Geometry support in Xerosere/Phlex

Brett Viren

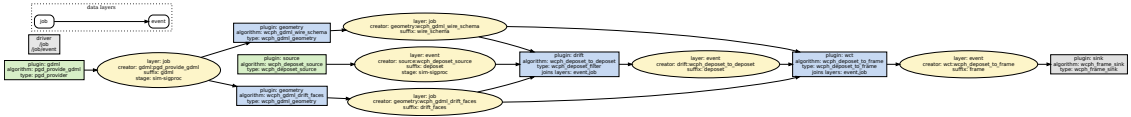
November 25, 2026

My goals for geometry support in Phlex

- Supply Wire-Cell Toolkit (WCT) islands inside Phlex with geometry info.
- Support larsoft-style GDML but work toward more standard GDML.
- Progressive interpretation: from general purpose to specific consumer needs.
- Be cautious of memory usage, especially related to “wires” representations.

Basic design

Phlex *providers* and *algorithms*[1] producing geometry products as a small **job**-layer graph.



1. A **job**-layer *provider* supplies GDML data product (first yellow ellipse).
2. It is converted by two more **job**-layer algorithms to two **WCT-specific** objects.
3. These **two geom objects** + **event**-layer object goes to the WCT *depo-drift* algorithm.
4. These **two geom objects** + **drifted depos** go to WCT *depo-to-frame* algorithm.

New geometry consumers can provide new gdml-object transformation algorithms.

- But, DUNE should look to develop a *geometry commons* (eg, wire/channel map).

[1] Not *resources*. Thanks to Kyle for this guidance.

New package: `gdml-objects`

General purpose package (no Phlex) provides:

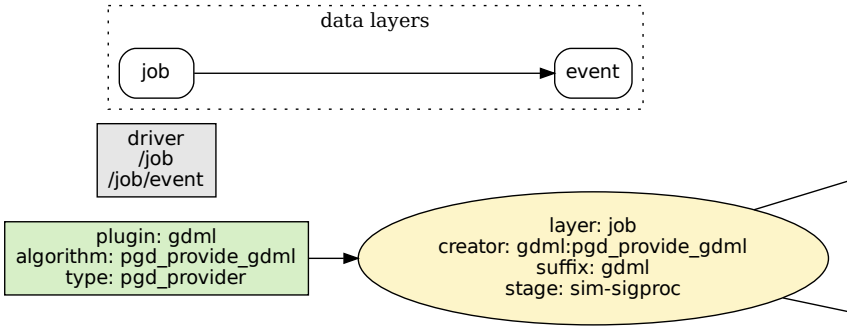
- GDML parser using `expat` and `hep-units` libraries.
- Evaluates GDML strings with `parse()` from `hep-units`.
- Produces C++ `struct` types realizing GDML's XML schema.

```cpp
#include "gdml/objects.hpp"
auto doc = gdml::read("dunevd10kt_3view_30deg_v6_refactored.gdml");
const auto& tpc = doc.volume("volTPC");
for (auto pv : tpc.physvols) {
    const auto& p = doc.physvols[pv];
    // doc.volumes[p.volume].name, p.position.value (mm), p.rotation.value (rad)
}
double halfz = doc.solid("TPCWireZ").at("z") / 2;   // GDML tube z is the full length
```

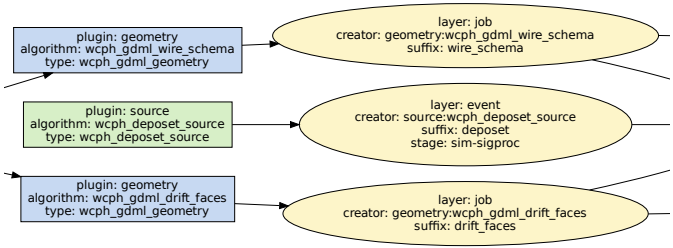
New package: `phlex-gdml`

A Phlex *provider* of **GDML objects** populating the **job** layer.

- A generic package (no DUNE), depends only on `phlex` and `gdml-objects`.



Updated package: `wire-cell-phlex`



Directly feeds Wire-Cell with wire and drift boundary geometry.

