

OPTICAL DETECTOR GEOMETRY / INFRASTRUCTURE

Ben Jones,
MIT

Two Updates Today:

- 1. Ben Jones, Optical Detector Infrastructure + Geometry
- 2. Christie Chiu, Optical Detector Signal Processing + Hit Finding

Renaming Process

- Brian requested that everything previously called “PMT_____” be called “OpDet_____” to better accommodate non-PMT based optical detectors
- This change was made to LArG4, EventGenerator, Simulation, PhotonPropagation, OpticalDetector, Geometry.
- Sorry for the few leftover loose ends which caused problems / compatibility issues.

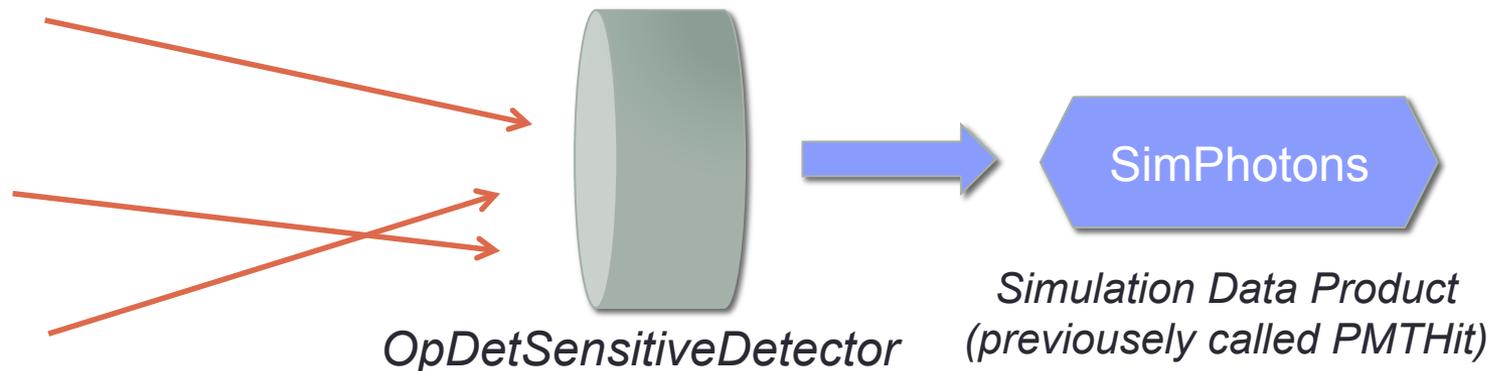
File Name	Size	Line Count	Age	Author	Change Description
MuNuclearSplittingProcessXSecBias.cxx	12.1 kB	1936	9 months	Brian Rebel	changes to make u
MuNuclearSplittingProcessXSecBias.h	2.8 kB	1936	9 months	Brian Rebel	changes to make u
OpBoundaryProcessSimple.cxx	9.1 kB	1936	9 months	Brian Rebel	changes to make u
OpBoundaryProcessSimple.hh	5.7 kB	1936	9 months	Brian Rebel	changes to make u
OpDetLookup.cxx	3.2 kB	2834	5 days	Benjamin Jones	Tie the LArG4 OpDe
OpDetLookup.h	2 kB	2834	5 days	Benjamin Jones	Tie the LArG4 OpDe
OpDetReadoutGeometry.cxx	4.4 kB	2834	5 days	Benjamin Jones	Tie the LArG4 OpDe
OpDetReadoutGeometry.h	1.7 kB	2825	5 days	Benjamin Jones	More variable name
OpDetSensitiveDetector.cxx	2.4 kB	2825	5 days	Benjamin Jones	More variable name
OpDetSensitiveDetector.h	2 kB	2825	5 days	Benjamin Jones	More variable name
OpticalPhysics.cxx	7 kB	1936	9 months	Brian Rebel	changes to make u
OpticalPhysics.hh	3.1 kB	1936	9 months	Brian Rebel	changes to make u
ParticleListAction.cxx	15.9 kB	2825	5 days	Benjamin Jones	More variable name
ParticleListAction.h	3.6 kB	2643	3 months	Brian Rebel	add code to preven

New Base Classes

- Two new base classes for PMT (OpDet) signal processing will be described in Christies talk:
 - *raw::OpDetPulse*
 - *recob::OpHit*

