TPC Simulation with LArSoft

Adam Lister
adam.lister@wisc.edu
University of Wisconsin — Madison

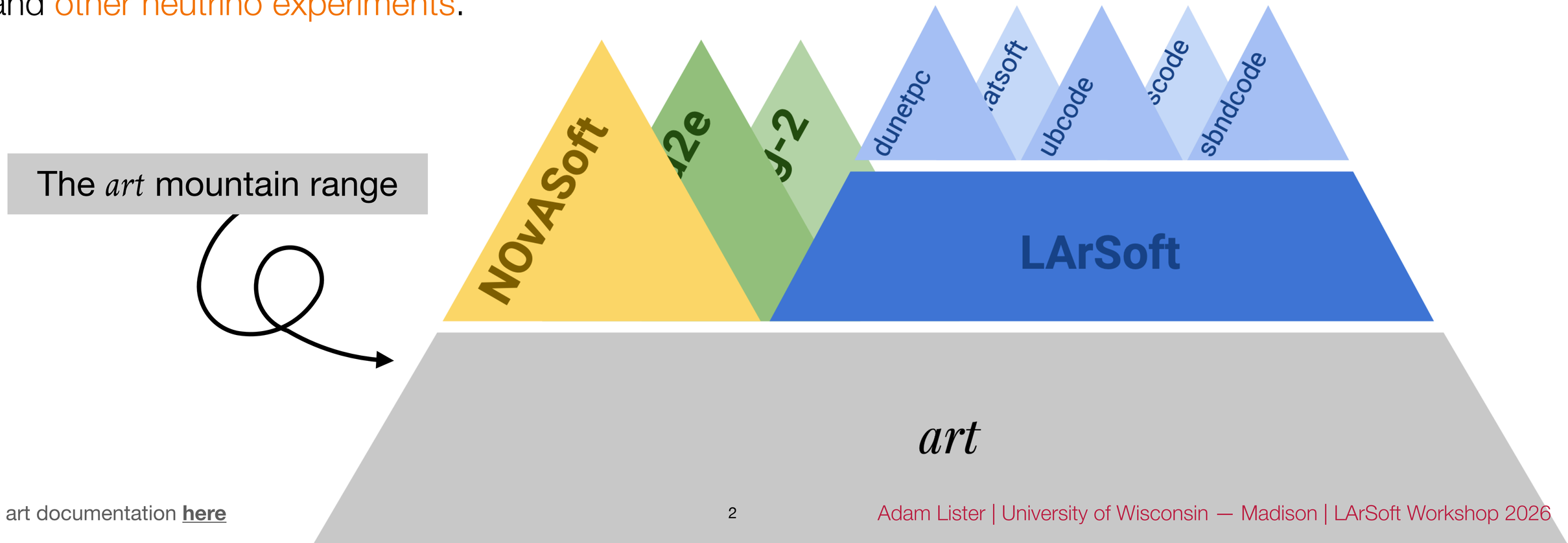


11th UK LArTPC Software and Analysis workshop | October 2026

What's a LArSoft?

LArSoft is a layer of modules, classes, and tools that are built on the *art* event processing framework, and are common to all of the liquid argon experiments.

art supports a lot of the Fermilab physics programme: muon experiments, liquid argon experiments, and other neutrino experiments.



See (old) art documentation **here**

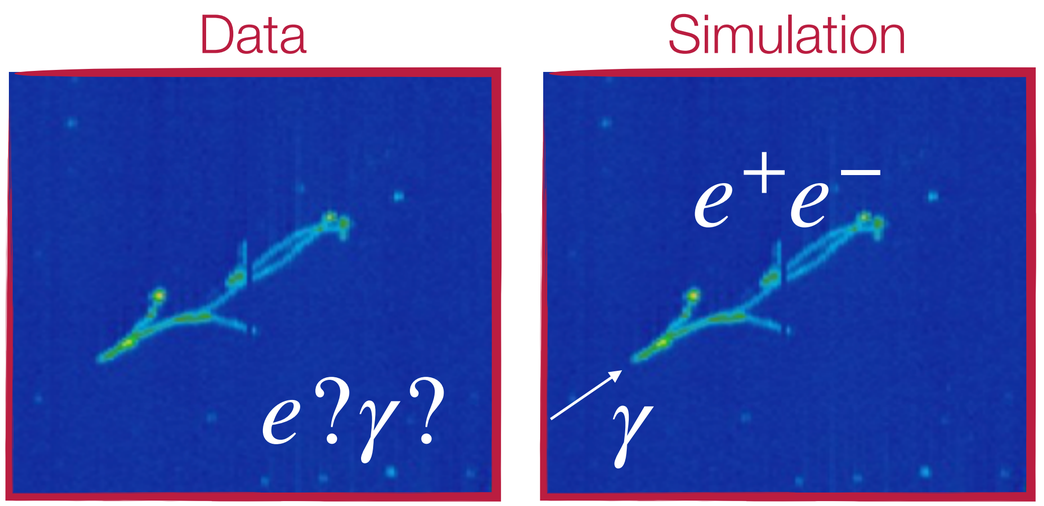
What's a LArSoft?

art/LArSoft can be used to apply reconstruction algorithms to our data, or to build a simulation of our experiments from scratch.

Why would we want to simulate our experiments?

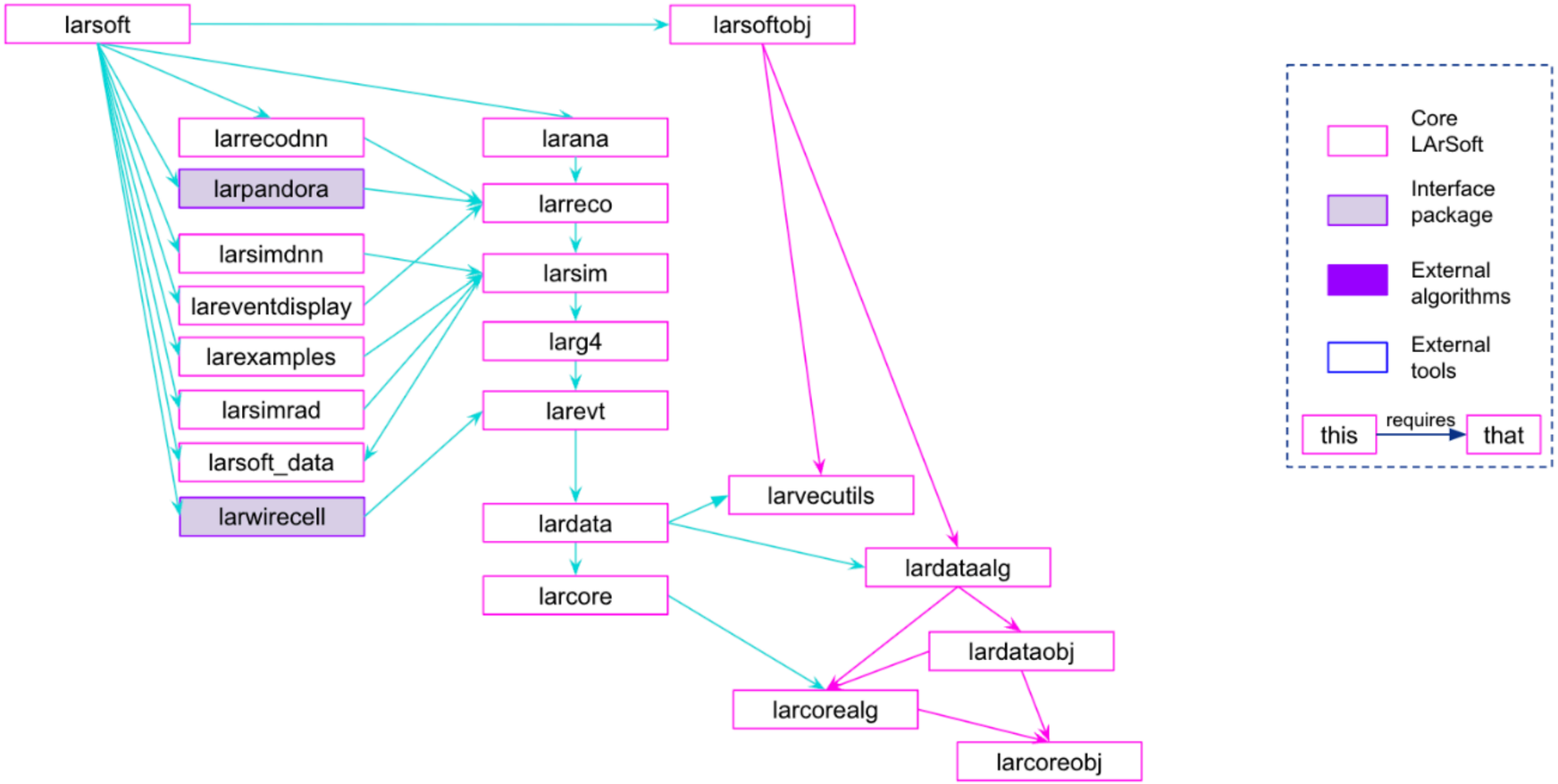
In simulation, we have access to truth information, so we know a given neutrino's flavour, energy, interaction point, and the particles it produced.

Fantastically useful for validating reconstruction and analyses are doing what we want!



What's a LArSoft?

LArSoft is modular, and is built up of many packages with a complex dependency chain…

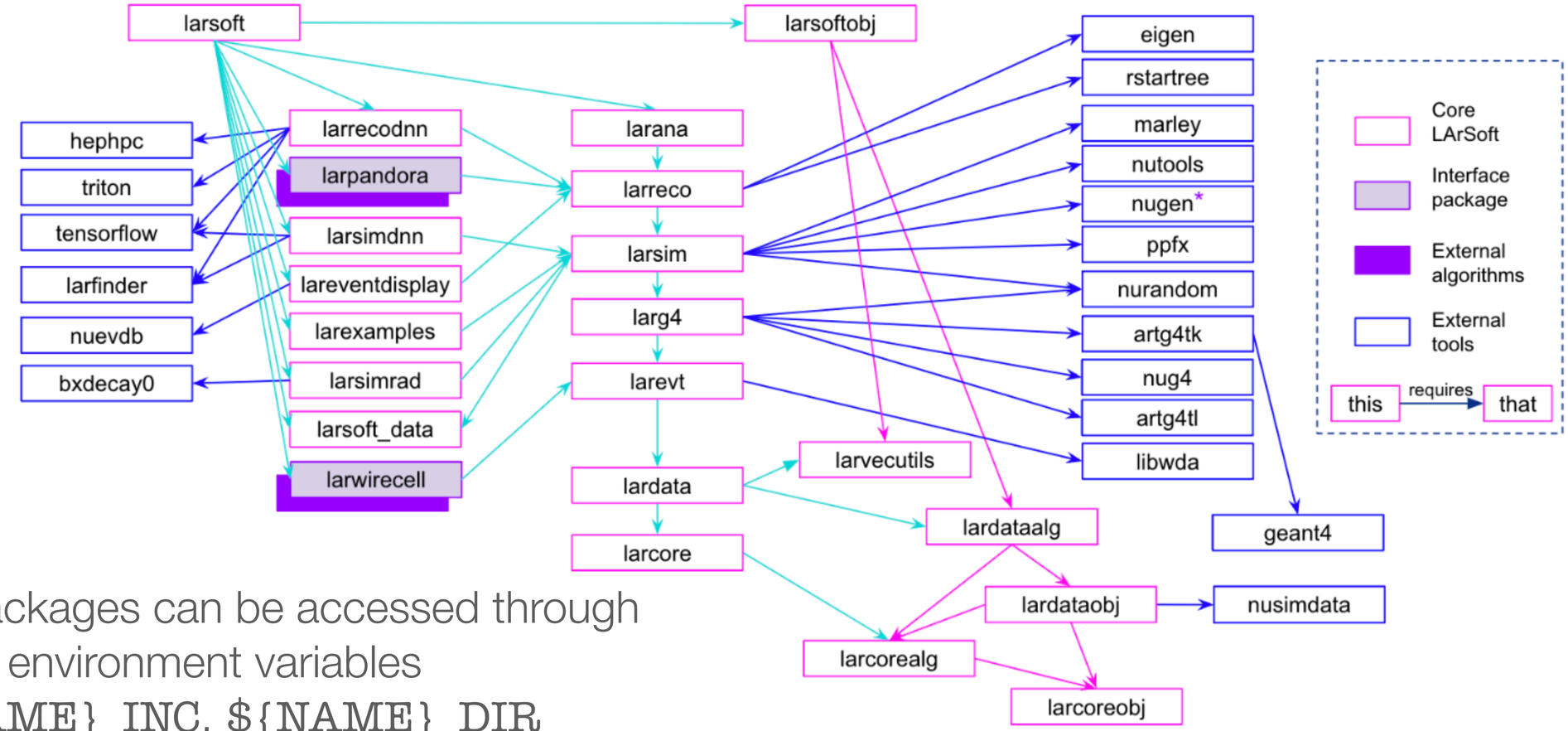


* from LArSoft website

What's a LArSoft?

LArSoft is modular, and is built up of many packages with a complex dependency chain…

which gets even worse when you take into accout the external products it depents on



different packages can be accessed through environment variables

`${NAME}_INC`, `${NAME}_DIR`

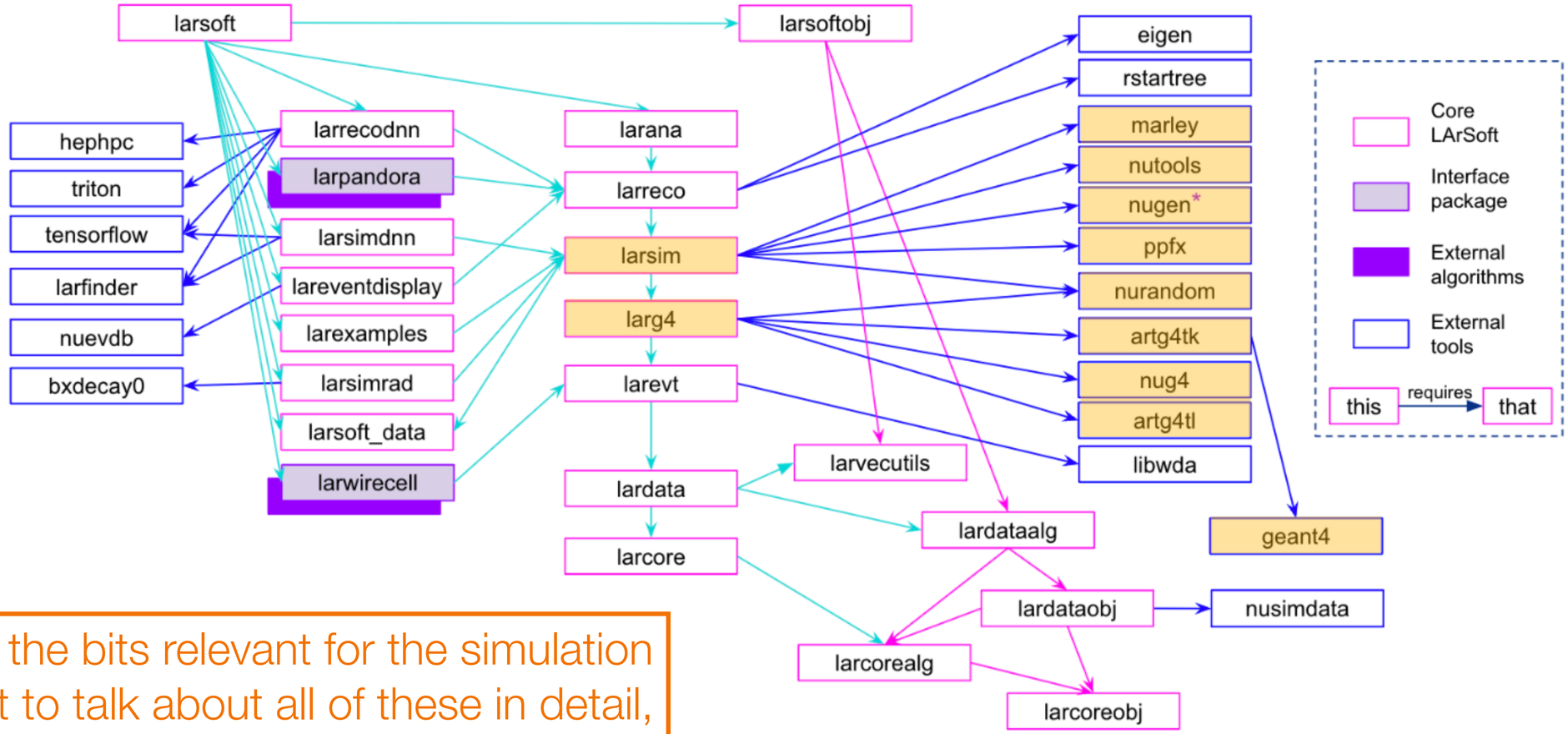
ie `$LARSIM_INC`, `$LARSIM_DIR`

* from LArSoft website

What's a LArSoft?

