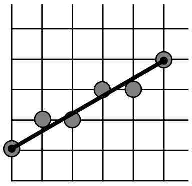
Rasterization

- Clip primitives to viewing window.
- Transform clipped primitives to device coordinates.
- Points: round floating point coordinates to nearest pixel coordinates.
- Lines: determine the coordinates of all pixels that "lie" on the line.



An Incremental Algorithm

- Input: endpoints (x_1, y_1) and (x_2, y_2)
- Compute $a = \dfrac{(y_2 - y_1)}{(x_2 - x_1)}$
- Line equation $y = ax + (y_1 - ax_1) = ax + b$
- for x := x1 to x2
  - y := Round(ax + b)
  - DrawPixel(x, y)

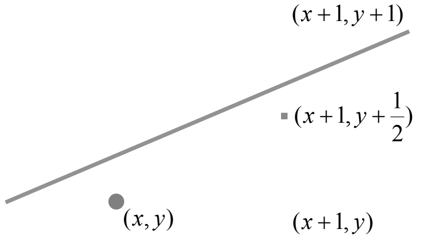
Incremental Algorithm

- Note: when x is incremented by 1, y is incremented by a

```
DrawPixel(x1, y1)
y := y1
for  x := x1+1  to  x2
    y  := y + a
    DrawPixel( x, Round(y) )
```

The Midpoint Algorithm

- Assumption: line has slope ‘a' between 0 and 1



- Idea: choose the next pixel by checking if the midpoint is above or below the line.

The Midpoint Algorithm

⌘Each line can be written using the line equation:

$$F(x, y) = Ax + By + C = 0$$

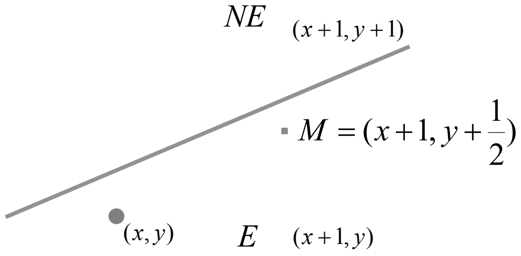
⌘The relation between any point (x,y) and the line can be determined by the sign of F(x,y):

- F(x,y) == 0 for points ON the line
- F(x,y) < 0 for points ABOVE the line
- F(x,y) > 0 for points BELOW the line

The Midpoint Algorithm

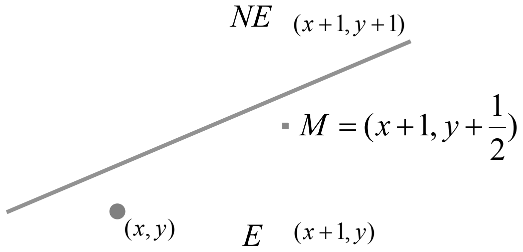
⌘Given a chosen pixel (x,y), the next pixel will be:

- (x+1,y) if F(M) <= 0 (denote this pixel by *E*)
- (x+1,y+1) if F(M) > 0 (denote this pixel by *NE*)



