



Introduction to Fun4All for CORE

Jin Huang

Brookhaven National Lab

Fun4All EIC simulation and reconstruction

Concept from detector team
(an example with 1.5T magnet shown here)

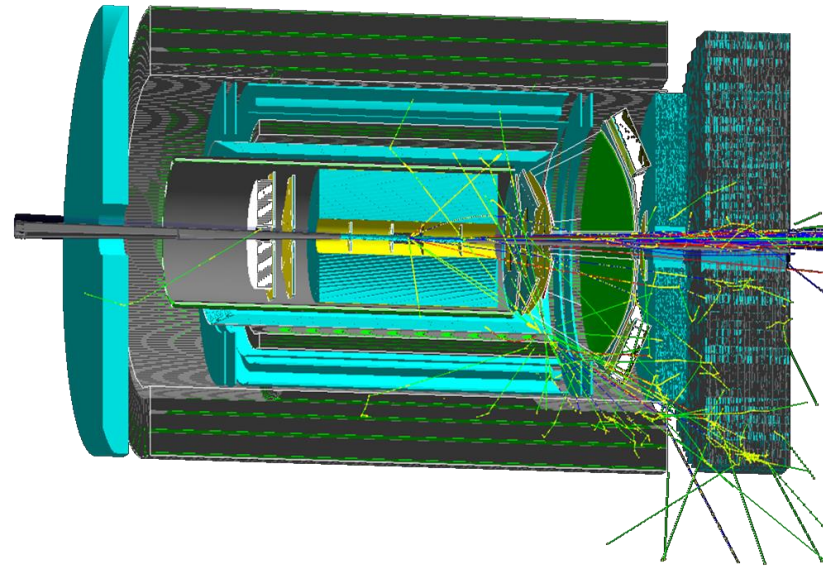


Fun4All sim. and reconstruction

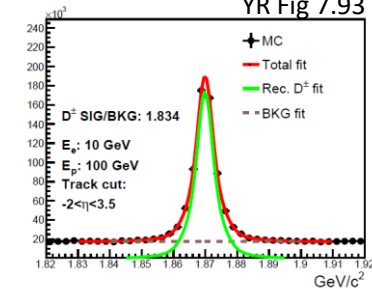


Validating performance

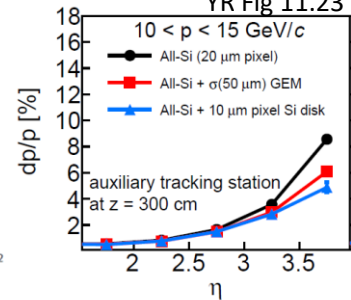
Yellow Report (YR) Fig 11.65



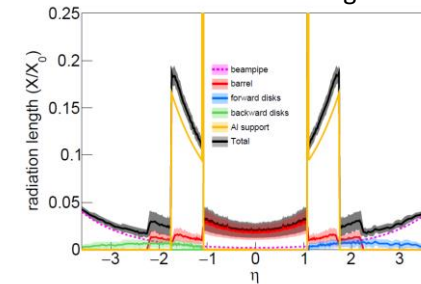
YR Fig 7.93



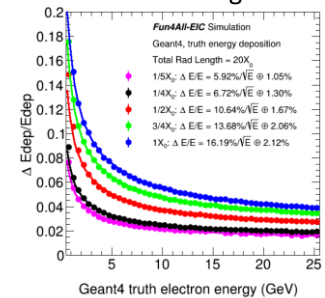
YR Fig 11.23



YR Fig 11.73



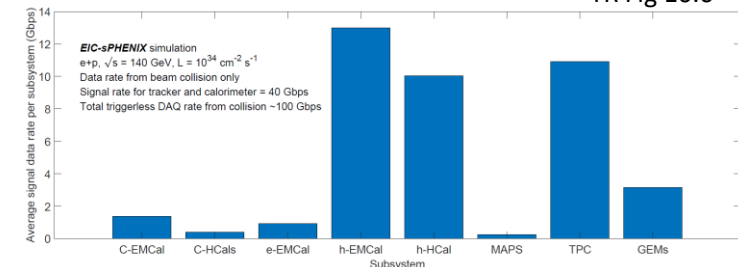
YR Fig 11.55



Note:

- Figures of a generic EIC detector are used for illustration purpose
- Non-collaboration specific yellow report figures shown
- Also used in sim+reco. in stages of proposal preparation for all three EIC proto-collaborations

YR Fig 10.6

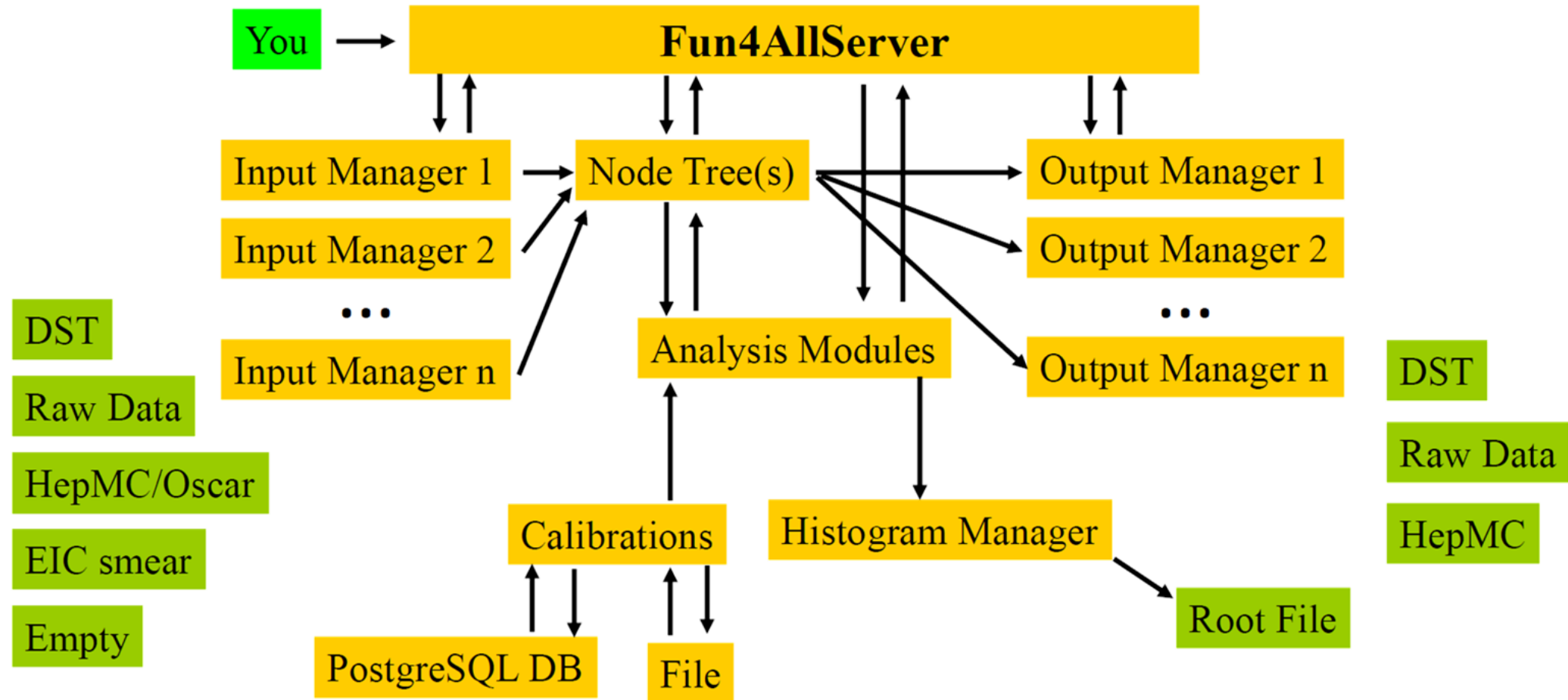


Fun4All is ... the glue that binds them all

Fun4All is

- ▶ Built to handle 100PB real data and 1B event simulation annually
- ▶ C++1x-based framework and integrate many commonly used software packages, e.g. Pythia6/8, Geant4, ROOT, GenFit2, ACTS, FastJet, KFParticle
- ▶ The analysis is a continuous chain from the event generator/raw data up to analysis objects, e.g. jet reconstruction; steering flow of program using ROOT Cling macros
- ▶ Internal Node Tree our storage for data objects, support make snapshots at any state of the reconstruction/analysis
- ▶ Many IO formats, embedding, pileup, access to calibrations
- ▶ Continuously distributed over CVMFS and containerized

