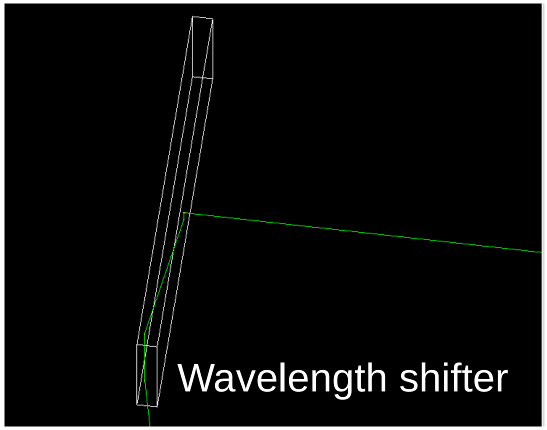
LArG4 refactoring status and plan

Hans Wenzel

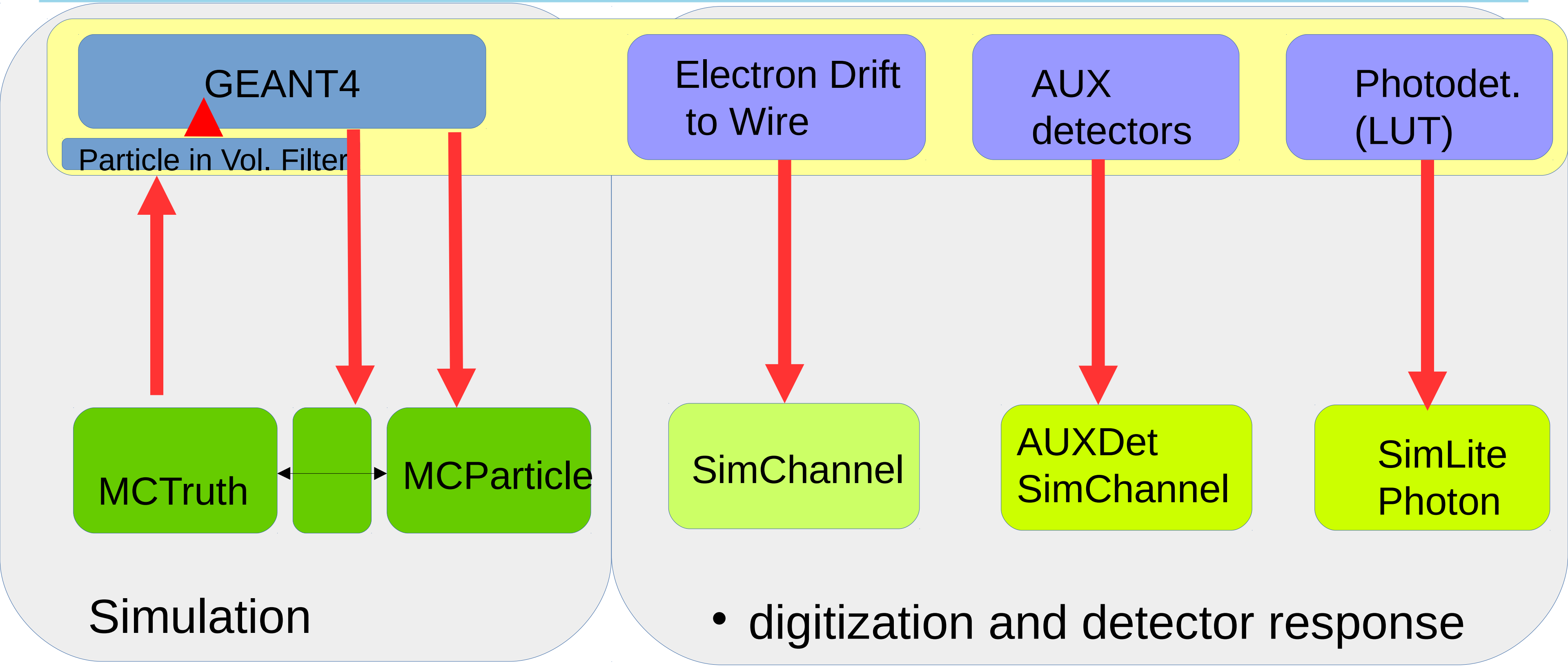
31th July 2018

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.
  - Read out optical Scintillation photons from the liquid Argon surrounding the TPC.
  - Be able to do fast (LUT) and complete simulation of optical photon by just switching fhcl parameters.
- digitization and detector response:
  - Ability to plug in models handling the correlations between ionization and Scintillation (e.g. Nest)
  - Ability to switch drift model.
- Support for dual phase..



