Grand Unification (of pixels software tools)

Stack of slides for future memory

How to fill-in this document so that it is useful

1. Fill in the github repository link

2. Take a look at the workflow goal (and fill free to comment on it)

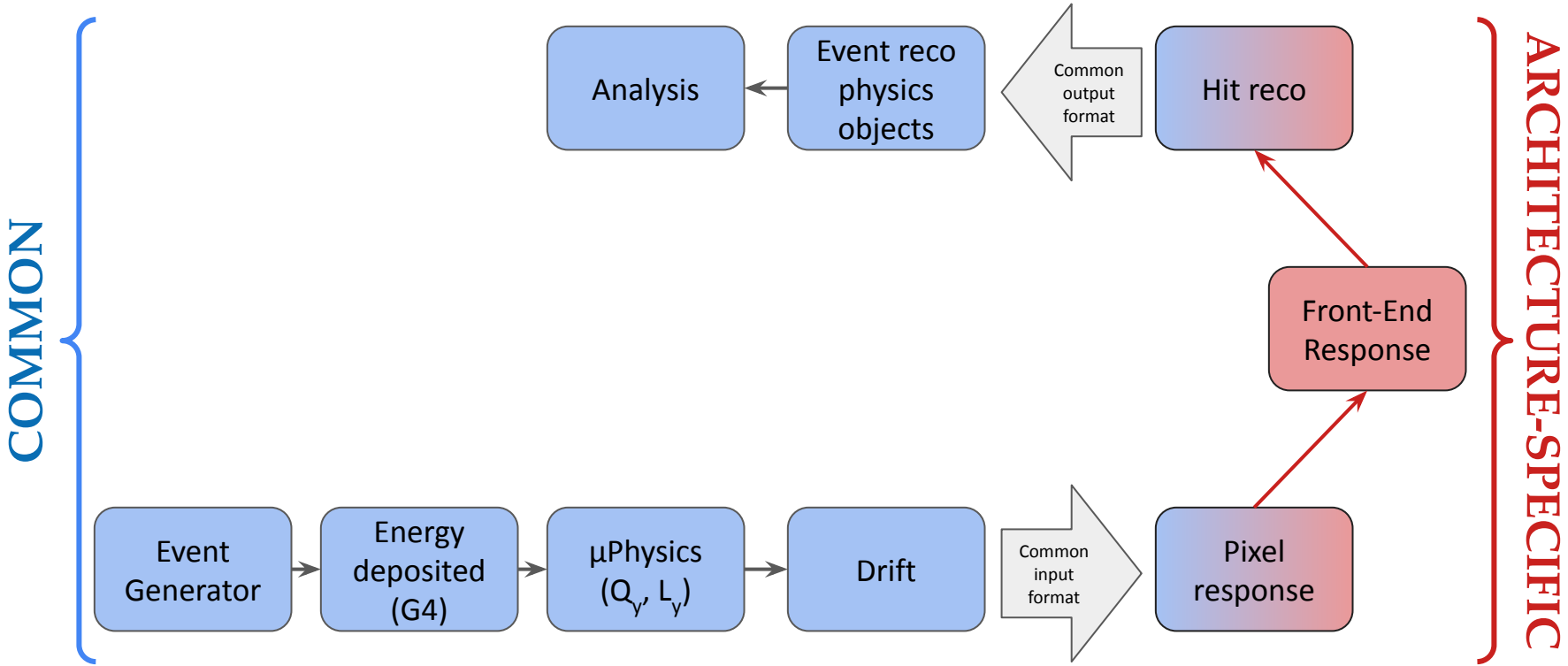
3. Fill-in the slides related to your architecture
   a. For each “logic block” in the workflow, try to input and output formats.
      We understand your workflow might not map exactly into this… do your best :)
   b. Make a special effort for the “pixel response” INPUT and the “hit reco” OUTPUT, these are our branching points:
      we would like the red blocks to be “architecture specific” and the blue blocks to be common.

4. Please note any connections to DUNE established practices

Current repositories

	Github link	Notes
GamPix		
LArPix		
production scripts	https://github.com/DUNE/2x2_sim	Quite developed. Developed for 2x2, also used by DUNE ND production.
geant4 wrapper	https://github.com/DUNE/edep-sim	
detector simulation	https://github.com/DUNE/larnd-sim	GPU accelerated
hit-level calibration	https://github.com/DUNE/ndlar_flow	
Solar		
Geant4 + detector simulation	https://github.com/SoLAr-Neutrinos/SOLAr-sim	
Q-Pix		

Workflow goal



GamPix: General notes (1)

Is there anything you'd like to add about the proposed workflow?