Fun4All framework: the glue that binds them all



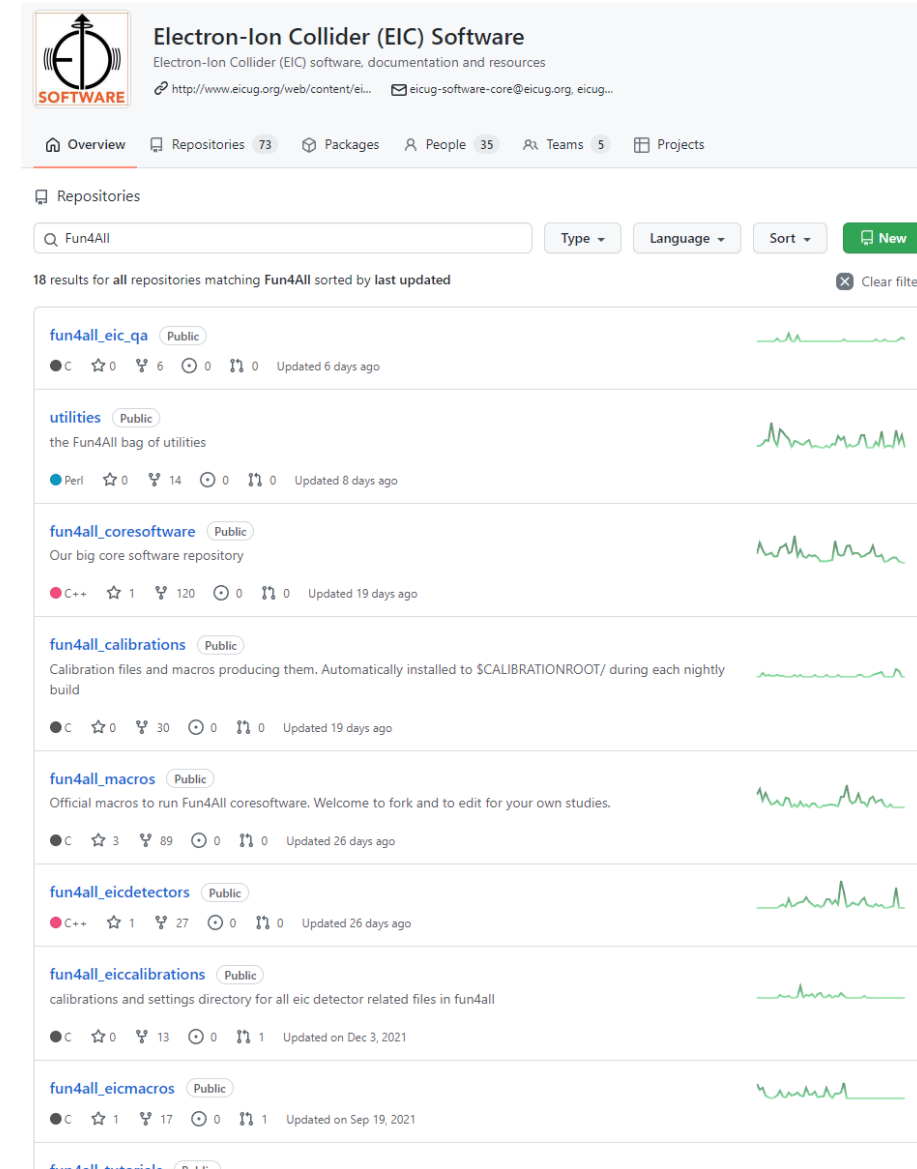
That's all there is to it, no backdoor communications – steered by ROOT macros

Fun4All-EIC Software Infrastructure



Repository organization

- ▶ EIC UG GitHub host a fork of Fun4All for EIC general use
 - Open-source software repository
 - Mirrored core-software from sPHENIX + EIC specific dev
 - Note: [sPHENIX](#) and [ECCE](#) use dedicated GitHub organization to have full control of repos
- ▶ EIC detectors built on top of Fun4All:
 - [fun4all_eicdetectors](#) [daily build, require pull request]: G4 detector description, reco. Module
 - [macros](#) [daily build, require pull request]: Detectors description and common macros
 - [fun4all_eiccalibrations](#) [daily build, require pull request]: calibration file, geometry descriptions



The screenshot shows the GitHub organization page for "Electron-Ion Collider (EIC) Software". The page displays a list of repositories under the "Repositories" tab. The search bar contains "Fun4All", and there are 18 results. The repositories listed are:

- [fun4all_eic_qa](#) (Public): Updated 6 days ago
- [utilities](#) (Public): the Fun4All bag of utilities, Updated 8 days ago
- [fun4all_coresoftware](#) (Public): Our big core software repository, Updated 19 days ago
- [fun4all_calibrations](#) (Public): Calibration files and macros producing them. Automatically installed to SCALIBRATIONROOT/ during each nightly build, Updated 19 days ago
- [fun4all_macros](#) (Public): Official macros to run Fun4All coresoftware. Welcome to fork and to edit for your own studies, Updated 26 days ago
- [fun4all_eicdetectors](#) (Public): Updated 26 days ago
- [fun4all_eiccalibrations](#) (Public): calibrations and settings directory for all eic detector related files in fun4all, Updated on Dec 3, 2021
- [fun4all_eicmacros](#) (Public): Updated on Sep 19, 2021

EIC build distribution

- ▶ Daily build and distribute via OpenScienceGrid CVMFS
- ▶ In place where sourced `eic_setup.sh` or `sphenix_setup.sh`:

```
singularity shell -B /cvmfs:/cvmfs /cvmfs/eic.opensciencegrid.org/singularity/rhic_sl7_ext.simg  
source /cvmfs/eic.opensciencegrid.org/default/opt/fun4all/core/bin/eic_setup.sh -n
```
- ▶ Use out of box in major NP/HEP scientific computing centers
 - Example to use at Jlab: https://ecce-eic.github.io/tutorials_example2_JLab.html#run-the-simulation-in-batch-mode-in-example-2a
- ▶ Install in your institution Linux sever or laptop via Linux VM:
<https://github.com/EIC/Singularity>
- ▶ Same container can be used for EIC-Smear fast simulation [e.g. EIC-Smear]

<https://github.com/EIC/Singularity>

Singularity container for EIC Fun4All

Singularity container for EIC Fun4All allows any user to run the EIC RCF/SDCC environment with the nightly builds on your local computers or on external high-performance computing clusters.

This repository is part of the [software tutorial](#), in particular for users offsite to [the BNL RACF computer center](#). This repository includes the instruction and local update macro for this Singularity container, which ensures binary reproducible simulation and reconstruction.

Daily validations: `updatebuild.sh --build=new` build passing

[standard macros](#) [git](#) [tutorials](#) [git](#) [code reference](#) [Doxygen](#) [last commit](#) [june 2021](#)

How do I navigate around all this mountain of code?

Start w/ search anything here

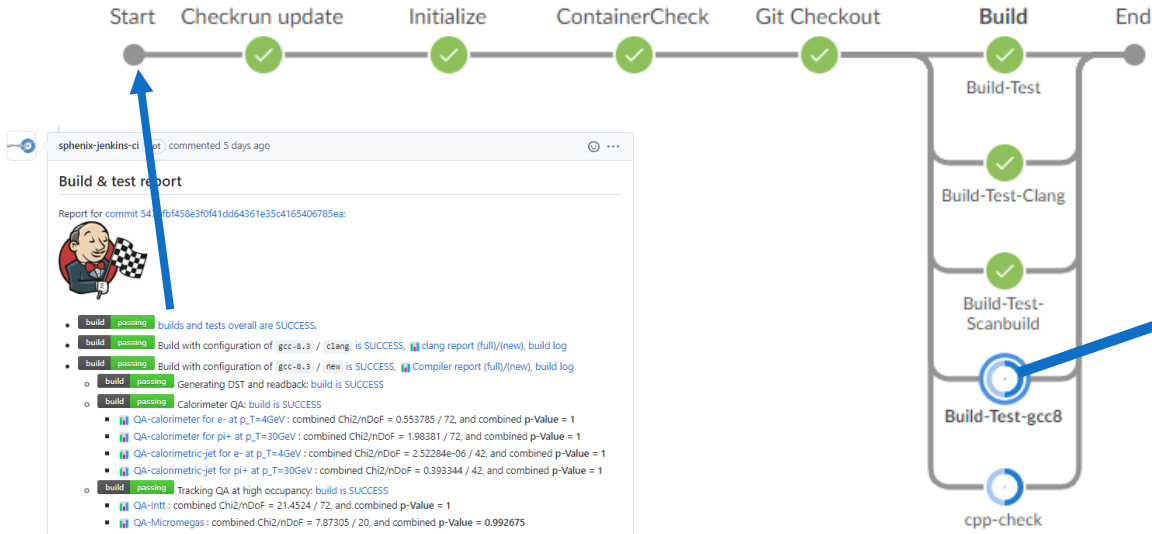
- ▶ eic.github.io/doxygen/
- ▶ All software on GitHub now digested in Doxygen via JenkinsCI
- ▶ Auto built at change of any repository
- ▶ Search anything in Doxygen site for [EIC GitHub software](#)

ECCE @ EIC Software
Reference for ECCE @ EIC simulation and reconstruction software on GitHub

The screenshot displays the Doxygen interface for the ECCE @ EIC Software project. The top navigation bar includes 'Home page', 'Related Pages', 'Modules', 'Namespaces', 'Classes', 'Files', and 'External Links'. A search bar is located in the top right corner. The left sidebar shows a 'Class List' with various classes, including 'RawClusterBuilder', 'RawClusterContainer', 'RawClusterDeadAreaMask', 'RawClusterPositionCorrection', 'RawClusterUtility', 'RawClusterv1', 'RawTower', 'RawTowerBuilder', 'RawTowerBuilderByHitIndex', 'RawTowerCalibration', 'RawTowerCombiner', 'RawTowerContainer', 'RawTowerDeadMap', 'RawTowerDeadMapLoader', 'RawTowerDeadMapv1', 'RawTowerDeadTowerInterp', 'RawTowerDigitizer', 'RawTowerGeom', 'RawTowerGeomContainer', 'RawTowerGeomContainer_Cylinderv1', 'RawTowerGeomContainerv1', 'RawTowerGeomv1', 'RawTowerGeomv2', 'RawTowerGeomv3', 'RawTowerv1', 'rcdaqEventIterator', 'ReadEICFiles', 'RealQuadraticEquation', 'recoConsts', and 'RecursiveMomentumContainer'. The 'ReadEICFiles' class is selected and highlighted. The main content area shows the 'ReadEICFiles Class Reference' page. It includes an '#include' statement for 'ReadEICFiles.h' and an 'Inheritance diagram for ReadEICFiles' showing 'ReadEICFiles' inheriting from 'SubsysReco' and 'PHHepMCGenHelper', which both inherit from 'Fun4AllBase'. Below the inheritance diagram is a 'Collaboration diagram for ReadEICFiles' and a section for 'Public Member Functions' listing methods like 'Init', 'process_event', 'OpenInputFile', 'SetFirstEntry', and 'SetNodeName'. There are also sections for 'Public Member Functions inherited from SubsysReco', 'Public Member Functions inherited from Fun4AllBase', and 'Public Member Functions inherited from PHHepMCGenHelper'.

