Sam Willems COSC363 Assignment 1 Report

swi126 - 31643284

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: Sam Willems
Student ID: 31643284
Date: 28 March 2024

1000x1000px, 10 Max Steps, AA on, Soft Shadows with 10 samples, Time: about 5 minutes

Description

This ray tracing program includes all of the required base implementation details, as shown in the image on the title page. The base program was taken from the work done in lab 8, with the required additions. The entire program was developed on my personal computer using Visual Studio on Windows 11. For reference to the generation time of the images displayed in the report, the relevant hardware specs of my PC are as follows:
CPU: AMD Ryzen 5 5600X
GPU: Nvidia RTX3060ti
RAM: 32Gb DDR4 at 3200Mhz

Extra Features

Cylinder

This is the first extra feature added after the required features. The cylinder is hollow and does not feature a cap, despite trying to add this for quite a while. This was implemented by creating a cylinder class that inherits from the SceneObject class. It implements intersection using the quadratic formula to find the intersection points between the ray and cylinder. A cylinder is shown in the title image, on the right.

Refractive Sphere

The implementation for the refractive sphere is done in the trace function, and follows closely to the notes provided by the COSC363 lecturers. This uses Snell's law of refraction to determine the direction and thus colour of the ray as it is refracted by the sphere. This colour from the ray is mixed in with the object colour based on the refraction coefficient. This is shown in the middle of the title image.

Multiple Reflections via Parallel Mirrors

Reflections were added in the trace function by calculating the direction of the outgoing ray using the angle of the incident ray and the normal of the surface at that point. Multiple reflections were added by simply adding another mirror in parallel to the one at the back of the room, behind the camera. This is shown in the title image.

Anti-Aliasing using Super Sampling

Anti-Aliasing is used to reduce the jagged edges formed when rendering graphics that occur due to the nature of using a grid of square pixels to display the image. I implemented this using a technique known as super sampling. Instead of sending one ray through the centre of the pixel as done without anti-aliasing, the pixel is divided into four subpixels in each corner of the pixel, and then a ray is sent through the middle of each of these. The colour values found from tracing each of these rays is taken and then the final colour of the pixel is the average of these colours.

Without Anti-Aliasing, 500x500, 5 Max steps, 4.5 seconds.

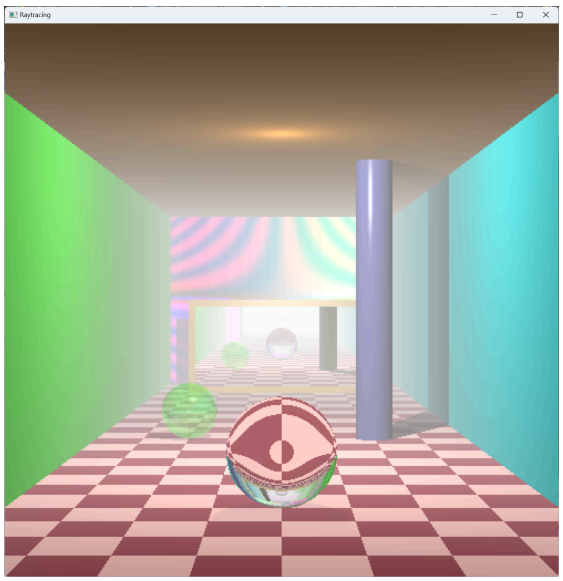
With Anti-Aliasing, 500x500, 5 Max steps, 18 seconds.

Procedural Pattern

A procedural pattern was added by taking the ray hit y and z values when it hit the back wall, and using these as inputs for mathematical functions to colour the wall. The x values was not used as this is the same for all points along the back wall. The colour values of each pixel are defined as follows:

```
// Procedural pattern on back wall
if (ray.index == 1) {
    color = glm::vec3(
        sin(ray.hit.x * ray.hit.y * 0.1) / 2 + 0.5,
        sin(ray.hit.x / 10) / 2 + 0.5,
        sin(ray.hit.x / ray.hit.y * 0.1) / 2 + 0.5);
    obj->setColor(color);
}
```

Fog



Fog was added by using a function that would mix in the colour white to the colour of the pixel, based on the distance that the ray had to travel through the scene. This is slightly different then the way provided by the lectures using the z-coordinate, as this would mean that a ray that bounced off a far away mirror and then back close to the origin, would have little fog, despite having travelled a large distance. The amount of fog mixed in is described using an exponential formula, as described in the code block below:

Using Fog, fog growth rate: 0.01, fog start distance: -50, 500x500, 5 Max steps, 4.6 seconds.

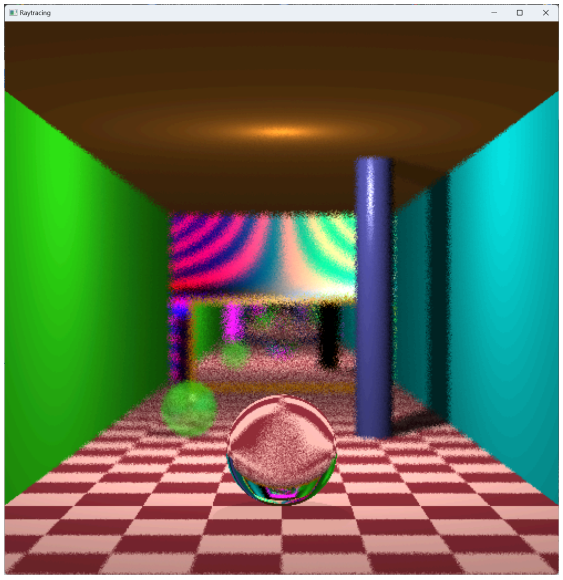
```
if (USE_FOG) {
    float lambda = 1 - glm::exp(FOG_GROWTH_RATE * (-FOG_START_DIST - ray.dist));
    lambda = lambda < 0 ? 0 : lambda;
    color = (1 - lambda) * color + lambda * glm::vec3(1, 1, 1);
}
```

Soft Shadows

Soft shadows are the fuzzy outlines of shadows, as opposed to a straight line between shadow and non shadow. This is often the case in real life, and comes from the light not being emitted from a single point, but from an area of emitted light and due to the different angles between the light source and object, edges slowly transition from full to no shadow. This is modelled in my program by taking a number of samples that are a close but random distance from the light source to the object, and taking the average colour of the ray from that point near the light to the object. On the edges of intersecting objects, the ray sometimes passes through and sometimes does not, leading to the fuzzy look.

Soft shadows with 5 samples, light radius of 10, 500x500, 5 max steps, 19.0 seconds.

Depth of Field



Depth of field is an optical effect that is often found in cameras, where a lens has a certain focal point where the image is in focus. Objects further away from this focal point are blurry. This is due to the image sensor not being a single point, but an area where light rays pass through the lens to hit. Due to the way that the lens bends the light, multiple places in the scene may hit the same area on the sensor if the ray does not pass through the focal point of the lens and thus the image becomes blurry. This effect worses as the ray becomes further from the focal point. This is modelled in the program by taking a number of samples that are close but randomly offset from the eye position by the value defined as the aperture, and passing them through the focal point. These rays return a colour that is then averaged to get the pixel colour. This means that objects in the scene that are far from the focal point have a wide area of random points selected to colour the pixel, leading to the blurry effect.

Depth of field with 5 samples, 80 focal distance and 0.7 aperture size, 500x500, 5 Max steps, 23.5 seconds.

References

Any code snippets that were derived or inspired by others come from the lecture and assignment notes provided by the COSC363 teaching team.