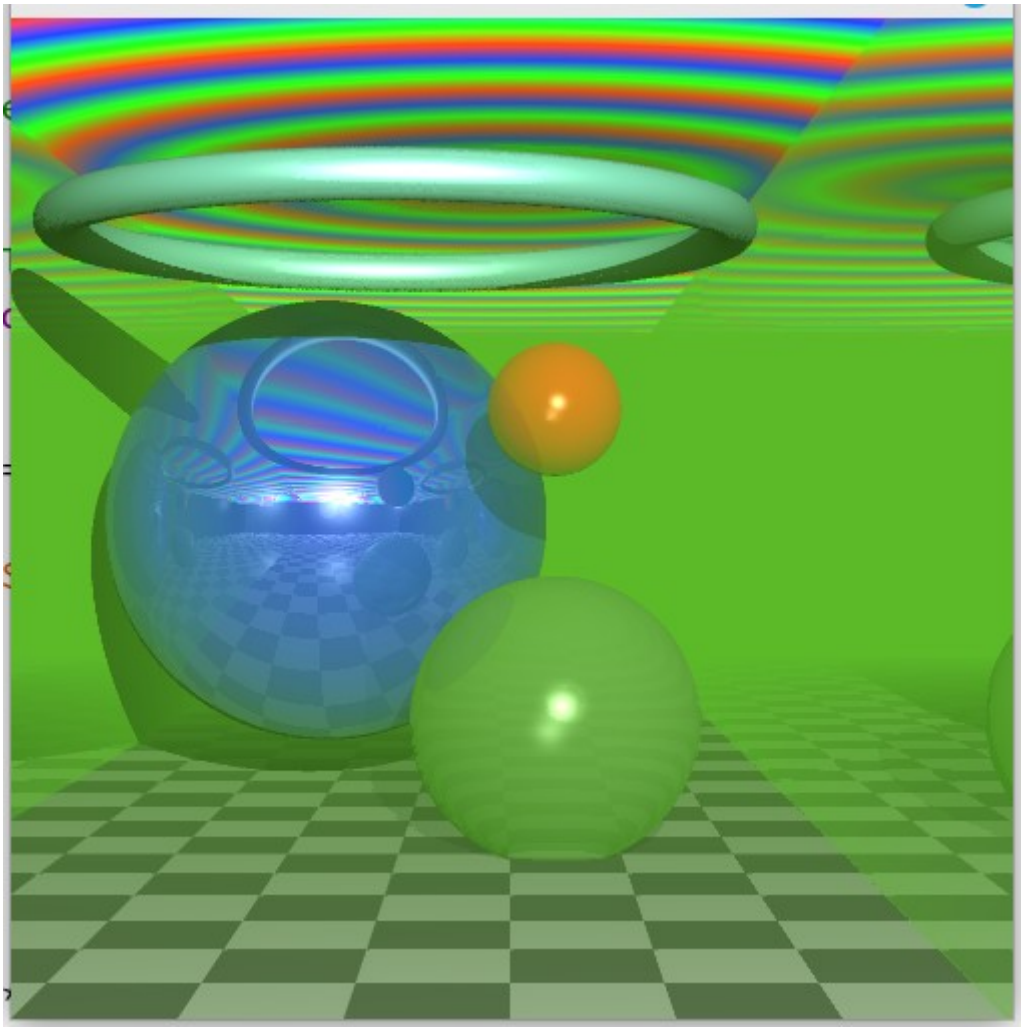
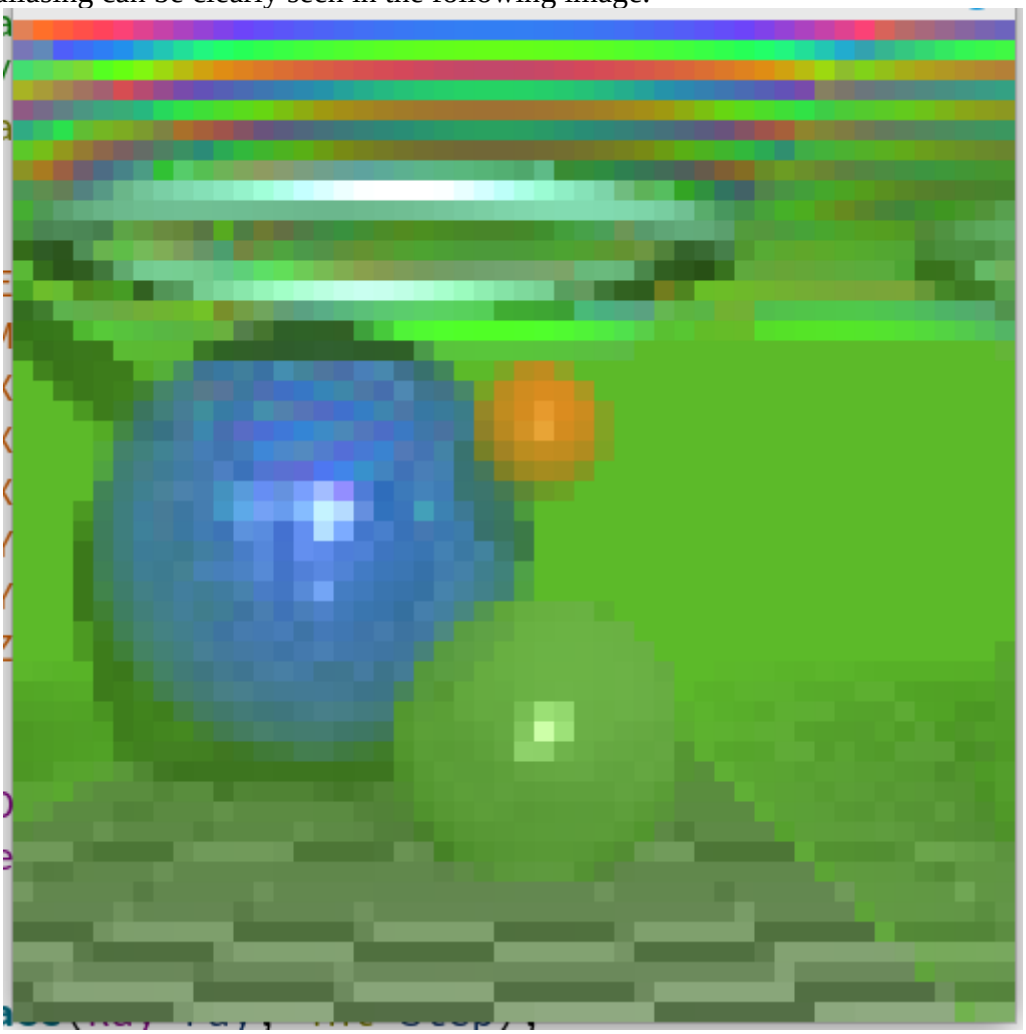
Over all the ray tracer can render an image containing shears, planes and tours, that can be reflective, refractive and or transparent. It takes around a minute to render the image and can be compiled by a Linux mechanic using c make.



