INTRODUCTION AND APPLICATION OF GEANT4: A COMPARISON OF DEPTH-DOSE

A THESIS IN
Physics

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
CHRISTIAN XAVIER BROCK

B.S., University of Missouri-Kansas City, 2017

Kansas City, Missouri
2019

INTRODUCTION AND APPLICATION OF GEANT4:

A COMPARISON OF DEPTH-DOSE

Christian Xavier Brock, Candidate for the Master of Science

University of Missouri-Kansas City, 2019

ABSTRACT

Geometry ANd Tracking 4 (GEANT4) is an object-oriented C++ based program that uses Monte Carlo methods to simulate the passage of particles through matter. A powerful application, GEANT4, allows the user to define variables such as material geometry, particle tracking, and detector physics. The versatility of GEANT4 has led it to be used by a diverse group of physicists in a variety of physics fields. This paper will act as a basic introduction and user guide for the first time user. In order to check the user's ability to use GEANT4 reliably, this paper will attempt to replicate select geometries from the previously published results of Carrier in his paper "Validation of GEANT4, an object-oriented Monte Carlo Toolkit, for simulation in medical physics".

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the College of Arts and Sciences have examined a thesis titled "Application of GEANT4 to electrodynamics and medical physics," presented by Christian Xavier Brock, candidate for the Master of Science degree, and certify that in their opinion it is worthy of acceptance

CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGMENTS

I would like first to thank my graduate adviser Zhu for all the work he has put into helping me write this thesis. With out the motivation he provided and effort he put into guiding me through the process, this thesis would have never happened. I would like to thank him for the extra push or two he provided when I needed it. Next I would like the entire staff of the UMKC physics department. Without there patience, guidance, and kindness, I would not be the man or scientist I am today. I am grateful to: Fred, for his wisdom and willingness to always put the students first, Libby, for teaching me to take pride in my work and for being there for the start of both my Undergrad and Graduate degree, Paul, for instilling in me quality communication skills and for his friendly approachable demeanor, again to Zhu, for teaching when to ask good question and all the laughs, Caruso, for bridging the world of theoretical thought with hands-on skills and for instilling a level of professionalism in me that I never thought I would have, And lastly, but certainly not least Daphne for all the work she does for the department. I will forever be grateful to you all and always look back fondly on my experience at UMKC.

Secondly, I would like to thank the high-school teachers that allowed me to find my passion, Mr.Bishop, and Mrs.Forsythe.

And to my parents for supporting me and for pushing me to be the best I can be. With out there relentless pursuit to better the lives of there children, I would not be here.

Lastly, to my Fiancee Aman, for all her love, support, and patience during my education. And, for all the laughs that have allowed me to keep going.

CHAPTER 1

INTRODUCTION

Use of GEANT4

Geometry ANd Tracking 4th edition (GEANT4), is an object-orientated C++ toolkit created by CERN for the simulation of particles as they interact with different types of fields and matter. GEANT4 is a versatile and powerful Monte Carlo Based toolkit that can be highly customized by the user to meet their specific needs. By using GEANT4, the user can build there own 'virtual world' in which they can carry out their experiments. With the ability to define variables such as target geometry, particle gun physics, materials, etc., the capabilities are quite vast.

GEANT4 uses "an abundant set of physics models to handle the interactions of particles with matter across a very wide energy range." [1] For example, in GEANT4's low-energy simulations, physical processes such as Rayleigh effect, the photoelectric effect, the Compton effect, pair production, the Auger effect, ionization, bremsstrahlung, positron annihilation, atomic relaxation, and multiple scattering are used, with the limitation that particles can be tracked down to 250 eV. [2]

After the installation and building of GEANT4, the user will find many preinstalled examples that are 'ready to go.' Some of these preinstalled examples include electromagnetism, exotic physics, fields, hadronic, medical, radioactive decay. This list is by no means the limits of what GEANT4 is capable of but serves only as a brief illustration of the diversity of GEANT4 applications. Users are free to use the examples as they are, or they can edit the source code, allowing the example to be tailored to meet their individual and specific needs. If, however, there exists no example that closely resembles the desired application, the user can choose to build a new one from scratch using the tools that GEANT4 provides.

Given the proven flexibility and usefulness of GEANT4, I believe that the development of a skilled GEANT4 research group would be a valuable addition to any physics department. Therefore the goal of this thesis is to demonstrate an understanding of GEANT4. However, the apparent size and complexity of GEANT4 to a first time user, such as myself, can be a bit overwhelming. Nonetheless, I intend for this thesis to act as an introduction and reference guide for the new user in the hope that by the end, the new user will be able to run and alter examples within GEANT4 reliably. Leaving them ready to use GEANT4 for their own research goals.

To help the new user ease into GEANT4 and gain both familiarity and confidence with the basics, we first learn about what GEANT4 is and what it can do as well as what the current literature is using it for. We then explore its use by narrowing the scope of use to just one GEANT4 example. In order to use these examples as a way of building familiarity and confidence with GEANT4, we try to mimic the geometry and results of a previously published work by Carrier. [2]

Organization of this Thesis

As stated in the previous section, it is my goal that this thesis will serve as an introduction and reference guide for the new GEANT4 user. As such, a discussion of GEANT4's history, general capabilities, and some of the current literature will be laid out in Chapter Two. Chapter Three will attempt to show the user the structure of GEANT4. Ideally, Chapter Three will better prepare the new user to understand how examples run from start to finish so that they are more capable of running and editing examples. I discuss my installation and use of GEANT4 in Chapter Four. Chapter Five will contain my procedure and results.

Lastly, future work and conclusion will be in Chapter Six.

CHAPTER 2

GEANT4 PAST AND PRESENT

History of GEANT4

As the experiments carried out by physicists became more complex and more ambitious, the time and resources needed to test and carry out new ideas increased at an extraordinary rate. Thankfully, the digital age allowed us to use computers to run complex simulations, cutting down the time and resources needed for the great feats of the modern world. However, we quickly reached a point where a single project would have to run multiple programs, many of which were interdependent, this caused a slowing down of progress once again. Enter GEANT. Initially created in 1978 to help condense the growing number of programs needed in the new modern simulations of particle-mater interaction, down to one concise, user-friendly program.[3] Eventually, GEANT was taken over by CERN, as it had become the popular choice for High Energy Physics (HEP), being used by projects such as ALEPH, L3, OPAL. All of which used its initial FORTRAN based code. GEANT's FORTRAN based code would see three revisions, ending with GEANT3.

The end of the FORTRAN error was due to CERN's new project started in 1994. GEANT4, the fourth rendition of the GEANT lineup that would break away from its former FORTRAN based platform. Instead, GEANT4 was rewritten entirely using a C++ object-based platform. GEANT4 saw its first public release back in 1998, and ever since has been receiving regular updates, expansions, and maintenance. Using "object-oriented methods help manage complexity and limit dependencies by defining a uniform interface and common organizational principles for all physics models. Within this framework the functionality of models can be more easily recognized and understood, and the creation and addition of new models is a well-defined procedure that entails little or no modification to the existing code". [1]

With GEANT4, users can create a unique 'virtual reality' that can be used to develop, model, and research a detector before it is built and used. [4] We can simulate any number of events within this world to develop an understanding of how our system would behave in the real world. Due to the user-friendly focus and toolkit nature of GEANT4, we now see GEANT4 assisting with the development of fields outside of HEP, having expanded to fields such as optics, radioactive decay, medical physics, and much more.

Current Work in GEANT4

GEANT4 can be applied to a wide variety of disciplines; as such, discussing all the applications is beyond the scope of this paper. Instead, we will focus briefly on studies that the use of GEANT4 to study dose with a particular emphasis on radiation damage.

Mavragani cites GEANT4 and GEANT4-DNA as being some of the "Most widely used radiation transport and track-structure Monte Carlo (MC) codes for the simulation of radiation-induced damage." [5] They used GEANT4 to "[assess] the energy deposit and specific energy in various cell compartments." [5] Specifically, they were looking at complex DNA damage, a common field of study using GEANT4-DNA. Similarly, in Francis et al. "Monte-Carlo simulations were carried out using Geant4 with the Livermore and the Geant4-DNA processes to assess the effects of gold nanoparticles as radiation enhancers in a water like medium." [6] And again, Shamshiri et al., looked at "the direct effect of monoenergetic protons and alpha particles on DNA molecules, as the biological endpoint, ... using the Geant4-DNA extensions of the Geant4 Toolkit." [7] In their paper, they compared different DNA geometries, and the number of Direct Single-Strand Breaks (SSB), Double-Strand Breaks (DSB), and Total-Strand Breaks (TSB) in the DNA.

Along the same lines, Wu used GEANT4 to “prepare a simple model of a spacecraft in the Geant4 Monte Carlo toolkit and used this model to calculate the dose depth distribution of four typical heavy ions that would pass through the spacecraft.” [8] And lastly, Yano used GEANT4 to investigate “the dosimetric effect of magnetic fields in the treatment of lung tumors”, finding correlations between the magnetic field orientation and the dose at water-lung vs. lung-water interfaces.” [9]

As we see, some exciting things are being done with GEANT4, particularly regarding radiation therapy/damage. Thanks to the power of the GEANT4(-DNA) toolkit, studying radiation damage has become safer and arguably more ethical, opening up the possibility for new discoveries or refined medical techniques. However, according to Mavragani, both GEANT4 and GEANT4-DNA are “Complex toolkits, [that are] computationally intensive [and] require users with advanced programming skills.”[5]

Given my level of coding skills and my time constraints, I could not contribute to the study of DNA damage and radiation therapy using GEANT4 , but, I do believe that my physics department will have students more than capable of doing so. I also believe that GEANT4 can be of use to students who are interested in medical physics. So to help the future users gain familiarity with GEANT4 so that they can explore this new field, I have built this thesis as an introduction/user guide to GEANT4 with a focus on depth-dose.

CHAPTER 3

OVERVIEW OF GEANT4

How GEANT4 Works

In an attempt to better understand what it is the user can do with GEANT4, not only when running an application, but when building an application, we will layout the basic structure and procedure that GEANT4 follows to build and run an example. We first concern ourselves with the main steps the program takes in general when we run an experiment. I will then later attempt to contrast these steps with a specific example in Chapter Five.

A typical GEANT4 experiment has two main steps: Initialization and Run. In the Initialization step, the user can set the environment parameters. Initialization is where GEANT4 builds the virtual world and sets the rules and initial conditions. When GEANT4 is in initialization, it is undergoing the construction of: our world volume, objects in the world, materials and material properties, particles and particles' initial conditions, and the physics, as well as the table and calculations used during the Run phase.

The Run phase is where the actions happen. In this step, particles are fired and tracked as they interact with world objects. It is at this point that the user gets the data from the experiment. A more detailed layout of Initialization and Run will be explained in Chapter Five as I walk the user through a typical example.

Structure of GEANT4 Class Interactions

In this section, we describe the different components that makeup GEANT4 and how they interact with one another. We will also cover the advantages of the C++ object-oriented language. The culmination of these two should allow the user to understand better how to use GEANT4 and prepare them for editing examples for their specific needs.

Most of the information presented in this section comes from [10], one of the best authorities on the mater and a highly recommended read for the first time GEANT4 user. Within the Agostinelli paper, they describe how they wanted the GEANT4 toolkit to be modular, flexible, as well as transparent and open to user validation. As a result of these goals, they chose to give the toolkit a modular and hierarchical structure that had a uni-directional flow between its sub-domains. Agostinelli defines "the key domains of the simulation of the passage of particles through matter [as] :

- Geometry and Materials
- Particle Interactions in the matter
- Tracking management
- Digitisation and Hit Management
- Event and Track Management
- Visualization and Visualization framework
- User Interface"

Each of these primary domains has a distinct and corresponding responsibility. As such, each domain has a list of classes that help manage and carry out there functions.

