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 programing 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 TAnalyser

Add template for plots. Map, distribution, etc.

Member TAnalyser::~TAnalyser ()

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:

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

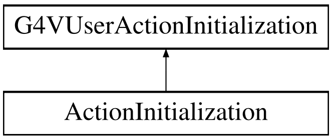
Chapter 7

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.

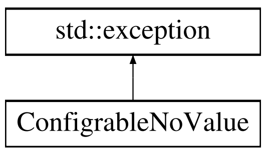
The documentation for this class was generated from the following files:

- /home/ychoi/ATOM/geant4/main/inc/Colour.h
- /home/ychoi/ATOM/geant4/main/src/Colour.cpp

7.6 ConfigrableNoValue Class Reference

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

Inheritance diagram for ConfigrableNoValue:



Public Member Functions

- ConfigrableNoValue (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 ConfigrableNoValue()

```
ConfigrableNoValue::ConfigrableNoValue (
            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 * ConfigrableNoValue::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 ConfigrableNoValue::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 &copy)
- CppConfigDictionary & operator= (const CppConfigDictionary &copy)
- 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 &copy)
- CppConfigDictionary & operator+= (const CppConfigDictionary &copy)

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 &copy)

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::getSub↩
ConfigSetWithName ( ) 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 &copy)
- CppConfigFile & operator= (const CppConfigFile &copy)
- 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 &copy)

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

configFile	

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

configFile	

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.

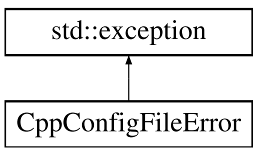
The documentation for this class was generated from the following files:

- /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.

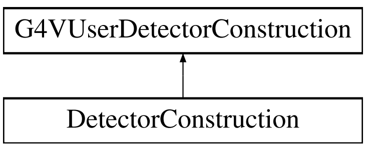
The documentation for this class was generated from the following file:

- /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 alpideType)
- 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.

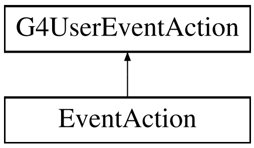
The documentation for this class was generated from the following files:

- /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 alpideMaterial

```
G4Material* Material::alpideMaterial [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 elO

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

Definition at line 27 of file Material.h.

7.17.4.13 elSi

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

Definition at line 25 of file Material.h.

7.17.4.14 elTi

```
G4Element* Material::elTi [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.

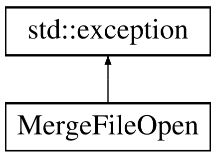
The documentation for this class was generated from the following files:

- /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.

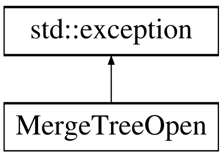
The documentation for this class was generated from the following file:

- /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.

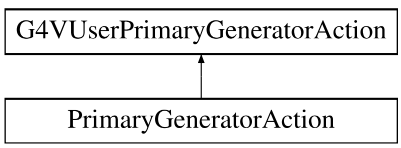
The documentation for this class was generated from the following file:

- /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.

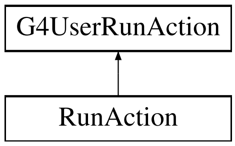
The documentation for this class was generated from the following files:

- /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.

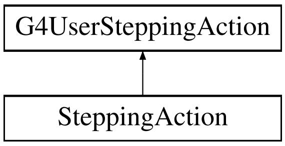
The documentation for this class was generated from the following files:

- /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.

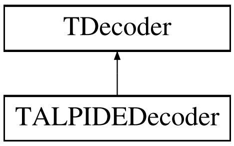
The documentation for this struct was generated from the following file:

- /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.

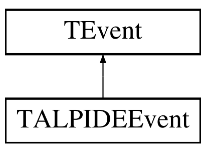
The documentation for this class was generated from the following files:

- /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 &copy)
- TALPIDEEvent & operator= (const TALPIDEEvent &copy)
- 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 IsDestructed () 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:

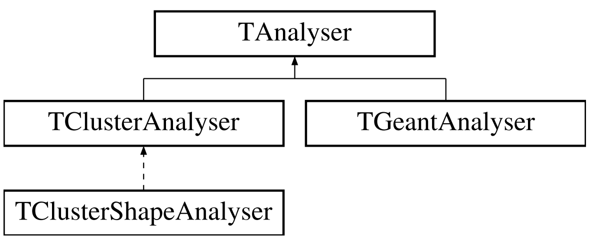
- /home/ychoi/ATOM/alpide/daq/inc/TALPIDEEvent.h
- /home/ychoi/ATOM/alpide/daq/src/TALPIDEEvent.cpp

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 sotre 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

inputFile	
expData	

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

typeName	
clusterSizeRange	

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

config	Draw configuration for map title, directory and filename
clusters	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

config	
clusters	

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

typeName	

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

fileName	

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

treeName	

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

typeName	
config	

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

typeName	
config	

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

config	

Array for 'TALPIDEEvent' class. The events extracted from TTree will be saved here.

Variable for storing previous time.

Store initial event to 'TALPIDEEvent' 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.

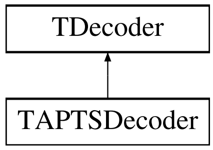
The documentation for this class was generated from the following files:

- /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.

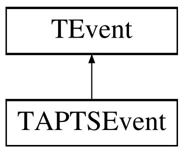
The documentation for this class was generated from the following files:

- /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 &copy)
- TCluster & operator= (const TCluster &copy)
- 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:

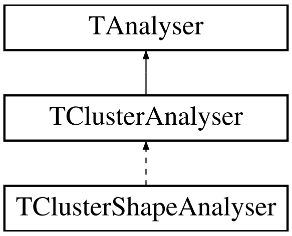
- /home/ychoi/ATOM/alpide/cluster/inc/TCluster.h
- /home/ychoi/ATOM/alpide/cluster/src/TCluster.cpp
- /home/ychoi/ATOM/alpide/cluster/src/TClusterOperator.cpp

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 &copy)
- ∼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 IsDestructed () 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 imformations 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

analyser	

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 &copy)
- 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

clusters	
clusterSize	

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 visulaizes shape informations form matrix form. `TImage` class is used for saving plots. The extra pixel on border of plot is drawn for readability.

Parameters

clusterMatrix	

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 ( )
```

Specificate 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

clusterSize	

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

descend	

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.

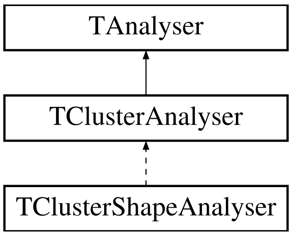
The documentation for this class was generated from the following files:

- /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 IsDestructed () 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 &copy)
- ∼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

analyser	

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

typeName	
clusterSizeRange	

Definition at line 47 of file TClusterShapeAnalyser.cpp.

7.38.4.2 IsDestructed()

```
bool TClusterShapeAnalyser::IsDestructed ( ) 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

typeName	
config	

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

typeName	
config	

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::m←
ClusterShapeSet  [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.

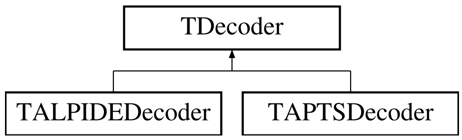
The documentation for this class was generated from the following files:

- /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 detector←CoordY, 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.

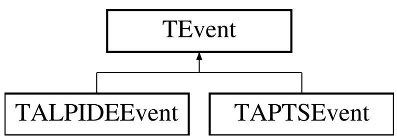
The documentation for this class was generated from the following files:

- /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 &copy)
- TExperimentData & operator= (const TExperimentData &copy)
- ∼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< TALPIDEEvent * > & 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<TALPIDEEvent*> TExperimentData::mEvents [private]
```

Definition at line 27 of file TExperimentData.h.

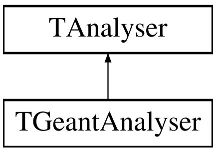
The documentation for this class was generated from the following files:

- /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:

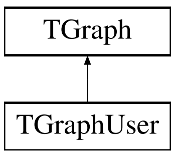
- /home/ychoi/ATOM/alpide/comparison/inc/TGraphCompare.h
- /home/ychoi/ATOM/alpide/comparison/src/TGraphCompare.cpp

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 mUniversialMarkerStyle
- Size_t mUniversialMarkerSize
- Style_t mUniversialLineStyle
- Size_t mUniversiallineWidth
- 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

config	

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

config	

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::graphSetAnd←
Attribute [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 mUniversialLineStyle

`Style_t TGraphUser::mUniversialLineStyle [private]`

Definition at line 34 of file TGraphUser.h.

7.47.4.11 mUniversiallineWidth

`Size_t TGraphUser::mUniversiallineWidth [private]`

Definition at line 35 of file TGraphUser.h.

7.47.4.12 mUniversialMarkerSize

`Size_t TGraphUser::mUniversialMarkerSize [private]`

Definition at line 33 of file TGraphUser.h.

7.47.4.13 mUniversialMarkerStyle

`Style_t TGraphUser::mUniversialMarkerStyle [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.

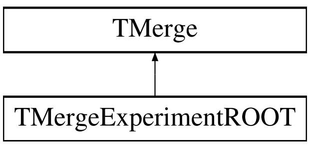
The documentation for this class was generated from the following files:

- /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.

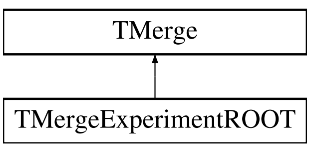
The documentation for this class was generated from the following files:

- /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 > &inputFile↩ Names)
- 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.

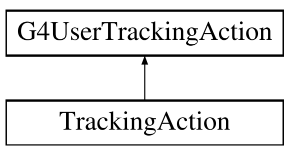
The documentation for this class was generated from the following files:

- /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 genKineteicEnergy
- 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 genKineteicEnergy

```
double TrackTuple::genKineteicEnergy
```

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 stucture 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 stucture 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 &copy)
- TThreshold & operator= (const TThreshold &copy)
- 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 > & coodrdinate,
            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 > & coodrdinate,
            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 mIthr
- 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 mIthr

```
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 (yochoi@cern.ch)

Version

0.1

Date

13-04-2024

Copyright

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 "TF1.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:
```

```
    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;
public:
    TAnalyser() = default;
    TAnalyser(TFile* inputFile, std::unordered_map<std::string, TExperimentData*> expData);
    ~TAnalyser();

    TTree* openTree(std::string treeName);
    void storeEvents(CppConfigDictionary settingConfig);
    void doMasking(int mMaskOver);
    void openOutputGraphFile(std::string_view fileName);
    void openDirectory(std::string_view typeName);

    // void setSavePath(const std::filesystem::path& savePath);
    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);
    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);
    void saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config);
    void saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config);
    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);

private:
    UInt_t fBits;
public:
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(UInt_t f) const { return (bool) ((fBits & f) != 0); }
};

#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.

```

#ifndef __TTHRESHOLDANALYSER__
#define __TTHRESHOLDANALYSER__

#ifdef __TTHRESHOLDANALYSER_HEADER__
#include <iostream>
#include <sstream>

#include "RtypesCore.h"
#include "TH1D.h"
#include "TH2D.h"
#include "TCanvas.h"

#include "cpptqdm.h"

#include "TThreshold.h"
#endif

#include<vector>
#include <fstream>
 // #include "TThreshold.h"
 // #include "cpptqdm.h"
class TThreshold;
class TH1;
class TH2;

class TThresholdAnalyser {
private:
    std::ifstream mFile;
    int mVcasn, mIthr;
    std::vector<TThreshold*> mThresholds;
    TH1* mThresholdDistribution;
    TH1* mErrorDistribution;
    TH2* mThresholdmap;
    TH1* mChi2NdfDistribution;
public:
    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;
};

#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.

```
#define __TANALYSER_HEADERS__

#include "TAnalyser.h"

TAnalyser::TAnalyser(TFile* inputFile, std::unordered_map<std::string, TExperimentData*> expData) : mInputFile(inputFile), mExpData(expData), fBits(kNotDeleted) {
    std::filesystem::path inputPath = mInputFile->GetName();
    std::clog « "TAnalyser object for \033[1;32m" « inputPath.stem() « "\033[0m is armed" « std::endl;
    mTree = openTree("hit");
    mTree->SetBranchAddress("ChipID", &mInput.chipid);
    mTree->SetBranchAddress("TimeStamp", &mInput.timeStamp);
    mTree->SetBranchAddress("X", &mInput.x);
    mTree->SetBranchAddress("Y", &mInput.y);

    gErrorIgnoreLevel = kWarning;
}

TAnalyser::~TAnalyser() {
    if ( mOutputFile != nullptr && !mOutputFile->IsDestructed() ) {
        mOutputFile->Close();
        mOutputFile = nullptr;
    }
    // for ( const auto& pair : mHitmaps ) {
    //  if ( !pair.second->IsDestructed() ) {
    //      delete pair.second;
    //  }
    // }
    // for ( const auto& pair : mExpData ) {
    //  if ( !pair.second->IsDestructed() ) {
    //      delete pair.second;
    //  }
    // }
    // if ( mInputFile != nullptr && !mInputFile->IsDestructed() ) {
    //  mInputFile->Close();
    //  delete mInputFile;
    //  mInputFile = nullptr;
    // }
    // if ( mTree != nullptr && !mTree->IsDestructed() ) {
    //  delete mTree;
    //  mTree = nullptr;
    // }
    // if ( mExpSettingLegend != nullptr && !mExpSettingLegend->IsDestructed() ) {
    //  delete mExpSettingLegend;
    //  mExpSettingLegend = nullptr;
    // }
    std::clog « "TAnalyser object is terminated" « std::endl;
}

TTree* TAnalyser::openTree(std::string treeName) {
    try {
        if ( mInputFile->Get(static_cast<TString>(treeName)) == nullptr ) throw(treeName);
        else if ( std::string(mInputFile->Get(static_cast<TString>(treeName))->ClassName()) != "TTree" ) throw(treeName);
        return static_cast<TTree*>(mInputFile->Get(static_cast<TString>(treeName)));
    } catch ( std::string treeName ) {
        std::cerr « "There are no " « treeName « "TTree in this TFile" « std::endl;
        exit(1);
    }
}

void TAnalyser::openOutputGraphFile(std::string_view fileName) {
    mOutputFile = new TFile(static_cast<TString>(fileName), "RECREATE");
    mIsOutputGraph = true;
}

void TAnalyser::openDirectory(std::string_view typeName) {
    mOutputFile->cd();
    TDirectory* directory = mOutputFile->mkdir(static_cast<TString>(typeName));
    mDirectorySet.insert_or_assign(std::string(typeName), directory);
}

void TAnalyser::storeEvents(CppConfigDictionary config) {
    std::clog « "Extracting Events..." « std::endl;

    std::vector<TALPIDEEvent*> tempEvents;
    UInt_t preTime = 0;
    tempEvents.push_back(new TALPIDEEvent());
    tempEvents.back()->setEvent(0);
    tempEvents.back()->setTime(static_cast<long int>(0));

    ProgressBar* pbar = new ProgressBar(mTree->GetEntries());
    TH1I* timeStampHist = new TH1I("TSHist", "", 50001, 0, 50000);
```

```
    for ( int entry = 0; entry < mTree->GetEntries(); entry++ ) {
        pbar->printProgress();
        mTree->GetEntry(entry);
        if ( mInput.timeStamp == preTime ) {
            tempEvents.back()->pushData({mInput.x, mInput.y});
        } else {
            tempEvents.back()->removeDuplication();
            tempEvents.back()->sortPixel();
            if ( config.hasKey("time_stamp_cut") ) {
                if ( tempEvents.back()->getNData() > stoi(config.find("time_stamp_cut")) ) {
                    tempEvents.pop_back();
                }
            }
            tempEvents.push_back(new TALPIDEEvent());
            tempEvents.back()->setEvent(mInput.timeStamp);
            tempEvents.back()->setTime(mInput.timeStamp);
            tempEvents.back()->pushData({mInput.x, mInput.y});
            preTime = mInput.timeStamp;
        }
    }
    tempEvents.back()->removeDuplication();
    tempEvents.back()->sortPixel();

    for ( auto& event : tempEvents ) {
        if ( event->getNData() > 200000 ) std::cout « event->getNData();
        timeStampHist->SetBinContent(event->getEvent(), event->getNData());
    }

    TCanvas* can = new TCanvas("TSCanvas", "", 2000, 1000);
    timeStampHist->Draw();
    std::filesystem::path filePath(config.find("output_path"));
    std::filesystem::create_directories(filePath);
    std::filesystem::path file = filePath;
    file /= "timeStamp.png";
    can->SaveAs(static_cast<TString>(std::string(file)));

    TH1D* timeStampHist2 = new TH1D("TSHist2", "", 1000, 0, 1000);
    for ( int i = 0; i < 50000; i++ ) {
        timeStampHist2->Fill(timeStampHist->GetBinContent(i));
    }
    TCanvas* can2 = new TCanvas("TSCanvas2", "", 2000, 1000);
    timeStampHist2->Draw();
    std::filesystem::path filePath2(config.find("output_path"));
    std::filesystem::create_directories(filePath2);
    std::filesystem::path file2 = filePath2;
    file2 /= "timeStampHist.png";
    can2->SetLogy();
    can2->SaveAs(static_cast<TString>(std::string(file2)));
    delete pbar;
    std::clog « "The " « tempEvents.size() « " of events is extracted from " « mTree->GetEntries() « " stamps." « std::endl;
    mExpData.find("Basic")->second->setEvents(std::move(tempEvents));
}

void TAnalyser::doMasking(int mMaskOver) {
    TMatrix2D<int> matrix(1025, 513);
    for ( int x = 0; x < 1025; x++ ) {
        for ( int y = 0; y < 513; y++ ) {
            if ( mHitmaps.find("Basic")->second->GetBinContent(x, y) > mMaskOver ) {
                matrix.setElement(x - 1, y - 1, 1);
            }
        }
    }

    std::vector<TALPIDEEvent*> tempEvents;
    std::vector<TALPIDEEvent*> noiseEvents;

    for ( const TALPIDEEvent* event : mExpData.find("Basic")->second->getEvents() ) {
        tempEvents.push_back(new TALPIDEEvent());
        tempEvents.back()->setEvent(event->getEvent());
        tempEvents.back()->setTime(event->getTime());
        noiseEvents.push_back(new TALPIDEEvent());
        noiseEvents.back()->setEvent(event->getEvent());
        noiseEvents.back()->setTime(event->getTime());
        for ( const std::pair<int, int>& pixel : event->getData() ) {
            if ( matrix.getElement(pixel.first, pixel.second) != 1 ) {
                tempEvents.back()->pushData(std::move(pixel));
            } else {
                noiseEvents.back()->pushData(std::move(pixel));
            }
        }
    }

    TExperimentData* maskedData = new TExperimentData();
    maskedData->setEvents(std::move(tempEvents));
    mExpData.insert_or_assign("Masked", maskedData);
```

```
    TExperimentData* noisePixelData = new TExperimentData();
    noisePixelData->setEvents(std::move(noiseEvents));
    mExpData.insert_or_assign("NoisePixel", noisePixelData);
}

void TAnalyser::setExpSettingLegend(CppConfigDictionary settingConfig) {
    mExpSettingLegend = new TPaveText(.78, .1, .981, .65, "NDC");
    if ( settingConfig.hasKey("NTRG") ) {
        mExpSettingLegend->AddText(static_cast<TString>("NTRG= " + settingConfig.find("NTRG")));
    }
    mExpSettingLegend->AddText(static_cast<TString>("NEVT= " +
std::to_string(mExpData.find("Basic")->second->getEvents().size())));
    if ( settingConfig.hasKey("THRESHOLD") ) {
        mExpSettingLegend->AddText(static_cast<TString>("Threshold= " +
settingConfig.find("THRESHOLD")));
    }
    if ( settingConfig.hasKey("COL_LENGTH") ) {
        mExpSettingLegend->AddText(static_cast<TString>("Collimator Length = " +
settingConfig.find("COL_LENGTH")));
    }
    if ( settingConfig.hasKey("COL_DIAMETER") ) {
        mExpSettingLegend->AddText(static_cast<TString>("Collimator Diameter = " +
settingConfig.find("COL_DIAMETER")));
    }
    if ( settingConfig.hasKey("AIR_PRESSURE") ) {
        mExpSettingLegend->AddText(static_cast<TString>("Air Pressure = " +
settingConfig.find("AIR_PRESSURE")));
    }

    mExpSettingLegend->SetTextAlign(11);
    mExpSettingLegend->SetTextFont(42);
}

TExperimentData* TAnalyser::getAnEventSet(std::string_view typeName) const {
    if ( mExpData.count(std::string(typeName)) ) {
        return mExpData.find(std::string(typeName))->second;
    } else {
        return new TExperimentData();
    }
}

TH2D* TAnalyser::getHitPlot(const CppConfigDictionary& config, const std::vector<TALPIDEEvent*>&
events) {
    // Static variable for numbering
    static int iHMap = 0;
    // Allocate a 2d histogram.
    std::string plotTitle = "";
    if ( config.hasKey("title") ) {
        plotTitle += config.find("title");
    }
    if ( config.hasKey("x_title") ) {
        plotTitle += "; " + config.find("x_title");
    } else {
        plotTitle += "; ";
    }
    if ( config.hasKey("y_title") ) {
        plotTitle += "; " + config.find("y_title");
    } else {
        plotTitle += "; ";
    }

    TH2D* map = new TH2D(Form("hmap%d", iHMap), static_cast<TString>(plotTitle), 1024, 0, 1024, 512,
0, 512);
    // Fill data
    for ( const TALPIDEEvent* event : events ) {
        for ( const std::pair<int, int>& pixel : event->getData() ) {
            map->Fill(pixel.first, pixel.second);
        }
    }
    // Canvas setting
    TCanvas* canvas = new TCanvas(Form("hmapCan%d", iHMap), "", 2500, 1000);
    canvas->SetMargin(.07, .35, .12, .08);
    map->GetXaxis()->SetTitleOffset(1.4);
    map->GetXaxis()->SetLabelOffset(0.003);
    // Find directory for saving clustermap. If it doesn't exist, then make the directory with mother
directories.
    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    std::filesystem::create_directories(filePath);
    // Draw plot with options
    if ( config.hasKey("options") ) {
        for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
            map->Draw(static_cast<TString>(optionName));
            mExpSettingLegend->Draw("SAME");
            std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
            if ( config.hasKey("extension") ) {
```

```
                file.replace_extension(config.find("extension"));
            } else {
                file.replace_extension("png");
            }
            canvas->SaveAs(static_cast<TString>(file));
        }
    } else {
        map->Draw();
        mExpSettingLegend->Draw("SAME");
        canvas->SaveAs(static_cast<TString>(filePath));
    }
    // Delete canvas;
    delete canvas;
    iHMap++;
    return map;
}

void TAnalyser::saveHitmap(std::string typeName, const CppConfigDictionary& config) {
    std::clog « "Generating \033[1;32m" « typeName « " Hitmap\033[1;0m..." « std::endl;
    if ( !mHitmaps.count(typeName) ) {
        mHitmaps.insert_or_assign(typeName, getHitPlot(config, mExpData.find(typeName)->second->getEvents()));
        if ( mIsOutputGraph ) {
            mDirectorySet.find(std::string(typeName))->second->cd();
            mHitmaps.find(std::string(typeName))->second->Write("hitmap");
            mOutputFile->cd();
        }
    } else {
        getHitPlot(config, mExpData.find(typeName)->second->getEvents());
    }
}

void TAnalyser::doDivideBySize(std::string_view typeName) {
    TClusterDivideData* clusterDivideData = new TClusterDivideData(mExpData.find(std::string(typeName))->second->getClusters());
    mDivideData.insert_or_assign(std::string(typeName), clusterDivideData);
}


TH2D* TAnalyser::getClusterPlot(const CppConfigDictionary& config, const std::vector<TCluster*>& clusters) {
    // Static variable for numbering
    static int iMap = 0;
    // Allocate a 2d histogram.
    std::string plotTitle = "";
    if ( config.hasKey("title") ) {
        plotTitle += config.find("title");
    }
    if ( config.hasKey("x_title") ) {
        plotTitle += "; " + config.find("x_title");
    } else {
        plotTitle += "; ";
    }
    if ( config.hasKey("y_title") ) {
        plotTitle += "; " + config.find("y_title");
    } else {
        plotTitle += "; ";
    }

    TH2D* map = new TH2D(Form("map%d", iMap), static_cast<TString>(plotTitle), 1024, 0, 1024, 512, 0, 512);
    // Fill data
    std::vector<int> interestSizeSet = getClusterSizeRange(config);
    for ( const TCluster* cluster : clusters ) {
        if ( !interestSizeSet.empty() ) {
            if ( std::find(interestSizeSet.begin(), interestSizeSet.end(), cluster->getSize()) != interestSizeSet.end() ) {
                map->Fill(cluster->getCenter().first, cluster->getCenter().second);

            }
        } else {
            map->Fill(cluster->getCenter().first, cluster->getCenter().second);
        }
    }
    // Canvas setting
    TCanvas* canvas = new TCanvas(Form("mapCan%d", iMap), "", 2500, 1000);
    canvas->SetMargin(.07, .35, .12, .08);
    map->GetXaxis()->SetTitleOffset(1.4);
    map->GetXaxis()->SetLabelOffset(0.003);
    // Find directory for saving clustermap. If it doesn't exist, then make the directory with mother directories.
    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    std::filesystem::create_directories(filePath);

    // Draw plot with options
    if ( config.hasKey("options") ) {
```

```
        for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
            map->Draw(static_cast<TString>(optionName));
            mExpSettingLegend->Draw("SAME");
            std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
            if ( config.hasKey("extension") ) {
                file.replace_extension(config.find("extension"));
            } else {
                file.replace_extension("png");
            }
            canvas->SaveAs(static_cast<TString>(file));
        }
    } else {
        map->Draw();
        mExpSettingLegend->Draw("SAME");
        std::filesystem::path file = filePath / (config.find("filename"));
        if ( config.hasKey("extension") ) {
            file.replace_extension(config.find("extension"));
        } else {
            file.replace_extension("png");
        }
        canvas->SaveAs(static_cast<TString>(file));
    }
    // Delete canvas;
    delete canvas;
    iMap++;
    return map;
}
TH1D* TAnalyser::getClustersizePlot(const CppConfigDictionary& config, const std::vector<TCluster*>& clusters) {
    static int iDistribution = 0;
    TString distName = Form("distribution%d", iDistribution);
    TString distTitle = "";
    distTitle += config.hasKey("title") ? config.find("title") : "";
    distTitle += config.hasKey("x_title") ? "; " + config.find("x_title") : ";";
    distTitle += config.hasKey("y_title") ? "; " + config.find("y_title") : ";";
    Int_t nBins = 0;
    Float_t maxBin = 0;
    Float_t minBin = 0;
    if ( config.hasKey("distribution_info") ) {
        nBins = config.getSubConfig("distribution_info").hasKey("nbins") ? stoi(config.getSubConfig("distribution_info").find("nbins")) : 80;
        maxBin = config.getSubConfig("distribution_info").hasKey("max") ? stoi(config.getSubConfig("distribution_info").find("max")) + .5 : 80.5;
        minBin = config.getSubConfig("distribution_info").hasKey("min") ? stoi(config.getSubConfig("distribution_info").find("min")) - .5 : .5;
    } else {
        nBins = 80;
        maxBin = 80.5;
        minBin = 0.5;
    }
    TH1D* distribution = new TH1D(distName, distTitle, nBins, minBin, maxBin);
    for ( const TCluster* cluster : clusters ) {
        distribution->Fill(cluster->getSize()); // Fill clustersize to clustersize distribution
    }
    TCanvas* canvas = new TCanvas(Form("distributionCan%d", iDistribution), "", 2500, 1000);
    canvas->SetMargin(.07, .28, .12, .08);
    distribution->GetXaxis()->SetTitleOffset(1.4);
    distribution->GetXaxis()->SetLabelOffset(0.003);

    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    std::filesystem::create_directories(filePath);

    if ( config.hasKey("options") ) {
        for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
            if ( optionName == "logy" ) {
                distribution->Draw();
                mExpSettingLegend->Draw("SAME");
                canvas->SetLogy();
                std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
                if ( config.hasKey("extension") ) {
                    file.replace_extension(config.find("extension"));
                } else {
                    file.replace_extension("png");
                }
                canvas->SaveAs(static_cast<TString>(file));
            } else if ( optionName == "basic" ) {
                distribution->Draw();
                mExpSettingLegend->Draw("SAME");
                std::filesystem::path file = filePath / config.find("filename");
                if ( config.hasKey("extension") ) {
                    file.replace_extension(config.find("extension"));
                } else {
                    file.replace_extension("png");
                }
                canvas->SetLogy(0);
                canvas->SaveAs(static_cast<TString>(file));
```

```
            }
        }
    } else {
        distribution->Draw();
        mExpSettingLegend->Draw("SAME");
        std::filesystem::path file = filePath / (config.find("filename"));
        if ( config.hasKey("extension") ) {
            file.replace_extension(config.find("extension"));
        } else {
            file.replace_extension("png");
        }
        canvas->SaveAs(static_cast<TString>(file));
    }
    delete canvas;
    iDistribution++;
    return distribution;
}

void TAnalyser::saveClustermap(std::string typeName, const CppConfigDictionary& config) {
    std::clog « "Generating \033[1;32mClustermap\033[1;0m..." « std::endl;
    if ( !mClustermaps.count(typeName) ) {
        mClustermaps.insert_or_assign(typeName, getClusterPlot(config, mExpData.find(typeName)->second->getClusters()));
        if ( mIsOutputGraph ) {
            mDirectorySet.find(std::string(typeName))->second->cd();
            mClustermaps.find(std::string(typeName))->second->Write("clustermap");
            mOutputFile->cd();
        }
    } else {
        getClusterPlot(config, mExpData.find(typeName)->second->getClusters());
    }
}

void TAnalyser::saveClustersize(std::string typeName, const CppConfigDictionary& config) {
    std::clog « "Generating \033[1;32mCluster Size Distribution\033[1;0m..." « std::endl;
    if ( !mClustersizes.count(typeName) ) {
        mClustersizes.insert_or_assign(typeName, getClustersizePlot(config, mExpData.find(typeName)->second->getClusters()));
        if ( mIsOutputGraph ) {
            mDirectorySet.find(std::string(typeName))->second->cd();
            mClustersizes.find(std::string(typeName))->second->Write("clustersize");
            mOutputFile->cd();
        }
    } else {
        getClustersizePlot(config, mExpData.find(typeName)->second->getClusters());
    }
}

void TAnalyser::setClusterDataWithShape(const std::vector<int>& clusterSizeRange) {
    for ( const int clusterSize : clusterSizeRange ) {
        std::vector<TCluster*> clustersWithShape;
    }
}

std::vector<int> TAnalyser::getClusterSizeRange(const CppConfigDictionary& privateProperty) {
    std::vector<int> clusterSizeRange;
    if ( privateProperty.hasKey("interest_size") ) {
        for ( const std::string& rangeStr : privateProperty.getSubConfig("interest_size").getValueList() ) {
            if ( rangeStr.find('.') != std::string::npos ) {
                for ( int i = stoi(rangeStr.substr(0, rangeStr.find('.'))); i < stoi(rangeStr.substr(rangeStr.find('.') + 3)) + 1; i++ ) {
                    clusterSizeRange.push_back(i);
                }
            } else {
                clusterSizeRange.push_back(stoi(rangeStr));
            }
        }
    } else {
        for ( int i = 0; i < 100; i++ ) {
            clusterSizeRange.push_back(i);
        }
    }
    return clusterSizeRange;
}


void TAnalyser::doShaping(std::string_view typeName, const std::vector<int>& clusterSizeRange) {
    // The number of total shape and the maximum number of shapes of each cluster sizes.
    int nTotalShape = 0;
    int maxMode = 0;
    // The array of objects of TClusterShape class.
    std::vector<TClusterShape*> clusterShapes;
    // This stores and extracts clusters shape informations.
    for ( const int clusterSize : clusterSizeRange ) {
        if ( mClustersizes.find(std::string(typeName))->second->GetBinContent(clusterSize) != 0 ) {
            // This gets cluster bunch from TClusterDivideData object which classifies clusters by
```

```
their size.
            const std::vector<TCluster*> divideData = mDivideData.find(std::string(typeName))->second->getClusterOfSize(clusterSize);
            // This inserts cluster size information and clusters to TClusterShape objects.
            TClusterShape* clusterShape = new TClusterShape(clusterSize, divideData);
            // This extracts shape informations.
            clusterShape->identifyShapes();
            // This sorts shapes according to its size of long axis.
            clusterShape->sortShapes();
            //
            clusterShapes.push_back(clusterShape);
            nTotalShape += clusterShape->getClusterShapeInfos().size();
            maxMode = std::max(maxMode, static_cast<int>(clusterShape->getClusterShapeInfos().size()));
        }
    }
    mClusterShapeSet.insert_or_assign(std::string(typeName), clusterShapes);
    mNTotalShapeSet.insert_or_assign(std::string(typeName), nTotalShape);
    mMaxModeSet.insert_or_assign(std::string(typeName), maxMode);
}
void TAnalyser::saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config) {
    // Print log message
    std::clog « "Generating " « "\033[1;32m" « "individual shapes" « "\033[1;0m" « "..." « std::endl;

    // Get the total number of shapes for progress bar.
    int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;

    // Get save path from config file.
    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    std::filesystem::create_directories(filePath);

    // Init progress bar
    ProgressBar pBar(nTotalShape);

    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize(); // The cluster size of shape set.
        int iClusterShape = 0; // The i-th cluster shape
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress(); // Print progress bar.

            int shapeWidth = shapeInfo.mClusterMatrix->getNRow(); // Shape width of matrix
            int shapeHeight = shapeInfo.mClusterMatrix->getNColumn(); // Shape height of matrix

            // Initialize canvas
            TCanvas* canvas = new TCanvas(Form("%d_%d.png", clusterSize, iClusterShape), "", (shapeWidth + 2) * 500, (shapeHeight + 2) * 500);
            // Set canvas margin
            canvas->SetMargin(0., 0., 0., 0.);
            // Draw shape on canvas
            shapeInfo.mClusterMap->Draw();
            // Fill colour
            shapeInfo.mClusterMap->SetDrawOption("COLZ");
            // Remove histogram axis
            shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);

            // Fill line for seperating pixels
            TLine* line = new TLine();
            line->SetLineColorAlpha(kRed, 6. / 8);
            for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
                for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
                    if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
                        double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
                        double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
                        double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
                        double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
                        line->DrawLine(xlow, ylow, xup, ylow); // Bottom
                        line->DrawLine(xlow, yup, xup, yup); // Top
                        line->DrawLine(xlow, ylow, xlow, yup); // Left
                        line->DrawLine(xup, ylow, xup, yup); // Right
                    }
                }
            }

            // Set title of shape
            TText* title = new TText(.5, 1. - .5 / (shapeHeight + 2), Form("%d-th Cluster Shape of Cluster Size %d", iClusterShape, clusterSize));
            title->SetTextAlign(23);
            title->SetNDC();
            title->Draw();

            // Save shape figure
            std::filesystem::path file = filePath / (config.find("filename") + "_" + std::to_string(clusterSize) + "_" + std::to_string(iClusterShape));
            if ( config.hasKey("extension") ) {
                file.replace_extension(config.find("extension"));
```

```
            } else {
                file.replace_extension("png");
            }
            canvas->SaveAs(static_cast<TString>(file));

            iClusterShape++;

            delete title;
            title = nullptr;
            delete line;
            line = nullptr;
            delete canvas;
            canvas = nullptr;
        }
    }
}

int calNIncludePixel(const TMatrix2D<int>* matrix) {
    int centreX = 0, centreY = 0;
    int clusterSize = 0;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            if ( matrix->getElement(pixelX, pixelY) == 1 ) {
                centreX += pixelX * 2;
                centreY += pixelY * 2;
                clusterSize++;
            }
        }
    }
    int radiusSquare = 0;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            if ( matrix->getElement(pixelX, pixelY) == 1 ) {
                int distance = pow(abs(2 * pixelX * clusterSize - centreX) + clusterSize, 2) + pow(abs(2 * pixelY * clusterSize - centreY) + clusterSize, 2);
                radiusSquare = std::max(radiusSquare, distance);
            }
        }
    }
    int count = 0;
    for ( int x = floor((centreX - sqrt(radiusSquare)) / (2 * clusterSize)); x < ceil((centreX + sqrt(radiusSquare)) / (2 * clusterSize)) + 1; x++ ) {
        for ( int y = floor((centreY - sqrt(radiusSquare)) / (2 * clusterSize)); y < ceil((centreY + sqrt(radiusSquare)) / (2 * clusterSize)) + 1; y++ ) {
            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 ) {
                count++;
            }
        }
    }
    return count;
}

double calRatioOfRadius(const TMatrix2D<int>* matrix) {
    int centreX = 0, centreY = 0;
    int clusterSize = 0;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            if ( matrix->getElement(pixelX, pixelY) == 1 ) {
                centreX += pixelX * 2;
                centreY += pixelY * 2;
                clusterSize++;
            }
        }
    }
    int longRadiusSquare = 0;
    int shortRadiusSquare = 100000000;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            bool hasValue = matrix->getElement(pixelX, pixelY) == 1;
            bool isWorldBorder = pixelX == 0 || pixelY == 0 || pixelX == matrix->getNRow() - 1 || pixelY == matrix->getNColumn() - 1;
            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;
            if ( hasValue ) {
                int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY * clusterSize - centreY, 2);
                longRadiusSquare = std::max(longRadiusSquare, distance);
            }
            if ( (hasValue && isWorldBorder) || (hasValue && !isWorldBorder && isBorder) ) {
                int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY * clusterSize - centreY, 2);
                shortRadiusSquare = std::min(shortRadiusSquare, distance);
            }
        }
    }
```

```
    if ( matrix->getNRow() == 1 || matrix->getNColumn() == 1 ) {
        shortRadiusSquare = 0;
    }
    return sqrt(static_cast<double>(shortRadiusSquare) / longRadiusSquare);
}

void TAnalyser::saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config) {
    // Print out a log
    std::cout « "Generating " « "\033[1;32m" « "Informations of shapes with same size" « "\033[1;0m" « "..." « std::endl;

    // Settting output file path
    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    // The creation of directories of output path
    std::filesystem::create_directories(filePath);
    // Call configs. First element is config name and second one is config values.
    std::unordered_map<std::string, CppConfigDictionary> plotConfigList = config.getSubConfig("ratio_distribution").getSubConfigSetWithName();
    // Call histograms
    std::unordered_map<std::string, TH1D*> distributionSet;
    // Set the information of histograms from config file
    for ( const auto& plotConfig : plotConfigList ) {
        // Plot name is config name. It isn't value of "name" key.
        std::string plotName = plotConfig.second.find("name");
        // Set plot title and x, y label.
        TString plotTitle = plotConfig.second.hasKey("title") ? plotConfig.second.find("title") : "";
        TString plotXTitle = plotConfig.second.hasKey("x_title") ? plotConfig.second.find("x_title") : "";
        TString plotYTitle = plotConfig.second.hasKey("y_title") ? plotConfig.second.find("y_title") : "";
        // Set number of bin and min and max of x-direction range.
        Int_t plotNBin = plotConfig.second.hasKey("n_bin") ? stoi(plotConfig.second.find("n_bin")) : 100;
        Float_t plotXMin = plotConfig.second.hasKey("x_min") ? stof(plotConfig.second.find("x_min")) : 0.;
        Float_t plotXMax = plotConfig.second.hasKey("x_max") ? stof(plotConfig.second.find("x_max")) : 1.;

        // Add histograms to map.
        distributionSet.insert_or_assign(plotConfig.first, new TH1D(static_cast<TString>(plotName), plotTitle + "; " + plotXTitle + "; " + plotYTitle, plotNBin, plotXMin, plotXMax));
    }

    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        // Get cluster size.
        int clusterSize = clusterShape->getClusterSize();
        if ( clusterSize < stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_min")) || clusterSize > stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_max")) ) {
            continue;
        }
        // Initialize canvas.
        TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
        // int binNum = clusterShape->getClusterShapeInfos().size();
        TGraph* areaRatioGraph = new TGraph();
        TGraph* radiusRatioGraph = new TGraph();

        int iShape = 0;
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            // pBar.printProgress();
            for ( const auto& plotConfig : plotConfigList ) {
                if ( plotConfig.second.find("name") == "area_ratio" ) {
distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix));
                }
                if ( plotConfig.second.find("name") == "area_ratio_with_entry" ) {
distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix), shapeInfo.mEntry);
                }
                if ( plotConfig.second.find("name") == "radius_ratio" ) {
distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix));
                }
                if ( plotConfig.second.find("name") == "radius_ratio_with_entry" ) {
distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix), shapeInfo.mEntry);
                }
            }

            areaRatioGraph->SetPoint(iShape, iShape, static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix));
            radiusRatioGraph->SetPoint(iShape, iShape, calRatioOfRadius(shapeInfo.mClusterMatrix));
            // std::cout « clusterSize « "\t" « iShape « "\t" « static_cast<double>(clusterSize) /
```

```
calNIncludePixel(shapeInfo.mClusterMatrix) « "\t" « calRatioOfRadius(shapeInfo.mClusterMatrix) « "\t"
« shapeInfo.mEntry « std::endl;
            iShape++;
        }
        TF1* line1 = new TF1("line1", "0.8", 0, iShape);
        TF1* line2 = new TF1("line2", "0.5", 0, iShape);

        areaRatioGraph->SetTitle(Form("Informations for cluster shapes in cluster size %d",
clusterSize));
        areaRatioGraph->GetXaxis()->SetTitle("i-th cluster shape");
        areaRatioGraph->GetXaxis()->SetTitleOffset(1.4);
        areaRatioGraph->GetXaxis()->SetLabelOffset(0.003);
        areaRatioGraph->GetYaxis()->SetTitle("Ratio");
        areaRatioGraph->GetYaxis()->SetTitleOffset(0.7);
        areaRatioGraph->SetMarkerStyle(45);
        areaRatioGraph->SetMarkerSize(4);
        areaRatioGraph->SetMarkerColor(kRed);
        areaRatioGraph->SetMaximum(1);
        areaRatioGraph->SetMinimum(0);
        areaRatioGraph->Draw("AP");
        radiusRatioGraph->SetMarkerStyle(41);
        radiusRatioGraph->SetMarkerSize(4);
        radiusRatioGraph->SetMarkerColor(kBlue);
        radiusRatioGraph->Draw("P");
        mExpSettingLegend->Draw("SAME");
        line1->SetLineColor(kPink + 2);
        // line1->Draw("SAME");
        line2->SetLineColor(kCyan - 7);
        // line2->Draw("SAME");
        TLegend* legend = new TLegend(.78, .7, .98, .92);
        legend->SetHeader("Ratios", "c");
        legend->AddEntry(areaRatioGraph, "Pixels / Full pixels", "p");
        legend->AddEntry(radiusRatioGraph, "Short radius / Long radius", "p");
        legend->Draw();
        canvas->SetMargin(.07, .28, .12, .08);
        std::filesystem::path file = filePath;

        file /= config.getSubConfig("graphs").find("filename") + "_" + std::to_string(clusterSize);
        file.replace_extension(config.find("extension"));

        canvas->SaveAs(static_cast<TString>(file));

        delete line1;
        delete line2;
        delete areaRatioGraph;
        delete radiusRatioGraph;
        delete canvas;
    }

    for ( const auto& plotConfig : plotConfigList ) {
        TString canvasName = "can" + plotConfig.first;
        Int_t canvasWidth = plotConfig.second.hasKey("canvas_width") ?
stoi(plotConfig.second.find("canvas_width")) : 500;
        Int_t canvasHeight = plotConfig.second.hasKey("canvas_height") ?
stoi(plotConfig.second.find("canvas_height")) : 500;
        TCanvas* canvas = new TCanvas(canvasName, "", canvasWidth, canvasHeight);

        distributionSet.find(plotConfig.first)->second->SetStats(0);

        Int_t y_min = plotConfig.second.hasKey("y_min") ? stoi(plotConfig.second.find("y_min")) : 0;
        Int_t y_max = plotConfig.second.hasKey("y_max") ? stoi(plotConfig.second.find("y_max")) :
1000;
        distributionSet.find(plotConfig.first)->second->SetMinimum(y_min);
        distributionSet.find(plotConfig.first)->second->SetMaximum(y_max);
        distributionSet.find(plotConfig.first)->second->Draw();

        std::filesystem::path file = filePath;
        file /= plotConfig.second.find("filename");
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));

        delete canvas;
        canvas = nullptr;
    }
}

void TAnalyser::saveSameSizeShapes(std::string_view typeName, const CppConfigDictionary config) {
    std::cout « "Generating " « "\033[1;32m" « "Shapes with same sized" « "\033[1;0m" « "..." «
std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;
```

```cpp

    const int nominalWidth = 200;
    const int nominalHeader = 100;
    const int nPlotInRow = 10;
    int plotsWidth = nPlotInRow * nominalWidth;

    ProgressBar pBar(nTotalShape);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        int nClusterShape = clusterShape->getClusterShapeInfos().size();

        int plotsHeight = (floor((nClusterShape - 1.) / 10.) + 1.) * nominalWidth;

        TCanvas* canvas = new TCanvas(Form("shapes%d", clusterSize), "", plotsWidth, plotsHeight + nominalHeader);

        int iClusterShape = 0;

        std::vector<TPad*> padSet;
        std::vector<TLine*> lineSet;
        std::vector<TText*> textSet;

        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();

            int width = shapeInfo.mClusterMatrix->getNRow() + 2;
            int height = shapeInfo.mClusterMatrix->getNColumn() + 2;

            double padCenterX = (1. / nPlotInRow) * (.5 + (iClusterShape % 10));
            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.);

            double padWidth = (1. / (2. * nPlotInRow));
            double padHeight = (static_cast<double>(plotsHeight) / (plotsHeight + nominalHeader)) / (2. * ((floor(static_cast<double>(nClusterShape) / nPlotInRow) + 1.)));

            TPad* pad = new TPad(Form("pad%d", iClusterShape), "", padCenterX - padWidth, padCenterY - padHeight, padCenterX + padWidth, padCenterY + padHeight);
            pad->SetMargin(0., 0., .5 * (1. - static_cast<double>(height) / width), .5 * (1. - static_cast<double>(height) / width));
            pad->Draw();
            pad->cd();
            shapeInfo.mClusterMap->SetDrawOption("COL");
            shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->Draw("col");
            pad->SetFrameLineWidth(0);
            padSet.push_back(pad);

            TLine* line = new TLine();
            line->SetLineColorAlpha(kRed, 6. / 8);
            for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
                for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
                    if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
                        double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
                        double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
                        double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
                        double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
                        line->DrawLine(xlow, ylow, xup, ylow); // Bottom
                        line->DrawLine(xlow, yup, xup, yup); // Top
                        line->DrawLine(xlow, ylow, xlow, yup); // Left
                        line->DrawLine(xup, ylow, xup, yup); // Right
                    }
                }
            }
            lineSet.push_back(line);

            TText* numberingText = new TText(padCenterX, padCenterY, Form("%d", iClusterShape));
            numberingText->SetNDC();
            numberingText->SetTextAlign(22);
            numberingText->SetTextSize(.3 * nominalWidth / (plotsHeight + nominalHeader));
            numberingText->SetTextColorAlpha(kBlack, 5. / 8.);

            canvas->cd();
            numberingText->Draw();
            textSet.push_back(numberingText);
            iClusterShape++;
        }
        TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight), Form("Cluster shapes with cluster size %d", clusterSize));
        titleText->SetTextSize(.5 * nominalHeader / (plotsHeight + nominalHeader));
        titleText->SetTextAlign(22);
        titleText->SetNDC();
        titleText->Draw();
        std::filesystem::path file = filePath / (config.find("filename") + "_" +
```

```
std::to_string(clusterSize));
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));

        delete titleText;
        titleText = nullptr;

        for ( TPad* pad : padSet ) {
            delete pad;
            pad = nullptr;
        }

        for ( TLine* line : lineSet ) {
            delete line;
            line = nullptr;
        }

        for ( TText* text : textSet ) {
            delete text;
            text = nullptr;
        }

        delete canvas;
        canvas = nullptr;
    }
}

void TAnalyser::saveTotalShapes(std::string_view typeName, const CppConfigDictionary config) {
    // Print log
    std::cout « "Generating \033[1;32mTotal shapes\033[1;0m..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    int nWidth = mMaxModeSet.find(std::string(typeName))->second;
    int nHeight = mClusterShapeSet.find(std::string(typeName))->second.size();

    int nominalWidth = 100;
    int nominalHeader = 100;

    int plotsWidth = nWidth * nominalWidth;
    int plotsHeight = nHeight * nominalWidth;
    TCanvas* canvas = new TCanvas("tShape", "cluster shape", plotsWidth + nominalHeader, plotsHeight + nominalHeader);
    int iClusterSize = 0;
    std::vector<TPad*> padSet;
    std::vector<TLine*> lineSet;
    std::vector<TText*> textSet;

    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        int iClusterShape = 0;
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();
            int width = shapeInfo.mClusterMap->GetNbinsX();
            int height = shapeInfo.mClusterMap->GetNbinsY();

            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);
            padSet.push_back(pad);
            pad->Draw();
            pad->cd();
            pad->SetMargin(0., 0., .5 * (1. - (double) height / width), .5 * (1. - (double) height / width));

            shapeInfo.mClusterMap->Draw();
            shapeInfo.mClusterMap->SetDrawOption("COL");
            shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
            TLine* line = new TLine();
            lineSet.push_back(line);
            line->SetLineColorAlpha(kRed, 6. / 8);
            for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
                for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
                    if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
                        double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
                        double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
                        double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
```

```
                        double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
                        line->DrawLine(xlow, ylow, xup, ylow); // Bottom
                        line->DrawLine(xlow, yup, xup, yup); // Top
                        line->DrawLine(xlow, ylow, xlow, yup); // Left
                        line->DrawLine(xup, ylow, xup, yup); // Right
                    }
                }
            }
            pad->SetFrameLineWidth(0);
            canvas->cd();
            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));
            textSet.push_back(numberingText);
            numberingText->SetNDC();
            numberingText->SetTextAlign(22);
            numberingText->SetTextSize(.4 * nominalWidth / (plotsHeight + nominalHeader));
            numberingText->SetTextColor(kBlack);
            numberingText->Draw();
            iClusterShape++;
        }
        TText* sizeText = new TText((double) nominalHeader * .5 / (nominalHeader + plotsWidth),
((double) plotsHeight / (plotsHeight + nominalHeader)) * (((double) nHeight - iClusterSize - .5) /
nHeight), Form("%d", clusterSize));
        textSet.push_back(sizeText);
        sizeText->SetNDC();
        sizeText->SetTextAlign(22);
        sizeText->SetTextSize(.6 * nominalWidth / (plotsHeight + nominalHeader));
        sizeText->SetTextColor(kBlack);
        sizeText->Draw();
        iClusterSize++;
    }
    TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight),
static_cast<TString>("Total Cluster Shapes"));
    titleText->SetTextSize(.8 * nominalHeader / (plotsHeight + nominalHeader));
    titleText->SetTextAlign(22);
    titleText->SetNDC();
    titleText->Draw();

    std::filesystem::path file = filePath / config.find("filename");
    file.replace_extension(config.find("extension"));
    canvas->SaveAs(static_cast<TString>(file));

    delete titleText;
    titleText = nullptr;

    for ( TPad* pad : padSet ) {
        delete pad;
        pad = nullptr;
    }

    for ( TLine* line : lineSet ) {
        delete line;
        line = nullptr;
    }

    for ( TText* text : textSet ) {
        delete text;
        text = nullptr;
    }

    delete canvas;
    canvas = nullptr;
}

void TAnalyser::saveSameSizeShapeEntry(std::string_view typeName, const CppConfigDictionary config) {
    std::cout « "Generating \033[1;32mEntry of shapes with same size\033[1;0m..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
        int binNum = clusterShape->getClusterShapeInfos().size();
        TH1D* distribution = new TH1D(Form("shapeEntry%d", clusterSize),
Form(static_cast<TString>("Shape Entry with cluster size %d"), clusterSize), binNum, -.5, binNum -
.5);
        int iShape = 0;
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();
            distribution->Fill(iShape, shapeInfo.mEntry);
```

```
            iShape++;
        }
        distribution->GetXaxis()->SetNdivisions(10, 10, 0, true);
        distribution->GetXaxis()->SetTitle("i-th cluster size");
        distribution->GetXaxis()->SetTitleOffset(1.4);
        distribution->GetXaxis()->SetLabelOffset(0.003);
        distribution->GetYaxis()->SetTitle("Entry");
        distribution->GetYaxis()->SetTitleOffset(0.7);
        distribution->Draw("HIST");
        mExpSettingLegend->Draw("SAME");
        canvas->SetMargin(.07, .28, .12, .08);

        std::filesystem::path file = filePath / (config.find("filename") + "_" + std::to_string(clusterSize));
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));

        delete distribution;
        delete canvas;
    }
}

void TAnalyser::saveTotalShapeEntry(std::string_view typeName, const CppConfigDictionary config) {
    std::cout « "Generating \033[1;32mEntry of total shapes\033[1;0m..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    int nXbin = 0;
    int nYbin = 0;
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        nYbin = std::max(nYbin, static_cast<int>(clusterShape->getClusterShapeInfos().size()));
        nXbin = std::max(nXbin, clusterShape->getClusterSize());
    }
    TCanvas* canvas = new TCanvas("shapeEntryTotal", "shape entry", 2500, 1000);
    TH2D* distribution = new TH2D("shapeEntryTotal", static_cast<TString>(config.find("plot_titles")), nXbin, -nXbin - .5, -.5, nYbin, -nYbin + .5, .5);
    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        int iShape = 0;
        for ( const TShapeInfo shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();
            distribution->Fill(-clusterSize, -iShape, shapeInfo.mEntry);
            iShape++;
        }
    }
    for ( int iXbin = 1; iXbin < distribution->GetNbinsX(); ++iXbin ) {
        distribution->GetXaxis()->SetBinLabel(iXbin, Form("%g", floor(-1 * distribution->GetXaxis()->GetBinLowEdge(iXbin))));
    }

    for ( int iYbin = 1; iYbin < distribution->GetNbinsY(); ++(++iYbin) ) {
        distribution->GetYaxis()->SetBinLabel(iYbin, Form("%g", floor(-1 * distribution->GetYaxis()->GetBinLowEdge(iYbin))));
    }

    distribution->SetStats(0);
    if ( config.hasKey("options") ) {
        for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
            distribution->Draw(static_cast<TString>(optionName));
            canvas->SetPhi(10);
            canvas->SetTheta(25);
            canvas->SetLogz();
            std::filesystem::path file = filePath;
            file /= (config.find("filename") + "_" + optionName);
            file.replace_extension(config.find("extension"));
            canvas->SaveAs(static_cast<TString>(file));
        }
    } else {
        distribution->Draw();
        std::filesystem::path file = filePath;
        file /= config.find("filename");
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));
    }

    delete distribution;
    delete canvas;
}
```

