute(duration * 0.001))) + 1 : 2) + (getMinute(left) > 99 ?
floor(log10(getMinute(left))) + 1 : 2));
00043         printPoint = now;
00044         std::cout << "\r" << std::setw(3) << std::setfill(' ') << std::fixed << (int) percent
<< "%" << std::setw(floor(iBar * percent / 100 - 0.000001)) << std::setfill('=') << ""
<< std::setw(floor(iBar * (100 - percent) / 100) + 1) << std::setfill(' ') << "]"
00045         << std::setw(floor(log10(mSetSize)) + 1) << std::setfill(' ') << called << "/" << mSetSize
00046         << " ["
00047         << std::setw(2) << std::setfill('0') << getMinute(duration * 0.001) << ":" << std::setw(2) <<
std::setfill('0') << getSecond(duration * 0.001) << "<"
00048         << std::setw(2) << std::setfill('0') << getMinute(left) << ":" << std::setw(2) <<
std::setfill('0') << getSecond(left) << ", "
00049         << std::setprecision(10 - 1 - (speed == 0 ? 1 : (int) log10(abs((int) speed)) + 1)) << speed
00050         << "it/s]";
00051     }
00052 }
00053
00054 }

```

8.181 /home/ychoi/ATOM/pycpp/src/cppUnit.cpp File Reference

```
#include "cppUnit.h"
```

Functions

- `std::ostream & operator<< (std::ostream &os, Unit &ref)`
- `std::ostream & operator<< (std::ostream &os, const Unit &ref)`
- `std::ostream & operator<< (std::ostream &os, Quantity &ref)`
- `std::ostream & operator<< (std::ostream &os, const Quantity &ref)`

8.181.1 Function Documentation

8.181.1.1 operator<<() [1/4]

```
std::ostream & operator<< (
    std::ostream & os,
    const Quantity & ref )
```

Definition at line 398 of file `cppUnit.cpp`.

8.181.1.2 operator<<() [2/4]

```
std::ostream & operator<< (
    std::ostream & os,
    const Unit & ref )
```

Definition at line 240 of file [cppUnit.cpp](#).

8.181.1.3 operator<<() [3/4]

```
std::ostream & operator<< (
    std::ostream & os,
    Quantity & ref )
```

Definition at line 393 of file [cppUnit.cpp](#).

8.181.1.4 operator<<() [4/4]

```
std::ostream & operator<< (
    std::ostream & os,
    Unit & ref )
```

Definition at line 225 of file [cppUnit.cpp](#).

8.182 cppUnit.cpp

[Go to the documentation of this file.](#)

```
00001 #include "cppUnit.h"
00002
00003 Unit::Unit() { }
00004
00005 Unit::Unit(std::string_view unit) {
00006     setUnit(unit);
00007 }
00008
00009 std::vector<std::tuple<std::string, std::string, Unit> Quantity::userQuantity = { };
00010 std::vector<std::pair<std::string, std::array<int, 7>> Unit::userUnits = { };
00011 std::vector<std::string> Unit::exceptPrefixList = {"m", "mol", "cd"};
00012 bool Unit::isUserUnit = false;
00013
00014 void Unit::setUserUnit(std::string_view newUnit, std::string_view newSiUnit) {
00015     Unit object;
00016     std::vector<std::string> store;
00017     object.seperate(store, newSiUnit);
00018     object.vanishSlash(store);
00019     object.removePrefix(store);
00020
00021     int iNow = 0;
00022     for ( std::string& str : store ) {
00023         if ( isalpha(str[0]) ) {
00024             object.setUnitCount(store[iNow], stoi(store[iNow + 1]));
00025         }
00026         iNow++;
00027     }
00028     userUnits.push_back({std::string(newUnit), object.getUnitCount()});
00029     if ( prefix.count(newUnit[0]) ) {
00030         exceptPrefixList.push_back(std::string(newUnit));
00031     }
00032     isUserUnit = true;
00033 }
00034
00035
00036 void Quantity::setUserQuantity() {
00037     std::ifstream unitFile("~/source/ATOM/src/pycpp/UserUnit.txt");
```

```

00038     std::string str;
00039
00040     while ( getline(unitFile, str) ) {
00041         std::istringstream strstr(str);
00042         std::string newUnit, relation, newSiUnit;
00043         strstr » newUnit » relation » newSiUnit;
00044         Unit::setUserUnit(newUnit, newSiUnit);
00045         Quantity::userQuantity.push_back(std::make_tuple(newUnit, relation, Unit(newSiUnit)));
00046     }
00047     unitFile.close();
00048 }
00049
00050 void Unit::setUnit(std::string_view unit) {
00051     std::vector<std::string> store;
00052
00053     if ( seperate(store, unit) || vanishSlash(store) || removePrefix(store) ) { }
00054
00055     int iNow = 0;
00056     for ( std::string& str : store ) {
00057         if ( isalpha(str[0]) ) {
00058             setUnitCount(store[iNow], stoi(store[iNow + 1]));
00059         }
00060         iNow++;
00061     }
00062 }
00063
00064 bool Unit::seperate(std::vector<std::string>& store, std::string_view unit) {
00065     int iStart = 0;
00066     int iNow = 0;
00067
00068     bool doubleSlash = false;
00069     bool isNum = false;
00070
00071     bool error = false;
00072     for ( const char letter : unit ) {
00073         if ( letter == '*' ) {
00074             store.push_back(static_cast<std::string>(unit.substr(iStart, iNow - iStart)));
00075             store.push_back(static_cast<std::string>(unit.substr(iNow, 1)));
00076             iStart = iNow + 1;
00077         } else if ( letter == '/' ) {
00078             if ( doubleSlash ) {
00079                 std::cerr « "Unit notation to write '/' twice is forbidden" « std::endl;
00080                 error = true;
00081             }
00082             store.push_back(static_cast<std::string>(unit.substr(iStart, iNow - iStart)));
00083             store.push_back(static_cast<std::string>(unit.substr(iNow, 1)));
00084             iStart = iNow + 1;
00085             doubleSlash = true;
00086         } else if ( (letter == '-' || isdigit(letter)) && !isNum ) {
00087             store.push_back(static_cast<std::string>(unit.substr(iStart, iNow - iStart)));
00088             isNum = true;
00089             iStart = iNow;
00090         } else if ( isNum && !isdigit(letter) ) {
00091             iStart = iNow;
00092             isNum = false;
00093         }
00094         if ( iNow == unit.size() - 1 ) {
00095             store.push_back(static_cast<std::string>(unit.substr(iStart)));
00096         }
00097         iNow++;
00098     }
00099     return error;
00100 }
00101
00102 bool Unit::vanishSlash(std::vector<std::string>& store) {
00103     int iNow = 0;
00104     bool afterSlash = false;
00105     while ( store.end() - store.begin() - iNow > 0 ) {
00106         std::string str = store[iNow];
00107         if ( str == "/" ) {
00108             afterSlash = true;
00109             store[iNow] = "*";
00110         }
00111         if ( (str[0] == '-' || isdigit(str[0])) && afterSlash ) {
00112             store[iNow] = "-" + store[iNow];
00113         }
00114         if ( isalpha(str[0]) && ((store[iNow + 1][0] != '-' && !isdigit(store[iNow + 1][0])) ||
00115 (store.begin() + iNow + 1) == store.end()) ) {
00116             store.insert(store.begin() + iNow + 1, "1");
00117         }
00118         iNow++;
00119     }
00120     return false;
00121 }
00122 bool Unit::removePrefix(std::vector<std::string>& store) {
00123     mDigit = 0;

```

```

00124
00125     int iNow = 0;
00126     for ( std::string_view str : store ) {
00127         if ( prefix.count(str[0]) ) {
00128             if ( std::find(exceptPrefixList.begin(), exceptPrefixList.end(), str) !=
exceptPrefixList.end() ) {
00129                 } else if ( str.substr(1) == "g" ) {
00130                     mDigit += (prefix.find(str[0])->second - 3) * stoi(store[iNow + 1]);
00131                     store[iNow] = "kg";
00132                 } else {
00133                     mDigit += prefix.find(str[0])->second * stoi(store[iNow + 1]);
00134                     store[iNow] = str.substr(1);
00135                 }
00136             } else if ( isalpha(str[0]) ) {
00137             }
00138             iNow++;
00139         }
00140         return false;
00141     }
00142
00143 void Unit::setUnitCount(std::string unit, int power) {
00144     if ( unit == "m" ) {
00145         plusCount({1 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power});
00146     } else if ( unit == "kg" ) {
00147         plusCount({0 * power, 1 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power});
00148     } else if ( unit == "s" ) {
00149         plusCount({0 * power, 0 * power, 1 * power, 0 * power, 0 * power, 0 * power, 0 * power});
00150     } else if ( unit == "A" ) {
00151         plusCount({0 * power, 0 * power, 0 * power, 1 * power, 0 * power, 0 * power, 0 * power});
00152     } else if ( unit == "K" ) {
00153         plusCount({0 * power, 0 * power, 0 * power, 0 * power, 1 * power, 0 * power, 0 * power});
00154     } else if ( unit == "mol" ) {
00155         plusCount({0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 1 * power, 0 * power});
00156     } else if ( unit == "cd" ) {
00157         plusCount({0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 1 * power});
00158     } else if ( isUserUnit ) {
00159         for ( const std::pair<std::string, std::array<int, 7>&userUnit : userUnits ) {
00160             if ( userUnit.first == unit ) {
00161                 plusCount({userUnit.second[0] * power, userUnit.second[1] * power, userUnit.second[2]
* power, userUnit.second[3] * power, userUnit.second[4] * power, userUnit.second[5] * power,
userUnit.second[6] * power});
00162             }
00163         }
00164     } else {
00165         std::cerr << "'" << unit << "; is un-defined unit. Set user difine option." << std::endl;
00166     }
00167 }
00168
00169 void Unit::plusCount(std::array<int, 7> nums) {
00170     unitCount[0] += nums[0];
00171     unitCount[1] += nums[1];
00172     unitCount[2] += nums[2];
00173     unitCount[3] += nums[3];
00174     unitCount[4] += nums[4];
00175     unitCount[5] += nums[5];
00176     unitCount[6] += nums[6];
00177 }
00178
00179 bool Unit::operator==(const Unit& ref) const {
00180     return ref.unitCount == unitCount;
00181 }
00182
00183 bool Unit::operator!=(const Unit& ref) const {
00184     return ref.unitCount != unitCount;
00185 }
00186
00187 Unit Unit::operator*(const Unit& ref) const {
00188     Unit result = *this;
00189     result *= ref;
00190     return result;
00191 }
00192
00193 Unit Unit::operator*=(const Unit& ref) {
00194     mDigit += ref.mDigit;
00195     for ( int i = 0; i < 7; i++ ) {
00196         unitCount[i] += ref.unitCount[i];
00197     }
00198     for ( std::string_view nonBasic : ref.nonBasicUnit ) {
00199         if ( find(nonBasicUnit.begin(), nonBasicUnit.end(), nonBasic) != nonBasicUnit.end() ) {
00200             nonBasicUnit.push_back(static_cast<std::string>(nonBasic));
00201         }
00202     }
00203     return *this;
00204 }
00205
00206 Unit Unit::operator/(const Unit& ref) const {
00207     Unit result = *this;

```

