COSC363 Assignment 2

Ray Tracing

Artemis Hingston 83986198

May 2024

0.1 Introduction

This project was definitely a challenge, and I would say I am most proud of how the scene looks when both anti-aliasing and soft shadows are enabled, especially with how smooth the shadows look - even if it takes and hour to load, a 600x increase from running the standard scene.
The part I struggled most with, especially at the start, was my conceptualisation of these objects, and the scene in 3D space. Mostly with the depth of the scene, and some of the orientation of objects, and for a long time I didn't think that the planar mirror was working, as it only reflected the table and objects, and then only either the back/North wall, or the right/West wall, depending on the angle. After changing some of the dimensions of the surrounding box, decreasing the overall depth of the scene, and angling the mirror so that it no longer aligned with any of the three axes, I managed to get a better understanding of how the objects were working together. I definitely found this less intuitive than working with the Alien World OpenGL2 project, likely since I could not move the camera through the scene and change perspective that way.
That being said, this final version of the scene manages to look suitably 3D, and the placement of reflective and refractive surfaces, as well as shadows, go a long way it aiding that.

0.2 Features

Itemised list of all extra features

- Cone Implementation
- Cylinder Implementation
- Cylinder Cap
- Textured Sphere
- Textured Cylinder
- Refractive Objects
- Fog
- Multiple reflections
- Adaptive anti-aliasing
- Soft Shadows

0.2.1 Cone and Cylinder Implementation

The Cone and Cylinder classes were based off of the Sphere class, and the equations from lectures. For both the cone and the cylinder, the intersection calculations involve solving quadratic equations derived from their implicit surface definitions. For a cylinder, the initial intersection is calculated by ignoring the y-axis, solving the equation $(x - x_c)^2 + (z - z_c)^2 = r^2$. For a cone, the intersection takes into account the linearly varying radius along the height.
There is one pink cone on the left of the table, with a light pink cylinder around it, four cylinders as legs of the table, a transparent cylinder near the middle of the scene, a textured cylinder under the globe, and four cylinders as checker pieces on the chessboard.

0.2.2 Cylinder Cap

Giving cylinders a cap makes them look more like 3D objects. The cylinder's cap is added by checking if the intersection point lies above the top base. If so, the intersection with the cap is calculated using the equation $t_1 = \frac{y_c + h - y_0}{d_y}$. The intersection must be within the circular boundary defined by the cylinder's radius. The caps are most visible on the checker pieces on the chessboard at the right of the scene.

0.2.3 Textured Sphere

A sphere is textured by mapping its surface coordinates to texture coordinates (u, v) using spherical coordinates. The equations used are $u = \frac{1}{2} + \frac{arctan2(z,x)}{2\pi}$ and $v = \frac{1}{2} - \frac{arcsin(y)}{\pi}$. These coordinates are then used to fetch color values from the texture image. There is a textured sphere at the centre right of the scene, looking like a world mapped-globe.

0.2.4 Textured Cylinder

The textured cylinder was created by using the sphere equation for the x and z coordinates, and mapping the y coordinates the same way as one would a plane. i.e. texcoords uses $u = \frac{1}{2} + \frac{arctan2(z,x)}{s\pi}$, and texcoordt uses $(ray.hit.y - b_1)/(b_2 - b_1)$. The textured cylinder is on the right of the scene under the globe, using the same texture as the ceiling/top plane.

0.2.5 Refractive Objects

The refractive sphere found on the left of the scene, above the cone, has a refractive index of 1.5, like a glass sphere. The scene behind the sphere appears inverted, and the planar mirror and subsequent reflection are strongly visible.

0.2.6 Fog

Fog effect is implemented by blending the object's color with the fog color based on the distance from the camera. The scene with fog enabled can be seen in fig.1b, and compared to the other variations.

0.2.7 Multiple reflections

The multiple reflections can be seen on the large planar mirror at the rear of the scene, reflecting off of the reflective sphere on the right of the scene, and so forth.

0.2.8 Adaptive anti-aliasing

Anti-aliasing reduces visual artifacts and makes the overall image 'smoother', and is particularly visible when looking at edges or sharp lines, such as on the chess board, or the circumference of the spheres and their shadows. This is implemented by recursively subdividing each pixel into smaller sub-pixels and casting rays to sample colors. If the color variation between adjacent pixels exceeds a threshold, further sub-division occurs. The process continues up to a specified recursion limit. The final color is the average of the sampled colors, providing smoother transitions and reducing jagged edges. The scene with anti-aliasing enable can be seen in fig.2a, and compared against fig.1a.

