Light Simulation and Reconstruction Tutorial

Vincent Basque*
UK LArSoft Workshop 2026



THE UNIVERSITY of EDINBURGH

*slides are **heavily** based on previous versions from Patrick Green & Andrzej Szelc

Introduction

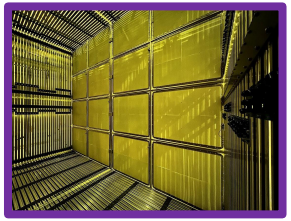
- This tutorial will go over how to simulate the light and explore different scenarios.
  - We will be using the SBND geometry today but any geometry works the same (remember that the geometry/GDML is detector dependent!).
  - While LArG4 generates both charge and light, we will only look at light here!
- Tutorial is divided into two parts:
  - Running the light simulation and looking at “truth-level” quantities.
  - Running the detector response simulation (“`detsim`”) and reconstruction (“`reco`”) to look at the reconstructed quantities.
- Each parts will be introduced separately with specific tasks that you can work through at your own pace.
  - Talk and share with your neighbour! Did you get similar results? Why or why not?
- **Let’s make some light!**



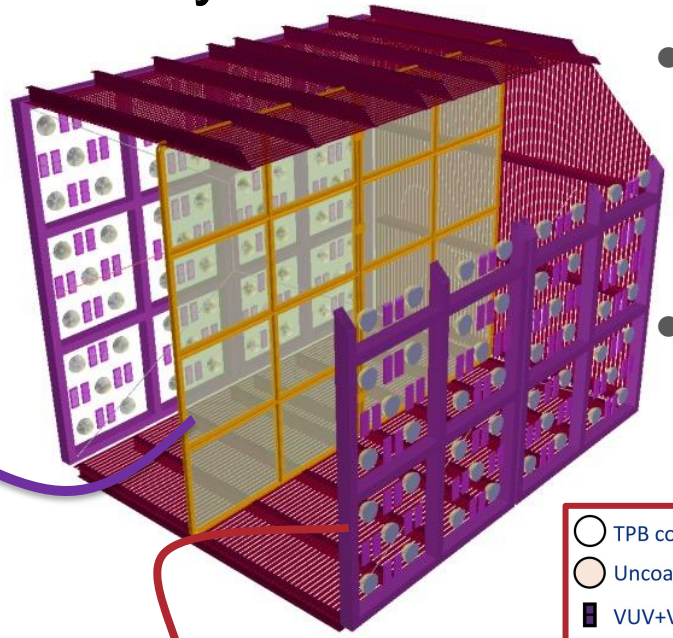
Part 1: Running the light simulation

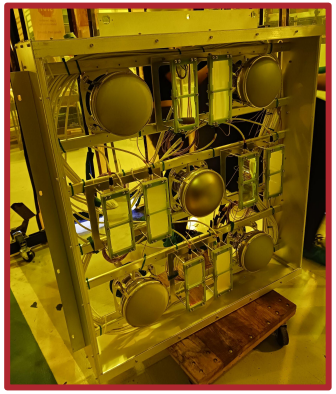


SBND geometry reminder - 2 TPCs & PDS



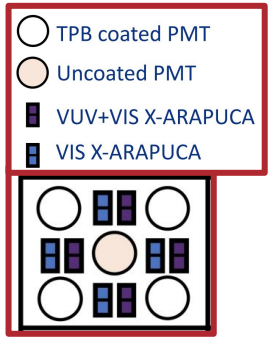
TPB coated reflector HV cathode

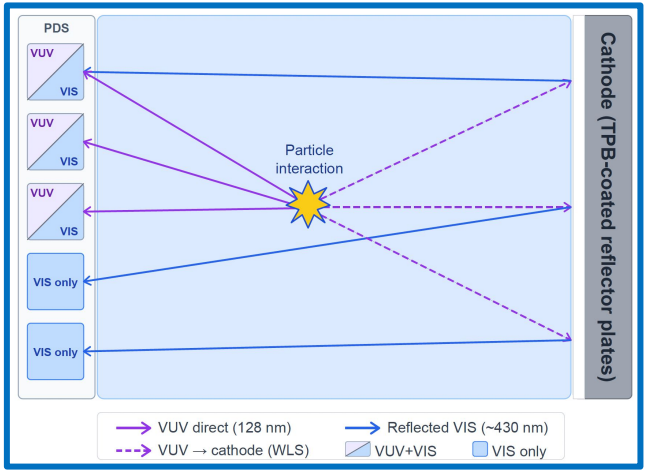




Dual photon detection technologies

- SBND photon detection system (PDS):
  - active elements (anode)
    - *PMTs and X-ARAPUCAs*
  - Passive element (cathode)
    - *Wavelength-shifting reflectors*
- Two light component "direct"/vuv & "reflected"/visible (TPC independent!)





How do we run light only simulation?

- We need events!
- **Reminder:** many ways to generate events (event generator, TextFileGen, SingleParticle…). We will use ***SingleParticle.***
  - This is flexible and *easy* to use. You decide on the particles and their parameters you want to generate.
- First we will generate a 2 GeV muon (PDG: 13) in one of the SBND TPCs using: optical_tutorial_sim_muons.fcl
  - It is located in Workshop/Photon/fcl in your local sbndcode src directory.
- Simulation process chain:
  - Generator -> LArG4 -> light_sim -> analyzer
  - The analyzer module will provide root TTrees with truth-level information based on the light we simulated.

```
# generator parameters
physics.producers.generator.PadOutVectors: true