```

00208     result /= ref;
00209     return result;
00210 }
00211
00212 Unit Unit::operator/=(const Unit& ref) {
00213     mDigit -= ref.mDigit;
00214     for ( int i = 0; i < 7; i++ ) {
00215         unitCount[i] -= ref.unitCount[i];
00216     }
00217     for ( std::string_view nonBasic : ref.nonBasicUnit ) {
00218         if ( find(nonBasicUnit.begin(), nonBasicUnit.end(), nonBasic) != nonBasicUnit.end() ) {
00219             nonBasicUnit.push_back(static_cast<std::string>(nonBasic));
00220         }
00221     }
00222     return *this;
00223 }
00224
00225 std::ostream& operator<(std::ostream& os, Unit& ref) {
00226     std::string unitTable[7] = {"m", "kg", "s", "A", "K", "mol", "cd"};
00227     int multiCount = 7 - std::count(std::begin(ref.unitCount), std::end(ref.unitCount), (int) 0);
00228     for ( int i = 0; i < 7; i++ ) {
00229         if ( ref.unitCount[i] != 0 ) {
00230             os << unitTable[i] << (int) ref.unitCount[i];
00231             multiCount--;
00232             if ( multiCount > 0 ) {
00233                 os << "*";
00234             }
00235         }
00236     }
00237     return os;
00238 }
00239
00240 std::ostream& operator<(std::ostream& os, const Unit& ref) {
00241     std::string unitTable[7] = {"m", "kg", "s", "A", "K", "mol", "cd"};
00242     int multiCount = 7 - std::count(std::begin(ref.unitCount), std::end(ref.unitCount), (int) 0);
00243     for ( int i = 0; i < 7; i++ ) {
00244         if ( ref.unitCount[i] != 0 ) {
00245             os << unitTable[i] << (int) ref.unitCount[i];
00246             multiCount--;
00247             if ( multiCount > 0 ) {
00248                 os << "*";
00249             }
00250         }
00251     }
00252     return os;
00253 }
00254
00255 const int Unit::getDigit() const {
00256     return mDigit;
00257 }
00258
00259 const std::array<int, 7>& Unit::getUnitCount() const {
00260     return unitCount;
00261 }
00262
00263 const std::vector<std::string> Unit::getNonBasicUnits() const {
00264     return nonBasicUnit;
00265 }
00266
00267 const std::string Unit::getUnit() const {
00268     std::string unitTable[7] = {"m", "kg", "s", "A", "K", "mol", "cd"};
00269     int multiCount = 7 - std::count(std::begin(unitCount), std::end(unitCount), (int) 0);
00270     std::string result;
00271     for ( int i = 0; i < 7; i++ ) {
00272         if ( unitCount[i] != 0 ) {
00273             result += unitTable[i] + std::to_string((int) unitCount[i]);
00274             multiCount--;
00275             if ( multiCount > 0 ) {
00276                 result += "*";
00277             }
00278         }
00279     }
00280     return result;
00281 }
00282
00283 Quantity::Quantity(std::string quantity) {
00284     int posChar = 0;
00285     for ( const uint letter : quantity ) {
00286         if ( isdigit(letter) || (letter == '.') ) {
00287             posChar++;
00288         } else {
00289             break;
00290         }
00291     }
00292     mNum = stof(quantity.substr(0, posChar));
00293     mDigit = floor(log10(mNum));
00294     mNum = mNum * pow(10, -mDigit);

```

```

00295
00296     mUnit = Unit(quantity.substr(posChar));
00297     mDigit += mUnit.getDigit();
00298 }
00299
00300 Quantity::Quantity(double num, std::string unit) {
00301     mDigit = floor(log10(mDigit));
00302     mNum = mNum * pow(10, -mDigit);
00303     mUnit = Unit(unit);
00304 }
00305
00306 double Quantity::getNum() const {
00307     return mNum * pow(10, mDigit);
00308 }
00309
00310 double Quantity::getNum(std::string_view unit) const {
00311     Unit givenUnit(unit);
00312     if ( mUnit != givenUnit ) {
00313         std::cerr << "The operands doesn't have same dimension." << mUnit << "!=" << givenUnit <<
std::endl;
00314     }
00315     return mNum * pow(10, mDigit - givenUnit.getDigit());
00316 }
00317
00318 const std::string Quantity::getUnit() const {
00319     return mUnit.getUnit();
00320 }
00321
00322 const std::string Quantity::getQuantity() const {
00323     return std::to_string(mNum * pow(10, mDigit));
00324 }
00325
00326 const std::string Quantity::getQuantity(std::string_view unit) const {
00327     Unit givenUnit(unit);
00328     if ( mUnit != givenUnit ) {
00329         std::cerr << "The operands doesn't have same dimension." << mUnit << "!=" << givenUnit <<
std::endl;
00330     }
00331     return std::to_string(mNum * pow(10, mDigit - givenUnit.getDigit())) +
static_cast<std::string>(unit);
00332 }
00333
00334 Quantity Quantity::operator+(const Quantity& ref) const {
00335     Quantity result = *this;
00336     result += ref;
00337     return result;
00338 }
00339 Quantity Quantity::operator-(const Quantity& ref) const {
00340     Quantity result = *this;
00341     result -= ref;
00342     return result;
00343 }
00344 Quantity Quantity::operator*(const Quantity& ref) const {
00345     Quantity result = *this;
00346     result *= ref;
00347     return result;
00348 }
00349 Quantity Quantity::operator/(const Quantity& ref) const {
00350     Quantity result = *this;
00351     result /= ref;
00352     return result;
00353 }
00354 Quantity Quantity::operator+=(const Quantity& ref) {
00355     if ( mUnit != ref.mUnit ) {
00356         std::cerr << "The operands doesn't have same dimension." << mUnit << "!=" << ref.mUnit <<
std::endl;
00357     } else {
00358         int mainDigit = std::max(mDigit, ref.mDigit);
00359         mNum = mNum * pow(10, mDigit - mainDigit) + ref.mNum * pow(10, ref.mDigit - mainDigit);
00360         mDigit = mainDigit + floor(log10(mNum));
00361         mNum = mNum * pow(10, -floor(log10(mNum)));
00362     }
00363     return *this;
00364 }
00365 Quantity Quantity::operator-=(const Quantity& ref) {
00366     if ( mUnit != ref.mUnit ) {
00367         std::cerr << "The operands doesn't have same dimension." << mUnit << "!=" << ref.mUnit <<
std::endl;
00368     } else {
00369         int mainDigit = std::max(mDigit, ref.mDigit);
00370         mNum = mNum * pow(10, mDigit - mainDigit) - ref.mNum * pow(10, ref.mDigit - mainDigit);
00371         mDigit = mainDigit + floor(log10(abs(mNum)));
00372         mNum = mNum * pow(10, -floor(log10(abs(mNum))));
00373     }
00374     return *this;
00375 }
00376 Quantity Quantity::operator*=(const Quantity& ref) {

```

```

00377     int mainDigit = std::max(mDigit, ref.mDigit);
00378     mNum = mNum * pow(10, mDigit - mainDigit) * ref.mNum * pow(10, ref.mDigit - mainDigit);
00379     mDigit = mainDigit + floor(log10(abs(mNum)));
00380     mNum = mNum * pow(10, -floor(log10(abs(mNum))));
00381     mUnit *= ref.mUnit;
00382     return *this;
00383 }
00384 Quantity Quantity::operator/=(const Quantity& ref) {
00385     int mainDigit = std::max(mDigit, ref.mDigit);
00386     mNum = mNum * pow(10, mDigit - mainDigit) / (ref.mNum * pow(10, ref.mDigit - mainDigit));
00387     mDigit = mainDigit + floor(log10(abs(mNum)));
00388     mNum = mNum * pow(10, -floor(log10(abs(mNum))));
00389     mUnit /= ref.mUnit;
00390     return *this;
00391 }
00392
00393 std::ostream& operator<<(std::ostream& os, Quantity& ref) {
00394     os << ref.mNum << "e" << ref.mDigit << " " << ref.mUnit;
00395     return os;
00396 }
00397
00398 std::ostream& operator<<(std::ostream& os, const Quantity& ref) {
00399     os << ref.mNum << "e" << ref.mDigit << " " << ref.mUnit;
00400     return os;
00401 }

```

8.183 /home/ychoi/ATOM/README.md File Reference

8.184 /home/ychoi/ATOM/simulation/inc/TEntrySimulation.h File Reference

```

#include <cmath>
#include <iostream>
#include "TMath.h"
#include "TGraph2D.h"
#include "TCanvas.h"

```

Classes

- struct [TDisk](#)
- struct [TDetector](#)
- class [TEntrySimulation](#)

8.185 TEntrySimulation.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __TENTRYSIMULATION__
00002 #define __TENTRYSIMULATION__
00003
00004 #include <cmath>
00005 #include <iostream>
00006
00007 #include "TMath.h"
00008 #include "TGraph2D.h"
00009 #include "TCanvas.h"
00010
00011 struct TDisk {
00012     double radius;
00013     double coordZ;
00014     bool isBelong(double x, double y) {
00015         if ( pow(x, 2) + pow(y, 2) < pow(radius, 2) ) {
00016             return true;

