



Escola Brasileira de Neutrinos

Simulação da interação de neutrinos em LArTPCs Parte 2*

Laura Paulucci
ITA

laura.paulucci@ita.br

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



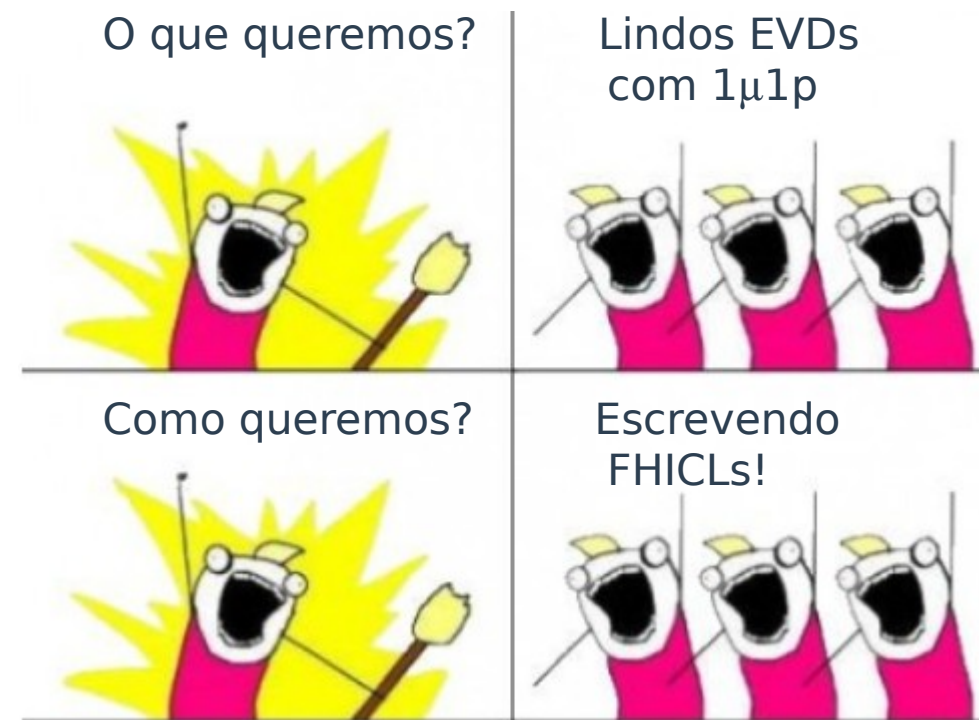
Para que serve esta parte da escola?

- Proposta: apresentar como é feita a simulação de interações em uma LArTPC
 - › 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?
-  prática
- Vamos escrever nosso primeiro arquivo FHICL → queremos entender a sua estrutura
 - Vamos gerar eventos
 - Vamos observar estes eventos no event display



Objetivo

- Gerar 10 eventos com 1 muon e 1 proton usando o gerador de partículas (Single particle gun)
 - } Vamos escrever o fhicl do zero
 - } Não vamos precisar instalar nada do código do DUNE
 - Somente para fazer uma das tarefas bônus



A estrutura do arquivo FHICL

A sintaxe do arquivo FHICL é ter pares nome-valor(es) que são usados para declarar diferentes parâmetros

Os parâmetros podem ser definidos

- um a um

```
pi: 3.14159
beam: "NuMI"
output_name: "my_beautiful_file.root"
```

- ou em sequências

```
seq1: [1, 2, 3] // A sequence of integers
seq2: [3.14, 2.17, "Gaussian"] // Sequence of floats and words
seq3: [1, 3.14, [2, "Gaussian"], 5] // Sequences are also
allowed as elements
```

- Sequências podem ser reescritas. O índice inicial de qualquer sequência é zero!

```
seq2[2]: "Uniform" // "Gaussian has been changed by "Uniform"
```



A estrutura do arquivo FHICL

A sintaxe do arquivo FHICL é ter pares nome-valor(es) que são usados para declarar diferentes parâmetros

Comentários

- Podem ser escritos com a sintaxe de C++ ou Python

```
pi: 3.14159 // This is a comment  
beam: "NuMI" # This is also a comment  
output_name: "my_beautiful_file.root"
```

A estrutura do arquivo FHICL

Tabelas

- Uma tabela pode receber a mesma configuração de outra + novos parâmetros usando @table::tab_name

```
tab3:{  
    @tab::tab1  
    time: 1.9  
}
```

O que é o mesmo que

```
tab3:{  
    pi:3.1415  
    beam: "NuMI"  
    seq1: [3.14, 2.17, "Gaussian"]  
    time: 1.9  
}
```



A estrutura do arquivo FHICL

Prólogos

- Contêm configurações que podem ser acessadas em outros arquivos
- Usar um *prolog* permite definir valores alternativos para colocar nas simulações/análises

```
BEGIN_PROLOG
    bnb: 8      // 8 GeV beam
    numi: 120   // 120 GeV beam
END_PROLOG

BeamEnergy: @local::numi
```

- Um *prolog* pode ser escrito em um arquivo, depois este pode ser incluído em outro arquivo e os valores definidos podem ser usados.

ATIVIDADE

Vamos escrever nosso primeiro arquivo FHICL!!!



Acessando as máquinas

1. Acessar o cluster do CCCT-CP

```
ssh -X -p 9060 visitante@hpc.cp.utfpr.edu.br  
senha: acasaesua
```

2. Acessar a máquina DuneXX

```
ssh -X neutrinoID@10.20.74.NN
```

3. Ative o container

```
source container.sh
```

4. Ative o setup do dune

```
source setupEBN.sh
```

5. Ative o VNC em outro terminal (senha: ebn2025)

```
Ver instruções no Slack!
```



Arquivo FHICL - passo 0: novo arquivo

- Crie uma pasta nova

```
mkdir tutorial_sim  
cd tutorial_sim
```

- Crie um arquivo novo

- `touch prod_1mu1p_dune1x2x6.fcl`

- Abra esse arquivo com o seu editor favorito (emacs -nw, vi, nano, etc)
por exemplo, `nano prod_1mu1p_dune1x2x6.fcl`



Arquivo FHICL - passo 1: esqueleto

- A estrutura básica do arquivo deve conter todas essas seções e se parecer com isso

Agora precisamos preenchê-lo!



```
#include

process_name:

services:
{
}

source:
{
}

physics:
{
}

outputs:
{
}
```

