Geant4: why move to 10.3.p01?

Hans Wenzel

LarSoft coordination meeting

11th April 2017

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 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 (deexcitation..)

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 steplimit 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

Adding and configuring optical physics/setp 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/setp 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();
```

Available Reference Physics lists (at the moment)

```
// Physics List name defined via
// environmental variable
// 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"
```

```
// The following em options are available:
   //"",
   //"_EMV"
   //"_EMX"
   //"_EMY"
   //"_EMZ"
   //"_LIV"
   //"_PEN"
   //"__GS"
```

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;
}
```

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

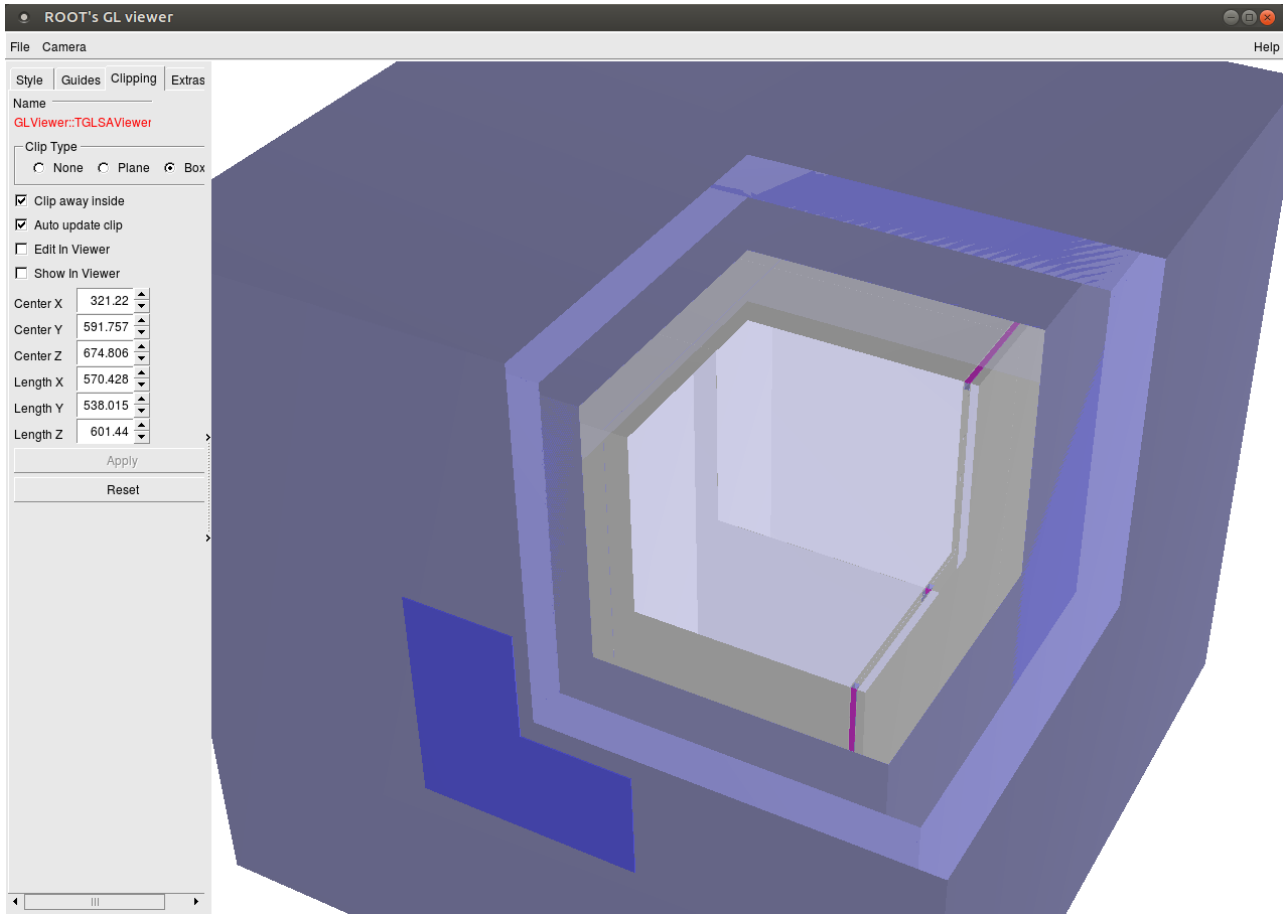
Soon Yung Yun, Hans Wenzel

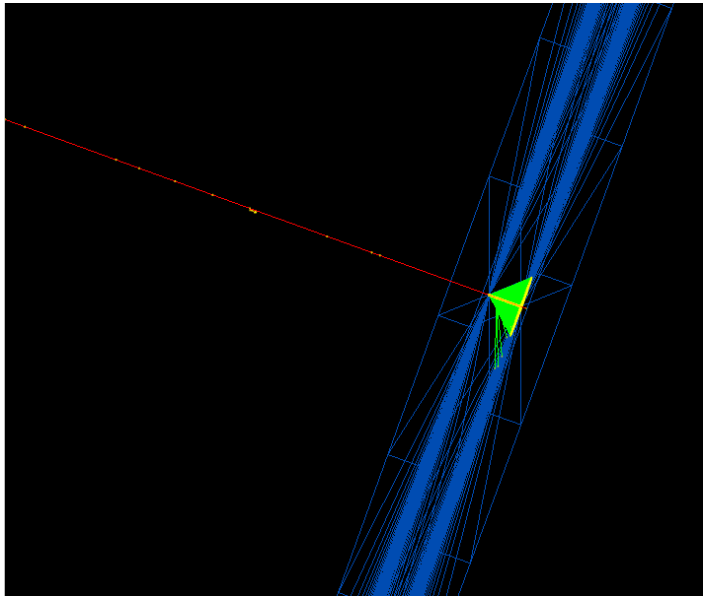
https://github.com/hanswenzel/lArTest

./lArTest protodune.gdml muons.in

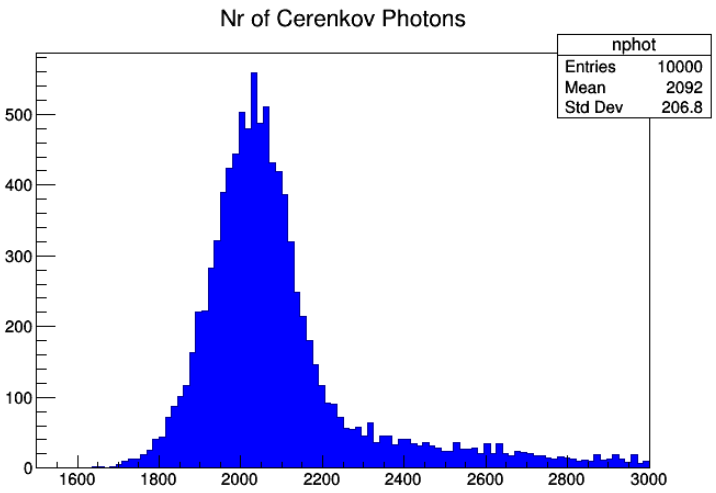
Soon added all the hooks for profiling and will talk about it in a future meeting.
Plan:

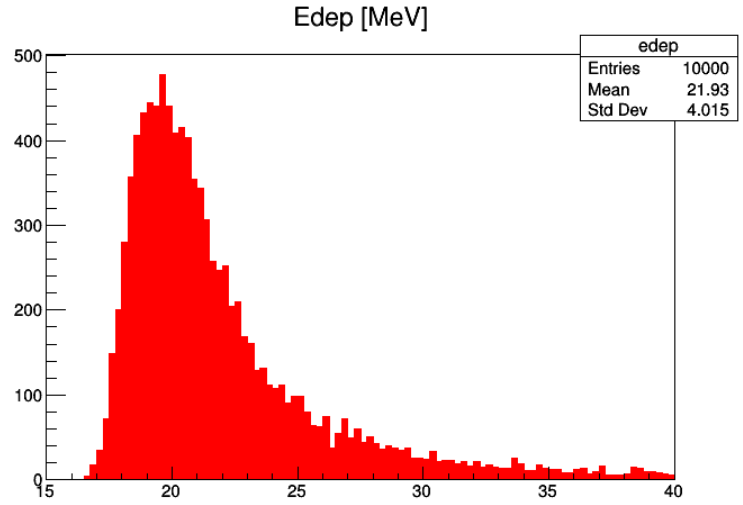
- Move all opt. Properties into gdml.
- Extend gdml to:
  - Assign step limits to volumes.
  - Assign sensitive detectors to volumes.





6 cm thick Carbon disk, with optical properties.

