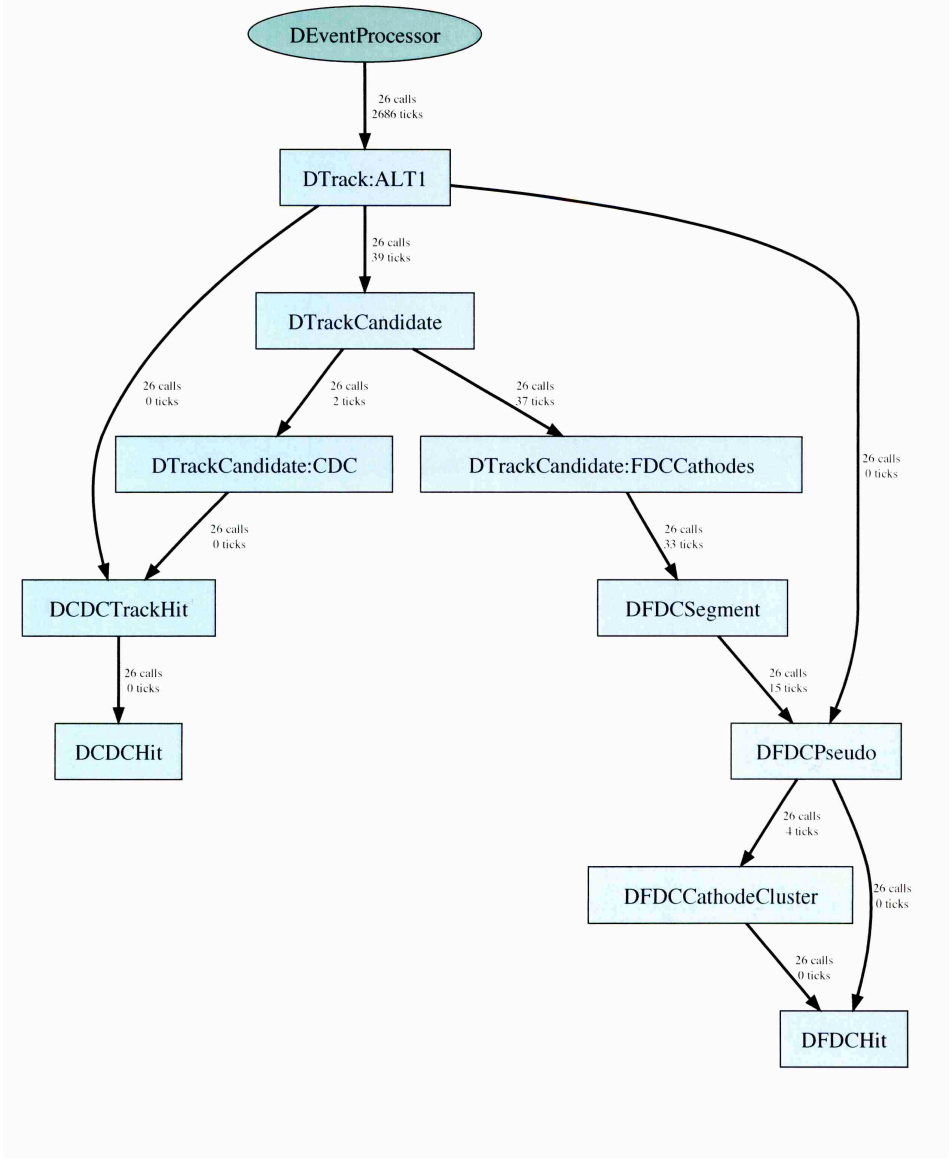
Highlights of Tracking Related Work Done Since March Tracking Review

