# Computer Graphics Assignment 2 Report

Name: Ruofei Qian

Student ID: s1872408

E-mail: s1872408@ed.ac.uk

This coursework is aimed to fill in a partial implemented ray tracer algorithm using C++ based on a given structure.

## 1. Basic ray tracer

By applying light source and camera data from a json file, the ray tracer should be able to render different kind of objects to the result ppm format image.

After ray casting, the render function in RayTracer class can then calculate an initial result iteratively for each pixel In each calculation in the loop, a new Ray instance is created to work with all objects, whether there is a hit happen, if it is, that pixel buffer will return color of that shape, otherwise return the background color.

### 1.1 Ray casting

The rays can be seen as multiple vectors coming from an original point to all the pixels. So, after getting the width and height from the camera class and setting the start point in the coordinate, we can use for loops to generate all the ray directions from the origin to the canvas. if the ray vector does not collide with any object, the return color should be the background color (obtained in the scene class from json file), otherwise it should return the color of that object. So that all the rays can be casted by setting their origins and directions, the results can be obtained from a light source instance.

In the RayHitStructs class, the Ray struct should contain attributes include ray type, two Vec3f which represent original point and its direction respectively, a constructor that receive these two Vec3f as parameters to set both values and another getPoint function to calculate the distance between the origin and final hit point; similarly, the Hit struct contains a Vec3f point showing the hit point, as well another Vec3f normal to calculate the normalized result, a float t to calculate the getPoint in Ray struct, and finally a bool value isHit to record whether this ray has hit an object.

### 1.2 Camera

Camera abstract class obtains the width and height of the canvas from json file, bases on given camera type to identify which kind of camera should be created, PinHole or ThinLens.

### 1.3 Light source

This class get relevant position and intensity from the json file, as well I calculate all the ray directions here, so I need virtual getters that to be implemented in AreaLight or PointLight class to return the original light point and the directions array that contains all the ray direction.

### 1.4 Shapes

Here I implement the Sphere class, there is an intersect function to be implemented from the super class Shape as the intersect should be distinctive for different objects. Here for the sphere class, the formula is based on the quadratic formula because the ray may hit the sphere twice at a near point as well a far point, here I only care about the near point. Then all calculated result will be saved in an Hit instance for ray tracing. There is another color function which return the color of current sphere.