Cosc 364 Assignment 2

Raytracing

Daniel mcinnes
11933946

The ray tracing program i made had all of the minimum requirements but i have also done Anti-aliasing through super sampling and i also have a refractive sphere (please see screen shots).

Both the transparent and the refractive sphere give the ground below a lighter shadow. The refractive sphere gives a set lighter shadow but the transparent sphere gives a shadow lightness based on the transparency of the object. There is also a color added to the shadow based on the color of the transparent sphere, in the case of the demo this light is blue.

The refractive sphere is a sphere that distorts light as it passes through like a real glass sphere would do. This is done by calculating the new angle of the ray when it changes between refraction indexes, the Snell's Law of Refraction ($n_1 sin(\theta)_1 = n_2 sin(\theta)_2$) is what determines what the angle of the new ray will be. When rearranged the eq becomes $\sqrt{1 - (\frac{n_1}{n_2})^2 (1 - (d.n)^2)}$ where n1 and n2 are the refraction of the outside and the inside of the object, n is the normal and d is the incoming ray. But when working in opengl the refraction calculation is done with the glm::refract() function, this function has to be called when the ray both enters and leaves the object.

I also added a procedural pattern on the back plane of the scene. This is done by adding two colors based on the x and y pos of the ray after thay have been through a sin and cos function. The equation is "outputColor = (baseColor1 * sin(x pos) * 2) + (baseColor2 * sin(y pos) * 2)" this adds a cool fence pattern with blue and red stripes with glowing white spots where they meet and black filling up the rest. This pattern also gets dulled in the shadows in a interesting way giving a blur effect this is because of the sharp drop off in the edges getting smoothed out

The last thing I added was anti aliasing, anti aliasing is where jagged lines are smoothed to give an overall better picture quality. anti aliasing was done in this case with supersampling. Supersampling is where a higher image is rendered and then it is downscaled to fit the resolution but taking the average of the pixels. In the case of this ray tracing program each pixel had 4 slightly offset rays sent out in a square pattern, the color values of each ray was then averaged to get a final color. The effect can be seen in the two pictures below.

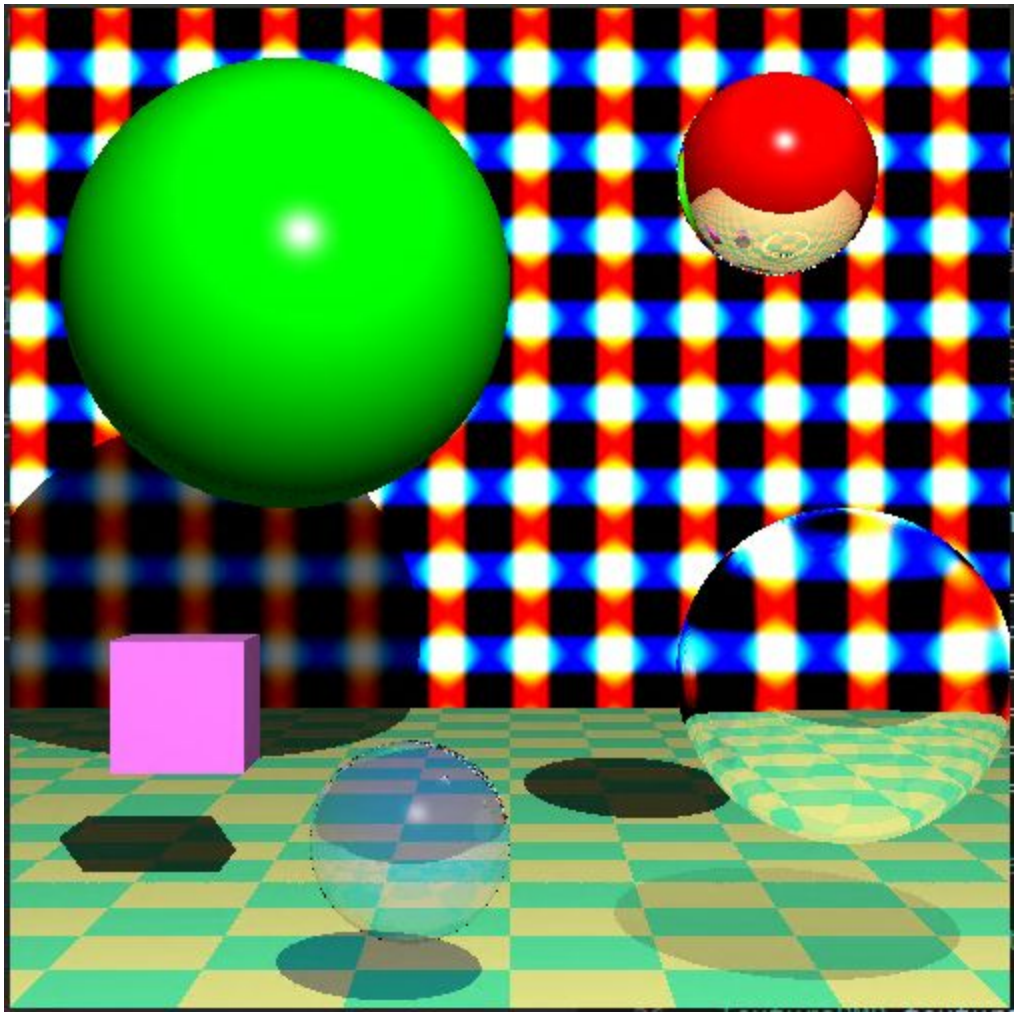
When making this i tried to add other shapes but had problems and in the end ran out of time to get any added. The broken code was left in but was left unused. The pattern on the backplane I added also makes the reflections look broken at a quick glance but is in fact working fine, this caused me up for an hour before i realized.

The run times with no AA was about 5 seconds and with AA was about 20 seconds. This makes sense as with AA on there was 4 times the amount of ray send out so 4 times the calculations needed

The program can be build with the command “make” and then ran with the “./RayTracer.out” command. If the make command does not work the “cmake ./” may fix it. (LINUX ONLY)

Files can also be found at “https://eng-git.canterbury.ac.nz/dmc245/363_2"

NO AA:



WITH AA: