ELECTRON ION COLLIDER USER GROUP
SOFTWARE NEWS

M. Diefenthaler on behalf of the
EICUG Software Working Group

EICUG Software Working Group

67 members

Convener
A. Bressan (Trieste)
M. Diefenthaler (JLAB)
T. Wenaus (BNL)

Mailing list eicug-software@eicug.org
subscribe via Google Group
Repository http://gitlab.com/eic
Website http://www.eicug.org/web/content/eic-software

Core Group


J. Adam (BNL)


M. Asai (SLAC)


N. Brei (JLAB)


A. Bressan (Trieste)


W. Deconinck (Manitoba)


M. Diefenthaler (JLAB)


J. Furletova (JLAB)

Picture
S. Furletov (JLAB)


S. Joosten (ANL)


K. Kauder (BNL)


A. Kiselev (BNL)

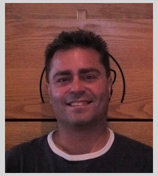

J. Lauret (BNL)


D. Lawrence (JLAB)


C. Pinkenburg (BNL)


M. Potekhin (BNL)


D. Romanov (JLAB)


M. Ungaro (JLAB)


T. Wenaus (BNL)

Working Group Goals

Support Yellow Report Initiative

Role of **Software Working Group** in Yellow Reports organization

Physics Working Group

Conveners

Detector Working Group

Conveners + **ex-offico**

Simulation team: Software Working Group

Role of Software Working Group in Yellow Report initiative

Develop

Support

Workflow environment for EIC simulations

- **to use** (tools, documentation, support) **and**
- **to grow with user input** (direction, documentation, tools)

Involvement from EICUG

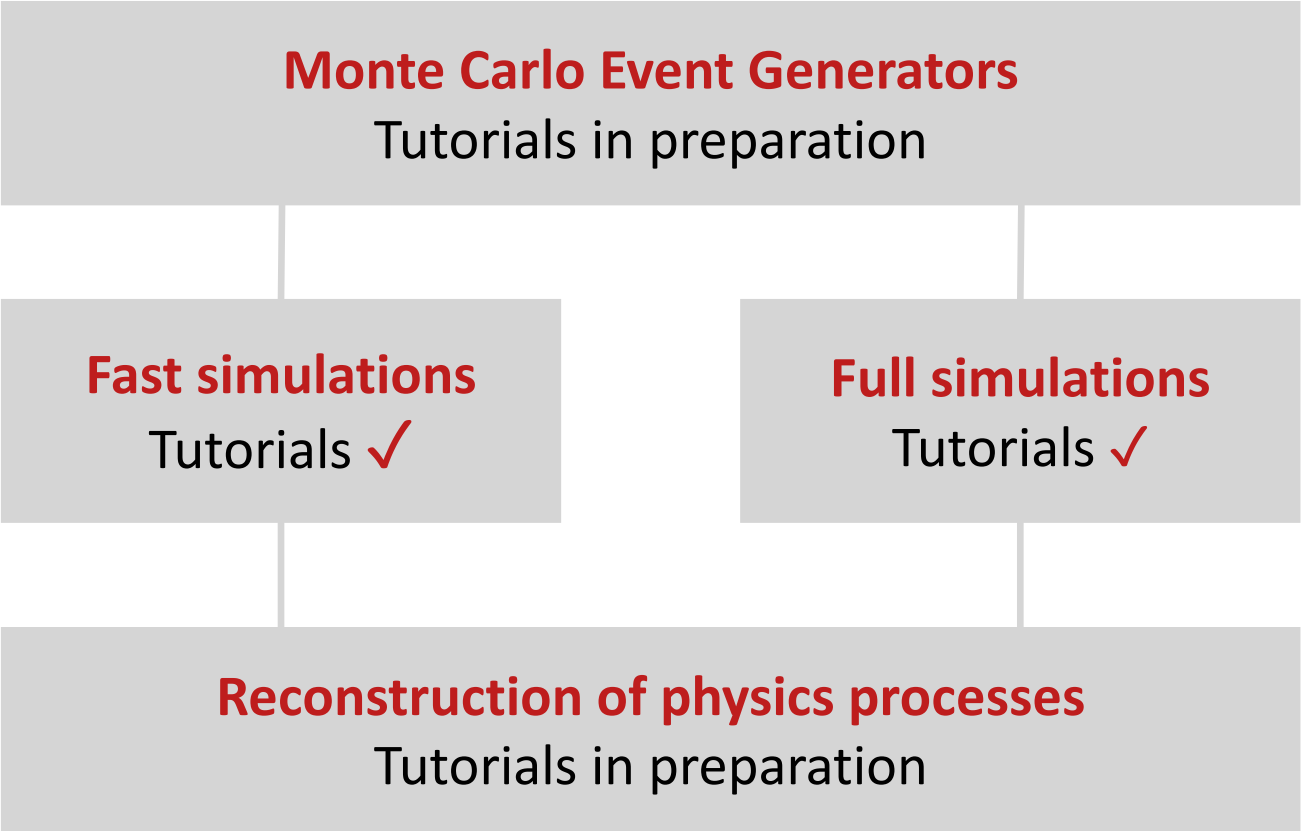
- **Coordinate simulations** Please continue to reach out to us.
- **Analysis preservation** Please make your software available and integrate it.
- **Design detectors** We rely on your expertise.
- **Developing reconstruction algorithms** We rely on your expertise.
- **Develop physics analysis** We rely on your expertise.

EIC Software

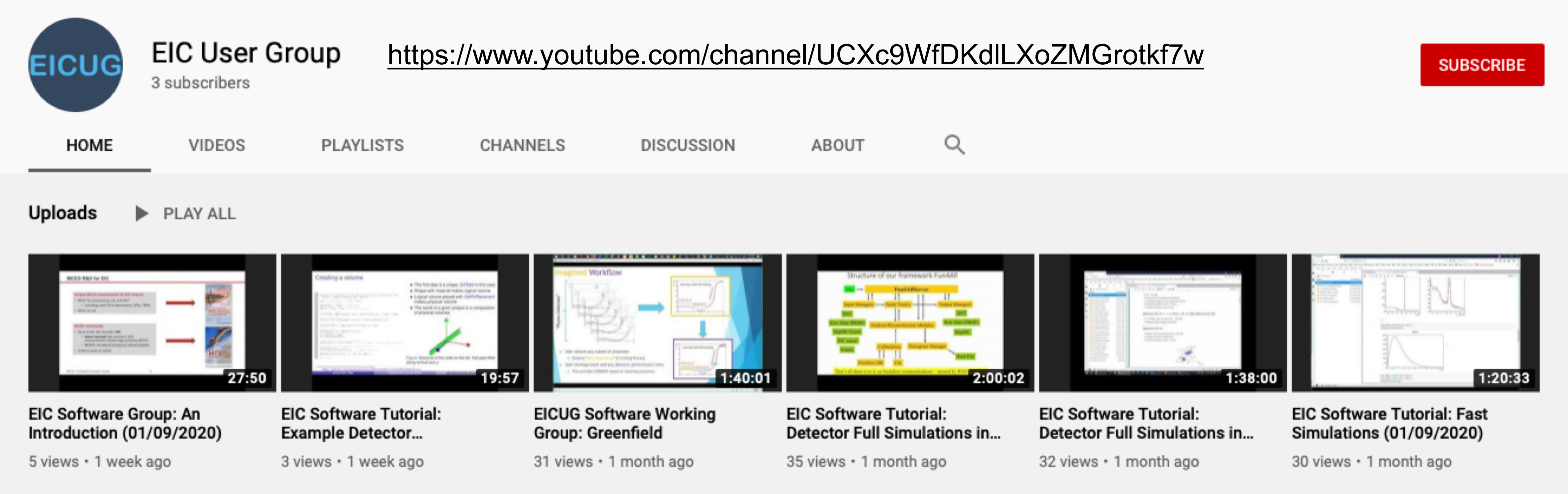
Simulation of physics processes
Physics Working Group

Simulation of detector responses
Detector Working Group

Physics analysis
Physics and Detector Working Groups



Online tutorials



Recordings from tutorials

- **Fast Simulation Tutorial** Introduction to JupyterLab workspace, using fast simulations as example
- **Detector Full Simulation Tutorials** Geant4 for EIC, how to modify existing detector concepts, and how to integrate a new detector into one of the existing detector concepts.