physics.producers.generator.PDG: [13]
physics.producers.generator.P0: [2.0] # GeV
physics.producers.generator.SigmaP: [0]
physics.producers.generator.PDist: 0
physics.producers.generator.X0: [100.]
physics.producers.generator.Y0: [0]
physics.producers.generator.Z0: [50]
physics.producers.generator.T0: [0]
physics.producers.generator.Theta0XZ: [0]
physics.producers.generator.Theta0YZ: [0]
physics.producers.generator.SigmaThetaXZ: [0]
physics.producers.generator.SigmaThetaYZ: [0]
```



But what is in optical_tutorial_sim_muons.fcl ?



```
producers:
{
  rns: { module_type: "RandomNumberSaver" }

  # Generation
  generator: @local::sbnd_singlep

  # A dummy module that forces the G4 physics list to be loaded
  loader: { module_type: "PhysListLoader" }

  # The geant4 step
  largeant: @local::sbnd_larg4

  # Creation of ionization electrons and scintillation photons, inside the active volume
  ionandscint: @local::sbnd_ionandscint

  # Creation of ionization electrons and scintillation photons, outside the active volume
  ionandscintout: @local::sbnd_ionandscint_out

  # Light propagation inside the active volume, semi-analytic model
  pdfastsim: @local::sbnd_pdfastsim_par

  # Light propagation outside the active volume, photon library
  pdfastsimout: @local::sbnd_pdfastsim_pvs

  # Electron propogation
  simdrift: @local::sbnd_simdrift

  # Truth-level reconstruction
  mcreco: @local::sbnd_mcreco
}
```

single particle generator

LArG4

Hybrid model light simulation

Configuration of semi-analytic and/or library models (detector-specific)

```
sbnd_pdfastsim_par {

    # Direct (VUV)
    VUVTiming:    @local::sbnd_vuv_timing_parameterization
    VUVHits:      @local::sbnd_vuv_RS100cm_hits_parameterization

    # Reflected (Visible)
    VISTiming:    @local::sbnd_vis_timing_parameterization
    VISHits:      @local::sbnd_vis_RS100cm_hits_parameterization

    DoReflectedLight: true
    IncludePropTime: true
    ...
}
```

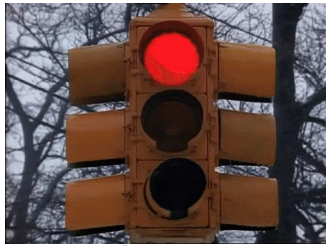
reflected light

propagation time

Also include an analyzer that will allow us to access the truth level information: (SBND-specific but similar tools available in other experiments)

```
analyzers:
{
  # Analyzer to count number of photons arriving on photo-detectors
  opanalyzer: @local::OpDetAnalyzer
}
```