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);
```

```
    mThresholdmap = new TH2D("thrmap", "Threshold map", 1024, -0.5, 1023.5, 512, -0.5, 511.5);
    mChi2NdfDistribution = new TH1D("chi2", "Chi2 and NDoF", 200, 0, 200);
    int nHigh = 0;
    int nLow = 0;
    int nUnDefined = 0;
    for ( const TThreshold* threshold : mThresholds ) {
        if ( threshold->getCondition() == ThrCondition::good ) {
            mThresholdDistribution->Fill(threshold->getThreshold());
            mErrorDistribution->Fill(threshold->getError());
            mThresholdmap->SetBinContent(threshold->getX(), threshold->getY(),
threshold->getThreshold());
            mChi2NdfDistribution->Fill(threshold->getQualityFactor());
        } else if ( threshold->getCondition() == ThrCondition::bad_too_high ) {
            nHigh++;
            mThresholdmap->SetBinContent(threshold->getX(), threshold->getY(), 50);
        } else if ( threshold->getCondition() == ThrCondition::bad_too_low ) {
            nLow++;
            mThresholdmap->SetBinContent(threshold->getX(), threshold->getY(), 0);
        } else if ( threshold->getCondition() == ThrCondition::bad_undefine ) {
            nUnDefined++;
        }
    }
    std::cout « mThresholdDistribution->GetMean() « std::endl;
    std::cout « mThresholdDistribution->GetStdDev() « std::endl;
    std::cout « mErrorDistribution->GetMean() « std::endl;
    std::cout « mErrorDistribution->GetStdDev() « std::endl;
    std::cout « mThresholdDistribution->GetEntries() « std::endl;
    std::cout « nLow « std::endl;
    std::cout « nHigh « std::endl;
    std::cout « nUnDefined « std::endl;
}

void TThresholdAnalyser::saveThresholdDistribution(std::string_view title) const {
    TCanvas* can = new TCanvas("thrDist", "Threshold Distribution; Threshold [$500 \times e^-$];
Entry", 1000, 500);
    mThresholdDistribution->Draw();
    can->SaveAs(static_cast<const TString>(title));
}

void TThresholdAnalyser::saveErrorDistribution(std::string_view title) const {
    TCanvas* can = new TCanvas("errDist", "Error Distribution;Threshold [$e^-$]; Entry", 1000, 500);
    mErrorDistribution->Draw();
    can->SaveAs(static_cast<const TString>(title));
}

void TThresholdAnalyser::saveThresholdmap(std::string_view title) const {
    TCanvas* can(new TCanvas("thrmap", "", 1000, 500));
    mThresholdmap->SetTitle("Threshold Distribution; rows; coloums");
    mThresholdmap->SetMinimum(0);
    mThresholdmap->SetMaximum(50);
    mThresholdmap->SetStats(0);
    mThresholdmap->Draw("COLZ0");
    can->SaveAs(static_cast<const TString>(title));
}

void TThresholdAnalyser::saveQualityDistribution(std::string_view title) const {
    TCanvas* can = new TCanvas("chi2", "Fit quality Distribution; Chi2/NDoF; entry", 1000, 500);
    mChi2NdfDistribution->Draw();
    can->SaveAs(static_cast<const TString>(title));
}
```

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.

```
#ifndef __TCLUSTER__
#define __TCLUSTER__

#ifdef __TCLUSTER_HEADER__
#endif

#include<vector>
#include<algorithm>
#include<cmath>

class TCluster {
private:
    int mEvent;
    int mTime;
    std::vector <std::pair<int, int» mPixels;
    int mMinX = 1024, mMinY = 512;
    int mMaxX = 0, mMaxY = 0;
    std::pair<double, double> center;
    double mLongRadius;
    // std::pair<double, double> stdevInAxis; /**< standard deviation in x and y direction */
    // double stdev; /**< Root of mean of squared distance from cluster centre */
    int size = 0;
    // TMatrix2D<int> shape; /**< The shape matrix for analyse cluster shape */

public:
    TCluster();
    TCluster(int event, int time);
    TCluster(const TCluster& copy);
    TCluster& operator=(const TCluster& copy);
    TCluster(TCluster&& move);
    TCluster& operator=(TCluster&& move);
    ~TCluster();

    // Add Function
    void AddPixel(const std::pair<int, int>& pixel);
    void AddCluster(const TCluster& cluster);

    // Compare functions
    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 calStdevInAxis();
    // void calStdev();
    void calSize();
    // void calShape();

    // Getter
    const std::pair<double, double> getCenter() const;
    const double getLongRadius() const;
    // const std::pair<double, double> getStdevInAxis() const;
    // const double getStdev() const;
    const int getSize() const;
    // const TMatrix2D<int>& getShape() const;

    // Setter for member
    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);

    // Getter for member
    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;

private:
    unsigned int fBits;
public:
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
};

#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.

```
#ifndef __TCLUSTERDIVIDEDATA__
#define __TCLUSTERDIVIDEDATA__
```

```

#ifdef __TCLUSTERDIVIDEDATA_HEADER__
#include <iostream>
#include "TCluster.h"
#endif

#include<unordered_map>
#include<vector>

class TCluster;

class TClusterDivideData {
private:
    std::unordered_map<int, std::vector<TCluster*>> mClusterData;
public:
    TClusterDivideData(const std::vector<TCluster*>& clusters);
    TClusterDivideData(const TClusterDivideData& copy);
    const std::vector<TCluster*>& getClusterOfSize(const int clusterSize);
};

#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 (yochoi@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.

```cpp

#ifndef __TCLUSTERIZATION__
#define __TCLUSTERIZATION__

#ifdef __TCLUSTERIZATION_HEADER__
#include "cpptqdm.h"

#include "TALPIDEEvent.h"
#include "TCluster.h"
#endif

#include <vector>

class TALPIDEEvent;
class TCluster;

class TClusterization {
    std::vector<TALPIDEEvent*> mEvents;
    std::vector<TCluster*> mClusters;
public:
    TClusterization() = delete;
    TClusterization(const std::vector<TALPIDEEvent*>& events);
    ~TClusterization();

    void removeConsecutionPixels(); // Remove later pixels if it fired continuously.

    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;
};

#endif
```

8.17 /home/ychoi/ATOM/alpide/cluster/inc/TClusterShape.h File Reference

Tools for extracting cluster shape image.

```cpp
#include <vector>
#include <unordered_map>
```

Classes

- struct TShapeInfo

    The information set stucture 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.

Author

Yongjun Choi (yochoi@cern.ch)

Version

0.1

Date

16-04-2024

Copyright

Definition in file TClusterShape.h.

8.18 TClusterShape.h

Go to the documentation of this file.

```
#ifndef __TCLUSTERSHAPE__
#define __TCLUSTERSHAPE__

#ifdef __TCLUSTERSHAPE_HEADER__
#include "TCluster.h"
#include "TH2I.h"
#include "TMatrix2D.h"
#endif

#include<vector>
#include<unordered_map>

class TH2I;

class TCluster;
template<typename T> class TMatrix2D;

struct TShapeInfo {
    TCluster* mPresidentCluster;
    TMatrix2D<int>* mClusterMatrix;
    TH2I* mClusterMap;
    int iShape;
    int mEntry;
    int mShortBinN;
    int mLongBinN;
};

class TClusterShape {
private:
    int mClusterSize;
    std::vector<TCluster*> mClusterOriginSet;
    std::vector<TShapeInfo> mClusterShapeInfos;
public:
    TClusterShape();
    TClusterShape(const int clusterSize, const std::vector<TCluster*>& clusters);
    TClusterShape(const std::vector<TCluster*> clusters);
    // TClusterShape(const std::vector<TCluster*> clusters, const int clusterSize);
    ~TClusterShape();

    void identifyShapes();
    void sortShapes(bool descend = true);
    void calClusterInfo(TShapeInfo& shapeInfo, TCluster* cluster);
    TMatrix2D<int>* clusterMatrix(const TCluster* cluster);
    TH2I* clusterMap(const TMatrix2D<int>* clusterMatrix);

    const std::vector<TShapeInfo>& getClusterShapeInfos() const;

    void setClusterSize(const int ClusterSize);
```

```
    const int getClusterSize() const;
private:
    unsigned int fBits;
public:
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
};

#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 (yochoi@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.

```
#ifndef __TEXPERIMENTDATA__
#define __TEXPERIMENTDATA__

#ifdef __TEXPERIMENTDATA_HEADER__
#include "TALPIDEEvent.h"
#include "TCluster.h"
#endif

#include<vector>

class TALPIDEEvent;
class TCluster;

class TExperimentData {
private:
    std::vector<TALPIDEEvent*> mEvents;
    std::vector<TCluster*> mClusters;
    // std::vector<TALPIDEEvent*> mMaskedEvents;
    // std::vector<TCluster*> mMaskedClusters;
    // std::vector<TALPIDEEvent*> mNoiseEvents;
    // std::vector<TCluster*> mNoiseClusters;

public:
    TExperimentData();
    TExperimentData(const TExperimentData& copy);
    TExperimentData& operator=(const TExperimentData& copy);
    ~TExperimentData();

    void setEvents(const std::vector<TALPIDEEvent*>& events);
    void setClusters(const std::vector<TCluster*>& clusters);
    // void setMaskedEvents(const std::vector<TALPIDEEvent*>&& maskedEvents);
    // void setMaskedClusters(const std::vector<TCluster*>&& maskedClusters);
    // void setNoiseEvents(const std::vector<TALPIDEEvent*>&& noiseEvents);
    // void setNoiseClusters(const std::vector<TCluster*>&& noiseClusters);
    const std::vector<TALPIDEEvent*> getEvents() const;
    const std::vector<TCluster*> getClusters() const;
    // const std::vector<TALPIDEEvent*> getMaskedEvents() const;
    // const std::vector<TCluster*> getMaskedClusters() const;
    // const std::vector<TALPIDEEvent*> getNoiseEvents() const;
    // const std::vector<TCluster*> getNoiseClusters() const;

private:
    unsigned int fBits;
public:
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
};

#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.

```
#ifndef __TMATRIX2D__
#define __TMATRIX2D__

#include <iostream>

template <typename numT>
class TMatrix2D;

template<typename numT>
std::ostream& operator« (std::ostream& os, const TMatrix2D<numT>& matrix);

template <typename numT>
class TMatrix2D {
private:
    std::vector<std::vector<numT» mMatrix;
    int mNRow, mNColumn;
public:
    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);

    friend std::ostream& operator«<>(std::ostream& os, const TMatrix2D<numT>& 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 numT>
TMatrix2D<numT>::TMatrix2D() { }

template<typename numT>
TMatrix2D<numT>::TMatrix2D(int nRow, int nColumn) : mNRow(nRow), mNColumn(nColumn) {
    for ( int iRow = 0; iRow < nRow; iRow++ ) {
        std::vector<numT> temp;
        for ( int iColumn = 0; iColumn < nColumn; iColumn++ ) {
            temp.push_back(0);
        }
        mMatrix.push_back(temp);
    }
}

template<typename numT>
TMatrix2D<numT>::TMatrix2D(const std::vector<std::vector<numT»& matrix) {
    mNRow = matrix.size();
    mNColumn = matrix[0].size();
    mMatrix.assign(matrix.begin(), matrix.end());
}

template<typename numT>
void TMatrix2D<numT>::setMatrix(const std::vector<std::vector<numT»& matrix) {
    mNRow = matrix.size();
    mNColumn = matrix[0].size();
    mMatrix.clear();
    mMatrix.assign(matrix.begin(), matrix.end());
}

template<typename numT>
void TMatrix2D<numT>::setElement(const int iRow, const int iColumn, const numT element) {
    mMatrix[iRow][iColumn] = element;
}

template<typename numT>
const numT TMatrix2D<numT>::getElement(const int iRow, const int iColumn) const {
    try {
        if ( iRow < 0 || iRow > mNRow ) throw iRow;
        if ( iColumn < 0 || iColumn > mNColumn ) throw iColumn;
        return mMatrix[iRow][iColumn];
    } catch ( int iRow ) {
        std::cerr « "The row index shoud be positive and smaller than " « mNRow « ". Current: " « iRow
« std::endl;
        return mMatrix[0][0];
    }
}

template<typename numT>
const int TMatrix2D<numT>::getNRow() const {
    return mNRow;
}

template<typename numT>
const int TMatrix2D<numT>::getNColumn() const {
    return mNColumn;
}

template<typename numT>
const std::pair<int, int> TMatrix2D<numT>::getDimension() const {
    return {mNRow, mNColumn};
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::isSameDimension(const TMatrix2D<numT2>& matrix) {
    if ( mNRow == matrix.getNRow() && mNColumn == matrix.getNColumn() ) {
        return true;
    } else {
        return false;
    }
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::isSame(const TMatrix2D<numT2>& matrix) {
    for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
```

```
            if ( mMatrix[iRow][iColumn] != matrix.getElement(iRow, iColumn) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasYSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(mNRow - iRow - 1, iColumn) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasXSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(iRow, mNColumn - iColumn - 1) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasXYSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(mNRow - iRow - 1, mNColumn - iColumn - 1) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasRSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(mNColumn - iColumn - 1, iRow) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasRXSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(iColumn, iRow) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasRYSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(mNColumn - iColumn - 1, mNRow - iRow - 1) ) {
                return false;
            }
        }
    }
    return true;
}
```

```
template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasRXYSymmetry(const TMatrix2D<numT2>& matrix) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            if ( mMatrix[iRow][iColumn] != matrix.getElement(iColumn, mNRow - iRow - 1) ) {
                return false;
            }
        }
    }
    return true;
}

template<typename numT>
template<typename numT2>
bool TMatrix2D<numT>::hasHomeomorphism(const TMatrix2D<numT2>& matrix) {
    if ( mNRow == mNColumn && mNRow == matrix.getNRow() && mNColumn == matrix.getNColumn() ) {
        if ( isSame(matrix) ) {
            return true;
        } else if ( hasXSymmetry(matrix) ) {
            return true;
        } else if ( hasYSymmetry(matrix) ) {
            return true;
        } else if ( hasXYSymmetry(matrix) ) {
            return true;
        } else if ( hasRSymmetry(matrix) ) {
            return true;
        } else if ( hasRXSymmetry(matrix) ) {
            return true;
        } else if ( hasRYSymmetry(matrix) ) {
            return true;
        } else if ( hasRXYSymmetry(matrix) ) {
            return true;
        } else {
            return false;
        }
    } else if ( mNRow == matrix.getNRow() && mNColumn == matrix.getNColumn() ) {
        if ( isSame(matrix) ) {
            return true;
        } else if ( hasXSymmetry(matrix) ) {
            return true;
        } else if ( hasYSymmetry(matrix) ) {
            return true;
        } else if ( hasXYSymmetry(matrix) ) {
            return true;
        } else {
            return false;
        }
    } else if ( mNRow == matrix.getNColumn() && mNColumn == matrix.getNRow() ) {
        if ( hasRSymmetry(matrix) ) {
            return true;
        } else if ( hasRXSymmetry(matrix) ) {
            return true;
        } else if ( hasRYSymmetry(matrix) ) {
            return true;
        } else if ( hasRXYSymmetry(matrix) ) {
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
}

template<typename numT>
std::ostream& operator«(std::ostream& os, const TMatrix2D<numT>& matrix) {
    for ( const std::vector<numT>& aRow : matrix.mMatrix ) {
        for ( const numT& element : aRow ) {
            os « element « " ";
        }
        os « std::endl;
    }
    os « "    Dimension: " « matrix.mNRow « "X" « matrix.mNColumn;
    return os;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator+=(const TMatrix2D<numT2>& matrix) {
    try {
        if ( !isSameDimension(matrix) ) throw matrix;
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
                mMatrix[iRow][iColumn] += matrix.getElement(iRow, iColumn);
            }
```

```
        }
    } catch ( TMatrix2D<numT> matrix ) {
        std::cerr « "The dimension should be same for plus operator. (" « mNRow « "X" « mNColumn « " != " « matrix.mNRow « "X" « matrix.mNColumn « ")" « std::endl;
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator+(const TMatrix2D<numT2>& matrix) {
    try {
        if ( !isSameDimension(matrix) ) throw matrix;
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
                mMatrix[iRow][iColumn] += matrix.getElement(iRow, iColumn);
            }
        }
    } catch ( TMatrix2D<numT> matrix ) {
        std::cerr « "The dimension should be same for plus operator. (" « mNRow « "X" « mNColumn « " != " « matrix.mNRow « "X" « matrix.mNColumn « ")" « std::endl;
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator-=(const TMatrix2D<numT2>& matrix) {
    try {
        if ( !isSameDimension(matrix) ) throw matrix;
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
                mMatrix[iRow][iColumn] -= matrix.getElement(iRow, iColumn);
            }
        }
    } catch ( TMatrix2D<numT> matrix ) {
        std::cerr « "The dimension should be same for minus operator. (" « mNRow « "X" « mNColumn « " != " « matrix.mNRow « "X" « matrix.mNColumn « ")" « std::endl;
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator-(const TMatrix2D<numT2>& matrix) {
    try {
        if ( !isSameDimension(matrix) ) throw matrix;
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
                mMatrix[iRow][iColumn] -= matrix.getElement(iRow, iColumn);
            }
        }
    } catch ( TMatrix2D<numT> matrix ) {
        std::cerr « "The dimension should be same for minus operator. (" « mNRow « "X" « mNColumn « " != " « matrix.mNRow « "X" « matrix.mNColumn « ")" « std::endl;
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator*=(const numT2 scalar) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            mMatrix[iRow][iColumn] *= scalar;
        }
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator*(const numT2 scalar) {
    for ( int iRow = 0; iRow < mNRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < mNColumn; iColumn++ ) {
            mMatrix[iRow][iColumn] *= scalar;
        }
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator*=(const TMatrix2D<numT2>& matrix) {
    try {
        if ( mNRow != matrix.getNColumn() ) throw matrix;

```

```
        TMatrix2D<numT> newMatrix(mNRow, matrix.getNColumn());
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            for ( int iColumn = 0; iColumn < matrix.getNColumn(); iColumn++ ) {
                for ( int iOperand = 0; iOperand < mNColumn; iOperand++ ) {
                    newMatrix.setElement(iRow, iColumn, newMatrix.getElement(iRow, iColumn) + mMatrix[iRow][iOperand] * matrix.getElement(iOperand, iColumn));
                }
            }
        }
        *this = newMatrix;
    } catch ( TMatrix2D<numT> matrix ) {
        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;
    }
    return *this;
}

template<typename numT>
template<typename numT2>
TMatrix2D<numT>& TMatrix2D<numT>::operator*(const TMatrix2D<numT2>& matrix) {
    try {
        if ( mNRow != matrix.getNColumn() ) throw matrix;

        TMatrix2D<numT> newMatrix(mNRow, matrix.getNColumn());
        for ( int iRow = 0; iRow < mNRow; iRow++ ) {
            for ( int iColumn = 0; iColumn < matrix.getNColumn(); iColumn++ ) {
                for ( int iOperand = 0; iOperand < mNColumn; iOperand++ ) {
                    newMatrix.setElement(iRow, iColumn, newMatrix.getElement(iRow, iColumn) + mMatrix[iRow][iOperand] * matrix.getElement(iOperand, iColumn));
                }
            }
        }
        *this = newMatrix;
    } catch ( TMatrix2D<numT> matrix ) {
        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;
    }
    return *this;
}




#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.

```
#define __TCLUSTER_HEADER__
#include "TCluster.h"

TCluster::TCluster() : fBits(kNotDeleted) { };

TCluster::TCluster(int event, int time) : mEvent(event), mTime(time), fBits(kNotDeleted) { }

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) {
    mPixels.assign(copy.mPixels.begin(), copy.mPixels.end());
}

TCluster& TCluster::operator=(const TCluster& copy) {
    mEvent = copy.mEvent;
    mTime = copy.mTime;
    mPixels.assign(copy.mPixels.begin(), copy.mPixels.end());
    mMinX = copy.mMinX;
    mMinY = copy.mMinY;
    mMaxX = copy.mMaxX;
    mMaxY = copy.mMaxY;

    return *this;
}

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) { }

TCluster& TCluster::operator=(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;
    return *this;
}

TCluster::~TCluster() { }

void TCluster::AddPixel(const std::pair<int, int>& pixel) {
    mPixels.push_back(pixel);
    mMaxY = std::max(mMaxY, pixel.second);
}

void TCluster::AddCluster(const TCluster& cluster) {
    for ( const std::pair<int, int>& pixel : cluster.getPixels() ) {
        mPixels.push_back(pixel);
    }
    mMaxY = std::max(mMaxY, cluster.getMaxY());
}

bool TCluster::isNeighbour(const std::pair<int, int>& pixel) const {
    bool neighbour = false;
    for ( std::pair<int, int> mPixel : mPixels ) {
        if ( getDistance(pixel, mPixel) <= 1 ) {
            neighbour = true;
            break;
        }
    }
    return neighbour;
}

bool TCluster::isNeighbour(const TCluster& cluster) const {
    bool neighbour = false;
    for ( std::pair<int, int> comparePixel : cluster.getPixels() ) {
        neighbour = isNeighbour(comparePixel);
        if ( neighbour ) break;
    }
    return neighbour;
}

bool TCluster::isContain(const std::pair<int, int>& pixel) const {
    bool neighbour = false;
    if ( find(mPixels.begin(), mPixels.end(), pixel) != mPixels.end() ) {
        neighbour = true;
    }
    return neighbour;
}

bool TCluster::isContain(const TCluster& cluster) const {
```

```
    bool neighbour = false;
    for ( std::pair<int, int> comparePixel : cluster.getPixels() ) {
        neighbour = isContain(comparePixel);
        if ( neighbour ) break;
    }
    return neighbour;
}

const int TCluster::getDistance(const std::pair<int, int>& pixel1, const std::pair<int, int>& pixel2) const {
    return abs(pixel1.first - pixel2.first) + abs(pixel1.second - pixel2.second);
}

void TCluster::calMembers() {
    calMinMax();
    calCenter();
    calSize();
    calLongRadius();
    // calShape();
    // calStdevInAxis();
    // calStdev();
}

void TCluster::calMinMax() {
    for ( std::pair<int, int> pixel : mPixels ) {
        mMinX = std::min(pixel.first, mMinX);
        mMinY = std::min(pixel.second, mMinY);
        mMaxX = std::max(pixel.first, mMaxX);
        mMaxY = std::max(pixel.second, mMaxY);
    }
}

void TCluster::calCenter() {
    double x = 0., y = 0.;
    for ( std::pair<int, int> pixel : mPixels ) {
        x += pixel.first;
        y += pixel.second;
    }
    x = (double) x / mPixels.size();
    y = (double) y / mPixels.size();
    center = {x, y};
}

void TCluster::calSize() {
    size = mPixels.size();
}

void TCluster::calLongRadius() {
    double maxDistance = 0.;
    for ( const std::pair<int, int>& pixel : mPixels ) {
        double distance = pow(abs(pixel.first - center.first) + .5, 2) + pow(abs(pixel.second - center.second) + .5, 2);
        maxDistance = std::max(maxDistance, distance);
    }
    mLongRadius = sqrt(maxDistance);
}

const std::pair<double, double> TCluster::getCenter() const {
    return center;
}

const int TCluster::getSize() const {
    return size;
}

const double TCluster::getLongRadius() const {
    return mLongRadius;
}

// Setter for member
void TCluster::setEvent(const int event) { mEvent = event; }
void TCluster::setTimeStamp(const int time) { mTime = time; }
void TCluster::setMinX(const int minX) { mMinX = minX; }
void TCluster::setMinY(const int minY) { mMinY = minY; }
void TCluster::setMaxX(const int maxX) { mMaxX = maxX; }
void TCluster::setMaxY(const int maxY) { mMaxY = maxY; }

// Getter for member
const std::vector <std::pair<int, int» TCluster::getPixels() const { return mPixels; }
const int TCluster::getEvent() const { return mEvent; }
const int TCluster::getTimeStamp() const { return mTime; }
const int TCluster::getMinX() const { return mMinX; }
const int TCluster::getMinY() const { return mMinY; }
const int TCluster::getMaxX() const { return mMaxX; }
const int TCluster::getMaxY() const { return mMaxY; }
```

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.

```
#define __TCLUSTERDIVIDEDATA_HEADER__
#include "TClusterDivideData.h"

TClusterDivideData::TClusterDivideData(const std::vector<TCluster*>& clusters) {
    for ( TCluster* cluster : clusters ) {
        int clusterSize = cluster->getSize();
        if ( mClusterData.count(clusterSize) ) {
            mClusterData.find(clusterSize)->second.push_back(cluster);
        } else {
            std::vector<TCluster*> set;
            set.push_back(cluster);
            mClusterData.insert_or_assign(clusterSize, set);
        }
    }
}

TClusterDivideData::TClusterDivideData(const TClusterDivideData& copy) :
      mClusterData(copy.mClusterData) { }


const std::vector<TCluster*>& TClusterDivideData::getClusterOfSize(const int clusterSize) {
    try {
        if ( !mClusterData.count(clusterSize) ) throw clusterSize;
        return mClusterData.find(clusterSize)->second;
    } catch ( int clusterSize ) {
        std::cerr « "There are no clusters having cluster Size " + std::to_string(clusterSize) «
      std::endl;
        std::vector<TCluster*>* empty;
        return *empty;
    }
}
```

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<TALPIDEEvent*>& 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 -
      1]->getData().end(), mEvents[iEvent]->getData()[iPixel]) != mEvents[iEvent - 1]->getData().end() ||
      (!preDeletedPixel.empty() && find(preDeletedPixel.begin(), preDeletedPixel.end(),
      mEvents[iEvent]->getData()[iPixel]) != preDeletedPixel.end()) ) {
00019                 currentDeletedPixel.push_back(mEvents[iEvent]->getData()[iPixel]);
00020                 mEvents[iEvent]->removePixel(mEvents[iEvent]->getData()[iPixel]);
00021             }
00022         }
00023         preDeletedPixel.clear();
00024         preDeletedPixel = std::move(currentDeletedPixel);
00025         currentDeletedPixel.clear();
00026     }
00027 }
00028
00029 void TClusterization::addNewCluster(const std::pair<int, int>& pixel, std::vector<TCluster*>&
      clusters, int iEvent, int iTime) {
00030     TCluster* cluster = new TCluster(iEvent, iTime);
00031     cluster->AddPixel(pixel);
00032     clusters.push_back(cluster);
00033 }
00034
00035 void TClusterization::removeIndependentCluster(const std::pair<int, int>& pixel,
      std::vector<TCluster*>& clusters) {
00036     if ( clusters[0]->getMaxY() + 2 < pixel.second ) {
00037         int cut = 0;
00038         for ( const TCluster* cluster : clusters ) {
00039             if ( cluster->getMaxY() + 2 < pixel.second ) {
00040                 cut++;
00041             } else break;
00042         }
00043         mClusters.insert(mClusters.end(), clusters.begin(), clusters.begin() + cut);
00044         clusters.erase(clusters.begin(), clusters.begin() + cut);
00045     }
00046 }
00047
00048 bool TClusterization::clusterLitmusTest(const std::pair<int, int>& pixel, std::vector<TCluster*>&
      clusters) {
00049     bool clusterExist = false;
00050     int iCluster = 0;
00051     int iOrigin = 0;
00052     for ( TCluster*& clustered : clusters ) {
```

```
        if ( clustered->isNeighbour(pixel) ) {
            if ( clusterExist == false ) {
                clustered->AddPixel(pixel);
                clusterExist = true;
                iOrigin = iCluster;
            } else {
                clusters[iOrigin]->AddCluster(*clustered);
                clusters.erase(clusters.begin() + iCluster);
                break;
            }
        }
        iCluster++;
    }

    return clusterExist;
}

void TClusterization::clusterize() {
    std::clog « "Generate clusters from events..." « std::endl;

    ProgressBar* pBar = new ProgressBar(mEvents.size());

    for ( TALPIDEEvent*& event : mEvents ) {
        pBar->printProgress();
        std::vector<TCluster*> clusterCandidate;
        bool isFirst = true;
        for ( const std::pair<int, int>& pixel : event->getData() ) {
            if ( isFirst ) {
                addNewCluster(pixel, clusterCandidate, event->getEvent(), event->getTime());
                isFirst = false;
                continue;
            }
            removeIndependentCluster(pixel, clusterCandidate);
            bool clusterExist = clusterLitmusTest(pixel, clusterCandidate);
            if ( !clusterExist ) {
                addNewCluster(pixel, clusterCandidate, event->getEvent(), event->getTime());
            } // Make new cluster if no relation with any cluster candidates.
        }
        if ( !clusterCandidate.empty() ) {
            mClusters.insert(mClusters.end(), clusterCandidate.begin(), clusterCandidate.end());
            clusterCandidate.clear();
        } // Move remained cluster candidates to cluster
    }
    delete pBar;
    pBar = nullptr;
    std::clog « "Total " « mClusters.size() « " clusters are extracted from " « mEvents.size() « "
events." « std::endl « std::endl;
    for ( TCluster* cluster : mClusters ) {
        cluster->calMembers();
    }
}

const std::vector<TCluster*>& TClusterization::getClusters() const {
    return mClusters;
}
```

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.]