Arquivo FHiCL - passo 2: includes

- Já vimos que FHiCLs podem herdar configurações de outros FHiCLs
 - } Fazemos isso com os #include
 - } Há diferentes tipos de includes:
 - Serviços específicos dos experimentos
- #include "services_dune.fcl"
 - Arquivos de configuração que contêm prólogos
- #include "singles_dune.fcl"
- Adicione-os ao seu arquivo

```
# Experiment specific includes
#include "services_dune.fcl"

# Configuration files containing prologs
#include "singles_dune.fcl"
```

```
process_name:
```

```
services:
```

```
{
```

```
}
```

```
source:
```

```
{
```

```
}
```

```
physics:
```

```
{
```

```
}
```

```
outputs:
```

```
{
```

```
}
```



Arquivo FHICL - passo 3: process_name

- Agora precisamos dar um nome ao processo que vamos rodar
- process_name define um nome para a coleção de módulos que rodarão no seu FHICL
- Este nome precisa ser único, ou seja, não é possível usar o mesmo nome mais de uma vez sobre um dado arquivo art-root
 - › Mensagem de erro se você tentar
 - › É possível rodar os mesmos módulos sob outro nome
- Dê um nome que faça sentido. Ex: SinglesGen

```
# Experiment specific includes
#include "services_dune.fcl"

# Configuration files containing prologs
#include "singles_dune.fcl"
```

```
process_name: SinglesGen
```

```
services:
```

```
{
```

```
}
```

```
source:
```

```
{
```

```
}
```

```
physics:
```

```
{
```

```
}
```

```
outputs:
```

```
{
```

```
}
```



Arquivo FHICL - passo 4: serviços

- Começado a incluir a configuração!
- Na parte dos serviços serão incluídos serviços específicos para a simulação como
 - } Geometria do detector
 - } Propriedades físicas
 - } Gerenciamento de arquivos

Tabela dos serviços de simulação do DUNE, definida em services_dune.fcl

```
services:
{
  @table::dunefd_1x2x6_simulation_services
  TFileService: { fileName: "prod_1mu1p_workshop_hist.root" }
  RandomNumberGenerator: {}
}
```



Arquivo FHICL - passo 5: source

- Aqui precisamos definir as configurações sobre o arquivo de entrada para o que queremos rodar

```
source:
{
  module_type: EmptyEvent
  timestampPlugin:
  {
    plugin_type: "GeneratedEventTimestamp"
  }
  maxEvents: 10
  firstRun: 1
  firstEvent: 1
}
```

O módulo deve começar com um empty event. Como esse é o primeiro passo da nossa cadeia de simulação, o módulo não deve receber nenhum arquivo de input.

*Caso o módulo deva receber informações de um arquivo, deve ser usado module_type: ROOTInput



Arquivo FHICL - passo 5: source

- Aqui precisamos definir as configurações sobre o arquivo de entrada para o que queremos rodar

```
source:
{
  module_type: EmptyEvent
  timestampPlugin:
  {
    plugin_type: "GeneratedEventTimestamp"
  }
  maxEvents: 10
  firstRun: 1
  firstEvent: 1
}
```

Gera uma timestamp

Número padrão*
de eventos para
simular

Valores iniciais
de run e evento

*Se -1, processa todos os eventos
presentes no arquivo de entrada

Arquivo FHICL - passo 6: física

- Aqui configuramos o(s) módulo(s) que vão de fato rodar no evento

Filters:
removem coisas
que não nos
interessam nos
eventos

Obs: Somente
producers e filters
modificam
eventos, não
analyzers

```
physics:
{
  producers:
  {
    rns:
    {
      module_type: "RandomNumberSaver"
    }
    generator: @local::dunefd_singlep
  }
  analyzers: {}
  filters: {}

  simulate: [rns, generator]
  stream1 : [out1]

  trigger_paths: [simulate]
  end_paths      : [stream1]
}
```

Producers:
Módulos que
adicionam
informação ao
arquivo artROOT

Analyzers:
módulos de
análise que
operam sobre os
eventos
colocados no
arquivo de saída



Arquivo FHICL - passo 7: output

- Aqui precisamos definir as configurações sobre o arquivo de entrada para o que queremos rodar

```
outputs:
{
  out1:
  {
    module_type: RootOutput
    fileName: "prod_1mulp_workshop_%p-%tc.root"
  }
}
```

Arquivo de saída tem extensão .root

Nome do arquivo de saída



Arquivo FHICL - completo!

- Seu lindo primeiro FHICL deve se parecer com este:

```
# Experiment specific includes
#include "services_dune.fcl"

# Configuration files containing prologs
#include "singles_dune.fcl"

process_name: SinglesGen

services:
{
  #@table::services
  @table::dunefd_1x2x6_simulation_services
  TFileService: { fileName: "prod_1mu1p_workshop_hist.root" }
  RandomNumberGenerator: {}
}

source:
{
  module_type: EmptyEvent
  timestampPlugin:
  {
    plugin_type: "GeneratedEventTimestamp"
  }
  maxEvents: 10
  firstRun: 1
  firstEvent: 1
}

physics:
```

```
physics:
{
  producers:
  {
    rns:
    {
      module_type: "RandomNumberSaver"
    }
    generator: @local::dunefd_singlep
  }
  analyzers: {}
  filters: {}

  simulate: [rns, generator]
  stream1 : [out1]

  trigger_paths: [simulate]
  end_paths      : [stream1]
}

outputs:
{
  out1:
  {
    module_type: RootOutput
    fileName: "prod_1mu1p_workshop_%p-%tc.root"
  }
}
```



Arquivo FHICL - passo 7: como rodá-lo

- No LArSoft, arquivos FHICL são lidos e executados com o comando **lar**
- Há vários argumentos que podem ser passados com o comando **lar**. Os principais são
 - } -c, --config → FHICL para rodar
 - } -s, --source → o arquivo de entrada (artROOT criado em estágio anterior)
 - } -n, --evts → número de eventos para rodar
 - } -o, --output → substitui o nome do arquivo de saída padrão
 - } -k, --nskip → número de eventos a serem pulados
 - } -T, --TFileName → substitui o nome do arquivo de saída padrão do *analyzer*



Arquivo FHICL - passo 7: como rodá-lo

- Vamos usar o comando `lar` para rodar o FHICL que criamos

```
lar -c prod_1mu1p_dune1x2x6.fcl
```