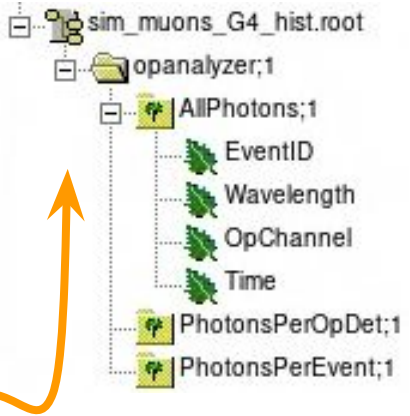
Task 1.0: Getting everything ready



- We will use the same sbndcode installation from the Simulation Tutorial
  - No need to build it again!
- If your container is still active (or if you have your own scripts) you can skip the setting up steps below:
  - /cvmfs/oasis.opensciencegrid.org/mis/apptainer/current/bin/apptainer shell --shell=/bin/bash --env PS1="(CONTAINER) [\H@\w]$ " -B /cvmfs,/opt,/run/user,/etc/hostname,/etc/hosts,/etc/krb5.conf,/tmp,/var/run/user --ipc --pid /cvmfs/singularity.opensciencegrid.org/fermilab/fnal-dev-sl7:latest
  - cd <your-working-directory>
  - source /cvmfs/sbnd.opensciencegrid.org/products/sbnd/setup_sbnd.sh
  - source localProducts_larsoft_v10_14_02_04_e26_prof/setup
  - mrbslp
- Make a new directory, give a name (photon_tutorial) in your home directory and copy optical_tutorial_sim_muons.fcl
  - This .fcl is found in <your-working-directory>/src/sbndcode/sbndcode/Workshop/Photon/fcl/

Task 1.1: Running the light simulation





- **Time to run your first light simulation for 1 event!**
  You are generating many thousands of photons! It takes a while to do! Look at the printouts as they appear.

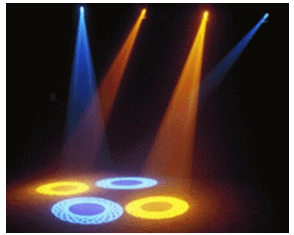
  ✓ `lar -c optical_tutorial_sim_muons.fcl -n 1`
- The OpDetAnalyzer module will make an output root file with the truth-level information & and artroot file to use later:
  - **AllPhotons** - contains information about each and all created photons
  - **PhotonsPerOpDet** - contains number of photons arriving at light detector
  - **PhotonsPerEvent** - contains total number of photons detected per event
- Use TBrowser to look at the **AllPhotons** tree:
  - ? *Look at **Wavelength** and **Time** quantities, are they what you expect? What about **OpChannel**? Can you see something there?*
  - ? *Can you extract the simulated scintillation light slow time constant of argon?* (hint: use Fit panel from TBrowser and try to fit an exponential and look at the slope).

Root hint:
Root -l sim_muons_G4_hist.root
New TBrowser
And click on the file from the list
(There are others ways with the command lines if you know them)

Task 1.2: How about placing the muon *somewhere else?*

- Your first muon was in the middle of one of the TPCs… Scroll all the way to the end of optical_tutorial_sim_muons.fcl to see parameters
- X0, Y0, Z0 are the start coordinate of the particles
- Move it closer to the CPA (X0=25cm) or APA (X0=175cm)
  - And rerun lar -c optical_tutorial_sim_muons.fcl -n 1 with your modified .fcl but use “-o” and “-T”
  - lar -c optical_tutorial_sim_muons.fcl -n 1 -o sim_muons_G4_25cm.root -T sim_muons_G4_25cm_hist.root
- *What will that to do the total amount of light? Will it change?*
  - Have a look at the PerEvent tree to see how it changes
- *Do you see a difference between the VUV and visible light at these positions? Why?* (hint: look at earlier slides or Patrick’s)

