Jason Louie
V00804645

CSC 305 Assignment 3 Report

The lights in the screenshots for chapters 6 and on seem to change from white to black during the conversion from PPM to JPG format. See PPM folder in the deliverable to see the white light instead of the images in this report.

Chapter 1

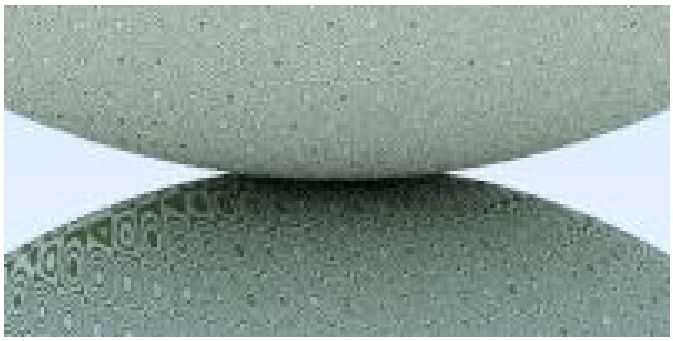
- Created a time variable and added it in the rays and camera. The camera will generate rays within the given time interval.
- Created new header file “moving_sphere.h” that creates a moving object
- Implemented random_scene in main from previous book but it crashes on my laptop, must be due to the stress on the computer

Chapter 2

- Created “aabb.h”: axis-aligned bounding boxes
- Implemented bounding_box() to implement a hierarchy of all primitives (sphere and moving_sphere) which represent as leaf nodes in a hierarchy
- Created “bvh.h” class to represent a container of hitables

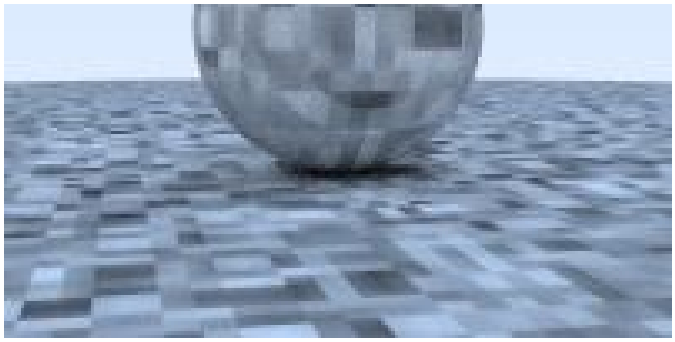
Chapter 3

- Created textured material class that takes in a pointer to a texture class
    - See screenshot below: it is hard to see since it is rendered at a low resolution but it is slightly noticeable at the left side of the bottom sphere

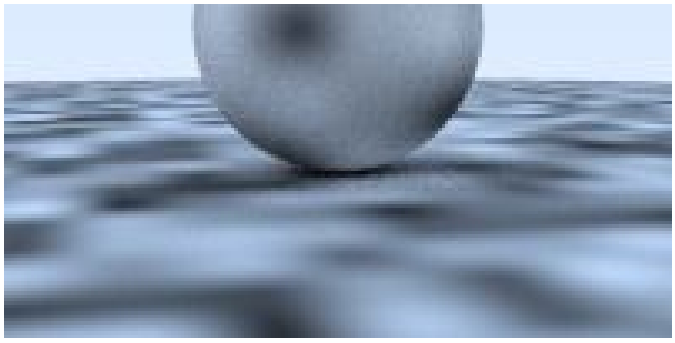


Chapter 4

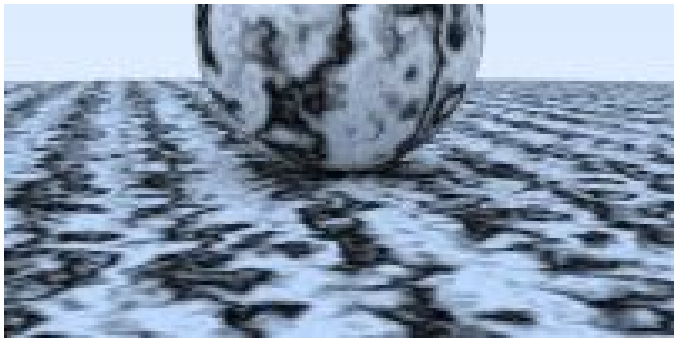
- Created “perlin” class to take in a point in space as input and produce randomized values around it for smooth and round spheres/objects
- Added linear interpolation to reduce “white noise” (screenshot)



