

Tracking Embedding -- What to Generate on Node Tree?

David Stewart

2022.06.13

Per Event

- Per event: $n_{\text{track}} \in 1 \dots n_{\text{tracks}}$
- Per n_{track} : n_{g4hit} (track gas ionizations) $\in 1 \dots n_{\text{g4hit}}$ `PHG4HitContainer`
 - `PHG4HitContainer` filled in various `g4detectors/PHG4*SteppingAction.cc`
- Per n_{g4hit} : $n_{\text{electrons}}$ (electron hits – also saved as g4hits , principally on TPC pads), calculated in `PHG4TpcElectronDrift.cc`
 - Question:
 - `PHG4TpcElectronDrift.cc` iterates over all g4hit's in `PHG4HitContainer` from the node tree at one. Is it ok to assume that the g4hits are entered sequentially per track?

For example: while not all g4hits come from embedded tracks, once a g4hit comes from track $n_{\text{track}} = i$ then is it guaranteed that no subsequent g4hit will come from $n_{\text{track}} < i$? In my tests, this is true.

What do we want to save to the node tree from the $n_{\text{electrons}}$'s in the pad planes?

- Previously:
 - *Every drifted* n_{electron} was saved in a multimap of g4hits in the pad plane
 - Possible information of g4hit:

```
virtual float get_x(const int) const {return NAN;}
virtual float get_y(const int) const {return NAN;}
virtual float get_z(const int) const {return NAN;}
virtual float get_px(const int) const {return NAN;}
virtual float get_py(const int) const {return NAN;}
virtual float get_pz(const int) const {return NAN;}
virtual float get_local_x(const int) const {return NAN;}
virtual float get_local_y(const int) const {return NAN;}
virtual float get_local_z(const int) const {return NAN;}
virtual float get_t(const int) const {return NAN;}
virtual float get_edep() const {return NAN;}
virtual float get_eion() const {return NAN;}
virtual float get_light_yield() const {return NAN;}
virtual float get_path_length() const {return NAN;}
virtual unsigned int get_layer() const {return UINT_MAX;}
virtual PHG4HitDefs::keytype get_hit_id() const {return ULONG_LONG_MAX;}

virtual int get_detid() const {return INT_MIN;}
virtual int get_shower_id() const {return INT_MIN;}
virtual int get_scint_id() const {return INT_MIN;}
virtual int get_row() const {return INT_MIN;}
virtual int get_trkid() const {return INT_MIN;}
virtual int get_strip_z_index() const {return INT_MIN;}
virtual int get_strip_y_index() const {return INT_MIN;}
virtual int get_ladder_z_index() const {return INT_MIN;}
virtual int get_ladder_phi_index() const {return INT_MIN;}
virtual int get_index_i() const {return INT_MIN;}
virtual int get_index_j() const {return INT_MIN;}
virtual int get_index_k() const {return INT_MIN;}
virtual int get_index_l() const {return INT_MIN;}
virtual int get_hit_type() const {return INT_MIN;}
```

