# Report for Rasterisation Coursework COM30115

**Andreas Lazarou (97597), Durgadevi Rajendran (97481)**

## Rasteriser

To start with, we followed the coursework instructions and developed the code to project the vertices of all triangles onto the screen. Then, we coded the function that joins these vertices as per the specifications for a Cornell Box image, as shown in figure 1. We then extended the code to colour the image using a function that considered every triangle of the image as an independent shape and filled the required colours, as shown in figure 2; however, as it can be seen from the figure, there is no depth buffer in the image – this can be seen from the position of blue cuboid where it is overlapping with the red cube.

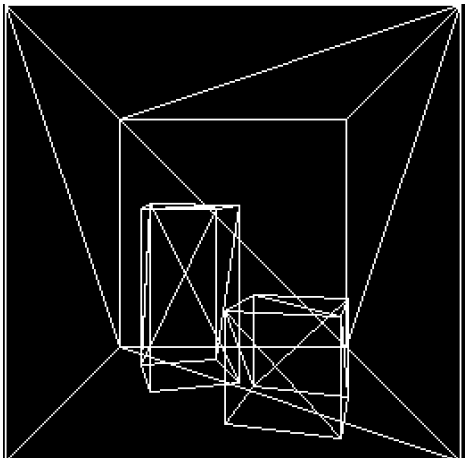

Figure 1

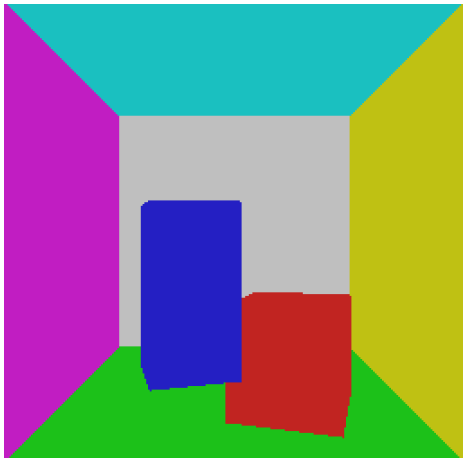

Figure 2

In order to sort this, we moved from considering the pixels instead of the vertices of the triangles. We calculated the depth of each pixel and then used the computed depth co-ordinates to populate the pixels row-wise, thus interpolating the depth over the triangle. The resulting outcome is then an image with the cuboid behind the cube, as seen in figure 3.

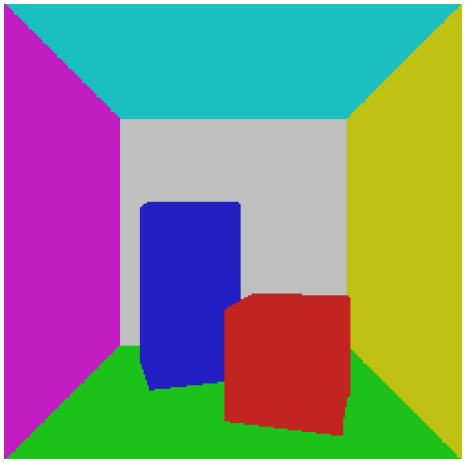

Figure 3

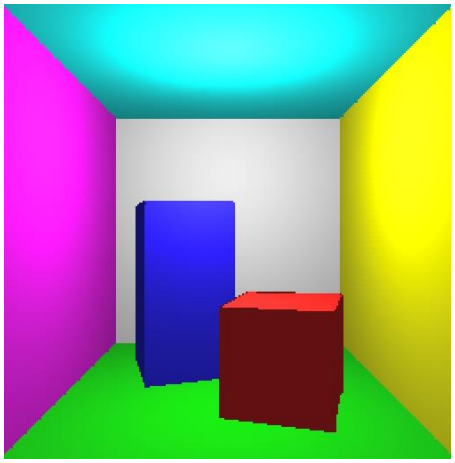

Figure 4

The image now reflected the objects in their respective positions and as expected, the rendering was faster than the Raytracer. We then developed the function to add lighting to the cornell box. First, we developed the function to illuminate every vertex of the image, which resulted in a partially illuminated

image. We then used the approach to get better lighting by illuminating each pixel in the image. The resulting image is as shown above in Figure 4.

**Extension 1: Texture Mapping**

As part of the first extension, we have tried to implement a texture mapping feature, in which we have changed the back and bottom walls with a bitmap image. This has been done by reading the pixel colours from each pixel of the image and replace the pixels of the triangles pertaining to the walls with the image pixels one by one. By this, the walls at back and bottom are replaced with the image, as shown in the figure below:

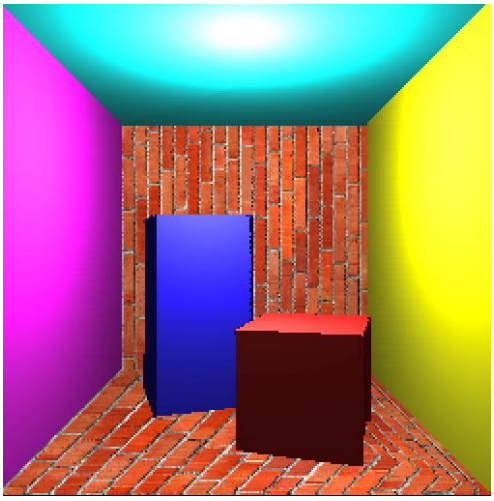

Figure 5

**Extension 2: Object Loading**

As part of the second extension, we have tried to implement the concept of object loading, in which we have loaded the universal Stanford Bunny object on to the cube in the Cornell box. We have also included the texture mapping feature on the bottom wall to give a different flavour to the image. This object loading has been implemented by reading an '.obj' file containing special coordinates for a bunny and drawing triangles of the bunny at the required location. We faced a minor challenging in positioning the bunny at the correct location (on the cube) whilst implementing this extension. This extension can be seen on the figure below:

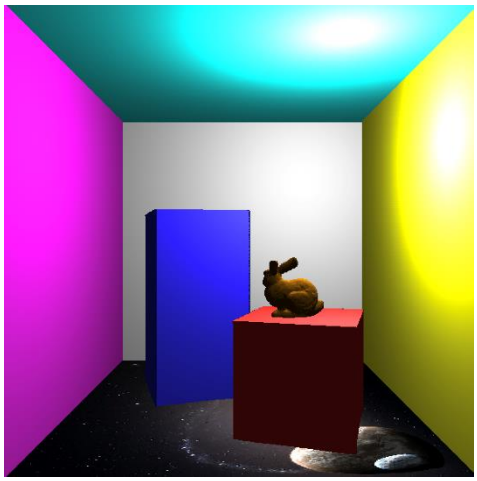

Figure 6