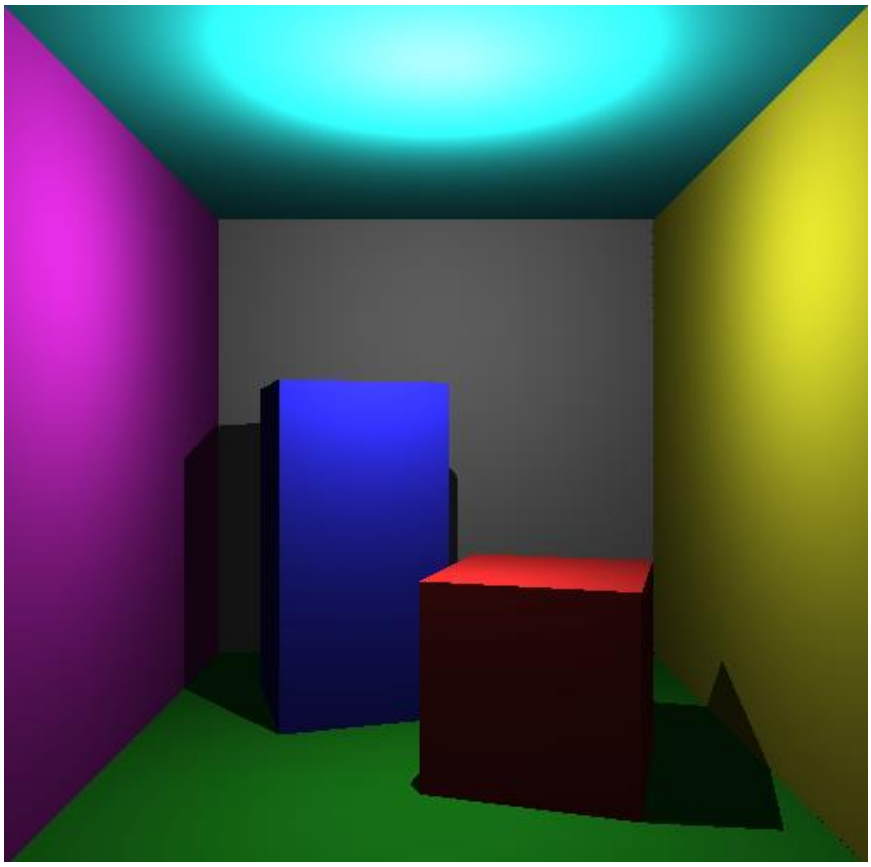
Computer Graphics Report - Ray Tracer

Introduction

With the beginning of this unit, we were given the task of producing a Ray Tracer, a rendering software. Needless to say, this was a rather enjoyable unit for both of us and the coursework was considered fun and interesting throughout.

Base CW

We completed all sections of the described part of the coursework, The result of which was an image that looked like below.