May 16, 2008 David Lawrence

Modularization of Tracking Code

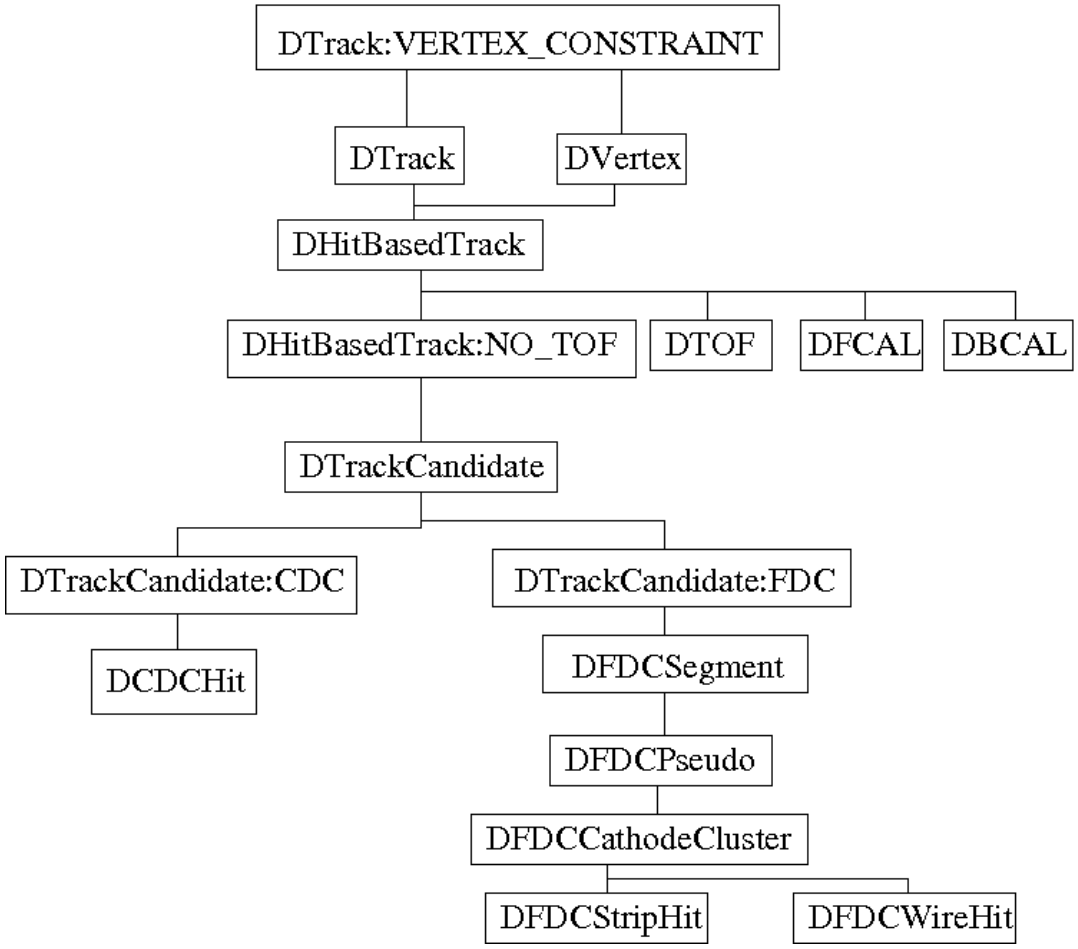


On April 4th Simon, Mark and Dave met to discuss how the information flow for track reconstruction could best be broken up into “factories”.

The diagram to the left shows the current state of the code as observed by JANA.

