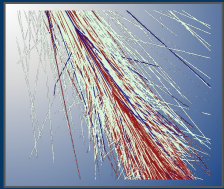
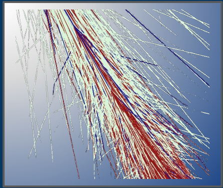


Introduction to the CORSIKA upgrade project

Ralf Ulrich

Scientific aim



A new framework for

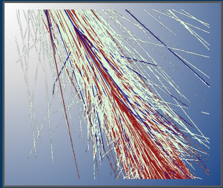
"particle transport with stochastic and continuous processes",

as a stable and solid working horse for astroparticle physics for the next decades,

making most efficient use of expensive and limited scientific and computational resources,

supporting experimental work, as well as physics advances.

Outline, General brief overview

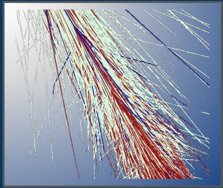


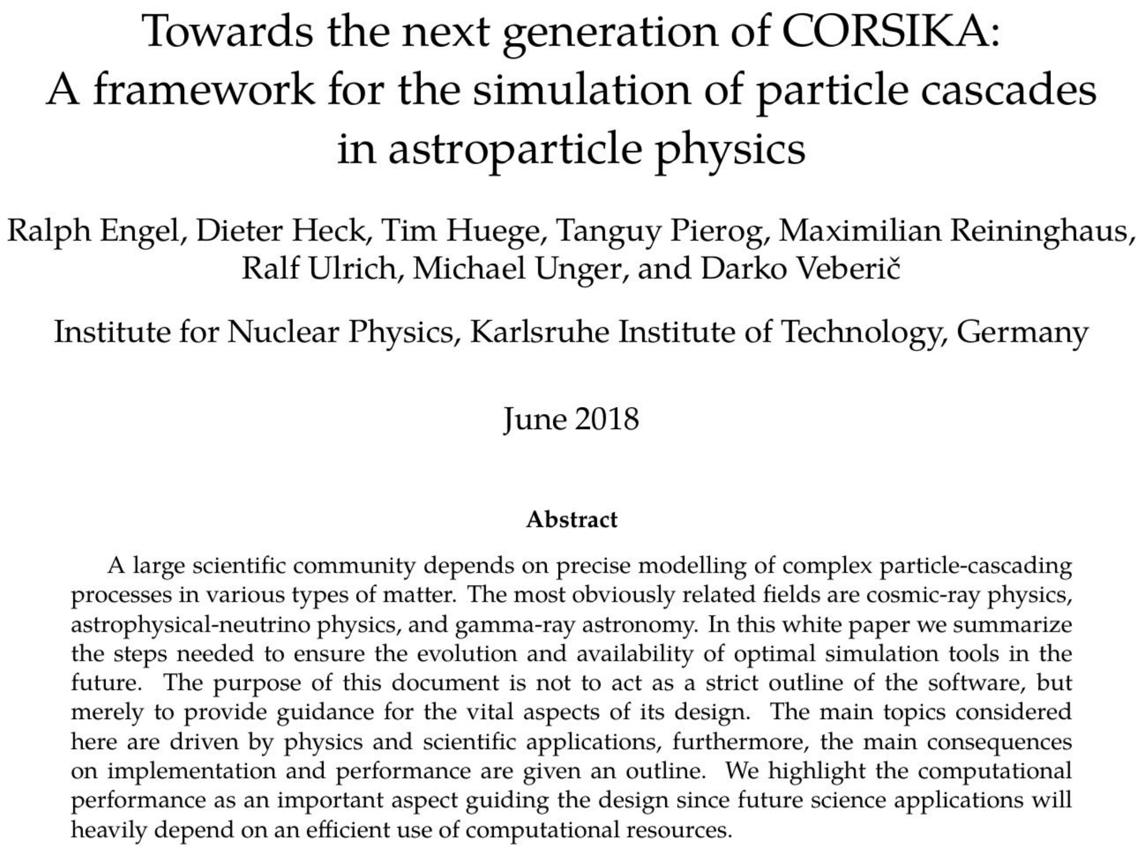
What is needed; important improvements; main considerations; biggest challenges

- We discuss a new project:

  every relevant line of code will have to be typed from scratch.
- Every decision counts, and can be extremely relevant for applications.
- Build on experience from CORSIKA:
  - what has worked,
  - what is important, and
  - where are the existing limitations.