“ The toolkit offers the user the ability to create a geometrical model with a (possibly) large number of components of different shapes and materials, and to define ‘sensitive’ elements that record information (hits) needed to simulate detector responses (digitizations)”. [10] The user can then have their primary particles interact with this model world, where they can choose to generate their primary particles from either an internal or external source. Before the user runs an example, they can choose and modify the implementation of the physics processes for an event, as well as add new processes to the currently existing ones. Interactions such as these, in my experience, are best done through a graphical user interface. See the section on graphical interfaces in Chapter Four.

Looking at the primary domains listed earlier and comparing them to Figure 3.1, we see how these domains interact with one another. The vector heads represent the uni-directional flow of class interactions within GEANT4. Thus, the class at the bottom passes its information to the classes at the top, with no circular dependencies, everything is independent and modular. The Agostinelli group made these classes independent, unidirectional, and transparent so that each step in the generation and run of an example was independently customizable. For example, “the way information is extracted from the database is separate from the way it is accessed and used, giving the opportunity of using different databases and allowing their applicability to be tailored by particle, energy, material, etc.” [10]

We can use the class flow chart in figure 3.1 to walk our way through how a general example proceeds from start to finish.

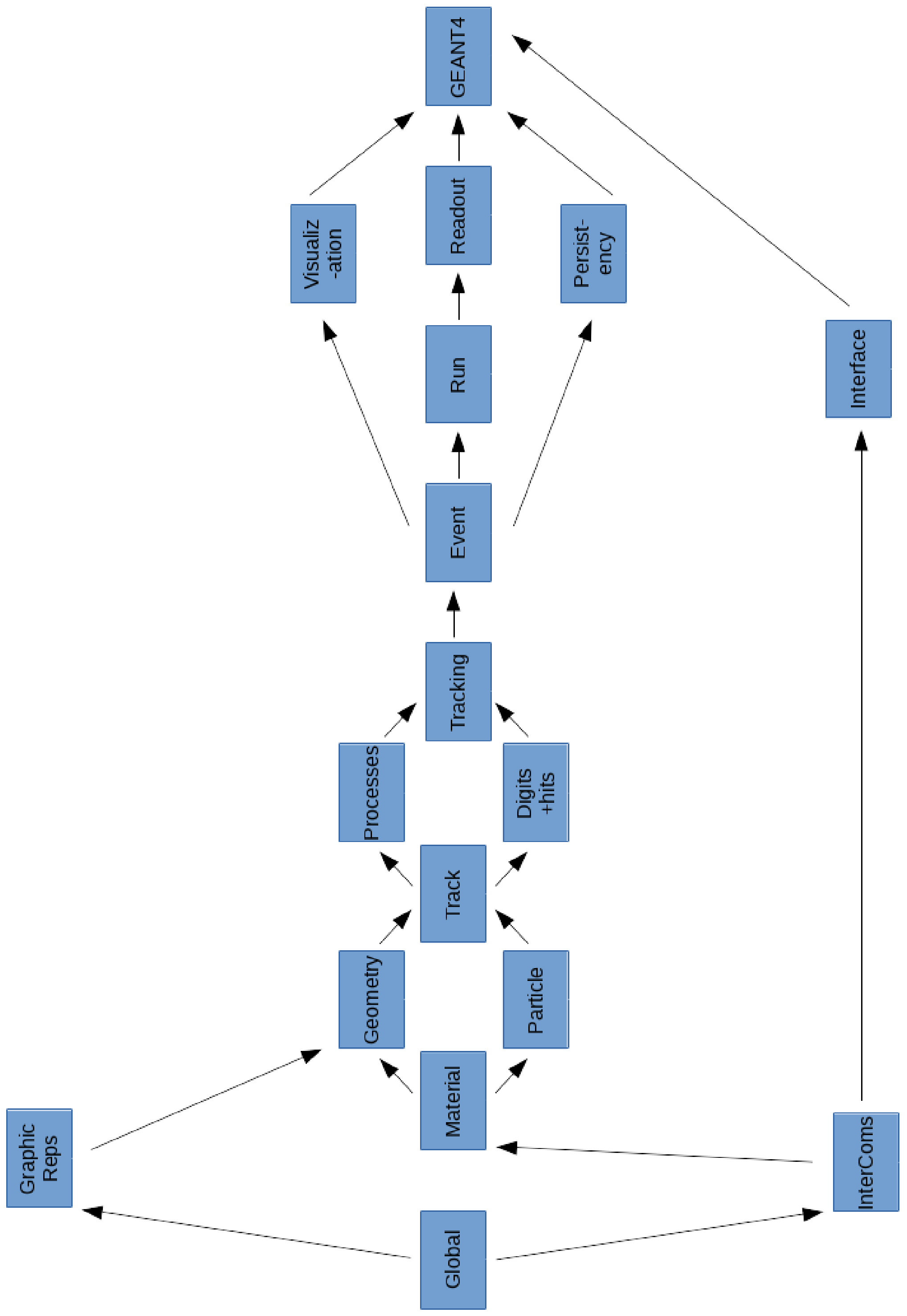


Figure 3.1: Class category diagram of GEANT4. The vectors represent the uni-directional dependencies; the category at the vector head uses the adjoining category.

From the diagram, we see that the Global category is the first to be used. This category is in charge of the system of units, constants, numerics, and random number handling. [10] The Graphical Representations category and the Intercoms category both use Global. The Intercoms mainly act as a manager between the user and GEANT4 via the user interface. Graphical Representations plays a role in the volumes for detector description and navigation in the geometry model. Our information is then passed through to the Material, Particle, and Geometry categories. The first two of which "implement facilities necessary to describe the physical properties of particles and materials for the simulation of particle-matter interactions." [3] The last of which "offers the ability to describe a geometrical structure and propagate particles efficiently through it." [1] At this point, we get the categories that are responsible for describing our actions. The first category to play a role in this is the "Track category [which] contains classes responsible for the tracks and steps, [and is] used by the Process category, which contains implementations of models of physical interaction: electromagnetic interactions of leptons, photons, hadrons and ions, and hadronic interactions." [1]

At this point, all the categories mentioned so far are used either directly or indirectly for Tracking, the category that "manages their contribution to the evolution of a track's state and undertakes to provides information in sensitive volumes for hits and digitization." [10] At this point, the Event category manages event tracks, and the Run "manages a collection of events that share a common beam and detector implementation." [1] At this point, Event passes the data to the Readout through Run and passes data to the Visualization and Persistency categories.

Understanding the interactions and flow of these categories and the architectural design is crucial for a thorough understanding of the structure and behavior of GEANT4. As such, the author strongly suggests the reader read section

2.3 of the Agostinelli group for a quality outline of the GEANT4 architecture.

CHAPTER 4

WORKING WITH GEANT4

Installing GEANT4

In this section, a summary of the installation procedure used in this thesis will be laid out for the reader to replicate. There are three primary operating systems on which the user can install GEANT4; these include Windows 10, macOS, and Scientific Linux. The installation instructions for Windows 10 or macOS will not be covered here. The user is encouraged to see the GEANT4 installation website if they are using one of these operating systems. The version of GEANT4 used in this paper was version 10.015.p01 and was ran on Scientific Linux 7 (at the time believed to be SL6).

There are three primary prerequisites required to run GEANT4; some may come installed on the operating system; the user will need to check for these prerequisites. These include a C++ compiler, CMake 3.3 or higher, and the GEANT4 source code. Other prerequisites may exist for optional components of GEANT4.

Visualization interfaces are included in the list of optional components and will need to be individually 'activated' during the cmake command. GEANT4 has multiple graphical user interfaces the user can choose from, each designed for one particular application or another. The one I used was the OpenGl with the QT interface. I chose this visualizer for three primary reasons.

The first reason is that it required the fewest additional prerequisites to use. A requirement I added only after I ambitiously and foolishly tried to get all the prerequisites and visualizers working despite my limited Linux experience or knowledge. This exploit cost me much needless frustration and wasted time, as each prerequisite presented a unique challenge (each one having a different installation procedure). As such, I ended up valuing the visualizer with the fewest prerequisites.

Secondly, after running the stripped-down GEANT4 with no add-ons, I found that running GEANT4 with no visualization or with the default 2-d noninteractive visualizer made it rather difficult to check or understand what I was doing. So having a visualization that I could interact with became my second requirement.

Lastly, using GEANT4 with no addons was next to impossible for me as I did not know what commands I could use, what they do, or how to use them. GEANT4 did have a HELP menu I could navigate through by using the terminal, but this quickly becomes a tedious time sink as the user is left to sift through the command tree before they found the one they need. They would then have to back out of the HELP menu and type in the command. The OpenGL with QT interface has a keyword search function for commands. It also includes descriptions of function, use, and allowed the user to output the command directly to the command line with a simple double click. Significantly speeding up the time of use.

There are many other configurations in which one can choose to run GEANT4 beyond that of visualizations. Which configuration the user chooses to run GEANT4 in will change the installation procedure; as such, this section will cover only how I installed GEANT4. If the user wishes to very from the installation described in this section, they are encouraged to read the installation documentation on GEANT4. There were two installations of GEANT4 used in this Thesis. One was the initial basic install of GEANT4; the other included the OpenGL with the QT visualizer and user interface.

Basic Install

The user can find the installation procedure in the APPENDIX titles “Useful links”, and as cited, however, the necessary steps will be provided here for quick future reference. To begin the installation, download the source code. Next, “Unpack the Geant4 source package geant4.10.05.tar.gz to a location of your choice. For illustration only, this guide will assume it has been unpacked in a directory named /path/to so that the Geant4 source package sits in a subdirectory” [11]

- /path/to/geant4.10.05

“We refer to this directory as the source directory. The next step is to create a directory in which to configure and run the build and store the build products. This directory should not be the same as, or inside, the source directory. In this guide, we create this build directory alongside our source directory:” [11]

- cd /path/to
- mkdir geant4.10.05-build

“ To configure the build, change into the build directory and run CMake:” [11]

- cd /path/to/geant4.10.05-build
- cmake -DCMAKE_INSTALL_PREFIX=/path/to/geant4.10.05-install -DGEANT4_INSTALL_DATA=ON /path/to/geant4.10.05

The user will see a long output string with the final lines.

- Configuring done

- Generating done

- Build files have been written to: /path/to/geant4.10.05-build

at this point, the user will need to run the following command to determine their maximum number of processors. [12]

- lscpu

Where the user wants N=CPU(s), such that we can run the next GEANT4 command as

- make -jN

Note, the user can use a number fewer than N but no greater. For me, this was N=4. The final command will be

- make install

The installation is now complete. At this point, the user will need to read the section of this thesis titled "How to Build and Run an Example."

Install with Visualization

Again, for OpenGL with QT visualization and user interface, there is a required addition of two prerequisite packages QT4 (>= 4.6 or QT5) headers and libraries, and OpenGL or MesaGL headers and Libraries. See the "USEFUL LINKS" for links to the prerequisites and their installation procedure. I chose OpenGL and QT4 for this install. After all the prerequisites where installed and verified, the installation procedure was the same as the above with the modification of the CMake line, which should now read:

- cmake -DCMAKE_INSTALL_PREFIX=/path/to/geant4.10.05-install -DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_QT=ON -DGEANT4_USE_OPENGL_X11 /path/to/geant4.10.05

The install will follow the procedure laid out above.

