Exploring Geant4 Performance in Optical Processes

Mentee: Felipe De Figueiredo (Long Island University)
Mentors: James Hirschauer, Hans-Joachim Wenzel (Fermilab)

Abstract

Abstract: Simulations of particle interactions with matter is one of the main tools used by particle physicists. Geant4 is a toolset used to perform these simulations, yielding much needed information regarding how particles interact with materials in detectors. One of the processes that occur when particles interact with materials is the generation of scintillation and Ĉerenkov light, resulting in optical photons with distinct properties. The amount of optical photons generated by these processes can be immense when performing these simulations. In this project, we explore the performance measurements between Multithreaded and Multiprocessed Geant4.

Table of Contents

1- What I've Learned:

- Geant4
    - What is it?
    - Build Geometry, Physics Lists
    - User Actions, Sensitive Detectors
    - Optical Physics
    - Analysis

2- Setting up Experiment:

- Geometry is given in the form a GDML(Geometry Detector Modeling Language (XML)) setup.
- Performance Measurements

3- Experiment Results:

- Experimental Validation
- Performance Results
- Acknowledgements

Geant4: Introduction

- Geant4 is a toolkit for the simulation of the passage of particles through matter.
    - Geant4 is a very flexible toolkit, allowing the creation of very simple to extremely complex applications.

- CaTS(Calorimetry and Tracker Simulation) is a framework based on Geant4. Used for example: Calorimetry and Tracking detectors.
    - Developed by Hans Wenzel

How to use it

- Geant4 is written in C++, to build applications you need to use CMake to create the makefiles and configure settings for the project

- You can build the project, and then run it like any other executable.

Build Geometry and Physics Lists

- To simulate particles going through materials, you'll need to construct the geometry and set up the physics lists

For the physics lists, you might use G4VModularPhysicsList class:

```
#pragma once
#include "G4VModularPhysicsList.hh"
#include "G4EmStandardPhysics.hh"
#include "G4OpticalPhysics.hh"

class TestPhysicsList : public G4VModularPhysicsList
{
public:
  TestPhysicsList();
  ~TestPhysicsList();

};
```

Then implement the specific physics lists desired:

```
TestPhysicsList::TestPhysicsList()
{

  RegisterPhysics (new G4EmStandardPhysics());
  RegisterPhysics (new G4OpticalPhysics());

}
```

Building Geometry: GDML Files

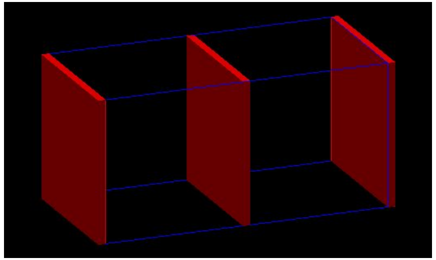
- You can build geometry by implementing the G4VUserDetectorConstruction class:
- GDML(Geometry Detector Modeling Language(XML)) Files are useful, allows you to change the geometry at runtime without having re-compile the project

C++

```cpp
class testDetector : public G4VUserDetectorConstruction
{

public:
        testDetector(const G4GDMLParser& parser);
        ~testDetector();

        virtual G4VPhysicalVolume *Construct();
        virtual void ConstructSDandField();

private:

        const G4GDMLParser& fParser;

};
```

GDML

```xml
<matrix name="RS" coldim="1" values="1.0" />
<matrix name="FTC" coldim="1" values="7.*ns" />
<matrix name="STC" coldim="1" values="1400.*ns" />
<matrix name="RY1" coldim="1" values="0.75" />
<matrix name="RY2" coldim="1" values="0.25" />
<matrix name="YR" coldim="1" values="0.8" />
<matrix name="SCINTLAR" coldim="2"
    values="7.74901*eV 0.000238409       7.79551*eV 0.000322081       7.842*eV 0.0004
<matrix name="RINDEXLAR" coldim="2"
    values="1.7712*eV 1.5 1.77634*eV 1.5 1.78148*eV 1.5 1.78662*eV 1.5 1.79176*eV 1.5 1
<matrix name="ABSLENGTHLAR" coldim="2"
    values="7.74901*eV 10.0*m 7.77525*eV 10.0*m 7.80167*eV 10.0*m 7.82827*eV 10.0*m 7.8
<matrix name="RAYLEIGHLAR" coldim="2"
    values="7.74901*eV 90.0*cm 7.77525*eV 90.0*cm 7.80167*eV 90.0*cm 7.82827*eV 90.0*cm
<rotation name="rPMTRotation1" unit="deg" x="90" y="270" z="0" />
<matrix coldim="2" name="REFLECTIVITY"
    values="2.034*eV 1 2.068*eV 1 2.103*eV 1 2.139*eV 1 2.177*eV 1 2.216*eV 1 2.256*eV
```