Current LArG4_module



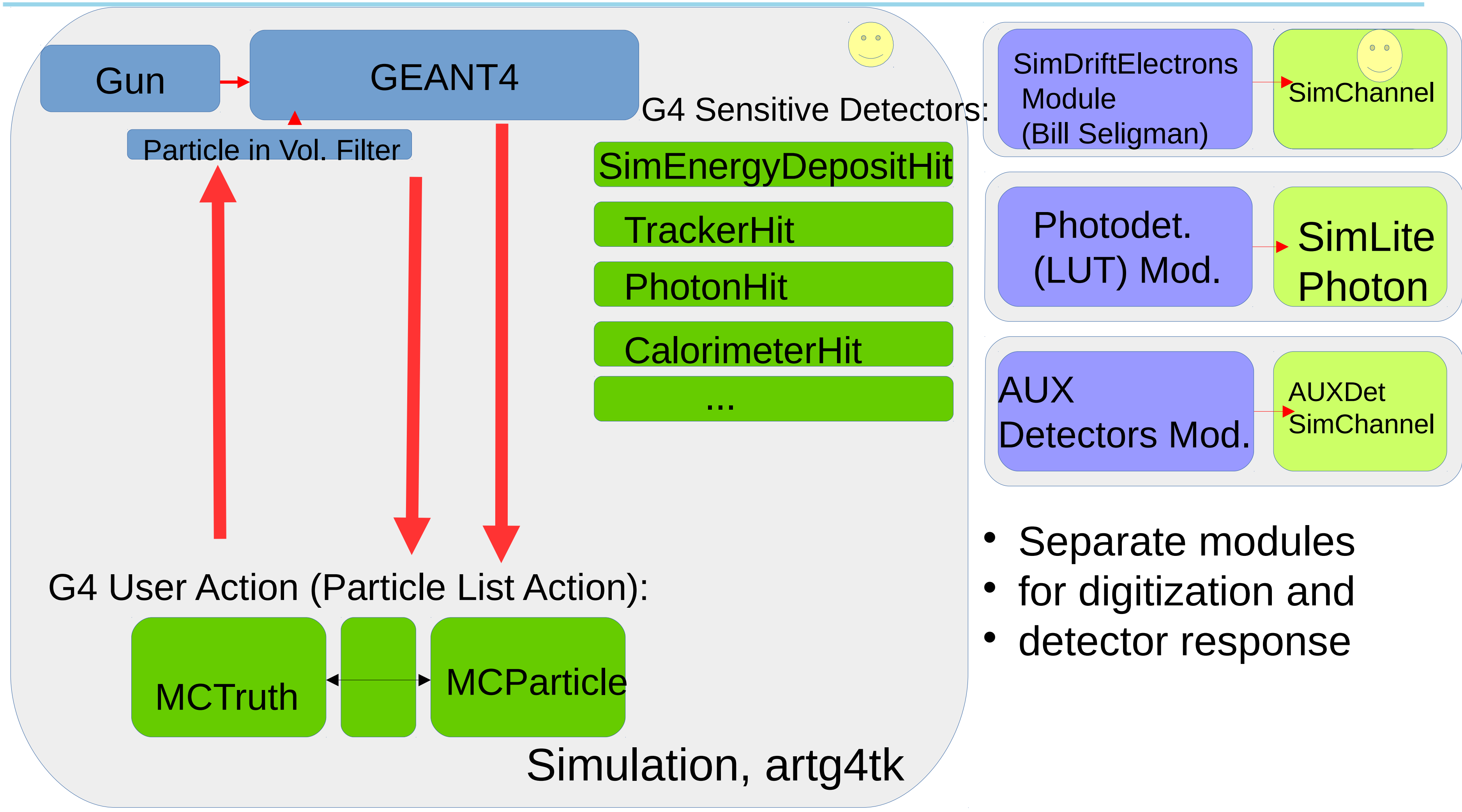
LArG4_module (what’s wrong with it?)

- 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…
-

LArG4_module (what’s wrong with it?)

- Photon detector that doesn’t chack if the incident particle is actually an optical photon
- The list goes on and on ….