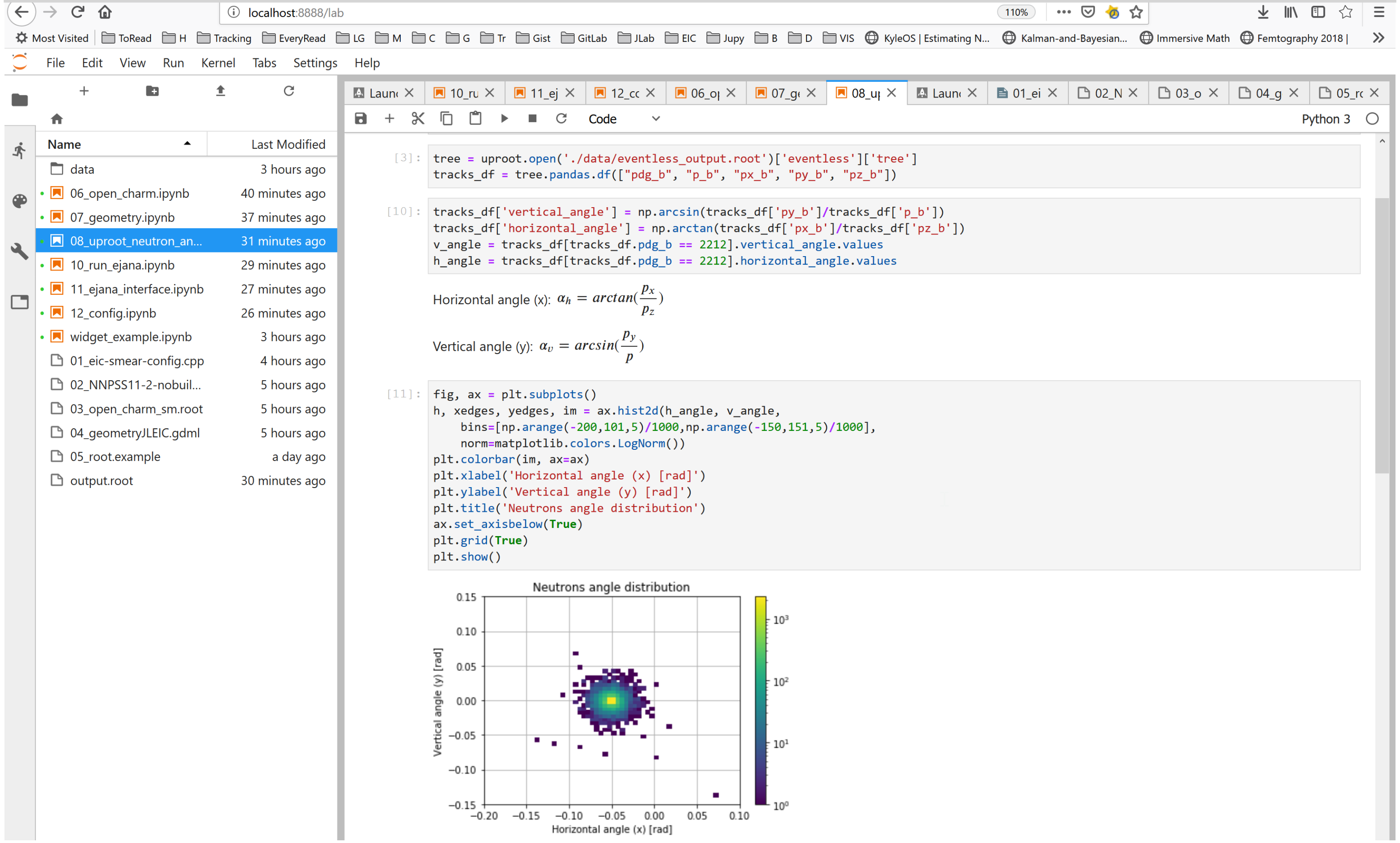
Approach

Workflow environment for Yellow Report initiative

Collaborative workspace for EICUG

JupyterLab

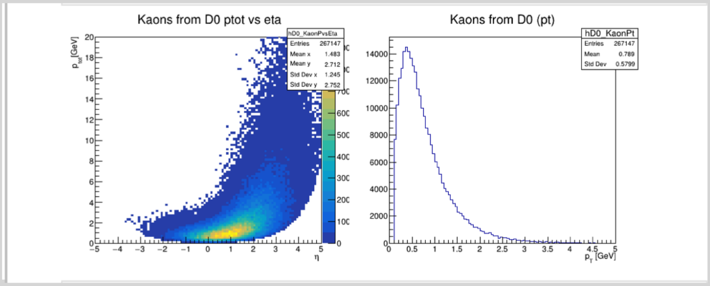
- web-based interactive analysis environment



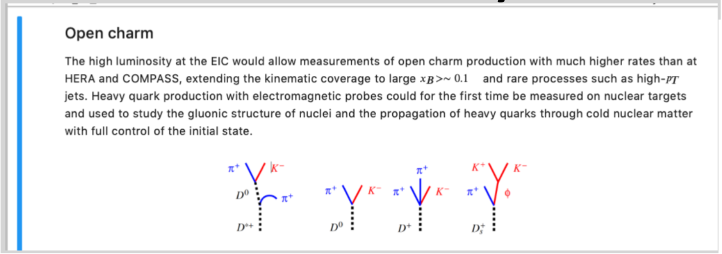
Jupyter Notebooks

- writing analysis code

- visualization of results

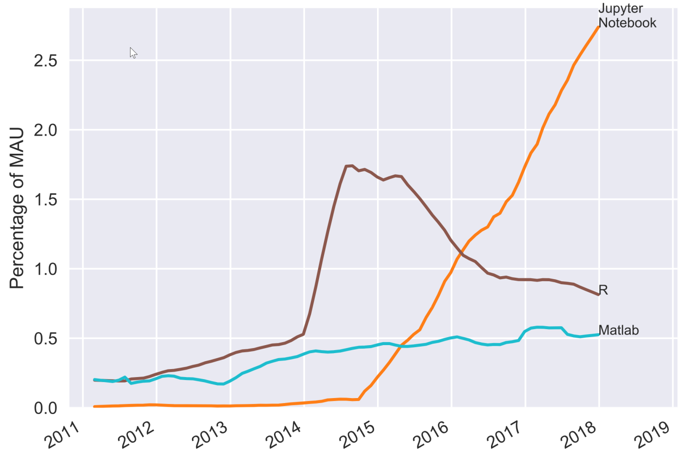


- narrative of the analysis



JupyterLab environment

- **bridge to modern data science**, e.g.,



- *Nature* **563**, 145-146 (2018): “Why Jupyter is data scientists’ computational notebook of choice”
- more than three million Jupyter Notebooks publicly available on GitHub

- **collaborative workspace** to create and share Jupyter Notebooks

- **web-based interactive analysis environment** accessible, consistent, reproducible analyses

- **fully extensible and modular** build a collection of analyses and analysis tools → **collection of Yellow Report studies**

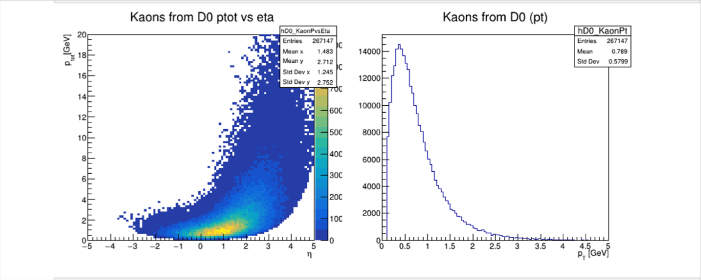
Jupyter Notebooks

- **writing analysis code**

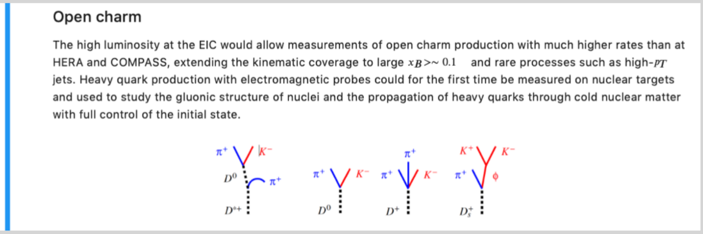
Python

Root/C++

- **visualization of results**



- **narrative of the analysis**



Modular design

Escaping complexity scaling trap

- provide interfaces to internal layers
- interaction between layers must be clear

Modularity each layer must be replaceable

"simple"	JupyterLab web interface
"moderate"	analysis scripts, eic-smear, python
"complex"	eJANA, plugins, C++
expert	JANA, Geant4, ROOT, EicRoot, fun4all

../data/beagle_eD.txt

```
[3]: jana.run()
```

Total events processed: 10001 (~ 10.0 kevt)

▸ **Full log**

▾ **Run command**

```
ejana
-Pplugins=beagle_reader,vmeson,event_writer
-Pnthreads=1
-Pnevents=10000
-Poutput=beagle.root
../data/beagle_eD.txt
-Pjana:debug_plugin_loading=1
```

Users and developers work on their **preferred** layer which will be always their simplest layer.

Report

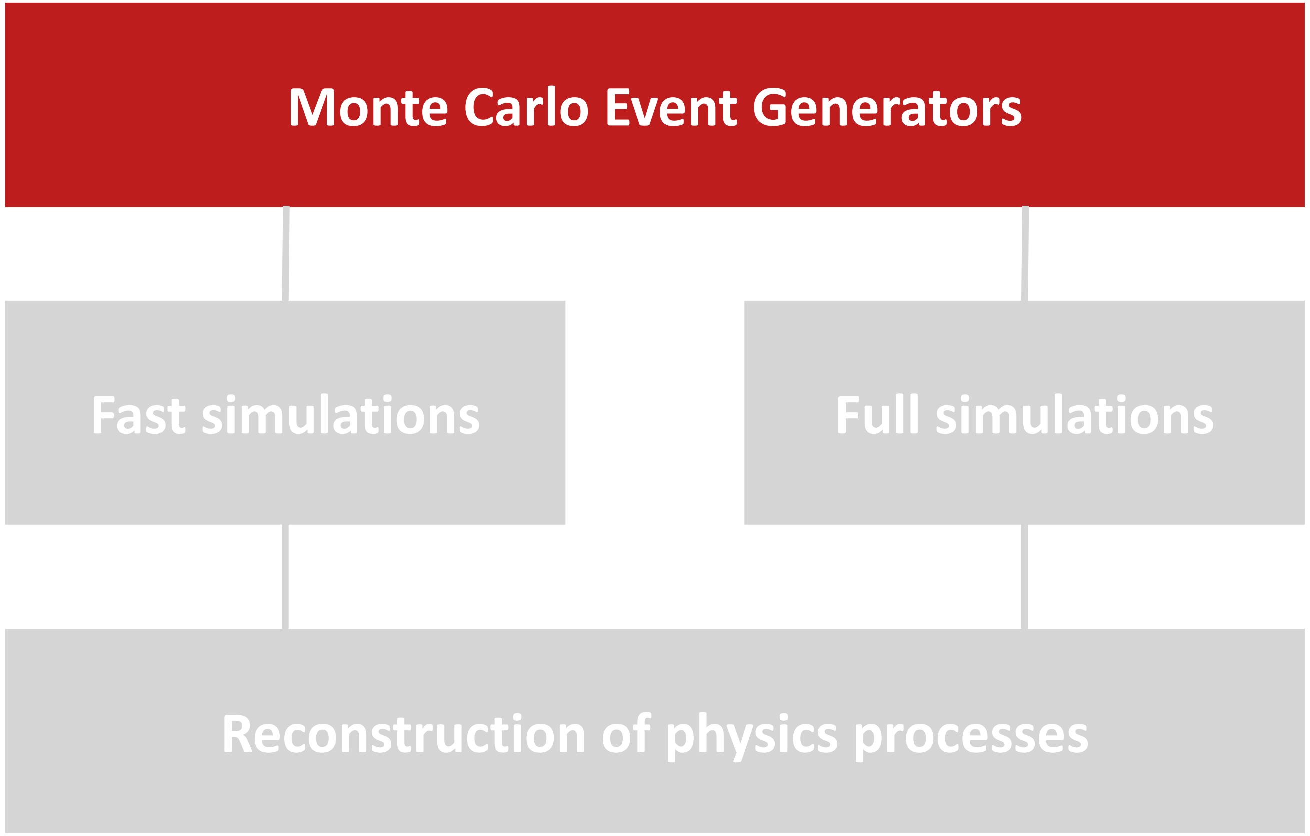
EIC Software Status

Simulations of physics processes and detector responses

Simulation of physics processes

Simulation of detector responses

Physics analysis



Broad collection of event generators used for EIC

Monte Carlo Event Generators (MCEG) from BNL task force

The following event generators are available:

- ep
  - DJANGOH: (un)polarised DIS generator with QED and QCD radiative effects for NC and CC events.
  - gmc_trans: A generator for semi-inclusive DIS with transverse-spin- and transverse-momentum-dependent distributions.
  - LEPTO: A leptoproduction generator - used as a basis for PEPSI and DJANGOH
  - LEPTO-PHI: A version of LEPTO with "Cahn effect" (azimuthal asymmetry) implemented
  - MILOU: A generator for deeply virtual Compton scattering (DVCS), the Bethe-Heitler process and their interference.
  - PYTHIA: A general-purpose high energy physics event generator.
  - PEPSI: A generator for polarised leptoproduction.
  - RAPGAP: A generator for deeply inelastic scattering (DIS) and diffractive $e + p$ events.
- eA
  - BeAGLE: Benchmark eA Generator for LEptoproduction - UNDER CONSTRUCTION - a generator to simulate ep/eA DIS events including nuclear shadowing effects (based on DPMJetHybrid)
  - DPMJet: a generator for very low Q2/real photon physics in eA
  - DPMJetHybrid: a generator to simulate ep/eA DIS events by employing PYTHIA in DPMJet
  - Sartre is an event generator for exclusive diffractive vector meson production and DVCS in ep and eA collisions based on the dipole model.