- Used method of hermite cubic to smoothen the interpolation (screenshot)
    - Subtracting the xyz values with their floor of the respective values: x - floor(x)...



- Implemented turbulence to simulate realistic properties. This is done by finding the sum of repeated calls of noise
- Directly use turbulence by creating a marble effect by making the colour proportional to a sine curve. Then have the turbulence regulate the phase and create the “wave-like” appearance

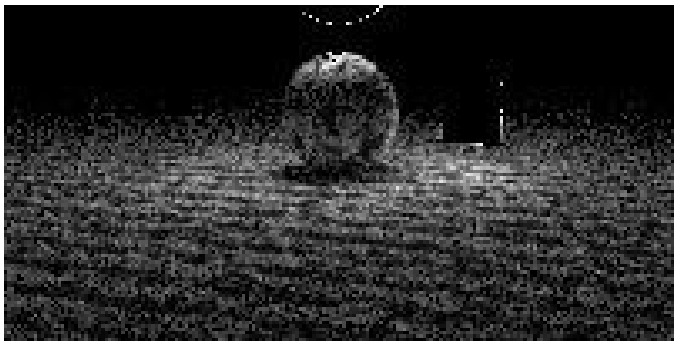


Chapter 5

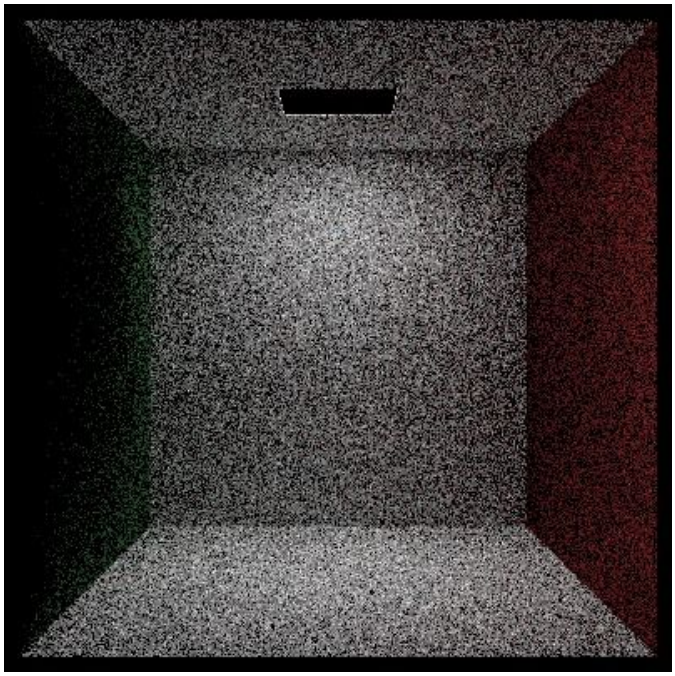
- Created a utility function that changes the values according to phi and theta
    - Phi = atan = a range from -Pi to Pi
    - Theta = asin(z) = a range from -Pi / 2 to Pi / 2
- Created a class for image textures to hold an image
- stb_header.h file and use of earth picture on Google images is accredited

Chapter 6

- Some screenshots from this point on have changed resolution for clarity from 200x200 to 400x400, although this affects render performance, the output produces a clearer image
- Created a class “diffuse_light” that emits light and adjusted accordingly to my colour function in main
- Created a class “xy_rect” to define a rectangle in the x-y plane along with its hit function to correctly orientate shapes in 3D space
    - X-z and y-z planes are also implemented



