Simulação da interação de neutrinos em LArTPCs
Parte 1*

Laura Paulucci
ITA

*Material baseado em LArSoft TPC Simulation, por Anyssa Navrer-Agasson, 1st DUNE LArSoft Workshop, CERN, 2025

Para que serve esta parte da escola?

- Proposta: apresentar como é feita a simulação de interações em uma LArTPC (teoria e prática)
  - Quais os passos para gerar eventos*?
  - Quais as ferramentas utilizadas em cada passo?
  - Como as diferentes partes da simulação se integram?
  - O que é gerado em cada passo?

Saber como os eventos são simulados ajudará a entender o porquê da reconstrução seguir os passos que segue

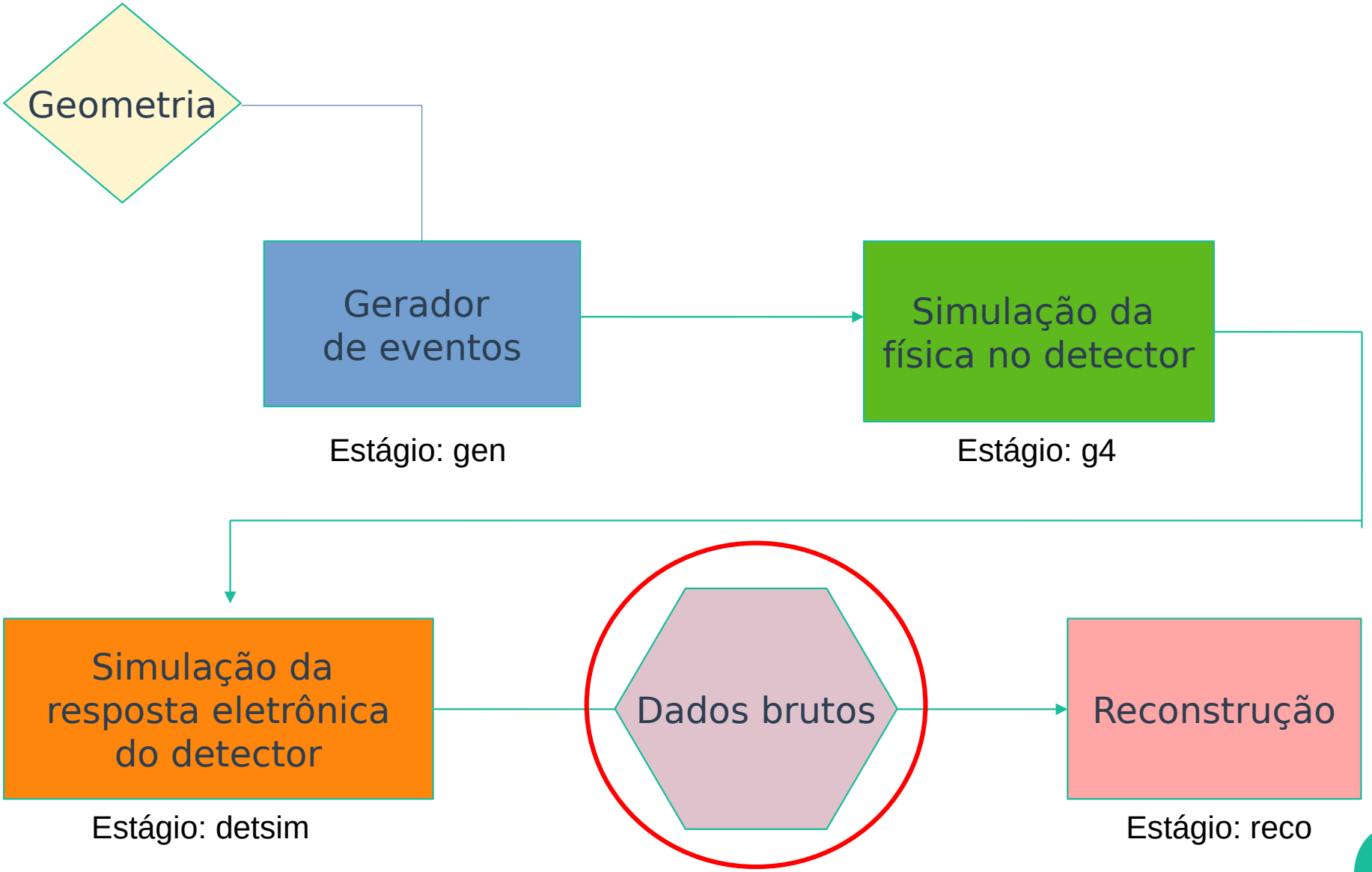
Para que serve esta parte da escola?

Tom Junk

- What is an event?
  - Physicists: A location in spacetime: (t,x,y,z)
  - Physicists: An interaction between particles
  - Statisticians: A sample drawn from a sample space. Your entire dataset is an "event"
  - DAQ people: A triggered readout of a detector ("trigger record")
  - The smallest amount of data *art* can process.

*Evento: a interação de uma partícula primária e tudo o que vier depois gerando sinal nos detectores dentro de um intervalo de tempo pré-determinado

Cadeia de simulação

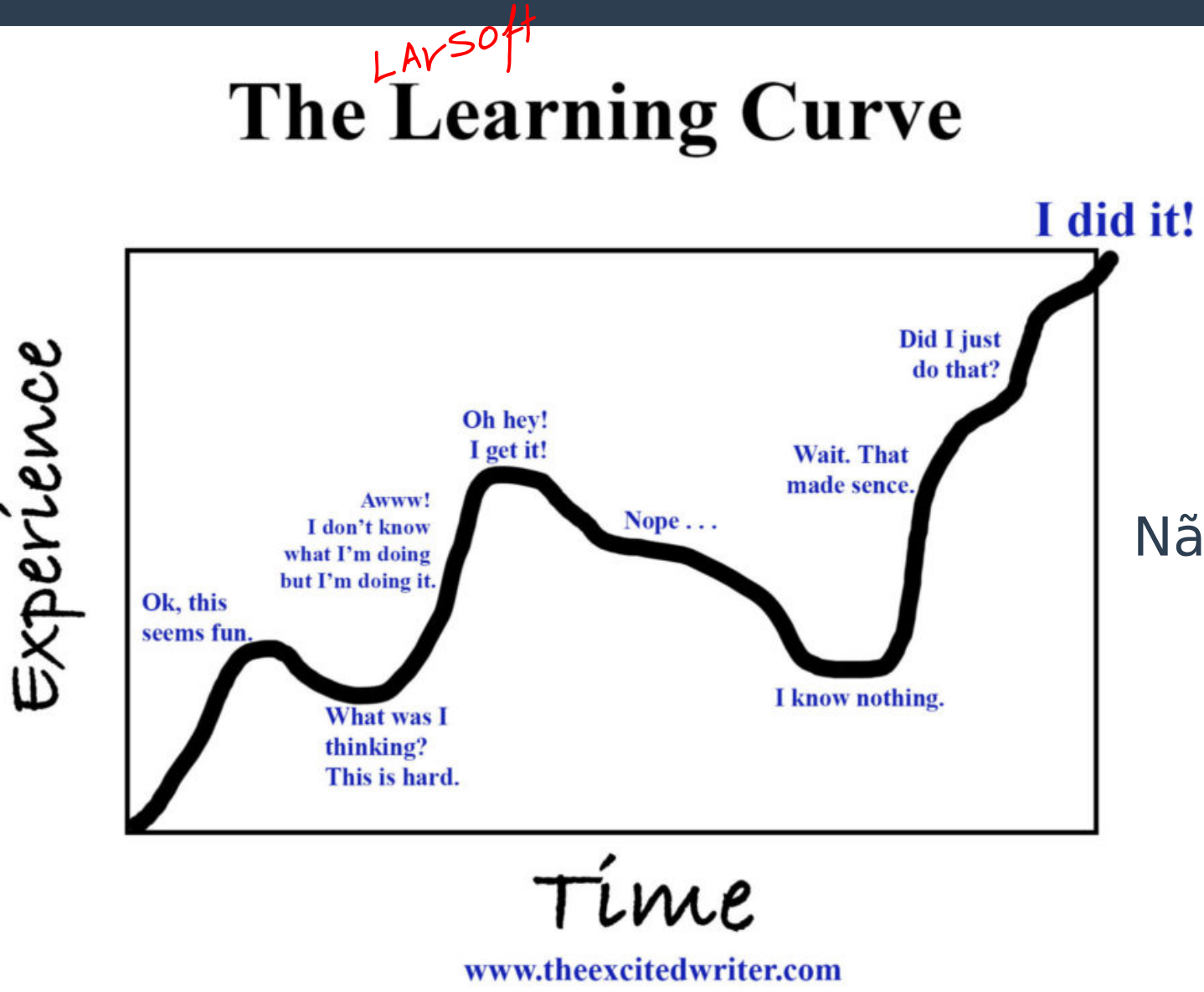


(O famigerado) Liquid Argon Software - LArSoft

- Software base para a simulação e reconstrução em experimentos de neutrinos que usam LArTPC
  - Baseado na framework chamada *art** (Analysis, Reconstruction, and Trigger) desenvolvida pela divisão de Scientific Computing do Fermilab
  - Lê e produz **data products**
  - Arquivos de saída são do tipo *artROOT*
    - Não é o mesmo que arquivos ROOT! Possui data products, não NTuples/TTrees

