Mu2eG4 Project Progress

Yuguang Chen

Project Starting Point

- g4test03 seems to have contained most first-step information/calculations
- Its main work:
  1. Creates an empty event;
  2. Runs the module EventGenerator which contains:

     mu2e conversion electrons *

     p from μ/π capture
- It's located at Offline/Mu2eG4/src/g4test_03.fcl

  Easy edit:
  ```
  root -l
  new TBrowser();
  ```
  *When fiddling around, don't open *.pdf file. This crashes the Root.
- The following work will be based on g4test_03

Project Starting Point

- The G4 code has most physics process turned on, including decays scatterings, energy loss, hadronic interactions, etc.. The code that builds the Mu2e world inside G4 is in:

  ~/Mu2eG4/src/Mu2eWorld.cc

  It may need a fix:

  delete _rhs; //FIXME: avoid the delete (Line 361)

Different types/functions of files

- Source file: *.cc

  Buildup the physics, e.g. ~src/Mu2eWorld.cc (is able to edit through TBrowser)

  to compile the source code:

  ```
  scons lib/lib*_module.so
  ```

  or

  ```
  scons -j 4
  ```

- Configuration file: *.fcl (is able to edit through TBrowser)

  This is a set of instructions for the framework, e.g. ~fcl/beamline.fcl

  Can generate *.root files:

  ```
  mu2e -c *fcl >& *.log
  ```

- Histogram file: *.root, *.cint (is able to edit through TBrowser)

  Browse the ntuple/histogram/plots, etc., e.g. g4study.cint

  to browse the file:

  ```
  browse *.root
  root -l *.cint
  ```

  or

  ```
  new TBrowser();
  ```

- TBrowser can also edit and compile the macro, so just use it!

The project structure

- The Mu2eG4/src/Mu2eWorld.cc defines the physics. Here is where we can change the physics, say, geometric parameters and relative rates.
- g4test_03.fcl runs 200 events, reads hits out of them and stores the ntuple in g4test_03.root

  Event generator: StoppedMuonConversionGun (should've applied genconfig_*.txt)

  genconfig_01.txt enables pi capture and cosmic rays
- The Root defined script g4test_03.cint currently reads g4test_03.root to run root and put histograms/scatter plots into a pdf file.

  There is another possibly easier way to complete the task

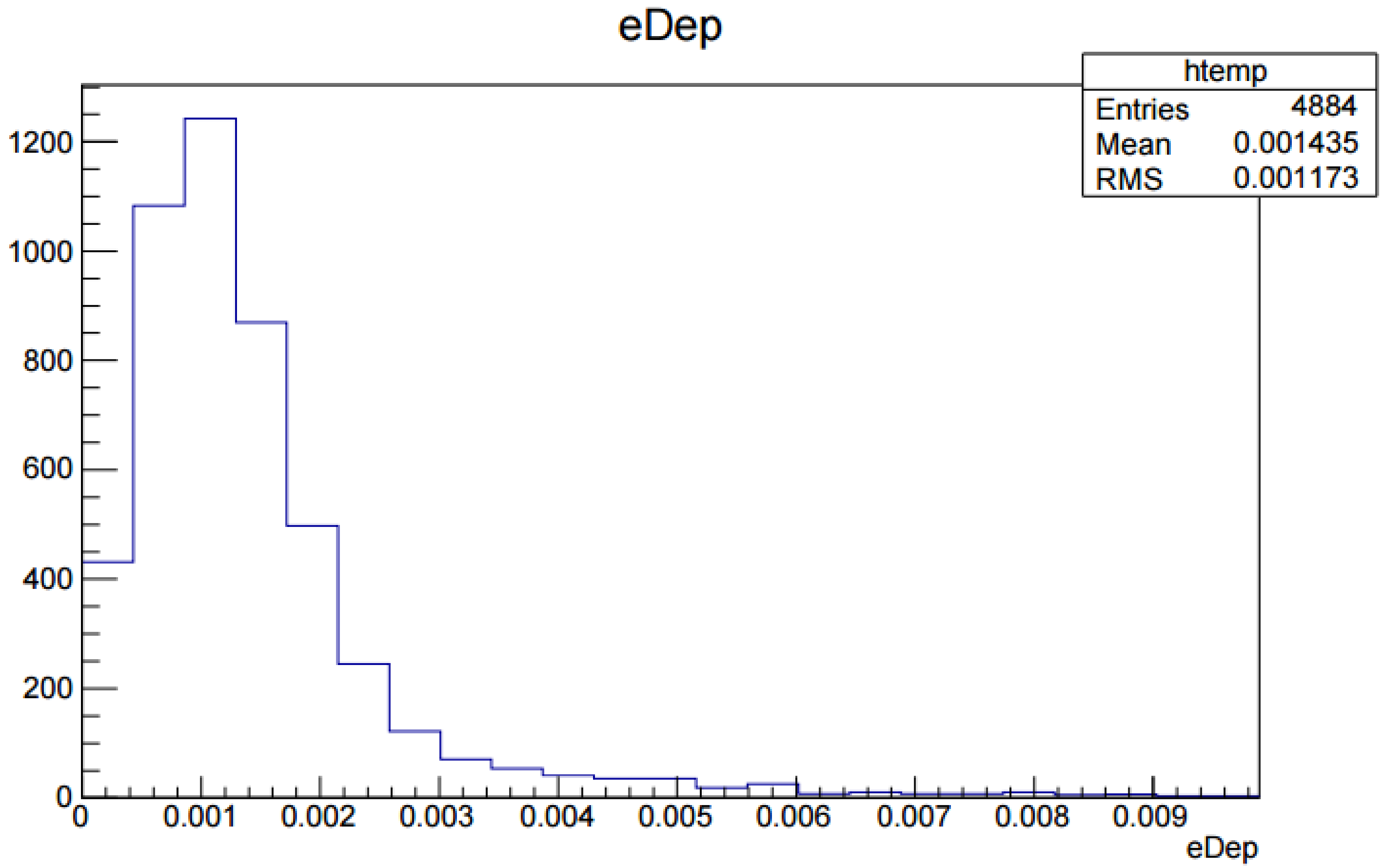
The project structure -- Some Histograms

