Ryan Perkins

Professor Arabnia

CSCI 4810

13 September 2018

Line Drawing Summary

The basic line drawing and Bresenham’s algorithms are two different approaches to activating the correct pixels in order to successfully draw a line. When implementing these algorithms in Java, there were many challenges. Both algorithms at their base only handle a few cases which led to the need for extensive special case handlers.

When writing the basic line drawing algorithm, it only worked when ΔX > ΔY, or when the line did not have a steep slope. To handle this, the implementation checks to see which variable (x or y) increases at a faster rate. This will then be used to calculate the increment for the X and Y variables; as well as how long to run the loop. After the addition of this check, the implementation drew lines of all types.

For the Bresenham’s algorithm, it originally only covers lines drawn in octant zero and needed to work for all octaves. To deal with this, the implementation checks to see if ΔY > ΔX and then based on that, chooses the correct calculations for the randomized coordinates. The other octaves are then functional because of the different cases for each line case. Test data is shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # of Lines Drawn | Basic Algorithm  Time(ms) | | Bresenham’s  Time(ms) | |
|  | B/W | Rainbow | B/W | Rainbow |
| 10 | 26 | 22 | 31 | 24 |
| 100 | 79 | 85 | 79 | 92 |
| 1000 | 290 | 346 | 286 | 555 |
| 10000 | 1332 | 1432 | 1364 | 1477 |
| 100000 | 10737 | 11056 | 10798 | 11162 |

As shown in the data, there is not a large difference in time in milliseconds between the basic algorithm and Bresenham’s algorithm, even for the rainbows. However, although the algorithm does not affect the time, the number of lines drawn does.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # of Lines Drawn | Basic Algorithm  Time(ms) | | | | Bresenham’s  Time(ms) | | | |
|  | 1px | 10px | 100px | 1000px | 1px | 10px | 100px | 1000px |
| 1000 | 214 | 167 | 191 | 341 | 156 | 160 | 193 | 393 |
| 10000 | 809 | 841 | 887 | 2260 | 784 | 848 | 930 | 2226 |
| 100000 | 5470 | 5593 | 5773 | 18562 | 5524 | 5521 | 5639 | 17670 |

When changing the length of the lines, the data shows that it increased the time it took for the critical loop to execute significantly. The algorithm is able to draw lines of shorter lengths at the same speeds but as seen above, the time increase from 100px to 1000px is greater than the jump from 1px to 10px and 10px to 100px combined.