DD4hep and Simulation with DDSim

André Sailer

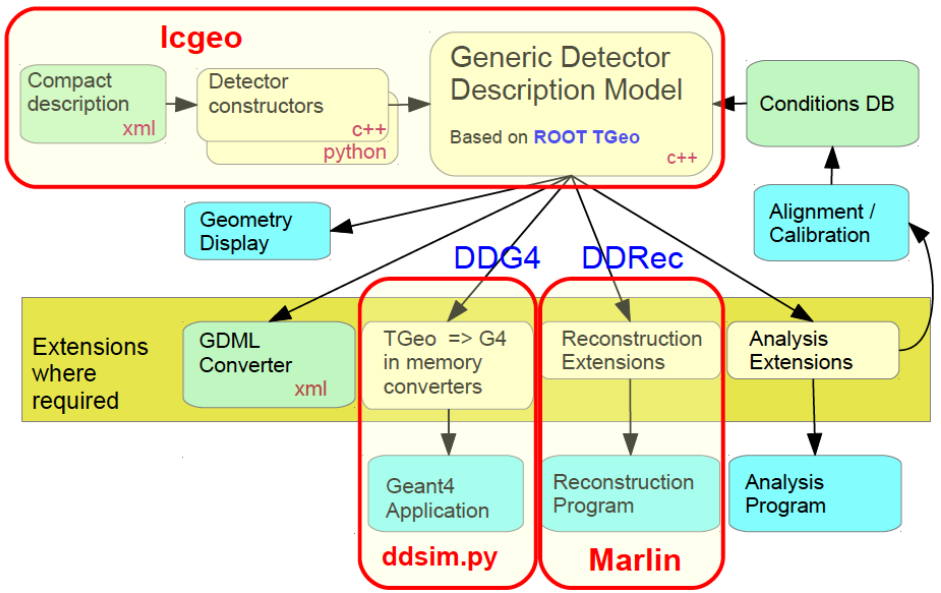
CERN-EP-LCD

CLIC Workshop
January 19, 2016

Table of Contents

DD4hep

- DD4hep: Single source of Geometry for Simulation, Reconstruction, Analysis, . . .
- lcgeo: Collection of linear collider detector (CLICdp,ILD,SiD) and testbeam (FCal) geometry drivers
- ddsim(.py): The program to run simulation with Geant4
  - Provide the same functionality as Mokka and SLIC (and more)
  - Including the many small features included over years of using them
  - Should be easy to use for users



DDG4

- DDG4: DD4hep gateway to Geant4
- TGeo geometry is converted to Geant4 geometry in memory
- Enable access to Geant4 functionality via (user provided) plug-ins
  - Generation, Event Action, Tracking Action, Stepping Action, SensitiveDetector, PhysicsList
  - Configurable via XML, C++ script and **python**

```
#...
gen = DDG4.GeneratorAction(kernel,"LCIOInputAction/LCIO1" )
gen.Input="LCIOFileReader|"+inputFile
#...
```

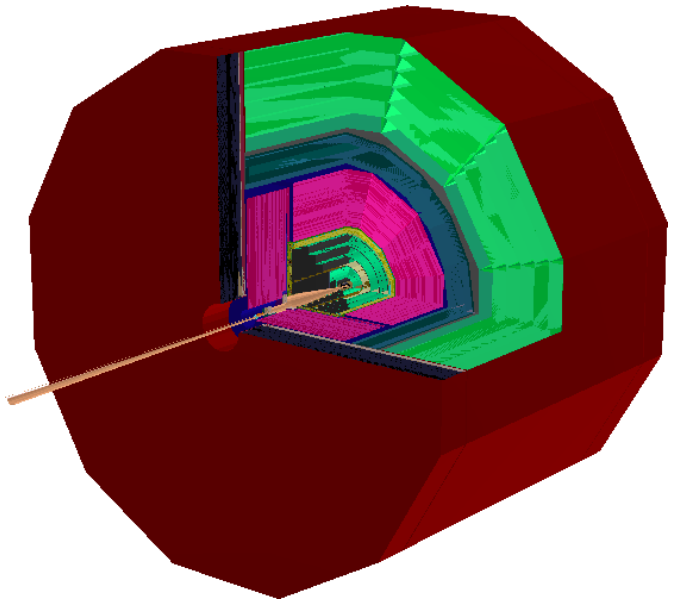
Full Detector Simulation Recipe

Ingredients:

- Geometry, Magnetic Field, and Sensitive Detectors
- Monte Carlo Events or Particle Gun
- PhysicsList
- MC-Truth linking