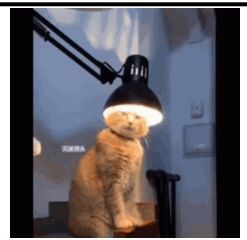
```
# generator parameters
physics.producers.generator.PadOutVectors: true
physics.producers.generator.PDG: [13]
physics.producers.generator.P0: [2.0] # GeV
physics.producers.generator.SigmaP: [0]
physics.producers.generator.PDist: 0
physics.producers.generator.X0: [100.]
physics.producers.generator.Y0: [0]
physics.producers.generator.Z0: [50]
physics.producers.generator.T0: [0]
physics.producers.generator.Theta0XZ: [0]
physics.producers.generator.Theta0YZ: [0]
physics.producers.generator.SigmaThetaXZ: [0]
physics.producers.generator.SigmaThetaYZ: [0]
```

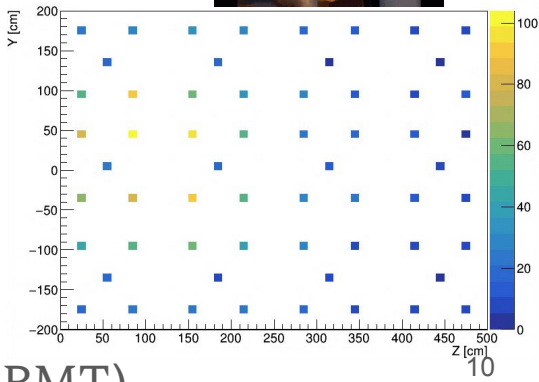


Task 1.3: what is the simulated distribution of the light?

- Remember OpChannel? It was hard to read… Let's change that! We made a simple ROOT macro that can do that for you!
  - Workshop/Photon/macro/PlotPhotonsYZ.cc -> looks at the front face of the PDS wall dead-on (slide 4 earlier)
- Try it on the muon you generated previously! *How does it look?*
- *How about changing your muons to electrons?* Copy it to your directory and run it for 5 events to see differences!
  - lar -c optical_tutorial_sim_electrons.fcl -n 5
- *How does it look? Is it different than the muons?*
- **Bonus task 1:** *can you plot the direct and reflected light separately?* (Modify your macro for CountDirect or CountReflected). *Are they similar? Is one more diffuse?*
- **Bonus task 2:** *What about X-ARAPUCAs? Can you plot the light they detected?* (Modify your macro to set !isPMT)

Root hint:
Root -l
.L PlotPhotonsYZ.cc
PlotPhotonsYZ("sim_muons_G4_hist.root ", 1)





Task 1 Summary

- That's the end of the first task looking at the simulated truth-level light information using the SBND geometry!
  - You can do the same with other detectors that are using LArSoft!
- You now have the tools to generate a simple light simulation using the single particle and can see what it does!
- What's next? Well real detector data doesn't have truth-level quantities so let's look reconstructing events.





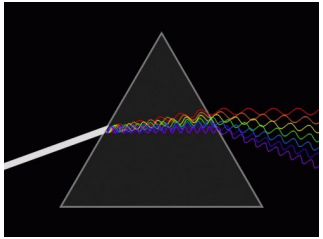
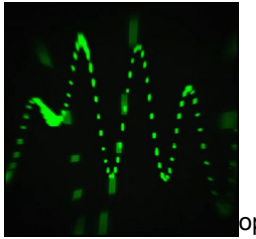
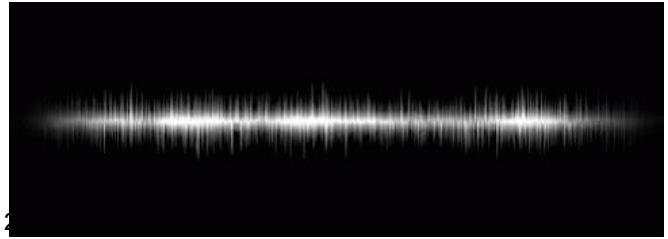
Part 2: Photo-detector response simulation and reconstruction





How can we simulate the photo-detector response?

- So far, we have generated photons at truth-level but we now need to model what a realistic photo-detector response would look like:
  - We mean making waveforms electronics response, noise, etc (all photo-detector dependent!)
- We will use the OpDetDigitizerSBND module to produce light waveforms!
- To do so we will use a .fcl that takes products from the generation stage. Copy it in your directory: Workshop/Photon/fcl/optical_tutorial_detsim.fcl
- This runs the *standard detsim* in SBND for both the PMTs and X-ARAPUCAs with their own electronics response models.
- We've attached an analyzer to look at the waveform detsim produces.