Description of project





Towards the next generation of CORSIKA:
A framework for the simulation of particle cascades
in astroparticle physics

Ralph Engel, Dieter Heck, Tim Huege, Tanguy Pierog, Maximilian Reininghaus,
Ralf Ulrich, Michael Unger, and Darko Veberič

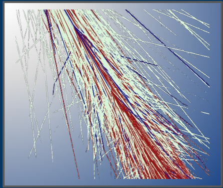
Institute for Nuclear Physics, Karlsruhe Institute of Technology, Germany

June 2018

Abstract

A large scientific community depends on precise modelling of complex particle-cascading processes in various types of matter. The most obviously related fields are cosmic-ray physics, astrophysical-neutrino physics, and gamma-ray astronomy. In this white paper we summarize the steps needed to ensure the evolution and availability of optimal simulation tools in the future. The purpose of this document is not to act as a strict outline of the software, but merely to provide guidance for the vital aspects of its design. The main topics considered here are driven by physics and scientific applications, furthermore, the main consequences on implementation and performance are given an outline. We highlight the computational performance as an important aspect guiding the design since future science applications will heavily depend on an efficient use of computational resources.

CORSIKA status

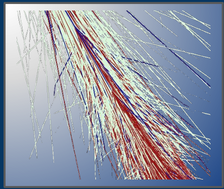


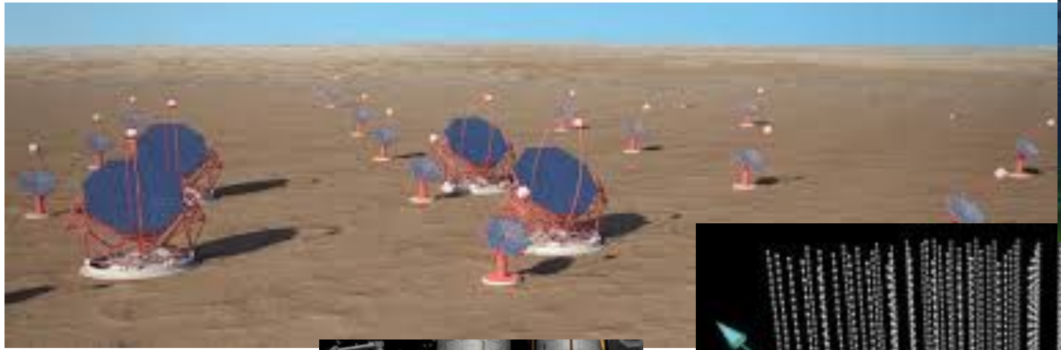
The success of CORSIKA since almost 30 years is one of the foundations of astroparticle physics as it is today.

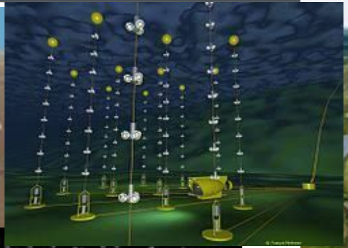
The biggests contribution of CORSIKA was to serve as a common reference frame for cosmic ray related physics.