However, as Murphy's laws dictate, things went wrong. Despite having successfully installed the prerequisites and adding the required D-flags for the OpenGL with the QT interface, the visualizer that I should be seeing was not appearing, either by default or with the explicit commands. I attempted to get a working installation of GEANT4 with this visualization on my native Scientific Linux PC, and a native Windows, as well as a remote Lewis account provided by the University. Linux (using ssh) and Windows (using MobaXterm) where both used to remotely access the Lewis account.

All attempts failed with for one reason or another. So I abandoned my goal of a native GEANT4 and focused my attention on getting a working GEANT4 on the remote Lewis computer provided by the university as this had IT support that I could contact. After contacting the IT in-charge of the accounts on Lewis, we found that they had the intended visualization and GUI working just fine on their end, it was only failing to show up on my end regardless of whether or not Lewis was accessed on Linux or Windows (It was defaulting to the original 2-d viewer). See fig 4.1 and 4.2, respectively.

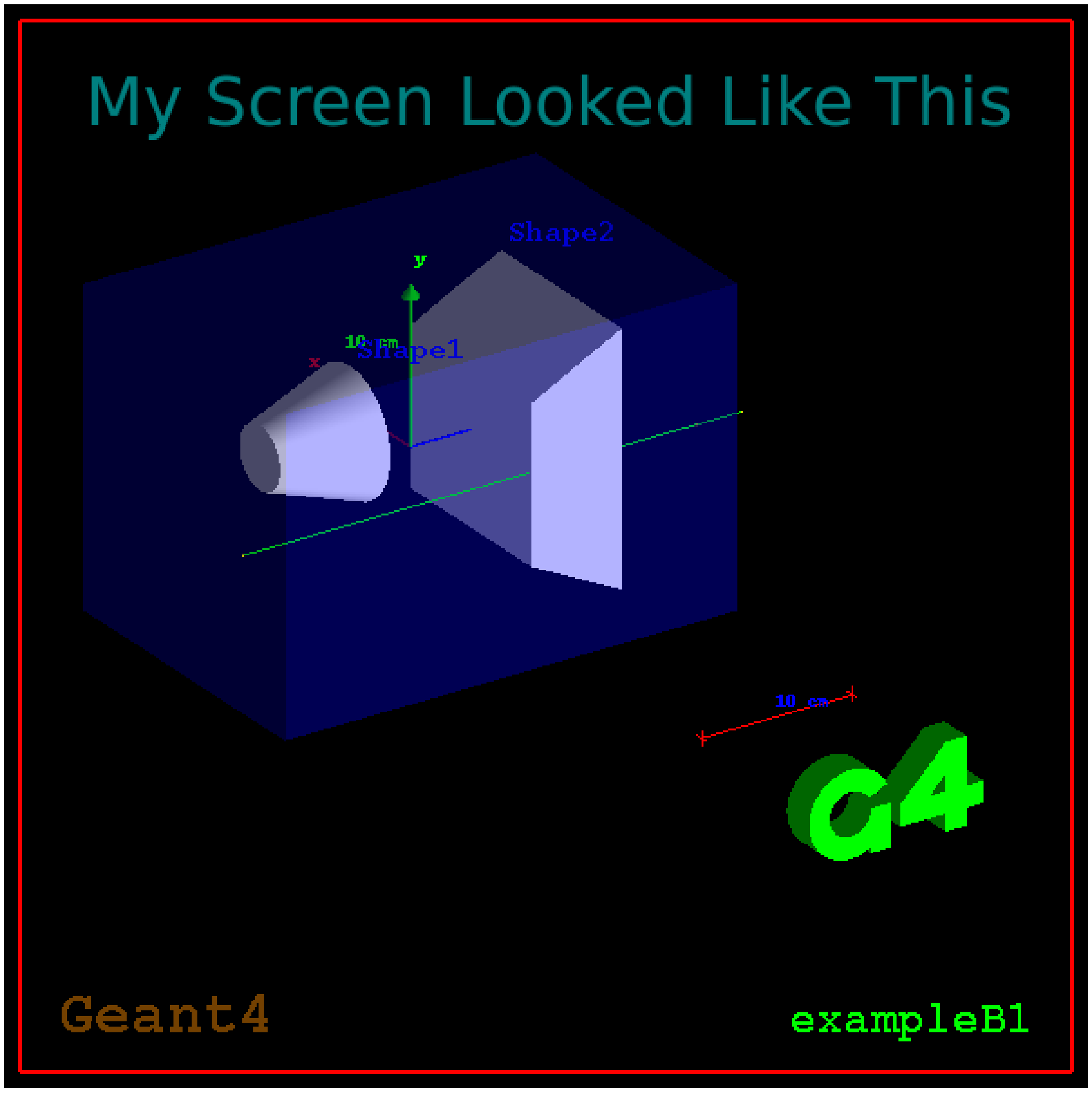
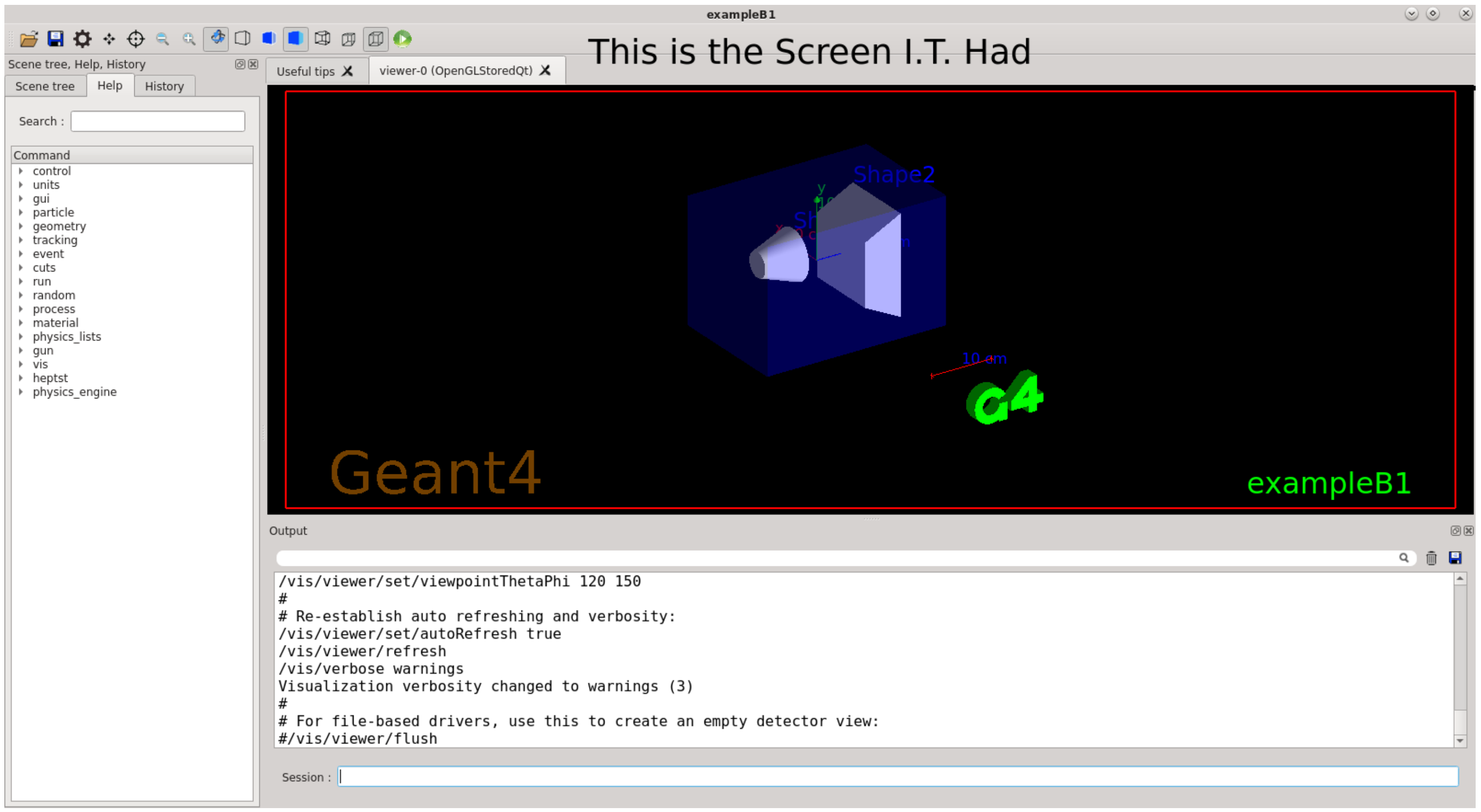


Figure 4.1: Visualization provided on the users end (Default)



```
/vis/viewer/set/viewpointThetaPhi 120 150
#
# Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/viewer/refresh
/vis/verbose warnings
Visualization verbosity changed to warnings (3)
#
# For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush
```

Figure 4.2: Visualization and User interface provided on the IT's end (OpenGL)

With both IT and myself confused by this, IT decided we could create a singularity containing GEANT4 with a working visualizer/GUI and copied this to my native Linux system. The singularity is a file system that contains the OS and the working GEANT4; this includes all the needed prerequisites for both GEANT4 and the visualizer. Once on the local machine, both the visualization and GUI worked correctly. Nither IT or I were able to understand the exact cause of the problem. Despite the many frustrations I was experiencing, this was one of my favorite experiences of the thesis. This was due to the very helpful and friendly Predrag Lazic, who, through our conversations I discovered, was a Croatian physicist turned IT. I truly enjoyed my conversations with him. I also greatly appreciate the above and beyond help that he provided.

Navigating Through GEANT4

This section will cover the basic commands that the user should concern themselves with to navigate themselves through GEANT4 and run some examples. After installing GEANT4, it will be assumed that the OpenGL visualization and GUI was installed, the user should find themselves in $\$/path/to/GEANT4\_INSTALL$ directory. To see the available directories under the current location using the command *ls*, for a list. This command will output a list of directory names. We want to move into the directory *share*, To move through directories, use the command: *cd desired_directory_name*. We need: *cd share*. Similarly we can use: *cd desired_directory_path* and go straight to the examples directory that we will be working in for the duration of this thesis. This would look like: cd $\$/path/to/GEANT4\_INSTALL/share/Geant4-10.5.1/examples$. Here we can do our list command (*ls*) and look for the following sub-directories.

- Basic
- Novice
- Advanced
- Extended

Each directory above contains a list of different ready to go examples. They will, however, need a basic set up as none of them are pre-compiled. For more information on all available examples and what they can do, it is recommended to visit the GEANT4 GitLab page. From here forward, all we will be concerned with is using example/extended/electromagnetic/TestEm11 for reasons to later be explained.

- First, we need to enter into the desired directory with the *cd* command

  cd extended

- Second, we will need to make a build-directory with the *mkdir* command. Note: This can be named whatever the user would like but the author finds this naming convention to be convenient.

  mkdir TestEm3-build

- Next, enter into the build directory.

  cd TestEm3-build

- Now we need to run our make commands: The first is to compile, the second builds.

  cmake ..

  make

- The example can then be used by running the executable

  $./TestEm11$

Using a GUI

In this section, we will go over how to run some basic experiments in the TestEM11 example using the OpenGL viewer and GUI. It needs to be noted that some examples have different requirements for executing GEANT4 with visualization; as such, the README documentation for the example in question should be read. The user can read the README within the GEANT4 directory with *vim*, or by reading the GitLab documentation. A link to the TestEM11 README will be posted in "SUGGESTED READINGS".

To run the TestEm11 example with visualization and GUI, we need only follow the steps above, assuming the user has installed GEANT4 with all required prerequisites. The resulting window should look like Figure 4.3.

