



Introduction to LArSoft

Brian Rebel
October, 2009

Outline



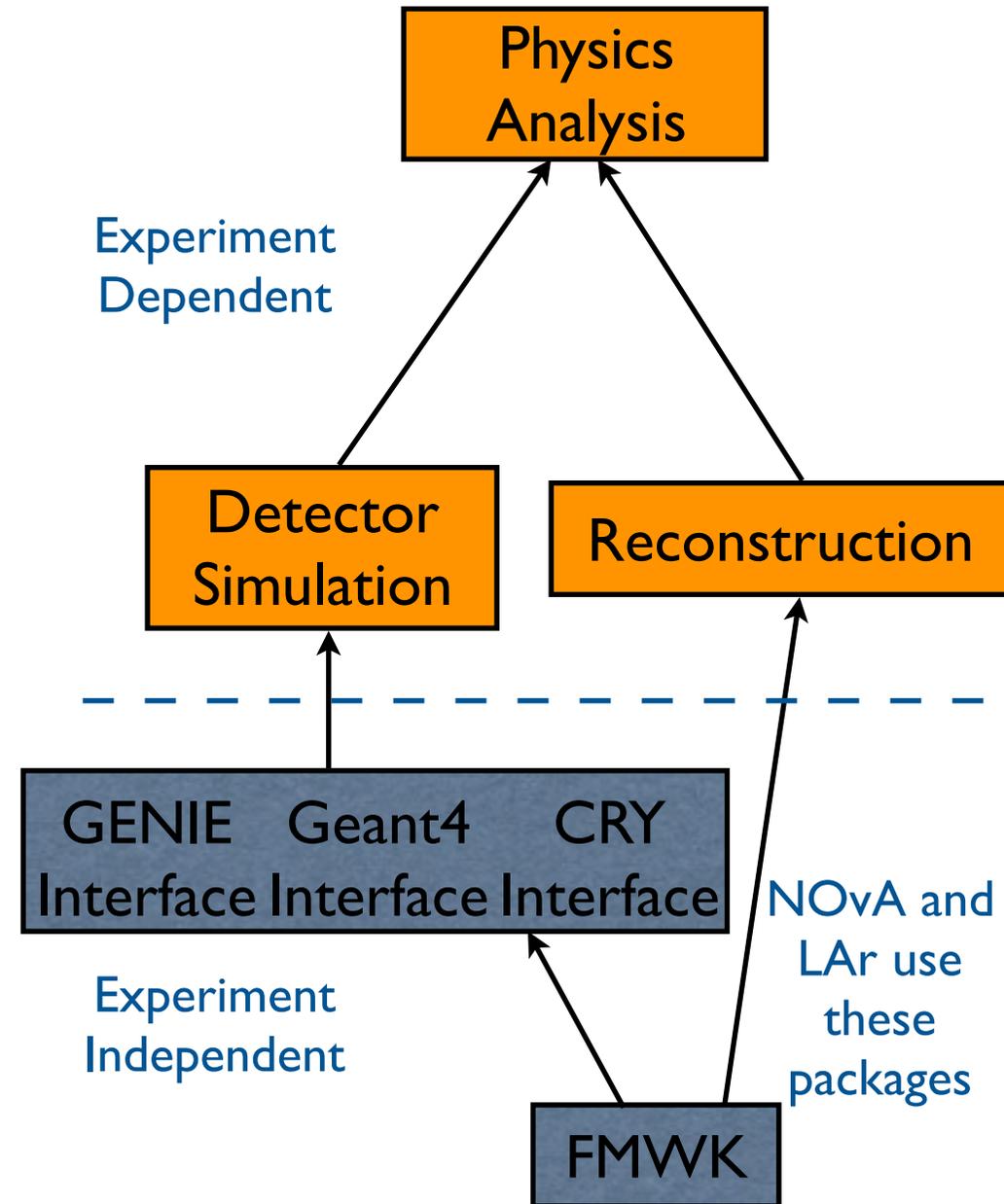
- Overview of NuSoft at FNAL
- Getting setup to use the code
- FMWK - the base of LArSoft
- LArSoft - the simulation and reconstruction code
- How to get involved



