

Mixed Data Coincidence Analysis Software

1.0

Generated by Doxygen 1.7.4

Fri Jun 15 2012 21:54:49

Contents

1	Main Page	1
1.1	Introduction	1
1.2	Installation	1
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Module Documentation	13
7.1	Unit tests	13
7.1.1	Detailed Description	14
8	Class Documentation	15
8.1	CoincidenceFinder Class Reference	15
8.1.1	Detailed Description	18
8.1.2	Constructor & Destructor Documentation	18
8.1.2.1	CoincidenceFinder	18
8.1.3	Member Function Documentation	19

8.1.3.1	DoDecay	19
8.1.3.2	DoImplantation	20
8.1.3.3	DoOther	21
8.1.3.4	getDetectorCount	22
8.1.3.5	getFailureReason	23
8.2	CommandLineArgument Class Reference	23
8.2.1	Detailed Description	24
8.2.2	Constructor & Destructor Documentation	24
8.2.2.1	CommandLineArgument	24
8.3	CommandLineException Class Reference	24
8.3.1	Detailed Description	25
8.3.2	Constructor & Destructor Documentation	25
8.3.2.1	CommandLineException	25
8.4	CommandLineInterpreter Class Reference	26
8.4.1	Detailed Description	26
8.4.2	Constructor & Destructor Documentation	26
8.4.2.1	CommandLineInterpreter	26
8.4.3	Member Function Documentation	27
8.4.3.1	readFlaggedCommand	28
8.5	ElementLookupTable Class Reference	28
8.5.1	Detailed Description	29
8.5.2	Member Function Documentation	29
8.5.2.1	lookupElement	29
8.5.2.2	lookupElement	30
8.5.2.3	stringToUpper	30
8.6	EventHit Class Reference	30
8.6.1	Detailed Description	32
8.7	EventHit_TEST Class Reference	32
8.7.1	Detailed Description	32
8.8	FileEventParser Class Reference	33
8.8.1	Detailed Description	33
8.8.2	Constructor & Destructor Documentation	34
8.8.2.1	FileEventParser	34
8.9	FileEventParser_TEST Class Reference	34

8.9.1 Detailed Description	35
8.10 GammaEnergyChart Class Reference	35
8.10.1 Detailed Description	36
8.10.2 Constructor & Destructor Documentation	36
8.10.2.1 GammaEnergyChart	36
8.10.3 Member Function Documentation	37
8.10.3.1 addCoincidence	37
8.11 GammalIdentifier Class Reference	37
8.11.1 Detailed Description	38
8.11.2 Constructor & Destructor Documentation	38
8.11.2.1 GammalIdentifier	38
8.11.3 Member Function Documentation	39
8.11.3.1 addCoincidence	39
8.12 GammaLine Class Reference	39
8.12.1 Detailed Description	40
8.12.2 Constructor & Destructor Documentation	40
8.12.2.1 GammaLine	40
8.12.3 Member Function Documentation	41
8.12.3.1 AbsVal	41
8.12.3.2 addBackground	41
8.12.3.3 addCoincidenceIfValid	41
8.12.3.4 standardDeviation	42
8.12.3.5 sum	42
8.13 GenericUnitTest Class Reference	43
8.13.1 Detailed Description	43
8.14 HitType Class Reference	44
8.14.1 Detailed Description	44
8.15 ScorePair Class Reference	44
8.15.1 Detailed Description	45
8.15.2 Member Function Documentation	45
8.15.2.1 operator<	45
8.16 Settings Struct Reference	45
8.16.1 Detailed Description	47

9	File Documentation	49
9.1	analysis.cc File Reference	49
9.1.1	Detailed Description	49
9.1.2	Function Documentation	50
9.1.2.1	decodeInput	50
9.1.2.2	initSettings	51
9.1.2.3	main	53
9.1.2.4	nsToReadableTime	55
9.1.2.5	printDetectorCount	56
9.1.2.6	printSettings	57
9.1.2.7	printStatistics	58
9.2	analysis.hh File Reference	59
9.2.1	Detailed Description	60
9.2.2	Function Documentation	61
9.2.2.1	decodeInput	61
9.2.2.2	initSettings	63
9.2.2.3	main	64
9.2.2.4	nsToReadableTime	66
9.2.2.5	printDetectorCount	67
9.2.2.6	printSettings	68
9.2.2.7	printStatistics	69
9.3	CoincidenceFinder.cc File Reference	70
9.3.1	Detailed Description	71
9.4	CoincidenceFinder.hh File Reference	71
9.4.1	Detailed Description	71
9.5	CommandLineArgument.cc File Reference	72
9.5.1	Detailed Description	72
9.6	CommandLineArgument.hh File Reference	72
9.6.1	Detailed Description	72
9.7	CommandLineException.cc File Reference	73
9.7.1	Detailed Description	73
9.8	CommandLineException.hh File Reference	73
9.8.1	Detailed Description	73
9.8.2	Define Documentation	74

9.8.2.1	STRING	74
9.9	CommandLineInterpreter.cc File Reference	74
9.9.1	Detailed Description	74
9.10	CommandLineInterpreter.hh File Reference	74
9.10.1	Detailed Description	75
9.11	ElementLookupTable.cc File Reference	75
9.11.1	Detailed Description	75
9.12	ElementLookupTable.hh File Reference	75
9.12.1	Detailed Description	76
9.13	EventHit.cc File Reference	76
9.13.1	Detailed Description	76
9.14	EventHit.hh File Reference	76
9.14.1	Detailed Description	77
9.15	EventHit_TEST.cc File Reference	77
9.15.1	Detailed Description	77
9.16	EventHit_TEST.hh File Reference	78
9.16.1	Detailed Description	78
9.17	FileEventParser.cc File Reference	79
9.17.1	Detailed Description	79
9.18	FileEventParser.hh File Reference	79
9.18.1	Detailed Description	79
9.19	FileEventParser_TEST.cc File Reference	80
9.19.1	Detailed Description	80
9.20	FileEventParser_TEST.hh File Reference	80
9.20.1	Detailed Description	81
9.21	GammaEnergyChart.cc File Reference	81
9.21.1	Detailed Description	81
9.22	GammaEnergyChart.hh File Reference	81
9.22.1	Detailed Description	82
9.23	GammaIdentifier.cc File Reference	82
9.23.1	Detailed Description	82
9.24	GammaIdentifier.hh File Reference	82
9.24.1	Detailed Description	83
9.25	GammaLine.cc File Reference	83

9.25.1 Detailed Description	83
9.26 GammaLine.hh File Reference	84
9.26.1 Detailed Description	84
9.27 GenericUnitTest.cc File Reference	84
9.27.1 Detailed Description	85
9.28 GenericUnitTest.hh File Reference	85
9.28.1 Detailed Description	85
9.29 HitType.cc File Reference	85
9.29.1 Detailed Description	85
9.30 HitType.hh File Reference	86
9.30.1 Detailed Description	86
9.31 RunTests.cc File Reference	86
9.31.1 Detailed Description	87
9.32 RunTests.hh File Reference	87
9.32.1 Detailed Description	88
9.33 ScorePair.cc File Reference	88
9.33.1 Detailed Description	88
9.34 ScorePair.hh File Reference	88
9.34.1 Detailed Description	89

Chapter 1

Main Page

1.1 Introduction

The Mixed Data Coincidence Analysis Software can be used to analyze mixed data from the EventMixer software using a variety of settings.

1.2 Installation

For installation, download the MDCAS from [here](#), unpack it and run "make".

Chapter 2

Todo List

Class `EventHit_TEST` Add more test cases.

Class `FileEventParser_TEST` Add more test cases.

Group `UnitTests` Create more unit tests

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Unit tests	13
----------------------	----

Chapter 4

Class Index

4.1 Class Hierarchy

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

CoincidenceFinder	15
CommandLineArgument	23
CommandLineException	24
CommandLineInterpreter	26
ElementLookupTable	28
EventHit	30
FileEventParser	33
GammaEnergyChart	35
GammaIdentifier	37
GammaLine	39
GenericUnitTest	43
EventHit_TEST	32
FileEventParser_TEST	34
HitType	44
ScorePair	44
Settings	45

Chapter 5

Class Index

5.1 Class List

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

CoincidenceFinder	15
CommandLineArgument (Represents an argument the program expects on the command line)	23
CommandLineException (Thrown when command line arguments are invalid)	24
CommandLineInterpreter (Interprets command line arguments according to a list of CommandLineArguments (acceptable flags), and throws an exception if unknown flags are encountered)	26
ElementLookupTable (Converts element designations (for example NA, Na, na) to atomic numbers (for example 11))	28
EventHit (Contains information about a single event hit, not to be confused with an event)	30
EventHit_TEST (Test class for the EventHit class)	32
FileEventParser (Creates a pointer to a file, and reads events from that file)	33
FileEventParser_TEST (Test class for the FileEventParser class)	34
GammaEnergyChart	35
GammaIdentifier (Utilizes the GammaEnergyChart to find the appropriate matches for the found gammas)	37
GammaLine	39
GenericUnitTest (A generic unit test)	43
HitType (Some extended hit information)	44
ScorePair (Pair score with A and Z, also implement comparison operator)	44
Settings (Contains the settings available when using the command line)	45

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

analysis.cc (Main source file for the analysis software)	49
analysis.hh (Main header file for the analysis software)	59
CoincidenceFinder.cc (Source file for the CoincidenceFinder class)	70
CoincidenceFinder.hh (Header file for the CoincidenceFinder class)	71
CommandLineArgument.cc (Source file for the CommandLineArgument class)	72
CommandLineArgument.hh (Header file for the CommandLineArgument class)	72
CommandLineException.cc (Source file for the CommandLineException class)	73
CommandLineException.hh (Header file for the CommandLineException class)	73
CommandLineInterpreter.cc (Source file for the CommandLineInterpreter class)	74
CommandLineInterpreter.hh (Header file for the CommandLineInterpreter class)	74
ElementLookupTable.cc (Source file for the ElementLookupTable class)	75
ElementLookupTable.hh (Header file for the ElementLookupTable class)	75
EventHit.cc (Source file for the EventHit class)	76
EventHit.hh (Header file for the EventHit class)	76
EventHit_TEST.cc (Source file for the EventHit test class)	77
EventHit_TEST.hh (Header file for the EventHit test class)	78
FileEventParser.cc (Source file for the FileEventParser class)	79
FileEventParser.hh (Header file for the FileEventParser class)	79
FileEventParser_TEST.cc (Source file for the FileEventParser test class)	80
FileEventParser_TEST.hh (Header file for the FileEventParser test class)	80
GammaEnergyChart.cc (Source file for the GammaEnergyChart class)	81
GammaEnergyChart.hh (Header file for the GammaEnergyChart class)	81
GammalIdentifier.cc (Source file for the GammalIdentifier class)	82
GammalIdentifier.hh (Header file for the GammalIdentifier class)	82
GammaLine.cc (Source file for the GammaLine class)	83
GammaLine.hh (Header file for the GammaLine class)	84
GenericUnitTest.cc (Generic test class)	84

GenericUnitTest.hh (Generic test class)	85
HitType.cc (Source file for the HitType class)	85
HitType.hh (Header file for the HitType class)	86
RunTests.cc (The unit test program, executes unit tests)	86
RunTests.hh (Header for the unit test program, executes unit tests)	87
ScorePair.cc (Source file for the ScorePair class)	88
ScorePair.hh (Header file for the ScorePair class)	88

Chapter 7

Module Documentation

7.1 Unit tests

Unit tests for the ENSDF++ software.

Classes

- class [EventHit_TEST](#)
Test class for the [EventHit](#) class.
- class [FileEventParser_TEST](#)
Test class for the [FileEventParser](#) class.

Files

- file [EventHit_TEST.hh](#)
Header file for the [EventHit](#) test class.
- file [EventHit_TEST.cc](#)
Source file for the [EventHit](#) t.
- file [FileEventParser_TEST.hh](#)
Header file for the [FileEventParser](#) test class.
- file [FileEventParser_TEST.cc](#)
Source file for the [FileEventParser](#) t.
- file [RunTests.hh](#)
Header for the unit test program, executes unit tests.
- file [RunTests.cc](#)
The unit test program, executes unit tests.

7.1.1 Detailed Description

Unit tests for the ENSDF++ software.

Author

Rikard Lundmark

Todo

Create more unit tests

Chapter 8

Class Documentation

8.1 CoincidenceFinder Class Reference

Public Member Functions

- [CoincidenceFinder](#) ([FileEventParser](#) *fParser, double maxWait, int nTubes=40, float detEps=1E-2, float minGe=1E-2, float maxGe=1E20, unsigned int multimplant=1, float decayCut=1E20)
Constructor, constructs the object.
- [~CoincidenceFinder](#) ()
Destructor, destroys the object.
- pair< [EventHit](#) *, [EventHit](#) * > [getNextCoincidence](#) ()
Returns the next coincidence pair. Please note that this might be invalid if the hasMoreCoincidences returns false. Calling this method will update the next coincidence to the next one automatically.
- bool [hasMoreCoincidences](#) ()
Returns true if there are more coincidences to be read from the class, otherwise false.
- int [getFailureReason](#) (int reason)
Returns the failure reason.
- int [getNoHits](#) ()
Get number of hits.
- int [getNoImplantations](#) ()
Get number of implantations.
- int [getNoDecays](#) ()
Get number of decays.
- int [getNoOther](#) ()
Get number of others.
- int [getNoCorrectImplantations](#) ()
Get number of correct implantations.
- int [getNoDoubleHit](#) ()

- Get number of double hits.*

 - int [getNoGeHit](#) ()
- Get number of Ge hits.*

 - int [getNoNonAssociatedDecays](#) ()
- Get number of non-associated decays.*

 - int [getNoCorrConnected](#) ()
- Get number of correctly connected implant-decays.*

 - int [getNoErrConnected](#) ()
- Get number of incorrectly connected implant-decays.*

 - int [getNoDestroyedByOther](#) ()
- Get number of implantations destroyed by other-events.*

 - int [getNoCorrCoincidence](#) ()
- Get number of correct coincidences.*

 - int [getNoFoundCoincidences](#) ()
- Get number of found coincidences.*

 - double [getTimeSpan](#) ()
- Get time span.*

 - vector< int > [getDetectorCount](#) ()
- Get detector count for each scintillator tube.*

Private Member Functions

- void [FindNextCoincidence](#) ()

Finds the next coincidence and replaces the nextCoincidence with this. If no coincidence was found, the hasMoreCoincidences is set to false.
- void [DoImplantation](#) (EventHit *myHit)

Called from the FindNextCoincidence routine ONLY.
- int [DoDecay](#) (EventHit *myHit)

Called from the FindNextCoincidence routine ONLY.
- void [DoOther](#) (EventHit *myHit)

Called from the FindNextCoincidence routine ONLY.

