Computer Graphics

(Graphics Primitives – Line and Circle Drawing)

Lecture 002

Scan Conversion

- The major disadvantage of raster systems compared to vector systems arises from the discrete nature of pixel representation

- Primitives such as lines and polygons are specified in terms of their endpoints (vertices) and must be *scan-converted* into their component pixels in the frame buffer.

- Displaying graphics primitives on a raster scan system involves the task of choosing which pixel to turn on and to which intensity (we'll now go through some of these algorithms)

Line Drawing (A Straightforward Algorithm)

- $y = mx + c$ for endpoints (X_a, Y_a) and (X_b, Y_b)
- Calculating y value between $X_a <= X <= X_b$, where:
  - $m = dY / dX$
  - $c = Ya – Xa (m)$
- Note that for the case when $|m| = 1$, there will be exactly one pixel turned on in every column between Xa and Xb; and exactly one pixel turned on in every row between Ya and Yb.
- It is desirable that for $|m| < 1$, there is exactly one pixel in every column and at least one pixel in every row. Similarly for $|m| > 1$.
- However this is very inefficient since for each iteration of X or Y, we require a floating point multiply plus addition and rounding.

Line Drawing (The incremental DDA Algorithm)

- The previous algorithm can be improved upon if we note that the multiplication can be eliminated as follows:
  - $dY = (m(X+dX) + c) - (mX + c)$
    $dY = mX + m(dX) + c - mX - c$
    $dY = m(dX)$
- Hence, while looping through the x-coordinate ($dX = 1$) we can deduce the new vale of Y by adding m (dY) to the previous one, instead of re-calculating it.
- Similarly, dX can be calculated by adding 1/m to the previous value of X.

Line Drawing (The Midpoint Line Drawing Algorithm – Pitteway (1967))

- The drawback of the previous algorithm is that rounding y (or x) to and integer takes time. Also the variables for y and x need to be of type real since the slope (m) is a fraction.

- In practice we know that dY and dX can either by 0 or 1 (because of the rounding operation)

- Hence for 0 <= m <= 1, given that the previous pixel is (Xp,Yp), then the next pixel (Xp+1,Yp+1) is either (Xp +1,Yp) i.e. East or (Xp +1, Yp +1) i.e. NE

- E is plotted if the point M (midpoint of E and NE) is above the line, or else NE is plotted if M is below the line.

- Work out derivation

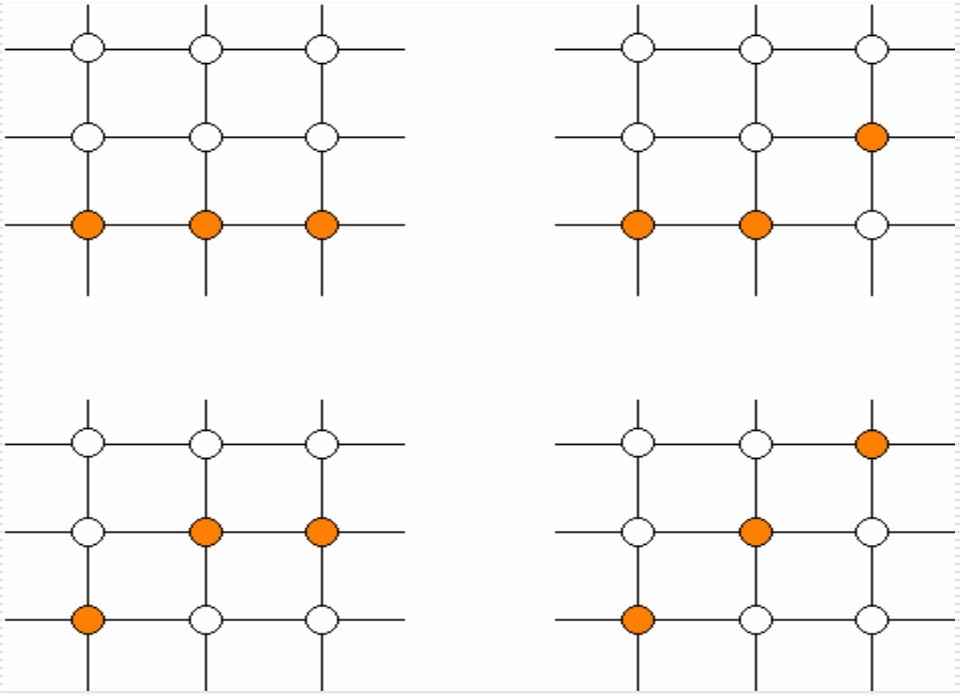
Line Drawing (The Bresenham Algorithm)

- Note that the value of C is always an integer
- The value of y is always an integer
- The value of C can be computed from the previous one by adding an integer value which does not depend on the x and y-coordinates of the current pixel.
- The Bresenham algorithm selects the pixel which is nearest to the line, hence one can see that both algorithms are effectively doing the same thing.

Line Drawing (The Double Step Midpoint Algorithm)

- An algorithm which attempts to draw two pixels at a time instead of one.

- For 0 <= m <= 1 the four possible patterns are the following:



Line Drawing (The Double Step Midpoint Algorithm)

- Patterns 1 and 4 cannot occur on the same line !!
- If the slope (m) is less than ½ , pattern 4 cannot occur.
- If the slope (m) is greater then ½ , pattern 1 cannot occur.
- Thus for any given line only three of these patterns can be used. Either 1,2 and 3 or 2,3 and 4.
- For 0 <= m <= ½ , only patterns 1,2 and 3 are used.
- Check out the values for C (no need for derivations)

Circle Drawing (The Straightforward Algorithm)

- Note the eight way symmetry of the circle. PlotPixel will actually draw eight pixels of the circumference.
- Algorithm uses either the Cartesian equations of a circle or it's polar form equivalent.
- The algorithm is very inefficient due to the use of floating point operations.
- Note that when using polar coordinates dA is set to 1/r and that this could result in points being plotted more than once
- A value for delta theta > (1/r) could be used to generate dotted circles.

Circle Drawing (The Midpoint Circle Drawing Algorithm)

- Given that the circle equation is given by $f(x) = r - x$ and that (Xp,Yp) is the previous pixel plotted, then we choose from the two candidates pixels E and SE by considering whether the boundary of the circle passes above or below the midpoint M.

- Check out the derivation for the values of C when moving to E and SE.

- Initially $C(0,r) = (1) + (r - \frac{1}{2}) - r$

  $= 1 + r - r + \frac{1}{4} - r$

  $= 5/4 - r$

  $= (1-r) + \frac{1}{4}$ (we can remove the $\frac{1}{4}$)

Circle Drawing (The Midpoint Circle Drawing Algorithm using second order differences)

- No need to go through derivation, however do note that by using second order differences we can get an optimised midpoint algorithm.

- Note that integer multiplication is utilised only once in order to calculate the initial value of incSE. From then onwards only integer addition is used.