LArSoft is modular, and is built up of many packages with a complex dependency chain…

which gets even worse when you take into accout the external products it depents on

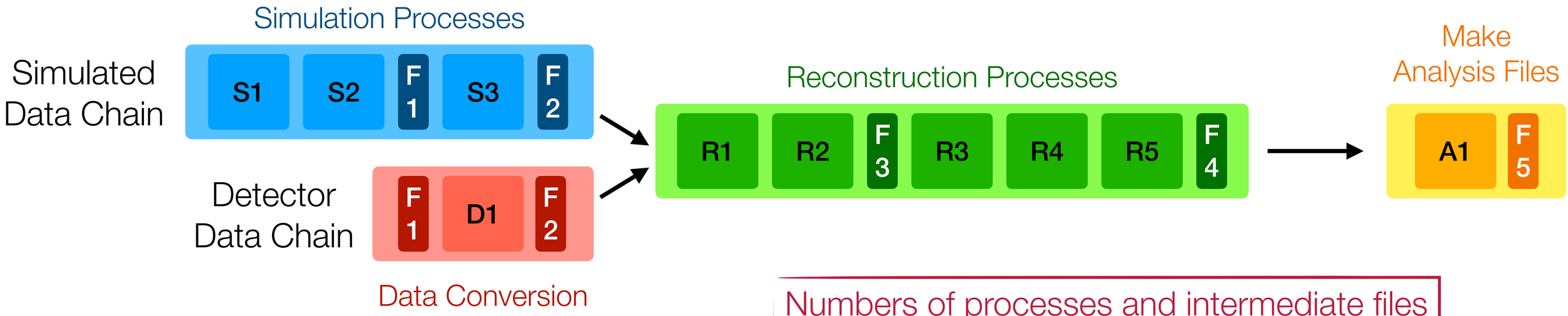


These are the bits relevant for the simulation
I won't get to talk about all of these in detail,
but some will look familiar by the end!

* from LArSoft website

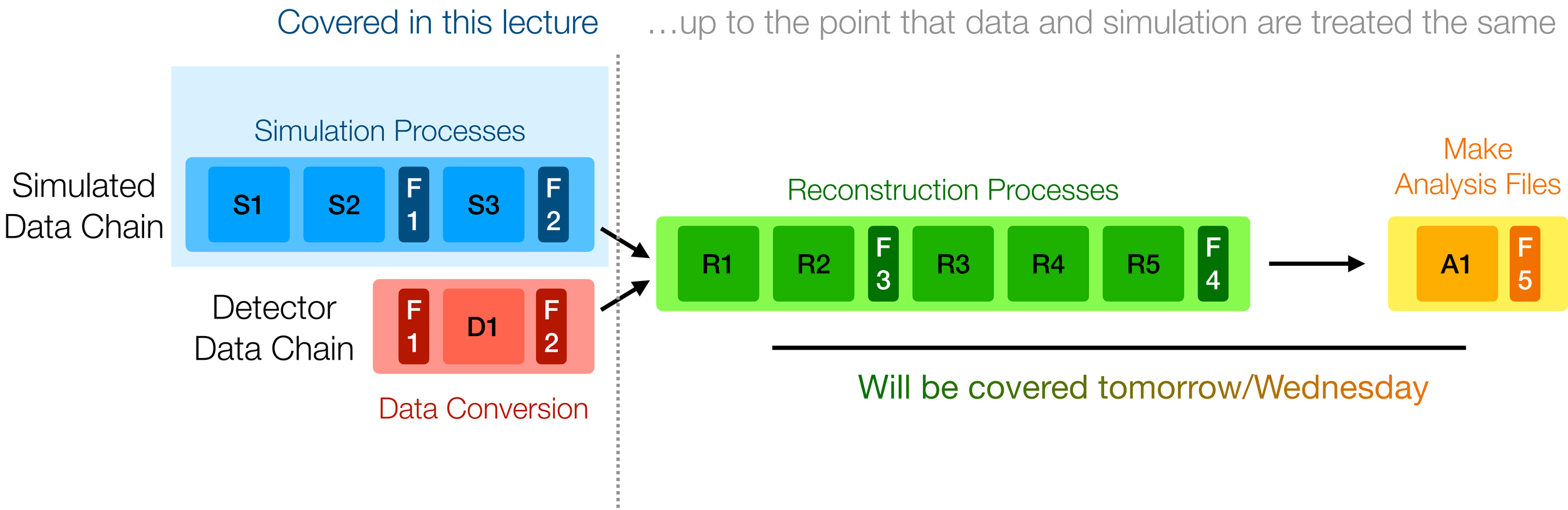
Anatomy of a LArSoft WorkFlow

Effectively, all art-based experimental simulation/reconstruction chains will look *something* like this with sim/reco processes (**S1…3**, **R1…5**) chained together to make intermediate files (**F1…5**)



Numbers of processes and intermediate files will vary from experiment to experiment

Anatomy of a LArSoft WorkFlow



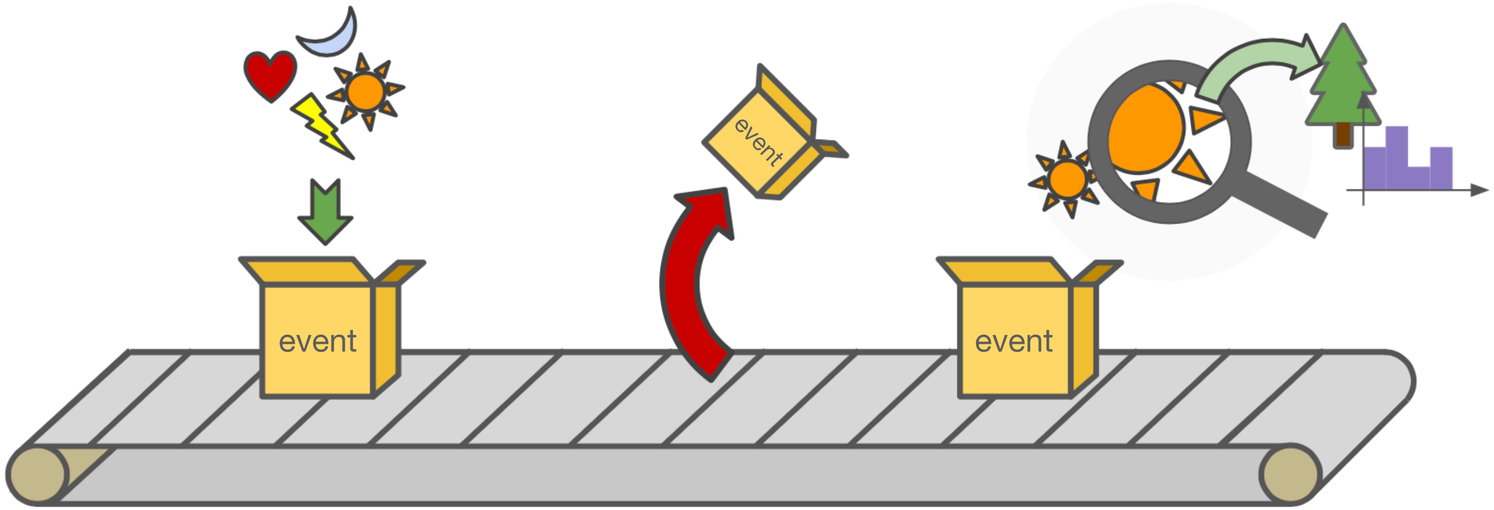
Anatomy of a LArSoft WorkFlow

The processes that we're dealing with for simulation are event generation, particle propagation, and detector response



LArSoft Modules

LArSoft is built on modules



Producer Module
Add data products to the event

Filter Module
Remove events based on some criteria

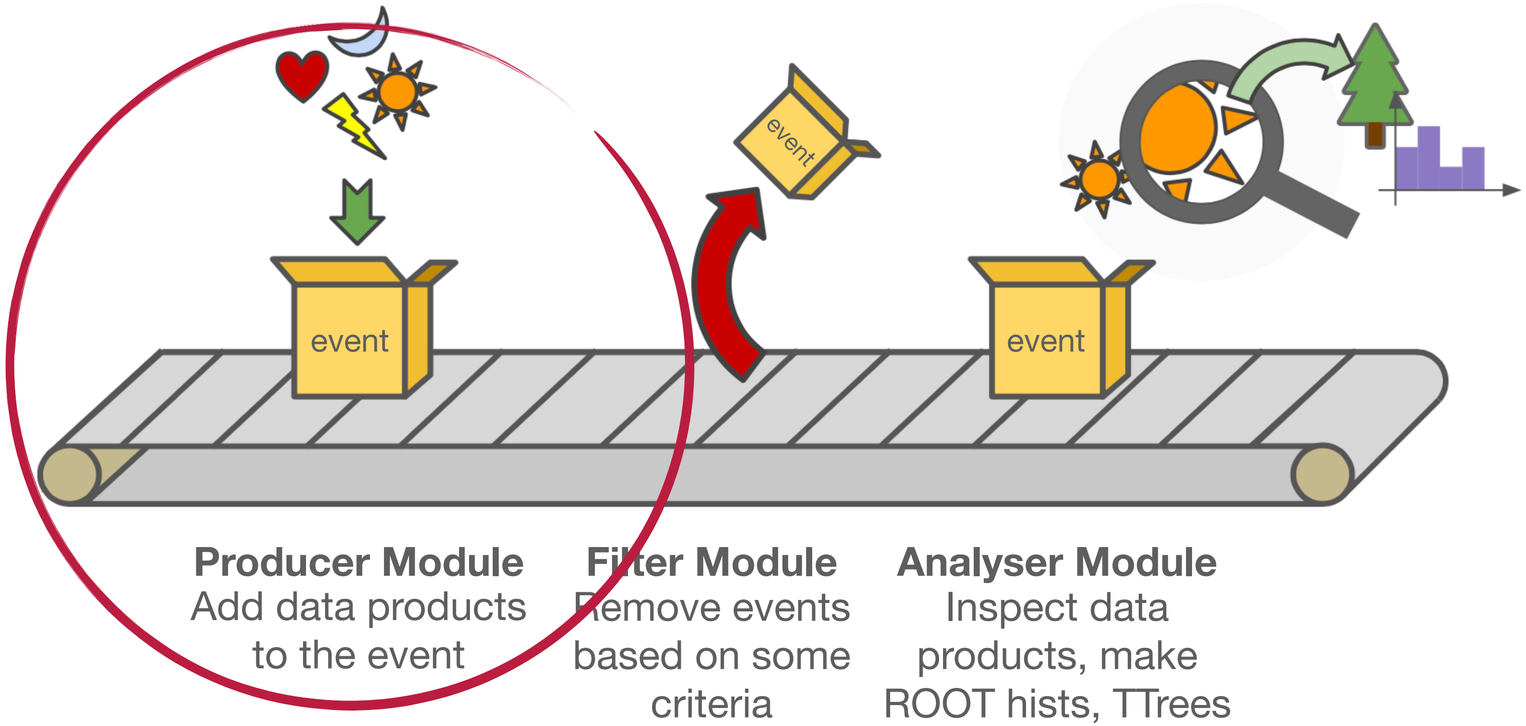
Analyser Module
Inspect data products, make ROOT hists, TTrees

A module is a self-contained piece of code that can generate new *data products* (stuff) or operate on *data products* that are already in files

LArSoft Modules

For simulation and reconstruction, almost entirely dealing with with producer modules

LArSoft is built on modules



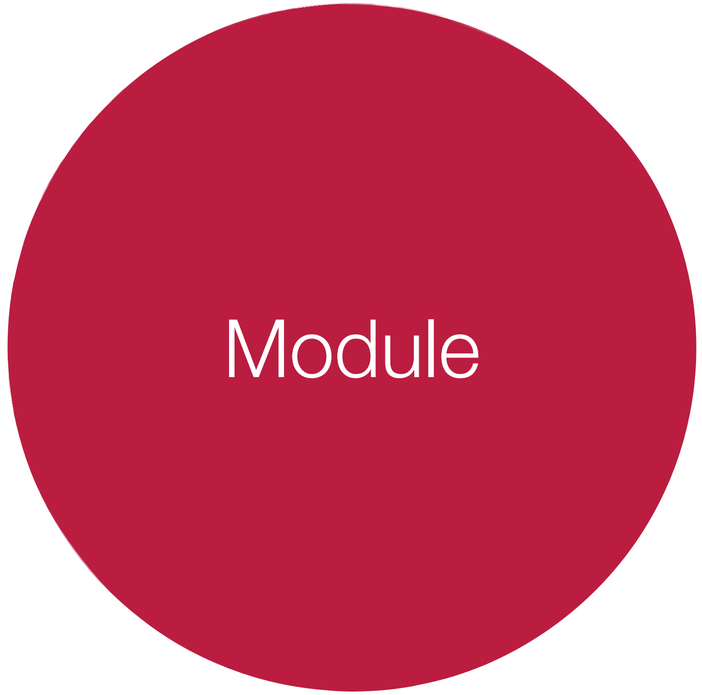
Producer Module
Add data products to the event

Filter Module
Remove events based on some criteria

Analyser Module
Inspect data products, make ROOT hists, TTrees

A module is a self-contained piece of code that can generate new *data products* (stuff) or operate on *data products* that are already in files

LArSoft Modules



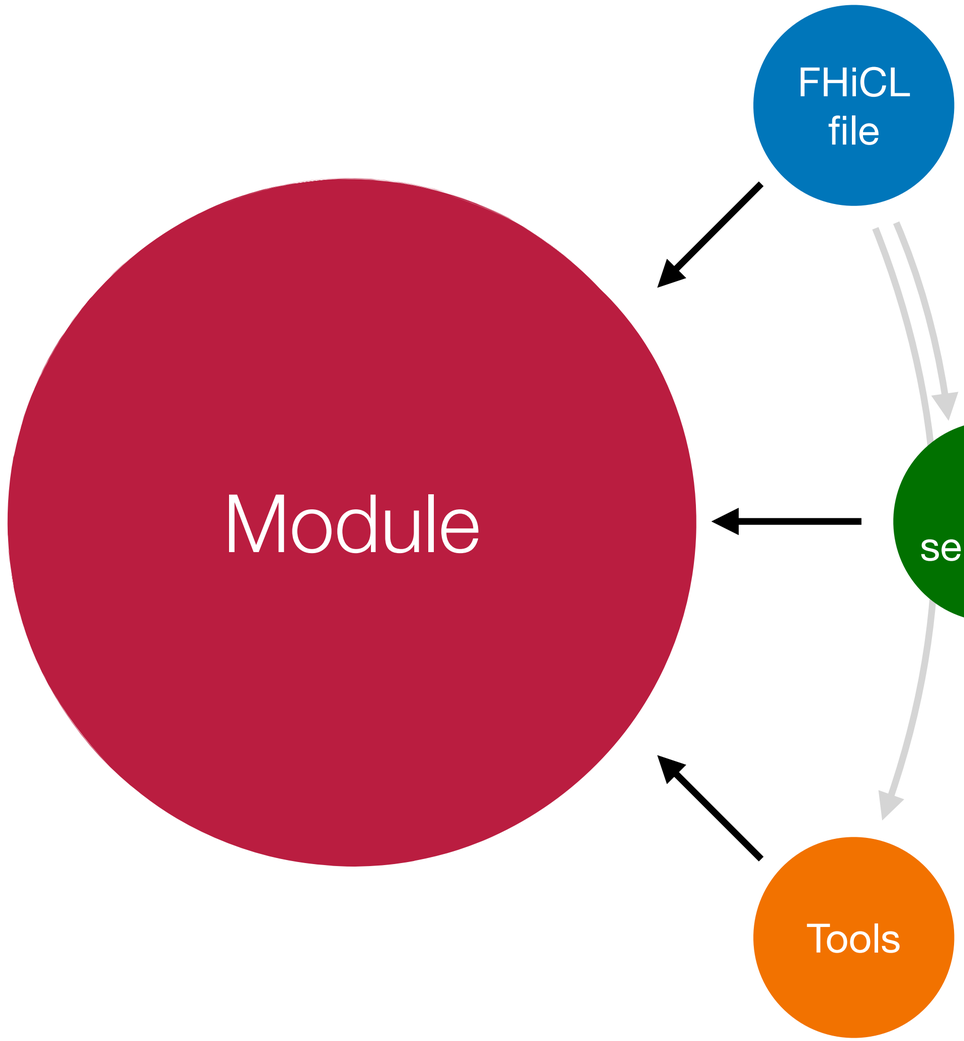
A module should do one thing,
eg generate events, process a reconstruction algorithm

Many modules can be linked together

Several key functions have an order that art knows to follow

Module Constructor	Constructor for the module
beginJob	Runs at the beginning of the job
beginEvent	Runs at the beginning of every event
produce/filter/analyze	“For every event, do these things”
endEvent	Runs at the end of every event
endJob	Runs at the end of every subrun