Fun4All Jenkins-CI Pipeline Workflow

- CI check in upstream sPHENIX-Fun4All repo



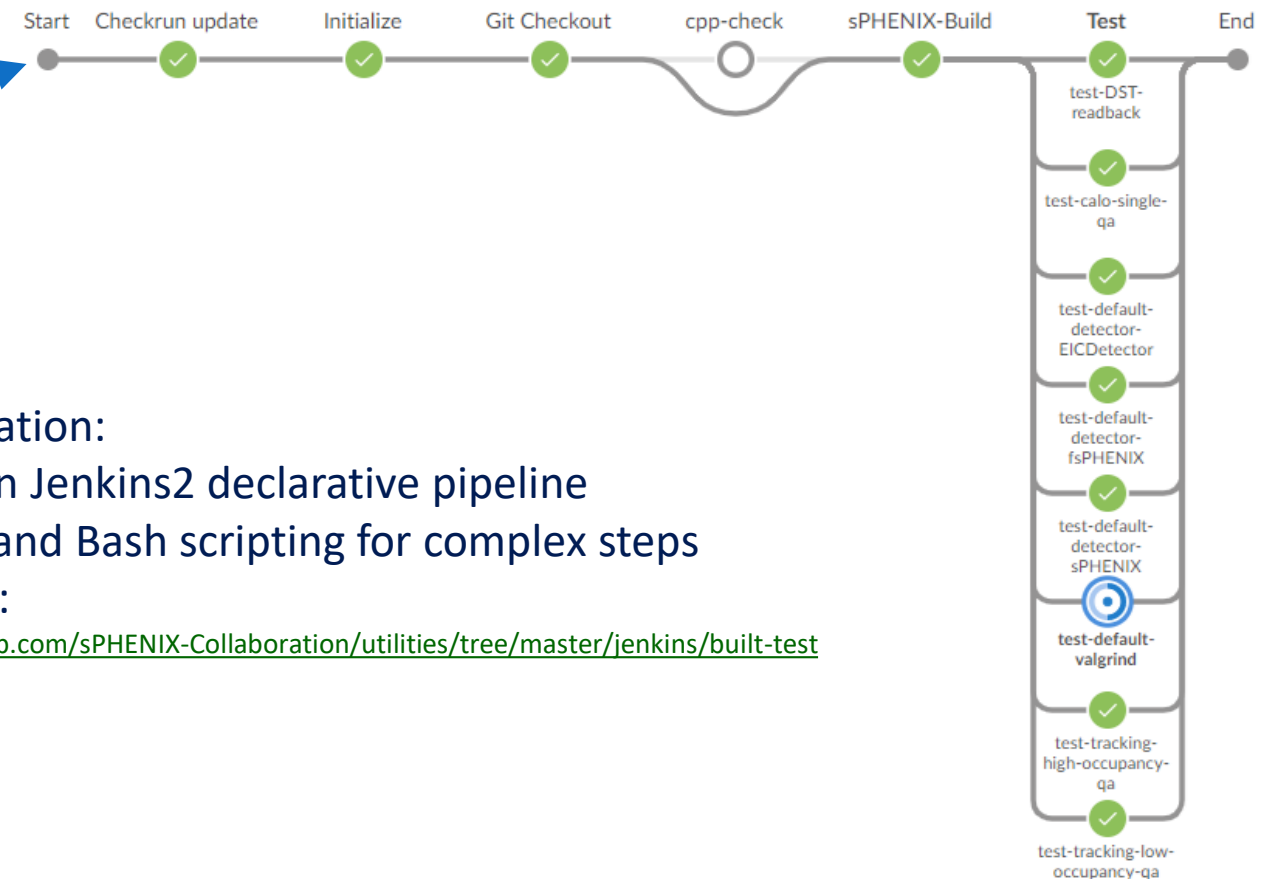
sphenix-jenkins-ci bot commented 5 days ago

Build & test report

Report for commit 5411bf459e3f0f41dd64361e35c4165406785ee:

- build passing builds and tests overall are SUCCESS.
- build passing Build with configuration of gcc-8.3 / clang is SUCCESS, clang report (full)(new), build log
- build passing Build with configuration of gcc-8.3 / new is SUCCESS, Compiler report (full)(new), build log
- build passing Generating DST and readback: build is SUCCESS
 - build passing Calorimeter QA: build is SUCCESS
 - QA-calorimeter for e- at p_T=4GeV: combined Chi2/nDoF = 0.553785 / 72, and combined p-Value = 1
 - QA-calorimeter for pi+ at p_T=30GeV: combined Chi2/nDoF = 1.98381 / 72, and combined p-Value = 1
 - QA-calorimetric-jet for e- at p_T=4GeV: combined Chi2/nDoF = 2.52284e-06 / 42, and combined p-Value = 1
 - QA-calorimetric-jet for pi+ at p_T=30GeV: combined Chi2/nDoF = 0.393344 / 42, and combined p-Value = 1
 - build passing Tracking QA at high occupancy: build is SUCCESS
 - QA-Intt: combined Chi2/nDoF = 21.4524 / 72, and combined p-Value = 1
 - QA-Micromegas: combined Chi2/nDoF = 7.87305 / 20, and combined p-Value = 0.992675
 - QA-Mvtx: combined Chi2/nDoF = 19.9297 / 54, and combined p-Value = 0.999994
 - QA-Tpc: combined Chi2/nDoF = 26.1392 / 56, and combined p-Value = 0.999778
 - QA-tracking: combined Chi2/nDoF = 16.4744 / 38, and combined p-Value = 0.99909
 - QA-vertexing: combined Chi2/nDoF = 52.7677 / 98, and combined p-Value = 0.999948
 - build passing Tracking QA at low occupancy: build is SUCCESS
 - QA-Intt: combined Chi2/nDoF = 15.9328 / 72, and combined p-Value = 1
 - QA-Micromegas: combined Chi2/nDoF = 17.9183 / 20, and combined p-Value = 0.592789
 - QA-Mvtx: combined Chi2/nDoF = 6.85174 / 54, and combined p-Value = 1
 - QA-Tpc: combined Chi2/nDoF = 28.7617 / 56, and combined p-Value = 0.999058
 - QA-tracking: combined Chi2/nDoF = 22.6706 / 42, and combined p-Value = 0.993553
 - QA-vertexing: combined Chi2/nDoF = 86.0741 / 98, and combined p-Value = 0.799809
 - build passing Tracking QA for Pythia DO-jet: build is SUCCESS
 - QA-Intt: combined Chi2/nDoF = 13.7902 / 72, and combined p-Value = 1
 - QA-Micromegas: combined Chi2/nDoF = 11.1369 / 20, and combined p-Value = 0.942594
 - QA-Mvtx: combined Chi2/nDoF = 20.3097 / 54, and combined p-Value = 0.999992
 - QA-Tpc: combined Chi2/nDoF = 63.5889 / 56, and combined p-Value = 0.226809
 - QA-tracking: combined Chi2/nDoF = 18.7187 / 36, and combined p-Value = 0.992242
 - QA-vertexing: combined Chi2/nDoF = 53.804 / 98, and combined p-Value = 0.999917
 - build passing system gcc-8.3, build new: run the default sPHENIX macro: build is SUCCESS, output
 - build passing system gcc-8.3, build new: run the overlap check for sPHENIX macro: build is SUCCESS, output
 - build unstable system gcc-8.3, build new: Valgrind test build is UNSTABLE, valgrind report
 - build passing Build with configuration of gcc-8.3 / scan is SUCCESS, scan-build report (full)(new), build log
 - build passing cpp-check is SUCCESS, cppcheck report (full)(new)

Automatically generated by sPHENIX Jenkins continuous integration



Implementation:

- Based on Jenkins2 declarative pipeline
- Python and Bash scripting for complex steps
- Git SCM: <https://github.com/sPHENIX-Collaboration/utilities/tree/master/jenkins/built-test>

<https://github.com/sPHENIX-Collaboration/coresoftware/pull/1415#issuecomment-1036911557>

Software modules



Event generators

- ▶ Running example given in sample EIC sim macro:
https://github.com/ECCE-EIC/macros/blob/master/detectors/EICDetector/Fun4All_G4_EICDetector.C
- ▶ Arbitrary set-of-particle generators for testing
- ▶ Main event generator input format: HepMC files
- ▶ Built-in generators : PYTHIA8.3, PYTHIA6, SARTRE
- ▶ Reads events generated in EIC-Smear package [talk: Kolja]
 - Share the same Singularity image and CVMFS vol. Source different setup macros.
 - Allow comparison of same event sample processed by EIC-Smear and by the Fun4All simulation-reco.
- ▶ Support multiple background pile ups

Note: All EIC full event generator has EIC beam parameters applied by default: see next section

