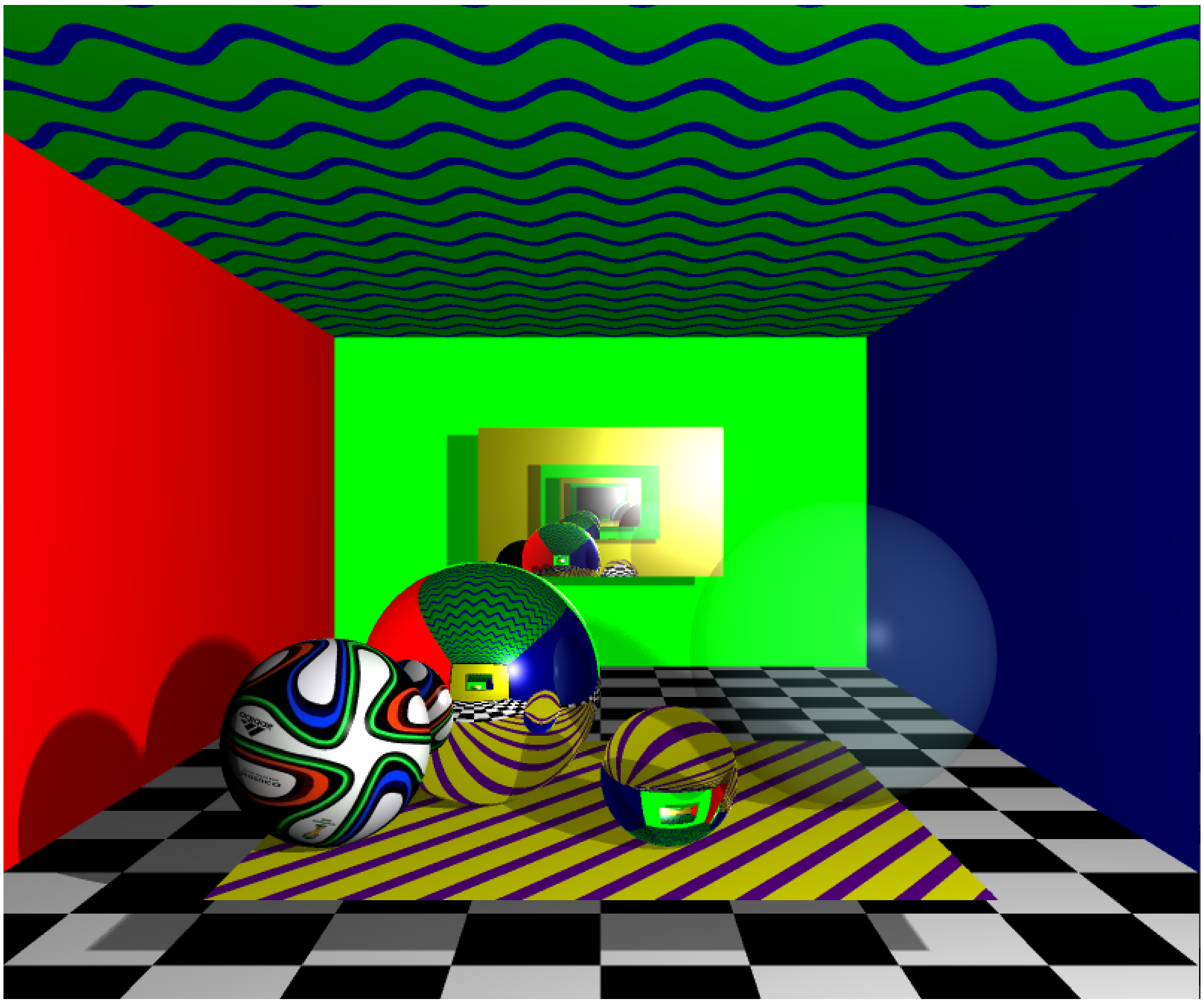
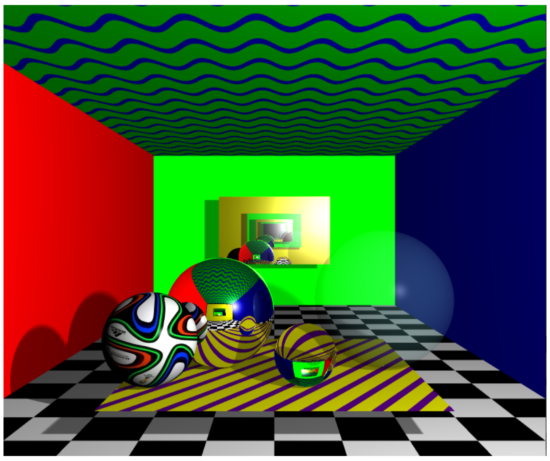
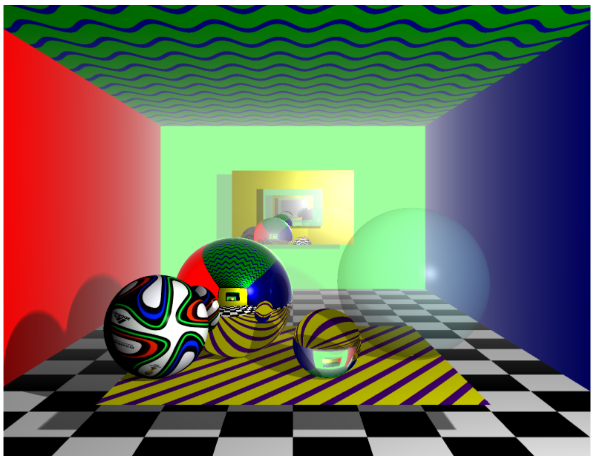
COSC363:
Computer Graphics Assignment 2
Ray Tracer



