

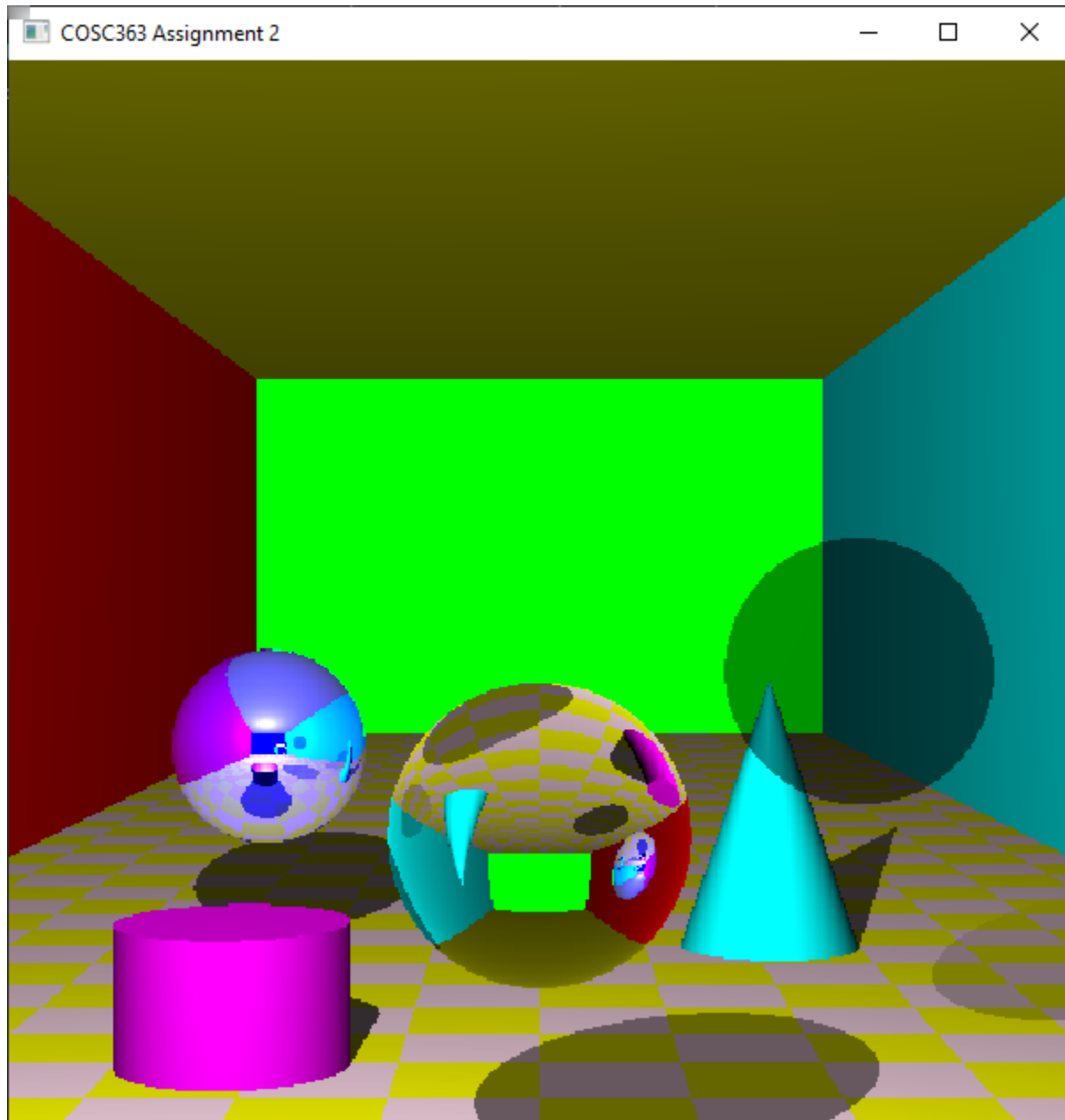
COSC363 Assignment 2 Report

Name: Jake Wilson

ID: 49606681

UserName: jwi187

Ray Tracer Output



Basic Features:

As you can see all of the Basic Ray Tracer Features have been implemented.

- The scene is set in a room of 6 axis-aligned planes which have different colors/patterns.
- The Sphere on the right is a transparent sphere.
- All of the objects cast shadows with the Transparent and Refractive objects casting lighter shadows
- The sphere on the left has a mirror-like reflection with surrounding objects being reflected.
- The Floor Plane has a Checkered Pattern.

