

GArSoft Updates

ND-GAr: HPgTPC+ECAL Weekly Meeting

28 March 2023

Leo Bellantoni

Shown by Anezka in Collab week

- Two instances of **GENIEGen** can not work in the same GArSoft *art* job.
 - Suppose you were interested, as I am, in studying electrons in the ECAL with overlays. You might create a **GENIEGen** which only sampled the ν_e flux in the TPC and a second instance of **GENIEGen** which had the default generation all over the MPD, including in the ECAL.
 - This does not work. You do not get energy deposits from GEANT from the 2nd instance.
 - Heinous details in Appendix
 - And they ARE heinous. There seems to be something wrong in `TGeoNavigator::SearchNode(...)` for this particular case.
 - This *might* affect LArSoft, as well. I did not know about their use cases though.
 - We are running v6.22 of root; v6.26 exists. `TGeoNavigator` hasn't changed between the versions, but maybe the real issue is in some other, underlying code.

Shown by Anezka in Collab week

- The strip spitter algorithm really wasn't working that well and was producing ghost hits which backtracked to **MCParticles** which were physically far away.
 - Fundamentally, the basic idea is sound. Our strip segmentation is smaller than our resolution from photon timing. But we really need to work out an algorithm that is based on our tiles/layer geometry.
 - I turned it off by default. (Vivek had independently turned it off a long time ago)
- The speed of light in the scintillator was still 1. This meant that the readout simulation put the pulses at the wrong point in time, and the reco exactly cancelled this blunder. But if the pulse's location was reconstructed beyond the end of the strip, it gets pushed to be back inside the strip in reco – that is sensible if the reason for reconstruction beyond the end of the strip is readout noise.

Shown by Anezka in Collab week

- The reconstruction geometry data is in the `gdml` files. The algorithms to manipulate it are in (1) `Geometry/GeometryCore.*` and (2) `Geometry/ChannelMapAlgs/*` where there is a different piece of code for each particular type of geometry. The data and the methods are connected by having the right entries in certain `fcl` files.
 - The code had a hard-wired assumption about what was in the `gdml` file.
 - Another bug, discussed later, is of the same sort; the code has hard-wired assumptions about the data.
 - This is far from an optimal scheme, but a rebuild is a large undertaking.
 - And we are not, I think, ready to settle on our calorimetry geometry for all time. That decision will be very helpful in such a rebuild. We'd only have to get one version to be perfect.

More recent fixes

- A not-very frequent bug in the track vertexing which happened when trying to invert a singular matrix was found. Fix was just to put it into a `try . . . catch` block and not create a vertex.
- Bug reported by Vivek on 21 Feb, where `digiHits` are not reconstructed as `simHits`, was tracked down to octagonal geometry algorithms in `Geometry/ChannelMapAlgs` applied to dodecagonal geometry data on `Geometry/gdml`.
 - But! This might not improve things, as in debugging this I saw that some of the calorimeter cells are not sensibly located in Endcap vs. Barrel. So a fixed algorithm could be exposing more bad data.
- The anatree branch `TrajMCPIndex` was being incorrectly computed. This bug has been there since it was added to the anatree.
- Fixed the fact that the geometry routines `PointInECALBarrel` and `PointInECALEndcap` did not actually check if the point is in the ECAL at all. (It didn't have to before there was MuID).

More recent fixes

- Since we aren't strip splitting, we get hit time from SiPMs to locate the hit. I discovered & fixed a case where time resolution put the hit at the very end of a strip; then `GeometryCore::getStripLength` locates the hit as outside the strip and gives you some other length rather than the strip length, which give you the wrong hit time in the Reco and then the `BackTracker` thinks you've got an out of hit time and fails to track all the cluster energy back to the primary particle.

A bunch of upgrades

- During a momentary epileptic fit in `ifdh`, I switched `GENIEGen.fcl` to get the (November 2017 optimized) optimized flux via direct copy from `/cvmfs/dune.osgstorage.org/pnfs/fnal.gov/usr/dune/persistent/stash/Flux/...` rather than using `ifdh` to take it from `/pnfs/dune/persistent/users/ljf26/...`
- If a proton hit a nucleus in the calorimeter, and that interaction produced a neutron which then traveled through the calorimeter and then made another cluster which you fed to the backtracker... the backtracker used to return the proton. Now it returns the neutron, unless you change a new `fcl` parameter in some file or another
- `BackTracker README.rtf` is updated; `README.md` is gone. but only until somebody decides they want it.