Neutrino Software @ FNAL



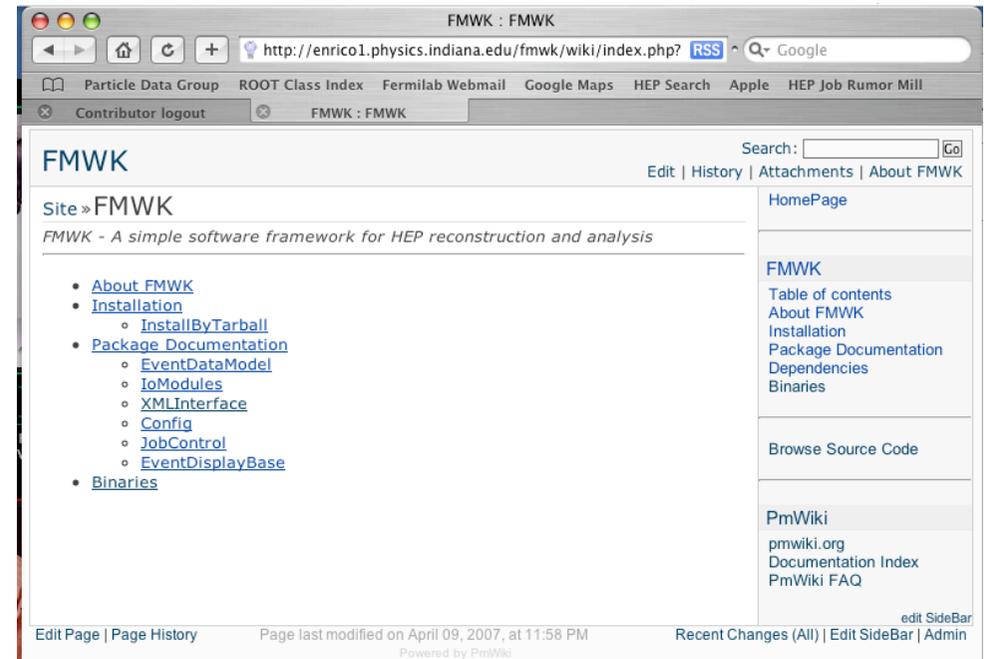
- Recent effort made to get FNAL neutrino experiments on common framework, using common tools
- NOvA, ArgoNeuT, μ BooNE using FMWK
- Use of common framework enables sharing of interfaces to flux files, GENIE (neutrino generator), Geant4, etc
- Enables people working on multiple experiments to only learn one framework



FMWK



- FMWK is a generic analysis software framework designed for particle physics experiments
- Underlying design driver is “What does a physicist need?” If there is a conflict between OO design principals and physics ease of use, side with physics
- Designed to be lightweight
- The home page for FMWK is <http://enrico1.physics.indiana.edu/fmwk/wiki/index.php?n=Site.FMWK>
- Makes use of ROOT
- Provides several basics packages
- Website has some documentation on a these classes



The FMWK packages

Most users will want to learn about the EventDataModel, Config, and JobControl packages. The IoModules, XMLInterface, and EventDisplayBase packages typically work behind the scenes.

"User interface" packages

- [EventDataModel](#) provides an interface to the event data store and allows storage of user-defined ROOT objects.
- [Config](#) provides a uniform interface to run-time configuration data
- [JobControl](#) defines the basic units of user reconstruction and analysis code and provides tools for assembling those modules into jobs

"Expert level" packages

- [IoModules](#) provides the interface between the EventDataModel and I/O streams such as files
- [XMLInterface](#) provides an interface between the xerces-c XML parser and user code that uses XML to create and manipulate C++ objects
- [EventDisplayBase](#) provides GUI interfaces to the Config and JobControl packages and contains boilerplate code for generating menus and detector views
- [SRT_FMWK²](#) SRT build configuration files



Main » Philosophy

The philosophy behind FMWK is probably best summarized in the book "Getting Real" which you can read online by following [this link](#). FMWK emerged as a reaction I had to other software frameworks for HEP experiments I've seen that I felt made interaction with the experiment's data a chore for the typical physicist. These frameworks often try to solve problems that the typical physicist has no need for and in the process makes what should be simple transactions complicated. Here are some of my goals for FMWK:

1. **Simple things should be simple, hard things should be possible.** If there is a conflict between between these two, err on the side of simple. A good framework should make the "every day" work as simple as possible. If a user is sophisticated enough to pose a complicated problem, they are probably sophisticated enough that they don't need a framework.
2. **Make choices.** Many frameworks I've seen get complicated because they try to support every style of working. In FMWK, I've tried to provide exactly one good way to do things. Sometimes this means that when a user asks "Can I do such and such?" and the answer is "No, you can't". I think that's OK in the interest of preserving simplicity. Simple questions end up having straight forward answers and user's are not presented with a bewildering array of configuration choices to make. Once they are comfortable with the framework, they don't spend time wondering "Should I structure the code this way or that way?"; the framework provides one answer to this question and the developer can spend more time thinking about the reconstruction or analysis task they have in mind.
3. **Think about what people want, not what they "should" want** If a framework makes choices that satisfies the user's needs and expectations, they will work within the framework. Otherwise they will work around the framework resulting in a set of code for an experiment that does not work well together.
4. **Keep it real.** I built FMWK for a running experiment and I only added features as they were required. I spent almost zero time thinking about problems that were not right in front of me. The result is a small, compact code base that is (I hope) easy to update and iterate on as new problems arise. I think this is a problem with many HEP frameworks designed without reference to specific reconstruction and analysis tasks: they invest large amounts of time, thought, effort, and code on problems that 99% of users will never encounter and result in a code base that puts abstract software design goals ahead of the scientific goals of the experiment.