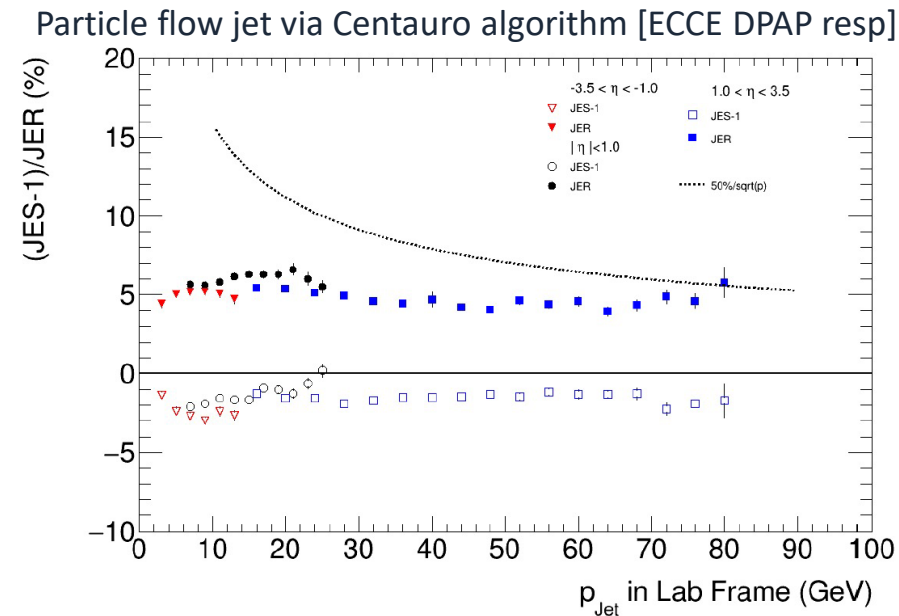
<https://eic.github.io/software/mcgen.html>

Monte Carlo Event Generators

- PYTHIA6
- BeAGLE
- DJANGO
- MILOU
- RAPGAP
- PEPSI
- eSTARlight (external link)
- Sartre (external link)

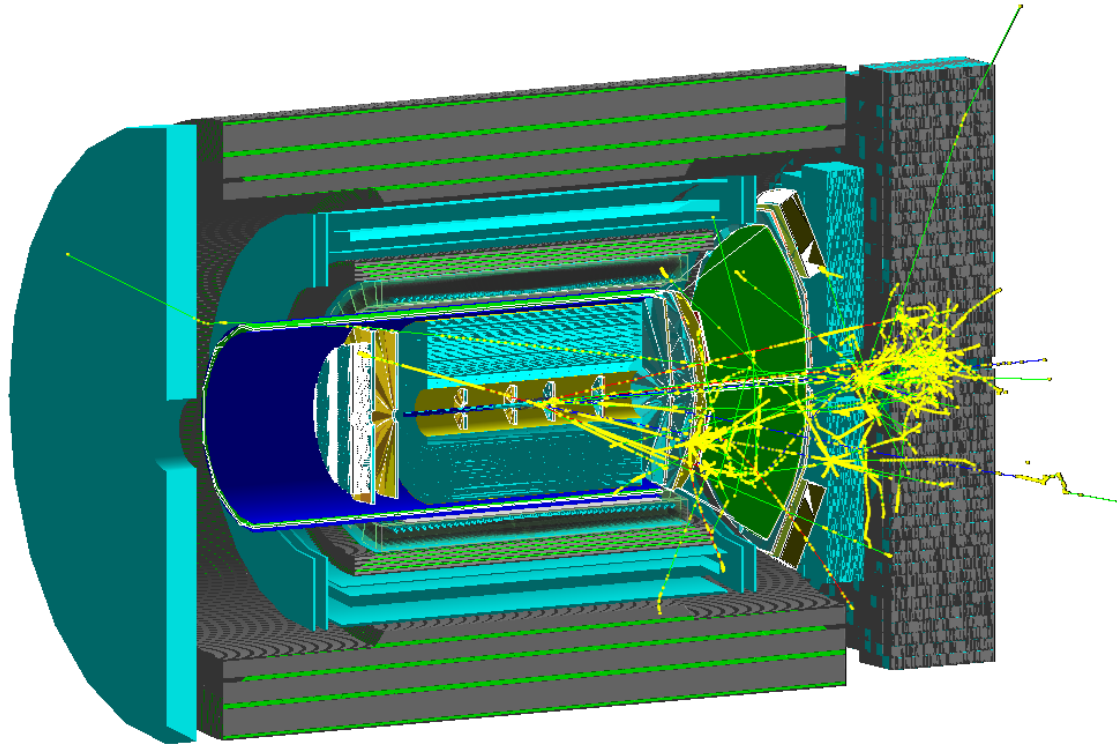
Example Reconstruction Modules

- ▶ Tracking:
 - Use [GenFit2 for fast prototyping \(PHG4TrackFastSim\)](#), widely used in YR tracking studies
 - ACTS comes built in and use by sPHENIX for speed gain. Not a real issue for EIC though
- ▶ Calo reco:
 - Many choices of [clusterizers](#), [FastJet](#), Particle flow jet via Centauro algorithm
- ▶ PID reco:
 - Loglikelihood-based PID interface : <https://indico.bnl.gov/event/13060/contributions/54740/attachments/37240/61349/HadronPID.pdf>
 - GenFit2 for TOF, e.g. track length and timing smearing
- ▶ Resonance search/HF reconstruction:
 - [KFParticle](#) , HF jet tagging
- ▶ Detailed truth tracing evaluation chain
 - e.g. what [portion of reco jet is from a truth jet](#)



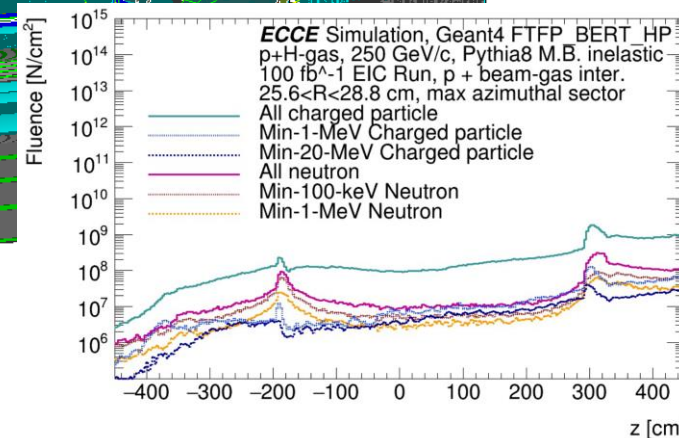
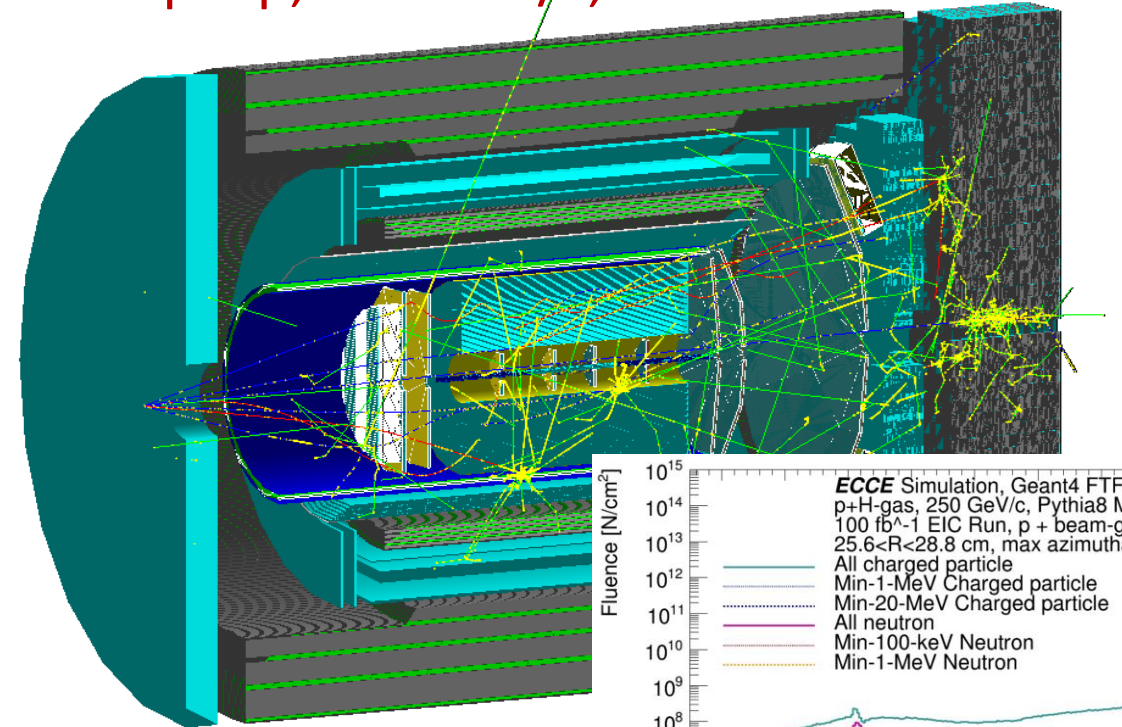
Background embedding: beam gas interaction

