LArSoft event generation and detector simulation

Erica Snider

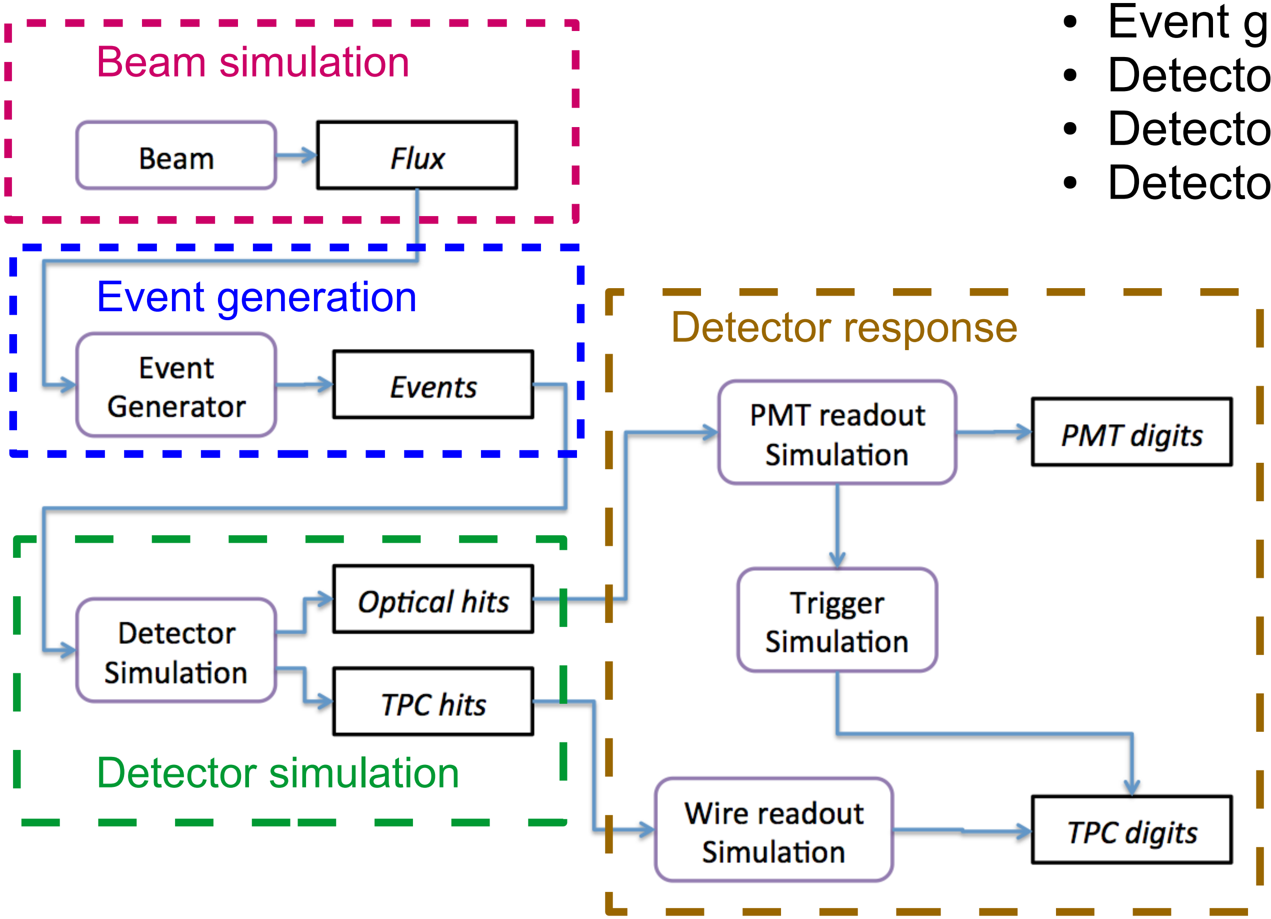
Fermilab

art/LArSoft Course
August 3—7, 2015
Fermilab

Goals for this session

- Summarize the available generation and simulation code
- Introduce how to configure and describe detector to LArSoft
- Introduce detector-specific response functions
- Learn how to use GEANT4 and event generators in LArSoft

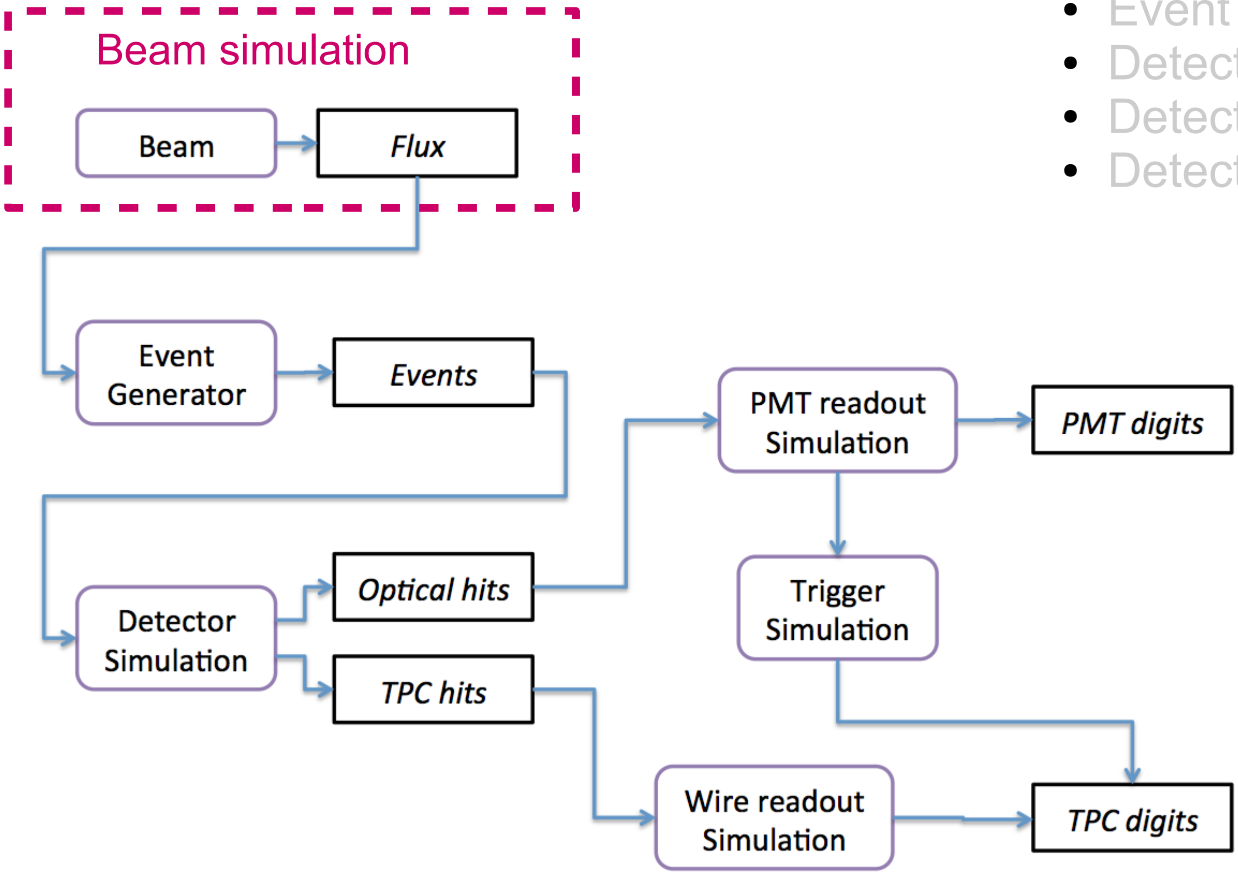
Simulation workflow



Will step through

- Beam simulation
- Event generation
- Detector description
- Detector simulation
- Detector response

From W Seligman

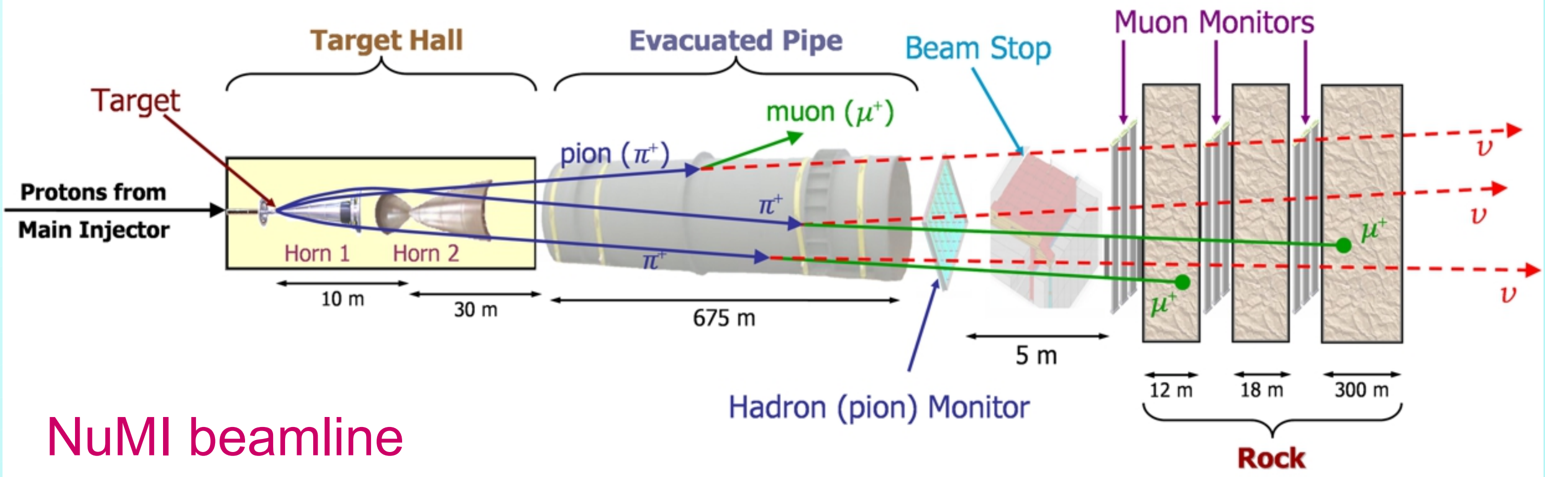


- Beam simulation
- Event generation
- Detector description
- Detector simulation
- Detector response

From W Seligman

Beam simulation

“Typical” neutrino beamline

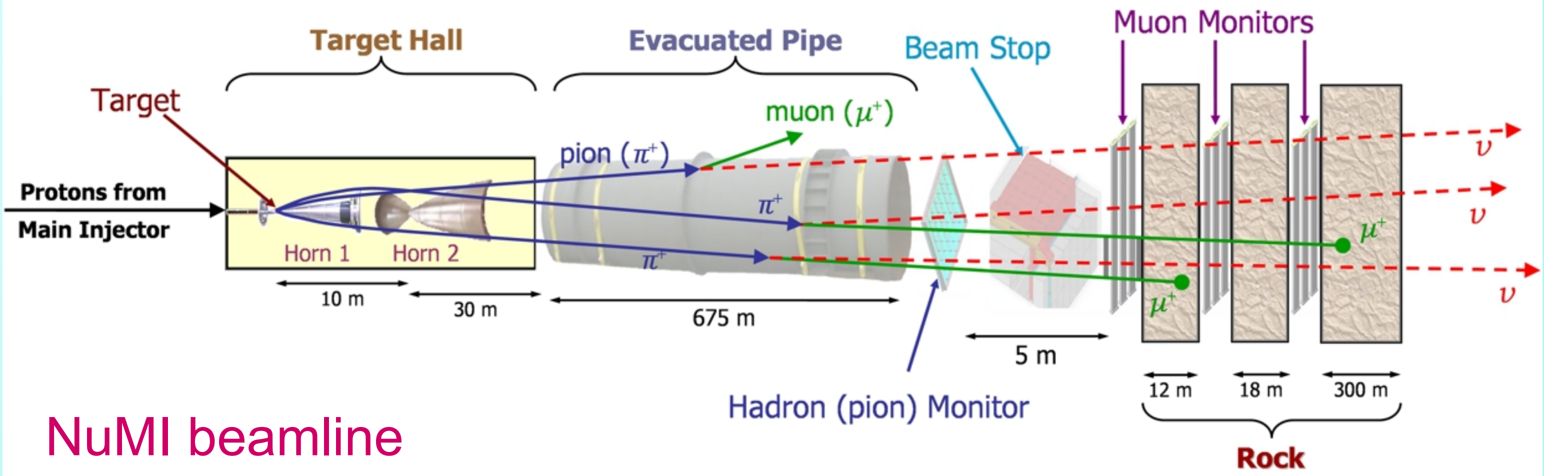


NuMI beamline

- An important (sometimes only!) tool in neutrino flux estimation
- Models
  - Incident proton beam
  - Particle production in the target and surrounding material
  - Focusing horn(s)
  - Hadron and lepton decays with neutrinos in the final state

Beam simulation

“Typical” neutrino beamline



NuMI beamline

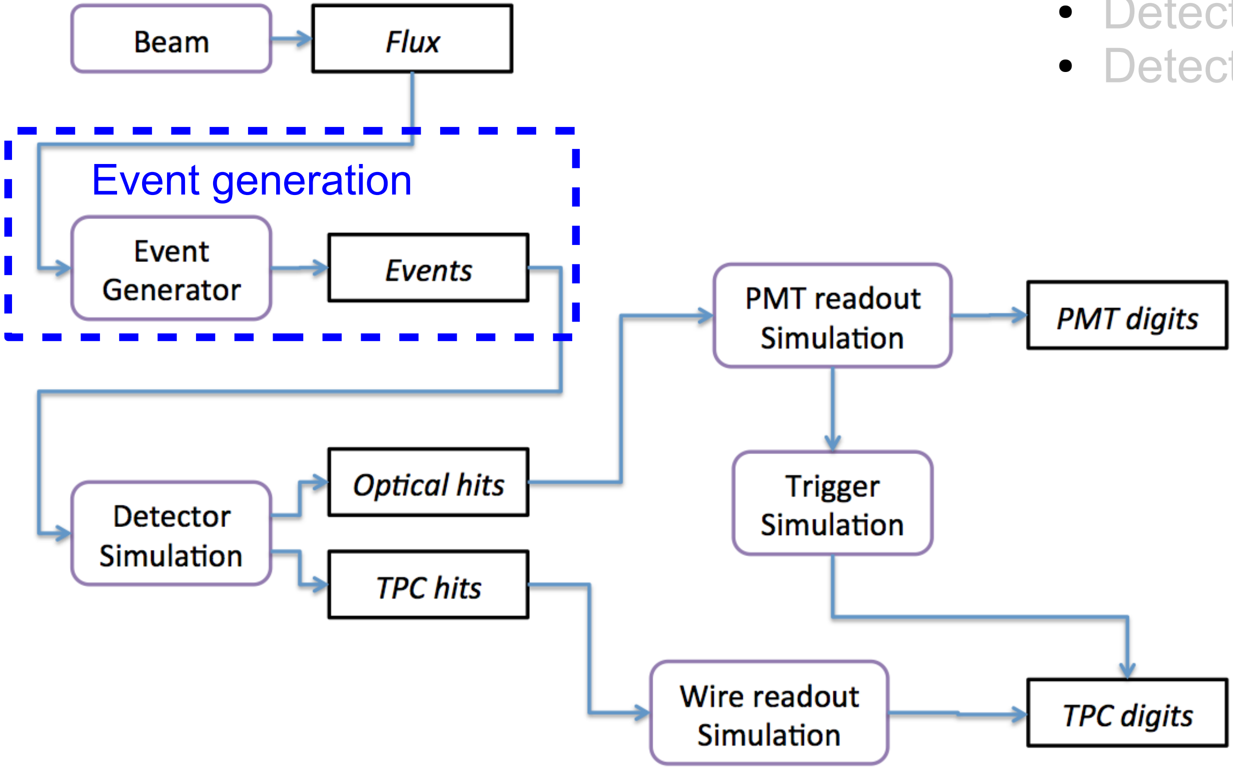
- Produces “flux files” as output
  - Information about collected neutrinos stored in standardized `TTree`
    - LArSoft structure defined in dk2nu product shipped with LArSoft distributions
  - Allows tracking neutrino back through particle decay to production point of ultimate parent hadron
  - Used as input to neutrino event generators
- Typically run as stand-alone step
  - Re-use the flux files in multiple MC production runs
    (ie, mosts people won't need to do this...)

Beam simulation used by LArSoft

- GEANT4-based
  - NUMI: `g4numi`
    - Documented under “NuMi Beam Simulation” redmine project
      - *https://cdcvs.fnal.gov/redmine/projects/numi-beam-sim/wiki*
  - BNB: `BooNEG4Beam` (the old) + `BooNEG4Beam` (the new)
    - Originally written for MiniBooNE, being used by MicroBooNE
    - The old is documented on the uBooNE DocDB. The new is not yet in production.
  - LBNF: ??
- FLUKA has also been used by some experiments

  FLUKA: A. Ferrari, P.R. Sala, A. Fasso`, and J. Ranft, CERN-2005-10 (2005), INFN/TC_05/11, SLAC-R-773
  http://www.fluka.org/fluka.php
  - Not currently interfaced to LArSoft, but may be in the future
- Some accompanying programs may be needed in various cases
  - `BooNEG4NuMI` (old): need to convert MiniBooNE-specific output into dk2nu `TTree`
    - Private code. The new `BooNEG4NuMI` creates `simb::MCTruth` directly
  - `gsimple`: select neutrinos that pass a specified plane in front of a detector
    - Documented under the GENIE redmine wiki page
      - *https://cdcvs.fnal.gov/redmine/projects/genie/wiki/Generating_GSimpleNtpFlux_files*

- Beam simulation
- Event generation
- Detector description
- Detector simulation
- Detector response



From W Seligman

Event generation

- Produces final state particles from a specified physics process
  - Output used as input to detector simulation (or for direct studies)
  - A variety of generator types available / needed in LArSoft
    - Neutrino interaction events
    - Cosmic rays
    - Nucleon decays
    - Single particles
- A number of techniques used to interface to LArSoft
  - Direct: generator is called by LArSoft → creates `simb::MCTruth`
  - Indirect: generator output file read by LArSoft → creates `simb::MCTruth`
  - Embedded: the code lives in LArSoft

All interfaces are in `larsim/EventGenerator` or “nutools”

An aside: the “nutools” ups product

- General purpose tools for neutrino experiments
  - Shared between NOvA, LArTPC experiments (at least)
  - Distributed with LArSoft, set up automatically by LArSoft setup procedure
- Contents:

$NUTOOLS_DIR/source/
    EventDisplayBase/
    EventGeneratorBase/
    G4Base/
    IFDatabase/
    MagneticField/
    NuBeamWeights/
    NuReweight/
    SimulationBase/

Interfaces to GENIE, CRY

Interfaces / services for DB access
to NOvA-style DB
(also used by DUNE / LArIAT)

Data products used by simulation
simb::MCTruth
simb::MCParticle

See https://cdcvs.fnal.gov/redmine/projects/nusoftart/wiki for details

Event generators available to LArSoft

- Neutrino event generators
  - GENIE
    C. Andreopoulos et al., Nucl. Instrum. Meth., A614, pp. 87–104, 2010.
    http://www.genie-mc.org
    - General purpose (ie, not written for a specific experiment)
    - Direct interface: larsim/EventGenerator/GENIE/GENIEGen_module.cc
  - NuWro
    http://borg.ift.uni.wroc.pl/nuwro/
    - General purpose. Different model(s) of final state nuclear interactions, resonance production,...
    - Indirect interface: larsim/EventGenerator/NuWroGen_module.cc
  - NUANCE
    - Originally written for IMB, then adapted to MiniBooNE
    - Indirect interface: larsim/EventGenerator/NUANCEGen_module.cc
    - Source code / libraries are ***not*** distributed with LArSoft

      ...don't know where / when / who uses this at present...

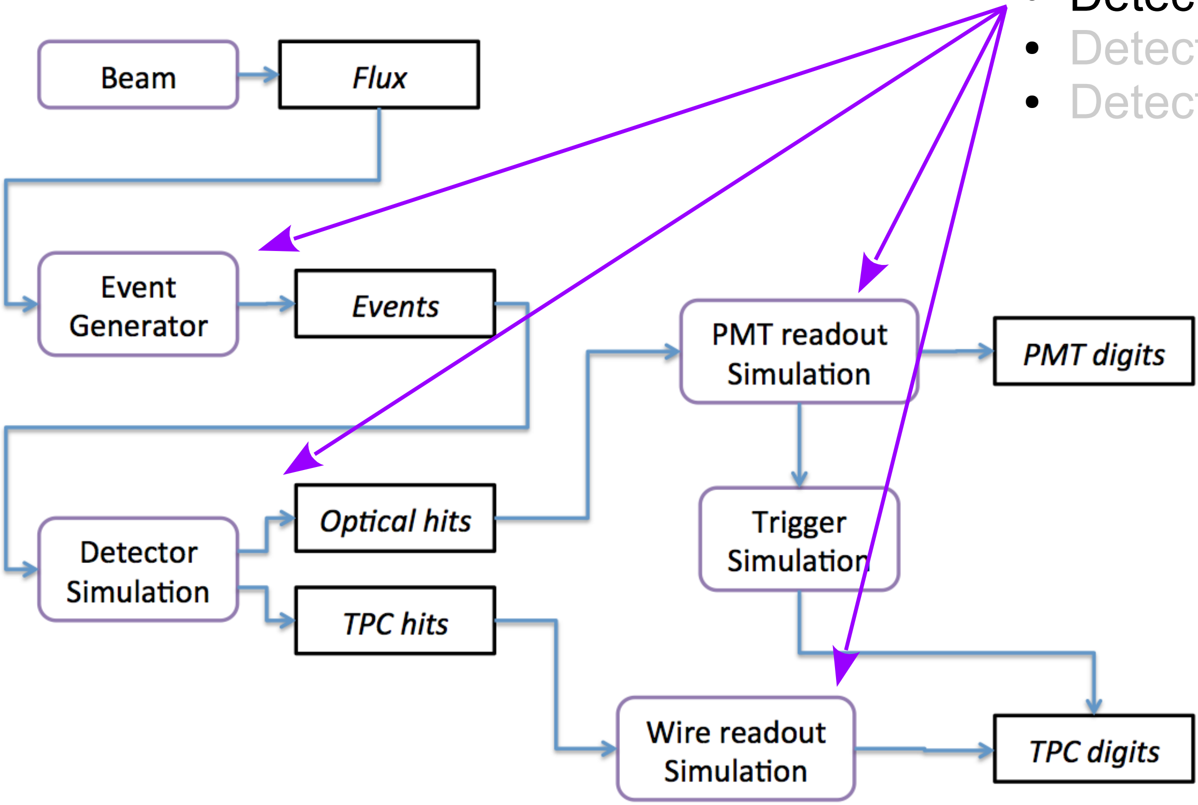
Event generators available to LArSoft

- Cosmic ray generators
  - CRY
    - Lives in the “`cry`” ups product distributed with LArSoft
      - Maintained at LLNL: http://nuclear.llnl.gov/simulation/main.html
    - Direct interface: `larsim/EventGenerator/CRY/CosmicsGen_module.cc`
      - Output directly to `simb::MCTruth`
  - GaisserParam
    - Underground cosmic muons
    - Embedded in LArSoft: `larsim/EventGenerator/GaisserParam/GaisserParam_module.cc`
      - Output directly to `simb::MCTruth`
    - Documented in the module.

Event generators available to LArSoft

- Other generators
  - NDKGen
    - Nucleon decay generator using GENIE
    - Indirect interface: `larsim/EventGenerator/NDKGen_module.cc`
  - SingleGen
    - Single particle event generator. Detector agnostic. (!!)
    - Embedded in LArSoft: `larsim/EventGenerator/SingleGen_module.cc`
    - All prodsingle*.fcl files everywhere use SingleGen
  - TextFileGen
    - Generates an event corresponding to contents of input text file in hepevt format
    - Embedded in LArSoft: `larsim/EventGenerator/TextFileGen_module.cc`
  - RadioGen
    - Models signals generated by radioactive decays in and around LAr
    - Uses decay tables, TGraphs of relevant spectra stored in `larsoft_data` product
    - Embedded in LArSoft: `larsim/EventGenerator/RadioGen_module.cc`

- Beam simulation
- Event generation
- Detector description
- Detector simulation
- Detector response

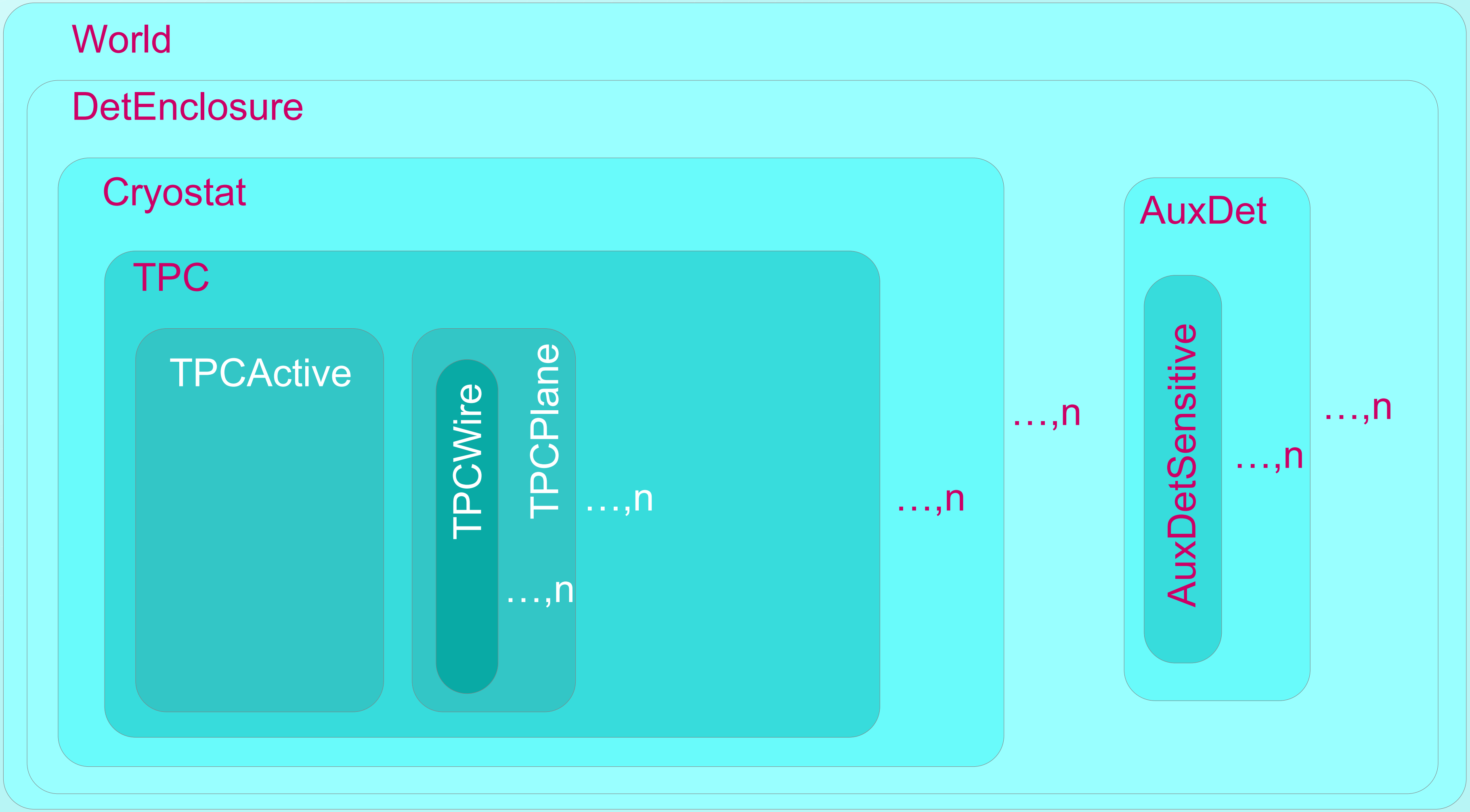


From W Seligman

Detector geometry

Hierarchy of geometry volumes

- Material associated with most of these

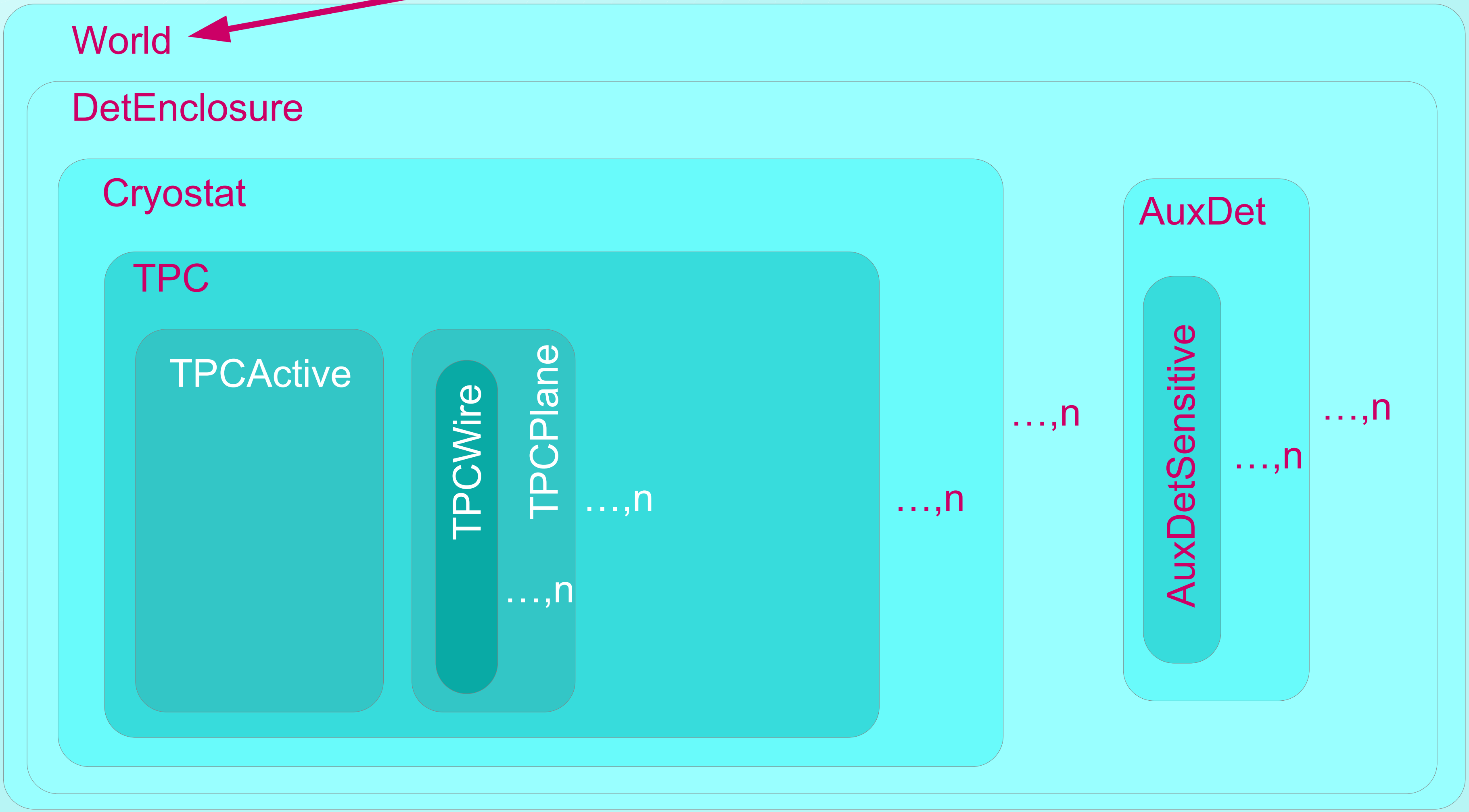


Detector geometry

Hierarchy of geometry volumes

- Material associated with most of these

E.g., buildings, overburden, surrounding dirt / rock

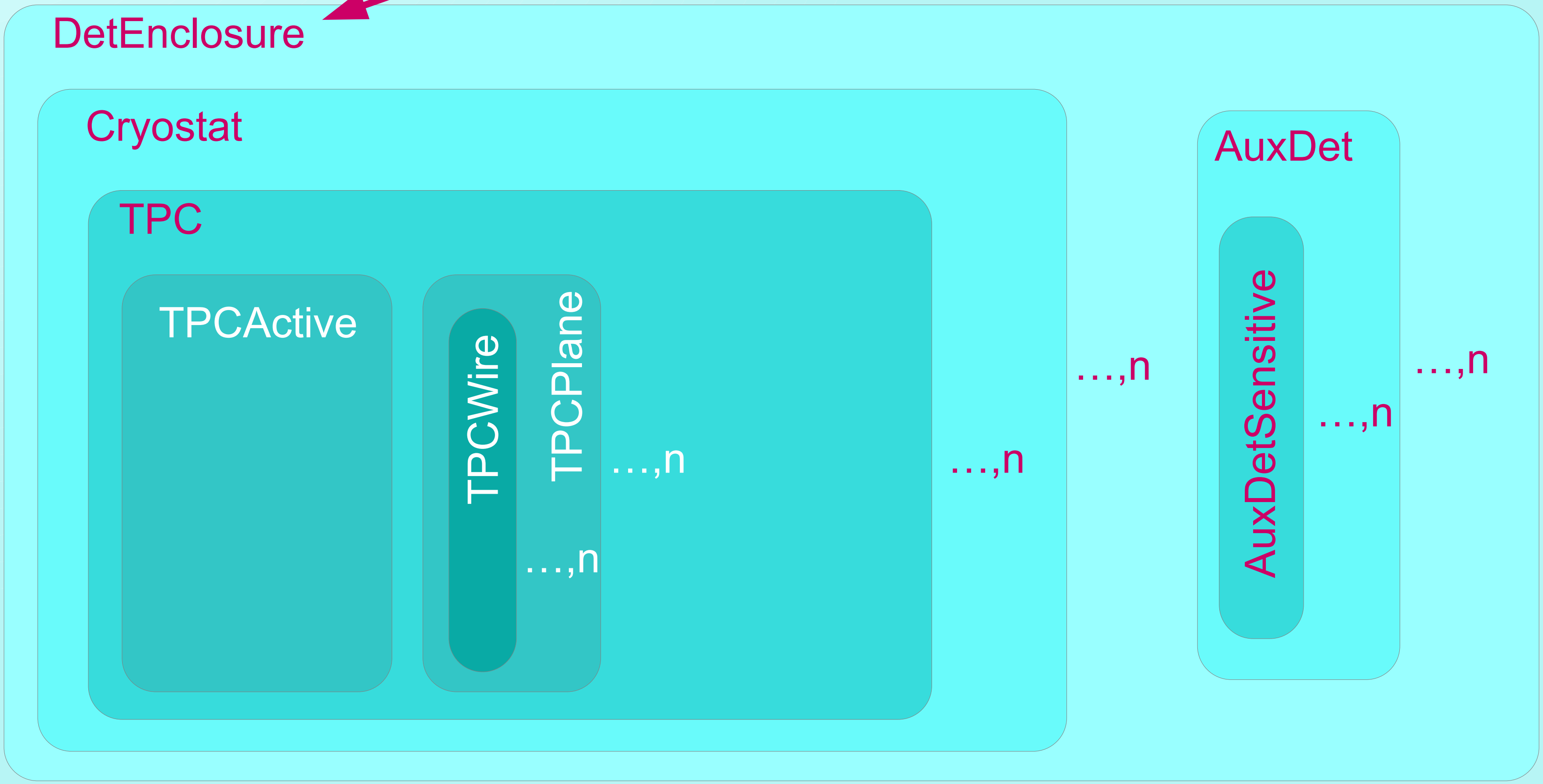


Detector geometry

Hierarchy of geometry volumes

- Material associated with most of these

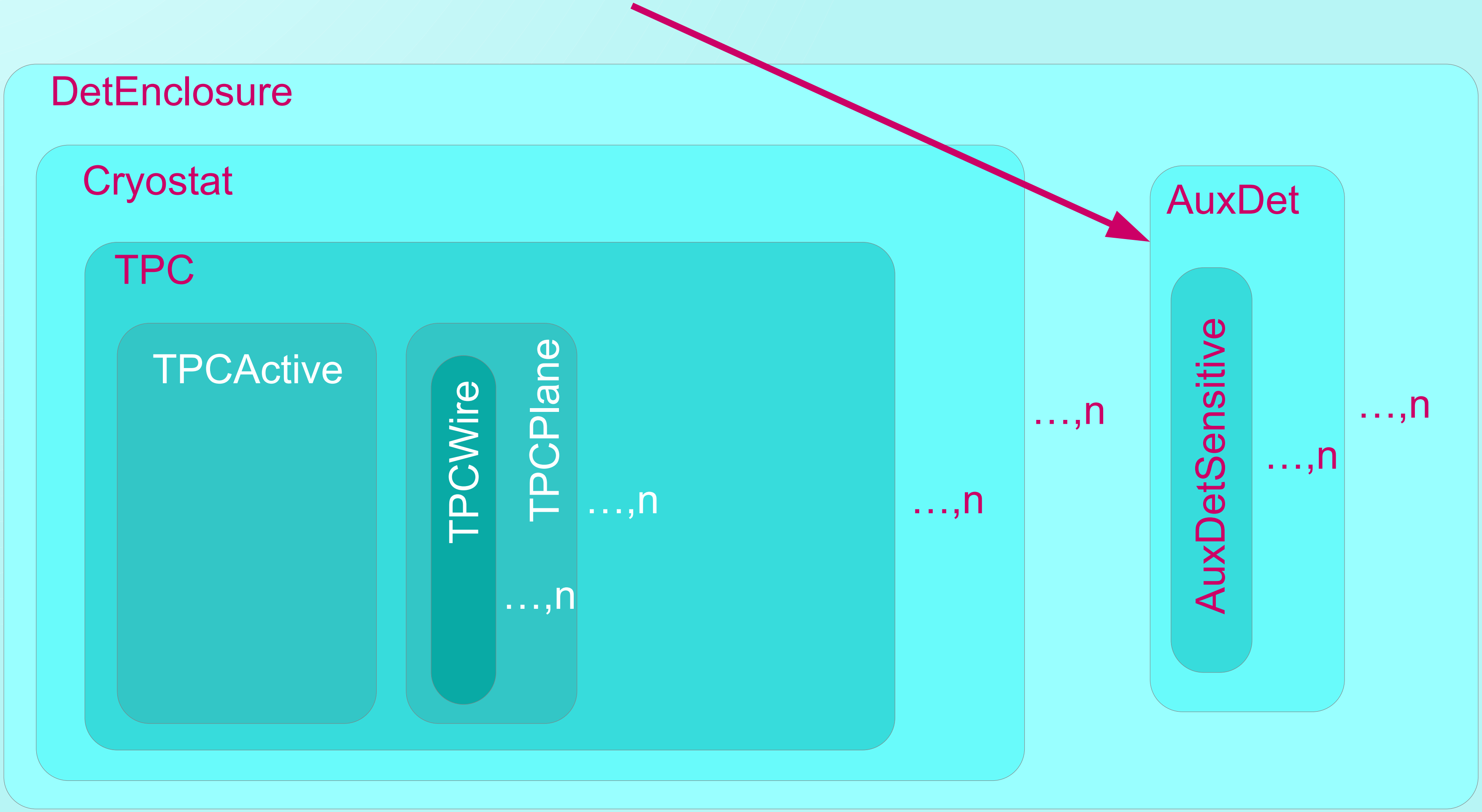
E.g., containment structure for cryostat, front end boards, etc.