From https://wiki.bnl.gov/eic/index.php/Simulations and available in https://gitlab.com/eic/mceg

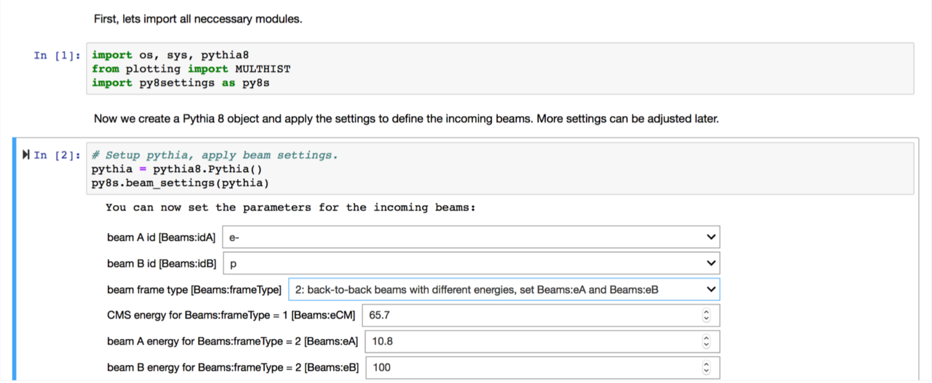
Next steps More examples and make **common MC productions** available on https://hepsim.jlab.org

MCEGs in EIC Software Collection

Pythia8+DIRE

- available in **eic-mceg**
- **soon** in other containers
- **next** updates for photoproduction, diffractive processes, radiative effects etc.

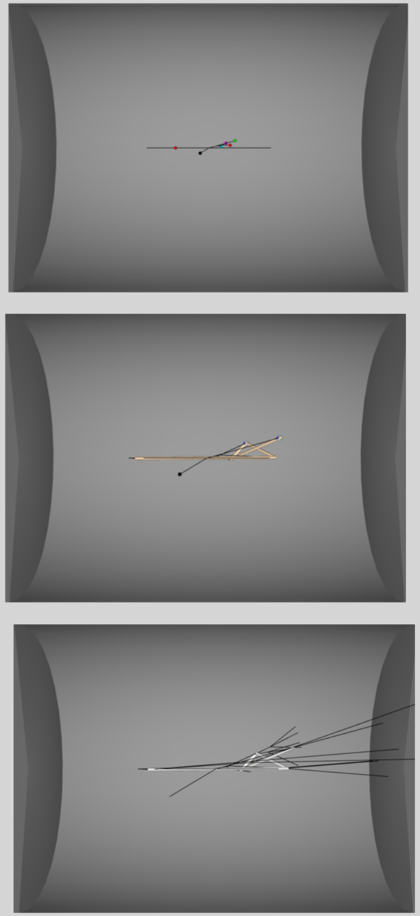
Ongoing JupyterLab integration



Next eSTARlight, Herwig, Sherpa (and MCEGs in previous slide)

Benchmark processes Work with Physics Working Group

Visualization of ep collision

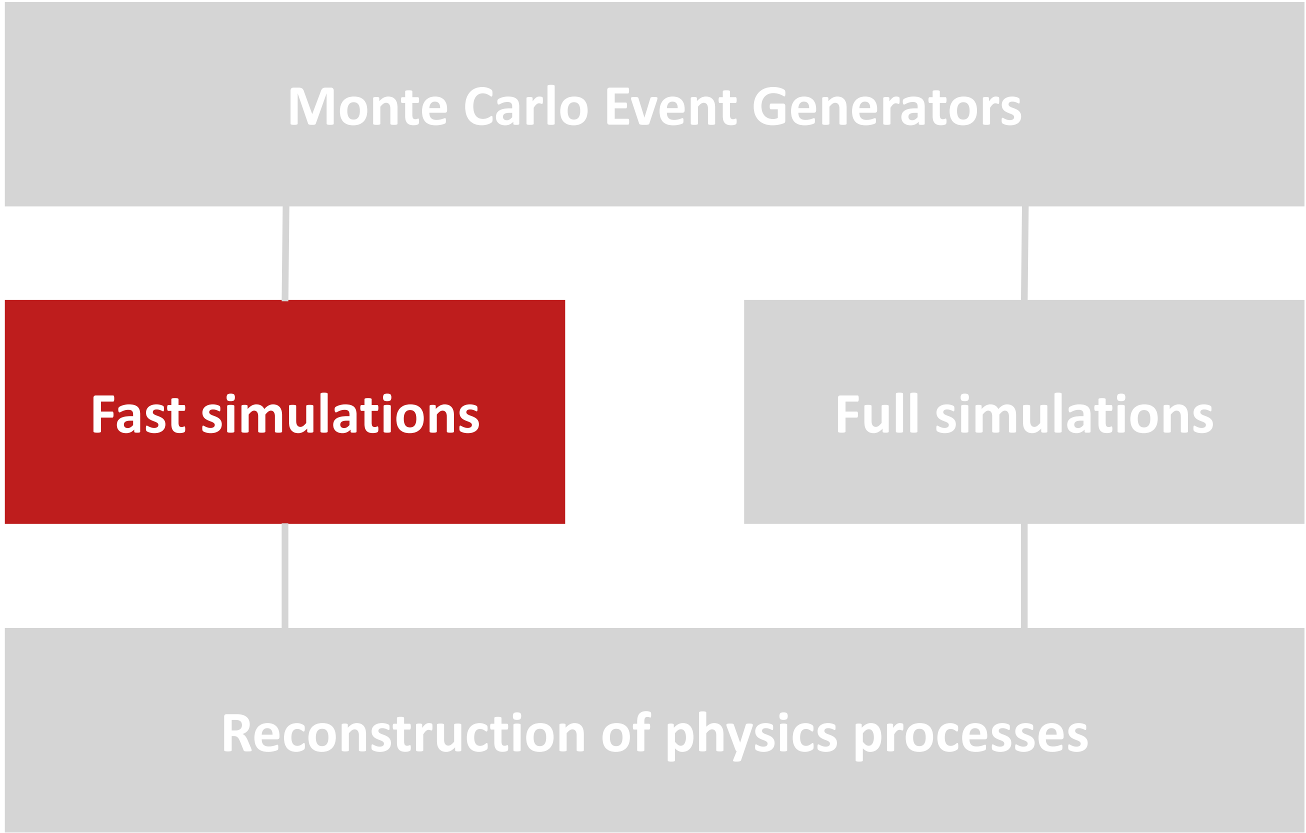


Simulations of physics processes and detector responses

Simulation of physics processes

Simulation of detector responses

Physics analysis

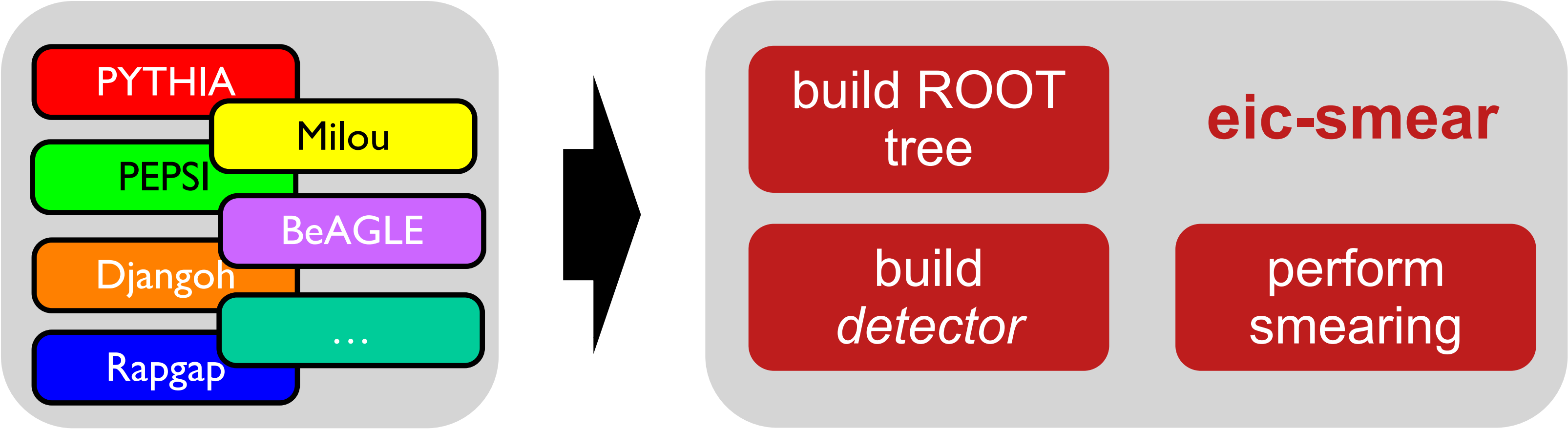


eic-smear

ROOT application for fast simulations

- study sensitivity to known parameterizations
- vary parameters to constrain needed performance
- **fast** O(1000) events / s
- **agile** features on request, soon *dE/dx*

Neither a full detector simulation **nor** *Geant4 light*.



- transforms MCEG output into Tparticle four-vectors
  - final, stable only (π, K, p, e, …)
- applies **known** smearing particle-by-particle

(single) quantity, X, to smear: E, p, θ, φ + Function defining σ(X) = f([E, p, θ, φ]) + Acceptance for X in E, p, θ, φ, pT, pZ

TFormula variations and additions can be done in a handful of lines

Electromagnetic calorimeter in eic-smear

Page, Chue, Aschenauer, arXiv:1911.00657

Component	Pseudorapidity Range	Resolution
Mid-Back EMCal	$-2 < \eta < -1$	$\frac{7\%}{\sqrt{E}} \oplus 1\%$

Need to convert to θ, (2.437, 2.872)

- Make a Device:

```
Smear::Device emcalMidBck(Smear::kE, "sqrt(0.01*0.01*E*E + 0.07*0.07*E)");
```

Smears energy (kE)
Others: kP, kPhi, …

Translate relative resolution σ/*E* to absolute σ

$$(\sigma/E)^2 = \left(7\%/\sqrt{E}\right)^2 + 1\%^2 \Rightarrow \quad \sigma = \sqrt{7\%^2 E + 1\%^2 E^2}$$

- Endow it with an Acceptance Zone and "Species":

```
Smear::Acceptance::Zone emMidBck( 2.4366, 2.8726, 0., TMath::TwoPi(), …
emcalMidBck.Accept.AddZone(emMidBck);
emcalMidBck.Accept.SetGenre(Smear::kElectromagnetic);
```

Range in
θ, ϕ, E, p, pt, pz

- `kAll, kElectromagnetic, kHadronic`
  OR
- PIDs, e.g. `Accept.AddParticle(2112);`

- Add to Detector:

```
Smear::Detector det;
det.AddDevice(emcalMidBck);
```

- full example in scripts/smearBeAST.cxx
- class documentation at www4.rcf.bnl.gov/~eickolja/

JLEIC Smear (C++)

