WARWICK
THE UNIVERSITY OF WARWICK

Raptorr/TREx interface code

9th January 2019, HPTPC Raptorr hackathon
Jennifer Haigh

Requirements

- Need to be able to run TREx reconstruction code over HPTPC CCD images. New code to:
  - read in CCD images from a Raptorr event and convert nontrivial pixels to TREx hits (TRexLoader);
  - pass hits to TREx library code and steer it to perform the reconstruction;
  - get TREx output objects and convert to Raptorr products, and put into the event.

- Need to be able to use files with reconstructed objects on systems which do not have a copy of the reconstruction. Raptorr build cannot require TREx libraries.
  - Cmake build configured to propagate build to reco directory iff TREXSYS environment variable is set.
  - New RecoHit, RecoTrack, RecoVertex classes in Raptorr obj directory are used for output. These don’t depend on any TREx code.

TRExLoader class

- Uses two config parameters, “ccdlabel” and “chargeThreshold”.
- Uses standard Raptorr interface to pull in rapobj::FakeCCDImage objects from an existing event.
  - Presently, selects only the first image with the correct type and label (from ccdlabel parameter). No attempt to stitch the images.
  - Aware that input format has since changed and code needs updating.
- Just iterate over all bins in the CCD histogram, and generate a trex::TTPCHitPad for every bin with a content above the value of the chargeThreshold parameter.
- Class has placeholders to true hits and tracks, but nothing currently implemented since there is no truth matching in the input data at present (obviously would only be for MC anyway).

I/O classes

```
namespace rapobj{
  class RecoHit {
    public:
      RecoHit() : fPosition(0,0,0), fCharge(0), fTime(0){}

      RecoHit(const TVector3& position, double charge, double time=0.) :
        fPosition(position), fCharge(charge), fTime(time) {}

      TVector3 GetPosition() const {return fPosition;}
      double GetCharge() const {return fCharge;}
      double GetTime() const {return fTime;}

    private:
      TVector3 fPosition;
      double fCharge;
      double fTime;

      ClassDef(RecoHit,1);
  };
```

Hit is just position, charge, time

```
namespace rapobj{
  class RecoVertex {
    public:
      RecoVertex(){}

    private:

      ClassDef(RecoVertex,1);
  };
```

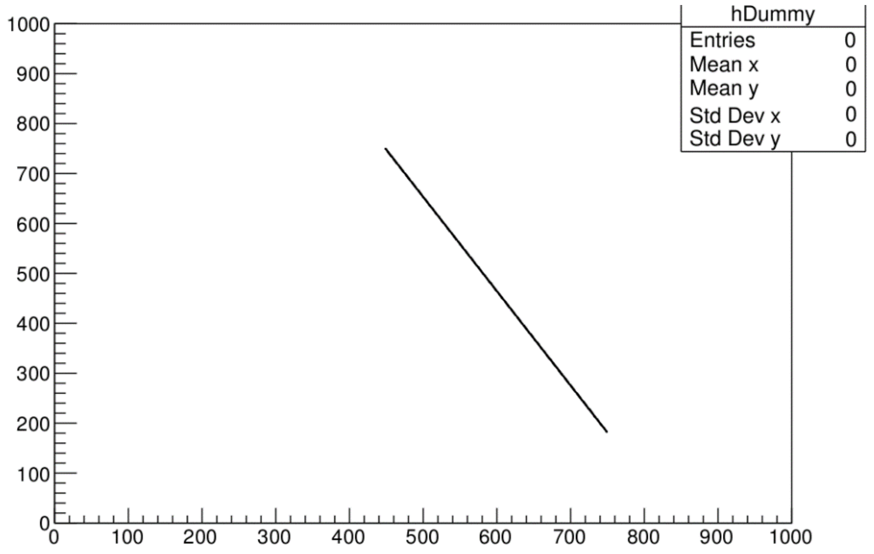
Vertex class is trivial; all information encoded in relationships

```
namespace rapobj{
  class RecoTrack {
    public:
      RecoTrack(){}

    //Reconstruction information
    void SetdEdx(double val){fdEdx=val;}
    void SetLength(double val){fdEdx=val;}
    void SetChargeSum(double val){fdEdx=val;}
    void SetPID(int val){fPID=val;}

    double GetdEdx() const {return fdEdx;}
    double GetLength() const {return fLength;}
    double GetChargeSum() const {return fChargeSum;}
    int GetPID() const {return fPID;}

    //Truth matching quantities
    void SetCompleteness(double val){fdEdx=val;}
    void SetCleanliness(double val){fdEdx=val;}
    void SetNTrueHitsFound(int val){fNTrueHitsFound=val;}

    double GetCompleteness() const {return fCompleteness;}
    double GetCleanliness() const {return fCleanliness;}
    int GetNTrueHitsFound() const {return fNTrueHitsFound;}

    private:
    double fdEdx;
    double fLength;
    double fChargeSum;
    int fPID;
    double fCompleteness;
    double fCleanliness;
    int fNTrueHitsFound;

  ClassDef(RecoTrack,1);
  };
```

Track contains metadata but hit ownership/vertex connections coded using raptorr relationships

rapRunTREx.exe

- Executable using standard Raptorr event loop logic to run CCD events through the reconstruction.
  - No physics tweaks were made to TREx as part of Raptorr integration.
- Looks for input parameters “tracklabel”, “vertexlabel”, “hitlabel” to decide what to call output objects.
- Vertices are parents of all connected tracks, and of hits in the vertex region. Tracks are parents of all hits along themselves.
- Only tested on Dom’s fake data so far.
  - Single tracks reliably reconstructed.


hDummy
Entries 0
Mean x 0
Mean y 0
Std Dev x 0
Std Dev y 0