Detector geometry

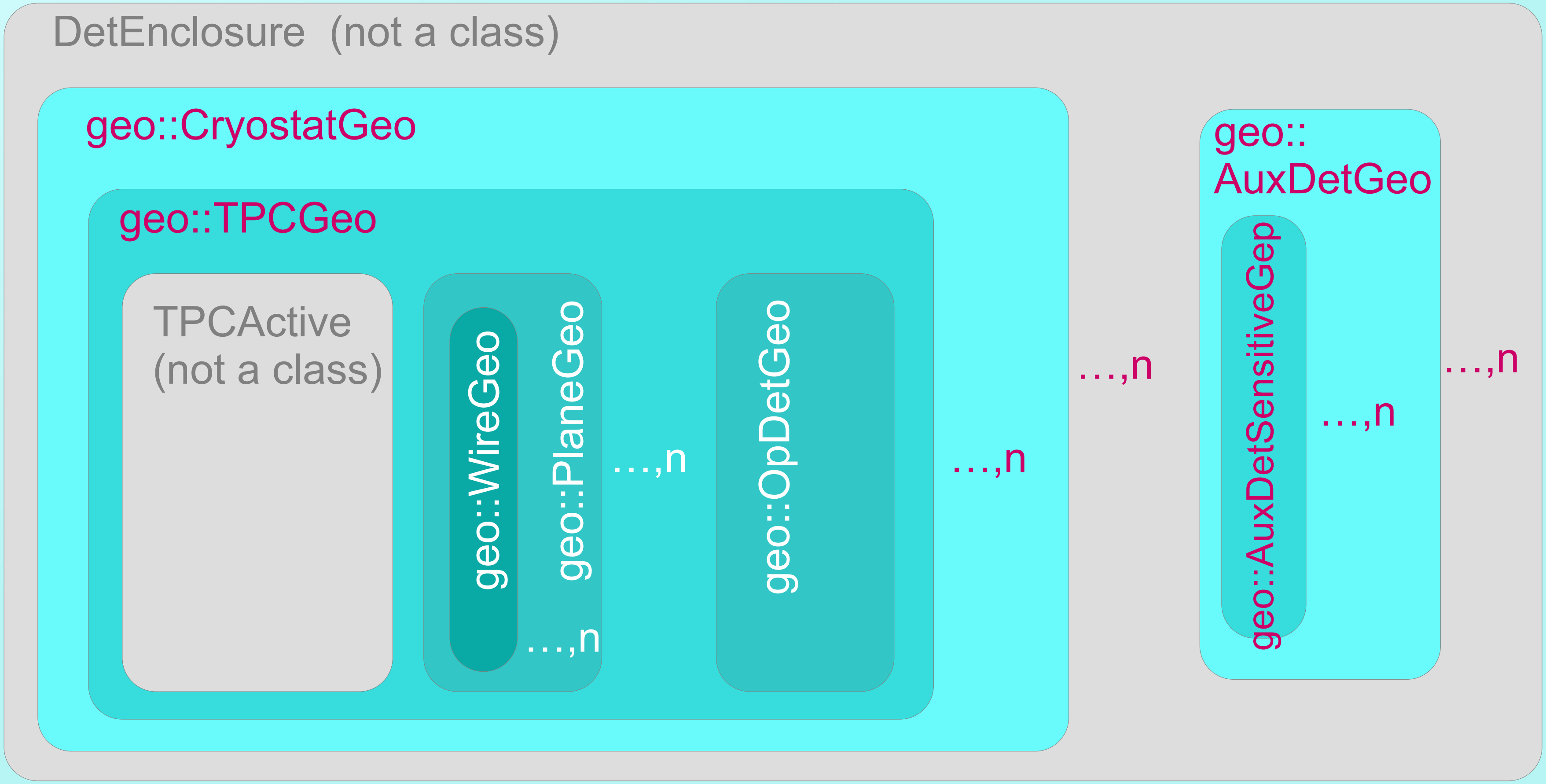
Auxiliary detectors

- For detector elements outside the cryostat
- E.g., cosmic ray counters around cryostat, upstream detectors in a test beam, etc.



Detector geometry

Class structure reflects this hierarchy



All geometry classes in larcore/Geometry

DetEnclosure (not a class)

geo::CryostatGeo

geo::TPCGeo

TPCActive (not a class)

geo::WireGeo

…,n

geo::PlaneGeo

…,n

geo::OpDetGeo

…,n

…,n

geo:: AuxDetGeo

geo::AuxDetSensitiveGep

…,n

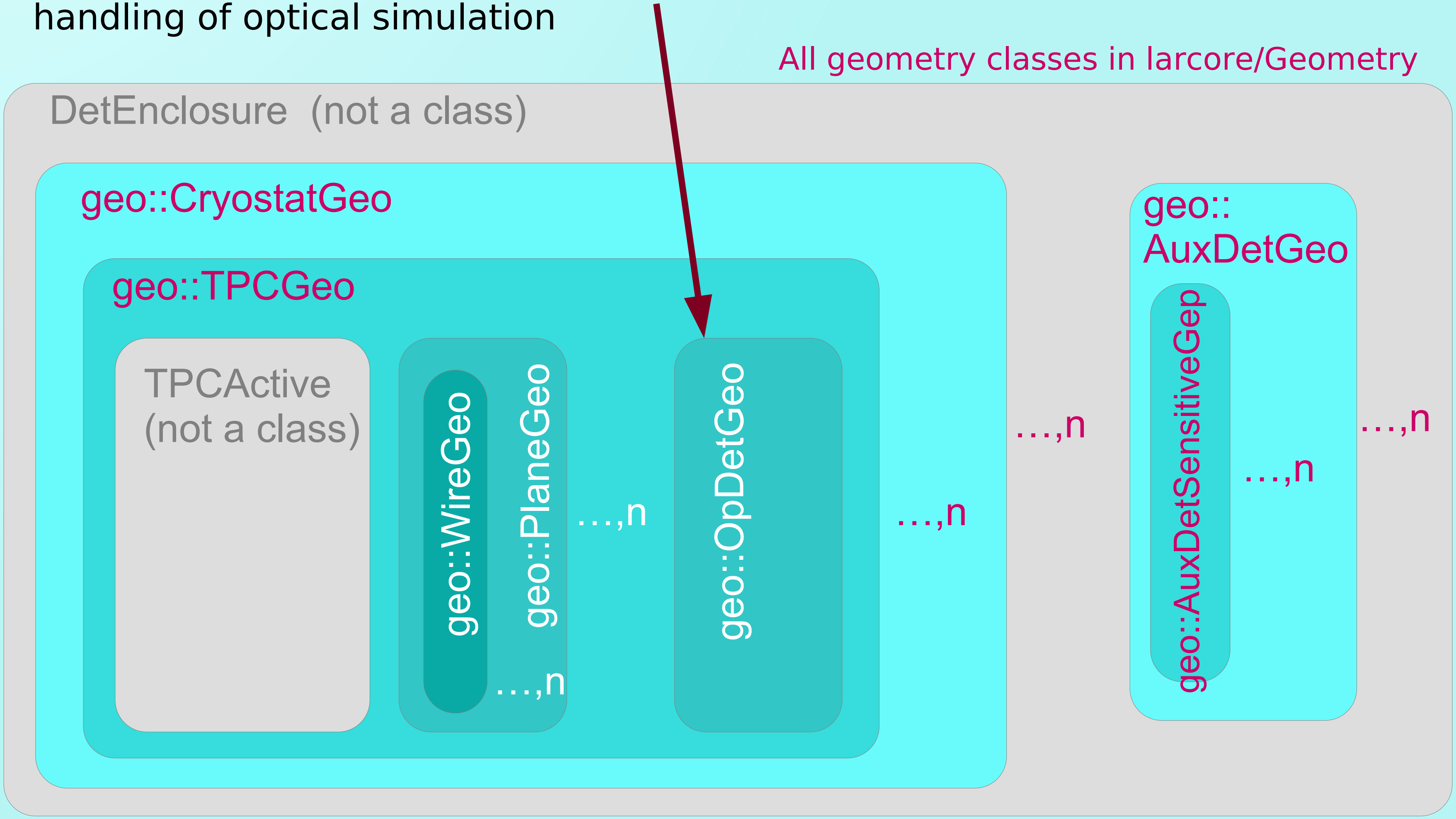
…,n

Detector geometry

Class structure reflects this hierarchy

- Adds “parallel” volume `geo::OpDetGeo` to represent / separate handling of optical simulation

All geometry classes in larcore/Geometry

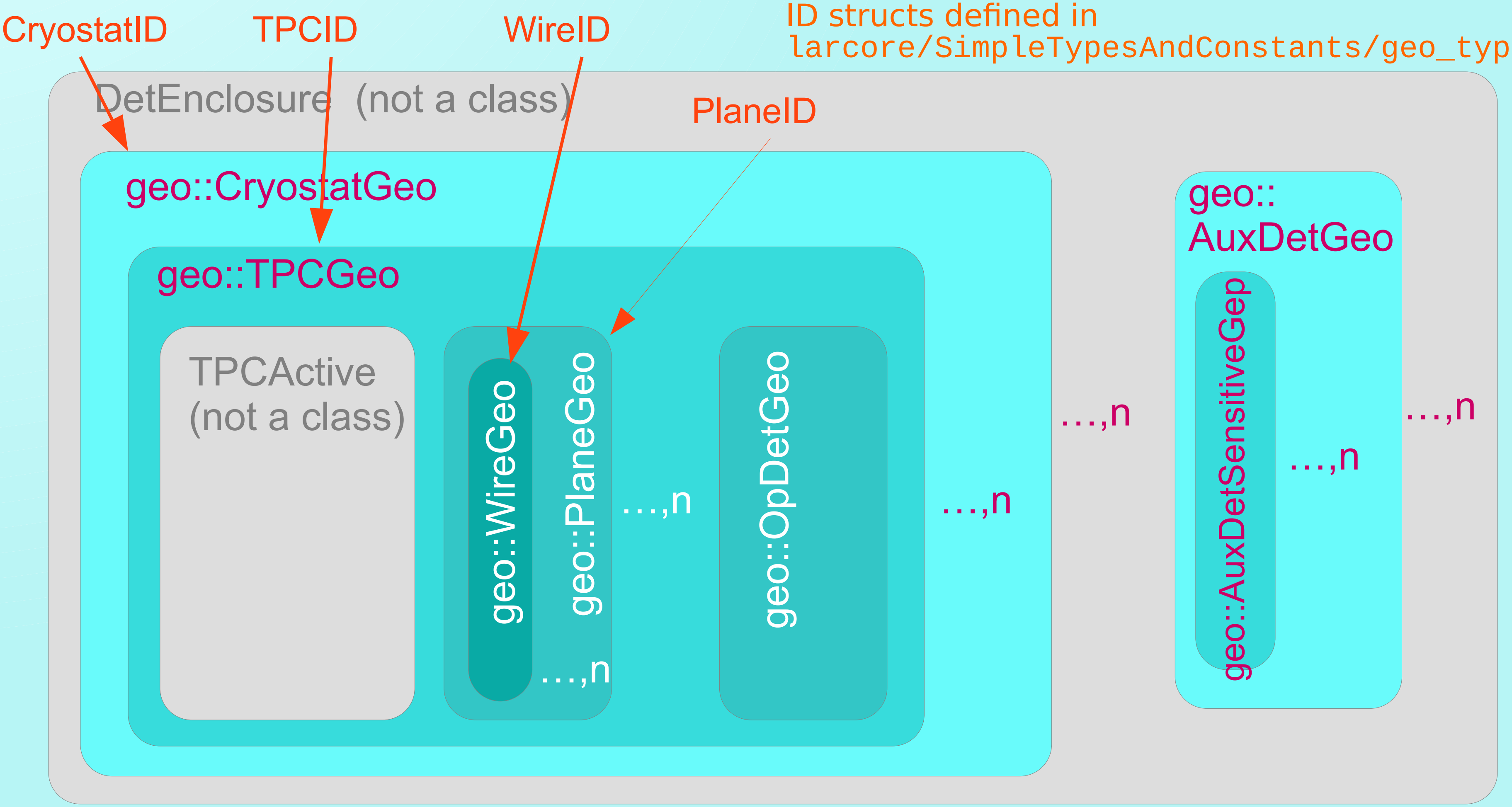


Detector geometry

Access the geometry classes via `Geometry` service

- Use ID objects to specify which instance of TPC geometry objects you want

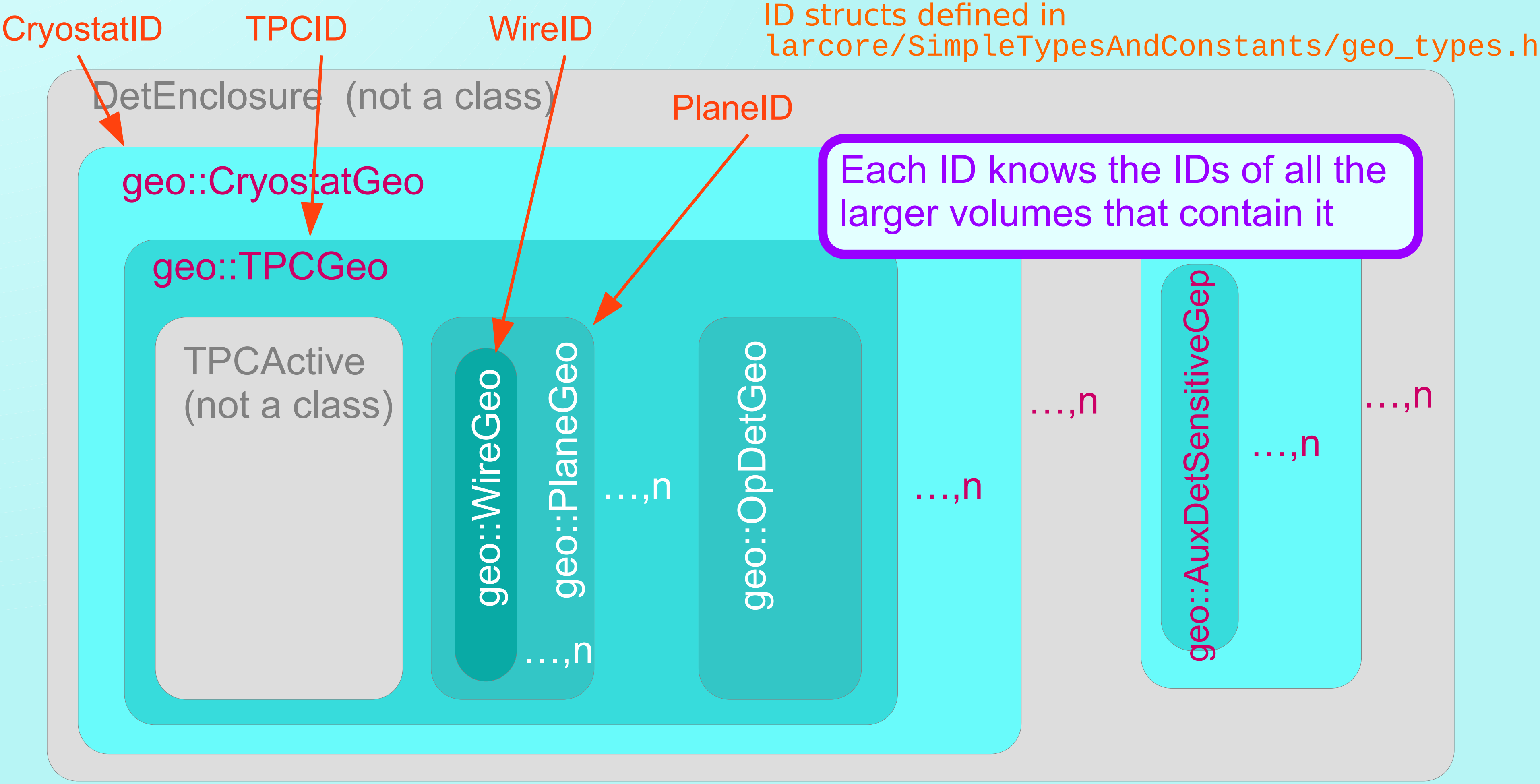
ID structs defined in
`larcore/SimpleTypesAndConstants/geo_types.h`