User Actions

- In the applications written so far, I've used mostly a stacking Action:

Stacking Action

- Implemented using the G4VUserStackingAction
- Whenever a new particle(Track) is created, the stacking action is called.

```
#include "stackingAction.h"

stackingAction::stackingAction() : G4UserStackingAction()
{
  currentEvent = 0;
  man = G4AnalysisManager::Instance();
}

stackingAction::~stackingAction()
{ }

G4ClassificationOfNewTrack stackingAction::ClassifyNewTrack(const G4Track* newTrack)
{

  G4ClassificationOfNewTrack classification = fWaiting;
```

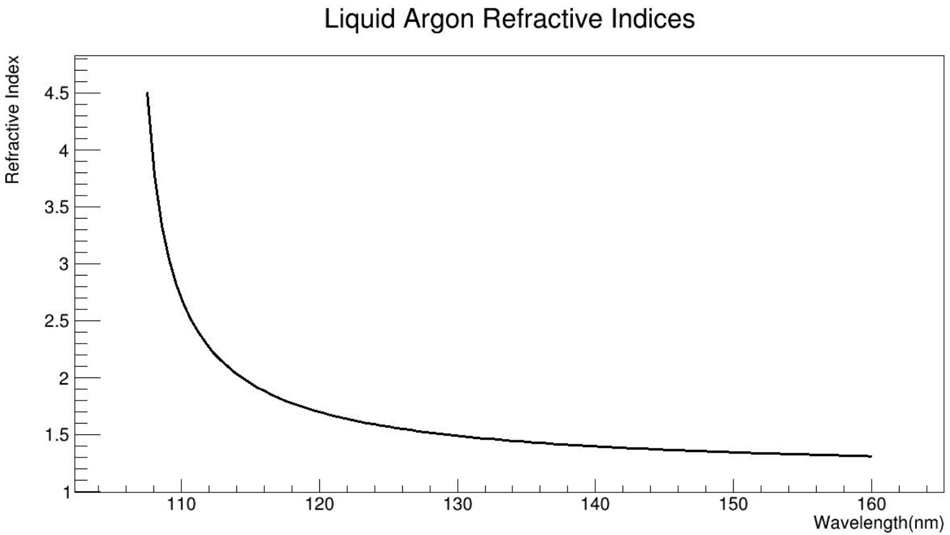
Optical Physics

- Optical properties are part of the material properties in the GDML file. They have to be provided by the user before Scintillation and Čerenkov processes can work.
  - **Indices of refraction**

Sellmeier Equation for LAr

$$n^2 = a_0 + \frac{a_{UV}\lambda^2}{\lambda^2 - \lambda_{UV}^2} + \frac{a_{IR}\lambda^2}{\lambda^2 - \lambda_{IR}^2}$$

Generate indices



Into GDML file

```
<matrix name="RINDEXLAR" coldim="2"
    values="7.74901*eV 1.31339 7.75673*eV 1.31368 7.76445*eV 1.31397 7.77217*eV
    7.94201*eV 1.32107 7.94973*eV 1.32139 7.95745*eV 1.32172 7.96517*eV 1.32204
    1.32958 8.14273*eV 1.32994 8.15045*eV 1.3303 8.15817*eV 1.33067 8.16589*eV
    8.33573*eV 1.33947 8.34345*eV 1.33987 8.35117*eV 1.34028 8.35889*eV 1.34069
    1.35014 8.53645*eV 1.3506 8.54417*eV 1.35105 8.55189*eV 1.35151 8.55961*eV
    8.72945*eV 1.36269 8.73717*eV 1.3632 8.74489*eV 1.36372 8.75261*eV 1.36424
    1.37641 8.93017*eV 1.377 8.93789*eV 1.37759 8.94561*eV 1.37819 8.95333*eV 1
    9.12317*eV 1.3928 9.13089*eV 1.39348 9.13861*eV 1.39416 9.14633*eV 1.39485
    1.41105 9.32389*eV 1.41184 9.33161*eV 1.41264 9.33933*eV 1.41344 9.34705*eV
    9.51689*eV 1.4333 9.52461*eV 1.43424 9.53233*eV 1.43518 9.54005*eV 1.43613
    1.45871 9.71761*eV 1.45983 9.72533*eV 1.46095 9.73305*eV 1.46208 9.74077*eV
    9.91061*eV 1.49061 9.91833*eV 1.49197 9.92605*eV 1.49335 9.93377*eV 1.49473
    1.52836 10.1113*eV 1.53005 10.119*eV 1.53176 10.1268*eV 1.53348 10.1345*eV
    10.3043*eV 1.57791 10.312*eV 1.58009 10.3198*eV 1.58228 10.3275*eV 1.5845
    10.505*eV 1.64282 10.5128*eV 1.64572 10.5205*eV 1.64866 10.5282*eV 1.65163
    10.7058*eV 1.73174 10.7135*eV 1.73583 10.7212*eV 1.73997 10.7289*eV 1.74418
    1.85553 10.9065*eV 1.86163 10.9142*eV 1.86785 10.9219*eV 1.87417 10.9296*eV
    2.06094 11.1072*eV 2.07141 11.1149*eV 2.08214 11.1226*eV 2.09314 11.1304*eV
    11.3002*eV 2.45554 11.3079*eV 2.47841 11.3156*eV 2.50217 11.3234*eV 2.52688
    11.5009*eV 3.65008 11.5086*eV 3.75265 11.5164*eV 3.86576 11.5241*eV 3.99131
```

