Chapter 2

Lines

See the website for details on how the programming assignment will be graded. Each lab specifies a complete list of filenames to be used in the lab; use these names so that the grader can find your work.

It is important that you complete each step before going on to the next, as the exercises build upon one another. You will find it helpful to diagram the action of each method or function as you go along. If you have difficulty in some step, DO NOT proceed before resolving it. You will not be able to fully appreciate the remaining content of the lab and you are likely to compound the problem.

2.1 Introduction

This lab will acquaint the student with Bresenham's integer-based line drawing algorithm and direct the creation of a visual program which will implement this algorithm.

2.2 Lines and Bresenham's Algorithm

Given are points $(x_a,\, y_a)$ and $(x_b,\, y_b)$; a line is to be drawn between them. The following are true for any point $(x,\, y)$ on the line.

$$\Delta x = x_b - x_a \tag{2.1}$$

$$\Delta y = y_b - y_a \tag{2.2}$$

$$m = \frac{\Delta y}{\Delta x} = \frac{y_b - y_a}{x_b - x_a} = \frac{y - y_a}{x - x_a} \tag{2.3}$$

$$\frac{y - y_a}{x - x_a} = \frac{\Delta y}{\Delta x} \tag{2.4}$$

$$y - y_a = \frac{\Delta y}{\Delta x}(x - x_a) \tag{2.5}$$

$$y = \frac{\Delta y}{\Delta x}(x - x_a) + y_a \tag{2.6}$$

The given points are assumed to be ordered so that the first lies to the left of the second, facilitating the use of a loop that increments from the first to the second; should this not be the case, the point coordinates may be swapped so that it is the case. The following inequality may then be assumed true for the line to be drawn.

$$x_a < x_b \tag{2.7}$$

If one were to then step from x_a to x_b one pixel at a time, calculating corresponding y values, it is apparent upon reflection that holes will be left in the line generated if y changes faster than x. Consequently, the following assumption is made to preclude this from situation (for now).

$$0 \ < \ \frac{\Delta y}{\Delta x} < 1 \tag{2.8}$$

Given this assumption, an increment of one in x will result in an increment of no more than one in y. For this case, the line drawing algorithm developed by Bresenham determines pixel centers for successive integral values of x from x_a to x_b. Because of the slope assumption, the pixels for x_{i-1} and x_i will be positioned in one of two ways: (1) the second will lie immediately to the right of the first, on the same horizontal row or (2) the second will be to the right and one row above, diagonally adjacent to the first; either way, there are no breaks in the line generated. The two pixel possibilities for x_i are labeled S_i and T_i corresponding to the row above and the current row, respectively. The true mathematical line passes through some value y^*. In order to discern which pixel to use for the line, an error measure e is associated with each; the error measure can be interpreted as the vertical distance along $x = x_i$ when the line passes between the two pixel centers.

$$\begin{aligned} S_i &= (x_i,\ y_{i-1}+1) && (2.9)\\ T_i &= (x_i,\ y_{i-1}) && (2.10)\\ y^* &= \frac{\Delta y}{\Delta x}(x_i - x_a) + y_a && (2.11)\\ e(S_i) &= (y_{i-1}+1) - y^* && (2.12)\\ e(T_i) &= y^* - y_{i-1} && (2.13) \end{aligned}$$

The smaller of the errors should indicate which point to use. To see that this is indeed the case, observe that the mathematical line will either pass above S_i, between S_i and T_i, or below T_i. When the line passes above S_i, clearly S_i should be selected; in this case, $e(S_i)$ is negative while $e(T_i)$ is positive, making $e(S_i)$ smaller. Conversely, when the line passes below T_i, T_i should be chosen; in this case, $e(T_i)$ is negative while $e(S_i)$ is positive, making $e(T_i)$ smaller. When the mathematical line passes between S_i and T_i, the error associated with both points is positive; the point T_i is chosen when the error associated with it is less than that for S_i. For all three cases, the point whose error is smaller is selected; consider the following manipulations of this inequality.

$$\begin{aligned} e(T_i) &< e(S_i) && (2.14)\\ e(T_i) - e(S_i) &< 0 && (2.15) \end{aligned}$$

Substitutions are made from Equations (2.12) and (2.13).

$$\begin{aligned} [y^* - y_{i-1}] - [(y_{i-1}+1) - y^*] &< 0 && (2.16)\\ y^* - y_{i-1} - y_{i-1} - 1 + y^* &< 0 && (2.17)\\ 2y^* - 2y_{i-1} - 1 &< 0 && (2.18) \end{aligned}$$

A substitution is then made from Equation (2.11).

$$\begin{aligned} 2\left[\frac{\Delta y}{\Delta x}(x_i - x_a) + y_a\right] - 2y_{i-1} - 1 &< 0 && (2.19)\\ 2\frac{\Delta y}{\Delta x}(x_i - x_a) + 2(y_a - y_{i-1}) - 1 &< 0 && (2.20)\\ 2\Delta y(x_i - x_a) + 2\Delta x(y_a - y_{i-1}) - \Delta x &< 0 && (2.21) \end{aligned}$$

Let the expression on the left hand side of the inequality be represented by e_i. The change in e from x_i to x_{i+1}, to be labeled Δe, may then be determined.