GamPix: Event Generator

Event Generator

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

GamPix: Energy Deposited

Energy deposited (G4)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

GamPix: Microphysics

µPhysics
(Q_y, L_y)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

GamPix: Drift

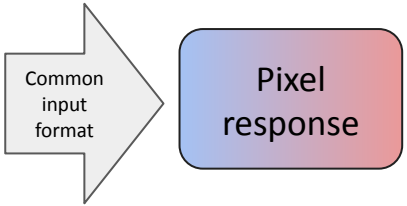
Drift

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

GamPix: Pixel response



Please describe how that's done in your framework

List Input format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

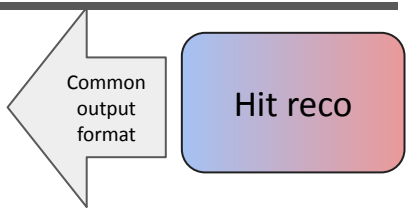
List Output format:

GamPix: Front-End-Response (architecture)

Front-End Response

But feel free to add info you think would be helpful for the community

GamPix: Hit Reconstruction



Please describe how that's done in your framework

List Input format:

List Output format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

GamPix: Event Reconstruction/Physics Obj

Event reco physics objects

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

GamPix: Event Reconstruction/Physics Obj

Analysis

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

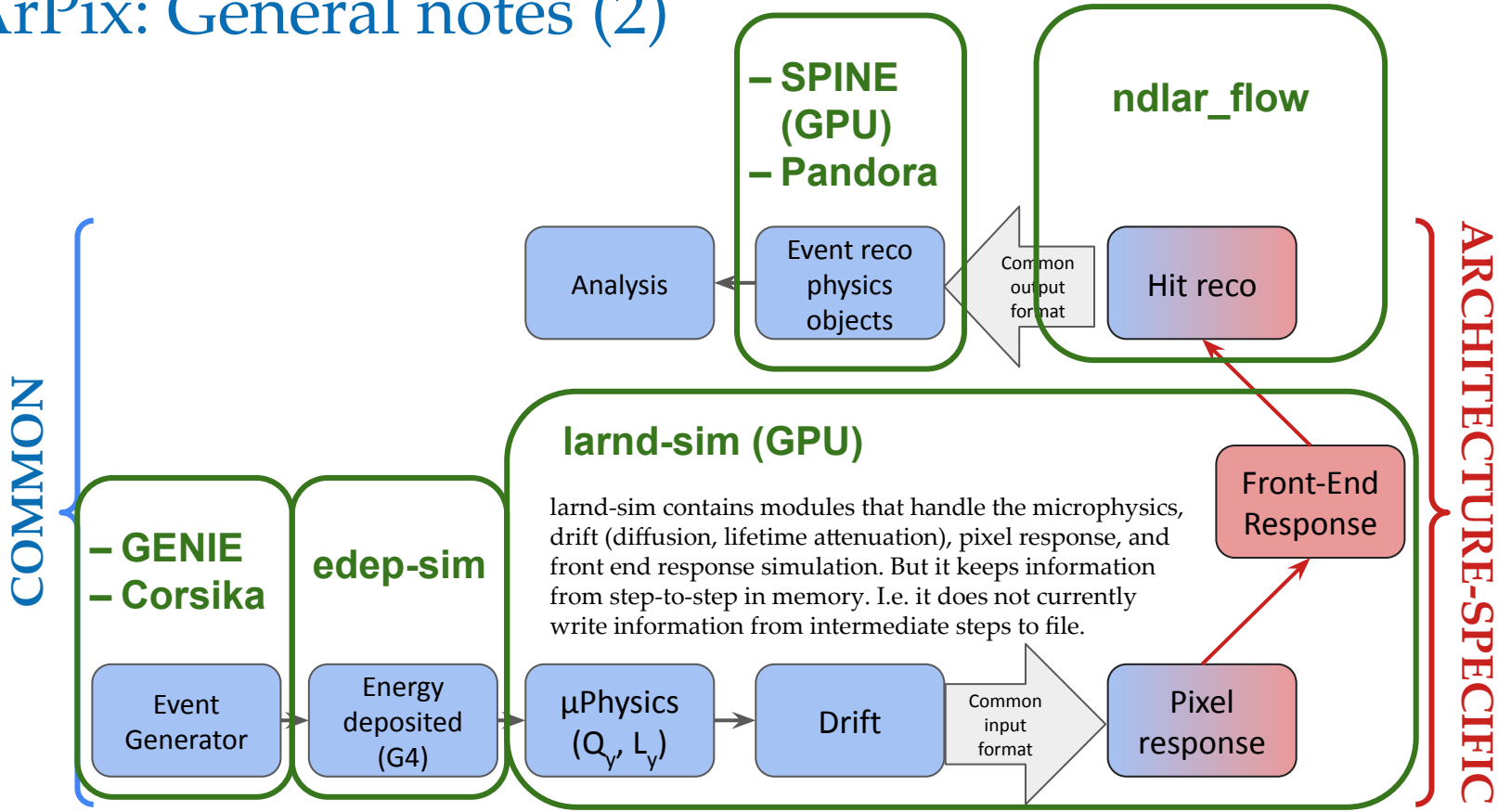
GamPix: General notes (2)