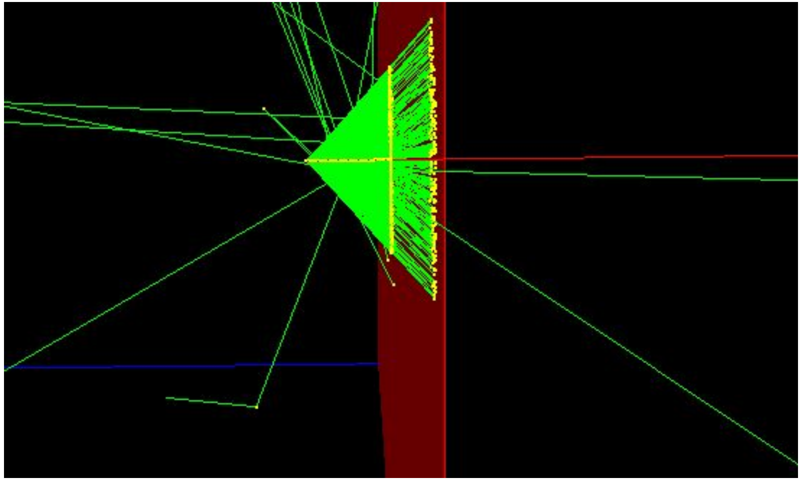
Additional Optical Properties That Have to be Provided

- Refraction index
- Absorption Lengths
- Rayleigh Scattering Lengths
- Reflectivity of Optical Surfaces
- Reflection Efficiency
- Emission Spectrum of Scintillating Material
- Scintillation Time Constants(Rise and Fall Time)
- Scintillation Yield (Optical Photons produced per MeV of deposited ionization energy)

Ĉerenkov Radiation

-When charged particles travel through a medium faster than the speed of light in that medium, they emit prompt radiation. This radiation is emitted in the shape of a cone, described by the equation:

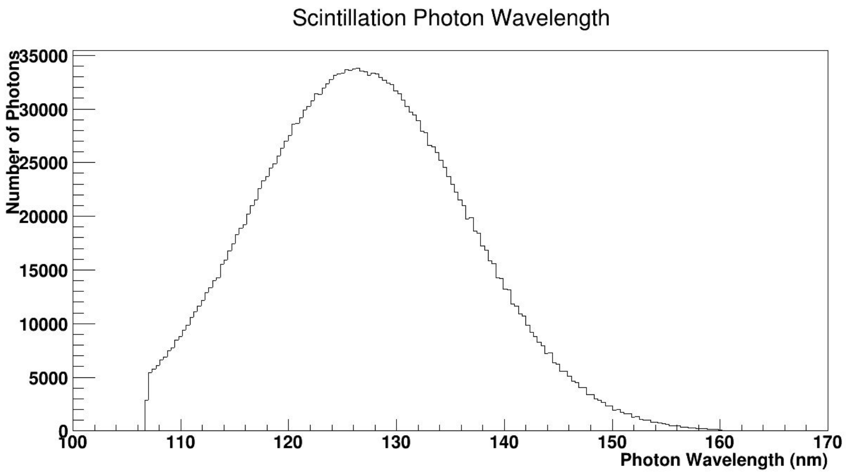
$$\cos\theta = \frac{1}{n\beta}$$



Scintillation

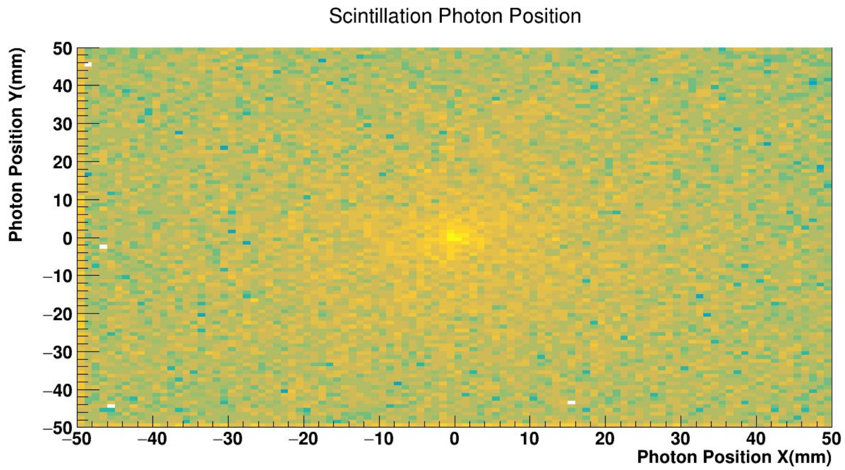
-Scintillation light is emitted by certain materials when transversed by charged ionizing particles.

-In Geant4, we see expected wavelength spectrums of photons emitted by scintillation:



-Since scintillation is an isotropic process, we expect it to be released from all different directions equally.

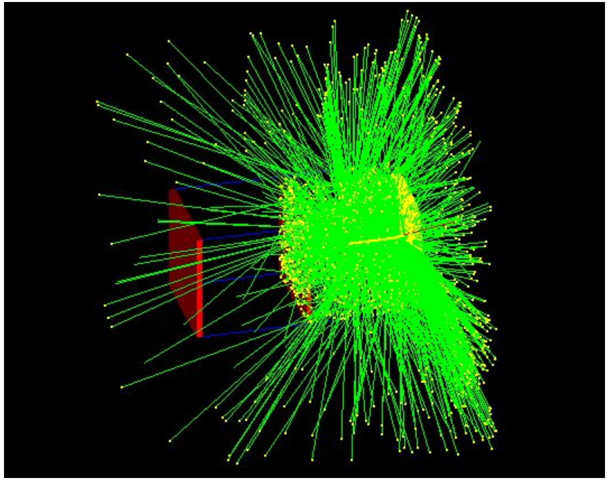
-Compared to Ĉerenkov light, it is not prompt



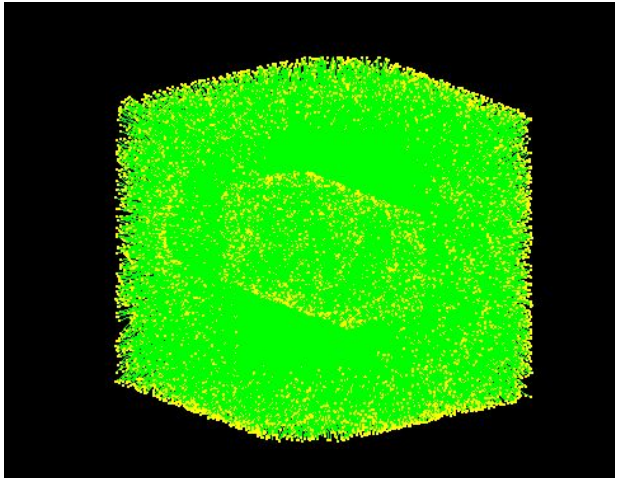
Scintillation Challenges

-Liquid Argon has scintillation yields of 50,000 photons emitted per 1 MeV of energy deposited in the material. An ionizing particle deposits 2 MeV per cm in LAr

-With high scintillation yields, each single event can take minutes to simulate.



