COSC363 Assignment 2

Ollie Chick (67683982)

0 Instructions

To view the ray tracer output, run `./RayTracer`. To generate the output, run `./build-instructions`.

To generate the output then view it, run `./runme.sh`.

1 Overview

My ray tracer (Figure 1) includes three spheres (two are reflective, one is refractive), a box, a cone, a pyramid, a cylinder, a triangular prism and a planar floor.

The class SceneObject is used to represent all objects in the scene. The 2D classes Plane and Triangle, and the 3D classes Sphere, Cone, and Cylinder are all subclasses of it.

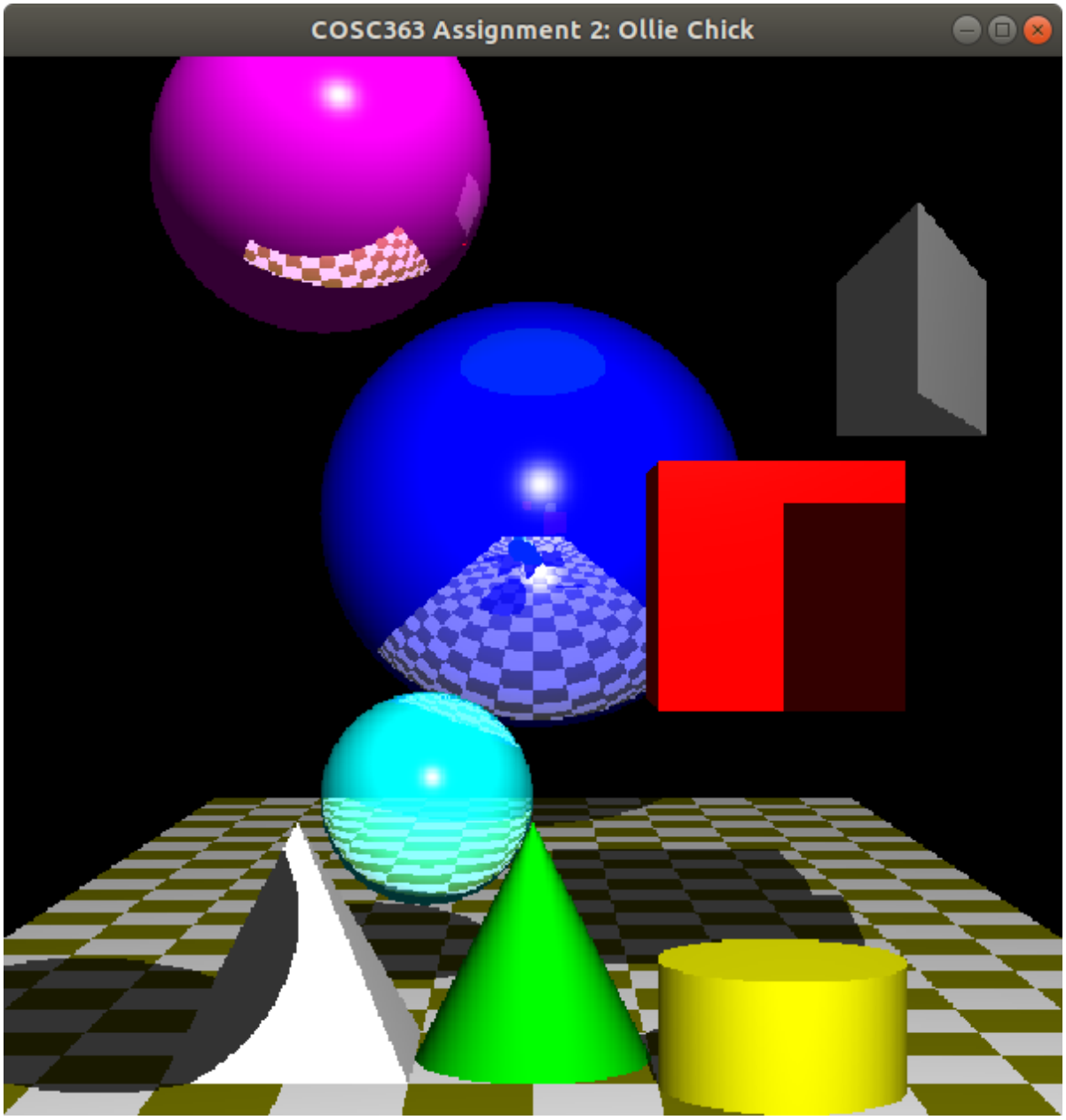

Figure 1: The ray tracer

2. Minimum requirements

2.1 Planar surface with a pattern

The chequered floor is created using a single plane. Plane has an overloaded constructor with a chequered flag (this is set to false in the original constructor), which, if set to true, renders the surface as chequered (with each square being 5 units by 5 units). It does this by overriding

SceneObject's `getColor(glm::vec3 pt)` method with a method that checks if the chequered flag is true, and if so, returns either white or the plane's colour based on what point is being coloured , and if not, returns the plane's colour.

2.2 Box

The box is created using six planes, with the chequered flag set to false.

3. Extra features

3.1 Cone

The cone (Figure 2) is created by extending the SceneObject class. It is defined by the center of the cone's base, the radius of the cone's base, its height, and its colour.

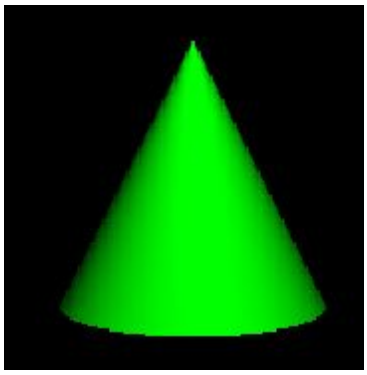

Figure 2: The cone

To calculate the distance between a ray's origin and its intersection with a cone (the intersect method), I adapted an answer from Constantinos Glynos on StackOverflow (https://stackoverflow.com/a/34184097/8355496). The solutions to the quadratic equation are found, then any intersections above the top or below the bottom of the cone are discarded.