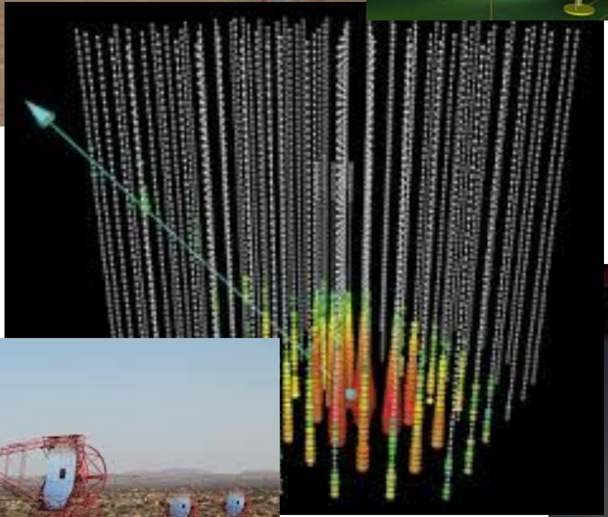
Essentially all experiments are using it.

Experimental landscape (partly)

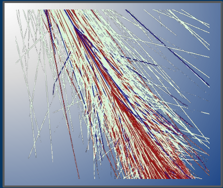






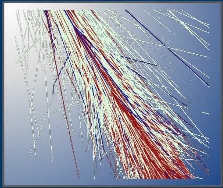


CORSIKA evolution



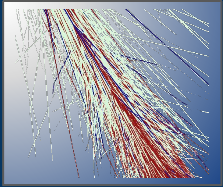
- CORSIKA started as dedicated tool for KASCADE here at KIT.
- It was rapidly adapted to other experimental environments and requirements and has continued to be grow since then.
  → CHERENKOV, THINNING, CURVED, SLANT, CONEX, PARALLEL, ...
- FORTRAN77 with extensive use of pre-compiler logic and a lot of highly efficient explicitly coded optimization.
- Significant complexity of code. Very hard to further extend, understand, maintain and debug.

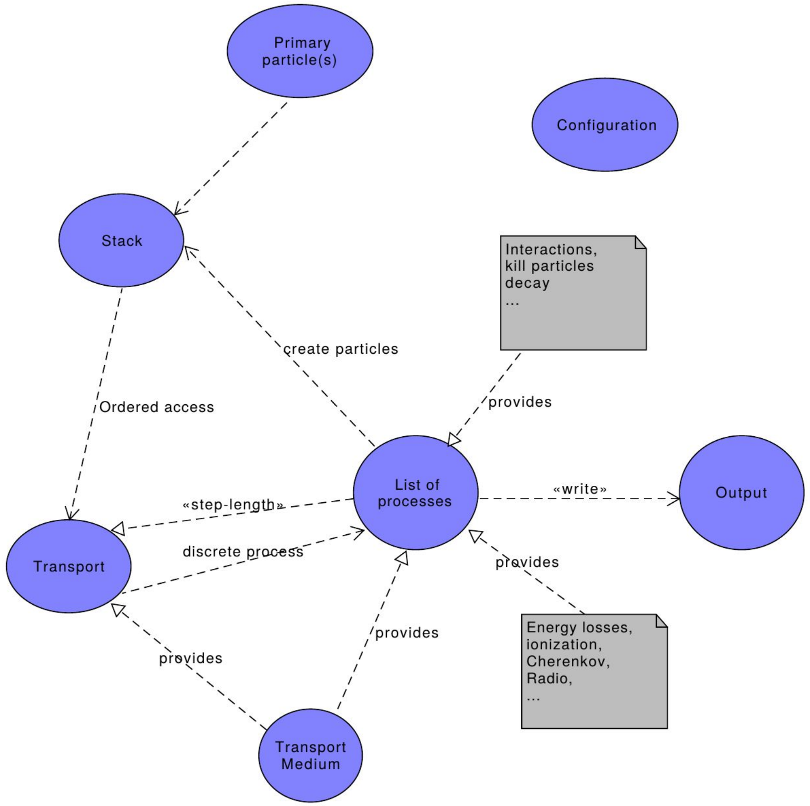
Limitations of CORSIKA so far



- Interaction medium is air.
- Processes, like Radio emission, cannot influence the main simulation loop to e.g. change the step lengths, etc.
- Every process must be able affect the tracking/step-size,
- The system of “processes” and “interaction models” is very rigid,
- Missing physics, e.g. kaon oscillations, and quantum numbers: helicity, spins…
- Much more flexible output options, single-file tier,
- Much more powerful HISTORY options,
- No upward going Cherenkov photons.

