Migrating to the refactored *larg4*

David Rivera

University of Pennsylvania

September 6, 2019

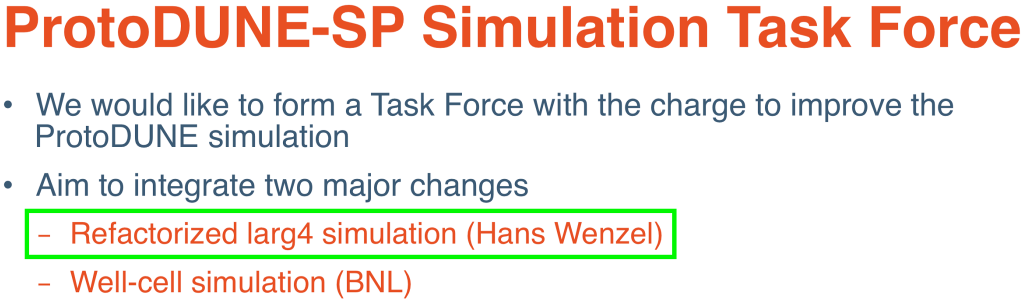
Table of Contents

1. Introduction

2. ProtoDUNE migration

3. Backup

Task Force



ProtoDUNE-SP Simulation Task Force

- We would like to form a Task Force with the charge to improve the ProtoDUNE simulation
- Aim to integrate two major changes
  - Refactorized larg4 simulation (Hans Wenzel)
  - Well-cell simulation (BNL)

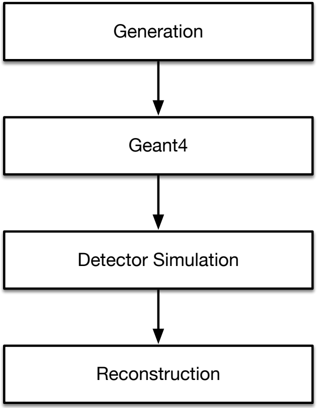
Introduction

- Two options for particle propagation within LArSoft: larsim/LArG4 (legacy) and larg4 (refactored)
- Both interface to Geant4:
  - legacy utilizes a helper class provided by nug4, namely nug4/G4Base (via the g4b::G4Helper)
  - refactored utilizes artg4tk

Reference materials:

- nutools/G4Base
- larg4 Wiki
- artg4tk Wiki

Documentation and demo: Migrating to the refactored larg4



LArSoft Simulation chain

Conventions

- larsim/LArG4 will be referred to as Legacy
- the refactored larg4 will be referred to as larg4
- ProtoDUNE Single-Phase will be referred to as PDSP
- Geant4 and G4 will be used interchangeably

Refactored LArG4

Standard – **larsim/LArG4** AKA **Legacy**

- depends on nug4
- ConfigurablePhysicsList.h
- Optical simulation in Legacy was taken out of Geant and adapted from the Peter Gumplinger's original G4 implementations
  - **TheScintillationProcess** → SetScintillationYield()
  - there can be only one scintillating material in the optical simulation (LAr)

Refactored – **LArG4**

- depends on artg4tk (artg4 tool kit)
- Access to reference physics lists + extensions
- Updated OpticalPhysics in G4
  - scintillation properties are attached to the materials
  - can have any number of scintillating materials in the detector (e.g. LAr and plastic scintillator)

See Hans Wenzel's presentation from the DUNE collaboration meeting for a more comprehensive list of features and improvements of the refactored larg4 over Legacy: slides

Goals

- Customization of the physics list tailored to the interest of the physics under investigation
  - Low Energy Physics:
    - Solar Neutrinos
    - Neutron capture
    - Shielding