FMWK - Details



- All basic jobs are completed in FMWK
 - EVD handles keeping track of data, reconstruction objects, MC, etc.
 - Config allows for run time adjustment of parameters without recompiling
 - JobControl allows one to build up simple or complex jobs
- These packages are the only ones that typical users will ever need to do reconstruction, etc
- Expert level classes handle
 - I/O of ROOT files
 - XML interface for configuration files
 - GUI interfacing for event displays
- Requires 2 external packages
 - ROOT (Cern package for analysis/display)
 - Xerxes-C (XML interpreter)

FMVVK - Lingo



- There is a certain level of lingo associated with any framework
- Look through the FMVVK wiki for more details
- Executable is called ana

Jobs are constructed from several elements:

nodes

are the smallest element which perform a single, well-defined reconstruction task. For example, nodes might be responsible for track finding, track fitting, applying pulse height corrections, etc. etc.

sequences

are collections of related nodes. Use of sequences is optional but they can make the construction of large jobs more convenient. For example a user might want to "run the track fitting" prior to running their own piece of analysis. That single concept "the track fitting" may require several nodes ordered and configured in a very specific way. Rather than make the user assemble this collection, its better if an expert assembles these nodes once in a sequence called, say, "Tracking". To the non-expert user, this sequence is a simple way to guarantee that the tracking they are running in their job is correct and standard.

jobs

are collection of nodes and sequences. The ana executable will run all the jobs declared to it.

These relationships are expressed in input files using XML.

FMWK - Lingo



- EventDataModel keeps track of information for each event
- Each section listed below is a TFolder, this is not an ntuple based storage
- Provides a bit more flexibility than ntuples
- Make subfolders for different algorithms, ie tracking folders in Reco

The event data model divides the event up into several sections.

- Header: The basic event header
- DAQ: Additional information from the DAQ. For example, trigger words, status codes, ...
- Raw: The uncalibrated raw data directly from the DAQ
- RawAUX: An auxiliary raw data branch. Useful for storing data from a single detector sub system which dominates the data size. Other detector sub-systems can be analyzed without having to load this branch, improving the I/O speed of those analyses.
- Cal: Calibrated detector data. These objects would typically be the raw data converted into physics units (charge in pe's, times in ns, etc.). Reconstruction would typically use these objects and not require loading of the Raw data branch.
- Reco: This would contain reconstruction objects (tracks, vertices, etc.)
- Summary: "DST"-style summary objects. Nice when scanning for interesting events to have this summary information in its own small branch to allow for rapid searches for interesting events.
- User: A scratch area for any user data.
- MC: The Monte Carlo truth information from the generator
- DetSim: The Monte Carlo information (tracks, his, etc.) from the detector simulation.

FMVWK - Lingo



Creating a configuration

An example is most likely the simplest way to understand how the configuration package works. The following XML defines a configuration parameter set "AConfig" who's version is "default":

```
<configdoc>
  <config name="AConfig" version="default">

    <param name="AnInt"> <int> 1 </int>
      Shows how to create and integer parameter
    </param>

    <param name="Afloat"> <float> 3.1415 </float>
      Shows how to create a floating point parameter
    </param>

    <param name="SomeFloats"> <float> 1.0 2.0 3.0 </float>
      Shows how to create a list of floating point numbers
    </param>

    <param name="Astring"> <string> TestString </string>
      Shows how to create a single string
    </param>

    <param name="SomeStrings"> <string> 'A' 'list' 'of' 'strings' </string>
      Shows how to create a list of strings
    </param>

  </configdoc>
```

- Configurations are derivable to change just a few options of defaults, etc

nutools - Details



- Nutools is a set of packages designed to abstract commonly needed software among different neutrino experiments
- Part of the nusoft repository
- Currently used by NOvA and LArSoft
- 3 Packages
 - EventGeneratorBase - contains interfaces to GENIE (neutrino interaction) and CRY (cosmic rays) generators; also has xml files for setting parameters for each
 - SimulationBase - contains data objects to hold basic particle information about each generated event/spill including StdHep information and flux information
 - Header - basic metadata storage object; stores information about file generation environment, random numbers, detector used to produce file, etc



Getting Setup for LArSoft



- The wiki is your friend to setup the code - <http://www.nevis.columbia.edu/twiki/bin/viewauth/LArSoft/WebHome>
 - Username: LArSoft
 - Password: argon!
- Home page shows you how to get a FNAL computing account
- Home page also details how to ensure you can access the cvs repository
- Instructions for how to setup the code are under the Using LArSoft links
- Description of packages are also there
- Basically, you should look on the wiki first for all questions
- If it ain't there, then please add it

LArSoft development home page

The LArSoft software is designed to work for all planned and running liquid argon experiments at Fermilab. It is written in C++ and built on the [ROOT](#) data analysis software and the [EMWK](#) framework for HEP experiments. The releases of the software are managed using an [SRT](#) distribution.