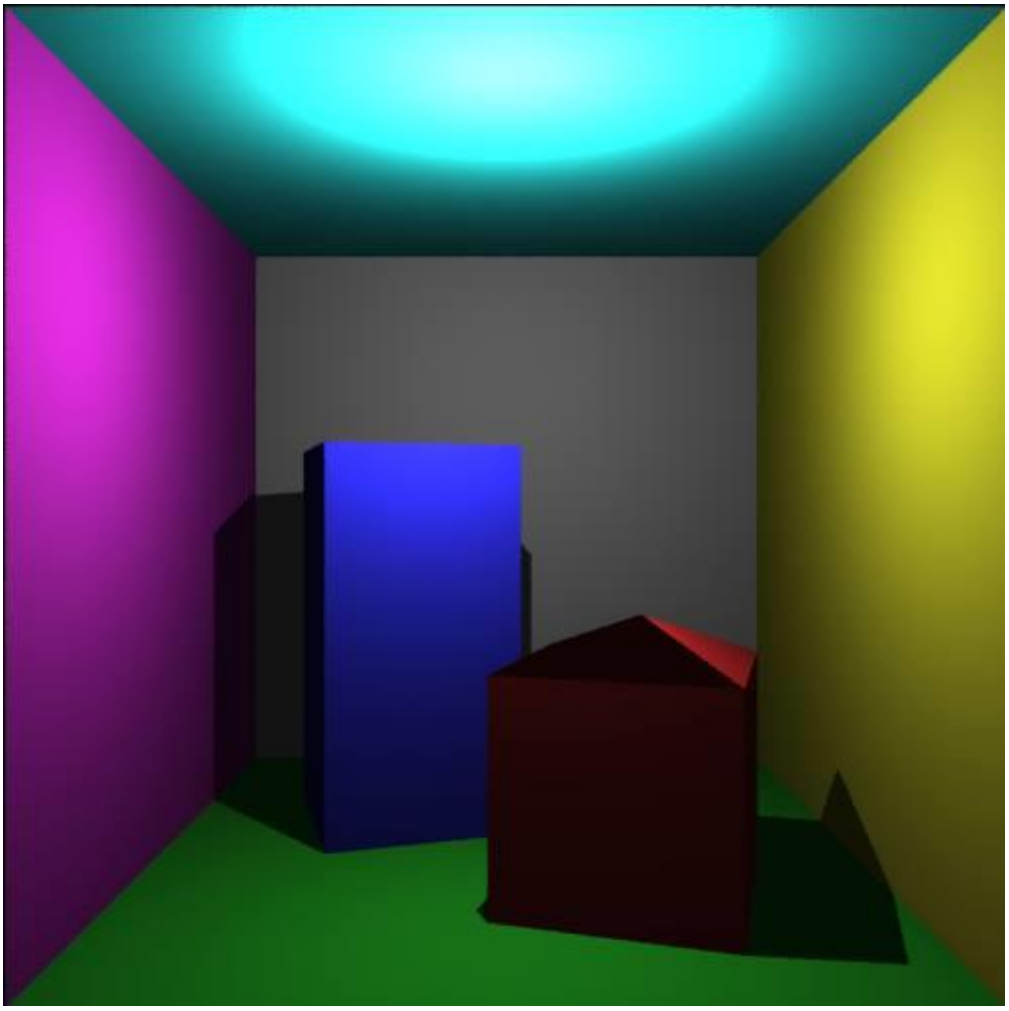
Parallelisation

In order to speed up the rendering we use OPENMP to take advantage of the fact that the ray tracer works in a very parallelizable way; each variable going into the FOR loop of the draw function does not change while in the loop. This lead to a massive improvement in render time, increasing the number of frames per second we could achieve.

Depth Perception

In order to add depth perception (and in the process, smooth edges) we modelled an aperture and defined an in focus distance at which point things would be in focus. This was done by shooting a ray from the camera through each pixel as before but instead of looking for intersections we found the point where the distance was the in-focus distance. We then sent a large number ( ) of rays from random points on the aperture in the direction of this focus point and averaged their colour to produce a final colour for the pixel, producing the “blurriness” of unfocused objects, which can be seen partially

in the image below. It is not easy to focus perfectly on an object, and that is the reason this image is not perfectly portraying our work.



Photon Mapping

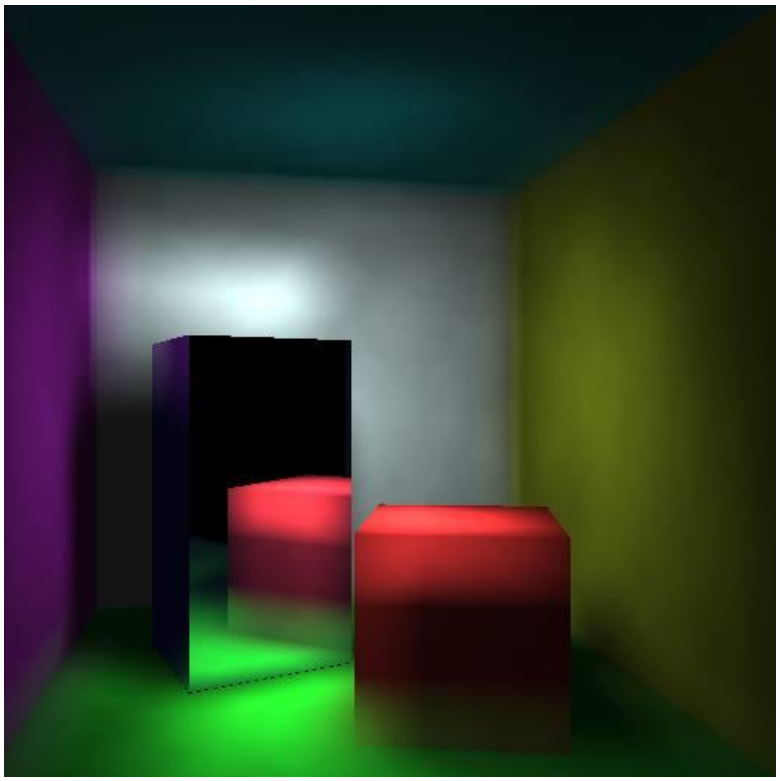
We next wanted to model a more realistic and natural looking lighting, therefore we decided to use photon mapping. Photon mapping entails the use of photons bouncing around the scene, causing light, as well as colour, being transfer across every triangle, depending on the triangle's reflectance characteristics. One can think of photon mapping to work as the ray tracer, shooting rays from the camera through each pixel on the image plane, trying to find intersections with triangles in the scene, but in this case shooting photons waiting for intersections with objects to produce an increase in light intensity on the surface the photons hit. Initially, we were shooting a total of 10000 photons, which were randomly directed, and updating their random position in each frame. This would produce a cool "foggy" effect, seen in the figure below, but nonetheless, not the desired final result.

