COSC363 Assignment 2 Report

The Basic Ray Tracer

Student ID: 43832772

Name: Hamesh Ravji

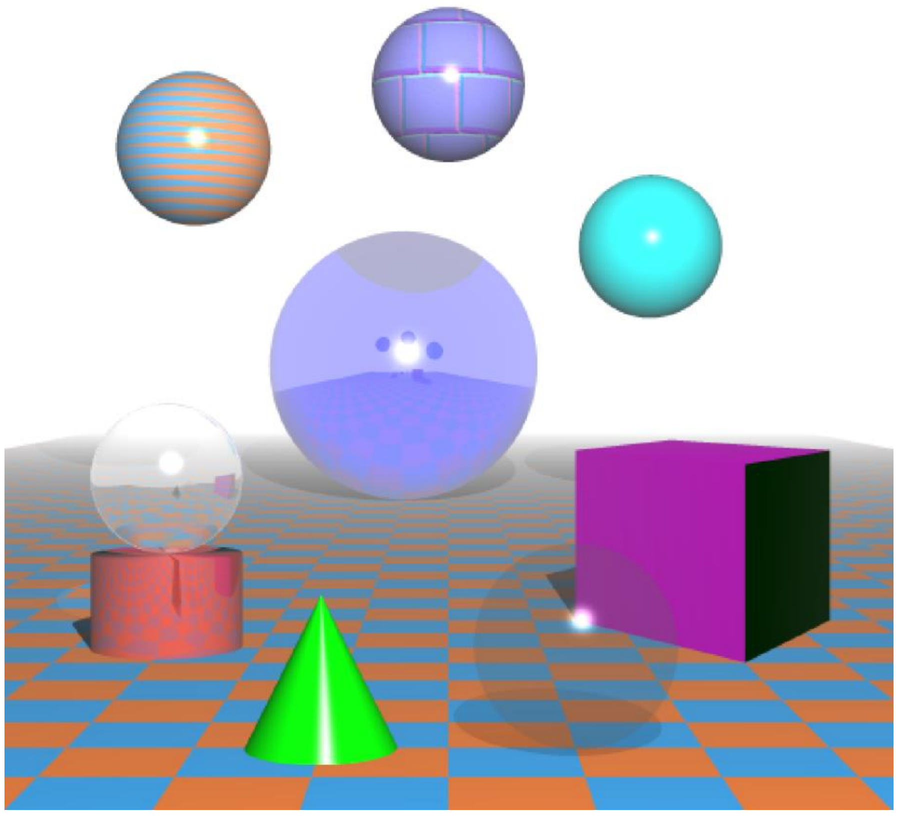

Figure 1: Assignment 2

Build Command:

g++ -Wall -o “Assignment” “Assignment.cpp” TextureBMP.cpp Plane.cpp Sphere.cpp SceneObject.cpp Ray.cpp Cone.cpp Cylinder.cpp -lm -lGL -lGLU -lglut

Build Time:

The program takes approximately 47 seconds to display the scene when executed with anti-aliasing on at a level 4. Without anti-aliasing, the build takes approximately 3 seconds. From this, we can say that anti-aliasing is very computationally expensive as just increasing the anti-aliasing level to a level 4 caused the execution time to take more than 15 times as long as it did to display the scene without anti-aliasing. This is due to the increased number of rays required to produce for each pixel being 4 where previously without anti-aliasing it was 1.

Minimum Requirements

a. Spatial Arrangement

Objects spread evenly in a way that each object can be seen clearly, including their patterns and textures.

b. Transparent

The scene contains a clear transparent sphere placed towards the front. The sphere is also very slightly reflective. When making objects transparent, shadows

```
if (obj->isTransparent() && step < MAX_STEPS)
{
    Ray transparentRay(ray.hit, ray.dir);
    glm::vec3 transparentCol = trace(transparentRay, 1);
    color = color + (0.9f * transparentCol);
}
```

must also be made lighter, as indicated in figure 2.

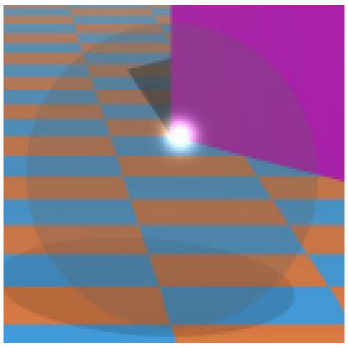

Figure 2

c. Shadows

As shown in Figure 1, all objects in the scene cast shadows and transparent and refractive objects also have lighter shadows.

d. Objects constructed using a set of Planes

The scene contains a cube as shown in figure 3, which is constructed of 6 planes; top, bottom, left, right, back, and front.

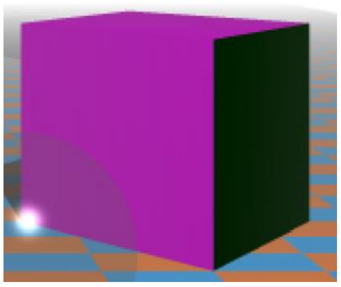

Figure 3

e. Planar surface with a Chequered Pattern

The floor of the scene contains a blue and orange chequered pattern.

Extra Features:

1. Cylinder

To construct a Cylinder, the required parameters are the coordinates of the centre of the base of the cylinder, as well as a height, and radius. The Cylinder is constructed using various equations given in the lecture notes;

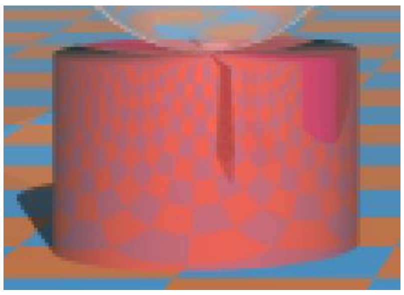

Figure 4

Ray Equation:

$$x = x_0 + d_x t; \qquad y = y_0 + d_y t; \qquad z = z_0 + d_z t;$$

Where x_0 and x_0 represent values associated with the ray's origin and d denotes ray direction.

The quadratic equation is used to find the two points of intersection, if any exist.

Intersection Equation:

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

Where R represents the radius of the cylinder.

Surface Normal Vector:

(un-normalized) $\mathbf{n} = (x - x_c,\ 0,\ z - z_c)$

(normalized) $\mathbf{n} = (\ (x - x_c)/R,\ \ 0,\ \ (z - z_c\ )/R\ )$

The Cylinder class contains methods to calculate the points of intersection and the surface normal vectors of traced rays.

2. Cone

To construct a Cone, the required parameters are the coordinates of the centre of the base of the cone, as well as a height, and radius. Constructing the cone requires some of the same equations as the Cylinder class, such as the Ray equation.

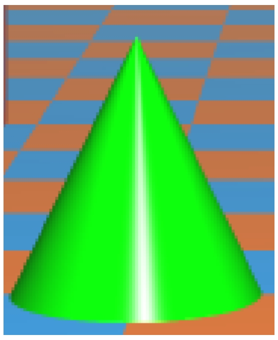
Figure 5

The quadratic equation is used to find the points of intersection, if any exist.

Intersection Equation:

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

Where R represents the radius of the cone and h represents the height of the cone.

Surface Normal Vector:

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

3. Refraction

The scene contains a clear refractive sphere placed on top of a cylinder object. This sphere is also slightly reflective to give a better representation of glass. To make the shape refractive, two normal and two refractive rays are computed representing rays going in and out of the sphere. The chosen ETA for the generated sphere is 1.003. The refractive objects also contain a lighter shadow as required.

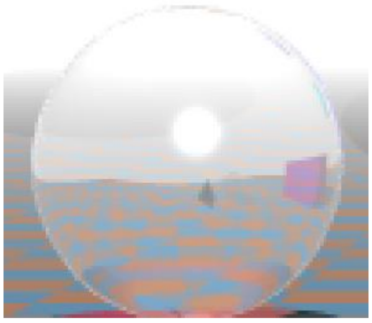
Figure 6

4. Non-Planar Textured Sphere

The sphere is textured by sphere_tex.bmp. This texture just contains some purple bricks as indicated in the image to the right.

```
if (ray.index == 6)
{
    glm::vec3 n = obj->normal(ray.hit);
    double tu = asin(n.x)/M_PI + 0.5;
    double tv = asin(n.y)/M_PI + 0.5;
    obj->setColor(texture.getColorAt(tu, tv));
}
```

