Development of the Micromegas TPC Analysis Code

Peter Hayman

November 28, 2011

Outline

- Background
  - **Brief code history**
  - **Old code structure**
  - **Old code limitations**
- Recent Improvements
  - **Current code structure**
  - **Current code limitations**
- Plans for the Near Future
  - **MarlinTPC**
  - **Micromegas Simulation**

Background – History

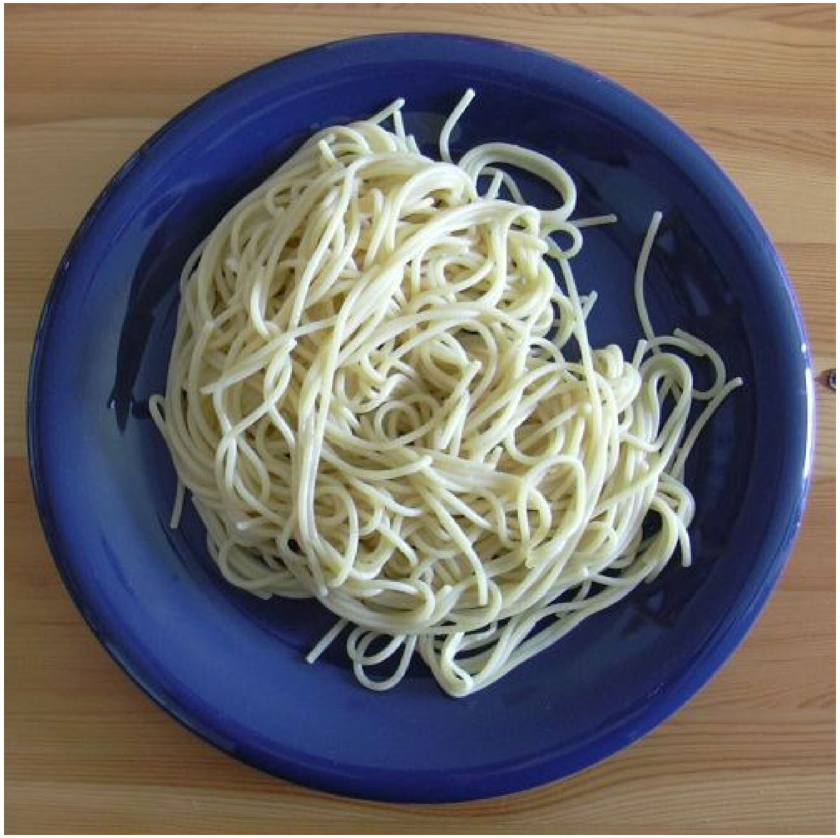
- Our analysis code was developed (mostly) independently at Carleton
- It began life as a FORTRAN95 program, but was eventually machine-code translated into (technically) C++
- This code was modified many times over the years, and was used to develop the analysis process
- It was mostly successful for its purpose
- At the beginning of the summer, the code consisted of several unique programs...

Background – Structure

- **NativeToLCIO**
  - Converts data from the native file format of the detector hardware to the LCIO standard
- **Main Code**
  - DD: creates dense data files from LCIO
  - PRF: determines track fits based on pad response function (prf) supplied by user
  - BIAS: calculates and saves values used for bias and reso ROOT scripts
- **ROOT Scripts:**
  - PRF: used to determine goodness of fit of the prf with chosen parameters
  - BIAS: calculates and corrects for signal bias inherent to the detector
  - RESO: calculates the resolution

Background – Structure

- The actual code layout, however, looked more like...