Geometry and Detector Model

See presentation by M. Petric on Friday



- All drivers ready for simulation
- CLIC detector model parameters not yet finalised

Magnetic Field Configuration

Tested configurations

- Constant (Solenoid) Field
- 2D Field Map: B_r, B_z for given r, z

Stepping configuration

- Configuration of the propagator of particles in the field
- E.g.: Largest Acceptable Step: Default 1 km, lead to random crashes in Mokka
- Can also control type of stepper and parameters for its precision

Monte Carlo Generator Information and Handling

Reading files from MC Generators

- HepMC, lcio, HEPEvt, stdhep
- Skipping of events at start of file
- Treating particles unknown to Geant (e.g.: some B-Baryons with non-zero decay length)

Boost and Smearing of the vertex

- Lorentz-Boost for the crossing angle
- Smearing or moving of initial vertex position, if desired

MC-Truth Information during the simulation

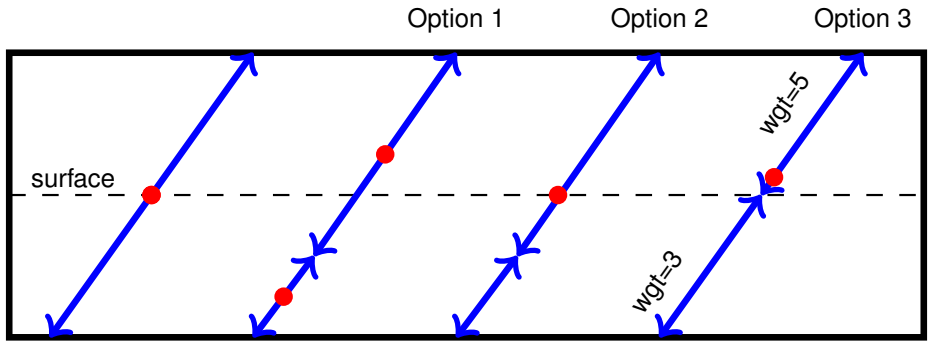
- Linking hits and particles that caused them

Simulation Parameters

- PhysicsList: All physics lists from Geant4 are available
  - Also allows electromagnetic extension to be chosen
- Can be extended: E.g. to add certain particles
- Range-Cut: Material dependent energy threshold for production of secondaries
  - Global Range-Cut in simulation setup
  - Local Range-Cut via regions in the detector XML

Sensitive Detector: Tracker

- Many different options how to store the particles passing through the sensor

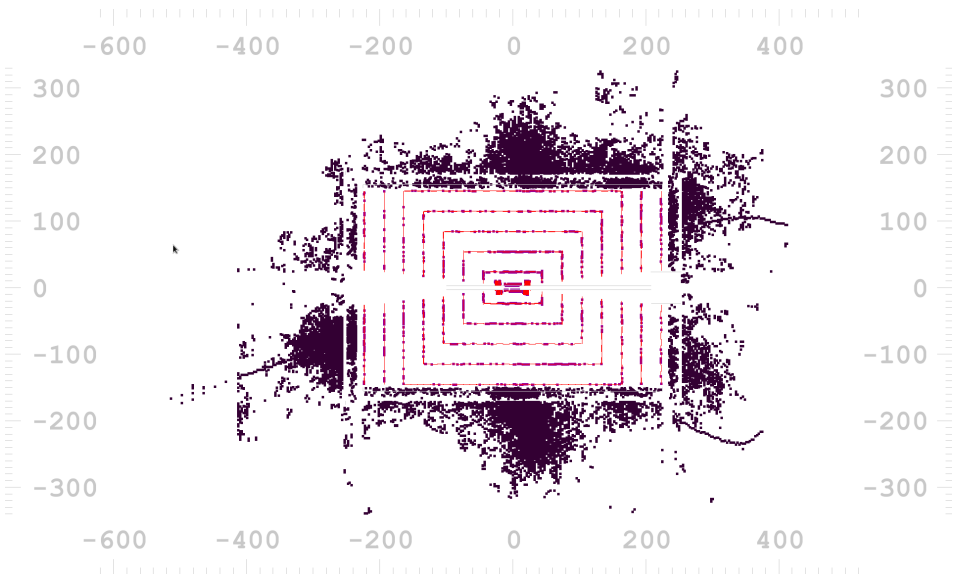


1. Store each *step* as a *hit*: Many hits for one particle in one volume. Each hit is placed at the average of each step
2. Sum the *steps* in the sensitive volume as a single *hit*, hit is placed at the average of all steps
3. Sum the *steps* and place the hit at the energy weighted position of the steps

