sim_telarray

2024-02-22

Generated by Doxygen 1.9.1

Chapter 1

The sim_telarray package

1.1 The sim_telarray program

The simulation program itself, `sim_telarray`, is used for simulations of systems (also called arrays) of imaging atmospheric Cherenkov telescopes (IACTs). The telescopes can be located in arbitrary places, no need for a regular array, and each telescope can differ from the other telescopes or all might be identical.

The main input to the program are 'photon bunches' with a weight indicating that the bunch stands for multiple photons or a fraction of a photon. These photon bunches can be generated by the CORSIKA airshower simulation program or by the LightEmission package included with the sim_telarray package. The purpose of LightEmission is to simulate calibration and test measurements to verify the assumptions made in the `sim_telarray` telescope models etc.

For configuring the telescope models at program startup, `sim_telarray` comes with a `cpp` (C Pre-Processor) like pre-processor that is better matched to the configuration preprocessing - but it can also be told to use `cpp` or any other preprocessor.

1.2 The LightEmission package

The LightEmission package can be obtained separately and installed stand-alone, and it is also included with the `sim_telarray` package and gets installed with that. It consists of a library for the common and repetitive work and a set of rather short programs for setting up a number of typical light source configurations in use with IACTs. For a range of applications these programs, with their run-time options, will be sufficient, for other applications they may serve as examples for developing a more specific tool.

1.3 Auxiliary programs

Apart from the pre-processor, `pfp`, `sim_telarray` comes with a number of auxiliary programs, including

- pfp : The configuration pre-processor. Its syntax resembles that of the C preprocessor but it differs in many ways.
- sspp : The 'Simple Selection Pre-Processor'. In contrast to `pfp` it has very simple input, from which the lines relevant to the requested telescope get extracted.

- multipipe_corsika : Can be placed between the CORSIKA/IACT output, written into a `multipipe_corsika` pipe, and one or several `sim_telarray` telescope simulations working at the same time on the same photon bunches but some differences in their configuration.
- extract_corsika_multi: Extract data for one or more subsets of telescopes from CORSIKA IACT data.
- extract_corsika_tel : Extract data for a subset of telescopes from CORSIKA IACT data.
- norm_spe : Tool for proper normalization of the given single photo-electron amplitude distributions for `sim_telarray` (which must have a mean amplitude of 1.0).
- corsika_autoinputs : A tool for setting up a separate working directory for every CORSIKA run, apply some processing on the CORSIKA control input (in addition to what might be done with the `pfp` pre-processor) for incrementing run numbers, refreshing random number seeds, and applying some changes needed for specific CORSIKA versions or interaction models.
- testeff : Evaluate the spectral photon detection efficiency of telescope configuration without actually running the simulation.
- rpolist : Test and evaluate the interpolation of data tables as used in many places with `sim_telarray`.
- gen_seedlist : Generate a list of random number generator seeds for later production with a limited variety in random detector settings (for which distinct calibration runs would be produced as well.)

1.4 Tools for setting up camera and mirror configuration files

The camera configuration file is a rather complex configuration data file combining different aspects of the camera model in order to make sure that the different parts are used in a consistent set. For example, the trigger definitions are specific to the pixel numbering used in the same file. There are not only tools available for setting up different kinds of cameras but also to translate trigger definitions from one pixel numbering to another one, even allowing for small changes in pixel positions between the two pixel definitions.

1.4.1 Tools in Cameras

- hesspix : Set up a camera definition file with majority trigger sectors as in HESS-I.
- hess2pix : Set up a camera definition file with majority trigger sectors as in HESS-II.
- mapmtpix : Set up a camera definition suitable for Multi-Anode PMTs.
- smartpix : Set up a camera definition for SmartPixel where each pixel receives trigger information from its neighbours.
- modular_camera : Can set up different types of cameras but has to be compiled with corresponding definitions first.
- scale_pix : Scale and/or rotate pixel positions from an existing camera definition.
- pixel_remap_trg : Translate the trigger definitions from one camera definition file to another camera definition file with a different pixel numbering schemes describing an equivalent set of pixels which may be scaled/rotated and otherwise slightly differing from those in the first file.
- fov : Calculate the FoV (Field-of-View) of a camera from its definition file plus some effective focal length, according to different definitions of FoV.
- drawdrawers : ...

1.4.2 Tools in Mirrors

- plot_mirrors : Create a mirror definition file for a single reflector telescope, using circular, hexagonal or square mirrors, with different limitations defining the inner and outer edge of the area instrumented with mirrors.
- draw_mirrors : Produce a simple xfig drawing of a the mirrors on a single reflector telescope, together with the camera, based on an existing mirror definition file.
- conical_taylor : Convert a surface as parametrized by Zemax into its pure Taylor expansion as used for dual mirror telescopes in sim_telarray.

1.5 Tools in the hessio module / hessioxxx package

Not strictly part of sim_telarray but of the EventIO data handling software for sim_telarray (commonly known as 'hessio'):

- read_simtel (read_hess, read_cta) : Read sim_telarray data files, show the data contents, do some reconstruction, produce plots, ...
- read_iact : Read and show contents of data from the CORSIKA IACT module.
- select_iact : Select a subset of the CORSIKA/IACT data, for example light emitted by specific types or energies of particles etc. before feeding that into sim_telarray.
- list_histograms : List/print/show histograms in a 'hdata' file.
- add_histograms : Add up all histograms from any number of histogram files produced by sim_telarray or read_simtel.
- hdata2root : Convert 'hdata' histogram files into ROOT histogram files.
- extract_simtel : Extract data for a subset of telescopes from a sim_telarray data file, renumbering the IDs of the extracted telescopes.
- merge_simtel : Combine sim_telarray simulations for the same CORSIKA events, as processed seperately in different sim_telarray, into a common sim_telarray data file.
- extract_calibevent : Extract the 'hidden' dark/pedestal/flatfield events produced by sim_telarray into normal events (one type per output file).
- split_hessio : Separate data into an event data file and monitoring/calibration data for each telescope, for testing later event merging.
- listio : Low-level listing of data blocks in any EventIO data file/stream.
- statio : Show statistics of data blocks seen in any EventIO data file.
- filterio : Select or de-select certain types of data blocks from and EventIO data file.
- fcat : Stream data, decompressed, from any File/URL from which the fileopen function in the EventIO (hessio) library can read, independent of the contents. Decompression is based on filename suffix only.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 The testeff program

Data Structures

- struct effcalc

  Collect original and derived values in nanometer steps, even if (presently) not wavelength dependent.

Macros

- #define **MAX_ALT** 1000
- #define **MAX_WL** 2000
- #define **MAX_MIRRORS** 4000

Typedefs

- typedef struct effcalc **EffCalc**

Functions

- double **rhof_** (double __attribute__((unused)) *height)
- double **thick_** (double __attribute__((unused)) *height)
- double **heigh_** (double __attribute__((unused)) *thick)
- double nsb_za_scale (double airmass)

  Primitive scaling of NSB with zenith angle.
- void **strip_comments** (char *line)
- int read_spe_check (const char *fname)

  Read single-p.e.
- void **syntax** (int argc, char **argv)
- int main (int argc, char **argv)

Variables

- double **airlightspeed** = 29.9792458/1.0002256
- int **global_verbose** = 0

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 main()

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

< Nominal focal length, distance of inner mirrors from camera [m]

< Effective focal length, for image scale [m]

< Curvature of dish shape, dish assumed spherical [m]

< Pixel area and solid angle per pixel on sky

< Effective solid angle for albedo contribution to NSB

< Observation level altitude, as configured [m]

< Same, converted to [cm]

< Atmospheric transmission at site in question.

< Assumes any scattered photon as lost, thus of limited value for NSB.

< Should correspond to Benn & Ellison NSB.

< By default correct NSB from B&E site to selected site, and angle-dependent

Definition at line 278 of file testeff.c.

References effcalc::alb_solid_angle, effcalc::alb_wl, effcalc::degraded, effcalc::iwl, MAX_WL, effcalc::pixel_area, effcalc::qe_only, effcalc::qe_ref, effcalc::ref_only, effcalc::solid_angle, and effcalc::tel_trans.

6.1.2.2 nsb_za_scale()

```
double nsb_za_scale (
            double airmass )
```

Primitive scaling of NSB with zenith angle.

At least air-glow emission component of NSB light increases with airmass. Star light and zodiacal light are independent of airmass. Stray light from cities may rapidly increase with airmass but is ignored. Keep in mind this is a very primitive scaling since we do not have the separate component spectra available. For a more realistic scaling you are advised to obtain zenith-angle dependent model spectra and use them without further corrections and scaling applied.

Definition at line 82 of file testeff.c.

6.1.2.3 read_spe_check()

```
int read_spe_check (
            const char * fname )
```

Read single-p.e.

response and check that it seems reasonable.

Definition at line 120 of file testeff.c.

6.2 The corsika_autoinputs program

Macros

- #define **NEED_DEV_URANDOM** 1

Functions

- int **new_run** (const string &conf_path)
- void **syntax** ()
- int main (int argc, char **argv)

6.2.1 Detailed Description

6.2.2 Function Documentation

6.2.2.1 main()

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

< Corsika 7.5xxx

< Corsika 7.6xxx

< Corsika 7.7xxx

Definition at line 295 of file corsika_autoinputs.cc.

6.3 The extract_corsika_multi program

Macros

- #define **MAXTEL** 1000
- #define **MAXSELECT** 20
- #define **MAX_ARRAY** 200

Functions

- void stop_signal_function (int isig)

  Stop the program gracefully when it catches an INT or TERM signal.
- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- void pipe_signal_function (int isig)

  This signal handler gets involved when a SIGPIPE is produced on input or output.
- void format_num (char ∗text, size_t maxl, double num)

  Special formatting of floating point numbers.
- void hup_signal_function (int __attribute__((unused)) isig)

  Produce intermediate output when the program catches an HUP signal.
- void syntax (void)

  Tell how to run this program.
- int **main** (int argc, char ∗∗argv)

Variables

- static int **interrupted** = 0
- static int **broken** = 0

6.3.1 Detailed Description

6.3.2 Function Documentation

6.3.2.1 hup_signal_function() [1/2]

```
void hup_signal_function (
            int __attribute__((unused)) isig )
```

Produce intermediate output when the program catches an HUP signal.

Parameters

isig	Signal number.

Returns

(none)

Definition at line 131 of file extract_corsika_multi.c.

References hup_signal_function().

6.3.2.2 hup_signal_function() [2/2]

```
void hup_signal_function (
             int isig )
```

Produce intermediate output when the program catches an HUP signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 4343 of file sim_telarray.c.

Referenced by hup_signal_function().

6.3.2.3 stop_signal_function()

```
void stop_signal_function (
             int isig )
```

Stop the program gracefully when it catches an INT or TERM signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 87 of file extract_corsika_multi.c.

References interrupted.

6.4 The extract_corsika_tel program

Macros

- #define **MAXTEL** 1000
- #define **MAX_ARRAY** 200

Functions

- void stop_signal_function (int isig)

  Stop the program gracefully when it catches an INT or TERM signal.
- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- void pipe_signal_function (int isig)

  This signal handler gets involved when a SIGPIPE is produced on input or output.
- void format_num (char ∗text, size_t maxl, double num)

  Special formatting of floating point numbers.
- void hup_signal_function (int __attribute__((unused)) isig)

  Produce intermediate output when the program catches an HUP signal.
- int setenv_int_array (const char ∗name, int ∗ival, int nval)

  Set an environment value with a comma-separated list of integers.
- int setenv_dbl_array (const char ∗name, double ∗dval, int nval)

  Set an environment value with a comma-separated list of floating point values.
- void syntax (void)

  Tell how to run this program.
- int **main** (int argc, char ∗∗argv)

Variables

- static int **interrupted** = 0
- static int **broken** = 0

6.4.1 Detailed Description

6.4.2 Function Documentation

6.4.2.1 hup_signal_function() [1/2]

```
void hup_signal_function (
            int __attribute__((unused)) isig )
```

Produce intermediate output when the program catches an HUP signal.

Parameters

isig	Signal number.

Returns

(none)

Definition at line 130 of file extract_corsika_tel.c.

References hup_signal_function().

6.4.2.2 hup_signal_function() [2/2]

```
void hup_signal_function (
            int isig )
```

Produce intermediate output when the program catches an HUP signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 4343 of file sim_telarray.c.

Referenced by hup_signal_function().

6.4.2.3 stop_signal_function()

```
void stop_signal_function (
            int isig )
```

Stop the program gracefully when it catches an INT or TERM signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 86 of file extract_corsika_tel.c.

References interrupted.

6.5 The multipipe_corsika program

Data Structures

- struct pipecmd
- struct postproc_entry

Typedefs

- typedef struct postproc_entry **PostProc**

Functions

- void stop_signal_function (int isig)

  Stop the program gracefully when it catches an INT or TERM signal.
- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- void pipe_signal_function (int isig)

  This signal handler gets involved when a SIGPIPE is produced on input or output.
- void **seq_handler** (int signo, siginfo_t *info, void *context)
- void close_all ()

  Close all output pipes.
- void format_num (char *text, size_t maxl, double num)

  Special formatting of floating point numbers.
- pid_t find_corsika_pid ()

  Try to find the CORSIKA process in the list of ancestors of the current process.
- void show_procinfo (pid_t pid)

  Print information about a process, based on the process ID.
- int add_procinfo (pid_t pid)

  If no info about the given process ID was printed before and not too many processed got printed so far, print process information and keep track that this process ID was already shown.
- void show_procgroup ()

  Show the processes that are in the same process group as the current process.
- double **tvtime** (void)
- void hup_signal_function (int __attribute__((unused)) isig)

  Produce intermediate output when the program catches an HUP signal.
- void seq_handler (int signo, siginfo_t *info, void __attribute__((unused)) *context)

  Handle the signal that the telescope simulation should send in sequential mode after having processed a block of data.
- PostProc * add_postproc_cmd (PostProc *pp_start, const char *cmd)

  Add a post-processing command at the end of an existing list or as the first of a list.
- int setenv_int_array (const char *name, int *ival, int nval)

  Set an environment value with a comma-separated list of integers.
- int setenv_dbl_array (const char *name, double *dval, int nval)

  Set an environment value with a comma-separated list of floating point values.
- void syntax (void)

  Tell how to run this program.
- int **main** (int argc, char **argv)

Variables

- struct pipecmd * **pipe_base**
- static int **pcount** = -2
- static int **signo_seq_ini** = 0
- static int **nsleep** = 3
- static int **sleep_time** [5] = { 0, 1, 10, 30, 0 }
- static size_t **sleep_stats** [5]
- static int **startup_slept**
- static int **interrupted** = 0
- static int **broken** = 0
- static int **seq_ready** = 0
- static int n_list_proc = 0

  Number of processes about which we already showed info.
- static int list_proc [20]

  Process IDs of which info was shown.
- static int max_list_proc = 20

  Maximum number of process IDs to show info.
- static size_t **n_pipes** = 0
- static int * **optional_pipes**
- static double **tvstart** = 0

6.5.1 Detailed Description

6.5.2 Function Documentation

6.5.2.1 find_corsika_pid()

```
pid_t find_corsika_pid (
            void  )
```

Try to find the CORSIKA process in the list of ancestors of the current process.

Unless the CORSIKA process identifies itself via the CORSIKA_PID environment variable (which would work on all systems), we have to go from one process to its parent process and so on until we find one with a name that looks like this is CORSIKA. The latter method is Linux-specific and may be even specific to certain ranges of Linux kernel versions.

The CORSIKA process ID is only relevant for the sequential mode where CORSIKA would be sent to sleep while one telescope simulation after the other is processing data one at a time. In the default mode all processes would try to run in parallel to the extend that the system can handle.

Definition at line 271 of file multipipe_corsika.c.

6.5.2.2 hup_signal_function() [1/2]

```
void hup_signal_function (
            int __attribute__((unused)) isig )
```

Produce intermediate output when the program catches an HUP signal.

Parameters

isig	Signal number.

Returns

(none)

Definition at line 202 of file multipipe_corsika.c.

References hup_signal_function().

6.5.2.3 hup_signal_function() [2/2]

```
void hup_signal_function (
            int isig )
```

Produce intermediate output when the program catches an HUP signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 4343 of file sim_telarray.c.

References iobuf, save_histograms(), and stop_signal_function().

Referenced by hup_signal_function().

6.5.2.4 seq_handler()

```
void seq_handler (
            int signo,
            siginfo_t * info,
            void __attribute__((unused)) * context )
```

Handle the signal that the telescope simulation should send in sequential mode after having processed a block of data.

Note: this signal handler is established by sigaction() such that it remains active at all times and doesn't need to be reloaded. Other signal handlers established by signal() may have to be reloaded (depending on BSD versus System V type system behaviour).

See `man sigaction` for a description of parameters.

Definition at line 325 of file multipipe_corsika.c.

6.5.2.5 show_procinfo()

```
void show_procinfo (
            pid_t pid )
```

Print information about a process, based on the process ID.

This includes the location of the program executed and command line arguments (although the latter could be modified by the program). Since this output would follow error messages about signals it also goes to stderr rather than stdout.

Note: portability of the method beyond the Linux ecosystem has not been investigated so far.

Definition at line 457 of file multipipe_corsika.c.

Referenced by add_procinfo().

6.5.2.6 stop_signal_function()

```
void stop_signal_function (
            int isig )
```

Stop the program gracefully when it catches an INT or TERM signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 150 of file multipipe_corsika.c.

References interrupted.

6.6 The portable file (originally: Fortran) preprocessor

Data Structures

- struct IncPath

Macros

- #define **_GNU_SOURCE** 1
- #define **WITH_EXISTS** 1
- #define **TRUE** 1
- #define **FALSE** 0
- #define **ERRNUM** 4
- #define **MAXFNAME** 80
- #define **DEFAULT_OUTPUT_EXTENSION** ".for"
- #define **EXTENSION_REQUIRED**
- #define **error**(msg) fprintf(stderr,"%s (%d): %s\n",fname,line_num,msg)
- #define **MAX_IF** 33
- #define **MAXLEN** 10000

Functions

- int include_file (char ∗fname, int flag, int ifexists)

  Include and process text from a file into what is currently being processed.
- void **open_output** (char ∗)
- void **define** (TREE ∗, int, long, const char ∗)
- int **getword** (char ∗, int ∗, char ∗, int, char, char)
- TREE ∗ **expand_if** (char ∗)
- TREE ∗ **expand_if_line** (char ∗)
- int **inputprocess** (char ∗line0, char ∗line, int len, int verb)
- void **syntax** (int ec)
- char ∗ **strupr** (char ∗)
- char ∗ **strlwr** (char ∗)
- char ∗ **my_strcasestr** (char ∗haystack, const char ∗needle)
- int **main** (int argc, char ∗∗argv)

Variables

- char ∗ **IncPath::path**
- struct IncPath ∗ **IncPath::next**
- static struct IncPath **inc_root**
- static struct IncPath ∗∗ **inc_next**
- FILE ∗ **output**
- TREE ∗ **common**
- TREE ∗ **defined**
- TREE ∗ **TrueTree**
- TREE ∗ **FalseTree**
- int **in_common**
- int **inc_level**
- int **new_output**
- char **message** [2000]

- static const char * **empty_string** = ""
- static int **return_code** = 0
- static char **control_character** = '#'
- static int **quiet** = 0
- static int **historian** = 0
- static int **varsub** = 0
- static int **error_message_mode** = 0
- static int **current_line_num** = 0
- static const char * **current_fname** = "(No active file)"
- static int **add_missing_newline** = 1

6.6.1 Detailed Description

6.6.2 Function Documentation

6.6.2.1 include_file()

```
int include_file (
            char * fname,
            int flag,
            int ifexists )
```

Include and process text from a file into what is currently being processed.

Parameters

fname	Name of the file to include
flag	If 0 use plain file name as given (typically for a file that must be in the current directory), otherwise search it in the list of include directories. For the latter action, use '#include <filename>' rather than '#include "filename"' or '#include filename'.
ifexists	If this variable is non-zero, any attempt to include a non-existing file is silently ignored.

Returns

0 for success, -1 for failure.

Definition at line 473 of file pfp.c.

Chapter 7

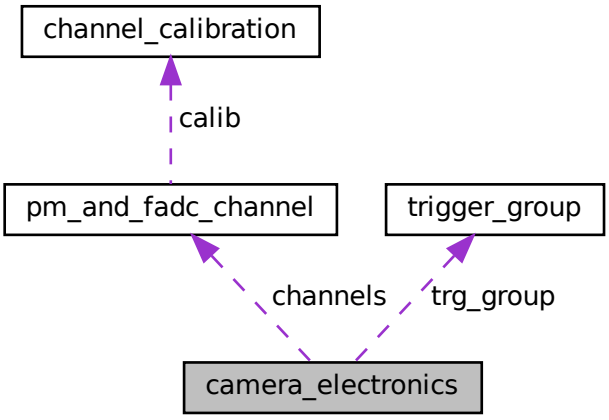
Data Structure Documentation

7.1 camera_electronics Struct Reference

Electronics of a whole camera and its parameters:

`#include <mc_aux.h>`

Collaboration diagram for camera_electronics:



Data Fields

- int telescope

  The 'official' number (ID) for the telescope.
- int itel

  Internal telescope counter (starting at zero).
- int pixels

  Number of pixels.

- int num_gains

  One or two gains possible (if WITH_LOW_GAIN_CHANNEL).
- struct pm_and_fadc_channel * **channels**
- int disc_bins

  Number of time bins used for discr./comparator simulation.
- int disc_start

  No. of bins by which discr./comp. sim. is ahead of FADC.
- int disc_rise_steps

  Time sub-steps for discr./comp. output signal rise.
- int disc_fall_steps

  Time sub-steps for discr./comp. output signal fall-off.
- int use_logic_trigger

  Use logic signals from disc./comp. instead of analogue outputs.
- int fadc_bins

  Number of bins digitized in simulation (usually more than recorded).
- int fadc_per_channel

  Number of FADCs per readout channel (each with a slower sampling, phased to mimick reported sampling rate, if >1).
- int channels_per_chip

  Number of channels readout per chip.
- int sum_bins

  Number of bins in recorded ADC data or sum (read-out or integration window).
- int sum_offset

  Number of bins before telescope trigger where summing starts.
- int sum_start

  Bin where summation started for present image.
- int longsum_bins

  For 'long' events used in place of sum_bins.
- int longsum_offset

  For 'long' events used in place of sum_offset.
- int fadc_max_signal

  Maximum value of digitized signal amplitude.
- int fadc_max_sum

  Maximum value of recorded signal sum.
- double * xspe_prompt

  Random number lookup tables for single.
- double * yspe_prompt

  photoelectron amplitude spectrum for 'prompt' pulses.
- double * xspe_bkgrnd

  Same for nightsky background including ion.
- double * yspe_bkgrnd

  feedback.
- double spe_norm

  Ratio of actually achieved mean single-p.e. signal to expected.
- int nspe_prompt

  Size of preceding tables for signal.
- int nspe_bkgrnd

  and background.
- int use_comp_ped

  Compensation of pedestal values is active (1) or not (0).
- int peak_sensing

If > 0 use peak sensing of given window size instead of ADC sum.

- int shape_length

  Actual length of pulse shape in bins.
- double shape [OVERSAMPLING ∗MAX_SHAPE_LENGTH]

  Single photoelectron.
- int bkg_shape_length

  Simpler treatment for background from nightsky.
- double bkg_shape [MAX_SHAPE_LENGTH]

  Pulse shape from nightsky.
- double max_int_frac

  Maximum fraction of pulse shape within sum_bins (not applicable to peak sensing)
- double max_pixtm_frac

  Maximum fraction of pulse shape within local/global peak window (not applicable to peak sensing)
- int disc_shape_length

  Length of signal at discriminator [# of bins ∗ DISC_BITS_PER_BIN].
- double disc_shape [MAX_SHAPE_LENGTH]

  Pulse shape at discriminators.
- int default_trigger_type

  0: majority, 1: analog sum, 2: dig. sum
- int trigger_mode_bits

  Bit mask of trigger types present in camera configuration.
- int with_analog_trigger

  Non-zero if any trigger groups with majority or analog sum.
- int with_majo_trigger

  Non-zero if there are any trigger groups with majority trigger.
- int with_asum_trigger

  Non-zero if an analog sum trigger was defined.
- int asum_shape_length

  Length of signal shaped for analog sum (optional).
- int asum_shape_offset

  Offset where shaping is done, typically offset of peak.
- double **asum_shape** [MAX_SHAPE_LENGTH]
- double asum_clipping

  Clipping level for analog sums, if any.
- double asum_threshold

  Threshold for analog sum trigger.
- int with_dsum_trigger

  Non-zero if a digital sum trigger was defined.
- int dsum_pedsub

  If non-zero, the pedestal (integer) is subtracted before shaping.
- int dsum_zero_clip

  If non-zero, clip negative values after shaping at zero, before further summation.
- int dsum_prescale [2]

  After shaping and before clipping and possible pre-summation, an integer multiplication[0]/division[1] is applied as pre-scaling.
- int dsum_presum_max

  Before a final bit shift the patch-wise pre-sum cannot be larger than that.
- int dsum_presum_shift

  After pre-summation and clipping a (right) bit-shift may be applied.
- int dsum_shape_length

  Length of signal shaped for digital sum (optional, high-gain only).

- int dsum_shape_offset

  Offset where shaping is done, typically offset of peak.
- dsum_t **dsum_shape** [MAX_SHAPE_LENGTH]
- int dsum_clipping

  Clipping level for digital sums after shaping, if any. [ADC counts].
- int dsum_pre_clipping

  Clipping level for digital sums before shaping, if any. [ADC counts].
- int dsum_threshold

  Threshold for digital sum trigger. [ADC counts].
- int dsum_ignore_below

  If > 0, use shaped clipped signal only where exceeding this value. [ADC counts].
- double random_mono_prob

  Randomly choose events to pass through stereo selection even if mono.
- int muon_mono_threshold [2]

  Thresholds on number of trigger groups and trigger regions (collections of groups) for muon-ring-enhanced events acceptable as mono.
- int long_event_threshold

  Thresholds on number of trigger groups for testing the long event case (actual conditions likely to change).
- int long_event

  Event should be read out as a long event.
- int mono_muon

  Event contains a muon ring candidate and should be accepted as mono event.
- int mono_random

  Event was randomly selected to be accepted as mono event.
- double frequency

  FADC frequency in MHz;.
- double interval

  Corresponding time interval in nanoseconds.
- double trigger_delay_compensation [MAX_TRG_TYPES]

  Trigger type specific delay compensation [ns].
- double photon_delay

  Incoming photons are delayed by so many nanoseconds.
- double fadc_noise

  Dark noise / digitising noise (FADC counts).
- double fadc_noise_lg

  Same for low-gain channel.
- double quantum_efficiency [MAX_LAMBDA]

  Q.E.
- double max_qe_rel

  Maximum of relative quantum efficiencies of all channels.
- double collection_efficiency

  Photoelectron collection efficiency; is included in the quantum_efficiency.
- double optics_efficiency [MAX_LAMBDA]

  Everything before camera window, is part of

- int min_wl_eff

  Minimum wavelength where the telescope has any efficiency.
- int max_wl_eff

  Maximum wavelength where the telescope has any efficiency.
- double lightguide_reflectivity

  Reflectivity of the 'funnel' before PMs.

- double transit_time_jitter

  Jitter of transit times for each pm [ns].
- double simple_threshold

  Simple 'discriminator' threshold in units of photoelectrons (only used when discriminator signals not fully simulated).
- double signal_area

  Area under signal pulse shape.
- double bkgrnd_area

  Area under background pulse shape.
- double disc_area

  Area under discriminator pulse shape.
- int fadc_ac_coupled

  Is 1 if FADCs are AC coupled.
- int disc_ac_coupled

  Is 1 if discriminators are AC coupled.
- double time_offset

  Time offset of start of FADC memory with respect to shower core crossing observation level [ns].
- double nominal_delay

  Nominal value of the delay of telescope trigger signals at the central station depending on 'global' alt/az and assuming a plane light front propagating with the speed of light in air.
- double telescope_delay

  Telescope-specific delay to be added at the array trigger level, correcting for things like optical transit time (depending on focal length and displacement of reflector w.r.t.
- double pedestal_sysvar

  Systematic variation of pedestals.
- double trigger_time

  Time of first telescope trigger relative to start of FADC memory.
- double trigger_time_by_type [MAX_TRG_TYPES]

  Similar but separate for trigger types.
- double nom_disc_threshold

  Nominal discriminator threshold. [mV].
- double nom_disc_amplitude

  Nominal single p.e. amplitude [mV] at discriminator.
- double nom_disc_gate_length

  Nominal gate length. [ns].
- double nom_disc_delay

  Nominal time over threshold needed. [ns].
- double nom_disc_sigsum_over_thr

  Nominal signal sum required over threshold. [mV∗ns].
- double disc_hysteresis

  Hysteresis of discr./comparator [mV].
- double pe_conversion

  P.e. per (F)ADC peak ampl.
- double trigger_current_limit

  Pixels are excluded from trigger if the.
- double disc_output_amplitude

  Output amplitude of discr./comparator [mV].
- double disc_output_var_percent

  By how many percent the output varies.
- double teltrig_min_time

  Minimum time of sector trigger over minimum [ns].
- double teltrig_min_sigsum

Minimum signal sum at sector trigger.

- double pixeltrg_time_step

  Time step [ns] at which a fired discriminator etc. gets reported (if > 0).
- double multiplicity_offset

  [-0.5...0.5] offset of multiplicity
- int num_trigger_groups

  Number of trigger groups used (0 for NN trigger).
- struct trigger_group * trg_group

  List of trigger groups.
- int groups_required

  Number of fired groups required for tel. trigger.
- int groups_triggered

  Numer of groups triggered.
- int triggered

  No. of triggered pixels.
- int nn_triggered

  No. of next-neighbour triggered pixels.
- int trigger_pixels_req

  No. of pixels required for telescope trigger.
- int trigger_nn_pixels_req

  No. of pixels with a next-neighbour required.
- int telescope_triggered

  Is 1 if the telescope has triggered.
- int telescope_suppressed

  Even though triggered, no data gets recorded.
- int triggered_by_type [MAX_TRG_TYPES]

  Are 1 if triggered by majority/analog sum/digital sum.
- double central_time

  Time of delay-corrected central trigger time.
- double central_time_by_type [MAX_TRG_TYPES]

  Central time separate for trigger types.
- int simulated

  Is 1 if the signal simulation was done.
- int full_simulation

  Configured simulation level: Level 0: Only integral signals.
- int pulse_analysis

  0: no pixel pulse timing analysis, else with.
- int sum_before_peak

  How many slices before peak should be included in sum.
- int sum_after_peak

  How many slices after peak should be included in sum.
- double peak_frac [3]

  Fractions of peak at which rise and fall get evaluated.
- double global_peak_pos

  Position of peak in pulse sum of significant pixels or weighted mean of peak positions of sign.
- int laser_calib_done

  If flatfielding calibration done.
- int laser_external_trigger

  Is 1 if the laser events should assume an external trigger.
- double laser_photons [MAX_LASER_LEVELS]

  Mean number of laser photons per pixel.

- double laser_photons_var [MAX_LASER_LEVELS]

  Common relative fluctuation of photons per laser shot.
- double laser_pulse_offset

  Time offset [ns] of laser pulse in readout window (external trigger).
- double laser_pulse_exptime

  Laser pulse exponential decay time [ns].
- double laser_pulse_sigtime

  Laser pulse duration gaussian shape (r.m.s.) [ns].
- double laser_pulse_twidth

  Laser pulse duration top-hat shape (full width) [ns].
- double laser_wavelength

  Wavelength [nm] at which 'laser' emits light.
- double led_photons

  Mean number of LED photons per pixel.
- double led_photons_var

  Common relative fluctuation of photons per led event.
- double led_pulse_offset

  Time offset [ns] of LED pulses in readout window.
- double led_pulse_sigtime

  LED pulse duration gaussian shape (r.m.s.) [ns].
- double median_time

  Median arrival time of all photons in camera with respect to shower core at obs.
- double phase_delay

  Random, ([-0.5:0.5[)∗fadc_per_channel, delay as part of fadc_delay.
- double fadc_delay

  Delay added, taking into account of random FADC phase at time of arrival.
- double photons_all

  All photons hitting telescope fiducial sphere.
- double photons_atm

  Photons surviving atmospheric transmission.
- double photons_atm_300_550

  Photons surv. atm. tr. in the 300 to 550 nm range.
- double photons_atm_300_600

  Photons surv. atm. tr. in the 300 to 600 nm range.
- double photons_atm_300_650

  Photons surv. atm. tr. in the 300 to 650 nm range.
- double photons_atm_400

  Photons surv. atm. tr. in the 350 to 450 nm range.
- double photons_atm_qe

  Photons surviving atmospheric transmission, mirror reflectivity (except funnel), and max.
- double photons_tel_300_550

  Representing (but not actually counting) the number.
- double photons_cam_300_550

  Representing (but not actually counting) the number.
- double photons_pix_300_550

  Representing photons hitting pixels just before applying QE.
- double photons_signal_ideal

  Sum of ideal signal from photons detected (photo-electrons).
- int photons_detected

  Number of photons detected (photo-electrons).
- int significant_pixels

No. of pixels above zero-suppression cut.

- long **amplitude_histogram** [356]
- double time_profile [MAX_FADC_BINS]

  Filled for each event with the

- double rise_time

  Rise time of time profile [ns].
- double time_fwhm

  Full width at half maximum of profile [ns].
- double ref_fadc_amp

  Reference (F)ADC amplitude.
- double ref_disc_amp

  Reference ampl. at discriminator/comparator.
- double ref_pm_gain

  Reference PM gain factor.
- double ref_fadc_pedestal

  Reference high-gain channel pedestal.
- int ref_fadc_comp_pedestal

  Reference compensated high-gain channel pedestal.
- double ref_fadc_comp_pedestal_err

  Error for compensated high-gain channel pedestal.

7.1.1 Detailed Description

Electronics of a whole camera and its parameters:

Definition at line 1174 of file mc_aux.h.

7.1.2 Field Documentation

7.1.2.1 asum_shape_length

```
int camera_electronics::asum_shape_length
```

Length of signal shaped for analog sum (optional).

[# of bins ∗ DISC_BITS_PER_BIN] (independent of disc_shape)

Definition at line 1240 of file mc_aux.h.

7.1.2.2 asum_shape_offset

```
int camera_electronics::asum_shape_offset
```

Offset where shaping is done, typically offset of peak.

[# of bins ∗ DISC_BITS_PER_BIN]

Definition at line 1242 of file mc_aux.h.

7.1.2.3 collection_efficiency

```
double camera_electronics::collection_efficiency
```

Photoelectron collection efficiency; is included in the quantum_efficiency.

array above for non-NSB light above but is used separately for NSB pixel rates.

Definition at line 1286 of file mc_aux.h.

7.1.2.4 dsum_shape_length

```
int camera_electronics::dsum_shape_length
```

Length of signal shaped for digital sum (optional, high-gain only).

```
# of bins
```

Definition at line 1255 of file mc_aux.h.

7.1.2.5 dsum_shape_offset

```
int camera_electronics::dsum_shape_offset
```

Offset where shaping is done, typically offset of peak.

[# of bins]

Definition at line 1257 of file mc_aux.h.

7.1.2.6 fadc_delay

```
double camera_electronics::fadc_delay
```

Delay added, taking into account of random FADC phase at time of arrival.

[ns]

Definition at line 1397 of file mc_aux.h.

Referenced by delay_signals().

7.1.2.7 full_simulation

```
int camera_electronics::full_simulation
```

Configured simulation level: Level 0: Only integral signals.

Level 1: FADC simulation but not discriminator. Level 2: FADC and discriminator fully simulated.

Definition at line 1366 of file mc_aux.h.

Referenced by fill_traces().

7.1.2.8 global_peak_pos

```
double camera_electronics::global_peak_pos
```

Position of peak in pulse sum of significant pixels or weighted mean of peak positions of sign.

pixels.

Definition at line 1375 of file mc_aux.h.

Referenced by pulse_shape_analysis().

7.1.2.9 laser_external_trigger

```
int camera_electronics::laser_external_trigger
```

Is 1 if the laser events should assume an external trigger.

Use this at least for low intensity like single-p.e. calibration.

Definition at line 1379 of file mc_aux.h.

7.1.2.10 median_time

```
double camera_electronics::median_time
```

Median arrival time of all photons in camera with respect to shower core at obs.

level. [ns]

Definition at line 1394 of file mc_aux.h.

7.1.2.11 multiplicity_offset

```
double camera_electronics::multiplicity_offset
```

[-0.5...0.5] offset of multiplicity

threshold w.r.t. nominal trigger condition

Definition at line 1348 of file mc_aux.h.

7.1.2.12 optics_efficiency

```
double camera_electronics::optics_efficiency[MAX_LAMBDA]
```

Everything before camera window, is part of

quantum_efficiency[] above.

Definition at line 1288 of file mc_aux.h.

7.1.2.13 photons_atm_qe

```
double camera_electronics::photons_atm_qe
```

Photons surviving atmospheric transmission, mirror reflectivity (except funnel), and max.

Q.E.

Definition at line 1405 of file mc_aux.h.

Referenced by convert_mc_pe_sum().

7.1.2.14 photons_cam_300_550

```
double camera_electronics::photons_cam_300_550
```

Representing (but not actually counting) the number.

of photons (after optics) in the 300 to 550 nm range hitting pixels.

Definition at line 1409 of file mc_aux.h.

7.1.2.15 photons_tel_300_550

```
double camera_electronics::photons_tel_300_550
```

Representing (but not actually counting) the number.

of photons (before optics) in the 300 to 550 nm range hitting pixels.

Definition at line 1407 of file mc_aux.h.

7.1.2.16 quantum_efficiency

```
double camera_electronics::quantum_efficiency[MAX_LAMBDA]
```

Q.E.

times mirror reflectivity times camera transmission as a vector where the index is wavelength in steps of nanometers.

Definition at line 1281 of file mc_aux.h.

7.1.2.17 shape

```
double camera_electronics::shape[OVERSAMPLING *MAX_SHAPE_LENGTH]
```

Single photoelectron.

pulse shape with peak set to 1.

Definition at line 1213 of file mc_aux.h.

Referenced by fill_traces().

7.1.2.18 telescope_delay

```
double camera_electronics::telescope_delay
```

Telescope-specific delay to be added at the array trigger level, correcting for things like optical transit time (depending on focal length and displacement of reflector w.r.t.

the centre of the fiducial sphere), ... For comparing trigger times of different telescope types, some calibration may be needed.

Definition at line 1314 of file mc_aux.h.

7.1.2.19 teltrig_min_sigsum

```
double camera_electronics::teltrig_min_sigsum
```

Minimum signal sum at sector trigger.

over threshold [pV∗s].

Definition at line 1344 of file mc_aux.h.

7.1.2.20 time_profile

```
double camera_electronics::time_profile[MAX_FADC_BINS]
```

Filled for each event with the

n hottest non-overflow significant pixels (n <= PROFILE_PIXELS).

Definition at line 1419 of file mc_aux.h.

7.1.2.21 trigger_current_limit

```
double camera_electronics::trigger_current_limit
```

Pixels are excluded from trigger if the.

current is above this threshold [uA].

Definition at line 1338 of file mc_aux.h.

7.1.2.22 trigger_time

```
double camera_electronics::trigger_time
```

Time of first telescope trigger relative to start of FADC memory.

[ns]

Definition at line 1325 of file mc_aux.h.

Referenced by pulse_shape_analysis(), and sum_adc_bins().

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

- mc_aux.h

7.2 camera_image_plot_param Struct Reference

Some parameters for image plots.

```
#include <mc_aux.h>
```

Data Fields

- double range
  Dynamic range scale [p.e.].
- double gamma
  Gamma correction coefficient.

7.2.1 Detailed Description

Some parameters for image plots.

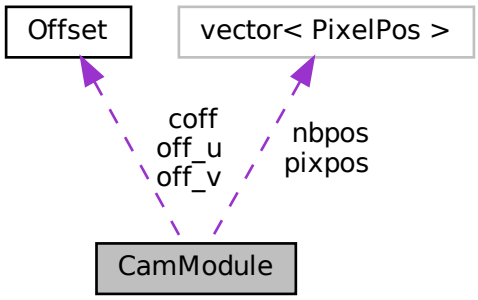
Definition at line 1528 of file mc_aux.h.

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

- mc_aux.h

7.3 CamModule Class Reference

Collaboration diagram for CamModule:



Public Member Functions

- size_t **size** (void) const
- size_t **num_nb** (void) const
- PixelPos **Position** (size_t inmod) const
- PixelPos **Neighbour** (size_t inmod) const
- const Offset & **Offset_u** () const
- const Offset & **Offset_v** () const
- int **Assign** (int m, int p)
- void **PrintPixels** ()
- PixelList **List** () const
- double **Distance** (const CamModule &m) const
- int **PixelType** () const
- const Offset & **CentralOffset** () const
- void **CheckNB** (const CamModule &cm2, int ic2)
- int **ModuleId** () const
- int **PixelId** () const
- int **NBsqr** (int ix, int iy) const
- **CamModule** (Offset c, double scale, MPG &g, double sc_u, double sc_v)

Private Attributes

- vector< PixelPos > **pixpos**
- vector< PixelPos > **nbpos**
- Offset **off_u**
- Offset **off_v**
- int **mod_id**
- int **pix_id**
- int **pixel_type**
- Offset **coff**
- double **mod_scale_u**
- double **mod_scale_v**
- int **nb_sqr** [3][3]

7.3.1 Detailed Description

Definition at line 606 of file modular_camera.cc.

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

- modular_camera.cc

7.4 channel_calibration Struct Reference

Calibration parameters for a PM and its electronics channel(s):

`#include <mc_aux.h>`

Data Fields

- double pedestal_sum
  Pedestal sum of all FADCs per channel.
- double sigma_pedestal_sum
  Variation of that sum.
- double pedestal [MAX_PER_CHANNEL]
  Pedestals in individual FADCs.
- double pedestal_nsb [MAX_PER_CHANNEL]
  Pedestals in individual FADCs including possible DC shift by NSB.
- double sigma_pedestal [MAX_PER_CHANNEL]
  Variation of pedestal bin contents.
- double laser
  Laser amplitude.
- double sigma_laser
  Variation of laser amplitude.
- double laser_time
  Average time offset of laser pulses. [ns].
- double calib_rel
  Flatfielding calibration (relative to other pixels).
- double amp_to_npe
  Amplitude to #(p.e.) conversion.
- int pedestal_comp_add
  Value added to high-gain FADC values to achieve compensated pedestal.
- double pedestal_comp_err
  Error introduced in calculating the compensation for high-gain.

7.4.1 Detailed Description

Calibration parameters for a PM and its electronics channel(s):

Definition at line 990 of file mc_aux.h.

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

- mc_aux.h

7.5 cmdline_conf Struct Reference

Configuration from command line ('-C' or '-W' option) is saved for re-use.

Collaboration diagram for cmdline_conf:

Data Fields

- char * conf
  Pointer to saved configuration text.
- struct cmdline_conf * next
  Pointer to next safed configuration.

7.5.1 Detailed Description

Configuration from command line ('-C' or '-W' option) is saved for re-use.

Definition at line 1312 of file sim_config.c.

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

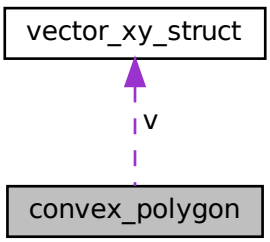
- sim_config.c

7.6 convex_polygon Struct Reference

A convex polygon with no positions showing up twice, except first=last, and all turns being right turns or all turns being left turns (no zero or 180 deg turns).

`#include <mc_aux.h>`

Collaboration diagram for convex_polygon:



Data Fields

- int rtp

  Right turns in polygon (clock-wise): 1, left turns: -1.
- size_t np

  Number of points for polygon, not counting the last one being the same as the first.
- VectorXY * v

  All the corners from first point back to first=last point.
- double **xc**
- double yc

  Assumed center of polygon [cm].
- double r2max

  Maximum distance squared of corners from center [cm^2].

7.6.1 Detailed Description

A convex polygon with no positions showing up twice, except first=last, and all turns being right turns or all turns being left turns (no zero or 180 deg turns).

Definition at line 1622 of file mc_aux.h.

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

- mc_aux.h

7.7 cubic_params Struct Reference

Cubic spline interpolation (natural cubic splines = scheme 3, clampled cubic splines = scheme 4)

`#include <rpolator.h>`

Data Fields

- double **a**
- double **b**
- double **c**
- double d

  *r=xp-x[i], yp = a + b*r + c*r^2 + d*r^3 = ((d*r + c) * r + b) * r + a;*

7.7.1 Detailed Description

Cubic spline interpolation (natural cubic splines = scheme 3, clampled cubic splines = scheme 4)

Definition at line 43 of file rpolator.h.

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

- rpolator.h

7.8 effcalc Struct Reference

Collect original and derived values in nanometer steps, even if (presently) not wavelength dependent.

Data Fields

- int iwl

  Wavelength in nanometers (enforced step width of 1 nm)
- double qe_only

  Quantum efficiency.
- double ref_only

  Mirror reflection (effective ref1∗ref2 for dual-mirror)
- double qe_ref

  Accounting for both reflection and quantum efficiency.
- double tel_trans

  Telescope transmission factor for non-explicit shadowing.
- double degraded

  Degradation factor (assumed mirror but not camera)
- double efilter

  Efficiency in camera entrance.
- double efunnel

  Efficiency in light concentrators (funnels)
- double e0

  Camera efficiency (no reflectivity, no shadowing)
- double e1

  Telescope efficiency (mirrors, shadowing, camera)
- double e2

  Total efficiency (atmosphere and telescope)
- double atm_trans_inf

  Atmospheric transmission for starlight ('infinite' distance = top of atmosphere, at given z.a.)
- double atm_trans_chl

  Atmospheric transmission for Cherenkov light from given atmospheric depth, at given z.a.
- double ch_pe

  Contribution to p.e. from Cherenkov light at given wavelength (per 1 nm)
- double **nsb_pe_rate_raw**
- double **nsb_pe_rate**
- double **alb_pe_rate_raw**
- double **alb_pe_rate**
- double nsblight

  NSB flux after site correction, (airglow) zenith angle scaling and correction for inclined path.
- double nsblight_eff

  Efficiency applied for NSB but not geometric factors.
- double nsblight0

  Original NSB flux value from table.
- double nsblight0_eff

  Efficiency applied but not geometric factors (original NSB)
- double nsb_from_fnu0

  Original NSB flux value derived from the f_nu column. Supposed to match nsblight0.
- double **nsb_from_fnu_eff**
- double atm_corr

Correction factor applied.

- double nsb_scaled

  Scaling factor applied.

- double mirror_area

  Mirror area, either from own evaluation (no camera etc. shadowing at all) or as given.

- double pixel_area

  Pixel active area, not counting any gaps between pixels [cm^ 2].

- double solid_angle

  Solid angle of sky imaged onto pixel active area [sr].

- double alb_wl

  Albedo = diffuse reflection on ground, either fixed value or w.l. dependent.

- double alb_solid_angle

  Effective solid angle for albedo contribution to NSB, from angles beyond edge of mirror coverage.

7.8.1 Detailed Description

Collect original and derived values in nanometer steps, even if (presently) not wavelength dependent.

Definition at line 187 of file testeff.c.

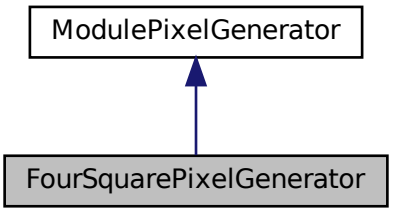
The documentation for this struct was generated from the following file:

- testeff.c

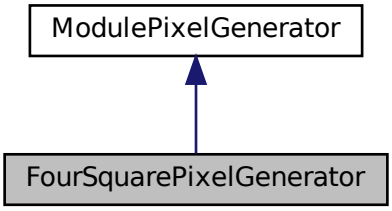
7.9 FourSquarePixelGenerator Class Reference

Four square pixels.

Inheritance diagram for FourSquarePixelGenerator:



Collaboration diagram for FourSquarePixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.9.1 Detailed Description

Four square pixels.

Definition at line 483 of file modular_camera.cc.

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

- modular_camera.cc

7.10 imaging_setup Struct Reference

The setup of telescope specific things.

Data Fields

- double source_azimuth

  Azimuth angle of source or 0.
- double source_altitude

  Altitude angle of source or 0.
- double reference_position [3]

  Reference position with respect to the observation level.
- int base_telescope_number

  Number of first telescope, e.g.

- double telescope_phi

  Azimuth angle of the telescope(s) Unit: degrees from north towards east.
- double telescope_theta

  Zenith angle of the telescope Unit: degrees.
- double telescope_random_angle

  Random (known) misalignment of the telescope in each axis.
- double telescope_random_error

  Random (unknown) alignment error of the telescope in each axis.
- int reverse_mode

  Is 1 if reverse positioning is used.
- double random_viewing_ring [2]

  Minimum and maximum radius of ring [degrees] around basic phi/theta direction to which system viewing direction is randomized.
- int bypass_optics

  Bypass or partially bypass optics ray-tracing if non-zero.
- char optics_config_name [100]

  Meta-parameter (has no effect)
- char optics_config_version [100]

  Meta-parameter (has no effect)
- char optics_config_variant [100]

  Meta-parameter (has no effect)
- int mirrors

  Number of mirrors on the telescope.
- int parabolic_dish

  Outline of dish parabolic or Davies-Cotton?
- int flip_mirrors

  Flip mirror x <-> y (and hex type 1 <-> 3).
- double focal_length

  Nominal focal length of overall reflector (mechanical or as provided).
- double effective_focal_length [5]

  Effective focal length(s) of overall reflector and image displacements: mean flen / flen for x / flen for y / displacements in x/y.
- double effective_mirror_area

  Meta-parameter (has no effect in sim_telarray but may be extracted by testeff). Unit: cm^ 2.
- double dish_shape_length

  Dish curvature length, best equal to focal length.
- double mirror_flen

  Standard focal length of mirror tiles.
- double mirror_flen_grading

  Gradual change of focal length from inner to outer mirrors.
- double mirror_flen_random [2]

  Random error in focal lengths (sigma, max) [cm].
- double mirror_flen_scale [MAX_MIRRORS]

  List of focal length scaling factors.
- double focus_offset [4]

  Distance of the starlight focus from the camera pixels (lightguides) [cm], and its zenith angle dependence as for the mirror alignment errors.
- double mirror_x [MAX_MIRRORS]

  X position of all mirrors. Unit: cm.
- double mirror_y [MAX_MIRRORS]

  Y position of all mirrors. Unit: cm.

- double mirror_diameter [MAX_MIRRORS]

  Diameters of all mirrors. Unit: cm.
- double mirror_type [MAX_MIRRORS]

  Type: 0 (circ.), 1 (hex, flat -> x), 2 (hex, flat edge -> y).
- double mirror_offset

  Offset of mirror backplane from fixed point of telescope mount (positive if fixed point is between mirror and focus).
- double axes_offset [2]

  The first value is a horizontal offset between the (vertical) azimuth axis and the (horizontal) altitude rotation axis.
- double mirror_rnd_align_hori [4]

  Fluctuations of the horizontal mirror alignment angle with respect to nominal alignment.
- double mirror_rnd_align_vert [4]

  Same as before but for the vertical projection.
- double mirror_rnd_align_distance

  Fluctuations in the aligned distance from the focus.
- double mirror_dc_opt [3]

  Optimisation parameters relative to simple Davies-Cotton design.
- double mirror_rnd_ref_angle [3]

  Gaussian random fluctuations of reflection angles, due to small-scale surface deviations, with one or two components.
- double mirror_degraded_reflection

  Mirror reflectivity (primary, to be accurate) is by the given factor worse than the nominal table (at all wavelengths).
- double mirror2_degraded_reflection

  Secondary mirror (if any) is by given factor worse than the nominal table (at all wavelengths).
- int mirror_class

  Segmented (0), paraboloid (1), general solid prim.+sec. (2), Fresnel lens (3)
- double primary_diameter

  Can be used instead of mirror list file to set primary mirror diameter. [cm].
- double secondary_diameter

  Can be used instead of mirror list file to set secondary mirror diameter. [cm].
- double secondary_shadow_diameter

  Effective diameter of secondary for shadowing (-1: use secondary_diameter). [cm].
- double secondary_shadow_offset

  z position of secondary shadowing element (0: automatic from polynomial). [cm]
- double primary_hole

  Diameter of a hole in the centre of the primary mirror (mirror_class==2 only).
- double secondary_hole

  Diameter of a hole in the centre of the secondary mirror (mirror_class==2 only).
- double secondary_baffle [5]

  z1, z2, R1, dR, R2; for now R2=0. or R2=R1 (cylinder only, cone coming later). [cm]
- double primary_parameters [MAX_NPAR_MIRR]

  Parameters describing a general primary. Only for mirror_class==2.
- double primary_ref_radius

  Radius used in these units when applying parameters. Unit: cm.
- double secondary_parameters [MAX_NPAR_MIRR]

  Parameters describing a general secondary. Only for mirror_class==2.
- double secondary_ref_radius

  Radius used in these units when applying parameters. Unit: cm.
- double focal_surface_parameters [MAX_NPAR_MIRR]

  Parameters describing the shape of the focal surface. Only for mirror_class==2.
- double focal_surface_ref_radius

  Radius used in these units when applying parameters. Unit: cm.
- int pixels_parallel

Defines whether pixels are aligned parallel to the optical axis (1) or are inclined normal to the focal surface (0).

- char prm_segments_fname [MAXFN]

  Name of file with definition of sets of segments of the primary mirror (mirror_class=2 only).
- char sec_segments_fname [MAXFN]

  Name of file with definition of sets of segments of the secondary mirror (mirror_class=2 only).
- double lens_refidx_nominal

  Nominal index of refraction of Fresnel lens material.
- char camera_config_name [100]

  Meta-parameter (has no effect)
- char camera_config_version [100]

  Meta-parameter (has no effect)
- char camera_config_variant [100]

  Meta-parameter (has no effect)
- int camera_type

  Camera geometry type (1: hexagonal (obsolete), 2: square (obsolete), 3: custom).
- int camera_pixels

  Number of pixels in camera (allowed values depend on geometry type).
- int camera_body_shape

  Shape of camera body for shadowing: 0 (circular, default), 1,3: hex, 2:square.
- double camera_body_diameter

  Diameter of (by default circular) camera body (for shadowing).
- double camera_depth

  Depth of the camera body (for shadowing). Unit: cm.
- double telescope_transmission [MAX_TELTRANS]

  Account for absorption by masts etc. (constant or function of tan(off-axis angle)
- double camera_transmission

  Global transmission factor of camera (including the plexiglass window).
- char camera_filter_fname [MAXFN]

  Name of file with wavelength dependence of camera window transmission or other component acting as a filter (for effective angle of incidence).
- double lightguide_reflectivity

  Reflectivity of the lightguide material in front of the camera.
- double camera_degraded_efficiency

  Actual efficiency of camera window, funnels, PMTs is lower than configured if this is set to a value less than 1.0, independent of position.
- int dead_pixels [MAX_DEAD_PIXELS]

  < Note: quantum efficiency is separate.
- double dead_pix_prob

  Probability of a pixel randomly assigned as dead.
- double dead_board_prob

  Probability of all pixels readout by the same board assigned as dead.
- double dead_module_prob

  Probability of all pixels in the same module assigned as dead.
- double collection_efficiency

  Photoelectron collection efficiency in the photomultiplier first stage.
- char * spe_fname

  Name of file with single photo electron amplitude spectrum.
- int nspe

  Size of lookup table for creating random numbers according to the given single p.e.
- double adjust_gain

  Common multiplicative adjustment for FADC and discr.

- double gain_variation

  By how the gain may vary between PMs after the voltage has been adjusted for approximately the same gains.
- double qe_variation

  Quantum efficiency variation between PMs.
- double pm_gain

  Only used for DC current.
- int flatfielding

  Is 1 if camera is flatfielded (in gain∗QE), 0 if not (only gain adjusted).
- int pixel_cells

  For SiPM, if saturation by cells already hit is to be taken into account.
- double pm_transit_time [4]

  0: Total transit time of the PM [ns] at nominal voltage.
- double transit_time_jitter

  Jitter in nanoseconds, separately for each photo-electron.
- double pm_voltage_variation

  Variations of transit times are usually caused by needing different voltages to reach the same gains.
- double pm_gain_index

  Gain rises as given power of the PM voltage.
- double transit_err

  Error in any compensation of transit time [ns].
- double transit_calib_err

  Accuracy with which transit delay + compensation can be known [ns].
- double transit_comp_step

  Step size in electronic delay compensation [ns].
- double transit_comp_err

  Errors made in evaluating necessary electronic delay compensation [ns].
- int afterpulse_alt

  If 1, use the afterpulse spectrum defined by the following parameters as an alternative to those in the third column of the file defined by spe_fname.
- int num_gains

  Number of gains per pixel, allowing to simulate just a single gain for one telescope but dual gains for another.
- int fadc_bins

  No. of FADCs bins to be filled.
- int fadc_per_channel

  How many FADCs work in parallel with corrensping delays.
- int channels_per_chip

  Number of readout channels per chip.
- int disc_bins

  Number of time bins used for discr./comparator simulation.
- int disc_start

  No. of bins by which discr./comp. sim. is ahead of FADC.
- int fadc_sum_bins

  Number of bins summed up in ADC sum data.
- int fadc_sum_offset

  Number of bins before telescope trigger where summing starts.
- int fadc_longsum_bins

  Number of bins summed up in ADC sum data for long events.
- int fadc_longsum_offset

  Number of bins before telescope trigger where summing starts in long events.
- int fadc_max_signal

  Maximum possible ADC signal amplitude per interval (ADC counts); 0: built-in limit (MAX_FADC_SIGNAL) applies.

- int fadc_max_sum

  Maximum of recorded sum of ADC signals; 0: no additional limit (i.e. MAX_FADC_SIGNAL∗no. of bins).
- int fadc_max_signal_lg

  Max. signal for low-gain channel; if <0 then same limit as for high-gain applies.
- int fadc_max_sum_lg

  Max. sum for low-gain channel; if <0 then same limit as for high-gain applies.
- double trigger_delay_compensation [MAX_TRG_TYPES]

  An additional delay specific for each type of trigger.
- double photon_delay

  Additional delay added to photon arrival times. Unit: ns.
- double fadc_frequency

  FADC frequency. Unit: MHz.
- double fadc_noise

  Gaussian noise in digitisation.
- double fadc_noise_lg

  Gaussian noise in digitisation.
- double fadc_amplitude

  Signal amplitude in mV for single photoelectrons after PM, preamplifier, cable, and shaper at the inputs of the FADC.
- double fadc_amplitude_lg

  Same for low gain channel (if any).
- double hglg_variation

  Relative variation of high-gain-over-low-gain from pixel to pixel w.r.t.
- double disc_amplitude

  Similar after amplifier at the input of the discriminators/comparators.
- double disc_threshold

  Discrim./compar. threshold. Unit: mV.
- double disc_threshold_variation

  Channel to channel variations [mV].
- double disc_scale_threshold

  Scale configured disc./comp. threshold by this factor.
- double disc_gate_length

  Effective discr. gate length [ns].
- double disc_gate_length_variation

  and variation of it [ns].
- double disc_time_over_threshold

  Time over threshold required [ns].
- double disc_time_over_thr_var

  Pixel-to-pixel variation of it. [ns].
- double disc_sigsum_over_thr

  Signal sum over threshold required before discriminator/comparator switching.
- double disc_sigsum_over_thr_var

  Pixel-to-pixel variation of it.
- double disc_rise_time

  Rise time of discr./comp. output. [ns].
- double disc_fall_time

  Fall time of discr./comp. output. [ns].
- double disc_hysteresis

  Hysteresis of comparator switching. [mV].
- double disc_output_amplitude

  Output amplitude of discr./comparator [mV].
- double disc_output_var_percent

By how many percent the output varies.

- double pixeltrg_time_step

  Time step [ns] in which time of triggering a pixel (firing a discriminator or such) is to be recorded, if at all.
- double teltrig_min_time

  Minimum time of sector trigger over threshold (flexible camera only) [ns].
- double teltrig_min_sigsum

  Minimum signal sum at sector trigger over threshold [pV∗s].
- double fadc_pedestal

  Nominal FADC pedestal value.
- double fadc_pedestal_dev

  Deviation of FADCs for same channel.
- double fadc_pedestal_var

  Channel-to-channel variation.
- double fadc_pedestal_err

  Assumed error in initial calibration.
- double fadc_pedestal_sysvar

  Systematic (e.g.
- double fadc_pedestal_lg

  Nominal low-gain FADC pedestal value (if different from HG)
- double fadc_pedestal_dev_lg

  Deviation of low-gain FADCs for same channel (if different from HG).
- double fadc_pedestal_var_lg

  Channel-to-channel variation for LG (if different from HG).
- double fadc_pedestal_err_lg

  Assumed error in initial calibration for LG (if different from HG).
- double fadc_pedestal_sysvar_lg

  Systematic (e.g.
- double fadc_sensitivity

  FADC counts per mV voltage.
- double fadc_sensitivity_variation

  Relative variations in sensitivity (even for FADCs of the same channel).
- double fadc_sensitivity_lg

  FADC counts per mV voltage for low-gain channel (if different from HG)
- double fadc_sensitivity_variation_lg

  Relative variations in sensitivity for low-gain channel (even for FADCs of the same channel).
- int fadc_comp_pedestal

  Compensated nominal pedestal value, after readout plus individual compensation offset.
- int fadc_comp_pedestal_lg

  Low-gain channel compensated nominal pedestal value.
- double fadc_comp_pedestal_err

  R.m.s.
- double fadc_comp_pedestal_err_lg

  R.m.s.
- int fadc_ac_coupled

  Set to 1 if FADCs are AC coupled.
- int disc_ac_coupled

  Similar for the discriminators.
- double nightsky_background [MAX_PIXELS]

  Number of photoelectrons per nanosecond.
- double nsb_scaling_factor

  Common scaling factor for all telescopes (global parameter).