0.2.9 Soft Shadows

Soft shadows look more realistic than hard shadows, as real-world scenarios do not tend to have point lights that result in hard shadows. Soft shadows are generated by sampling multiple shadow rays within a small circular region around the light source using glm::diskRand(r). For each sample point, a shadow ray is traced to determine occlusion. The colors from these rays are averaged to produce a shadow that varies in intensity, resulting in softer shadow edges and more realistic shading. The scene with soft shadows enabled can be seen in fig.2b, and compared against fig.1a.

The effects of both anti-aliasing and stochastic sampling for shadows are most prominent in fig.3, where the anti-aliasing smooths out the otherwise 'spotty' shadows, giving a much more realistic effect.

0.3 Estimated Timings

When run on a Linux VM, estimated run times are as follows:

- Standard < 6 seconds
- Fog only < 6 seconds

- Soft shadows only $<$ 1 minute
- Anti-aliasing only $<$ 6 minutes
- Soft shadows and anti-aliasing $\simeq$ 1 hour

0.4 Build

The program should be able to be built and run as per *Compiling COSC363 Labs with VSCode (Linux)*
It was created and tested in a linux environment, using Visual Studio Code.
Several globals at the start of the document can be modified for testing with and without anti-aliasing, fog, and soft shadows.

0.5 References

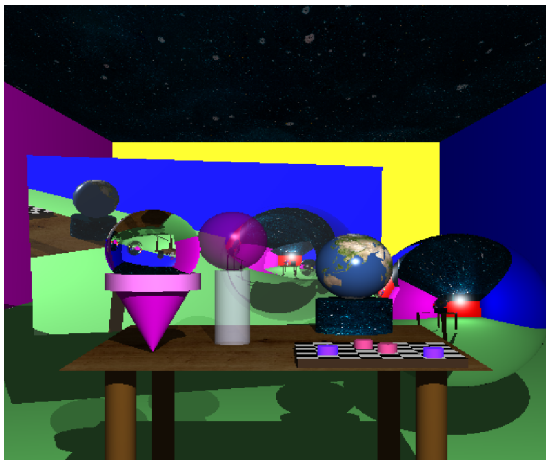
Several textures have been used in the creation of this scene.
The oak texture used ofr the table is from Architextures, https://architextures.org/textures/785
The texture used for the ceiling and pedestal of the globe is from FreePick,
https://www.freepik.com/free-photo/white-blobs-dark-blue-water_2233183.htm
The globe map texture, as well as code and processes from the lecture and lab material for COSC363 have also been used.
The equations for the textured sphere were found at https://en.wikipedia.org/wiki/UV_mapping

0.6 Declaration

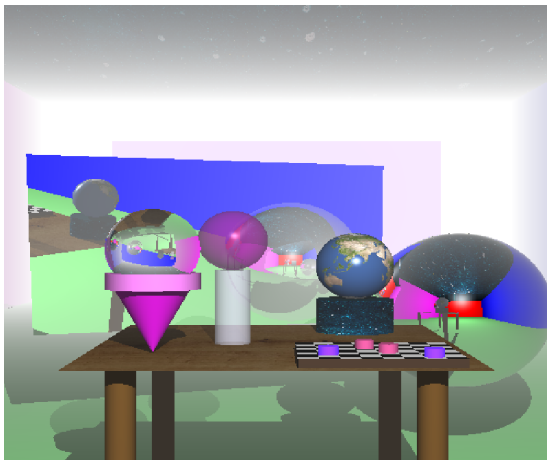
I declare that this assignment submission represents my own work (except for allowed material provided in the course), and that ideas or extracts from other sources are properly acknowledged in the report. I have not allowed anyone to copy my work with the intention of passing it off as their own work.
Name: Artemis Hingston
Student ID: 83986198
Date: 03/06/2024