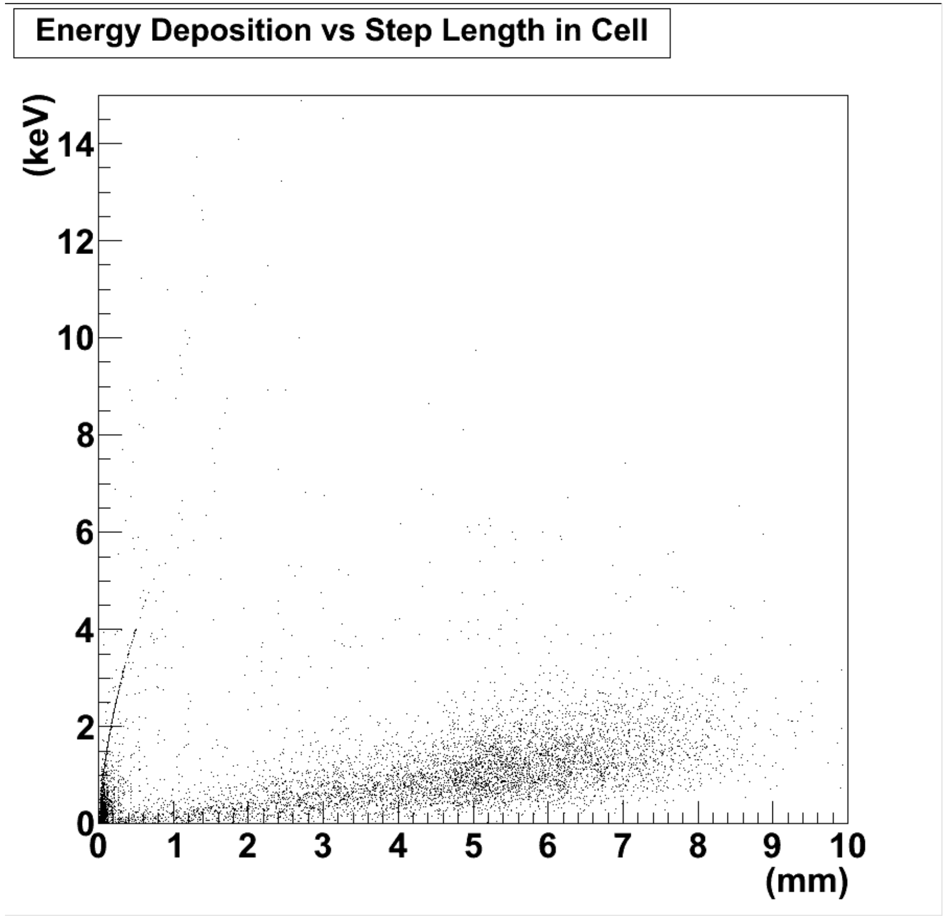
- Run the following code to generate .root histograms:

```
mu2e -c Mu2eG4/fcl/g4test_03.fcl >& ~/g4test_03.log
```