The three primary sections that the user should concern themselves with are labeled A, B, C. Section A, is where our visualizations will be displayed. Section B, within this section, we have a search bar that we can use to search the output, the output itself, and the session box where commands are entered. A complete list of commands can be found to the left in section C.

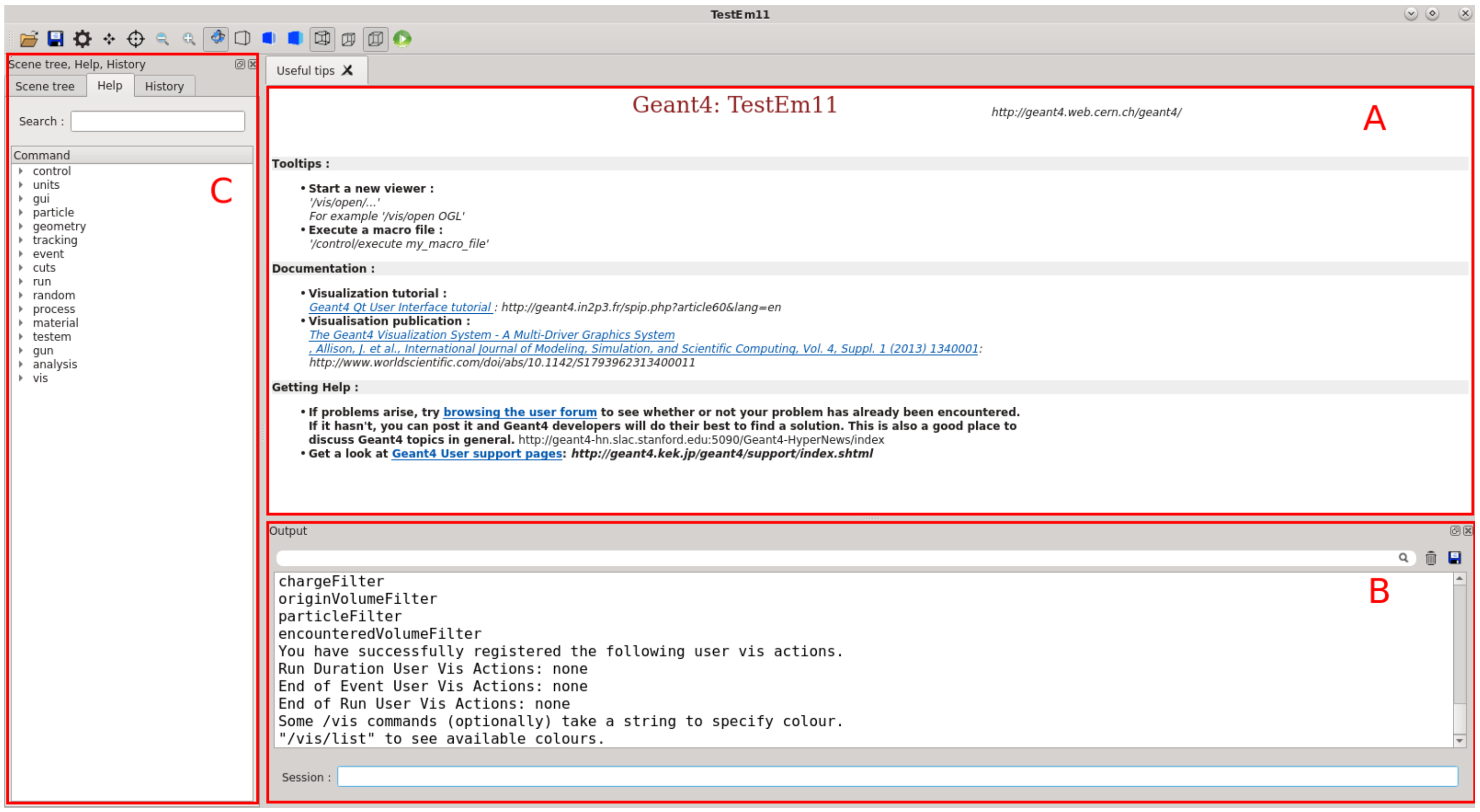
To see the visualization, we can click on 'control' in section C, then double click 'execute', this will put the command in the session bar located in section B. Click in the session bar and type "vis.mac" at the end so that it looks like

- /control/execute vis.mac

, and hit enter. This command will load the macro file for visualization. A macro file is a document containing an ordered list of commands; this allows us to save the time it takes to enter commands one after another or repeat a given prosses multiple times. For illustration purposes, we can now run the command

- /run/beamOn 10

We will look at how to run the TestEm11 example in the following chapter, where we will be comparing our results with that of Carrier.



Geant4: TestEm11   http://geant4.web.cern.ch/geant4/

Tooltips :

- **Start a new viewer :**
  '/vis/open/...'
  For example '/vis/open OGL'
- **Execute a macro file :**
  '/control/execute my_macro_file'

Documentation :

- **Visualization tutorial :**
  Geant4 Qt User Interface tutorial : http://geant4.in2p3.fr/spip.php?article60&lang=en
- **Visualisation publication :**
  The Geant4 Visualization System - A Multi-Driver Graphics System
  , Allison, J. et al., International Journal of Modeling, Simulation, and Scientific Computing, Vol. 4, Suppl. 1 (2013) 1340001:
  http://www.worldscientific.com/doi/abs/10.1142/S1793962313400011

Getting Help :

- **If problems arise, try browsing the user forum to see whether or not your problem has already been encountered. If it hasn't, you can post it and Geant4 developers will do their best to find a solution. This is also a good place to discuss Geant4 topics in general.** http://geant4-hn.slac.stanford.edu:5090/Geant4-HyperNews/index
- **Get a look at Geant4 User support pages: *http://geant4.kek.jp/geant4/support/index.shtml***

```
chargeFilter
originVolumeFilter
particleFilter
encounteredVolumeFilter
You have successfully registered the following user vis actions.
Run Duration User Vis Actions: none
End of Event User Vis Actions: none
End of Run User Vis Actions: none
Some /vis commands (optionally) take a string to specify colour.
"/vis/list" to see available colours.
```

Figure 4.3: Visualization of TestEm11 on Start Up (OpenGL)

CHAPTER 5

VALIDATION OF GEANT4

Validation of the Carrier Geometrys

To check my capabilities of using GEANT4 as a new user, I tried to replicate the previously published results of Carrier. Particularly, I wanted to focus on trying to replicate the results found in Figures 2.a, 2.b, 3.a, 3.b, 4.a, 4.b, and 4.c of his paper, where the geometries represented in these graphs are laid out by Carrier in the first two paragraphs of Section C.1 of his paper (see citation [2]).

However, in an email with an advisor of Carrier's, Louis Archambault, I learned that the geometries used by Carrier where built from scratch. As this was far more than I was able to do starting off, I decided to see if I could find a pre-installed example that would have the basic functionality I needed.

To start, I needed to find a way to get GEANT4 to generate a Depth-Dose Graph. Asking around on the relevant forums and searching around on the internet, I learned that my best options for examples that could meet my needs where the TestEm11 and the TestEm12 examples.

As a quick aside for the future user, never be afraid to post on a topic/product specific forums if you need help with understanding how to use particular software, code, toolkit, ect., they are extremely useful and can save you a great deal of time and frustration. You can always find something else to productively use your time while you are waiting on a response. Do use discretion when posting online, do not post content deemed sensitive or confidential. Links to all my forum posts will be provided in the Appendix. Secondly, researching the whole of the internet can sometimes be a pain, particularly when you believe the results lie somewhere in a particular web-page but are having trouble navigating it, or as in this case, there may too much to sift through in a timely manner. To save time and narrow down the search, you can Google "site:https://yourwebsitehere.wha/tever/ your AND search NOT words", and this will search only the intended site down the tree. A link to the Google operator will also be provided in the Appendix.

Looking at the README for the Electromagnetic package, we see that both TestEm11 and TestEm12 are capable of plotting depth dose in rectangular and spherical geometries, respectively. As the geometries that I am trying to replicate in Carrier's work are Cartesian based, some even explicitly calling for rectangular slabs, I chose to use TestEm11.

With this, it would seem that only a small alteration to the TestEM11 code would be needed if I am to replicate his work exactly. However, as this was meant to be a 'use and check' exercise, I chose to replicate only the parameter that I was able to replicate by issuing the built-in commands.

The README for this clarified that Histogram #8, which states "longitudinal energy profile (in MeV.cm2/g), as a function of x/r0 Where r0 is the range of the primary particle" needed to be called to graph the depth dose. So I included the command for activating the generation of this histogram in my macro file. A description of the command necessary in section C of our GUI. It reads as:

- /analysis/h1/set id nbBins valMin valMax unit
- USED: /analysis/h1/set 8 100 0. 1.0 none

and we can chose the file name with the command

- /analysis/setFileName File_Name.root

Next, I had to figure out what parameters I could match with the built-in commands of TestEm11. I recognize that my results will vary from those obtained by Carrier as a result of this constraint, but we expect to see a trend that looks very similar. The parameters we found that could be controlled with commands and replicated are as follows:

- Homogeneous
- Simi-infinite Object Volume
- of Molybdenum or Beryllium
- Irradiated by a Normally Incident
- Mono-Energetic
- Electron Beam
- With Energies (.5 MeV and 1.0 MeV for Mo) and (.521 MeV and 1.033 MeV for Be)

Some interpretation of Carrier's work was made to arrive at these parameters. For example, he states that "1000000 primary electrons" [2] are used in the first geometry, and "a total of 1000000 primary electrons are used" [2] in the second geometry. When the /gun/number/ 1000000, I ended up with a much higher energy deposition than Carrier, however, when this was set to 1, and /run/BeamON 1000000 was set, I ended up with an energy deposition remarkably close. Secondly, as I found no explicit way to make a semi-infinite slab within GEANT4, it was assumed that the slab is semi-infinite when compared to the electron (In this case, a 'large' 1 cm cube was used as to big a cube lead to longer computation times). Lastly, I could not find any documentation or clarification on what a ‘broad’ beam is or how to use it. So for this comparison, I was only able to meet the mono-energetic and normally incident parameters and assumed that the broad beam was referring to the default beam setting.

With a growing list of commands and multiple geometries to replicate, many of which I wanted to run multiple times, I decided to find a preinstalled macro file that I could alter to use for my runs. As a macro file is just a list of command for the program to execute, alteration of this file would save me the time of having to enter each command one by one; instead, they could all be passed to GEANT4 with the macro execution command

- /control/execute NAME_OF_FILE.mac

I chose to model my macro file after the sabdia.mac file provided with TestEm11. This file included several commands that I needed to replicate the first and second Carrier geometries, commands that set parameters such as the absorber number, size, and type, the physics used, the particular type and energy, the histogram information, and the run command. See the altered file attached in “SUGGESTED READINGS” named FinalRun.mac file. Note, this file can be used for all the geometries by commenting and un-commenting the appropriate section. Again, the user does not have to use a macro file; however, a macro file will make the process much quicker and convenient as the order of commands does matter.

To set up the geometry, we needed to make the rectangular box of multiple layers into one layer. A note from TestEm3 told us that, “The number of absorbers and the number of layers can be set to 1. In this case, we have a unique homogeneous block of matter, which looks like a bubble chamber rather than a calorimeter .” [13]. We could then set the number of layers in TestEm11 to one, so we could have a homogeneous block of Mo.