- double nsb_autoscale_airmass [2]

  Allow for automatic NSB scaling unless manual scaling factor is set (and not 1.0).
- double nsb_offaxis_factor [MAX_TELTRANS]

  Fall-off of NSB with off-axis angle, including explicit and implied shadowing.
- double nsb_gain_drop_scale

  NSB level at which gain drops to half of nominal value. [GHz].
- double laser_photons [MAX_LASER_LEVELS]

  Number of 'laser' (or other light source) photons at each PM.
- double laser_photons_var [MAX_LASER_LEVELS]

  Relative variation of 'laser' photons (shot to shot).
- int laser_events [MAX_LASER_LEVELS]

  Number of 'laser' events at start of run.
- double laser_pulse_offset

  Time offset [ns] of laser pulse in readout window (external trigger).
- double laser_pulse_exptime

  Laser pulse exponential decay time [ns].
- double laser_pulse_sigtime

  Laser pulse duration gaussian shape (r.m.s.) [ns].
- double laser_pulse_twidth

  Laser pulse duration top-hat shape (width) [ns].
- double laser_wavelength

  Wavelength [nm] at which 'laser' emits light.
- double led_photons

  Number of in-lid LED photons at each PM.
- double led_photons_var

  Relative variation of LED photons.
- double led_pulse_offset

  Time offset [ns] of LED pulses in readout window.
- double led_pulse_sigtime

  In-lid LED pulse duration gaussian shape (r.m.s.) [ns].
- int laser_external_trigger

  0: need normal telescope trigger, 1: external trigger
- int led_events

  In-lid LED events at start of run.
- int closed_pedestal_events

  Pedestal events at start of run.
- int opened_pedestal_events

  with camera lid closed/opened.
- double simple_threshold

  Simple threshold.
- double trigger_current_limit

  Pixels above this limit are excluded from the trigger.
- int trigger_pixels

  Number of pixels required for single telescope trigger.
- int trigger_neighboured

  Number of neighboured pixels required for single telescope trigger.
- int trigger_telescopes

  Number of telescopes required for the system trigger.
- int telescope_ignore [MAX_IGNORE]

  List of telescopes ignored (numbers starting at 1, i.e.
- int full_simulation

Full simulation of FADC signals if > 0, of discriminators if > 1.

- int pulse_analysis

  0: no pulse timing analysis, else with.
- int sum_before_peak

  How many slices before peak should be included in sum.
- int sum_after_peak

  How many slices after peak should be included in sum.
- double multiplicity_offset

  [-0.5...0.5] offset of mult.
- char default_trigger_type [32]

  Majority, AnalogSum, or DigitalSum.
- char asum_shaping_fname [MAXFN]

  Name with pre-sum shaping of analog signal for sum trigger.
- char dsum_shaping_fname [MAXFN]

  Name with pre-sum shaping of digital signal for sum trigger.
- double asum_clipping

  If single pixel signal exceeds this level after shaping, clip it for analog sum. [mV].
- double asum_threshold

  Threshold when an analog sum results in a trigger. [mV].
- double asum_shape_offset

  Additional offset in time of shaping for analog sum. [ns].
- double dsum_shape_offset

  Additional offset in time of shaping for digital sum. [ns].
- int dsum_shaping_renorm

  If non-zero, the (positive part) of the shaping kernel will be normalized.
- int dsum_clipping

  If single pixel signal exceeds this level after shaping, clip it for digital sum. [ADC counts].
- int dsum_pre_clipping

  If single pixel signal exceeds this level before shaping, clip it for digital sum. [ADC counts].
- double dsum_threshold

  Threshold when a digital sum results in a trigger.
- int dsum_ignore_below

  If > 0, use shaped clipped signal only where exceeding this value. [ADC counts].
- int dsum_pedsub

  If non-zero, the pedestal (integer) is subtracted before shaping.
- int dsum_zero_clip

  If non-zero, clip negative values after shaping at zero, before further summation.
- int dsum_prescale [2]

  After shaping and before clipping and possible pre-summation, an integer multiplication[0]/division[1] is applied as pre-scaling.
- int dsum_presum_max

  Before a final bit shift the patch-wise pre-sum cannot be larger than that.
- int dsum_presum_shift

  After pre-summation and clipping a (right) bit-shift may be applied.
- double random_mono_prob

  Randomly choose events to pass through stereo selection even if mono.
- int muon_mono_threshold [2]

  Thresholds on number of trigger groups and trigger regions (collections of groups) for muon-ring-enhanced events acceptable as mono.
- int long_event_threshold

  Thresholds on number of trigger groups for testing the long event case (actual conditions likely to change).

- int fake_trigger

  Fake a trigger decision and time, for nth p.e. if n>0, for median if n==-1.
- int force_fake_trigger

  If one, fake trigger forced to even trigger with fake_trigger=0.
- double fake_trg_delay

  Time added to derived fake trigger time since start of simulation window.
- char camera_config_fname [MAXFN]

  Additional configuration file for more flexible camera configuration.
- double camera_scale_factor

  Scale all pixel+camera dimensions.
- double image_pe_range

  Range scale for image colours.
- double image_gamma_coeff

  Contrast parameter.
- int output_format

  Select a specific output format (if several available).
- int zero_suppression

  Zero-suppression mode.
- int data_reduction

  Data reduction mode.
- int peak_sensing

  If > 0 use peak sensing of given window size instead of ADC sum.
- double min_photons

  Minimum no. of photons required before doing simulation.
- int min_photoelectrons

  Minimum no. of photonelectrons required for electronics simulation.
- int store_photoelectrons

  Store p.e. data if more than that number are registered (-1: never registered).
- double tailcut_scale

  Tail-cut procedure threshold scale factor.

7.10.1 Detailed Description

The setup of telescope specific things.

Definition at line 104 of file sim_config.c.

7.10.2 Field Documentation

7.10.2.1 adjust_gain

```
double imaging_setup::adjust_gain
```

Common multiplicative adjustment for FADC and discr.

ampl. and PM gain (for small tel. to tel. variations).

Definition at line 271 of file sim_config.c.

7.10.2.2 afterpulse_alt

```
int imaging_setup::afterpulse_alt
```

If 1, use the afterpulse spectrum defined by the following parameters as an alternative to those in the third column of the file defined by spe_fname.

If zero, additional afterpulse signals are generated.

Definition at line 301 of file sim_config.c.

7.10.2.3 axes_offset

```
double imaging_setup::axes_offset[2]
```

The first value is a horizontal offset between the (vertical) azimuth axis and the (horizontal) altitude rotation axis.

A positive value corresponds to an altitude axis towards the reflector. If the altitude axis is on the optical axis, the second parameter is zero. Otherwise it represents the displacement perpendicular to the optical axis and the altitude axis of the altitude rotation axis w.r.t to the reflector. If both values are non-zero but equal, the optical axis coincides with the azimuth axis for vertical pointing.

Definition at line 164 of file sim_config.c.

7.10.2.4 base_telescope_number

```
int imaging_setup::base_telescope_number
```

Number of first telescope, e.g.

2 if starting with CT2.

Definition at line 114 of file sim_config.c.

7.10.2.5 camera_body_diameter

```
double imaging_setup::camera_body_diameter
```

Diameter of (by default circular) camera body (for shadowing).

Unit: cm

Definition at line 244 of file sim_config.c.

7.10.2.6 camera_degraded_efficiency

```
double imaging_setup::camera_degraded_efficiency
```

Actual efficiency of camera window, funnels, PMTs is lower than configured if this is set to a value less than 1.0, independent of position.

Splitting up camera and mirror(s) degradation relevant with bypass_optics.

Definition at line 254 of file sim_config.c.

7.10.2.7 camera_transmission

```
double imaging_setup::camera_transmission
```

Global transmission factor of camera (including the plexiglass window).

For spectral/angular dependence use filter.

Definition at line 248 of file sim_config.c.

7.10.2.8 dead_pixels

```
int imaging_setup::dead_pixels[MAX_DEAD_PIXELS]
```

< Note: quantum efficiency is separate.

List of broken pixels.

Definition at line 258 of file sim_config.c.

7.10.2.9 disc_sigsum_over_thr

```
double imaging_setup::disc_sigsum_over_thr
```

Signal sum over threshold required before discriminator/comparator switching.

[mV∗ns]

Definition at line 353 of file sim_config.c.

7.10.2.10 dish_shape_length

```
double imaging_setup::dish_shape_length
```

Dish curvature length, best equal to focal length.

Unit: cm. Normally 0 to adapt it to focal length.

Definition at line 143 of file sim_config.c.

7.10.2.11 dsum_threshold

```
double imaging_setup::dsum_threshold
```

Threshold when a digital sum results in a trigger.

[ADC counts] Actually only using the integer part; double needed for RandomRates scripts

Definition at line 457 of file sim_config.c.

7.10.2.12 effective_focal_length

```
double imaging_setup::effective_focal_length[5]
```

Effective focal length(s) of overall reflector and image displacements: mean flen / flen for x / flen for y / displacements in x/y.

(for image scale). May be zero if not known. Unit: cm.

Definition at line 139 of file sim_config.c.

7.10.2.13 fadc_comp_pedestal

```
int imaging_setup::fadc_comp_pedestal
```

Compensated nominal pedestal value, after readout plus individual compensation offset.

Per sample. Negative values indicate that no compensation gets applied.

Definition at line 385 of file sim_config.c.

7.10.2.14 fadc_comp_pedestal_err

```
double imaging_setup::fadc_comp_pedestal_err
```

R.m.s.

error in finding the necessary offset to get from the actual pedestal to the compensated one.

Definition at line 390 of file sim_config.c.

7.10.2.15 fadc_comp_pedestal_err_lg

```
double imaging_setup::fadc_comp_pedestal_err_lg
```

R.m.s.

error in finding the necessary offset to get from the low-gain actual pedestal to the compensated one. Values below zero indicate to use the high-gain channel value.

Definition at line 392 of file sim_config.c.

7.10.2.16 fadc_comp_pedestal_lg

```
int imaging_setup::fadc_comp_pedestal_lg
```

Low-gain channel compensated nominal pedestal value.

Values below zero indicate to use the high-gain channel value.

Definition at line 388 of file sim_config.c.

7.10.2.17 fadc_pedestal_sysvar

```
double imaging_setup::fadc_pedestal_sysvar
```

Systematic (e.g.

due to temperature) variation of baselines.

Definition at line 371 of file sim_config.c.

7.10.2.18 fadc_pedestal_sysvar_lg

```
double imaging_setup::fadc_pedestal_sysvar_lg
```

Systematic (e.g.

due to temperature) variation of baselines for LG (if different from HG).

Definition at line 377 of file sim_config.c.

7.10.2.19 fadc_sensitivity_variation_lg

```
double imaging_setup::fadc_sensitivity_variation_lg
```

Relative variations in sensitivity for low-gain channel (even for FADCs of the same channel).

Only used if different from HG.

Definition at line 383 of file sim_config.c.

7.10.2.20 focal_length

```
double imaging_setup::focal_length
```

Nominal focal length of overall reflector (mechanical or as provided).

Unit: cm.

Definition at line 137 of file sim_config.c.

7.10.2.21 gain_variation

```
double imaging_setup::gain_variation
```

By how the gain may vary between PMs after the voltage has been adjusted for approximately the same gains.

Unit: fraction.

Definition at line 274 of file sim_config.c.

7.10.2.22 hglg_variation

```
double imaging_setup::hglg_variation
```

Relative variation of high-gain-over-low-gain from pixel to pixel w.r.t.

nominal ratio.

Definition at line 342 of file sim_config.c.

7.10.2.23 lens_refidx_nominal

```
double imaging_setup::lens_refidx_nominal
```

Nominal index of refraction of Fresnel lens material.

Only for mirror_class==3.

Definition at line 231 of file sim_config.c.

7.10.2.24 mirror2_degraded_reflection

```
double imaging_setup::mirror2_degraded_reflection
```

Secondary mirror (if any) is by given factor worse than the nominal table (at all wavelengths).

Assigning a separate factor from the primary only relevant for bypass_optics==1.

Definition at line 201 of file sim_config.c.

7.10.2.25 mirror_flen

```
double imaging_setup::mirror_flen
```

Standard focal length of mirror tiles.

Unit: cm. Normally 0 to adapt it to focal length.

Definition at line 145 of file sim_config.c.

7.10.2.26 mirror_offset

```
double imaging_setup::mirror_offset
```

Offset of mirror backplane from fixed point of telescope mount (positive if fixed point is between mirror and focus).

Unit: cm.

Definition at line 160 of file sim_config.c.

7.10.2.27 mirror_rnd_align_distance

```
double imaging_setup::mirror_rnd_align_distance
```

Fluctuations in the aligned distance from the focus.

Unit: cm.

Definition at line 187 of file sim_config.c.

7.10.2.28 mirror_rnd_align_hori

```
double imaging_setup::mirror_rnd_align_hori[4]
```

Fluctuations of the horizontal mirror alignment angle with respect to nominal alignment.

These depend on the zenith angle / altitude. The terms here are [0]: the minimum alignment error [1]: the zenith angle where minimum is achieved (i.e. where alignment is done) [2]: term proportional to difference in sine(altitude) [3]: term proportional to difference in cosine(altitude)

Definition at line 175 of file sim_config.c.

7.10.2.29 mirror_rnd_ref_angle

```
double imaging_setup::mirror_rnd_ref_angle[3]
```

Gaussian random fluctuations of reflection angles, due to small-scale surface deviations, with one or two components.

The first parameter ([0]) is the projected r.m.s. of the first component. If two components are used, [1] is the fraction following the second component and [2] is the projected r.m.s. of that second component.

Definition at line 191 of file sim_config.c.

7.10.2.30 multiplicity_offset

```
double imaging_setup::multiplicity_offset
```

[-0.5...0.5] offset of mult.

threshold w.r.t. nominal

Definition at line 445 of file sim_config.c.

7.10.2.31 nightsky_background

```
double imaging_setup::nightsky_background[MAX_PIXELS]
```

Number of photoelectrons per nanosecond.

For every single pixel separately! [GHz]

Definition at line 400 of file sim_config.c.

7.10.2.32 nsb_offaxis_factor

```
double imaging_setup::nsb_offaxis_factor[MAX_TELTRANS]
```

Fall-off of NSB with off-axis angle, including explicit and implied shadowing.

The first parameter is a function number (like second in telescope_transmission, function 0 meaning that no such factor gets applied).

Definition at line 404 of file sim_config.c.

7.10.2.33 nspe

```
int imaging_setup::nspe
```

Size of lookup table for creating random numbers according to the given single p.e.

spectrum (only used with the VERY_FAST_SPE compile-time option).

Definition at line 266 of file sim_config.c.

7.10.2.34 pixels_parallel

```
int imaging_setup::pixels_parallel
```

Defines whether pixels are aligned parallel to the optical axis (1) or are inclined normal to the focal surface (0).

Only for mirror_class==2. A value of 2 is supposed to mean that pixels within the same module are parallel and that the module is aligned to the normal of the focal surface at the module center, with pixel fronts in a plane at a height matching on average the focal surface.

Definition at line 221 of file sim_config.c.

7.10.2.35 pm_transit_time

```
double imaging_setup::pm_transit_time[4]
```

0: Total transit time of the PM [ns] at nominal voltage.

If the first dynode is operated at a fixed voltage, additional values are non-zero: 1: transit time up to first dynode [ns], 2a: fraction of total nominal voltage at 1st dyn. (value 3a is zero) or 2b: voltage between cathode and first dynode [V], 3b: total nominal voltage for nominal gain (average) [V]. applied to first dynode (if at fixed voltage).

Definition at line 283 of file sim_config.c.

7.10.2.36 pm_voltage_variation

```
double imaging_setup::pm_voltage_variation
```

Variations of transit times are usually caused by needing different voltages to reach the same gains.

Unit: fraction.

Definition at line 291 of file sim_config.c.

7.10.2.37 primary_hole

```
double imaging_setup::primary_hole
```

Diameter of a hole in the centre of the primary mirror (mirror_class==2 only).

Unit: cm.

Definition at line 210 of file sim_config.c.

7.10.2.38 qe_variation

```
double imaging_setup::qe_variation
```

Quantum efficiency variation between PMs.

Unit: fraction.

Definition at line 278 of file sim_config.c.

7.10.2.39 secondary_hole

```
double imaging_setup::secondary_hole
```

Diameter of a hole in the centre of the secondary mirror (mirror_class==2 only).

Unit: cm.

Definition at line 212 of file sim_config.c.

7.10.2.40 simple_threshold

```
double imaging_setup::simple_threshold
```

Simple threshold.

Unit: p.e.s (used only without full sim.)

Definition at line 426 of file sim_config.c.

7.10.2.41 source_altitude

```
double imaging_setup::source_altitude
```

Altitude angle of source or 0.

if source assumed at camera centre

Definition at line 108 of file sim_config.c.

7.10.2.42 telescope_ignore

```
int imaging_setup::telescope_ignore[MAX_IGNORE]
```

List of telescopes ignored (numbers starting at 1, i.e.

CT 1, ...).

Definition at line 437 of file sim_config.c.

7.10.2.43 trigger_current_limit

```
double imaging_setup::trigger_current_limit
```

Pixels above this limit are excluded from the trigger.

[uA]

Definition at line 428 of file sim_config.c.

7.10.2.44 trigger_delay_compensation

```
double imaging_setup::trigger_delay_compensation[MAX_TRG_TYPES]
```

An additional delay specific for each type of trigger.

in order to compensate for systematic differences when using multiple trigger types in parallel. Unit: ns.

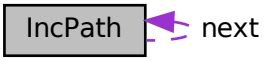
Definition at line 330 of file sim_config.c.

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

- sim_config.c

7.11 IncPath Struct Reference

Collaboration diagram for IncPath:



Data Fields

- char * **path**
- struct IncPath * **next**

7.11.1 Detailed Description

Definition at line 166 of file pfp.c.

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

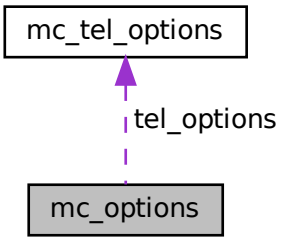
- pfp.c

7.12 mc_options Struct Reference

Options of the simulation passed through to low-level functions.

`#include <mc_aux.h>`

Collaboration diagram for mc_options:



Data Fields

- struct mc_tel_options tel_options [MAX_TEL]

  Telescope-specific options.
- int verbose

  More verbose (>0) or more quiet (<0) output than normal.
- char * input_fname

  Input file name.
- char * pe_list_fname

  Output file with list of photoelectrons in eventio format.
- char * output_fname

  Output file of simulation in whatever format.
- int new_output

Replace any existing output file rather than append, if non-zero.

- char * plot_fname

  ASCII output file with tables relevant for plotting.
- char * histogram_fname

  ASCII output file for histograms.
- char * image_fname

  Name for Postscript camera images.
- double power_law

  Wanted power law (e.g. -2.67).
- int only_triggered_arrays

  Is non-zero if only triggered arrays should.
- int only_triggered_telescopes

  Is non-zero if only data from triggered.
- int save_photons

  Save incoming photon bunches from CORSIKA (bit 0) and/or photo-electrons are stored (bit 1).
- int save_pe_amp

  Photo-electrons are intended to be stored including their amplitude.
- int save_calib_pe

  Save signal photo-electrons also in calibration events (laser/LED type).
- int movie_flag

  If time sequence should be included in images.
- int sky_is_variable

  If the nightsky should be recalculated after the pointing is changed (useful only with stars but takes a lot of time).
- int ignore_nontrig

  Ignore non-triggered showers/events in MC output.
- int ignore_mcdata

  Ignore all MC data in output.
- long iobuf_max

  Size limit for I/O buffers (input and pass-through).
- long iobuf_output_max

  Similar size limit for output I/O buffers.
- char * random_state

  State file for random generator or 'none' or 'auto'.
- int always_with_aweight

  Make MCEvent output always with area weights.
- int all_wl_random

  All photon bunches have random wavelength.
- int select_light

  0: all photon bunches, 1: only pos. bs, -1: only neg. bs, 2: only zero wl, 3: only non-zero wl
- int max_events

  Program should stop after processing so many events.
- int max_trig_events

  Program should stop after processing so many triggered events.
- double array_clock_window

  Hide the CORSIKA zero time by an array-wide random offset.

7.12.1 Detailed Description

Options of the simulation passed through to low-level functions.

Definition at line 836 of file mc_aux.h.

7.12.2 Field Documentation

7.12.2.1 all_wl_random

```
int mc_options::all_wl_random
```

All photon bunches have random wavelength.

Needed for special runs where primary particles are marked with non-zero w.l.

Definition at line 867 of file mc_aux.h.

7.12.2.2 only_triggered_arrays

```
int mc_options::only_triggered_arrays
```

Is non-zero if only triggered arrays should.

be written to the output file.

Definition at line 850 of file mc_aux.h.

7.12.2.3 only_triggered_telescopes

```
int mc_options::only_triggered_telescopes
```

Is non-zero if only data from triggered.

telescopes should be written to the output file.

Definition at line 852 of file mc_aux.h.

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

- mc_aux.h

7.13 mc_particles Struct Reference

Keep a copy of particles at ground level, in either 'bunch' or 'bunch3d' format, as-is.

```
#include <mc_aux.h>
```

Data Fields

- int **jarray**
- int **itel**
- double **photons**
- struct bunch * **bunches**
- struct bunch3d * **bunches3d**
- int **maxb**
- int **maxb3**
- int **nparticles**
- int **b_or_b3**

7.13.1 Detailed Description

Keep a copy of particles at ground level, in either 'bunch' or 'bunch3d' format, as-is.

Definition at line 1810 of file mc_aux.h.

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

- mc_aux.h

7.14 mc_run Struct Reference

Per-run information on what was actually simulated.

```
#include <mc_aux.h>
```

Data Fields

- int run
  Run number.
- int corsika_version
  *CORSIKA version number * 1000.*
- time_t run_start
  Date when CORSIKA run was started.
- double sim_start
  When sim_telarray was started (time_t or hi-res).
- int atmosphere
  Model atmosphere number (99 for custom filename).
- double height
  Height of observation level [m].
- double e_min
  Lower limit of simulated energies [TeV].
- double e_max
  Upper limit of simulated energies [TeV].
- double slope
  Spectral index of power-law spectrum.
- double radius

Radius within which cores are thrown at random. [m].

- double radius1

  Distance parameter 1 in CSCAT [m].
- double radius2

  Distance parameter 2 in CSCAT [m].
- double area

  Area over which offsets may be scattered [m^2].
- int num_arrays

  Number of arrays simulated.
- double theta_min

  Lower limit of zenith angle [degrees].
- double theta_max

  Upper limit of zenith angle [degrees].
- double phi_min

  Lower limit of azimuth angle [degrees].
- double phi_max

  Upper limit of azimuth angle [degrees].
- double viewcone_min

  Minimum of VIEWCONE range [degrees].
- double viewcone_max

  Maximum of VIEWCONE range [degrees].
- double wlen_min

  Lower limit of Cherenkov wavelength range [nm].
- double wlen_max

  Upper limit of Cherenkov wavelength range [nm].
- double bunchsize

  Cherenkov bunch size.
- long num_showers

  Number of showers to be simulated in run (CORSIKA nominal).
- long sim_showers

  Number of showers completely simulated (up to event end).
- long sim_events

  Number of events (showers∗use) completely simulated.
- long trg_events

  Number of events that triggered.
- int iact_options

  Option flags in CORSIKA (VOLUMEDET etc.)
- int low_E_model

  Low energy interaction model flags.
- int low_E_detail

  More details on low E interaction model (version etc.)
- int high_E_model

  High energy interaction model flags.
- int high_E_detail

  More details on high E interaction model (version etc.)
- double bfield_bx

  x' component of B field [mT].
- double bfield_bz

  z component of B field [mT].
- double bfield_rot

  rotation angle mag. N -> geogr. N [degrees].
- double start_depth

  Atmospheric depth where primary particle started.

7.14.1 Detailed Description

Per-run information on what was actually simulated.

Definition at line 881 of file mc_aux.h.

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

- mc_aux.h

7.15 mc_tel_options Struct Reference

Telescope-specific options passed through to low-level functions.

`#include <mc_aux.h>`

Data Fields

- double tailcut_scale

  In the 'tail-cut' scheme used here, we can apply a common scale factor to the various threshold levels applied.
- double min_photons

  Minimum no. of photons required for full simulation.
- int min_photoelectrons

  Minimum no. of photoelectrons required for electronics simulation.
- int store_photoelectrons

  Store p.e. data if at least that number are registered (also needs (save_photons&2)).
- int output_format

  See 'sim_conv2hess.c' for interpretation.
- int zero_suppression

  Zero-suppression mode.
- int data_reduction

  Data reduction mode.
- int peak_sensing

  0: ADC sums, 1: ADC sum is peak in one slice, >=2: ADC sum is peak in sliding sum in n bins.
- int bypass_optics

  Bypass or partially bypass optics ray-tracing if non-zero.
- int use_fake_trigger

  0: no fake trigger, 1: fake trigger instead of trigger derived from signals.
- int fake_trigger_pe

  Fake a trigger decision and time, for nth p.e. if n>0, for median if n<0 (and we have >=n p.e.).
- double fake_trg_delay

  Time added to derived fake trigger time since start of simulation window.

7.15.1 Detailed Description

Telescope-specific options passed through to low-level functions.

Definition at line 817 of file mc_aux.h.

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

- mc_aux.h

7.16 mirror_segment_map Struct Reference

Data Fields

- int segment_set

  Index of the corresponding mirror_segmentation structure.
- int iseg

  Which of its nseg segments is being addressed?

7.16.1 Detailed Description

Definition at line 1660 of file mc_aux.h.

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

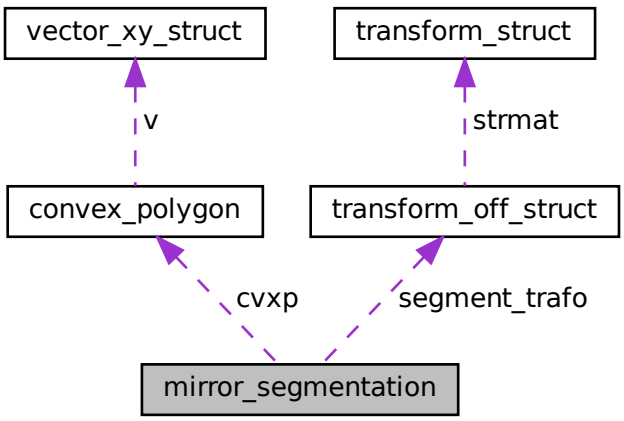
- mc_aux.h

7.17 mirror_segmentation Struct Reference

Segmentation of primary and secondary in dual mirror optics.

`#include <mc_aux.h>`

Collaboration diagram for mirror_segmentation:



Data Fields

- int type

  0: circles, 1: hexagons, 2: squares, 4: ring segments, 5: convex polygon
- int nseg

  Number of segments in this set.
- int first

  Segment ID of first segment in the set.
- double phi0

  Start of first segment (ring) or rotation angle of segment (hex/sq). [radians].
- double dphi

  The step in orientation angle from one segment of the set to the next (ring only). [radians].
- double rc

  Distance of segment center from telescope axis. [cm].
- double **xc**
- double **yc**
- double zc

  Position of the center of the first segment (yc=zc=0 for ring). [cm].
- double * **xcr**
- double * ycr

  Individual rotated centers for ring segments only (all with the same zc). [cm].
- double diameter

  Diameter of the segment [cm]. For types 0,1,2 on surface, for type 4 in projection.
- double radius

  Half of diameter. [cm].
- double **rmin**
- double rmax

  Minimum and maximum radius (type 4 only). [cm].
- double r2max

  Maximum radius squared in projection for detailed processing (1.0/1.5/2.0∗radius^2, depending on type), including some margin.
- double gap

  Gap between segments (ring segments in one set only) [cm].
- double **ci**
- double si

  Cosine and sine of inclination angle, filled at first use to reduce evaluations of polynomial derivative.
- double **cp**
- double sp

  Cosine and sine of rotation angle (phi).
- double **cx**
- double sx

  Cosine and sine of position angle.
- double **axx**
- double **axy**
- double **axz**
- double **ayx**
- double **ayy**
- double ayz

  Affine transformation coefficients, instead of four explicit rotations.
- ConvexPolygon * cvxp

  For calculating hits with polygon-shaped mirror segments.
- struct transform_off_struct * segment_trafo

  Transformation between ideally aligned and misaligned segment (separate for each of nseg pieces).

7.17.1 Detailed Description

Segmentation of primary and secondary in dual mirror optics.

Each set can cover one or multiple segments.

Definition at line 1634 of file mc_aux.h.

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

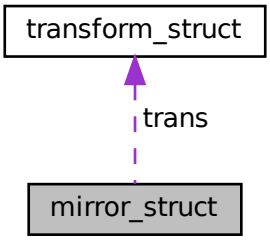
- mc_aux.h

7.18 mirror_struct Struct Reference

Parameters for a telescope mirror:

`#include <mc_aux.h>`

Collaboration diagram for mirror_struct:



Data Fields

- double x
  X position of mirror centre [cm] with respect to telescope optics frame (where (x,y)=(0,0) defines optical axis).
- double y
  Y position of mirror centre [cm].
- double z
  Z position of mirror centre [cm].
- double r
  Radius of curvature [cm].
- double f
  Focal length [cm].
- double d
  Mirror diameter [cm].
- int type

0: circ., 1,2: hex (flat-> x,y)

- double distance

  Distance from camera centre [cm].
- double inclination

  Inclination of mirror to tel. axis.
- double phi

  Azimuth angle of inclined mirror.
- struct transform_struct trans

  Telescope optics <-> mirror transformation.
- double **xm**
- double **ym**
- double **zm**
- double **xc** [6]
- double **yc** [6]
- double **diameter**
- double **dflen**
- double **cflen**
- double **mflen**
- int **shape**
- char * **id**

7.18.1 Detailed Description

Parameters for a telescope mirror:

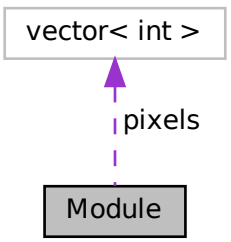
Definition at line 1592 of file mc_aux.h.

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

- mc_aux.h
- draw_mirrors.c
- plot_mirrors.c

7.19 Module Class Reference

Collaboration diagram for Module:



Public Member Functions

- **Module** (int modu, double x, double y)
- void **add_pixel** (int p)

Data Fields

- double **xp**
- double **yp**
- int **mod_id**
- vector< int > **pixels**

7.19.1 Detailed Description

Definition at line 63 of file mapmtpix.cc.

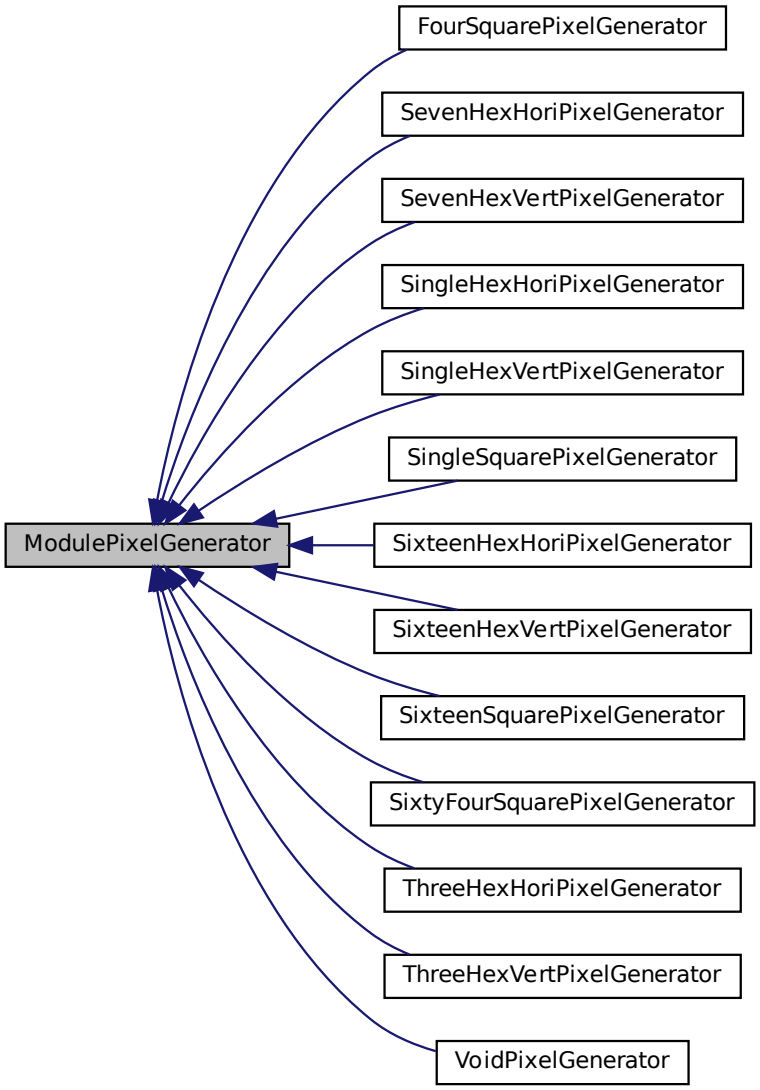
The documentation for this class was generated from the following file:

- mapmtpix.cc

7.20 ModulePixelGenerator Class Reference

Base class for all pixel generators.

Inheritance diagram for ModulePixelGenerator:



Public Member Functions

- virtual vector< PixelPos > **PixelPositions** () const =0
- virtual vector< PixelPos > **NeighbourPositions** () const =0
- virtual Offset **Offset_u** () const =0
- virtual Offset **Offset_v** () const =0
- virtual int **PixelType** () const =0
- string **TypeName** () const
- string **ShortTypeName** () const

7.20.1 Detailed Description

Base class for all pixel generators.

Definition at line 198 of file modular_camera.cc.

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

- modular_camera.cc

7.21 mpl_wordlist Struct Reference

Collaboration diagram for mpl_wordlist:

Data Fields

- char * **name**
- char * **value**
- struct mpl_wordlist * **next**

7.21.1 Detailed Description

Definition at line 634 of file sim_config.c.

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

- sim_config.c

7.22 Offset Class Reference

Offset of pixels or modules w.r.t.

Public Member Functions

- **Offset** (double dxi, double dyi)
- Offset **operator*** (double f) const
- Offset **operator+** (Offset o) const
- Offset **operator-** (Offset o) const
- double **Length** () const
- double **ProjLength** (const Offset &o) const
- ostream & **print** (ostream &os) const
- **Offset** (double dxi, double dyi)
- Offset **operator*** (double f) const
- Offset **operator+** (Offset o) const
- Offset **operator-** (Offset o) const
- double **Length** () const
- double **ProjLength** (const Offset &o) const
- ostream & **print** (ostream &os) const
- **Offset** (double dxi, double dyi)
- Offset **operator*** (double f) const
- Offset **operator+** (Offset o) const
- Offset **operator-** (Offset o) const
- double **Length** () const
- double **ProjLength** (const Offset &o) const
- ostream & **print** (ostream &os) const
- **Offset** (double dxi, double dyi)
- Offset **operator*** (double f) const
- Offset **operator+** (Offset o) const
- Offset **operator-** (Offset o) const
- double **Length** () const
- double **ProjLength** (const Offset &o) const
- ostream & **print** (ostream &os) const

Data Fields

- double **dx**
- double **dy**

7.22.1 Detailed Description

Offset of pixels or modules w.r.t.

each other, can be scaled, added, subtracted.

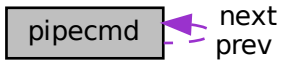
Definition at line 55 of file dsum_patchify.cc.

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

- dsum_patchify.cc
- fov.cc
- modular_camera.cc
- pixel_remap_trg.cc

7.23 pipecmd Struct Reference

Collaboration diagram for pipecmd:



Data Fields

- char * cmd
  The command executed.
- FILE * pf
  The file pointer returned by popen().
- pid_t proc_pid
  In synchronous mode, the process to signal completion.
- int got_signal
  If we got a signal, usually a broken pipe.
- int optional
  Was the specific command explicitely marked as optional?
- int unit
  Which unit might get reported with this pipe?
- struct pipecmd * **next**
- struct pipecmd * **prev**

7.23.1 Detailed Description

Definition at line 60 of file multipipe_corsika.c.

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

- multipipe_corsika.c

7.24 Pix_Type Struct Reference

List of pixel types.

`#include <mc_aux.h>`

Data Fields

- int is_defined

  Is 1 after the pixel type was properly defined.
- double r

  Radius fully enclosing pixel [cm].
- double r2

  *Radius squared [cm**2].*
- int pixel_shape

  0: circular, 1: hexagonal flat x, 2: square, 3: hexagonal flat y.
- int pm_type

  We can have different PM types in one camera.
- int cathode_shape

  Shape of free cathode area, as pixel_shape.
- double half_size

  Half flat-to-flat size [cm] if hex. or square.
- double pixel_depth

  Depth of PM below pixel entrance [cm].
- double pixel_cathode_r

  Visible cathode radius [cm].
- double pixel_cathode_r2

  *Square of visible cathode radius [cm**2].*
- double pixel_cathode_hs

  Half flat-to-flat size [cm] if hex. or square.
- double reflectivity

  Reflectivity of the lightguide which is assumed as wavelength independent.
- double transparency

  Transparency of funnel system for photons directly hitting the PM.
- int **funnel_angle_filled**
- int **funnel_wl_filled**
- double funnel_angle_table [1000]

  Effective funnel transparency as a.
- double funnel_wl_table [MAX_LAMBDA]

  Effective funnel transparency(wavelength)
- int angle_table_size

  Used to check if index is in range.
- int pixel_cells

  For SiPM, if saturation by cells already hit is to be taken into account.

7.24.1 Detailed Description

List of pixel types.

Definition at line 1480 of file mc_aux.h.

7.24.2 Field Documentation

7.24.2.1 funnel_angle_table

```
double Pix_Type::funnel_angle_table[1000]
```

Effective funnel transparency as a.

function of 0<tan(theta)<10.

Definition at line 1502 of file mc_aux.h.

Referenced by cathode_hit().

7.24.2.2 funnel_wl_table

```
double Pix_Type::funnel_wl_table[MAX_LAMBDA]
```

Effective funnel transparency(wavelength)

multiplied with angle value.

Definition at line 1504 of file mc_aux.h.

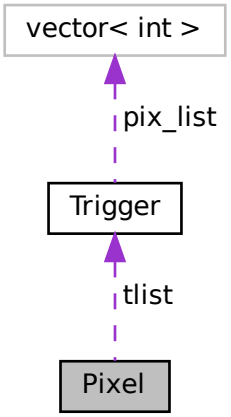
Referenced by cathode_hit().

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

- mc_aux.h

7.25 Pixel Class Reference

Collaboration diagram for Pixel:



Public Member Functions

- **Pixel** (int pid, int modu, int modc, double x, double y)
- **Pixel** (int pid, double x, double y)

Data Fields

- double **xp**
- double **yp**
- int **pix_id**
- int **mod_id**
- int **mod_cnt**
- Trigger **tlist**

7.25.1 Detailed Description

Definition at line 53 of file mapmtpix.cc.

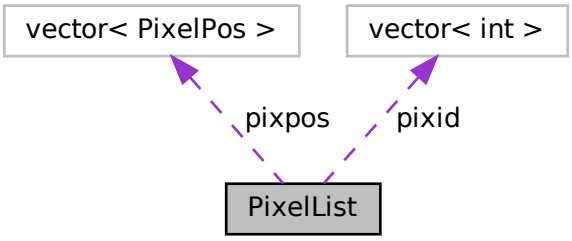
The documentation for this class was generated from the following files:

- mapmtpix.cc
- smartpix.cc

7.26 PixelList Class Reference

Lists of pixels, holding both positions and pixel IDs.

Collaboration diagram for PixelList:



Public Member Functions

- PixelList **operator+** (const PixelList &pl) const
- PixelList & **operator+=** (const PixelList &pl)
- size_t **size** () const
- size_t **active_size** () const
- vector< PixelPos >::iterator **begin** ()
- vector< PixelPos >::iterator **end** ()
- PixelList **NeighbourOf** (size_t k, double r) const
- PixelList **TopologicalNeighbourOf** (size_t k, double r, int shape) const
- PixelList & **push_back** (const PixelPos &p)
- ostream & **printx** (ostream &os) const
- ostream & **print** (ostream &os) const
- const PixelPos & **Pixel** (size_t i) const
- const PixelPos & **operator[ ]** (size_t i) const
- PixelList **operator+** (const PixelList &pl) const
- PixelList & **operator+=** (const PixelList &pl)
- size_t **size** () const
- PixelList **NeighbourOf** (size_t k, double r) const
- ostream & **printx** (ostream &os) const
- ostream & **print** (ostream &os) const

Data Fields

- vector< PixelPos > **pixpos**
- vector< int > **pixid**

7.26.1 Detailed Description

Lists of pixels, holding both positions and pixel IDs.

Can be concatenated.

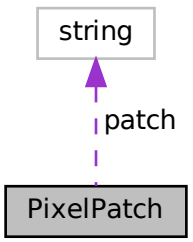
Definition at line 194 of file fov.cc.

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

- fov.cc
- modular_camera.cc

7.27 PixelPatch Class Reference

Collaboration diagram for PixelPatch:



Data Fields

- int **ipix**
- string **patch**

7.27.1 Detailed Description

Definition at line 405 of file pixel_remap_trg.cc.

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

- pixel_remap_trg.cc

7.28 PixelPos Class Reference

Positions of pixels, also including pixel shape type (although not really needed)

Public Member Functions

- **PixelPos** (const char *line)
- PixelPos **operator+** (const Offset &o) const
- PixelPos & **operator+=** (const Offset &o)
- double **Distance** () const
- double **Distance** (const PixelPos &p) const
- int **Type** () const
- PixelPos **operator*** (double f) const
- Offset **OffsetFrom** (const PixelPos &p) const
- Offset **ToOffset** () const
- bool **InRadius** (double r) const
- bool **InRadius** (double r, const PixelPos &p) const
- bool **InXflatHex** (double wx) const
- bool **InXflatHex** (double wx, const PixelPos &p) const
- bool **InYflatHex** (double wy) const
- bool **InYflatHex** (double wy, const PixelPos &p) const
- ostream & **print** (ostream &os) const
- **PixelPos** (double xp, double yp, int ip=0, int tp=1, int mp=0, int bp=0, int cp=0, int bip=0, int ap=1)
- **PixelPos** (const char *line)
- PixelPos **operator+** (const Offset &o) const
- PixelPos & **operator+=** (const Offset &o)
- double **Distance** () const
- double **Distance** (const PixelPos &p) const
- int **Type** () const
- PixelPos **operator*** (double f) const
- Offset **OffsetFrom** (const PixelPos &p) const
- Offset **ToOffset** () const
- bool **InRadius** (double r) const
- bool **InRadius** (double r, const PixelPos &p) const
- bool **InXflatHex** (double wx) const
- bool **InXflatHex** (double wx, const PixelPos &p) const
- bool **InYflatHex** (double wy) const

- bool **InYflatHex** (double wy, const PixelPos &p) const
- ostream & **print** (ostream &os) const
- **PixelPos** (double xp, double yp, int it=0)
- PixelPos **operator+** (const Offset &o) const
- double **Distance** () const
- double **Distance** (const PixelPos &p) const
- int **Type** () const
- PixelPos **operator**∗ (double f) const
- Offset **OffsetFrom** (const PixelPos &p) const
- Offset **ToOffset** () const
- bool **InRadius** (double r) const
- bool **InRadius** (double r, const PixelPos &p) const
- bool **InXflatHex** (double wx) const
- bool **InXflatHex** (double wx, const PixelPos &p) const
- bool **InYflatHex** (double wy) const
- bool **InYflatHex** (double wy, const PixelPos &p) const
- **PixelPos** (const char ∗line)
- PixelPos **operator+** (const Offset &o) const
- PixelPos & **operator+=** (const Offset &o)
- PixelPos & **Transform** (double sc, double rot)
- double **Distance** () const
- double **Distance** (const PixelPos &p) const
- int **Type** () const
- PixelPos **operator**∗ (double f) const
- Offset **OffsetFrom** (const PixelPos &p) const
- Offset **ToOffset** () const
- bool **InRadius** (double r) const
- bool **InRadius** (double r, const PixelPos &p) const
- bool **InXflatHex** (double wx) const
- bool **InXflatHex** (double wx, const PixelPos &p) const
- bool **InYflatHex** (double wy) const
- bool **InYflatHex** (double wy, const PixelPos &p) const
- ostream & **print** (ostream &os) const

Data Fields

- double **x**
- double **y**
- int **id**
- int **type**
- int **module**
- int **board**
- int **channel**
- int **bid**
- int **active**
- int **pixtype**

7.28.1 Detailed Description

Positions of pixels, also including pixel shape type (although not really needed)

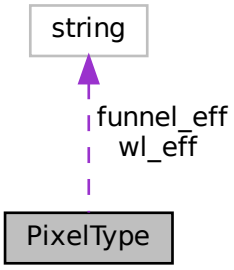
Definition at line 118 of file dsum_patchify.cc.

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

- dsum_patchify.cc
- fov.cc
- modular_camera.cc
- pixel_remap_trg.cc

7.29 PixelType Class Reference

Collaboration diagram for PixelType:



Public Member Functions

- **PixelType** (const char *line)
- **PixelType** (const char *line)
- **PixelType** (const char *line)

Data Fields

- int **pix_type**
- int **pmt_type**
- int **cathode_shape**
- int **funnel_shape**
- double **cdiameter**
- double **diameter**
- double **depth**
- string **funnel_eff**
- string **wl_eff**

7.29.1 Detailed Description

Definition at line 81 of file dsum_patchify.cc.

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

- dsum_patchify.cc
- fov.cc
- pixel_remap_trg.cc

7.30 PixPos Class Reference

Public Member Functions

- **PixPos** (const char ∗line)

Data Fields

- double **x**
- double **y**

7.30.1 Detailed Description

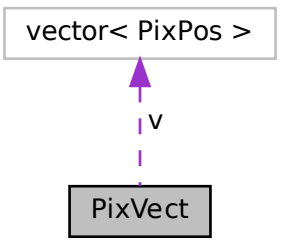
Definition at line 45 of file drawdrawers.cc.

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

- drawdrawers.cc

7.31 PixVect Class Reference

Collaboration diagram for PixVect:



Public Member Functions

- vector< PixPos >::iterator **begin** ()
- vector< PixPos >::iterator **end** ()
- size_t **size** () const
- void **push_back** (const PixPos &p)

Private Attributes

- vector< PixPos > **v**

7.31.1 Detailed Description

Definition at line 70 of file drawdrawers.cc.

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

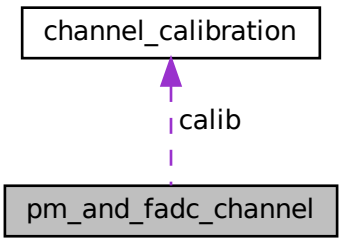
- drawdrawers.cc

7.32 pm_and_fadc_channel Struct Reference

Electronics for one PM (including accumulated signals):

`#include <mc_aux.h>`

Collaboration diagram for pm_and_fadc_channel:



Data Fields

- double qe_rel

  Quantum efficiency relative to avrg.
- double gain_rel

  Relative gain factor (common to HG+LG).
- double fadc_amplitude

  In mV at peak of single photoelectron.
- double disc_amplitude

  Same for signal at discriminator.
- double fadc_off_scale

  Scaling factor for FADC offset which is multiplied by background photoelectron rate to achieve effective AC coupling.
- double fadc_ped_shift

  Pedestal shift resulting from NSB for DC coupling.
- double disc_off_scale

  Similar for discriminator/comparator.
- double pedestal [MAX_PER_CHANNEL]

  Actual pedestal in FADC units.
- double sensitivity [MAX_PER_CHANNEL]

  ADC counts per mV of signal.
- double sensitivity_lg [MAX_PER_CHANNEL]

  Same for low-gain channel.
- double voltage_rel

  Voltage relative to expected average.
- double transit_delay

  Transit time delay of PM.
- double background

  Photoelectrons per nanosecond.
- double nightsky

  Backgr. contrib. from smooth nightsky.
- double starlight

  Backgr. contrib. from list of stars.
- double nsb_pixfact

  Off-axis dependent scaling factor for NSB + star light.
- double current

  DC current [uA] due to background.
- fadc_data_t signal [MAX_FADC_BINS]

  Digitized FADC signal.
- int disc_output_intamp

  Disc.
- int disc_input_to

  If >= 0, this channel is added to another one.
- double ideal_signal

  Without digitisation and background.
- double median_time

  Median time (not weighted) after PM.
- double disc_threshold

  Discriminator threshold [mV].
- double min_sigsum_over_thr

  Minimum integral signal over threshold after the signal exceeded the threshold.
- int cherenkov_pe

Photo-electrons from Cherenkov light.

- int min_bins_over_thr

  Time slices over threshold needed.
- disc_data_t trigger [MAX_TRIG_BINS]

  Above/below trigger at bin fraction.
- int trigger_disabled

  Non-zero if pixel cannot trigger.
- int triggered

  Non-zero if the pixel has triggered.
- int triggered_in_time

  If trigger is coincident.
- int nn_triggered

  If pixel has NN triggered at same time.
- int gate_length

  Gate width as no.
- int gate_strict

  Comparator (0) or discriminator style (1).
- struct channel_calibration calib

  Calibration parameters.
- int has_crosstalk

  For extended configuration:
- int channel_id

  Channel no. on readout chip.
- int chip_id

  Chip no. on readout card.
- int card_id

  Card no. in readout module.
- int module_id

  Module no. in camera.
- int module_id_x

  Module ID in camera.
- int pixel_id

  Unique pixel no. in camera.
- int is_off

  Is 1 if pixel configured as off.
- int disc_sum_id

  Corresponding discriminator sum.
- int num_groups

  Number of pixel groups for trigger.
- int * to_groups

  List of groups for trigger.
- int significant

  The following data is only set after the conversion to CT data format:
- int is_in_image

  Is 1 if pixel included in image analysis.
- int overflow

  Is 1 if FADC range exceeded.
- long sum_adc

  Sum of raw ADC values in readout interval.
- int sum_bins

  Number of time bins over which we summed up.

- int pixeltrg_time_int

  Sample sub-interval when pixel discriminator first fired.
- double pixel_sat_coeff

  Saturation coefficient is zero or 1 / number of cells in pixel.
- double peak_simple

  Simple pulse peak (max. bin).
- double peak_pp

  Pulse peak from pulse analysis.
- double peak_sc

  Similar but at time of pixel.
- double peak_pos

  Per pixel pulse analysis:
- double pulse_rise [3]

  Position of first rise to 20%, 50%, and 80% of peak.
- double pulse_width [3]

  Width at 20%, 50%, and 80% of peak.
- double pulse_t_over_thr

  Time over threshold.
- double pulse_sum_loc

  Pulse summed up around local pixel pulse maximum.
- double pulse_sum_glob

  Pulse summed up around global pulse maximum.

7.32.1 Detailed Description

Electronics for one PM (including accumulated signals):

Definition at line 1025 of file mc_aux.h.

7.32.2 Field Documentation

7.32.2.1 disc_output_intamp

```
int pm_and_fadc_channel::disc_output_intamp
```

Disc.

/comp. output amplitude in units of percents of nominal amplitude (i.e. 100+/-10).

Definition at line 1062 of file mc_aux.h.

7.32.2.2 gate_length

```
int pm_and_fadc_channel::gate_length
```

Gate width as no.

of 1/{DISC_BITS_PER_BIN}th of bins.

Definition at line 1079 of file mc_aux.h.

Referenced by save_or_restore_all_channels().

7.32.2.3 has_crosstalk

```
int pm_and_fadc_channel::has_crosstalk
```

For extended configuration:

Is 1 if crosstalk is relevant.

Definition at line 1085 of file mc_aux.h.

7.32.2.4 min_sigsum_over_thr

```
double pm_and_fadc_channel::min_sigsum_over_thr
```

Minimum integral signal over threshold after the signal exceeded the threshold.

[disc. amp. $*$ internal disc. time step]

Definition at line 1069 of file mc_aux.h.

7.32.2.5 peak_pos

```
double pm_and_fadc_channel::peak_pos
```

Per pixel pulse analysis:

Position of pulse peak [time slices].

Definition at line 1116 of file mc_aux.h.

Referenced by pulse_shape_analysis().

7.32.2.6 pixeltrg_time_int

```
int pm_and_fadc_channel::pixeltrg_time_int
```

Sample sub-interval when pixel discriminator first fired.

(in DISC_BITS_PER_BIN finer sampling than FADC bins).

Definition at line 1108 of file mc_aux.h.

7.32.2.7 sensitivity

```
double pm_and_fadc_channel::sensitivity[MAX_PER_CHANNEL]
```

ADC counts per mV of signal.

This is identical for high-gain and low-gain channels. H-g and l-g only differ in amplitude.

Definition at line 1047 of file mc_aux.h.

Referenced by save_or_restore_all_channels().

7.32.2.8 significant

```
int pm_and_fadc_channel::significant
```

The following data is only set after the conversion to CT data format:

Is 1 if pixel signal is significant in conversion to CT format.

Definition at line 1098 of file mc_aux.h.

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

- mc_aux.h

7.33 pm_and_fadc_temporary Struct Reference

Temporary data for a PMT and its electronics channels which.

Data Fields

- double disc_input [MAX_TRIG_BINS ∗DISC_BITS_PER_BIN]

  Only needed for analog sum into majority trigger.
- char disc_output [MAX_TRIG_BINS ∗DISC_BITS_PER_BIN]

  Only meaningful if pixel triggered.
- float asum_shaped_clipped [MAX_TRIG_BINS ∗DISC_BITS_PER_BIN]

  Only filled if there are analog (clipped) sum trigger groups.
- dsum_t dsum_shaped_clipped [MAX_FADC_BINS]

  Only filled if there are digital (clipped) sum trigger groups.

7.33.1 Detailed Description

Temporary data for a PMT and its electronics channels which.

is only kept while processing a single telescope trigger. The data elements are re-used when processing the next telescope.

Definition at line 45 of file sim_signal.c.

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

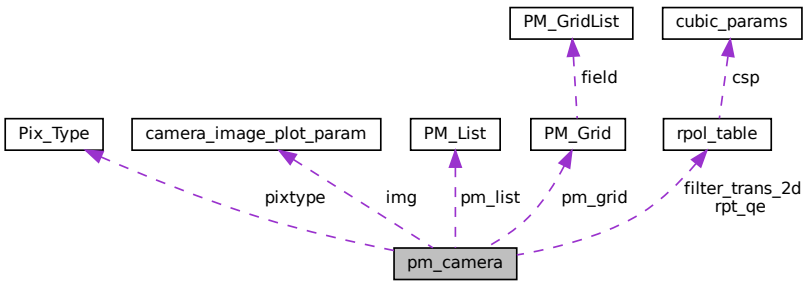
- sim_signal.c

7.34 pm_camera Struct Reference

Geometric and optical parameters of a camera.

`#include <mc_aux.h>`

Collaboration diagram for pm_camera:



Data Fields

- int telescope

  The 'official' number (ID) for the telescope.
- int itel

  Internal telescope counter (starting at zero).
- int camera_type

  1: hexagonal, 2: square, 3: custom
- int pixels

  Number of pixels in the camera.
- int min_pix_type

  Lowest pixel type used.
- int max_pix_type

  Highest pixel type used.
- struct Pix_Type pixtype [MAX_PIX_TYPES]

  Definition of pixel types.
- struct PM_List * pm_list

  List of individual PMs (if custom) or NULL.
- struct PM_Grid pm_grid

  for accelerated pixel search with custom camera.
- double camera_body_diameter

  Diameter of camera body for shadowing [cm].
- double pixel_size

  Flat-to-flat 'size' (=separation) of pixels [cm].
- double pixel_depth

  Depth of PM below pixel entrance [cm].
- double pixel_x_pos [MAX_PIXELS]

  X position of pixel [cm] as seen from the camera front, X->north in zenith with telescope at azimuth=0.
- double pixel_y_pos [MAX_PIXELS]

  Y position of pixel [cm].
- int camera_pixel_assignment_initialized

  Is 1 if initialisation is done.
- int square_camera_pixel_rows

  No. of columns/rows of square camera.
- double pixel_cathode_r_squared

  *Square of visible cathode radius [cm**2].*
- double pixel_lightguide_extend

  Fraction of pixel 'size' to which the lightguide is working; the rest is dead.
- double lightguide_reflectivity

  Reflectivity of the lightguide which is assumed as wavelength independent.
- struct camera_image_plot_param img

  Image plot parameters.
- double cam_rot

  Camera rotation angle (as in telescope optics).
- int curved_surface

  Is 1 if the pixels are not in a focal plane.
- int pixels_parallel

  Is 1 if the pixels look along the optical axis.
- int module_parallel

  Is 1 if the pixels in a module (drawer) are looking in the same direction.
- double filter_trans [MAX_LAMBDA]

1-D (or max. of 2-D) filter transmission table for quick access.

- int with_filter_2d

  Short-hand flag for having to use the 2-D table (0: no, 1: yes).
- RpolTable * filter_trans_2d

  If we have a 2-D (wl,angle) transmission table.
- RpolTable * rpt_qe

  Keep link to quantum efficiency table for reference only.

7.34.1 Detailed Description

Geometric and optical parameters of a camera.

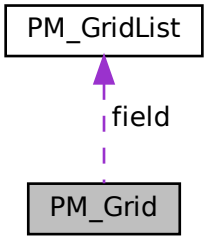
Definition at line 1535 of file mc_aux.h.

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

- mc_aux.h

7.35 PM_Grid Struct Reference

Collaboration diagram for PM_Grid:



Data Fields

- double **x_low**
- double x_high

  Lower and upper boundary in x [cm].
- double **y_low**
- double y_high

  Lower and upper boundary in y [cm].
- double **dxm1**
- double dym1

  nx/(x_high-x_low) for index calculation
- int **nx**
- int ny

  Number of grid elements in each dimension.
- struct PM_GridList * field

  Which pixels touch that grid element.

7.35.1 Detailed Description

Definition at line 1470 of file mc_aux.h.

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

- mc_aux.h

7.36 PM_GridList Struct Reference

Structure for rectangular grid used for accelerated pixel search.

`#include <mc_aux.h>`

Data Fields

- int num_pm
  Number of pixels touching grid element.
- int i_pm
  Temporary index.
- int * list
  List of pixel numbers.

7.36.1 Detailed Description

Structure for rectangular grid used for accelerated pixel search.

Definition at line 1464 of file mc_aux.h.

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

- mc_aux.h

7.37 PM_List Struct Reference

List of photomultipliers for custom camera.

`#include <mc_aux.h>`

Data Fields

- double **x**
- double **y**
- double z

  Position of pixel centre [cm].
- double **cx**
- double sx

  Pixel rotation for radial projection.
- double **nx**
- double **ny**
- double axx

  Normal vector is (nx,ny,1) as for focal surface.
- double **axy**
- double **axz**
- double **ayx**
- double **ayy**
- double **ayz**
- double **azx**
- double **azy**
- double azz

  Coefficients for affine transformation with inclined pixels.
- int pix_type

  Pixel type.
- int module

  Pixel module (drawer) number.
- double dz

  Manually set displacement along z axis w.r.t. focal surface.
- double rot

  Manually set rotation angle around normal vector.

7.37.1 Detailed Description

List of photomultipliers for custom camera.

Definition at line 1512 of file mc_aux.h.

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

- mc_aux.h

7.38 postproc_entry Struct Reference

Collaboration diagram for postproc_entry:

Data Fields

- char * **cmd**
- struct postproc_entry * **next**

7.38.1 Detailed Description

Definition at line 131 of file multipipe_corsika.c.

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

- multipipe_corsika.c

7.39 reconstructed Struct Reference

Reconstructed parameters of the showers: (beware: lengths in [m] and energies in [TeV] - as in HEGRA analysis)

`#include <mc_aux.h>`

Data Fields

- double energy
  Shower energy [TeV].
- double azimuth
  Shower direction azimuth [deg].
- double altitude
  Shower direction altitude above horizon [deg].
- double **xcore**
- double **ycore**
- double zcore
  Shower core position [m].
- double m_scwid
  Mean scaled width of images [deg].
- double m_sclen
  Mean scaled length of images [deg].
- double amplitude [MAX_TEL][3]
  Image amplitude and halo [p.e.].
- double ximg [MAX_TEL]
  X position of image centre in camera. [deg].
- double yimg [MAX_TEL]
  Y position of image centre in camera. [deg].
- double img_angle [MAX_TEL]
  Angle of image major axis with camera X [deg].
- double width [MAX_TEL]
  Image parameter 'width' [deg].
- double length [MAX_TEL]
  Image parameter 'length' [deg].
- double dis [MAX_TEL]

Image parameter 'dis' (distance) [deg].

- double miss [MAX_TEL]

  Image parameter 'miss' [deg].
- double conc [MAX_TEL]

  Image parameter 'conc' (concentration)
- double azwidth [MAX_TEL]

  Image parameter 'azwidth' [deg].
- double azlength [MAX_TEL]

  Image parameter 'azlength' [deg].
- double alpha [MAX_TEL]

  Image parameter 'alpha' [deg].
- double tel_core_distance [MAX_TEL]

  Offset of telescopes from shower core.
- double tel_core_dist_3d [MAX_TEL]

  Offset of telescopes from shower axis.
- double core_distance

  Offset of CT3 from shower axis.
- double core_dist_3d

  Offset of CT3 from shower axis in 3D.
- double theta

  Offset of reconstructed from simulated shower direction [deg].
- double rise_time [MAX_TEL]

  Rise time of time profile [ns].
- double time_fwhm [MAX_TEL]

  Full width at half maximum of profile [ns].
- int img_pixels [MAX_TEL]

  Number of pixels used for image.
- int rec_level

  *0: nothing, 1: image param., 2: shower dir, ... */*

7.39.1 Detailed Description

Reconstructed parameters of the showers: (beware: lengths in [m] and energies in [TeV] - as in HEGRA analysis)

Definition at line 957 of file mc_aux.h.