LArSoft documentation

- [Accessing LArSoft](#)
- [Using LArSoft](#)
 - [Fermilab Installation](#)
 - [Nevis Installation](#)
 - [Installing a Local Copy](#)
 - [Editing Code](#)
 - [Adding A Package to LArSoft](#)
 - [Running Jobs](#)
 - [Interactive ROOT sessions](#)
- [Package Documentation](#)
- [Getting started with an analysis task](#)
- [Available files](#)
 - [Data](#)

Getting Access to FNAL Computing

In order to work on LArSoft development you will need to have access to the FNAL computing resources. The procedure for getting a FNAL account is described at

http://computing.fnal.gov/xms/Services/Getting_Started/introduction_to_Computing_at_Fermilab

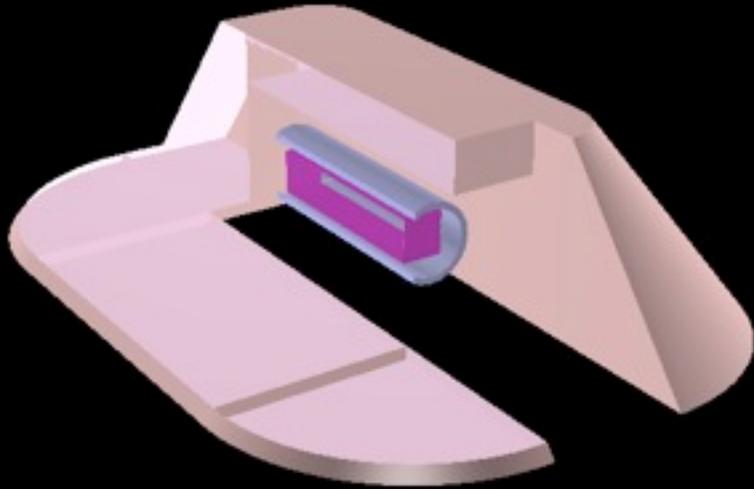
If you will be in residence at the lab, follow the instructions for an **on-site visitor**. If you will be based at your home institution, follow the instructions for an **off-site visitor**.



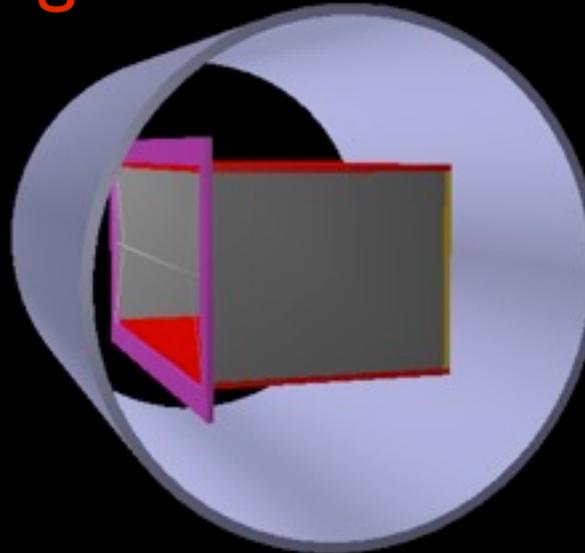
LArSoft

- FMWK does not provide classes to do reconstruction or analysis - experiments write their own
- LArSoft is an SRT based distribution of code for the LAr experiments - MTS, ArgoNeuT, μ BooNE, LBNE
- Each detector just needs to add a new geometry description
- Reconstruction knows how to access different geometries, but not dependent on any one

