# CSE306 Computer Graphics

# Project 1 Report

Carolina Nina Matos

May 2022

## Introduction

For this project, I implemented a simple ray tracer with a couple of features. I would have liked to further develop it to include more, but unfortunately struggled a lot and thus did not have time to do so. Currently it supports:

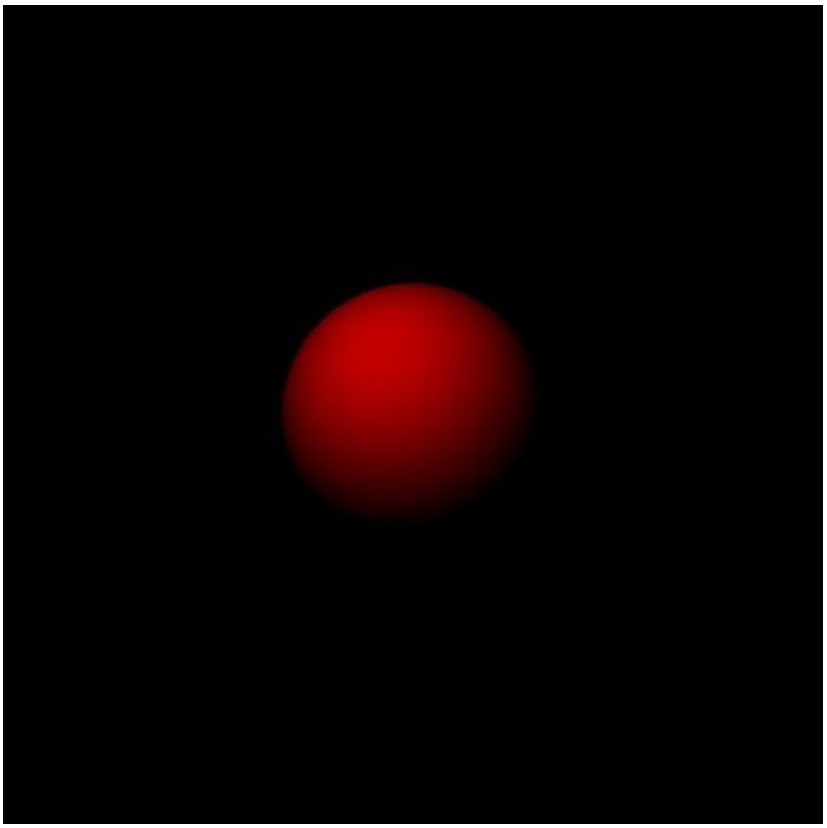
- Diffuse and mirror surfaces
- Anti-aliasing, with an option to toggle it on or off
- Gamma correction
- Refractive surfaces (i.e. transparent surfaces, with variable refractive index)

The effects of these features will be illustrated in this report. I placed my camera at (0,0,55) and used a light intensity of 8*10^9. The illusion of a room was made using large spheres of different colors; relative to the camera, I placed a green wall in front, a red one on top, a blue one on the bottom and a pink one behind. This allowed me to test how different materials would reflect light of different colors.