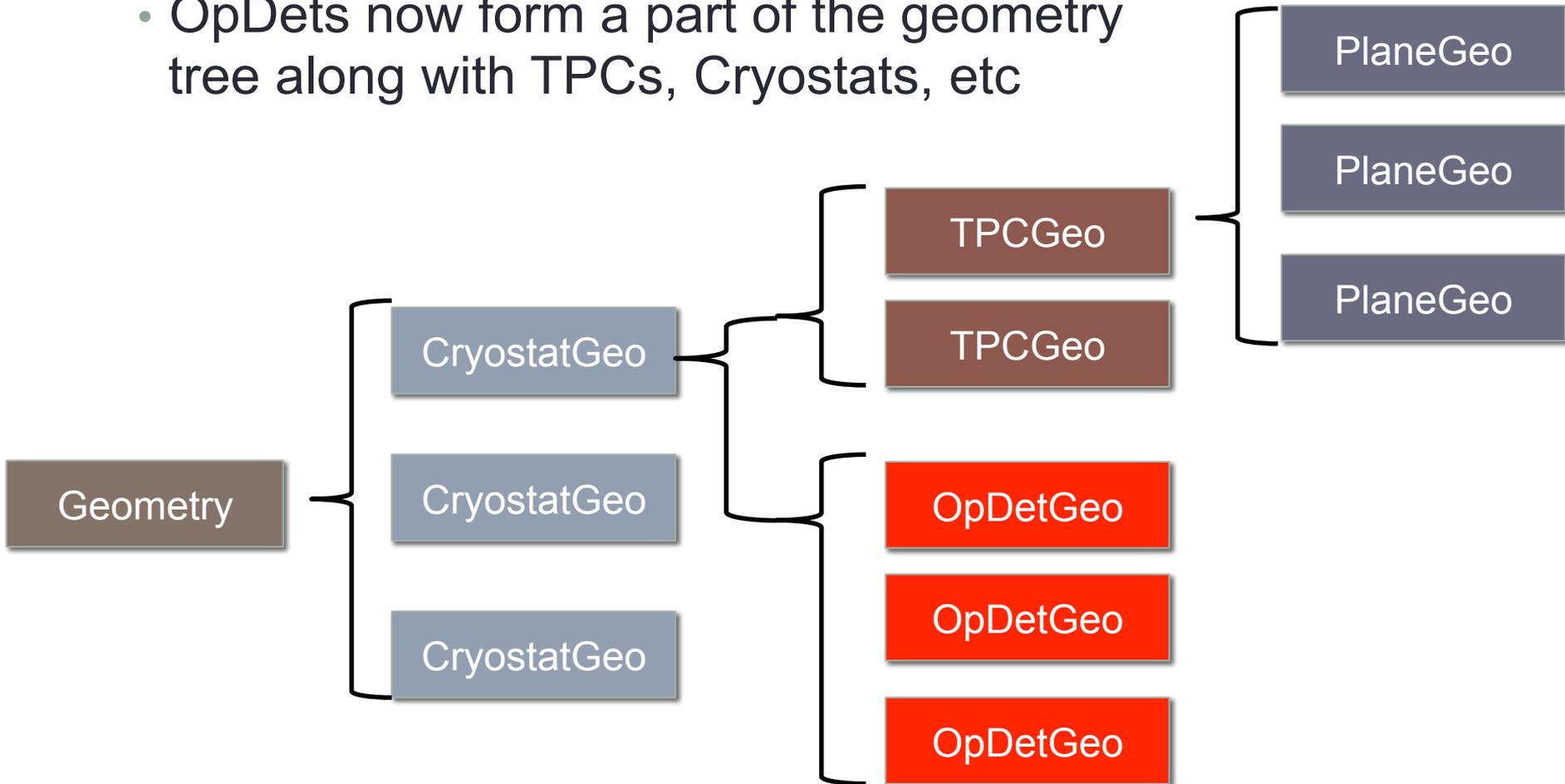
Geometry Changes

- In previous implementation, LArG4 found sensitive elements by specified name in gdml file, and gave each an ID.
- This meant that volumes and ID's were not accessible outside LArG4.
- This now globalized, in the sense that Geometry now controls ID assignments and knows about OpDet positioning
- The mapping between Geometry and LArG4 objects is handled by the LArG4/OpDetLookup object.