```
#include "TCluster.h"

bool TCluster::operator==(const TCluster& cluster) const {
    int aX = mMaxX - mMinX;
    int aY = mMaxY - mMinY;
    int bX = cluster.mMaxX - cluster.mMinX;
    int bY = cluster.mMaxY - cluster.mMinY;
    if ( mPixels.size() != cluster.mPixels.size() ) {
```

```
        return false;
    }
    if ( !(((aX == bX) && (aY == bY)) || ((aX == bY) && (aY == bX))) ) {
        return false;
    }
    return true;
}

bool TCluster::operator!=(const TCluster& cluster) const {
    int aX = mMaxX - mMinX;
    int aY = mMaxY - mMinY;
    int bX = cluster.mMaxX - cluster.mMinX;
    int bY = cluster.mMaxY - cluster.mMinY;

    if ( mPixels.size() != cluster.mPixels.size() ) {
        return true;
    }

    if ( !(((aX == bX) && (aY == bY)) || ((aX == bY) && (aY == bX))) ) {
        return true;
    }
    return true;
}
```

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) :
      mClusterSize(clusterSize), fBits(kNotDeleted) {
00020     mClusterOriginSet.assign(clusters.begin(), clusters.end());
00021 }
00022
00023 // TClusterShape::TClusterShape(const std::vector<TCluster*> clusters, const int clusterSize) :
      mClusterSize(clusterSize) {
00024 //   //  for ( TCluster* cluster : clusters ) {
00025 //   //      if ( cluster->getSize() == clusterSize ) {
00026 //   //          mClusterWithN.push_back(cluster);
00027 //   //      }
```

```
// //  }
// }

TClusterShape::TClusterShape(const std::vector<TCluster*> clusters) {
    // mClusterWithN.assign(clusters.begin(), clusters.end());
}

TClusterShape::~TClusterShape() {
    for ( TShapeInfo shapeInfo : mClusterShapeInfos ) {
        delete shapeInfo.mClusterMap;
        delete shapeInfo.mClusterMatrix;

    }
}

void TClusterShape::identifyShapes() {
    bool isFirst = true;
    int iShape = 0;
    for ( TCluster* cluster : mClusterOriginSet ) {
        if ( isFirst ) {
            TShapeInfo shapeInfo;
            shapeInfo.mPresidentCluster = cluster;
            shapeInfo.mEntry = 1;
            shapeInfo.iShape = iShape;
            iShape++;
            calClusterInfo(shapeInfo, cluster);
            shapeInfo.mClusterMap = clusterMap(shapeInfo.mClusterMatrix);
            mClusterShapeInfos.push_back(shapeInfo);
            std::vector<TCluster*> newClusterSet;
            isFirst = false;
            continue;
        }
        TMatrix2D<int>* checkingCluster = clusterMatrix(cluster);
        bool isHomoemorphismExist = false;
        for ( TShapeInfo& shapeInfo : mClusterShapeInfos ) {
            TMatrix2D<int>* comparedCluster = shapeInfo.mClusterMatrix;
            if ( comparedCluster->hasHomeomorphism(*checkingCluster) ) {
                shapeInfo.mEntry++;
                isHomoemorphismExist = true;
                // mClusterSameSizeSet.find(shapeInfo.iShape)->second.push_back(cluster);
                break;
            }
        }

        if ( !isHomoemorphismExist ) {
            TShapeInfo shapeInfo;
            shapeInfo.mPresidentCluster = cluster;
            shapeInfo.mEntry = 1;
            shapeInfo.iShape = iShape;
            iShape++;
            calClusterInfo(shapeInfo, cluster);
            shapeInfo.mClusterMap = clusterMap(shapeInfo.mClusterMatrix);
            mClusterShapeInfos.push_back(shapeInfo);
            std::vector<TCluster*> newClusterSet;
        }
    }
}
void TClusterShape::sortShapes(bool descend) {
    std::sort(mClusterShapeInfos.begin(), mClusterShapeInfos.end(),
              [ ](TShapeInfo& a, TShapeInfo& b) {
                  if ( a.mEntry != b.mEntry ) {
                      return a.mEntry > b.mEntry;
                  } else if ( a.mClusterMatrix->getNRow() != b.mClusterMatrix->getNRow() ) {
                      return a.mClusterMatrix->getNRow() < b.mClusterMatrix->getNRow();
                  } else {
                      return a.mClusterMatrix->getNColumn() < b.mClusterMatrix->getNColumn();
                  }
              }
    );
}

TH2I* TClusterShape::clusterMap(const TMatrix2D<int>* clusterMatrix) {
    static int numbering;
    int nRow = clusterMatrix->getNRow();
    int nColumn = clusterMatrix->getNColumn();

    TH2I* map = new TH2I(Form("_%d", numbering), "", nRow + 2, 0, nRow + 2, nColumn + 2, 0, nColumn + 2);
    for ( int iRow = 0; iRow < nRow; iRow++ ) {
        for ( int iColumn = 0; iColumn < nColumn; iColumn++ ) {
            if ( clusterMatrix->getElement(iRow, iColumn) == 1 ) {
                map->Fill(iRow + 1, iColumn + 1);
            }
        }
    }

    map->GetXaxis()->SetNdivisions(nRow + 2, 0, 0, true);
```

```
    for ( int i = 1; i <= map->GetNbinsX(); ++i ) {
        map->GetXaxis()->SetBinLabel(i, "");
    }
    for ( int i = 1; i <= map->GetNbinsY(); ++i ) {
        map->GetYaxis()->SetBinLabel(i, "");
    }
    map->GetYaxis()->SetNdivisions(nColumn + 2, 0, 0, true);
    map->GetZaxis()->SetNdivisions(0, 0, 0, true);
    map->SetStats(0);

    numbering++;
    return map;
}


void TClusterShape::calClusterInfo(TShapeInfo& shapeInfo, TCluster* cluster) {
    int xBinN = cluster->getMaxX() - cluster->getMinX();
    int yBinN = cluster->getMaxY() - cluster->getMinY();
    bool longHeight = yBinN > xBinN ? true : false;
    shapeInfo.mLongBinN = longHeight ? yBinN : xBinN;
    shapeInfo.mShortBinN = longHeight ? xBinN : yBinN;

    shapeInfo.mClusterMatrix = new TMatrix2D<int>(shapeInfo.mLongBinN + 1, shapeInfo.mShortBinN + 1);
    if ( longHeight ) {
        for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
            shapeInfo.mClusterMatrix->setElement(pixel.second - cluster->getMinY(), pixel.first -
    cluster->getMinX(), 1);
        }
    } else {
        for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
            shapeInfo.mClusterMatrix->setElement(pixel.first - cluster->getMinX(), pixel.second -
    cluster->getMinY(), 1);
        }
    }

}

TMatrix2D<int>* TClusterShape::clusterMatrix(const TCluster* cluster) {
    int xBinN = cluster->getMaxX() - cluster->getMinX();
    int yBinN = cluster->getMaxY() - cluster->getMinY();
    TMatrix2D<int>* matrix = new TMatrix2D<int>(xBinN + 1, yBinN + 1);
    for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
        matrix->setElement(pixel.first - cluster->getMinX(), pixel.second - cluster->getMinY(), 1);
    }
    return matrix;
}

void TClusterShape::setClusterSize(const int clusterSize) {
    mClusterSize = clusterSize;
}
const std::vector<TShapeInfo>& TClusterShape::getClusterShapeInfos() const {
    return mClusterShapeInfos;
}
const int TClusterShape::getClusterSize() const {
    return mClusterSize;
}
```

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.

```
#define __TEXPERIMENTDATA_HEADER__
#include "TExperimentData.h"
#include<iostream>
TExperimentData::TExperimentData() : fBits(kNotDeleted) { }

TExperimentData::TExperimentData(const TExperimentData& copy) : mEvents(copy.mEvents),
mClusters(copy.mClusters) {
    std::cout « "mEvets" « mEvents.size() « std::endl;
    std::cout « "mClusters" « mClusters.size() « std::endl;

}

TExperimentData& TExperimentData::operator=(const TExperimentData& copy) {
    mEvents.clear();
    mEvents.assign(copy.mEvents.begin(), copy.mEvents.end());

    mClusters.clear();
    mClusters.assign(copy.mClusters.begin(), copy.mClusters.end());

    fBits = copy.fBits;
    return *this;
}


TExperimentData::~TExperimentData() {
    for ( TALPIDEEvent* event : mEvents ) {
        if ( event != nullptr && !event->IsDestructed() ) {
            delete event;
            event = nullptr;
        }
    }
    for ( TCluster* cluster : mClusters ) {
        if ( cluster != nullptr && !cluster->IsDestructed() ) {
            delete cluster;
            cluster = nullptr;
        }
    }
    // for ( TALPIDEEvent* maskedEvent : mMaskedEvents ) {
    //  if ( maskedEvent != nullptr && !maskedEvent->IsDestructed() ) {
    //      delete maskedEvent;
    //      maskedEvent = nullptr;
    //  }
    // }
    // for ( TCluster* maskedCluster : mMaskedClusters ) {
    //  if ( maskedCluster != nullptr && !maskedCluster->IsDestructed() ) {
    //      delete maskedCluster;
    //      maskedCluster = nullptr;
    //  }
    // }
    // for ( TALPIDEEvent* noiseEvent : mNoiseEvents ) {
    //  if ( noiseEvent != nullptr && !noiseEvent->IsDestructed() ) {
    //      delete noiseEvent;
    //      noiseEvent = nullptr;
    //  }
    // }
    // for ( TCluster* noiseCluster : mNoiseClusters ) {
    //  if ( noiseCluster != nullptr && !noiseCluster->IsDestructed() ) {
    //      delete noiseCluster;
    //      noiseCluster = nullptr;
    //  }
    // }
}


void TExperimentData::setEvents(const std::vector<TALPIDEEvent*>& events) {
    mEvents = events;
}

void TExperimentData::setClusters(const std::vector<TCluster*>& clusters) {
    mClusters = clusters;
}

// void TExperimentData::setMaskedEvents(const std::vector<TALPIDEEvent*>&& maskedEvents) {
//  mMaskedEvents = maskedEvents;
// }

// void TExperimentData::setMaskedClusters(const std::vector<TCluster*>&& maskedClusters) {
//  mMaskedClusters = maskedClusters;
// }

// void TExperimentData::setNoiseEvents(const std::vector<TALPIDEEvent*>&& noiseEvents) {
//  mNoiseEvents = noiseEvents;
```

```
// }

// void TExperimentData::setNoiseClusters(const std::vector<TCluster*>&& noiseClusters) {
//  mNoiseClusters = noiseClusters;
// }

const std::vector<TALPIDEEvent*> TExperimentData::getEvents() const {
    return mEvents;
}

const std::vector<TCluster*> TExperimentData::getClusters() const {
    return mClusters;
}

// const std::vector<TALPIDEEvent*> TExperimentData::getMaskedEvents() const {
//  return mMaskedEvents;
// }

// const std::vector<TCluster*> TExperimentData::getMaskedClusters() const {
//  return mMaskedClusters;
// }

// const std::vector<TALPIDEEvent*> TExperimentData::getNoiseEvents() const {
//  return mNoiseEvents;
// }

// const std::vector<TCluster*> TExperimentData::getNoiseClusters() const {
//  return mNoiseClusters;
// }
```

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.

```
#ifndef __TGRAPHCOMPARE__
#define __TGRAPHCOMPARE__

#ifdef __TGRAPHCOMPARE_HEADER__
#include <iostream>
#include<sstream>
#include "Rtypes.h"
#include "TFile.h"
#include "TCanvas.h"
#include "TH1D.h"
#include "TH2D.h"
#include "TString.h"
#include "TColor.h"
#include "TLegend.h"
#include "CppConfigFile.h"
#include "TGraphAsymmErrors.h"
#endif

#include <unordered_map>
#include <vector>
#include<filesystem>

class TFile;
class TH1D;
```

```
class Configurable;

class TGraphCompare {
private:
    std::unordered_map<std::string, TFile*> mGraphFileSet;
    std::vector<TH1D*> mClusterSizeSet;
    std::vector<std::string> mGraphInfoSet;
public:
    TGraphCompare(const std::vector<std::string>& graphFilePath);
    void TCompareClusterSize(std::string_view typeName, const CppConfigDictionary config);
    // void TCompareClusterSizeRatio(std::string_view typeName, const Configurable* config);
    // void TCompareClusterSize3D(std::string_view typeName, const Configurable* config);
    TH1D* getClustersizeHistogram(std::string_view pathInRoot, std::string operation);

};

#endif

// struct TGraphObjects {
//  std::string name;
//  std::string operation;
//  std::string legend;
//  double weighting;
//  bool operator < (const TGraphObjects& other) const { return true; }
//  bool operator==(const TGraphObjects& other) const { return name == other.name; }
//  TH1D* getClustersizeHistogram(std::string_view pathInRoot, const std::unordered_map<std::string, TFile*>& graphFileSet);
// };
```

8.37 /home/ychoi/ATOM/alpide/comparison/inc/TMerge.h File Reference

Tools for integrate same configue 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 configue but seperate file.

Author

Yongjun Choi (yochoi@cern.ch)

Version

0.1

Date

17-04-2024

Copyright

Definition in file TMerge.h.

8.38 TMerge.h

Go to the documentation of this file.

```
#ifndef __TMERGE__
#define __TMERGE__

#ifdef __TMERGE_HEADER__
#include "TFile.h"
#include "TTree.h"
#include "cpptqdm.h"
#endif

#include <vector>
#include <string>

#include "TFileFormat.h"

class TFile;

class MergeFileOpen : public std::exception {
public:
    std::string message;
public:
    MergeFileOpen(std::string_view fileName) {
        message = static_cast<std::string>(fileName) + " cannot open";
    }
    const char* what() const throw() {
        return message.c_str();
    }
};

class MergeTreeOpen : public std::exception {
public:
    std::string message;
public:
    MergeTreeOpen(std::string_view fileName) {
        message = "Tree of " + static_cast<std::string>(fileName) + " cannot be accessed";
    }
    const char* what() const throw() {
        return message.c_str();
    }
};

class TMerge {
protected:
    std::string mOutputFileName;
    std::vector<std::string> mInputFileNames;
public:
    TMerge(std::string_view outputFileName, const std::vector<std::string>& inputFileNames);
};

class TMergeExperimentROOT : public TMerge {
private:
    TFile* mOutputFile;
    TInputRoot mInputValue;
    TInputRoot mOutputValue;
public:
    TMergeExperimentROOT(std::string_view outputFileName, const std::vector<std::string>& inputFileNames);
    void mergeFile();
    ~TMergeExperimentROOT();
};

#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*)
      mGraphFileSet.find(stringSet[0])->second->Get(static_cast<TString>(pathInRoot));
00021     TH1D* newHist = new TH1D(*hist1);
00022     if ( stringSet.size() == 3 ) {
00023         TH1D* hist2 = (TH1D*)
      mGraphFileSet.find(stringSet[2])->second->Get(static_cast<TString>(pathInRoot));
00024         if ( stringSet[1] == "+" ) {
```

```
            newHist->Add(hist2);
            for ( int i = 0; i < 80; i++ ) {
                newHist->SetBinError(i, sqrt(hist1->GetBinContent(i) + hist2->GetBinContent(i)));
            }
        } else if ( stringSet[1] == "-" ) {
            newHist->Add(hist2, -1);
            for ( int i = 0; i < 80; i++ ) {
                newHist->SetBinError(i, sqrt(hist1->GetBinContent(i) + hist2->GetBinContent(i)));
            }
        } else if ( stringSet[1] == "*" ) {
            newHist->Multiply(hist2);
            for ( int i = 0; i < 80; i++ ) {
                newHist->SetBinError(i, sqrt(hist1->GetBinContent(i) * hist2->GetBinContent(i) * (hist1->GetBinContent(i) + hist2->GetBinContent(i))));
            }
        } else if ( stringSet[1] == "/" ) {
            newHist->Divide(hist2);
            for ( int i = 0; i < 80; i++ ) {
                newHist->SetBinError(i, sqrt(pow(hist1->GetBinContent(i) / hist2->GetBinContent(i), 4) + 1));
            }
        }
    }
    return newHist;
}

int getColor(const std::string colorName) {
    if ( colorName == "kRed" ) {
        return kRed;
    } else if ( colorName == "kMagenta" ) {
        return kMagenta;
    } else if ( colorName == "kBlue" ) {
        return kBlue;
    } else if ( colorName == "kCyan" ) {
        return kCyan + 2;
    } else if ( colorName == "kOrange" ) {
        return kOrange;
    } else {
        return kWhite;
    }
}

int getStyle(const int index) {
    if ( index == 0 ) {
        return 1;
    } else if ( index == 1 ) {
        return 8;
    } else if ( index == 2 ) {
        return 10;
    } else if ( index == 3 ) {
        return 6;
    } else if ( index == 4 ) {
        return 9;
    } else {
        return 0;
    }
}

TGraphCompare::TGraphCompare(const std::vector<std::string>& graphFilePath) {
    for ( std::string_view filePath : graphFilePath ) {
        std::filesystem::path path(filePath);
        TFile* file = new TFile(static_cast<TString>(path));
        mGraphFileSet.insert_or_assign(std::string(path.stem()), file);
    }
}

void TGraphCompare::TCompareClusterSize(std::string_view typeName, const CppConfigDictionary config) {
    TString canvasName = "can_" + config.find("name");
    Int_t canvasWidth = stoi(config.getSubConfig("canvas").find("width"));
    Int_t canvasHeight = stoi(config.getSubConfig("canvas").find("height"));
    TCanvas* canvas = new TCanvas(canvasName, "", canvasWidth, canvasHeight);

    TString legendTitle = config.getSubConfig("legend").hasKey("title") ? config.getSubConfig("legend").find("title") : "";
    Float_t legendXmin = config.getSubConfig("legend").hasKey("x_min") ? stof(config.getSubConfig("legend").find("x_min")) : 0.7;
    Float_t legendXmax = config.getSubConfig("legend").hasKey("x_max") ? stof(config.getSubConfig("legend").find("x_max")) : 0.7;
    Float_t legendYmin = config.getSubConfig("legend").hasKey("y_min") ? stof(config.getSubConfig("legend").find("y_min")) : 0.9;
    Float_t legendYmax = config.getSubConfig("legend").hasKey("y_max") ? stof(config.getSubConfig("legend").find("y_max")) : 0.9;
    Float_t legendDivide = config.getSubConfig("legend").hasKey("divide") ? stoi(config.getSubConfig("legend").find("divide")) : 1;
    TLegend* legend = new TLegend(legendXmin, legendYmin, legendXmax, legendYmax);
    legend->SetHeader(legendTitle, "c");
    legend->SetNColumns(legendDivide);
```

```
    std::map<std::string, CppConfigDictionary> plotConfigList;
    for ( const auto& pair : config.getSubConfig("plots").getSubConfigSetWithName() ) {
        plotConfigList.insert_or_assign(pair.first, pair.second);
    }
    std::map<std::string, TH1D*> distributionSet;

    std::string rootPath = config.find("root_path");
    for ( const auto& plotConfig : plotConfigList ) {
        distributionSet.insert_or_assign(plotConfig.first, getClustersizeHistogram(rootPath, plotConfig.second.find("histogram")));
    }

    bool isFirst;
    for ( const auto& distribution : distributionSet ) {
        if ( isFirst ) {

            isFirst = false;
        }

        double nEntry = 0;
        int csMin = stoi(config.find("cluster_size_of_interest_min"));
        int csMax = stoi(config.find("cluster_size_of_interest_max"));

        for ( int clusterSize = 1; clusterSize < 81; clusterSize++ ) {
            if ( clusterSize < csMin || clusterSize > csMax ) {
                distribution.second->SetBinContent(clusterSize, 0);
            }
        }

        if ( config.hasKey("normalization") && (config.find("normalization") == "true") ) {
            distribution.second->Scale(1. / nEntry);
        } else if ( plotConfigList.find(distribution.first)->second.hasKey("scale") ) {
            Float_t scaleFactor = stof(plotConfigList.find(distribution.first)->second.find("scale"));
            distribution.second->Scale(scaleFactor);
        }


        for ( int clusterSize = csMin; clusterSize < csMax + 1; clusterSize++ ) {
            nEntry += distribution.second->GetBinContent(clusterSize);
        }
        TString legendTitle = plotConfigList.find(distribution.first)->second.find("legend_title");
        TString strEntry = Form("%.2f", round(nEntry * 100) / 100.);
        if ( plotConfigList.find(distribution.first)->second.hasKey("scale_error_max") ) {
            TString strEntryPlus = Form("%.2f", abs(round(nEntry * (stof(plotConfigList.find(distribution.first)->second.find("scale_error_max")) - 1) * 100) / 100.));
            TString strEntryMinus = Form("%.2f", abs(round(nEntry * (1 - stof(plotConfigList.find(distribution.first)->second.find("scale_error_min"))) * 100) / 100.));
            legend->AddEntry(distribution.second, legendTitle + "(" + strEntry + "+" + strEntryPlus + "-" + strEntryMinus + " in " + std::to_string(csMin) + " ~ " + std::to_string(csMax) + ")", "l");
        } else {
            Float_t scaleFactor = stof(plotConfigList.find(distribution.first)->second.find("scale"));
            TString strError = Form("%.2f", sqrt(nEntry * scaleFactor));
            legend->AddEntry(distribution.second, legendTitle + "(" + strEntry + char(0x00B1) + strError + " in " + std::to_string(csMin) + " ~ " + std::to_string(csMax) + ")", "l");
        }

        TString title = config.find("title");
        TString xTitle = config.find("x_title");
        TString yTitle = config.find("y_title");

        distribution.second->SetMinimum(stof(config.find("y_min")));
        distribution.second->SetMaximum(stof(config.find("y_max")));
        distribution.second->GetXaxis()->SetRangeUser(csMin - .5, csMax + .5);
        distribution.second->SetTitle(title + "; " + xTitle + "; " + yTitle);
distribution.second->SetLineColor(getColor(plotConfigList.find(distribution.first)->second.find("line_color")));

        Float_t lineWidth = config.hasKey("line_width") ? stof(config.find("line_width")) : 1.;
        distribution.second->SetLineWidth(lineWidth);
        Float_t lineStyle = plotConfigList.find(distribution.first)->second.hasKey("line_style") ? stof(plotConfigList.find(distribution.first)->second.find("line_style")) : 1.;
        distribution.second->SetLineStyle(lineStyle);

        distribution.second->SetStats(0);
        distribution.second->Draw("SAME HISTE");
        if ( plotConfigList.find(distribution.first)->second.hasKey("scale_error_max") ) {
            TGraphAsymmErrors* systemGraph = new TGraphAsymmErrors();
            for ( int clusterSize = csMin; clusterSize < csMax + 1; clusterSize++ ) {
                systemGraph->SetPoint(clusterSize, clusterSize, distribution.second->GetBinContent(clusterSize));
                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));
            }
            Float_t alpha = stof(config.find("error_box_alpha"));
```

```
            systemGraph->SetFillColorAlpha(getColor(plotConfigList.find(distribution.first)->second.find("line_color")), alpha);
            systemGraph->Draw("SAME E2");
        }
    }

    legend->Draw();
    if ( config.hasKey("logy") && (config.find("logy") == "true") ) {
        canvas->SetLogy();
    }
    std::filesystem::path outputPath(config.find("output_path"));
    std::filesystem::create_directories(outputPath);
    outputPath /= config.find("name");
    outputPath.replace_extension(config.find("extension"));

    canvas->SaveAs(static_cast<TString>(outputPath));
}
```

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) : mOutputFileName(outputFileName) {
00005     mInputFileNames.assign(inputFileNames.begin(), inputFileNames.end());
00006 }
00007
00008 TMergeExperimentROOT::TMergeExperimentROOT(std::string_view outputFileName, const std::vector<std::string>& inputFileNames) : TMerge(outputFileName, inputFileNames) {
00009     mOutputFile = new TFile(static_cast<TString>(outputFileName), "RECREATE");
00010 }
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;
```

```
        for ( std::string_view inputFileName : mInputFileNames ) {
            TFile* inputFile = new TFile(static_cast<TString>(inputFileName), "READ");
            if ( !inputFile->IsOpen() ) {
                throw MergeFileOpen(inputFileName);
            } else if ( inputFile->Get("hit") == nullptr ) {
                throw MergeTreeOpen(inputFileName);
            }
            std::clog « inputFileName « std::endl;
            TTree* inputTree = static_cast<TTree*>(inputFile->Get("hit"));
            inputTree->SetBranchAddress("ChipID", &mInputValue.chipid);
            inputTree->SetBranchAddress("TimeStamp", &mInputValue.timeStamp);
            inputTree->SetBranchAddress("X", &mInputValue.x);
            inputTree->SetBranchAddress("Y", &mInputValue.y);
            ProgressBar* pbar = new ProgressBar(inputTree->GetEntries());
            for ( int entry = 0; entry < inputTree->GetEntries(); entry++ ) {
                pbar->printProgress();
                inputTree->GetEntry(entry);
                mOutputValue.chipid = mInputValue.chipid;
                mOutputValue.timeStamp = mInputValue.timeStamp + preTime;
                mOutputValue.x = mInputValue.x;
                mOutputValue.y = mInputValue.y;
                outputTree->Fill();
            }
            delete pbar;
            preTime = mOutputValue.timeStamp + 1;
            inputFile->Close();
        }
        mOutputFile->Write();
    } catch ( MergeFileOpen error ) {
        std::cerr « error.what() « std::endl;
        exit(1);
    } catch ( MergeTreeOpen error ) {
        std::cerr « error.what() « std::endl;
        exit(1);
    }
}

TMergeExperimentROOT::~TMergeExperimentROOT() {
    mOutputFile->Close();
}
```

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.

```
#ifndef __TALPIDEDECODER__
#define __TALPIDEDECODER__

#ifdef __TALPIDEDECODER_HEADER__
#include<array>

#include "TALPIDEEvent.h"
#endif

#include<vector>
#include<filesystem>
```

```
#include<cmath>

#include "TDecoder.h"

class TALPIDEEvent;

class TALPIDEDecoder : public TDecoder {
private:
    std::vector<std::unique_ptr<TALPIDEEvent» alpides;
    int index_ = 0;

public:
    TALPIDEDecoder(const std::filesystem::path& binaryPath);
    TALPIDEDecoder(const std::string& binaryPath);
    void decode();
    std::vector<std::unique_ptr<TALPIDEEvent» getData();
private:
    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);
};
#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.

```
#ifndef __TALPIDEEVENT__
#define __TALPIDEEVENT__

#ifdef __TALPIDEEVENT_HEADER__
#include <unordered_set>
#endif

#include <utility>
#include <vector>
#include<algorithm>

#include "TEvent.h"

class TALPIDEEvent : public TEvent {
private:
    long int iTime;
    std::vector<std::pair<int, int» data;

public:
    TALPIDEEvent();
    TALPIDEEvent(const TALPIDEEvent& copy);
    TALPIDEEvent& operator=(const TALPIDEEvent& copy);
    TALPIDEEvent(TALPIDEEvent&& move);
    TALPIDEEvent& operator=(TALPIDEEvent&& move);

```

```
    // Setter
    void setTime(const long int time);

    // Getter
    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;

private:
    unsigned int fBits;
public:
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
};

#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.

```
#define __TALPIDEDECODER_HEADER__
#include "TALPIDEDecoder.h"

const std::array<uint8_t, 16> stray_sequence{0xaa, 0xaa, 0xaa, 0xaa, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

TALPIDEDecoder::TALPIDEDecoder(const std::filesystem::path& binaryPath) : TDecoder(binaryPath) { }
TALPIDEDecoder::TALPIDEDecoder(const std::string& binaryPath) : TDecoder(binaryPath) { }

void TALPIDEDecoder::decode() {
    preTest();
    while ( !isDone() ) {
        inputEvent();
    }
}

std::vector<std::unique_ptr<TALPIDEEvent» TALPIDEDecoder::getData() {
    return std::move(alpides);
}

void TALPIDEDecoder::inputEvent() {
    int numEvent = hex_to_dec(&*getBinaryData().begin() + index_ + 4, 4);
    long int time = hex_to_dec(&*getBinaryData().begin() + index_ + 8, 8);
    std::unique_ptr<TALPIDEEvent> alpide(new TALPIDEEvent());
    alpide->setEvent(numEvent);
    alpide->setTime(time);
    index_ += 4 * 4;

    if ( bitwise_and(getBinaryData()[index_], 0xF0) == 0xE0 ) { // chip has empty frame
        if ( (getBinaryData()[index_ + 2] != 0xFF) || getBinaryData()[index_ + 3] != 0xFF ) {
            std::cerr « "Assertion Error at empty frame test" « std::endl;
            throw std::bad_exception();
        }
        index_ += 4;
    } else if ( bitwise_and(getBinaryData()[index_], 0xF0) == 0xA0 ) {
        index_ += 2;
        int n = getBinaryData().size();
        int reg = 0;
        while ( index_ < n ) {
            uint8_t data0 = getBinaryData()[index_];
            if ( bitwise_and(data0, 0xC0) == 0x00 ) {  // data  long
                int d = bitwise_or(bitwise_or((reg * pow(2, 14)), bitwise_and(data0, 0x3F) * pow(2, 8)), getBinaryData()[index_ + 1]);
                int x = bitwise_or(bitwise_and((int) (d / pow(2, 9)), 0x3FE), bitwise_and(bitwise_xor(d, (int) (d / pow(2, 1))), 0x1));
                int y = bitwise_and((int) d / pow(2, 1), 0x1FF);
                alpide->pushData({x, y});
                uint8_t data2 = getBinaryData()[index_ + 2];
                d += 1;
                while ( data2 ) {
                    if ( data2 & 1 ) {
                        int x = bitwise_or(bitwise_and((int) (d / pow(2, 9)), 0x3FE), bitwise_and(bitwise_xor(d, (int) (d / pow(2, 1))), 0x1));
                        int y = bitwise_and((int) d / pow(2, 1), 0x1FF);
                        alpide->pushData({x, y});
                    }
                    data2 = (int) (data2 / pow(2, 1));
                    d += 1;
                }
                index_ += 3;
            } else if ( bitwise_and(data0, 0xC0) == 0x40 ) {
                int d = bitwise_or(bitwise_or((reg * pow(2, 14)), bitwise_and(data0, 0x3F) * pow(2, 8)), getBinaryData()[index_ + 1]);
                int x = bitwise_or(bitwise_and((int) (d / pow(2, 9)), 0x3FE), bitwise_and(bitwise_xor(d, (int) (d / pow(2, 1))), 0x1));
                int y = bitwise_and((int) d / pow(2, 1), 0x1FF);
                alpide->pushData({x, y});
                index_ += 2;
            } else if ( bitwise_and(data0, 0xE0) == 0xC0 ) {
                reg = bitwise_and(data0, 0x1F);
                index_ += 1;
            } else if ( bitwise_and(data0, 0xF0) == 0xB0 ) {
                index_ += 1;
                index_ = static_cast<int> ((index_ + 3) / 4 * 4);
                break;
            } else if ( data0 == 0xFF ) {
                index_ += 1;
            } else {
                std::cout « "DEBUG: data[i-10,i+10]:";
                for ( int j = 0; j < 20; j++ ) {
                    std::cout « getBinaryData()[index_ + j - 10];
                }
```

```
                std::cout « "ValueError: invalid literal" « std::endl;
                throw std::bad_exception();
            }
        }
    }
    index_ += 4;
    alpides.push_back(std::move(alpide));
}



void TALPIDEDecoder::preTest() {
    if ( index_ == 0 && strayTest() ) {
        index_ += 16;
    }
    headerTest();
}

bool TALPIDEDecoder::strayTest() {
    if ( std::equal(getBinaryData().begin(), getBinaryData().begin() + 16, stray_sequence.begin()) ) {
        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.";
        return true;
    } else {
        return false;
    }
}

void TALPIDEDecoder::headerTest() {
    if ( !std::equal(getBinaryData().begin(), getBinaryData().begin() + 4, stray_sequence.begin()) ) {
        std::cerr « "Assertion Error in header, not 0x AA AA AA AA" « std::endl;
        throw std::bad_exception();
    }
}

long int TALPIDEDecoder::hex_to_dec(const uint8_t* data, const int& digits) {
    long int val = 0;
    for ( int digit = 0; digit < digits; digit++ ) {
        val += data[digit] * pow(256, digit);
    }
    return val;
}

int TALPIDEDecoder::bitwise_and(int v1, int v2) {
    int result = 0;
    uint8_t digit = 0;
    while ( true ) {
        result += (v1 % 2 * v2 % 2) * pow(2, digit);
        v1 = v1 / 2;
        v2 = v2 / 2;
        digit++;
        if ( v1 == 0 || v2 == 0 ) break;
    }
    return result;
}

int TALPIDEDecoder::bitwise_or(int v1, int v2) {
    int result = 0;
    uint8_t digit = 0;
    while ( true ) {
        result -= ((v1 % 2 - 1) * (v2 % 2 - 1) - 1) * pow(2, digit);
        v1 = v1 / 2;
        v2 = v2 / 2;
        digit++;
        if ( v1 == 0 && v2 == 0 ) break;
    }
    return result;
}

int TALPIDEDecoder::bitwise_xor(int v1, int v2) {
    int result = 0;
    uint8_t digit = 0;
    while ( true ) {
        result += (v1 % 2 + v2 % 2) % 2 * pow(2, digit);
        v1 = v1 / 2;
        v2 = v2 / 2;
        digit++;
        if ( v1 == 0 && v2 == 0 ) break;
    }
    return result;
}

bool TALPIDEDecoder::isDone() {
    if ( index_ >= getDataLength() ) return true;
    else return false;
}
```

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));
```

```
}

const long int TALPIDEEvent::getTime() const {
    return iTime;
}

const std::vector<std::pair<int, int»& TALPIDEEvent::getData() const {
    return data;
}

void TALPIDEEvent::removePixel(const std::pair<int, int>& coordinate) {
    data.erase(std::find(data.begin(), data.end(), coordinate));
}

void TALPIDEEvent::removeDuplication() {
    std::vector<std::pair<int, int» uniqueData;

    auto arrayHash = [ ](const std::pair<int, int>& arr) {
        return std::hash<int>()(arr.first) ^ (std::hash<int>()(arr.second) « 1);
        };

    std::unordered_set<std::pair<int, int>, decltype(arrayHash)> uniqueSet(data.size() * 2, arrayHash);

    for ( const auto& arr : data ) {
        if ( uniqueSet.find(arr) == uniqueSet.end() ) {
            uniqueData.push_back(arr);
            uniqueSet.insert(arr);
        }
    }
    data = std::move(uniqueData);
}

void TALPIDEEvent::sortPixel() {
    std::sort(data.begin(), data.end(), [ ](const std::pair<int, int>& a, const std::pair<int, int>& b) {
        if ( a.second != b.second ) {
            return a.second < b.second;  // Sort by first element in ascending order
        }
        return a.first < b.first;  // If the first elements are equal, sort by second element in ascending order
            });
}

const int TALPIDEEvent::getNData() const {
    return data.size();
}
```

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 "TF1.h"
00009 #endif
00010
00011 #include<array>
00012 #include<algorithm>
00013
00014 class TGraph;
00015 class TF1;
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<TF1> 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>& coodrdinate, const std::array<int, 50>& dacs);
00044     TThreshold(int x, int y, std::array<int, 50>&& dacs);
00045     TThreshold(const std::array<int, 2>& coodrdinate, 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;
```

```
    const double getQualityFactor() const;

    const ThrCondition getCondition() const;
};

#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.

```
#ifndef __TTHRESHOLDCOMPARE__
#define __TTHRESHOLDCOMPARE__

#ifdef __TTHRESHOLDCOMPARE_HEADER__
#endif

#include <vector>
#include<memory>

class TThreshold;

// #include "Headers.h"
// #include "TThreshold.h"

class TThresholdCompare {
private:
    std::vector<std::unique_ptr<TThreshold*» thresholds_;
public:
    TThresholdCompare();
    TThresholdCompare(std::vector<std::unique_ptr<TThreshold*» thresholds);


};

#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) :
      mX(coordinate[0]), mY(coordinate[1]) {
00016     std::copy(std::begin(dacs), std::end(dacs), std::begin(mDacs));
00017 }
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) :
      mX(coordinate[0]), mY(coordinate[1]) {
00024     std::move(std::begin(dacs), std::end(dacs), std::begin(mDacs));
00025 }
00026
00027 TThreshold::TThreshold(const TThreshold& copy) : mX(copy.mX), mY(copy.mY), mThr(copy.mThr),
      mErr(copy.mErr) {
00028     std::copy(std::begin(copy.mDacs), std::end(copy.mDacs), std::begin(mDacs));
00029 }
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     std::move(std::begin(move.mDacs), std::end(move.mDacs), std::begin(mDacs));
00042 }
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;
```

```
        mErr = 100;
        return ThrCondition::bad_too_high;
    } else {
        std::array<int, 50> adcs;
        std::iota(std::begin(adcs), std::end(adcs), 1);
        thresholdGraph.reset(new TGraph(50, std::begin(adcs), std::begin(mDacs)));
        fitFunction.reset(new TF1("fitFunction", "[0]*TMath::Erf((x-[1])/[2])+[3]", 0., 50.));
        bool quality = false;
        int count = 0;
        while ( !quality ) {
            fitFunction->SetParameters(20, 10 * count + 10, 10, 25);
            thresholdGraph->Fit("fitFunction", "q");
            mThr = fitFunction->GetParameter(1);
            mErr = fitFunction->GetParameter(2);
            mQualityFactor = fitFunction->GetChisquare() / fitFunction->GetNDF();
            if ( mQualityFactor < 100. ) {
                quality = true;
            }
            if ( count > 4 ) {
                mThr = -1;
                mErr = -1;
                return ThrCondition::bad_undefine;
            }
            count++;
        }
        if ( mThr > 0 && mThr < 50 ) {
            return ThrCondition::good;
        } else {
            return ThrCondition::bad_undefine;
        }
    }
}

void TThreshold::savePlot() {
    std::unique_ptr<TCanvas> can(new TCanvas("can", "can", 500, 500));
    thresholdGraph->SetTitle(static_cast<TString>("Threshold Graph at " + std::to_string(mX) + ", " + std::to_string(mY) + "; ADC[$500 \times e^-$]; DAC[# of Fire]"));
    thresholdGraph->Draw();
    // can->SaveAs(static_cast<TString>("data/" + std::to_string(mX) + "_" + std::to_string(mY) + ".png"));
}

const double TThreshold::getX() const {
    return mX;
}

const double TThreshold::getY() const {
    return mY;
}

const double TThreshold::getThreshold() const {
    return mThr;
}

const double TThreshold::getError() const {
    return mErr;
}

const double TThreshold::getQualityFactor() const {
    return mQualityFactor;
}

const ThrCondition TThreshold::getCondition() const {
    return mCondition;
}
```

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.

```
#ifndef __TAPTSDECODER__
#define __TAPTSDECODER__

#ifdef __TAPTSDECODER_HEADER__
#include "TAPTSEvent.h"
#endif

#include<vector>
#include<array>
#include<string>
#include<cmath>

#include "TDecoder.h"

class TAPTSEvent;

class TAPTSDecoder : public TDecoder {
private:
    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;

public:
    TAPTSDecoder(const std::filesystem::path& binaryPath);
    TAPTSDecoder(const std::string& binaryPath);
    void decode();
    std::vector<TAPTSEvent*> getData();
private:
    void inputEvent();
    void preTest(); // do range, header, missing test and calculate data length of Event.
    void rangeTest(); // Test whether the index number is overflowed or not.
    void headerTest(); // Test whether an event is normal or not. The first 4 num should be 0xAA.
    void missingTest(); // Test whether the data is overflowed or not.
    int getEventLength(); // Calculate the data length of an event.
    void postTest();
    bool isDone();
};
#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.