- The histograms can be browsed in TBrowser

The project structure -- Some Histograms





The project structure

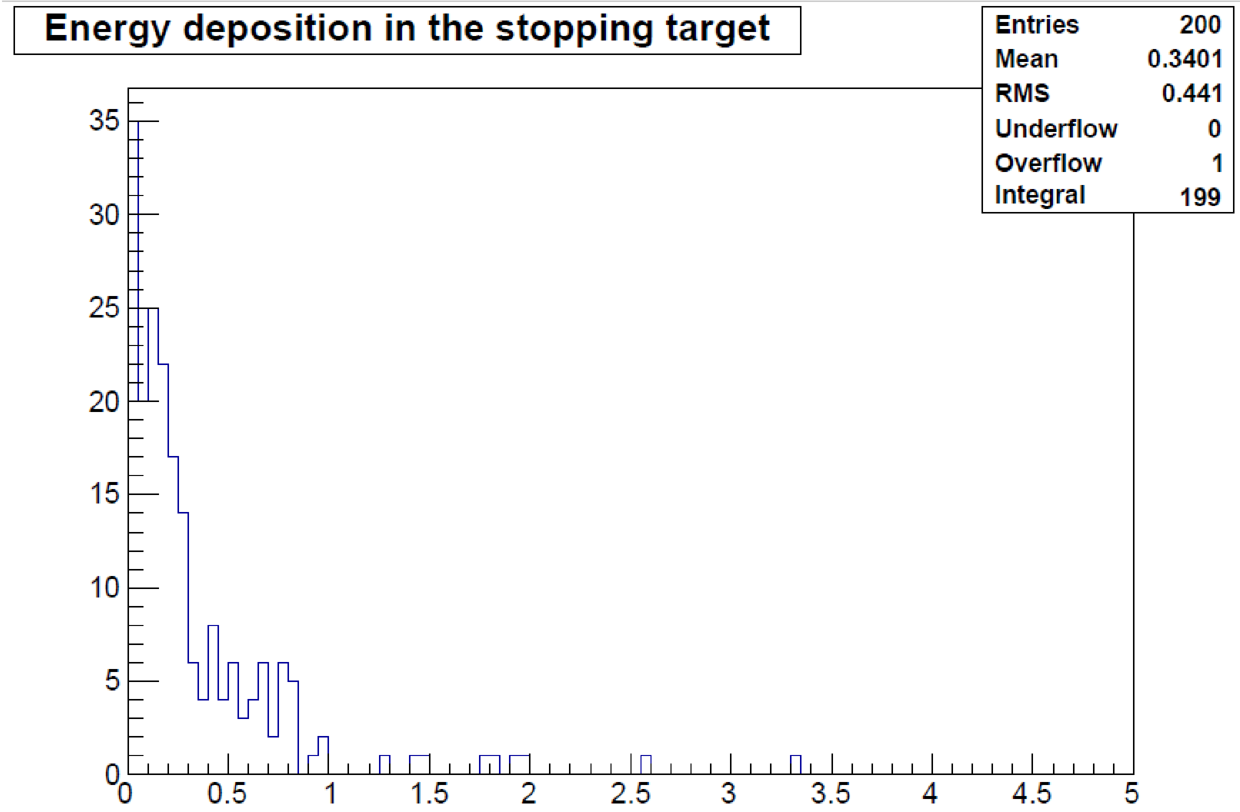
- Analyses/src/ReadBack_module.cc reads back the hits created by G4 and makes histograms.