geo::OpDetGeo

- OpDets now form a part of the geometry tree along with TPCs, Cryostats, etc



Labeling Convention

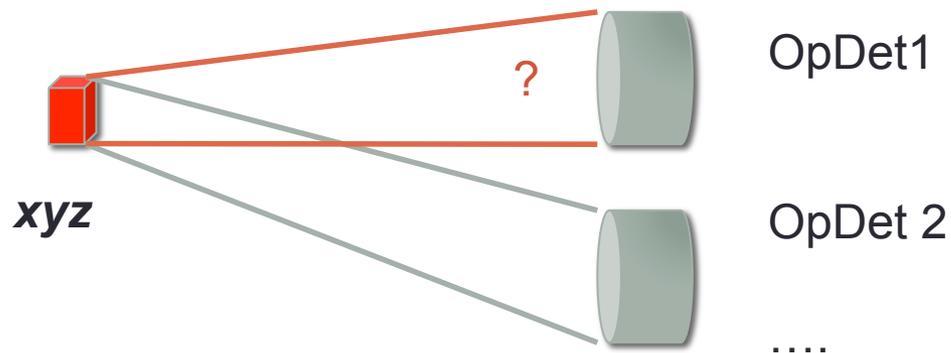
- OpDets exist within a given cryostat
- Similarly to Wires, every OpDet has a unique ID, OpChannel
- Each also has an address (c,o), Cryostat (o to N_cryo) and OpDet (0 to N_opdet)
- Methods added to geometry to facilitate conversion between the two schemes, like for wires and planes.
- Everything stored in the event is indexed by OpChannel only.

root / trunk / Geometry / Geometry.h

```
275 // Convert OpDet, Cryo into OpChannel
276 int      OpDetCryoToOpChannel(unsigned int o, unsigned int c=0);
277
278 // Convert OpChannel into Cryo and OpDet
279 void     OpChannelToCryoOpDet(unsigned int OpChannel, unsigned int& o, unsigned int & c);
280
```

PhotonVisibilityService

- Service added to PhotonPropagation package to facilitate optical reconstruction algorithms
- Main function: Given a point xyz in the detector, how likely is it that 1PE produced there will be seen at each OpDet?
- Two modes of operation :
 - 1) **Simple** – $1/r^2$ and solid angle to opdet surface
 - (no reflections, etc)
 - 2) **Library** – use fastsim library (tbi)
 - (full reflections, scattering, etc accounted for. But requires filled library – not yet available for MicroBooNE)



Simple Mode Being Implemented

```
class PhotonVisibilityService {
public:

    PhotonVisibilityService(fhicl::ParameterSet const& pset, art::ActivityRegistry& reg);
    ~PhotonVisibilityService();

    void reconfigure(fhicl::ParameterSet const& p);

    void SetVisibilityModel(int model) { fVisModel = model; }
    int GetVisibilityModel()          { return fVisModel; }

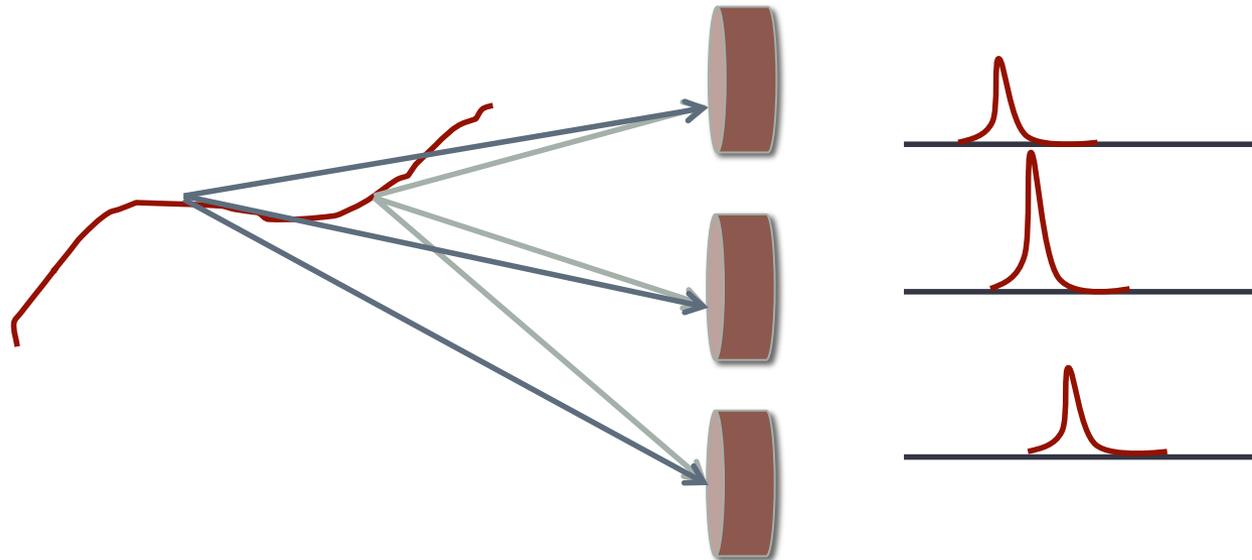
    double GetQuenchingFactor(double dQdx);

    double DistanceToOpDet(          double* xyz, int OpDet );
    double SolidAngleFactor(         double* xyz, int OpDet );
    double GetVisibility(            double* xyz, int OpDet);

    std::vector<double> GetAllVisibilities( double* xyz );
};
```