Detector geometry

Access the geometry classes via `Geometry` service

- Use ID objects to specify which instance of TPC geometry objects you want



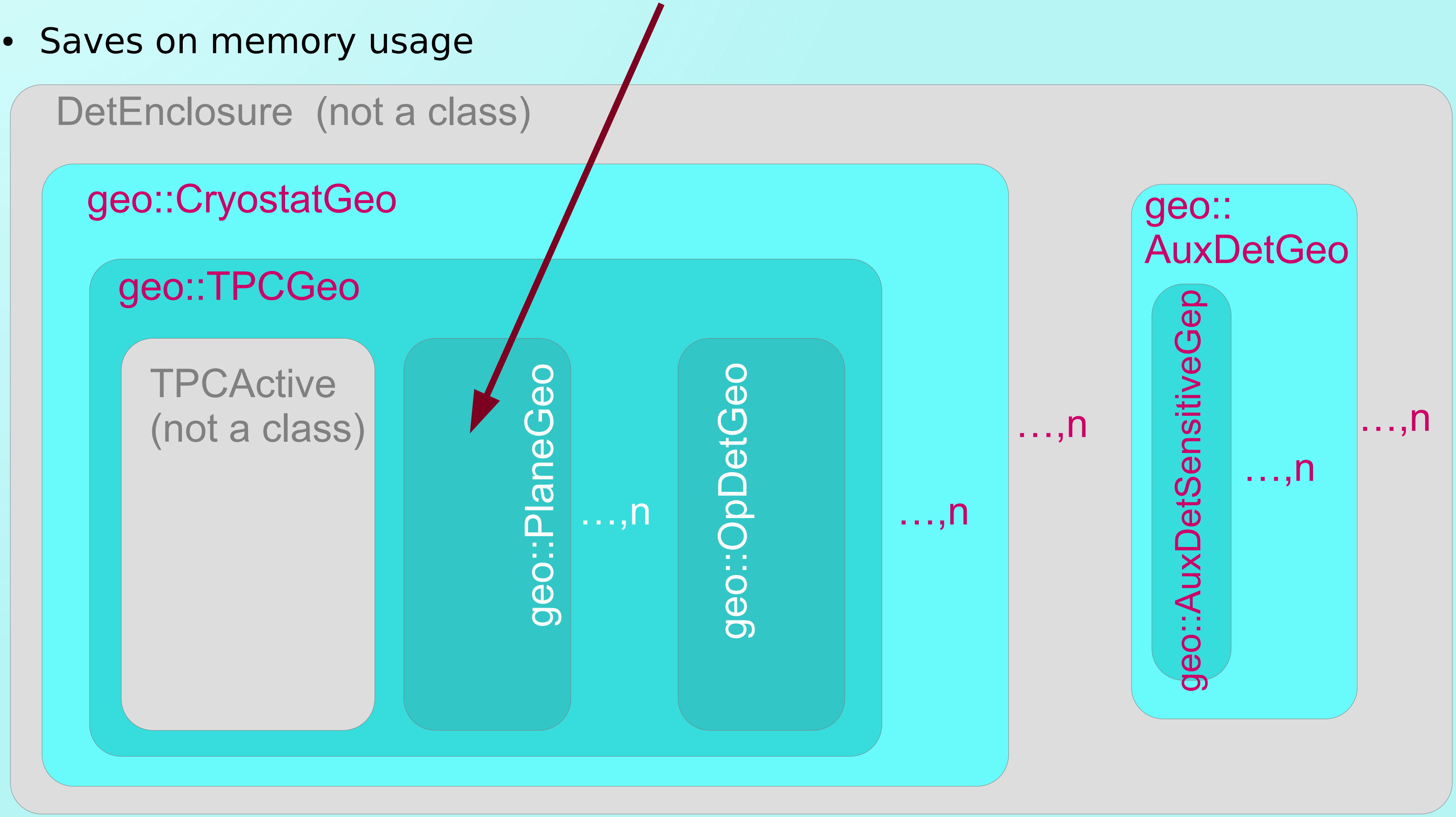
ID structs defined in
`larcore/SimpleTypesAndConstants/geo_types.h`

Each ID knows the IDs of all the larger volumes that contain it

Detector geometry

Don't usually need the wire volumes to be defined

- Often define “`nowires`” geometries that exclude the wire volumes, otherwise identical
- Saves on memory usage



Detector geometry specification

- Geometry specified with Geometry Description Markup Language (GDML)

  http://gdml.web.cern.ch/GDML/

  - A declarative geometry language understood by `root` and `GEANT4`
  - The official GDML files currently live in experiment repositories
    - `<expt repo>/Geometry/gdml/*.gdml,`

      `<expt repo>/Geo/gdml/*.gdml, etc`
  - The GDML flles are not necessarily the “master” geometry description
    - Most experiments have written Perl or C programs to write the actual GDML files
    - DUNE plans to use a DB to construct the GDML
  - The main point:
    - There should be **exactly one** authoritative source for geometry information
    - Must use GDML file to tell LArSoft about it

Accessing detector information

- `geo::Geometry` (in `larcore/Geometry/Geometry.h`)
  - An art service that provides common interface to detector geometry
  - Access geometry via the `geo:GeometryCore` interface
  - Several methods to access geometry classes
    - By element ID, for example:

```
PlaneGeo const & Plane( geo::PlaneID id ) const;
PlaneGeo const & GetElement( geo::PlaneID id ) const;
PlaneGeo const * PlanePtr( geo::PlaneID & id ) const;
PlaneGeo const * GetElementPtr( geo::PlaneID id ) const;
```

    - Iteration over all IDs of a given type, for example:

```
void GetBeginID( geo::TPCID & id ) const;          void GetEndID( geo::TPCID & id ) const;
std::set<PlaneID> const & PlaneIDs() const;
plane_id_iterator begin_plan_id() const;           plane_id_iterator end_plane_id() const;
plane_iterator begin_plane() const;                plane_iterator end_plane() const;
```

These methods / iterators allow iteration over geometry / readout units without nested loops, needing to know how many of what is there

The above methods available for each level in geometry class structure

Accessing detector information

- `geo::ChannelMapAlg`
  - Provides the mapping between geometry objects and readout channels
    - Refer to geometry objects using the geometry ID structs already introduced
    - Refer to TPC DAQ channels using raw::ChannelID_t
      - Defined in `larcore/SimpleTypesAndConstants/RawTypes.h`
    - Concept of optical channel IDs also
  - ChannelMap available from the `Geometry` service

```
ChannelMapAlg const * cma = geom->ChannelMap();
```

  - Detector-specific, so `ChannelMapAlg` is an abstract class
    - Detector-specific implementation is provided via `ExptGeoHelperInterface` service, which itself is an abstract class for the service.
      - Uses the art feature “service interface class”: specifies an interface for a service
      - Allows run-time selection of art service implementation in a fcl file
      - Helper service not exposed to users of Geometry, but geometry authors need to know

Accessing detector information

- `geo::ChannelMapAlg`
  - Provides the mapping between geometry objects and readout channels
    - Refer to geometry objects using the geometry ID structs already introduced
    - Refer to TPC DAQ channels using raw::ChannelID_t
      - Defined in `larcore/SimpleTypesAndConstants/RawTypes.h`
    - Concept of optical channel IDs also
  - ChannelMap available from the `Geometry` service

```
ChannelMapAlg const * cma = geom->ChannelMap();
```

  - Detector-specific, so `ChannelMapAlg` is an abstract class
    - Detector-specific implementation is provided via `ExptGeoHelperInterface`

The use of service interface classes is a common pattern throughout LArSoft
Allows experiment / detector-specific implementations for shared services

Look for “service_provider” line in fcl files

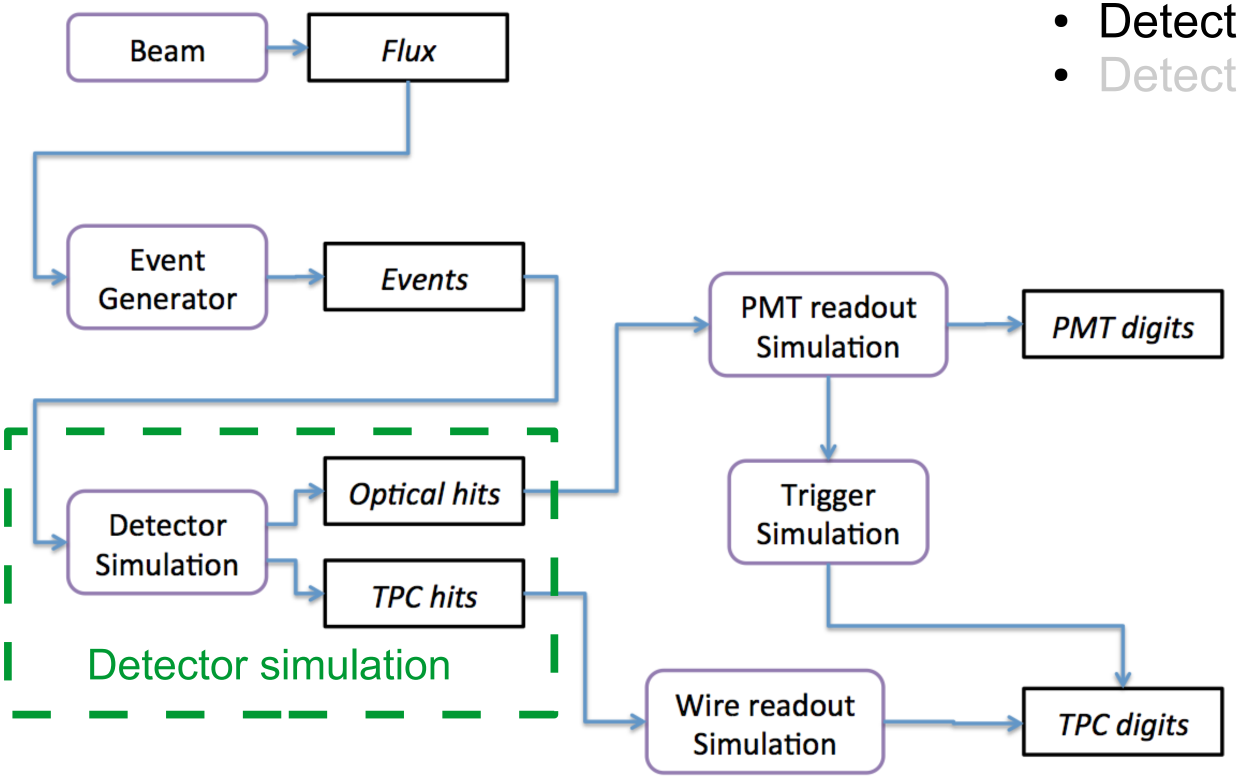
Accessing detector information

- `util::DetectorProperties`
  - An art service that provides access to information about the detector
    - Sampling rate, number of time samples per drift window
    - Trigger time offset
    - Time offsets between wire planes
    - ADC tick to distance conversion
    - etc.
  - Data currently filled from input fcl files
  - Working to allow use of a database to fill values
  - Lives in `lardata/Utilities`

Accessing detector information

- `util::LArProperties`
  - An art service that provides access to liquid argon properties
    - Density
    - Drift velocity
    - Birks correction
    - Electric field
    - Temperature
    - Electron lifetime
    - etc.
  - Also currently filled from fcl input
  - And also currently working to allow use of a database to fill values
  - Lives in `lardata/Utilities`

- Beam simulation
- Event generation
- Detector description
- Detector simulation
- Detector response

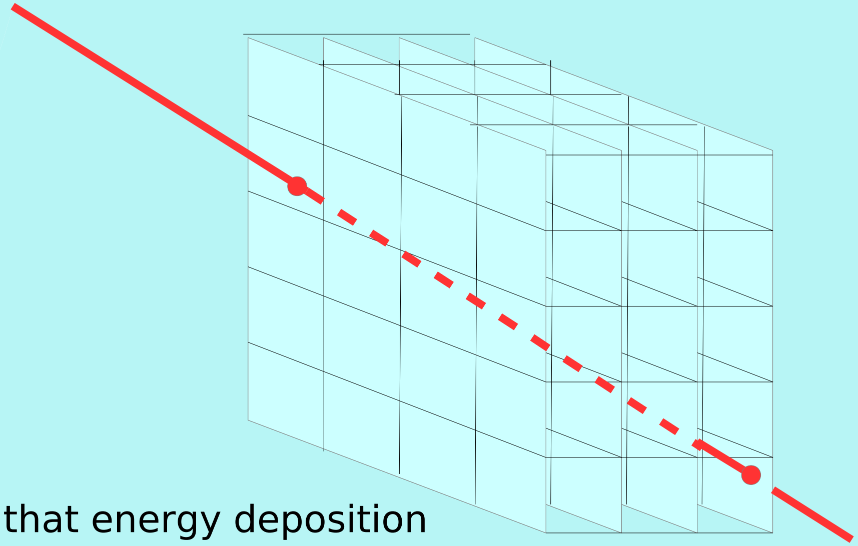


From W Seligman

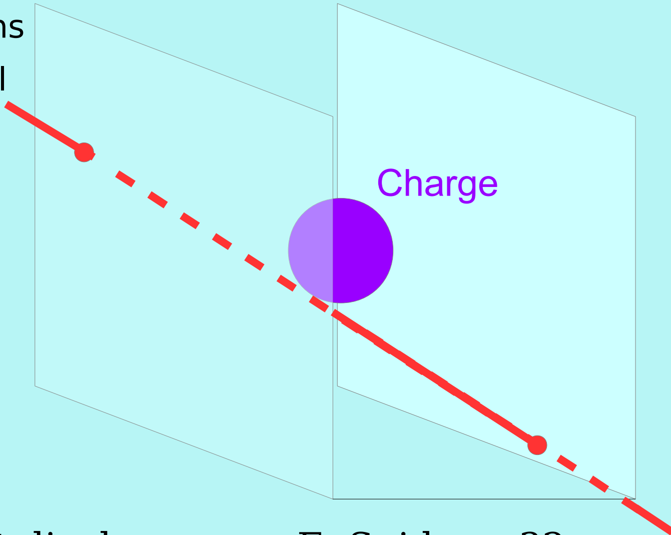
Detector simulation

- The `larg4::LArG4` module: `GEANT4` interface
  - Configures `GEANT4`
    - Passes GDML file and material properties to `GEANT4`
    - Creates “parallel worlds” for wire, optical, and auxiliary detector geometries
      - `LArVoxelReadoutGeometry, OpDetReadoutGeometry, AuxDetReadoutGeometry`
      - Each registers call-backs to classes that know how to simulate that sub-system
    - Registers particle list
  - Processes interactions one at a time
  - Collects the results and stores it in the art::Event
    - Truth information at this stage
      - simb::MCParticle
        - *Trajectory, momentum, PDG code, ...*
      - art::Assns< simb::MCTruth, simb::MCParticle >
      - sim::SimChannel
      - sim::SimPhotons or sim::SimPhotonsLite
      - sim::AuxDetSimChannel

Detector simulation



- TPC simulation
  - LAr volume split into “voxels” (3-D pixels)
    - 0.3 mm is typical in current simulations
  - `GEANT4` deposits energy in each voxel
  - `larg4::LArVoxelReadout`
    - Calculates number of electrons created by that energy deposition
      - Calculation performed by `sim::IonizationAndScintillation` singleton
      - Allows proper correlation between N electrons and photons
      - Uses `sim:ISCalculation` abstract interface for the model
    - Drifts cluster of electrons as if at center
    - Generates sequence of arrival times for each wire
      - Includes effect of longitudinal and transverse diffusion
    - Fills `sim::SimChannel` with this (truth) information



Detector simulation

- TPC simulation (cont'd)
  - `larg4::OpDetSensitiveDetector`
    - Calculates number of photoelectrons at each PMT
      - Uses sim::IonizationAndScintillation singleton to get correct N photons
      - Performs a lookup that depends upon originating voxel, terminating PMT
    - Does ***not*** perform a detailed arrival time calculation
      - Time resolution is ~20 ns
    - Fills `sim::SimPhotons` (tracks individual photon)
      or `sim::SimPhotonsLite` (only tracks N photons detected)

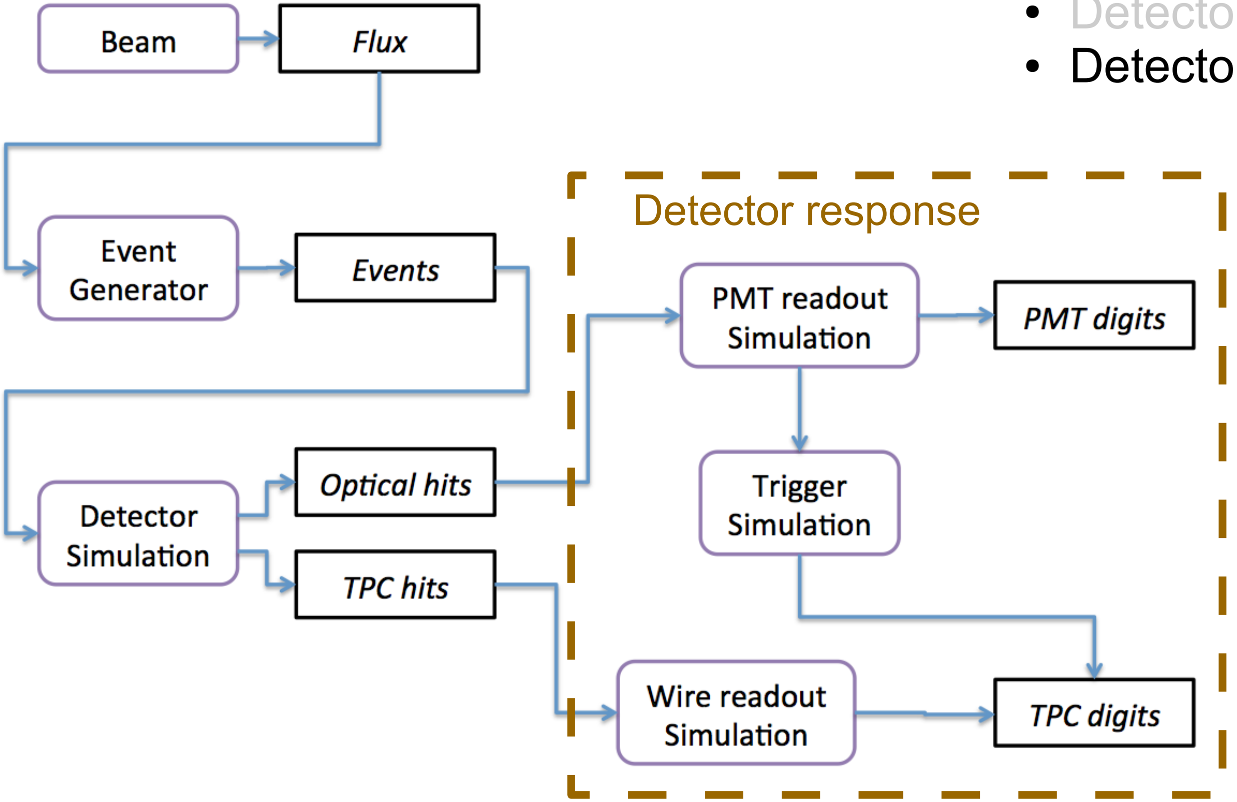
A comment about photon propagation calculation

- Have a slow method and a “fast” method
- Slow method tracks individual photons
- Fast uses `phot::PhotonVisibilityService` (in `larsim/PhotonPropagation`)
  - Lookup table to determine probability of observing an isotropically produced photon within a given voxel
  - Can include effects of geometry, scattering, attenuation, quantum efficiency

Detector simulation

- Auxiliary detector simulation
  - Geometry specifies N `AuxDetSensitiveGeo` per `AuxDetGeo`
  - GEANT4 deposits energy in each `AuxDetSensitiveGeo`
  - `larg4::AuxDetReaout`
    - Stores information about the energy deposition in `sim::AuxDetSimChannel`
    - Adds it to the event
  - Details of how the detector turns that energy into digitized signals is (currently) handled outside the simulation (e.g., in the simulation analysis)