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

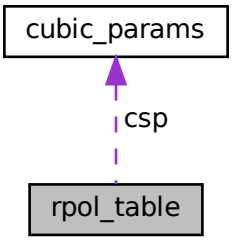
- mc_aux.h

7.40 rpol_table Struct Reference

Structure describing an interpolation table, interpolation scheme and selected options.

`#include <rpolator.h>`

Collaboration diagram for rpol_table:



Data Fields

- int ndim
  1 or 2 dimension(s), for the independent variable(s) that is.
- size_t **nx**
- size_t ny
  No.
- double * x
  Supporting points in x.
- double * y
  Supporting points in y (if ndim=2 and ny>1)
- double * z
  *Table values (size nx for 1-dim or nx*ny for 2-dim)*
- double * zxmax
  Optional max.
- double * zxmin
  Optional min.
- double aux
  This auxiliary value may be useful to the application but is not used internally.
- double **xmin**
- double xmax
  Range covered in x.
- double **dx**
- double dxi
  Step size in x and inverse of it, for equidistant only.
- double xrise
  +1 if x values are in ascending order; -1 for descending
- double **ymin**
- double ymax

Range covered in y (if ndim=2)

- double **dy**
- double dyi

  Step size in y and inverse of it, for equidistant only (2-D).
- double yrise

  +1 if y values are in ascending order; -1 for descending
- double **zmin**
- double zmax

  Range covered in result values.
- char * fname

  Name of the file from which the table was loaded (includes all options).
- char * options

  Options used in option parameter or NULL pointer.
- int equidistant

  Equidistant support points make life much easier (bit 0: x, bit 1: y).
- int scheme

  Requested interpolation scheme (0: nearest, 1: linear, 2: quadratic, 3: natural cubic spline, 4: clampled cubic spline).
- int clipping

  0: Extrapolate with edge value, 1: zero outside range.
- int remapped

  If user code remaps x, y, and/or z values w.r.t.
- int zxreq

  Flag activated when options indicate that a set of zxmax values should be provided.
- int **logs**
- int **xlog**
- int **ylog**
- int zlog

  Log applied to any x/y axis, to x axis, y axis, z axis?
- CsplinePar * csp

  Cubic spline parameters (scheme 3 and 4 only), need one-time initialisation.
- int use_count

  Indicates how often a table is in use, but not safe enough to make it a smart pointer.

7.40.1 Detailed Description

Structure describing an interpolation table, interpolation scheme and selected options.

Definition at line 51 of file rpolator.h.

7.40.2 Field Documentation

7.40.2.1 ny

`size_t rpol_table::ny`

No.

of entries in x and (optional) y points.

Definition at line 54 of file rpolator.h.

Referenced by simple_rpol1d_table().

7.40.2.2 remapped

```
int rpol_table::remapped
```

If user code remaps x, y, and/or z values w.r.t.

the table input, it should mark this here to avoid repeating it. (Make sure to call rpol_check_equi_range() after you remap!)

Definition at line 74 of file rpolator.h.

7.40.2.3 zxmax

```
double* rpol_table::zxmax
```

Optional max.

of z values along each y line (at each x value)

Definition at line 58 of file rpolator.h.

Referenced by rpol_free().

7.40.2.4 zxmin

```
double* rpol_table::zxmin
```

Optional min.

of z values along each y line (at each x value)

Definition at line 59 of file rpolator.h.

Referenced by rpol_free().

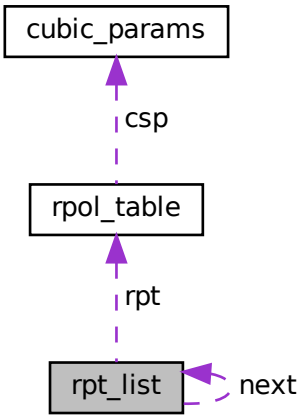
The documentation for this struct was generated from the following file:

- rpolator.h

7.41 rpt_list Struct Reference

Registered interpolation tables allow re-use of tables without having to load them again.

Collaboration diagram for rpt_list:



Data Fields

- struct rpol_table * **rpt**
- struct rpt_list * **next**

7.41.1 Detailed Description

Registered interpolation tables allow re-use of tables without having to load them again.

Available for simple (2-column) 1-D interpolation and for rectangular-grid 2-D interpolation with y grid values in header line, x grid values in first column.

Definition at line 546 of file rpolator.c.

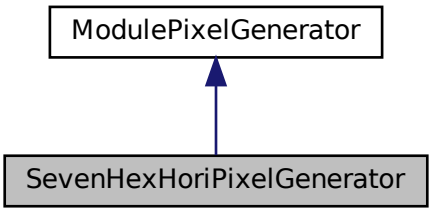
The documentation for this struct was generated from the following file:

- rpolator.c

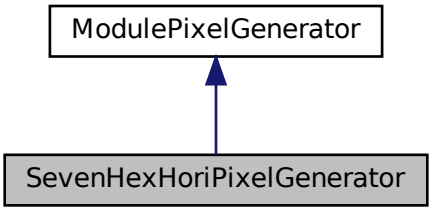
7.42 SevenHexHoriPixelGenerator Class Reference

Seven hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Inheritance diagram for SevenHexHoriPixelGenerator:



Collaboration diagram for SevenHexHoriPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.42.1 Detailed Description

Seven hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Definition at line 306 of file modular_camera.cc.

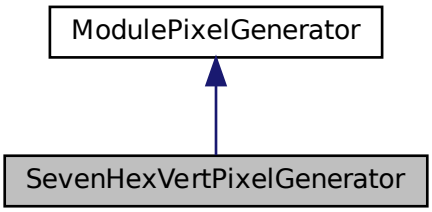
The documentation for this class was generated from the following file:

- modular_camera.cc

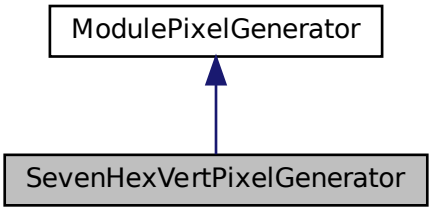
7.43 SevenHexVertPixelGenerator Class Reference

Seven hexagonal pixels with flat-to-flat in y direction, corner-to-corner in x direction.

Inheritance diagram for SevenHexVertPixelGenerator:



Collaboration diagram for SevenHexVertPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.43.1 Detailed Description

Seven hexagonal pixels with flat-to-flat in y direction, corner-to-corner in x direction.

Definition at line 405 of file modular_camera.cc.

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

- modular_camera.cc

7.44 simtel_analysis_data Struct Reference

Data used in image parameters analysis.

Data Fields

- int **pixels**
- double **pixel_amp** [MAX_PIXELS]
- double **xpos** [MAX_PIXELS]
- double **ypos** [MAX_PIXELS]
- double **pixel_size** [MAX_PIXELS]
- double **pixel_area** [MAX_PIXELS]
- int **num_neighbours** [MAX_PIXELS]
- int **list_neighbours** [MAX_PIXELS][MAX_NEIGHBOURS]
- int **num_neighbours2** [MAX_PIXELS]
- int **list_neighbours2** [MAX_PIXELS][MAX_NEIGHBOURS2]
- int **num_in_image**
- int **is_in_image** [MAX_PIXELS]
- int **list_in_image** [MAX_PIXELS]
- double **amplitude**
- double **xmean**
- double **ymean**
- double **direction**
- double **orientation**
- double **width**
- double **length**
- double **distance**
- double **conc**
- double **miss**
- double **alpha**
- double **skewness**
- double **kurtosis**
- double **tm_slope**
- double **tm_width1**
- double **tm_width2**
- double **tm_rise**
- double **tm_residual**

7.44.1 Detailed Description

Data used in image parameters analysis.

Local use only. Without this 'online' analysis, the output will not contain any reconstructed image or shower data - but we can always reconstruct it later from the raw data - saving some memory here.

Definition at line 127 of file sim_conv2hess.c.

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

- sim_conv2hess.c

7.45 simulated_shower_parameters Struct Reference

Basic parameters of the simulated showers: (beware: lengths in [m] and energies in [TeV] - as in HEGRA analysis)

`#include <mc_aux.h>`

Data Fields

- int shower
  Shower number.
- int array
  Array number = shower usage number.
- double energy
  Shower energy [TeV].
- double azimuth
  Shower direction azimuth [deg].
- double altitude
  Shower direction altitude above horizon.
- double **xcore**
- double **ycore**
- double zcore
  Shower core position [m].
- double aweight
  Area weight to be used with non-uniform core-position sampling [m^ 2].
- double x0
  Atmospheric depth where particle was started [g/cm^ 2].
- double h1int
  Height a.s.l. of first interaction [m].
- double core_dist_3d
  Distance of reference point (array center) from shower axis.
- double tel_core_dist_3d [MAX_TEL]
  Offset of telescopes from shower axis.
- int particle
  Primary particle type [CORSIKA code].
- int have_longi
  Indicates if vertical profiles were found in data.
- double step_longi
  *Step size of vertical profiles [g/cm**2].*
- double xmax
  *Depth of shower maximum from all particles [g/cm**2].*
- double emax
  Depth of shower maximum from positrons and electrons.
- double cmax
  *Depth of maximum of Cherenkov light emission [g/cm**2].*
- double hmax
  Height of shower maximum (from xmax above) [m] a.s.l.
- double xlongi [MAX_LONGI]
  Vertical profile of all particles.
- double elongi [MAX_LONGI]
  Vertical profile of all electrons+positrons.
- double clongi [MAX_LONGI]
  Vertical profile of Cherenkov light emission.

7.45.1 Detailed Description

Basic parameters of the simulated showers: (beware: lengths in [m] and energies in [TeV] - as in HEGRA analysis)

Definition at line 926 of file mc_aux.h.

7.45.2 Field Documentation

7.45.2.1 have_longi

```
int simulated_shower_parameters::have_longi
```

Indicates if vertical profiles were found in data.

If not, the followig numbers should be all zeroes.

Definition at line 941 of file mc_aux.h.

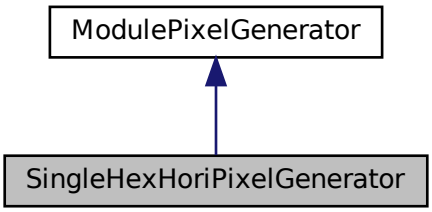
The documentation for this struct was generated from the following file:

- mc_aux.h

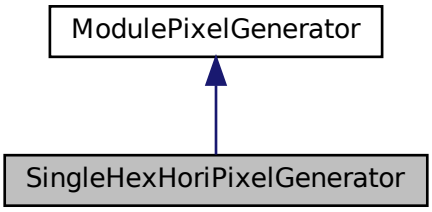
7.46 SingleHexHoriPixelGenerator Class Reference

Hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Inheritance diagram for SingleHexHoriPixelGenerator:



Collaboration diagram for SingleHexHoriPixelGenerator:



Public Member Functions

- vector< PixelPos > **PixelPositions** () const
- vector< PixelPos > **NeighbourPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.46.1 Detailed Description

Hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Definition at line 257 of file modular_camera.cc.

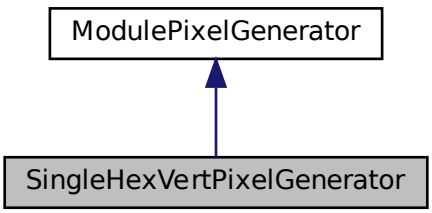
The documentation for this class was generated from the following file:

- modular_camera.cc

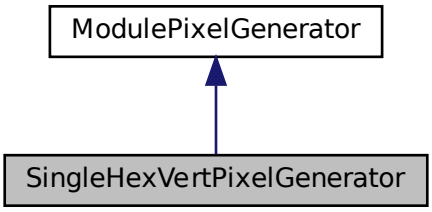
7.47 SingleHexVertPixelGenerator Class Reference

Hexagonal pixels with flat-to-flat in y direction, corner-to-corner in x direction.

Inheritance diagram for SingleHexVertPixelGenerator:



Collaboration diagram for SingleHexVertPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.47.1 Detailed Description

Hexagonal pixels with flat-to-flat in y direction, corner-to-corner in x direction.

Definition at line 356 of file modular_camera.cc.

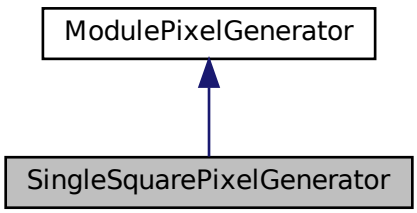
The documentation for this class was generated from the following file:

- modular_camera.cc

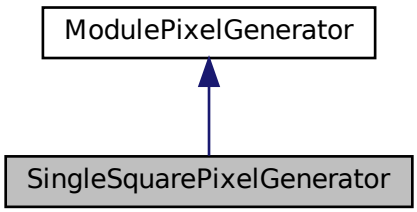
7.48 SingleSquarePixelGenerator Class Reference

Single square pixels.

Inheritance diagram for SingleSquarePixelGenerator:



Collaboration diagram for SingleSquarePixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.48.1 Detailed Description

Single square pixels.

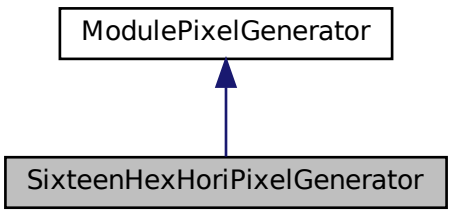
Definition at line 455 of file modular_camera.cc.

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

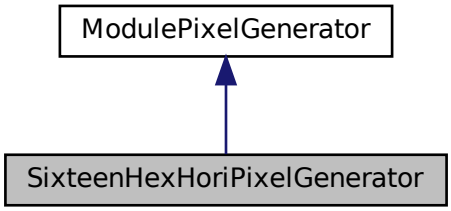
- modular_camera.cc

7.49 SixteenHexHoriPixelGenerator Class Reference

Inheritance diagram for SixteenHexHoriPixelGenerator:



Collaboration diagram for SixteenHexHoriPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.49.1 Detailed Description

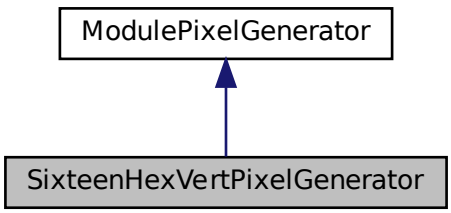
Definition at line 333 of file modular_camera.cc.

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

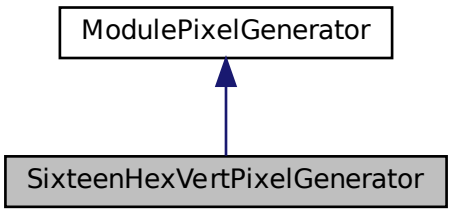
- modular_camera.cc

7.50 SixteenHexVertPixelGenerator Class Reference

Inheritance diagram for SixteenHexVertPixelGenerator:



Collaboration diagram for SixteenHexVertPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.50.1 Detailed Description

Definition at line 432 of file modular_camera.cc.

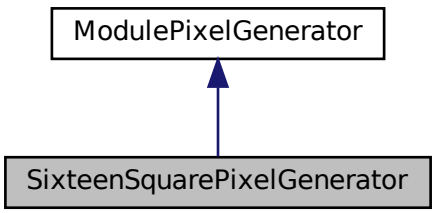
The documentation for this class was generated from the following file:

- modular_camera.cc

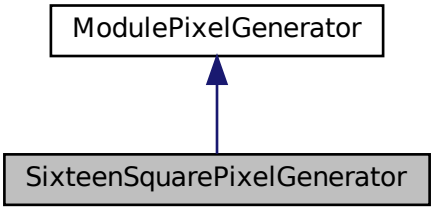
7.51 SixteenSquarePixelGenerator Class Reference

Sixteen square pixels.

Inheritance diagram for SixteenSquarePixelGenerator:



Collaboration diagram for SixteenSquarePixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.51.1 Detailed Description

Sixteen square pixels.

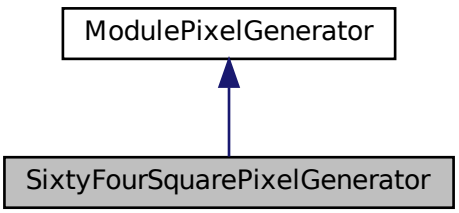
Definition at line 507 of file modular_camera.cc.

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

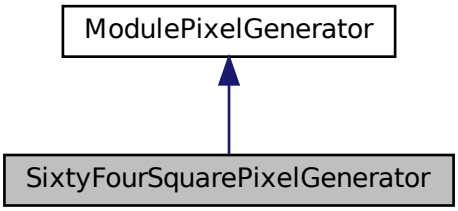
- modular_camera.cc

7.52 SixtyFourSquarePixelGenerator Class Reference

Inheritance diagram for SixtyFourSquarePixelGenerator:



Collaboration diagram for SixtyFourSquarePixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.52.1 Detailed Description

Definition at line 543 of file modular_camera.cc.

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

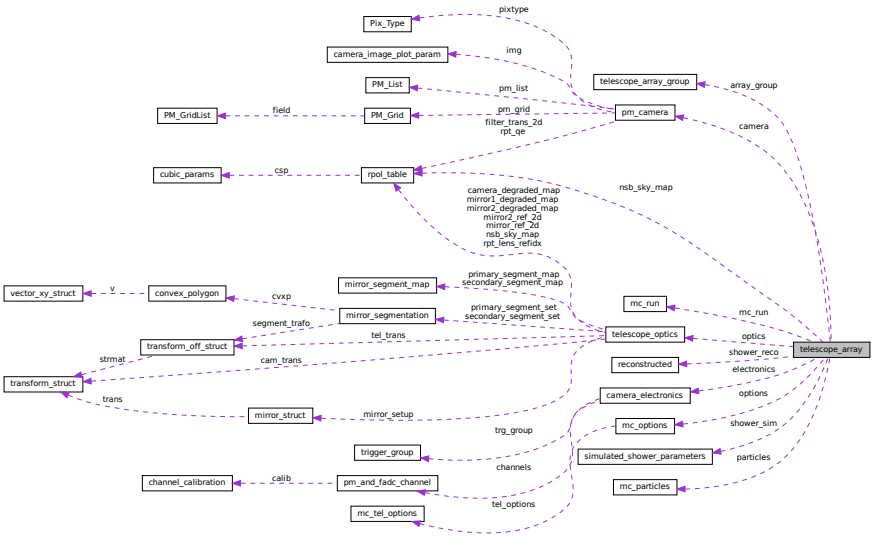
- modular_camera.cc

7.53 telescope_array Struct Reference

Parameters relevant for the whole array of telescopes: Actually some refer to ONE array (usually the one presently simulated), some to ANY array (the layout), and some to ALL arrays (random shifts).

`#include <mc_aux.h>`

Collaboration diagram for telescope_array:



Data Fields

- double longitude

  Geographic longitude (positive towards east) [deg].
- double latitude

  Geographic latitude (positive towards north) [deg].
- double obs_height

  Height of observation level [cm] from shower simulation.
- double atm_height

  Height assumed for atmospheric transmission [cm].
- double refpos [3]

  Reference position with respect to obs.
- int ntel

  Number of telescopes simulated per array.
- int max_tel

  Maximum number of telescopes acceptable.
- int narray

  Number of arrays with random shifts per shower.
- double xtel [MAX_TEL]

  X positions of telescopes ([cm] -> north)
- double ytel [MAX_TEL]

  Y positions of telescopes ([cm] -> west)
- double ztel [MAX_TEL]

  Z positions of telescopes ([cm] -> up)
- double rtel [MAX_TEL]

  Radius of spheres enclosing telescopes [cm].
- double stel [MAX_TEL]

  Separation from nearest other telescope [cm].
- double toff

Time offset from first interaction to the moment when the extrapolated primary flying with the vacuum speed of light would be at the observation level.

- double xoff [MAX_ARRAY]

  X offsets of the randomly shifted arrays [cm].
- double yoff [MAX_ARRAY]

  Y offsets of the randomly shifted arrays [cm].
- double aweight [MAX_ARRAY]

  Area weight for non-uniformly distributed core offsets (may be 0 when older data with uniform offsets are read).
- double azimuth

  Nominal azimuth angle of telescope system [deg].
- double altitude

  Nominal altitude angle of telescope system [deg].
- double source_azimuth

  Azimuth of assumed source. [deg].
- double source_altitude

  Altitude of assumed source. [deg].
- double convergent_pos [3]

  Reference position [m] for convergent/divergent observations.
- int conv_div_opt

  Non-zero if any telescope is configured for convergent or divergent pointing.
- double mean_convergent_depth

  Mean atmospheric depth of telescope-specific convergence. Zero for all parallel.
- double trigger_window

  The time window within which central trigger signals must be seen, after correction for nominal delay [ns].
- int min_tel_trigger

  No. of triggered telescoped needed for array trigger.
- int tel_triggered

  No. of telescopes triggered in current array.
- int array_triggered

  Is 1 if current array was triggered.
- int with_aweight

  Is 1 if input data came with weights.
- int telescope_ignore [MAX_IGNORE]

  List of telescopes to be ignored in simulation.
- int closed_pedestal_events

  Number of pedestal events with camera closed.
- int opened_pedestal_events

  Number of pedestal events with camera opened.
- int led_events

  Number of events with lid LEDs firing.
- int laser_events [MAX_LASER_LEVELS]

  Number of laser events.
- struct simulated_shower_parameters shower_sim

  Shower parameters simulated.
- struct reconstructed shower_reco

  and reconstructed.
- struct telescope_optics * optics

  Optics and mounts.
- struct pm_camera * camera

  Camera parameters.
- struct camera_electronics * electronics

Electronics.

- struct mc_options * options

  File names etc.
- struct mc_run **mc_run**
- double wfront_comp_az

  Azimuth angle for which plane wavefront compensation was evaluated. [deg].
- double wfront_comp_alt

  Altitude angle for which plane wavefront compensation was evaluated. [deg].
- double wfront_comp_ls

  Assumed light speed (in air) for plane wavefront compensation. [cm/ns].
- int **num_array_groups**
- struct telescope_array_group **array_group** [MAX_ARRAY_GROUPS]
- RpolTable * nsb_sky_map

  NSB background scaling factor (az,alt) or NULL.
- struct mc_particles particles

  Particles at ground level; only allocated on demand.
- int **run**
- int **event**

7.53.1 Detailed Description

Parameters relevant for the whole array of telescopes: Actually some refer to ONE array (usually the one presently simulated), some to ANY array (the layout), and some to ALL arrays (random shifts).

Definition at line 1827 of file mc_aux.h.

7.53.2 Field Documentation

7.53.2.1 aweight

```
double telescope_array::aweight[MAX_ARRAY]
```

Area weight for non-uniformly distributed core offsets (may be 0 when older data with uniform offsets are read).

[cm^2]

Definition at line 1848 of file mc_aux.h.

7.53.2.2 refpos

```
double telescope_array::refpos[3]
```

Reference position with respect to obs.

level [cm] or divergent pointing w.r.t. nominal array viewing direction.

Definition at line 1833 of file mc_aux.h.

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

- mc_aux.h

7.54 telescope_array_group Struct Reference

A collection of telescopes which can form an array trigger condition.

`#include <mc_aux.h>`

Data Fields

- int num_telescopes
  Number of telescopes in one group.
- int * telescope
  List of telescope IDs in group.
- int num_required
  Number of specific telescope that must have fired.
- int * req_telescope
  Required telescopes list.
- int trg_type_required
  -1: any, 0: majority, 1: an. sum, 2: dig. sum , 3: dig. majority (not yet supported)
- int hard_stereo
  0: readout also if other group triggered, 1: only readout if this group triggered.
- double trigger_window
  Group-specific trigger window (optional) [ns].
- double min_sep
  Minimum separation between two telescopes both contributing to a trigger [cm].
- int min_telescopes
  Minimum no. of telescopes required or 0.
- int triggered
  Number of telescopes with telescope trigger.
- int has_triggered
  Is 1 if group has a coincidence trigger.

7.54.1 Detailed Description

A collection of telescopes which can form an array trigger condition.

Definition at line 1791 of file mc_aux.h.

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

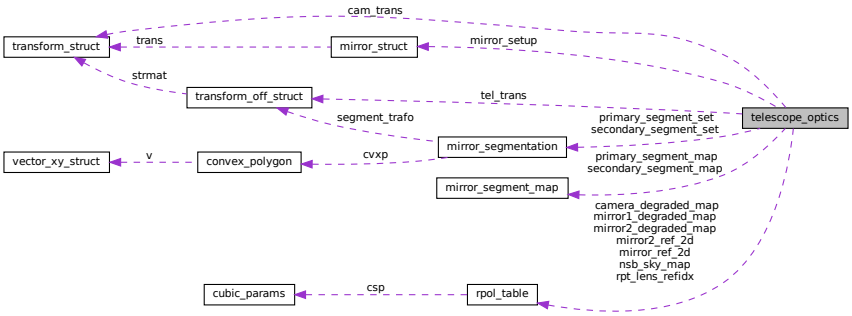
- mc_aux.h

7.55 telescope_optics Struct Reference

Parameters for the optics of a telescope (except the camera):

`#include <mc_aux.h>`

Collaboration diagram for telescope_optics:



Data Fields

- int telescope

  The 'official' number (ID) for the telescope.
- int itel

  Internal telescope counter (starting at zero).
- struct transform_off_struct tel_trans

  Ground <-> telescope optics transformation. May or may not include offsets between axes.
- struct transform_struct cam_trans

  Telescope optics <-> camera transformation.
- Mirror mirror_setup [MAX_MIRRORS]

  Setup/alignment of individual mirrors.
- int mirrors

  No. of mirrors present.
- int num_teltrans

  No. of coefficients used in expanding telescope transmission.
- int method_teltrans

  Method to be used for expanding telescope transmission (0: constant)
- int camera_body_shape

  Shape: 0: circ. 1,3: hex, 2: square; for shadowing only.
- double camera_body_diameter

  Outer diameter of camera body, used for shadowing. [cm].
- double camera_body_radius

  Half of outer diameter [cm].
- double camera_body_radius_squared

  Square of half of outer diameter [cm].
- double camera_depth

  Depth of camera body, used for shadowing. [cm].
- double camera_transmission

  For bookkeeping only. Multiplied into quantum efficiency.

- double telescope_transmission [MAX_TELTRANS]

  Account for shadowing by masts etc. (constant or function in tan(off-axis angle)).
- double max_teltrans

  Maximum transmission value over the field-of-view.
- double mirror_area

  Total area of individual mirrors corrected for inclination [cm^ 2].
- double mirror_diameter

  Diameter of optics / primary mirror. May have some margin for segment corners.
- double focal_length

  Nominal focal length of mirrors and the whole reflector. [cm].
- double focus_offset

  Since we may be focusing on showers at finite distances the entrance of the lightguides can be behind the nominal focus point.
- double overall_offset

  Offset of the centre of the point where the optical axis intersects the sphere of mirror centres with respect to the fixed pos.
- double az_alt_offset

  Offset between azimuth axis and altitude axis. [cm].
- double alt_optics_offset

  Offset between altitude axis and optical axis. [cm].
- double effective_focal_length

  Effective focal length of mirrors for focal plane (projection) image scale. [cm].
- double effective_focal_length_x

  Scale for x projection may differ if mirror not rotationally symmetric. [cm].
- double effective_focal_length_y

  Scale for y projection may differ if mirror not rotationally symmetric. [cm].
- double effective_focal_length_dx

  Asymmetric mirror may result in effective image displacement in x. [cm].
- double effective_focal_length_dy

  Asymmetric mirror may result in effective image displacement in y. [cm].
- double assumed_focal_length

  If the real effective one is not known, we may have a guess, for internal use only. [cm].
- double dslen

  Dish shape length [cm]. Resulting central curvature radius depends on parabolic or not (2x for par.)
- double mflen

  Mirror focal length (0: individually adapted) [cm].
- double mirror_flen_grading

  Center-to-edge grading of mirror focal lengths [cm].
- double mirror_flen_random [2]

  Gaussian r.m.s. and top-hat component of random mirror focal lengths. [cm][cm].
- double mirror_rnd_ref_angle

  Microscopic random reflection angle (first or only component). Now: [radians] (was [deg])
- double mirror_rnd_ref_angle2

  Microscopic random reflection angle (optional second component). Now: [degrees] (was [deg])
- double mirror_rnd_ref_frac2

  Fraction of reflection cases following the second component.
- double mirror_rnd_align_h

  Effective horizontal mirror alignment accuracy at given zenith angle. [radians].
- double mirror_rnd_align_v

  Effective vertical mirror alignment accuracy at given zenith angle. [radians].
- double mirror_rnd_distance

Random component of distance of mirror from focus. [cm].

- double mirror_dc_opt [3]

  Parameters which may (or may not) improve overall PSF with DC optics. (Deprecated)
- double azimuth

  Azimuth angle of tel. orientation. [degrees].
- double azimuth_nom

  Nominal value of azimuth. [degrees].
- double azimuth_basic

  Azimuth angle without random setting of direction. [degrees].
- double altitude

  Altitude angle of tel. orientation. [degrees].
- double altitude_nom

  Nominal value of altitude. [degrees].
- double altitude_basic

  Altitude angle without random setting of direction. [degrees].
- double random_angle

  Known random angle in each axis. [degrees].
- double random_error

  Unknown random angle in each axis. [degrees].
- int reverse_mode

  Rev. or normal mode of alt-az mount.
- int ∗ mirror_zone_link [PHI_ZONES][RAD_ZONES]

  Assign each mirror to one or.
- int mirrors_in_zone [PHI_ZONES][RAD_ZONES]

  several zones in azimuth and
- double rad_zone_width

  radius to speed up raytracing.
- int parabolic_dish

  0: Davies-Cotton, 1: parabolic base shape (for mirror class=0 only)
- int mirror_class

  0: standard segmented primary of DC or parabolic shape, 1: single solid parabolic primary, 2: general primary + secondary optics (optionally segmented) + curved focal surface.
- int npar_primary

  No. of parameters describing a general primary. Only for mirror_class==2.
- int npar_secondary

  No. of parameters describing a general secondary. Only for mirror_class==2.
- int npar_focal

  No. of parameters describing a general focal surface. Only for mirror_class==2.
- double primary_parameters [MAX_NPAR_MIRR]

  Parameters describing a general primary. Only for mirror_class==2.
- double secondary_parameters [MAX_NPAR_MIRR]

  Parameters describing a general secondary. Only for mirror_class==2.
- double focal_surface_parameters [MAX_NPAR_MIRR]

  Parameters describing the shape of the focal surface. Only for mirror_class==2.
- double primary_hole

  Diameter of a hole in the centre of the primary mirror (mirror_class==2 only).
- double secondary_hole

  Diameter of a hole in the centre of the secondary mirror (mirror_class==2 only).
- double primary_offset

  Any additional offset of the primary mirror from the common optics reference point (on top of mirror offset, i.e.
- double secondary_offset

Offset of the secondary mirror from the common optics reference point. Only for mirror_class==2.

- double camera_offset

  Offset of the camera front from the common optics reference point. Only for mirror_class==2.
- double secondary_diameter

  Diameter of secondary mirror. May include mount in addition to pure mirror, for shadowing.
- double secondary_shadow_diameter

  Diameter of secondary mirror. May include mount in addition to pure mirror, for shadowing.
- double secondary_shadow_radius_squared

  Square of shadowing radius of secondary. Only for mirror_class==2.
- double secondary_shadow_offset

  z position of secondary shadowing element (0: to be initialized from polynomial). [cm]
- double secondary_baffle [6]

  z1, z2, r1, dr, r2, c parameters of a baffle around the secondary mirror. Only for mirror_class==2.
- double camera_rotation

  Camera rotation angle.
- double cos_cam_rot

  Cosine of rotation angle.
- double sin_cam_rot

  Sine of rotation angle.
- int num_primary_segment_sets

  Number of segmentation sets of the primary in dual mirror optics. Only for mirror_class==2.
- int num_primary_segments

  Total number of primary segments, as ring segment sets can account for more than one segment.
- int num_secondary_segment_sets

  Number of segmentation sets of the secondary in dual mirror optics. Only for mirror_class==2.
- int num_secondary_segments

  Total number of secondary segments, as ring segment sets can account for more than one segment.
- int have_secondary_baffle

  Flag for presence of a baffle around the secondary mirror (0: no baffle, 1: with cylindrical baffle).
- struct mirror_segmentation primary_segment_set [MAX_SEGMENTS]

  Only for mirror_class==2.
- struct mirror_segmentation secondary_segment_set [MAX_SEGMENTS]

  Only for mirror_class==2.
- struct mirror_segment_map * **primary_segment_map**
- struct mirror_segment_map * **secondary_segment_map**
- int with_prim_segm_misaligned

  Is 1 if misalignment for segments of the primary has been set up.
- int with_sec_segm_misaligned

  Is 1 if misalignment for segments of the secondary has been set up.
- int with_mirror_ref_2d

  Short-hand flag for having to use 2-D with (primary) mirror reflection (0: no, 1: yes).
- int with_mirror2_ref_2d

  Short-hand flag for having to use 2-D with secondary mirror reflection (0: no, 1: yes).
- RpolTable * mirror_ref_2d

  If we have a 2-D (wl,angle) primary mirror reflectivity table.
- RpolTable * mirror2_ref_2d

  If we have a 2-D (wl,angle) secondary mirror reflectivity table.
- double mirror_degraded_reflection

  Need this separately for NSB p.e. rate re-calculation. Intended for primary.
- double mirror2_degraded_reflection

  Need this separately for NSB p.e. rate re-calculation. Intended for seconday.

- double camera_degraded_efficiency

  Need this separately for NSB p.e. rate re-calculation. Intended for camera.
- RpolTable * mirror1_degraded_map

  Map of additional degradation of reflection on surface of primary mirror.
- RpolTable * mirror2_degraded_map

  Map of additional degradation of reflection on surface of secondary mirror.
- RpolTable * camera_degraded_map

  Map of additional degradation of camera (assumed in focal surface).
- double lens_refidx_nominal

  Nominal index of refraction of the material of a Fresnel lens, only for mirror_class==3.
- RpolTable * rpt_lens_refidx

  Wavelength dependency of lens index of refraction, only for mirror_class==3.
- RpolTable * nsb_sky_map

  NSB background scaling factor (az,alt) or NULL. May be copy of array-wide map.
- double optical_depth_tel [MAX_LAMBDA]

  Atmospheric transmission optical depth for vertical path.
- double optical_depth_focus [MAX_LAMBDA]

  Atmospheric transmission optical depth for a typical path.
- double pathlen_tel

  Optical pathlength assumed for in-telescope atmospheric transmission.
- double convergent_distance

  Distance from reference position [m] for convergence.
- double convergent_height

  Height above observation plane [m] for convergence.
- double convergent_depth

  It may as well be specified in terms of atmospheric depth;.

7.55.1 Detailed Description

Parameters for the optics of a telescope (except the camera):

Definition at line 1667 of file mc_aux.h.

7.55.2 Field Documentation

7.55.2.1 focus_offset

```
double telescope_optics::focus_offset
```

Since we may be focusing on showers at finite distances the entrance of the lightguides can be behind the nominal focus point.

[cm]

Definition at line 1689 of file mc_aux.h.

Referenced by tel_setup_secondary().

7.55.2.2 mirror_area

```
double telescope_optics::mirror_area
```

Total area of individual mirrors corrected for inclination [cm^2].

Not accounting for any shadowing.

Definition at line 1685 of file mc_aux.h.

7.55.2.3 optical_depth_focus

```
double telescope_optics::optical_depth_focus[MAX_LAMBDA]
```

Atmospheric transmission optical depth for a typical path.

of photons within the telescope (order of focal length for 1M type telescope).

Definition at line 1781 of file mc_aux.h.

7.55.2.4 optical_depth_tel

```
double telescope_optics::optical_depth_tel[MAX_LAMBDA]
```

Atmospheric transmission optical depth for vertical path.

from center of telescope sphere to nominal level of atmtrans table.

Definition at line 1779 of file mc_aux.h.

7.55.2.5 overall_offset

```
double telescope_optics::overall_offset
```

Offset of the centre of the point where the optical axis intersects the sphere of mirror centres with respect to the fixed pos.

of the mount. [cm] Has no impact on mirror-camera distance.

Definition at line 1693 of file mc_aux.h.

Referenced by tel_setup_secondary().

7.55.2.6 primary_hole

```
double telescope_optics::primary_hole
```

Diameter of a hole in the centre of the primary mirror (mirror_class==2 only).

Unit: cm.

Definition at line 1739 of file mc_aux.h.

7.55.2.7 primary_offset

```
double telescope_optics::primary_offset
```

Any additional offset of the primary mirror from the common optics reference point (on top of mirror offset, i.e. not really needed). Only for mirror_class==2.

Definition at line 1743 of file mc_aux.h.

Referenced by tel_setup_secondary().

7.55.2.8 secondary_hole

```
double telescope_optics::secondary_hole
```

Diameter of a hole in the centre of the secondary mirror (mirror_class==2 only).

Unit: cm.

Definition at line 1741 of file mc_aux.h.

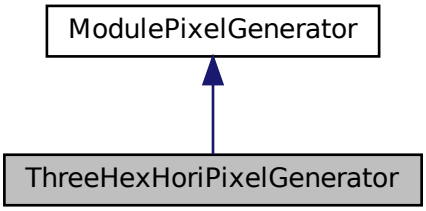
The documentation for this struct was generated from the following file:

- mc_aux.h

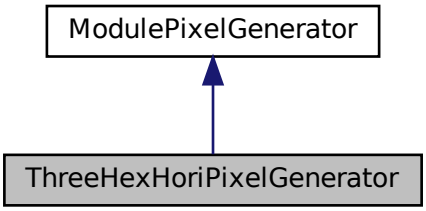
7.56 ThreeHexHoriPixelGenerator Class Reference

Three hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Inheritance diagram for ThreeHexHoriPixelGenerator:



Collaboration diagram for ThreeHexHoriPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.56.1 Detailed Description

Three hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Definition at line 283 of file modular_camera.cc.

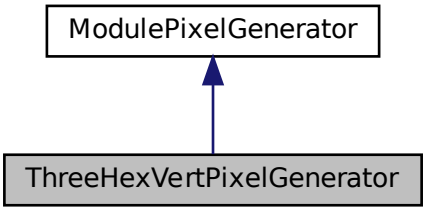
The documentation for this class was generated from the following file:

- modular_camera.cc

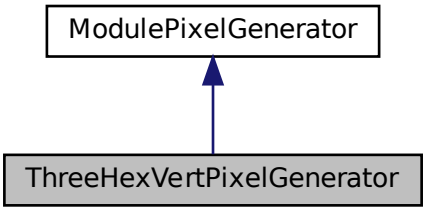
7.57 ThreeHexVertPixelGenerator Class Reference

Three hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Inheritance diagram for ThreeHexVertPixelGenerator:



Collaboration diagram for ThreeHexVertPixelGenerator:



Public Member Functions

- vector< PixelPos > **NeighbourPositions** () const
- vector< PixelPos > **PixelPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.57.1 Detailed Description

Three hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.

Definition at line 382 of file modular_camera.cc.

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

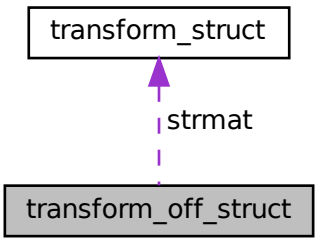
- modular_camera.cc

7.58 transform_off_struct Struct Reference

Transformation with non-intersecting rotation axes:

`#include <mc_aux.h>`

Collaboration diagram for transform_off_struct:



Data Fields

- double offset0 [3]
  Offset before start of rotation.
- double rot1 [3][3]
  Rotation matrix for first rotation.
- double offset1 [3]
  Offset between rotations.
- double rot2 [3][3]
  Rotation matrix for second rotation.
- double offset2 [3]
  Offset after rotations.
- int simple
  No offset in use if simple==1.
- struct transform_struct strmat
  Simple transformation matrix for no-offset case (=rot2∗rot1).

7.58.1 Detailed Description

Transformation with non-intersecting rotation axes:

Definition at line 1580 of file mc_aux.h.

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

- mc_aux.h

7.59 transform_struct Struct Reference

Coordinate shift and rotation:

`#include <mc_aux.h>`

Data Fields

- double offset [3]

  Offset between coordinate frames (subtracted before forward rotation).
- double rot [3][3]

  Rotation matrix (forward: first index is column, second is row).

7.59.1 Detailed Description

Coordinate shift and rotation:

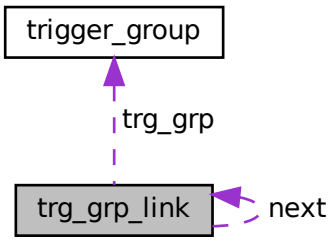
Definition at line 1573 of file mc_aux.h.

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

- mc_aux.h

7.60 trg_grp_link Struct Reference

Collaboration diagram for trg_grp_link:



Data Fields

- struct trigger_group **trg_grp**
- struct trg_grp_link * **next**

7.60.1 Detailed Description

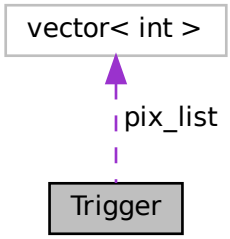
Definition at line 6200 of file sim_imaging.c.

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

- sim_imaging.c

7.61 Trigger Class Reference

Collaboration diagram for Trigger:



Public Member Functions

- void **add_pixel** (int p)
- size_t **size** (void)
- int **operator[ ]** (int i)
- void **add_pixel** (int p)
- size_t **size** (void)
- int **operator[ ]** (int i)

Private Attributes

- vector< int > **pix_list**

7.61.1 Detailed Description

Definition at line 44 of file mapmtpix.cc.

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

- mapmtpix.cc
- smartpix.cc

7.62 trigger_group Struct Reference

A collection of pixels which can form a telescope trigger condition.

`#include <mc_aux.h>`

Data Fields

- int trigger_mode

  0: Discriminator/comparator majority logic, typically with second threshold applied to sum of 'logic' output signals where logic signals may have a finite rise and fall time.
- int num_pixels

  Number of pixels in one group.
- int * pixel

  List of pixel numbers in group (not including pre-sum pixels).
- int with_disc_sums

  Analog pixel sums before majority trigger if non-zero.
- int * num_disc_sums

  Number of discriminator sums in groups.
- int ** disc_sum_from

  List of pixels from where sums are formed (into matching pixel[...] entry).
- int with_fadc_presums

  Digital pixel pre-sums before digital sum trigger if non-zero.
- int * num_fadc_presums

  Number of pre-sums (after shaping and pre-scaling) in groups.
- int ** fadc_presum_from

  List of pixels from where digital pre-sums are formed .
- int num_required

  Number of pixels that must have fired.
- int * req_pixel

  Required pixels list (e.g. SmartPixel).
- int min_pixels

  Minimum no. of pixels required or 0.
- int triggered

  Number of channels with single trigger.
- int triggered_coinc

  Largest number of coincident channels.
- int group_triggered

  Is 1 if group has coincidence trigger.
- double time

  Time of trigger (if any). [ns].
- double true_pe_amp

  Sum of true amplitudes of signal p.e. in contributing pixels [mean p.e.].
- int true_pe_count

  Number of registered signal p.e. in contributing pixels.

7.62.1 Detailed Description

A collection of pixels which can form a telescope trigger condition.

Definition at line 1130 of file mc_aux.h.

7.62.2 Field Documentation

7.62.2.1 trigger_mode

```
int trigger_group::trigger_mode
```

0: Discriminator/comparator majority logic, typically with second threshold applied to sum of 'logic' output signals where logic signals may have a finite rise and fall time.

1: Analog sum mode, with additional shaping of 'discriminator' input pulses. 2: Digital sum mode, with additional shaping applied to digitized signal.

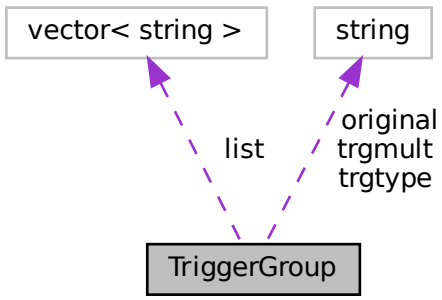
Definition at line 1132 of file mc_aux.h.

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

- mc_aux.h

7.63 TriggerGroup Class Reference

Collaboration diagram for TriggerGroup:



Public Member Functions

- **TriggerGroup** (const string &line)
- ostream & **print** (ostream &os) const

Data Fields

- string **original**
- string **trgtype**
- string **trgmult**
- vector< string > **list**

7.63.1 Detailed Description

Definition at line 185 of file pixel_remap_trg.cc.

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

- pixel_remap_trg.cc

7.64 vector_xy_struct Struct Reference

Vector (2-D) with starting point and direction.

`#include <mc_aux.h>`

Data Fields

- double x
  X positions of polyon corners. [cm].
- double y
  Y positions of polyon corners. [cm].
- double cx
  X direction to next corner. [cm].
- double cy
  Y direction to next corner. [cm].

7.64.1 Detailed Description

Vector (2-D) with starting point and direction.

Definition at line 1611 of file mc_aux.h.

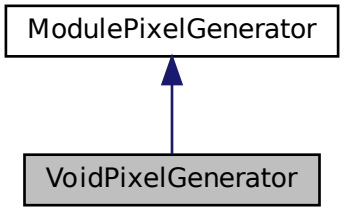
The documentation for this struct was generated from the following file:

- mc_aux.h

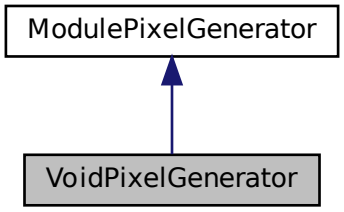
7.65 VoidPixelGenerator Class Reference

A real although dummy generator, not having any pixels.

Inheritance diagram for VoidPixelGenerator:



Collaboration diagram for VoidPixelGenerator:



Public Member Functions

- vector< PixelPos > **PixelPositions** () const
- vector< PixelPos > **NeighbourPositions** () const
- Offset **Offset_u** () const
- Offset **Offset_v** () const
- int **PixelType** () const

7.65.1 Detailed Description

A real although dummy generator, not having any pixels.

Definition at line 243 of file modular_camera.cc.

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

- modular_camera.cc

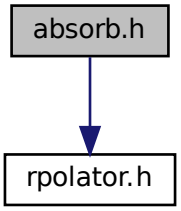
Chapter 8

File Documentation

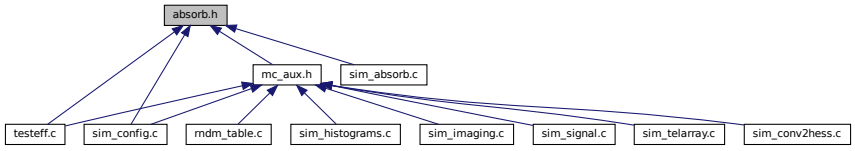
8.1 absorb.h File Reference

Functions definitions for loading transmission, reflectivitiy, and QE tables.

`#include "rpolator.h"`
Include dependency graph for absorb.h:



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



Functions

- double height_for_transmission (void)

  Report back the actual height level for the atmospheric transmission table.
- int read_trans (const char *setup_trans_fname, double altitude)

  Read table of atmospheric transmission data.
- int read_qe_ref (const char *qe_fname, const char *mirror_ref_fname, const char *mirror2_ref_fname, double *quantum_efficiency, double *optics_efficiency, int mirror_class, int bypass_optics, int max_lambda)

  Read quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers.
- double atmospheric_transmission (int iwl, double zem, double airmass)

  Find out direct atmospheric transmission probability between point of emission and point of detection assumed at the nominal altitude for the transmission table.
- double atmospheric_transmission_x (int iwl, double zem, double airmass)

  Same function as atmospheric_transmission() but with explanations.
- double atmospheric_transmission2 (int iwl, double zem, double airmass, double od2tel, double od2focus)

  Find out direct atmospheric transmission probability between point of emission and point of detection.
- int rpt_qe_ref (RpolTable *rpt_qe, RpolTable *rpt_ref, RpolTable *rpt_ref2, double *quantum_efficiency, double *optics_efficiency, int mirror_class, int bypass_optics, int max_lambda, double mirror_degraded, double mirror2_degraded)

  Obtain quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers, similar to read_qe_ref() but from previously read rpolator tables.

8.1.1 Detailed Description

Functions definitions for loading transmission, reflectivitiy, and QE tables.

Author

Konrad Bernloehr

8.1.2 Function Documentation

8.1.2.1 atmospheric_transmission()

```
double atmospheric_transmission (
            int iwl,
            double zem,
            double airmass )
```

Find out direct atmospheric transmission probability between point of emission and point of detection assumed at the nominal altitude for the transmission table.

This function is not suitable for horizontal paths.

Parameters

iwl	Integer step wavelength [nm]
zem	Emission altitude a.s.l. [cm]
airmass	Multiplication factor of traversed atmospheric column due to inclined path. Usually 1/cos(zenith angle).

Returns

Atmospheric transmission probability (between 0 and 1).

Definition at line 634 of file sim_absorb.c.

8.1.2.2 atmospheric_transmission2()

```
double atmospheric_transmission2 (
            int iwl,
            double zem,
            double airmass,
            double od2tel,
            double od2focus )
```

Find out direct atmospheric transmission probability between point of emission and point of detection.

In contrast to atmospheric_transmission() we also include the effects of the telescope altitude above the observation level (or actually the nominal altitude of the transmission table which is supposed to match the height of the CORSIKA observation level) plus an effective pathlength through the instrument. For efficiency reasons, the corresponding optical depths have to be prepared beforehand rather than running the same interpolation again and again. Placement of optical elements of the telescope and actual optical paths are considered too much detail. This function is not suitable for horizontal paths.

Parameters

iwl	Integer step wavelength [nm]
zem	Emission altitude a.s.l. [cm]
airmass	Multiplication factor of traversed atmospheric column due to inclined path. Usually 1/cos(zenith angle).
od2tel	Optical depth between telescope altitude and the nominal transmission table level at the wavelength in question, for a vertical path (to be multiplied with airmass as needed).
od2focus	Optical depth between the telescope altitude and an altitude higher by the effective pathlength through the instrument (a little more than the nominal focal length for 1M but less than that for 2M telescope types), at the given wavelength, with no airmass correction to be applied.

Returns

Atmospheric transmission probability (between 0 and 1).

Definition at line 742 of file sim_absorb.c.

8.1.2.3 read_qe_ref()

```
int read_qe_ref (
            const char * qe_fname,
            const char * mirror_ref_fname,
```

```
            const char * mirror2_ref_fname,
            double * quantum_efficiency,
            double * optics_efficiency,
            int mirror_class,
            int bypass_optics,
            int max_lambda )
```

Read quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers.

Parameters

qe_fname	File name of table with quantum efficiency or PDE.
mirror_ref_fname	File name of mirror reflectivity.
mirror2_ref_fname	File name of secondary mirror reflectivity.
quantum_efficiency	Array for resulting combined efficiency of QE (or PDE) and any reflectivity involved (depending on mirror class and bypass_optics), guaranteed to be at least of size max_lambda (for wavelengths in nanometers).
optics_efficiency	Array for resulting reflectivity which is also applied as part of quantum_efficiency, just to be able to disentangle them where needed. Can be NULL if not used.
mirror_class	Mirror class (0: segmented single reflector made of spherical mirror tiles, 1: single parabolic reflector, 2: dual mirror)
bypass_optics	Option to bypass part or all of the optics in ray-tracing (0: no bypassing, 1: bypass primay, 2: bypass all)
max_lambda	Guaranteed size of quantum_efficiency array and optics_effieinces, i.e. limit of wavelengths (in nanometers) covered.

Returns

0 (OK), -1 (error)

Definition at line 268 of file sim_absorb.c.

8.1.2.4 rpt_qe_ref()

```
int rpt_qe_ref (
            RpolTable * rpt_qe,
            RpolTable * rpt_ref,
            RpolTable * rpt_ref2,
            double * quantum_efficiency,
            double * optics_efficiency,
            int mirror_class,
            int bypass_optics,
            int max_lambda,
            double mirror_degraded,
            double mirror2_degraded )
```

Obtain quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers, similar to read_qe_ref() but from previously read rpolator tables.

If any of the reflectivity tables is a 2-D table, it will count the maximum over all incidence angles.

Parameters

rpt_qe	Rpolator table with quantum efficiency or PDE.
rpt_ref	Rpolator table with mirror reflectivity.
rpt_ref2	Rpolator table with secondary mirror reflectivity (for mirror_class 2 only).
quantum_efficiency	Array for resulting combined efficiency of QE (or PDE) and any reflectivity involved (depending on mirror class and bypass_optics), guaranteed to be at least of size max_lambda (for wavelengths in nanometers). Where 2-D tables are involved this is the maximum for any angle.
optics_efficiency	Array for resulting reflectivity (or product of reflectivities for mirror_class=2) which is also applied as part of quantum_efficiency, just to be able to disentangle them where needed. Where 2-D tables are involved this is the maximum for any angle. Can be NULL if not used.
mirror_class	Mirror class (0: segmented single reflector made of spherical mirror tiles, 1: single parabolic reflector, 2: dual mirror)
bypass_optics	Option to bypass part or all of the optics in ray-tracing (0: no bypassing, 1: bypass primay, 2: bypass all)
max_lambda	Guaranteed size of quantum_efficiency array and optics_effieinces, i.e. limit of wavelengths (in nanometers) covered.
mirror_degraded	Optical efficiency degradation. In case of any bypassing of the optics, this is specifically for the primary mirror (or the lens in case of Fresnel optics). Ignored for bypass_optics >= 1.
mirror2_degraded	Optical efficiency degradation of secondary mirror. Only for dual-mirror optics. Ignored for bypass_optics == 2.

Returns

0 (OK), -1 (error)

Definition at line 459 of file sim_absorb.c.

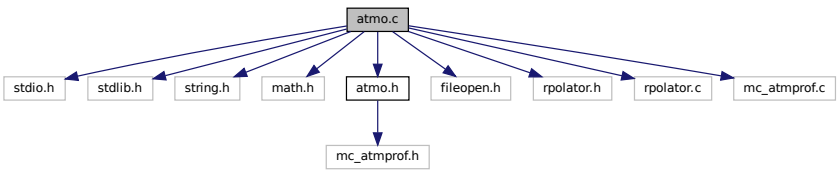
References rpol_table::ndim, rpolate(), rpolate_1d(), and rpol_table::scheme.