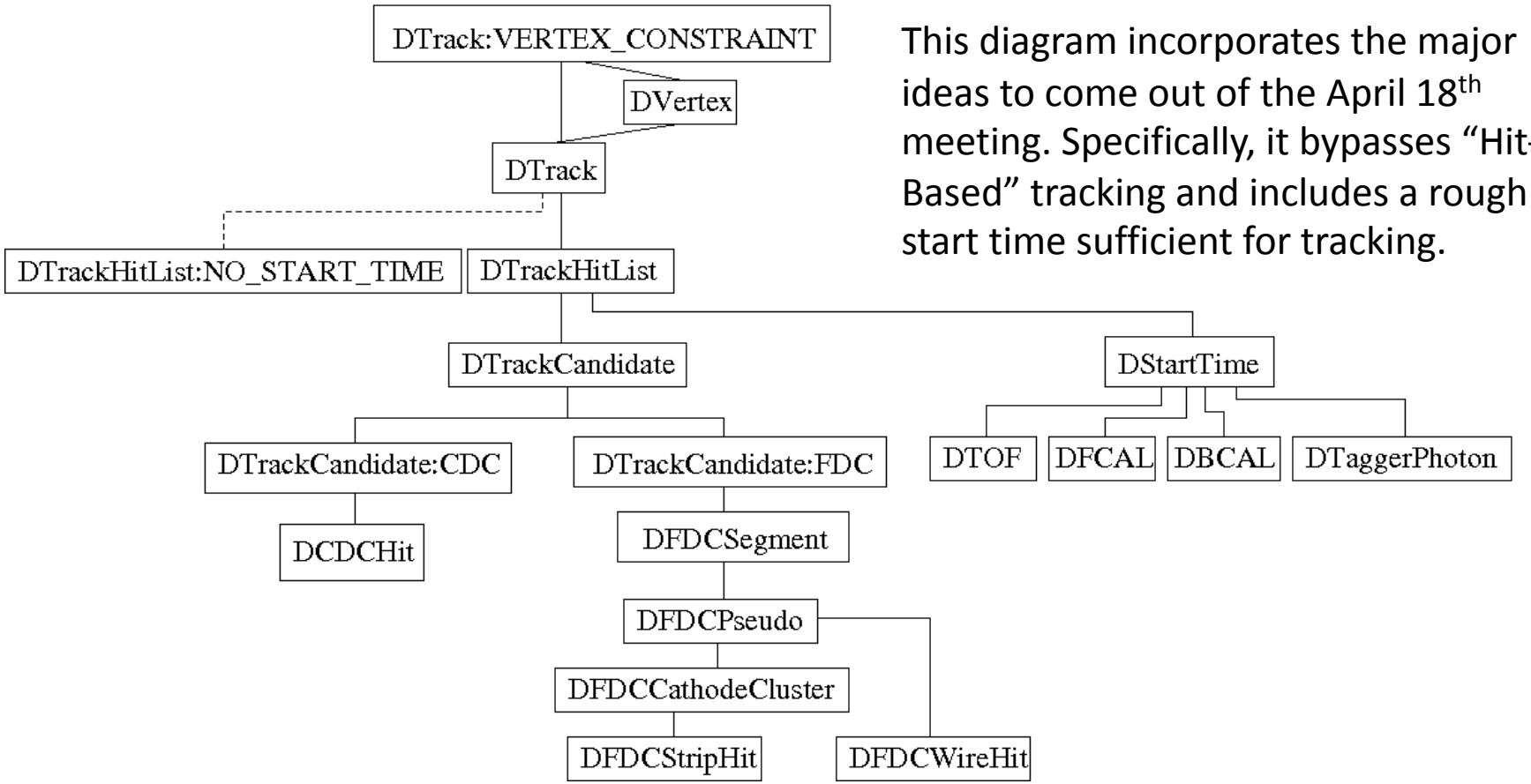
The main problem with the current design is that it does not allow a clean and obvious way to include the start time that will be needed before the final tracking (using drift times) can be done.

Modularization of Tracking Code cont.