```

```

00017         } else {
00018             return false;
00019         }
00020     }
00021 };
00022
00023 struct TDetector {
00024     double width, height;
00025     double coordX, coordY, coordZ;
00026     bool isBelong(double x, double y) {
00027         if ( (abs(x - coordX) < width / 2) && (abs(y - coordY) < height / 2) ) {
00028             return true;
00029         } else {
00030             return false;
00031         }
00032     }
00033 };
00034
00035 class TEntrySimulation {
00036 private:
00037     TDisk source;
00038     TDisk upperDisk;
00039     TDisk lowerDisk;
00040     TDetector detector;
00041 public:
00042     void setInitGeometry(double diskRadius, double upperDiskCoordZ, double lowerDiskCoordZ, double
detectorWidth, double detectorHeight, double detectorCoordZ);
00043     void setSource(double sourceRadius);
00044     void setCollimator(double diskRadius, double upperDiskCoordZ, double lowerDiskCoordZ);
00045     void setDetector(double detectorWidth, double detectorHeight, double detectorCoordX, double
detectorCoordY, double detectorCoordZ);
00046
00047     double doCount();
00048 };
00049
00050
00051
00052 #endif

```

8.186 /home/ychoi/ATOM/simulation/src/TEntrySimulation.cpp File Reference

```
#include "TEntrySimulation.h"
```

8.187 TEntrySimulation.cpp

[Go to the documentation of this file.](#)

```

00001 #include "TEntrySimulation.h"
00002
00003 void TEntrySimulation::setInitGeometry(double diskRadius, double upperDiskCoordZ, double
lowerDiskCoordZ, double detectorWidth, double detectorHeight, double detectorCoordZ) {
00004     upperDisk.radius = diskRadius;
00005     upperDisk.coordZ = upperDiskCoordZ;
00006     lowerDisk.radius = diskRadius;
00007     lowerDisk.coordZ = lowerDiskCoordZ;
00008     detector.width = detectorWidth;
00009     detector.height = detectorHeight;
00010     detector.coordZ = detectorCoordZ;
00011 }
00012
00013 void TEntrySimulation::setSource(double sourceRadius) {
00014     source.radius = sourceRadius;
00015     source.coordZ = 0.;
00016 }
00017
00018 void TEntrySimulation::setCollimator(double diskRadius, double upperDiskCoordZ, double
lowerDiskCoordZ) {
00019     upperDisk.radius = diskRadius;
00020     upperDisk.coordZ = upperDiskCoordZ;
00021     lowerDisk.radius = diskRadius;
00022     lowerDisk.coordZ = lowerDiskCoordZ;
00023 }

```

```

00024
00025 void TEntrySimulation::setDetector(double detectorWidth, double detectorHeight, double detectorCoordX,
double detectorCoordY, double detectorCoordZ) {
00026     detector.width = detectorWidth;
00027     detector.height = detectorHeight;
00028     detector.coordX = detectorCoordX;
00029     detector.coordY = detectorCoordY;
00030     detector.coordZ = detectorCoordZ;
00031 }
00032
00033 double TEntrySimulation::doCount() {
00034     int nCount = 0;
00035     int totCount = 0;
00036     double effAngle = 0.;
00037     double sourceStep = source.radius / 50.;
00038     int iPoint = 0;
00039     for ( double x = -source.radius; x < source.radius + 2 * sourceStep; x += sourceStep ) {
00040         for ( double y = -source.radius; y < source.radius + 2 * sourceStep; y += sourceStep ) {
00041             if ( source.isBelong(x, y) ) {
00042                 for ( double phi = 0.; phi < 2 * TMath::Pi() - (TMath::Pi() / 3600.); phi +=
(TMath::Pi() / 1800.) ) {
00043                     for ( double theta = (TMath::Pi() / 2.); theta < TMath::Pi() - (TMath::Pi() /
3600.); theta += (TMath::Pi() / 1800.) ) {
00044                         totCount++;
00045                         if ( upperDisk.isBelong(x - upperDisk.coordZ * cos(phi) * tan(theta), y -
upperDisk.coordZ * sin(phi) * tan(theta)) ) {
00046                             if ( lowerDisk.isBelong(x - lowerDisk.coordZ * cos(phi) * tan(theta), y -
lowerDisk.coordZ * sin(phi) * tan(theta)) ) {
00047                                 if ( detector.isBelong(x - detector.coordZ * cos(phi) * tan(theta), y
- detector.coordZ * sin(phi) * tan(theta)) ) {
00048                                     nCount++;
00049                                 }
00050                             }
00051                         }
00052                     }
00053                 }
00054                 if ( upperDisk.isBelong(x, y) ) {
00055                     nCount++;
00056                 }
00057                 totCount++;
00058                 effAngle += static_cast<double>(nCount) / totCount;
00059                 nCount = 0;
00060                 totCount = 0;
00061                 iPoint++;
00062             }
00063         }
00064     }
00065     // effAngle = static_cast<double>(nCount) / totCount;
00066     effAngle /= iPoint;
00067     return effAngle;
00068 }

```

8.188 /home/ychoi/ATOM/trashcan/TClusterAnalyser.cpp File Reference

```
#include "TClusterAnalyser.h"
```

Macros

- `#define __TCLUSTERANALYSER_HEADERS__`

8.188.1 Macro Definition Documentation

8.188.1.1 __TCLUSTERANALYSER_HEADERS__

```
#define __TCLUSTERANALYSER_HEADERS__
```

Definition at line 1 of file [TClusterAnalyser.cpp](#).

8.189 TClusterAnalyser.cpp

[Go to the documentation of this file.](#)

```

00001 #define __TCLUSTERANALYSER_HEADERS__
00002
00003 #include "TClusterAnalyser.h"
00009 TClusterAnalyser::TClusterAnalyser(const TAnalyser& analyser) : TAnalyser(analyser),
    fBits(kNotDeleted) {
00010     std::clog << "TClusterAnalyser object is armed." << std::endl;
00011 }
00012
00013 TClusterAnalyser::TClusterAnalyser(const TClusterAnalyser& copy) : TAnalyser(copy) {
00014     std::clog << "Copy TClusterAnalyser object is armed." << std::endl;
00015 }
00020 TClusterAnalyser::~TClusterAnalyser() {
00021     // for ( const auto& pair : mClustermaps ) {
00022     //     delete pair.second;
00023     // }
00024     // for ( const auto& pair : mClustersizes ) {
00025     //     delete pair.second;
00026     // }
00027     // for ( const auto& pair : mClusterDataWithShape ) {
00028     //     for ( const auto& pair2 : pair.second ) {
00029     //         for ( const TCluster* cluster : pair2.second ) {
00030     //             delete cluster;
00031     //         }
00032     //     }
00033     // }
00034     std::clog << "TClusterAnalyser object is terminated." << std::endl;
00035 }
00036
00037 // std::vector<int> getClusterSizeRange(const CppConfigDictionary& privateProperty) {
00038 //     std::vector<int> clusterSizeRange;
00039 //     if ( privateProperty.hasKey("interest_size") ) {
00040 //         for ( const std::string& rangeStr :
00041 //             privateProperty.getSubConfig("interest_size").getValueList() ) {
00042 //             if ( rangeStr.find('.') != std::string::npos ) {
00043 //                 for ( int i = stoi(rangeStr.substr(0, rangeStr.find('.'))); i <
00044 //                     stoi(rangeStr.substr(rangeStr.find('.') + 3)) + 1; i++ ) {
00045 //                     clusterSizeRange.push_back(i);
00046 //                 }
00047 //             } else {
00048 //                 clusterSizeRange.push_back(stoi(rangeStr));
00049 //             }
00050 //         }
00051 //     } else {
00052 //         for ( int i = 0; i < 100; i++ ) {
00053 //             clusterSizeRange.push_back(i);
00054 //         }
00055 //     }
00056 //     return clusterSizeRange;
00057 // }
00058 // /**
00059 //  * @brief Generalized function to draw clustermap
00060 //  *
00061 //  * @param config Draw configuration for map title, directory and filename
00062 //  * @param clusters Dataset to draw
00063 //  * @return const TH2*
00064 //  */
00065 // TH2D* TClusterAnalyser::getClusterPlot(const CppConfigDictionary& config, const
    std::vector<TCluster*>& clusters) {
00066 //     // Static variable for numbering
00067 //     static int iMap = 0;
00068 //     // Allocate a 2d histogram.
00069 //     std::string plotTitle = "";
00070 //     if ( config.hasKey("title") ) {
00071 //         plotTitle += config.find("title");
00072 //     }
00073 //     if ( config.hasKey("x_title") ) {
00074 //         plotTitle += "; " + config.find("x_title");
00075 //     } else {
00076 //         plotTitle += "; ";
00077 //     }
00078 //     if ( config.hasKey("y_title") ) {
00079 //         plotTitle += "; " + config.find("y_title");
00080 //     } else {
00081 //         plotTitle += "; ";
00082 //     }
00083 //     TH2D* map = new TH2D(Form("map%d", iMap), static_cast<TString>(plotTitle), 1024, 0, 1024, 512, 0,
    512);
00084 //     // Fill data
00085 //     std::vector<int> interestSizeSet = getClusterSizeRange(config);
00086 //     for ( const TCluster* cluster : clusters ) {

