**Cannon 2 Report**

**Kara’s Writeup:**

**Vector.py**

The Vector3D class was by far the most tedious part of the assignment for me. Not only was it mentally frustrating that I had to restart the Cannon 1 assignment since I switched from C# to python, but python was my least favorite programming language to begin with. However, throughout these couple weeks this assignment has made me realize how simplistic python is when implementing certain programs compared to the other languages, I felt more comfortable with. The first thing I did when implementing this project was make the Vector3d class. It took the most time because it took me awhile to understand the syntax when converting the C++ program to python. For example, the keyword “self” was new to me, yet it’s needed to access the attributes and methods of the class. Making the constructors was also different for me because using \_*init*\_ was needed to initialize the objects state. Another difficulty I struggled with was trying to figure out how to use operator overloading in python. Unlike C++ and C# where operator overloading is defined before the opening of the {} of a function, python declares it inside of the function. It was a bit odd at first, but once I got used to the syntax and researched more I realized it’s a lot simpler than implementing it in C++ or C#. In fact, I found it a lot more readable and easier to write. I made some mistakes at first when implementing the operator overloading functions such as passing in \_x, \_y, \_z or returning the wrong value, but with the collaboration of both me and Jacky it was easily fixed.

**Basic.py**

When first looking at the C++ Basic class I thought it looked very simplistic and straight forward. However, after I was done implementing the Basic.py class I realized that when comparing it to the C++ Basic class, the C++ class started to look confusing instead! The first thing I did was analyze the headers and functions of the C++ class. I saw that it used trigonometric functions and saw that it had several math computational headers. Thus, it seemed most obvious to research what import is needed in python to have access to trigonometric functions. To my finding it was the import math. Once I figured out that header, implementing this class was probably the easiest one to do.

**Ball.py**

This class wasn’t too hard to implement as well. Some major differences was instead of calling the operator overloading operators like EX: \*= in C++ we would just call the function name itself “smul”(stands for scalar multiply). For me personally it makes more sense calling the function you want to use by its name because typically when you want to call a function you just call the name of the function, however it C++ it causes the actual operator needed to be overloaded, so, personally I liked the syntax better in python in this case. Some minor differences were the syntax such as replacing drand48() with just random, however that wasn’t too much of a issue. Some nice features that I really enjoyed with python is the lack of headers. C++ has a lot more files to include and sometimes when you work with bigger projects it’s easy to forget to include headers and files.

**Main.py**

The first thing when implementing the main file was analyze how it was done in C++. Not going to lie this file was the most overwhelming file at first for me. It was probably due to the fact that I haven’t worked with OpenGl in either language. However, once I researched more and looked at OpenGl Python examples I saw that OpenGl was almost exactly as the C++ code and most if not all the code regarding to the OpenGl import was copy and pasted from the original C++ code. However, a difference that I did notice was the creation of Nodes. When first learning OOP, Nodes and pointers were my least favorite thing to learn. However, pythons’ syntax for some reason is a lot easier to declare a node and pass a pointer to another variable. Another difficulty I encountered was how Process the Keys to shoot and quit. While looking at more examples I found out that Python uses “utf-8” do access those keys while as in C++ you can just declare a number or char without needing to call a specific decoder. To come to a conclusion the .main file was overwhelming, but majority of it was a copy and paste for the OpenGl portions.

**Conclusion:**

Overall, this program was very tedious and took a lot of research from both of us. However, surprisingly this assignment made me realize the benefits of knowing multiple languages. If I was given this assignment with no code to help me. I would much rather use python instead of C++, due to the fact of how simple and compact python is to write in. Once I got beyond the syntactic urges I had due to the habit or programming in C languages, python’s writability was much more simplistic and easier to follow along compared to the C++ code. In fact, I had to research a bit of the C++ code just because I didn’t fully understand the syntax.

**Work Cited:**

<https://www.geeksforgeeks.org/self-in-python-class/>

<https://www.geeksforgeeks.org/operator-overloading-in-python/>

<https://www.tutorialspoint.com/python/python_nodes.htm>

<https://www.geeksforgeeks.org/mathematical-functions-in-python-set-3-trigonometric-and-angular-functions/>

<https://www.geeksforgeeks.org/__init__-in-python/>

<https://butterflyofdream.wordpress.com/2016/04/27/pyopengl-keyboard-wont-respond/>

<https://dev.to/suvink/graphic-designing-using-opengl-and-python-beginners-1i4f>

**Jacky’s Writeup**

When working with Python on trying to get the OpenGL process to work, I had several troubles while trying to navigate the easiest way to duplicate the functionality of the sample program that was provided.  The mathematical portions throughout the code were some of the easiest parts to code, especially because Python is a straightforward programming language when it comes to mathematics. For some reason I had tried to add certain variables into the main functions of the code and had trouble navigating where errors were coming from since I’d have to keep fixing values for said variables, I ended up fixing this issue by using global variables, so that the variables could be accessed more easily. Some of the assignment operators for the math were hard to remember at first since they can be shorthand for an equation for example: x += 3 is equivalent to x = x + 3. Another issue I had when translating the code to Python from C++ was managing to work around some void functions, but I was able to solve that by using the def function on Python. The Target translation was one of the places where the def function was used to sub in for the void function in C++. It was just an easier option to define the given function and use if statements and for loops in order to achieve the same functionality of the sample program. The Target portion of the code is meant to give the ball a place/area to be shot at. Throughout the translations I made in the Vector3d I added the global variables like stated before and numerous def functions to help to give the program its entire functionality which is shooting the balls out of the cannon. As a whole the process of translating this program over to C++ was challenging and lengthy, it helped me to gain a better understanding of the Python language and the differences in operators and functions.