```
#ifndef __TAPTSEVENT__
#define __TAPTSEVENT__

#include "TEvent.h"

#include <array>

class TAPTSEvent : public TEvent {
private:
    int iFrame;
    std::array<int, 16> data{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

public:
    TAPTSEvent();

    // Setter
    void setFrame(int frame);

    // Getter
    int getFrame();
    std::array<int, 16>& getData();
};

#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),
      mapping.begin())
00012         : std::copy(std::begin(APTS_PIXEL_ADC_MAPPING), std::end(APTS_PIXEL_ADC_MAPPING),
      mapping.begin());
00013     // Set mapping array.
00014
00015     while ( !isDone() ) {
00016         inputEvent();
00017     }
00018 }
```

```
std::vector<TAPTSEvent*> TAPTSDecoder::getData() {
    return aptss;
}

void TAPTSDecoder::inputEvent() {
    preTest();

    std::vector<uint8_t> eventData(getBinaryData().begin() + index_, getBinaryData().begin() + index_ + lenEvent_);
    // A part of binary data.

    nFrame_ = lenEvent_ / 40;
    // Set number of frame.

    for ( int iFrame = 0; iFrame < nFrame_; iFrame++ ) {
        TAPTSEvent* apts = new TAPTSEvent();
        apts->setEvent(iEvent_);
        for ( int j = 0; j < 16; j++ ) {
            int share1 = eventData[iFrame * 40 + 2 * j];
            int share2 = eventData[iFrame * 40 + 2 * j + 1];
            int rest1 = 0;
            int rest2 = 0;
            for ( int digit = 0; digit < 8; digit++ ) {
                rest1 = share1 % 2;
                rest2 = share2 % 2;
                apts->getData()[mapping[digit]] += rest1 * pow(2, 15 - j);
                apts->getData()[mapping[digit + 8]] += rest2 * pow(2, 15 - j);
                share1 = ceil(share1 / 2);
                share2 = ceil(share2 / 2);
            }
        }
        apts->setFrame(iFrame);

        int trailer = eventData[iFrame * 40 + 39] + eventData[iFrame * 40 + 38] * 256;
        if ( iFrame < nFrame_ - 1 ) {
            if ( trailer != 0xFEFE ) std::cout « "Unexpected frame trailer 0x" « std::hex « trailer « " for frame " « iFrame + 1 « " out of " « std::dec « nFrame_ « std::endl;
        } else {
            if ( trailer != 0xAEAE ) std::cout « "Unexpected event trailer 0x" « std::hex « trailer « " for frame " « iFrame + 1 « " out of " « std::dec « nFrame_ « std::endl;
        }
        aptss.push_back(apts);
    }
    index_ += lenEvent_;
    postTest();
    iEvent_++;
}

void TAPTSDecoder::postTest() {
    if ( index_ + 12 <= getDataLength() && std::equal(getBinaryData().begin() + index_, getBinaryData().begin() + index_ + 3, expected_footer.begin()) ) {
        index_ += 16;
    }
}

void TAPTSDecoder::preTest() {
    rangeTest();
    headerTest();
    lenEvent_ = getEventLength();
    index_ += 8;
    missingTest();
}

void TAPTSDecoder::rangeTest() {
    if ( !(index_ + 8 <= getDataLength()) ) {
        std::cerr « "Ev " « iEvent_ « ": Unexpected data length(" « getDataLength() - 1 « " < 8)" « std::endl;
    }
}

void TAPTSDecoder::headerTest() {
    if ( !std::equal(getBinaryData().begin() + index_, getBinaryData().begin() + index_ + 3, expected_header.begin()) ) {
        std::cout « "Ev " « iEvent_ « std::hex « int(getBinaryData()[index_]) « "-" « int(getBinaryData()[index_ + 1]) « "-" « int(getBinaryData()[index_ + 2]) « "-" « int(getBinaryData()[index_ + 3]) « std::endl;
    }
}

void TAPTSDecoder::missingTest() {
    if ( !(index_ + lenEvent_ <= getDataLength()) ) {
        std::cerr « "Ev " « iEvent_ « ":Missing data (" « index_ « " " « lenEvent_ « " " « getDataLength() « ")" « std::endl;
    }
}
```

```
int TAPTSDecoder::getEventLength() {
    int lenEvent_ = 0;
    for ( int i = 0; i < 4; i++ ) {
        lenEvent_ += getBinaryData()[index_ + 4 + i] * pow(256, i);
    }
    return lenEvent_;
}

bool TAPTSDecoder::isDone() {
    if ( index_ >= getDataLength() ) return true;
    else return false;
}
```

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.

```
#include "TAPTSEvent.h"

TAPTSEvent::TAPTSEvent() : TEvent() { }

void TAPTSEvent::setFrame(int frame) {
    iFrame = frame;
}

int TAPTSEvent::getFrame() {
    return iFrame;
}

std::array<int, 16>& TAPTSEvent::getData() {
    return data;
}
```

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.

```
#ifndef __TDECODER__
#define __TDECODER__

#ifdef __TDECODER__
#include<iostream>
#endif

#include<fstream>
#include<vector>
#include<filesystem>
#include<string>

class TDecoder {
private:
    std::ifstream binaryFile_;
    int dataLength_;
    std::vector<uint8_t> binaryData_;
public:
    //Constructor
    TDecoder(const std::filesystem::path& binaryPath);
    TDecoder(const std::string& binaryPath);

    void readFile();

    int getDataLength();
    std::vector<uint8_t>& getBinaryData();
};

#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.

```
#ifndef __TEVENT__
#define __TEVENT__

class TEvent {
private:
    int iEvent;

public:
    TEvent();
    virtual ~TEvent();
    void setEvent(const int event);
    const int getEvent() const;
};

#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.

```
#include "TDecoder.h"

TDecoder::TDecoder(const std::filesystem::path& path) {
    if ( !std::filesystem::exists(path) ) {
        std::cerr « "File " « path « " cannot be found." « std::endl;
    }
    binaryFile_ = std::ifstream(path, std::ios::binary);
}

TDecoder::TDecoder(const std::string& path) {
    if ( !std::filesystem::exists(path) ) {
        std::cerr « "File " « path « " cannot be found." « std::endl;
    }
    binaryFile_ = std::ifstream(path, std::ios::binary);
}

void TDecoder::readFile() {
    char buf[sizeof(uint8_t)];
    while ( binaryFile_.read(buf, sizeof(buf)) ) {
        binaryData_.push_back(static_cast<uint8_t>(*buf));
    }
    dataLength_ = binaryData_.size();
}

int TDecoder::getDataLength() {
    return dataLength_;
}

std::vector<uint8_t>& TDecoder::getBinaryData() {
    return binaryData_;
}
```

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.

```
#include "TEvent.h"

TEvent::TEvent() { }

TEvent::~TEvent() { }

void TEvent::setEvent(const int event) {
    iEvent = event;
}

const int TEvent::getEvent() const {
    return iEvent;
}
```

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.

```
#ifndef __TGRAPHUSER__
#define __TGRAPHUSER__

#include "CppConfigFile.h"
#include "TCanvas.h"
#include "TGraph.h"
#include "TGraphErrors.h"
#include "TLegend.h"
#include "TAxis.h"
#include "TMultiGraph.h"
#include<filesystem>

class TGraphUser : public TGraph {
private:
    TCanvas* mCanvas; // Canvas for graph
    TLegend* mLegend; // Legend
    TMultiGraph* mMultiGraph; // Multi-graph
private:
    TString savePath; // Save path and file name of graph
    std::string mFileName;

    TString title, x_title, y_title; // Graph title, x-axis and y-axis titles
    std::vector<std::tuple<TGraph*, Style_t, Size_t, Style_t, Size_t» graphSetAndAttribute; // Graph, marker style, marker size, line style, line size

    std::array<Float_t, 4> canvasMargins; // Left, Right, Bottom, Top
    std::array<Float_t, 4> canvasOffsets;

    Style_t mUniversialMarkerStyle;
    Size_t mUniversialMarkerSize;
    Style_t mUniversialLineStyle;
    Size_t mUniversiallineWidth;

    TString mLegendTitle; // Title of legend
    std::array<Float_t, 4> mLegendPoints; // The cornor points of legend
public:
    TGraphUser(const CppConfigDictionary& config);
    void AddGraph(TGraph* graph);
    void AddGraph(TGraph* graph, const CppConfigDictionary& graphConfig);
    void Save(const std::filesystem::path& outputPath = ".");
private:
    void setCanvas(const CppConfigDictionary& config);
    void setLegend(const CppConfigDictionary& config);

};


#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")) :
      mUniversialLineStyle;
00042     Size_t lineWidth = graphConfig.hasKey("line_width") ? stof(graphConfig.find("line_width")) :
      mUniversiallineWidth;
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"))
      : mUniversialMarkerStyle;
00049     Size_t markerSize = graphConfig.hasKey("marker_size") ? stof(graphConfig.find("marker_size")) :
      mUniversialMarkerSize;
00050     Color_t markerColour = graphConfig.hasKey("marker_colour") ?
      colourSet.find(graphConfig.find("marker_colour"))->second : 0;
```

```
    graph->SetMarkerStyle(markerStyle);
    graph->SetMarkerSize(markerSize);
    graph->SetMarkerColor(markerColour);

    TString graphOption = graphConfig.hasKey("graph_option") ? graphConfig.find("graph_option") : "pl";
    mLegend->AddEntry(graph, legendLabel, graphOption);
    mMultiGraph->Add(graph, graphOption);
}


void TGraphUser::setCanvas(const CppConfigDictionary& config) {
    Int_t canvasRow = config.hasKey("canvas_row") ? stoi(config.find("canvas_row")) : 1000;
    Int_t canvasWidth = config.hasKey("canvas_width") ? stoi(config.find("canvas_width")) : 1000;

    mCanvas = new TCanvas();
    mCanvas->SetCanvasSize(canvasRow, canvasWidth);

    // Set Canvas Titles
    if ( config.hasKey("titles") ) {
        CppConfigDictionary titles = config.getSubConfig("titles");
        title = titles.hasKey("title") ? titles.find("title") : "";
        x_title = titles.hasKey("x_title") ? titles.find("x_title") : "";
        y_title = titles.hasKey("y_title") ? titles.find("y_title") : "";
    }

    // Set Canvas Margin
    if ( config.hasKey("canvas_margin") ) {
        CppConfigDictionary canvasMargin = config.getSubConfig("canvas_margin");
        Float_t marginLeft = canvasMargin.hasKey("left") ? stod(canvasMargin.find("left")) : .1;
        Float_t marginRight = canvasMargin.hasKey("right") ? stod(canvasMargin.find("right")) : .1;
        Float_t marginTop = canvasMargin.hasKey("top") ? stod(canvasMargin.find("top")) : .1;
        Float_t marginBottom = canvasMargin.hasKey("bottom") ? stod(canvasMargin.find("bottom")) : .1;
        canvasMargins = {marginLeft, marginRight, marginBottom, marginTop};
    } else {
        canvasMargins = {.1, .1, .1, .1};
    }

    // Set Canvas Offsets
    if ( config.hasKey("offsets") ) {
        CppConfigDictionary canvasOffset = config.getSubConfig("offsets");
        Float_t xLabelOffset = canvasOffset.hasKey("x_label") ? stod(canvasOffset.find("x_label")) : .01;
        Float_t yLabelOffset = canvasOffset.hasKey("y_label") ? stod(canvasOffset.find("y_label")) : .01;
        Float_t xTitleOffset = canvasOffset.hasKey("x_title") ? stod(canvasOffset.find("x_title")) : 1.;
        Float_t yTitleOffset = canvasOffset.hasKey("y_title") ? stod(canvasOffset.find("y_title")) : 1.;
        canvasOffsets = {xLabelOffset, yLabelOffset, xTitleOffset, yTitleOffset};
    } else {
        canvasOffsets = {.01, .01, 1., 1.};
    }

    mUniversialMarkerStyle = config.hasKey("marker_style") ? stoi(config.find("marker_style")) : 1;
    mUniversialMarkerSize = config.hasKey("marker_size") ? stoi(config.find("marker_size")) : 1;
    mUniversialLineStyle = config.hasKey("line_style") ? stoi(config.find("line_style")) : 1;
    mUniversiallineWidth = config.hasKey("line_width") ? stoi(config.find("line_width")) : 1;
}

void TGraphUser::setLegend(const CppConfigDictionary& config) {
    if ( config.hasKey("legend") ) {
        CppConfigDictionary legendConfig = config.getSubConfig("legend");
        mLegendTitle = legendConfig.hasKey("title") ? legendConfig.find("title") : "";
        Float_t xMin = legendConfig.hasKey("x_min") ? stof(legendConfig.find("x_min")) : 0.7;
        Float_t yMin = legendConfig.hasKey("y_min") ? stof(legendConfig.find("y_min")) : 0.7;
        Float_t xMax = legendConfig.hasKey("x_max") ? stof(legendConfig.find("x_max")) : 0.9;
        Float_t yMax = legendConfig.hasKey("y_max") ? stof(legendConfig.find("y_max")) : 0.9;
        mLegendPoints = {xMin, yMin, xMax, yMax};
    } else {
        mLegendTitle = "Legend";
        mLegendPoints = {0.7, 0.7, 0.9, 0.9};
    }
    mLegend = new TLegend(mLegendPoints[0], mLegendPoints[1], mLegendPoints[2], mLegendPoints[3]);
    mLegend->SetHeader(mLegendTitle, "C");
}
```

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 nbinx = 100;
00019
00020     TH1D* histogram = new TH1D("histo", "histo", nbinx, 0, 2.5);
00021
00022     for ( Int_t i = 0; i < nbinx + 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 / nbinx;;  // 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",
      "(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,
      R, 0, 2 * pi);
00041         func1->SetParameter(0, c_L);
00042         func1->SetParameter(1, c_R);
00043
```

```

        // func1->Draw("COL");
        // func1->Draw("Surf");
        // func1->Draw("same");
        //EPS = Epsilon
        //EPS default = 1.e-9

        Double_t integral = func1->Integral(0, R, 0, 2 * pi);
        // cout« integral «endl;


        Double_t EffSolAngle = integral * 4 * pi;
        cout « "Radius of collimator = " « c_R « endl;
        cout « "Effective Solid Angle = " « EffSolAngle « endl;
        // Double_t integralError = func1->IntegralError(0,2.5,0,2*pi);
        // cout« integralError «endl;
        //??

        Int_t binNumberFactor = nbinx / 2.5;
        histogram->SetBinContent(c_R * binNumberFactor, EffSolAngle);

    }



    histogram->Draw("");
    TF1* quadratic = new TF1("quadratic", "[0]*pow(x,2)+[1]", 0, 2.5);
    histogram->Fit(quadratic);
    histogram->Draw("SAME");
    // // histogram->Draw("ALP");
    // TF1 *exponential = new TF1("exponential","exp([0]*x+[1])",0,2.5);
    // histogram->Fit(exponential);
    // histogram->Draw("SAME");




    // // activity
    // Double_t activityDensity = s_A/(pi*pow(s_R,2));
    // // cout « "ActivityDensity" « endl;
    // // cout « activityDensity « endl;
    // Double_t TheoreticalValue=integral*activityDensity;
    // cout « "Effective Activity" «endl;
    // cout « TheoreticalValue « endl;

    // //10min Entry
    // Double_t entry_10min = TheoreticalValue*600;
    // cout« "10min Entry" «endl;
    // cout«entry_10min«endl;


    // // (rho-6)/rho
    // 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);
    // cout«func2->Integral(0,2.5,0,2*pi)«endl;
    // func2->Draw("Surf");

}
```

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) {}
```

```
    void setDAC(int dac) {
        dac_.push_back(dac);
    }
    void setADC(int adc) {
        adc_.push_back(adc);
    }
    void setDAC(std::vector<int> dac) {
        dac_.reserve(dac_.size() + dac.size());
        dac_.insert(dac_.end(), dac.begin(), dac.end());
    }
    void setADC(std::vector<int> adc) {
        adc_.reserve(adc_.size() + adc.size());
        adc_.insert(adc_.end(), adc.begin(), adc.end());
    }
    std::string getKind() {
        return kind_;
    }
    std::vector<Int_t> getDAC() {
        return dac_;
    }
    std::vector<Int_t> getADC() {
        return adc_;
    }
};

ArgumentParser add_parser(int argc, char** argv) {
    ArgumentParser parser = ArgumentParser(argc,argv).setDescription("Show DAC scan results");
    parser.add_argument("filename").help("Name of the file to analysis").add_finish();
    parser.add_argument("--path").help("Output plots path").set_default(".").add_finish();
    parser.add_argument("--no-fit").help("Do fit?").set_default("true").add_finish();
    parser.parse_args();
    return parser;
}

void file_open(std::ifstream& inputfile, ArgumentParser parser) {
    std::filesystem::path inputfilepath(parser.get_value<std::string>("filename"));
    if (!std::filesystem::exists(inputfilepath)) {
        std::cerr « "File not found: " « inputfilepath « std::endl;
    }
    inputfile = std::ifstream(inputfilepath, std::ios::binary);
}

void input_data(std::vector<DACtoADC>& data, std::ifstream& inputfile) {
    std::string line;

    std::vector<int> dacs;
    std::vector<int> adcs;
    std::string old = "";
    std::string kind;
    std::string dac;
    std::string adc;

    while (getline(inputfile,line)) {
        std::istringstream iss(line);
        std::getline(iss, kind, '\t');
        if (old != kind) {
            DACtoADC temp(old);
            temp.setDAC(dacs);
            temp.setADC(adcs);
            data.push_back(temp);
            old = kind;
            dacs.clear();
            adcs.clear();
        }
        std::getline(iss, dac, '\t');
        dacs.push_back(stoi(dac));
        std::getline(iss, adc, '\t');
        adcs.push_back(stoi(adc));
    }
    data.erase(data.begin());
}

void get_graph(std::vector<TGraph>& graphs, const std::vector<DACtoADC>& datas) {
    for (DACtoADC data : datas) {
        graphs.push_back(TGraph((data.getADC()).size(),data.getDAC().data(),data.getADC().data()));
        graphs.back().SetName((TString) data.getKind());
        graphs.back().SetTitle("Scan of " + (TString) data.getKind() + " DAC; DAC Setting; ADC");
    }
    int i = 0;
    for (TGraph& graph : graphs) {
        if (i < 9) {
            graph.SetMarkerStyle(20);
            graph.SetMarkerSize(.5);
            graph.SetMarkerColor(i+1);
        } else {
            graph.SetMarkerStyle(22);
            graph.SetMarkerSize(.5);
```

```
            graph.SetMarkerColor(i-7);
        }
        i++;
    }
}

void get_multi_graph(TMultiGraph& mg, std::vector<TGraph>& graphs) {
    for (TGraph& graph : graphs) {
        mg.Add(&graph);
    }
}

void set_legend(TLegend& legend, std::vector<TGraph>& graphs){
    legend.SetNColumns(2);

    TMarker mark;
    for (TGraph graph : graphs) {
        graph.SetMarkerSize(1.5);
        legend.AddEntry(graph.Clone(),graph.GetName(),"p");
    }
}

void make_root_file(TMultiGraph& mg, std::vector<TGraph>& graphs, TLegend& legend) {
    TFile rootFile("data/value.root","RECREATE");
    rootFile.mkdir("DAC");
    rootFile.cd("DAC");
    mg.SetName("Total");
    mg.SetTitle("Summary plot; DAC Setting; ADC");
    TCanvas mcan("Total","Merge Canvas", 1200,1000);
    mg.Draw("AP");
    legend.Draw();
    mcan.SetLeftMargin(0.12);
    mcan.Write();
    mg.Write();
    for (TGraph& graph : graphs) {
        graph.Write();
    }
    rootFile.Close();
}

int main(int argc, char** argv) {
    ArgumentParser parser = add_parser(argc,argv);

    std::ifstream inputfile;
    file_open(inputfile, parser);

    std::vector<DACtoADC> data;
    input_data(data, inputfile);

    std::vector<TGraph> graphs;
    get_graph(graphs,data);

    inputfile.close();

    TLegend legend(0.12,0.65,0.45,0.9);
    set_legend(legend,graphs);

    TMultiGraph mg;
    get_multi_graph(mg,graphs);

    make_root_file(mg, graphs, legend);
    return 0;
}
```

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
      hitrate of an ALPIDE");
00018     parser.add_argument("rawdata").metavar("FILENAME").help("raw data file to be
      processed").add_finish();
00019     parser.add_argument("--bins").add_minor_argument("-b").type("int").add_domain({"1", "2", "4", "8",
      "16", "32"}).help("bin size").set_default("1").add_finish();
00020     parser.add_argument("--max").add_minor_argument("-M").type("int").help("color scale
      limit").add_finish();
```

```
    parser.add_argument("--path").help("Output plots path").set_default(".").add_finish();
    parser.add_argument("--dump-raw-hits").help("Dump hit pixel addresses for each event to file").set_default("false").add_finish();
    parser.add_argument("--dump-acc-hits").help("Dump hit pixel addresses sorted by frequency to file").set_default("false").add_finish();
    parser.parse_args();
    return parser;
}

std::ifstream open_file(ArgumentParser parser) {
    std::filesystem::path outfilepath(parser.get_value<std::string>("rawdata"));
    std::string outfilename = outfilepath.stem();

    if (!std::filesystem::exists(outfilepath)) {
        std::cerr « "File not found: " « outfilepath « std::endl;
    }
    std::ifstream outfile(outfilepath,std::ios::binary);
    return outfile;
}

std::vector<uint8_t> read_file(std::ifstream& outfile, int& nev) {
    char buf[sizeof(uint8_t)];
    std::vector<uint8_t> values;

    while (outfile.read(buf, sizeof(buf))) {
        values.push_back(0);
        memcpy(&values[nev], buf, sizeof(values[nev]));
        nev++;
    }
    return values;
}

std::vector<ALPIDE::TEvent> decode_event(std::vector<uint8_t>& values) {
    int i = 0;
    std::vector<ALPIDE::TEvent> events;
    ALPIDE::TDecoder* decoder = new ALPIDE::TDecoder();
    while (i < values.size()) {
        events.push_back(decoder->alpide_decode_event(values,i));
    }
    return events;
}

TCanvas* draw_hitmap(std::vector<ALPIDE::TEvent>& events) {
    TCanvas* c1 = new TCanvas("c1","c1",2000,775);
    TH2I* hm = new TH2I("hm","Hitmap; Column; Row",1024,0,1024,512,0,512);
    for(ALPIDE::TEvent event : events) {
        for(std::array<int,2> coord : event.getCoords()) {
            hm->Fill(coord[0],coord[1]);
        }
    }

    hm->Draw("colz");
    hm->GetXaxis()->SetNdivisions(4,4,4,kFALSE);
    hm->GetXaxis()->SetTickLength(-0.035);
    hm->GetXaxis()->SetLabelOffset(0.03);
    hm->GetXaxis()->CenterTitle();
    hm->GetXaxis()->SetTitleOffset(1.4);

    hm->GetYaxis()->SetNdivisions(2,4,4,kFALSE);
    hm->GetYaxis()->SetTickLength(-0.018);
    hm->GetYaxis()->SetLabelOffset(0.017);
    hm->GetYaxis()->CenterTitle();
    hm->GetYaxis()->SetTitleOffset(0.9);

    hm->GetZaxis()->SetMaxDigits(2);
    hm->GetZaxis()->SetTitle("dE/d[Row]d[Column]");
    hm->GetZaxis()->SetTitleOffset(0.6);

    gStyle->SetTitleX(0.38);
    gStyle->SetTitleY(0.98);
    c1->SetTopMargin(0.085);
    c1->SetBottomMargin(0.115);
    c1->SetLeftMargin(0.065);
    c1->SetRightMargin(0.315);

    std::vector<int> hitrate={0};
    for (int i = 0; i < 1024; i++) {
        for (int j = 0; j < 1024; j++) {
            hitrate.push_back(hm->GetBinContent(i,j));
        }
    }

    sort(hitrate.rbegin(),hitrate.rend());
    TPaveText *pt = new TPaveText(1170,0,1515,256);
    double total = (events.size()*1024.*512.);
    pt->AddText("Hit rate    0 masked:");
    TString str = Form("    %.2e per pixel per event",accumulate(hitrate.begin(), hitrate.end(),
```

```
0)/total);
    pt->AddText(str);
    pt->AddText("Hit rate   10 masked:");
    str = Form("    %.2e per pixel per event",accumulate(hitrate.begin()+10, hitrate.end(), 0)/total);
    pt->AddText(str);
    pt->AddText("Hit rate  100 masked:");
    str = Form("    %.2e per pixel per event",accumulate(hitrate.begin()+100, hitrate.end(),
0)/total);
    pt->AddText(str);
    pt->AddText("Hit rate 1000 masked:");
    str = Form("    %.2e per pixel per event",accumulate(hitrate.begin()+1000, hitrate.end(),
0)/total);
    pt->AddText(str);
    c1->Update();
    pt->SetTextAlign(12);
    pt->Draw("SAME");

    return c1;
}

void save_canvas(ArgumentParser& parser, TCanvas*& c1) {
    std::filesystem::path outfilepath(parser.get_value<std::string>("rawdata"));
    std::string outfilename = outfilepath.stem();
    std::filesystem::path outputfile = parser.get_value<std::string>("path");
    outputfile.append(outfilename+".pdf");
    std::cout « outputfile « std::endl;
    c1->SaveAs((TString) outputfile);
}

TCanvas* draw_histogram(std::vector<ALPIDE::TEvent>& events) {
    gStyle->SetTitleX(0.5);
    gStyle->SetTitleY(1);
    TCanvas* c2 = new TCanvas("c2","c2",2000,1000);
    c2->Divide(2,1);
    c2->cd(1);
    TH1I* eventHist = new TH1I("h1","eventHist",150,0,events.back().getEventNum()+1);
    std::cout « (long int) events[0].getTime() « std::endl;
    TH1I* timeHist = new TH1I("h2","timeHist",150,events[0].getTime(),events.back().getTime());
    for (ALPIDE::TEvent event : events) {
        for (std::array<int,2> coord : event.getCoords()) {
            eventHist->Fill(event.getEventNum());
            timeHist->Fill(event.getTime());
        }
    }
    eventHist->Draw();
    c2->cd(2);
    timeHist->Draw();

    c2->SaveAs("data/histogram.png");

    return c2;
}
int main(int argc, char** argv) {
    TClock* clock = new TClock();
    std::cout « "ALPIDE Decoding..." « std::endl;

    ArgumentParser parser = set_parse(argc,argv);

    std::ifstream outfile = open_file(parser);

    int nev = 0;
    std::vector<uint8_t> values = read_file(outfile,nev);

    std::vector<ALPIDE::TEvent> events = decode_event(values);

    TCanvas* c1 = draw_hitmap(events);

    save_canvas(parser, c1);

    draw_histogram(events);

    clock->EndProgram();
    return 0;
}
```

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     mParser.setDescription("Draw plots for analysis data");
00026     mParser.add_argument("config").help("Config file").set_default("default").add_finish();
00027     mParser.parse_args();
00028 }
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 }
```

```
void ControlExperimentComparison::initComparison() {
    std::filesystem::path inputPath = mConfig->getConfig("FileList").find("input_path");
    std::vector<std::string> inputNameSet =
mConfig->getConfig("FileList").getSubConfig("input_file_name").getValueList();
    for ( std::string_view inputName : inputNameSet ) {
        mFileSet.push_back(std::string((inputPath / inputName).replace_extension(".root")));
    }
    mCompare = new TGraphCompare(mFileSet);
    // mCompare->setObject(mConfig->getConfig("DrawingObject"));
    for ( const std::string& typeName : mTypeNameSet ) {
        CppConfigDictionary typeConfig = mConfig->getConfig(typeName);
        if ( typeConfig.hasKey("clustersize") ) {
            CppConfigDictionary shapeConfig = typeConfig.getSubConfig("clustersize");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mCompare->TCompareClusterSize(typeName, shapeConfig);
        }
        // if ( mConfig->hasConfig("clustersize_ratio") ) {
        //  CppConfigDictionary shapeConfig = mConfig->getConfig("clustersize_ratio");
        //  shapeConfig += mConfig->getConfig("SharedProperty");
        //  mCompare->TCompareClusterSizeRatio(typeName, shapeConfig);
        // }
        // if ( mConfig->hasConfig("clustersize_3D") ) {
        //  CppConfigDictionary shapeConfig = mConfig->getConfig("clustersize_3D");
        //  shapeConfig += mConfig->getConfig("SharedProperty");
        //  mCompare->TCompareClusterSize3D(typeName, shapeConfig);
        // }
    }
}

int main(int argc, char** argv) {
    ControlExperimentComparison controller(argc, argv);
    controller.setConfig();
    controller.initComparison();
}
```

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 "TF1.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 controling 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"));
```

```
    }
    if ( environmentConfig.hasKey("detector_coord_y") ) {
        detectorCoordX = stod(environmentConfig.find("detector_coord_y"));
    }
    simulation.setDetector(detectorX, detectorY, detectorCoordX, detectorCoordY, -(S2CDistance + C2DDistance + collimatorLength));

    // Set Collimator
    double collimatorRadius = 1.5;
    if ( environmentConfig.hasKey("collimator_radius") ) {
        collimatorRadius = stod(environmentConfig.find("collimator_radius"));
    }
    simulation.setCollimator(collimatorRadius, -S2CDistance, -(S2CDistance + collimatorLength));

    // TGraph* graph = new TGraph();
    std::vector<std::pair<std::string, std::vector<std::array<double, 2»» ratio;
    double divideCriteria = 0.;
    if ( graphConfig.find("type") == "comprehensive_point" ) {
        int nGraph = graphConfig.getSubConfig("graphs").getSubConfigSet().size();
        for ( int iGraph = 0; iGraph < nGraph; iGraph++ ) {
            std::vector<std::array<double, 2» tempRatio;
            double length = stod(graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].find("length"));
            std::string widthStr = graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].find("width");
            double min = stod(widthStr.substr(0, widthStr.find(':')));
            widthStr = widthStr.substr(widthStr.find(':') + 1);
            double max = stod(widthStr.substr(0, widthStr.find(':')));
            widthStr = widthStr.substr(widthStr.find(':') + 1);
            double gap = stod(widthStr.substr(0, widthStr.find(':')));
            for ( double width = min; width < max; width += gap ) {
                simulation.setDetector(detectorX, detectorY, detectorCoordX, detectorCoordY, -(S2CDistance + C2DDistance + length));
                simulation.setCollimator(sqrt(width / TMath::Pi()), -S2CDistance, -(S2CDistance + length));
                double entryRatio = simulation.doCount();
                tempRatio.push_back({width, (entryRatio / 2)});
                if ( width < 35.1 && width > 34.9 && length < 1.1 && length > 0.9 ) {
                    divideCriteria = (entryRatio / 2);
                }
                std::cout « length « "\t" « width « "\t" « entryRatio / 2 « std::endl;
            }
            ratio.push_back({std::string(graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].getConfigName()), tempRatio});
        }
    }
    std::vector<TGraph*> entryRatioGraph;

    TGraphUser* graph1 = new TGraphUser(graphConfig.getSubConfig("entryRatio"));
    for ( const std::pair<std::string, std::vector<std::array<double, 2»>&ratioSet : ratio ) {
        entryRatioGraph.push_back(new TGraph());
        for ( const std::array<double, 2>&point : ratioSet.second ) {
            entryRatioGraph.back()->AddPoint(point[0], point[1] / divideCriteria);
        }
        int nGraph = graphConfig.getSubConfig("graphs").getSubConfigSet().size();
        for ( int iGraph = 0; iGraph < nGraph; iGraph++ ) {
            if ( graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph].getConfigName() == ratioSet.first ) {
                graph1->AddGraph(entryRatioGraph.back(), graphConfig.getSubConfig("graphs").getSubConfigSet()[iGraph]);
            }
        }
    }

    for ( const CppConfigDictionary& expData : graphConfig.getSubConfig("exp_data").getSubConfigSet() ) {
        TGraphErrors* eGraph = new TGraphErrors();
        int i = 0;
        for ( const CppConfigDictionary& point : expData.getSubConfig("point").getSubConfigSet() ) {
            Double_t width = stod(point.find("width"));
            Double_t width_error = stod(point.find("width_error"));
            Double_t value = stod(point.find("value"));
            Double_t error = stod(point.find("error"));
            eGraph->SetPoint(i, width, value);
            eGraph->SetPointError(i, width_error, error);
            i++;
        }
        graph1->AddGraph(eGraph, expData);
    }


    // Set output path and create directories if it isn't.
    std::filesystem::path outputPath = fileConfig.find("output_path");
    std::filesystem::create_directories(outputPath);

    graph1->Save(outputPath);
```

```

    return 0;
}
```

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.

```
#include <iostream>
#include <fstream>
#include <filesystem>

// #include "TEvent.h"
#include "CppConfigFile.h"
#include "TAnalyser.h"
#include "TExperimentData.h"
#include "cppargs.h"
#include "TClusterDivideData.h"
#include "TClusterization.h"

#include "TCanvas.h"
#include "TFile.h"

class ControlExperimentAnalysis {
    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;

public:
    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);
};

ControlExperimentAnalysis::ControlExperimentAnalysis(int argc, char** argv) : mParser(argc, argv) {
    mParser.setDescription("Draw plots for analysis data");
    mParser.add_argument("config").help("Config file").set_default("default").add_finish();
    mParser.parse_args();
}

ControlExperimentAnalysis::~ControlExperimentAnalysis() {
    for ( auto& key : mSubConfigSet ) {
        delete key.second;
        key.second = nullptr;
    }
    for ( auto& key : mExpDataSet ) {
        delete key.second;
        key.second = nullptr;
    }
    if ( mInputFile != nullptr && !mInputFile->IsDestructed() ) {
        mInputFile->Close();
        delete mInputFile;
        mInputFile = nullptr;
    }
    if ( mAnalyser != nullptr && !mAnalyser->IsDestructed() ) {
        delete mAnalyser;
        mAnalyser = nullptr;
    }
}

void ControlExperimentAnalysis::setConfig() {
    std::string configPath = mParser.get_value<std::string>("config");
    mConfig = new CppConfigFile(configPath);
```

```
    mTypeNameSet = mConfig->getConfig("File").getSubConfig("type_name").getValueList();
    mInputFilePath = mConfig->getConfig("File").find("input_file");
}

void ControlExperimentAnalysis::openInputFile() {
    mInputFile = new TFile(static_cast<TString>(mInputFilePath), "READ");
}

void ControlExperimentAnalysis::setExpDataSet() {
    for ( const std::string& typeName : mTypeNameSet ) {
        mExpDataSet.insert_or_assign(typeName, new TExperimentData());
    }
}

void ControlExperimentAnalysis::doBasicAnalysis() {
    mAnalyser = new TAnalyser(mInputFile, mExpDataSet);
    mConfig->getConfig("Masking").hasKey("time_stamp_cut");
    mAnalyser->storeEvents(mConfig->getConfig("SharedProperty"));
    mAnalyser->setExpSettingLegend(mConfig->getConfig("ExperimentSetting"));
    if ( mConfig->getConfig("File").hasKey("output_graph") ) {
        mAnalyser->openOutputGraphFile(mConfig->getConfig("File").find("output_graph"));
        for ( std::string_view typeName : mTypeNameSet ) {
            mAnalyser->openDirectory(typeName);
        }
    }
}

void ControlExperimentAnalysis::doMasking() {
    mAnalyser->doMasking(stoi(mConfig->getConfig("Masking").find("cut")));
    for ( std::string_view typeName : mTypeNameSet ) {
        if ( typeName != "Basic" ) {
            mExpDataSet.insert_or_assign(std::string(typeName), mAnalyser->getAnEventSet(typeName));
        }
    }
}

void ControlExperimentAnalysis::drawHitmap() {
    if ( mConfig->getConfig("Basic").hasKey("hitmap") ) {
        CppConfigDictionary hitmapConfig = mConfig->getConfig("Basic").getSubConfig("hitmap");
        hitmapConfig += mConfig->getConfig("SharedProperty");
        mAnalyser->saveHitmap("Basic", hitmapConfig);
    }
    doMasking();
    for ( const std::string& typeName : mTypeNameSet ) {
        if ( typeName != "Basic" ) {
            if ( mConfig->getConfig(typeName).hasKey("hitmap") ) {
                CppConfigDictionary hitmapConfig =
mConfig->getConfig(typeName).getSubConfig("hitmap");
                hitmapConfig += mConfig->getConfig("SharedProperty");
                mAnalyser->saveHitmap(typeName, hitmapConfig);
            }
        }
    }
}

void ControlExperimentAnalysis::clusterization() {
    for ( const std::string& typeName : mTypeNameSet ) {
        TClusterization clusterization(mExpDataSet.find(std::string(typeName))->second->getEvents());
        clusterization.clusterize();
        mExpDataSet.find(std::string(typeName))->second->setClusters(clusterization.getClusters());
    }
}

void ControlExperimentAnalysis::drawClustermapAndClustersize() {

    for ( const std::string& typeName : mTypeNameSet ) {
        if ( mConfig->getConfig(typeName).hasKey("clustermap") ) {
            CppConfigDictionary hitmapConfig =
mConfig->getConfig(typeName).getSubConfig("clustermap");
            hitmapConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveClustermap(typeName, hitmapConfig);
        }
        if ( mConfig->getConfig(typeName).hasKey("clustersize") ) {
            CppConfigDictionary hitmapConfig =
mConfig->getConfig(typeName).getSubConfig("clustersize");
            hitmapConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveClustersize(typeName, hitmapConfig);
        }
    }
}

void ControlExperimentAnalysis::doDivideBySize() {
    mClusterRange = getClusterSizeRange(mConfig->getConfig("ShapeCut"));
    for ( std::string_view typeName : mTypeNameSet ) {
        mAnalyser->doDivideBySize(typeName);
    }
```

```
}

void ControlExperimentAnalysis::drawClusterShapeInfos() {
    for ( const std::string& typeName : mTypeNameSet ) {
        mAnalyser->doShaping(typeName, mClusterRange);
        if ( mConfig->getConfig(typeName).hasKey("shape_individual") ) {
            CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_individual");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveIndividualShapes(typeName, shapeConfig);
        }
        if ( mConfig->getConfig(typeName).hasKey("shape_same_size") ) {
            CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_same_size");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveSameSizeShapes(typeName, shapeConfig);
        }
        if ( mConfig->getConfig(typeName).hasKey("shape_total") ) {
            CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_total");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveTotalShapes(typeName, shapeConfig);
        }
        if ( mConfig->getConfig(typeName).hasKey("shape_same_size_entry") ) {
            CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_same_size_entry");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveSameSizeShapeEntry(typeName, shapeConfig);
        }
        if ( mConfig->getConfig(typeName).hasKey("shape_total_entry") ) {
            CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_total_entry");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveTotalShapeEntry(typeName, shapeConfig);
        }
        if ( mConfig->getConfig(typeName).hasKey("shape_info") ) {
            CppConfigDictionary shapeConfig = mConfig->getConfig(typeName).getSubConfig("shape_info");
            shapeConfig += mConfig->getConfig("SharedProperty");
            mAnalyser->saveSameSizeInfos(typeName, shapeConfig);
        }
    }
}


std::vector<int> ControlExperimentAnalysis::getClusterSizeRange(const CppConfigDictionary privateProperty) {
    std::vector<int> clusterSizeRange;
    if ( privateProperty.hasKey("cluster_size_oi") ) {
        for ( const std::string& rangeStr : privateProperty.getSubConfig("cluster_size_oi").getValueList() ) {
            if ( rangeStr.find('.') != std::string::npos ) {
                for ( int i = stoi(rangeStr.substr(0, rangeStr.find('.'))); i < stoi(rangeStr.substr(rangeStr.find('.') + 3)) + 1; i++ ) {
                    clusterSizeRange.push_back(i);
                }
            } else {
                clusterSizeRange.push_back(stoi(rangeStr));
            }
        }
    } else {
        for ( int i = 0; i < 100; i++ ) {
            clusterSizeRange.push_back(i);
        }
    }
    return clusterSizeRange;
}
```

