LArG4: What's the plan? Repeal and Replace?

Hans Wenzel

LarSoft coordination meeting

18th July 2017

Outline

- Strategy
- Replacing LArG4 components with components provided by Geant4.
- LarTest: what is it, some results.
- Plan.

Strategy

- Find out what we need and how it is done now.
- Establish good communication with Geant 4 collaboration and make use of local expertise here at Fermilab → make sure best performance with regards to memory, CPU, physics.
- Work with Geant4 collaboration and make sure that we have all the hooks and interfaces we need→ Replacing LArG4 components with components now provided by Geant4.
- Make LarTest part of the regular Geant4 profiling and performance testing procedure → directly benefit from performance improvements.

Replace

- G4OpticalPhysics (more than) replaces:
  - LArG4:
    - OpticalPhysics
    - OpFastScintillation
    - OpBoundaryProcessSimple

- G4PhysListFactoryAlt (more than) replaces:
  - LArG4:
    - ConfigurablePhysList
    - CustomPhysicsBuiltins
    - CustomPhysicsFactory

- G4SteplimiterPhysics replaces:
  - LArG4: voxel readout.

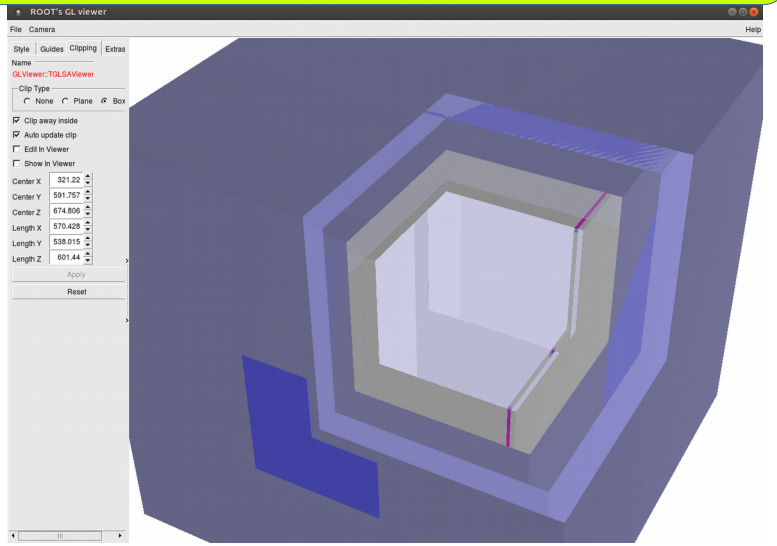
LarTest: stand alone Application to profile, test and measure the liquid Ar simulation.

Soon Yung Yun, Hans Wenzel

Uses: extend-able Physics List Factory (G4PhysListFactoryAlt), G4OpticalPhysics, G4StepLimiterPhysics, Profiling hooks, multi-threaded

Soon added all the hooks for profiling → work already paid off! see Soon's presentation at LarSoft WS. Profiling found ineffective access to material properties→ fix speeds up by 20%.

./lArTest protodune.gdml muons.in



Code: https://github.com/hanswenzel/lArTest

Available Reference Physics lists (at the moment)

The following physics lists are available:
"FTFP_BERT"
"FTFP_BERT_TRV"
"FTFP_BERT_ATL"
"FTFP_BERT_HP"
"FTFP_INCLXX"
"FTFP_INCLXX_HP"
"FTF_BIC"
"LBE"
"QBBC"
"QGSP_BERT"
"QGSP_BERT_HP"
"QGSP_BIC"
"QGSP_BIC_HP"
"QGSP_BIC_AllHP"
"QGSP_FTFP_BERT"
"QGSP_INCLXX"
"QGSP_INCLXX_HP"
"QGS_BIC"
"Shielding"
"ShieldingLEND"
"ShieldingM"
"NuBeam"

em options
"",
"_EMV"
"_EMX"
"_EMY"
"_EMZ"
"_LIV"
"_PEN"
"__GS"

Physics constructors:
- G4OpticalPhysics
- G4SteplimiterPhysics
- ...