8.2 atmo.c File Reference

Use of tabulated atmospheric profiles and atmospheric refraction.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "atmo.h"
#include "fileopen.h"
#include "rpolator.h"
#include "rpolator.c"
#include "mc_atmprof.c"
```

Include dependency graph for atmo.c:



Macros

- #define **NLAYX** 15

- #define M_PI (3.14159265358979323846264338327950288)
  /
- #define **WITH_RPOLATOR_CSPLINE** 1 /∗ Take advantage of cubic splines if we have them available ∗/
- #define **FAST_INTERPOLATION** always
- #define **FAST_INTERPOLATION2** always
- #define **FAST_INTERPOLATION3** always
- #define **WITH_THICKX_DIRECT**
- #define **UNUSED**(x) UNUSED_ ## x
- #define **MAX_PROFILE** 120
- #define **MAX_FAST_PROFILE** 40000
- #define _XSTR_(s) _STR_(s)

  Expand a macro first and then enclose in string.
- #define _STR_(s) #s

  Enclose in string without macro expansion.
- #define SHOW_MACRO(s)

  The following shows the same output after a "#define XYZ" and a "#define XYZ 1" (or compiled with "-DXYZ")
- int atmosphere

  The atmospheric profile number, 0 for built-in.
- static char ∗ atmprof_name = NULL

  File name for atmospheric profile table.
- static int num_prof

  Number of levels in original table.
- static double p_alt [MAX_PROFILE]

  Altitude levels in given table (converted to [cm], upward order).
- static double p_alt_rev [MAX_PROFILE]

  Altitude levels in given table (converted to [cm], reversed order).
- static double p_rho [MAX_PROFILE]

  Density at given levels [g/cm^3] (upward height order)
- static double p_rho_rev [MAX_PROFILE]

  Density at given levels [g/cm^3], reversed order.
- static double p_thick [MAX_PROFILE]

  Atmospheric thickness at given levels [g/cm^2].
- static double p_log_rho [MAX_PROFILE]

  Log of density at given levels (for better interpolation)
- static double p_log_thick [MAX_PROFILE]

  Log of atmospheric thickness at given levels (upward height order)
- static double p_log_thick_rev [MAX_PROFILE]

  Log of atmospheric thickness at given levels (reversed order)
- static double p_log_n1 [MAX_PROFILE]

  Log of (index of refraction minus 1.0) at given levels.
- static double p_bend_ray_hori_a [MAX_PROFILE]

  Coefficient for horizontal displacement refraction effect (reversed order)
- static double p_bend_ray_time_a [MAX_PROFILE]

  Coefficient for arrival time effect by refraction, density dependence (reversed order)
- static double p_bend_ray_time0 [MAX_PROFILE]

  Coefficient for arrival time effect by refraction, altitude dependence.
- static CsplinePar ∗ cs_alt_log_rho

Cubic spline parameters for log(density) versus altitude interpolation.

- static CsplinePar * cs_alt_log_thick

  Cubic spline parameters for log(thickness) versus altitude interpolation.
- static CsplinePar * cs_alt_log_n1

  Cubic spline parameters for log(n-1) versus altitude interpolation.
- static CsplinePar * cs_log_thick_alt

  Cubic spline parameters for altitude versus log(thickness) reverse interpolation.
- static CsplinePar * cs_bend_rho_hori_a

  Cubic spline parameters for horizontal displacement refraction effect.
- static CsplinePar * cs_bend_rho_time_a

  Cubic spline parameters for arrival time effect by refraction, density dependence.
- static CsplinePar * cs_bend_alt_time0

  Cubic spline parameters for arrival time effect by refraction, altitude dependence.
- static double top_of_atmosphere = 112.83e5

  Height of top of atmosphere [cm], = p_alt[num_prof-1].
- static double top_of_atm_table

  The heighest entry in the atmospheric profile table.
- static double bottom_of_atmosphere = 0.

  Bottom is normally at sea level but table could differ, = p_alt[0].
- static double bottom_log_thickness

  Thickness at bottom, depending on profile, = p_log_thick[0].
- static double top_log_thickness

  Thickness at top is zero, thus using an extrapolation from the second last value.
- static double top_layer_2nd_altitude

  Second highest altitude tabulated, using exponential density profile above that.
- static double top_layer_rho0

  Sea-level altitude matching profile in top layer, without scaling for thickness.
- static double top_layer_cfac

  Scaling factor needed to match thickness at second last level.
- static double top_layer_hscale

  Height scale of exponention density profile in top layer.
- static double top_layer_hscale_rho0_cfac

  *top_layer_rho0 * top_layer_cfac * top_layer_hscale*
- static double top_layer_hscale_rho0_cfac_inv

  1.
- static double top_layer_exp_top

  exp(-top_of_atmosphere/top_layer_hscale)
- static double * fast_p_alt

  Equidistant steps in altitude.
- static double * fast_p_log_rho

  Log(rho) at fast_p_alt steps.
- static double * fast_p_log_thick

  Log(thick) at fast_p_alt steps.
- static double * fast_p_log_n1

  Log(n-1) at fast_p_alt steps.
- static double fast_h_fac

  Inverse of height difference per step.
- static double * fast_p_thick

  Direct interpolation of thickness at fast_p_alt steps.
- static double * fast_p_rho

  Direct interpolation of density at fast_p_alt steps.

- static double * fast_p_n1

  Direct interpolation of n-1 at fast_p_alt steps.
- static double * fast_p_eq_log_thick

  Equidistant steps in log(thick)
- static double * fast_p_heigh_rev

  Altitude at given log(thick) steps.
- static double fast_log_thick_fac

  Inverse of log(thickness) difference per step.
- static double * fast_p_bend_ray_hori_a

  Coefficient for horizontal displacement refraction effect.
- static double * fast_p_bend_ray_time_a

  Coefficient for arrival time effect by refraction, density dependence.
- static double * fast_p_bend_ray_time0

  Coefficient for arrival time effect by refraction, altitude dependence.
- static double * fast_p_rho_rev

  Density steps used in fast interpolation, in order of incr.
- static double **fast_rho_fac**
- static double **etadsn** = 0.000283 * 994186.38 / 1222.656
- static double observation_level

  Altitude [cm] of observation level.
- static double **obs_level_refidx**
- static double **obs_level_thick**
- static void init_refraction_tables ()

  Initialize tables needed for atmospheric refraction.
- static void init_fast_interpolation ()

  An alternate interpolation method (which requires that the table is sufficiently fine-grained and equidistant) has to be initialized first.
- static void init_corsika_atmosphere ()

  Take the atmospheric profile from CORSIKA built-in functions.
- static void init_atmosphere_from_text_file ()

  Initialize atmospheric profiles.
- static void **init_atmosphere_from_atmprof** (void)
- static double **sum_log_dev_sq** (double a, double b, double c, int np, double *h, double *t, double *rho)
- static double atm_exp_fit (double h1, double h2, double *ap, double *bp, double *cp, double *s0, int *npp)

  Fit one atmosphere layer by an expontential density model.
- void **free_atmo_csplines** (void)
- static void trace_ray_planar (double emlev, double olev, double za, double step, double *xo, double *to, double *so)

  Trace a downward light ray in a planar atmosphere.
- static void init_common_atmosphere (void)

  Second-stage part of atmospheric profile initialization is common to CORSIKA default profile and tabulated profiles.
- void atmset_ (int *iatmo, double *obslev)

  Set number of atmospheric model profile to be used.
- void atmnam_ (const char *aname, double *obslev)

  Instead of setting the atmospheric profile by number, it gets set by name, also indicated by setting profile number to 99.
- void atm_init (AtmProf *aprof)

  This variant is not usable from the FORTRAN code side, thus no underscore at the end of the function name.
- double rhofx_ (double *height)

  Density of the atmosphere as a function of altitude.
- double thickx_ (double *height)

  *Atmospheric thickness [g/cm**2] as a function of altitude.*

- double refim1x_ (double *height)

  Index of refraction minus 1 as a function of altitude [cm].
- double refidx_ (double *height)

  Index of refraction as a function of altitude [cm].
- double heighx_ (double *thick)

  *Altitude [cm] as a function of atmospheric thickness [g/cm**2].*
- void raybnd_ (double *zem, cors_dbl_t *u, cors_dbl_t *v, double *w, cors_dbl_t *dx, cors_dbl_t *dy, cors_dbl_t *dt)

  Calculate the bending of light due to atmospheric refraction.
- static double sum_log_dev_sq (double a, double b, double c, int np, double *h, double *t, double *UNUSED(rho))

  Measure of deviation of model layers from tables.
- static double fn_thick (double h, int nl, double *hl, double *a, double *b, double *c)

  Corresponding to CORSIKA built-in function THICK; only used to show fit results.
- static double fn_rhof (double h, int nl, double *hl, double *UNUSED(a), double *b, double *c)

  Corresponding to CORSIKA built-in function RHOF; only used to show fit results.
- void atmfit_ (int *nlp, double *hlay, double *aatm, double *batm, double *catm)

  Fit the tabulated density profile for CORSIKA EGS part.
- void **show_atmo_macros** (void)

8.2.1 Detailed Description

Use of tabulated atmospheric profiles and atmospheric refraction.

Author

Konrad Bernloehr

Date

```
CVS $Date: 2023/06/11 20:52:52 $
```

Version

```
CVS $Revision: 1.54 $
```

This file provides code for use of external atmospheric models (in the form of text-format tables) with the CORSIKA program. Six atmospheric models as implemented in the MODTRAN program and as tabulated in MODTRAN documentation (F.X. Kneizys et al. 1996, 'The MODTRAN 2/3 Report and LOWTRAN 7 Model', Phillips Laboratory, Hanscom AFB, MA 01731-3010, U.S.A.) are provided as separate files (atmprof1.dat ... atmprof6.dat). User-provided atmospheric models should be given model numbers above 6. For model number 99 you may actually define a non-standard file name, rather than atmprof99.dat.
Note that for the Cherenkov part and the hadronic (and muon) part of CORSIKA the table values are directly interpolated but the electron/positron/gamma part (derived from EGS) uses special layers (at present 4 with exponential density decrease and the most upper layer with constant density). Parameters of these layers are fitted to tabulated values but not every possible atmospheric model fits very well with an exponential profile. You are adviced to check that the fit matches tabulated values to sufficient precision in the altitude ranges of interest to you. Try to adjust layer boundary altitudes in case of problems. The propagation of light without refraction (as implemented in CORSIKA, unless using the CURVED option) and with refraction (as implemented by this software) assumes a plane-parallel atmosphere. Instead of a single set of starting layer boundaries, newer version (Jan. 2019) try different sets of boundaries and use the most promising set for further fit improvement.
There are different ways to load and interpolate an atmospheric profile table. The usual procedure is that after loading the values from the file, the nearly exponential structure is taken advantage of and a relatively large number

(order 10k) of equidistant support points are pre-interpolated from the original non-equidistant levels of the input point, saving binary-search of the matching interval for each interpolation. This FAST_INTERPOLATION option is activated by default but can be disabled by compiling with '-DNO_FAST_INTERPOLATION' (or '-DNO_FAST_↩ INTERPOLATION2' if only fast interpolation for the reverse direction, from thickness to altitude, should be disabled.) The initial pre-interpolation used to be a linear interpolation in altitude versus log(density) or log(thickness) or log(n-1), respectively but with the 'rpolator' table interpolation code this can be turned into cubic spline interpolation. Unless problems with the CORSIKA build procedure prevent that the rpolator code and its cubic spline interpolation are intended to be the default scheme from now on (Jan. 2019). In either case for the default setting, the rpolator code and be enforced with '-DWITH_RPOLATOR' or disabled with '-DNO_RPOLATOR'. Cubic spline interpolation can be disabled with '-DNO_RPOLATOR_CSPLINE'. The fine-grained interpolation under the FAST_INTERPOLATION scheme continues to be linear in altitude versus log(density) etc.
Because the CORSIKA build procedure does not know yet about the additional source file, it gets included when compiling atmo.c, unless compiled with '-DWITH_RPOLATOR_SEPARATE'.

8.2.2 Macro Definition Documentation

8.2.2.1 SHOW_MACRO

```
#define SHOW_MACRO(
              s )
```

Value:

```
    if ( strcmp(#s,_XSTR_(s)) != 0 ) \
    { if ( strcmp("", _XSTR_(s)) != 0 && strcmp("1", _XSTR_(s)) != 0 ) \
       printf( "    " #s "=" _XSTR_(s) "\n" ); else printf( "    " #s "\n" ); }
```

The following shows the same output after a "#define XYZ" and a "#define XYZ 1" (or compiled with "-DXYZ")
Other non-empty, not "1" assigned values are shown explicitly. Undefined macros are not shown. Neither are any assigned to itself.
Definition at line 3056 of file atmo.c.

8.2.3 Function Documentation

8.2.3.1 atm_init()

```
void atm_init (
            AtmProf * aprof )
```

This variant is not usable from the FORTRAN code side, thus no underscore at the end of the function name.
It is there for doing the atmophere initialization like in the other cases but already passing the pre-filled AtmProf (struct mc_atmprof) along, for example after reading CORSIKA data written by IACT/atmo package version 1.60 and up.

Parameters

aprof	Pointer to atmospheric profile table structure.

Returns

(none)

Definition at line 1197 of file atmo.c.
References atmosphere, atmprof_name, and observation_level.

8.2.3.2 atmfit_()

```
void atmfit_ (
            int * nlp,
```

```
            double * hlay,
            double * aatm,
            double * batm,
            double * catm )
```

Fit the tabulated density profile for CORSIKA EGS part.
Fitting of the tabulated atmospheric density profile by piecewise exponential parts as used in CORSIKA. The fits are constrained by fixing the atmospheric thicknesses at the boundaries to the values obtained from the table. Note that not every atmospheric profile can be fitted well by the CORSIKA piecewise models (4∗exponential + 1∗constant density). In particular, the tropical model is known to be a problem. Setting the boundary heights manually might help. The user is advised to check at least once that the fitted layers represent the tabulated atmosphere sufficiently well, at least at the altitudes most critical for the observations (usually at observation level and near shower maximum but depending on the user's emphasis, this may vary).
Fit all layers (except the uppermost) by exponentials and (if ∗nlp $>$ 0) try to improve fits by adjusting layer boundaries. The uppermost layer has constant density up to the 'edge' of the atmosphere.
This function may be called from CORSIKA.
Parameters (all pointers since function is called from Fortran):

Parameters

nlp	Number of layers (5, or negative of that if boundaries set manually)
hlay	Vector of layer (lower) boundaries.
aatm,batm,catm	Parameters as used in CORSIKA.

Definition at line 2101 of file atmo.c.

8.2.3.3 atmnam_()

```
void atmnam_ (
            const char * aname,
            double * obslev )
```

Instead of setting the atmospheric profile by number, it gets set by name, also indicated by setting profile number to 99.
It does not actually initialize the atmosphere - this still should be done with atmset, using profile number 99 for that purpose. You need to call atmnam_ before atmset to make use of this feature.
Definition at line 1148 of file atmo.c.
References atmosphere, atmprof_name, init_atmosphere_from_text_file(), and observation_level.

8.2.3.4 atmset_()

```
void atmset_ (
            int * iatmo,
            double * obslev )
```

Set number of atmospheric model profile to be used.
The atmospheric model is initialized first before the interpolating functions can be used. For efficiency reasons, the functions rhofx_(), thickx_(), ... don't check if the initialisation was done.
This function is called if the 'ATMOSPHERE' keyword is present in the CORSIKA input file.
The function may be called from CORSIKA to initialize the atmospheric model via 'CALL ATMSET(IATMO,OBSLEV)' or such.

Parameters

iatmo	(pointer to) atmospheric profile number; negative for CORSIKA built-in profiles.
obslev	(pointer to) altitude of observation level [cm]

Returns

(none)

Definition at line 1115 of file atmo.c.
References atmosphere, atmprof_name, init_atmosphere_from_text_file(), init_corsika_atmosphere(), and observation_level.

8.2.3.5 heighx_()

```
double heighx_ (
            double * thick )
```

Altitude [cm] as a function of atmospheric thickness [g/cm**2].
This function can be called from Fortran code as HEIGHX(THICK).

Parameters

thick	(pointer to) atmospheric thickness [g/cm**2]

Returns

altitude [cm]

Definition at line 1403 of file atmo.c.
References bottom_log_thickness, bottom_of_atmosphere, fast_log_thick_fac, num_prof, p_thick, top_layer_↩ 2nd_altitude, top_layer_exp_top, top_layer_hscale, top_layer_hscale_rho0_cfac_inv, top_layer_rho0, top_log_↩ thickness, and top_of_atmosphere.

8.2.3.6 init_atmosphere_from_text_file()

```
static void init_atmosphere_from_text_file (
            void ) [static]
```

Initialize atmospheric profiles.
Internal function for initialising both external and CORSIKA built-in atmospheric profiles. If any CORSIKA built-in profile should be used, it simply calls init_corsika_atmosphere().
Otherwise, atmospheric models are read in from text-format tables. The supplied models 1-6 are based on output of the MODTRAN program. For the interpolation of relevant parameters (density, thickness, index of refraction, ...) all parameters are transformed such that linear interpolation can be easily used.
Definition at line 888 of file atmo.c.
References atmosphere, atmprof_name, init_corsika_atmosphere(), num_prof, and read_table_v().
Referenced by atmnam_(), and atmset_().

8.2.3.7 init_corsika_atmosphere()

```
static void init_corsika_atmosphere (
            void ) [static]
```

Take the atmospheric profile from CORSIKA built-in functions.
For use of the refraction bending corrections together with the CORSIKA built-in atmospheres, the atmosphere tables are constructed from the CORSIKA RHOF and THICK functions. Note that the refraction index in this case is without taking the effect of wator vapour into account.
Definition at line 821 of file atmo.c.
References atmosphere, and top_of_atmosphere.
Referenced by atmset_(), and init_atmosphere_from_text_file().

8.2.3.8 init_refraction_tables()

```
static void init_refraction_tables (
            void  )  [static]
```

Initialize tables needed for atmospheric refraction.
Initialize the correction tables used for the refraction bending of the light paths. It is called once after the atmospheric profile has been defined.
Definition at line 382 of file atmo.c.
Referenced by init_common_atmosphere().

8.2.3.9 raybnd_()

```
void raybnd_ (
            double * zem,
            cors_dbl_t * u,
            cors_dbl_t * v,
            double * w,
            cors_dbl_t * dx,
            cors_dbl_t * dy,
            cors_dbl_t * dt )
```

Calculate the bending of light due to atmospheric refraction.
Path of light through the atmosphere including the bending by refraction. This function assumes a plane-parallel atmosphere. Coefficients for corrections from straight-line propagation to refraction-bent path are numerically evaluated when the atmospheric model is defined. Note that while the former mix of double/float data types may appear odd, it was determined by the variables present in older CORSIKA to save conversions. With CORSIKA 6.0 all parameters are of double type.
This function may be called from FORTRAN as CALL RAYBND(ZEM,U,V,W,DX,DY,DT)

Parameters

zem	Altitude of emission above sea level [cm]
u	Initial/Final direction cosine along X axis (updated)
v	Initial/Final direction cosine along Y axis (updated)
w	Initial/Final direction cosine along Z axis, positive is downwards (updated)
dx	Position in CORSIKA detection plane [cm] (updated)
dy	Position in CORSIKA detection plane [cm] (updated)
dt	Time of photon [ns]. Input: emission time. Output: time of arrival in CORSIKA detection plane.

Definition at line 1502 of file atmo.c.
References observation_level, and rhofx_().

8.2.3.10 refidx_()

```
double refidx_ (
            double * height )
```

Index of refraction as a function of altitude [cm].
This function can be called from Fortran code as REFIDX(HEIGHT).

Parameters

height	(pointer to) altitude [cm]

Returns

index of refraction

Definition at line 1387 of file atmo.c.
References refim1x_().
Referenced by trace_ray_planar().

8.2.3.11 refim1x_()

```
double refim1x_ (
            double * height )
```

Index of refraction minus 1 as a function of altitude [cm].
This function can be called from Fortran code as REFIM1X(HEIGHT). This part has been split off from refidx_ in order to be able to access values small compared to 1.0 without big (relative) rounding errors.

Parameters

height	(pointer to) altitude [cm]

Returns

index of refraction minus one

Definition at line 1345 of file atmo.c.
References bottom_of_atmosphere, fast_p_alt, fast_p_n1, p_log_n1, rpol_cspline(), and top_of_atmosphere.
Referenced by refidx_().

8.2.3.12 rhofx_()

```
double rhofx_ (
            double * height )
```

Density of the atmosphere as a function of altitude.
This function can be called from Fortran code as RHOFX(HEIGHT).

Parameters

height	(pointer to) altitude [cm]

Returns

density [g/cm∗∗3]

Definition at line 1235 of file atmo.c.
References bottom_of_atmosphere, fast_p_alt, fast_p_rho, p_rho, rpol_cspline(), and top_of_atmosphere.
Referenced by raybnd_().

8.2.3.13 thickx_()

```
double thickx_ (
            double * height )
```

Atmospheric thickness [g/cm∗∗2] as a function of altitude.
This function can be called from Fortran code as THICKX(HEIGHT).

Parameters

height	(pointer to) altitude [cm]

Returns

thickness [g/cm∗∗2]

Definition at line 1283 of file atmo.c.
References bottom_of_atmosphere, fast_p_alt, fast_p_thick, num_prof, p_alt, p_thick, rpol_cspline(), top_layer↩
_2nd_altitude, top_layer_exp_top, top_layer_hscale, top_layer_hscale_rho0_cfac, top_layer_rho0, and top_of_↩
atmosphere.

8.2.3.14 trace_ray_planar()

```
static void trace_ray_planar (
            double emlev,
            double olev,
            double za,
            double step,
            double * xo,
            double * to,
            double * so )  [static]
```

Trace a downward light ray in a planar atmosphere.

Parameters

emlev	Emission level altitude [cm]
olev	Observation level altitude [cm] (below emlev)
za	Zenith angle (positive downwards)
step	Step length [cm]
xo	Arrival position at observation level [cm] (returned only), positive value expected for downward bending.
to	Travel time to observation level [ns] (returned only).
so	Path length travelled [cm] (returned only).

Definition at line 319 of file atmo.c.
References h2, and refidx_().

8.2.4 Variable Documentation

8.2.4.1 fast_p_rho_rev

```
double* fast_p_rho_rev  [static]
```

Density steps used in fast interpolation, in order of incr.
density
Definition at line 251 of file atmo.c.

8.2.4.2 top_layer_hscale_rho0_cfac_inv

```
double top_layer_hscale_rho0_cfac_inv  [static]
```

1.
/(top_layer_rho0 ∗ top_layer_cfac ∗ top_layer_hscale)
Definition at line 226 of file atmo.c.
Referenced by heighx_(), and init_common_atmosphere().

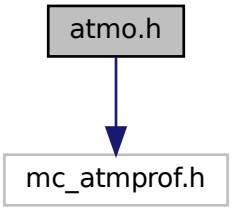
8.2.4.3 top_log_thickness

```
double top_log_thickness  [static]
```

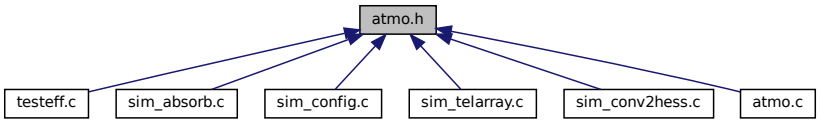
Thickness at top is zero, thus using an extrapolation from the second last value.
for fast interpolation in reverse direction (thickness to height).
Definition at line 218 of file atmo.c.
Referenced by heighx_().

8.3 atmo.h File Reference

Use of tabulated atmospheric profiles and atmospheric refraction.
`#include "mc_atmprof.h"`
Include dependency graph for atmo.h:



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



- #define CORSIKA_VERSION 6900
  /
- #define **VECTOR_SIZE** 4
- typedef double **cors_dbl_t**
- void atmset_ (int *iatmo, double *obslev)

  Set number of atmospheric model profile to be used.
- void atmnam_ (const char *aname, double *obslev)

  Instead of setting the atmospheric profile by number, it gets set by name, also indicated by setting profile number to 99.
- void atm_init (AtmProf *aprof)

  This variant is not usable from the FORTRAN code side, thus no underscore at the end of the function name.
- double rhofx_ (double *height)

  Density of the atmosphere as a function of altitude.
- double thickx_ (double *height)

  *Atmospheric thickness [g/cm**2] as a function of altitude.*
- double refim1x_ (double *height)

  Index of refraction minus 1 as a function of altitude [cm].
- double refidx_ (double *height)

Index of refraction as a function of altitude [cm].

- double heighx_ (double *thick)

  *Altitude [cm] as a function of atmospheric thickness [g/cm**2].*

- void raybnd_ (double *zem, cors_dbl_t *u, cors_dbl_t *v, double *w, cors_dbl_t *dx, cors_dbl_t *dy, cors_dbl_t *dt)

  Calculate the bending of light due to atmospheric refraction.

- void **raybnd_vec_** (double *zem, cors_dbl_t *u, cors_dbl_t *v, double *w, cors_dbl_t *dx, cors_dbl_t *dy, cors_dbl_t *dt)
- void atmfit_ (int *nlp, double *hlay, double *aatm, double *batm, double *catm)

  Fit the tabulated density profile for CORSIKA EGS part.

- double rhof_ (double *height)

  The CORSIKA built-in density lookup function.

- double thick_ (double *height)

  The CORSIKA built-in function for vertical atmospheric thickness (overburden).

- double heigh_ (double *thick)

  The CORSIKA built-in function for the height as a function of overburden.

8.3.1 Detailed Description

Use of tabulated atmospheric profiles and atmospheric refraction.

Author

Konrad Bernloehr

Date

CVS $Date: 2021/10/23 11:44:56 $

Version

CVS $Revision: 1.14 $

8.3.2 Function Documentation

8.3.2.1 atm_init()

```
void atm_init (
            AtmProf * aprof )
```

This variant is not usable from the FORTRAN code side, thus no underscore at the end of the function name.
It is there for doing the atmophere initialization like in the other cases but already passing the pre-filled AtmProf (struct mc_atmprof) along, for example after reading CORSIKA data written by IACT/atmo package version 1.60 and up.

Parameters

aprof	Pointer to atmospheric profile table structure.

Returns

(none)

Definition at line 1197 of file atmo.c.
References atmosphere, atmprof_name, and observation_level.

8.3.2.2 atmfit_()

```
void atmfit_ (
              int * nlp,
              double * hlay,
              double * aatm,
              double * batm,
              double * catm )
```

Fit the tabulated density profile for CORSIKA EGS part.
Fitting of the tabulated atmospheric density profile by piecewise exponential parts as used in CORSIKA. The fits are constrained by fixing the atmospheric thicknesses at the boundaries to the values obtained from the table. Note that not every atmospheric profile can be fitted well by the CORSIKA piecewise models (4∗exponential + 1∗constant density). In particular, the tropical model is known to be a problem. Setting the boundary heights manually might help. The user is advised to check at least once that the fitted layers represent the tabulated atmosphere sufficiently well, at least at the altitudes most critical for the observations (usually at observation level and near shower maximum but depending on the user's emphasis, this may vary).
Fit all layers (except the uppermost) by exponentials and (if ∗nlp $>$ 0) try to improve fits by adjusting layer boundaries. The uppermost layer has constant density up to the 'edge' of the atmosphere.
This function may be called from CORSIKA.
Parameters (all pointers since function is called from Fortran):

Parameters

nlp	Number of layers (5, or negative of that if boundaries set manually)
hlay	Vector of layer (lower) boundaries.
aatm,batm,catm	Parameters as used in CORSIKA.

Definition at line 2101 of file atmo.c.

8.3.2.3 atmnam_()

```
void atmnam_ (
              const char * aname,
              double * obslev )
```

Instead of setting the atmospheric profile by number, it gets set by name, also indicated by setting profile number to 99.
It does not actually initialize the atmosphere - this still should be done with atmset, using profile number 99 for that purpose. You need to call atmnam_ before atmset to make use of this feature.
Definition at line 1148 of file atmo.c.
References atmosphere, atmprof_name, init_atmosphere_from_text_file(), and observation_level.

8.3.2.4 atmset_()

```
void atmset_ (
              int * iatmo,
              double * obslev )
```

Set number of atmospheric model profile to be used.
The atmospheric model is initialized first before the interpolating functions can be used. For efficiency reasons, the functions rhofx_(), thickx_(), ... don't check if the initialisation was done.
This function is called if the 'ATMOSPHERE' keyword is present in the CORSIKA input file.
The function may be called from CORSIKA to initialize the atmospheric model via 'CALL ATMSET(IATMO,OBSLEV)' or such.

Parameters

iatmo	(pointer to) atmospheric profile number; negative for CORSIKA built-in profiles.
obslev	(pointer to) altitude of observation level [cm]

Returns

(none)

Definition at line 1115 of file atmo.c.
References atmosphere, atmprof_name, init_atmosphere_from_text_file(), init_corsika_atmosphere(), and observation_level.

8.3.2.5 heigh_()

```
double heigh_ (
            double * thick )
```

The CORSIKA built-in function for the height as a function of overburden.
The CORSIKA built-in function for the height as a function of overburden.
Definition at line 182 of file sim_telarray.c.

8.3.2.6 heighx_()

```
double heighx_ (
            double * thick )
```

Altitude [cm] as a function of atmospheric thickness [g/cm**2].
This function can be called from Fortran code as HEIGHX(THICK).

Parameters

thick	(pointer to) atmospheric thickness [g/cm**2]

Returns

altitude [cm]

Definition at line 1403 of file atmo.c.
References bottom_log_thickness, bottom_of_atmosphere, fast_log_thick_fac, num_prof, p_thick, top_layer_↩ 2nd_altitude, top_layer_exp_top, top_layer_hscale, top_layer_hscale_rho0_cfac_inv, top_layer_rho0, top_log_↩ thickness, and top_of_atmosphere.

8.3.2.7 raybnd_()

```
void raybnd_ (
            double * zem,
            cors_dbl_t * u,
            cors_dbl_t * v,
            double * w,
            cors_dbl_t * dx,
            cors_dbl_t * dy,
            cors_dbl_t * dt )
```

Calculate the bending of light due to atmospheric refraction.
Path of light through the atmosphere including the bending by refraction. This function assumes a plane-parallel atmosphere. Coefficients for corrections from straight-line propagation to refraction-bent path are numerically evaluated when the atmospheric model is defined. Note that while the former mix of double/float data types may appear odd, it was determined by the variables present in older CORSIKA to save conversions. With CORSIKA 6.0 all parameters are of double type.
This function may be called from FORTRAN as CALL RAYBND(ZEM,U,V,W,DX,DY,DT)

Parameters

zem	Altitude of emission above sea level [cm]

Parameters

u	Initial/Final direction cosine along X axis (updated)
v	Initial/Final direction cosine along Y axis (updated)
w	Initial/Final direction cosine along Z axis, positive is downwards (updated)
dx	Position in CORSIKA detection plane [cm] (updated)
dy	Position in CORSIKA detection plane [cm] (updated)
dt	Time of photon [ns]. Input: emission time. Output: time of arrival in CORSIKA detection plane.

Definition at line 1502 of file atmo.c.
References observation_level, and rhofx_().

8.3.2.8 refidx_()

```
double refidx_ (
            double * height )
```

Index of refraction as a function of altitude [cm].
This function can be called from Fortran code as REFIDX(HEIGHT).

Parameters

height	(pointer to) altitude [cm]

Returns

index of refraction

Definition at line 1387 of file atmo.c.
References refim1x_().
Referenced by trace_ray_planar().

8.3.2.9 refim1x_()

```
double refim1x_ (
            double * height )
```

Index of refraction minus 1 as a function of altitude [cm].
This function can be called from Fortran code as REFIM1X(HEIGHT). This part has been split off from refidx_ in order to be able to access values small compared to 1.0 without big (relative) rounding errors.

Parameters

height	(pointer to) altitude [cm]

Returns

index of refraction minus one

Definition at line 1345 of file atmo.c.
References bottom_of_atmosphere, fast_p_alt, fast_p_n1, p_log_n1, rpol_cspline(), and top_of_atmosphere.
Referenced by refidx_().

8.3.2.10 rhof_()

```
double rhof_ (
            double * height )
```

The CORSIKA built-in density lookup function.
The CORSIKA built-in density lookup function.
[cm].
Definition at line 153 of file sim_telarray.c.

8.3.2.11 rhofx_()

```
double rhofx_ (
            double * height )
```

Density of the atmosphere as a function of altitude.
This function can be called from Fortran code as RHOFX(HEIGHT).

Parameters

height	(pointer to) altitude [cm]

Returns

density [g/cm∗∗3]

Definition at line 1235 of file atmo.c.
References bottom_of_atmosphere, fast_p_alt, fast_p_rho, p_rho, rpol_cspline(), and top_of_atmosphere.
Referenced by raybnd_().

8.3.2.12 thick_()

```
double thick_ (
            double * height )
```

The CORSIKA built-in function for vertical atmospheric thickness (overburden).
The CORSIKA built-in function for vertical atmospheric thickness (overburden).
[cm].
Definition at line 168 of file sim_telarray.c.

8.3.2.13 thickx_()

```
double thickx_ (
            double * height )
```

Atmospheric thickness [g/cm∗∗2] as a function of altitude.
This function can be called from Fortran code as THICKX(HEIGHT).

Parameters

height	(pointer to) altitude [cm]

Returns

thickness [g/cm∗∗2]

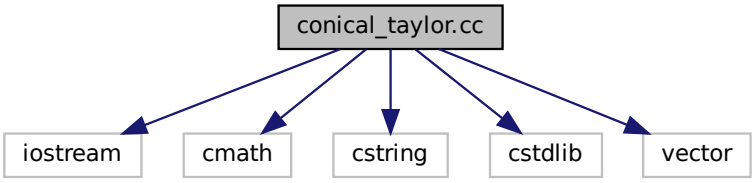
Definition at line 1283 of file atmo.c.
References bottom_of_atmosphere, fast_p_alt, fast_p_thick, num_prof, p_alt, p_thick, rpol_cspline(), top_layer↩
_2nd_altitude, top_layer_exp_top, top_layer_hscale, top_layer_hscale_rho0_cfac, top_layer_rho0, and top_of_↩
atmosphere.

8.4 conical_taylor.cc File Reference

Convert from Zemax-style surface definition to pure Taylor expansion.

```
#include <iostream>
#include <cmath>
#include <cstring>
#include <cstdlib>
#include <vector>
```

Include dependency graph for conical_taylor.cc:



Functions

- vector< double > contaylor (double c, double k, size_t n)

  Taylor series expansion of a function of the form z(r) = (c∗r^2) / (1 + sqrt(1-(1+k)∗c^2∗r^2)) around r_0 = 0.
- void **syntax** (const char ∗prg)
- int **main** (int argc, char ∗∗argv)

8.4.1 Detailed Description

Convert from Zemax-style surface definition to pure Taylor expansion.
Not needed for any sim_telarray productions.

Author

Konrad Bernloehr

8.4.2 Function Documentation

8.4.2.1 contaylor()

```
vector<double> contaylor (
            double c,
            double k,
            size_t n )
```

Taylor series expansion of a function of the form z(r) = (c∗r^2) / (1 + sqrt(1-(1+k)∗c^2∗r^2)) around r_0 = 0.
Only even coefficients can result from it.
Definition at line 45 of file conical_taylor.cc.

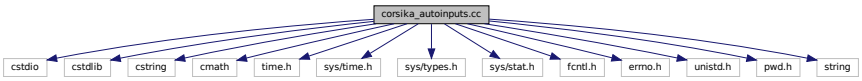
8.5 corsika_autoinputs.cc File Reference

A program for filling random and host/user dependent parameters in CORSIKA configuration.

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <time.h>
```

```
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <pwd.h>
#include <string>
```

Include dependency graph for corsika_autoinputs.cc:



Macros

- #define **NEED_DEV_URANDOM** 1

Functions

- int **new_run** (const string &conf_path)
- void **syntax** ()
- int main (int argc, char **argv)

8.5.1 Detailed Description

A program for filling random and host/user dependent parameters in CORSIKA configuration.

Author

Konrad Bernloehr

The CORSIKA parameters RUNNR, HOST, USER, and SEED (4 times) will be written to standard output, with the run number automatically incremented from a file "run.conf" and the seeds randomized. The "run.conf" should be in a common directory (write access required) and not in the working directory of the CORSIKA run. Proper file locking is attempted when multiple jobs are started at once.
CORSIKA can optionally be asked to run in a dedicated working directory.

```
Program syntax: ./corsika_autoinputs [ options ] [ template_inputs ]
Options:
    --run corsika_binary
                Run specified CORSIKA binary, see details below.
    --run-number rnum (or: -R rnum)
                Use specific run number.
    --epos      Generate basic input cards needed to run with EPOS model.
                Also activated if the CORSIKA_EPOS environment variable is set
                or the real CORSIKA program file name contains '_EPOS_'.
    --no-epos   This is not with EPOS model, even if CORSIKA_EPOS is set
                or the real program file name contains '_EPOS_'.
    --transmission fname   If an atmospheric transmission table is needed
                (only when compiling with iact_full5.c under certain
                conditions, never with iact.c) a symlink for that can be
                attempted as well.
    --efficiency fname   Simular, if a quantum efficiency table is needed.
                (Note: this is not for the CEFFIC option.)
    --reflectivity fname   Simular, mirror reflectivity.
    --generate-run-number (or: -g; use -G to format as run sub-directory)
    --path config_path (or: -p config_path)
    --keep-seeds
                Keep the SEED parameters from the template inputs.
    --path run-conf-path  (or '-p ...' or environment variable RUN_CONF_PATH)
                Set path like for 'config_path' argument.
```

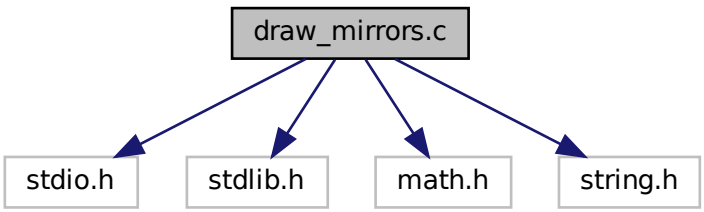
```
  --corsika-version version  (or '-v ...' or CORSIKA_VERSION environment variable)
              Process for an assumed 6.x or 7.x CORSIKA version.
with the following parameters:
  corsika_binary: If present, a working directory will be created and the
              CORSIKA program will be run.
  rnum:       A fixed run number, typically defined at job submission,
              instead of automatically incrementing run numbers.
  config_path: The path name where a file "run.conf" is maintained,
              containing a run number to be incremented by one whenever
              this program is run (and the '--run-number' option not used).
              If the path is not specified by a "-p" option, and not by
              a RUN_CONF_PATH environment variable, and not by a
              path component of the optional template_inputs argument,
              the current directory is assumed (".").
  template_inputs: A CORSIKA configuration file which may or may not
              contain the input parameters defined by this program.
              All lines not starting with one of the parameter names
              RUNNR, HOST, USER, SEED will be copied to standard output.
              If no such file is specified, only the RUNNR, HOST, USER, and SEED
              seed parameters are written to standard output.
```

8.6 draw_mirrors.c File Reference

Draw single-reflector telescope mirror facets in FIG format.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
```

Include dependency graph for draw_mirrors.c:



Data Structures

- struct mirror_struct

  Parameters for a telescope mirror:

Macros

- #define **NPSC** 200

Typedefs

- typedef struct mirror_struct **MIRROR**

Functions

- int **add_mirror** (double xm, double ym, double zm, double radmin, double radmax, double diameter, double dflen, double cflen, double mflen, int shape)

- double **xpro** (double x, double y, double z, double theta_pro, double phi_pro)
- double **ypro** (double x, double y, double z, double theta_pro, double phi_pro)
- int plot_mirror_3d (int im, FILE ∗out, double rad, double theta_pro, double phi_pro, double dflen, double cflen, int is_dc, double cam_x, double cam_y)

  Plot a mirror of given position, size, shape according to the dish shape and focal length in some projection.
- int **plot_camera_3d** (double rcam, FILE ∗out, double rad, double theta_pro, double phi_pro, double dflen, double cflen, int camera_shape, double shift_x, double shift_y)
- void **plot_scale** (FILE ∗out, double rad, double theta_pro, double phi_pro, double flen, double shift_x, double shift_y, double cross_x, double cross_y)
- void **syntax** (const char ∗prg)
- int **main** (int argc, char ∗∗argv)

Variables

- int **nmirror**
- MIRROR **mirror** [5000]
- double **total_area** = 0.
- double **surface_area** = 0.
- double **rmx** = 0.0
- double **rmax** = 0.
- double **rmax_c** = 0.
- double **rot** = 0.
- int **show_id** = 0

8.6.1 Detailed Description

Draw single-reflector telescope mirror facets in FIG format.

Author

Konrad Bernloehr

8.6.2 Function Documentation

8.6.2.1 plot_mirror_3d()

```
int plot_mirror_3d (
            int im,
            FILE * out,
            double rad,
            double theta_pro,
            double phi_pro,
            double dflen,
            double cflen,
            int is_dc,
            double cam_x,
            double cam_y )
```

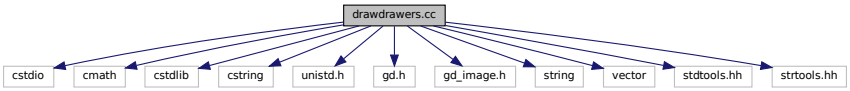
Plot a mirror of given position, size, shape according to the dish shape and focal length in some projection.
Note that the 'dish focal length' parameter used to interpreted differently, depending on whether it was compiled for Davies-Cotton or not. Now a run-time parameter (is_dc).
Definition at line 159 of file draw_mirrors.c.

8.7 drawdrawers.cc File Reference

Draw an image of a camera as indicated in the camera definition file.

```
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <unistd.h>
#include "gd.h"
#include "gd_image.h"
#include <string>
#include <vector>
#include "stdtools.hh"
#include "strtools.hh"
```

Include dependency graph for drawdrawers.cc:



Data Structures

- class PixPos
- class PixVect

Functions

- int **main** ()

8.7.1 Detailed Description

Draw an image of a camera as indicated in the camera definition file.

Author

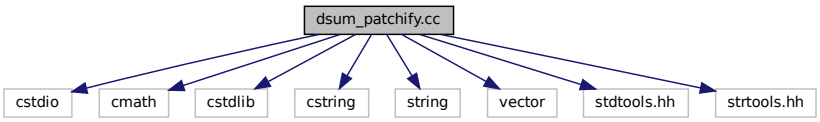
Konrad Bernloehr

8.8 dsum_patchify.cc File Reference

Convert a sim_telarray camera configuration with DigitalSumTrigger lines in plain (non-patch) format into one with pixels grouped into "patches", using the same as already used for majority trigger logic with preceding "super-pixel" analog sum.

```
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <string>
#include <vector>
#include "stdtools.hh"
#include "strtools.hh"
```

Include dependency graph for dsum_patchify.cc:



Data Structures

- class Offset

  Offset of pixels or modules w.r.t.
- class PixelType
- class PixelPos

  Positions of pixels, also including pixel shape type (although not really needed)

Functions

- ostream & **operator**<< (ostream &os, const Offset &o)
- ostream & **operator**<< (ostream &os, const PixelPos &p)
- void **syntax** (int argc, char **argv, int iarg)
- template<class T >
  void **print_implode** (vector< T > v, const string &d=",", const string &dbeg="", const string &dend="", const string &dsec="(same)")
- int **main** (int argc, char **argv)

8.8.1 Detailed Description

Convert a sim_telarray camera configuration with DigitalSumTrigger lines in plain (non-patch) format into one with pixels grouped into "patches", using the same as already used for majority trigger logic with preceding "super-pixel" analog sum.
That means converting from something like DigitalSumTrigger * of 1 2 3 4 5 6 to DigitalSumTrigger * of 1[2,3] 4[5,6]

Author

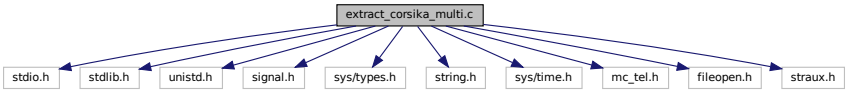
Konrad Bernloehr

8.9 extract_corsika_multi.c File Reference

This file extracts data for selected one or more selected sets of telescopes from CORSIKA IACT data.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <string.h>
#include <sys/time.h>
#include "mc_tel.h"
#include "fileopen.h"
#include "straux.h"
```

Include dependency graph for extract_corsika_multi.c:



Macros

- #define **MAXTEL** 1000
- #define **MAXSELECT** 20
- #define **MAX_ARRAY** 200

Functions

- void stop_signal_function (int isig)

  Stop the program gracefully when it catches an INT or TERM signal.
- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- void pipe_signal_function (int isig)

  This signal handler gets involved when a SIGPIPE is produced on input or output.
- void format_num (char ∗text, size_t maxl, double num)

  Special formatting of floating point numbers.
- void hup_signal_function (int __attribute__((unused)) isig)

  Produce intermediate output when the program catches an HUP signal.
- void syntax (void)

  Tell how to run this program.
- int **main** (int argc, char ∗∗argv)

Variables

- static int **interrupted** = 0
- static int **broken** = 0

8.9.1 Detailed Description

This file extracts data for selected one or more selected sets of telescopes from CORSIKA IACT data.
This is derived from extract_corsika_tel.c.

Author

Konrad Bernloehr

Date

2015-04-27 (initial version)

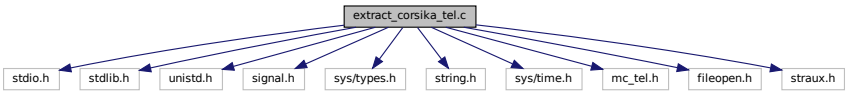
8.10 extract_corsika_tel.c File Reference

This file extracts data for selected telescopes from CORSIKA IACT data.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
```

```
#include <sys/types.h>
#include <string.h>
#include <sys/time.h>
#include "mc_tel.h"
#include "fileopen.h"
#include "straux.h"
```

Include dependency graph for extract_corsika_tel.c:



Macros

- #define **MAXTEL** 1000
- #define **MAX_ARRAY** 200

Functions

- void stop_signal_function (int isig)

  Stop the program gracefully when it catches an INT or TERM signal.
- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- void pipe_signal_function (int isig)

  This signal handler gets involved when a SIGPIPE is produced on input or output.
- void format_num (char ∗text, size_t maxl, double num)

  Special formatting of floating point numbers.
- void hup_signal_function (int __attribute__((unused)) isig)

  Produce intermediate output when the program catches an HUP signal.
- int setenv_int_array (const char ∗name, int ∗ival, int nval)

  Set an environment value with a comma-separated list of integers.
- int setenv_dbl_array (const char ∗name, double ∗dval, int nval)

  Set an environment value with a comma-separated list of floating point values.
- void syntax (void)

  Tell how to run this program.
- int **main** (int argc, char ∗∗argv)

Variables

- static int **interrupted** = 0
- static int **broken** = 0

8.10.1 Detailed Description

This file extracts data for selected telescopes from CORSIKA IACT data.
It also sets up the same environment variables as multipipe_corsika.

Author

Konrad Bernloehr

Date

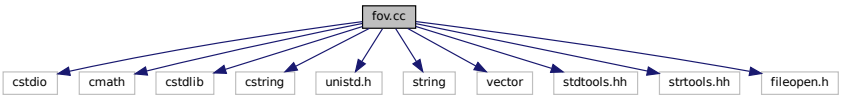
2013-02-26 (initial version)

8.11 fov.cc File Reference

Calculate camera field-of-view according to different definitions.

```
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <unistd.h>
#include <string>
#include <vector>
#include "stdtools.hh"
#include "strtools.hh"
#include "fileopen.h"
```

Include dependency graph for fov.cc:



Data Structures

- class Offset

  Offset of pixels or modules w.r.t.
- class PixelType
- class PixelPos

  Positions of pixels, also including pixel shape type (although not really needed)
- class PixelList

  Lists of pixels, holding both positions and pixel IDs.

Functions

- ostream & **operator**<< (ostream &os, const Offset &o)
- ostream & **operator**<< (ostream &os, const PixelPos &p)
- ostream & **operator**<< (ostream &os, const PixelList &p)
- int **topo_nb_check** (double x1, double y1, double x2, double y2, int shape, double r)
- double **radial_pixel_intersect** (double phi, const PixelPos &p, double r, int shape)
- void **syntax** (int argc, char **argv, int iarg)
- int **main** (int argc, char **argv)

8.11.1 Detailed Description

Calculate camera field-of-view according to different definitions.

Author

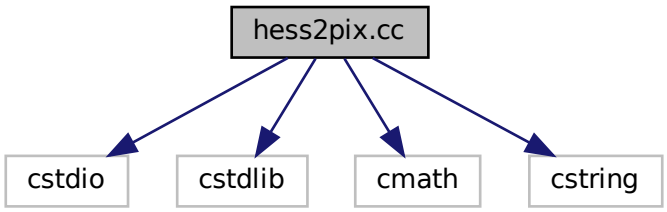
Konrad Bernloehr

8.12 hess2pix.cc File Reference

Generate H.E.S.S.

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
```

```
#include <cstring>
```
Include dependency graph for hess2pix.cc:



Macros

- #define **NUM_DRAWERS** 128
- #define **MAX_PIX** 2048
- #define **NUM_SECTORS** (12∗13)

Functions

- int **main** (int argc, char ∗∗argv)

8.12.1 Detailed Description

Generate H.E.S.S.
II camera pixel list for 2048 pixels camera.

Author

Konrad Bernloehr

8.13 hess_defaults.h File Reference

Setup of parameters for generic telescope simulatation with defaults for H.E.S.S.

Variables

- static CONFIG_ITEM cfgitems [ ]

  Default configuration parameters for H.E.S.S.
- static CONFIG_ITEM cfg_trans [ ]

  Atmospheric transparency table filename.
- static CONFIG_ITEM cfg_main [ ]

  Global configuration parameters.

8.13.1 Detailed Description

Setup of parameters for generic telescope simulatation with defaults for H.E.S.S.

Author

Konrad Bernloehr

8.13.2 Variable Documentation

8.13.2.1 cfg_trans

```
CONFIG_ITEM cfg_trans[]  [static]
```

Initial value:

```
=
{
   { "ATMOSPHERIC_TRANSMISSION", "Text", sizeof(setup_trans_fname)-1,
        setup_trans_fname, NULL,
        "hess_atmo_trans.dat"
   },
   { NULL_CONFIG_ITEM }
}
```

Atmospheric transparency table filename.
Definition at line 462 of file hess_defaults.h.
Referenced by config_setup_atm_transmission().

8.13.2.2 cfgitems

```
CONFIG_ITEM cfgitems[]  [static]
```

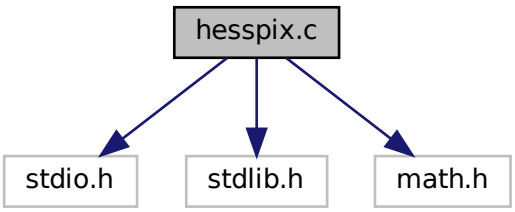
Default configuration parameters for H.E.S.S.
telescope simulatation
Definition at line 38 of file hess_defaults.h.
Referenced by config_setup_imaging().

8.14 hesspix.c File Reference

Generate H.E.S.S.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

Include dependency graph for hesspix.c:



Functions

- int **main** ()

8.14.1 Detailed Description

Generate H.E.S.S.
camera pixel list for 768 pixels camera.

Author

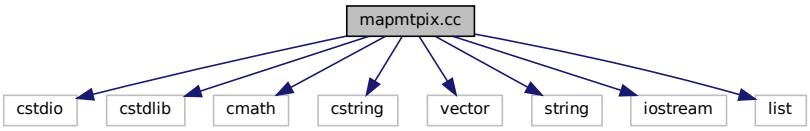
Konrad Bernloehr

8.15 mapmtpix.cc File Reference

Generate pixel list for cameras made of multi-anode PMTs.

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <cstring>
#include <vector>
#include <string>
#include <iostream>
#include <list>
```

Include dependency graph for mapmtpix.cc:



Data Structures

- class Trigger
- class Pixel
- class Module

Macros

- #define **USE_H8500** 1

Functions

- double **sq** (double x)
- int **main** (int argc, char **argv)

8.15.1 Detailed Description

Generate pixel list for cameras made of multi-anode PMTs.

Author

Konrad Bernloehr

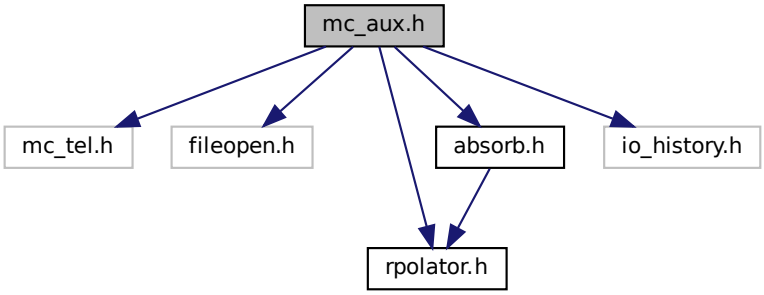
8.16 mc_aux.h File Reference

Internal data structures and prototypes for telescope simulation.

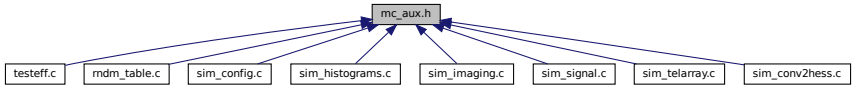
```
#include "mc_tel.h"
#include "fileopen.h"
#include "rpolator.h"
#include "absorb.h"
```

```
#include "io_history.h"
```

Include dependency graph for mc_aux.h:



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



Data Structures

- struct mc_tel_options

  Telescope-specific options passed through to low-level functions.
- struct mc_options

  Options of the simulation passed through to low-level functions.
- struct mc_run

  Per-run information on what was actually simulated.
- struct simulated_shower_parameters

  Basic parameters of the simulated showers: (beware: lengths in [m] and energies in [TeV] - as in HEGRA analysis)
- struct reconstructed

  Reconstructed parameters of the showers: (beware: lengths in [m] and energies in [TeV] - as in HEGRA analysis)
- struct channel_calibration

  Calibration parameters for a PM and its electronics channel(s):
- struct pm_and_fadc_channel

  Electronics for one PM (including accumulated signals):
- struct trigger_group

  A collection of pixels which can form a telescope trigger condition.
- struct camera_electronics

  Electronics of a whole camera and its parameters:
- struct PM_GridList

  Structure for rectangular grid used for accelerated pixel search.
- struct PM_Grid
- struct Pix_Type

  List of pixel types.
- struct PM_List

List of photomultipliers for custom camera.

- struct camera_image_plot_param

  Some parameters for image plots.
- struct pm_camera

  Geometric and optical parameters of a camera.
- struct transform_struct

  Coordinate shift and rotation:
- struct transform_off_struct

  Transformation with non-intersecting rotation axes:
- struct mirror_struct

  Parameters for a telescope mirror:
- struct vector_xy_struct

  Vector (2-D) with starting point and direction.
- struct convex_polygon

  A convex polygon with no positions showing up twice, except first=last, and all turns being right turns or all turns being left turns (no zero or 180 deg turns).
- struct mirror_segmentation

  Segmentation of primary and secondary in dual mirror optics.
- struct mirror_segment_map
- struct telescope_optics

  Parameters for the optics of a telescope (except the camera):
- struct telescope_array_group

  A collection of telescopes which can form an array trigger condition.
- struct mc_particles

  Keep a copy of particles at ground level, in either 'bunch' or 'bunch3d' format, as-is.
- struct telescope_array

  Parameters relevant for the whole array of telescopes: Actually some refer to ONE array (usually the one presently simulated), some to ANY array (the layout), and some to ALL arrays (random shifts).

Macros

- #define WITH_BYPASS_OPTICS 1 /∗ Only symbolic now, no longer tested in code ∗/

  ---------— Always enable the option to bypass the optics ----------—
- #define PULSE_ANALYSIS 1

  ----------— Definitions for event analysis and output ----------—
- #define **WITH_PULSE_ANALYSIS** 1
- #define **ANALYSE_SHOWER** 1
- #define OVERSAMPLING 40

  ----------— Definitions of various array sizes and limits ---------—
- #define MAX_SHAPE_LENGTH 500

  Maximum length of shape in FADC bins for each offset.
- #define MAX_TRG_TYPES 4

  How many trigger types can be handled.
- #define WITH_MAJORITY_TRG 1

  You better don't try to disable this one.
- #define **WITH_ANALOGSUM_TRG** 1
- #define **WITH_DIGITALSUM_TRG** 1
- #define MAX_SEGMENTS 128

  CTA_ULTRA implies CTA.
- #define MAX_ARRAY_GROUPS 16

  Maximum number of array subsets which can form a system trigger.
- #define **MAX_PIXELS** 1024

- #define MAX_FADC_BINS 128

  Maximum number of FADC time bins.
- #define MAX_TRIG_BINS 128

  Maximum number of trigger time bins (width as for FADC)
- #define MAX_ARRAY 100

  The largest no.
- #define MAX_PIX_TYPES 10

  Maximum number of different pixel types.
- #define MAX_INPUTS_IN_TRG MAX_PIXELS

  Maximum number of channels for each trigger group.
- #define MAX_NEIGHBOURS 8

  Maximum number of neighbouring pixels.
- #define MAX_LASER_LEVELS 10

  Maximum number of different laser/LED amplitudes.
- #define MAX_NPAR_MIRR 20

  Maximum number of parameters describing a general mirror shape.
- #define MAX_TRG_TYPES 4

  How many trigger types can be handled.
- #define PHI_ZONES 10

  Compile-time override of the most relevant limits:
- #define **RAD_ZONES** 12
- #define PROFILE_PIXELS 5

  No.
- #define MAX_BUNCHES 5000000

  The maximum number of allowed bunches per telescope.
- #define MAX_PHOTOELECTRONS 2500000

  The maximum number of allowed photo-electrons per telescope.
- #define MAX_PIXEL_PHOTOELECTRONS 25000

  The maximum per pixel above which we can ignore more p.e.s in the same pixel.
- #define MAX_PARTICLES 1000

  Not actually used so far, effectively limited by MAX_BUNCHES.
- #define MAX_TELTRANS 6

  Function coefficients for telescope transmission.
- #define MAX_LAMBDA 1000

  In nanometer steps.
- #define MAX_LONGI 1071

  Largest size of longitudinal distribution we can handle.
- #define **REVISED_NSB_HANDLING**
- #define **CALIB_EVENT_LOOP_BY_TEL** 1
- #define Nair(hkm) (1.+0.0002814∗exp(-0.0947982∗(hkm)-0.00134614∗(hkm)∗(hkm)))

  Refraction index of air as a function of height in km (0km<=h<=8km), approximation.
- #define **DISC_BITS_PER_BIN** 8

Typedefs

- typedef unsigned char fadc_data_t

  Sufficient for an 8-bit FADC.
- typedef unsigned char disc_data_t

  Bit field for discriminator output.
- typedef int **dsum_t**
- typedef struct mirror_struct **Mirror**
- typedef struct vector_xy_struct **VectorXY**
- typedef struct convex_polygon **ConvexPolygon**

Functions

- double * make_random_table (double *x, double *y, int n, int ni, int nr)

  Make very fast lookup table for rather flat random number distribution.
- double random_from_table (double *xr, int nr, double randnr)

  Very fast random number table lookup.
- int make_random_ipol_table (double *x, double *y, int n, double **xr, double **yr)

  Make fast lookup table for rather any number distribution.
- double random_from_ipol_table (double *xr, double *yr, int n, double randnr)

  Fast random number table lookup.
- int read_spe (char *fname, double **xspe_prompt, double **yspe_prompt, double **xspe_bkgrnd, double **yspe_bkgrnd, int *nspe_prompt, int *nspe_bkgrnd, int afterpulse_alt, double *norm_fact)

  Read the single photo-electron response distribution of PMs.
- int read_pulse_shape (int num_gains, char *fname_fadc, char *fname_disc, double binwidth, double *fshape, double *fshape_lg, int *flength, double *farea, double *farea_lg, double *bshape, double *bshape_lg, int *blength, double *barea, double *barea_lg, double *dshape, int *dlength, double *darea)

  Read the shapes of pulses at the FADCs and at the discriminator inputs.
- double delay_signals (struct camera_electronics *el)

  Common delay added to all photon arrival times.
- void laser_calib_eval (struct camera_electronics *el)

  Evaluate calibration parameters which would normally be obtained from measurements with lasers or LEDs (incl.
- void init_pm_electronics (struct pm_and_fadc_channel *ch, struct camera_electronics *el, double fadc_amp, double fadc_amp_lg, double disc_amp, double qe_var, double g_var, double hglg_var, int flatfielding, double transit, double transit1, double v1frac, double voltage_var, double gain_index, double transit_error, double transit_calib_error, double transit_comp_step, double transit_comp_err, double pedestal, double pedestal_dev, double pedestal_var, double pedestal_err, double pedestal_lg, double pedestal_dev_lg, double pedestal_var_lg, double pedestal_err_lg, double sensitivity, double sens_var, double sensitivity_lg, double sens_var_lg, double background, double gain, double disc_threshold, double disc_var, double gate_length_bins, double gate_length_var_bins, double gate_delay_bins, double gate_delay_var_bins, double disc_sigsum_bins, double disc_sigsum_var_bins, double trigger_current_limit, int fadc_per_channel)

  Photo-multiplier and electronics behaviour is set up.
- void create_pm_signals (int *pe_counts, int *istart, double *atimes, double *aamp, double t0, struct camera_electronics *el)

  Create the photomultiplier signals in FADC and at discriminator for one telescope.
- void pulse_shape_analysis (struct camera_electronics *el)

  Timing analysis of the digitized pulse shape (in reality by FPGA or signal processor).
- void telescope_trigger (struct pm_camera *cam, struct camera_electronics *el)

  Find out if (and when) a single telescope is triggered.
- void array_trigger (struct telescope_array *array, struct camera_electronics *elec, int iarray)

  Find out if (and when) the array of telescopes is triggered.
- void sum_adc_bins (struct telescope_array *array)

  Calculate (F)ADC sums in a time interval common to all channels.
- int read_shape (const char *fname, double *shape, int maxlen, double timestep, double scale, int *shape_len, int *toffsteps)

  Read a pulse shape given in arbitrary (but increasing) time steps and rebin it to the desired steps (typically ADC clock cycle or a multiple thereof.
- void **dhsort** (double *dnum, int nel)
- int solve_quadratic_equation (double a, double b, double c, double *x1, double *x2)

  *Solve a quadratic equation a*x**2 + b*x + c = 0.*
- int reflect_on_spherical_mirror (double r, double mrad, double arnd, double *p, double *d, int type, double phi, double *angle)

  Reflect a photon on a spherical mirror.
- int reflect_on_parabolic_mirror (double r, double mrad, double arnd, double *p, double *d, int type, double phi, double *angle)

Reflect a photon on a parabolic mirror.

- int refract_in_fresnel_lens (double r, double mrad, double arnd, double *p, double *d, int type, double n, double *angle)

  Refract a photon in a thin fresnel lens (using framework from reflection in parabolic mirror)
- int **reflect_on_polynomial_mirror** (double r, double mrad, double arnd, double *p, double *d, int type, double phi, int npar, double *params, double hole_diam, double *angle)
- double poly_eval_even (double x, int npar, const double *params)

  Evaluate an even polynomial.
- double poly_deriv_even (double x, int npar, const double *params)

  Evaluate the derivative of an even polynomial.
- void make_trafo (double alpha_ang, double beta_ang, double gamma_ang, double x, double y, double z, struct transform_struct *trans)

  Create rotation matrix and translation vector.
- void **make_trafo_off** (const struct transform_struct *t1, const struct transform_struct *t2, const double *o3, struct transform_off_struct *ts)
- void transform (double *p, double *d, const struct transform_struct *ts, int mode)

  Rotation plus translation transformation (forw.
- void transform_off (double *p, double *d, const struct transform_off_struct *ts, int mode)

  Transformation with extra offsets between rotation axes (forw.
- int tel_newdir (double phi, double theta, double tel_rand, struct telescope_optics *optics)

  Set up new transformations between ground and telescope.
- int tel_setup_primary (struct telescope_optics *optics, double phi, double theta)

  Set up all the transformations between ground, telescope, mirror, and camera reference frames, for telescopes with the camera in the primary focus (mirror_class 0 or 1).
- int tel_setup_secondary (struct telescope_optics *optics, double phi, double theta)

  Set up all the transformations between ground, telescope, mirror, and camera reference frames, for telescopes with the camera in the primary focus (mirror_class 0 or 1).
- int tel_mirror_grid_setup (struct telescope_optics *optics)

  Setup mirror 'grid' for faster raytracing (in case of many mirrors)
- int pm_grid_setup (struct pm_camera *camera)

  Setup of a grid on which PM hits can be searched faster.
- int pm_grid_search (struct pm_camera *camera, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to_camera)

  Faster search of PM hits on a grid over the camera front.
- int camera_setup_inclined_pixels (struct pm_camera *camera, struct telescope_optics *optics)

  Positions (in z) and optionally alignment of pixels set up to follow focal surface.
- int trace_photon_in_segmented (struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to_camera, int *mirror, double *releff, double rlambda, double distance)

  Trace a photon through the whole telescope to the camera (if hit).
- int trace_photon_in_paraboloid (struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to_camera, int *mirror, double *releff, double rlambda, double distance)

  Trace a photon through the whole telescope to the camera (if hit).
- int trace_photon_in_fresnel (struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to_camera, int *mirror, double *releff, double rlambda, double distance)

  Trace a photon through the whole telescope to the camera (if hit).
- int trace_photon_with_secondary (struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to_camera, int *mirror, double *releff, double rlambda, double distance, int bypass)

  Trace a photon through the whole telescope to the camera (if hit).
- int read_table (const char *fname, int maxrow, double *col1, double *col2)

Low-level reading of 2-column data tables up to given number of data rows.

- int assign_camera_pixels (struct pm_camera *cam)

  Setup the camera layout and the constants needed for fast reference between photon 'impact' position and pixel number.
- const char * **geo_text** (int geo)
- int **assign_flexible_camera** (struct pm_camera *cam, struct camera_electronics *el, char *fname, double scale_factor, int default_min_pixels)
- int camera_hit (struct pm_camera *cam, struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to_camera)

  Find out which (if any) pixel of the camera is hit.
- int cathode_hit (struct pm_camera *cam, int ipix, double x, double y, double xs, double ys, int iwl, double *releff)

  Find out if photocathode is hit.
- void offset_in_camera (double ref_azimuth, double ref_altitude, struct telescope_optics *optics, double *xoff, double *yoff)

  Calculate offset of a reference sky position in the camera plane.
- int read_mirror_segments (const char *fname, struct mirror_segmentation *set_sets, size_t max_sets)

  Read a configuration file with segmentation for either primary or secondary of a dual mirror telescope.
- ConvexPolygon * **make_convex_polygon** (size_t n, double *xp, double *yp)
- int **inside_convex_polygon** (double x, double y, ConvexPolygon *cp)
- int baffle_intersection (double *photon_pos, double *photon_dir, double *baffle, int fwd_only)

  Check if a photon path intersects with a baffle which can be either a piece of cylinder or a cone (and may have a finite wall thickness).
- int **nearest_int** (double d)
- void config_setup_imaging (void)

  Interface used for configuration of imaging and electronics.
- void config_setup_atm_transmission (void)

  Interface required for atmospheric transmission configuration.
- int init_setup (int argc, char **argv, struct telescope_array *array, struct telescope_optics *tel_optics, struct pm_camera *camera, struct camera_electronics *electronics, struct mc_options *options)

  Initialize the whole setup and the different pieces of the simulation.
- int convergence_correction (struct telescope_array *array)

  Correct the telescope pointing for convergent viewing.
- int randomize_viewing_direction (struct telescope_array *array)

  Changing viewing directions from shower to shower.
- MetaParamList * metapar_deliver (int itel)

  Deliver the locally maintained set of metaparameters to the external code.
- void **init_fill_mc_norm_pulse** (int ntel, struct camera_electronics *elec)
- void convert_check_setup (void)

  One-time check if main array size limits in sim_telarray are compatible with producing output data via the hessio library.
- void **convert_config** (struct telescope_array *array, int format, char *output_fname, int replace)
- void convert_runheader (struct telescope_array *array, int run)

  Convert run header.
- void convert_input_lines (struct linked_string *list)

  Convert the CORSIKA input configuration lines to the output file.
- void convert_mc_event (struct telescope_array *array)

  Convert MC event data (per shower usage).
- void convert_mc_photons (int shower, int iarray, int itel, double photons, struct bunch *bunches, int nbunches)

  Save CORSIKA photon bunches into the output data file.
- void convert_mc_photons3d (int shower, int iarray, int itel, double photons, struct bunch3d *bunches3d, int nbunches)

Save CORSIKA photon bunches of 3D type into the output data file.

- void convert_mc_pe_list (int shower, int array, int tel, int npe, int flags, int pixels, int *pe_counts, int *tstart, double *t, double *a, int *phot_counts)

  Save list of all 'detected' photo-electrons from Cherenkov photons.
- void convert_calib_pe_list (int type, int array, int tel, int npe, int flags, int pixels, int *pe_counts, int *tstart, double *t, double *a, int *phot_counts)

  Save list of all 'detected' photo-electrons from internal calibration event photons.
- void **convert_basic_data** (struct telescope_array *array, int run, int event, int write_all_pixels, int only_if_↩ triggered)
- void **convert_calibdata** (struct telescope_array *array, int run, int event, int type)
- void **convert_analyse_shower** (struct telescope_array *array)
- void convert_corsika_particles (struct telescope_array *array, int jarray, int itel, double photons, struct bunch *bunches, int nbunches)

  If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.
- void convert_corsika_particles3d (struct telescope_array *array, int jarray, int itel, double photons, struct bunch3d *bunches3d, int nbunches)

  If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.
- void convert_eventdata (struct telescope_array *array, int run, int event, int write_all_pixels, int only_if_↩ triggered)

  Convert data from internal structures to structures as used for the HESS telescope system, optionally analyse the simulated showers (by the same methods as used for the measured data) and write the converted raw data to the output file.
- void convert_runend (struct telescope_array *array)

  End of run in conversion to HESS format.
- void convert_finish (struct telescope_array *array)

  Finished with all conversions, clean up and close output file.
- int init_histograms (struct telescope_array *array, struct pm_camera *camera, struct camera_electronics *electronics)

  Initialize (book) all wanted histograms.
- int **fill_histograms** (struct telescope_array *array, int iarray, struct camera_electronics *electronics, double energy, double wt, double alt, double az)
- double line_point_distance (double x1, double y1a, double z1, double cx, double cy, double cz, double x, double y, double z)

  Distance between a straight line and a point in space.
- double **find_max_pos** (double *y, int n)
- double Square (double x)

  Function returning the square of its argument (instead of a macro).
- void sim_calib_events (int events, int laser, int open, struct telescope_array *array, double *atimes, double *aamp, long len_atimes)

  Simulate calibration events in the Cherenkov telescopes.
- double rhof_ (double *height)

  Density of atmosphere [g/cm^3] at given height a.s.l.
- double thick_ (double *height)

  'Thickness' (column density) of atmosphere [g/cm^2] from space down to given height a.s.l.
- double refid_ (double *height)

  Index of refraction at given height a.s.l.
- double refim1_ (double *height)

  Index of refraction minus one at given height a.s.l.
- double heigh_ (double *thick)

  Inverse of thick() function.
- const char * get_simtel_prog (void)

Retrieve the name under which sim_telarray was apparently started.

- const char * get_simtel_prog_path (void)

  Retrieve the program path under which sim_telarray was apparently started.

8.16.1 Detailed Description

Internal data structures and prototypes for telescope simulation.
This file contains definitions for internal data structures used in the telescope simulation, for dimensions of various things, and prototypes of relevant functions.

Author

Konrad Bernloehr

Date

1997 ... 2023

8.16.2 Macro Definition Documentation

8.16.2.1 MAX_ARRAY

