Description of the Fermilab software school data objects

Matthew Herndon[1]

[1] *University of Wisconsin, Madison, Wisconsin, Fermi National Accelerator Laboratory, Illinois*

This document describes the data objects used in the Fermilab software school. The objects include a StripSet with raw digitized data for the strip sensors of the toy simulated detector as well as Hits, Tracks, GenTracks and associated HitSets, TracksSets and GenTrackSets. In addition, the formats used to store the data objects to file are described.

I. INTRODUCTION

Each objects is described below including the object's class and the methods and format used to write the objects. Sets are described with the associated data object.

The detector geometry of the strip sensors is defined in [1].

II. THE STRIPSET

A. StripSet class

Implemented in the StripSet.hh and StripSet.cc files in the DataObjects directories.

The StripSet contains a std::vector of typedef LayerStripMap with one map per layer of the detector. The detector is dynamically defined from a geometry file and the vector and size are initialized at run time. Each layerStripMap is a map of key int stripNumber and adc value. This allows the strip data to be stored in a sparse and ordered format.

Access methods:

- int numberLayers() To assist in iterating over the vector or directly over the layerMaps
- strips(): const reference access to the vector using a generic method. The underlying vector container could be changed to several different container types without effecting the user interface.
- layerStrips(int layer): const reference access to a the strip data from a single layer. Note that a map is retrieved since maps have access methods that are not common to all containers.
- stripNumber(const iterator) or stripNumber(map pair) to get the strip Number
- stripAdc(const iterator) or stripAdc(map pair) to get the strip ADC
- insertStrip(unsigned int layer, int stripNumber, int adc): To insert data into the set.
- print(ostream&): To print to an a user specified ostream.

B. StripSet IO

Implemented in the StripSetIO.hh and StripSetIO.cc files in the Algorithms directories.

Input and output to disk in controlled by the StripSetIO helper class. Reading and writing are performed by the readEvent and writeEvent functions.

The data is stored, or "streamed", to disk in bit packed format to illustrate a typical way date is stored in compact format in many experiments. The data is stored in individual bytes or pairs of bytes for information that does not fit in one byte. The data structure is as follows.

- Strips: In clear text to identify the Set
- 1 byte, version: The version is checked on input to make sure the correct streamer code is being used.
- - 1 byte, layer number: 0-9 (repeated once per layer)
- - 1 byte, number of strips: max 256
- – 2 bytes. strip number max 4048 (these items repeated number of strips times)
- – 1 byte. adc max 64

Note layer numbers are not necessary to the structure but are included to facilitate synchronization with write and read operation..

A number of helper functions exist in the StripHitFunctions file in Geometry directories. These functions can convert from strip number and layer to local and global positions and back again and identify whether a strip number is valid on a given layer.

C. binary input and output

Data input and output is controlled via standard C++ library binary input and output functions. std::ofstream, ifstream. Byte writing is performed for write and read functions as:

- stripdata.write reinterpret_cast const char *(&myInt), 1);
- stripdata.read (myCharByte, 1);

III. THE GENHIT AND GENHITSET

A. GenHit class

Implemented in the GenHit.hh and GenHit.cc files in the DataObjects directories.

The GenHit class contains:

- TVector3 _position
- int _layer
- Int _trackNumber

This class serves for generator GenHits only.

Constructor:

- Hit(hitPosition,layer,trackNumber)

Access methods:

- position():... names correspond to the data members
- layer()

- trackNumber()
- print(ostream&): To print to an a user specified ostream.

Accessor are named for class variables. For example position() retrieves the hit _position.
A number of helper functions exist in the StripHitFunctions file in Geometry directories. These functions can convert from hit position to local and strip number position or back identify whether a hit position at a sensor plane is within the active area of the sensor.

B. GenHitSet class

Implemented in the GenHitSet.hh and GenHitSet.cc files in the DataObjects directories.

The GenHitSet contains a typedef GenHitSetContainer of type std::vector of GenHits.

Access methods:

- genHits(): const reference access to the vector using a generic method. The vector index indexes the GenHit number to allow direct access to a hit of a known number.
- insertGenHit(GenHit): Provided to insert data into the set.
- print(ostream&): To print to an a user specified ostream that iterates over the GenHits calling the print method of each GenHit..

