CORSIKA8 development - Status report

Source code re-factory

A. Augusto Alves Jr

Presented at CORSIKA development meeting - KIT, Karlshuhe
October 8, 2020

Karlsruhe Institute of Technology

Status

- No source code was removed or modified (mostly).
- No changes in the interfaces. No feature added or removed.
- The framework is basically header only now. Components implemented in FORTRAN (Sibyll, UrQMD) and C/C++ (Pythia and others) will be discussed later.
- External components were removed. I will add them back one by one from now.

Summary of implemented changes

- Include guards removed in favor of `\#pragma once`.
- Header files extensions adjusted. `.hpp` for C++ and `.h` for C.
- `.inl` files introduced where necessary. Post-included and holding classes implementations. All them are now in the the folder `detail`.
- Removed doxygen documentation in the `.inl`.
- Implementations details and other components not supposed to be part of the interface moved to `namespace corsika::detail`. Example: previously nested classes.
- Public interface `namespace corsika`. All the doxygen implementation hosted in the `.hpp` and `.h` files.
- I am making an effort to have one class per file. Not concluded yet.
- Tests and examples moved to dedicated directories.
- External components like Eigen, Units to be placed in `detail/external`.

Building

Basically, the only components needing a building process are:

- Examples, tests and documentation.
- Sibyll and UrQMD. Implemented in FORTRAN.

The FORTRAN components should be built as external libraries to be linked at compile time with applications. Inside CORSIKA8, wrapper classes should be used to expose the necessary interface. Already done for Sibyll but pending for UrQMD. Indeed, implementing interface/wrapper class for every external physics related component should be the standard in CORSIKA8.

About Pythia: I will promote the interface to header only.

Delivery and next steps

- Delivery: two weeks from now.
- Immediate next steps: review and test.
- Roadmap to merge into master:
  - About two weeks: delivery a functional refactored version of framework. Then make sure it is working identically to the original fork version (`4d1baa1d625947ce9c3ca86bd0f281065e41966c`).
  - Freeze and merge back the changes from origin/master into the refactored fork.
  - Make sure refactored framework produces identical results to current the frozen origin/master
  - Tag accordingly the frozen and merge the refactored, finishing the first refactoring round.