geirs2Panic

Richard J. Mathar

Generated by Doxygen 1.8.6

Wed Mar 19 2014 17:22:32

Contents

1 Todo List 1

2 Hierarchical Index 1

2.1 Class Hierarchy . . . 1

3 Class Index 2

3.1 Class List . . . 2

4 File Index 3

4.1 File List . . . 3

5 Class Documentation 4

5.1 Bbox2D Class Reference . . . 4

5.1.1 Detailed Description . . . 6

5.1.2 Constructor & Destructor Documentation . . . 6

5.1.3 Member Function Documentation . . . 6

5.1.4 Member Data Documentation . . . 7

5.2 Circle2D Class Reference . . . 7

5.2.1 Detailed Description . . . 9

5.2.2 Constructor & Destructor Documentation . . . 10

5.2.3 Member Function Documentation . . . 10

5.2.4 Member Data Documentation . . . 10

5.3 FitsImg2Asc Class Reference . . . 10

5.3.1 Detailed Description . . . 11

5.3.2 Constructor & Destructor Documentation . . . 12

5.3.3 Member Function Documentation . . . 12

5.3.4 Member Data Documentation . . . 13

5.4 GeirsPanic Class Reference . . . 13

5.4.1 Detailed Description . . . 15

5.4.2 Constructor & Destructor Documentation . . . 15

5.4.3 Member Function Documentation . . . 15

5.4.4 Member Data Documentation . . . 17

5.5 Histo Class Reference . . . 17

5.5.1 Detailed Description . . . 19

5.5.2 Constructor & Destructor Documentation . . . 19

5.5.3 Member Function Documentation . . . 20

5.5.4 Member Data Documentation . . . 21

5.6 Histos Class Reference . . . 22

5.6.1 Detailed Description . . . 23

5.6.2 Constructor & Destructor Documentation . . . 23

5.6.3 Member Function Documentation . . . 24
5.6.4 Member Data Documentation . . . 25
5.7 Line2D Class Reference . . . 26
5.7.1 Detailed Description . . . 27
5.7.2 Constructor & Destructor Documentation . . . 27
5.7.3 Member Function Documentation . . . 27
5.7.4 Member Data Documentation . . . 28
5.8 PixImage Class Reference . . . 28
5.8.1 Detailed Description . . . 29
5.8.2 Constructor & Destructor Documentation . . . 29
5.8.3 Member Function Documentation . . . 30
5.8.4 Member Data Documentation . . . 31
5.9 Point2D Class Reference . . . 32
5.9.1 Detailed Description . . . 33
5.9.2 Constructor & Destructor Documentation . . . 34
5.9.3 Member Function Documentation . . . 34
5.9.4 Member Data Documentation . . . 37
5.10 RotTrans2D Class Reference . . . 37
5.10.1 Detailed Description . . . 38
5.10.2 Constructor & Destructor Documentation . . . 38
5.10.3 Member Function Documentation . . . 39
5.10.4 Member Data Documentation . . . 39
5.11 Square2D Class Reference . . . 40
5.11.1 Detailed Description . . . 41
5.11.2 Constructor & Destructor Documentation . . . 41
5.11.3 Member Function Documentation . . . 42
5.11.4 Member Data Documentation . . . 44
5.12 Tria2D Class Reference . . . 44
5.12.1 Detailed Description . . . 46
5.12.2 Constructor & Destructor Documentation . . . 46
5.12.3 Member Function Documentation . . . 46
5.12.4 Member Data Documentation . . . 48
5.13 Window Class Reference . . . 48
5.13.1 Detailed Description . . . 51
5.13.2 Constructor & Destructor Documentation . . . 51
5.13.3 Member Function Documentation . . . 52
5.13.4 Member Data Documentation . . . 56
5.14 WindowSet Class Reference . . . 57
5.14.1 Detailed Description . . . 58
5.14.2 Constructor & Destructor Documentation . . . 59

5.14.3 Member Function Documentation . . . 59
5.14.4 Member Data Documentation . . . 60

6 File Documentation **60**

6.1 Bbox2D.cxx File Reference . . . 60
6.2 Bbox2D.h File Reference . . . 61
6.3 Circle2D.cxx File Reference . . . 62
6.4 Circle2D.h File Reference . . . 63
6.5 config.h File Reference . . . 64
6.5.1 Macro Definition Documentation . . . 64
6.6 fitsImg2Asc.cxx File Reference . . . 65
6.6.1 Detailed Description . . . 66
6.6.2 Function Documentation . . . 66
6.7 FitsImg2Asc.cxx File Reference . . . 68
6.8 FitsImg2Asc.h File Reference . . . 68
6.9 fitsImgRot.cxx File Reference . . . 69
6.9.1 Detailed Description . . . 70
6.9.2 Function Documentation . . . 70
6.10 geirs2Panic.cxx File Reference . . . 70
6.10.1 Detailed Description . . . 71
6.10.2 Function Documentation . . . 71
6.11 GeirsPanic.cxx File Reference . . . 74
6.12 GeirsPanic.h File Reference . . . 74
6.13 Histo.cxx File Reference . . . 75
6.13.1 Function Documentation . . . 76
6.14 Histo.h File Reference . . . 76
6.14.1 Function Documentation . . . 77
6.15 Histos.cxx File Reference . . . 77
6.16 Histos.h File Reference . . . 78
6.17 Line2D.cxx File Reference . . . 79
6.18 Line2D.h File Reference . . . 79
6.19 PixImage.cxx File Reference . . . 80
6.20 PixImage.h File Reference . . . 81
6.21 Point2D.cxx File Reference . . . 82
6.21.1 Function Documentation . . . 83
6.22 Point2D.h File Reference . . . 84
6.22.1 Function Documentation . . . 84
6.23 RotTrans2D.cxx File Reference . . . 85
6.24 RotTrans2D.h File Reference . . . 85
6.25 Square2D.cxx File Reference . . . 86

6.26 Square2D.h File Reference . . . 87
6.27 Tria2D.cxx File Reference . . . 88
6.28 Tria2D.h File Reference . . . 89
6.29 Window.cxx File Reference . . . 90
6.29.1 Macro Definition Documentation . . . 91
6.30 Window.h File Reference . . . 91
6.31 WindowSet.cxx File Reference . . . 92
6.32 WindowSet.h File Reference . . . 92

Index **94**

1 Todo List

Member GeirsPanic::exec (bool doTrim, bool doMimax, double north, bool doflip, bool doaltaz, bool doGeo, bool verbose, bool remov)

Divert windows into header units instead inserting the gaps into the single image.

Author

Richard J. Mathar

Member Histo::init (const valarray< float > &arr, const int Nbin, const double range[])

Check that Nbin is larger than zero etc.

Since

2013-01-28
2013-02-22 percentiles added
2013-06-07 NaN's ignored.
2013-12-04 cumulative bins added

Author

Richard J. Mathar

Member Histos::init (vector< valarray< float > > &arr, const int Nbin, const double *range, vector< string > &iname)

Check that Nbin is larger than zero etc.

Since

2013-01-28

Author

Richard J. Mathar

Member PixImage::apply (const RotTrans2D &rt) const

Carry over NaN pixel values (BLANK) to the result.

Since

2013-07-06

2 Hierarchical Index

2.1 Class Hierarchy

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

Bbox2D 4

FitsImg2Asc 10

GeirsPanic 13

Histo 17

Histos 22

Line2D 26

PixImage 28

Point2D 32

Circle2D 7

RotTrans2D 37

Square2D 40

Tria2D 44

Window 48

WindowSet 57

3 Class Index

3.1 Class List

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

Bbox2D
A 2-dimensional rectangular boundary box parallel to the two Cartesian axes 4

Circle2D
A circle represented by center point coordinate and radius 7

FitsImg2Asc 10

GeirsPanic 13

Histo 17

Histos 22

Line2D
An oriented line section represented by the 2-dimensional coordinates of starting and terminating point 26

PixImage
A rectangular 2-dimensional array of pixels with individual values 28

Point2D
A point with 2 coordinates represented in a Cartesian coordinate system 32

RotTrans2D
A rotation followed by a translation 37

Square2D
A square represented by the four vertices of the corners 40

Tria2D
A triangle represented by the Cartesian coordinates of its three vertices 44

Window 48

WindowSet 57

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

Bbox2D.cxx 60

Bbox2D.h 61

Circle2D.cxx 62

Circle2D.h 63

config.h 64

FitsImg2Asc.cxx 68

fitsImg2Asc.cxx
The program fitsImg2Asc converts the image in the primary HDU of a FITS file to an ASCII format 65

FitsImg2Asc.h 68

fitsImgRot.cxx
FitsImgRot is a C++ program which rotates a FITS image by a variable angle about a variable point 69

geirs2Panic.cxx
The program geirs2Panic is a postprocessor for FITS mosaic files that have been generated by GEIRS 70

GeirsPanic.cxx 74

GeirsPanic.h 74

Histo.cxx 75

Histo.h 76

Histos.cxx 77

Histos.h 78

Line2D.cxx 79

Line2D.h 79

PixImage.cxx 80

PixImage.h 81

Point2D.cxx 82

Point2D.h 84

RotTrans2D.cxx 85

RotTrans2D.h 85

Square2D.cxx 86

Square2D.h 87

Tria2D.cxx 88

Tria2D.h 89

Window.cxx 90

Window.h 91

WindowSet.cxx 92

WindowSet.h 92

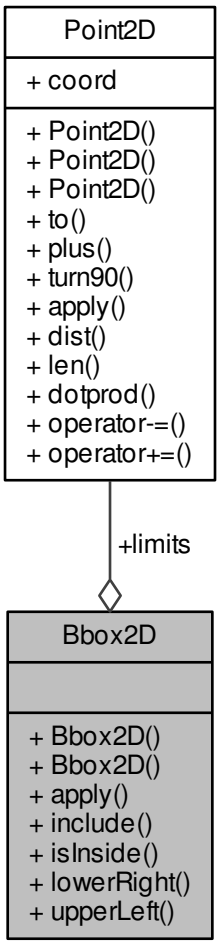
5 Class Documentation

5.1 Bbox2D Class Reference

A 2-dimensional rectangular boundary box parallel to the two Cartesian axes.

`#include <Bbox2D.h>`

Collaboration diagram for Bbox2D:



Public Member Functions

- Bbox2D (const Point2D &ptll, const Point2D &ptur)

  Create a boundary box given two corner points.
- Bbox2D ()
- Bbox2D apply (const RotTrans2D &rt) const

  Compute the bounding box of the quadrangle that is rotated-translated.
- void include (const Point2D &pt)
- bool isInside (const Point2D &pt) const
- Point2D lowerRight () const
- Point2D upperLeft () const

Public Attributes

- Point2D limits [2]

  The two points at the lower left and upper right corner.

5.1.1 Detailed Description

A 2-dimensional rectangular boundary box parallel to the two Cartesian axes.

Since

2013-07-04

Author

Richard J. Mathar

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Bbox2D::Bbox2D (const Point2D & *pt1*, const Point2D & *pt2*)

Create a boundary box given two corner points.

Parameters

pt1	The first corner.
pt2	The second corner.

5.1.2.2 Bbox2D::Bbox2D ( )

Create an invalid bounding box. Default ctor.

5.1.3 Member Function Documentation

5.1.3.1 Bbox2D Bbox2D::apply (const RotTrans2D & *rt*) const

Compute the bounding box of the quadrangle that is rotated-translated.

Parameters

in	*rt*	The rotation-translation to be applied

Returns

The bounding box which includes all 4 rotated corner points of this bounding box.

5.1.3.2 void Bbox2D::include (const Point2D & *pt*)

Extend the bounding box such that a point is inside.

Parameters

in	*pt*	The point that may expand the bounding box.

5.1.3.3 bool Bbox2D::isInside (const Point2D & *pt*) const

Decide whether a point is inside (or on the rim) of the bounding box.

Parameters

in	*pt*	The point to be tested for insidedness.

Returns

True if the point is inside.

5.1.3.4 Point2D Bbox2D::lowerRight ( ) const

Compute the lower right corner of the bounding box.

Returns

The point in the bounding box with maximum x and minimum y coordinate.

5.1.3.5 Point2D Bbox2D::upperLeft ( ) const

Compute the upper left corner of the bounding box.

Returns

The point in the bounding box with minimum x and maximum y coordinate.

5.1.4 Member Data Documentation

5.1.4.1 Point2D Bbox2D::limits[2]

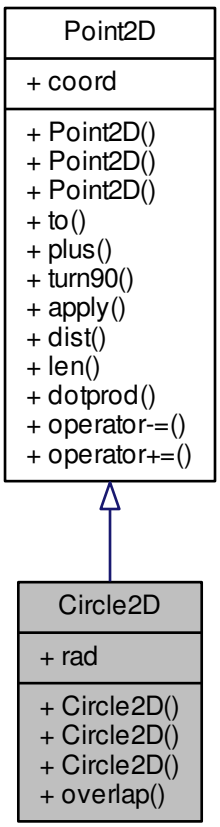
The two points at the lower left and upper right corner.

5.2 Circle2D Class Reference

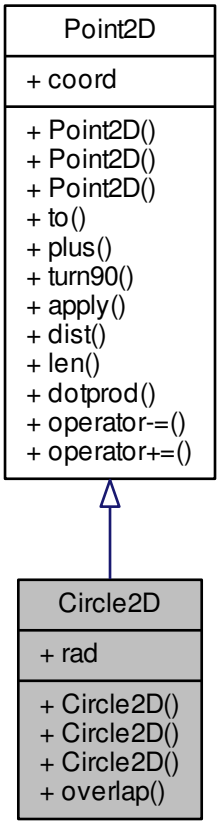
A circle represented by center point coordinate and radius.

`#include <Circle2D.h>`

Inheritance diagram for Circle2D:



Collaboration diagram for Circle2D:



Public Member Functions

- Circle2D (double x, double y, double r)
- Circle2D (const Point2D ctr, double r)
- Circle2D ()
- bool overlap (const Circle2D &oth) const

Public Attributes

- double rad

  The radius.

5.2.1 Detailed Description

A circle represented by center point coordinate and radius.

Since

2013-07-03

Author

Richard J. Mathar

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Circle2D::Circle2D (double *x*, double *y*, double *r*)

Create a circle given its Cartesian coordinates of the center and radius.

Parameters

x	The x coordinate of the center.
y	The y coordinat of the centere
r	The radius.

5.2.2.2 Circle2D::Circle2D (const Point2D *ctr*, double *r*)

Create a circle given the mid point and the radius.

Parameters

ctr	The center point of the cricle.
r	The radius of the circle.

5.2.2.3 Circle2D::Circle2D ( )

Create a circle of zero radius at the origin of coordinates.

5.2.3 Member Function Documentation

5.2.3.1 bool Circle2D::overlap (const Circle2D & *oth*) const

Check whether this circle overlaps with another one.

Parameters

oth	The companion circle which may intersect this one.

Returns

True if the circles overlap. False if they don't overlap or touch only at a point.

5.2.4 Member Data Documentation

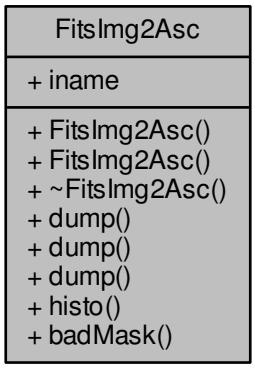
5.2.4.1 double Circle2D::rad

The radius.

5.3 FitsImg2Asc Class Reference

`#include <FitsImg2Asc.h>`

Collaboration diagram for FitsImg2Asc:



Public Member Functions

- FitsImg2Asc (char *fitsInname)

  Constructor.
- FitsImg2Asc (int argc, char *fitsInname[])

  Constructor.
- ~FitsImg2Asc ()

  Destructor.
- void dump ()

  Dump the entire image in ASCII style to cout.
- void dump (int range[4], FITS *ifits)

  Dump the image in ASCII style to cout.
- void dump (string range)

  Dump the image in ASCII style to cout.
- void histo (int Nbin, double range[2], string detsize, const bool doLogHist, const string epsout, const string txtout)

  Generate a gnuplot X11 window with the histogram(s).
- void badMask (double range[2], string detsize, const string fitsout, int flipxy, string txtout, bool iraf)

  Generate a FITS file with a bad mask of pixels.

Public Attributes

- vector< string > iname

  Name of the input file.

5.3.1 Detailed Description

Since

2012-11-25

Author

Richard J. Mathar

5.3.2 Constructor & Destructor Documentation

5.3.2.1 FitsImg2Asc::FitsImg2Asc (char * *fitsIname*)

Constructor.

Parameters

in	*fitsname*	The name of the FITS file to be read.

Since

2012-11-25

Author

Richard J. Mathar

5.3.2.2 FitsImg2Asc::FitsImg2Asc (int *argc,* char * *fitsIname[ ]*)

Constructor.

Parameters

in	*fitsname*	The names of the FITS file to be read.

Since

2013-01-29

Author

Richard J. Mathar

5.3.2.3 FitsImg2Asc::~FitsImg2Asc ( )

Destructor.

5.3.3 Member Function Documentation

5.3.3.1 void FitsImg2Asc::dump ( )

Dump the entire image in ASCII style to cout.

5.3.3.2 void FitsImg2Asc::dump (int *range[4],* FITS * *ifits*)

Dump the image in ASCII style to cout.

Parameters

in	*range*	Four values denoting the pixel range. The lower value of x, the upper value of x (exclusive), the lower value of y and the upper value of y (exclusive) of the pixel region to be plotted.

5.3.3.3 void FitsImg2Asc::dump (string *range*)

Dump the image in ASCII style to cout.

Parameters

in	*range*	Four values denoting the lower value of x, the upper value of x (exclusive), the lower value of y and the upper value of y (exclusive) of the pixel region to be plottet.

5.3.3.4 void FitsImg2Asc::histo (int *Nbin,* double *range[2],* string *detsize,* const bool *doLogHist,* const string *epsout,* const string *txtout*)

Generate a gnuplot X11 window with the histogram(s).