```

```

00087 //         if ( !interestSizeSet.empty() ) {
00088 //             if ( std::find(interestSizeSet.begin(), interestSizeSet.end(), cluster->getSize()) !=
interestSizeSet.end() ) {
00089 //                 map->Fill(cluster->getCenter().first, cluster->getCenter().second);
00090
00091 //             }
00092 //         } else {
00093 //             map->Fill(cluster->getCenter().first, cluster->getCenter().second);
00094 //         }
00095 //     }
00096 // // Canvas setting
00097 // TCanvas* canvas = new TCanvas(Form("mapCan%d", iMap), "", 2500, 1000);
00098 // canvas->SetMargin(.07, .35, .12, .08);
00099 // map->GetXaxis()->SetTitleOffset(1.4);
00100 // map->GetXaxis()->SetLabelOffset(0.003);
00101 // // Find directory for saving clustermap. If it doesn't exist, then make the directory with mother
directories.
00102 // std::filesystem::path filePath(config.find("output_path"));
00103 // filePath /= config.find("subdirectory");
00104 // std::filesystem::create_directories(filePath);
00105
00106 // // Draw plot with options
00107 // if ( config.hasKey("options") ) {
00108 //     for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
00109 //         map->Draw(static_cast<TString>(optionName));
00110 //         mExpSettingLegend->Draw("SAME");
00111 //         std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
00112 //         if ( config.hasKey("extension") ) {
00113 //             file.replace_extension(config.find("extension"));
00114 //         } else {
00115 //             file.replace_extension("png");
00116 //         }
00117 //         canvas->SaveAs(static_cast<TString>(file));
00118 //     }
00119 // } else {
00120 //     map->Draw();
00121 //     mExpSettingLegend->Draw("SAME");
00122 //     std::filesystem::path file = filePath / (config.find("filename"));
00123 //     if ( config.hasKey("extension") ) {
00124 //         file.replace_extension(config.find("extension"));
00125 //     } else {
00126 //         file.replace_extension("png");
00127 //     }
00128 //     canvas->SaveAs(static_cast<TString>(file));
00129 // }
00130 // // Delete canvas;
00131 // delete canvas;
00132 // iMap++;
00133 // return map;
00134 // }
00135 // /**
00136 //  * @brief
00137 //  *
00138 //  * @param config
00139 //  * @param clusters
00140 //  * @return TH1D*
00141 //  */
00142 // TH1D* TClusterAnalyser::getClustersizePlot(const CppConfigDictionary& config, const
std::vector<TCluster*>& clusters) {
00143 //     static int iDistribution = 0;
00144 //     TString distName = Form("distribution%d", iDistribution);
00145 //     TString distTitle = "";
00146 //     distTitle += config.hasKey("title") ? config.find("title") : "";
00147 //     distTitle += config.hasKey("x_title") ? "; " + config.find("x_title") : ";";
00148 //     distTitle += config.hasKey("y_title") ? "; " + config.find("y_title") : ";";
00149 //     Int_t nBins = 0;
00150 //     Float_t maxBin = 0;
00151 //     Float_t minBin = 0;
00152 //     if ( config.hasKey("distribution_info") ) {
00153 //         nBins = config.getSubConfig("distribution_info").hasKey("nbins") ?
stoi(config.getSubConfig("distribution_info").find("nbins")) : 80;
00154 //         maxBin = config.getSubConfig("distribution_info").hasKey("max") ?
stoi(config.getSubConfig("distribution_info").find("max")) + .5 : 80.5;
00155 //         minBin = config.getSubConfig("distribution_info").hasKey("min") ?
stoi(config.getSubConfig("distribution_info").find("min")) - .5 : .5;
00156 //     } else {
00157 //         nBins = 80;
00158 //         maxBin = 80.5;
00159 //         minBin = 0.5;
00160 //     }
00161 //     TH1D* distribution = new TH1D(distName, distTitle, nBins, minBin, maxBin);
00162 //     for ( const TCluster* cluster : clusters ) {
00163 //         distribution->Fill(cluster->getSize()); // Fill clustersize to clustersize distribution
00164 //     }
00165 //     TCanvas* canvas = new TCanvas(Form("distributionCan%d", iDistribution), "", 2500, 1000);
00166 //     canvas->SetMargin(.07, .28, .12, .08);
00167 //     distribution->GetXaxis()->SetTitleOffset(1.4);

```

```

00168 // distribution->GetXAxis()->SetLabelOffset(0.003);
00169
00170 // std::filesystem::path filePath(config.find("output_path"));
00171 // filePath /= config.find("subdirectory");
00172 // std::filesystem::create_directories(filePath);
00173
00174 // if ( config.hasKey("options") ) {
00175 //     for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
00176 //         if ( optionName == "logy" ) {
00177 //             distribution->Draw();
00178 //             mExpSettingLegend->Draw("SAME");
00179 //             canvas->SetLogy();
00180 //             std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
00181 //             if ( config.hasKey("extension") ) {
00182 //                 file.replace_extension(config.find("extension"));
00183 //             } else {
00184 //                 file.replace_extension("png");
00185 //             }
00186 //             canvas->SaveAs(static_cast<TString>(file));
00187 //         } else if ( optionName == "basic" ) {
00188 //             distribution->Draw();
00189 //             mExpSettingLegend->Draw("SAME");
00190 //             std::filesystem::path file = filePath / config.find("filename");
00191 //             if ( config.hasKey("extension") ) {
00192 //                 file.replace_extension(config.find("extension"));
00193 //             } else {
00194 //                 file.replace_extension("png");
00195 //             }
00196 //             canvas->SetLogy(0);
00197 //             canvas->SaveAs(static_cast<TString>(file));
00198 //         }
00199 //     }
00200 // } else {
00201 //     distribution->Draw();
00202 //     mExpSettingLegend->Draw("SAME");
00203 //     std::filesystem::path file = filePath / (config.find("filename"));
00204 //     if ( config.hasKey("extension") ) {
00205 //         file.replace_extension(config.find("extension"));
00206 //     } else {
00207 //         file.replace_extension("png");
00208 //     }
00209 //     canvas->SaveAs(static_cast<TString>(file));
00210 // }
00211 // delete canvas;
00212 // iDistribution++;
00213 // return distribution;
00214 // }
00215
00216 // void TClusterAnalyser::saveClustermap(std::string typeName, const CppConfigDictionary& config) {
00217 //     std::clog << "Generating \033[1;32mClustermap\033[1;0m..." << std::endl;
00218 //     if ( !mClustermaps.count(typeName) ) {
00219 //         mClustermaps.insert_or_assign(typeName, getClusterPlot(config,
00220 //             mExpData.find(typeName)->second->getClusters()));
00221 //         if ( mIsOutputGraph ) {
00222 //             mDirectorySet.find(std::string(typeName))->second->cd();
00223 //             mClustermaps.find(std::string(typeName))->second->Write("clustermap");
00224 //             mOutputFile->cd();
00225 //         }
00226 //     } else {
00227 //         getClusterPlot(config, mExpData.find(typeName)->second->getClusters());
00228 //     }
00229 // }
00230 // void TClusterAnalyser::saveClustersize(std::string typeName, const CppConfigDictionary& config) {
00231 //     std::clog << "Generating \033[1;32mCluster Size Distribution\033[1;0m..." << std::endl;
00232 //     if ( !mClustersizes.count(typeName) ) {
00233 //         mClustersizes.insert_or_assign(typeName, getClustersizePlot(config,
00234 //             mExpData.find(typeName)->second->getClusters()));
00235 //         if ( mIsOutputGraph ) {
00236 //             mDirectorySet.find(std::string(typeName))->second->cd();
00237 //             mClustersizes.find(std::string(typeName))->second->Write("clustersize");
00238 //             mOutputFile->cd();
00239 //         }
00240 //     } else {
00241 //         getClustersizePlot(config, mExpData.find(typeName)->second->getClusters());
00242 //     }
00243 // }
00244 // void TClusterAnalyser::setClusterDataWithShape(const std::vector<int>& clusterSizeRange) {
00245 //     for ( const int clusterSize : clusterSizeRange ) {
00246 //         std::vector<TCluster*> clustersWithShape;
00247 //     }
00248 // }
00249
00250
00251
00252

```

```

00253 // void TClusterAnalyser::saveHitmapByClustersize(const CppConfigDictionary& config) {
00254 // // std::filesystem::create_directories(mSavePath / "hitmap_by_cluster_size");
00255 // // for ( int clusterSize = 1; clusterSize < 80; clusterSize++ ) {
00256 // //   TH2D* clusterHitmap = new TH2D(Form("hitmap%d", clusterSize), Form("Hitmap of cluster size
    %d", clusterSize), 1024, 0, 1024, 512, 0, 512);
00257 // //   TH2D* clusterClustermap = new TH2D(Form("clustermap%d", clusterSize), Form("Clustermap of
    cluster size %d", clusterSize), 1024, 0, 1024, 512, 0, 512);
00258 // //   TH1D* clusterBinFire = new TH1D(Form("fired%d", clusterSize), Form("The number of fired of
    each bins in cluster size %d", clusterSize), 50, 0, 50);
00259 // //   for ( const TCluster* cluster : mExpData->getClusters() ) {
00260 // //     if ( cluster->getSize() == clusterSize ) {
00261 // //       for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
00262 // //         clusterHitmap->Fill(pixel.first, pixel.second);
00263 // //       }
00264 // //       clusterClustermap->Fill(cluster->getCenter().first, cluster->getCenter().second);
00265 // //     }
00266 // //   }
00267 // //   for ( int iRow = 0; iRow < 1024; iRow++ ) {
00268 // //     for ( int iColumn = 0; iColumn < 512; iColumn++ ) {
00269 // //       if ( clusterHitmap->GetBinContent(iRow, iColumn) != 0 ) {
00270 // //         clusterBinFire->Fill(clusterHitmap->GetBinContent(iRow, iColumn));
00271 // //       }
00272 // //     }
00273 // //   }
00274 // //   if ( clusterBinFire->GetEntries() != 0 ) {
00275 // //     TCanvas* hcanvas = new TCanvas(Form("hcan%d", clusterSize), "", 2000, 1000);
00276 // //     clusterHitmap->Draw();
00277 // //     hcanvas->SaveAs(static_cast<TString>(mSavePath / "hitmap_by_cluster_size" /
    "hitmap_cs_" + std::to_string(clusterSize) + ".png"));
00278 // //     TCanvas* ccanvas = new TCanvas(Form("ccan%d", clusterSize), "", 2000, 1000);
00279 // //     clusterClustermap->Draw();
00280 // //     ccanvas->SaveAs(static_cast<TString>(mSavePath / "hitmap_by_cluster_size" /
    "clustermap_cs_" + std::to_string(clusterSize) + ".png"));
00281 // //     TCanvas* dcanvas = new TCanvas(Form("dcan%d", clusterSize), "", 1000, 1000);
00282 // //     clusterBinFire->Draw();
00283 // //     dcanvas->SetLogy();
00284 // //     dcanvas->SaveAs(static_cast<TString>(mSavePath / "hitmap_by_cluster_size" /
    "fire_distribution_cs_" + std::to_string(clusterSize) + ".png"));
00285
00286 // //   }
00287 // // }
00288 // }

```

8.190 /home/ychoi/ATOM/trashcan/TClusterAnalyser.h File Reference

Control cluster analysis process and save plots.

```

#include <vector>
#include "TAnalyser.h"

```

Classes

- class [TClusterAnalyser](#)
Communicating execute file for controlling cluster research.

