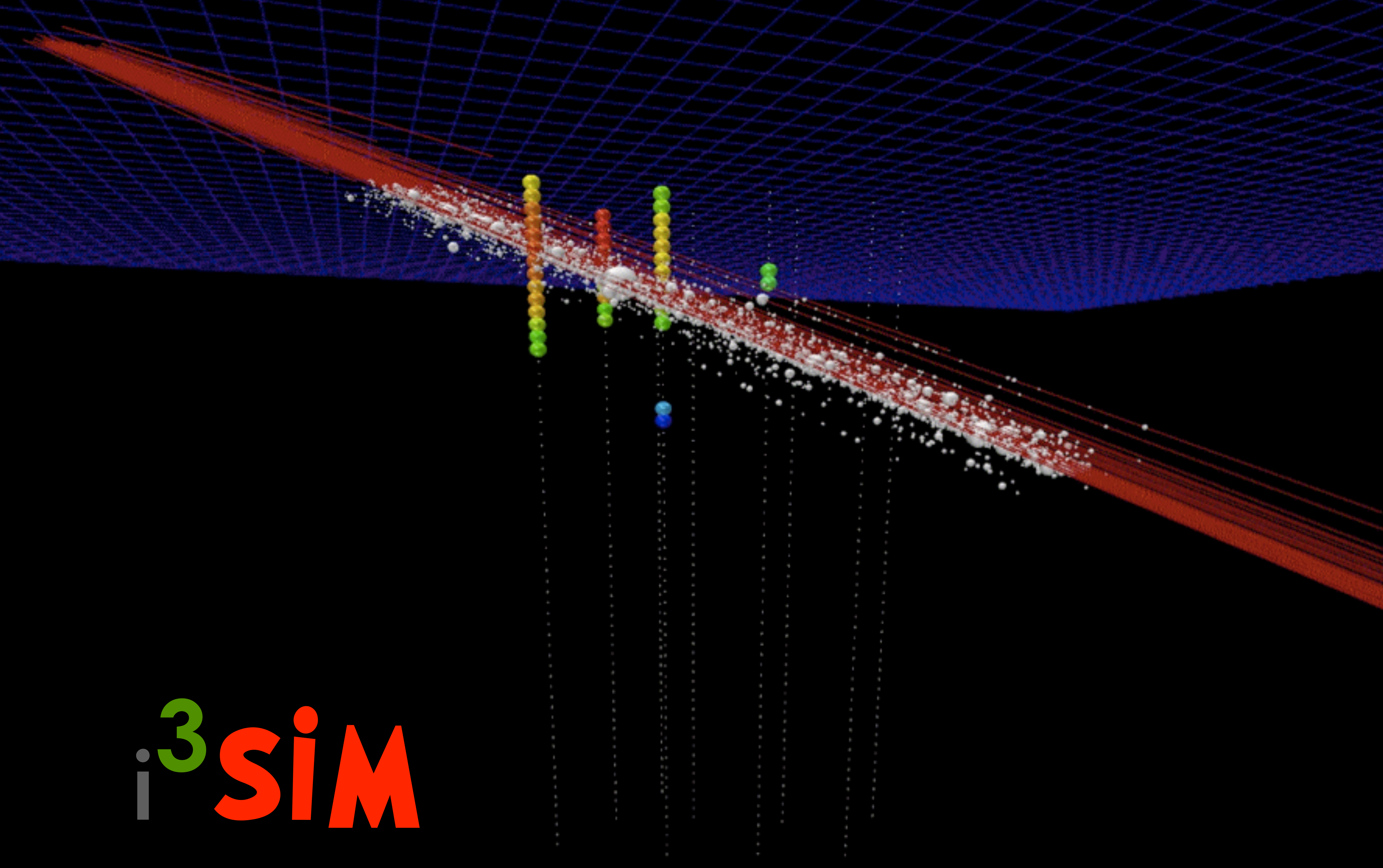


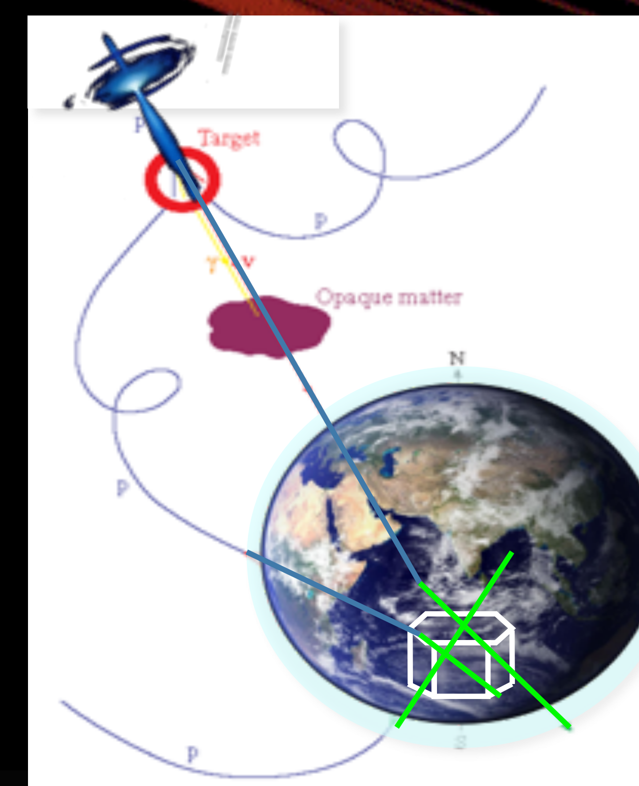
i3sim

IceCube Bootcamp 2016

Juan Carlos Díaz Vélez

Motivation

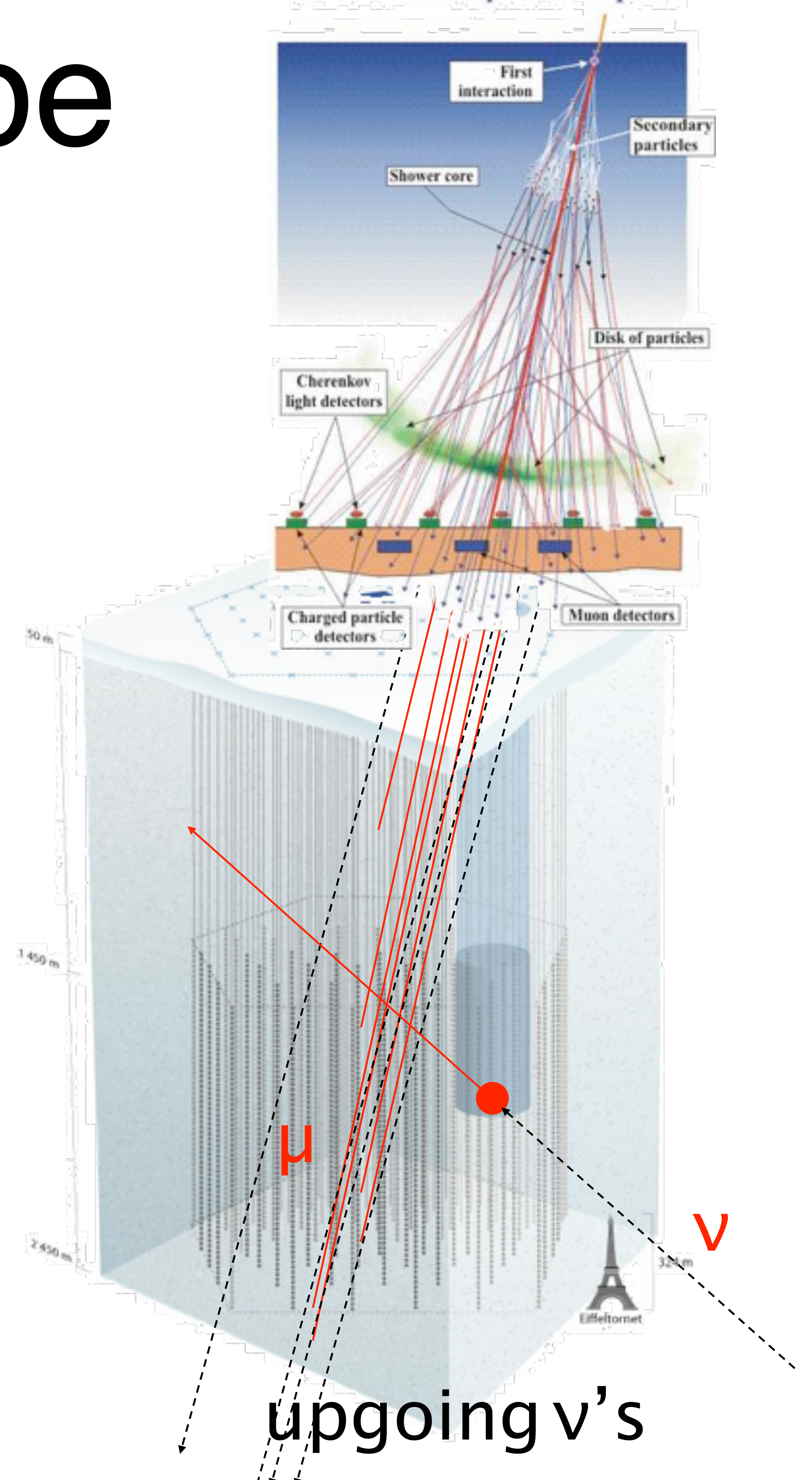
1. Need to understand our detector
2. Test reconstruction algorithms
3. Understand the backgrounds in our analyses
    1. Atmospheric muons
    2. Atmospheric neutrinos
4. Develop and test blind analyses



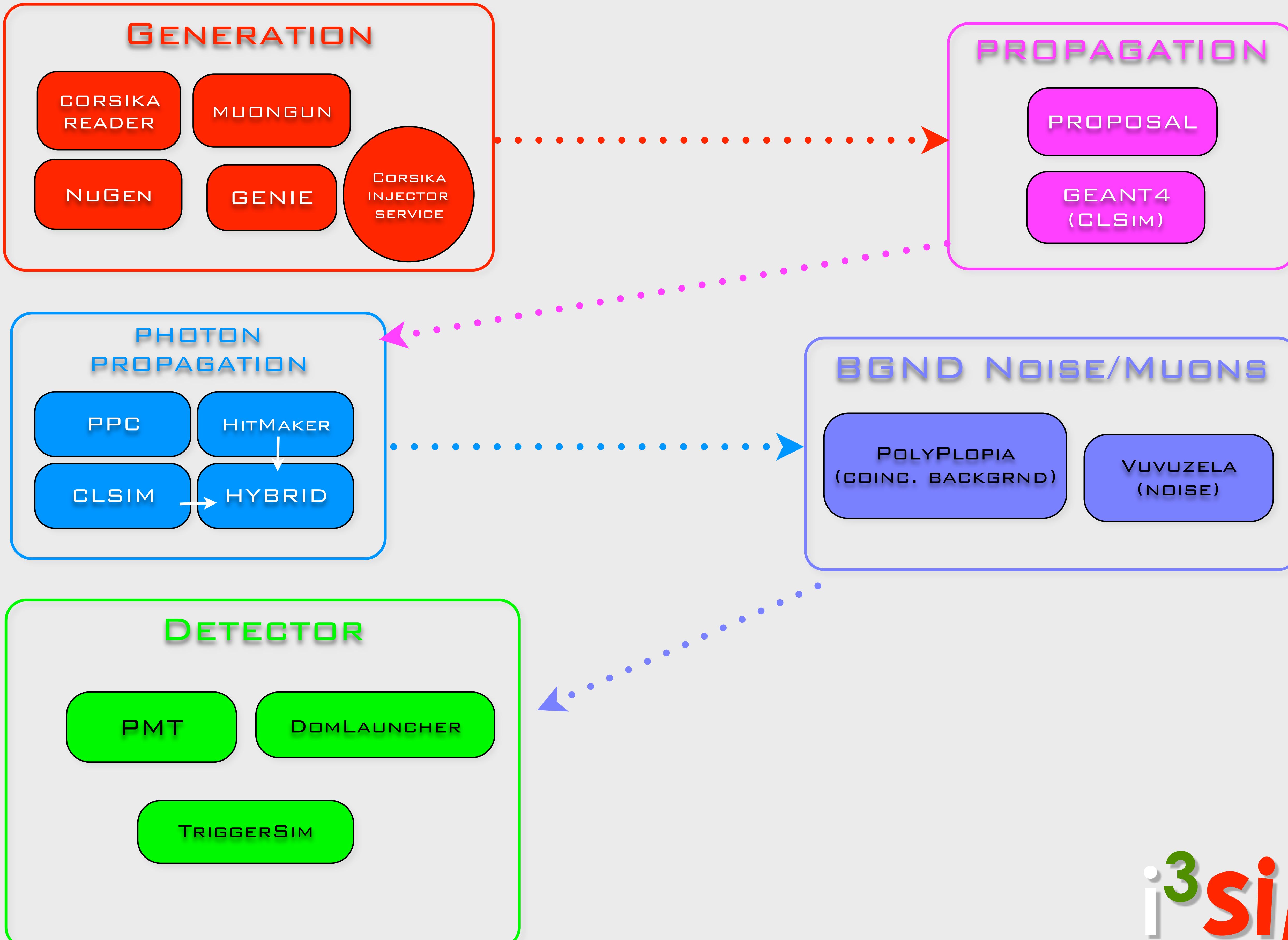
Simulation	Reconstruction
tree<I3Particle> (direction, position, energy, type)	I3Particle (direction, position, energy)
OM, vector<MCPE>	*NPEs*
OM, vector<MCPulse>	OM, vector<I3RecoPulse>
OM, vector<DOMLaunches> (digitized, PTM waveforms)	OM, vector<DOMLaunches> (digitized, PTM waveforms)

Events in icecube

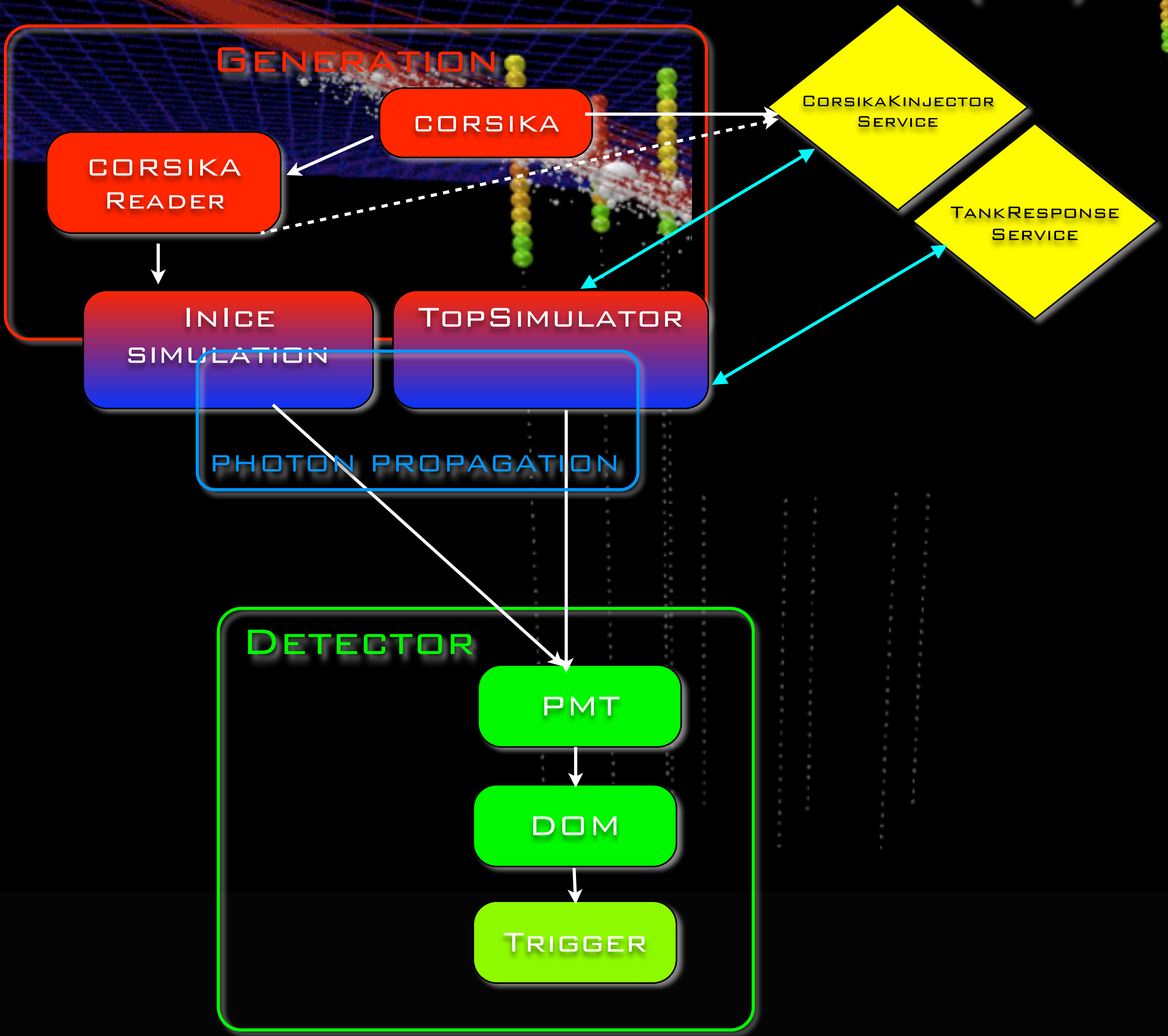
- Air shower detection @ surface
- Penetrating muon detection in deep ice
- Events dominated by cosmic ray muons : 10^6 μ for every ν that interacts in IceCube
- Upgoing ν's (atmospheric)



downgoing ν's


Generation
CORSIKA READER
MUONGUN
NuGen
GENIE
Corsika injector service
PROPAGATION
PROPOSAL
GEANT4 (CLSim)
PHOTON PROPAGATION
PPC
HitMaker
CLSIM
HYBRID
BGND Noise/Muons
PolyPlopia (coinc. backgrnd)
Vuvuzela (noise)
Detector
PMT
DomLauncher
TriggerSim

simulaton chain (IT)



Generators

- Cosmic-ray Air Showers:
  - **CORSIKA** (FORTRAN stand-alone)
  - **corsika-reader**: IceTray reader for standard format
  - **CorsikaInjectorService** (IceTop)
- Muons:
  - **MuonGun**: parametrization of flux of atm. muons under the ice.
- Neutrinos:
  - **neutrino-generator**: injects neutrinos, propagates them through Earth, forces interaction in detector volume.
  - **genie-icetray**: detailed simulation of neutrino interactions with GENIE. (Used for low-energy simulations)
  - **LeptonInjector / NuFSGen** (not yet available): weighted leptons +weights to account for flux models, interaction models, in-earth propagation, etc.