Parameters

in	*Nbin*	The number of bins in the histogram. A value less than zero will trigger default setting of Nbin by the program. The actual number will then be the square root of the maximum number of data values in any of the arrays.
in	*range*	The lower [0] and upper [1] limit of the points on the horizontal axis (ADU) to be used.
in	*detsize*	A string of the form [xmin:xmax,ymin:ymax] selecting a XY ranges of pixels. If empy, the entire frame is examined.
in	*doLogHist*	Print a logarithmic, not a linear vertical scale.
in	*epsout*	If not an empty string, use this file for plotting, not the X11 screen.
in	*txtout*	If not an empty string, use this text file to store the histogram, not the X11 screen.

Since

2013-01-29

5.3.3.5 void FitsImg2Asc::badMask (double *range[2],* string *detsize,* const string *fitsout,* int *flipxy,* string *txtout,* bool *iraf*)

Generate a FITS file with a bad mask of pixels.

Parameters

in	*range*	The lower [0] and upper [1] limit of the points on the horizontal axis (ADU) to be used.
in	*detsize*	A string of the form [xmin:xmax,ymin:ymax] selecting a XY ranges of pixels. If empy, the entire frame is examined.
in	*fitsout*	Name of the FITS output file.
in	*flipxy*	If the LSB bit (bit 0) is set, flip along x coordinate. If the bit 1 is set, flip along y coordinate.
in	*txtout*	Name of the file with the GEIRS specific list of bad pixels. Not used if null or an empty file name.

Since

2013-12-04

5.3.4 Member Data Documentation

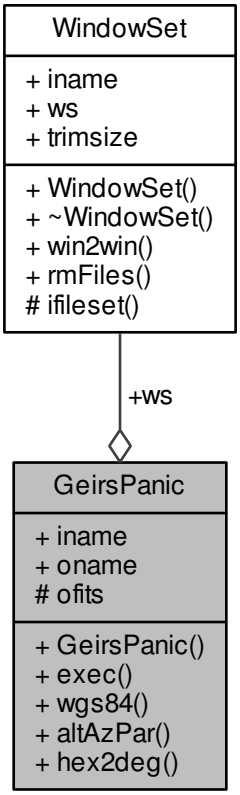
5.3.4.1 vector<string> FitsImg2Asc::iname

Name of the input file.

5.4 GeirsPanic Class Reference

`#include <GeirsPanic.h>`

Collaboration diagram for GeirsPanic:



Public Member Functions

- GeirsPanic (char *fitsInname, const char *fitsOutname, int gap, int chpsz)
- void exec (bool doTrim, bool doMimax, double north, bool doflip, bool doaltaz, bool doGeo, bool verbose, bool remov)

  Modify the FITS header and optionally rotate/flip the image.

Static Public Member Functions

- static void wgs84 (const double lolatlat[3], double xyz[3])

  Convert geographic longitude, latitude and sea leavel to geocentric Cartesian coords.
- static void altAzPar (double ha, double dec, double phi, double aap[3])

  Derive altitude (complementary zenith), azimuth and paralactic angle.
- static double hex2deg (const string &hexstr, bool isdeg)

  Convert a sedecimal-String to degrees.

Public Attributes

- string iname

  Name of the input file.

- string oname

  Name of the output file.
- WindowSet ws

  The set of subwindows.

Protected Attributes

- FITS * ofits

  The FITS file that will be created.

5.4.1 Detailed Description

Since

2012-10-23

Author

Richard J. Mathar

5.4.2 Constructor & Destructor Documentation

5.4.2.1 GeirsPanic::GeirsPanic (char * *fitslname,* const char * *fitsOname,* int *gap,* int *chpsz*)

Parameters

in	*fitsname*	The name of the FITS file to be read.
in	*fitstpl*	The name of the FITS file to be created. This may be the empty string ("") indicating that the file name is to be determined with a standard rule.
in	*gap*	The gap between the chips of the mosaic in units of pixels.
in	*chpsz*	The edge length of the single chip in pixel units. This would be 2048 for Hawaii2 and Hawaii2-RG, or 1024 for Hawaii1.

Since

2012-10-23

Author

Richard J. Mathar

5.4.3 Member Function Documentation

5.4.3.1 void GeirsPanic::exec (bool *doTrim,* bool *doMimax,* double *north,* bool *doflip,* bool *doaltaz,* bool *doGeo,* bool *verbose,* bool *remov*)

Modify the FITS header and optionally rotate/flip the image.

Parameters

in	*doTrim*	If true, remove (blanken) the 4 pixels at the edges of each detector.
in	*doMimax*	If true, write DATAMIN, DATAMAX, DATAMED etc to the header.

in	*gap*	The size of the crossing gap (pixels) in the mosaic.
in	*chpsz*	The size of a single chip (pixels) of the mosaic.
in	*north*	The direction angle of North in the current image (+90=up, +180 = left)
in	*doflip*	If true, flip the image. The main idea is to trigger a swap of the Eastern side relative to North, if previously found to be wrong. There is of course an effect of this option on the direction of North in the final image, unless this was exactly +90 or -90.
in	*doaltaz*	If true, derive ALT, AZ and PARANG from latitude, HA and DEC. Warning: this will only produce correct values if the keywords HA, DEC and GEOPOS_L are degrees in the original header, or HA and DEC are given in hex-decimal colon-separated notation.
in	*doGeo*	If true, add OBSGEO-[XYZ] to the keywords.
in	*verbose*	If true, write more verbose output.
in	*remov*	If true, the initial file (or the initial file set if windowed) is removed on exit. This is always done if requested, and may lead to loss of data if anything is wrong with the new file that is created.

Since

2012-10-18

Todo Divert windows into header units instead inserting the gaps into the single image.

Author

Richard J. Mathar

If any keywords were changed or added, write a HISTORY line keeping track of the template file, and remove the CHECKSUM keyword if present.

5.4.3.2 void GeirsPanic::wgs84 (const double *lolatalt[3],* double *xyz[3]*) [static]

Convert geographic longitude, latitude and sea leavel to geocentric Cartesian coords.

Parameters

in	*lolatalt*	The longitude (lambda, degrees), latitude (phi, degrees) and altitude (meters) of the site. The latitude is the topocentric one, not the geocentric one.
out	*xyz*	The x, y and z coordinates in the Greenwhich centered coordinates (meters).

Since

2012-10-25

5.4.3.3 void GeirsPanic::altAzPar (double *ha,* double *dec,* double *phi,* double *aap[3]*) [static]

Derive altitude (complementary zenith), azimuth and paralactic angle.

Note that this will fail (drastically) if any of the units in the original header for the hour angle (HA), declination (DEC) or latitude (OBSGEO-L) were not degrees.

Parameters

in	*ha*	Hour angle at the epoch (deg)
in	*dec*	Declination at the epoch (deg).
in	*phi*	Observer's latitude (deg).

out	*aap*	The altitude (deg), azimuth (deg, South to West) and parallactic angle (deg)

Since

2012-10-25

5.4.3.4 double GeirsPanic::hex2deg (const string & *hexstr,* bool *isdeg*) [static]

Convert a sedecimal-String to degrees.

Parameters

in	*hexstr*	A string with optional sign, integer number, colon, integer number, colon and integer or floating point number.
in	*isdeg*	If true, assume that the string is in DD:MM:SS.ss format. If false the string is in HH:MM:SS.ss format. This implies an additional factor 15 to move on to degrees. If the string contains a sign (plus or minus), this parameter is ignored and the function assumes degrees.

Returns

The value in degrees.

Since

2012-11-13

5.4.4 Member Data Documentation

5.4.4.1 string GeirsPanic::iname

Name of the input file.

This may also be the stem of the window set, without the _win∗.fits.

5.4.4.2 string GeirsPanic::oname

Name of the output file.

5.4.4.3 WindowSet GeirsPanic::ws

The set of subwindows.

This may be a simple single full frame.

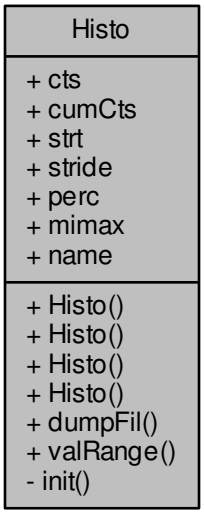
5.4.4.4 FITS∗ GeirsPanic::ofits [protected]

The FITS file that will be created.

5.5 Histo Class Reference

`#include <Histo.h>`

Collaboration diagram for Histo:



Public Member Functions

- Histo ()

  Constructor.
- Histo (const valarray< float > &arr, const int Nbin, const double range[], string &nam)
- Histo (const valarray< float > &arr, const int Nbin, string &nam)

  Constructor.
- Histo (const valarray< float > &arr, string &nam)

  Constructor.
- void dumpFil (const char *fname)

  Create the bins in xy gnuplot format. For each number in the histogram, a short horizontal line is produced, such that concatenation of these horizontal lines with straight line segments produces a step function along the x axis.

Static Public Member Functions

- static int valRange (const valarray< float > &arr, double mimax[2])

  Determine smallest and largest value in all elements of the array.

Public Attributes

- vector< int > cts

  The counts on a per-bin basis.
- vector< int > cumCts

  The cumulative counts on a per-bin basis.
- double strt

  The smallest range of the first bin.
- double stride

The width of each bin.

- double perc [7]

  Percentile abscissas.

- double mimax [3]

  Minimum and maximum value and arithmetic mean.

- string name

  Some kind of file name or tag attached to this statistics.

Private Member Functions

- void init (const valarray< float > &arr, const int Nbin, const double range[])

  Principal part of the constructor.

5.5.1 Detailed Description

Since

2013-01-29

Author

Richard J. Mathar

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Histo::Histo ( )

Constructor.

Empty histogram. The main purpose of this construction is to allow allocations of vectors of histograms.

Since

2013-01-28

Author

Richard J. Mathar

5.5.2.2 Histo::Histo (const valarray< float > & *arr*, const int *Nbin*, const double *range[]*, string & *nam*)

5.5.2.3 Histo::Histo (const valarray< float > & *arr*, const int *Nbin*, string & *nam*)

Constructor.

The range of the bins is automatically set to reach out to the smallest and largest value in the array.

Parameters

arr	The array of values to be binned.
Nbin	The number of bins into which the values are split.

Since

2013-01-28

Author

Richard J. Mathar

5.5.2.4 Histo::Histo (const valarray< float > & *arr,* string & *nam*)

Constructor.

The number of bins is set equal to the square root of the number of values in the array. The range of the bins is automatically set to reach out to the smallest and largest value in the array.

Parameters

arr	The array of values to be binned.

Since

2013-01-28

Author

Richard J. Mathar

5.5.3 Member Function Documentation

5.5.3.1 int Histo::valRange (const valarray< float > & *arr,* double *mimax[2]*) [static]

Determine smallest and largest value in all elements of the array.

NaN-values are not taken into account (ignored). In that sense, this function is an improvement relative to the undefined result if min() or max() of the array are taken as defined in the STL.

Parameters

in	*arr*	The array to be scanned.
out	*mimax*	The minimum in [0], the maximum in [1]. If the array does not contain values that are not NaN's, both values are unchanged.

Returns

The number of non-NaN's in the array.

Since

2013-06-07

5.5.3.2 void Histo::dumpFil (const char * *fname*)

Create the bins in xy gnuplot format. For each number in the histogram, a short horizontal line is produced, such that concatenation of these horizontal lines with straight line segments produces a step function along the x axis.

Parameters

in	*fname*	The file to be generated.

5.5.3.3 void Histo::init (const valarray< float > & *arr,* const int *Nbin,* const double *range[]*) [private]

Principal part of the constructor.

The statistics (minimum and maximum and the percentiles) is computed over the entire range of the arr-values that are not NaN's, not only over the values within the interval specified by the range argument.

Parameters

arr	The array of values to be binned.
Nbin	The number of bins into which the values are split.
range	The minimum and maximum value in the first and last bin.

Todo Check that Nbin is larger than zero etc.

Since

2013-01-28
2013-02-22 percentiles added
2013-06-07 NaN's ignored.
2013-12-04 cumulative bins added

Author

Richard J. Mathar

5.5.4 Member Data Documentation

5.5.4.1 vector<int> Histo::cts

The counts on a per-bin basis.

Provides by its length an explicit count of the bins.

5.5.4.2 vector<int> Histo::cumCts

The cumulative counts on a per-bin basis.

cumCts[i] contains sum_{k=0..n} cts[k].

Since

2013-12-04

5.5.4.3 double Histo::strt

The smallest range of the first bin.

5.5.4.4 double Histo::stride

The width of each bin.

5.5.4.5 double Histo::perc[7]

Percentile abscissas.

[0] the 3 sigma level (left), [1] the 2 sigm left, [2] the 1 sigma left, [3] the median, [4] the 1 sigma right, [5] the 2 sigma right, [6] the 3 sigma right.

5.5.4.6 double Histo::mimax[3]

Minimum and maximum value and arithmetic mean.

Since

2013-06-07

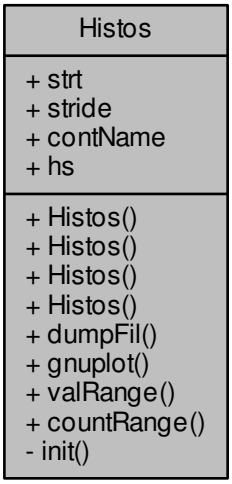
5.5.4.7 string Histo::name

Some kind of file name or tag attached to this statistics.

5.6 Histos Class Reference

`#include <Histos.h>`

Collaboration diagram for Histos:



Public Member Functions

- Histos ()

  Constructor.
- Histos (vector< valarray< float > > &arr, vector< string > &iname)

  Constructor.
- Histos (vector< valarray< float > > &arr, const int Nbin, vector< string > &iname, string &contName)

  Constructor.
- Histos (vector< valarray< float > > &arr, const int Nbin, vector< string > &iname, string &contName, double range[2])

  Constructor.
- void dumpFil (const char *fname)

  Create ASCII file in xy gnuplot format.
- void gnuplot (const char *gplDfile, const char *gplfile, const bool doLogHist, string epsout)

  Generate a gnuplot shell command list.

Static Public Member Functions

- static void valRange (vector< valarray< float > > &arr, double mimax[2])

  Determine smallest and largest value in all elements of the arrays.
- static int countRange (vector< valarray< float > > &arr)

Public Attributes

- double strt
  The smallest range of the first bin.
- double stride
  The width of each bin.
- string contName
  Some type of contents description.
- vector< Histo > hs

Private Member Functions

- void init (vector< valarray< float > > &arr, const int Nbin, const double *range, vector< string > &iname)
  Principal part of the constructor.

5.6.1 Detailed Description

Since

2013-01-29

Author

Richard J. Mathar

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Histos::Histos ( )

Constructor.

Empty histogram. The main purpose of this construction is to allow allocations of vectors of histograms.

Since

2013-01-28

Author

Richard J. Mathar

5.6.2.2 Histos::Histos (vector< valarray< float > > & *arr*, vector< string > & *iname*)

Constructor.

Parameters

arr	The array of values to be binned.
iname	The associated list of names, one per arr component.

Since

2013-01-28

Author

Richard J. Mathar

5.6.2.3 Histos::Histos (vector< valarray< float > > & *arr*, const int *Nbin*, vector< string > & *iname*, string & *conts*)

Constructor.

Parameters

arr	The array of values to be binned.
Nbin	A number of bins to be used. If less than 1, the program depicts a default on its own.
iname	Name of the files associated with the arr.
conts	A global simple name for the entire collection. Mainly for use with display headers and the like.

Since

2013-01-28

Author

Richard J. Mathar

5.6.2.4 Histos::Histos (vector< valarray< float > > & *arr*, const int *Nbin*, vector< string > & *iname*, string & *conts*, double *range[2]*)

Constructor.

Parameters

	arr	The array of values to be binned.
	Nbin	A number of bins to be used. If less than 1, the program depicts a default on its own.
	iname	Name of the files associated with the arr.
	conts	A global simple name for the entire collection. Mainly for use with display headers and the like.
in	*range*	

Since

2013-01-28

Author

Richard J. Mathar

5.6.3 Member Function Documentation

5.6.3.1 void Histos::dumpFil (const char * *fname*)

Create ASCII file in xy gnuplot format.

Parameters

in	*fname*	The file to be generated.

Since

2013-01-29
2013-12-04 cumulative sum column added.

Author

Richard J. Mathar

5.6.3.2 void Histos::gnuplot (const char * *fdname*, const char * *fpltname*, const bool *doLogHist*, string *epsout*)

Generate a gnuplot shell command list.

Parameters

in	*fdname*	The file that contains the data to be plotted.
in	*fpltname*	The file with the gnuplot commands.
in	*doLogHist*	Print a logarithmic, not a linear vertical scale
in	*epsout*	Print diagram into the EPS file, not on screen.

Since

2013-01-29

Author

Richard J. Mathar

5.6.3.3 void Histos::valRange (vector< valarray< float > > & *arr,* double *mimax[2]*) [static]

Determine smallest and largest value in all elements of the arrays.

NaN-values are not taken into account (ignored).

Parameters

in	*arr*	The arrays to be scanned.
out	*mimax*	The minimum in [0], the maximum in [1] If the array arr has zero size, the mimax[] are unchanged on return.

5.6.3.4 int Histos::countRange (vector< valarray< float > > & *arr*) [static]

Parameters

in	*arr*	The arrays to be scanned.

Returns

5.6.3.5 void Histos::init (vector< valarray< float > > & *arr,* const int *Nbin,* const double * *range,* vector< string > & *iname*) [private]

Principal part of the constructor.

Parameters

arr	The array of values to be binned.
Nbin	The number of bins into which the values are split.
range	The minimum and maximum value in the first and last bin.

Todo Check that Nbin is larger than zero etc.

Since

2013-01-28

Author

Richard J. Mathar

5.6.4 Member Data Documentation

5.6.4.1 double Histos::strt

The smallest range of the first bin.

5.6.4.2 double Histos::stride

The width of each bin.

5.6.4.3 string Histos::contName

Some type of contents description.

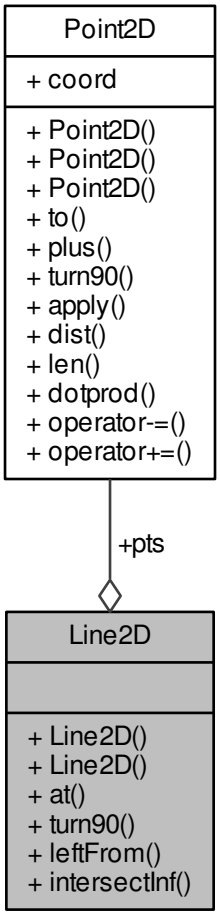
5.6.4.4 vector<Histo> Histos::hs

5.7 Line2D Class Reference

An oriented line section represented by the 2-dimensional coordinates of starting and terminating point.

