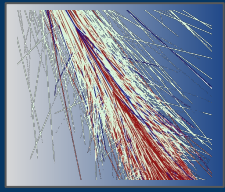
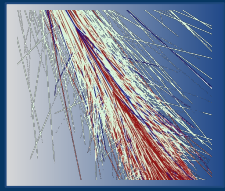


Hands-On
CORSIKA 8 by example

Ralf Ulrich

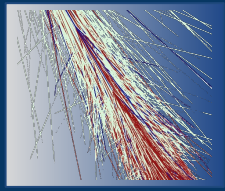
Note



- You can follow the examples and recipes here to a large degree on your own machine

- Preconditions are a recent linux or Mac system, or alternatively docker with root permissions

gitlab.ikp.kit.edu



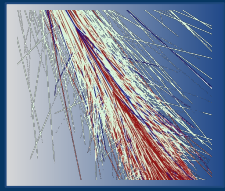
https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika

is a free and open site. You can always visit and observe. You may register on the server to get personalized access to discussions, contributions etc.

If you want to access code via git
(i.e. git clone git@gitlab.ikp.kit.edu:AirShowerPhysics/corsika.git)
you also need to upload your SSH key beforehand. However, you can always download code via https directly.

→ personal registration is very useful, in case of problems mail us: there is a whitelist of domains to prevent spam, we are happy to include your institution!

Automatic unit testing



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

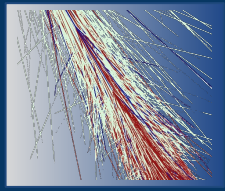
Each push on the gitlab server triggers a series of automatic builds and tests, mostly executed in Mexico.

We always test gcc7 and clang8 debug (-O0) as well as release (-O2) builds.

If tests fail, new code will not be integrated into main development.

There are also special jobs that can be manually triggered to produce doxygen documentation, coverage reports, sanity checks.

docker containers corsika/devel



https://hub.docker.com/r/corsika/devel/tags

Set of prepared docker containers to run CORSIKA 8 in several different Standard configurations. This is also the baseline for the automatic unit testing.

Extremely convenient, but you need root permissions on your system.

Container based distribution will become more important in the future.

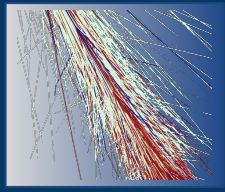
https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika-docker

Independent gitlab repository with description of containers.
See example applications in README.md in particular:

https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika-docker#importing-the-local-corsika-checkout-into-the-container

KIT

Coding guidelines



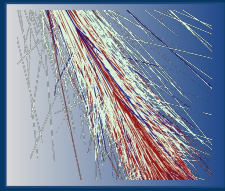
There are a set of basic coding guidelines (still in development) part of the repository, see CONTRIBUTING.md

The purely C++ style is defined as .clang-format file. This is enforced as part of automatic unit testing on gitlab, and you can configure your favorite editor to follow those rules.

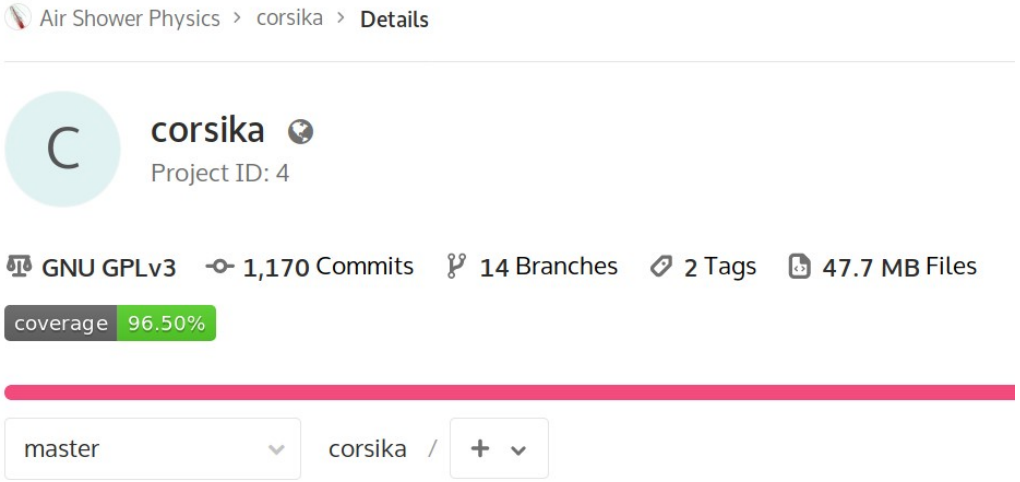
- Code is by definition extremely uniform in style.
- Very easy to further improve starting from where we are now.