In this assignment, the lab 6 and 7 ray tracer code has been extended by implementing six planes to create a room environment. The room consists of various objects, including a chequered floor plane, a desk with a diagonal pattern, and four spheres with different properties. Additionally, there are other features added such as multiple reflections, fog, textured sphere, anti-aliasing, procedural pattern, and shadows to enhance the visual quality of the rendered image.The first sphere is reflective, the second sphere is refractive with an index of refraction of n = 1.5, the third sphere is a light blue transparent sphere, and the fourth sphere represents a soccer ball with a textured surface.
To simulate multiple reflections,two mirrors have been placed facing each other in the room. One mirror is located on the front plane, while the other is on the back plane. This setup creates a visually interesting effect by reflecting objects multiple times.

FOG


Before

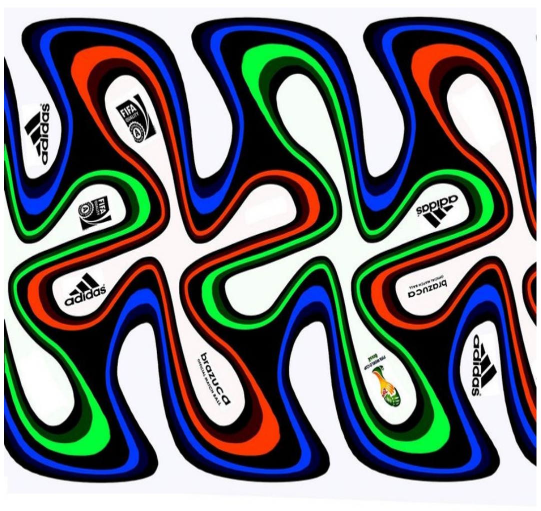

After

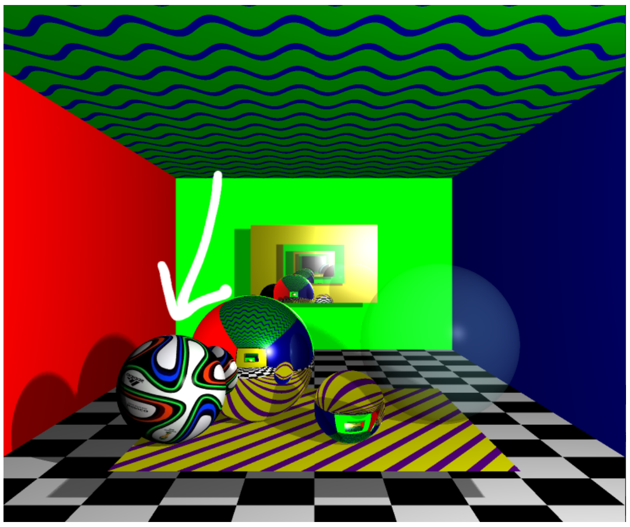
$$\lambda = \frac{(ray.hit.z) - z_1}{z_2 - z_1}$$