- Funcionou? Na tela, deve ter aparecido algo parecido com isso:

```
TrigReport ----- Event summary -----
TrigReport Events total = 10 passed = 10 failed = 0

TrigReport ----- Modules in End-path -----
TrigReport      Run      Success      Error Name
TrigReport      10        10          0 out1

TimeReport ----- Time summary [sec] -----
TimeReport CPU = 1.086477 Real = 1.671442

MemReport ----- Memory summary [base-10 MB] -----
MemReport VmPeak = 1312.13 VmHWM = 575.91

Art has completed and will exit with status 0.
```

status 0:  programa executado sem erros



Arquivo FHICL - passo 8: configurações

- Dê um ls na sua pasta. Você deve agora ter um arquivo com um nome como

```
prod_1mu1p_workshop_SinglesGen-20250721T193608.root
```

process_name

Timestamp

- O que este arquivo contém?
 - } Você não especificou no seu FHICL nada sobre o tipo de evento que você queria gerar!
 - } Seu FHICL usou os parâmetros padrão incluídos em singles_dune.fcl



Arquivo FHICL - passo 8: configurações

- Nós vimos este arquivo mais cedo! Ele herda de outro FHICL e faz algumas modificações

O momento inicial, ângulos e posição inicial são alterados em relação ao definido em singles.fcl

```
#include "singles.fcl"

BEGIN_PROLOG

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

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

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



Arquivo FHICL - passo 8: configurações

- Os demais parâmetros são iguais aos daqui
- Ou seja, você gerou 10 muons com posição inicial (-1474, -351, 0), energia de 6 GeV, tempo inicial 0 e direção inicial paralela ao solo
- Não é isso que queríamos! 

Obs: você pode descobrir tudo isso com o fhicl-dump

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
  ThetaXZ:               [ 0. ]    #angle in XZ plane (degrees)
  ThetaYZ:               [ -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

argoneut_singlep: @local::standard_singlep

microboone_singlep: @local::standard_singlep

microboone_singlep.Theta0YZ: [0.0] # beam is along the z axis.

microboone_singlep.X0: [125] # in cm in world coordinates, ie x = 0 is at the wire plane

microboone_singlep.Z0: [50] # in cm in world coordinates

END_PROLOG



Arquivo FHICL - passo 8: configurações

- Os demais parâmetros são iguais aos daqui
- Ou seja, você gerou 10 muons com posição inicial (-1474, -351, 0), energia de 6 GeV, tempo inicial 0 e direção inicial paralela ao solo
- Não é isso que queríamos! 

Obs: você pode descobrir tudo isso com o fhicl-dump

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:
  SigmaX:
  SigmaY:
  SigmaZ:
  SigmaT:
  PosDist:
  TDist:
  Theta0X:
  Theta0Y:
  SigmaThetaXZ:         [ 0. ]    #in degrees
  SigmaThetaYZ:         [ 0. ]    #in degrees
  AngleDist:            "Gaussian" # 0 - uniform, 1 - gaussian
}
```

Importante

Para gerar diversas partículas ao mesmo tempo, o parâmetro 'PadOutVectors' deve ser modificado para 'true'.

random_singlep: @local::standard_singlep

random_singlep.ParticleSelectionMode: "singleRandom" #randomly select one particle from the list

argoneut_singlep: @local::standard_singlep

microboone_singlep: @local::standard_singlep

microboone_singlep.Theta0YZ: [0.0] # beam is along the z axis.

microboone_singlep.X0: [125] # in cm in world coordinates, ie x = 0 is at the wire plane

microboone_singlep.Z0: [50] # in cm in world coordinates

END_PROLOG



Arquivo FHICL - passo 8: configurações

- Nós queremos 10 eventos com $1\mu 1p$!
- Precisamos modificar o nosso FHICL!
 - Dica 1: posição, momento, etc... são definidos como sequências. No momento, apenas para 1 partícula mas nada te impede de adicionar mais
 - Dica 2: Você pode consultar os valores de PDG [aqui](#)
- Tentem modificar o seu FHICL reescrevendo parâmetros para gerar 10 eventos com as seguintes características:

	Start position	Momentum	Start time	θ_{xz}	θ_{yz}
Muon	(-100, 300, 100)	700 MeV	500	-10°	35°
Proton	(-100, 300, 100)	800 MeV	500	0°	30°



Arquivo FHICL - passo 8: configurações

Tempo de trabalho



Arquivo FHICL - passo 8: configurações

- Você só precisa adicionar as seguintes linhas ao final do seu FHICL

```
physics.producers.generator.PadOutVectors: true
physics.producers.generator.PDG: [13, 2212]
physics.producers.generator.PDist: "Gaussian"
physics.producers.generator.X0: [ -100]
physics.producers.generator.Y0: [ 300.]
physics.producers.generator.Z0: [ 100]
physics.producers.generator.T0: [ 500.0]
physics.producers.generator.PosDist: "uniform"
physics.producers.generator.SigmaX: [ 0 ]
physics.producers.generator.SigmaY: [ 0 ]
physics.producers.generator.SigmaZ: [ 0 ]
physics.producers.generator.SigmaT: [ 500.0 ]
physics.producers.generator.P0: [ 0.7, 0.8 ]
physics.producers.generator.SigmaP: [0.]
physics.producers.generator.AngleDist: "Uniform"
physics.producers.generator.Theta0XZ: [ -10., 35. ]
physics.producers.generator.Theta0YZ: [ 0., 30. ]
physics.producers.generator.SigmaThetaXZ: [ 0 ]
physics.producers.generator.SigmaThetaYZ: [ 0. ]
```



Arquivo FHICL - passo 9: data products

- Agora que você rodou a simulação com os parâmetros corretos, vamos checar o que contém o arquivo de saída

- Rode

```
lar -c eventdump.fcl -s your_output_file.root -n 1
```

- Só vamos olhar 1 evento
- Deve aparecer na tela algo assim (+ algumas coisas)



```
21-Jul-2025 14:41:14 CDT opened input file prod_indep SinglesGen 20250721195000.root
Begin processing the 1st record. run: 1 subRun: 0 event: 1 at 21-Jul-2025 14:41:14 CDT
PRINCIPAL TYPE: Event
PROCESS NAME | MODULE LABEL.. | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE..... | SIZE
SinglesGen.. | generator..... | ..... | std::vector<simb::MCTruth>... | ...1
SinglesGen.. | rns..... | ..... | std::vector<art::RNGsnapshot> | ...1
SinglesGen.. | TriggerResults | ..... | art::TriggerResults..... | ...1
```



Todos os produtos gerados
pelo estágio SinglesGen

Data product
MCTruth!