Private Attributes

- int [noHits](#)

Number of hits.
- int [failureReasons](#) [8]

Information about why events were classified as "other".
- int [noImplantations](#)

Number of implantations..
- int [noDecays](#)

Number of decays.
- int [noOther](#)

- Number of events other than decay or implant.*

 - int [noCorrectImplantations](#)
- Number of correct implantations.*

 - int [noDoubleHit](#)
- Number of double hits. Note that double-hits reduces the number of usable implantations.*

 - int [noGeHit](#)
- Number of decays resulting in Ge hits.*

 - int [noNonAssociatedDecays](#)
- Number of decays not associated with a correct implantation.*

 - int [noCorrConnected](#)
- Number of decays connected with the correct implantation.*

 - int [noErrConnected](#)
- Number of decays connected with wrong implantation. This should be orders of magnitude lower than noCorrConnected, otherwise we have serious problems...*

 - int [noCorrCoincidence](#)
- Number of correct coincidences.*

 - int [noDestroyedByOther](#)
- Number of implantations destroyed by another event.*

 - int [noFoundCoincidences](#)
- Numer of found coincidences.*

 - double [timeSpan](#)
- Time span.*

 - unsigned int [multipleImplant](#)
- Multiple implantations allowed.*

 - float [decayCutTime](#)
- Decay cut time.*

 - bool [myMoreCoincidences](#)
- True if there are more coincidences to be read out from this object.*

 - vector< pair< list< [EventHit](#) * >, double > > [myScintillatorTubes](#)
- The [FileEventParser](#) connected to this particular event.*

 - [FileEventParser](#) * [myParser](#)
- The [FileEventParser](#) connected to this particular event.*

 - list< pair< [EventHit](#) *, [EventHit](#) * > > [nextCoincidence](#)
- Cointains the next coincidence(s). This may cointain NULL when the hasMoreCoincidences variable is set to false.*

 - double [maxDecayWait](#)
- Maximum time to wait for a decay before deeming the tube empty.*

 - int [numberOfTubes](#)
- Number of scintillator tubes in detector.*

 - float [detectorEpsilon](#)
- Detector epsilon.*

 - float [minGeEnergy](#)
- Minimum germanium energy.*

 - float [maxGeEnergy](#)

Maximum germanium energy.

- `vector< int > detectorCount`

Detector count.

8.1.1 Detailed Description

Definition at line 25 of file CoincidenceFinder.hh.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 `CoincidenceFinder::CoincidenceFinder (FileEventParser * fParser, double maxWait, int nTubes = 40, float detEps = 1E-2, float minGe = 1E-2, float maxGe = 1E20, unsigned int multimplant = 1, float decayCut = 1E20)`

Constructor, constructs the object.

Parameters

<i>fParser</i>	FileEventParser to read from.
<i>maxWait</i>	Maximum decay wait time.
<i>nTubes</i>	Number of scintillator tubes in detector.
<i>detEps</i>	Minimum energy to consider for the beta detector.
<i>minGe</i>	Minimum energy to consider for the germanium detector.
<i>maxGe</i>	Maximum energy to consider for the germanium detector.
<i>multimplant</i>	Number of allowed multiple implantations.
<i>decayCut</i>	Decay cut, if more than this the particle is considered as disappeared.

Definition at line 3 of file CoincidenceFinder.cc.

References `detectorCount`, `detectorEpsilon`, `failureReasons`, `FindNextCoincidence()`, `myParser`, `noCorrCoincidence`, `noCorrConnected`, `noCorrectImplantations`, `noDecays`, `noDestroyedByOther`, `noDoubleHit`, `noErrConnected`, `noFoundCoincidences`, `noGeHit`, `noHits`, `noImplantations`, `noNonAssociatedDecays`, `noOther`, `numberOfTubes`, and `timeSpan`.

```

        :multipleImplant (multiImplant), decayCutTime (decayCut), maxDecayWait (maxWait),
          minGeEnergy (minGe), maxGeEnergy (maxGe)
    {
        numberOfTubes = nTubes;
        detectorEpsilon = detEps;
        myParser = fParser;
        for(int i = 0; i<numberOfTubes; i++)
        {
            pair<list<EventHit*>, double> tmp;
            tmp.second=0-maxWait;
            myScintillatorTubes.push_back (tmp);
            detectorCount.push_back (0);
        }

        //For statistics purpose.
        noHits = 0;
    }

```

```

noImplantations = 0;
noDecays = 0;
noOther = 0;
noCorrectImplantations = 0;
noDoubleHit = 0;
noGeHit = 0;
noNonAssociatedDecays = 0;
noCorrConnected = 0;
noErrConnected = 0;
noDestroyedByOther = 0;
noCorrCoincidence = 0;
noFoundCoincidences = 0;
timeSpan = 0;

memset(failureReasons, 0, sizeof(failureReasons));

FindNextCoincidence();
}

```

8.1.3 Member Function Documentation

8.1.3.1 int CoincidenceFinder::DoDecay (EventHit * *myHit*) [private]

Called from the FindNextCoincidence routine ONLY.

Parameters

<i>myHit</i>	The hit to pair decay with.
--------------	-----------------------------

Definition at line 93 of file CoincidenceFinder.cc.

References HitType::affectedScintillators, decayCutTime, detectorCount, detectorEpsilon, EventHit::eventno, EventHit::Ge, EventHit::implantationType, maxDecayWait, maxGeEnergy, minGeEnergy, myMoreCoincidences, nextCoincidence, noCorrCoincidence, noCorrConnected, noDecays, noErrConnected, noGeHit, noNonAssociatedDecays, and EventHit::time.

Referenced by FindNextCoincidence().

```

{
    ++noDecays;
    int toReturn = 0;
    if(myHit->Ge>detectorEpsilon)
    {
        ++noGeHit;
    }

    if(!myScintillatorTubes[myHit->implantationType.affectedScintillators.front()].
        first.empty() && myHit->time - myScintillatorTubes[myHit->implantationType.
            affectedScintillators.front()].second < decayCutTime) //This decay can be associa
            ted with one/more specific implantation(s), hooray!
    {
        bool corrConnected = false;

        for(list<EventHit*>::iterator ig = myScintillatorTubes[myHit->
            implantationType.affectedScintillators.front()].first.begin(); ig!=myScintillator

```

```

Tubes[myHit->implantationType.affectedScintillators.front()].first.end(); ig++)
{
    if ((*ig)->eventno==myHit->eventno)
    {
        corrConnected = true;
        ++noCorrConnected;
    }
    else
    {
        ++noErrConnected;
    }
}

//Now, if we could only have the gamma energy...
if (myHit->Ge>minGeEnergy && myHit->Ge<maxGeEnergy)
{
    //Now we have coincidence beta-gamma. Let's store the both events, and
    clean up.
    for (list<EventHit*>::iterator ig = myScintillatorTubes[myHit->
implantationType.affectedScintillators.front()].first.begin(); ig!=myScintillator
Tubes[myHit->implantationType.affectedScintillators.front()].first.end(); ig++)
    {
        nextCoincidence.push_back (make_pair (*ig,myHit));
        ++toReturn;
    }
    myMoreCoincidences = true;
    ++detectorCount [myHit->implantationType.affectedScintillators.front()];

    if (corrConnected)
        ++noCorrCoincidence;
}
myScintillatorTubes[myHit->implantationType.affectedScintillators.front()].
first.clear();

myScintillatorTubes[myHit->implantationType.affectedScintillators.front()].
second=-maxDecayWait;
}
else
{
    ++noNonAssociatedDecays;
}

return toReturn;
}

```

8.1.3.2 void CoincidenceFinder::DoImplantation (EventHit * *myHit*) [private]

Called from the FindNextCoincidence routine ONLY.

Parameters

<i>myHit</i>	The hit to do implantation with.
--------------	----------------------------------

Definition at line 51 of file CoincidenceFinder.cc.

References HitType::affectedScintillators, EventHit::implantationType, maxDecayWait, multipleImplant, noCorrectImplantations, noDoubleHit, noImplantations, EventHit::time, and timeSpan.

Referenced by FindNextCoincidence().

```
{
    ++noImplantations;
    int scNr=myHit->implantationType.affectedScintillators.front();
    timeSpan = myHit->time; //in ns.
    if(myScintillatorTubes[scNr].second+maxDecayWait<myHit->time ) //Available for
        implantation (time-out).
    {
        ++noCorrectImplantations;
        if(myScintillatorTubes[scNr].first.size()>=multipleImplant) //We cannot add
            more, yet it is available, so it is OK to clear it.
        {
            if(myScintillatorTubes[scNr].second>0) //If we have reset the time be
            hind zero, we shall not delete the object since this means that we have returned
            this event to somewhere else.
            for(list<EventHit*>::iterator ig = myScintillatorTubes[scNr].first.be
            gin(); ig!=myScintillatorTubes[scNr].first.end(); ig++)
            {
                delete *ig;
                ig=myScintillatorTubes[scNr].first.erase(ig);
                ig--;
            }
            myScintillatorTubes[scNr].first.clear();
        }

        myScintillatorTubes[scNr].first.push_back(myHit);
        if(myScintillatorTubes[scNr].first.size()>=multipleImplant) //No more, plea
        se.
            myScintillatorTubes[scNr].second=myHit->time;
        else
            myScintillatorTubes[scNr].second=-maxDecayWait; //Add more, please.

        return;
    }
    else //not available for implantation, delete all elements and mark as unavaili
    ble.
    {
        ++noDoubleHit;
        myScintillatorTubes[scNr].second=myHit->time; //mark it as unavailable for
        an even longer time.
        delete myHit;
        for(list<EventHit*>::iterator ig = myScintillatorTubes[scNr].first.begin();
        ig!=myScintillatorTubes[scNr].first.end(); ig++)
        {
            delete *ig;
        }
        myScintillatorTubes[scNr].first.clear();
        return;
    }
}
```

8.1.3.3 void CoincidenceFinder::DoOther (EventHit * myHit) [private]

Called from the FindNextCoincidence routine ONLY.

<If we have more than this, we probably haven't destroyed that much... hopefully.

Parameters

<i>myHit</i>	The hit to do the other with.
--------------	-------------------------------

Definition at line 145 of file CoincidenceFinder.cc.

References `HitType::affectedScintillators`, `EventHit::implantationType`, `noDestroyedByOther`, `noOther`, and `EventHit::time`.

Referenced by `FindNextCoincidence()`.

```
{
    ++noOther;
    if (myHit->implantationType.affectedScintillators.size() < 4)
    {
        for (vector<int>::iterator it = myHit->implantationType.
            affectedScintillators.begin(); it != myHit->implantationType.affectedScintillators.
            end(); it++)
        {
            for (list<EventHit*>::iterator ig = myScintillatorTubes[*it].first.begin
                ( ); ig != myScintillatorTubes[*it].first.end(); ig++)
            {
                if (myScintillatorTubes[*it].second > 0)
                {
                    delete *ig;
                    noDestroyedByOther++;
                }
                ig = myScintillatorTubes[*it].first.erase(ig);
                ig--;
            }
            myScintillatorTubes[*it].second = myHit->time; //Block.
        }
    }
    delete myHit;
    return;
}
```

8.1.3.4 vector< int > CoincidenceFinder::getDetectorCount ()

Get detector count for each scintillator tube.

Returns

The detector count for each scintillator tube.

Definition at line 292 of file CoincidenceFinder.cc.

References `detectorCount`.

Referenced by `printDetectorCount()`.

```
{
    return detectorCount;
}
```

8.1.3.5 int CoincidenceFinder::getFailureReason (int *reason*)

Returns the failure reason.

Parameters

<i>reason</i>	Reason
---------------	--------

Definition at line 212 of file CoincidenceFinder.cc.

References failureReasons.

Referenced by printStatistics().

```
{  
    return failureReasons[reason];  
}
```

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

- [CoincidenceFinder.hh](#)
- [CoincidenceFinder.cc](#)

8.2 CommandLineArgument Class Reference

Represents an argument the program expects on the command line.

```
#include <CommandLineArgument.hh>
```

Public Member Functions

- [CommandLineArgument](#) ()
Constructor, sets the members to default values.
- [CommandLineArgument](#) (string mFlag, int mNumberOfArguments)
Constructor, sets the members to the values sent.

Public Attributes

- string [flag](#)
The flag (without the -) which should be expected.
- int [numberOfArguments](#)
The number of arguments to be expected.

8.2.1 Detailed Description

Represents an argument the program expects on the command line.

Sent to the [CommandLineInterpreter](#), this class will throw a [CommandLineException](#) if it encounters a flag it does not recognize.

Author

Rikard Lundmark

Definition at line 25 of file `CommandLineArgument.hh`.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 `CommandLineArgument::CommandLineArgument (string mFlag, int mNumberOfArguments)`

Constructor, sets the members to the values sent.

Parameters

<i>mFlag</i>	flag = mFlag
<i>mNumberOfArguments</i>	numberOfArguments = mNumberOfArguments

Definition at line 8 of file `CommandLineArgument.cc`.

References `flag`, and `numberOfArguments`.

```
{
    flag=mFlag;
    numberOfArguments = mNumberOfArguments;
}
```

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

- [CommandLineArgument.hh](#)
- [CommandLineArgument.cc](#)

8.3 CommandLineException Class Reference

Thrown when command line arguments are invalid.

```
#include <CommandLineException.hh>
```

Public Member Functions

- [CommandLineException](#) (string m="exception!")

Constructor.

- [~CommandLineException](#) () throw ()

Destructor.

- const char * [what](#) () const throw ()

To implement Exception.

Private Attributes

- string [msg](#)

The message string for this exception.

8.3.1 Detailed Description

Thrown when command line arguments are invalid.

Author

Rikard Lundmark

Definition at line 28 of file CommandLineException.hh.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 CommandLineException::CommandLineException (string *m* = "exception!")

Constructor.

Parameters

<i>m</i>	The exception string.
----------	-----------------------

Definition at line 4 of file CommandLineException.cc.

```
:msg (m)
{
}
}
```

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

- [CommandLineException.hh](#)
- [CommandLineException.cc](#)

8.4 CommandLineInterpreter Class Reference

Interprets command line arguments according to a list of CommandLineArguments (acceptable flags), and throws an exception if unknown flags are encountered.

```
#include <CommandLineInterpreter.hh>
```

Public Member Functions

- [CommandLineInterpreter](#) (int argc, char *argv[], list< [CommandLineArgument](#) > myAcceptableFlags)

Constructor.

Exceptions

CommandLineException	if the CommandLineArguments does not match argv and argc (if there are unknown flags).
--------------------------------------	--

- vector< string > [readFlaggedCommand](#) (string flag)

Returns a vector<string> containing all arguments for a specific flag. Returns an empty vector if there are no arguments for this flag.

- vector< string > [readFlaglessCommands](#) ()

Returns all arguments without flags.

Private Attributes

- map< string, vector< string > > [myFlaggedCommands](#)

The current flagged commands.

- vector< string > [myFlaglessCommands](#)

The current flagless commands.

8.4.1 Detailed Description

Interprets command line arguments according to a list of CommandLineArguments (acceptable flags), and throws an exception if unknown flags are encountered.

Author

Rikard Lundmark

Definition at line 42 of file CommandLineInterpreter.hh.

8.4.2 Constructor & Destructor Documentation

- ##### 8.4.2.1 [CommandLineInterpreter::CommandLineInterpreter](#) (int argc, char * argv[], list< [CommandLineArgument](#) > myAcceptableFlags)

Constructor.

Exceptions

<i>CommandLineException</i>	if the CommandLineArguments does not match argv and argc (if there are unknown flags).
---	--

Parameters

<i>argc</i>	The number of command line arguments.
<i>argv</i>	The actual arguments.
<i>myAcceptableFlags</i>	The flags which should be accepted by this interpreter.