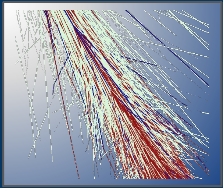
Particle cascade process





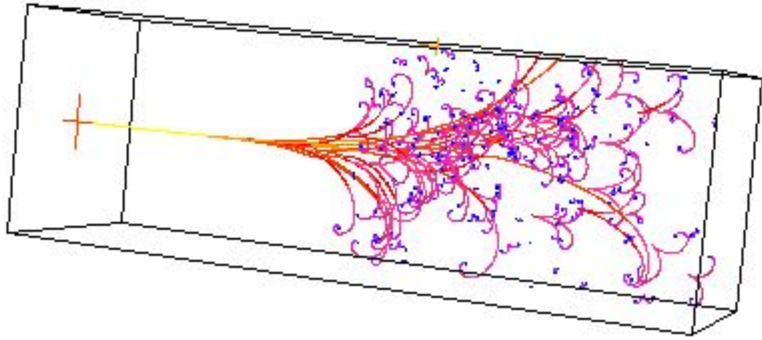
Main challenges

Electron gamma cascades



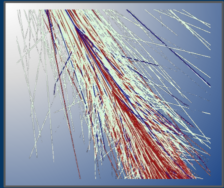
The electron gamma cascade is maybe the most important part of the shower

- We assume we can model it accurately
- It needs most of the computation resources
- It is directly responsible for most: Cherenov, Radio emission



Currently:
customized EGS4 code, there is special physics in CORSIKA missing in essentially all other similar projects, e.g. LPM effect.
However, basically impossible to port into a new C++ project.

Possible solution



1. Start with CONEX electron-gamma code (also EGS4)

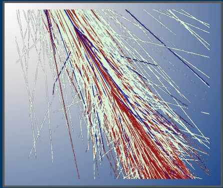
2. Link to GEANT4 processes

3. Use other version of EGS (EGCnrc)

4. Full re-implementation of physics

5. Other solution

Random numbers

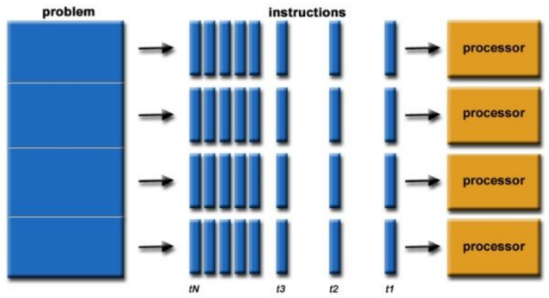


Random number generation in an inherently multi-core and parallel environment while ensuring the full reproducibility of simulations.

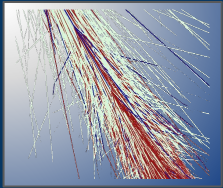
We need to be clear about what is really needed and required.

This can quickly become a very significant complication.

Parallel Computing



Full and transparent integration of cascade-equations, full transport, and thinning methods

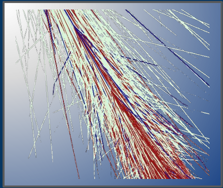


- **Plan**:
  Seamless combination of different computing methods in user-defined phase-space of the cascades.

- **Where are the limits?**
  How can dE/dX, Cherenkov, lateral structure, radio production etc. be simulated in cascade equation?