LArSoft Modules

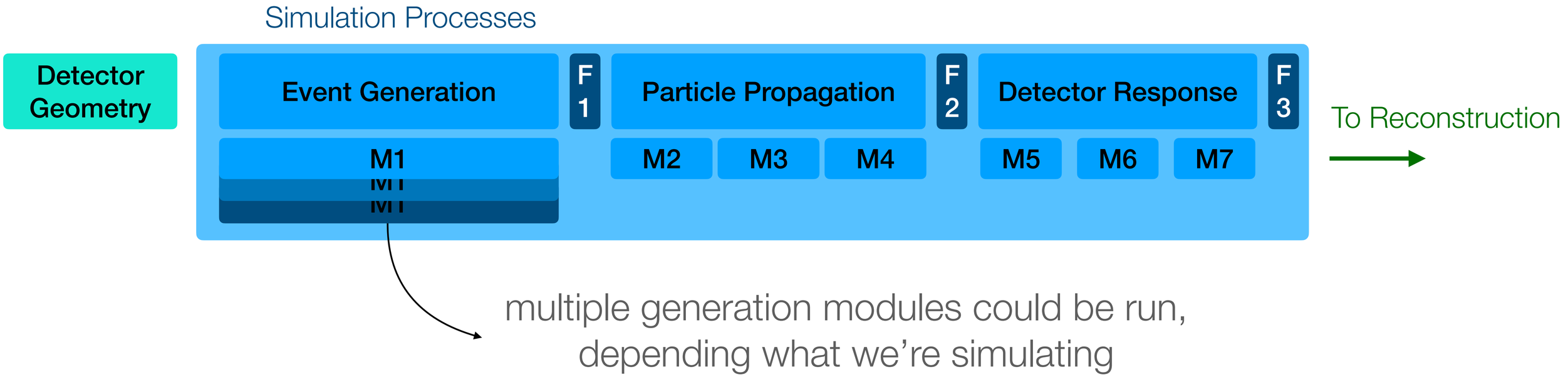


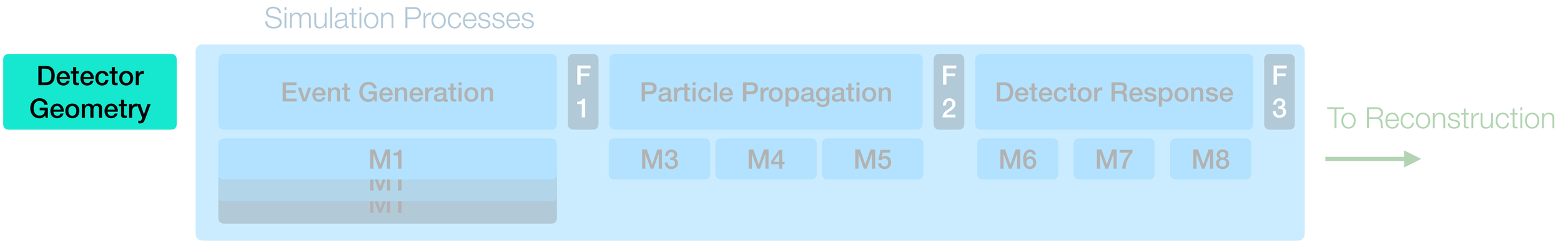
FHiCL: **F**ermilab **Hi**erarchical **C**onfiguration **L**anguage
Configuration file that tells art/LArSoft what modules, to run, and how to configure modules, services, tools

Utility *singleton* class that is configured once per job and can be accessed from any module
eg Geometry service for detector geo information
TFileService for interfacing with ROOT files

Auxiliary utility class that can be accessed by a module generally for algorithms that might have use in multiple modules. Unlike services, can be instantiated as many times as you like, with different configurations.

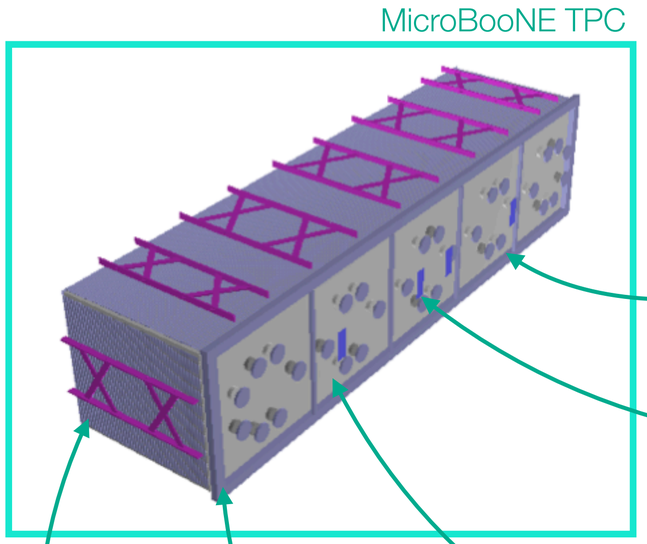
So expanding the example from earlier, it looks something like this…
with modules M1…M7




Simulation Processes
Detector Geometry
Event Generation
F 1
Particle Propagation
F 2
Detector Response
F 3
M1
M3
M4
M5
M6
M7
M8
To Reconstruction

A detector geometry is provided in **GDML format**

Geometry **D**escription **M**arkup **L**anguage




```
<material name="STEEL_STAINLESS_Fe7Cr2Ni" formula="STEEL_STAINLESS_Fe7Cr2Ni">
 <D value="7.9300" unit="g/cm3"/>
 <fraction n="0.0010" ref="carbon"/>
 <fraction n="0.1792" ref="chromium"/>
 <fraction n="0.7298" ref="iron"/>
 <fraction n="0.0900" ref="nickel"/>
</material>
```

volFieldCageTube{Top/Bot/Front/Back}

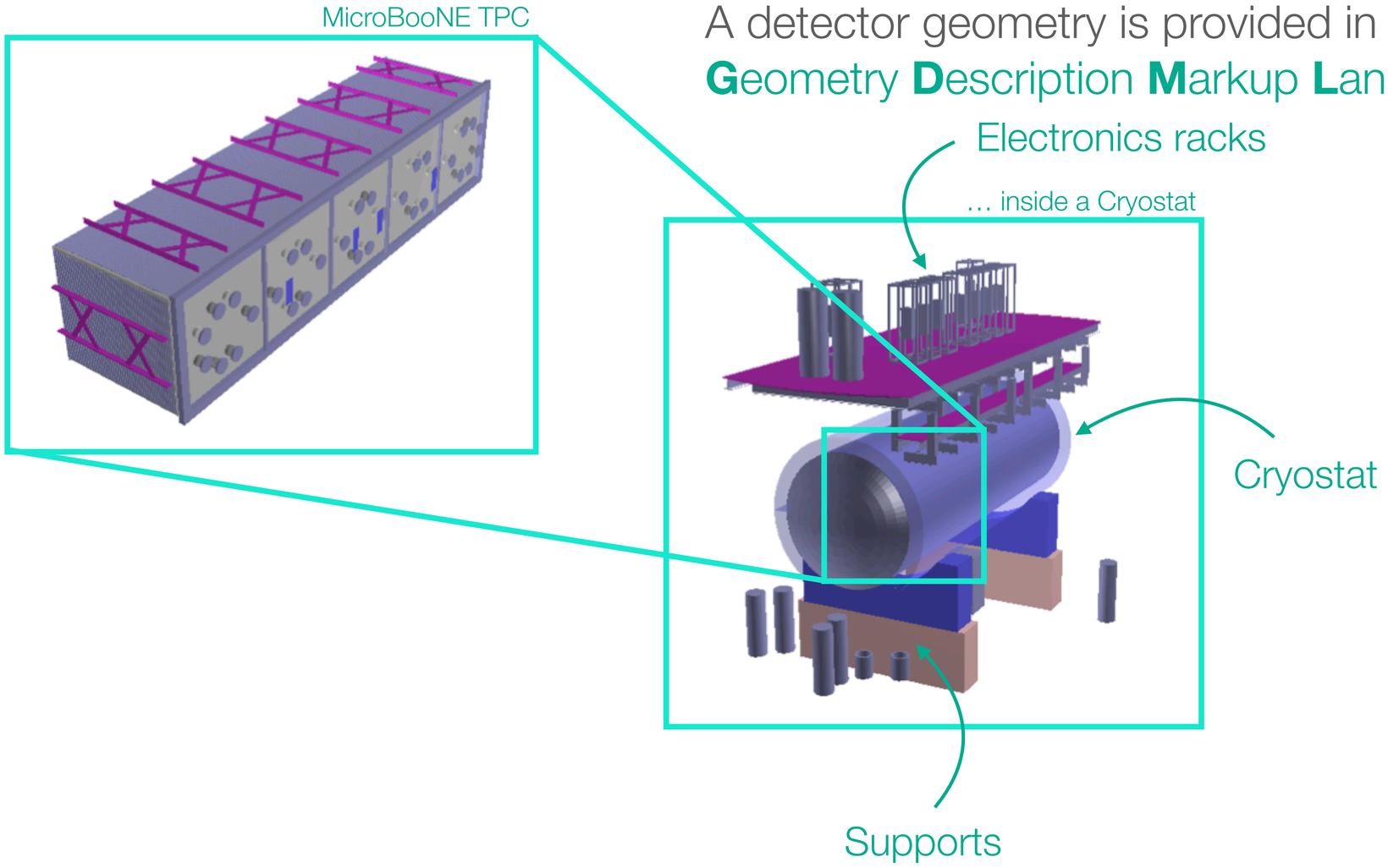
```
17  <structure>
18   <volume name="volFieldCageTubeTop">
19    <materialref ref="STEEL_STAINLESS_Fe7Cr2Ni"/>
20    <solidref ref="FieldCageTubeZ"/>
21   </volume>
22   <volume name="volFieldCageTubeBot">
23    <materialref ref="STEEL_STAINLESS_Fe7Cr2Ni"/>
24    <solidref ref="FieldCageTubeZ"/>
25   </volume>
26   <volume name="volFieldCageTubeFront">
27    <materialref ref="STEEL_STAINLESS_Fe7Cr2Ni"/>
28    <solidref ref="FieldCageTubeY"/>
29   </volume>
30   <volume name="volFieldCageTubeBack">
31    <materialref ref="STEEL_STAINLESS_Fe7Cr2Ni"/>
32    <solidref ref="FieldCageTubeY"/>
33   </volume>
34  </structure>
```

```
1   <solids>
2    <tube name="FieldCageTubeZ"
3     rmin="0.635"
4     rmax="1.27"
5     z="1026.84"
6     deltaphi="360"
7     aunit="deg"
8     lunit="cm"/>
9    <tube name="FieldCageTubeY"
10    rmin="0.635"
11    rmax="1.27"
12    z="222.84"
13    deltaphi="360"
14    aunit="deg"
15    lunit="cm"/>
16  </solids>
```

Each component is defined by a volume with a **geometry** and **material**, which can then be placed in the correct place to build up the detector

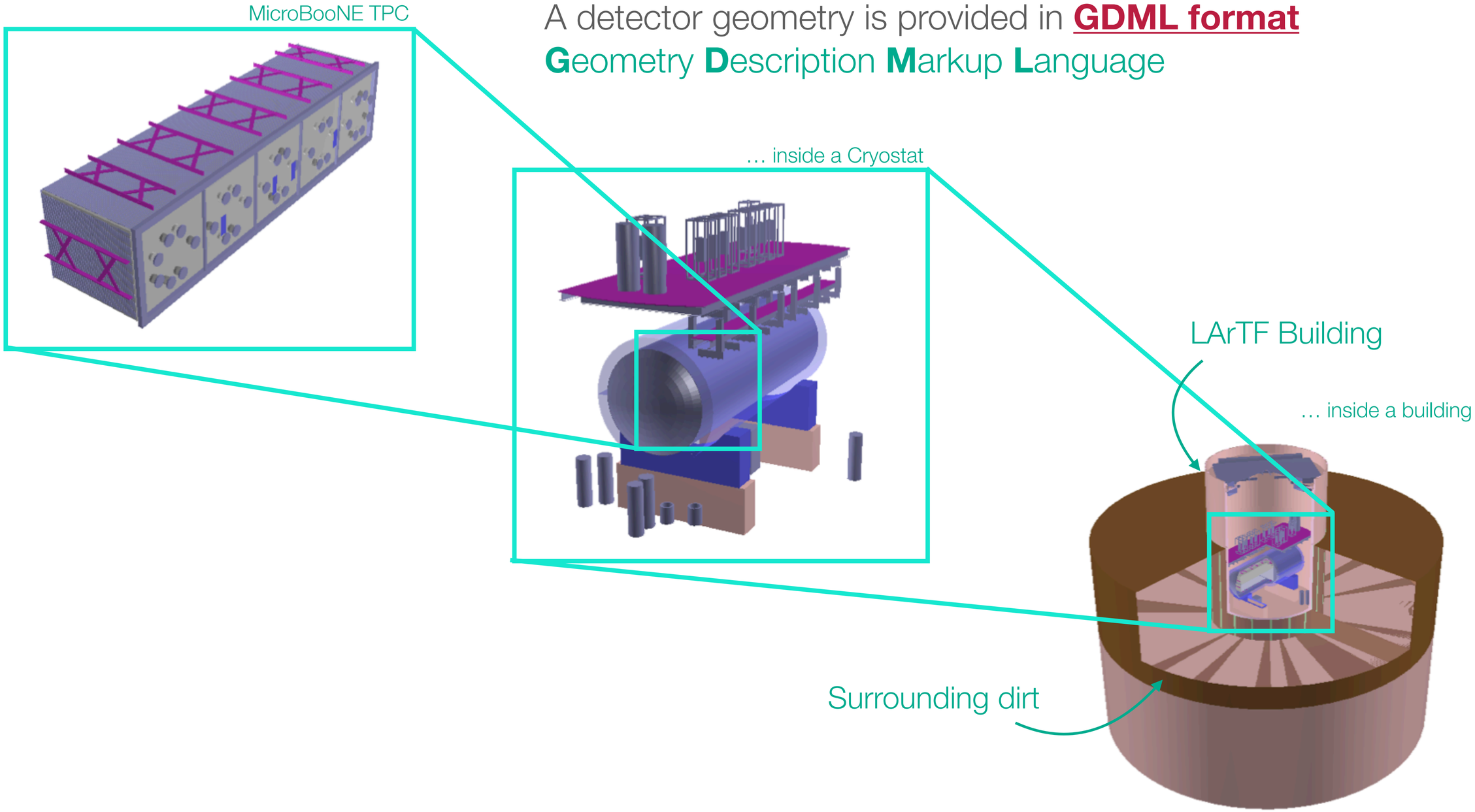
A detector geometry is provided in **GDML format**

Geometry **D**escription **M**arkup **L**anguage

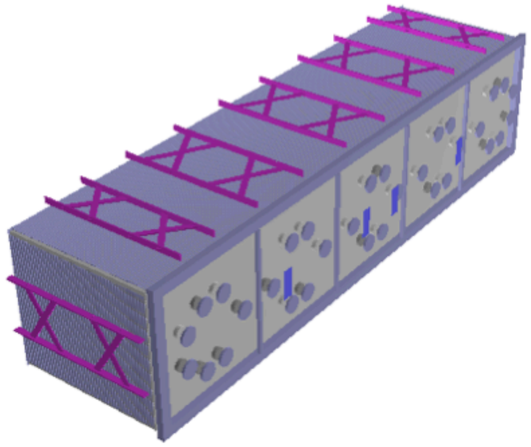


A detector geometry is provided in **GDML format**

Geometry **D**escription **M**arkup **L**anguage

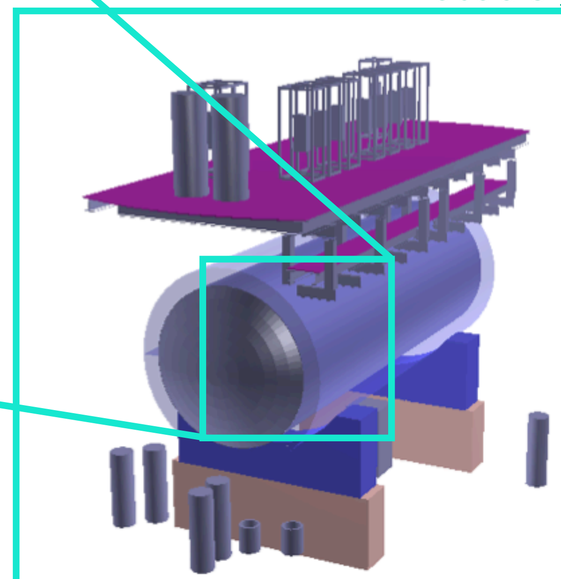


MicroBooNE TPC

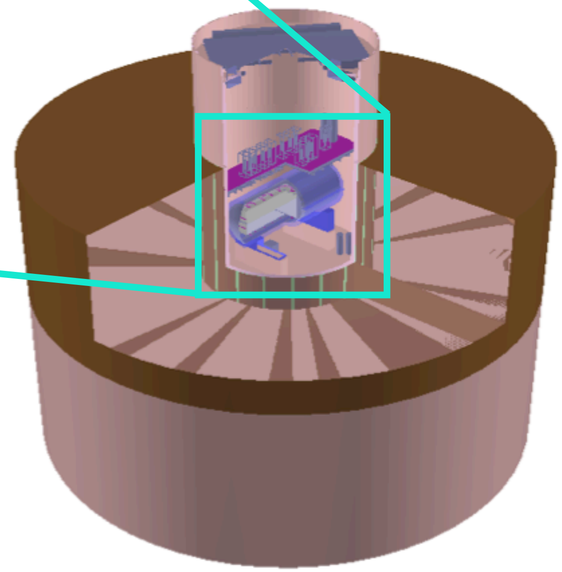


A detector geometry is provided in **GDML format**
Geometry **D**escription **M**arkup **L**anguage

… inside a Cryostat

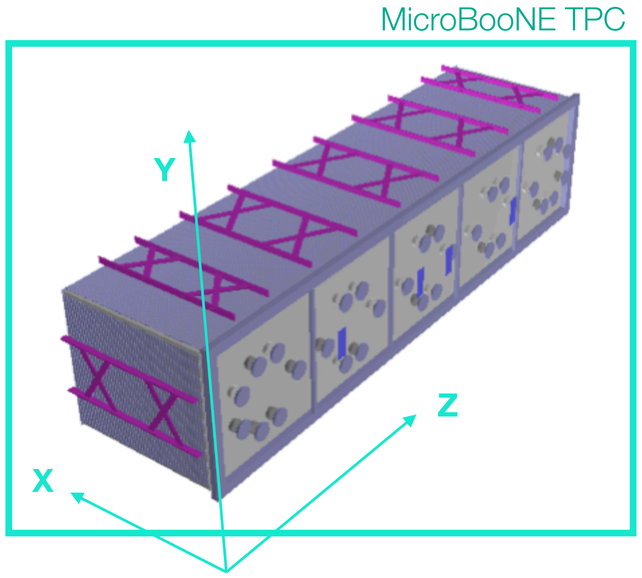


… inside a building



Needs to be detailed!