Definition at line 3 of file CommandLineInterpreter.cc.

References `myFlaggedCommands`, and `myFlaglessCommands`.

```
{
    for(int i = 0; i<argc; i++)
    {
        string temp(argv[i]);
        if(temp[0]=='-') //we have a flag...
        {
            temp = temp.substr(1,temp.size());
            bool found = false;
            for(list<CommandLineArgument>::iterator it = myAcceptableFlags.begin();
                it!=myAcceptableFlags.end(); it++)
            {
                if(temp.compare(it->flag)==0)
                {
                    if(i+it->numberOfArguments>argc)
                        throw CommandLineException("Insufficient arguments for flag.\n");

                    vector<string> myList;
                    for(int k = i+1; k<=i+it->numberOfArguments; k++)
                    {
                        string tmp(argv[k]);
                        myList.push_back(tmp);
                    }
                    myFlaggedCommands[temp] = myList;
                    i+=it->numberOfArguments;
                    found=true;
                }
            }
            if(!found)
                throw CommandLineException("Command flag not recognized");
        }
        else
        {
            myFlaglessCommands.push_back(argv[i]);
        }
    }
}
```

8.4.3 Member Function Documentation

8.4.3.1 `vector< string > CommandLineInterpreter::readFlaggedCommand (string flag)`

Returns a `vector<string>` containing all arguments for a specific flag. Returns an empty vector if there are no arguments for this flag.

Parameters

<i>flag</i>	The flag to read
-------------	------------------

Definition at line 39 of file `CommandLineInterpreter.cc`.

References `myFlaggedCommands`.

Referenced by `decodeInput()`, and `initSettings()`.

```
{
    if (myFlaggedCommands.find(flag) != myFlaggedCommands.end())
    {
        if (myFlaggedCommands[flag].empty())
            myFlaggedCommands[flag].push_back("");
        return myFlaggedCommands[flag];
    }
    else
    {
        return vector<string>();
    }
}
```

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

- [CommandLineInterpreter.hh](#)
- [CommandLineInterpreter.cc](#)

8.5 ElementLookupTable Class Reference

Converts element designations (for example NA, Na, na) to atomic numbers (for example 11).

```
#include <ElementLookupTable.hh>
```

Public Member Functions

- [ElementLookupTable \(\)](#)
Constructor, constructs the LookupTable.
- unsigned short int [lookupElement](#) (string elementToLookup)
Look up a specific element.
- string [lookupElement](#) (unsigned short int elementToLookup)
Look up a specific element.

Protected Member Functions

- void [stringToUpper](#) (string &s)
Converts a string to upper case.

Protected Attributes

- map< string, unsigned short int > [myTable](#)
Maps the element strings to their atomic numbers.
- map< unsigned short int, string > [myReversedTable](#)
Maps the element numbers to their strings.

8.5.1 Detailed Description

Converts element designations (for example NA, Na, na) to atomic numbers (for example 11).

Author

Rikard Lundmark

Note

The elements are hard-coded.

Definition at line 27 of file ElementLookupTable.hh.

8.5.2 Member Function Documentation

8.5.2.1 unsigned short int ElementLookupTable::lookupElement (string *elementToLookup*)

Look up a specific element.

Returns

the atomic number of the element, or 0 if the element was not found.

Definition at line 130 of file ElementLookupTable.cc.

References [myTable](#), and [stringToUpper\(\)](#).

```
{
    stringToUpper(elementToLookup);
    if(myTable.find(elementToLookup)!=myTable.end())
        return myTable[elementToLookup];
    else
        return 0;
}
```

8.5.2.2 string ElementLookupTable::lookupElement (unsigned short int *elementToLookup*)

Look up a specific element.

Returns

empty string if the element was not found.

Definition at line 139 of file ElementLookupTable.cc.

References myReversedTable.

```
{
    if(myReversedTable.find(elementToLookup)!=myReversedTable.end())
        return myReversedTable[elementToLookup];
    else
        return "";
}
```

8.5.2.3 void ElementLookupTable::stringToUpper (string & s) [protected]

Converts a string to upper case.

Parameters

s	String to convert to uppercase.
----------	---------------------------------

Definition at line 148 of file ElementLookupTable.cc.

Referenced by lookupElement().

```
{
    std::string::iterator i = s.begin();
    std::string::iterator end = s.end();

    while (i != end) {
        *i = std::toupper((unsigned char)*i);
        ++i;
    }
}
```

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

- [ElementLookupTable.hh](#)
- [ElementLookupTable.cc](#)

8.6 EventHit Class Reference

Contains information about a single event hit, not to be confused with an event.

```
#include <EventHit.hh>
```

Public Member Functions

- void [ComputeHitType](#) ()
Computes type of implantation. Call when parameters are correctly set.
- string [toString](#) ()
- [EventHit](#) (int nTubes=40, float detEpsilon=1E-2, float dRatio=1E2)
Constructor, constructs the object.

Public Attributes

- int [A](#)
A of the event.
- int [Z](#)
Z of the event.
- int [eventno](#)
Event number.
- double [time](#)
Time of the event hit, relative the beginning of the event.
- float [UFSP](#)
Upper front scintillator plate energy deposition.
- float [LFSP](#)
Lower front scintillator plate energy deposition.
- float [BSP](#)
Back scintillator plate energy deposition.
- float [Ge](#)
Ge detector energy deposition.
- vector< float > [DET](#)
- [HitType](#) [implantationType](#)
Type of implantation.

Private Member Functions

- int [GetDominating](#) ()

Private Attributes

- int [numberOfTubes](#)
Number of scintillator tubes in detector.
- float [detectorEpsilon](#)
Detector epsilon.
- float [dominateRatio](#)
Ratio for domination.

8.6.1 Detailed Description

Contains information about a single event hit, not to be confused with an event.

Definition at line 35 of file EventHit.hh.

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

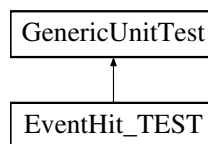
- [EventHit.hh](#)
- [EventHit.cc](#)

8.7 EventHit_TEST Class Reference

Test class for the [EventHit](#) class.

```
#include <EventHit_TEST.hh>
```

Inheritance diagram for EventHit_TEST:



Public Member Functions

- int [runUnitTests](#) () const
Run the unit tests.

Protected Member Functions

- bool [DoubleEquality](#) (double d1, double d2, double eps) const
Double equality.
- bool [EventHit_ClassifiesCorrectly_AssertTrue](#) () const
Test.

8.7.1 Detailed Description

Test class for the [EventHit](#) class.

Author

Rikard Lundmark

Todo

Add more test cases.

Definition at line 47 of file EventHit_TEST.hh.

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

- [EventHit_TEST.hh](#)
- [EventHit_TEST.cc](#)

8.8 FileEventParser Class Reference

Creates a pointer to a file, and reads events from that file.

```
#include <FileEventParser.hh>
```

Public Member Functions

- [FileEventParser](#) (char *filename, int nTubes=40, float detEpsilon=1E-2, float dRatio=1E2)
Constructor, constructs the [FileEventParser](#).
- [~FileEventParser](#) ()
Destructor.
- [EventHit *](#) [getNextEventHit](#) ()
Returns the next event from the file.
- bool [hasMoreEvents](#) ()
Returns the fileEmpty variable.

Private Attributes

- FILE * [myFileStream](#)
The file stream that this object is connected to. Opened upon creation, closed upon destruction.
- bool [fileEmpty](#)
True if file is now empty. If a [getNextEvent\(\)](#) is called when this is true, a zero result will be returned.
- int [numberOfTubes](#)
Number of scintillator tubes in detector.
- float [detectorEpsilon](#)
Detector epsilon.
- float [dominateRatio](#)
Ratio for domination.

8.8.1 Detailed Description

Creates a pointer to a file, and reads events from that file.

Definition at line 25 of file FileEventParser.hh.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 FileEventParser::FileEventParser (char * *filename*, int *nTubes* = 40, float *detEpsilon* = 1E-2, float *dRatio* = 1E2)

Constructor, constructs the [FileEventParser](#).

Parameters

<i>filename</i>	The file to read events from.
-----------------	-------------------------------

Definition at line 3 of file FileEventParser.cc.

References [detectorEpsilon](#), [dominateRatio](#), [fileEmpty](#), [myFileStream](#), and [numberOfTubes](#).

```
{
    numberOfTubes = nTubes;
    detectorEpsilon = detEpsilon;
    dominateRatio = dRatio;
    myFileStream = fopen(filename, "r");
    if(myFileStream==NULL)
    {
        fprintf (stderr, "Could not open input event file %s, exiting.", filename);
        exit(1);
    }
    fileEmpty = false;
}
```

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

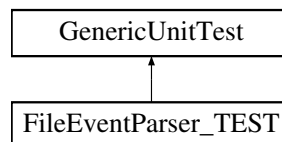
- [FileEventParser.hh](#)
- [FileEventParser.cc](#)

8.9 FileEventParser_TEST Class Reference

Test class for the [FileEventParser](#) class.

```
#include <FileEventParser_TEST.hh>
```

Inheritance diagram for FileEventParser_TEST:



Public Member Functions

- int [runUnitTests](#) () const

Run the unit tests.

Protected Member Functions

- bool [DoubleEquality](#) (double d1, double d2, double eps) const
Double equality.
- bool [FileEventParser_OpensFile_AssertTrue](#) () const
Test.
- bool [FileEventParser_ReturnsCorrectNumberOfHits_AssertTrue](#) () const
Test.
- bool [FileEventParser_ReturnsCorrectHits_AssertTrue](#) () const
Test.

8.9.1 Detailed Description

Test class for the [FileEventParser](#) class.

Author

Rikard Lundmark

Todo

Add more test cases.

Definition at line 46 of file FileEventParser_TEST.hh.

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

- [FileEventParser_TEST.hh](#)
- [FileEventParser_TEST.cc](#)

8.10 GammaEnergyChart Class Reference

Public Member Functions

- [GammaEnergyChart](#) (char *gammaFile, int minA=-100000, int maxA=1000000, int minZ=-100000, int maxZ=1000000, float tol=3E-3, float bgTol=5E-2)
Constructor, constructs the object.
- [~GammaEnergyChart](#) ()
Destructor, destroys the object.
- void [addCoincidence](#) (pair< [EventHit](#) *, [EventHit](#) * > toAdd, float epsilon)
Adds a gamma line.
- map< pair< int, int >, vector< [GammaLine](#) * > > * [getGammaLines](#) ()

Private Attributes

- `map< pair< int, int >, vector< GammaLine * > > * myGammaLines`
Gamma data from text file. Key is <A, Z>, value is a vector with gamma lines.

8.10.1 Detailed Description

Definition at line 23 of file `GammaEnergyChart.hh`.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 `GammaEnergyChart::GammaEnergyChart (char * gammaFile, int minA = -1000000, int maxA = 1000000, int minZ = -1000000, int maxZ = 1000000, float tol = 3E-3, float bgTol = 5E-2)`

Constructor, constructs the object.

Parameters

<i>gammaFile</i>	File containing gamma data of interesting nuclides.
<i>minA</i>	Discard A below this value.
<i>maxA</i>	Discard A above this value.
<i>minZ</i>	Discard Z below this value.
<i>maxZ</i>	Discard Z above this value.

Definition at line 15 of file `GammaEnergyChart.cc`.

References `myGammaLines`.

```
{
    FILE * myFileStream = fopen(gammaFile,"r");
    if(myFileStream==NULL)
    {
        fprintf (stderr,"Could not open input event file %s, exiting.", gammaFile);
        exit(1);
    }
    myGammaLines = new map<pair<int, int>, vector<GammaLine*> >;
    //Read in all gamma lines
    while(!feof(myFileStream))
    {
        int tempA, tempZ;
        float tempGamma, tempProb;
        fscanf(myFileStream, "%d %d %e %e", &tempA, &tempZ, &tempGamma, &tempProb);

        tempGamma/=1000;
        if(!(tempA< minA || tempA > maxA || tempZ < minZ || tempZ > maxZ))
        {
            (*myGammaLines)[make_pair(tempA, tempZ)].push_back(new GammaLine(tol, bgTol));
            (*myGammaLines)[make_pair(tempA, tempZ)].back()->gamma = tempGamma;
            (*myGammaLines)[make_pair(tempA, tempZ)].back()->probability = tempProb;
        }
    }
}
```

```

    }
}

```

8.10.3 Member Function Documentation

8.10.3.1 `void GammaEnergyChart::addCoincidence (pair< EventHit *, EventHit * > toAdd, float epsilon)`

Adds a gamma line.

Parameters

<i>toAdd</i>	The eventhit to add.
<i>epsilon</i>	Tolerance of energy.

Definition at line 40 of file GammaEnergyChart.cc.

References `myGammaLines`.

Referenced by `GammalIdentifier::addCoincidence()`.

```

{
    for(map< pair< int , int > , vector < GammaLine * > > ::iterator it =
        myGammaLines->begin(); it!=myGammaLines->end(); it++)
    {
        for(vector<GammaLine*>::iterator ig = it->second.begin(); ig!=it->second.en
            d(); ig++)
        {
            //cout << (*ig)->gamma << " " << toAdd.second->Ge << " " << abs((*ig)
            ->gamma-toAdd.second->Ge) << " " << (*ig)->gamma-toAdd.second->Ge << " EPS: " <<
            epsilon << endl;
            (*ig)->addCoincidenceIfValid(toAdd);
        }
    }
}

```

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

- [GammaEnergyChart.hh](#)
- [GammaEnergyChart.cc](#)

8.11 GammalIdentifier Class Reference

Utilizes the [GammaEnergyChart](#) to find the appropriate matches for the found gammas.

```
#include <GammalIdentifier.hh>
```

Public Member Functions

- [GammalIdentifier](#) ([GammaEnergyChart](#) *toUse, double tol, float minGamma, float maxGamma, float bgTol=0.05, bool forceCorr=false)

Constructor.

- [~GammaIdentifier](#) ()

Destructor.

- void [addCoincidence](#) (pair< [EventHit](#) *, [EventHit](#) * > toAdd)

Add a detected coincidence for analysis.

- list< [ScorePair](#) > [doSimpleScoring](#) ()

Does a simple (non-probabilistic) scoring of line count.

- list< [ScorePair](#) > [doProbabilisticScoring](#) ()

Does a more advanced probabilistic scoring.

- string [getMATLABstring](#) (int figure)
- string [getPeakString](#) ()

Private Attributes

- [GammaEnergyChart](#) * **myChart**
- double **tolerance**
- double **bgTolerance**
- float **minGammaEnergy**
- float **maxGammaEnergy**
- bool **forceCorrect**
- list< pair< [EventHit](#) *, [EventHit](#) * > > **coincidences**
- [ElementLookupTable](#) * **myTable**

8.11.1 Detailed Description

Utilizes the [GammaEnergyChart](#) to find the appropriate matches for the found gammas.

Definition at line 34 of file `GammalIdentifier.hh`.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 `GammalIdentifier::GammalIdentifier (GammaEnergyChart * toUse, double tol, float minGamma, float maxGamma, float bgTol = 0.05, bool forceCorr = false)`

Constructor.