A bunch of upgrades

- Muon ID added to BackTracker!

Some info about how this works:

In `readoutsim`, energy deposits are tagged to the `MCParticle` which creates it.

`BackTrackerCore::ClusterToMCParticles` makes a list of all the `MCParticle` contributions and sorts them largest to smallest.

For each contribution used to take each `MCParticle` and work its way up the tree of MC decays until it found a particle which originated in the gas, using `BackTrackerCore::FindTPCEve`.

Now, if the cluster is in the MuID, it looks for a particle which originates in either the gas or ECAL, using `BackTrackerCore::FindECALeVe`.

A bunch of upgrades

- Added 2 methods to the geometry code, `PointInMuIDBarrel` and `PointInMuIDEndcap`. Their use is pretty obvious.
- I cleaned up some truly perverse indentation in `Geometry/GeometryCore.*` and elsewhere; spiffed up a few comments too.
- The poorly named `MCTrkID` anatree branch is now called `MCPTrkID`.
- An upgrade from Vivek Jain: we now have anatree branches for the calorimeter layer of Sim and Digi hits in both the ECAL and the MuID
- Replaced `/Ana/DSTproduction` with new and much improved `/Ana/ExampleAnatreeUse`. Consider using the (small) analysis framework in this area if you are launching on a new anatree based study.
- I removed the coherent pion analysis code from the anatree. It really doesn't belong there and nobody's using it either.
- Still have the dE/dx code here, though. It does need work before it can be put into the reco code.

Next

1. Tune the clustering parameter using electrons in events with overlays.
2. Spend some time looking at track reco failures.
 1. Maybe we should just throw some packaged algorithm at our tracking?
 2. If the track reco failure level gets low enough, try dE/dx again.
3. Retune the ECAL – reco track matching
4. Take another go at identification of ν_e events
5. As we get results from TOAD re the pulse shape, incorporate them into the TPC simulation.

Appendix of Heinous Details

- Each **GENIEGen** module calls the **GENIEHelper** constructor, from line 204 of the **GENIEGen** constructor. It feed as an argument the geometry manager in the geometry service also created in the **GENIEHelper** constructor. The two calls to the service provider at line 185 of **GENIEGen::GENIEGen** evidently create different services, both of which contain pointers to the ROOT global variable **gGeoManager**, which is defined and initialized in the **TGeoManager** class. I am not sure how our **GeometryCore** creates the (presumably only) instance of the **TGeoManager** class though.
- Then **GENIEGen::beginJob()** calls **GENIEHelper::InitializeGeometry()** twice, once for each generator. It seems to run the same way each time, using the same **gGeoManager** but creating different **GENIEHelper::fGeomD** values for each *art* module.
- Then **GENIEHelper::Sample(...)** is called from **GENIEGen::produce(...)** and seems OK, as far as I can tell. It's called many times for each event; first it is called with the first instance of **GENIEHelper**, and then subsequently with the other instance of **GENIEHelper**.

Appendix of Heinous Details

- When we go to create energy deposits in GArG4, we fail. What happens is that at e.g. line 111 of `EnergyDepositAction::SteppingAction(...)` or similar lines in `AuxDetAction::ECALSteppingAction()`, `AuxDetAction::MuIDSteppingAction()` or other `AuxDetAction` stepping actions, the call to `fGeo->FindNode(...)` doesn't do what it should. The result is that no energy deposits are made and the subsequent readout simulation, reconstruction and analysis jobs have empty outputs.
- The failure is like so: the stepping action method calls the `GeometryCore::FindNode(...)` which is a 1-line wrapper to ROOT global `gGeoManager`. I've gone into the ROOT code in `TGeoManager`, and see that it calls `TGeoNavigator::FindNode(Double_t, Double_t, Double_t)`. At line 1513 of `TGeoNavigator.cxx`, the `TGeoNavigator::SearchNode(...)` method is called and it always returns 0. I can't figure out why. Something with the `fLevel` variable?