`#include <Line2D.h>`

Collaboration diagram for Line2D:



Public Member Functions

- Line2D (const Point2D &strt, const Point2D &fini)
- Line2D ()

- Point2D at (double t) const
- Point2D turn90 () const
- bool leftFrom (const Point2D &pt) const
- Point2D intersectInf (const Line2D &oth) const

Public Attributes

- Point2D pts [2]
  The terminal points of start and end.

5.7.1 Detailed Description

An oriented line section represented by the 2-dimensional coordinates of starting and terminating point.

Since

2013-07-03

Author

Richard J. Mathar

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Line2D::Line2D (const Point2D & *strt*, const Point2D & *fini*)

Create a line that connects two given points.

Parameters

strt	The starting point.
fini	The terminal point.

5.7.2.2 Line2D::Line2D ( )

Create a line of zero length at the origin.

5.7.3 Member Function Documentation

5.7.3.1 Point2D Line2D::at (double *t*) const

Determine the point at line parameter t.

Parameters

t	The line parameter.

Returns

A point at distance t relative to the starting point. The point is the starting point if t=0 and the terminal point if t=1.

5.7.3.2 Point2D Line2D::turn90 ( ) const

Create a vector (i.e, a direction) with an orientation turned 90 degrees ccw.

5.7.3.3 bool Line2D::leftFrom (const Point2D & *pt*) const

Decide whether a point is to the left or to the right of the line (the line considered of infinite length)

Parameters

pt	The pivotal point.

Returns

True if the point to the left or on the line.

5.7.3.4 Point2D Line2D::intersectInf (const Line2D & *oth*) const

Determine the point of intersection between this infinite line and another infinite line.

Parameters

oth	The other line (virtually extended both ways to infinity)

Returns

The point of intersection.

5.7.4 Member Data Documentation

5.7.4.1 Point2D Line2D::pts[2]

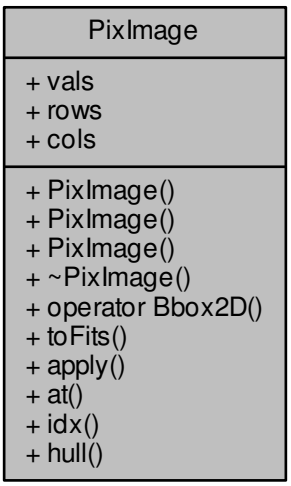
The terminal points of start and end.

5.8 PixImage Class Reference

A rectangular 2-dimensional array of pixels with individual values.

`#include <PixImage.h>`

Collaboration diagram for PixImage:



Public Member Functions

- PixImage (const int ncols, const int nrows)

  Create an empty image with all values set to zero.
- PixImage ()

  Create an image of zero size (without pixels).
- PixImage (FITS &fit, int slic=0)

  Read the pixel values from a slice of a FITS cube.
- ~PixImage ()
- operator Bbox2D () const

  Generate the bounding box of this pixel array.
- FITS * toFits (const char *foutname) const

  Generate a FITS file with this image as the primary HDU.
- PixImage * apply (const RotTrans2D &rt) const
- double at (const int row, const int col) const

  Read/get the value of the intensity at a specific pixel.
- int idx (const int row, const int col) const

  Obtain the flat index of a (x,y) pair of indices.
- Square2D hull (const RotTrans2D &rt, const int row, const int col, int idxReg[2][2], const Bbox2D &bbrot) const

Public Attributes

- valarray< double > vals

  The values (intensities) within the pixels.
- int rows

  Number of rows.
- int cols

  Number of columns.

5.8.1 Detailed Description

A rectangular 2-dimensional array of pixels with individual values.

Since

2013-07-04

Author

Richard J. Mathar

5.8.2 Constructor & Destructor Documentation

5.8.2.1 PixImage::PixImage (const int *ncols*, const int *nrows*)

Create an empty image with all values set to zero.

Parameters

in	*ncols*	The number of columns in the image. The same as the number of pixels along x.
out	*nrows*	The number of rows in the image. The same as the number of pixels along y.

5.8.2.2 PixImage::PixImage ( )

Create an image of zero size (without pixels).

5.8.2.3 PixImage::PixImage (FITS & *fit,* int *slic* = 0)

Read the pixel values from a slice of a FITS cube.

Parameters

fit	The FITS object with at least one image plane in the Primary HDU.
slic	The 0-based index into the FITS cube.

5.8.2.4 PixImage::~PixImage ( )

Destructor. Release the allocated memory areas.

5.8.3 Member Function Documentation

5.8.3.1 PixImage::operator Bbox2D ( ) const

Generate the bounding box of this pixel array.

Returns

The quadrangular bounding box.

5.8.3.2 FITS * PixImage::toFits (const char * *foutname*) const

Generate a FITS file with this image as the primary HDU.

Parameters

in	*foutname*	

Returns

The FITS file with a dummy header.

Since

2013-07-06

5.8.3.3 PixImage * PixImage::apply (const RotTrans2D & *rt*) const

Generate a quadrangle of pixels by rotation and translation of a indexed pixel.

Todo Carry over NaN pixel values (BLANK) to the result.

Since

2013-07-06

5.8.3.4 double PixImage::at (const int *row,* const int *col*) const

Read/get the value of the intensity at a specific pixel.

Parameters

in	*row*	
in	*col*	

Returns

The current contents at that pixel

5.8.3.5 int PixImage::idx (const int *row*, const int *col*) const

Obtain the flat index of a (x,y) pair of indices.

Parameters

in	*row*	
in	*col*	

Returns

The 0-based index col+(number of columns)∗row.

Since

2013-07-06

5.8.3.6 Square2D PixImage::hull (const RotTrans2D & *rt*, const int *row*, const int *col*, int *idxReg[2][2]*, const Bbox2D & *bbrot*) const

Generate a hull (an encompassing integer rectangle) of the image of a pixel

Parameters

in	*rt*	The rotation-translation to be applied to the pixel.
in	*row*	The row index of the pixel.
in	*col*	The column index of the pixel.
in	*bbrot*	The boundary box of the rotated entire original array.
out	*idxReg*	The index region of the image of the pixel under the operation. idxReg[0][] refers to the lower left corner, idxReg[1][] to the upper right. idxReg[][0] refers to the x coordinates, idxReg[][1] to the y coordinates. idx[1][] is non-incluse (as usual in C/C++/Java): The pixel (row,col) will cover parts of idxReg[0][0]..idxReg[0][1]-1 along x and idxReg[1][0]..idxRe[1][1]-1 along y after rotation.

Returns

The rotated pixel in the original coordinate system.

5.8.4 Member Data Documentation

5.8.4.1 valarray<double> PixImage::vals

The values (intensities) within the pixels.

5.8.4.2 int PixImage::rows

Number of rows.

5.8.4.3 int PixImage::cols

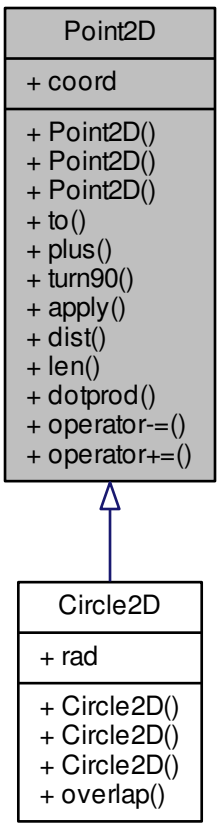
Number of columns.

5.9 Point2D Class Reference

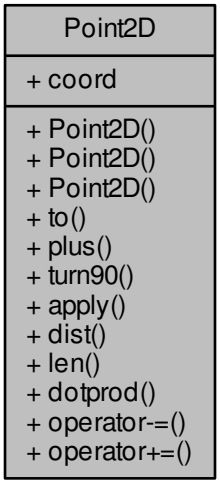
A point with 2 coordinates represented in a Cartesian coordinate system.

`#include <Point2D.h>`

Inheritance diagram for Point2D:



Collaboration diagram for Point2D:



Public Member Functions

- Point2D (double x, double y)
- Point2D (const double xandy[2])
- Point2D ()
- Point2D to (const Point2D &oth) const
- Point2D plus (const Point2D &oth) const

  Add this point (as a vector) to another point.
- Point2D turn90 () const
- Point2D apply (const RotTrans2D &rt) const
- double dist (const Point2D &oth) const
- double len () const
- double dotprod (const Point2D &oth) const
- Point2D & operator-= (const Point2D &rhs)

  Subtract another point, interpreting both points as vectors.
- Point2D & operator+= (const Point2D &rhs)

  Add another point, interpreting both points as vectors.

Public Attributes

- double coord [2]

  The two Cartesian coordinates x and y of the point.

5.9.1 Detailed Description

A point with 2 coordinates represented in a Cartesian coordinate system.

Since

2013-07-03

Author

Richard J. Mathar

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Point2D::Point2D (double *x*, double *y*)

Create a point given its two Cartesian coordinates.

Parameters

x	The x coordinate.
y	The y coordinate

5.9.2.2 Point2D::Point2D (const double *xandy[2]*)

Create a point given its coordinate vector.

Parameters

xandy	The x coordinate in [0], the y coordinate in [1].

5.9.2.3 Point2D::Point2D ( )

Create a point at the origin of coordinates.

5.9.3 Member Function Documentation

5.9.3.1 Point2D Point2D::to (const Point2D & *oth*) const

Compute the direction from this point to another. Equivalent description would be to constructe the vector to the other point minus this.

Parameters

oth	The other point to connect this to.

5.9.3.2 Point2D Point2D::plus (const Point2D & *oth*) const

Add this point (as a vector) to another point.

Parameters

oth[in]	The other point to attach this to.

Returns

The point which has x and y coordinates which are sums of the x and y coordinates of this and oth.

5.9.3.3 Point2D Point2D::turn90 ( ) const

Create a point that is rotated by 90 degrees ccw .

Returns

A point with coordinates (-y,+x) if this =(+x,+y).

5.9.3.4 Point2D Point2D::apply (const RotTrans2D & *rt*) const

Apply a rotation-translation to this point. The effect is the same as calling RotTrans2D::apply(const Point2D &) .

Parameters

in	*rt*	The rotation-translation operator to be applied.

Returns

A point with coordinates rotated then translated by rt.

5.9.3.5 double Point2D::dist (const Point2D & *oth*) const

Compute the distance to another point.

Parameters

oth	The other reference point.

Returns

The distance between this and oth. Square root of the sum of the squared differences in both Cartesian coordinates.

5.9.3.6 double Point2D::len ( ) const

Compute the length of the vector from the origin to this point.

Returns

The distance between this and the origin.

5.9.3.7 double Point2D::dotprod (const Point2D & *oth*) const

Compute the dot product of this point (as a vector) with another.

Returns

The dot product this.x ∗oth.x +this.y∗oth.y.

5.9.3.8 Point2D & Point2D::operator-= (const Point2D & *rhs*)

Subtract another point, interpreting both points as vectors.

Conceptionally this is (up to a sign flip or up to a role reversal) the same as the function to(const Point2D &). So the same result would be obtained by the call rhs.to(∗this).

Parameters

rhs	The point to be subtracted.

Returns

A point where coordinates are individually reduced by those of rhs.

5.9.3.9 Point2D & Point2D::operator+= (const Point2D & *rhs*)

Add another point, interpreting both points as vectors.

Parameters

rhs	The point to be added.

Returns

A point where coordinates are individually extended by those of rhs.

5.9.4 Member Data Documentation

5.9.4.1 double Point2D::coord[2]

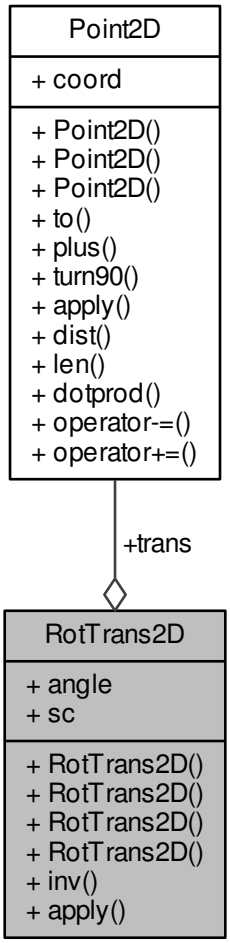
The two Cartesian coordinates x and y of the point.

5.10 RotTrans2D Class Reference

A rotation followed by a translation.

`#include <RotTrans2D.h>`

Collaboration diagram for RotTrans2D:



Public Member Functions

- RotTrans2D (double ang, double tx, double ty)
- RotTrans2D (double ang, const Point2D &t)
- RotTrans2D (double ang, const Point2D &fixpt, const Point2D &t)
- RotTrans2D ()
- RotTrans2D inv () const

  Generate the inverse operation of this.
- Point2D apply (const Point2D &pt) const

Public Attributes

- double angle

  The angle of rotation (radians).
- Point2D trans

  The translation vector.
- double sc [2]

  The sine and cosine of the angle.

5.10.1 Detailed Description

A rotation followed by a translation.

- The class represents an active ccw rotation around (0,0) specified by an angle followed by a translation specified by its 2 delta coordinates.

A rotation around another fixed point followed by a translation is also represented by this class, but mapped on a rotation around (0,0).

Note that the two components of the transformation, the rotation and translation, are not commutative operations in general.

Note also that the translation vector is to be interpreted in the original coordinate system (as indicated by *active*).

Since

2013-07-04

Author

Richard J. Mathar

5.10.2 Constructor & Destructor Documentation

5.10.2.1 RotTrans2D::RotTrans2D (double *ang*, double *x*, double *y*)

Create a rotation-translation given the angle and translation vector.

Parameters

ang	The angle (ccw, radians)
x	The x coordinate of the translation.

y	The y coordinate of the translation

5.10.2.2 RotTrans2D::RotTrans2D (double *ang,* const Point2D & *t*)

Create a rotation-translation given the angle and translation vector.

Parameters

ang	The angle (ccw, radians)
t	The translation vector represented by a point in the original coordinates.

5.10.2.3 RotTrans2D::RotTrans2D (double *ang,* const Point2D & *fixpt,* const Point2D & *t*)

Create a rotation-translation given the angle and translation vector. The initial rotation is specified by an angle and a fixed point of the rotation.

Parameters

ang	The angle (ccw, radians)
fixpt	The fixed point for the rotation.
t	The translation vector represented by a point in the original coordinates.

5.10.2.4 RotTrans2D::RotTrans2D ( )

Create a dummy rotation-translation (which is not rotating or translating). This results in a rotation by an angle of zero and translation by a vector of zero length.

5.10.3 Member Function Documentation

5.10.3.1 RotTrans2D RotTrans2D::inv ( ) const

Generate the inverse operation of this.

Returns

The left inverse operation M', defined by M'∗this = identity, or the right inverse operation M' defined by this∗-M'=identity. In the operator products, the latter one (the one to the right of the multiplication symbol) is executed first.

5.10.3.2 Point2D RotTrans2D::apply (const Point2D & *pt*) const

Apply the transformation to a point.

Parameters

pt	The point to be rotated-translated.

Returns

The rotated coordinates.

5.10.4 Member Data Documentation

5.10.4.1 double RotTrans2D::angle

The angle of rotation (radians).

Points are rotated ccw in the 2D plane by this angle.

5.10.4.2 Point2D RotTrans2D::trans

The translation vector.

5.10.4.3 double RotTrans2D::sc[2]

The sine and cosine of the angle.

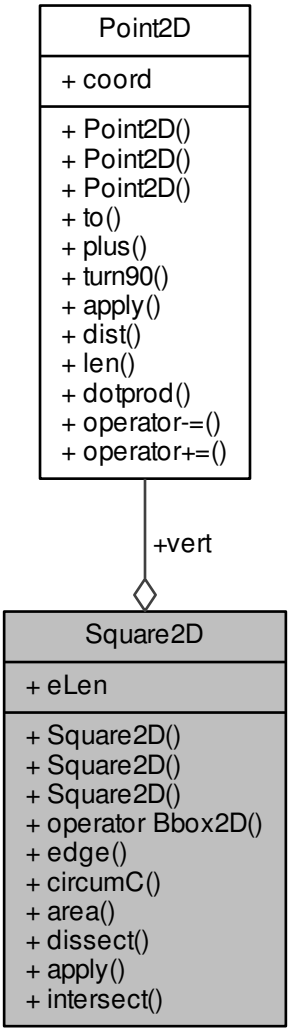
Stored separately to avoid re-computation of these two values if the rotation is applied often.

5.11 Square2D Class Reference

A square represented by the four vertices of the corners.

`#include <Square2D.h>`

Collaboration diagram for Square2D:



Public Member Functions

- Square2D (const Point2D &pt1, const Point2D &pt2)

  Create a planar square given the first two adjacent vertices.
- Square2D (const Point2D &pt1, const Point2D &pt2, const Point2D &pt3, const Point2D &pt4)

  Create a planar square given all four vertices.
- Square2D ()
- operator Bbox2D () const

  Generate the bounding box of this square.
- Line2D edge (int no) const
- Circle2D circumC () const
- double area () const
- Tria2D ∗ dissect () const
- Square2D apply (const RotTrans2D &rt) const

  Compute the square that is rotated-translated.
- vector< Tria2D > intersect (const Square2D &oth) const

  Compute the common area (intersection) between this and another square.

Public Attributes

- Point2D vert [4]

  The points at the vertices.
- double eLen

  Edge length.

5.11.1 Detailed Description

A square represented by the four vertices of the corners.

Since

2013-07-03

Author

Richard J. Mathar

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Square2D::Square2D (const Point2D & *pt1*, const Point2D & *pt2*)

Create a planar square given the first two adjacent vertices.

The third and fourth vertex are defined by orthogonal extension off the baseline between the first and second vertex.

Parameters

pt1	The first vertex.
pt2	The second vertex.

5.11.2.2 Square2D::Square2D (const Point2D & *pt1*, const Point2D & *pt2*, const Point2D & *pt3*, const Point2D & *pt4*)

Create a planar square given all four vertices.

The four vertices are supposed to be given in ccs order. No checking on the length and mutual orthogonality of the edges is done!