Important note to users and developers:
Everything in Framework directory is not intended for normal users. Only developers have to work here.
For physicists/users is is sufficient to look at the Processes, Setup and Documentation directories.

Coverage reports

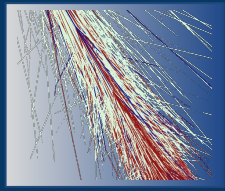


Coverage is determined automatically for each change on the master branch.
Goal: coverage → 100%
Detailed report is available for inspection:
download coverage build artifact, unpack, open with firefox



Unit testing is in OK state, and can be further improved from here.

Issues, Bugs, Feature requests, Discussions



Link to gitlab

This is the most important and direct place to contribute

Issues are discussed, worked on, fixed and lead to progress of the project

Air Shower Physics › corsika › Issues

Open 73 | Closed 130 | All 203 | Edit issues | New issue

Search or filter results... | Milestone

Add number of calls in Process interface — 1
#205 · opened 5 days ago by Ralf Ulrich · First full release: CORSIKA 8.0.0 · Feature request — updated 4 days ago

We need to specialize SecondaryProcess for thinning — 0
#204 · opened 1 week ago by Ralf Ulrich · Full physics demonstratror · Discussion · Feature request — updated 1 week ago

Release-type tests fail for cascade_example — 30
#203 · opened 1 week ago by Ralf Ulrich · ICRC2019 · Important · Bug — updated 1 day ago

Follow-up from "Resolve "baryon and hadron number as particle property"" 0 of 1 task completed — 0
#197 · opened 2 weeks ago by Felix Riehn · First full release: CORSIKA 8.0.0 · Development — updated 2 weeks ago

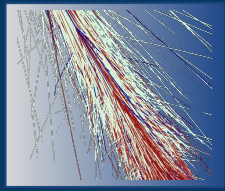
I/O compatible with CXROOT — 0
#191 · opened 3 weeks ago by Tanguy Pierog · First full release: CORSIKA 8.0.0 · Feature request — updated 3 weeks ago

I/O compatible with CORSIKA 7 ? — 1
#190 · opened 3 weeks ago by Tanguy Pierog · First full release: CORSIKA 8.0.0 · Feature request — updated 3 weeks ago

advanced ParticleCut process 0 of 2 tasks completed — 0
#187 · opened 3 weeks ago by Felix Riehn · Full physics demonstratror · Development · Feature request — updated 1 day ago

Select and document clean git workflow — 5

Reporting a bug

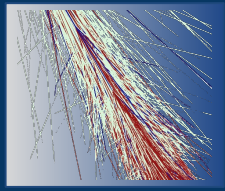


A bug report must contain all information to reliably reproduce the wrong behavior:

- exact version used
- environment
- any special setup or changes
- Bug behavior
- Expected behavior

And: any bug should trigger the automatic unit testing. If it doesn’t, one of the first things to do is to add a new test that demonstrates the failure.