Smearing depends on type of the particle and its theta/pseudorapidity

Energy smearing for e and gammas

```
if (p.Eta() > -3.5 && p.Eta() < -1.1) {
    double c1 = gRandom->Gaus(0, 0.02 * sqrt(p.E()));
    double c2 = gRandom->Gaus(0, 0.001 * p.E());
    double c3 = gRandom->Gaus(0, 0.005);
    double sigma=c1 + c2+c3;
    new_e= p.E() + sigma;
```

$$\sigma(E) \sim a\sqrt{E} \oplus b \cdot E \oplus c$$

Energy smearing for neutrons

```
// Neutron coming to Zero Degree calorimeter
if(abs(particle->pdg) == 2112 && p.Theta() < 0.01){
      // zdc
    new_e = gRandom->Gaus(p.E(), 1. * sqrt(p.E()));
```

PT and angle smearing for charged particles

```
double_t mynewPt = gRandom->Gaus(p.Pt(), 0.01 * pow(p.Pt(), 2) + 0.005 * p.Pt());

double_t mynewTheta = gRandom->Gaus(p.Theta(),0.001);
double_t mynewPhi = gRandom->Gaus(p.Phi(),0.001);
```

VTX smearing for charged particles

```
double vtx_smear = fabs(gRandom->Gaus(0, 25.));
gRandom->Sphere(x, y, z, abs(vtx_smear));
```

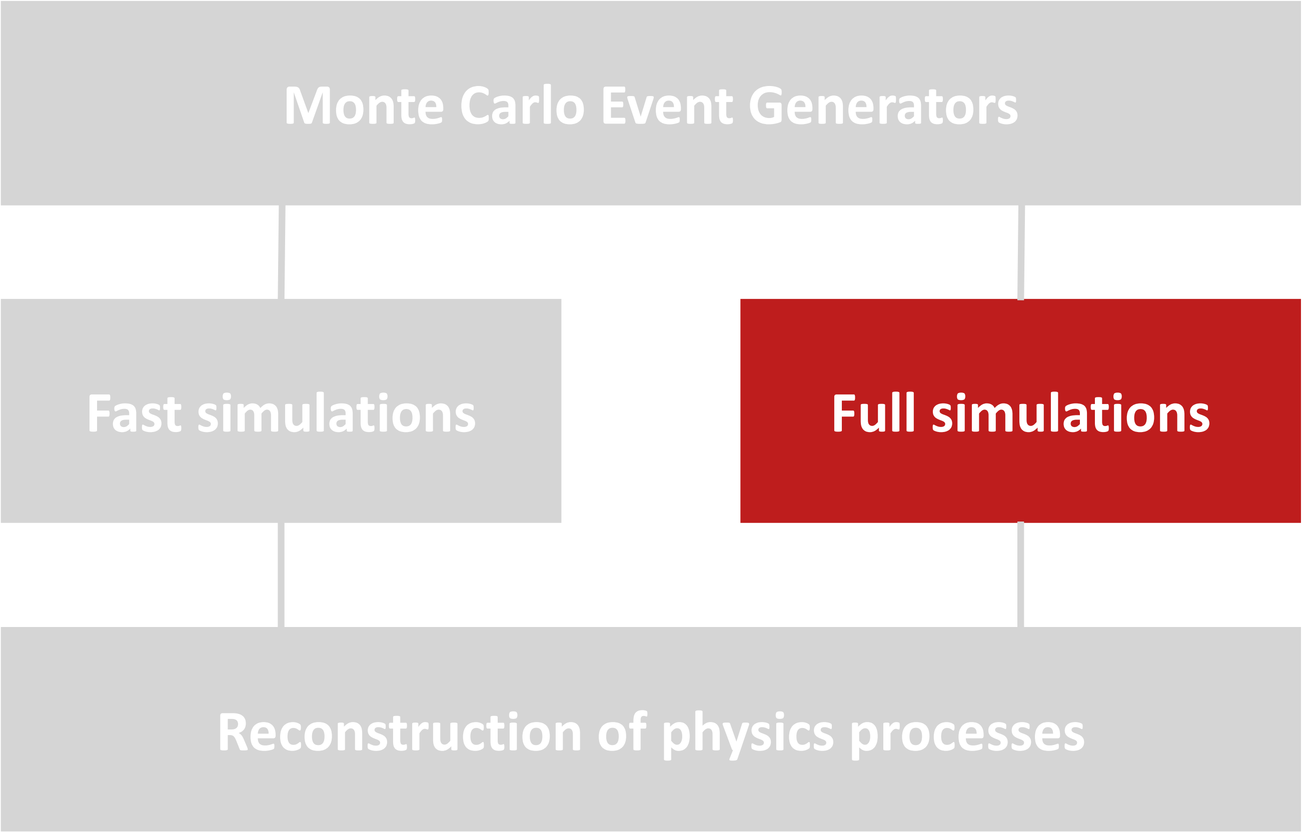
PID to be implemented …

Simulations of physics processes and detector responses

Simulation of physics processes

Simulation of detector responses

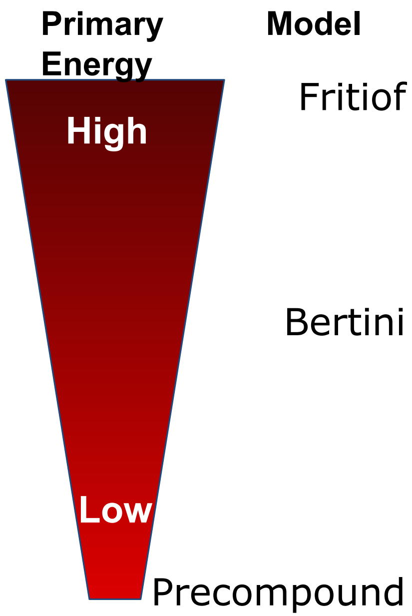
Physics analysis



Detector simulations

EIC

- detector (and physics) simulations rely on Geant4
- energy range is different from LHC
- validation, tuning and extension including test beam studies required



- **collaboration with Geant4 Collaboration (liaison: M. Asai)**
- **Geant4 for EIC**
  - Geant4 10.6 recommended (released Dec. 6, 2019)
  - maintain EIC physics list
  - coordinate input for Geant4 validation based on EIC physics list
- **Geant4 infrastructure for Yellow Reports initiative**
  - common repository for detector R&D for EIC
  - common detector description in Geant4 (C++) and not yet DD4hep (sub-detectors developed in Geant4 (C++))
  - possible common detector naming convention for EIC
  - possible common hits output structure
  - tutorials on how to implement and integrate subdetector in EIC detector concepts

Two solutions proposed

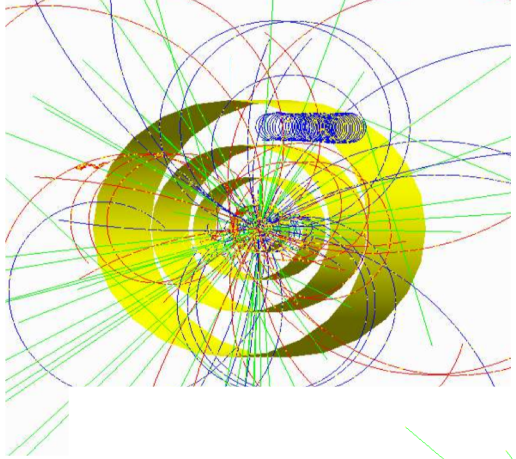
1. detector simulations in **fun4all**
2. Geant4 application **g4e**, part of ESCalate

Why two options?

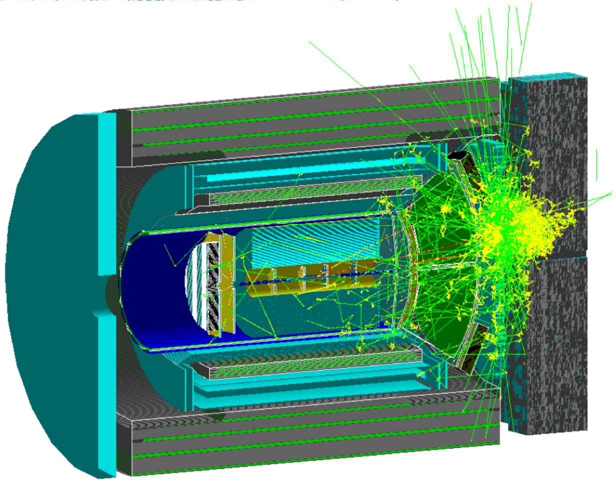
- At The Software Working Group was caught by the start of the "Yellow Report" effort with two ongoing developments for full simulations:
  - **fun4all**, originated from within (s)PHENIX, mature and centered around the use of ROOT macros
  - **g4e**, build up for the EIC (and therefore in a *younger* stage of development) constructed as a pure GEANT4 application (and integrated into JupyterLab environment)
- Each of the two is supported by a core team of developers.
- We put forward both options, leaving the *users* the freedom to choose base on their coding preferences.
- We will take advantage of the two codes to cross-check few selected and critical results in order to improve our confidence in the outcome of the simulations.

Fun4All + GEANT4

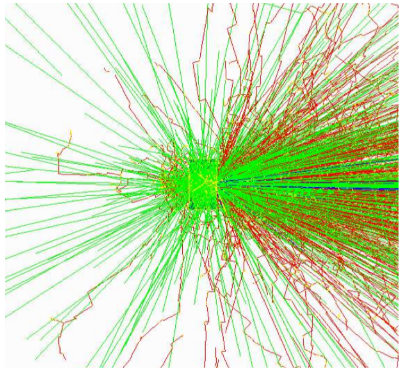
- Mature Framework based on ROOT, steering with ROOT macros
- Modular – each detector is its own entity
- No central code needs to be modified when adding new detectors
- Detectors are combined using ROOT macros
- Distribution as singularity container + libraries in cvmfs*
- Daily builds + Continuous Integration
- No geometry model enforced
- Interface to eic-smear: most EIC specific Event generators accessible
- Pre-canned configurations for EIC-sPHENIX and partial JLEIC
- Used to provide input for our EIC detector LOI**
- Generic Volumes (box, cylinder, cone) can be implemented no macro level



Pythia8 in a six layer silicon detector mockup and 2T field



Sarte as seen by an EIC detector



10 GeV Au on water phantom (NASA Space Radiation Lab)

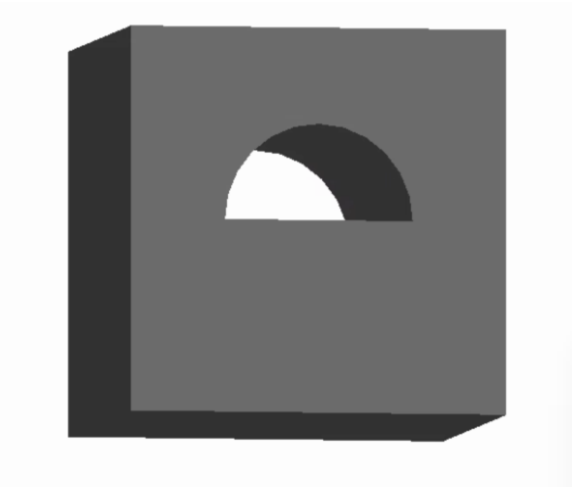
*Installation: https://github.com/EIC-Detector/Singularity

