



## Geometry refactoring release candidate

*(part 5 of 5)*

Kyle J. Knoepfel  
LArSoft coordination meeting  
20 February 2024

# Previous talks on geometry refactorization

- 9/20/22 **Status of Geometry service changes to accommodate pixel readouts**  
<https://indico.fnal.gov/event/56265/>
- 11/29/22 **Status of Geometry service changes to accommodate pixel readouts (part 2)**  
<https://indico.fnal.gov/event/57355/>
- 2/21/23 **Disentangling ChannelMapAlg from GeometryCore**  
*or Status of Geometry service changes to accommodate pixel readouts (part 3)*  
<https://indico.fnal.gov/event/58509/>
- 9/19/23 **Geometry refactorization to support pixel readouts**  
*Status report (part 4)*  
<https://indico.fnal.gov/event/61411/>

# Motivation

- **LArSoft will support pixel geometries**
- **Significant adjustments to `larcorealg` were required**  
Much of the geometry code assumed wire readouts
- **This talk covers those big adjustments.**
  - I will present the general tasks required to support pixel geometries.
  - The C++ and configuration adjustments required for LArSoft v10

# Work plan to support pixel geometries

## 1. Remove deprecated code (released)

- Replace bare integer types with geometry IDs

```
728 - const geo::CryostatGeo& cryostat = geom->Cryostat(cryo);  
728 + const geo::CryostatGeo& cryostat = geom->Cryostat(geo::CryostatID(cryo));
```

- Keep only `geo::Point_t` and `geo::Vector_t` vector types

## 2. Adjust iteration patterns (released)

```
342 - for (const auto& tpcid : geom->IterateTPCIDs()) {  
342 + for (const auto& tpcid : geom->Iterate<geo::TPCID>()) {
```

- Additional adjustments required to support PyROOT usage.

# Work plan to support pixel geometries

## 1. Remove deprecated code (released)

- Replace bare integer types with geometry IDs

```
728 - const geo::CryostatGeo& cryostat = geom->Cryostat(cryo);  
728 + const geo::CryostatGeo& cryostat = geom->Cryostat(geo::CryostatID(cryo));
```

- Keep only geo::Point\_t and geo::Vector\_t vector types

## 2. Adjust iteration patterns (released)

```
342 - for (const auto& tpcid : geom->IterateTPCIDs()) {  
342 + for (const auto& tpcid : geom->Iterate<geo::TPCID>()) {
```

- Additional adjustments required to support PyROOT usage.

## 3. Disentangle ChannelMapAlg and GeometryCore (done)

- Alters initialization sequence of Geometry and ExptGeoHelperInterface services
- Includes ownership adjustment of GeoObjectSorter

# Work plan to support pixel geometries

**3. Extract PlaneGeo objects from TPCGeo (done)**

**4. Introduce readout geometry classes (done)**

- Rename ChannelMapAlg → WireReadoutGeom
- Rename ExptGeoHelperInterface → WireReadout
- Refactor builders and sorters

**5. Separate AuxDetGeometryCore elements from GeometryCore (done)**

# Work plan to support pixel geometries

## 3. Extract `PlaneGeo` objects from `TPCGeo` (done)

## 4. Introduce readout geometry classes (done)

- Rename `ChannelMapAlg` → `WireReadoutGeom`
- Rename `ExptGeoHelperInterface` → `WireReadout`
- Refactor builders and sorters

## 5. Separate `AuxDetGeometryCore` elements from `GeometryCore` (done)

## 6. Validation (where things are now)

- Demonstrate that changes made above do not introduce physics changes.
- ***Need your help!***

## 7. Support pixel geometries (not done)

- Skeleton interface developed by Tom Junk

# geo::WireReadoutGeom (formerly known as geo::ChannelMapAlg)

Until now, most users have not directly interacted with `geo::ChannelMapAlg`.

- It has been a LArSoft provider that is owned by `geo::GeometryCore`.
- The specific channel-map provider is loaded by the `ExptGeoHelperInterface` service configured for the *art* job.



# geo::WireReadoutGeom (formerly known as geo::ChannelMapAlg)

Until now, most users have not directly interacted with `geo::ChannelMapAlg`.

- It has been a LArSoft provider that is owned by `geo::GeometryCore`.
- The specific channel-map provider is loaded by the `ExptGeoHelperInterface` service configured for the `art` job.

With v10, wire-specific information is provided by the `geo::WireReadoutGeom` provider (or `geo::WireReadout` `art` service) (e.g.):

```
46 - art::ServiceHandle<geo::Geometry const> geo;
47 - unsigned int nplanes = geo->Nplanes();
46 + auto const& wireReadoutGeom = art::ServiceHandle<geo::WireReadout const>()->Get();
47 + unsigned int nplanes = wireReadoutGeom.Nplanes();
```

Iteration through planes and wires is supported by `geo::WireReadout(Geom)` (e.g.):

```
145 - for (auto const& plane : geo->Iterate<geo::PlaneGeo>(tpcid)) {
145 + for (auto const& plane : wireReadoutGeom.Iterate<geo::PlaneGeo>(tpcid)) {
146 146     maxwire = (plane.Nwires() - 1 > maxwire) ? plane.Nwires() - 1 : maxwire;
147 147 }
```

# geo::WireReadoutGeom (formerly known as geo::ChannelMapAlg)

Configuration change:

```
59 59
60 - lartpcdetector_geometry_helper: {
61 -   service_provider : StandardGeometryHelper
62 + lartpcdetector_readout: {
63 +   service_provider : StandardWireReadout
64 }
65 }
66
67 lartpcdetector_geometry_services: {
68   GeometryConfigurationWriter: {}
69   Geometry: @local::lartpcdetector_geometry
70 -   ExptGeoInterfaceHelper: @local::lartpcdetector_geometry_helper
71 +   WireReadout: @local::lartpcdetector_readout
72 }
```

The general *art* configuration looks like:

```
services.WireReadout: {
  service_provider: <ExperimentSpecificWireReadout>
  SortingParameters: { tool_type: MyWireReadoutSorter ... }
  ...
}
```

# geo::AuxDetGeometryCore

With LArSoft v10, users must access auxiliary geometry information through the `geo::AuxDetGeometryCore` provider (or `geo::AuxDetGeometry art` service).

