Graphics Assignment 3

Raytracing

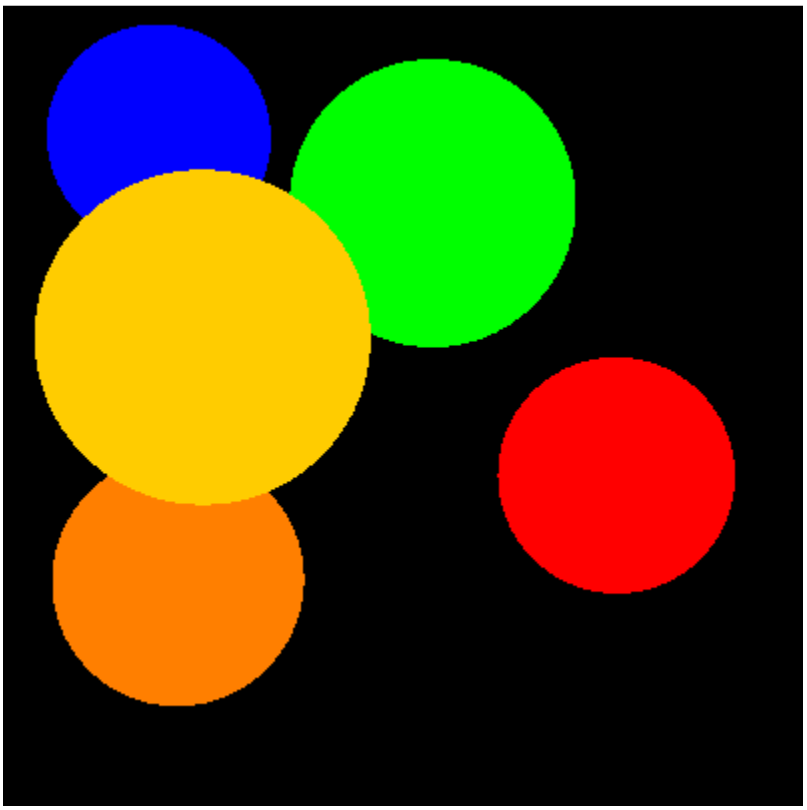
Anthony Di Donato 100517433

I, Anthony Di Donato, certify that this work is my own, submitted for CSCI 3090U in compliance with the UOIT Academic Integrity Policy.

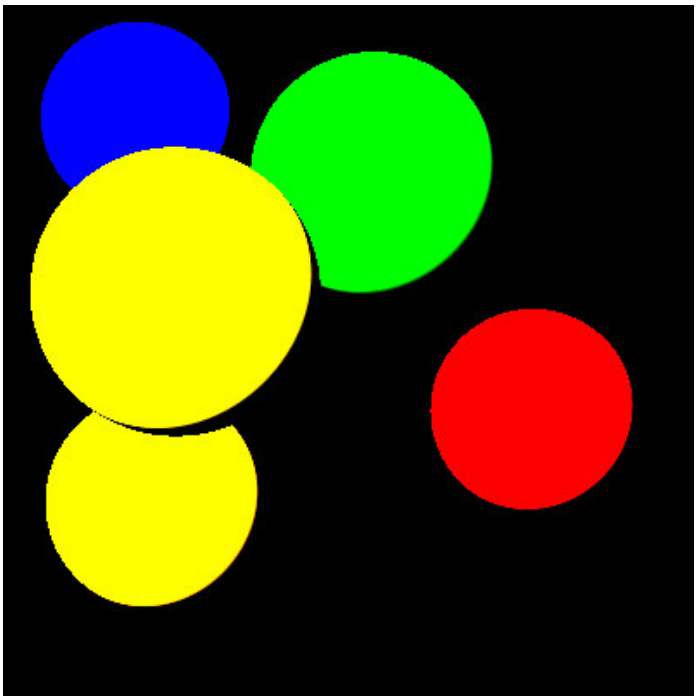
This was my favorite assignment so far. I think the structure and pacing were fair and I liked not having to use opengl for once.

