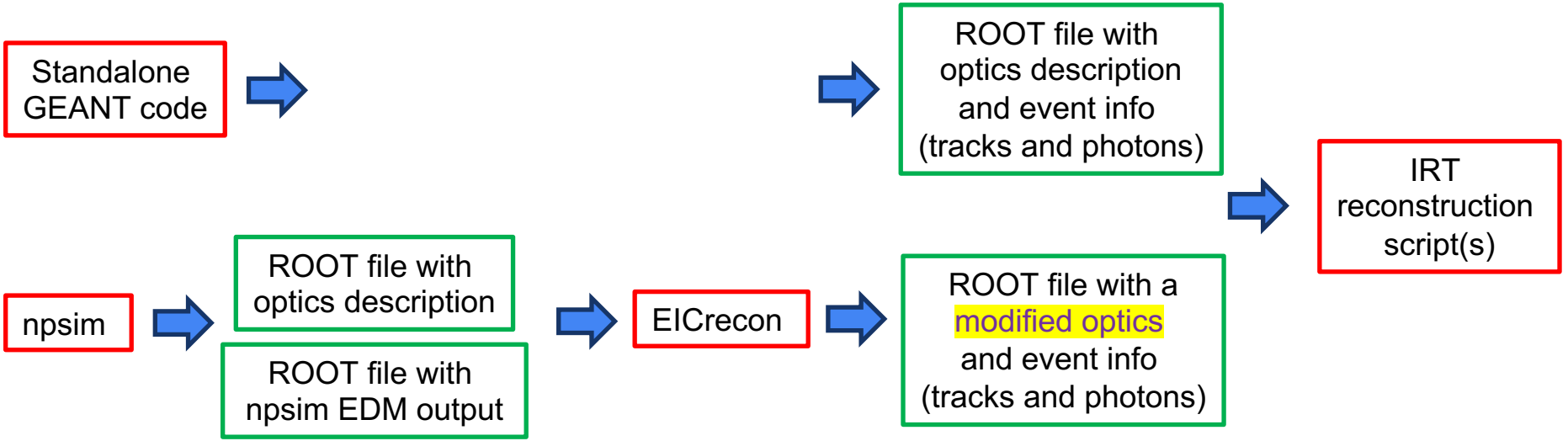
Update on IRT 2.0 implementation in EICrecon

Alexander Kiselev (BNL)

ePIC pfRICH DSC weekly meeting, April 17, 2025

Intermediate setup for debugging



- Path shown in the bottom chart is the present debugging stage
- A simplified detector (codename QRICH) used as a primary testbed

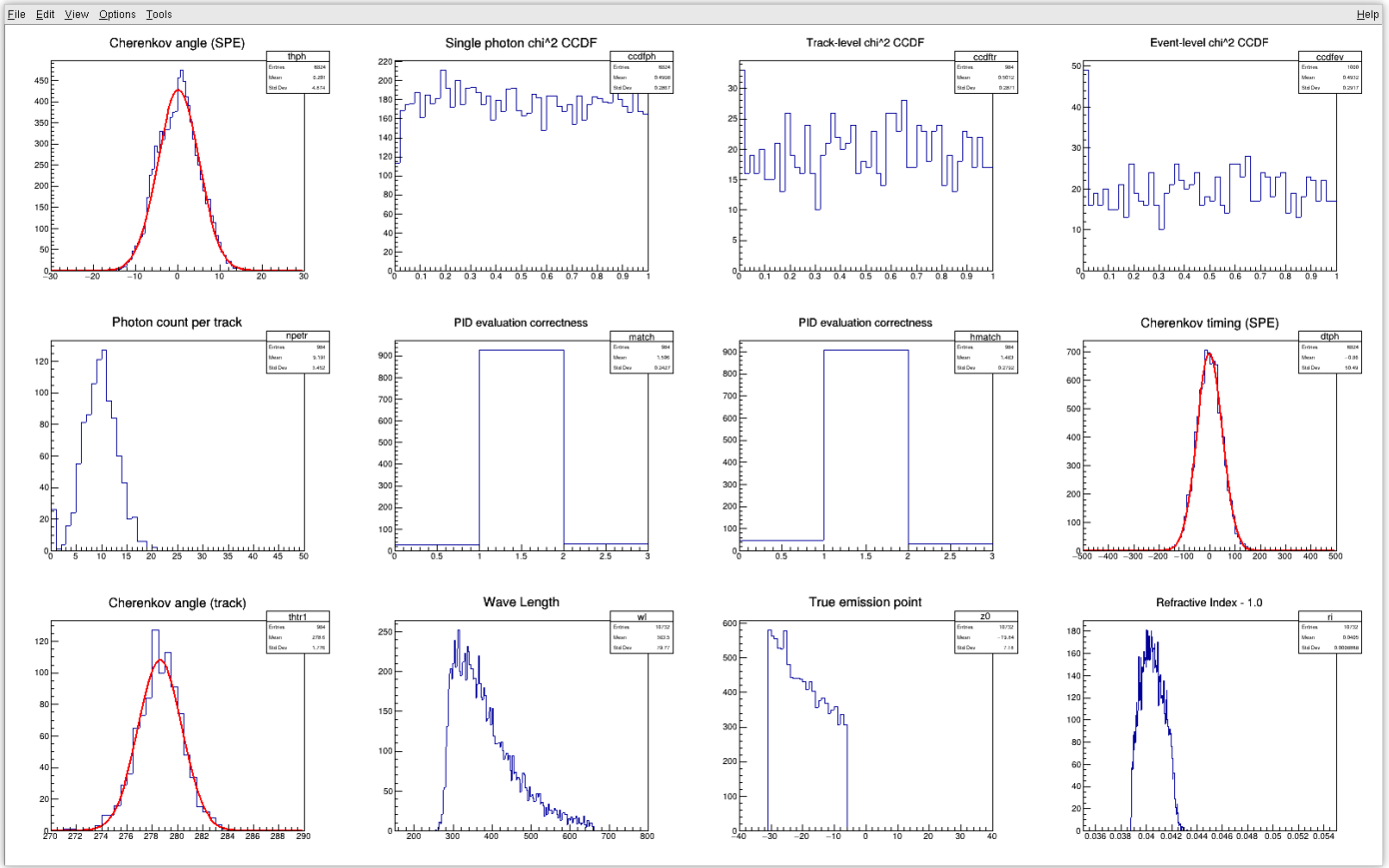
Present status of *QRICH* implementation