8.190.1 Detailed Description

Control cluster analysis process and save plots.

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

2024-04-09

Copyright

Copyright (c) 2024

Definition in file [TClusterAnalyser.h](#).

8.191 TClusterAnalyser.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef __TCLUSTERANALYSER__
00013 #define __TCLUSTERANALYSER__
00014
00015 #ifdef __TCLUSTERANALYSER_HEADERS__
00016 #include <iostream>
00017
00018 #include "TFile.h"
00019 #include "TDirectory.h"
00020 #include "TCanvas.h"
00021 #include "TPaveText.h"
00022 #include "TH1D.h"
00023 #include "TH2D.h"
00024
00025 #include "CppConfigFile.h"
00026 #include "cpptqdm.h"
00027
00028 #include "TCluster.h"
00029 #include "TExperimentData.h"
00030 #endif
00031
00032 #include <vector>
00033
00034 #include "TAnalyser.h"
00035
00036 class TH1D;
00037 class TH2D;
00038
00039 class Configurable;
00040 class TCluster;
00041
00052 class TClusterAnalyser : public TAnalyser {
00053 protected:
00054     // std::unordered_map<std::string, TH2D*> mClustermaps;
00055     // std::unordered_map<std::string, TH1D*> mClustersizes;
00056     // std::unordered_map<std::string, std::unordered_map<int, std::vector<TCluster*>>
00057     mClusterDataWithShape;
00058 public:
00059     //Constructor
00060     TClusterAnalyser() = default;
00061     TClusterAnalyser(const TAnalyser& analyser);
00062     TClusterAnalyser(const TClusterAnalyser& copy);
00063     ~TClusterAnalyser();
00064
00065     TH2D* getClusterPlot(const CppConfigDictionary& config, const std::vector<TCluster*>& clusters);
00066     TH1D* getClustersizePlot(const CppConfigDictionary& config, const std::vector<TCluster*>&
00067     clusters);
00068     void setClusterDataWithShape(const std::vector<int>& clusterSizeRange);
00069
00070     void saveClustermap(std::string typeName, const CppConfigDictionary& config);
00071     void saveClustersize(std::string typeName, const CppConfigDictionary& config);
00072     void saveHitmapByClustersize(const CppConfigDictionary& config);
00073
00074 private:
00075     unsigned int fBits;
00076 public:

```

```
00075     enum {
00076         kNotDeleted = 0x02000000
00077     };
00078     bool IsDestructed() const { return !TestBit(kNotDeleted); }
00079     bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
00080 };
00081
00082 #endif
```

8.192 /home/ychoi/ATOM/trashcan/TClusterShapeAnalyser.cpp File Reference

Tools for analysing shape property of cluster.

```
#include "TClusterShapeAnalyser.h"
```

Macros

- `#define __TCLUSTERSHAPEANALYSER_HEADER__`

Functions

- `int calNIncludePixel (const TMatrix2D< int > *matrix)`
- `double calRatioOfRadius (const TMatrix2D< int > *matrix)`

8.192.1 Detailed Description

Tools for analysing shape property of cluster.

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

08-05-2024

Copyright

Copyright (c) 2024

Definition in file [TClusterShapeAnalyser.cpp](#).

8.192.2 Macro Definition Documentation

8.192.2.1 __TCLUSTERSHAPEANALYSER_HEADER__

```
#define __TCLUSTERSHAPEANALYSER_HEADER__
```

Definition at line 11 of file [TClusterShapeAnalyser.cpp](#).

8.192.3 Function Documentation

8.192.3.1 calNIncludePixel()

```
int calNIncludePixel (
    const TMatrix2D< int > * matrix )
```

Definition at line 161 of file [TClusterShapeAnalyser.cpp](#).

8.192.3.2 calRatioOfRadius()

```
double calRatioOfRadius (
    const TMatrix2D< int > * matrix )
```

Definition at line 193 of file [TClusterShapeAnalyser.cpp](#).

8.193 TClusterShapeAnalyser.cpp

[Go to the documentation of this file.](#)

```
00001
00011 #define __TCLUSTERSHAPEANALYSER_HEADER__
00012 #include "TClusterShapeAnalyser.h"
00013
00019 TClusterShapeAnalyser::TClusterShapeAnalyser(const TClusterAnalyser& analyser) :
    TClusterAnalyser(analyser), fBits(kNotDeleted) {
00020     // Print out a log
00021     std::clog << "TClusterShapeAnalyser object is armed." << std::endl;
00022 }
00023
00028 TClusterShapeAnalyser::~TClusterShapeAnalyser() {
00029     // Destroy objects of TClusterShape if it isn't destructed.
00030     for ( const auto& key : mClusterShapeSet ) {
00031         for ( auto& ele : key.second ) {
00032             if ( !ele->IsDestructed() ) {
00033                 delete ele;
00034             }
00035         }
00036     }
00037     // Print out a log
00038     std::clog << "TClusterShapeAnalyser object is terminated." << std::endl;
00039 }
00040
00047 void TClusterShapeAnalyser::doShaping(std::string_view typeName, const std::vector<int>&
    clusterSizeRange) {
00048     // The number of total shape and the maximum number of shapes of each cluster sizes.
00049     int nTotalShape = 0;
00050     int maxMode = 0;
00051     // The array of objects of TClusterShape class.
00052     std::vector<TClusterShape*> clusterShapes;
00053     // This stores and extracts clusters shape informations.
00054     for ( const int clusterSize : clusterSizeRange ) {
00055         if ( mClusterSizes.find(std::string(typeName))->second->GetBinContent(clusterSize) != 0 ) {
00056             // This gets cluster bunch from TClusterDivideData object which classifies clusters by
                their size.
```

```

00057         const std::vector<TCluster*> divideData =
mDivideData.find(std::string(typeName))>second->getClusterOfSize(clusterSize);
00058         // This inserts cluster size information and clusters to TClusterShape objects.
00059         TClusterShape* clusterShape = new TClusterShape(clusterSize, divideData);
00060         // This extracts shape informations.
00061         clusterShape->identifyShapes();
00062         // This sorts shapes according to its size of long axis.
00063         clusterShape->sortShapes();
00064         //
00065         clusterShapes.push_back(clusterShape);
00066         nTotalShape += clusterShape->getClusterShapeInfos().size();
00067         maxMode = std::max(maxMode,
static_cast<int>(clusterShape->getClusterShapeInfos().size()));
00068     }
00069 }
00070 mClusterShapeSet.insert_or_assign(std::string(typeName), clusterShapes);
00071 mNTotalShapeSet.insert_or_assign(std::string(typeName), nTotalShape);
00072 mMaxModeSet.insert_or_assign(std::string(typeName), maxMode);
00073 }
00080 void TClusterShapeAnalyser::saveIndividualShapes(std::string_view typeName, const CppConfigDictionary
config) {
00081     // Print log message
00082     std::clog << "Generating " << "\033[1;32m" << "individual shapes" << "\033[1;0m" << "..." << std::endl;
00083
00084     // Get the total number of shapes for progress bar.
00085     int nTotalShape = mNTotalShapeSet.find(std::string(typeName))>second;
00086
00087     // Get save path from config file.
00088     std::filesystem::path filePath(config.find("output_path"));
00089     filePath /= config.find("subdirectory");
00090     std::filesystem::create_directories(filePath);
00091
00092     // Init progress bar
00093     ProgressBar pBar(nTotalShape);
00094
00095     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))>second ) {
00096         int clusterSize = clusterShape->getClusterSize(); // The cluster size of shape set.
00097         int iClusterShape = 0; // The i-th cluster shape
00098         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00099             pBar.printProgress(); // Print progress bar.
00100
00101             int shapeWidth = shapeInfo.mClusterMatrix->getNRow(); // Shape width of matrix
00102             int shapeHeight = shapeInfo.mClusterMatrix->getNColumn(); // Shape height of matrix
00103
00104             // Initialize canvas
00105             TCanvas* canvas = new TCanvas(Form("%d_%d.png", clusterSize, iClusterShape), "",
(shapeWidth + 2) * 500, (shapeHeight + 2) * 500);
00106             // Set canvas margin
00107             canvas->SetMargin(0., 0., 0., 0.);
00108             // Draw shape on canvas
00109             shapeInfo.mClusterMap->Draw();
00110             // Fill colour
00111             shapeInfo.mClusterMap->SetDrawOption("COLZ");
00112             // Remove histogram axis
00113             shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
00114             shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
00115
00116             // Fill line for separating pixels
00117             TLine* line = new TLine();
00118             line->SetLineColorAlpha(kRed, 6. / 8);
00119             for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
00120                 for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
00121                     if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
00122                         double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
00123                         double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
00124                         double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
00125                         double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
00126                         line->DrawLine(xlow, ylow, xup, ylow); // Bottom
00127                         line->DrawLine(xlow, yup, xup, yup); // Top
00128                         line->DrawLine(xlow, ylow, xlow, yup); // Left
00129                         line->DrawLine(xup, ylow, xup, yup); // Right
00130                     }
00131                 }
00132             }
00133
00134             // Set title of shape
00135             TText* title = new TText(.5, 1. - .5 / (shapeHeight + 2), Form("%d-th Cluster Shape of
Cluster Size %d", iClusterShape, clusterSize));
00136             title->SetTextAlign(23);
00137             title->SetNDC();
00138             title->Draw();
00139
00140             // Save shape figure
00141             std::filesystem::path file = filePath / (config.find("filename") + "_" +
std::to_string(clusterSize) + "_" + std::to_string(iClusterShape));
00142             if ( config.hasKey("extension") ) {
00143                 file.replace_extension(config.find("extension"));

```



