# LINE DRAWING

Description:

Given the specification for a straight line, find the collection of addressable pixels which most closely approximates this line.

Goals: (not all of them are achievable with the discrete space of a raster device)

- Straight lines should appear straight.
- Lines should start and end accurately, matching endpoints with connecting lines.
- Lines should have constant brightness.
- Lines should be drawn as rapidly as possible.

Problem:

- How do we determine which pixels to illuminate to satisfy the above goals?
- Vertical, horizontal, and lines with slope = +/- 1 easy.
- Others create problems - staircasing/ jaggies - aliasing

## Direct Solution:

Solve y=mx+b where (0,b) is the y-intercept and m is the slope.

Go from x0 to x1 calculate round(y) from the equation.

Take an example, b=1 (starting point (0,1)) and m = 3/5.

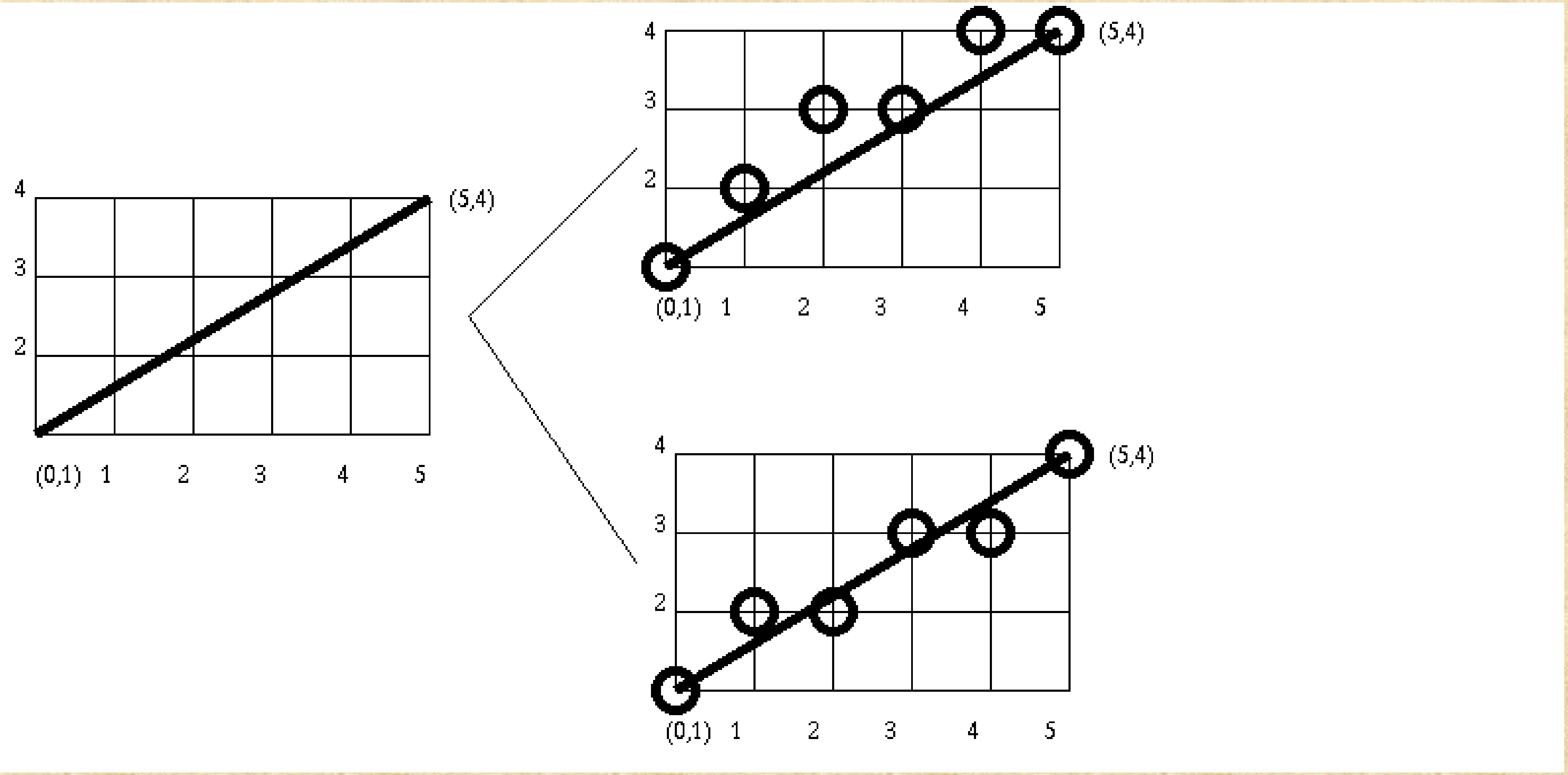
Then x=1, y= 2
x=2, y= 2
x=3, y= 3
x=4, y= 4
x=5, y= 4.0. For results, see next slide.

Why is this undesired?

