COSC363 Assignment 2 – Ray Tracer report

Joan Claire Teves – 77136301

Build process/ Environment

I developed this program using visual studio on a windows platform. Without the anti-aliasing the program takes around 15 seconds to generate output on my personal computer. And with anti-aliasing implementation it takes around 60 seconds.

Features

Cylinder Class

The cylinder class has methods that calculate the intersection and the normal. Equations used were as given in the lectures.

$$t^2\left({d_x}^2 + {d_z}^2\right) + 2t\left(d_x(x_0 - x_c) + d_z(z_0 - z_c)\right) + \left((x_0 - x_c)^2 + (z_0 - z_c)^2 - R^2\right) - 0$$

```
float Cylinder::intersect(glm::vec3 p0, glm::vec3 dir) {
    glm::vec3 d = p0 - center;
    float a = (dir.x * dir.x) + (dir.z * dir.z);
    float b = 2 * (dir.x * d.x + dir.z * d.z);
    float c = (d.x * d.x) + (d.z * d.z) - (radius * radius);
```

```
glm::vec3 Cylinder::normal(glm::vec3 p)
{
    glm::vec3 d = p - center;
    glm::vec3 n = glm::vec3(d.x, 0, d.z);
    n = glm::normalize(n);
    return n;
}
```

Cone Class

The cone class was implemented similarly to the cylinder class with the same methods calculating the intersection and the normal with different equations that was also provided in class.

$$(x - x_c)^2 + (z - z_c)^2 = \left(\frac{R}{h}\right)^2 (h - y + y_c)^2$$

```
float Cone::intersect(glm::vec3 pos, glm::vec3 dir)
{
    glm::vec3 d = pos - center;
    float yd = height - pos.y + center.y;
    float tan = (radius / height) * (radius / height);
    float a = (dir.x * dir.x) + (dir.z * dir.z) - (tan * (dir.y * dir.y));
    float b = 2 * (d.x * dir.x + d.z * dir.z + tan * yd * dir.y);
    float c = (d.x * d.x) + (d.z * d.z) - (tan * (yd * yd));
    float delta = (b * b) - 4 * (a * c);
```

```
glm::vec3 Cone::normal(glm::vec3 p)
{
    glm::vec3 d = p - center;
    float r = sqrt(d.x * d.x + d.z * d.z);
    glm::vec3 n = glm::vec3(d.x, r * (radius / height), d.z);
    n = glm::normalize(n);
    return n;
}
```

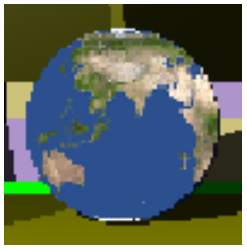
Refraction

Refraction of the right middle sphere is done by recursively tracing the rays. In each iteration there are 2 normal vectors and refraction rays calculated (the ray going in and out of the sphere). The sphere has an ETA of 1.5.

Transparency

The very left blue sphere has a transparency of 80% (0.8). This is also done by recursively tracing the rays. But in each iteration the current color is multiplied by (color * transparency)

Textured sphere



I used the equation in Wikipedia: UV Mapping to use the Earth.bmp to texture the sphere that was mentioned in the slides:

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

```
glm::vec3 center(-2.5, -3.0, -70.);
glm::vec3 d = glm::normalize(ray.hit - center);
float u = 0.5f + ((atan2(d.z, d.x)) / (2 * M_PI));
float v = 0.5f + (asin(d.y) / M_PI);
color = texture.getColorAt(u, v);
```

Textured cylinder

Similar to the sphere I used the same formula to calculate u, and the v value was d.y.

```
glm::vec3 center(-13., -15., -68.);
glm::vec3 d = glm::normalize(ray.hit - center);
float u = 0.5f + ((atan2(d.z, d.x)) / (2 * M_PI));
float v = d.y;
color = texture2.getColorAt(u, v);
```