```

00144         } else {
00145             file.replace_extension("png");
00146         }
00147         canvas->SaveAs(static_cast<TString>(file));
00148
00149         iClusterShape++;
00150
00151         delete title;
00152         title = nullptr;
00153         delete line;
00154         line = nullptr;
00155         delete canvas;
00156         canvas = nullptr;
00157     }
00158 }
00159 }
00160
00161 int calNIncludePixel(const TMatrix2D<int>* matrix) {
00162     int centreX = 0, centreY = 0;
00163     int clusterSize = 0;
00164     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00165         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00166             if ( matrix->getElement(pixelX, pixelY) == 1 ) {
00167                 centreX += pixelX * 2;
00168                 centreY += pixelY * 2;
00169                 clusterSize++;
00170             }
00171         }
00172     }
00173     int radiusSquare = 0;
00174     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00175         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00176             if ( matrix->getElement(pixelX, pixelY) == 1 ) {
00177                 int distance = pow(abs(2 * pixelX * clusterSize - centreX) + clusterSize, 2) +
00178                     pow(abs(2 * pixelY * clusterSize - centreY) + clusterSize, 2);
00179                 radiusSquare = std::max(radiusSquare, distance);
00180             }
00181         }
00182     }
00183     int count = 0;
00184     for ( int x = floor((centreX - sqrt(radiusSquare)) / (2 * clusterSize)); x < ceil((centreX +
00185         sqrt(radiusSquare)) / (2 * clusterSize)) + 1; x++ ) {
00186         for ( int y = floor((centreY - sqrt(radiusSquare)) / (2 * clusterSize)); y < ceil((centreY +
00187             sqrt(radiusSquare)) / (2 * clusterSize)) + 1; y++ ) {
00188             if ( static_cast<int>(std::pow(abs(2 * x * clusterSize - centreX) + clusterSize, 2)) +
00189                 static_cast<int>(std::pow(abs(2 * y * clusterSize - centreY) + clusterSize, 2)) <= radiusSquare ) {
00190                 count++;
00191             }
00192         }
00193     }
00194     return count;
00195 }
00196
00197 double calRatioOfRadius(const TMatrix2D<int>* matrix) {
00198     int centreX = 0, centreY = 0;
00199     int clusterSize = 0;
00200     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00201         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00202             if ( matrix->getElement(pixelX, pixelY) == 1 ) {
00203                 centreX += pixelX * 2;
00204                 centreY += pixelY * 2;
00205                 clusterSize++;
00206             }
00207         }
00208     }
00209     int longRadiusSquare = 0;
00210     int shortRadiusSquare = 100000000;
00211     for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
00212         for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
00213             bool hasValue = matrix->getElement(pixelX, pixelY) == 1;
00214             bool isWorldBorder = pixelX == 0 || pixelY == 0 || pixelX == matrix->getNRow() - 1 ||
00215                 pixelY == matrix->getNColumn() - 1;
00216             bool isBorder = isWorldBorder ? true : matrix->getElement(pixelX - 1, pixelY) == 0 ||
00217                 matrix->getElement(pixelX, pixelY - 1) == 0 || matrix->getElement(pixelX + 1, pixelY) == 0 ||
00218                 matrix->getElement(pixelX, pixelY + 1) == 0;
00219             if ( hasValue ) {
00220                 int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY *
00221                     clusterSize - centreY, 2);
00222                 longRadiusSquare = std::max(longRadiusSquare, distance);
00223             }
00224             if ( (hasValue && isWorldBorder) || (hasValue && !isWorldBorder && isBorder) ) {
00225                 int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY *
00226                     clusterSize - centreY, 2);
00227                 shortRadiusSquare = std::min(shortRadiusSquare, distance);
00228             }
00229         }
00230     }
00231 }

```

```

00222     if ( matrix->getNRow() == 1 || matrix->getNColumn() == 1 ) {
00223         shortRadiusSquare = 0;
00224     }
00225     return sqrt(static_cast<double>(shortRadiusSquare) / longRadiusSquare);
00226 }
00227
00234 void TClusterShapeAnalyser::saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary
config) {
00235     // Print out a log
00236     std::cout << "Generating " << "\033[1;32m" << "Informations of shapes with same size" << "\033[1;0m" <<
    "... " << std::endl;
00237
00238     // Setting output file path
00239     std::filesystem::path filePath(config.find("output_path"));
00240     filePath /= config.find("subdirectory");
00241     // The creation of directories of output path
00242     std::filesystem::create_directories(filePath);
00243     // Call configs. First element is config name and second one is config values.
00244     std::unordered_map<std::string, CppConfigDictionary> plotConfigList =
    config.getSubConfig("ratio_distribution").getSubConfigSetWithName();
00245     // Call histograms
00246     std::unordered_map<std::string, TH1D*> distributionSet;
00247     // Set the information of histograms from config file
00248     for ( const auto& plotConfig : plotConfigList ) {
00249         // Plot name is config name. It isn't value of "name" key.
00250         std::string plotName = plotConfig.second.find("name");
00251         // Set plot title and x, y label.
00252         TString plotTitle = plotConfig.second.hasKey("title") ? plotConfig.second.find("title") : "";
00253         TString plotXTitle = plotConfig.second.hasKey("x_title") ? plotConfig.second.find("x_title") :
    "";
00254         TString plotYTitle = plotConfig.second.hasKey("y_title") ? plotConfig.second.find("y_title") :
    "";
00255         // Set number of bin and min and max of x-direction range.
00256         Int_t plotNBin = plotConfig.second.hasKey("n_bin") ? stoi(plotConfig.second.find("n_bin")) :
    100;
00257         Float_t plotXMin = plotConfig.second.hasKey("x_min") ? stof(plotConfig.second.find("x_min")) :
    0.;
00258         Float_t plotXMax = plotConfig.second.hasKey("x_max") ? stof(plotConfig.second.find("x_max")) :
    1.;
00259         // Add histograms to map.
00260         distributionSet.insert_or_assign(plotConfig.first, new TH1D(static_cast<TString>(plotName),
    plotTitle + " ; " + plotXTitle + " ; " + plotYTitle, plotNBin, plotXMin, plotXMax));
00261     }
00262
00263     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
00264     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00265         // Get cluster size.
00266         int clusterSize = clusterShape->getClusterSize();
00267         if ( clusterSize < stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_min"))
    || clusterSize > stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_max")) ) {
00268             continue;
00269         }
00270         // Initialize canvas.
00271         TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
00272         // int binNum = clusterShape->getClusterShapeInfos().size();
00273         TGraph* areaRatioGraph = new TGraph();
00274         TGraph* radiusRatioGraph = new TGraph();
00275
00276         int iShape = 0;
00277         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00278             // pBar.printProgress();
00279             for ( const auto& plotConfig : plotConfigList ) {
00280                 if ( plotConfig.second.find("name") == "area_ratio" ) {
00281                     distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) /
    calNIncludePixel(shapeInfo.mClusterMatrix));
00282                 }
00283                 if ( plotConfig.second.find("name") == "area_ratio_with_entry" ) {
00284                     distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) /
    calNIncludePixel(shapeInfo.mClusterMatrix), shapeInfo.mEntry);
00285                 }
00286                 if ( plotConfig.second.find("name") == "radius_ratio" ) {
00287                     distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix));
00288                 }
00289                 if ( plotConfig.second.find("name") == "radius_ratio_with_entry" ) {
00290                     distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix),
    shapeInfo.mEntry);
00291                 }
00292             }
00293         }
00294         areaRatioGraph->SetPoint(iShape, iShape, static_cast<double>(clusterSize) /
    calNIncludePixel(shapeInfo.mClusterMatrix));
00295         radiusRatioGraph->SetPoint(iShape, iShape, calRatioOfRadius(shapeInfo.mClusterMatrix));
00296     }

```

```

00297         // std::cout << clusterSize << "\t" << iShape << "\t" << static_cast<double>(clusterSize) /
        calNIncludePixel(shapeInfo.mClusterMatrix) << "\t" << calRatioOfRadius(shapeInfo.mClusterMatrix) << "\t"
        << shapeInfo.mEntry << std::endl;
00298         iShape++;
00299     }
00300     TF1* line1 = new TF1("line1", "0.8", 0, iShape);
00301     TF1* line2 = new TF1("line2", "0.5", 0, iShape);
00302
00303     areaRatioGraph->SetTitle(Form("Informations for cluster shapes in cluster size %d",
        clusterSize));
00304     areaRatioGraph->GetXaxis()->SetTitle("i-th cluster shape");
00305     areaRatioGraph->GetXaxis()->SetTitleOffset(1.4);
00306     areaRatioGraph->GetXaxis()->SetLabelOffset(0.003);
00307     areaRatioGraph->GetYaxis()->SetTitle("Ratio");
00308     areaRatioGraph->GetYaxis()->SetTitleOffset(0.7);
00309     areaRatioGraph->SetMarkerStyle(45);
00310     areaRatioGraph->SetMarkerSize(4);
00311     areaRatioGraph->SetMarkerColor(kRed);
00312     areaRatioGraph->SetMaximum(1);
00313     areaRatioGraph->SetMinimum(0);
00314     areaRatioGraph->Draw("AP");
00315     radiusRatioGraph->SetMarkerStyle(41);
00316     radiusRatioGraph->SetMarkerSize(4);
00317     radiusRatioGraph->SetMarkerColor(kBlue);
00318     radiusRatioGraph->Draw("P");
00319     mExpSettingLegend->Draw("SAME");
00320     line1->SetLineColor(kPink + 2);
00321     // line1->Draw("SAME");
00322     line2->SetLineColor(kCyan - 7);
00323     // line2->Draw("SAME");
00324     TLegend* legend = new TLegend(.78, .7, .98, .92);
00325     legend->SetHeader("Ratios", "c");
00326     legend->AddEntry(areaRatioGraph, "Pixels / Full pixels", "p");
00327     legend->AddEntry(radiusRatioGraph, "Short radius / Long radius", "p");
00328     legend->Draw();
00329     canvas->SetMargin(.07, .28, .12, .08);
00330     std::filesystem::path file = filePath;
00331
00332     file /= config.getSubConfig("graphs").find("filename") + "_" + std::to_string(clusterSize);
00333     file.replace_extension(config.find("extension"));
00334
00335     canvas->SaveAs(static_cast<TString>(file));
00336
00337     delete line1;
00338     delete line2;
00339     delete areaRatioGraph;
00340     delete radiusRatioGraph;
00341     delete canvas;
00342 }
00343
00344 for ( const auto& plotConfig : plotConfigList ) {
00345     TString canvasName = "can" + plotConfig.first;
00346     Int_t canvasWidth = plotConfig.second.hasKey("canvas_width") ?
00347         stoi(plotConfig.second.find("canvas_width")) : 500;
00347     Int_t canvasHeight = plotConfig.second.hasKey("canvas_height") ?
00348         stoi(plotConfig.second.find("canvas_height")) : 500;
00348     TCanvas* canvas = new TCanvas(canvasName, "", canvasWidth, canvasHeight);
00349
00350     distributionSet.find(plotConfig.first)->second->SetStats(0);
00351
00352     Int_t y_min = plotConfig.second.hasKey("y_min") ? stoi(plotConfig.second.find("y_min")) : 0;
00353     Int_t y_max = plotConfig.second.hasKey("y_max") ? stoi(plotConfig.second.find("y_max")) :
1000;
00354     distributionSet.find(plotConfig.first)->second->SetMinimum(y_min);
00355     distributionSet.find(plotConfig.first)->second->SetMaximum(y_max);
00356     distributionSet.find(plotConfig.first)->second->Draw();
00357
00358     std::filesystem::path file = filePath;
00359     file /= plotConfig.second.find("filename");
00360     file.replace_extension(config.find("extension"));
00361     canvas->SaveAs(static_cast<TString>(file));
00362
00363     delete canvas;
00364     canvas = nullptr;
00365 }
00366 }
00367
00368 void TClusterShapeAnalyser::saveSameSizeShapes(std::string_view typeName, const CppConfigDictionary
        config) {
00369     std::cout << "Generating " << "\033[1;32m" << "Shapes with same sized" << "\033[1;0m" << "..." <<
        std::endl;
00370
00371     std::filesystem::path filePath(config.find("output_path"));
00372     if ( config.hasKey("subdirectory") ) {
00373         filePath /= config.find("subdirectory");
00374     }
00375     std::filesystem::create_directories(filePath);

```