- `*´ and `/´ are expensive
- Round() function needed
- Can get gaps in the line (if slope > 1)

Take another example:
y=10x+2
x=1, y=12
x=2, y=22

## Using next highest



## Using Round

## DDA - Digital Difference Analyzer

**Incremental Algorithm.**

Based on $y = (y1-y0)/(x1-x0)\, x + b$

**Assume $x1 > x0$ and $|dx| > |dy|$**

**(can be easily modified for the other cases.)**

**The Algorithm:**

```
dx = x1-x0
dy = y1 -y0
m = dy/dx
y=y0
for (x=x0 to x1)
    draw_point (x, round(y))
    y=y+m
end for
```

**Problems:**

**Still uses floating point and round() inside the loop.**

**How can we get rid of these?**

## MIDPOINT LINE ALGORITHM

Incremental Algorithm
Given the choice of the current pixel, which one do we choose next
(Assume first octant)
E or NE?

Equations:

1. $y = (dy/dx)\ x + B$

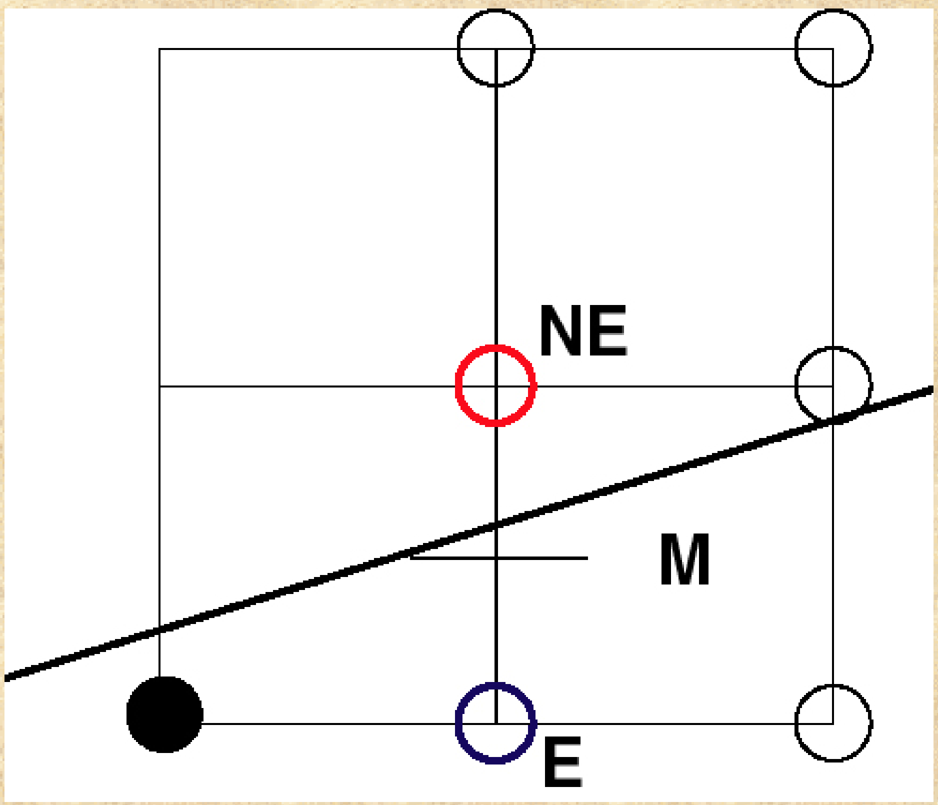
2. $F(x,y) = ax + by + c = 0$

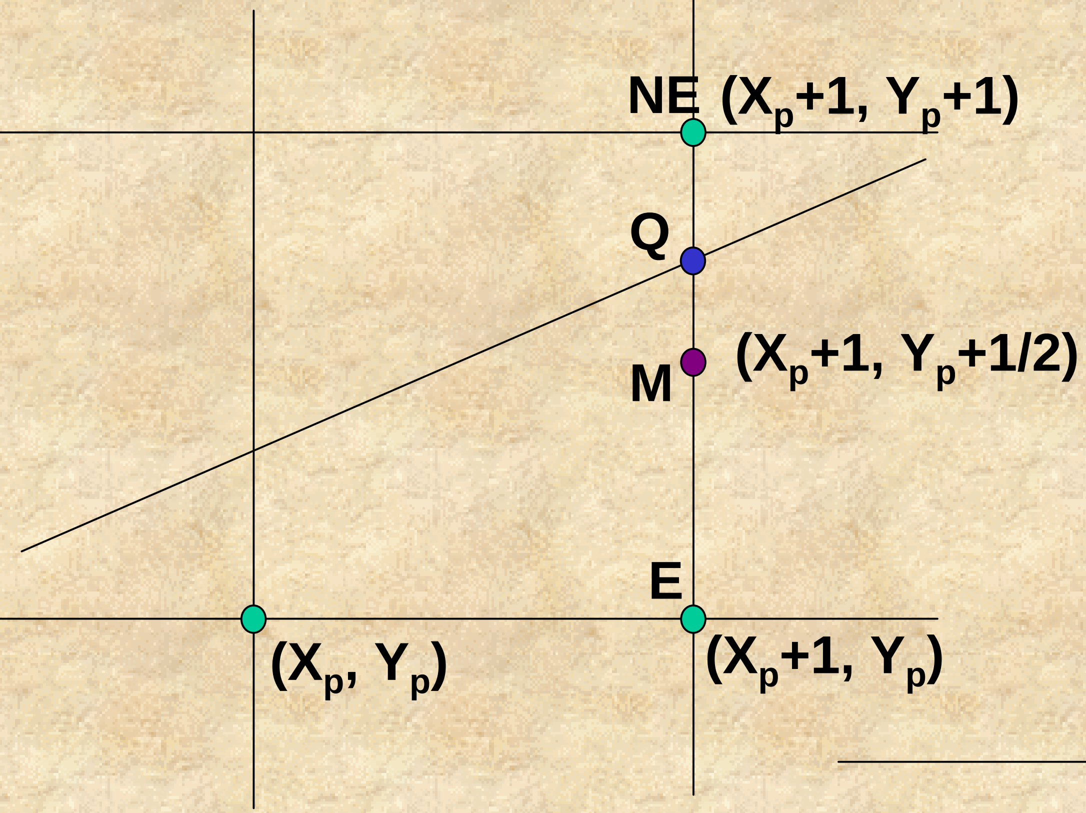
Gives: $F(x,y) = dy*x - dx *y + B*dx = 0$