- /testem/det/setNbOfAbsor 1
- /testem/det/setAbsor 1 G4_Mo 1 cm
- /testem/det/setSizeYZ 1 cm

There was no mention of which physics standard Carrier used, so I selected "emstandard_opt3" as it was described as "best standard EM options". It was not until later, after rereading the email from Louis, that I noticed he recommended I use the physics list from the MedLinac example, which defaults to using "emstandard_opt3". As the physics list can often be the trickiest part (It was unclear whether this is in regards to coding or in getting accurate results).

Finally, looking at the README for TestEm11 we see that "there are a minimum of 5 parameters [that] define the geometry

- The Number of Absorbers (NbOfAbsor)
- The Material of Each Absorber,
- The Thickness of Each Absorber,
- The Transverse Dimension of the Stack (sizeYZ),
- The Number of Divisions of Each Absorber (NbOfDivisions)

My edited sabdia.mac then met the minimum requirements of both the example and the first Geometry of Carrier without altering any of the code. I saved my new macro file as the attached FinalRun.mac file.

I was then able to execute the macro file. The details of this will be covered in the next section. An attempt to run the Second Carrier geometries with no code alteration was made as well and will be covered below.

Results of First Carrier Geometry Validation

Below I have provided figures comparing the original work done by Carrier with my results. The graphs were generated by using ROOT. Each run would write and save to a file.root. This root file could then be opened in ROOT with one of two methods: open in the terminal with “rootbrowse file.root” or with a coded file.C and the ROOT terminal command “.x file.C”.

To build the graphs found below, I had to code a file.C that could read the file.root and write it to the same graph as the Carrier data, where the Carrier data was formed by taking 50 (XY) values from the graphs and importing them into a file.csv file; this was done with LibreOffice Calc. It is important to note that the file.csv needed to be saved with the field deliminator being [space] and the text deliminator being (“”) for ROOT to be able to read them. The ROOT.C file will be attached in the appendix titled “Suggested Readings” for reference. Aesthetic alterations were made to the graphs to make them easier to view. Both the Carrier data and my results where displayed using a polynomial of 6th degree best fit.

There are four primary criteria we looked at to determine the quality of our validation runs. The first two are the Minimum and Maximum $\frac{z}{R_0}$ depth x-values and there corresponding Energy Deposition y-value. The second was the xy-value of the Energy deposition peak. Lastly, we looked at whether or not the overall trend of the graph looked similar.

The first set we have in Figures 5.1 is Beryllium at .521 MeV. We see that the Gun run and the Carrier graph are in strong overall agreement. The GPS run, however, has the general shape and peak structure but has an entirely different Full-width-half-max (FWHM) and X-value for its peak.

For the Beryllium samples at 1.033 MeV, see figure 5.2, again, we see a good similarity between all three data sets, with the GPS still showing a slight shift compared to the others but this time to the right.

The following Figure 5.3 shows the Molybdenum at 0.5 MeV. Again the GPS has a weird behavior where it is left-shifted and has a smaller FWHM. The Gun run, while higher than Carrier's, is still within the predictive range of Sandia, EGSnrc, and MCNP showed in Carrier's paper.

Last, of the first geometry, we have Figure 5.4, the graphs for the 1.0 MeV sample of Molybdenum. Our graphical trends match once again, with the GPS and the Gun runs being identical and therefore represented by the teal line. It is worth noting that there is little disparity between the Beryllium's maximum energy deposition, where there is a noticeable difference for the Molybdunum's maximum energy deposition with an increase in this difference as energy increases.

For the cases of water at 100 KeV, 1 MeV, and 10 MeV; Figures 5.5, 5.6, 5.7 respectively, all graphs are largely off, resembling only the general trend of their Carrier counterparts. The energy deposition appears to be peaking at a much higher value than that of Carrier. There is no current explanation of why the last graph for water at 10 MeV using Gun did not even finish. It is also observed that for water as the energy increases the magnitude difference decreases,

It is unknown where these discrepancies are coming from, but there is a correlation between the material conductivity and the energy deposition error. With Beryllium having a value of $2.5 \times 10^7 \frac{\text{S}}{\text{m}}$ and Molybdenum $2 \times 10^7 \frac{\text{S}}{\text{m}}$. So it would seem that as the conductivity lowers the error increases. I see this trend continuing for the water cube (which should be ideal water, so a value of $2 \times 10^{-6} \frac{\text{S}}{\text{m}}$). It is here that I see that I am missing some critical "detail or detail's" that Carrier includes, but I have yet to determine precisely what.

Overall, it seems that the Gun runs are the most consistent, as are the runs of 1 MeV. I have no explanation for this.

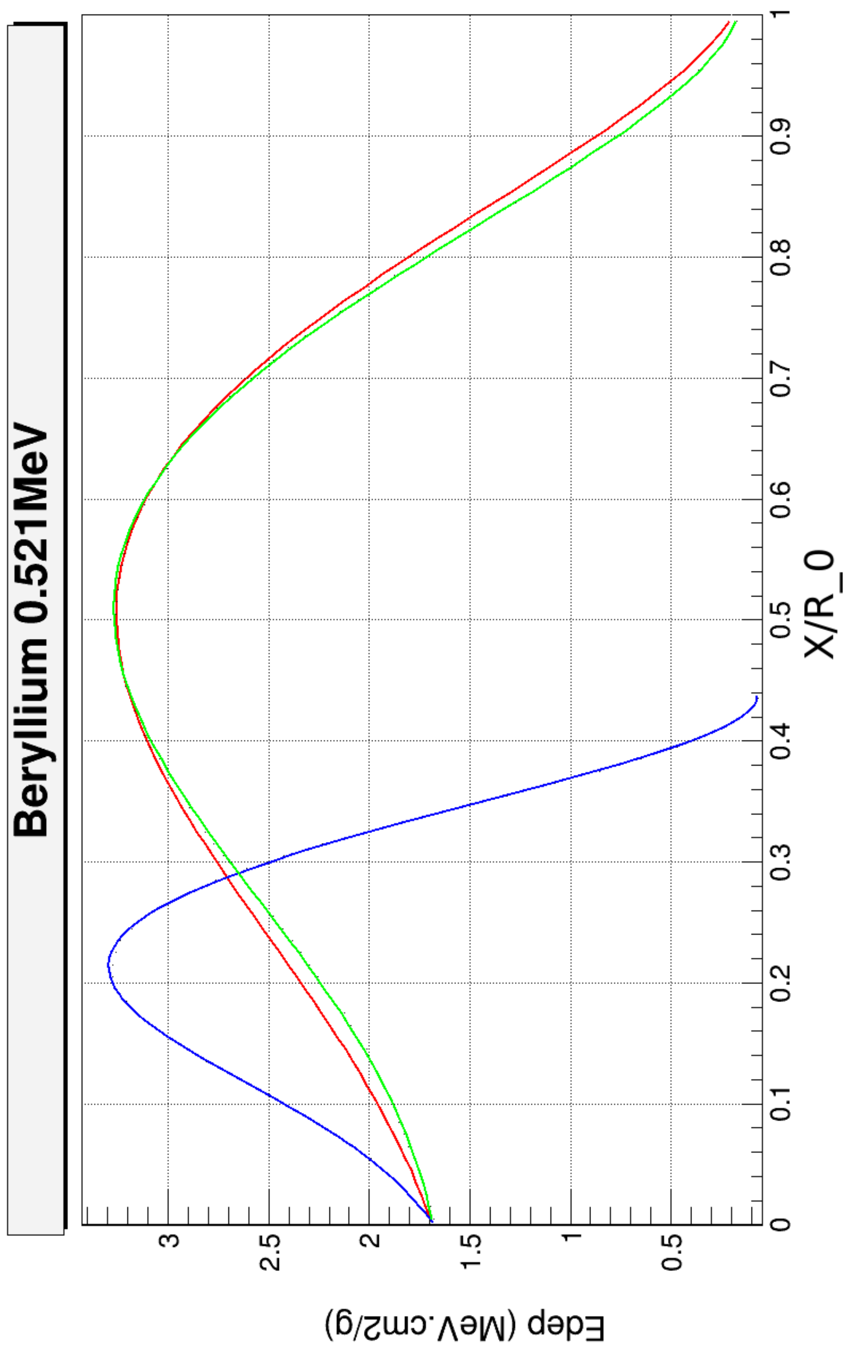


Figure 5.1: Beryllium at 0.521 MeV : Carrier's data in red. Gun run in Green. GPS run in Blue.

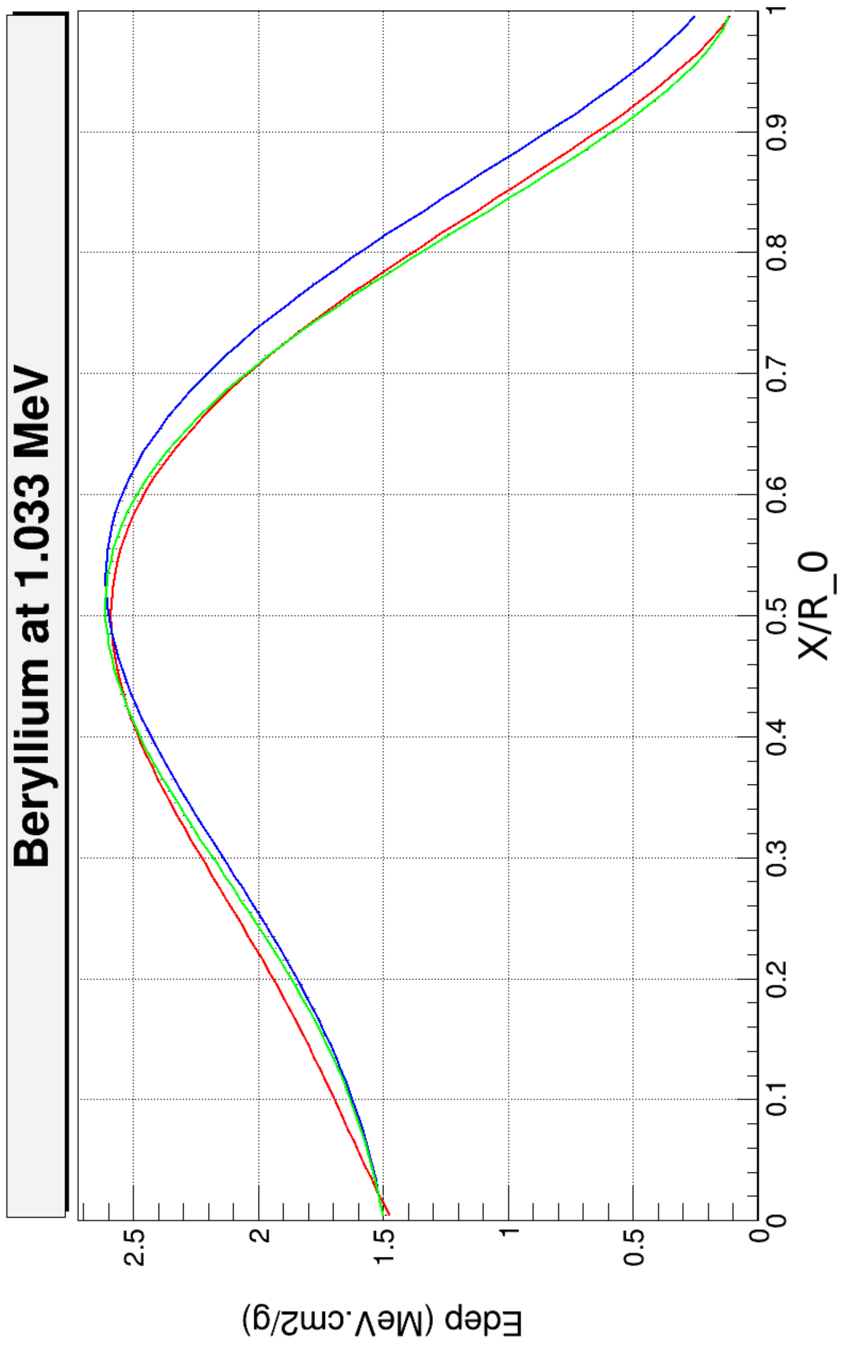


Figure 5.2: Beryllium at 1.033 MeV: Carrier data in red. Gun run in Green. GPS run in Blue.

