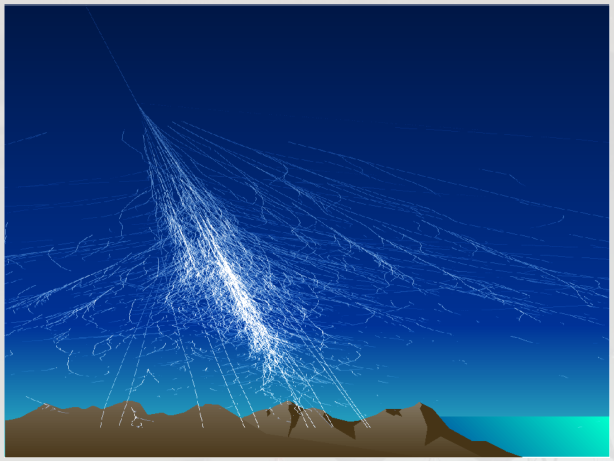
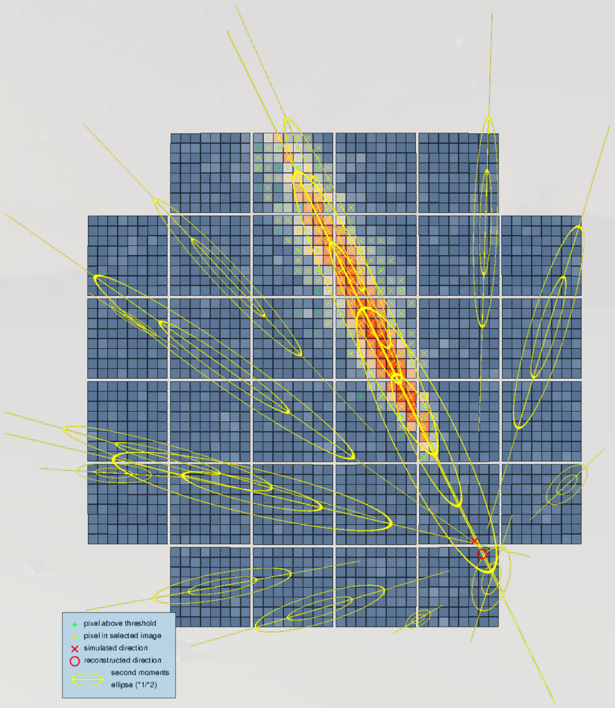
Sim_telarray: introduction, ingredients & configuration

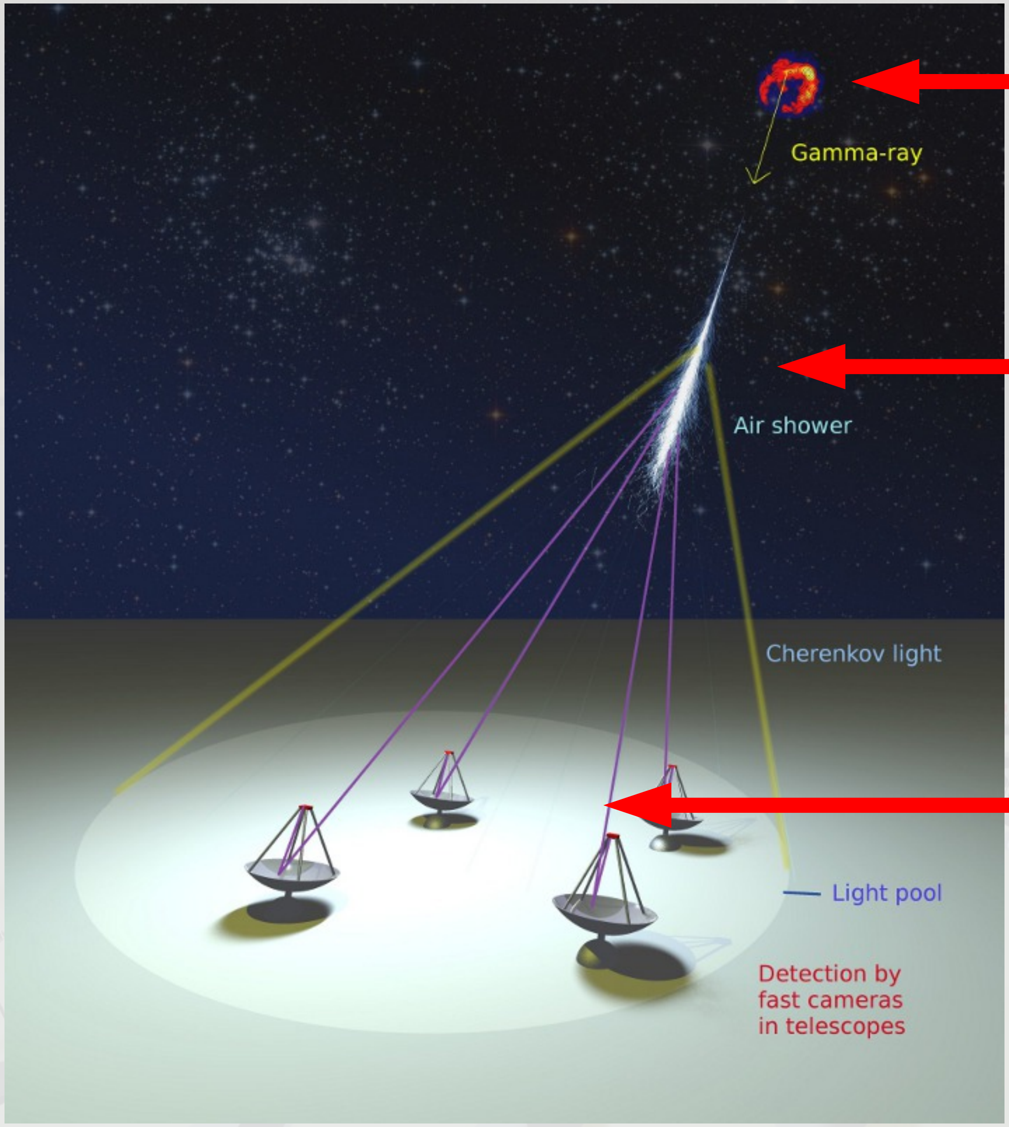


K. Bernlöhr
MPIK Heidelberg



CTA Analysis & Simulation WG Bootcamp
DESY, Zeuthen
2017-06-19

Simulation steps



Emission and propagation to Earth: Astrophysics (not dealt with here).

