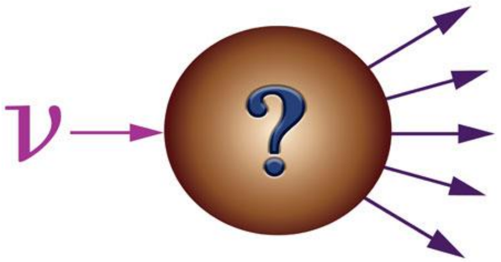
resourcefulmanager.com
"NOW TURN YOUR ATTENTION TO ITEM NO. 167 ON YOUR AGENDA ..."

# LAr Software Ideas for Merging

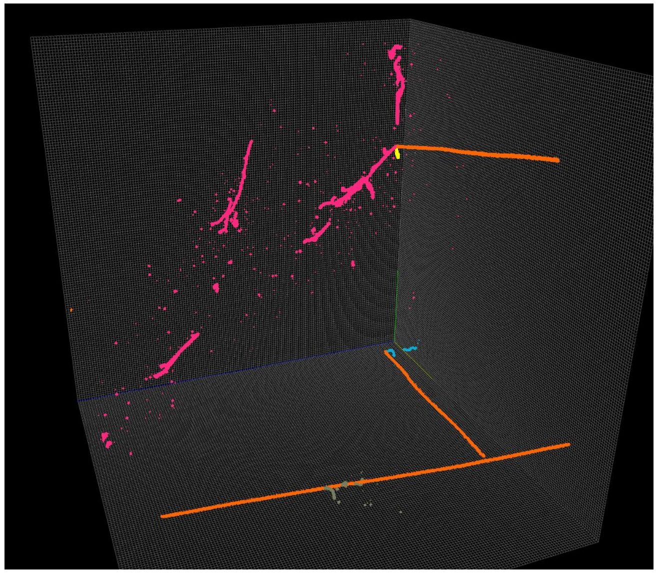
Kazu @ SLAC

# Simulation workflow

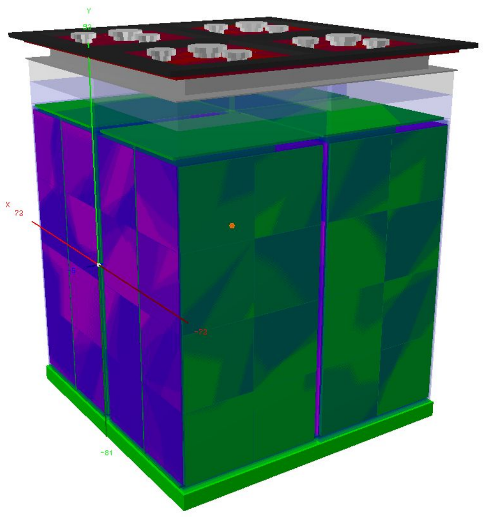
- Event generation
- Particle tracking
- Detector response



Neutrino-Nucleus interactions in the detector and surrounding materials



Tracking of generated and secondary particles in the detector and surrounding materials

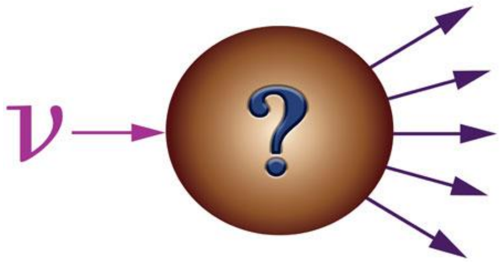


Compute charge/light production, propagate to sensitive detector elements, produce readout output

## Simulation workflow

- Event generation
- Particle tracking
- Detector response

No need to be merged. But better to have a stand-alone, shared event generator library. Also better to have a stand-alone, shared data product library (this part is reused by particle tracking simulation).