Need to understand neutrinos that interact in material surrounding the LAr, could be a background to analysis!



Coordinate System

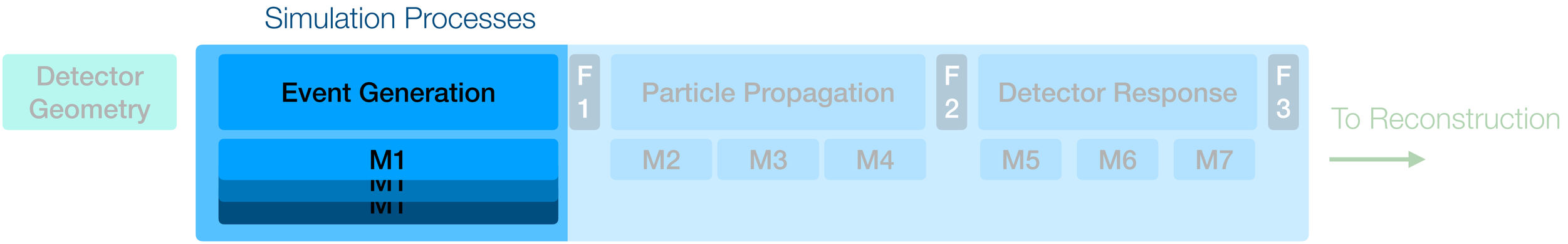
Z always points in the beam direction
Y points away from the Earth's surface
X direction is chosen to define a right-handed coordinate system

Origin is detector-specific

In general, you won’t need to deal with GDMLs directly

Experiments generally have these in place
You’ll get detector geometry from the Geometry service within LArSoft

A. Hackenburg Thesis: https://doi.org/10.2172/1423219


Simulation Processes
Detector Geometry
Event Generation
M1
F 1
Particle Propagation
M2
M3
M4
F 2
Detector Response
M5
M6
M7
F 3
To Reconstruction

The Simplest Generator: Single Particles

Can be used to simulate single particles in detectors

inside your prodsingle fcl file you'll have something like this

```
physics.producers.generator.PDG: [ 11 ]
physics.producers.generator.PosDist: 0
physics.producers.generator.P0:  [ 0.05 ]
physics.producers.generator.SigmaP: [ 0.0 ]
physics.producers.generator.X0: [0.0]
physics.producers.generator.Y0: [0.0]
physics.producers.generator.Z0: [300.0]
physics.producers.generator.SigmaX: [ 600. ]
physics.producers.generator.SigmaY: [ 600. ]
physics.producers.generator.SigmaZ: [ 600. ]
physics.producers.generator.AngleDist: 0
physics.producers.generator.Theta0XZ: [ 0. ]
physics.producers.generator.Theta0YZ: [ 0. ]
physics.producers.generator.SigmaThetaXZ: [ 180. ]
physics.producers.generator.SigmaThetaYZ: [ 90. ]
source.maxEvents: 10
```

Tells job which particle species to simulate

LEPTONS	
e^-	11
ν_e	12
μ^-	13
ν_μ	14
τ^-	15
ν_τ	16

LIGHT BARYONS	
p	2212
n	2112

LIGHT $I = 1$ MESONS	
π^0	111
π^+	211

STRANGE MESONS	
K_L^0	130
K_S^0	310
K^0	311
K^+	321

Full particle zoo available from **the PDG**

The Simplest Generator: Single Particles

Can be used to simulate single particles in detectors

inside your prodsingle fcl file you'll have something like this

```
physics.producers.generator.PDG: [ 11 ]
physics.producers.generator.PosDist: 0
physics.producers.generator.P0:  [ 0.05 ]
physics.producers.generator.SigmaP: [ 0.0 ]
physics.producers.generator.X0: [0.0]
physics.producers.generator.Y0: [0.0]
physics.producers.generator.Z0: [300.0]
physics.producers.generator.SigmaX: [ 600. ]
physics.producers.generator.SigmaY: [ 600. ]
physics.producers.generator.SigmaZ: [ 600. ]
physics.producers.generator.AngleDist: 0
physics.producers.generator.Theta0XZ: [ 0. ]
physics.producers.generator.Theta0YZ: [ 0. ]
physics.producers.generator.SigmaThetaXZ: [ 180. ]
physics.producers.generator.SigmaThetaYZ: [ 90. ]
source.maxEvents: 10
```

Tells job which particle species to simulate

… with some momentum
(with some spread, sigma)

… at some position
(with some spread, sigma)

… at some angle
(with some spread, sigma)

The Simplest Generator: Single Particles

Running the fhicl file will produce an *art*root file

artroot file != root file

you *can* open an artroot file in root,
but navigating it is going to become painful quite quickly

Not for the faint of heart

The Simplest Generator: Single Particles

Running the fhicl file will produce an *art*root file
You can see what's in an *art*root file by using eventdump.fcl

```
Begin processing the 1st record. run: 1 subRun: 0 event: 1 at 21-Oct-2026 04:38:10 CDT
PRINCIPAL TYPE: Event
PROCESS NAME | MODULE LABEL.. | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE......... | SIZE
SinglesGen.. | generator..... | ..................... | std::vector<simb::MCTruth> | ...1
SinglesGen.. | TriggerResults | ..................... | art::TriggerResults....... | ...1
```

These are just labels defined by the fhicl file

```
process_name: SinglesGen
```

```
producers:
{
  generator: @local::microboone_singlep
}
```

What type of data product was put into the event?

Class definitions can be found in doxygen

LArSoft doxygen DUNE doxygen SBN doxygen

The Simplest Generator: Single Particles

Running the fhicl file will produce an *art*root file
You can see what's in an *art*root file by using eventdump.fcl

```
Begin processing the 1st record. run: 1 subRun: 0 event: 1 at 21-Oct-2026 04:38:10 CDT
PRINCIPAL TYPE: Event
PROCESS NAME | MODULE LABEL.. | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE..... | SIZE
SinglesGen.. | generator..... | ..................... | std::vector<simb::MCTruth> | ...1
SinglesGen.. | TriggerResults | ..................... | art::TriggerResults..... | ...1
```

Public Member Functions

	MCTruth ()
const **simb::MCGeneratorInfo** &	**GeneratorInfo** () const
simb::Origin_t	**Origin** () const
int	**NParticles** () const
const **simb::MCParticle** &	**GetParticle** (int i) const

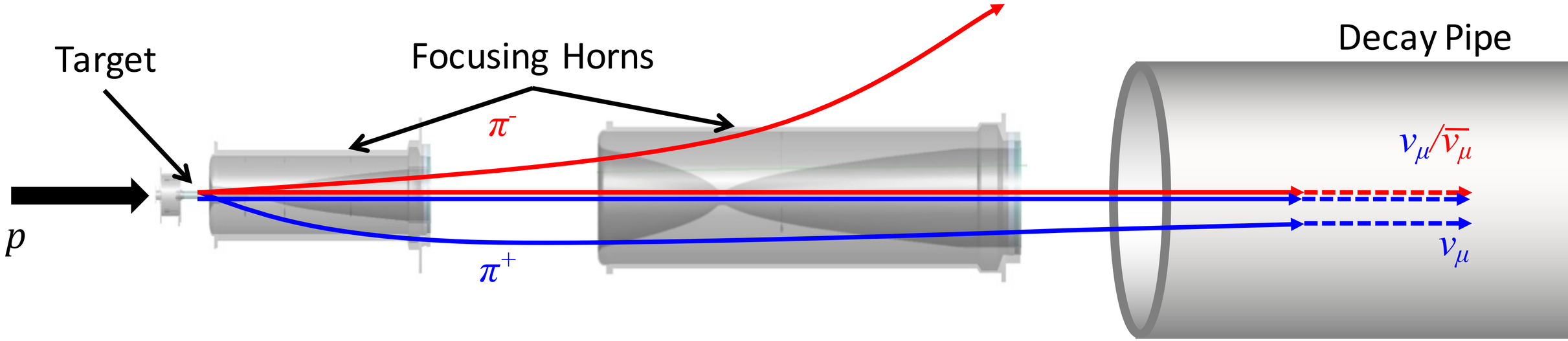
MCTruth Doxygen page

A generator will produce a set of MCTruth objects

These store information about what generator produced the event, and a collection of `MCParticles`, which define particle species, start points, momenta, and energies

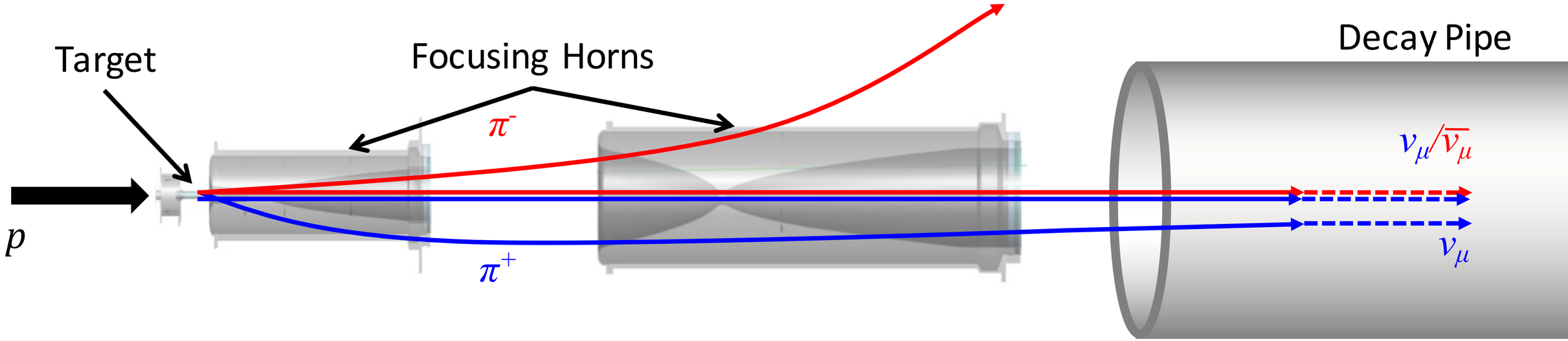
Generating Neutrinos

First up, we will need a beam simulation!
Simulate protons on the target, focusing of secondary particles, their decay into neutrinos, and their propagation to detectors

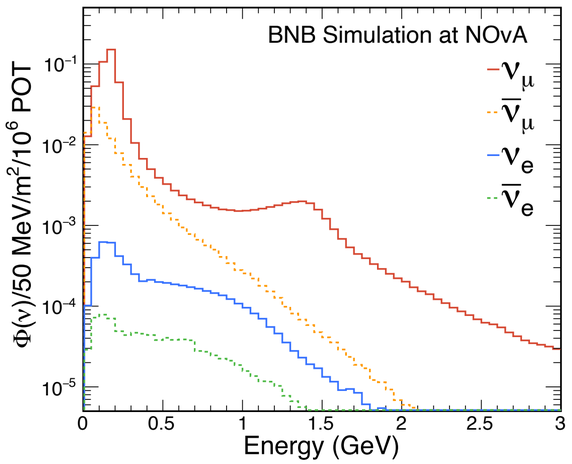


Generating Neutrinos

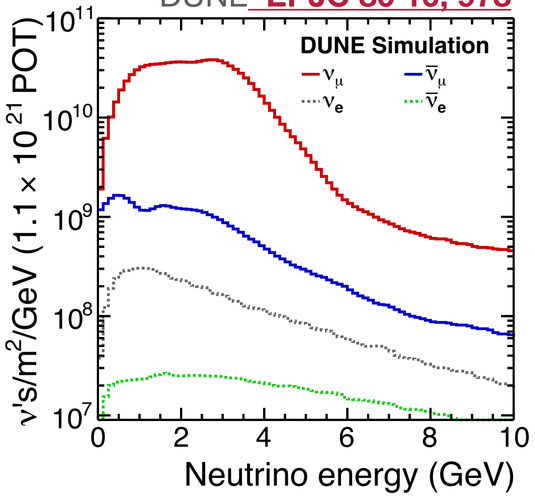
First up, we will need a beam simulation!
Simulate protons on the target, focusing of secondary particles, their decay into neutrinos, and their propagation to detectors



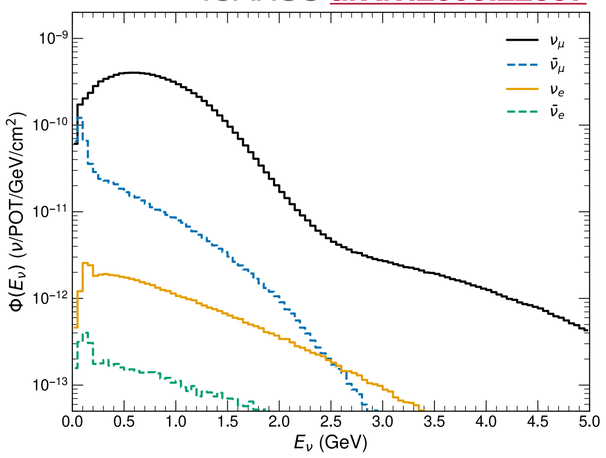
NOvA Simulation



DUNE **EPJC 80 10, 978**



ICARUS **arXiv:2603.22557**



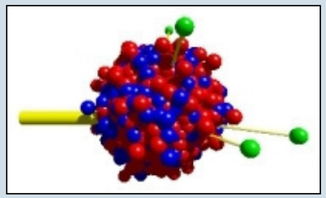
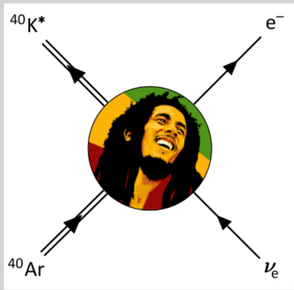
Done outside of LArSoft!

Produce “flux” files in one of a few formats that can be read into LArSoft generators

Generating Neutrinos

These neutrino flux files can then be passed into your favourite neutrino interaction generator

GENIE has become the standard for the LAr world, but there are many generators available

GENIE	NEUT	NuWro	GiBBU	MARLEY
Genie UNIVERSAL NEUTRINO GENERATOR & GLOBAL FIT				

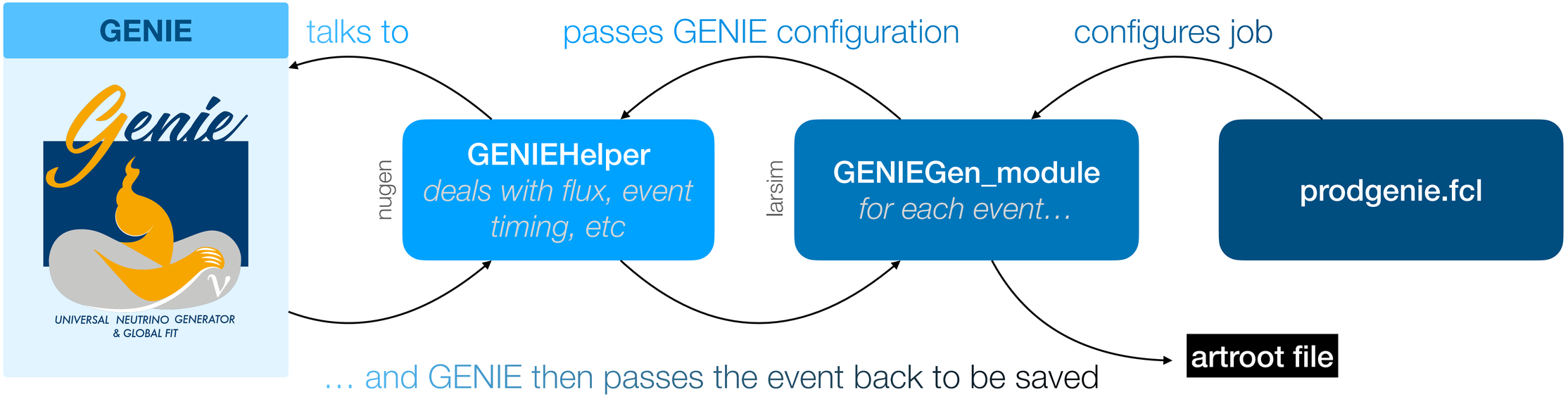
other generators are available

These generators randomly sample physics processes to decide **whether a given neutrino will interact**, **what process it will interact by** (quasi-elastic, resonant pion production, meson exchance current), **what particles it will produce, their dynamics**, and **whether the particles undergo any final state interactions**

Generating Neutrinos

These neutrino flux files can then be passed into your favourite neutrino interaction generator

GENIE has become the standard for the LAr world, but there are many generators available