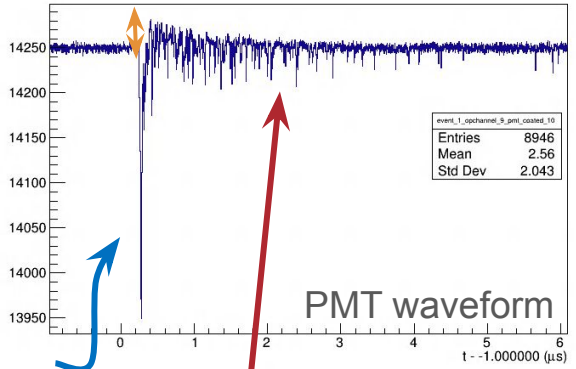
What does OpDetDigitizer module do?

```
producers:
{
  rns:       { module_type: "RandomNumberSaver" }
  opdaq:     @local::sbnd_opdetdigitizer
}
analyzers:
{
  wvfana: @local::wvf_ana
}
```

Each PE swapped for an electronics response (here constructed from parameters). Noise then added to the waveform

Analyzer will let us look at these waveforms

Overshoot due to electronics



Main light pulse

Late light pulses

Different responses for PMTs / X-ARAPUCA:

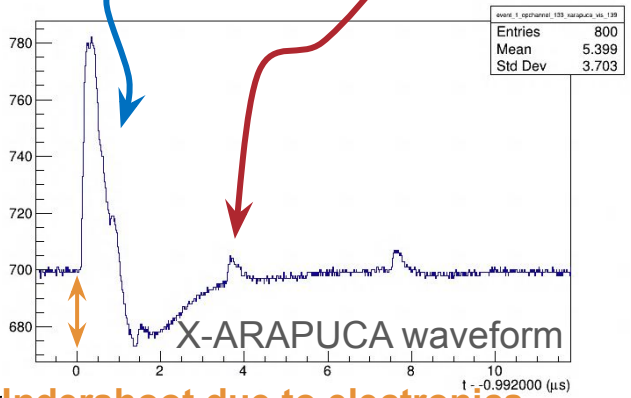
```
sbnd_digipmt_alg:
{
  # Parameters for ideal SER simulation
  PMTRiseTime:          3.8         #ns
  PMTFallTime:          13.7        #ns
  PMTMeanAmplitude:     0.9         #in pC
  TransitTime:          55.1        #ns
  PMTChargeToADC:       -25.97      #charge to adc factor

  # Parameters for test bench SER simulation
  PMTSinglePEmodel:       true      #false for ideal PMT response
  PMTDataFile:            "OpDetSim/digi_pmt_sbnd_v2int0.root"

  # Time delays
  TTS:                  2.4         #Transit Time Spread in ns
  CableTime:            135         #time delay of the 30 m long
```

```
sbnd_digiarapuca_alg:
{
  # module_type:        "DigiArapucaSBNDAlg"
  #Assume 25v bias with sensl c series SiPM. Values
  #Values of MaxAmplitude, BackTime and VoltageToAD

  ArapucaVoltageToADC:      151.5   #mV to ADC
  ArapucaBaselineRMS:       2.6     #in ADC counts
  ArapucaDarkNoiseRate:     10.0    #in Hz
  CrossTalk:                0.2     #20% probabili
  ArapucaBaseline:          1500    #ADC counts
  ArapucaPulseLength:       4000.0  #ns
  ArapucaPeakTime:          260.0   #ns
  ArapucaMeanAmplitude:     0.12    #mV
  ArapucaRiseTime:          9.0     #ns
```



Undershoot due to electronics

How will you run the optical reconstruction?