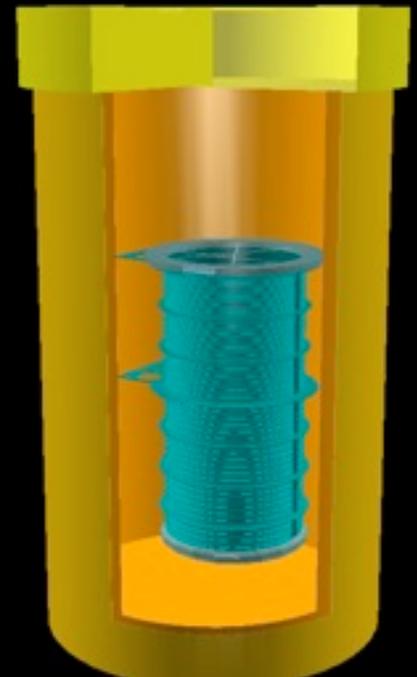
μ BooNE



ArgoNeuT



Bo





Getting Setup @ FNAL



- A setup script is provided to set all relevant environmental variables like PATH, LD_LIBRARY_PATH, ROOTSYS, etc
- Code currently lives in /afs/fnal.gov/files/data/argoneut/d01/lar/larsoft/
- source the setup/setup_larsoft_fnal.(c)sh in that directory
- Will print a lot of output to screen telling you what variables have been set for Geant4

```
Terminal — ssh — 84x50

<flxi09.fnal.gov> source $LAR/d01/lar/larsoft/setup/setup_larsoft_fnal.csh
On this machine the G4SYSTEM=Linux-g++
On this machine the G4INSTALL=/afs/fnal.gov/files/data/argoneut/d01/lar/external/geant4/geant4.9.2
On this machine the G4INCLUDE=/afs/fnal.gov/files/data/argoneut/d01/lar/external/geant4/geant4.9.2/include
On this machine the G4LIB=/afs/fnal.gov/files/data/argoneut/d01/lar/external/geant4/geant4.9.2/lib
On this machine the G4LEVELGAMMADATA=/afs/fnal.gov/files/data/argoneut/d01/lar/external/geant4/geant4.9.2/data/PhotonEvaporation2.0
```

LArSoft - Details



- LArSoft is available from cvs
- Need FNAL kerberos principal to access it
- Can run it from FNAL using the flxi* machines
- Intensity Frontier cluster will be up in January and will migrate to those machines at that time
- Build system is currently SRT
- Has been built successfully on machines outside of FNAL

LArSoft - Details



AnalysisExample	DriftElectrons	Header	RawData	tap
ArgoSimpleReco	EventDataModel	include	RecoBase	Utilities
bin	EventDisplay	IoModules	results	xml
CalData	EventDisplayBase	JobControl	setup	XMLInterface
Config	EventGenerator	LArG4	Simulation	
CVSROOT	EventGeneratorBase	lib	SimulationBase	
DetSim	Geometry	man	SoftRelTools	
doc	GNUmakefile	MCCheckOut	SRT_LAR	

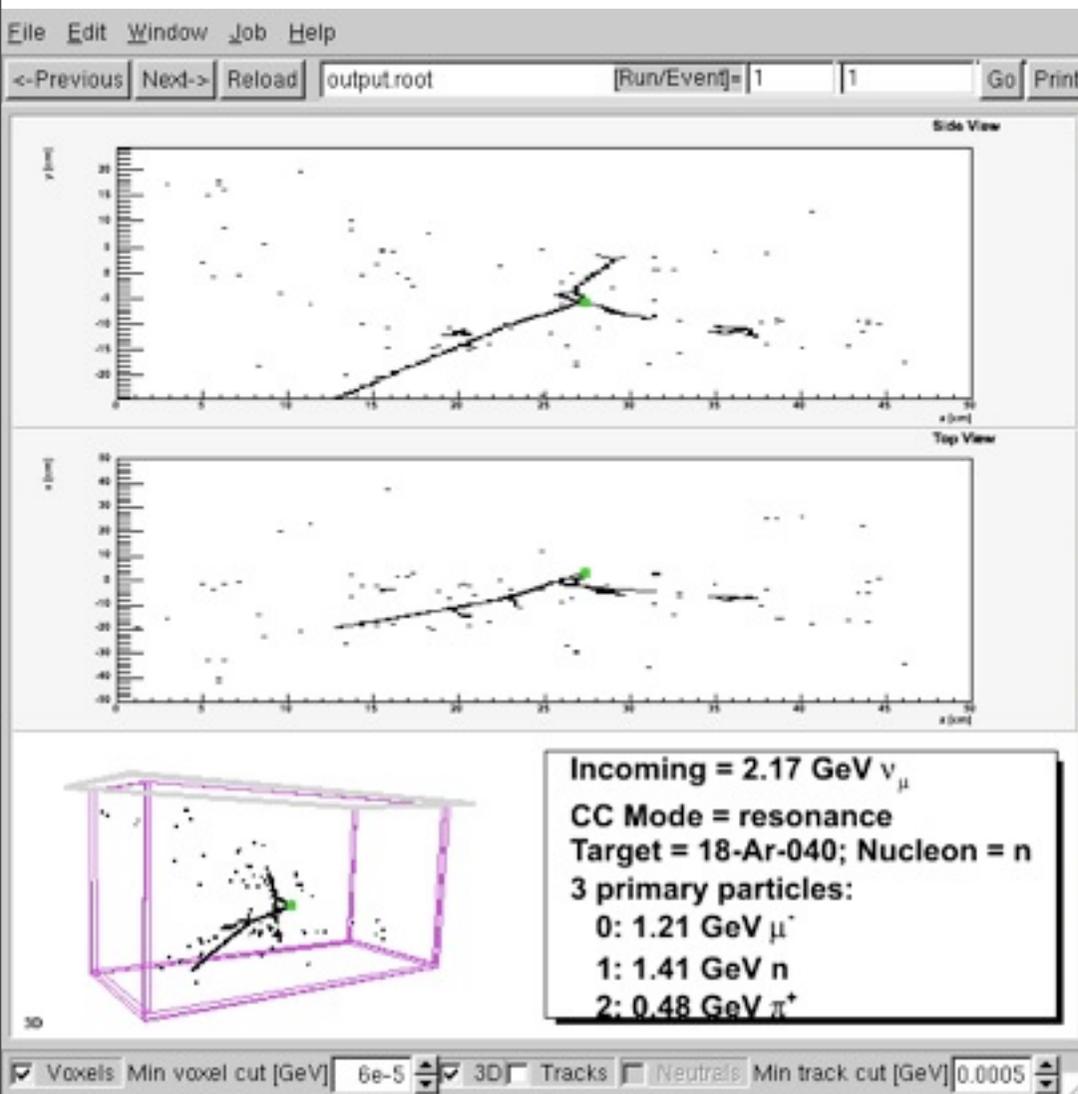
- Current package list above
- FMWK packages in
- Nutools packages in
- Remaining packages are all part of LArSoft
- Simulation related packages in
- Reconstruction packages in
- Clearly need reconstruction effort