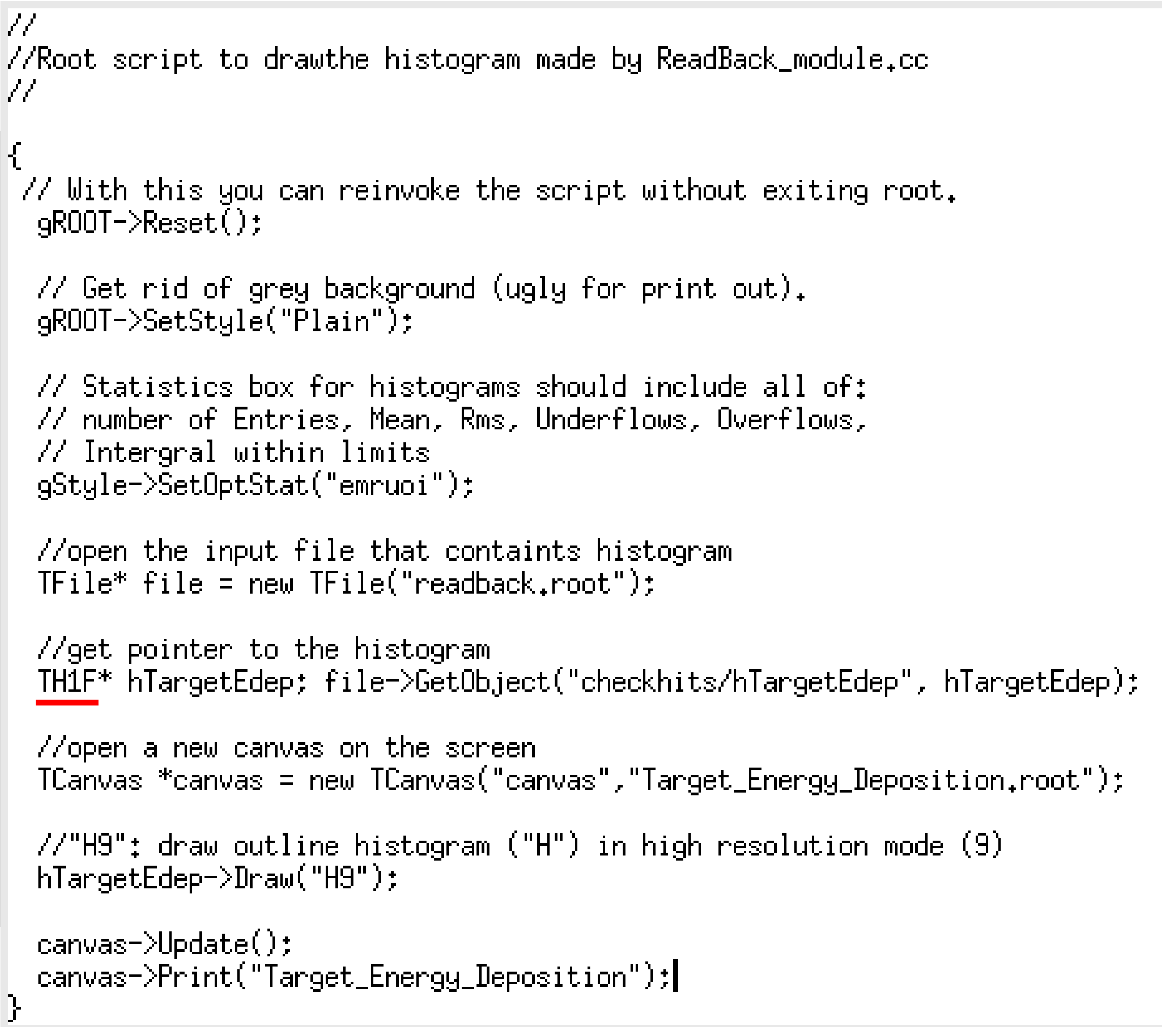
  ```
  mu2e -c Analyses/test/readback.fcl
  ```

  This reads back the data file made by g4test_03.fcl and remakes the histograms and ntuples
- We can edit g4test_03.cint to instead read readback.root (switch the base name)

  Its *.cc source has more information extracted from the data and the source is open with clear instructions at: Here
- Our work will simply be changing the ReadBack file (.cc, .fcl) or the g4test_03.cint file

The project structure -- Some Histograms





```
//
//Root script to drawthe histogram made by ReadBack_module.cc
//

{
 // With this you can reinvoke the script without exiting root.
  gROOT->Reset();

  // Get rid of grey background (ugly for print out).
  gROOT->SetStyle("Plain");

  // Statistics box for histograms should include all of:
  // number of Entries, Mean, Rms, Underflows, Overflows,
  // Intergral within limits
  gStyle->SetOptStat("emruoi");

  //open the input file that containts histogram
  TFile* file = new TFile("readback.root");

  //get pointer to the histogram
  TH1F* hTargetEdep; file->GetObject("checkhits/hTargetEdep", hTargetEdep);

  //open a new canvas on the screen
  TCanvas *canvas = new TCanvas("canvas","Target_Energy_Deposition.root");

  //"H9": draw outline histogram ("H") in high resolution mode (9)
  hTargetEdep->Draw("H9");

  canvas->Update();
  canvas->Print("Target_Energy_Deposition");
}
```

How to make histograms

- Some more instructions, see Here, Chapter 17: Making a Histogram