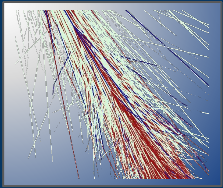
This might be one of the most relevant questions to answer. The impact on accuracy and computational resource needs can be enormous.

GPU optimization



- What the calculations best ported to GPU?
- How do we best keep code that runs both on CPU and GPU 100% synchronized?
- What are most intelligent ways to transport data between CPU and GPU?
- What type of GPUs to support?

Supercomputer applications



Which technology for parallel running: MPI?

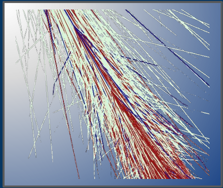
Combine parallel and GPU for maximum speed?

What are the requirements from the community, and how to serve them:

- Ultra-high energy showers, can run months on a single CPU
- Ultra-high number of small energy showers for TeV gamma-ray and neutrino background calculations. Typically very small number of particles on ground.

Some details

Programming considerations



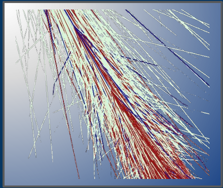
Provide assistance to avoid many simple errors, mistakes and repetition of trivial code.

Physicists should never be forced to know in what values of parameters in a specific “reference frame”, they work with variables explicitly in the correct frame:

- Coordinate systems
- Lorentz transformation
- Particle IDs
- Physical units

For scientists for good coding practice, and sufficient documentation.

Framework

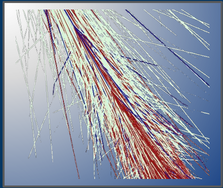


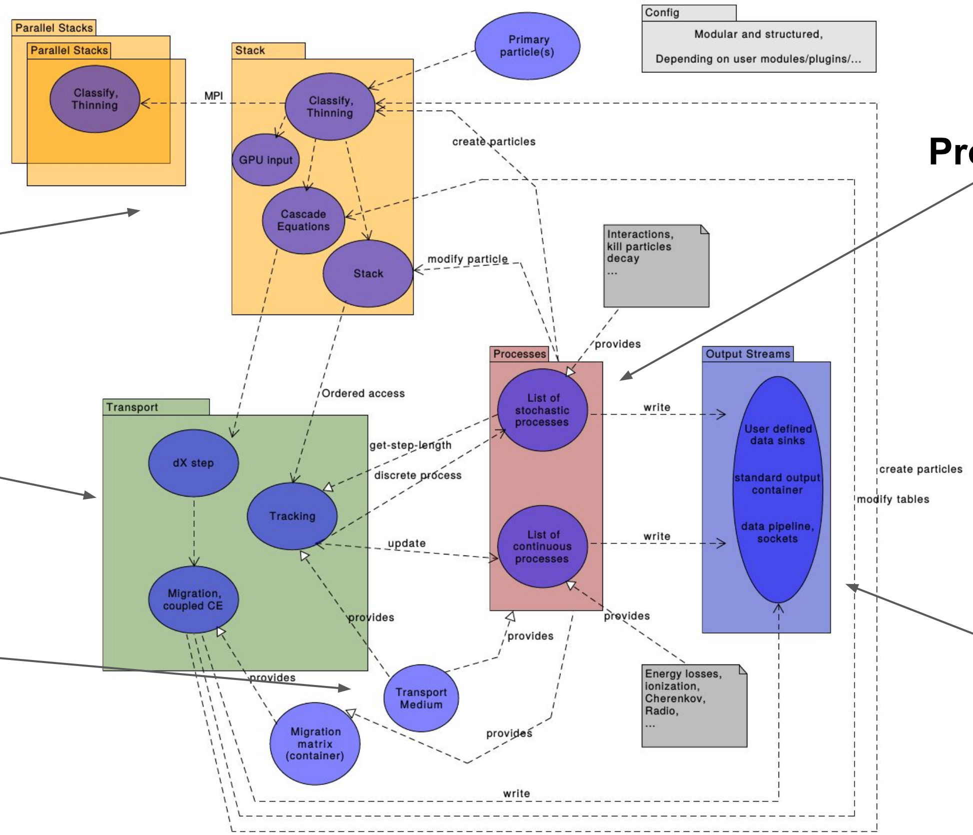
There is a core part, where all calculations are strictly done using only the provided infrastructure of the framework,

there is also an external part, where calculations are done fully outside of the scope of the framework,