- ▶ Built-in Pythia8 for beam gas interaction background generation



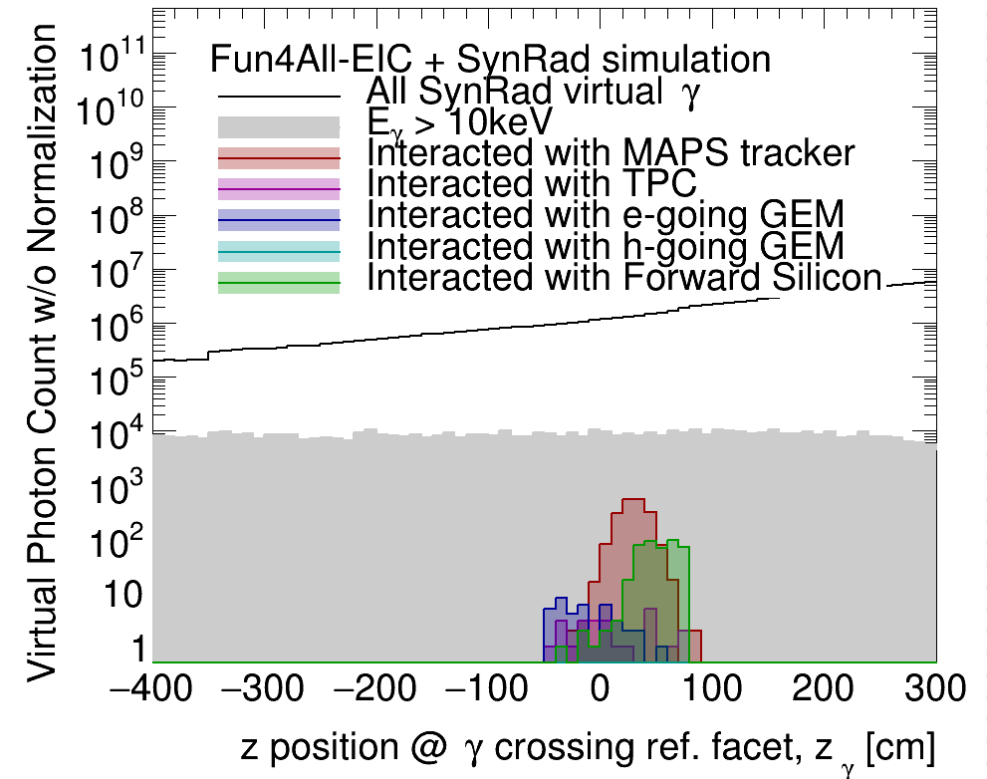
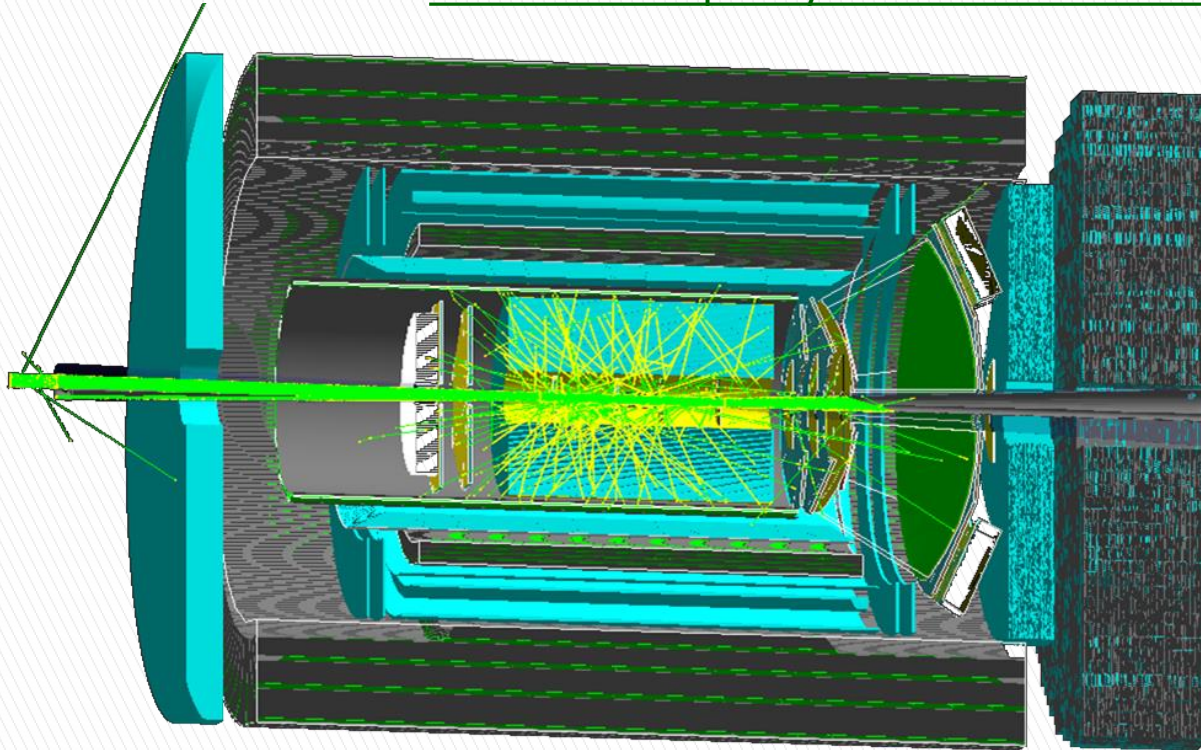
$e+p$ DIS 18+275 GeV/c
 $Q^2 \sim 100$ (GeV/c)²

Beam gas event
 $p + p$, 275 GeV/c, at $z = -4$ m



Background embedding : Synchrotron radiation

- Fun4All has [interface to input Synchrotron Photon simulation](#), used in Synchrotron study leading to CD-1 review



100k Synchrotron photon in full detector simulation [YR Fig 10.12]

Detector background as function of beam-pipe exit-location

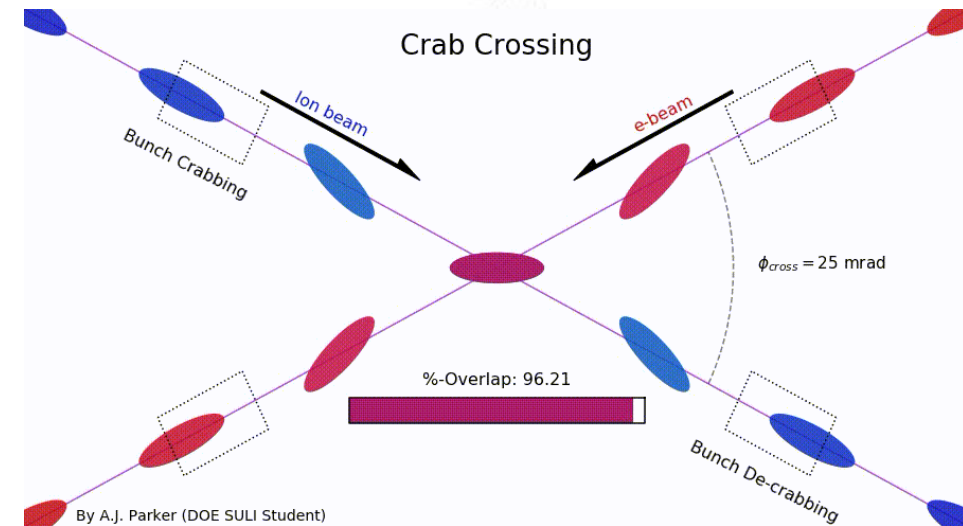
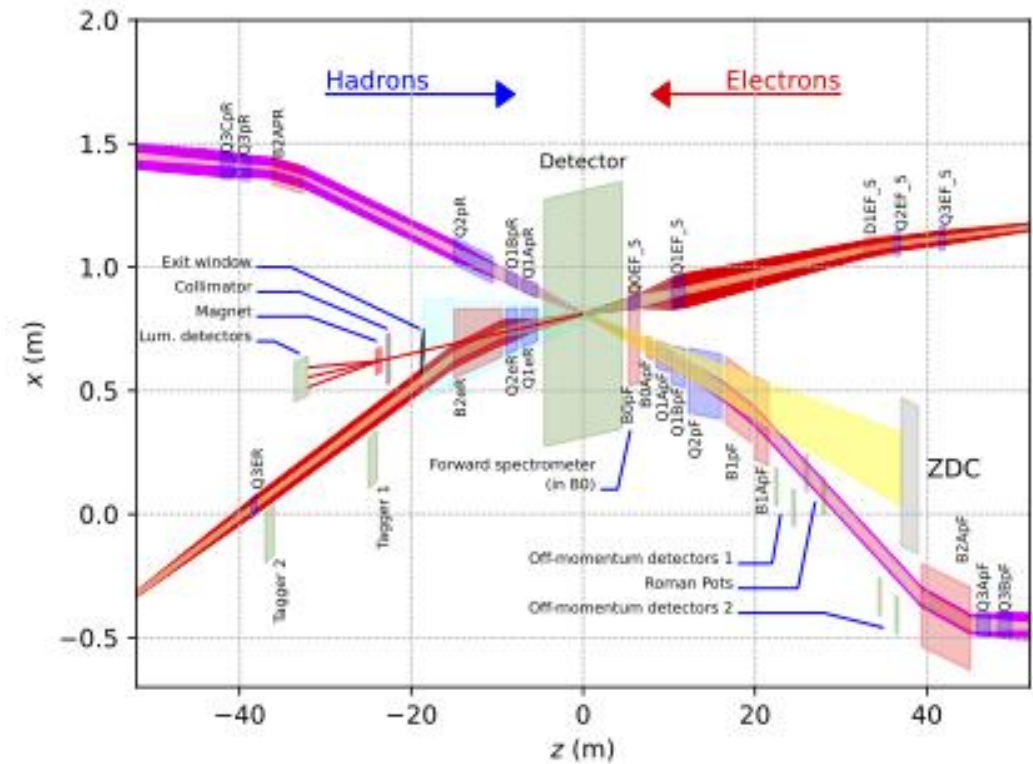
Note: all photons simulated for detector interaction, without cuts on z or energy. EIC/July-2020 lattice & chamber