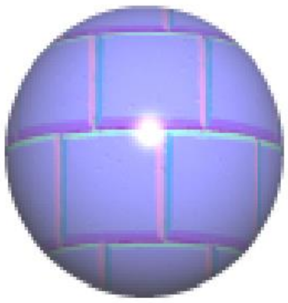
Figure 7

5. Non-Planar Sphere with Procedural Pattern

The generated pattern is generated with the code below:

```
if (ray.index == 5)
{
    //random non-chequered pattern
    float amp = 0.4;
    float ix = ((ray.hit.x + 50) * M_PI)/20;
    float hity = abs(ray.hit.y);
    float checky = fmod(hity, 20);
    if (checky < 20) ix += M_PI;
    float iy = amp * cos(ix) + amp;
    if (fmod(iy, (amp * 2)) < fmod(hity, (amp * 2))) color = glm::vec3(1, 135.0/255.0, 67.0/255.0); // ora
    else color = glm::vec3(67.0/255.0, 177.0/255.0, 1); // blue
    obj->setColor(color);
}
```

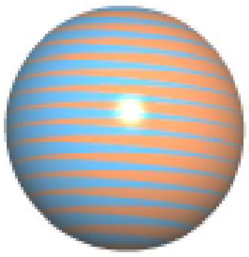
Figure 8

The generated pattern provides the sphere with the pattern shown to the right.

6. Anti-Aliasing (Super-Sampling)

Without super sampling, the result contains a finite set of rays generated through a discretized image space, as stated in the lecture notes. Result appear with distortion and jaggedness along the edges of polygons and shadows. Anti-aliasing can improve the quality of an image drastically, by dividing each pixel into four sub-pixels where a primary ray is generated through the centre of each sub-pixel and the sum of the colour values is divided by the number of sub-pixels to get the average colour value.

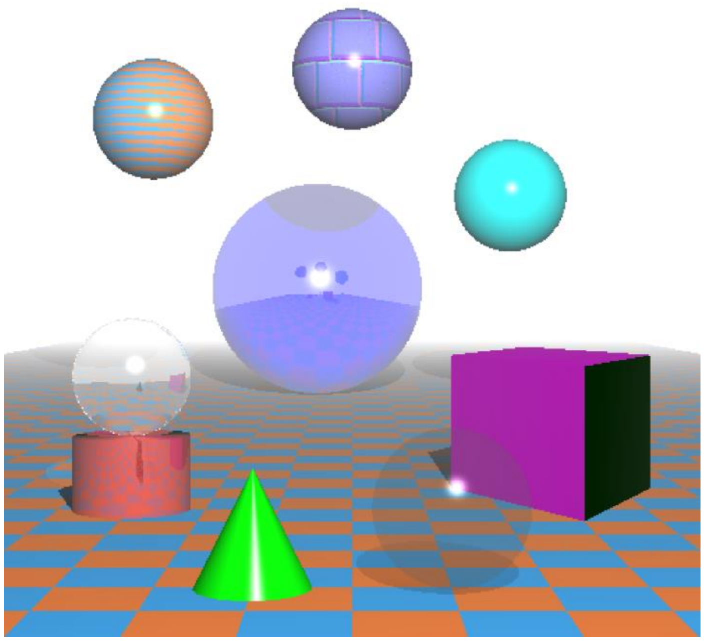

Figure 9: Without Anti-Aliasing

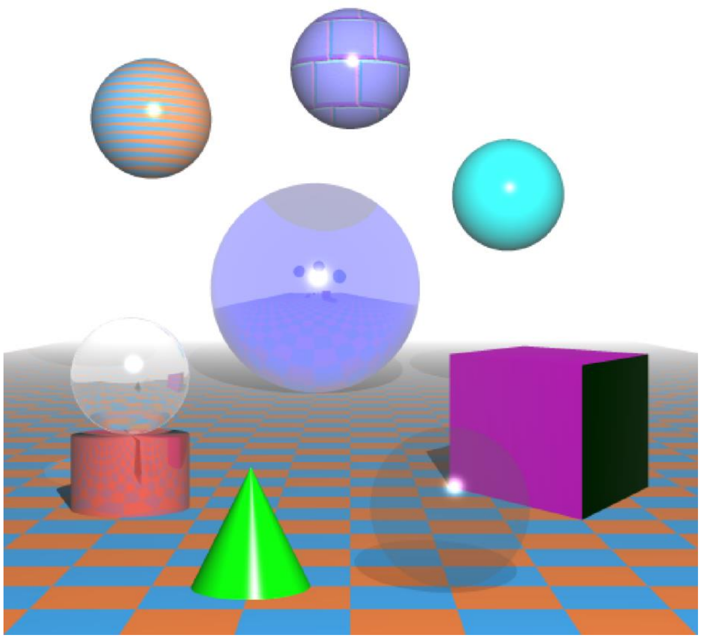

Figure 10: With Anti-Aliasing

7. Fog

Fog easy to add, only requiring two lines and settings the background colour to white.