```

00376
00377     int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;
00378
00379     const int nominalWidth = 200;
00380     const int nominalHeader = 100;
00381     const int nPlotInRow = 10;
00382     int plotsWidth = nPlotInRow * nominalWidth;
00383
00384     ProgressBar pBar(nTotalShape);
00385     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00386         int clusterSize = clusterShape->getClusterSize();
00387         int nClusterShape = clusterShape->getClusterShapeInfos().size();
00388
00389         int plotsHeight = (floor((nClusterShape - 1.) / 10.) + 1.) * nominalWidth;
00390
00391         TCanvas* canvas = new TCanvas(Form("shapes%d", clusterSize), "", plotsWidth, plotsHeight +
nominalHeader);
00392
00393         int iClusterShape = 0;
00394
00395         std::vector<TPad*> padSet;
00396         std::vector<TLine*> lineSet;
00397         std::vector<TText*> textSet;
00398
00399         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00400             pBar.printProgress();
00401
00402             int width = shapeInfo.mClusterMatrix->getNRow() + 2;
00403             int height = shapeInfo.mClusterMatrix->getNColumn() + 2;
00404
00405             double padCenterX = (1. / nPlotInRow) * (.5 + (iClusterShape % 10));
00406             double padCenterY = (static_cast<double>(plotsHeight) / static_cast<double>(plotsHeight +
nominalHeader)) / ((floor((nClusterShape - 1) / static_cast<double>(nPlotInRow))) + 1.) * .5 * (2. *
(floor((nClusterShape - 1) / static_cast<double>(nPlotInRow)) - floor(iClusterShape /
static_cast<double>(nPlotInRow))) + 1.);
00407
00408             double padWidth = (1. / (2. * nPlotInRow));
00409             double padHeight = (static_cast<double>(plotsHeight) / (plotsHeight + nominalHeader)) /
(2. * ((floor(static_cast<double>(nClusterShape) / nPlotInRow) + 1.)));
00410
00411             TPad* pad = new TPad(Form("pad%d", iClusterShape), "", padCenterX - padWidth, padCenterY -
padHeight, padCenterX + padWidth, padCenterY + padHeight);
00412             pad->SetMargin(0., 0., .5 * (1. - static_cast<double>(height) / width), .5 * (1. -
static_cast<double>(height) / width));
00413             pad->Draw();
00414             pad->cd();
00415             shapeInfo.mClusterMap->SetDrawOption("COL");
00416             shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
00417             shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
00418             shapeInfo.mClusterMap->Draw("col");
00419             pad->SetFrameLineWidth(0);
00420             padSet.push_back(pad);
00421
00422             TLine* line = new TLine();
00423             line->SetLineColorAlpha(kRed, 6. / 8);
00424             for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
00425                 for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
00426                     if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
00427                         double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
00428                         double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
00429                         double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
00430                         double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
00431                         line->DrawLine(xlow, ylow, xup, ylow); // Bottom
00432                         line->DrawLine(xlow, yup, xup, yup); // Top
00433                         line->DrawLine(xlow, ylow, xlow, yup); // Left
00434                         line->DrawLine(xup, ylow, xup, yup); // Right
00435                     }
00436                 }
00437             }
00438             lineSet.push_back(line);
00439
00440             TText* numberingText = new TText(padCenterX, padCenterY, Form("%d", iClusterShape));
00441             numberingText->SetNDC();
00442             numberingText->SetTextAlign(22);
00443             numberingText->SetTextSize(.3 * nominalWidth / (plotsHeight + nominalHeader));
00444             numberingText->SetTextColorAlpha(kBlack, 5. / 8.);
00445
00446             canvas->cd();
00447             numberingText->Draw();
00448             textSet.push_back(numberingText);
00449             iClusterShape++;
00450         }
00451         TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight),
Form("Cluster shapes with cluster size %d", clusterSize));
00452         titleText->SetTextSize(.5 * nominalHeader / (plotsHeight + nominalHeader));
00453         titleText->SetTextAlign(22);
00454         titleText->SetNDC();

```

```

00455         titleText->Draw();
00456         std::filesystem::path file = filePath / (config.find("filename") + "_" +
std::to_string(clusterSize));
00457         file.replace_extension(config.find("extension"));
00458         canvas->SaveAs(static_cast<TString>(file));
00459
00460         delete titleText;
00461         titleText = nullptr;
00462
00463         for ( TPad* pad : padSet ) {
00464             delete pad;
00465             pad = nullptr;
00466         }
00467
00468         for ( TLine* line : lineSet ) {
00469             delete line;
00470             line = nullptr;
00471         }
00472
00473         for ( TText* text : textSet ) {
00474             delete text;
00475             text = nullptr;
00476         }
00477
00478         delete canvas;
00479         canvas = nullptr;
00480     }
00481 }
00482
00483 void TClusterShapeAnalyser::saveTotalShapes(std::string_view typeName, const CppConfigDictionary
config) {
00484     // Print log
00485     std::cout << "Generating \033[1;32mTotal shapes\033[1;0m..." << std::endl;
00486
00487     std::filesystem::path filePath(config.find("output_path"));
00488     if ( config.hasKey("subdirectory") ) {
00489         filePath /= config.find("subdirectory");
00490     }
00491     std::filesystem::create_directories(filePath);
00492
00493     int nWidth = mMaxModeSet.find(std::string(typeName))->second;
00494     int nHeight = mClusterShapeSet.find(std::string(typeName))->second.size();
00495
00496     int nominalWidth = 100;
00497     int nominalHeader = 100;
00498
00499     int plotsWidth = nWidth * nominalWidth;
00500     int plotsHeight = nHeight * nominalWidth;
00501     TCanvas* canvas = new TCanvas("tShape", "cluster shape", plotsWidth + nominalHeader, plotsHeight +
nominalHeader);
00502     int iClusterSize = 0;
00503     std::vector<TPad*> padSet;
00504     std::vector<TLine*> lineSet;
00505     std::vector<TText*> textSet;
00506
00507     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
00508     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00509         int clusterSize = clusterShape->getClusterSize();
00510         int iClusterShape = 0;
00511         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00512             pBar.printProgress();
00513             int width = shapeInfo.mClusterMap->GetNbinsX();
00514             int height = shapeInfo.mClusterMap->GetNbinsY();
00515
00516             TPad* pad = new TPad(Form("pad%d_%d", iClusterSize, iClusterShape), "pad", (double)
nominalHeader / (plotsWidth + nominalHeader) + ((double) plotsWidth / (plotsWidth + nominalHeader)) *
((double) iClusterShape / nWidth), ((double) plotsHeight / (plotsHeight + nominalHeader)) * (((double)
nHeight - iClusterSize - 1) / nHeight), (double) nominalHeader / (plotsWidth + nominalHeader) +
((double) plotsWidth / (plotsWidth + nominalHeader)) * (double) (iClusterShape + 1) / nWidth,
((double) plotsHeight / (plotsHeight + nominalHeader)) * (((double) nHeight - iClusterSize) /
nHeight), -1, 1);
00517             padSet.push_back(pad);
00518             pad->Draw();
00519             pad->cd();
00520             pad->SetMargin(0., 0., .5 * (1. - (double) height / width), .5 * (1. - (double) height /
width));
00521
00522             shapeInfo.mClusterMap->Draw();
00523             shapeInfo.mClusterMap->SetDrawOption("COL");
00524             shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
00525             shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
00526             TLine* line = new TLine();
00527             lineSet.push_back(line);
00528             line->SetLineColorAlpha(kRed, 6. / 8);
00529             for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
00530                 for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
00531                     if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {

```