Generators (cont.)

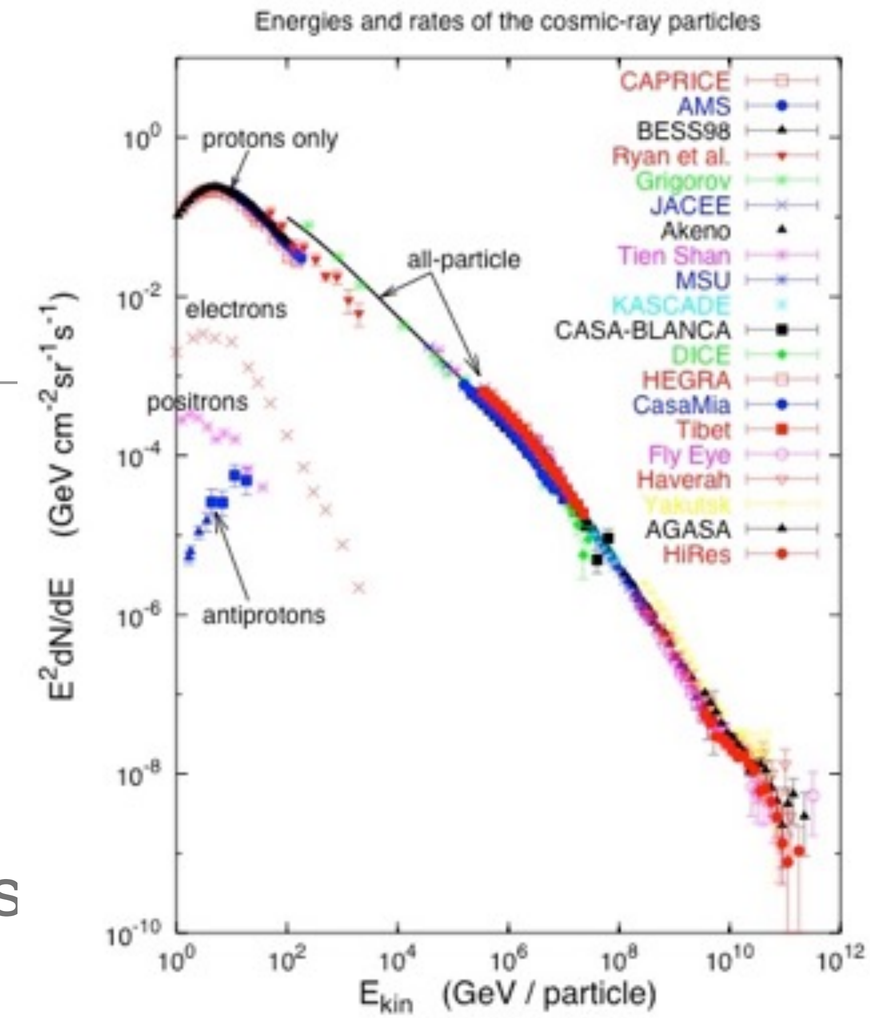
- Other:
  - **wimpsim-reader**: IceTray interface for WimpSim (FORTRAN stand-alone)

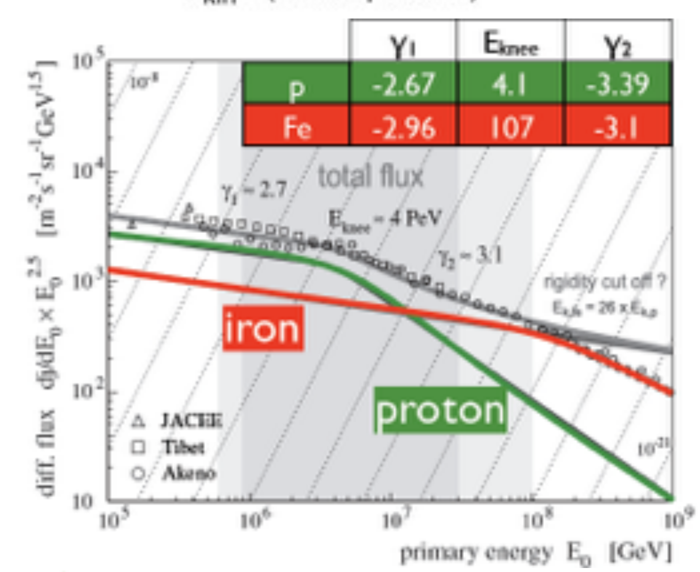
Generators : CORSIKA

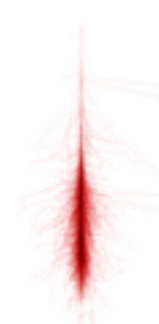
(**CO**smic **R**ay **SI**mulations for **KA**scade)

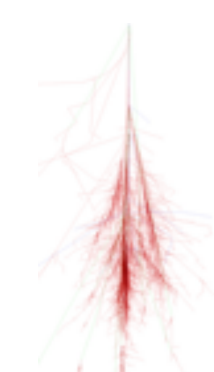
- FORTRAN code developed for KASCADE
  - Detailed simulation of extensive air showers initiated by high energy cosmic ray particles including protons, light nuclei up to iron, photons and many other particles.

- weighted events : artificially flat spectrum
  - better livetime efficiency @ 10 TeV but poor efficiency @ TeV
  - energy-targeted generation of (H,He,CNO,Mg,Fe) with $E^{-1(2)}$

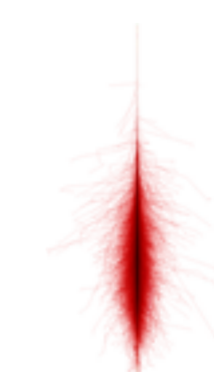





Photon shower


Proton shower


Iron shower

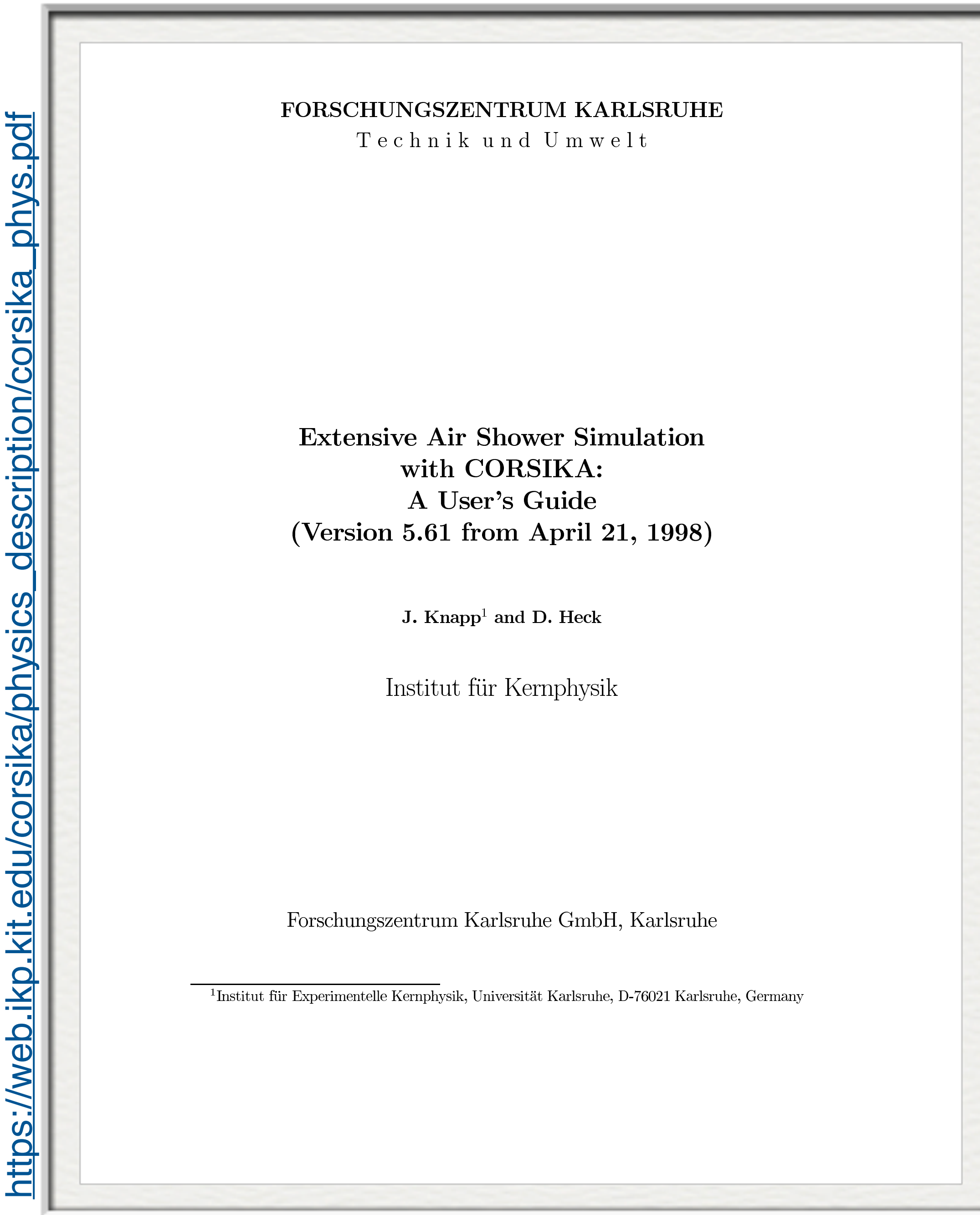

Muon shower

Generators : CORSIKA

(**CO**smic **R**ay **SI**mulations for **KA**scade)

- Particles are tracked through the atmosphere until they undergo reactions with the air nuclei or - in the case of instable secondaries - decay.
- The hadronic interactions at high energies may be described by several reaction models alternatively:
  - *VENUS*, *QGSJET*, and *DPMJET* (Gribov-Regge theory),
  - SIBYLL (minijet model).
  - *neXus*, *EPOS* (combination of *QGSJET* and *VENUS*).
  - *HDPM* (Dual Parton Model).
- Hadronic interactions at lower energies:
  - *GHEISHA*, *FLUKA* , or *UrQMD* models.
- For electromagnetic interactions
  - Tailored version of *EGS4*.
  - Analytical *NKG* formulas.

https://web.ikp.kit.edu/corsika/physics_description/corsika_phys.pdf


FORSCHUNGSZENTRUM KARLSRUHE
Technik und Umwelt

Extensive Air Shower Simulation
with CORSIKA:
A User's Guide
(Version 5.61 from April 21, 1998)

J. Knapp[1] and D. Heck

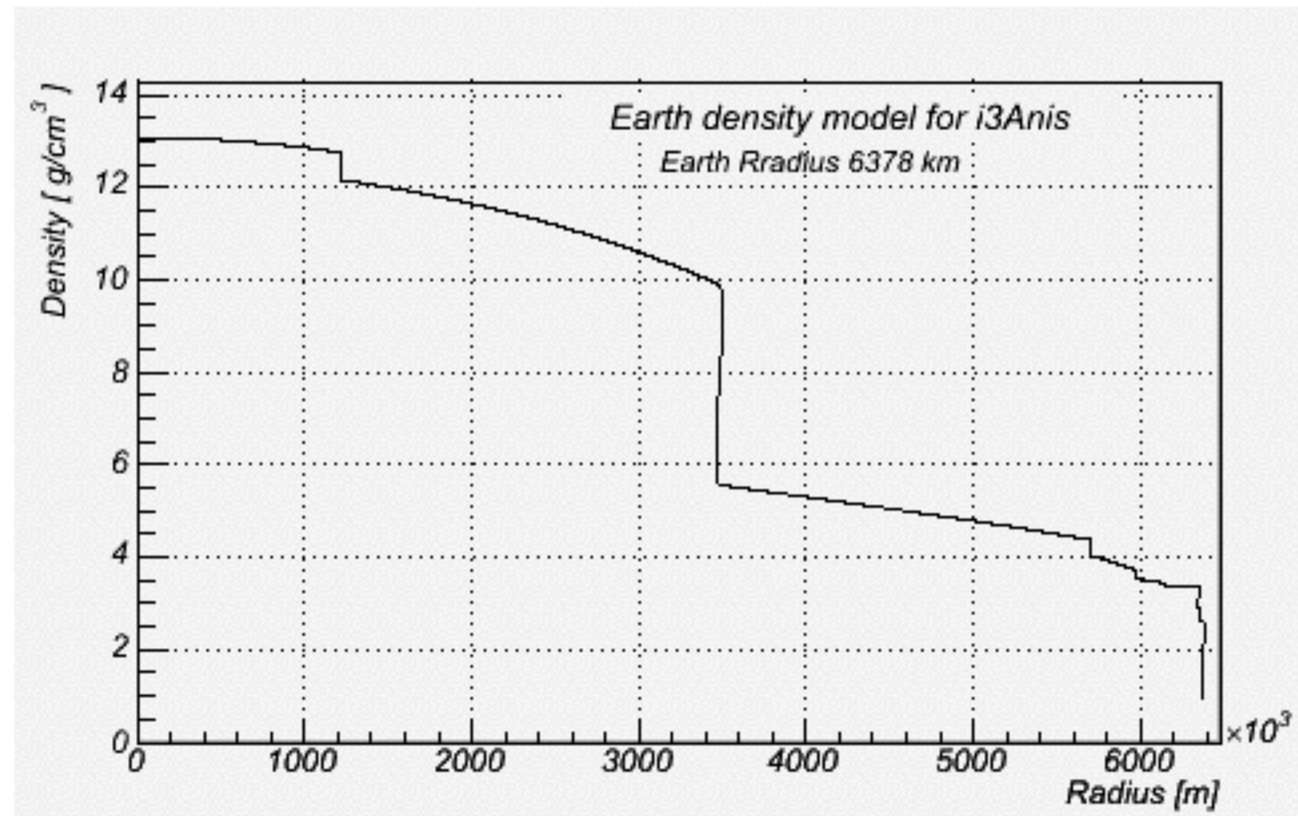
Institut für Kernphysik

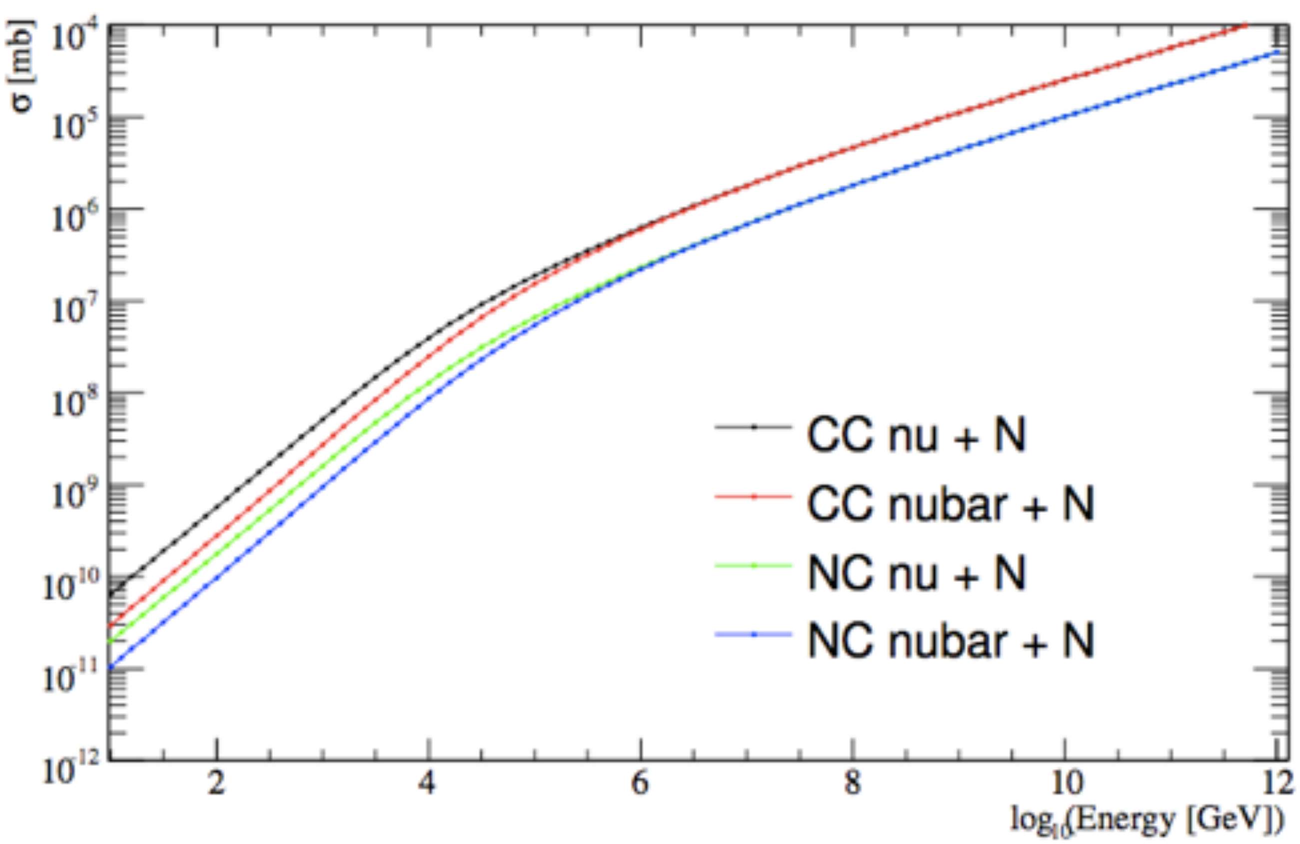
Forschungszentrum Karlsruhe GmbH, Karlsruhe

