Report 1

Hrishikesh Jedhe Deshmukh - 210050073
Sankalan Baidya - 210050141

March 2024

1 Whitted Ray Tracing

1.1 Point Lights

We created a new class inside ***light.hpp*** and ***light.cpp*** named `point_light_t` to represent point lights. It computes direct illumination using the **Phong Model**. The **diffuse component** of direct lighting is just, the color of the light multiplied by the diffuse coefficient multiplied by the cos angle of the incident ray and the outside normal. The **specular component** is the multiplicative aggregation of the color of the light, the specular coeffient and the cos angle between the reflected ray and the view direction. The ambient component is always a constant for a particular scene

For handling shadows we added a new procedure ***compute_shadows*** inside the `point_light_t` class. Given a hitpoint for an object, it shoots a shadow ray towards the light direction and sees if the ray intersects with any object in front of the light. If so, the diffuse and specular component of the direct lighting is not outputted for that pixel(In the case lighting = ambient), such that this pixel is darker than if it were under lighting.

1.2 Whitted Integrator

We added code to `whitted_integrator_t` to handle secondary reflected and transmitted rays. To do this, we calculated the vectors corresponding to reflected and transmitted rays and then called the function recursively on them. We also added an attenuation factor to account for the fact that the intensity of light rays decrease with each bounce, which is handled by the transmission and reflection cofficients. In this we also had to account for total internal reflection. We just check if the value of *sine* of the angle is more than 1 or not using **Snell's law**. If it is, we just neglect the transmitted recursion and only consider the value of illumination obtained from reflected (and direct) rays.
We also had to create the **Cornell scene**. So for that we had to add the plane object. We represented a plane object using a point and 2 vectors. The point represents one of the four corners of the plane, while 2 vectors represent the sides that intersect at that point. Lengths of those vectors denotes the extent of the plane. We had to also add code for calculating intersection of a ray with the plane. For that we used the parametric form of the ray and the equation of the plane obtained from its normal. Then checked if the point lies in the extent of the plane. We approximated the area light by using the **pointlight**.