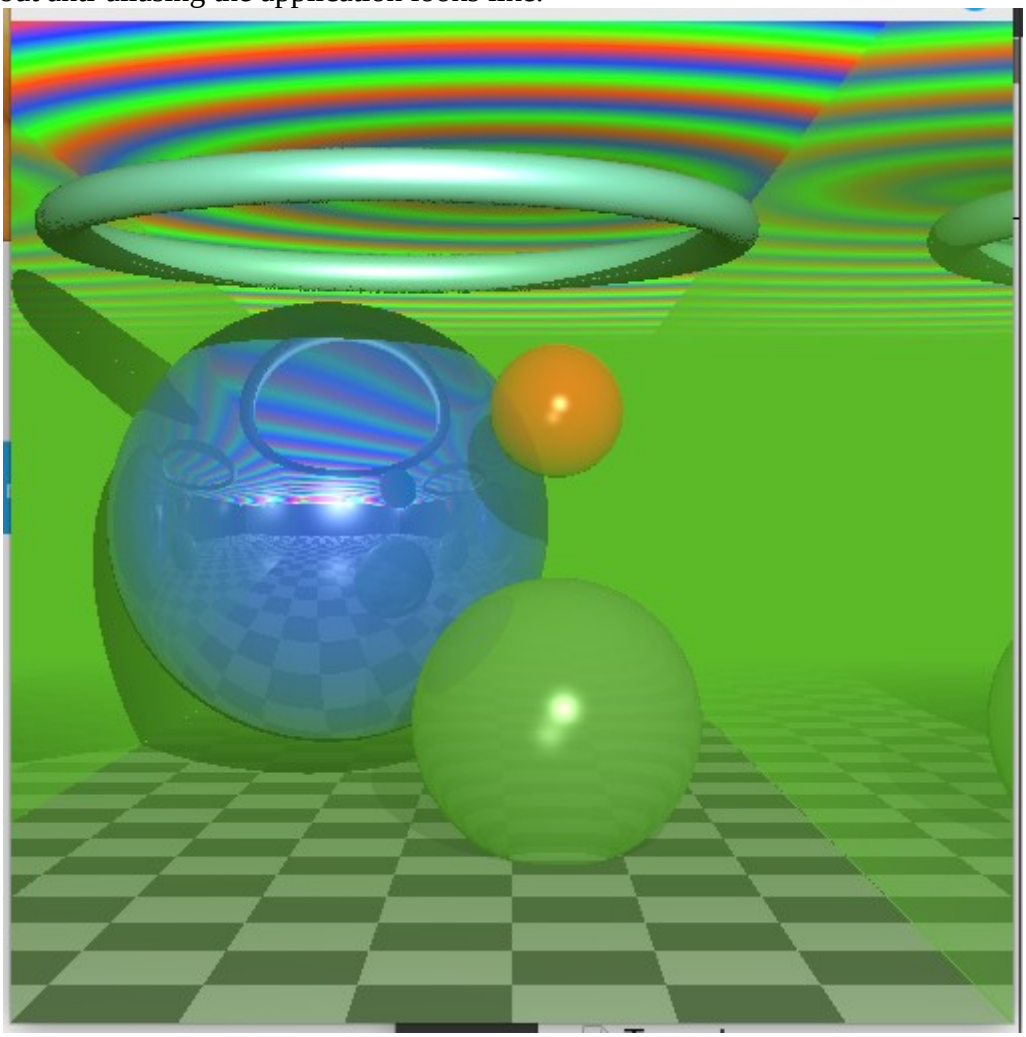
the tours generated by using a spherical bounding volume, followed by using a quadratic equation that was got from (http://blog.marcinchwedczuk.pl/ray-tracing-torus) the equation was solved using code from (https://github.com/erich666/GraphicsGems/blob/master/gems/Roots3And4.c) once the equation was solved the lowest value that was more then 0 was returned if it exists otherwise it would return a -1. the tours mostly displays fine but dithering like artifacts occur I believe this is caused by floating point error.