Even though the pictures I created look (as far as I can tell) identical to the example pictures in the assignment, here they are:

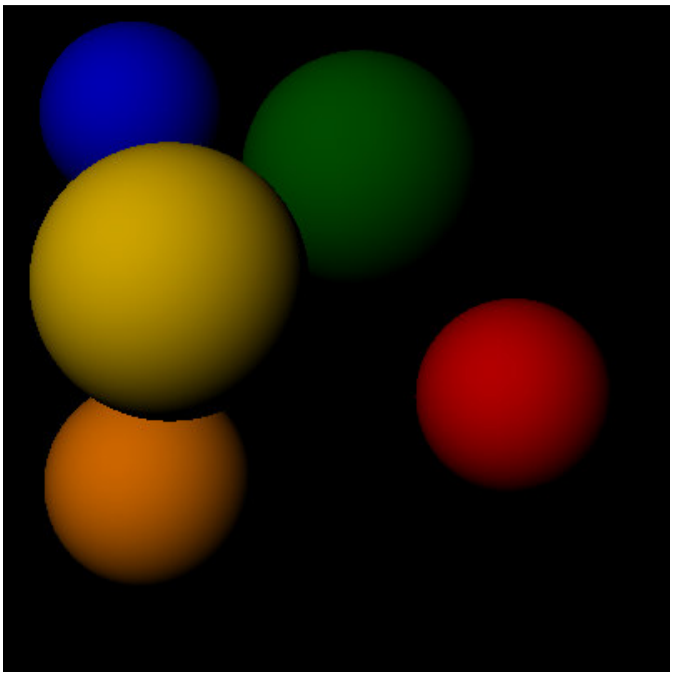
I also included a few other pictures that were produced along the way that don't look like the examples, but looked neat in their own ways.



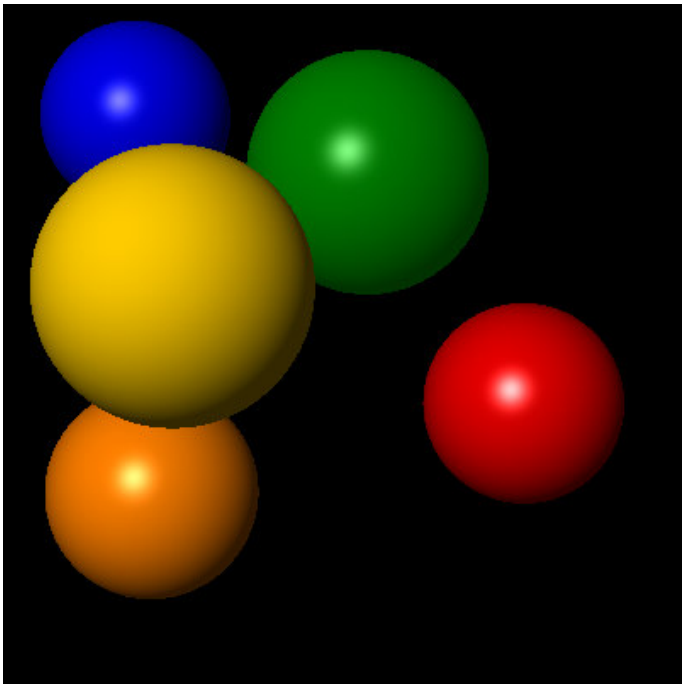
The first part, incredibly simple. Just draw the color of the object that is intersected. No multiplication depending on lighting, no lighting at all really, just the raw colors.



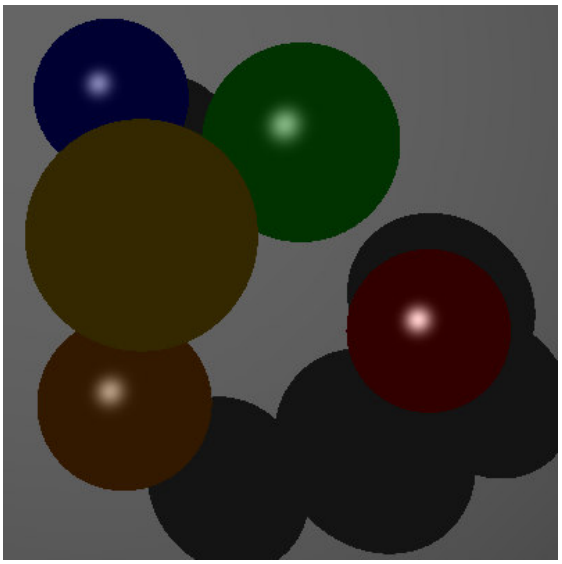
My first attempt at the basic diffuse lighting, it looked like this I believe because I didn't normalize the normal vector.

