Francis Smedley - fs13832 Constantinos Theophilou – ct14872

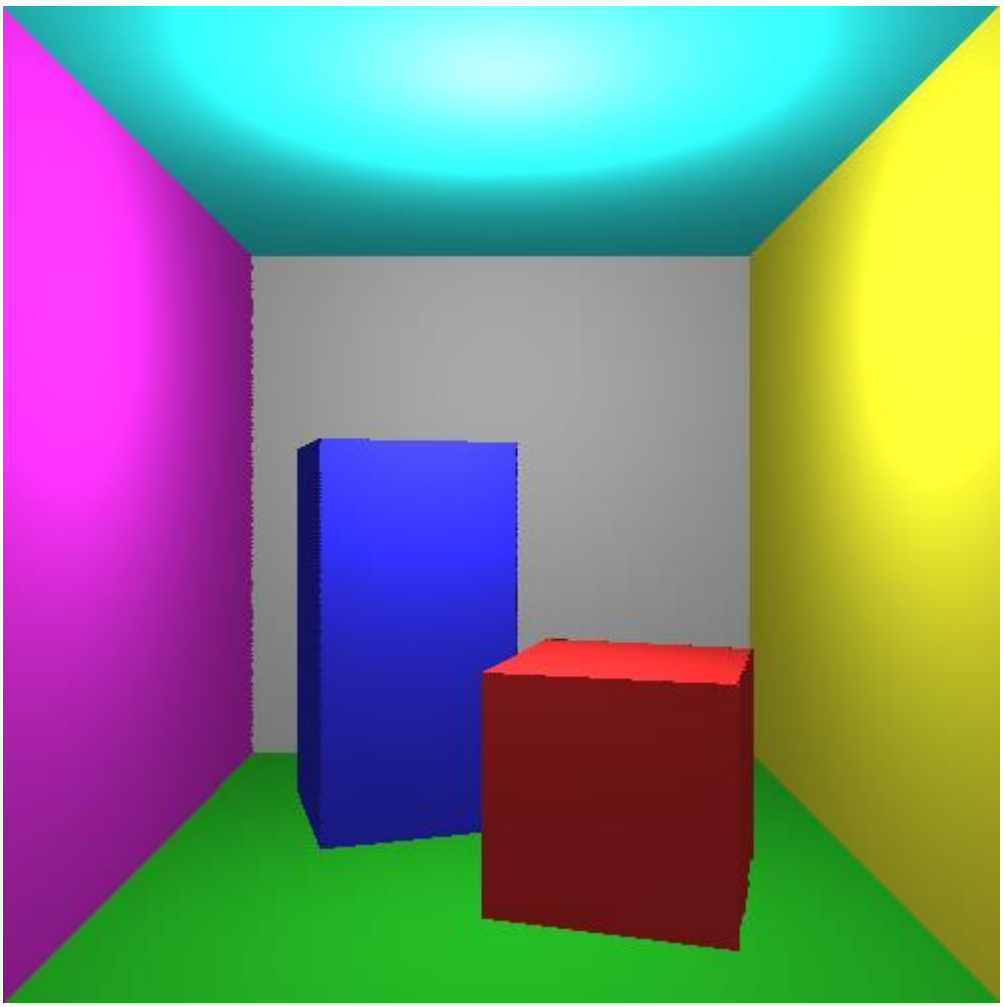
Computer Graphics Report - Rasterizer

Introduction

With the beginning of this unit, we were given the task of producing a Rasterizer, a rendering software. The rasterizer was the more cumbersome out of the two assignments, but yet again an unforgettable experience.

Base CW

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



Clipping

Our main focus for the rasterizer was to perform clipping of the scene whilst still maintaining the per pixel shading that we performed in the main body of the coursework.
This required us to clip each triangle and then recovering the world Z value of the new clipped point. Recovering the new world z value required a formula with two cases, we will discuss this in more detail later.