Table of Contents

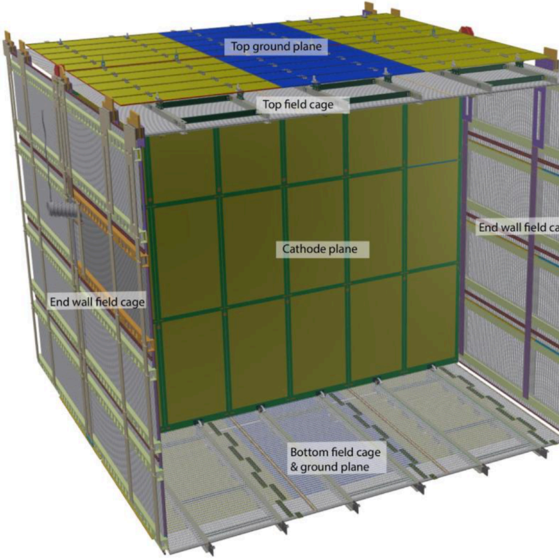
1 Introduction

2 ProtoDUNE migration

3 Backup

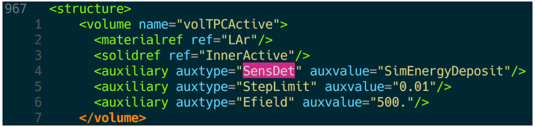
ProtoDUNE Single-Phase

- There is general interest in migrating from legacy to the refactored simulation chain in PDSP
- This is driven mainly by the desire to customize physics lists
- Also, would like to have the ability to do a more *natural* optical simulation with multiple scintillating materials
  - Optical physics within G4 have advanced over the last decade



First Attempt

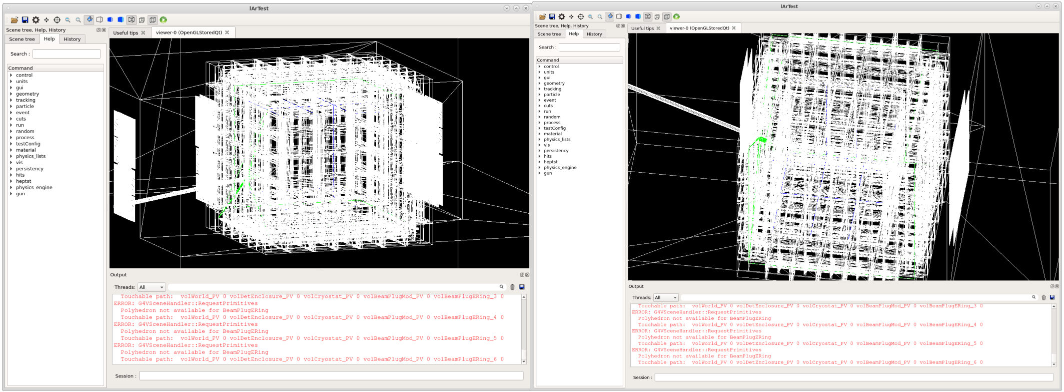
- Hans provided an example refactorization of the 3x1x1 dual-phase detector
  - see Larsoft Feature #22466
- Declared the Liquid Argon volumes as charge sensitive detectors
  - **protodune_v5_refactored.gdml**
  - **protodune_v5_refactored_nowires.gdml**
- Neglected the optical aspect of the simulation, for simplicity
- Redefined the protoDUNE services in the same spirit as the example provided by Hans
- Created corresponding G4→Reconstruction fhicl files
- Also a modified version of the protoDUNE event display fhcil
  - **protoDUNE_refactored_g4.fcl**
  - **protoDUNE_refactored_detsim.fcl**
  - **protoDUNE_refactored_reco.fcl**
  - **evd_refactored_protoDUNE.fcl**



```
<structure>
  <volume name="volTPCActive">
   <materialref ref="LAr"/>
   <solidref ref="InnerActive"/>
   <auxiliary auxtype="SensDet" auxvalue="SimEnergyDeposit"/>
   <auxiliary auxtype="StepLimit" auxvalue="0.01"/>
   <auxiliary auxtype="Efield" auxvalue="500."/>
  </volume>
```

