MAGIC-TDAS 03-06
030430/OBlanch

How To Use the camera simulation program 0.7

O Blanch
<blanch@ifae.es>
A Moralejo
<moralejo@pd.infn.it>

April 30, 2004

Abstract

This is the user guide of the Montecarlo simulation of the MAGIC camera, version 0.7

Contents

1 Introduction

The version 0.7 of the MAGIC camera simulation has ben written by Oscar Blanch and Abelardo Moralejo. It is based on previous versions, the first of which was written by Jose Carlos González. The original code has been enhanced with modifications and improvements by Oscar Blanch, Dirk Petry, Harald Kornmayer and Raul Orduña. This program is able to process files created with the Reflector simulation program [8]. In addition, files from the StarField adder program [3] can also be read in to account for the influence of individual stars in the FOV. The camera program simulates the photomultiplier and the behaviour of the trigger and FADC system. The output of the program is in the same format as the merpped real data (except of course for the addditional containers carrying the MC information).

In this version 0.7, the option of multi-telescope simulation has been added. One can simulate the response for several telescopes with different trigger conditions, different quantum efficiencies..., as long as one reflector file is supplied for each telescope. A drawback of the current simulation chain is that Reflector 0.6 does not write to its output the information on the telescope positions in the Corsika coordinate system, and hence this information must be supplied by the user in the input card of camera (to be fixed in the future). This does not affect in any sense the single telescope simulation.

2 Installation

First you should get the current version of the program. The easiest way to get it is from the web page:

http://hegra1.mppmu.mpg.de/MAGICWeb/private/montecarlo.

This page is internal and therefore you will need a password. There you have to choose Download. You will find the current version of the camera program for MAGIC and you can get a zip file with all the needed code. From here on, we will explain the procedure to get the camera program running from this zip file. If you got the code from somewhere else you would have to make sure that you got all the needed code (see Appendix B) and then follow the instructions from point 3.

1. Decompress the file using:

   gunzip camera*.gz

2. Unpack the tar file with:

   tar xvf camera*.tar

3. Change to Camera directory:

   cd MagicProgs/Simulation/Detector/Camera

4. Run the Camera Installation script:

   camera_install

5. Check the dependencies with the operating system. The program is supposed to run under osf and Linux.

   make depend

Do not worry if you get some warnings. If an error occurs this may be due to the fact, that there is no configuration file ready for the system variable "$OSTYPE". You will need to create a file called config.mk.$OSTYPE with the right configuration for your system. A first try could be just copy one of the files that already exists for a similar system (for instance: cp config.mk.linux config.mk.linux-gnu).

6. Compile the program using:

   make all

7. You may get some compilation warnings at this stage too, but they are not fatal. Compilation should end with:

   Compiling creadparam.cxx
   Compiling camera.cxx
   Linking camera ...
   Done.

   After this step the program is ready to run but you will need some input files that are described in the next section. Check that you have all the needed input files and that they are right. Run the program with: camera < input.card

3 Needed files

The camera program needs several input files to do its job.

1. The input card (called "input.card" in the directory of the camera program) is used to control the job. In section 4 this input.card is extensively described.

2. A reflector file contains the information of the simulated photons that hit the camera plane. This is the main input for the processing of camera. The actual reflector file that one is going to use must be specified in the input card. You can find links to MC reflector files produced by the different MAGIC groups at:

   `http://hegra1.mppmu.mpg.de/MAGIC/private/montecarlo/productionstatus.html`

3. The quantum efficiency for the photo-detectors as a function of the wavelength is read from a file given in the input.card and which should have a certain format. There are several files, for different photosensor assumptions. They are located in the directory MagicProgs/Simulation/Detector/Data:

   - "qe-emi.RFL.dat" for the EMI photomultipliers, which were supposed to be used in the first phase of MAGIC.
   - "qe-emi-coat.RFL.dat" for the EMI photomultipliers with the diffusor and wavelength shifter, as presently installed in the MAGIC camera [1].
   - "qe-ipd.RFL.dat" for the photo detectors of the future using intevac intensified avalanche photodiodes of average QE 45% in allwavelengths between 300 and 600 nm. These are the detectors assumed as the default in the MC studies presented in the MAGIC design report.

- "qe-intevac-hpd.RFL.dat" for optimistic future hybrid photodiodes development. These have been routinely used for simulations of the MAGIC clone.

  Bear in mind that the measurements of QE have been performed using a setup which guarantees a perfect photoelectron collection by the first dynode. During real operation, about 10% of the photoelectrons produced in the photocathode fail to reach the first dynode [2]. This effect is taken into account somewhere else (see below).

4. The file **MagicProgs/Simulation/Detector/Data/LightCollection.dat** contains the light collection efficiency of inner and outer pixels as a function of the angle between the photon trajectory (after reflection on the mirror dish) and the camera plane (given in the range from 0 to 180°, although efficiency for an angle Ψ is assumed to be the same as for $180° - \Psi$). This light collection efficiency must include the transmission of the plexiglas window of the camera, the reflectivity of the light guides (Winston cones), and the collection efficiency of photoelectrons by the first dynode of the PMTs. The present default values are based on data observation and some studies done with WC and plexiglass window prototypes. In this file the values are differnt for inner and outer pixels, despite the differences between inner and outer pixels are not yet well known. There exist and ols file (LightCollection.dat.old) with temptative values that are the same for inner and outer pixels.

5. If one wants to simulate the **diffuse** Night Sky Background (NSB), it is necessary to have pre-generated files containing a noise database, produced with the "starrespo" program, which is included in the camera program package (see section 9). The use of a noise database allows speeding up the execution of camera.

6. If one wants to simulate the effect of light from individual stars, apart from the NSB database described above, a special reflector file containing the stars that are in the field of view is needed. It can be generated with the StarFieldAdder program [3].

4 Input card

The input card is a text file which sets all parameters and options of the camera program. It consists of a chain of commands that are read sequentially in the order that they are written in the card. This implies that if one defines something more than once, or uses commands that are not compatible, only the last definition or command will be used by the program, and the others ignored.

4.1 Mandatory commands

No camera input card can miss the following commands:

camera 0.7

It should be the first line of the file. The camera program uses it to check if it is really an input card for the current version of camera.

input_file 0 /hd02/Maggi/RefData/Gamma/gam_15_on.rfl

The command input_file tells the program where the reflector file to be processed can be found. The file has to be a reflector file, otherwise the camera will stop. You can specify the file name by the absolute path (as above) or by the relative path with respect to the actual directory. The number before the file name indicates the telescope identifier (0 to n-1 if n telescopes are simulated). Be aware that in multi-telescope operation the camera program does not check that the all reflector files that are passed to the Telescopes came from the same shower. At the present moment this is responsability of the user.

qe_file 0 ../Data/qe-emi-coat.RFL.dat

Here, one specifies from which file the quantum efficiency data should be read. Of course to achieve a realistic result, the reflector file must contain Cherenkov photons in the full range of the PMTs. By default our reflector files contain photons between 290 and 600 nm. One should not try to use camera on our standard reflector files setting photodetectors with non negligible QE outside that range.

nsb_directory /hd02/Maggi/StarLight/Small/

To speed up the simulation, the response of the electronic chain of MAGIC to the NSB is taken from a pre-built data base (see section 9). This command tells the camera program the location of the data base, and is mandatory only if one has chosen to include night sky background light (either diffuse or from individual stars) in the simulation (nsb_on command, see below).

nsb_dir_outer /hd02/Maggi/StarLight/Large/

It may happen that the outer (and larger) pixels are set to a different gain (see command fadc_outer below). It would not affect only the data, but also the noise, and therefore the NSB database for inner and outer pixels may be different. This command tells the camera program where is the data base for the outer pixels. Again, it is not a mandatory command if no NSB simulation is requested.

data_file MagicMC01.dat

This command gives to the program the name of the output text file where it will store a brief summary of trigger statistics (number of processed events and fraction of triggers). The file name can use a relative (as is shown here) or absolute path.

root_file MagicMC01.root

The root_file commands indicates the name of the output root file. This file will have the final information of the MAGIC simulation and it will be written in the root format that has been defined [4]. The file name can use relative (as shown here) or absolute path.

end_file

The end_file command marks the end of the input card. Anything after this line will be ignored by the camera program.