Parameters

<i>toUse</i>	Energy chart used for identification.
<i>tol</i>	Tolerance.
<i>minGamma</i>	Min gamma energy.
<i>maxGamma</i>	Max gamma energy.

Definition at line 3 of file `GammalIdentifier.cc`.

```
:myChart(toUse), tolerance(tol), bgTolerance(bgTol), minGammaEnergy(minGamma),
```

```

        maxGammaEnergy(maxGamma), forceCorrect(forceCorr)
    {
        myTable = new ElementLookupTable();
    }

```

8.11.3 Member Function Documentation

8.11.3.1 void GammaIdentifier::addCoincidence (pair< EventHit *, EventHit * > toAdd)

Add a detected coincidence for analysis.

Parameters

<i>toAdd</i>	Implant-decay pair
--------------	--------------------

Definition at line 14 of file GammaIdentifier.cc.

References GammaEnergyChart::addCoincidence().

```

{
    myChart->addCoincidence(toAdd, tolerance);
    coincidences.push_back(toAdd);
}

```

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

- [GammaIdentifier.hh](#)
- [GammaIdentifier.cc](#)

8.12 GammaLine Class Reference

Public Member Functions

- [GammaLine](#) (float tol, float bgTol)
- void [addCoincidenceIfValid](#) (pair< [EventHit](#) *, [EventHit](#) * > toAdd)
Add a coincidence if it should be done.
- float [getBackgroundMean](#) ()
Get mean value of background.
- float [getBackgroundSTD](#) ()
Get standard deviation of background.

Public Attributes

- float [gamma](#)
Gamma energy.
- float [probability](#)

Probability.

- vector< pair< [EventHit](#) *, [EventHit](#) * > > [coincidences](#)

Coincidences (valid) added to this gamma line.

Private Member Functions

- float [AbsVal](#) (float toAbs)
Absolute value.
- void [addBackground](#) (float bg)
- float [standardDeviation](#) (vector< float > toDeviate)
Compute standard deviation.
- float [meanValue](#) (vector< float > toMean)
Compute mean value.
- float [sum](#) (vector< float > toSum)
Sum.

Private Attributes

- float [tolerance](#)
Tolerance.
- float [bgTolerance](#)
Background tolerance.
- vector< float > [background](#)
Background events.

8.12.1 Detailed Description

Definition at line 21 of file GammaLine.hh.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 GammaLine::GammaLine (float *tol*, float *bgTol*)

Parameters

<i>tol</i>	Tolerance when adding a gamma energy.
<i>bgTol</i>	Background tolerance when adding that energy.

Definition at line 3 of file GammaLine.cc.

References [background](#).

```
:tolerance(tol), bgTolerance(bgTol)
{
    for(float i = tol-bgTol; i<=tol+bgTol; i+=2*tol)
```



```

    {
        background.push_back(0);
    }
}

```

8.12.3 Member Function Documentation

8.12.3.1 float GammaLine::AbsVal (float *toAbs*) [private]

Absolute value.

Parameters

<i>toAbs</i>	Value to take abs of
--------------	----------------------

Definition at line 36 of file GammaLine.cc.

Referenced by addCoincidenceIfValid().

```

{
    if(toAbs<0)
        return -toAbs;
    return toAbs;
}

```

8.12.3.2 void GammaLine::addBackground (float *bg*) [private]

Parameters

<i>bg</i>	Only add valid background, plz.
-----------	---------------------------------

Definition at line 12 of file GammaLine.cc.

References background, bgTolerance, and tolerance.

Referenced by addCoincidenceIfValid().

```

{
    for(unsigned int i = 0; i<background.size(); i++)
    {
        if((tolerance-bgTolerance+i*tolerance)>bg)
        {
            ++background[i];
            return;
        }
    }
}

```

8.12.3.3 void GammaLine::addCoincidenceIfValid (pair< EventHit *, EventHit * > *toAdd*)

Add a coincidence if it should be done.

Parameters

<i>toAdd</i>	The coincidence to add.
--------------	-------------------------

Definition at line 24 of file GammaLine.cc.

References AbsVal(), addBackground(), bgTolerance, coincidences, gamma, and tolerance.

```
{
    if (AbsVal (toAdd.second->Ge-gamma) < tolerance)
    {
        coincidences.push_back (toAdd);
    }
    if (AbsVal (toAdd.second->Ge-gamma) < bgTolerance)
    {
        addBackground (toAdd.second->Ge);
    }
}
```

8.12.3.4 float GammaLine::standardDeviation (vector< float > *toDeviate*) [private]

Compute standard deviation.

Parameters

<i>toDeviate</i>	To deviate...
------------------	---------------

Definition at line 44 of file GammaLine.cc.

References meanValue().

Referenced by getBackgroundSTD().

```
{
    if (toDeviate.size() < 2)
        return 0;
    float mean = meanValue(toDeviate);
    float stdev = 0;
    for (vector<float>::iterator it = toDeviate.begin(); it != toDeviate.end(); it++)
    {
        stdev += pow(*it - mean, 2);
    }
    stdev /= (toDeviate.size() - 1);
    return sqrt(stdev);
}
```

8.12.3.5 float GammaLine::sum (vector< float > *toSum*) [private]

Sum.

Parameters

<i>toSum</i>	Elements to sum.
--------------	------------------

Definition at line 64 of file GammaLine.cc.

Referenced by `meanValue()`.

```
{
    float sum = 0;
    for(vector<float>::iterator it = toSum.begin(); it!=toSum.end(); it++)
    {
        sum+=*it;
    }
    return sum;
}
```

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

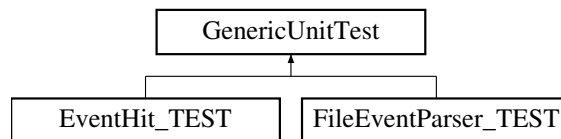
- [GammaLine.hh](#)
- [GammaLine.cc](#)

8.13 GenericUnitTest Class Reference

A generic unit test.

```
#include <GenericUnitTest.hh>
```

Inheritance diagram for GenericUnitTest:



Public Member Functions

- virtual int [runUnitTests](#) () const =0
Run the tests.
- [GenericUnitTest](#) ()
Constructor, constructs the class.

8.13.1 Detailed Description

A generic unit test.

To be extended.

Definition at line 17 of file GenericUnitTest.hh.

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

- [GenericUnitTest.hh](#)
- [GenericUnitTest.cc](#)

8.14 HitType Class Reference

Some extended hit information.

```
#include <HitType.hh>
```

Public Member Functions

- [HitType \(\)](#)
Constructor.

Public Attributes

- bool [wasImplantation](#)
True if implantation.
- bool [wasDecay](#)
True if decay.
- bool [wasOther](#)
True if other.
- vector< int > [affectedScintillators](#)
If implantation or decay, this only contains one number.
- int [statvar](#)
Contains statistics about why we had a failure to identify.

8.14.1 Detailed Description

Some extended hit information.

Definition at line 17 of file HitType.hh.

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

- [HitType.hh](#)
- [HitType.cc](#)

8.15 ScorePair Class Reference

Pair score with A and Z, also implement comparision operator.

```
#include <ScorePair.hh>
```

Public Member Functions

- bool [operator<](#) ([ScorePair](#) rhs)

Public Attributes

- int [A](#)
A of the nuclide.
- int [Z](#)
Z of the nuclide.
- double [score](#)
Score.

8.15.1 Detailed Description

Pair score with A and Z, also implement comparision operator.

Used for scoring and finding which nuclide the gamma lines most probably are related to.

Definition at line 19 of file ScorePair.hh.

8.15.2 Member Function Documentation

8.15.2.1 `bool ScorePair::operator<(ScorePair rhs)` `[inline]`

Parameters

<i>rhs</i>	Score pair for comparisionComparision operator.
------------	---

Definition at line 25 of file ScorePair.hh.

References [score](#).

```
{ return score > rhs.score; }
```

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

- [ScorePair.hh](#)

8.16 Settings Struct Reference

Contains the settings available when using the command line.

```
#include <analysis.hh>
```

Public Attributes

- string [coincidenceFile](#)
Concidence file (MATLAB code printed to this file). Will be overwritten.

- string [timeDistributionFile](#)
Time distribution file. Will be overwritten.
- string [mixedFile](#)
File containing mixed data to analyze.
- string [gammaFile](#)
File containing the gamma levels to use for analysis.
- float [tolerance](#)
Tolerance.
- float [bgTolerance](#)
Background tolerance.
- unsigned int [multiImplant](#)
Multiple implantation allowed.
- int [nTubes](#)
Number of scintillator tubes.
- int [nTubesPerRow](#)
Number of tubes per row.
- float [geEpsilon](#)
Energy resolution of Ge detector.
- float [betaEpsilon](#)
Energy epsilon of beta detectors.
- float [dRatio](#)
Domination ratio (ratio for considering only one scintillator)
- float [minGe](#)
Minimum Ge energy to consider.
- float [maxGe](#)
Maximum Ge energy to consider.
- float [maxDecayWait](#)
Maximum decay wait.
- float [decayCut](#)
Decay cut.
- bool [forceCorrect](#)
Fake all identifications to be correct.
- unsigned short int [minA](#)
Minimum A to consider in the identification.
- unsigned short int [maxA](#)
Maximum A to consider in the identification.
- unsigned short int [minZ](#)
Minimum Z to consider in the identification.
- unsigned short int [maxZ](#)
Maximum Z to consider in the identification.
- bool [detectorCount](#)
Print number of coincidences per detector.
- bool [failInfo](#)
Print info about failed coincidences.

8.16.1 Detailed Description

Contains the settings available when using the command line.

Definition at line 39 of file analysis.hh.

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

- [analysis.hh](#)

Chapter 9

File Documentation

9.1 analysis.cc File Reference

Main source file for the analysis software.

```
#include "analysis.hh"
```

Functions

- [CommandLineInterpreter](#) * [decodeInput](#) (int argc, char **argv)
Verifies input, exists and prints a usage message if this does not conform with expected command line.
- string [nsToReadableTime](#) (double time)
Convert ns to readable time.
- void [printDetectorCount](#) ([CoincidenceFinder](#) *F, int numberPerRow)
Print detector count to stdout.
- void [printStatistics](#) ([CoincidenceFinder](#) *F)
Print statistics from the Coincidence finder.
- void [printSettings](#) ([Settings](#) *toPrint)
Prints the settings to stdout.
- [Settings](#) * [initSettings](#) ([CommandLineInterpreter](#) *myInterpreter)
Initialize a new settings object from a command line interpreter.
- int [main](#) (int argc, char **argv)
Main method.

9.1.1 Detailed Description

Main source file for the analysis software.

Author

Rikard Lundmark

Definition in file [analysis.cc](#).

9.1.2 Function Documentation

9.1.2.1 CommandLineInterpreter* decodeInput (int argc, char ** argv)

Verifies input, exists and prints a usage message if this does not conform with expected command line.

Parameters

<i>argc</i>	Number of arguments.
<i>argv</i>	The actual arguments.

Definition at line 3 of file analysis.cc.

References `CommandLineInterpreter::readFlaggedCommand()`.

Referenced by `main()`.

```
{
    list<CommandLineArgument> myArguments;
    myArguments.push_back(CommandLineArgument("tolerance",1));
    myArguments.push_back(CommandLineArgument("mixedFile",1));
    myArguments.push_back(CommandLineArgument("gammaFile",1));
    myArguments.push_back(CommandLineArgument("coincidenceFile",1));
    myArguments.push_back(CommandLineArgument("minA",1));
    myArguments.push_back(CommandLineArgument("maxA",1));
    myArguments.push_back(CommandLineArgument("minZ",1));
    myArguments.push_back(CommandLineArgument("maxZ",1));
    myArguments.push_back(CommandLineArgument("geEpsilon",1));
    myArguments.push_back(CommandLineArgument("betaEpsilon",1));
    myArguments.push_back(CommandLineArgument("numberOfTubes",1));
    myArguments.push_back(CommandLineArgument("numberPerRow",1));
    myArguments.push_back(CommandLineArgument("maxDecayWait",1));
    myArguments.push_back(CommandLineArgument("dominationFactor", 1));
    myArguments.push_back(CommandLineArgument("gLow", 1));
    myArguments.push_back(CommandLineArgument("gHigh", 1));
    myArguments.push_back(CommandLineArgument("bgTolerance",1));
    myArguments.push_back(CommandLineArgument("decayCut",1));
    myArguments.push_back(CommandLineArgument("detectorCount", 0));
    myArguments.push_back(CommandLineArgument("failInfo", 0));
    myArguments.push_back(CommandLineArgument("help",0));
    myArguments.push_back(CommandLineArgument("forceCorrect",0));
    myArguments.push_back(CommandLineArgument("multiImplant",1));
    myArguments.push_back(CommandLineArgument("timeDistributionFile",1));
    CommandLineInterpreter * myInterpreter = new CommandLineInterpreter(argc, argv,
        myArguments);

    if(!myInterpreter->readFlaggedCommand("help").empty() || myInterpreter->
        readFlaggedCommand("mixedFile").empty() || myInterpreter->readFlaggedCommand("gam
        maFile").empty())
    {
        cout << "The following arguments are valid for the analysis program:" << en
        dl;
        cout << setiosflags(ios::left);
        cout << setw(5) << " " << setw(40) << "-multiImplant <integer value>" << "A
        llow multiple implantation and cross-associate decays." << endl;
    }
}
```

```

    cout << setw(5) << " " << setw(40) << "-detectorCount" << "Count coincidenc
es for different scintillators." << endl;
    cout << setw(5) << " " << setw(40) << "-failInfo" << "Print info about fail
ed ID:s"<< endl;
    cout << setw(5) << " " << setw(40) << "-forceCorrect" << "Force output grap
hs to contain correct lines instead of actual identified lines."<< endl;
    cout << setw(5) << " " << setw(40) << "-tolerance <double value>" << "Gamma
tolerance for identification." << endl;
    cout << setw(5) << " " << setw(40) << "-mixedFile <file name>" << "File con
taining mixed data to analyze." << endl;
    cout << setw(5) << " " << setw(40) << "-gammaFile <file name>" << "File con
taining gamma lines for isotopes, to use for identification" << endl;
    cout << setw(5) << " " << setw(40) << "-coincidenceFile <file name>" << "Fi
le to output coincidence gamma energies to." << endl;
    cout << setw(5) << " " << setw(40) << "-minA <integer value>" << "Minimum A
to include in analysis." << endl;
    cout << setw(5) << " " << setw(40) << "-maxA <integer value>" << "Maximum A
to include in analysis." << endl;
    cout << setw(5) << " " << setw(40) << "-minZ <integer value>" << "Minimum Z
to include in analysis." << endl;
    cout << setw(5) << " " << setw(40) << "-maxZ <integer value>" << "Maximum Z
to include in analysis." << endl;
    cout << setw(5) << " " << setw(40) << "-gLow <double value>" << "Only count
Ge energies above this limit (in MeV)." << endl;
    cout << setw(5) << " " << setw(40) << "-gHigh <double value>" << "Only coun
d Ge energies below this limit (in MeV)." << endl;
    cout << setw(5) << " " << setw(40) << "-geEpsilon <double value>" << "Detec
tor epsilon (threshold energy) for Ge detector. Default 50 keV." << endl;
    cout << setw(5) << " " << setw(40) << "-betaEpsilon <double value>" << "Det
ector epsilon (threshold energy) for beta detector. Default 500 keV." << endl;
    cout << setw(5) << " " << setw(40) << "-numberOfTubes <integer value>" << "
Number of scintillator tubes in detector. Default 40" << endl;
    cout << setw(5) << " " << setw(40) << "-numberPerRow <integer value>" << "N
umber of tubes per row for scintillator count printing. Default 10" << endl;
    cout << setw(5) << " " << setw(40) << "-maxDecayWait <double value>" << "Ma
ximum wait time for decay before deeming it timeout. Default 1E10 ns." << endl;
    cout << setw(5) << " " << setw(40) << "-decayCut <double value>" << "Maximu
m wait time for decay before not accepting implants. Default value 1E10 ns." << e
ndl;
    cout << setw(5) << " " << setw(40) << "-dominationFactor <double value>" <<
"Factor to determine when a decay in a tube is so dominating that detections in
other tubes can be ignored." << endl;
    cout << setw(5) << " " << setw(40) << "-timeDistributionFile <file name>" <
< "File do dump peak time distributions to. Requires -coincidenceFile." << endl;

    cout << setw(5) << " " << setw(40) << "-bgTolerance <double value>" << "Siz
e around the peaks that background will be measured in order to determine if we h
ave a real peak." << endl;
    cout << setw(5) << " " << setw(40) << "-help" << "Displays this help messag
e." << endl;
    exit(0);
}
return myInterpreter;
}

```

9.1.2.2 Settings* initSettings (CommandLineInterpreter * myInterpreter)

Initialize a new settings object from a command line interpreter.