Parameters

pt1	The first vertex.
pt2	The second vertex.
pt3	The third vertex.
pt4	The fourth vertex.

5.11.2.3 Square2D::Square2D ( )

Create an empty square with four vertices all at the origin.

5.11.3 Member Function Documentation

5.11.3.1 Square2D::operator Bbox2D ( ) const

Generate the bounding box of this square.

Returns

The quadrangular bounding box.

5.11.3.2 Line2D Square2D::edge (int *no*) const

Create a line that connects vertex number 'no' with the next vertex along the edge.

Parameters

no	The edge number in the range 0 to 3.

Returns

The line that connects point no with point no+1. Edge 0 connects vertex 0 to 1. Edge 1 connects vertex 1 to 2. Edge 2 connects vertex 2 to 3. Edge 3 connects vertex 3 to 0.

5.11.3.3 Circle2D Square2D::circumC ( ) const

Compute the circumcircle.

Returns

The circumcircle. The perimeter of the circle touches all 4 vertices of the Square.

5.11.3.4 double Square2D::area ( ) const

Compute the area.

Returns

The area. Square of the edge length

5.11.3.5 Tria2D * Square2D::dissect ( ) const

Dissect the square into two triangles along a diagonal.

Returns

The two triangles. The orientation of the triangles is individually the same as the orientation of the square. We also guarantee that the first two edges of each triangle are orthogonal, which implies that the circumcircles of these are centered at the (common) longest edge.

5.11.3.6 Square2D Square2D::apply (const RotTrans2D & *rt*) const

Compute the square that is rotated-translated.

Parameters

in	*rt*	The rotation-translation to be applied

Returns

The square defined by individual rotation of all 4 vertices of this.

5.11.3.7 vector< Tria2D > Square2D::intersect (const Square2D & *oth*) const

Compute the common area (intersection) between this and another square.

Returns

The vector of mutually non-intersecting pieces of the common area of both polygons. If there is no overlap, the vector that is returned is empty.

5.11.4 Member Data Documentation

5.11.4.1 Point2D Square2D::vert[4]

The points at the vertices.

5.11.4.2 double Square2D::eLen

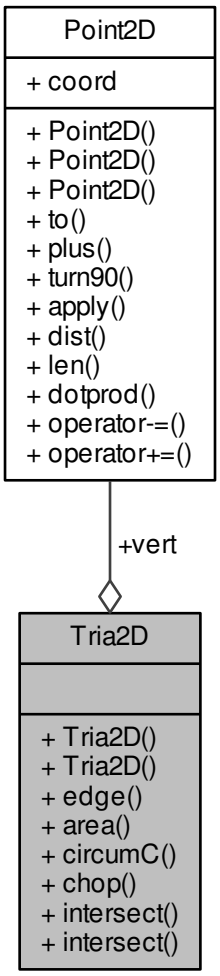
Edge length.

5.12 Tria2D Class Reference

A triangle represented by the Cartesian coordinates of its three vertices.

`#include <Tria2D.h>`

Collaboration diagram for Tria2D:



Public Member Functions

- Tria2D (const Point2D &pt1, const Point2D &pt2, const Point2D &pt3)
- Tria2D ()
- Line2D edge (int no) const
- double area () const
- Circle2D circumC () const
- vector< Tria2D > chop (const Line2D &lin) const
- vector< Tria2D > intersect (const Tria2D &oth) const

  Compute the common portion (intersection) between this and another triangle.
- vector< Tria2D > intersect (const Square2D &oth) const

  Compute the common area (intersection) between this and a square.

Public Attributes

- Point2D vert [3]

The three points at the vertices.

5.12.1 Detailed Description

A triangle represented by the Cartesian coordinates of its three vertices.

Since

2013-07-03

Author

Richard J. Mathar

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Tria2D::Tria2D (const Point2D & *pt1*, const Point2D & *pt2*, const Point2D & *pt3*)

Create a planar triangle given its three vertices.

Parameters

pt1	The first vertex.
pt2	The second vertex.
pt3	The third vertex.

5.12.2.2 Tria2D::Tria2D ( )

Create an empty triangle with three vertices all at the origin.

5.12.3 Member Function Documentation

5.12.3.1 Line2D Tria2D::edge (int *no*) const

Create a line that connects vertex number 'no' with the next vertex along the edge.

Parameters

no	The edge number in the range 0 to 2.

Returns

The line that connects point no with point no+1. Edge 0 connects vertex 0 to 1. Edge 1 connects vertex 1 to 2. Edge 2 connects vertex 2 to 0.

5.12.3.2 double Tria2D::area ( ) const

Compute the area of the triangle.

Returns

The area of all points within the triangle.

5.12.3.3 Circle2D Tria2D::circumC ( ) const

Compute the circumcircle.

Returns

The circumcircle. By definition, the perimeter of the circle touches all three vertices of the triangle.

5.12.3.4 vector< Tria2D > Tria2D::chop (const Line2D & *lin*) const

Compute the portion of this that is left from the infinite line lin.

Parameters

lin	The line that cuts the 2D plane in half.

Returns

The vector of mutually non-intersecting pieces of the subarea of this that is to the left from the line.

5.12.3.5 vector< Tria2D > Tria2D::intersect (const Tria2D & *oth*) const

Compute the common portion (intersection) between this and another triangle.

Returns

The vector of mutually non-intersecting pieces of the common area of both triangles. If they do not overlap, the vector is empty.

5.12.3.6 vector< Tria2D > Tria2D::intersect (const Square2D & *oth*) const

Compute the common area (intersection) between this and a square.

Returns

The vector of mutually non-intersecting pieces of the common area of both polygons.. If they do not overlap, the vector is empty.

5.12.4 Member Data Documentation

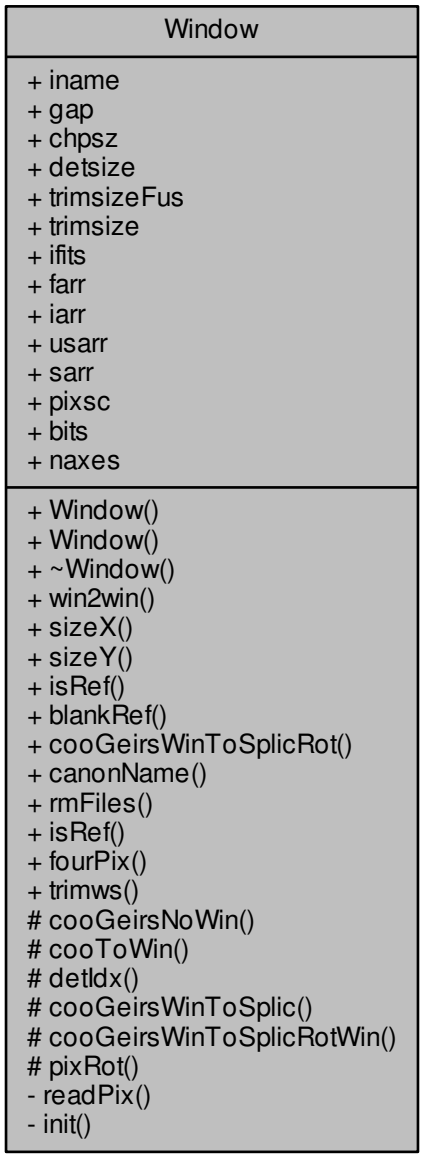
5.12.4.1 Point2D Tria2D::vert[3]

The three points at the vertices.

5.13 Window Class Reference

`#include <Window.h>`

Collaboration diagram for Window:



Public Member Functions

- Window (char *fitsInname, int gp=0, int csiz=2048)
- Window (string fitsInname, int gp=0, int csiz=2048)
- ~Window ()

  Destructor.
- void win2win (const int rotRight90, const bool flip)

  Compute the window location after splicing and optionial flip-rotation.
- int sizeX () const

  The width along x in pixel units.

- int sizeY () const

  The width along y in pixel units.
- bool isRef (int xin, int yin) const

  Decide whether a pixel is one of the reference pixel in one of the 4 frame borders.
- void blankRef (int blank)

  Fill any pixel in one of the four pixel borders (within the 4pixel limit) by the blank.
- void cooGeirsWinToSplicRot (int coo[2], const int rotRight90, const bool flip)

  Convert the coordinate from fused (zero-gap) windowed to global mosaic coordinate (spliced) system.
- string canonName ()

  Generate a <INSTRU>.YYYY-MM-DDTHH-MM-SS.sss.fits string name.
- void rmFiles (bool verbose=false)

  Delete the file that contains the data used on construction.

Static Public Member Functions

- static bool isRef (int xin, int yin, int xchipsize, int ychipsize)

  Decide whether a pixel is one of the reference pixel in one of the 4 frame borders.
- static void fourPix (const string &tsiz, int coo[4], bool relwidth)

  Extract the [xl:xh,yl:yh] four parameters from the bracket.
- static string trimws (const string &instring)

  Strip leading and trailing white space from a string.

Public Attributes

- string iname

  Name of the input file.
- int gap

  Gap (measured in pixels) between the chips of the mosaic.
- int chpsz

  Size (measured in pixels) of the single chip of the mosaic.
- int detsize [4]

  Size (measured in pixels) of the fused array of chips.
- int trimsizeFus [4]

  Window origin and extension in the coordinate system of the fused (gapless) mosaic.
- int trimsize [4]

  Window origin and extension in the coordinate system of the spliced mosaic.
- FITS * ifits

  The FITS file that will be read/scanned.
- valarray< float > farr

  array of pixels in floating point units.
- valarray< int > iarr

  Array of pixels in 32bit integer units.
- valarray< unsigned short > usarr

  Array of pixels in 16bit integer units.
- valarray< unsigned short > sarr

  Array of pixels in 16bit integer units.
- float pixsc

  Pixel scale in units of degrees on the sky.
- long bits

  The indication of image type (integer, short, float...)
- long naxes

  The indication of dimension of the image (2=flat, 3=cubes,...)

Protected Member Functions

- void cooGeirsNoWin (int coo[2])

  Convert the coordinate in the fused (zero-gap) windowed coordinate system to non-windowed.
- void cooToWin (int coo[2])

  Convert the coordinate in non-windowed coordinate system to windowed.
- void detIdx (const int coo[2], int idx[2])

  Compute the chip number in the fused (zero-gap) global mosaic coordinate system.
- void cooGeirsWinToSplic (int coo[2])

  Convert the coordinate from fused (zero-gap) windowed to global mosaic coordinate (spliced) system.
- void cooGeirsWinToSplicRotWin (int coo[2], const int rotRight90, const bool flip)

  Convert the coordinate from fused (zero-gap) windowed to mosaic windowed/rotated coordinate (spliced) system.

Static Protected Member Functions

- static void pixRot (int coo[2], int rotRight90, bool flip)

  Map a pixel in the input rectangular array to a pixel in the output array.

Private Member Functions

- void readPix (bool verbose)

  Read the pixel data of the primary data unit (image).
- void init (bool verbose)

  Scan the bare primary information in the FITS header.

5.13.1 Detailed Description

Since

2012-11-08

Author

Richard J. Mathar

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Window::Window (char * *fitsIname*, int *gp* = 0, int *csiz* = 2048)

Parameters

in	*fitsname*	The name of the FITS file to be read.
in	*gp*	The width of the gap between individual chips in the mosaic in units of pixels.
in	*csiz*	The dimension along each edge of each chip in pixels.

Since

2012-10-23

Author

Richard J. Mathar

5.13.2.2 Window::Window (string *fitsIname*, int *gp* = 0, int *csiz* = 2048)

Parameters

in	*fitsname*	The name of the FITS file to be read.
in	*gp*	The width of the gap between individual chips in the mosaic in units of pixels.
in	*csiz*	The dimension along each edge of each chip in pixels.

Since

2012-10-23

Author

Richard J. Mathar

5.13.2.3 Window::~Window ( )

Destructor.

5.13.3 Member Function Documentation

5.13.3.1 void Window::win2win (const int *rotRight90,* const bool *flip*)

Compute the window location after splicing and optionial flip-rotation.

Parameters

in	*rotRight90*	Number of right-90 rotations requested.
in	*flip*	If true, perform antoher right-left flip around the y-coordinate.

5.13.3.2 int Window::sizeX ( ) const

The width along x in pixel units.

Returns

The horizontal size of the window in pixels. This refers to the original orientation, not the optionally rotated or flipped one.

Since

2012-11-07

5.13.3.3 int Window::sizeY ( ) const

The width along y in pixel units.

Returns

The vertical size of the window in pixels. This refers to the original orientation, not the optionally rotated or flipped one.

Since

2012-11-07

5.13.3.4 bool Window::isRef (int *xin,* int *yin,* int *xchipsize,* int *ychipsize*) [static]

Decide whether a pixel is one of the reference pixel in one of the 4 frame borders.

This supports elimination of the reference pixels on Hawaii2 detectors.

Parameters

in	*xin*	The x coordinate in the range 0 to xchipsize-1 (inclusive) Independent of windowing, this coordinates refers to x=0 as the lower left edge coordinate of the entire mosaic.
in	*yin*	The y coordinate in the range 0 to ychipsize-1 (inclusive) Independent of windowing, this coordinates refers to y=0 as the lower left edge coordinate the entire mosaic.
in	*xchipsize*	The number of pixels along x.
in	*ychipsize*	The number of pixels along y.

Returns

true if (xin,yin) lie in one of the 4-pixel white frames around the full frame.

5.13.3.5 bool Window::isRef (int *xin,* int *yin*) const

Decide whether a pixel is one of the reference pixel in one of the 4 frame borders.

This supports elimination of the reference pixels on Hawaii2 detectors.

Parameters

in	*xin*	The x coordinate in the range 0 to xchipsize-1 (inclusive) Independent of windowing, this coordinates refers to x=0 as the lower left edge coordinate of the entire mosaic.
in	*yin*	The y coordinate in the range 0 to ychipsize-1 (inclusive) Independent of windowing, this coordinates refers to y=0 as the lower left edge coordinate the entire mosaic.

Returns

true if (xin,yin) lie in one of the 4-pixel white frames around the full frame.

5.13.3.6 void Window::blankRef (int *blank*)

Fill any pixel in one of the four pixel borders (within the 4pixel limit) by the blank.

This supports elimination of the reference pixels on Hawaii2 detectors.

Parameters

in	*blank*	The value to replace all reference values.

Since

2012-11-19

5.13.3.7 void Window::cooGeirsWinToSplicRot (int *coo[2],* const int *rotRight90,* const bool *flip*)

Convert the coordinate from fused (zero-gap) windowed to global mosaic coordinate (spliced) system.

Parameters

in,out	*coo*	

Since

2012-11-07

5.13.3.8 void Window::fourPix (const string & *tsiz,* int *coo[4],* bool *relwidths*) [static]

Extract the [xl:xh,yl:yh] four parameters from the bracket.

Parameters

in	*tsiz*	with four integer values, colon and comma-separated. The interpretation for relwidths=true is that the value of ranges in x is from xl (inclusive) to xl+xh (exclusive) and in y from yl (inclusive) to yl+yh (exclusive). The interpretation for relwidths=false is that the value of ranges in x is from xl (inclusive) to xh (exclusive) and in y from yl (inclusive) to yh (exclusive).
out	*coo*	The four coordinates xl, xh, yl, yh. If the format of the string does not provide the two brackets, two colons and comma, the values are not changed.
in	*relwidths*	If true, the 2nd pair of coordinates is interpreted as relative sizes (widths). If false, the pair is in the same absolute coordinate system as the first pair.

Since

2012-11-06

5.13.3.9 string Window::canonName ( )

Generate a <INSTRU>.YYYY-MM-DDTHH-MM-SS.sss.fits string name.

Since

2012-11-20

Author

Richard J. Mathar

5.13.3.10 string Window::trimws (const string & *instring*) [static]

Strip leading and trailing white space from a string.

Parameters

in	*instring*	The initial string from which nonprintable leading and trailing characters or white space (tabs, blanks) ought to be removed.

Returns

The string with leading and trailing blanks, tabs etc removed.

5.13.3.11 void Window::rmFiles (bool *verbose* = false)

Delete the file that contains the data used on construction.

Obviously, this is a risky operation and may lead to loss of data.

Parameters

in	*verbose*	If true, print names of files deleted to stdout.

Since

2012-11-20

Author

Richard J. Mathar

5.13.3.12 void Window::cooGeirsNoWin (int *coo[2]*) [inline], [protected]

Convert the coordinate in the fused (zero-gap) windowed coordinate system to non-windowed.

Parameters

in,out	*coo*	On input, the x and y coordinate in the window and fused coordinate system. On output, the x and y coordinate in the window and fused coordinate system.

5.13.3.13 void Window::cooToWin (int *coo[2]*) [inline],[protected]

Convert the coordinate in non-windowed coordinate system to windowed.

Parameters

in,out	*coo*	On input, the x and y coordinate in the windowed spliced coordinate system. On output, the x and y coordinate in the global spliced coordinate system.

5.13.3.14 void Window::detIdx (const int *coo[2],* int *idx[2]*) [inline],[protected]

Compute the chip number in the fused (zero-gap) global mosaic coordinate system.

Parameters

in	*coo*	
out	*idx*	

Since

2012-11-07

5.13.3.15 void Window::pixRot (int *coo[2],* int *rotRight90,* bool *flip*) [static],[protected]

Map a pixel in the input rectangular array to a pixel in the output array.

Parameters

in,out	*coo*	[0] the x and [1] the y coordinate before and after the rotation. On output, the two coordinates may have picked up any of the 4 sign combinations. So this is a pure rotation on the Gaussian integer coordinate system. No shift is involved to translate the coordinates back to some sort of first quadrant.
in	*rotRight90*	Count of right rotations in units of 90 degrees. This value is used modulo 4, so effectively between 1 and 3. 0 has no effect, unless combined with the flip.
in	*flip*	Request a flip around the y-axis. This equals sign toggling of x-coordinates after any rotation.

5.13.3.16 void Window::cooGeirsWinToSplic (int *coo[2]*) [protected]

Convert the coordinate from fused (zero-gap) windowed to global mosaic coordinate (spliced) system.

Parameters

in,out	*coo*	

Since

2012-11-07

5.13.3.17 void Window::cooGeirsWinToSplicRotWin (int *coo[2],* const int *rotRight90,* const bool *flip*) [protected]

Convert the coordinate from fused (zero-gap) windowed to mosaic windowed/rotated coordinate (spliced) system.

