LArG4 refactoring

Hans Wenzel

23rd Oct 2018

Outline

- Reminder
  - Requirements
  - Changes to Geant4
  - Artg4tk
- Finishing up:
  - Providing examples and documentation
  - auxdets

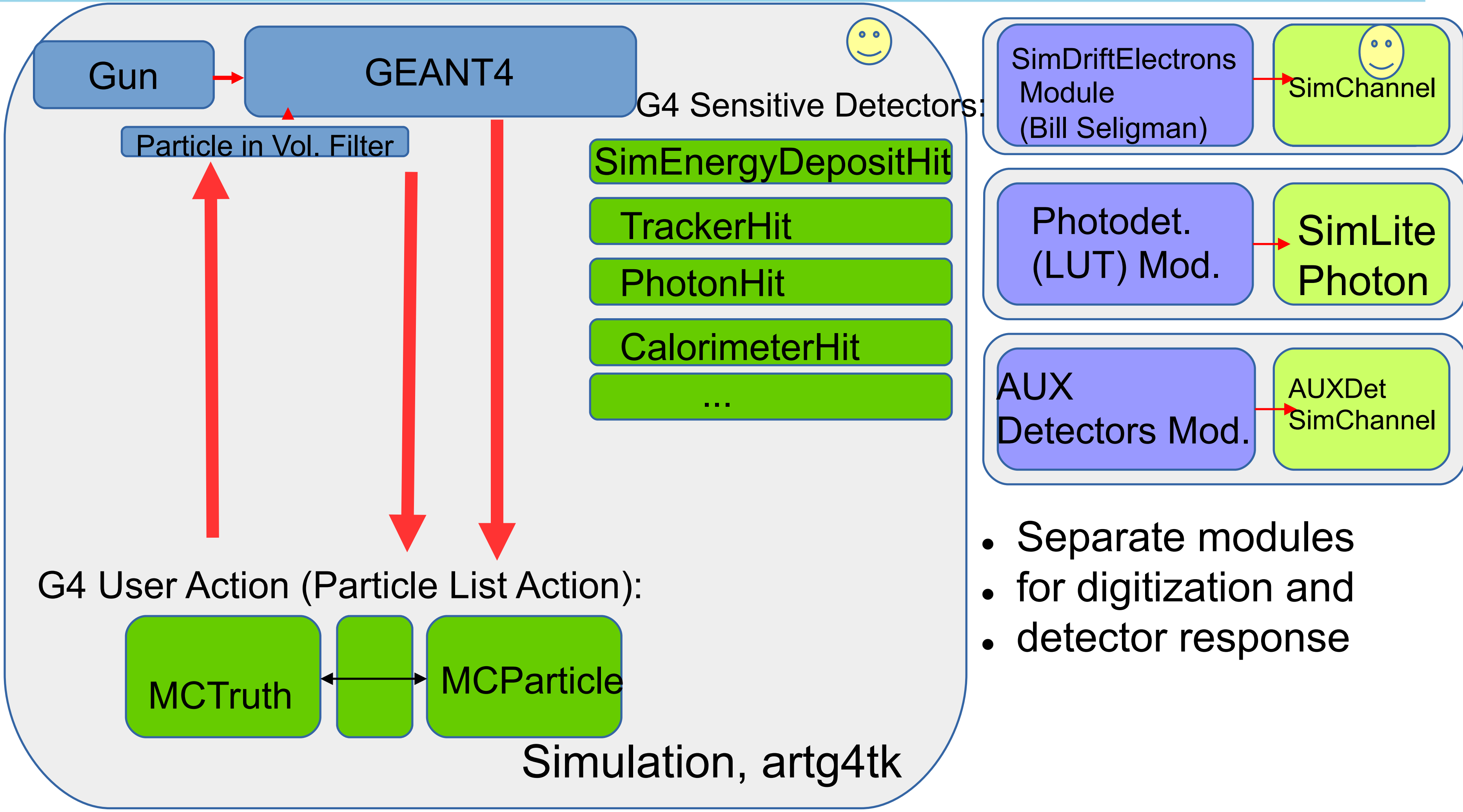
Requirements

- Separate Simulation from digitization and detector response
- Simulation:
  - completely depend on tools provided by Geant4 and use the provided interfaces (or work with the Geant4 collaboration to make them available), make sure that they are efficient with regards to CPU and memory (profiling).
  - Have access to all physics lists and physics constructors and processes provided by GEANT4
  - Be able to describe complete detector systems → liquid Argon TPC is just one possible sensitive Detector.
  - Two modes: full optical simulation (making use of the Geant4 capabilities), use of look up tables for optical response
- Separate digitization and detector response:
  - Ability to plug in models handling the correlations between ionization and Scintillation (e.g. Nest)
  - Ability to switch drift model (detector response).
-

Changes to Geant4

- Developed the Geant4 Physics List factory (Robert Hatcher)
- Made sure the optical physics constructor allows us to be configured in a way relevant to liquid Argon TPC simulation. E.g. get access to optical photons without putting them on the stack. (P. Gumplinger, me)
- Created validation suite to check various processes relevant to neutrino physics e.g.:
    - Checked and fixed various cross sections (especially kaons) relevant to e.g. neutrino experiments $\rightarrow$ add relevant data to DoSSiER (Geant4 validation database).
    - Fixed bug in Bertini cascade leading to stopped kaons being dropped.
    - Fixed bugs in Cerenkov/scintillation physics (no reset when entering material without optical properties.)
- Included liquid Argon application lArTest in regular Geant4 profiling:
    - Optimized access to material properties (10% CPU improvement) (Soon, me)

Refactored



- Separate modules
- for digitization and
- detector response

Artg4tk

- Started with artg4 that Adam created. This provides interface between Geant4 and art. (and depends only on art and Geant4)
- Artg4tk was extended to include
  - Use of physics list factory.
  - Use gdml to define geometry, material properties etc.
    - Extend gdml to assign sensitive detectors to logical volumes, visualization properties, fields…
  - library of typical sensitive detectors: e.g. Tracker, Photon Detector, Calorimeter, dual read out calorimeter, SimEnergyDepositSD… → assigned to LV in gdml, each SD knows how to inject hits into art event stream and takes care of uniquely naming the hit collections.
  - Vary model parameters
- Very flexible framework which allows to create different detector configurations quickly → no knowledge of Geant4 internals necessary. Good for detector R&D.