Some timing results and summary

Configuration	Time for 10000 5GeV muons
Cerenkov on, Scintillation off, Stacking off (new)	5.14 [sec]
Cerenkov on, Scintillation off, Stacking on, photons killed in UserStackingAction (for long time the recommended way)	12.1 [sec]
Cerenkov on, Scintillation off, Stacking on, tracking	1 min 43.2 sec.

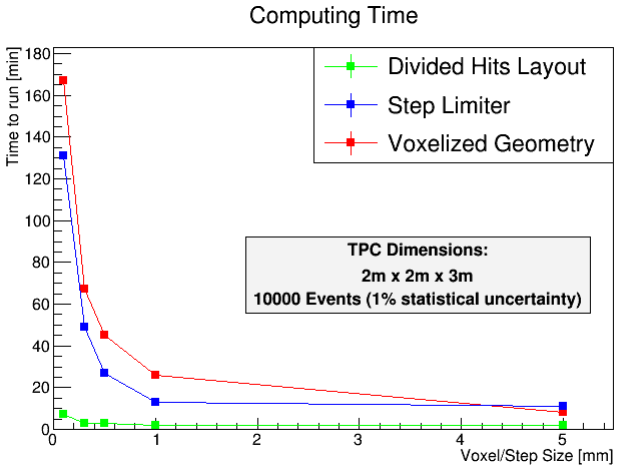
- Latest version of Geant4 10.3.p01 provides easy, flexible and efficient way to configure the Geant4 optical physics. Just what we asked for. No more ripping out of Geant4 functions necessary!
- Since then profiling has revealed that access to material properties (done at each step) was done in inefficient way → better implementation (Soon) results in 20% better CPU performance and gets rid of memory churn. For results see:

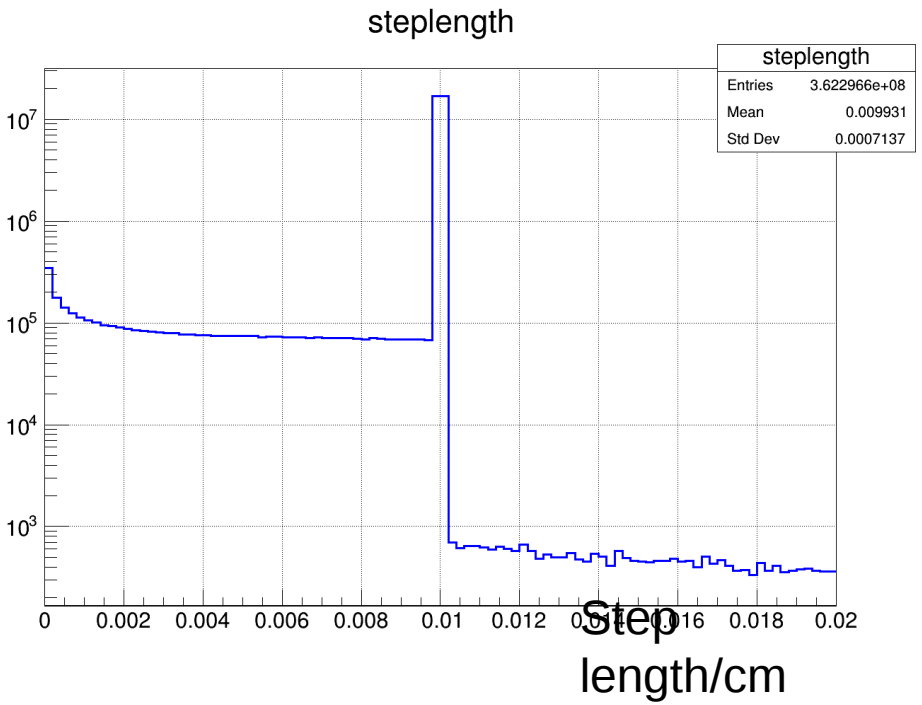
  https://g4cpt.fnal.gov/g4p/oss_10.3.r06_lArTest_01/compare.html

→ unfortunately to gain the 20% one needs to use a patched version of geant4.

