Eic-Smear: Status and Plans

Known Issues	Status
Bug undoes p_T, p_z smearing	See last slide, if there's time
Quirky behavior in example BeAST	Existing implementation was special purpose, testing general purpose version
Particle loss reported by Dima	Can't replicate – is this still an issue?

On-going development	Details
Expanded PID support	Solicit matrices; add support for dE/dx, maybe others?
Expanded acceptance	Solicit acceptance parameterization, potentially tweak Acceptance classes
Pythia8 support	Duplicate existing machinery that runs pythia6 via ROOT inside the framework – side-steps ascii file generation

Foreseen development, depending on priorities and demand:

- HepMC input (and output?) – could profit from existing code in eJANA, DELPHES, my own work
- Granularity
- (Displaced) vertex smearing
- the unknown unknown

Thoughts on transition or coexistence with DELPHES

During the Yellow Report effort, smearer(s)

- **must** support existing Fortran MCEGs
- **should not** upend large existing work and codebase
- **should** enforce unity: Result of ReferenceDetector2T 1.4 cannot be different whether using framework A, B, or C

A potential path, as I see it:

- Eic-smear remains the default and reference, **but**
- with support from a DELPHES expert (Hi Miguel ☺), DELPHES@EIC starts life as a side project as time allows
- First step only needs a DelphesReader for `BuildTree()` trees – I should be able to deliver one pretty quickly
- Expert is in charge of translating eic-smear macros to Tcl cards
  - Once consistency is demonstrated, DELPHES can become part of the software package
  - Repeat this step for every versioned detector
- If widely accepted (future poll?), priority can flip. But that's a dangerous step deep into the YR program

Do we have time to talk about the one bug?

Smeared particles have independent fields

for p, ϕ, θ, pT, px, py, pz, that can be smeared independently

- that's not a bug! Allows flexibility if you know what you're doing

```
173    Double32_t px;          ///< x component of particle momentum
174    Double32_t py;          ///< y component of particle momentum
175    Double32_t pz;          ///< z component of particle momentum
176    Double32_t E;           ///< Energy of particle
177    Double32_t pt;          ///< Transverse momentum of particle
178    Double32_t p;           ///< Total momentum of particle
179    Double32_t theta;       ///< Polar angle
180    Double32_t phi;         ///< Azimuthal angle
```

Bug: At some point, the original developer apparently decided against potential inconsistencies.
Result: pT and pz smearing are effectively ignored.

```
146    prtOut->px = prtOut->p * sin(prtOut->theta) * cos(prtOut->phi);
147    prtOut->py = prtOut->p * sin(prtOut->theta) * sin(prtOut->phi);
148    prtOut->pt = sqrt(pow(prtOut->px, 2.) + pow(prtOut->py, 2.));
149    prtOut->pz = prtOut->p * cos(prtOut->theta);
```

Question to this group: What is the preferred fix?

1. Restore flexibility → allows careless users to shoot themselves in the foot
2. Maintain only a consistent three vector → Removes flexibility (which currently isn't used or working)
   a. Details? Does smeared pT conserve |p| or theta?
3. Allow both, with switches and expert options → Fiddly, for a mostly unused cause