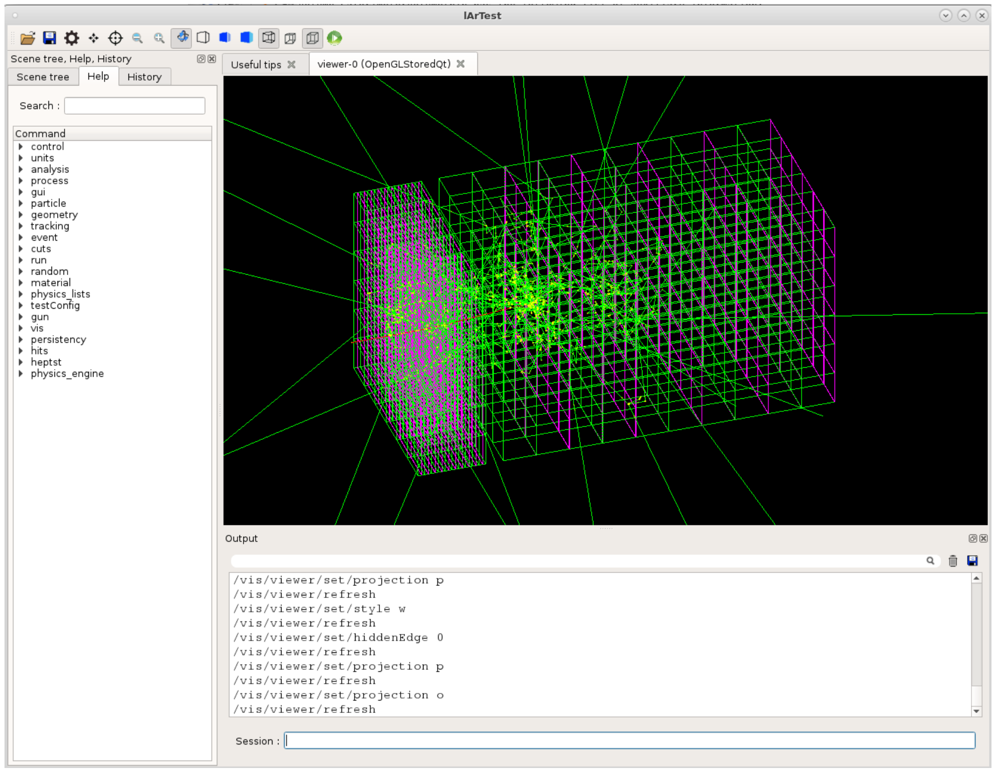
Status

- larg4 repository contains the bits that depend on the various components of larsoft e.g. SimEnergyDepositSD, AuxDetSD, ParticleListAction (core of artg4tk just depends on art and Geant4)
- Created PrimaryGeneratorService that takes MCTruth as input and feeds to Geant4, tested using the SingleGen module. (in addition Geant4 particle Gun, GenParticle, Hepevt are available)
- Special sensitive detector that produces the SimEnergyDeposit (currently uses Geant4 to calculate Sc. Photons → add tools that add correlation)
- SimDriftElectrons_module.cc (provided by Bill Seligman, Wes Ketchum).
- Ported ParticleListAction (Event, Stepping, Tracking Action) to artg4tk as a service, removed dependencies from nutools (now just just need to push the result into the event)
- Provide analysis module demonstrating access to the produced data products.

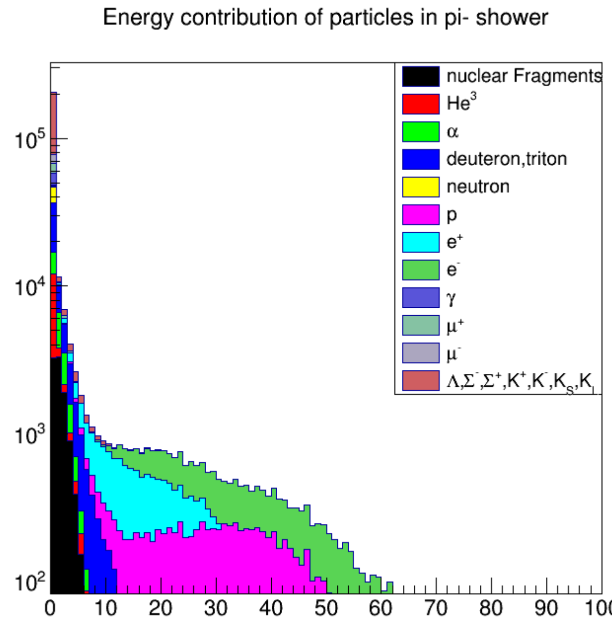
Working in the following git feature branches:

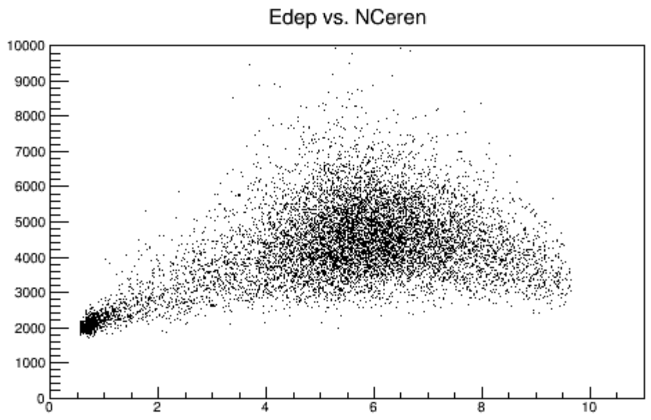
artg4tk: feature/wenzel_artg4tk_cleanup
lardataobj: feature/wenzel_larg4refac_phase2_1
larg4: feature/wenzel_larg4refac_phase2_1

Example dual readout calorimeter (eic):



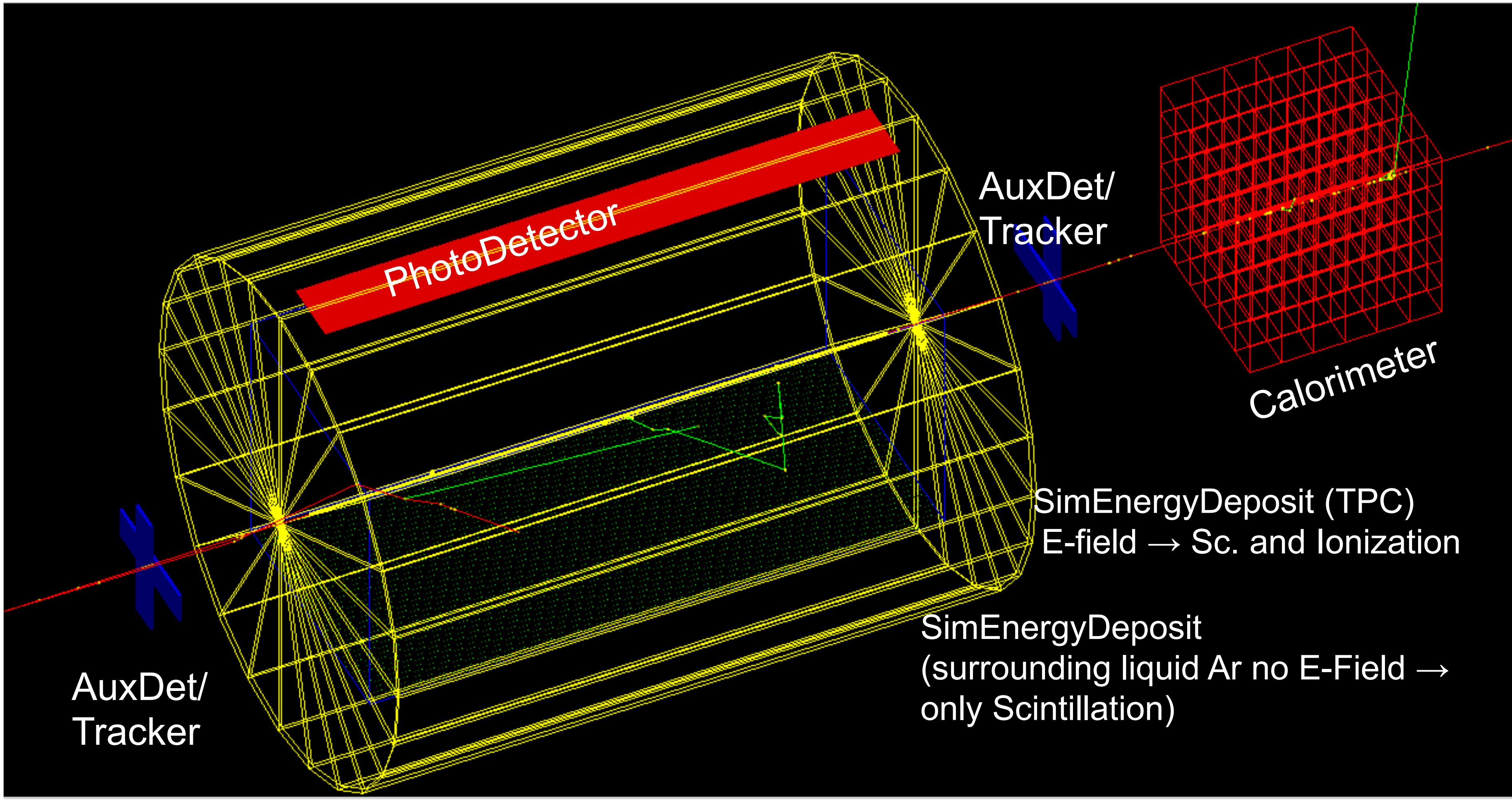
```
/vis/viewer/set/projection p
/vis/viewer/refresh
/vis/viewer/set/style w
/vis/viewer/refresh
/vis/viewer/set/hiddenEdge 0
/vis/viewer/refresh
/vis/viewer/set/projection p
/vis/viewer/refresh
/vis/viewer/set/projection o
/vis/viewer/refresh
```





