Mu2e Experiment: Simulation of the Production Target Design

Albert Szewczyk

Advisor: Michael Hedges, Target Systems Department, Accelerator Division

Summer 2024

Abstract:

The Mu2e experiment will search for coherent, neutrinoless conversion of muons to elections in an aluminum stopping target. A simulation program is used to experiment and design the production target of the Mu2e experiment. The module uses programing languages and libraries to work. The program called the Offline uses C++ as the programing language in the module. To construct the simulation, the Geant4 library constructs shapes and uses them in the simulation program, ROOT. The offline program simplifies the creation of the tube by organizing them into parameter objects and nesting them with other variables required to construct the tube-like center point, material, or if it is allowed to conflict with other solids. The parameter varies for every solid tube. The geometry file uploads the parameters by declaring and organizing the parameters in C++ using the configuration tools. The author describes the development and the future of this project.

Acknowledgements

I would like to express my gratitude to Michael Hedges for offering me the opportunity to participate in the CCI program and for assisting me in the process. I have always appreciated him mentoring me throughout my time for the duration of my project. I also appreciate the collaborators on my project including Kevin Lynch, Kateryna Havryshchuk, and Madeline Bloomer for being significant contributors and collaborators on my project. Thank you for guiding me throughout the time I had during this internship.

Table of Contents

Background:

The Mu2e experiment will search for discoveries beyond the standard model. Specifically, it will search for charged lepton flavor violation (CLFV) with discovery potential. In theory, CLFV must occur through neutrino loop effects. However, the probability of this occurrence is small. The probability of a CLFV free muon decay to an electron plus a photon is less than 10^{-54}. Therefore, if this decay or any CLFV signatures are observed, then it would require physics beyond the standard model.

The Fermilab accelerator complex will deliver 8 GeV protons to the Mu2e experiment. The protons strike the tungsten production target, which is supported by a structure resembling a bicycle wheel. Proton interactions with atoms in the production target produce pions, which decay to muons. The muons will travel in a solenoidal field onto the aluminum stopping target. The stopped muons will then interact with the nuclei of the stopping target inside the detector solenoid. The resulting electrons are emitted from the stopping target and are detected by the Mu2e experiment.

Introduction to the simulation program:

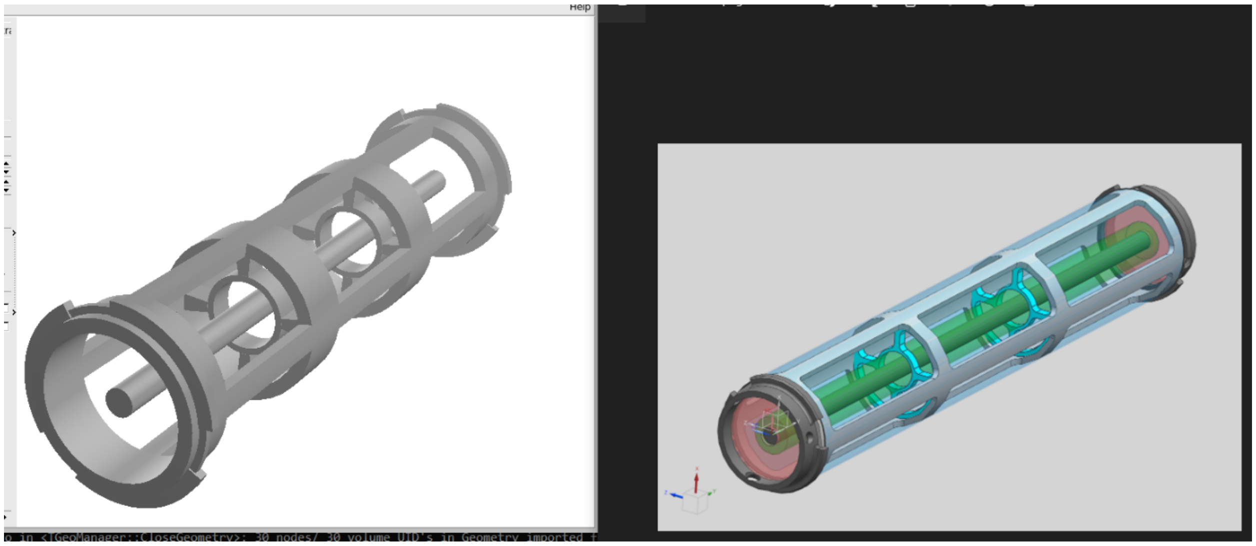
The Mu2e experiment uses Geant4 within the Mu2e Offline framework to simulate the production target. Typically, a unique C++ module is used to construct a given shape, or "geometry" in the simulation framework. As a result, every iteration of the production target requires multiple geometry variables, which makes simulating alternative designs very costly. Instead, we found that target designs can be very well modeled as multiple cylindrical tubes of varying lengths, thicknesses, radiuses, and phi-spans. We then developed a module that can dynamically build arbitrary target designs, approximated as cylinders, without a user needing to supply custom C++ code their design. The module loops through vectors of parameters for each target feature declared in a single geometry file. For example, instead of declaring parameters in multiple variables, and sometimes multiple files, vectors of parameters in a single file construct multiple cylindrical object. To construct a cylinder in this module, the outer and inner radiuses are necessary along with the height, phi-span, and material.