Eventually, we concluded to shooting a number of photons on each frame, as before, but this time we stored the old photons, incrementally increasing the total number of photons, currently existing in the scene, each complying to the following conditions:

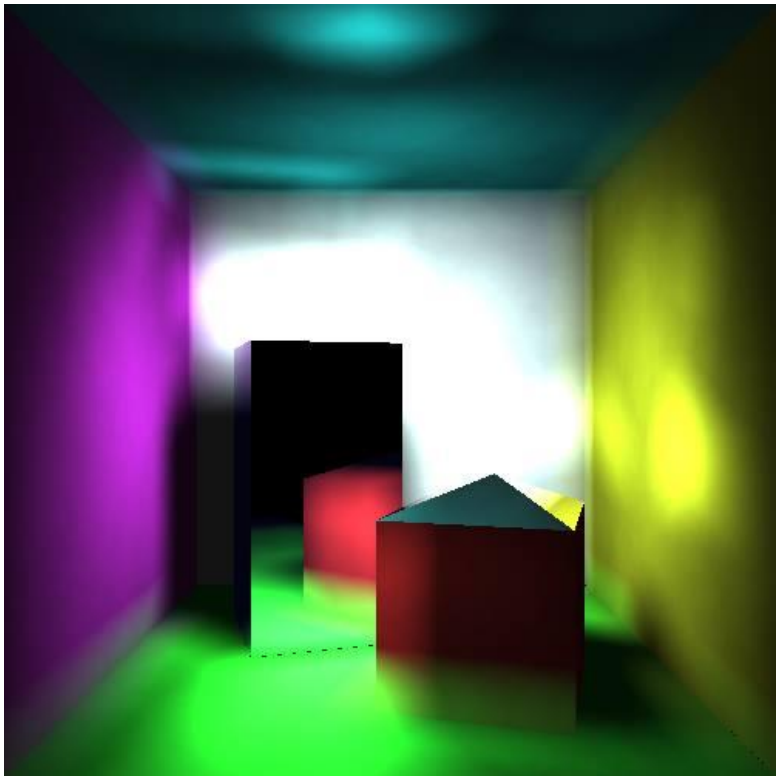
- If it hits a perfectly matt object then the photon is absorbed,
- Else, if it hits a perfectly reflecting surface the photon is reflected (and traced until it is absorbed),
- Otherwise the photon is either absorbed or reflected at a probability dependent on the reflectance value (0 is perfectly matt, 1 is a perfect mirror), named “decision” in the code provided.

During rendering, we then Trace rays from each pixel as before, if the surface is matt instead of just finding the direct light to the intersection point as before we find the colour/power of all the near photons and add them together to produce the colour value. The power of the photons is scaled dependent on the total number of photons as well as how far the photon is away from that particular point. If the surface is a mirror, we reflect that ray and instead use the photons that are present in at the new reflected intersection point. This technique is time costly but produces images such as the one below.