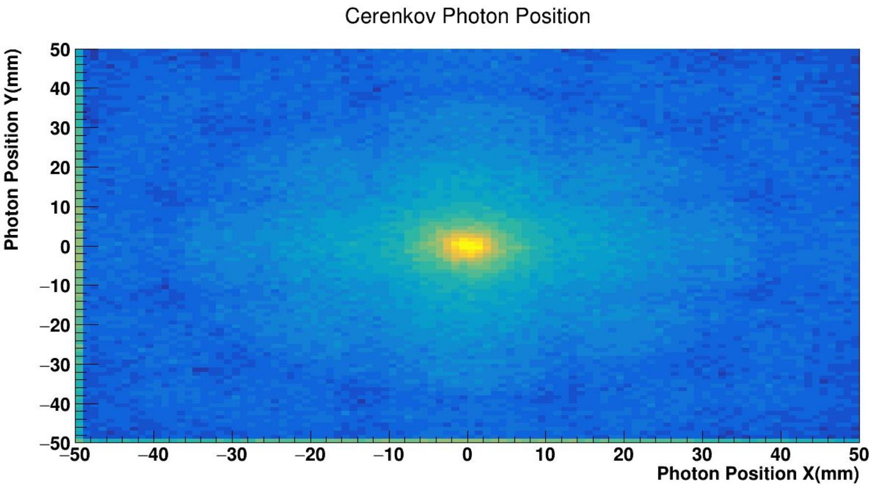
Scintillation Off

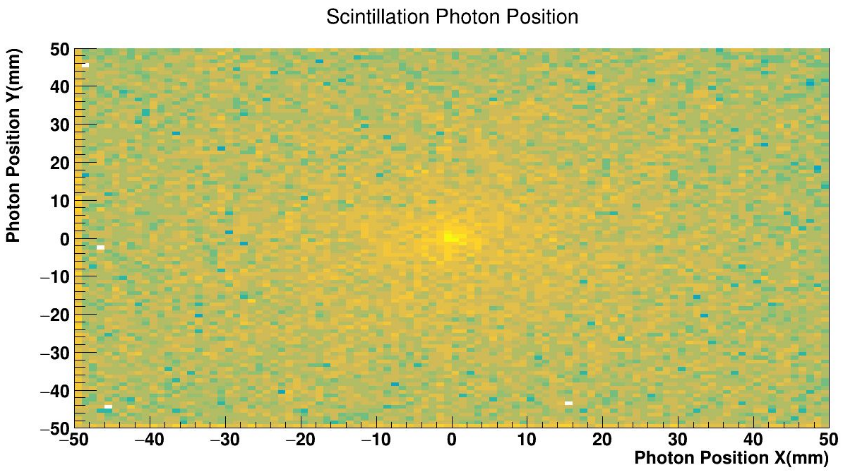


Scintillation On

Analysis

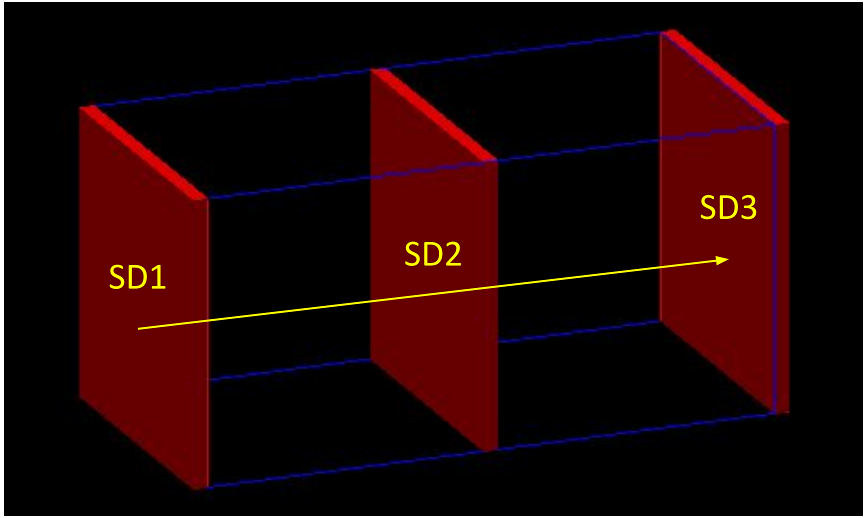
- Using G4AnalysisManager, you can make ROOT histograms and ntuples off of information registered at the various user actions:



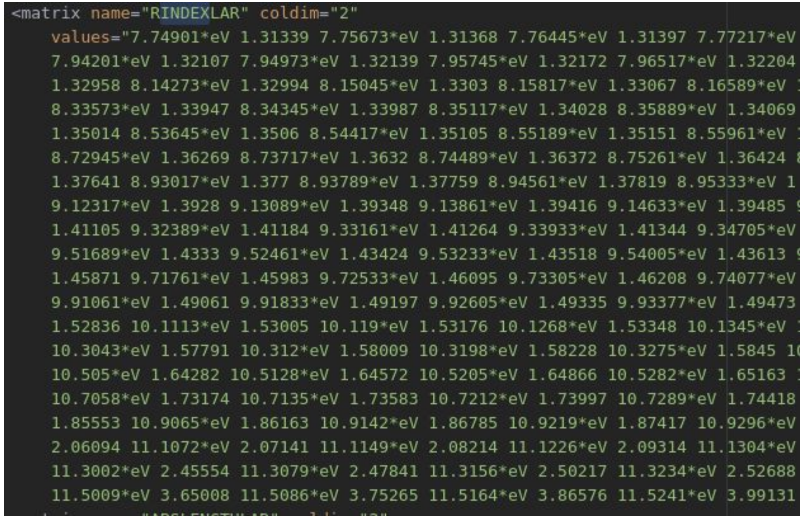


Experiment: Setting Up GDML File

- Here we see the geometry of the experiment, a box filled with Liquid Argon and three photodetectors inside the calorimeter cell(10x10x20mm):



- Using LAr sellmeier coefficients, we can generate the necessary indices of refraction:

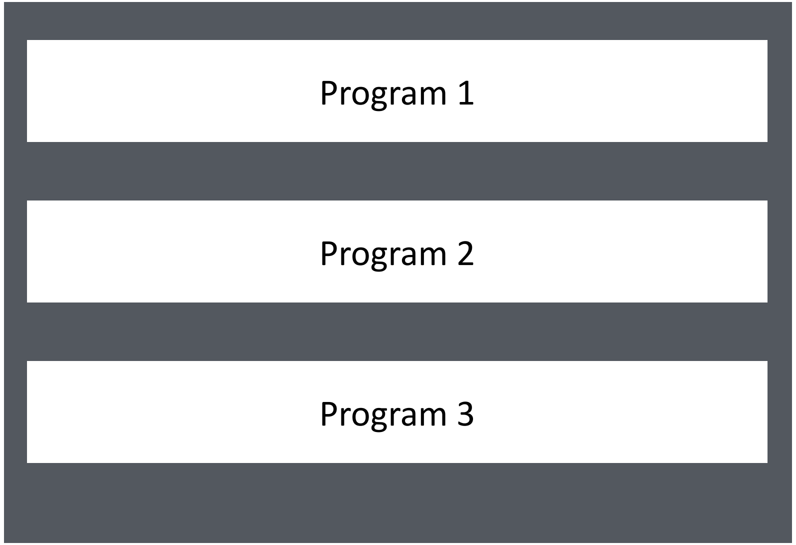


```
<matrix name="RINDEXLAR" coldim="2"
    values="7.74901*eV 1.31339 7.75673*eV 1.31368 7.76445*eV 1.31397 7.77217*eV
    7.94201*eV 1.32107 7.94973*eV 1.32139 7.95745*eV 1.32172 7.96517*eV 1.32204
    1.32958 8.14273*eV 1.32994 8.15045*eV 1.3303 8.15817*eV 1.33067 8.16589*eV
    8.33573*eV 1.33947 8.34345*eV 1.33987 8.35117*eV 1.34028 8.35889*eV 1.34069
    1.35014 8.53645*eV 1.3506 8.54417*eV 1.35105 8.55189*eV 1.35151 8.55961*eV
    8.72945*eV 1.36269 8.73717*eV 1.3632 8.74489*eV 1.36372 8.75261*eV 1.36424
    1.37641 8.93017*eV 1.377 8.93789*eV 1.37759 8.94561*eV 1.37819 8.95333*eV 1
    9.12317*eV 1.3928 9.13089*eV 1.39348 9.13861*eV 1.39416 9.14633*eV 1.39485
    1.41105 9.32389*eV 1.41184 9.33161*eV 1.41264 9.33933*eV 1.41344 9.34705*eV
    9.51689*eV 1.4333 9.52461*eV 1.43424 9.53233*eV 1.43518 9.54005*eV 1.43613
    1.45871 9.71761*eV 1.45983 9.72533*eV 1.46095 9.73305*eV 1.46208 9.74077*eV
    9.91061*eV 1.49061 9.91833*eV 1.49197 9.92605*eV 1.49335 9.93377*eV 1.49473
    1.52836 10.1113*eV 1.53005 10.119*eV 1.53176 10.1268*eV 1.53348 10.1345*eV
    10.3043*eV 1.57791 10.312*eV 1.58009 10.3198*eV 1.58228 10.3275*eV 1.5845 1
    10.505*eV 1.64282 10.5128*eV 1.64572 10.5205*eV 1.64866 10.5282*eV 1.65163
    10.7058*eV 1.73174 10.7135*eV 1.73583 10.7212*eV 1.73997 10.7289*eV 1.74418
    1.85553 10.9065*eV 1.86163 10.9142*eV 1.86785 10.9219*eV 1.87417 10.9296*eV
    2.06094 11.1072*eV 2.07141 11.1149*eV 2.08214 11.1226*eV 2.09314 11.1304*eV
    11.3002*eV 2.45554 11.3079*eV 2.47841 11.3156*eV 2.50217 11.3234*eV 2.52688
    11.5009*eV 3.65008 11.5086*eV 3.75265 11.5164*eV 3.86576 11.5241*eV 3.99131
```