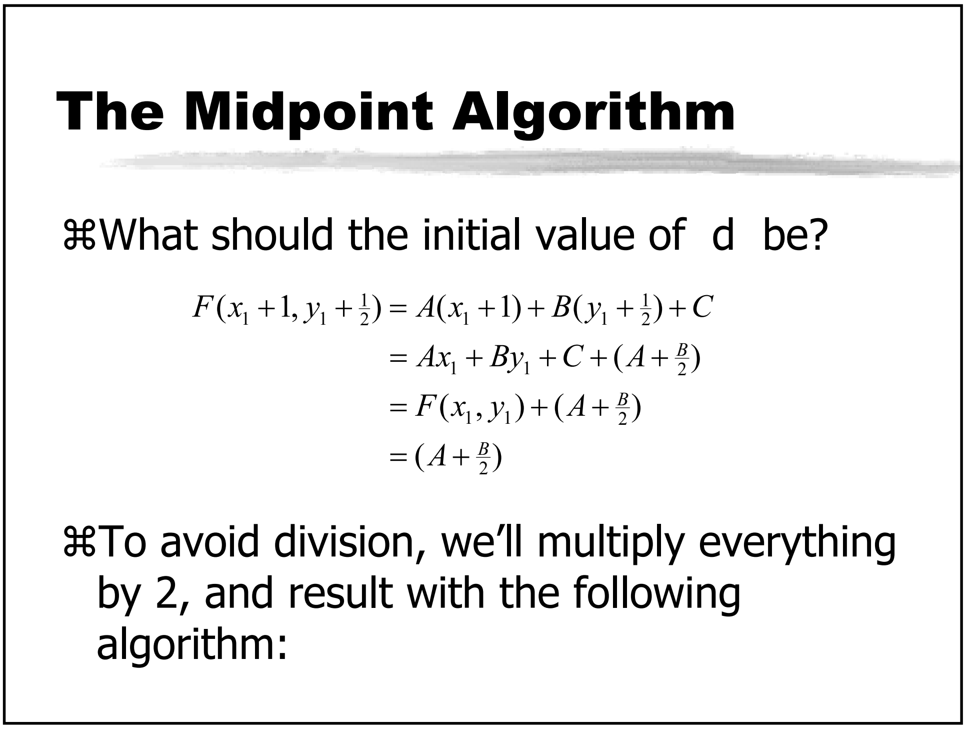
The Midpoint Algorithm

- For each pixel compute: $d = A(x+1) + B(y+\frac{1}{2}) + C$
- Make a decision based on sign of d
- Incrementally update M and d



The Midpoint Algorithm

- If d <= 0, we choose (x+1, y) (E)
  - M = M + (1,0) => d = d + A
- If d > 0, we choose (x+1, y+1) (NE)
  - M = M + (1,1) => d = d + (A + B)
- Each iteration we compute d by adding either A or (A+B), based on the sign of d

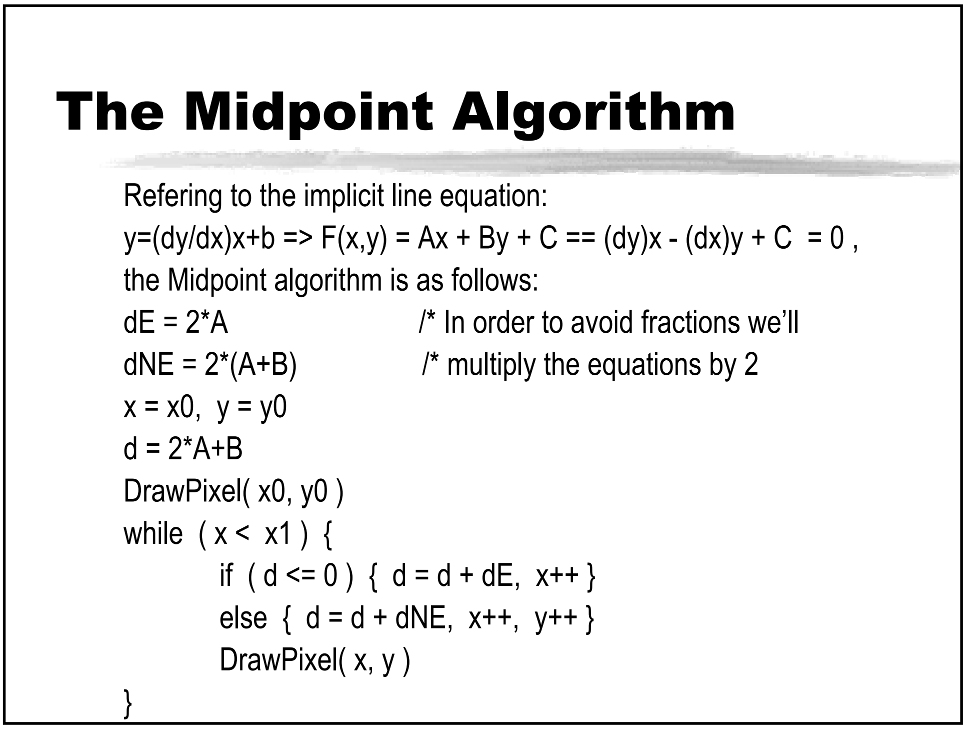


The Midpoint Algorithm

- What should the initial value of d be?

$$\begin{aligned} F(x_1+1, y_1+\tfrac{1}{2}) &= A(x_1+1)+B(y_1+\tfrac{1}{2})+C \\ &= Ax_1+By_1+C+(A+\tfrac{B}{2}) \\ &= F(x_1, y_1)+(A+\tfrac{B}{2}) \\ &= (A+\tfrac{B}{2}) \end{aligned}$$

- To avoid division, we’ll multiply everything by 2, and result with the following algorithm:



The Midpoint Algorithm

Refering to the implicit line equation:

y=(dy/dx)x+b => F(x,y) = Ax + By + C == (dy)x - (dx)y + C = 0 ,

the Midpoint algorithm is as follows:

```
dE = 2*A                    /* In order to avoid fractions we’ll
dNE = 2*(A+B)               /* multiply the equations by 2
x = x0,  y = y0
d = 2*A+B
DrawPixel( x0, y0 )
while  ( x <  x1 ) {
        if  ( d <= 0 )  {  d = d + dE,  x++ }
        else  {  d = d + dNE,  x++,  y++ }
        DrawPixel( x, y )
}
```