```

00532         double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
00533         double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
00534         double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
00535         double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
00536         line->DrawLine(xlow, ylow, xup, ylow); // Bottom
00537         line->DrawLine(xlow, yup, xup, yup); // Top
00538         line->DrawLine(xlow, ylow, xlow, yup); // Left
00539         line->DrawLine(xup, ylow, xup, yup); // Right
00540     }
00541 }
00542 }
00543     pad->SetFrameLineWidth(0);
00544     canvas->cd();
00545     TText* numberingText = new TText((double) nominalHeader / (plotsWidth + nominalHeader) +
((double) plotsWidth / (plotsWidth + nominalHeader)) * ((double) iClusterShape + .5) / nWidth),
((double) plotsHeight / (plotsHeight + nominalHeader)) * ((double) nHeight - iClusterSize) /
nHeight), Form("%d", shapeInfo.mEntry));
00546     textSet.push_back(numberingText);
00547     numberingText->SetNDC();
00548     numberingText->SetTextAlign(22);
00549     numberingText->SetTextSize(.4 * nominalWidth / (plotsHeight + nominalHeader));
00550     numberingText->SetTextColor(kBlack);
00551     numberingText->Draw();
00552     iClusterShape++;
00553 }
00554     TText* sizeText = new TText((double) nominalHeader * .5 / (nominalHeader + plotsWidth),
((double) plotsHeight / (plotsHeight + nominalHeader)) * ((double) nHeight - iClusterSize - .5) /
nHeight), Form("%d", clusterSize));
00555     textSet.push_back(sizeText);
00556     sizeText->SetNDC();
00557     sizeText->SetTextAlign(22);
00558     sizeText->SetTextSize(.6 * nominalWidth / (plotsHeight + nominalHeader));
00559     sizeText->SetTextColor(kBlack);
00560     sizeText->Draw();
00561     iClusterSize++;
00562 }
00563     TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight),
static_cast<TString>("Total Cluster Shapes"));
00564     titleText->SetTextSize(.8 * nominalHeader / (plotsHeight + nominalHeader));
00565     titleText->SetTextAlign(22);
00566     titleText->SetNDC();
00567     titleText->Draw();
00568
00569     std::filesystem::path file = filePath / config.find("filename");
00570     file.replace_extension(config.find("extension"));
00571     canvas->SaveAs(static_cast<TString>(file));
00572
00573     delete titleText;
00574     titleText = nullptr;
00575
00576     for ( TPad* pad : padSet ) {
00577         delete pad;
00578         pad = nullptr;
00579     }
00580
00581     for ( TLine* line : lineSet ) {
00582         delete line;
00583         line = nullptr;
00584     }
00585
00586     for ( TText* text : textSet ) {
00587         delete text;
00588         text = nullptr;
00589     }
00590
00591     delete canvas;
00592     canvas = nullptr;
00593 }
00594
00595 void TClusterShapeAnalyser::saveSameSizeShapeEntry(std::string_view typeName, const
CppConfigDictionary config) {
00596     std::cout << "Generating \033[1;32mEntry of shapes with same size\033[1;0m..." << std::endl;
00597
00598     std::filesystem::path filePath(config.find("output_path"));
00599     if ( config.HasKey("subdirectory") ) {
00600         filePath /= config.find("subdirectory");
00601     }
00602     std::filesystem::create_directories(filePath);
00603
00604     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
00605     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00606         int clusterSize = clusterShape->getClusterSize();
00607         TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
00608         int binNum = clusterShape->getClusterShapeInfos().size();
00609         TH1D* distribution = new TH1D(Form("shapeEntry%d", clusterSize),
Form(static_cast<TString>("Shape Entry with cluster size %d"), clusterSize), binNum, -.5, binNum -
.5);

```

```

00610         int iShape = 0;
00611         for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
00612             pBar.printProgress();
00613             distribution->Fill(iShape, shapeInfo.mEntry);
00614             iShape++;
00615         }
00616         distribution->GetXaxis()->SetNdivisions(10, 10, 0, true);
00617         distribution->GetXaxis()->SetTitle("i-th cluster size");
00618         distribution->GetXaxis()->SetTitleOffset(1.4);
00619         distribution->GetXaxis()->SetLabelOffset(0.003);
00620         distribution->GetYaxis()->SetTitle("Entry");
00621         distribution->GetYaxis()->SetTitleOffset(0.7);
00622         distribution->Draw("HIST");
00623         mExpSettingLegend->Draw("SAME");
00624         canvas->SetMargin(.07, .28, .12, .08);
00625
00626         std::filesystem::path file = filePath / (config.find("filename") + "_" +
std::to_string(clusterSize));
00627         file.replace_extension(config.find("extension"));
00628         canvas->SaveAs(static_cast<TString>(file));
00629
00630         delete distribution;
00631         delete canvas;
00632     }
00633 }
00634
00635 void TClusterShapeAnalyser::saveTotalShapeEntry(std::string_view typeName, const CppConfigDictionary
config) {
00636     std::cout << "Generating \033[1;32mEntry of total shapes\033[1;0m..." << std::endl;
00637
00638     std::filesystem::path filePath(config.find("output_path"));
00639     if ( config.hasKey("subdirectory") ) {
00640         filePath /= config.find("subdirectory");
00641     }
00642     std::filesystem::create_directories(filePath);
00643
00644     int nXbin = 0;
00645     int nYbin = 0;
00646     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00647         nYbin = std::max(nYbin, static_cast<int>(clusterShape->getClusterShapeInfos().size()));
00648         nXbin = std::max(nXbin, clusterShape->getClusterSize());
00649     }
00650     TCanvas* canvas = new TCanvas("shapeEntryTotal", "shape entry", 2500, 1000);
00651     TH2D* distribution = new TH2D("shapeEntryTotal", static_cast<TString>(config.find("plot_titles")),
nXbin, -nXbin - .5, -.5, nYbin, -nYbin + .5, .5);
00652     ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
00653     for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
00654         int clusterSize = clusterShape->getClusterSize();
00655         int iShape = 0;
00656         for ( const TShapeInfo shapeInfo : clusterShape->getClusterShapeInfos() ) {
00657             pBar.printProgress();
00658             distribution->Fill(-clusterSize, -iShape, shapeInfo.mEntry);
00659             iShape++;
00660         }
00661     }
00662     for ( int iXbin = 1; iXbin < distribution->GetNbinsX(); ++iXbin ) {
00663         distribution->GetXaxis()->SetBinLabel(iXbin, Form("%g", floor(-1 *
distribution->GetXaxis()->GetBinLowEdge(iXbin))));
00664     }
00665
00666     for ( int iYbin = 1; iYbin < distribution->GetNbinsY(); ++iYbin ) {
00667         distribution->GetYaxis()->SetBinLabel(iYbin, Form("%g", floor(-1 *
distribution->GetYaxis()->GetBinLowEdge(iYbin))));
00668     }
00669
00670     distribution->SetStats(0);
00671     if ( config.hasKey("options") ) {
00672         for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
00673             distribution->Draw(static_cast<TString>(optionName));
00674             canvas->SetPhi(10);
00675             canvas->SetTheta(25);
00676             canvas->SetLogz();
00677             std::filesystem::path file = filePath;
00678             file /= (config.find("filename") + "_" + optionName);
00679             file.replace_extension(config.find("extension"));
00680             canvas->SaveAs(static_cast<TString>(file));
00681         }
00682     } else {
00683         distribution->Draw();
00684         std::filesystem::path file = filePath;
00685         file /= config.find("filename");
00686         file.replace_extension(config.find("extension"));
00687         canvas->SaveAs(static_cast<TString>(file));
00688     }
00689
00690     delete distribution;
00691     delete canvas;

```

```
00692 }
```

8.194 /home/ychoi/ATOM/trashcan/TClusterShapeAnalyser.h File Reference

Tools for analysing and drawing cluster shape.

```
#include <vector>
#include <unordered_map>
#include "TClusterAnalyser.h"
```

Classes

- class [TClusterShapeAnalyser](#)

8.194.1 Detailed Description

Tools for analysing and drawing cluster shape.

Author

Yongjun Choi (ychoi@cern.ch)

Version

0.1

Date

16-04-2024

Copyright

Copyright (c) 2024

Definition in file [TClusterShapeAnalyser.h](#).

8.195 TClusterShapeAnalyser.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef __TCLUSTERSHAPEANALYSER__
00012 #define __TCLUSTERSHAPEANALYSER__
00013
00014 #ifdef __TCLUSTERSHAPEANALYSER_HEADER__
00015 #include <iostream>
00016
00017 #include "TCanvas.h"
00018 #include "TH2I.h"
00019 #include "TText.h"
00020 #include "TLine.h"
00021 #include "TPaveText.h"
00022 #include "TGraph.h"
00023 #include "TFl.h"
00024 #include "TLegend.h"
00025
00026 #include "cpptqdm.h"
00027 #include "CppConfigFile.h"
00028
00029 #include "TClusterShape.h"
00030 #include "TCluster.h"
00031 #include "TClusterDivideData.h"
00032 #include "TExperimentData.h"
00033 #include "TMatrix2D.h"
00034 #endif
00035
00036 #include <vector>
00037 #include <unordered_map>
00038
00039 #include "TClusterAnalyser.h"
00040
00041 class TClusterShape;
00042
00043 class TClusterShapeAnalyser : protected TClusterAnalyser {
00044 private:
00045     std::unordered_map<std::string, std::vector<TClusterShape*>> mClusterShapeSet;
00046     std::unordered_map<std::string, int> mNTotalShapeSet;
00047     std::unordered_map<std::string, int> mMaxModeSet;
00048 public:
00049     TClusterShapeAnalyser(const TClusterAnalyser& analyser);
00050     ~TClusterShapeAnalyser();
00051     void doShaping(std::string_view typeName, const std::vector<int>& clusterSizeRange);
00052     void saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config);
00053     void saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config);
00054     void saveSameSizeShapes(std::string_view typeName, const CppConfigDictionary config);
00055     void saveTotalShapes(std::string_view typeName, const CppConfigDictionary config);
00056     void saveSameSizeShapeEntry(std::string_view typeName, const CppConfigDictionary config);
00057     void saveTotalShapeEntry(std::string_view typeName, const CppConfigDictionary config);
00058 private:
00060     unsigned int fBits;
00061 public:
00062     enum {
00063         kNotDeleted = 0x02000000
00064     };
00065     bool IsDestroyed() const { return !TestBit(kNotDeleted); }
00066     bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
00067 };
00068
00069 #endif

```