The normal is calculated as a normalised vector from the center of the cone (at the height of the point) to the point.

3.2 Cylinder

The cylinder (Figure 3) is created by extending the SceneObject class. It is defined by the center of the cylinder's base, the radius of the cylinder's base, its height, and its colour.

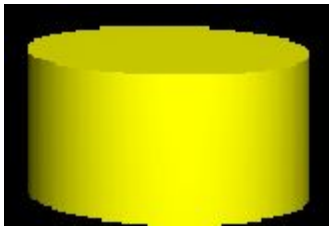

Figure 3: The cylinder

To calculate the distance between a ray's origin and its intersection with a cylinder (the intersect method), I adapted the solution in the notes. The solutions to the quadratic equation are found, then any intersections above the top or below the bottom of the cylinder are discarded.

The normal is calculated by first checking if the point lies on the top or bottom cap; if so, then they are the positive or negative y axis, respectively. If not, then it is calculated as a normalised vector from the center of the cylinder (at the height of the point) to the point.

3.3 Pyramid

The pyramid (Figure 4) is created using four triangles. The Triangle class is similar to the Plane class, except it only has three points (a, b, c), and the isInside and normal functions are adjusted to account for there only being three points.

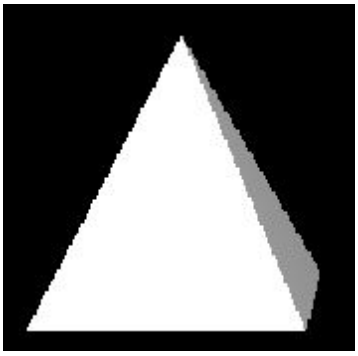

Figure 4: The pyramid

3.4 Triangular prism

The triangular prism (Figure 5) is created using two triangles and three planes.

Figure 5: The triangular prism

3.4 Refraction

Refraction (seen in the cyan sphere in Figure 1) is done using the implementation in the notes. First, the direction and initial point (first intersection) of the refracted ray inside the object is calculated, then the direction and initial point (second intersection) of the refracted ray outside the object is calculated. This second ray is then traced recursively, and the resulting colour added to the colour returned by the trace function.