**https://arxiv.org/pdf/1402.1209.pdf
https://indico.bnl.gov/event/5283/attachments/20546/27556/eic-sphenix-dds-final-2018-10-30.pdf

For details: see selected Fun4All presentations https://www.phenix.bnl.gov/WWW/publish/pinkenbu/EIC/

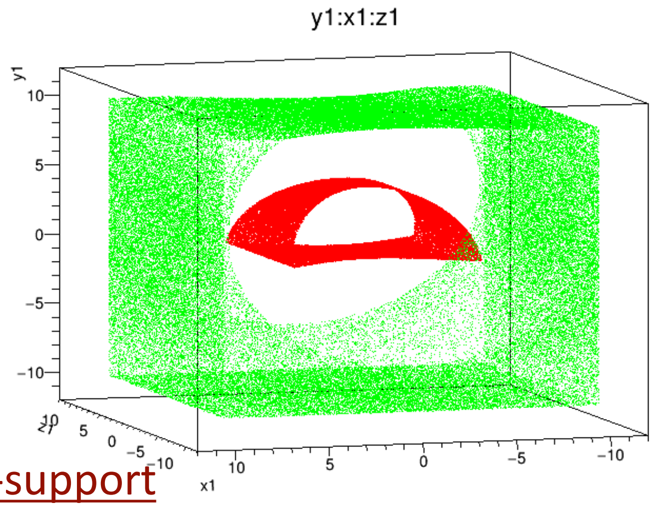
Implementing a Detector in Fun4All

Simplest Example, more sophisticated to come:
https://github.com/EIC-Detector/g4exampledetector:
simple/source: Simplest case - everything hardcoded, only active volumes
simple/macro: Fun4All_G4_Example01.C to run the show (and save Hits in ntuple)



Example01: block with
½ cylindrical hole

Let's call your detector PDirc*, 3 classes need to be implemented :
G4PDircSubsystem → interface between Fun4All and Detector
G4PDircDetector → GEANT4 Construct method
G4PDircSteppingAction → select which quantities to store for each hit
*Detector names can be set on the command line but you do not want identically named sources



Geantino Scan to verify geometry using entry/exit coordinates of geantino tracks

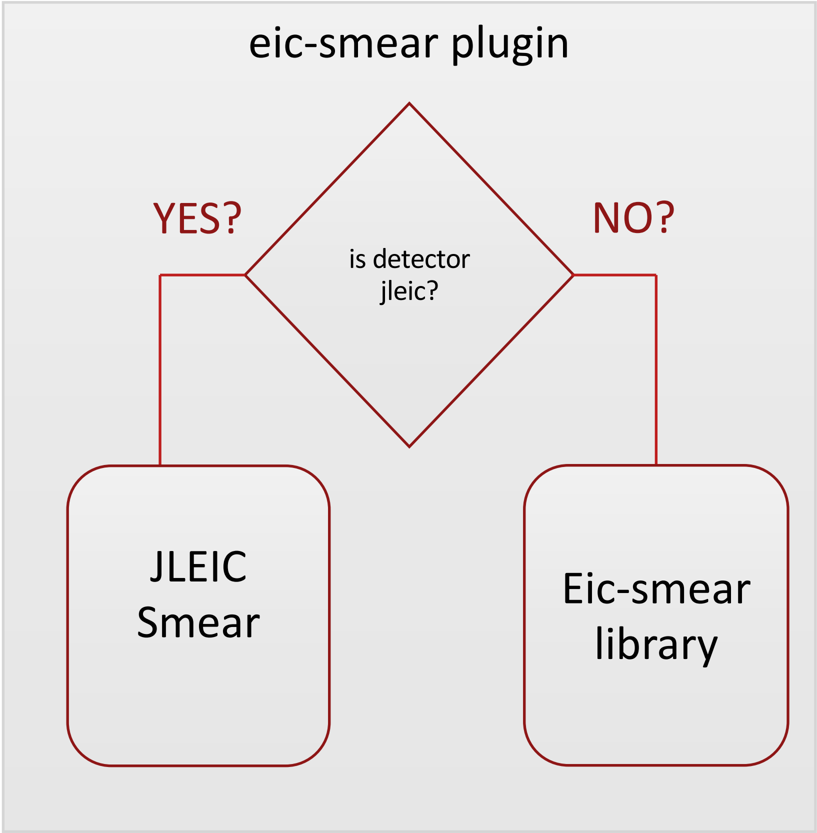
> You will find that help will always be given at Hogwarts to those who ask for it.
>
> – Dumbledore

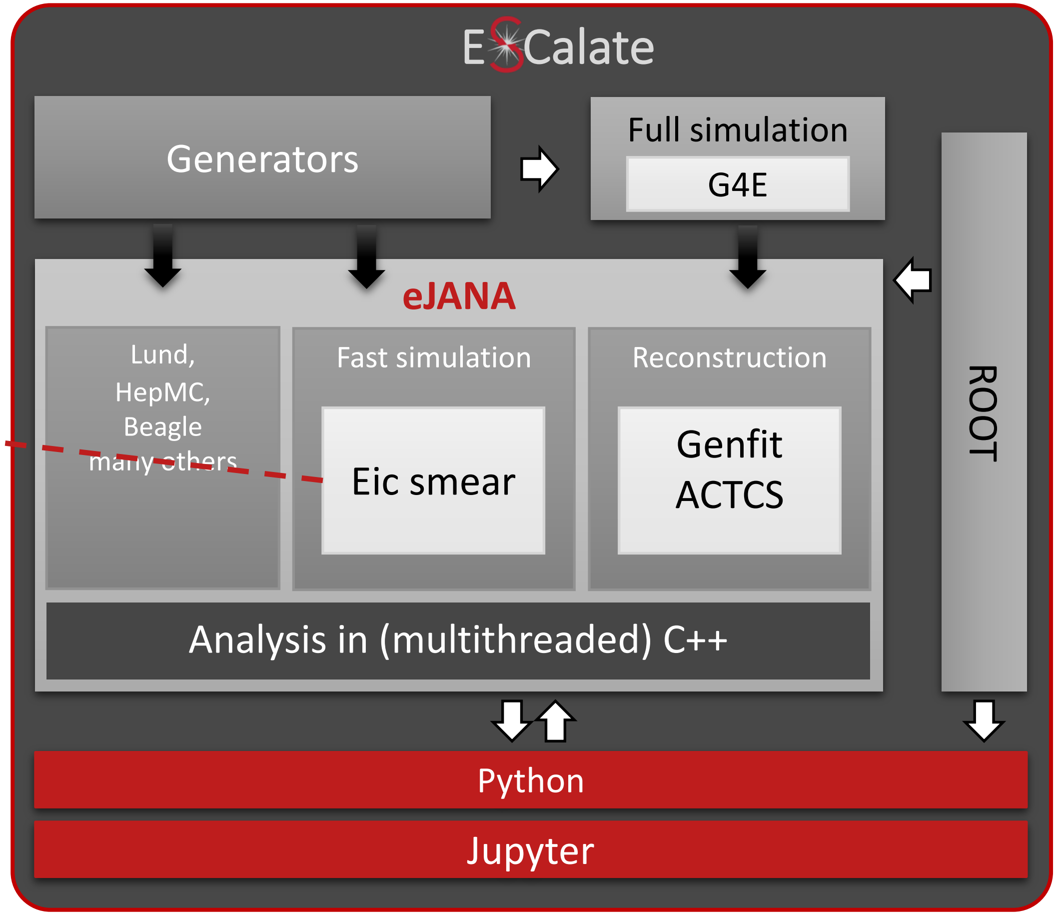
Tutorials:
https://github.com/EIC-Detector/tutorials

Join mattermost channel for support:
https://chat.sdcc.bnl.gov/eic/channels/fun4all-software-support

Email:
Chris Pinkenburg pinkenburg@bnl.gov
Jin Huang jhuang@bnl.gov

ESCalate software stack for EIC simulations



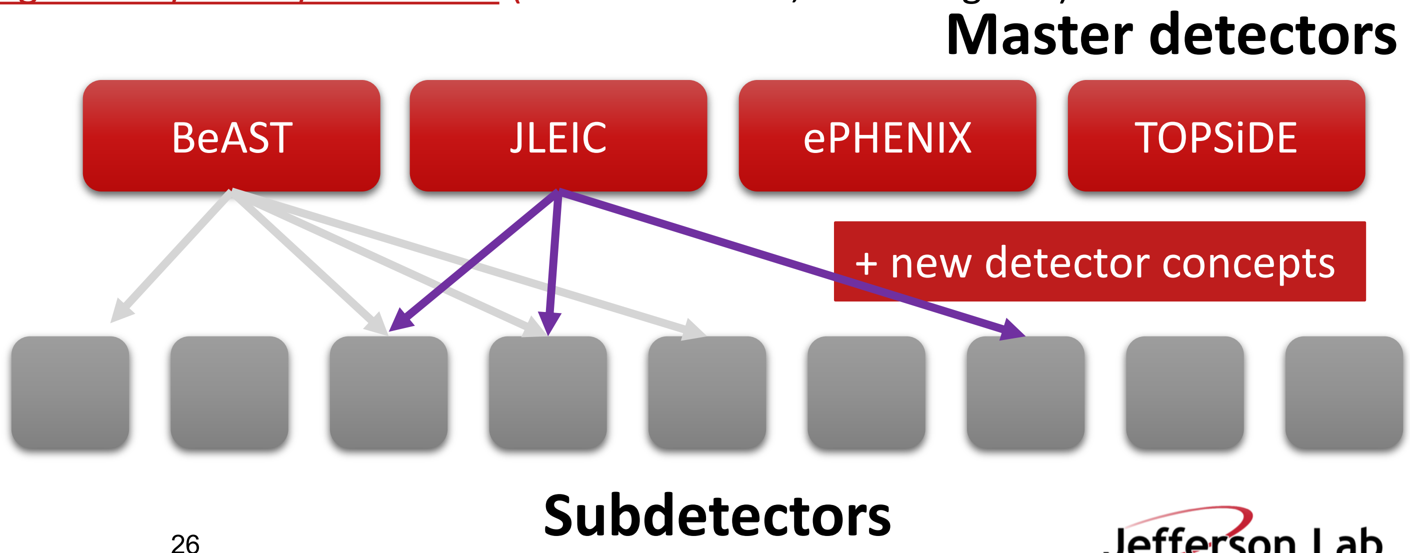


Escalate recent updates

- **Detector updates** on RHIC beamline and support detectors (Tagger, LMON, roman pods, ZDC, etc.)
- **On fly build** (Ninja+PCH+Unity build+JOC). Fast partial self compilation on start. Consider all G4E code as ROOT macros but running on optimized C++ speed. Change in a detector? +3 sec **once** on start.
- **Multithreading** (still beta). With fast (lower details) configuration get ~1000kHz on 8 cores. Benchmarks and examples are upcoming.
- **Updates on parts:** Randomized particle guns (g4e and g4epy), root output with flattened files, many others.
- **High priority work in process**:
  - Updates in the documentation
  - Work on direct python use is being prototyped
  - Examples on TGeo and STEP detectors
  - Examples on eJANA plugins: https://jeffersonlab.github.io/JANA2/index.html (Tutorial Section, HOWTO guide)