Step limiter (limit 0.01cm) in action:

Here:
Step limiter only applied to charged particles. The longer steps are due to neutral particles (neutrons, γ's)





https://g4cpt.fnal.gov/g4p/oss_10.3.r06_lArTest_01/compare.html

Plan

- Short term:
  - Make sure LarSoft works with latest geant 4 version (10.3.p01).
  - Run LarTest from within art.
  - Define data product that registers (position,time,dE/dx, #photons at each step).
  - Make all the parameters accessible via fhicl.
  - Add electron recombination (box algo.)
  - Add drift of electrons to wire planes → wiresim.
  - Compare results of replacement with current LarSoft.
- Gdml:
  - Use to define optical properties.
  - Use (formulas) loops→can use dump to produce output compatible with e.g. root gdml viewer.
  - Extend to be able to attach various sensitive detectors to logical volumes (a la artg4tk).
  - Extend to add visualization hooks.
  - Define step limits?
  - Longer term:
    - Separate electron drift from simulation make it a separate module → drift to plane /surface e.g. for dual phase.

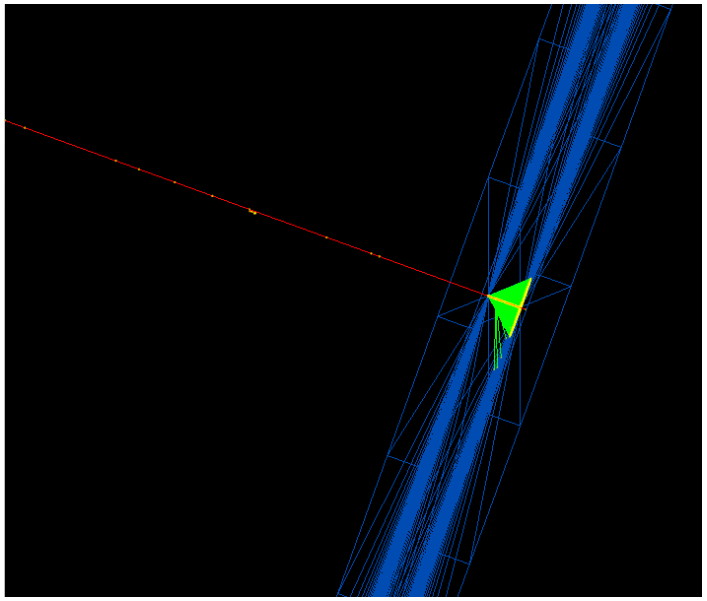
Backup

LArG4

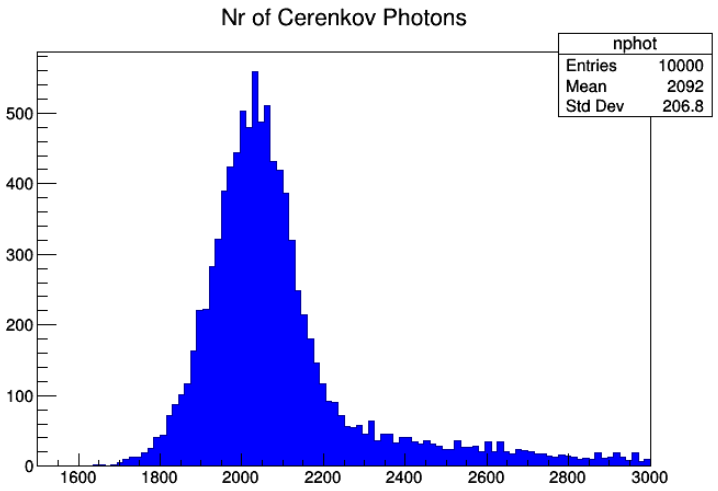
Breaks with what we said about the tasks of simulation:

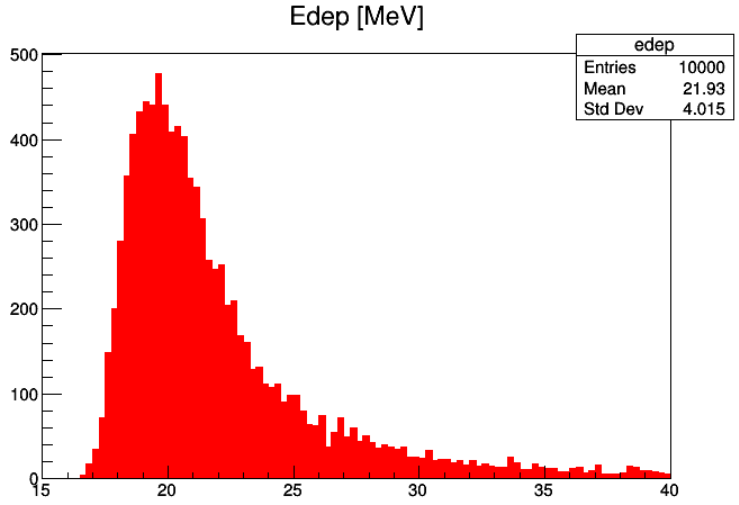
- Really specific to liquid Argon TPC.
- Deals with energy deposition and electric charge transport.
- Takes e.g. G4Scintillation out of geant 4 and changes it: OpFastScintillation
- Uses simplified non-geant 4 boundary process. The other boundary processes provided by Geant 4 are not available. Should work with Geant 4 collaboration if additional simplified boundary process is necessary → have not proposed that to geant4 yet.

Now that Geant4 provided the tools we asked for we can fix this!



6 cm thick Carbon disk,
with optical properties.