Inner Active TPC volume

- color refs set in the gdml can be visualized in g4

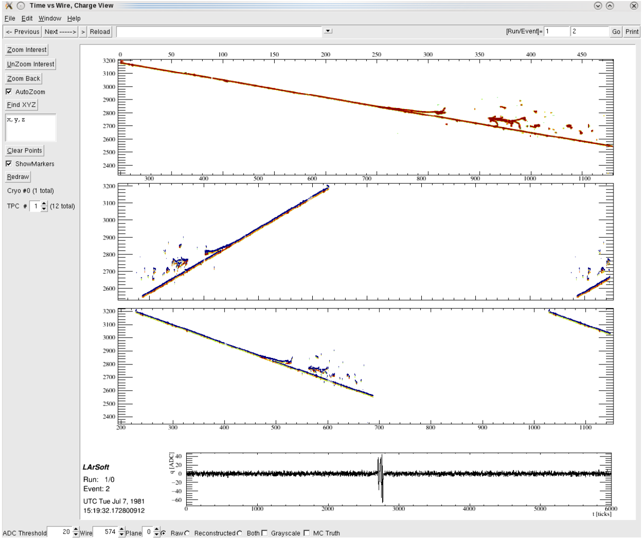


```
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_3 0
ERROR: G4VSceneHandler::RequestPrimitives
  Polyhedron not available for BeamPlugERing
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_4 0
ERROR: G4VSceneHandler::RequestPrimitives
  Polyhedron not available for BeamPlugERing
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_5 0
ERROR: G4VSceneHandler::RequestPrimitives
  Polyhedron not available for BeamPlugERing
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_6 0
```

```
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_3 0
ERROR: G4VSceneHandler::RequestPrimitives
  Polyhedron not available for BeamPlugERing
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_4 0
ERROR: G4VSceneHandler::RequestPrimitives
  Polyhedron not available for BeamPlugERing
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_5 0
ERROR: G4VSceneHandler::RequestPrimitives
  Polyhedron not available for BeamPlugERing
  Touchable path:  volWorld_PV 0 volDetEnclosure_PV 0 volCryostat_PV 0 volBeamPlugMod_PV 0 volBeamPlugERing_6 0
```

G4 visualization of the protoDUNE v5 geometry generated by H. Wenzel

Example Event : 6GeV μ^-



[Run/Event]= 1 2

☑ AutoZoom

☑ ShowMarkers

TPC # 1 (12 total)

LArSoft

Run: 1/0

Event: 2

UTC Tue Jul 7, 1981

15:19:32.172800912

ADC Threshold 20 Wire 574 Plane 0 ● Raw ○ Reconstructed ○ Both □ Grayscale □ MC Truth

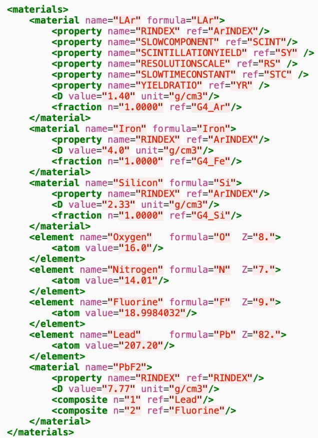
6 GeV muon simulated in the refactored framework

- See the demo

Example within Larsoft

- The ProtoDUNE migration example can be found on the Redmine Wiki for larg4 : ProtoDUNE Example wiki
- Initial commit of the working PDSP refactored files pushed and available as of larsoft v08_25_00
- **dunetpc/fcl/protodune/g4/protoDUNE_refactored_g4.fcl**
- **dunetpc/fcl/protodune/detsim/protoDUNE_refactored_detsim.fcl**
- **dunetpc/fcl/protodune/reco/protoDUNE_refactored_reco.fcl**
- **dunetpc/fcl/evd/evd_refactored_protoDUNE.fcl**
- **dunetpc/dune/Utilities/services_refactored_pdune.fcl**

To Do



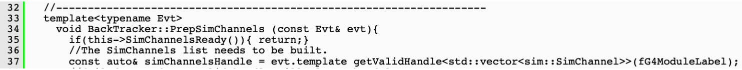
```xml
<materials>
    <material name="LAr" formula="LAr">
        <property name="RINDEX" ref="ArINDEX"/>
        <property name="SLOWCOMPONENT" ref="SCINT"/>
        <property name="SCINTILLATIONYIELD" ref="SY" />
        <property name="RESOLUTIONSCALE" ref="RS" />
        <property name="SLOWTIMECONSTANT" ref="STC" />
        <property name="YIELDRATIO" ref="YR" />
        <D value="1.40" unit="g/cm3"/>
        <fraction n="1.0000" ref="G4_Ar"/>
    </material>
    <material name="Iron" formula="Iron">
        <property name="RINDEX" ref="ArINDEX"/>
        <D value="4.0" unit="g/cm3"/>
        <fraction n="1.0000" ref="G4_Fe"/>
    </material>
    <material name="Silicon" formula="Si">
        <property name="RINDEX" ref="ArINDEX"/>
        <D value="2.33" unit="g/cm3"/>
        <fraction n="1.0000" ref="G4_Si"/>
    </material>
    <element name="Oxygen"   formula="O"  Z="8.">
        <atom value="16.0"/>
    </element>
    <element name="Nitrogen" formula="N"  Z="7.">
        <atom value="14.01"/>
    </element>
    <element name="Fluorine" formula="F"  Z="9.">
        <atom value="18.9984032"/>
    </element>
    <element name="Lead"     formula="Pb" Z="82.">
        <atom value="207.20"/>
    </element>
    <material name="PbF2">
        <property name="RINDEX" ref="RINDEX"/>
        <D value="7.77" unit="g/cm3"/>
        <composite n="1" ref="Lead"/>
        <composite n="2" ref="Fluorine"/>
    </material>
</materials>
```