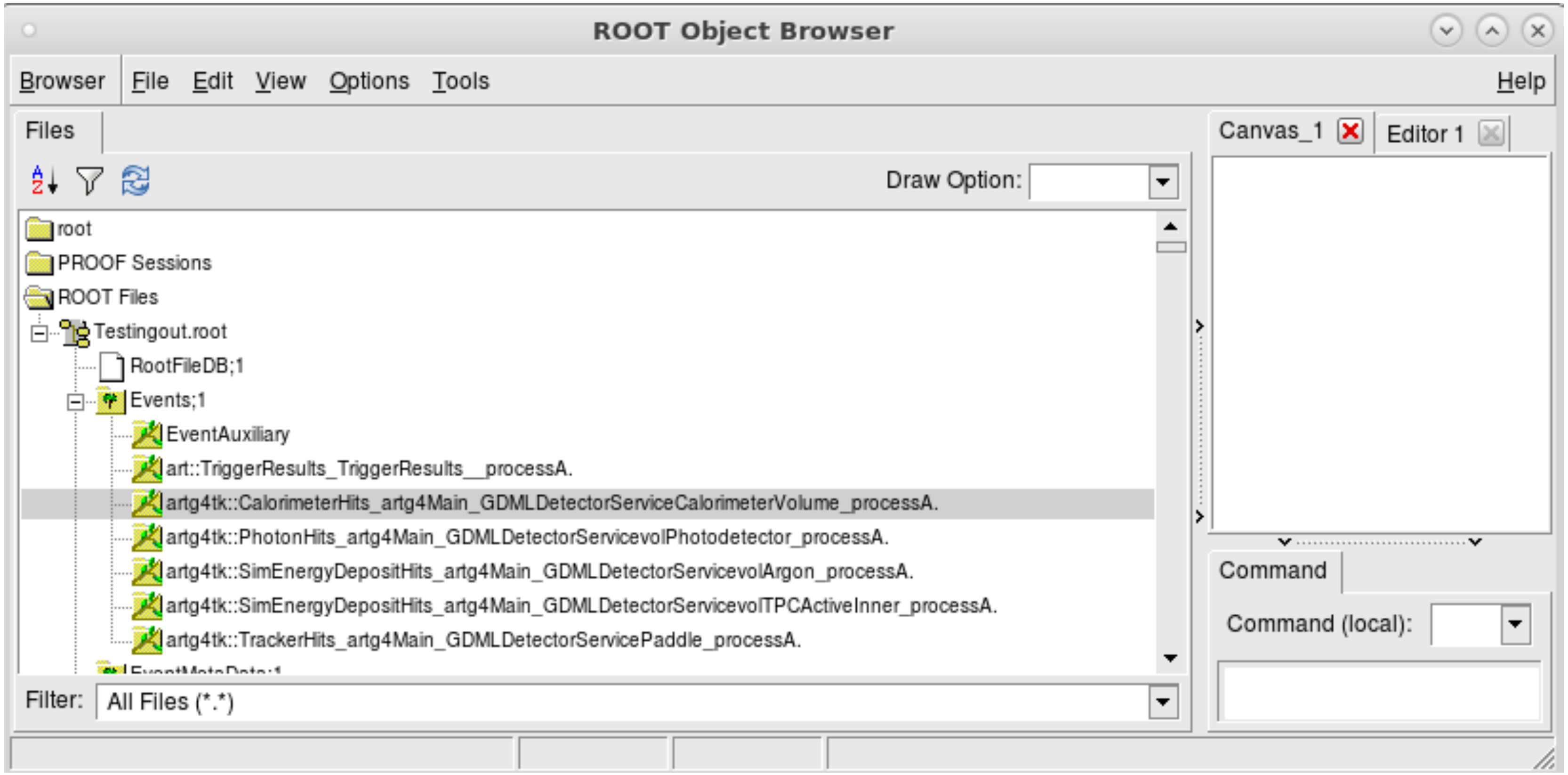
- gdml file
- fcl file
- analysis module
- Root macros to analyze the result
- Documentation (redmine twiki)

Example:
https://cdcvs.fnal.gov/redmine/projects/artg4tk/repository/revisions/develop/entry/gdml/lArDet.gdml



Running artg4tk and Resulting Hit Collections in the EDM

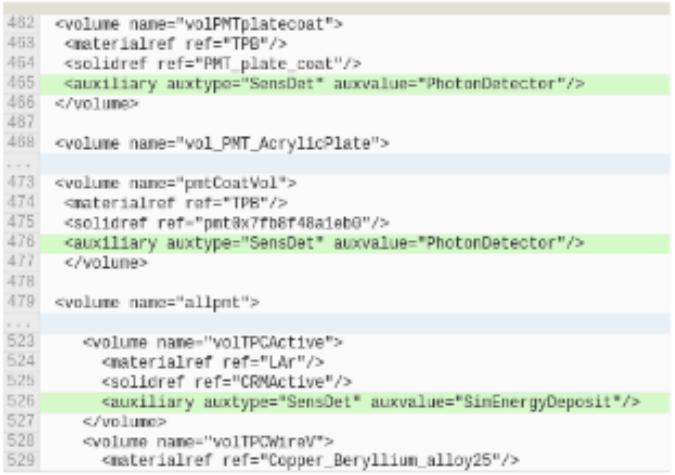
art -c lArDet.fcl



```
root
PROOF Sessions
ROOT Files
  Testingout.root
    RootFileDB;1
    Events;1
      EventAuxiliary
      art::TriggerResults_TriggerResults__processA.
      artg4tk::CalorimeterHits_artg4Main_GDMLDetectorServiceCalorimeterVolume_processA.
      artg4tk::PhotonHits_artg4Main_GDMLDetectorServicevolPhotodetector_processA.
      artg4tk::SimEnergyDepositHits_artg4Main_GDMLDetectorServicevolArgon_processA.
      artg4tk::SimEnergyDepositHits_artg4Main_GDMLDetectorServicevolTPCActiveInner_processA.
      artg4tk::TrackerHits_artg4Main_GDMLDetectorServicePaddle_processA.
```

Testing the refactored larsoft (Jose Alfonso Soto Oton)

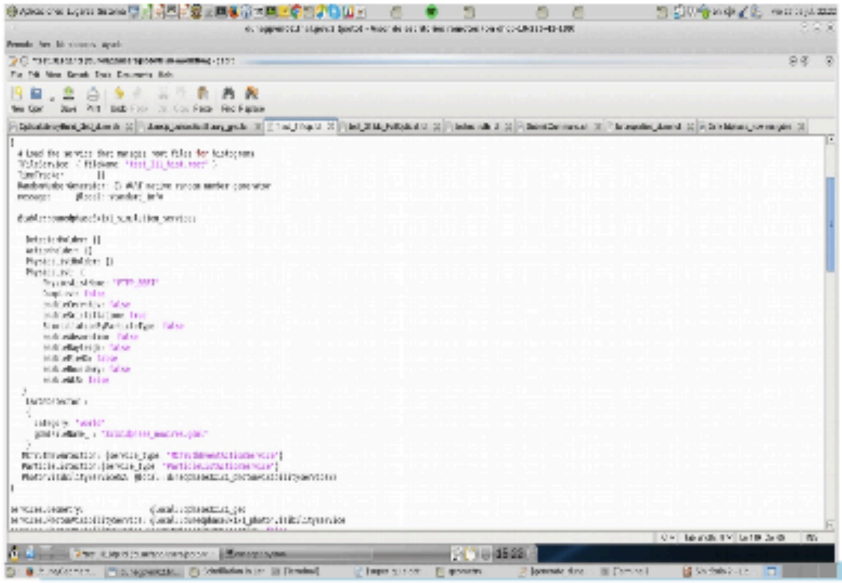
- The new refactored larsoft has been tested using the 3x1x1 dual phase TPC geometry.
- Only a few changes are needed to run it:
- In the GDML:



```
<volume name="volPMTplatecoat">
 <materialref ref="TPB"/>
 <solidref ref="PMT_plate_coat"/>
 <auxiliary auxtype="SensDet" auxvalue="PhotonDetector"/>
</volume>

<volume name="vol_PMT_AcrylicPlate">
...
<volume name="pmtCoatVol">
 <materialref ref="TPB"/>
 <solidref ref="pmt0x7fb8f48a1eb0"/>
 <auxiliary auxtype="SensDet" auxvalue="PhotonDetector"/>
 </volume>

<volume name="allpmt">
...
   <volume name="volTPCActive">
     <materialref ref="LAr"/>
     <solidref ref="CRMActive"/>
     <auxiliary auxtype="SensDet" auxvalue="SimEnergyDeposit"/>
   </volume>
   <volume name="volTPCWireV">
     <materialref ref="Copper_Beryllium_alloy25"/>
```