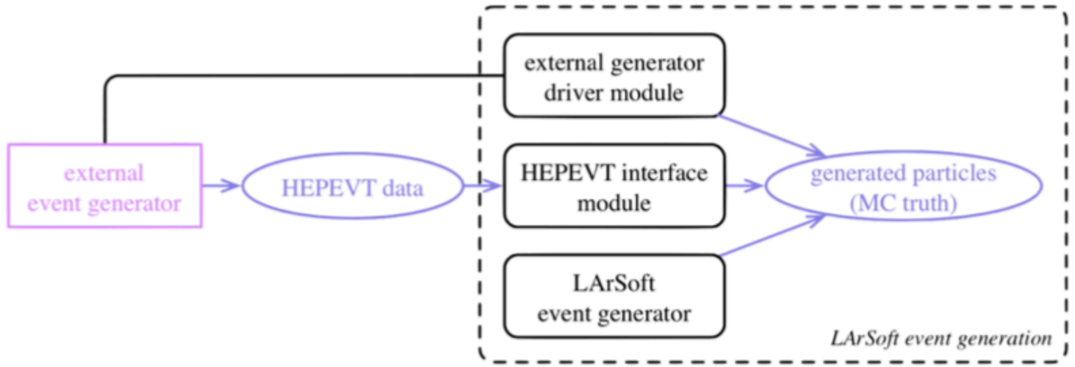
The generators are not native LArSoft tools, and so they have some interface so that LArSoft can tell the generator to generate an event

Other Generators

Cosmic Ray simulations:
Cosmic-induced particles are a major background to surface LArTPCs, and are a useful calibration source

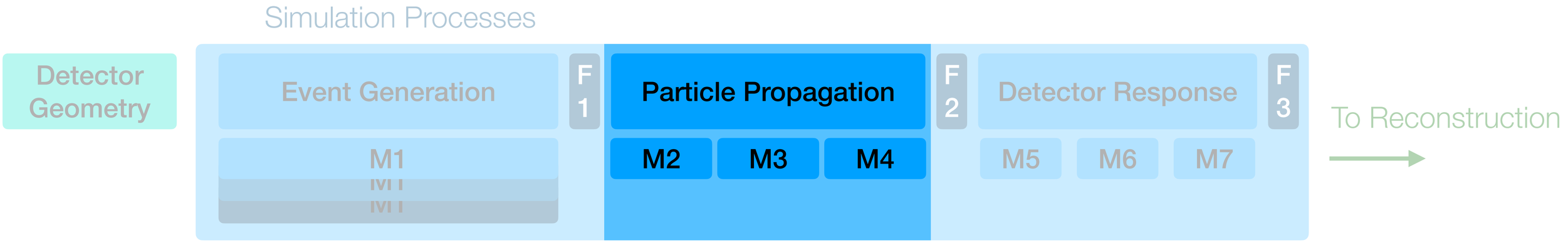
- **CRY**
- **CORSIKA**

TextFileGen:
For when there is no plugin for art/LArSoft, most useful for Beyond Standard Model searches that might not be covered by available generators



Uses **HEPEVT** standard data format

The route to generating events might be slightly different,
but **in the end we end up with a set of MCTruth objects**

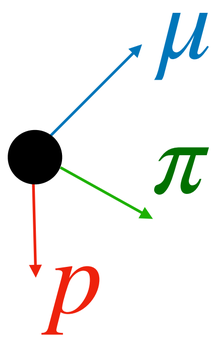

Simulation Processes
Detector Geometry
Event Generation
F 1
Particle Propagation
F 2
Detector Response
F 3
M1
M2
M3
M4
M5
M6
M7
To Reconstruction

Particle Propagation

Reminder: The MCTruth objects are really just a vertex position and a list of particles produced with their directions/momenta

As these particles travel, “stuff” happens, and we simulate this with Geant4





There are alternatives, eg **FLUKA**, or **MCNP**, but Geant4 is the standard and almost certainly what you will be using.

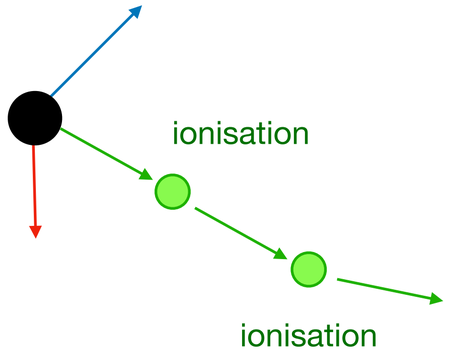
Particle Propagation

Reminder: The MCTruth objects are really just a vertex position and a list of particles produced with their directions/momenta

As these particles travel, “stuff” happens, and we simulate this with Geant4



Each time Geant4 takes a step, it randomly samples what physics processes could happen



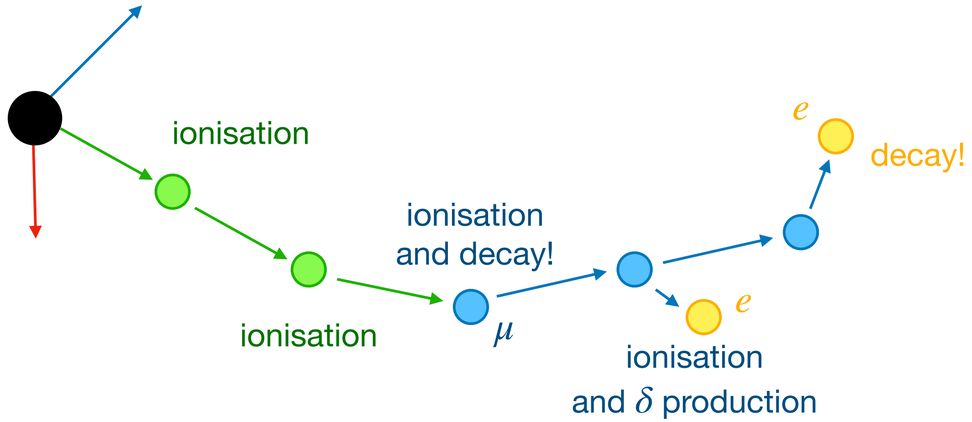
Particle Propagation

Reminder: The MCTruth objects are really just a vertex position and a list of particles produced with their directions/momenta

As these particles travel, “stuff” happens, and we simulate this with Geant4



Each time Geant4 takes a step, it randomly samples what physics processes could happen



When a new particle is produced, it is added to the list of particles to track

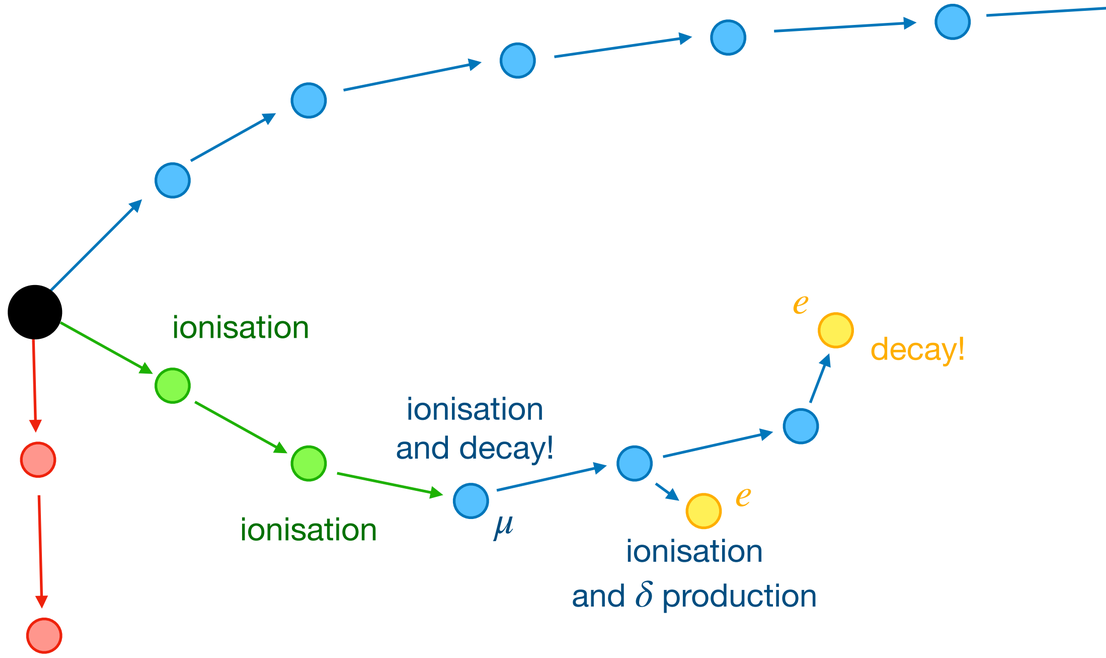
Particle Propagation

Reminder: The MCTruth objects are really just a vertex position and a list of particles produced with their directions/momenta

As these particles travel, “stuff” happens, and we simulate this with Geant4



Each time Geant4 takes a step, it randomly samples what physics processes could happen



When a new particle is produced, it is added to the list of particles to track

… and it iterates until it has tracked all particles

Particle Propagation

Exactly what processes are considered will change based on how Geant4 is configured

Different physics live in different “physics lists” and are fhicl configurable

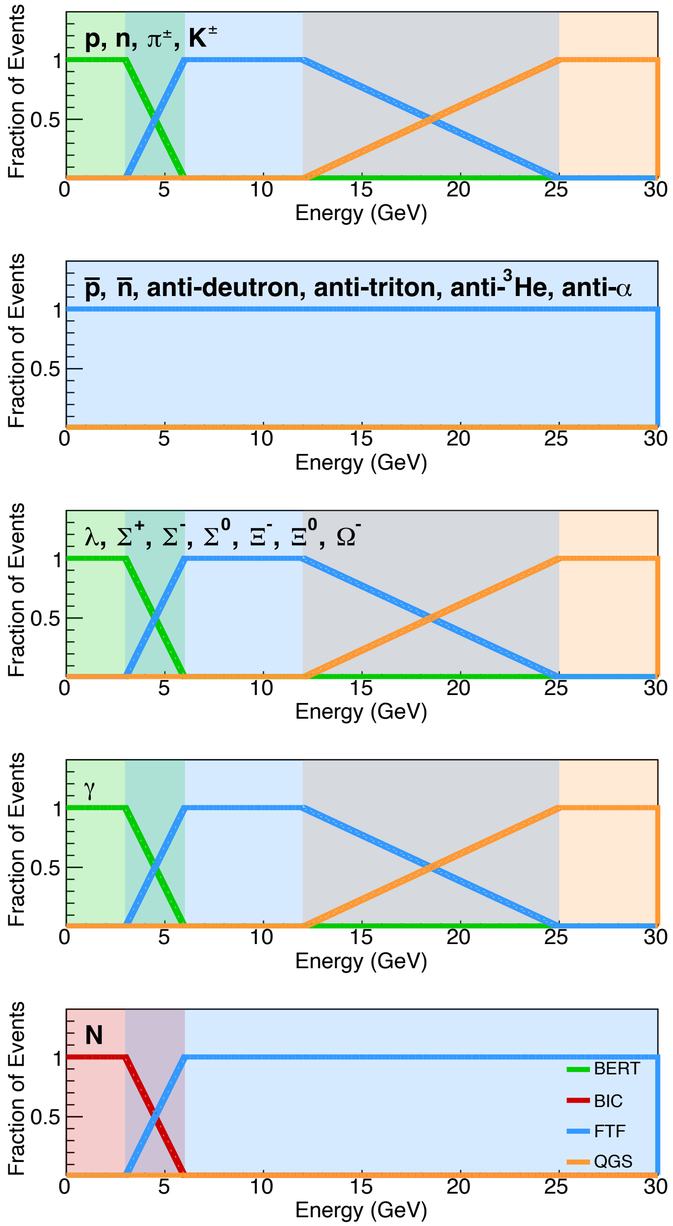
Uses Quark Gluon String model at high energies ← **QGSP_BERT_HP_EMZ** → “EMZ” or “Opt4” is collection of most accurate EM models

Bertini Cascade model at low energies

High Precision neutron model

Details on the physics lists can be found in the **G4 physics list guide**

Models for Geant4 10.6 configured with QGSP_BERT_HP_EMZ



Particle Propagation

Exactly what processes are considered will change based on how Geant4 is configured

Different physics live in different “physics lists” and are fhicl configurable

Uses Quark Gluon String model at high energies ← **QGSP_BERT_HP_EMZ** → “EMZ” or “Opt4” is collection of most accurate EM models

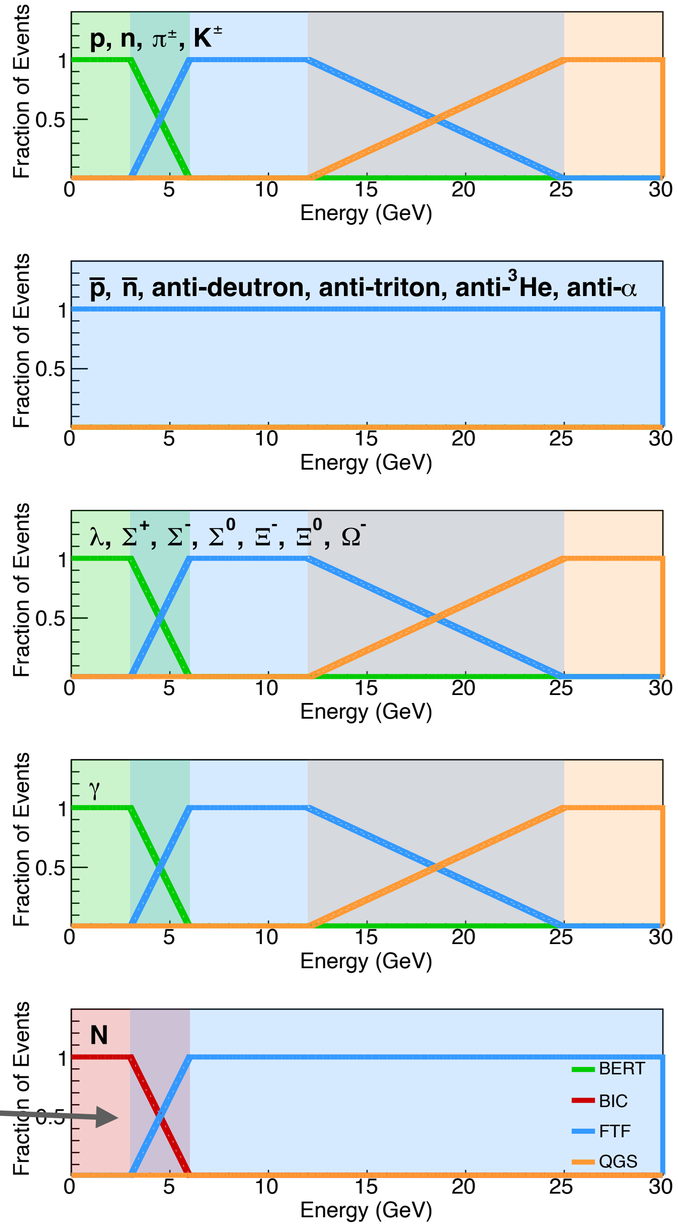
Bertini Cascade model at low energies

High Precision neutron model

Details on the physics lists can be found in the **G4 physics list guide**

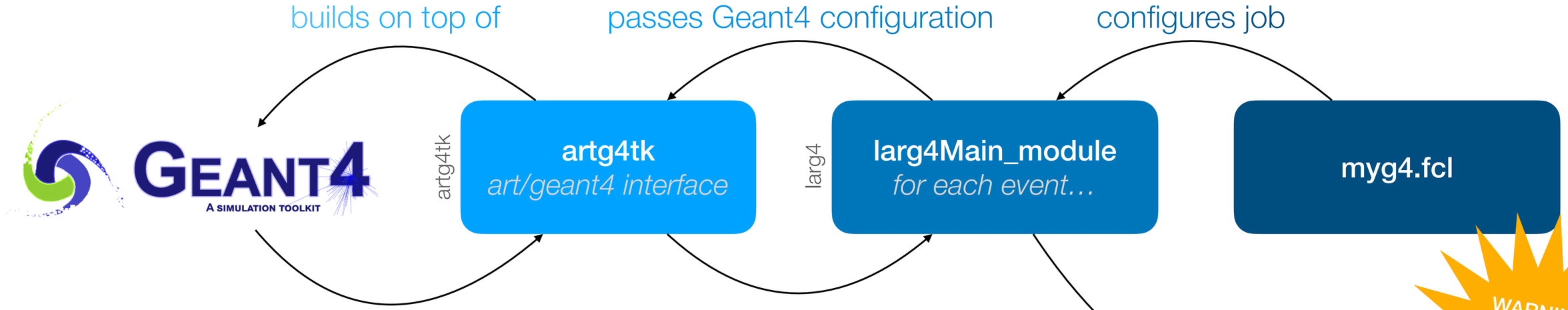
In transition regions, G4 randomly chooses the model, linearly scaling down (up) the fraction of the time it chooses the low-energy (high-energy) model.

