CLSim

“OpenCL simulation”

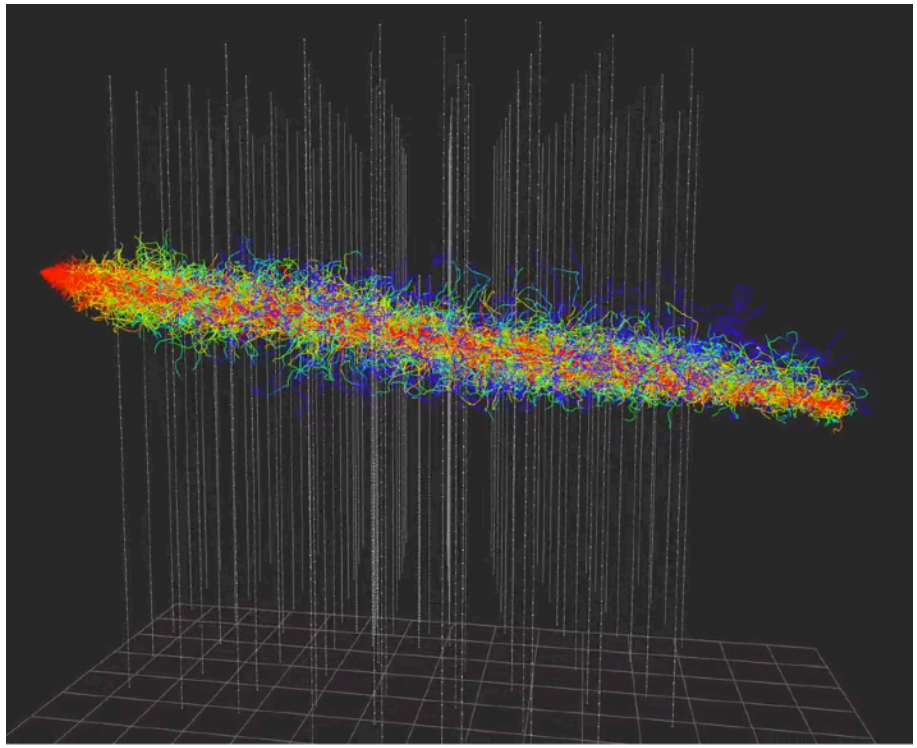
Jakob van Santen
IceCube Simulation Workshop, 2020-10-19

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

DESY.

History

- Originally written by Claudio Kopper (then NIKHEF) for KM3NeT, ca. 2011
- Implemented in SeaTray, a fork of IceTray
- Photon propagation in water/ice on GPUs and CPUs, implemented in OpenCL (source generated and compiled at runtime for settings and target device at hand)
- Particle propagation in Geant4
- Now: entire IceCube simulation chain between initial particle generation and electronics simulation



