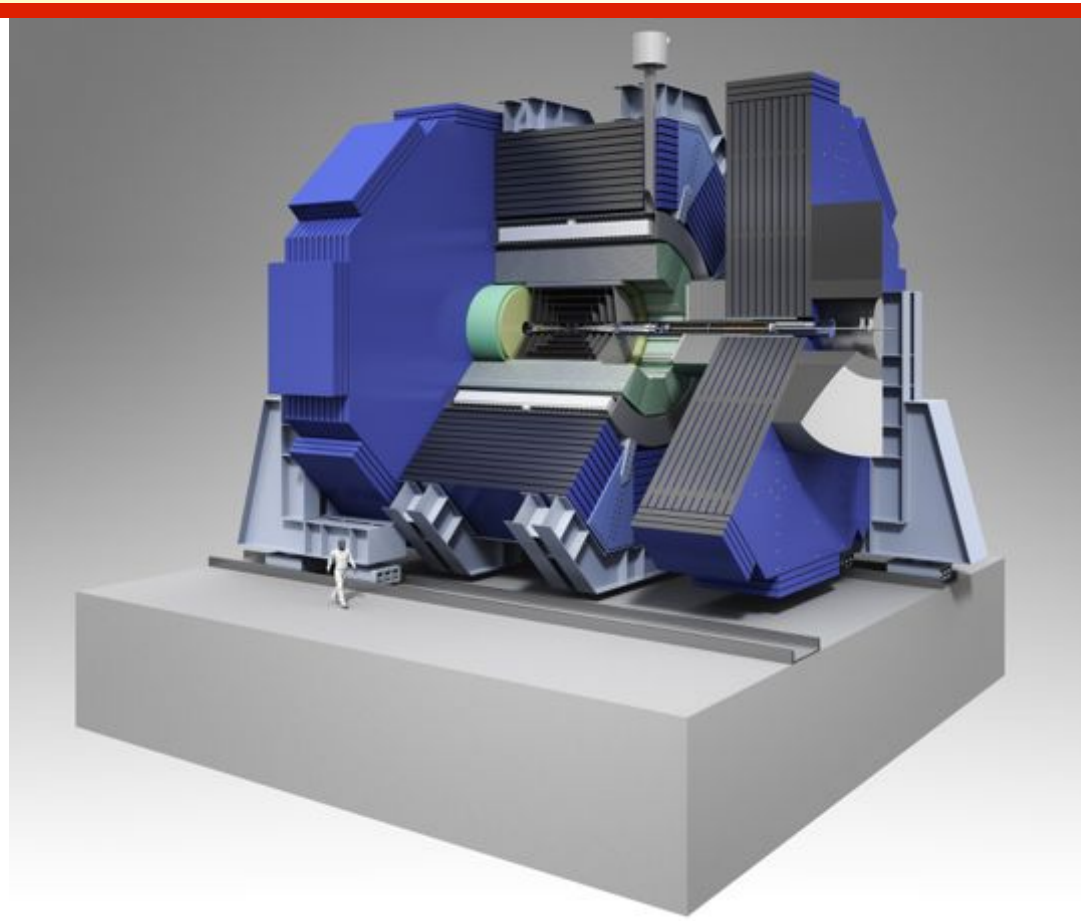
SiD Software Status

ECFA Linear Collider Workshop, 3 June 2016

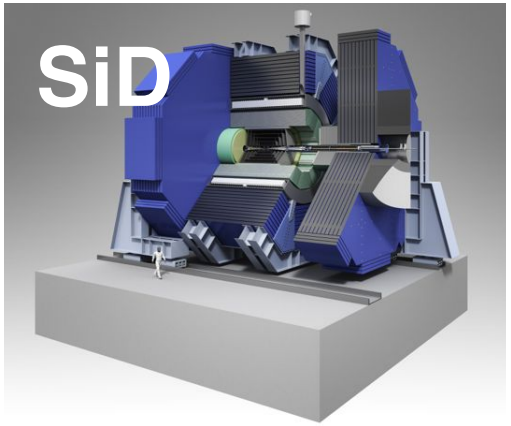
- DD4HEP model status
- Reconstruction status
- To do list and plans

Aidan Robson
Dan Protopopescu
Bogdan Mishchenko

University of Glasgow | Experimental Particle Physics



Existing model



Detector geometry	lcdd generated from xml
Detector simulation	slic (GEANT4)
Reconstruction	org.lcsim & slicPandora
Event data model	LCIO

DD4HEP chain

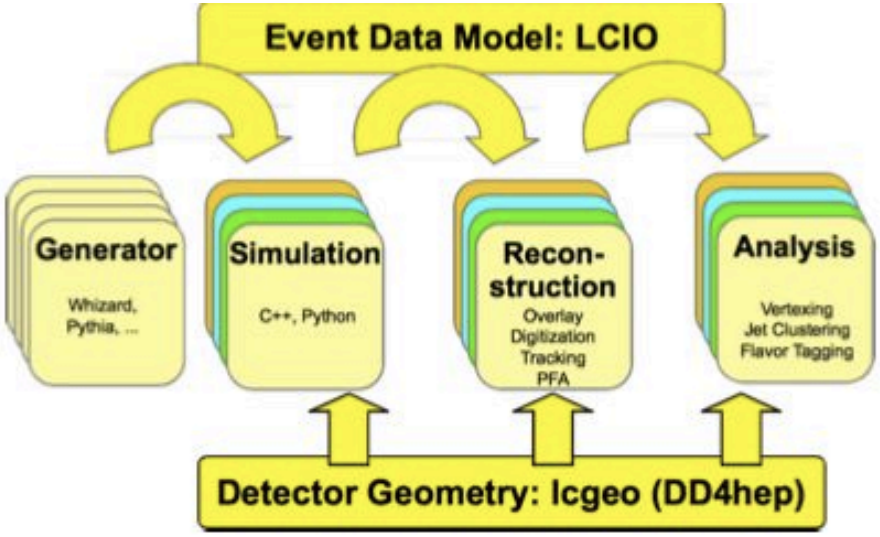
- SiD simulation and reconstruction developed by SLAC – large effort from Norman Graf, Jeremy McCormick and colleagues
- SLAC has reduced the available effort over several years; recently became zero
- Software chain can be run, but maintenance/development increasingly difficult
- At LCWS15, SiD decided to switch to DD4HEP to align with ILD/CLICdp and profit from shared effort
- Frank Gaede made initial port of SiD 'LOI3' model to DD4HEP; further development in Glasgow

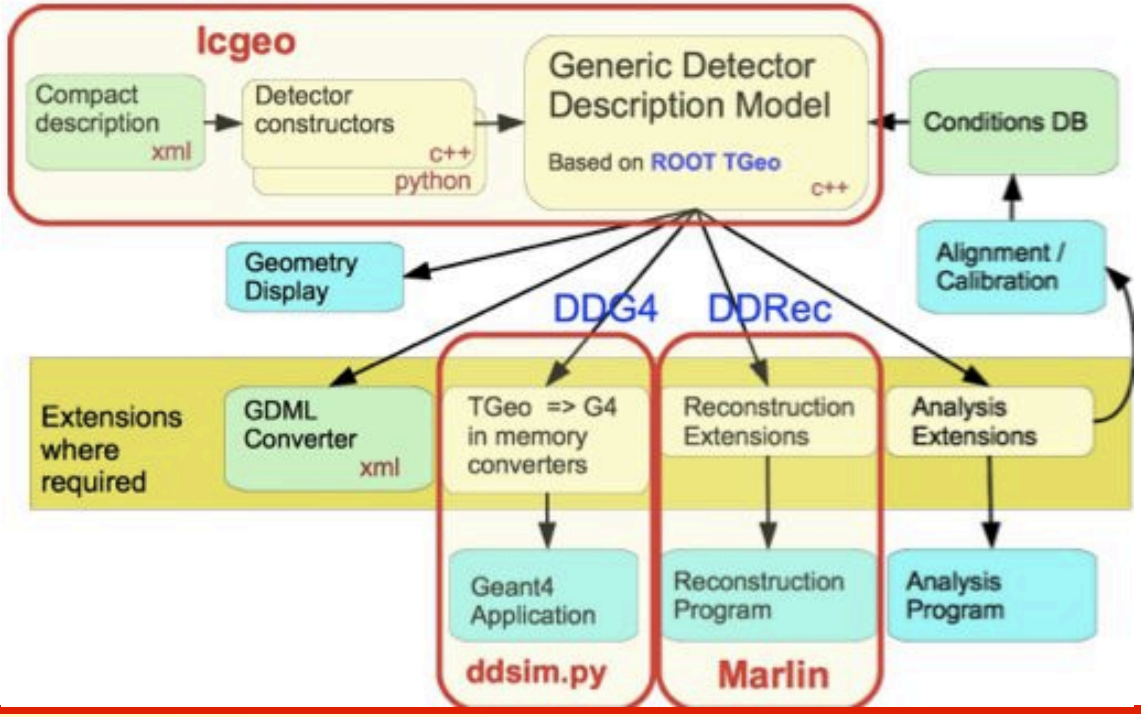
DD4HEP framework

Single detector description

compact xml

-> interpreted by C++ ‘driver’ to create model for geant passed into ddsim, and reconstruction