[1] Institut für Experimentelle Kernphysik, Universität Karlsruhe, D-76021 Karlsruhe, Germany

Generators : neutrino-generator

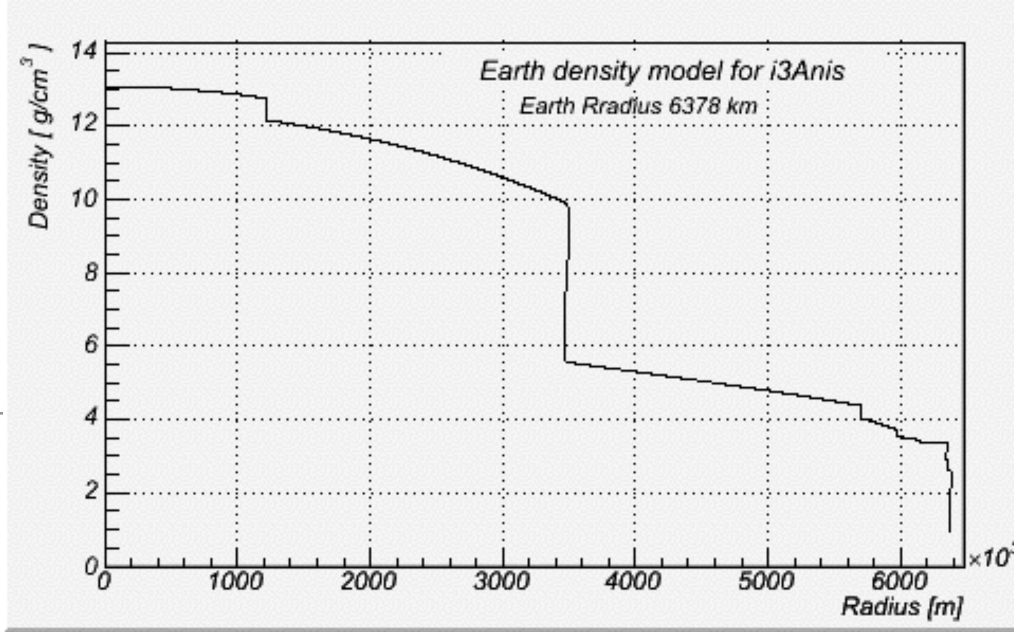
- produce a $E^{-\gamma}$ ν_μ, ν_e, ν_τ with
  - PRELIM Earth's density model





- parton distribution functions
- prop & interaction of neutrinos into a weight : flexible spectral weight

Generators : neutrino-generator



Propagation

neutrino on the earth surface and propagate tawards detector until "DistanceExit" (after which charged lepton created are considered invisible)

If initial primary interacted, secondary particles are created. Then we propagate secondaries:

- 1) if neutrinos, propagate further
- 2) if tau, decay
- 3) if charged lepton created inside detection volume --- make it the final inice particle and find its parent neutrino and make it the Primary particle.
- 4) if charged lepton created outside detection volume, see if propagates to inside volume

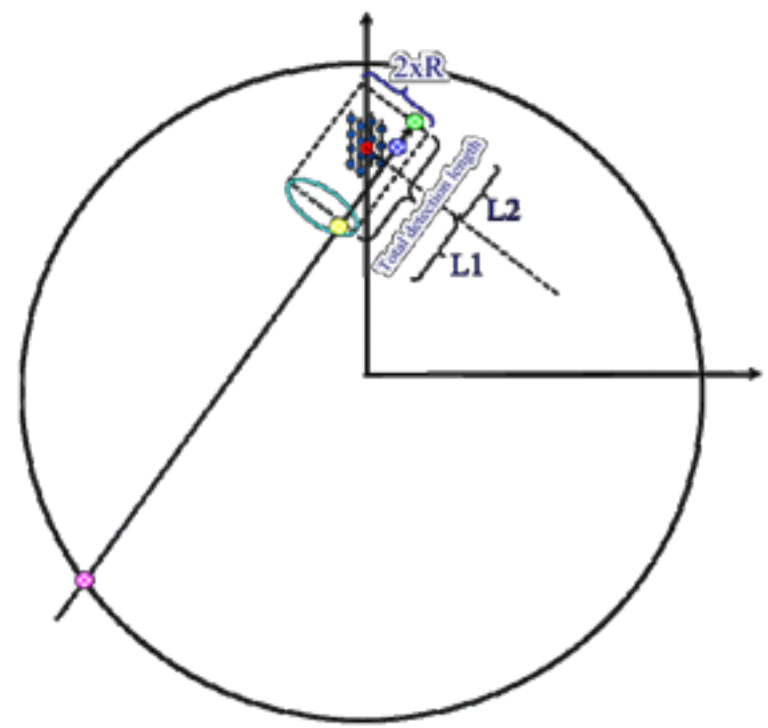
This loops over until every particles reach to the end of volume.

While PropagateParticle is looping over all the particles and neutrino might interact with matter to create secondary neutrino/charged lepton/hadrons and those particles are added to the end of vector of propagated particles to be further searched.

Generators : neutrino-generator

- Calculates the propagation probability (i.e. that the neutrino will reach the detector)

- It forces an interaction within a volume around the detector and computes the probability of this interaction



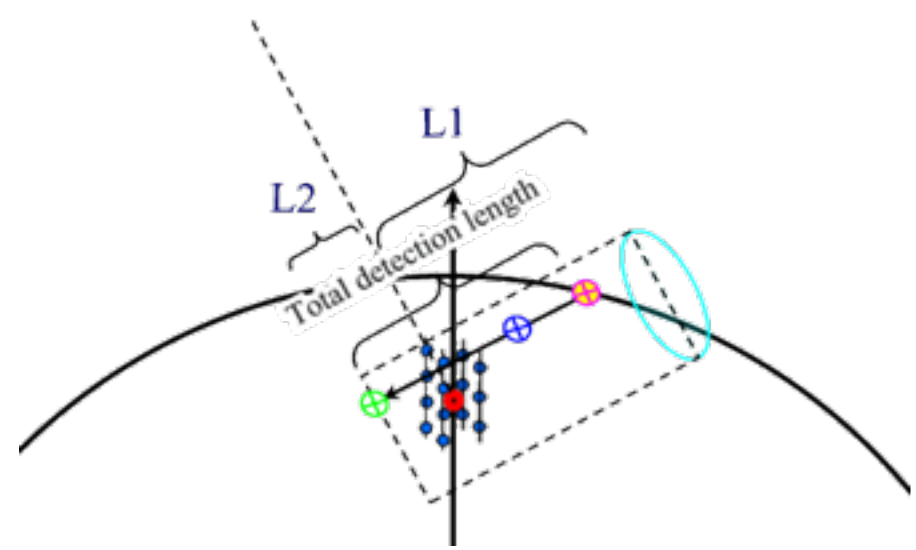
$$OneWeight = \left(\frac{P_{int}}{E^{-\gamma}}\right) \cdot \int_{E_{min}}^{E_{max}} E^{-\gamma} dE \cdot Area \cdot \Omega \cdot T [GeV \cdot cm^2 \cdot sec \cdot sr]$$

where $Pint = TotalInteractionProbabilityWeight$, $E^{-\gamma}$ is the neutrino generation energy spectrum shape, $Emin$ and $Emax$ is the minimum and maximum generation energy of neutrinos, $Area$ is the generation surface, Ω the generation solid angle and $T = 1sec$ is the timescale.

- The weight corresponding to a given theoretically motivated neutrino flux is

$$w_i = \frac{OneWeight_i}{NEvents} \times \frac{d\Phi_\nu(E_\nu)}{dE_\nu}$$

- For more details on how to use OneWeight see:



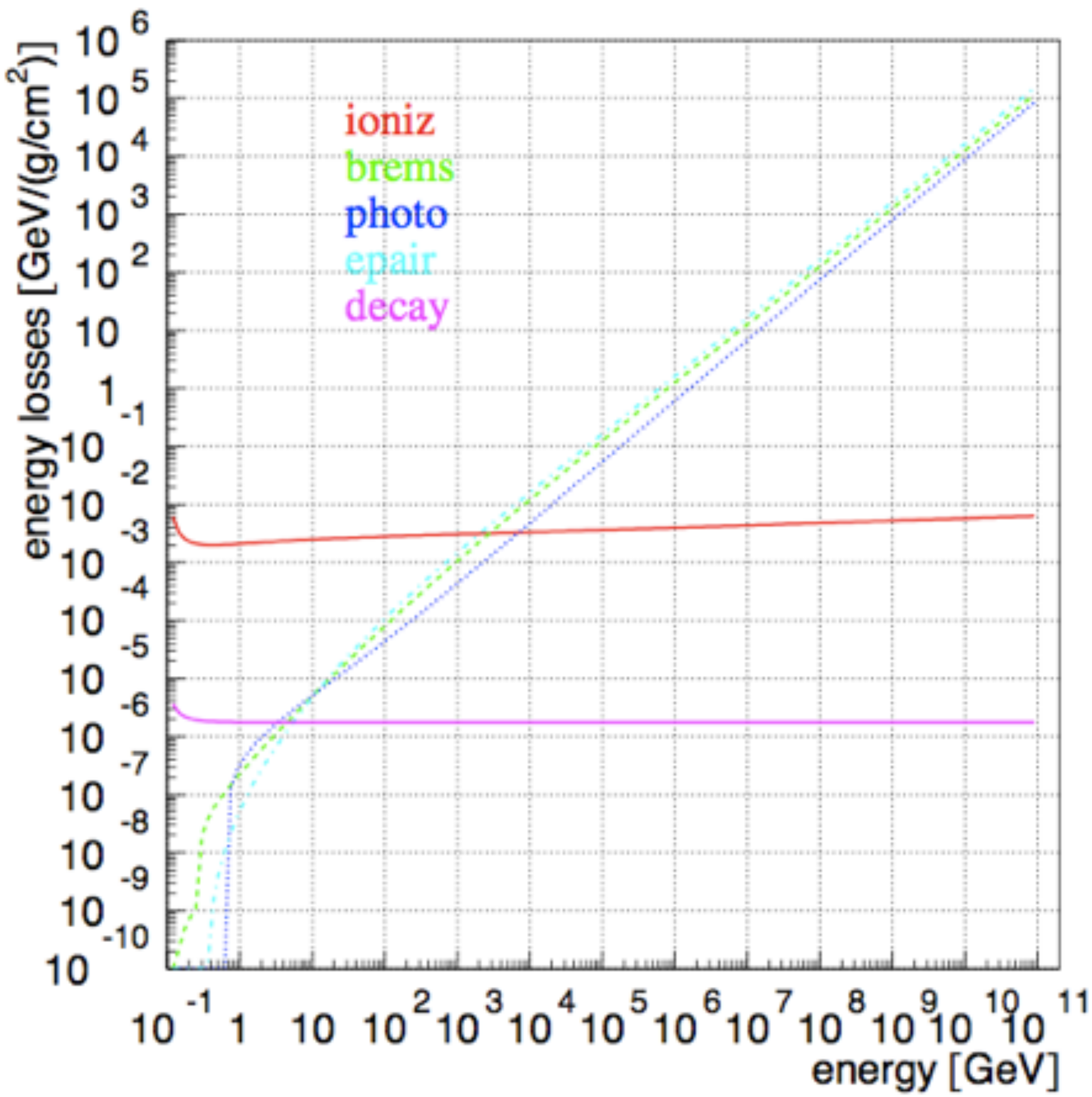
https://docushare.icecube.wisc.edu/dsweb/Get/Document-44937/OneWeight.pdf

Lepton propagation

- PROPOSAL: parametrized interactions with the medium.
  - Stochastic energy losses include:
    - Ionization
    - electron-Pairproduction
    - Bremsstrahlung
    - Photonuclear Interaction
    - “Decay”
- GEANT4: Detailed particle propagation in media.
  - 3rd-party G4 library used by CLSim to propagate leptons for low-energy simulations (CPU-intensive).

propagator :PROPOSAL

propagates μ, e, τ & monopoles



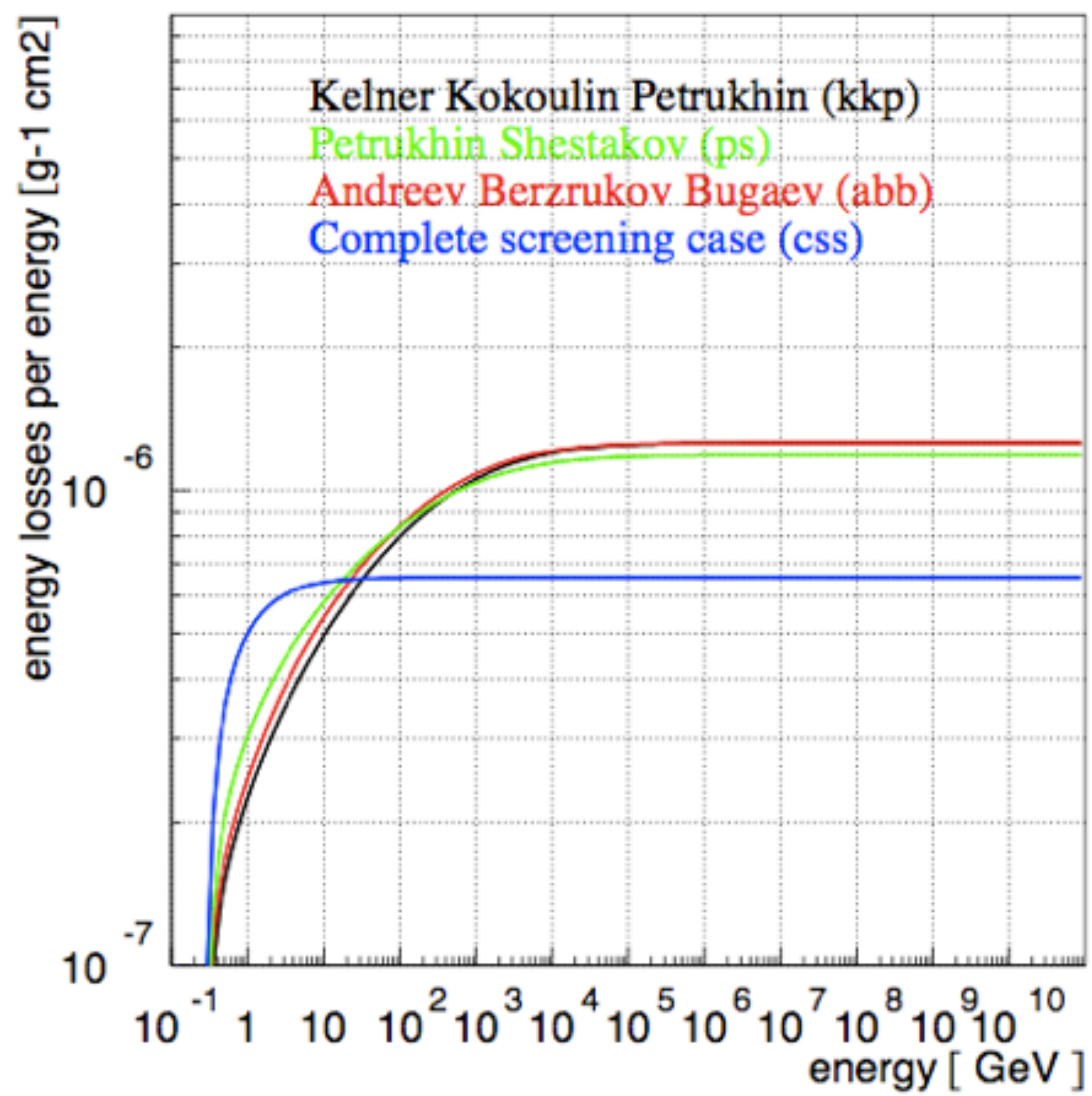
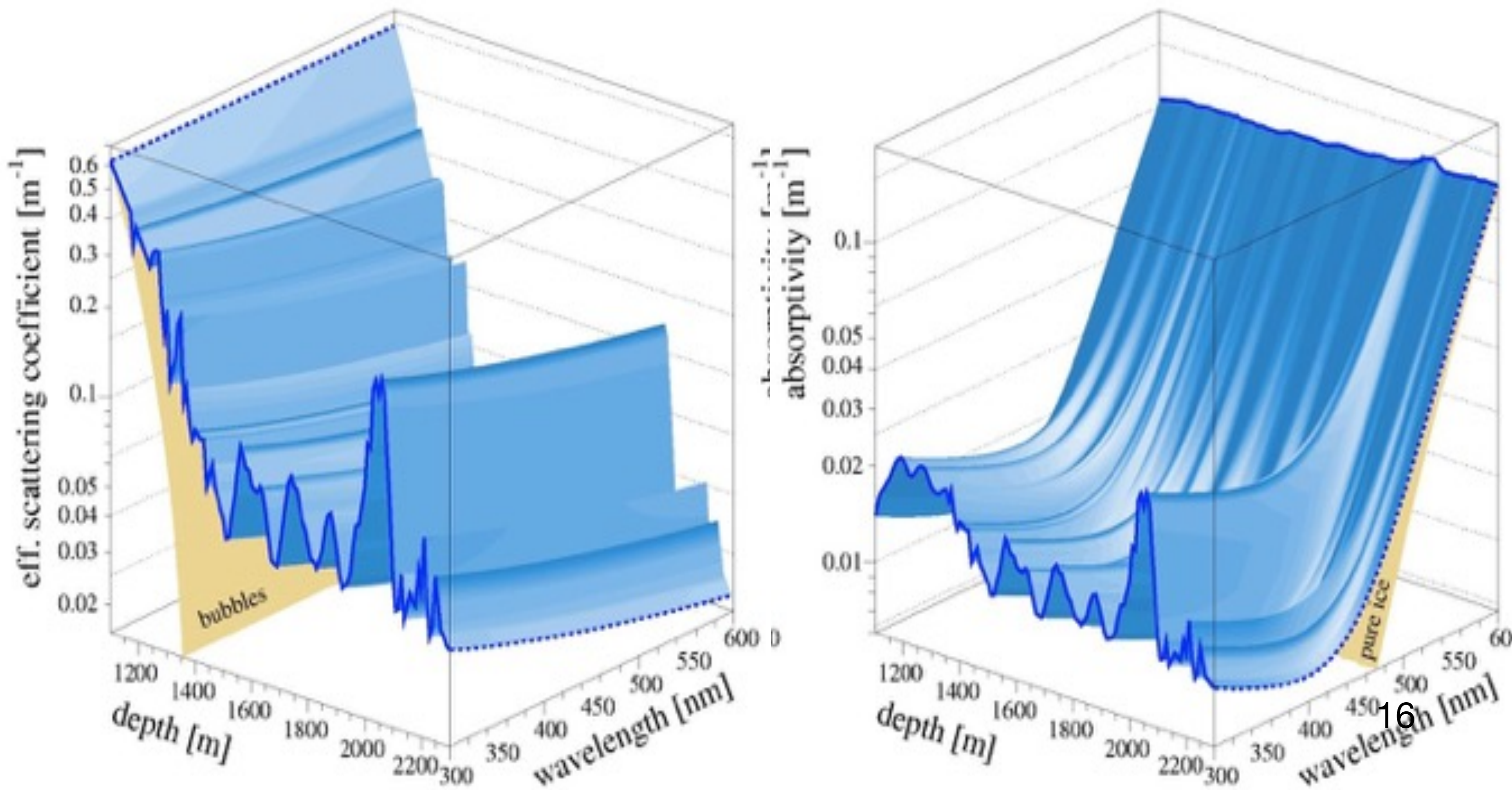