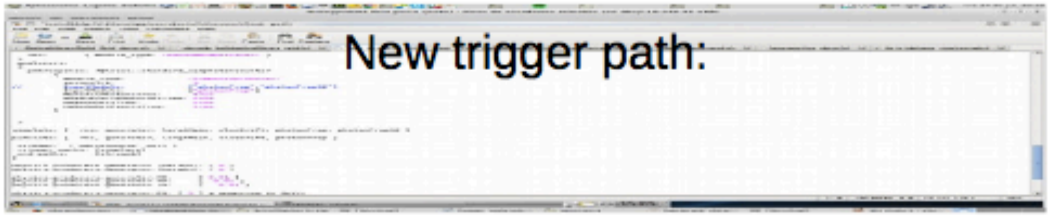
- In the FHICL:

List of new services:



[illegible]

New trigger path:

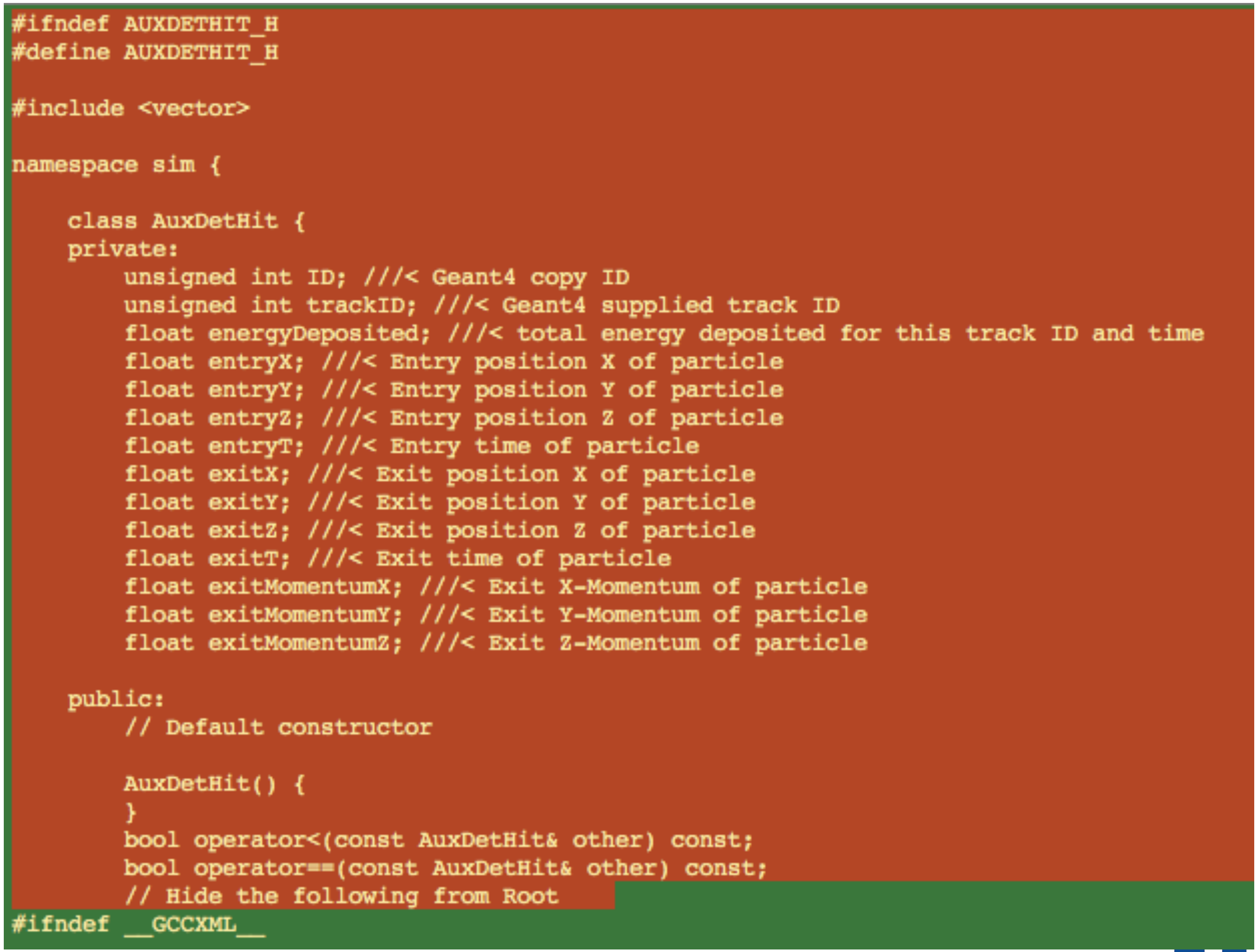


[illegible]