```
#define MAX_ARRAY 100
```

The largest no.
of arrays to be handled (i.e. how often a CORSIKA shower is re-used)
Definition at line 694 of file mc_aux.h.

8.16.2.2 MAX_BUNCHES

```
#define MAX_BUNCHES 5000000
```

The maximum number of allowed bunches per telescope.
Override with CORSIKA_MAX_BUNCHES env. var.
Definition at line 742 of file mc_aux.h.

8.16.2.3 MAX_LAMBDA

```
#define MAX_LAMBDA 1000
```

In nanometer steps.
Lots of code still relying on hard-coded 1000
Definition at line 763 of file mc_aux.h.

8.16.2.4 MAX_PHOTOELECTRONS

```
#define MAX_PHOTOELECTRONS 2500000
```

The maximum number of allowed photo-electrons per telescope.
Override with MAX_PHOTOELECTRONS env. var.
Definition at line 746 of file mc_aux.h.

8.16.2.5 MAX_PIXEL_PHOTOELECTRONS

```
#define MAX_PIXEL_PHOTOELECTRONS 25000
```

The maximum per pixel above which we can ignore more p.e.s in the same pixel.
Override with MAX_PIXEL_PHOTOELECTRONS env. var.
Definition at line 750 of file mc_aux.h.

8.16.2.6 MAX_SEGMENTS

```
#define MAX_SEGMENTS 128
```

CTA_ULTRA implies CTA.
CTA_MAX also implies CTA
Definition at line 332 of file mc_aux.h.

8.16.2.7 MAX_TRG_TYPES [1/2]

```
#define MAX_TRG_TYPES 4
```

How many trigger types can be handled.
Support up to three or four different trigger types.
Definition at line 709 of file mc_aux.h.

8.16.2.8 MAX_TRG_TYPES [2/2]

```
#define MAX_TRG_TYPES 4
```

How many trigger types can be handled.
Support up to three or four different trigger types.
Definition at line 709 of file mc_aux.h.

8.16.2.9 OVERSAMPLING

```
#define OVERSAMPLING 40
```

---------- Definitions of various array sizes and limits ----------
Sampling of signal shape at different phase offsets.
Definition at line 105 of file mc_aux.h.

8.16.2.10 PHI_ZONES

```
#define PHI_ZONES 10
```

Compile-time override of the most relevant limits:
Polar grid for mirror tile lookup
Definition at line 731 of file mc_aux.h.

8.16.2.11 PROFILE_PIXELS

```
#define PROFILE_PIXELS 5
```

No.
of pixels to be used for time profile
Definition at line 737 of file mc_aux.h.

8.16.3 Function Documentation

8.16.3.1 array_trigger()

```
void array_trigger (
            struct telescope_array * array,
            struct camera_electronics * elec,
            int iarray )
```

Find out if (and when) the array of telescopes is triggered.
There may be multiple conditions for an array trigger and if any of them is met the array is declared triggered. The data recorded would include any extra triggered telescopes even if not part of an array trigger condition - with two

exceptions: a) telescopes only listed in 'hardware stereo' conditions get their 'telescope_triggered' flag reset if the hardware stereo group did not trigger; b) alternate mono triggers from random or muon-ring selections would ask data from other telescopes to be suppressed if the alternate mono condition is the only accepted trigger. This function sets values in the provided telescope_array struct.

Parameters

array	Pointer to the data for the entire array of telescopes.
elec	Pointer to the camera electronics data for all telescopes.
iarray	Which of multiple randomly offset arrays are we testing for?

Definition at line 3771 of file sim_signal.c.
References camera_electronics::central_time.

8.16.3.2 baffle_intersection()

```
int baffle_intersection (
            double * photon_pos,
            double * photon_dir,
            double * baffle,
            int fwd_only )
```

Check if a photon path intersects with a baffle which can be either a piece of cylinder or a cone (and may have a finite wall thickness).
Note that this intersection does not try to find the first intersection as in shortest path to possible intersection points but only if there is any intersection at all.

Parameters

photon_pos	Initial position of photon (x,y,z) in optics system. If there is an intersection, it gets updated with the identified intersection position.
photon_dir	Direction cosines of photon (cx,cy,cz) in optics system.
baffle	Baffle parameters (z1,z2,r1,dr,r2,[c]).
fwd_only	If this is non-zero only forward intersections are counted.

Returns

0 (no intersection), 1 (intersecting with baffle)

Definition at line 4780 of file sim_imaging.c.
References line_ztube_intersection(), and solve_quadratic_equation().

8.16.3.3 camera_hit()

```
int camera_hit (
            struct pm_camera * cam,
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * xcam,
            double * ycam,
            double * sxcam,
            double * sycam,
            double * time_to_camera )
```

Find out which (if any) pixel of the camera is hit.

Parameters

cam	Camera configuration
photon_pos	Position of ray intersection with focal surface
photon_dir	Direction of incoming ray in camera system
xcam	On entry: x position in camera system; on exit: in pixel system
ycam	On entry: y position in camera system; on exit: in pixel system
sxcam	On entry: x slope in camera system; on exit: in pixel system
sycam	On entry: y slope in camera system; on exit: in pixel system
time_to_camera	On entry: time to focal surface; on exit: time to pixel

Returns

-1 (no pixel hit) or pixel ID

Definition at line 7711 of file sim_imaging.c.
References pm_camera::camera_pixel_assignment_initialized, camera_setup_inclined_pixels(), pm_camera::camera_type, telescope_optics::focal_surface_parameters, telescope_optics::npar_focal, pm_grid_search(), poly_deriv_even(), and pm_camera::telescope.

8.16.3.4 cathode_hit()

```
int cathode_hit (
            struct pm_camera * cam,
            int ipix,
            double x,
            double y,
            double xs,
            double ys,
            int iwl,
            double * releff )
```

Find out if photocathode is hit.
Find out if the photocathode is hit directly or after reflection on the lightguide, or if the photon hits a gap between pixels.

Parameters

cam	Camera configuration
ipix	Pixel ID
x	X position of hit in pixel entrance plane, w.r.t. pixel center [cm]
y	Y position of hit in pixel entrance plane, w.r.t. pixel center [cm]
sx	X slope of incident ray in pixel frame (tan)
sy	Y slope of incident ray in pixel frame (tan)
iwl	Wavelength index [nm]
releff	Relative efficiency along the photon path, w.r.t. the efficiency applied before ray-tracing; gets multiplied here with factor related to incidence on pixel (and wavelength, if applicable).

Returns

1 - photocathode (directly) hit, 2 - lightguide hit (configuration case without efficiency table(s)) 3 - pixel lightguide efficiency table applied -1 - gap at edge of lightguide hit, -2 - no hit, -3 - internal absorption.

Definition at line 7790 of file sim_imaging.c.
References Pix_Type::angle_table_size, pm_camera::camera_type, Pix_Type::cathode_shape, Pix_Type::funnel_angle_table, Pix_Type::funnel_wl_table, pm_camera::lightguide_reflectivity, M_PI, MAX_LAMBDA, PM_List::pix

_type, Pix_Type::pixel_cathode_r2, pm_camera::pixel_cathode_r_squared, Pix_Type::pixel_depth, pm_camera::pixel_depth, Pix_Type::pixel_shape, pm_camera::pixtype, pm_camera::pm_list, Pix_Type::r2, RandFlat(), Pix_Type::reflectivity, pm_camera::telescope, and Pix_Type::transparency.

8.16.3.5 convergence_correction()

```
int convergence_correction (
            struct telescope_array * array )
```

Correct the telescope pointing for convergent viewing.
The correction is based on the assumption that the telescopes have all the same basic pointing direction. If the telescopes have separate pointing directions, with corrections for convergent viewing already taken into account there, then no additional convergence correction should be configured.
Definition at line 5745 of file sim_config.c.
References telescope_array::altitude, mc_run::atmosphere, telescope_array::azimuth, telescope_array::camera, telescope_array::conv_div_opt, telescope_optics::convergent_depth, telescope_optics::convergent_height, telescope_array::electronics, camera_electronics::itel, pm_camera::itel, telescope_optics::itel, M_PI, telescope_array::mean_convergent_depth, telescope_array::ntel, telescope_array::optics, telescope_array::xtel, telescope_array::ytel, and telescope_array::ztel.

8.16.3.6 convert_corsika_particles()

```
void convert_corsika_particles (
            struct telescope_array * array,
            int jarray,
            int itel,
            double photons,
            struct bunch * bunches,
            int nbunches )
```

If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.

Parameters

array	array number (usually 999 for particles)
tel	telescope number (usually 999 for particles)
photons	sum of photons (and fractions) in this device
bunches3d	list of 3D photon bunches
nbunches	number of elements in bunch list

Definition at line 4362 of file sim_conv2hess.c.
References telescope_array::particles.

8.16.3.7 convert_corsika_particles3d()

```
void convert_corsika_particles3d (
            struct telescope_array * array,
            int jarray,
            int itel,
            double photons,
            struct bunch3d * bunches3d,
            int nbunches )
```

If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.

Parameters

array	array number (usually 999 for particles)
tel	telescope number (usually 999 for particles)
photons	sum of photons (and fractions) in this device
bunches3d	list of 3D photon bunches
nbunches	number of elements in bunch list

Definition at line 4409 of file sim_conv2hess.c.
References telescope_array::particles.

8.16.3.8 convert_eventdata()

```
void convert_eventdata (
            struct telescope_array * array,
            int run,
            int event,
            int write_all_pixels,
            int only_if_triggered )
```

Convert data from internal structures to structures as used for the HESS telescope system, optionally analyse the simulated showers (by the same methods as used for the measured data) and write the converted raw data to the output file.

Parameters

array	Structure tree of array and telescope specific data
run	Run number
event	Event number (100∗event + array number)
write_all_pixels	1 to write even pixels with no significant signal, 0 otherwise
only_if_triggered	1 to write only data of triggered telescopes, 0 to write data of all telescopes.

Returns

(none)

Definition at line 4461 of file sim_conv2hess.c.

8.16.3.9 convert_finish()

```
void convert_finish (
            struct telescope_array * array )
```

Finished with all conversions, clean up and close output file.

Parameters

array	Structure tree of array and telescope specific data.

Returns

(none)

Definition at line 5211 of file sim_conv2hess.c.

8.16.3.10 convert_input_lines()

```
void convert_input_lines (
            struct linked_string * list )
```

Convert the CORSIKA input configuration lines to the output file.
This is done here by simply rewriting it in the same format.
Definition at line 2432 of file sim_conv2hess.c.

8.16.3.11 convert_mc_event()

```
void convert_mc_event (
            struct telescope_array * array )
```

Convert MC event data (per shower usage).
If once-per-run material like run headers and configuration has not been written yet, writing of that material is triggered first. If the shower-specific data is not written yet, this is also accomplished before the shower-usage specific data is eventually written.
Definition at line 2475 of file sim_conv2hess.c.

8.16.3.12 convert_runend()

```
void convert_runend (
            struct telescope_array * array )
```

End of run in conversion to HESS format.
Write end-of-run statistics.

Parameters

array	Structure tree of array and telescope specific data.

Returns

(none)

Definition at line 5193 of file sim_conv2hess.c.
References hsdata, and telescope_array::ntel.

8.16.3.13 convert_runheader()

```
void convert_runheader (
            struct telescope_array * array,
            int run )
```

Convert run header.
Note that configuration may not be set yet. Output is, therefore, delayed until the first event.

Parameters

ntel	Number of telescopes in the array.
optics	Structure with optics-specific data.
run	Run number.

Returns

(none)

Definition at line 2193 of file sim_conv2hess.c.
References telescope_array::altitude, telescope_array::azimuth, telescope_array::conv_div_opt, mc_run::corsika_version, mc_run::e_max, mc_run::e_min, mc_run::height, hsdata, mc_options::ignore_mcdata, ignore_mcdata, mc_options::image_fname, M_PI, telescope_array::mean_convergent_depth, telescope_array::min_tel_trigger, telescope_array::ntel, mc_run::num_arrays, mc_run::num_showers, telescope_array::optics, telescope_array::options, mc_run::phi_max, mc_run::phi_min, mc_run::radius, mc_run::run_start, mc_run::slope, telescope_optics::telescope, telescope_array::telescope_ignore, mc_run::theta_max, mc_run::theta_min, mc_run::viewcone_max, mc_run::viewcone_min, telescope_array::xtel, telescope_array::ytel, and telescope_array::ztel.

8.16.3.14 create_pm_signals()

```
void create_pm_signals (
            int * pe_counts,
            int * istart,
            double * atimes,
            double * aamp,
            double t0,
            struct camera_electronics * el )
```

Create the photomultiplier signals in FADC and at discriminator for one telescope.

Parameters

pe_counts	The numbers of Cherenkov (or laser/LED) photo-electrons in each pixel (no NSB).
istart	The offset in the global p.e. times list where p.e.s of each pixel are starting (cumulative pe_counts of earlier pixels).
atimes	The list of times of all signal p.e.s in the camera (arrival time at the photo sensor). In [ns] since the time the photons would have crossed the CORSIKA observation level (with t=0 when the primary particle would have crossed that level).
aamp	NULL or array like atimes to keep the random p.e. amplitudes.
t0	The median time of all p.e.s minus a (20+x)% fraction of the readout window, so that atimes[i]-t0 is typically early in the common readout window.
el	Pointer to all the camera electronics details of the telescope.

Definition at line 1480 of file sim_signal.c.
References MAX_FADC_BINS, and pm_and_fadc_channel::signal.

8.16.3.15 delay_signals()

```
double delay_signals (
            struct camera_electronics * el )
```

Common delay added to all photon arrival times.
Add a common delay to all photons at one telescope because shower arrival is not correlated with the FADC clock. This function should be called exactly once per event.
Definition at line 1169 of file sim_signal.c.
References camera_electronics::fadc_bins, camera_electronics::fadc_delay, camera_electronics::fadc_per_channel, camera_electronics::interval, camera_electronics::phase_delay, camera_electronics::photon_delay, and RandFlat().

8.16.3.16 init_pm_electronics()

```
void init_pm_electronics (
            struct pm_and_fadc_channel * ch,
```

```
            struct camera_electronics * el,
            double fadc_amp,
            double fadc_amp_lg,
            double disc_amp,
            double qe_var,
            double g_var,
            double hglg_var,
            int flatfielding,
            double transit,
            double transit1,
            double v1frac,
            double voltage_var,
            double gain_index,
            double transit_error,
            double transit_calib_error,
            double transit_comp_step,
            double transit_comp_err,
            double pedestal,
            double pedestal_dev,
            double pedestal_var,
            double pedestal_err,
            double pedestal_lg,
            double pedestal_dev_lg,
            double pedestal_var_lg,
            double pedestal_err_lg,
            double sensitivity,
            double sens_var,
            double sensitivity_lg,
            double sens_var_lg,
            double background,
            double gain,
            double disc_threshold,
            double disc_var,
            double gate_length_bins,
            double gate_length_var_bins,
            double gate_delay_bins,
            double gate_delay_var_bins,
            double disc_sigsum_bins,
            double disc_sigsum_var_bins,
            double trigger_current_limit,
            int fadc_per_channel )
```

Photo-multiplier and electronics behaviour is set up.
Initialize constants needed for the simulation of the response of one channel (photo-multiplier, flash-ADC, discriminator, DC current)
Definition at line 702 of file sim_signal.c.

8.16.3.17 init_setup()

```
int init_setup (
            int argc,
            char ** argv,
            struct telescope_array * array,
            struct telescope_optics * tel_optics,
            struct pm_camera * camera,
            struct camera_electronics * electronics,
            struct mc_options * options )
```

Initialize the whole setup and the different pieces of the simulation.

Initialize the whole setup (default setup and telescope-specific setup), read in all the auxiliary data tables like quantum efficiencies and so on, and initialize all the constants needed for the simulation. Command-line arguments are passed to do_config().
Definition at line 2787 of file sim_config.c.
References imaging_setup::focus_offset, and imaging_setup::mirrors.

8.16.3.18 laser_calib_eval()

```
void laser_calib_eval (
            struct camera_electronics * el )
```

Evaluate calibration parameters which would normally be obtained from measurements with lasers or LEDs (incl. flatfielding).
Definition at line 1189 of file sim_signal.c.
Referenced by convert_initial_moni_calib().

8.16.3.19 line_point_distance()

```
double line_point_distance (
            double xp1,
            double yp1,
            double zp1,
            double cx,
            double cy,
            double cz,
            double x,
            double y,
            double z )
```

Distance between a straight line and a point in space.

Parameters

xp1,yp1,zp1	reference point on the line
cx,cy,cz	direction cosines of the line
x,y,z	point in space

Returns

distance

Definition at line 53 of file rec_tools.c.

8.16.3.20 make_random_ipol_table()

```
int make_random_ipol_table (
            double * x,
            double * y,
            int n,
            double ** xr,
            double ** yr )
```

Make fast lookup table for rather any number distribution.
The following procedures can be used for relatively fast production of random numbers according to a given table of (not to be normalized) probability values. Tails of a distribution should be sufficiently well reproduced within the limits imposed a) by the chosen generator of flat random numbers (really double, i.e. 48-bit abscissa, or effectively float, i.e. 24-bit abscissa) and b) by the assumption that the probability is constant within each bin of the input table. The bin limits are calculated with the assumption of equidistant bins where the abscissa values of the input table give the middle of each bin. This must hold at least for the two lowest and the two highest bins.

Parameters

x	X coordinates of tabulated 'curve'
y	Y coordinates of tabulated 'curve'
n	number of points in provided table
xr	Pointer to returned X table (or NULL)
yr	Pointer to returned Y table (or NULL)

Returns

0 (o.k.), -1 (error)

Definition at line 185 of file rndm_table.c.
Referenced by read_spe().

8.16.3.21 make_random_table()

```
double* make_random_table (
            double * x,
            double * y,
            int n,
            int ni,
            int nr )
```

Make very fast lookup table for rather flat random number distribution.
To have a very fast production of random numbers according to a given table of (not to be normalized) probability table, a corresponding table of abscissa bins with equal content is calculated. A flat random number in [0:1] then tells in which bin and where in this bin the wanted random number has to be. This procedure assumes that the probability varies rather modestly. With nr=10000, for example, it would be impossible to reproduce the tail of a distribution where the probability falls to well below one part in 10000 of the peak probability. Before applying this procedure, check which ni and nr values you need to reproduce your give table with sufficient accuracy.
The bin limits are calculated with the assumption of equidistant bins where the abscissa values of the input table give the middle of each bin. This must hold at least for the two lowest and the two highest bins.

Parameters

x	X coordinates of tabulated 'curve'
y	Y coordinates of tabulated 'curve'
n	number of points in provided table
ni	number of points used in intermediate table
nr	number of points (minus 1) in final lookup table

Returns

pointer to final lookup table (or NULL in case of error)

Definition at line 72 of file rndm_table.c.
References rpol().
Referenced by read_spe().

8.16.3.22 make_trafo()

```
void make_trafo (
            double alpha_ang,
            double beta_ang,
            double gamma_ang,
```

```
            double x,
            double y,
            double z,
            struct transform_struct * trans )
```

Create rotation matrix and translation vector.

Parameters

alpha_ang	Rotation angle for first rotation, around z axis (e.g. azimuth) [radians].
beta_ang	Rotation angle for second rotation, around y' axis (e.g. zenith angle) [radians].
gamma_ang	Rotation angle for third rotation, around z'' axis [radians].
x	X coordinate of center of rotation.
y	Y coordinate of center of rotation.
z	Z coordinate of center of rotation.
trans	Transformation parameters filled for later forward and backward transformations.

Definition at line 1850 of file sim_imaging.c.
Referenced by tel_newdir(), and tel_setup_secondary().

8.16.3.23 metapar_deliver()

```
MetaParamList * metapar_deliver (
            int itel )
```

Deliver the locally maintained set of metaparameters to the external code.

Parameters

itel	The telescope index (0 ... MAX_TEL-1, or -1 for global parameters).

Returns

Pointer to the prepared metaparameter set or NULL.

Definition at line 1252 of file sim_config.c.

8.16.3.24 offset_in_camera()

```
void offset_in_camera (
            double ref_azimuth,
            double ref_altitude,
            struct telescope_optics * optics,
            double * xoff,
            double * yoff )
```

Calculate offset of a reference sky position in the camera plane.
Note: the returned offsets are in linear units.

Parameters

ref_azimuth	Azimuth of sky position [deg]
ref_altitude	Altitude of sky position [deg]
optics	Transformation settings for telescope optics.
xoff	Filled on return with linear X offset [cm].
yoff	Filled on return with linear y offset [cm].

Definition at line 7957 of file sim_imaging.c.
References telescope_optics::altitude, telescope_optics::azimuth, telescope_optics::focal_length, and M_PI.

8.16.3.25 pm_grid_search()

```
int pm_grid_search (
            struct pm_camera * camera,
            double * photon_pos,
            double * photon_dir,
            double * xcam,
            double * ycam,
            double * sxcam,
            double * sycam,
            double * time_to_camera )
```

Faster search of PM hits on a grid over the camera front.

Parameters

cam	Camera configuration
photon_pos	Position of ray intersection with focal surface
photon_dir	Direction of incoming ray in camera system
xcam	On entry: x position in camera system; on exit: in pixel system
ycam	On entry: y position in camera system; on exit: in pixel system
sxcam	On entry: x slope in camera system; on exit: in pixel system
sycam	On entry: y slope in camera system; on exit: in pixel system
time_to_camera	On entry: time to focal surface; on exit: time to pixel

Returns

-1 (no pixel hit) or pixel ID

Definition at line 7539 of file sim_imaging.c.
References PM_List::axx, PM_List::azz, pm_camera::camera_pixel_assignment_initialized, pm_camera::curved_surface, PM_Grid::dym1, PM_Grid::field, Pix_Type::half_size, PM_GridList::list, M_PI, PM_GridList::num_pm, PM_Grid::ny, PM_List::pix_type, Pix_Type::pixel_shape, pm_camera::pixels_parallel, pm_camera::pixtype, pm_camera::pm_grid, pm_camera::pm_list, Pix_Type::r2, PM_Grid::x_high, PM_Grid::y_high, and PM_List::z.
Referenced by camera_hit().

8.16.3.26 poly_deriv_even()

```
double poly_deriv_even (
            double x,
            int npar,
            const double * params )
```

Evaluate the derivative of an even polynomial.

x The position (here usually the offset from the optical axis)

at which the derivative of the polynomial should be evaluated.

npar The number of (usually non-zero) coefficients to use in the evaluation.

param The list of even polynomial coefficients.

Returns

fprim(x)=d/dx f(x)=params[1]∗2∗x+params[2]∗4∗x∗(x∗x)+...

Definition at line 1496 of file sim_imaging.c.
Referenced by camera_hit(), camera_setup_inclined_pixels(), and reflect_on_polynomial_mirror().

8.16.3.27 poly_eval_even()

```
double poly_eval_even (
            double x,
            int npar,
            const double * params )
```

Evaluate an even polynomial.

x The position (here usually the offset from the optical axis)

at which the polynomial should be evaluated.

npar The number of (usually non-zero) coefficients to use in the evaluation.

param The list of even polynomial coefficients.

Returns

f(x)=params[0]+params[1]∗(x∗x)+params[2]∗(x∗x)∗(x∗x)+...

Definition at line 1472 of file sim_imaging.c.
Referenced by camera_setup_inclined_pixels(), and reflect_on_polynomial_mirror().

8.16.3.28 pulse_shape_analysis()

```
void pulse_shape_analysis (
            struct camera_electronics * el )
```

Timing analysis of the digitized pulse shape (in reality by FPGA or signal processor).
Note that only the high-gain channel is processed if several gains are available.
Definition at line 2791 of file sim_signal.c.
References pm_and_fadc_channel::calib, camera_electronics::fadc_bins, camera_electronics::fadc_per_channel, camera_electronics::global_peak_pos, camera_electronics::groups_triggered, camera_electronics::interval, camera_electronics::long_event, camera_electronics::longsum_bins, camera_electronics::longsum_offset, MAX↩_FADC_BINS, camera_electronics::num_gains, camera_electronics::peak_frac, pm_and_fadc_channel::peak_pos, channel_calibration::pedestal, camera_electronics::pixels, camera_electronics::pulse_analysis, pm_and_fadc_↩channel::pulse_rise, pm_and_fadc_channel::pulse_sum_glob, pm_and_fadc_channel::pulse_sum_loc, pm_and↩_fadc_channel::pulse_t_over_thr, pm_and_fadc_channel::pulse_width, pm_and_fadc_channel::signal, camera_↩electronics::sum_after_peak, camera_electronics::sum_before_peak, camera_electronics::sum_bins, camera_↩electronics::sum_offset, camera_electronics::sum_start, camera_electronics::telescope, and camera_electronics↩::trigger_time.

8.16.3.29 random_from_ipol_table()

```
double random_from_ipol_table (
            double * xr,
            double * yr,
            int n,
            double randnr )
```

Fast random number table lookup.
Get a random number from a lookup table as created by make_random_ipol_table() for a tabulated probability distribution.

Parameters

xr	Table as returned from make_random_ipol_table()
yr	Table as returned from make_random_ipol_table()
n	as provided to make_random_ipol_table()
randnr	A uniform pseudorandom number in [0:1]

Returns

random number according to table

Definition at line 243 of file rndm_table.c.
References rpol().
Referenced by fill_traces(), and read_spe().

8.16.3.30 random_from_table()

```
double random_from_table (
            double * xr,
            int nr,
            double randnr )
```

Very fast random number table lookup.
Get a random number from a lookup table as created by make_random_table() for a tabulated probability distribution.
Note that regions with probabilities below 1/nr of the peak probability are not well represented.

Parameters

xr	The lookup table from make_random_table()
nr	The same nr as given to make_random_table()
randnr	A uniform pseudorandom number in [0:1]

Returns

random number according to table

Definition at line 141 of file rndm_table.c.
Referenced by fill_traces(), and read_spe().

8.16.3.31 randomize_viewing_direction()

```
int randomize_viewing_direction (
            struct telescope_array * array )
```

Changing viewing directions from shower to shower.
On a per shower / array basis the viewing direction of all telescopes is randomized in such a way that the actual viewing directions are within a ring around the basic viewing direction given.

Note: the pattern of stars in the cameras is left unchanged unless the preprocessor flag VARIABLE_SKY_↩ BACKGROUND is set.
Definition at line 6137 of file sim_config.c.

8.16.3.32 read_pulse_shape()

```
int read_pulse_shape (
            int num_gains,
            char * fname_fadc,
            char * fname_disc,
            double binwidth,
            double * fshape,
            double * fshape_lg,
            int * flength,
            double * farea,
            double * farea_lg,
            double * bshape,
            double * bshape_lg,
            int * blength,
            double * barea,
            double * barea_lg,
            double * dshape,
            int * dlength,
            double * darea )
```

Read the shapes of pulses at the FADCs and at the discriminator inputs.
There are two possible FADC pulse shape formats: binned and unbinned. Each of them has to include a low-gain pulse shape if compiled WITH_LOW_GAIN_CHANNEL. Binned and unbinned format are distinguished by the presence or absence of a comment '# T=...' with the corresponding bin step in nanoseconds. The unbinned format requires an initial first column for the time [ns] corresponding to each line. The low-gain pulse is always the last of two or three columns (in binned or unbinned format, respectively).
The signal at the discriminator or comparator input is always in unbinned format, i.e. with leading time column.

Parameters

fname_fadc	File name with pulse shapes for digital signal (one or two gains)
fname_disc	Input signal at discriminator or comparator.
binwidth	Time length of (F)ADC sampling interval [ns].
fshape	Memory address where to store high-gain FADC pulse shape.
fshape_lg	Memory address where to store low-gain FADC pulse shape.
flength	Write common length of FADC pulse shapes.
farea	Write pulse area [ns] of HG pulse at amplitude=1.
farea_lg	Write pulse area [ns] of LG pulse at amplitude=1.
bshape	Like fshape but for background (NSB) signals.
bshape_lg	Like fshape_lg but for background (NSB) signals.
blength	Like flength but for background (NSB) signals.
barea	Like farea but for background (NSB) signals.
barea_lg	Like farea_lg but for background (NSB) signals.
dshape	Memory address where to store pulse shape at discriminator/comparator input.
dlength	Write length of discr./comp. pulse shape.
darea	Write pulse area [ns] of discr./comp. pulse at amplitude=1.

Returns

0 (OK), -1 (error)

Definition at line 232 of file sim_signal.c.
References MAX_SHAPE_LENGTH, OVERSAMPLING, rpol_cspline(), rpol_linear(), and set_1d_cubic_params().

8.16.3.33 read_spe()

```
int read_spe (
            char * fname,
            double ** xspe_prompt,
            double ** yspe_prompt,
            double ** xspe_bkgrnd,
            double ** yspe_bkgrnd,
            int * nspe_prompt,
            int * nspe_bkgrnd,
            int afterpulse_alt,
            double * norm_fact )
```

Read the single photo-electron response distribution of PMs.

Parameters

fname	Name of the file from which to read.
xspe_prompt	Amplitude values (mean p.e. units) for prompt response.
yspe_prompt	Probability of prompt response at give amplitudes.
xspe_bkgrnd	Amplitude values (mean p.e. units) for background response.
yspe_bkgrnd	Probability of background response (including afterpulsing) at give amplitudes.
nspe_prompt	Number of data points for prompt response.
nspe_bkgrnd	Number of data points for background response.
afterpulse_alt	If non-zero, use prompt values also for background. Afterpulsing has to be added separately later then.
norm_fact	The mean amplitude per single p.e. actually generated (should be close to 1.0).

Returns

0 (OK), -1 (error)

Definition at line 77 of file sim_signal.c.
References make_random_ipol_table(), make_random_table(), random_from_ipol_table(), and random_from_↩ table().

8.16.3.34 read_table()

```
int read_table (
            const char * fname,
            int maxrow,
            double * col1,
            double * col2 )
```

Low-level reading of 2-column data tables up to given number of data rows.

Parameters

fname	Name of file to be opened.
maxrow	Maximum number of (non-empty, non-comment) rows of data to read.
col1	Array where values of column 1 are to be copied to.
col2	Array where values of column 2 are to be copied to.

Returns

Number of data rows read (usable values in col1 and col2) or -1 (error).

Definition at line 100 of file rpolator.c.
Referenced by read_shape().

8.16.3.35 refid_()

```
double refid_ (
            double * height )
```

Index of refraction at given height a.s.l.
[cm].
Definition at line 210 of file sim_telarray.c.
References refim1_().

8.16.3.36 refim1_()

```
double refim1_ (
            double * height )
```

Index of refraction minus one at given height a.s.l.
[cm].
Definition at line 196 of file sim_telarray.c.
Referenced by refid_().

8.16.3.37 reflect_on_parabolic_mirror()

```
int reflect_on_parabolic_mirror (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
            int type,
            double phi_m,
            double * angle )
```

Reflect a photon on a parabolic mirror.

Parameters

r	radius of central curvature of mirror surface (twice its focal length)
mrad	half of mirror diameter (in case of hexagonal or square mirrors measured flat to flat)
arnd	random error in reflection angle
p	(x,y,z) starting point on entry, position of reflection point on return
d	(cx,cy,cz) direction before reflection on entry, direction after reflection (if any) on return
type	mirror type (0: circular, 1,3: hexagonal, 2: square
phi_m	Azimuth angle of mirror (actually it should be the negative azimuth angle of the dish centre in the mirror frame but is for now the azimuth angle of the mirror in the dish frame – which can be slightly different for inclined mirrors but is considered good enough to avoid artificial shadowing).
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (mirror not hit), -2 (on backside)

Note: multiple reflections are not taken into account
Definition at line 1033 of file sim_imaging.c.
References M_PI, RandGauss(), and solve_quadratic_equation().

8.16.3.38 reflect_on_spherical_mirror()

```
int reflect_on_spherical_mirror (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
            int type,
            double phi_m,
            double * angle )
```

Reflect a photon on a spherical mirror.

Parameters

r	radius of curvature of mirror surface (twice its focal length)
mrad	half of mirror diameter (in case of hexagonal or square mirrors measured flat to flat)
arnd	random error in reflection angle
p	(x,y,z) starting point on entry, position of reflection point on return
d	(cx,cy,cz) direction before reflection on entry, direction after reflection (if any) on return
type	mirror shape type (0: circular, 1,3: hexagonal, 2: square)
phi_m	Azimuth angle of mirror (actually it should be the negative azimuth angle of the dish centre in the mirror frame but is for now the azimuth angle of the mirror in the dish frame – which can be slightly different for inclined mirrors but is considered good enough to avoid artificial shadowing).
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (mirror not hit), -2 (on backside)

Note: multiple reflections are not taken into account
Definition at line 791 of file sim_imaging.c.
References M_PI, RandGauss(), and solve_quadratic_equation().

8.16.3.39 refract_in_fresnel_lens()

```
int refract_in_fresnel_lens (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
            int type,
            double n,
            double * angle )
```

Refract a photon in a thin fresnel lens (using framework from reflection in parabolic mirror)

Parameters

r	Radius of central curvature of refracting lens surface (= flen ∗ (n-1) in ideal case).
mrad	Half of lens diameter (in case of hexagonal or square lens shape measured flat to flat)
arnd	random error in resulting angle
p	(x,y,z) starting point on entry, position of point on thin lens on return (z=0).
d	(cx,cy,cz) direction before lens on entry, direction after lens (if any) on return
type	lens shape type (0: circular, 1,3: hexagonal, 2: square
n	Index of refraction of lens material (w.r.t. air). Note: focal length = r / (n-1).
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (lens not hit), -2 (on backside)

Note: reflections in the lens etc. are not taken into account (should be part of transmission='reflectivity' curve) Boundaries between Fresnel lens steps are ignored as well as any diffraction.
Definition at line 1247 of file sim_imaging.c.
References M_PI, and RandGauss().

8.16.3.40 rhof_()

```
double rhof_ (
            double * height )
```

Density of atmosphere [g/cm^3] at given height a.s.l.
[cm].
Definition at line 153 of file sim_telarray.c.

8.16.3.41 sim_calib_events()

```
void sim_calib_events (
            int events,
            int laser,
            int open,
            struct telescope_array * array,
            double * atimes,
            double * aamp,
            long maxpe )
```

Simulate calibration events in the Cherenkov telescopes.
This can be pedestal events (lid opened or closed) and laser events. (Updating of calibration constants needs to be implemented. This function is not presently used by default.)

Parameters

events	Number of events of same type to be simulated
laser	1 for laser events, 0 otherwise
open	1 if camera lid is open, 0 if closed
array	structure of internal array configuration
atimes	buffer space for temporarily storing photo-electron times
aamp	buffer space for temporarily storing photo-electron amplitudes (may be NULL if not interested in that)
maxpe	The maximum number of p.e. supported by the supplied buffer(s)

Returns

(none)

Definition at line 289 of file sim_telarray.c.

8.16.3.42 solve_quadratic_equation()

```
int solve_quadratic_equation (
            double a,
            double b,
            double c,
            double * x1,
            double * x2 )
```

Solve a quadratic equation a∗x∗∗2 + b∗x + c = 0.

Returns

0: OK, we have solutions, -1: no solution, -2: a=b=c=0

Definition at line 716 of file sim_imaging.c.
Referenced by baffle_intersection(), line_ztube_intersection(), reflect_on_parabolic_mirror(), reflect_on_↩ polynomial_mirror(), and reflect_on_spherical_mirror().

8.16.3.43 sum_adc_bins()

```
void sum_adc_bins (
            struct telescope_array * array )
```

Calculate (F)ADC sums in a time interval common to all channels.
Alternatively, this can be the peak amplitude instead of the sum, depending on the peak_sensing option. The start of the integration window is determined by the trigger. This function also checks for overflows.
Definition at line 4196 of file sim_signal.c.
References telescope_array::electronics, camera_electronics::fadc_bins, camera_electronics::fadc_max↩ _signal, camera_electronics::fadc_max_sum, camera_electronics::groups_triggered, camera_electronics↩ ::interval, camera_electronics::long_event, camera_electronics::longsum_bins, camera_electronics::longsum_↩ offset, telescope_array::ntel, telescope_array::options, pm_and_fadc_channel::overflow, mc_tel_options::peak↩ _sensing, camera_electronics::pixels, pm_and_fadc_channel::signal, camera_electronics::simulated, pm_↩ and_fadc_channel::sum_adc, pm_and_fadc_channel::sum_bins, camera_electronics::sum_bins, camera_↩ electronics::sum_offset, camera_electronics::sum_start, mc_options::tel_options, camera_electronics::telescope, and camera_electronics::trigger_time.

8.16.3.44 tel_newdir()

```
int tel_newdir (
            double phi,
            double theta,
            double tel_rand,
            struct telescope_optics * optics )
```

Set up new transformations between ground and telescope.
This function is called when the viewing direction is changed (and at configuration time).

Parameters

phi	Telescope pointing azimuth angle (N->E) as far as known [rad].
theta	Telescope pointing zenith angle as far as known [rad].
tel_rand	Additional and not known error in telescope azimuth and altitude angles.
optics	Telescope optics parameters.

Definition at line 2121 of file sim_imaging.c.
References telescope_optics::alt_optics_offset, telescope_optics::az_alt_offset, make_trafo(), transform_off↩
_struct::offset0, transform_off_struct::offset1, transform_off_struct::offset2, RandGauss(), transform_struct::rot, transform_off_struct::rot1, transform_off_struct::rot2, transform_off_struct::simple, transform_off_struct::strmat, and telescope_optics::tel_trans.
Referenced by tel_setup_secondary().

8.16.3.45 tel_setup_primary()

```
int tel_setup_primary (
            struct telescope_optics * optics,
            double phi,
            double theta )
```

Set up all the transformations between ground, telescope, mirror, and camera reference frames, for telescopes with the camera in the primary focus (mirror_class 0 or 1).
@paran optics Optics parameters

Parameters

phi	Azimuth angle (N->E) [rad]
theta	Zenith angle [rad]

< Focal length of the optical system (for image scale) [cm]
< Dish shape scale length; = radius of curvature (half of r.o.c.) for DC (parabolic) [cm]
< Focal length of mirror facets without explicit value in config file [cm]
< Grading of mirror facet focal length (outer mirror f minus inner mirror f) [cm]
< Two components: R.m.s. and maximum spread of assigned mirror focal lengths (incl. neg. f value in config file) [cm]
< 0 for Davies-Cotton, 1 for parabolic dish shape.
Definition at line 2239 of file sim_imaging.c.

8.16.3.46 telescope_trigger()

```
void telescope_trigger (
            struct pm_camera * cam,
            struct camera_electronics * el )
```

Find out if (and when) a single telescope is triggered.
The trigger decision is made up for any number of trigger groups. Trigger groups can overlap in the list of pixels contributing. Different schemes/algorithms are available: majority/analog sum/digital sum. The telescope trigger time is the earliest time of any trigger group fired. This function sets values in the provided camera electronics struct.

Parameters

cam	Poiter to geometric and optical parameters of the camera.
el	Pointer to all the camera electronics details of the telescope.

Definition at line 3095 of file sim_signal.c.

8.16.3.47 thick_()

```
double thick_ (
            double * height )
```

'Thickness' (column density) of atmosphere [g/cm^2] from space down to given height a.s.l.
[cm].

Definition at line 168 of file sim_telarray.c.

8.16.3.48 trace_photon_in_fresnel()

```
int trace_photon_in_fresnel (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance )
```

Trace a photon through the whole telescope to the camera (if hit).
This variant is for a Fresnel lens.
Definition at line 3750 of file sim_imaging.c.

8.16.3.49 trace_photon_in_paraboloid()

```
int trace_photon_in_paraboloid (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance )
```

Trace a photon through the whole telescope to the camera (if hit).
This variant is for a parabolic single mirror.
Definition at line 3408 of file sim_imaging.c.

8.16.3.50 trace_photon_in_segmented()

```
int trace_photon_in_segmented (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance )
```

Trace a photon through the whole telescope to the camera (if hit).

This variant is for the usual segmented mirror setup with dish shape ranging from Davies-Cotton to parabolic, always with spherical mirror tiles.
Definition at line 2717 of file sim_imaging.c.

8.16.3.51 trace_photon_with_secondary()

```
int trace_photon_with_secondary (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance,
            int bypass )
```

Trace a photon through the whole telescope to the camera (if hit).
This variant is for a secondary mirror optics with primary and secondary mirrors in one piece each. Both mirros can have rather general rotation-symmetric shapes, defined by even polynomial coefficients.
Definition at line 4963 of file sim_imaging.c.

8.16.3.52 transform()

```
void transform (
            double * p,
            double * d,
            const struct transform_struct * ts,
            int mode )
```

Rotation plus translation transformation (forw.
and backw.)

Parameters

p	Position 3-D vector (offset to be applied). Rewritten.
d	Direction 3-D vector (only rotated). Rewritten.
ts	Transformation structure (with only one offset).
mode	Use +1 for forward, -1 for backward transformation.

Definition at line 1955 of file sim_imaging.c.
References transform_struct::offset, and transform_struct::rot.
Referenced by transform_off().

8.16.3.53 transform_off()

```
void transform_off (
            double * p,
            double * d,
            const struct transform_off_struct * ts,
            int mode )
```

Transformation with extra offsets between rotation axes (forw.
and backw.)

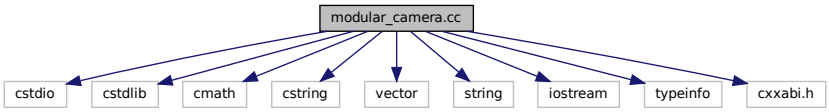
Definition at line 2019 of file sim_imaging.c.
References transform_off_struct::offset0, transform_off_struct::offset1, transform_off_struct::offset2, transform_off_struct::rot1, transform_off_struct::rot2, transform_off_struct::simple, transform_off_struct::strmat, and transform().

8.17 modular_camera.cc File Reference

Generate pixel and trigger list for camera made up by modules (or 'drawers') of pixels.

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <cstring>
#include <vector>
#include <string>
#include <iostream>
#include <typeinfo>
#include <cxxabi.h>
```

Include dependency graph for modular_camera.cc:



Data Structures

- class Offset

  Offset of pixels or modules w.r.t.
- class PixelPos

  Positions of pixels, also including pixel shape type (although not really needed)
- class PixelList

  Lists of pixels, holding both positions and pixel IDs.
- class ModulePixelGenerator

  Base class for all pixel generators.
- class VoidPixelGenerator

  A real although dummy generator, not having any pixels.
- class SingleHexHoriPixelGenerator

  Hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.
- class ThreeHexHoriPixelGenerator

  Three hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.
- class SevenHexHoriPixelGenerator

  Seven hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.
- class SixteenHexHoriPixelGenerator
- class SingleHexVertPixelGenerator

  Hexagonal pixels with flat-to-flat in y direction, corner-to-corner in x direction.
- class ThreeHexVertPixelGenerator

  Three hexagonal pixels with flat-to-flat in x direction, corner-to-corner in y direction.
- class SevenHexVertPixelGenerator

  Seven hexagonal pixels with flat-to-flat in y direction, corner-to-corner in x direction.
- class SixteenHexVertPixelGenerator

- class SingleSquarePixelGenerator

  Single square pixels.
- class FourSquarePixelGenerator

  Four square pixels.
- class SixteenSquarePixelGenerator

  Sixteen square pixels.
- class SixtyFourSquarePixelGenerator
- class CamModule

Macros

- #define _XSTR_(s) _STR_(s)

  Expand a macro first and then enclose in string.
- #define _STR_(s) #s

  Enclose in string without macro expansion.
- #define **SHOW_MACRO**(s)
- #define MPG SevenHexHoriPixelGenerator

  The default generator is the 7-pixel horizontal hex one.

Functions

- Offset **operator**∗ (double f, const Offset &o)
- ostream & **operator**<< (ostream &os, const Offset &o)
- ostream & **operator**<< (ostream &os, const PixelList &p)
- void **show_trigger** (int scheme, const string forwhat, const vector< CamModule > &cmv, const PixelList &all, const string &min_mult, double scale, double sep, double module_gap, double max_nb_dist, double max_nb2_dist)
- int main (int argc, char ∗∗argv)

Variables

- static double **surface_radius** = 0.

8.17.1 Detailed Description

Generate pixel and trigger list for camera made up by modules (or 'drawers') of pixels.
Compile with one of the following definitions: HEX_1H HEX_3H HEX_7H HEX_1V HEX_7V HEX_16H HEX_16V SQR_1 SQR_4 SQR_16 SQR_64

Author

Konrad Bernloehr

8.17.2 Macro Definition Documentation

8.17.2.1 SHOW_MACRO

```
#define SHOW_MACRO(
                s )
```

Value:

```
    if ( strcmp(#s,_XSTR_(s)) != 0 ) \
    { if ( strcmp("", _XSTR_(s)) != 0 && strcmp("1", _XSTR_(s)) != 0 ) \
       cerr « #s "=" _XSTR_(s) ; else cerr « #s ; }
```

Definition at line 52 of file modular_camera.cc.

8.17.3 Function Documentation

8.17.3.1 main()

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

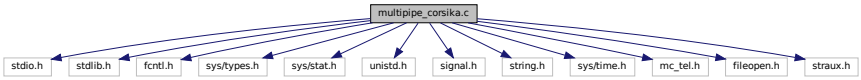
< pixel spacing [cm] (this differs from smartpix program!)
< open pixel size is smaller than scale by this amount.
< Max number of module rows in (u,v,w) (excluding central module).
< Alternate condition: maximum radius of pixel position.
< Hexagonal camera size limit.
Definition at line 1113 of file modular_camera.cc.
References MPG.

8.18 multipipe_corsika.c File Reference

This file contains a multiplexer for CORSIKA IACT data from CORSIKA itself to several telescope simulations (and to disk or anywhere else, if desired).

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <sys/time.h>
#include "mc_tel.h"
#include "fileopen.h"
#include "straux.h"
```

Include dependency graph for multipipe_corsika.c:



Data Structures

- struct pipecmd
- struct postproc_entry

Macros

- #define **MYPATHLEN** 8193
- #define **MAXTEL** 1000
- #define **MAX_ARRAY** 200

Typedefs

- typedef struct postproc_entry **PostProc**

Functions

- void stop_signal_function (int isig)

  Stop the program gracefully when it catches an INT or TERM signal.
- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- void pipe_signal_function (int isig)

  This signal handler gets involved when a SIGPIPE is produced on input or output.
- void **seq_handler** (int signo, siginfo_t *info, void *context)
- void close_all ()

  Close all output pipes.
- void format_num (char *text, size_t maxl, double num)

  Special formatting of floating point numbers.
- pid_t find_corsika_pid ()

  Try to find the CORSIKA process in the list of ancestors of the current process.
- void show_procinfo (pid_t pid)

  Print information about a process, based on the process ID.
- int add_procinfo (pid_t pid)

  If no info about the given process ID was printed before and not too many processed got printed so far, print process information and keep track that this process ID was already shown.
- void show_procgroup ()

  Show the processes that are in the same process group as the current process.
- double **tvtime** (void)
- void hup_signal_function (int __attribute__((unused)) isig)

  Produce intermediate output when the program catches an HUP signal.
- void seq_handler (int signo, siginfo_t *info, void __attribute__((unused)) *context)

  Handle the signal that the telescope simulation should send in sequential mode after having processed a block of data.
- PostProc * add_postproc_cmd (PostProc *pp_start, const char *cmd)

  Add a post-processing command at the end of an existing list or as the first of a list.
- int setenv_int_array (const char *name, int *ival, int nval)

  Set an environment value with a comma-separated list of integers.
- int setenv_dbl_array (const char *name, double *dval, int nval)

  Set an environment value with a comma-separated list of floating point values.
- void syntax (void)

  Tell how to run this program.
- int **main** (int argc, char **argv)

Variables

- struct pipecmd * **pipe_base**
- static int **pcount** = -2
- static int **signo_seq_ini** = 0
- static int **nsleep** = 3
- static int **sleep_time** [5] = { 0, 1, 10, 30, 0 }
- static size_t **sleep_stats** [5]
- static int **startup_slept**
- static int **interrupted** = 0
- static int **broken** = 0
- static int **seq_ready** = 0
- static int n_list_proc = 0

  Number of processes about which we already showed info.
- static int list_proc [20]

Process IDs of which info was shown.

- static int max_list_proc = 20

  Maximum number of process IDs to show info.

- static size_t **n_pipes** = 0
- static int * **optional_pipes**
- static double **tvstart** = 0

8.18.1 Detailed Description

This file contains a multiplexer for CORSIKA IACT data from CORSIKA itself to several telescope simulations (and to disk or anywhere else, if desired).

Author

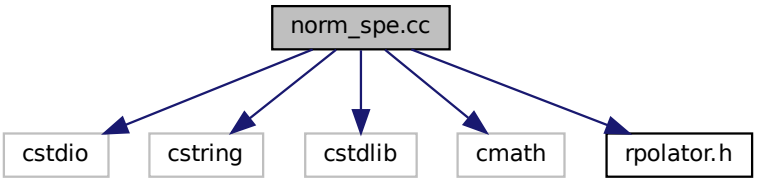
Konrad Bernloehr

8.19 norm_spe.cc File Reference

Normalize a single-p.e.

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include "rpolator.h"
```

Include dependency graph for norm_spe.cc:



Macros

- #define **MAX_SPE** 10000

Functions

- int **read_table3b** (const char *fname, int maxrow, double *col1, double *col2, double *col3)
- void **syntax** (const char *prgm)
- int **main** (int argc, char **argv)

Variables

- int **no_action** = 0
- int **no_action2** = 0
- int **no_action3** = 0
- int **have_col3** = 1

8.19.1 Detailed Description

Normalize a single-p.e.
amplitude distribution.

Author

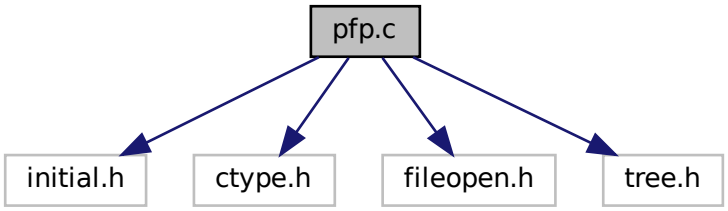
Konrad Bernloehr

8.20 pfp.c File Reference

Preprocessor accepting a syntax similar to the C preprocessor but less intrusive and simpler.

```
#include "initial.h"
#include <ctype.h>
#include "fileopen.h"
#include "tree.h"
```

Include dependency graph for pfp.c:



Data Structures

- struct IncPath

Macros

- #define **_GNU_SOURCE** 1
- #define **WITH_EXISTS** 1
- #define **TRUE** 1
- #define **FALSE** 0
- #define **ERRNUM** 4
- #define **MAXFNAME** 80
- #define **DEFAULT_OUTPUT_EXTENSION** ".for"
- #define **EXTENSION_REQUIRED**
- #define **error**(msg) fprintf(stderr,"%s (%d): %s\n",fname,line_num,msg)
- #define **MAX_IF** 33
- #define **MAXLEN** 10000

Functions

- int include_file (char ∗fname, int flag, int ifexists)

  Include and process text from a file into what is currently being processed.
- void **open_output** (char ∗)
- void **define** (TREE ∗, int, long, const char ∗)
- int **getword** (char ∗, int ∗, char ∗, int, char, char)

- TREE * **expand_if** (char *)
- TREE * **expand_if_line** (char *)
- int **inputprocess** (char *line0, char *line, int len, int verb)
- void **syntax** (int ec)
- char * **strupr** (char *)
- char * **strlwr** (char *)
- char * **my_strcasestr** (char *haystack, const char *needle)
- int **main** (int argc, char **argv)

Variables

- static struct IncPath **inc_root**
- static struct IncPath ** **inc_next**
- FILE * **output**
- TREE * **common**
- TREE * **defined**
- TREE * **TrueTree**
- TREE * **FalseTree**
- int **in_common**
- int **inc_level**
- int **new_output**
- char **message** [2000]
- static const char * **empty_string** = ""
- static int **return_code** = 0
- static char **control_character** = '#'
- static int **quiet** = 0
- static int **historian** = 0
- static int **varsub** = 0
- static int **error_message_mode** = 0
- static int **current_line_num** = 0
- static const char * **current_fname** = "(No active file)"
- static int **add_missing_newline** = 1

8.20.1 Detailed Description

Preprocessor accepting a syntax similar to the C preprocessor but less intrusive and simpler.
This preprocessor was initially termed the 'Portable Fortran Preprocessor' but is now used for preprocessing configuration files. Compared to the C preprocessor the main distinguishing factor is that defined variables are only replaced where explicitely included in '$()' and variable substitution is enabled with the '-V' option. Only in '#if' expression are variables automatically substituted.

Author

Konrad Bernloehr

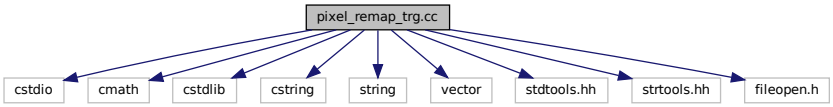
8.21 pixel_remap_trg.cc File Reference

Take two sim_telarray camera definitions, with a given scaling factor and rotation angle between them and find the pixel mapping between first and second file and then remap all pixel IDs in all trigger definitions from the first file to match the pixels from the second file.

```
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <string>
#include <vector>
#include "stdtools.hh"
```

```
#include "strtools.hh"
#include "fileopen.h"
```

Include dependency graph for pixel_remap_trg.cc:



Data Structures

- class Offset

  Offset of pixels or modules w.r.t.
- class PixelType
- class PixelPos

  Positions of pixels, also including pixel shape type (although not really needed)
- class TriggerGroup
- class PixelPatch

Functions

- ostream & **operator**<< (ostream &os, const Offset &o)
- ostream & **operator**<< (ostream &os, const PixelPos &p)
- ostream & **operator**<< (ostream &os, const TriggerGroup &t)
- void **syntax** (int argc, char **argv, int iarg)
- template<class T >
  void **print_implode** (vector< T > v, const string &d=",", const string &dbeg="", const string &dend="", const string &dsec="(same)")
- char * **strip_comment_line** (char *line)
- int **pixel_extract** (FILE *f, PixelType &pt, vector< PixelPos > &ppv)
- int **trg_extract** (FILE *f, const vector< string > &trgs, vector< TriggerGroup > &tgv)
- int **main** (int argc, char **argv)

8.21.1 Detailed Description

Take two sim_telarray camera definitions, with a given scaling factor and rotation angle between them and find the pixel mapping between first and second file and then remap all pixel IDs in all trigger definitions from the first file to match the pixels from the second file.

Author

Konrad Bernloehr

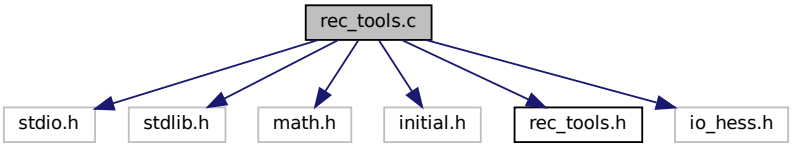
8.22 rec_tools.c File Reference

Tools for shower geometric reconstruction.