Obtain code on your own computer

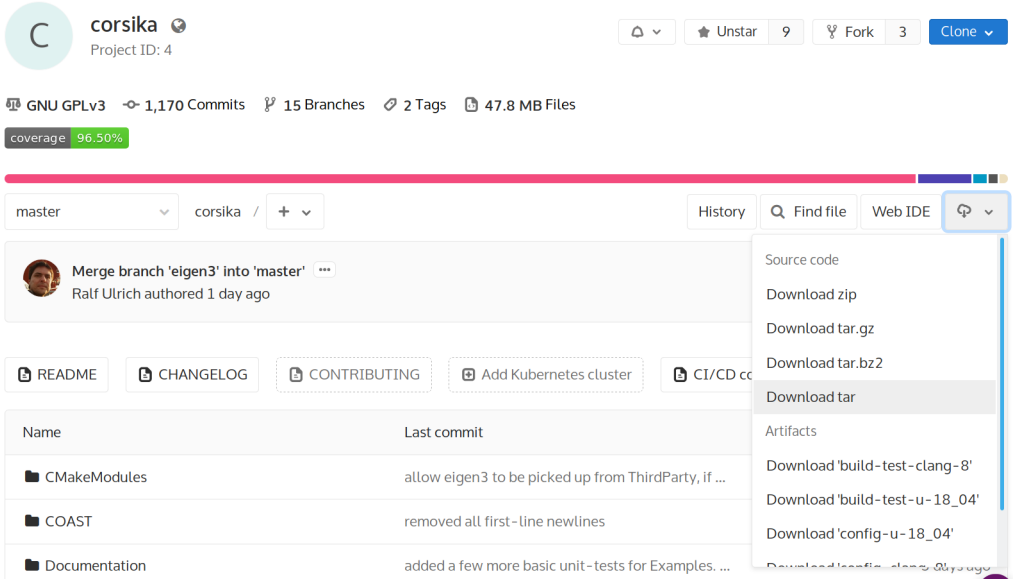


Prerequisites

sudo apt install cmake g++ git

or <u>alternatively</u>

sudo apt install docker.io

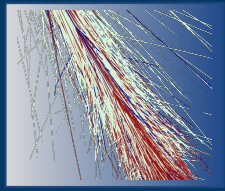


Get Code

either git clone git@gitlab.ikp.kit.edu:AirShowerPhysics/corsika.git
or git clone https://gitlab.ikp.kit.edu/AirShowerPhysics/corsika.git
or <u>alternatively</u> as packed file from
gitlab.ikp.kit.edu/AirShowerPhysics/corsika website
then: unzip/tar xzvf corsika-master.zip/tar.gz

KIT

Compile and run tests



Either with packages (dependencies) installed on your system

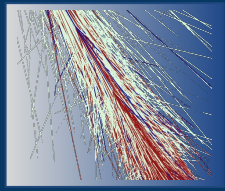
```
cd corsika
mkdir ../corsika-build
cd ../corsika-build
cmake ../corsika
make -j4  &&  make test
```

Or via docker container

```
sudo docker run --rm -it -v `pwd`:/corsika/corsika corsika/devel:u-18.04
mkdir build && cd build && cmake ../corsika/corsika
make -j4 && make test
```

KIT

Simulate air shower

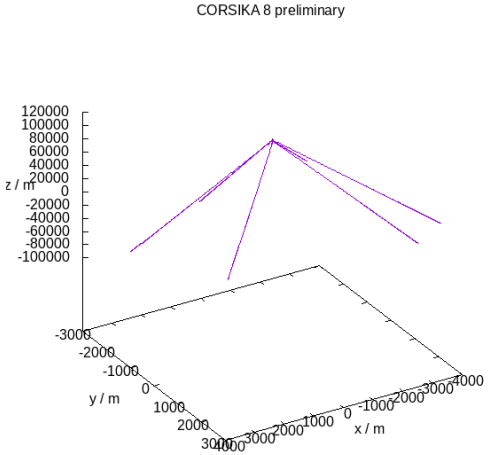


Documentation/Examples/cascade_example

sudo apt-get install gnuplot

../corsika/Tools/plot_tracks.sh tracks.dat

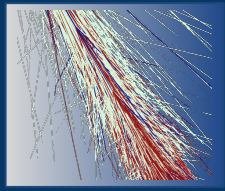
firefox tracks.dat.gif



(vertical 4TeV He shower)