4.2 Optional commands

Some other commands in the input card set certain parameters of the camera simulation. These commands are optional: if they are omitted in the input card, reasonable default values will be used. The commands are the following:

ct_num 1

The ct_num command tells the program how many telescopes will be analised.

ct_geom 1

The ct_geom indicates the camera geometry for each telescope (number, size and positions of pixels). The available geometries are:

1 MGeomCamMagic : the present geometry of Magic.

2 MGeomCamMagic919 : it is a camera of the same size than the present camera but all pixels have the size of our present inner pixels.

3 MGeomCamMagicHG : same camera size and configuration than the current camera but with half diameter pixels. HG stands for "High Granularity".

4 not defined.

5 MGeomCamECO1000 : camera and pixels with the same angular size than the current one, but scaled for a telescope with focal of 34.5 meters and $\simeq$ 1000 m^2 mirror surface.

6 MGeomCamECO1000HG : like MGeomCamECO1000, but with half angular size pixels (high granularity).

7 not defined.

8 MCamGeomCT1 : CT1 camera.

9 MCamGeomCT1Daniel : CT1 camera (D. Kranich's version).

The number following the command contains one digit per telescope, first digit is geometry for telescope 0, second digit for telescope 1 and so on. For instance, ct_geom 1115, for a 4-telescope configuration means that telescopes 0, 1 and 2 will have the standard MAGIC camera geometry, while telescope 3 will have the camera defined by MGeomCamECO1000.

trigger_prop 3.0 0.25 1.0 2.0

The trigger simulation needs several inputs defining the behaviour of the trigger. They are given through the "trigger_prop" input command.
The first number is the length of the gate that the discriminator of one pixel opens each time the signal of this pixel passes from below to above the threshold.
The next number means the minimum time (in ns) that two pixels have to be "on" (overlapping their discriminator logical outputs) to be considered in coincidence (0.25 is the minimum that the software can handle).
The shape of the single photo-electron response is also needed and it is given with the third and the last number. Currently, the shape is assumed to be a gaussian with amplitude in mV (third number) and FWHM in ns (last number).

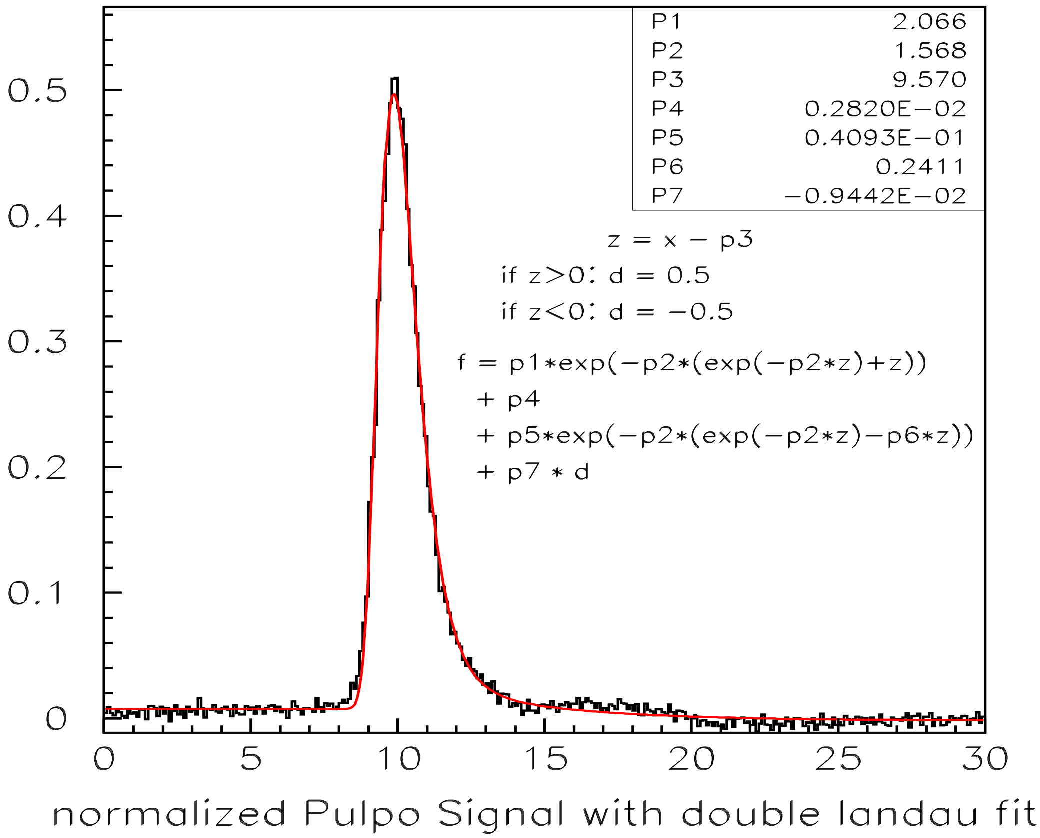

Figure 1: Shape of the FADC response to a single phe as measured usign the Pulpo set-up. In this set-up the pulse is generated by a pulser instead of the PMT but the rest of the chain (transmitter board, optic fibers, reciver boardsa and FADC) are the actual ones of the MAGIC Telescope.

trig_noise 0.3

The sigma of the gaussian electronic fluctuations of the amplitude of the single photoelectron pulse which is used for the trigger simulation. The number indicates the sigma as fraction of the amplitude of the pulse (i.e., 0.3 corresponds to fluctuations of 30% of the amplitude).

fadc_prop 1 4.0 5.0

The FADC simulation needs to know the single photoelectron response of the actual FADC. Currently two different pulse shapes can be chosen:

- 0 in the first value means that a gaussian with a given area in FADC counts (second number) and FWHM in ns (third number) is used.

- 1 in the first value means a parametrization of FADC shape extracted from actual data (figure 1). In that case the second number is again the integral of the response shape in FADC counts, and the third number is ignored since the width of that response is fixed.

fadc_outer 1 4.0 5.0

The different size of the outer pixels (in some geometries like the most important one : the actual Magic geometry) may need to set different gains for inner and outer pixels. This command does the same than the previous one but fot the outer pixels, allowing to set a different gain for them. For instance, to "flat-field" the camera one would set a gain (second number) in outer pixels 4 times lower than for inner ones, so that the signal in the FADC for a fixed light density would be the same in inner and outer pixels (which have an area 4 times larger).

fadc_noise 2.0 2.0

This sets the noise to be added to the signals in the FADC. Both numbers indicate the sigma in ADC counts of a gaussian noise: the first number is related with the noise that is produced before the spliting of the signal in the two gain branches (therefore in the low gain it will be attenuated, like for instance the noise due to the optical transmitter boards); the second number is a noise more intrinsic to the FADC itself, so that it will be observed equally both in high and low gain.

fadc_pedestal 10.0

It allows to set a gloabal mean pedestal for all the FADC. Units are FADC counts. It is better to use an integer number, in some ocassions we have observed that a non-integer mean pedestal may lead to some (not fatal) problems if combined with some other program options.

starfield_file /usr/users/magicsol/RefData/CrabNebula.rfl

This is the name of the file containing the star field information. This file should be a reflector file generated by the starfield adder program [3]. Any other reflector file would also work, but it most likely would give non-sense results. If this command is not in the input card, the program will assume that there is no light from individual stars.

mirror_fraction 4 0.95

Fraction of mirrors that contributes to the light reflected (second number). The first number is Telescope index, therefore it indicates to which telescope this fraction has to be applied. This input command is useful to take into account the lack of mirrors respect to the ideal case used in the Reflector simulation.

sigma_xy_cm_spot 3.0 3.0

Sigma in x (first number) and y (second number), in cm, for additional gaussian spread of the mirror spot. Bear in mind that this applies the same gaussian spread everywhere in the camera plane, that is, we are somehow neglecting the effect of optical aberrations (not completely, because aberrations were taken into account by the reflector program when calculating the impact point of each photon written in the reflector file).

sigma_x_cm_spot 3.0 or **sigma_y_cm_spot 3.0**

It makes the same than the previous input command but only for x or y.

misspoint_deg 0.2 0.1

Additional misspointing to the one generated while running the reflector simulation in degrees. The first value is for the x axis and the second for the y axis on the camera plane. This command is intended to reproduce without having to re-run the Reflector a relativelly small misspointing (to 0.5 degrees). One should not use it for larger values since the Reflector only saves photons in a limited region around the camera center (by default a circle of 70 cm radius), and one may start to reach the limit of this region. Another reason not to use it for large angles is that this command simply shifts the images on the camera plane, without accounting for aberrations.

telescope_location_cm 0 3000. -7000. 0.