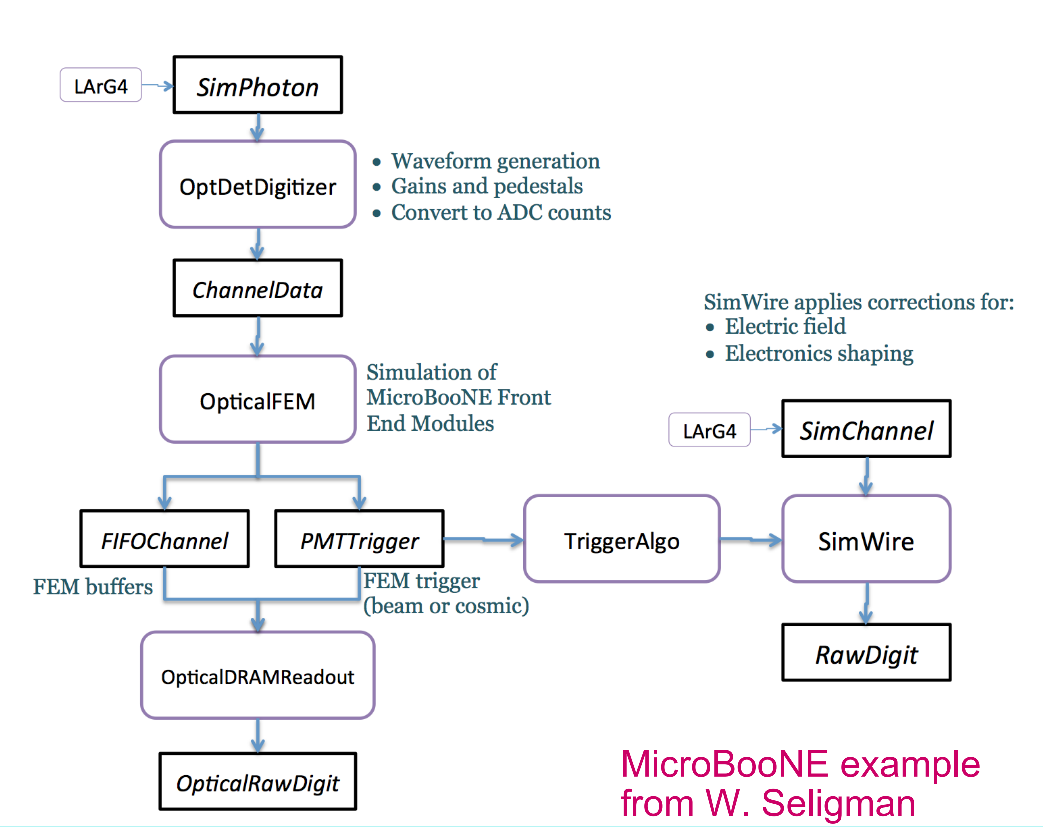
- Beam simulation
- Event generation
- Detector description
- Detector simulation
- Detector response



From W Seligman

The detector response simulation

- The remaining steps are performed after GEANT4 is done
  - Model charge / photon transport effects not in the detector simulation
  - Model DAQ response for given input charge, detected photons
    - May include triggers
  - Digitize the final output signals



- Waveform generation
- Gains and pedestals
- Convert to ADC counts

Simulation of MicroBooNE Front End Modules

FEM buffers

FEM trigger (beam or cosmic)

SimWire applies corrections for:

- Electric field
- Electronics shaping

MicroBooNE example from W. Seligman

The detector response simulation

- `detsim::SimWire` module

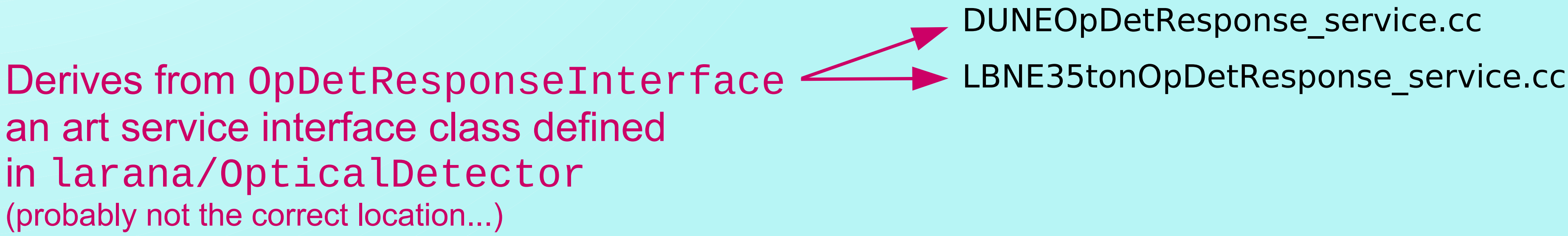
> Note: Typically held in experiment repositories with different name
>
> - E.g., `uboonecode/uboone/DetSim/SimWireMicroBooNE_module.cc`
> - The code is nearly identical, so this separation is probably unnecessary...

- Simulates signal generation on a TPC wire
  - Drift charge displacements due to electric field non-uniformities
  - Field effects
    - Raw signal induction due to motion of charge by / to wire
    - Induced signals on neighboring wires
  - Electronics effects: signal shaping, noise
  - Performs calculation in frequency space
- Output is `raw::RawDigit`

The detector-specifc parts...
...but changes to configuration
might cover the differences

The detector response simulation

- Optical detector response
  - Highly detector-dependent, and changing rapidly
  - Essentially no code in core LArSoft that handles optical response
    - For MicroBooNE: uboonecode/uboone/OpticalDetectorSim/*
    - For DUNE: lbnecode/lbne/OpticalDetector/OpDetDigitizerLBNE_module.cc



DUNEOpDetResponse_service.cc

LBNE35tonOpDetResponse_service.cc

Derives from `OpDetResponseInterface` an art service interface class defined in `larana/OpticalDetector`
(probably not the correct location...)

Will not go into further details here...

The detector response simulation

- Optical detector response (cont'd)

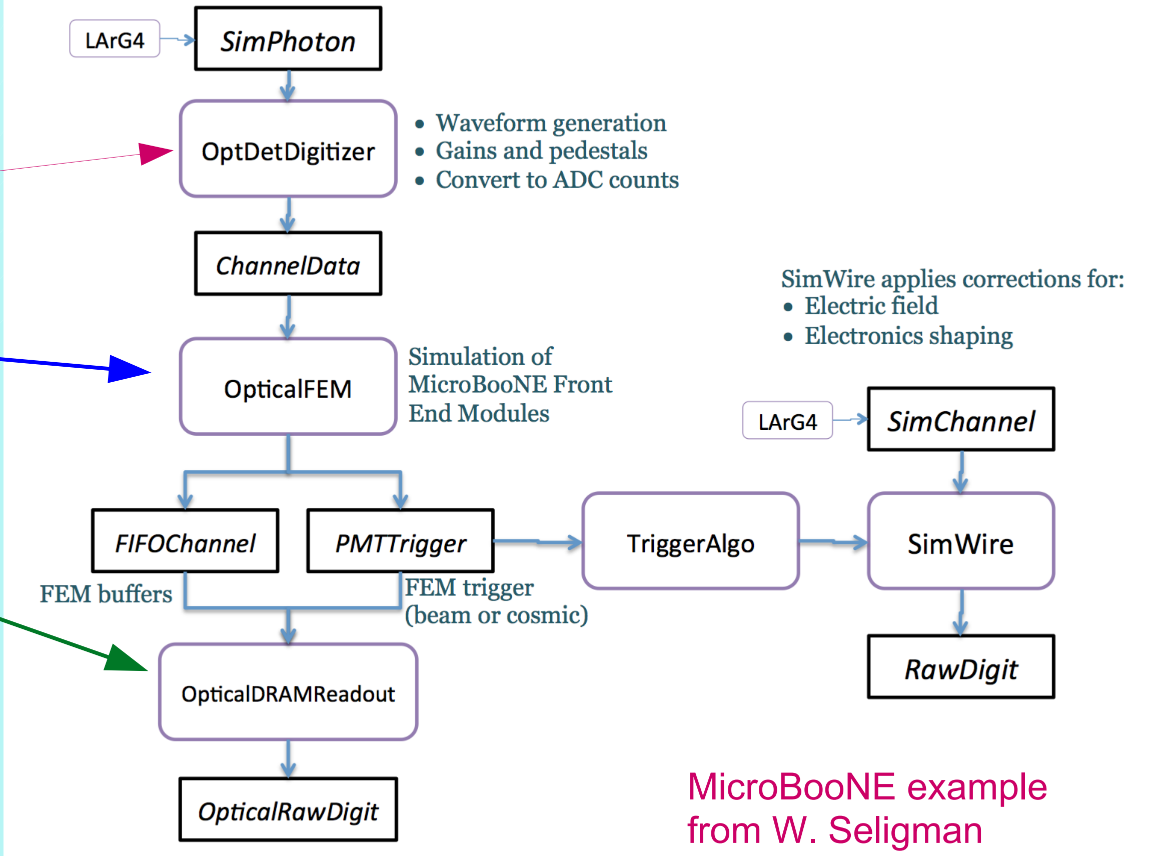
...but can stroll through
the MicroBooNE example

UBOpticalADCSim_module.cc
Waveform generation

OpticalFEM_module.cc
Converts waveform to DAQ output format needed for trigger simulation

OpticalDRAMReadout_module.cc
Converts DAQ format to raw::OpticalRawDigit

Code is in uboonecode/uboone/OpticalDetectorSim



MicroBooNE example
from W. Seligman

Running the generators and simulation

Event generator example: `prodsingle`

```
#include "services_lbne.fcl"
#include "singles_lbne.fcl"
#include "largeantmodules_lbne.fcl"
#include "detsimmodules_lbne.fcl"

process_name: SinglesGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "single35t_hist.root" }
  Timing:       {}
  RandomNumberGenerator: {} #ART native random number generator
  user:         @local::lbne35t_simulation_services
}
#services.user.ExptGeoHelperInterface:
@local::lbne_geometry_helper
#services.user.Geometry.GDML: "lbne35t4apa_v3.gdml"

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   1           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }

 # define the producer and filter modules for this
 # path, order matters,
 simulate: [ generator, largeant, daq, rns, simcounter ]

 # define the output stream
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [stream1]
}

# block to define where the output goes.
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "single35t_gen.root" #default file name
 }
}
```

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

Event generator example: `prodsingle`

```
#include "services_lbne.fcl"
#include "singles_lbne.fcl"
#include "largeantmodules_lbne.fcl"
#include "detsimmodules_lbne.fcl"

process_name: SinglesGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "single35t_hist.root" }
  Timing:       {}
  RandomNumberGenerator: {} #ART native random number generator
  user:         @local::lbne35t_simulation_services
}
#services.user.ExptGeoHelperInterface:
@local::lbne_geometry_helper
#services.user.Geometry.GDML: "lbne35t4apa_v3.gdml"

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   1           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }
```

Native art random number service

```
 # define the producer and filter modules for this
 # path, order matters,
 simulate: [ generator, largeant, daq, rns, simcounter ]

 # define the output stream
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [stream1]
}

# block to define where the output goes.
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "single35t_gen.root" #default file name
 }
}
```

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

Event generator example: `prodsingle`

```
#include "services_lbne.fcl"
#include "singles_lbne.fcl"
#include "largeantmodules_lbne.fcl"
#include "detsimmodules_lbne.fcl"

process_name: SinglesGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "single35t_hist.root" }
  Timing:       {}
  RandomNumberGenerator: {} #ART native random number generator
  user:         @local::lbne35t_simulation_services
}
#services.user.ExptGeoHelperInterface:
@local::lbne_geometry_helper
#services.user.Geometry.GDML: "lbne35t4apa_v3.gdml"

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   1           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }
```

```
 # define the producer and filter modules for this
 # path, order matters,
 simulate: [ generator, largeant, daq, rns, simcounter ]

 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [stream1]
}

# block to define where the output goes.
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "single35t_gen.root" #default file name
 }
}
```

Defined in lbnecode/lbne/Utilities/services_lbne.fcl

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

...so look there and find:

lbne35t_simulation_services: @local::lbne35t_g4_services
lbne35t_simulation_services.LArFFT: @local::lbne35t_larfft
lbne35t_simulation_services.SignalShapingServiceLBNE35t: @local::lbne35t_signalshapingservice
lbne35t_simulation_services.PhotonVisibilityService: @local::lbne35t_photonvisibilityservice
lbne35t_simulation_services.BackTracker: @local::lbne35t_backtracker

```
#inclu
#inclu
#inclu
#inclu

proces

servic
{
  # Lo
  TFileService: { fileName: "single35t_hist.root" }
  Timing:       {}
  RandomNumberGenerator: {} #ART native random number generator
  user:         @local::lbne35t_simulation_services
}
#services.user.ExptGeoHelperInterface:
@local::lbne_geometry_helper
#services.user.Geometry.GDML: "lbne35t4apa_v3.gdml"

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   1           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }
```

```
 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [stream1]
}

# block to define where the output goes.
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "single35t_gen.root" #default file name
 }
}
```

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

...so look there and find:

lbne35t_simulation_services: @local::lbne35t_g4_services
lbne35t_simulation_services.LArFFT: @local::lbne35t_larfft
lbne35t_simulation_services.SignalShapingServiceLBNE35t: @local::lbne35t_signalshapingservice
lbne35t_simulation_services.PhotonVisibilityService: @local::lbne35t_photonvisibilityservice

lbne35t_g4_services: @local::lbne35t_gen_services
lbne35t_g4_services.LArG4Parameters: @local::lbne35t_largeantparameters
lbne35t_g4_services.LArVoxelCalculator: @local::lbne35t_larvoxelcalculator
lbne35t_g4_services.PhotonVisibilityService: @local::lbne35t_photonvisibilityservice
lbne35t_g4_services.OpDetResponseInterface: @local::lbne35t_opdetresponse

```
#inclu
#inclu
#inclu
#inclu

proces

serv
{
  #
  TF
  Ti                                                                       the paths that do
  R
  us
}
#ser
@loc                                                                       es.
#services.user.Geometry.GDML: "lbne35t4apa_v3.gdml"

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   1           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }
```

```
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "single35t_gen.root" #default file name
 }
}
```

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

...so look there and find:

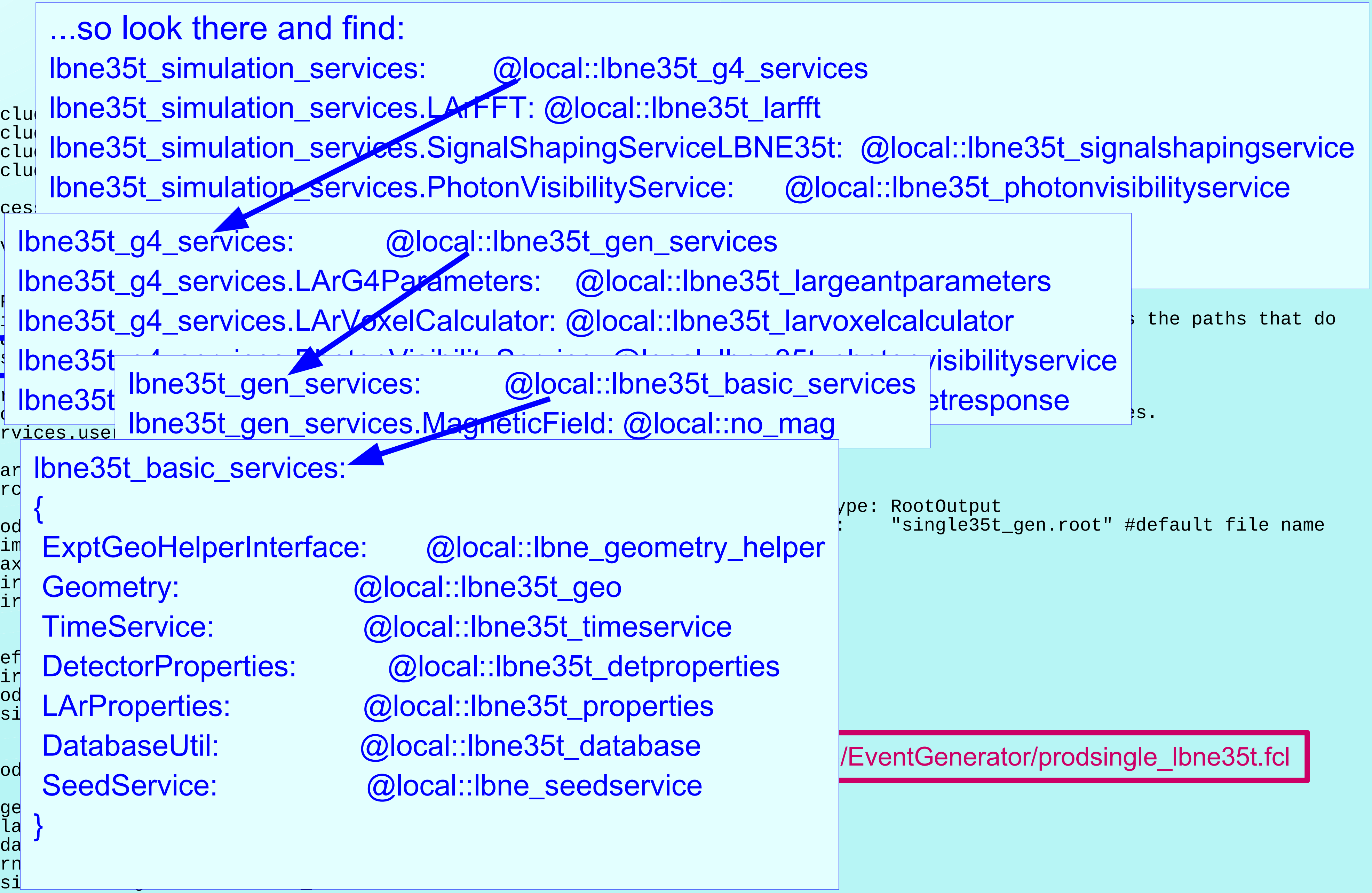
lbne35t_simulation_services: @local::lbne35t_g4_services
lbne35t_simulation_services.LArFFT: @local::lbne35t_larfft
lbne35t_simulation_services.SignalShapingServiceLBNE35t: @local::lbne35t_signalshapingservice
lbne35t_simulation_services.PhotonVisibilityService: @local::lbne35t_photonvisibilityservice

lbne35t_g4_services: @local::lbne35t_gen_services
lbne35t_g4_services.LArG4Parameters: @local::lbne35t_largeantparameters
lbne35t_g4_services.LArVoxelCalculator: @local::lbne35t_larvoxelcalculator
lbne35t_g4_services.PhotonVisibilityService: @local::lbne35t_photonvisibilityservice
lbne35t
etresponse

lbne35t_gen_services: @local::lbne35t_basic_services
lbne35t_gen_services.MagneticField: @local::no_mag

```
#inclu
#inclu
#inclu
#inclu

proces

serv
{
  #
  TF
  Ti                                                                                 s the paths that do
  Ro
  us
}
#ser
@loc                                                                                  es.
#services.user
                                                                        {
#Start each new event with an empty event.                               out1:
source:                                                                  {
{                                                                          module_type: RootOutput
  module_type: EmptyEvent                                                  fileName:    "single35t_gen.root" #default file name
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }            }
  maxEvents:   1           # Number of events to create                 }
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }
```