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;
```

```
int main(int argc, char * argv[]) {

    TApplication app("app", &argc, argv);

    ComponentTcad2d tcad;
    tcad.Initialise("data/pixel_des.grd", "data/pixel_des.dat");

    ComponentAnalyticField field;
    ViewCell view;
    view.SetComponent(&field);
    view.Plot2d();

    // // Define the medium.
    // MediumSilicon si;
    // si.SetTemperature(293.);

    // // Make a plot of the drift velocities.
    // plottingEngine.SetDefaultStyle();

    // // Sensor thickness [cm]
    // constexpr double d = 100.e-4;
    // constexpr double w = 280.e-4;

    // // Make a component with constant drift field and weighting field.
    // // Bias voltage [V]
    // constexpr double vbias = -0.;
    // ComponentConstant uniformField;
    // uniformField.SetArea(-2 * d, 0., -2 * d, 2 * d, d, 2 * d);
    // uniformField.SetMedium(&si);
    // uniformField.SetElectricField(0, vbias / d, 0);
    // uniformField.SetWeightingField(0, -1. / d, 0, "pad");

    // // Depletion voltage [V]
    // constexpr double vdep = .7;
    // // Make a component with linear drift field.
    // auto eLinear = [d,vbias,vdep](const double /*x*/, const double y,
    //                                const double /*z*/,
    //                                double& ex, double& ey, double& ez) {
    //     ex = ez = 0.;
    //     ey = (vbias - vdep) / d + 2 * y * vdep / (d * d);
    // };
    // ComponentUser linearField;
    // linearField.SetArea(-2 * d, 0., -2 * d, 2 * d, d, 2 * d);
    // linearField.SetMedium(&si);
    // linearField.SetElectricField(eLinear);
    // // std::string efield = "ey = " + std::to_string((vbias - vdep) / d) +
    // //                      " + 2 * y * " + std::to_string(vdep / (d * d));
    // // linearField.SetElectricField(efield);

    // // Make a component with analytic weighting field for a strip or pixel.
    // constexpr double pitch = 28.e-4;
    // constexpr double halfpitch = 0.5 * pitch;
    // ComponentAnalyticField wField;
    // wField.SetMedium(&si);
    // wField.AddPlaneY(0, vbias, "back");
    // wField.AddPlaneY(d, 0, "front");
    // wField.AddStripOnPlaneY('z', d, -halfpitch, halfpitch, "strip");
    // wField.AddPixelOnPlaneY(d, -halfpitch, halfpitch,
    //                            -halfpitch, halfpitch, "pixel");

    // // Create a sensor.
    // Sensor sensor;
    // sensor.AddComponent(&linearField);
    // const std::string label = "strip";
    // sensor.AddElectrode(&wField, label);

    // // Plot the drift field if requested.
    // constexpr bool plotField = true;
    // if (plotField) {
    //     ViewField* fieldView = new ViewField();
    //     fieldView->SetSensor(&sensor);
    //     fieldView->SetArea(-0.5 * d, 0, 0.5 * d, d);
    //     fieldView->PlotContour("ey");
    // }
    // // Plot the weighting potential if requested.
    // constexpr bool plotWeightingField = true;
    // if (plotWeightingField) {
    //     ViewField* wfieldView = new ViewField();
    //     wfieldView->SetComponent(&wField);
    //     wfieldView->SetArea(-0.5 * d, 0, 0.5 * d, d);
    //     wfieldView->PlotContourWeightingField("strip", "v");
    // }

    // // Set the time bins.
    // const unsigned int nTimeBins = 1000;
    // const double tmin =  0.;
```

```
    // const double tmax = 10.;
    // const double tstep = (tmax - tmin) / nTimeBins;
    // sensor.SetTimeWindow(tmin, tstep, nTimeBins);

    // // Set up Heed.
    // TrackHeed track(&sensor);
    // // Set the particle type and momentum [eV/c].
    // track.SetParticle("pion");
    // track.SetMomentum(180.e9);

    // // Simulate electron/hole drift lines using MC integration.
    // AvalancheMC drift(&sensor);
    // // Use steps of 1 micron.
    // drift.SetDistanceSteps(1.e-4);

    // // Plot the signal if requested.
    // constexpr bool plotSignal = true;
    // TCanvas* cSignal = nullptr;
    // if (plotSignal) {
    //     cSignal = new TCanvas("cSignal", "", 600, 600);
    // }

    // constexpr bool plotDrift = true;
    // ViewDrift* driftView = nullptr;
    // TCanvas* cDrift = nullptr;
    // if (plotDrift) {
    //     cDrift = new TCanvas("cDrift", "", 600, 600);
    //     driftView = new ViewDrift();
    //     driftView->SetArea(-0.5 * d, 0, -0.5 * d, 0.5 * d, d, 0.5 * d);
    //     driftView->SetCanvas(cDrift);
    //     track.EnablePlotting(driftView);
    // }
    // // Flag to randomise the position of the track.
    // constexpr bool smearx = true;
    // constexpr unsigned int nEvents = 10;
    // // Flag to save the signal to a file.
    // constexpr bool writeSignal = true;
    // for (unsigned int i = 0; i < nEvents; ++i) {
    //     if (plotDrift) driftView->Clear();
    //     // Reset the signal.
    //     sensor.ClearSignal();
    //     if (i % 10 == 0) std::cout « i « "/" « nEvents « "\n";
    //     // Simulate a charged-particle track.
    //     double xt = 0.;
    //     if (smearx) xt = -0.5 * pitch + RndmUniform() * pitch;
    //     track.NewTrack(xt, 0, 0, 0, 0, 1, 0);
    //     // Retrieve the clusters along the track.
    //     for (const auto& cluster : track.GetClusters()) {
    //     // Loop over the electrons in the cluster.
    //     for (const auto& electron : cluster.electrons) {
    //         // Simulate the electron and hole drift lines.
    //         if (plotDrift) {
    //         drift.DisablePlotting();
    //         if (RndmUniform() < 0.01) drift.EnablePlotting(driftView);
    //         }
    //         drift.DriftElectron(electron.x, electron.y, electron.z, electron.t);
    //     }
    //     }
    //     if (plotSignal) {
    //     sensor.PlotSignal(label, cSignal);
    //     cSignal->Update();
    //     gSystem->ProcessEvents();
    //     }
    //     if (plotDrift) {
    //     constexpr bool twod = true;
    //     driftView->Plot(twod);
    //     cDrift->Update();
    //     gSystem->ProcessEvents();
    //     }
    //     // Save the induced current signal to a file.
    //     if (writeSignal) {
    //     char filename[50];
    //     sprintf(filename, "signal_%05d.txt", i);
    //     std::ofstream outfile;
    //     outfile.open(filename, std::ios::out);
    //     for (unsigned int j = 0; j < nTimeBins; ++j) {
    //         const double t = (j + 0.5) * tstep;
    //         const double f = sensor.GetSignal(label, j);
    //         const double fe = sensor.GetElectronSignal(label, j);
    //         const double fh = sensor.GetIonSignal(label, j);
    //         outfile « t « "  " « f « "  " « fe « "  " « fh « "\n";
    //     }
    //     outfile.close();
    //     }
    // }

    // if (plotSignal || plotDrift ||
```

```
//     plotField || plotWeightingField) {
//     app.Run();
// }
}
```

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.

```
#include <iostream>
#include <vector>
#include <string>

#include "cppargs.h"
#include "CppConfigFile.h"
#include "TMerge.h"

ArgumentParser set_parse(int argc, char** argv) {
    ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw plots for analysis data");
    parser.add_argument("config").help("Config file").set_default("default").add_finish();
    parser.parse_args();
    return parser;
}

int main(int argc, char** argv) {
    ArgumentParser parser = set_parse(argc, argv);
    CppConfigFile* config = new CppConfigFile(parser.get_value<std::string>("config"));

    TMergeExperimentROOT* merge = new
 TMergeExperimentROOT(config->getConfig("Merge").find("output_file"),
 config->getConfig("Merge").getSubConfig("input_files").getValueList());

    merge->mergeFile();

    delete config;
    delete merge;
    return 0;
}
```

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 "QBBC.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) {
```

```
    ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw plots for analysis data");
    parser.add_argument("--config").add_minor_argument("-c").help("Config file for controling simulation").set_default("").add_finish();
    parser.parse_args();
    return parser;
}

void DetectorConstruct(DetectorConstruction* detectorConstructor, const CppConfigDictionary& config) {
    double airPressure = Quantity(config.find("air_pressure")).getNum("bar");
    detectorConstructor->SetWorld(airPressure);

    double hallDiameter = Quantity(config.find("hall_diameter")).getNum("mm");
    std::string standType = config.find("stand_type");
    detectorConstructor->SetStand(standType, hallDiameter);

    double shieldWidth = Quantity(config.find("shield_width")).getNum("mm");
    if ( !(shieldWidth < 0.001) ) {
        detectorConstructor->SetShield(shieldWidth);
    }

    std::string alpideType = config.find("alpide_type");
    detectorConstructor->SetALPIDE(alpideType);

    double distance = Quantity(config.find("distance")).getNum("mm");
    detectorConstructor->Construct(standType, distance);
}

void SetParticleSource(G4SingleParticleSource* particleGun, const CppConfigDictionary& config) {
    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
    std::string particleType = config.find("particle_type");
    G4ParticleDefinition* particleDefinition = particleTable->FindParticle(particleType);
    particleGun->SetParticleDefinition(particleDefinition); // /gps/particle alpha
    particleGun->GetPosDist()->SetPosDisType("Plane"); // /gps/pos/type Plane
    particleGun->GetPosDist()->SetPosDisShape("Circle"); // /gps/pos/shape
    particleGun->GetPosDist()->SetRadius(2.5 * mm); // /gps/pos/radius 2.5 * mm
    particleGun->GetPosDist()->SetPosRot1(G4ThreeVector(0., 0., 1.)); // /gps/pos/rot1 0. 0. 1.
    particleGun->GetPosDist()->SetPosRot2(G4ThreeVector(1., 0., 0.)); // /gps/pos/rot2 1. 0. 0.
    particleGun->GetAngDist()->SetAngDistType("iso"); // /gps/ang/type iso
    particleGun->GetEneDist()->SetEnergyDisType("Mono");
    particleGun->GetEneDist()->ApplyEnergyWeight(false); // /gps/ene/type Mono
    double distance = Quantity(config.find("distance")).getNum("mm");
    particleGun->GetPosDist()->SetCentreCoords(G4ThreeVector(0., distance, 0.)); // /gps/pos/centre 0. 2. 0.
}

int main(int argc, char** argv) {
    TTimer timer;
    // Get config file using argument parser.
    ArgumentParser parser = set_parse(argc, argv);
    CppConfigFile config(parser.get_value<std::string>("config"));
    Quantity::setUserQuantity();

    // Set random engine
    CLHEP::RanecuEngine* RandomEngine = new CLHEP::RanecuEngine;
    G4Random::setTheEngine(RandomEngine);

    // Define Run Manager
    G4RunManager* runManager = new G4RunManager;

    // Analysis Manager
    AnalysisManager* analysisManager = new AnalysisManager(config.getConfig("Analysis").find("output_file"));

    // Geometry construction
    DetectorConstruction* detectorConstructor = new DetectorConstruction();
    DetectorConstruct(detectorConstructor, config.getConfig("Geometry"));
    runManager->SetUserInitialization(detectorConstructor);

    // Set physics
    G4VModularPhysicsList* QBBCList = new QBBC(0);
    G4VUserPhysicsList* physicsList = QBBCList;
    physicsList->SetVerboseLevel(0);
    runManager->SetUserInitialization(physicsList);

    // Set initial action
    G4VUserActionInitialization* actionInitialator = new ActionInitialization();
    runManager->SetUserInitialization(actionInitialator);

    // Set Particle Source
    G4GeneralParticleSourceData* GPSData = G4GeneralParticleSourceData::Instance();
    G4SingleParticleSource* particleGun = GPSData->GetCurrentSource();
    SetParticleSource(particleGun, config.getConfig("Source"));

    runManager->Initialize();

    std::ifstream distFile(config.getConfig("Energy").find("alpha_energy_distribution"));
    std::string line;
```

```
    getline(distFile, line);
    while ( getline(distFile, line) ) {
        std::stringstream sstr(line, ',');
        std::string energyStr;
        std::string intensityStr;
        std::getline(sstr, energyStr, ',');
        std::getline(sstr, intensityStr, ',');
        std::getline(sstr, intensityStr, ',');
        if ( intensityStr == "" ) {
            continue;
        }
        double energy = stoi(energyStr) * 0.001;
        int intensity = static_cast<int>(round(stod(intensityStr) * 2000));
        particleGun->GetEneDist()->SetMonoEnergy(energy * MeV);
        runManager->BeamOn(intensity);
    }
    analysisManager->closeBook();

    delete analysisManager;
    delete runManager;
    delete RandomEngine;
    timer.EndProgram();
    return 0;
}
```

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.

```
#include <filesystem>

#include "TGeantAnalyser.h"
#include "cppargs.h"

ArgumentParser set_parse(int argc, char** argv) {
    ArgumentParser parser = ArgumentParser(argc, argv).setDescription("Draw plots for simulation");
    parser.add_argument("simulFile").metavar("FILENAME").help("simulation file").add_finish();
    parser.add_argument("--output").help("output path").set_default("default").add_finish();
    parser.parse_args();
    return parser;
}

int main(int argc, char** argv) {
    // ArgumentParser parser = set_parse(argc, argv);
    // std::filesystem::path input(parser.get_value<std::string>("simulFile"));
    // std::filesystem::path path;

    // if ( parser.get_value<std::string>("output") == "default" ) {
    //  path = std::filesystem::absolute(input.parent_path());
    // } else {
    //  path = parser.get_value<std::string>("output");
    // }

    // std::filesystem::path stem = input.stem();
    // TFile* file = new TFile(static_cast<TString>((input.replace_extension(".root")).string()), "READ");
    // path.append(stem.string());
    // std::filesystem::create_directory(path);
    // TGeantAnalyser drawer(std::move(file));
    // drawer.refineData();
    // std::filesystem::path energyLossFile = "energy_loss_" + stem.string();
    // drawer.saveEnergyLossDistribution((path / energyLossFile.replace_extension(".png")).string());
    // std::filesystem::path clustermapFile = "clustermap_" + stem.string();
    // drawer.saveClustermap((path / clustermapFile.replace_extension(".png")).string());
    // std::filesystem::path doubleClusterFile = "Doubled_cluster_frequencymap_" + stem.string();
    // drawer.saveDoubleClusterFrequencymap((path / doubleClusterFile.replace_extension(".png")).string());

    return 0;
}
```

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", "thresholed_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.

```
#ifndef __TGEANTANALYSER__
#define __TGEANTANALYSER__

#include "TAnalyser.h"

class TGeantAnalyser : public TAnalyser {
private:

    // std::unique_ptr<TTree> tSource;
    // std::unique_ptr<TTree> tTrack;

    // std::unique_ptr<TH1D> eLossHist[6];
    // std::unique_ptr<TH1D> eDep;
    // std::unique_ptr<TH2D> clusterMap;
    // std::unique_ptr<TH2D> angleMap;
    // std::unique_ptr<TH2D> doubleClusterMap;
    // double clusterDensity;
public:
    TGeantAnalyser() = delete;
    // TGeantAnalyser(TString inputFile);
    // TGeantAnalyser(TFile* inputFile);
    // ~TGeantAnalyser();
    // void refineData();
    // void saveEnergyLossDistribution(std::string_view output_file_Name);
    // void saveTotalDepositEnergyInALPIDE(std::string_view output_file_Name);
    // void saveClustermap(std::string_view output_file_Name);
    // void saveAnglemap(std::string_view output_file_Name);
    // void saveDoubleClusterFrequencymap(std::string_view output_file_Name);
    // const double getClusterDensity() const;
};

#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.

```
#include "TGeantAnalyser.h"

// TGeantAnalyser::TGeantAnalyser(TString inputFile) : TAnalyser(inputFile) {}
// TGeantAnalyser::TGeantAnalyser(TFile* inputFile) : TAnalyser(inputFile) {}

// TGeantAnalyser::~TGeantAnalyser() {}

// void TGeantAnalyser::refineData() {
    // Float_t Eloss[6] = {1,1,1,1,1,1};
    // std::array<Double_t,3> position;
    // bool pass[6] = {false,false,false,false,false,false};
    // std::array<Double_t, 2> angle;
    // Double_t minX=3., minZ=3., maxX=-3., maxZ=-3.;
    // Int_t cntParticle = 0;

    // tTrack->SetBranchAddress("posX",&position[0]);
    // tTrack->SetBranchAddress("posY",&position[1]);
    // tTrack->SetBranchAddress("posZ",&position[2]);
    // tTrack->SetBranchAddress("Eloss",&Eloss);
    // tTrack->SetBranchAddress("Passed",&pass);
    // tTrack->SetBranchAddress("dirTheta",&angle[0]);
    // tTrack->SetBranchAddress("dirPhi",&angle[1]);

    // for (int entry = 0; entry < tTrack->GetEntries(); entry++) {
    //     tTrack->GetEntry(entry);
    //     if ((pass[3] || pass[4]) && TMath::Abs(position[0]) < 15. && TMath::Abs(position[2]) < 6.5)
{
    //         clusterMap->Fill(position[0], position[2]);
    //         doubleClusterMap->Fill(position[0],position[2]);
    //         angleMap->Fill(angle[0], angle[1]);
    //         minX=TMath::Min(minX,position[0]);
    //         minZ=TMath::Min(minZ,position[2]);
    //         maxX=TMath::Max(maxX,position[0]);
    //         maxZ=TMath::Max(maxZ,position[2]);
    //         cntParticle++;
    //     }
    //     eLossHist[0]->Fill(Eloss[0]);
    //     eLossHist[1]->Fill(Eloss[1]);
    //     eLossHist[2]->Fill(Eloss[2]);
    //     eLossHist[3]->Fill(Eloss[3]);
    //     eLossHist[4]->Fill(Eloss[4]);
    //     eLossHist[5]->Fill(Eloss[5]);
    //     if (pass[3] && pass[4]) {
    //         eDep->Fill(Eloss[3]+Eloss[4]);
    //     }
    // }
    // Double_t area = TMath::Pi() * pow(((maxX - minX) + (maxZ - minZ) )/4,2);

    // clusterDensity = (0.00025 * 4300 * cntParticle / area) / tTrack -> GetEntries();
    // Double_t probability = (pow(30./32.,2)-TMath::Pi()*(0.09991))/pow(30./32.,2);
    // Double_t density = 0.;
    // for (int x = 0; x < 32; x++) {
    //     for (int y = 0; y < 16; y++) {
    //         density = (0.0025 * 4300 * doubleClusterMap->GetBinContent(x,y) / pow(30./32.,2)) /
tTrack->GetEntries();
    //         doubleClusterMap->SetBinContent(x,y,1-pow(probability, density));
    //         if (doubleClusterMap->GetBinContent(x,y) > 1.) {
    //             std::cout « density « std::endl;
    //         }
    //     }
    // }
// }

// void TGeantAnalyser::saveEnergyLossDistribution(std::string_view title) {
    // for (std::unique_ptr<TH1D>& hist : eLossHist) {
    //     std::unique_ptr<TCanvas> canvase(new TCanvas("can","can",1610,1000));
    //     hist->SetBinContent(1,0);
    //     hist->Draw();
    //     canvase->SaveAs(static_cast<TString>(title));
    // }
// }

// void TGeantAnalyser::saveTotalDepositEnergyInALPIDE(std::string_view title) {
    // std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
    // eDep->Draw();
    // can->SaveAs(static_cast<const TString>(title));
// }

// void TGeantAnalyser::saveClustermap(std::string_view title) {
    // std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
    // clusterMap->Draw();
    // can->SaveAs(static_cast<const TString>(title));
```

```
// }

// void TGeantAnalyser::saveAnglemap(std::string_view title) {
    // std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
    // angleMap->Draw();
    // can->SaveAs(static_cast<const TString>(title));
// }

// void TGeantAnalyser::saveDoubleClusterFrequencymap(std::string_view title) {
    // std::unique_ptr<TCanvas> can(new TCanvas("cmap","Simulated cluster map",1000,500));
    // // doubleClusterMap->SetMaximum(1.);
    // doubleClusterMap->SetStats(0);
    // doubleClusterMap->Draw("COLZ");
    // can->SaveAs(static_cast<const TString>(title));
// }

// const double TGeantAnalyser::getClusterDensity() const {
    // return clusterDensity;
// }
```

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.

```
#ifndef __ACTIONINITIALIZATION__
#define __ACTIONINITIALIZATION__

#ifdef __ACTIONINITIALIZATION_HEADER__
#include "G4ios.hh"
#include "RunAction.h"
#include "EventAction.h"
#include "SteppingAction.h"
#include "TrackingAction.h"
#include "PrimaryGeneratorAction.h"
#endif

#include "G4VUserActionInitialization.hh"


class ActionInitialization : public G4VUserActionInitialization {
public:
    ActionInitialization();
    virtual ~ActionInitialization();

    virtual void Build() const;
    virtual void BuildForMaster() const;
};

#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.

```
#ifndef __ANALYSISMANAGER__
#define __ANALYSISMANAGER__

#ifdef __ANALYSISMANAGER_HEADER__
#include "G4Run.hh"
#include "G4Event.hh"
#include "G4Track.hh"
#include "G4Step.hh"
#include "G4RunManager.hh"
#include"TFile.h"
#include"TTree.h"
#endif

#include <string>

class TFile;
class TTree;

class G4Run;
class G4Event;
class G4Track;
class G4Step;

struct RunTuple {
    int runID;
    int nEvents;
};

struct EventTuple {
    int runID;
    int eventGlobalID;
    int eventLocalID;
    int nTracks;
};

struct TrackTuple {
    int eventGlobalID;
    int trackGlobalID;
    int trackLocalID;
    int trackParentLocalID;
    std::string particleName;
    std::string processName;
    std::string volumeName;
    double genTime;
    double genPosition[3];
    double genKineteicEnergy;
    double genMomentum[3];
    int nSteps;
};

struct StepTuple {
    int trackGlobalID;
    int stepGlobalID;
    std::string volumeName;
    double time;
    double position[3];
    double kineticEnergy;
    double momentum[3];
    double deltaEnergy;
    double totalDepositEnergy;
    double nonIonizingEnergyLoss;
};

class AnalysisManager {
private:
    int runID = 0;
    int eventID = 0;
```

```
    int trackID = 0;
    int stepID = 0;

    static AnalysisManager* fInstance;

    TFile* mOutputFile;
    TTree* mRunTree;
    TTree* mEventTree;
    TTree* mTrackTree;
    TTree* mStepTree;

    RunTuple runTuple;
    EventTuple eventTuple;
    TrackTuple trackTuple;
    StepTuple stepTuple;
public:
    AnalysisManager();
    AnalysisManager(std::string name);
    ~AnalysisManager();

    static AnalysisManager* Instance();

    void setEventID(const int id) { eventID = id; }
    void setTrackID(const int id) { trackID = id; }
    int getEventID() const { return eventID; }
    int getTrackID() const { return trackID; }

    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() { return runTuple; }
    EventTuple& getEventTuple() { return eventTuple; }
    TrackTuple& getTrackTuple() { return trackTuple; }
    StepTuple& getStepTuple() { return stepTuple; }
};

#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.

```
#ifndef __COLOUR__
#define __COLOUR__

#ifdef __COLOUR_HEADER__
#include "G4VisAttributes.hh"
#endif

class G4VisAttributes;

class Colour {
private:
    G4VisAttributes* standColour;
    G4VisAttributes* screenColour;
    G4VisAttributes* alpideColour;
    G4VisAttributes* boardColour;
```

```
public:
    Colour();

    void setStandColour();
    void setScreenColour();
    void setAlpideColour();
    void setBoardColour();

    G4VisAttributes* getStandColour() const;
    G4VisAttributes* getScreenColour() const;
    G4VisAttributes* getAlpideColour() const;
    G4VisAttributes* getBoardColour() const;
};

#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* ALPIDEEpitaxialLogical = nullptr;
00056     G4AssemblyVolume* ALPIDEAssembly = nullptr;
00057     G4LogicalVolume* CarrierBoardLogical = nullptr;
00058
```

```
    SourceType sourceType;
    StandType standType = StandType::none;
    G4double sEnergy = 5.4;
    G4double sDistance = 10.;
    G4double cDiameter = 1.;
public:
    DetectorConstruction();
    virtual ~DetectorConstruction();
    virtual G4VPhysicalVolume* Construct();
    G4VPhysicalVolume* Construct(G4String standType, G4double distance);

    // Setter
    void SetWorld(G4double air_pressure);
    void SetStand(G4String standType, G4double hallDiameter = 0.);
    void SetShield(G4double shieldWidth);
    void SetALPIDE(G4String alpideType);
    void SetCarrierBoard();

    // Getter
    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:
    void DefineCommands();
};

#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.

```
#ifndef __EVENTACTION__
#define __EVENTACTION__

#ifdef __EVENTACTION_HEADER__
#include "G4Event.hh"
#include "AnalysisManager.h"
#include "RunAction.h"
#include "G4GeneralParticleSourceData.hh"
#include "G4SingleParticleSource.hh"
#endif

#include "G4UserEventAction.hh"

class RunAction;
class G4Event;
typedef int G4int;

class EventAction : public G4UserEventAction {
private:
    G4int iStartTrack;

public:
    EventAction(RunAction* runAction);
    virtual ~EventAction();

    virtual void BeginOfEventAction(const G4Event* event);
    virtual void EndOfEventAction(const G4Event* event);
};

#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.

```
#ifndef __MATERIAL__
#define __MATERIAL__

#ifdef __MATERIAL_HEADER__
#include "G4Element.hh"
#include "G4Material.hh"
// #include "G4NistManager.hh"
#include "G4SystemOfUnits.hh"
#endif

class G4Material;
class G4Element;

class Material {
private:
    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;
public:
    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();
};

#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.

```
#ifndef __PRIMARYGENERATORACTION__
#define __PRIMARYGENERATORACTION__

#ifdef __PRIMARYGENERATORACTION_HEADER__
#include "G4ios.hh"
#include "G4GeneralParticleSource.hh"
#endif

#include "G4VUserPrimaryGeneratorAction.hh"

class G4GeneralParticleSource;

class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction {
private:
    G4GeneralParticleSource* fParticleGun;
public:
    PrimaryGeneratorAction();
    virtual ~PrimaryGeneratorAction();

    void setParticleGun();

    void GeneratePrimaries(G4Event* anEvent);

    const G4GeneralParticleSource* GetParticleGun() const;
};

#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.

```
#ifndef __RUNACTION__
#define __RUNACTION__

#ifdef __RUNACTION_HEADER__
#include "G4ios.hh"
#include "G4Run.hh"
#include "AnalysisManager.h"
#endif

#include "G4UserRunAction.hh"
// #include "G4Run.hh"

// #include "PrimaryGeneratorAction.h"
// #include "AnalysisManager.h"
class AnalysisManager;

class RunAction : public G4UserRunAction {
public:
    RunAction();
    virtual ~RunAction();

    virtual void BeginOfRunAction(const G4Run*);
    virtual void EndOfRunAction(const G4Run*);

    AnalysisManager* fAnalysisManager;
};

#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.

```
#ifndef __SOLID__
#define __SOLID__

#ifdef __SOLID_HEADER__
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4SubtractionSolid.hh"
#include "G4UnionSolid.hh"
#include "G4SystemOfUnits.hh"
#endif

class G4VSolid;
```

```


class Solid {
private:
    G4VSolid* alphaStandSolid;
    G4VSolid* betaStandSolid;
    G4VSolid* newStandSolid;
    G4VSolid* screenSolid;
    G4VSolid* alpideCircuitSolid;
    G4VSolid* alpideEpitaxialSolid;
    G4VSolid* boardSolid;
public:
    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;
};

#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.

```
#ifndef __STEPPINGACTION__
#define __STEPPINGACTION__

#ifdef __STEPPINGACTION_HEADER__
#include "G4ios.hh"
#include "EventAction.h"
#include "AnalysisManager.h"
#endif

#include "G4UserSteppingAction.hh"

class EventAction;
class G4Step;

class SteppingAction : public G4UserSteppingAction {
public:
    SteppingAction(EventAction* eventAction);
    virtual ~SteppingAction();

    virtual void UserSteppingAction(const G4Step*);
};

#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.

```
#ifndef __TRACKINGACTION__
#define __TRACKINGACTION__

#ifdef __TRACKINGACTION_HEADER__
#include "G4ios.hh"
#include "AnalysisManager.h"
#endif

#include "G4UserTrackingAction.hh"

class TrackingAction : public G4UserTrackingAction {
public:
    TrackingAction();
    ~TrackingAction();

    virtual void PreUserTrackingAction(const G4Track*);
    virtual void PostUserTrackingAction(const G4Track*);
};

#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.

```
#define __ACTIONINITIALIZATION_HEADER__
#include "ActionInitialization.h"

ActionInitialization::ActionInitialization() : G4VUserActionInitialization() {
    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " is armed" « G4endl;
}

ActionInitialization::~ActionInitialization() {
    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " is terminated" « G4endl;
}

void ActionInitialization::Build() const {
    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " boots" « "\033[1;36m" « " Primary Generator Action" « "\033[0m" « G4endl;
    SetUserAction(new PrimaryGeneratorAction);
    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " boots" « "\033[1;36m" « " Run Action" « "\033[0m" « G4endl;
    RunAction* runAction = new RunAction;
    SetUserAction(runAction);

    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " boots" « "\033[1;36m" « " Event Action" « "\033[0m" « G4endl;
    EventAction* eventAction = new EventAction(runAction);
    SetUserAction(eventAction);

    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " boots" « "\033[1;36m" « " Stepping Action" « "\033[0m" « G4endl;
    SetUserAction(new SteppingAction(eventAction));

    G4cout « "\033[1;35m" « "ActionInitializer" « "\033[0m" « " boots" « "\033[1;36m" « " Tracking Action" « "\033[0m" « G4endl;
    SetUserAction(new TrackingAction);
}

void ActionInitialization::BuildForMaster() const {
    G4cout « "ActionInitialization::BuildForMaster()" « G4endl;
    RunAction* runAction = new RunAction;
    SetUserAction(runAction);
}
```

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.

```
#define __ANALYSISMANAGER_HEADER__
#include "AnalysisManager.h"

AnalysisManager* AnalysisManager::fInstance = nullptr;

AnalysisManager::AnalysisManager() {
    G4cout « "\033[1;35m" « "AnalysisManger" « "\033[0m" « " is armed" « G4endl;
    fInstance = this;
}

AnalysisManager::AnalysisManager(std::string name) {
    G4cout « "\033[1;35m" « "AnalysisManger" « "\033[0m" « " is armed" « G4endl;
    fInstance = this;
    openBook(name);
}

AnalysisManager::~AnalysisManager() {
    G4cout « "\033[1;35m" « "AnalysisManager" « "\033[0m" « " is terminated" « G4endl;
}

AnalysisManager* AnalysisManager::Instance() {
    if ( fInstance == 0 ) {
        fInstance = new AnalysisManager();
    }
    return fInstance;
}

void AnalysisManager::openBook(std::string name) {
    mOutputFile = new TFile(static_cast<TString>(name), "RECREATE");
    setRunTree();
    setEventTree();
    setTrackTree();
    setStepTree();
}

void AnalysisManager::setRunTree() {
    mRunTree = new TTree("runTree", "Tree for recording run informations");
    mRunTree->Branch("Run ID", &runTuple.runID, "runID/I");
    mRunTree->Branch("Number of Event", &runTuple.nEvents, "nEvent/I");
}

void AnalysisManager::setEventTree() {
    mEventTree = new TTree("eventTree", "Tree for recording event informations");
    mEventTree->Branch("Event Global ID", &eventTuple.eventGlobalID, "eventGlobalID/I");
    mEventTree->Branch("Event Local ID", &eventTuple.eventLocalID, "eventLocalID/I");
    mEventTree->Branch("Referenced Run ID", &eventTuple.runID, "runID/I");
    mEventTree->Branch("Number of Track", &eventTuple.nTracks, "nTrack/I");
}

void AnalysisManager::setTrackTree() {
    mTrackTree = new TTree("trackTree", "Tree for recording track informations");
    mTrackTree->Branch("Track Local ID", &trackTuple.trackLocalID, "trackLocalID/I");
    mTrackTree->Branch("Track Global ID", &trackTuple.trackGlobalID, "trackGlobalID/I");
    mTrackTree->Branch("Referenced Event Global ID", &trackTuple.eventGlobalID, "eventGlobalID/I");
    mTrackTree->Branch("Parent Local ID", &trackTuple.trackParentLocalID, "trackParentLocalID/I");
    mTrackTree->Branch("Number of Steps", &trackTuple.nSteps, "nSteps/I");
    mTrackTree->Branch("Particle Name", &trackTuple.particleName, "particleName");
    mTrackTree->Branch("Process Name", &trackTuple.processName, "processName");
    mTrackTree->Branch("Generating Volume Name", &trackTuple.volumeName, "volumeName");
    mTrackTree->Branch("Generating Time", &trackTuple.genTime, "genTime/D");
    mTrackTree->Branch("Generating Position X", &trackTuple.genPosition[0], "genPositionX/D");
    mTrackTree->Branch("Generating Position Y", &trackTuple.genPosition[1], "genPositionY/D");
    mTrackTree->Branch("Generating Position Z", &trackTuple.genPosition[2], "genPositionZ/D");
    mTrackTree->Branch("Generating Kinetic Energy", &trackTuple.genKineteicEnergy, "genKineteicEnergy/D");
    mTrackTree->Branch("Generating Momentum X", &trackTuple.genMomentum[0], "genMomentumX/D");
    mTrackTree->Branch("Generating Momentum Y", &trackTuple.genMomentum[1], "genMomentumY/D");
    mTrackTree->Branch("Generating Momentum Z", &trackTuple.genMomentum[2], "genMomentumZ/D");
}

void AnalysisManager::setStepTree() {
    mStepTree = new TTree("stepTree", "Tree for recording step information");
    mStepTree->Branch("Step Global ID", &stepTuple.stepGlobalID, "stepGlobalID/I");
    mStepTree->Branch("Referenced Track Global ID", &stepTuple.trackGlobalID, "trackGlobalID/I");
    mStepTree->Branch("Volume Name", &stepTuple.volumeName, "volumeName");
    mStepTree->Branch("Time", &stepTuple.time, "time/D");
    mStepTree->Branch("Position X", &stepTuple.position[0], "time/D");
    mStepTree->Branch("Position Y", &stepTuple.position[1], "time/D");
    mStepTree->Branch("Position Z", &stepTuple.position[2], "time/D");
    mStepTree->Branch("Kinetic Energy", &stepTuple.kineticEnergy, "time/D");
    mStepTree->Branch("Momentum X", &stepTuple.momentum[0], "time/D");
    mStepTree->Branch("Momentum Y", &stepTuple.momentum[1], "time/D");
```