To import a subdetector:

- *SubDetectorInterface* class implementation
- Subscribe to various standard Geant 4 actions *(SteppingAction, StackingAction, etc)*
- Define subdetector's place in one or many "master" detectors



Accelerator interface

Accelerator design (beam elements)

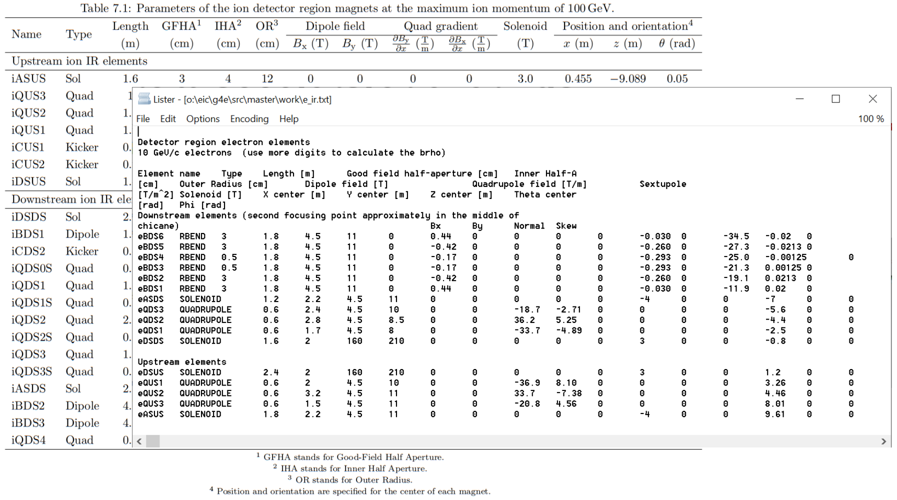


Table 7.1: Parameters of the ion detector region magnets at the maximum ion momentum of 100 GeV.

Name	Type	Length	GFHA[1]	IHA[2]	OR[3]	Dipole field		Quad gradient		Solenoid	Position and orientation[4]		
		(m)	(cm)	(cm)	(cm)	B_x (T)	B_y (T)	$\frac{\partial B_y}{\partial x}$ $(\frac{T}{m})$	$\frac{\partial B_x}{\partial x}$ $(\frac{T}{m})$	(T)	x (m)	z (m)	θ (rad)
Upstream ion IR elements													
iASUS	Sol	1.6	3	4	12	0	0	0	0	3.0	0.455	−9.089	0.05
iQUS3	Quad	1											
iQUS2	Quad	1.											
iQUS1	Quad	1.											
iCUS1	Kicker	0.											
iCUS2	Kicker	0.											
iDSUS	Sol	1.											
Downstream ion IR ele													
iDSDS	Sol	2.											
iBDS1	Dipole	1.											
iCDS2	Kicker	0.											
iQDS0S	Quad	0.											
iQDS1	Quad	1.											
iQDS1S	Quad	0.											
iQDS2	Quad	2.											
iQDS2S	Quad	0.											
iQDS3	Quad	1.											
iQDS3S	Quad	0.											
iASDS	Sol	2.											
iBDS2	Dipole	4.											
iBDS3	Dipole	4.											
iQDS4	Quad	0.											

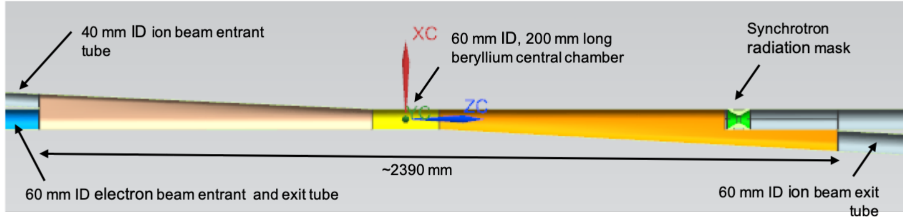
[1] GFHA stands for Good-Field Half Aperture.
[2] IHA stands for Inner Half Aperture.
[3] OR stands for Outer Radius.
[4] Position and orientation are specified for the center of each magnet.

Lister - [o:\eic\g4e\src\master\work\e_ir.txt]

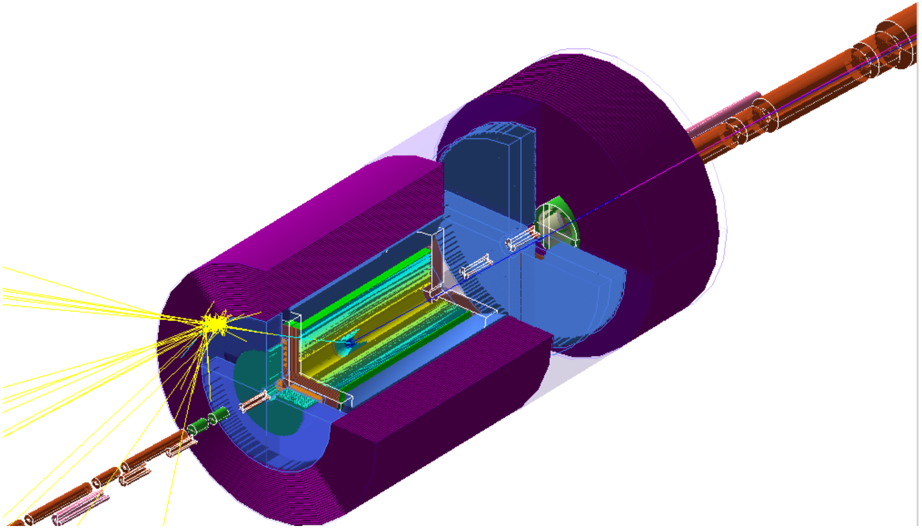
File Edit Options Encoding Help 100 %

```
Detector region electron elements
10 GeV/c electrons  (use more digits to calculate the brho)

Element name    Type    Length [m]      Good field half-aperture [cm]   Inner Half-A
[cm]    Outer Radius [cm]       Dipole field [T]                Quadrupole field [T/m]          Sextupole
[T/m^2] Solenoid [T]    X center [m]    Y center [m]    Z center [m]    Theta center
[rad]   Phi [rad]
Downstream elements (second focusing point approximately in the middle of
chicane)                                        Bx      By      Normal  Skew
eBDS6   RBEND   3       1.8     4.5     11      0       0.44    0       0       0       0       -0.030  0       -34.5   -0.02   0
eBDS5   RBEND   3       1.8     4.5     11      0       -0.42   0       0       0       0       -0.260  0       -27.3   -0.0213 0
eBDS4   RBEND   0.5     1.8     4.5     11      0       -0.17   0       0       0       0       -0.293  0       -25.0   -0.00125        0
eBDS3   RBEND   0.5     1.8     4.5     11      0       -0.17   0       0       0       0       -0.293  0       -21.3   0.00125 0
eBDS2   RBEND   3       1.8     4.5     11      0       -0.42   0       0       0       0       -0.260  0       -19.1   0.0213  0
eBDS1   RBEND   3       1.8     4.5     11      0       0.44    0       0       0       0       -0.030  0       -11.9   0.02    0
eASDS   SOLENOID        1.2     2.2     4.5     11      0       0       0       0       0       -4      0       0       -7      0       0
eQDS3   QUADRUPOLE      0.6     2.4     4.5     10      0       0       -18.7   -2.71   0       0       0       0       -5.6    0       0
eQDS2   QUADRUPOLE      0.6     2.8     4.5     8.5     0       0       36.2    5.25    0       0       0       0       -4.4    0       0
eQDS1   QUADRUPOLE      0.6     1.7     4.5     8       0       0       -33.7   -4.89   0       0       0       0       -2.5    0       0
eDSDS   SOLENOID        1.6     2       160     210     0       0       0       0       0       3       0       0       -0.8    0       0

Upstream elements
eDSUS   SOLENOID        2.4     2       160     210     0       0       0       0       0       3       0       0       1.2     0       0
eQUS1   QUADRUPOLE      0.6     2       4.5     10      0       0       -36.9   8.10    0       0       0       0       3.26    0       0
eQUS2   QUADRUPOLE      0.6     3.2     4.5     11      0       0       33.7    -7.38   0       0       0       0       4.46    0       0
eQUS3   QUADRUPOLE      0.6     1.5     4.5     11      0       0       -20.8   4.56    0       0       0       0       8.01    0       0
eASUS   SOLENOID        1.8     2.2     4.5     11      0       0       0       0       0       -4      0       0       9.61    0       0
```

Engineering Design (CAD)



Detector Simulations (Geant4)



Tuning

Status

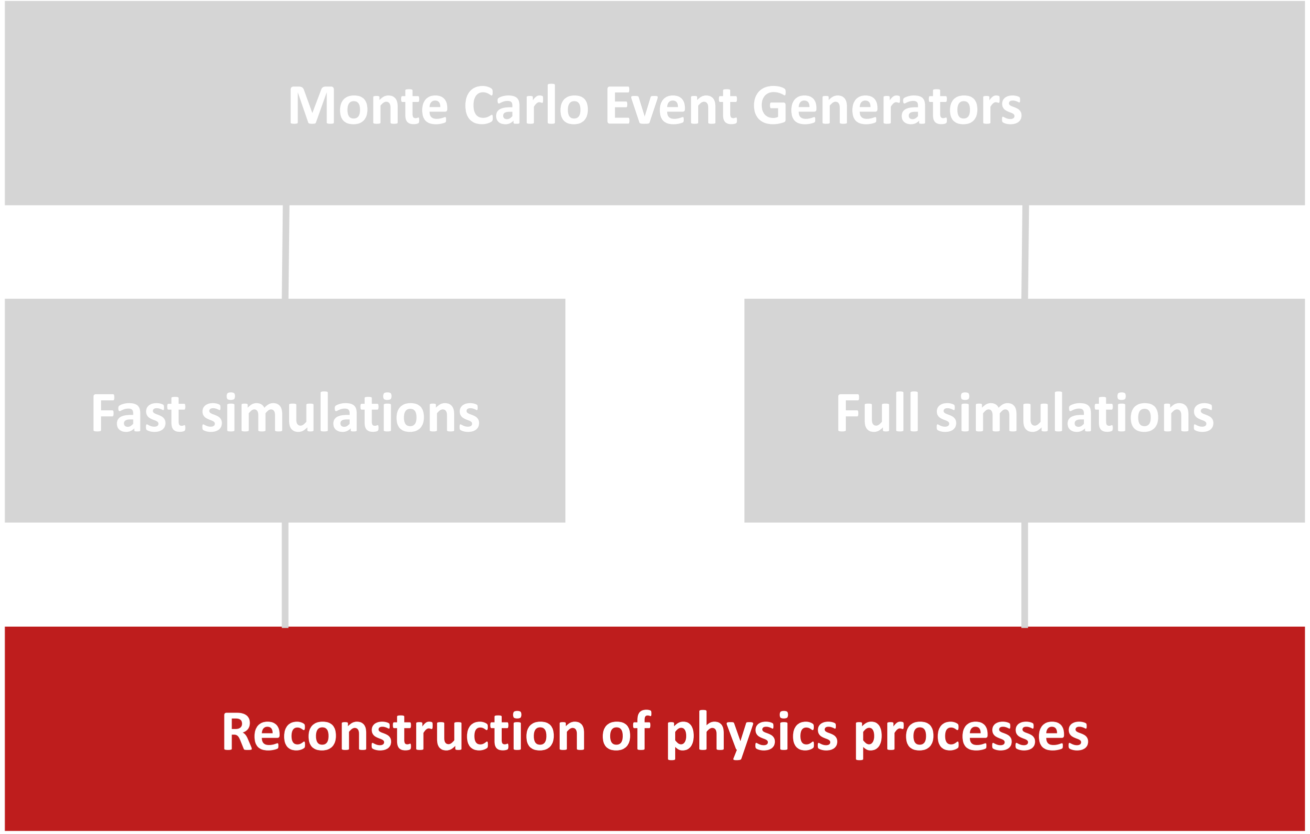
eRHIC and JLEIC information available
Common interface under active development