Conflict with 2 Dimensional Vectors and Results:

However, a conflict occurred as another parameter was necessary for getting to the center point of the tube. The center point is a set of coordinates with a number for each position that a solid will be placed (the x, y, and z directions). A vector is declared for the set of numbers, and the vector would translate the solid in the simulation program. However, the Mu2e Offline framework did not support taking in two dimensional vectors from geometry files. There were two options available to resolve this issue. The first option was creating an iterator in the file reader to take in two dimensional vectors. But the module was not reading them properly. The other option was to take in the vector as a vector string, and traverse through the points. This resulted in core dumping errors because the algorithm did not traverse through the full algorithm.

The coordinates were placed outside the logical volume (the volume in which tubes were allowed to be simulated). However, the geometry file had the span parameters in degrees. The program had to be changed to convert degrees to radians. Once that was done, the central cylinder of the tube was successfully constructed in Geant4, exported to GDML, and then visualized in the ROOT program. Following the successful construction of the tube, the dimensions of the tube were changed by adding more dimensions to the tube by polishing the outside structure.

Figure 1: Constructed tube design in ROOT on the left compared to the expected design from using CAD in the right.



```
o in <TGeoManager::CloseGeometry>: 30 nodes/ 30 volume UID's in Geometry imported f
```

Formation of the Wire Mesh:

Despite the successful construction of the tube, a wire mesh was needed inside of the tube. There are several options to achieving this. One option is creating multiple boxes of wires and organizing them to create a mesh. The other two options were using polycones or using tubes to create the mesh. In the tube, the goal was to create a mesh between the innermost tube and the outer tube holders. The box option would not have worked because of the change in the center point that was needed each time. The algorithm to create a mesh wire from boxes was not enough. Because of this, using polycones became the next option. The Geant4 application and developer handbook became a guide to determine how a mesh could be achievable. Theoretically, polycones can construct multiple extruded tubes. In Geant4, polycones are constructed with the following, an array of coordinate points, inner radius points, and outer radius points. It was theorized that the polycones could create a set of extruded tubes by setting some parts of the polycone having equal inner and outer radius.

After being unable to create the mesh with the polycone, it was experimented with creating multiple tubes in the program. During the process of changing into tubes, the modified algorithm contained multiple tubes in a separate class. Each set was for each dimension direction. One loop would create a set of hollow tubes. Another loop would create wires from the center of the mesh. The third set of tubes would connect the tubes from the first two sets all together in the form of a small wire. The loops were nested three times to change the parameters. The first set of tubes contained two nested loops. The first loop was to increase the radius for the next tube to be constructed. The outer loop would change the center of the tube to add the next set of tube wires. In the second and third nested loops, the inner loop would change the starting span of the tube. The total span remained constant for these tubes. The middle loop would change the radius and the outer loop would change the center of the tube. The loops have the same algorithm except for the third loop having a different starting center point to avoid overlapping with the tubes from the first set. In the method, all the parameters of every tube and its center point are stored in vectors and are retrieved from the tube construction program file. The file loops through the number of tubes based on the parameters taken from the geometry file. The loop would use a method in the offline called nest Tubs to nest all the parameters with the core tube parameters so that the information can be used to construct the solid in Geant4. The original program would take in a set of parameters to create tubes rather than creating a set of tubes from a few parameters.