Now:
open file Documentation/Examples/cascade_example.cc
With your favorite text editor
→ Change parameters → run again

cascade_example.cc



Link to cascade_example.cc on gitlab

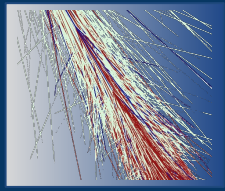
Look for main parameter section:

```
// setup particle stack, and add primary particle
setup::Stack stack;
stack.Clear();
const Code beamCode = Code::Nucleus;
const int nuclA = 4;
const int nuclZ = int(nuclA / 2.15 + 0.7);
const HEPMassType mass = GetNucleusMass(nuclA, nuclZ);
const HEPEnergyType E0 = nuclA * 1_TeV;
double theta = 0.;
double phi = 0.;
```

Modify, make and run again

KIT

cascade_example.cc

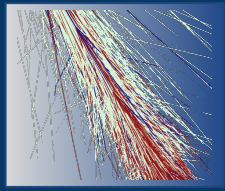


Physics definitions and shower setup:

```
// setup processes, decays and interactions
tracking_line::TrackingLine tracking;
stack_inspector::StackInspector<setup::Stack> stackInspect(1, true, E0);

random::RNGManager::GetInstance().RegisterRandomStream("s_rndm");
random::RNGManager::GetInstance().RegisterRandomStream("pythia");
process::sibyll::Interaction sibyll;
process::sibyll::NuclearInteraction sibyllNuc(sibyll, env);
process::sibyll::Decay decay;
process::particle_cut::ParticleCut cut(20_GeV);

process::track_writer::TrackWriter trackWriter("tracks.dat");
process::energy_loss::EnergyLoss eLoss;

// assemble all processes into an ordered process list
auto sequence = stackInspect << sibyll << sibyllNuc << decay << eLoss << cut
                             << trackWriter;

// define air shower object, run simulation
cascade::Cascade EAS(env, tracking, sequence, stack);
EAS.Init();
EAS.Run();
```

KIT

Documentation – Doxygen



Either run “documentation” step on gitlab server (if you have permissions), then download the result.

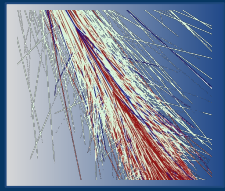
Or locally: install doxygen, and dot, then run make doxygen and look at

Documentation/Doxygen/html/index.html

This is already very useful for developers, however, it is far from complete.

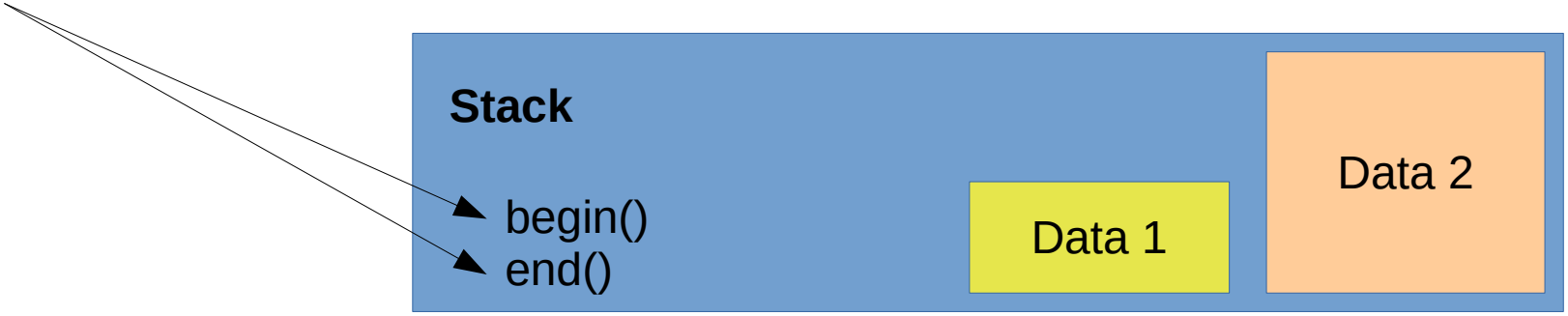
At this workshop: discuss best options for “developer guide documentation” and then work towards releasing this end of 2019.

