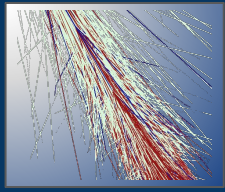


Infrastructure and Procedures

Ralf Ulrich

Note:

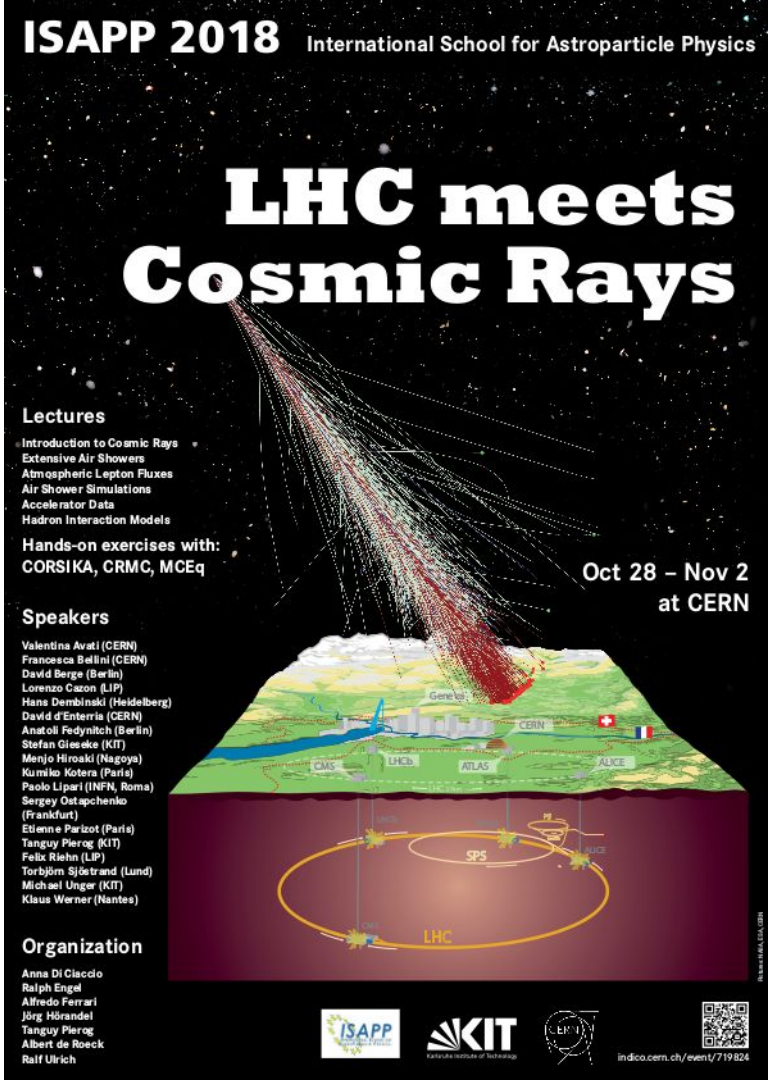
Upcoming CORSIKA school at CERN

Very comprehensive programm,

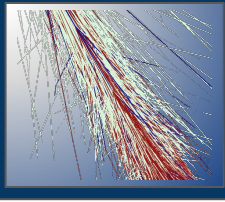
Excellent speakers,

Registration open: NOW

https://indico.cern.ch/event/719824/



Technical infrastructure and services



https://gitlab.kit.ikp.kit.edu/AirShowerPhysics

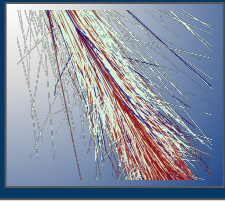
Wiki service for development documentation

Issues (tasks+bugs), and Milestones

Mailing list at KIT: corsika-development@lists.kit.edu

Meetings on: indico.scc.kit.edu

Fast feedback, test+validation

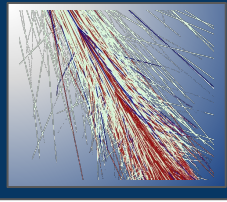


Either use gitlab/CI or jenkins

for all commits and/or pull-requests.

Service will be provided by KIT as part of the hosting.

Dependencies



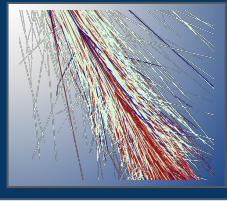
- Avoid third party dependencies
- Consider re-implementation if advantageous
- Hide third party libraries behind custom interfaces.

Likely candidates for dependencies:

- C++14
- CMake
- git
- Doxygen
- PyPind [optional]
- HDF5, (ROOT)
- HEPMC [optional]

- boost:
  - file access
  - command line
  - configuration parser
  - random numbers
  - unit testing
  - physics units
  - Histogramming

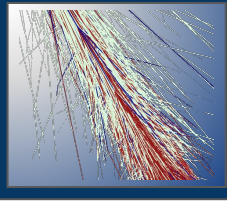
Procedures, Organization



- Developers join the AirShowerPhysics group
- forck code from AirShowerPhysics group
- develop in your own environment
- Prepare pull-request for changes
  - Limited size, only as much change as needed
  - Unit-test evaluation
  - Validation
  - Code review
  - Discussion: (semi) regular virtual meetings, or else. See discussion later.
- Either you work on new features, or fixes/improvements of existing code
  - The proposed change to the repository should reflect that
  - Size of code change as small as possible

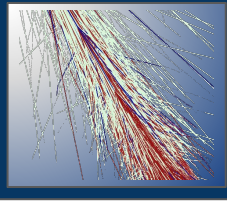
→ need to communicate regularly about progress, status and plans

Coding conventions

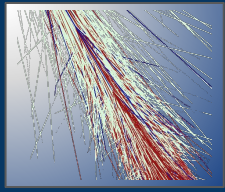


- In order to make this working, we need to agree on coding conventions
- Discuss and decide on conventions,
- Upload conventions on wiki page,
- Evaluate all changes wrt. to this.