Beam crossing afterburner in Fun4All



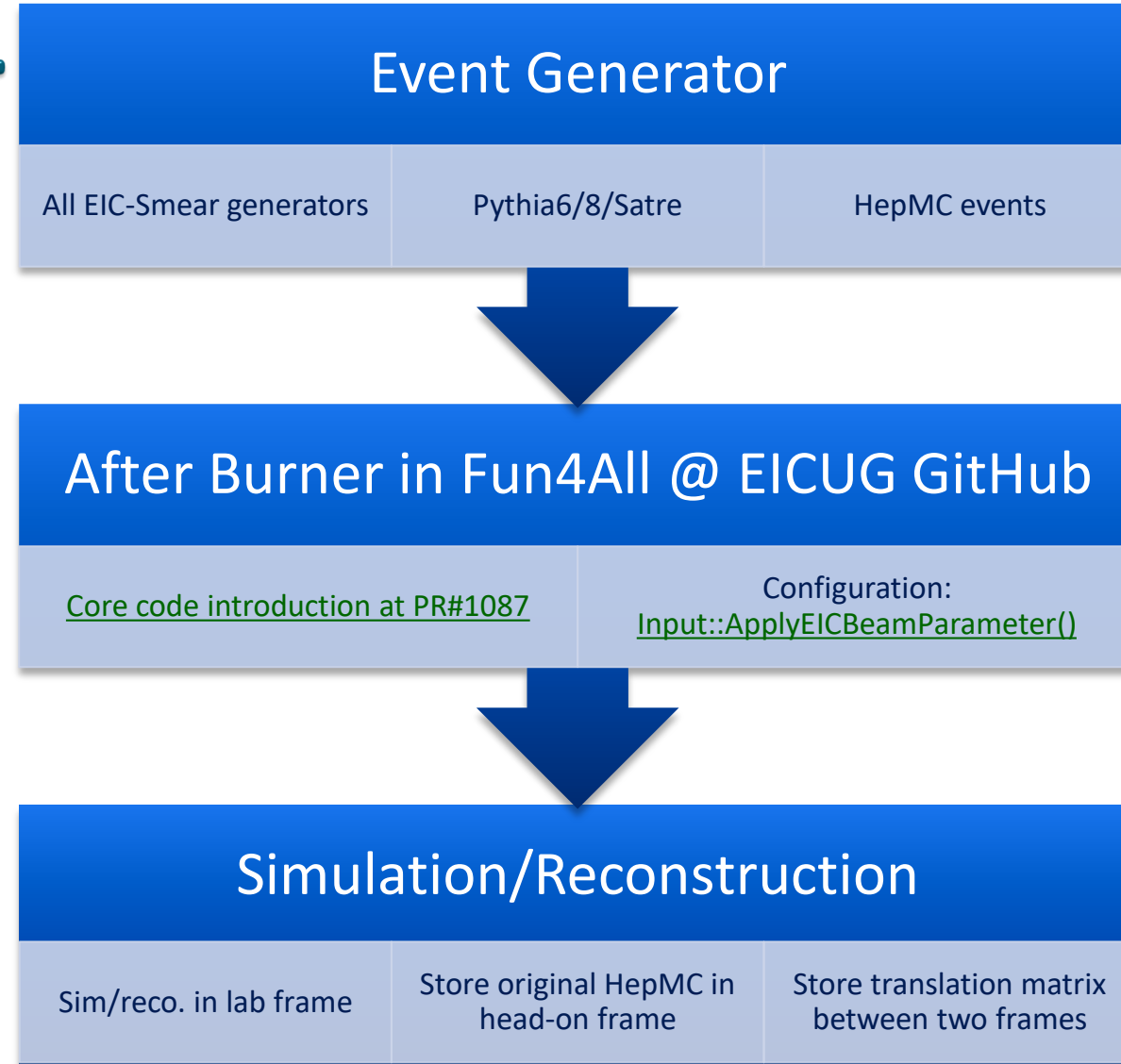
Leading order beam effects

- ▶ At EIC unique accelerator with diverse beam effect [[ref: CDR](#)]
 - -25 to +35 mrad beam crossing angle, both supported in ECCE setup
 - Angular beam divergence: $O(100\mu\text{rad})$
 - Crab crossing (bunch-z dependent angle smear): $O(<100\mu\text{rad})$
 - Beam energy spread $O(10^{-4})$
 - Beam vertex spread from 10cm h-bunch collider with 1-cm e-bunch at finite crossing angle
- ▶ Interestingly, sPHENIX will run w/ 2mrad beam crossing angle at RHIC [[ref: BUP](#)]
 - Common interest in sim/reco./theory
 - Also use Fun4All simulation+reconstruction framework [[link](#)]



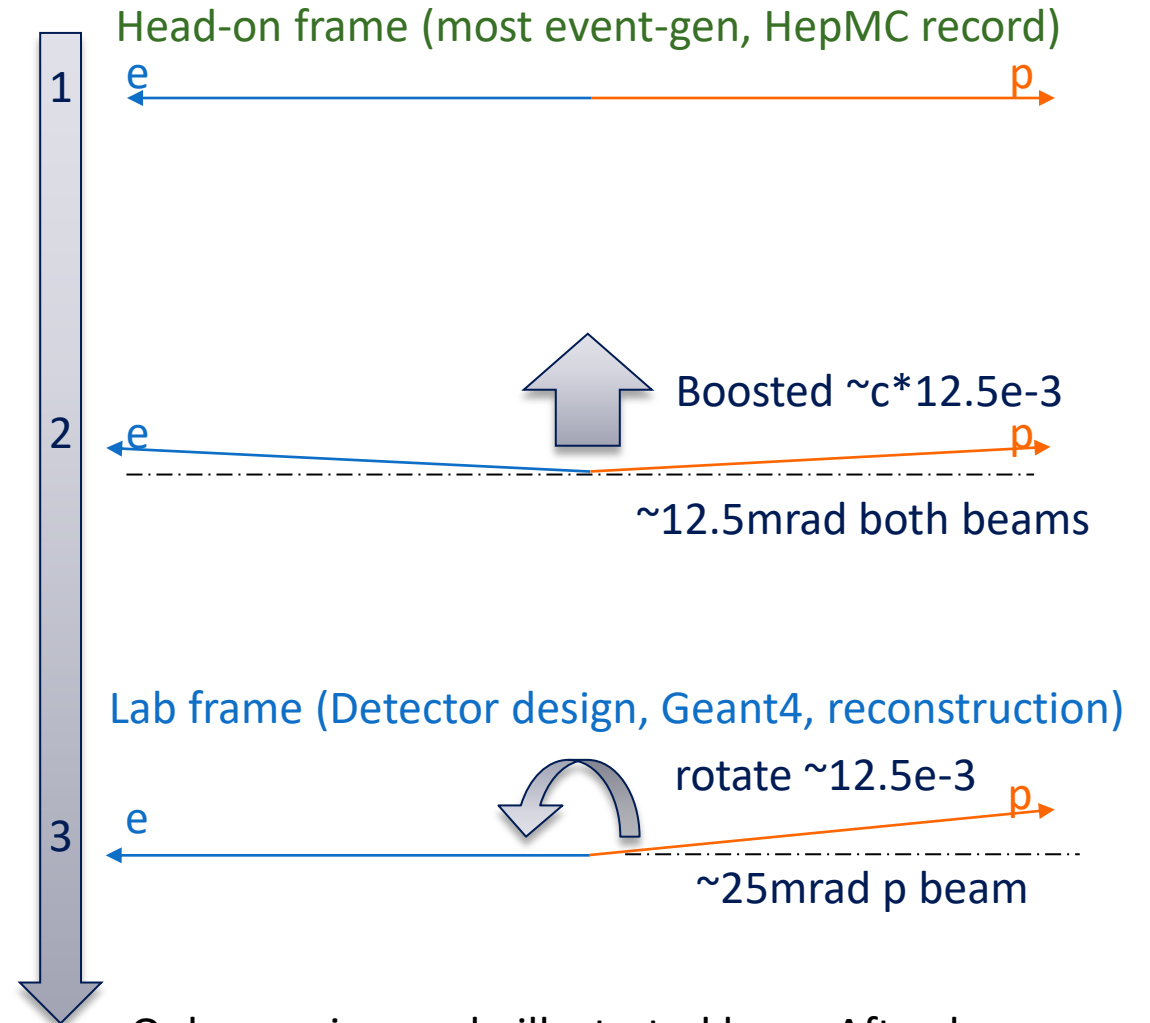
Beam effects in Fun4All sim.

- ▶ Not all event generator support beam effects while beam crossing and other effects are essential parts of EIC experiment
- ▶ After burner introduced to boost frame of any HepMC/EICSmear event of head-on collision to the lab frame with beam crossing, etc.
 - Note: $\sqrt{s_{eN}}$ is not changed in after-burner as it is boost invariant. Effect is small for most non-threshold measurement $O(10^{-4})$
- ▶ Book-keeping
 - Truth vertex, beam angle (variation event-by-event)
 - Truth and reco transformation matrix for reco objects \rightarrow head-on frame