Parameters

in,out	*coo*	

Since

2012-11-07

5.13.3.18 void Window::readPix (bool *verbose*) [private]

Read the pixel data of the primary data unit (image).

Parameters

in	*verbose*	If true, write some output to stdout.

Since

2012-10-18

Author

Richard J. Mathar

5.13.3.19 void Window::init (bool *verbose*) [private]

Scan the bare primary information in the FITS header.

Read INSTRUME to have some input on the mosaic size. Read DETSIZE which is the preferred way to have some input on the mosaic size. Read SAVEAREA (GEIRS specific, deprecated) and TRIMSIZE to determine the window location

Author

Richard J. Mathar

5.13.4 Member Data Documentation

5.13.4.1 string Window::iname

Name of the input file.

The principal use is to look into the primary header for the SAVEARE (TRIMSIZE) keyword that relocates this window in the coordinate system of the mosaic of fused chips.

5.13.4.2 int Window::gap

Gap (measured in pixels) between the chips of the mosaic.

This is considered the same horizontally and vertically.

5.13.4.3 int Window::chpsz

Size (measured in pixels) of the single chip of the mosaic.

This is only considering quadratic chips (same x and y extent)

5.13.4.4 int Window::detsize[4]

Size (measured in pixels) of the fused array of chips.

This is the product of chip count and chpsz in the horizontal and vertical directions. The interpretation is that the xrange is from [0] to [1] (exclusive), the y range from [2] to [3] (exclusive) and [0]=[2]=0 with the usual 0-based C/C++/Java convention.

5.13.4.5 int Window::trimsizeFus[4]

Window origin and extension in the coordinate system of the fused (gapless) mosaic.

The four components of the array are following the same convention as detsize.

5.13.4.6 int Window::trimsize[4]

Window origin and extension in the coordinate system of the spliced mosaic.

The four components of the array are following the same convention as detsize. If some subareas of the window are located in any but the lower-left chip, the components will differ (and be larger than) the associated components in trimsizeFus.

5.13.4.7 FITS* Window::ifits

The FITS file that will be read/scanned.

5.13.4.8 valarray<float> Window::farr

array of pixels in floating point units.

Empty if this is not a floating point type.

5.13.4.9 valarray<int> Window::iarr

Array of pixels in 32bit integer units.

Empty if this is not a integer type.

5.13.4.10 valarray<unsigned short> Window::usarr

Array of pixels in 16bit integer units.

Empty if this is not a unsigned short integer type.

5.13.4.11 valarray<unsigned short> Window::sarr

Array of pixels in 16bit integer units.

Empty if this is not a short integer type.

5.13.4.12 float Window::pixsc

Pixel scale in units of degrees on the sky.

To be used to convert pixels to WCS coordinates.

5.13.4.13 long Window::bits

The indication of image type (integer, short, float...)

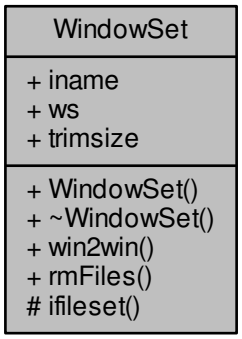
5.13.4.14 long Window::naxes

The indication of dimension of the image (2=flat, 3=cubes,...)

5.14 WindowSet Class Reference

```
#include <WindowSet.h>
```

Collaboration diagram for WindowSet:



Public Member Functions

- WindowSet (const char *fitsInname, int gp=0, int csiz=2048)
- ~WindowSet ()

  Destructor.
- void win2win (const int rotRight90, const bool flip)

  Compute the window location after splicing and optional flip-rotation.
- void rmFiles (bool verbose=false)

  Remove the input files from the file system.

Public Attributes

- string iname

  Name of the input file.
- vector< Window * > ws

  The individual windows.
- int trimsize [4]

  Window superframe origin and extension in the coordinate system of the spliced mosaic.

Static Protected Member Functions

- static vector< string > ifileset (const char *fitsIname)

  Determine the list of all fits file names that match the base name.

5.14.1 Detailed Description

Since

2012-11-08

Author

Richard J. Mathar

5.14.2 Constructor & Destructor Documentation

5.14.2.1 WindowSet::WindowSet (const char * *fitsIname,* int *gp* = 0, int *csiz* = 2048)

Parameters

in	*fitsname*	The name of the FITS file to be read.
in	*gp*	The width of the gap between individual chips in the mosaic in units of pixels.
in	*csiz*	The dimension along each edge of each chip in pixels.

Since

2012-10-23

Author

Richard J. Mathar

5.14.2.2 WindowSet::~WindowSet ( )

Destructor.

5.14.3 Member Function Documentation

5.14.3.1 void WindowSet::win2win (const int *rotRight90,* const bool *flip*)

Compute the window location after splicing and optional flip-rotation.

Parameters

in	*rotRight90*	Number of right-90 rotations requested.
in	*flip*	If true, perform antoher right-left flip around the y-coordinate.

5.14.3.2 void WindowSet::rmFiles (bool *verbose* = false)

Remove the input files from the file system.

This hazardous action removes the FITS files that have been used to define the windows and their contents.

Parameters

in	*verbose*	If true, print each file's name removed to stdout.

Since

2012-11-20

Author

Richard J. Mathar

5.14.3.3 vector< string > WindowSet::ifileset (const char * *fitsIname*) [static],[protected]

Determine the list of all fits file names that match the base name.

Parameters

in	*fitsname*	The basename name of the FITS file to be read.

Returns

The existing files of the format of the argument with suffix .fits, and the existing files with suffices *_wini.fits, with i a set of consecutive increasing integer numbers.

5.14.4 Member Data Documentation

5.14.4.1 string WindowSet::iname

Name of the input file.

The principal use is to look into the primary header for the SAVEARE (TRIMSIZE) keyword that relocates this window in the coordinate system of the mosaic of fused chips.

5.14.4.2 vector<Window *> WindowSet::ws

The individual windows.

At this point an ugly decision is made to collect the vector of pointers to windows, because the Window class contains a pointer to FITS for which no copy ctor exists, so no copy ctor for Window exists (and that would be needed to avoid that at the time the dtor of ws[] goes out of scope the FITS contents is deleted, although we'd like to have acess to the keywords later on...)

5.14.4.3 int WindowSet::trimsize[4]

Window superframe origin and extension in the coordinate system of the spliced mosaic.

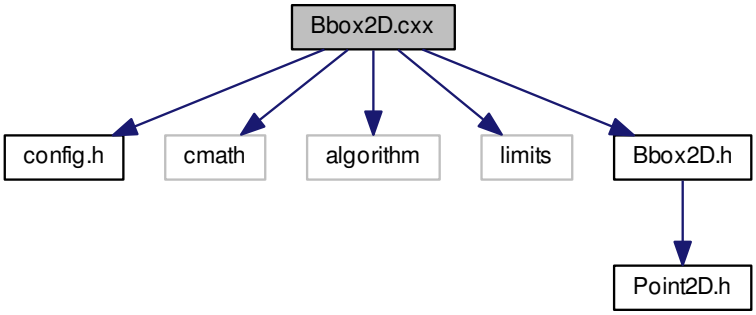
This information is only relevant if one unites all windows in the output, but not if the subareas are individually spread over extension headers.

6 File Documentation

6.1 Bbox2D.cxx File Reference

```
#include "config.h"
#include <cmath>
#include <algorithm>
#include <limits>
#include "Bbox2D.h"
```

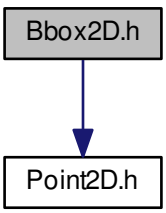
Include dependency graph for Bbox2D.cxx:



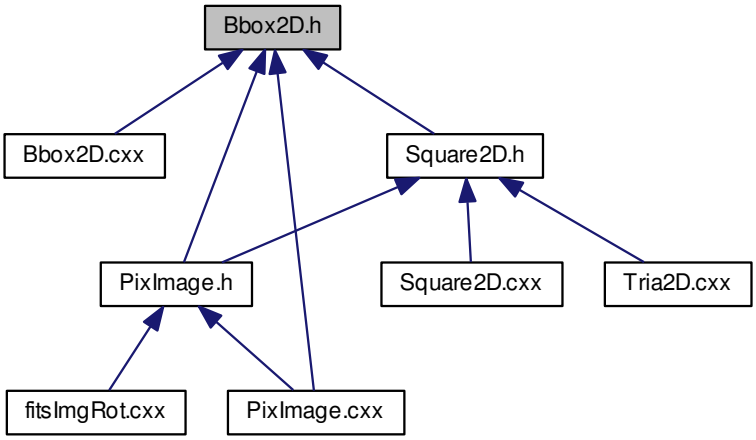
6.2 Bbox2D.h File Reference

```
#include "Point2D.h"
```

Include dependency graph for Bbox2D.h:



This graph shows which files directly or indirectly include this file:



Classes

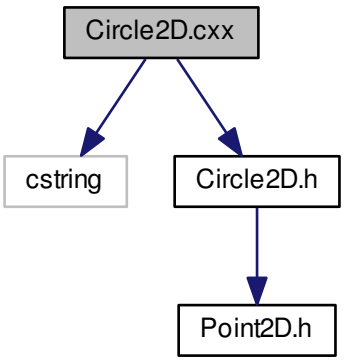
- class Bbox2D

  A 2-dimensional rectangular boundary box parallel to the two Cartesian axes.

6.3 Circle2D.cxx File Reference

```
#include <cstring>
#include "Circle2D.h"
```

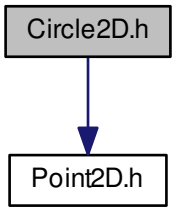
Include dependency graph for Circle2D.cxx:



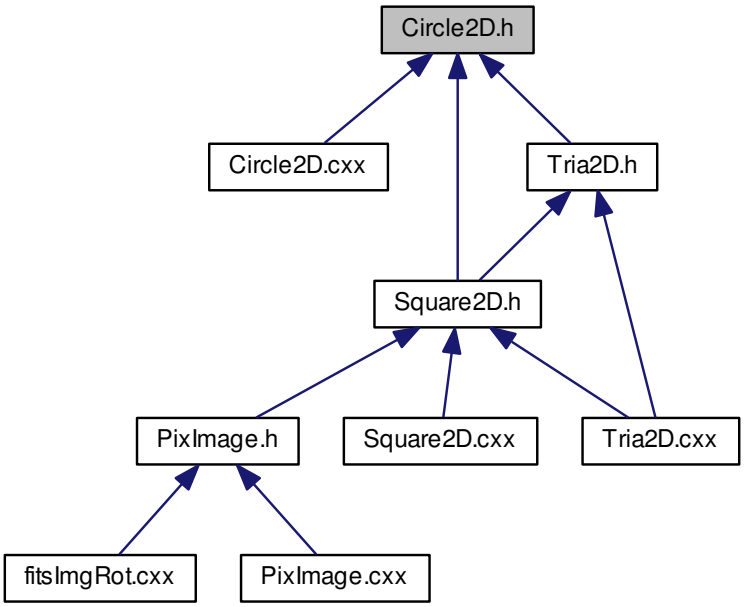
6.4 Circle2D.h File Reference

```
#include "Point2D.h"
```

Include dependency graph for Circle2D.h:



This graph shows which files directly or indirectly include this file:



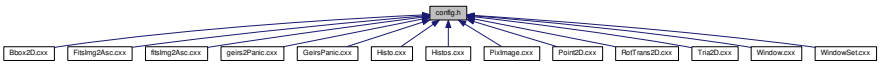
Classes

- class Circle2D

  A circle represented by center point coordinate and radius.

6.5 config.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define HAVE_CMATH
- #define HAVE_CSTRING
- #define HAVE_INTTYPES_H
- #define HAVE_LIBCCFITS
- #define HAVE_LIBM
- #define HAVE_MATH_H
- #define HAVE_MEMCPY
- #define HAVE_MEMORY_H
- #define HAVE_SINCOS
- #define HAVE_STDINT_H
- #define HAVE_STDLIB_H
- #define HAVE_STRINGS_H
- #define HAVE_STRING_H
- #define HAVE_SYS_STAT_H
- #define HAVE_SYS_TYPES_H
- #define HAVE_UNISTD_H
- #define PACKAGE_BUGREPORT
- #define PACKAGE_NAME
- #define PACKAGE_STRING
- #define PACKAGE_TARNAME
- #define PACKAGE_URL
- #define PACKAGE_VERSION
- #define STDC_HEADERS

6.5.1 Macro Definition Documentation

6.5.1.1 #define HAVE_CMATH

6.5.1.2 #define HAVE_CSTRING

6.5.1.3 #define HAVE_INTTYPES_H

6.5.1.4 #define HAVE_LIBCCFITS

6.5.1.5 #define HAVE_LIBM

6.5.1.6 #define HAVE_MATH_H

6.5.1.7 #define HAVE_MEMCPY

6.5.1.8 #define HAVE_MEMORY_H

6.5.1.9 #define HAVE_SINCOS

6.5.1.10 #define HAVE_STDINT_H

6.5.1.11 #define HAVE_STDLIB_H

6.5.1.12 #define HAVE_STRINGS_H

6.5.1.13 #define HAVE_STRING_H

6.5.1.14 #define HAVE_SYS_STAT_H

6.5.1.15 #define HAVE_SYS_TYPES_H

6.5.1.16 #define HAVE_UNISTD_H

6.5.1.17 #define PACKAGE_BUGREPORT

6.5.1.18 #define PACKAGE_NAME

6.5.1.19 #define PACKAGE_STRING

6.5.1.20 #define PACKAGE_TARNAME

6.5.1.21 #define PACKAGE_URL

6.5.1.22 #define PACKAGE_VERSION

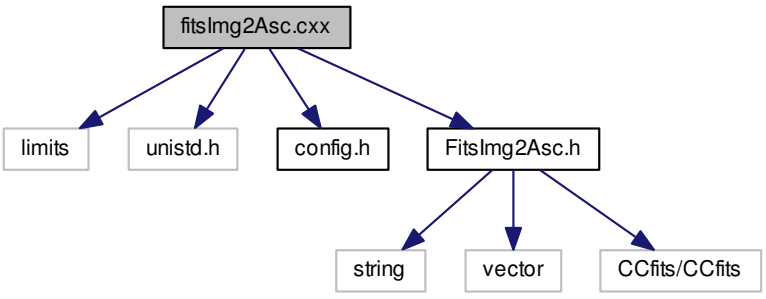
6.5.1.23 #define STDC_HEADERS

6.6 fitsImg2Asc.cxx File Reference

The program fitsImg2Asc converts the image in the primary HDU of a FITS file to an ASCII format.

```
#include <limits>
#include <unistd.h>
#include "config.h"
#include "FitsImg2Asc.h"
```

Include dependency graph for fitsImg2Asc.cxx:



Functions

- void usage (char *argv0)

  Print a usage syntax message detailing the command line.

- int main (int argc, char *argv[])

  The program fitsImg2Asc converts the image in the primary HDU of a FITS file to an ASCII format.

6.6.1 Detailed Description

The program fitsImg2Asc converts the image in the primary HDU of a FITS file to an ASCII format.

6.6.2 Function Documentation

6.6.2.1 void usage (char * *argv0*)

Print a usage syntax message detailing the command line.

Parameters

in	*argv0*	The name of the main program.

Since

2012-10-18

6.6.2.2 int main (int *argc,* char * *argv[]*)

The program fitsImg2Asc converts the image in the primary HDU of a FITS file to an ASCII format.

The input file is an image in the FITS file, of which only the first slice is taken if this is a FITS cube.

- 3D view The standard syntax is

  `fitsImg2Asc fitsIn.fits > fitsout.plt`

  or

  `fitsImg2Asc -r '[xstrt:xend,ystrt:yend]' fitsIn.fits > fitsout.plt`

  which means that the command line argument is the name of the FITS file with the image, and the output is redirected into some other file. This output file will be roughly a factor of 4 larger than the input file, and the program will run slowlier than some people would expect. In almost all cases the field will be too crowded to get useful response times from gnuplot while rotating the image, so the option -r allows to take only a rectangular subarea of the image, where the 4 arguments are 0-based ranges for the two pixel axes.

  The output contains lines with three values, which is the x coordinate of the pixel, the y coordinate of the pixel, and the value in the image at this pixel.

  This ASCII file has been targeted for use with a gnuplot(1) session e.g. like: gnuplot gnuplot> set xlabel "x pixel" gnuplot> set ylabel "y pixel" gnuplot> set zlabel "adu" gnuplot> splot "...." wi li # insert the fitsout.plt file data name here gnuplot> set grid gnuplot> set contour base gnuplot> replot gnuplot> set logscale z gnuplot> replot

  Example: `fitsImg2Asc -r '[200:800,1200:1800]' fitsIn.fits > fitsout.plt`

- Histogram The syntax to generate a gnuplot X11 window or EPS file with a histogram of pixel ADU values is triggered by the -h option as follows:

  `fitsImg2Asc -h [-l] [-N bins] [-m min] [-M max] [-a [xmin:xmax,ymin-:ymax]] [-o fitsout.eps] fitsIn.fits [fitsIn.fits ... ]`

  The option -N followed by a positive integer number can be used to specify the number of bins to be generated. If not provided, the program uses a default.

  The option -m followed by a number is the minimum number on the horizontal axis which delimits the range of ADU's to be shown.

  The option -M followed by a number is the maximum number on the horizontal axis which delimits the range of ADU's to be shown.

  The option -l means a logarithmic scale is used on the vertical axis.

  The option -a followed by four positive integer numbers in brackets selects a quadrangular region (of FITS coordinates) in the image to be scanned. The default is to scan all pixels in the image.

The option -o followed by a file name causes the plot to be moved to a EPS file, not to the screen.

The option -t followed by a file name causes the histogram to be dumped into the named ASCII file.

The other command line arguments are existing fits files which will be read. The gnuplot display shows the count of pixels on a per-file basis which fall into a range of ADU's. There are two variants of showing the result, one with a logarithmic scale for the counts in the bins, one with a linear scale.