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl



...so look there and find:

```
lbne35t_simulation_services:       @local::lbne35t_g4_services
lbne35t_simulation_services.LArFFT: @local::lbne35t_larfft
lbne35t_simulation_services.SignalShapingServiceLBNE35t:  @local::lbne35t_signalshapingservice
lbne35t_simulation_services.PhotonVisibilityService:     @local::lbne35t_photonvisibilityservice
```

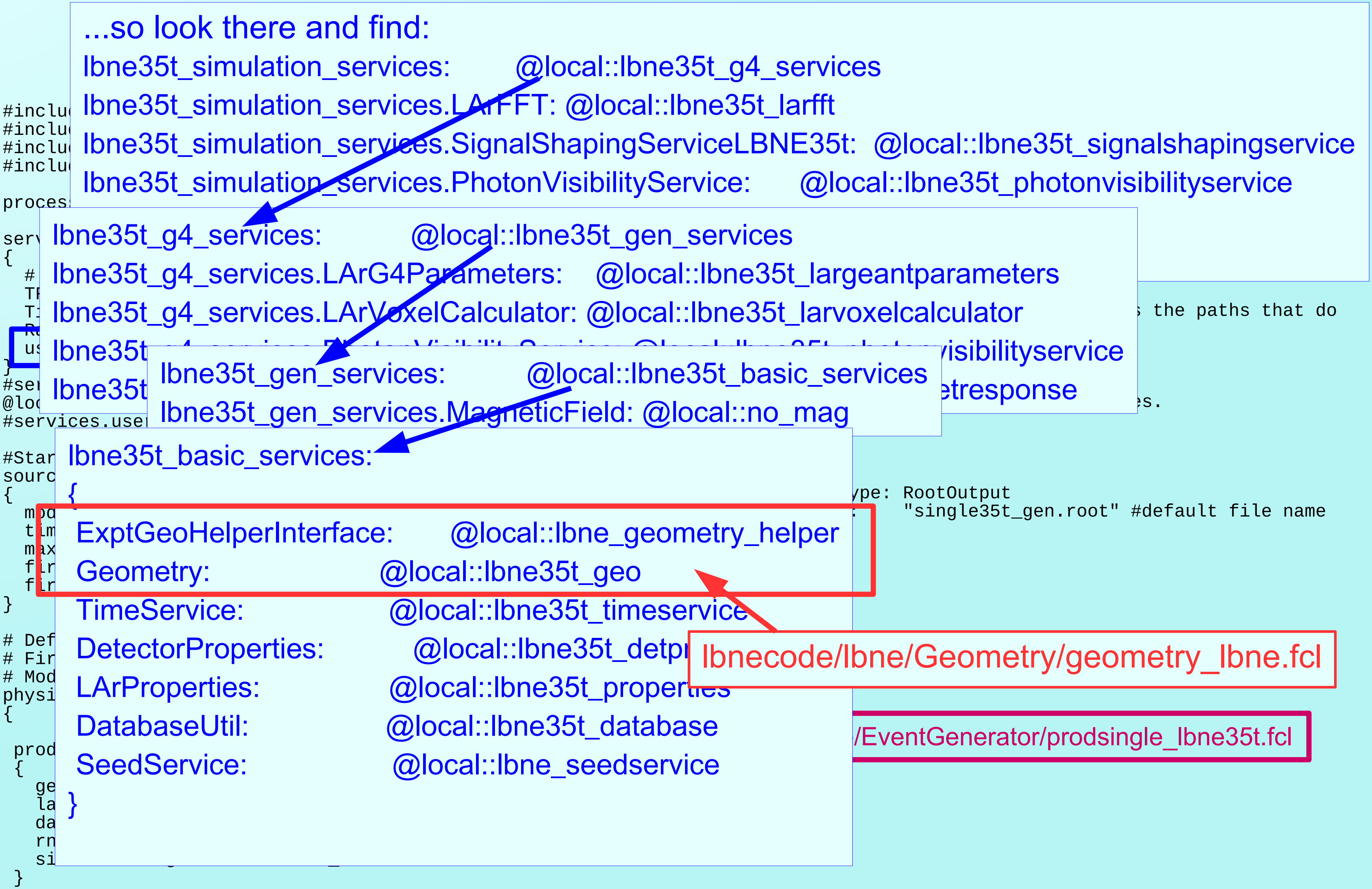
```
lbne35t_g4_services:         @local::lbne35t_gen_services
lbne35t_g4_services.LArG4Parameters:    @local::lbne35t_largeantparameters
lbne35t_g4_services.LArVoxelCalculator: @local::lbne35t_larvoxelcalculator
lbne35t                                                    visibilityservice
lbne35t                                                    etresponse
```

```
lbne35t_gen_services:        @local::lbne35t_basic_services
lbne35t_gen_services.MagneticField: @local::no_mag
```

```
lbne35t_basic_services:
{
 ExptGeoHelperInterface:      @local::lbne_geometry_helper
 Geometry:                   @local::lbne35t_geo
 TimeService:                 @local::lbne35t_timeservice
 DetectorProperties:          @local::lbne35t_detproperties
 LArProperties:               @local::lbne35t_properties
 DatabaseUtil:                @local::lbne35t_database
 SeedService:                 @local::lbne_seedservice
}
```

```
#inclu
#inclu
#inclu
#inclu

proces

serv
{
  #
  TF
  Ti                                                                       s the paths that do
  Ro
  us
}
#ser
@loc                                                                       es.
#services.user

#Star
sourc
{                                                       ype: RootOutput
  mod                                                   :    "single35t_gen.root" #default file name
  tim
  max
  fir
  fir
}

# Def
# Fir
# Mod
physi
{

 prod
 {
   ge
   la
   da
   rn
   si
 }
```

/EventGenerator/prodsingle_lbne35t.fcl



...so look there and find:

```
lbne35t_simulation_services:       @local::lbne35t_g4_services
lbne35t_simulation_services.LArFFT: @local::lbne35t_larfft
lbne35t_simulation_services.SignalShapingServiceLBNE35t:  @local::lbne35t_signalshapingservice
lbne35t_simulation_services.PhotonVisibilityService:     @local::lbne35t_photonvisibilityservice
```

```
lbne35t_g4_services:         @local::lbne35t_gen_services
lbne35t_g4_services.LArG4Parameters:    @local::lbne35t_largeantparameters
lbne35t_g4_services.LArVoxelCalculator: @local::lbne35t_larvoxelcalculator
lbne35t                                                    visibilityservice
lbne35t                                                   etresponse
```

```
lbne35t_gen_services:        @local::lbne35t_basic_services
lbne35t_gen_services.MagneticField: @local::no_mag
```

```
lbne35t_basic_services:
{
 ExptGeoHelperInterface:      @local::lbne_geometry_helper
 Geometry:                    @local::lbne35t_geo
 TimeService:                 @local::lbne35t_timeservice
 DetectorProperties:          @local::lbne35t_detpr
 LArProperties:               @local::lbne35t_properties
 DatabaseUtil:                @local::lbne35t_database
 SeedService:                 @local::lbne_seedservice
}
```

lbnecode/lbne/Geometry/geometry_lbne.fcl

/EventGenerator/prodsingle_lbne35t.fcl

```
#inclu
#inclu
#inclu
#inclu

proces

serv
{
  #
  TF
  Ti                                                          s the paths that do
  Ro
  us
}
#ser                                                        es.
@loc
#services.user

#Star
sourc
{                                           ype: RootOutput
  mod                                       :     "single35t_gen.root" #default file name
  tim
  max
  fir
  fir
}

# Def
# Fir
# Mod
physi
{

 prod
 {
   ge
   la
   da
   rn
   si
 }
```

```
lbne35t_geo: {
 Name:     "lbne35t4apa_v4"
 # Choose GDML file and set detector version similarly
 GDML:        "lbne35t4apa_v4.gdml"
 ROOT:        "lbne35t4apa_v4.gdml"
 SortingParameters: { DetectorVersion: "lbne35t4apa_v4"  }
 SurfaceY:         0.0e2          # in cm, vertical distance to the surface
 DisableWiresInG4:   true
}

lbne_geometry_helper: {
 service_provider : LBNEGeometryHelper
}
```

lbnecode/lbne/Geometry/geometry_lbne.fcl

```
lbne35t_ba
{
 ExptGeoHelperInterface:      @local::lbne_geometry_helper
 Geometry:                    @local::lbne35t_geo
 TimeService:                 @local::lbne35t_timeservice
 DetectorProperties:            @local::lbne35t_detpr
 LArProperties:               @local::lbne35t_properties
 DatabaseUtil:                @local::lbne35t_database
 SeedService:                 @local::lbne_seedservice
}
```

```
...so look
lbne35t_s
lbne35t_s
lbne35t_s
lbne35t_s
```

```
gnalshapingservice
isibilityservice
```

```
lbne35t_g4_
lbne35t_g4_
lbne35t_g4_
lbne35t_g4
lbne35t
lbne
lbne
```

```
#inclu
#inclu
#inclu
#inclu

proces

serv
{
  #
  TF
  Ti
  R
  us
}
#ser
@loc
#services.user

#Star
sourc
{
  mod
  tim
  max
  fir
  fir
}

# Def
# Fir
# Mod
physi
{

 prod
 {
   ge
   la
   da
   rn
   si
 }
```

```
s the paths that do
e
es.
ype: RootOutput
:       "single35t_gen.root" #default file name
```

/EventGenerator/prodsingle_lbne35t.fcl

```
lbne35t_geo: {
 Name:     "lbne35t4apa_v4"
 # Choose GDML file and set detector version similarly
 GDML:        "lbne35t4apa_v4.gdml"
 ROOT:        "lbne35t4apa_v4.gdml"
 SortingParameters: { DetectorVersion: "lbne35t4apa_v4"  }
 SurfaceY:          0.0e2          # in cm, vertical distance to the surface
 DisableWiresInG4:   true
}

lbne_geometry_helper: {
 service_provider : LBNEGeometryHelper
}
```

The geometry

To get the correct ChannelMapAlg

```
lbne35t_ba
{
 ExptGeoHelperInterface:      @local::lbne_geometry_helper
 Geometry:                    @local::lbne35t_geo
 TimeService:                 @local::lbne35t_timeservice
 DetectorProperties:          @local::lbne35t_detpr
 LArProperties:               @local::lbne35t_properties
 DatabaseUtil:                @local::lbne35t_database
 SeedService:                 @local::lbne_seedservice
}
```

lbnecode/lbne/Geometry/geometry_lbne.fcl

/EventGenerator/prodsingle_lbne35t.fcl

...so look
lbne35t_s
lbne35t_s
lbne35t_s
lbne35t_s

gnalshapingservice
isibilityservice

lbne35t_g4_
lbne35t_g4_
lbne35t_g4_
lbne35t_g4
lbne35t

lbne
lbne

e

```
#inclu
#inclu
#inclu
#inclu

proces

serv
{
  #
  TF
  Ti
  Ri
  us
}
#ser
@loc
#services.user

#Star
sourc
{
  mod
  tim
  max
  fir
  fir
}

# Def
# Fir
# Mod
physi
{

 prod
 {
   ge
   la
   da
   rn
   si
 }
```

```
s the paths that do

es.

ype: RootOutput
:     "single35t_gen.root" #default file name
```

Event generator example: `prodsingle`

```
#include "services_lbne.fcl"
#include "singles_lbne.fcl"
#include "largeantmodules_lbne.fcl"
#include "detsimmodules_lbne.fcl"

process_name: SinglesGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "single35t_hist.root" }
  Timing:       {}
  RandomNumberGenerator: {} #ART native random number generator
  user:         @local::lbne35t_simulation_services
}
#services.user.ExptGeoHelperInterface:
@local::lbne_geometry_helper
#services.user.Geometry.GDML: "lbne35t4apa_v3.gdml"

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   1           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
 {
   generator: @local::lbne35t_singlep
   largeant:  @local::lbne35t_largeant
   daq:       @local::lbne35t_simwire
   rns:       { module_type: "RandomNumberSaver" }
   simcounter: @local::lbne35t_simcounter
 }
```

EmptyEvent input module

```
 # define the producer and filter modules for this
 # path, order matters,
 simulate: [ generator, largeant, daq, rns, simcounter ]

 # define the output stream
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [stream1]
}

# block to define where the output goes.
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "single35t_gen.root" #default file name
 }
}
```

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

Event generator example: `prodsingle`

```
#include "services_lbne.fcl"
#include "singles_lbne.fcl"
#include "largeantmodules_lbne.fcl"
#include "detsimmodules_lbne.fcl"

process_name: SinglesGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "single35t_hist.root" }
  Timing:       {}
  RandomNumberGenerator: {} #ART native random number generator
  user:         @local::lbne35t_simulation_services
}
#services.user.ExptGeoHelperInterface:
@local::lbne_geometry_helper
#services.user.Geometry.GDML: "lbne35t

#Start each new event with an empty ev
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "Gen
  maxEvents:   1           # Number of
  firstRun:    1           # Run numbe
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
# First modules are defined; they are scheduled later.
# Modules are grouped by type.
physics:
{

 producers:
  {
    generator: @local::lbne35t_singlep
    largeant:  @local::lbne35t_largeant
    daq:       @local::lbne35t_simwire
    rns:       { module_type: "RandomNumberSaver" }
    simcounter: @local::lbne35t_simcounter
  }
```

```
 # define the producer and filter modules for this
 # path, order matters,
 simulate: [ generator, largeant, daq, rns, simcounter ]

 # define the output stream
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [stream1]
}

pes.

#default file name
```

The complete generation – simulation workflow
Generator = SingleGen
Then:
- LArG4
- SimWireLBNE35t

lbnecode/lbne/EventGenerator/prodsingle_lbne35t.fcl

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl"
#include  “tsimmodules_microboone.fcl"
#include  “triggersim_microboone.fcl"
#include  ”opticaldetectorsim_microboone.fcl"
#include  “mccheatermodules.fcl"

process_name: GenieGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "genie_hist_uboone.root" }
  Timing:       {}
  RandomNumberGenerator: {} # ART native random number generator
  user:         @local::microboone_full_services
}
services.user.BackTracker: @local::microboone_backtracker

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   5           # Number of events to create
  firstRun:    1           # Run number to use for this file
  firstEvent:  1           # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

```
 analyzers:
 {
   largana:   @local::microboone_largeantana
 }

 # define the producer and filter modules for this path
 # filters reject all following items.  see lines starting
 # physics.producers below
 simulate: [ generator, largeant, backtrack, optdigitizer,
           optfem, triggersim, optreadout, daq ]
 analyzeIt:  [ largana ]
 # define the output stream, there could be more than one
 # if using filters
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event,
 #ie filters and producers
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [analyzeIt, stream1]
}

#block to define where the output goes.  if you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl"
#include  “tsimmodules_microboone.fcl"
#include  “triggersim_microboone.fcl"
#include  ”opticaldetectorsim_microboone.fcl"
#include  “mccheatermodules.fcl"

process_name: GenieGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "genie_hist_uboone.root" }
  Timing:       {}
  RandomNumberGenerator: {} # ART native random number generator
  user:         @local::microboone_full_services
}
services.user.BackTracker: @local::microboone_backtracker

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   5          # Number of events to create
  firstRun:    1          # Run number to use for this file
  firstEvent:  1          # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

Defined in uboonecode/uboone/Utilities/services_microboone.fcl

```
 analyzers:
 {
   largana:   @local::microboone_largeantana
 }

 # define the producer and filter modules for this path
 simulate: [ generator, largeant, backtrack, optdigitizer,
           optfem, triggersim, optreadout, daq ]
 analyzeIt:  [ largana ]
 # define the output stream, there could be more than one
 # if using filters
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event,
 #ie filters and producers
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [analyzeIt, stream1]
}

#block to define where the output goes.  if you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

Defined in uboonecode/uboone/Utilities/services_microboone.fcl

microboone_full_services: @local::microboone_simulation_services
microboone_full_services.BackTracker: @local::microboone_backtracker

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl"
#include  "detsimmodules_microboone.fcl"
#includ
#includ
#includ

process_name: GenieGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "genie_hist_uboone.root" }
  Timing:       {}
  RandomNumberGenerator: {} # ART native random number generator
  user:         @local::microboone_full_services
}
services.user.BackTracker: @local::microboone_backtracker

#Start each new event with an empty event.
source:
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   5          # Number of events to create
  firstRun:    1          # Run number to use for this file
  firstEvent:  1          # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

```
 analyzers:

 }

                                      s for this path
                                      see lines starting

 simulate: [ generator, largeant, backtrack, optdigitizer,
             optfem, triggersim, optreadout, daq ]
 analyzeIt:  [ largana ]
 # define the output stream, there could be more than one
 # if using filters
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event,
 #ie filters and producers
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [analyzeIt, stream1]
}

#block to define where the output goes.  if you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl
#includ
#incluc
#incluc
#incluc

process name: GenieGen
```

```
{
  module_type: EmptyEvent
  timestampPlugin: { plugin_type: "GeneratedEventTimestamp" }
  maxEvents:   5          # Number of events to create
  firstRun:    1          # Run number to use for this file
  firstEvent:  1          # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

```
 analyzers:
 }

                  s for this path
                 see lines starting

 simulate: [ generator, largeant, backtrack, optdigitizer,
```

```
 end_paths:     [analyzeIt, stream1]
}

#block to define where the output goes.  if you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

Defined in uboonecode/uboone/Utilities/services_microboone.fcl

```
microboone_full_services:                @local::microboone_simulation_services
microboone_full_services.BackTracker:    @local::microboone_backtracker
```

```
microboone_simulation_services:                                     @local::microboone_g4_services
microboone_simulation_services.LArFFT:                              @local::microboone_larfft
microboone_simulation_services.SignalShapingServiceMicroBooNE:  @local::microboone_signalshapingservi
microboone_simulation_services.UBOpticalChConfig:                   @local::microboone_optical_ch_config
microboone_simulation_services.UBOpReadoutMap:                      @local::microboone_opreadoutmap
microboone_simulation_services.ExptGeoHelperInterface:              @local::microboone_geometry_helper
```

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl
#includ
#inclu
#inclu
#inclu

process_name: GenieGen
```

```
s for this path
see lines starting

simulate: [ generator, largeant, backtrack, optdigitizer,
```

Defined in uboonecode/uboone/Utilities/services_microboone.fcl

```
microboone_full_services:                @local::microboone_simulation_services
microboone_full_services.BackTracker:        @local::microboone_backtracker
```

```
microboone_simulation_services:                                @local::microboone_g4_services
micr
micr
micr
micr
micr
```