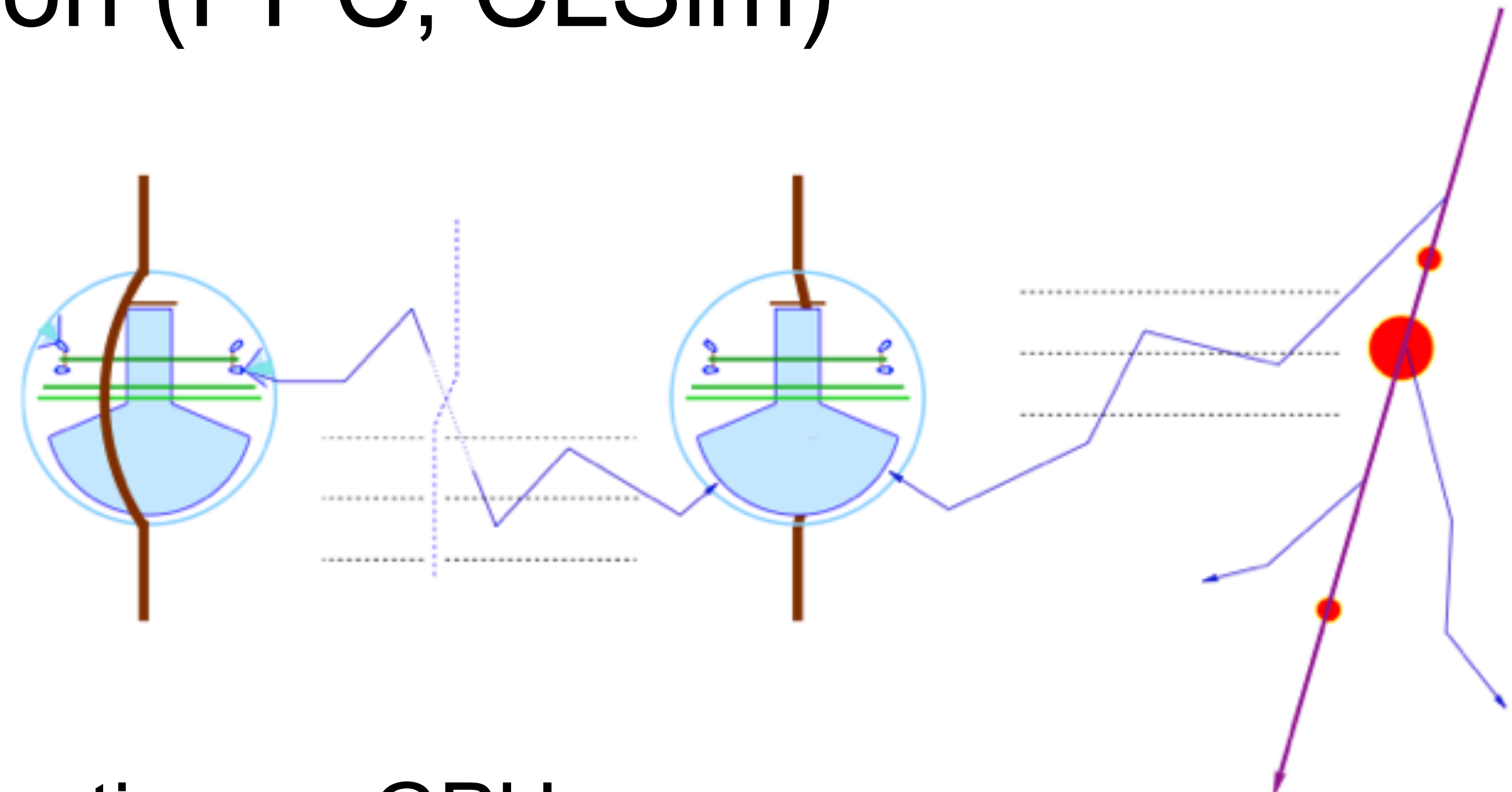
Figure 35: Bremsstrahlung cross section parameterizations for muons

Photon Propagation

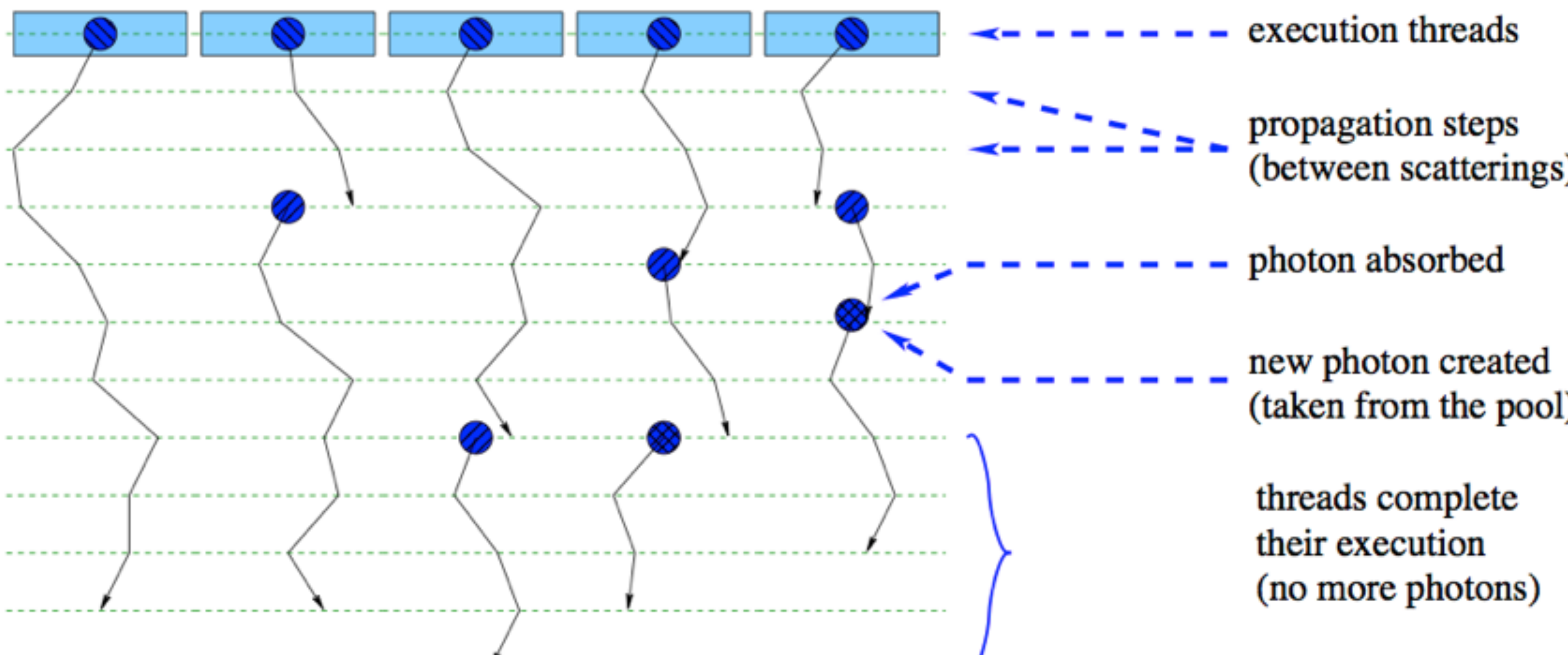
- μ energy lost + cascades ➙ photons ➙ p.e.
  - Photon propagation : ice properties + PMT response + DOM glass/gel
    - Pre-generated lookup splined table :
      - I3PhotonicsHitMaker
      - Amplitude and time distribution
    - Direct photon tracking
      - PPC
      - CLSim
    - Hybrid photon tracking
      - HitMaker + CLSim



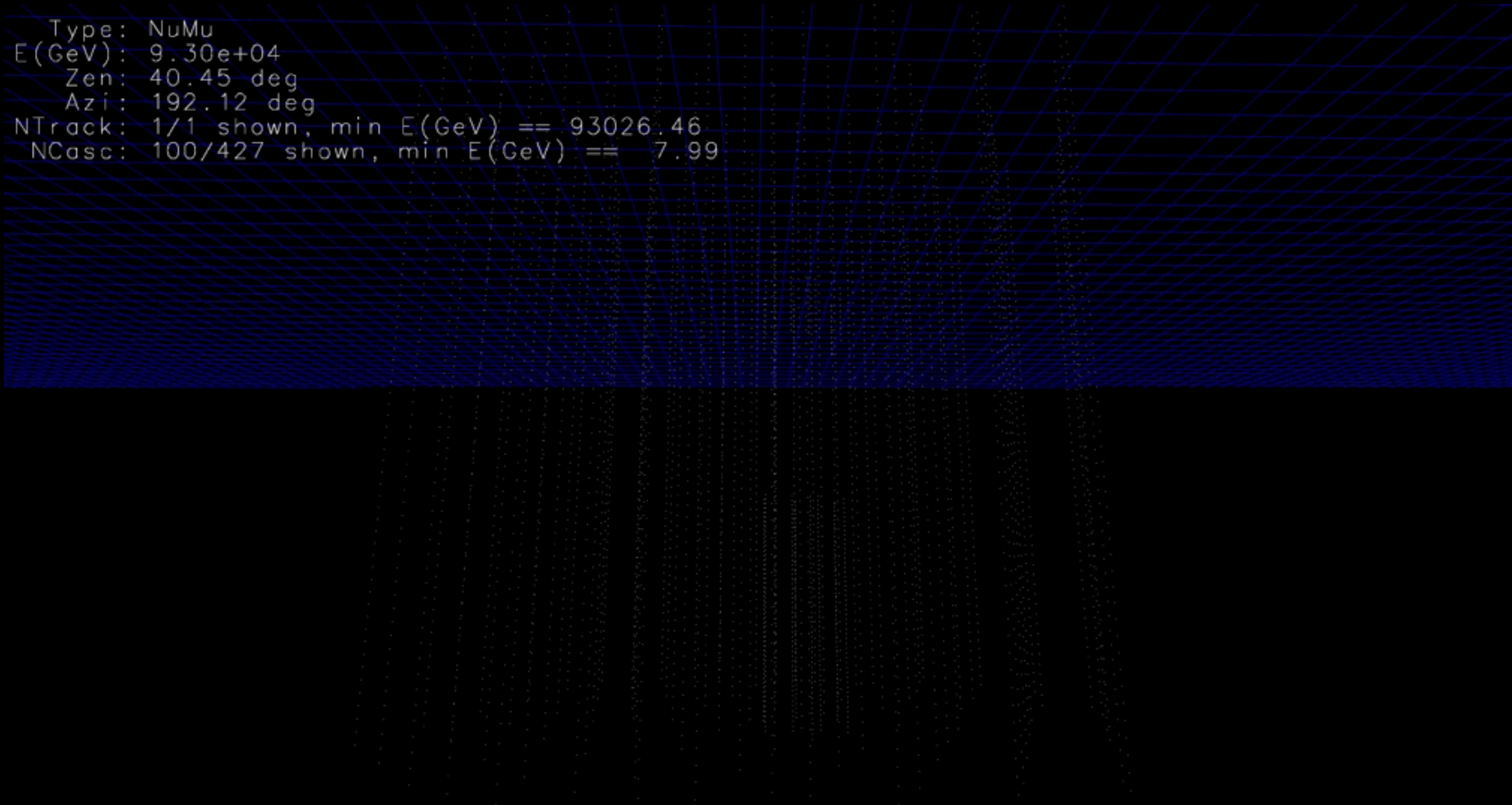
Photon Propagation (PPC, CLSim)



Direct photon propagation on GPU



Photon Propagation (PPC, CLSim)

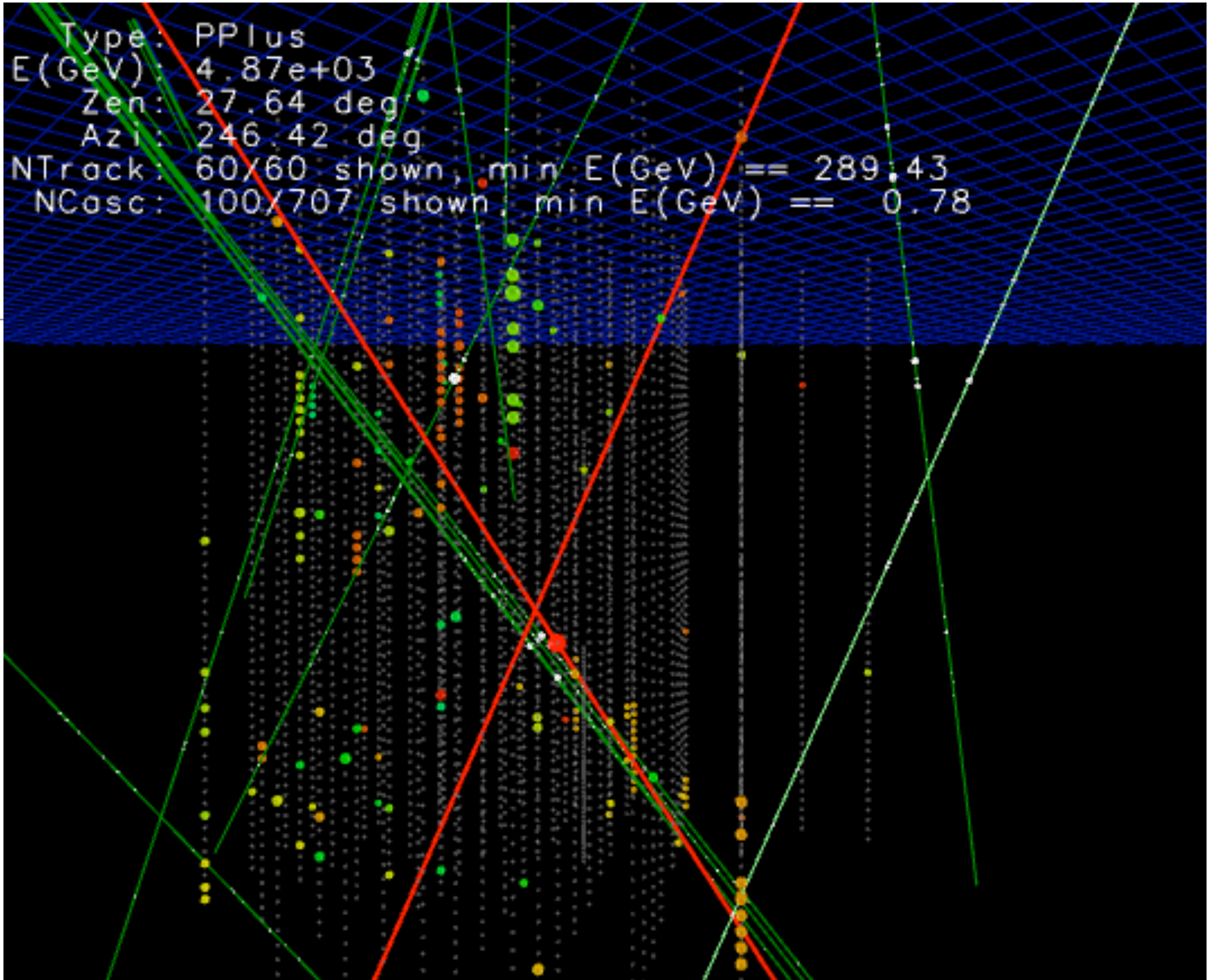


http://icecube.wisc.edu/~ckopper/muon_with_photons.mov

Diplopia

(from gr. διπλόος, "double", and ὄψ, ὀπός, "vision")

Coincident atmospheric shower events in IceCube



Diplopia

(from gr. διπλόος, "double", and ὄψ, ὀπός, "vision")