C. GenHitSet IO

Implemented in the GenHitSetIO.hh and GenHitSetIO.cc files in the Algorithms directories.

Input and output to disk in controlled by the GenHitSetIO helper class. Reading and writing are performed by the readEvent and writeEvent functions.

The data is stored in text format for simplicity.

- Hits: text string to identify the set
- version: The version is checked on input to make sure the correct streamer code is being used.
- number of hits
- hit number (repeated number of hits times)
- - x position
- - y position
- - z position
- - layer number
- - track number

IV. THE HIT AND HITSET

A. Hit class

Implemented in the Hit.hh and Hit.cc files in the DataObjects directories.

The Hit class contains:

- TVector3 _position
- int _layer
- int _numberStrips
- int _charge
- bool _goodHit
- double _resolution

The class server for reconstructed Hits only. Associated information such as Hit resolutions are set using information from DetectorGeometry though a resolution variable exists per hit to allow for any resolution model to be developed. Also note that a hit is classified of as "bad" if it has a large ADC value or more than two associated strips indicating that it is actually due to several overlapping hits from different tracks.

Constructors:

- Hit(hitPosition,layer,numberStrips,charge,goodhit,resolution)

Access methods:

- TVector3 position():... names correspond to the data members
- layer()
- numberStrips()
- charge()
- goodHit()
- resolution
- print(ostream&): To print to an a user specified ostream.

A number of helper functions exist in the StripHitFunctions file in Geometry directories. These functions can convert from hit position to local and strip number position or back identify whether a hit position at a sensor plane is within the active area of the sensor.

B. HitSet class

Implemented in the HitSet.hh and HitSet.cc files in the DataObjects directories.

The HitSet contains a typedef HitSetContainer of type std::vector of Hits.

Access methods:

- hits(): const reference access to the vector using a generic method. The vector index indexes the Hit number to allow direct access to a hit of a known number.
- insertHit(Hit): Provided to insert data into the set.
- print(ostream&): To print to an a user specified ostream that iterates over the Hits calling the print method of each Hit..

C. HitSet IO

Implemented in the HitSetIO.hh and HitSetIO.cc files in the Algorithms directories.

Input and output to disk in controlled by the HitSetIO helper class. Reading and writing are performed by the readEvent and writeEvent functions.

The data is stored in text format for simplicity.

- Hits: text string to identify the set
- version: The version is checked on input to make sure the correct streamer code is being used.
- number of hits
- hit number (repeated number of hits times)
- - x position
- - y position
- - z position
- - layer number
- - number of strips
- - charge
- - goodHit
- - resolution

V. THE GENTRACK AND GENTRACKSET

A. GenTrack class

Implemented in the GenTrack.hh and GenTrack.cc files in the DataObjects directories.

The GenTrack class contains:

- TLorentzVector _lorentzVector
- int _charge
- TVector3 _dr, point of closest approach to the reference point, 0,0,0

Constructor:

- GenTrack(TLorentzVector,int charge, TVector3 dr position of closest approach)
- itemize

Access:

- lorentzVector()::... names correspond to the data members
- charge()
- dR(): _dr
- makeHelix(TVector3 bField,curvatureC): function which takes the magnetic field and the curvature constant and returns a Helix which can be used for operations like finding intersections with layers. Note the magnetic field value is needed since it may not be uniform and the radius of curvature would be different at different points. With the standard bField from the DetectorGeometry you get the radius of curvature at the origin.
- print(ostream&) to print to an a user specified ostream.

B. GenTrackSet class

Implemented in the GenTrackSet.hh and GenTrackSet.cc files in the DataObjects directories.

The GenTrackSet contains a typedef GenTrackSetContainer of type std::vector of GenTracks.
Access:

- genTracks(): const reference access to the vector using a generic method. The vector index indexes the GenTrack number to allow direct access to a hit of a known number.
- insertTrack(GenTrack): Provided to insert data into the set.
- print(ostream&): to print to an a user specified ostream that iterates over the GenTracks calling the print method of each GenTrack..

C. GenTrackSet IO

Implemented in the GenTrackSetIO.hh and GenTrackSetIO.cc files in the Algorithms directories.