```
{
  mo
  ti
  ma
  fi
  fi
}

# Def
physi
{

 pro
 {
   ge
   la
   ba
   op
   op
   tr
   op
   da
 }
```

```
microboone_g4_services:                          @local::microboone_g4_dark_services
microboone_g4_services.PhotonVisibilityService: @local::microboone_photonvisibilityservice
microboone_g4_services.LArProperties.ScintYield:         24000
microboone_g4_services.LArProperties.ScintPreScale:        0.01   # Prescale production by 0.01, correct
                                                 # MUST match between g4 and detsim
microboone_g4_services.LArProperties.EnableCerenkovLight: false # Cerenkov light OFF by default
microboone_g4_services.LArG4Parameters.UseCustomPhysics:  true
microboone_g4_services.LArG4Parameters.EnabledPhysics:    [ "Em",
                                             "FastOptical",
                                             "SynchrotronAndGN",
                                             "Ion",
                                             "Hadron",
                                             "Decay",
                                             "HadronElastic",
                                             "Stopping",
                                             "NeutronTrackingCut" ]
```

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl"
#includ
#inclu
#inclu
#inclu

process_name: GenieGen
```

```
analyzers:
}
s for this path
see lines starting
simulate: [ generator, largeant, backtrack, optdigitizer,
```

Defined in uboonecode/uboone/Utilities/services_microboone.fcl

```
microboone_full_services:                  @local::microboone_simulation_services
microboone_full_services.BackTracker:          @local::microboone_backtracker
```

```
microboone_simulation_services:                  @local::microboone_g4_services
micro
micro
micro
micro
micro
correct
ault
```

```
microboone_g4_services:                 @local::microboone_g4_dark_services
microboone_g4_services.PhotonVisibilityService: @local::microboone_photonvisibilityservice
microbo
microbo
microbo
microbo
microbo
```

```
microboone_g4_dark_services:                    @local::microboone_gen_services
microboone_g4_dark_services.LArG4Parameters:    @local::microboone_largeantparameters
microboone_g4_dark_services.LArVoxelCalculator: @local::microboone_larvoxelcalculator
microboone_g4_dark_services.LArG4Parameters.KeepEMShowerDaughters: true
microboone_g4_dark_services.LArG4Parameters.StoreTrajectories: true
microboone_g4_dark_services.LArG4Parameters.UseModBoxRecomb: true
```

```
"SynchrotronAndGN",
"Ion",
"Hadron",
"Decay",
"HadronElastic",
"Stopping",
"NeutronTrackingCut" ]
```

```
{
  mod
  tin
  max
  fi
  fi
}

# Def
physi
{

 proc
 {
   ge
   la
   ba
   op
   op
   tr
   op
   da
 }
```

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl
#includ
#includ
#includ                                                    s for this path
#includ                                                    see lines starting

process_name: GenieGen                                     simulate: [ generator, largeant, backtrack, optdigitizer,
```

Defined in uboonecode/uboone/Utilities/services_microboone.fcl

microboone_full_services: @local::microboone_simulation_services
microboone_full_services.BackTracker: @local::microboone_backtracker

microboone_simulation_services: @local::microboone_g4_services

micro
micro
micro
micro
mic

correct
ault

microboone_g4_services: @local::microboone_g4_dark_services
microbo
microbo
microbo
microbo
microbo

microboone_g4_dark_services: @local::microboone_gen_services
microboone_g4_dark_services.LArG4Parameters: @local::microboone_largeantparam
roboone_larvoxelcalcu
lator
microboone_g4_dark_services.LArG4Parameters.KeepEMShowerDaughters: true
microboone_g4_dark_services.LArG4Parameters.StoreTrajectories: true
microboone_g4_dark_services.LArG4Parameters.UseModBoxRecomb: true

microboone_gen_services: @local::microboone_basic_services
microboone_gen_services.MagneticField: @local::no_mag

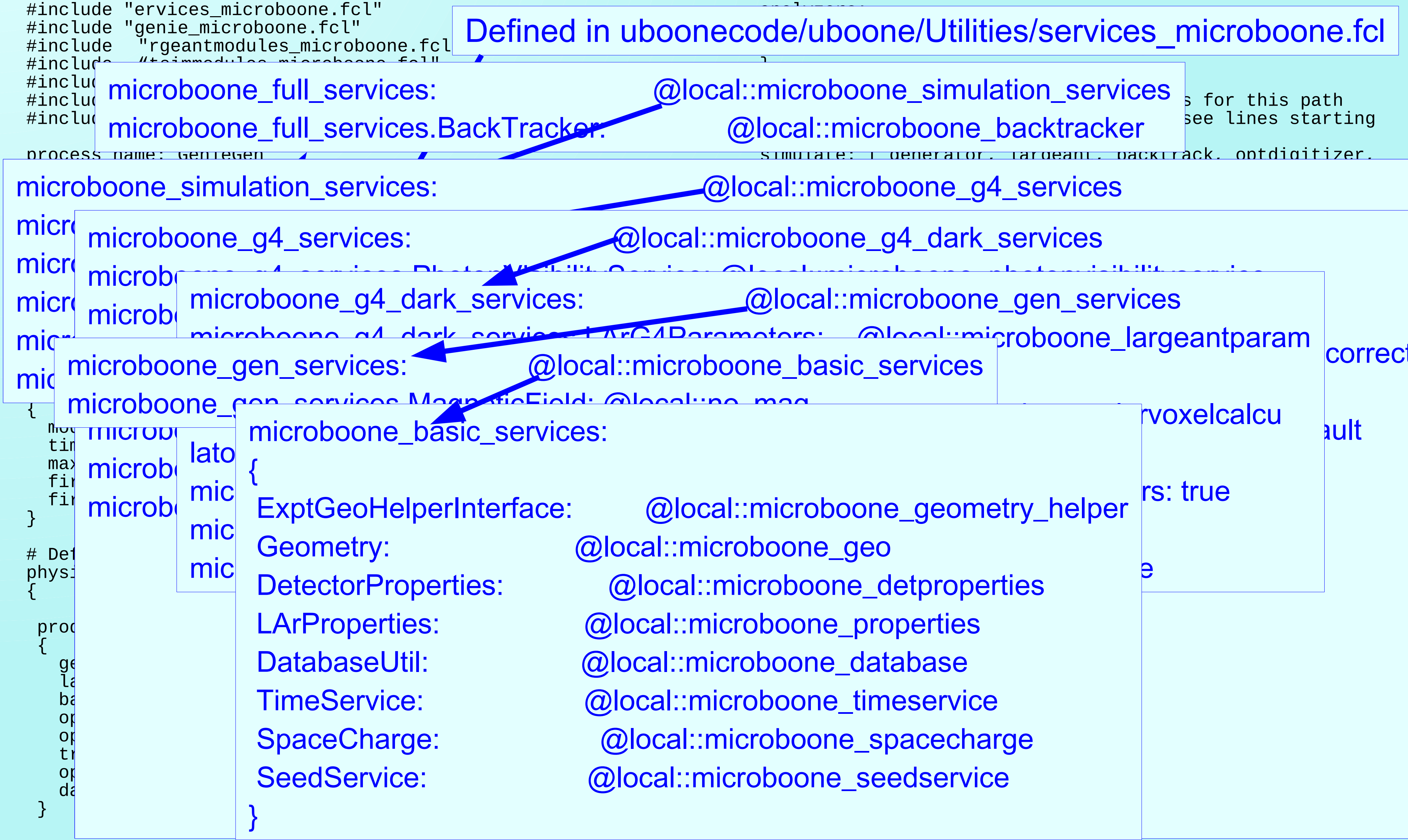
```
{
  mo
  tir
  max
  fir
  fir
}

# Def
physi
{

 proc
 {
   ge
   la
   ba
   op
   op
   tr
   op
   da
 }
```

"Ion",
"Hadron",
"Decay",
"HadronElastic",
"Stopping",
"NeutronTrackingCut"]

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl
#inclu
#inclu
#inclu
#inclu

process_name: GenieGen
{
  mod
  tim
  max
  fir
  fir
}

# Def
physi
{

 proc
 {
   ge
   la
   ba
   op
   op
   tr
   op
   da
 }
```

```
analyzers:
}

s for this path
see lines starting
simulate: [ generator, largeant, backtrack, optdigitizer,
```



Defined in uboonecode/uboone/Utilities/services_microboone.fcl

microboone_full_services: @local::microboone_simulation_services
microboone_full_services.BackTracker: @local::microboone_backtracker

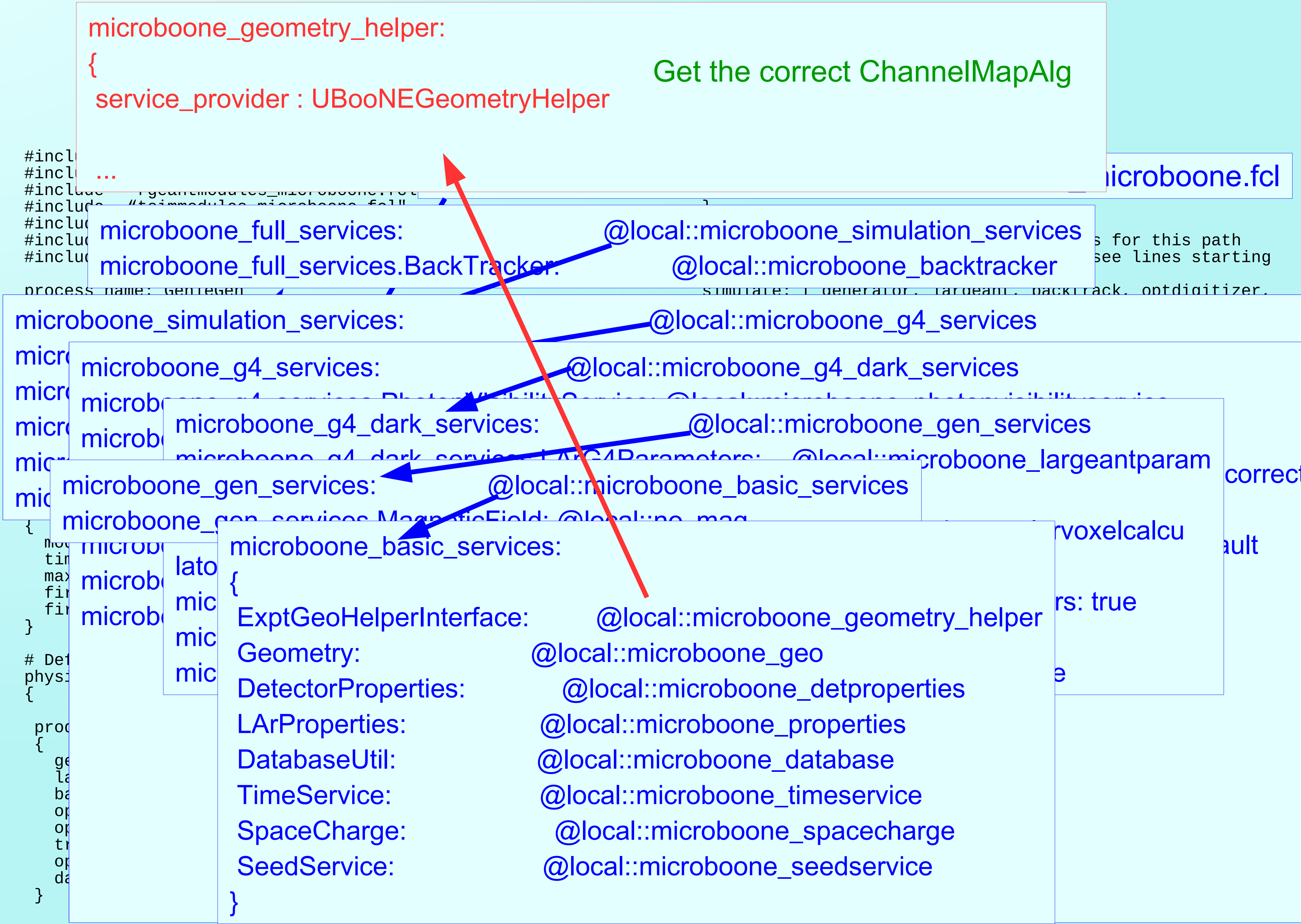
microboone_simulation_services: @local::microboone_g4_services
micr
micr
micr
micr
mic

microboone_g4_services: @local::microboone_g4_dark_services
microboone_g4_services.PhotonVisibilityService: @local::microboone_photonvisibilityservice
microb
microb
microb
microb

microboone_g4_dark_services: @local::microboone_gen_services
microboone_g4_dark_services.LArG4Parameters: @local::microboone_largeantparam
correct
lato
rvoxelcalcu
ault
mic
rs: true
mic
mic
e

microboone_gen_services: @local::microboone_basic_services
microboone_gen_services.MagneticField: @local::no_mag

```
microboone_basic_services:
{
 ExptGeoHelperInterface:        @local::microboone_geometry_helper
 Geometry:                      @local::microboone_geo
 DetectorProperties:            @local::microboone_detproperties
 LArProperties:                 @local::microboone_properties
 DatabaseUtil:                  @local::microboone_database
 TimeService:                   @local::microboone_timeservice
 SpaceCharge:                   @local::microboone_spacecharge
 SeedService:                   @local::microboone_seedservice
}
```



```
microboone_geometry_helper:
{
 service_provider : UBooNEGeometryHelper

...
```

Get the correct ChannelMapAlg

```
#incl
#incl
#include   "lgeantmodules_microboone.fcl"
#include   "tpcimmodules_microboone.fcl"
#inclu
#inclu
#inclu

process_name: GenieGen
```

...icroboone.fcl

```
s for this path
see lines starting
simulate: [ generator, largeant, backtrack, optdigitizer,
```

```
microboone_full_services:                @local::microboone_simulation_services
microboone_full_services.BackTracker:        @local::microboone_backtracker
```

```
microboone_simulation_services:              @local::microboone_g4_services
micro
micro
micro
micro
mic
```

```
microboone_g4_services:                  @local::microboone_g4_dark_services
microboone_g4_services.PhotonVisibilityService: @local::microboone_photonvisibilityservice
microbo
microbo
microbo
microbo
```

```
microboone_g4_dark_services:                 @local::microboone_gen_services
microboone_g4_dark_services.LArG4Parameters:    @local::microboone_largeantparam
lato
mic
mic
mic
```

correct

rvoxelcalcu

ault

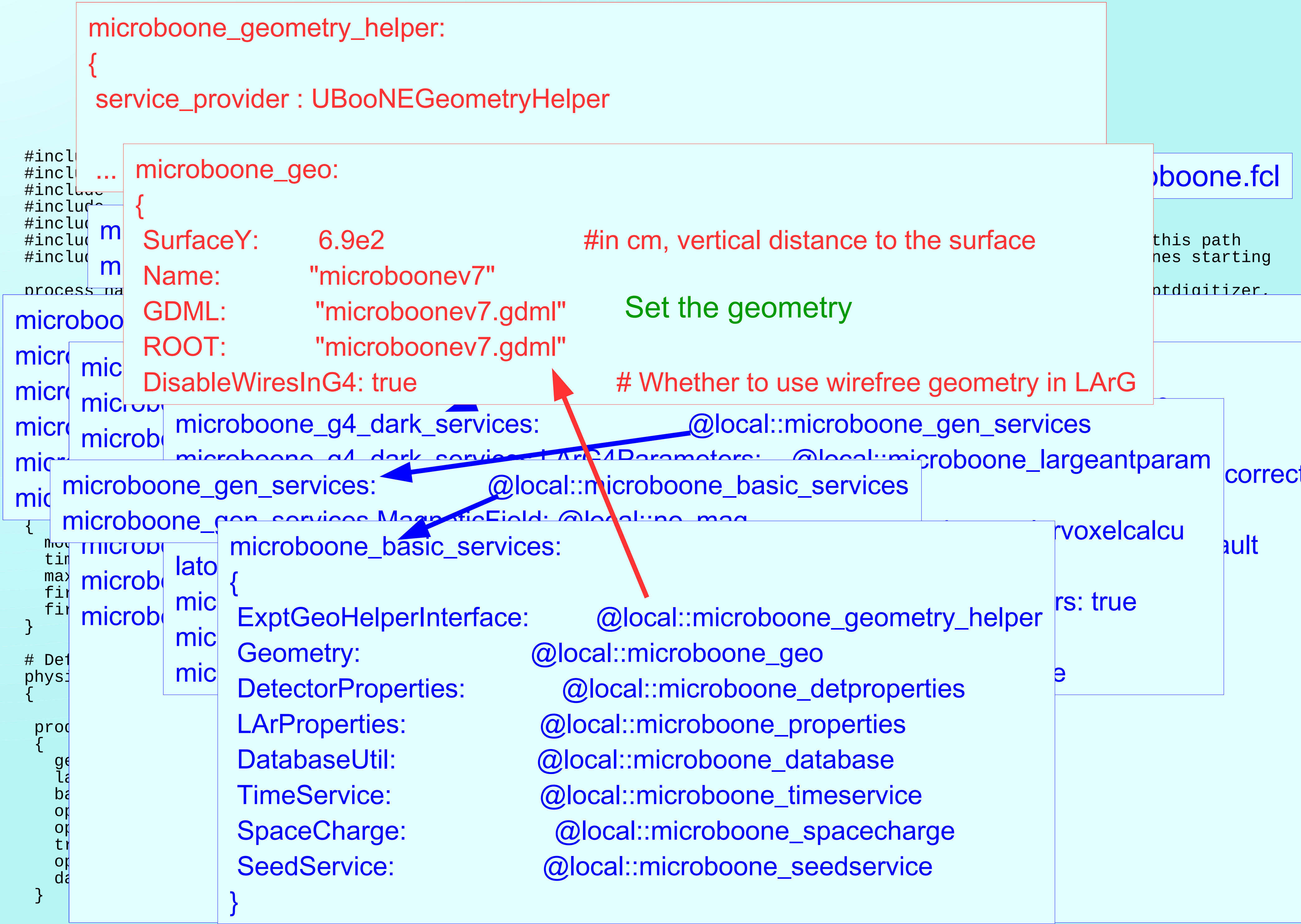
rs: true

e

```
microboone_gen_services:             @local::microboone_basic_services
microboone_gen_services.MagneticField: @local::no_mag
```

```
{
  mo
  tim
  max
  fir
  fir
}

# Def
physi
{

 proc
 {
   ge
   la
   ba
   op
   op
   tr
   op
   da
 }
```

```
microboone_basic_services:
{
 ExptGeoHelperInterface:       @local::microboone_geometry_helper
 Geometry:                     @local::microboone_geo
 DetectorProperties:           @local::microboone_detproperties
 LArProperties:                @local::microboone_properties
 DatabaseUtil:                 @local::microboone_database
 TimeService:                  @local::microboone_timeservice
 SpaceCharge:                  @local::microboone_spacecharge
 SeedService:                  @local::microboone_seedservice
}
```



```
microboone_geometry_helper:
{
 service_provider : UBooNEGeometryHelper

...
```

```
microboone_geo:
{
 SurfaceY:       6.9e2                 #in cm, vertical distance to the surface
 Name:          "microboonev7"
 GDML:           "microboonev7.gdml"
 ROOT:           "microboonev7.gdml"
 DisableWiresInG4: true                # Whether to use wirefree geometry in LArG
```