Background – Limitations

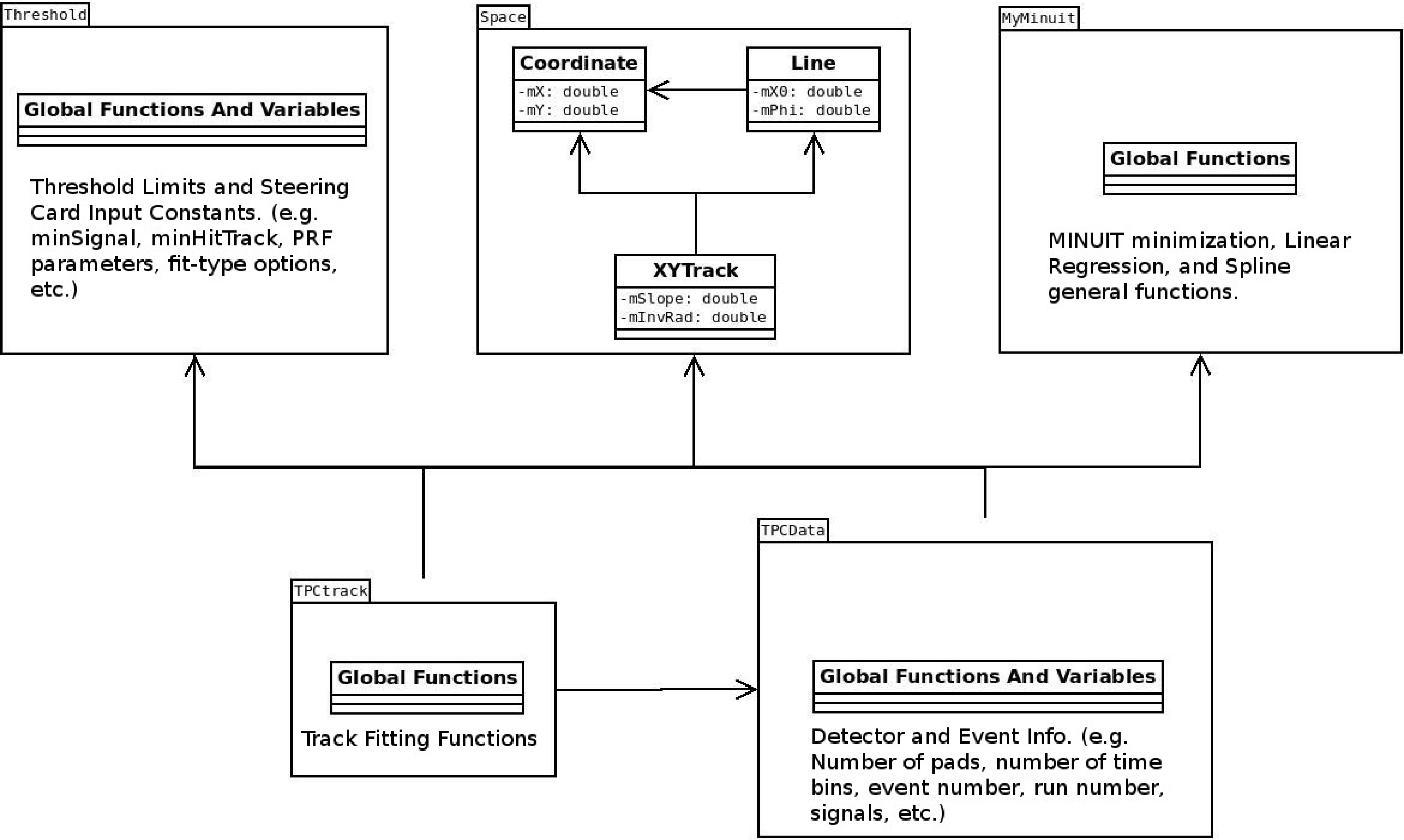
- Besides having the structure of a bowl of spaghetti, the code was:
  - **Slow**
  - **Extremely difficult to modify and understand**
  - **Very incompatible with concurrent international development**

Improvements – Structure

- The structure of the operation of the code is mostly unchanged, with only small modifications to improve ease-of-use (ex. command-line arguments)
- Underlying source code was vastly changed – specifically the main code.
- Proper programming practice was implemented (so it can now really be called C++)
- Biggest memory leaks have been plugged

Improvements – Structure

Globals:



Threshold

Global Functions And Variables

Threshold Limits and Steering Card Input Constants. (e.g. minSignal, minHitTrack, PRF parameters, fit-type options, etc.)

Space

Coordinate

-mX: double
-mY: double

Line

-mX0: double
-mPhi: double

XYTrack

-mSlope: double
-mInvRad: double

MyMinuit

Global Functions

MINUIT minimization, Linear Regression, and Spline general functions.