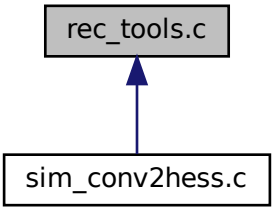
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "initial.h"
#include "rec_tools.h"
```

```
#include "io_hess.h"
```

Include dependency graph for rec_tools.c:



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



Macros

- #define **MAX_TEL** 100
- #define **WT_DISP** 1

Functions

- double line_point_distance (double xp1, double yp1, double zp1, double cx, double cy, double cz, double x, double y, double z)

  Distance between a straight line and a point in space.
- void angles_to_offset (double obj_azimuth, double obj_altitude, double azimuth, double altitude, double focal_length, double ∗xoff, double ∗yoff)

  Transform telescope and object Alt/Az to offset in camera.
- void offset_to_angles (double xoff, double yoff, double azimuth, double altitude, double focal_length, double ∗obj_azimuth, double ∗obj_altitude)

  Transform from offset in camera to corresponding Az/Alt.
- void get_shower_trans_matrix (double azimuth, double altitude, double trans[ ][3])

  Calculate transformation matrix.
- void cam_to_ref (double ximg, double yimg, double phi, double ref_azimuth, double ref_altitude, double cam_rot, double azimuth, double altitude, double focal_length, double ∗axref, double ∗ayref, double ∗phiref)

  Transform from one camera to common reference frame.
- int intersect_lines (double xp1, double yp1, double phi1, double xp2, double yp2, double phi2, double ∗xs, double ∗ys, double ∗sang)

  Intersect pairs of lines.
- static double **square** (double a)

- int shower_geometric_reconstruction (int ntel, const double *amp, const double *ximg, const double *yimg, const double *phi, const double *disp, const double *xtel, const double *ytel, const double *ztel, const double *az, const double *alt, const double *flen, const double *cam_rot, double ref_az, double ref_alt, int flag, double *shower_az, double *shower_alt, double *var_dir, double *xc, double *yc, double *var_core)

  Simple reconstruction by intersecting pairs of lines.
- double angle_between (double azimuth1, double altitude1, double azimuth2, double altitude2)

  Calculate the angle between two directions given in spherical coordinates.

8.22.1 Detailed Description

Tools for shower geometric reconstruction.
Shower geometric reconstruction based on the major axes of the telescope images. The image parameters from each telescope are transformed to a common reference frame first before the average intersection point of all images is calculated in plane coordinates.

Author

Konrad Bernloehr

Date

2000 ... 2019

8.22.2 Function Documentation

8.22.2.1 angle_between()

```
double angle_between (
            double azimuth1,
            double altitude1,
            double azimuth2,
            double altitude2 )
```

Calculate the angle between two directions given in spherical coordinates.

Returns

The angle between the two directions in units of radians.

Definition at line 558 of file rec_tools.c.
References M_PI.

8.22.2.2 angles_to_offset()

```
void angles_to_offset (
            double obj_azimuth,
            double obj_altitude,
            double azimuth,
            double altitude,
            double focal_length,
            double * xoff,
            double * yoff )
```

Transform telescope and object Alt/Az to offset in camera.
Transform from given telescope and object angles (Az/Alt) to the offset the object has in the camera plane.
This does not account for any rotation of the camera and its pixels.
Definition at line 79 of file rec_tools.c.
Referenced by cam_to_ref().

8.22.2.3 cam_to_ref()

```
void cam_to_ref (
            double ximg,
            double yimg,
            double phi,
            double ref_azimuth,
            double ref_altitude,
            double cam_rot,
            double azimuth,
            double altitude,
            double focal_length,
            double * axref,
            double * ayref,
            double * phiref )
```

Transform from one camera to common reference frame.
Transform from the camera plane coordinate system of a telescope looking to altitude/azimuth to a plane coordinate system of a potential telescope looking to a reference direction ref_azimuth,ref_altitude and having unit focal length. Rotation of image angles is accounted for but not imaging errors.
Definition at line 206 of file rec_tools.c.
References angles_to_offset(), and offset_to_angles().

8.22.2.4 get_shower_trans_matrix()

```
void get_shower_trans_matrix (
            double azimuth,
            double altitude,
            double trans[][3] )
```

Calculate transformation matrix.
Calculate transformation matrix from horizontal reference frame to one z axis in the given Az/Alt direction and the x axis in the plane defined by Az/Alt and zenith.
Definition at line 173 of file rec_tools.c.

8.22.2.5 intersect_lines()

```
int intersect_lines (
            double xp1,
            double yp1,
            double phi1,
            double xp2,
            double yp2,
            double phi2,
            double * xs,
            double * ys,
            double * sang )
```

Intersect pairs of lines.
Intersect a pair of straight lines in a plane and return the intersection point and the angle at which the lines intersect.
Definition at line 246 of file rec_tools.c.
References M_PI.

8.22.2.6 line_point_distance()

```
double line_point_distance (
            double xp1,
            double yp1,
            double zp1,
```

```
            double cx,
            double cy,
            double cz,
            double x,
            double y,
            double z )
```

Distance between a straight line and a point in space.

Parameters

xp1,yp1,zp1	reference point on the line
cx,cy,cz	direction cosines of the line
x,y,z	point in space

Returns

distance

Definition at line 53 of file rec_tools.c.

8.22.2.7 offset_to_angles()

```
void offset_to_angles (
            double xoff,
            double yoff,
            double azimuth,
            double altitude,
            double focal_length,
            double * obj_azimuth,
            double * obj_altitude )
```

Transform from offset in camera to corresponding Az/Alt.
Transform from the offset an object or image has in the camera plane of a telescope to the corresponding Az/Alt. This does not account for any rotation of the camera and its pixels. (xoff and yoff are assumed to be corrected for camera rotation).
Definition at line 128 of file rec_tools.c.
Referenced by cam_to_ref().

8.22.2.8 shower_geometric_reconstruction()

```
int shower_geometric_reconstruction (
            int ntel,
            const double * amp,
            const double * ximg,
            const double * yimg,
            const double * phi,
            const double * disp,
            const double * xtel,
            const double * ytel,
            const double * ztel,
            const double * az,
            const double * alt,
            const double * flen,
            const double * cam_rot,
            double ref_az,
            double ref_alt,
            int flag,
```

```
double * shower_az,
double * shower_alt,
double * var_dir,
double * xc,
double * yc,
double * var_core )
```

Simple reconstruction by intersecting pairs of lines.
Simple geometric shower reconstruction by intersecting pairs of straigh lines (from major axis of second moments ellipses after transformation to a common plane), first for the shower direction and then for the core position. No errors on reconstructed direction or core position are calculated. This should sooner or later be superceded by a fit procedure taking advantage of estimated errors on image positions and angles.

Parameters

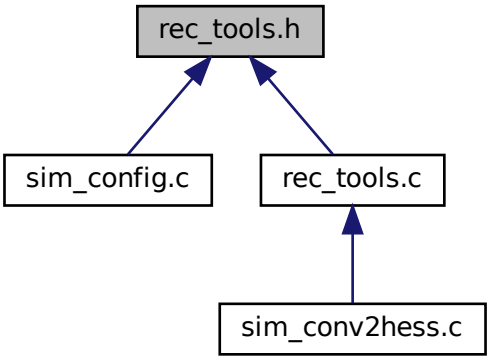
ntel	The number of telescopes with suitable images.
amp	The image amplitudes in each suitable telescope [p.e.].
ximg	The image c.o.g. x positions in the local camera coordinate systems.
yimg	The image c.o.g. y positions in the local camera coordinate systems.
phi	The image major axis direction [rad].
disp	The DISP parameter (1.-width/length), used for giving preference to elongated images. Set all to 1.0 if unknown or no preference wanted. Can also be passed as a NULL pointer instead.
xtel	The x coordinate of the telescope positions within array [m].
ytel	The y coordinate of the telescope positions within array [m].
ztel	The z coordinate of the telescope positions within array [m].
az	The azimuth angles to which the telescopes are pointing (N->E->S->W) [rad].
alt	The altitude angles to which the telescopes are pointing [rad].
flen	The focal length to which ximg and yimg are scaled (1.0 if in units of radians, otherwise flen is in meters).
cam_rot	Camera rotation angle [rad].
ref_az	The reference azimuth angle (system nominal azimuth) [rad].
ref_alt	The reference altitude angle (system nominal altitude) [rad].
flag	Use the reconstucted direction to derive the core position (0) or use the nominal direction for that (1 or any other non-zero). The second version may sightly improve core distance and thus energy accuracy for well-defined point sources.
shower_az	Return the reconstructed shower azimuth angle (N->E->S->W) [rad].
shower_alt	Return the reconstructed shower altitude angle [rad].
var_dir	Variance (dx**2+dy**2)/ntel of reconstructed direction for more than two images. Can be NULL if you are not interested in it.
xc	Return the reconstructed core position x coordinate (at z=0) [m].
yc	Return the reconstructed core position y coordinate (at z=0) [m].
var_core	Variance (dx**2+dy**2)/ntel of reconstructed core position for more than two images. Can be NULL if you are not interested in it.

Definition at line 372 of file rec_tools.c.

8.23 rec_tools.h File Reference

Interface for tools for shower geometric reconstruction.

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



Functions

- void angles_to_offset (double obj_azimuth, double obj_altitude, double azimuth, double altitude, double focal_length, double ∗xoff, double ∗yoff)

  Transform telescope and object Alt/Az to offset in camera.
- void offset_to_angles (double xoff, double yoff, double azimuth, double altitude, double focal_length, double ∗obj_azimuth, double ∗obj_altitude)

  Transform from offset in camera to corresponding Az/Alt.
- void get_shower_trans_matrix (double azimuth, double altitude, double trans[ ][3])

  Calculate transformation matrix.
- void cam_to_ref (double ximg, double yimg, double phi, double ref_azimuth, double ref_altitude, double cam_rot, double azimuth, double altitude, double focal_length, double ∗axref, double ∗ayref, double ∗phiref)

  Transform from one camera to common reference frame.
- int intersect_lines (double xp1, double yp1, double phi1, double xp2, double yp2, double phi2, double ∗xs, double ∗ys, double ∗sang)

  Intersect pairs of lines.
- int shower_geometric_reconstruction (int ntel, const double ∗amp, const double ∗ximg, const double ∗yimg, const double ∗phi, const double ∗disp, const double ∗xtel, const double ∗ytel, const double ∗ztel, const double ∗az, const double ∗alt, const double ∗flen, const double ∗cam_rot, double ref_az, double ref_alt, int flag, double ∗shower_az, double ∗shower_alt, double ∗var_dir, double ∗xc, double ∗yc, double ∗var_core)

  Simple reconstruction by intersecting pairs of lines.
- double angle_between (double azimuth1, double altitude1, double azimuth2, double altitude2)

  Calculate the angle between two directions given in spherical coordinates.
- double line_point_distance (double xp1, double yp1, double zp1, double cx, double cy, double cz, double x, double y, double z)

  Distance between a straight line and a point in space.

8.23.1 Detailed Description

Interface for tools for shower geometric reconstruction.

Author

Konrad Bernloehr

Date

2001 ... 2018

8.23.2 Function Documentation

8.23.2.1 angle_between()

```
double angle_between (
            double azimuth1,
            double altitude1,
            double azimuth2,
            double altitude2 )
```

Calculate the angle between two directions given in spherical coordinates.

Returns

The angle between the two directions in units of radians.

Definition at line 558 of file rec_tools.c.
References M_PI.

8.23.2.2 angles_to_offset()

```
void angles_to_offset (
            double obj_azimuth,
            double obj_altitude,
            double azimuth,
            double altitude,
            double focal_length,
            double * xoff,
            double * yoff )
```

Transform telescope and object Alt/Az to offset in camera.
Transform from given telescope and object angles (Az/Alt) to the offset the object has in the camera plane.
This does not account for any rotation of the camera and its pixels.
Definition at line 79 of file rec_tools.c.
Referenced by cam_to_ref().

8.23.2.3 cam_to_ref()

```
void cam_to_ref (
            double ximg,
            double yimg,
            double phi,
            double ref_azimuth,
            double ref_altitude,
            double cam_rot,
            double azimuth,
            double altitude,
            double focal_length,
            double * axref,
            double * ayref,
            double * phiref )
```

Transform from one camera to common reference frame.
Transform from the camera plane coordinate system of a telescope looking to altitude/azimuth to a plane coordinate system of a potential telescope looking to a reference direction ref_azimuth,ref_altitude and having unit focal length.
Rotation of image angles is accounted for but not imaging errors.

Definition at line 206 of file rec_tools.c.
References angles_to_offset(), and offset_to_angles().

8.23.2.4 get_shower_trans_matrix()

```
void get_shower_trans_matrix (
            double azimuth,
            double altitude,
            double trans[][3] )
```

Calculate transformation matrix.
Calculate transformation matrix from horizontal reference frame to one z axis in the given Az/Alt direction and the x axis in the plane defined by Az/Alt and zenith.
Definition at line 173 of file rec_tools.c.

8.23.2.5 intersect_lines()

```
int intersect_lines (
            double xp1,
            double yp1,
            double phi1,
            double xp2,
            double yp2,
            double phi2,
            double * xs,
            double * ys,
            double * sang )
```

Intersect pairs of lines.
Intersect a pair of straight lines in a plane and return the intersection point and the angle at which the lines intersect.
Definition at line 246 of file rec_tools.c.
References M_PI.

8.23.2.6 line_point_distance()

```
double line_point_distance (
            double xp1,
            double yp1,
            double zp1,
            double cx,
            double cy,
            double cz,
            double x,
            double y,
            double z )
```

Distance between a straight line and a point in space.

Parameters

xp1,yp1,zp1	reference point on the line
cx,cy,cz	direction cosines of the line
x,y,z	point in space

Returns

distance

Definition at line 53 of file rec_tools.c.

8.23.2.7 offset_to_angles()

```
void offset_to_angles (
            double xoff,
            double yoff,
            double azimuth,
            double altitude,
            double focal_length,
            double * obj_azimuth,
            double * obj_altitude )
```

Transform from offset in camera to corresponding Az/Alt.
Transform from the offset an object or image has in the camera plane of a telescope to the corresponding Az/Alt. This does not account for any rotation of the camera and its pixels. (xoff and yoff are assumed to be corrected for camera rotation).
Definition at line 128 of file rec_tools.c.
Referenced by cam_to_ref().

8.23.2.8 shower_geometric_reconstruction()

```
int shower_geometric_reconstruction (
            int ntel,
            const double * amp,
            const double * ximg,
            const double * yimg,
            const double * phi,
            const double * disp,
            const double * xtel,
            const double * ytel,
            const double * ztel,
            const double * az,
            const double * alt,
            const double * flen,
            const double * cam_rot,
            double ref_az,
            double ref_alt,
            int flag,
            double * shower_az,
            double * shower_alt,
            double * var_dir,
            double * xc,
            double * yc,
            double * var_core )
```

Simple reconstruction by intersecting pairs of lines.
Simple geometric shower reconstruction by intersecting pairs of straigh lines (from major axis of second moments ellipses after transformation to a common plane), first for the shower direction and then for the core position. No errors on reconstructed direction or core position are calculated. This should sooner or later be superceded by a fit procedure taking advantage of estimated errors on image positions and angles.

Parameters

ntel	The number of telescopes with suitable images.
amp	The image amplitudes in each suitable telescope [p.e.].

Parameters

ximg	The image c.o.g. x positions in the local camera coordinate systems.
yimg	The image c.o.g. y positions in the local camera coordinate systems.
phi	The image major axis direction [rad].
disp	The DISP parameter (1.-width/length), used for giving preference to elongated images. Set all to 1.0 if unknown or no preference wanted. Can also be passed as a NULL pointer instead.
xtel	The x coordinate of the telescope positions within array [m].
ytel	The y coordinate of the telescope positions within array [m].
ztel	The z coordinate of the telescope positions within array [m].
az	The azimuth angles to which the telescopes are pointing (N->E->S->W) [rad].
alt	The altitude angles to which the telescopes are pointing [rad].
flen	The focal length to which ximg and yimg are scaled (1.0 if in units of radians, otherwise flen is in meters).
cam_rot	Camera rotation angle [rad].
ref_az	The reference azimuth angle (system nominal azimuth) [rad].
ref_alt	The reference altitude angle (system nominal altitude) [rad].
flag	Use the reconstucted direction to derive the core position (0) or use the nominal direction for that (1 or any other non-zero). The second version may sightly improve core distance and thus energy accuracy for well-defined point sources.
shower_az	Return the reconstructed shower azimuth angle (N->E->S->W) [rad].
shower_alt	Return the reconstructed shower altitude angle [rad].
var_dir	Variance (dx**2+dy**2)/ntel of reconstructed direction for more than two images. Can be NULL if you are not interested in it.
xc	Return the reconstructed core position x coordinate (at z=0) [m].
yc	Return the reconstructed core position y coordinate (at z=0) [m].
var_core	Variance (dx**2+dy**2)/ntel of reconstructed core position for more than two images. Can be NULL if you are not interested in it.

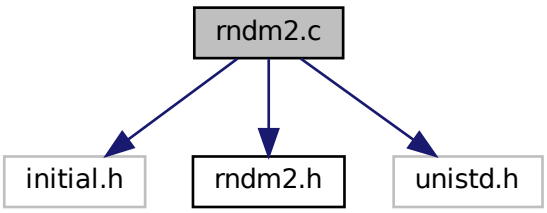
Definition at line 372 of file rec_tools.c.

8.24 rndm2.c File Reference

Random number generators adapted from HEP Random C++ code.

```
#include "initial.h"
#include "rndm2.h"
#include <unistd.h>
```

Include dependency graph for rndm2.c:



Macros

- #define **false** 0
- #define **true** 1
- #define **CHECK_RNG**()

Functions

- void **RandSetVerbose** ()
- void **RandSetQuiet** ()
- static void **init** (void)
- void **SetRandomEngine** (PFVD_t f)
- void Ranlux_setSeed (long nseed, int nlux)

  Ranlux random generator initialisation.
- void Ranlux_setSeeds (long *nseeds, int nlux)

  Set the seed numbers for the Ranlux random number generator.
- void Ranlux_getStatus (int *pseed, int seed_table[24], int *pi_lag, int *pj_lag, int *pcount24, double *pcarry)

  Get the current status of the Ranlux random number generator.
- void Ranlux_setStatus (int *pseed, int seed_table[24], int *pi_lag, int *pj_lag, int *pcount24, double *pcarry)

  Set the current status of the Ranlux random number generator.
- void Ranlux_saveStatus (const char *fname)

  Save the current status of the Ranluy random generator to a file.
- void Ranlux_restoreStatus (const char *fname)

  Restore the status of the Ranlux random generator from a file.
- void Ranlux_showStatus ()

  The the current status of the Ranlux random number generator.
- double Ranlux_RandFlat ()

  Generate random numbers with flat distribution in]0:1[using Ranlux algorithm.
- void Ranlux_RandFlatArray (size_t size, double *vect)

  Generate vectors of 'flat' random numbers with Ranlux algorithm.
- void **RandSetGenerator** (const char *rng_name)
- void **RandSetSeed** (long rseed)
- double RandFlat ()

  Generate random numbers with uniform distibution in]0:1[.
- unsigned long RandInt (unsigned long n)

  Generate integer random numbers with uniform distibution in [0:n-1[.
- void RandFlatArray (size_t size, double *vect)

  Generate vectors of random numbers with uniform distibution in]0:1[.
- void **RandGauss_setFlag** (HepBoolean val)
- HepBoolean **RandGauss_getFlag** ()
- void **RandGauss_setVal** (double nextVal)
- double **RandGauss_getVal** ()
- double RandGauss (double mean, double sigma)

  Generate random numbers following a Gaussian distribution.
- void **RandPoisson_setOldMean** (double val)
- double **RandPoisson_getOldMean** ()
- double **RandPoisson_getMaxMean** ()
- void **RandPoisson_setPStatus** (double sq, double alxm, double g)
- double * **RandPoisson_getPStatus** ()
- static double gammln (double xx)

  Returns the value ln(Gamma(xx) for xx > 0.
- unsigned long RandPoisson (double xm)

  Generate random numbers following Poisson distribution.

- double RandExponential (double mean)

  Generate random numbers following an exponential distribution.
- void **RandSaveStatus** (const char ∗fname)
- void **RandRestoreStatus** (const char ∗fname)

Variables

- static int **rand_verbose** = 1
- static int **nskip**
- static int **luxury** = 3
- static double **float_seed_table** [24]
- static int **i_lag**
- static int **j_lag**
- static double **carry**
- static int **count24**
- static int **int_modulus**
- static double **mantissa_bit_24**
- static double **mantissa_bit_12**
- static long **seed** = 19780503
- static int **lux** = 3
- static int **initialized** = 0
- static long **theSeed**
- static long ∗ **theSeeds**
- static PFVD_t **flatEngine** = NULL
- static HepBoolean **set**
- static double **nextGauss**
- static double **rstatus** [3]
- static double **oldm** = -1.0
- static double **meanMax** = 2.0e9

8.24.1 Detailed Description

Random number generators adapted from HEP Random C++ code.
Random number generators for flat]0:1[, Gauss, Poisson and exponential pseudo random numbers. The source code has been adapted from the HEP Random C++ code included in the GEANT 4 prototype and is based on developments mainly at CERN. For the flat random numbers the 'Ranlux' generator was chosen.

Author

Adapted by Konrad Bernloehr

Date

11 July 1997

8.24.2 Macro Definition Documentation

8.24.2.1 CHECK_RNG

```
#define CHECK_RNG( )
```

Value:

```
if ( flatEngine == NULL ) \
flatEngine = Ranlux_RandFlat
```

Definition at line 52 of file rndm2.c.

8.24.3 Function Documentation

8.24.3.1 gammln()

```
static double gammln (
            double xx )  [static]
```

Returns the value ln(Gamma(xx) for xx $>$ 0.

Full accuracy is obtained for xx $>$ 1. For 0 $<$ xx $<$ 1. the reflection formula (6.1.4) can be used first. (Adapted from Numerical Recipes in C)
Definition at line 774 of file rndm2.c.

8.24.3.2 RandExponential()

```
double RandExponential (
            double mean )
```

Generate random numbers following an exponential distribution.

Parameters

mean	The mean value of the exponential distribution.

Returns

Random number ($>$ 0)

Definition at line 887 of file rndm2.c.
Referenced by fill_traces().

8.24.3.3 RandFlat()

```
double RandFlat (
            void  )
```

Generate random numbers with uniform distibution in]0:1[.
The configured engine for 'flat' distribution is used.

Returns

A new random number.

Definition at line 613 of file rndm2.c.
Referenced by cathode_hit(), delay_signals(), and fill_traces().

8.24.3.4 RandFlatArray()

```
void RandFlatArray (
            size_t size,
            double * vect )
```

Generate vectors of random numbers with uniform distibution in]0:1[.
The configured engine for 'flat' distribution is used.

Parameters

size	The number of random numbers to generate.
vect	The vector to hold the resulting random numbers; must be long enough to hold 'size' doubles.

Definition at line 659 of file rndm2.c.

8.24.3.5 RandGauss()

```
double RandGauss (
            double mean,
            double sigma )
```

Generate random numbers following a Gaussian distribution.
Gaussian random numbers are generated two at the time, so every other time this is called we just return a number generated the time before.

Parameters

mean	Mean value of the Gaussian distribution.
sigma	The standard deviation of the Gaussian distribution.

Returns

A new random number.

Definition at line 712 of file rndm2.c.
Referenced by convert_initial_moni_calib(), fill_traces(), reflect_on_parabolic_mirror(), reflect_on_polynomial_↩ mirror(), reflect_on_spherical_mirror(), refract_in_fresnel_lens(), and tel_newdir().

8.24.3.6 RandInt()

```
unsigned long RandInt (
            unsigned long n )
```

Generate integer random numbers with uniform distibution in [0:n-1[.
The configured engine for 'flat' distribution is used.

Returns

A new random number.

Definition at line 630 of file rndm2.c.

8.24.3.7 RandPoisson()

```
unsigned long RandPoisson (
            double xm )
```

Generate random numbers following Poisson distribution.
Returns as a floating-point number an integer value that is a random deviation drawn from a Poisson distribution of mean xm, using flat() as a source of uniform random numbers. (Adapted from Numerical Recipes in C)

Parameters

xm	Mean value of Poisson distribution.

Returns

Random number (integer>, >= 0)

Definition at line 810 of file rndm2.c.

8.24.3.8 Ranlux_restoreStatus()

```
void Ranlux_restoreStatus (
            const char * fname )
```

Restore the status of the Ranlux random generator from a file.

Parameters

Name	of file to be read (assuming "Ranlux.conf" if it is NULL).

Definition at line 362 of file rndm2.c.

8.24.3.9 Ranlux_saveStatus()

```
void Ranlux_saveStatus (
            const char * fname )
```

Save the current status of the Ranluy random generator to a file.

Parameters

Name	of file to be written (assuming "Ranlux.conf" if it is NULL).

Definition at line 323 of file rndm2.c.

8.24.3.10 Ranlux_setSeed()

```
void Ranlux_setSeed (
            long nseed,
            int nlux )
```

Ranlux random generator initialisation.
The initialisation is carried out using a Multiplicative Congruential generator using formula constants of L'Ecuyer as described in "A review of pseudorandom number generators" (Fred James) published in Computer Physics Communications 60 (1990) pages 329-344

Parameters

nseed	Seed number
nlux	Luxury level (normally: 0 <= nlux <= 4)

Luxury level is set in the same way as the original FORTRAN routine.

level 0 (p=24): equivalent to the original RCARRY of Marsaglia and Zaman, very long period, but fails many tests.

level 1 (p=48): considerable improvement in quality over level 0, now passes the gap test, but still fails spectral test.

level 2 (p=97): passes all known tests, but theoretically still defective.

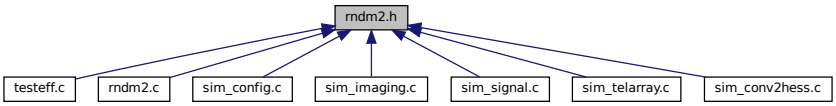
level 3 (p=223): DEFAULT VALUE. Any theoretically possible correlations have very small chance of being observed.

level 4 (p=389): highest possible luxury, all 24 bits chaotic.

Definition at line 134 of file rndm2.c.

8.25 rndm2.h File Reference

Prototypes for random number generators adapted from HEP Random C++ code.
This graph shows which files directly or indirectly include this file:



Typedefs

- typedef int **HepBoolean**
- typedef double(∗ **PFVD_t**) (void)

Functions

- void **SetRandomEngine** (PFVD_t f)
- void Ranlux_setSeed (long seed, int lux)

  Ranlux random generator initialisation.
- void Ranlux_setSeeds (long ∗seeds, int lux)

  Set the seed numbers for the Ranlux random number generator.
- void Ranlux_getStatus (int ∗pseed, int seed_table[24], int ∗pi_lag, int ∗pj_lag, int ∗pcount24, double ∗pcarry)

  Get the current status of the Ranlux random number generator.
- void Ranlux_setStatus (int ∗pseed, int seed_table[24], int ∗pi_lag, int ∗pj_lag, int ∗pcount24, double ∗pcarry)

  Set the current status of the Ranlux random number generator.
- void Ranlux_saveStatus (const char ∗fname)

  Save the current status of the Ranluy random generator to a file.
- void Ranlux_restoreStatus (const char ∗fname)

  Restore the status of the Ranlux random generator from a file.
- void Ranlux_showStatus (void)

  The the current status of the Ranlux random number generator.
- double Ranlux_RandFlat (void)

  Generate random numbers with flat distribution in]0:1[using Ranlux algorithm.
- void Ranlux_RandFlatArray (size_t size, double ∗vect)

  Generate vectors of 'flat' random numbers with Ranlux algorithm.
- void **RandGauss_setFlag** (HepBoolean val)
- HepBoolean **RandGauss_getFlag** (void)
- void **RandGauss_setVal** (double nextVal)

- double **RandGauss_getVal** (void)
- void **RandPoisson_setOldMean** (double val)
- double **RandPoisson_getOldMean** (void)
- double **RandPoisson_getMaxMean** (void)
- void **RandPoisson_setPStatus** (double sq, double alxm, double g)
- double * **RandPoisson_getPStatus** (void)
- void **RandSetGenerator** (const char *rng_name)
- void **RandSetVerbose** (void)
- void **RandSetQuiet** (void)
- void **RandSetSeed** (long seed)
- double RandFlat (void)

  Generate random numbers with uniform distibution in]0:1[.
- unsigned long RandInt (unsigned long n)

  Generate integer random numbers with uniform distibution in [0:n-1[.
- void RandFlatArray (size_t size, double *vect)

  Generate vectors of random numbers with uniform distibution in]0:1[.
- double RandGauss (double mean, double sigma)

  Generate random numbers following a Gaussian distribution.
- unsigned long RandPoisson (double xm)

  Generate random numbers following Poisson distribution.
- double RandExponential (double mean)

  Generate random numbers following an exponential distribution.
- void **RandSaveStatus** (const char *fname)
- void **RandRestoreStatus** (const char *fname)

8.25.1 Detailed Description

Prototypes for random number generators adapted from HEP Random C++ code.

Author

Konrad Bernloehr

Date

11 July 1997

8.25.2 Function Documentation

8.25.2.1 RandExponential()

```
double RandExponential (
            double mean )
```

Generate random numbers following an exponential distribution.

Parameters

mean	The mean value of the exponential distribution.

Returns

Random number (> 0)

Definition at line 887 of file rndm2.c.
Referenced by fill_traces().

8.25.2.2 RandFlat()

```
double RandFlat (
            void  )
```

Generate random numbers with uniform distibution in]0:1[.
The configured engine for 'flat' distribution is used.

Returns

A new random number.

Definition at line 613 of file rndm2.c.
Referenced by cathode_hit(), delay_signals(), and fill_traces().

8.25.2.3 RandFlatArray()

```
void RandFlatArray (
            size_t size,
            double * vect )
```

Generate vectors of random numbers with uniform distibution in]0:1[.
The configured engine for 'flat' distribution is used.

Parameters

size	The number of random numbers to generate.
vect	The vector to hold the resulting random numbers; must be long enough to hold 'size' doubles.

Definition at line 659 of file rndm2.c.

8.25.2.4 RandGauss()

```
double RandGauss (
            double mean,
            double sigma )
```

Generate random numbers following a Gaussian distribution.
Gaussian random numbers are generated two at the time, so every other time this is called we just return a number generated the time before.

Parameters

mean	Mean value of the Gaussian distribution.
sigma	The standard deviation of the Gaussian distribution.

Returns

A new random number.

Definition at line 712 of file rndm2.c.
Referenced by convert_initial_moni_calib(), fill_traces(), reflect_on_parabolic_mirror(), reflect_on_polynomial_↩ mirror(), reflect_on_spherical_mirror(), refract_in_fresnel_lens(), and tel_newdir().

8.25.2.5 RandInt()

```
unsigned long RandInt (
            unsigned long n )
```

Generate integer random numbers with uniform distibution in [0:n-1[.
The configured engine for 'flat' distribution is used.

Returns

A new random number.

Definition at line 630 of file rndm2.c.

8.25.2.6 RandPoisson()

```
unsigned long RandPoisson (
            double xm )
```

Generate random numbers following Poisson distribution.
Returns as a floating-point number an integer value that is a random deviation drawn from a Poisson distribution of mean xm, using flat() as a source of uniform random numbers. (Adapted from Numerical Recipes in C)

Parameters

xm	Mean value of Poisson distribution.

Returns

Random number (integer>, >= 0)

Definition at line 810 of file rndm2.c.

8.25.2.7 Ranlux_restoreStatus()

```
void Ranlux_restoreStatus (
            const char * fname )
```

Restore the status of the Ranlux random generator from a file.

Parameters

Name	of file to be read (assuming "Ranlux.conf" if it is NULL).

Definition at line 362 of file rndm2.c.

8.25.2.8 Ranlux_saveStatus()

```
void Ranlux_saveStatus (
            const char * fname )
```

Save the current status of the Ranluy random generator to a file.

Parameters

Name	of file to be written (assuming "Ranlux.conf" if it is NULL).

Definition at line 323 of file rndm2.c.

8.25.2.9 Ranlux_setSeed()

```
void Ranlux_setSeed (
            long nseed,
```

```
int nlux )
```

Ranlux random generator initialisation.
The initialisation is carried out using a Multiplicative Congruential generator using formula constants of L'Ecuyer as described in "A review of pseudorandom number generators" (Fred James) published in Computer Physics Communications 60 (1990) pages 329-344

Parameters

nseed	Seed number
nlux	Luxury level (normally: 0 <= nlux <= 4)

Luxury level is set in the same way as the original FORTRAN routine.

level 0 (p=24): equivalent to the original RCARRY of Marsaglia and Zaman, very long period, but fails many tests.

level 1 (p=48): considerable improvement in quality over level 0, now passes the gap test, but still fails spectral test.

level 2 (p=97): passes all known tests, but theoretically still defective.

level 3 (p=223): DEFAULT VALUE. Any theoretically possible correlations have very small chance of being observed.

level 4 (p=389): highest possible luxury, all 24 bits chaotic.

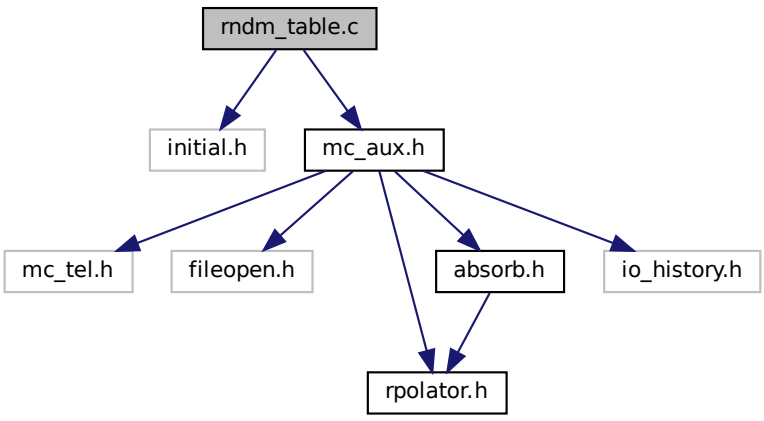
Definition at line 134 of file rndm2.c.

8.26 rndm_table.c File Reference

Random number generation with arbitrary distribution tables.

```
#include "initial.h"
#include "mc_aux.h"
```

Include dependency graph for rndm_table.c:



Functions

- double * make_random_table (double *x, double *y, int n, int ni, int nr)

  Make very fast lookup table for rather flat random number distribution.
- double random_from_table (double *xr, int nr, double randnr)

  Very fast random number table lookup.
- int make_random_ipol_table (double *x, double *y, int n, double **xr, double **yr)

  Make fast lookup table for rather any number distribution.
- double random_from_ipol_table (double *xr, double *yr, int n, double randnr)

  Fast random number table lookup.

8.26.1 Detailed Description

Random number generation with arbitrary distribution tables.

Author

Konrad Bernloehr

8.26.2 Function Documentation

8.26.2.1 make_random_ipol_table()

```
int make_random_ipol_table (
            double * x,
            double * y,
            int n,
            double ** xr,
            double ** yr )
```

Make fast lookup table for rather any number distribution.
The following procedures can be used for relatively fast production of random numbers according to a given table of (not to be normalized) probability values. Tails of a distribution should be sufficiently well reproduced within the limits imposed a) by the chosen generator of flat random numbers (really double, i.e. 48-bit abscissa, or effectively float, i.e. 24-bit abscissa) and b) by the assumption that the probability is constant within each bin of the input table. The bin limits are calculated with the assumption of equidistant bins where the abscissa values of the input table give the middle of each bin. This must hold at least for the two lowest and the two highest bins.

Parameters

x	X coordinates of tabulated 'curve'
y	Y coordinates of tabulated 'curve'
n	number of points in provided table
xr	Pointer to returned X table (or NULL)
yr	Pointer to returned Y table (or NULL)

Returns

0 (o.k.), -1 (error)

Definition at line 185 of file rndm_table.c.
Referenced by read_spe().

8.26.2.2 make_random_table()

```
double* make_random_table (
            double * x,
```

```
            double * y,
            int n,
            int ni,
            int nr )
```

Make very fast lookup table for rather flat random number distribution.
To have a very fast production of random numbers according to a given table of (not to be normalized) probability table, a corresponding table of abscissa bins with equal content is calculated. A flat random number in [0:1] then tells in which bin and where in this bin the wanted random number has to be. This procedure assumes that the probability varies rather modestly. With nr=10000, for example, it would be impossible to reproduce the tail of a distribution where the probability falls to well below one part in 10000 of the peak probability. Before applying this procedure, check which ni and nr values you need to reproduce your give table with sufficient accuracy.
The bin limits are calculated with the assumption of equidistant bins where the abscissa values of the input table give the middle of each bin. This must hold at least for the two lowest and the two highest bins.

Parameters

x	X coordinates of tabulated 'curve'
y	Y coordinates of tabulated 'curve'
n	number of points in provided table
ni	number of points used in intermediate table
nr	number of points (minus 1) in final lookup table

Returns

pointer to final lookup table (or NULL in case of error)

Definition at line 72 of file rndm_table.c.
References rpol().
Referenced by read_spe().

8.26.2.3 random_from_ipol_table()

```
double random_from_ipol_table (
            double * xr,
            double * yr,
            int n,
            double randnr )
```

Fast random number table lookup.
Get a random number from a lookup table as created by make_random_ipol_table() for a tabulated probability distribution.

Parameters

xr	Table as returned from make_random_ipol_table()
yr	Table as returned from make_random_ipol_table()
n	as provided to make_random_ipol_table()
randnr	A uniform pseudorandom number in [0:1]

Returns

random number according to table

Definition at line 243 of file rndm_table.c.
References rpol().
Referenced by fill_traces(), and read_spe().

8.26.2.4 random_from_table()

```
double random_from_table (
            double * xr,
            int nr,
            double randnr )
```

Very fast random number table lookup.
Get a random number from a lookup table as created by make_random_table() for a tabulated probability distribution.
Note that regions with probabilities below 1/nr of the peak probability are not well represented.

Parameters

xr	The lookup table from make_random_table()
nr	The same nr as given to make_random_table()
randnr	A uniform pseudorandom number in [0:1]

Returns

random number according to table

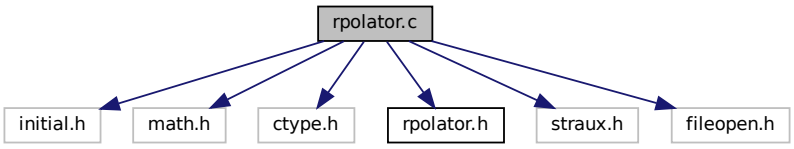
Definition at line 141 of file rndm_table.c.
Referenced by fill_traces(), and read_spe().

8.27 rpolator.c File Reference

Reading of configuration data tables and interpolation.

```
#include "initial.h"
#include <math.h>
#include <ctype.h>
#include "rpolator.h"
#include "straux.h"
#include "fileopen.h"
```

Include dependency graph for rpolator.c:



Data Structures

- struct rpt_list

  Registered interpolation tables allow re-use of tables without having to load them again.

Functions

- static void **strip_comments** (char *line)
- int read_table (const char *fname, int maxrow, double *col1, double *col2)

  Low-level reading of 2-column data tables up to given number of data rows.
- int **read_table2** (const char *fname, int maxrow, double *col1, double *col2)
- int read_table3 (const char *fname, int maxrow, double *col1, double *col2, double *col3)

read_table3() and so on have more columns than read_table but are still only suitable for 1-D interpolation.

- int **read_table4** (const char *fname, int maxrow, double *col1, double *col2, double *col3, double *col4)
- int **read_table5** (const char *fname, int maxrow, double *col1, double *col2, double *col3, double *col4, double *col5)
- int read_table_v (const char *fname, FILE *fptr, size_t *nrow, size_t ncol, double ***col, const size_t *selcol)

  Read tables any length (up to some ridiculous maximum) with the requested columns either in natural order or picking columns as requested.
- struct rpol_table * read_rpol_table (const char *fname, int nd, const char *ymarker, const char *options)

  General function for loading interpolation table combines 1-D and 2-D grid case.
- struct rpol_table * read_rpol1d_table (const char *fn)

  Simplified version for loading a 1-d table with two columns.
- struct rpol_table * read_rpol2d_table (const char *fn, const char *ymarker)

  Simplified version for loading a 2-d table 1+ny columns (ny=available y values in header line).
- struct rpol_table * read_rpol3d_table (const char *fn)

  Simplified version for loading a 2-d table with x/y/z in three columns (x/y intervals rectangular).
- int rpol_is_verbose (void)

  Report what verbosity level is set for the rpolator code.
- int rpol_set_verbose (int lvl)

  Set the verbosity level for the rpolator code, return old level.
- void rpol_info (struct rpol_table *rpt)

  Show information about given interpolation table.
- void rpol_info_lvl (struct rpol_table *rpt, int lvl)

  Report table info at given temporary verbosity level.
- void rpol_free (struct rpol_table *rpt, int removing)

  Free a previously allocated interpolation table data structure.
- void **rpol_check_equi_range** (struct rpol_table *rpt)
- struct rpol_table * simple_rpol1d_table (const char *label, double *x, double *y, int n, int clip)

  A simplified way of setting up a 1-D rpol table for local use, without reading any files.
- static void interp (double x, double *v, int n, int *ipl, double *rpl)

  Linear interpolation with binary search algorithm.
- double rpol (double *x, double *y, int n, double xp)

  Linear interpolation with binary search algorithm.
- double rpol_nearest (double *x, double *y, int n, double xp, int eq, int clip)

  Nearest value (not actually interpolation) in 1-D with with either direct access for equidistant table or with binary search algorithm.
- double rpol_linear (double *x, double *y, int n, double xp, int eq, int clip)

  Linear interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm.
- double rpol_2nd_order (double *x, double *y, int n, double xp, int eq, int clip)

  Second/third order interpolation in 1-D with clipping option outside range.
- CsplinePar * set_1d_cubic_params (double *x, double *y, int n, int clamped)

  Set up cubic spline parameters for n-1 intervals resulting from n data points.
- static double **csx** (double r, const CsplinePar *cp)
- double rpol_cspline (double *x, double *y, const CsplinePar *csp, int n, double xp, int eq, int clip)

  Cubic spline interpolation in 1-D with clipping option outside range.
- double rpol_2d_linear (double *x, double *y, double *z, int nx, int ny, double xp, double yp, int eq, int clip)

  Linear interpolation in 2-D.
- double rpolate_1d (struct rpol_table *rpt, double x, int scheme)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation.
- double rpolate_1d_lin (struct rpol_table *rpt, double x)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation, with linear interpolation scheme hard-wired (independent of any '#@RPOL@' header line).

- double rpolate_2d (struct rpol_table *rpt, double x, double y, int scheme)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 2-D table interpolation.
- double rpolate (struct rpol_table *rpt, double x, double y, int scheme)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) generic for 1-D and 2-D tables.

Variables

- static int **rpol_verbosity** = -1
- static struct rpt_list * **rpt_list_base**

8.27.1 Detailed Description

Reading of configuration data tables and interpolation.
In contrast to the older low-level table reading and rpol() 1-D linear interpolation code, the function here allow flexible switching between 1-D and two different formats of 2-D tables, with support for different interpolation schemes (default being linear).

Author

Konrad Bernloehr

Date

2017, ..., 2023

8.27.2 Function Documentation

8.27.2.1 interp()

```
static void interp (
            double x,
            double * v,
            int n,
            int * ipl,
            double * rpl )  [static]
```

Linear interpolation with binary search algorithm.
Linear interpolation between data point in sorted (i.e. monotonic ascending [no descending yet]) order. This function determines between which two data points the requested coordinate is and where between them. If the given coordinate is outside the covered range, the value for the corresponding edge is returned.
A binary search algorithm is used for fast interpolation.

Parameters

x	Input: the requested coordinate
v	Input: tabulated coordinates at data points
n	Input: number of data points
ipl	Output: the number of the data point following the requested coordinate in the given sorting (1 <= ipl <= n-1)
rpl	Output: the fraction (x-v[ipl-1])/(v[ipl]-v[ipl-1]) with 0 <= rpl <= 1

Definition at line 2225 of file rpolator.c.
References rpol_table::x.
Referenced by rpol(), rpol_2d_linear(), rpol_2nd_order(), rpol_cspline(), rpol_linear(), and rpol_nearest().

8.27.2.2 read_rpol1d_table()

```
struct rpol_table * read_rpol1d_table (
            const char * fn )
```

Simplified version for loading a 1-d table with two columns.
The text free format input file is expected to contain (at least) two columns for x and y values. Any further columns are ignored. Comment and empty lines are ignored as well.
Definition at line 564 of file rpolator.c.
References read_rpol_table().

8.27.2.3 read_rpol2d_table()

```
struct rpol_table * read_rpol2d_table (
            const char * fn,
            const char * ymarker )
```

Simplified version for loading a 2-d table 1+ny columns (ny=available y values in header line).
The text input file is expected to contain a header line where, following the given marker text, the y values to which the following values correspond are listed. The number ny of distinct, ascending-order values in that line defines the number of z expected in the following data lines. The x value is the first value in each data line. If a data line contains more than 1+ny values the extra values are ignored. Comment (except header line) and empty lines are ignored.
Definition at line 582 of file rpolator.c.
References read_rpol_table().

8.27.2.4 read_rpol3d_table()

```
struct rpol_table * read_rpol3d_table (
            const char * fn )
```

Simplified version for loading a 2-d table with x/y/z in three columns (x/y intervals rectangular).
Functionally equivalent to read_rpol2d_table() but the y values are not given just once in a specially marked header line but repeated as the second value in each line. Instead of one header line plus nx data lines with 1+ny values the input to this function is expected to contain nx∗ny lines with three values each. While x and y values do not need to be equidistant, they are required to match a rectangular grid, with the same distinct, ascending-order y values appearing for each x value and the same distinct, ascending-order x values appearing for each y value, eather as x1/y1/z11, x2/y1/z21, ... or x1/y/z11, x1/y2/z12, ...
Definition at line 603 of file rpolator.c.
References read_rpol_table().

8.27.2.5 read_rpol_table()

```
struct rpol_table * read_rpol_table (
            const char * fname,
            int nd,
            const char * ymarker,
            const char * options )
```

General function for loading interpolation table combines 1-D and 2-D grid case.

Parameters

fname	Text input file name
nd	dimension/format parameter with the following possible values: 1: 1-D (2-column) input expected, 2: 2-D (1+ny columns) input with marker indicating special line for y values, 3: 2-D (3 columns) with repeated x and y values (matching rectangular grid), 0: format entirely defined in the data file, requiring the first line to start as '#@RPOL@', -1,-2,-3: If the first line starts as '#@RPOL@', this line defines the format and otherwise it is falling back to format 1, 2, or 3, respectively.
ymarker	The marker indicating the special header line listing the y values with nd=2 only (otherwise ignored).

Definition at line 1119 of file rpolator.c.
Referenced by read_rpol1d_table(), read_rpol2d_table(), and read_rpol3d_table().

8.27.2.6 read_table()

```
int read_table (
            const char * fname,
            int maxrow,
            double * col1,
            double * col2 )
```

Low-level reading of 2-column data tables up to given number of data rows.

Parameters

fname	Name of file to be opened.
maxrow	Maximum number of (non-empty, non-comment) rows of data to read.
col1	Array where values of column 1 are to be copied to.
col2	Array where values of column 2 are to be copied to.

Returns

Number of data rows read (usable values in col1 and col2) or -1 (error).

Definition at line 100 of file rpolator.c.
Referenced by read_shape().

8.27.2.7 read_table_v()

```
int read_table_v (
            const char * fname,
            FILE * fptr,
            size_t * nrow,
            size_t ncol,
            double *** col,
            const size_t * selcol )
```

Read tables any length (up to some ridiculous maximum) with the requested columns either in natural order or picking columns as requested.
All data areas are dynamically allocated (and must be fresh beforehand).
On memory allocation errors a -1 error code is returned but already allocated parts are not released and the file may still be opened.

Parameters

fname	Name or URL of file to read. @paran fptr File pointer if file already open or NULL if fname has to be opened.
nrow	Pointer to number of rows with valid data (pass address of a size_t variable). Input value used to guide initial allocation, not fixing actual rows to read.
ncol	Number of columns of data requested to be read.
col	Pointer to where pointers to column-wise data get allocated. (Need to pass address of a double $**$ variable.)
selcol	NULL for natural column order or pointer to ncol (natural, $>$=1) column numbers in selected order. Example: { 1, 7, 5 } will place data from the first column under ($*$col)[0], data from the seventh column under ($*$col)[1], and data from the fifth column under ($*$col)[2].

Returns

0 (OK), -1 (memory error), -2 (invalid parameter), -3 (asking for too many columns), -4 (too many rows of data, stopped reading).

Definition at line 341 of file rpolator.c.
Referenced by init_atmosphere_from_text_file().

8.27.2.8 rpol()

```
double rpol (
            double * x,
            double * y,
            int n,
            double xp )
```

Linear interpolation with binary search algorithm.
Linear interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. This is the old-style function without any option for equidistant support points or clipping. Note that rpol(px,py,n,xp) is the same as rpol_linear(px,py,n,xp,0,0).
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value

Returns

Interpolated value

Definition at line 2321 of file rpolator.c.
References interp(), rpol_table::x, and rpol_table::y.
Referenced by make_random_table(), random_from_ipol_table(), and read_shape().

8.27.2.9 rpol_2nd_order()

```
double rpol_2nd_order (
            double * x,
            double * y,
            int n,
            double xp,
            int eq,
            int clip )
```

Second/third order interpolation in 1-D with clipping option outside range.
Second to third order interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm. Instead of third order Lagrange interpolation it uses left- and right-sided 2nd order interpolation and a weighted mean between the two variants, rendering it effectively third order except for the intervals next to borders where it degenerates to 2nd order.
Higher-order interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval. Since there is no initialization phase, this is actually slower than the cubic spline interpolation.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Interpolated value

Definition at line 2514 of file rpolator.c.
References rpol_table::dxi, interp(), rpol_linear(), rpol_table::x, and rpol_table::y.

8.27.2.10 rpol_cspline()

```
double rpol_cspline (
            double * x,
            double * y,
            const CsplinePar * csp,
            int n,
            double xp,
            int eq,
            int clip )
```

Cubic spline interpolation in 1-D with clipping option outside range.
Cubic spline interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm. Quadratic interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval. Because of the overhead of calculating the cubic spline parameters, those have to be initialized before the interpolation can be used. Initialisation has to be for either natural or clampled cubic splines.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
csp	Input: Cubic spline parameters (a,b,c,d) for each of n-1 intervals
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Interpolated value

Definition at line 2801 of file rpolator.c.
References interp(), and rpol_linear().
Referenced by read_pulse_shape(), refim1x_(), rhofx_(), and thickx_().

8.27.2.11 rpol_free()

```
void rpol_free (
            struct rpol_table * rpt,
            int removing )
```

Free a previously allocated interpolation table data structure.
This is dangerous to used if the pointer is used in more than one place! Keep in mind that each time you ask to load the same table again you will just get a copy of the same pointer again. If that could be the case you better don't force deleting the structure itself.
Definition at line 796 of file rpolator.c.
References rpol_table::csp, rpol_table::fname, rpol_table::options, rpol_table::use_count, rpol_table::x, rpol_table::y, rpol_table::z, rpol_table::zxmax, and rpol_table::zxmin.

8.27.2.12 rpol_linear()

```
double rpol_linear (
            double * x,
            double * y,
            int n,
            double xp,
            int eq,
            int clip )
```

Linear interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm.
Linear interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Interpolated value

Definition at line 2434 of file rpolator.c.
References rpol_table::dxi, interp(), rpol_table::x, and rpol_table::y.
Referenced by read_pulse_shape(), rpol_2nd_order(), and rpol_cspline().

8.27.2.13 rpol_nearest()

```
double rpol_nearest (
            double * x,
            double * y,
            int n,
            double xp,
            int eq,
            int clip )
```

Nearest value (not actually interpolation) in 1-D with with either direct access for equidistant table or with binary search algorithm.

Take the nearest data point in sorted (i.e. monotonic ascending [no descending yet]) order. The selected value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Nearest value

Definition at line 2358 of file rpolator.c.
References rpol_table::dxi, interp(), rpol_table::x, and rpol_table::y.

8.27.2.14 rpolate()

```
double rpolate (
            struct rpol_table * rpt,
            double x,
            double y,
            int scheme )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) generic for 1-D and 2-D tables.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it can represent either a 1-D or 2-D table.
x	The x coordinate value at which the 1-D or 2-D table is to be interpolated.
y	The y coordinate value at which a 2-D table is to be interpolated (ignored for 1-D). If non-zero for 1-D tables, a warning may be issued.
scheme	Interpolation scheme: 0 ... 4 (not all implemented) for a specific user-defined scheme or -1 (or other values outside of [0:4] range) for the scheme determined when the table was read and allocated.

Definition at line 3221 of file rpolator.c.
Referenced by rpt_qe_ref().

8.27.2.15 rpolate_1d()

```
double rpolate_1d (
            struct rpol_table * rpt,
            double x,
            int scheme )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it should represent a 1-D table.
x	The x coordinate (abscissa) value at which the 1-D table is to be interpolated.
scheme	Interpolation scheme: 0 ... 4 for a specific user-defined scheme or -1 (or other values outside of [0:4] range) for the scheme determined when the table was read and allocated. For 2-D tables for which an upper envelope in x projection was requested a -1 scheme will interpolate in this upper envelope and -2 the lower envelope.

Definition at line 2988 of file rpolator.c.
Referenced by rpolate_1d_lin(), rpolate_2d(), and rpt_qe_ref().

8.27.2.16 rpolate_1d_lin()

```
double rpolate_1d_lin (
            struct rpol_table * rpt,
            double x )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation, with linear interpolation scheme hard-wired (independent of any '#@RPOL@' header line).
This is the most direct equivalence to the older rpol() function.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it should represent a 1-D table.
x	The x coordinate (abscissa) value at which the 1-D table is to be interpolated.

Definition at line 3101 of file rpolator.c.
References rpolate_1d().

8.27.2.17 rpolate_2d()

```
double rpolate_2d (
            struct rpol_table * rpt,
            double x,
            double y,
            int scheme )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 2-D table interpolation.
Fall-back to 1-D only after issueing a warning.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it should represent a 2-D table.
x	The x coordinate value at which the 2-D table is to be interpolated.
y	The y coordinate value at which the 2-D table is to be interpolated (ignored for 1-D).
scheme	Interpolation scheme: 0 ... 4 (not all implemented) for a specific user-defined scheme or -1 (or other values outside of [0:4] range) for the scheme determined when the table was read and allocated.

Definition at line 3124 of file rpolator.c.
References rpol_table::fname, rpol_table::ndim, and rpolate_1d().

8.27.2.18 set_1d_cubic_params()

```
CsplinePar* set_1d_cubic_params (
            double * x,
            double * y,
            int n,
            int clamped )
```

Set up cubic spline parameters for n-1 intervals resulting from n data points.
The resulting cubic spline can either be a 'natural' one (second derivative is zero at the edges) or a clamped one (first derivative is fixed, currently to zero, at the edges).

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
clamped	Input: 0 (natural cubic spline) or 1 (clamped cubic spline)

Returns

Allocated array of four parameters each defining the third order polynomial in each interval. In case of invalid input parameters a NULL pointer is returned.

Definition at line 2673 of file rpolator.c.
References cubic_params::d, and rpol_table::x.
Referenced by init_common_atmosphere(), and read_pulse_shape().

8.27.2.19 simple_rpol1d_table()

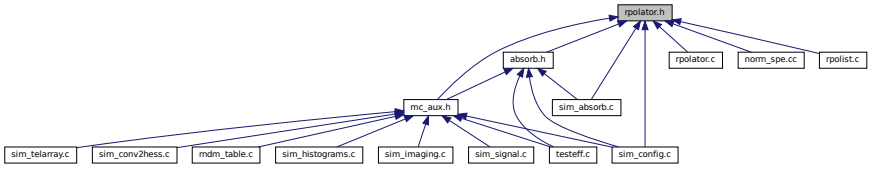
```
struct rpol_table* simple_rpol1d_table (
            const char * label,
            double * x,
            double * y,
            int n,
            int clip )
```

A simplified way of setting up a 1-D rpol table for local use, without reading any files.
The returned pointer is also not hooked into the global linked list, thus is supposed to be safe to be removed after use.
Definition at line 2155 of file rpolator.c.
References rpol_table::clipping, rpol_table::fname, rpol_table::ndim, rpol_table::ny, rpol_table::scheme, rpol_↩table::use_count, rpol_table::x, rpol_table::y, and rpol_table::z.

8.28 rpolator.h File Reference

Memory structure and interfaces for rpolator interpolation code.

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



Data Structures

- struct cubic_params

  Cubic spline interpolation (natural cubic splines = scheme 3, clampled cubic splines = scheme 4)
- struct rpol_table

  Structure describing an interpolation table, interpolation scheme and selected options.

Macros

- #define **WITH_RPOLATOR** 1

Typedefs

- typedef struct cubic_params **CsplinePar**
- typedef struct rpol_table **RpolTable**

Functions

- int read_table (const char ∗fname, int maxrow, double ∗col1, double ∗col2)

  Low-level reading of 2-column data tables up to given number of data rows.
- int **read_table2** (const char ∗fname, int maxrow, double ∗col1, double ∗col2)
- int read_table3 (const char ∗fname, int maxrow, double ∗col1, double ∗col2, double ∗col3)

  read_table3() and so on have more columns than read_table but are still only suitable for 1-D interpolation.
- int **read_table4** (const char ∗fname, int maxrow, double ∗col1, double ∗col2, double ∗col3, double ∗col4)
- int **read_table5** (const char ∗fname, int maxrow, double ∗col1, double ∗col2, double ∗col3, double ∗col4, double ∗col5)
- int read_table_v (const char ∗fname, FILE ∗fptr, size_t ∗nrow, size_t ncol, double ∗∗∗col, const size_t ∗selcol)

  Read tables any length (up to some ridiculous maximum) with the requested columns either in natural order or picking columns as requested.
- struct rpol_table ∗ read_rpol1d_table (const char ∗fn)

  Simplified version for loading a 1-d table with two columns.
- struct rpol_table ∗ read_rpol2d_table (const char ∗fn, const char ∗ymarker)

  Simplified version for loading a 2-d table 1+ny columns (ny=available y values in header line).
- struct rpol_table ∗ read_rpol3d_table (const char ∗fn)

  Simplified version for loading a 2-d table with x/y/z in three columns (x/y intervals rectangular).
- struct rpol_table ∗ read_rpol_table (const char ∗fname, int nd, const char ∗ymarker, const char ∗options)

  General function for loading interpolation table combines 1-D and 2-D grid case.
- struct rpol_table ∗ simple_rpol1d_table (const char ∗label, double ∗x, double ∗y, int n, int clip)

  A simplified way of setting up a 1-D rpol table for local use, without reading any files.
- int rpol_is_verbose (void)

  Report what verbosity level is set for the rpolator code.
- int rpol_set_verbose (int lvl)

  Set the verbosity level for the rpolator code, return old level.

- void rpol_info (struct rpol_table *rpt)

  Show information about given interpolation table.
- void rpol_info_lvl (struct rpol_table *rpt, int lvl)

  Report table info at given temporary verbosity level.
- void rpol_free (struct rpol_table *rpt, int removing)

  Free a previously allocated interpolation table data structure.
- void **rpol_check_equi_range** (struct rpol_table *rpt)
- double rpol (double *x, double *y, int n, double xp)

  Linear interpolation with binary search algorithm.
- double rpol_nearest (double *x, double *y, int n, double xp, int eq, int clip)

  Nearest value (not actually interpolation) in 1-D with with either direct access for equidistant table or with binary search algorithm.
- double rpol_linear (double *x, double *y, int n, double xp, int eq, int clip)

  Linear interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm.
- double rpol_2nd_order (double *x, double *y, int n, double xp, int eq, int clip)

  Second/third order interpolation in 1-D with clipping option outside range.
- CsplinePar * set_1d_cubic_params (double *x, double *y, int n, int clamped)

  Set up cubic spline parameters for n-1 intervals resulting from n data points.
- double rpol_cspline (double *x, double *y, const CsplinePar *csp, int n, double xp, int eq, int clip)

  Cubic spline interpolation in 1-D with clipping option outside range.
- double rpol_2d_linear (double *x, double *y, double *z, int nx, int ny, double xp, double yp, int eq, int clip)

  Linear interpolation in 2-D.
- double rpolate_1d (struct rpol_table *rpt, double x, int scheme)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation.
- double rpolate_1d_lin (struct rpol_table *rpt, double x)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation, with linear interpolation scheme hard-wired (independent of any '#@RPOL@' header line).
- double rpolate_2d (struct rpol_table *rpt, double x, double y, int scheme)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 2-D table interpolation.
- double rpolate (struct rpol_table *rpt, double x, double y, int scheme)

  High-level interpolation function (user code only has to keep a pointer to the allocated object) generic for 1-D and 2-D tables.

8.28.1 Detailed Description

Memory structure and interfaces for rpolator interpolation code.

Author

Konrad Bernloehr

Date

2017 ... 2019

8.28.2 Function Documentation

8.28.2.1 read_rpol1d_table()

```
struct rpol_table* read_rpol1d_table (
            const char * fn )
```

Simplified version for loading a 1-d table with two columns.
The text free format input file is expected to contain (at least) two columns for x and y values. Any further columns are ignored. Comment and empty lines are ignored as well.
Definition at line 564 of file rpolator.c.
References read_rpol_table().

8.28.2.2 read_rpol2d_table()

```
struct rpol_table* read_rpol2d_table (
            const char * fn,
            const char * ymarker )
```

Simplified version for loading a 2-d table 1+ny columns (ny=available y values in header line).
The text input file is expected to contain a header line where, following the given marker text, the y values to which the following values correspond are listed. The number ny of distinct, ascending-order values in that line defines the number of z expected in the following data lines. The x value is the first value in each data line. If a data line contains more than 1+ny values the extra values are ignored. Comment (except header line) and empty lines are ignored.
Definition at line 582 of file rpolator.c.
References read_rpol_table().

8.28.2.3 read_rpol3d_table()

```
struct rpol_table* read_rpol3d_table (
            const char * fn )
```

Simplified version for loading a 2-d table with x/y/z in three columns (x/y intervals rectangular).
Functionally equivalent to read_rpol2d_table() but the y values are not given just once in a specially marked header line but repeated as the second value in each line. Instead of one header line plus nx data lines with 1+ny values the input to this function is expected to contain nx∗ny lines with three values each. While x and y values do not need to be equidistant, they are required to match a rectangular grid, with the same distinct, ascending-order y values appearing for each x value and the same distinct, ascending-order x values appearing for each y value, eather as x1/y1/z11, x2/y1/z21, ... or x1/y/z11, x1/y2/z12, ...
Definition at line 603 of file rpolator.c.
References read_rpol_table().

8.28.2.4 read_rpol_table()

```
struct rpol_table* read_rpol_table (
            const char * fname,
            int nd,
            const char * ymarker,
            const char * options )
```

General function for loading interpolation table combines 1-D and 2-D grid case.

Parameters

fname	Text input file name
nd	dimension/format parameter with the following possible values: 1: 1-D (2-column) input expected, 2: 2-D (1+ny columns) input with marker indicating special line for y values, 3: 2-D (3 columns) with repeated x and y values (matching rectangular grid), 0: format entirely defined in the data file, requiring the first line to start as '#@RPOL@', -1,-2,-3: If the first line starts as '#@RPOL@', this line defines the format and otherwise it is falling back to format 1, 2, or 3, respectively.
ymarker	The marker indicating the special header line listing the y values with nd=2 only (otherwise ignored).

Definition at line 1119 of file rpolator.c.
Referenced by read_rpol1d_table(), read_rpol2d_table(), and read_rpol3d_table().

8.28.2.5 read_table()

```
int read_table (
            const char * fname,
            int maxrow,
            double * col1,
            double * col2 )
```

Low-level reading of 2-column data tables up to given number of data rows.

Parameters

fname	Name of file to be opened.
maxrow	Maximum number of (non-empty, non-comment) rows of data to read.
col1	Array where values of column 1 are to be copied to.
col2	Array where values of column 2 are to be copied to.

Returns

Number of data rows read (usable values in col1 and col2) or -1 (error).

Definition at line 100 of file rpolator.c.

8.28.2.6 read_table_v()

```
int read_table_v (
            const char * fname,
            FILE * fptr,
            size_t * nrow,
            size_t ncol,
            double *** col,
            const size_t * selcol )
```

Read tables any length (up to some ridiculous maximum) with the requested columns either in natural order or picking columns as requested.
All data areas are dynamically allocated (and must be fresh beforehand).
On memory allocation errors a -1 error code is returned but already allocated parts are not released and the file may still be opened.

Parameters

fname	Name or URL of file to read. @paran fptr File pointer if file already open or NULL if fname has to be opened.
nrow	Pointer to number of rows with valid data (pass address of a size_t variable). Input value used to guide initial allocation, not fixing actual rows to read.
ncol	Number of columns of data requested to be read.
col	Pointer to where pointers to column-wise data get allocated. (Need to pass address of a double $**$ variable.)
selcol	NULL for natural column order or pointer to ncol (natural, $>$=1) column numbers in selected order. Example: { 1, 7, 5 } will place data from the first column under ($*$col)[0], data from the seventh column under ($*$col)[1], and data from the fifth column under ($*$col)[2].

Returns

0 (OK), -1 (memory error), -2 (invalid parameter), -3 (asking for too many columns), -4 (too many rows of data, stopped reading).

Definition at line 341 of file rpolator.c.
Referenced by init_atmosphere_from_text_file().

8.28.2.7 rpol()

```
double rpol (
             double * x,
             double * y,
             int n,
             double xp )
```

Linear interpolation with binary search algorithm.
Linear interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. This is the old-style function without any option for equidistant support points or clipping. Note that rpol(px,py,n,xp) is the same as rpol_linear(px,py,n,xp,0,0).
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value

Returns

Interpolated value

Definition at line 2321 of file rpolator.c.
References interp(), rpol_table::x, and rpol_table::y.
Referenced by make_random_table(), random_from_ipol_table(), and read_shape().

8.28.2.8 rpol_2nd_order()

```
double rpol_2nd_order (
             double * x,
             double * y,
             int n,
             double xp,
             int eq,
             int clip )
```

Second/third order interpolation in 1-D with clipping option outside range.
Second to third order interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm. Instead of third order Lagrange interpolation it uses left- and right-sided 2nd order interpolation and a weighted mean between the two variants, rendering it effectively third order except for the intervals next to borders where it degenerates to 2nd order.
Higher-order interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval. Since there is no initialization phase, this is actually slower than the cubic spline interpolation.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Interpolated value

Definition at line 2514 of file rpolator.c.
References rpol_table::dxi, interp(), rpol_linear(), rpol_table::x, and rpol_table::y.

8.28.2.9 rpol_cspline()

```
double rpol_cspline (
            double * x,
            double * y,
            const CsplinePar * csp,
            int n,
            double xp,
            int eq,
            int clip )
```

Cubic spline interpolation in 1-D with clipping option outside range.
Cubic spline interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm. Quadratic interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval. Because of the overhead of calculating the cubic spline parameters, those have to be initialized before the interpolation can be used. Initialisation has to be for either natural or clampled cubic splines.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
csp	Input: Cubic spline parameters (a,b,c,d) for each of n-1 intervals
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Interpolated value

Definition at line 2801 of file rpolator.c.
References interp(), and rpol_linear().
Referenced by read_pulse_shape(), refim1x_(), rhofx_(), and thickx_().

8.28.2.10 rpol_free()

```
void rpol_free (
            struct rpol_table * rpt,
            int removing )
```

Free a previously allocated interpolation table data structure.
This is dangerous to used if the pointer is used in more than one place! Keep in mind that each time you ask to load the same table again you will just get a copy of the same pointer again. If that could be the case you better don't force deleting the structure itself.
Definition at line 796 of file rpolator.c.
References rpol_table::csp, rpol_table::fname, rpol_table::options, rpol_table::use_count, rpol_table::x, rpol_table::y, rpol_table::z, rpol_table::zxmax, and rpol_table::zxmin.

8.28.2.11 rpol_linear()

```
double rpol_linear (
            double * x,
            double * y,
            int n,
            double xp,
            int eq,
            int clip )
```

Linear interpolation in 1-D with with either direct access for equidistant table or with binary search algorithm.
Linear interpolation between data point in sorted (i.e. monotonic ascending or descending) order. The resulting interpolated value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Interpolated value

Definition at line 2434 of file rpolator.c.
References rpol_table::dxi, interp(), rpol_table::x, and rpol_table::y.
Referenced by read_pulse_shape(), rpol_2nd_order(), and rpol_cspline().

8.28.2.12 rpol_nearest()

```
double rpol_nearest (
            double * x,
            double * y,
            int n,
            double xp,
            int eq,
            int clip )
```

Nearest value (not actually interpolation) in 1-D with with either direct access for equidistant table or with binary search algorithm.

Take the nearest data point in sorted (i.e. monotonic ascending [no descending yet]) order. The selected value is returned as a return value. If the table is known to be provided at equidistant supporting points, direct access is preferred. Otherwise a binary search algorithm is used to find the proper interval.
This function calls interp() to find out where to interpolate.

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
xp	Input: Coordinate of requested value
eq	Input: If non-zero: table is at equidistant points.
clip	Input: Zero: no clipping; extrapolate with left/right edge value outside range. Non-zero: clip at edges; return 0. outside supported range.

Returns

Nearest value

Definition at line 2358 of file rpolator.c.
References rpol_table::dxi, interp(), rpol_table::x, and rpol_table::y.

8.28.2.13 rpolate()

```
double rpolate (
            struct rpol_table * rpt,
            double x,
            double y,
            int scheme )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) generic for 1-D and 2-D tables.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it can represent either a 1-D or 2-D table.
x	The x coordinate value at which the 1-D or 2-D table is to be interpolated.
y	The y coordinate value at which a 2-D table is to be interpolated (ignored for 1-D). If non-zero for 1-D tables, a warning may be issued.
scheme	Interpolation scheme: 0 ... 4 (not all implemented) for a specific user-defined scheme or -1 (or other values outside of [0:4] range) for the scheme determined when the table was read and allocated.