Optical material properties

- Define the optical properties of the relevant materials in the geometry file
- Consider ways to provide physical properties as configuration parameters for the G4 stage
    - E.g. for now the E-field is hard-coded in the geometry file
- Purge refactored services
- Continue validation process
- Compare resource usage between the new and the legacy frameworks

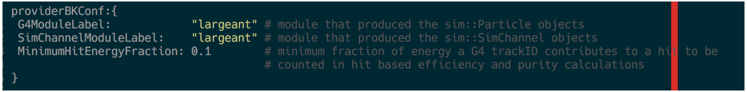
Known and Addressed Issues

1. Incompatibility with Backtracking
   - The Backtracking service expects simChannel info from from the largeant module. However, the simChannel objects are filled outside of larg4 in the refactored framework
   - TrackIDs are assigned uniquely, this is incompatible with BackTracking as well
2. Out-of-date geometry
   - v5 is not the latest version of the protoDUNE geometry. As of 8/13/19 we have a version 7. However the changes in the lates version mainly address Aux. detectors which are not yet accounted for in the refactored ProtoDUNE example.



```
//----------------------------------------------------------------------
template<typename Evt>
  void BackTracker::PrepSimChannels (const Evt& evt){
    if(this->SimChannelsReady()){ return;}
    //The SimChannels list needs to be built.
    const auto& simChannelsHandle = evt.template getValidHandle<std::vector<sim::SimChannel>>(fG4ModuleLabel);
```

backtracker.tcc



```
providerBKConf:{
 G4ModuleLabel:             "largeant" # module that produced the sim::Particle objects
 SimChannelModuleLabel:     "largeant" # module that produced the sim::SimChannel objects
 MinimumHitEnergyFraction: 0.1         # minimum fraction of energy a G4 trackID contributes to a hit to be
                                       # counted in hit based efficiency and purity calculations
}
```

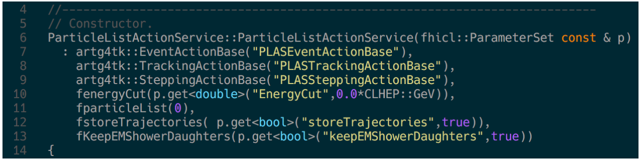
backtrackerservice.fcl

Known and Addressed Issues continued

Solutions:

1. An additional *SimChannelModuleLabel* parameter has been added to the Backtracker class
   - If not provided, SimChannelModuleLabel defaults to the G4ModuleLabel
   - Currently in a feature branch of larsim (feature/drivera_larg4_compatibilty)
   - Change approved by J. Stock, the maintainer of the BackTrackers
2. larg4 TrackID offset was not being reset after each event (monotonically increasing, unlike Legacy) i
   - added a reset to the endOfEventAction for the ParticleListActionService
   - Changed approved by Hans Wenzel
3. The v7 version of the PDSP geometry was refactored
   - The perl script utilized to generate the standard version of the v7 geometry was copied and adapted to function for the refactored geometry
   - New geometry currently lives in a feature branch of dunetpc (feature/drivera_larg4_compatibility)

Work in Progress



```
//----------------------------------------------------------------------------
// Constructor.
ParticleListActionService::ParticleListActionService(fhicl::ParameterSet const & p)
  : artg4tk::EventActionBase("PLASEventActionBase"),
    artg4tk::TrackingActionBase("PLASTrackingActionBase"),
    artg4tk::SteppingActionBase("PLASSteppingActionBase"),
    fenergyCut(p.get<double>("EnergyCut",0.0*CLHEP::GeV)),
    fparticleList(0),
    fstoreTrajectories( p.get<bool>("storeTrajectories",true)),
    fKeepEMShowerDaughters(p.get<bool>("keepEMShowerDaughters",true))
{
```

larg4::ParticleListActionService constructor

- Attempted to simulate cosmics in the refactored framework, keeping as much physics as possible
- Using a high precision physics list
- Memory consumption too large!
- Debugging session with Paul Russo from the scientific comptuting division
- Utilized memory profilers to track down the main agressors
- Three main factors: the SimDriftedElectronClusters, the G4 Step limit, and ROOT buffering