Is there anything else you'd like to add?

LArPix: General notes (1)

- Detailed tutorial on running the full production chain: https://github.com/DUNE/2x2_sim/wiki/Tutorial-on-running-2x2_sim-Apr2024
- We have had several 2x2 simulation production campaigns. Corresponding files are publicly accessible by navigating https://portal.nersc.gov/project/dune/data/2x2/simulation/productions/
- Handles for truth-level backtracking are retained throughout the entire chain ✅
- Tutorial on navigating "flow files" (i.e. output from our hit-level reconstruction and calibration code): https://github.com/DUNE/2x2_sim/wiki/Tutorial-for-reading-a-flow-or-a-larndsim-output-file

LArPix: General notes (2)



LArPix: Event Generator

Event Generator

Is there anything particularly good/peculiar for this block in your framework?

We typically use GENIE to generate neutrino events and Corsika to simulate cosmic activity.
For the latter, convert to GENIE RooTracker-like format with
https://github.com/soleti/corsika2RooTracker

List Input format: dk2nu format

List Output format: GENIE RooTracker format

LArPix: Energy Deposited

Energy deposited (G4)

Is there anything particularly good/peculiar for this block in your framework?

We use a tool developed for generic energy deposition simulation in particle detectors (albeit with LArTPC applications in mind).

We have a conversion script that we run on the edep-sim output root file, which formats the output for larnd-sim read in and writes it to a .hdf5 file:
https://github.com/DUNE/larnd-sim/blob/develop/cli/dumpTree.py

List Input format: GENIE RooTracker format

List Output format: https://github.com/DUNE/edep-sim?tab=readme-ov-file#output-tree-format

LArPix: Energy Deposited

Energy deposited (G4)

Is there anything particularly g

We use a tool developed for ger
LArTPC applications in mind).

We have a conversion script tha
larnd-sim read in and writes it t
https://github.com/DUNE/larnd

for

List Input format: GENIE RooT

List Output format: https://gith

Output array data types from conversion script (= input for larnd-sim). Segments are used directly by larnd-sim, while other info is retained for backtracking purposes.

```
segments_dtype = np.dtype([("event_id","u4"), ("vertex_id", "u8"), ("file_vertex_id", "u8"), ("segment_id", "u4"),
                           ("z_end", "f4"),("traj_id", "i4"), ("file_traj_id", "u4"), ("tran_diff", "f4"),
                           ("z_start", "f4"), ("x_end", "f4"),
                           ("y_end", "f4"), ("n_electrons", "u4"),
                           ("pdg_id", "i4"), ("x_start", "f4"),
                           ("y_start", "f4"), ("t_start", "f4"),
                           ("t0_start", "f8"), ("t0_end", "f8"), ("t0", "f8"),
                           ("dx", "f4"), ("long_diff", "f4"),
                           ("pixel_plane", "i4"), ("t_end", "f4"),
                           ("dEdx", "f4"), ("dE", "f4"), ("t", "f4"),
                           ("y", "f4"), ("x", "f4"), ("z", "f4"),
                           ("n_photons","f4")], align=True)

trajectories_dtype = np.dtype([("event_id","u4"), ("vertex_id", "u8"), ("file_vertex_id", "u8"),
                               ("traj_id", "i4"), ("file_traj_id", "u4"), ("parent_id", "i4"), ("primary", "?"),
                               ("E_start", "f4"), ("pxyz_start", "f4", (3,)),
                               ("xyz_start", "f4", (3,)), ("t_start", "f8"),
                               ("E_end", "f4"), ("pxyz_end", "f4", (3,)),
                               ("xyz_end", "f4", (3,)), ("t_end", "f8"),
                               ("pdg_id", "i4"), ("start_process", "i4"),
                               ("start_subprocess", "i4"), ("end_process", "i4"),
                               ("end_subprocess", "i4"), ("dist_travel", "f4")], align=True)

vertices_dtype = np.dtype([("event_id","u4"), ("vertex_id","u8"), ("file_vertex_id", "u8"),
                           ("x_vert","f4"), ("y_vert","f4"), ("z_vert","f4"),
                           ("t_vert","f4"), ("t_event","f4")], align=True)
```