Models for Geant4 10.6 configured with QGSP_BERT_HP_EMZ



Geant4 in LArSoft

Similar to with generators, G4 is an external tool, and we use a module, LArG4 (or largeant), which talks to **artg4tk**, which interfaces art and Geant4



…though for the propagation step we're not just running largeant - what are these extra modules doing?

```
producers:
{

  #retain largeant name for compatibility
  largeant:     @local::protodune_larg4
  IonAndScint:  @local::protodunehd_ionandscint
  IonAndScintExternal: @local::protodunehd_ionandscint_external
  PDFastSim:    @local::protodune_hd_pdfastsim_par
  PDFastSimExternal: @local::protodune_hd_pdfastsim_pvs_external
  rns: {module_type: "RandomNumberSaver"}
}
```

WARNING: Could change per experiment!

An Aside: Navigating fcl files

FHiCL files are genius because they allow you to inherit a configuration and overwrite a specific parameter

mybasefhicl.fcl

```
paramblock: {
   myparam1: true
   myparam2:10
}
```



mymodifiedfhicl.fcl

```
#include mybasefhicl.fcl

paramblock.myparam2: 20
```

```
producers:
{

  #retain largeant name for compatibility
  largeant:     @local::protodune_larg4
  IonAndScint:  @local::protodunehd_ionandscint
  IonAndScintExternal: @local::protodunehd_ionandscint_external
  PDFastSim:    @local::protodune_hd_pdfastsim_par
  PDFastSimExternal: @local::protodune_hd_pdfastsim_pvs_external
  rns: {module_type: "RandomNumberSaver"}
}
```

An Aside: Navigating fcl files

FHiCL files are genius because they allow you to inherit a configuration and overwrite a specific parameter

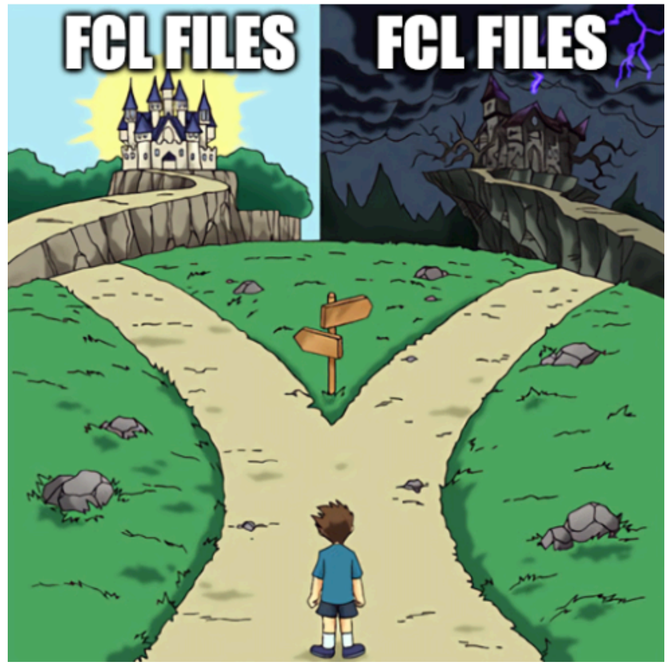
mybasefhicl.fcl

```
paramblock: {
   myparam1: true
   myparam2:10
}
```

mymodifiedfhicl.fcl

```
#include mybasefhicl.fcl

paramblock.myparam2: 20
```



FHiCL files are a nightmare because they allow you to inherit a configuration and overwrite a specific parameter

mybasefhicl.fcl

```
paramblock: {
   myparam1: true
   myparam2:10
}
```

- mymodifiedfhicl_1.fcl
  - mymodifiedfhicl_1_1.fcl
    - mymodifiedfhicl_1_1_1.fcl
- mymodifiedfhicl_2.fcl
  - mymodifiedfhicl_2_1.fcl
    - mymodifiedfhicl_2_1_1.fcl

```
producers:
{

  #retain largeant name for compatibility
  largeant:     @local::protodune_larg4
  IonAndScint:  @local::protodunehd_ionandscint
  IonAndScintExternal: @local::protodunehd_ionandscint_external
  PDFastSim:    @local::protodune_hd_pdfastsim_par
  PDFastSimExternal: @local::protodune_hd_pdfastsim_pvs_external
  rns: {module_type: "RandomNumberSaver"}
}
```

An Aside: Navigating fcl files

Not just a problem in theory, it happens a lot in development!

```
standard_g4_protodunehd_stage2_sce_E500.fcl
|--standard_g4_protodunehd_stage2.fcl
|----standard_g4_protodunehd.fcl
|------services_refactored_pdune.fcl
|--------services_dune.fcl
|--------larg4services_dune.fcl
|----------geometry_dune.fcl
|------LArG4_dune.fcl
|--------LArG4.fcl
|------IonAndScint_dune.fcl
|------PDFastSim_dune.fcl
|--------PDFastSimPAR.fcl
|----------opticalsimparameterisations.fcl
|----------scintillationtime_tool.fcl
|--------PDFastSimPVS.fcl
|----------scintillationtime_tool.fcl
|--------ScintTimeXeDoping.fcl
|--------opticalsimparameterisations_dune.fcl
```

You can manually navigate FHICL inheritance by looking in the paths in `$FHICL_FILE_PATH`

alternatively use `findfcl` or `find-fhicl.sh` if they're available on your machine (implementation in backups!)

```
producers:
{

  #retain largeant name for compatibility
  largeant:     @local::protodune_larg4
  IonAndScint:  @local::protodunehd_ionandscint
  IonAndScintExternal: @local::protodunehd_ionandscint_external
  PDFastSim:    @local::protodune_hd_pdfastsim_par
  PDFastSimExternal: @local::protodune_hd_pdfastsim_pvs_external
  rns: {module_type: "RandomNumberSaver"}
}
```

An Aside: Navigating fcl files

`fhicl-dump` is another useful tool which will expand all configurations in the fhicl file, and should be available when LArSoft is setup.

A full, top-level fhicl file which you could run

```
#include "services_refactored_pdune.fcl"
#include "singles_dune.fcl"

process_name: SinglesGen

services:
{
  TFileService: { fileName: "hist.root" }
  RandomNumberGenerator: {}
}

source:
{
  module_type: EmptyEvent
  maxEvents:   10
}

physics:
{
 producers:
 {
   generator: @local::microboone_singlep
 }

 simulate: [ generator ]
 stream1:  [ out1 ]
 trigger_paths: [simulate]
 end_paths:     [stream1]
}

outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "artroot.root"
 }
}
```

Full configuration, attained with fhicl-dump

```
# Produced from 'fhicl-dump' using:
#   Input  : myprodsingle.fcl
#   Policy : cet::filepath_maker
#   Path   : "FHICL_FILE_PATH"

outputs: {  # ./myprodsingle.fcl:32
   out1: {  # ./myprodsingle.fcl:34
      fileName: "artroot.root"  # ./myprodsingle.fcl:36
      module_type: "RootOutput"  # ./myprodsingle.fcl:35
   }
}
physics: {  # ./myprodsingle.fcl:19
   end_paths: [  # ./myprodsingle.fcl:28
      "stream1"  # ./myprodsingle.fcl:28
   ]
   producers: {  # ./myprodsingle.fcl:21
      generator: {  # ./myprodsingle.fcl:22
         AngleDist: "Gaussian"  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:31
         P0: [  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:12
            6  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:12
         ]
         PDG: [  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:11
            13  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:11
         ]
         PDist: "Gaussian"  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:14
         PadOutVectors: false  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:9
         ParticleSelectionMode: "all"  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:8
         PosDist: "uniform"  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:25
         SigmaP: [  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:13
            0  # /cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/job/singles.fcl:13
         ]
```

`-a` option will print out a comment telling you where a given parameter is defined

Geant4 in LArSoft

```
Apptainer> find $LARSIM_DIR -name "IonAndScint_module.cc"
/cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/source/larsim/IonizationScintillation/IonAndScint_module.cc
```

which gives us a hint to look in LARSIM

```
Apptainer> lar --print-available module | grep IonAndScint
270. IonAndScint                              producer        / [ external source ] /larsim/IonizationScintillation/IonAndScint_module.cc
```

```
producers: {
   IonAndScint: {
      ISCalcAlg: "Separate"
      Instances: "LArG4DetectorServicevolTPCActiveInner"
      SavePriorSCE: true
      module_type: "IonAndScint"
   }
   IonAndScintExternal: {
      ISCalcAlg: "Separate"
      Instances: "LArG4DetectorServicevolTPCActiveOuter"
      SavePriorSCE: true
      module_type: "IonAndScint"
   }
```

Once you know the module name, you can get a clue where to find it with lar --print-available module

fhicl-dump

```
producers:
{

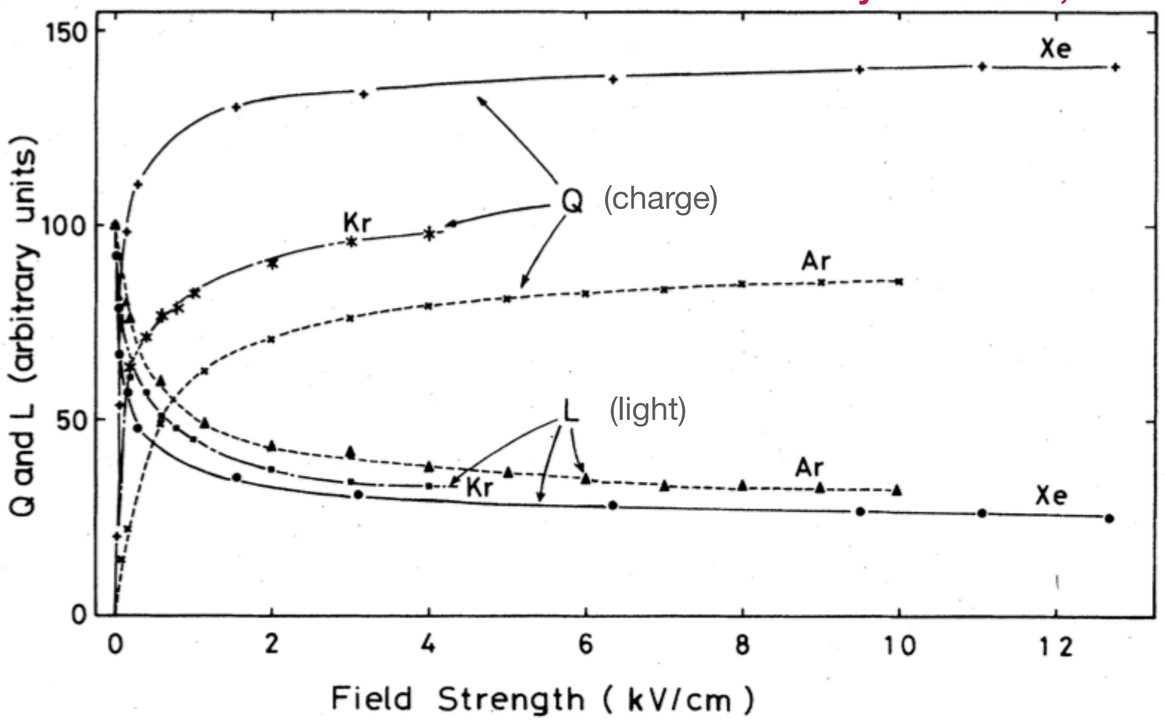
  #retain largeant name for compatibility
  largeant:     @local::protodune_larg4
  IonAndScint:  @local::protodunehd_ionandscint
  IonAndScintExternal: @local::protodunehd_ionandscint_external
  PDFastSim:    @local::protodune_hd_pdfastsim_par
  PDFastSimExternal: @local::protodune_hd_pdfastsim_pvs_external
  rns: {module_type: "RandomNumberSaver"}
}
```

Armed with this, we can find out that the IonAndScint and IonandScintExternal configurations are both calling the same module, IonAndScint, just with different options

Geant4 in LArSoft

```
Apptainer> find $LARSIM_DIR -name "IonAndScint_module.cc"
/cvmfs/larsoft.opensciencegrid.org/products/larsim/v09_44_00/source/larsim/IonizationScintillation/IonAndScint_module.cc
```

Phys. Rev. B 20, 3486



The IonAndScint module takes the output of largeant and applies recombination effects that depend on the *E*-field.

This is why there are two calls to the module, the first is inside the TPC, where there is an E field, the second is outside the TPC, where there isn't

```
producers:
{

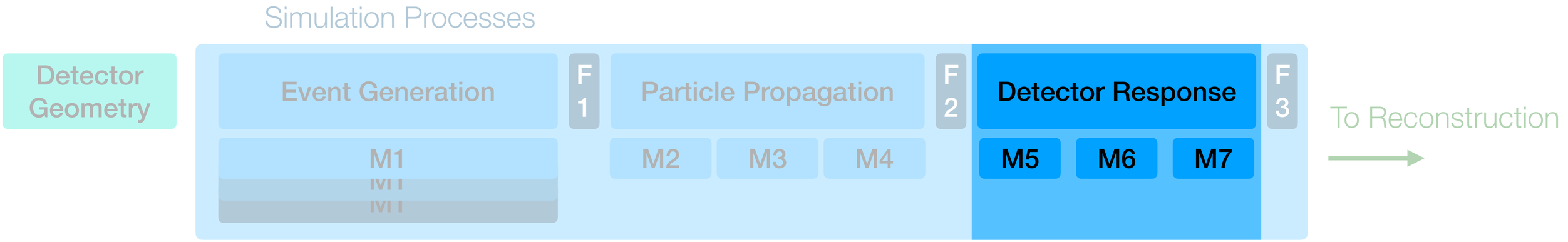
  #retain largeant name for compatibility
  largeant:     @local::protodune_larg4
  IonAndScint:  @local::protodunehd_ionandscint
  IonAndScintExternal: @local::protodunehd_ionandscint_external
  PDFastSim:    @local::protodune_hd_pdfastsim_par
  PDFastSimExternal: @local::protodune_hd_pdfastsim_pvs_external
  rns: {module_type: "RandomNumberSaver"}
}
```

What Do We have After LArG4?

PROCESS NAME	MODULE LABEL	PRODUCT INSTANCE NAME	DATA PRODUCT TYPE	SIZE
SinglesGen..	generator		std::vector<simb::MCTruth>	...1
SinglesGen..	TriggerResults		art::TriggerResults	...1
G4.....	IonAndScintExternal		std::vector<sim::SimEnergyDeposit>	...0
G4.....	PDFastSimExternal..		std::vector<sim::SimPhotonsLite>	.160
G4.....	PDFastSimExternal..		std::vector<sim::OpDetBacktrackerRecord>	...0
G4.....	PDFastSim.....		std::vector<sim::OpDetBacktrackerRecord>	..80
G4.....	rns.....		std::vector<art::RNGsnapshot>	...7
G4.....	PDFastSim.....		std::vector<sim::SimPhotonsLite>	.160
G4.....	IonAndScint.....		std::vector<sim::SimEnergyDeposit>	.879
G4.....	largeant.....	LArG4DetectorServicevolTPCActiveInner	std::vector<sim::SimEnergyDeposit>	...?
G4.....	TriggerResults.....		art::TriggerResults	...1
G4.....	largeant.....		std::vector<simb::MCParticle>	.114
G4.....	largeant.....		sim::ParticleAncestryMap	...-
G4.....	largeant.....		art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo>	.114
G4.....	IonAndScint.....	priorSCE	std::vector<sim::SimEnergyDeposit>	.879
G4.....	IonAndScintExternal	priorSCE	std::vector<sim::SimEnergyDeposit>	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCActiveOuter	std::vector<sim::SimEnergyDeposit>	...?

If we look in the output artroot file (using eventdump.fcl), you can see we have:

- **The `MCTruth`s from the generator stage**
- **A set of `MCParticle`s from running the `largeant` module** and an **association (`art:Assn`) that lets us map between `MCParticle`s and the `MCTruth` object.**
- **Some `SimEnergyDeposit`s from largeant that have been *dropped***
- **The output `SimEnergyDeposit`s from running the `IonAndScint` module**
- A bunch of stuff related to photon/PDS simulation that will be covered tomorrow


Simulation Processes
Detector Geometry
Event Generation
F 1
Particle Propagation
F 2
Detector Response
F 3
M1
M2
M3
M4
M5
M6
M7
To Reconstruction

Detector Simulation



We have some simulated particles that have recombination applied, but what would our detector actually see?