*Será substituída por *Phlex* (Parallel, hierarchical, and layered execution of data) “em breve”

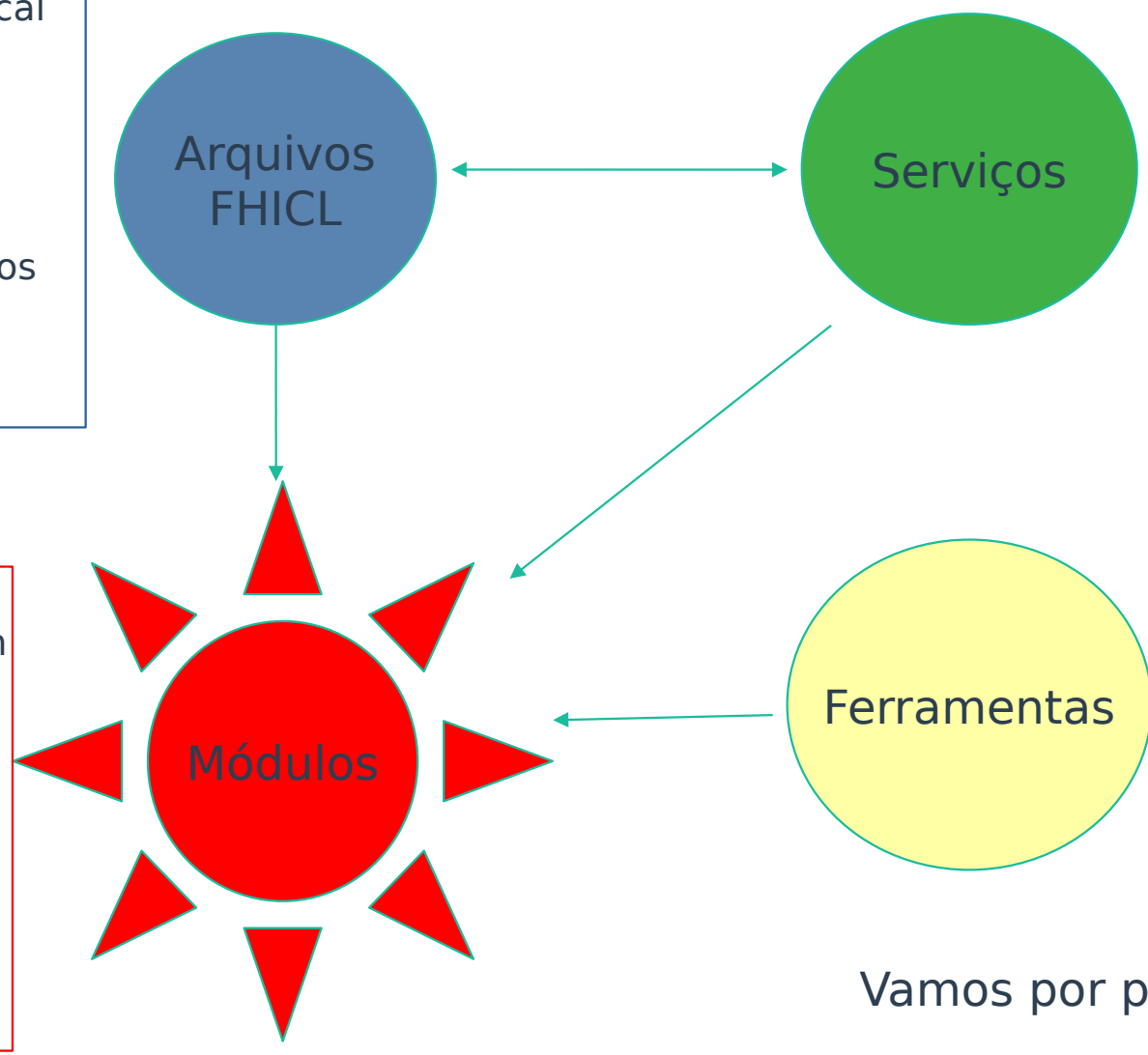
(O famigerado) Liquid Argon Software - LArSoft



Não desista!

LArSoft: organização

Fermilab **Hi**erarchical **C**onfiguration **L**anguage)
- Um por módulo, usado para configuração
  - Lista os serviços usados
  - Define valores para variáveis



Acessados pelos módulos
- Fornecem info sobre
  - Geometria
  - Propriedades físicas

Algoritmos
- Organizados com fases de inicialização, input, processamento de eventos e output
- Lê e cria data products do LArSoft

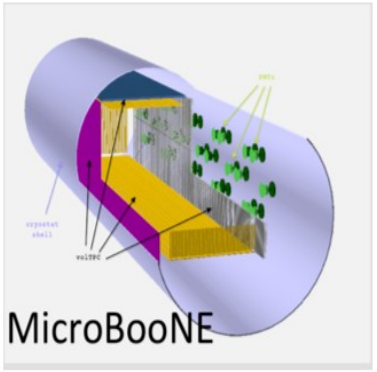
Utilitários configuráveis chamadas dentro dos módulos

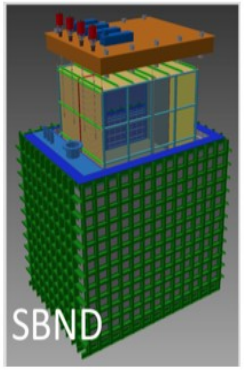
Vamos por partes...

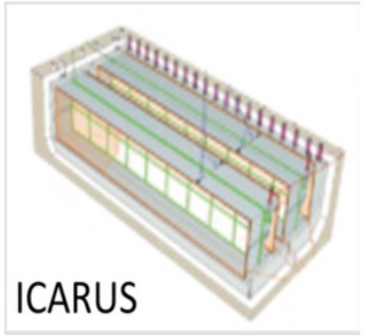
Geometria

Construindo um detector

- Cada experimento precisa definir a sua geometria
- Geometrias são escritas em GDML (Geometry Description Markup Language)
  - LArSoft tem requerimentos específicos para definição de alguns elementos-chave
  - Detectores possuem duas versões, com fios e sem (_nowires)
    - Primeira usada para determinar a posição dos fios e suas propriedades (ângulos, pitch)
    - A segunda é de fato usada na simulação (economiza tempo e memória)