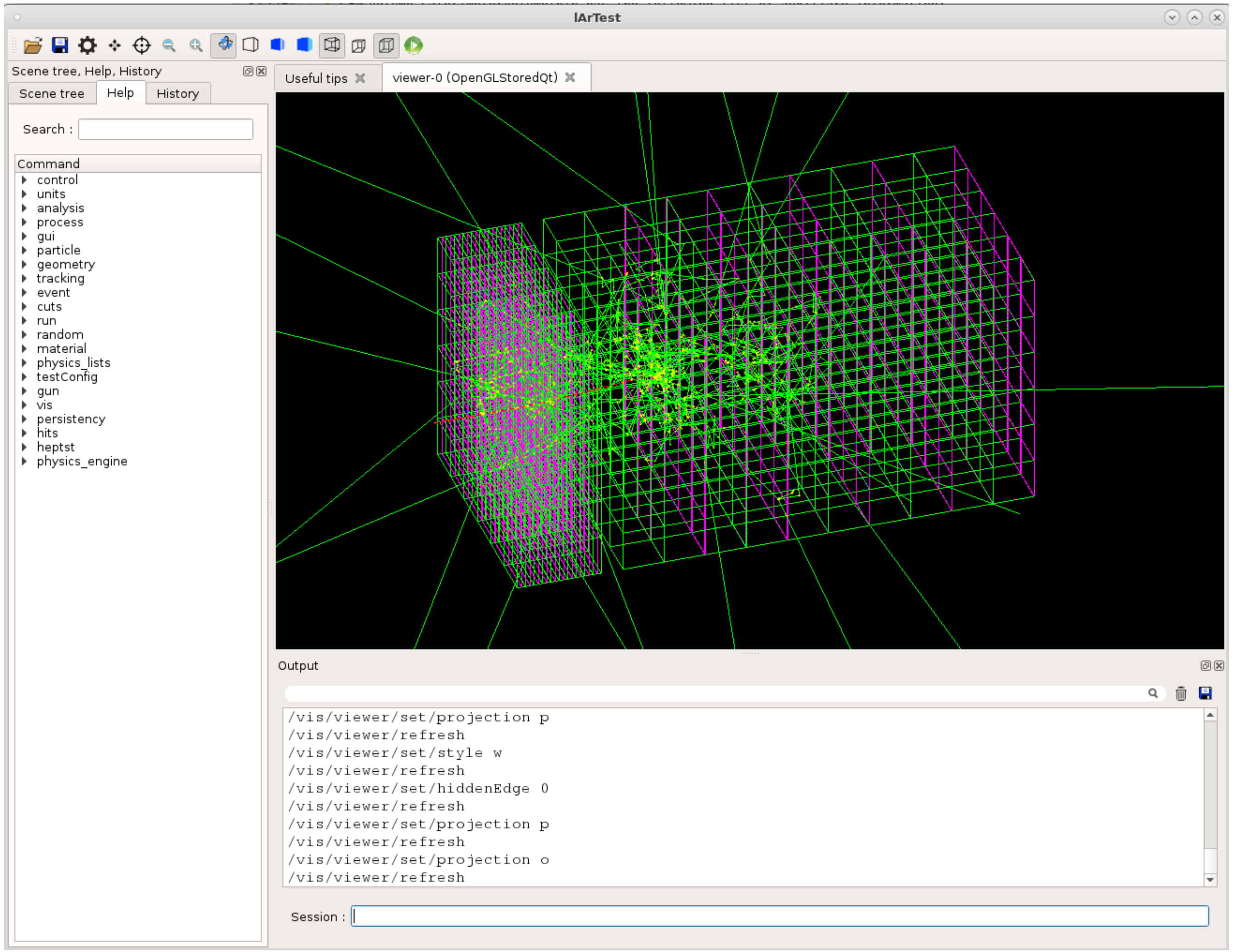
Auxdets

- Good:
  - Implemented as Geant4 sensitive volume.
- Bad:
  - Voxelized
  - Scheme to define channel id’s
  - Requires non sensitive envelope around the sensitive volumes.

AuxdetHit

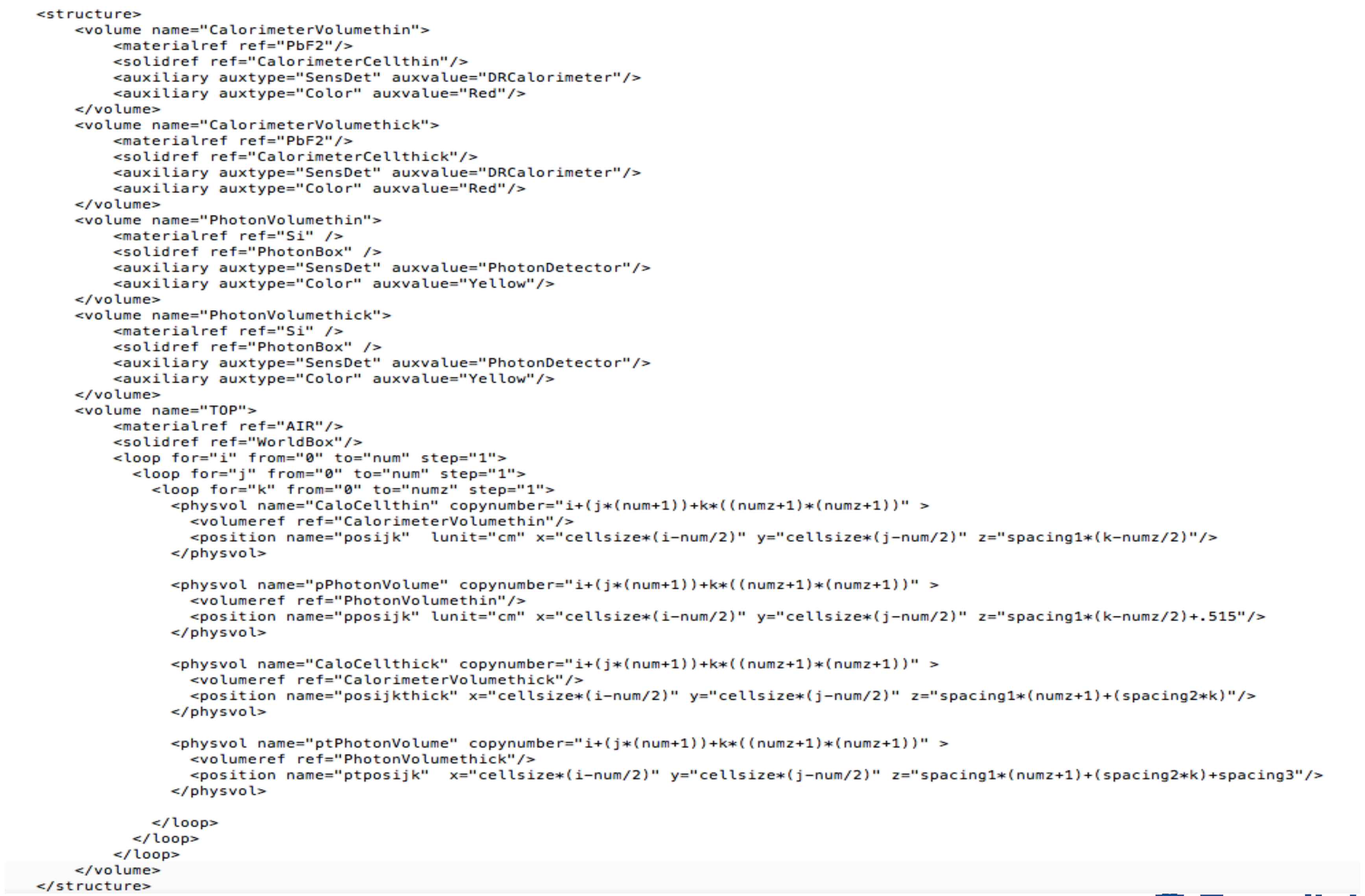


```
#ifndef AUXDETHIT_H
#define AUXDETHIT_H

#include <vector>

namespace sim {

    class AuxDetHit {
    private:
        unsigned int ID; ///< Geant4 copy ID
        unsigned int trackID; ///< Geant4 supplied track ID
        float energyDeposited; ///< total energy deposited for this track ID and time
        float entryX; ///< Entry position X of particle
        float entryY; ///< Entry position Y of particle
        float entryZ; ///< Entry position Z of particle
        float entryT; ///< Entry time of particle
        float exitX; ///< Exit position X of particle
        float exitY; ///< Exit position Y of particle
        float exitZ; ///< Exit position Z of particle
        float exitT; ///< Exit time of particle
        float exitMomentumX; ///< Exit X-Momentum of particle
        float exitMomentumY; ///< Exit Y-Momentum of particle
        float exitMomentumZ; ///< Exit Z-Momentum of particle

    public:
        // Default constructor

        AuxDetHit() {
        }
        bool operator<(const AuxDetHit& other) const;
        bool operator==(const AuxDetHit& other) const;
        // Hide the following from Root
#ifndef __GCCXML__
```



```
/vis/viewer/set/projection p
/vis/viewer/refresh
/vis/viewer/set/style w
/vis/viewer/refresh
/vis/viewer/set/hiddenEdge 0
/vis/viewer/refresh
/vis/viewer/set/projection p
/vis/viewer/refresh
/vis/viewer/set/projection o
/vis/viewer/refresh
```

