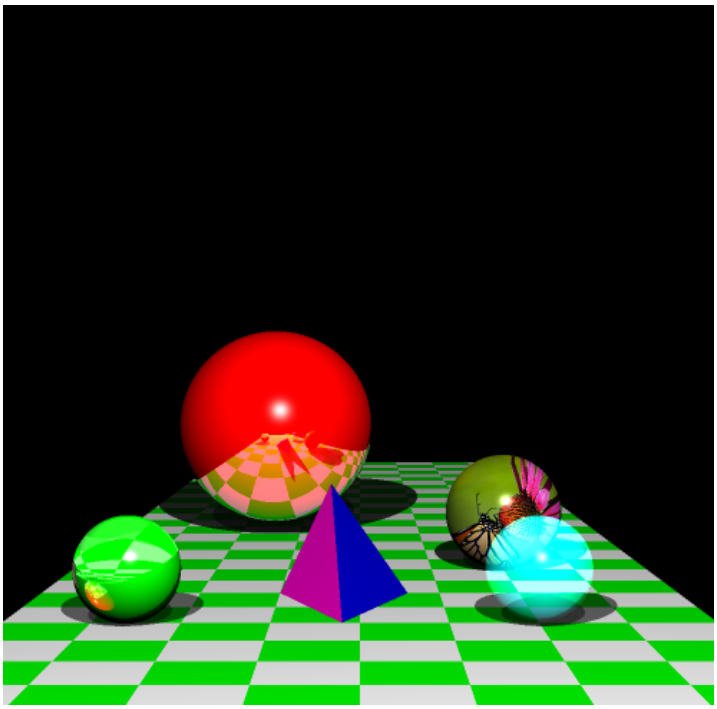
COSC 363: Computer Graphics Assignment 2

Leo Carolan - 98105803



The Scene:

My raytracing scene consists of a floor plane, 4 spheres each with different properties. One has the butterfly texture mapped to it, another is reflective, another is refractive and the other is transparent.
There are also a set of 4 planes shaped to create a pyramid in the centre of the scene. The scene takes about 7 - 10 seconds to generate with anti-aliasing and about 3-5 seconds without. I believe the scene has a meaningful spatial arrangement of objects. The reflective red sphere is placed at the back of the scene to allow all other objects to be reflected off it, including the pyramid to allow for its shadows to be shown with the lights position. The transparent blue sphere is placed in front of the textured sphere to show off its transparency. Anti aliasing is also used to produce a more aesthetically pleasing view of the scene.
I also believe my scene has some negative attributes to it such as the lack of complex features/objects such as a cone, cylinder or torus and no procedural pattern generated on any surface.

Extra features:

Refraction:

The green sphere situated on the left side of the scene is refractive. This is achieved by refracting the incoming rays by a refractive index eta (1/1.33) and then refracting again by 1/eta (1.33) for the exiting rays. We then trace those rays and obtain the color values they hit to produce the resulting reflection onto the sphere. The sphere also causes lighter shadows to be cast.

Multiple light sources and shadows:

I have created a second light source that comes from the -x direction, which causes an additional shadow to be cast for each object within the scene. The shadows have the same intensity for each light source, however a more intense shadow is created when the two shadows intersect.

Anti-aliasing:

Anti aliasing was implemented using supersampling in which each pixel was divided into four quadrants, and four rays are generated through the centre of each quadrant. The average of the colour values traced by the rays is computed and used to set the colour of each pixel in the scene. This creates a much smoother appearance along the edges of objects and shadows. This is evident between figure 1 and figure 2.