Particle cascade ("air shower") is normally simulated with CORSIKA, up to light propagation to positions of individual telescopes.

Shower simulation

Cherenkov light atm. transmission, optical properties of telescopes, photon detection, nightsky background, electronic signals, trigger decisions, digitization of signals, ...

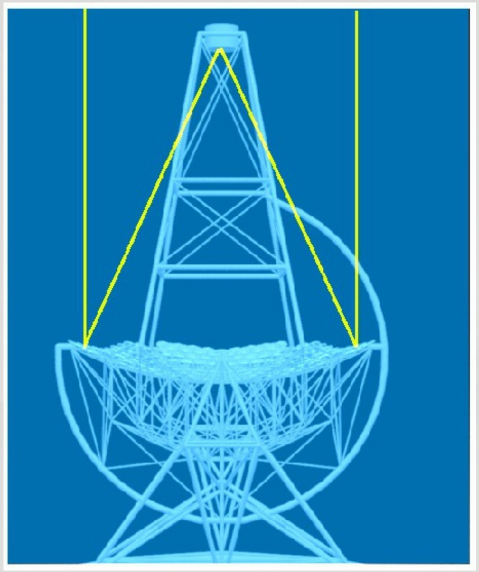
Telescope simulation

Sim_telarray

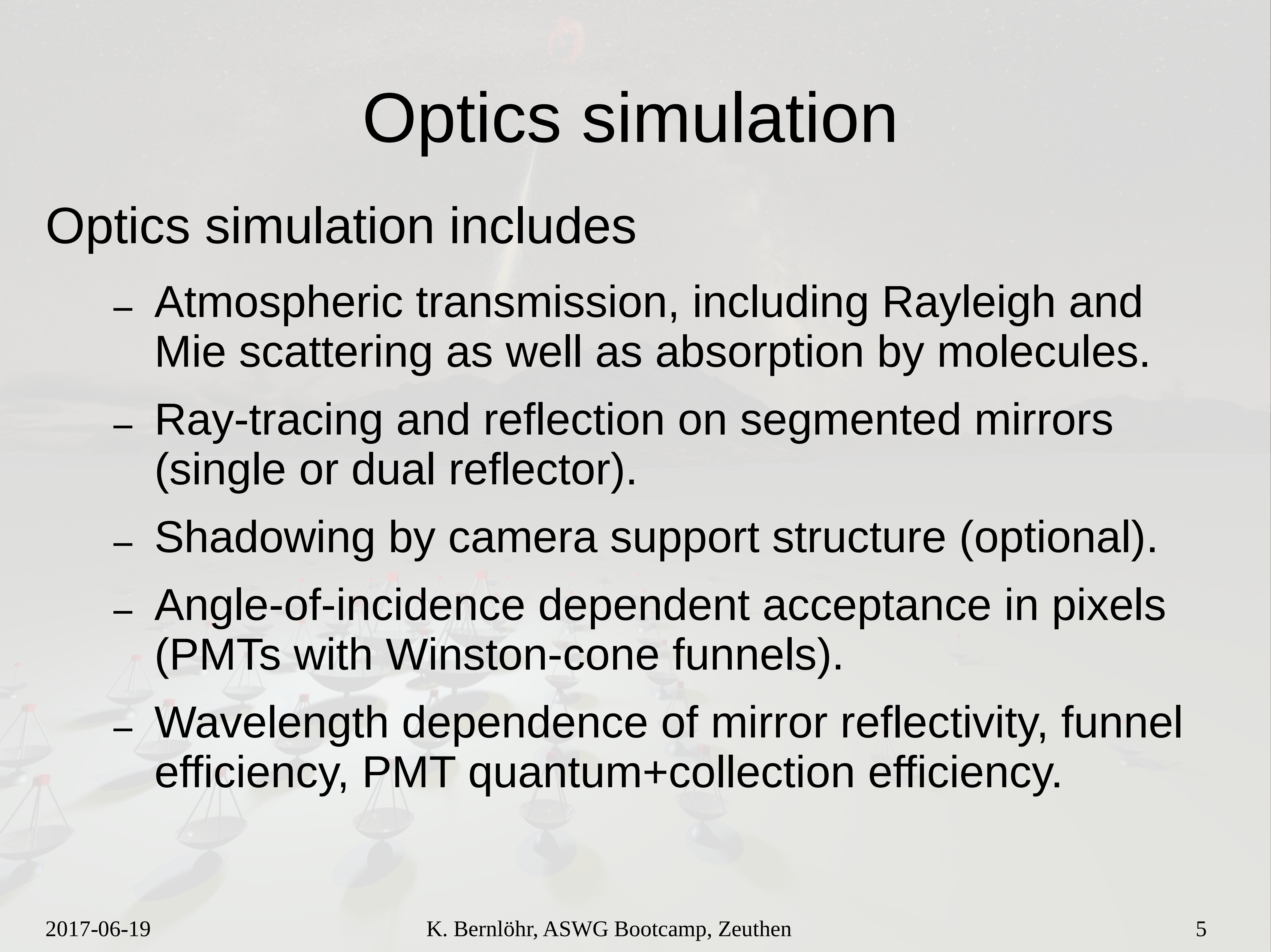
Simulation of arrays of many IACTs.

- Input: Cherenkov light photons (from CORSIKA)
  - MC-true data passed through
- Output: Digital read-out like from real instruments
  - with some simplifications (see later)
- Configuration entirely command line / data-driven
  - same code can simulate H.E.S.S. or CTA or ..., just by reading different configuration files,
  - pre-processor allows for different telescope types,
  - compiler definitions do adjust memory footprint.