Simulations of physics processes and detector responses

Simulation of physics processes

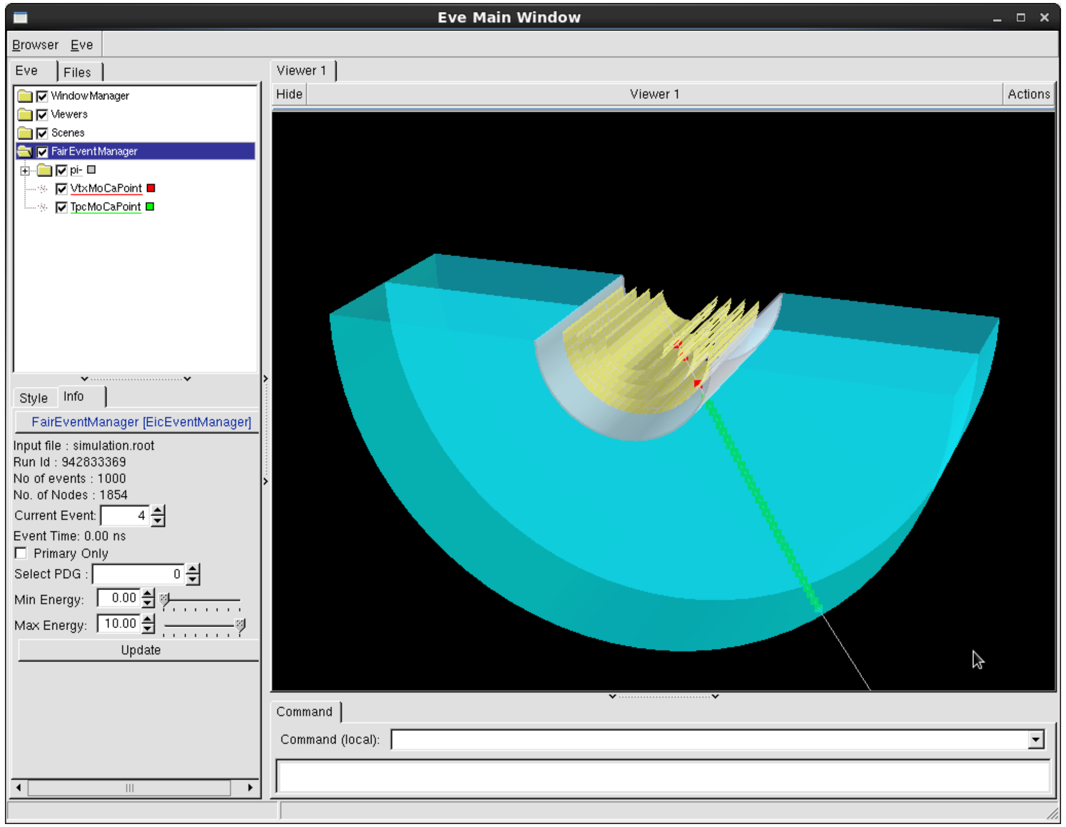
Simulation of detector responses

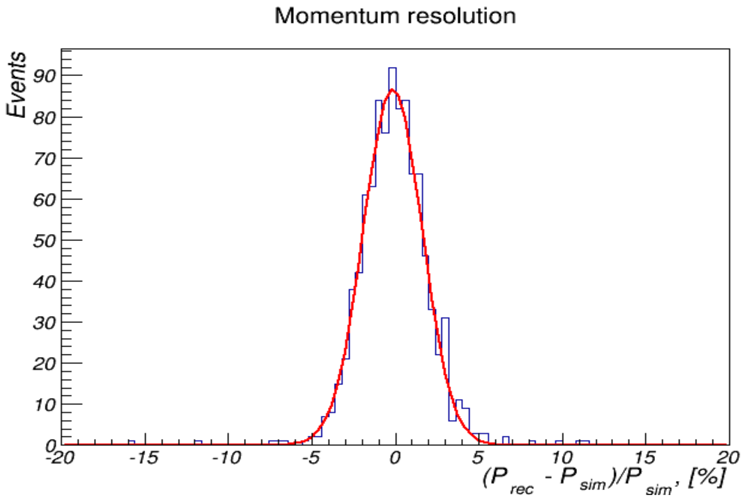
Physics analysis

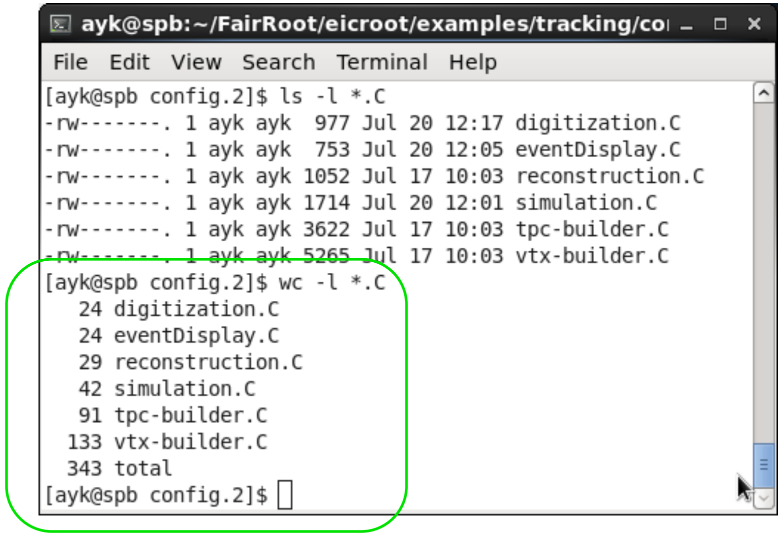


EicRoot: Example tracking study

Consider vertex tracker + TPC in 3T field; shoot 10 GeV/c pions at θ=75°



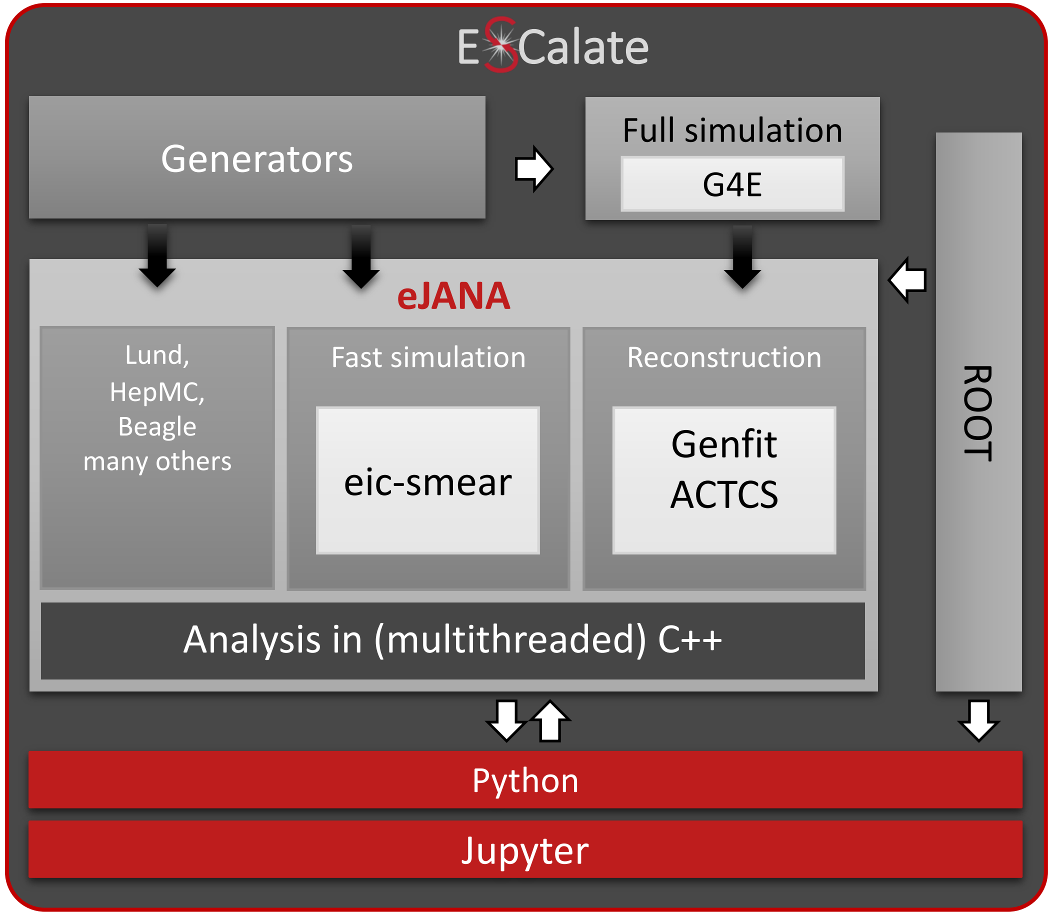




-> see examples/tracking/config.2 directory for details

- Once Docker image is downloaded it takes <5 minutes to generate this plot

Reconstruction in ESCalate



Reconstruction software

- Implemented vertexing with ACTS + GENFIT for tracking.
- Expanding to full ACTS tracking.

ESCalate Reconstruction Example: Simplified tracking efficiency analysis

```
class TrackingEfficiencyProcessor : public JEventProcessor {

private:
    std::shared_ptr<JGlobalRootLock> m_lock;
    TH1D* h1d_pt_reco;

    // More histograms

public:
    void Init() {

        m_lock = GetApplication()->GetService<JGlobalRootLock>();
        h1d_pt_reco = new TH1D("pt_reco", "reco pt", 100,0,10);
    }

    void Process(const std::shared_ptr<const JEvent>& event) {

        auto reco_tracks = event->Get<RecoTrack>();
        auto mc_tracks = event->Get<McTrack>();

        m_lock->acquire_write_lock();

        for (auto reco_track : reco_tracks) {
            h1d_pt_reco->Fill(reco_track->p.Pt());
        }

        m_lock->release_lock();
    }
```

Note that this processor knows nothing about which factory created the RecoTracks.

