ICS183: Bresenham's algorithm

These notes describe a classic line rasterization algorithm originally published by in 1965 in a paper by the title *Algorithm for Computer Control of a Digital Plotter* by Jack Bresenham of IBM. The algorithm takes two ordered pairs of integers, representing the endpoints of a line segment, and determines the collection of "pixels", also represented by 2D integer coordinates, that best approximates the mathematical line connecting the two end points. In general, only the two endpoints lie exactly on the line, so the algorithm must decide which pixels in between are "close enough" to be considered on the line. The beauty of Bresenham's algorithm is that it operates entirely in integer arithmetic, and is therefore well-suited to low-level graphics hardware. Bresenham's algorithm, and variations of it, are commonly implemented in today's graphics accelerator chips.

Implicit equation of a line:

To derive Bresenham's algorithm, we begin with the common *slope-intercept* formulation of a line, where the slope is simply the ratio of the change in y to the change in x. That is,

$$y \;=\; mx + b \;=\; \frac{\Delta y}{\Delta x}\, x + b. \tag{1}$$

The constant b is the vaue of y at which the line intercepts the Y-axis; thus, $(0, b)$ is a point on the line. Here, Δx and Δy might be obtained using two distinct points on the line, as we shall do below. Observe that equation (1) represents y as a function of x, and can express every line in the plane except for a vertical line, which essentially corresponds to an infinite slope. (We will handle vertical lines as a special case.) Alternatively, we can rearrange the equation so that it provides an *implicit* representation of the line. In particular, we may write

$$F(x, y) \;=\; \Delta y\, x - \Delta x\, y + \Delta x\, b \;=\; 0. \tag{2}$$

This function specifies the very same line, but implicitly; that is, any point (x, y) that satisfies $F(x, y) = 0$ is on the line, but F does not specify how to compute y given x, or vise versa.

In addition to specifying the line, the funtion F assumes a different sign at points that lie on different sides of the line. In particular, $F(x, y) < 0$ when (x, y) lies above the line, and $F(x, y) > 0$ when

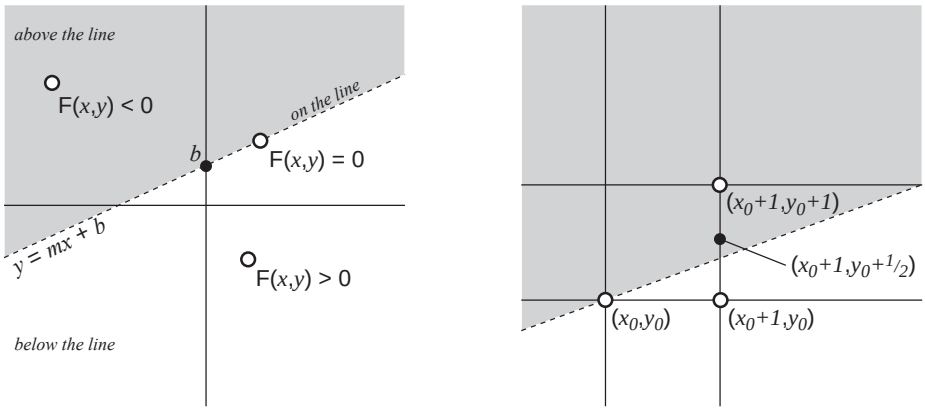

Figure 1: *(Left) The implicit function F that defines the line also partitions the plane into three regions; $F(x, y) < 0$ above the line, $F(x, y) = 0$ on the line, and $F(x, y) > 0$ below the line. (Right) Starting at point (x_0, y_0) the next point to be rasterized in drawing the line will be either $(x_0 + 1, y_0)$ or $(x_0 + 1, y_0 + 1)$. The choice depends upon whether $F(x_0 + 1, y_0 + \frac{1}{2})$ is positive or negative.*

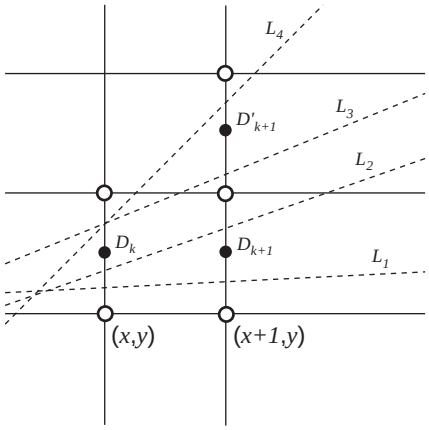

Figure 2: *As the rasterization process proceeds, we compute each decision variable incrementally from the prvious decision variable. The increment depends on the previous decision that was made. Both lines L_1 and L_2 above would result in one increment for moving from K_k to D_{k+1}, and lines L_3 and L_4 would result in a different increment in moving from D_k to D'_{k+1}.*

(x, y) lies below the line. See Figure 1. This is the property that Bresenham's algorithm exploits in determining which pixels best approximates a give line.