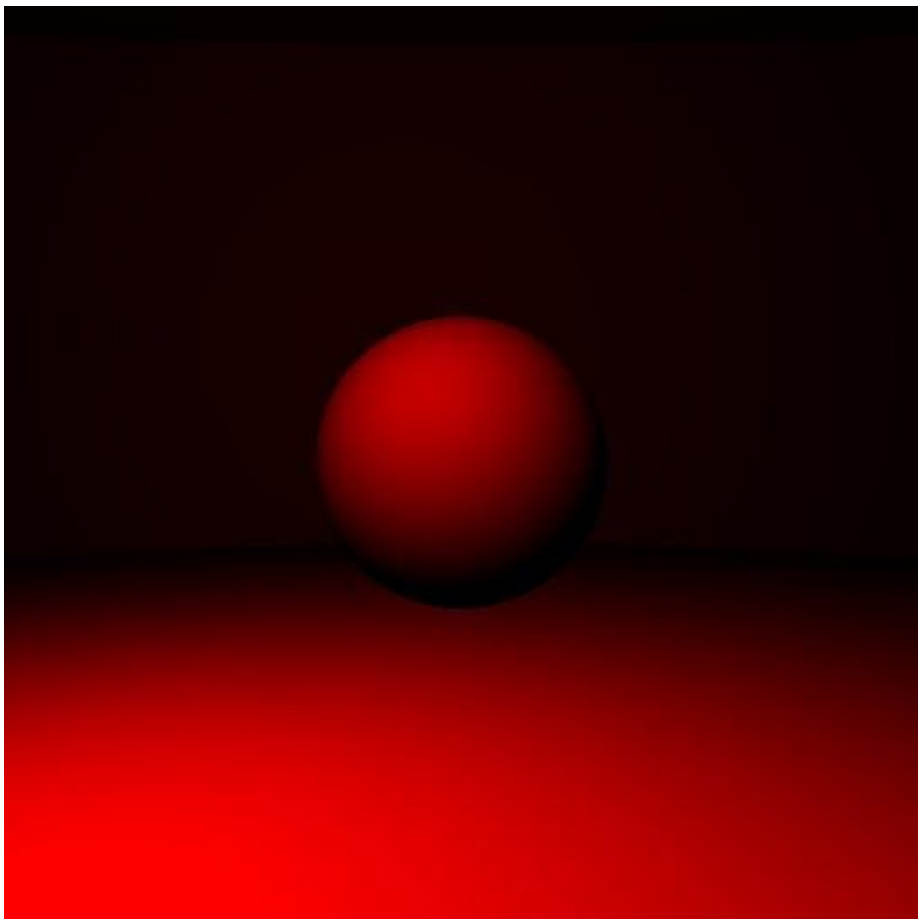
The stage is set up by initialising a number of spheres (i.e. instances of the Sphere class), with customizable material, radius, position, color, and, optionally, refractive index. These are then ‘collected’ together and put into an array, which is used to define the Stage. The stage currently only has one point light source, but the idea was to make the code easily extensible so that in the future, more light sources could be included.

## Initial attempts

I began by trying to render a single sphere. This was done by implementing a sphere and ray class, then manually computing the color in my main function when sending a ray through each pixel. The result is below - this was before I added in a stage ("scene") with walls.



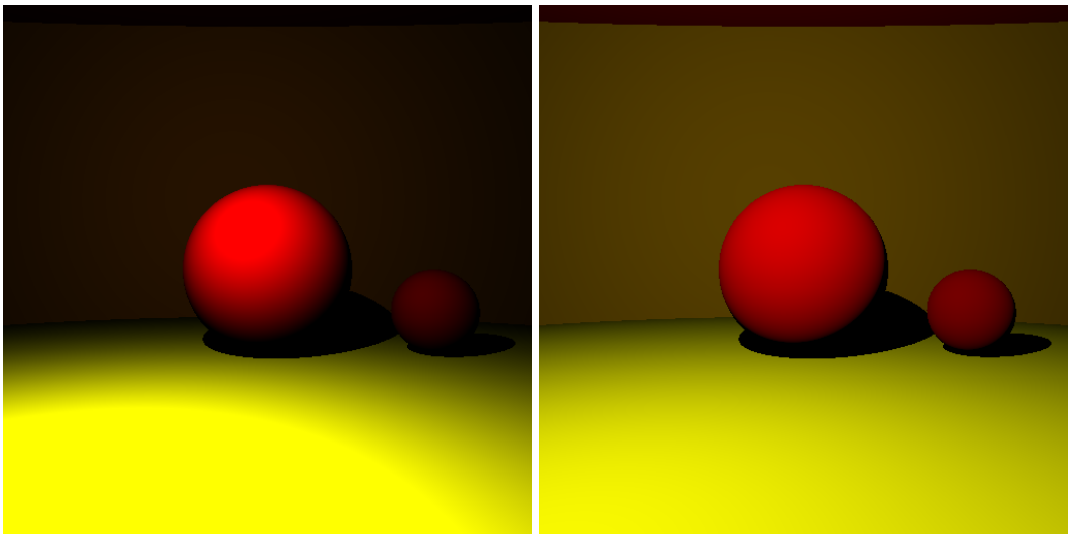
The image clearly appears quite dark, with a very high level of contrast. It also has no further context - there is only a single sphere. To test my stage class, I then added in the walls. This was challenging, as initially there was a problem with calculating the intersections for all the spheres, but once this was solved, I was able to render the image in the following page. In both cases, the rendering took only a couple of seconds, less than a minute.



Above we can now see the floor, as well as a hint of the back wall, but the lighting is not great (there is no visible drop-shadow) and there is still very high contrast. This led to the next step - I then tried to include another sphere beside the original one and played with the sizes, as well as the colors of the walls.