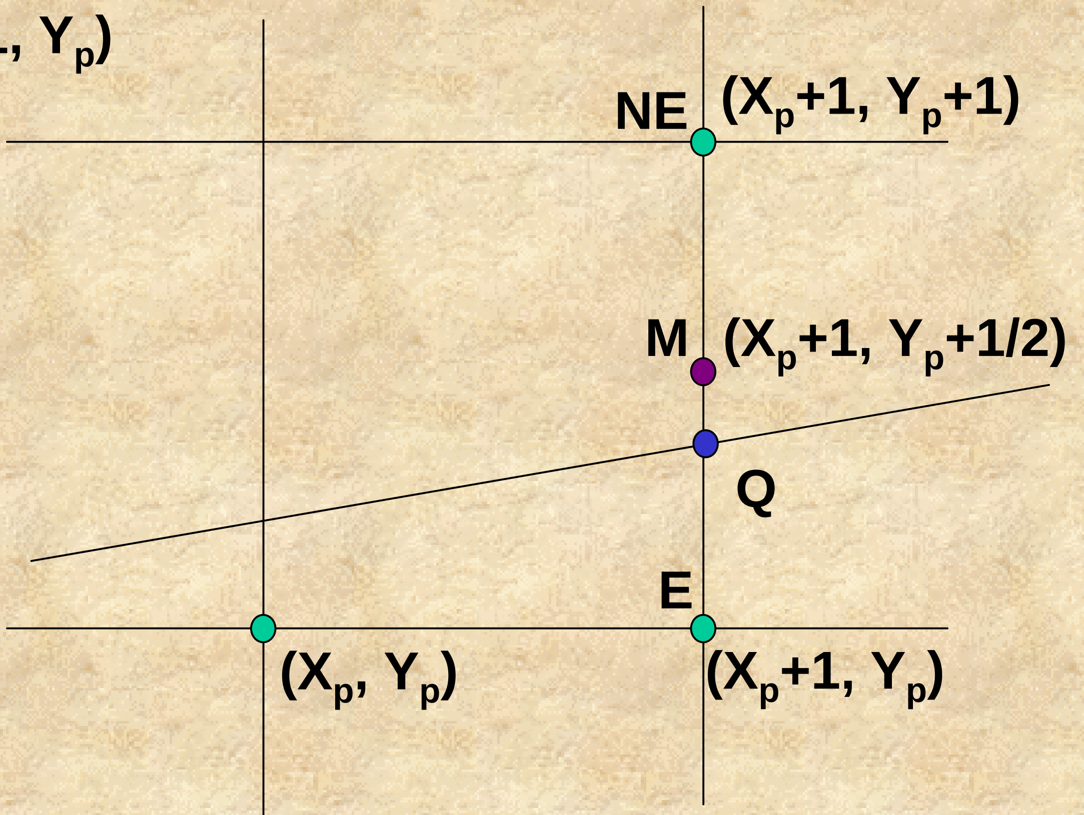
$\Longrightarrow a = dy,\ b = -dx,\ c = B*dx$

$F(x,y) > 0$ if point below the line

$F(x,y) < 0$ if point above the line







```
If F(M) > 0      /*Q is above M */
        then Select NE
        /*M is below the line*/

        else  Select E
        /* also with F(M) = 0  */
```

Evaluate mid-point M using a decision variable d = F(X,Y):

$d = F(X_p+1, Y_p+1/2) = a(X_p+1)+b(Y_p+1/2)+c$; at M, Set $d_{old} = d$

Based on the sign of d, you choose E or NE.