JINST

PUBLISHED BY IOP PUBLISHING FOR SISSA MEDIALAB

RECEIVED: February 27, 2018
ACCEPTED: April 23, 2018
PUBLISHED: July 6, 2018

Ionization electron signal processing in single phase LArTPCs. Part I. Algorithm Description and quantitative evaluation with MicroBooNE simulation

MicroBooNE Collaboration

C. Adams,[i] R. An,[j] J. Anthony,[c] J. Asaadi,[aa] M. Auger,[a] L. Bagby,[h] S. Balasubramanian,[ee] B. Baller,[h] C. Barnes,[p] G. Barr,[s] M. Bass,[s,b] F. Bay,[bb] A. Bhat,[x] K. Bhattacharya,[t] M. Bishai,[b] A. Blake,[l] T. Bolton,[k] L. Camilleri,[g] D. Caratelli,[g] R. Castillo Fernandez,[h] F. Cavanna,[h] G. Cerati,[h] H. Chen,[b] Y. Chen,[a] E. Church,[t] D. Cianci,[g] E. Cohen,[y] G.H. Collin,[o] J.M. Conrad,[o] M. Convery,[w] L. Cooper-Troendle,[ee] J.I. Crespo-Anadón,[g] M. Del Tutto,[s] D. Devitt,[l] A. Diaz,[o] S. Dytman,[u] B. Eberly,[w] A. Ereditato,[a] L. Escudero Sanchez,[c] J. Esquivel,[x] J.J. Evans,[n] A.A. Fadeeva,[g] B.T. Fleming,[ee] W. Foreman,[d] A.P. Furmanski,[n] D. Garcia-Gamez,[n] G.T. Garvey,[m] V. Genty,[g] D. Goeldi,[a] S. Gollapinni,[r] E. Gramellini,[ee] H. Greenlee,[h] R. Grosso,[c] R. Guenette,[s,i] P. Guzowski,[n] A. Hackenburg,[ee] P. Hamilton,[x] O. Hen,[o] J. Hewes,[n] C. Hill,[n] J. Ho,[d] G.A. Horton-Smith,[k] A. Hourlier,[o] E.-C. Huang,[m] C. James,[h] J. Jan de Vries,[c] L. Jiang,[u] R.A. Johnson,[e] J. Joshi,[b] H. Jostlein,[h] Y.-J. Jwa,[g] D. Kaleko,[g] G. Karagiorgi,[g] W. Ketchum,[h] B. Kirby,[b] M. Kirby,[h] T. Kobilarcik,[h] I. Kreslo,[a]

2018 JINST 13

In principle, each detector could have a custom detector simulation

In practice, many pieces of the response simulation are common to all experiments, and so we hand off the detector response simulation to WireCell

Detector Simulation

To get to a fully simulated waveform, M, we need five pieces

$$M = (\text{Depo} * \text{Drift} * \text{Duct} + \text{Noise}) * \text{Digit}$$

Detector Simulation

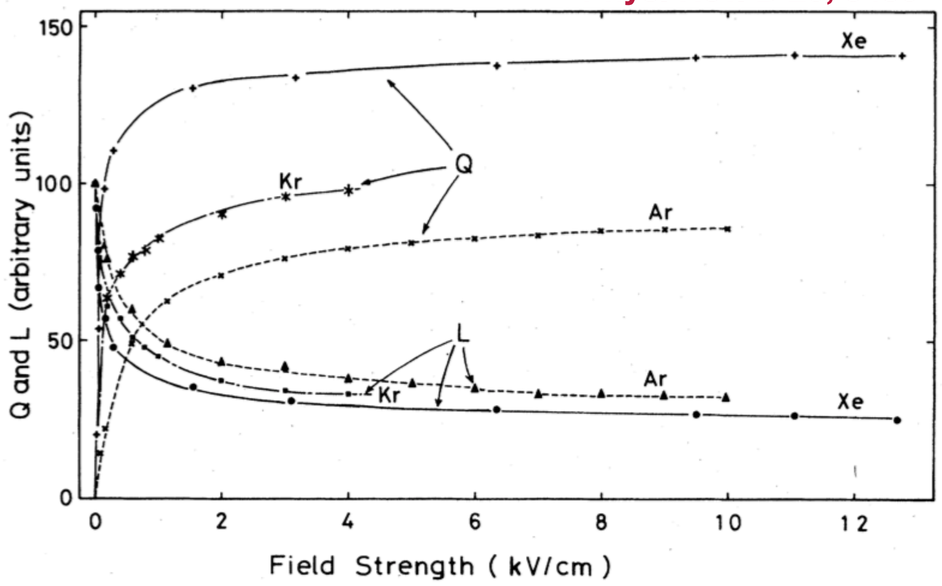
To get to a fully simulated waveform, M, we need five pieces

$$M = (\text{Depo} * \text{Drift} * \text{Duct} + \text{Noise}) * \text{Digit}$$

Depo the energy depositions are what we got from the previous step ✅



Phys. Rev. B 20, 3486



Detector Simulation

To get to a fully simulated waveform, M, we need five pieces

$$M = (\text{Depo} * \text{Drift} * \text{Duct} + \text{Noise}) * \text{Digit}$$

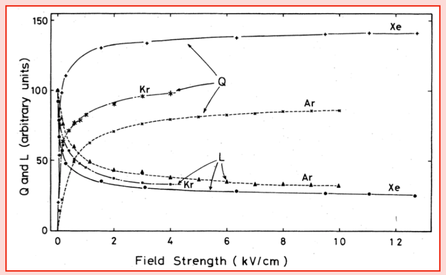
"Drift" not simulated, effects are parameterised and applied!

Electron lifetime + diffusion (longitudinal and transverse) depend on the time electrons spent in the LAr and the electric field

Deposition

GEANT4 A SIMULATION TOOLKIT

Phys. Rev. B 20, 3486



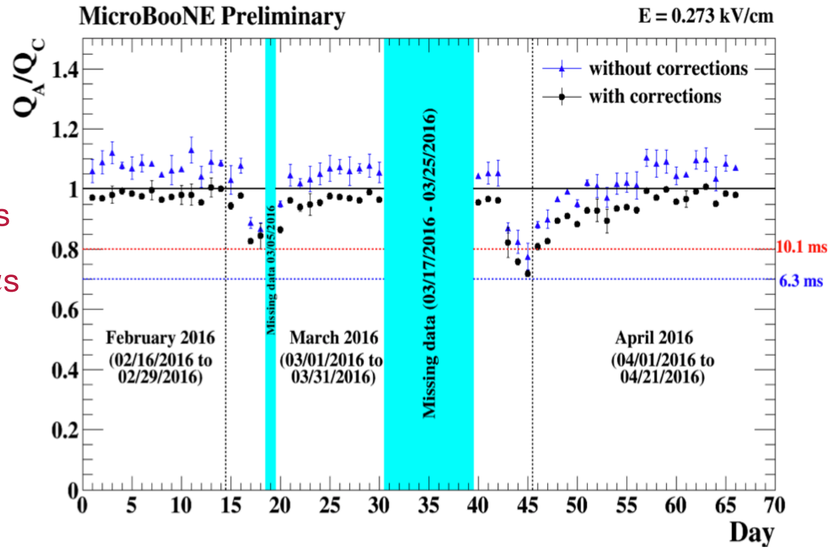
These are fhicl configurable

```
structs: {
   DL: 4
   DT: 8.8
   driftSpeed: 1.565
   lifetime: 35
}
```

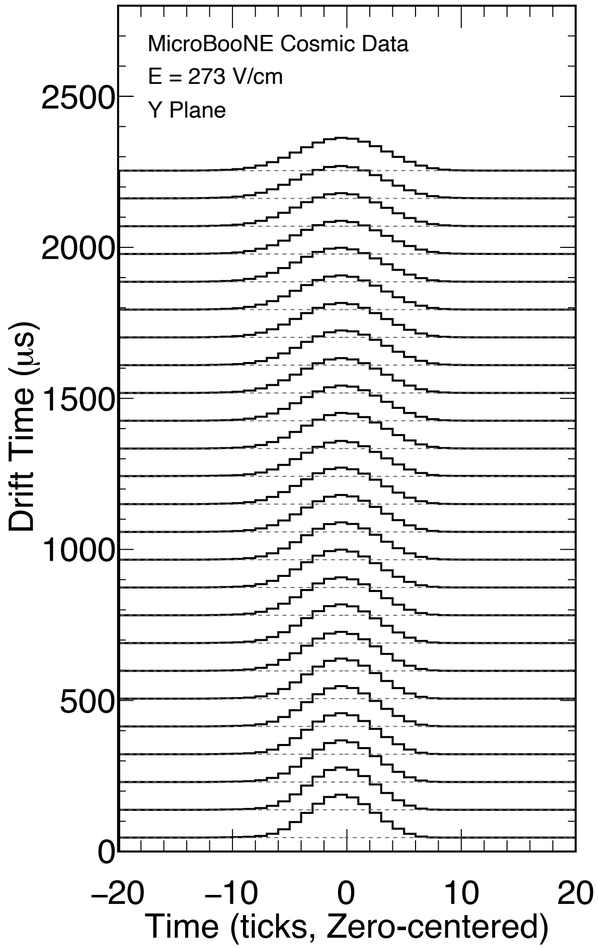
cm^2/s

$mm/\mu s$

ms



JINST 16 P09025



Detector Simulation

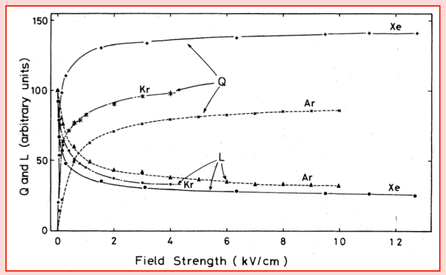
To get to a fully simulated waveform, M, we need five pieces

$$M = (\text{Depo} * \text{Drift} * \text{Duct} + \text{Noise}) * \text{Digit}$$

Deposition

GEANT4
A SIMULATION TOOLKIT

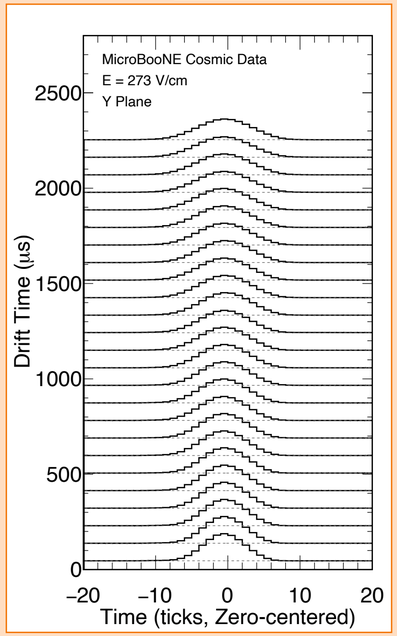
Phys. Rev. B 20, 3486



Drift

Electron lifetime + diffusion

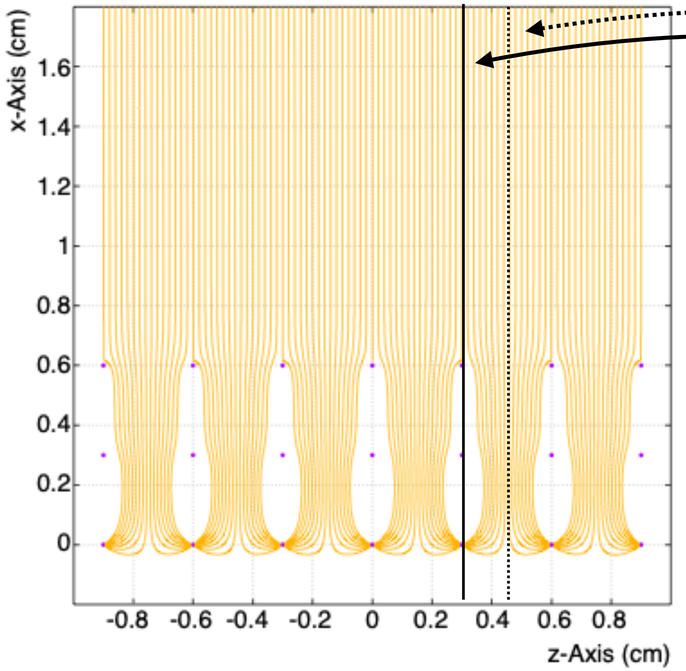
JINST 16 P09025

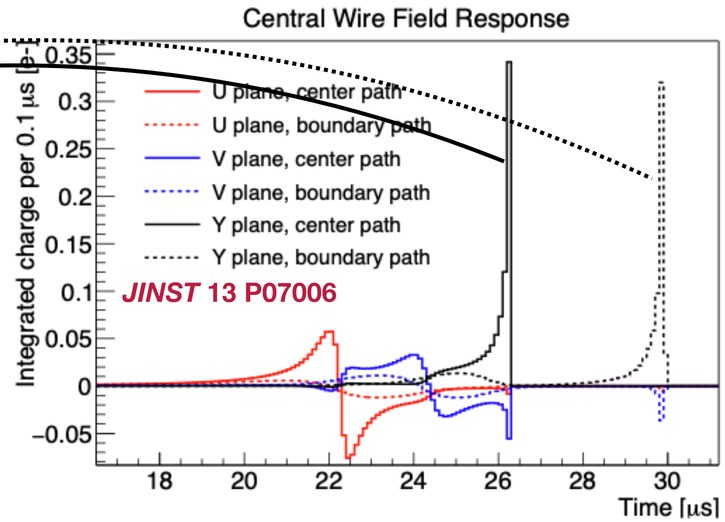


Simulating the Field Response and Electronics Response

"how does a given wire respond to a single electron"

JINST 13 P07006





field is bipolar on induction planes,
unipolar on collection planes

Detector Simulation

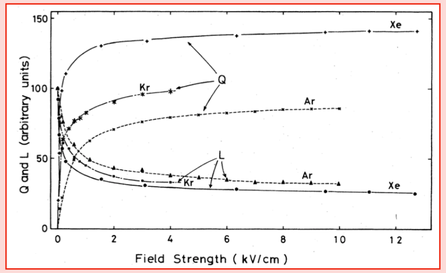
To get to a fully simulated waveform, M, we need five pieces

$$M = (\text{Depo} * \text{Drift} * \text{Duct} + \text{Noise}) * \text{Digit}$$

Deposition

GEANT4 A SIMULATION TOOLKIT

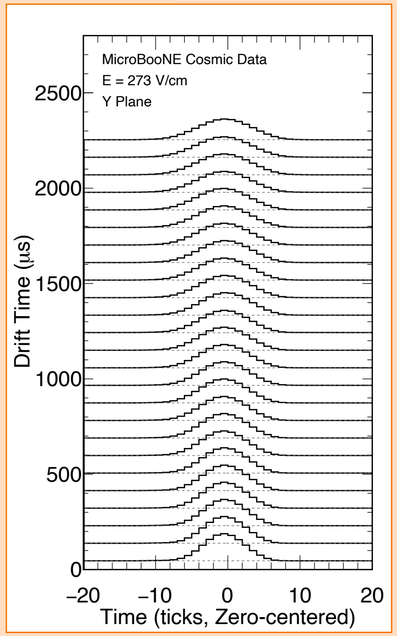
Phys. Rev. B 20, 3486



Drift

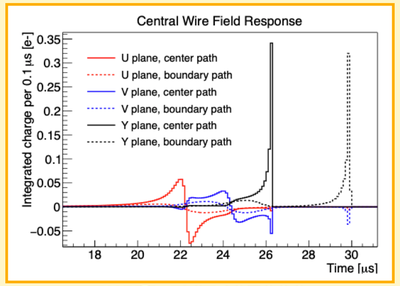
Electron lifetime + diffusion

JINST 16 P09025

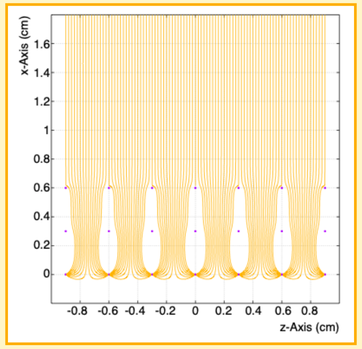


Duct

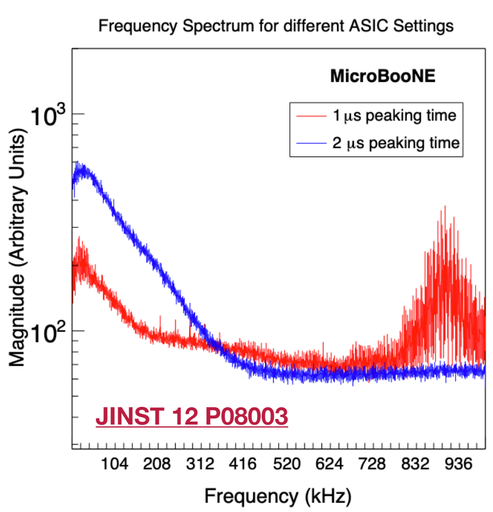
Field/Electronic Response

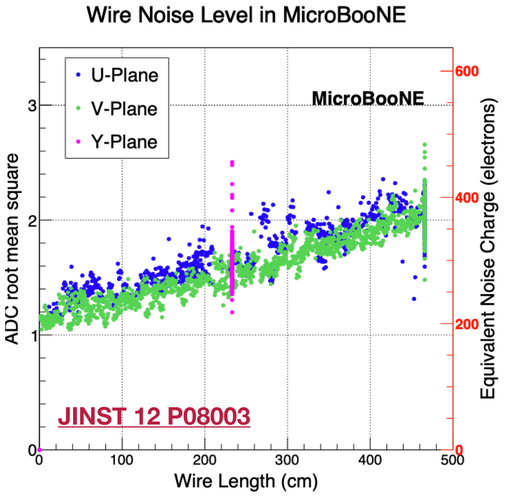


JINST 13 P07006



Noise depends on the specifics of how the electronics are configured and scales as wire length