Parameters

<i>myInterpreter</i>	Initialize from this.
----------------------	-----------------------

Definition at line 206 of file analysis.cc.

References Settings::betaEpsilon, Settings::bgTolerance, Settings::coincidenceFile, Settings::decayCut, Settings::detectorCount, Settings::dRatio, Settings::failInfo, Settings::forceCorrect, Settings::gammaFile, Settings::geEpsilon, Settings::maxA, Settings::maxDecayWait, Settings::maxGe, Settings::maxZ, Settings::minA, Settings::minGe, Settings::minZ, Settings::mixedFile, Settings::multiImplant, Settings::nTubes, Settings::nTubesPerRow, CommandLineInterpreter::readFlaggedCommand(), Settings::timeDistributionFile, and Settings::tolerance.

Referenced by main().

```
{
    Settings * S = new Settings();
    S->tolerance = 0.003; //MeV
    if(myInterpreter->readFlaggedCommand("tolerance").size()==1)
        S->tolerance = atof(myInterpreter->readFlaggedCommand("tolerance").front().c_str());
    S->bgTolerance = 0.05; //MeV

    S->multiImplant = myInterpreter->readFlaggedCommand("multiImplant").size()==1?atoi(myInterpreter->readFlaggedCommand("multiImplant").front().c_str()):1;

    S->nTubes = myInterpreter->readFlaggedCommand("numberOfTubes").size()==1?atoi(myInterpreter->readFlaggedCommand("numberOfTubes").front().c_str()):40;
    S->nTubesPerRow = myInterpreter->readFlaggedCommand("numberPerRow").size()==1?atoi(myInterpreter->readFlaggedCommand("numberPerRow").front().c_str()):10;

    S->geEpsilon = myInterpreter->readFlaggedCommand("geEpsilon").size()==1?atof(myInterpreter->readFlaggedCommand("geEpsilon").front().c_str()):5E-2;
    S->betaEpsilon = myInterpreter->readFlaggedCommand("betaEpsilon").size()==1?atof(myInterpreter->readFlaggedCommand("betaEpsilon").front().c_str()):5E-1;

    S->dRatio = myInterpreter->readFlaggedCommand("dominationFactor").size()==1?atof(myInterpreter->readFlaggedCommand("dominationFactor").front().c_str()):1E3;
    S->minGe = myInterpreter->readFlaggedCommand("gLow").size()==1?atof(myInterpreter->readFlaggedCommand("gLow").front().c_str()):5E-1;

    S->maxGe = myInterpreter->readFlaggedCommand("gHigh").size()==1?atof(myInterpreter->readFlaggedCommand("gHigh").front().c_str()):3E0;

    S->maxDecayWait= myInterpreter->readFlaggedCommand("maxDecayWait").size()==1?atof(myInterpreter->readFlaggedCommand("maxDecayWait").front().c_str()):1E10;

    S->decayCut= myInterpreter->readFlaggedCommand("decayCut").size()==1?atof(myInterpreter->readFlaggedCommand("decayCut").front().c_str()):1E20;

    S->mixedFile = myInterpreter->readFlaggedCommand("mixedFile").size()==1?(myInterpreter->readFlaggedCommand("mixedFile").front()):("");
    S->forceCorrect = !myInterpreter->readFlaggedCommand("forceCorrect").empty();

    S->gammaFile =myInterpreter->readFlaggedCommand("gammaFile").size()==1?(myInterpreter->readFlaggedCommand("gammaFile").front()):("");
    S->minA = ((myInterpreter->readFlaggedCommand("minA").size()==1)?atoi(myInterpreter->readFlaggedCommand("minA").front().c_str()):(-1000));
    S->maxA = ((myInterpreter->readFlaggedCommand("maxA").size()==1)?atoi(myInterpreter->readFlaggedCommand("maxA").front().c_str()):1000);
}
```

```

S->minZ = ((myInterpreter->readFlaggedCommand("minZ").size()==1)?atoi(myInterpr
eter->readFlaggedCommand("minZ").front().c_str()):(-1000));
S->maxZ = ((myInterpreter->readFlaggedCommand("maxZ").size()==1)?atoi(myInterpr
eter->readFlaggedCommand("maxZ").front().c_str()):(1000));

S->coincidenceFile = myInterpreter->readFlaggedCommand("coincidenceFile").size(
)==1?(myInterpreter->readFlaggedCommand("coincidenceFile").front()):("");

S->detectorCount = !myInterpreter->readFlaggedCommand("detectorCount").empty();

S->timeDistributionFile = (myInterpreter->readFlaggedCommand("timeDistributionF
ile").size()==1)?(myInterpreter->readFlaggedCommand("timeDistributionFile").front
()):"";
S->failInfo = !myInterpreter->readFlaggedCommand("failInfo").empty();

return S;
}

```

9.1.2.3 int main (int argc, char ** argv)

Main method.

Returns

Exit status.

Parameters

<i>argc</i>	Number of arguments the program was called with.
<i>argv</i>	All the arguments.

Definition at line 252 of file analysis.cc.

References Settings::betaEpsilon, Settings::bgTolerance, Settings::coincidenceFile, Settings::decayCut, decodeInput(), Settings::detectorCount, Settings::dRatio, Settings::failInfo, Settings::forceCorrect, Settings::gammaFile, Settings::geEpsilon, CoincidenceFinder::getNextCoincidence(), CoincidenceFinder::hasMoreCoincidences(), initSettings(), Settings::maxA, Settings::maxDecayWait, Settings::maxGe, Settings::maxZ, Settings::minA, Settings::minGe, Settings::minZ, Settings::mixedFile, Settings::multiImplant, Settings::nTubes, Settings::nTubesPerRow, printDetectorCount(), printSettings(), printStatistics(), Settings::timeDistributionFile, and Settings::tolerance.

```

{
    CommandLineInterpreter * myInterpreter = decodeInput(argc, argv);
    Settings * S = initSettings(myInterpreter);
    printSettings(S);

    FileEventParser * myEventParser = new FileEventParser((char*)S->mixedFile.c_str
(), S->nTubes, S->betaEpsilon, S->dRatio);

    list<pair<EventHit*, EventHit*> > foundCoincidences;

    CoincidenceFinder * myFinder = new CoincidenceFinder(myEventParser, S->
maxDecayWait, S->nTubes, S->geEpsilon, S->minGe, S->maxGe, S->multiImplant, S->
decayCut);
}

```

```

while(myFinder->hasMoreCoincidences())
{
    foundCoincidences.push_back(myFinder->getNextCoincidence());
}

printStats(myFinder);

if(S->coincidenceFile.size()>0)
{
    FILE * coinFile = fopen(S->coincidenceFile.c_str(),"w");
    list<pair<EventHit*, EventHit*> > deletable(foundCoincidences);
    FILE * tFile = NULL;
    if(S->timeDistributionFile.size()>0)
    {
        tFile = fopen(S->timeDistributionFile.c_str(),"w");
    }

    if(coinFile!=NULL)
    {
        int count = 0;
        while(!deletable.empty())
        {
            ++count;
            GammaEnergyChart * myEnergyChart =
                new GammaEnergyChart((char*)S->gammaFile.c_str(), S->minA, S->
maxA, S->minZ, S->maxZ, S->tolerance, S->bgTolerance);

            GammaIdentifier * myIdentifier = new GammaIdentifier(myEnergyChart,
S->tolerance, S->minGe, S->maxGe, S->bgTolerance, S->forceCorrect);
            int A = deletable.front().first->A;
            int Z = deletable.front().first->Z;
            for(list<pair<EventHit*, EventHit*> >::iterator it = deletable.beg
in(); it!=deletable.end(); it++)
            {

                if(it->first->A==A && it->first->Z==Z)
                {
                    myIdentifier->addCoincidence(*it);
                    it=deletable.erase(it);
                    --it;
                }
            }
            fprintf(coinFile,"%s\n",myIdentifier->getMATLABString(count).c_str(
));
            if(tFile!=NULL)
            {
                fprintf(tFile,"%s\n",myIdentifier->getPeakString().c_str());
            }

            delete myIdentifier;
            delete myEnergyChart;
        }
    }
}

if(S->detectorCount)
{
    printDetectorCount(myFinder, S->nTubesPerRow);
}

```

```

if(S->failInfo)
{
    cout << "The failed ID:s:" << endl << endl;
    for(list<pair<EventHit*, EventHit*> >::iterator it = foundCoincidences.begin(); it!=foundCoincidences.end(); it++)
    {
        if(it->first->eventno!=it->second->eventno)
        {
            cout << "Time diff (ms): " << (it->second->time - it->first->time)/
1E6 << endl;
            cout << it->first->toString();
            cout << it->second->toString() << endl;
            cout << endl;
        }
    }
}

set<EventHit*> toDelete;
//Clean up.
for(list<pair<EventHit*, EventHit*> >::iterator it = foundCoincidences.begin(); it!=foundCoincidences.end(); it++)
{
    toDelete.insert(it->first);
    toDelete.insert(it->second);
}
//we have to check as so not make a multiple-delete... that would indeed be bad
.
for(set<EventHit*>::iterator it = toDelete.begin(); it!=toDelete.end(); it++)
{
    delete *it;
}

delete myEventParser;
return 0;
}

```

9.1.2.4 string nsToReadableTime (double *time*)

Convert ns to readable time.

Parameters

<i>time</i>	Time in ns to convert
-------------	-----------------------

Definition at line 65 of file analysis.cc.

Referenced by printStatistics().

```

{
    double second = 1E9;
    double minute = 60*second;
    double hour = 60*minute;
    double day = 24*hour;
    double month = 30*day;
    double year = 12*month;
    stringstream ss;

```

```

if(((int)(time/year))>0)
{
    ss << ((int)(time/year)) << "year" << (((int)(time/year))>1?"s:"");
    time-=year*((int)(time/year));
    ss << " ";
}
if(((int)(time/month))>0)
{
    ss << ((int)(time/month)) << "month" << (((int)(time/month))>1?"s:"");
    time-=month*((int)(time/month));
    ss << " ";
}
if(((int)(time/day))>0)
{
    ss << ((int)(time/day)) << "day" << (((int)(time/day))>1?"s:"");
    time-=day*((int)(time/day));
    ss << " ";
}
if(((int)(time/hour))>0)
{
    ss << ((int)(time/hour)) << "hour" << (((int)(time/hour))>1?"s:"");
    time-=hour*((int)(time/hour));
    ss << " ";
}
if(((int)(time/minute))>0)
{
    ss << ((int)(time/minute)) << "minute" << (((int)(time/minute))>1?"s:"");
    time-=minute*((int)(time/minute));
    ss << " ";
}
if(((int)(time/second))>0)
{
    ss << ((int)(time/second)) << "second" << (((int)(time/second))>1?"s:"");
    time-=second*((int)(time/second));
    ss << " ";
}
return ss.str();
}

```

9.1.2.5 void printDetectorCount (CoincidenceFinder * F, int numberPerRow)

Print detector count to stdout.

Parameters

<i>F</i>	To print detector count from.
<i>numberPer-Row</i>	Number per row.

Definition at line 113 of file analysis.cc.

References CoincidenceFinder::getDetectorCount().

Referenced by main().

```

{
    vector<int> count = F->getDetectorCount();
    int rowbreak = 0;

```



```

cout << "Detector count:" << endl;
cout << setfill(' ');
for(vector<int>::iterator it = count.begin(); it!=count.end(); it++)
{
    cout << setw(5) << *it << " ";
    if(++rowbreak>=numberPerRow)
    {
        rowbreak=0;
        cout << endl;
    }
}
cout << endl;
}

```

9.1.2.6 void printSettings (Settings * toPrint)

Prints the settings to stdout.

Parameters

<i>toPrint</i>	The settings to print.
----------------	------------------------

Definition at line 173 of file analysis.cc.

References Settings::betaEpsilon, Settings::bgTolerance, Settings::coincidenceFile, Settings::decayCut, Settings::detectorCount, Settings::dRatio, Settings::failInfo, Settings::forceCorrect, Settings::gammaFile, Settings::geEpsilon, Settings::maxA, Settings::maxDecayWait, Settings::maxGe, Settings::maxZ, Settings::minA, Settings::minGe, Settings::minZ, Settings::mixedFile, Settings::multiImplant, Settings::nTubes, Settings::nTubesPerRow, Settings::timeDistributionFile, and Settings::tolerance.