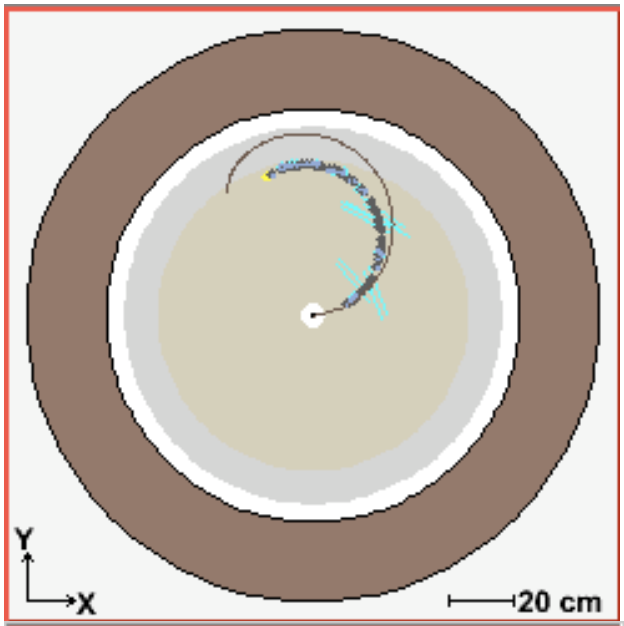
This diagram was presented by David as a starting point to the discussion. It was scribbled on a bit and at the end, several significant suggestions were made on how it might be improved….

Modularization of Tracking Code cont.

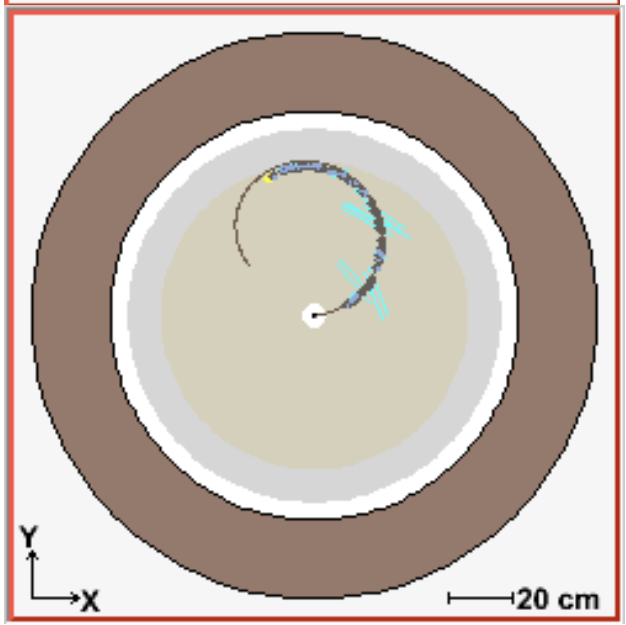


This diagram incorporates the major ideas to come out of the April 18th meeting. Specifically, it bypasses "Hit-Based" tracking and includes a rough start time sufficient for tracking.