Coincident atmospheric shower events in IceCube

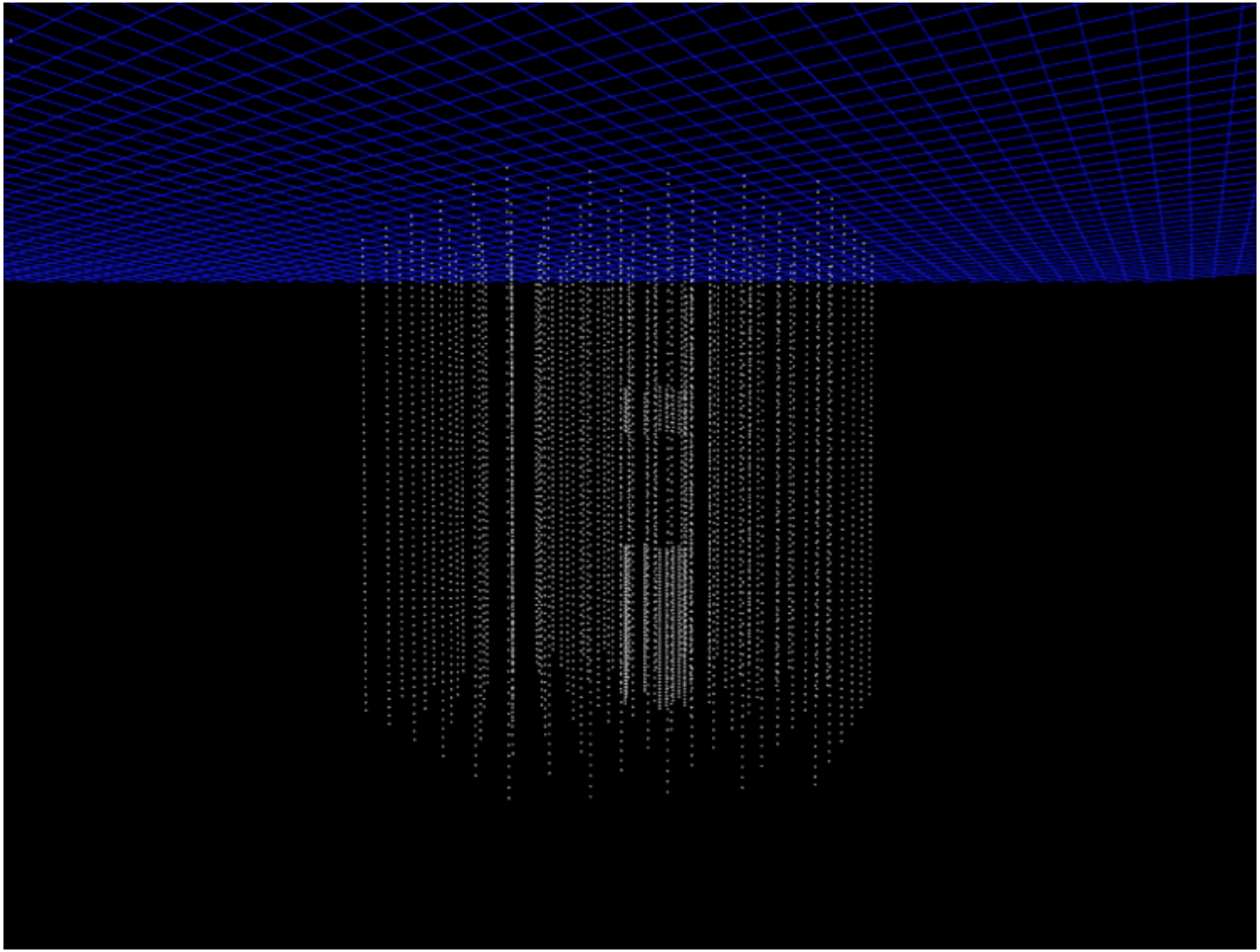
- **diplopia::PoissonMerger**
  - Injects background event read from a separate file on top of primary events in the chain by sampling from a Poisson distribution over a time window Δt.
  - Also makes use of a *CoincidentEventService* that could be drop-in replaced with other event services such as a MuonGun-based service.
  - Writes a separate I3MCTree with background particles.
  - Writes a combined I3MCPE map for signal and background.

- **diplopia::MPHitFilter**
  - Removes events that don't produce light in the detector and removes branches of I3MCTrees whose particles don't produce enough PEs in the detector,
  - Reduces the storage requirements.

- It is then up to CoincidenceAfterProcessing from trigger-sim to split up and clean up events based on triggers.

Diplopia

(from gr. διπλόος, "double", and ὄψ, ὀπός, "vision")

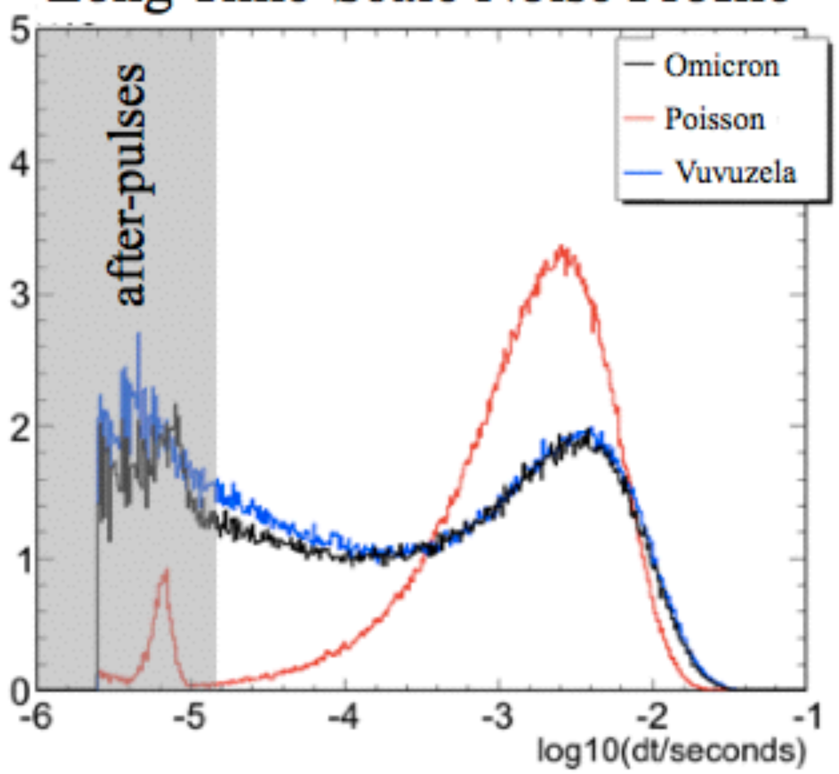


http://icecube.wisc.edu/~juancarlos/video/10msec_w_noise.mp4

Noise Generation

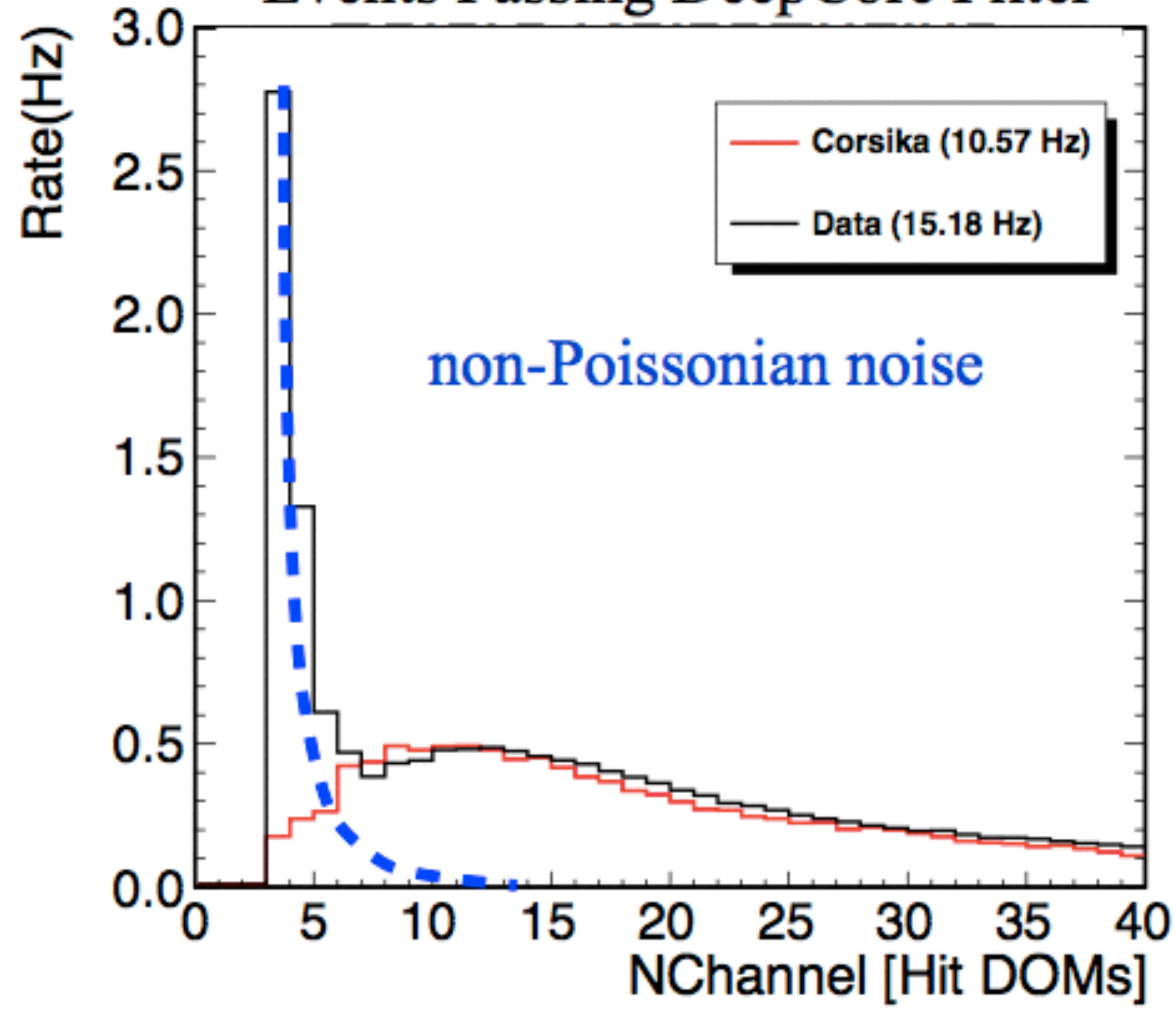
Previous simulation used simplified Poissonian model. Vuvuzela uses exponential for **thermal** and **radioactive decays** and log-normal for **scintillation**.

Long Time-Scale Noise Profile



*Courtesy of M.Larson (U.Alabama)

Events Passing DeepCore Filter



*Courtesy of J.Koskinen (PSU)

Noise Generation

—> (MCPEs)

Noise Model

Thermal Noise (~few Hz)
[Poisson process]

~ ms Timescales

+

Radioactive Decay in Glass
[Poisson process]

~ ms Timescales

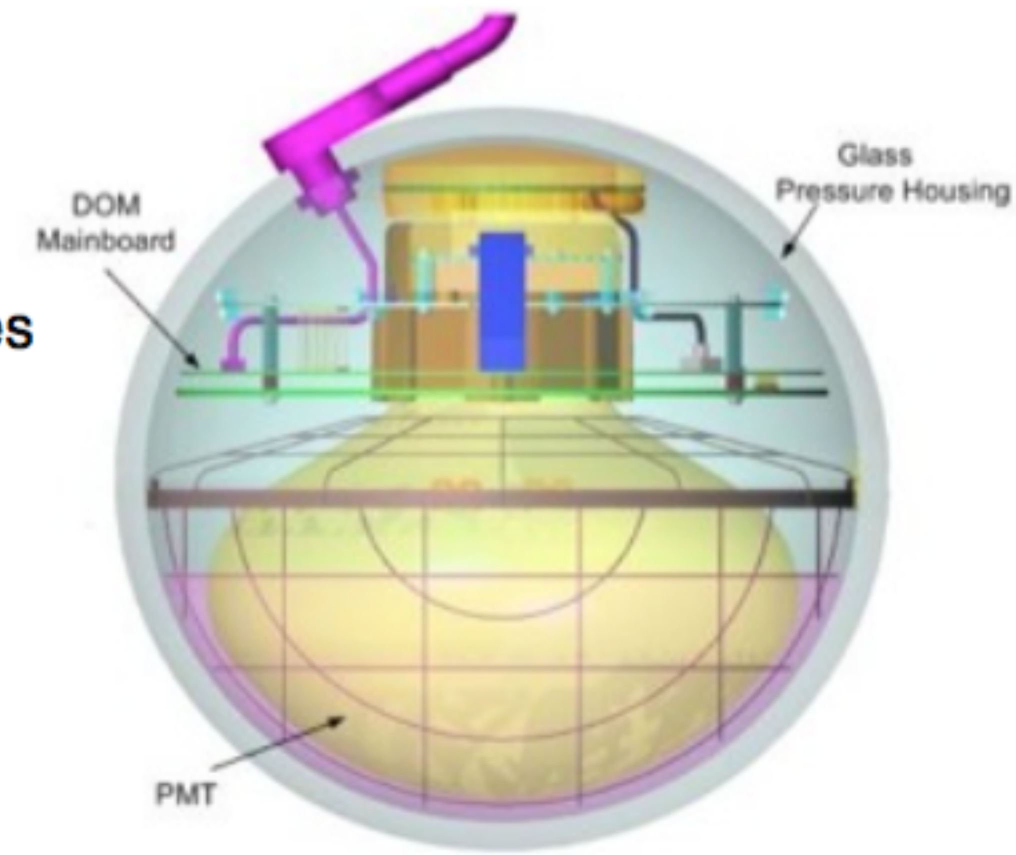
↓

Energy deposited in glass

↓

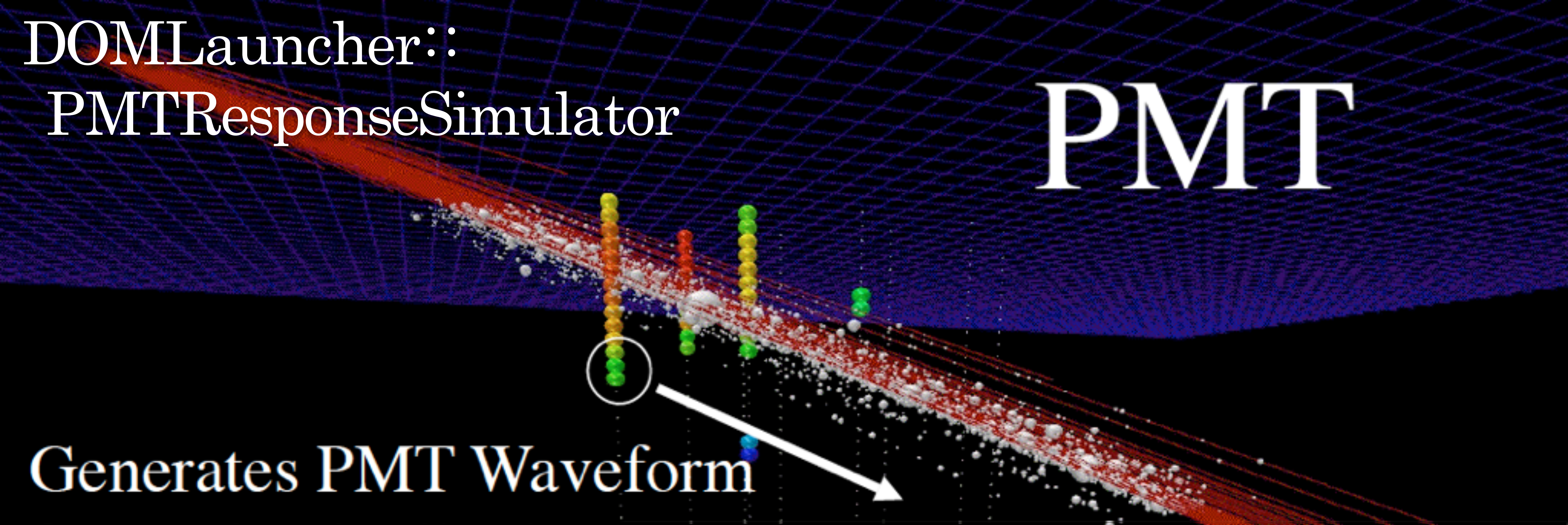
Glass scintillates/fluoresces
over long timescale
[Log-normal]

≲ 500 µs Timescales



DOMLauncher:: PMTResponseSimulator

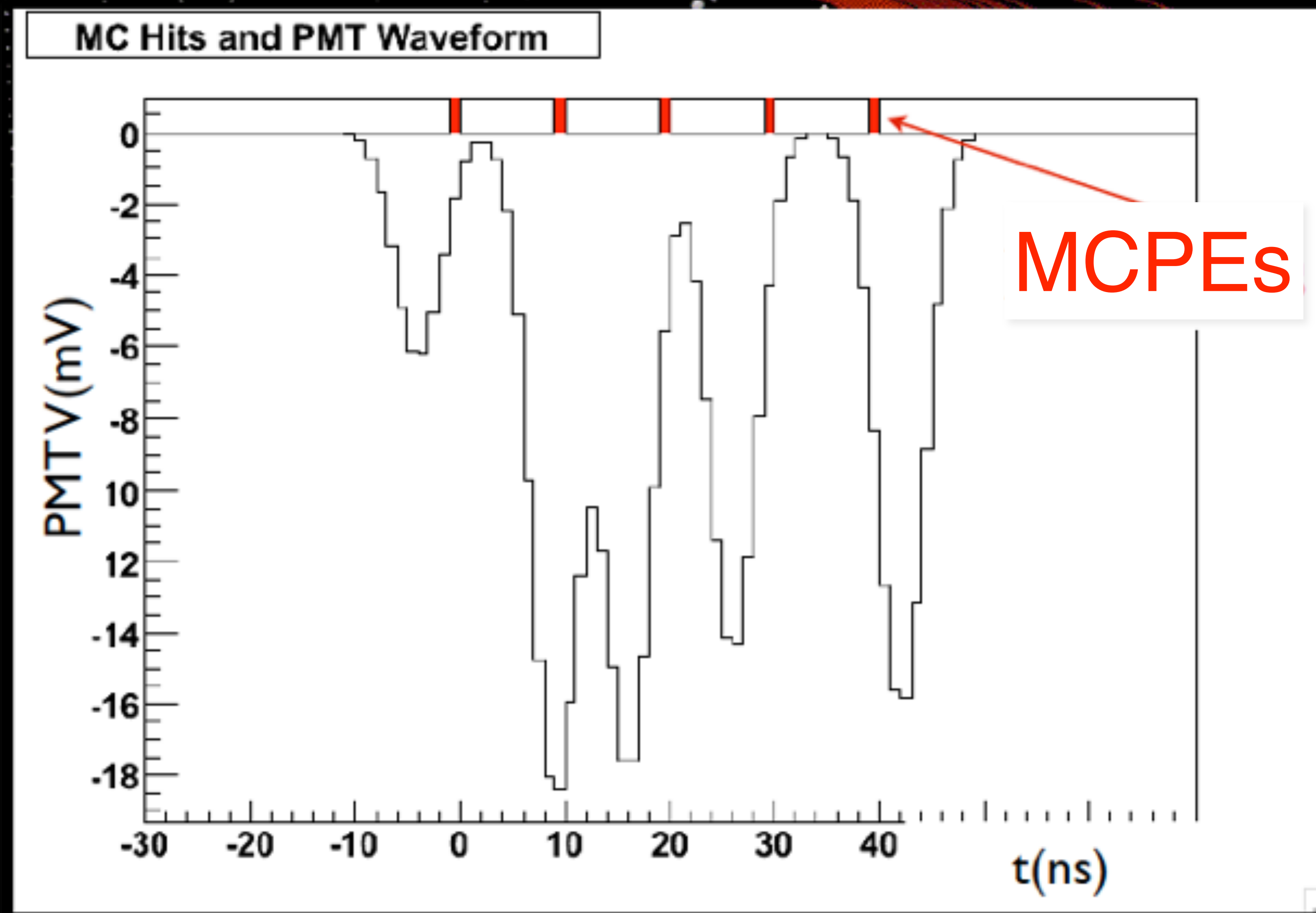
PMT



Generates PMT Waveform

From distribution of (combined) MCPEs.