Referenced by main().

```

{
    cout << "++++++";
    cout << "Analysis settings:";
    cout << "++++++";
    cout << endl;
    cout << setiosflags(ios::left);
    cout << setfill(' ');
    cout << setw(50) << "Coincidence file:" << toPrint->coincidenceFile << endl;
    cout << setw(50) << "Time distribution file:" << toPrint->timeDistributionFile
        << endl;
    cout << setw(50) << "Mix file (data to analyze):" << toPrint->mixedFile << endl;
    ;
    cout << setw(50) << "Gamma data file:" << toPrint->gammaFile << endl;
    cout << endl;
    cout << setw(50) << "Tolerance (MeV):" << toPrint->tolerance << endl;
    cout << setw(50) << "Background tolerance (MeV):" << toPrint->bgTolerance << endl;
    cout << setw(50) << "#Multi-implantations allowed:" << toPrint->multiImplant << endl;
    cout << setw(50) << "Number of scintillator tubes:" << toPrint->nTubes << endl;
    ;
    cout << setw(50) << "Number of tubes per row:" << toPrint->nTubesPerRow << endl;
    ;
    cout << setw(50) << "Ge detector epsilon (MeV):" << toPrint->geEpsilon << endl;
}

```

```

cout << setw(50) << "Beta detector epsilon (MeV):" << toPrint->betaEpsilon << endl;
cout << setw(50) << "Domination ratio:" << toPrint->dRatio << endl;
cout << setw(50) << "Minimum Ge energy (MeV):" << toPrint->minGe << endl;
cout << setw(50) << "Maximum Ge energy (MeV):" << toPrint->maxGe << endl;
cout << setw(50) << "Maximum decay wait (ns):" << toPrint->maxDecayWait << endl;
;
cout << setw(50) << "Decay cut (ns):" << toPrint->decayCut << endl;
cout << setw(50) << "Force correct identification:" << (toPrint->forceCorrect?"yes":"no") << endl;
cout << setw(50) << "A range:" << toPrint->minA << "-" << toPrint->maxA << endl;
;
cout << setw(50) << "Z range:" << toPrint->minZ << "-" << toPrint->maxZ << endl;
;
cout << setw(50) << "Display detector count:" << (toPrint->detectorCount?"yes":"no") << endl;
cout << setw(50) << "Display info about ID failures:" << (toPrint->failInfo?"yes":"no") << endl;
cout << endl;
}

```

9.1.2.7 void printStatistics (CoincidenceFinder * F)

Print statistics from the Coincidence finder.

Parameters

<i>F</i>	To print statistics from.
----------	---------------------------

Definition at line 131 of file analysis.cc.

References CoincidenceFinder::getFailureReason(), CoincidenceFinder::getNoCorrCoincidence(), CoincidenceFinder::getNoCorrConnected(), CoincidenceFinder::getNoCorrectImplantations(), CoincidenceFinder::getNoDecays(), CoincidenceFinder::getNoDestroyedByOther(), CoincidenceFinder::getNoDoubleHit(), CoincidenceFinder::getNoErrConnected(), CoincidenceFinder::getNoFoundCoincidences(), CoincidenceFinder::getNoGeHit(), CoincidenceFinder::getNoHits(), CoincidenceFinder::getNoImplantations(), CoincidenceFinder::getNoNonAssociatedDecays(), CoincidenceFinder::getNoOther(), CoincidenceFinder::getTimeSpan(), and nsToReadableTime().

Referenced by main().

```

{
//For statistic purposes.
cout << "=====";
cout << "Statistics:";
cout << "=====";
cout << endl;
cout << setiosflags(ios::left);
cout << setfill('_');
cout << setw(50) << "# events processed:" << F->getNoHits() << endl;
cout << setw(50) << "Time of last implantation: " << nsToReadableTime(F->
    getTimeSpan()) << endl;
cout << "Classification of events:" << endl;
cout << setw(50) << "----># implantations:" << F->getNoImplantations() << endl;
}

```

```

cout << setw(50) << "# Correct implantations:" << F->getNoCorrectImplantations(
) << endl;
cout << setw(50) << "# Double hits:" << F->getNoDoubleHit() << endl;
cout << setw(50) << "---># decays:" << F->getNoDecays() << endl;
cout << setw(50) << "# Non-associated decays:" << F->getNoNonAssociatedDecays()
<< endl;
cout << setw(50) << "# Decays without Ge-hit:" << F->getNoGeHit() << endl;
cout << setw(50) << "# Correctly connected implant-decay:" << F->
getNoCorrConnected() << endl;
cout << setw(50) << "# Incorrectly connected implant-decay:" << F->
getNoErrrConnected() << endl;
cout << setw(50) << "---># other:" << F->getNoOther() << endl;
cout << setw(50) << "# Destroyed by other:" << F->getNoDestroyedByOther() << en
dl;
cout << endl;
cout << "Reasons for other-classification:" << endl;
cout << setw(50) << "Particle went through FSP and BSP:" << F->
getFailureReason(0) << endl;
cout << setw(50) << "Particle through FSP but no implant:" << F->
getFailureReason(1) << endl;
cout << setw(50) << "Particle through FSP but implant in several tubes:" << F->
getFailureReason(2) << endl;
cout << setw(50) << "Particle not through SP, and no tube:" << F->
getFailureReason(3) << endl;
cout << setw(50) << "Particle not through SP, and several tubes:" << F->
getFailureReason(4) << endl;
cout << setw(50) << "Particle through BSP, not FSP, no tube:" << F->
getFailureReason(5) << endl;
cout << setw(50) << "Particle through BSP, not FSP, several tubes:" << F->
getFailureReason(6) << endl;
cout << setw(50) << "Particle through BSP, not FSP, one tube:" << F->
getFailureReason(7) << endl;

cout << endl;

cout << setw(50) << "Found coincidences:" << F->getNoFoundCoincidences() << end
l;
cout << setw(50) << "# correct:" << F->getNoCorrCoincidence() << endl;
cout << setw(50) << "Ratio:" << (float)F->getNoCorrCoincidence()/((float)F->
getNoFoundCoincidences())*100 << "%" << endl;

cout << endl;
}

```

9.2 analysis.hh File Reference

Main header file for the analysis software.

```

#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <list>
#include <utility>
#include <string>

```

```

#include <iomanip>
#include <sstream>
#include <set>
#include "GammaEnergyChart.hh"
#include "FileEventParser.hh"
#include "EventHit.hh"
#include "CoincidenceFinder.hh"
#include "CommandLineInterpreter.hh"
#include "CommandLineException.hh"
#include "CommandLineArgument.hh"
#include "GammaIdentifier.hh"
#include "ScorePair.hh"

```

Classes

- struct [Settings](#)

Contains the settings available when using the command line.

Functions

- string [nsToReadableTime](#) (double time)
Convert ns to readable time.
- int [main](#) (int argc, char **argv)
Main method.
- [CommandLineInterpreter](#) * [decodeInput](#) (int argc, char **argv)
Verifies input, exists and prints a usage message if this does not conform with expected command line.
- void [printStatistics](#) ([CoincidenceFinder](#) *F)
Print statistics from the Coincidence finder.
- void [printDetectorCount](#) ([CoincidenceFinder](#) *F, int numberPerRow)
Print detector count to stdout.
- void [printSettings](#) ([Settings](#) *S)
Prints the settings to stdout.
- [Settings](#) * [initSettings](#) ([CommandLineInterpreter](#) *myInterpreter)
Initialize a new settings object from a command line interpreter.

9.2.1 Detailed Description

Main header file for the analysis software.

Author

Rikard Lundmark

Definition in file [analysis.hh](#).

9.2.2 Function Documentation**9.2.2.1 CommandLineInterpreter* decodeInput (int argc, char ** argv)**

Verifies input, exists and prints a usage message if this does not conform with expected command line.

Parameters

<i>argc</i>	Number of arguments.
<i>argv</i>	The actual arguments.

Definition at line 3 of file analysis.cc.

References `CommandLineInterpreter::readFlaggedCommand()`.

Referenced by `main()`.

```
{
    list<CommandLineArgument> myArguments;
    myArguments.push_back(CommandLineArgument("tolerance",1));
    myArguments.push_back(CommandLineArgument("mixedFile",1));
    myArguments.push_back(CommandLineArgument("gammaFile",1));
    myArguments.push_back(CommandLineArgument("coincidenceFile",1));
    myArguments.push_back(CommandLineArgument("minA",1));
    myArguments.push_back(CommandLineArgument("maxA",1));
    myArguments.push_back(CommandLineArgument("minZ",1));
    myArguments.push_back(CommandLineArgument("maxZ",1));
    myArguments.push_back(CommandLineArgument("geEpsilon",1));
    myArguments.push_back(CommandLineArgument("betaEpsilon",1));
    myArguments.push_back(CommandLineArgument("numberOfTubes",1));
    myArguments.push_back(CommandLineArgument("numberPerRow",1));
    myArguments.push_back(CommandLineArgument("maxDecayWait",1));
    myArguments.push_back(CommandLineArgument("dominationFactor", 1));
    myArguments.push_back(CommandLineArgument("gLow", 1));
    myArguments.push_back(CommandLineArgument("gHigh", 1));
    myArguments.push_back(CommandLineArgument("bgTolerance",1));
    myArguments.push_back(CommandLineArgument("decayCut",1));
    myArguments.push_back(CommandLineArgument("detectorCount", 0));
    myArguments.push_back(CommandLineArgument("failInfo", 0));
    myArguments.push_back(CommandLineArgument("help",0));
    myArguments.push_back(CommandLineArgument("forceCorrect",0));
    myArguments.push_back(CommandLineArgument("multiImplant",1));
    myArguments.push_back(CommandLineArgument("timeDistributionFile",1));
    CommandLineInterpreter * myInterpreter = new CommandLineInterpreter(argc, argv,
        myArguments);

    if(!myInterpreter->readFlaggedCommand("help").empty() || myInterpreter->
        readFlaggedCommand("mixedFile").empty() || myInterpreter->readFlaggedCommand("gam
        maFile").empty())
    {
        cout << "The following arguments are valid for the analysis program:" << en
```

```

dl;
cout << setiosflags(ios::left);
cout << setw(5) << " " << setw(40) << "-multiImplant <integer value>" << "A
llow multiple implantation and cross-associate decays." << endl;
cout << setw(5) << " " << setw(40) << "-detectorCount" << "Count coincidenc
es for different scintillators." << endl;
cout << setw(5) << " " << setw(40) << "-failInfo" << "Print info about fail
ed ID:s"<< endl;
cout << setw(5) << " " << setw(40) << "-forceCorrect" << "Force output grap
hs to contain correct lines instead of actual identified lines."<< endl;
cout << setw(5) << " " << setw(40) << "-tolerance <double value>" << "Gamma
tolerance for identification." << endl;
cout << setw(5) << " " << setw(40) << "-mixedFile <file name>" << "File con
taining mixed data to analyze." << endl;
cout << setw(5) << " " << setw(40) << "-gammaFile <file name>" << "File con
taining gamma lines for isotopes, to use for identification" << endl;
cout << setw(5) << " " << setw(40) << "-coincidenceFile <file name>" << "Fi
le to output coincidence gamma energies to." << endl;
cout << setw(5) << " " << setw(40) << "-minA <integer value>" << "Minimum A
to include in analysis." << endl;
cout << setw(5) << " " << setw(40) << "-maxA <integer value>" << "Maximum A
to include in analysis." << endl;
cout << setw(5) << " " << setw(40) << "-minZ <integer value>" << "Minimum Z
to include in analysis." << endl;
cout << setw(5) << " " << setw(40) << "-maxZ <integer value>" << "Maximum Z
to include in analysis." << endl;
cout << setw(5) << " " << setw(40) << "-gLow <double value>" << "Only count
Ge energies above this limit (in MeV)." << endl;
cout << setw(5) << " " << setw(40) << "-gHigh <double value>" << "Only coun
d Ge energies below this limit (in MeV)." << endl;
cout << setw(5) << " " << setw(40) << "-geEpsilon <double value>" << "Detec
tor epsilon (treshold energy) for Ge detector. Default 50 keV." << endl;
cout << setw(5) << " " << setw(40) << "-betaEpsilon <double value>" << "Det
ector epsilon (treshold energy) for beta detector. Default 500 keV." << endl;
cout << setw(5) << " " << setw(40) << "-numberOfTubes <integer value>" << "
Number of scintillator tubes in detector. Default 40" << endl;
cout << setw(5) << " " << setw(40) << "-numberPerRow <integer value>" << "N
umber of tubes per row for scintillator count printing. Default 10" << endl;
cout << setw(5) << " " << setw(40) << "-maxDecayWait <double value>" << "Ma
ximum wait time for decay before deeming it timeout. Default 1E10 ns." << endl;
cout << setw(5) << " " << setw(40) << "-decayCut <double value>" << "Maximu
m wait time for decay before not accepting implants. Default value 1E10 ns." << e
ndl;
cout << setw(5) << " " << setw(40) << "-dominationFactor <double value>" <<
"Factor to determine when a decay in a tube is so dominating that detections in
other tubes can be ignored." << endl;
cout << setw(5) << " " << setw(40) << "-timeDistributionFile <file name>" <
< "File do dump peak time distributions to. Requires -coincidenceFile." << endl;

cout << setw(5) << " " << setw(40) << "-bgTolerance <double value>" << "Siz
e around the peaks that background will be measured in order to determine if we h
ave a real peak." << endl;
cout << setw(5) << " " << setw(40) << "-help" << "Displays this help messag
e." << endl;
exit(0);
}
return myInterpreter;
}

```

9.2.2.2 Settings* initSettings (CommandLineInterpreter * myInterpreter)

Initialize a new settings object from a command line interpreter.

Parameters

<i>myInterpreter</i>	Initialize from this.
----------------------	-----------------------

Definition at line 206 of file analysis.cc.

References Settings::betaEpsilon, Settings::bgTolerance, Settings::coincidenceFile, Settings::decayCut, Settings::detectorCount, Settings::dRatio, Settings::failInfo, Settings::forceCorrect, Settings::gammaFile, Settings::geEpsilon, Settings::maxA, Settings::maxDecayWait, Settings::maxGe, Settings::maxZ, Settings::minA, Settings::minGe, Settings::minZ, Settings::mixedFile, Settings::multiImplant, Settings::nTubes, Settings::nTubesPerRow, CommandLineInterpreter::readFlaggedCommand(), Settings::timeDistributionFile, and Settings::tolerance.

