Updates in optical physics

Daren Sawkey

Geant4 Collaboration Meeting
Sept 22, 2021

Summary

- Usability enhancements
  - Some changes to user code required
  - Some redundant functionality removed
- Small speedups
  - Not on scale helpful to FNAL
- Examples with detection (wls) and gdml (OpNovice) working

Usability

Use optical physics like this:

```
auto physicsList = new FTFP_BERT;
auto opticalPhysics = new G4OpticalPhysics();

auto opticalParams =
    G4OpticalParameters::Instance();
opticalParams->SetBoundaryInvokeSD(true);

physicsList->RegisterPhysics(opticalPhysics);
runManager->SetUserInitialization(physicsList);
```

Use pre-packaged physics list in most cases

User-defined parameters live here. Modelled after G4EmParameters

See the examples

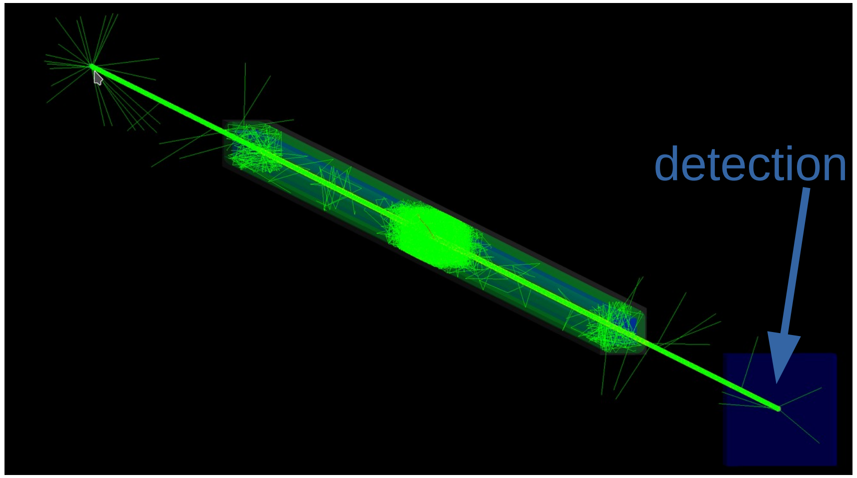
WLS example

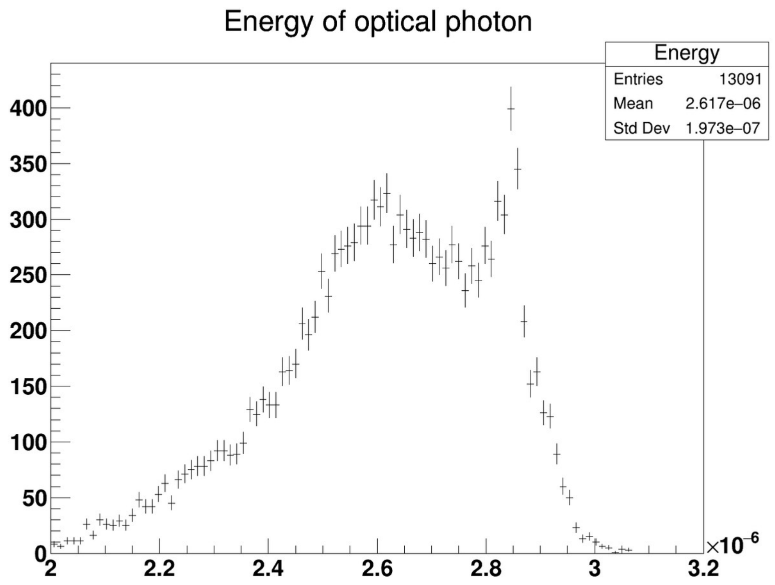
(wave length shifting)

Now it works!

Scintillator -> WLS fibre -> detector

- Added vis attributes
- Bug fixes
  - WLS fibre is built!
- Switched to G4OpBoundary::invokeSD
  - SD detects optical photons
  - So does UserSteppingAction
- Add some histograms

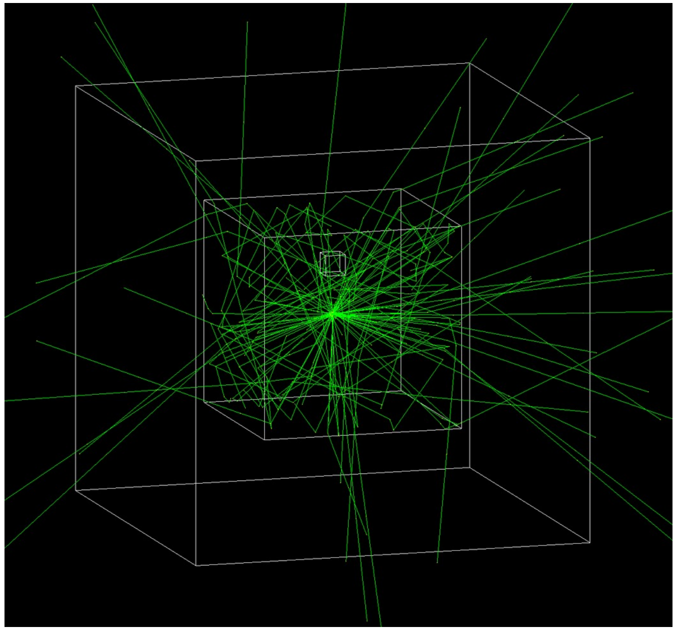




OpNovice example with GDML

- Option to read detector, including material properties, from GDML
  - (by Hans Wenzel)
  - Otherwise the same as regular OpNovice

```
<matrix coldim="2" name="REFLECTIVITY"
values="2.034*eV 0.3 4.136*eV 0.5"/>
```