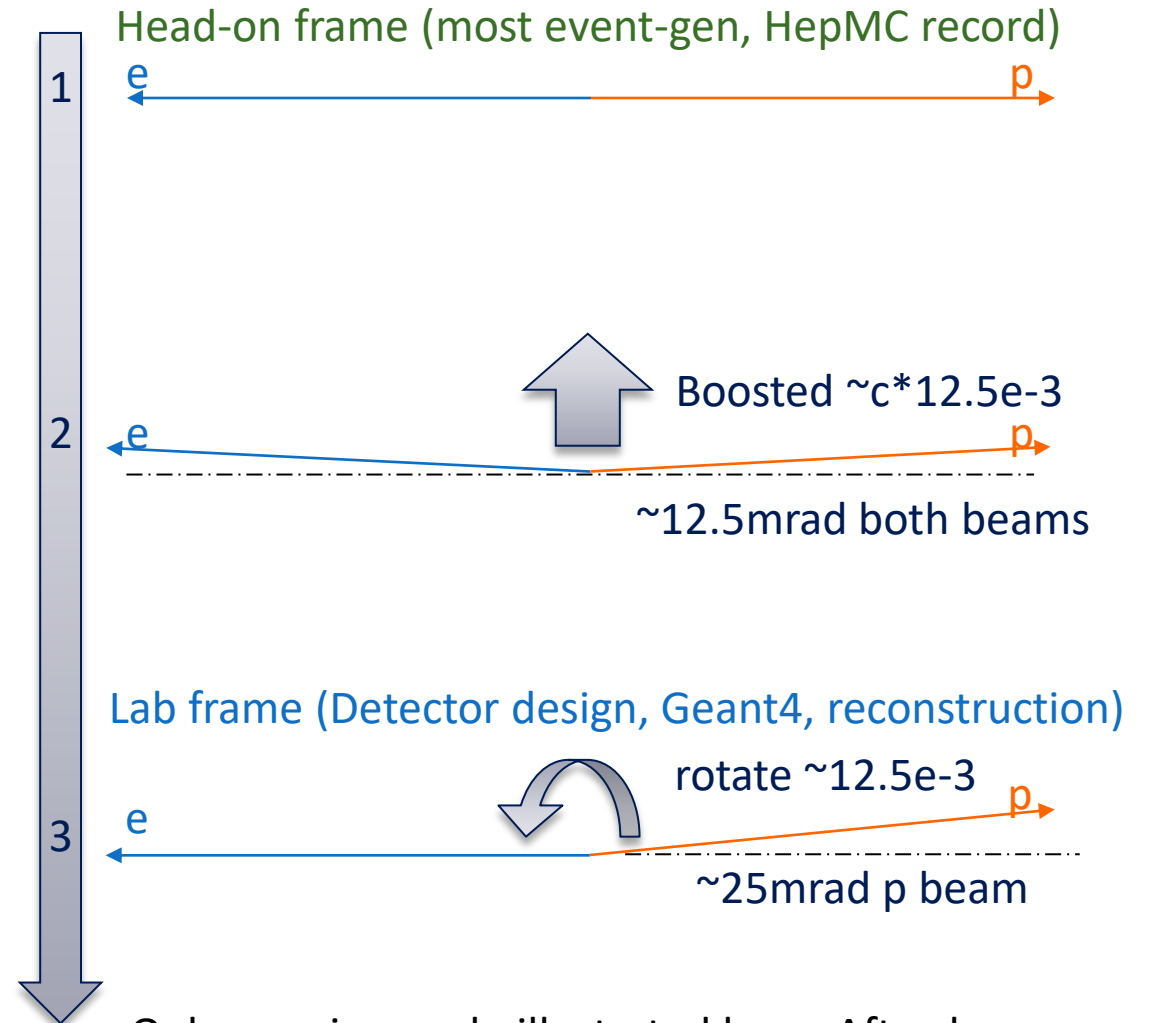
How does it work: the transformation

- ▶ Head-on \rightarrow Lab transformation: one boost + one rotation
- ▶ Precise solution for relativistic beam
 - Minimal modification to beam energy, or $\text{RMS}(\Delta p)$
 - Significant simplify config management independent of beam specie and energy
 - Visual aid for interpretation of x-ing effects on EIC observables
- ▶ Non-relativistic beam correction
 - Correction At order $O(\theta_{xing} \gamma_{Beam}^{-2}) \ll$ beam divergence



How does it work: algorithm flow

- ▶ Input via user macro [\[link\]](#) for beam angle, divergence, vertex shift in space time
- ▶ Calculate the boost-rotation-shift [\[link\]](#) that is used to translate a head-on-collision event generator's record to the lab frame and use in Geant4 simulation inputs
- ▶ Apply the boost-rotation-shift from event generator to G4 simulation input [\[link\]](#)
- ▶ Bookkeeping to allow analysis to reverse the transformation from lab observable to event generator frame

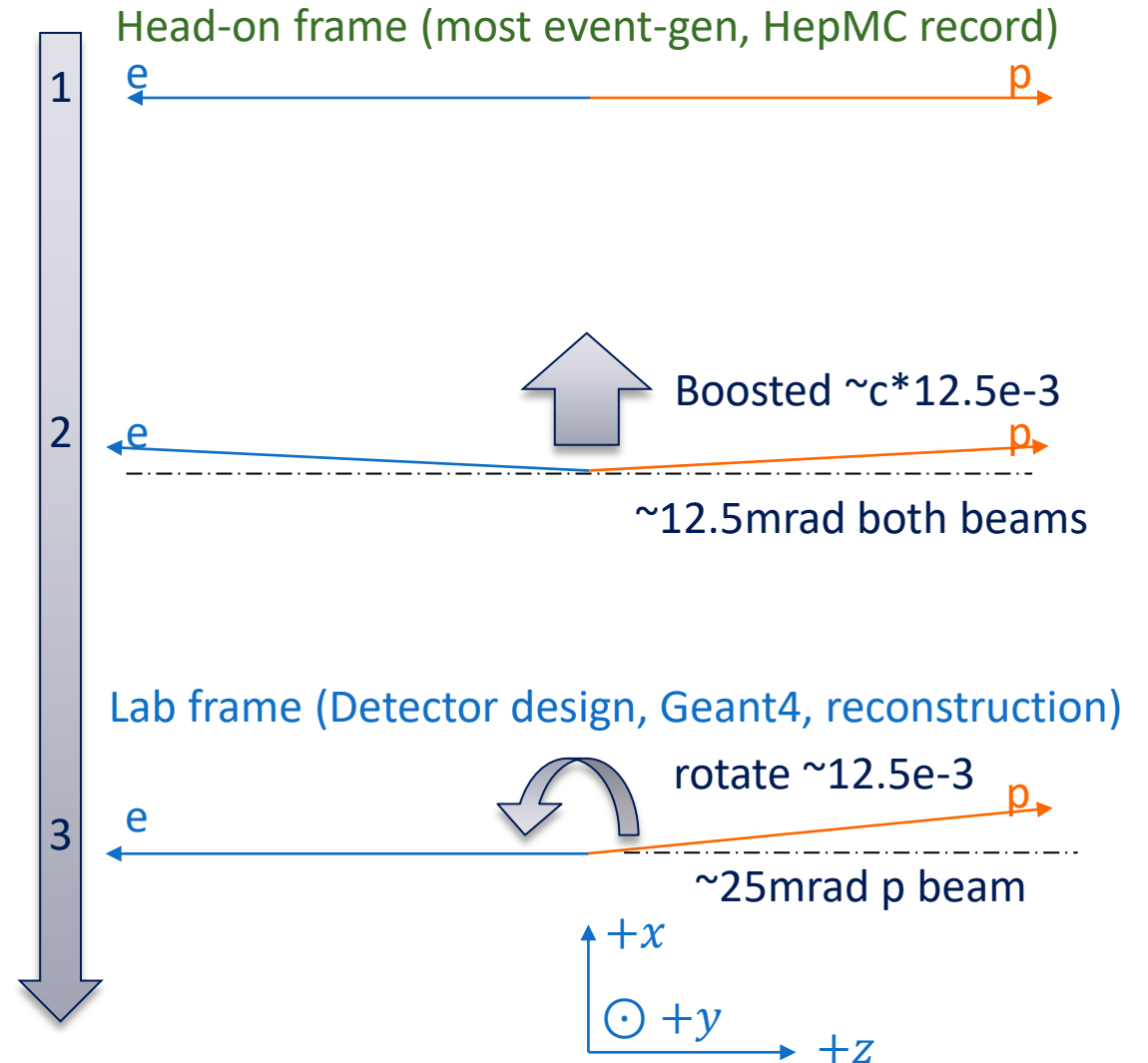


Only crossing angle illustrated here. After-burner handles all beam effects [\[link\]](#)

Coordinate system

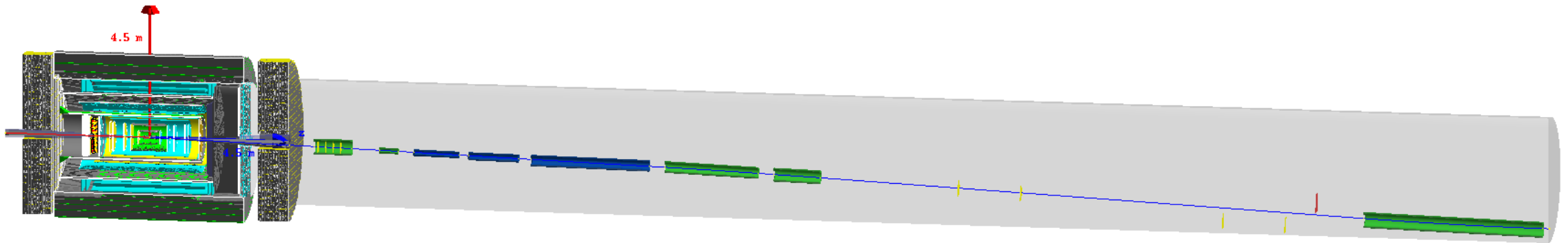
- ▶ Head-on frame as used in most event generator and stored in HepMCEventMap is different from the Lab Frame as used in Detector design, Geant4 simulation and reconstruction
- ▶ In lab frame, electron is along $-z$ axis, i.e. along symmetric axis of exp. and no B-bending
 - $+z$ axis: inverse of electron beam direction
 - $+y$ axis: up
 - $+x$ axis: $y \times z$, towards center of RHIC ring
- ▶ Note: from head-on to lab frame, beam energy increase by