color = (1- λ) color + λ white

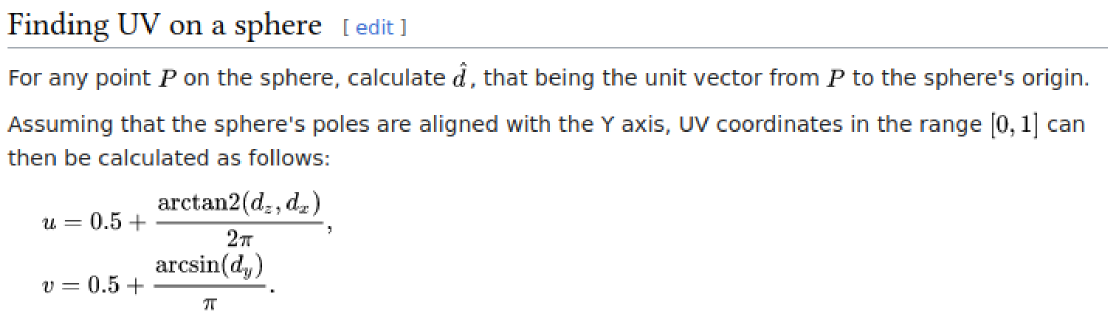
In the ray tracer, the option is provided for the user to enable or disable fog through the main function. By default, the fog is turned off for a better visual appearance. When the fog_state is enabled (true), the code calculates the gamma value based on the z coordinate of the intersection point between z1=-40 and z2=-120. The fog effect is then applied within this range, gradually blending the original color with white based on the gamma value. This creates a foggy effect where objects farther away from the camera appear more faded.

Textured sphere





A texture to one of the spheres, giving it the appearance of a FIFA ball. This was achieved by mapping a texture image onto the sphere using UV mapping techniques.



Finding UV on a sphere [edit]

For any point P on the sphere, calculate $\hat{d}$, that being the unit vector from P to the sphere's origin.

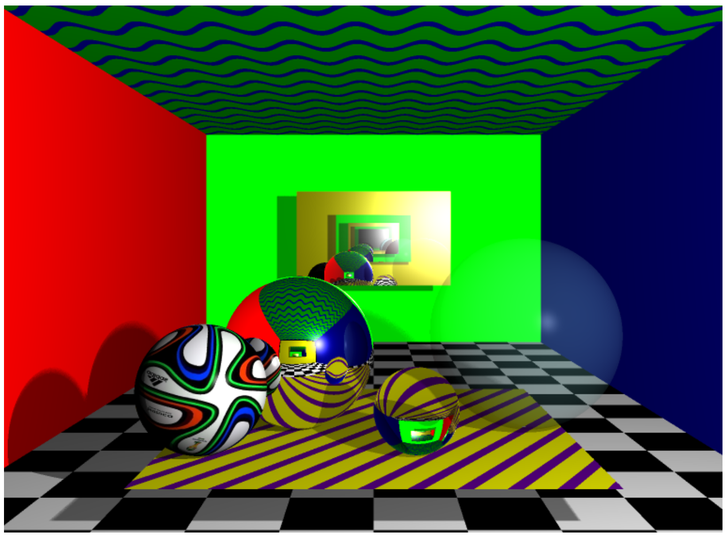
Assuming that the sphere's poles are aligned with the Y axis, UV coordinates in the range $[0, 1]$ can then be calculated as follows:

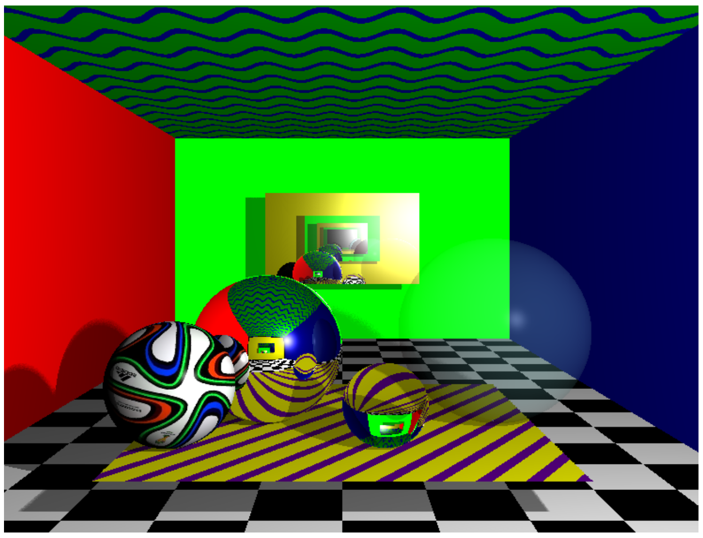
$$u = 0.5 + \frac{\arctan2(d_z, d_x)}{2\pi},$$
$$v = 0.5 + \frac{\arcsin(d_y)}{\pi}.$$