Set the geometry

```
microboone_g4_dark_services:                  @local::microboone_gen_services
microboone_g4_dark_services.LArG4Parameters:   @local::microboone_largeantparam
```

```
microboone_gen_services:                @local::microboone_basic_services
microboone_gen_services.MagneticField: @local::no_mag
```

```
microboone_basic_services:
{
 ExptGeoHelperInterface:        @local::microboone_geometry_helper
 Geometry:                     @local::microboone_geo
 DetectorProperties:             @local::microboone_detproperties
 LArProperties:                 @local::microboone_properties
 DatabaseUtil:                  @local::microboone_database
 TimeService:                   @local::microboone_timeservice
 SpaceCharge:                     @local::microboone_spacecharge
 SeedService:                    @local::microboone_seedservice
}
```

```
#incl
#incl
#include
#include
#inclu
#inclu
#inclu
process_na
{
  mo
  tim
  max
  fir
  fir
}
# Def
physi
{
 proc
 {
   ge
   la
   ba
   op
   op
   tr
   op
   da
 }
```

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl"
#include  “tsimmodules_microboone.fcl"
#include  “triggersim_microboone.fcl"
#include  ”opticaldetectorsim_microboone.fcl"
#include  “mccheatermodules.fcl"

process_name: GenieGen

services:
{
  # Load the service that manages root files for histograms.
  TFileService: { fileName: "genie_hist_uboone.root" }
  Timing:       {}
  RandomNumberGenerator: {} # ART native random number generator
  user:         @local::microboone_full_services
}
services.user.BackTracker: @local::microboone_backtracker

#Start each new event with an empty event.
source:
{
  module_ty
  timestamp
  maxEvents:    5          # Number of events to create
  firstRun:     1          # Run number to use for this file
  firstEvent:   1          # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

Defined in uboonecode/uboone/EventGenerator/GENIE/genie_microboone.fcl

```
 analyzers:
 {
   largana:   @local::microboone_largeantana
 }

 # define the producer and filter modules for this path
 # filters reject all following items.  see lines starting
 # physics.producers below
 simulate: [ generator, largeant, backtrack, optdigitizer,
             optfem, triggersim, optreadout, daq ]
 analyzeIt:  [ largana ]
 # define the output stream, there could be more than one
 # if using filters
 stream1:  [ out1 ]

 # trigger_paths is a keyword and contains the paths that
 # modify the art::event,
 #ie filters and producers
 trigger_paths: [simulate]

 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [analyzeIt, stream1]

#block to define where the output goes.  If you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

```
microboone_genie_simple:              @local::microboone_genie
microboone_genie_simple.FluxType:       "simple_flux"
microboone_genie_simple.FluxFiles:
["uboonebeam/bnb_gsimple_fluxes_02.28.2014_470/gsimple_microboone-470-onaxis_mc_nu_dummy_ntrd_*.root"]
microboone_genie_simple.EventsPerSpill: 0
microboone_genie_simple.POTPerSpill: 5e12

microboone_genie:                     @local::standard_genie
microboone_genie.BeamName:            "booster"
microboone_genie.GlobalTimeOffset:   1.6e6             #microboone reads out 1.6ms before the spill
```

```
th
rting
izer,
one
that
```

Defined in uboonecode/uboone/EventGenerator/GENIE/genie_microboone.fcl

```
#Start each new event with an empty event
source:
{
  module_ty
  timestamp
  maxEvents:   5          # Number of events to create
  firstRun:    1          # Run number to use for this file
  firstEvent:  1          # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

```
 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [analyzeIt, stream1]

#block to define where the output goes.  if you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

```
microboone_genie_simple:                @local::microboone_genie
microboone_genie_simple.FluxType:       "simple_flux"
microboone_genie_simple.FluxFiles:
["uboonebeam/bnb_gsimple_fluxes_02.28.2014_470/gsimple_microboone-470-onaxis_mc_nu_dummy_ntrd_*.root"]
microboone_genie_simple.EventsPerSpill: 0
microboone_genie_simple.POTPerSpill: 5e12
```

In larsim/EventGenerator/GENIE/genie.fcl

```
microboone_genie:                 @local::standard_genie
microboone_genie.BeamName:             "booster"
microboone_genie.GlobalTimeOffset:   1.6e6             #microboone reads out 1.6ms before the spill
```

```
th
rting

izer,

one

that
```

```
#Start each new event with an empty event
source:
{
  module_ty
  timestamp
  maxEvents:   5            # Number of events to create
  firstRun:    1            # Run number to use for this file
  firstEvent:  1            # number of first event in the file
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

```
 # end_paths is a keyword and contains the paths that do
 # not modify the art::Event,
 end_paths:     [analyzeIt, stream1]

#block to define where the output goes.  If you defined a
# filter in the physics block and put it in the
# trigger_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

Defined in uboonecode/uboone/EventGenerator/GENIE/genie_microboone.fcl

uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Event generator example 2: GENIE

```
#include "ervices_microboone.fcl"
#include "genie_microboone.fcl"
#include  "rgeantmodules_microboone.fcl"
#include  “tsimmodules_microboone.fcl"
#include  “triggersim_microboone.fcl"
#include  ”opticaldetectorsim_microboone.fcl"
#include  “mccheatermodules.fcl"

process_name: GenieGen

services:
{
  # Load th
  TFileServ
  Timing:
  RandomNum
  user:
}
services.us

#Start each
source:
{
  module_ty
  timestamp
  maxEvents
  firstRun:
  firstEven
}

# Define and configure some modules to do work on each event.
physics:
{

 producers:
 {
   generator:    @local::microboone_genie_simple
   largeant:     @local::microboone_largeant
   backtrack:    @local::standard_backtrackerloader
   optdigitizer: @local::microboone_optical_adc_sim
   optfem:       @local::microboone_optical_fem_sim
   triggersim:   @local::ubtrigger_singlep
   optreadout:   @local::microboone_optical_dram_readout_sim
   daq:          @local::microboone_simwire
 }
```

```
 analyzers:
 {
   largana:   @local::microboone_largeantana
 }

 # define the producer and filter modules for this path
 # filters reject all following items.  see lines starting
 # physics.producers below
 simulate: [ generator, largeant, backtrack, optdigitizer,
             optfem, triggersim, optreadout, daq ]
 analyzeIt:  [ largana ]
 # define the output stream, there could be more than one
          ing filters
           :  [ out1 ]

          er_paths is a keyword and contains the paths that
          / the art::event,
          ers and producers
          _paths: [simulate]

          aths is a keyword and contains the paths that do
          odify the art::Event,
          ns:     [analyzeIt, stream1]

          o define where the output goes.  if you defined a
           in the physics block and put it in the
          r_paths then you need to put a
# SelectEvents: {SelectEvents: [XXX]} entry in the output
# stream you want those to go to, where XXX is the label
# of the filter module(s)
outputs:
{
 out1:
 {
   module_type: RootOutput
   fileName:    "genie_gen_uboone.root" #default file name,
 }
}
```

The event generation and simulation workflow:
- GENIEGen module
- LArG4 module
- UBOpticalADCSim module
- OpticalFEM module
- Trigger simulation (did not talk about this...)
- OpticalDRAMreadout module
- SimWireMicroBooNE module

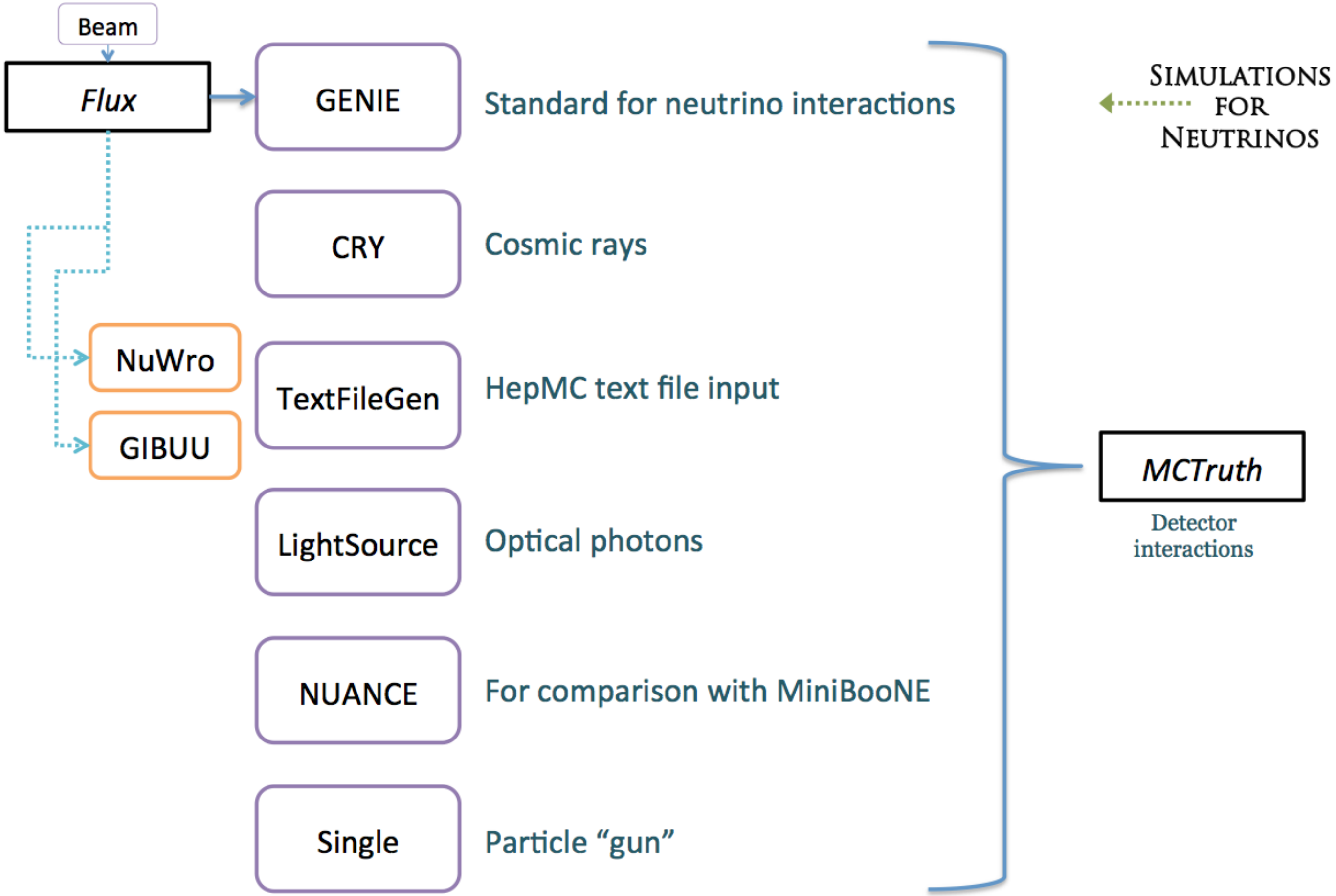
uboonecode/uboone/EventGenerator/prodgenie_uboone.fcl

Summary

- Provided a broad overview of event generation and simulation infrastructure in LArSoft
- Introduced geometric and other classes needed to specify detector properties using shared interfaces
- Showed two examples of running event generators and simulation within LArSoft

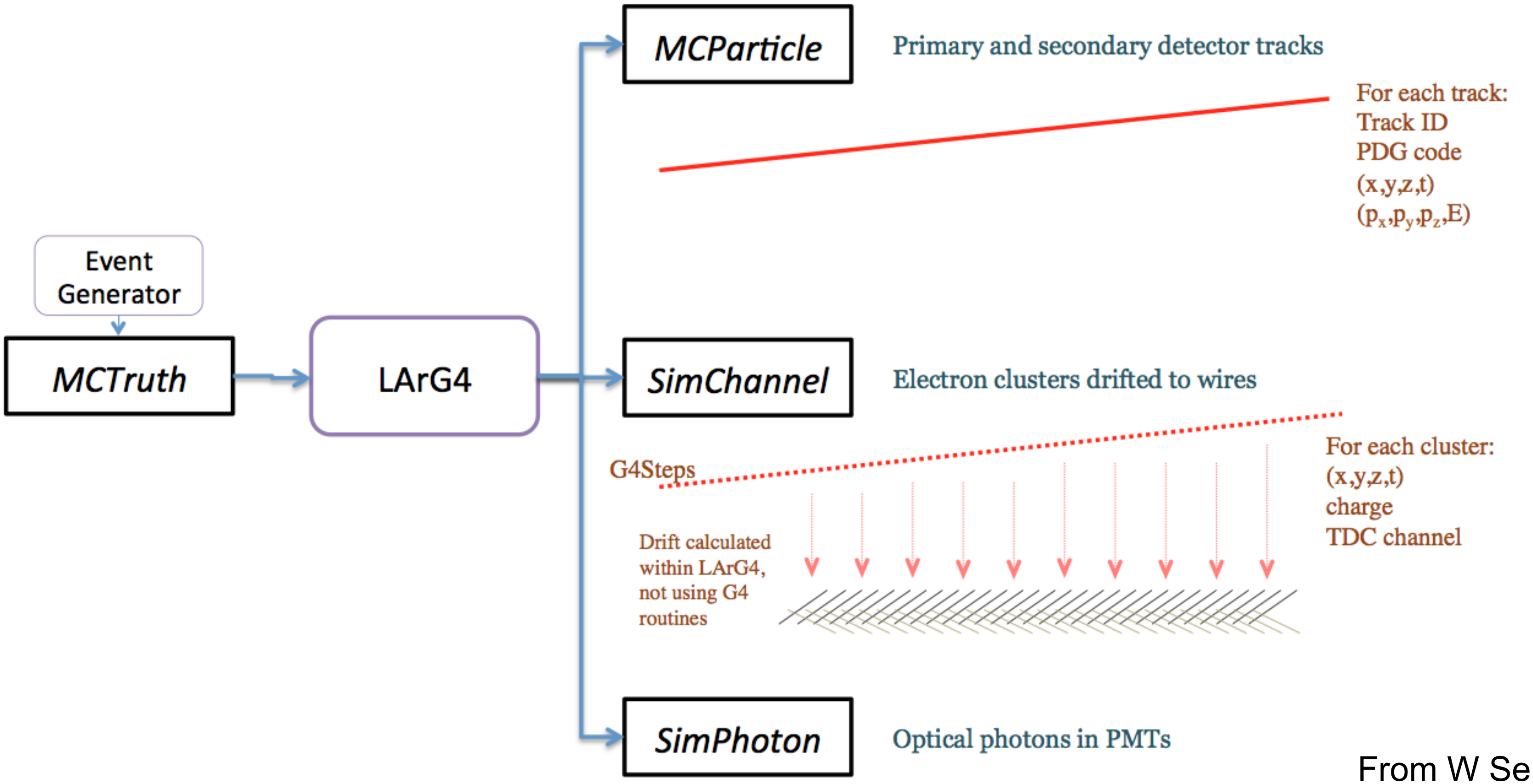
Backup

Event generators



From W Seligman

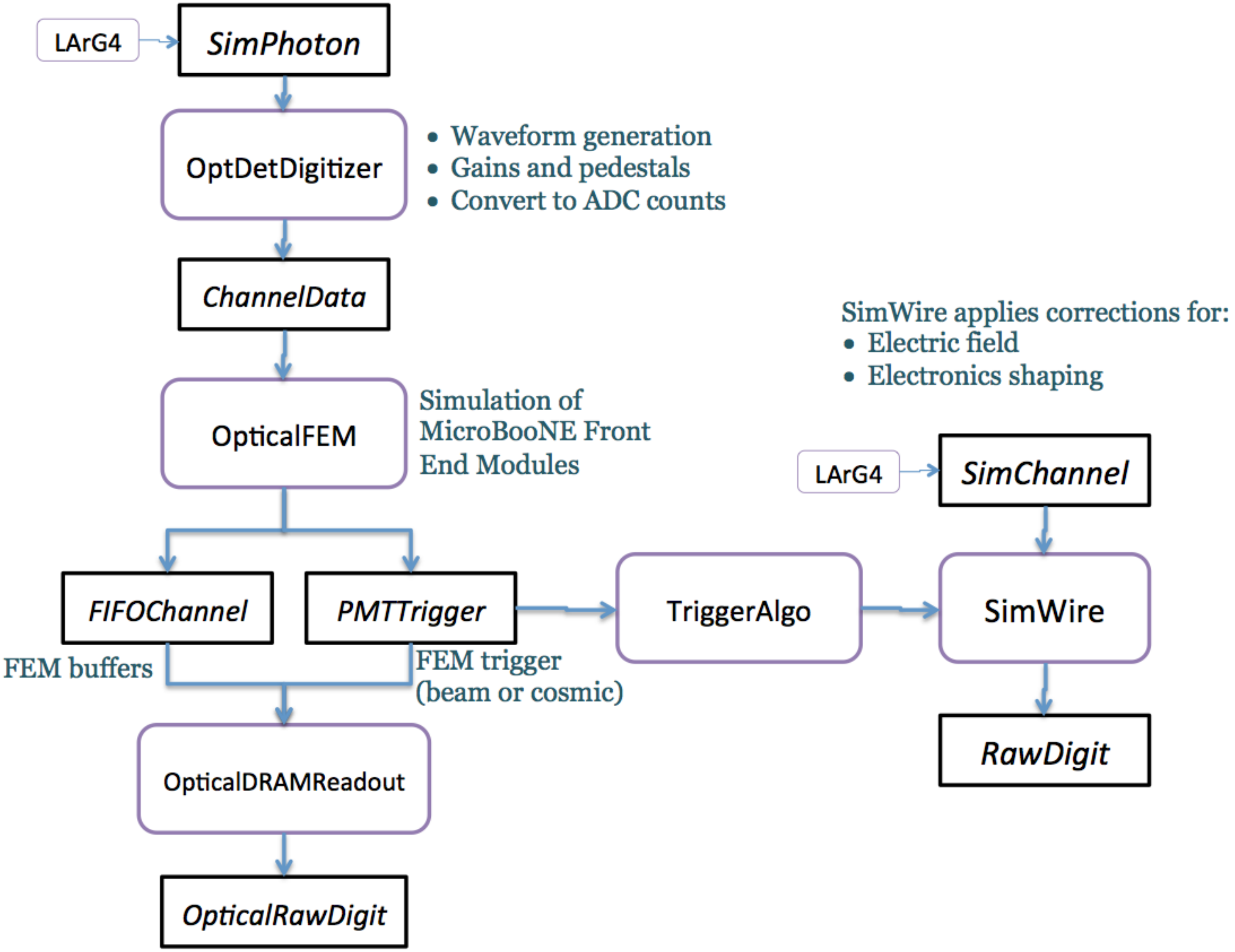
Detector simulation



From W Seligman

Simulation task workflow

Detector response and digitization



- Waveform generation
- Gains and pedestals
- Convert to ADC counts

Simulation of MicroBooNE Front End Modules

FEM buffers

FEM trigger (beam or cosmic)

SimWire applies corrections for:

- Electric field
- Electronics shaping