```
    mStepTree->Branch("Momentum Z", &stepTuple.momentum[2], "time/D");
    mStepTree->Branch("Delta Energy", &stepTuple.deltaEnergy, "time/D");
    mStepTree->Branch("Total Deposit Energy", &stepTuple.totalDepositEnergy, "time/D");
    mStepTree->Branch("Non Ionizing Deposit Energy", &stepTuple.nonIonizingEnergyLoss, "time/D");
}

void AnalysisManager::RecordingRun(const G4Run* run) {
    runID = run->GetRunID();
    runTuple.runID = runID;
    runTuple.nEvents = run->GetNumberOfEvent();
    mRunTree->Fill();
    mRunTree->AutoSave();
}

void AnalysisManager::RecordingEvent(const G4Event* event) {
    eventTuple.eventGlobalID = eventID;
    eventTuple.eventLocalID = event->GetEventID();
    eventTuple.runID = runID;
    mEventTree->Fill();
    mEventTree->AutoSave();
    eventID++;
}

void AnalysisManager::RecordingTrackStart(const G4Track* track) {
    trackTuple.eventGlobalID = eventID;
    trackTuple.trackGlobalID = trackID;
    if ( track->GetTrackID() != 1 ) {
        trackTuple.trackParentLocalID = track->GetParentID();
        trackTuple.processName = track->GetCreatorProcess()->GetProcessName();
    }
    trackTuple.particleName = track->GetParticleDefinition()->GetParticleName();
    trackTuple.volumeName = track->GetVolume()->GetName();
    trackTuple.genTime = track->GetGlobalTime();
    trackTuple.genPosition[0] = track->GetPosition()[0];
    trackTuple.genPosition[1] = track->GetPosition()[1];
    trackTuple.genPosition[2] = track->GetPosition()[2];
    trackTuple.genKineteicEnergy = track->GetKineticEnergy();
    trackTuple.genMomentum[0] = track->GetMomentum()[0];
    trackTuple.genMomentum[1] = track->GetMomentum()[1];
    trackTuple.genMomentum[2] = track->GetMomentum()[2];
    trackID++;
}

void AnalysisManager::RecordingTrackEnd(const G4Track* track) {
    trackTuple.nSteps = track->GetCurrentStepNumber();
    mTrackTree->Fill();
    mTrackTree->AutoSave();
    trackTuple.trackParentLocalID = 0;
    trackTuple.processName = "";
}

void AnalysisManager::RecordingStep(const G4Step* step) {
    stepTuple.stepGlobalID = stepID;
    stepTuple.trackGlobalID = trackID;
    stepTuple.position[0] = step->GetPostStepPoint()->GetPosition()[0];
    stepTuple.position[1] = step->GetPostStepPoint()->GetPosition()[1];
    stepTuple.position[2] = step->GetPostStepPoint()->GetPosition()[2];
    if ( (abs(stepTuple.position[0]) < 149) && (abs(stepTuple.position[1]) < 149 &&
(abs(stepTuple.position[2]) < 74)) ) {
        stepTuple.volumeName = step->GetPostStepPoint()->GetPhysicalVolume()->GetName();
    } else {
        stepTuple.volumeName = "World";
    }
    stepTuple.time = step->GetPostStepPoint()->GetLocalTime();
    stepTuple.kineticEnergy = step->GetPostStepPoint()->GetKineticEnergy();
    stepTuple.momentum[0] = step->GetPostStepPoint()->GetMomentum()[0];
    stepTuple.momentum[1] = step->GetPostStepPoint()->GetMomentum()[1];
    stepTuple.momentum[2] = step->GetPostStepPoint()->GetMomentum()[2];
    stepTuple.totalDepositEnergy = step->GetTotalEnergyDeposit();
    stepTuple.nonIonizingEnergyLoss = step->GetNonIonizingEnergyDeposit();
    mStepTree->Fill();
    mStepTree->AutoSave();

    stepID++;
}

void AnalysisManager::closeBook() {
    mOutputFile->cd();
    mRunTree->Write();
    mEventTree->Write();
    mTrackTree->Write();
    mStepTree->Write();
    mOutputFile->Close();
}
```

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,
      checkOverlaps);
00026     }
00027
00028     if ( StandLogical != nullptr ) {
00029         if ( standType == "type1" ) {
00030             StandPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, (distance - 1.) * mm, 0. *
      mm), StandLogical, "StandLogical", WorldLogical, false, 0, checkOverlaps);
00031         } else if ( standType == "type2" ) {
00032             StandPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, (distance - 52.5) * mm, 0. *
      mm), StandLogical, "StandLogical", WorldLogical, false, 0, checkOverlaps);
00033         } else if ( standType == "type3" ) {
00034             StandPhysical->SetTranslation(G4ThreeVector(0. * mm, (distance + 2) * mm, -1.25 * mm));
00035         }
00036     }
00037
00038     if ( ShieldLogical != nullptr ) {
00039         if ( standType == "type1" ) {
00040             ShieldPhysical->SetTranslation(G4ThreeVector(0. * mm, (distance - 1. - 4. - 0.5 * 0.108) *
      mm, 0. * mm));
00041         } else if ( standType == "type2" ) {
```

```
            ShieldPhysical = new G4PVPlacement(0, G4ThreeVector(0. * mm, (distance - 56. - 0.5 * 0.108) * mm, 0. * mm), ShieldLogical, "ShieldLogical", WorldLogical, false, 0, checkOverlaps);
        } else if ( standType == "type3" ) {
            ShieldPhysical->SetTranslation(G4ThreeVector(0. * mm, (distance - 1. - .5 * 0.108) * mm, -1.25 * mm));
        }
    }

    if ( CarrierBoardLogical != nullptr ) {
        G4Transform3D t3d = G4Transform3D();
        ALPIDEAssembly->MakeImprint(WorldLogical, t3d);
        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);
    }

    return WorldPhysical;
}

void DetectorConstruction::SetWorld(G4double air_pressure) {
    material->setWorldMaterial(air_pressure);
    G4Box* solidWorld = new G4Box("World", .5 * 300. * mm, .5 * 300. * mm, .5 * 150. * mm);
    WorldLogical = new G4LogicalVolume(solidWorld, material->getWorldMaterial(), "World");
}

void DetectorConstruction::SetStand(G4String standType, G4double hallDiameter) {
    G4VSolid* solid;
    if ( standType == "type1" ) {
        solids->setAlphaStandSolid();
        solid = solids->getAlphaStandSolid();
    } else if ( standType == "type2" ) {
        solids->setBetaStandSolid();
        solid = solids->getBetaStandSolid();
    } else if ( standType == "type3" ) {
        solids->setNewStandSolid(hallDiameter);
        solid = solids->getNewStandSolid();
    } else {
        std::cerr « "\033[1;31m" « "Invalid stand type" « "\033[0m" « std::endl;
        exit(0);
    }
    StandLogical = new G4LogicalVolume(solid, material->getStandMaterial(), "StandLogical");
    StandLogical->SetVisAttributes(colour->getStandColour());
}

void DetectorConstruction::SetShield(G4double shieldWidth) {
    solids->setScreenSolid();
    G4VSolid* solid = solids->getScreenSolid();
    ShieldLogical = new G4LogicalVolume(solid, material->getScreenMaterial(), "ShieldLogical");
    ShieldLogical->SetVisAttributes(colour->getScreenColour());
}

void DetectorConstruction::SetALPIDE(G4String alpideType) {
    if ( alpideType == "insensitive" ) {
        G4VSolid* ALPIDECircuitSolid = solids->getAlpideCircuitSolid();
        G4VSolid* ALPIDEEpitaxialSolid = solids->getAlpideEpitaxialSolid();

        ALPIDECircuitLogical = new G4LogicalVolume(ALPIDECircuitSolid, material->getAlpideMaterial(), "ALPIDECircuitLogical");
        ALPIDECircuitLogical->SetVisAttributes(colour->getAlpideColour());
        ALPIDECircuitLogical->SetUserLimits(new G4UserLimits(1 * um, 1 * um));
        ALPIDEEpitaxialLogical = new G4LogicalVolume(ALPIDEEpitaxialSolid, material->getAlpideMaterial(), "ALPIDEEpitaxialLogical");
        ALPIDEEpitaxialLogical->SetVisAttributes(colour->getAlpideColour());
        ALPIDEEpitaxialLogical->SetUserLimits(new G4UserLimits(1 * um, 1 * um));

        ALPIDEAssembly = new G4AssemblyVolume();
        G4RotationMatrix* Ra = new G4RotationMatrix(0.0 * deg, 0.0 * deg, 0.0 * deg);
        G4ThreeVector circuitVector = G4ThreeVector(0, (50.0 - 11.0 * 0.5) * um, 0);
        ALPIDEAssembly->AddPlacedVolume(ALPIDECircuitLogical, circuitVector, Ra);
        G4ThreeVector epitaxialVector = G4ThreeVector(0, -(11.0) * 0.5 * um, 0);
        ALPIDEAssembly->AddPlacedVolume(ALPIDEEpitaxialLogical, epitaxialVector, Ra);

        SetCarrierBoard();
    } else {
        G4cerr « "\033[1;31m" « "Invalid ALPIDE type" « "\033[0m" « G4endl;
        exit(0);
    }
}

void DetectorConstruction::SetCarrierBoard() {
    G4VSolid* solid = solids->getBoardSolid();
    CarrierBoardLogical = new G4LogicalVolume(solid, material->getBoardMaterial(), "CarrierBoardLogical");
    CarrierBoardLogical->SetVisAttributes(colour->getBoardColour());
}

G4LogicalVolume* DetectorConstruction::GetScoringStand() const {
    return StandLogical;
```

```
}

G4LogicalVolume* DetectorConstruction::GetScoringShield() const {
    return ShieldLogical;
}

G4LogicalVolume* DetectorConstruction::GetScoringALPIDECircuit() const {
    return ALPIDECircuitLogical;
}

G4LogicalVolume* DetectorConstruction::GetScoringALPIDEEpitaxial() const {
    return ALPIDEEpitaxialLogical;
}

G4LogicalVolume* DetectorConstruction::GetScoringCarrierBoard() const {
    return CarrierBoardLogical;
}
```

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);
```

```
}

void Material::setBoardMaterial() {
    fibrousGlass = new G4Material("fibrousGlass", 2.74351 * g / cm3, 7);

    G4Material* SiO2 = new G4Material("SiO2", 2.20 * g / cm3, 2);
    SiO2->AddElement(elSi, 1);
    SiO2->AddElement(elO, 2);

    G4Material* Al2O3 = new G4Material("Al2O3", 3.97 * g / cm3, 2);
    Al2O3->AddElement(elAl, 2);
    Al2O3->AddElement(elO, 3);

    G4Material* Fe2O3 = new G4Material("Fe2O3", 5.24 * g / cm3, 2);
    Fe2O3->AddElement(elFe, 2);
    Fe2O3->AddElement(elO, 3);

    G4Material* CaO = new G4Material("CaO", 3.35 * g / cm3, 2);
    CaO->AddElement(elCa, 1);
    CaO->AddElement(elO, 1);

    G4Material* MgO = new G4Material("MgO", 3.58 * g / cm3, 2);
    MgO->AddElement(elMg, 1);
    MgO->AddElement(elO, 1);

    G4Material* Na2O = new G4Material("Na2O", 2.27 * g / cm3, 2);
    Na2O->AddElement(elNa, 2);
    Na2O->AddElement(elO, 1);

    G4Material* TiO2 = new G4Material("TiO2", 4.24 * g / cm3, 2);
    TiO2->AddElement(elTi, 1);
    TiO2->AddElement(elO, 2);

    fibrousGlass->AddMaterial(SiO2, 60.0 * perCent);
    fibrousGlass->AddMaterial(Al2O3, 11.8 * perCent);
    fibrousGlass->AddMaterial(Fe2O3, 0.1 * perCent);
    fibrousGlass->AddMaterial(CaO, 22.4 * perCent);
    fibrousGlass->AddMaterial(MgO, 3.4 * perCent);
    fibrousGlass->AddMaterial(Na2O, 1.0 * perCent);
    fibrousGlass->AddMaterial(TiO2, 1.3 * perCent);

    epoxyResin = new G4Material("epoxyResin", 1.1250 * g / cm3, 4);
    epoxyResin->AddElement(elC, 38);
    epoxyResin->AddElement(elH, 40);
    epoxyResin->AddElement(elO, 6);
    epoxyResin->AddElement(elBr, 4);

    boardMaterial = new G4Material("Fr4", 1.98281 * mg / cm3, 2);
    boardMaterial->AddMaterial(fibrousGlass, 53 * perCent);
    boardMaterial->AddMaterial(epoxyResin, 47 * perCent);
}

G4Material* Material::getWorldMaterial() const {
    return worldMaterial;
}

G4Material* Material::getStandMaterial() const {
    return standMaterial;
}

G4Material* Material::getScreenMaterial() const {
    return screenMaterial;
}

G4Material* Material::getAlpideMaterial() const {
    return alpideMaterial;
}

G4Material* Material::getBoardMaterial() const {
    return boardMaterial;
}
```

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.

```
#define __PRIMARYGENERATORACTION_HEADER__
#include "PrimaryGeneratorAction.h"

PrimaryGeneratorAction::PrimaryGeneratorAction() : G4VUserPrimaryGeneratorAction(), fParticleGun(0) {
    G4cout « "\033[1;36m" « "Primary Generator Action" « "\033[0m" « " is armed" « G4endl;
    // G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
    // G4String particleName;

    fParticleGun = new G4GeneralParticleSource();
    // fParticleGun->SetParticlePosition(G4ThreeVector(0., +14.0 * mm, 0.));
    // fParticleGun->SetParticleDefinition(particleTable->FindParticle(particleName = "alpha"));
}

PrimaryGeneratorAction::~PrimaryGeneratorAction() {
    G4cout « "\033[1;36m" « "Primary Generator Action" « "\033[0m" « " is terminated" « G4endl;
    delete fParticleGun;
}

void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent) {
    fParticleGun->GeneratePrimaryVertex(anEvent);
}

const G4GeneralParticleSource* PrimaryGeneratorAction::GetParticleGun() const {
    return fParticleGun;
}
```

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.

```
#define __RUNACTION_HEADER__
#include "RunAction.h"

RunAction::RunAction() : G4UserRunAction() {
    G4cout « "\033[1;36m" « "Run Action" « "\033[0m" « " is armed" « G4endl;
    fAnalysisManager = AnalysisManager::Instance();
}

RunAction::~RunAction() {
    G4cout « "\033[1;36m" « "Run Action" « "\033[0m" « " is terminated" « G4endl;
}

void RunAction::BeginOfRunAction(const G4Run* run) {
    G4int nRun = run->GetNumberOfEventToBeProcessed();
    G4cout « "\033[1;32m" « "Run #" « run->GetRunID() « "\033[0m" « "» " « "\033[1;32m" « nRun « "\033[0m";
}

void RunAction::EndOfRunAction(const G4Run* run) {
    AnalysisManager* analysisManager = AnalysisManager::Instance();
    analysisManager->RecordingRun(run);
}
```

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. *
      deg);
```

```
    G4VSolid* solidCollimater = new G4Tubs("solidCollimater", 0., 3.0 * mm, 8.0 * mm, 0., 360.0 * deg);

    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));
    alphaStandSolid = new G4SubtractionSolid("alphaStandSolid", subtraction1, solidCollimater, new G4RotationMatrix(0. * deg, 90. * deg, 90. * deg), G4ThreeVector(0.0 * mm, 4.0 * mm, 0.0 * mm));
}

void Solid::setBetaStandSolid() {
    G4VSolid* solidBody1 = new G4Box("solidBody1", 70. * mm * 0.5, 7.0 * mm * 0.5, 70. * mm * 0.5);
    G4VSolid* solidBody1Hall1 = new G4Tubs("solidBody1Hall1", 0., 15. * mm * 0.5, 6. * mm * 0.5, 0., 360.0 * deg);
    G4VSolid* solidBody1Hall2 = new G4Tubs("solidBody1Hall2", 0., 13. * mm * 0.5, 7. * mm * 0.5, 0., 360.0 * deg);

    G4VSolid* solidBody2 = new G4Tubs("solidBody2", 2. * mm * 0.5, 30. * mm * 0.5, 59. * mm * 0.5, 0., 360.0 * deg);
    G4VSolid* solidBody2Hall = new G4Tubs("solidBody2Hall", 0., 14. * mm * 0.5, 10.2 * mm * 0.5, 0., 360.0 * deg);

    G4VSolid* subtraction = new G4SubtractionSolid("subtraction", solidBody1, solidBody1Hall1, new G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., .5 * mm, 0.));
    G4VSolid* solidStand1 = new G4SubtractionSolid("solidStand1", subtraction, solidBody1Hall2, new G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., 0. * mm, 0.));
    G4VSolid* solidStand2 = new G4SubtractionSolid("solidStand2", solidBody2, solidBody2Hall, new G4RotationMatrix(0. * deg, 0. * deg, 0. * deg), G4ThreeVector(0., 0., -24.6 * mm));

    betaStandSolid = new G4UnionSolid("betaStandSolid", solidStand1, solidStand2, new G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., 33. * mm, 0.));
}

void Solid::setNewStandSolid(double diameter) {
    G4VSolid* solidBody = new G4Box("solidBody", 15. * mm * 0.5, 6. * mm * 0.5, 17.5 * mm * 0.5);
    G4VSolid* solidCollimatedHall = new G4Tubs("solidCollimatedHall", 0., diameter * mm * 0.5, 1.1 * mm * 0.5, 0., 360.0 * deg);
    G4VSolid* solidHolderHall = new G4Tubs("solidHolderHall", 0., 13 * mm * 0.5, 5.2 * mm * 0.5, 0., 360.0 * deg);

    G4VSolid* subtraction = new G4SubtractionSolid("subtraction", solidBody, solidCollimatedHall, new G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., -2.5 * mm, 1.25 * mm));
    newStandSolid = new G4SubtractionSolid("newStandSolid", subtraction, solidHolderHall, new G4RotationMatrix(0. * deg, 90. * deg, 0. * deg), G4ThreeVector(0., .59 * mm, 1.25 * mm));
}

void Solid::setScreenSolid() {
    screenSolid = new G4Box("screenSolid", 30. * mm * 0.5, 0.108 * mm * 0.5, 30. * mm * 0.5);
}

void Solid::setAlpideCircuitSolid() {
    alpideCircuitSolid = new G4Box("ALPIDECircuitSolid", 30.0 * mm * 0.5, 11.0 * um * 0.5, 15.0 * mm * 0.5);
}

void Solid::setAlpideEpitaxialSolid() {
    alpideEpitaxialSolid = new G4Box("ALPIDEEpitaxialSolid", 30.0 * mm * 0.5, (50.0 - 11.0) * um * 0.5, 15.0 * mm * 0.5);
}

void Solid::setBoardSolid() {
    G4Box* solidBody = new G4Box("solidBody", 70. * mm * 0.5, 1.0 * mm * 0.5, 70. * mm * 0.5);
    G4Box* solidEmpty = new G4Box("solidEmpty", 31. * mm * 0.5, 3 * mm, 12.8 * mm * 0.5);
    boardSolid = new G4SubtractionSolid("CarrierBoardSolid", solidBody, solidEmpty, G4Translate3D(0., 0., 2.7 * mm));
}

G4VSolid* Solid::getAlphaStandSolid() const {
    return alphaStandSolid;
}

G4VSolid* Solid::getBetaStandSolid() const {
    return betaStandSolid;
}

G4VSolid* Solid::getNewStandSolid() const {
    return newStandSolid;
}

G4VSolid* Solid::getScreenSolid() const {
    return screenSolid;
}

G4VSolid* Solid::getAlpideCircuitSolid() const {
    return alpideCircuitSolid;
}

G4VSolid* Solid::getAlpideEpitaxialSolid() const {
```

```
    return alpideEpitaxialSolid;
}

G4VSolid* Solid::getBoardSolid() const {
    return boardSolid;
}
```

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))))) /
      pow(10., -floor(log10(func->GetParError(0)))) « std::endl;
00011     int digit = -floor(log10(value));
00012     if ( floor(value * pow(10, digit)) == 1 ) {
00013         return {round(value * pow(10, digit + 1)) / pow(10, digit + 1), digit + 1};
00014     } else {
00015         return {floor(value * pow(10, digit)) / pow(10, digit), digit};
00016     }
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
      10^{%d}", correctedValue, correctedError, correctedDigit);
00029 }
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},
```

```
                                            {666500, 708000, 717200, 712500, 741279, 746400},
                                            {2660000, 2500000, 2516000, 2536000, 2510000},
                                            {2522000, 2518000, 2530000, 2555000},
                                            {2620000, 2533000, 2550000, 2566000, 2532000, 2480000},
                                            {2510000, 2517000, 2590000, 2514000, 2571000, 2450000},
                                            {2510000, 2640000, 2680000, 2650000, 2740000, 2730000},
                                            {0.0708, 0.066, 0.0604, 0.0642, 0.0585},
                                            {1.011, 1.022, 1.013, 1.015},
                                            {4.65, 4.53, 4.55, 4.56, 4.56, 4.57},
                                            {10.16, 10.25, 10.24, 10.22, 10.21, 10.24},
                                            {34.6, 34.8, 34.8, 34.9, 35.1, 35.5}};

    std::vector<std::vector<double»errors = {{30, 20, 20, 18, 20},
                                            {30, 30, 40, 30},
                                            {300, 170, 60, 80, 40, 80},
                                            {200, 300, 200, 300, 400, 130},
                                            {300, 300, 400, 700, 600, 500},
                                            {40000, 20000, 13000, 14000, 30000},
                                            {8000, 11000, 10000, 14000, 30000},
                                            {40000, 7000, 8000, 10000, 8000, 40000},
                                            {20000, 14000, 30000, 7000, 12000, 40000},
                                            {70000, 40000, 50000, 30000, 30000, 40000},
                                            {0.0019, 0.0012, 0.0011, 0.0011, 0.0013},
                                            {0.007, 0.008, 0.007, 0.008},
                                            {0.08, 0.03, 0.03, 0.03, 0.03, 0.08},
                                            {0.1, 0.09, 0.13, 0.07, 0.08, 0.18},
                                            {0.9, 0.6, 0.7, 0.4, 0.4, 0.6}};

    std::vector<TString> titleSet = {
        "#phi 1 collimator; i-th photo; Pixel of collimator hole in photo",
        "#phi 2 collimator; i-th photo; Pixel of collimator hole in photo",
        "#phi 3 collimator; i-th photo; Pixel of collimator hole in photo",
        "#phi 4 collimator; i-th photo; Pixel of collimator hole in photo",
        "#phi 7 collimator; i-th photo; Pixel of collimator hole in photo",
        "#phi 1 collimator; i-th photo; Pixel of collimator case in photo",
        "#phi 2 collimator; i-th photo; Pixel of collimator case in photo",
        "#phi 3 collimator; i-th photo; Pixel of collimator case in photo",
        "#phi 4 collimator; i-th photo; Pixel of collimator case in photo",
        "#phi 7 collimator; i-th photo; Pixel of collimator case in photo",
        "#phi 1 collimator; i-th photo; Area of collimator hole",
        "#phi 2 collimator; i-th photo; Area of collimator hole",
        "#phi 3 collimator; i-th photo; Area of collimator hole",
        "#phi 4 collimator; i-th photo; Area of collimator hole",
        "#phi 7 collimator; i-th photo; Area of collimator hole"};

    std::vector<TString> pathSet = {
        "build/output/hole_size/phi1_collimator_hole_pixels.pdf",
        "build/output/hole_size/phi2_collimator_hole_pixels.pdf",
        "build/output/hole_size/phi3_collimator_hole_pixels.pdf",
        "build/output/hole_size/phi4_collimator_hole_pixels.pdf",
        "build/output/hole_size/phi7_collimator_hole_pixels.pdf",
        "build/output/hole_size/phi1_collimator_case_pixels.pdf",
        "build/output/hole_size/phi2_collimator_case_pixels.pdf",
        "build/output/hole_size/phi3_collimator_case_pixels.pdf",
        "build/output/hole_size/phi4_collimator_case_pixels.pdf",
        "build/output/hole_size/phi7_collimator_case_pixels.pdf",
        "build/output/hole_size/phi1_collimator_area.pdf",
        "build/output/hole_size/phi2_collimator_area.pdf",
        "build/output/hole_size/phi3_collimator_area.pdf",
        "build/output/hole_size/phi4_collimator_area.pdf",
        "build/output/hole_size/phi7_collimator_area.pdf"
    };

    for ( int iPlot = 0; iPlot < values.size(); iPlot++ ) {
        TCanvas* canvas = new TCanvas("can", "", 600, 600);
        TGraphErrors* graph = new TGraphErrors();

        double maxValue = 0.;
        double minValue = 10E+12;

        for ( int i = 0; i < values[iPlot].size(); i++ ) {
            maxValue = maxValue < values[iPlot][i] + errors[iPlot][i] ? values[iPlot][i] + errors[iPlot][i] : maxValue;
            minValue = minValue > values[iPlot][i] - errors[iPlot][i] ? values[iPlot][i] - errors[iPlot][i] : minValue;
            graph->SetPoint(i, i, values[iPlot][i]);
            graph->SetPointError(i, 0, errors[iPlot][i]);
        }

        graph->SetTitle(titleSet[iPlot]);
        graph->SetMarkerStyle(8);
        graph->GetXaxis()->SetNdivisions(values[iPlot].size() + 1, 0, 0, kTRUE);
        graph->GetXaxis()->SetLimits(-1, values[iPlot].size());
        graph->SetMaximum(maxValue + (maxValue - minValue) * .5);
        graph->SetMinimum(minValue - (maxValue - minValue) * .1);
        graph->Draw("AP");

```

```
        TF1* func = new TF1("func", "[0]", 0, values[iPlot].size() - 1);
        graph->Fit(func, "R");
        func->Draw("SAME");

        TPaveText* legend = new TPaveText(.3, .7, .9, .9, "NDC");
        legend->AddText("Fit parameter");
        legend->AddText("Value: " + EffectiveExpression(func->GetParameter(0), func->GetParError(0)));
        legend->AddText(Form("Chi2/NDoF: %f", sqrt(func->GetChisquare() / func->GetNDF())));
        legend->Draw();

        canvas->SetLeftMargin(0.15);
        canvas->SaveAs(pathSet[iPlot]);

        delete func;
        func = nullptr;
        delete graph;
        graph = nullptr;
        delete canvas;
        canvas = nullptr;
    }

    TGraphErrors* graph2 = new TGraphErrors();
    graph2->SetPoint(0, 0.0631, 0.0001415);
    graph2->SetPointError(0, 0.0005, 0.0000006);
    graph2->SetPoint(1, 1.015, 0.00708);
    graph2->SetPointError(1, 0.003, 0.00003);
    graph2->SetPoint(2, 4.554, 0.0525);
    graph2->SetPointError(2, 0.014, 0.0003);
    graph2->SetPoint(3, 10.22, 0.1404);
    graph2->SetPointError(3, 0.03, 0.0005);
    graph2->SetPoint(4, 35.0, 0.3343);
    graph2->SetPointError(4, 0.2, 0.0005);

    TCanvas* canvas2 = new TCanvas("can2", "", 600, 600);
    graph2->SetTitle("Experiment vs Simulation; Collimator hole width [mm^{2}]; Experiment /
 Simulation");
    graph2->SetMarkerStyle(8);
    graph2->SetMarkerSize(.1);
    graph2->SetMinimum(0);
    graph2->SetMaximum(.4);
    graph2->Draw("AP");
    canvas2->SetLeftMargin(.12);
    // canvas2->SetLogx();
    // canvas2->SetLogy();
    canvas2->SaveAs("build/output/hole_size/compare_exp_simul.pdf");
}
```

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.

```
#ifndef __CPPCONFIGDICTIONARY__
#define __CPPCONFIGDICTIONARY__
```

```

#include <iostream>
#include <string>
#include <sstream>
#include <unordered_map>
#include <vector>

#include "CppConfigError.h"

class CppConfigDictionary {
    std::string mConfigName;
    std::unordered_map<std::string, std::string> mDictionary;
    std::vector<CppConfigDictionary> mSubConfigDictionary;
public:
    CppConfigDictionary();
    CppConfigDictionary(const CppConfigDictionary& copy);
    CppConfigDictionary& operator=(const CppConfigDictionary& copy);
    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;

    friend std::ostream& operator<<(std::ostream& os, const CppConfigDictionary& copy);
    CppConfigDictionary& operator+(const CppConfigDictionary& copy);
    CppConfigDictionary& operator+=(const CppConfigDictionary& copy);
};


#endif
```

8.157 /home/ychoi/ATOM/pycpp/config/inc/CppConfigError.h File Reference

```
#include <exception>
#include <string>
```

Classes

- class CppConfigFileError
- class ConfigrableNoValue

8.158 CppConfigError.h

Go to the documentation of this file.

```
#ifndef __CPPCONFIGERROR__
#define __CPPCONFIGERROR__

#include<exception>
#include<string>

class CppConfigFileError : public std::exception {
public:
    std::string mMessage;
public:
```

```
    CppConfigFileError(std::string_view errorType, std::string_view parameter) {
        if ( errorType == "FileExist" ) {
            mMessage = "The Config File \033[1;32m" + std::string(parameter) + "\033[1;0m doesn't exist ";
        } else if ( errorType == "ConfigDictionaryExist" ) {
            mMessage = "The Config Dictionary \033[1;32m" + std::string(parameter) + "\033[1;0m doesn't exist";
        }
    }
    const char* what() const throw() {
        return mMessage.c_str();
    }
};


class ConfigrableNoValue : public std::exception {
public:
    std::string message;
public:
    ConfigrableNoValue(std::string_view key, std::string_view configName) {
        message = static_cast<std::string>(key) + " cannot be found in list of values of " + static_cast<std::string>(configName);
    }
    const char* what() const throw() {
        return message.c_str();
    }
};

#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.

```
#ifndef __CPPCONFIG__
#define __CPPCONFIG__

#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <unordered_map>
#include <fstream>
#include<filesystem>
#include<algorithm>
#include<fstream>

```

```
#include "CppConfigDictionary.h"
#include "CppConfigError.h"

class CppConfigFile {
private:
    std::vector<CppConfigDictionary> mConfigs;
public:
    CppConfigFile();
    CppConfigFile(std::string_view configFile);
    CppConfigFile(const CppConfigFile& copy);
    CppConfigFile& operator=(const CppConfigFile& copy);
    CppConfigFile(CppConfigFile&& move);
    CppConfigFile& operator=(CppConfigFile&& move);
    ~CppConfigFile();
    void addConfig(std::string_view configFile);
    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;
    friend std::ostream& operator«(std::ostream& os, const CppConfigFile& copy);
};



#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 &copy)

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
```

```
}
CppConfigDictionary& CppConfigDictionary::operator=(const CppConfigDictionary& copy) {
    mConfigName = copy.mConfigName;
    mDictionary = copy.mDictionary;
    mSubConfigDictionary = copy.mSubConfigDictionary;

    return *this;
}
CppConfigDictionary::CppConfigDictionary(CppConfigDictionary&& move) : mConfigName(move.mConfigName),
mDictionary(move.mDictionary), mSubConfigDictionary(move.mSubConfigDictionary) {

}
CppConfigDictionary& CppConfigDictionary::operator=(CppConfigDictionary&& move) {
    mConfigName = move.mConfigName;
    mDictionary = move.mDictionary;
    mSubConfigDictionary = move.mSubConfigDictionary;

    mDictionary.clear();
    mSubConfigDictionary.clear();

    return *this;
}


void CppConfigDictionary::addDictionary(std::string_view key, std::string_view value) {
    mDictionary.insert_or_assign(std::string(key), std::string(value));
}

void CppConfigDictionary::addSubConfigDictionary(const CppConfigDictionary& subConfigDictionary) {
    mSubConfigDictionary.push_back(subConfigDictionary);
}

const bool CppConfigDictionary::hasKey(std::string_view key) const {
    for ( const auto& dict : mSubConfigDictionary ) {
        if ( dict.getConfigName() == key ) {
            return true;
        }
    }
    if ( mDictionary.count(std::string(key)) ) {
        return true;
    } else {
        return false;
    }
}

const std::string& CppConfigDictionary::find(const std::string& key) const {
    if ( key.find('/') == std::string::npos ) {
        if ( mDictionary.count(key) == 1 ) {
            return mDictionary.find(key)->second;
        } else {
            return "";
        }
    } else {
        if ( hasKey(key.substr(0, key.find('/')).substr(0, key.find('/'))) ) {
            return getSubConfig(key.substr(0, key.find('/'))).find(key.substr(key.find('/') + 1));
        } else {
            return "";
        }
    }
}

std::string_view CppConfigDictionary::getConfigName() const {
    return mConfigName;
}

std::ostream& operator«(std::ostream& os, const CppConfigDictionary& copy) {
    if ( copy.getConfigName().length() < 8 ) {
        os « copy.getConfigName() « "\t\t";
    } else if ( copy.getConfigName().length() < 16 ) {
        os « copy.getConfigName() « "\t";
    } else {
        os « copy.getConfigName() « std::endl « "\t\t";
    }
    for ( auto& dict : copy.mDictionary ) {
        os « "\033[1;47m" « dict.first « "\033[1;0m" « ", ";
    }
    for ( auto& subConfig : copy.mSubConfigDictionary ) {
        os « "\033[1;42m" « subConfig.getConfigName() « "\033[1;0m" « ", ";
    }
    os « std::endl;
    return os;
}

const std::vector<std::string> CppConfigDictionary::getKeyList() const {
    std::vector<std::string> returnArray;

    for ( const auto& pair : mDictionary ) {
```

```
        returnArray.push_back(pair.first);
    }

    for ( const CppConfigDictionary& subConfigDictionary : mSubConfigDictionary ) {
        for ( std::string_view subConfigKey : subConfigDictionary.getKeyList() ) {
            returnArray.push_back(std::string(subConfigDictionary.getConfigName()) + "/" + std::string(subConfigKey));
        }
    }
    return returnArray;
}

std::unordered_map<std::string, std::string> CppConfigDictionary::getDictionary() {
    return mDictionary;
}

const CppConfigDictionary& CppConfigDictionary::getSubConfig(std::string_view key) const {
    try {
        bool isExist = false;
        for ( const CppConfigDictionary& subConfigDictionary : mSubConfigDictionary ) {
            if ( subConfigDictionary.getConfigName() == key ) {
                isExist = true;
                return subConfigDictionary;
            }
        }
        if ( !isExist ) { throw CppConfigFileError("ConfigDictionaryExist", key); }
    } catch ( CppConfigFileError error ) {
        std::cerr « error.what() « std::endl;
    }
}

const std::vector<std::string> CppConfigDictionary::getValueList() const {
    std::vector<std::string> returnArray;

    for ( const auto& pair : mDictionary ) {
        returnArray.push_back(pair.second);
    }

    return returnArray;
}

CppConfigDictionary& CppConfigDictionary::operator+(const CppConfigDictionary& copy) {
    for ( const auto& pair : copy.mDictionary ) {
        mDictionary.insert_or_assign(pair.first, pair.second);
    }
    for ( const CppConfigDictionary& solo : copy.mSubConfigDictionary ) {
        mSubConfigDictionary.push_back(solo);
    }

    return *this;
}

CppConfigDictionary& CppConfigDictionary::operator+=(const CppConfigDictionary& copy) {
    for ( const auto& pair : copy.mDictionary ) {
        mDictionary.insert_or_assign(pair.first, pair.second);
    }
    for ( const CppConfigDictionary& solo : copy.mSubConfigDictionary ) {
        mSubConfigDictionary.push_back(solo);
    }

    return *this;
}

const std::vector<CppConfigDictionary> CppConfigDictionary::getSubConfigSet() const {
    return mSubConfigDictionary;
}

const std::unordered_map<std::string, CppConfigDictionary> CppConfigDictionary::getSubConfigSetWithName() const {
    std::unordered_map<std::string, CppConfigDictionary> returnMap;
    for ( const CppConfigDictionary& configDictionary : mSubConfigDictionary ) {
        returnMap.insert_or_assign(std::string(configDictionary.getConfigName()), configDictionary);
    }
    return returnMap;
}
```

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.

```
#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 &copy)

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.

```
#include "CppConfigFile.h"

CppConfigFile::CppConfigFile() = default;

CppConfigFile::CppConfigFile(std::string_view configFile) {
    // Find config file. If the file doens't exist, then it omits error message.
    try {
        if ( !std::filesystem::exists(configFile) ) { throw CppConfigFileError("FileExist", configFile); }
        addConfig(configFile);
    } catch ( CppConfigFileError error ) {
        error.what();
        exit(1);
    }
}

CppConfigFile::CppConfigFile(const CppConfigFile& copy) {
    for ( const CppConfigDictionary& dict : copy.mConfigs ) {
        mConfigs.push_back(dict);
    }
}
CppConfigFile& CppConfigFile::operator=(const CppConfigFile& copy) {
    for ( const CppConfigDictionary& dict : copy.mConfigs ) {
        mConfigs.push_back(dict);
    }
    return *this;
}
CppConfigFile::CppConfigFile(CppConfigFile&& copy) {
    for ( const CppConfigDictionary& dict : copy.mConfigs ) {
        mConfigs.push_back(dict);
    }
```

```cpp
}
CppConfigFile& CppConfigFile::operator=(CppConfigFile&& copy) {
    for ( const CppConfigDictionary& dict : copy.mConfigs ) {
        mConfigs.push_back(dict);
    }
    return *this;
}

void CppConfigFile::addConfig(std::string_view configFile) {
    // Open config file
    std::ifstream conf{std::string(configFile)};

    // The reading line
    std::string line;
    // The key
    std::string key;
    // The value is stored as std::string form
    std::string value;

    // Read config file line by line
    std::vector<std::string> valueArray;
    bool isFirst = true;
    while ( getline(conf, line) ) {
        // Remove comment out part
        std::string result;
        bool escapeNext = false;

        for ( char c : line ) {
            if ( escapeNext ) {
                if ( c == '#' ) {
                    result += '#';
                } else {
                    result += '\\';
                    result += c;
                }
                escapeNext = false;
            } else {
                if ( c == '\\' ) {
                    escapeNext = true;
                } else {
                    result += c;
                }
            }
        }
        if ( escapeNext ) {
            result += '\\'; // Add trailing backslash if input ends with a backslash
        }
        line = result;
        line.erase(remove(line.begin(), line.end(), '\t'), line.end());
        if ( line[0] == '#' ) {
            line = "";
        }
        if ( line.find('=') != std::string::npos ) {
            line.erase(remove(line.begin(), line.begin() + line.find('='), ' '), line.begin() + line.find('='));
        } else {
            line.erase(remove(line.begin(), line.end(), ' '), line.end());
        }
        if ( line == "" ) { // Pass the blank line
            continue;
        } else if ( line.find('[') != std::string::npos && (line[line.find('[') - 1] != '\\') ) { // The line started with '[' is considered as Configurable name
            if ( !isFirst ) {
                mConfigs.push_back(getConfigFromArray(key, valueArray));
                valueArray.clear();
            } else {
                isFirst = false;
            }
            key = line.substr(line.find('[') + 1, line.find(']') - 1);
            // Make new configurable
        } else {
            valueArray.push_back(line);
        }
    }
    if ( valueArray.size() != 0 ) {
        mConfigs.push_back(getConfigFromArray(key, valueArray));
    }
}

void CppConfigFile::addConfig(std::string_view configTitle, const std::vector<std::string>& configArray) {
    mConfigs.push_back(getConfigFromArray(configTitle, configArray));
}


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

```
    std::string key, value;
    CppConfigDictionary returnConfigDictionary(motherKey);
    int numBra = 0;
    for ( int lineNum = 0; lineNum < valueArray.size(); lineNum++ ) {
        std::string line = valueArray[lineNum];
        if ( (line.find('=') != std::string::npos && line.find('{') == std::string::npos) && numBra == 0 ) {
            line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() + line.find('=') + 2);
            line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() + line.find('=') + 2);

            key = line.substr(0, line.find('='));
            value = line.substr(line.find('=') + 1);
            while ( true ) {
                if ( value[0] == '\t' || value[0] == ' ' ) {
                    value.erase(value.begin());
                } else {
                    break;
                }
            }
            returnConfigDictionary.addDictionary(key, value);
            key = "";
            value = "";
        } else if ( (line.find('=') != std::string::npos) && (line.find('{') != std::string::npos) && (line[line.find('{') - 1] != '\\') ) {
            line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() + line.find('=') + 2);
            line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() + line.find('=') + 2);
            key = line.substr(0, line.find('='));
            std::vector<std::string> tempValueArray;
            int numSubBra = 0;
            lineNum++;
            while ( true ) {
                line = valueArray[lineNum];
                if ( (line.find('{') != std::string::npos) && (line[line.find('{') - 1] != '\\') ) {
                    line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() + line.find('=') + 2);
                    line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() + line.find('=') + 2);
                    tempValueArray.push_back(line);
                    numSubBra++;
                } else if ( (line.find('}') != std::string::npos) && (line[line.find('}') - 1] != '\\') ) {
                    if ( numSubBra != 0 ) {
                        line.erase(remove(line.begin(), line.end(), '\t'), line.end());
                        line.erase(remove(line.begin(), line.end(), ' '), line.end());
                        tempValueArray.push_back(line);
                    } else {
                        returnConfigDictionary.addSubConfigDictionary(getConfigFromArray(key, tempValueArray));
                        break;
                    }
                    numSubBra--;
                } else {
                    line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() + line.find('=') + 2);
                    line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() + line.find('=') + 2);
                    tempValueArray.push_back(line);
                }
                lineNum++;
            }
        } else if ( (line.find('=') != std::string::npos) && (line.find('{') != std::string::npos) && (line[line.find('{') - 1] == '\\') ) {
            line.erase(remove(line.begin(), line.end(), '\\'), line.end());
            line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, '\t'), line.begin() + line.find('=') + 2);
            line.erase(remove(line.begin(), line.begin() + line.find('=') + 2, ' '), line.begin() + line.find('=') + 2);
            key = line.substr(0, line.find('='));
            value = line.substr(line.find('=') + 1);
            while ( true ) {
                if ( value[0] == '\t' || value[0] == ' ' ) {
                    value.erase(value.begin());
                } else {
                    break;
                }
            }
            returnConfigDictionary.addDictionary(key, value);
            key = "";
            value = "";
        }
    }
    return returnConfigDictionary;
}
```

```

const std::vector<std::string> CppConfigFile::getConfigurableNameList() const {
    std::vector<std::string> configurableNameList;
    for ( const CppConfigDictionary& config : mConfigs ) {
        configurableNameList.push_back(std::string(config.getConfigName()));
    }
    return configurableNameList;
}


const CppConfigDictionary CppConfigFile::getConfig(std::string_view configTitle) const {
    CppConfigDictionary rConfig;
    for ( auto& config : mConfigs ) {
        if ( config.getConfigName() == configTitle ) {
            rConfig = config;
            break;
        }
    }
    return rConfig;
}

const bool CppConfigFile::hasConfig(std::string_view configTitle) const {
    bool hasValue = false;
    for ( auto& config : mConfigs ) {
        if ( config.getConfigName() == configTitle ) {
            hasValue = true;
        }
    }
    return hasValue;
}

std::ostream& operator«(std::ostream& os, const CppConfigFile& copy) {
    std::cout « "Config list (white: keys, green: subConfig)" « std::endl;
    for ( auto& config : copy.mConfigs ) {
        os « config;
    }
    return os;
}

CppConfigFile::~CppConfigFile() { }
```

8.167 /home/ychoi/ATOM/pycpp/inc/cppargs.h File Reference

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <cstdarg>
#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 {
```

```
    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
};

class Argument;

class HelpMessage {
private:
    std::string usage;
public:
    HelpMessage(std::string prog, std::string description);
    void print(std::vector<Argument>& Pos_args, std::vector<Argument>& Opt_args);
};

enum ARGTYPES {
    INT,
    DOUBLE,
    BOOL,
    STRING,
    NONE
};

class Argument {
private:
    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;

public:
    Argument(std::string str);

    // Setter
    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();

    // Getter
    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;
};

class ArgumentParser {
private:
    std::vector<std::string> argv;                              //
    std::string prog = "";                          // The name of the program (default: argv[0])
    std::string description = "";                   // A description of what the program does
    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;
public:
    ArgumentParser(int _argc, char** _argv);
    ~ArgumentParser();

    ArgumentParser& setDescription(std::string _description) { description = _description; return *this; }
    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); // You can set the type in int, double, bool, string.
    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);
};

#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.

```
#ifndef __CPPTIMER__
#define __CPPTIMER__

#include <iostream>
#include <ctime>

class TTimer {
private:
    clock_t start, finish;
    double duration;
public:
    TTimer();
    void Measure();
    void EndProgram();
};

#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.

