Jason Louie
V00804645

CSC 305 Assignment 1 Report

Standard Requirements

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

-Hitable is a simple header file that contains the struct of hit_record as well as a pure virtual function “hit” that is used in sphere.h to see if the ray comes in contact with the object.

-Material is an abstract class that holds the behaviour of objects when rays are projected onto it. Within the header file consists of the different types of materials like lambertian, metal and dielectric. Each material conveys different behaviours; metal for example, reflect some of the light and scatter in different directions.

-The hit_record header is defined as a log of all rays that have hit an object. Providing details like the position and normal of the ray and object.

-The camera aids in antialiasing: a sphere attains its “roundness” by taking multiple camera positions(viewpoints) on that object.

When put altogether, we create a relationship when a ray shoots toward an object: if the ray comes in contact with an object(hitable), the object is recorded into hit_record and takes in the information like the position and material type. Afterwards the ray’s light acts accordingly to the material’s property to reflect or refract through the object. For the final touches, the objects are sharper according to the closer distance the object is from the camera.

Explain how a ray tracer implements the transport of light.

- Talk about how your rays bounce off objects

Rays may bounce or pierce through the object based on the object’s material. A ray “r” onto a metal object for example, is bounced in another direction by means of calculating r + 2*b where “b” is a vector. Since the newly produced ray “r2” is being shot in the direction opposite to “r” with weaker strength, the formula must be subtracted instead of using addition. The final touches are done by multiplying “b” with a unit vector for the direction and .

- Explain how diffusion of light is modeled by a Lambertian material

A ray onto a lambertian model may behave in two different ways: the first being that it scatters by reflection with weakened strength (attenuation) or scatters by absorption of the model based on its darkness. Alternatively, it can also be a product of both reflection and absorption.

Advanced Requirements
Further explain how a ray tracer implements the transport of light.

- Relate your ray tracer to Kajiya's Rendering Equation:
    - https://en.wikipedia.org/wiki/Rendering_equation

Form Kajiya's Rendering Equation, the radiance is associated with attenuation in a ray tracer. The more attenuation, the greater reflection radiance. The outgoing light equates to the radiance plus reflected light. All incoming light (at all directions) denotes the additional rays from different angles shooting at the object. The reflection is similar to the glass marbles case: there is little to no shadowing on the ground surface of the marble since the rays aren't absorbed.

L = radiance (brightness/intensity)
Outgoing light intensity = emitted light intensity(sending light) + reflected light
Reflected light = incoming light at all direction intensity * surface reflection and cosine of incident angle

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

Reflection
-Smooth metals reflect rays in a mathematical approach as stated in the previous question. This time instead of lambertian materials absorbing most of the ray, rays onto metal materials reflect at a stronger intensity.

- When the camera angle is "steep" or up close at an obtuse angle, dielectric materials reflect rays similarly to a mirror. At the steepest angle, the attenuation is at a maximum value of one and the rays are reflected closer in parallel with the object.

Refraction
-Due to the high attenuation of metal materials, rays do not refract onto the metal object. Instead, it is reflected as stated before with a lower attenuation.

-When an incoming ray enters a dielectric object of high transmission at some angle "a", the ray tracer should produce another ray at a different angle "b" that begins at the point of contact on the object and through and out of the object. Since the image is two-dimensional, the ray that exits from the object displays out of place. See Chapter 10's image for reference of the left sphere and notice the difference in the lining of the ground when looking through the sphere. Just like when looking down at your hand underwater, your hand may look cut off depending on the angle you're looking at.

- Explain how a ray tracer can be used to model a lens.

Given the “lookfrom” positions and “lookat” direction as stated in Peter Shirley’s book, we can gauge the distance from the lens to a virtual plane where that distance is in focus. The rays can then begin on the lens itself and shoot toward the virtual plane by computing the projection on the plane.

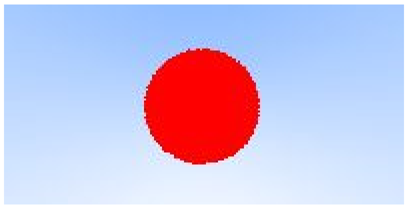
Checkpoints

Chapter 1 to 3 - Background and format:

This checkpoint has achieved a background image created in a PPM format. A file has been created in main() and every value has been written into the file. The format of the file consists of the following in integers:

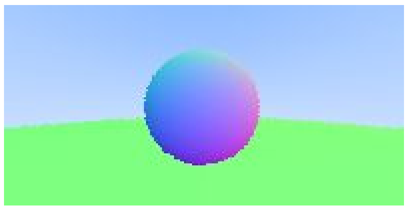
- P3 - the ASCII colour configuration
- 200 - # of columns or width
- 100 - # of rows or height
- 255 - # of max colours
- Remaining values are pixels in triplets to represent RGB values

Chapter 4 - Sphere Creation:



Two functions “hitSphere” and “colour” is used to create a red two-dimensional sphere. HitSphere is a function that returns true if the ray comes in contact with a sphere. This is found by checking the discriminant is greater than or equal to zero. The discriminant returns a negative number if the ray does not hit any spheres in its path; otherwise it has one or two points of contact on the sphere. The colour function utilizes hitSphere() by checking if the ray has hit the sphere, thus changing the pixel colour to red; otherwise it is the background.