→ thus, there is a frontier, depending on the boundary between core and extern code, where all conventions must be carefully adapted and converted.

More realistic picture





Stack

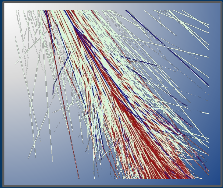
Tracking and stepping

Environment

Processes

Output

Particle production and multiplicative cascades



In a cascading process at ultra-high energy the storage and management of particles is one of the most important tasks.

Consider an electromagnetic cascade, with a typical 1$\rightarrow$ 2 splitting at each vertex.

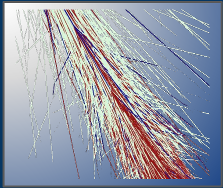
There is memory for 50 particles in the program.

You can keep particles from 2^n=50, n=5 steps in memory

But if you at every collision process the lowest energy particle first, you very quickly come to the level where particles are removed from the cascade.

In this example you can simulate 50/2 = 25 steps with the same resources.

Some notes on the stack



The stack is where particle data is stored.

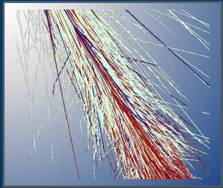
We need no particle object class, just a reference on the stack.

Stack can be anything internally (fortran, std::vector, std::map, file, combination of everything), only the interface must be well defined.

For physicist and users this must look like:

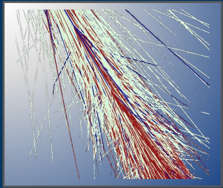
```
for (auto particle : stack) {
   particle.GetEnergy(showerFrame);
   ...
}
```

The main program



Task: empty the stack → then finished

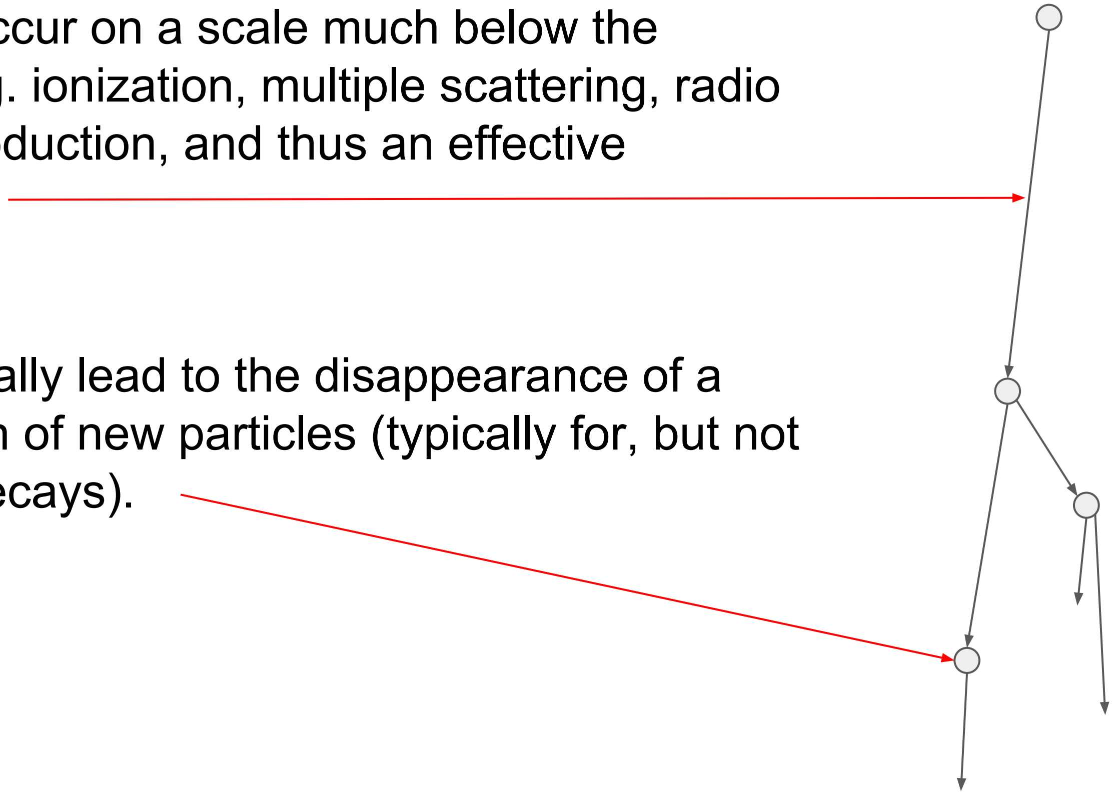
```
stack.add(primaryParticle);
while (!stack.Empty()) {
  while (!stack.Empty()) {
    auto particle = stack.Get();
    Step(particle);
  }
  cascadeEquations.Solve();
}
```