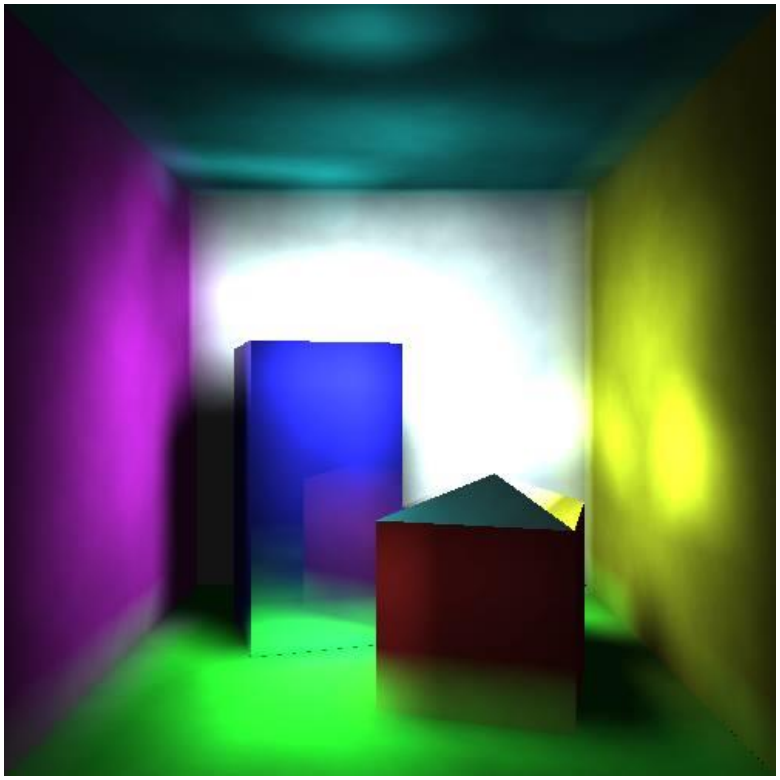
Next we present some configuration changes, when varying values such as the power intensity of the photons. Please refer to captions, placed below each image, stating the configurations used.



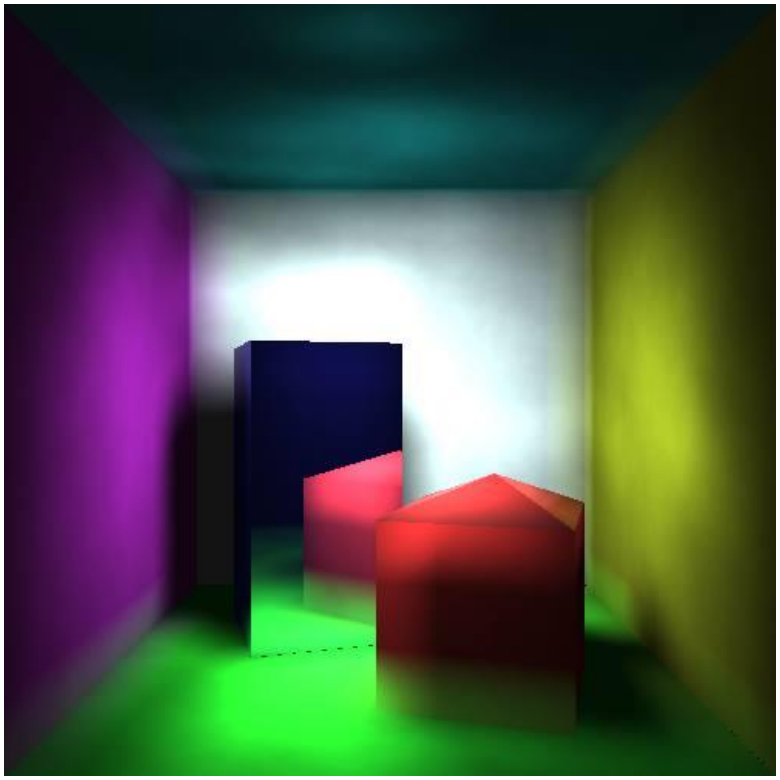
In this render, we let the ray tracer run for a while, as you can clearly see from the softness of the light hitting the wall. Additionally, we increased the distance in which the photons' colours would be averaged, thus the larger colour bleeding area on the red block. Increasing this distance entailes that more photons would be used to assign the colour of that area.