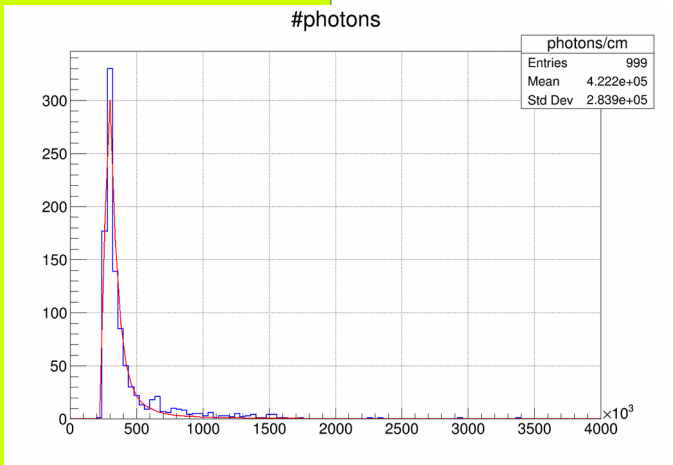


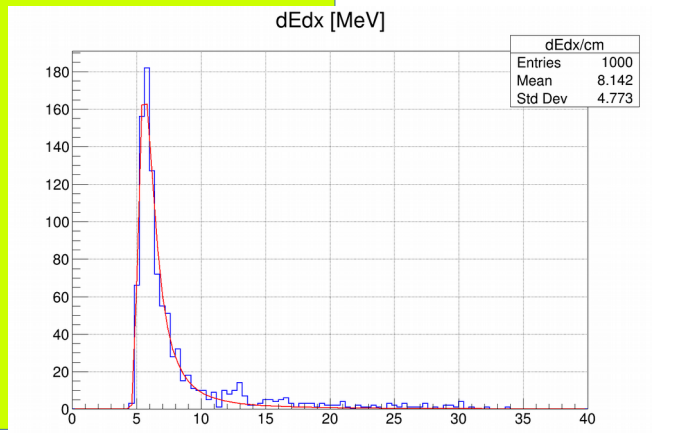
Fermilab

LArDataObj

```
#ifndef LARDATAOBJ_SIMULATION_SIMEDEP_H
#define LARDATAOBJ_SIMULATION_SIMEDEP_H
// C/C++ standard libraries
#include <string>
#include <vector>
namespace sim {
  struct SimEDep{
    double time;
    float xpos;
    float ypos;
    float zpos;
    float ds;
    float energy;
    Int NrofPhotons;
    int   trackID;
    int   pdgCode;
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<gdml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchem
aLocation="http://service-spi.web.cern.ch/service-spi/app/releases/GDML/schema/g
dml.xsd">
<materials>
 </materials>
   <solids>
      <box name="WorldBox" lunit="cm" x="500" y="500" z="1000"/>
      <box name="ArgonVolume"  lunit="cm" x="400" y="400" z="900"/>
   </solids>
   <structure>
      <volume name="volTPCActiveInner">
         <materialref ref="G4_lAr"/>
         <solidref ref="ArgonVolume"/>
      </volume>
      <volume name="TOP">
         <materialref ref="G4_AIR"/>
         <solidref ref="WorldBox"/>
         <physvol name="pCalorimeterVolume">
            <volumeref ref="volTPCActiveInner"/>
            <position name="Calpos" x="0" y="0" z="0"/>
         </physvol>
      </volume>
   </structure>
   <setup version="1.0" name="Default">
      <world ref="TOP"/>
   </setup>
</gdml>
```





Adding and configuring optical physics/step limiter (c++)

```
    G4PhysListFactory factory;
    G4VModularPhysicsList* phys = NULL;
    G4String physName = "";
   char* path = getenv("PHYSLIST");
    if (path) {
        physName = G4String(path);
    } else {
        physName = "FTFP_BERT"; // default
    }
    // reference PhysicsList via its name
    if (factory.IsReferencePhysList(physName)) {
        phys = factory.GetReferencePhysList(physName);
    }
    // now add optical physics constructor:
    G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
    phys->RegisterPhysics(opticalPhysics);
    // Cerenkov off by default
    opticalPhysics->Configure(kCerenkov, false);
    opticalPhysics->SetCerenkovStackPhotons(false);
    // Scintillation on by default, optical photons are not put on the stack
    opticalPhysics->Configure(kScintillation, true);
    opticalPhysics->SetScintillationYieldFactor(1.0);
    opticalPhysics->SetScintillationExcitationRatio(0.0);
    opticalPhysics->SetScintillationStackPhotons(false);
    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(100);
    opticalPhysics->SetMaxBetaChangePerStep(10.0);
    //StepLimiter:
    G4cout << ConfigurationManager::getInstance()->GetdoAnalysis() << G4endl;
    if (ConfigurationManager::getInstance()->GetstepLimit()) {
        G4cout << "step limiter enabled limit: " << ConfigurationManager::getInstance()->Getlimitval() * cm << " cm" << G4endl;
        phys->RegisterPhysics(new G4StepLimiterPhysics());
    }
    phys->DumpList();
```

Adding and configuring optical physics/step limiter

Use G4PhysList"Factory" and select one of the reference physics lists and em options.
Then register optical physics/ and step limit physics constructor:

```
G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
phys→RegisterPhysics(opticalPhysics);
phys->RegisterPhysics(new G4StepLimiterPhysics());
```

Then add optical properties and step limits to the material of interest when building the geometry:
e.g. for the step limiter:

```
G4double mxStep = ConfigurationManager::getInstance()->Getlimitval();
G4UserLimits *fStepLimit = new G4UserLimits(mxStep);
logicTarget->SetUserLimits(fStepLimit);
```

Alternatively use the new Factory developed by Robert Hatcher → will become standard

Detector simulation

Geant4: simulation of liquid Argon TPC is nothing special (besides matching steps to wire readout, having both ionization and scintillation) . User provides geometry and material properties (including optical) → spits out energy deposit, # of optical photons (if desired track to photo sensor,absorption) this info is added as data object to the event. Charge transport wire response etc. is passed to different module.

Simulation DONE

Responsibility of Geant 4 (or other provider of detector simulation) collaboration:

- This is done in efficient matter (10.3.p01).

Easy to configure (10.3.p01: G4OpticalPhysics to replace OpticalPhysics, Step limiter constructor)

- Access to all physics processes
- Physics is done correctly.
- In general provide all necessary interfaces (e.g. to all optical processes not just one selected one)
- Should be be more general not just lAr -TPC (+ auxdetector) specific there are TOF, Calorimeters, optical detectors …..
-

Optical Photon Processes in GEANT4

Optical photon production in Geant4:

- Cerenkov Process.
- Scintillation Process (Birks suppression can be particle dependent!).
- Transition Radiation.