Coordinates of telescopes in Corsika system, given in cm. First number is the telescope id, the other three are x, y, z. In this case, telescope 0 is placed at x = +30 m, y = -70 m, z = 0. This option is necessary only for running in stereo mode. Please be very careful to set the same positions that have been used in the previous steps of simulation (MMCS and reflector), since the camera program has no way to check them! In single mode the telescope position is always at the origin of coordinates and it is not necessary to set it in the input card.

trigger_delay

Delay in ns between beginning of FADC time window and the trigger instant. Useful to shift the peaks of pulses in the FADCs to the desired position (usually the same that one sees in the real data).

nsb_mean 0.15 10

To control the diffuse Night Sky Background (NSB) one uses this command.
The first number sets the amplitude of the diffuse NSB. This is given in number of photoelectrons per one inner pixel of the current camera with the 240 m^2 MAGIC mirror per ns. This means that if we use a different geometry (larger mirror or larger pixels) or camera (different QE) the photoelectron rate will be automatically scaled by the program to account for this.
The second value defines the minimum number of photoelectrons from the shower to simulate the NSB for each shower. Since many (most) of the showers in the reflector files

have produced few or no photons in the camera, we may speed up the camera simulation by telling the program to skip all these showers. In the example above, all showers with less than 10 photoelectrons produced by Cherenkov photons will be skipped, since with any reasonable trigger conditions those showers have a negligible probability to trigger the telescope. Of course if one wants to study noise looking at how do “empty” events (no C-photons) look like, this second number has to be set to 0.

seeds 69184 10406

This gives the seeds for the random number generators.

There are also few that gives some input parameters and, simultaneously, switch on or off some options of the camera program:

trigger_single 0 4.0 4 2

This command sets a single trigger condition for the camera simulation. The first number is the telescope id (0 in this case), then comes the discriminator threshold in mV (4.0), the multiplicity (4, the number of pixels required to trigger), and the topology(2). Topology = 2 indicates close-compact neighbour pixels (this means that if any of the 4 pixels is taken out of the group, the remaining ones are still neighbours). Other possible topologies are 0 (1 pixel with n-1 neighbours) and 1 (n neighbours). The trigger_single is the standard way of running camera.

trigger_loop 4.0 6.0 0.5 2 4 0 2

It switches the trigger loop mode on. The three first numbers are the lower limit, the upper limit and the step for the loop over the dicsriminator threshold (all of them in mV). The next two are the limits for the loop over multiplicity, and the last two the limits of the topology loop. This trigger mode is meant to be used to perform trigger studies and, therefore, the camera program does not write the Raw Evt data in the root file. It is faster to run the telescope in the trigger loop mode with a number of conditions than running many times the camera with a single condition. The MultiTelescope option is not supported in the loop mode.
Be careful, too big ranges in the different parameters may crash the job without further explanation, due to a memory overhead.

pixel_thres 37 7.0

This command introduces a particular threshold (7.0) for a given pixel (37).

secure_disc 50.0 7.0

It switches on the option to rise the discriminator threshold for high DC currents (it would kick out trigger due to stars) and gives the parameters for that. The first number is the maximum DC and the second the level at which the discriminator is set.

select_energy 0 10000000000

This command selects the range of energy (GeV). Only the primary showers generated with energy inside this range will be analysed.

And finally several commands that switch on or off different options of the camera program:

nsb_on

This switches the Night Sky Background (NSB) simulation on, both the diffuse light of the NSB and the star light.

nsb_off

This switches the Night Sky Background (NSB) simulation off, both the diffuse light of the NSB and the star light.

sfr_on

This switches the rotation of the Star Field on.

elec_noise_off

When this command is in the input card, the camera simulation program does not introduce any noise comming from hardware considerations (electronics, optic links,...)

write_all_events

If this input parameter is in the input card, the information will be saved in the root file for all the simulated showers. Otherwise it is only saved for the ones that trigger.

nowrite_RawEvt

The raw data like information (MRawEvtData containers with FADC information) coming from the simulation will not be written to the root file, if this input parameter is set. Remind that in trigger_loop mode raw data will never be written in the output file.

nowrite_McEvt

The Monte Carlo information about the shower generation will not be written to the root file, if this input parameter is set.

write_McTrig

It tells the camera program that it has to write some information about the trigger simulation (threshold, multiplicity, and topology...) in the root file.

write_McFadc

It tells the camera program that it has to write some information about the fadc simulation (pedestals, single photoelectron response, ...) in the root file.

trigger_scan

It activates the interactive showing of the trigger signal for each event. It does not work in batch mode.

fadc_scan

It activates the interactive showing of the FADC signal for each event. It does not work in batch mode.

skip

This command works interactively, therefore it is not foreseen for a batch job. It allows skipping some of the generated showers.

5 Output files

The camera program produces two output files, which are given with the input card, by the commands root_file and data_file.

The "data_file" is an ASCII file with a brief summary of the trigger statistics. Here is an example:

163851 event(s), with a total of 13923137 C.photons

Fraction of triggers:

Thres 4, Multi 4, Topo0: 0.00671342% (11 out of 163851)

Thres 4, Multi 4, Topo1: 0.00671342% (11 out of 163851)

Thres 4, Multi 4, Topo2: 0.00427217% (7 out of 163851)

The "root_file" is a root file, which can have several trees depending on with which option the camera program is executed. There are mainly three sets of data inside this file, which are spread over two trees and several branches (see figure 2).

The first set is the raw data. This is the actual output of the simulation, the data that simulates the real data of the MAGIC telescope. There are two branches, which are called MRawEvtData and MRawEvtHeader, in the Events tree. These branches have the same structure as the real (merpped) data (see documentation about Raw Root Format [4, 5]).

The Monte Carlo information about the generation of the shower in the atmosphere can also be found in the file. It contains information like the energy of the primary particles, the direction of the shower... All this is available in the branches called MMcEvt inside the Events tree[4], and in MMcCorsikaRunHeader and MMcRunHeader in the RunHeaders tree.

Finally, there is a set of data with information about the simulation of the Detector. This set is split in several branches: the MMcTrigHeader, MMcFadcHeader, MMcRunHeader, MMcConfigRunHeader, MRunHeader and MGeomCam, which are in the RunHeaders tree, and the MMcTrig, which is in the