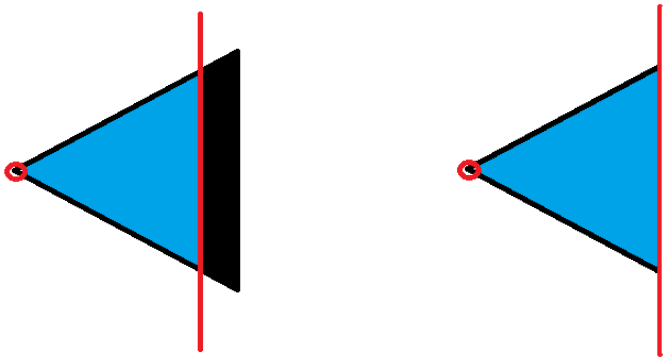
Handling all the possible cases

This was probably the most tiring and endless part of the clipping procedure. As one can imagine, a triangle can move in and out of a bounding box in multiple ways, for example one of the vertices can move out of the bounding box or two vertices can. This entails the need of a plethora of possible cases to ensure that when a triangle is out of the screen it should be clipped to avoid rendering unnecessary objects which are not visible.

Firstly, we assigned an out code for each vertex depending on which part of the screen it lies. If the out code of all vertices making up a triangle is 0, then we render that triangle, otherwise we move on to clipping that triangle.

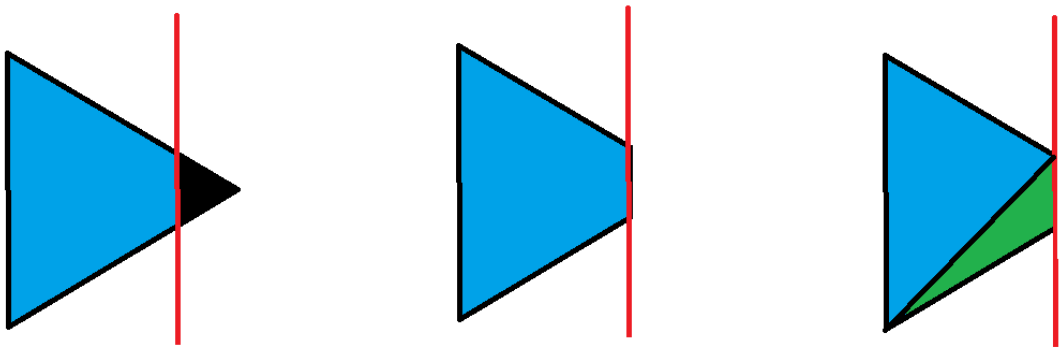
Then, is where we were required to break the task into smaller, easier to handle tasks, divide and conquer approach. There are two main cases:

1. If two vertices of a triangle are outside the screen then we easily find the new points where the screen cuts the edges between the vertices and assign the triangle to the new vertices found; easy clipping. In the illustration below, we show how this works.