$$E_{Lab} = E_{Headon} / \cos(\text{crossing angle} / 2)$$



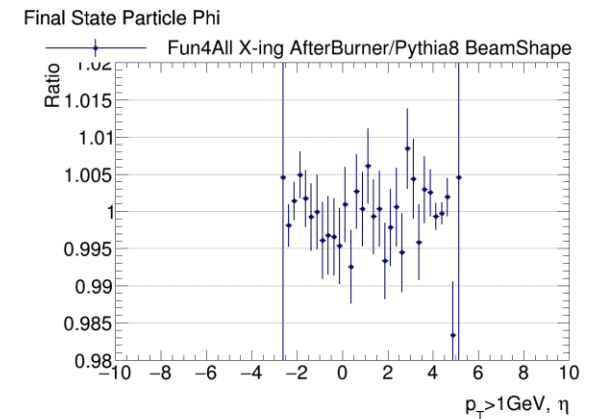
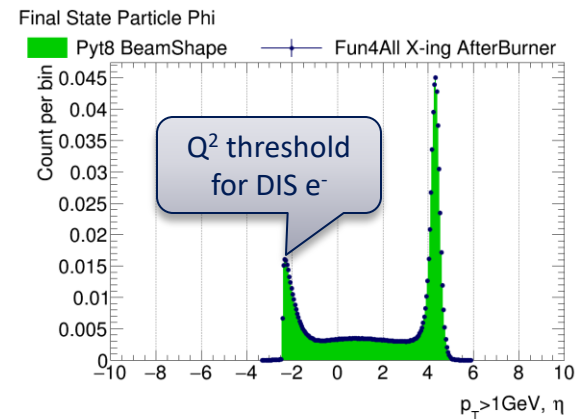
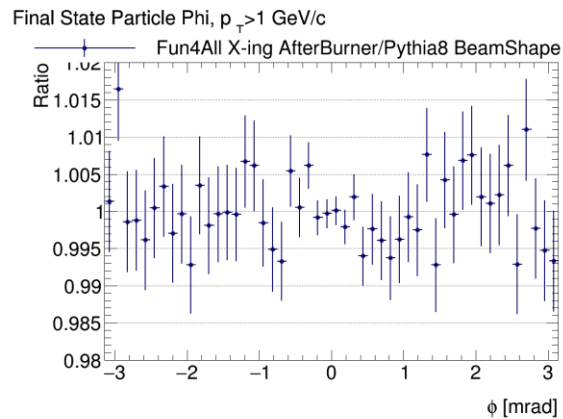
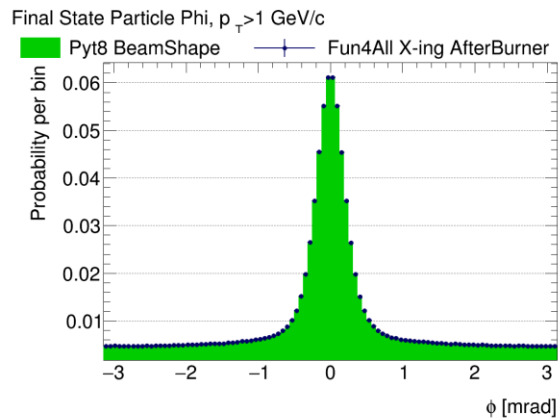
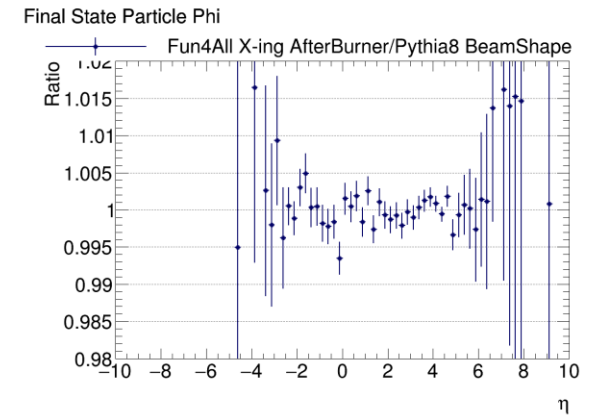
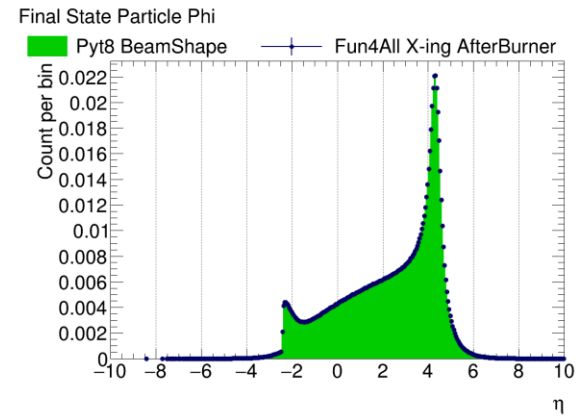
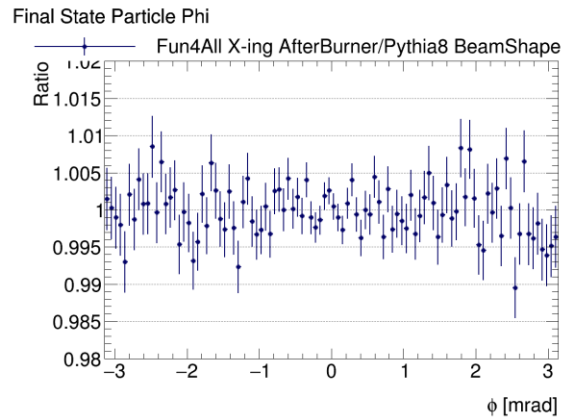
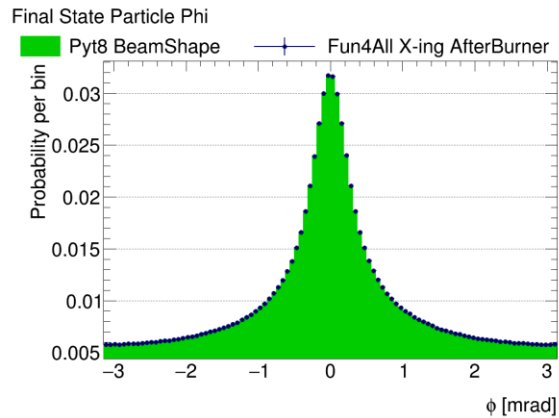
Beam transport checks

- ▶ Tested with proton and electron beam particle passing through each other in the head-on frame
- ▶ Boost-rotated to lab frame and validate the beam propagation through far-forward beamline
- ▶ Reference: <https://github.com/ECCE-EIC/macros/pull/26> from Bill Lee



Direct comparison: Fun4all afterburner vs Pythia8

- ▶ 1M Pythia8 events -> Fun4All beam afterburner -> G4 ↔ compared to 1M Pythia8 BeamShape [\[link\]](#). Also checked with IP6/8 and low-high beam configuration [\[link\]](#)
- ▶ Consistency well beyond the 1% stat. uncertainty provided by the test sample



Summary

- ▶ Fun4All: C++1x-based framework and integrate many commonly used software packages, e.g. Pythia6/8, Geant4, ROOT, GenFit2, ACTS, FastJet, KFParticle
- ▶ Widely used in EIC YR, also used for sim+reco. in stages of proposal preparation for all three EIC proto-collaborations
- ▶ Built-in EIC beam effect afterburner
- ▶ Main code for EIC use at <https://github.com/eic?q=Fun4All&type=all&language=&sort=>
 - For CORE use, suggest at least CORE-controlled macros and calibration repositories

Extra information



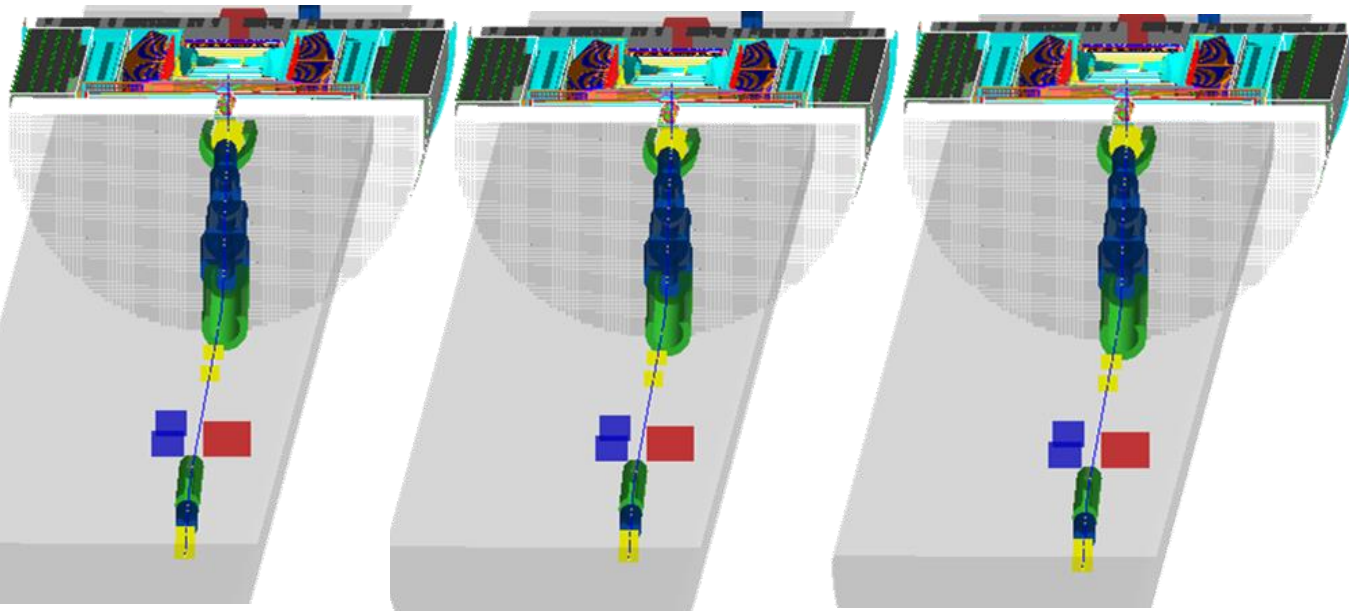
Beam steering at IP8, see also Bill's work

Work by Dhevan Gangadharan
(University of Houston)

Fun4all support sanmoutanous study for IP6 and IP8 in ECCE

- Same event-gen file for both IPs
- Beam effect applied in Fun4All depending on the IP selection

Fun4all Beam Steering Capability



IP8 41GeV proton

IP8 100GeV proton

IP8 275GeV proton