Functionality

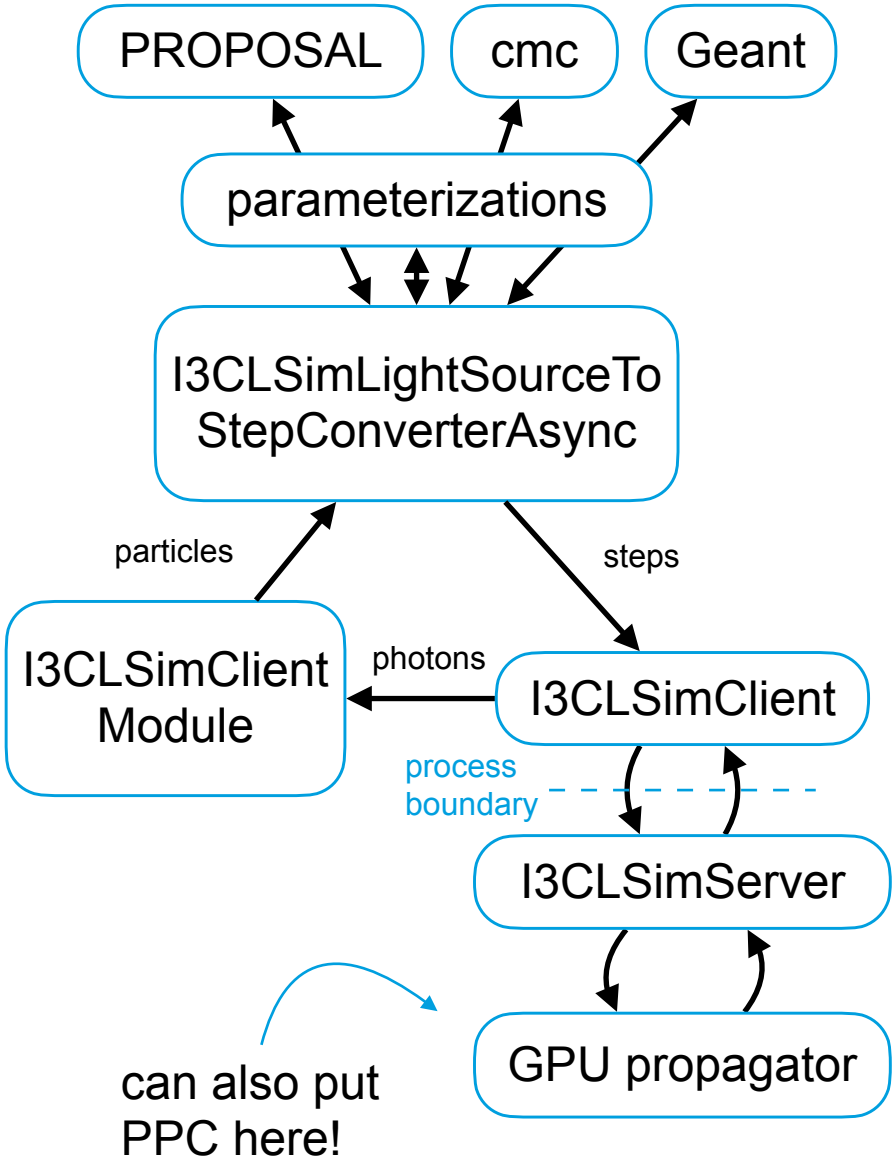
- **Photon tracking**: short segments of Cherenkov-emitting track (“steps”) -> Cherenkov photons on the surfaces of spheres (“photons”)
- **PE conversion**: photons -> photoelectrons on cathodes of specific PMTs (“PEs”)
- **Particle propagation**:
  - IceSim-style: initial muons & cascades -> elementary light sources (track and cascade segments), elementary light sources -> steps (via parameterization)
  - Geant4: initial particles -> steps
- **Tabulation**: steps -> photon fluxes in spatial cells
- **No ice model fitting**: models taken from ppc

Special considerations

- Photon propagation exploits **parallelism** of GPUs
- Individual events need to be grouped (or split) to fit in one invocation of the propagation kernel
- Variable-size intermediate objects (propagated particles, detected photons) can become a problem when thousands of frames are in flight
- -> integrate particle propagation and photon->PE conversion into photon propagation loop