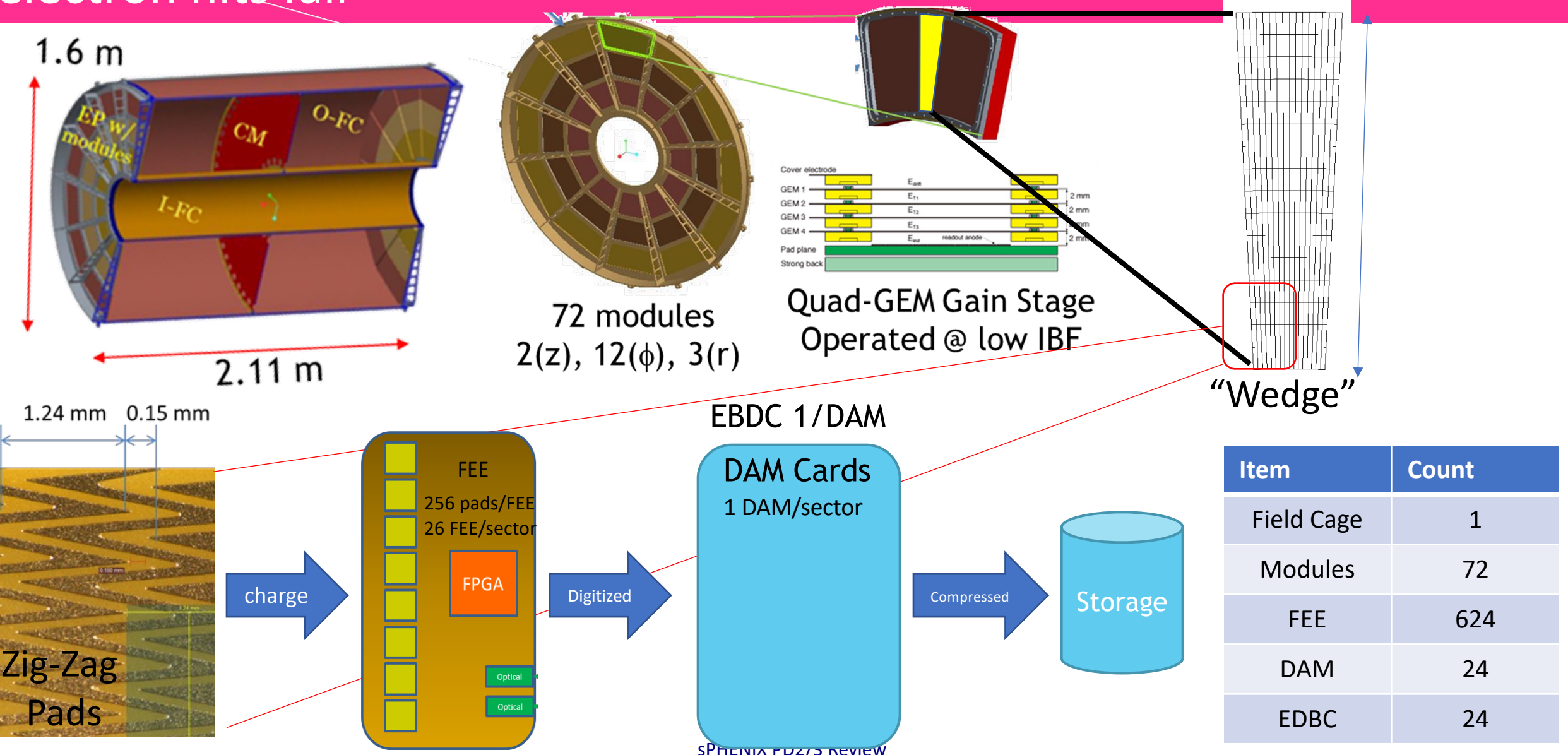
Depending on virtual members implemented, far too much information for far too many individual electron hits on the pads; too slow with a multimap in reconstruction and track matching!

Minimal, and very naïve answer

- Calculate mean and standard deviation of 3-space coordinates for each ionizing n-hit (i.e. of the resulting $n_{\text{electrons}}$ from the electron drift and amplification) : $\mu_x, \sigma_x, \mu_y, \sigma_y, \mu_z, \sigma_z$
 - This can be stored as an array<double, 6> per ionizing g4hit (the data of an `TrkrHitTruthClusters` object)
- Per track, build a vector of `TrkrHitTruthClusters`, one for each ionizing g4hit
- Per event, store a map of "track-id -> vector(`TrkrHitTruthClusters`)"

Very small amount of information and fast. However, not helpful for lookup in embedding the electron pad g4hits into the data itself

When reconstructing tracks, interested in where in the pads the electron hits fall



What information to actually save to node tree?

- Smaller and simpler is better
- Presumably must interface well with `simulation/g4simulation/g4tpc/PHG4TpcPadPlaneReadout.cc`
- Therefore:
 - `offline/packages/tpc/TpcDefs.h` has function to generate `TrkrDefs::hitkey`: 32 bits – 16 for time bin, 16 for pad id
- Do we want vectors of `TrkrDefs::hitkeys` pointing back to the originating embedded tracks? (perhaps with a weighting for electrons per pad?)
- If there are many electrons per pad, do we wish to save mean and variance of ϕ and z of electrons within the pad instead of individual electrons, or can we measure that granularity?