What we have been doing…

Simulation

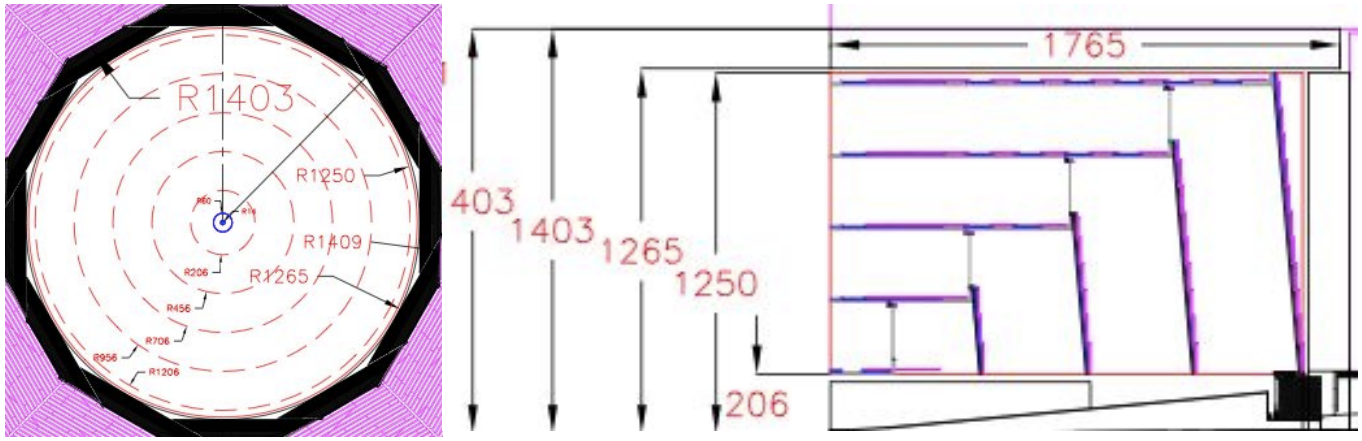
- Updating xmls to latest naming and structure conventions
  - e.g. allows model to be included in automatic tests (see Andre's DD4HEP talk)
- Updating dimensions to latest engineering drawing (ongoing)
  - updates since original SiD concept ('LOI3')
- Moved readouts & plugins to the individual detector descriptions
  - ensures modularity
- Verified visualisations
- Checked overlaps

Strategy: use generic DD4HEP drivers, put all SiD-specific parts in xml for now – makes sense while DD4HEP development still very active.

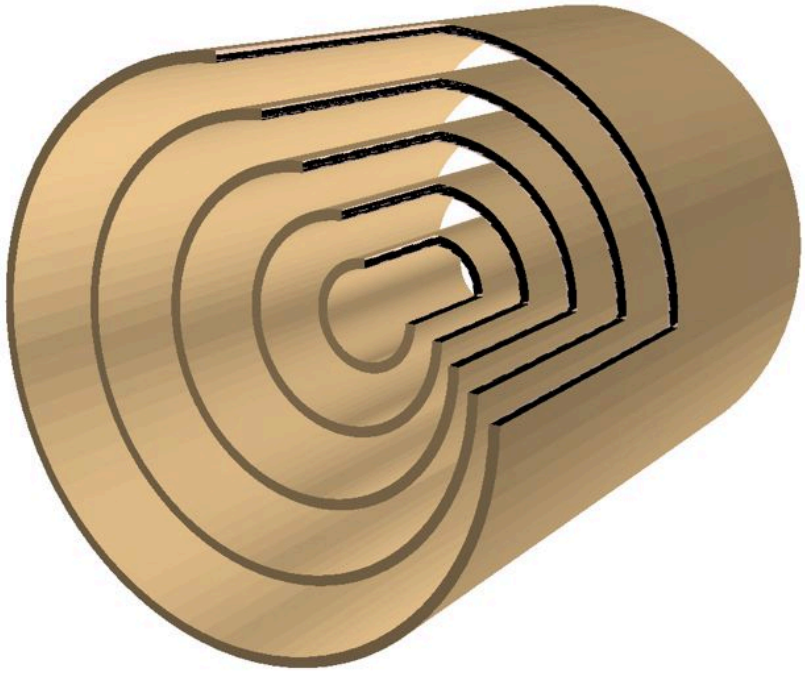
Detailed/custom drivers can come later if needed.

Slow but steady progress – e.g. last week realised muon endcap wasn't being simulated (typo in xml), now fixed.

Tracker barrel

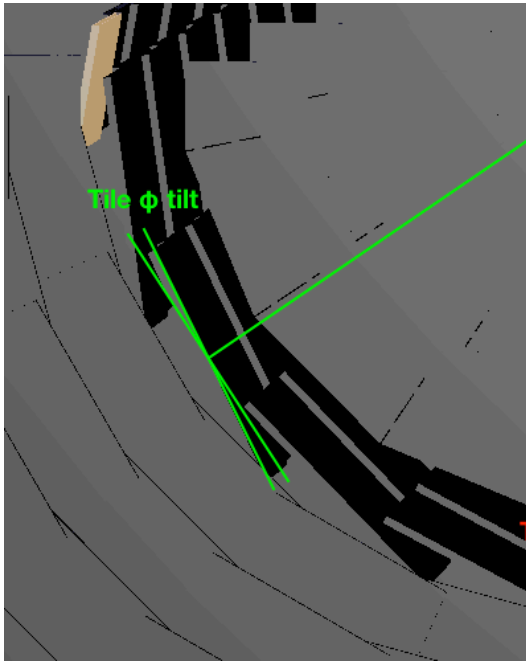


engineering drawings



5 barrel layers; here showing detector envelopes

module detail inside envelope



Tracker barrel

original ported xml – all dimensions hard-coded

```
<layer module="SiTrackerModule_Layer1" id="1">
  <barrel_envelope inner_r="215.075*mm" outer_r="245.0*mm" z_length="578 * 2*mm"/>
  <rphi_layout phi_tilt="0.17506*rad" nphi="20" phi0="0.*rad" rc="216.355*mm + 5.0*mm" dr="0.0"/>
  <z_layout dr="4.0*mm" z0="512.128*mm" nz="13"/>
</layer>
<layer module="SiTrackerModule_Layer2" id="2">
  <barrel_envelope inner_r="465.075*mm" outer_r="501.0*mm" z_length="749.8 * 2*mm"/>
  <rphi_layout phi_tilt="0.12217*rad" nphi="38" phi0="0.087*rad" rc="466.355*mm + 5.0*mm" dr="0.0"/>
  <z_layout dr="4.0*mm" z0="690.605*mm" nz="17"/>
</layer>
<layer module="SiTrackerModule_Layer3" id="3">
```

updated xml – as much as possible is computed from a small number of parameters

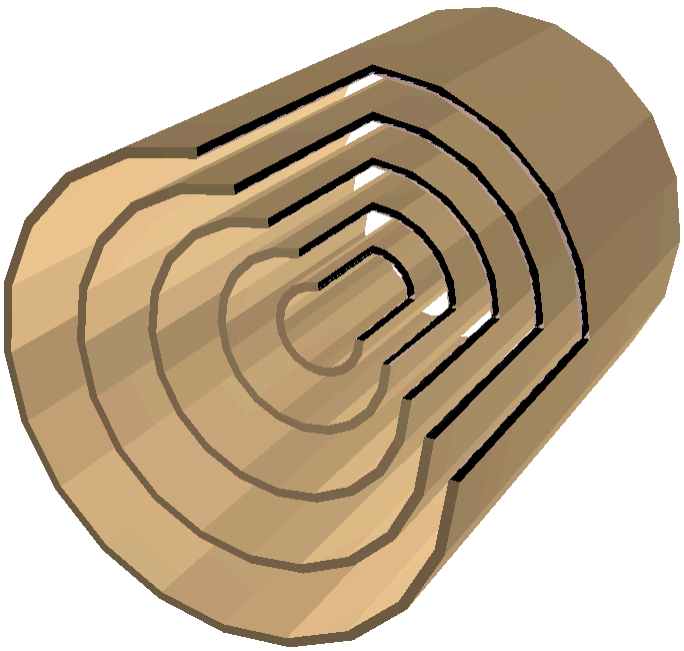
```
<constant name="SiTracker_outer_radius" value="1235.0*mm"/>
<constant name="SiTracker_inner_radius" value="206.00*mm"/>
<constant name="SiTracker_zmax" value="1637.0*mm"/><!-- half-length -->
<constant name="SiTracker_EndcapLayers" value="4"/>
<constant name="SiTracker_BarrelLayers" value="SiTracker_EndcapLayers + 1"/>
```