Each sensitive volume is uniquely defined by name and copy number



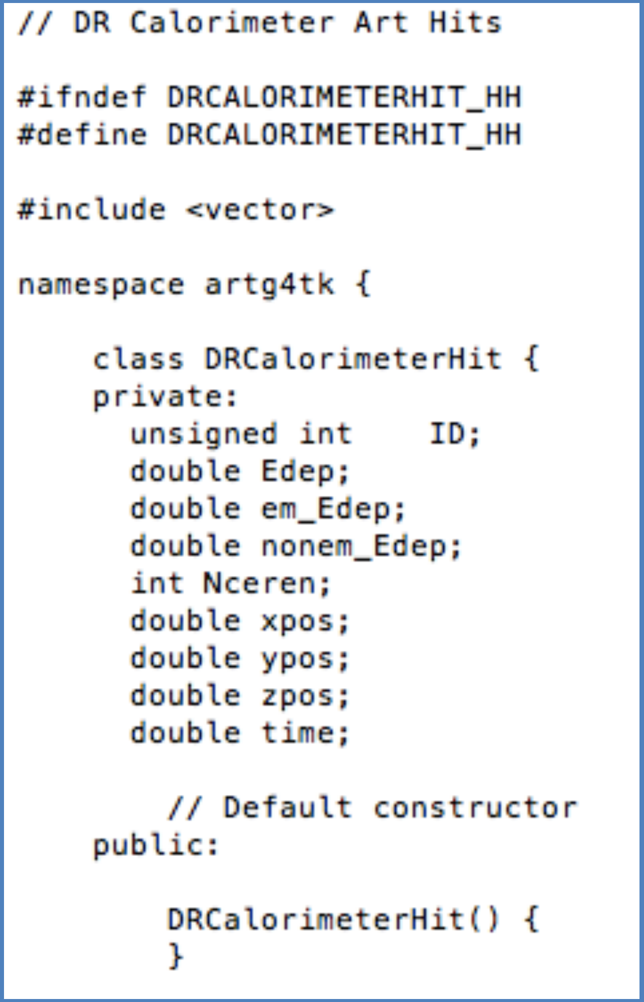
```
<structure>
    <volume name="CalorimeterVolumethin">
        <materialref ref="PbF2"/>
        <solidref ref="CalorimeterCellthin"/>
        <auxiliary auxtype="SensDet" auxvalue="DRCalorimeter"/>
        <auxiliary auxtype="Color" auxvalue="Red"/>
    </volume>
    <volume name="CalorimeterVolumethick">
        <materialref ref="PbF2"/>
        <solidref ref="CalorimeterCellthick"/>
        <auxiliary auxtype="SensDet" auxvalue="DRCalorimeter"/>
        <auxiliary auxtype="Color" auxvalue="Red"/>
    </volume>
    <volume name="PhotonVolumethin">
        <materialref ref="Si" />
        <solidref ref="PhotonBox" />
        <auxiliary auxtype="SensDet" auxvalue="PhotonDetector"/>
        <auxiliary auxtype="Color" auxvalue="Yellow"/>
    </volume>
    <volume name="PhotonVolumethick">
        <materialref ref="Si" />
        <solidref ref="PhotonBox" />
        <auxiliary auxtype="SensDet" auxvalue="PhotonDetector"/>
        <auxiliary auxtype="Color" auxvalue="Yellow"/>
    </volume>
    <volume name="TOP">
        <materialref ref="AIR"/>
        <solidref ref="WorldBox"/>
        <loop for="i" from="0" to="num" step="1">
          <loop for="j" from="0" to="num" step="1">
            <loop for="k" from="0" to="numz" step="1">
              <physvol name="CaloCellthin" copynumber="i+(j*(num+1))+k*((numz+1)*(numz+1))" >
                <volumeref ref="CalorimeterVolumethin"/>
                <position name="posijk"  lunit="cm" x="cellsize*(i-num/2)" y="cellsize*(j-num/2)" z="spacing1*(k-numz/2)"/>
              </physvol>

              <physvol name="pPhotonVolume" copynumber="i+(j*(num+1))+k*((numz+1)*(numz+1))" >
                <volumeref ref="PhotonVolumethin"/>
                <position name="pposijk" lunit="cm" x="cellsize*(i-num/2)" y="cellsize*(j-num/2)" z="spacing1*(k-numz/2)+.515"/>
              </physvol>

              <physvol name="CaloCellthick" copynumber="i+(j*(num+1))+k*((numz+1)*(numz+1))" >
                <volumeref ref="CalorimeterVolumethick"/>
                <position name="posijkthick" x="cellsize*(i-num/2)" y="cellsize*(j-num/2)" z="spacing1*(numz+1)+(spacing2*k)"/>
              </physvol>

              <physvol name="ptPhotonVolume" copynumber="i+(j*(num+1))+k*((numz+1)*(numz+1))" >
                <volumeref ref="PhotonVolumethick"/>
                <position name="ptposijk"  x="cellsize*(i-num/2)" y="cellsize*(j-num/2)" z="spacing1*(numz+1)+(spacing2*k)+spacing3"/>
              </physvol>

            </loop>
          </loop>
        </loop>
    </volume>
</structure>
```

Translate to unique appropriate channel mapping

Channel mapper depending on your experiment

In this case just reverse the formula

```
unsigned int lay = (int)(hit.GetID()/(numz*numz));
unsigned int col = (int)(((hit.GetID()-kk*(numz*numz)))/numy);
unsigned int row = (int)(hit.GetID()-kk*(numz*numz)-jj*numy);
cout << "ID:  "<<hit.GetID()<<"  Layer:  ”<< lay
                                          <<"  Column:  " << col
                                         << "  row:  "<< row
                                         <<  endl;
```



```
// DR Calorimeter Art Hits

#ifndef DRCALORIMETERHIT_HH
#define DRCALORIMETERHIT_HH

#include <vector>

namespace artg4tk {

    class DRCalorimeterHit {
    private:
      unsigned int    ID;
      double Edep;
      double em_Edep;
      double nonem_Edep;
      int Nceren;
      double xpos;
      double ypos;
      double zpos;
      double time;

        // Default constructor
    public:

        DRCalorimeterHit() {
        }
```

Finally:

- Should finish up AuxDetSD and AuxDetHit today.
- Git feature branches :
  - artg4tk: feature/wenzel_artg4tk_cleanup
  - lardataobj: feature/wenzel_larg4refac_phase2_1
  - larg4: feature/wenzel_larg4refac_phase2_1

Backup

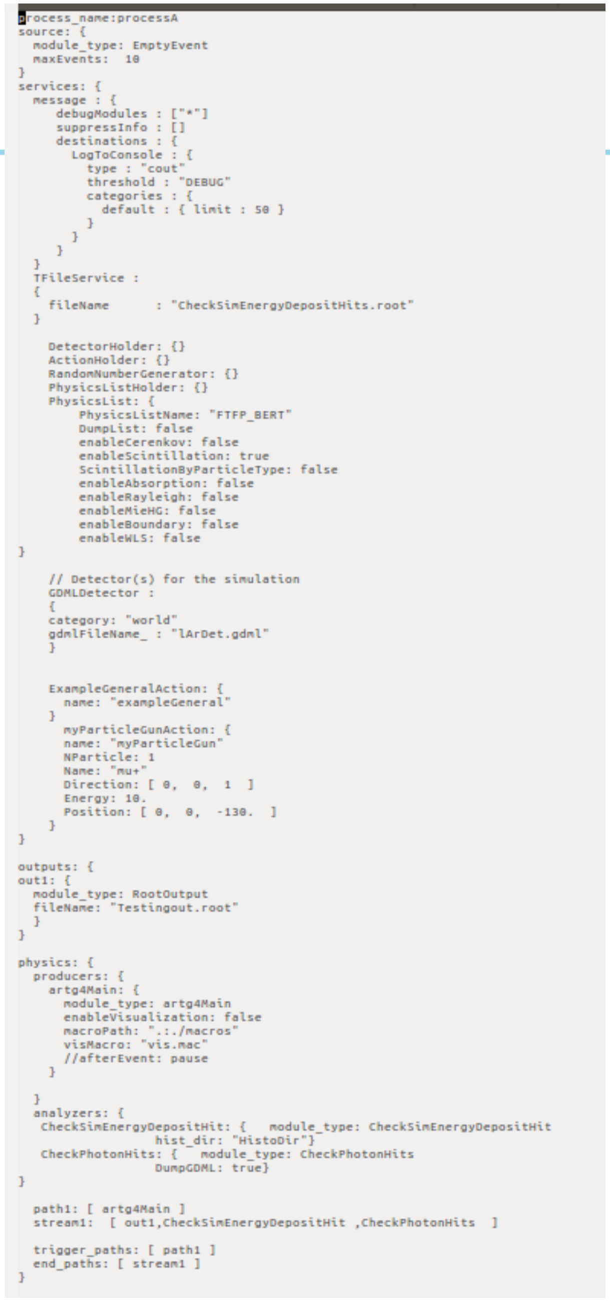
GDML(+ extensions): a complete description of detector configuration (at runtime)

- Materials, volumes etc.….
- Assign step-limits to specific volumes.
- Optical properties (bulk and surface)
- Assignment of sensitive detectors of predefined type to logical volumes→ automatically trigger the creation and filling of the appropriate hit collections
- Assignment of optical surfaces
- Visualization attributes (color, solid,….)
- Makes use of formulas and loops to keep gdml file compact
- Homogeneous electric field (no electric field→ no separation of charge)
-