Particle stacks: from idea to reality



Stack: container of data stored (arbitrarily distributed) in memory

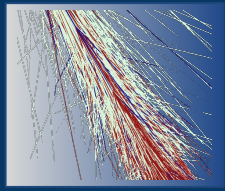
StackIterator: points to one single element (particle) on Stack



Fundamental demonstration with single double:
Framework/StackInterface/testStackInterface.cc

Full demonstration with particle data:
Documentation/Examples/stack_example.cc

Stack(s) in action, e.g. Processes/Sibyll/Decay.cc



Actual Code 06/2019

```
template <>
void Decay::DoDecay(SetupProjectile& vP) {
  using geometry::Point;
  using namespace units::si;

  fCount++;
  SibStack ss;
  ss.Clear();
  const particles::Code pCode = vP.GetPID();
  // copy particle to sibyll stack
  ss.AddParticle(process::sibyll::ConvertToSibyllRaw(pCode), vP.GetEnergy(),
                 vP.GetMomentum(),
                 // setting particle mass with Corsika values, may be inconsistent
                 // with sibyll internal values
                 particles::GetMass(pCode));
  // remember position
  Point const decayPoint = vP.GetPosition();
  TimeType const t0 = vP.GetTime();
  // remember if particles is unstable
  // auto const priorIsUnstable = IsUnstable(pCode);
  // switch on decay for this particle
  SetUnstable(pCode);
  PrintDecayConfig(pCode);

  // call sibyll decay
  cout << "Decay: calling Sibyll decay routine.." << endl;
  decsib_();

  // reset to stable
  SetStable(pCode);
  // print output
  int print_unit = 6;
  sib_list_(print_unit);
```

```
  // copy particles from sibyll stack to corsika
  for (auto& psib : ss) {
    // FOR NOW: skip particles that have decayed in Sibyll, move to iterator?
    if (psib.HasDecayed()) continue;
    // add to corsika stack
    vP.AddSecondary(
        tuple<particles::Code, units::si::HEPEnergyType, corsika::stack::MomentumVector,
              geometry::Point, units::si::TimeType>{
            process::sibyll::ConvertFromSibyll(psib.GetPID()), psib.GetEnergy(),
            psib.GetMomentum(), decayPoint, t0});
  }
  // empty sibyll stack
  ss.Clear();
}
```

See also, Thursday

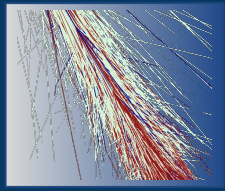
10:50 **DynStack / Memory managment - implementation and features** 20m

Overview of different features and possibilites of an extensive Stack/Memory managment implementation.

Speaker: Dominik Baack (TU Dortmund)