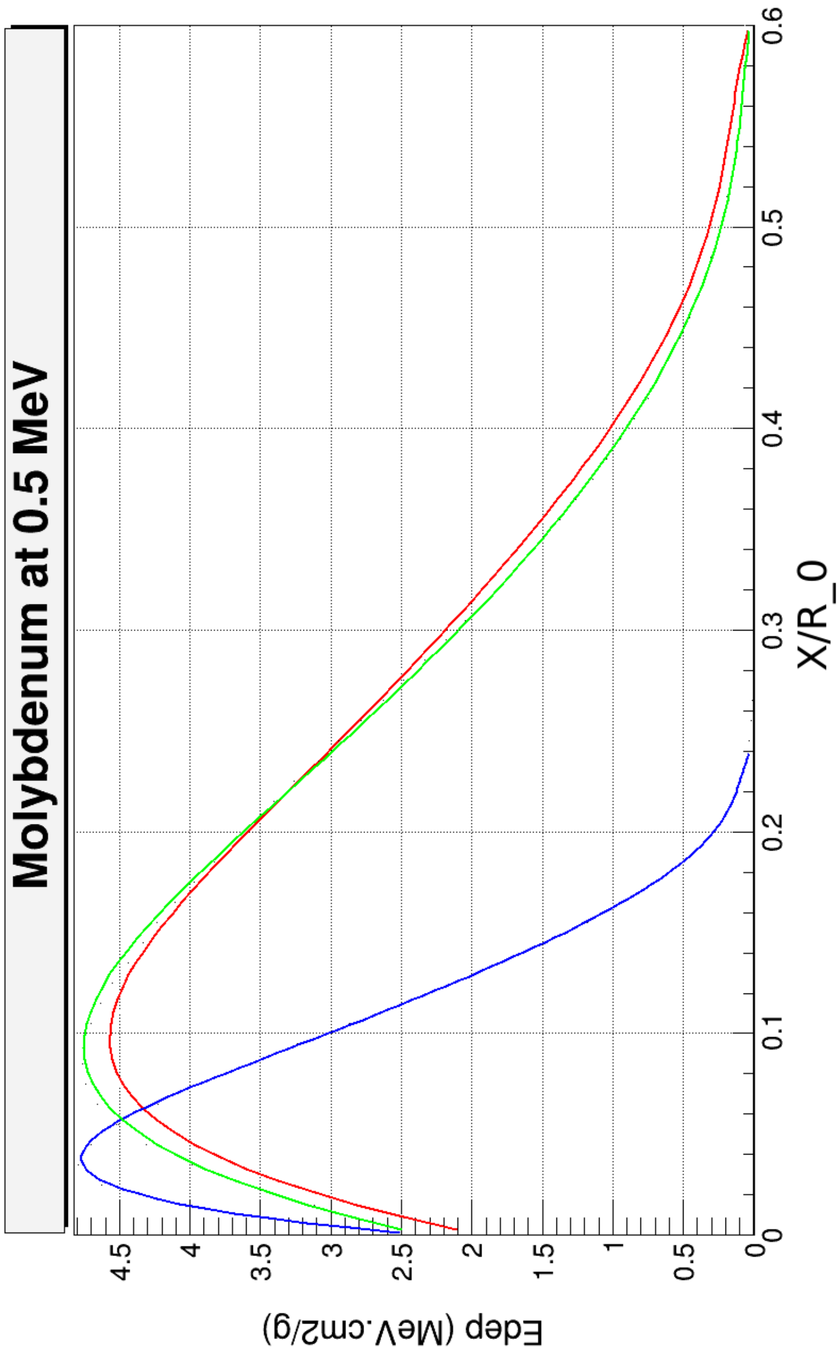


Figure 5.3: Carrier's Molybdenum at 0.5 MeV in red. Gun run in Green. GPS run in Blue.

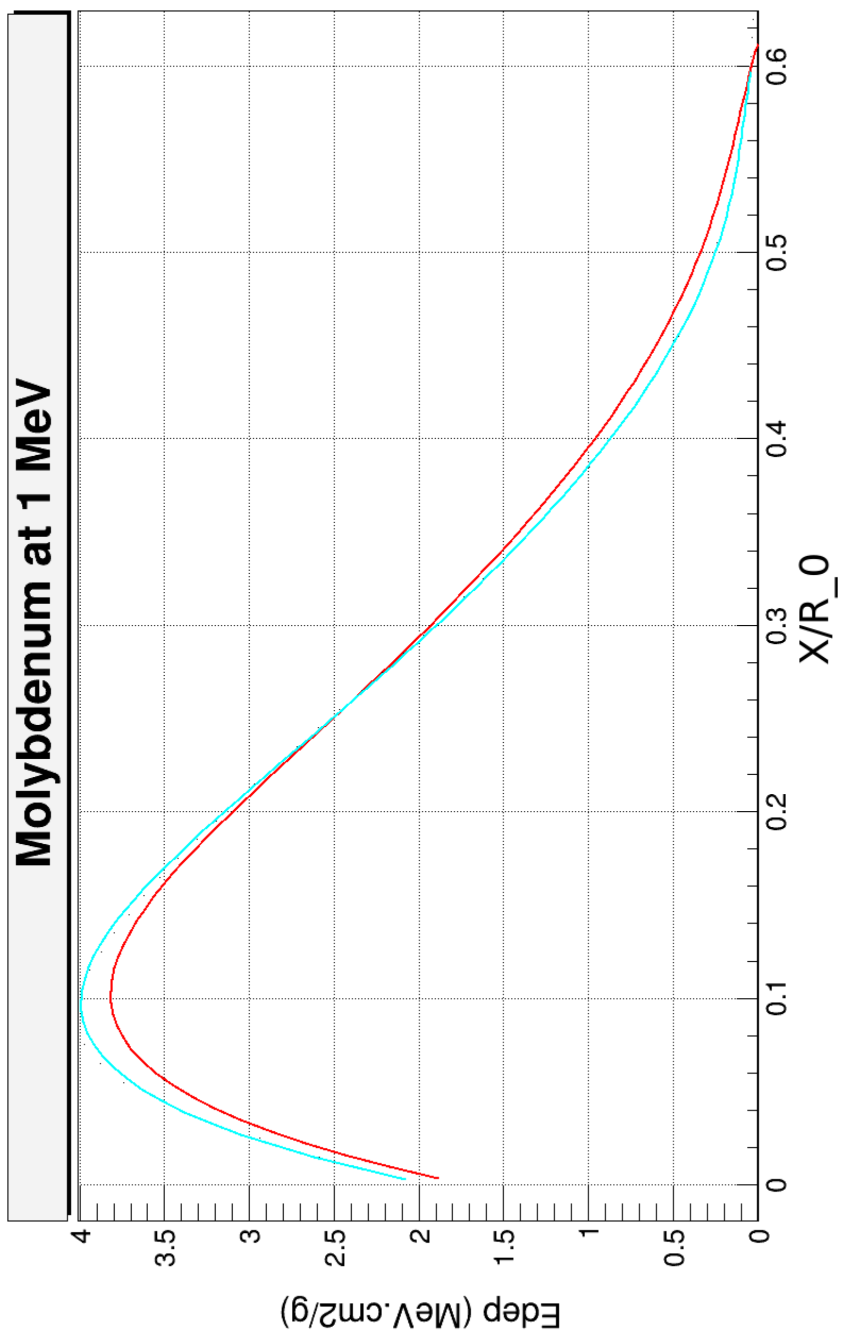


Figure 5.4: Molybdenum at 1 MeV: Carrier's data in red. Gun and GPS run in Teal.

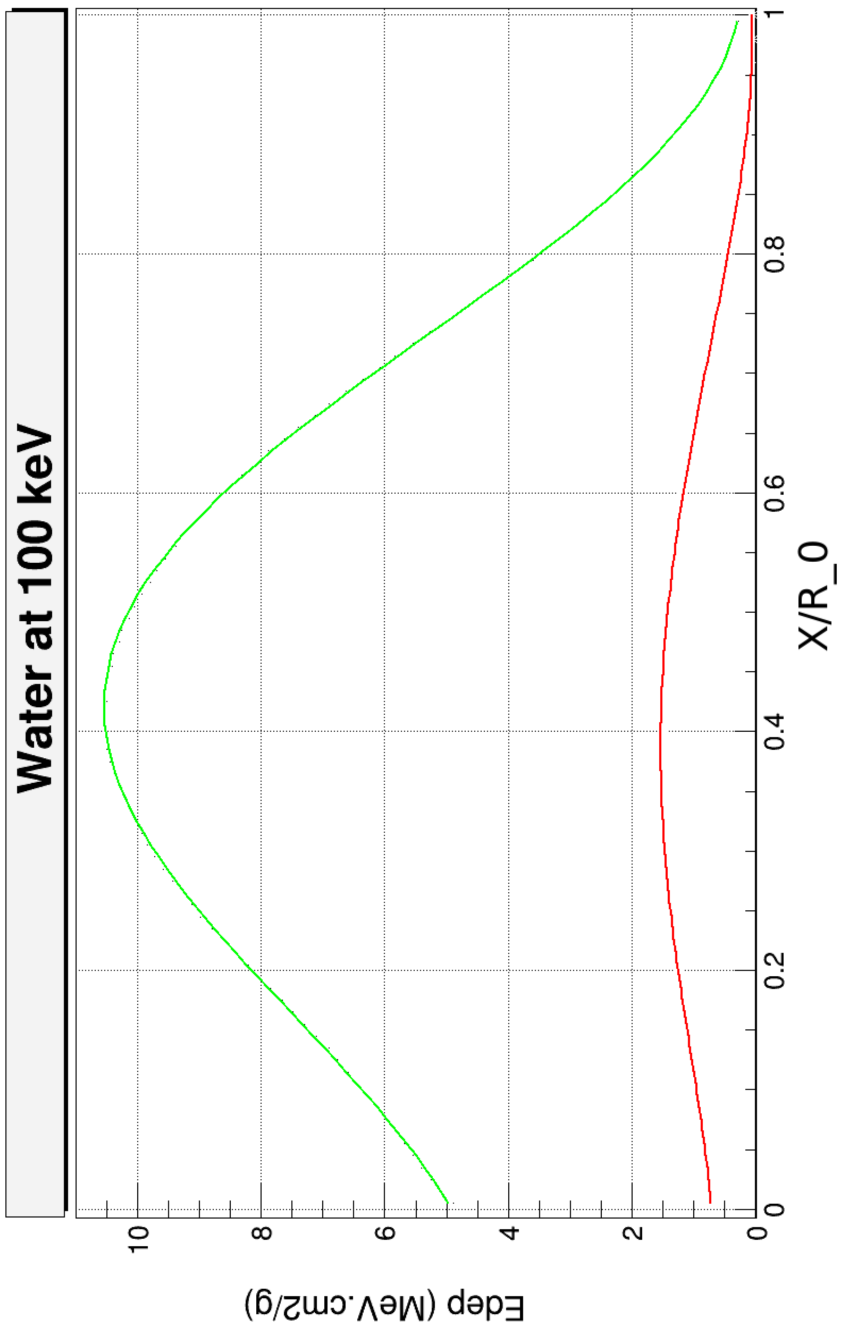


Figure 5.5: Water at 100 keV: Carrier's data in red. Gun run in Green.