Performance Measurement

With single threaded programs, one can use multiple processes to utilize more of its computing cores

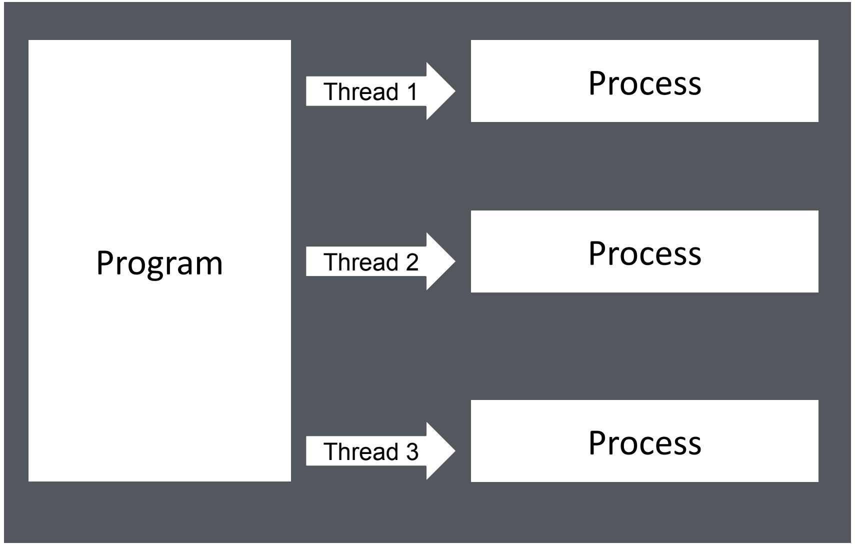
- Simply running multiple instances of the program at the same time, for example.



- With multithreaded programs, one can use multiple threads:

Master Thread

Worker Threads

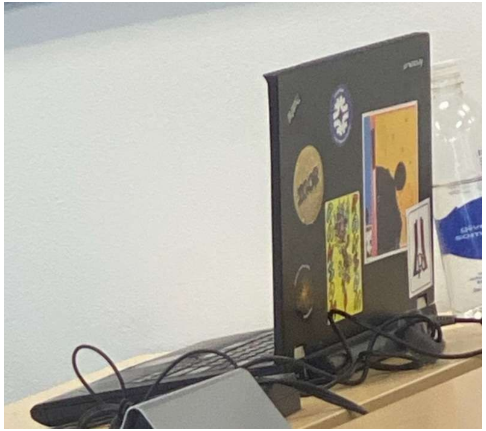


Performance Measurement Continued

- We will measure the number of events done per second.
  - Single threaded application
  - CaTS for multithreaded, with same GDML file

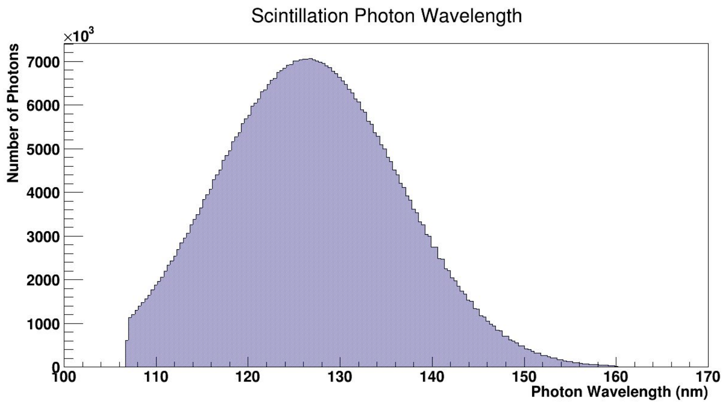
- We will also measure the amount of memory used by single threaded and multithreaded Geant4

- All this run in this machine:
  - 12Gb RAM
  - Intel(R) Core(TM) i5-2540M CPU @ 2.60GHz