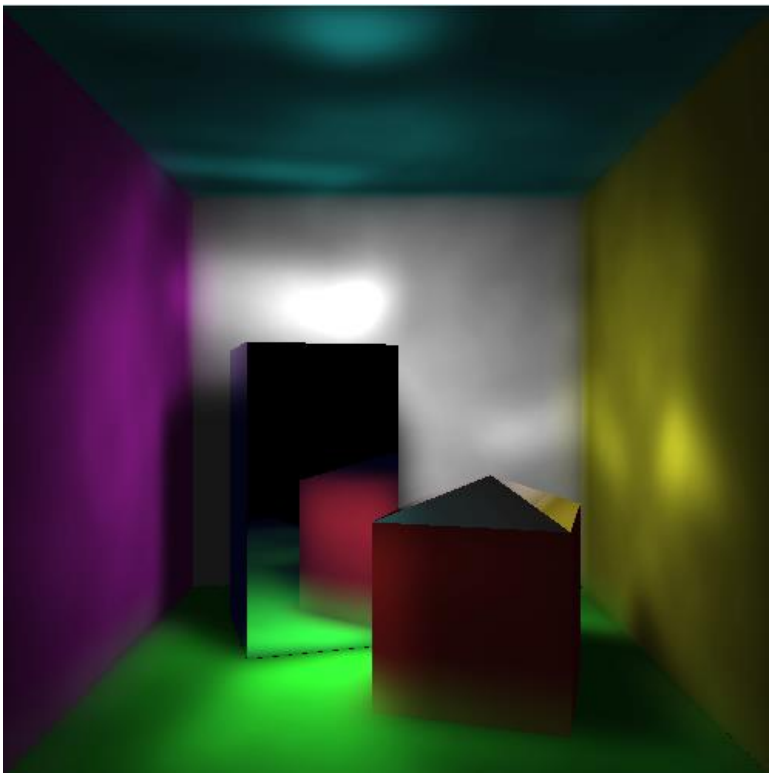
In this render, we increased the power intensity of photons, as well as added a pyramidal roof to the red block. As you can probably see, the scene is extremely bright, with clear light reflections off of the roof of the red block on to the ceiling and side walls.



In this render, we reduced the reflectiveness of the blue block to be 0.5 reflective. This entails that photons were bounced back approximately half of the time, producing this very unique blue glass-like texture which partly reflects light. Moreover, we reduced the power that photons emmited to get to a more natural looking scene.



In this render, we changed the reflectiveness of the blue block to be 0.8 reflective, while the red block's roof is 0.2 reflective. As you can see the blue colour is still inherited on the 0.8 reflective side of the blue block. Moreover, the red block's roof still shows that there exists a slight reflection getting colour bleeding from the white and yellow walls.



This render is by far one of the best we were able to produce. The colour bleeding is done more naturally, fading, instead of the straight line that was produced previously. This was achieved by using the distance of nearby photons. The closer the photon the greater is power and thus would contribute more on the colour being produced.

Controls

Here is a list of the buttons assigned to perform certain tasks when interacting with the scene:

- Camera Movement:
  - w – move forward;
  - s – move backward;
  - a – move left;
  - d – move right;
  - q – move up;
  - e – move down;
- Camera Rotation (arrow keys):
  - ↓ - rotate down;
  - ↑ - rotate up;
  - ← - rotate left;
  - → - rotate right;
- Reset Camera position and rotation:
  - r – reset position;
- Light Source Movement:
  - i – move light source in;
  - k – move light source out;
  - j – move light source left;
  - l – move light source right;
  - u – move light source up;
  - o – move light source down;
- Photon Mapping:
  - z – turn on photon mapping;
  - x – turn off photon mapping;
- Depth of Field:
  - c – turn on Depth of Field;
  - v – turn off Depth of Field;
  - b – decrease focal distance;
  - n – increase focal distance;

Conclusion

Working towards building a fully functional ray tracer was by far more enjoyable than building a rasterizer. The extensions were very interesting and we managed to gather enough knowledge on this field to keep us interested in computer graphics, given that the results are as beautiful as they seem in the images provided above.