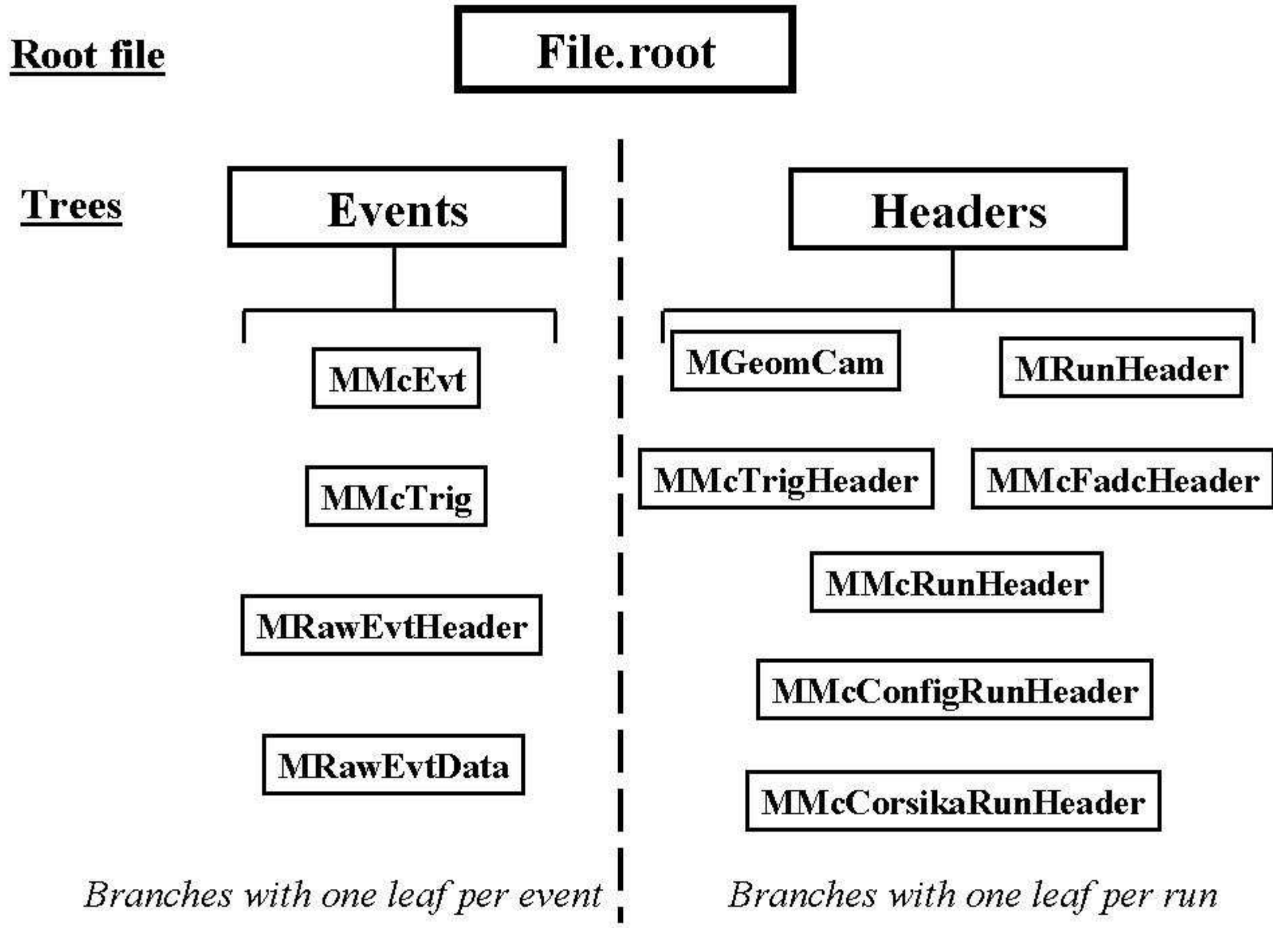

Figure 2: Diagram of root_file structure.

Events tree. The former contain common information for the whole run. The later has information related to the trigger studies for each event. For more precise information of what is stored in each branch check [4].

If more than one telescope is simulated, or the "trigger_loop" option (see section 4)is on, some of the branches will appear more than once. For each simulated telescope there will be one: MRawEvtData, MRawEvtHeader, MMcTrig, MMcTrigHeader, MMcFadcHeader, MGeomCam. And for each trigger condition analised there will be one: MMcTrig, MMcTrigHeader and MMcFadcHeader. In the trigger loop case there is no information about Raw events (MRawEvtData and MRawEvtHeader) since it is thought to be used only for trigger conditions optimization. The different branches of the same type can be differenciated by a number index (MMcTrig;1, MMcTRig;2, ...). Note that the camera program is not ready to simulate simultaneously more than one telecope in the trigger loop mode.

6 An overview of the Camera Simulation

The camera simulation is the last step of the MAGIC MonteCarlo simulation. In it, not only the camera behaviour is simulated but also the behaviour of night sky background light (NSB). And it includes most of the detector or environment characteristics that can change: diffuse NSB, stars in the field of view, trigger rules, electronic noise and so on. This allows to study a lot of data taking conditions without having to run the whole MC simulation chain again. It would be enough to pass the already generated reflector files through the Camera simulation program with the new desired conditions.

6.1 Before Runnig camera

Before running the camera simulation program, one needs to:

1. Run the MMCs, which simulates the showers in the atmosphere, for the desired conditions of incident particles[6]. The current version of MMCS is 6.019, and it is just a (slightly) modified version of the CORSIKA program [7].
2. Run the Reflector program, which traces the light to the camera plane for the desired conditions of the reflector, accounting also for the absorption in the atmosphere [8].
3. (Optional) Run the StarFieldAdder [3], to get the number of photons in the camera plane due to the stars in the field of view.
4. (Optional) Generate a database for the signal in the trigger and FADC due to the Night Sky Background (see section 9). Needed only if simulation of NSB, either diffuse or from individual stars, is required.

All these has only to be redone if one wants to change some of the conditions: characteristics of incident particles, atmospheric absorption, reflector performance [1].

6.2 Night Sky Background

To get realistic results, the signal produced due to the NSB should be considered together with the photons coming from atmospheric showers. Actually there are two different types of NSB that should be treated slightly differently: the diffuse NSB and the star light. Even if all this contributions are put together in the Camera program, they are simulated by different programs. Cherenkov photons from showers are generated with MMCs and Reflector. Both contributions of the NSB come from peripheral programs: the StarFieldAdder and the StarResponse programs.

6.2.1 Starlight simulation

Due to the large mirror area, MAGIC will be sensitive to stars up to a magnitude of 10. These stars will contribute locally to the noise in the camera and therefore cannot be neglected. The so called StarFieldAdder program was developed to simulate the star light that reaches the camera plane. This program considers all stars in the field of view of the camera around a chosen direction using data from a star catalog SKY2000. Then it calculates how many photons of which wavelength (presently 290 nm - 800 nm are simulated) will hit a circular surface of the mirror radius in the integration time and which are the director cosines of each photon. Arrival time, wavelength and ground position are randomized within the integration time, the four wavebands (U, B, V, R) and the mirror area respectively with flat distributions.

[1] Note that the Camera simulation program has been improved to allow the tuning of MOST of the reflector characteristics that may change (point spread function, misspointing, reflector reflectivity)

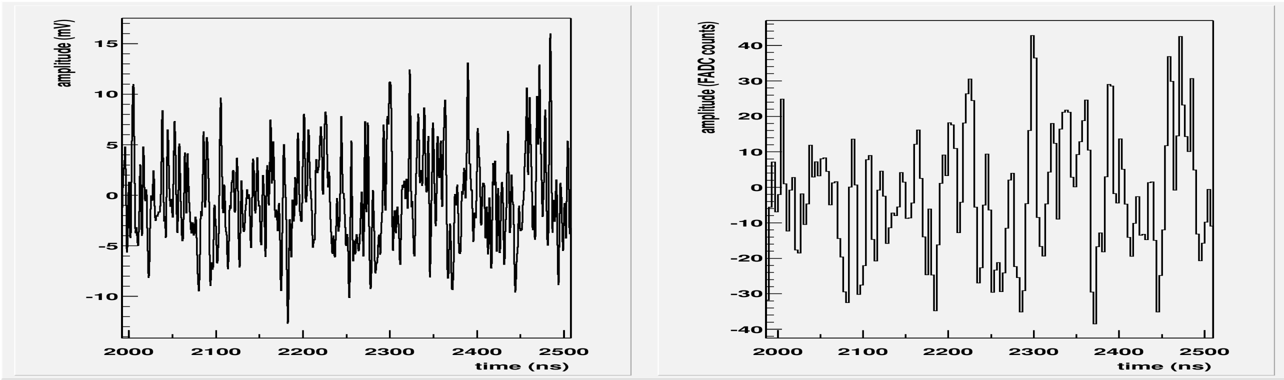

Figure 3: The response of a pixel which has 10.0 photoelectrons per ns from the NSB. On the left plot the analog signal that goes into the trigger system is plotted. On the right plot the content in the FADC system is shown.

StarFielAdder finally writes the generated photons into a binary file of MMCs format which can be read by the Reflector simulation. The output from the Reflector can then be fed to the Camera that will pixelize those photons and process them to get the contribution to the trigger and FADC systems due to the the Starlight.

6.2.2 NSB database

If the Camera program had to generate all the photons coming from the NSB for each event, it would slow down significantly the simulation. Therefore a database is generated and the Camera program just reads the response due to the NSB from there and adds it to the shower light. This program comes toghether with the camera distribution.

This is done with the StarResponse program(see section 9). This program simulates the response for the analogue signal, which will go together with the shower contributions to the trigger discriminator and the FADC system signal, of one channel for a given number of photo electrons per ns (see figure 3). Actually it generates these signals for a much longer time (10 microsec) than that used for each simulated shower, allowing to create a database. The Camera program will read only the needed time interval, choosing the starting point randomly.

The version 0.7 of the Camera simulation program does check that is reading a coherent version of the NSB database (produced with the same parameters for the camera electronics).

6.3 Camera simulation

The Camera program runs over a reflector file (one for each CT that one wants to analyse) and comprises the pixelization, the addition of the NSB, the light guides, the behaviour of the PMTs, and the trigger and FADC system. For each shower, the output is in the same format as the real MAGIC data. This is written to disk together with information about the simulation: shower characteristics, mirror reflectivity, quantum efficiency (QE), trigger condintions and everything that can be useful for data analysis or that can change.