Let us now consider a line segment whose slope is between 0 and 1, running from "left" to "right"; that is, a segment whose end points (x_0, y_0) and (x_1, y_1) are such that $x_0 \leq x_1$, $y_0 \leq y_1$, and $y_1 - y_0 \leq x_1 - x_0$. Constraining the line as such allows us to make a few simplifying assumptions that make the derivation of Bresenham's algorithm very straightforward. Once we have handled this particular case, all the other lines will be easy to handle in a similar manner.

Now consider the figure on the right of Figure 1. This depicts the first pixel on the line segment, at (x_0, y_0), and the first "decision" that the algorithm must make; namely, should the next pixel be $(x_0 + 1, y_0)$, or $(x_0 + 1, y_0 + 1)$. These are the only two logical choices, given that the slope of the line is between 0 and 1. To make this decision, we simply consider the point that is midway between the two possibilities and determine whether this point is "above" or "below" the line; that is, we look at the sign of $F(x_0 + 1, y_0 + \frac{1}{2})$. We shall call the resulting value of F the *decision variable*, as it will be used to decide between the two possible pixels to select next. Note that the magnitue of the decision variable is irrelevant – only its sign is needed to make the decision. Plugging the coordinates of the intermediate point into F and simplifying, we have

$$\begin{aligned} F(x_0 + 1, y_0 + \frac{1}{2}) &= \Delta y\,(x_0 + 1) - \Delta x\,(y_0 + \frac{1}{2}) + \Delta x\,b && (3)\\ &= (\Delta y\,x_0 - \Delta x\,y_0 + \Delta x\,b) + (\Delta y - \frac{1}{2}\Delta x) && (4)\\ &= F(x_0, y_0) + (\Delta y - \frac{1}{2}\Delta x) && (5)\\ &= \Delta y - \frac{1}{2}\Delta x, && (6) \end{aligned}$$

where the final simplification is due to the fact that $F(x_0, y_0) = 0$, since the point (x_0, y_0) is on the line. Finally, since it is only the sign of this value that we care about, not its magnitude, we can

simplify it a bit by multiplying by 2. Thus, our first decision variable is

$$D \;=\; 2\,\Delta y - \Delta x. \tag{7}$$

The decision variable in equation (7) allows us to make the first decision – that is, should the second pixel have the same y value as the first, or is it 1 higher? But what about subsequent decisions? In all cases we could simply compute F at the intermediate point and look at its sign. However, there is a slightly more efficient method, namely *incremental computation.* Instead of computing the value of F anew at each decision point, we simply compute how it *differs* from the previous decision variable.

Figure 2 depicts three potential decision points and their corresponding decision variables, D_k, D_{k+1} and D'_{k+1}. Assuming that D_k has already been computed, which of D_{k+1} and D'_{k+1} should be computed next, and how do they differ from D_k? The differences are easy to compute, as follows:

$$F(x+2, y+\frac{3}{2}) - F(x+1, y+\frac{1}{2}) \;=\; \Delta y - \Delta x \tag{8}$$

$$F(x+2, y+\frac{1}{2}) - F(x+1, y+\frac{1}{2}) \;=\; \Delta y \tag{9}$$

To keep the imcrements consistent with our previous definition of D, we must again multiply them by two. We can now give the complete algorithm.

Rasterize the line from (x_0, y_0) to (x_1, y_1), where all coordinates are integers that satisfy $x_0 \le x_1$, $y_0 \le y_1$ and $y_1 - y_0 \le x_1 - x_0$. Thus, the line is assumed to run from left to right and have a slope between 0 and 1 (i.e. between 0 and 45 degrees). Lines of all other slopes can be handled using minor variations of this code.

```
function Bresenham( x0, y0, x1, y1 )
 1   Δx ← x1 − x0
 2   Δy ← y1 − y0
 3   D ← 2Δy − Δx
 4   x ← x0
 5   y ← y0
 6   SetPixel( x, y )
 7   while x < x1 do
 8         x ← x + 1
 9         if D < 0 then
10               D ← D + 2Δy
11         else
12               y ← y + 1
13               D ← D + 2(Δy − Δx)
14         endif
15         SetPixel( x, y )
16   endwhile
```