Outputs I3MCPulseSeries for each DOM.

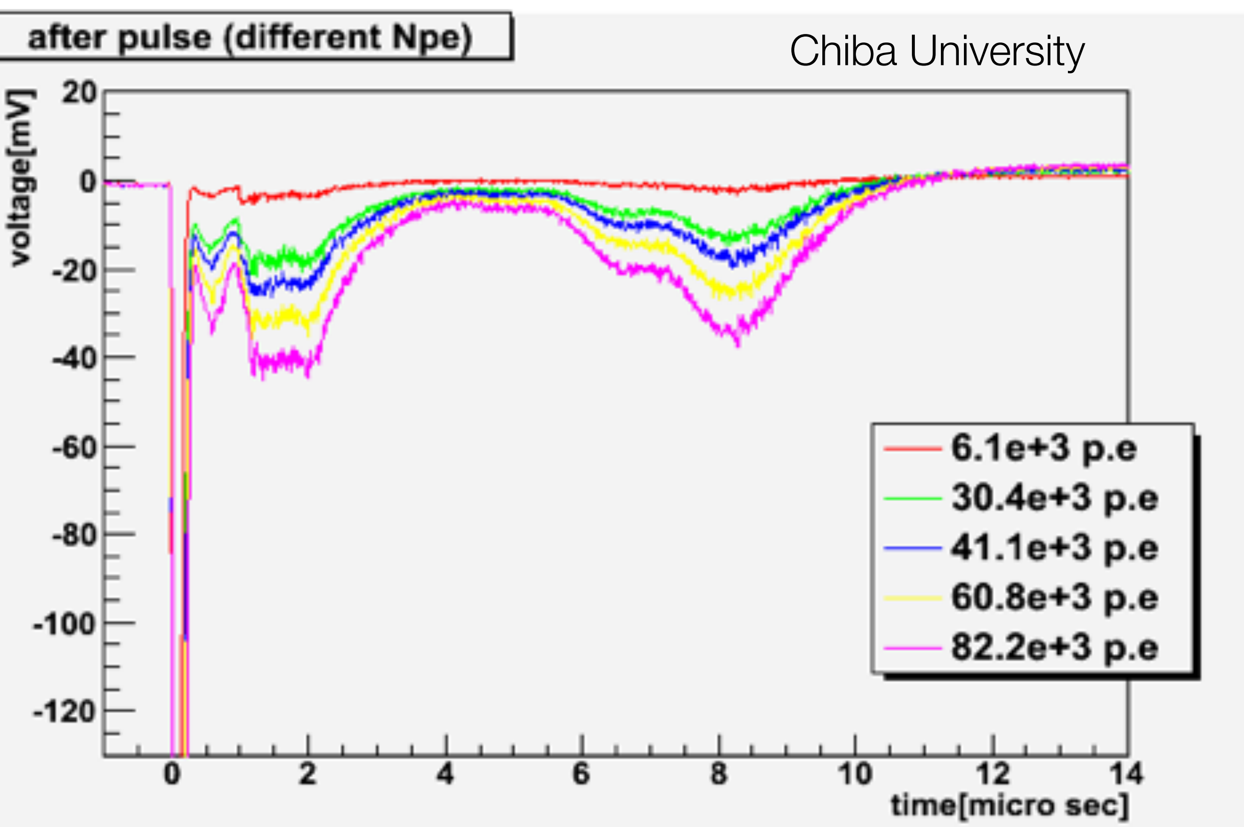


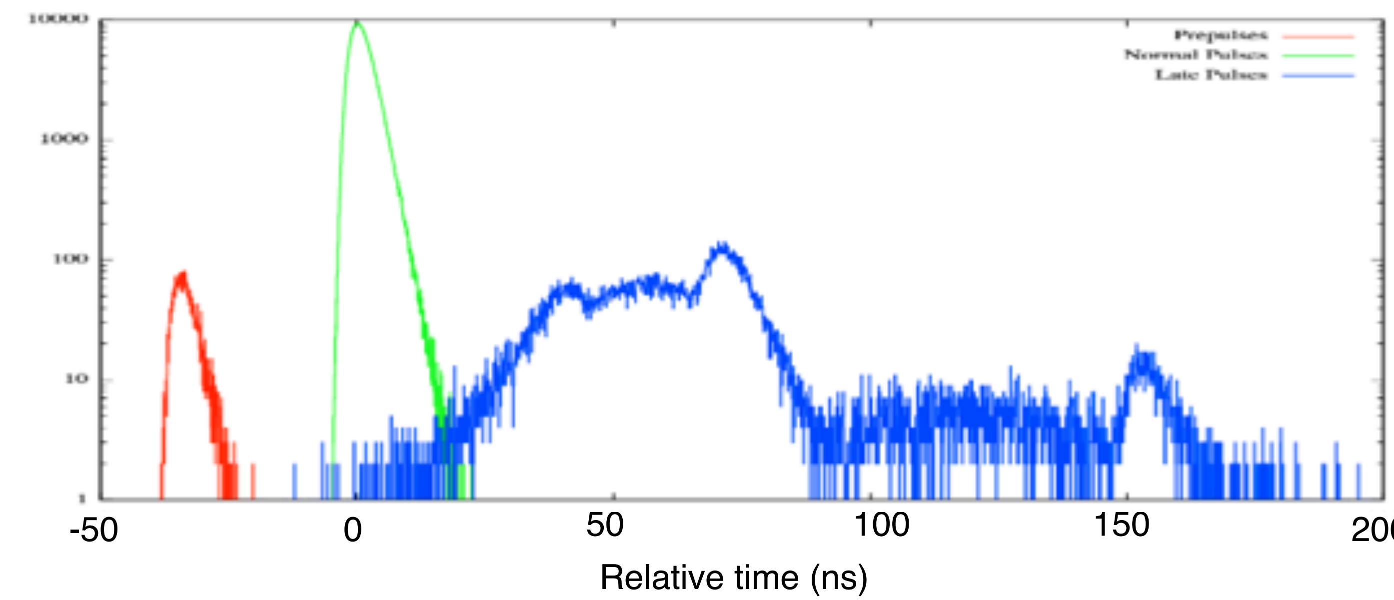
PAL pulses

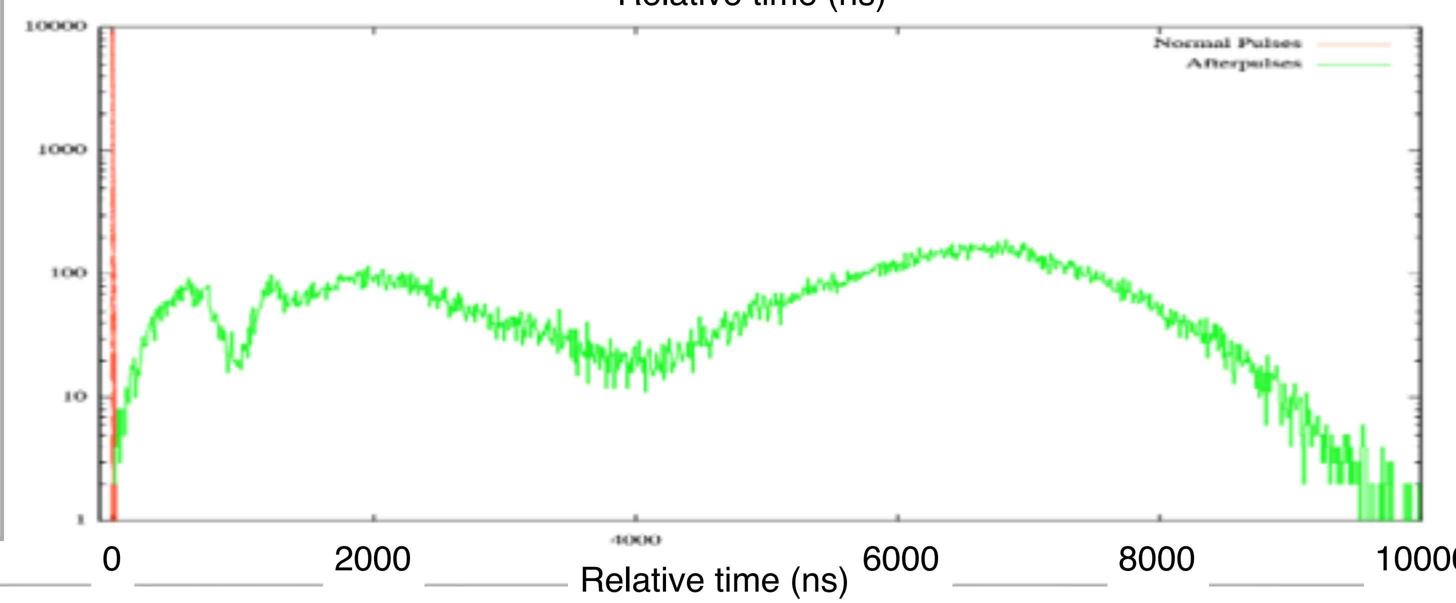
Pre-pulses: photoelectrons ejected from the first dynode,

After-pulses: ionization of residual gases by electrons accelerated in the space between dynode

Late pulses: electrons backscatter from dynode to cathode.







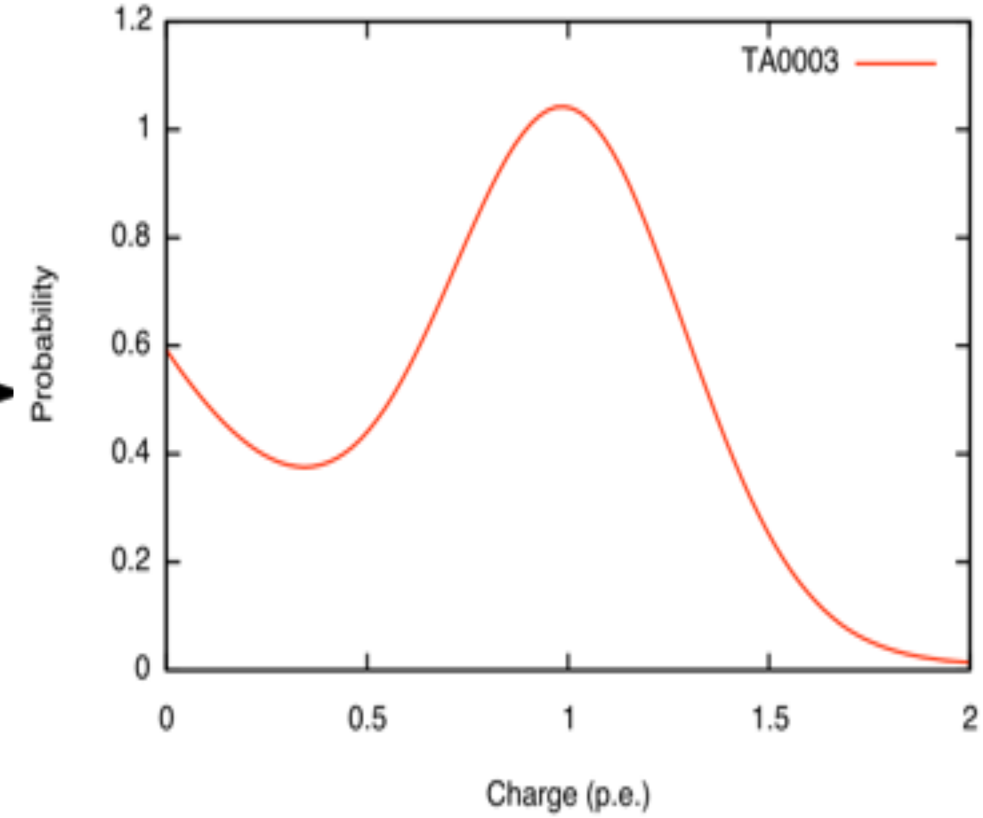
PMTResponseSimulator

In- and output:

- Takes MCPEs as input.
- Produces weighted MCHits.

Processing MCPEs :

- Give each MCPE a weight corresponding to the pulse charge that photon would yield.
- Generate prepulses, late pulses and after pulses.
- Apply time jitter.
- Simulate the effect of saturation.

