CSC 305 Assignment 1 – Ray Tracer

Leanne Feng
V00825004

Explain the design of a ray tracer from the perspective of object-oriented programming. Explain the relationships between the hitable, the material, the hit_record, camera (Translate these names to equivalent concepts if you wrote your ray tracer differently.):

The key point to design of ray tracer is to send rays through pixels and compute what color is seen in the direction of those rays. We need to calculate which ray goes from the eye to a pixel, when ray his an object, the output will be the color of the intersection point pixels. The number of rays in a scene is same as the product of height and width of the scene. We can use hit_record structure to output the intersection point's' normal vector. The material pointer in the hit_record will tell us how rays interact with the surface. When color() routine gets the hi_record it can call member functions of material pointer to find out what ray, if any, is scattered. Camera class represents the virtual scene. It tracks location and facing of the camera, and may be responsible for generating viewing rays. We can also simulates different effects such as metal, refraction, reflection and so on by using material class.

Explain how a ray tracer implements the transport of light.Talk about how your rays bounce off objects:

First of all, rays are generated by using equation: r(t) = o + td, where representing origin, ray parameter, and direction. Ray bounce off objects need to classify by its material. For lambertian materials, reflected ray is scattered randomly. It can either always scatter and attenuate by reflectance R, or it can scatter with no attenuation but absorb the fraction 1 - R of the rays. Or it can be a mixture of the two above. The metal material, the reflected ray direction of a ray v is v + 2B where B is dot(v, N), where N is normal vector. In dielectric material ray can refract using n sin(theta) = n'sin(theta'), where n and n' are the refractive indices and reflect ray with angle.

Explain how diffusion of light is modeled by a Lambertian (diffuse) material:

First of all, a random_in_unit_sphere() function is created for a random vector that is smaller than 1. Then we will get a new random reflected point target with same origin by sum up normal vector, random vector and random_in_unit_sphere(). Subtract target vector by p to get a reflected new ray vector.

Combine the original vector p and the reflected ray vector by calling ray(), the product will be the ray hits a diffuse material object then reflects randomly.

Further explain how a ray tracer implements the transport of light. Relate your ray tracer to Kajiya's Rendering Equation:

The rendering equation describes the total amount of light emitted from a point x along a particular viewing direction. The outgoing light (Lo) = emitted light (Le) + reflected light. The reflected light itself is the sum from all directions of the incoming light (Li) multiplied by the surface reflection and cosine of the incident angle, where the equation is very similar to our assignment color computing equation.

Explain how reflection and refraction of light are modeled by metal and dielectric materials:

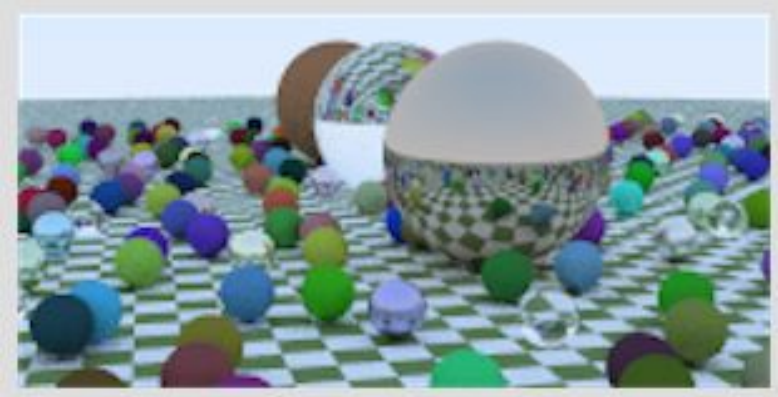
In metal material, it uses reflected ray equation: v - 2*dot(v,n)*n, the addition of fuzz*random_in_unit_sphere() simplys makes the metal material closer to the real metal fuzziness using fuzz parameter. The bigger the sphere, the fuzzier the reflections will be. In dielectric material, the reflected ray also uses equation: v - 2*dot(v,n)*n, the refracted ray uses the equation: n sin(theta) = n'sin(theta'), where n and n' are the refractive indices and reflect ray with angle. The refractive function can also check if a ray can be refracted.

Explain how a ray tracer can be used to model a lens:

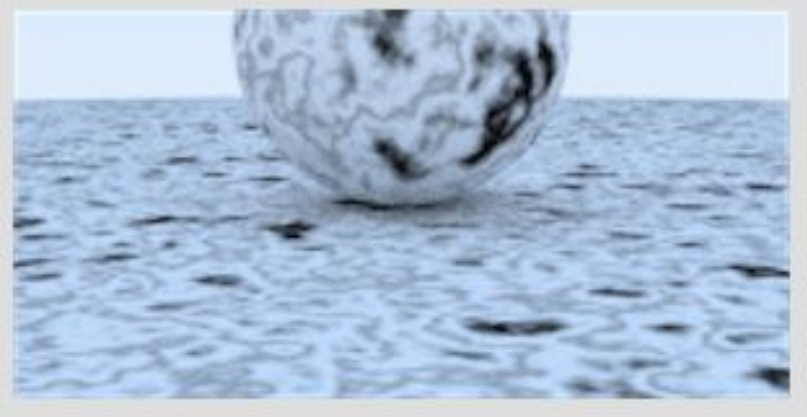
The "aperture" is a hole to control how big the lens is effectively. For our code, We could simulate the order: sensor, then lens, then aperture, and figure out where to send the rays and flip the image once computed. Start rays from the surface of the lens, and send them toward a virtual film plane, by finding projection of film on the plane that is in focus(at the distance focus_dist).

Screenshots 1:

Materials including: multiple sphere, solid textures, metal material, dielectric material, anti-aliasing, defocus blur, background.



Screenshots 2: A marble-like effect(perlin noise)



Screenshots 3, 4: including rectangles and rectangular lights, translate, rotate, volumes