LArPix: Microphysics

µPhysics
(Q_y, L_y)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

LArPix: Drift

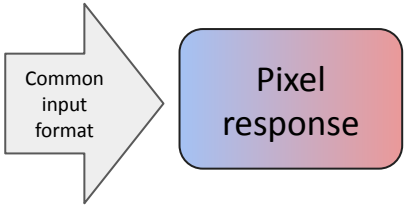
Drift

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

LArPix: Pixel response



Please describe how that's done in your framework

List Input format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

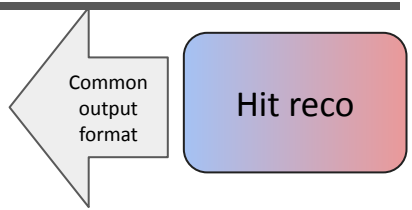
List Output format:

LArPix: Front-End-Response (architecture)

Front-End Response

But feel free to add info you think would be helpful for the community

LArPix: Hit Reconstruction



Please describe how that's done in your framework

This stage processes the raw data (either simulated or real) in several important ways:

1. Event building: groups primitive data objects into clusters corresponding to "events"
2. Maps the data products from software space to physics space (x,y,z)
   a. Including clock calibration, t0 assignment, drift projection
3. Applies readout calibrations to convert ADC word to detected charge
   a. May be detector/run/ASIC version specific
4. Applies detector calibrations to convert detected charge to energy deposited

Recent ad hoc additions at request of analyzers. Kevin W. is not a fan. This info is accessible via reference datasets.

x	f8, pixel x location [cm]
y	f8, pixel y location [cm]
z	f8, pixel z location [cm]
t_drift	u8, drift time [tick = 100ns]
ts_pps	f8, PPS packet timestamp [tick = 100ns]
io_group	u8, io group ID (PACMAN number)
io_channel	u8, io channel ID (related to PACMAN number & PACMAN UART Number)
chip_id	u8, chip_id on tile
channel_id	u8, channel_id on single chip (0-63)
Q_raw	f8, hit charge [ke-] (uncalibrated for ADC droop)
Q	f8, calibrated hit charge with ADC droop corrections
E	f8, hit energy [MeV]

Hit objects have associated "reference datasets" that indicate event groups, truth backtracking (for MC), and more.

List Input format: packet objects (LArPix data products)

List Output format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

LArPix: Event Reconstruction/Physics Obj

Event reco physics objects

Is there anything particularly good/peculiar for this block in your framework?

SPINE and Pandora eventually fill their information into CAFs (Common Analysis Format/Files(?)), which are a DUNE (and other experiment) standard.

List Input format:

List Output format:

LArPix: Event Reconstruction/Physics Obj

Analysis

Is there anything particularly good/peculiar for this block in your framework?

Honestly, at this stage this is mostly happening in jupyter notebooks and the like. CAFAna is being explored to analyze CAFs that are written by event reconstruction packages. But a lot of the work at this stage is lower level analysis and done “on the side”. This code is often stored locally or spread across personal GitHubs.

List Input format:

List Output format:

LArPix: General notes (3)

Is there anything else you'd like to add?