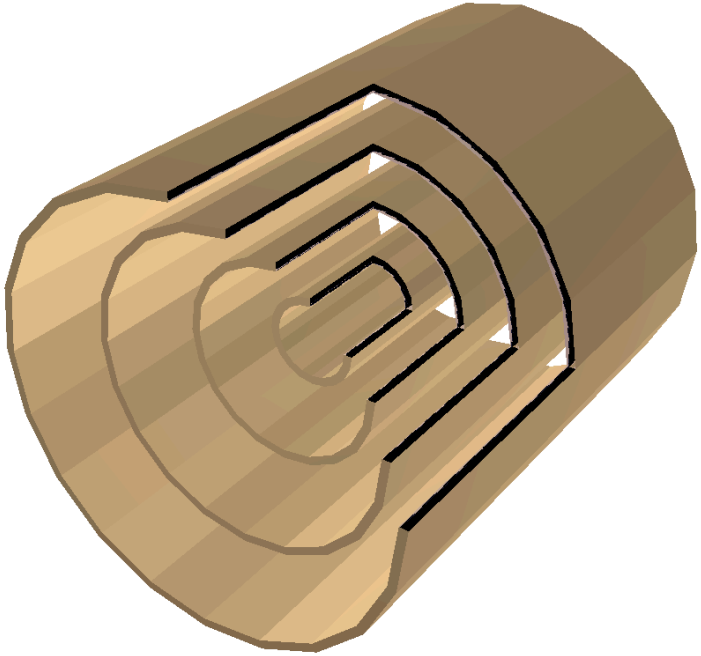
–> allows studies

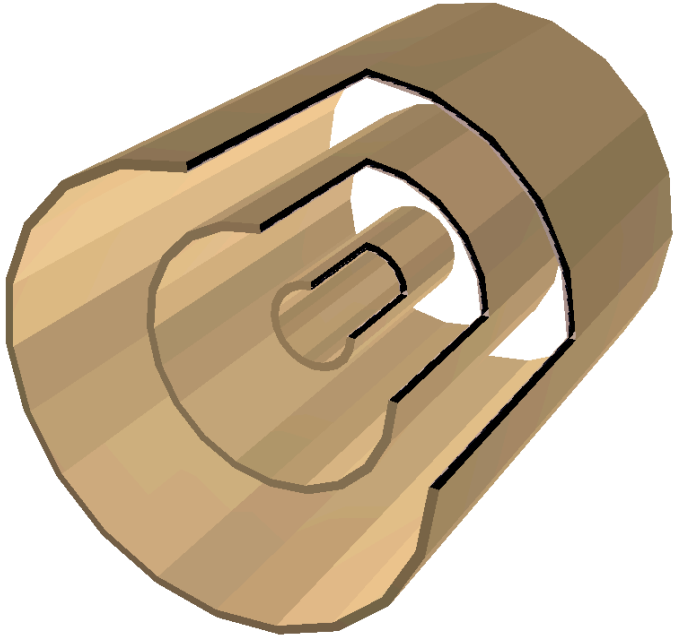
```
<layer module="SiTrackerModule_Layer1" id="1">
  <!--barrel_envelope inner_r="215.075*mm" outer_r="245.0*mm" z_length="578 * 2*mm"/-->
  <barrel_envelope inner_r="SiTrackerBarrel_inner_rc - 6.2*mm" outer_r="SiTrackerBarrel_inner_rc + 45.0*mm" z_length="(SiTrackerBarrel_inner_
+ 60*mm)*2"/>
  <rphi_layout phi_tilt="0.17506*rad" nphi="SiTrackerBarrel_inner_nphi" phi0="0.*rad" rc="SiTrackerBarrel_inner_rc" dr="0.0"/>
  <z_layout dr="4.0*mm" z0="SiTrackerBarrel_inner_z0" nz="SiTrackerBarrel_inner_nz"/>
</layer>
<layer module="SiTrackerModule_Layer2" id="2">
  <!--barrel_envelope inner_r="465.075*mm" outer_r="501.0*mm" z_length="749.8 * 2*mm"/-->
  <barrel_envelope inner_r="SiTrackerBarrel_rc_2 - 6.2*mm" outer_r="SiTrackerBarrel_rc_2 + 45.0*mm" z_length="(SiTrackerBarrel_z0_2 + 60*mm)*
">
```

Tracker barrel

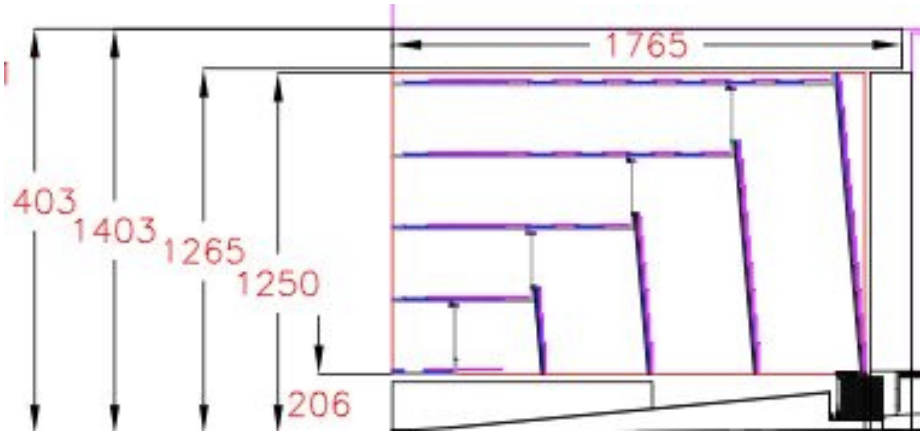
e.g. very straightforward to alter number of layers:



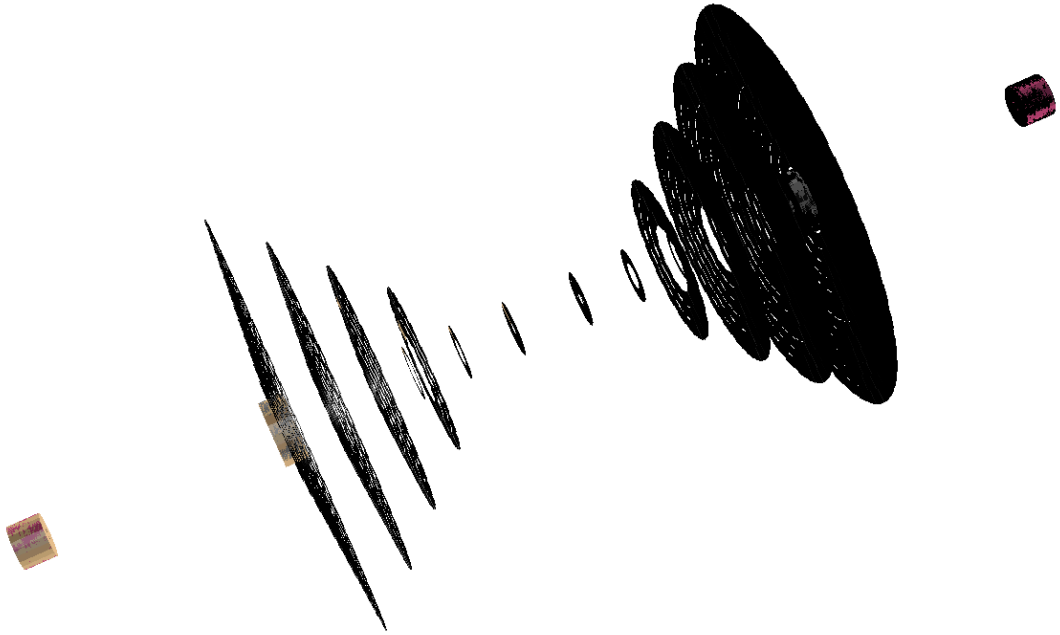


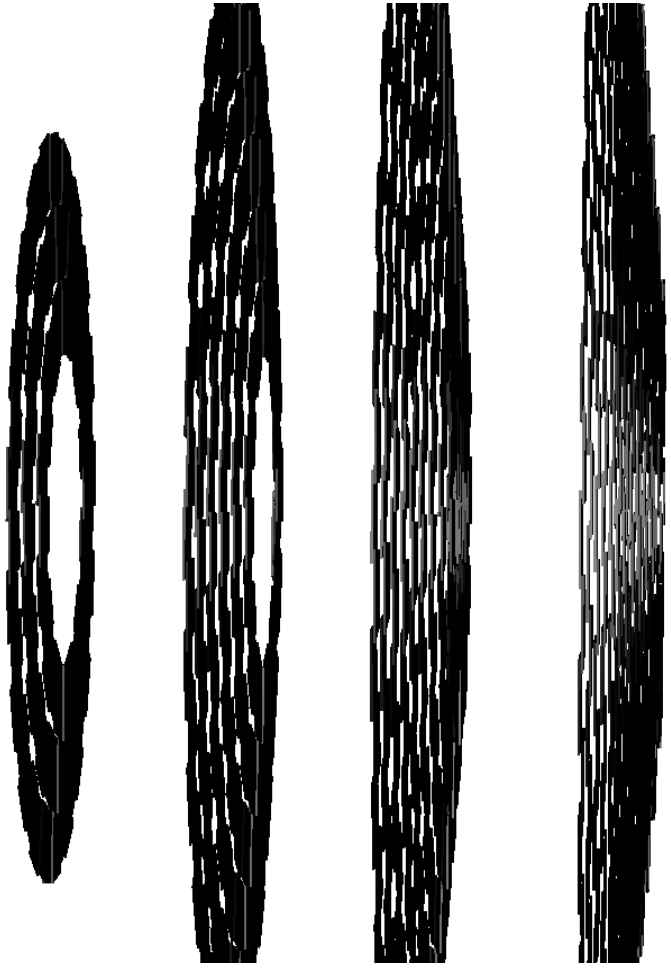


Tracker endcaps



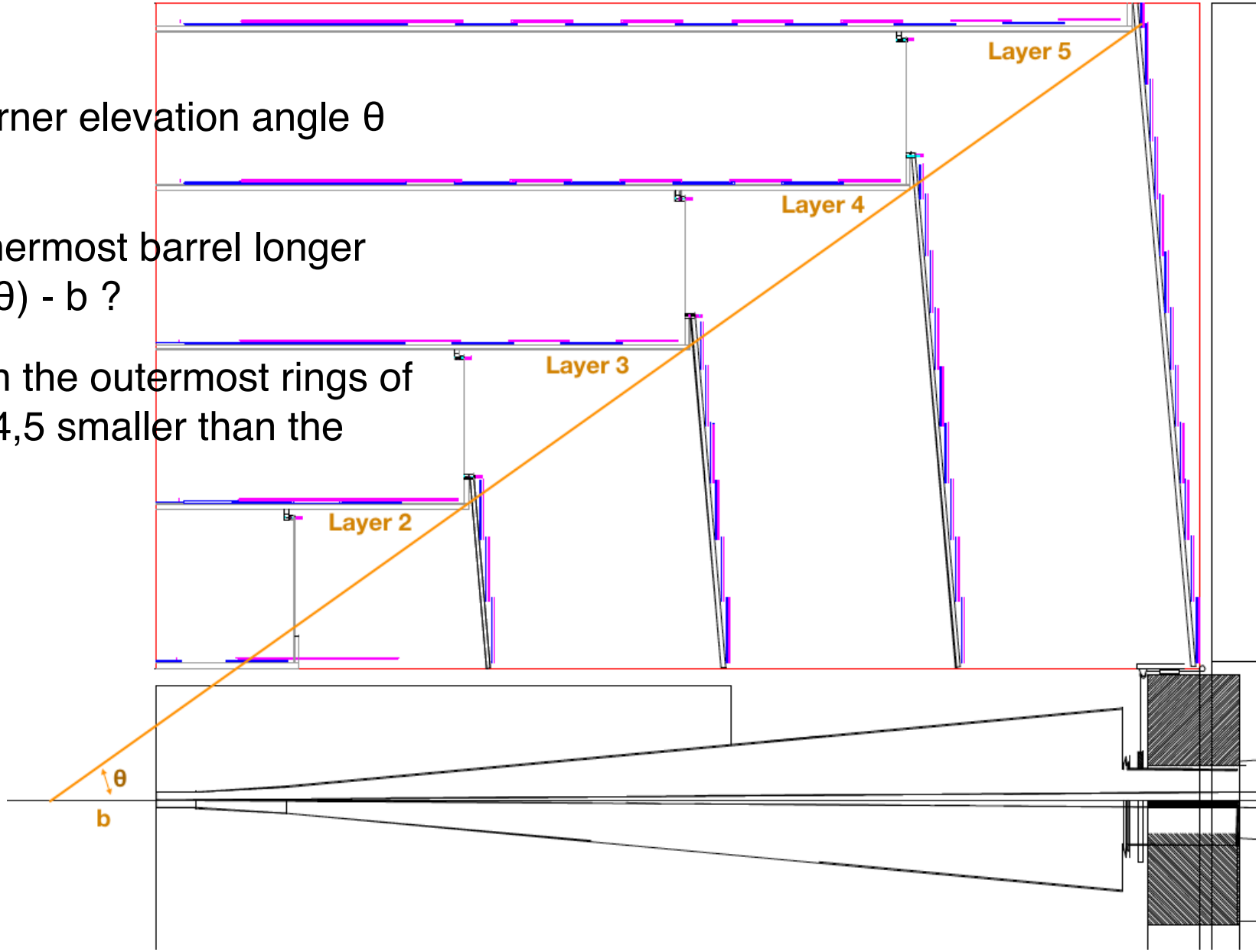
Similarly, endcap parameters now computed from barrel where possible



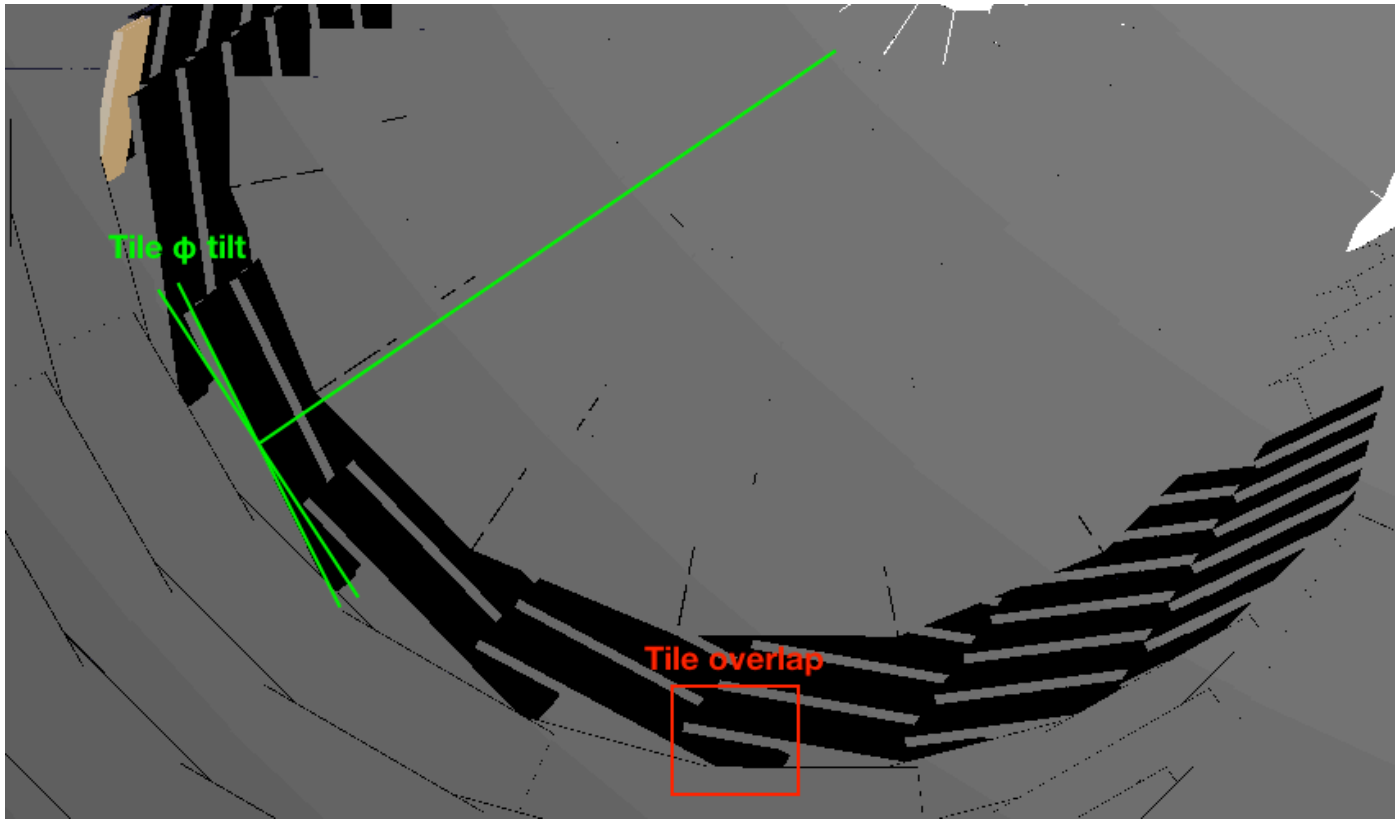


Some tracker questions…

- how is the corner elevation angle θ set?
- why is the innermost barrel longer than r_inner*cot(θ) - b ?
- are the tiles in the outermost rings of endcap layers 3,4,5 smaller than the rest?



Some tracker questions…



For calculating the number of modules along z and around the barrel circumference, we need to know

- what is the tile (petal) module overlap (in mm)?
- how is the tile ϕ tilt calculated?