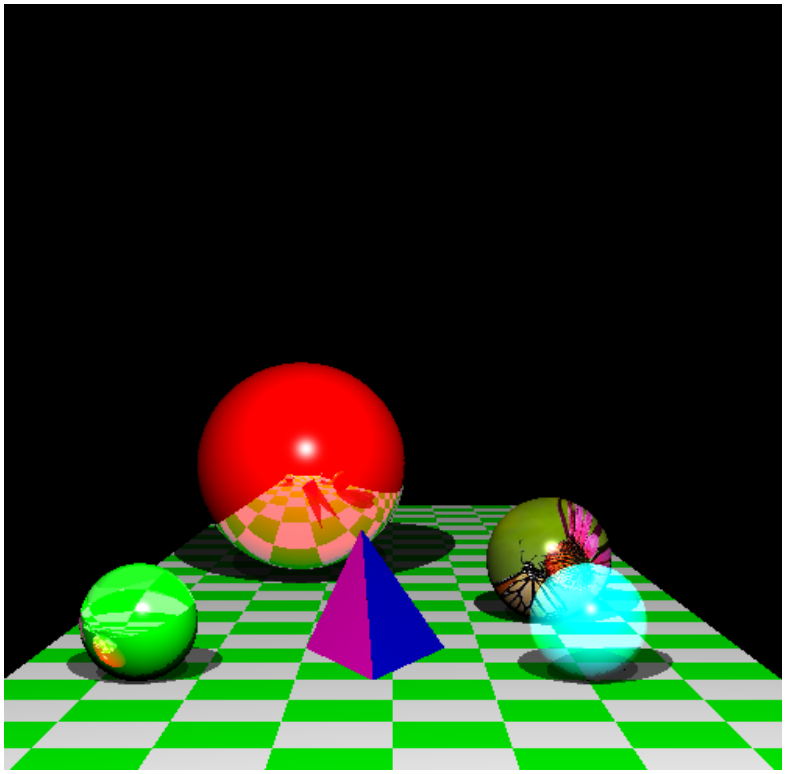


Figure 1: Without anti aliasing

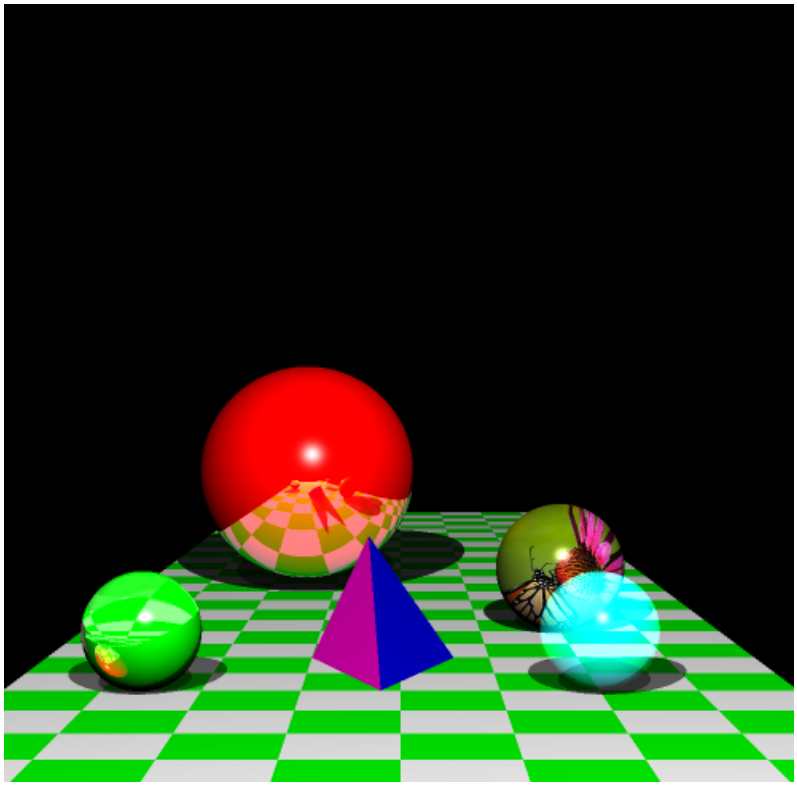


Figure 2: With anti aliasing

A non-planar object textured using an image:

The sphere in the right behind the transparent blue sphere is textured with Butterfly.BMP. This is done by using the normal of each vertex. Texture coordinates are generated based on the angle of the surface at each point.

```
float texcoords = asin(obj->normal(ray.hit).x / M_PI + 0.5);
float texcoordt = asin(obj->normal(ray.hit).y / M_PI + 0.5);
if (texcoords > 0 && texcoords < 1 &&
        texcoordt > 0 && texcoordt < 1) {
        obj->setColor(texture.getColorAt(texcoords, texcoordt));
}
```

FIgure 3: Texture code

Fog:

Fog was implemented by changing the background color to a color of 0.97, and then, in the trace function, the function provided in the lecture notes (figure 4) is used. The fog factor is then used to linearly blend the colour values of the scene within z1 and z2 (-65, -215). This results in a nice fog effect to occur within the scene.