The information in the preceding slides is trying to highlight the major points about the different steps and the data formats of the most important objects in between. **It is by no means complete.**

The best way to understand the full data formats is to inspect some example files:

File type	Link to file download area
GENIE output	https://portal.nersc.gov/project/dune/data/2x2/simulation/productions/MiniRun5_1E19_RHC/MiniRun5_1E19_RHC.genie.nu/GTRAC/
edep-sim output	https://portal.nersc.gov/project/dune/data/2x2/simulation/productions/MegaRun5_1E20_RHC/run-edep-sim/MegaRun5_1E20_RHC.edep.nu/EDEPSIM/0000000/
larnd-sim input	https://portal.nersc.gov/project/dune/data/2x2/simulation/productions/MiniRunF6.3_1E19_FHC/run-convert2h5/MiniRunF6.3_1E19_FHC.convert2h5/EDEPSIM_H5/0000000/
larnd-sim output = ndlar_flow input	https://portal.nersc.gov/project/dune/data/2x2/simulation/productions/MiniRunF6.3_1E19_FHC/run-larnd-sim/MiniRunF6.3_1E19_FHC.larnd/LARNDSIM/0000000/
ndlar_flow output	https://portal.nersc.gov/project/dune/data/2x2/simulation/productions/MiniRun6.2_1E19_RHC/MiniRun6.2_1E19_RHC.flow/FLOW/0000000/

Solar: General notes (1)

Is there anything you'd like to add about the proposed workflow?

- We may want to start thinking about a shared procedure for the propagation of optical photons

Solar: Event Generator

Event Generator

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

Solar: Energy Deposited

Energy deposited (G4)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

Solar: Microphysics

µPhysics
(Q_ν, L_ν)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

Solar: Drift

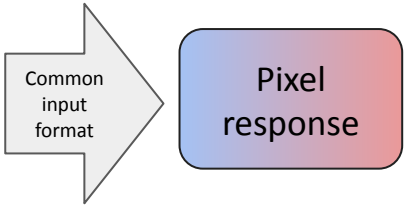
Drift

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

Solar: Pixel response



Please describe how that's done in your framework

List Input format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

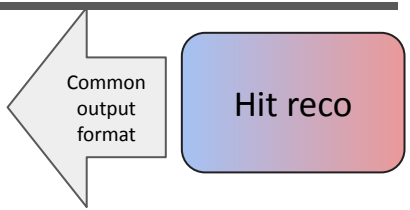
List Output format:

Solar: Front-End-Response (architecture)

Front-End Response

But feel free to add info you think would be helpful for the community

Solar: Hit Reconstruction



Please describe how that's done in your framework

List Input format:

List Output format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

Solar: Event Reconstruction/Physics Obj

Event reco physics objects

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

Solar: Event Reconstruction/Physics Obj

Analysis

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

Solar: General notes (2)

Is there anything else you'd like to add?

QPix: General notes (1)

Is there anything you'd like to add about the proposed workflow?

- We may want to start thinking about a shared procedure for the propagation of optical photons

QPix: Event Generator

Event Generator

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

QPix: Energy Deposited

Energy deposited (G4)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

QPix: Microphysics

µPhysics
(Q_y, L_y)

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

QPix: Drift

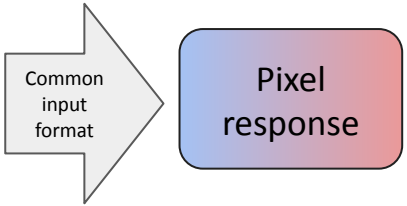
Drift

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

QPix: Pixel response



Please describe how that's done in your framework

List Input format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

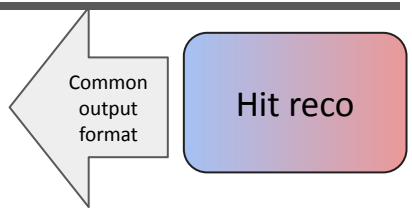
List Output format:

QPix: Front-End-Response (architecture)

Front-End Response

But feel free to add info you think would be helpful for the community

QPix: Hit Reconstruction



Please describe how that's done in your framework

List Input format:

List Output format (THIS IS SUPER IMPORTANT IN THIS BLOCK):

QPix: Event Reconstruction/Physics Obj

Event reco physics objects

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

QPix: Event Reconstruction/Physics Obj

Analysis

Is there anything particularly good/peculiar for this block in your framework?

List Input format:

List Output format:

QPix: General notes (2)

Is there anything else you'd like to add?