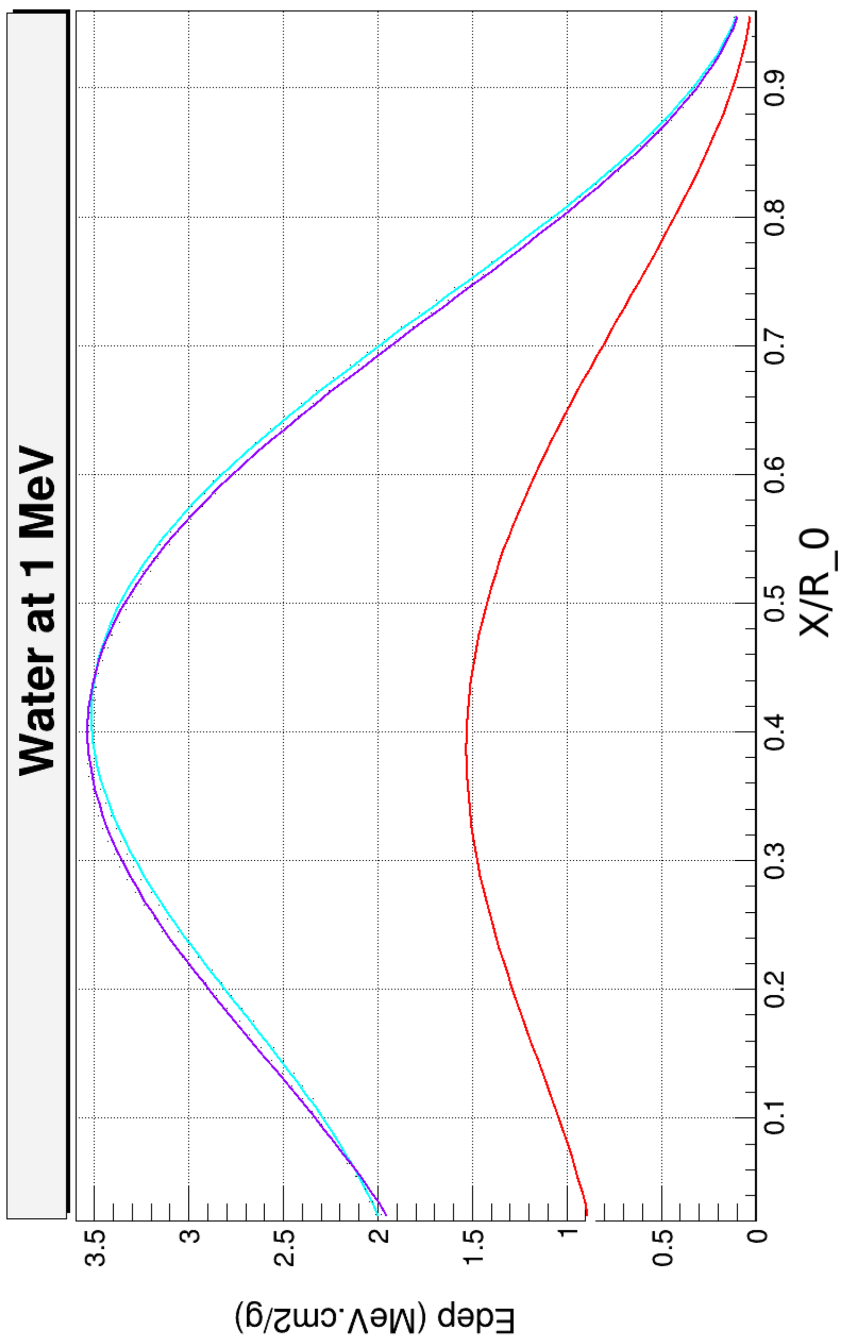


Figure 5.6: Carrier's Water at 1 MeV in red. Gun and GPS in teal. Cut run in purple.

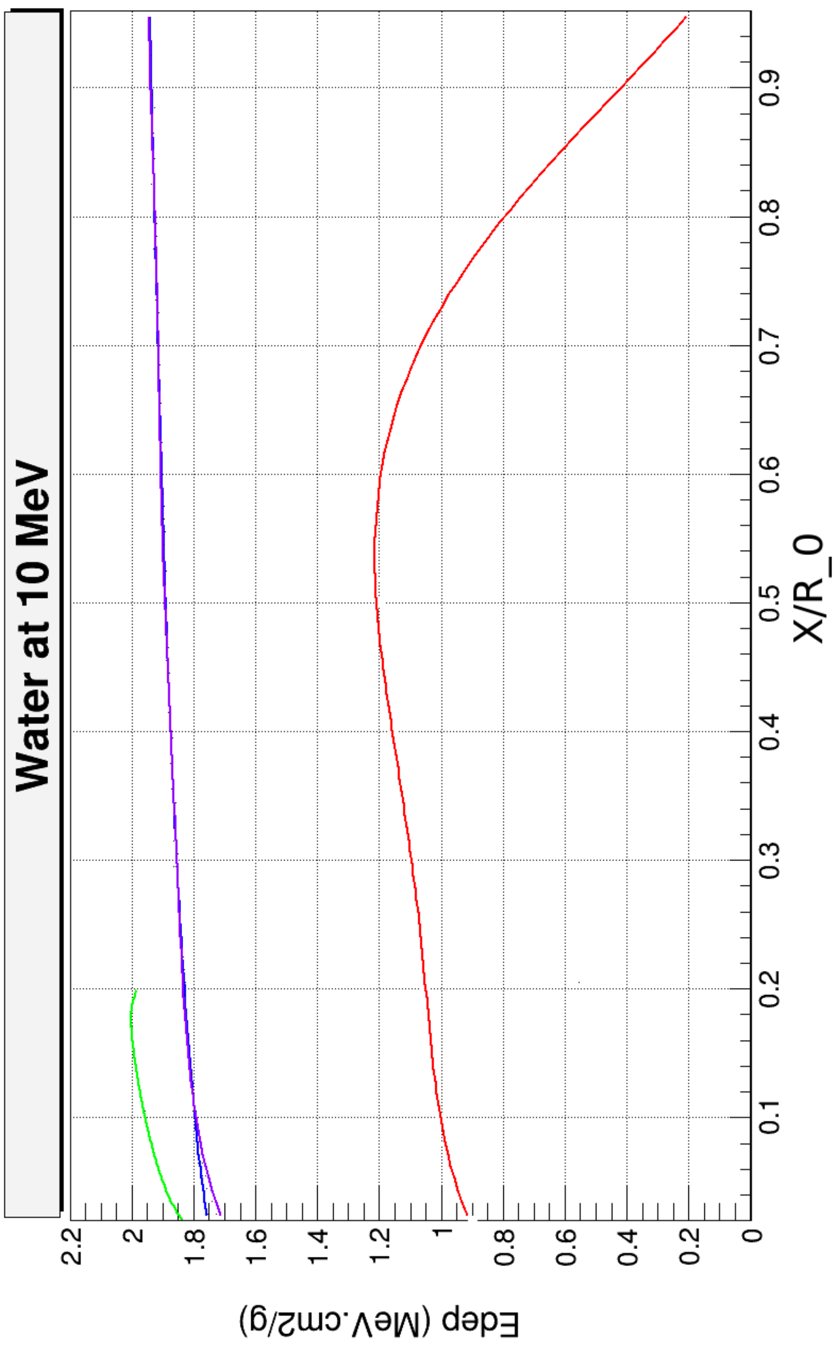


Figure 5.7: Carrier's Water at 10 MeV in red. Gun run in Green. GPS run in Blue. Cut run in purple.

Post Code Alteration Validation

After Running the code as it came with the default GUN commands, I attempted to alter the code at a basic level to achieve more accurate graphs. The first alteration to the code that we made was to globally replace G4ParticleGun with G4GeneralParticleSource, doing so would give us several options when using our gun that we did not have earlier. The primary benefits of the G4GeneralParticleSource package are that "it allows the specifications of the spectral, spatial, and angular distribution of the primary source particles."

To use G4GeneralParticleSource (gps), I refer to the documentation from CERN on General Particle source stated that "G4GeneralParticleSource is used exactly the same way as G4ParticleGun in a Geant4 application. In existing applications, one can simply change your PrimaryGeneratorAction by globally replacing G4ParticleGun with G4GeneralParticleSource." The only occurrences of G4ParticleGun that where found were within the files PrimaryGeneratorAction.cc and PrimaryGeneratorAction.hh files (See SUGGESTED READINGS). There was; however, one function that I had to remove its *int* from and two lines that needed to be commented out for a successful re-'make' of the example after making our switch. The lines committed in the PrimaryGeneratorAction.cc file where

- fParticleGun->SetParticleEnergy(500*keV);
- fParticleGun->SetParticleMomentumDirection(G4ThreeVector(1.,0.,0.));

and the int that we had to delete was in the line

- fParticleGun = new G4GeneralParticleSource(1);

Also, from the PrimaryGeneratorAction.cc file, it is worth noting that 'make' gave me a clear indication as to what was causing a 'make' error and where it was located. It also gave possible fixes that can serve as a good guideline. After making these changes, I was able to change the commands in the macro file. See below for sample macro files representative of the before and after of the Gun and GPS systems.

```
####################################
# For Mo at 1 MeV using the original GUN commands
#
/control/verbose 2
/run/verbose 2
#
# Set to one Homogeneous slab
/testem/det/setNbOfAbsor 1
# Make slab one Material Molybdenum 1 cm thick
/testem/det/setAbsor 1 G4_Mo 1 cm
# Set other lengths to 1 cm
/testem/det/setSizeYZ 1 cm
#
# Use standard em physics option
/testem/phys/addPhysics emstandard_opt3
#
/run/initialize
#
# Set Range cuts electrons and gammas (Default from Sandria macro)
/run/setCut 1 mm
#
/testem/gun/setDefault
# Set particles gun to use electrons
```

```
/gun/particle e-
# set number of particles to be 'fired'
/gun/number 1
# Set electron energy
/gun/energy 1 MeV
#
# Set root file name
/analysis/setFileName First_Carrier_Geometry_Mo_1MeV_Gun.root
#Normalized Edep vs lenght Graph, graph 8, 100 bins, from 0 to 1, no units
/analysis/h1/set 8 100 0. 1.0 none
#
/run/printProgress 10000
#
# Total number of particles ran
/run/beamOn 1000000
#####################################
#####################################
# For Mo at 1 MeV using the mew GPS commands
#
/control/verbose 2
/run/verbose 2
#
# Set to one Homogeneous slab
/testem/det/setNbOfAbsor 1
# Make slab one Material Molybdenum 1 cm thick
/testem/det/setAbsor 1 G4_Mo 1 cm
# Set other lengths to 1 cm
```

```
/testem/det/setSizeYZ 1 cm
#
# Use standard em physics option
/testem/phys/addPhysics emstandard_opt3
#
/run/initialize
#
# Set Range cuts electrons and gammas (Default from Sandria macro)
/run/setCut 1 mm
#
/testem/gun/setDefault
# Set gps to use electrons
/gps/particle e-
# Sets the source positional distribution type /gps/pos/type Beam
# Sets the center co-ordinates of the source /gps/position -.5 0. 0. cm
# Sets the energy distribution type /gps/ene/type Mono
# Sets the momentum direction /gps/direction 1. 0. 0.
# set number of particles to be 'fired'
/gun/number 1
# Set electron energy
/gun/energy 1 MeV
#
# Set root file name
/analysis/setFileName First_Carrier_Geometry_Mo_1MeV_GPS.root
#Normalized Edep vs lenght Graph, graph 8, 100 bins, from 0 to 1, no units
/analysis/h1/set 8 100 0. 1.0 none
#
```

```
        /run/printProgress 10000
        #
        # Total number of particles ran
        /run/beamOn 1000000
#####################################
```

The section in the GPS macro, where the set cut commands exist, was only used in water, as shown in Figure 5.6 and 5.7. Initally, the GPS code was fist ran for Molybdenum and Water at 1 MeV to verify that using GPS was no different from using the Gun command. It was then later applied to all runs. The macro file was then run again for Water at all energy values with the addition of the range cuts to secondary electrons and gammas, but only 1 MeV and 10 MeV yielded viable results.