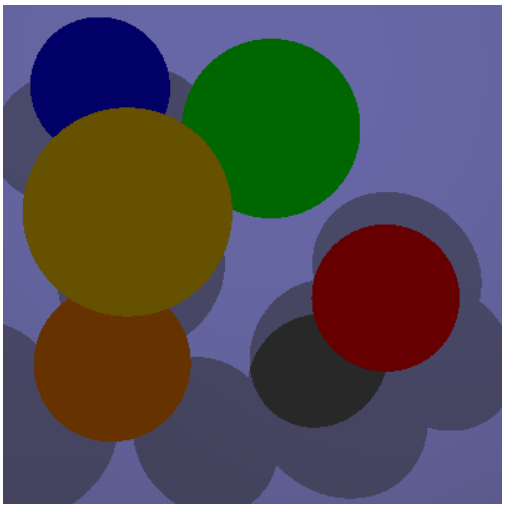


The successful diffuse lighting picture, not that difficult to implement, the only problem I had was trying to figure out what some of the components of the equation in the lecture slides were.



After implementing specular and ambient lighting. This was an easy progression from the diffuse part. I did change the color calculation here to keep separate values for each type of lighting and add them all together at the end.





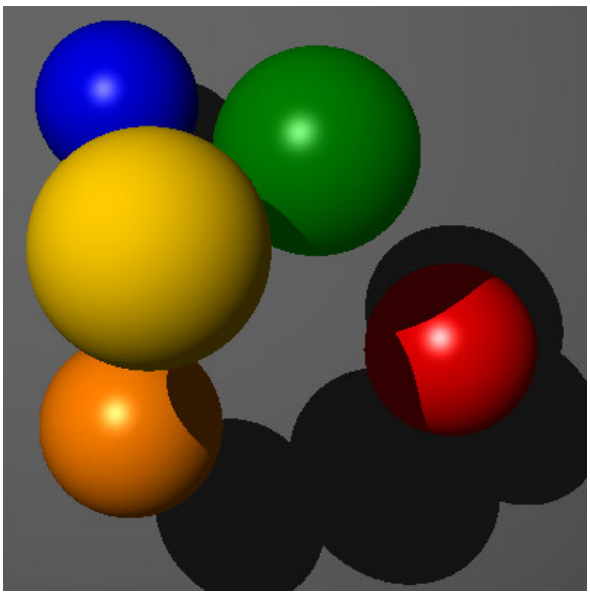
Both of these pictures demonstrate the initial problem I had with shadows. The problem arose due to the intersect method returning the minimum value, but not excluding negative values. This lead to all the spheres looking like they were in the shade because they all negatively intersected with the big grey sphere in the background when using a vector towards the light.

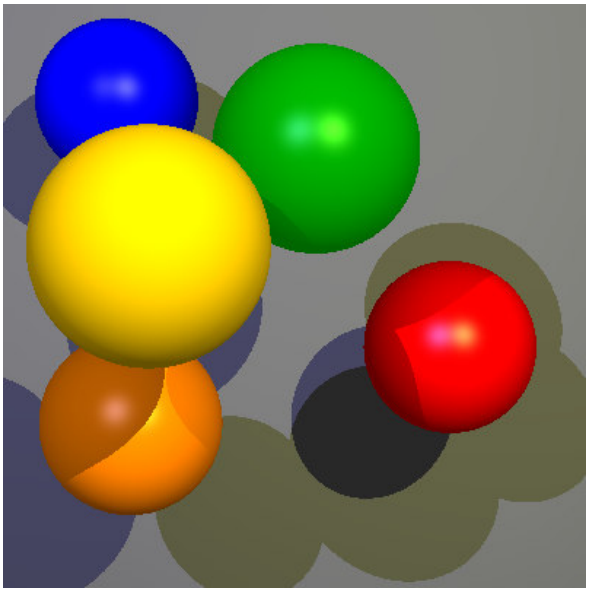
A ”band-aid” fix I used to temporarily fix this problem for this part was this block of code (which worked up until the refraction):