Some tracker questions…

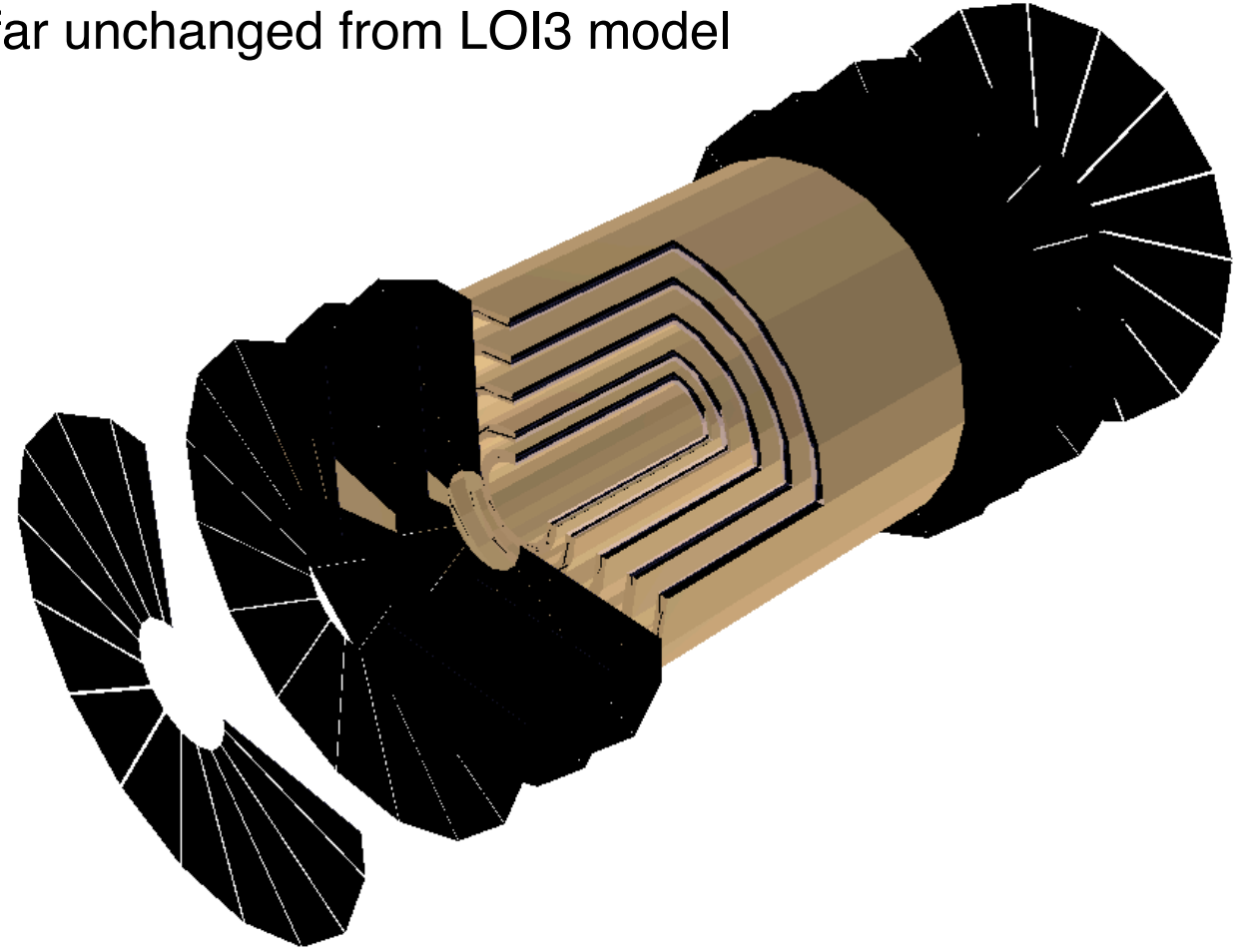
```
<layer module="SiTrackerModule_Layer2" id="2">
  <barrel_envelope inner_r="465.075*mm" outer_r="501.0*mm" z_length="749.8 * 2*mm"/>
  <rphi_layout phi_tilt="0.12217*rad" nphi="38" phi0="0.087*rad" rc="466.355*mm + 5.0*mm" dr="0.0"/>
  <z_layout dr="4.0*mm" z0="690.605*mm" nz="17"/>
</layer>
```

- how are the barrel ϕ rotations set? What is the reason for these rotations?

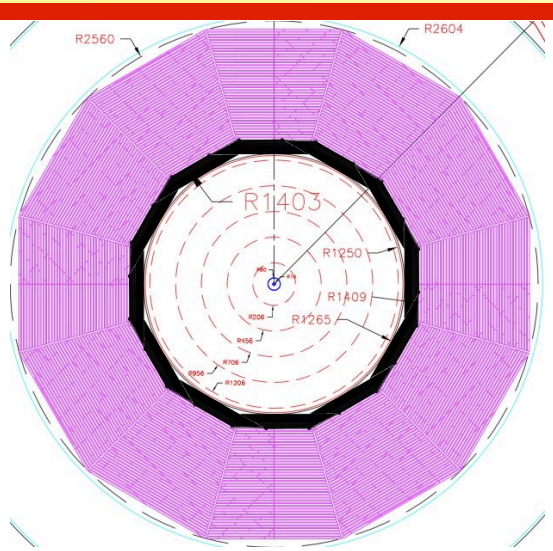
- why are the thicknesses of the kapton and copper layers (components) increasing from Module1 through Module5, while all other module layers stay the same?