Chapter 5 - Shading:



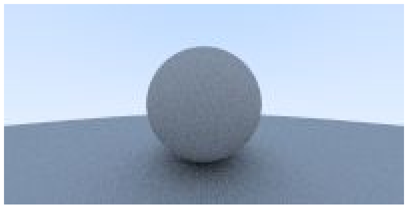
The gradient colour of the sphere is done by altering the colour and hitSphere function. It checks that if the ray hits the sphere, it can find the point on the sphere where it came in contact with it. This is used to find the vector normal to the point on the sphere. Additionally, I have set up a struct in a header file to keep track of the rays that have detected a hit for the use of creating multiple spheres.

Chapter 6 - Antialiasing:

To obtain smoother spheres, a camera class has been created and will be used as a spawn point for rays at different positions. Different positions are attained by using a random number generator provided by the “cstdlib” library by use of the rand() function. To reach values from zero to one (zero is inclusive, one is exclusive), The rand function is divided by the RAND_MAX macro since rand() returns values from 0 to the maximum number.

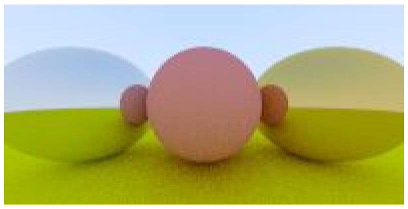
```
      (float)rand() / RAND_MAX
```
//Random number generator that produced values of n. 0 <= n > 1

Chapter 7 - Lambertian Materials:



Lambertian objects reflect and absorb rays, one more than the other depending on the darkness of the sphere. To reflect a ray, another ray is produced (at a reduced colour since some of the light is being absorbed) from the point on the sphere to another point randomized on the sphere while also tangent to the original ray.

Chapter 8 - Metal Materials:

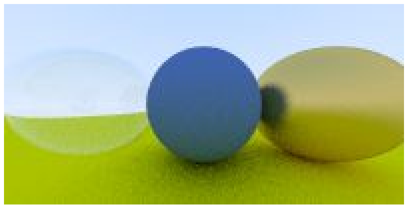


The metal spheres (left and right spheres) are achieved similarly to a sphere of diffuse material; this time reflecting the ray instead of absorbing a portion of it. This newly reflected ray is found by a ray coming in contact with a sphere and computing the following equation:

```
      ray - 2 * dot(ray, unit_vector) * unit_vector
```

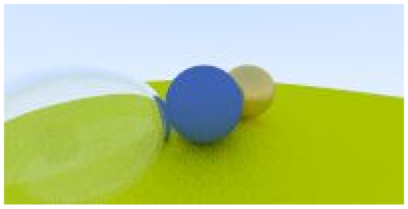
For additional effects, a metal sphere has a fuzziness trait that is prominent in larger metal spheres. Compare the right sphere in Chapter 9 and Chapter 8 to see the difference.

Chapter 9 - Dielectric Materials:



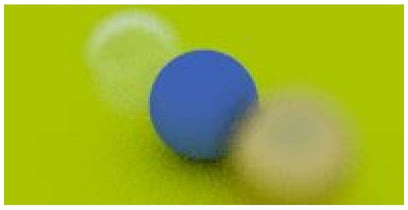
Dielectric materials (left sphere) contain combinations of reflected and refracted rays. Refraction is done by following Christophe Schlick's polynomial approximation to produce rays when shot at different and closer angles. When attenuation (labelled diminish in my code) is a larger number, the object does not absorb any of the rays.

Chapter 10 - Positional Camera:



Different camera angles are obtained by using a plane that acts as a lens. The "lookFrom" vector is the origin of camera and acts as a pivot point which can rotate the imaginary plane. We use a vector "normUp" to distinguish orientation by determining which way is up to help aid in the camera tilt and position.

Chapter 11 - Depth of Field:



The depth of field or defocus blur sharpens the pixels that are up close to the camera and dulls pixels that are further away. From the lens, the function randDiskUnit()'s purpose is to spawn random points on the lens to send rays onto the object. By using an aperture, the spheres gain depth and thickness around its edges; the bigger aperture, the more light and defocus blur.