The structure of the program is as follows. First, the input parameters are read. Second, some databases are generated and some values needed for the simulation (quantum eficiency, light guides, NSB) are stored in memory. Finally the shower analysis is done.

The initial step of the camera simulation is reading all the parameters from the input card. It configures both the format of the output file and the execution parameters of the program, as is described in 4.

Then several needed files, which are specified through the input card, are read and their contents are stored in memmory. The read files are:

1. Star Field File: It does content the position of photons in the camera plane due to the stars in the field of view. The Camera program reads the star field file and uses its information to calculate the average NSB photoelectron (PhE) rate from the star field for each pixel considering the atmospheric extinction and the QE of the photo detectors. Later on, the number of PhEs per ns from the diffuse NSB, which is an input parameter of the program, is added to get the final NSB PhEs rate for each pixel. With this information the NSB database will be read filling the individual pixel responses with the signal that corresponds to their rates.

2. Quantum Eficiency (QE): A file with the specific quantum eficiency for each pixel as function of the wavelength is read and QE(λ) is stored in memory. The QE for each PMT has been generated taking into acount the measurement of some PMT and the dispersion on them [2] (figure 4). Actually the QE values stored in the file assume a perfect collection efficiency of photoelectrons from the photocathode onto the first dynode. In reality, there is about a 10% of the produced photoelectrons that do not manage to arrive at the first dynode. This effect is taken into account through the numbers stores in the LightCollection.dat file (see next point).

3. Light Collection: the effect of the light guides (Winston cones), plexiglas window in front of the camera and photoelectron collection on the first PMT dynode (not included in the QE file) will also be applied and it is read form the file : "../Data/LightCollection.dat". There the the fraction of photons (in arbitrary units) that reach the cathode are quoted as function of the angle [9]

The following step consists in generating a database for the electronic noise, both for the trigger and FADC channels.

Before starting to read the showers simulated with MMCs and the Reflector, the Camera simulation program prepares the output file and fills most of the header information with the run header read from the reflector file or with camera running conditions. In this phase a pedestal run is also simulated in order to get the pedestal under the conditions that will be aplied.

Then the loop over all showers starts. What is done for each shower depends on input parameters. In any case the Camera starts to read photons from the shower. For each photon, the first step is to find out which pixel it hits on the camera plane. For that , the information of the photon position is read from the reflector and both the misspointing (a fix shift to some direction) and the point spread function (a sigma of gaussian distribution for x and y axis) corrections are applied. After that, the wavelength dependent QE for each PMT is taken into account to decide if either a PhE is produced or not.

For each PhE leaving the photo cathode we use a standard response, to generate the analog signal of that PMT, separately for the trigger and the FADC system. At present, it is a parametrization of the current FADC response, but a gaussian for the trigger chain, despite it can be modified if the actual response turns out to be different. The amplitude of the response function is chosen randomly according to the distribution shown in figure 5.

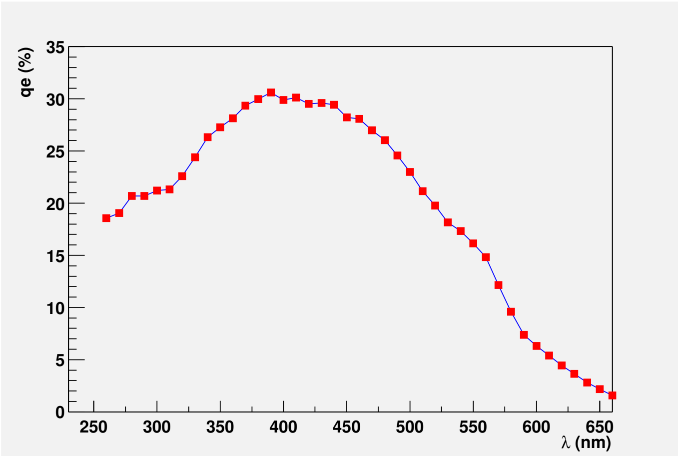

Figure 4: quantum efficiency of a typical MAGIC PMT

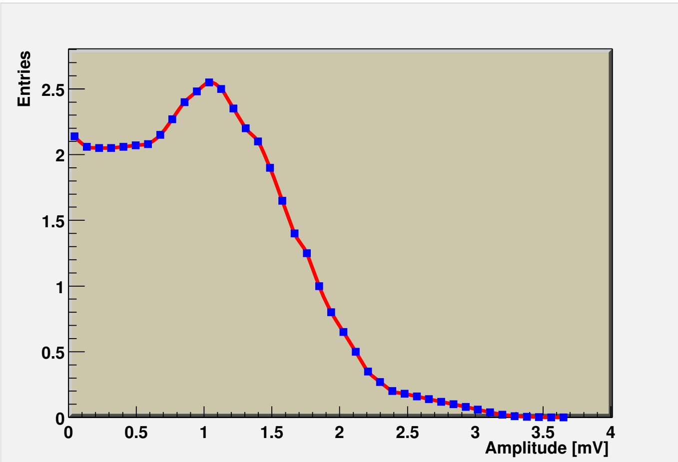

Figure 5: The distribution of the amplitude of the standard response function.

By superimopsing the signal coming from all photons of one pixel, taking the arrival times into account, the response of the trigger and FADC system for that pixel is computed. Once all photons are read and its convertion to PhE is done, the Camera will (if the corresponding conditons are on) rotate the Star Field and superimpose the NSB. If the option "sfr_on" is in the input card, the NSB in each pixel is computed taking into account the rotation of the StarField. Since the pixelization of the photons coming from the stars in the FoV is not really fast and doing it for each event would considerably slow down the program, the rotation is done in a pixel base. The rotation angle is computed from the zenith and azimuth angle of the Telescope pointing and the new value for each pixel is a weighted mean of the pixels that previuosly were in its place. This is done for all pixels in the camera and one gets the analog signal. On top of that the electronic noise is added. At this point a baseline subtraction is done in order to simulate the AC coupling that exists between the PMT output and the signal used by the discriminator trigger and read by the FADC.

The simulation of the trigger electronics starts by checking whether the generated analog signal exceeds the discriminator trigger threshold. In that case a digital output signal of a given length (it can be set through the input card) for those pixels is generated. By checking next neighbour conditions (NN) at a given time the first level trigger is simulated. The NN condition are defined by:

- Multiplicity: the minimum number (N) of pixels that are required to be on.
- Topology: how are these pixels distributed in the camera.
  - 0: A central pixel plus N-1 neighbours.
  - 1: N neighbour pixels.
  - 2: N close compact neighbours, which means that removing any pixel the N-1 remaining pixels are still neighbours.

If the given NN condition is fulfilled, a first level trigger signal is generated and the content of the FADC system, which is just the digitalization of the above mentioned analog signal taking into account the high and low gain channel, is written to disk both for the low and high gain. Then the Camera program starts the simulation for the next shower.

Once all showers have been read, some summary information that goes to the Headers tree is written, the output files closed and the program finishes.

7 Bug trace

Here we try to keep a follow up of the bugs that have been found and solved from the previous version to this one.

1. Light Guides: The convention for the zero reference of the incident angle of photons in the camera plane was wrong (180 degrees flip) and the conversion from rad to deg was inverted. Luckily both effects almost canceled out and the difference in number of phe's due to this error was only of about 2-3%. The effect was small compared to that of the overestimated light collection efficiency (in the old default file **lightguides.dat**, now called **LightCollection.dat**), which formerly neglected transmission of plexiglas ($\simeq 0.96$) and collection efficiency of the first PMT dynode ($\simeq 0.90$). Both problems have been fixed now but will need further tuning in the near future.

2. Digital noise: formerly the digital noise in the FADCs was introduced after the signal in the slices had already been truncated to the range 0 - 255 counts. This often resulted in very saturated signals to appear as unsaturated (due to a negative contribution of the noise), which confused later the analysis routines in MARS leading to underestimated signals in some pixels. Only happened if digital noise was not set to zero.

8 Most frequent problems

8.1 Compiling and linking

- Library libranlib.a I

  /usr/bin/ld: cannot open -lranlib: No such file or directory
  collect2: ld returned 1 exit status
  make: *** [camera] Error 1

  Try to repeat the step 4 of the installation.