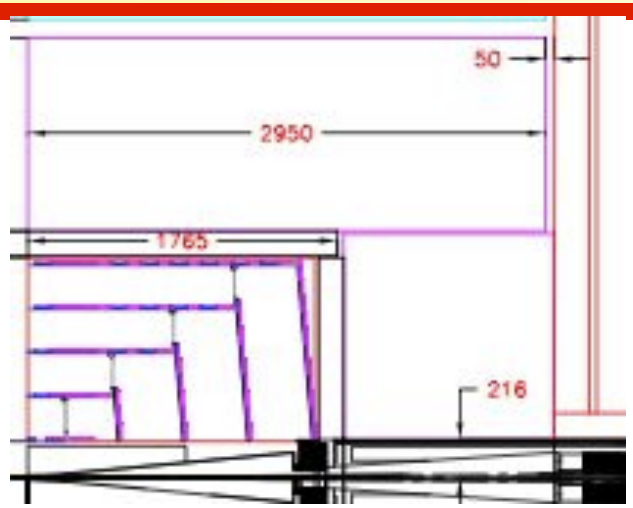
Vertex

Vertex detector so-far unchanged from LOI3 model

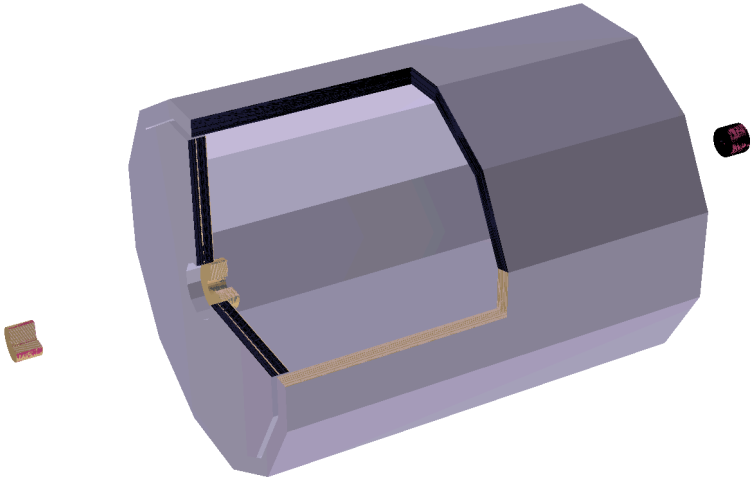


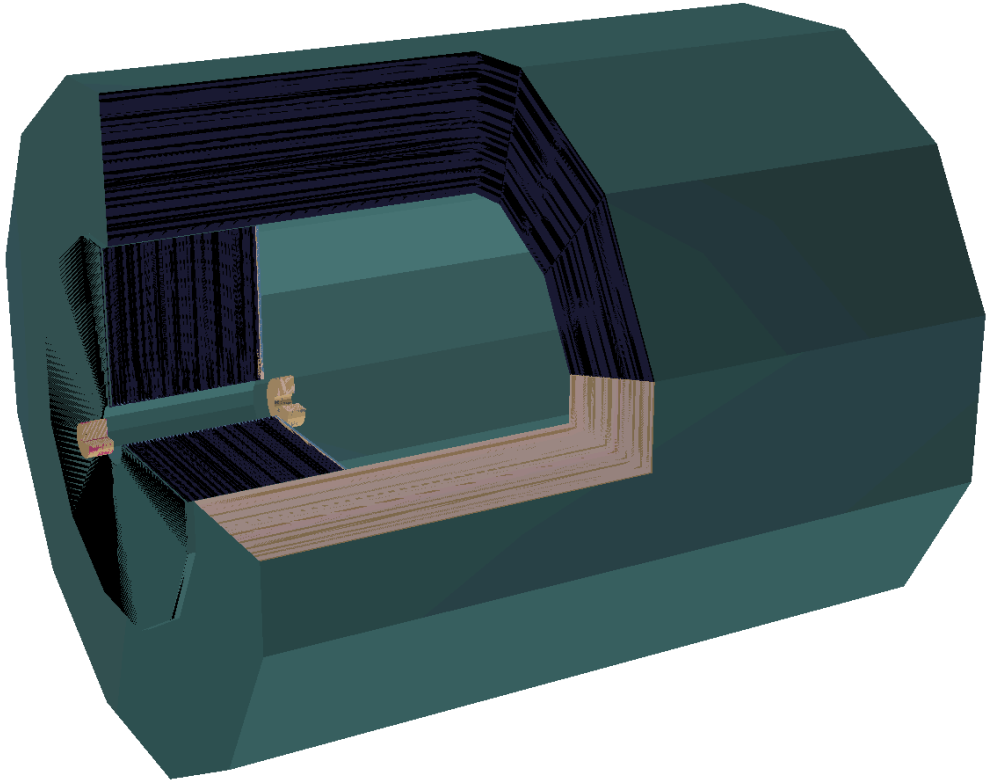
ECAL / HCAL

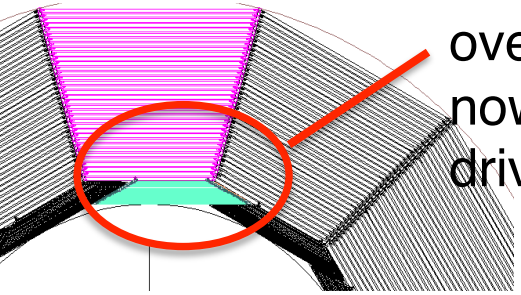




20+10 layer SiW ECAL
40 layer Steel HCAL
-> RPC needs to be updated to scintillator



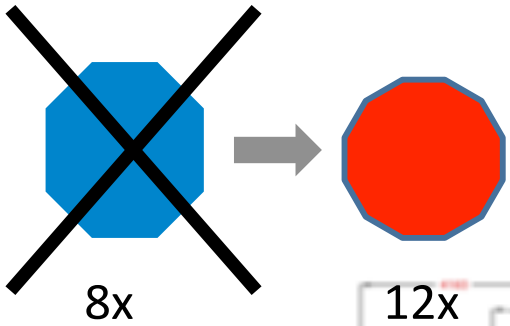


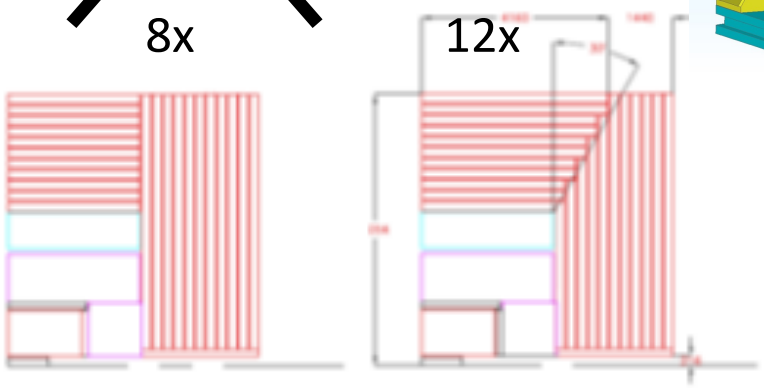


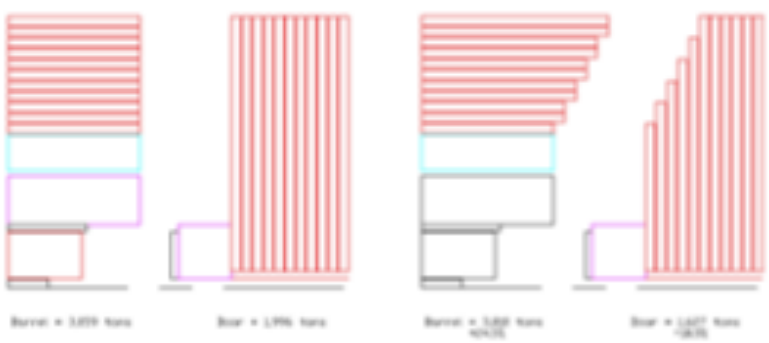
overlap not modelled just now – that is for a custom driver in a subsequent step

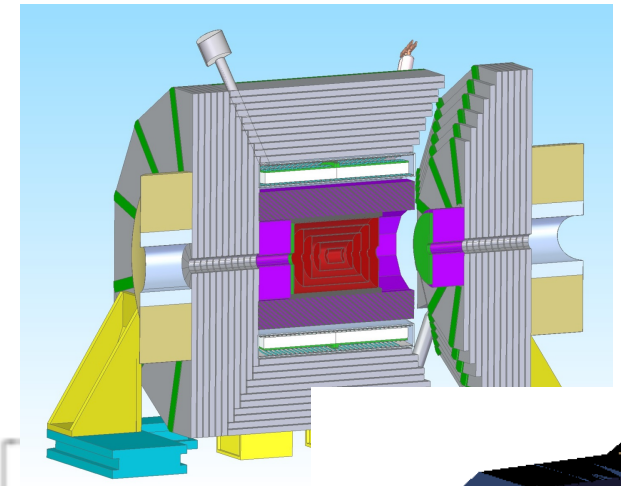
Muon

Yoke/Muon system changes since DBD:



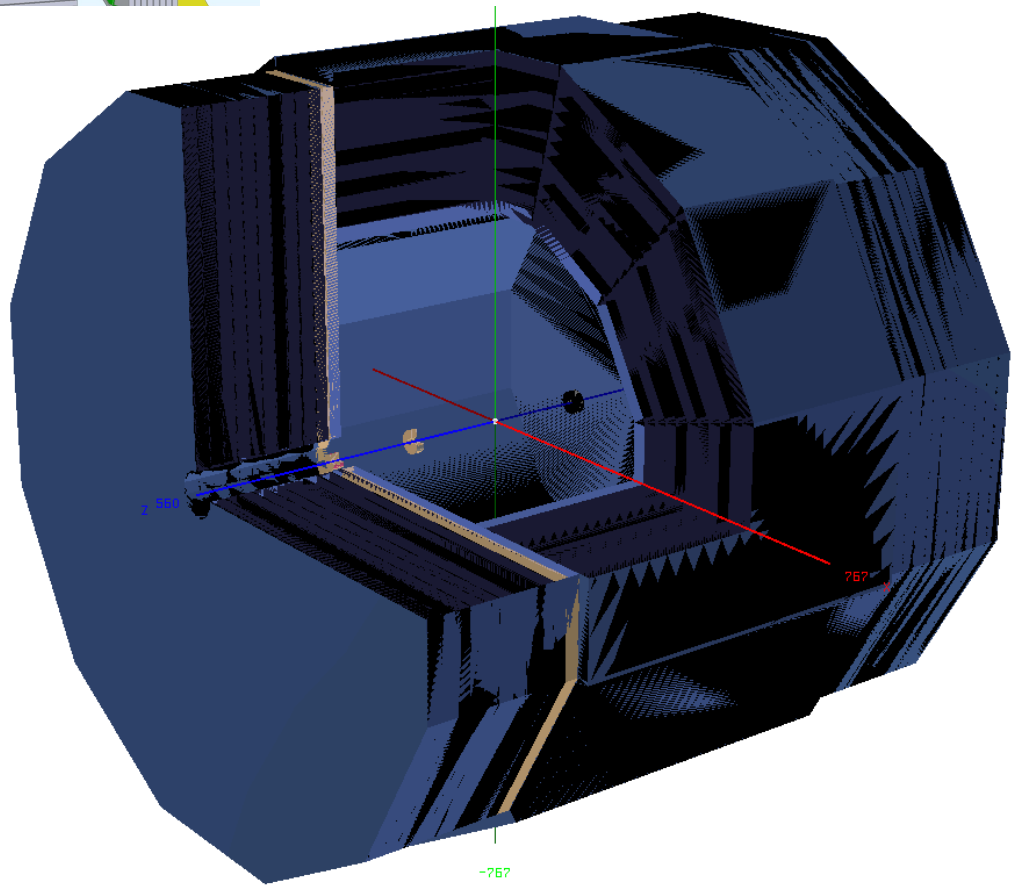






12-sided geometry
implemented

30° barrel/door join
not yet implemented



Driver summary

SiD Subcomponent	XML	Driver
Vertex Barrel	sidloi3/in work	DD4hep_SiTrackerBarrel
Vertex Endcap	sidloi3/in work	DD4hep_SiTrackerEndcap2
Tracker Barrel	SiTrackerBarrel_o1_v01_00	DD4hep_SiTrackerBarrel
Tracker Endcap	SiTrackerEndcap_o1_v01_00	DD4hep_SiTrackerEndcap2 and custom
ECal Barrel	ECalBarrel_o1_v01_00	GGenericCalBarrel_o1_v01
ECal Endcap	ECalEndcap_o1_v01_00	GGenericCalEndcap_o1_v01
HCal Barrel	HCalBarrel_o1_v01_00	GGenericCalBarrel_o1_v01
HCal Endcap	HCalEndcap_o1_v01_00	GGenericCalEndcap_o1_v01
Muon Barrel	MuonBarrel_o1_v01_00	GGenericCalBarrel_o1_v01
Muon Endcap	MuonEncap_o1_v01_00	GGenericCalEndcap_o1_v01
LumiCal	LumiCal_o1_v01_00	DD4hep_CylindricalEndcapCalorimeter
BeamCal	BeamCal_o1_v01_00	DD4hep_ForwardDetector
Beam pipe	sidloi3/in work	DD4hep_PolyconeSupport/TubeSegment
Solenoid	Solenoid_o1_v01_00	Solenoid_o1_v01