Creation of the wire mesh with G4 Union Solid:

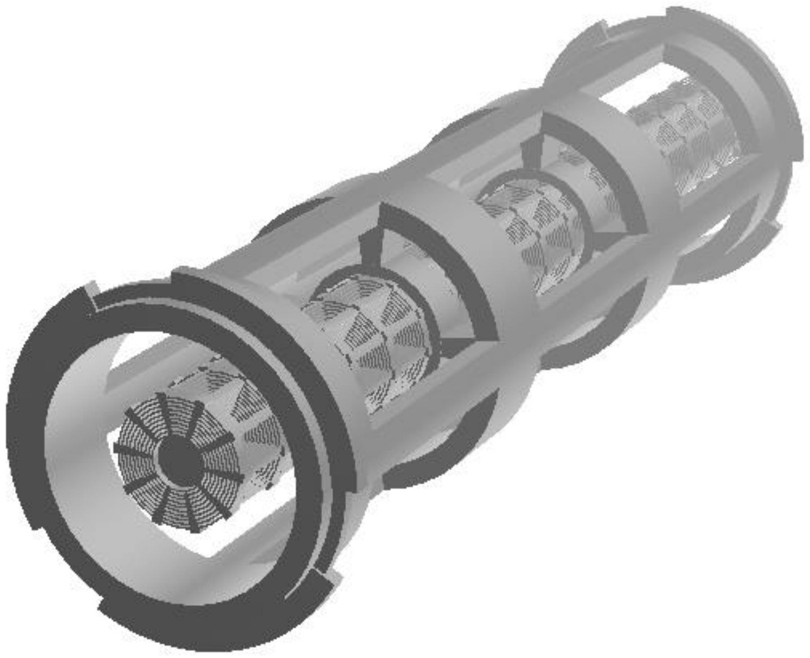
After the tubes of the mesh were analyzed, the algorithm in the wire mesh had been reduced to only use two nested loops. The horizontal loop had been unchanged. The other two loops were combined to construct the wires to connect the tube. As a result, less memory was allocated for constructing the tubes. This had allowed more wires to be constructed in the mesh, allowing the mesh in the simulation to be more compatible with the current mesh design. Eventually, G4 Union Solid worked with a limit on the quantity of wires in the mesh that could be used, indicating that using Union Solid to create one mesh was possible, but for a limited number of wires.

Future Progress:

In the future, there will be more progress that will be made on this tube. The program was able to use the resources in the Mu2e Offline library and use them to construct the tubes. However, the Offline appears to support certain shapes based on the necessities for previous parts of Mu2e to have it constructed. An analysis and collaboration with the developers on the TgtTube project may be necessary to determine how union solid is utilized and executed. The creation of algorithms on creating multiple tubes will also have an impact on probable future experiments. Simulations for future experiments can create meshes of wires using the wire mesh algorithms or modify them based on the shapes of the tube that will be necessary for the tube project. The creation of the wire meshes in simulations will require more experimentation needed. In addition, the issue with G4MultiUnion will need to be supported on root to allow more experimentation with the wire mesh.

The gdml files constructed while running the code will also be used in the future. The construction of the production tube was one aspect of the simulation. The other aspect is simulating the launching of protons into the simulation. The physics software FLUKA will take the gdml file and for use in simulating the collision of protons with the tube.

Figure 2: Constructed tube in ROOT with the wire meshes.



Closing Statement:

Constructing the production target beam for simulating the Mu2e experiment has been able to be performed in Geant4 despite the wire mesh being a work in progress. With modifications to the program constructing the production target tube, it is possible to simulate the production target tube with collision events occurring in the simulation program. Each tube can have their parameters, material, center, or visibility modified without having to recompile the simulation program, reducing the time needed for every trial on the simulation of the production target beam tube. The wire mesh will need room for improvement in the program, but the construction of the mesh does not setback the program to conduct simulation experiments with the production target.