Conventions - first incomplete suggestions



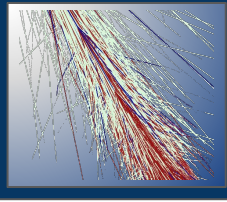
- Good baseline summary: https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines.html
- Code should also visually be synchronized to some level
- Code must be accompanied with inline comments. A well written code is self-explanatory to a large extend.
- Programmers are required to keep the “users guide” up-to-date.
- doxygen commands must be used where possible.
- One aspect of choices of the style should be to minimize the probability of programming errors. For example pointers should be used only where absolutely necessary, and that should never be exposed to the user.
- We will favour static over dynamic polymorphism.
- Test-driven development is encouraged.
- A high coverage of code by tests will be a prime criterion for acceptance.



Discussion on future collaborative work

Ralf Ulrich

Proposed milestones (I)



Milestones 0:

June 2018

ongoing workshop

Milestone 1:

Sept 2018

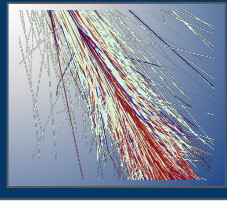
first limited set of (core) support code that will allow collaborators to work on own contributions, implementations: units, transformations, particles, interfaces

Milestone 2:

Dec 2018

Main program loop and first trivial (non-realistic) particle cascade: Heitler-type shower, configuration, output, log-files, (thinning)

Proposed milestones (II)



Milestone 3:

February 2019

first working code that can actually do physics:
SIBYLL2.3, UrQMD, PYTHIA8 for decays, muons physics, multi-thinning

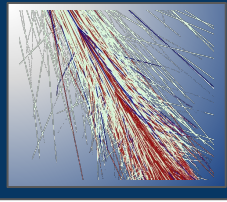
Milestone 4:

Summer 2019

First full shower version: Electron-gamma cascade, parallel, CE, radio, Cherenkov, history

Remember: October 26th 2019: 30 years of CORSIKA

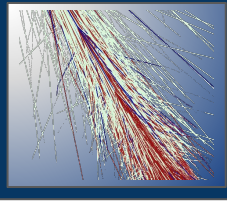
Tasks, essential infrastructure



Roughly ordered by relevance

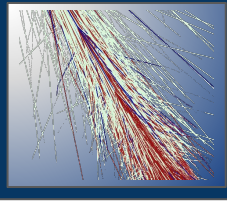
- Unit testing environment
- Physical units
- Configuration
- Coordinates and transformations
- Stack with hooks, and particle data access
- Definition and implementation of first physics processes interfaces
- Output file: studies, first definition and implementation
- Logging, Debugging

Tasks, essential physics



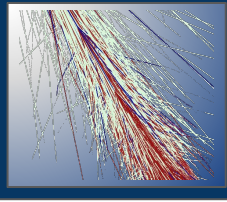
- Tracking in magnetic fields and matter
- Electron-gamma cascades
- Definition and implementation of most basic environment
- Muon physics
- Neutrino physics
- Optical photon physics

Tasks, not essential immediately



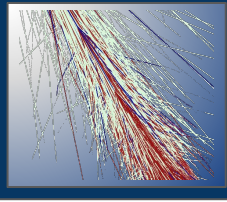
- General GPU-optimization, other very specific coding and evtl. new hardware
- Other hadronic interaction models
- General geometry definition
- Final output file format
- TCP input/output etc.
- Anything related to a better description of physics closer to detectors
- ...

First estimate of effort



Core functionality:	24 person*month
Fundamental physics:	24 person*month
Extended desired functionality:	36 person*month
Maintenance+support:	6 months*person/year

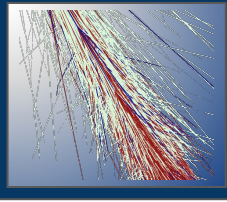
How to collaborate



- “Current” model
  CORSIKA is developed at KIT, additional code is contributed from outside. Collaborators contribute only what is most directly relevant for them.
- “Open” model
  CORSIKA is developed together by groups that make a specified commitment. Defined relevant parts of the main code are maintained by different groups. Tasks are distributed also by importance, not only physics.

There exists a seamless transition between “current” and “open”. Also a mixture is possible.

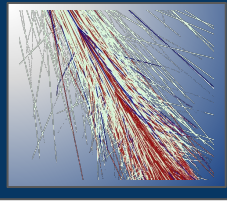
Structures



In minimal case, we do not need more structure than outlined already, including a loose series of virtual (and real) meetings to discuss specific points, progress, status.

In the more close collaborative approach, we need regular meetings and also defined responsibilities to make collaborative decisions.

Direct scientific output of project



- CORSIKA user's guide, 2019
- CORSIKA main physics reference paper: 2019 or 2020

- Eventually:
  - Further papers about air shower physics
  - New major improvement of CORSIKA
  - New major idea of physics application of CORSIKA

Feedback and discussion