Currently, the ProtoDUNE migration is maintained in feature branches for larsim/larg4/dunetpc:
feature/drivera_backtracker_larg4_compatibility

- Current changes are based on larsoft v08_27_01
- will merge latest updates and commit to develop soon

Impressions

- Low barrier of entry concerning the geometry
  - Modifying the GDML file for LArTPC experiments only *requires* specifying a G4 Step Limit and declaring the Active TPC volumes as a **SensitiveDetector**
- Straight-forward to define a separate set of fhicl files that takes advantage of the refactored larg4
  - G4 stage: Only need to become familiar with the Refactored Physics Constructor (artg4tk)
  - Post-G4 stages: Only need to override the SimChannelLabel to match the one for the elecDrift Module (or drift module of your choice)
- At a glance, the physics make a lot more sense
  - Reference physics lists are widely used and are supported by the Geant collaboration

Table of Contents

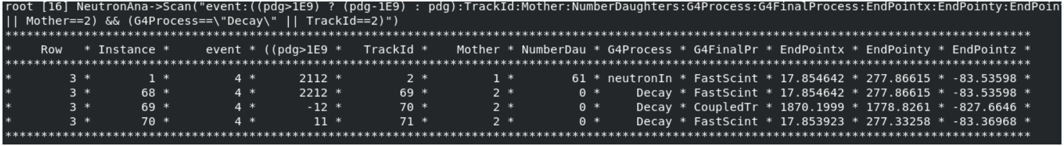
1. Introduction

2. ProtoDUNE migration

3. Backup

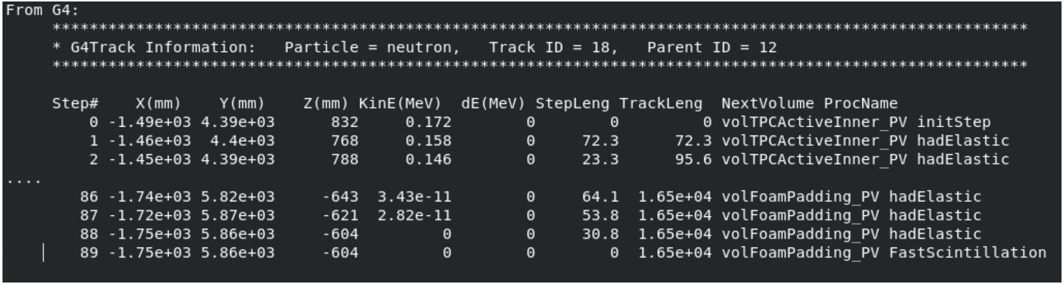
Neutron Study in Legacy

- Produced various samples of 10 MeV neutrons at the center of TPC1 (larsoft numbering, APA3-active)
- **Issue 1:** simb::MCParticle->EndProcess() for secondary neutrons often returns *FastScintillation*
- **Issue 2:** Some neutrons ending with FastScintillation processes come to rest in the ProtoDUNEFoam
- **Issue 3:** At rest neutrons subsequently decay. . . (n→ p + e^- + $\overline{\nu}_e$)
  - Neutron EndProcess is still marked as FastScintillation
  - simb::MCParticle→Process() for proton, e^-, and $\overline{\nu}_e$ returns *Decay*