The telescope simulation



- Detailed atmospheric transmission tables (“extinction”, optical depth).
- Full optical ray-tracing: segmented non-ideal optics. Can be by-passed.
- Shadowing partially explicit and partially as efficiency factor.
- Photon detection efficiencies, effectively including PMT coll.eff. / SiPM optical cross-talk.
- Nightsky background; including afterpulsing in PMTs.
- Noise, digitization of signals, switching behaviour of discriminators or comparators, …

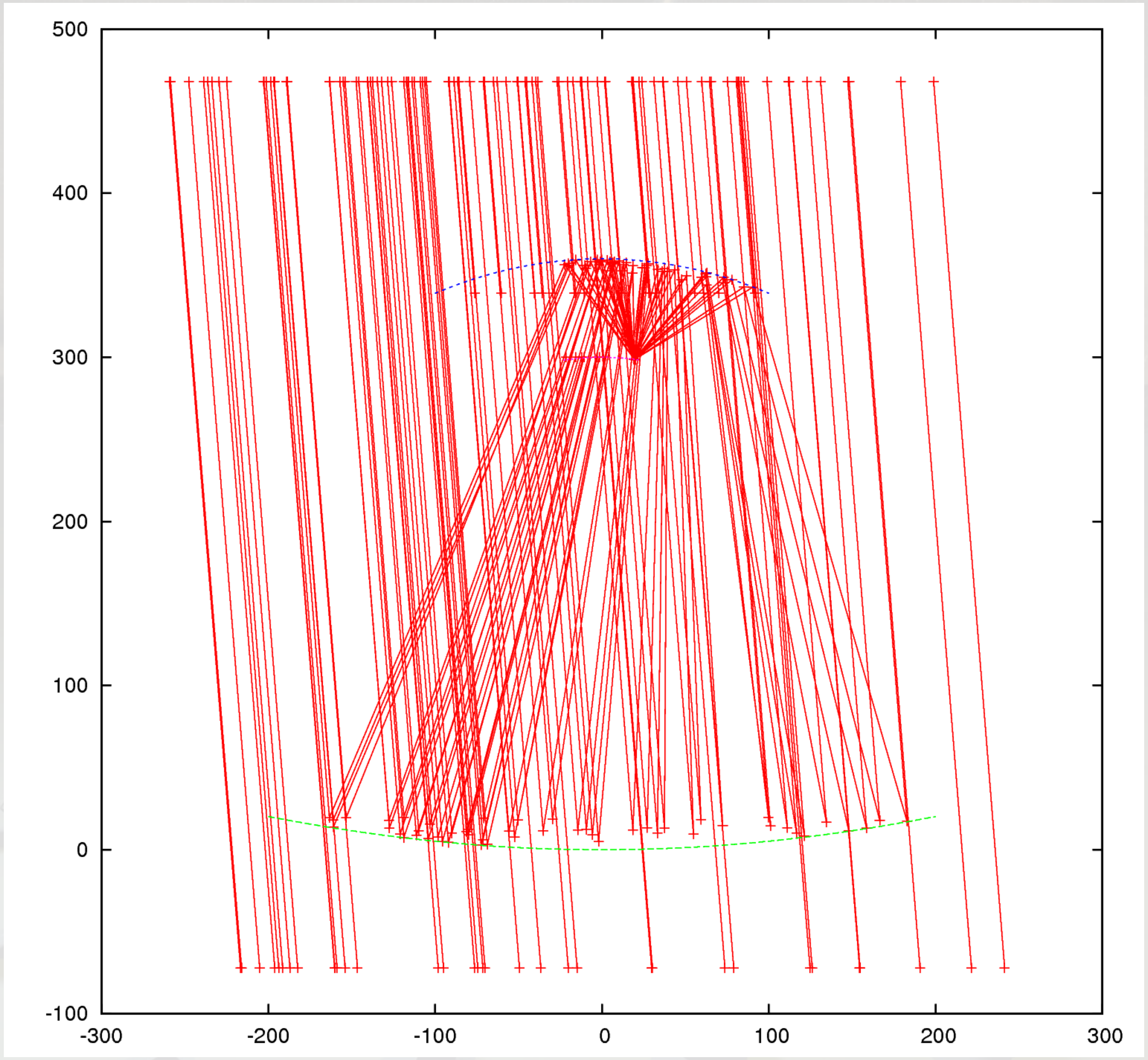


Optics simulation

Optics simulation includes

- Atmospheric transmission, including Rayleigh and Mie scattering as well as absorption by molecules.
- Ray-tracing and reflection on segmented mirrors (single or dual reflector).
- Shadowing by camera support structure (optional).
- Angle-of-incidence dependent acceptance in pixels (PMTs with Winston-cone funnels).
- Wavelength dependence of mirror reflectivity, funnel efficiency, PMT quantum+collection efficiency.