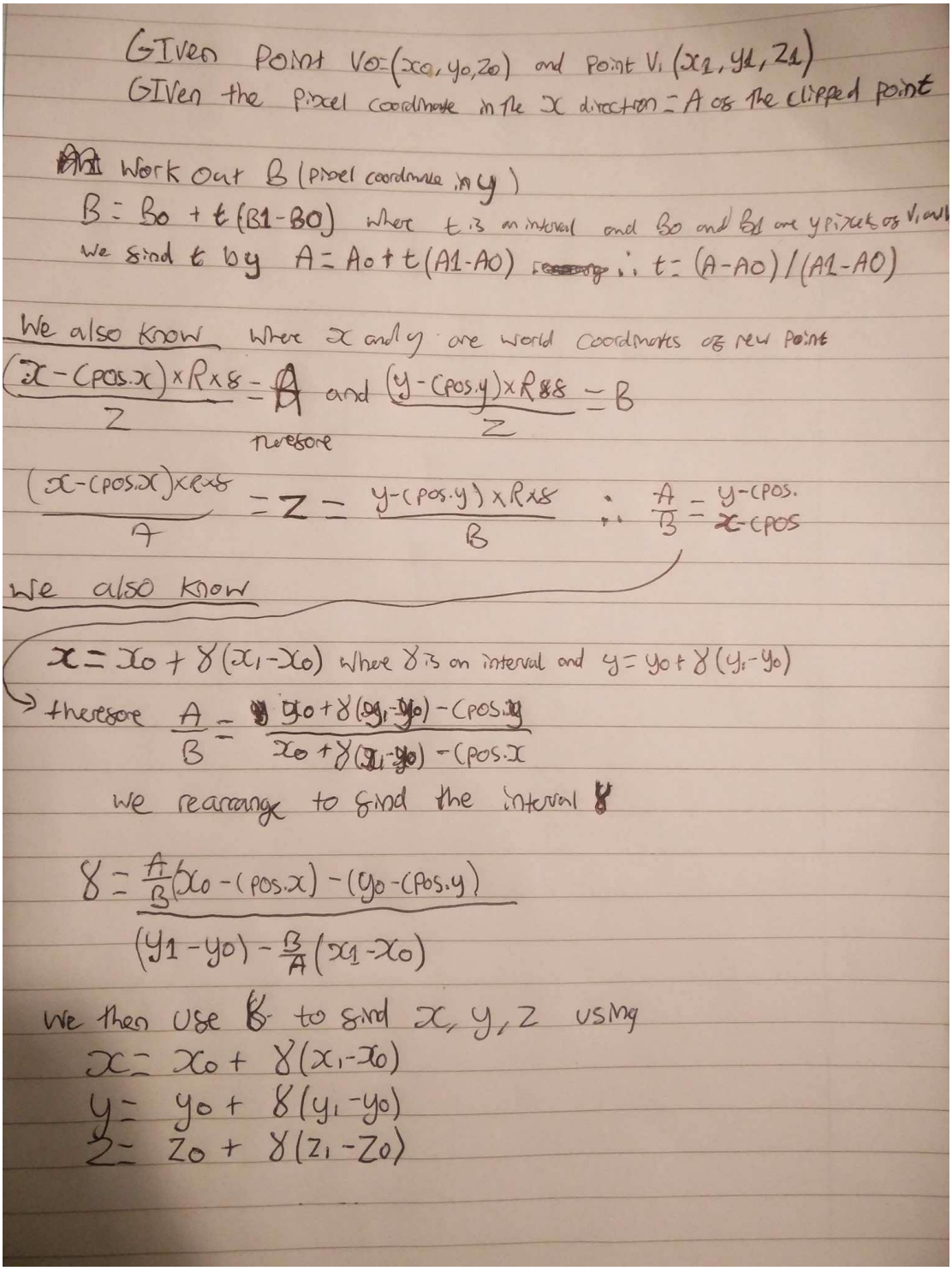
The red line is the screen, while the black part of the triangle is the part that should be clipped. The triangle changes from the whole (both blue and black parts) to the blue part.

2. If one of the triangle's vertices is out of the screen, then more computation must be done. When one of the vertices of a triangle is clipped, a four-sided shape is formed. Since the rasteriser renders triangles, this four-sided shape should be divided into two new triangles as can be seen in the illustration below. The two new triangles are the blue and green triangles