Propagando as partículas

- Agora que conseguimos sucesso em gerar as partículas de interesse, vamos ao próximo estágio: g4

 } É hora de algo acontecer com nossas partículas

- Usaremos o FHICL padrão para a geometria que usamos

`lar -c standard_g4_dune10kt_1x2x6.fcl -s seu_arquivo_gen.root -n 1`

- Se você não especificar um nome para o arquivo de saída (-o), ele será chamado `your_generated_file_g4.root`
- Seu processo deve rodar sem problemas (code 0)



Propagando as partículas

- Vamos olhar novamente o arquivo de saída com o eventdump

Produtos
do
estágio
anterior

Produtos
deste
estágio

```
Begin processing the 1st record. Part 1 Subant 0 event: 1 at 03 Feb 2023 19:52:11 CEST
```

PROCESS NAME	MODULE LABEL..	PRODUCT INSTANCE NAME.....	DATA PRODUCT TYPE.....	SIZE
SinglesGen..	generator.....		std::vector<simb::MCTruth>.....	...1
SinglesGen..	rns.....		std::vector<art::RNGsnapshot>.....	...1
SinglesGen..	TriggerResults		art::TriggerResults.....	...1
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneVInner0	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneZInner0	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCOuter.....	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCInner1.....	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCInner0.....	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneZOuter.	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneZInner1	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	PDFastSim.....		std::vector<sim::OpDetBacktrackerRecord>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneVInner1	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneVOuter.	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	rns.....		std::vector<art::RNGsnapshot>.....	...4
G4.....	largeant.....	LArG4DetectorServicevolTPCActiveInner0	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	PDFastSim.....	Reflected.....	std::vector<sim::SimPhotonsLite>.....	.480
G4.....	PDFastSim.....		std::vector<sim::SimPhotonsLite>.....	.480
G4.....	IonAndScint...		std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCActiveInner1	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	TriggerResults		art::TriggerResults.....	...1
G4.....	largeant.....		std::vector<simb::MCParticle>.....	..17
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneUInner0	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneUInner1	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	PDFastSim.....	Reflected.....	std::vector<sim::OpDetBacktrackerRecord>.....	...0
G4.....	largeant.....		sim::ParticleAncestryMap.....	...-
G4.....	largeant.....		art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo>	..17
G4.....	largeant.....	LArG4DetectorServicevolTPCPlaneUOuter.	std::vector<sim::SimEnergyDeposit>.....	...0
G4.....	largeant.....	LArG4DetectorServicevolTPCActiveOuter.	std::vector<sim::SimEnergyDeposit>.....	...0



Finalmente, os dados simulados

- O último estágio da nossa simulação é o detsim
 - } É hora de gerar a resposta da TPC e do PDS
- Usaremos o FHICL padrão para a geometria que usamos
 - `lar -c standard_detsim_dune10kt_1x2x6.fcl -s seu_arquivo_g4.root`
- Vai levar um tempinho para rodar por conta do wirecell
- Se você não especificar um nome para o arquivo de saída (-o), ele será chamado `your_input_file_detsim.root`
- Seu processo deve rodar sem problemas (code 0)
- Vamos rodar o eventdump uma última vez
 - } Você identifica todos os data products, novos e antigos?



O event display

- Agora nós temos dados simulados que podemos reconstruir e analisar (amanhã)
- Mas não sabemos que cara eles têm :-(
- Mas o LArSoft tem um display de eventos que você pode usar para visualizar os dados gerados e ter certeza de que tudo funcionou como esperado
- Para rodar o event display

```
lar -c evd_dune10kt_1x2x6.fcl -s your_detsim_file.root
```