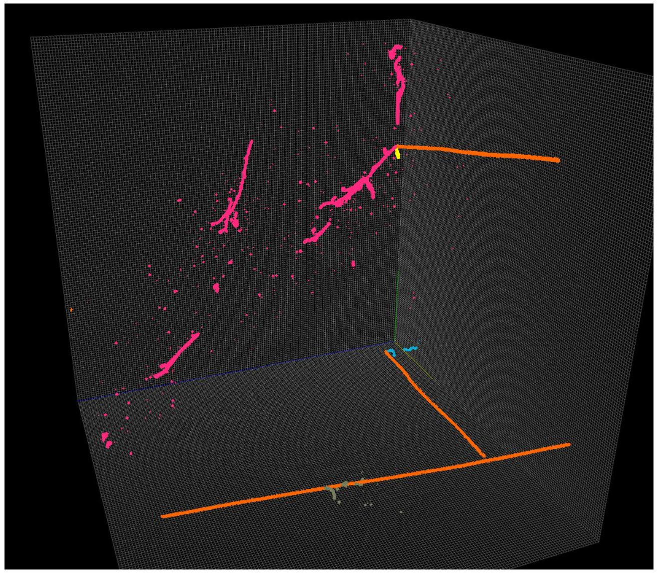
Neutrino-Nucleus interactions in the detector and surrounding materials

## Simulation workflow

- Event generation
- Particle tracking
- Detector response

**No need, but better to be merged.** **Best:** a stand-alone, shared library. Also better to have a stand-alone. **Better**: shared data product library (this part is reused by particle tracking simulation).

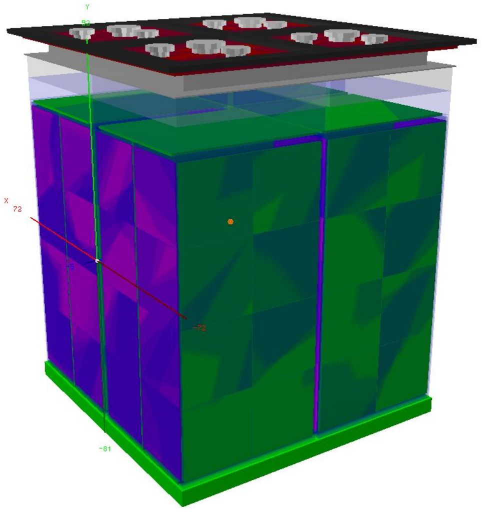
**Should be merged.** Particle tracking and energy deposition. Store particle type, dE and dX. **Need**: unified GDML, common Physics List, common version underlying software (ROOT, GeGeD-ND, Geant4). **Better**: common input & output data product library.


Tracking of generated and secondary particles in the detector and surrounding materials

## Simulation workflow

- Event generation
- Particle tracking
- Detector response

**No need to be merged.**



Compute charge/light production, propagate to sensitive detector elements, produce readout output

**No need, but better to be merged.** **Best:** a stand-alone, shared library. Also better to have a stand-alone. **Better**: shared data product library (this part is reused by particle tracking simulation).

**Should be merged.** Particle tracking and energy deposition. Store particle type, dE and dX. **Need**: unified GDML, common Physics List, common version underlying software (ROOT, GeGeD-ND, Geant4). **Better**: common input & output data product library.

## Simulation workflow

- Event generation
- Particle tracking
- Detector response

**No need to be merged.**

**No need, but better to be merged.** **Best:** a stand-alone, shared library. Also better to have a stand-alone. **Better**: shared data product library (this part is reused by particle tracking simulation).

**Should be merged.** Particle tracking and energy deposition. Store particle type, dE and dX. **Need**: unified GDML, common Physics List, common version underlying software (ROOT, GeGeD-ND, Geant4). **Better**: common input & output data product library.

**Extra Thoughts**

Algorithms and data products should be implemented in a separate library than event processing framework (i.e. art) so that implementation can be done outside the framework (e.g. ArgonBox, bare Geant4)

## Simulation workflow

- Event generation
- Particle tracking
- Detector response

**No need to be merged.**

**No need, but better to be merged.** **Best:** a stand-alone, shared library. Also better to have a stand-alone. **Better**: shared data product library (this part is reused by particle tracking simulation).