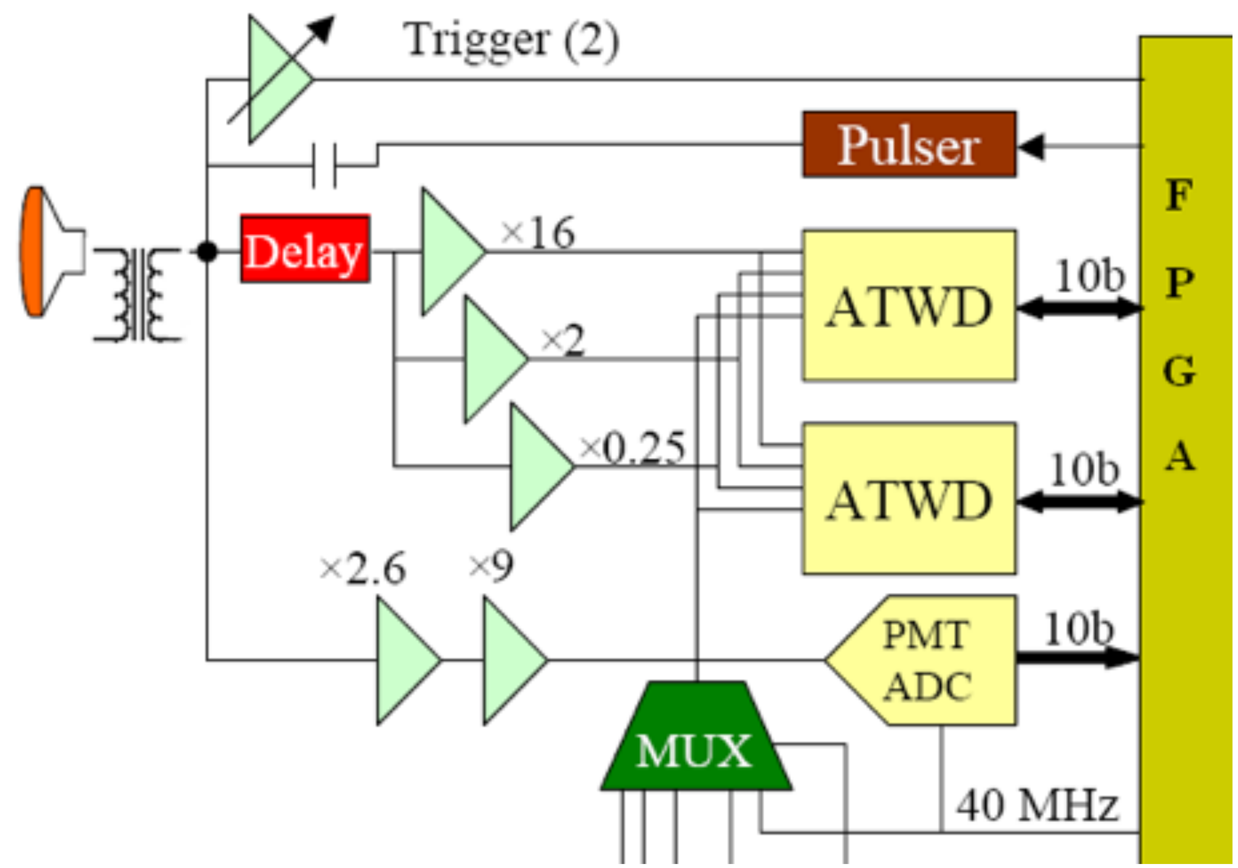


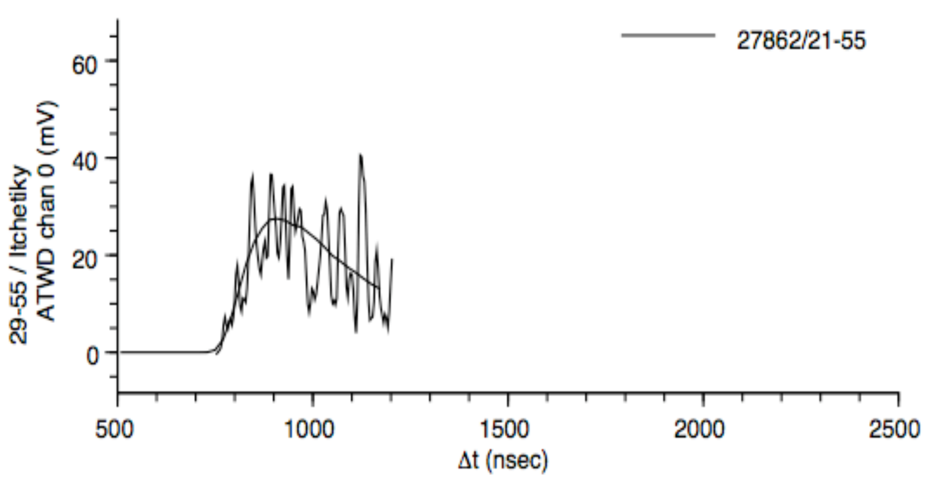
Weights fromSPE Charge Distribution

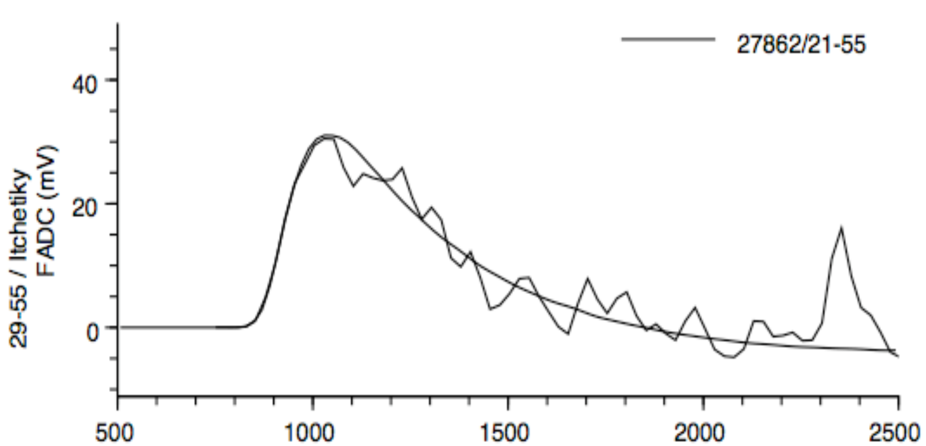
http://software.icecube.wisc.edu/documentation/projects/DOMLauncher/index.html

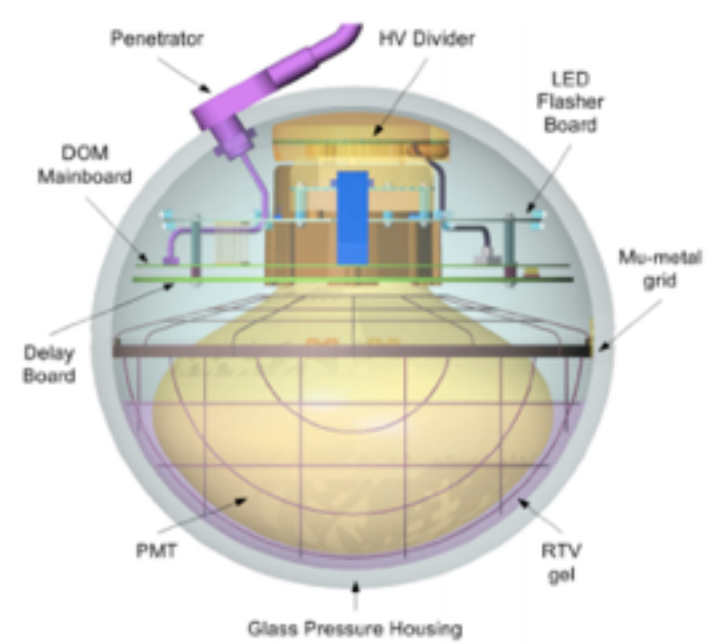
DOMLauncher: DOM mother board simulation

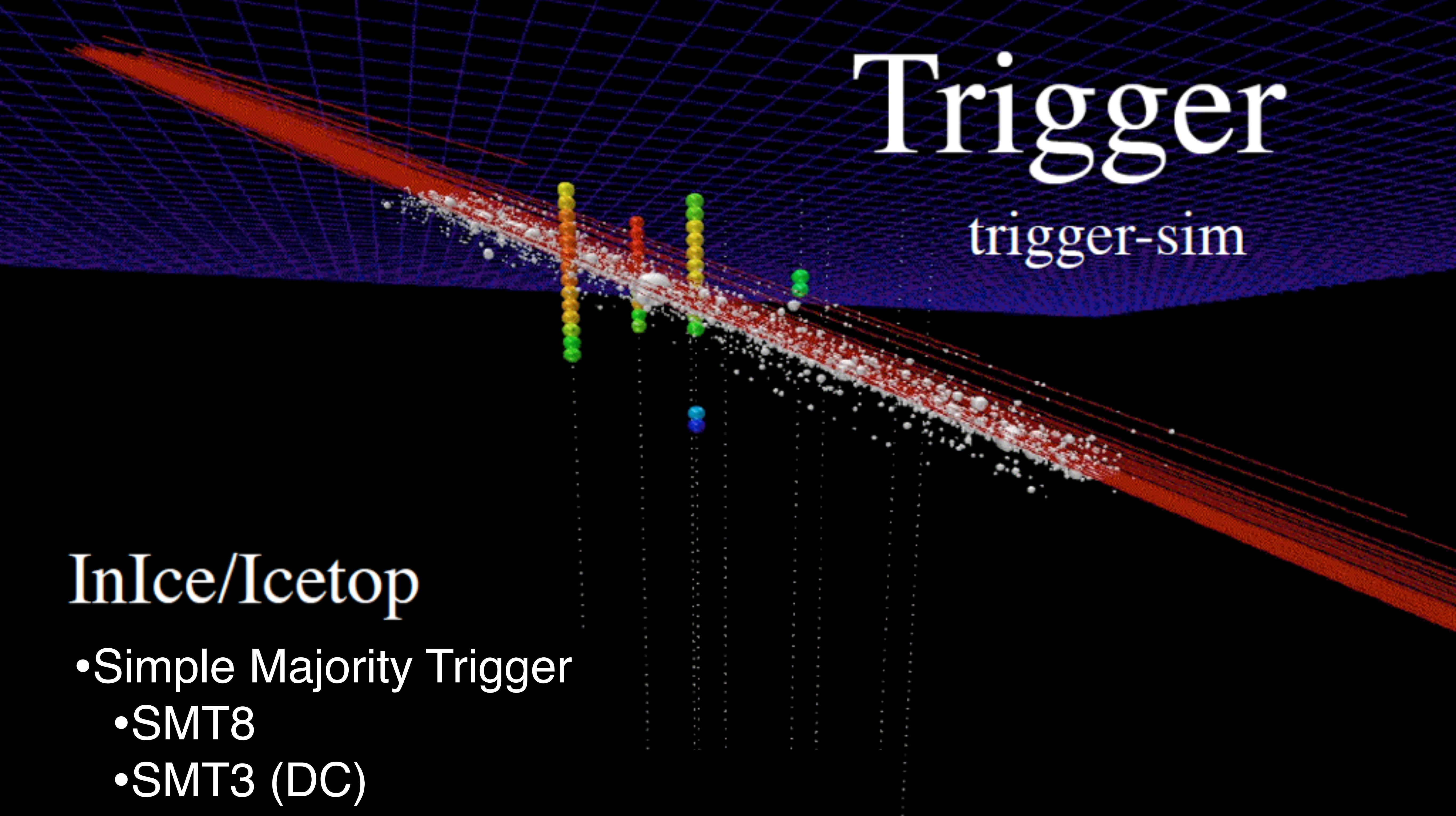
- Discriminator
- LC-logic
- Digitization
- Simulated effects
  - Electronic noise in the digitizers
  - Beacon launches (CPU triggered launches)
  - The FPGA Clock phase
  - RAPcal time uncertainty











Trigger

trigger-sim

InIce/Icetop

- Simple Majority Trigger
  - SMT8
  - SMT3 (DC)
- Cluster Trigger
- Volume Trigger
- SlowMP

Global Trigger
I3TimeShifter

Trigger Types

- **S**imple **M**ultiplicity **T**rigger (SMT)
  - *N* HLC hits or more in a time window
  - Example: InIce SMT8 with N_hits ≥ 8 in 5 μs
  - readout window around this captures early and late hits (-4 μs, +6 μs)

- **String** trigger (a.k.a. Cluster trigger in DAQ-land)
  - *N* HLC hits out of *M* DOMs on a string in a time window
  - Example: 5 hits from a run of 7 adjacent DOMs in a time window of 1500 ns

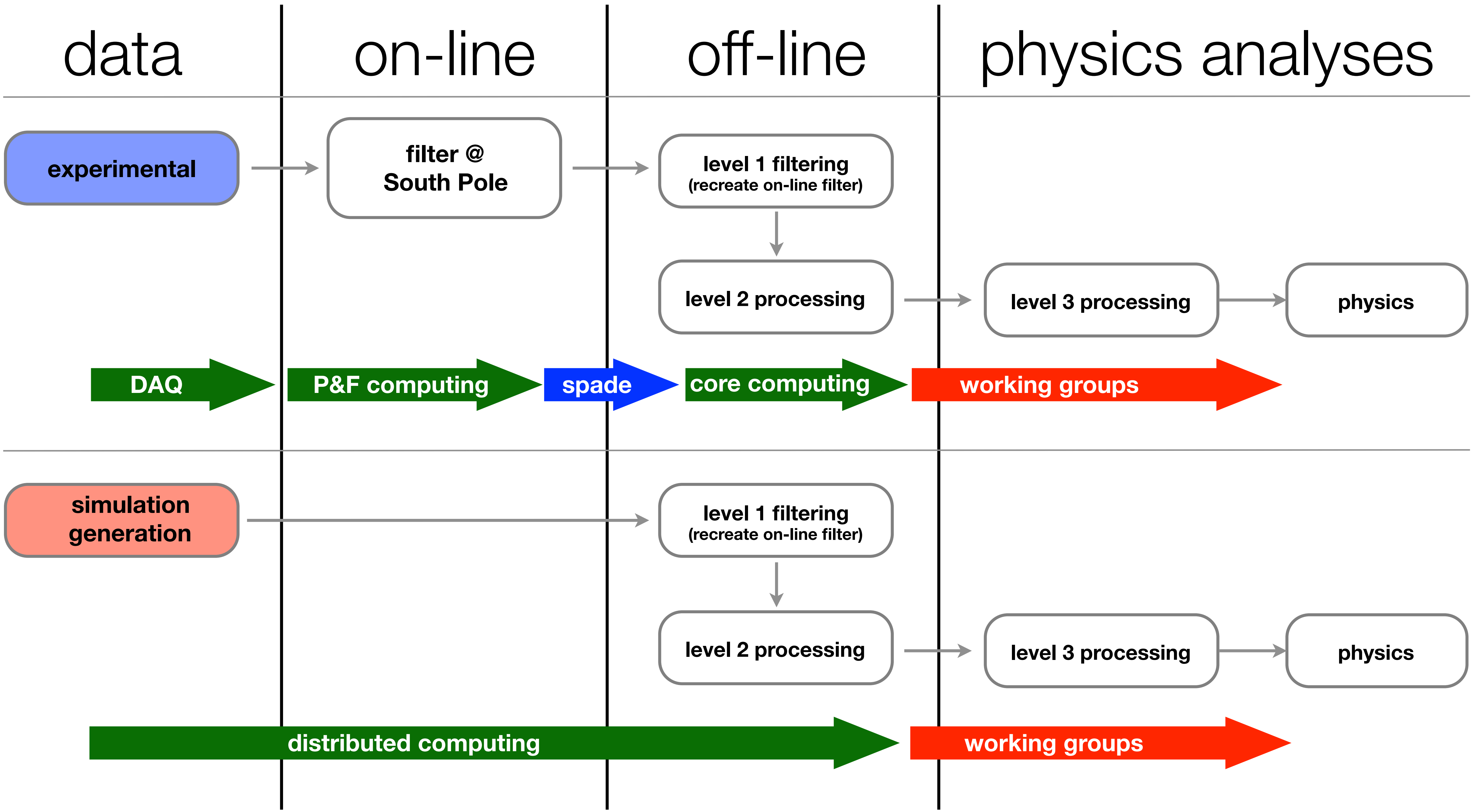
- **Volume** trigger (a.k.a Cylinder trigger in DAQ-land)
  - simple majority of HLC hits (SMT4) with volume element including one layer of strings around a center string
  - cylinder height is 5 DOM-layers (2 up and down from the selected DOM).

- **Slo**w **P**article trigger (SLOP)
  - slow-moving hits along a track
  - lengths of the order of 500μs and extending up to milliseconds

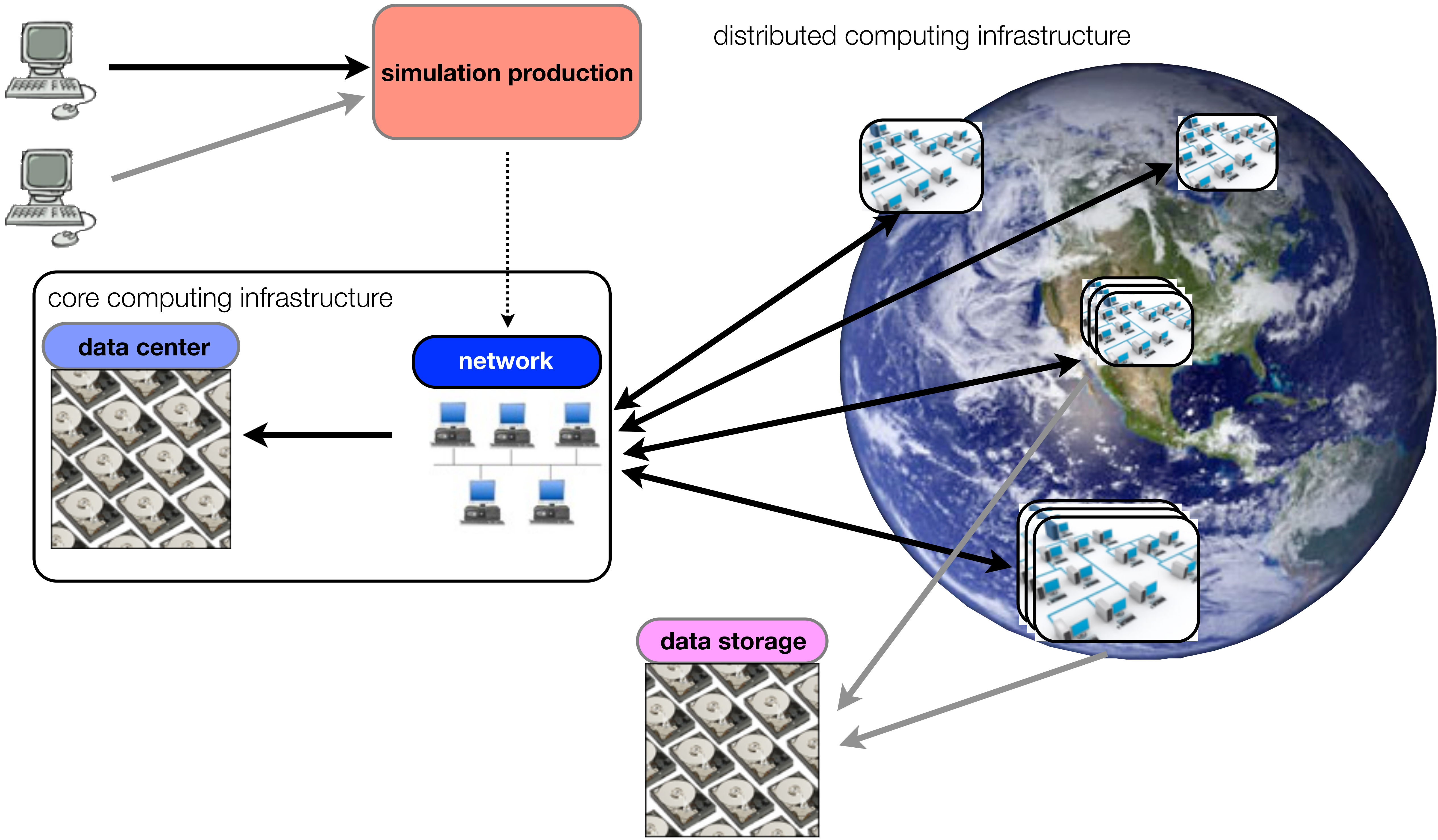
- ~~**Fixed Rate** trigger~~, **Min**imum **Bias** trigger, ~~**Calibration** trigger~~