Case I. Chosen E:

$d_{new} = F(X_p+2, Y_p+1/2) = a(X_p+2)+b(Y_p+1/2)+c$

$(\Delta d)_E = d_{new} - d_{old} = a$ /* = *dy* */

Case II. Chosen NE:

$d_{new} = F(X_p+2, Y_p+3/2) = a(X_p+2)+b(Y_p+3/2)+c$

$(\Delta d)_{NE} = d_{new} - d_{old} = a + b$ /* = *dy – dx* */

Update using $d_{new} = d_{old} + \Delta d$

## Midpoint criteria:

$d = F(M) = F(X_p+1, Y_p+1/2);$

if $d > 0$ choose NE

$d <= 0$ choose E

**Case EAST :**

increment M by 1 in x

$d_{new} = F(M_{new}) = F(X_p+2, Y+1/2)$

$(\Delta d)_E = d_{new} - d_{old} = a = dy$

$(\Delta d)_E = dy$

**Case NORTHEAST:**

increment M by 1 in both x and y

$d_{new} = F(M_{new}) = F(X_p+2, Y_p+1\ 1/2)$

$(\Delta d)_{NE} = d_{new} - d_{old} = a + b = dy - dx$

$(\Delta d)_{NE} = dy - dx$

**What is $d_{start}$?**

$d_{start} = F(x_0+1, y_0+ 1/2)$

$= ax_0 + a + by_0 + b/2 + c \quad = F(x_0, y_0) + a + b/2 = dy - dx/2$

**Let's get rid of the fraction and see what we end up with for all the variables:**

$d_{start} = 2dy - dx$

$(\Delta d)_E = 2dy$

$(\Delta d)_{NE} = 2(dy - dx)$

## The Midpoint Line Algorithm

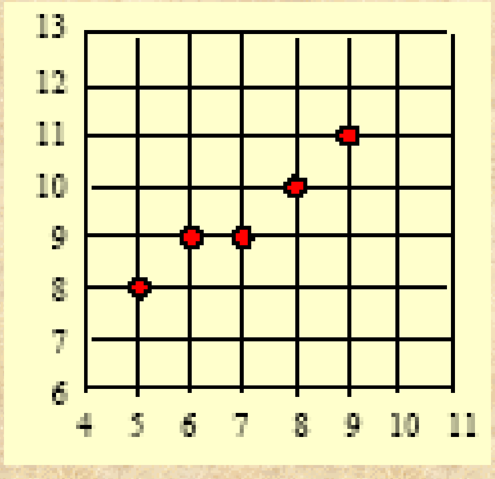
```
x       = x0;              y       = y0;
dy      = y1-y0 ;          dx      = x1 - x0;

d       = 2dy – dx;
$(\Delta d)_E$  = 2dy;
$(\Delta d)_{NE}$ = 2(dy - dx);

PlotPoint(x,y)

while (x <= x1)
        if d <=0    /* Choose E */
                 d = d + $(\Delta d)_E$  ;

        else       /* Choose NE */
                 d = d +  $(\Delta d)_{NE}$ ;
                  y = y+1
        endif

        x = x+1
         PlotPoint(x,y)
end while
```

**Example:**

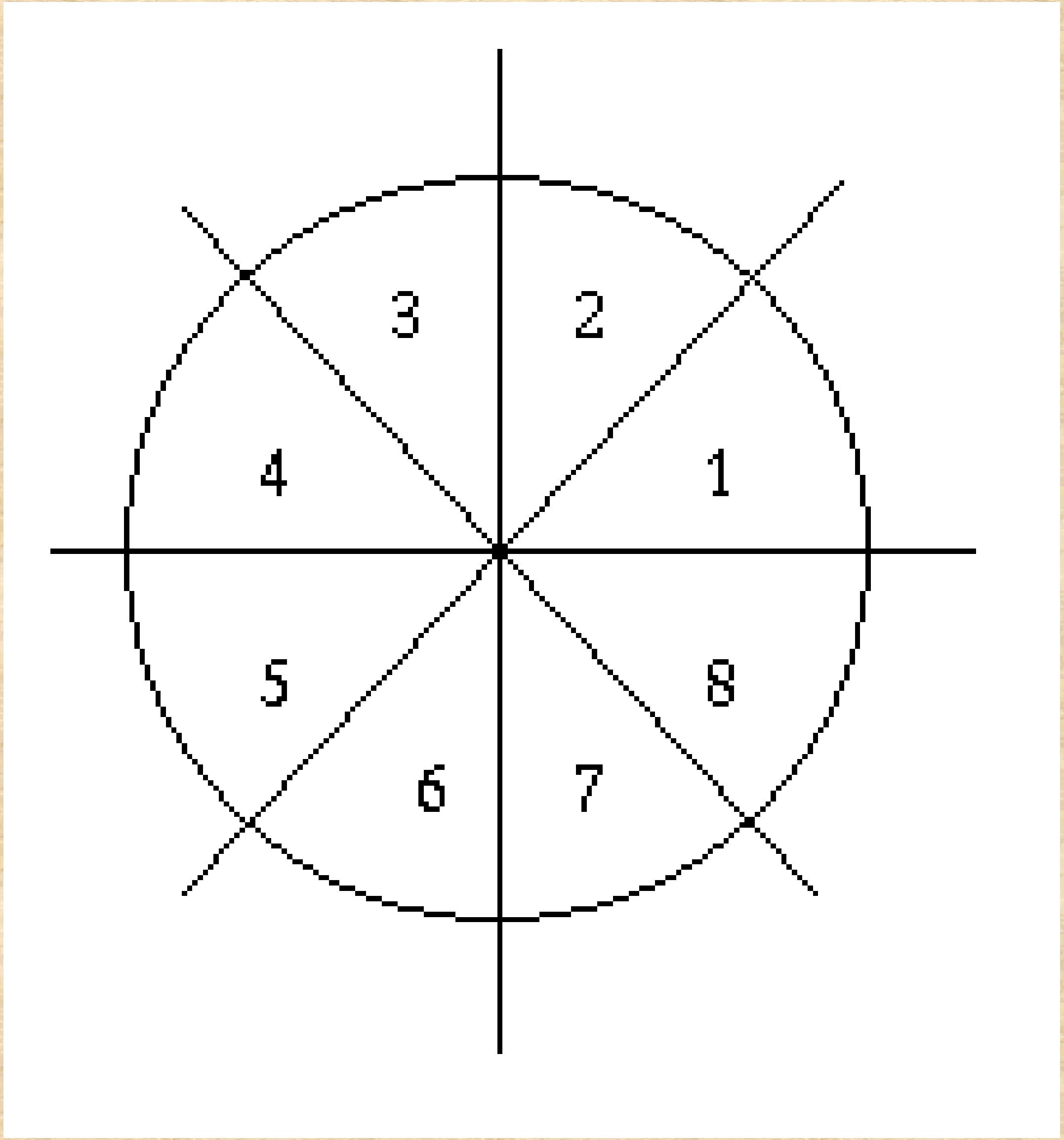
**Starting point:**
**5,8**
**Ending point:**
**9,11**

