

Description of the Fermilab software school data objects

Matthew Herndon¹

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 layerStripMaps 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:

- getStrips(): 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.
- getLayerStripMap(int layer): const reference access to a the strip data from a single layer. The method explicitly indicates that a map is retrieved since maps have access methods methods that are not common to all containers.
- insertStrip(unsigned int layer, int stripNumber, int acd): 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 test 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 and adc: strip number 11 bytes, max 2048, and adc 5 bytes, max 32. (repeated number of strips times)

Note event numbers and 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 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 HIT AND HITSET

A. Hit class

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

The `Hit` class contains:

- `TVector3 _hitPosition`
- `int _layer`
- `int _numberStrips`
- `Int _trackNumber` (only for generator level hits)

The class servers for both generator Hits and reconstructed Hits. Associated information such as Hit resolutions are stored in the `DetectorGeometry`. Also note that a hit can be 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,trackNumber)`
- `Hit(hitPosition,layer,numberStrips`

Access methods:

- `getHitPosition():...` names correspond to the data members
- `print(ostream&):` To print to an a user specified ostream.

B. HitSet class

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

The HitSet is a typedef hitSet of type std::vector of Hits.

Access methods:

- getHits(): 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..

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 identify whether a hit position at a sensor plane is within the active area of the sensor.

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: 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
- - track number, -1 if none

IV. 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

- int `_dr`, point of closest approach to the reference point, 0,0,0

Constructor:

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

Access:

- `getLorentzVector():...` names correspond to the data members
- `getPosition(): _dr`
- `makeHelix(curvatureC)`: function which takes the curvature constant in the magnetic field 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 helix parameters would be different at different points.
- `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 `genTrackSet` of type `std::vector` of `GenTracks`.

Access:

- `getGenTracks()`: 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 is 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`: 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`

V. 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
- _trackHitSet, a typedef 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 on a number of hits.

Access:

- getHelix():... names correspond to the data members
The getHelix() function returns a Helix object which contains the basic track parameters.
 - Helix accessor functions called as getHelix().getDr() ...:
 - getDr(): Impact parameter, distance in the x-y plane 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.
 - getPhi0(): Phi angle from the reference point to the point of closest approach.
 - getKappa(): inverse Pt signed to indicate handedness + right handed (negative charge) - left handed (positive charge)
 - getDz(): z distance to reference point at point of closest approach
 - getTanL(): pZ/pT or inverse cot(theta) where theta is the dip angle which is zero for track perpendicular to the z axis and 90 degrees for parallel.
- getLorentzVecor() get a root TLorentzVector object based on the track parameters.
- print(ostream&): to print to an a user specified ostream.

The Track and Helix objects are described in more detail in the header files Track.hh and Helix.hh in DataObjects directories.

B. TrackSet class

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

The TrackSet is a typedef trackSet of type std::vector of GenTracks.

Access:

- getTracks(): 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 is 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: to identify the set
- version: The version is checked on input to make sure the correct streamer code is being used.
- number of GenTracks
- - Track number (repeated number of GenTrackNumber 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] track.pdf note, M. Herndon (2014)