Computer Graphics

Lecture 8

Line Drawing Algorithms

DDA Algorithm: The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx using equations

$$\Delta y = m \, \Delta x$$

$$\Delta x = \Delta y / m$$

Note: These two equations we derived in the last lecture. Check lecture 7 notes for these two equations.

We sample the line at unit intervals in one coordinate and determine corresponding integer values nearest the line path for the other coordinate.

Consider first a line with positive slope, as shown in Fig. 8-1. If the slope is less than or equal to 1, we sample at unit x intervals ($\Delta x = 1$) and compute each successive y value as

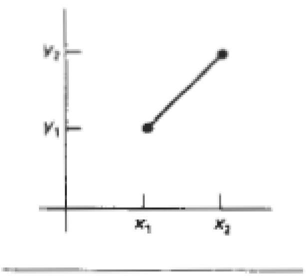

Fig 8-1

$$y_{k+1} = y_k + m \qquad (2.6)$$

Subscript k takes integer values starting from 1, for the first point, and increases by 1 until the final endpoint is reached. Since m can be any real number between 0 and 1, the calculated y values must be rounded to the nearest integer.

For lines with a positive slope greater than 1, we reverse the roles of x and y. That is, we sample at unit y intervals ($\Delta y = 1$) and calculate each succeeding x value as

$$x_{k+1} = x_k + (1/m) \tag{2.7}$$

Equations 2.6 and 2.7 are based on the assumption that lines are to be processed from the left endpoint to the right endpoint (Fig. 8-1). If this processing is reversed, so that the starting endpoint is at the right, then either we have $\Delta x = -1$ and

$$y_{k+1} = y_k - m \tag{2.8}$$

or (when the slope is greater than I) we have $\Delta y = -1$ with

$$x_{k+1} = x_k - (1/m) \tag{2.9}$$

Equations 2.6 through 2.9 can also be used to calculate pixel positions along a line with negative slope. If the absolute value of the slope is less than 1 and the start endpoint is at the left, we set $\Delta x = 1$ and calculate y values with Eq. 2.6

When the start endpoint is at the right (for the same slope), we set $\Delta x = -1$ and obtain y positions from Eq. 2.8. Similarly, when the absolute value of a negative slope is water than 1, we use $Ay = -1$ and Eq. 2.9 or we use $\Delta y = 1$ and Eq. 2.7.

The DDA algorithm is a faster method for calculating pixel positions than the direct use of Eq. 2.1(refer Lecture 7 notes). It eliminates the multiplication in Eq. 2.1 by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to step to pixel positions along the line path. The accumulation of round-off error in successive additions of the floating-point increment, however, can cause the calculated pixel positions to drift away from the true line path for long line segments. Furthermore, the rounding operations and floating-point arithmetic in procedure line DDA are still time-consuming. We can improve the performance of the DDA algorithm by separating the increments m and l / m into integer and fractional parts so that all calculations are reduced to integer operations. A method for calculating l / m increments in integer steps will be discussed in future lectures.