Refactored



- Separate modules
- for digitization and
- detector response

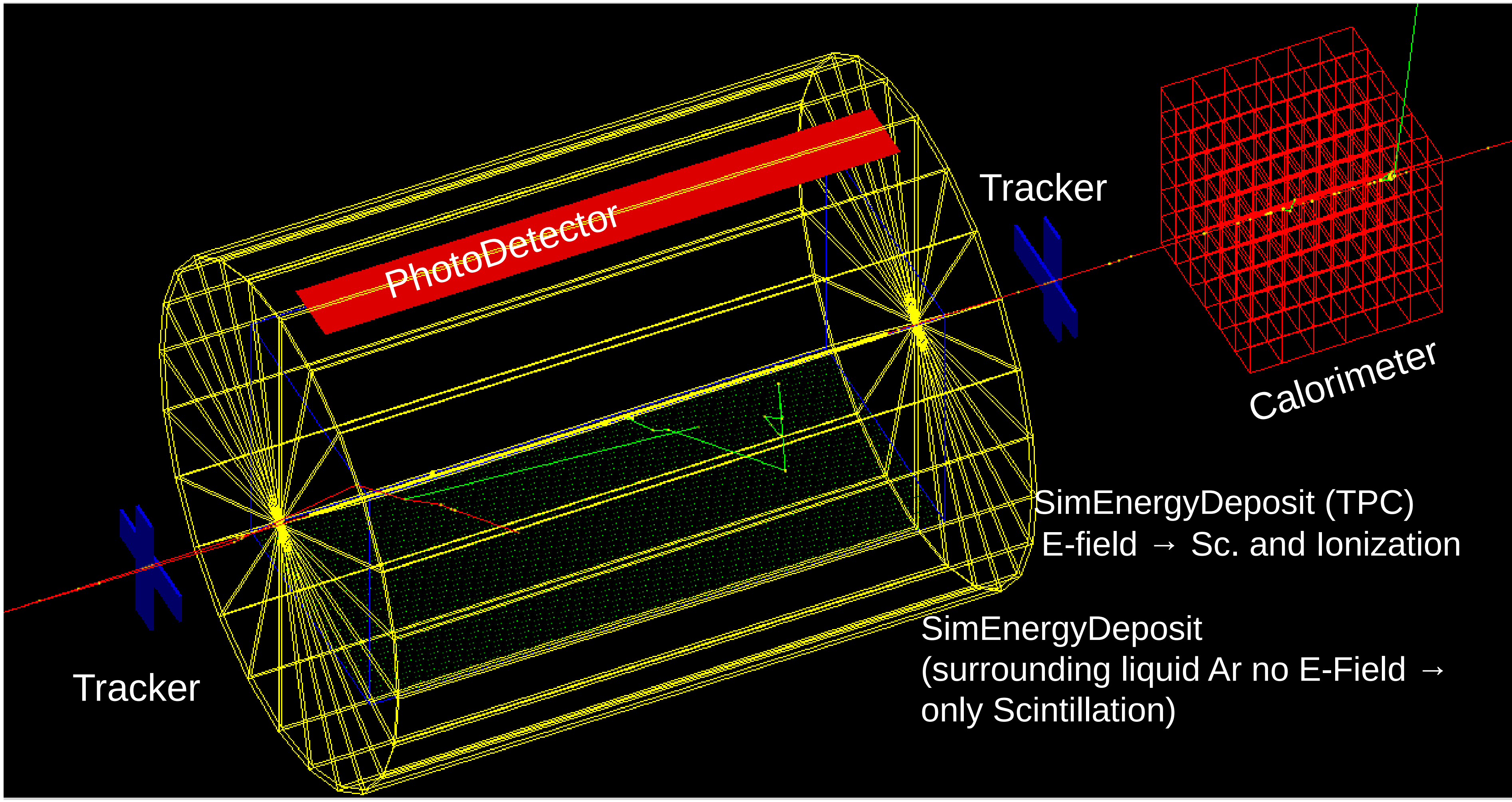
Artg4tk

- Started with artg4 that Adam Lyon 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 → to estimate systematic uncertainties.
- Very flexible framework which allows to create different detector configurations quickly → no knowledge of Geant4 internals necessary. Good for detector R&D.