**Successive steps:**

- d=2, (6,9)
- d=0, (7,9)
- d=6, (8,10)
- d=4, (9,11)

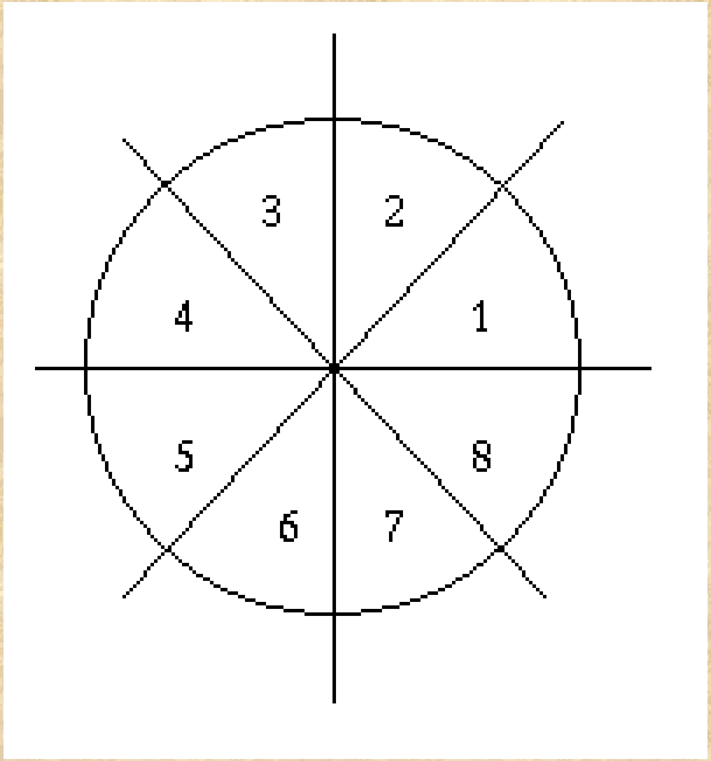


## We have considered lines in the first Quadrant only. What about the rest?



## How do you generalize this to the other octants?

| Octant | Change |
|---|---|
| 1 | none. |
| 2 | Switch roles of x & y. |
| 3 | Switch roles of x & y. Use (4) |
| 4 | Draw from P1 to P0; Use (8) |
| 5 | Draw from P1 to P0. |
| 6 | Draw from P1 to P0; Use (2) |
| 7 | Switch roles of x & y, Use (8) |
| 8 | Use y = y - 1. |



**Draw from P1 to P0:**

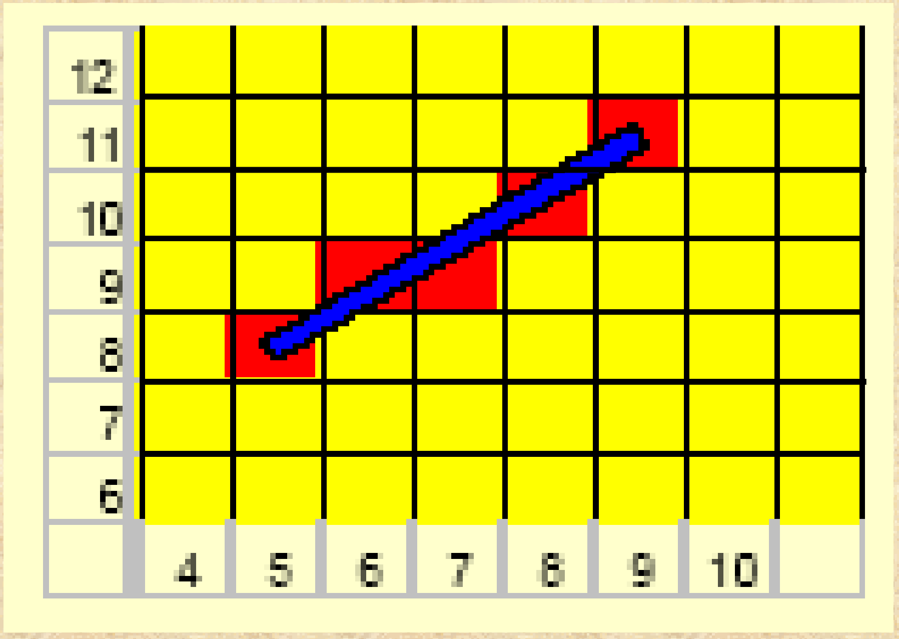
swap(P0,P1).