Architecture: multi-process CLSim

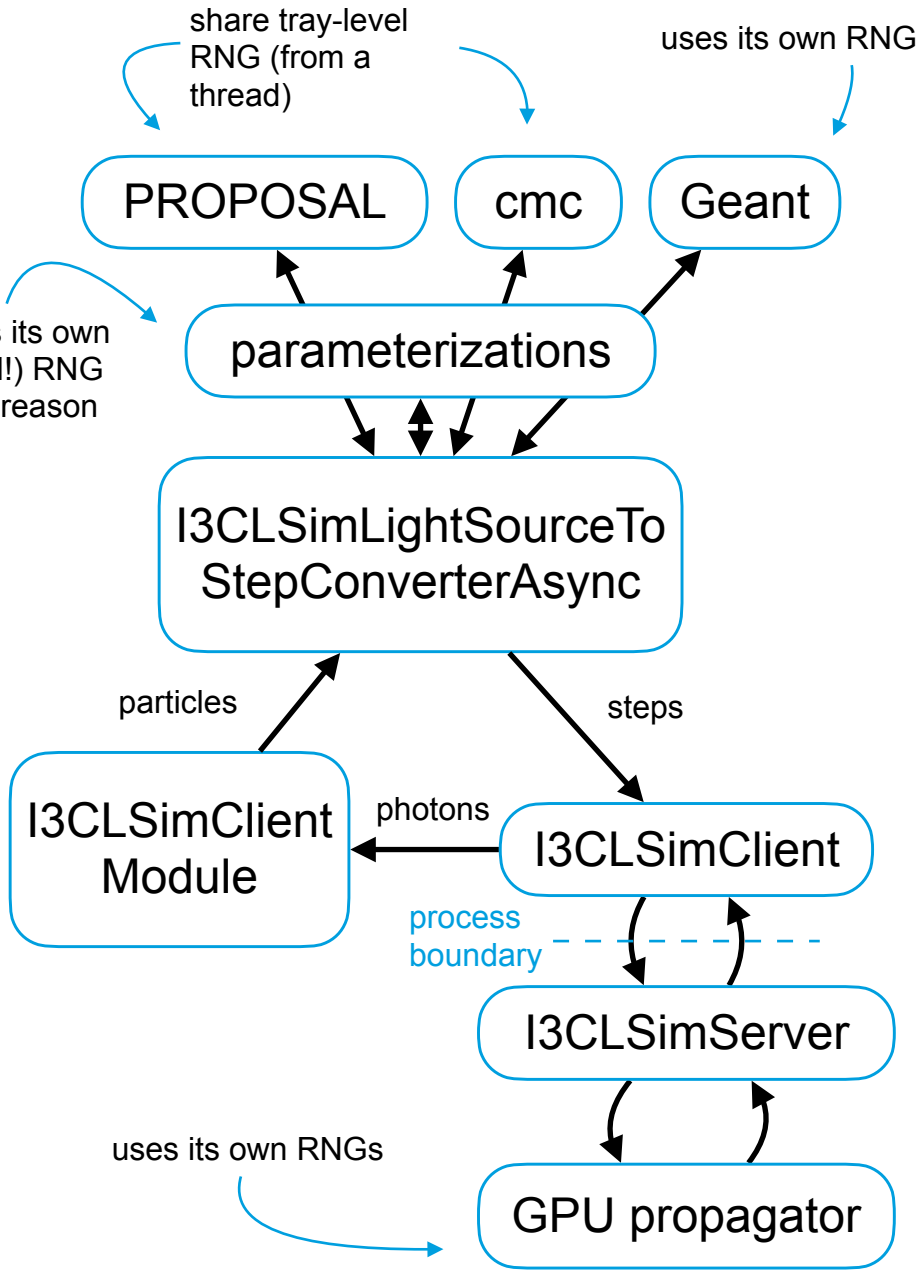
- I3CLSimClientModule harvests light sources from frames
- I3CLSimLightSourceToStepConverterAsync converts light sources to steps (interacts with threads in I3CLSimClientModule)
- I3CLSimClient/Server communicate with (potentially shared) GPU propagator
- Most components implemented in abstract interfaces and can be swapped out at runtime. User-swappable components are single-threaded.



Limitations

Weak reproducibility

- Each GPU thread uses an independent random number sequence. Good?
- But: random stream is tied to the thread -> results depend on mapping of work items to threads
- New problem: results also depend on the number of photons previously handled by the same thread
  - Clients interfere with each other
  - Server maintains state between propagations
- Results are reproducible, but only if all functions are executed by the same threads in the same order



Is OpenCL dead?

- NVIDIA won the GPU wars, and never liked OpenCL
- No OpenCL implementation for NVIDIA/POWER9 (no fancy DoE supercomputer for you!)
- Apple has also lost interest
- But: CLSim is not strongly tied to OpenCL.
  - Abandon incrementally.
  - Can replace the propagation implementation (c.f. CUDA/Optix backport from NVIDIA https://github.com/RamonaNV/offline_production)
  - Some features (e.g. the ubiquitous GetOpenCLCode()) may need a re-think

Upgrade/Gen2 readiness

- Geometry assumes spherical modules, all with the same radius. “Pill-shaped” module support exists in a branch.
- Local hole ice scattering still languishes in a fork.
- No general support for detailed structure of modules, harnesses, etc.
- Variety of photon->PE converters for Upgrade modules, of varying quality. Should be finalized, ideally before Upgrade data shows up.

Discuss