- Library libranlib.a II

  ../lib/libranlib.a: could not read symbols: Archive has
  no index; run ranlib to add one
  collect2: ld returned 1 exit status
  make: *** [camera] Error 1

  The library libranlib.a most likely is not the one needed for your system. Check in the directory MagicProg/Simulation/Detector/lib if the library libranlib.a is linked to the one with the extension corresponding to your operation system.

- Dynamic libraries

  /cern/root/root/bin/rootcint:
  error while loading shared libraries:libCint.so:
  cannot load shared object file: No such file or directory

  The dynamic libraries requested for the camera program are not in the environment path. You should add to LD_LIBRARY_PATH the path where the libraries of root (see http://root.cern.ch). If you are using a MAGIC standard machine the neede libraries are in /cern/root/root/lib/. Therefore you can solve it typing:

  export LD_LIBRARY_PATH=/cern/root/root/lib/

  or

  setenv LD_LIBRARY_PATH=/cern/root/root/lib/

  And then type make all again.

- ROOT version You may also have some problems while linking or compiling due to the root version that you have installed. This version of the camera porgram has been checked to work under root version 3.05.07.

8.2 Running

- File not found

  Check that you put the right path and filename in the input card.

- New operator

  > Fatal in ¡operator new¿: storage exhausted aborting

  It usually happens when the camera program is executed in loop mode. The new operator has a maximum space that can allocate. If one tries to analyses too many trigger configurations and wants to store the information of the Trigger simulation this space can be exceeded. Try to analyse less trigger conditions or to not write the trigger information in the root file (comment the "write_McTrig" command in the input card).

- Unexpected mode

  If the camera program does not work in the mode you expected, make sure that you did not leave some commands after the ones you put that cancel what you are asking to the program. The input card works sequentially and, therefore, the last command before the "end_file" has priority over the others.

- Wrong data base for NSB

  There can be several problems while trying to read the NSB data base. First of all one should check if the given path is right (becarefull, in should end with "/"). If it is, the problem is most likely due to a wrong data base. It can be due to several reasons:

  - The camera program cannot get the number of required photoelectrons from the existing files, then a larger data base should be generated.
  - The trigger and FADC properties used to generate the data base are not the same than the ones that the camera program is using, then a new one with the right trigger and FADC properties has to be produced.
  - The version of the NSB data base is not the one required by the the Camera, then one should generate a new data base with the StarResponse program distributed toghether with the camera version that one is using.

  Note that in this version of the camera simulation program, 2 databases for the NSB are requested, one for the small pixels and another for the large ones. This is because one may need to put a different gain for small and large pixels and it would also mean that the effect of the NSB is different for each of them. Obvioulsy if one decides to leave them with the same gain and shape, there is no need to produce the same database twice.

9 Appendix A: StarResponse program

The StarResponse program generates the data base that the simulation of the camera will use to get the FADC and trigger response due to the NSB. To install the program one should do the following:

1. Change to StarResponse directory:
   cd MagicProgs/Simulation/Detector/StarResponse

2. Check the dependencies with the operating system:

   make depend

3. Compile the program using:

   make all

4. Run the program:

   starrespo starresponse.par

The "starresponse.par" file has several parameters, which are given through input commands, that are needed to run "starrespo". They are the following:

1. Path where the outcoming data base will be stored.

   database_path /hd02/Maggi/StarLight/

2. Values of photoelectrons per ns for which the data base will be generated: lower limit, upper limit and precision step.

   simulated_phe 0.0 10.0 0.01

3. Shape for the single photoelectron trigger response: shape identification (0 means gaussian), amplitude and FWHM.

   trig_properties 0 1.0 2.0

4. Shape for the single photoelectron FADC response: shape identification (0 means gaussian, 1 realistic pulse measured with the Pulpo setup), integral in adc counts and FWHM (the latter is used only for gaussian pulse!).

   fadc_properties 1 6.3 5.0

5. The "starrespo" program writes root files, which are not needed but have interesting histograms for NSB studies, if the following command is in the starresponse.par file:

   write_root

6. Finally, there is the end file command, which marks the end of the input card. Any thing after it will not exist for the starrespo program.

   end_file

Be aware that the camera program asks for precision of 0.01 phe/ns if there is less than 1.0 phe/ns and 0.1 precision for larger contributions of the NSB. Therfore it is recomended to run the StarResponse program twice for each trigger and FADC condition one wants to simulate, for instance:

1. **simulated_phe** 0.0 1.0 0.01

2. **simulated_phe** 1.0 50.0 0.1

10 Appendix B: Needed files

Here are all the files of the Camera project. One can check before trying to compile or whatever, that there is the whole set of files.

```
MagicProgs/Simulation/Detector/Camera/MDiag.cxx
MagicProgs/Simulation/Detector/Camera/MDiag.h
MagicProgs/Simulation/Detector/Camera/MDiagLinkDef.h
MagicProgs/Simulation/Detector/Camera/MDiagdict.cxx
MagicProgs/Simulation/Detector/Camera/MDiagdict.h
MagicProgs/Simulation/Detector/Camera/Makefile
MagicProgs/Simulation/Detector/Camera/Makefile.conf.darwin
MagicProgs/Simulation/Detector/Camera/Makefile.conf.general
MagicProgs/Simulation/Detector/Camera/Makefile.conf.linux
MagicProgs/Simulation/Detector/Camera/Makefile.conf.linux-gnu
MagicProgs/Simulation/Detector/Camera/Makefile.conf.osf1
MagicProgs/Simulation/Detector/Camera/Makefile.conf.osf5.1
MagicProgs/Simulation/Detector/Camera/Makefile.rules
MagicProgs/Simulation/Detector/Camera/README
MagicProgs/Simulation/Detector/Camera/camera_install
MagicProgs/Simulation/Detector/Camera/camera-v.h
MagicProgs/Simulation/Detector/Camera/camera.cxx
MagicProgs/Simulation/Detector/Camera/camera.h
MagicProgs/Simulation/Detector/Camera/config.mk.Linux
MagicProgs/Simulation/Detector/Camera/config.mk.linux
MagicProgs/Simulation/Detector/Camera/config.mk.linux-gnu
MagicProgs/Simulation/Detector/Camera/config.mk.osf1
MagicProgs/Simulation/Detector/Camera/creadparam.cxx
MagicProgs/Simulation/Detector/Camera/creadparam.h
MagicProgs/Simulation/Detector/Camera/input.card
MagicProgs/Simulation/Detector/Camera/LinkDef.h
MagicProgs/Simulation/Detector/Camera/moments.cxx
MagicProgs/Simulation/Detector/Camera/moments.h
MagicProgs/Simulation/Detector/StarResponse/MStarLight.cxx
MagicProgs/Simulation/Detector/StarResponse/MStarLight.hxx
MagicProgs/Simulation/Detector/StarResponse/Makefile
MagicProgs/Simulation/Detector/StarResponse/config.mk.Linux
MagicProgs/Simulation/Detector/StarResponse/config.mk.linux
MagicProgs/Simulation/Detector/StarResponse/config.mk.linux-gnu
MagicProgs/Simulation/Detector/StarResponse/config.mk.osf1
MagicProgs/Simulation/Detector/StarResponse/srreadparam.cxx
MagicProgs/Simulation/Detector/StarResponse/srreadparam.h
MagicProgs/Simulation/Detector/StarResponse/starresponse.cxx
MagicProgs/Simulation/Detector/StarResponse/starresponse.hxx
MagicProgs/Simulation/Detector/StarResponse/starresponse.par
MagicProgs/Simulation/Detector/include-GENERAL/
MagicProgs/Simulation/Detector/include-GENERAL/CVS/
MagicProgs/Simulation/Detector/include-GENERAL/CVS/Root
```

```
MagicProgs/Simulation/Detector/include-GENERAL/CVS/Repository
MagicProgs/Simulation/Detector/include-GENERAL/CVS/Entries
MagicProgs/Simulation/Detector/include-GENERAL/DllImport.h
MagicProgs/Simulation/Detector/include-GENERAL/RConfig.h
MagicProgs/Simulation/Detector/include-GENERAL/Rtypes.h
MagicProgs/Simulation/Detector/include-GENERAL/jcdebug.h
MagicProgs/Simulation/Detector/include-GENERAL/jcmacros.h
MagicProgs/Simulation/Detector/include-GENERAL/lagrange.h
MagicProgs/Simulation/Detector/include-GENERAL/ranlib.h
MagicProgs/Simulation/Detector/include-GENERAL/DllImport.h
MagicProgs/Simulation/Detector/include-GENERAL/RConfig.h
MagicProgs/Simulation/Detector/include-GENERAL/Rtypes.h
MagicProgs/Simulation/Detector/include-GENERAL/jcdebug.h
MagicProgs/Simulation/Detector/include-GENERAL/jcmacros.h
MagicProgs/Simulation/Detector/include-GENERAL/lagrange.h
MagicProgs/Simulation/Detector/include-GENERAL/ranlib.h
MagicProgs/Simulation/Detector/include-CORSIKA/COREvent.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/COREvent.hxx
MagicProgs/Simulation/Detector/include-CORSIKA/COREventEnd.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/COREventEnd.hxx
MagicProgs/Simulation/Detector/include-CORSIKA/COREventHeader.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/COREventHeader.hxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORParticle.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORParticle.hxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORRunEnd.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORRunEnd.hxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORRunHeader.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORRunHeader.hxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORStatfile.cxx
MagicProgs/Simulation/Detector/include-CORSIKA/CORStatfile.hxx
MagicProgs/Simulation/Detector/include-MC/MCCphoton.cxx
MagicProgs/Simulation/Detector/include-MC/MCCphoton.hxx
MagicProgs/Simulation/Detector/include-MC/MCEventHeader.cxx
MagicProgs/Simulation/Detector/include-MC/MCEventHeader.hxx
MagicProgs/Simulation/Detector/include-MC/MCEventHeader_2.cxx
MagicProgs/Simulation/Detector/include-MC/MCEventHeader_2.hxx
MagicProgs/Simulation/Detector/include-MC/MCRunHeader.cxx
MagicProgs/Simulation/Detector/include-MC/MCRunHeader.hxx
MagicProgs/Simulation/Detector/include-MTrigger/MGTriggerSignal.cxx
MagicProgs/Simulation/Detector/include-MTrigger/MGTriggerSignal.hxx
MagicProgs/Simulation/Detector/include-MTrigger/MTrigger.cxx
MagicProgs/Simulation/Detector/include-MTrigger/MTrigger.hxx
MagicProgs/Simulation/Detector/include-MTrigger/TABLE_NEXT_NEIGHBOUR
MagicProgs/Simulation/Detector/include-MTrigger/TABLE_PIXELS_IN_CELLS
MagicProgs/Simulation/Detector/include-MFadc/MFadc.cxx
MagicProgs/Simulation/Detector/include-MFadc/MFadc.hxx
MagicProgs/Simulation/Detector/include-MFadc/MGFadcSignal.cxx
```

```
MagicProgs/Simulation/Detector/include-MFadc/MGFadcSignal.hxx
MagicProgs/Simulation/Detector/include-MLons/MLons.cxx
MagicProgs/Simulation/Detector/include-MLons/MLons.hxx
MagicProgs/Simulation/Detector/lib/README
MagicProgs/Simulation/Detector/lib/libranlib.a.linux
MagicProgs/Simulation/Detector/lib/libranlib.a.osf1
MagicProgs/Simulation/Detector/Data/LightCollection.dat
MagicProgs/Simulation/Detector/Data/qe-emi-coat.RFL.dat
MagicProgs/Simulation/Detector/Data/qe-emi-coat.dat
MagicProgs/Simulation/Detector/Data/qe-emi.RFL.dat
MagicProgs/Simulation/Detector/Data/qe-emi.dat
MagicProgs/Simulation/Detector/Data/qe-ipd.RFL.dat
MagicProgs/Simulation/Detector/Data/qe-ipd.dat
MagicProgs/Simulation/Detector/Data/qe-intevac_hpd.RFL.dat
MagicProgs/Simulation/Detector/Data/qe-intevac_hpd.dat
MagicProgs/include-Classes/MBase/MEvtLoop.cc
MagicProgs/include-Classes/MBase/BaseIncl.h
MagicProgs/include-Classes/MBase/BaseLinkDef.h
MagicProgs/include-Classes/MBase/MAGIC.h
MagicProgs/include-Classes/MBase/MArgs.cc
MagicProgs/include-Classes/MBase/MArgs.h
MagicProgs/include-Classes/MBase/MArray.cc
MagicProgs/include-Classes/MBase/MArray.h
MagicProgs/include-Classes/MBase/MArrayB.cc
MagicProgs/include-Classes/MBase/MArrayB.h
MagicProgs/include-Classes/MBase/MArrayS.cc
MagicProgs/include-Classes/MBase/MArrayS.h
MagicProgs/include-Classes/MBase/MAstro.cc
MagicProgs/include-Classes/MBase/MAstro.h
MagicProgs/include-Classes/MBase/MClone.cc
MagicProgs/include-Classes/MBase/MClone.h
MagicProgs/include-Classes/MBase/MContinue.cc
MagicProgs/include-Classes/MBase/MContinue.h
MagicProgs/include-Classes/MBase/MDirIter.cc
MagicProgs/include-Classes/MBase/MDirIter.h
MagicProgs/include-Classes/MBase/MEvtLoop.h
MagicProgs/include-Classes/MBase/MFilter.cc
MagicProgs/include-Classes/MBase/MFilter.h
MagicProgs/include-Classes/MBase/MGGroupFrame.cc
MagicProgs/include-Classes/MBase/MGGroupFrame.h
MagicProgs/include-Classes/MBase/MGList.cc
MagicProgs/include-Classes/MBase/MGList.h
MagicProgs/include-Classes/MBase/MGMenu.cc
MagicProgs/include-Classes/MBase/MGMenu.h
MagicProgs/include-Classes/MBase/MGTask.cc
MagicProgs/include-Classes/MBase/MGTask.h
MagicProgs/include-Classes/MBase/MInputStreamID.cc
```

```
MagicProgs/include-Classes/MBase/MInputStreamID.h
MagicProgs/include-Classes/MBase/MIter.cc