Anti-aliasing

I used super-sampling where I divided each pixel into four sub-pixels. The four rays are generated at the center of each segment, and the average of the color values traced from the 4 rays are calculated and returned. This helped reduce the jaggedness along edges of polygons and shadows. However, this slowed down the build considerably by around 4 times slower.

```
glm::vec3 antiAliasing(float xp, float yp, glm::vec3 eye, float pixelSize)
{
    float quarter = pixelSize * 0.25;
    float threeQuarter = pixelSize * 0.75;
    glm::vec3 colorSum(0);
    glm::vec3 avg(0.25);

    glm::vec2 quarters[4]{
        glm::vec2(quarter, quarter),
        glm::vec2(quarter, threeQuarter),
        glm::vec2(threeQuarter, quarter),
        glm::vec2(threeQuarter, threeQuarter),
    };
    for (int i = 0; i < 4; i++) {
        Ray ray = Ray(eye, glm::vec3(xp + quarters[i].x, yp + quarters[i].y, -EDIST));
        ray.normalize();
        colorSum += trace(ray, 1);
    }
    colorSum *= avg;
    return colorSum;
}
```

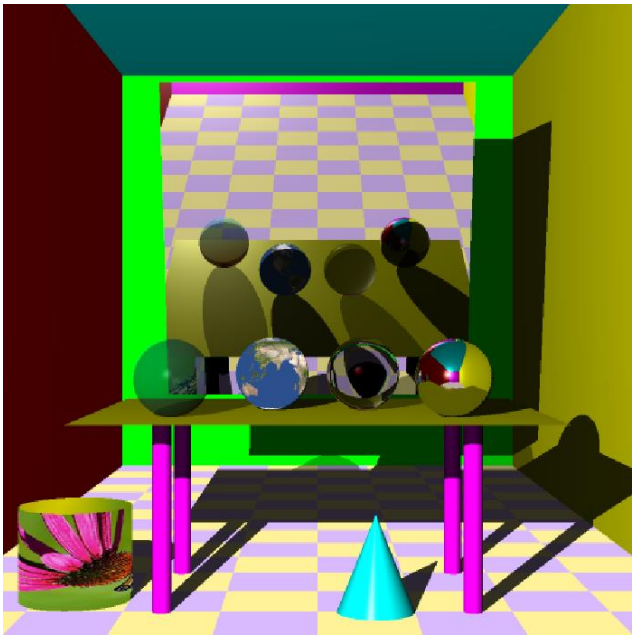


Figure 1 Scene with anti-aliasing

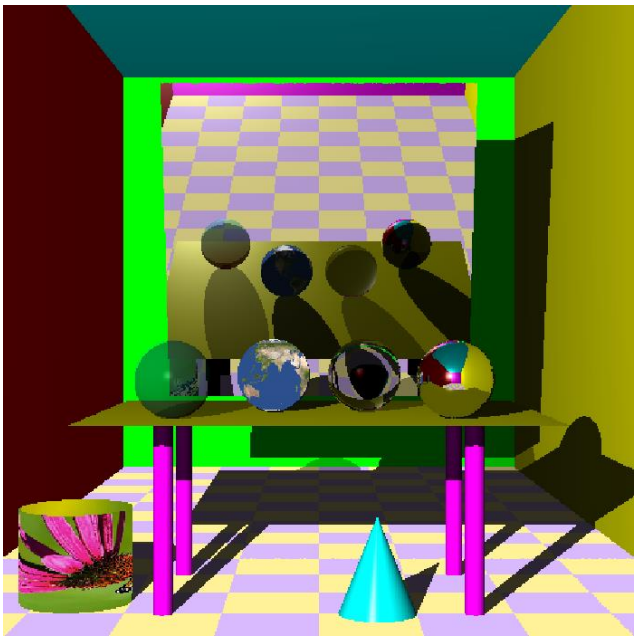


Figure 2 Scene without anti-aliasing

Fog

I used the equation provided in the lecture notes to calculate the linear variation of the fog (λ) from λ=0 (no fog) to λ=1 (max fog density). The fog can be enabled by setting the global variable fogSet = true.