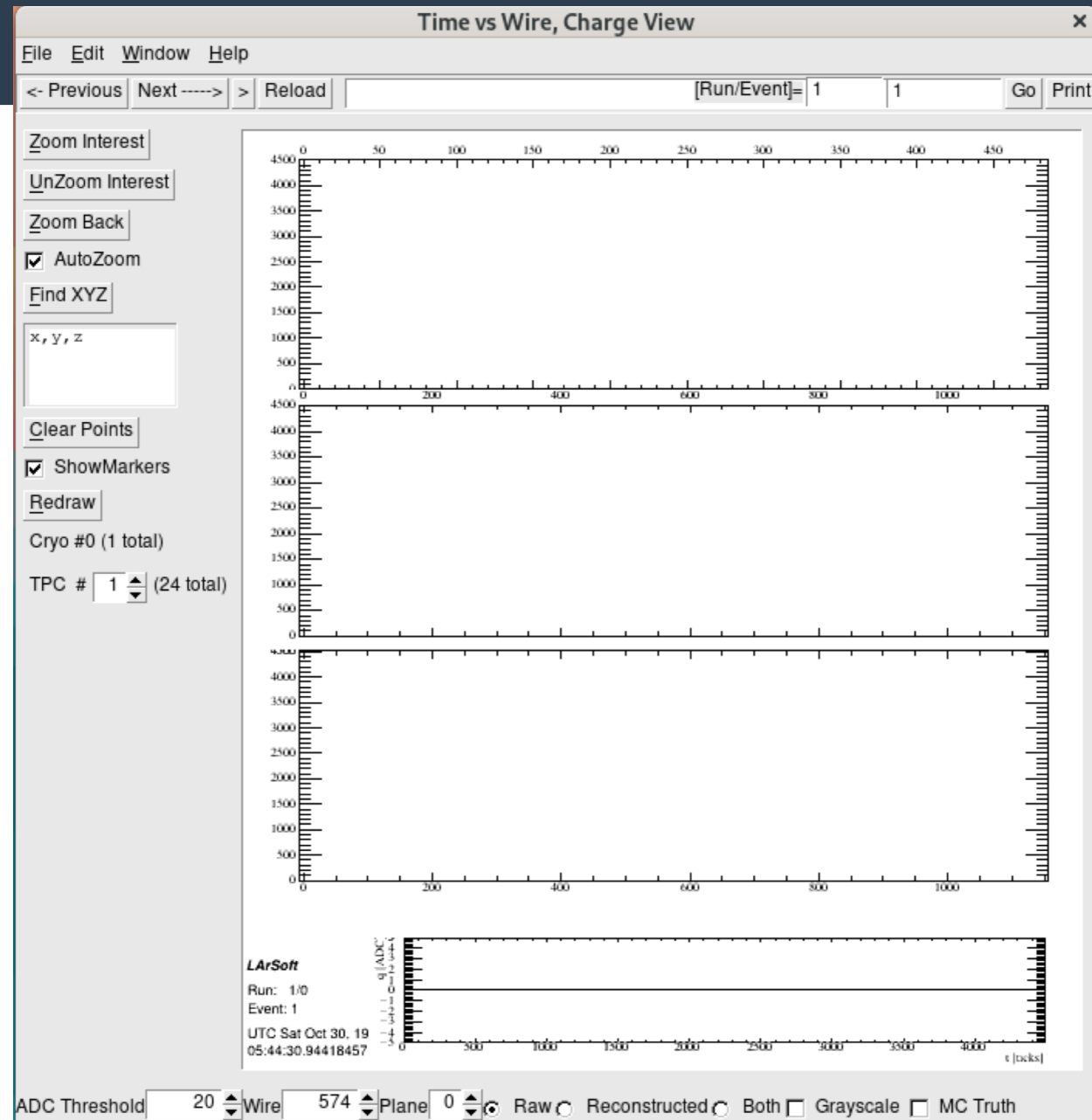


O event display

- Você deve ver isso no seu VNC
- Mas não aparece nada!

Sem pânico, isso é normal!

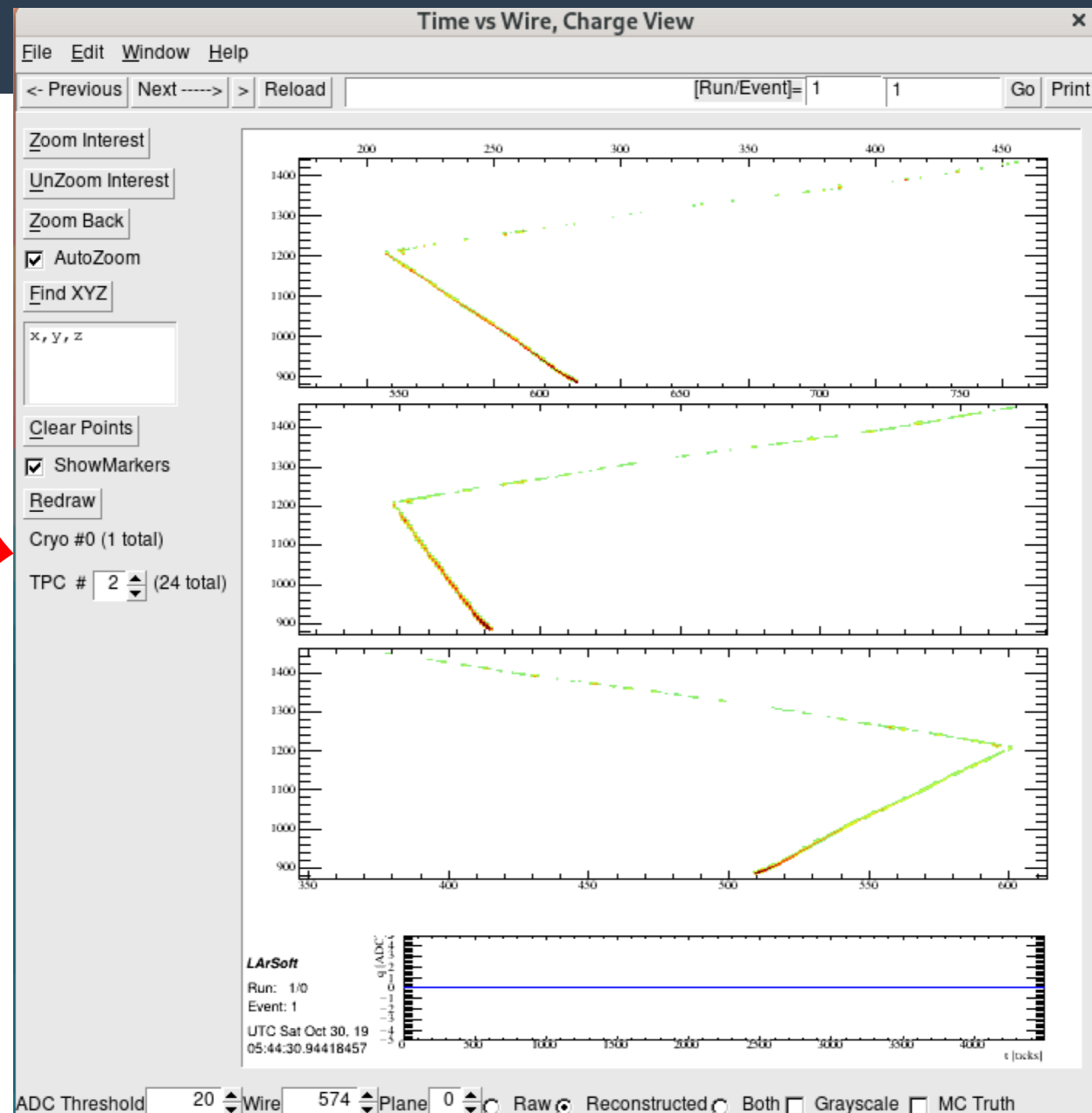
O volume do detector é dividido em várias diferentes regiões e os sinais tipicamente ficam restritos a somente algumas