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

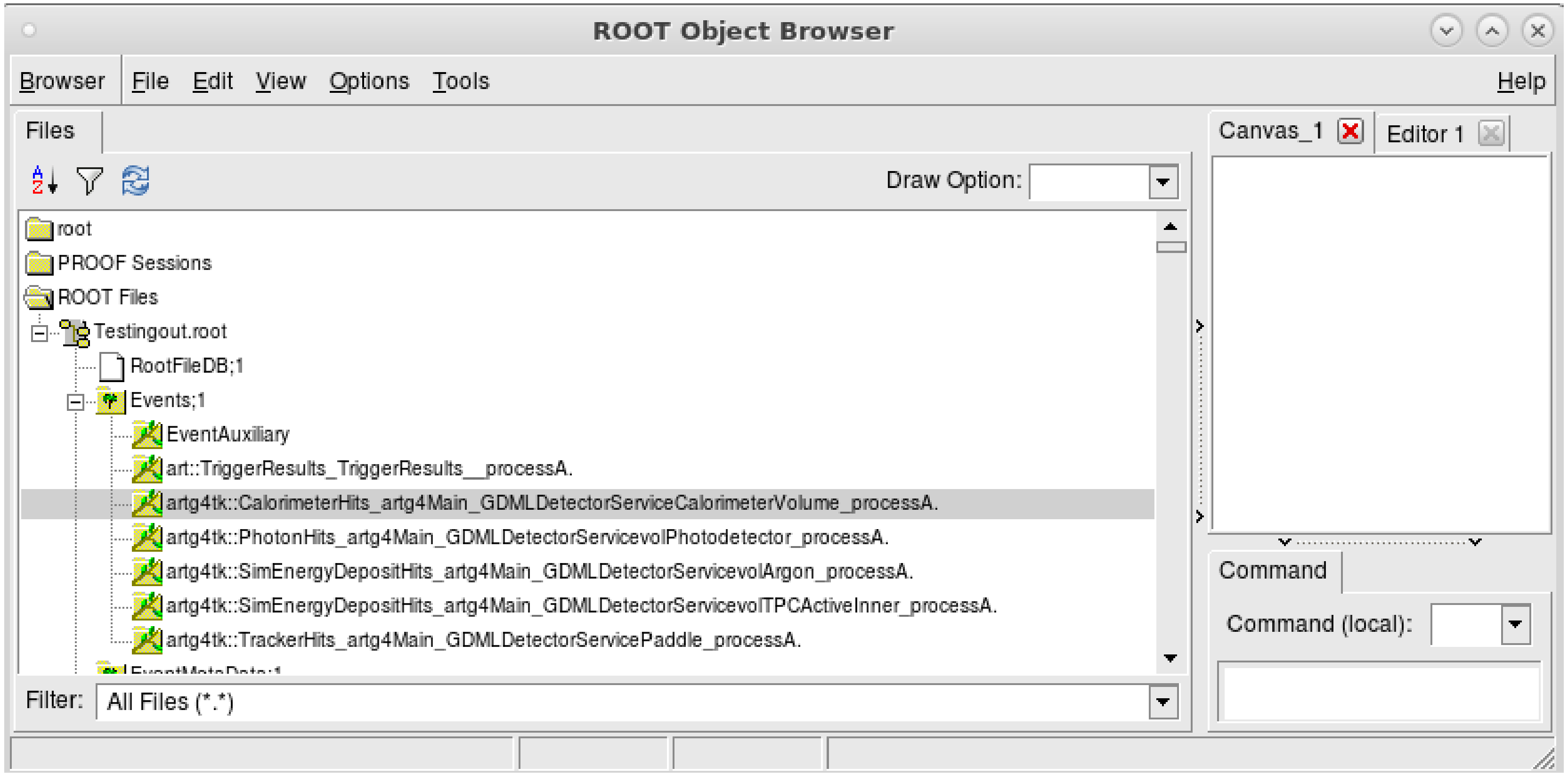
- Materials, volumes etc.….
- Assign step-limits to specific volumes.
- Optical properties (bulk and surface)
- Assignment of sensitive detectors of predefined type to logical volumes→ automatically trigger the creation and filling of the appropriate hit collections
- Assignment of optical surfaces
- 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)
-

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


PhotoDetector
Tracker
Calorimeter
SimEnergyDeposit (TPC)
E-field → Sc. and Ionization
SimEnergyDeposit
(surrounding liquid Ar no E-Field → only Scintillation)
Tracker

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.

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 -> 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)

Status

- Created release candidate last week. Lynn created larsoft v07_00_00 with the new larg4. No change unless changes to fhcl and gdml file.
- larg4 repository contains the bits that depend on the various components of larsoft e.g. SimEnergyDepositSD, 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).
- TrackerSD to replace Auxdetector.
- 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.

Big thanks to Paul Russo, Lynn and Bill Seligman for lots of help and discussions!

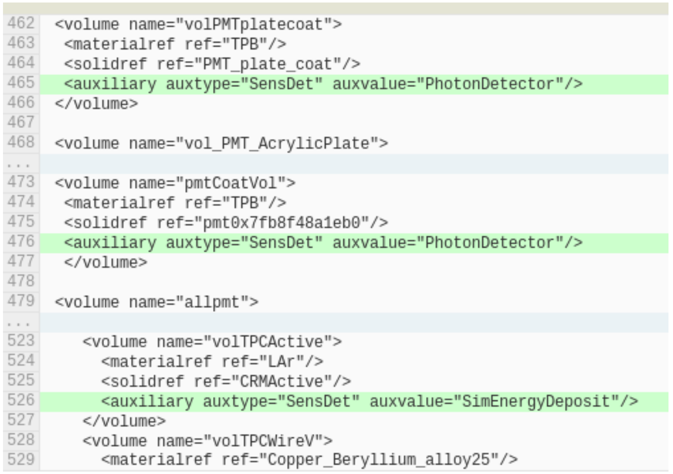
How to use it

- Changes to gdml file:
    - Attach Sensitive detector to logical volumes.
    - Provide optical properties in gdml.
    - Provide step limit in gdml

- Changes to fhcl file
    - Select new geant 4 module.
    - New electron drift module.