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

Figure 4: Fog factor

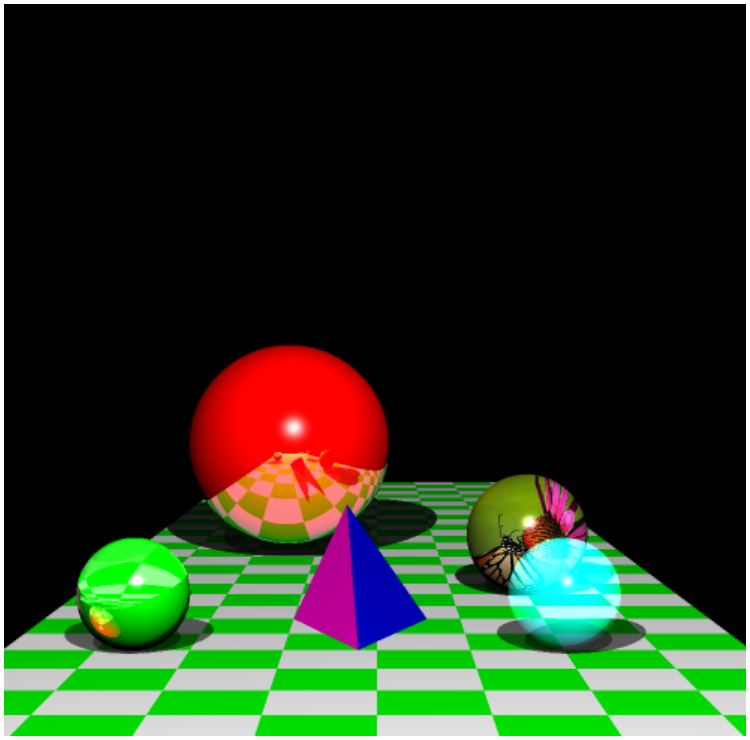


Figure 5: Without fog

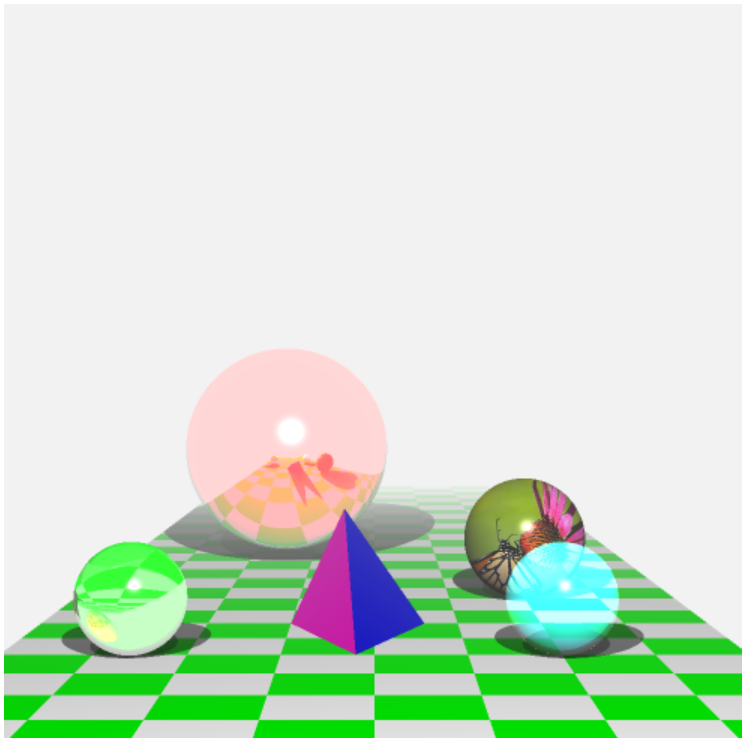


Figure 6: With fog

Build Process:

1. Open visual studio code
2. file-> openfolder ->FOLDERNAME -> ok
3. open new terminal in VS Code
4. Enter cmake .
5. Enter make
6. Enter ./RayTracer.out

References:

1. https://www.mvps.org/directx/articles/spheremap.htm
2. Note07_RayTracing.pdf