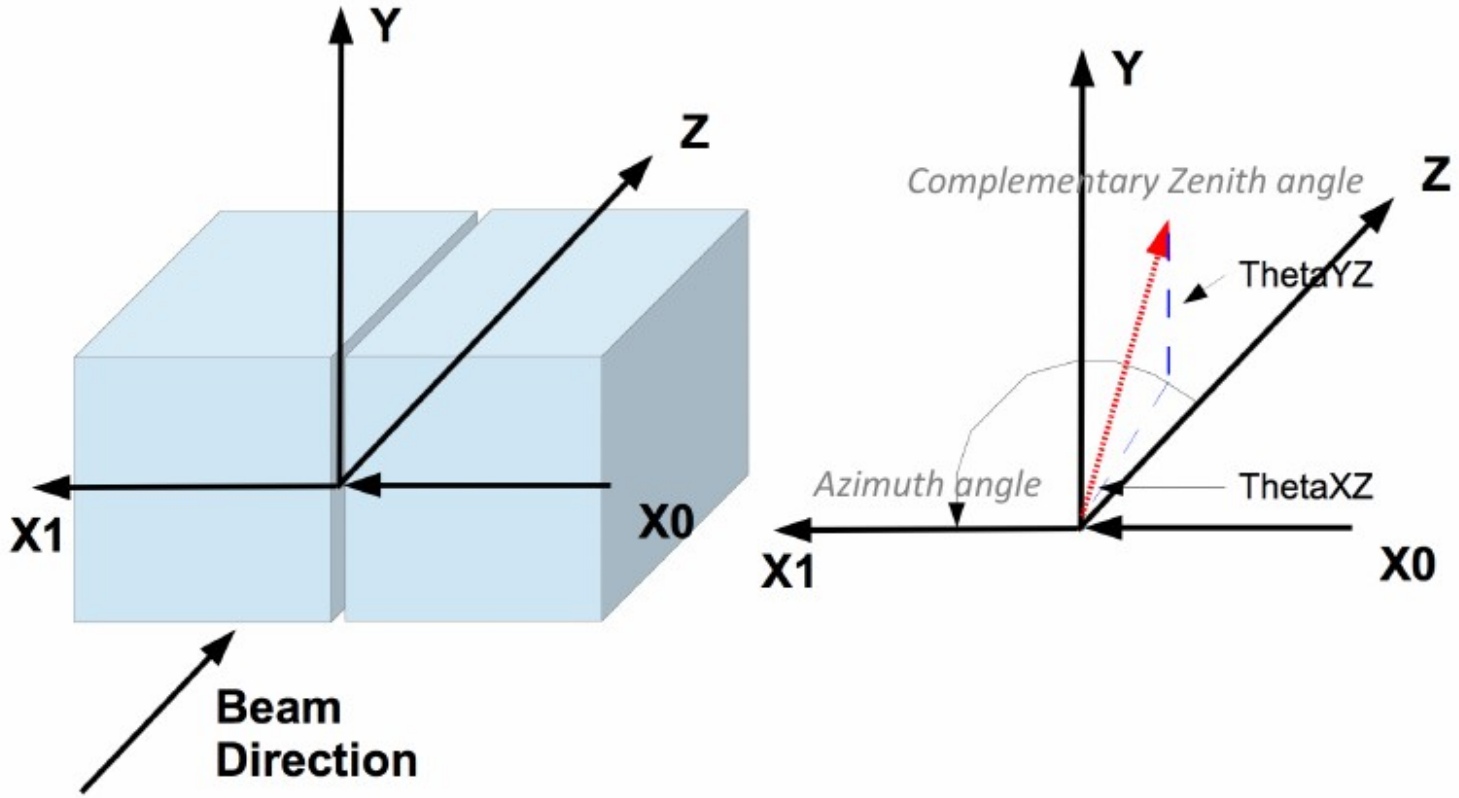




Construindo um detector

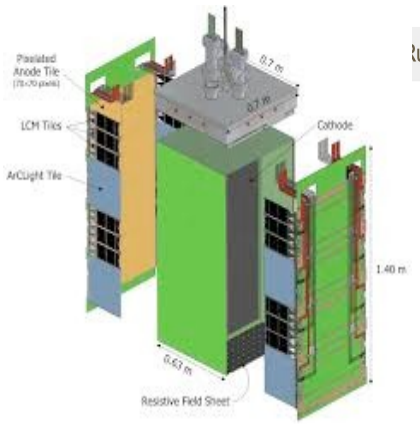
Sistema de coordenadas

A origem do sistema depende do experimento

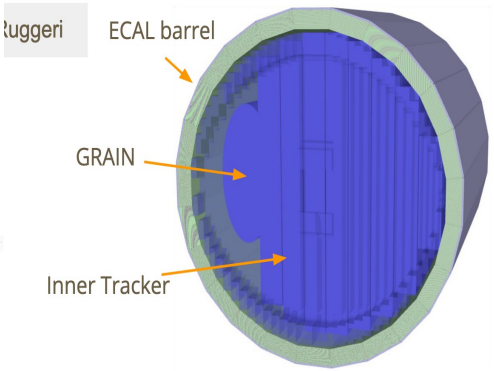


Para todos os detectores: Z aumenta ao longo da direção do feixe, Y aumenta para longe do centro da Terra e o sentido de X é dado para formar um sistema de mão direita

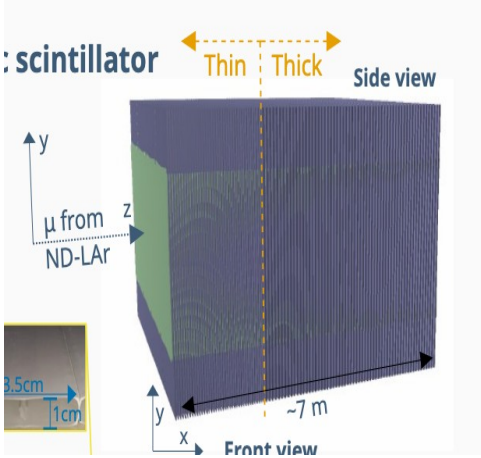
Geometrias no DUNE



ND LAr

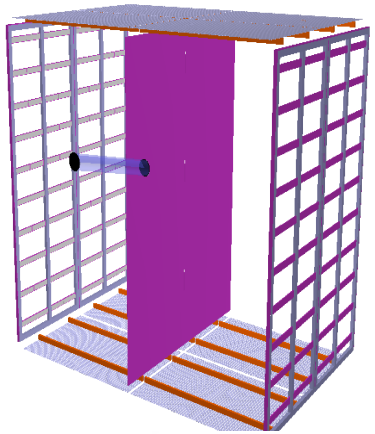


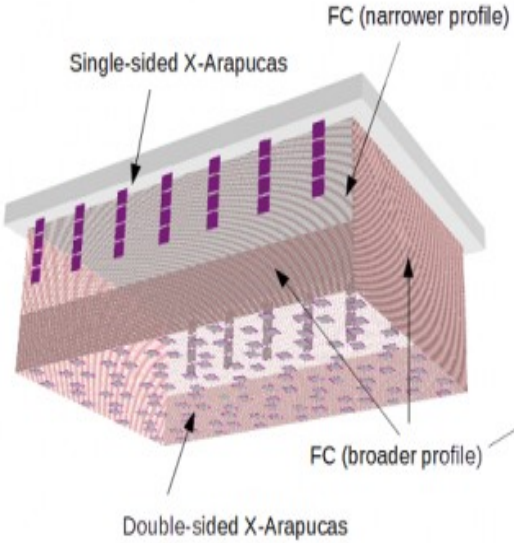
SAND



TMS

ProtoDUNE – Horizontal Drift





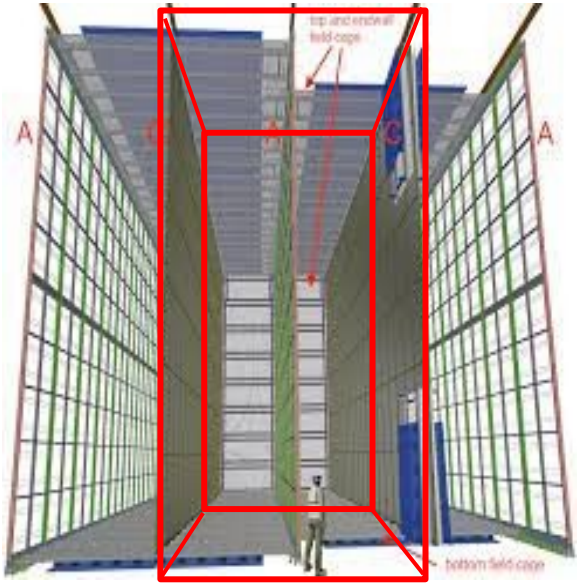
Far Detector – Vertical Drift

- Far Detectors e ProtoDUNE → LArSoft
- Near Detector → NÃO É LArSoft

Geometria

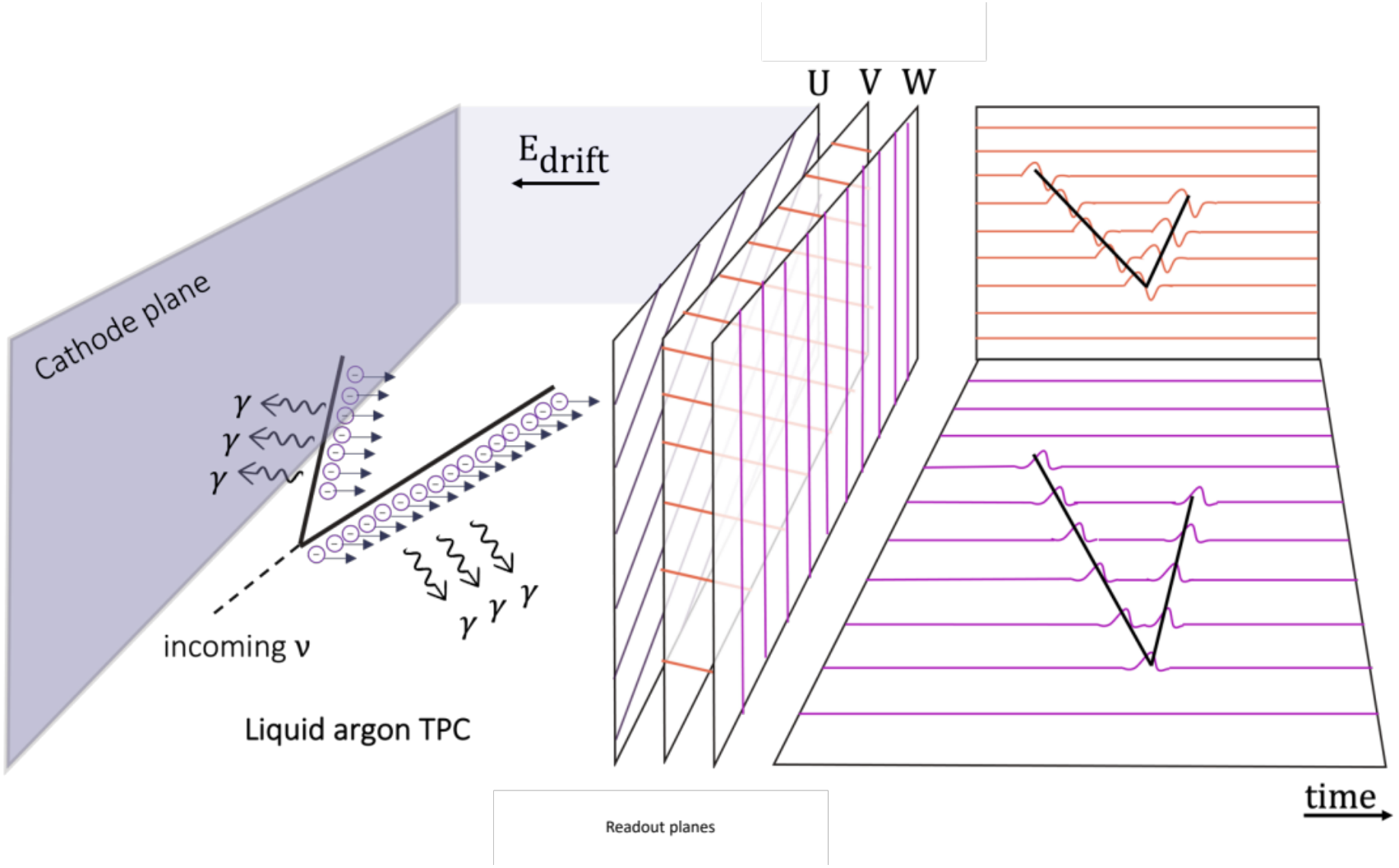
Far Detector Horizontal Drift

- Geometria reduzida: 1x2x6
  - 1 plano de APA de largura, 2 APAs de altura, 6 APAs de comprimento
  - Volume central
- Até o momento, todos os estudos no FD-HD feitos nessa geo*



*A geo do FD-HD com 10 kton está pronta e uma produção de MC prevista já já!

Liquid Argon Time-Projection Chamber



Gerador de eventos

Criando interações de partículas

- A simulação no LArSoft começa com a geração de eventos
- Diferentes geradores encontrados em larsim/EventGenerator
  - Partículas individualmente (SingleGen)
  - Interações de neutrinos (GENIE)
  - Raios cósmicos (CORSIKA)
  - Neutrinos de MeV (MARLEY)
  - Radiológicos (RadioGen)
  - Lidos de um arquivo texto (TextFileGen)

É possível combinar geradores para criar eventos mais complexos

Gerador de eventos

GENIE

https://genie-mc.github.io/index.html

- O mais popular gerador da interação de neutrinos no mercado
- Recebe arquivos de fluxo e a especificação de onde os neutrinos devem interagir (volWorld, volTPC...)
- Usa o fluxo de neutrinos adequado para cada experimento
- Gera os produtos da interação de neutrinos (secundários) usando modelagem de processos estado-da-arte
  - É possível especificar os tipos de interação a serem considerados (CCQE, MEC, DIS...)
- Permite obter a exposição (em POT - Protons on Target) da amostra
- Flexível para gerar neutrinos de interações BSM

Gerador de eventos

Single Particle Gun

- É o gerador mais simples disponível
- Gera partículas individuais ou interações simples
- Opção para gerar múltiplas partículas
- Escolher uma aleatoriamente de uma lista por evento
- Gerar mais de uma partícula por evento
- Parâmetros de configuração (FHICL): tipo de partícula, posição, momento e como variam (distribuição uniforme, gaussiana)

Estágio gen: saída

- Primeiro estágio da cadeia (em geral não tem arquivo de entrada)
- Objetos do tipo simb::MCTruth (um por gerador utilizado) que contém:
  - Informação do gerador
  - Lista de partículas (simb::MCParticle) com código PDG, posição, momento, ...
  - Informação sobre a interação do neutrino (se houver)

Simulação da física no detector

Estágio g4

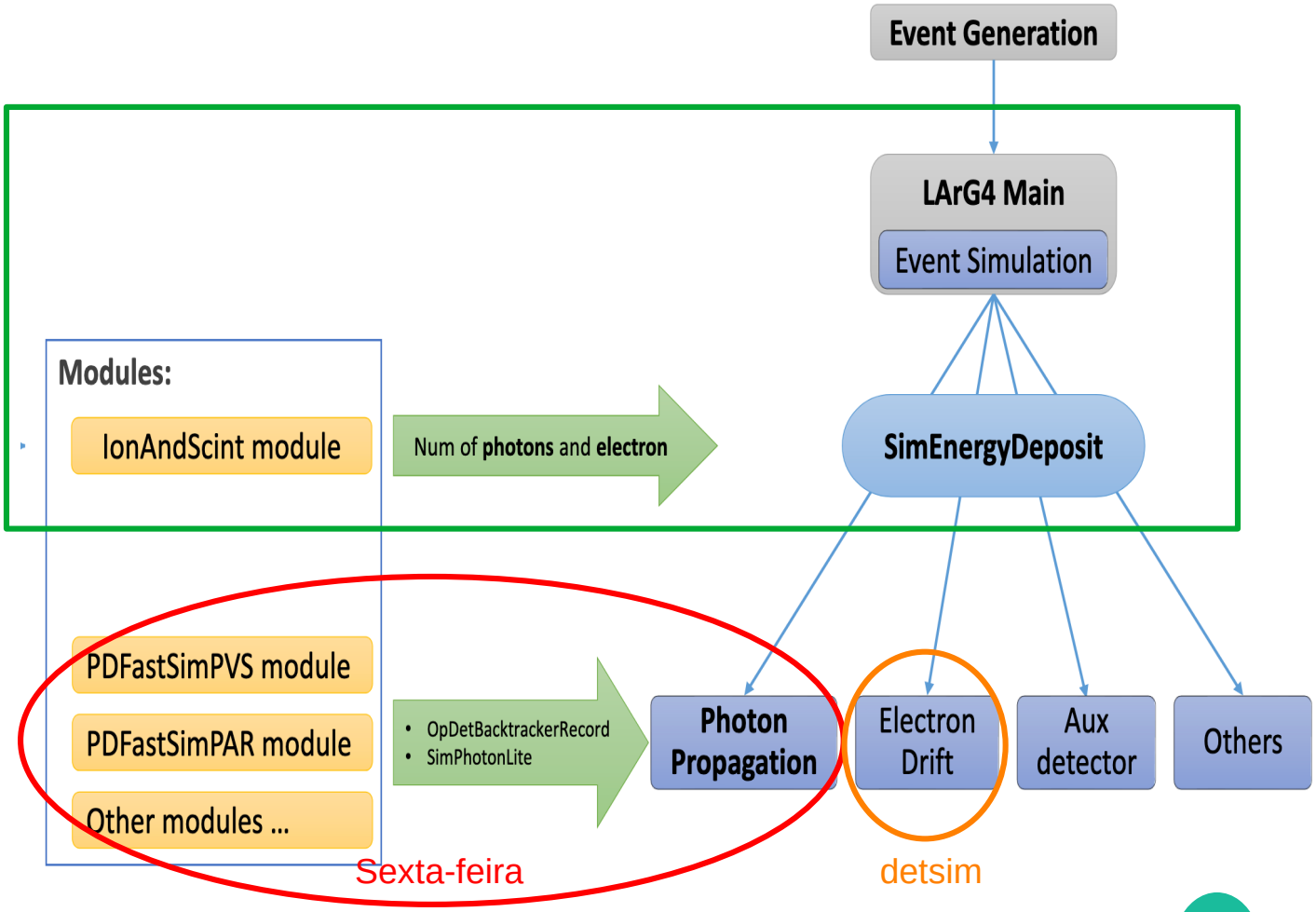
Arquivo de entrada é a saída do estágio gen

- Responsável pela propagação e interações das partículas gerados no estágio gen no interior do detector
  - Registra os depósitos de energia ao longo do traço das partículas
  - Cria partículas secundárias provenientes das interações
  - Registra tudo em um objeto para futuramente obter a verdade de Monte Carlo (BackTrackers)

Como???

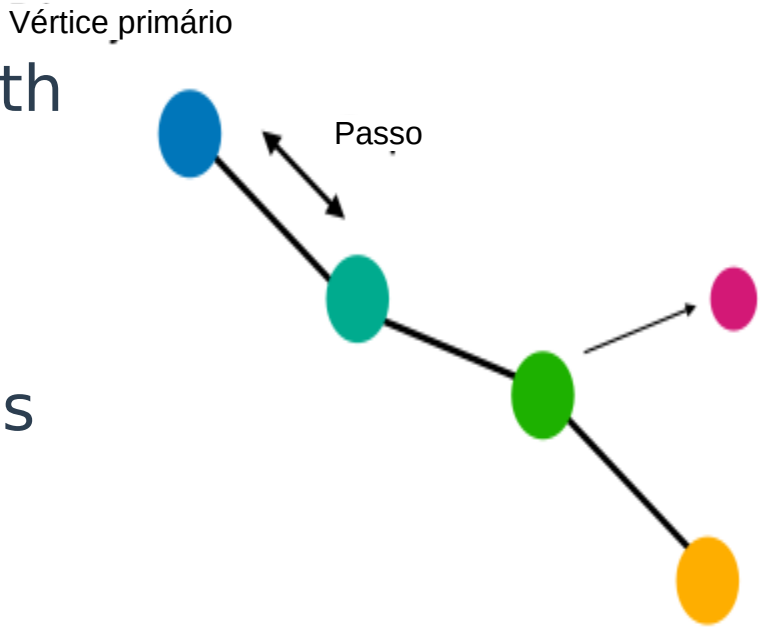
Estágio g4

- A propagação e criação de secundários é feita pelo LArG4
- Usamos o Geant4 através de uma interface art/ Geant4 chamada *artg4tk*



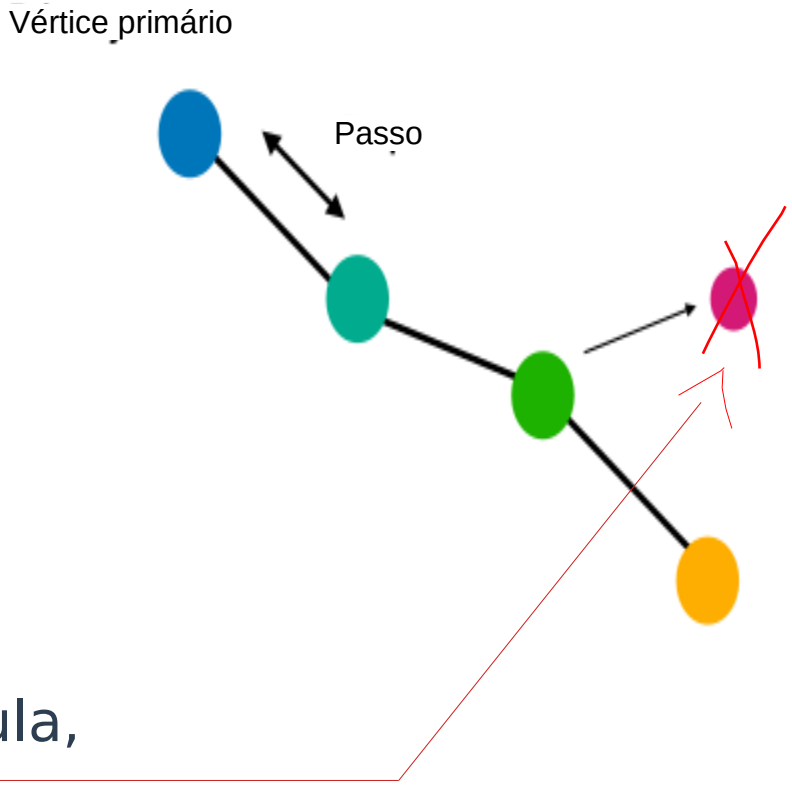
Estágio g4: LArG4

- Para cada partícula no objeto MCTruth
  - Gera o vértice de interação do primário
  - Acompanha a partícula em passos
  - Para cada passo
    - Avalia a deposição de energia
    - Determina os processos físicos que ocorrem
    - Adiciona as partículas secundárias criadas (se houver) à lista de partículas



Estágio g4: LArG4

- É configurado via arquivo fhicl
  - Lista de física (quais processos serão considerados?)
  - Parâmetros físicos adicionais
  - Quais partículas acompanhar na simulação
  - ...
- É possível incluir cortes no tipo de partícula, na energia...
- Os depósitos de energia são usados no módulo *IonAndScint* para obter o número de elétrons de ionização e fótons de cintilação por depósito



Estágio g4: IonAndScint

- Responsável por traduzir E depositada em elétrons de ionização e fótons de cintilação
- Carga: Modelo de Birks, Box modificado ou LArQL
  - LArQL adicional a dependência adicional com dE/dx e campo E
- Luz: Método Separated ou LArQL → mais na sexta-feira

$$\frac{dQ}{dx} = \left(\frac{A_B}{1 + \frac{k_B}{\xi \rho_{LAr}} \frac{dE}{dx}} + \chi_0 \left(dE/dx \right) f_{corr} \left(\xi, dE/dx \right) \right) \frac{1}{W_{ion}} \frac{dE}{dx}$$

Estágio g4: Saída

- Objeto simb::MCTruth do estágio gen
- Nova coleção de simb::MCParticle contendo as partículas secundárias geradas neste estágio
- Coleção de sim::SimEnergyDeposit contendo os depósitos de energia
  - Posição, tempo, fótons de cintilação e elétrons de ionização por depósito
- Um objeto sim::ParticleAncestryMap contendo a ancestralidade das partículas
- Associações entre MCTruth, MCParticle e GeneratedParticleInfo
- Objeto SimPhotons/SimPhotonsLite → sexta-feira

Simulação da resposta eletrônica do detector

Estágio detsim

Arquivo de entrada é a saída do estágio g4

- Faz a deriva dos elétrons de ionização até os planos de leitura de carga
  - Transforma a informação física dos elétrons em uma resposta digitalizada do detector
  - Inclui a simulação da forma do sinal e ruído eletrônico
  - Gera dados simulados equivalentes ao que é obtido do experimento real
- Faz o mesmo para os fótons de cintilação → sexta-feira

Simulação da resposta eletrônica do detector

Estágio detsim: wirecell

https://wirecell.bnl.gov/

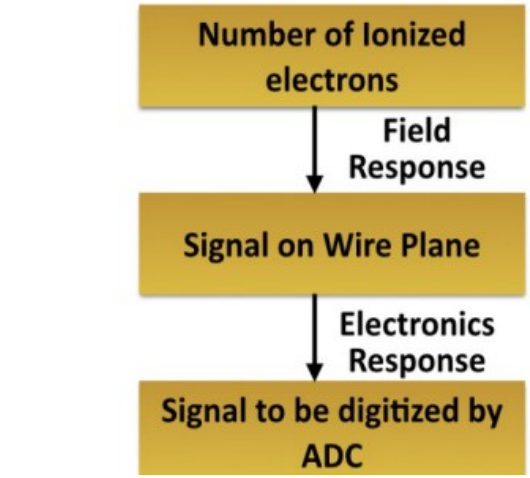
- Impurezas absorvem carga
  - Lifetime do e- em LAr
- Difusão (transversal e longitudinal) espalha espacialmente a carga

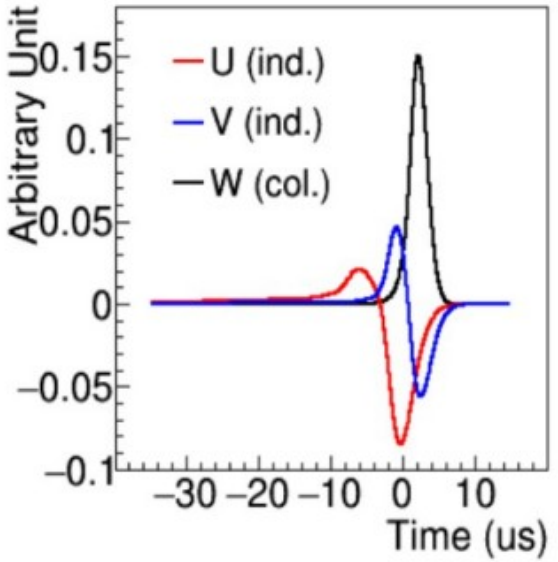
$$\sigma^2 = 2\,Dt$$

- Flutuações na absorção de elétrons

<Energy deposit>
* x, y, z, t, # of e

Para um drift de 3,6m e E=500 V/m, difusão longitudinal de 1,8mm e transversal de 2,5mm



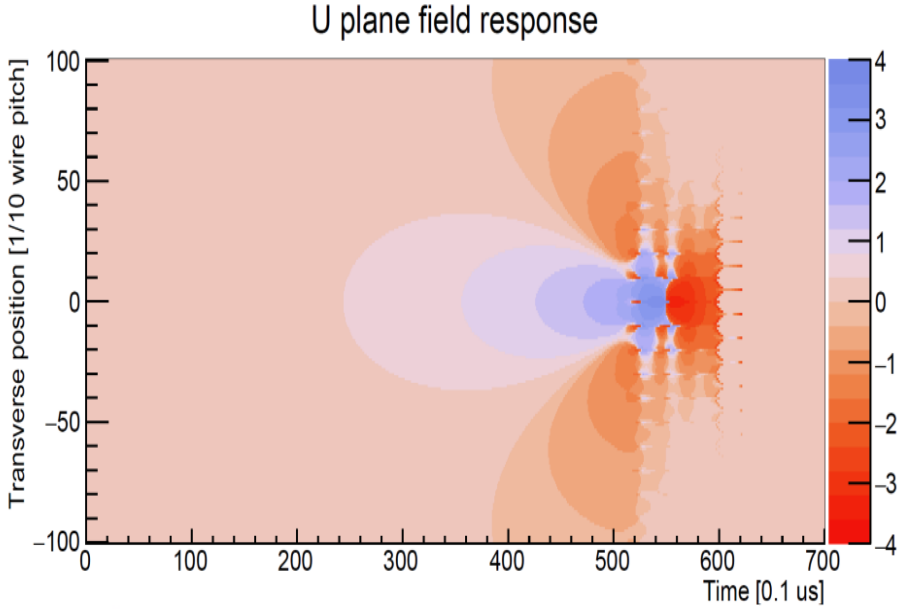


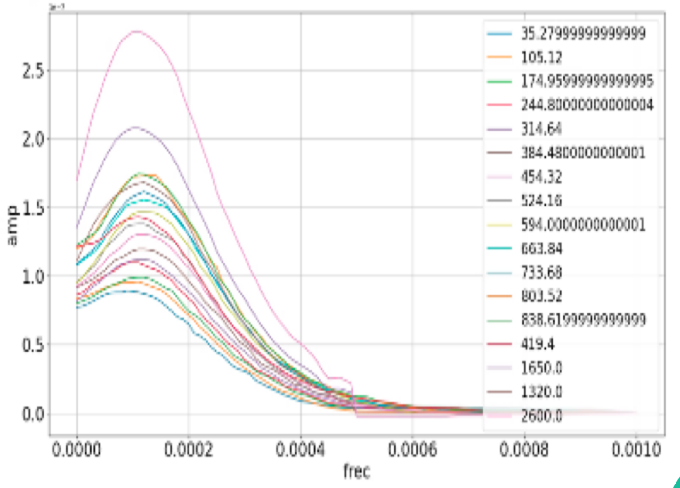
Estágio detsim: wirecell

Wire-Cell

https://wirecell.bnl.gov/

- Os elétrons induzem sinal não só no fio/strip mais próximo mas também nos vizinhos
- Wirecell provê
  - Simulação de deriva 2D, incluindo esse efeito
  - Processamento do sinal 2D, desemaranhando este efeito
  - Modelo de ruído eletrônico que leva em conta o tamanho dos fios





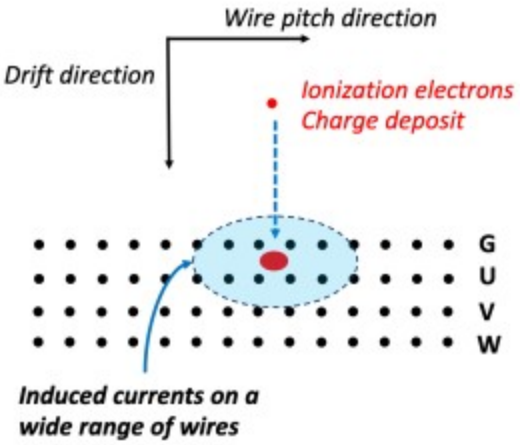
Simulação da resposta eletrônica do detector

Estágio detsim: wirecell

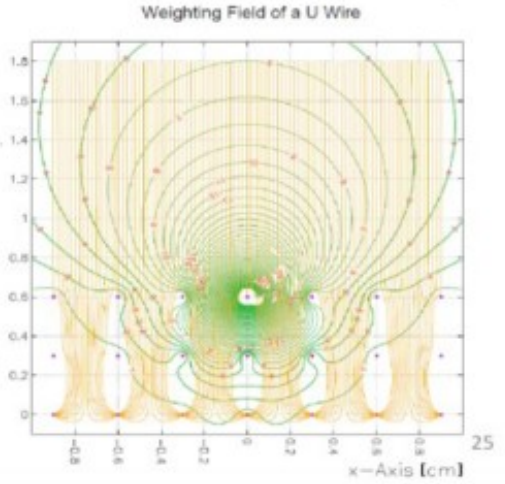
https://wirecell.bnl.gov/

- Para a deriva dos elétrons e field response
  - Calcula a corrente induzida de acordo com o depósito de energia
  - Leva em conta efeitos de longo alcance e constrói a field response a partir de uma simulação 2D no Garfield
  - Considera difusão, a lifetime do elétron...
- A digitalização do sinal simulado é uma convolução da distribuição dos elétrons, resposta do campo e resposta da eletrônica (acoplamento AC, pré-amplificador, ruído...)

→ raw waveform 



2D GARFIELD simulation : The field extends beyond a single wire region



Simulação da resposta eletrônica do detector

Estágio detsim: wirecell

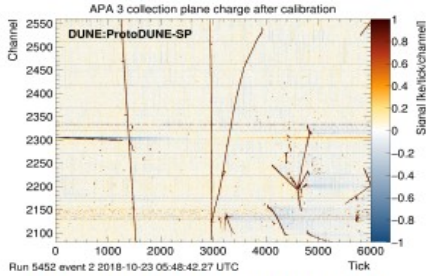
https://wirecell.bnl.gov/

- Wirecell também faz o processamento do sinal!
- A partir dos dados brutos, queremos extrair os efeitos das respostas do campo e da eletrônica de modo a recuperar a informação da carga

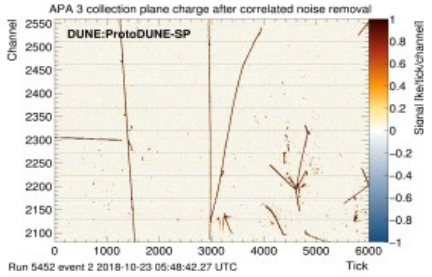
  → carga versus tempo

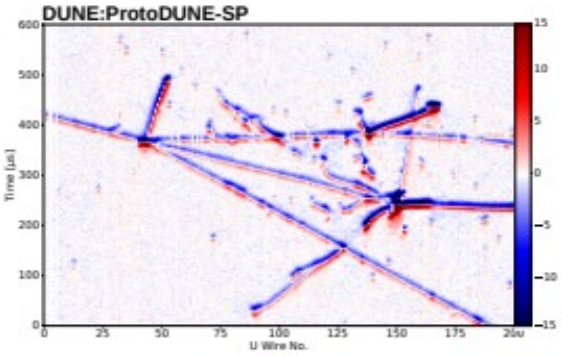
- Para isso, wirecell
  - Filtra o ruído
  - Deconvolui o sinal