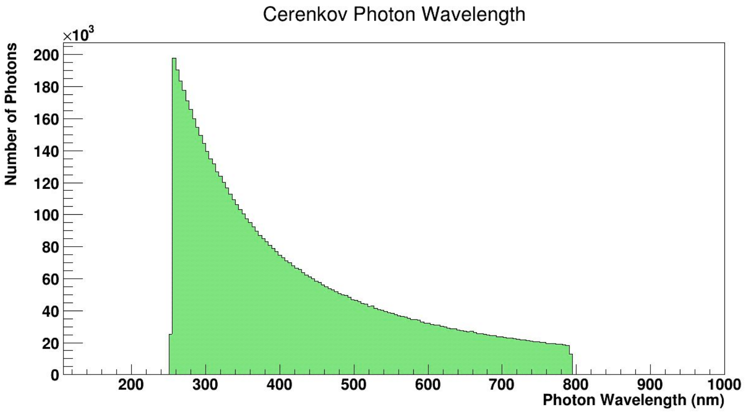
Experimental Validation

-To ensure the optical properties for the material are correct, data from the photons are recorded:



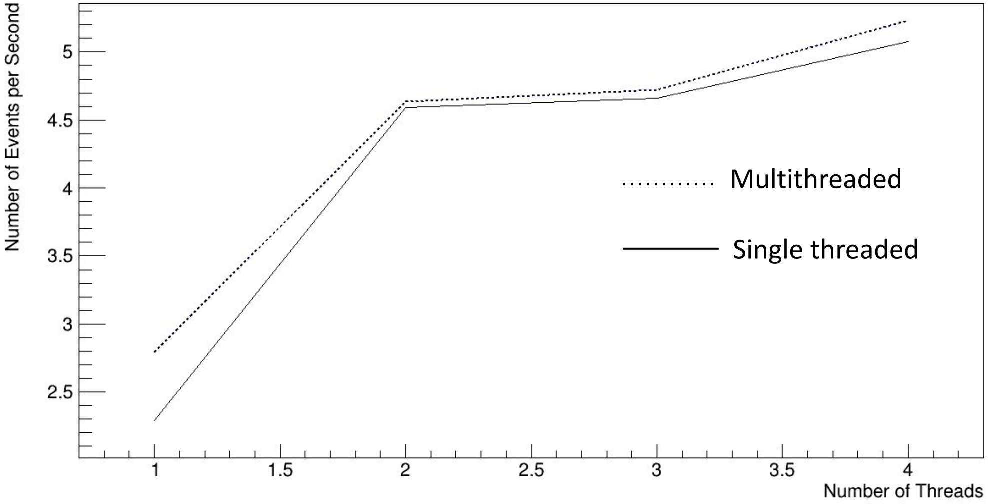
Wavelength of scintillation photons in LAr. *Photons are generated in the spectrum expected by LAr optical properties.*

-Same with Ĉerenkov:



Wavelength of Ĉerenkov photons in PbF2. *Photons are generated in the spectrum defined by the refractive indices of PbF2.*

Performance Results



- CPU Usage scales in the same way.

- In addition, we observe the memory increase in the multithreaded case is much lower than running multiple single threaded processes.
  - *Allowing you to use all CPU Cores when memory is sparse.*

Conclusion

- What did I learn?
  - Learned how to build and run Geant4 applications, a program used by particle physics and many other domains.
  - Learned about the Optical Processes in Geant4.
  - Learned how to analyze data generated by Geant4.
  - Would be able to apply this knowledge in a future projects requiring simulations.

Acknowledgements

- Thanks for my mentors: Dr. James Hirschauer and Dr. Hans-Joachim Wenzel
- Thanks for the Geant4 collaboration for making documentation very accessible!
- Thanks for Hans for making CaTS!
- Thanks for all PURSUE interns for helping me along the way!

Bibliography

1- *The Geant 4 Collaboration (Official Webpage)* https://geant4.web.cern.ch/

2- *CaTS Framework*, *Hans-Joachim Wenzel*
https://github.com/hanswenzel/CaTS

3- *Peculiarities in the Simulation of Optical Physics in Geant4, Erik Dietz-Laursonn, arXiv:1612.05162v1*

4- *Index of refraction, Rayleigh scattering length, and Sellmeier coefficients in solid and liquid argon and xenon, Emily Grace, arXiv:1502.04213v4*

5- Scintillating properties of today available lead tungstate crystals, M. Follin,V. Sharyy, J-P. Bard, M. Korzhik, D. Yvon, arXiv:2103.13106v2