Definition at line 3221 of file rpolator.c.
Referenced by rpt_qe_ref().

8.28.2.14 rpolate_1d()

```
double rpolate_1d (
            struct rpol_table * rpt,
            double x,
            int scheme )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it should represent a 1-D table.
x	The x coordinate (abscissa) value at which the 1-D table is to be interpolated.
scheme	Interpolation scheme: 0 ... 4 for a specific user-defined scheme or -1 (or other values outside of [0:4] range) for the scheme determined when the table was read and allocated. For 2-D tables for which an upper envelope in x projection was requested a -1 scheme will interpolate in this upper envelope and -2 the lower envelope.

Definition at line 2988 of file rpolator.c.
Referenced by rpolate_1d_lin(), rpolate_2d(), and rpt_qe_ref().

8.28.2.15 rpolate_1d_lin()

```
double rpolate_1d_lin (
            struct rpol_table * rpt,
            double x )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 1-D table interpolation, with linear interpolation scheme hard-wired (independent of any '#@RPOL@' header line).
This is the most direct equivalence to the older rpol() function.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it should represent a 1-D table.
x	The x coordinate (abscissa) value at which the 1-D table is to be interpolated.

Definition at line 3101 of file rpolator.c.
References rpolate_1d().

8.28.2.16 rpolate_2d()

```
double rpolate_2d (
            struct rpol_table * rpt,
            double x,
            double y,
            int scheme )
```

High-level interpolation function (user code only has to keep a pointer to the allocated object) limited to 2-D table interpolation.
Fall-back to 1-D only after issueing a warning.

Parameters

rpt	Pointer to interpolation table structure, previously set up with read_rpol_table. Keep in mind that it gets only allocated once and, if you want to free it, you should not free it more than once. In the case of this function, it should represent a 2-D table.
x	The x coordinate value at which the 2-D table is to be interpolated.
y	The y coordinate value at which the 2-D table is to be interpolated (ignored for 1-D).
scheme	Interpolation scheme: 0 ... 4 (not all implemented) for a specific user-defined scheme or -1 (or other values outside of [0:4] range) for the scheme determined when the table was read and allocated.

Definition at line 3124 of file rpolator.c.
References rpol_table::fname, rpol_table::ndim, and rpolate_1d().

8.28.2.17 set_1d_cubic_params()

```
CsplinePar* set_1d_cubic_params (
            double * x,
            double * y,
            int n,
            int clamped )
```

Set up cubic spline parameters for n-1 intervals resulting from n data points.
The resulting cubic spline can either be a 'natural' one (second derivative is zero at the edges) or a clamped one (first derivative is fixed, currently to zero, at the edges).

Parameters

x	Input: Coordinates for data table
y	Input: Corresponding values for data table
n	Input: Number of data points
clamped	Input: 0 (natural cubic spline) or 1 (clamped cubic spline)

Returns

Allocated array of four parameters each defining the third order polynomial in each interval. In case of invalid input parameters a NULL pointer is returned.

Definition at line 2673 of file rpolator.c.
References cubic_params::d, and rpol_table::x.
Referenced by init_common_atmosphere(), and read_pulse_shape().

8.28.2.18 simple_rpol1d_table()

```
struct rpol_table* simple_rpol1d_table (
            const char * label,
            double * x,
            double * y,
            int n,
            int clip )
```

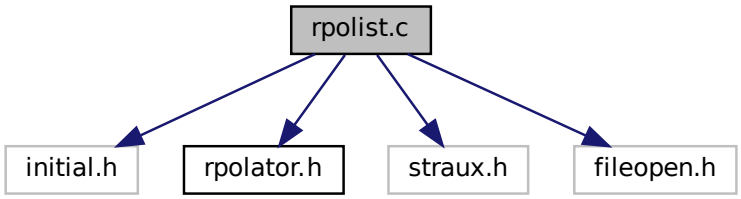
A simplified way of setting up a 1-D rpol table for local use, without reading any files.
The returned pointer is also not hooked into the global linked list, thus is supposed to be safe to be removed after use.
Definition at line 2155 of file rpolator.c.
References rpol_table::clipping, rpol_table::fname, rpol_table::ndim, rpol_table::ny, rpol_table::scheme, rpol_↩table::use_count, rpol_table::x, rpol_table::y, and rpol_table::z.

8.29 rpolist.c File Reference

List rpolator-suitable data tables in uniform format.

```
#include "initial.h"
#include "rpolator.h"
#include "straux.h"
#include "fileopen.h"
```

Include dependency graph for rpolist.c:



Functions

- void **printz** (double z, int lopt)
- void **syntax** (const char ∗opt)
- double **rpolate_lim** (struct rpol_table ∗rpt, double x, double y, int scheme, double ∗lim)
- int **main** (int argc, char ∗∗argv)

8.29.1 Detailed Description

List rpolator-suitable data tables in uniform format.
Tables (as text files) suitable for the rpolator interpolation code can come in different formats and not all are immediately suitable for foreign visualization/plotting tools. Therefore this tool converts them into a uniform 'x z' or 'x y z' format. The tool has additional features for showing table properties.

Author

Konrad Bernloehr

Date

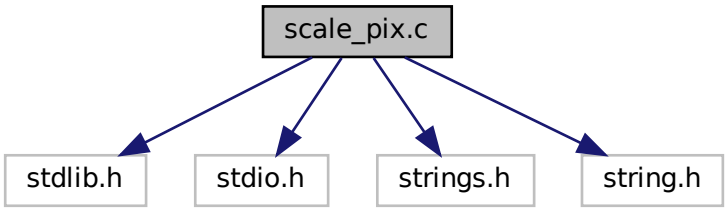
2017 ... 2022

8.30 scale_pix.c File Reference

Scale pixel spacing in sim_telarray camera config file but does not change PixType line, i.e.

```
#include <stdlib.h>
#include <stdio.h>
#include <strings.h>
#include <string.h>
```

Include dependency graph for scale_pix.c:



Functions

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

8.30.1 Detailed Description

Scale pixel spacing in sim_telarray camera config file but does not change PixType line, i.e. the size of the pixel itself.

Author

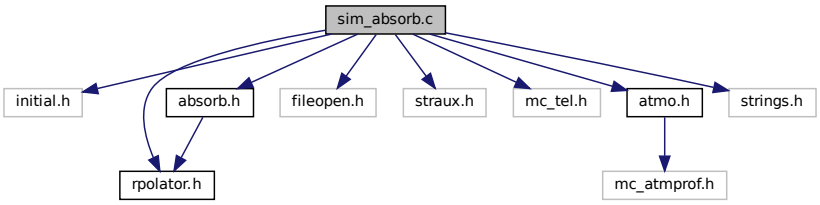
Konrad Bernloehr

8.31 sim_absorb.c File Reference

Atmospheric absorption and scattering for Cherenkov light.

```
#include "initial.h"
#include "rpolator.h"
#include "absorb.h"
#include "fileopen.h"
#include "straux.h"
#include "mc_tel.h"
#include "atmo.h"
#include <strings.h>
```

Include dependency graph for sim_absorb.c:



Macros

- #define **CONST**
- #define **CSPLINE_OPT** ""
- #define MAX_TRANS_HEIGHT 100

  The maximum number of height columns in the transmission table.
- #define MAX_WL 1200 /∗ From 200 to 1200 nm in 1 nm step perhaps ∗/

  The maximum number of wavelength steps in the tables.

Functions

- double height_for_transmission ()

  Report back the actual height level for the atmospheric transmission table.
- int read_trans (const char ∗setup_trans_fname, double altitude)

  Read table of atmospheric transmission data.
- int read_qe_ref (const char ∗qe_fname, const char ∗mirror_ref_fname, const char ∗mirror2_ref_fname, double ∗quantum_efficiency, double ∗optics_efficiency, int mirror_class, int bypass_optics, int max_lambda)

  Read quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers.
- int rpt_qe_ref (RpolTable ∗rpt_qe, RpolTable ∗rpt_ref, RpolTable ∗rpt_ref2, double ∗quantum_efficiency, double ∗optics_efficiency, int mirror_class, int bypass_optics, int max_lambda, double mirror_degraded, double mirror2_degraded)

  Obtain quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers, similar to read_qe_ref() but from previously read rpolator tables.
- double atmospheric_transmission (int iwl, double zem, double airmass)

  Find out direct atmospheric transmission probability between point of emission and point of detection assumed at the nominal altitude for the transmission table.
- double atmospheric_transmission_x (int iwl, double zem, double airmass)

  Same function as atmospheric_transmission() but with explanations.
- double atmospheric_transmission2 (int iwl, double zem, double airmass, double od2tel, double od2focus)

  Find out direct atmospheric transmission probability between point of emission and point of detection.

Variables

- static double h2

  The bottom level, in km a.s.l, as indicated in the transmission table.
- static double h1 [MAX_TRANS_HEIGHT]

  Corresponding starting altitudes for table columns [km], in increasing order.
- static double logh1cm [MAX_TRANS_HEIGHT]

  Logarithm of h1 values.
- static int **nh1**
- static double **trans** [MAX_WL][MAX_TRANS_HEIGHT]

- static int **wl** [MAX_WL]
- static int **nwl**
- static int **trans_ok**

8.31.1 Detailed Description

Atmospheric absorption and scattering for Cherenkov light.
This file contains the code to determine the amount of direct atmospheric absorption and scattering (i.e. excluding multiple scattering) for Cherenkov light from any given height. A plane-parallel atmosphere is assumed.

Author

Konrad Bernloehr

Date

1997 ... 2023

8.31.2 Function Documentation

8.31.2.1 atmospheric_transmission()

```
double atmospheric_transmission (
            int iwl,
            double zem,
            double airmass )
```

Find out direct atmospheric transmission probability between point of emission and point of detection assumed at the nominal altitude for the transmission table.
This function is not suitable for horizontal paths.

Parameters

iwl	Integer step wavelength [nm]
zem	Emission altitude a.s.l. [cm]
airmass	Multiplication factor of traversed atmospheric column due to inclined path. Usually 1/cos(zenith angle).

Returns

Atmospheric transmission probability (between 0 and 1).

Definition at line 634 of file sim_absorb.c.

8.31.2.2 atmospheric_transmission2()

```
double atmospheric_transmission2 (
            int iwl,
            double zem,
            double airmass,
            double od2tel,
            double od2focus )
```

Find out direct atmospheric transmission probability between point of emission and point of detection.
In contrast to atmospheric_transmission() we also include the effects of the telescope altitude above the observation level (or actually the nominal altitude of the transmission table which is supposed to match the height of the CORSIKA observation level) plus an effective pathlength through the instrument. For efficiency reasons, the corresponding optical depths have to be prepared beforehand rather than running the same interpolation again and again. Placement of optical elements of the telescope and actual optical paths are considered too much detail. This function is not suitable for horizontal paths.

Parameters

iwl	Integer step wavelength [nm]
zem	Emission altitude a.s.l. [cm]
airmass	Multiplication factor of traversed atmospheric column due to inclined path. Usually 1/cos(zenith angle).
od2tel	Optical depth between telescope altitude and the nominal transmission table level at the wavelength in question, for a vertical path (to be multiplied with airmass as needed).
od2focus	Optical depth between the telescope altitude and an altitude higher by the effective pathlength through the instrument (a little more than the nominal focal length for 1M but less than that for 2M telescope types), at the given wavelength, with no airmass correction to be applied.

Returns

Atmospheric transmission probability (between 0 and 1).

Definition at line 742 of file sim_absorb.c.

8.31.2.3 read_qe_ref()

```
int read_qe_ref (
            const char * qe_fname,
            const char * mirror_ref_fname,
            const char * mirror2_ref_fname,
            double * quantum_efficiency,
            double * optics_efficiency,
            int mirror_class,
            int bypass_optics,
            int max_lambda )
```

Read quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers.

Parameters

qe_fname	File name of table with quantum efficiency or PDE.
mirror_ref_fname	File name of mirror reflectivity.
mirror2_ref_fname	File name of secondary mirror reflectivity.
quantum_efficiency	Array for resulting combined efficiency of QE (or PDE) and any reflectivity involved (depending on mirror class and bypass_optics), guaranteed to be at least of size max_lambda (for wavelengths in nanometers).
optics_efficiency	Array for resulting reflectivity which is also applied as part of quantum_efficiency, just to be able to disentangle them where needed. Can be NULL if not used.
mirror_class	Mirror class (0: segmented single reflector made of spherical mirror tiles, 1: single parabolic reflector, 2: dual mirror)
bypass_optics	Option to bypass part or all of the optics in ray-tracing (0: no bypassing, 1: bypass primay, 2: bypass all)
max_lambda	Guaranteed size of quantum_efficiency array and optics_effieinces, i.e. limit of wavelengths (in nanometers) covered.

Returns

0 (OK), -1 (error)

Definition at line 268 of file sim_absorb.c.

8.31.2.4 rpt_qe_ref()

```
int rpt_qe_ref (
            RpolTable * rpt_qe,
            RpolTable * rpt_ref,
            RpolTable * rpt_ref2,
            double * quantum_efficiency,
            double * optics_efficiency,
            int mirror_class,
            int bypass_optics,
            int max_lambda,
            double mirror_degraded,
            double mirror2_degraded )
```

Obtain quantum efficiency curve and mirror reflectivity curves as functions of wavelength in nanometers, similar to read_qe_ref() but from previously read rpolator tables.
If any of the reflectivity tables is a 2-D table, it will count the maximum over all incidence angles.

Parameters

rpt_qe	Rpolator table with quantum efficiency or PDE.
rpt_ref	Rpolator table with mirror reflectivity.
rpt_ref2	Rpolator table with secondary mirror reflectivity (for mirror_class 2 only).
quantum_efficiency	Array for resulting combined efficiency of QE (or PDE) and any reflectivity involved (depending on mirror class and bypass_optics), guaranteed to be at least of size max_lambda (for wavelengths in nanometers). Where 2-D tables are involved this is the maximum for any angle.
optics_efficiency	Array for resulting reflectivity (or product of reflectivities for mirror_class=2) which is also applied as part of quantum_efficiency, just to be able to disentangle them where needed. Where 2-D tables are involved this is the maximum for any angle. Can be NULL if not used.
mirror_class	Mirror class (0: segmented single reflector made of spherical mirror tiles, 1: single parabolic reflector, 2: dual mirror)
bypass_optics	Option to bypass part or all of the optics in ray-tracing (0: no bypassing, 1: bypass primay, 2: bypass all)
max_lambda	Guaranteed size of quantum_efficiency array and optics_effieinces, i.e. limit of wavelengths (in nanometers) covered.
mirror_degraded	Optical efficiency degradation. In case of any bypassing of the optics, this is specifically for the primary mirror (or the lens in case of Fresnel optics). Ignored for bypass_optics >= 1.
mirror2_degraded	Optical efficiency degradation of secondary mirror. Only for dual-mirror optics. Ignored for bypass_optics == 2.

Returns

0 (OK), -1 (error)

Definition at line 459 of file sim_absorb.c.
References rpol_table::ndim, rpolate(), rpolate_1d(), and rpol_table::scheme.

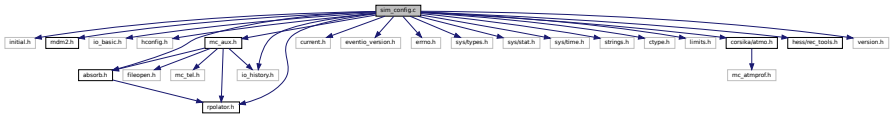
8.32 sim_config.c File Reference

Configure all the things needed for the simulation.

```
#include "initial.h"
#include "rndm2.h"
#include "io_basic.h"
#include "hconfig.h"
#include "mc_aux.h"
```

```
#include "current.h"
#include "io_history.h"
#include "eventio_version.h"
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <strings.h>
#include <ctype.h>
#include <limits.h>
#include "rpolator.h"
#include "absorb.h"
#include "corsika/atmo.h"
#include "hess/rec_tools.h"
#include "version.h"
```

Include dependency graph for sim_config.c:



Data Structures

- struct imaging_setup

  The setup of telescope specific things.
- struct mpl_wordlist
- struct cmdline_conf

  Configuration from command line ('-C' or '-W' option) is saved for re-use.

Macros

- #define **square**(x) ((x)∗(x))
- #define **MAXFN** 1024
- #define MAX_DEAD_PIXELS 20

  Maximum number of pixels which may be dead.
- #define STAR_PHOTONS_STRING "4000"

  The number of photons thrown on a telescope per star as a string.
- #define **MAX_STARS** 100

Typedefs

- typedef struct mpl_wordlist **MplWordList**

Functions

- void **strip_comments** (char ∗line)
- static int **metaparam_config_handler** (const char ∗op, CONFIG_VALUES ∗cfg_val)
- static int **set_tel_altitude** (const char ∗alt, CONFIG_VALUES ∗cfg_val)
- static int save_or_restore_all_channels (struct camera_electronics ∗electronics, int ntel, const char ∗fname)

  Save or restore parameters of all the electronics channels.
- int assign_array_trigger (struct telescope_array ∗array, char ∗fname, int default_min_tel)

  Used if different multiplicities apply to different array subsets.
- static int set_tel_altitude (const char ∗alt, CONFIG_VALUES __attribute__((unused)) ∗cfg_val)

Set the telescope altitude angle as such instead of the zenith angle.

- static int mpl_add (MplWordList *mpl, const char *text)

  Internal helper function for metaparam_config_handler to add a parameter name to either set of name lists.
- static int mpl_set (MplWordList *mpl, const char *text)

  Internal helper function for metaparam_config_handler to set a non-configuration metaparameter by hand.
- static int mpl_remove (MplWordList *mpl, const char *text)

  Internal helper function for metaparam_config_handler to remove a parameter name from either set of name lists.
- static int mpl_clear (MplWordList *mpl)

  Internal helper function for metaparam_config_handler to all entries from either set of name lists.
- static void mpl_show (MplWordList *mpl)

  Internal helper function for metaparam_config_handler to show all entries in either set of name lists.
- static int mpl_fill (int itel, int tel_id)

  Fills a MetaParamList (either global or telescope-specific) with names and current values.
- MetaParamList * metapar_deliver (int itel)

  Deliver the locally maintained set of metaparameters to the external code.
- static int metaparam_config_handler (const char *op, CONFIG_VALUES __attribute__((unused)) *cfg_val)

  Manage the list of metaparameters to be stored in the data.
- void config_setup_imaging ()

  Interface used for configuration of imaging and electronics.
- void config_setup_atm_transmission ()

  Interface required for atmospheric transmission configuration.
- void add_to_cmdline_config (char *conf, struct cmdline_conf *cf)

  Configuration from command line ('-C' or '-W' option) is saved for re-use.
- int cmdline_config (struct cmdline_conf *cf)

  Saved configuration from command line is applied again to override config file.
- int reconfig_include (char *conf)

  Configuration files may include other configuration files.
- int count_cmdline_config (struct cmdline_conf *cf)

  Count the number of saved configuration parameters.
- static void **cmdl_syntax** (void)
- static int do_config (int argc, char **argv)

  Setup of default configuration and interpretation of command line.
- static int setup_starlight (char *fname, double *star_light, int npix, struct telescope_optics *optics, struct pm_camera *cam, double trans)

  Add light from individual stars.
- static int **setup_sky_map** (RpolTable *nsb_sky_map, double *skymap_fact, int npix, struct telescope_optics *optics, struct pm_camera *cam)
- int **nearest_int** (double d)
- static void kill_pixel (struct camera_electronics *electronics, int j)

  Set a given pixel as dead, resulting in no signal or trigger from incoming light and no DC current.
- int get_random_seed_from_file (char *want_seed, size_t srs)

  Read a list of pre-generated seeds for the random generator and chose a way to pick one of these seeds.
- int init_setup (int argc, char **argv, struct telescope_array *array, struct telescope_optics *tel_optics, struct pm_camera *camera, struct camera_electronics *electronics, struct mc_options *options)

  Initialize the whole setup and the different pieces of the simulation.
- int convergence_correction (struct telescope_array *array)

  Correct the telescope pointing for convergent viewing.
- static void rnd_vw_dir (double *azimuth_out, double *altitude_out, double azimuth_in, double altitude_in, double rand_phi, double rand_rad)

  Used from randomize_viewing_direction()
- int randomize_viewing_direction (struct telescope_array *array)

  Changing viewing directions from shower to shower.

Variables

- double airlightspeed

  The speed of light used for light propagation at the observation level.
- int **global_verbose**
- static int **cfg_verbose** = 0
- static int max_used_telescopes

  The maximum number of telescopes used.
- static struct imaging_setup setup

  Used again for each telescope.
- static char spe_fname [MAXFN]

  File name for single p.e. response.
- static char fadc_pulse_fname [MAXFN]

  File name for (F)ADC pulse shape.
- static char disc_pulse_fname [MAXFN]

  Same for shape at discriminator/comparator.
- static char mirror_ref_fname [MAXFN]

  Mirror reflectivity (wavelength dependent).
- static char mirror2_ref_fname [MAXFN]

  Mirror reflectivity (wavelength dependent) of secondary mirror.
- static char mirror_list_fname [MAXFN]

  List of mirror positions (not used with flexible camera).
- static char mirror1_degraded_map [MAXFN]

  File with interpolation table mapping reflection degradation on primary.
- static char mirror2_degraded_map [MAXFN]

  File with interpolation table mapping reflection degradation on secondary.
- static char camera_degraded_map [MAXFN]

  File with interpolation table mapping camera efficiency degradation.
- static char qe_fname [MAXFN]

  File name for quantum efficiency curve.
- static char channel_save_restore_fname [MAXFN]

  File to save channel data.
- static char array_trigger_fname [MAXFN]

  File with more specific array trigger than n-fold of all.
- static double array_trigger_window

  The time window within which central trigger signals must be seen, after correction for nominal delay [ns].
- static int always_with_aweight

  Is 1 if area weights should always be recorded.
- static char setup_trans_fname [MAXFN]

  Atmospheric transmission file.
- static char input_fname_list [10240]

  List of input files from command line.
- static char input_fname [MAXFN]

  Single input file to be read.
- static char **plot_fname** [MAXFN]
- static char histogram_fname [MAXFN]

  File with histograms (use 'cvt2' to convert to hbook).
- static char **pe_list_fname** [MAXFN]
- static char output_fname [MAXFN]

  Output in HESS/HEGRA-specific data format.
- static int new_output = 0

Replace any existing output file rather than append, if non-zero.

- static char stars_fname [MAXFN]

  List with stars shining.
- static long star_photons

  The number of photons used for star light.
- static char nsb_sky_map_fname [MAXFN]

  Sky position (az, alt) dependent scaling factor to be evaluated for each pixel (global parameter).
- static char image_fname [MAXFN]

  File for Postscript camera images.
- static char imaging_list [MAXFN]

  List of star light photon imaging.
- static int movie_flag

  True of images for movie to be made.
- static double power_law

  Power law of weighted spectrum.
- static double site_altitude

  Altitude of observation level [m].
- static int only_triggered_arrays

  True if only showers shich triggered the array are considered.
- static int only_triggered_telescopes

  True if only triggered telescopes are read out.
- static int save_photons

  Bit 0: True if to save photon data to output file; bit 1: save p.e., ...
- static int save_pe_amp

  Bit 0: Save signal p.e. together with their amplitude, bit 1: like bit 1 in save_photons.
- static int save_calib_pe

  Save signal photo-electrons also in calibration events (laser/LED type).
- static int sky_is_variable

  True if sky is recalculated after pointing changes.
- static int all_wl_random

  Assume all photon w.l.s are random.
- static int select_light

  0: all photon bunches, 1: only pos. bs, -1: only neg. bs, 2: only zero wl, 3: only non-zero wl
- static double array_clock_window

  A common offset (maximum, randomized) to array-wide times of readout start etc.
- static char config_release [100]

  In case configuration files include a release string of its own. (No effect)
- static char config_version [100]

  In case configuration files include a version string of its own. (No effect)
- static char array_config_name [100]

  Meta-parameter (has no effect)
- static char array_config_version [100]

  Meta-parameter (has no effect)
- static char array_config_variant [100]

  Meta-parameter (has no effect)
- static char site_config_name [100]

  Meta-parameter (has no effect)
- static char site_config_version [100]

  Meta-parameter (has no effect)
- static char site_config_variant [100]

  Meta-parameter (has no effect)

- static double convergent_pos [3]

  Reference x/y/z position [m] from where nominal viewing direction is true direction.
- static double convergent_distance

  Distance [m] from reference position where viewing directions should intersect.
- static double convergent_height

  Height above obs.
- static double convergent_depth

  Atmospheric depth [g/cm^ 2] where viewing directions should intersect.
- static int ignore_nontrig

  Ignore writing MC data for non-triggered showers.
- static int ignore_mcdata

  Ignore all MC data; try looking like real data.
- static long iobuf_max

  Size limit for input I/O buffers.
- static long iobuf_output_max

  Size limit for output I/O buffers.
- static int max_events

  Program should stop after processing so many events.
- static int max_trig_events

  Program should stop after processing so many triggered events.
- static char random_state [MAXFN]

  The file in which the state of the random number generator is stored.
- static char random_generator [60]

  The name of the random number generator to be used.
- static char random_seed [MAXFN]

  String with random seed number or "auto" or the name of a file from which to pick (plus method hint).
- static MplWordList mpl_global

  The list of parameter names for global metaparameters.
- static MplWordList mpl_tel

  The list of parameter names for telescope-specific metaparameters.
- static MetaParamList **metapar_global**
- static MetaParamList **metapar_tel** [MAX_TEL]
- static struct cmdline_conf **cmdl_cf_root** = { NULL, NULL }
- static struct cmdline_conf **cmdl_cf_weak** = { NULL, NULL }
- static const char * preprocessor = "pfp -v -l."

  No variable substitution as in C preprocessor.
- static const char * prepro_stdin = "-"

  Option needed to preprocess from stdin.
- static char **prepro_cmdline** [10241]
- int **id_seg_prm**
- int **id_seg_sec**
- static struct imaging_setup * **telescope_setup** [MAX_TEL]

8.32.1 Detailed Description

Configure all the things needed for the simulation.
This file contains code to configure the telescope optics and electronics used in the simulation and to setup all constants used for the simulation.

Author

Konrad Bernloehr

Date

1997, 1999-2001, ..., 2010 ... 2015-2020, 2021, 2022, 2023

8.32.2 Function Documentation

8.32.2.1 convergence_correction()

```
int convergence_correction (
            struct telescope_array * array )
```

Correct the telescope pointing for convergent viewing.
The correction is based on the assumption that the telescopes have all the same basic pointing direction. If the telescopes have separate pointing directions, with corrections for convergent viewing already taken into account there, then no additional convergence correction should be configured.
Definition at line 5745 of file sim_config.c.
References telescope_array::altitude, mc_run::atmosphere, telescope_array::azimuth, telescope_array::camera, telescope_array::conv_div_opt, telescope_optics::convergent_depth, telescope_optics::convergent_height, telescope_array::electronics, camera_electronics::itel, pm_camera::itel, telescope_optics::itel, M_PI, telescope_↩ array::mean_convergent_depth, telescope_array::ntel, telescope_array::optics, telescope_array::xtel, telescope_↩ array::ytel, and telescope_array::ztel.

8.32.2.2 do_config()

```
static int do_config (
            int argc,
            char ** argv )  [static]
```

Setup of default configuration and interpretation of command line.
Function used for interpretation of command-line arguments and for seting up the default configuration.
Definition at line 1463 of file sim_config.c.
References get_simtel_prog_path().

8.32.2.3 get_random_seed_from_file()

```
int get_random_seed_from_file (
            char * want_seed,
            size_t srs )
```

Read a list of pre-generated seeds for the random generator and chose a way to pick one of these seeds.
The following methods are available, as indicated in front of the file name: file-by-run: Use the CORSIKA_↩ RUN environment variable. file-by-time: Use a combination of seconds and microseconds of current time. file-by-random: Use an auto-generated random number (involves /dev/urandom). The method-specific number, modulo the number of available seeds, determines which of the pre-generated seeds will be picked and returned in random_↩ seed.

Parameters

want_seed	As input a string with method and file name, as output the selected pre-generated seed.
srs	The size of the character string field want_seed.

Returns

0 (OK), -1 (error)

Definition at line 2631 of file sim_config.c.

8.32.2.4 init_setup()

```
int init_setup (
            int argc,
```

```
            char ** argv,
            struct telescope_array * array,
            struct telescope_optics * tel_optics,
            struct pm_camera * camera,
            struct camera_electronics * electronics,
            struct mc_options * options )
```

Initialize the whole setup and the different pieces of the simulation.
Initialize the whole setup (default setup and telescope-specific setup), read in all the auxiliary data tables like quantum efficiencies and so on, and initialize all the constants needed for the simulation. Command-line arguments are passed to do_config().
Definition at line 2787 of file sim_config.c.
References imaging_setup::focus_offset, and imaging_setup::mirrors.

8.32.2.5 kill_pixel()

```
static void kill_pixel (
            struct camera_electronics * electronics,
            int j )  [static]
```

Set a given pixel as dead, resulting in no signal or trigger from incoming light and no DC current.
Electronic noise remains.
Definition at line 2577 of file sim_config.c.
References pm_and_fadc_channel::background, pm_and_fadc_channel::calib, pm_and_fadc_channel::current, pm_and_fadc_channel::disc_amplitude, pm_and_fadc_channel::fadc_amplitude, pm_and_fadc_channel::fadc_↩ped_shift, camera_electronics::fadc_per_channel, pm_and_fadc_channel::is_off, pm_and_fadc_channel::nsb_↩pixfact, channel_calibration::pedestal, channel_calibration::pedestal_comp_add, channel_calibration::pedestal_↩nsb, pm_and_fadc_channel::qe_rel, and camera_electronics::use_comp_ped.

8.32.2.6 metapar_deliver()

```
MetaParamList* metapar_deliver (
            int itel )
```

Deliver the locally maintained set of metaparameters to the external code.

Parameters

itel	The telescope index (0 ... MAX_TEL-1, or -1 for global parameters).

Returns

Pointer to the prepared metaparameter set or NULL.

Definition at line 1252 of file sim_config.c.

8.32.2.7 metaparam_config_handler()

```
static int metaparam_config_handler (
            const char * op,
            CONFIG_VALUES __attribute__((unused)) * cfg_val )  [static]
```

Manage the list of metaparameters to be stored in the data.
There are two sets of lists, one for global configuration parameter and one for telescope-specific ones. Any known configuration parameter (other than internal, and not a function type, ...) can be placed into either set. Parameter name abbreviations are accepted just as for the actual configuration (case-insensitive).

Parameters

op	Operand of the 'METAPARam' configuration line. Assumed syntax: [GLOBAL | TELESCOPE | ANY] { ADD list | SET name=value | REMOVE list | CLEAR } where 'list' is a space- or comma-separated list of parameter names (not abbreviated). If the first part is omitted, 'TELESCOPE' gets assumed. Everything case-insensitive.

Returns

0 (OK), -1 or -2 (error)

Definition at line 676 of file sim_config.c.

8.32.2.8 mpl_add()

```
static int mpl_add (
            MplWordList * mpl,
            const char * text )  [static]
```

Internal helper function for metaparam_config_handler to add a parameter name to either set of name lists.

Parameters

text	Parameter name or comma-separated list of parameter names. Any acceptable abbreviation is OK. Duplicates are skipped.

Returns

>=0 (OK, indicates number of parameter names found), or <0 (error)

Definition at line 809 of file sim_config.c.

8.32.2.9 mpl_fill()

```
static int mpl_fill (
            int itel,
            int tel_id )  [static]
```

Fills a MetaParamList (either global or telescope-specific) with names and current values.
Since this has to be called at a specific point in the configuration process, the contents are preserved for later use.

Parameters

itel	Internal telescope index (0 ... MAX_TEL-1), or -1 for global.
tel↩ _id	The ID number of the telescope, or -1 for global.

Returns

0 (OK), -1 (error)

Definition at line 1190 of file sim_config.c.

8.32.2.10 mpl_remove()

```
static int mpl_remove (
            MplWordList * mpl,
            const char * text )  [static]
```

Internal helper function for metaparam_config_handler to remove a parameter name from either set of name lists.
Can also be used to remove a manually set '∗name=value' by its name.
Definition at line 1012 of file sim_config.c.

8.32.2.11 mpl_set()

```
static int mpl_set (
            MplWordList * mpl,
            const char * text )  [static]
```

Internal helper function for metaparam_config_handler to set a non-configuration metaparameter by hand.

Parameters

text	Parameter name=value assignment. The '=' is needed. To avoid confusion with real configuration parameters a '∗' is prepended to the name actually used. Parameters with existing names may get their value changed.

Returns

0 or 1 (OK, indicates number of set or changed), or <0 (error)

Definition at line 908 of file sim_config.c.

8.32.2.12 randomize_viewing_direction()

```
int randomize_viewing_direction (
            struct telescope_array * array )
```

Changing viewing directions from shower to shower.
On a per shower / array basis the viewing direction of all telescopes is randomized in such a way that the actual viewing directions are within a ring around the basic viewing direction given.
Note: the pattern of stars in the cameras is left unchanged unless the preprocessor flag VARIABLE_SKY_↩ BACKGROUND is set.
Definition at line 6137 of file sim_config.c.

8.32.2.13 save_or_restore_all_channels()

```
static int save_or_restore_all_channels (
            struct camera_electronics * electronics,
            int ntel,
            const char * fname )  [static]
```

Save or restore parameters of all the electronics channels.
Normally, the simulation starts with new random fluctuations of PM gains, high voltages and corresponding time delays etc. If more runs should be made with the same gains etc., their values can be saved at the end of all processing and restored at the beginning the next time the program is started.
Definition at line 5886 of file sim_config.c.
References pm_and_fadc_channel::background, pm_and_fadc_channel::calib, pm_and_fadc_channel::current, pm_and_fadc_channel::disc_amplitude, pm_and_fadc_channel::disc_threshold, pm_and_fadc_channel::fadc↩ _amplitude, camera_electronics::fadc_per_channel, pm_and_fadc_channel::gate_length, channel_calibration↩ ::laser, channel_calibration::laser_time, channel_calibration::pedestal, pm_and_fadc_channel::pedestal, channel↩ _calibration::pedestal_sum, camera_electronics::pixels, pm_and_fadc_channel::qe_rel, pm_and_fadc_channel↩ ::sensitivity, camera_electronics::telescope, pm_and_fadc_channel::transit_delay, and pm_and_fadc_channel↩ ::trigger_disabled.

8.32.2.14 setup_starlight()

```
static int setup_starlight (
            char * fname,
            double * star_light,
            int npix,
            struct telescope_optics * optics,
            struct pm_camera * cam,
            double trans )  [static]
```

Add light from individual stars.
Add star light to the setup (if wanted). Star light is raytraced through the telescope like Cherenkov light. Assumed wavelength of the light source(s)
Definition at line 1993 of file sim_config.c.

8.32.3 Variable Documentation

8.32.3.1 convergent_depth

```
double convergent_depth  [static]
```

Atmospheric depth [g/cm^2] where viewing directions should intersect.
This takes precedence over convergent_height and convergent_distance if != 0.
Definition at line 573 of file sim_config.c.

8.32.3.2 convergent_height

```
double convergent_height  [static]
```

Height above obs.
level [m] where viewing directions should intersect. This takes precedence over convergent_distance if != 0.
Definition at line 570 of file sim_config.c.

8.32.3.3 random_generator

```
char random_generator[60]  [static]
```

The name of the random number generator to be used.
Unless compiled with -DWITH_GSL_RNG it must be "Ranlux".
Definition at line 592 of file sim_config.c.

8.32.3.4 random_state

```
char random_state[MAXFN]  [static]
```

The file in which the state of the random number generator is stored.
'none' or 'auto' cause initialisation from seconds+microseconds of time.
Definition at line 589 of file sim_config.c.

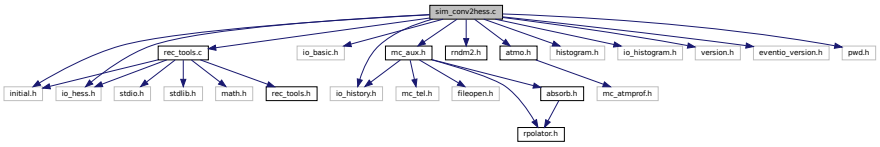
8.33 sim_conv2hess.c File Reference

Conversion from internal data to HESS or CTA eventio data format.

```
#include "initial.h"
#include "io_basic.h"
#include "io_history.h"
#include "mc_aux.h"
#include "rndm2.h"
#include "atmo.h"
#include "io_hess.h"
```

```
#include "histogram.h"
#include "io_histogram.h"
#include "version.h"
#include "eventio_version.h"
#include <pwd.h>
#include "rec_tools.c"
```

Include dependency graph for sim_conv2hess.c:



Data Structures

- struct simtel_analysis_data

  Data used in image parameters analysis.

Macros

- #define **MAX_NEIGHBOURS2** 24

Functions

- static void **convert_check_begin_tel_array** (int shower, int iarray)
- static void **convert_check_end_tel_array** (void)
- static void image_clean_1 (int itel, struct camera_electronics ∗el, struct pm_camera ∗cam, double img_scale, double pe[ ], double ∗psumall, double ∗psumsel, double ts)

  Calculate image parameters from pixel intensities in once camera.
- static void convert_write_headers ()

  Writing of run headers and configuration material when first event starts.
- static void **convert_image_analysis** (struct telescope_array ∗array, int only_if_triggered)
- static void convert_shower_analysis (struct telescope_array ∗array, int only_if_triggered)

  Call HESS-specific shower analysis functions to analyse the simulated shower right now.
- static int find_significant_pixels (AdcData ∗raw, double ∗pedestal, double thresholds[3], int nb_opt, int num↩ _nb[ ], int nb[ ][MAX_NEIGHBOURS])

  Find significant pixels for built-in zero suppression.
- static void convert_initial_moni_calib (struct telescope_array ∗array, HTime now)

  Convert the initial monitoring and calibration data from the internal structures for the output.
- static void convert_mc_pe_sum (struct telescope_array ∗array)

  Convert various numbers of photons hitting the telescope fiducial spheres, how much of that is in some predefined wavelength range and what is left after atmospheric propagation and after final photo-electron detection.
- static double **dist2** (double x, double y)
- static void ps_plot (struct telescope_array ∗array, int itel)

  Write a Postscript camera plot to a dedicated file, with pixels orientated as the corresponding position in the sky would be seen (altitude angle upwards, azimuth to the right).
- static void ps_plot_movie (struct telescope_array ∗array, int itel, int ibin)

  Write Postscript of camera image at each time interval to a dedicated file.
- static void print_pix_col (double n_o_r, FILE ∗psfile, double gamma_coeff, int mode)

  Print a false-colour RGB value for a pixel intensity.
- static int **print_pixel_def** (FILE ∗psfile, struct pm_camera ∗cam, double scale)

- static void **ps_camera_plot** (FILE *psfile, struct pm_camera *cam, double scale, double *amp_list, double amp_offset, double amp_range, double *showval, int num_showval, double showval_scale, const char *unit_showval, double gamma_coeff, int mode)
- void convert_check_setup ()

  One-time check if main array size limits in sim_telarray are compatible with producing output data via the hessio library.
- void convert_config (struct telescope_array *array, int __attribute__((unused)) format, char *output_fname, int replace)

  Configure parameters needed for the conversion of internal simulation data to the raw data as provided by the HESS/CTA system of telescopes.
- void convert_runheader (struct telescope_array *array, int run)

  Convert run header.
- void convert_input_lines (struct linked_string *list)

  Convert the CORSIKA input configuration lines to the output file.
- void convert_mc_event (struct telescope_array *array)

  Convert MC event data (per shower usage).
- void convert_mc_photons (int shower, int iarray, int itel, double photons, struct bunch *bunches, int nbunches)

  Save CORSIKA photon bunches into the output data file.
- void convert_mc_photons3d (int shower, int iarray, int itel, double photons, struct bunch3d *bunches3d, int nbunches)

  Save CORSIKA photon bunches of 3D type into the output data file.
- void convert_calib_pe_list (int type, int iarray, int itel, int npe, int flags, int pixels, int *pe_counts, int *tstart, double *t, double *a, int *photon_counts)

  Save list of all 'detected' photo-electrons from internal calibration event photons.
- void convert_mc_pe_list (int shower, int iarray, int itel, int npe, int flags, int pixels, int *pe_counts, int *tstart, double *t, double *a, int *photon_counts)

  Save list of all 'detected' photo-electrons from Cherenkov photons.
- void convert_basic_data (struct telescope_array __attribute__((unused)) *array, int __attribute__((unused)) run, int __attribute__((unused)) event, int __attribute__((unused)) write_all_pixels, int __attribute__((unused)) only_if_triggered)

  Convert data from internal structures to structures as used for the HESS telescope system (basic data only).
- void **convert_calibdata** (struct telescope_array *array, int run, int event, int type)
- void convert_corsika_particles (struct telescope_array *array, int jarray, int itel, double photons, struct bunch *bunches, int nbunches)

  If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.
- void convert_corsika_particles3d (struct telescope_array *array, int jarray, int itel, double photons, struct bunch3d *bunches3d, int nbunches)

  If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.
- void convert_eventdata (struct telescope_array *array, int run, int event, int write_all_pixels, int only_if_↩ triggered)

  Convert data from internal structures to structures as used for the HESS telescope system, optionally analyse the simulated showers (by the same methods as used for the measured data) and write the converted raw data to the output file.
- void convert_runend (struct telescope_array *array)

  End of run in conversion to HESS format.
- void convert_finish (struct telescope_array *array)

  Finished with all conversions, clean up and close output file.

Variables

- int **global_verbose**
- static AllHessData hsdata

  The outer structure containing all converted data.
- static IO_BUFFER * iobuf

  The I/O buffer specific for writing the converted output data.
- static int **runh_set** = 0
- static int **config_set** = 0
- static int **ignore_mcdata** = 0
- static struct linked_string * **input_lines**
- static double zero_thresh [3] = { 200., 100., 50. }

  Thresholds used in the pixel zero-suppression.
- struct simtel_analysis_data **ana_data** [MAX_TEL]
- static char **ps_head1** [ ]
- static char **ps_head2** [ ]
- static char **ps_head3** [ ]
- static char **ps_begin_page1** [ ]
- static char **ps_begin_page2** [ ]
- static char **ps_end_page** [ ]
- static char **ps_trailer** [ ]
- static char **alt_az_arrow** [ ]
- static int **ps_num_page** = 0
- static IO_ITEM_HEADER **item_header_tel_array**

8.33.1 Detailed Description

Conversion from internal data to HESS or CTA eventio data format.
This file contains the code to convert from the internal data representation of the telescope simulation to the data structures and output formats used sim_telarray output data. This applies to both HESS and CTA simulations.
This file also covers producing Postscript camera plots of the signals, calibration values, etc. etc. in each pixel.

Author

Konrad Bernloehr

Date

1999, 2000, ..., 2010, ..., 2019, 2020, 2021, 2022, 2023, 2024

8.33.2 Function Documentation

8.33.2.1 convert_basic_data()

```
void convert_basic_data (
            struct telescope_array __attribute__((unused)) * array,
            int __attribute__((unused)) run,
            int __attribute__((unused)) event,
            int __attribute__((unused)) write_all_pixels,
            int __attribute__((unused)) only_if_triggered )
```

Convert data from internal structures to structures as used for the HESS telescope system (basic data only).
This is used for both shower and calibration events.
param array Structure tree of array and telescope specific data. param run Run number. param event Event number (100*event + array number). param write_all_pixels 1 to write even pixels with no significant signal, 0 otherwise. param only_if_triggered 1 to write only data of triggered telescopes, 0 to write data of all telescopes.

Returns

(none)

Definition at line 2752 of file sim_conv2hess.c.

8.33.2.2 convert_config()

```
void convert_config (
            struct telescope_array * array,
            int __attribute__((unused)) format,
            char * output_fname,
            int replace )
```

Configure parameters needed for the conversion of internal simulation data to the raw data as provided by the HESS/CTA system of telescopes.

Parameters

array	Structure tree of array and telescope specific data.
format	Not defined so far (there is only one format).
output_fname	File name for simulated data.
replace	If non-zero, a fresh file is started instead of appending.

Returns

(none)

Definition at line 1545 of file sim_conv2hess.c.

8.33.2.3 convert_corsika_particles()

```
void convert_corsika_particles (
            struct telescope_array * array,
            int jarray,
            int itel,
            double photons,
            struct bunch * bunches,
            int nbunches )
```

If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.

Parameters

array	array number (usually 999 for particles)
tel	telescope number (usually 999 for particles)
photons	sum of photons (and fractions) in this device
bunches3d	list of 3D photon bunches
nbunches	number of elements in bunch list

Definition at line 4362 of file sim_conv2hess.c.
References telescope_array::particles.

8.33.2.4 convert_corsika_particles3d()

```
void convert_corsika_particles3d (
```

```
            struct telescope_array * array,
            int jarray,
            int itel,
            double photons,
            struct bunch3d * bunches3d,
            int nbunches )
```

If the CORSIKA output contains data on particles arriving at ground level, we preserve that by simply copying to the output file.

Parameters

array	array number (usually 999 for particles)
tel	telescope number (usually 999 for particles)
photons	sum of photons (and fractions) in this device
bunches3d	list of 3D photon bunches
nbunches	number of elements in bunch list

Definition at line 4409 of file sim_conv2hess.c.
References telescope_array::particles.

8.33.2.5 convert_eventdata()

```
void convert_eventdata (
            struct telescope_array * array,
            int run,
            int event,
            int write_all_pixels,
            int only_if_triggered )
```

Convert data from internal structures to structures as used for the HESS telescope system, optionally analyse the simulated showers (by the same methods as used for the measured data) and write the converted raw data to the output file.

Parameters

array	Structure tree of array and telescope specific data
run	Run number
event	Event number (100∗event + array number)
write_all_pixels	1 to write even pixels with no significant signal, 0 otherwise
only_if_triggered	1 to write only data of triggered telescopes, 0 to write data of all telescopes.

Returns

(none)

Definition at line 4461 of file sim_conv2hess.c.

8.33.2.6 convert_finish()

```
void convert_finish (
            struct telescope_array * array )
```

Finished with all conversions, clean up and close output file.

Parameters

array	Structure tree of array and telescope specific data.

Returns

(none)

Definition at line 5211 of file sim_conv2hess.c.

8.33.2.7 convert_initial_moni_calib()

```
static void convert_initial_moni_calib (
            struct telescope_array * array,
            HTime now )  [static]
```

Convert the initial monitoring and calibration data from the internal structures for the output.
Most (usally all) of them will not change during a simulation run. They are all to be recorded before the first event (including potential internal calibration events).
This includes:

- The 'MC Pixel Monitor' data which is actually a collection of values as used internally.
- The 'Monitoring' data with the effective pedestal values (after any internal adjustments in raw data), scaled values of pixel currents and high-voltages (scaling factors being rather abitrary). In addition, there is a number of values which were needed for the data model in the original HESS cameras but are only filled formally with hard-coded numbers, thus should be ignored.
- The 'Laser Calibration' data, corresponding to what might be derived from calibration light sources, including single-p.e. response with and without flatfielding corrections as well as time delays expected in the light-flasher signals. These values are generally not exactly what is used internally but deviate by an assumed random number corresponding to the expected accuracy achieved by the calibration. For the 'true' values, see the 'MC Pixel Monitor' part.

Parameters

array	Structure tree of array and telescope specific data.
now	Time to which the data is intended to correspond - usually just the time when the simulation was run.

Returns

(none)

Definition at line 4022 of file sim_conv2hess.c.
References channel_calibration::amp_to_npe, pm_and_fadc_channel::background, pm_and_fadc_channel::calib, channel_calibration::calib_rel, pm_and_fadc_channel::current, telescope_array::electronics, pm_and_fadc_channel::fadc_amplitude, pm_and_fadc_channel::gain_rel, hsdata, iobuf, pm_and_fadc_channel::is_off, camera_electronics::itel, camera_electronics::laser_calib_done, laser_calib_eval(), channel_calibration::laser_time, camera_electronics::max_int_frac, camera_electronics::max_pixtm_frac, camera_electronics::nom_disc_threshold, telescope_array::ntel, camera_electronics::num_gains, channel_calibration::pedestal_comp_add, channel_calibration::pedestal_nsb, camera_electronics::pixels, pm_and_fadc_channel::qe_rel, RandGauss(), channel_calibration::sigma_pedestal, camera_electronics::telescope, pm_and_fadc_channel::transit_delay, pm_and_fadc_channel::trigger_disabled, camera_electronics::use_comp_ped, and pm_and_fadc_channel::voltage_rel.

8.33.2.8 convert_input_lines()

```
void convert_input_lines (
            struct linked_string * list )
```

Convert the CORSIKA input configuration lines to the output file.
This is done here by simply rewriting it in the same format.
Definition at line 2432 of file sim_conv2hess.c.

8.33.2.9 convert_mc_event()

```
void convert_mc_event (
            struct telescope_array * array )
```

Convert MC event data (per shower usage).
If once-per-run material like run headers and configuration has not been written yet, writing of that material is triggered first. If the shower-specific data is not written yet, this is also accomplished before the shower-usage specific data is eventually written.
Definition at line 2475 of file sim_conv2hess.c.

8.33.2.10 convert_mc_pe_sum()

```
static void convert_mc_pe_sum (
            struct telescope_array * array )  [static]
```

Convert various numbers of photons hitting the telescope fiducial spheres, how much of that is in some predefined wavelength range and what is left after atmospheric propagation and after final photo-electron detection.

Parameters

array	Structure tree of array and telescope specific data.

Returns

(none)

Definition at line 4315 of file sim_conv2hess.c.
References telescope_array::electronics, hsdata, ignore_mcdata, iobuf, camera_electronics::itel, telescope_array::ntel, camera_electronics::photons_all, camera_electronics::photons_atm, camera_electronics::photons_atm_300_600, camera_electronics::photons_atm_400, camera_electronics::photons_atm_qe, camera_electronics::photons_detected, and camera_electronics::telescope.

8.33.2.11 convert_runend()

```
void convert_runend (
            struct telescope_array * array )
```

End of run in conversion to HESS format.
Write end-of-run statistics.

Parameters

array	Structure tree of array and telescope specific data.

Returns

(none)

Definition at line 5193 of file sim_conv2hess.c.
References hsdata, and telescope_array::ntel.

8.33.2.12 convert_runheader()

```
void convert_runheader (
            struct telescope_array * array,
            int run )
```

Convert run header.
Note that configuration may not be set yet. Output is, therefore, delayed until the first event.

Parameters

ntel	Number of telescopes in the array.
optics	Structure with optics-specific data.
run	Run number.

Returns

(none)

Definition at line 2193 of file sim_conv2hess.c.
References telescope_array::altitude, telescope_array::azimuth, telescope_array::conv_div_opt, mc_run::corsika_version, mc_run::e_max, mc_run::e_min, mc_run::height, hsdata, mc_options::ignore_mcdata, ignore_mcdata, mc_options::image_fname, M_PI, telescope_array::mean_convergent_depth, telescope_array::min_tel_trigger, telescope_array::ntel, mc_run::num_arrays, mc_run::num_showers, telescope_array::optics, telescope_array::options, mc_run::phi_max, mc_run::phi_min, mc_run::radius, mc_run::run_start, mc_run::slope, telescope_optics::telescope, telescope_array::telescope_ignore, mc_run::theta_max, mc_run::theta_min, mc_run::viewcone_max, mc_run::viewcone_min, telescope_array::xtel, telescope_array::ytel, and telescope_array::ztel.

8.33.2.13 convert_shower_analysis()

```
static void convert_shower_analysis (
            struct telescope_array * array,
            int only_if_triggered )  [static]
```

Call HESS-specific shower analysis functions to analyse the simulated shower right now.
All of it depends on compiling with the PULSE_ANALYSIS flag and most of it also with the ANALYSE_SHOWER flag.

Parameters

array	Structure tree of array and telescope specific data

Returns

(none)

Definition at line 3718 of file sim_conv2hess.c.

8.33.2.14 find_significant_pixels()

```
static int find_significant_pixels (
            AdcData * raw,
            double * pedestal,
            double thresholds[3],
            int nb_opt,
            int num_nb[],
            int nb[][MAX_NEIGHBOURS] )  [static]
```

Find significant pixels for built-in zero suppression.
This built-in zero suppression uses a non-standard image cleaning algorithm and not the usual two-level cleaning. Neigbouring pixels are also marked to be retained in output data.
Definition at line 3933 of file sim_conv2hess.c.

8.33.2.15 image_clean_1()

```
static void image_clean_1 (
            int itel,
```

```
            struct camera_electronics * el,
            struct pm_camera * cam,
            double img_scale,
            double pe[],
            double * psumall,
            double * psumsel,
            double ts )  [static]
```

Calculate image parameters from pixel intensities in once camera.
Not adapted for cameras sampling the same channel with multiple FADCs (like HEGRA).
Definition at line 3088 of file sim_conv2hess.c.
References channel_calibration::amp_to_npe, pm_and_fadc_channel::calib, camera_electronics::num_gains, pm_and_fadc_channel::overflow, channel_calibration::pedestal_nsb, camera_electronics::pixels, pm_and_fadc_↩channel::sum_adc, and camera_electronics::sum_bins.

8.33.2.16 ps_plot()

```
static void ps_plot (
            struct telescope_array * array,
            int itel )  [static]
```

Write a Postscript camera plot to a dedicated file, with pixels orientated as the corresponding position in the sky would be seen (altitude angle upwards, azimuth to the right).
This can be used for a single (F)ADC sum, signal traces (one camera plot for each sample), while the companion ps_plot_movie() function writes one plot (page) per sample.
Definition at line 606 of file sim_conv2hess.c.
References telescope_array::camera, telescope_array::electronics, camera_electronics::itel, pm_camera::itel, telescope_optics::itel, telescope_array::optics, pm_camera::pixel_x_pos, pm_camera::pixel_y_pos, camera_↩electronics::pixels, and camera_electronics::telescope.

8.33.2.17 ps_plot_movie()

```
static void ps_plot_movie (
            struct telescope_array * array,
            int itel,
            int ibin )  [static]
```

Write Postscript of camera image at each time interval to a dedicated file.
This file can later be converted to a movie.
Definition at line 1197 of file sim_conv2hess.c.
References telescope_array::camera, pm_camera::camera_body_diameter, telescope_array::electronics, camera_image_plot_param::gamma, mc_options::image_fname, pm_camera::img, camera_electronics::itel, pm↩_camera::itel, telescope_array::options, pm_camera::pixel_x_pos, pm_camera::pixel_y_pos, camera_electronics↩::pixels, camera_image_plot_param::range, and camera_electronics::telescope.

8.33.3 Variable Documentation

8.33.3.1 alt_az_arrow

```
char alt_az_arrow[]  [static]
```

Initial value:

```
=
      "n 18000 26000 m "
      "0 100 rl 200 -100 rl -200 -100 rl 0 100 rl -1000 0 rl "
      "cp gs 20 slw black s gr\n"
      "txt5 18700 26100 mtxt (Az) tblack\n"
      "n 17000 25000 m "
      "100 0 rl -100 -200 rl -100 200 rl 100 0 rl 0 1000 rl "
      "cp gs 20 slw black s gr\n"
      "txt5 17000 24600 mtxt (Alt) tblack\n"
      "gs 17800 25500 tr %f rot -17800 -25500 tr\n"
```

```
        "n 17800 25500 m "
        "0 100 rl 200 -100 rl -200 -100 rl 0 100 rl -300 0 rl "
        "cp gs 10 slw black s gr\n"
        "txt2 17950 25350 mtxt (y) tblack\n"
        "n 17500 25200 m "
        "100 0 rl -100 -200 rl -100 200 rl 100 0 rl 0 300 rl "
        "cp gs 10 slw black s gr\n"
        "txt2 17700 25200 mtxt (x) tblack\n"
      "gr\n"
```

Definition at line 304 of file sim_conv2hess.c.

8.33.3.2 iobuf

```
IO_BUFFER* iobuf  [static]
```

The I/O buffer specific for writing the converted output data.
Note that this pointer is local to this file!
Definition at line 88 of file sim_conv2hess.c.
Referenced by convert_calib_pe_list(), convert_initial_moni_calib(), convert_mc_pe_sum(), convert_write_↩ headers(), hup_signal_function(), and main().

8.33.3.3 ps_begin_page1

```
char ps_begin_page1[]  [static]
```

Initial value:

```
=
"%%Page: "
```

Definition at line 281 of file sim_conv2hess.c.

8.33.3.4 ps_begin_page2

```
char ps_begin_page2[]  [static]
```

Initial value:

```
=
"save\n"
"10 setmiterlimit\n"
"n -1000 31000 m -1000 -1000 l 22000 -1000 l 22000 31000 l cp clip\n"
" 0.02835 0.02835 sc\n"
"gs\n"
"7.500 slw\n"
"black\n"
```

Definition at line 283 of file sim_conv2hess.c.

8.33.3.5 ps_end_page

```
char ps_end_page[]  [static]
```

Initial value:

```
=
"gr\n"
"showpage\n"
```

Definition at line 291 of file sim_conv2hess.c.

8.33.3.6 ps_head1

```
char ps_head1[]  [static]
```

Initial value:

```
=
"%!PS-Adobe-2.0\n"
"%%Title: H.E.S.S. Telescope Simulation\n"
"%%Creator: sim_hessarray"
```

Definition at line 155 of file sim_conv2hess.c.

8.33.3.7 ps_trailer

```
char ps_trailer[]  [static]
```

Initial value:

```
=
"rs\n"
```

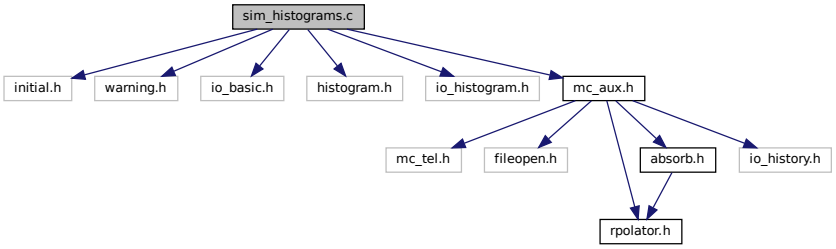
Definition at line 295 of file sim_conv2hess.c.