## Gamma correction and direct lighting

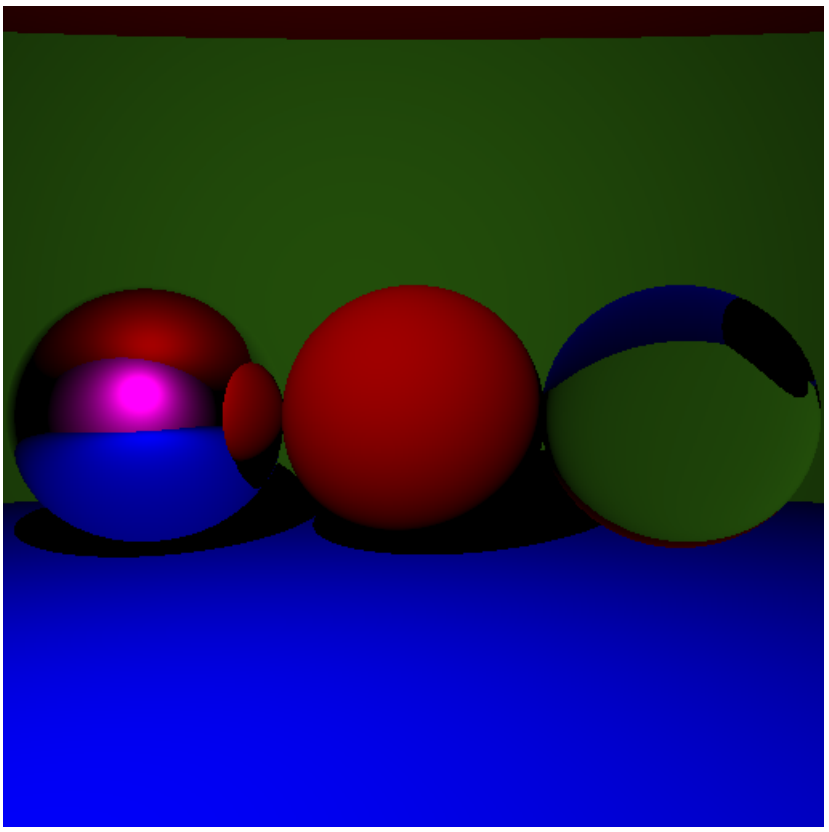
To solve the problem of an overly high contrast, I gamma-corrected the image. Below, one can see the difference between the corrected image on the right and the original image on the left.



In the corrected image, the walls are now more visible, as are the drop shadows. The harsh drop shadows are caused by the direct lighting, though a more realistic image could be rendered by using indirect lighting to soften these shadows. Unfortunately, I did not have time to implement this.

## Different surfaces

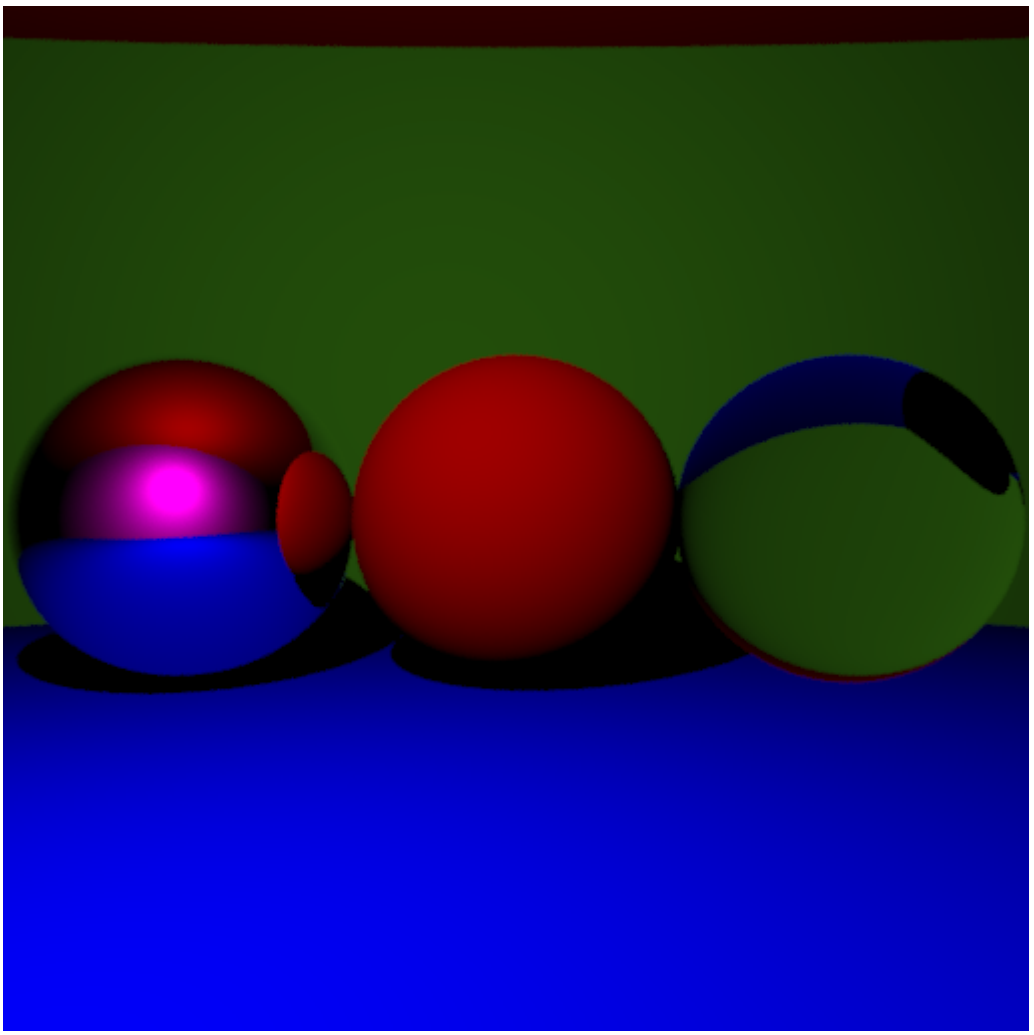
I implemented 3 main compatible surfaces - diffuse surfaces, mirror surfaces and transparent surfaces. One can change the refractive index of the transparent surface as desired. For the example below, I used a refractive index of 1.4 for the glass ball.