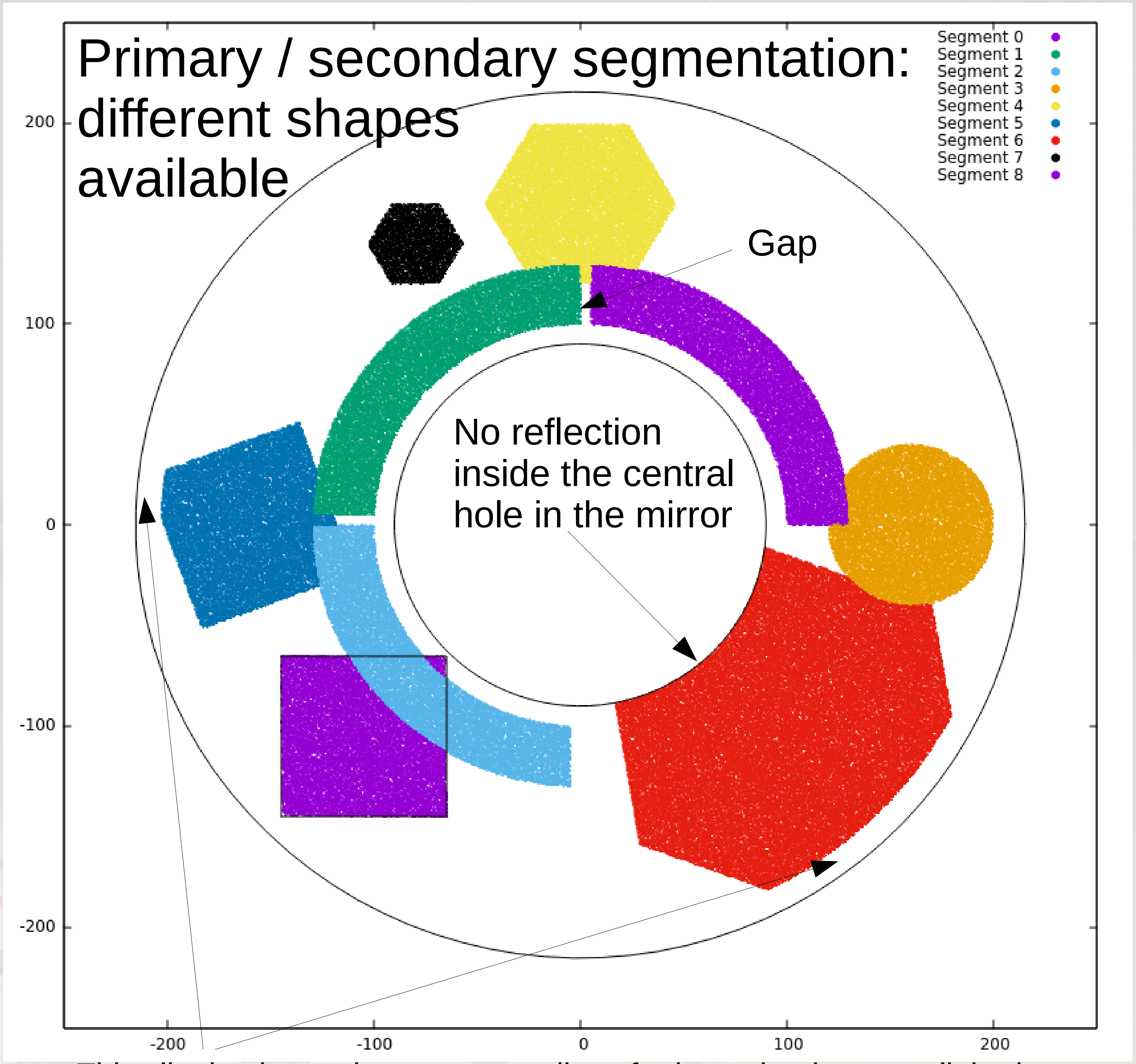
Ray-tracing with dual mirror optics



Reflectors not segmented.

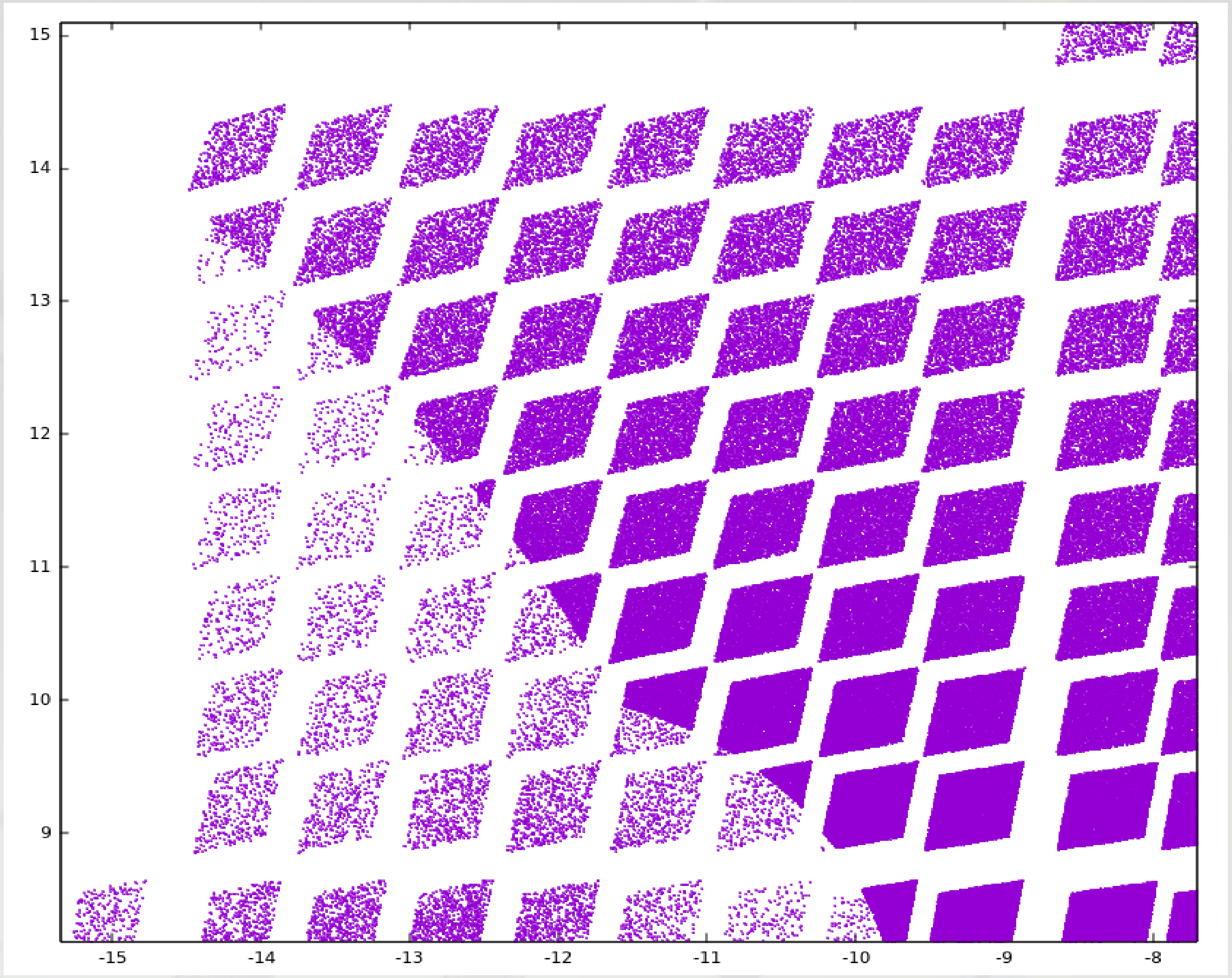
No shadowing of support structures implemented.

Primary / secondary segmentation: different shapes available



This clipping is not due to outer radius of primary but because light does not get reflected on the secondary (only photons on focal surface written).