Processes:

- Refraction and Reflection at medium boundaries.
- Bulk Absorption.
- Rayleigh/Mie scattering.
- Wavelength shifting

→ we want it all when doing full optical simulation enable with simple switch.

User has to provide optical properties of material as function of photon momentum (e.g. refraction index, absorption length, scattering length, surface properties, scintillation yield and spectrum, time constant.…)

Problem: until recently optical photons were always put on the stack → very expensive operation. If you didn't want to track them you had to kill them via e.g in the G4UserStackingAction. But often you just want to count the optical photons!
(e.g. when doing the parameterized optical response)

(LArG4) Requirements: We asked for it!

Worked with the Geant 4 collaboration to meet the requirements of the liquid Argon community. In geant4.10.3.01 all the proper interfaces and access methods are in place we now have:

- Convenient way to add and configure step-limiter process for selected volumes → e.g. to match step length to wire read out in liquid Ar.
- Convenient way to add optical physics.
  - configure and switch selected optical processes on/off.
  - Possibility to disable stacking but retain access to the number of produced photons
  - Use splines/functions to provide smooth optical property input (refraction index as function of photon momentum, scintillation spectrum…) → no more un-physical steps in distribution from optical processes.

Will show how this now can be done with G4OpticalPhysics physics constructor. Thanks to Peter Gumplinger for providing the optical code.

Other argument for upgrading better physics

Adding and configuring optical physics/step limiter (c++)

```
    G4PhysListFactory factory;
    G4VModularPhysicsList* phys = NULL;
    G4String physName = "";
   char* path = getenv("PHYSLIST");
    if (path) {
        physName = G4String(path);
    } else {
        physName = "FTFP_BERT"; // default
    }
    // reference PhysicsList via its name
    if (factory.IsReferencePhysList(physName)) {
        phys = factory.GetReferencePhysList(physName);
    }
    // now add optical physics constructor:
    G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
    phys->RegisterPhysics(opticalPhysics);
    // Cerenkov off by default
    opticalPhysics->Configure(kCerenkov, false);
    opticalPhysics->SetCerenkovStackPhotons(false);
    // Scintillation on by default, optical photons are not put on the stack
    opticalPhysics->Configure(kScintillation, true);
    opticalPhysics->SetScintillationYieldFactor(1.0);
    opticalPhysics->SetScintillationExcitationRatio(0.0);
    opticalPhysics->SetScintillationStackPhotons(false);
    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(100);
    opticalPhysics->SetMaxBetaChangePerStep(10.0);
    //StepLimiter:
    G4cout << ConfigurationManager::getInstance()->GetdoAnalysis() << G4endl;
    if (ConfigurationManager::getInstance()->GetstepLimit()) {
        G4cout << "step limiter enabled limit: " << ConfigurationManager::getInstance()->Getlimitval() * cm << " cm" << G4endl;
        phys->RegisterPhysics(new G4StepLimiterPhysics());
    }
    phys->DumpList();
```

Accessing the Nr. Of optical (Scintillation) photons produced in sensitive detector (TrackerSD)

```
G4bool TrackerSD::ProcessHits(G4Step* aStep,G4TouchableHistory*) {
    G4double edep = aStep->GetTotalEnergyDeposit();
    if (edep == 0.) return false;
    G4int photons = 0;
    G4SteppingManager* fpSteppingManager = G4EventManager::GetEventManager()
          ->GetTrackingManager()->GetSteppingManager();
    G4StepStatus stepStatus = fpSteppingManager->GetfStepStatus();
    if (stepStatus != fAtRestDoItProc) {
       G4ProcessVector* procPost = fpSteppingManager->GetfPostStepDoItVector();
       size_t MAXofPostStepLoops = fpSteppingManager->GetMAXofPostStepLoops();
       for (size_t i3 = 0; i3 < MAXofPostStepLoops; i3++) {
          if ((*procPost)[i3]->GetProcessName() == "Scintillation") {
             G4Scintillation* proc1 = (G4Scintillation*) (*procPost)[i3];
             photons += proc1->GetNumPhotons();
          }
       }
    }
    return true;
}
```