```
#ifndef __CPPTQDM__
#define __CPPTQDM__

#include <chrono>
#include <iostream>
#include <sys/ioctl.h>
#include <unistd.h>
#include <vector>
#include <iomanip>
#include <math.h>

class ProgressBar {
private:
    std::chrono::system_clock::time_point start_time;
    int mTerminalWidth;
    int mSetSize;
    std::chrono::system_clock::time_point printPoint;
    int called = 0;


public:
    ProgressBar();
    ProgressBar(int setSize);
    ~ProgressBar();

    void getTerminalLength();

    void printProgress();
    int getSecond(int num);
    int getMinute(int num);
};


#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;
```

```
private:
    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);
};

class Quantity {
private:
    double mNum;
    int mDigit;
    Unit mUnit;
    static std::vector<std::tuple<std::string, std::string, Unit» userQuantity;
public:
    static void setUserQuantity(); // There are user unit txt file should be located in same
    directory.
    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);

    friend std::ostream& operator«(std::ostream& os, Quantity& ref);
    friend std::ostream& operator«(std::ostream& os, const Quantity& ref);
};

#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 sentense_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             sentense_size += arg.getArgDenoteOut().size() + 2;
00014             if ( arg.getArgDomain().size() != 0 ) {
00015                 int rest = 0;
00016                 std::cout « " {";
00017                 sentense_size += 2;
00018                 for ( std::string& domain : arg.getArgDomain() ) {
00019                     if ( rest < arg.getArgDomain().size() - 1 ) {
00020                         std::cout « domain « ",";
00021                         sentense_size += domain.size() + 1;
00022                     } else {
00023                         std::cout « domain;
```

```
                        sentense_size += domain.size();
                    }
                    rest++;
                }
                std::cout « "}";
                sentense_size += 1;
            }
            if ( arg.getArgMinorOpt().size() != 0 ) {
                for ( std::string& minorOpt : arg.getArgMinorOpt() ) {
                    std::cout « ", " « minorOpt;
                    sentense_size += 2;
                }
                if ( arg.getArgDomain().size() != 0 ) {
                    int rest = 0;
                    std::cout « " {";
                    sentense_size += 2;
                    for ( std::string& domain : arg.getArgDomain() ) {
                        if ( rest < arg.getArgDomain().size() - 1 ) {
                            std::cout « domain « ",";
                            sentense_size += domain.size() + 1;
                        } else {
                            std::cout « domain;
                            sentense_size += domain.size();
                        }
                        rest++;
                    }
                    std::cout « "}";
                    sentense_size += 1;
                }
            }
            if ( sentense_size < 9 ) {
                std::cout « "\t\t\t";
            } else if ( sentense_size < 17 ) {
                std::cout « "\t\t";
            } else if ( sentense_size < 25 ) {
                std::cout « "\t";
            } else {
                std::cout « std::endl « "\t\t\t";
            }
            std::cout « arg.getArgDescription();
            std::cout « std::endl;
            sentense_size = 0;
        }
    }
    std::cout « std::endl;
    std::cout « "optional arguments:" « std::endl;
    std::cout « "  -h, --help\t\tshow this help message and exit" « std::endl;
    if ( Opt_args.size() != 0 ) {
        for ( Argument& arg : Opt_args ) {
            std::cout « "  " « arg.getArgDenoteOut();
            sentense_size += 2 + arg.getArgDenoteOut().size();
            if ( arg.getArgDomain().size() != 0 ) {
                int rest = 0;
                std::cout « " {";
                sentense_size += 2;
                for ( std::string& domain : arg.getArgDomain() ) {
                    if ( rest < arg.getArgDomain().size() - 1 ) {
                        std::cout « domain « ",";
                        sentense_size += domain.size() + 1;
                    } else {
                        std::cout « domain;
                        sentense_size += domain.size();
                    }
                    rest++;
                }
                std::cout « "}";
                sentense_size += 1;
            }
            if ( arg.getArgMinorOpt().size() != 0 ) {
                for ( std::string minorOpt : arg.getArgMinorOpt() ) {
                    std::cout « ", " « minorOpt;
                    sentense_size += 2;
                }
                if ( arg.getArgDomain().size() != 0 ) {
                    int rest = 0;
                    std::cout « " {";
                    sentense_size += 2;
                    for ( std::string& domain : arg.getArgDomain() ) {
                        if ( rest < arg.getArgDomain().size() - 1 ) {
                            std::cout « domain « ",";
                            sentense_size += domain.size() + 1;
                        } else {
                            std::cout « domain;
                            sentense_size += domain.size();
                        }
                        rest++;
                    }
```

```
                    rest = 0;
                    std::cout « "}";
                    sentense_size += 1;
                }
            }
            if ( sentense_size < 8 ) {
                std::cout « "\t\t\t";
            } else if ( sentense_size < 16 ) {
                std::cout « "\t\t";
            } else if ( sentense_size < 24 ) {
                std::cout « "\t";
            } else {
                std::cout « std::endl « "\t\t\t";
            }
            std::cout « arg.getArgDescription();
            std::cout « std::endl;
            sentense_size = 0;
        }
    }
}

// Argument class
Argument::Argument(std::string str) : _argOpt(str) { }

void Argument::setArgType(ARGTYPES typ) {
    _argType = typ;
}

void Argument::setArgValue(std::string str) {
    _argValueList.push_back(str);
}

void Argument::setArgValueList(std::vector<std::string> strList) {
    _argValueList.reserve(_argValueList.size() + strList.size());
    _argValueList.insert(_argValueList.end(), strList.begin(), strList.end());
}

void Argument::replaceArgValueList(std::vector<std::string> strList) {
    _argValueList.clear();
    _argValueList.reserve(strList.size());
    _argValueList.assign(strList.begin(), strList.end());
}

void Argument::setArgDomain(std::string str) {
    _argDomain.push_back(str);
}

void Argument::setArgDomain(std::vector<std::string> strList) {
    _argDomain.reserve(_argDomain.size() + strList.size());
    _argDomain.insert(_argDomain.end(), strList.begin(), strList.end());
}

void Argument::setArgName(std::string str) {
    _argName = str;
}

void Argument::setArgOpt(std::string str) {
    _argOpt = str;
}

void Argument::setArgMinorOpt(std::string str) {
    _argMinorOpt.push_back(str);
}

void Argument::setArgMinorOpt(std::vector<std::string> strList) {
    _argMinorOpt.reserve(_argMinorOpt.size() + strList.size());
    _argMinorOpt.insert(_argMinorOpt.end(), strList.begin(), strList.end());
}

void Argument::setArgDenoteOut(std::string str) {
    _argDenoteOut = str;
}

void Argument::setArgDenoteIn(std::string str) {
    _argDenoteIn = str;
}

void Argument::setArgDescription(std::string description) {
    _description = description;
}

void Argument::isArgMulti() {
    _isArgMulti = true;
}

void Argument::notArgMulti() {
    _isArgMulti = false;
```

```
}

void Argument::isArgConst() {
    _isArgConst = true;
}

void Argument::notArgConst() {
    _isArgConst = false;
}

ARGTYPES Argument::getArgType() const {
    return _argType;
}

std::string Argument::getArgValue(const int order) const {
    return _argValueList[order];
}

std::vector<std::string> Argument::getArgValueList() const {
    return _argValueList;
}

std::string Argument::getArgDomain(const int order) const {
    return _argDomain[order];
}

std::vector<std::string> Argument::getArgDomain() const {
    return _argDomain;
}

std::string Argument::getArgName() const {
    return _argName;
}

std::string Argument::getArgOpt() const {
    return _argOpt;
}

std::vector<std::string> Argument::getArgMinorOpt() const {
    return _argMinorOpt;
}

std::string Argument::getArgDenoteOut() const {
    return _argDenoteOut;
}

std::string Argument::getArgDenoteIn() const {
    return _argDenoteIn;
}

std::string Argument::getArgDescription() const {
    return _description;
}

bool Argument::getArgMulti() const {
    return _isArgMulti;
}

bool Argument::getArgConst() const {
    return _isArgConst;
}

// ArgumentParser class
ArgumentParser::ArgumentParser(int _argc, char** _argv) {
    for ( int arg = 0; arg < _argc; arg++ ) {
        argv.push_back(_argv[arg]);
    }
    std::string key = "Positional";
    std::vector<std::string> temp_value;

    bool onlyPositional = true;

    for ( std::string& arg : argv ) {
        if ( arg == "--help" || arg == "-h" ) {
            needHelp = true;
        }
        if ( arg[0] != '-' && arg[0] != '.' ) {
            temp_value.push_back(arg);
        }
        if ( arg[0] == '-' ) {
            argv_init.insert({key, temp_value});
            key = arg;
            temp_value.clear();
        }
        if ( arg == argv.back() && onlyPositional ) {
            argv_init.insert({key, temp_value});
        }
```

```
    }
}

ArgumentParser::~ArgumentParser() { }

ArgumentParser& ArgumentParser::add_argument(const std::string& opts) {
    args_temp = new Argument(opts);
    std::string name;
    for ( int i = 0; i < opts.length(); i++ ) {
        if ( opts[i] != '-' ) {
            name = opts.substr(i, opts.length());
            break;
        }
    }
    args_temp->setArgName(name);
    args_temp->setArgDenoteIn(name);
    args_temp->setArgDenoteOut(opts);
    return *this;
}

ArgumentParser& ArgumentParser::add_minor_argument(const std::string& opts) {
    args_temp->setArgMinorOpt(opts);
    return *this;
}

ArgumentParser& ArgumentParser::add_domain(const std::vector<std::string>& opts) {
    args_temp->setArgDomain(opts);
    return *this;
}

ArgumentParser& ArgumentParser::dest(const std::string& str) {
    args_temp->setArgDenoteIn(str);
    return *this;
}

ArgumentParser& ArgumentParser::metavar(const std::string& str) {
    args_temp->setArgDenoteOut(str);
    return *this;
}

ArgumentParser& ArgumentParser::set_const() {
    args_temp->isArgConst();
    return *this;
}

ArgumentParser& ArgumentParser::nargs() {
    args_temp->isArgMulti();
    return *this;
}

ArgumentParser& ArgumentParser::type(std::string typeOpt) {
    ARGTYPES argType;
    if ( typeOpt == "int" ) {
        argType = ARGTYPES::INT;
    } else if ( typeOpt == "double" ) {
        argType = ARGTYPES::DOUBLE;
    } else if ( typeOpt == "bool" ) {
        argType = ARGTYPES::BOOL;
    } else if ( typeOpt == "string" ) {
        argType = ARGTYPES::STRING;
    } else {
        argType = ARGTYPES::NONE;
        std::cout « "NO TYPE WARNING (SET TO ARBITRARY TYPE)" « std::endl;
    }
    args_temp->setArgType(argType);
    return *this;
}

ArgumentParser& ArgumentParser::help(std::string message) {
    args_temp->setArgDescription(message);
    return *this;
}

ArgumentParser& ArgumentParser::set_default(std::string value) {
    args_temp->setArgValue(value);
    return *this;
}

ARGTYPES ArgumentParser::detType(const std::string& str) {
    std::istringstream iss(str);
    int intVal;
    float floatVal;
    if ( iss » intVal && iss.eof() ) {
        return ARGTYPES::INT;
    } else if ( iss.clear(), iss.seekg(0), iss » floatVal && iss.eof() ) {
        return ARGTYPES::DOUBLE;
    } else if ( str == "true" || str == "false" ) {
```

```
        return ARGTYPES::BOOL;
    } else {
        return ARGTYPES::STRING;
    }
}

void ArgumentParser::add_finish() {
    Argument* args_add = args_temp;
    args_add->getArgOpt()[0] == '-' ? Opt_args.push_back(*args_add) : Pos_args.push_back(*args_add);
    args_temp = nullptr;
}

void ArgumentParser::parse_args() {
    if ( needHelp ) {
        HelpMessage* message = new HelpMessage(prog, description);
        message->print(Pos_args, Opt_args);
        exit(0);
    }

    if ( Pos_args.size() > argv_init["Positional"].size() ) {
        std::cout « "ERROR: Not enough required arguments." « std::endl;
        exit(0);
    }

    int numOfMulti = 0;
    for ( Argument& pos : Pos_args ) {
        if ( pos.getArgMulti() ) {
            numOfMulti++;
        }
    }
    if ( numOfMulti > 2 || (numOfMulti == 1 && !Pos_args.back().getArgMulti()) ) {
        std::cout « "Cannot classify that the arguments are belong to group" « std::endl;
        exit(0);
    }

    int numArg = 0;
    for ( Argument& opt : Pos_args ) {
        if ( numArg < Pos_args.size() - 1 ) {
            opt.replaceArgValueList({(argv_init.find("Positional")->second)[0]});
            argv_init.find("Positional")->second.erase(argv_init.find("Positional")->second.begin());
        } else {
            opt.replaceArgValueList(argv_init.find("Positional")->second);
        }
        numArg++;
    }

    for ( Argument& opt : Pos_args ) {
        if ( opt.getArgType() == ARGTYPES::NONE ) {
            opt.setArgType(detType(opt.getArgValueList()[0]));
        } else {
            if ( opt.getArgValueList().size() != 0 && opt.getArgType() != detType(opt.getArgValueList()[0]) ) {
                std::cout « "Not proper type variable" « std::endl;
                exit(0);
            }
        }
        if ( !opt.getArgMulti() && opt.getArgValueList().size() > 1 ) {
            std::cout « "Cannot enter more than 2 variables into this variable" « std::endl;
            exit(0);
        }
    }

    for ( std::pair<std::string, std::vector<std::string» argv : argv_init ) {
        if ( argv.first == "Positional" ) continue;
        bool isArguExist = false;
        for ( Argument& opt : Opt_args ) {
            if ( opt.getArgOpt() == argv.first ) {
                isArguExist = true;
                opt.replaceArgValueList(argv.second);
                break;
            }
        }
        if ( !isArguExist ) {
            std::cout « "Undefined argument" « std::endl;
            exit(0);
        }
    }
    for ( Argument& opt : Opt_args ) {
        if ( opt.getArgType() == ARGTYPES::NONE ) {
            opt.setArgType(detType(opt.getArgValueList()[0]));
        } else {
            if ( opt.getArgValueList().size() != 0 && opt.getArgType() != detType(opt.getArgValueList()[0]) ) {
                std::cout « "Not proper type variable" « std::endl;
                exit(0);
            }
        }
```

```
        if ( !opt.getArgMulti() && opt.getArgValueList().size() > 1 ) {
            std::cout « "Cannot enter more than 2 variables into this variable" « std::endl;
            exit(0);
        }
    }
}

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

}
template<>
int ArgumentParser::get_value<int>(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    int returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            returnValue = stoi(pos.getArgValueList()[0]);
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            returnValue = stoi(opt.getArgValueList()[0]);
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::INT ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}

template<>
double ArgumentParser::get_value<double>(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    double returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            returnValue = stod(pos.getArgValueList()[0]);
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            returnValue = stod(opt.getArgValueList()[0]);
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::DOUBLE ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}

template<>
bool ArgumentParser::get_value<bool>(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    bool returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            returnValue = pos.getArgValueList()[0] == "true" ? true : false;
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
```

```
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            returnValue = opt.getArgValueList()[0] == "true" ? true : false;
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::BOOL ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}
template<>
std::string ArgumentParser::get_value<std::string>(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    std::string returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            returnValue = pos.getArgValueList()[0];
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            returnValue = opt.getArgValueList()[0];
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::STRING ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}

template<>
std::vector<int> ArgumentParser::get_value<std::vector<int»(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    std::vector<int> returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            for ( std::string val : pos.getArgValueList() ) {
                returnValue.push_back(stoi(val));
            }
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            for ( std::string val : opt.getArgValueList() ) {
                returnValue.push_back(stoi(val));
            }
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::INT ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }

    return returnValue;
}
```

```
template<>
std::vector<double> ArgumentParser::get_value<std::vector<double»(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    std::vector<double> returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            for ( std::string val : pos.getArgValueList() ) {
                returnValue.push_back(stod(val));
            }
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            for ( std::string val : opt.getArgValueList() ) {
                returnValue.push_back(stod(val));
            }
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::DOUBLE ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}

template<>
std::vector<bool> ArgumentParser::get_value<std::vector<bool»(const std::string& valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    std::vector<bool> returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            std::transform(pos.getArgValueList().begin(), pos.getArgValueList().end(),
std::back_inserter(returnValue), [ ](const std::string& str) { return str == "true" ? true : false;
});
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
        if ( opt.getArgName() == valueName ) {
            std::transform(opt.getArgValueList().begin(), opt.getArgValueList().end(),
std::back_inserter(returnValue), [ ](const std::string& str) { return str == "true" ? true : false;
});
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::BOOL ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}
template<>
std::vector<std::string> ArgumentParser::get_value<std::vector<std::string»(const std::string&
valueName) {
    bool isexist = false;
    ARGTYPES returnType = ARGTYPES::NONE;
    std::vector<std::string> returnValue;
    for ( Argument pos : Pos_args ) {
        if ( pos.getArgName() == valueName ) {
            returnValue = pos.getArgValueList();
            returnType = pos.getArgType();
            isexist = true;
            break;
        }
    }
    for ( Argument opt : Opt_args ) {
```

```
        if ( opt.getArgName() == valueName ) {
            returnValue = opt.getArgValueList();
            returnType = opt.getArgType();
            isexist = true;
            break;
        }
    }
    if ( !isexist ) {
        std::cout « "No proper argument" « std::endl;
        exit(0);
    }
    if ( returnType != ARGTYPES::STRING ) {
        std::cout « "Not proper type" « std::endl;
        exit(0);
    }
    return returnValue;
}
```

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.

```
#include "cppTimer.h"

TTimer::TTimer() {
    start = clock();
}

void TTimer::Measure() {
    finish = clock();
    duration = (double)(finish-start) / CLOCKS_PER_SEC;
    std::cout « "Time from the start flows " « duration « "s" « std::endl;
}

void TTimer::EndProgram() {
    finish = clock();
    duration = (double)(finish-start) / CLOCKS_PER_SEC;
    std::cout « "Total run time of this program is " « duration « "s" « std::endl;
    std::cout « "Good bye!" « std::endl;
}
```

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.

```
#include "cpptqdm.h"

ProgressBar::ProgressBar(int setSize) : mSetSize(setSize) {
    getTerminalLength();
    start_time = std::chrono::system_clock::now();
    printPoint = start_time;
    std::cout « "\x1b[?25l";
}

```

```
ProgressBar::ProgressBar() {
    getTerminalLength();
    start_time = std::chrono::system_clock::now();
    printPoint = start_time;
}

ProgressBar::~ProgressBar() {
    std::cout « "\x1b[?25h" « std::endl;
}

void ProgressBar::getTerminalLength() {
    struct winsize w;
    ioctl(STDOUT_FILENO, TIOCGWINSZ, &w);
    mTerminalWidth = w.ws_col;
}

int ProgressBar::getMinute(int num) {
    return (num % 3600) / 60;
}

int ProgressBar::getSecond(int num) {
    return num % 60;
}

void ProgressBar::printProgress() {
    called++;
    std::chrono::system_clock::time_point now = std::chrono::system_clock::now();
    if ( std::chrono::duration_cast<std::chrono::milliseconds>(now - printPoint).count() > 1 || called == mSetSize || called == 1 ) {
        double percent = (double) called / mSetSize * 100;
        double duration = std::chrono::duration_cast<std::chrono::milliseconds>(now - start_time).count();
        double speed = 1000. * called / duration;
        int left = (mSetSize - called) / speed;
        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));
        printPoint = now;
        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(' ') « "]"
            « std::setw(floor(log10(mSetSize)) + 1) « std::setfill(' ') « called « "/" « mSetSize
            « " ["
            « std::setw(2) « std::setfill('0') « getMinute(duration * 0.001) « ":" « std::setw(2) « std::setfill('0') « getSecond(duration * 0.001) « "<"
            « std::setw(2) « std::setfill('0') « getMinute(left) « ":" « std::setw(2) « std::setfill('0') « getSecond(left) « ", "
            « std::setprecision(10 - 1 - (speed == 0 ? 1 : (int) log10(abs((int) speed)) + 1)) « speed « "it/s]";
    }

}
```

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.

```
#include "cppUnit.h"

Unit::Unit() { }

Unit::Unit(std::string_view unit) {
    setUnit(unit);
}

std::vector<std::tuple<std::string, std::string, Unit» Quantity::userQuantity = { };
std::vector<std::pair<std::string, std::array<int, 7»> Unit::userUnits = { };
std::vector<std::string> Unit::exceptPrefixList = {"m", "mol", "cd"};
bool Unit::isUserUnit = false;

void Unit::setUserUnit(std::string_view newUnit, std::string_view newSiUnit) {
    Unit object;
    std::vector<std::string> store;
    object.seperate(store, newSiUnit);
    object.vanishSlash(store);
    object.removePrefix(store);

    int iNow = 0;
    for ( std::string& str : store ) {
        if ( isalpha(str[0]) ) {
            object.setUnitCount(store[iNow], stoi(store[iNow + 1]));
        }
        iNow++;
    }
    userUnits.push_back({std::string(newUnit), object.getUnitCount()});
    if ( prefix.count(newUnit[0]) ) {
        exceptPrefixList.push_back(std::string(newUnit));
    }
    isUserUnit = true;

}

void Quantity::setUserQuantity() {
    std::ifstream unitFile("~/source/ATOM/src/pycpp/UserUnit.txt");
```

```
    std::string str;

    while ( getline(unitFile, str) ) {
        std::istringstream strstr(str);
        std::string newUnit, relation, newSiUnit;
        strstr » newUnit » relation » newSiUnit;
        Unit::setUserUnit(newUnit, newSiUnit);
        Quantity::userQuantity.push_back(std::make_tuple(newUnit, relation, Unit(newSiUnit)));
    }
    unitFile.close();
}

void Unit::setUnit(std::string_view unit) {
    std::vector<std::string> store;

    if ( seperate(store, unit) || vanishSlash(store) || removePrefix(store) ) { }

    int iNow = 0;
    for ( std::string& str : store ) {
        if ( isalpha(str[0]) ) {
            setUnitCount(store[iNow], stoi(store[iNow + 1]));
        }
        iNow++;
    }
}

bool Unit::seperate(std::vector<std::string>& store, std::string_view unit) {
    int iStart = 0;
    int iNow = 0;

    bool doubleSlash = false;
    bool isNum = false;

    bool error = false;
    for ( const char letter : unit ) {
        if ( letter == '*' ) {
            store.push_back(static_cast<std::string>(unit.substr(iStart, iNow - iStart)));
            store.push_back(static_cast<std::string>(unit.substr(iNow, 1)));
            iStart = iNow + 1;
        } else if ( letter == '/' ) {
            if ( doubleSlash ) {
                std::cerr « "Unit notation to write '/' twice is forbidden" « std::endl;
                error = true;
            }
            store.push_back(static_cast<std::string>(unit.substr(iStart, iNow - iStart)));
            store.push_back(static_cast<std::string>(unit.substr(iNow, 1)));
            iStart = iNow + 1;
            doubleSlash = true;
        } else if ( (letter == '-' || isdigit(letter)) && !isNum ) {
            store.push_back(static_cast<std::string>(unit.substr(iStart, iNow - iStart)));
            isNum = true;
            iStart = iNow;
        } else if ( isNum && !isdigit(letter) ) {
            iStart = iNow;
            isNum = false;
        }
        if ( iNow == unit.size() - 1 ) {
            store.push_back(static_cast<std::string>(unit.substr(iStart)));
        }
        iNow++;
    }
    return error;
}

bool Unit::vanishSlash(std::vector<std::string>& store) {
    int iNow = 0;
    bool afterSlash = false;
    while ( store.end() - store.begin() - iNow > 0 ) {
        std::string str = store[iNow];
        if ( str == "/" ) {
            afterSlash = true;
            store[iNow] = "*";
        }
        if ( (str[0] == '-' || isdigit(str[0])) && afterSlash ) {
            store[iNow] = "-" + store[iNow];
        }
        if ( isalpha(str[0]) && ((store[iNow + 1][0] != '-' && !isdigit(store[iNow + 1][0])) ||
(store.begin() + iNow + 1) == store.end()) ) {
            store.insert(store.begin() + iNow + 1, "1");
        }
        iNow++;
    }
    return false;
}

bool Unit::removePrefix(std::vector<std::string>& store) {
    mDigit = 0;
```

```

    int iNow = 0;
    for ( std::string_view str : store ) {
        if ( prefix.count(str[0]) ) {
            if ( std::find(exceptPrefixList.begin(), exceptPrefixList.end(), str) != exceptPrefixList.end() ) {
            } else if ( str.substr(1) == "g" ) {
                mDigit += (prefix.find(str[0])->second - 3) * stoi(store[iNow + 1]);
                store[iNow] = "kg";
            } else {
                mDigit += prefix.find(str[0])->second * stoi(store[iNow + 1]);
                store[iNow] = str.substr(1);
            }
        } else if ( isalpha(str[0]) ) {
        }
        iNow++;
    }
    return false;
}

void Unit::setUnitCount(std::string unit, int power) {
    if ( unit == "m" ) {
        plusCount({1 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power});
    } else if ( unit == "kg" ) {
        plusCount({0 * power, 1 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power});
    } else if ( unit == "s" ) {
        plusCount({0 * power, 0 * power, 1 * power, 0 * power, 0 * power, 0 * power, 0 * power});
    } else if ( unit == "A" ) {
        plusCount({0 * power, 0 * power, 0 * power, 1 * power, 0 * power, 0 * power, 0 * power});
    } else if ( unit == "K" ) {
        plusCount({0 * power, 0 * power, 0 * power, 0 * power, 1 * power, 0 * power, 0 * power});
    } else if ( unit == "mol" ) {
        plusCount({0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 1 * power, 0 * power});
    } else if ( unit == "cd" ) {
        plusCount({0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 0 * power, 1 * power});
    } else if ( isUserUnit ) {
        for ( const std::pair<std::string, std::array<int, 7»&userUnit : userUnits ) {
            if ( userUnit.first == unit ) {
                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});
            }
        }
    } else {
        std::cerr « "’" « unit « "; is un-defined unit. Set user difine option." « std::endl;
    }
}

void Unit::plusCount(std::array<int, 7> nums) {
    unitCount[0] += nums[0];
    unitCount[1] += nums[1];
    unitCount[2] += nums[2];
    unitCount[3] += nums[3];
    unitCount[4] += nums[4];
    unitCount[5] += nums[5];
    unitCount[6] += nums[6];
}

bool Unit::operator==(const Unit& ref) const {
    return ref.unitCount == unitCount;
}

bool Unit::operator!=(const Unit& ref) const {
    return ref.unitCount != unitCount;
}

Unit Unit::operator*(const Unit& ref) const {
    Unit result = *this;
    result *= ref;
    return result;
}

Unit Unit::operator*=(const Unit& ref) {
    mDigit += ref.mDigit;
    for ( int i = 0; i < 7; i++ ) {
        unitCount[i] += ref.unitCount[i];
    }
    for ( std::string_view nonBasic : ref.nonBasicUnit ) {
        if ( find(nonBasicUnit.begin(), nonBasicUnit.end(), nonBasic) != nonBasicUnit.end() ) {
            nonBasicUnit.push_back(static_cast<std::string>(nonBasic));
        }
    }
    return *this;
}

Unit Unit::operator/(const Unit& ref) const {
    Unit result = *this;
```

```
    result /= ref;
    return result;
}

Unit Unit::operator/=(const Unit& ref) {
    mDigit -= ref.mDigit;
    for ( int i = 0; i < 7; i++ ) {
        unitCount[i] -= ref.unitCount[i];
    }
    for ( std::string_view nonBasic : ref.nonBasicUnit ) {
        if ( find(nonBasicUnit.begin(), nonBasicUnit.end(), nonBasic) != nonBasicUnit.end() ) {
            nonBasicUnit.push_back(static_cast<std::string>(nonBasic));
        }
    }
    return *this;
}

std::ostream& operator«(std::ostream& os, Unit& ref) {
    std::string unitTable[7] = {"m", "kg", "s", "A", "K", "mol", "cd"};
    int multiCount = 7 - std::count(std::begin(ref.unitCount), std::end(ref.unitCount), (int) 0);
    for ( int i = 0; i < 7; i++ ) {
        if ( ref.unitCount[i] != 0 ) {
            os « unitTable[i] « (int) ref.unitCount[i];
            multiCount--;
            if ( multiCount > 0 ) {
                os « "*";
            }
        }
    }
    return os;
}

std::ostream& operator«(std::ostream& os, const Unit& ref) {
    std::string unitTable[7] = {"m", "kg", "s", "A", "K", "mol", "cd"};
    int multiCount = 7 - std::count(std::begin(ref.unitCount), std::end(ref.unitCount), (int) 0);
    for ( int i = 0; i < 7; i++ ) {
        if ( ref.unitCount[i] != 0 ) {
            os « unitTable[i] « (int) ref.unitCount[i];
            multiCount--;
            if ( multiCount > 0 ) {
                os « "*";
            }
        }
    }
    return os;
}

const int Unit::getDigit() const {
    return mDigit;
}

const std::array<int, 7>& Unit::getUnitCount() const {
    return unitCount;
}

const std::vector<std::string> Unit::getNonBasicUnits() const {
    return nonBasicUnit;
}

const std::string Unit::getUnit() const {
    std::string unitTable[7] = {"m", "kg", "s", "A", "K", "mol", "cd"};
    int multiCount = 7 - std::count(std::begin(unitCount), std::end(unitCount), (int) 0);
    std::string result;
    for ( int i = 0; i < 7; i++ ) {
        if ( unitCount[i] != 0 ) {
            result += unitTable[i] + std::to_string((int) unitCount[i]);
            multiCount--;
            if ( multiCount > 0 ) {
                result += "*";
            }
        }
    }
    return result;
}

Quantity::Quantity(std::string quantity) {
    int posChar = 0;
    for ( const uint letter : quantity ) {
        if ( isdigit(letter) || (letter == '.') ) {
            posChar++;
        } else {
            break;
        }
    }
    mNum = stof(quantity.substr(0, posChar));
    mDigit = floor(log10(mNum));
    mNum = mNum * pow(10, -mDigit);
```

```
    mUnit = Unit(quantity.substr(posChar));
    mDigit += mUnit.getDigit();
}

Quantity::Quantity(double num, std::string unit) {
    mDigit = floor(log10(mDigit));
    mNum = mNum * pow(10, -mDigit);
    mUnit = Unit(unit);
}

double Quantity::getNum() const {
    return mNum * pow(10, mDigit);
}

double Quantity::getNum(std::string_view unit) const {
    Unit givenUnit(unit);
    if ( mUnit != givenUnit ) {
        std::cerr « "The operands doesn’t have same dimension." « mUnit « "!=" « givenUnit « std::endl;
    }
    return mNum * pow(10, mDigit - givenUnit.getDigit());
}

const std::string Quantity::getUnit() const {
    return mUnit.getUnit();
}

const std::string Quantity::getQuantity() const {
    return std::to_string(mNum * pow(10, mDigit));
}

const std::string Quantity::getQuantity(std::string_view unit) const {
    Unit givenUnit(unit);
    if ( mUnit != givenUnit ) {
        std::cerr « "The operands doesn’t have same dimension." « mUnit « "!=" « givenUnit « std::endl;
    }
    return std::to_string(mNum * pow(10, mDigit - givenUnit.getDigit())) + static_cast<std::string>(unit);
}

Quantity Quantity::operator+(const Quantity& ref) const {
    Quantity result = *this;
    result += ref;
    return result;
}
Quantity Quantity::operator-(const Quantity& ref) const {
    Quantity result = *this;
    result -= ref;
    return result;
}
Quantity Quantity::operator*(const Quantity& ref) const {
    Quantity result = *this;
    result *= ref;
    return result;
}
Quantity Quantity::operator/(const Quantity& ref) const {
    Quantity result = *this;
    result /= ref;
    return result;
}
Quantity Quantity::operator+=(const Quantity& ref) {
    if ( mUnit != ref.mUnit ) {
        std::cerr « "The operands doesn’t have same dimension." « mUnit « "!=" « ref.mUnit « std::endl;
    } else {
        int mainDigit = std::max(mDigit, ref.mDigit);
        mNum = mNum * pow(10, mDigit - mainDigit) + ref.mNum * pow(10, ref.mDigit - mainDigit);
        mDigit = mainDigit + floor(log10(mNum));
        mNum = mNum * pow(10, -floor(log10(mNum)));
    }
    return *this;
}
Quantity Quantity::operator-=(const Quantity& ref) {
    if ( mUnit != ref.mUnit ) {
        std::cerr « "The operands doesn’t have same dimension." « mUnit « "!=" « ref.mUnit « std::endl;
    } else {
        int mainDigit = std::max(mDigit, ref.mDigit);
        mNum = mNum * pow(10, mDigit - mainDigit) - ref.mNum * pow(10, ref.mDigit - mainDigit);
        mDigit = mainDigit + floor(log10(abs(mNum)));
        mNum = mNum * pow(10, -floor(log10(abs(mNum))));
    }
    return *this;
}
Quantity Quantity::operator*=(const Quantity& ref) {
```

```
    int mainDigit = std::max(mDigit, ref.mDigit);
    mNum = mNum * pow(10, mDigit - mainDigit) * ref.mNum * pow(10, ref.mDigit - mainDigit);
    mDigit = mainDigit + floor(log10(abs(mNum)));
    mNum = mNum * pow(10, -floor(log10(abs(mNum))));
    mUnit *= ref.mUnit;
    return *this;
}
Quantity Quantity::operator/=(const Quantity& ref) {
    int mainDigit = std::max(mDigit, ref.mDigit);
    mNum = mNum * pow(10, mDigit - mainDigit) / (ref.mNum * pow(10, ref.mDigit - mainDigit));
    mDigit = mainDigit + floor(log10(abs(mNum)));
    mNum = mNum * pow(10, -floor(log10(abs(mNum))));
    mUnit /= ref.mUnit;
    return *this;
}

std::ostream& operator«(std::ostream& os, Quantity& ref) {
    os « ref.mNum « "e" « ref.mDigit « " " « ref.mUnit;
    return os;
}

std::ostream& operator«(std::ostream& os, const Quantity& ref) {
    os « ref.mNum « "e" « ref.mDigit « " " « ref.mUnit;
    return os;
}
```

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.