TPCtrack

Global Functions

Track Fitting Functions

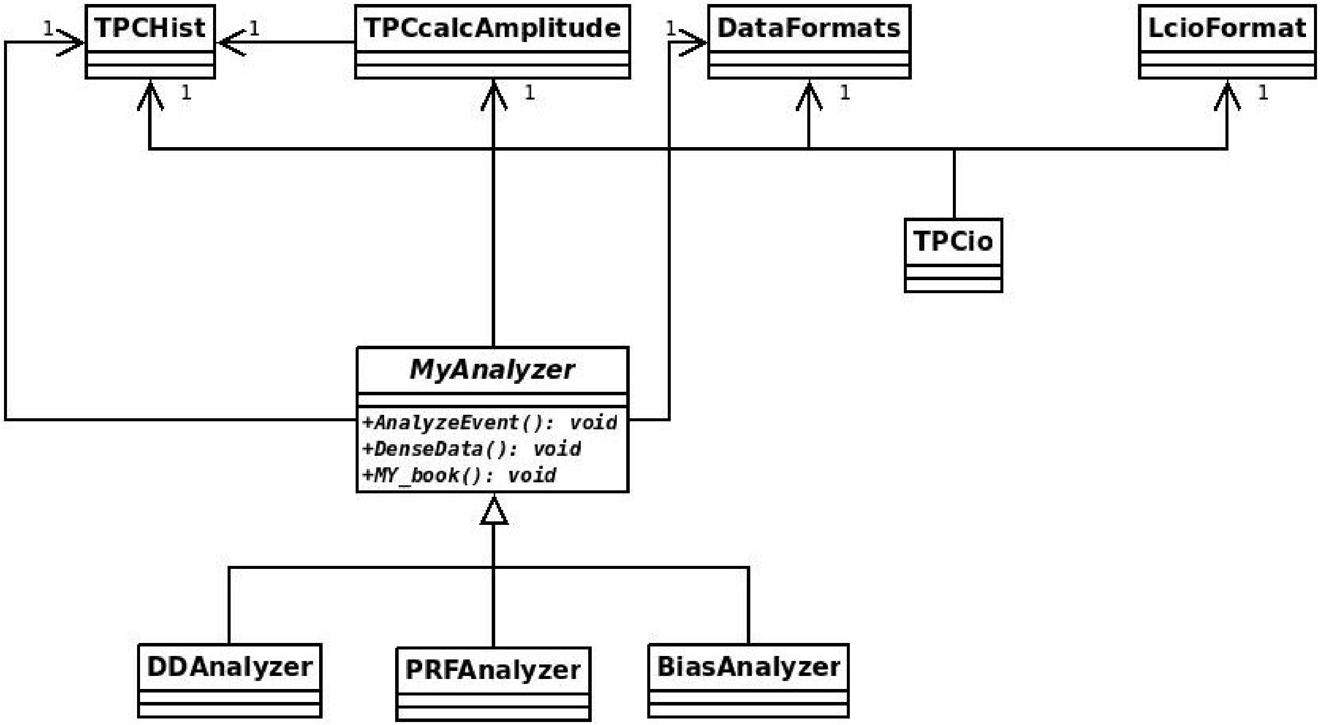
TPCData

Global Functions And Variables

Detector and Event Info. (e.g. Number of pads, number of time bins, event number, run number, signals, etc.)

Improvements – Structure

Classes:



Improvements – Limitations

- The code is now much more readable and far simpler to actually develop. However,
  - **It is still fairly slow**
  - **The ROOT scripts are virtually untouched and still very detached from the main code (which, while undesirable for a unified analysis code package, is potentially useful later on)**
  - **Lack of communication between the scripts and the code prevents implementation of some potentially very useful improvements (e.g., reducing overflow rejections)**
  - **Still completely detached from the international effort**

Future

- What's the next step?
  - **Could continue structural improvements and unification of different processes into one large analysis package, but...**
  - **Could also kill several small flying creatures with one stone and integrate our code with MarlinTPC.**

Future – MarlinTPC

- MarlinTPC is the global effort to develop a single analysis code package for all the different prototype TPCs being developed.
- It is far from complete, but it has a solid foundation, and the MPGD TPC is the last prototype that is unrepresented
- Furthermore, now seems to be the optimal time to integrate our code, as demonstrated by the number of small flying animals this one stone will hit...