KIT

Framework/Cascade/Cascade.h



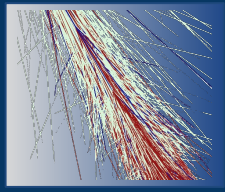
Framework/Cascade/Cascade.h

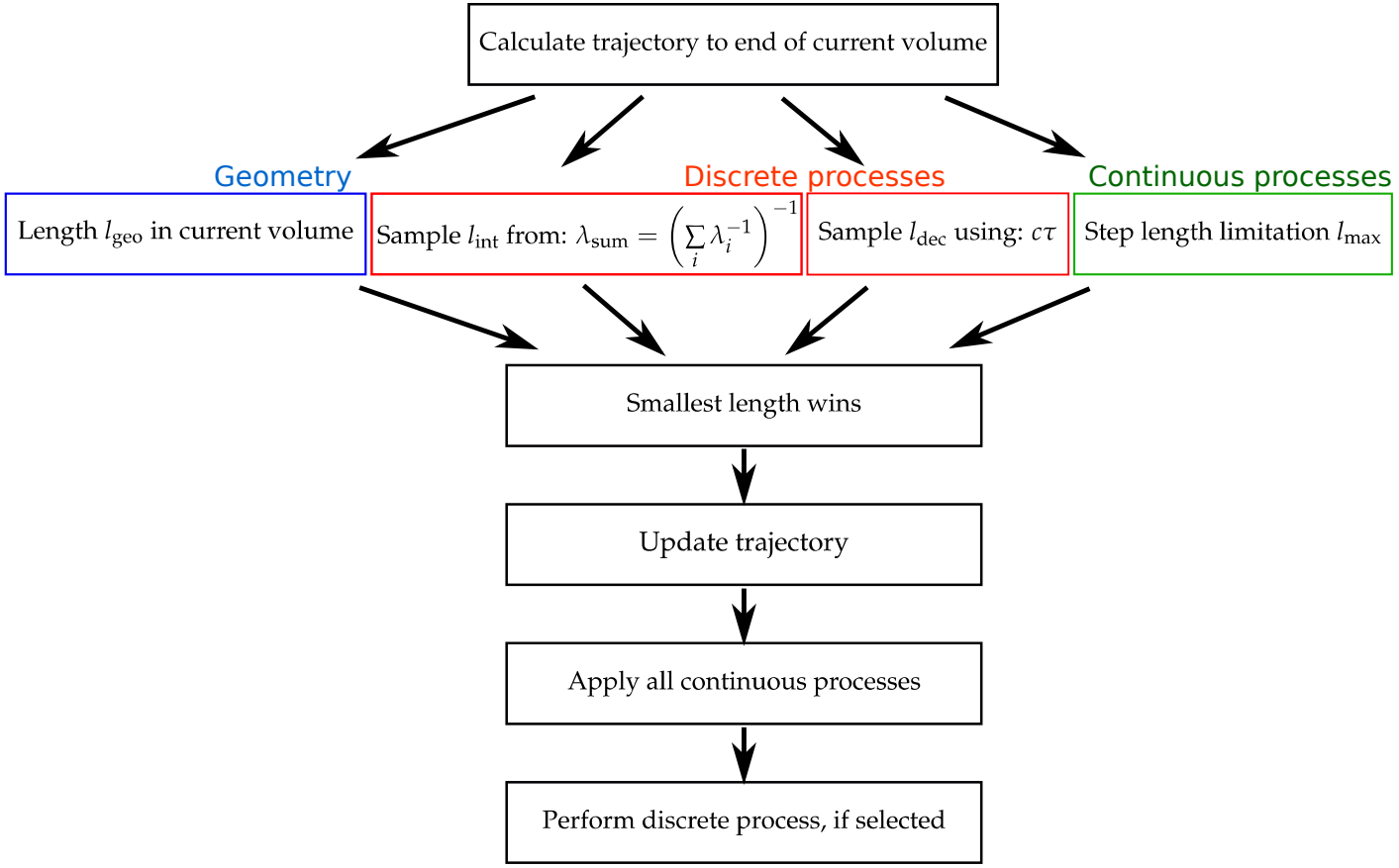
Cascade is the place where Tracking, Physics Processes, and Particle Stacks are linked together to build an air shower cascade.

See Cascade::Run() method.

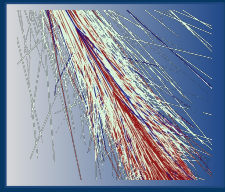
See Cascade::Step(Particle& vParticle) method.

Cascade::Step





COAST

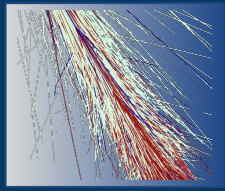


CORSIKA 8 provides an inverse-COAST interface. This allows to write physics Code in CORSIKA 8 framework and run it inside CORSIKA 7 for checks/tests.

- You need CORSIKA 7 installed on your computer, and you need to have COAST
- Installed (via one of the dX options in coconut)
- Set COAST_DIR environment variable to CORSIKA 7 directory.
- Re-configure CORSIKA 8 with “cmake -DWITH_COAST=1”
- Edit COAST/COASTProcess.cc, compile with make
- Set COAST_USER_LIB environment variable to location of libCOAST.so
  i.e. export COAST_USER_LIB=$PWD/COAST
- re-run CORSIKA 7 coconut, select option d3

→ For technical limitations only ContinuousProcesses can be used here.

Summary



Quick tour through the main infrastructure of the CORSIKA 8 framework.

Some insights into most important functionality and mechanisms.

First impression on how to simulate air showers. Many more details during the next days.