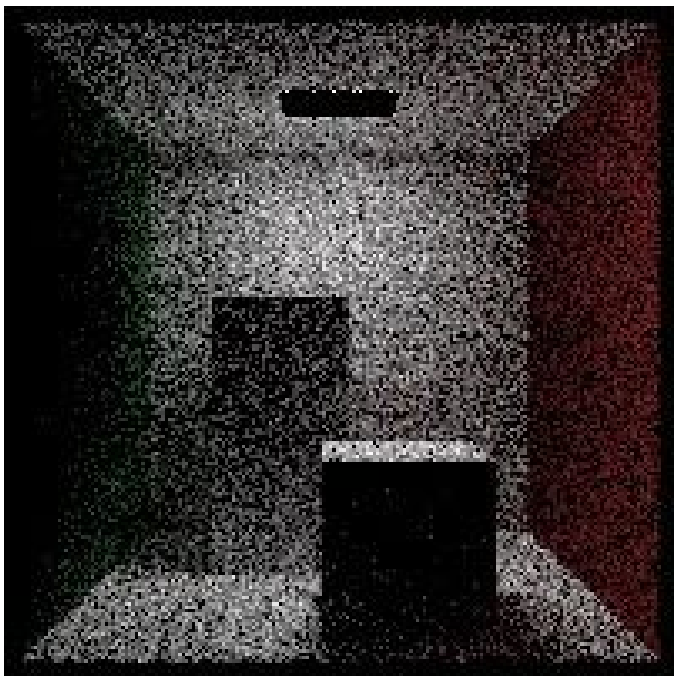
- Created cornell box scene and flipped normals to orientate the walls correctly, otherwise the walls will have a dark shade



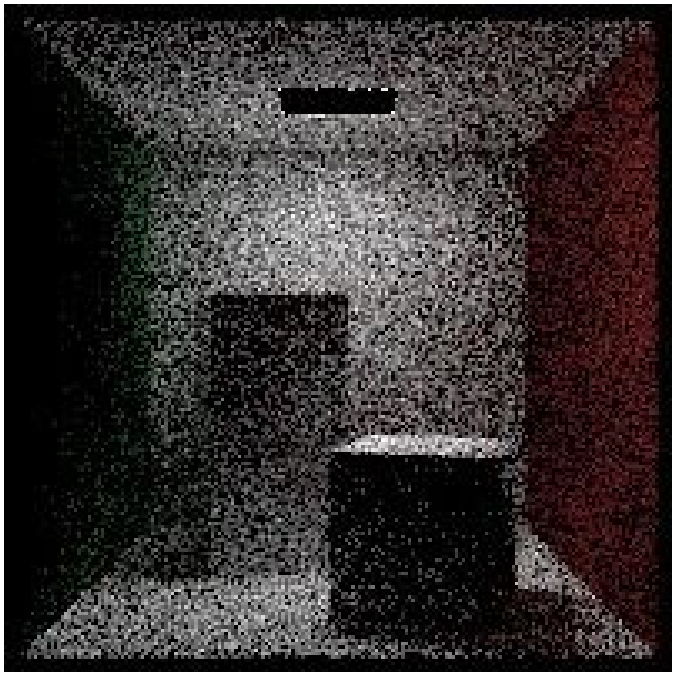
** Rendered at 400x400 resolution for clarity