- IRT 2.0 reconstruction factory input emulated in full (tracks, photons), including
  - <n> ~ 1.040 aerogel properties
  - HRPPD QE
  - Acrylic filter material as used by dRICH

-> Getting expected photon count

- Event tree exported into a ROOT file which a standalone .C script can process
- Geometry extensions
  - 9x9 real size HRPPD photon matrix
  - An outer cylindrical mirror to imitate reflections and a multi-path configuration
- Visualization
  - Make use of GEANT Qt display functionality in eic-shell

Present status of *QRICH* implementation



A “benchmark” panel produced by a post-processing script

Towards a generalized pfRICH/dRICH implementation

- C++ and XML geometry files are of course (Q,PF,D)RICH detector specific
  - Example: https://github.com/eic/epic/blob/irt-2.0/src/PFRICH_geo.cpp

- A unified RICH-IRT plugin [(Q,PF,D)RICH agnostic]
  - https://github.com/eic/EICrecon/blob/irt-2.0/src/detectors/RICH-IRT/RICH-IRT.cc
  - Looks for EICrecon command-line keys *-PQ(PF,D)RICH:config=<JSON config file>*
    - Be aware that plugin name should not coincide with a detector name for this to work

- A partly unified (and detector specific) JSON configuration file
  - QRICH example: https://github.com/eic/EICrecon/blob/irt-2.0/sandbox/qrich-reco.json
  - Mandatory: IRT optics input file, radiator command line style options, photosensor QE
  - Whatever else may need to be turned on/off and / or configured in IRT 2.0 pass
  - Optional and / or detector specific stuff: TBD [HRPPD vs SiPM digitization?]

Towards a generalized pfRICH/dRICH implementation

- A unified IRT 2.0 algorithm / factory interface [(Q,PF,D)RICH agnostic]
  - https://github.com/eic/EICrecon/blob/irt-2.0/src/algorithms/pid/IrtDebugging.h
  - https://github.com/eic/EICrecon/blob/irt-2.0/src/algorithms/pid/IrtDebugging.cc
  - https://github.com/eic/EICrecon/blob/irt-2.0/src/global/pid/IrtDebugging_factory.h

- Neither IrtGeo nor ActsGeo geometry services or their equivalent used [for now]
  - IRT optics is imported as a ROOT C++ class instance image
  - RICH-IRT.cc builds a projection plane request for ACTS engine on its own