Referenced by main().

```
{
    Settings * S = new Settings();
    S->tolerance = 0.003; //MeV
    if(myInterpreter->readFlaggedCommand("tolerance").size()==1)
        S->tolerance = atof(myInterpreter->readFlaggedCommand("tolerance").front().c_str());
    S->bgTolerance = 0.05; //MeV

    S->multiImplant = myInterpreter->readFlaggedCommand("multiImplant").size()==1?atoi(myInterpreter->readFlaggedCommand("multiImplant").front().c_str()):1;

    S->nTubes = myInterpreter->readFlaggedCommand("numberOfTubes").size()==1?atoi(myInterpreter->readFlaggedCommand("numberOfTubes").front().c_str()):40;
    S->nTubesPerRow = myInterpreter->readFlaggedCommand("numberPerRow").size()==1?atoi(myInterpreter->readFlaggedCommand("numberPerRow").front().c_str()):10;

    S->geEpsilon = myInterpreter->readFlaggedCommand("geEpsilon").size()==1?atof(myInterpreter->readFlaggedCommand("geEpsilon").front().c_str()):5E-2;
    S->betaEpsilon = myInterpreter->readFlaggedCommand("betaEpsilon").size()==1?atof(myInterpreter->readFlaggedCommand("betaEpsilon").front().c_str()):5E-1;

    S->dRatio = myInterpreter->readFlaggedCommand("dominationFactor").size()==1?atof(myInterpreter->readFlaggedCommand("dominationFactor").front().c_str()):1E3;
    S->minGe = myInterpreter->readFlaggedCommand("gLow").size()==1?atof(myInterpreter->readFlaggedCommand("gLow").front().c_str()):5E-1;

    S->maxGe = myInterpreter->readFlaggedCommand("gHigh").size()==1?atof(myInterpreter->readFlaggedCommand("gHigh").front().c_str()):3E0;

    S->maxDecayWait= myInterpreter->readFlaggedCommand("maxDecayWait").size()==1?atof(myInterpreter->readFlaggedCommand("maxDecayWait").front().c_str()):1E10;

    S->decayCut= myInterpreter->readFlaggedCommand("decayCut").size()==1?atof(myInterpreter->readFlaggedCommand("decayCut").front().c_str()):1E20;

    S->mixedFile = myInterpreter->readFlaggedCommand("mixedFile").size()==1?(myInterpreter->readFlaggedCommand("mixedFile").front().c_str());
    S->forceCorrect = !myInterpreter->readFlaggedCommand("forceCorrect").empty();
}
```

```

S->gammaFile = myInterpreter->readFlaggedCommand("gammaFile").size()==1?(myInter
preter->readFlaggedCommand("gammaFile").front()):("");
S->minA = ((myInterpreter->readFlaggedCommand("minA").size()==1)?atoi(myInterpr
eter->readFlaggedCommand("minA").front().c_str()):(-1000));
S->maxA = ((myInterpreter->readFlaggedCommand("maxA").size()==1)?atoi(myInterpr
eter->readFlaggedCommand("maxA").front().c_str()):1000));
S->minZ = ((myInterpreter->readFlaggedCommand("minZ").size()==1)?atoi(myInterpr
eter->readFlaggedCommand("minZ").front().c_str()):(-1000));
S->maxZ = ((myInterpreter->readFlaggedCommand("maxZ").size()==1)?atoi(myInterpr
eter->readFlaggedCommand("maxZ").front().c_str()):1000));

S->coincidenceFile = myInterpreter->readFlaggedCommand("coincidenceFile").size(
)==1?(myInterpreter->readFlaggedCommand("coincidenceFile").front()):("");

S->detectorCount = !myInterpreter->readFlaggedCommand("detectorCount").empty();

S->timeDistributionFile = (myInterpreter->readFlaggedCommand("timeDistributionF
ile").size()==1)?(myInterpreter->readFlaggedCommand("timeDistributionFile").front
()):"";
S->failInfo = !myInterpreter->readFlaggedCommand("failInfo").empty();

return S;
}

```

9.2.2.3 int main(int argc, char ** argv)

Main method.

Returns

Exit status.

Parameters

<i>argc</i>	Number of arguments the program was called with.
<i>argv</i>	All the arguments.

Definition at line 252 of file analysis.cc.

References Settings::betaEpsilon, Settings::bgTolerance, Settings::coincidenceFile, Settings::decayCut, decodeInput(), Settings::detectorCount, Settings::dRatio, Settings::failInfo, Settings::forceCorrect, Settings::gammaFile, Settings::geEpsilon, CoincidenceFinder::getNextCoincidence(), CoincidenceFinder::hasMoreCoincidences(), initSettings(), Settings::maxA, Settings::maxDecayWait, Settings::maxGe, Settings::maxZ, Settings::minA, Settings::minGe, Settings::minZ, Settings::mixedFile, Settings::multiImplant, Settings::nTubes, Settings::nTubesPerRow, printDetectorCount(), printSettings(), printStatistics(), Settings::timeDistributionFile, and Settings::tolerance.

```

{
    CommandLineInterpreter * myInterpreter = decodeInput(argc, argv);
    Settings * S = initSettings(myInterpreter);
    printSettings(S);

    FileEventParser * myEventParser = new FileEventParser((char*)S->mixedFile.c_str
(), S->nTubes, S->betaEpsilon, S->dRatio);
}

```



```

list<pair<EventHit*, EventHit*> > foundCoincidences;

CoincidenceFinder * myFinder = new CoincidenceFinder(myEventParser, S->
    maxDecayWait, S->nTubes, S->geEpsilon, S->minGe, S->maxGe, S->multiImplant, S->
    decayCut);

while(myFinder->hasMoreCoincidences())
{
    foundCoincidences.push_back(myFinder->getNextCoincidence());
}

printStatistics(myFinder);

if(S->coincidenceFile.size()>0)
{
    FILE * coinFile = fopen(S->coincidenceFile.c_str(),"w");
    list<pair<EventHit*, EventHit*> > deletable(foundCoincidences);
    FILE * tFile = NULL;
    if(S->timeDistributionFile.size()>0)
    {
        tFile = fopen(S->timeDistributionFile.c_str(),"w");
    }

    if(coinFile!=NULL)
    {
        int count = 0;
        while(!deletable.empty())
        {
            ++count;
            GammaEnergyChart * myEnergyChart =
                new GammaEnergyChart((char*)S->gammaFile.c_str(), S->minA, S->
                maxA, S->minZ, S->maxZ, S->tolerance, S->bgTolerance);

            GammaIdentifier * myIdentifier = new GammaIdentifier(myEnergyChart,
                S->tolerance, S->minGe, S->maxGe, S->bgTolerance, S->forceCorrect);
            int A = deletable.front().first->A;
            int Z = deletable.front().first->Z;
            for(list<pair<EventHit*, EventHit*> >::iterator it = deletable.beg
in(); it!=deletable.end(); it++)
            {

                if(it->first->A==A && it->first->Z==Z)
                {
                    myIdentifier->addCoincidence(*it);
                    it=deletable.erase(it);
                    --it;
                }
            }
            fprintf(coinFile,"%s\n",myIdentifier->getMATLABString(count).c_str(
));
            if(tFile!=NULL)
            {
                fprintf(tFile,"%s\n",myIdentifier->getPeakString().c_str());
            }

            delete myIdentifier;
            delete myEnergyChart;
        }
    }
}

```

```

    }

    if(S->detectorCount)
    {
        printDetectorCount(myFinder, S->nTubesPerRow);
    }

    if(S->failInfo)
    {
        cout << "The failed ID:s:" << endl << endl;
        for(list<pair<EventHit*, EventHit*> >::iterator it = foundCoincidences.begin(); it!=foundCoincidences.end(); it++)
        {
            if(it->first->eventno!=it->second->eventno)
            {
                cout << "Time diff (ms): " << (it->second->time - it->first->time)/
1E6 << endl;
                cout << it->first->toString();
                cout << it->second->toString() << endl;
                cout << endl;
            }
        }
    }

    set<EventHit*> toDelete;
    //Clean up.
    for(list<pair<EventHit*, EventHit*> >::iterator it = foundCoincidences.begin(); it!=foundCoincidences.end(); it++)
    {
        toDelete.insert(it->first);
        toDelete.insert(it->second);
    }
    //we have to check as so not make a multiple-delete... that would indeed be bad
    .
    for(set<EventHit*>::iterator it = toDelete.begin(); it!=toDelete.end(); it++)
    {
        delete *it;
    }

    delete myEventParser;
    return 0;
}

```

9.2.2.4 string nsToReadableTime (double *time*)

Convert ns to readable time.

Parameters

<i>time</i>	Time in ns to convert
-------------	-----------------------

Definition at line 65 of file analysis.cc.

Referenced by printStatistics().

```

{
    double second = 1E9;

```

```

double minute = 60*second;
double hour = 60*minute;
double day = 24*hour;
double month = 30*day;
double year = 12*month;
stringstream ss;
if(((int)(time/year))>0)
{
    ss << ((int)(time/year)) << "year" << (((int)(time/year))>1?"s:"");
    time-=year*((int)(time/year));
    ss << " ";
}
if(((int)(time/month))>0)
{
    ss << ((int)(time/month)) << "month" << (((int)(time/month))>1?"s:"");
    time-=month*((int)(time/month));
    ss << " ";
}
if(((int)(time/day))>0)
{
    ss << ((int)(time/day)) << "day" << (((int)(time/day))>1?"s:"");
    time-=day*((int)(time/day));
    ss << " ";
}
if(((int)(time/hour))>0)
{
    ss << ((int)(time/hour)) << "hour" << (((int)(time/hour))>1?"s:"");
    time-=hour*((int)(time/hour));
    ss << " ";
}
if(((int)(time/minute))>0)
{
    ss << ((int)(time/minute)) << "minute" << (((int)(time/minute))>1?"s:"");
    time-=minute*((int)(time/minute));
    ss << " ";
}
if(((int)(time/second))>0)
{
    ss << ((int)(time/second)) << "second" << (((int)(time/second))>1?"s:"");
    time-=second*((int)(time/second));
    ss << " ";
}
return ss.str();
}

```

9.2.2.5 void printDetectorCount (CoincidenceFinder * *F*, int *numberPerRow*)

Print detector count to stdout.

Parameters

<i>F</i>	To print detector count from.
<i>numberPer-Row</i>	Number per row.

Definition at line 113 of file analysis.cc.

References CoincidenceFinder::getDetectorCount().

Referenced by main().

```
{
    vector<int> count = F->getDetectorCount();
    int rowbreak = 0;
    cout << "Detector count:" << endl;
    cout << setfill(' ');
    for(vector<int>::iterator it = count.begin(); it!=count.end(); it++)
    {
        cout << setw(5) << *it << " ";
        if(++rowbreak>=numberPerRow)
        {
            rowbreak=0;
            cout << endl;
        }
    }
    cout << endl;
}
```

9.2.2.6 void printSettings (Settings * S)

Prints the settings to stdout.

Parameters

S	The settings to print.
----------	------------------------

Definition at line 173 of file analysis.cc.

References Settings::betaEpsilon, Settings::bgTolerance, Settings::coincidenceFile, Settings::decayCut, Settings::detectorCount, Settings::dRatio, Settings::failInfo, Settings::forceCorrect, Settings::gammaFile, Settings::geEpsilon, Settings::maxA, Settings::maxDecayWait, Settings::maxGe, Settings::maxZ, Settings::minA, Settings::minGe, Settings::minZ, Settings::mixedFile, Settings::multiImplant, Settings::nTubes, Settings::nTubesPerRow, Settings::timeDistributionFile, and Settings::tolerance.

Referenced by main().

```
{
    cout << "++++++";
    cout << "Analysis settings:";
    cout << "++++++";
    cout << endl;
    cout << setiosflags(ios::left);
    cout << setfill('_');
    cout << setw(50) << "Coincidence file:" << toPrint->coincidenceFile << endl;
    cout << setw(50) << "Time distribution file:" << toPrint->timeDistributionFile
        << endl;
    cout << setw(50) << "Mix file (data to analyze):" << toPrint->mixedFile << endl
        ;
    cout << setw(50) << "Gamma data file:" << toPrint->gammaFile << endl;
    cout << endl;
    cout << setw(50) << "Tolerance (MeV):" << toPrint->tolerance << endl;
    cout << setw(50) << "Background tolerance (MeV):" << toPrint->bgTolerance << endl;
    cout << setw(50) << "#Multi-implantations allowed:" << toPrint->multiImplant <<
```

```

    endl;
    cout << setw(50) << "Number of scintillator tubes:" << toPrint->nTubes << endl;

    cout << setw(50) << "Number of tubes per row:" << toPrint->nTubesPerRow << endl;
    ;
    cout << setw(50) << "Ge detector epsilon (MeV):" << toPrint->geEpsilon << endl;

    cout << setw(50) << "Beta detector epsilon (MeV):" << toPrint->betaEpsilon << endl;
    endl;
    cout << setw(50) << "Domination ratio:" << toPrint->dRatio << endl;
    cout << setw(50) << "Minimum Ge energy (MeV):" << toPrint->minGe << endl;
    cout << setw(50) << "Maximum Ge energy (MeV):" << toPrint->maxGe << endl;
    cout << setw(50) << "Maximum decay wait (ns):" << toPrint->maxDecayWait << endl;
    ;
    cout << setw(50) << "Decay cut (ns):" << toPrint->decayCut << endl;
    cout << setw(50) << "Force correct identification:" << (toPrint->forceCorrect?"yes":
    "no") << endl;
    cout << setw(50) << "A range:" << toPrint->minA << "-" << toPrint->maxA << endl;
    ;
    cout << setw(50) << "Z range:" << toPrint->minZ << "-" << toPrint->maxZ << endl;
    ;
    cout << setw(50) << "Display detector count:" << (toPrint->detectorCount?"yes":
    "no") << endl;
    cout << setw(50) << "Display info about ID failures:" << (toPrint->failInfo?"yes":
    "no") << endl;
    cout << endl;
}

```

9.2.2.7 void printStatistics (CoincidenceFinder * F)

Print statistics from the Coincidence finder.

Parameters

<i>F</i>	To print statistics from.
----------	---------------------------

Definition at line 131 of file analysis.cc.

References CoincidenceFinder::getFailureReason(), CoincidenceFinder::getNoCorrCoincidence(), CoincidenceFinder::getNoCorrConnected(), CoincidenceFinder::getNoCorrectImplantations(), CoincidenceFinder::getNoDecays(), CoincidenceFinder::getNoDestroyedByOther(), CoincidenceFinder::getNoDoubleHit(), CoincidenceFinder::getNoErrConnected(), CoincidenceFinder::getNoFoundCoincidences(), CoincidenceFinder::getNoGeHit(), CoincidenceFinder::getNoHits(), CoincidenceFinder::getNoImplantations(), CoincidenceFinder::getNoNonAssociatedDecays(), CoincidenceFinder::getNoOther(), CoincidenceFinder::getTimeSpan(), and nsToReadableTime().

Referenced by main().

```

{
    //For statistic purposes.
    cout << "=====";
    cout << "Statistics:";
    cout << "=====";
    cout << endl;
    cout << setiosflags(ios::left);
}

```

```

cout << setfill('_');
cout << setw(50) << "# events processed:" << F->getNoHits() << endl;
cout << setw(50) << "Time of last implantation: " << nsToReadableTime(F->
    getTimeSpan()) << endl;
cout << "Classification of events:" << endl;
cout << setw(50) << "----># implantations:" << F->getNoImplantations() << endl;
cout << setw(50) << "# Correct implantations:" << F->getNoCorrectImplantations(
    ) << endl;
cout << setw(50) << "# Double hits:" << F->getNoDoubleHit() << endl;
cout << setw(50) << "----># decays:" << F->getNoDecays() << endl;
cout << setw(50) << "# Non-associated decays:" << F->getNoNonAssociatedDecays()
    << endl;
cout << setw(50) << "# Decays without Ge-hit:" << F->getNoGeHit() << endl;
cout << setw(50) << "# Correctly connected implant-decay:" << F->
    getNoCorrConnected() << endl;
cout << setw(50) << "# Incorrectly connected implant-decay:" << F->
    getNoErrConnected() << endl;
cout << setw(50) << "----># other:" << F->getNoOther() << endl;
cout << setw(50) << "# Destroyed by other:" << F->getNoDestroyedByOther() << en
    dl;
cout << endl;
cout << "Reasons for other-classification:" << endl;
cout << setw(50) << "Particle went through FSP and BSP:" << F->
    getFailureReason(0) << endl;
cout << setw(50) << "Particle through FSP but no implant:" << F->
    getFailureReason(1) << endl;
cout << setw(50) << "Particle through FSP but implant in several tubes:" << F->
    getFailureReason(2) << endl;
cout << setw(50) << "Particle not through SP, and no tube:" << F->
    getFailureReason(3) << endl;
cout << setw(50) << "Particle not through SP, and several tubes:" << F->
    getFailureReason(4) << endl;
cout << setw(50) << "Particle through BSP, not FSP, no tube:" << F->
    getFailureReason(5) << endl;
cout << setw(50) << "Particle through BSP, not FSP, several tubes:" << F->
    getFailureReason(6) << endl;
cout << setw(50) << "Particle through BSP, not FSP, one tube:" << F->
    getFailureReason(7) << endl;

cout << endl;

cout << setw(50) << "Found coincidences:" << F->getNoFoundCoincidences() << end
    l;
cout << setw(50) << "# correct:" << F->getNoCorrCoincidence() << endl;
cout << setw(50) << "Ratio:" << (float)F->getNoCorrCoincidence() / ((float)F->
    getNoFoundCoincidences()) * 100 << "%" << endl;

cout << endl;
}

```

9.3 CoincidenceFinder.cc File Reference

Source file for the [CoincidenceFinder](#) class.

```
#include "CoincidenceFinder.hh"
```

9.3.1 Detailed Description

Source file for the [CoincidenceFinder](#) class.