Future – MarlinTPC

- Small Flying Animal #1:
  - **Modularity of ROOT scripts and code processes is ideally suited to the Marlin design, while Marlin contributes a unified analysis package**
- Small Flying Animal #2:
  - **Many basic elements of the procedure are already implemented in Marlin, such as detector layout (GEAR), constants (LCCD), and global containers (LCIO)**
- Small Flying Animal #3:
  - **The similar pad-based detectors already implemented in Marlin seem to be at about the same stage we are with respect to track fitting, but are lacking our careful bias and residuals checks**

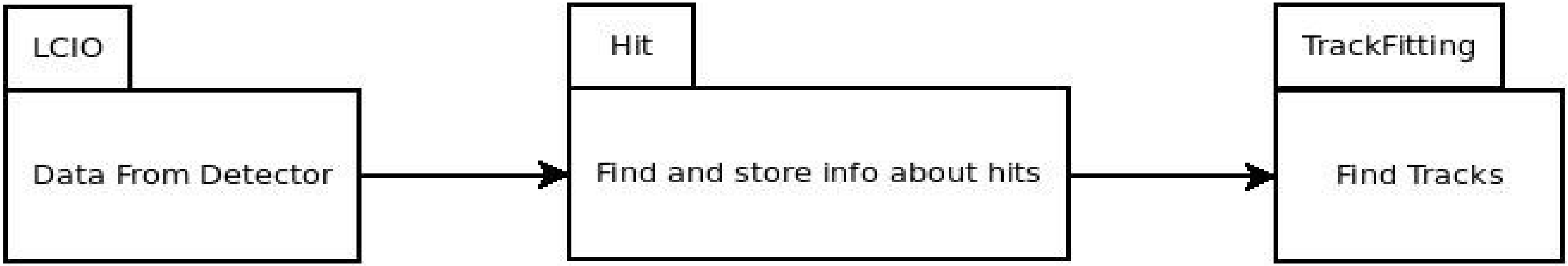
Future – MarlinTPC

- The consequences of each of these points, are as follows:
  - **Our pre-existing modularity means that the most difficulty and effort in integration will be due to learning the Marlin environment**
  - **The pre-existing Marlin tools means that improved overflow checks should be trivial to implement, and having access to a polar coordinate system means accurate representation of the detector, and an easier extension to curved tracks**
  - **The current stage of development on all sides means that our involvement will allow us to pool our resources and develop a strong trackfitting algorithm**

Future – Simulation

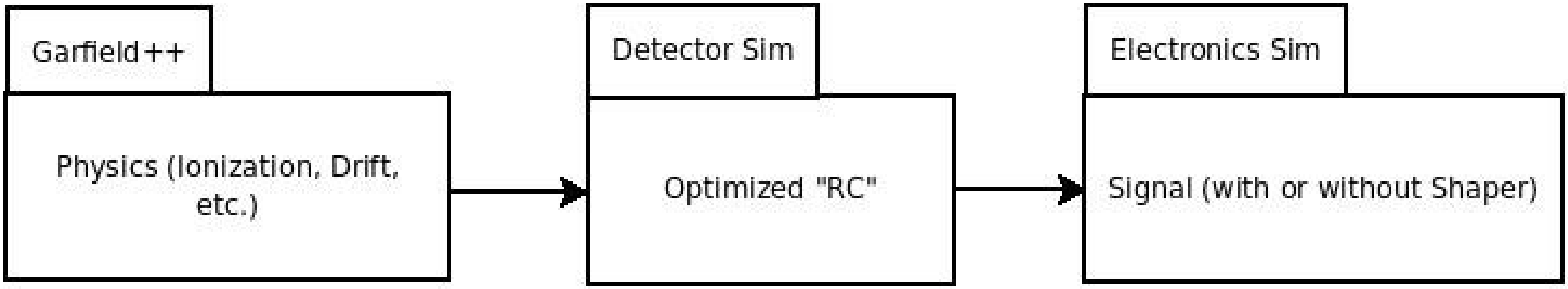
- Concurrently developing simulation of Micromegas detector