$$\begin{aligned}
e_i &= 2\Delta y(x_i - x_a) + 2\Delta x(y_a - y_{i-1}) - \Delta x && (2.22)\\
e_{i+1} &= 2\Delta y(x_{i+1} - x_a) + 2\Delta x(y_a - y_i) - \Delta x && (2.23)\\
&= 2\Delta y(x_i + 1 - x_a) + 2\Delta x(y_a - y_i) - \Delta x && (2.24)\\
\Delta e &= e_{i+1} - e_i && (2.25)\\
&= 2\Delta y(x_i + 1 - x_a) + 2\Delta x(y_a - y_i) - \Delta x \\
&\quad -[2\Delta y(x_i - x_a) + 2\Delta x(y_a - y_{i-1}) - \Delta x] && (2.26)\\
&= 2\Delta y(1) + 2\Delta x(-y_i + y_{i-1}) && (2.27)\\
&= 2(\Delta y - \Delta x(y_i - y_{i-1})) && (2.28)
\end{aligned}$$

In the event that e_i is negative, the point T_i is closer to the mathematical line and no increment in y takes place.

$$\begin{aligned}
x_i &= x_{i-1} + 1 && (2.29)\\
y_i &= y_{i-1} && (2.30)\\
\Delta e_{noInc} &= 2(\Delta y - \Delta x(y_{i-1} - y_{i-1})) && (2.31)\\
&= 2\Delta y && (2.32)
\end{aligned}$$

If e_i were positive, the point S_i is closer to the true line and an increment in y is necessary.

$$\begin{aligned}
x_i &= x_{i-1} + 1 && (2.33)\\
y_i &= y_{i-1} + 1 && (2.34)\\
\Delta e_{Inc} &= 2(\Delta y - \Delta x((y_{i-1} + 1) - y_{i-1})) && (2.35)\\
&= 2(\Delta y - \Delta x) && (2.36)
\end{aligned}$$

At the beginning of the line segment to be generated, the following are true.

$$\begin{aligned}
x_0 &= x_a && (2.37)\\
y_0 &= y_a && (2.38)\\
e_1 &= 2\Delta y(x_1 - x_a) + 2\Delta x(y_a - y_0) - \Delta x && (2.39)\\
&= 2\Delta y((x_a + 1) - x_a) + 2\Delta x(y_a - y_a) - \Delta x && (2.40)\\
&= 2\Delta y - \Delta x && (2.41)
\end{aligned}$$

Repetitive calculations will now generate the remainder of the points along the segment.

A similar algorithm may be derived for a line whose slope is greater than one, and further algorithms may be derived for negative slopes.

2.3 Example

Consider the line to be drawn between pixels (2, 3) and (9, 8).

$$\begin{aligned}
\Delta x &= 9 - 2 && (2.42)\\
&= 7 && (2.43)\\
\Delta y &= 8 - 3 && (2.44)\\
&= 5 && (2.45)\\
\Delta e_{noInc} &= 2\Delta y && (2.46)\\
&= 2[5] && (2.47)\\
&= 10 && (2.48)\\
\Delta e_{Inc} &= 2(\Delta y - \Delta x) && (2.49)\\
&= 2([5] - [7]) && (2.50)\\
&= -4 && (2.51)
\end{aligned}$$

The initial conditions are as follows.

$$\begin{aligned}
x_0 &= 2 && (2.52)\\
y_0 &= 3 && (2.53)\\
e_1 &= 2\Delta y - \Delta x && (2.54)\\
&= 2[5] - [7] && (2.55)\\
&= 3 && (2.56)
\end{aligned}$$

The mathematical line passes through the center of the pixel at (2, 3). Since e is initially positive, the mathematical line passes closer to the pixel at (3, 4) than that at (3, 3); the fact that y is incremented in turn dictates the next value of e (add -4 rather than add 10). See the second column of the Table 2.1. Subsequent columns are arrived at by applying the same logic.

i:	0	1	2	3	4	5	6	7	8
e_i:		3	-1	9	5	1	-3	7	4
x_i:	2	3	4	5	6	7	8	9	10
y_i:	3	4	4	5	6	7	7	8	9

Table 2.1: Example Bresenham Pixels

2.4 Assignments

2.4.1 Numeric Results

Write a console-oriented program which will begin by prompting the user for the name of a disk file, e. g. *lines.txt*. The first line of the file will contain an integer n indicating how many subsequent lines the file contains; each of the remaining n lines of the file will correspond to one mathematical line for which pixels are to be calculated. Following is an illustration of the file format:

```
3
x1a y1a x1b y1b
```

```
x2a y2a x2b y2b
x3a y3a x3b y3b
```

Your program should calculate and print the coordinates of the pixels to be drawn to represent each of the lines in the file in a format similar to that of Table 2.1; include the error calculation for each loop iteration.

2.4.2 Graphical Results

Write a program which will create a 500 x 500 frame window and then present the user with a dialog box requesting the name of a disk file; the individual lines of this file differ from those of the preceding program in that they also contain color information:

```
xa ya xb yb red green blue
```

Render each of the lines in this file in the frame window.

2.4.3 Program Submission Details

When the assignment has been completed and tested thorougly, submit one zip file containing all files associated with the assignment by using the course submission utility. (An evaluation copy of WinZip may be downloaded at *http://www.winzip.com/ddchomea.htm.*) Zipping the files will preserve the project and prevent corruption of the source code. (For example, Hotmail inserts carriage returns after about 80 characters of text.)

See the course website *Grading Criteria* page for details on how programming assignments will be graded.

If twenty or thirty minutes have been spent working on the same problem with no discernible progress, an impasse may have been reached for which assistance should be sought. In this case, contact the instructor or teaching assistant during office hours or by e-mail for assistance. When requesting assistance via e-mail, the following steps will reduce the time required to resolve the problem:

- Include the word "help" and the course number in the e-mail subject line.
- Describe the problem two ways: first, by describing the offensive behavior of the program and second, by identifying what appears to be the troubled area of program code.

The nature of the problem may be such that an office call will be more effective. For an office call, in addition to the program itself bring all associated paperwork detailing algorithm development and data structure design. For more on assistance, see the *Assistance* link on the course website.