refraction though the gray shear where made by making a ray when ever a ray collides with the object and using glm:refract to decide the direction of that ray.

The adaptive anti-aliasing was done by first calculating the color of every pixel with out anti-aliasing and then going though every pixel and checking that it's color distance squared is not more then 0.1 from any of it's neighbors then it will be subdivided and in that subdivision, will only be subdivided if it has a color distance squared more then 0.01 to all it's other sub pixels in the subdivision. The color of the subdivision and pixel is then calculated by the average of it's subdivisions. This meant reduces hard borders between pixels.
This anti-aliasing can be clearly seen in the following image.



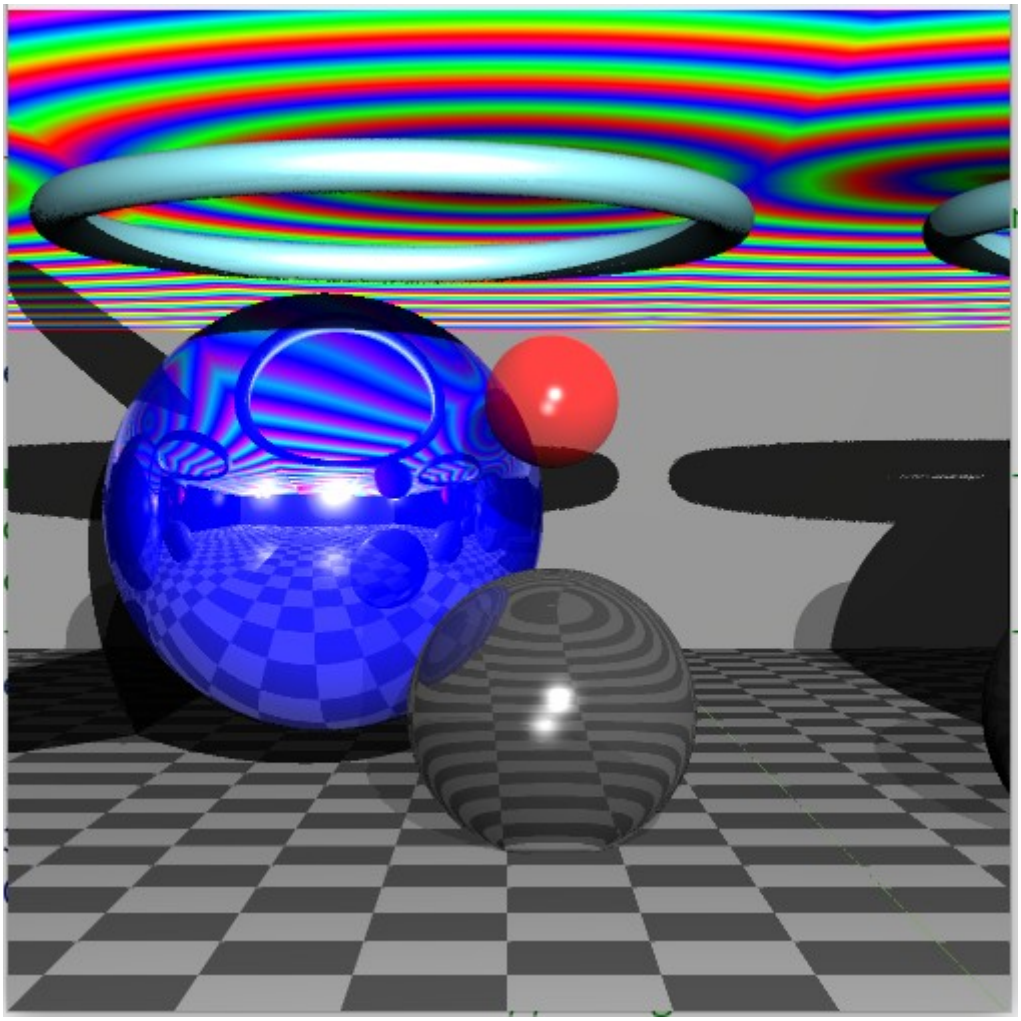
And with out anti-aliasing the application looks like.



on the ceiling is a procedural generated Patten where it’s color linearly interpolating between blue and red to between red and green and then green and blue, cycling like that depending on it’s distance from a point in the ceiling. The brightness of this color is also determined by it’s distance from that point with it getting brighter the farther from that point you go.

To the left and right are paralegal mirror-like surfaces and the reflections from these recursive reflections from these mirror like surfaces can be seen in the surface of the blue shpear.

the fog is generated from by interpolating the color of in the ray tracing algorithm between the normal out color and the fog color using the z position of the hit point. With out the fog the application looks like.



I declare that this assignment submisison 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.
Daniel Lowe
89907240
12/05/2022