After setting t to the minimum

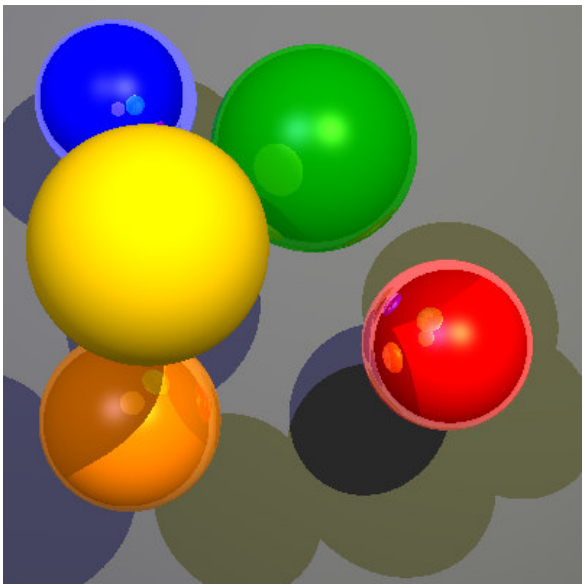
```
t = max(0.0, t);
	if (t == 0)
	{
		return Hit::NO_HIT();
	}
```

Which removed negative values… but also wouldn’t give a positive value in the same direction.

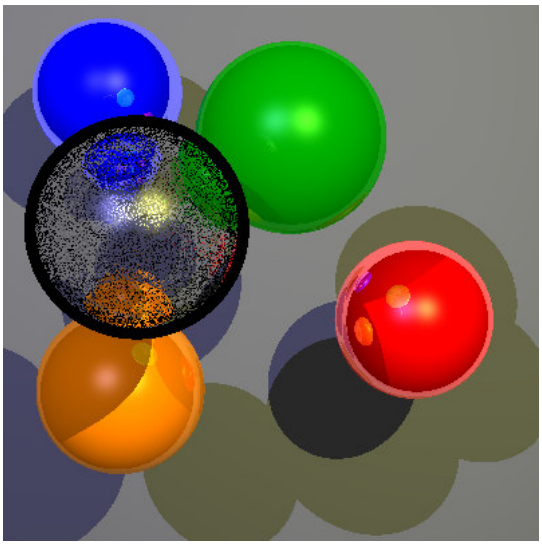




The finished product after fixing the intersections.



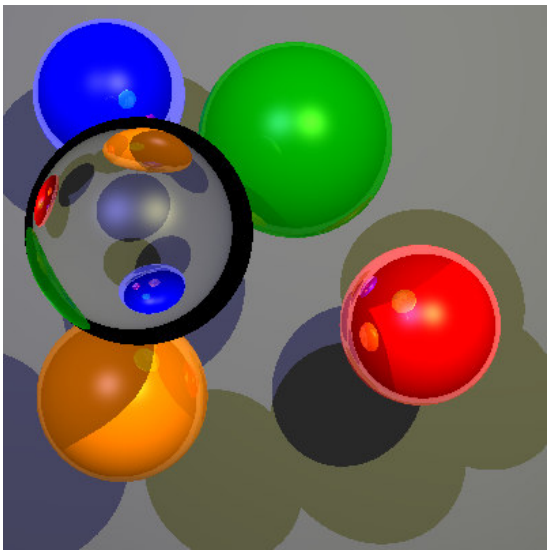
Reflections were actually incredibly easy, only a few lines of code and changing the method slightly to pass an additional counting variable.



For the final part I did refraction, the first attempt didn't go great… Partially because I used the reflection number instead of eta.

This was the part that gave me the most trouble, partially because some of the other parts worked but I had implemented them slightly incorrectly so while they worked for some applications they did not for all. So in addition to making the refraction work I had to fix the previous parts.

(While I didn't keep any of the pictures there was a lot of attempts which ended with that clear sphere being black)



The finished product, produced with scene06.yaml