- The procedure for the analysis is, basically,



Simulation

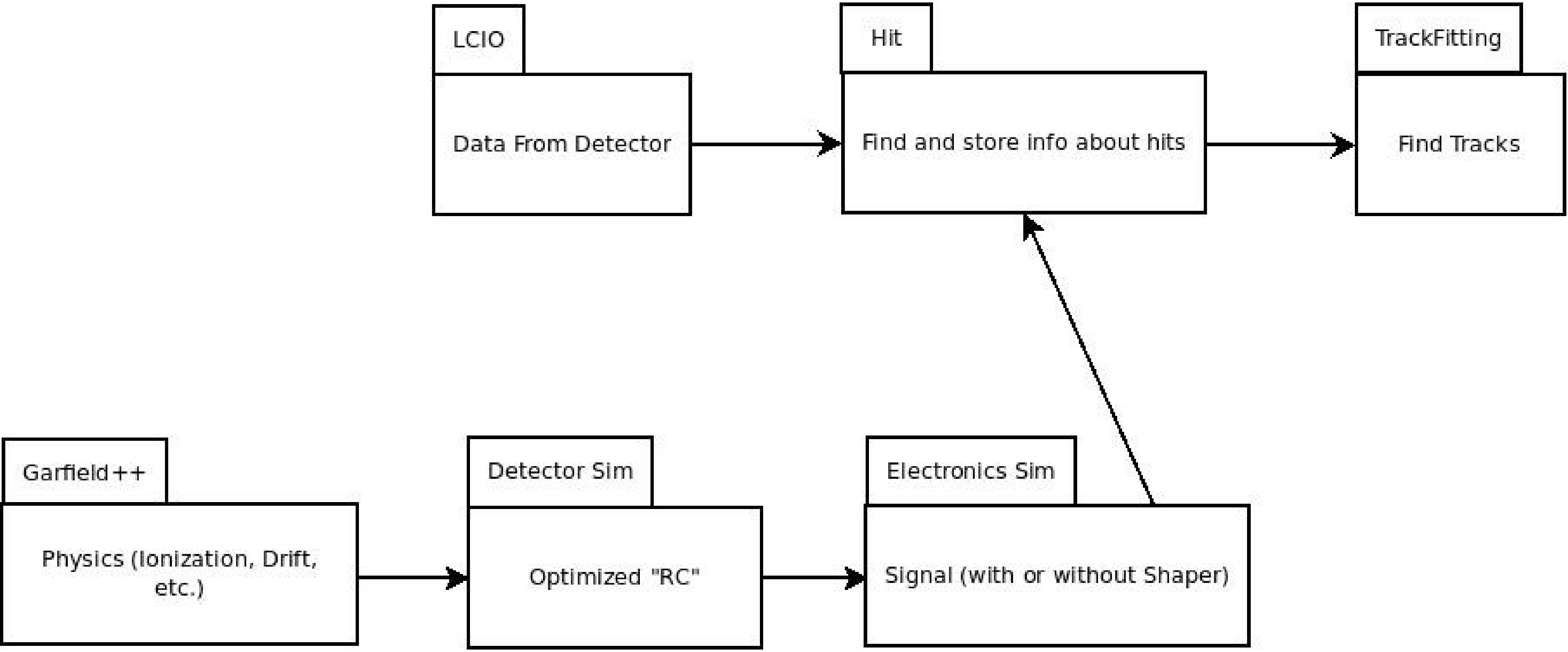
The simulation will perform the following calculations,


Garfield++
Physics (Ionization, Drift, etc.)
Detector Sim
Optimized "RC"
Electronics Sim
Signal (with or without Shaper)

Simulation

And this will fit in with the analysis work, by simply replacing the detector data with the simulated data in the analysis procedure.

Simulation



LCIO
Data From Detector
Hit
Find and store info about hits
TrackFitting
Find Tracks
Garfield++
Physics (Ionization, Drift, etc.)
Detector Sim
Optimized "RC"
Electronics Sim
Signal (with or without Shaper)

End