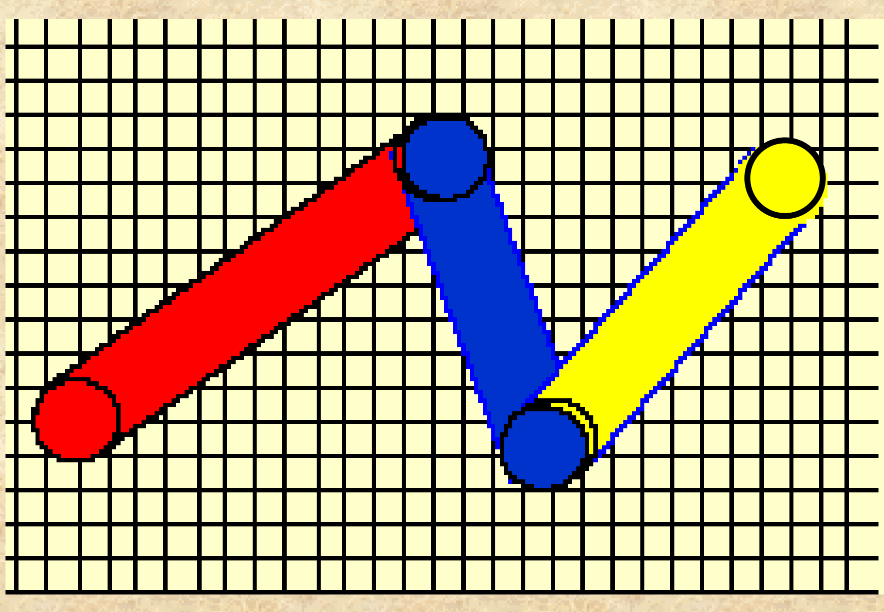
**Use y = y-1:**

dy=-dy; y=y-1.

**Switch X & Y:**

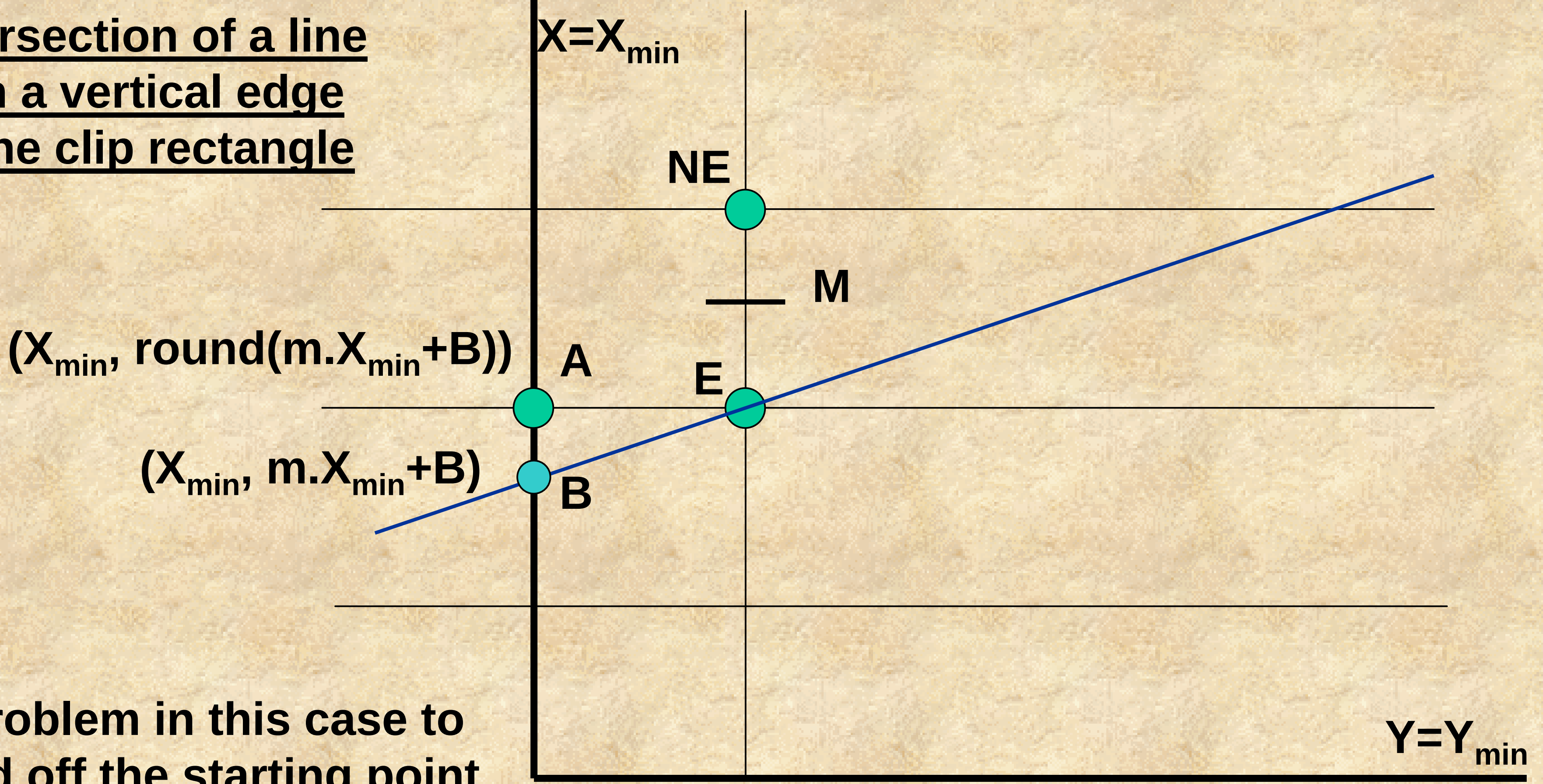
Swap (x1, y1), Swap (x0, y0 ), Swap (dx, dy),
plotpoint(y,x)