Appendix:

Information on the parameters of the Geant4 tubes:

https://docs.google.com/spreadsheets/d/1xg2lLKN8P9IW0-wUKlJNKj_bDYg5XNfb2Y8KT_6XUo8/edit?usp=sharing

References and Resources

https://github.com/Mu2e/

https://github.com/abshev19/prodTgtG4/tree/MultiCylinder

https://indico.fnal.gov/event/64676/contributions/291035/attachments/179079/244374/2024-06-sist.pdf

https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/fo/BookForApplicationDevelopers.pdf

Project Contributors and Participants

Name	Role	Affiliation
Albert Szewczyk	CCI Intern	Harper College and University of Illinois Urbana Champaign
Michael Hedges	Scientist, Target Systems Department	Fermilab
Kateryna Havryshchuk	SIST Intern	Amherst College
Kevin Lynch	Scientist, Target Systems Department Head.	Fermilab
Madeline Bloomer	SIST Intern	Emory University

Software Used

Ubuntu

Root

Geant4

C++ Programing Language

FHiCL

Facilities

Mu2eGVPM06 andMu2eGVPM07 Linux computers – construction of the C++ files, building and running the programs were performed on these computers.

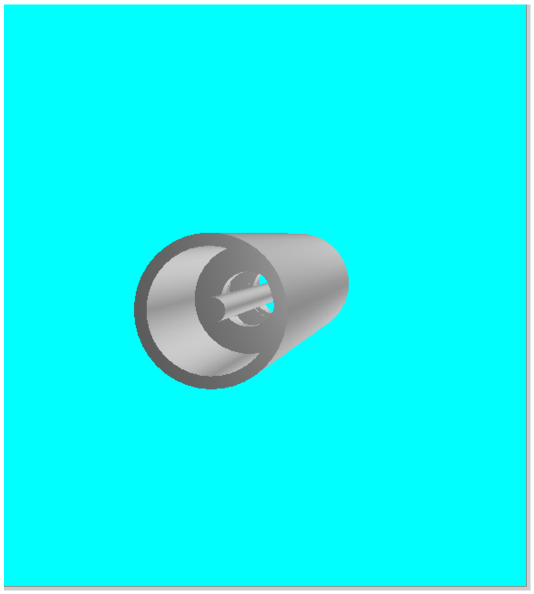
LINAC West Booster tower building- Office area

https://github.com/abshev19/prodTgtG4/tree/MultiCylinder

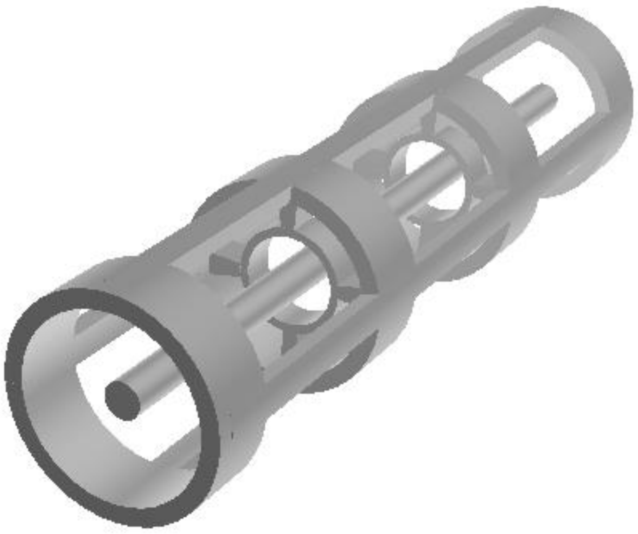
Notable Outcomes:

Attempts in creating the production target tube on ROOT:

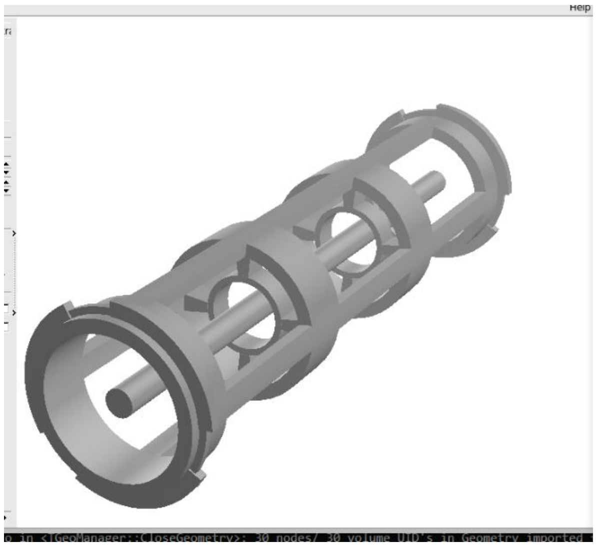
First attempt



Constructed tube.