- No more WCT "wires file" needed nor drift boundaries defined in WCT Jsonnet.
  - A new `larwirecell` version also brings this feature to *art* / LArSoft.
- **Internally replicates larsoft wire/channel map and ID ordering convention**.
  - (will come back to this point)

Guide for developers of algorithms needing novel geometry types

Step 1: Make a new *converter algorithm* consuming `gdml-objects`.

There is no step 2.

However, we should consider developing a *geometry commons* that abstracts across our various types of consumers.

- Hard mode: the full set of geometry consumers have yet to be ported/invented.

A *geometry commons*



Questions

What geometry info is specific to one consumer and what generalizes to multiple consumers?
How can we avoid memory bloat due to copies (almost exclusively wires are the problem)?
How can we share algorithmic conventions such as channel maps and sub-detector numbering?

Minimal answers so far

The **job**-layer `gdml-objects` and small refinement graph is (imo) generically appropriate.

Going forward

Geometry-consumer developers should get together to define the nature of a *geometry commons*.
The "private" wire/channel map for WC/Phlex may then be factored out to the *commons*.

`edep-sim-phlex` and GDML

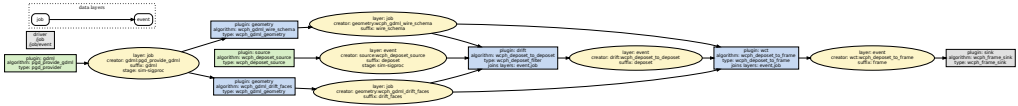
`edep-sim` is integrated and working in xerosere/phlex

- Can now read larsoft-compatible GDML, warts and all, **directly**.
- In Phlex, no good reason I see to force it to go through `gdml-objects`.

I feel this holds true for any other Geant4/Phlex integration that people may try.

Summary

- `gdml-objects` light-weight GDML parser and object model following GDML schema.
- `phlex-gdml` give generic and extensible way to access GDML geometry in Phlex.
- `wire-cell-phlex` now uses it to feed WCT, no more wires files!
- DUNE should consider what a *geometry commons* Phlex subgraph might look like.



FIN

Backups

Some questions for the geometry commons to answer

- What wire segments feed a given channel and vice versa?
- How do I iterate channels in order of wire attachments?
- How do I iterate channels in wire-in-plane order?
- Where is APA1 and what APAs are next to it?
- What is the nearest wire to a given point?

ID numbering contract - especially for Far Detector

Larsoft has several wire/channel mapping and “ordering” classes that define other ID numbers.

- Ported to `wire-cell-phlex` to give “private” larsoft-compatible numbering.
- May factor these out if/when more general data products can be defined.

But, can we formalize these conventions and then code to that contract?

- Reduce, remove variants - a single numbering rule that works everywhere is best

Wire-Cell achieves that[2].

- Formal enough to make a lawyer blush.
- Rules stated to apply to every tomographic LArTPC detector.
- But, currently doesn’t cover a wire/channel map convention.

Proposal: define a **DUNE offline ID schema** in a similarly comprehensive document.

[2] https://github.com/WireCell/wire-cell-toolkit/blob/master/util/docs/wire-schema.org

Can we move to a more standard GDML than LArSoft's variant?

- Volume name stenography (`volCryostat`, `volTPC`, other `vol*`, `*Sensitive*`)
  - `<auxiliary auxtype="Role" auxvalue="TPCActive"/>`
- Fixed nesting defines roles.
  - `<auxiliary auxtype="Role" auxvalue="TPCActive"/>`
- Some assumed volume shapes
  - define all shapes explicitly
- Detector identity inferred from the GDML file name.
  - `<setup name="..." version="...">`
- Wires vs no wires version
  - `<physvol><file name="wires.gdml"/>`, follow or not via a Role
- A `SensDet` aux gives name of data product class
  - Give label.
- Optical and material properties not taken from GDML
  - `<property>`, `<opticalsurface>`, `<skinsurface>`, `<bordersurface>`
- E-field set via FCL
  - use `<auxiliary>`, though there is no GDML standard for the `auxtype`.