MagicProgs/include-Classes/MBase/MIter.h
MagicProgs/include-Classes/MBase/MLog.cc
MagicProgs/include-Classes/MBase/MLog.h
MagicProgs/include-Classes/MBase/MLogHtml.cc
MagicProgs/include-Classes/MBase/MLogHtml.h
MagicProgs/include-Classes/MBase/MLogManip.cc
MagicProgs/include-Classes/MBase/MLogManip.h
MagicProgs/include-Classes/MBase/MLogPlugin.cc
MagicProgs/include-Classes/MBase/MLogPlugin.h
MagicProgs/include-Classes/MBase/MLogo.cc
MagicProgs/include-Classes/MBase/MLogo.h
MagicProgs/include-Classes/MBase/MParContainer.cc
MagicProgs/include-Classes/MBase/MParContainer.h
MagicProgs/include-Classes/MBase/MParList.cc
MagicProgs/include-Classes/MBase/MParList.h
MagicProgs/include-Classes/MBase/MPrint.cc
MagicProgs/include-Classes/MBase/MPrint.h
MagicProgs/include-Classes/MBase/MProgressBar.cc
MagicProgs/include-Classes/MBase/MProgressBar.h
MagicProgs/include-Classes/MBase/MReadSocket.cc
MagicProgs/include-Classes/MBase/MReadSocket.h
MagicProgs/include-Classes/MBase/MRunIter.cc
MagicProgs/include-Classes/MBase/MRunIter.h
MagicProgs/include-Classes/MBase/MSearch.cc
MagicProgs/include-Classes/MBase/MSearch.h
MagicProgs/include-Classes/MBase/MStatusDisplay.cc
MagicProgs/include-Classes/MBase/MStatusDisplay.h
MagicProgs/include-Classes/MBase/MTask.cc
MagicProgs/include-Classes/MBase/MTask.h
MagicProgs/include-Classes/MBase/MTaskInteractive.cc
MagicProgs/include-Classes/MBase/MTaskInteractive.h
MagicProgs/include-Classes/MBase/MTaskList.cc
MagicProgs/include-Classes/MBase/MTaskList.h
MagicProgs/include-Classes/MBase/MTime.cc
MagicProgs/include-Classes/MBase/MTime.h
MagicProgs/include-Classes/MBase/Makefile
MagicProgs/include-Classes/MRawFormat/MRawCrateArray.cc
MagicProgs/include-Classes/MRawFormat/MRawCrateArray.h
MagicProgs/include-Classes/MRawFormat/MRawCrateData.cc
MagicProgs/include-Classes/MRawFormat/MRawCrateData.h
MagicProgs/include-Classes/MRawFormat/MRawEvtData.cc
MagicProgs/include-Classes/MRawFormat/MRawEvtData.h
MagicProgs/include-Classes/MRawFormat/MRawEvtHeader.cc
MagicProgs/include-Classes/MRawFormat/MRawEvtHeader.h
MagicProgs/include-Classes/MRawFormat/MRawEvtPixelIter.cc
```

```
MagicProgs/include-Classes/MRawFormat/MRawEvtPixelIter.h
MagicProgs/include-Classes/MRawFormat/MRawFileRead.cc
MagicProgs/include-Classes/MRawFormat/MRawFileRead.h
MagicProgs/include-Classes/MRawFormat/MRawFileWrite.cc
MagicProgs/include-Classes/MRawFormat/MRawFileWrite.h
MagicProgs/include-Classes/MRawFormat/MRawRead.cc
MagicProgs/include-Classes/MRawFormat/MRawRead.h
MagicProgs/include-Classes/MRawFormat/MRawRunHeader.cc
MagicProgs/include-Classes/MRawFormat/MRawRunHeader.h
MagicProgs/include-Classes/MRawFormat/MRawSocketRead.cc
MagicProgs/include-Classes/MRawFormat/MRawSocketRead.h
MagicProgs/include-Classes/MRawFormat/Makefile
MagicProgs/include-Classes/MRawFormat/RawIncl.h
MagicProgs/include-Classes/MRawFormat/RawLinkDef.h
MagicProgs/include-Classes/mgeom/MCamEvent.cc
MagicProgs/include-Classes/mgeom/GeomIncl.h
MagicProgs/include-Classes/mgeom/GeomLinkDef.h
MagicProgs/include-Classes/mgeom/MGeomCam.cc
MagicProgs/include-Classes/mgeom/MGeomCam.h
MagicProgs/include-Classes/mgeom/MGeomCamCT1.cc
MagicProgs/include-Classes/mgeom/MGeomCamCT1.h
MagicProgs/include-Classes/mgeom/MGeomCamCT1Daniel.cc
MagicProgs/include-Classes/mgeom/MGeomCamCT1Daniel.h
MagicProgs/include-Classes/mgeom/MGeomCamECO1000.cc
MagicProgs/include-Classes/mgeom/MGeomCamECO1000.h
MagicProgs/include-Classes/mgeom/MGeomCamECO1000HG.cc
MagicProgs/include-Classes/mgeom/MGeomCamECO1000HG.h
MagicProgs/include-Classes/mgeom/MGeomCamMagic.cc
MagicProgs/include-Classes/mgeom/MGeomCamMagic.h
MagicProgs/include-Classes/mgeom/MGeomCamMagic919.cc
MagicProgs/include-Classes/mgeom/MGeomCamMagic919.h
MagicProgs/include-Classes/mgeom/MGeomCamMagicHG.cc
MagicProgs/include-Classes/mgeom/MGeomCamMagicHG.h
MagicProgs/include-Classes/mgeom/MGeomCorsikaCT.cc
MagicProgs/include-Classes/mgeom/MGeomCorsikaCT.h
MagicProgs/include-Classes/mgeom/MGeomMirror.cc
MagicProgs/include-Classes/mgeom/MGeomMirror.h
MagicProgs/include-Classes/mgeom/MGeomPMT.cc
MagicProgs/include-Classes/mgeom/MGeomPMT.h
MagicProgs/include-Classes/mgeom/MGeomPix.cc
MagicProgs/include-Classes/mgeom/MGeomPix.h
MagicProgs/include-Classes/mgeom/Makefile
MagicProgs/include-Classes/mgeom/MCamEvent.h
MagicProgs/include-Classes/mfileio/FileIOIncl.h
MagicProgs/include-Classes/mfileio/FileIOLinkDef.h
MagicProgs/include-Classes/mfileio/MCT1ReadAscii.cc
MagicProgs/include-Classes/mfileio/MCT1ReadAscii.h
```

```
MagicProgs/include-Classes/mfileio/MCT1ReadPreProc.cc
MagicProgs/include-Classes/mfileio/MCT1ReadPreProc.h
MagicProgs/include-Classes/mfileio/MChain.cc
MagicProgs/include-Classes/mfileio/MChain.h
MagicProgs/include-Classes/mfileio/MRead.cc
MagicProgs/include-Classes/mfileio/MRead.h
MagicProgs/include-Classes/mfileio/MReadMarsFile.cc
MagicProgs/include-Classes/mfileio/MReadMarsFile.h
MagicProgs/include-Classes/mfileio/MReadReports.cc
MagicProgs/include-Classes/mfileio/MReadReports.h
MagicProgs/include-Classes/mfileio/MReadRflFile.cc
MagicProgs/include-Classes/mfileio/MReadRflFile.h
MagicProgs/include-Classes/mfileio/MReadTree.cc
MagicProgs/include-Classes/mfileio/MReadTree.h
MagicProgs/include-Classes/mfileio/MWriteAsciiFile.cc
MagicProgs/include-Classes/mfileio/MWriteAsciiFile.h
MagicProgs/include-Classes/mfileio/MWriteFile.cc
MagicProgs/include-Classes/mfileio/MWriteFile.h
MagicProgs/include-Classes/mfileio/MWriteRootFile.cc
MagicProgs/include-Classes/mfileio/MWriteRootFile.h
MagicProgs/include-Classes/mfileio/Makefile
MagicProgs/include-Classes/mfileio/defines.h
MagicProgs/include-Classes/mfileio/structures.h
MagicProgs/include-Classes/mfileio/structures_rfl.h
MagicProgs/include-Classes/MMcFormat/MFadcDefine.h
MagicProgs/include-Classes/MMcFormat/MMcConfigRunHeader.cc
MagicProgs/include-Classes/MMcFormat/MMcConfigRunHeader.h
MagicProgs/include-Classes/MMcFormat/MMcCorsikaRunHeader.cc
MagicProgs/include-Classes/MMcFormat/MMcCorsikaRunHeader.h
MagicProgs/include-Classes/MMcFormat/MMcEvt.cxx
MagicProgs/include-Classes/MMcFormat/MMcEvt.hxx
MagicProgs/include-Classes/MMcFormat/MMcFadcHeader.cxx
MagicProgs/include-Classes/MMcFormat/MMcFadcHeader.hxx
MagicProgs/include-Classes/MMcFormat/MMcRunHeader.cxx
MagicProgs/include-Classes/MMcFormat/MMcRunHeader.hxx
MagicProgs/include-Classes/MMcFormat/MMcTrig.cxx
MagicProgs/include-Classes/MMcFormat/MMcTrig.hxx
MagicProgs/include-Classes/MMcFormat/MMcTrigHeader.cxx
MagicProgs/include-Classes/MMcFormat/MMcTrigHeader.hxx
MagicProgs/include-Classes/MMcFormat/MTriggerDefine.h
MagicProgs/include-Classes/MMcFormat/Makefile
MagicProgs/include-Classes/MMcFormat/McIncl.h
MagicProgs/include-Classes/MMcFormat/McLinkDef.h
MagicProgs/include-Classes/MMcFormat/Mdefine.h
MagicProgs/include-Classes/MMcFormat/readme.txt
MagicProgs/include-Classes/Makefile.conf.darwin
MagicProgs/include-Classes/Makefile.conf.general
```

```
MagicProgs/include-Classes/Makefile.conf.linux
MagicProgs/include-Classes/Makefile.conf.linux-gnu
MagicProgs/include-Classes/Makefile.conf.osf1
MagicProgs/include-Classes/Makefile.conf.osf5.1
MagicProgs/include-Classes/Makefile.rules
```

References

[1] D. Paneque et al., NIM A Volume 504, Issues 1-3, Pages 109-115 (2003)

[2] Private communication from D. Paneque.

[3] http://hegra1.mppmu.mpg.de/MAGIC/private/montecarlo/userguide.html

[4] http://magic.astro.uni-wuerzburg.de:81/mars/htmldoc/ClassIndex.html

[5] O. Blanch, Description of the root data format for the MAGIC telescope, **MAGIC-TDAS 02-01, 020101**

[6] D.Sobczynska, MMCS from CORSIKA 6.014, **MAGIC-TDAS 02-10, 020724**

[7] http://www-ik.fzk.de/ heck/corsika/

[8] A. Moralejo, The Reflector Simulation Program v.0.6, **MAGIC-TDAS 02-11, 021114**

[9] A. Robert, Diploma Thesis.