O event display

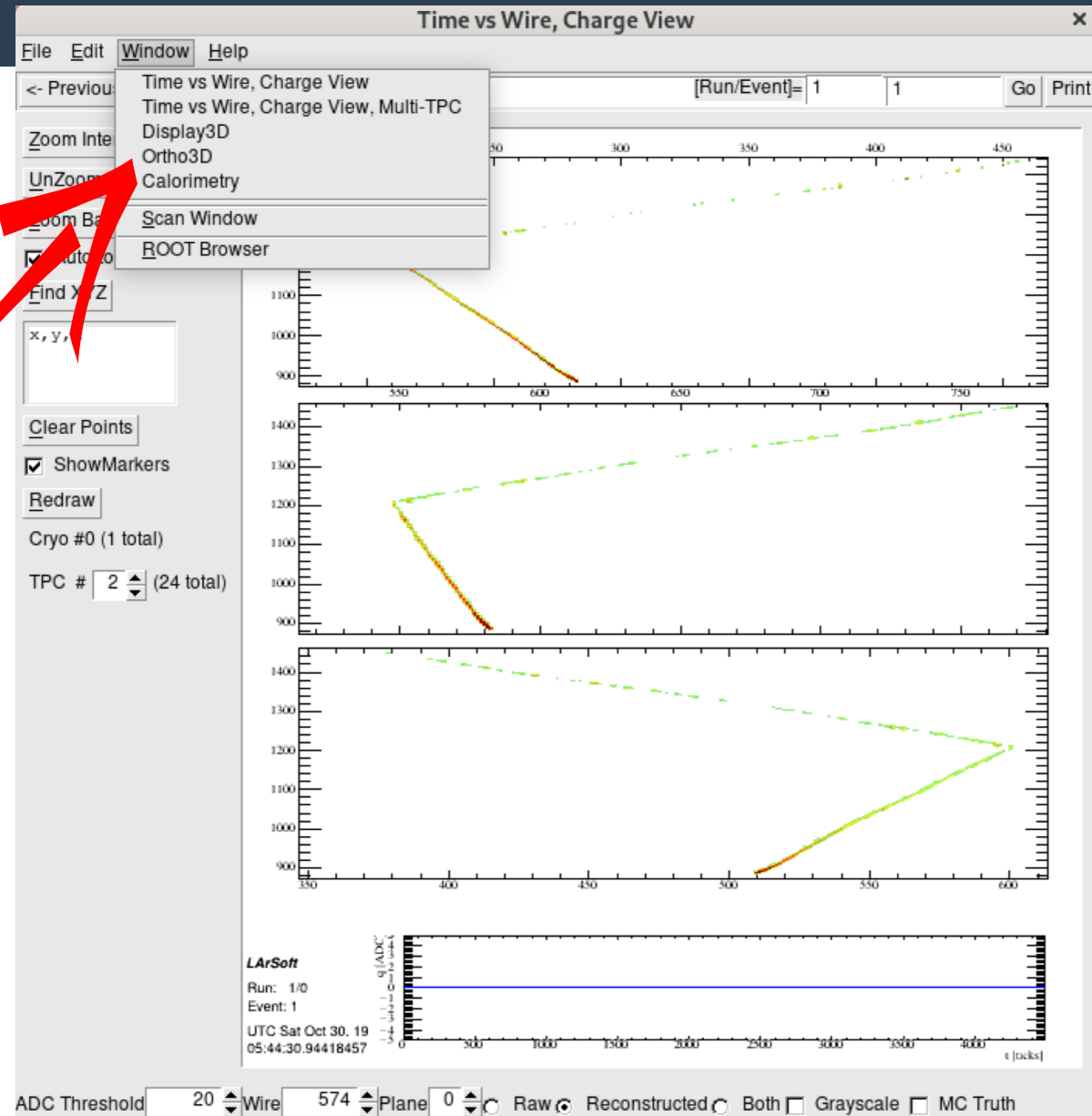
- Solução 1:
Mude o número da TPC
até encontrar o sinal

Qual é o muon e
qual é o próton?



O event display

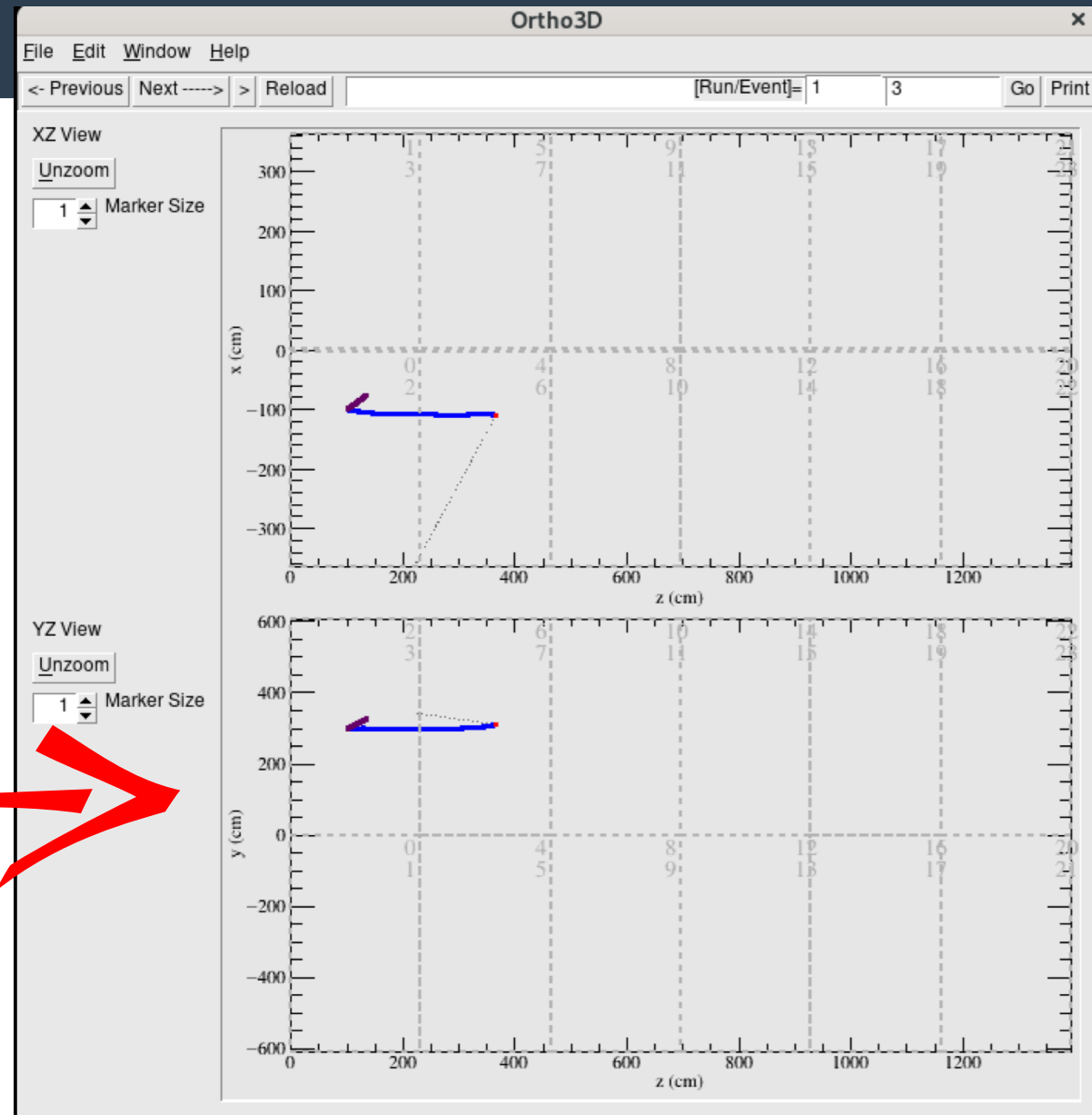
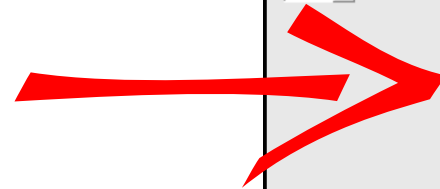
- Solução 1:
Mude o número da TPC até encontrar o sinal
- Solução 2:
- Vá ao Ortho3D view para encontrar a TPC onde está o sinal



O event display

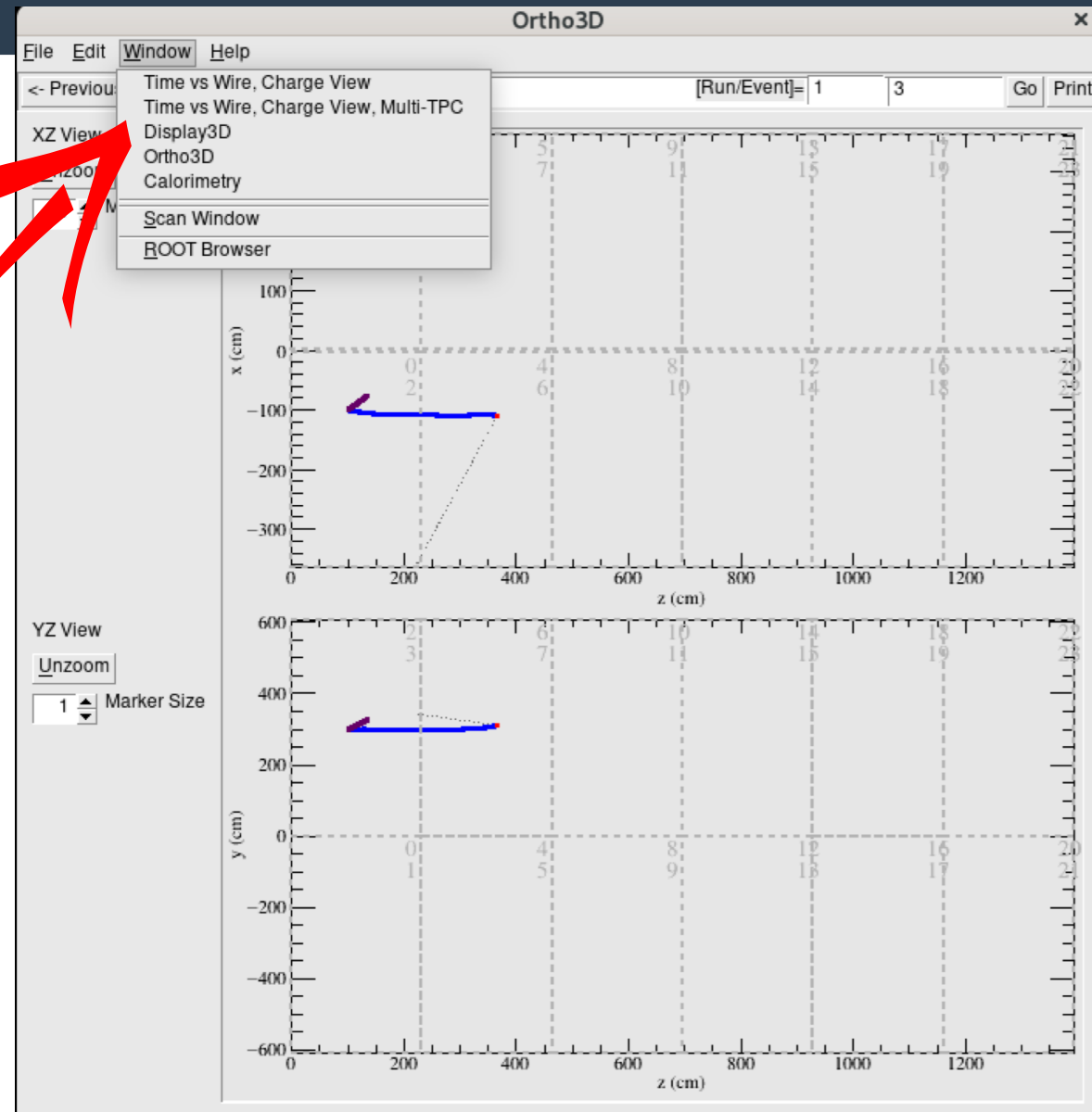
- Solução 1:
Mude o número da TPC até encontrar o sinal
- Solução 2:
- Vá ao Ortho3D view para encontrar a TPC onde está o sinal