Utilize B-Field Map in CDC Track Finder



Uniform B-field



Space-Point Averaged B-field

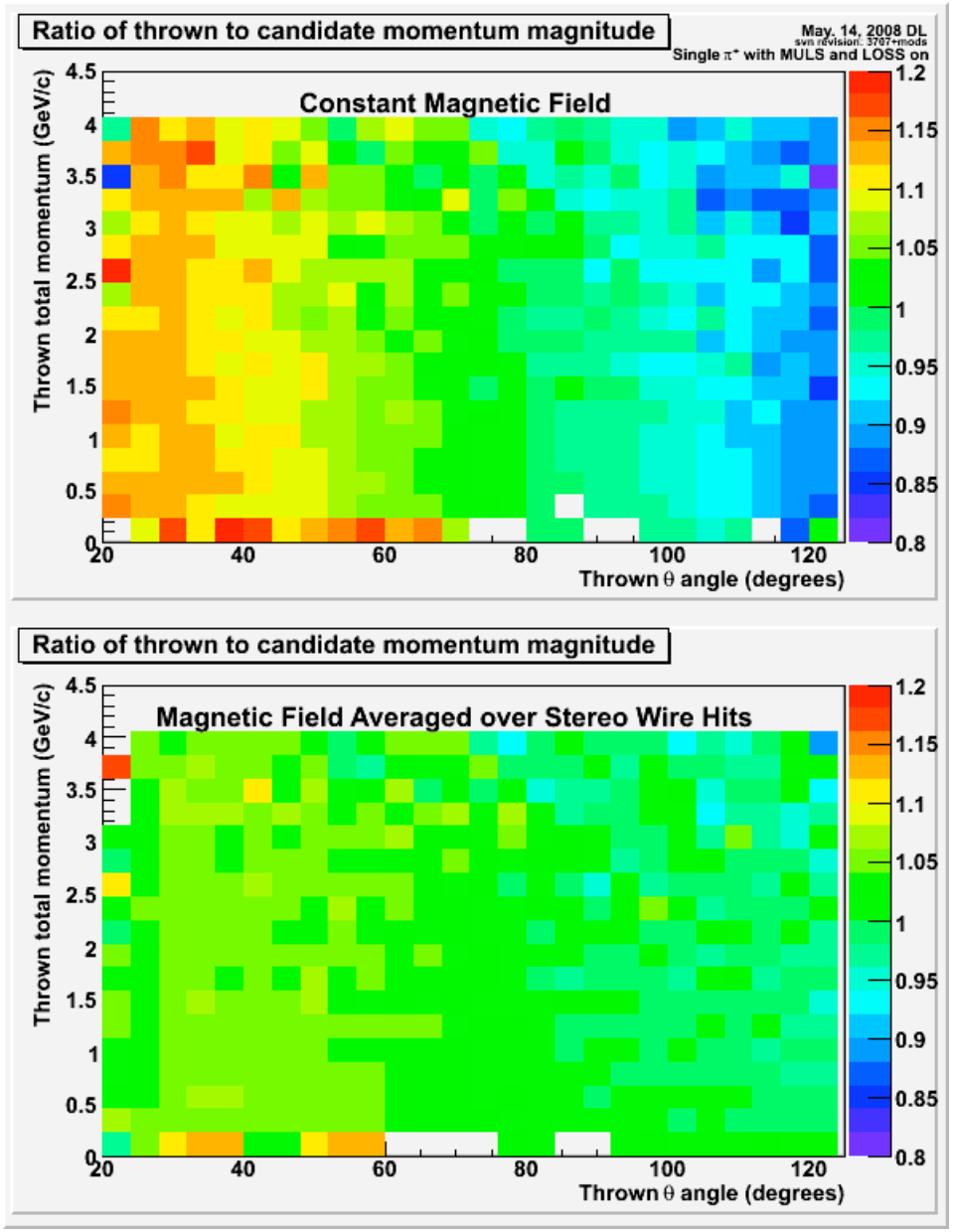
One of the priorities for tracking reconstruction is to improve the robustness of the current finder and fitter.

Part of this includes improving the quality of the candidate parameters coming from the CDC track finder.

The quality is somewhat limited by the “large” field gradient in the CDC. Until now, the parameters assumed a uniform field of -2.0T.

Recent changes scale the momentum by the average field at each of the stereo hits.