Author

Rikard Lundmark

Definition in file [CoincidenceFinder.cc](#).

9.4 CoincidenceFinder.hh File Reference

Header file for the [CoincidenceFinder](#) class.

```
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <list>
#include <utility>
#include <string>
#include "GammaEnergyChart.hh"
#include "FileEventParser.hh"
#include "EventHit.hh"
```

Classes

- class [CoincidenceFinder](#)

Defines

- #define [COINCIDENCE_FINDER_HH](#)
Inclusion guard.

9.4.1 Detailed Description

Header file for the [CoincidenceFinder](#) class.

Author

Rikard Lundmark

Definition in file [CoincidenceFinder.hh](#).

9.5 CommandLineArgument.cc File Reference

Source file for the [CommandLineArgument](#) class.

```
#include "CommandLineArgument.hh"
```

9.5.1 Detailed Description

Source file for the [CommandLineArgument](#) class.

Author

Rikard Lundmark

Definition in file [CommandLineArgument.cc](#).

9.6 CommandLineArgument.hh File Reference

Header file for the [CommandLineArgument](#) class.

```
#include <string>
```

Classes

- class [CommandLineArgument](#)
Represents an argument the program expects on the command line.

Defines

- #define [STRING](#)
Inclusion guard.
- #define [COMMANDLINEARGUMENT_H](#)
Inclusion guard.

9.6.1 Detailed Description

Header file for the [CommandLineArgument](#) class.

Author

Rikard Lundmark

Definition in file [CommandLineArgument.hh](#).

9.7 CommandLineException.cc File Reference

Source file for the [CommandLineException](#) class.

```
#include "CommandLineException.hh"
```

9.7.1 Detailed Description

Source file for the [CommandLineException](#) class.

Author

Rikard Lundmark

Definition in file [CommandLineException.cc](#).

9.8 CommandLineException.hh File Reference

Header file for the [CommandLineException](#) class.

```
#include <exception>
```

```
#include <string>
```

Classes

- class [CommandLineException](#)
Thrown when command line arguments are invalid.

Defines

- #define [STRING](#)
<Inclusion guard.
- #define [COMMANDLINEEXCEPTION_H](#)
Inclusion guard.

9.8.1 Detailed Description

Header file for the [CommandLineException](#) class.

Author

Rikard Lundmark

Definition in file [CommandLineException.hh](#).

9.8.2 Define Documentation

9.8.2.1 #define STRING

<Inclusion guard.

Inclusion guard

Definition at line 19 of file CommandLineException.hh.

9.9 CommandLineInterpreter.cc File Reference

Source file for the [CommandLineInterpreter](#) class.

```
#include "CommandLineInterpreter.hh"
```

9.9.1 Detailed Description

Source file for the [CommandLineInterpreter](#) class.

Author

Rikard Lundmark

Definition in file [CommandLineInterpreter.cc](#).

9.10 CommandLineInterpreter.hh File Reference

Header file for the [CommandLineInterpreter](#) class.

```
#include <string>
#include <vector>
#include <map>
#include <list>
#include "CommandLineArgument.hh"
#include "CommandLineException.hh"
```

Classes

- class [CommandLineInterpreter](#)

Interprets command line arguments according to a list of CommandLineArguments (acceptable flags), and throws an exception if unknown flags are encountered.

Defines

- `#define` [STRING](#)
Inclusion guard.
- `#define` [VECTOR](#)
Inclusion guard.
- `#define` [MAP](#)
Inclusion guard.
- `#define` [LIST](#)
Inclusion guard.
- `#define` [COMMANDLINEINTERPRETER_H](#)
Inclusion guard.

9.10.1 Detailed Description

Header file for the [CommandLineInterpreter](#) class.

Author

Rikard Lundmark

Definition in file [CommandLineInterpreter.hh](#).

9.11 ElementLookupTable.cc File Reference

Source file for the [ElementLookupTable](#) class.

```
#include "ElementLookupTable.hh"
```

9.11.1 Detailed Description

Source file for the [ElementLookupTable](#) class.

Author

Rikard Lundmark

Definition in file [ElementLookupTable.cc](#).

9.12 ElementLookupTable.hh File Reference

Header file for the [ElementLookupTable](#) class.

```
#include <string>
#include <map>
```

Classes

- class [ElementLookupTable](#)

Converts element designations (for example NA, Na, na) to atomic numbers (for example 11).

Defines

- #define [STRING](#)
Inclusion guard.
- #define [MAP](#)
Inclusion guard.

9.12.1 Detailed Description

Header file for the [ElementLookupTable](#) class.

Author

Rikard Lundmark

Definition in file [ElementLookupTable.hh](#).

9.13 EventHit.cc File Reference

Source file for the [EventHit](#) class.

```
#include "EventHit.hh"
```

9.13.1 Detailed Description

Source file for the [EventHit](#) class.

Author

Rikard Lundmark

Definition in file [EventHit.cc](#).

9.14 EventHit.hh File Reference

Header file for the [EventHit](#) class.

```
#include <sstream>
```

```
#include <string>
```

```
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <iostream>
#include "HitType.hh"
```

Classes

- class [EventHit](#)

Contains information about a single event hit, not to be confused with an event.

Defines

- #define [EVENTHIT_H](#)

Inclusion guard.

9.14.1 Detailed Description

Header file for the [EventHit](#) class.

Author

Rikard Lundmark

Definition in file [EventHit.hh](#).

9.15 EventHit_TEST.cc File Reference

Source file for the [EventHit](#) t.

```
#include "EventHit_TEST.hh"
```

9.15.1 Detailed Description

Source file for the [EventHit](#) t.

Author

Rikard Lundmark

Definition in file [EventHit_TEST.cc](#).

9.16 EventHit_TEST.hh File Reference

Header file for the [EventHit](#) test class.

```
#include <assert.h>
#include <iostream>
#include <list>
#include <string>
#include "FileEventParser.hh"
#include "EventHit.hh"
#include "GenericUnitTest.hh"
#include "HitType.hh"
```

Classes

- class [EventHit_TEST](#)
Test class for the [EventHit](#) class.

Defines

- #define [ASSERT_H](#)
Inclusion guard.
- #define [IOSTREAM](#)
Inclusion guard.
- #define [LIST](#)
Inclusion guard.
- #define [STRING](#)
Inclusion guard.
- #define [EVENTHIT_TEST_HH](#)
Inclusion guard.

9.16.1 Detailed Description

Header file for the [EventHit](#) test class.

Author

Rikard Lundmark

Definition in file [EventHit_TEST.hh](#).

9.17 FileEventParser.cc File Reference

Source file for the [EventHit](#) class.

```
#include "FileEventParser.hh"
```

9.17.1 Detailed Description

Source file for the [EventHit](#) class. Source file for the [FileEventParser](#) class.

Author

Rikard Lundmark

Definition in file [FileEventParser.cc](#).

9.18 FileEventParser.hh File Reference

Header file for the [FileEventParser](#) class.

```
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <string.h>
#include <iostream>
#include "EventHit.hh"
```

Classes

- class [FileEventParser](#)
Creates a pointer to a file, and reads events from that file.

Defines

- #define [FILEEVENTPARSER_H](#)
Inclusion guard.

9.18.1 Detailed Description

Header file for the [FileEventParser](#) class.

Author

Rikard Lundmark

Definition in file [FileEventParser.hh](#).

9.19 FileEventParser_TEST.cc File Reference

Source file for the [FileEventParser](#) t.

```
#include "FileEventParser_TEST.hh"
```

9.19.1 Detailed Description

Source file for the [FileEventParser](#) t.

Author

Rikard Lundmark

Definition in file [FileEventParser_TEST.cc](#).

9.20 FileEventParser_TEST.hh File Reference

Header file for the [FileEventParser](#) test class.

```
#include <assert.h>
#include <iostream>
#include <list>
#include <string>
#include "FileEventParser.hh"
#include "EventHit.hh"
#include "GenericUnitTest.hh"
```

Classes

- class [FileEventParser_TEST](#)
Test class for the [FileEventParser](#) class.

Defines

- #define [ASSERT_H](#)
Inclusion guard.
- #define [IOSTREAM](#)
Inclusion guard.
- #define [LIST](#)

- Inclusion guard.*
- #define [STRING](#)
- Inclusion guard.*
- #define [FILEEVENTPARSER_TEST_HH](#)
- Inclusion guard.*

9.20.1 Detailed Description

Header file for the [FileEventParser](#) test class.

Author

Rikard Lundmark

Definition in file [FileEventParser_TEST.hh](#).

9.21 GammaEnergyChart.cc File Reference

Source file for the [GammaEnergyChart](#) class.

```
#include "GammaEnergyChart.hh"
```

9.21.1 Detailed Description

Source file for the [GammaEnergyChart](#) class.

Author

Rikard Lundmark

Definition in file [GammaEnergyChart.cc](#).

9.22 GammaEnergyChart.hh File Reference

Header file for the [GammaEnergyChart](#) class.

```
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <utility>
#include <map>
#include "EventHit.hh"
#include "GammaLine.hh"
```

Classes

- class [GammaEnergyChart](#)

Defines

- #define [GAMMAENERGYCHART_HH](#)
Inclusion guard.

9.22.1 Detailed Description

Header file for the [GammaEnergyChart](#) class.

Author

Rikard Lundmark

Definition in file [GammaEnergyChart.hh](#).

9.23 GammalIdentifier.cc File Reference

Source file for the [GammalIdentifier](#) class.

```
#include "GammaIdentifier.hh"
```

9.23.1 Detailed Description

Source file for the [GammalIdentifier](#) class.

Author

Rikard Lundmark

Definition in file [GammalIdentifier.cc](#).

9.24 GammalIdentifier.hh File Reference

Header file for the [GammalIdentifier](#) class.

```
#include <sstream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
#include <vector>
```

```
#include <iostream>
#include <utility>
#include <algorithm>
#include <list>
#include <math.h>
#include "EventHit.hh"
#include "GammaEnergyChart.hh"
#include "ScorePair.hh"
#include "ElementLookupTable.hh"
```

Classes

- class [GammalIdentifier](#)

Utilizes the [GammaEnergyChart](#) to find the appropriate matches for the found gammas.

Defines

- #define [GAMMAIDENTIFIER_HH](#)

Inclusion guard.

9.24.1 Detailed Description

Header file for the [GammalIdentifier](#) class.

Author

Rikard Lundmark

Definition in file [GammalIdentifier.hh](#).

9.25 GammaLine.cc File Reference

Source file for the [GammaLine](#) class.

```
#include "GammaLine.hh"
```

9.25.1 Detailed Description

Source file for the [GammaLine](#) class.

Author

Rikard Lundmark

Definition in file [GammaLine.cc](#).

9.26 GammaLine.hh File Reference

Header file for the [GammaLine](#) class.

```
#include <vector>
#include <utility>
#include <iostream>
#include <math.h>
#include "EventHit.hh"
```

Classes

- class [GammaLine](#)

Defines

- #define [GAMMALINE_HH](#)
Inclusion guard.

9.26.1 Detailed Description

Header file for the [GammaLine](#) class.

Author

Rikard Lundmark

Definition in file [GammaLine.hh](#).

9.27 GenericUnitTest.cc File Reference

Generic test class.

```
#include "GenericUnitTest.hh"
```

9.27.1 Detailed Description

Generic test class.

Author

Rikard Lundmark

Definition in file [GenericUnitTest.cc](#).

9.28 GenericUnitTest.hh File Reference

Generic test class.

Classes

- class [GenericUnitTest](#)
A generic unit test.

Defines

- #define [GENERIC_UNIT_TEST_H](#)
Inclusion guard.

9.28.1 Detailed Description

Generic test class.

Author

Rikard Lundmark

Definition in file [GenericUnitTest.hh](#).

9.29 HitType.cc File Reference

Source file for the [HitType](#) class.

```
#include "HitType.hh"
```

9.29.1 Detailed Description

Source file for the [HitType](#) class.

Author

Rikard Lundmark

Definition in file [HitType.cc](#).

9.30 HitType.hh File Reference

Header file for the [HitType](#) class.

```
#include <vector>
```

Classes

- class [HitType](#)
Some extended hit information.

Defines

- #define [HITTYPE_H](#)
Inclusion guard.

9.30.1 Detailed Description

Header file for the [HitType](#) class.

Author

Rikard Lundmark

Definition in file [HitType.hh](#).

9.31 RunTests.cc File Reference

The unit test program, executes unit tests.

```
#include "RunTests.hh"
```

Functions

- int [main](#) ()
The main test function and program entry point.
- void [addTests](#) ()
Called by [main\(\)](#) to prepare the test cases.

9.31.1 Detailed Description

The unit test program, executes unit tests.

Author

Rikard Lundmark

Definition in file [RunTests.cc](#).

9.32 RunTests.hh File Reference

Header for the unit test program, executes unit tests.

```
#include <assert.h>
#include <iostream>
#include <list>
#include <string>
#include "GenericUnitTest.hh"
#include "FileEventParser_TEST.hh"
#include "EventHit_TEST.hh"
```

Defines

- `#define` [ASSERT_H](#)
Inclusion guard.
- `#define` [IOSTREAM](#)
Inclusion guard.
- `#define` [LIST](#)
Inclusion guard.
- `#define` [STRING](#)
Inclusion guard.
- `#define` [RUNTESTS_H](#)
Inclusion guard.

Functions

- `int` [main](#) ()
The main test function and program entry point.
- `void` [addTests](#) ()
Called by [main\(\)](#) to prepare the test cases.

Variables

- list< [GenericUnitTest](#) * > [myTests](#)

The existing test cases.

9.32.1 Detailed Description

Header for the unit test program, executes unit tests.

Author

Rikard Lundmark

Definition in file [RunTests.hh](#).

9.33 ScorePair.cc File Reference

Source file for the [ScorePair](#) class.

```
#include "ScorePair.hh"
```

9.33.1 Detailed Description

Source file for the [ScorePair](#) class.

Author

Rikard Lundmark

Definition in file [ScorePair.cc](#).

9.34 ScorePair.hh File Reference

Header file for the [ScorePair](#) class.

```
#include <algorithm>
```

Classes

- class [ScorePair](#)

Pair score with A and Z, also implement comparision operator.

Defines

- #define [SCOREPAIR_HH](#)

Inclusion guard.

9.34.1 Detailed Description

Header file for the [ScorePair](#) class.

Author

Rikard Lundmark

Definition in file [ScorePair.hh](#).