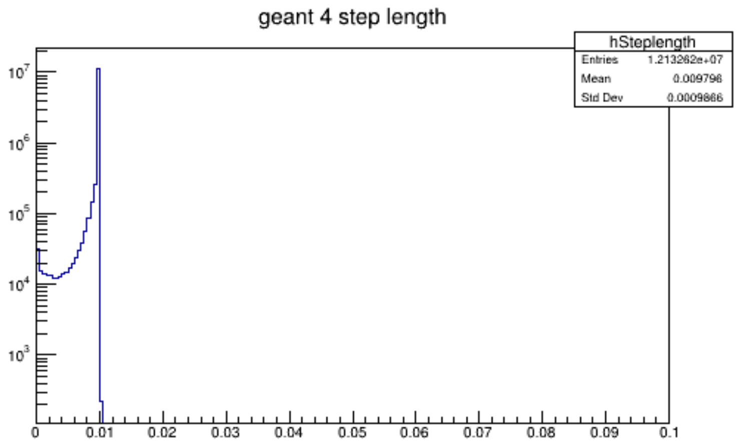
Physics

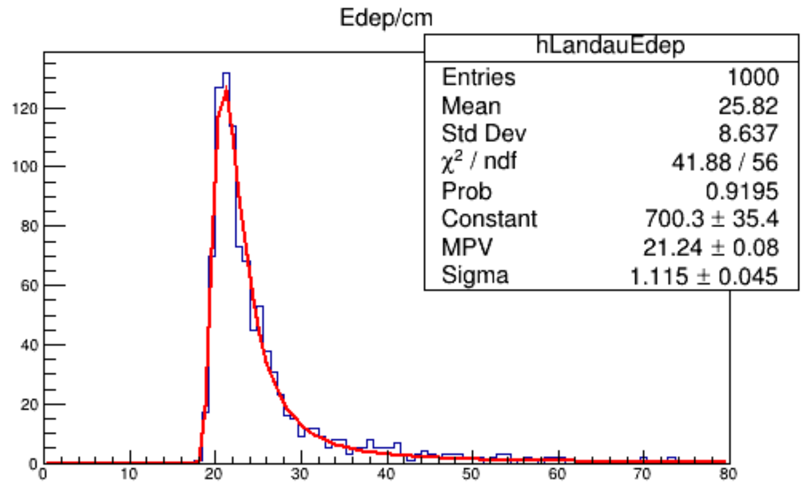
Use the new Geant4 physics list factory:

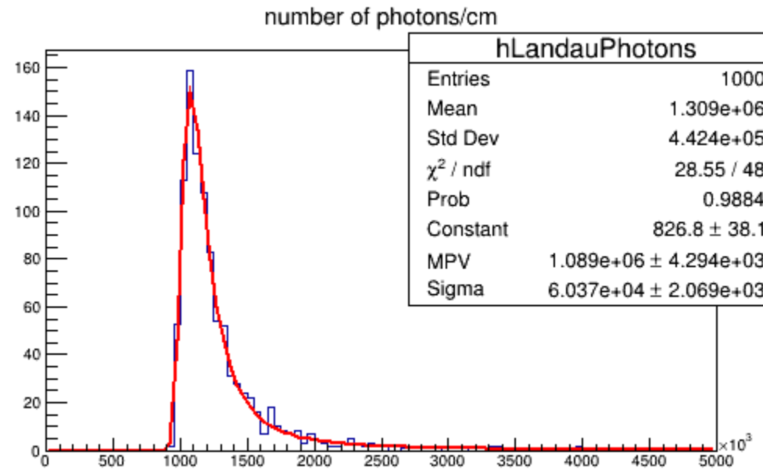
- Access to all reference physics list
- If you a brave you can register your own list
- Extendable to add all the available geant4 physics constructors e.g. we use:
  - Optical physics (complete: Rayleigh scattering, absorption, Cerenkov scintillation, boundary, surface, wls …..
  - Step limiter for charged particles in active TPC volume.
  - Time limiter for neutrons.
  - More precise em physics can be selected.
- Can be controlled via fhcl parameters
  -

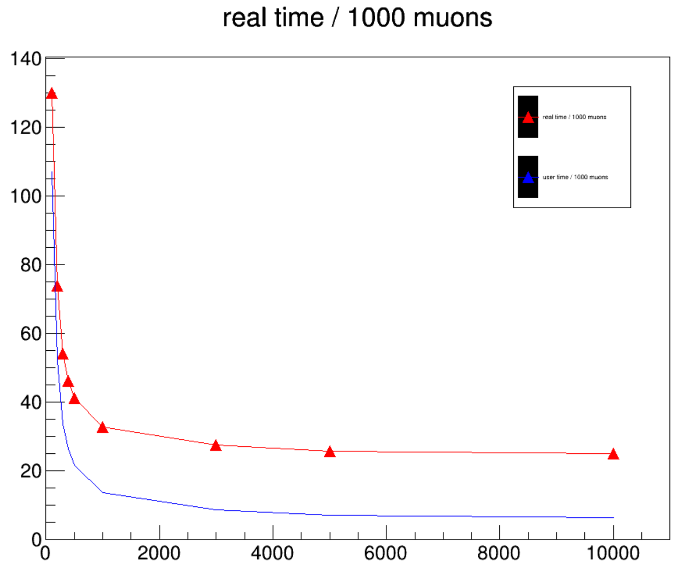


```
process_name:processA
source: {
  module_type: EmptyEvent
  maxEvents:  10
}
services: {
  message : {
     debugModules : ["*"]
     suppressInfo : []
     destinations : {
       LogToConsole : {
         type : "cout"
         threshold : "DEBUG"
         categories : {
           default : { limit : 50 }
         }
       }
     }
  }
  TFileService :
  {
    fileName       : "CheckSimEnergyDepositHits.root"
  }

    DetectorHolder: {}
    ActionHolder: {}
    RandomNumberGenerator: {}
    PhysicsListHolder: {}
    PhysicsList: {
        PhysicsListName: "FTFP_BERT"
        DumpList: false
        enableCerenkov: false
        enableScintillation: true
        ScintillationByParticleType: false
        enableAbsorption: false
        enableRayleigh: false
        enableMieHG: false
        enableBoundary: false
        enableWLS: false
}

    // Detector(s) for the simulation
    GDMLDetector :
    {
    category: "world"
    gdmlFileName_ : "lArDet.gdml"
    }


    ExampleGeneralAction: {
      name: "exampleGeneral"
    }
      myParticleGunAction: {
      name: "myParticleGun"
      NParticle: 1
      Name: "mu+"
      Direction: [ 0,  0,  1  ]
      Energy: 10.
      Position: [ 0,  0,  -130.  ]
    }
}

outputs: {
out1: {
  module_type: RootOutput
  fileName: "Testingout.root"
  }
}

physics: {
  producers: {
    artg4Main: {
      module_type: artg4Main
      enableVisualization: false
      macroPath: ".:./macros"
      visMacro: "vis.mac"
      //afterEvent: pause
    }

  }
  analyzers: {
   CheckSimEnergyDepositHit: {   module_type: CheckSimEnergyDepositHit
                  hist_dir: "HistoDir"}
   CheckPhotonHits: {   module_type: CheckPhotonHits
                  DumpGDML: true}
}

  path1: [ artg4Main ]
  stream1:  [ out1,CheckSimEnergyDepositHit ,CheckPhotonHits  ]

  trigger_paths: [ path1 ]
  end_paths: [ stream1 ]
}
```


geant 4 step length
hSteplength
Entries 1.213262e+07
Mean 0.009796
Std Dev 0.0009866