Let’s only use Genfit:

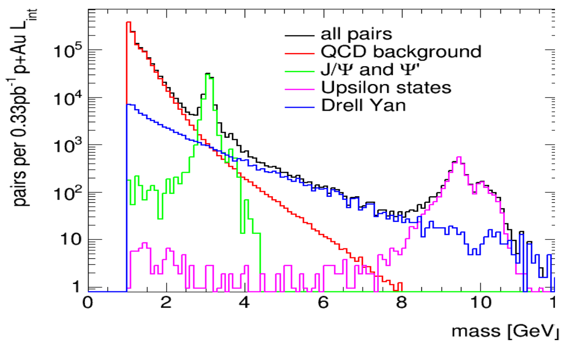
```
$ ejana
-Pplugins=g4e_reader,trk_fit,trk_eff,trk_fit_genfit
g4e_output.root
```

Let’s only use ACTS:

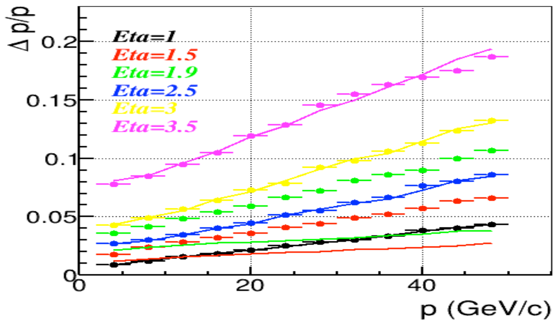
```
$ ejana
-Pplugins=g4e_reader,trk_fit,trk_eff,trk_fit_acts
g4e_output.root
```

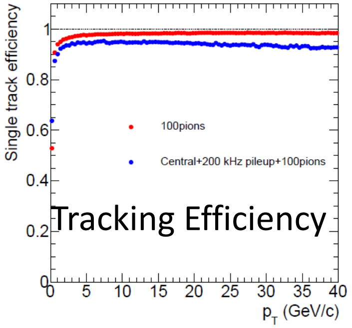
Fun4All Reconstruction

Forward Quarkonia

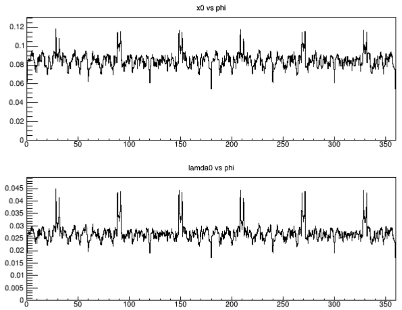


Forward Momentum Resolution



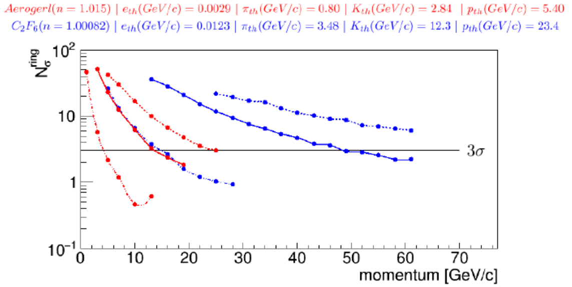


Tracking Efficiency

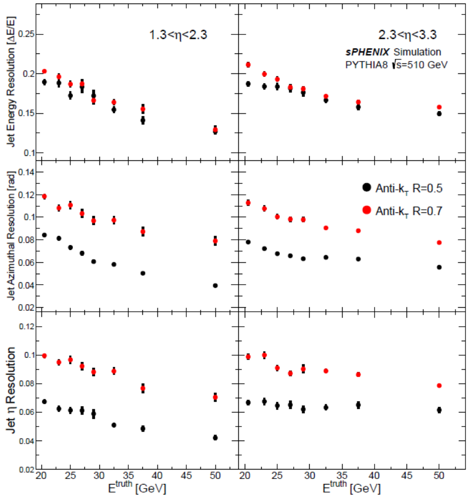


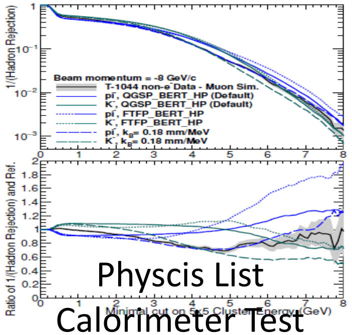
Material budget of inner tracker

Rich PID



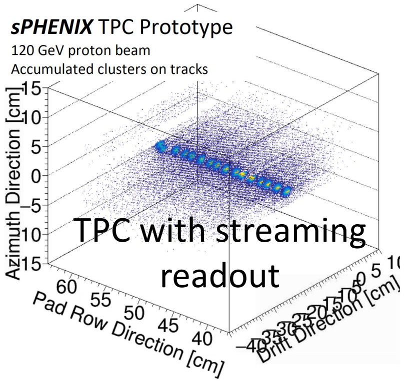
Forward Jet Energy Resolution





Physcis List

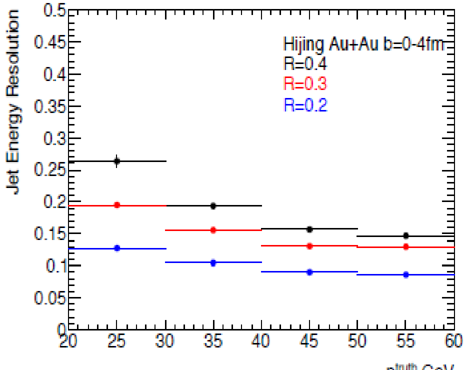
Calorimeter Test



TPC with streaming readout

- Modularity allows easy re-use of existing components
- Many existing reconstruction modules exist (digitization, granularity, clustering, tracking, jet finding, secondary vertices…)
- Interface to raw data from rcdaq (used in test eRD beams)
- Single Chain from event generator/raw data to final user output, no switching frameworks and impedance mismatches
- Snapshots of chain can be saved, chain can pick up from there
- Large and active user base

Central Barrel Jet Energy Resolution



Planning

Next Steps

Coordination and communication

- **Survey on simulation needs**
  - Please send us your answers by Friday.
  - Results shown in Q&A session on Saturday.

- **Fast Simulation Tutorial II**: MCEG, eic-smear, JupyterLab:
  - Thursday, April 9. Time will be announced.
  - Remote access via BlueJeans.
  - Will be recorded.

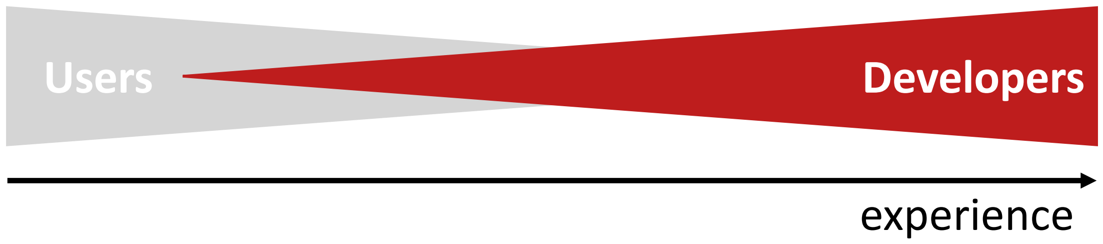
ELECTRON ION COLLIDER USER GROUP
SOFTWARE NEWS

- **Software News**
  - start making announcements on eicug-software@eicug.org, please subscribe
  - share summary on eicug-users@eicug.org and via monthly newsletter

Support

support team

Please help!



software-support@eicug.org

Mailing list (anyone can contact)
Google forum (for archive of support requests and start of knowledge base)

http://eicug.slack.com/

EICUG Slack workspace with software-support channel

EICUG Software Working Group

Andrea Bressan (INFN, University of Trieste)
Markus Diefenthaler (EIC^2, Jefferson Lab)
Torre Wenaus (Brookhaven Lab)
eicug-software@eicug.org

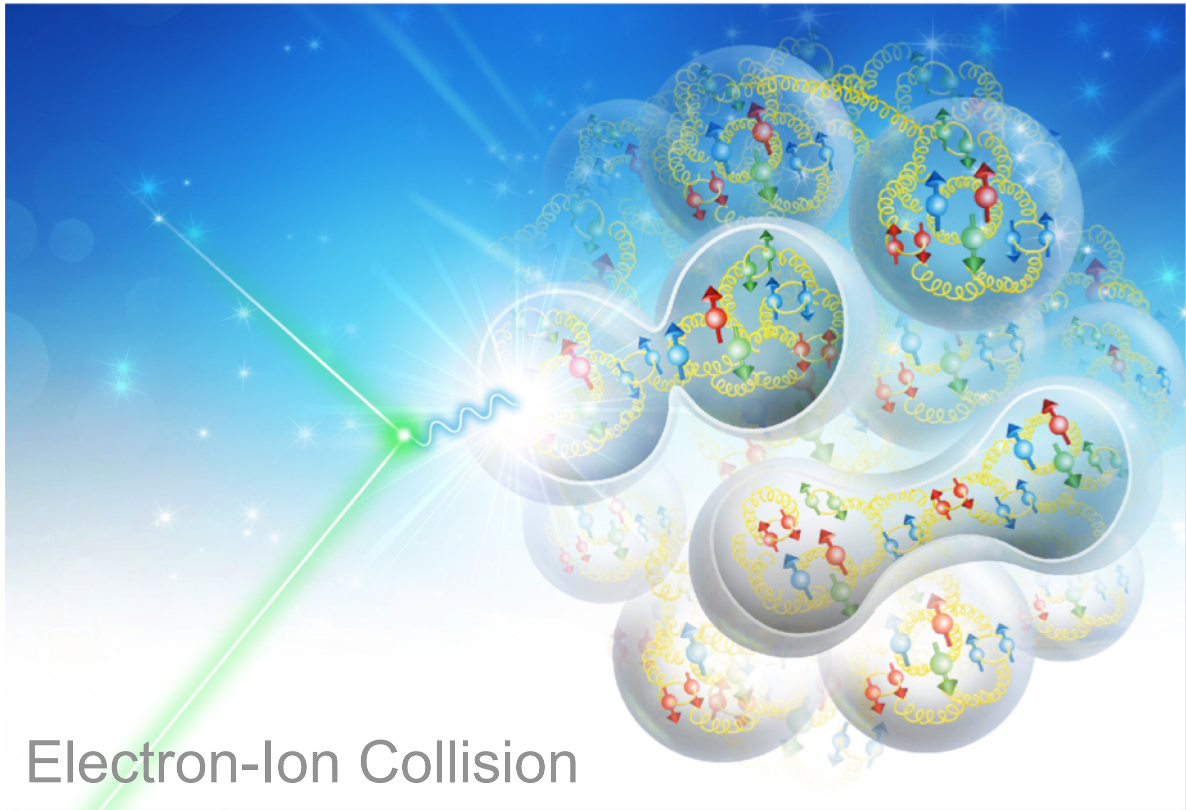
Workflow environment for EICUG

- **fast and full simulation** available and being extended with community input
- **documentation** started and being improved with community input
- **Support** will be available

Grow with user input

- excited to be core part EIC Physics and Detector Conceptual Development / Yellow Report

Q & A session on Saturday



BROOKHAVEN NATIONAL LABORATORY

Jefferson Lab

UNIVERSITÀ DEGLI STUDI DI TRIESTE