Homework 5 Report

Zack Fravel
010646947

Problem Statement:

The goal of this programming assignment was to implement in-line methods for class definitions provided for us. Specifically, we were tasked with implementing a distance method in our Point class, a normalize and dot method in our Vector class, a get_sample method in our Ray class, and finally a get_intersection method in our Sphere class. There are no inputs to the program, all testing is done through defining variables and using statements in main.

The main error handling required was for the get_intersection method in choosing between the different possible cases and negative directions. The program output is a series of print statements showing the answers to the calculations.

Design:

Since the class definitions were provided for us, there were not that many decisions to make design-wise. For all number variables I used floats and tried to keep the code as short as possible in the return methods. No data structures were really used other than primitive variables used in different calculation functions. The pros of this was it didn't take long to design and it is extendable.

```
Zacks-MacBook-Pro:hw5 Meryl$ ./classes

Test point class
p1 = (1,2,3)
p2 = (3,4,5)
dist12 = 3.4641
dist21 = 3.4641

Test vector class
v1 = (2,1,0)
v2 = (-1,1,0)
v2 = (-0.707107,0.707107,0)
v1dotv1 = 5
v1dotv2 = -0.707107
v2dotv2 = 1

Test ray class
r1(p1, v1) = (1,2,3) (0.894427,0.447214,0)
r2(p2, p1) = (3,4,5) (-0.57735,-0.57735,-0.57735)
r1(0.0) = (1,2,3)
r1(0.1) = (1.08944,2.04472,3)
r1(0.2) = (1.17889,2.08944,3)
r1(0.3) = (1.26833,2.13416,3)
r1(0.4) = (1.35777,2.17889,3)
r1(0.5) = (1.44721,2.22361,3)
r1(0.6) = (1.53666,2.26833,3)
r1(0.7) = (1.6261,2.31305,3)
r1(0.8) = (1.71554,2.35777,3)
r1(0.9) = (1.80498,2.40249,3)
r1(1.0) = (1.89443,2.44721,3)

Test sphere class
sphere = (0,0,0) 2
ray = (-3,0,0) (1,0,0)
        point = (-2,0,0)
        normal = (-1,0,0)
ray = (3,0,0) (1,0,0)
No intersection
```

Implementation:

The implementation process was fairly simple. I had already done the work needed for the dot, normalize, and distance methods in Homework 4 so I was able to reuse some code for that. The most difficult part of the program was the get_intersection method in the sphere class. I begin by defining all the different factors I'll need in my calculations and then based on the answer to the 'd' term on the wiki page I set my solutions after a conditional that checks if there's one solution, two solutions, or no solution. In total it took around 4 hours to complete the homework.

Testing:

I did my best to emulate the output provided to us for testing. I tested two points for the sphere and all the functions implemented. A sample output is provided. Everything works as expected.

Conclusions:

Overall, the programming project was a success! I can't think of much I would do differently next time since there weren't many different ways to do this. The overall result is now we can combine this with the Phong class from Homework 4 to create shaded and ray traced images.