Utilize B-Field Map in CDC Track Finder

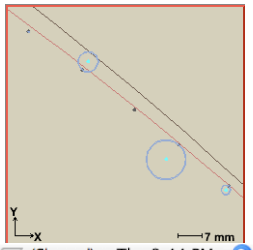


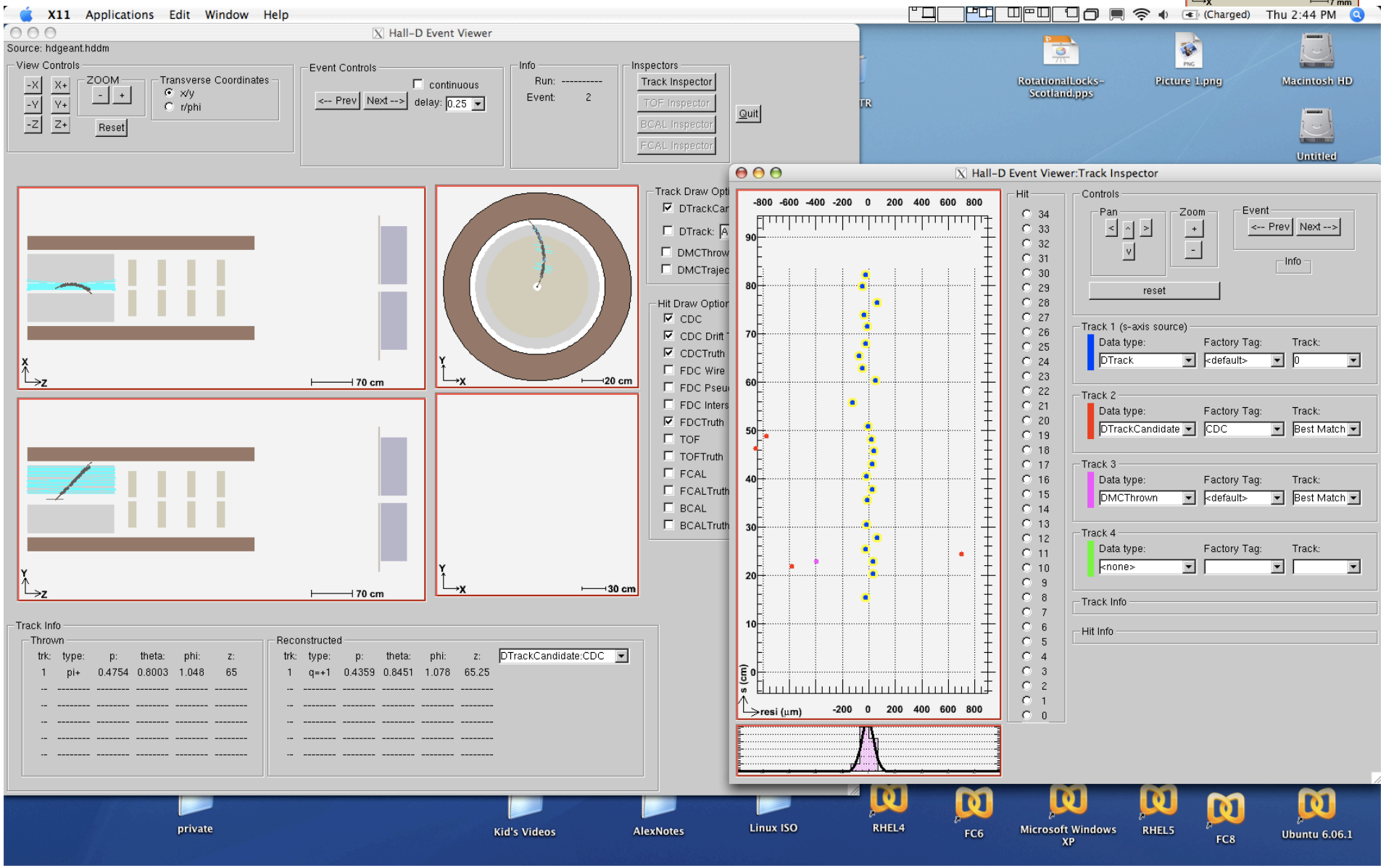
Uniform B-field

Space-Point Averaged B-field

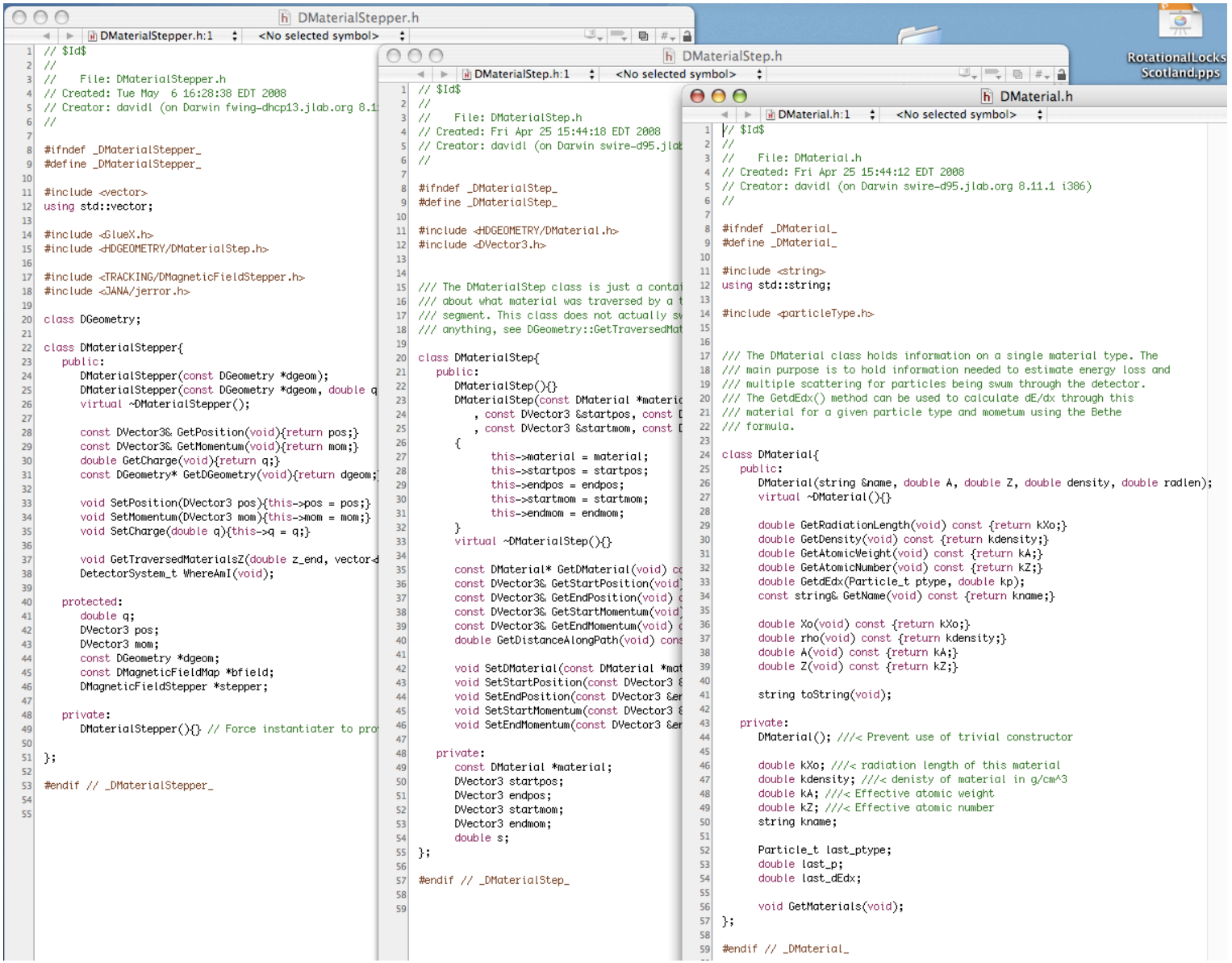
These two plots indicate the determination of the total momentum in the CDC track candidates when using a uniform and space-point averaged value for the magnetic field.

New Track Inspector in *hdview2*





Started Implementing DMaterial classes



DMaterialStepper.h

```
// $Id$
//
//    File: DMaterialStepper.h
// Created: Tue May  6 16:28:38 EDT 2008
// Creator: davidl (on Darwin fwing-dhcp13.jlab.org 8.1
//

#ifndef _DMaterialStepper_
#define _DMaterialStepper_

#include <vector>
using std::vector;

#include <GlueX.h>
#include <HDGEOMETRY/DMaterialStep.h>

#include <TRACKING/DMagneticFieldStepper.h>
#include <JANA/jerror.h>

class DGeometry;

class DMaterialStepper{
   public:
      DMaterialStepper(const DGeometry *dgeom);
      DMaterialStepper(const DGeometry *dgeom, double q
      virtual ~DMaterialStepper();

      const DVector3& GetPosition(void){return pos;}
      const DVector3& GetMomentum(void){return mom;}
      double GetCharge(void){return q;}
      const DGeometry* GetDGeometry(void){return dgeom;

      void SetPosition(DVector3 pos){this->pos = pos;}
      void SetMomentum(DVector3 mom){this->mom = mom;}
      void SetCharge(double q){this->q = q;}

      void GetTraversedMaterialsZ(double z_end, vector<
      DetectorSystem_t WhereAmI(void);

   protected:
      double q;
      DVector3 pos;
      DVector3 mom;
      const DGeometry *dgeom;
      const DMagneticFieldMap *bfield;
      DMagneticFieldStepper *stepper;

   private:
      DMaterialStepper(){} // Force instantiater to pro

};

#endif // _DMaterialStepper_
```