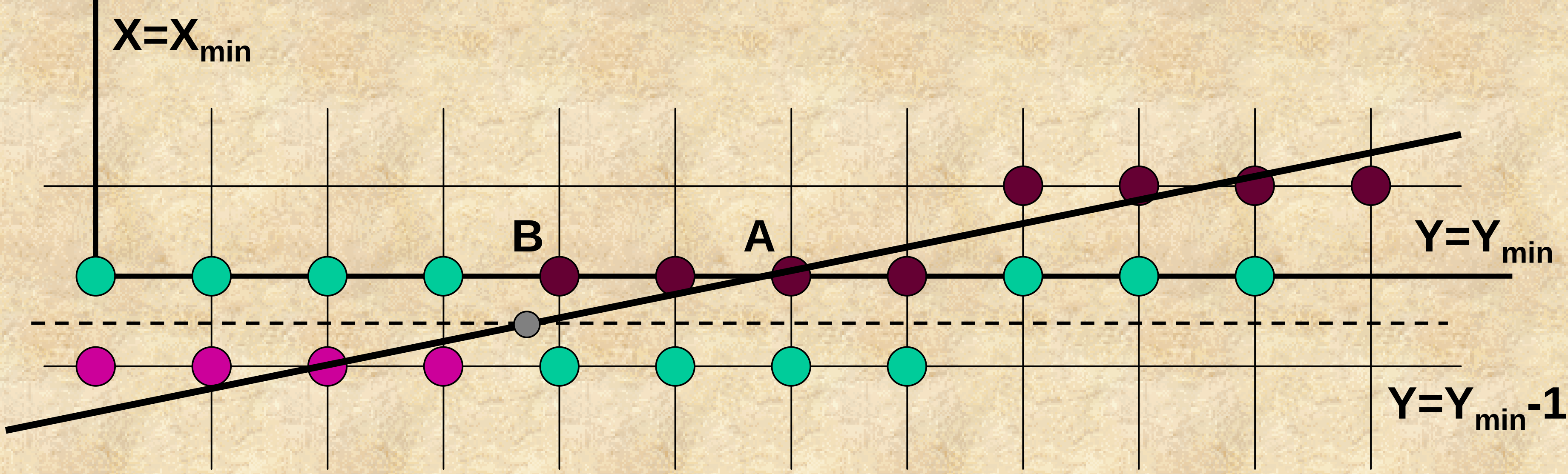
**Issues:** **Staircasing, Fat lines, end-effects and end-point ordering.**

## Intersection of a line with a vertical edge of the clip rectangle



No problem in this case to round off the starting point, as that would have been a point selected by mid-point criteria too.
Select A by rounding the intersection point coordinates at B.

What about $d_{start}$?
If you initialize the algorithm from A, and then scan convert, you are basically changing "dy" and hence the original slope of the line.
Hence, start by initializing from d(M), the mid-point in the next column, $X_{min}+1$, after clipping).