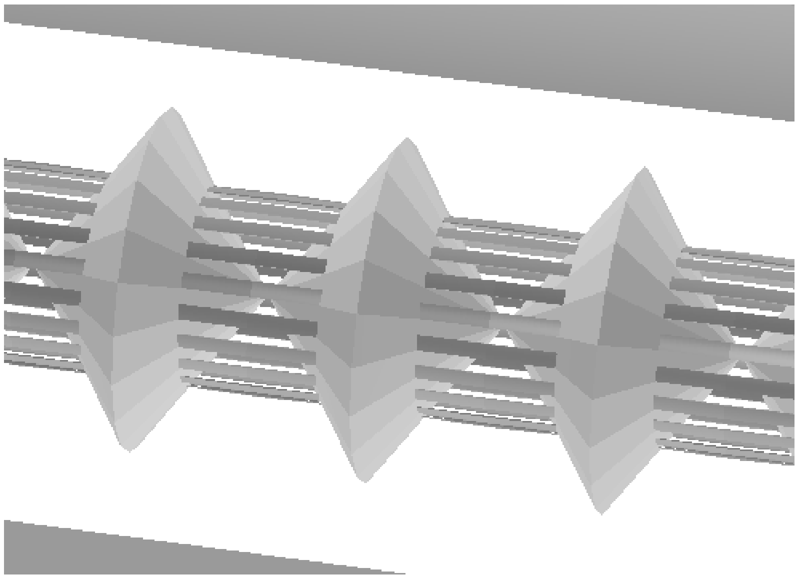


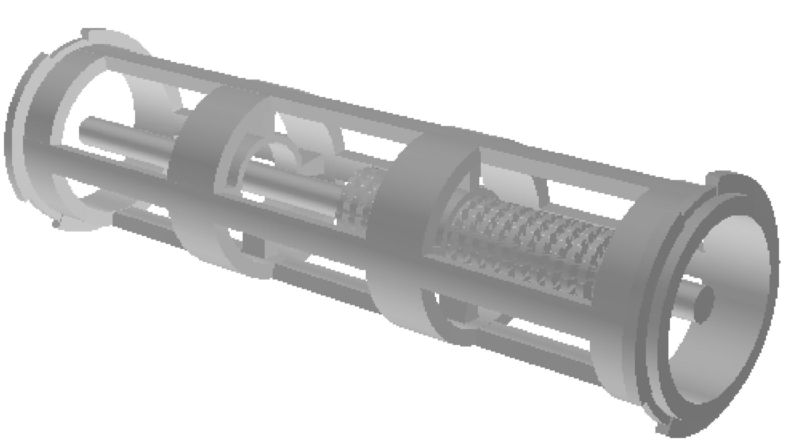
Adjusted Constructed Tube



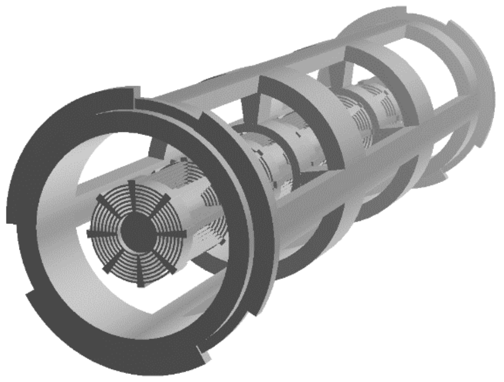
```
o in <TGeoManager::CloseGeometry>: 30 nodes/ 30 volume UID's in Geometry imported
```