Specifying material properties

- **Use vectors**
  - Run-time check that the vector of energies is the same length as the vector of values
- C arrays still work
  - But no protection against out-of-bounds reads

```
std::vector<G4double> energy = {
    2.00 * eV, 2.03 * eV, 2.06 * eV, 2.09 * eV, 2.12 * eV, 2.15 * eV, 2.18 * eV,
    ...
    3.26 * eV, 3.29 * eV, 3.32 * eV, 3.35 * eV, 3.38 * eV, 3.41 * eV, 3.44 * eV
  };

  std::vector<G4double> emissionFib = {
    0.05, 0.10, 0.30, 0.50, 0.75, 1.00, 1.50, 1.85, 2.30, 2.75,
    ...
    0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00
  };

  auto mptWLSfiber = new G4MaterialPropertiesTable();
  mptWLSfiber->AddProperty("WLSCOMPONENT", energy, emissionFib);
```

Scintillation material properties

Change required in user code

- The “enhanced” time constant properties of 10.7 is now the only method of specifying scintillation properties
  - Reduced need to build custom physics lists
  - 3 time constants, and particle-dependent yields with > 1 time constant
- Change to user code required if there is a material property with “FAST” or “SLOW” in its name
  - In most cases simply rename the material properties
    - FAST/SLOW -> 1/2/3
  - Documented in Book for Application Developers already in 10.7

Pre-defined optical material parameters

- Include MaterialProperties in the Geant4 distribution
  - Ideally users shouldn’t have to define their own properties for standard/uninteresting materials
- Use them as:

```
fiberProperty->AddProperty("RINDEX", "PMMA");
```

- So far, refractive indices for air, water, PMMA, fused silica
- Liquid argon to come (thanks to Hans Wenzel)
- Please, send your favourite values
  - Or add to materials/include/G4OpticalMaterialProperties.hh
  - Make sure the source license permits it

Creating new material property name

Change required in user code

- Previously, users had the “RIDNEX” problem
  - Thanks to Mike Kelsey for the name
- This compiles and runs fine but no Cerenkov photons are produced:

```
auto mpt = new G4MaterialPropertiesTable();
mpt->AddProperty("RIDNEX", energies, refractiveIndex);
```

- Now it is a run-time error
- If you do want to define a new property:

```
mpt->AddProperty("myProperty", energies, someValues, true);
```

Removal of some duplicated UI commands

Change required in user code

- If your command has “defaults” in it, remove “defaults”
  - `/process/optical/defaults/scintillation/setFiniteRiseTime`

    becomes

    `/process/optical/scintillation/setFiniteRiseTime`
  - `/process/optical/setTrackSecondariesFirst Cerenkov`

    becomes

    `/process/optical/cerenkov/setTrackSecondariesFirst`

    Same for scintillation

Both sets of commands exist in 10.7 and previous. Keep only 1 in 11.0

Scintillation by particle type

- Currently, need to specify absolute (not differential) yield for all particles
  - Electron, proton, deuteron, triton, alpha, ion
  - Often not interested or not known
- Proposal by Nate MacFadden to allow specifying a default differential yield for remaining particles
  - Seems fine; not yet implemented
- However scintillation by particle type still seems quite complicated.
  - Any ideas on how to improve it?

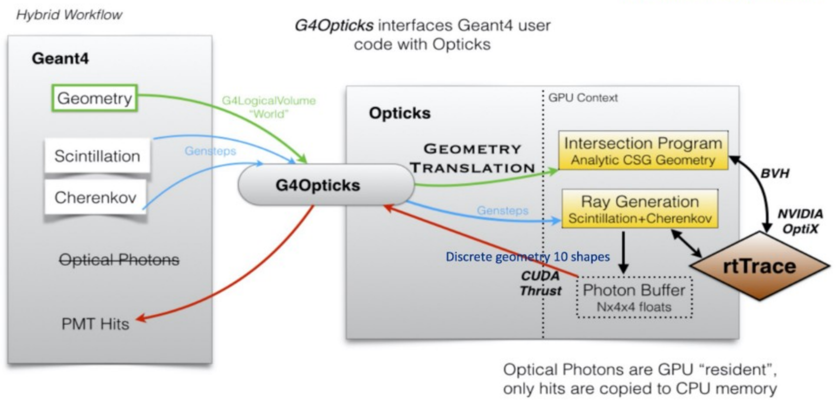
Bug fixes

- Protection against small Cerenkov steps not advancing primary (1992)
- Davis look-up table (G4OpBoundary) out of bounds read (2287)
- Scintillation not occuring for neutral particles (2372)
- Recalculate group velocity if RINDEX modified (2313)
- RealSurface data sets zlib-compressed (2241)
  - 800 MB to 127 MB

Speed-ups

- 100x speed-up wanted by experiments not going to come about by incremental improvement
- Need something else:
  - See e.g. H. Wenzel talk on Opticks integration (GPU)
  - https://indico.cern.ch/event/1052654/contributions/4525304/

Figure from Simon's presentation



Material properties map becomes vector

- Previously material properties stored as

  `std::map<G4int, G4MaterialPropertyVector*>`

  - Similarly for material constant properties
- In 11.0, move to `std::vector<G4MaterialPropertyVector*>`
  - Access value internally with vector.at(int) instead of map.find(int)
  - 1-3% speed improvement (OpNovice2/electron.mac with O3 optimization)
- **No changes for user**

Summary

- Usability enhancements
  - Some changes to user code required
  - Some redundant functionality removed
- Small speedups
  - Not on scale helpful to FNAL
- Examples with detection (wls) and gdml (OpNovice) working