0.7 Appendix

0.7.1 Images

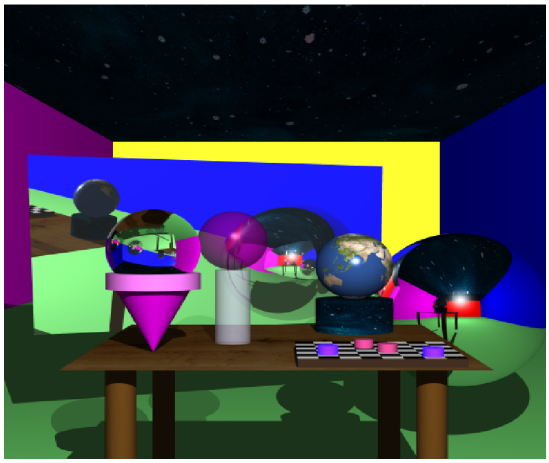


(a) Standard Scene

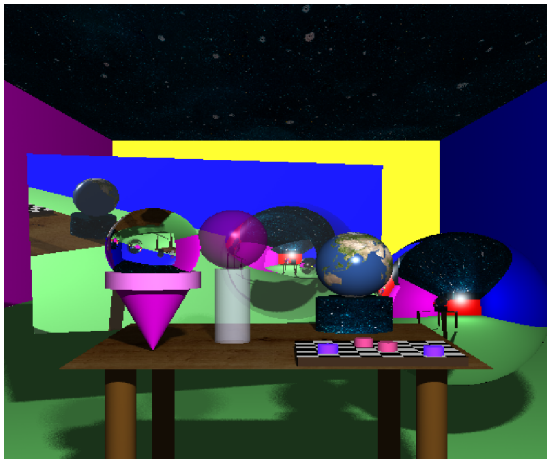


(b) Scene with fog enabled

Figure 1: Scene variations



(a) Scene with adaptive anti-aliasing



(b) Scene with soft shadows

Figure 2: Scene variations cont.

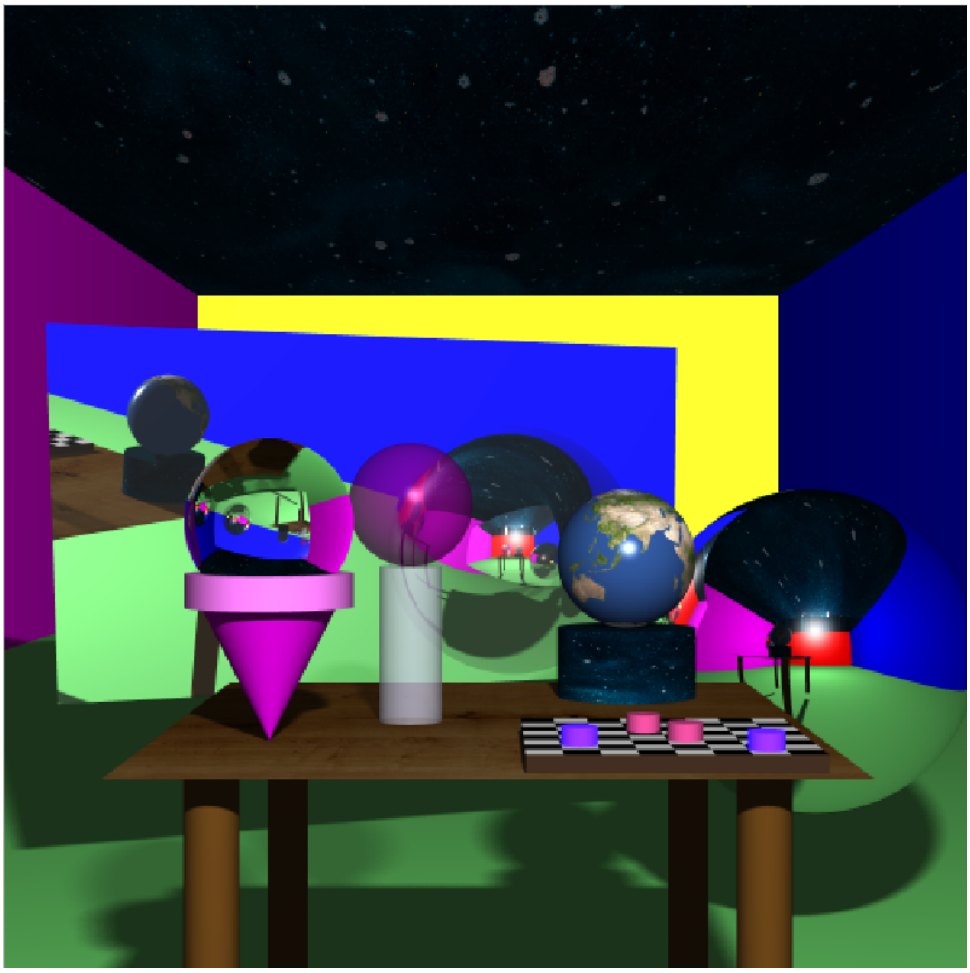


Figure 3: Scene with soft shadows and anti-aliasing