Example (which uses the shell's expansion mechanism of file names):

fitsImg2Asc -h -m 0 -M 200 aa000[1-8].fits fitsImg2Asc -h -m '-50' -M 50 -N 100 -a '[5:2044,5:2044]' aa000[1-8].fits

Example (which shows the selection mechanism of moving to a named extension HDU to locate the image):

fitsImg2Asc -h -m '-50' -M 50 -N 100 -a '[5:2044,5:2044]' aa0001.fits'[WIN1]'

- Bad pixel mask The syntax to generate a bad pixel mask is:

```
fitsImg2Asc -b [-a '[xstrt:xend,ystrt:yend]'] [-t textout] [-X] [-Y] -m minADU -M maxADU fitsIn.fits fitsOut.fits
```

The mask is defined by investigating the 2D image provided in the input file fitsIn.fits and writing the bad pixel mask to fitsOut.fits.

The option -a defines a subwindow of pixel ranges to be scanned for out-of-range values. The pixels inside the original window but outside that internal window are a frame which is also regarded of containing only bad pixels. So usually for a single Hawaii-2 RG image one would declare the 4-pixel frame to be bad with -a '[5:2044,5:2044]' .

The option -t followed by the name of a file lets the program generate that file where the bad pixels are listed (one by a line) with their 1-based integer x and y coordinates. (This is also the convention of the fixpix routine of the IRAF reduction. It does not produce the generalized format with four parameters equivalent to xstrt, xend, ystrt, yend to define bad pixel blocks.)

The option -X and similarly the option -Y trigger that the bad pixel mask is generated after flipping the image of fitsIn.fits left-right along x or up-down along y. This helps to generate the mask for images that had another WCS orientation convention than the images the bad pixel mask will be generated for.

The option -m (quasi mandatory) defines the minimum ADU value and the option -M (quasi mandatory) defines the maximum ADU value of pixels in fitsIn.fits supposed to be good.

The option -i causes the bad pixel mask to be written with values of 0 for good pixels and values of 1 for bad pixels (as in some IRAF conventions). The default (without the option) is to write 1's for good values and NaN for bad values, so it could be used for multiplicative application for generic pictures.

The two final arguments are the file names of the input image and of the bad pixel image to be created.

Parameters

in	*argc*	The number of command line switches and arguments.
in	*argv*	The vector of all command line arguments.

Since

2012-11-25
2013-11-11 Support histograms of images in extension headers
2013-12-04 Implemented bad pixel mask creation

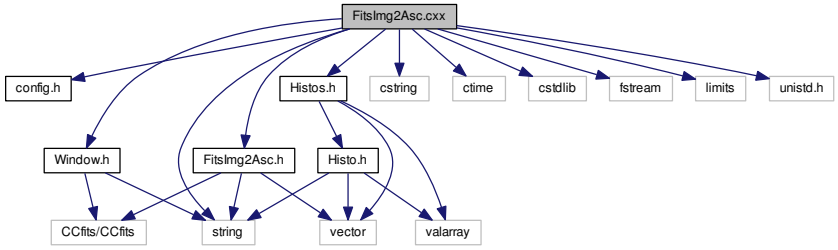
See Also

also http://heasarc.gsfc.nasa.gov/ftools/caldb/help/ftstat.html

6.7 FitsImg2Asc.cxx File Reference

```
#include "config.h"
#include <string>
#include <cstring>
#include <ctime>
#include <cstdlib>
#include <fstream>
#include <limits>
#include <unistd.h>
#include "FitsImg2Asc.h"
#include "Window.h"
#include "Histos.h"
```

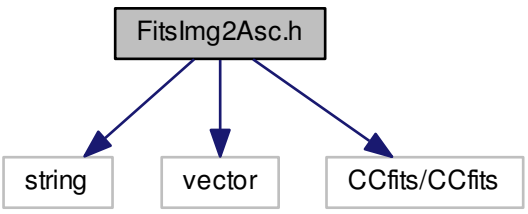
Include dependency graph for FitsImg2Asc.cxx:



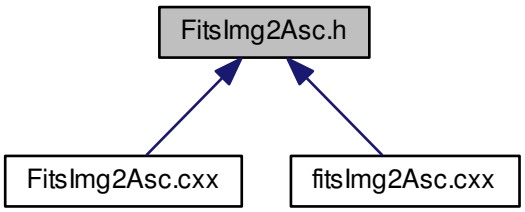
6.8 FitsImg2Asc.h File Reference

```
#include <string>
#include <vector>
#include <CCfits/CCfits>
```

Include dependency graph for FitsImg2Asc.h:



This graph shows which files directly or indirectly include this file:



Classes

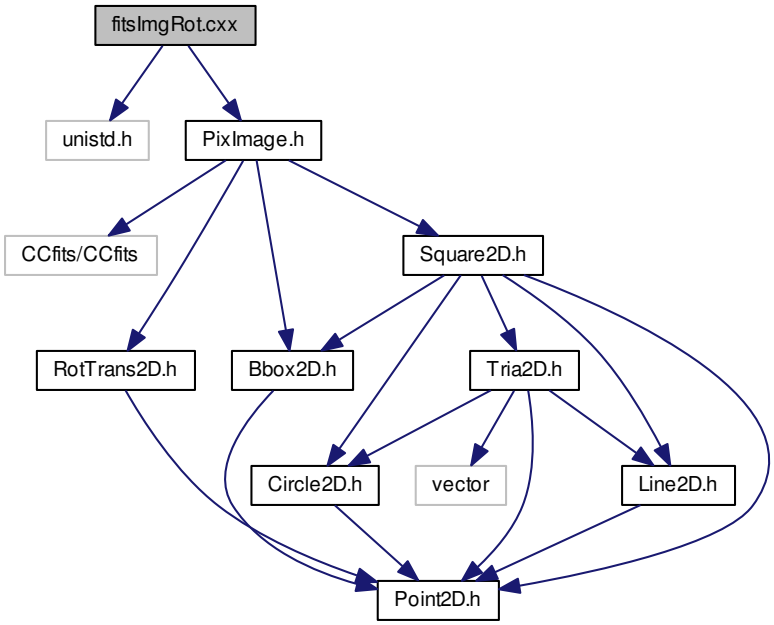
- class FitsImg2Asc

6.9 fitsImgRot.cxx File Reference

fitsImgRot is a C++ program which rotates a FITS image by a variable angle about a variable point.

```
#include <unistd.h>
#include "PixImage.h"
```

Include dependency graph for fitsImgRot.cxx:



Functions

- void usage (char *argv0)

  Print a usage syntax message detailing the command line.

6.9.1 Detailed Description

fitsImgRot is a C++ program which rotates a FITS image by a variable angle about a variable point.

6.9.2 Function Documentation

6.9.2.1 void usage (char * *argv0*)

Print a usage syntax message detailing the command line.

Parameters

in	*argv0*	The name of the main program.

Since

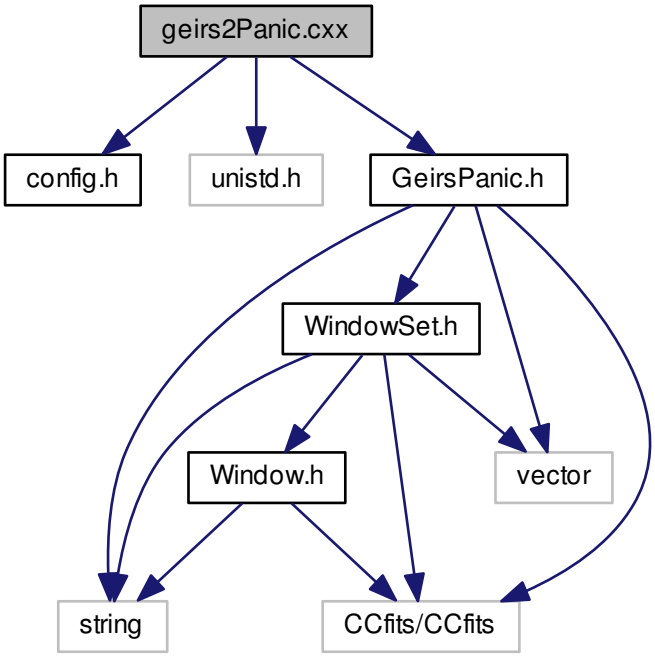
2012-10-18

6.10 geirs2Panic.cxx File Reference

The program geirs2Panic is a postprocessor for FITS mosaic files that have been generated by GEIRS.

```
#include "config.h"
#include <unistd.h>
#include "GeirsPanic.h"
```

Include dependency graph for geirs2Panic.cxx:



Functions

- void usage (char ∗argv0, int gap, int chpsz, double north)

  Print a usage syntax message detailing the command line.
- int main (int argc, char ∗argv[])

  The program is a postprocessor for FITS files of CAHA/PANIC that have been generated by GEIRS.

6.10.1 Detailed Description

The program geirs2Panic is a postprocessor for FITS mosaic files that have been generated by GEIRS.

6.10.2 Function Documentation

6.10.2.1 void usage (char ∗ *argv0,* int *gap,* int *chpsz,* double *north*)

Print a usage syntax message detailing the command line.

Parameters

in	*argv0*	The name of the main program.

Since

2012-10-18

6.10.2.2 int main (int *argc,* char * *argv[]*)

The program is a postprocessor for FITS files of CAHA/PANIC that have been generated by GEIRS.

The main and mandatory arguments are an existing FITS file of the 2x2 mosaic in the glued representation as generated by GEIRS, and the name of a non-existing FITS file that will be generated by the program.

The program supports

- insertion of a cross of undefined values in the mosaic, the definition of the North direction in the image and associated flip/rotation by multiples of 90 degrees to align that direction as good as possible with the up-direction in the image, plus insertion of the linear WCS scale transformation.

- insertion of ALT, AZ and PARANG angles in the output header.

- insertion of OBSGEO-X, OBSGEO-Y and OBSGEO-Z in the output header.

Note that in a field of view stretching across 1 deg, the atmospheric dispersion R = tan(z)*(n-1) (in radians) differs across the field by delta(R) = delta(z)/cos^2(z) *(n-1), where n-1=2e-4 is roughly the index of refraction at the mountain altitude of 2700 m. Since delta(z) is roughly 4000 pixels, the product of both factors amounts to roughly 1 pixel across the field, but that expected linear distortion is not yet included in the WCS keywords of the header in the output.

All of these tasks are actually covered by the instrument pipeline (Ibanez-Mengual et al, "The PANIC software system", SPIE 77402E, July 19 2012, doi:10.1117/12.856029) and do not need to be run on the mountain.

Parameters

in	*argc*	The number of command line switches and arguments.
in	*argv*	The vector of all command line arguments. The standard syntax is `geirs2Panic [-c chipsz] [-g gapPx] [-A] [-t] [-f] [-N degr] [-v] fitsIn [fitsOut.fits]` The options and their arguments are: • `-c cipsz` Sets the edge length of the single chip in the mosaic in units of pixels. This defines the regular distance at which gaps appear in the mosaic. The current default size is 2048. CAHA/LAICA users should set this to 4096. • `-g gapPx` This is the gap between any two chips in the mosaic, in units of pixels. There is a default which is some value of the literature, currently 167. CAHA/LAICA users should set this to 3207 (assuming a pixel scale of 0.225 arcsec/px and 2 times 100 arcsec overlap of the gap with the two adjacent chips). • `-A` Requests that the ALT, AZ and PARANG values are inserted into the header (or replaced if they exist). This uses standard spherical trigonometry and assumes that the latitude, hour angle and declination are all given in degrees in the FITS header. The latitude may either be `TELLAT` (preferred is the standard OBSGEO-B) and the other two are taken from HA and DEC in the FITS header. • `-t` Replaces the 4 pixels around each chip of the mosaic by the `BLANK` value. This is useful to avoid confusion of standard pipeline code that does not know how to interpret these engineering pixels. • `-N degr` Indicates that North is that many degrees in counter-clockwise position from the horizontal axis in the input image. So a value near +90 would be about ideal. This has two implications: The WCS sky transformation will have the correct entries, and values more than 45 degrees away from 90 will trigger a rotation by multiples of 90 degrees on output. • `-f` Indicates that East is on the wrong side of North in the input data. This triggers that an additional flip (alongside any rotations) of the images is done for the output. Note that this is associated with a flip of North and additional changes of the WCS matrix. • `-G` Triggers that the `OBSGEO-X OBSGEO-Y OBSGEO-Z` triple is added to the header on output. This works only if the three `TELLONG TELLAT` and `TELALT` or their standard `OBSGEO-B`, `OBSGEO-L` and `OBSGEO-H` values with are provided in the standard units in the input header. • `-v` generates more comments on standard output about what the program is doing. • `fitsIn` is either (i) a single file with suffix `.fits` which contains either a full frame or a subwindow of the mosaic, or (ii) a basis name for a set of GEIRS window files without their `_win`i<code>.fits parts. Note that any of these windows may already stretch across more than one of the chips of the mosaic, because GEIRS uses windowing in that manner. • `fitsOut.fits` is an optional file name with suffix `.fits` to be chosen for the output file. If this final argument is absent, the program choses a file name which is the concatenation of the instrument name (keyword INSTRUME) and date, keyword DATE-OBS (preferred) or DATE.

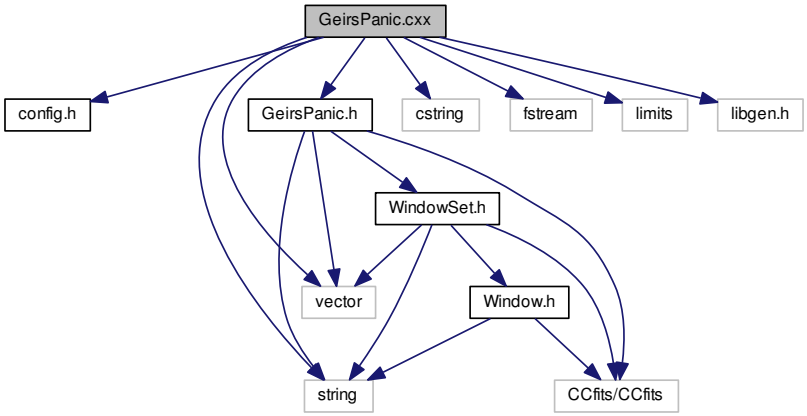
Since

2012-10-23

6.11 GeirsPanic.cxx File Reference

```
#include "config.h"
#include "GeirsPanic.h"
#include <string>
#include <cstring>
#include <vector>
#include <fstream>
#include <limits>
#include <libgen.h>
```

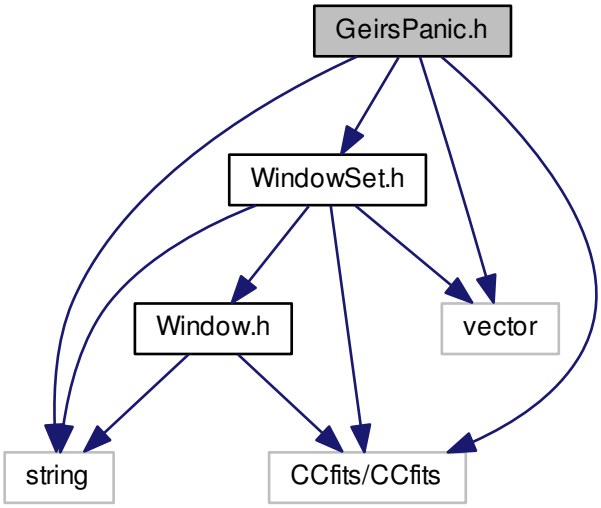
Include dependency graph for GeirsPanic.cxx:



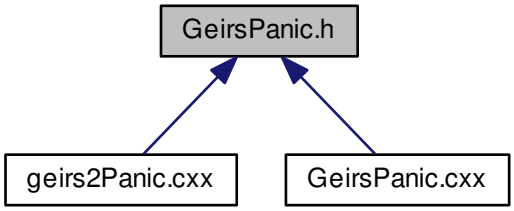
6.12 GeirsPanic.h File Reference

```
#include <string>
#include <vector>
#include <CCfits/CCfits>
#include "WindowSet.h"
```

Include dependency graph for GeirsPanic.h:



This graph shows which files directly or indirectly include this file:



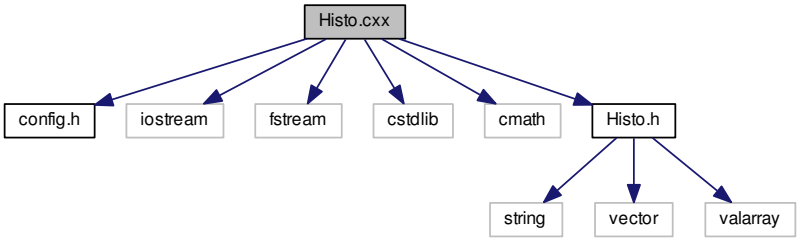
Classes

- class GeirsPanic

6.13 Histo.cxx File Reference

```
#include "config.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include "Histo.h"
```

Include dependency graph for Histo.cxx:



Functions

- int fsort (const void *v1, const void *v2)

  Support of the qsort call in init().

6.13.1 Function Documentation

6.13.1.1 int fsort (const void * *v1*, const void * *v2*)

Support of the qsort call in init().

Parameters

v1	The first constant to be compared.
v2	The second constant to be compared.

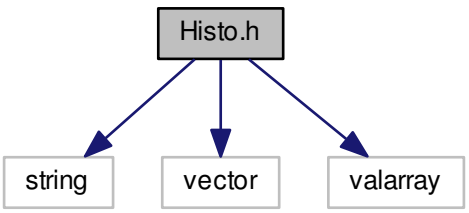
Returns

-1 if v1 is less than v2, +1 if v1 is larger than v2, 0 if both are equal.

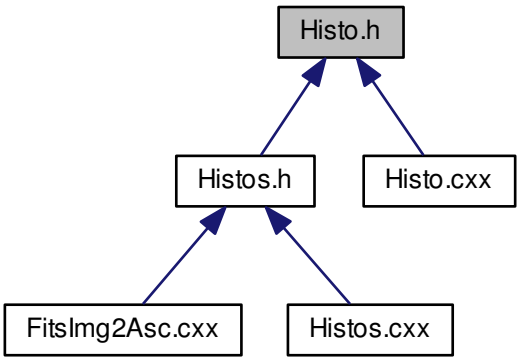
6.14 Histo.h File Reference

```
#include <string>
#include <vector>
#include <valarray>
```

Include dependency graph for Histo.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Histo

Functions

- int fsort (const void *v1, const void *v2)

  Support of the qsort call in init().