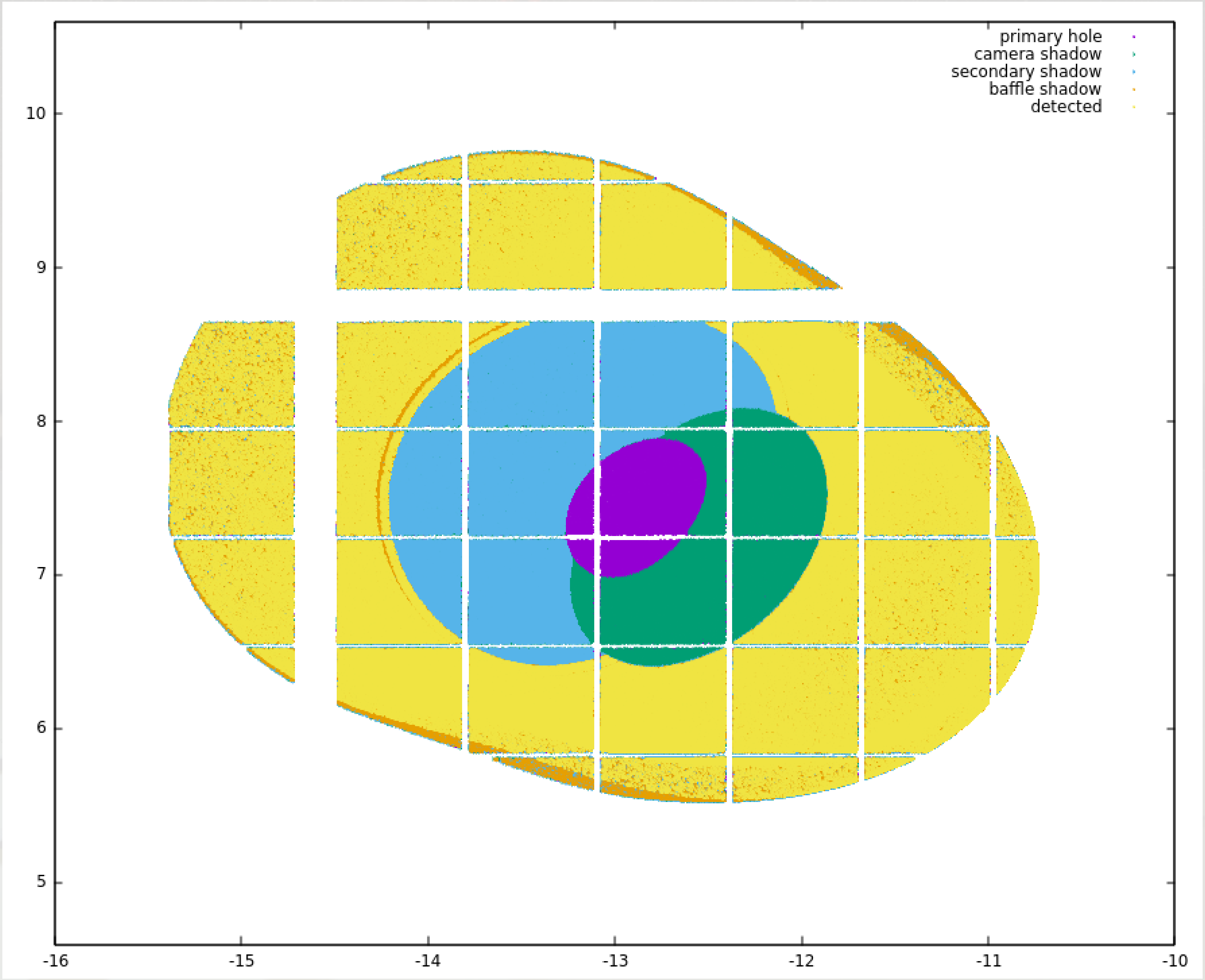
Inclined pixels



(light source out of focus)

Here: camera curvature exaggerated by factor 8. Showing pixel hits in one ASTRI module (with pixel positions not adapted to the strong curvature).

Considering extension to camera configuration file (keyword ‘Pixel’) to allow pixel surface deviating a bit from the shape given by the polynomial.


primary hole
camera shadow
secondary shadow
baffle shadow
detected
10
9
8
7
6
5
-16
-15
-14
-13
-12
-11
-10

Pulse shapes and electronics details

For a realistic simulation a lot of details need to be included, for example:

- PMT amplitude distribution, including afterpulses for nightsky photons
  ⇒ calculate NSB (night sky backgr.) triggers.
- Pulse shapes at input of comparator or discriminator.
- Pulse shapes of comp./discr. outputs to tel. trigger.
- Pulse shapes at analog ring sampler / FADC inputs (multiple gains).
- All of that should be configurable, if you want to be prepared for simulating new types of telescopes.

Sources and documentation

- Public source repository:
  - Sim_telarray:
    https://www.mpi-hd.mpg.de/hfm/~bernlohr/sim_telarray/
    only including minimal HESS-1 configuration files.
  - IACT/ATMO interface to CORSIKA:
    https://www.mpi-hd.mpg.de/hfm/~bernlohr/iact-atmo/
    also including (non-shower) light emission package.
- HESS-specific repository ...
- CTA-specific repository (with all config files):
  https://www.mpi-hd.mpg.de/hfm/CTA/MC/
  see there: Software/ (production), Software/Testing/, Doc/
  see also git web interface.

Older tutorial material

- At the CTA-MC group meeting in Heidelberg in Feb. 2013 a tutorial session was given. Even though configurations there still correspond to prod-2, it may be worth to have a look at the material, see
  - https://cta.cta-observatory.org/indico/conferenceDisplay.py?confId=355 (Tutorial 1 mainly), and
  - https://www.mpi-hd.mpg.de/hfm/CTA/MC/Software/Doc/Tutorial/ (in addition to PDF and screen recording also includes sources of tools and scripts executed.

  with examples focusing mainly on ray-tracing.

Historical background

- First implementation: 1996, for HEGRA
  - note: output directly in EventIO-based HEGRA raw data format.
- Rewrite for more flexible configuration in 1998/9 with output format intended originally for H.E.S.S. (= ‘hessio’) – which is not the H.E.S.S. raw data format.
- Starting in 2006 more and more additions intended for CTA, both to simulation code and extending the data format.

Sim_telarray design focus points

Sim_telarray design intended to focus on

- fast and efficient simulation,
- not every conceivable detector effect needs to be simulated in full detail – but relevant effects should be included in sufficient detail,
- all configuration parameters can be set at run-time (although there are hard-coded default values),
- backward-compatibility: same configuration should give equivalent results (except bugs),
- output stage interchangeable (HEGRA / hessio).