In the Carrier paper, he stated that for the Be and Mo geometries, as well as the water geometries: "The production thresholds are set to 1 keV for gammas and 10 keV for electrons". Replicating this was not as it had seemed because GEANT4 dose not do energy production cuts. Instead, "Each particle has a suggested cut in range (which is converted to energy for all materials)," Guesswork was used to determine the energy cuts for both particles until it was reported (in the output of section B) that their energies were close to the desired values.

Looking at Figure 5.6, it would seem that this addition has allowed the peak of our Water sample to shift over to a position more representative of Carrier's work. We also see a better trend match at the start of the 10 MeV Water run. However, there is still a significant disparity between our energy deposition and that of Carrier. Furthermore, it was found to be wildly inaccurate for other geometries.

The inaccuracies found when comparing Gun to GPS or either to Carrier most likely come from missing constraints. The only two obvious constraints that I was not able to handle where the voxals and the setting of max step size to 1% of the initial CSDA. It is unclear what role the voxels play or how to adjust the Max step to be 1% of the CSDA.

The only alteration made to the code was again in the PrimaryGeneratorAction.cc file. Where this file affects the G4VUserPrimaryGeneratorAction core class, in which the user defines all the details of initial particles. [14] The performance difference in original G4ParticleGun compared to G4GeneralParticleSource seem then to lie in an unknown discrepancy in the initial event state definition, i.e., number, energy, direction, type of particles, etc.,.

Referring back to the class categories found in Figure 3.1, we can narrow down the possibilities of where the discrepancies are occurring. The global category, being responsible for units, constraints, numerics, and random number handling, is likely not the problem as none of the commands or altered code affected any of these domains. Similarly, Geometry, Event and Run categories can be eliminated in the same manner.[14] This leaves Material, Particle, Geometry, Track, Process, and Tracking as our possible problem sources. If one was to continue trying to exactly replicate the results of Carrier, it would be best to look at the code and commands that fall under these class categories to see what discrepancies there may be between the Gun and GPS runs, and either run with the set up of Carrier. However, given that they wrote there code from scratch, this seems like an exercise in reverse engineering more than in running GEANT4.

Unknown constraints aside, we were able to show that we could reasonably replicate previously obtained results. Moreover, we showed that GEANT4 is relatively ‘plug and play’, allowing a new user to become moderately skilled in a short time. With a more advanced (C++) user could write there own code to either replicate existing work or to meet there own goals.

CHAPTER 6

DISCUSSION

Future Work

The possible applications of GEANT4 are numerous. The new user may decide to continue the work done here. In which case, the next step would be to narrow down where I have made errors in my use of GEANT4. More realistically, however, the user would want to do something else internally with GEANT4. However, there are still a few bits of general advice I could give having used GEANT4. The first bit of advice would be to understand C++. I feel that I lack a clear understanding of how C++ code works and that this has hindered both the quality and efficiency of my work. Secondly, try and find some basic examples that can accomplish your goals and practice them before attempting to write or alter code. Lastly, develop good documentation skills. Save frequently and using a subject-date-time stamp in the file name. This will save a lot of frustration and time and will help to organize things.

As far as current possible areas of study using GEANT4 go, the DNA and Medical applications are where I would start as these are likely to be newer and more marketable.

Conclusion

GEANT4 is a powerful, user-friendly toolkit for simulating the passage of particles through matter. It has a wide variety of preinstalled applications that are easy enough to use for the first time user when they use a good GUI/Visualizer for their application. The frequent use of GitLab and the README's will help any user greatly, as will the frequent use of Forms.

In this thesis, I was able to build an understanding of what GEANT4 is, as well as how it works. I was then able to show that the new user should be able to install GEANT4 with a few extra features with relative ease. The user then can select from a multitude of preinstalled examples to run. In this thesis, the TestEM11 example was chosen. The results from my attempted use were then compared to that of the previously published work of Carrier to verify my ability to use GEANT4. The results were mixed, with most discrepancies seeming too stem more from a difference in code (TestEm11 vs. Carrier) than from 'misuse' of GEANT4, as GEANT4 runs using commands. As long as a new user can familiarize themselves with the commands, they should have no issue running GEANT4. However, if they want to run original research, they will need proficient knowledge of C++ and of the GEANT4 structure to obtain reliable results.

APPENDIX OF USEFUL LINKS

Geant4 Installation Guide: `http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/InstallationGuide/html/index.html`

Geant4 TestEm11 README: `https://gitlab.cern.ch/geant4/geant4/blob/master/examples/extended/electromagnetic/TestEm11/README`

Geant4 OpenGL Prerequisite: `https://www.opengl.org/`

Geant4 Qt prerequisite: `https://www.qt.io/download`

APPENDIX OF SUGGESTED READINGS

Final Run Macro File

Altered PrimaryGeneratorAction.hh

Unaltered PrimaryGeneratorAction.hh

Altered PrimaryGeneratorAction.cc

Unaltered PrimaryGeneratorAction.cc

Extricable ROOT file for multi-graph construction using multiple file.root and file.csv

For all file.root, file.txt, file.C, file.csv, file.png generated from each geometry tested, ask Professor Zhu for the USB containing GEANT4 singularity. Files can be found under the directory titled ALL

REFERENCE LIST

[1]*Geant4 scope of application — IntroductionToGeant4 10.5 documentation*, http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/IntroductionToGeant4/html/IntroductionToG4.html (visited on 10/17/2019).

[2]J. F. Carrier, L. Archambault, L. Beaulieu, and R. Roy, "Validation of GEANT4, an object-oriented monte carlo toolkit, for simulations in medical physics", Medical Physics **31**, 484–492 (2004).

[3]R. Brun, R. Hagelberg, J. C. Lassalle, and M. Hansroul, *Simulation program for particle physics experiments, GEANT : user guide and reference manual*, CERN Document Server, (1978) https://cds.cern.ch/record/118715 (visited on 10/15/2019).

[4]*Introduction to geant4*, https://geant4.web.cern.ch/sites/geant4.web.cern.ch/files/geant4/support/training/CSC2000/CSCG4.pdf (visited on 11/20/2019).

[5]I. V. Mavragani, Z. Nikitaki, S. A. Kalospyros, and A. G. Georgakilas, "Ionizing radiation and complex DNA damage: from prediction to detection challenges and biological significance", Cancers **11**, 1789 (2019).

[6]Z. Francis, G. Montarou, S. Incerti, M. Bernal, and S. A. Zein, "A simulation study of gold nanoparticles localisation effects on radiation enhancement at the mitochondrion scale", Physica Medica **67**, 148–154 (2019).

[7]P. Shamshiri, G. Forozani, and A. Zabihi, "An investigation of the physics mechanism based on DNA damage produced by protons and alpha particles in a realistic DNA model", Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms **454**, 40–44 (2019).

[8]Z. Wu, H. Sun, Y. Ma, G. Liu, H. Zhao, J. Lu, and Y. Hu, "Radiation dose simulation of a water phantom in the cabin of an interplanetary spacecraft", Nuclear Inst. and Methods in Physics Research, B **462**, 62–67 (2020).

[9]M. Yano, F. Araki, and T. Ohno, "Geant4 monte carlo investigation of the magnetic field effect on dose distributions in low-density regions in magnetic resonance image-guided radiation therapy", Physica Medica **68**, 17–34 (2019).

[10]S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, F. Behner, L. Bellagamba, J. Boudreau, L. Broglia, A. Brunengo, H. Burkhardt, S. Chauvie, J. Chuma, R. Chytracek, G. Cooperman, G. Cosmo, P. Degtyarenko, A. Dell'Acqua, G. Depaola, D. Dietrich, R. Enami, A. Feliciello, C. Ferguson, H. Fesefeldt, G. Folger, F. Foppiano, A. Forti, S. Garelli, S. Giani, R. Giannitrapani, D. Gibin, J. Gómez Cadenas, I. González, G. Gracia Abril, G. Greeniaus, W. Greiner,

V. Grichine, A. Grossheim, S. Guatelli, P. Gumplinger, R. Hamatsu, K. Hashimoto, H. Hasui, A. Heikkinen, A. Howard, V. Ivanchenko, A. Johnson, F. Jones, J. Kallenbach, N. Kanaya, M. Kawabata, Y. Kawabata, M. Kawaguti, S. Kelner, P. Kent, A. Kimura, T. Kodama, R. Kokoulin, M. Kossov, H. Kurashige, E. Lamanna, T. Lampén, V. Lara, V. Lefebure, F. Lei, M. Liendl, W. Lockman, F. Longo, S. Magni, M. Maire, E. Medernach, K. Minamimoto, P. Mora de Freitas, Y. Morita, K. Murakami, M. Nagamatu, R. Nartallo, P. Nieminen, T. Nishimura, K. Ohtsubo, M. Okamura, S. O'Neale, Y. Oohata, K. Paech, J. Perl, A. Pfeiffer, M. Pia, F. Ranjard, A. Rybin, S. Sadilov, E. Di Salvo, G. Santin, T. Sasaki, N. Savvas, Y. Sawada, S. Scherer, S. Sei, V. Sirotenko, D. Smith, N. Starkov, H. Stoecker, J. Sulkimo, M. Takahata, S. Tanaka, E. Tcherniaev, E. Safai Tehrani, M. Tropeano, P. Truscott, H. Uno, L. Urban, P. Urban, M. Verderi, A. Walkden, W. Wander, H. Weber, J. Wellisch, T. Wenaus, D. Williams, D. Wright, T. Yamada, H. Yoshida, and D. Zschiesche, "Geant4—a simulation toolkit", Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250–303 (2003).

[11]*Building and installing — geant4 installation guide 10.5 documentation*, http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/InstallationGuide/html/installguide.html (visited on 10/21/2019).

[12]V. Gite, *Check how many CPUs are there in linux system*, nixCraft, (Oct. 15, 2018) https://www.cyberciti.biz/faq/check-how-many-cpus-are-there-in-linux-system/ (visited on 10/21/2019).

[13]*Examples/extended/electromagnetic/TestEm11 · master · geant4 / geant4*, GitLab, https://gitlab.cern.ch/geant4/geant4/tree/master/examples/extended/electromagnetic/TestEm11 (visited on 10/22/2019).

[14]https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwjs2Ivv1IXmAhUEeKwKHVWHBUcQFjABegQICxAE&url=https%3A%2F%2Findico.in2p3.fr%2Fevent%2F147%2Fcontributions%2F19459%2Fattachments%2F15853%2F19456%2FGeant4-S.Incerti.doc&usg=AOvVaw3Z6uq8MpHeMONXH_1P9jKI (visited on 11/25/2019).

VITA

Xavier Brock graduated Cameron R-1 High-school in Missouri, class of 2013. After a year working on an oil pipeline, he enrolled in the University of Missouri - Kansas city, where he obtain a BS in Physics in three years. He then began his Masters in Physics at the same institution, graduating in the fall of 2019.