- Just as we have done with the generator and detsim, we will fun a new .fcl that has modules acting on the artroot files from the previous stage.
- After running OpDetDigtizer, our simulation should be as close as real detector data as possible based on how we modelled it and set the parameters!
- Now we can reconstruct the signals into products and variables that can be used in physics analyses with optical_tutorial_reco.fcl.
  - Copy it from Workshop/Photon/fcl directory in your src/sbndcode/sbndcode repo.
- This .fcl will run the standard SBND optical/light reconstruction with a couple of simple analyzers to look at reconstructed quantities directly.



What's in optical_tutorial_reco.fcl?

```
producers:
{
  ### optical deconvolution
  opdecopmt:          @local::SBNDOpDeconvolutionPMT

  ### optical hit finders
  ophitpmt:           @local::SBNDDecoOpHitFinderPMT

  ### flash finders
  opflashtpc0:        @local::SBNDDecoSimpleFlashTPC0
  opflashtpc1:        @local::SBNDDecoSimpleFlashTPC1
}

#Load analyzers
#hitdumpertree from sbndcode/Commissioning/HitDumper_module
#  Analyzer from larana/OpticalDetector
analyzers:
{
  oprecoanatpc0: @local::standard_opflashana
  oprecoanatpc1: @local::standard_opflashana
  hitdumpertree: @local::hitdumper
}
```

- **Reminder!** Producer modules produces new data products and analyzer modules analyzes the data product already made.
- Here the producer will run the PMT deconvolutio then produce Optical hits and flashes (OpHits & OpFlashes).
  - In SBND there are two OpFlash one per TPC!
- The analyzer modules will look at those OpHits & OpFlashes that the producer made.
- **Note:** *here only the PMTs are being looked at but X-ARAPUCA hit-finding is done analogously.*

But what is *flash matching* and how does it run?

- We got light waveforms and OpHits? *So what*? Do we need that for anything?
- The final stage here is to try to match between the reconstructed light objects and the reconstructed TPC ones for each event.
  - Remember that both light and charge and produced in a complementary manner!
- In SBND this is at “reco2”

```
reco2: [ rns
       , pandora, pandoraTrack, pandoraShower
       , pandoraCalo, pandoraPid
       , crthit, crttrack
       , opt0finder
```

```
sbnd_opt0_finder:
{
  module_type:     "SBNDOpT0Finder"
  OpFlashProducers: ["opflashtpc0", "opflashtpc1"]
  TPCs: [0, 1]
  SliceProducer:   "pandora"
```



  - The goal of this module is to make a prediction of the light based on the TPC track using the same simulation method as the LArG4 stage. This prediction is compared with each OpFlash objects produced to find the best match.
- You will run/learn about the flash matching and use its information in the reconstruction/analysis tutorial! Here we are showing you how they are made.

Task 2.1: Detector response simulation

- It is now time to run detsim on your muon from Task 1 as the input:

  ```
  lar -c optical_tutorial_detsim.fcl -s sim_muons_G4.root
  ```

  - Here we are providing the artroot file with -s and no need for -n 1 since there's only 1 event!
- Open the _hist.root file. You should see a wvfana tree containing waveforms for each photo-detectors (both PMTs and X-ARAPUCAs!).
- Have a look at them by double clicking on them. *Do you see differences between PMTs and X-ARAPUCAs? Why?*
- Try to find waveforms with lots of light and very little (how far the pulse goes away from the baseline in ADC). You can use **AllPhotons** tree from Task 1 to help choose a channel to look at.



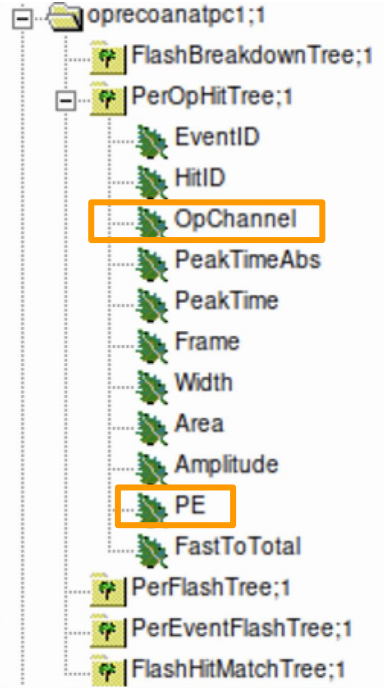
Task 2.2 Optical reconstruction - Hits



- Time to run the last .fcl of this tutorial! You want to run the optical_tutorial_reco.fcl on the output from the detsim stage as the source:

  lar -c optical_tutorial_reco.fcl -s sim_muons_G4_DetSim.root

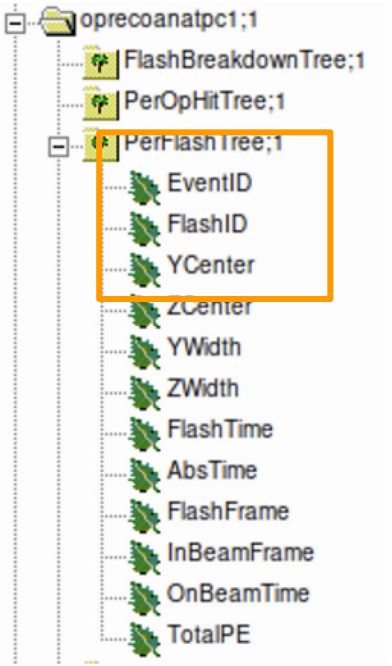
- Have a look at the OpHits from _hist.root (oprecoanatpc1/PerOpHitTree)
  - *Open the OpChannel and PE (photo-electron), do they make sense?*
- Let's plot the hits in the Y-Z distribution (like we did in Task 1):
- There's a second root macro to help you do this, copy it and run it
  - /Workshop/Photon/macro/PlotOpHit.YZ.cc and run it PlotPhotonsYZ.cc
- *How does this compare with the one you did with the truth-level quantities? Is the OpHintFinder performing well?*
- **Bonus:** *Do you notice something very different?* If you are on SBND you might know why ;).



Task 2.3 Optical reconstruction - Flashes



- You've just looked at the Hits, now let's look at the Flashes.
- Have a look at the oprecoanatpc1/PerFlashTree
- *Is the location of the flashes in the* Y-Z *plane where you expected them to be?* (hint compare to the OpHits)
- *What about the* flash widths*? Are they wider in* Y *or* Z*?*
- **Bonus:** do the same with the electrons (remember to run through detsim and reco stages before!)
  - *What are the main differences between the muon and electron flashes?*



Task 2 - Summary

- That’s the end of the second task looking at a simple light reconstruction in LArSoft using SBND as the detector.
- You now have the tools to reconstruct the light you generated earlier!
- What’s next? Light information is important in LArTPC analyses. It can be used to complement and enhance the TPC information through triggering on events and providing t0, event selection / background rejection, calorimetry, etc.

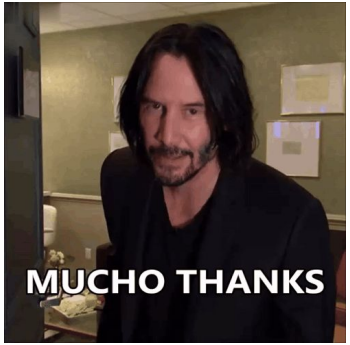




Light Tutorial Summary

- Congratulations! You have went through a basic example on how to generate light, apply detsim to simulate the detector response, and reconstruct them into quantities that can be used in physics analyses!
- There is so much more one can do with the light.
- You will have the chance to continue to use it in this workshop.
- If you want a DUNE example tutorial of light you can find it here:
  - https://indico.fnal.gov/event/73408/contributions/339034/attachments/197635/274930/202605_DUNESchool.pdf
- Thank you and we hope what you learned and use will be useful in the future!