**Should be merged.** Particle tracking and energy deposition. Store particle type, dE and dX. **Need**: unified GDML, common Physics List, common version underlying software (ROOT, GeGeD-ND, Geant4). **Better**: common input & output data product library.

**Extra Thoughts**

Algorithms and data products should be implemented in a separate library than event processing framework (i.e. art) so that implementation can be done outside the framework (e.g. ArgonBox, bare Geant4)

**A solution**: combined Geatn4 driver with modularized algorithm and data product libraries.

## Simulation workflow

- Event generation
- Particle tracking
- Detector response

**No need to be merged.**

**Extra Thoughts**

Algorithms and data products should be implemented in a separate library than event processing framework (i.e. art) so that implementation can be done outside the framework (e.g. ArgonBox, bare Geant4)

**No need, but better to be merged.** **Best:** a stand-alone, shared library. Also better to have a stand-alone. **Better**: shared data product library (this part is reused by particle tracking simulation).

**Should be merged.** Particle tracking and energy deposition. Store particle type, dE and dX. **Need**: unified GDML, common Physics List, common version underlying software (ROOT, GeGeD-ND, Geant4). **Better**: common input & output data product library.

**A solution**: combined Geatn4 driver with modularized algorithm and data product libraries.

# Practical Implementation?

- Start a place holder dune-nd-sim repository
  - Implement a common Geant4 driver (new largeant)

## Practical Implementation?

- Start a place holder dune-nd-sim repository
  - Implement a common Geant4 driver (new largeant)
- Develop detector response in a sub-detector repository
  - dune-nd-mpd-sim, dune-nd-lar-sim, etc…
  - Driver code can be in each repository OR dune-nd-sim.

## Practical Implementation?

- Start a place holder dune-nd-sim repository
  - Implement a common Geant4 driver (new largeant)
- Develop detector response in a sub-detector repository
  - dune-nd-mpd-sim, dune-nd-lar-sim, etc...
  - Driver code can be in each repository OR dune-nd-sim.
- Comments
  - **Major work**: a common Geant4 driver code
  - **Minor work**: code modularization in each sub-detector

## Practical Implementation?

- Start a place holder dune-nd-sim repository
  - Implement a common Geant4 driver (new largeant)
- Develop detector response in a sub-detector repository
  - dune-nd-mpd-sim, dune-nd-lar-sim, etc...
  - Driver code can be in each repository OR dune-nd-sim.
- Comments
  - **Major work**: a common Geant4 driver code
  - **Minor work**: code modularization in each sub-detector
  - **GArSoft**: it looks like an old LArSoft branch-off before modularization of libraries (i.e. not a brand-new development from art). It looks better to modularize libs instead of keep as a separate software. Challenge might be Geometry, but our recent development on generalizing readout element type might solve this issue.

## Simulation Eco-system

- **Redmine** for code repository
- Software distribution
  - **CVMFS** or **container** (Singularity/Docker)
- **Jenkins** build + **release process** (human needed)

## Simulation Eco-system

- **Redmine** for code repository
- Software distribution
  - **CVMFS** or **container** (Singularity/Docker)
- **Jenkins** build + **release process** (human needed)

## Reconstruction … currently all outside LArSoft

- Probably no need to merge, but if useful, we should share
- Libraries (HPC-ready, either GPU or KNL, also runs on CPU)
  - MAGMA (+sparsehash-dev) for linear algebra, CUDA for GPU kernels, AVX-512 (optional) for KNL kernels, Open-MP for many-core multi-threading, MPI+Horovod for data broadcasting, pytorch/sci-kit/OpenCV for ML/computer-vision algorithms
- Eco-system
  - **Github** (Free and superior to redmine in many aspect if not all)
  - **Travis-CI** (Free)
  - **Docker/Singularity** hub with build auto-trigger (Free)