- Hits need to be on the surface during reconstruction:
  - Option 2 guarantees hit on the surface
  - Option 1 and 2 need projection onto the surface
  - Option 1 needs some kind of hit merging during digitisation

Sensitive Detectors: Calorimeter

- No issue about where to place the hit: At the centre of the cell provided by the segmentation
- Detailed shower mode: Store all contributions (hit time) to the energy in a cell
- Thresholds to write out hits (e.g.: overlay needs to be considered as well)
- Will need some iteration between reconstruction and simulation



First Users and Validation

All the ingredients need to be validated for correct behaviour

- In recent months a number of issues were fixed
  - Sensitive detectors and hits
  - Skipping of events was not working for most file types
- Some features still to be implemented

First uses

- Started larger scale simulations for reconstruction developments
- Some background simulation have been done by A. Nurnberg
- ILD is also validating the simulation

Putting it All Together: Running the Simulation

- ddsim python program: provides an easy to configure interface to simulation
- steering file is basically a collection of parameters to run simulation
  - Input File, Output File, PhysicsList, GeometryModel,...
  - Many things configurable via command line, everything via steering file
- All the DDG4 plug-ins are then controlled for the user
  - E.g.: the user gives `inputFile='my.slcio'`, and the rest is taken care of

```
#...
gen = DDG4.GeneratorAction(kernel,"LCIOInputAction/LCIO1" )
gen.Input="LCIOFileReader|"+inputFile
#...
```

Steering File

Example steering file

```
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV, keV

SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v04.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"

SIM.part.minimalKineticEnergy = 1*MeV

SIM.filter.filters['edep3kev'] =
   dict(name="EnergyDepositMinimumCut/3keV",
        parameter={"Cut": 3.0*keV} )
```

- Parameters in the steering file can be overwritten from the command line
- Steering file can be generated: ddsim –dumpSteeringFile
  - Including documentation of parameters, all coming from the same source
  - Thanks to *reflection* in python

ddsim –steeringFile mySteer.py –inputFile Muons.HEPEvt -N10

Tab Completion

- For automatic completion of command line parameters: bash **bash-completion**, python: argcomplete
- Much more convenient interactive use

```
$ ddsim
--action.calo                --filter.tracker             --part.keepAllParticles
--action.mapActions          -G                           --part.minimalKineticEnergy
--action.tracker             --gun.direction              --part.printEndTracking
--compactFile                --gun.energy                 --part.printStartTracking
--crossingAngleBoost         --gun.isotrop                --part.saveProcesses
--dump                       --gun.multiplicity           --physics.decays
--dumpParameter              --gun.particle               --physics.list
--dumpSteeringFile           --gun.position               --physicsList
--enableDetailedShowerMode   -h                           --physics.rangecut
--enableGun                  --help                       --printLevel
--field.delta_chord          -I                           --random.file
--field.delta_intersection   --inputFiles                 --random.luxury
--field.delta_one_step       -M                           --random.replace_gRandom
--field.eps_max              --macroFile                  --random.seed
--field.eps_min              -N                           --random.type
--field.equation             --numberOfEvents             --runType
--field.largest_step         -O                           -S
--field.min_chord_step       --outputFile                 --skipNEvents
--field.stepper              --output.inputStage          --steeringFile
--filter.calo                --output.kernel              -v
--filter.filters             --output.part                --vertexOffset
--filter.mapDetFilter        --output.random              --vertexSigma
$ ddsim -
```

DDSim on the GRID

- New interface, and work-flow module necessary for ddsim
- Will be in iLCDirac release in February
- Very similar to the Mokka and SLIC interfaces

```
from ILCDIRAC....Applications import Mokka

mo = Mokka()
mo.setInputFile("muons.HEPEvt")
mo.setDetectorModel("CLIC_ILD_CDR")
mo.setSteeringFile("mysteer.steer")
mo.setMacFile('MyMacFile.mac')
mo.setStartFrom(10)
mo.setNumberOfEvents(30)
```

```
from ILCDIRAC....Applications import DDSim

dd = DDSim()
dd.setInputFile("muons.HEPEvt")
dd.setDetectorModel("CLIC_o3_v04")
dd.setSteeringFile("mysteer.steer")
dd.setStartFrom(10)
dd.setNumberOfEvents(30)
```

Conclusion

- Simulation with DDG4 is usable by a broader audience
- The Simulation is being validated
- Missing features are rapidly implemented

Acknowledgements and Thank You

Thanks to

M.Frank, F.Gaede, D.Hynds, S.Lu, N.Nikiforos,
A.Nurnberg, M.Petric, R.Simoniello

for Code, Material, Help, Testing etc.

Thank you for your Attention