- A sparse README.md: https://github.com/eic/EICrecon/tree/irt-2.0/sandbox

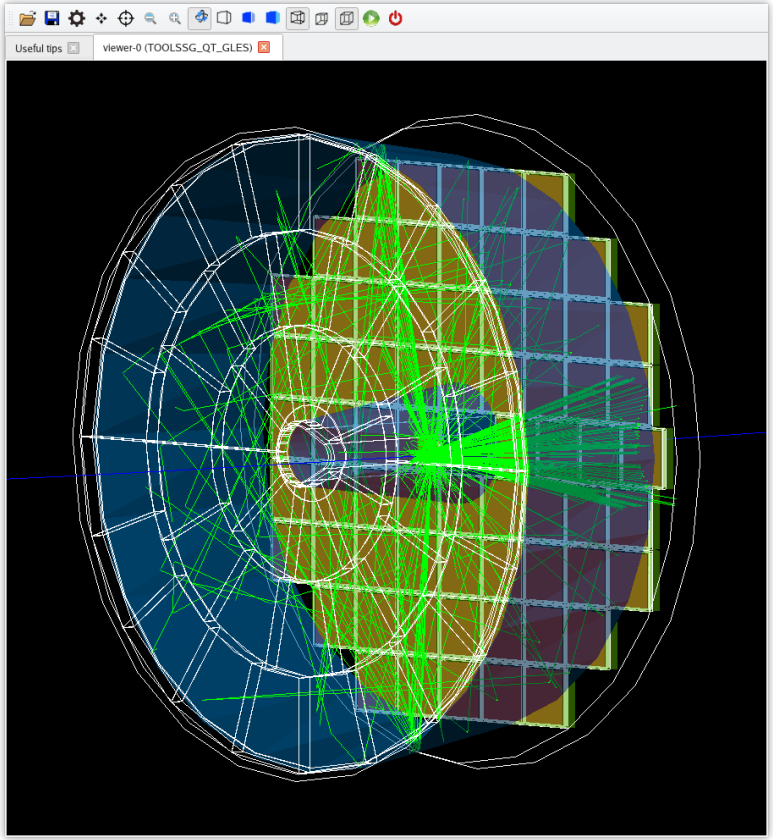
Present status of *pfRICH* implementation

- Geometry was in a usable starting point state as ported by Bill last year
  - Fixed a number of small bugs and now able to see MC hits
  - Fixed a bug in mirror description and now see reflected photons as well
  - Attached a proper aerogel and other materials
  - Partly re-organized the code structure to attach cell index tags

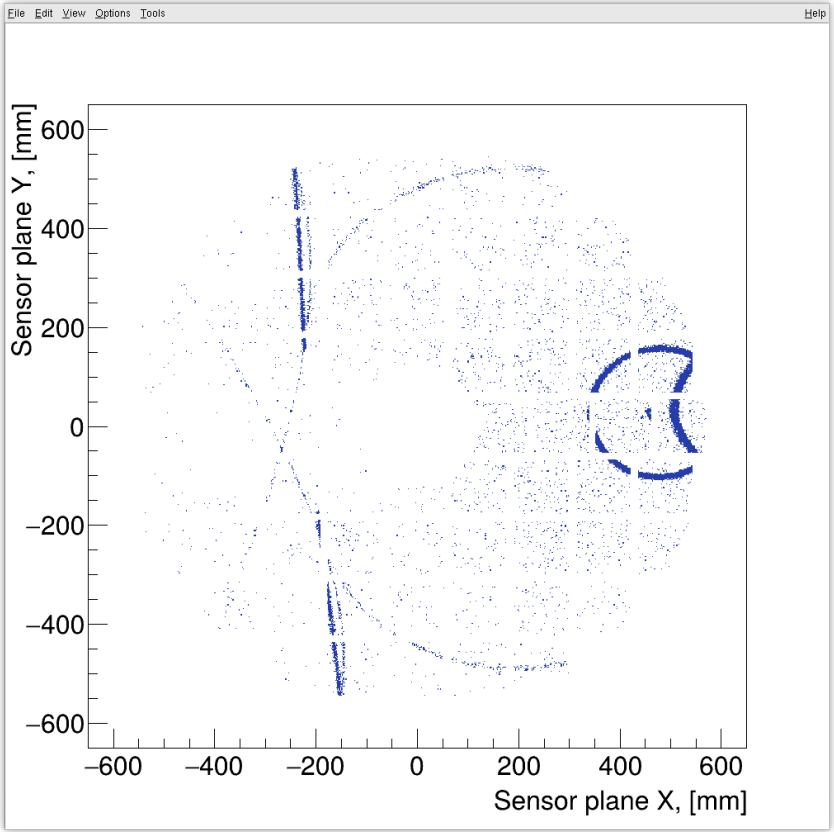
- Optics by now coded for direct hits only
  - Extension to a conical mirror reflection a la QRICH must be straightforward
  - Pyramid mirrors are missing all together though (not even described in the geometry)

- EICrecon plugins: same RICH-IRT.cc and IrtDebugging.cc as for QRICH
  - Able to see digitized hits (next slide)

Present status of *pfRICH* implementation



GEANT4 Qt display in eic-shell



1000 events accumulated (digitized hits)

Next steps

- Implement a single spherical mirror in QRICH
  - To confirm that IRT 2.0 (no sampling along the trajectory) works for dRICH gas radiator
- Confirm that using physical volumes (dd4hep implementation) rather than {logical volume & parent physical volume copies} in the geometry tree (pfRICH standalone code implementation) for sensitive volume elements (cell IDs) does not pose an issue for more complicated geometries (like dRICH sectors) with IRT algorithm

-> At this point we are sure the essential codes can be shared between pfRICH & dRICH

- pfRICH conical & pyramid mirrors implementation

- Calibration pass (and calibration data import in EICrecon)
- Standalone post-processing scripts -> integrate into IrtDebugging.cc plugin
- Represent IRT 2.0 digitization as an intermediate plugin
- Start thinking of a data model