8.34 sim_histograms.c File Reference

Book, fill, and save histograms.

```
#include "initial.h"
#include "warning.h"
#include "io_basic.h"
#include "histogram.h"
#include "io_histogram.h"
#include "mc_aux.h"
```

Include dependency graph for sim_histograms.c:



Macros

- #define **PULSE_ANALYSIS** 1
- #define **PULSE_SHAPE_HISTOGRAMS**
- #define **NUM_LARGEST** 21

Functions

- int save_histograms (char *histogram_fname, IO_BUFFER *iobuf)

  Save the available histograms in a file (using 'eventio' format).
- static double **square** (double x)
- int init_histograms (struct telescope_array *array, struct pm_camera *camera, struct camera_electronics *electronics)

  Initialize (book) all wanted histograms.
- int fill_histograms (struct telescope_array *array, int __attribute__((unused)) iarray, struct camera_electronics *electronics, double energy, double wt, double alt, double az)

  Fill histograms according to results from the simulation.

Variables

- static double **ped_stat** [MAX_TEL][MAX_PIXELS][1][3]

8.34.1 Detailed Description

Book, fill, and save histograms.

Author

Konrad Bernloehr

Date

1997 ... 2022

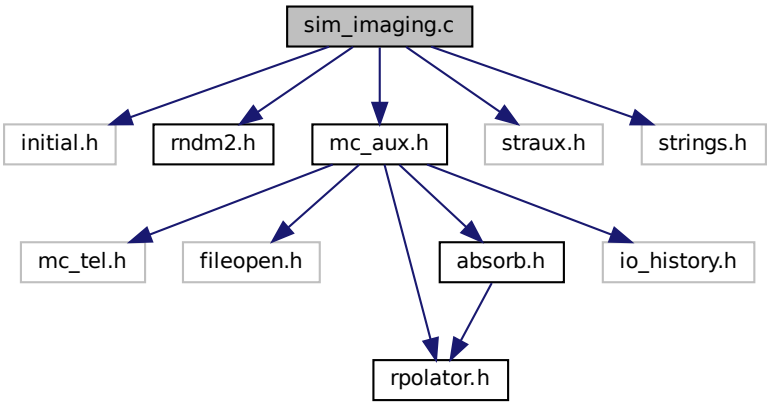
This file contains code to book, fill, and save histograms used to check the simulations and to evaluate results.

8.35 sim_imaging.c File Reference

Raytracing of photons through a multi-mirror telescope.

```
#include "initial.h"
#include "rndm2.h"
#include "mc_aux.h"
#include "straux.h"
#include <strings.h>
```

Include dependency graph for sim_imaging.c:



Data Structures

- struct trg_grp_link

Macros

- #define Nint(x) ((x)>0?(int)((x)+0.5):(int)((x)-0.5))

  The nearest integer to a floating point number.
- #define **min**(a, b) ((a)<(b)?(a):(b))
- #define **max**(a, b) ((a)>(b)?(a):(b))
- #define **Abs**(x) ((x)>=0?(x):-1*(x))
- #define **square**(x) ((x)*(x))
- #define **MAX_INPUT_SUM_IN_TRG** 8

Functions

- int line_plane_intersection (double xl[3], double cl[3], double xp[3], double cp[3], double xi[3])

  Intersection of a line with a plane normal to given vector.

- int line_ztube_intersection (double xl[3], double cl[3], double R, double zmin, double zmax, int fwd_only, double xi[3])

  Calculate if there is a valid intersection between a line and a tube (open cylinder) along the z axis.
- int solve_quadratic_equation (double a, double b, double c, double ∗x1, double ∗x2)

  Solve a quadratic equation a∗x∗∗2 + b∗x + c = 0.
- int reflect_on_spherical_mirror (double r, double mrad, double arnd, double ∗p, double ∗d, int type, double phi_m, double ∗angle)

  Reflect a photon on a spherical mirror.
- int reflect_on_parabolic_mirror (double r, double mrad, double arnd, double ∗p, double ∗d, int type, double phi_m, double ∗angle)

  Reflect a photon on a parabolic mirror.
- int refract_in_fresnel_lens (double r, double mrad, double arnd, double ∗p, double ∗d, int type, double n, double ∗angle)

  Refract a photon in a thin fresnel lens (using framework from reflection in parabolic mirror)
- double poly_eval_even (double x, int npar, const double ∗params)

  Evaluate an even polynomial.
- double poly_deriv_even (double x, int npar, const double ∗params)

  Evaluate the derivative of an even polynomial.
- int reflect_on_polynomial_mirror (double r, double mrad, double arnd, double ∗p, double ∗d, int type, double __attribute__((unused)) phi_m, int npar, double ∗params, double hole_diam, double ∗angle)

  Reflect a photon on a general mirror with rotational symmetry described by an even polynomial shape.
- void make_trafo (double alpha_ang, double beta_ang, double gamma_ang, double x, double y, double z, struct transform_struct ∗trans)

  Create rotation matrix and translation vector.
- void **make_trafo_off** (const struct transform_struct ∗t1, const struct transform_struct ∗t2, const double ∗o3, struct transform_off_struct ∗ts)
- void transform (double ∗p, double ∗d, const struct transform_struct ∗ts, int mode)

  Rotation plus translation transformation (forw.
- void transform_off (double ∗p, double ∗d, const struct transform_off_struct ∗ts, int mode)

  Transformation with extra offsets between rotation axes (forw.
- int tel_newdir (double phi, double theta, double tel_rand, struct telescope_optics ∗optics)

  Set up new transformations between ground and telescope.
- int tel_setup_primary (struct telescope_optics ∗optics, double phi, double theta)

  Set up all the transformations between ground, telescope, mirror, and camera reference frames, for telescopes with the camera in the primary focus (mirror_class 0 or 1).
- int tel_setup_secondary (struct telescope_optics ∗optics, double phi, double theta)

  Set up all the transformations between ground, telescope, mirror, and camera reference frames, for telescopes with the camera in the primary focus (mirror_class 0 or 1).
- int tel_mirror_grid_setup (struct telescope_optics ∗optics)

  Setup mirror 'grid' for faster raytracing (in case of many mirrors)
- int trace_photon_in_segmented (struct telescope_optics ∗optics, double ∗photon_pos, double ∗photon_dir, double ∗x_on_camera, double ∗y_on_camera, double ∗sx_camera, double ∗sy_camera, double ∗time_to↩ _camera, int ∗mirror, double ∗releff, double rlambda, double distance)

  Trace a photon through the whole telescope to the camera (if hit).
- int trace_photon_in_paraboloid (struct telescope_optics ∗optics, double ∗photon_pos, double ∗photon_dir, double ∗x_on_camera, double ∗y_on_camera, double ∗sx_camera, double ∗sy_camera, double ∗time_to↩ _camera, int ∗mirror, double ∗releff, double rlambda, double distance)

  Trace a photon through the whole telescope to the camera (if hit).
- int trace_photon_in_fresnel (struct telescope_optics ∗optics, double ∗photon_pos, double ∗photon_dir, double ∗x_on_camera, double ∗y_on_camera, double ∗sx_camera, double ∗sy_camera, double ∗time_to_↩ camera, int ∗mirror, double ∗releff, double rlambda, double distance)

  Trace a photon through the whole telescope to the camera (if hit).
- int read_mirror_segments (const char ∗fname, struct mirror_segmentation ∗seg_sets, size_t max_sets)

Read a configuration file with segmentation for either primary or secondary of a dual mirror telescope.

- ConvexPolygon * **make_convex_polygon** (size_t n, double *xp, double *yp)
- int **inside_convex_polygon** (double x, double y, ConvexPolygon *cp)
- int segment_hit (double *pos, int npar, double *poly_par, struct mirror_segmentation *seg_set)

  If the primary or secondary of a dual mirror optics are marked as segmented, this function is used to decide if and which segment of a set of segments is hit.
- int baffle_intersection (double *photon_pos, double *photon_dir, double *baffle, int fwd_only)

  Check if a photon path intersects with a baffle which can be either a piece of cylinder or a cone (and may have a finite wall thickness).
- int trace_photon_with_secondary (struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *x_on_camera, double *y_on_camera, double *sx_camera, double *sy_camera, double *time_to↩_camera, int *mirror, double *releff, double rlambda, double distance, int bypass)

  Trace a photon through the whole telescope to the camera (if hit).
- int assign_camera_pixels (struct pm_camera *cam)

  Setup the camera layout and the constants needed for fast reference between photon 'impact' position and pixel number.
- void **strip_comments** (char *line)
- const char * **geo_text** (int geo)
- int **assign_flexible_camera** (struct pm_camera *cam, struct camera_electronics *el, char *fname, double scale_factor, int default_min_pixels)
- int camera_setup_inclined_pixels (struct pm_camera *cam, struct telescope_optics *optics)

  Positions (in z) and optionally alignment of pixels set up to follow focal surface.
- int pm_grid_setup (struct pm_camera *camera)

  Setup of a grid on which PM hits can be searched faster.
- int pm_grid_search (struct pm_camera *camera, double *photon_pos, double *photon_dir, double *xcam, double *ycam, double *sxcam, double *sycam, double *time_to_camera)

  Faster search of PM hits on a grid over the camera front.
- int camera_hit (struct pm_camera *cam, struct telescope_optics *optics, double *photon_pos, double *photon_dir, double *xcam, double *ycam, double *sxcam, double *sycam, double *time_to_camera)

  Find out which (if any) pixel of the camera is hit.
- int cathode_hit (struct pm_camera *cam, int ipix, double x, double y, double xs, double ys, int iwl, double *releff)

  Find out if photocathode is hit.
- void offset_in_camera (double ref_azimuth, double ref_altitude, struct telescope_optics *optics, double *xoff, double *yoff)

  Calculate offset of a reference sky position in the camera plane.

Variables

- int **id_seg_prm**
- int **id_seg_sec**
- double airlightspeed

  The speed of light used for light propagation at the observation level.
- int **global_verbose**

8.35.1 Detailed Description

Raytracing of photons through a multi-mirror telescope.
This file contains code for the raytracing of photons through a multi-mirror telescope (normally in Davies-Cotton design) and into the pixels of either a hexagonal or square camera made of photomultipliers with reflective funnels (light guides) in front.

Author

Konrad Bernloehr

Date

1997 ... 2022

8.35.2 Function Documentation

8.35.2.1 baffle_intersection()

```
int baffle_intersection (
            double * photon_pos,
            double * photon_dir,
            double * baffle,
            int fwd_only )
```

Check if a photon path intersects with a baffle which can be either a piece of cylinder or a cone (and may have a finite wall thickness).
Note that this intersection does not try to find the first intersection as in shortest path to possible intersection points but only if there is any intersection at all.

Parameters

photon_pos	Initial position of photon (x,y,z) in optics system. If there is an intersection, it gets updated with the identified intersection position.
photon_dir	Direction cosines of photon (cx,cy,cz) in optics system.
baffle	Baffle parameters (z1,z2,r1,dr,r2,[c]).
fwd_only	If this is non-zero only forward intersections are counted.

Returns

0 (no intersection), 1 (intersecting with baffle)

Definition at line 4780 of file sim_imaging.c.
References line_ztube_intersection(), and solve_quadratic_equation().

8.35.2.2 camera_hit()

```
int camera_hit (
            struct pm_camera * cam,
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * xcam,
            double * ycam,
            double * sxcam,
            double * sycam,
            double * time_to_camera )
```

Find out which (if any) pixel of the camera is hit.

Parameters

cam	Camera configuration
photon_pos	Position of ray intersection with focal surface
photon_dir	Direction of incoming ray in camera system
xcam	On entry: x position in camera system; on exit: in pixel system
ycam	On entry: y position in camera system; on exit: in pixel system
sxcam	On entry: x slope in camera system; on exit: in pixel system

Parameters

sycam	On entry: y slope in camera system; on exit: in pixel system
time_to_camera	On entry: time to focal surface; on exit: time to pixel

Returns

-1 (no pixel hit) or pixel ID

Definition at line 7711 of file sim_imaging.c.
References pm_camera::camera_pixel_assignment_initialized, camera_setup_inclined_pixels(), pm_camera::camera_type, telescope_optics::focal_surface_parameters, telescope_optics::npar_focal, pm_grid_search(), poly_deriv_even(), and pm_camera::telescope.

8.35.2.3 cathode_hit()

```
int cathode_hit (
            struct pm_camera * cam,
            int ipix,
            double x,
            double y,
            double xs,
            double ys,
            int iwl,
            double * releff )
```

Find out if photocathode is hit.
Find out if the photocathode is hit directly or after reflection on the lightguide, or if the photon hits a gap between pixels.

Parameters

cam	Camera configuration
ipix	Pixel ID
x	X position of hit in pixel entrance plane, w.r.t. pixel center [cm]
y	Y position of hit in pixel entrance plane, w.r.t. pixel center [cm]
sx	X slope of incident ray in pixel frame (tan)
sy	Y slope of incident ray in pixel frame (tan)
iwl	Wavelength index [nm]
releff	Relative efficiency along the photon path, w.r.t. the efficiency applied before ray-tracing; gets multiplied here with factor related to incidence on pixel (and wavelength, if applicable).

Returns

1 - photocathode (directly) hit, 2 - lightguide hit (configuration case without efficiency table(s)) 3 - pixel lightguide efficiency table applied -1 - gap at edge of lightguide hit, -2 - no hit, -3 - internal absorption.

Definition at line 7790 of file sim_imaging.c.
References Pix_Type::angle_table_size, pm_camera::camera_type, Pix_Type::cathode_shape, Pix_Type::funnel_angle_table, Pix_Type::funnel_wl_table, pm_camera::lightguide_reflectivity, M_PI, MAX_LAMBDA, PM_List::pix_type, Pix_Type::pixel_cathode_r2, pm_camera::pixel_cathode_r_squared, Pix_Type::pixel_depth, pm_camera::pixel_depth, Pix_Type::pixel_shape, pm_camera::pixtype, pm_camera::pm_list, Pix_Type::r2, RandFlat(), Pix_Type::reflectivity, pm_camera::telescope, and Pix_Type::transparency.

8.35.2.4 line_plane_intersection()

```
int line_plane_intersection (
```

```
            double xl[3],
            double cl[3],
            double xp[3],
            double cp[3],
            double xi[3] )
```

Intersection of a line with a plane normal to given vector.

Parameters

xl	Point along a line
cl	Direction vector along the line (not necessarily normalized to one)
xp	Point on the plane
cp	Direction vector normal to the plane
xi	Point of intersection

Returns

1: regular intersection, 0: no intersection (line parallel to plane, outside) -1: line is in plane (return xl as a valid point)

Definition at line 402 of file sim_imaging.c.
Referenced by reflect_on_polynomial_mirror().

8.35.2.5 line_ztube_intersection()

```
int line_ztube_intersection (
            double xl[3],
            double cl[3],
            double R,
            double zmin,
            double zmax,
            int fwd_only,
            double xi[3] )
```

Calculate if there is a valid intersection between a line and a tube (open cylinder) along the z axis.

Parameters

xl	Point along a line.
cl	Direction vector along the line (not necessarily normalized to one).
R	Radius of tube.
zmin	Minimum z value of valid intersection point.
zmax	maximum z value of valid intersection point.
fwd_only	If non-zero only intersections in forward direction are valid.
xi	Intersection point.

Returns

0 (no intersection), 1 (we have an intersection)

Definition at line 449 of file sim_imaging.c.
References solve_quadratic_equation().
Referenced by baffle_intersection().

8.35.2.6 make_trafo()

```
void make_trafo (
            double alpha_ang,
            double beta_ang,
            double gamma_ang,
            double x,
            double y,
            double z,
            struct transform_struct * trans )
```

Create rotation matrix and translation vector.

Parameters

alpha_ang	Rotation angle for first rotation, around z axis (e.g. azimuth) [radians].
beta_ang	Rotation angle for second rotation, around y' axis (e.g. zenith angle) [radians].
gamma_ang	Rotation angle for third rotation, around z'' axis [radians].
x	X coordinate of center of rotation.
y	Y coordinate of center of rotation.
z	Z coordinate of center of rotation.
trans	Transformation parameters filled for later forward and backward transformations.

Definition at line 1850 of file sim_imaging.c.
Referenced by tel_newdir(), and tel_setup_secondary().

8.35.2.7 offset_in_camera()

```
void offset_in_camera (
            double ref_azimuth,
            double ref_altitude,
            struct telescope_optics * optics,
            double * xoff,
            double * yoff )
```

Calculate offset of a reference sky position in the camera plane.
Note: the returned offsets are in linear units.

Parameters

ref_azimuth	Azimuth of sky position [deg]
ref_altitude	Altitude of sky position [deg]
optics	Transformation settings for telescope optics.
xoff	Filled on return with linear X offset [cm].
yoff	Filled on return with linear y offset [cm].

Definition at line 7957 of file sim_imaging.c.
References telescope_optics::altitude, telescope_optics::azimuth, telescope_optics::focal_length, and M_PI.

8.35.2.8 pm_grid_search()

```
int pm_grid_search (
            struct pm_camera * camera,
            double * photon_pos,
            double * photon_dir,
            double * xcam,
            double * ycam,
            double * sxcam,
            double * sycam,
            double * time_to_camera )
```

Faster search of PM hits on a grid over the camera front.

Parameters

cam	Camera configuration
photon_pos	Position of ray intersection with focal surface
photon_dir	Direction of incoming ray in camera system
xcam	On entry: x position in camera system; on exit: in pixel system
ycam	On entry: y position in camera system; on exit: in pixel system
sxcam	On entry: x slope in camera system; on exit: in pixel system
sycam	On entry: y slope in camera system; on exit: in pixel system
time_to_camera	On entry: time to focal surface; on exit: time to pixel

Returns

-1 (no pixel hit) or pixel ID

Definition at line 7539 of file sim_imaging.c.
References PM_List::axx, PM_List::azz, pm_camera::camera_pixel_assignment_initialized, pm_camera::curved_surface, PM_Grid::dym1, PM_Grid::field, Pix_Type::half_size, PM_GridList::list, M_PI, PM_GridList::num_pm, PM_Grid::ny, PM_List::pix_type, Pix_Type::pixel_shape, pm_camera::pixels_parallel, pm_camera::pixtype, pm_camera::pm_grid, pm_camera::pm_list, Pix_Type::r2, PM_Grid::x_high, PM_Grid::y_high, and PM_List::z.
Referenced by camera_hit().

8.35.2.9 poly_deriv_even()

```
double poly_deriv_even (
            double x,
            int npar,
            const double * params )
```

Evaluate the derivative of an even polynomial.

x The position (here usually the offset from the optical axis)

at which the derivative of the polynomial should be evaluated.

npar The number of (usually non-zero) coefficients to use in the evaluation.

param The list of even polynomial coefficients.

Returns

fprim(x)=d/dx f(x)=params[1]∗2∗x+params[2]∗4∗x∗(x∗x)+...

Definition at line 1496 of file sim_imaging.c.
Referenced by camera_hit(), camera_setup_inclined_pixels(), and reflect_on_polynomial_mirror().

8.35.2.10 poly_eval_even()

```
double poly_eval_even (
            double x,
            int npar,
            const double * params )
```

Evaluate an even polynomial.

x The position (here usually the offset from the optical axis)

at which the polynomial should be evaluated.

npar The number of (usually non-zero) coefficients to use in the evaluation.

param The list of even polynomial coefficients.

Returns

f(x)=params[0]+params[1]∗(x∗x)+params[2]∗(x∗x)∗(x∗x)+...

Definition at line 1472 of file sim_imaging.c.
Referenced by camera_setup_inclined_pixels(), and reflect_on_polynomial_mirror().

8.35.2.11 reflect_on_parabolic_mirror()

```
int reflect_on_parabolic_mirror (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
            int type,
            double phi_m,
            double * angle )
```

Reflect a photon on a parabolic mirror.

Parameters

r	radius of central curvature of mirror surface (twice its focal length)
mrad	half of mirror diameter (in case of hexagonal or square mirrors measured flat to flat)
arnd	random error in reflection angle
p	(x,y,z) starting point on entry, position of reflection point on return
d	(cx,cy,cz) direction before reflection on entry, direction after reflection (if any) on return
type	mirror type (0: circular, 1,3: hexagonal, 2: square
phi_m	Azimuth angle of mirror (actually it should be the negative azimuth angle of the dish centre in the mirror frame but is for now the azimuth angle of the mirror in the dish frame – which can be slightly different for inclined mirrors but is considered good enough to avoid artificial shadowing).
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (mirror not hit), -2 (on backside)

Note: multiple reflections are not taken into account
Definition at line 1033 of file sim_imaging.c.
References M_PI, RandGauss(), and solve_quadratic_equation().

8.35.2.12 reflect_on_polynomial_mirror()

```
int reflect_on_polynomial_mirror (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
            int type,
            double __attribute__((unused)) phi_m,
            int npar,
            double * params,
            double hole_diam,
            double * angle )
```

Reflect a photon on a general mirror with rotational symmetry described by an even polynomial shape.

Parameters

r	radius of central curvature of mirror surface (twice its focal length) [ignored ??]
mrad	half of mirror diameter (in case of hexagonal or square mirrors measured flat to flat)
arnd	random error in reflection angle
p	(x,y,z) starting point on entry, position of reflection point on return
d	(cx,cy,cz) direction before reflection on entry, direction after reflection (if any) on return
type	mirror type (must be 0: circular)
phi_m	Ignored for circular shape.
npar	Number of parameters describing polynomial
params	Parameters describing polynomial
hole_diam	Diameter of a central hole in the mirror.
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (mirror not hit), -2 (on backside)

Note: multiple reflections on the same mirror or rays reflected more than once between primary and secondary are not taken into account.
Definition at line 1559 of file sim_imaging.c.
References line_plane_intersection(), M_PI, poly_deriv_even(), poly_eval_even(), RandGauss(), and solve_↩ quadratic_equation().

8.35.2.13 reflect_on_spherical_mirror()

```
int reflect_on_spherical_mirror (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
```

```
            int type,
            double phi_m,
            double * angle )
```

Reflect a photon on a spherical mirror.

Parameters

r	radius of curvature of mirror surface (twice its focal length)
mrad	half of mirror diameter (in case of hexagonal or square mirrors measured flat to flat)
arnd	random error in reflection angle
p	(x,y,z) starting point on entry, position of reflection point on return
d	(cx,cy,cz) direction before reflection on entry, direction after reflection (if any) on return
type	mirror shape type (0: circular, 1,3: hexagonal, 2: square)
phi_m	Azimuth angle of mirror (actually it should be the negative azimuth angle of the dish centre in the mirror frame but is for now the azimuth angle of the mirror in the dish frame – which can be slightly different for inclined mirrors but is considered good enough to avoid artificial shadowing).
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (mirror not hit), -2 (on backside)

Note: multiple reflections are not taken into account
Definition at line 791 of file sim_imaging.c.
References M_PI, RandGauss(), and solve_quadratic_equation().

8.35.2.14 refract_in_fresnel_lens()

```
int refract_in_fresnel_lens (
            double r,
            double mrad,
            double arnd,
            double * p,
            double * d,
            int type,
            double n,
            double * angle )
```

Refract a photon in a thin fresnel lens (using framework from reflection in parabolic mirror)

Parameters

r	Radius of central curvature of refracting lens surface (= flen $*$ (n-1) in ideal case).
mrad	Half of lens diameter (in case of hexagonal or square lens shape measured flat to flat)
arnd	random error in resulting angle
p	(x,y,z) starting point on entry, position of point on thin lens on return (z=0).
d	(cx,cy,cz) direction before lens on entry, direction after lens (if any) on return
type	lens shape type (0: circular, 1,3: hexagonal, 2: square
n	Index of refraction of lens material (w.r.t. air). Note: focal length = r / (n-1).
angle	Pointer for the calculated local incidence angle, if requested. Otherwise NULL pointer.

Returns

0 (reflected), -1 (lens not hit), -2 (on backside)

Note: reflections in the lens etc. are not taken into account (should be part of transmission='reflectivity' curve) Boundaries between Fresnel lens steps are ignored as well as any diffraction.
Definition at line 1247 of file sim_imaging.c.
References M_PI, and RandGauss().

8.35.2.15 segment_hit()

```
int segment_hit (
            double * pos,
            int npar,
            double * poly_par,
            struct mirror_segmentation * seg_set )
```

If the primary or secondary of a dual mirror optics are marked as segmented, this function is used to decide if and which segment of a set of segments is hit.
Sets of segments consist of equal segments only rotated around the telescope axis.
We have the following supported types of segments: 0: circles, 1: hexagons, 2: squares, 4: ring segments, 5↩ : polygon segments Hits to the first three are checked after transformation into the local frame while hits to ring segments and polygon segments can be decided based on the coordinates without any transformation.
Definition at line 4601 of file sim_imaging.c.
References mirror_segmentation::cvxp, mirror_segmentation::first, convex_polygon::np, mirror_segmentation↩ ::radius, mirror_segmentation::rmax, and mirror_segmentation::type.

8.35.2.16 solve_quadratic_equation()

```
int solve_quadratic_equation (
            double a,
            double b,
            double c,
            double * x1,
            double * x2 )
```

Solve a quadratic equation a∗x∗∗2 + b∗x + c = 0.

Returns

0: OK, we have solutions, -1: no solution, -2: a=b=c=0

Definition at line 716 of file sim_imaging.c.
Referenced by baffle_intersection(), line_ztube_intersection(), reflect_on_parabolic_mirror(), reflect_on_↩ polynomial_mirror(), and reflect_on_spherical_mirror().

8.35.2.17 tel_newdir()

```
int tel_newdir (
            double phi,
            double theta,
            double tel_rand,
            struct telescope_optics * optics )
```

Set up new transformations between ground and telescope.
This function is called when the viewing direction is changed (and at configuration time).

Parameters

phi	Telescope pointing azimuth angle (N->E) as far as known [rad].
theta	Telescope pointing zenith angle as far as known [rad].

Parameters

tel_rand	Additional and not known error in telescope azimuth and altitude angles.
optics	Telescope optics parameters.

Definition at line 2121 of file sim_imaging.c.
References telescope_optics::alt_optics_offset, telescope_optics::az_alt_offset, make_trafo(), transform_off_struct::offset0, transform_off_struct::offset1, transform_off_struct::offset2, RandGauss(), transform_struct::rot, transform_off_struct::rot1, transform_off_struct::rot2, transform_off_struct::simple, transform_off_struct::strmat, and telescope_optics::tel_trans.
Referenced by tel_setup_secondary().

8.35.2.18 tel_setup_primary()

```
int tel_setup_primary (
            struct telescope_optics * optics,
            double phi,
            double theta )
```

Set up all the transformations between ground, telescope, mirror, and camera reference frames, for telescopes with the camera in the primary focus (mirror_class 0 or 1).
@paran optics Optics parameters

Parameters

phi	Azimuth angle (N->E) [rad]
theta	Zenith angle [rad]

< Focal length of the optical system (for image scale) [cm]
< Dish shape scale length; = radius of curvature (half of r.o.c.) for DC (parabolic) [cm]
< Focal length of mirror facets without explicit value in config file [cm]
< Grading of mirror facet focal length (outer mirror f minus inner mirror f) [cm]
< Two components: R.m.s. and maximum spread of assigned mirror focal lengths (incl. neg. f value in config file) [cm]
< 0 for Davies-Cotton, 1 for parabolic dish shape.
Definition at line 2239 of file sim_imaging.c.

8.35.2.19 trace_photon_in_fresnel()

```
int trace_photon_in_fresnel (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance )
```

Trace a photon through the whole telescope to the camera (if hit).
This variant is for a Fresnel lens.
Definition at line 3750 of file sim_imaging.c.

8.35.2.20 trace_photon_in_paraboloid()

```
int trace_photon_in_paraboloid (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance )
```

Trace a photon through the whole telescope to the camera (if hit).
This variant is for a parabolic single mirror.
Definition at line 3408 of file sim_imaging.c.

8.35.2.21 trace_photon_in_segmented()

```
int trace_photon_in_segmented (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance )
```

Trace a photon through the whole telescope to the camera (if hit).
This variant is for the usual segmented mirror setup with dish shape ranging from Davies-Cotton to parabolic, always with spherical mirror tiles.
Definition at line 2717 of file sim_imaging.c.

8.35.2.22 trace_photon_with_secondary()

```
int trace_photon_with_secondary (
            struct telescope_optics * optics,
            double * photon_pos,
            double * photon_dir,
            double * x_on_camera,
            double * y_on_camera,
            double * sx_camera,
            double * sy_camera,
            double * time_to_camera,
            int * mirror,
            double * releff,
            double rlambda,
            double distance,
            int bypass )
```

Trace a photon through the whole telescope to the camera (if hit).

This variant is for a secondary mirror optics with primary and secondary mirrors in one piece each. Both mirros can have rather general rotation-symmetric shapes, defined by even polynomial coefficients.
Definition at line 4963 of file sim_imaging.c.

8.35.2.23 transform()

```
void transform (
            double * p,
            double * d,
            const struct transform_struct * ts,
            int mode )
```

Rotation plus translation transformation (forw.
and backw.)

Parameters

p	Position 3-D vector (offset to be applied). Rewritten.
d	Direction 3-D vector (only rotated). Rewritten.
ts	Transformation structure (with only one offset).
mode	Use +1 for forward, -1 for backward transformation.

Definition at line 1955 of file sim_imaging.c.
References transform_struct::offset, and transform_struct::rot.
Referenced by transform_off().

8.35.2.24 transform_off()

```
void transform_off (
            double * p,
            double * d,
            const struct transform_off_struct * ts,
            int mode )
```

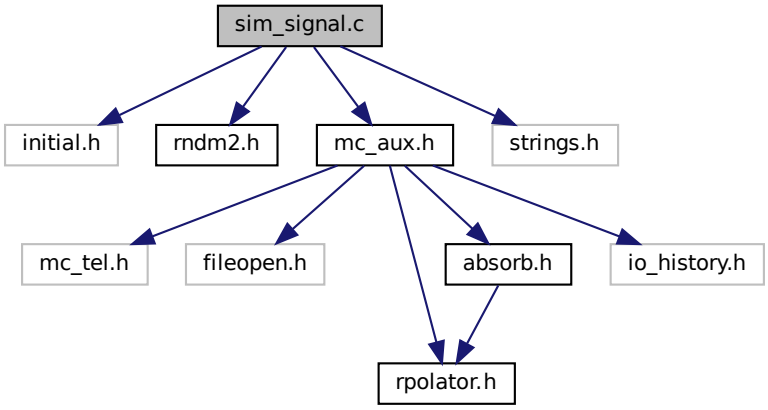
Transformation with extra offsets between rotation axes (forw.
and backw.)
Definition at line 2019 of file sim_imaging.c.
References transform_off_struct::offset0, transform_off_struct::offset1, transform_off_struct::offset2, transform_off_struct::rot1, transform_off_struct::rot2, transform_off_struct::simple, transform_off_struct::strmat, and transform().

8.36 sim_signal.c File Reference

Simulate signals and response of electronics.

```
#include "initial.h"
#include "rndm2.h"
#include "mc_aux.h"
#include <strings.h>
```

Include dependency graph for sim_signal.c:



Data Structures

- struct pm_and_fadc_temporary

  Temporary data for a PMT and its electronics channels which.

Macros

- #define **DISC_OFFSET_INIT** 1
- #define **FILL_TRACES** 1
- #define **FIND_GLOBAL_PEAK** 1

Functions

- int read_spe (char ∗fname, double ∗∗xspe_prompt, double ∗∗yspe_prompt, double ∗∗xspe_bkgrnd, double ∗∗yspe_bkgrnd, int ∗nspe_prompt, int ∗nspe_bkgrnd, int afterpulse_alt, double ∗norm_fact)

  Read the single photo-electron response distribution of PMs.
- int read_pulse_shape (int num_gains, char ∗fname_fadc, char ∗fname_disc, double binwidth, double ∗fshape, double ∗fshape_lg, int ∗flength, double ∗farea, double ∗farea_lg, double ∗bshape, double ∗bshape_lg, int ∗blength, double ∗barea, double ∗barea_lg, double ∗dshape, int ∗dlength, double ∗darea)

  Read the shapes of pulses at the FADCs and at the discriminator inputs.
- void init_pm_electronics (struct pm_and_fadc_channel ∗ch, struct camera_electronics ∗el, double fadc_amp, double fadc_amp_lg, double disc_amp, double qe_var, double g_var, double hglg_var, int flatfielding, double transit, double transit1, double v1frac, double voltage_var, double gain_index, double transit_error, double transit_calib_error, double transit_comp_step, double transit_comp_err, double pedestal, double pedestal_dev, double pedestal_var, double pedestal_err, double pedestal_lg, double pedestal_dev_lg, double pedestal_var_lg, double pedestal_err_lg, double sensitivity, double sens_var, double sensitivity_lg, double sens_var_lg, double background, double gain, double disc_threshold, double disc_var, double gate_length_bins, double gate_length_var_bins, double gate_delay_bins, double gate_delay_var_bins, double disc_sigsum_bins, double disc_sigsum_var_bins, double trigger_current_limit, int fadc_per_channel)

  Photo-multiplier and electronics behaviour is set up.
- double delay_signals (struct camera_electronics ∗el)

  Common delay added to all photon arrival times.
- void laser_calib_eval (struct camera_electronics ∗el)

  Evaluate calibration parameters which would normally be obtained from measurements with lasers or LEDs (incl.

- static void fill_traces (struct camera_electronics *el, struct pm_and_fadc_channel *ch, double sfact, double t, double *A, int bkg_flag, double *rawsignal, double *discriminator)

  Function for adding up pulse traces by a photo-electron (either signal or background) in all analog and digitized channels.
- void create_pm_signals (int *pe_counts, int *istart, double *atimes, double *aamp, double t0, struct camera_electronics *el)

  Create the photomultiplier signals in FADC and at discriminator for one telescope.
- void pulse_shape_analysis (struct camera_electronics *el)

  Timing analysis of the digitized pulse shape (in reality by FPGA or signal processor).
- void telescope_trigger (struct pm_camera *cam, struct camera_electronics *el)

  Find out if (and when) a single telescope is triggered.
- void array_trigger (struct telescope_array *array, struct camera_electronics *elec, int iarray)

  Find out if (and when) the array of telescopes is triggered.
- void sum_adc_bins (struct telescope_array *array)

  Calculate (F)ADC sums in a time interval common to all channels.
- int read_shape (const char *fname, double *shape, int maxlen, double timestep, double scale, int *shape_len, int *toffsteps)

  Read a pulse shape given in arbitrary (but increasing) time steps and rebin it to the desired steps (typically ADC clock cycle or a multiple thereof.

Variables

- static disc_data_t **bit** [DISC_BITS_PER_BIN]
- struct pm_and_fadc_temporary **temp_channels** [MAX_PIXELS]

8.36.1 Detailed Description

Simulate signals and response of electronics.
This file contains code to simulate the response of photomultiplier, Flash-ADCs and discriminators to the incident Cherenkov and night-sky light. Telescope and array trigger decisions are also simulated.

Author

Konrad Bernloehr

8.36.2 Function Documentation

8.36.2.1 array_trigger()

```
void array_trigger (
            struct telescope_array * array,
            struct camera_electronics * elec,
            int iarray )
```

Find out if (and when) the array of telescopes is triggered.
There may be multiple conditions for an array trigger and if any of them is met the array is declared triggered. The data recorded would include any extra triggered telescopes even if not part of an array trigger condition - with two exceptions: a) telescopes only listed in 'hardware stereo' conditions get their 'telescope_triggered' flag reset if the hardware stereo group did not trigger; b) alternate mono triggers from random or muon-ring selections would ask data from other telescopes to be suppressed if the alternate mono condition is the only accepted trigger. This function sets values in the provided telescope_array struct.

Parameters

array	Pointer to the data for the entire array of telescopes.
elec	Pointer to the camera electronics data for all telescopes.
iarray	Which of multiple randomly offset arrays are we testing for?

Definition at line 3771 of file sim_signal.c.
References camera_electronics::central_time.

8.36.2.2 create_pm_signals()

```
void create_pm_signals (
            int * pe_counts,
            int * istart,
            double * atimes,
            double * aamp,
            double t0,
            struct camera_electronics * el )
```

Create the photomultiplier signals in FADC and at discriminator for one telescope.

Parameters

pe_counts	The numbers of Cherenkov (or laser/LED) photo-electrons in each pixel (no NSB).
istart	The offset in the global p.e. times list where p.e.s of each pixel are starting (cumulative pe_counts of earlier pixels).
atimes	The list of times of all signal p.e.s in the camera (arrival time at the photo sensor). In [ns] since the time the photons would have crossed the CORSIKA observation level (with t=0 when the primary particle would have crossed that level).
aamp	NULL or array like atimes to keep the random p.e. amplitudes.
t0	The median time of all p.e.s minus a (20+x)% fraction of the readout window, so that atimes[i]-t0 is typically early in the common readout window.
el	Pointer to all the camera electronics details of the telescope.

Definition at line 1480 of file sim_signal.c.
References MAX_FADC_BINS, and pm_and_fadc_channel::signal.

8.36.2.3 delay_signals()

```
double delay_signals (
            struct camera_electronics * el )
```

Common delay added to all photon arrival times.
Add a common delay to all photons at one telescope because shower arrival is not correlated with the FADC clock. This function should be called exactly once per event.
Definition at line 1169 of file sim_signal.c.
References camera_electronics::fadc_bins, camera_electronics::fadc_delay, camera_electronics::fadc_per_↩channel, camera_electronics::interval, camera_electronics::phase_delay, camera_electronics::photon_delay, and RandFlat().

8.36.2.4 fill_traces()

```
static void fill_traces (
            struct camera_electronics * el,
            struct pm_and_fadc_channel * ch,
            double sfact,
            double t,
            double * A,
            int bkg_flag,
            double * rawsignal,
            double * discriminator )  [static]
```

Function for adding up pulse traces by a photo-electron (either signal or background) in all analog and digitized channels.

The photo-electron is defined by the "arrival" time and a prompt/background option that determines from which distribution the pulse peak amplitude is randomly generated.

Parameters

el	Pointer to all the camera electronics details. @paran ch Pointer to the electronics details for the involved pixel.
sfact	NSB-dependent amplitude scaling factor.
t	Time of arrival of photon (production of p.e.) on pixel.
A	Pointer to which random peak amplitude gets copied, if not NULL.
bkg_flag	Prompt/background handling flag. 0: prompt signal, 1: background with amplitude from same distribution as signal, 2: background with amplitude from folded distribution, 3: extra afterpulse signal from dedicated afterpulse distribution,
rawsignal	Array for be digitized signal (high-gain), before digitization.
rawsignal_lg	Same for low-gain channel. Only if low gain enabled at compile-time.
discriminator	Array for the discriminator/comparator/analog sum input signal.

Definition at line 1325 of file sim_signal.c.
References pm_and_fadc_channel::cherenkov_pe, pm_and_fadc_channel::disc_amplitude, camera_electronics::disc_start, pm_and_fadc_channel::fadc_amplitude, camera_electronics::fadc_bins, camera_electronics::full_simulation, pm_and_fadc_channel::ideal_signal, camera_electronics::interval, camera_electronics::nspe_bkgrnd, camera_electronics::nspe_prompt, camera_electronics::num_gains, OVERSAMPLING, pm_and_fadc_channel::pixel_sat_coeff, RandExponential(), RandFlat(), RandGauss(), random_from_ipol_table(), random_from_table(), camera_electronics::shape, camera_electronics::shape_length, camera_electronics::transit_time_jitter, camera_electronics::with_analog_trigger, camera_electronics::xspe_bkgrnd, camera_electronics::xspe_prompt, camera_electronics::yspe_bkgrnd, and camera_electronics::yspe_prompt.

8.36.2.5 init_pm_electronics()

```
void init_pm_electronics (
            struct pm_and_fadc_channel * ch,
            struct camera_electronics * el,
            double fadc_amp,
            double fadc_amp_lg,
            double disc_amp,
            double qe_var,
            double g_var,
            double hglg_var,
            int flatfielding,
            double transit,
            double transit1,
            double v1frac,
            double voltage_var,
            double gain_index,
            double transit_error,
            double transit_calib_error,
            double transit_comp_step,
            double transit_comp_err,
            double pedestal,
            double pedestal_dev,
            double pedestal_var,
            double pedestal_err,
            double pedestal_lg,
            double pedestal_dev_lg,
            double pedestal_var_lg,
```

```
            double pedestal_err_lg,
            double sensitivity,
            double sens_var,
            double sensitivity_lg,
            double sens_var_lg,
            double background,
            double gain,
            double disc_threshold,
            double disc_var,
            double gate_length_bins,
            double gate_length_var_bins,
            double gate_delay_bins,
            double gate_delay_var_bins,
            double disc_sigsum_bins,
            double disc_sigsum_var_bins,
            double trigger_current_limit,
            int fadc_per_channel )
```

Photo-multiplier and electronics behaviour is set up.
Initialize constants needed for the simulation of the response of one channel (photo-multiplier, flash-ADC, discriminator, DC current)
Definition at line 702 of file sim_signal.c.

8.36.2.6 laser_calib_eval()

```
void laser_calib_eval (
            struct camera_electronics * el )
```

Evaluate calibration parameters which would normally be obtained from measurements with lasers or LEDs (incl. flatfielding).
Definition at line 1189 of file sim_signal.c.
Referenced by convert_initial_moni_calib().

8.36.2.7 pulse_shape_analysis()

```
void pulse_shape_analysis (
            struct camera_electronics * el )
```

Timing analysis of the digitized pulse shape (in reality by FPGA or signal processor).
Note that only the high-gain channel is processed if several gains are available.
Definition at line 2791 of file sim_signal.c.
References pm_and_fadc_channel::calib, camera_electronics::fadc_bins, camera_electronics::fadc_per_channel, camera_electronics::global_peak_pos, camera_electronics::groups_triggered, camera_electronics::interval, camera_electronics::long_event, camera_electronics::longsum_bins, camera_electronics::longsum_offset, MAX_FADC_BINS, camera_electronics::num_gains, camera_electronics::peak_frac, pm_and_fadc_channel::peak_pos, channel_calibration::pedestal, camera_electronics::pixels, camera_electronics::pulse_analysis, pm_and_fadc_channel::pulse_rise, pm_and_fadc_channel::pulse_sum_glob, pm_and_fadc_channel::pulse_sum_loc, pm_and_fadc_channel::pulse_t_over_thr, pm_and_fadc_channel::pulse_width, pm_and_fadc_channel::signal, camera_electronics::sum_after_peak, camera_electronics::sum_before_peak, camera_electronics::sum_bins, camera_electronics::sum_offset, camera_electronics::sum_start, camera_electronics::telescope, and camera_electronics::trigger_time.

8.36.2.8 read_pulse_shape()

```
int read_pulse_shape (
            int num_gains,
            char * fname_fadc,
            char * fname_disc,
            double binwidth,
```

```
            double * fshape,
            double * fshape_lg,
            int * flength,
            double * farea,
            double * farea_lg,
            double * bshape,
            double * bshape_lg,
            int * blength,
            double * barea,
            double * barea_lg,
            double * dshape,
            int * dlength,
            double * darea )
```

Read the shapes of pulses at the FADCs and at the discriminator inputs.
There are two possible FADC pulse shape formats: binned and unbinned. Each of them has to include a low-gain pulse shape if compiled WITH_LOW_GAIN_CHANNEL. Binned and unbinned format are distinguished by the presence or absence of a comment '# T=...' with the corresponding bin step in nanoseconds. The unbinned format requires an initial first column for the time [ns] corresponding to each line. The low-gain pulse is always the last of two or three columns (in binned or unbinned format, respectively).
The signal at the discriminator or comparator input is always in unbinned format, i.e. with leading time column.

Parameters

fname_fadc	File name with pulse shapes for digital signal (one or two gains)
fname_disc	Input signal at discriminator or comparator.
binwidth	Time length of (F)ADC sampling interval [ns].
fshape	Memory address where to store high-gain FADC pulse shape.
fshape_lg	Memory address where to store low-gain FADC pulse shape.
flength	Write common length of FADC pulse shapes.
farea	Write pulse area [ns] of HG pulse at amplitude=1.
farea_lg	Write pulse area [ns] of LG pulse at amplitude=1.
bshape	Like fshape but for background (NSB) signals.
bshape_lg	Like fshape_lg but for background (NSB) signals.
blength	Like flength but for background (NSB) signals.
barea	Like farea but for background (NSB) signals.
barea_lg	Like farea_lg but for background (NSB) signals.
dshape	Memory address where to store pulse shape at discriminator/comparator input.
dlength	Write length of discr./comp. pulse shape.
darea	Write pulse area [ns] of discr./comp. pulse at amplitude=1.

Returns

0 (OK), -1 (error)

Definition at line 232 of file sim_signal.c.
References MAX_SHAPE_LENGTH, OVERSAMPLING, rpol_cspline(), rpol_linear(), and set_1d_cubic_params().

8.36.2.9 read_spe()

```
int read_spe (
            char * fname,
            double ** xspe_prompt,
            double ** yspe_prompt,
            double ** xspe_bkgrnd,
```

```
            double ** yspe_bkgrnd,
            int * nspe_prompt,
            int * nspe_bkgrnd,
            int afterpulse_alt,
            double * norm_fact )
```

Read the single photo-electron response distribution of PMs.

Parameters

fname	Name of the file from which to read.
xspe_prompt	Amplitude values (mean p.e. units) for prompt response.
yspe_prompt	Probability of prompt response at give amplitudes.
xspe_bkgrnd	Amplitude values (mean p.e. units) for background response.
yspe_bkgrnd	Probability of background response (including afterpulsing) at give amplitudes.
nspe_prompt	Number of data points for prompt response.
nspe_bkgrnd	Number of data points for background response.
afterpulse_alt	If non-zero, use prompt values also for background. Afterpulsing has to be added separately later then.
norm_fact	The mean amplitude per single p.e. actually generated (should be close to 1.0).

Returns

0 (OK), -1 (error)

Definition at line 77 of file sim_signal.c.
References make_random_ipol_table(), make_random_table(), random_from_ipol_table(), and random_from_table().

8.36.2.10 sum_adc_bins()

```
void sum_adc_bins (
            struct telescope_array * array )
```

Calculate (F)ADC sums in a time interval common to all channels.
Alternatively, this can be the peak amplitude instead of the sum, depending on the peak_sensing option. The start of the integration window is determined by the trigger. This function also checks for overflows.
Definition at line 4196 of file sim_signal.c.
References telescope_array::electronics, camera_electronics::fadc_bins, camera_electronics::fadc_max_signal, camera_electronics::fadc_max_sum, camera_electronics::groups_triggered, camera_electronics::interval, camera_electronics::long_event, camera_electronics::longsum_bins, camera_electronics::longsum_offset, telescope_array::ntel, telescope_array::options, pm_and_fadc_channel::overflow, mc_tel_options::peak_sensing, camera_electronics::pixels, pm_and_fadc_channel::signal, camera_electronics::simulated, pm_and_fadc_channel::sum_adc, pm_and_fadc_channel::sum_bins, camera_electronics::sum_bins, camera_electronics::sum_offset, camera_electronics::sum_start, mc_options::tel_options, camera_electronics::telescope, and camera_electronics::trigger_time.

8.36.2.11 telescope_trigger()

```
void telescope_trigger (
            struct pm_camera * cam,
            struct camera_electronics * el )
```

Find out if (and when) a single telescope is triggered.
The trigger decision is made up for any number of trigger groups. Trigger groups can overlap in the list of pixels contributing. Different schemes/algorithms are available: majority/analog sum/digital sum. The telescope trigger time is the earliest time of any trigger group fired. This function sets values in the provided camera electronics struct.

Parameters

cam	Poiter to geometric and optical parameters of the camera.
el	Pointer to all the camera electronics details of the telescope.

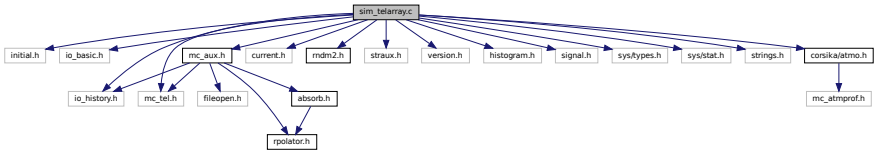
Definition at line 3095 of file sim_signal.c.

8.37 sim_telarray.c File Reference

This file contains the main program of the telescope simulation.

```
#include "initial.h"
#include "io_basic.h"
#include "io_history.h"
#include "current.h"
#include "mc_tel.h"
#include "rndm2.h"
#include "mc_aux.h"
#include "straux.h"
#include "version.h"
#include "histogram.h"
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <strings.h>
#include "corsika/atmo.h"
```

Include dependency graph for sim_telarray.c:



Macros

- #define Nint(x) ((x)>0?(int)((x)+0.5):(int)((x)-0.5))

  If defined we skip to the event end first to read profiles.
- #define **min**(a, b) ((a)<(b)?(a):(b))
- #define **max**(a, b) ((a)>(b)?(a):(b))
- #define **Abs**(x) ((x)>=0?(x):-1∗(x))
- #define **square**(x) ((x)∗(x))
- #define **_XSTR_**(s) _STR_(s)
- #define **_STR_**(s) #s
- #define **SHOW_MACRO**(s)
- #define **FTELL**(f) ftell(f)
- #define **FSEEK**(f, o, p) fseek(f,o,p)
- #define **OFF_T** long
- #define **SKIP_WITH_TOO_MANY_PE**

Functions

- void **dhsort** (double ∗d, int n)
- void stop_signal_function (int isig)

Stop the program gracefully when it catches an INT or TERM signal.

- void hup_signal_function (int isig)

  Produce intermediate output when the program catches an HUP signal.
- static void **get_aprof** (void)
- double rhof_ (double *height)

  Density of atmosphere [g/cm^3] at given height a.s.l.
- double thick_ (double *height)

  'Thickness' (column density) of atmosphere [g/cm^2] from space down to given height a.s.l.
- double heigh_ (double *thick)

  Inverse of thick() function.
- double refim1_ (double *height)

  Index of refraction minus one at given height a.s.l.
- double refid_ (double *height)

  Index of refraction at given height a.s.l.
- double Square (double x)

  Function returning the square of its argument (instead of a macro).
- double **find_max_pos** (double *y, int n)
- void sim_calib_events (int events, int laser, int open, struct telescope_array *array, double *atimes, double *aamp, long maxpe)

  Simulate calibration events in the Cherenkov telescopes.
- void set_simtel_prog_path (char *prg)

  Set the name and path under which sim_telarray was apparently started.
- const char * get_simtel_prog (void)

  Retrieve the name under which sim_telarray was apparently started.
- const char * get_simtel_prog_path (void)

  Retrieve the program path under which sim_telarray was apparently started.
- void report_command_line (int argc, char **argv)

  Report the complete command line how sim_telarray got started.
- void report_compiled (void)

  Report relevant compile-time definitions.
- void report_batch_job (void)

  Show batch job relevant environment variables, initially only for (former) Sun Grid Engine (SGE) and now also including SLURM.
- int report_env (const char *name)

  Report an environment variable, if it was set.
- int report_env_vars (void)

  Report environment variables which may change the behavior of sim_telarray.
- int main (int argc, char **argv)

  Main program of Cherenkov telescope simulation.

Variables

- static size_t max_bunches = MAX_BUNCHES

  The default maximum number of bunches can later be adapted to CORSIKA.
- static size_t **max_pe** = MAX_PHOTOELECTRONS
- static size_t **max_pix_pe** = MAX_PIXEL_PHOTOELECTRONS
- static size_t **max_particles** = MAX_PARTICLES
- static int interrupted = 0

  Set non-zero after catching an interrupt or terminate signal.
- struct linked_string corsika_inputs

  A sequence of input file names with CORSIKA IACT data.
- double airlightspeed = 29.9792458/1.0002201

The speed of light used for light propagation at the observation level.

- int **global_verbose** = 0
- static AtmProf * caprof = NULL

  Local copy of pointer to common atmospheric profile.
- static char * simtel_prog = NULL

  Keep record of the name under which sim_telarray was started.
- static char * **simtel_prog_path** = NULL

8.37.1 Detailed Description

This file contains the main program of the telescope simulation.

Author

Konrad Bernloehr

8.37.2 Macro Definition Documentation

8.37.2.1 Nint

```
#define Nint(
              x ) ((x)>0?(int)((x)+0.5):(int)((x)-0.5))
```

If defined we skip to the event end first to read profiles.
(only needed for older hacked CORSIKA version).
Definition at line 111 of file sim_telarray.c.

8.37.2.2 SHOW_MACRO

```
#define SHOW_MACRO(
              s )
```

Value:

```
    if ( strcmp(#s,_XSTR_(s)) != 0 ) \
    { if ( strcmp("", _XSTR_(s)) != 0 && strcmp("1", _XSTR_(s)) != 0 ) \
       printf( " " #s "=" _XSTR_(s) ); else printf( " " #s ); }
```

8.37.3 Function Documentation

8.37.3.1 heigh_()

```
double heigh_ (
            double * thick )
```

Inverse of thick() function.
The CORSIKA built-in function for the height as a function of overburden.
Definition at line 182 of file sim_telarray.c.

8.37.3.2 main()

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

Main program of Cherenkov telescope simulation.
For options and arguments see init_setup().
Definition at line 1072 of file sim_telarray.c.
References iobuf, and pm_camera::itel.

8.37.3.3 refid_()

```
double refid_ (
            double * height )
```

Index of refraction at given height a.s.l.
[cm].
Definition at line 210 of file sim_telarray.c.
References refim1_().

8.37.3.4 refim1_()

```
double refim1_ (
            double * height )
```

Index of refraction minus one at given height a.s.l.
[cm].
Definition at line 196 of file sim_telarray.c.
Referenced by refid_().

8.37.3.5 report_compiled()

```
void report_compiled (
            void  )
```

Report relevant compile-time definitions.
The following shows the same output after a "#define XYZ" and a "#define XYZ 1" (or compiled with "-DXYZ")
Other non-empty, not "1" assigned values are shown explicitly.
Undefined macros are not shown. Neither are any assigned to itself.
Definition at line 671 of file sim_telarray.c.

8.37.3.6 rhof_()

```
double rhof_ (
            double * height )
```

Density of atmosphere [g/cm^3] at given height a.s.l.
The CORSIKA built-in density lookup function.
[cm].
Definition at line 153 of file sim_telarray.c.

8.37.3.7 sim_calib_events()

```
void sim_calib_events (
            int events,
            int laser,
            int open,
            struct telescope_array * array,
            double * atimes,
            double * aamp,
            long maxpe )
```

Simulate calibration events in the Cherenkov telescopes.
This can be pedestal events (lid opened or closed) and laser events. (Updating of calibration constants needs to be implemented. This function is not presently used by default.)

Parameters

events	Number of events of same type to be simulated
laser	1 for laser events, 0 otherwise

Parameters

open	1 if camera lid is open, 0 if closed
array	structure of internal array configuration
atimes	buffer space for temporarily storing photo-electron times
aamp	buffer space for temporarily storing photo-electron amplitudes (may be NULL if not interested in that)
maxpe	The maximum number of p.e. supported by the supplied buffer(s)

Returns

(none)

Definition at line 289 of file sim_telarray.c.

8.37.3.8 stop_signal_function()

```
void stop_signal_function (
            int isig )
```

Stop the program gracefully when it catches an INT or TERM signal.

Parameters

isig	Signal number. Ultrix and DECunix only:
code	Code for exceptions or zero (not used).
scp	Context at time of signal (not used).

Returns

(none)

Definition at line 4301 of file sim_telarray.c.
References interrupted.
Referenced by hup_signal_function().

8.37.3.9 thick_()

```
double thick_ (
            double * height )
```

'Thickness' (column density) of atmosphere [g/cm^2] from space down to given height a.s.l.
The CORSIKA built-in function for vertical atmospheric thickness (overburden).
[cm].
Definition at line 168 of file sim_telarray.c.

8.38 simtel_doc.h File Reference

Add an introduction to doxygen-generated documentation.

8.38.1 Detailed Description

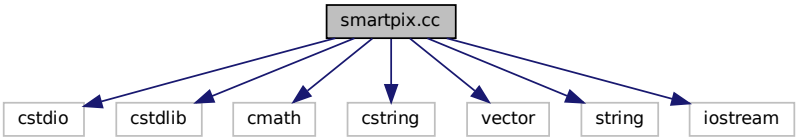
Add an introduction to doxygen-generated documentation.
This file is not included during compilation.

8.39 smartpix.cc File Reference

Generate pixel list for SmartPixels camera.

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <cstring>
#include <vector>
#include <string>
#include <iostream>
```

Include dependency graph for smartpix.cc:



Data Structures

- class Trigger
- class Pixel

Functions

- double **sq** (double x)
- int **main** (int argc, char **argv)

8.39.1 Detailed Description

Generate pixel list for SmartPixels camera.

Author

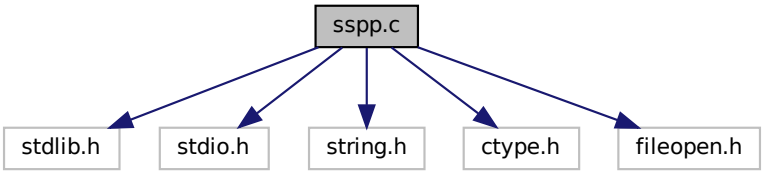
Konrad Bernloehr

8.40 sspp.c File Reference

Simple selection pre-processor.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "fileopen.h"
```

Include dependency graph for sspp.c:



Macros

- #define **WITH_FILEOPEN** 1

Functions

- void **syntax** (int argc, char ∗∗argv)
- int **main** (int argc, char ∗∗argv)

8.40.1 Detailed Description

Simple selection pre-processor.
Instead of the extensive preprocessing capabilities of 'pfp', this preprocessor can only select global/per-telescope configuration lines like those generated with 'sim_telarray ... -C list=no-internal ...'. It accepts all of the options accepted by pfp but ignores all but '-DTELESCOPE=...' and '-DPREFIX=...'. Each line is supposed to start with a tab-separated field that is used for the selection if the rest of the line is copied to the output or ignored.

Author

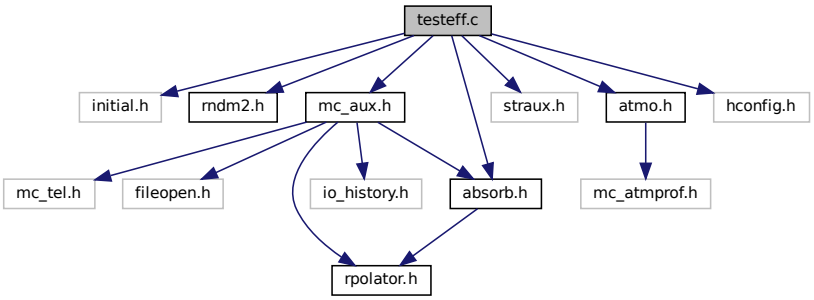
Konrad Bernloehr

8.41 testeff.c File Reference

Combine optical and sensor efficiencies as used in the telescope simulation.

```
#include "initial.h"
#include "rndm2.h"
#include "mc_aux.h"
#include "straux.h"
#include "atmo.h"
#include "absorb.h"
#include "hconfig.h"
```

Include dependency graph for testeff.c:



Data Structures

- struct effcalc

  Collect original and derived values in nanometer steps, even if (presently) not wavelength dependent.

Macros

- #define **MAX_SPE** 2000
- #define **MAX_ALT** 1000
- #define **MAX_WL** 2000
- #define **MAX_MIRRORS** 4000

Typedefs

- typedef struct effcalc **EffCalc**

Functions

- double **rhof_** (double __attribute__((unused)) *height)
- double **thick_** (double __attribute__((unused)) *height)
- double **heigh_** (double __attribute__((unused)) *thick)
- double nsb_za_scale (double airmass)

  Primitive scaling of NSB with zenith angle.
- void **strip_comments** (char *line)
- int read_spe_check (const char *fname)

  Read single-p.e.
- void **syntax** (int argc, char **argv)
- int main (int argc, char **argv)

Variables

- double **airlightspeed** = 29.9792458/1.0002256
- int **global_verbose** = 0

8.41.1 Detailed Description

Combine optical and sensor efficiencies as used in the telescope simulation.
Sim_telarray reads a number of tables specifying the wavelength dependence and, in some cases, also the angular dependence of various telescope components in the path of the incoming light. Instead of the detailed simulation, this program just interpolates the relevant tables and multiplies the efficiencies involved. Where angles of incidence get involved, they are only evaluated for the angles from the center of each mirror tile to the center of a camera. Dual-mirror telescopes have the distribution of angles represented via a set of virtual mirror tiles set up just to represent the distribution seen in full simulation. In dual-mirror configurations with significant angular dependence of reflectivities and camera, results of testeff should be used with caution.
Effects of 'shadowing' due to telescope structural elements in the path of incoming photons are totally neglected here. For single-reflector telescopes, a simplified calculation of NSB 'albedo' contributions, due to photons reflected on the ground and hitting the sensors from just beyond the edge of the mirrors is included.

Author

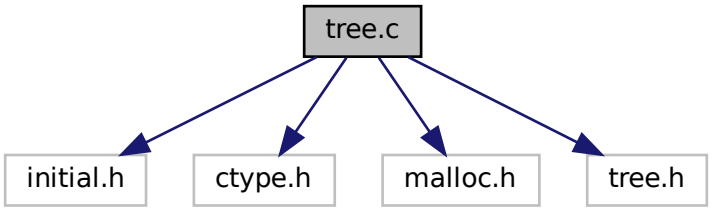
Konrad Bernloehr

8.42 tree.c File Reference

Binary tree management for pfp preprocessor defines.

```
#include "initial.h"
#include <ctype.h>
#include <malloc.h>
#include "tree.h"
```

Include dependency graph for tree.c:



Functions

- void **fatal** (const char ∗s, int r)
- char ∗ **strsave** (const char ∗s)
- TREE ∗ **tree** (TREE ∗p, const char ∗w)
- TREE ∗ **listdown** (TREE ∗p, const char ∗w)
- TREE ∗ **listup** (TREE ∗p)
- void **treeprint** (TREE ∗p, char ∗(∗func)(TREE ∗))
- char ∗ **print_tree_count** (TREE ∗p)
- char ∗ **print_tree_defined** (TREE ∗p)
- void **del_tree** (TREE ∗p)

8.42.1 Detailed Description

Binary tree management for pfp preprocessor defines.

Author

Konrad Bernloehr

Index