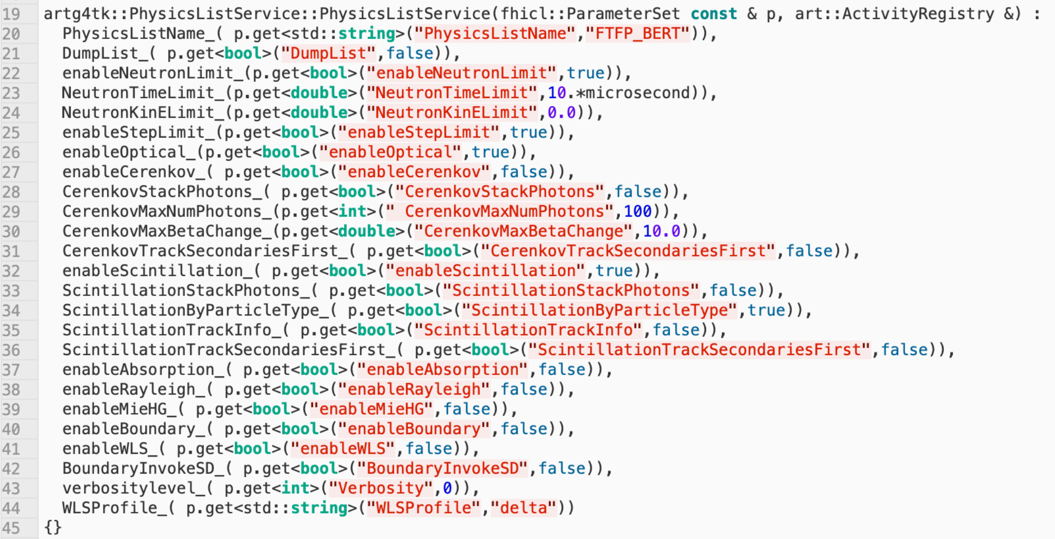
```
root [16] NeutronAna->Scan("event:((pdg>1E9) ? (pdg-1E9) : pdg):TrackId:Mother:NumberDaughters:G4Process:G4FinalProcess:EndPointx:EndPointy:EndPoin
|| Mother==2) && (G4Process==\"Decay\" || TrackId==2)")
************************************************************************************************************************************************
*    Row   * Instance *     event * ((pdg>1E9 *   TrackId *    Mother * NumberDau * G4Process * G4FinalPr * EndPointx * EndPointy * EndPointz *
************************************************************************************************************************************************
*        3 *        1 *         4 *      2112 *         2 *         1 *        61 * neutronIn * FastScint * 17.854642 * 277.86615 * -83.53598 *
*        3 *       68 *         4 *      2212 *        69 *         2 *         0 *     Decay * FastScint * 17.854642 * 277.86615 * -83.53598 *
*        3 *       69 *         4 *       -12 *        70 *         2 *         0 *     Decay * CoupledTr * 1870.1999 * 1778.8261 * -827.6646 *
*        3 *       70 *         4 *        11 *        71 *         2 *         0 *     Decay * FastScint * 17.853923 * 277.33258 * -83.36968 *
************************************************************************************************************************************************
```

G4 Printout



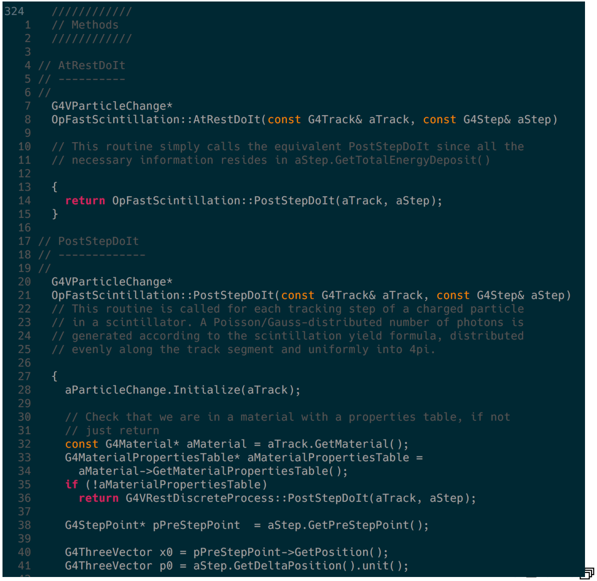
```
From G4:
     *********************************************************************************************************
     * G4Track Information:   Particle = neutron,   Track ID = 18,   Parent ID = 12
     *********************************************************************************************************

     Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
         0 -1.49e+03 4.39e+03      832     0.172        0        0         0 volTPCActiveInner_PV initStep
         1 -1.46e+03  4.4e+03      768     0.158        0     72.3      72.3 volTPCActiveInner_PV hadElastic
         2 -1.45e+03 4.39e+03      788     0.146        0     23.3      95.6 volTPCActiveInner_PV hadElastic
....
        86 -1.74e+03 5.82e+03     -643  3.43e-11        0     64.1  1.65e+04 volFoamPadding_PV hadElastic
        87 -1.72e+03 5.87e+03     -621  2.82e-11        0     53.8  1.65e+04 volFoamPadding_PV hadElastic
        88 -1.75e+03 5.86e+03     -604         0        0     30.8  1.65e+04 volFoamPadding_PV hadElastic
        89 -1.75e+03 5.86e+03     -604         0        0        0  1.65e+04 volFoamPadding_PV FastScintillation
```