Backup

Making plots

- The visual way:
  - root -l <my_file>_hist.root
  - new TBrowser()
  - Find the name of your .root file in the list
  - Select opanalyzer, select AllPhotons, right click on AllPhotons and select StartViewer.
  - You can plot any of the branches and apply cuts

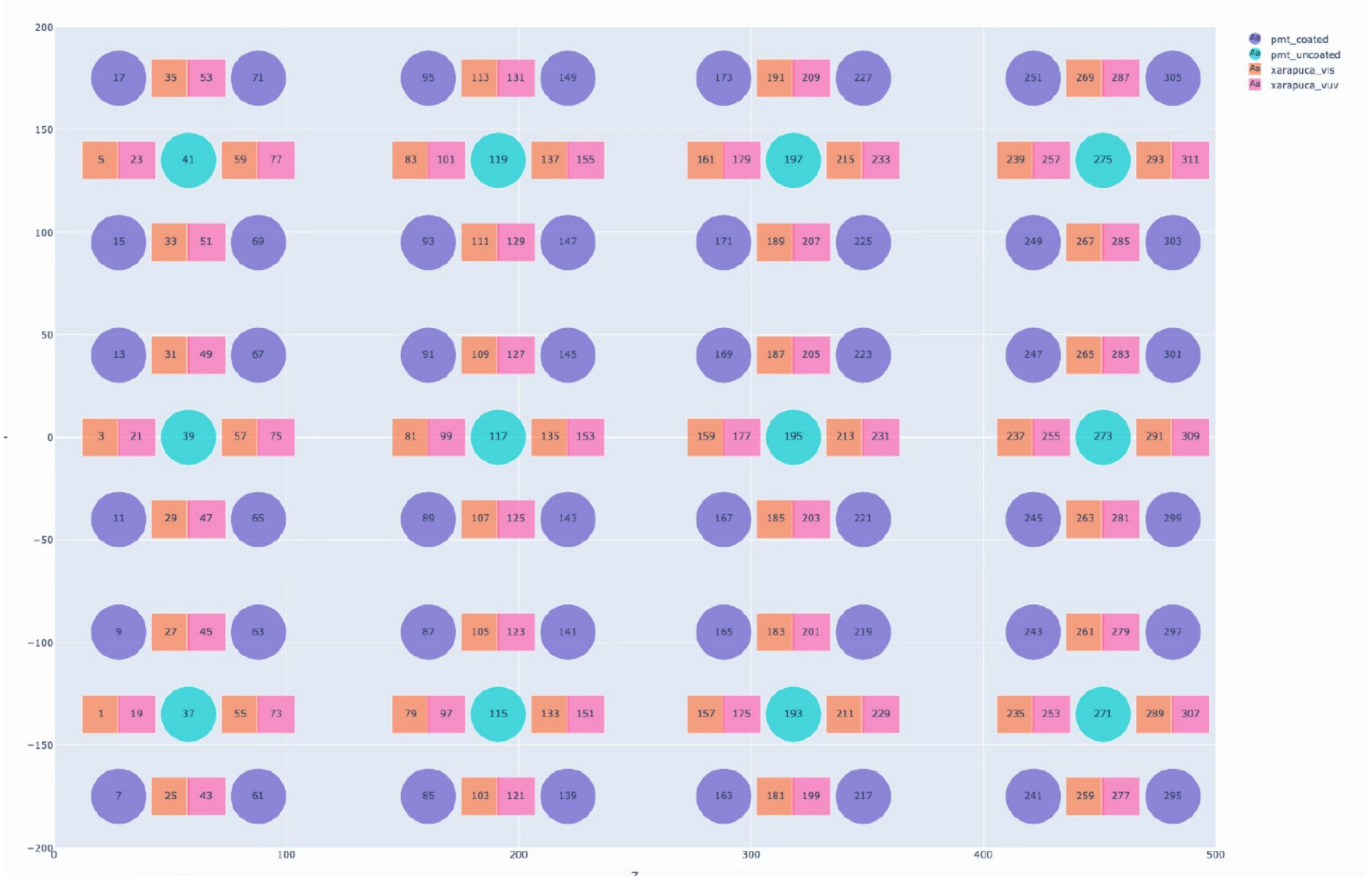
Making plots

- The script way:
  - Create a new file called myScript.C
  - In it write:

```
void myScript()
{
   TFile * fin = new TFile("<myfile>_hist.root","READ");
   TTree * mytree = (TTree *)fin->Get(“opanalyzer/AllPhotons");
   mytree->Draw("Time","");
}
```

  - Then to run: root -l myScript.C

Photon detector mapping: TPC 1, x > 0 cm



Photon detector mapping: TPC 0, x < 0 cm