After pedestal subtraction/calibration

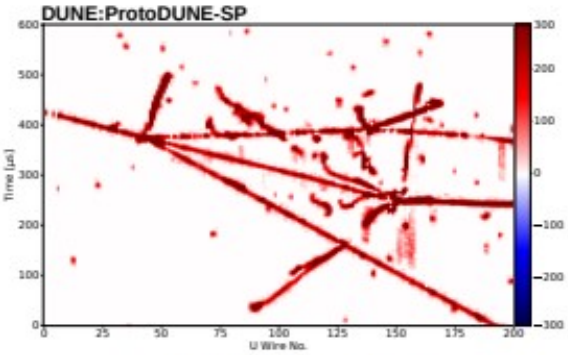


After ADC/time mitigation and tail/noise removal





(a) After Noise Filtering



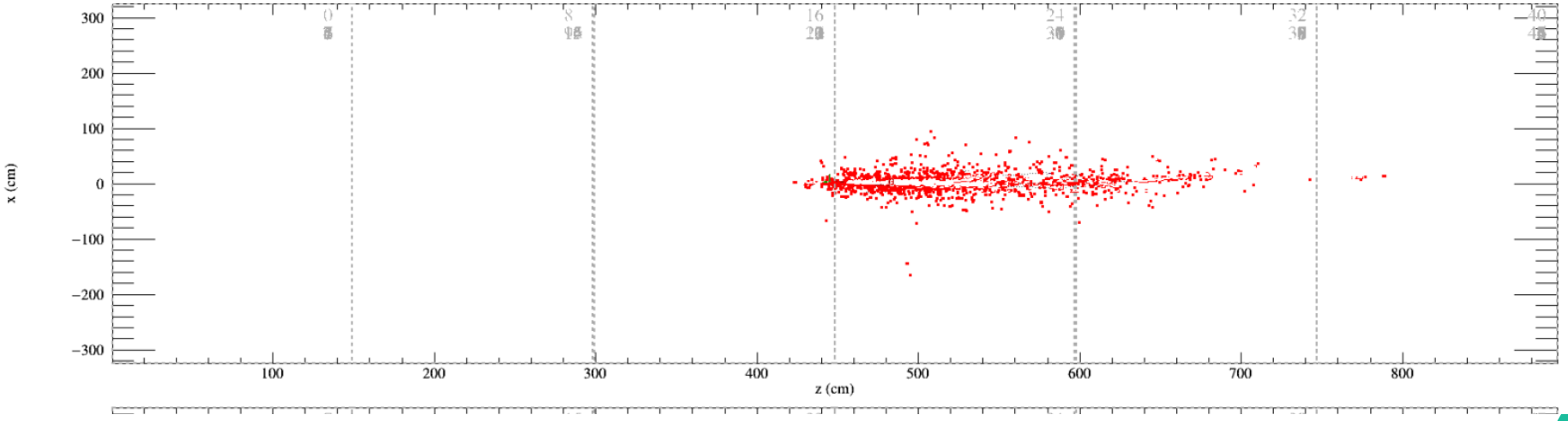
(b) After Deconvolution

Simulação da resposta eletrônica do detector

Estágio detsim: arquivo de saída

- Objetos dos estágios anteriores
- Coleção de recob::Wire com as waveforms simuladas para os diferentes canais das APAs/CRPs
- Coleção de raw::OpDetWaveform com as waveforms simuladas para os diferentes canais do PDS

Atenção: é possível salvar as raw:waveforms da carga, mas são grandes, portanto descartadas após o processamento como default



Resumo

- A simulação no LArSoft produz eventos que devem ser equivalentes aos dados raw do detector
  - Leva em conta a física dos processos e as respostas do detector
- A simulação no LArSoft tem muitos passos
  - Mas você sempre pode contar com os arquivos padrão produzidos e mantidos pela colaboração
- É modular e acomoda diferentes geometrias e processos
- Está em constante evolução!

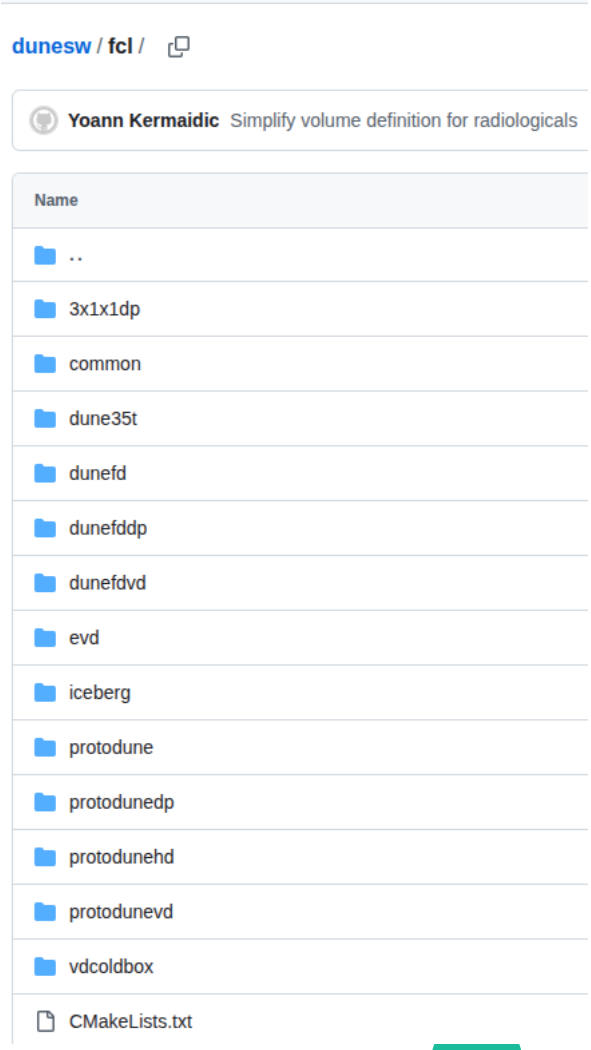
Mas como é que eu rodo de fato estes estágios?

- Através de arquivos fhicl!
- Eles dão as configurações para os diferentes estágios de simulação e reconstrução no LArSoft
  - Neles são definidos o que queremos rodar e como
- Por que ”Hierarchical” no nome?
  - FHiCLs podem herdar configuraçãoes de outros FhiCLs que podem herdar de outros FHiCLs que podem herdar...
  - Como o LArSoft tem sua estrutura em C++, que é uma linguagem orientada a objetos, parâmetros podem ser herdados de outros arquivos/configurações

E onde eu encontro os arquivos fhicl?

- FHICL de configuração de módulos → junto dos módulos!
  - Podem estar em diferentes repositórios do DUNE, se forem específicos
  - Podem estar nos diferentes repositórios do LArSoft, se forem mais gerais
- Os arquivos padrão para rodar o código do DUNE estão no dunesw

github.com/DUNE/dunesw/tree/develop/fcl



Arquivos fhicl: um exemplo

dunesw / fcl / dunefdvd / gen / background / **prodbackground_radiological_decay0_dunevd10kt_1x8x14.fcl**

lpaulucc LE reco2/bg gen adjustments

Code | Blame | 17 lines (13 loc) · 430 Bytes

```
1   #include "prodbackground_radiological_decay0_common_dunevd10kt.fcl"
2
3   services:
4   {
5       @table::services
6       @table::dunefdvd_1x8x14_3view_30deg_simulation_services
7   }
8
9   physics.producers:
10     {
11        @table::physics.producers
12        @table::dunefd_vertdrift_1x8x14_producers
13     }
14
15  physics.simulate: [rns, @sequence::dunefdvd_backgrounds_1x8x14 ]
16
17  outputs.out1.fileName:    "prodradiological_decay0_dunevd10kt_1x8x14_gen.root"
```