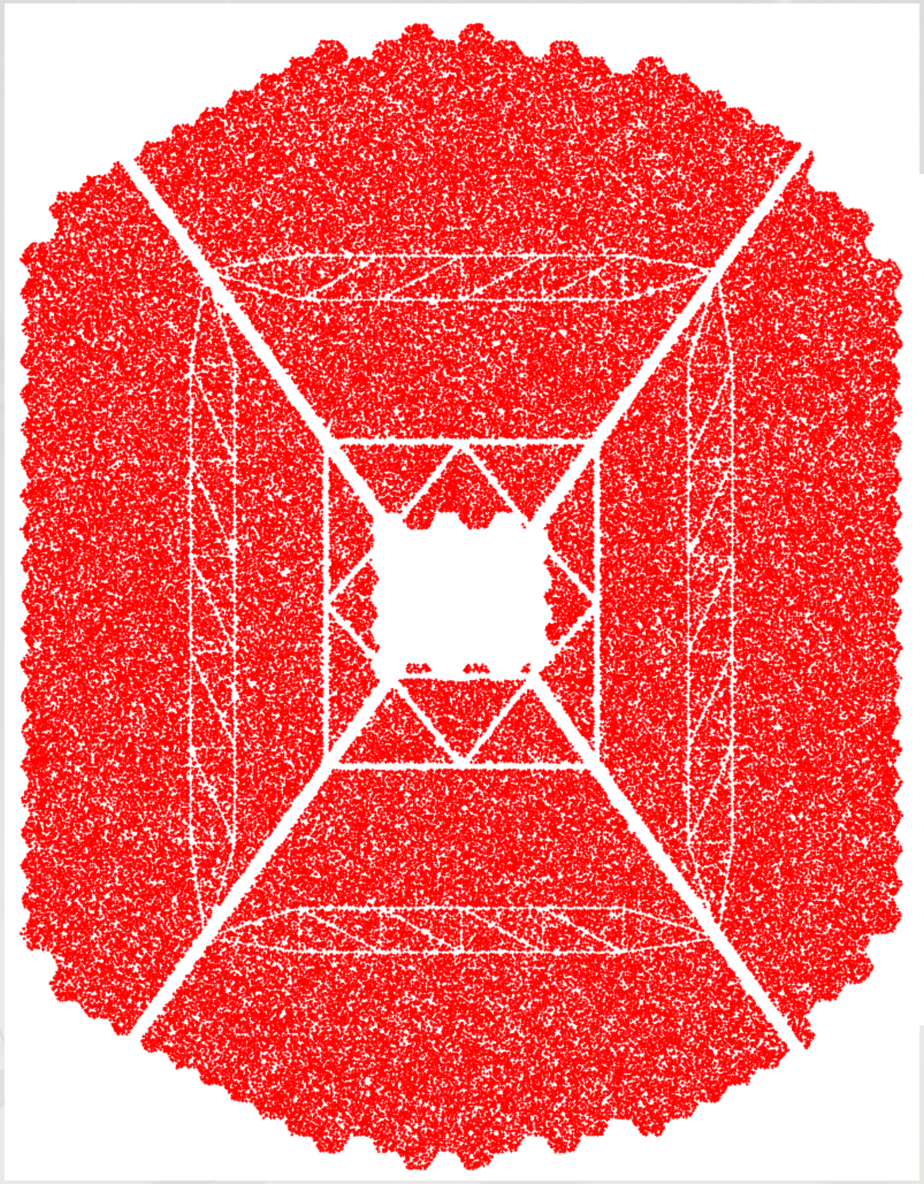
Important optimizations in sim_telarray

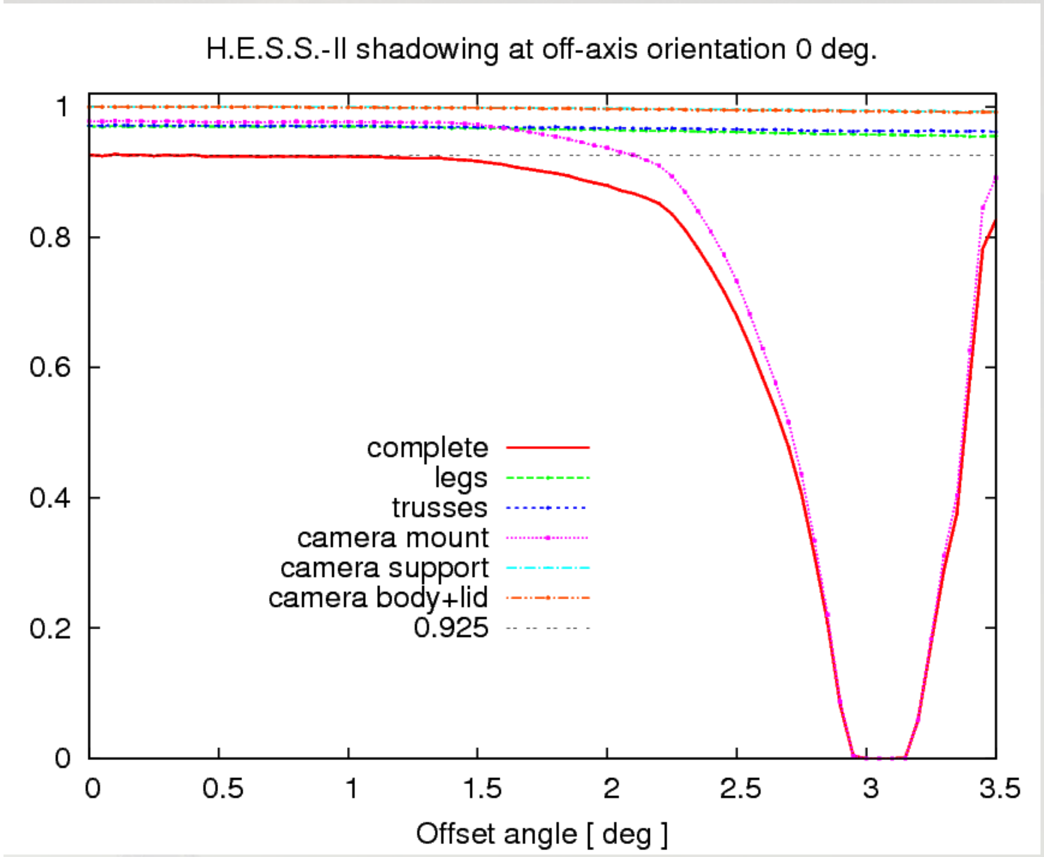
- There are two thresholds before simulating a telescope in an event in full detail:
  - MIN_PHOTONS: With fewer photons (sum of bunch sizes) hitting a telescope, no simulation started.
  - MIN_PHOTOELECTRONS: Ray-tracing and photon detection in pixels is done but with too few p.e. no simulation of the electronics chain is worth doing.
- Processing not in natural order but
  - first apply atm. extinction factor times maximum possible detection efficiency for any pixel first,
  - only for photons surviving here do the ray-tracing and apply remaining efficiency factors.