Current State of Development

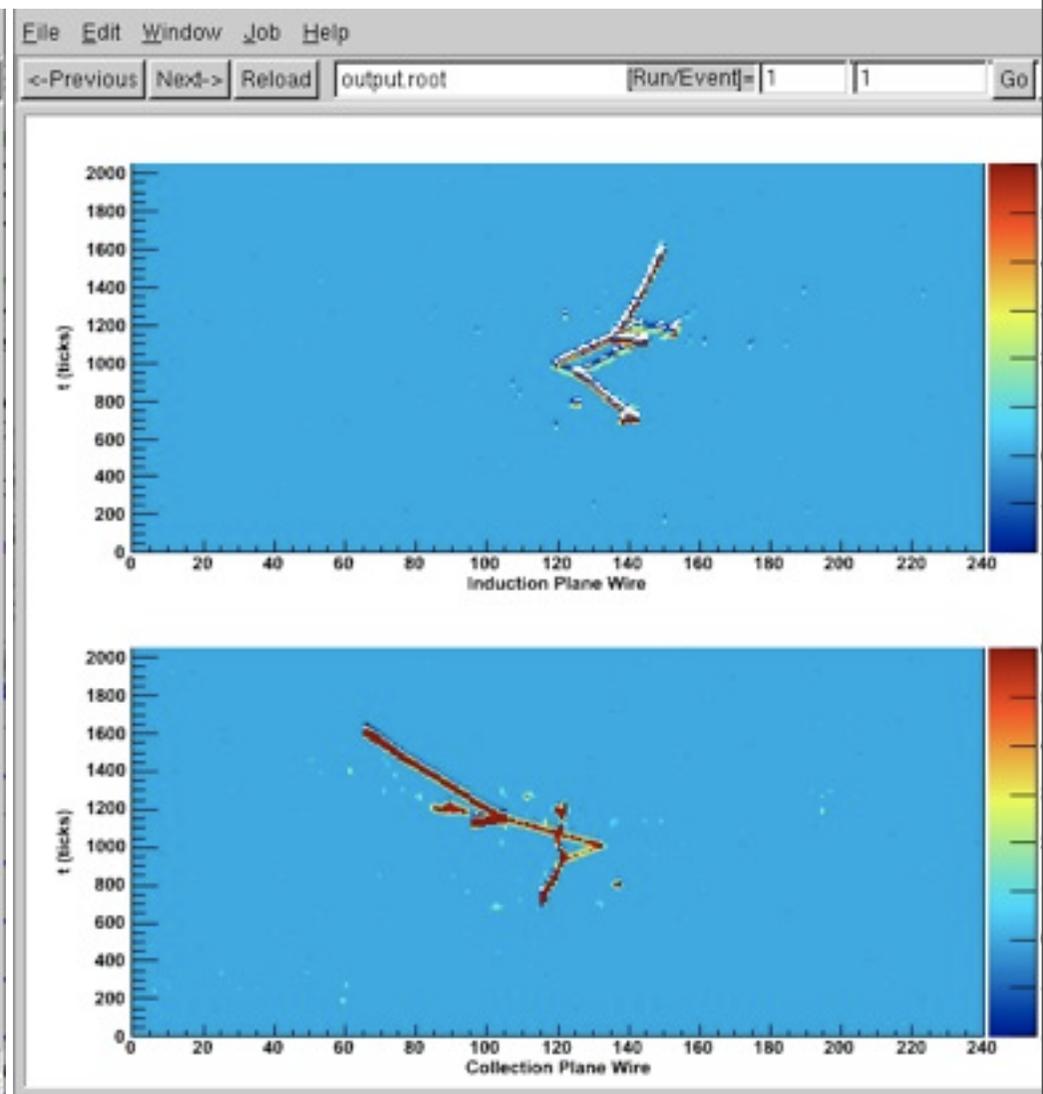


- LArSoft has benefited from needs of ArgoNeuT, μ BooNE
- Monte Carlo simulation is reasonably mature
 - Direct access of event generators for cosmic rays (CRY) and neutrino interactions (GENIE)
 - Direct access to Geant4 for energy deposition simulation
 - Conversion of energy to ionization electrons and readout simulation
- Hit finding routines are nearing completion
- Current need is for effort on reconstruction algorithms