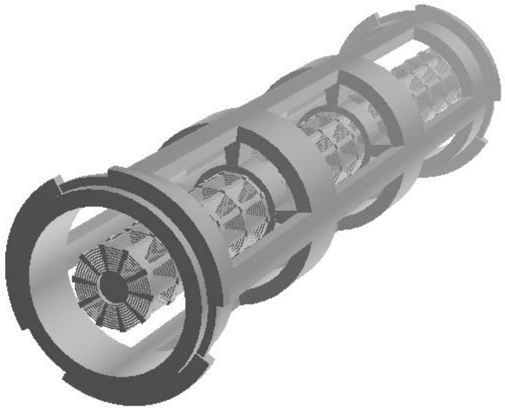
Tube with Attempted Polycone Wire Mesh

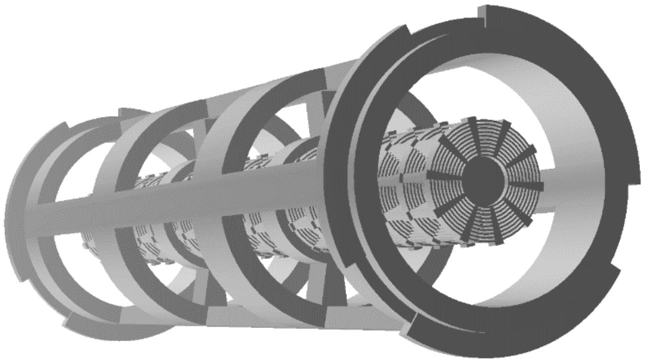




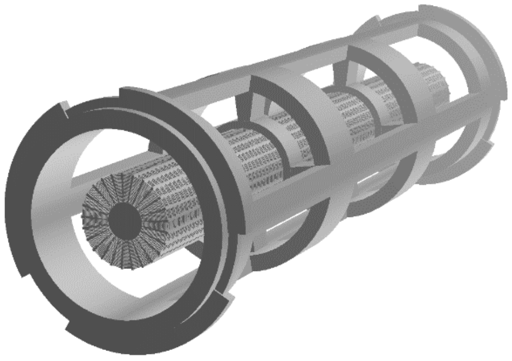
Tube with tube wire mesh (without G4 Union Solid)

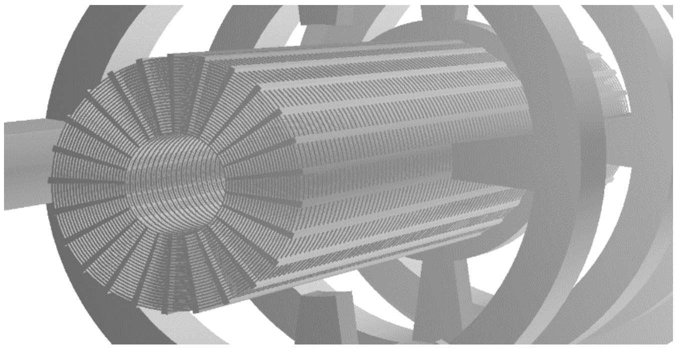




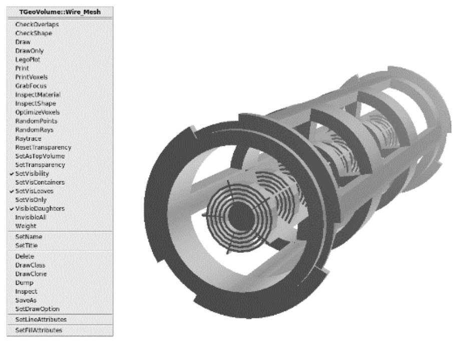


Production Target with more wires in the wire mesh. (without G4 Union Solid)





Construction of the production tube with the limit on the quantity of mesh wires for G4 Union Solid. Constructed using Union Solid.


TGeoVolume::Wire_Mesh
CheckOverlaps
CheckShape
Draw
DrawOnly
LegoPlot
Print
PrintVoxels
GrabFocus
InspectMaterial
InspectShape
OptimizeVoxels
RandomPoints
RandomRays
Raytrace
ResetTransparency
SetAsTopVolume
SetTransparency
✓ SetVisibility
SetVisContainers
✓ SetVisLeaves
SetVisOnly
✓ VisibleDaughters
InvisibleAll
Weight
SetName
SetTitle
Delete
DrawClass
DrawClone
Dump
Inspect
SaveAs
SetDrawOption
SetLineAttributes
SetFillAttributes