Adam Lieu (100451790)

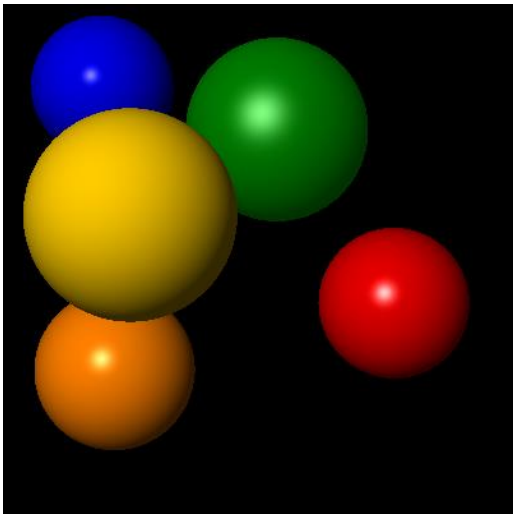
Assignment 2 – Ray Tracing:

"I, Adam Lieu, certify that this work is my own, submitted for CSCI 3090U in compliance with the UOIT Academic Integrity Policy"

Intersection and normals:

- For this part, I simply implemented the calculations for calculating the normal and intersection by first calculating a, b and c and using the quadratic formula to calculate for the roots and then using the roots to find whether the ray intersects a sphere. The second part was calculating the sphere's normal at the intersection point, which was just finding a vector for it.

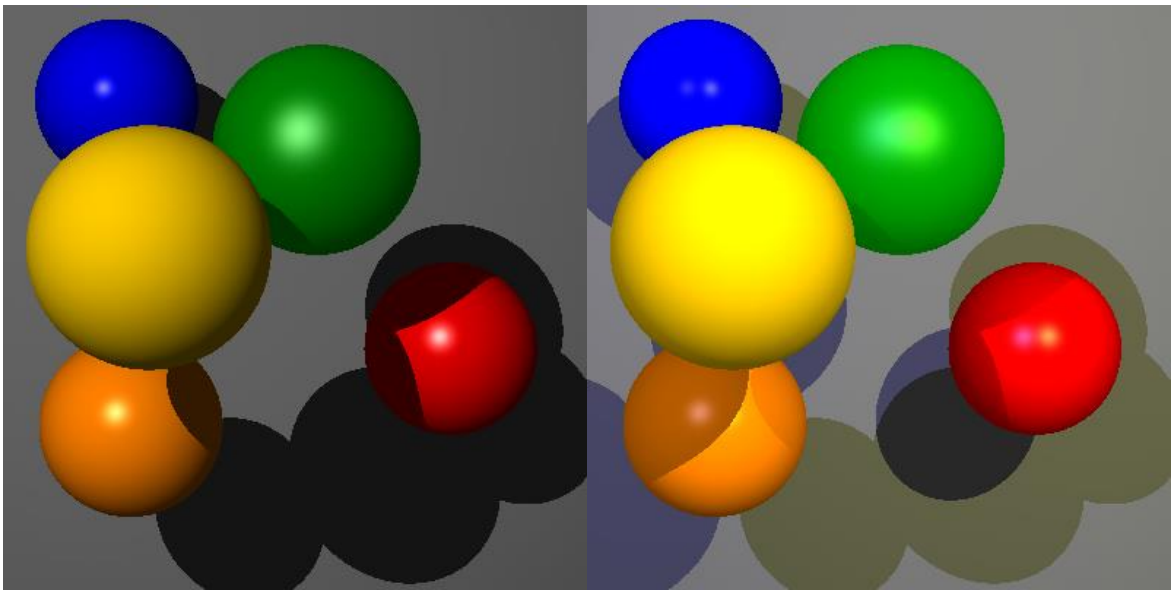
Phong Illumination:

- For this part, diffuse lighting was obtained by first calculating the vector L that is used in the calculation of diffuse lighting and then applying the diffuse lighting by adding it to "color".
- The second part of this section was to apply ambient and specular lighting, which was obtained by using the equations for ambient and specular from lectures and applying it to the code.
- Generally for this section, it would all be done in a For loop, where it would constantly recalculate things such as the vector L and R, that are both used in the calculations for diffuse and specular lighting respectively. In each loop iteration, the calculations from each lighting are added onto "color", giving us the expected result from the spheres.
- 

Adam Lieu (100451790)

Shadows:

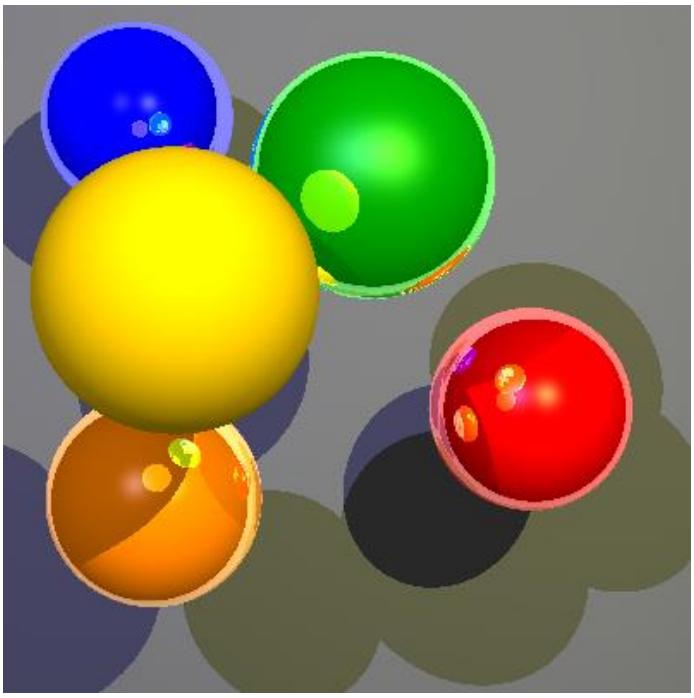
- For this section, I used the same code from the framework, given earlier on in “scene.cpp”, where it was used to find hit object and distance, and modified that to get shadows to work properly. I also used a Boolean value called “shadows” to determine when a shadow should be produced depending on the calculations for intersection. Furthermore, the code for the phong illumination was modified to be only applied when “shadows” is false, otherwise, ambient lighting would only be applied.
- Initially, the output gave me a “grainy” picture, where the shadows didn’t appear right. The code for “jiggling” rays was applied and fixed this problem.
- Attached are scene03.png and scene04.png respectively



Reflection:

- For this section, Material.h was extended to include reflect, refract and eta and raytracer.cpp was also modified in order to be able to parse the yaml files that had those values for the spheres.
- Scene.cpp was further modified to include the calculations for reflections, first initially a reflection ray that was called “testRay” that used the equation for a reflection ray. Recursion was used in the form of an If statement where if the coefficient for specular lighting was NOT zero, then we would apply recursion with the reflection ray to “color”.

Adam Lieu (100451790)



Extensions:

- For this section, an attempt was made to extend the program to have refraction
- However, I was unable to get it to work due to a stack overflow problem that occurred whenever scene06.yaml was used for testing.
- The solution was implemented however, by initializing the values for eta and the square root term in the “t” calculation.
- Since we know that if the square term is negative, the ray will not be refracted, meaning that no refraction will be applied when it enters the material and no refracted light contribution will occur if the ray is exiting the material.
- An If statement was implemented to check whether or not refraction was needed at first, then checking if the square root term was not negative before calculating for the direction vector of the ray and then applying it recursively, similar to the reflection part.
- Otherwise, it would skip that part and continue on.