Detector Simulation

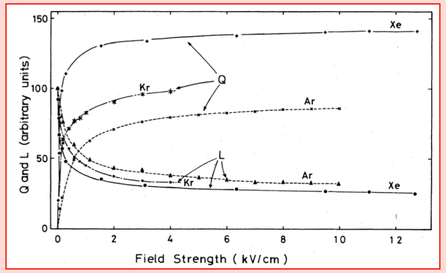
To get to a fully simulated waveform, M, we need five pieces

$$M = (\text{Depo} * \text{Drift} * \text{Duct} + \text{Noise}) * \text{Digit}$$

Deposition

GEANT4
A SIMULATION TOOLKIT

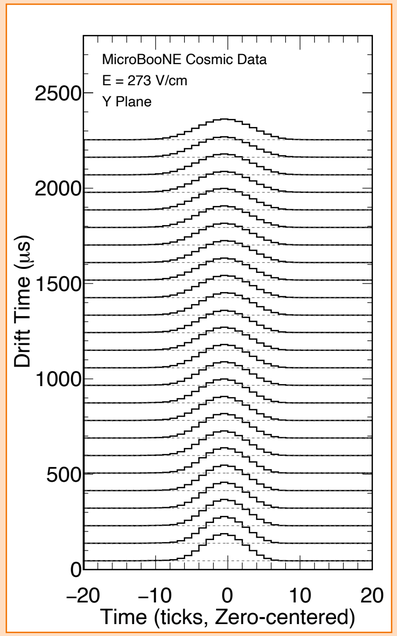
Phys. Rev. B 20, 3486



Drift

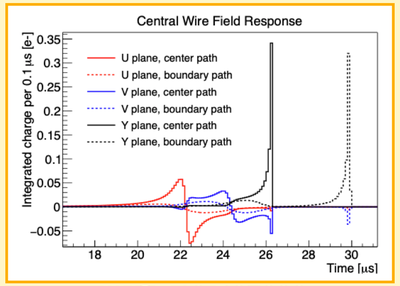
Electron lifetime + diffusion

JINST 16 P09025

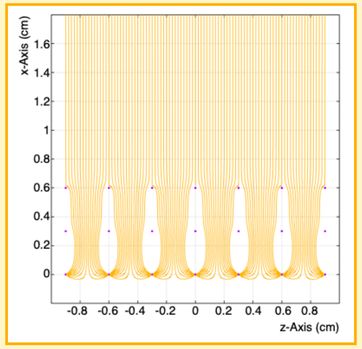


Duct

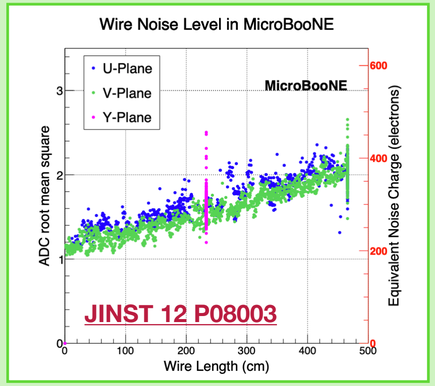
Field/Electronic Response



JINST 13 P07006



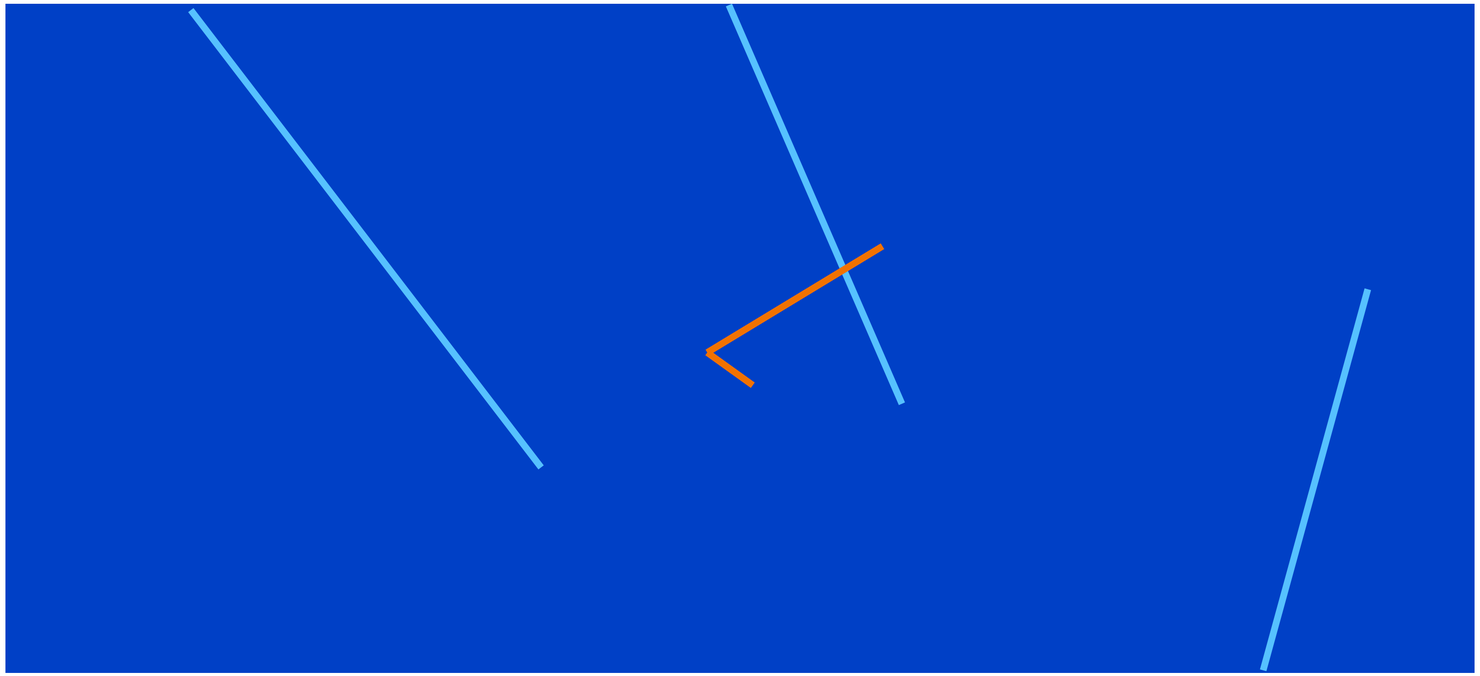
Noise



Digitize

Depends on the number of bits in the ADC, and the sampling rate

Cosmic Data Overlays



Some experiments choose to use “cosmic overlays” rather than simulating their cosmic background
Here, they overlay a **simulated neutrino** on a **data event that only contains cosmic rays**

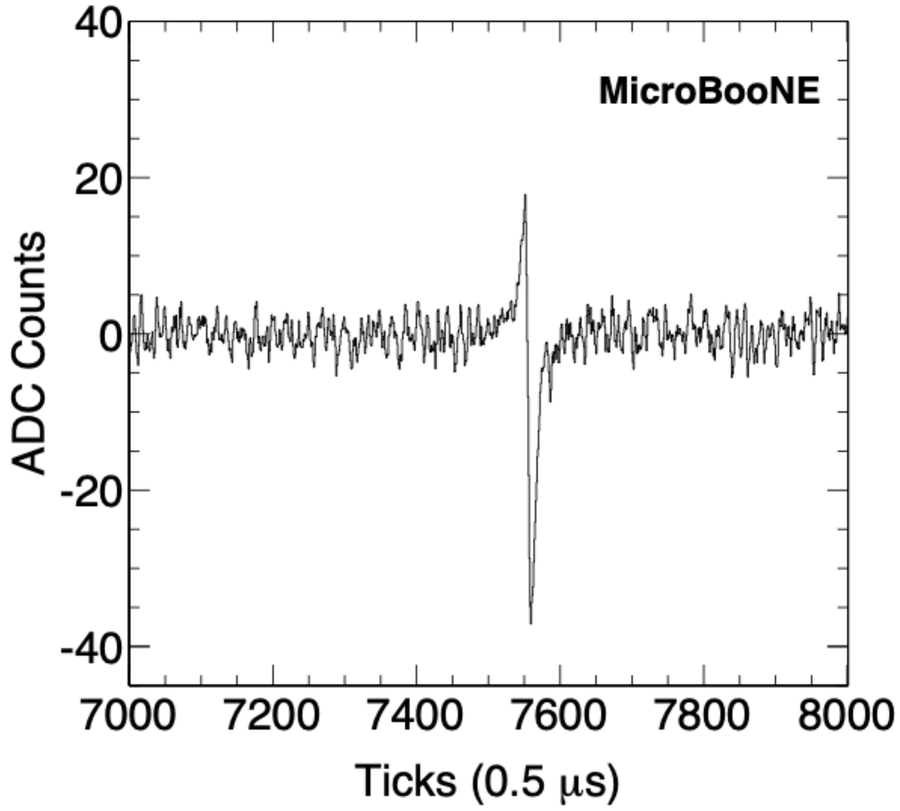
Benefits: free “perfect” noise distribution and distribution of cosmic rays
Challenges: Need a lot of unbiased trigger data, your detector simulation needs to be really good, otherwise ML algorithms could use wrongly simulated detector to pick up on data/mc differences!

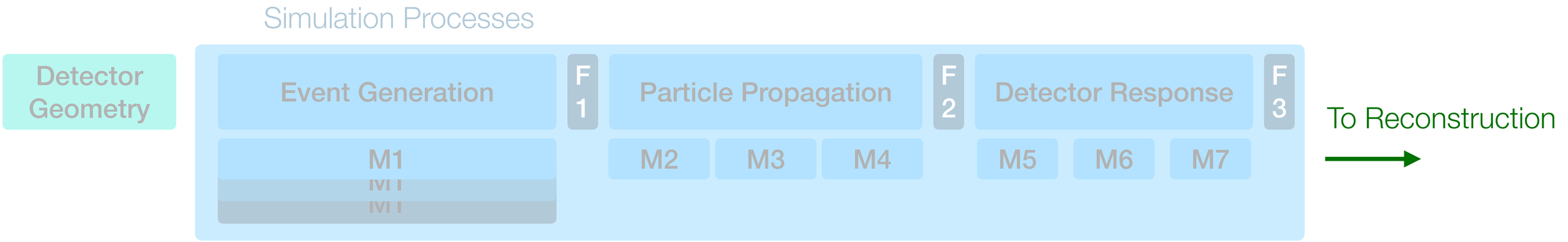
What Do We have After DetSim?

After DetSim we have two new collections of objects, `sim::SimChannels`, and `raw::RawDigits`

`SimChannels` hold all of the true charge/ energy deposited on each channel from simulation, including a vector of `sim::IDEs` that tell you which Geant4 tracks contributed to the deposited energy.

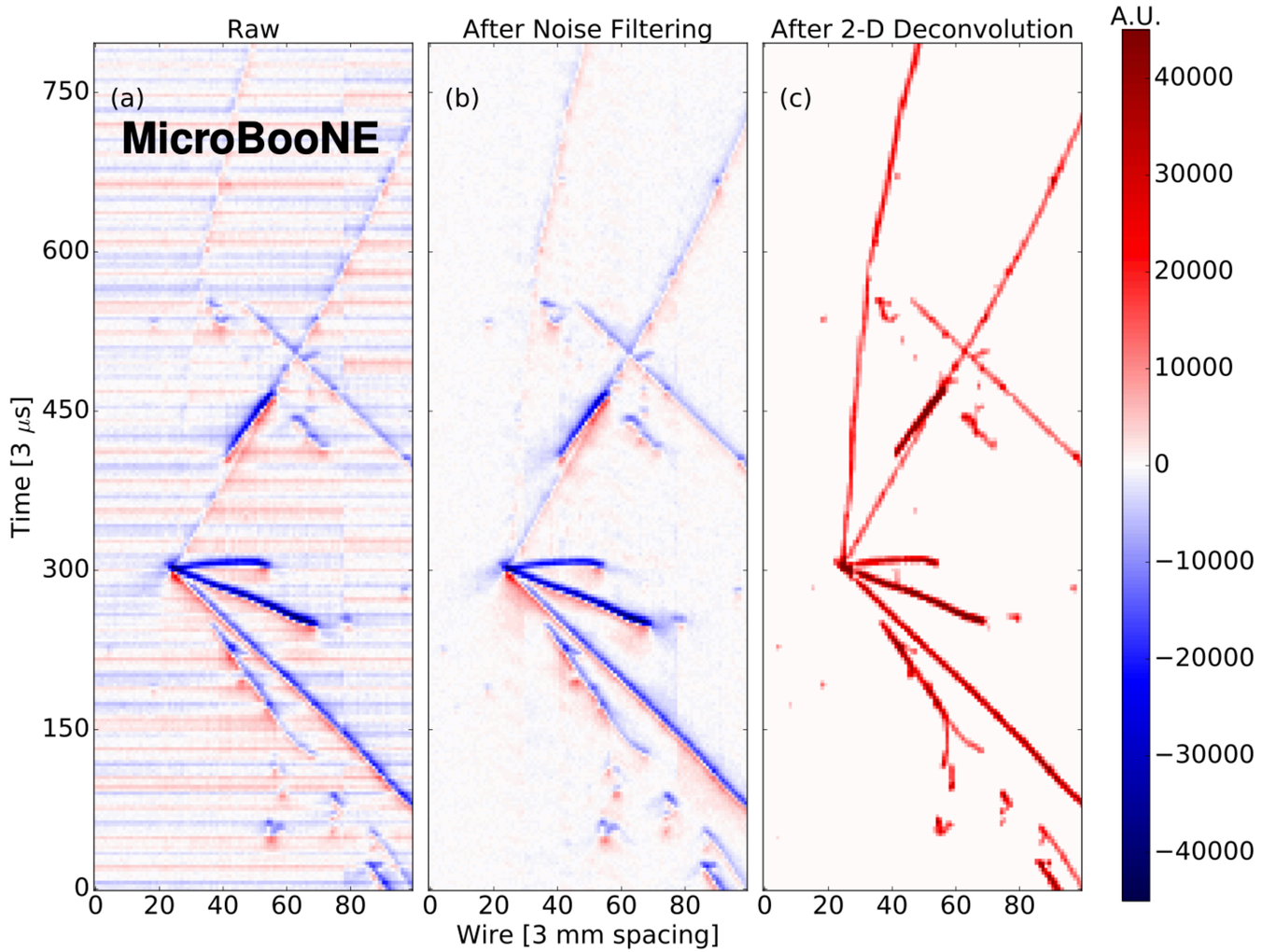
`rawDigits` are the end of our simulation chain, and hold just information we also get from data - effectively just waveforms




Simulation Processes
Detector Geometry
Event Generation
F 1
Particle Propagation
F 2
Detector Response
F 3
M1
M2
M3
M4
M5
M6
M7
To Reconstruction

Signal Processing!

JINST 13 P07007



Wire-Cell

First step of reconstruction

Applies to both simulation and data!
For simulation, this tries to undo all the work we just did so we can see residual effects.

Noise filtering is generally only done on data (we simulate post-noise-filtered noise spectrum)

Deconvolution tries to remove the effect of the electronics and field response,

Summary

- LArSoft is used to build simulations of our experiments, so that we can test our reconstruction and analyses
- Detector geometries and responses differ, but the tools for generation, propagation, and detector simulation are largely shared!
- The software for LAr is constantly evolving…



Summary

Phlex: What Comes After art

Why DUNE is changing frameworks, what Phlex lets you do, and what it means for your code

Gavin S. Davies (U. Mississippi), DUNE Global Software Lead

LArSoft Workshop, 2026

THE UNIVERSITY of MISSISSIPPI

DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

When does this reach you?

Now
art and LArSoft as usual. Physics code is being separated from art
← we are here

2027
Reconstruction running on Phlex, checked against art on ProtoDUNE data

2028
Phlex is the default for new code

2029
ProtoDUNE-III data: a chance to run Phlex on a real detector

2030
First far detector module starts taking physics data

For a PhD starting now
Your thesis will probably use both art and Phlex

9 LArSoft Workshop Gavin S. Davies | Phlex: What Comes After art

THE UNIVERSITY of MISSISSIPPI DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

LArSoft is the tool the field uses now,

Over the next 2-3 years, DUNE will move to PHLEX



LArSoft Cheat Sheet

- lar -c eventdump.fcl <input artroot file>
  - shows you what data products are in an artroot file
- fhicl-dump <input fcl file>
  - prints the entire configuration, -a shows path to where parameter is set
- find_fhicl.sh/findfcl <input fcl file>
  - finds the fcl file in the `$FHICL_FILE_PATH` (see next slide)
- lar --print-available <module/service/tool>
  - prints out all available modules, services, tools

code to find a fhicl file path from a fcl name

```
#!/bin/bash

function usage(){

   echo "INFO  : Usage `basename $0` [fclname]" >&2

}

if [ $# -lt 1 ];then
    usage
    exit 1
fi

THIS_FHICL_NAME=$1

if [ -z $FHICL_FILE_PATH ];then
    echo "ERROR : \$FHICL_FILE_PATH must be set before running" >&2
    exit 1
fi

# We need to split FHICL_FILE_PATH into a list of directories
THIS_FHICL_FILE_PATH=`echo $FHICL_FILE_PATH | tr ":" " "`

# Remove duplicated directory entries
THIS_FHICL_FILE_PATH=`echo $THIS_FHICL_FILE_PATH | tr " " "\n" | sort -u | tr "\n" " "`

#Now turn THIS_FHICL_FILE_PATH into an array
THIS_FHICL_FILE_PATH_ARRAY=( $THIS_FHICL_FILE_PATH )

#Now loop over the directories in fcl file path
for (( i = 0; i < ${#THIS_FHICL_FILE_PATH_ARRAY[@]}; i++ ))
do
    THIS_DIR=${THIS_FHICL_FILE_PATH_ARRAY[$i]}
    if [ ! -e $THIS_DIR ];then
      continue
    fi
    THIS_FIND_QUERY="find  $THIS_DIR -maxdepth 1 -iname '*$THIS_FHICL_NAME*'"
    eval $THIS_FIND_QUERY
done
```