E obtenha isso



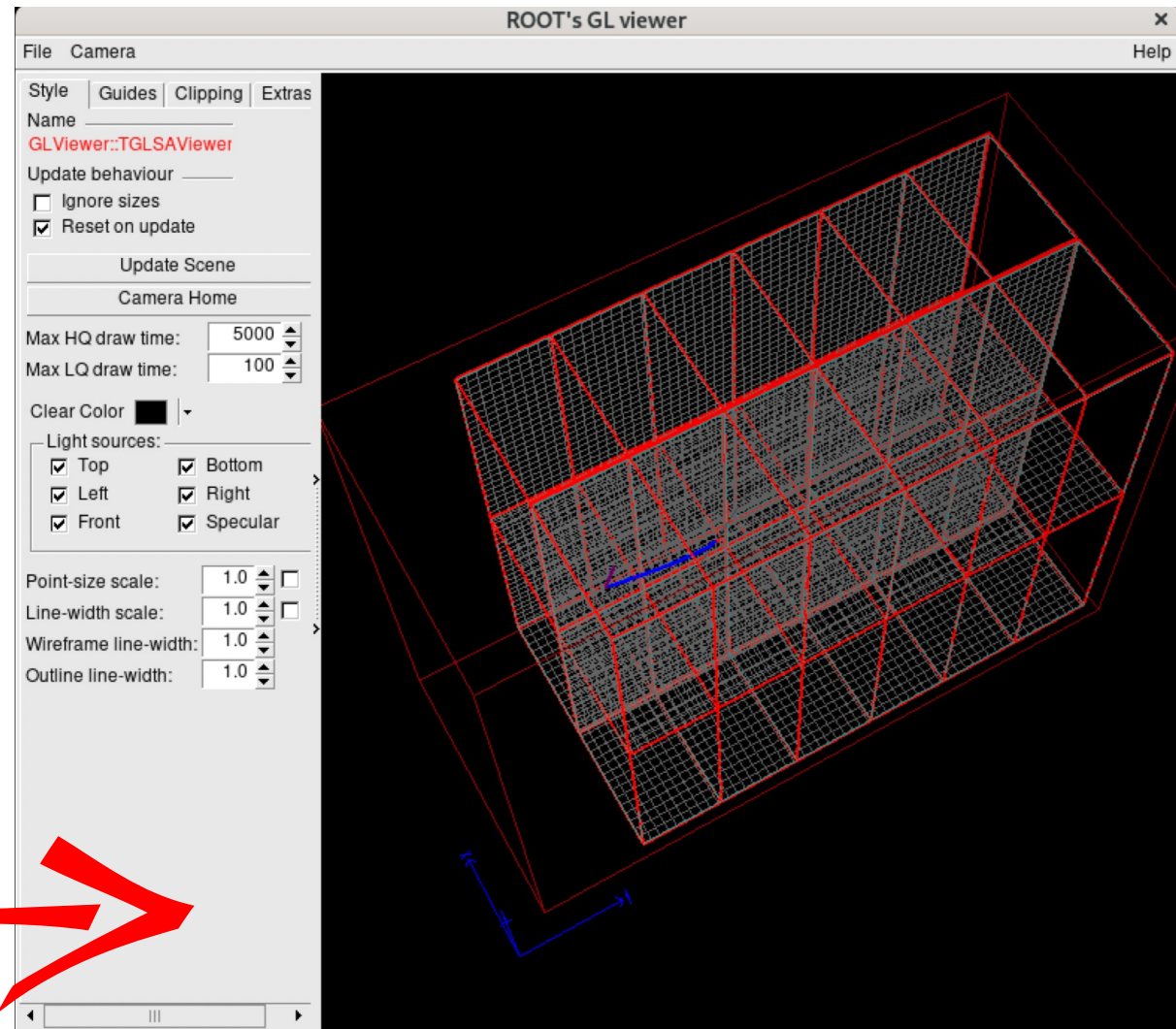
O event display

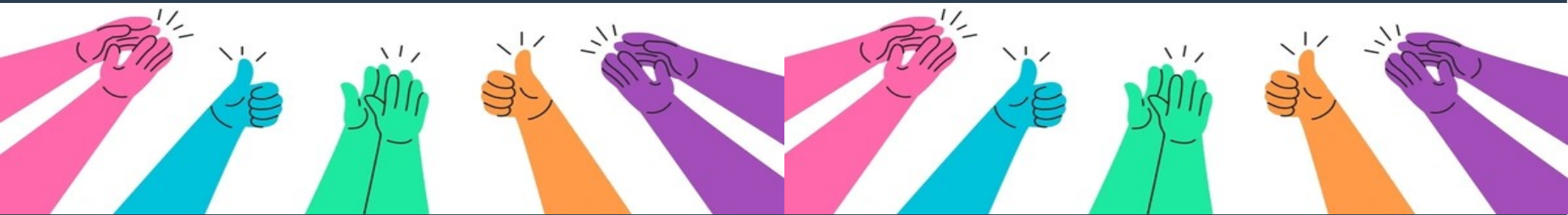
- Solução 1:
Mude o número da TPC até encontrar o sinal
- Solução 2:
- Vá ao Ortho3D view para encontrar a TPC onde está o sinal
- Solução 3:
- Vá para Display 3D view



O event display

- Solução 1:
Mude o número da TPC até encontrar o sinal
- Solução 2:
- Vá ao Ortho3D view para encontrar a TPC onde está o sinal
- Solução 3:
- Vá para Display 3D view para ver isso





Parabéns! Você gerou dados simulados com sucesso!!!



Bônus (?!)

- Modifique o seu FHICL para incluir alguma variação na distribuição angular das partículas geradas
- Repita os passos (gen, g4, detsim) e abra o event display
 - } Você consegue ver o efeito da variação angular?
- Agora faça o mesmo adicionando variação na posição inicial das partículas
- Finalmente, mude o seu FHICL (ou crie um novo, lendo o primeiro! Pouquíssimas linhas!!!) e gere 10 eventos com um elétron e um próton
 - } Você consegue identificar o chuveiro do elétron no EVD?



Mais Bônus (!!)

- E se você estiver em um experimento na superfície?
- Gere 10 eventos com um muon e um próton, como inicialmente, mas também adicione mais 5 muons (cósmicos)
 - } Os muons adicionais devem ser distribuídos uniformemente no volume do detector
- Rode os estágios gen, g4 e detsim
- Abra o event display e identifique o que está acontecendo



Bônus bônus!!!

- E se simularmos interações de neutrinos?
- Use um FHICL padrão do GENIE no estágio gen:
`prodgenie_nu_dune10kt_1x2x6.fcl`
- Rode até o estágio detsim e abra o event displays
- Fique à vontade para olhar (e testar!) outros FHICLs para a mesma geometria. Eles estão disponíveis em `dunesw/fcl/gen/genie`
 - } O que eles têm de diferente um do outro?



Resumo

- Este tutorial encerra aqui
 - } Espero que tenham gostado e que não estejam muito traumatizados!
- Agora vocês sabem como escrever e rodar um arquivo FHICL para gerar eventos em um detector e gerar dados simulados
 - } Não apague os seus arquivos detsim para 1μ1p, você voltará a usá-los amanhã
- Se você não conseguiu terminar as tarefas e/ou os bônus, sem problemas, continue no seu ritmo
 - } Você também pode explorar os FHICLs utilizados e criar novas simulações. Fique livre para criar!