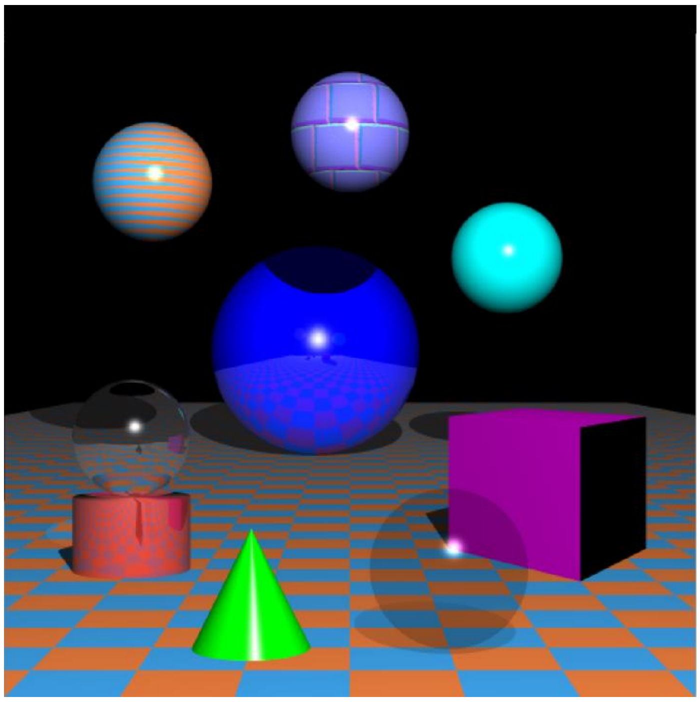

Figure 11: Without Fog

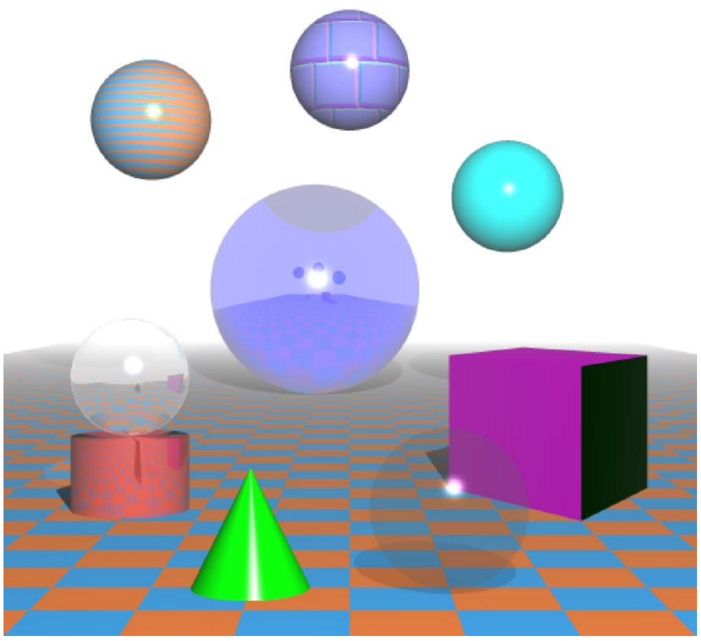

Figure 12: Without Fog

Success and Failures

One of the reasons I was not able to overcome the following challenges were time constraints.

- One of the challenges I faced but was unable to complete was implementing a second light source.
- Another challenge I faced but was unable to complete was implementing a spotlight, I found this quite difficult to implement as there were various other aspects to consider such as altering the shadow values when two shadows created from different light sources overlap.

References

1. Lecture Notes
2. Resource for Texture Mapping the Sphere
   - http://www.mvps.org/directx/articles/spheremap.htm
3. Sphere Texture
   - www.textures.com