Arquivos fhicl: um exemplo

dunesw / fcl / dunefdvd / gen / background / **prodbackground_radiological_decay0_common_dunevd10kt.fcl**

lpaulucc LE reco2/bg gen adjustments

Code | Blame | 49 lines (43 loc) · 1.25 KB

```
1  #include "services_dune.fcl"
2  #include "workflow_radiological_decay0_dunevd10kt.fcl"
```

dunesw / fcl / dunefdvd / gen / background / **workflow_radiological_decay0_dunevd10kt.fcl**

lpaulucc clean up in dunesim bkg model fhicls

Code | Blame | 158 lines (144 loc) · 7.06 KB

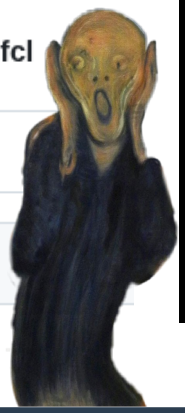
```
1  #include "dune_radiological_model_decay0_vd_1x8x6.fcl"
2  #include "dune_radiological_model_decay0_vd_1x8x14.fcl"
```

dunesim / dunesim / EventGenerator / Radiological / **dune_radiological_model_decay0_vd_1x8x14.fcl**

lpaulucc Config the same as original fhicls by Juergen

Code | Blame | 66 lines (46 loc) · 9.23 KB

```
1  #include "services_dune.fcl"
2  #include "dune_radiological_model_decay0_vd_1x8x6.fcl"
```



Uma ajudinha: findfcl.sh

- Script para encontrar um dado fhicl
  - Dará o caminho (path) para o fhicl
- Encontrar o fhicl pode ser útil para
  - entender onde um parâmetro em particular é definido
  - Saber quais fhicls você tem que incluir
  - …

Crie um novo arquivo e escreva isso nele

```
#!/bin/bash

if [ $# -ne 1 ]; then
  echo "Erro: favor passar um nome (ou expressão
regular) para o fcl"
  exit 1
fi
if [ -z ${FHICL_FILE_PATH+x} ]; then
  echo "Erro: FHICL_FILE_PATH nao definido!"
  exit 2
fi
SEARCH_PATHS=`echo $FHICL_FILE_PATH | sed
's/:/\n/g'`
for THIS_PATH in $SEARCH_PATHS; do
  if [ -d $THIS_PATH ]; then
    find $THIS_PATH -name $1
  fi
done
```

Encontrando nosso primeiro fhicl

- Vamos buscar um arquivo padrão que simula a geração de uma partícula no DUNE
- Para isso vá no terminal e digite

```
./findfcl.sh singles_dune.fcl
```

- O script deve ter dado o caminho onde este arquivo está localizado
- Use o seu editor de texto favorito e abra este arquivo

Encontrando nosso primeiro fhicl

- Alguns parâmetros estão definidos mas nem todos → tem um #include no topo!
  - Este fhicl herda configurações do singles.fcl
- Então vamos buscá-lo para descobrir o que mais está definido nele
  - Repita os passos anteriores com o singles.fcl

```
#include "singles.fcl"

BEGIN_PROLOG


################
###### FD ######
################

dunefd_singlep: @local::standard_singlep
dunefd_singlep.Theta0YZ:               [ 0.0 ]     # beam is along the z axis
dunefd_singlep.Theta0XZ:               [ 0.0 ]     # beam is along the z axis
dunefd_singlep.P0:                     [ 6. ]

# Start it in the first TPC, first cryostat
dunefd_singlep.X0:                     [ -1474. ]
dunefd_singlep.Y0:                     [ -351. ]
dunefd_singlep.Z0:                     [ 0. ]



################
###### 35t ######
################

dune35t_singlep: @local::standard_singlep
dune35t_singlep.Theta0YZ:                [ 0.0 ] # beam is along the z axis
dune35t_singlep.Theta0XZ:                [ 0.0 ] # beam is along the z axis
dune35t_singlep.X0:                      [ 100.0 ]     # move it into a region of x covering a drift volume
dune35t_singlep.Y0:                      [ 50.0 ]      # move it above the APA vertical gap
dune35t_singlep.Z0:                      [ 0.0 ]       # move it up a touch just in case

iceberg_singlep: @local::standard_singlep
iceberg_singlep.Theta0YZ: [ 0.0 ]  # particle gun pointed along z axis
iceberg_singlep.Theta0XZ: [ 0.0 ]
iceberg_singlep.X0:                      [ 50.0 ]     # move it into a region of x covering a drift volume
iceberg_singlep.Y0:                      [ 130.0 ]    # put it 40 cm from the top
iceberg_singlep.Z0:                      [ 0.0 ]      # going in +z so start at the side



END_PROLOG
```

Encontrando nosso primeiro fhicl

- Este é o arquivo base do single particle gun!
  - Contém todos os parâmetros configuráveis do módulo
  - Note que este é um arquivo independente de experimento → ele vive no LArSoft!
  - Tente identificar quais variáveis são modificadas e quais não são no nosso fhicl original

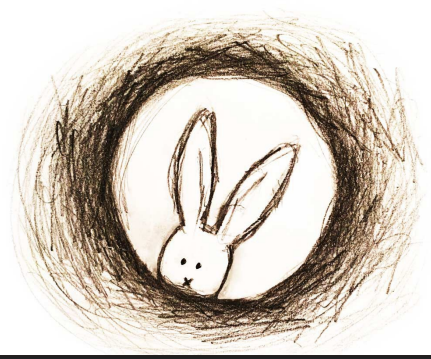
```
BEGIN_PROLOG

#no experiment specific configurations because SingleGen is detector agnostic

standard_singlep:
{
 module_type:           "SingleGen"
 ParticleSelectionMode: "all"       # 0 = use full list, 1 =  randomly select a single listed particle
 PadOutVectors:     false       # false: require all vectors to be same length
                                # true:  pad out if a vector is size one
 PDG:                   [ 13 ]      # list of pdg codes for particles to make
 P0:                    [ 6. ]      # central value of momentum for each particle
 SigmaP:                [ 0. ]      # variation about the central value
 PDist:                 "Gaussian"  # 0 - uniform, 1 - gaussian distribution
 X0:                    [ 25. ]     # in cm in world coordinates, ie x = 0 is at the wire plane
                                # and increases away from the wire plane
 Y0:                    [ 0. ]      # in cm in world coordinates, ie y = 0 is at the center of the TPC
 Z0:                    [ 20. ]     # in cm in world coordinates, ie z = 0 is at the upstream edge of
                                # the TPC and increases with the beam direction
 T0:                    [ 0. ]      # starting time
 SigmaX:                [ 0. ]      # variation in the starting x position
 SigmaY:                [ 0. ]      # variation in the starting y position
 SigmaZ:                [ 0.0 ]     # variation in the starting z position
 SigmaT:                [ 0.0 ]     # variation in the starting time
 PosDist:               "uniform"   # 0 - uniform, 1 - gaussian
 TDist:                 "uniform"   # 0 - uniform, 1 - gaussian
 Theta0XZ:              [ 0. ]      #angle in XZ plane (degrees)
 Theta0YZ:              [ -3.3 ]    #angle in YZ plane (degrees)
 SigmaThetaXZ:          [ 0. ]      #in degrees
 SigmaThetaYZ:          [ 0. ]      #in degrees
 AngleDist:             "Gaussian"  # 0 - uniform, 1 - gaussian
}

random_singlep: @local::standard_singlep
random_singlep.ParticleSelectionMode: "singleRandom"  #randomly select one particle from the list
```

Outra ajudinha: fhicl-dump



- Você pode saber quais são as variáveis de um dado fhicl de uma maneira muito simples

  fhicl-dump <nome>.fcl

- O resultado tende a ser enorme! Você pode fazer

  fhicl-dump <nome>.fcl > fcl_dump.txt

e abrir o fcl_dump.txt no editor de texto

```
# Produced from 'fhicl-dump' using:
#   Input  : prod_eminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6.fcl
#   Policy : cet::filepath_maker
#   Path   : "FHICL_FILE_PATH"

outputs: {
   out1: {
      compressionLevel: 1
      dataTier: "generated"
      fileName: "prod_eminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6_gen.root"
      module_type: "RootOutput"
   }
}
physics: {
   end_paths: [
      "stream1"
   ]
   producers: {
      generator: {
         AngleDist: 0
         P0: [
            2.55
         ]
         PDG: [
            11
         ]
         PDist: 0
         PadOutVectors: false
         ParticleSelectionMode: "all"
         PosDist: 0
         SigmaP: [
            2.45
         ]
         SigmaT: [
            500
         ]
         SigmaThetaXZ: [
            180
         ]
         SigmaThetaYZ: [
            90
         ]
         SigmaX: [
            360
```

fhicl-expand imprime TODOS os parâmetros dos fhicls incluídos

Nossa sequência de simulação

- Esta é uma sequência dos estágios que discutimos até agora

  prod_eminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6.fcl

  standard_g4_dune10kt_1x2x6.fcl

  standard_detsim_dune10kt_1x2x6.fcl

- Explore estes fhicls
- O que vamos simular?