J. Kelley - DAQ

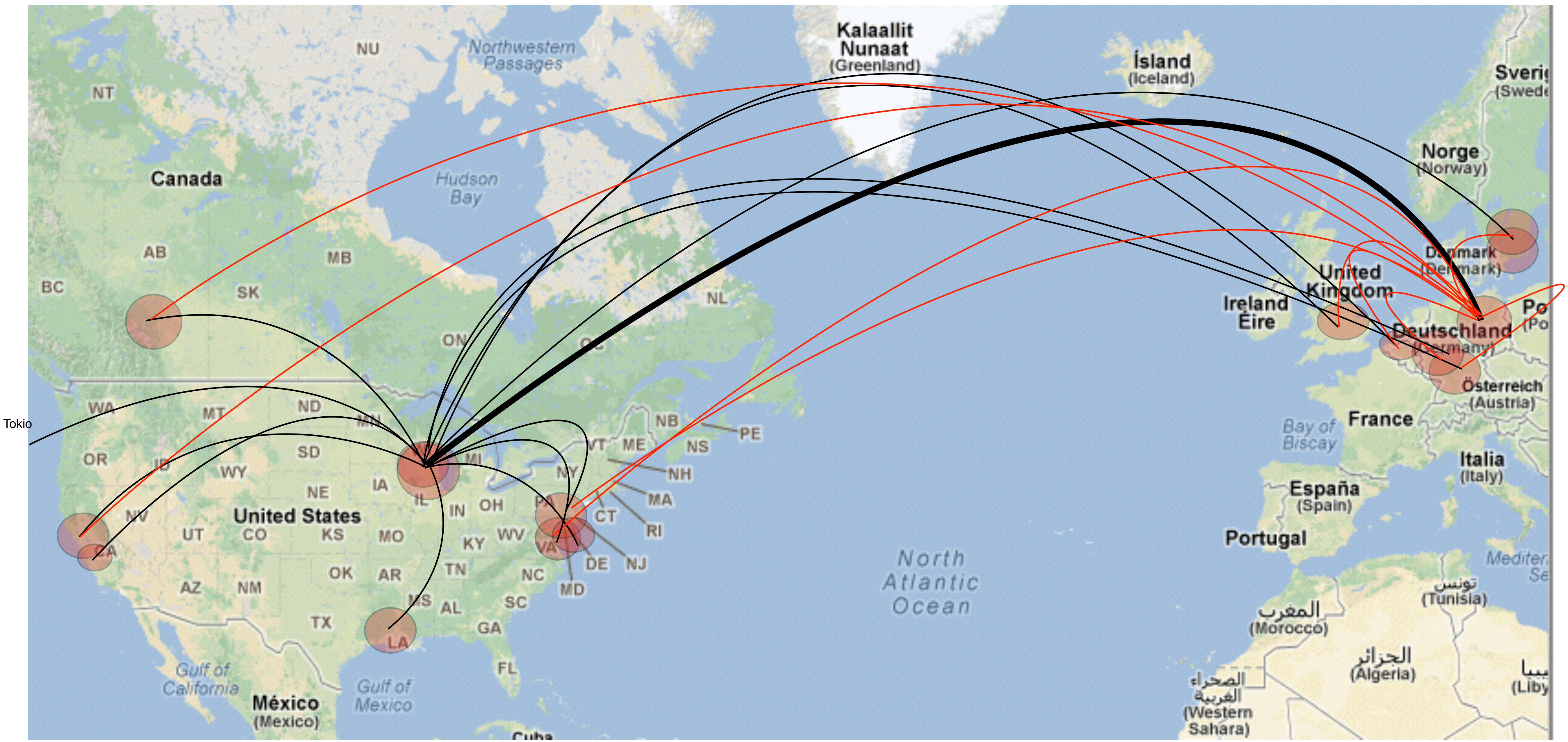
flow of experimental and simulation data



Simulation Production



IceCube Computing Resources



Simprod-Scripts

This project is a collection of scripts, tray segments and IceProd modules used in simulation production. The aim is to provide a central place with standard segments for running simulation in both production and privately.

- **Tray Segments**: IceTray meta-modules that contain several I3Modules with default parameters.
- **IceProd modules**: basic wrappers around tray segments that provide an interface for IceProd.
- **Scripts**: collection of python scripts used in simulation production
- **Examples**: The directory simprod-scripts/resources/examples contains a collection of example scripts for running IPModules
- **Tests:** are run on the build-bots to check that the different parts of the simulation are not broken with each commit to the software repository.

http://software.icecube.wisc.edu/documentation/projects/simprod_scripts/index.html

Simprod-Scripts

IceProd Modules $I3_SRC/simprod-scripts/python/modules

Corsika	NuGen
MuonGunGenerator	GENIE
Example	CLSim
API	PPC

Tray Segments $I3_SRC/simprod-scripts/python/segments

Calibration	GenerateNeutrinos
DetectorSim	GenerateNoiseTriggers
GenerateAirShowers	HybridPhotonicsCLSim
GenerateCosmicRayMuons	Polyplopia
GenerateFlashers	PropagateMuons
GenerateIceTopShowers	

http://software.icecube.wisc.edu/documentation/projects/simprod_scripts/index.html

Simprod-Scripts

Examples:

$I3_SRC/simprod-scripts/resources/examples/fullSimulation.py

```
./fullSimulation.py -n 10 --seed=43682 --datasetnumber=1 --runnumber=1 --no-hybrid --icemodel=SpiceLea \
                    --detector=IC86 --unshadowed-fraction=0.99 --flavor=NuTau --outfile=taus.i3 --skip-calibration \
                    --from-energy=1000 --to-energy=10000000 --include-gcd-in-outfile
```

-n 10 Simulate 10 events (the output file might contain fewer events because not everything will trigger).

--seed=43682 A random number generator seed

--datasetnumber=1 --runnumber=1 A combination of "dataset" and "run" numbers. You can keep the same seed for all of them, each one will give a distinct independent set of events. (So you can think of the combination of seed, dataset and run as the actual random number seed. Or, put another way, there is no need to have different seeds for different runs.)

--**no-hybrid** Do not use hybrid simulation mode, i.e. propagate everything using direct simulation. There will be no photon tables and things like ice anisotropy/SpiceLea will work.

--icemodel=SpiceLea The other two models are "Spice1" and "SpiceMie". Including things like WHAM would be trivial if you need it.

--detector=IC86 This will select a GCD file from $I3_PORTS automatically. Currently works for IC86 and IC79.

--**unshadowed-fraction**=0.99 This is the "DOMEfficiency", currently named like this for compatibility with other tools that use the same name.

--flavor=NuTau You can set this also to "NuE" and "NuMu".

--**outfile**=... The name of your final .i3 file you want to generate.

--from-energy=1000 --to-energy=10000000 The energy range in GeV. Currently not implemented for MuonGun which uses fixed energy ranges. Should be fixed soon.

--**include-gcd-in-outfile** Use this option if you want to generate output files with GCD frames in them. It makes them much easier to use, but of course wastes some space..

http://software.icecube.wisc.edu/documentation/projects/simprod_scripts/examples/fullSimulation.html

Simprod-Scripts

Scripts:

$I3_SRC/simprod-scripts/resources/scripts

(run the individual pieces as broken down by production tasks)

```
$  python nugen.py -h

        Usage: nugen.py [options]

        Options:
          -h, --help            show this help message and exit
          --no-execute          boolean condition to execute
          --gcdfile=GCDFILE     GeoCalibDetStatus filename
          --outputfile=OUTPUTFILE
                                Output filename
          --summaryfile=SUMMARYFILE
                                XMLSummary filename
          --mjd=MJD             MJD for the GCD file
          --seed=RNGSEED        RNG seed
          --procnum=RNGSTREAM
        …
```

http://software.icecube.wisc.edu/documentation/projects/simprod_scripts/examples/fullSimulation.html

Simprod-Scripts

Exercise: Running scripts**:**

```
icecube@M16:~$ ssh submitter
[submitter]$ cd /data/sim/sim-new/bootcamp16/
[submitter]$ condor_submit interactive_gpu.condor -interactive
Submitting job(s).
1 job(s) submitted to cluster 120263704.
Waiting for job to start…
Welcome to slot1@gtx-00.icecube.wisc.edu!

[gtx-00]$ cd $_CONDOR_SCRATCH_DIR
[gtx-00]$ eval `/cvmfs/icecube.opensciencegrid.org/py2-v1/setup.sh`
[gtx-00]$ /data/sim/sim-new/bootcamp16/combo/build/env-shell.sh
*******************************************************************
*                                                                 *
*                  W E L C O M E  to  I C E T R A Y               *
*                   Version combo.trunk     r147415               *
*                                                                 *
*                You are welcome to visit our Web site            *
*                        http://icecube.umd.edu                   *
*******************************************************************

[gtx-00]$ cp /data/sim/sim-new/bootcamp16/GeoCalibDetectorStatus_2015.57161_V0.i3.gz ./
[gtx-00]$ python $I3_BUILD/simprod-scripts/resources/scripts/nugen.py \
                 --gcdfile=GeoCalibDetectorStatus_2015.57161_V0.i3.gz \
                 --outputfile=nutau.i3 --nevents=100 \
                 --seed=123 --procnum=0  --nproc=1 \
                 --FromEnergy=1e5 --ToEnergy=1e6 --NuFlavor=NuTau
[gtx-00]$ dataio-pyshovel nutau.i3
```

Simprod-Scripts

Exercise: Running scripts**:**

```
[gtx-00]$ python $I3_BUILD/simprod-scripts/resources/scripts/ppc.py \
                --gcdfile=GeoCalibDetectorStatus_2015.57161_V0.i3.gz \
                --inputfilelist=nutau.i3 --outputfile=mcpes.i3   \
                --seed=123 --procnum=0 --nproc=1 --RunMPHitFilter=no \
                --ParticleType=NuTau --UseGPU

[gtx-00]$ dataio-pyshovel mcpes.i3 -lsimclasses

[gtx-00]$ python $I3_BUILD/simprod-scripts/resources/scripts/detector_ic86.py \
                --gcdfile=GeoCalibDetectorStatus_2015.57161_V0.i3.gz \
                --inputfile=mcpes.i3 --outputfile=det.i3 \
                --seed=123 --procnum=0 --nproc=1 --RunID=123

[gtx-00]$ dataio-pyshovel det.i3 -lsimclasses
```

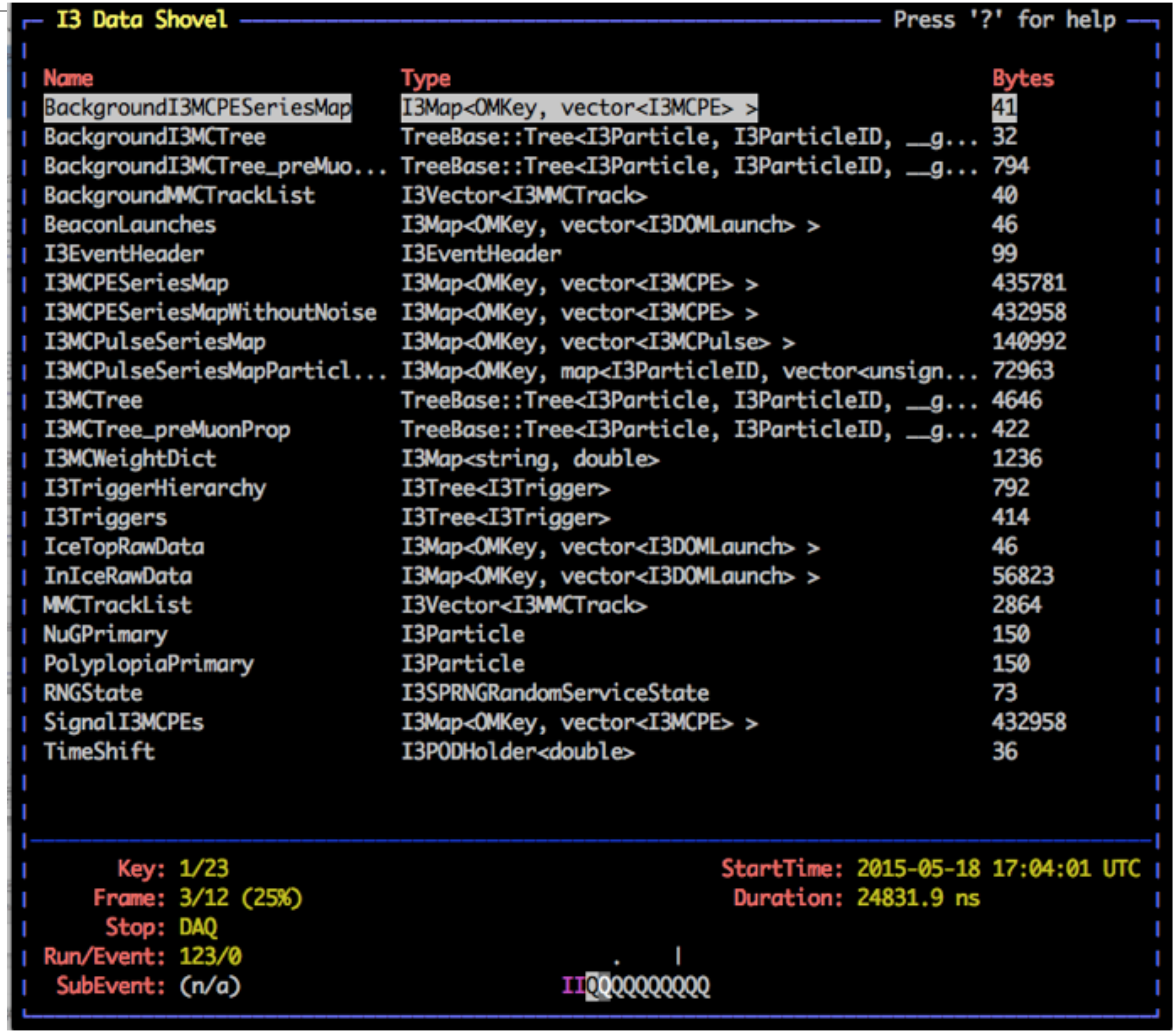
Simprod-Scripts

Exercise: Running scripts**:**

```
[gtx-00]$ python $I3_BUILD/simprod-scripts/resources/scripts/corsika.py \
                --gcdfile=GeoCalibDetectorStatus_2015.57161_V0.i3.gz \
                --nshowers=10000 --outputfile=corsika_bg.i3 --seed=1234 \
                --procnum=0 --nproc=1 --CORSIKAseed=123 --ranpri=2  \
                --corsikaVersion=v6960-5comp \
                --corsikaName=dcorsika --skipoptions=compress

[gtx-00]$ dataio-pyshovel corsika_bg.i3 -lsimclasses

[gtx-00]$ python $I3_BUILD/simprod-scripts/resources/scripts/polyplopia.py  \
              --gcdfile=GeoCalibDetectorStatus_2015.57161_V0.i3.gz \
              --inputfile=mcpes.i3 --outputfile=merged_pes.i3 \
              --seed=1234 --procnum=0 --nproc=1 --backgroundfile=corsika_bg.i3 --mctype=NuTau

[gtx-00]$ python $I3_BUILD/simprod-scripts/resources/scripts/detector_ic86.py \
                --gcdfile=GeoCalibDetectorStatus_2015.57161_V0.i3.gz \
                --inputfile=merged_pes.i3 --outputfile=det_wcoinc.i3 \
                --seed=123 --procnum=0 --nproc=1 --RunID=123

[gtx-00]$ dataio-pyshovel det_wcoinc.i3 -lsimclasses
```

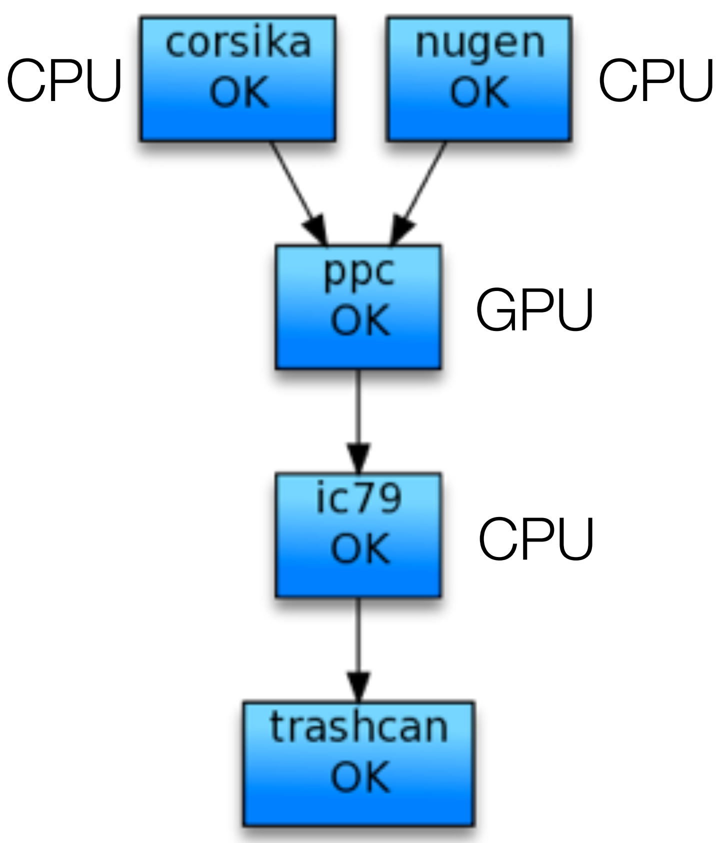
Simprod-Scripts

Exercise: Running scripts**: frame objects**



Directed Acyclic Graphs (DAGs) & GPU-based simulation

- DAGs allow for optimal usage of specialized computing resources such as GPUs
- Don't waste GPU time on CPU-intensive segments.
- Checkpoint simulation by storing intermediate files.
  - You can use the same MCPEs to drive different DOM efficiencies, geometries, filters, etc.
- CondorHT has builtin dagman for driving DAG jobs.



https://research.cs.wisc.edu/htcondor/dagman/dagman.html

More on simulation

1. http://wiki.icecube.wisc.edu/index.php/Simulation_Documentation_Wiki

2. http://wiki.icecube.wisc.edu/index.php/IceCuber's_Guide_to_IceSim_2.6

3. http://wiki.icecube.wisc.edu/index.php/Simulation_Production

4. http://internal.icecube.wisc.edu/simulation/datasets/dataset_category/PHYSICS

5. IRC @ irc.efnet.net: /join #icesim