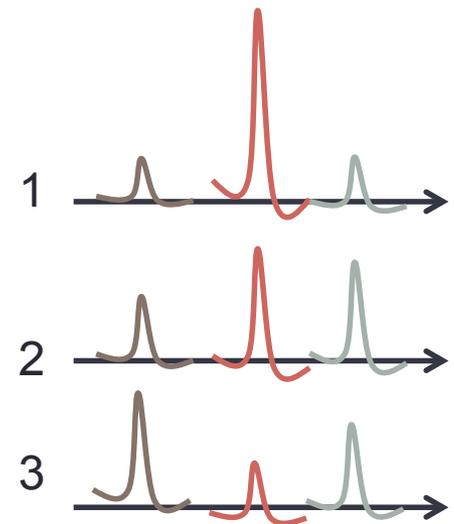
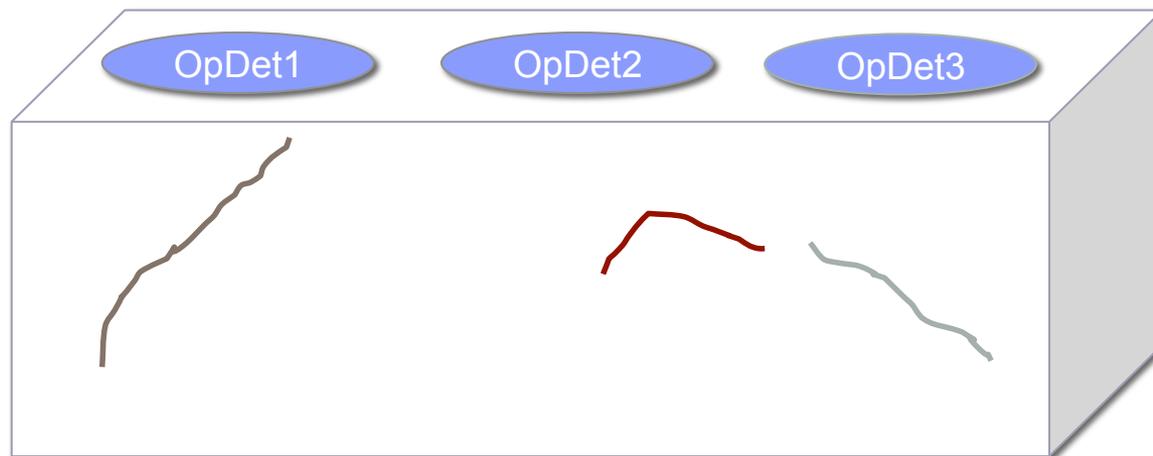
TrackTimeAssoc

- Analyzer in the OpticalDetector package
- Make a quick hypothesis for the light from each track in the event per PMT
- Step along a bezier track in uniform intervals, querying the visibility at each point and multiplying by local $dQdx$.
- Light production can be dropped by quenching function. Visibility and quenching are both controlled by the PhotonVisibilityService



Geometrical T0 Finding

- 1: Make hypotheses of relative amount of light per PMT for each track
- 2: Find subevents by matching large PMT signals in time
- 3: Likelihood fit to match track to light hypothesis, and find T0.



Where are we?

- 1: Make hypotheses of relative amount of light per PMT for each track
 - **Well under way from Ben**
- 2: Find subevents by matching PMT signals in time
 - **Well under way from Christie**
- 3: Likelihood fit to match track to light hypothesis, and find T0.
 - **The next step, but we do not expect big difficulties**
- 4: Figure out what to do with T0
 - **Reconstructed timing objects? Corrected coordinates for off-beam tracks? Michel finding algorithm? Etc...**