https://www.google.com/search?q=fifa+ball+texture&client=firefox-b-lm&sxsrf=APwXEdegvKWLVWmMT2xP606kRc0mdcYatg:1685695293576&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjm_7CpmKT_AhVOkIYBHQt7DG0Q_AUoAXoECAEQAw&biw=1920&bih=995&dpr=1#imgrc=Ayck-bMLhvQTIM

https://en.wikipedia.org/wiki/UV_mapping

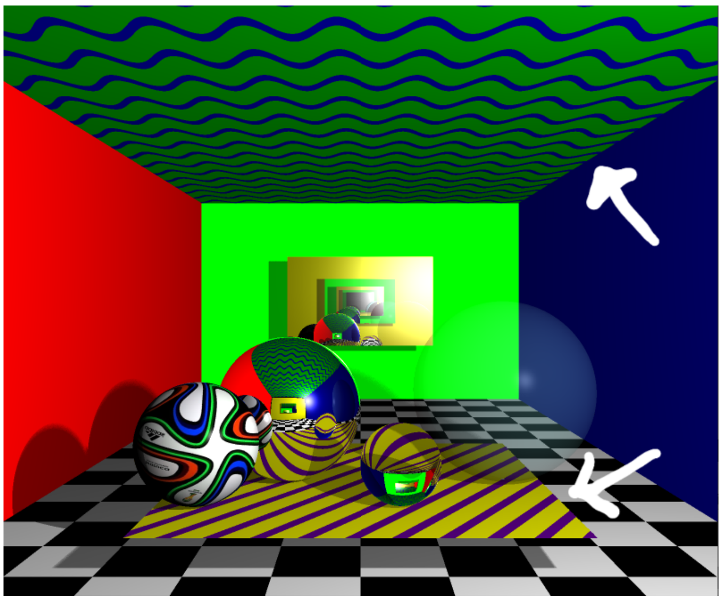
Anti-aliasing

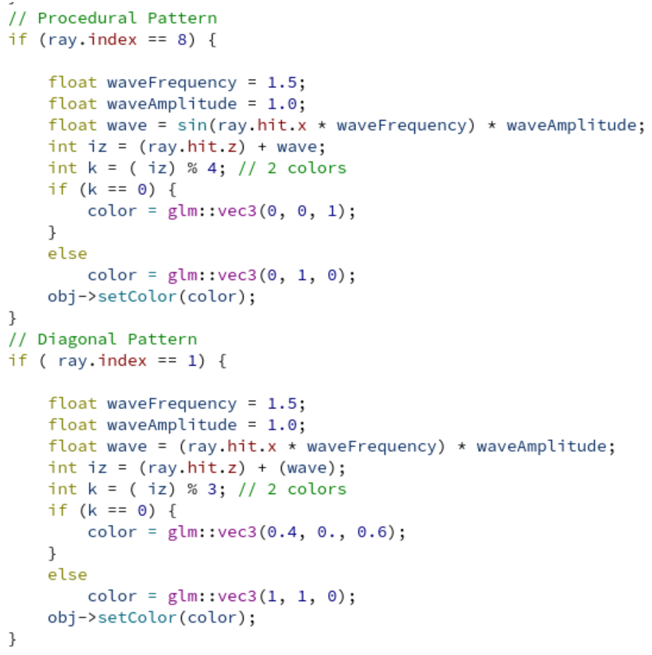

Anti-Aliasing


No Anti-Aliasing

To improve the image quality and reduce the appearance of jaggered edges, anti-aliasing has been implemented . The implementation involved subdividing each cell of the image plane into smaller sub-pixels. For each sub-pixel generated a primary ray and traced it using the trace function to obtain a color value. The color values of all sub-pixels within a cell were accumulated and averaged. Finally, the color is applied to each corresponding cell resulting in a smoother and more visually appealing image.