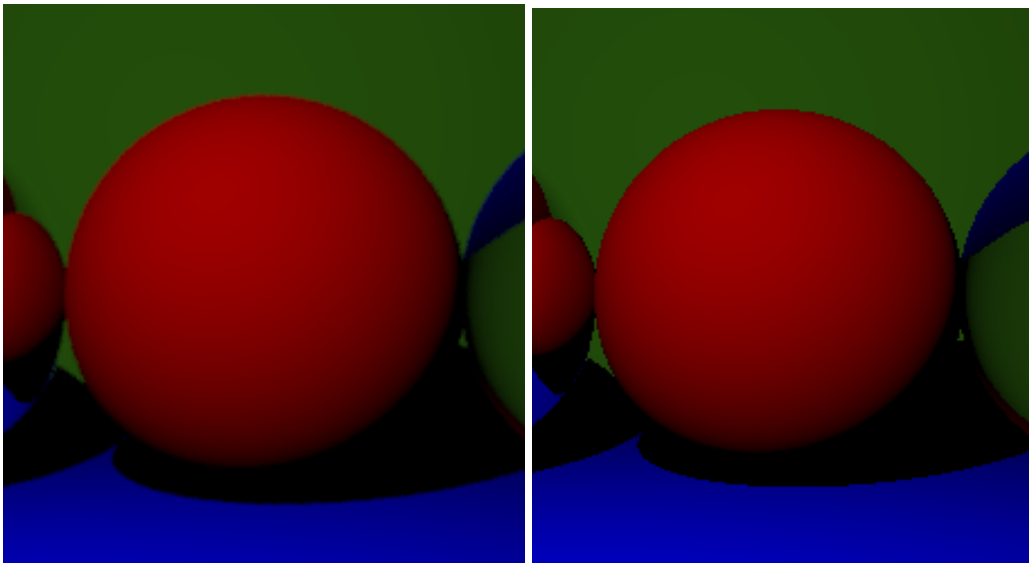
As one can see, the mirror ball perfectly reflects the room, and the transparent one distorts the colors around it, as the light is refracted. This image took approximately 55 seconds to render, with a ray depth (i.e. cap on the recursion for obtaining the color for mirror and transparent surfaces) of 10, and 15 rays per pixel. The image is a little bit dark, which in part is likely due to the choice of colors, but this is another issue that indirect lighting would likely help with.

## Anti aliasing

To reduce the jagged edges of the image, I added anti-aliasing, which essentially averages out the color of the various rays going through each pixel. This is contained within an if/endif statement which allowed me to test it, as well as to compare the images. The anti-aliased image equivalent to the above is below, in its original dimensions.



I used the same amount of rays as before, and this image took considerably longer to render: approximately 1 minute 35 seconds.



By placing the images side-by-side as above (the anti-aliased image being on the left), one can see that there is a difference; the edges are considerably smoother in the left image, which is particularly noticeable in the drop shadow.