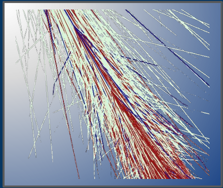
Tracking through medium, and processes

- Continuous processes occur on a scale much below the transport step-length, e.g. ionization, multiple scattering, radio emission, Cherenkov production, and thus an effective treatment can be used.

- Discrete processes typically lead to the disappearance of a particle and to production of new particles (typically for, but not limited to, collisions or decays).



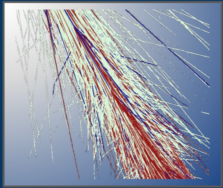
One simulation step



1. Determine step length
2. To transport
3. Apply all continuous processes
4. Apply one discrete process

```
Step(auto particle) {
  auto stepLength = MinimalStepLength(tracking, continuousProcesses, stochasticProcesses);
  auto trajectory = tracking.Propagate(particle, stepLength)
  for (auto cp : continuousProcesses) {
    cp.Propagate(particle, trajectory, stepLength);
  }
  // randomly select ONE or NONE stachastic process
  if (discreteProcess dp = SelectStochasticProcess(stepLength)) {
    dp.Interact(particle);
    }
}
```

Atmosphere → Environment



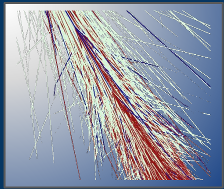
We need to supply a lot more information on the environment compared to before. The environment must be a plug-and-play object, where functionality can be provided for whatever is needed for the scientific application. The interface could include that, but this is not exclusive:

```
Environment::GetVolumeId(point)
Environment::GetVolumeBoundary(trajectory)
Environment::GetTargetParticle(point)
Environment::GetDensity(point)
Environment::GetIntegratedDensity(trajectory)
Environment::GetRefractiveIndex(point)
Environment::GetTemperature(point)
Environment::GetHumidity(point)
Environment::GetMagneticField(point)
Environment::GetElectricField(point)
```