Extra Features:

Refraction of Light Through an Object


The center Sphere Refracts light through it with the current setting set to an Index of Refraction of 1.4

Objects other Than a Plane or Sphere

- **Cylinder**

The Magenta Cylinder on the left is made with the code located in Cylinder.cpp and Cylinder.h. It is a subclass of the SceneObject class. The maths used is from Lecture 8 - Ray Tracing (Pages 39-45). The Intersection is:

Intersection equation:


$$t^2(d_x^2 + d_z^2) + 2t\{d_x(x_0 - x_c) + d_z(z_0 - z_c)\} + \{(x_0 - x_c)^2 + (z_0 - z_c)^2 - R^2\} = 0.$$

Which is a Quadratic so we can use the Quadratic equation,

$$x = (-b \pm \sqrt{b^2 - 4ac}) / 2a \text{ to work out the intercepts with the cylinder.}$$

Then we just check if the ray hits the cylinder and return the value when it does.

- **Cone**

The Teal Cone on the right behind the Transparent sphere is made with code in Cone.cpp and Cone.h. It is also a subclass of the SceneObject class. The maths used is from Lecture 8 - Ray Tracing (Pages 46 - 47). I found some code at the following address:

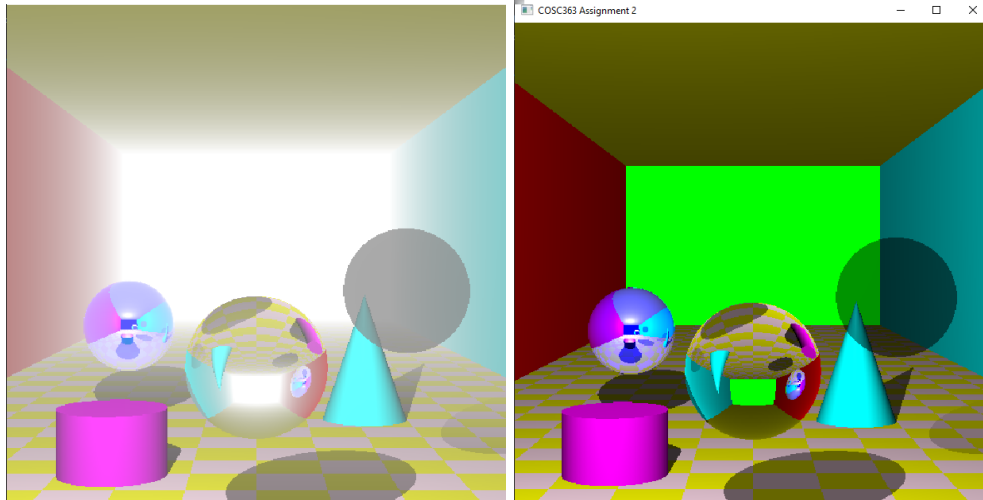
<https://stackoverflow.com/questions/34157690/points-of-intersection-of-vector-with-cone>

Which I made use of and it allowed me to then follow the same process I used in the Cylinder Class (checking intersects and returning appropriate values).

A clearly visible “Cap” for a cylinder or cone:

On the Cylinder on the left you can see the cap on the top, you can also see it in both the Refraction Sphere and Mirror sphere.

Fog:



I used the Equation given to us in the Assignment 2 Powerpoint Slides

$$\lambda = \frac{(ray.hit.z) - z_1}{z_2 - z_1}$$

$$color = (1 - \lambda) \text{ color} + \lambda \text{ white}$$

And here is my implementation:

```
//-----  
/--Fog Calculation - Uncomment to Use it--  
//-----  
  
/*  
float fogColorValue = ((ray.hit.z) - 0) / (-300 - 0));  
glm::vec3 fogColor(1.0, 1.0, 1.0); //white  
color = ((1 - fogColorValue) * color) + (fogColorValue * fogColor);  
*/
```

Time Estimation:

My Ray Tracer is set to 600x600 pixels and is run through Visual Studio. It takes 4-5 seconds to generate the output.

Build Process/Environment:

My Environment I used was Visual Studio 2019 and I used the Local Windows Debugger to Generate Output.

I declare that this assignment submission represents my own work (except for allowed material provided in the course), and that ideas or extracts from other sources are properly acknowledged in the report. I have not allowed anyone to copy

my work with the intention of passing it off as their own work.

Name: Jake Wilson

Student ID: 49606681

Date: 29/05/2023