Input and output to disk in controlled by the GenTrackSetIO helper class. Reading and writing are performed by the readEvent and writeEvent functions.

The data is stored in text format for simplicity.

- GenTracks: text string to identify the set
- version: The version is checked on input to make sure the correct streamer code is being used.
- number of GenTracks
- - GenTrack number (repeated number of GenTrackNumber times)
- - charge
- - px
- - py
- - pz
- - E
- - x position
- - y position
- - z position

VI. THE TRACK AND TRACKSET

A. Track class

Implemented in the Track.hh and Track.cc files in the DataObjects directories.

The Track class contains:

- Helix _helix, 5 track parameter helix
- int _covMatrix, 5x5 parameter covariance matrix
- double _chi2

- int _nDof
- _trackHits, a typedef TrackHitContainer of type vector of ints with hit index numbers.

Constructor:

- Tracks are constructed by the full set of parameters above. These parameters are generated by the BuildTrack helper function in the Algorithms directories. That class fits a track based the input hits.
- note that the Helix class includes the public setHelix() expert method for use in TrackFit where the helix must be updated frequently

Access:

- helix() get Helix object with track parameters
  The Helix helix parameter follow the convention of [2] and are described in detail there.
  Note that helix parameters are always referenced from (0,0,0) in the parametrization used in this code.
- TVectorD helixParam(), access to the helix parameter vector
- Helix parameter accessor functions called from the helix as helix().dR() or directly as dR() ...:
  - dR(): Impact parameter, distance in the x-y place to the reference point (0,0,0) at the point of closest approach. Signed positive to indicate the reference point is within the circle or negative is outside.
  - phi0(): Phi angle from the reference point to the point of closest approach.
  - kappa(): inverse Pt signed to indicate handedness + right handed (negative charge) - left handed (positive charge)
  - dZ(): z distance to reference point at point of closest approach
  - tanL(): pZ/pT or cot(theta) where theta is the dip angle which is 90 for track perpendicular to the z axis and 0 degrees for parallel to the position z axis.
- Other Helix accessors available from helix()
  - alpha() get inverse curvature constant in the magnetic field at the origin
  - radiusOfCurvatureAtOrigin()
  - radiusOfCurvature(TVector3 bField) at an arbitrary point with field bField
  - pT()
  - pZ()
  - cotTheta() same as tanL
  - cosTheta() where theta is the dip angle
  - sinTheta() where theta is the dip angle
- covMatrix()
- sigmDr() ... uncertainties for the five track parameters from the covariance matrix
- charge()
- chi2()
- nDof()
- chi2Prob()
- lorentzVector() get a root TLorentzVector object based on the track parameters
- trackHits() get TrackHitContainer vector of hit indices's
- numberHits()
- print(ostream&): to print to an a user specified ostream.

B. TrackSet class

Implemented in the TrackSet.hh and TrackSet.cc files in the DataObjects directories.

The TrackSet is a typedef TrackSetContainer of type std::vector of Tracks.

Access:

- tracks(): const reference access to the vector using a generic method. The vector index indexes the Track number to allow direct access to a hit of a known number.
- insertTrack(Track): provided to insert data into the set.
- print(ostream&): to print to an a user specified ostream that iterates over the Tracks calling the print method of each Track..

C. TrackSet IO

Implemented in the TrackSetIO.hh and TrackSetIO.cc files in the Algorithms directories.

Input and output to disk in controlled by the TrackSetIO helper class. Reading and writing are performed by the readEvent and writeEvent functions.

The data is stored in text format for simplicity.

- Tracks: text string to identify the set
- version: The version is checked on input to make sure the correct streamer code is being used.
- number of tracks
- - Track number (repeated number of trackNumber times)
- - charge
- - px
- - py
- - pz
- - E
- - x position
- - y position
- - z position
- - number of Hits
- – Hit index numbers (repeated number of Hits times)

The hit list could be used alone to restore the track using BuildTrack. Currently TrackSetIO is not used.

---

[1] detectorGeometry.pdf note, M. Herndon (2014)
[2] B. Li, K. Fujii and Y. Gao, Comput. Phys. Commun. **185**, 754 (2014) [arXiv:1305.7300 [physics.ins-det]].