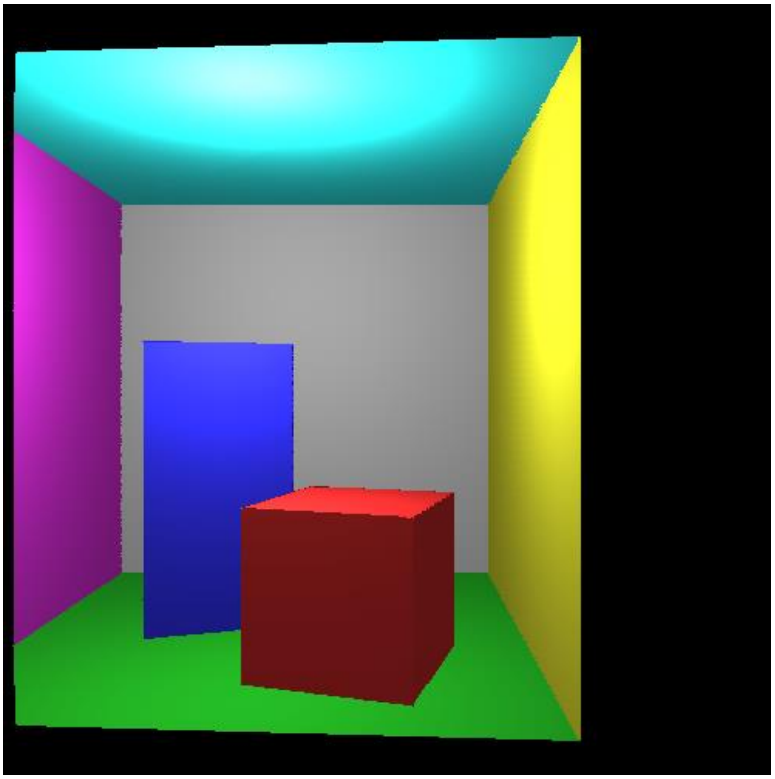


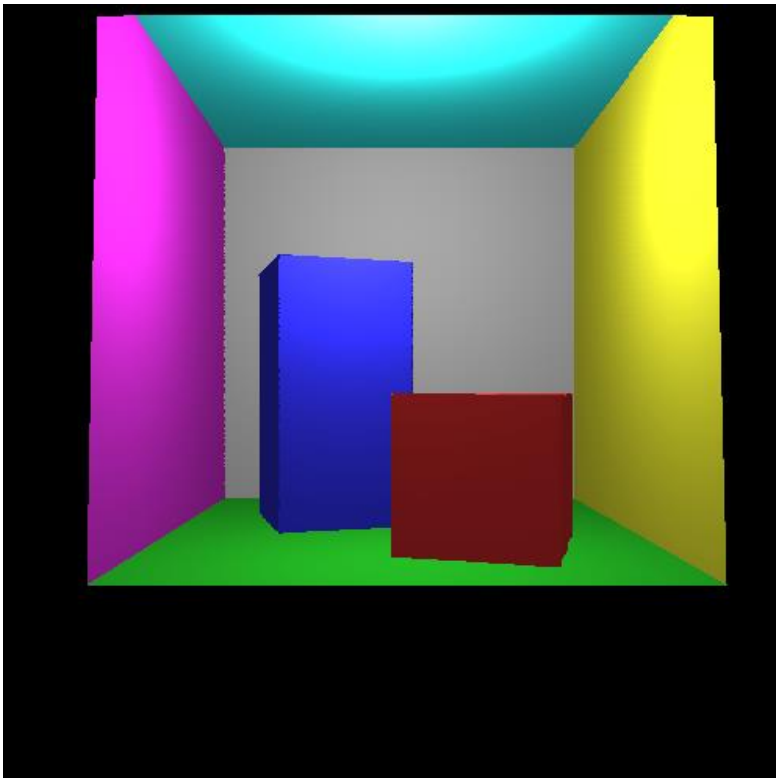
Once we have the new triangles we check that they don't need further clipping, if they do we add the new triangles to our triangle array and repeat the clipping process. This allows us to clip in several Axis. If they don't need clipping then we send them to our DrawPolygon function to be drawn.

While this may seem simple, the task was not as easy as it looks. It involved complicated calculations image coordinates to get the new clipped vertices correctly. We need the z value of each new vertex in order to shade the image correctly because we interpolate across the inverse depth when calculating the shading. Below is an image showing some of the calculations performed. We first find the screen location of the new clipped point (A,B) before finding the world location of the new point (x,y,z). We start by knowing the world coordinates of the two ends of the line V1 and V0 and either A or B depending on whether we are clipping in x or y.



Below are some images showing that clipping is performed. We increased the z-value of vertices so that it can be visible on the screen, otherwise it would not be obvious that clipping is done.





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 orientation:
  - r – reset position and rotation;
- 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;
- Switching to Clipping:
  - c – turn on clipping;
  - v – turn off clipping;
  - z – add gap to make clipping visible;
  - x – hide gad to make clipping invisible;

Conclusion

This assignment allowed us to work with a variety of techniques which will be very useful in the future. It was very enjoyable but still very complicated and tiring to produce the clipping. Nonetheless, we are glad we had this chance to show our programming skills for something related to gaming and graphics and we are very happy with the results.