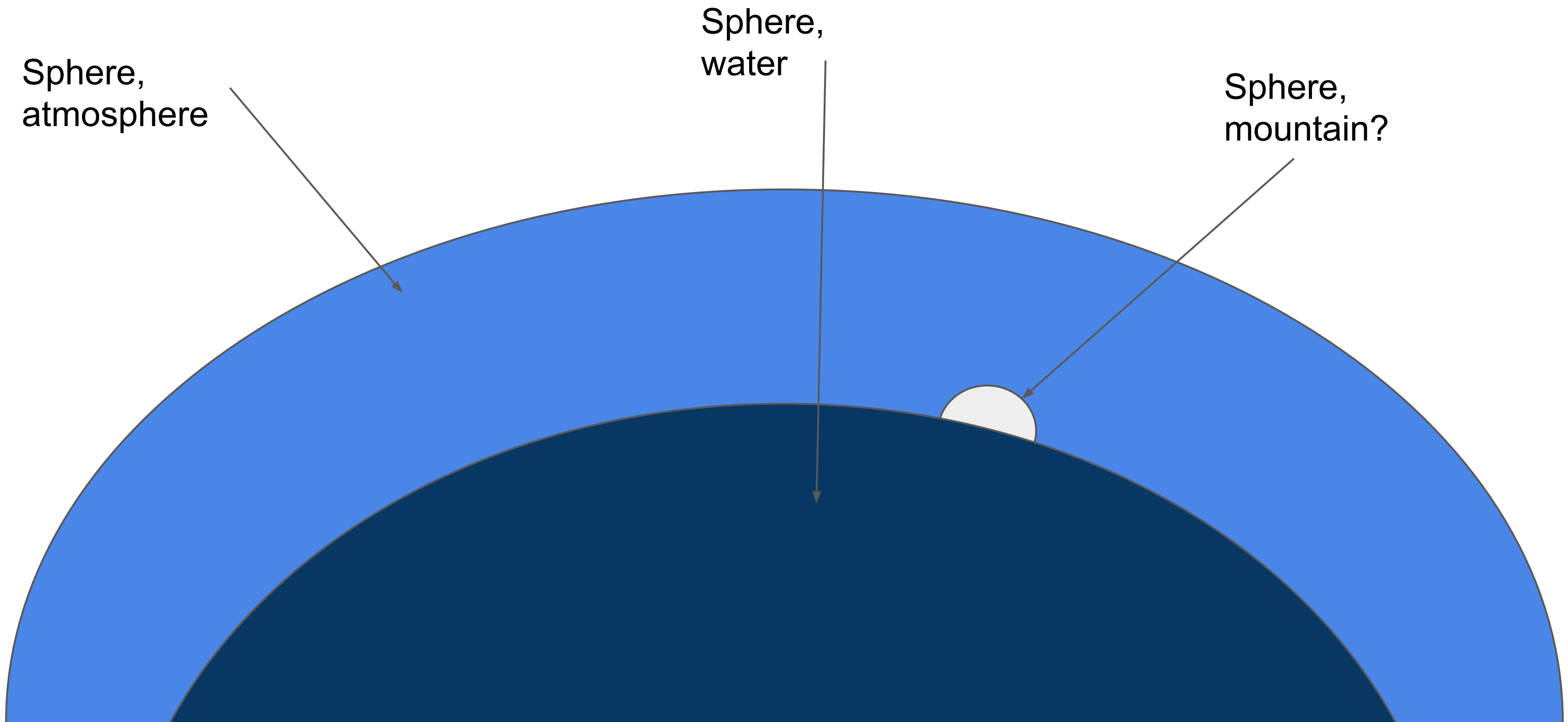
All of this is highly relevant for computing speed. The data from the environment is typically needed in each single step.

Question: what is the consequence and impact on GPU code?

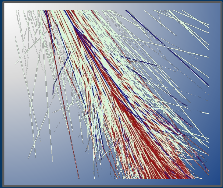
Geometry example



Probably start with sphere-only geometry. Add later what is needed in addition.



Summary



CORSIKA is a very successful program, very versatile, and extremely widely used for

- Planning and design of science cases and experiments,
- Event simulation, reconstruction and data analysis,
- Physics interpretation.

Increasing complexity requires major changes in CORSIKA to keep it functional for the next decades.

Improvements on: modularity, accuracy, flexibility, and emphasis on efficiency.