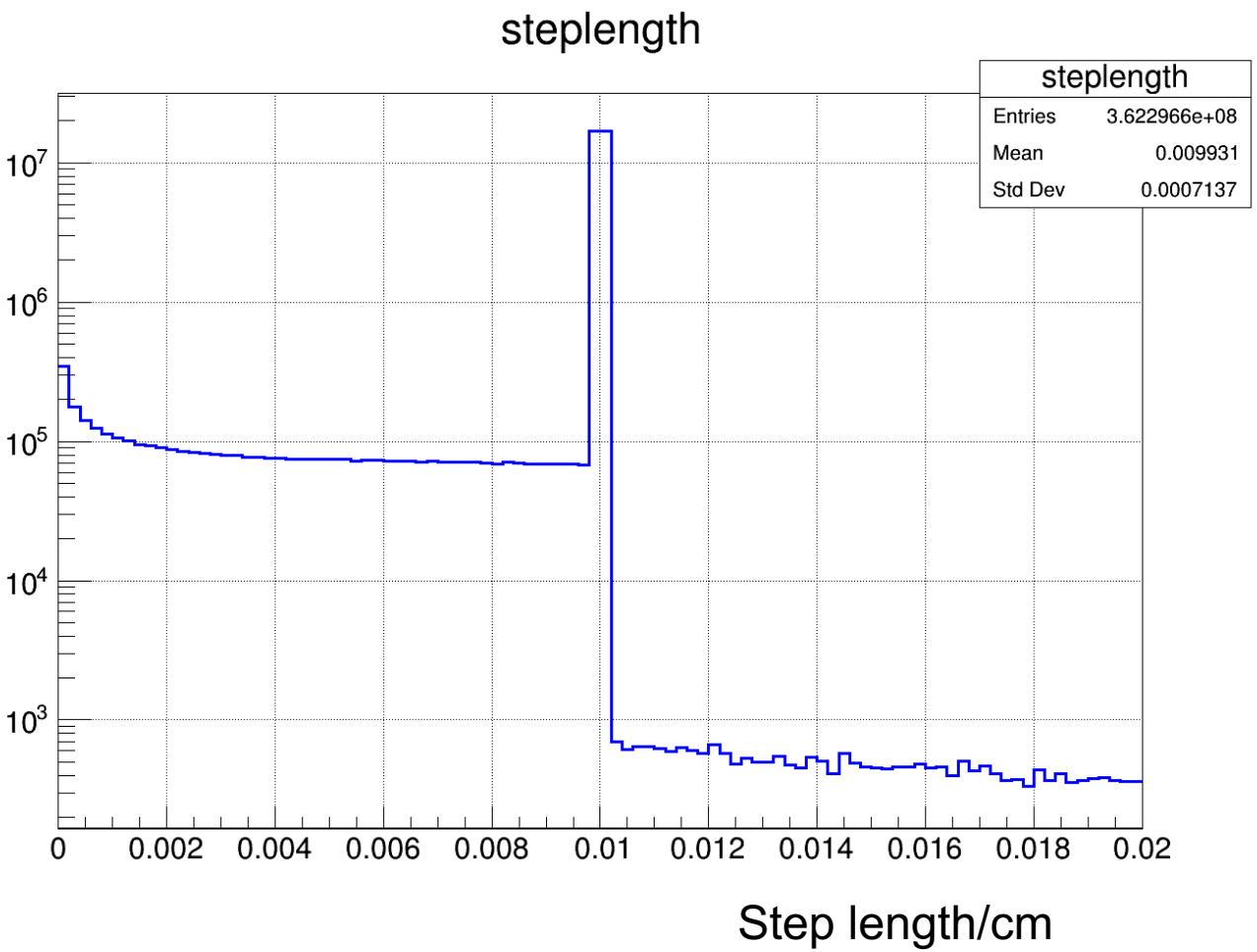




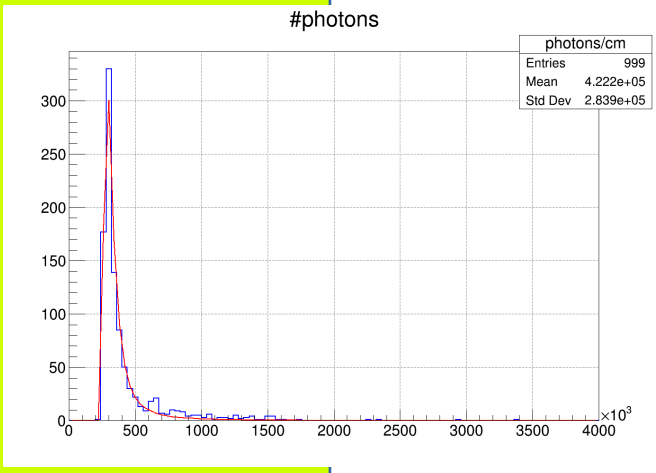
Fermilab

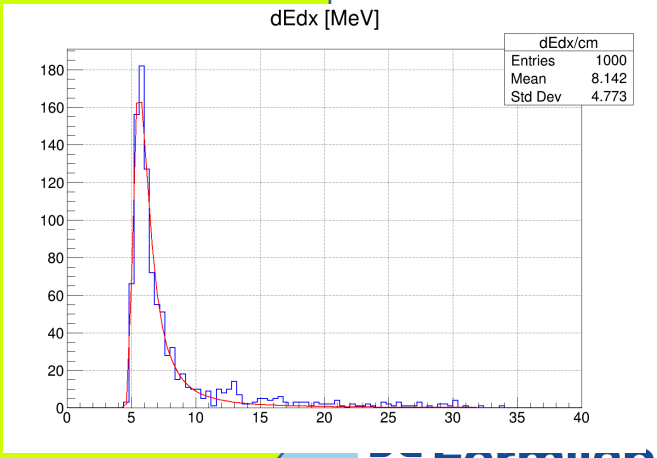
Step limiter (limit 0.01cm) in action:

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



```
<?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>
```





Conclusion

We got what we asked for → let's move!