■ slic port, ■ updated XML, ■ DD4hep generic drivers, ■ Generic ILD/CLIC/SiD drivers that include Pandora extensions

■ slic port
■ updated XML
■ DD4hep generic drivers,
■ Generic ILD/CLIC/SiD drivers that include Pandora extensions

Digi/Reco

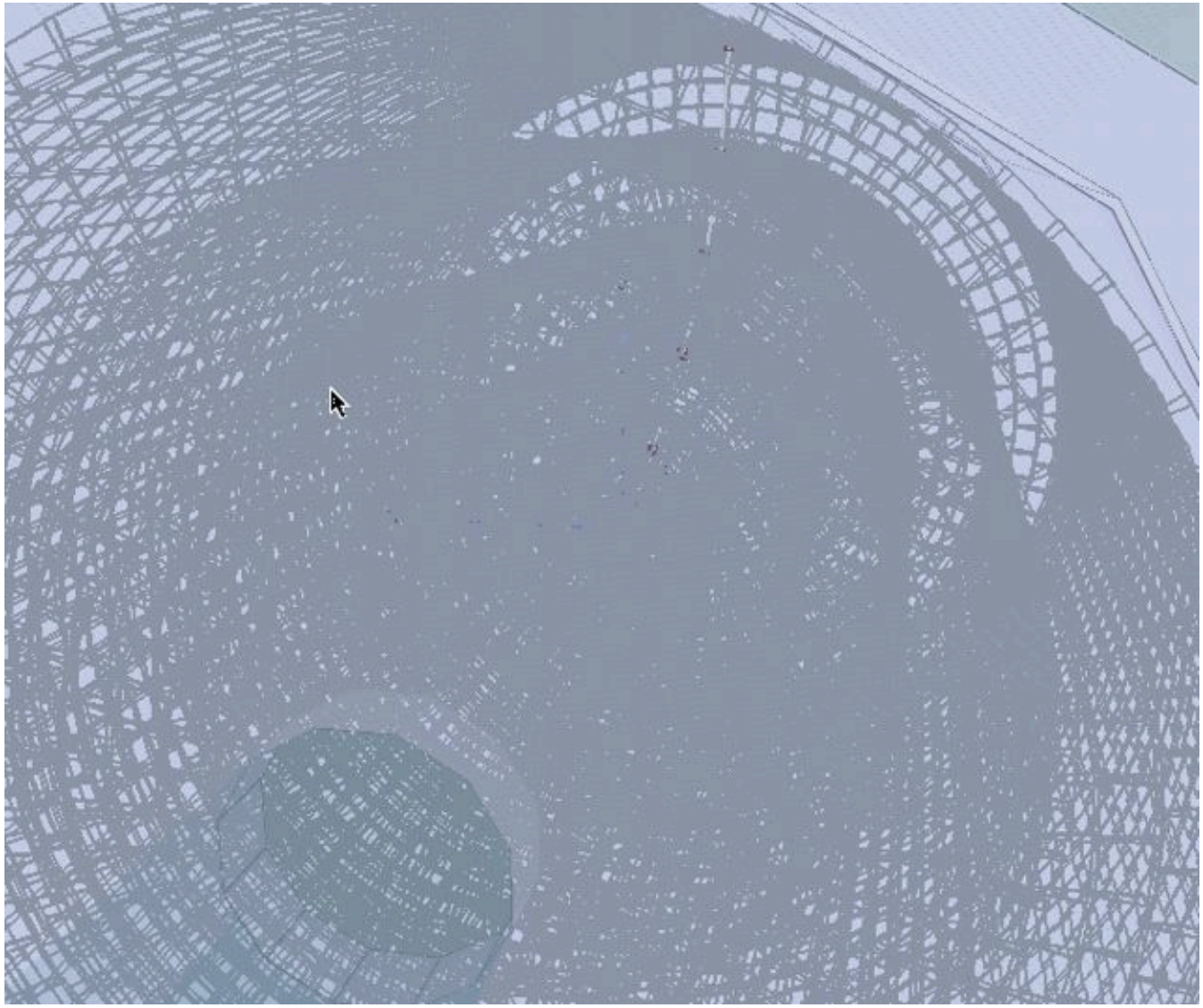
- Using DDPlanarDigiProcessor – working
- Using TruthTrackFinder for now – working.
  - intending to switch to CLIC-style tracking (great to see it converging)
- Using DDCaloDigi, DDSimpleMuonDigi
  - no longer crashing, but not behaving as expected – current work
- Will use DDMarlinPFAProcessor
  - need to write appropriate DDTrackCreator
- After whole chain is working, will need to go back and carefully set digitizers resolution settings, check realistic

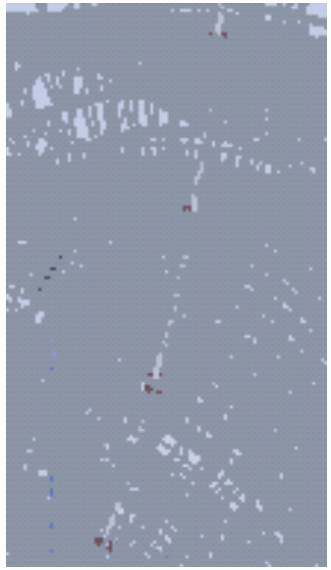
Then a large list:

pandora calibration
vertex-finding
flavour-tagging
pileup/overlay
…

New Glasgow full-time research masters student (Bogdan) starting to prepare benchmarking code

