Computer Graphics Coursework 1 Report

Brooklyn Mcswiney

Contents

1 Tasks

1.1 Drawing Pixels

Pixel Locations

Within the window the coordinates count from the top left of the window to the bottom right. This means that the pixel (0, 0) is in the top left corner, (w-1, 0) is in the top right corner and (0, h-1) is in the bottom left corner.

Figure 1: Particle field from the project implementation

1.2 Drawing Lines

Line Implementation

The line implementation works by starting with two points. The start and end of the line. The program will then take these values and calculate 2 numbers d_x and d_y. Then, the program will find which axis the line moves the most along (Known as the X or Y major). This is done through seeing which value between d_x and d_y is the greatest. Should d_x be greater we step along the X axis. The same is true should d_y be greater. The program then keeps track of this by saving a 'step' variable as the corresponding major.

The final step before moving onto deciding which pixels get drawn is to calculate the increase required each iteration in the x and y directions. This increase is found by dividing d_x and dy by the step variable that was saved earlier.

In order to draw the line the program now iterates over as many 'steps' needed to draw the line from the start to the end. During each iteration the program will check that the pixel is within the bounds of the surface and should this check return true the pixel will be drawn. This is done to avoid out of bounds errors within the program. The last stage of the loop is to move to the next pixel in the line by adding the x and y increase to our current pixel location.

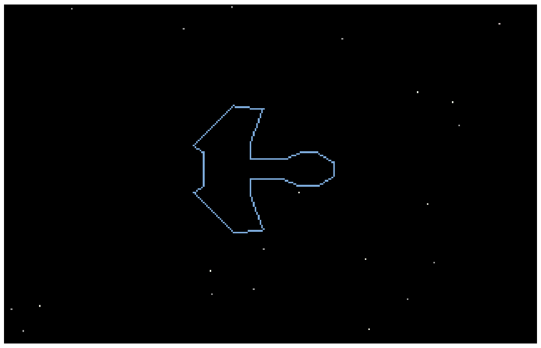


Figure 2: Ship drawn.

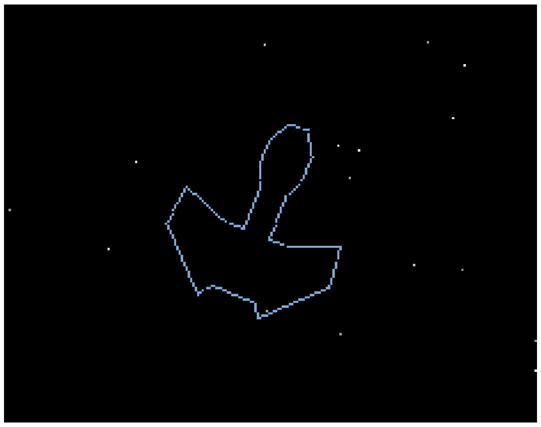
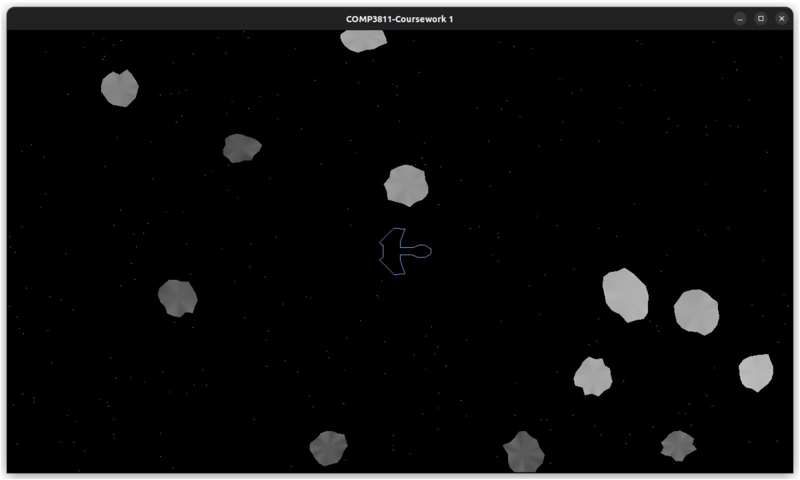


Figure 3: Ship rotated

1.3 Drawing Triangles

This algorithm begins by finding the possible bounds of the triangle by iterating through the points given to find the maximum and minimum coordinates. Once the bounds of the triangle are discovered the program will iterate through every pixel within the bounds. Every loop the program will check if the current pixel (X) is within the triangle by performing 3 half-plane tests. These tests are performed using the following algorithm $F(X) = n \cdot (X - P)$. Should $F(X) > 0$ be true for any of the lines the program continues onto the next pixel. Should the pixel pass all the half-plane tests the program will then check the pixel is within the window and if it is the pixel gets drawn.

1.4 Barycentric Interpolation



1.5 Blitting Images

1.6 Testing: Lines

1.7 Testing: Triangles

1.8 Benchmark: Blitting

1.9 Benchmark: Line drawing

1.10 Your own spaceship