Procedural pattern





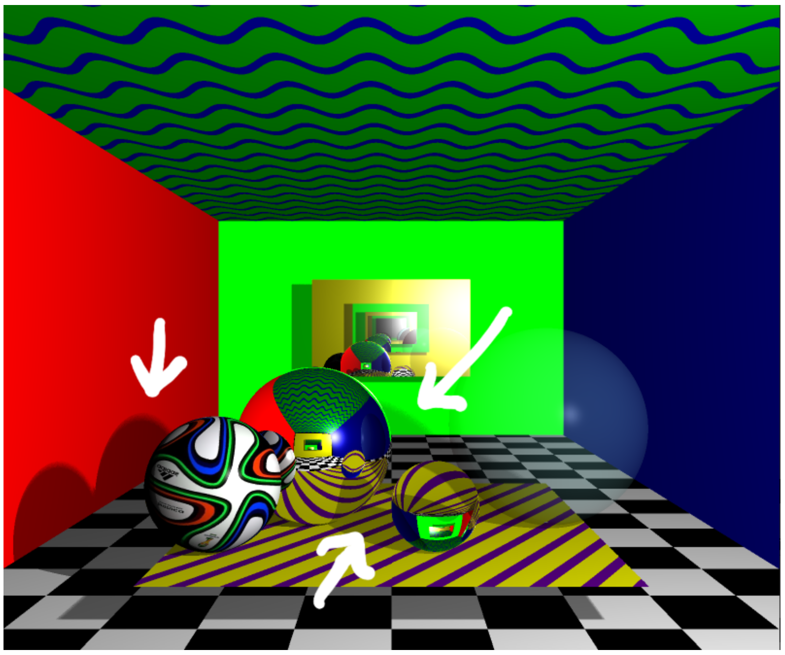
```
// Procedural Pattern
if (ray.index == 8) {

    float waveFrequency = 1.5;
    float waveAmplitude = 1.0;
    float wave = sin(ray.hit.x * waveFrequency) * waveAmplitude;
    int iz = (ray.hit.z) + wave;
    int k = ( iz) % 4; // 2 colors
    if (k == 0) {
        color = glm::vec3(0, 0, 1);
    }
    else
        color = glm::vec3(0, 1, 0);
    obj->setColor(color);
}
// Diagonal Pattern
if ( ray.index == 1) {

    float waveFrequency = 1.5;
    float waveAmplitude = 1.0;
    float wave = (ray.hit.x * waveFrequency) * waveAmplitude;
    int iz = (ray.hit.z) + (wave);
    int k = ( iz) % 3; // 2 colors
    if (k == 0) {
        color = glm::vec3(0.4, 0., 0.6);
    }
    else
        color = glm::vec3(1, 1, 0);
    obj->setColor(color);
}
```

A wavy pattern has been created on the ceiling and a diagonal pattern on the desk using an equation. The equation involves an amplitude (1.0) and frequency (1.5), which are used to calculate the wave based on the sin (ray's x-coordinate) . The resulting value is added to the ray's z-coordinate, and two colors (blue and green) are applied to create the desired wavy pattern.For the diagonal pattern the ray z is added with no sin calc therefore just diagonal.

Shadows



Shadows are cast by objects in the scene based on their interaction with light sources. A shadow factor of 0.3f for refractive and transparent spheres and 0.5f for all other objects. Additionally, for all the shadows a soft shadow is applied at the edges of the shadows.This is achieved by averaging the shadow intensities over multiple rays,resulting in a softer, more realistic shadow appearance.

Overall the program has succeeded in having a good visual appearance . A further improvement can be made with style of coding, and adding comments so that other users can have easier understanding of what the program lines does.

Declaration

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.
Name: Ismail Sarwari
Student ID: 73712637
Date:02/5/23

Build Command:
Unzip Assignment2 zip file and open qt creator.
Click File -> Open File or Project -> (click the cmake file within the Asssignment2 folder)
Click on Assignment2 -> RayTracer.out -> Source Files -> RayTracer.cpp
Finally click the green play button!

Note: with ant-aliasing off the program will take about 10 second to build. With anti-aliasing turned on the program takes about 1min.