6.14.1 Function Documentation

6.14.1.1 int fsort (const void * *v1*, const void * *v2*)

Support of the qsort call in init().

Parameters

v1	The first constant to be compared.
v2	The second constant to be compared.

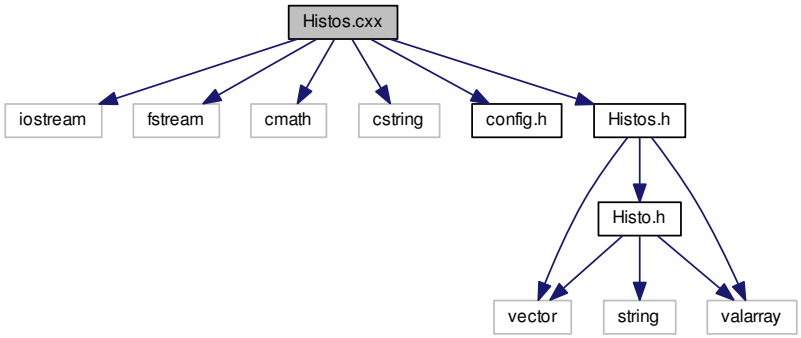
Returns

-1 if v1 is less than v2, +1 if v1 is larger than v2, 0 if both are equal.

6.15 Histos.cxx File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <cstring>
#include "config.h"
#include "Histos.h"
```

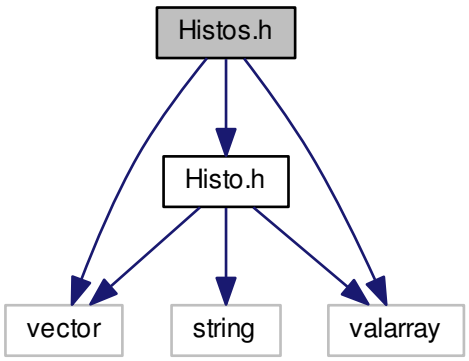
Include dependency graph for Histos.cxx:



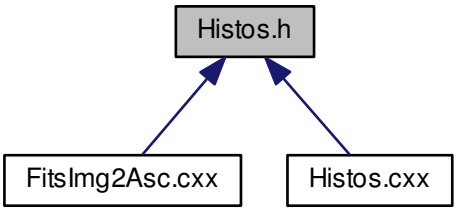
6.16 Histos.h File Reference

```
#include <vector>
#include <valarray>
#include "Histo.h"
```

Include dependency graph for Histos.h:



This graph shows which files directly or indirectly include this file:



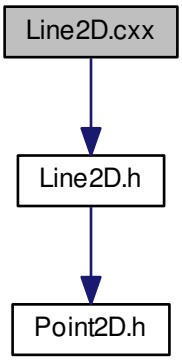
Classes

- class Histos

6.17 Line2D.cxx File Reference

```
#include "Line2D.h"
```

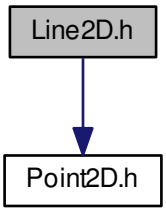
Include dependency graph for Line2D.cxx:



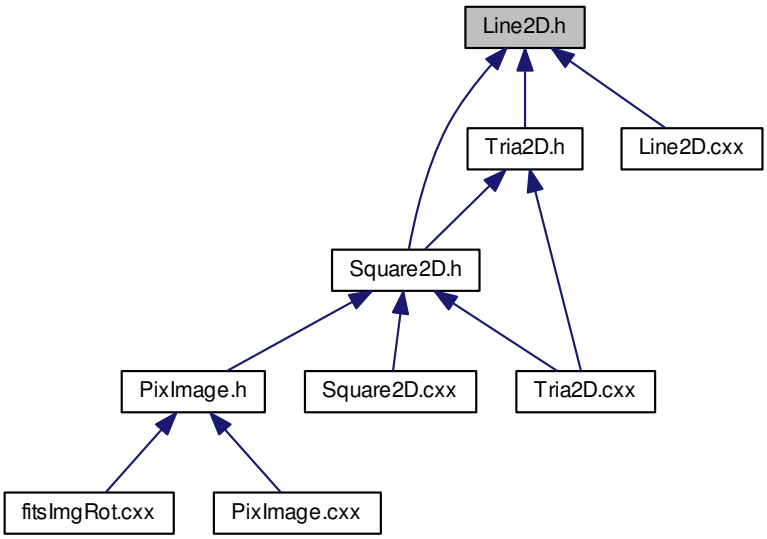
6.18 Line2D.h File Reference

```
#include "Point2D.h"
```

Include dependency graph for Line2D.h:



This graph shows which files directly or indirectly include this file:



Classes

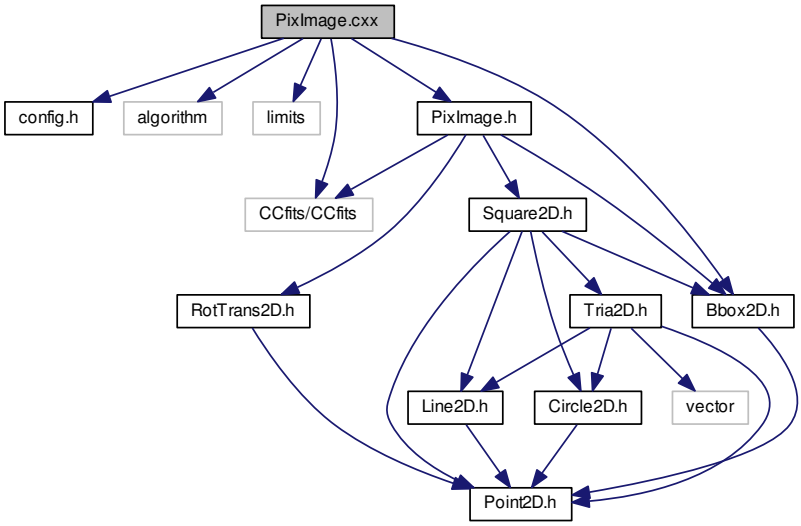
- class Line2D

  An oriented line section represented by the 2-dimensional coordinates of starting and terminating point.

6.19 PixImage.cxx File Reference

```
#include "config.h"
#include <algorithm>
#include <limits>
#include <CCfits/CCfits>
#include "PixImage.h"
#include "Bbox2D.h"
```

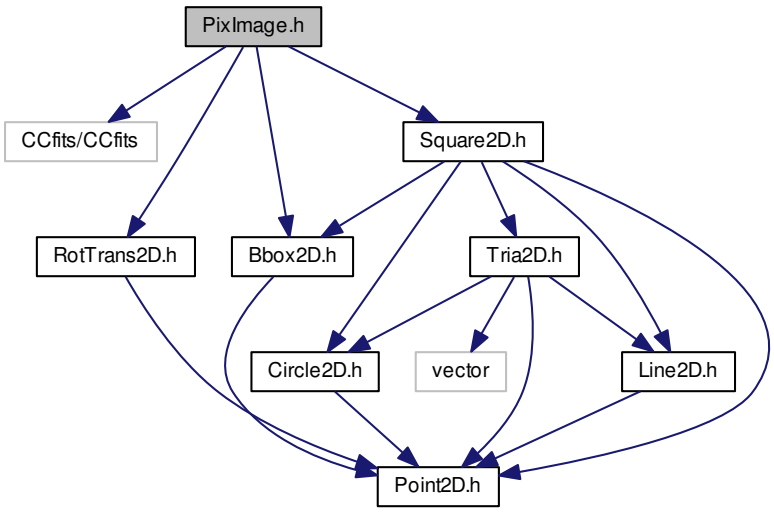
Include dependency graph for PixImage.cxx:



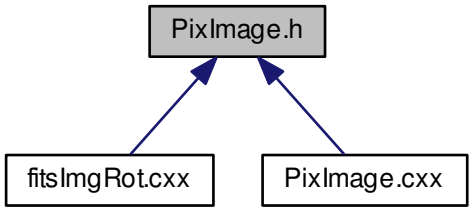
6.20 PixImage.h File Reference

```
#include <CCfits/CCfits>
#include "RotTrans2D.h"
#include "Bbox2D.h"
#include "Square2D.h"
```

Include dependency graph for PixImage.h:



This graph shows which files directly or indirectly include this file:



Classes

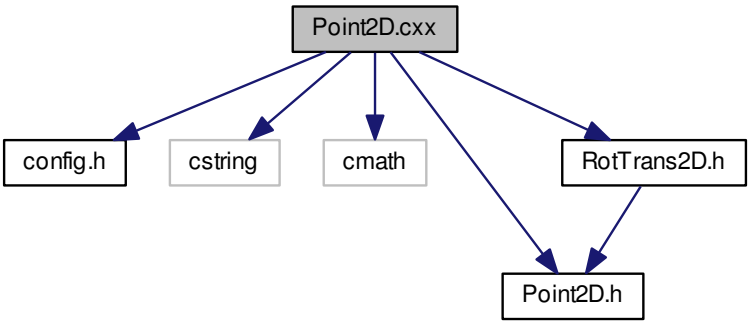
- class PixImage

  A rectangular 2-dimensional array of pixels with individual values.

6.21 Point2D.cxx File Reference

```
#include "config.h"
#include <cstring>
#include <cmath>
#include "Point2D.h"
#include "RotTrans2D.h"
```

Include dependency graph for Point2D.cxx:



Functions

- Point2D operator+ (const Point2D &left, const Point2D &right)

  Add two points, interpreting both points as vectors.

6.21.1 Function Documentation

6.21.1.1 Point2D operator+ (const Point2D & *left,* const Point2D & *right*)

Add two points, interpreting both points as vectors.

Parameters

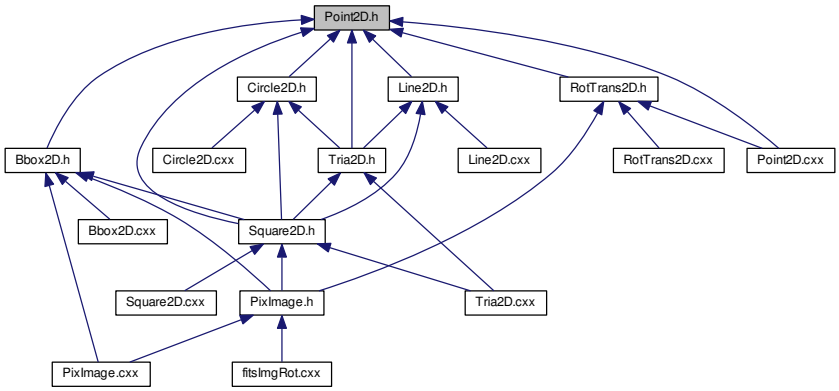
left	The point to the left of the + sign.
right	The point to right of the + sign.

Returns

A point with coordinates that are sums of coordinates of both arguments.

6.22 Point2D.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class Point2D

  A point with 2 coordinates represented in a Cartesian coordinate system.

Functions

- Point2D operator+ (const Point2D &left, const Point2D &right)

  Add two points, interpreting both points as vectors.

6.22.1 Function Documentation

6.22.1.1 Point2D operator+ (const Point2D & *left*, const Point2D & *right*)

Add two points, interpreting both points as vectors.

Parameters

left	The point to the left of the + sign.
right	The point to right of the + sign.

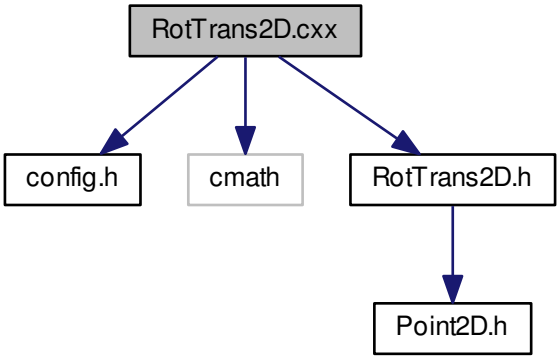
Returns

A point with coordinates that are sums of coordinates of both arguments.

6.23 RotTrans2D.cxx File Reference

```
#include "config.h"
#include <cmath>
#include "RotTrans2D.h"
```

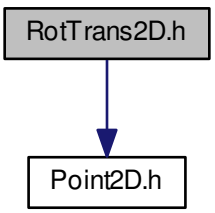
Include dependency graph for RotTrans2D.cxx:



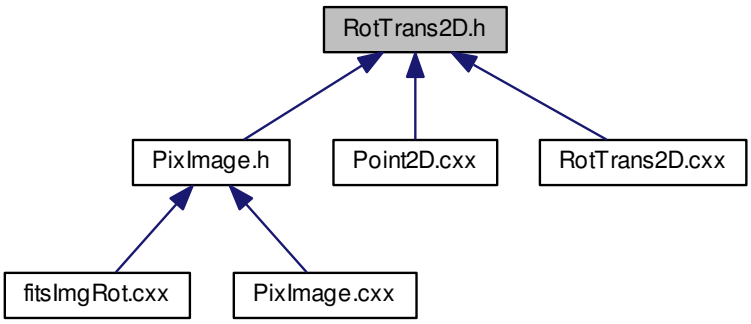
6.24 RotTrans2D.h File Reference

```
#include "Point2D.h"
```

Include dependency graph for RotTrans2D.h:



This graph shows which files directly or indirectly include this file:



Classes

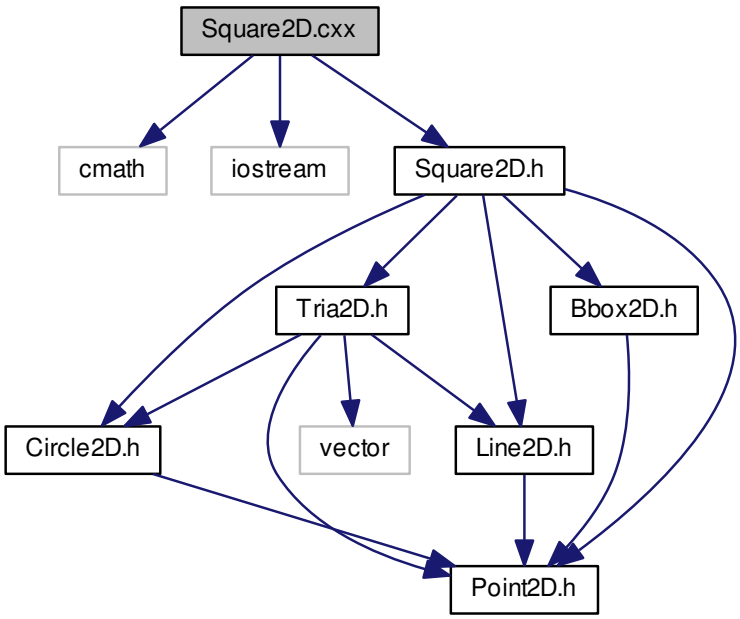
- class RotTrans2D

  A rotation followed by a translation.

6.25 Square2D.cxx File Reference

```
#include <cmath>
#include <iostream>
#include "Square2D.h"
```

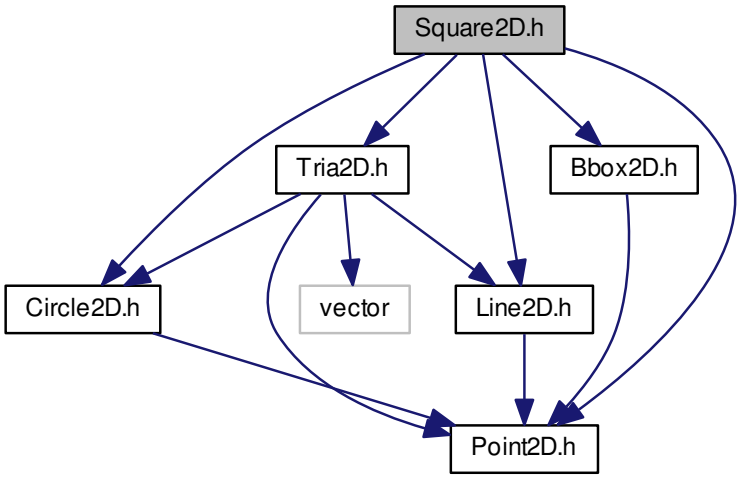
Include dependency graph for Square2D.cxx:



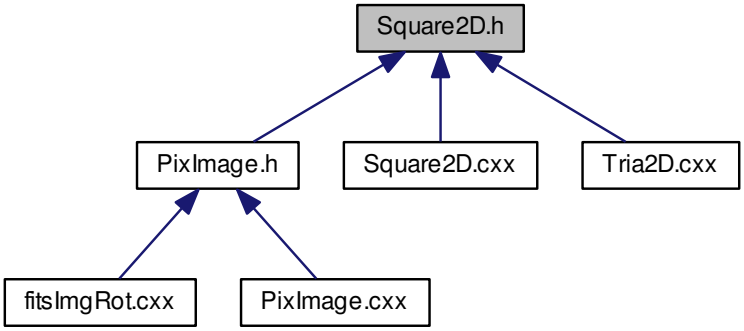
6.26 Square2D.h File Reference

```
#include "Point2D.h"
#include "Line2D.h"
#include "Circle2D.h"
#include "Tria2D.h"
#include "Bbox2D.h"
```

Include dependency graph for Square2D.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Square2D

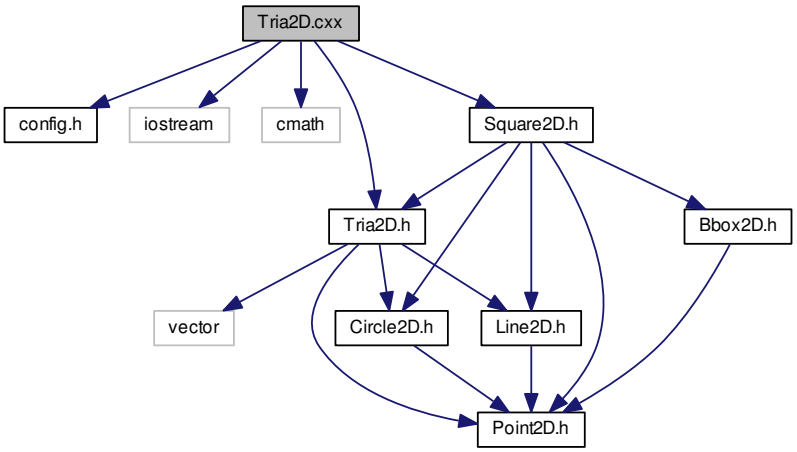
  A square represented by the four vertices of the corners.

6.27 Tria2D.cxx File Reference