Chp 7

- Added box class that holds an instance of rectangles

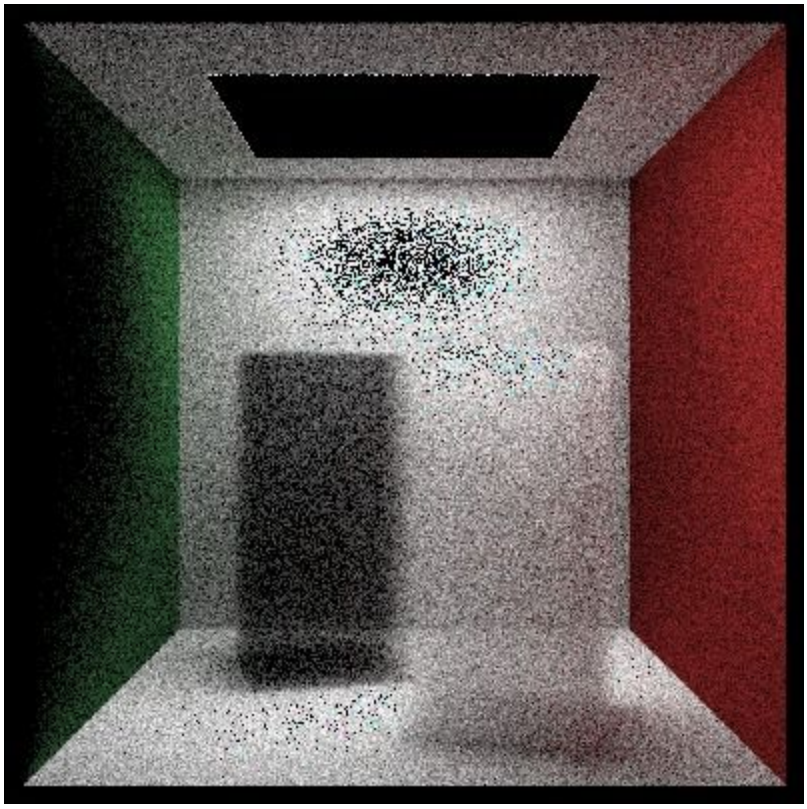


- Added translation and y-axis rotation to the boxes: rotate_y constructor and hit function to check if the ray hits the object



Chp 8

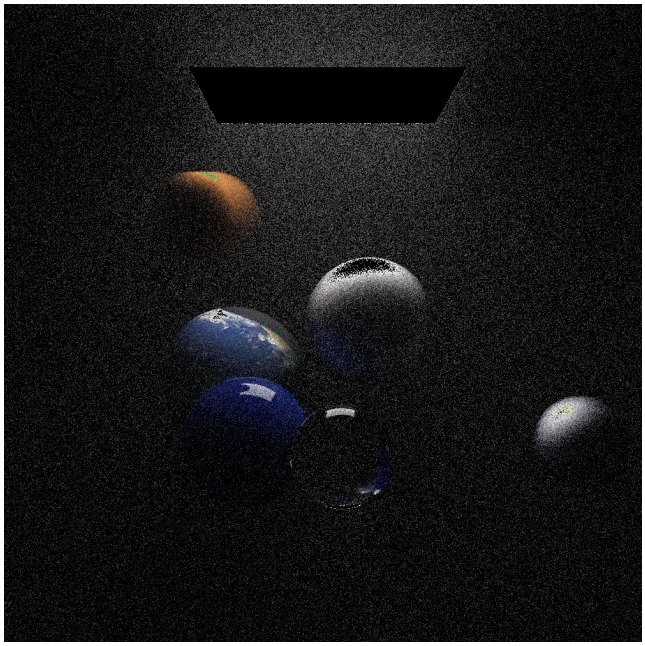
- Added a constant_medium class that utilizes the isotropic class under material.h with similar properties of hit and bounding_box functions
- The denser the object, the more likely it is to scatter: p = C * dL
    - dL = small distance
    - C = optical density of the volume



Final image

- Similarly to the random_scene(), I had run into problems where my program would crash due to my laptop's inability to generate so many rays. After narrowing down the problem, this line of code breaks the program:

```
List[l++] = new translate(new rotate_y(new bvh_node(boxlist2, ns,
0.0, 1.0), 15), vec3(-100, 270, 395));
```



** The 1000x1000 rendered image without the line of code above with variables nb = 1, ns = 2 in final(). Again, see the PPM file for a slightly more accurate image where the rectangular light is not black.