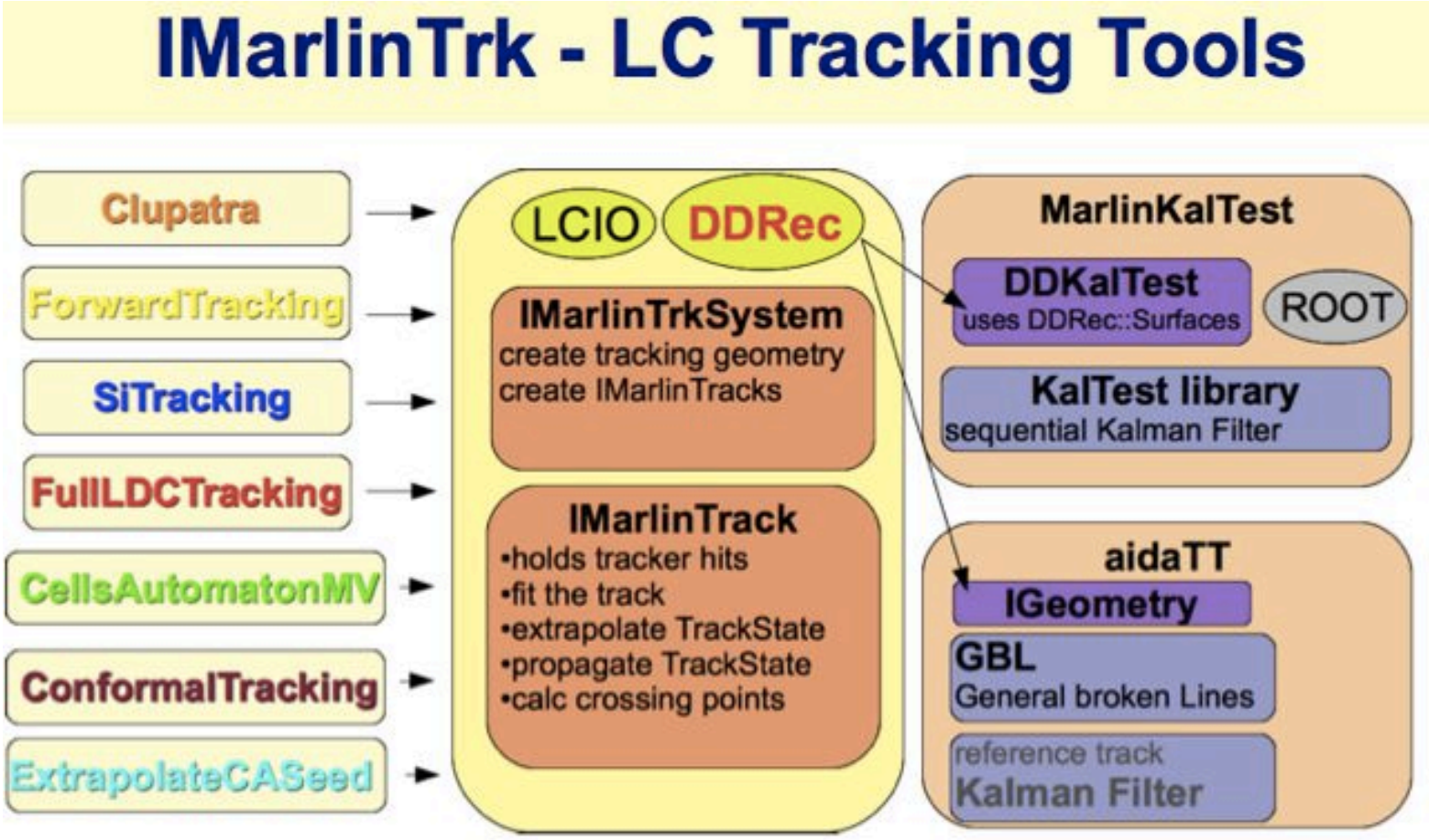
SiTrack





Tracking

See Frank Gaede's tracking talk from Tuesday s/w session



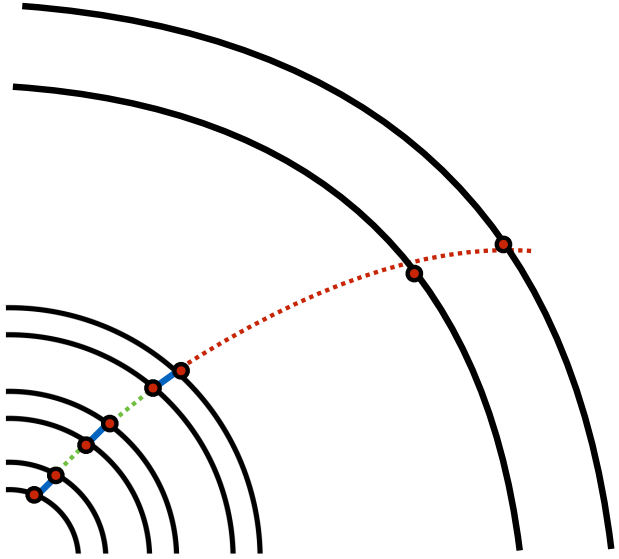
First look by ILD is very encouraging

Tracking

See Daniel Hynds' tracking talk from Tuesday s/w session

Tracking à la ILD - the barrel

- For tracks passing through the barrel region of the vertex detector, reconstruction is a combination of mini-vector generation, cellular automaton and track extrapolation
  - Mini vectors are created connecting two hits in neighbouring vertex barrel layers (taking advantage of the double-layer modules)
  - Cellular automaton is used to produce tracks from these mini vectors
  - These tracks are extrapolated into the tracker barrels and endcaps, with additional hits added to the track
- Recent speed improvements in the extrapolation tool

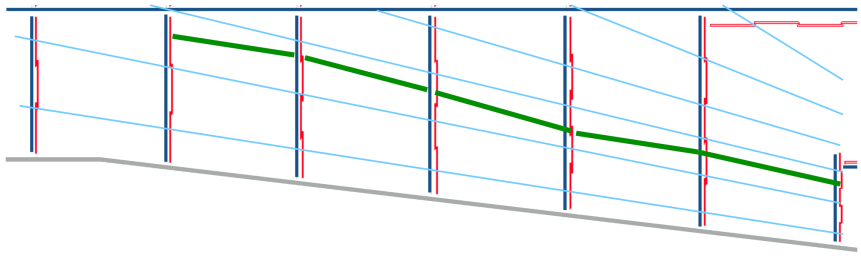


Tracking

See Daniel Hynds' tracking talk from Tuesday s/w session

Tracking à la ILD - the forward direction

- For tracks passing through the vertex endcap, reconstruction is performed using a pure cellular automaton
  - Based on the ILD forward tracking code, using hits from vertex + tracker endcaps
  - Geometry-determined sectors defined in order to reduce combinatorics, take region of $\Delta\phi$, $\Delta\theta$
    - Look for connections between neighbouring sectors
    - Take care of number of layers, neighbour search distance etc.



- Searches then made for cloned tracks
  - Track can be reconstructed both in the barrel and as a forward track - check for common hits
  - Track combination to be improved: try to use common code from ILD
  - So far used simple track merging with removal of the clone tracks

Additional conformal mapping approach being developed

How-to

No more installing of ilcsoft locally – everyone using cvmfs
(your local sysadmin may need to install/mount it on your machine(s))

```
source /cvmfs/ilc.desy.de/sw/x86_64_gcc48_sl6/v01-17-09/init_ilcsoft.sh
source /cvmfs/sft.cern.ch/lcg/releases/gcc/4.8.4/x86_64-slc6/setup.sh
export LCGRELEASES=/cvmfs/sft.cern.ch/lcg/releases/LCG_84
export PYTHONDIR=/cvmfs/sft.cern.ch/lcg/releases/LCG_84/Python/2.7.10/x86_64-slc6-gcc48-opt
export PATH=/cvmfs/sft.cern.ch/lcg/releases/LCG_84/Python/2.7.10/x86_64-slc6-gcc48-opt/bin:$PATH
export LD_LIBRARY_PATH=$PYTHONDIR/lib:$LD_LIBRARY_PATH
export PYTOOLSDIR=/cvmfs/sft.cern.ch/lcg/releases/LCG_84/pytools/1.9_python2.7/x86_64-slc6-gcc48-opt
export PYTHONPATH=/cvmfs/sft.cern.ch/lcg/releases/LCG_84/pytools/1.9_python2.7/x86_64-slc6-gcc48-opt/lib/python2.7/site-packages:
$PYTHONPATH
export PATH=$PYTOOLSDIR/bin:$PATH
```

```
ddsim --compactFile=SiD/compact/SiD_o1_v01/SiD_o1_v01.xml --runType=batch --inputFile
mcparticles.slcio -N=1 --outputFile=testSiD_o1_v01.slcio
```

```
Marlin SiDReconstruction.xml
```

Straightforward to switch to local version of packages that you are developing (need to add instructions to wiki)

How-to

https://twiki.ppe.gla.ac.uk/bin/view/LinearCollider/GlaSiD

GlaSD < LinearCollider < ...

https://twiki.ppe.gla.ac.uk/bin/view/LinearCollider/GlaSiD

University of Glasgow | Experimental Particle Physics

Jump | Search

- LinearCollider
- Log In or Register
- LinearCollider Web
- Create New Topic
- Index
- Search
- Changes
- Notifications
- RSS Feed
- Statistics
- Preferences

Webs
- ATLAS
- PUUKA
- DetDev
- Gridmon
- IT
- LHCb
- LinearCollider
- Main
- NA62
- Sandbox
- TWiki

TWiki > LinearCollider Web > GlaSiD (2016-05-19, DanProtopopescu)

Edit | Raw edit | Attach | Print version

Glasgow SiD howto

The starting point is the **sidloi3** model in `lcgeo` implemented by F. Gaede and based on the original `sidloi3` compact file. This was tested by simulating single muons and reconstructing them (see `lcgeo/example/run_sid_reco.xml`).

- Glasgow SiD howto
  - Building the model
  - Checking simulation+reconstruction chain
  - Implementing custom SiD ECal drivers
  - Repository
  - Adding HCal
  - Adding other subdetectors
  - Troubleshooting

Building the model

The model is using the generic detector drivers in DD4hep. For the calorimeters (Ecal/Hcal) the current (Oct 2015) CLIC drivers were used in order to have the DDRec extensions that will eventually be needed by DDMarlinPandora.

To compile, from the local `lcgeo/` directory:

```
source /cvmfs/sft.cern.ch/lcg/external/gcc/4.8.1/x86_64-slc6-gcc48-opt/setup.sh
export PATH=/cvmfs/sft.cern.ch/lcg/external/Python/2.7.4/x86_64-slc6-gcc48-opt/bin/:$PATH
export LD_LIBRARY_PATH=/cvmfs/sft.cern.ch/lcg/external/Python/2.7.4/x86_64-slc6-gcc48-opt/l
make -C build -j4 install
```

(Let us check/update before you rush off to try it…)

To do / plans

This is where I hoped to be by this workshop ; bit delayed

Top priority: (for AR to do)
Digitization – get working CaloDigi, MuonDigi (top)
Particle flow – write DDTrackCreator and get working DDMarlinPFAProcessor (top)
Commit all to repository, update wiki instructions and release (top)

Following:

Model
Tracker questions (medium)
Look at supports (medium)
Switch HCAL to scintillator (high)
Change to 30º barrel/end angle (low)
ECAL overlapping trapezoids (low)

Digitization
Check resolution parameters etc (high)
Use Daniel Jeans new CaloDigi (?high)
Implement hexagonal readout (medium/low)

Particle flow
Do pandora calibration (high)

Tracking
implement CLIC tracking (high)

High-level tool implementation
vertex-finding (high)
flavour-tagging (high)
– consider what samples needed

Production
pile-up / overlay
liaise with ILCdirac to run on grid
and use same stdhep files as ILD

–> performance plots a la DBD