```
#ifndef __TENTRYSIMULATION__
#define __TENTRYSIMULATION__

#include <cmath>
#include <iostream>

#include "TMath.h"
#include "TGraph2D.h"
#include "TCanvas.h"

struct TDisk {
    double radius;
    double coordZ;
    bool isBelong(double x, double y) {
        if ( pow(x, 2) + pow(y, 2) < pow(radius, 2) ) {
            return true;
```

```
        } else {
            return false;
        }
    }
};

struct TDetector {
    double width, height;
    double coordX, coordY, coordZ;
    bool isBelong(double x, double y) {
        if ( (abs(x - coordX) < width / 2) && (abs(y - coordY) < height / 2) ) {
            return true;
        } else {
            return false;
        }
    }
};

class TEntrySimulation {
private:
    TDisk source;
    TDisk upperDisk;
    TDisk lowerDisk;
    TDetector detector;
public:
    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();
};



#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.

```
#include "TEntrySimulation.h"

void TEntrySimulation::setInitGeometry(double diskRadius, double upperDiskCoordZ, double
lowerDiskCoordZ, double detectorWidth, double detectorHeight, double detectorCoordZ) {
    upperDisk.radius = diskRadius;
    upperDisk.coordZ = upperDiskCoordZ;
    lowerDisk.radius = diskRadius;
    lowerDisk.coordZ = lowerDiskCoordZ;
    detector.width = detectorWidth;
    detector.height = detectorHeight;
    detector.coordZ = detectorCoordZ;
}

void TEntrySimulation::setSource(double sourceRadius) {
    source.radius = sourceRadius;
    source.coordZ = 0.;
}

void TEntrySimulation::setCollimator(double diskRadius, double upperDiskCoordZ, double
lowerDiskCoordZ) {
    upperDisk.radius = diskRadius;
    upperDisk.coordZ = upperDiskCoordZ;
    lowerDisk.radius = diskRadius;
    lowerDisk.coordZ = lowerDiskCoordZ;
}
```

```

void TEntrySimulation::setDetector(double detectorWidth, double detectorHeight, double detectorCoordX, double detectorCoordY, double detectorCoordZ) {
    detector.width = detectorWidth;
    detector.height = detectorHeight;
    detector.coordX = detectorCoordX;
    detector.coordY = detectorCoordY;
    detector.coordZ = detectorCoordZ;
}

double TEntrySimulation::doCount() {
    int nCount = 0;
    int totCount = 0;
    double effAngle = 0.;
    double sourceStep = source.radius / 50.;
    int iPoint = 0;
    for ( double x = -source.radius; x < source.radius + 2 * sourceStep; x += sourceStep ) {
        for ( double y = -source.radius; y < source.radius + 2 * sourceStep; y += sourceStep ) {
            if ( source.isBelong(x, y) ) {
                for ( double phi = 0.; phi < 2 * TMath::Pi() - (TMath::Pi() / 3600.); phi += (TMath::Pi() / 1800.) ) {
                    for ( double theta = (TMath::Pi() / 2.); theta < TMath::Pi() - (TMath::Pi() / 3600.); theta += (TMath::Pi() / 1800.) ) {
                        totCount++;
                        if ( upperDisk.isBelong(x - upperDisk.coordZ * cos(phi) * tan(theta), y - upperDisk.coordZ * sin(phi) * tan(theta)) ) {
                            if ( lowerDisk.isBelong(x - lowerDisk.coordZ * cos(phi) * tan(theta), y - lowerDisk.coordZ * sin(phi) * tan(theta)) ) {
                                if ( detector.isBelong(x - detector.coordZ * cos(phi) * tan(theta), y - detector.coordZ * sin(phi) * tan(theta)) ) {
                                    nCount++;
                                }
                            }
                        }
                    }
                }
                if ( upperDisk.isBelong(x, y) ) {
                    nCount++;
                }
                totCount++;
                effAngle += static_cast<double>(nCount) / totCount;
                nCount = 0;
                totCount = 0;
                iPoint++;
            }
        }
    }
    // effAngle = static_cast<double>(nCount) / totCount;
    effAngle /= iPoint;
    return effAngle;
}
```

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.

```
#define __TCLUSTERANALYSER_HEADERS__

#include "TClusterAnalyser.h"
TClusterAnalyser::TClusterAnalyser(const TAnalyser& analyser) : TAnalyser(analyser), fBits(kNotDeleted) {
    std::clog « "TClusterAnalyser object is armed." « std::endl;
}

TClusterAnalyser::TClusterAnalyser(const TClusterAnalyser& copy) : TAnalyser(copy) {
    std::clog « "Copy TClusterAnalyser object is armed." « std::endl;
}
TClusterAnalyser::~TClusterAnalyser() {
    // for ( const auto& pair : mClustermaps ) {
    //  delete pair.second;
    // }
    // for ( const auto& pair : mClustersizes ) {
    //  delete pair.second;
    // }
    // for ( const auto& pair : mClusterDataWithShape ) {
    //  for ( const auto& pair2 : pair.second ) {
    //      for ( const TCluster* cluster : pair2.second ) {
    //          delete cluster;
    //      }
    //  }
    // }
    std::clog « "TClusterAnalyser object is terminated." « std::endl;
}

// std::vector<int> getClusterSizeRange(const CppConfigDictionary& privateProperty) {
//  std::vector<int> clusterSizeRange;
//  if ( privateProperty.hasKey("interest_size") ) {
//      for ( const std::string& rangeStr : privateProperty.getSubConfig("interest_size").getValueList() ) {
//          if ( rangeStr.find('.') != std::string::npos ) {
//              for ( int i = stoi(rangeStr.substr(0, rangeStr.find('.'))); i < stoi(rangeStr.substr(rangeStr.find('.') + 3)) + 1; i++ ) {
//                  clusterSizeRange.push_back(i);
//              }
//          } else {
//              clusterSizeRange.push_back(stoi(rangeStr));
//          }
//      }
//  } else {
//      for ( int i = 0; i < 100; i++ ) {
//          clusterSizeRange.push_back(i);
//      }
//  }
//  return clusterSizeRange;
// }

// /**
//  * @brief Generalized function to draw clustermap
//  *
//  * @param config Draw configuration for map title, directory and filename
//  * @param clusters Dataset to draw
//  * @return const TH2*
// */
// TH2D* TClusterAnalyser::getClusterPlot(const CppConfigDictionary& config, const std::vector<TCluster*>& clusters) {
//  // Static variable for numbering
//  static int iMap = 0;
//  // Allocate a 2d histogram.
//  std::string plotTitle = "";
//  if ( config.hasKey("title") ) {
//      plotTitle += config.find("title");
//  }
//  if ( config.hasKey("x_title") ) {
//      plotTitle += "; " + config.find("x_title");
//  } else {
//      plotTitle += "; ";
//  }
//  if ( config.hasKey("y_title") ) {
//      plotTitle += "; " + config.find("y_title");
//  } else {
//      plotTitle += "; ";
//  }

//  TH2D* map = new TH2D(Form("map%d", iMap), static_cast<TString>(plotTitle), 1024, 0, 1024, 512, 0, 512);
//  // Fill data
//  std::vector<int> interestSizeSet = getClusterSizeRange(config);
//  for ( const TCluster* cluster : clusters ) {
```

```
//      if ( !interestSizeSet.empty() ) {
//          if ( std::find(interestSizeSet.begin(), interestSizeSet.end(), cluster->getSize()) != interestSizeSet.end() ) {
//              map->Fill(cluster->getCenter().first, cluster->getCenter().second);

//          }
//      } else {
//          map->Fill(cluster->getCenter().first, cluster->getCenter().second);
//      }
//  }
//  // Canvas setting
//  TCanvas* canvas = new TCanvas(Form("mapCan%d", iMap), "", 2500, 1000);
//  canvas->SetMargin(.07, .35, .12, .08);
//  map->GetXaxis()->SetTitleOffset(1.4);
//  map->GetXaxis()->SetLabelOffset(0.003);
//  // Find directory for saving clustermap. If it doesn't exist, then make the directory with mother directories.
//  std::filesystem::path filePath(config.find("output_path"));
//  filePath /= config.find("subdirectory");
//  std::filesystem::create_directories(filePath);

//  // Draw plot with options
//  if ( config.hasKey("options") ) {
//      for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
//          map->Draw(static_cast<TString>(optionName));
//          mExpSettingLegend->Draw("SAME");
//          std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
//          if ( config.hasKey("extension") ) {
//              file.replace_extension(config.find("extension"));
//          } else {
//              file.replace_extension("png");
//          }
//          canvas->SaveAs(static_cast<TString>(file));
//      }
//  } else {
//      map->Draw();
//      mExpSettingLegend->Draw("SAME");
//      std::filesystem::path file = filePath / (config.find("filename"));
//      if ( config.hasKey("extension") ) {
//          file.replace_extension(config.find("extension"));
//      } else {
//          file.replace_extension("png");
//      }
//      canvas->SaveAs(static_cast<TString>(file));
//  }
//  // Delete canvas;
//  delete canvas;
//  iMap++;
//  return map;
// }
// /**
//  * @brief
//  *
//  * @param config
//  * @param clusters
//  * @return TH1D*
// */
// TH1D* TClusterAnalyser::getClustersizePlot(const CppConfigDictionary& config, const std::vector<TCluster*>& clusters) {
//  static int iDistribution = 0;
//  TString distName = Form("distribution%d", iDistribution);
//  TString distTitle = "";
//  distTitle += config.hasKey("title") ? config.find("title") : "";
//  distTitle += config.hasKey("x_title") ? "; " + config.find("x_title") : ";";
//  distTitle += config.hasKey("y_title") ? "; " + config.find("y_title") : ";";
//  Int_t nBins = 0;
//  Float_t maxBin = 0;
//  Float_t minBin = 0;
//  if ( config.hasKey("distribution_info") ) {
//      nBins = config.getSubConfig("distribution_info").hasKey("nbins") ? stoi(config.getSubConfig("distribution_info").find("nbins")) : 80;
//      maxBin = config.getSubConfig("distribution_info").hasKey("max") ? stoi(config.getSubConfig("distribution_info").find("max")) + .5 : 80.5;
//      minBin = config.getSubConfig("distribution_info").hasKey("min") ? stoi(config.getSubConfig("distribution_info").find("min")) - .5 : .5;
//  } else {
//      nBins = 80;
//      maxBin = 80.5;
//      minBin = 0.5;
//  }
//  TH1D* distribution = new TH1D(distName, distTitle, nBins, minBin, maxBin);
//  for ( const TCluster* cluster : clusters ) {
//      distribution->Fill(cluster->getSize()); // Fill clustersize to clustersize distribution
//  }
//  TCanvas* canvas = new TCanvas(Form("distributionCan%d", iDistribution), "", 2500, 1000);
//  canvas->SetMargin(.07, .28, .12, .08);
//  distribution->GetXaxis()->SetTitleOffset(1.4);
```

```
//  distribution->GetXaxis()->SetLabelOffset(0.003);

//  std::filesystem::path filePath(config.find("output_path"));
//  filePath /= config.find("subdirectory");
//  std::filesystem::create_directories(filePath);

//  if ( config.hasKey("options") ) {
//      for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
//          if ( optionName == "logy" ) {
//              distribution->Draw();
//              mExpSettingLegend->Draw("SAME");
//              canvas->SetLogy();
//              std::filesystem::path file = filePath / (config.find("filename") + "_" + optionName);
//              if ( config.hasKey("extension") ) {
//                  file.replace_extension(config.find("extension"));
//              } else {
//                  file.replace_extension("png");
//              }
//              canvas->SaveAs(static_cast<TString>(file));
//          } else if ( optionName == "basic" ) {
//              distribution->Draw();
//              mExpSettingLegend->Draw("SAME");
//              std::filesystem::path file = filePath / config.find("filename");
//              if ( config.hasKey("extension") ) {
//                  file.replace_extension(config.find("extension"));
//              } else {
//                  file.replace_extension("png");
//              }
//              canvas->SetLogy(0);
//              canvas->SaveAs(static_cast<TString>(file));
//          }
//      }
//  } else {
//      distribution->Draw();
//      mExpSettingLegend->Draw("SAME");
//      std::filesystem::path file = filePath / (config.find("filename"));
//      if ( config.hasKey("extension") ) {
//          file.replace_extension(config.find("extension"));
//      } else {
//          file.replace_extension("png");
//      }
//      canvas->SaveAs(static_cast<TString>(file));
//  }
//  delete canvas;
//  iDistribution++;
//  return distribution;
// }

// void TClusterAnalyser::saveClustermap(std::string typeName, const CppConfigDictionary& config) {
//  std::clog « "Generating \033[1;32mClustermap\033[1;0m..." « std::endl;
//  if ( !mClustermaps.count(typeName) ) {
//      mClustermaps.insert_or_assign(typeName, getClusterPlot(config, mExpData.find(typeName)->second->getClusters()));
//      if ( mIsOutputGraph ) {
//          mDirectorySet.find(std::string(typeName))->second->cd();
//          mClustermaps.find(std::string(typeName))->second->Write("clustermap");
//          mOutputFile->cd();
//      }
//  } else {
//      getClusterPlot(config, mExpData.find(typeName)->second->getClusters());
//  }
// }

// void TClusterAnalyser::saveClustersize(std::string typeName, const CppConfigDictionary& config) {
//  std::clog « "Generating \033[1;32mCluster Size Distribution\033[1;0m..." « std::endl;
//  if ( !mClustersizes.count(typeName) ) {
//      mClustersizes.insert_or_assign(typeName, getClustersizePlot(config, mExpData.find(typeName)->second->getClusters()));
//      if ( mIsOutputGraph ) {
//          mDirectorySet.find(std::string(typeName))->second->cd();
//          mClustersizes.find(std::string(typeName))->second->Write("clustersize");
//          mOutputFile->cd();
//      }
//  } else {
//      getClustersizePlot(config, mExpData.find(typeName)->second->getClusters());
//  }
// }

// void TClusterAnalyser::setClusterDataWithShape(const std::vector<int>& clusterSizeRange) {
//  for ( const int clusterSize : clusterSizeRange ) {
//      std::vector<TCluster*> clustersWithShape;
//  }
// }
```

```
// void TClusterAnalyser::saveHitmapByClustersize(const CppConfigDictionary& config) {
//  // std::filesystem::create_directories(mSavePath / "hitmap_by_cluster_size");
//  // for ( int clusterSize = 1; clusterSize < 80; clusterSize++ ) {
//  //  TH2D* clusterHitmap = new TH2D(Form("hitmap%d", clusterSize), Form("Hitmap of cluster size %d", clusterSize), 1024, 0, 1024, 512, 0, 512);
//  //  TH2D* clusterClustermap = new TH2D(Form("clustermap%d", clusterSize), Form("Clustermap of cluster size %d", clusterSize), 1024, 0, 1024, 512, 0, 512);
//  //  TH1D* clusterBinFire = new TH1D(Form("fired%d", clusterSize), Form("The number of fired of each bins in cluster size %d", clusterSize), 50, 0, 50);
//  //  // for ( const TCluster* cluster : mExpData->getClusters() ) {
//  //  //  if ( cluster->getSize() == clusterSize ) {
//  //  //      for ( const std::pair<int, int>& pixel : cluster->getPixels() ) {
//  //  //          clusterHitmap->Fill(pixel.first, pixel.second);
//  //  //      }
//  //  //      clusterClustermap->Fill(cluster->getCenter().first, cluster->getCenter().second);
//  //  //  }
//  //  // }
//  //  for ( int iRow = 0; iRow < 1024; iRow++ ) {
//  //      for ( int iColumn = 0; iColumn < 512; iColumn++ ) {
//  //          if ( clusterHitmap->GetBinContent(iRow, iColumn) != 0 ) {
//  //              clusterBinFire->Fill(clusterHitmap->GetBinContent(iRow, iColumn));
//  //          }
//  //      }
//  //  }
//  //  if ( clusterBinFire->GetEntries() != 0 ) {
//  //      TCanvas* hcanvas = new TCanvas(Form("hcan%d", clusterSize), "", 2000, 1000);
//  //      clusterHitmap->Draw();
//  //      // hcanvas->SaveAs(static_cast<TString>(mSavePath / "hitmap_by_cluster_size" / ("hitmap_cs_" + std::to_string(clusterSize) + ".png")));
//  //      TCanvas* ccanvas = new TCanvas(Form("ccan%d", clusterSize), "", 2000, 1000);
//  //      clusterClustermap->Draw();
//  //      // ccanvas->SaveAs(static_cast<TString>(mSavePath / "hitmap_by_cluster_size" / ("clustermap_cs_" + std::to_string(clusterSize) + ".png")));
//  //      TCanvas* dcanvas = new TCanvas(Form("dcan%d", clusterSize), "", 1000, 1000);
//  //      clusterBinFire->Draw();
//  //      dcanvas->SetLogy();
//  //      // dcanvas->SaveAs(static_cast<TString>(mSavePath / "hitmap_by_cluster_size" / ("fire_distribution_cs_" + std::to_string(clusterSize) + ".png")));

//  //  }
//  // }
// }
```

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 (yochoi@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.

```
#ifndef __TCLUSTERANALYSER__
#define __TCLUSTERANALYSER__

#ifdef __TCLUSTERANALYSER_HEADERS__
#include <iostream>

#include "TFile.h"
#include "TDirectory.h"
#include "TCanvas.h"
#include "TPaveText.h"
#include "TH1D.h"
#include "TH2D.h"

#include "CppConfigFile.h"
#include "cpptqdm.h"

#include "TCluster.h"
#include "TExperimentData.h"
#endif

#include <vector>

#include "TAnalyser.h"

class TH1D;
class TH2D;

class Configurable;
class TCluster;

class TClusterAnalyser : public TAnalyser {
protected:
    // 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;
public:
    //Constructor
    TClusterAnalyser() = default;
    TClusterAnalyser(const TAnalyser& analyser);
    TClusterAnalyser(const TClusterAnalyser& copy);
    ~TClusterAnalyser();

    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);

private:
    unsigned int fBits;
public:
```

```
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
};

#endif
```

8.192 /home/ychoi/ATOM/trashcan/TClusterShapeAnalyser.cpp File Reference

Tools for analysising 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 analysising shape property of cluster.

Author

Yongjun Choi (yochoi@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.
```

```
            const std::vector<TCluster*> divideData = mDivideData.find(std::string(typeName))->second->getClusterOfSize(clusterSize);
            // This inserts cluster size information and clusters to TClusterShape objects.
            TClusterShape* clusterShape = new TClusterShape(clusterSize, divideData);
            // This extracts shape informations.
            clusterShape->identifyShapes();
            // This sorts shapes according to its size of long axis.
            clusterShape->sortShapes();
            //
            clusterShapes.push_back(clusterShape);
            nTotalShape += clusterShape->getClusterShapeInfos().size();
            maxMode = std::max(maxMode, static_cast<int>(clusterShape->getClusterShapeInfos().size()));
        }
    }
    mClusterShapeSet.insert_or_assign(std::string(typeName), clusterShapes);
    mNTotalShapeSet.insert_or_assign(std::string(typeName), nTotalShape);
    mMaxModeSet.insert_or_assign(std::string(typeName), maxMode);
}
void TClusterShapeAnalyser::saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config) {
    // Print log message
    std::clog « "Generating " « "\033[1;32m" « "individual shapes" « "\033[1;0m" « "..." « std::endl;

    // Get the total number of shapes for progress bar.
    int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;

    // Get save path from config file.
    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    std::filesystem::create_directories(filePath);

    // Init progress bar
    ProgressBar pBar(nTotalShape);

    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize(); // The cluster size of shape set.
        int iClusterShape = 0; // The i-th cluster shape
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress(); // Print progress bar.

            int shapeWidth = shapeInfo.mClusterMatrix->getNRow(); // Shape width of matrix
            int shapeHeight = shapeInfo.mClusterMatrix->getNColumn(); // Shape height of matrix

            // Initialize canvas
            TCanvas* canvas = new TCanvas(Form("%d_%d.png", clusterSize, iClusterShape), "", (shapeWidth + 2) * 500, (shapeHeight + 2) * 500);
            // Set canvas margin
            canvas->SetMargin(0., 0., 0., 0.);
            // Draw shape on canvas
            shapeInfo.mClusterMap->Draw();
            // Fill colour
            shapeInfo.mClusterMap->SetDrawOption("COLZ");
            // Remove histogram axis
            shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);

            // Fill line for seperating pixels
            TLine* line = new TLine();
            line->SetLineColorAlpha(kRed, 6. / 8);
            for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
                for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
                    if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
                        double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
                        double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
                        double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
                        double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
                        line->DrawLine(xlow, ylow, xup, ylow); // Bottom
                        line->DrawLine(xlow, yup, xup, yup); // Top
                        line->DrawLine(xlow, ylow, xlow, yup); // Left
                        line->DrawLine(xup, ylow, xup, yup); // Right
                    }
                }
            }

            // Set title of shape
            TText* title = new TText(.5, 1. - .5 / (shapeHeight + 2), Form("%d-th Cluster Shape of Cluster Size %d", iClusterShape, clusterSize));
            title->SetTextAlign(23);
            title->SetNDC();
            title->Draw();

            // Save shape figure
            std::filesystem::path file = filePath / (config.find("filename") + "_" + std::to_string(clusterSize) + "_" + std::to_string(iClusterShape));
            if ( config.hasKey("extension") ) {
                file.replace_extension(config.find("extension"));
```

```
            } else {
                file.replace_extension("png");
            }
            canvas->SaveAs(static_cast<TString>(file));

            iClusterShape++;

            delete title;
            title = nullptr;
            delete line;
            line = nullptr;
            delete canvas;
            canvas = nullptr;
        }
    }
}

int calNIncludePixel(const TMatrix2D<int>* matrix) {
    int centreX = 0, centreY = 0;
    int clusterSize = 0;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            if ( matrix->getElement(pixelX, pixelY) == 1 ) {
                centreX += pixelX * 2;
                centreY += pixelY * 2;
                clusterSize++;
            }
        }
    }
    int radiusSquare = 0;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            if ( matrix->getElement(pixelX, pixelY) == 1 ) {
                int distance = pow(abs(2 * pixelX * clusterSize - centreX) + clusterSize, 2) + pow(abs(2 * pixelY * clusterSize - centreY) + clusterSize, 2);
                radiusSquare = std::max(radiusSquare, distance);
            }
        }
    }
    int count = 0;
    for ( int x = floor((centreX - sqrt(radiusSquare)) / (2 * clusterSize)); x < ceil((centreX + sqrt(radiusSquare)) / (2 * clusterSize)) + 1; x++ ) {
        for ( int y = floor((centreY - sqrt(radiusSquare)) / (2 * clusterSize)); y < ceil((centreY + sqrt(radiusSquare)) / (2 * clusterSize)) + 1; y++ ) {
            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 ) {
                count++;
            }
        }
    }
    return count;
}

double calRatioOfRadius(const TMatrix2D<int>* matrix) {
    int centreX = 0, centreY = 0;
    int clusterSize = 0;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            if ( matrix->getElement(pixelX, pixelY) == 1 ) {
                centreX += pixelX * 2;
                centreY += pixelY * 2;
                clusterSize++;
            }
        }
    }
    int longRadiusSquare = 0;
    int shortRadiusSquare = 100000000;
    for ( int pixelX = 0; pixelX < matrix->getNRow(); pixelX++ ) {
        for ( int pixelY = 0; pixelY < matrix->getNColumn(); pixelY++ ) {
            bool hasValue = matrix->getElement(pixelX, pixelY) == 1;
            bool isWorldBorder = pixelX == 0 || pixelY == 0 || pixelX == matrix->getNRow() - 1 || pixelY == matrix->getNColumn() - 1;
            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;
            if ( hasValue ) {
                int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY * clusterSize - centreY, 2);
                longRadiusSquare = std::max(longRadiusSquare, distance);
            }
            if ( (hasValue && isWorldBorder) || (hasValue && !isWorldBorder && isBorder) ) {
                int distance = pow(2 * pixelX * clusterSize - centreX, 2) + pow(2 * pixelY * clusterSize - centreY, 2);
                shortRadiusSquare = std::min(shortRadiusSquare, distance);
            }
        }
    }
```

```
    if ( matrix->getNRow() == 1 || matrix->getNColumn() == 1 ) {
        shortRadiusSquare = 0;
    }
    return sqrt(static_cast<double>(shortRadiusSquare) / longRadiusSquare);
}

void TClusterShapeAnalyser::saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config) {
    // Print out a log
    std::cout « "Generating " « "\033[1;32m" « "Informations of shapes with same size" « "\033[1;0m" « "..." « std::endl;

    // Settting output file path
    std::filesystem::path filePath(config.find("output_path"));
    filePath /= config.find("subdirectory");
    // The creation of directories of output path
    std::filesystem::create_directories(filePath);
    // Call configs. First element is config name and second one is config values.
    std::unordered_map<std::string, CppConfigDictionary> plotConfigList = config.getSubConfig("ratio_distribution").getSubConfigSetWithName();
    // Call histograms
    std::unordered_map<std::string, TH1D*> distributionSet;
    // Set the information of histograms from config file
    for ( const auto& plotConfig : plotConfigList ) {
        // Plot name is config name. It isn't value of "name" key.
        std::string plotName = plotConfig.second.find("name");
        // Set plot title and x, y label.
        TString plotTitle = plotConfig.second.hasKey("title") ? plotConfig.second.find("title") : "";
        TString plotXTitle = plotConfig.second.hasKey("x_title") ? plotConfig.second.find("x_title") : "";
        TString plotYTitle = plotConfig.second.hasKey("y_title") ? plotConfig.second.find("y_title") : "";
        // Set number of bin and min and max of x-direction range.
        Int_t plotNBin = plotConfig.second.hasKey("n_bin") ? stoi(plotConfig.second.find("n_bin")) : 100;
        Float_t plotXMin = plotConfig.second.hasKey("x_min") ? stof(plotConfig.second.find("x_min")) : 0.;
        Float_t plotXMax = plotConfig.second.hasKey("x_max") ? stof(plotConfig.second.find("x_max")) : 1.;

        // Add histograms to map.
        distributionSet.insert_or_assign(plotConfig.first, new TH1D(static_cast<TString>(plotName), plotTitle + "; " + plotXTitle + "; " + plotYTitle, plotNBin, plotXMin, plotXMax));
    }

    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        // Get cluster size.
        int clusterSize = clusterShape->getClusterSize();
        if ( clusterSize < stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_min")) || clusterSize > stoi(config.getSubConfig("ratio_distribution").find("cluster_size_oi_max")) ) {
            continue;
        }
        // Initialize canvas.
        TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
        // int binNum = clusterShape->getClusterShapeInfos().size();
        TGraph* areaRatioGraph = new TGraph();
        TGraph* radiusRatioGraph = new TGraph();

        int iShape = 0;
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            // pBar.printProgress();
            for ( const auto& plotConfig : plotConfigList ) {
                if ( plotConfig.second.find("name") == "area_ratio" ) {
distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix));
                }
                if ( plotConfig.second.find("name") == "area_ratio_with_entry" ) {
distributionSet.find(plotConfig.first)->second->Fill(static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix), shapeInfo.mEntry);
                }
                if ( plotConfig.second.find("name") == "radius_ratio" ) {
distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix));
                }
                if ( plotConfig.second.find("name") == "radius_ratio_with_entry" ) {
distributionSet.find(plotConfig.first)->second->Fill(calRatioOfRadius(shapeInfo.mClusterMatrix), shapeInfo.mEntry);
                }
            }

            areaRatioGraph->SetPoint(iShape, iShape, static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix));
            radiusRatioGraph->SetPoint(iShape, iShape, calRatioOfRadius(shapeInfo.mClusterMatrix));
```

```
            // std::cout « clusterSize « "\t" « iShape « "\t" « static_cast<double>(clusterSize) / calNIncludePixel(shapeInfo.mClusterMatrix) « "\t" « calRatioOfRadius(shapeInfo.mClusterMatrix) « "\t" « shapeInfo.mEntry « std::endl;
            iShape++;
        }
        TF1* line1 = new TF1("line1", "0.8", 0, iShape);
        TF1* line2 = new TF1("line2", "0.5", 0, iShape);

        areaRatioGraph->SetTitle(Form("Informations for cluster shapes in cluster size %d", clusterSize));
        areaRatioGraph->GetXaxis()->SetTitle("i-th cluster shape");
        areaRatioGraph->GetXaxis()->SetTitleOffset(1.4);
        areaRatioGraph->GetXaxis()->SetLabelOffset(0.003);
        areaRatioGraph->GetYaxis()->SetTitle("Ratio");
        areaRatioGraph->GetYaxis()->SetTitleOffset(0.7);
        areaRatioGraph->SetMarkerStyle(45);
        areaRatioGraph->SetMarkerSize(4);
        areaRatioGraph->SetMarkerColor(kRed);
        areaRatioGraph->SetMaximum(1);
        areaRatioGraph->SetMinimum(0);
        areaRatioGraph->Draw("AP");
        radiusRatioGraph->SetMarkerStyle(41);
        radiusRatioGraph->SetMarkerSize(4);
        radiusRatioGraph->SetMarkerColor(kBlue);
        radiusRatioGraph->Draw("P");
        mExpSettingLegend->Draw("SAME");
        line1->SetLineColor(kPink + 2);
        // line1->Draw("SAME");
        line2->SetLineColor(kCyan - 7);
        // line2->Draw("SAME");
        TLegend* legend = new TLegend(.78, .7, .98, .92);
        legend->SetHeader("Ratios", "c");
        legend->AddEntry(areaRatioGraph, "Pixels / Full pixels", "p");
        legend->AddEntry(radiusRatioGraph, "Short radius / Long radius", "p");
        legend->Draw();
        canvas->SetMargin(.07, .28, .12, .08);
        std::filesystem::path file = filePath;

        file /= config.getSubConfig("graphs").find("filename") + "_" + std::to_string(clusterSize);
        file.replace_extension(config.find("extension"));

        canvas->SaveAs(static_cast<TString>(file));

        delete line1;
        delete line2;
        delete areaRatioGraph;
        delete radiusRatioGraph;
        delete canvas;
    }

    for ( const auto& plotConfig : plotConfigList ) {
        TString canvasName = "can" + plotConfig.first;
        Int_t canvasWidth = plotConfig.second.hasKey("canvas_width") ? stoi(plotConfig.second.find("canvas_width")) : 500;
        Int_t canvasHeight = plotConfig.second.hasKey("canvas_height") ? stoi(plotConfig.second.find("canvas_height")) : 500;
        TCanvas* canvas = new TCanvas(canvasName, "", canvasWidth, canvasHeight);

        distributionSet.find(plotConfig.first)->second->SetStats(0);

        Int_t y_min = plotConfig.second.hasKey("y_min") ? stoi(plotConfig.second.find("y_min")) : 0;
        Int_t y_max = plotConfig.second.hasKey("y_max") ? stoi(plotConfig.second.find("y_max")) : 1000;
        distributionSet.find(plotConfig.first)->second->SetMinimum(y_min);
        distributionSet.find(plotConfig.first)->second->SetMaximum(y_max);
        distributionSet.find(plotConfig.first)->second->Draw();

        std::filesystem::path file = filePath;
        file /= plotConfig.second.find("filename");
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));

        delete canvas;
        canvas = nullptr;
    }
}

void TClusterShapeAnalyser::saveSameSizeShapes(std::string_view typeName, const CppConfigDictionary config) {
    std::cout « "Generating " « "\033[1;32m" « "Shapes with same sized" « "\033[1;0m" « "..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);
```

```

    int nTotalShape = mNTotalShapeSet.find(std::string(typeName))->second;

    const int nominalWidth = 200;
    const int nominalHeader = 100;
    const int nPlotInRow = 10;
    int plotsWidth = nPlotInRow * nominalWidth;

    ProgressBar pBar(nTotalShape);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        int nClusterShape = clusterShape->getClusterShapeInfos().size();

        int plotsHeight = (floor((nClusterShape - 1.) / 10.) + 1.) * nominalWidth;

        TCanvas* canvas = new TCanvas(Form("shapes%d", clusterSize), "", plotsWidth, plotsHeight +
nominalHeader);

        int iClusterShape = 0;

        std::vector<TPad*> padSet;
        std::vector<TLine*> lineSet;
        std::vector<TText*> textSet;

        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();

            int width = shapeInfo.mClusterMatrix->getNRow() + 2;
            int height = shapeInfo.mClusterMatrix->getNColumn() + 2;

            double padCenterX = (1. / nPlotInRow) * (.5 + (iClusterShape % 10));
            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.);

            double padWidth = (1. / (2. * nPlotInRow));
            double padHeight = (static_cast<double>(plotsHeight) / (plotsHeight + nominalHeader)) /
(2. * ((floor(static_cast<double>(nClusterShape) / nPlotInRow) + 1.)));

            TPad* pad = new TPad(Form("pad%d", iClusterShape), "", padCenterX - padWidth, padCenterY -
padHeight, padCenterX + padWidth, padCenterY + padHeight);
            pad->SetMargin(0., 0., .5 * (1. - static_cast<double>(height) / width), .5 * (1. -
static_cast<double>(height) / width));
            pad->Draw();
            pad->cd();
            shapeInfo.mClusterMap->SetDrawOption("COL");
            shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->Draw("col");
            pad->SetFrameLineWidth(0);
            padSet.push_back(pad);

            TLine* line = new TLine();
            line->SetLineColorAlpha(kRed, 6. / 8);
            for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
                for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
                    if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
                        double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
                        double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
                        double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
                        double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
                        line->DrawLine(xlow, ylow, xup, ylow); // Bottom
                        line->DrawLine(xlow, yup, xup, yup); // Top
                        line->DrawLine(xlow, ylow, xlow, yup); // Left
                        line->DrawLine(xup, ylow, xup, yup); // Right
                    }
                }
            }
            lineSet.push_back(line);

            TText* numberingText = new TText(padCenterX, padCenterY, Form("%d", iClusterShape));
            numberingText->SetNDC();
            numberingText->SetTextAlign(22);
            numberingText->SetTextSize(.3 * nominalWidth / (plotsHeight + nominalHeader));
            numberingText->SetTextColorAlpha(kBlack, 5. / 8.);

            canvas->cd();
            numberingText->Draw();
            textSet.push_back(numberingText);
            iClusterShape++;
        }
        TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight),
Form("Cluster shapes with cluster size %d", clusterSize));
        titleText->SetTextSize(.5 * nominalHeader / (plotsHeight + nominalHeader));
        titleText->SetTextAlign(22);
        titleText->SetNDC();
```

```
        titleText->Draw();
        std::filesystem::path file = filePath / (config.find("filename") + "_" + std::to_string(clusterSize));
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));

        delete titleText;
        titleText = nullptr;

        for ( TPad* pad : padSet ) {
            delete pad;
            pad = nullptr;
        }

        for ( TLine* line : lineSet ) {
            delete line;
            line = nullptr;
        }

        for ( TText* text : textSet ) {
            delete text;
            text = nullptr;
        }

        delete canvas;
        canvas = nullptr;
    }
}

void TClusterShapeAnalyser::saveTotalShapes(std::string_view typeName, const CppConfigDictionary config) {
    // Print log
    std::cout « "Generating \033[1;32mTotal shapes\033[1;0m..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    int nWidth = mMaxModeSet.find(std::string(typeName))->second;
    int nHeight = mClusterShapeSet.find(std::string(typeName))->second.size();

    int nominalWidth = 100;
    int nominalHeader = 100;

    int plotsWidth = nWidth * nominalWidth;
    int plotsHeight = nHeight * nominalWidth;
    TCanvas* canvas = new TCanvas("tShape", "cluster shape", plotsWidth + nominalHeader, plotsHeight + nominalHeader);
    int iClusterSize = 0;
    std::vector<TPad*> padSet;
    std::vector<TLine*> lineSet;
    std::vector<TText*> textSet;

    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        int iClusterShape = 0;
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();
            int width = shapeInfo.mClusterMap->GetNbinsX();
            int height = shapeInfo.mClusterMap->GetNbinsY();

            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);
            padSet.push_back(pad);
            pad->Draw();
            pad->cd();
            pad->SetMargin(0., 0., .5 * (1. - (double) height / width), .5 * (1. - (double) height / width));

            shapeInfo.mClusterMap->Draw();
            shapeInfo.mClusterMap->SetDrawOption("COL");
            shapeInfo.mClusterMap->GetXaxis()->SetAxisColor(0);
            shapeInfo.mClusterMap->GetYaxis()->SetAxisColor(0);
            TLine* line = new TLine();
            lineSet.push_back(line);
            line->SetLineColorAlpha(kRed, 6. / 8);
            for ( int i = 1; i <= shapeInfo.mClusterMap->GetNbinsX(); ++i ) {
                for ( int j = 1; j <= shapeInfo.mClusterMap->GetNbinsY(); ++j ) {
                    if ( shapeInfo.mClusterMap->GetBinContent(i, j) > 0 ) {
```

```
                        double xlow = shapeInfo.mClusterMap->GetXaxis()->GetBinLowEdge(i);
                        double xup = shapeInfo.mClusterMap->GetXaxis()->GetBinUpEdge(i);
                        double ylow = shapeInfo.mClusterMap->GetYaxis()->GetBinLowEdge(j);
                        double yup = shapeInfo.mClusterMap->GetYaxis()->GetBinUpEdge(j);
                        line->DrawLine(xlow, ylow, xup, ylow); // Bottom
                        line->DrawLine(xlow, yup, xup, yup); // Top
                        line->DrawLine(xlow, ylow, xlow, yup); // Left
                        line->DrawLine(xup, ylow, xup, yup); // Right
                    }
                }
            }
            pad->SetFrameLineWidth(0);
            canvas->cd();
            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));
            textSet.push_back(numberingText);
            numberingText->SetNDC();
            numberingText->SetTextAlign(22);
            numberingText->SetTextSize(.4 * nominalWidth / (plotsHeight + nominalHeader));
            numberingText->SetTextColor(kBlack);
            numberingText->Draw();
            iClusterShape++;
        }
        TText* sizeText = new TText((double) nominalHeader * .5 / (nominalHeader + plotsWidth), ((double) plotsHeight / (plotsHeight + nominalHeader)) * (((double) nHeight - iClusterSize - .5) / nHeight), Form("%d", clusterSize));
        textSet.push_back(sizeText);
        sizeText->SetNDC();
        sizeText->SetTextAlign(22);
        sizeText->SetTextSize(.6 * nominalWidth / (plotsHeight + nominalHeader));
        sizeText->SetTextColor(kBlack);
        sizeText->Draw();
        iClusterSize++;
    }
    TText* titleText = new TText(.5, 1. - .5 * nominalHeader / (nominalHeader + plotsHeight), static_cast<TString>("Total Cluster Shapes"));
    titleText->SetTextSize(.8 * nominalHeader / (plotsHeight + nominalHeader));
    titleText->SetTextAlign(22);
    titleText->SetNDC();
    titleText->Draw();

    std::filesystem::path file = filePath / config.find("filename");
    file.replace_extension(config.find("extension"));
    canvas->SaveAs(static_cast<TString>(file));

    delete titleText;
    titleText = nullptr;

    for ( TPad* pad : padSet ) {
        delete pad;
        pad = nullptr;
    }

    for ( TLine* line : lineSet ) {
        delete line;
        line = nullptr;
    }

    for ( TText* text : textSet ) {
        delete text;
        text = nullptr;
    }

    delete canvas;
    canvas = nullptr;
}

void TClusterShapeAnalyser::saveSameSizeShapeEntry(std::string_view typeName, const CppConfigDictionary config) {
    std::cout « "Generating \033[1;32mEntry of shapes with same size\033[1;0m..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        TCanvas* canvas = new TCanvas(Form("shapeEntry%d", clusterSize), "shapeEntry", 2500, 1000);
        int binNum = clusterShape->getClusterShapeInfos().size();
        TH1D* distribution = new TH1D(Form("shapeEntry%d", clusterSize), Form(static_cast<TString>("Shape Entry with cluster size %d"), clusterSize), binNum, -.5, binNum - .5);
```

```
        int iShape = 0;
        for ( const TShapeInfo& shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();
            distribution->Fill(iShape, shapeInfo.mEntry);
            iShape++;
        }
        distribution->GetXaxis()->SetNdivisions(10, 10, 0, true);
        distribution->GetXaxis()->SetTitle("i-th cluster size");
        distribution->GetXaxis()->SetTitleOffset(1.4);
        distribution->GetXaxis()->SetLabelOffset(0.003);
        distribution->GetYaxis()->SetTitle("Entry");
        distribution->GetYaxis()->SetTitleOffset(0.7);
        distribution->Draw("HIST");
        mExpSettingLegend->Draw("SAME");
        canvas->SetMargin(.07, .28, .12, .08);

        std::filesystem::path file = filePath / (config.find("filename") + "_" + std::to_string(clusterSize));
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));

        delete distribution;
        delete canvas;
    }
}

void TClusterShapeAnalyser::saveTotalShapeEntry(std::string_view typeName, const CppConfigDictionary config) {
    std::cout « "Generating \033[1;32mEntry of total shapes\033[1;0m..." « std::endl;

    std::filesystem::path filePath(config.find("output_path"));
    if ( config.hasKey("subdirectory") ) {
        filePath /= config.find("subdirectory");
    }
    std::filesystem::create_directories(filePath);

    int nXbin = 0;
    int nYbin = 0;
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        nYbin = std::max(nYbin, static_cast<int>(clusterShape->getClusterShapeInfos().size()));
        nXbin = std::max(nXbin, clusterShape->getClusterSize());
    }
    TCanvas* canvas = new TCanvas("shapeEntryTotal", "shape entry", 2500, 1000);
    TH2D* distribution = new TH2D("shapeEntryTotal", static_cast<TString>(config.find("plot_titles")), nXbin, -nXbin - .5, -.5, nYbin, -nYbin + .5, .5);
    ProgressBar pBar(mNTotalShapeSet.find(std::string(typeName))->second);
    for ( const TClusterShape* clusterShape : mClusterShapeSet.find(std::string(typeName))->second ) {
        int clusterSize = clusterShape->getClusterSize();
        int iShape = 0;
        for ( const TShapeInfo shapeInfo : clusterShape->getClusterShapeInfos() ) {
            pBar.printProgress();
            distribution->Fill(-clusterSize, -iShape, shapeInfo.mEntry);
            iShape++;
        }
    }
    for ( int iXbin = 1; iXbin < distribution->GetNbinsX(); ++iXbin ) {
        distribution->GetXaxis()->SetBinLabel(iXbin, Form("%g", floor(-1 * distribution->GetXaxis()->GetBinLowEdge(iXbin))));
    }

    for ( int iYbin = 1; iYbin < distribution->GetNbinsY(); ++(++iYbin) ) {
        distribution->GetYaxis()->SetBinLabel(iYbin, Form("%g", floor(-1 * distribution->GetYaxis()->GetBinLowEdge(iYbin))));
    }

    distribution->SetStats(0);
    if ( config.hasKey("options") ) {
        for ( const std::string& optionName : config.getSubConfig("options").getValueList() ) {
            distribution->Draw(static_cast<TString>(optionName));
            canvas->SetPhi(10);
            canvas->SetTheta(25);
            canvas->SetLogz();
            std::filesystem::path file = filePath;
            file /= (config.find("filename") + "_" + optionName);
            file.replace_extension(config.find("extension"));
            canvas->SaveAs(static_cast<TString>(file));
        }
    } else {
        distribution->Draw();
        std::filesystem::path file = filePath;
        file /= config.find("filename");
        file.replace_extension(config.find("extension"));
        canvas->SaveAs(static_cast<TString>(file));
    }

    delete distribution;
    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 (yochoi@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.

```cpp

#ifndef __TCLUSTERSHAPEANALYSER__
#define __TCLUSTERSHAPEANALYSER__

#ifdef __TCLUSTERSHAPEANALYSER_HEADER__
#include <iostream>

#include "TCanvas.h"
#include "TH2I.h"
#include "TText.h"
#include "TLine.h"
#include "TPaveText.h"
#include "TGraph.h"
#include "TF1.h"
#include "TLegend.h"

#include "cpptqdm.h"
#include "CppConfigFile.h"

#include "TClusterShape.h"
#include "TCluster.h"
#include "TClusterDivideData.h"
#include "TExperimentData.h"
#include "TMatrix2D.h"
#endif

#include <vector>
#include <unordered_map>

#include "TClusterAnalyser.h"

class TClusterShape;

class TClusterShapeAnalyser : protected TClusterAnalyser {
private:
    std::unordered_map<std::string, std::vector<TClusterShape*» mClusterShapeSet;
    std::unordered_map<std::string, int> mNTotalShapeSet;
    std::unordered_map<std::string, int> mMaxModeSet;
public:
    TClusterShapeAnalyser(const TClusterAnalyser& analyser);
    ~TClusterShapeAnalyser();
    void doShaping(std::string_view typeName, const std::vector<int>& clusterSizeRange);
    void saveIndividualShapes(std::string_view typeName, const CppConfigDictionary config);
    void saveSameSizeInfos(std::string_view typeName, const CppConfigDictionary config);
    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);

private:
    unsigned int fBits;
public:
    enum {
        kNotDeleted = 0x02000000
    };
    bool IsDestructed() const { return !TestBit(kNotDeleted); }
    bool TestBit(unsigned int f) const { return (bool) ((fBits & f) != 0); }
};

#endif
```