```
#include "config.h"
```

```
#include <iostream>
#include <cmath>
#include "Tria2D.h"
#include "Square2D.h"
```

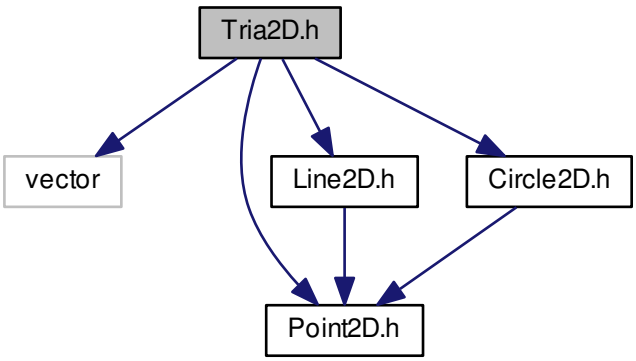
Include dependency graph for Tria2D.cxx:



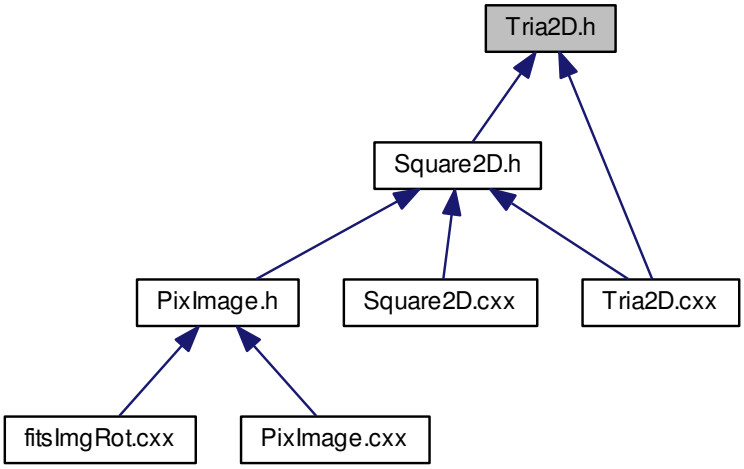
6.28 Tria2D.h File Reference

```
#include <vector>
#include "Point2D.h"
#include "Line2D.h"
#include "Circle2D.h"
```

Include dependency graph for Tria2D.h:



This graph shows which files directly or indirectly include this file:



Classes

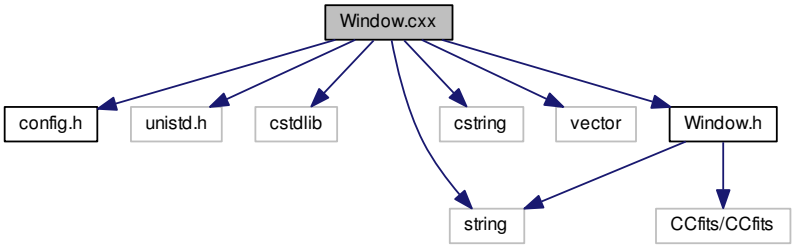
- class Tria2D

  A triangle represented by the Cartesian coordinates of its three vertices.

6.29 Window.cxx File Reference

```
#include "config.h"
#include <unistd.h>
#include <cstdlib>
#include <string>
#include <cstring>
#include <vector>
#include "Window.h"
```

Include dependency graph for Window.cxx:



Macros

- #define GEIRS2PANIC_WINDOW_TRIM

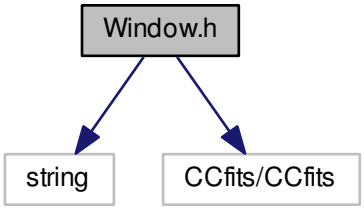
6.29.1 Macro Definition Documentation

6.29.1.1 #define GEIRS2PANIC_WINDOW_TRIM

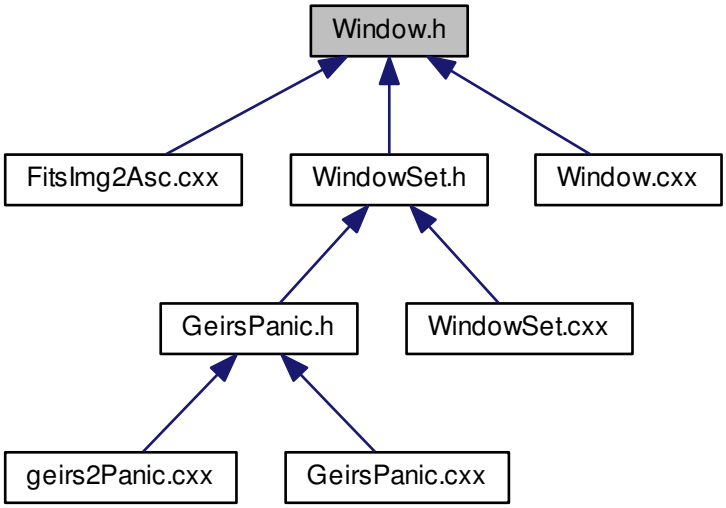
6.30 Window.h File Reference

```
#include <string>
#include <CCfits/CCfits>
```

Include dependency graph for Window.h:



This graph shows which files directly or indirectly include this file:



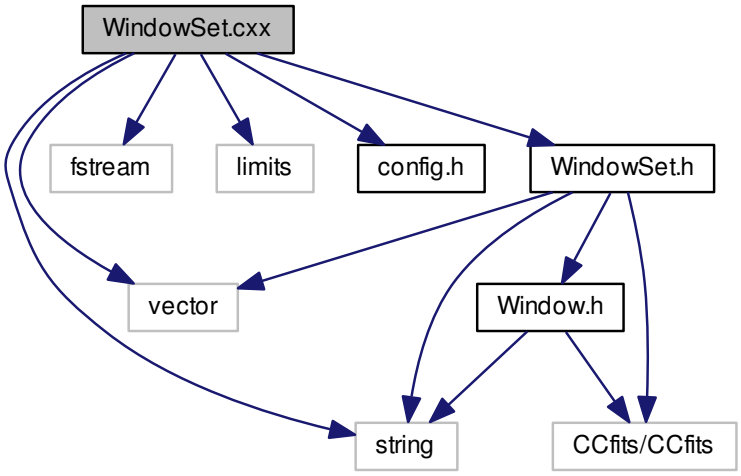
Classes

- class Window

6.31 WindowSet.cxx File Reference

```
#include <string>
#include <vector>
#include <fstream>
#include <limits>
#include "config.h"
#include "WindowSet.h"
```

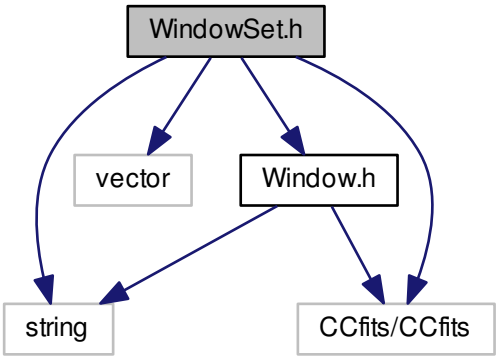
Include dependency graph for WindowSet.cxx:



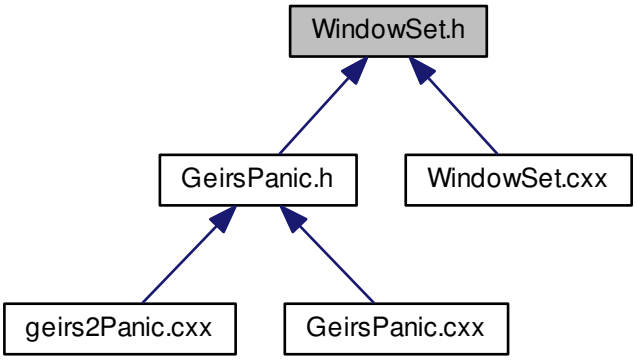
6.32 WindowSet.h File Reference

```
#include <string>
#include <vector>
#include <CCfits/CCfits>
#include "Window.h"
```

Include dependency graph for WindowSet.h:



This graph shows which files directly or indirectly include this file:



Classes

- class WindowSet

Index

~FitsImg2Asc
  FitsImg2Asc, 12
~PixImage
  PixImage, 30
~Window
  Window, 52
~WindowSet
  WindowSet, 59

altAzPar
  GeirsPanic, 16
angle
  RotTrans2D, 39
apply
  Bbox2D, 6
  PixImage, 30
  Point2D, 34
  RotTrans2D, 39
  Square2D, 42
area
  Square2D, 42
  Tria2D, 46
at
  Line2D, 27
  PixImage, 30

badMask
  FitsImg2Asc, 13
Bbox2D, 4
  apply, 6
  Bbox2D, 6
  Bbox2D, 6
  include, 6
  isInside, 6
  limits, 7
  lowerRight, 7
  upperLeft, 7
Bbox2D.cxx, 60
Bbox2D.h, 61
bits
  Window, 57
blankRef
  Window, 53

canonName
  Window, 54
chop
  Tria2D, 46
chpsz
  Window, 56
Circle2D, 7
  Circle2D, 10
  Circle2D, 10
  overlap, 10
  rad, 10
Circle2D.cxx, 62
Circle2D.h, 63
circumC
  Square2D, 42
  Tria2D, 46
cols
  PixImage, 31
config.h, 64
  HAVE_CMATH, 64
  HAVE_CSTRING, 64
  HAVE_INTTYPES_H, 64
  HAVE_LIBCCFITS, 64
  HAVE_LIBM, 64
  HAVE_MATH_H, 64
  HAVE_MEMCPY, 64
  HAVE_MEMORY_H, 64
  HAVE_SINCOS, 64
  HAVE_STDINT_H, 64
  HAVE_STDLIB_H, 64
  HAVE_STRING_H, 65
  HAVE_STRINGS_H, 65
  HAVE_SYS_STAT_H, 65
  HAVE_SYS_TYPES_H, 65
  HAVE_UNISTD_H, 65
  PACKAGE_BUGREPORT, 65
  PACKAGE_NAME, 65
  PACKAGE_STRING, 65
  PACKAGE_TARNAME, 65
  PACKAGE_URL, 65
  PACKAGE_VERSION, 65
  STDC_HEADERS, 65
contName
  Histos, 26
cooGeirsNoWin
  Window, 54
cooGeirsWinToSplic
  Window, 55
cooGeirsWinToSplicRot
  Window, 53
cooGeirsWinToSplicRotWin
  Window, 55
cooToWin
  Window, 55
coord
  Point2D, 37
countRange
  Histos, 25
cts
  Histo, 21
cumCts
  Histo, 21

detIdx
  Window, 55
detsize
  Window, 56

dissect
Square2D, 42
dist
Point2D, 36
dotprod
Point2D, 36
dump
FitsImg2Asc, 12
dumpFil
Histo, 20
Histos, 24

eLen
Square2D, 44
edge
Square2D, 42
Tria2D, 46
exec
GeirsPanic, 15

farr
Window, 57
FitsImg2Asc, 10
∼FitsImg2Asc, 12
badMask, 13
dump, 12
FitsImg2Asc, 12
FitsImg2Asc, 12
histo, 13
iname, 13
FitsImg2Asc.cxx, 68
fitsImg2Asc.cxx, 65
main, 66
usage, 66
FitsImg2Asc.h, 68
fitsImgRot.cxx, 69
usage, 70
fourPix
Window, 53
fsort
Histo.cxx, 76
Histo.h, 77

gap
Window, 56
geirs2Panic.cxx, 70
main, 71
usage, 71
GeirsPanic, 13
altAzPar, 16
exec, 15
GeirsPanic, 15
GeirsPanic, 15
hex2deg, 17
iname, 17
ofits, 17
oname, 17
wgs84, 16
ws, 17
GeirsPanic.cxx, 74
GeirsPanic.h, 74
gnuplot
Histos, 24

HAVE_CMATH
config.h, 64
HAVE_CSTRING
config.h, 64
HAVE_INTTYPES_H
config.h, 64
HAVE_LIBCCFITS
config.h, 64
HAVE_LIBM
config.h, 64
HAVE_MATH_H
config.h, 64
HAVE_MEMCPY
config.h, 64
HAVE_MEMORY_H
config.h, 64
HAVE_SINCOS
config.h, 64
HAVE_STDINT_H
config.h, 64
HAVE_STDLIB_H
config.h, 64
HAVE_STRING_H
config.h, 65
HAVE_STRINGS_H
config.h, 65
HAVE_SYS_STAT_H
config.h, 65
HAVE_SYS_TYPES_H
config.h, 65
HAVE_UNISTD_H
config.h, 65
hex2deg
GeirsPanic, 17
Histo, 17
cts, 21
cumCts, 21
dumpFil, 20
Histo, 19
init, 20
mimax, 21
name, 21
perc, 21
stride, 21
strt, 21
valRange, 20
histo
FitsImg2Asc, 13
Histo.cxx, 75
fsort, 76
Histo.h, 76
fsort, 77
Histos, 22
contName, 26

countRange, 25
dumpFil, 24
gnuplot, 24
Histos, 23, 24
hs, 26
init, 25
stride, 25
strt, 25
valRange, 25
Histos.cxx, 77
Histos.h, 78
hs
Histos, 26
hull
PixImage, 31

iarr
Window, 57
idx
PixImage, 31
ifileset
WindowSet, 59
ifits
Window, 57
iname
FitsImg2Asc, 13
GeirsPanic, 17
Window, 56
WindowSet, 60
include
Bbox2D, 6
init
Histo, 20
Histos, 25
Window, 56
intersect
Square2D, 44
Tria2D, 48
intersectInf
Line2D, 28
inv
RotTrans2D, 39
isInside
Bbox2D, 6
isRef
Window, 52, 53

leftFrom
Line2D, 27
len
Point2D, 36
limits
Bbox2D, 7
Line2D, 26
at, 27
intersectInf, 28
leftFrom, 27
Line2D, 27
Line2D, 27
pts, 28
turn90, 27
Line2D.cxx, 79
Line2D.h, 79
lowerRight
Bbox2D, 7

main
fitsImg2Asc.cxx, 66
geirs2Panic.cxx, 71
mimax
Histo, 21

name
Histo, 21
naxes
Window, 57

ofits
GeirsPanic, 17
oname
GeirsPanic, 17
operator Bbox2D
PixImage, 30
Square2D, 42
operator+
Point2D.cxx, 83
Point2D.h, 84
operator+=
Point2D, 36
operator-=
Point2D, 36
overlap
Circle2D, 10

PACKAGE_BUGREPORT
config.h, 65
PACKAGE_NAME
config.h, 65
PACKAGE_STRING
config.h, 65
PACKAGE_TARNAME
config.h, 65
PACKAGE_URL
config.h, 65
PACKAGE_VERSION
config.h, 65
perc
Histo, 21
PixImage, 28
~PixImage, 30
apply, 30
at, 30
cols, 31
hull, 31
idx, 31
operator Bbox2D, 30
PixImage, 29, 30
PixImage, 29, 30

rows, 31
toFits, 30
vals, 31
PixImage.cxx, 80
PixImage.h, 81
pixRot
Window, 55
pixsc
Window, 57
plus
Point2D, 34
Point2D, 32
apply, 34
coord, 37
dist, 36
dotprod, 36
len, 36
operator+=, 36
operator-=, 36
plus, 34
Point2D, 34
Point2D, 34
to, 34
turn90, 34
Point2D.cxx, 82
operator+, 83
Point2D.h, 84
operator+, 84
pts
Line2D, 28

rad
Circle2D, 10
readPix
Window, 56
rmFiles
Window, 54
WindowSet, 59
RotTrans2D, 37
angle, 39
apply, 39
inv, 39
RotTrans2D, 38, 39
RotTrans2D, 38, 39
sc, 40
trans, 39
RotTrans2D.cxx, 85
RotTrans2D.h, 85
rows
PixImage, 31

STDC_HEADERS
config.h, 65
sarr
Window, 57
sc
RotTrans2D, 40
sizeX
Window, 52
sizeY
Window, 52
Square2D, 40
apply, 42
area, 42
circumC, 42
dissect, 42
eLen, 44
edge, 42
intersect, 44
operator Bbox2D, 42
Square2D, 41, 42
Square2D, 41, 42
vert, 44
Square2D.cxx, 86
Square2D.h, 87
stride
Histo, 21
Histos, 25
strt
Histo, 21
Histos, 25

to
Point2D, 34
toFits
PixImage, 30
trans
RotTrans2D, 39
Tria2D, 44
area, 46
chop, 46
circumC, 46
edge, 46
intersect, 48
Tria2D, 46
Tria2D, 46
vert, 48
Tria2D.cxx, 88
Tria2D.h, 89
trimsize
Window, 57
WindowSet, 60
trimsizeFus
Window, 56
trimws
Window, 54
turn90
Line2D, 27
Point2D, 34

upperLeft
Bbox2D, 7
usage
fitsImg2Asc.cxx, 66
fitsImgRot.cxx, 70
geirs2Panic.cxx, 71
usarr
Window, 57

valRange
Histo, 20
Histos, 25
vals
PixImage, 31
vert
Square2D, 44
Tria2D, 48

wgs84
GeirsPanic, 16
win2win
Window, 52
WindowSet, 59
Window, 48
∼Window, 52
bits, 57
blankRef, 53
canonName, 54
chpsz, 56
cooGeirsNoWin, 54
cooGeirsWinToSplic, 55
cooGeirsWinToSplicRot, 53
cooGeirsWinToSplicRotWin, 55
cooToWin, 55
detIdx, 55
detsize, 56
farr, 57
fourPix, 53
gap, 56
iarr, 57
ifits, 57
iname, 56
init, 56
isRef, 52, 53
naxes, 57
pixRot, 55
pixsc, 57
readPix, 56
rmFiles, 54
sarr, 57
sizeX, 52
sizeY, 52
trimsize, 57
trimsizeFus, 56
trimws, 54
usarr, 57
win2win, 52
Window, 51
Window.cxx, 90
Window.h, 91
WindowSet, 57
∼WindowSet, 59
ifileset, 59
iname, 60
rmFiles, 59
trimsize, 60
win2win, 59
WindowSet, 59
WindowSet, 59
ws, 60
WindowSet.cxx, 92
WindowSet.h, 92
ws
GeirsPanic, 17
WindowSet, 60