Testing the refactored larsoft (Jose Alfonso Soto Oton)

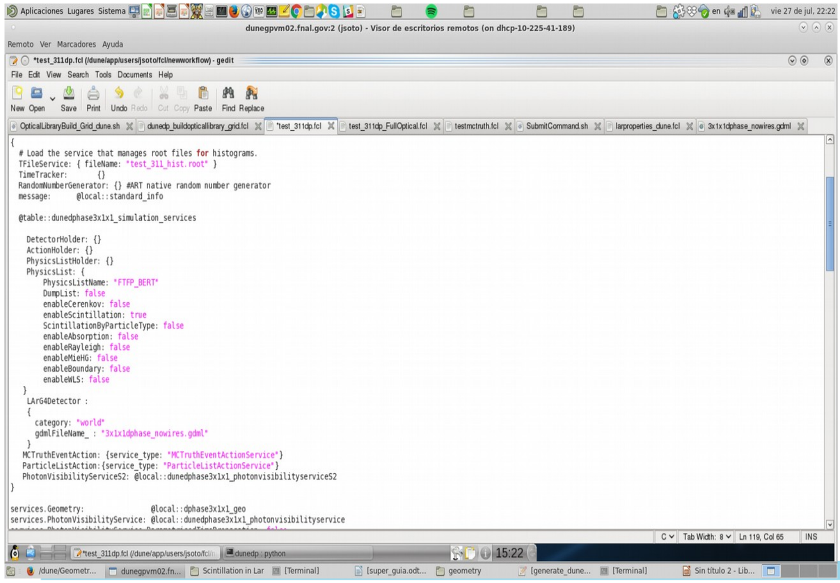
- The new refactored larsoft has been tested using the 3x1x1 dual phase TPC geometry.
- Only a few changes are needed to run it:
- In the GDML:



```
<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"/>
```

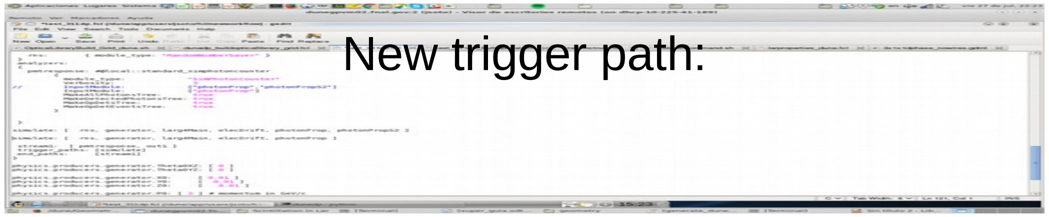
- In the FHICL:

List of new services:



```
{
 # Load the service that manages root files for histograms.
 TFileService: { fileName: "test_311_hist.root" }
 TimeTracker:       {}
 RandomNumberGenerator: {} #ART native random number generator
 message:      @local::standard_info

 @table::dunedphase3x1x1_simulation_services

   DetectorHolder: {}
   ActionHolder: {}
   PhysicsListHolder: {}
   PhysicsList: {
       PhysicsListName: "FTFP_BERT"
       DumpList: false
       enableCerenkov: false
       enableScintillation: true
       ScintillationByParticleType: false
       enableAbsorption: false
       enableRayleigh: false
       enableMieHG: false
       enableBoundary: false
       enableWLS: false
  }
   LArG4Detector :
   {
     category: "world"
     gdmlFileName_ : "3x1x1dphase_nowires.gdml"
   }
  MCTruthEventAction: {service_type: "MCTruthEventActionService"}
  ParticleListAction:{service_type: "ParticleListActionService"}
  PhotonVisibilityServiceS2: @local::dunedphase3x1x1_photonvisibilityserviceS2
}

services.Geometry:                 @local::dphase3x1x1_geo
services.PhotonVisibilityService: @local::dunedphase3x1x1_photonvisibilityservice
```

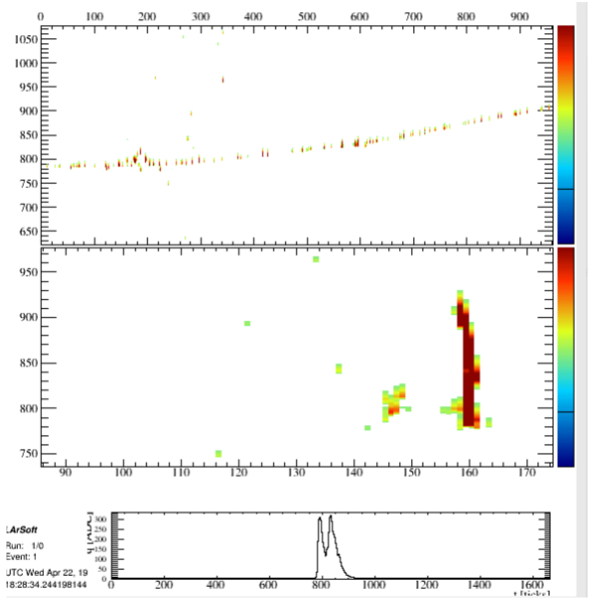
New trigger path:



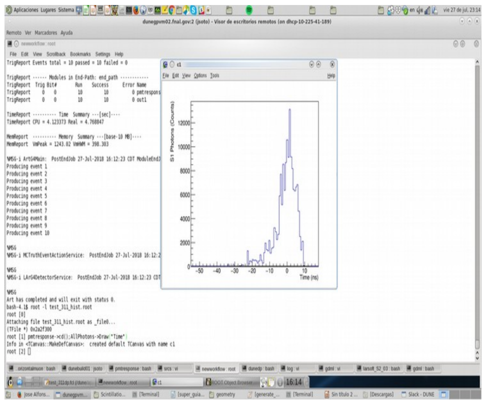
```
   rns:          { module_type: "RandomNumberSaver" }
  }
 analyzers:
 {
   pmtresponse: @local::standard_simphotoncounter
         {
           module_type:                  "SimPhotonCounter"
           Verbosity:                    3
//         InputModule:                  ["photonProp", "photonPropS2"]
           InputModule:                  ["photonProp"]
           MakeAllPhotonsTree:           true
           MakeDetectedPhotonsTree:      true
           MakeOpDetsTree:               true
           MakeOpDetEventsTree:          true
         }

 }
simulate: [  rns, generator, larg4Main, elecDrift, photonProp, photonPropS2 ]
simulate: [  rns, generator, larg4Main, elecDrift, photonProp ]
 stream1:   [ pmtresponse, out1 ]
 trigger_paths: [simulate]
 end_paths:     [stream1]
}
physics.producers.generator.Theta0XZ: [ 0 ]
physics.producers.generator.Theta0YZ: [ 0 ]
physics.producers.generator.X0:       [ 0.01 ]
physics.producers.generator.Y0:       [  0.01 ]
physics.producers.generator.Z0:       [   0.01 ]
physics.producers.generator.P0: [ 3 ] # momentum in GeV/c
```

Testing the refactored larsoft

- Three paths to test:
  - **Fast Optical simulation:** Electrons are drifted towards the new module, and photons propagated through t look-up table.
    - ✓Charge looks ok, and detsim can be run (see below)
    - ✓S2 photons are well generated. (see below the result of SimPhotonCounter)
    - ✗S1 photons are produced as SimPhotons, see below the result of SimPhotonCounter (SimPhotonsLite are needed, and PhotonLibraryPropagation still needs to fix some bugs)
  - **Optical simulation:** Electrons are drifted towards the new module, and photons are tracked by G4 up to th photon detectors.
    - ✗ Still not available.
  - **Generating photon libraries:** Track of photons without charged particles.
    - ✗ Still not available.

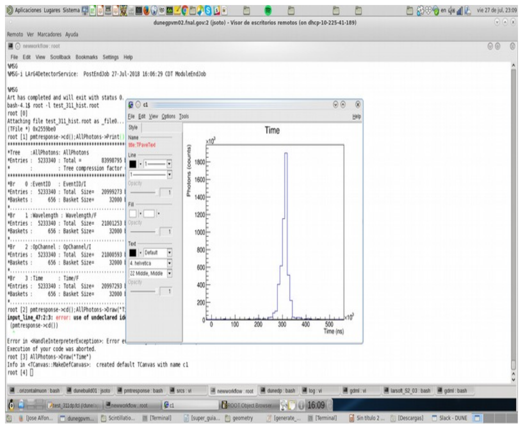


S1 photons



```
TrigReport Events total = 10 passed = 10 failed = 0

TrigReport ------ Modules in End-Path: end_path ------------
TrigReport  Trig Bit#        Run    Success     Error Name
TrigReport     0    0         10         10         0 pmtrespons
TrigReport     0    0         10         10         0 out1

TimeReport ---------- Time  Summary ---[sec]----
TimeReport CPU = 4.123373 Real = 4.768847

MemReport  ---------- Memory  Summary ---[base-10 MB]----
MemReport  VmPeak = 1243.02 VmHWM = 398.303

%MSG-i ArtG4Main:  PostEndJob 27-Jul-2018 16:12:23 CDT ModuleEndJ
Producing event 1
Producing event 2
Producing event 3
Producing event 4
Producing event 5
Producing event 6
Producing event 7
Producing event 8
Producing event 9
Producing event 10

%MSG
%MSG-i MCTruthEventActionService:  PostEndJob 27-Jul-2018 16:12:2

%MSG
%MSG-i LArG4DetectorService:  PostEndJob 27-Jul-2018 16:12:23 CDT

%MSG
Art has completed and will exit with status 0.
bash-4.1$ root -l test_311_hist.root
root [0]
Attaching file test_311_hist.root as _file0...
(TFile *) 0x2a2f300
root [1] pmtresponse->cd();AllPhotons->Draw("Time")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2] 
```