Example MC Event

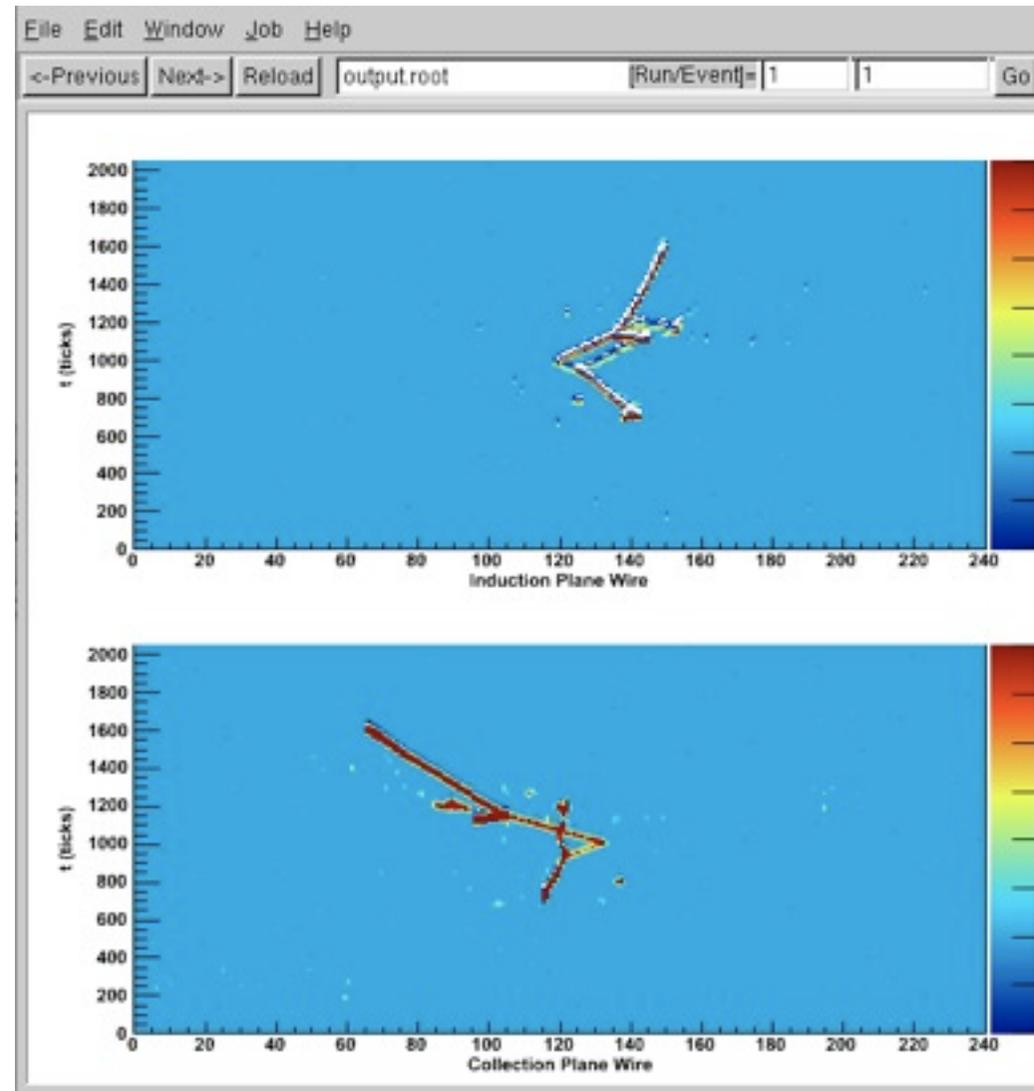


MC Energy
Deposition View



Simulated
Digitization View

Example MC Event



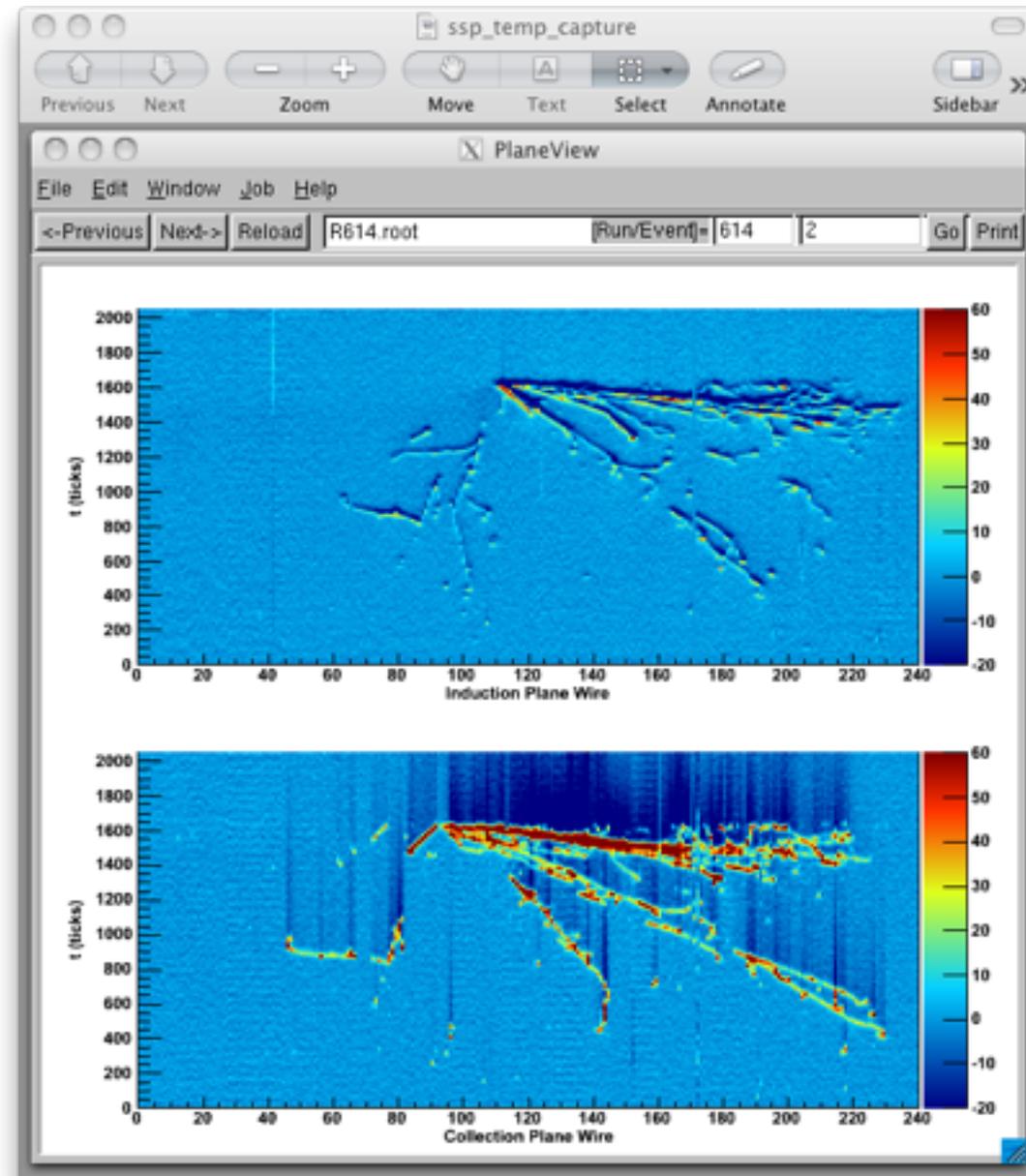
MC Energy
Deposition View

Simulated
Digitization View

Reconstruction



- ArgoNeuT event taken a week ago at right
- Shows that reconstruction will be a challenge
- Separation of different tracks, particle ID etc will be more complicated than in detectors with poorer resolution
- Field is wide open at this point for contributions



Current Contributors



- The list of LArSoft contributors is growing - several members from Columbia/Nevis, FNAL, MIT, Michigan State, and Yale
- Projects include
 - Improvements to detector geometry descriptions
 - Event display improvements
 - Simulation of light and photodetectors
 - Reconstruction of hits and tracks

Getting Involved



- Sign up for the LArSoft mailing list - send email to brebel@fnal.gov or gzeller@fnal.gov
- Participate in bi-weekly LArSoft video/phone meetings: Tuesdays @ 9 am CDT
- Contact Sam or me and suggest a project you would be interested in doing