Refactored Physics Constructor



```
artg4tk::PhysicsListService::PhysicsListService(fhicl::ParameterSet const & p, art::ActivityRegistry &) :
  PhysicsListName_( p.get<std::string>("PhysicsListName","FTFP_BERT")),
  DumpList_( p.get<bool>("DumpList",false)),
  enableNeutronLimit_(p.get<bool>("enableNeutronLimit",true)),
  NeutronTimeLimit_(p.get<double>("NeutronTimeLimit",10.*microsecond)),
  NeutronKinELimit_(p.get<double>("NeutronKinELimit",0.0)),
  enableStepLimit_(p.get<bool>("enableStepLimit",true)),
  enableOptical_(p.get<bool>("enableOptical",true)),
  enableCerenkov_( p.get<bool>("enableCerenkov",false)),
  CerenkovStackPhotons_( p.get<bool>("CerenkovStackPhotons",false)),
  CerenkovMaxNumPhotons_(p.get<int>(" CerenkovMaxNumPhotons",100)),
  CerenkovMaxBetaChange_(p.get<double>("CerenkovMaxBetaChange",10.0)),
  CerenkovTrackSecondariesFirst_( p.get<bool>("CerenkovTrackSecondariesFirst",false)),
  enableScintillation_( p.get<bool>("enableScintillation",true)),
  ScintillationStackPhotons_( p.get<bool>("ScintillationStackPhotons",false)),
  ScintillationByParticleType_( p.get<bool>("ScintillationByParticleType",true)),
  ScintillationTrackInfo_( p.get<bool>("ScintillationTrackInfo",false)),
  ScintillationTrackSecondariesFirst_( p.get<bool>("ScintillationTrackSecondariesFirst",false)),
  enableAbsorption_( p.get<bool>("enableAbsorption",false)),
  enableRayleigh_( p.get<bool>("enableRayleigh",false)),
  enableMieHG_( p.get<bool>("enableMieHG",false)),
  enableBoundary_( p.get<bool>("enableBoundary",false)),
  enableWLS_( p.get<bool>("enableWLS",false)),
  BoundaryInvokeSD_( p.get<bool>("BoundaryInvokeSD",false)),
  verbositylevel_( p.get<int>("Verbosity",0)),
  WLSProfile_( p.get<std::string>("WLSProfile","delta"))
{}
```

OpFastScintillation



```
 ////////////
 // Methods
 ////////////

// AtRestDoIt
// ----------
//
 G4VParticleChange*
 OpFastScintillation::AtRestDoIt(const G4Track& aTrack, const G4Step& aStep)

 // This routine simply calls the equivalent PostStepDoIt since all the
 // necessary information resides in aStep.GetTotalEnergyDeposit()

 {
   return OpFastScintillation::PostStepDoIt(aTrack, aStep);
 }

// PostStepDoIt
// -------------
//
 G4VParticleChange*
 OpFastScintillation::PostStepDoIt(const G4Track& aTrack, const G4Step& aStep)
 // This routine is called for each tracking step of a charged particle
 // in a scintillator. A Poisson/Gauss-distributed number of photons is
 // generated according to the scintillation yield formula, distributed
 // evenly along the track segment and uniformly into 4pi.

 {
   aParticleChange.Initialize(aTrack);

   // Check that we are in a material with a properties table, if not
   // just return
   const G4Material* aMaterial = aTrack.GetMaterial();
   G4MaterialPropertiesTable* aMaterialPropertiesTable =
    aMaterial->GetMaterialPropertiesTable();
   if (!aMaterialPropertiesTable)
    return G4VRestDiscreteProcess::PostStepDoIt(aTrack, aStep);

   G4StepPoint* pPreStepPoint  = aStep.GetPreStepPoint();

   G4ThreeVector x0 = pPreStepPoint->GetPosition();
   G4ThreeVector p0 = aStep.GetDeltaPosition().unit();
```