S2 photons



```
%MSG
%MSG-i LArG4DetectorService:  PostEndJob 27-Jul-2018 16:06:29 CDT ModuleEndJob

%MSG
Art has completed and will exit with status 0.
bash-4.1$ root -l test_311_hist.root
root [0]
Attaching file test_311_hist.root as _file0...
(TFile *) 0x2559be0
root [1] pmtresponse->cd();AllPhotons->Print()
******************************************************
*Tree    :AllPhotons: AllPhotons
*Entries :  5233340 : Total =        83998795
*        :          : Tree compression factor
******************************************************
*Br    0 :EventID   : EventID/I
*Entries :  5233340 : Total  Size=   20999273
*Baskets :      656 : Basket Size=      32000
*.....................................................
*Br    1 :Wavelength : Wavelength/F
*Entries :  5233340 : Total  Size=   21001253
*Baskets :      656 : Basket Size=      32000
*.....................................................
*Br    2 :OpChannel : OpChannel/I
*Entries :  5233340 : Total  Size=   21000593
*Baskets :      656 : Basket Size=      32000
*.....................................................
*Br    3 :Time      : Time/F
*Entries :  5233340 : Total  Size=   20997293
*Baskets :      656 : Basket Size=      32000
*.....................................................
root [2] pmtresponse->cd();AllPhotons->Draw("T
input_line_47:2:3: error: use of undeclared id
 (pmtresponse->cd())

Error in <HandleInterpreterException>: Error e
Execution of your code was aborted.
root [3] AllPhotons->Draw("Time")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [4] 
```

Next

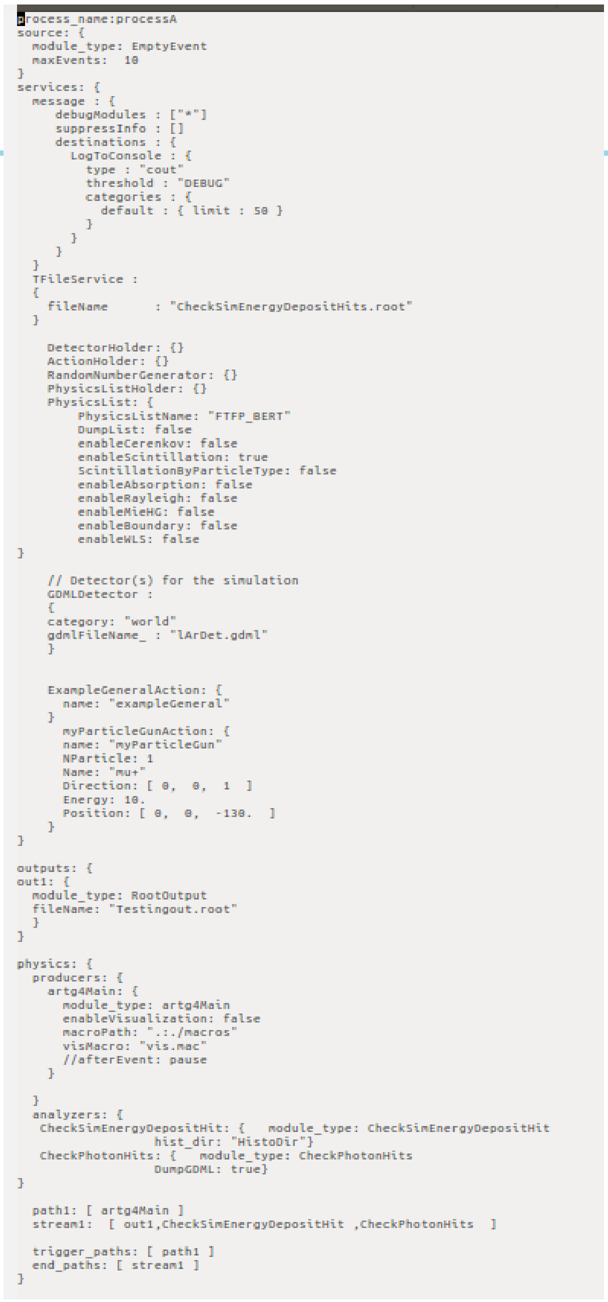
- started to implement some of the tasks (e.g. auxdetector) that were not part of phase one. But haven't committed it yet since I didn't want to get in the way of making a release.
- removing the last dependencies from the old code (most of it I just forgot to remove the include files.) and check if there is anything that needs to be implemented in the new system (MCDumper?).
- Prepare general gdml file that contains all optical properties for liquid Argon (absorption, wls, ri, particle dependend yields, rayleigh….) this can than just included in the gdml files.
- Figure out how to use splines when using gdml.
- Proper description of homgenous electric and magnetic fields.
- Replace material properties service by service that directly extracts all properties from gdml → more flexible, less error prone...