Many details neglected ...

- either because too CPU intense for prod. e.g.:
  - detailed shadowing of all support structures,
  - recovery of SiPM cells,
- and/or because of lack of knowledge, e.g.
  - noise spectra at trigger & (F)ADC inputs,
- or because implementing them in a backward-compatible way is rather tricky, e.g.
  - reflectivity is function of wavelength and angle.

Shadowing example: H.E.S.S.-II





Validation & Verification

- Discussion with all instrument teams
  - whether sim_telarray procedures are appropriate and accurate enough to represent the instruments,
  - identify parameters that should be updated,
  - collect test measurements and corresponding simulations
    - test measurement data to be archived for use also with later sim_telarray versions and other simulation codes.
- See redmine "Analysis and Simulations" pages for the ongoing process.

Compilation considerations

- For efficiency reasons many arrays in the code are of a pre-processor defined fixed size.
- Compiling with larger sizes than necessary will increase memory footprint, plus stack size:
  - You may also need ‘ulimit -S -s 12000’ or such.
- Where memory is not an issue you can use a version with big arrays (e.g. -DCTA_MAX_SC).
- Most CTA MC productions had fine-tuned sizes – which is the reason why the ‘build_all’ script offers so many options.

Building sim_telarray and friends

- You need
  - sim_telarray.tar.gz and a configuration data package
  - hessioxxx.tar.gz

  then unpack them and build manually:

```
export CTA_PATH=`/bin/pwd`
 (cd hessioxxx && make EXTRA_DEFINES=-DCTA_PROD3)
 (cd sim_telarray && make EXTRA_DEFINES=-DCTA_PROD3)
```

- or the complete package, e.g.
  - corsika7.5_simtelarray.tar.gz

  unpack it in new dir. and build, for example, with

  ./build_all baseline qgs2

Sim_telarray configuration system

- Every configuration parameter has a data type, number of elements, range of values, ...
- Order of parameter setting:
  - compiled-in default values,
  - pre-processing top-level configuration file
    - once for global parameters (e.g. atm. transmission table),
    - once for each telescope (with e.g. -DTELESCOPE=117),
  - command line (e.g.: -C telescope_theta=20) overrides defaults and config file settings.
- Many parameters are file names, with specific configuration data of their own (e.g. pulse shapes, mirror list, camera definition, ...)

Sim_telarray configuration system

- Configuration types can be
  - data (signed/unsigned int, float, double, strings) which is assigned as instructed, or
  - functions which are executed as instructed, e.g.
    ```
    -C SHOW=ALL
    ```
- Parameter names may have abbreviations and are otherwise case-insensitive, e.g.
  ```
  CAMERA_TRANSmission
  ```
  can be used as
  ```
  camera_trans = 0.9  % with optional comment
  ```
  or
  ```
  cAmeRa_tRansmis 0.9   % '=' not needed
  ```

Sim_telarray configuration system

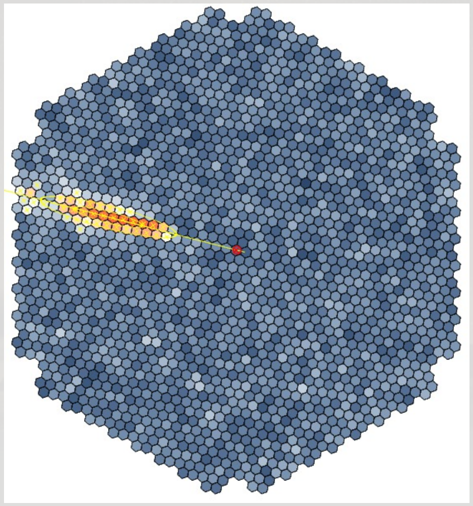
- Multi-element configuration data can be assigned
  - in natural order, e.g.
    secondary_baffle = 340.84,288.88,92.0,0.
  - globally, e.g.
    nightsky_background = all: 0.0549
  - or addressed individually, in ranges, groups of ranges, or with repetion factors, e.g.
    nightsky_background = all: 0.0549, 1777: 0.052
    *nightsky_background = 173-175: 0.045, 5*0.040*
    nightsky_background = (178-180,184,186-188): 0.
- Separation by commas and/or whitespace.
- Be CAREFUL with semicolons; in particular in comments: *par1=7; par2=13 % two lines; in one*

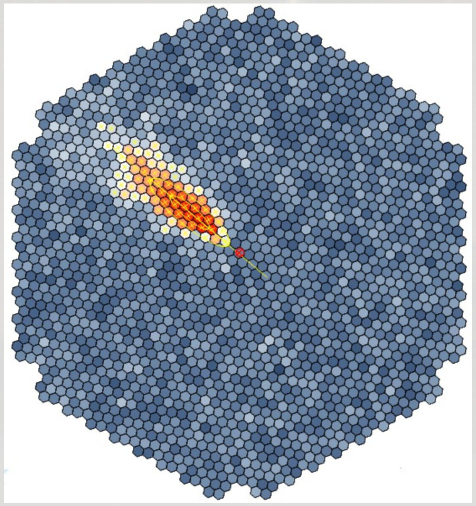
Configuration pre-processing

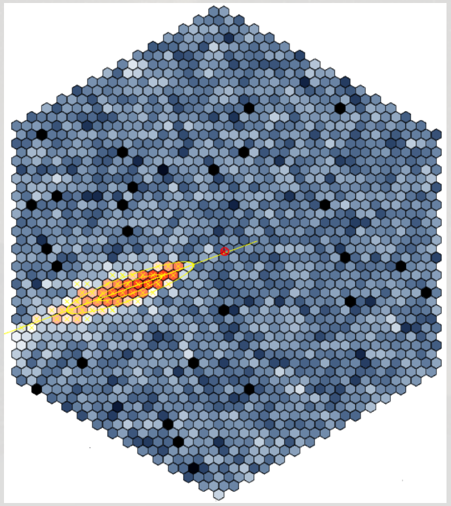
- Sim_telarray comes with its own preprocessor, somewhat similar to C/C++ pre-processor but simpler and macro replacements must be explicit. (You could use another pre-processor.)
- Once global pass with -DTELESCOPE=0
- One pass per telescope with telescope ID.
  - Telescope types configured by telescope ID in the top-level config file