DMaterialStep.h

```
// $Id$
//
//    File: DMaterialStep.h
// Created: Fri Apr 25 15:44:18 EDT 2008
// Creator: davidl (on Darwin swire-d95.jlab
//

#ifndef _DMaterialStep_
#define _DMaterialStep_

#include <HDGEOMETRY/DMaterial.h>
#include <DVector3.h>


/// The DMaterialStep class is just a contai
/// about what material was traversed by a t
/// segment. This class does not actually s
/// anything, see DGeometry::GetTraversedMat

class DMaterialStep{
   public:
      DMaterialStep(){}
      DMaterialStep(const DMaterial *materi
         , const DVector3 &startpos, const D
         , const DVector3 &startmom, const D
      {
            this->material = material;
            this->startpos = startpos;
            this->endpos = endpos;
            this->startmom = startmom;
            this->endmom = endmom;
      }
      virtual ~DMaterialStep(){}

      const DMaterial* GetDMaterial(void) co
      const DVector3& GetStartPosition(void)
      const DVector3& GetEndPosition(void) c
      const DVector3& GetStartMomentum(void)
      const DVector3& GetEndMomentum(void) c
      double GetDistanceAlongPath(void) cons

      void SetDMaterial(const DMaterial *mat
      void SetStartPosition(const DVector3 &
      void SetEndPosition(const DVector3 &en
      void SetStartMomentum(const DVector3 &
      void SetEndMomentum(const DVector3 &en

   private:
      const DMaterial *material;
      DVector3 startpos;
      DVector3 endpos;
      DVector3 startmom;
      DVector3 endmom;
      double s;
};

#endif // _DMaterialStep_
```

DMaterial.h

```
// $Id$
//
//    File: DMaterial.h
// Created: Fri Apr 25 15:44:12 EDT 2008
// Creator: davidl (on Darwin swire-d95.jlab.org 8.11.1 i386)
//

#ifndef _DMaterial_
#define _DMaterial_

#include <string>
using std::string;

#include <particleType.h>


/// The DMaterial class holds information on a single material type. The
/// main purpose is to hold information needed to estimate energy loss and
/// multiple scattering for particles being swum through the detector.
/// The GetdEdx() method can be used to calculate dE/dx through this
/// material for a given particle type and mometum using the Bethe
/// formula.

class DMaterial{
   public:
      DMaterial(string &name, double A, double Z, double density, double radlen);
      virtual ~DMaterial(){}

      double GetRadiationLength(void) const {return kXo;}
      double GetDensity(void) const {return kdensity;}
      double GetAtomicWeight(void) const {return kA;}
      double GetAtomicNumber(void) const {return kZ;}
      double GetdEdx(Particle_t ptype, double kp);
      const string& GetName(void) const {return kname;}

      double Xo(void) const {return kXo;}
      double rho(void) const {return kdensity;}
      double A(void) const {return kA;}
      double Z(void) const {return kZ;}

      string toString(void);

   private:
      DMaterial(); ///< Prevent use of trivial constructor

      double kXo; ///< radiation length of this material
      double kdensity; ///< denisty of material in g/cm^3
      double kA; ///< Effective atomic weight
      double kZ; ///< Effective atomic number
      string kname;

      Particle_t last_ptype;
      double last_p;
      double last_dEdx;

      void GetMaterials(void);
};

#endif // _DMaterial_
```