Backup

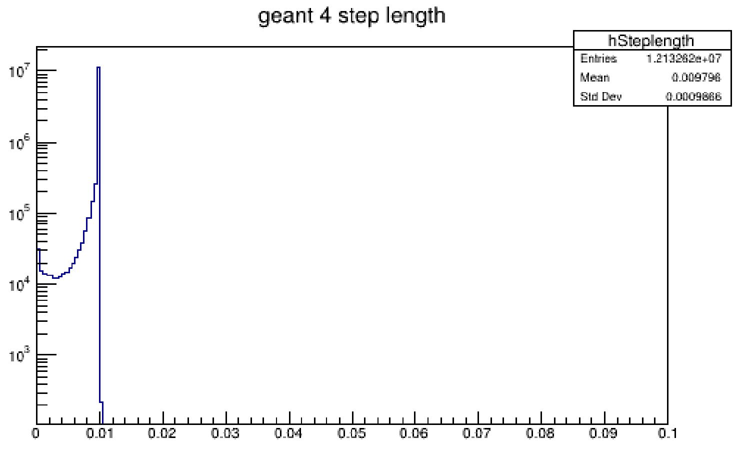
Physics

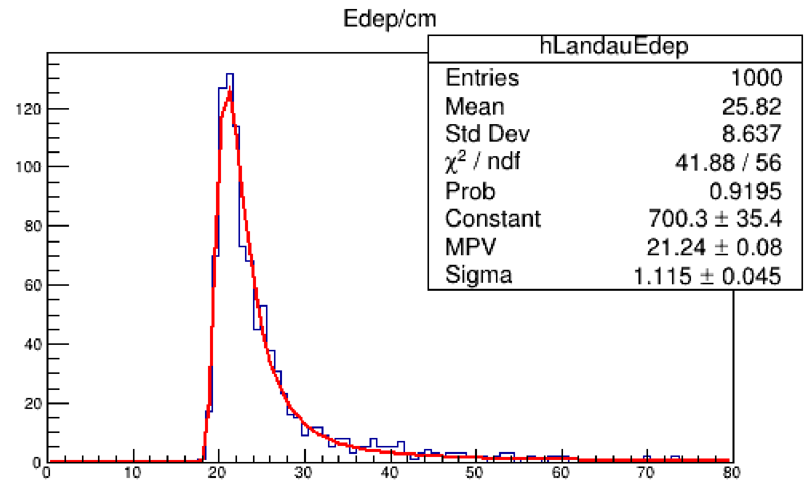
Use the new Geant4 physics list factory:

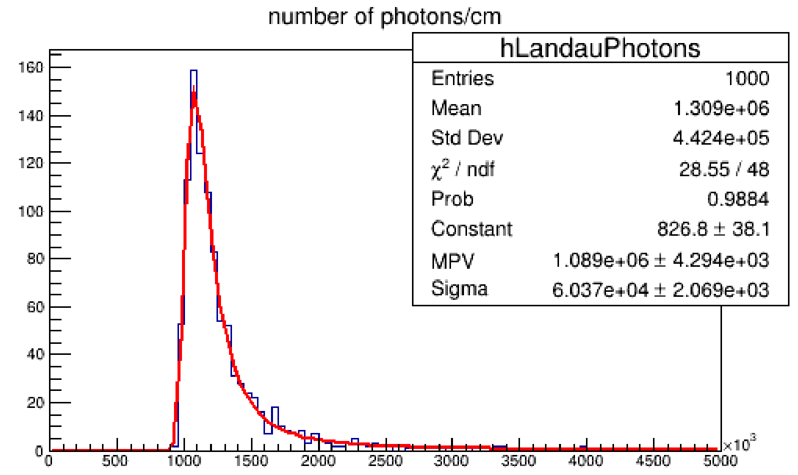
- Access to all reference physics list
- If you a brave you can register your own list
- Extendable to add all the available geant4 physics constructors e.g. we use:
  - Optical physics (complete: Rayleigh scattering, absorption, Cerenkov scintillation, boundary, surface, wls …..
  - Step limiter for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

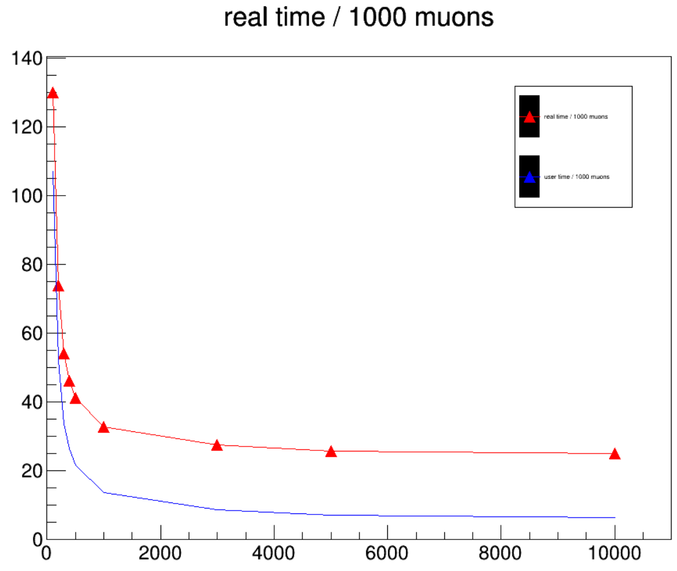


```
process_name:processA
source: {
  module_type: EmptyEvent
  maxEvents:  10
}
services: {
  message : {
     debugModules : ["*"]
     suppressInfo : []
     destinations : {
       LogToConsole : {
         type : "cout"
         threshold : "DEBUG"
         categories : {
           default : { 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 ]
}
```









step length

Fermilab

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