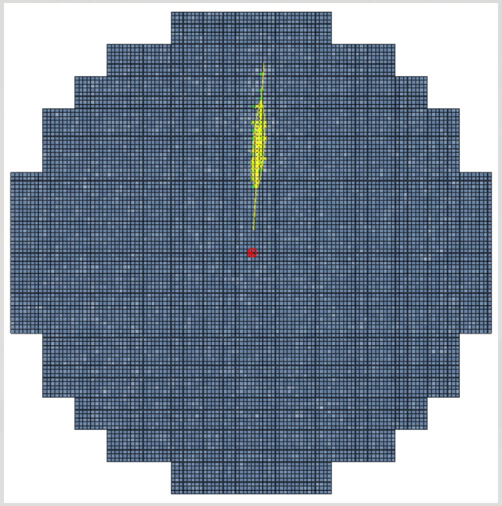
```
...
#elif TELESCOPE <= 41
# include <CTA-ULTRA6-MST-FlashCam.cfg>
```

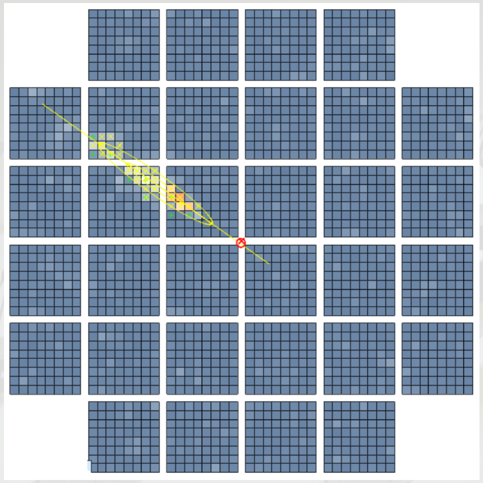
A selection of CTA cameras: Prod-3

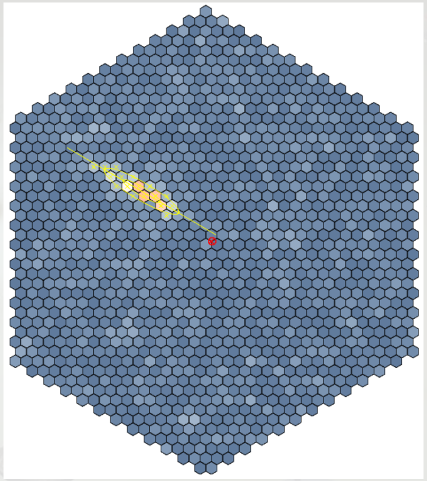

LST

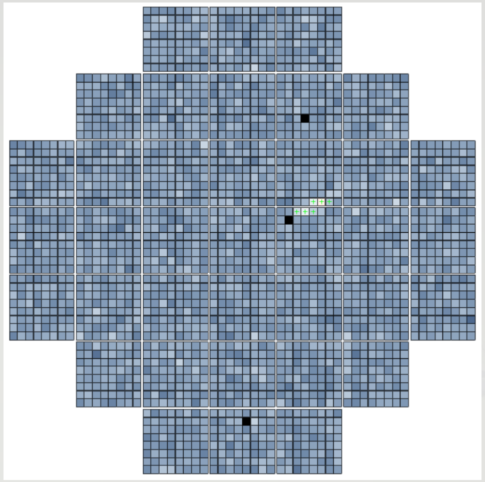

MST-NectarCam


MST-FlashCam


SC-MST = SCT


GCT (CHEC)


4-m DC = SST-1M


ASTRI

Parameter groups

- Global or site-specific parameters
  - e.g. atmospheric transmission
- Telescope optics related parameters
- Electronic pulse shapes, amplitude level and distribution, NSB pixel rates, ...
- Trigger, including majority, analog & digital sum
- Readout, incl. sampl. rate, no. of gains, window, .
- Array trigger
- Online analysis, pixel timing, plotting, output, ...
- Description see documentation and on redmine.

Top-level configuration files

- Command line: -c path/to/config-file -Icfg/CTA
- Default installation places CTA top-level configuration files in sim_telarray/cfg/CTA/.
- Prod-1 configuration:
  - CTA-ULTRA3*.cfg
- Prod-2 configuration:
  - CTA-ULTRA5*.cfg
- Prod-3 configuration:
  - CTA-ULTRA6*.cfg

Separating source/binary/cfg paths

- Installing binaries & libraries to different paths:
  - Binaries include RUNPATH; can be changed with ‘chrpath -r’ but not to a longer path name.
  - You may need LD_LIBRARY_PATH variable.
- You can install configuration files in a different (versioned?) path and either
  - use full paths with ‘-c’ and ‘-I’ options, or
  - use SIM_TELARRAY_CONFIG_PATH variable to completely replay configuration include paths, or
  - use SIMTEL_CONFIG_PATH variable to precede predefined paths with new paths, or
  - use symlink for cfg directory.

Running sim_telarray

- You can run sim_telarray manually – but you may want to look at a command line history from an example file first:
  - read_cta -h example.simtel.gz | less
- You can use the generic_run.sh script for prepared configurations (productions & tests), but some env. variables (CORSIKA_...) needed.
- You can use the production scripts which launch CORSIKA & sim_telarray.
- Dedicated scripts for artificial light sources with sim_telarray following.

Conclusion

- Showed some of the basic features of sim_telarray – and some of its limitations;
- where to find additional sources of documentation, tutorials, and examples;
- where to get the sources and how to build, either just sim_telarray manually or full installation.
- Explained the configuration system and how to use it from its default or from another location.
- Before the next session, please try to build the full package with the ‘build_all’ script as shown!

  ./build_all baseline qgs2 (will need close to 1 GB disk space).