- This separate provider (and service) has existed for a while, but not always used.

In C++ code:

```
74 - fGeo->FindAuxDetSensitiveAtPosition(worldPos, adNum, svNum);  
71 + fAuxDetGeom->FindAuxDetSensitiveAtPosition(worldPos, adNum, svNum);
```

Configuration:

```
services.AuxDetGeometry: {  
  SortingParameters: { tool_type: MyAuxDetSorter ... }  
  ReadoutInitializer: { tool_type: MyAuxDetInitializer ... }  
  ...  
}
```

# Changes to the sorters

Each sorter class contains virtual functions that, when overridden, provide the sorting behavior desired for a given level of the geometry hierarchy.

There are three sorter base classes:

- `geo::GeoObjectSorter` (core geometry)
- `geo::WireReadoutSorter` (wire-readout geometry, [new with v10](#))
- `geo::AuxDetGeoObjectSorter` (auxiliary geometry)

# Changes to the sorters

Each sorter class contains virtual functions that, when overridden, provide the sorting behavior desired for a given level of the geometry hierarchy.

There are three sorter base classes:

- `geo::GeoObjectSorter` (core geometry)
- `geo::WireReadoutSorter` (wire-readout geometry, **new with v10**)
- `geo::AuxDetGeoObjectSorter` (auxiliary geometry)

With v10, each sorting algorithm must model the *Compare* requirement as specified by the C++ standard template library and as used by the `std::sort` algorithm:

- [https://en.cppreference.com/w/cpp/named\\_req/Compare](https://en.cppreference.com/w/cpp/named_req/Compare)
- <https://en.cppreference.com/w/cpp/algorithm/sort>

Sorting based on comparing elements, not based on using the entire container.

Details in documentation in preparation.

## Other interface changes

- `geo::AuxDetGeometryCore` has customizable initialization.

# Other interface changes

- `geo::AuxDetGeometryCore` has customizable initialization.
- Changes in enumerations.

```
127 - typedef enum coordinates {
128 -     kXCoord, ///< X coordinate.
129 -     kYCoord, ///< Y coordinate.
130 -     kZCoord ///< Z coordinate.
131 - } Coord_t;
122 + enum class Coordinate { X, Y, Z };
```

```
162 - typedef enum driftdir {
163 -     kUnknownDrift, ///< Drift direction is unknown.
164 -     kPos,          ///< Drift towards positive values.
165 -     kNeg,          ///< Drift towards negative values.
166 -     kPosX = kPos, ///< Drift towards positive X values.
167 +     kNegX = kNeg  ///< Drift towards negative X values.
168 - } DriftDirection_t;
152 + enum class DriftSign {
153 +     Unknown, ///< Drift direction is unknown.
154 +     Positive, ///< Drift towards positive values.
155 +     Negative  ///< Drift towards negative values.
156 + };
157 + std::ostream& operator<<(std::ostream& os, DriftSign);
```

- New struct for accessing drift-axis information:
- Simplifications to ID interface (e.g. `geo::PlaneID`).

```
struct DriftAxis {
    Coordinate coordinate;
    DriftSign sign;
};
```

# LArSoft v10 release candidate

- The LArSoft v10 release candidate was out on Nov. 1, 2023.
- List of PRs and feature branches at right (not active links) and at above link.

***Pay close attention to the definitions of the sorters and the FHiCL-file changes.***

- Some updates likely in order as code has evolved since then.
- I will coordinate with Lynn on how to go about doing these updates.

[https://github.com/LArSoft/larsoft/releases/tag/v10\\_00\\_00rc0](https://github.com/LArSoft/larsoft/releases/tag/v10_00_00rc0)

## Pull requests

- [DeepLearnPhysics/Supera#22](#)
- [DUNE/dunesw#86](#)
- [DUNE/protoduneana#23](#)
- [DUNE/duneana#41](#)
- [DUNE/dunereco#71](#)
- [DUNE/dunedataprep#32](#)
- [DUNE/duneprototypes#35](#)
- [DUNE/dunesim#51](#)
- [DUNE/duneopdet#43](#)
- [DUNE/dunecore#95](#)
- [SBNSoftware/icarusalg#75](#)
- [SBNSoftware/sbncode#393](#)
- [SBNSoftware/sbndcode#393](#)
- [SBNSoftware/icaruscode#642](#)
- [SBNSoftware/sbnobj#98](#)

## Feature branches

- `ubana:feature/knoepfel_geom_separate`
- `ubcore:feature/knoepfel_geom_separate`
- `ubcrt:feature/knoepfel_geom_separate`
- `ubcv:feature/knoepfel_geom_separate`
- `ubevt:feature/knoepfel_geom_separate`
- `ublite:feature/knoepfel_geom_separate`
- `uboonecode:feature/knoepfel_geom_separate`
- `ubraw:feature/knoepfel_geom_separate`
- `ubreco:feature/knoepfel_geom_separate`
- `ubsim:feature/knoepfel_geom_separate`