$$\lambda = \frac{(ray.hit.z) - z_1}{z_2 - z_1}, color = (1 - \lambda)color + \lambda white$$

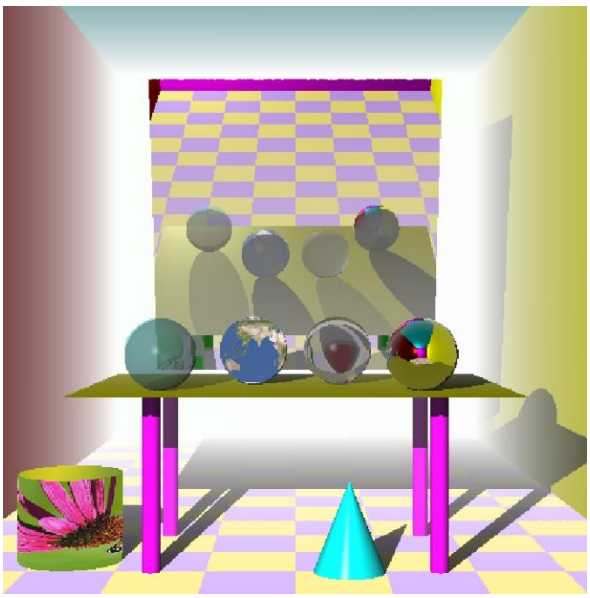


Figure 3 Scene with fog

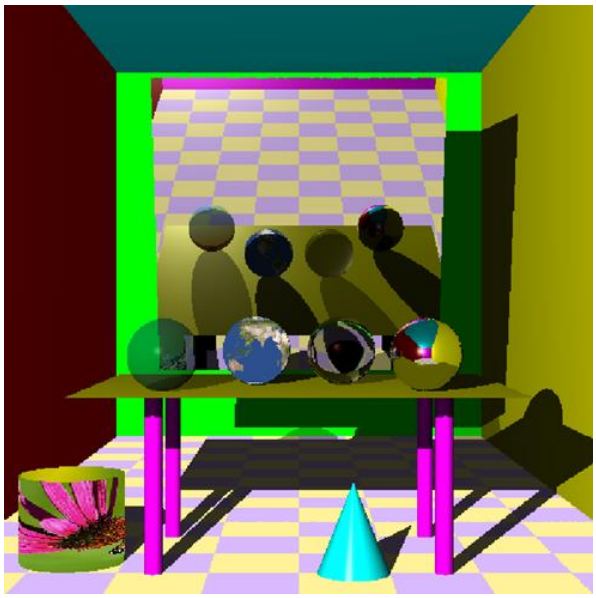


Figure 4 Scene without fog

```
float z1 = -60.;
float z2 = -131.;
float lam = fmod(((ray.hit.z) - z1) / (z2 - z1), 1.0);
color = ((1 - lam) * color) + (lam * (glm::vec3(1.0, 1.0, 1.0)));
```

Successes and Failures

Successes

- I had success in implementing most of the features that I wanted to complete. Although I hadn't really focused much on the visual or the color composition to make it aesthetically pleasing, I wanted to focus on trying to do as much implementation as I could and I believe I succeeded in the completing most of features that was on my list such as the fog, anti-aliasing, texturing, and the cylinder and cone class.

Failures

- One of the first failures I had in the beginning was not thoroughly reading the previous classes provided such as the Plane class. It was not until around halfway to doing the project that I noticed that I didn't put the points in right order to construct the plane (I thought it might've been a lighting issue).
- Consequently, I didn't realise sooner that the sceneObjects class had the setter and getter methods. I had been pushing the objects and keeping track of the index which has a hassle.

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: Joan Claire Teves

Student ID: 77136301

Date: 4/06/2024