Edep/cm
hLandauEdep
Entries 1000
Mean 25.82
Std Dev 8.637
χ² / ndf 41.88 / 56
Prob 0.9195
Constant 700.3 ± 35.4
MPV 21.24 ± 0.08
Sigma 1.115 ± 0.045


number of photons/cm
hLandauPhotons
Entries 1000
Mean 1.309e+06
Std Dev 4.424e+05
χ² / ndf 28.55 / 48
Prob 0.9884
Constant 826.8 ± 38.1
MPV 1.089e+06 ± 4.294e+03
Sigma 6.037e+04 ± 2.069e+03
×10³


real time / 1000 muons
real time / 1000 muons
user time / 1000 muons

step length

Assigning Sensitive detector to a volume:

```
<volume name="volTPCActiveInner">
    <materialref ref="LAr"/>
    <solidref ref="TPCVolume"/>
    <auxiliary auxtype="SensDet" auxvalue="SimEnergyDeposit"/>
    <auxiliary auxtype="Color" auxvalue="Blue"/>
    <auxiliary auxtype="StepLimit" auxvalue="0.01"/>
    <auxiliary auxtype="Efield" auxvalue="1000."/>
    <loop for="i" from="0" to="num" step="1">
        <physvol name="psenseWireVolume">
            <volumeref ref="SenseWire"/>
            <position name="posijk"  unit="mm" x="-200.0+(i+1)*5." y="-199.8" z="0"/>
        </physvol>
    </loop>
</volume>
<volume name="volPhotodetector">
    <materialref ref="Silicon"/>
    <solidref ref="PhotoBox"/>
    <auxiliary auxtype="SensDet" auxvalue="PhotonDetector"/>
    <auxiliary auxtype="Color" auxvalue="Red"/>
    <auxiliary auxtype="Solid" auxvalue="True"/>
</volume>
<volume name="volArgon">
    <materialref ref="LAr"/>
    <solidref ref="ArgonVolume"/>
    <auxiliary auxtype="SensDet" auxvalue="SimEnergyDeposit"/>
    <auxiliary auxtype="Color" auxvalue="Yellow"/>
    <physvol name="pCalorimeterVolume">
        <volumeref ref="volTPCActiveInner"/>
        <position name="Calpos" x="0" y="0" z="0"/>
    </physvol>
    <physvol name="pvolPhotodetector">
        <volumeref ref="volPhotodetector"/>
        <position name="photondetectorpos" unit="mm" x="0" y="391." z="0"/>
    </physvol>
</volume>
```

Configure the Physics in the fcl file

```
    DetectorHolder: {}
    ActionHolder: {}
    RandomNumberGenerator: {}
    PhysicsListHolder: {}
    PhysicsList: {
            PhysicsListName: "FTFP_BERT"
            DumpList: false
            enableCerenkov: false
            enableScintillation: true
            ScintillationByParticleType: false
            enableAbsorption: false
            enableRayleigh: false
            enableMieHG: false
            enableBoundary: false
            enableWLS: false
}

    // Detector(s) for the simulation
    GDMLDetector :
    {
    category: "world"
    gdmlFileName_ : "lArDet.gdml"
    }
```

Physics constructor

```
artg4tk::PhysicsListService::PhysicsListService(fhicl::ParameterSet const & p, art::ActivityRegistry &) :
  PhysicsListName_( p.get<std::string>("PhysicsListName","FTFP_BERT")),
  DumpList_( p.get<bool>("DumpList",true)),
  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>("ScintillationTrackSecondariesFirst",false)),
  enableScintillation_( p.get<bool>("enableScintillation",true)),
  ScintillationStackPhotons_( p.get<bool>("ScintillationStackPhotons",false)),
  ScintillationByParticleType_( p.get<bool>("ScintillationByParticleType",false)),
  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)),
  WLSProfile_( p.get<std::string>(" WLSProfile","delta"))
{}
```

```
G4VUserPhysicsList* artg4tk::PhysicsListService::makePhysicsList() {



  g4alt::G4PhysListFactory factory;
    // Access to registries and factories
    //
  G4PhysicsConstructorRegistry* g4pcr = G4PhysicsConstructorRegistry::Instance();
  G4PhysListRegistry* g4plr = G4PhysListRegistry::Instance();
  //
  // the following should be unneccessary at some point:
  //
  g4plr->AddPhysicsExtension("OPTICAL", "G4OpticalPhysics");
  g4plr->AddPhysicsExtension("STEPLIMIT", "G4StepLimiterPhysics");
  g4plr->AddPhysicsExtension("NEUTRONLIMIT", "G4NeutronTrackingCut");
  g4pcr->PrintAvailablePhysicsConstructors();
  g4plr->PrintAvailablePhysLists();
  G4VModularPhysicsList* phys = NULL;
  G4String physName = PhysicsListName_;
  if (enableOptical_)   physName=physName+"+OPTICAL";
  if (enableStepLimit_) physName=physName+"+STEPLIMIT";
  if (enableNeutronLimit_)  physName=physName+"+NEUTRONLIMIT";
  std::cout << " Name of Physics list: "<< physName<<std::endl;
  if (factory.IsReferencePhysList(physName)) {
    phys = factory.GetReferencePhysList(physName);
  }
    std::cout << phys->GetPhysicsTableDirectory() << std::endl;
    if (enableOptical_)
      {
                G4OpticalPhysics* opticalPhysics = (G4OpticalPhysics*) phys->GetPhysics("Optical");
                opticalPhysics->Configure(kCerenkov,  enableCerenkov_);
                opticalPhysics->SetCerenkovStackPhotons(CerenkovStackPhotons_);
                opticalPhysics->Configure(kScintillation,  enableScintillation_);
                opticalPhysics->SetScintillationStackPhotons(ScintillationStackPhotons_);
                opticalPhysics->SetScintillationByParticleType(ScintillationByParticleType_);
                opticalPhysics->SetScintillationTrackInfo(ScintillationTrackInfo_);
                opticalPhysics->SetTrackSecondariesFirst(kCerenkov, true);      // only relevant if we actually stack and trace the optical photons
                opticalPhysics->SetTrackSecondariesFirst(kScintillation, true); // only relevant if we actually stack and trace the optical photons
                opticalPhysics->SetMaxNumPhotonsPerStep(CerenkovMaxNumPhotons_);
                opticalPhysics->SetMaxBetaChangePerStep( CerenkovMaxBetaChange_);
                opticalPhysics->Configure(kAbsorption,enableAbsorption_);
                opticalPhysics->Configure(kRayleigh,enableRayleigh_);
                opticalPhysics->Configure(kMieHG,enableMieHG_);
                opticalPhysics->Configure(kBoundary,enableBoundary_);
                opticalPhysics->Configure(kWLS,enableWLS_);
      }
    if (enableNeutronLimit_)
      {
                G4NeutronTrackingCut * neutrcut = (G4NeutronTrackingCut*) phys->GetPhysics("neutronTrackingCut");
                neutrcut->SetTimeLimit(NeutronTimeLimit_);
      }
    if (DumpList_)
      {
                phys->DumpList();
                phys->DumpCutValuesTable();
      }
    return phys;
}


using artg4tk::PhysicsListService;
```

Outline

- Requirements
- What does LArG4 do?
- How to refactor it status thereof/ how to access artg4tk
- Selection of physics list, processes etc. fcl parameters
- artg4tk how to run it.
- How to define a detector in gdml.
- Some results

LArG4_module (what's wrong with it?)

- Monolithic hard to work with, not very flexible.
- Mixes digitization, electron drift, detector response …. with Geant4 simulation
- Mixes gdml description with properties extracted from Material properties service. (incomplete, magic words….) → gdml is a much more complete and flexible description
- Takes routines out of Geant4 and modifies them instead of using standard interfaces
- Physics processes incomplete, stuck with restrictions that existed at the time...
- Binds processes to specific material/volume (old restriction/implementation of Geant4, at the time optical properties were not treated as material properties→ only one material with opt. properties)
- Read out photons only in TPC volume. Where there is no electrical field all deposited energy goes to light
- Voxel readout. (even for Aux detectors)
- TPC implemented as Stepping Action not as Sensitive detector → each step needs to check which volume its in.
- Weird/inefficient interface to Geant 4 user actions (UserAction)
- Old physics list
- Magic words and hard coded stuff all over the place….
- Depends on nutools…
- …

Next

- Finish ParticleListAction
- Sit with Paul and Dual readout group to discuss how to plugin the photon table
- Plugin scintillation and Ionization as alternative to using Geant4 Scintillation.
- AuxDetector→TrackerSD→ TrackerHit→ module that reads them and creates the corresponding SimChannel.
- Clean up, make it less verbose, tagging packaging….