**<u>Intersection of a shallow line with a horizontal edge of the clip rectangle</u>**

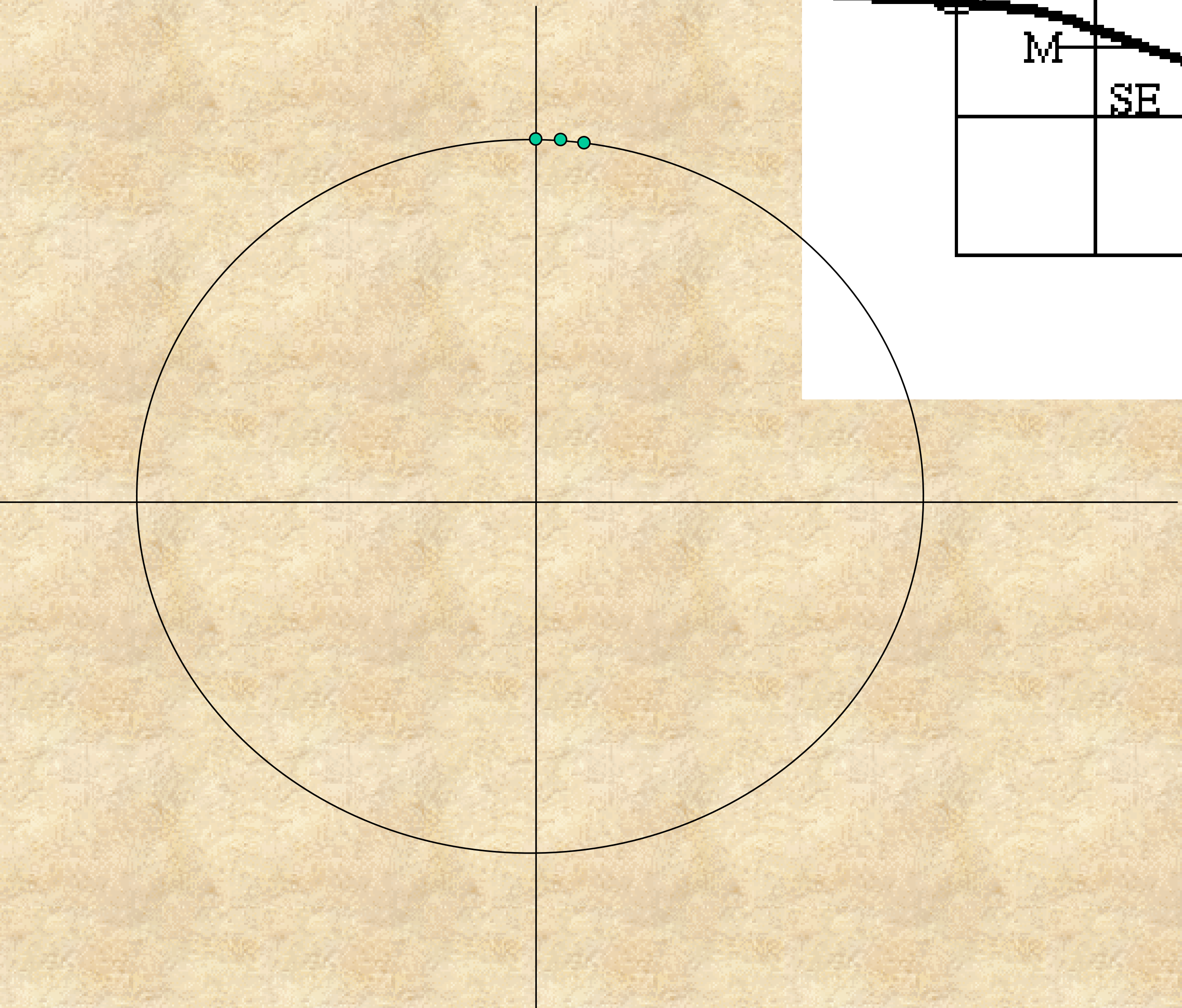
**Intersection of line with edge and then rounding off produces A, not B.**

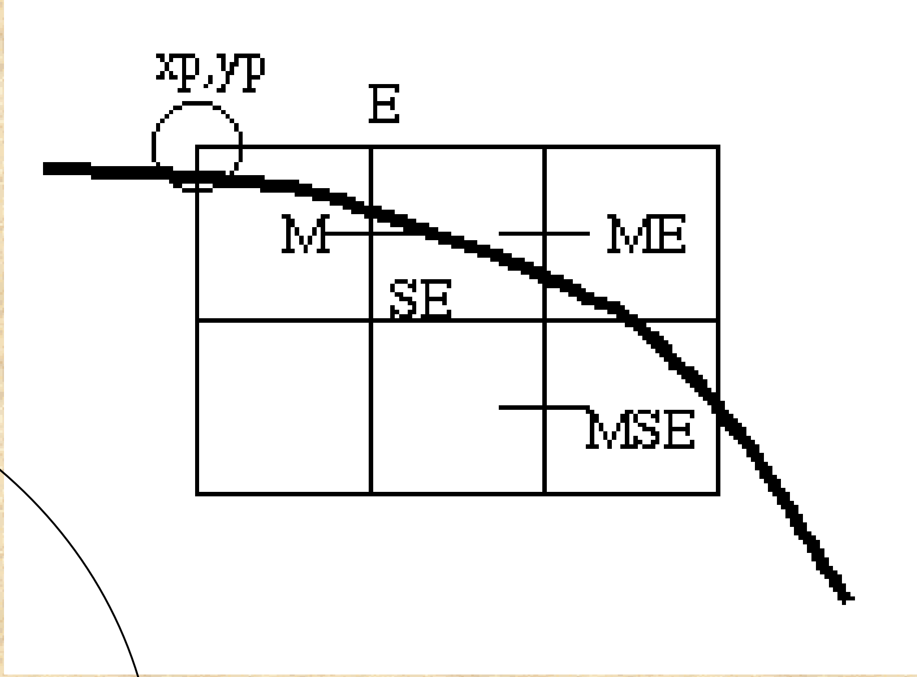
**To get B, as a part of the clipped line:**

**Obtain intersection of line with $Y_{min}$-1/2 and then round off.**

**B = $[\text{round}(X|_{Y_{min}-1/2}), Y_{min}]$**

**CIRCLE DRAWING**





## CIRCLE DRAWING

**Only considers circles centered at the origin with integer radii. Can apply translations to get non-origin centered circles.**

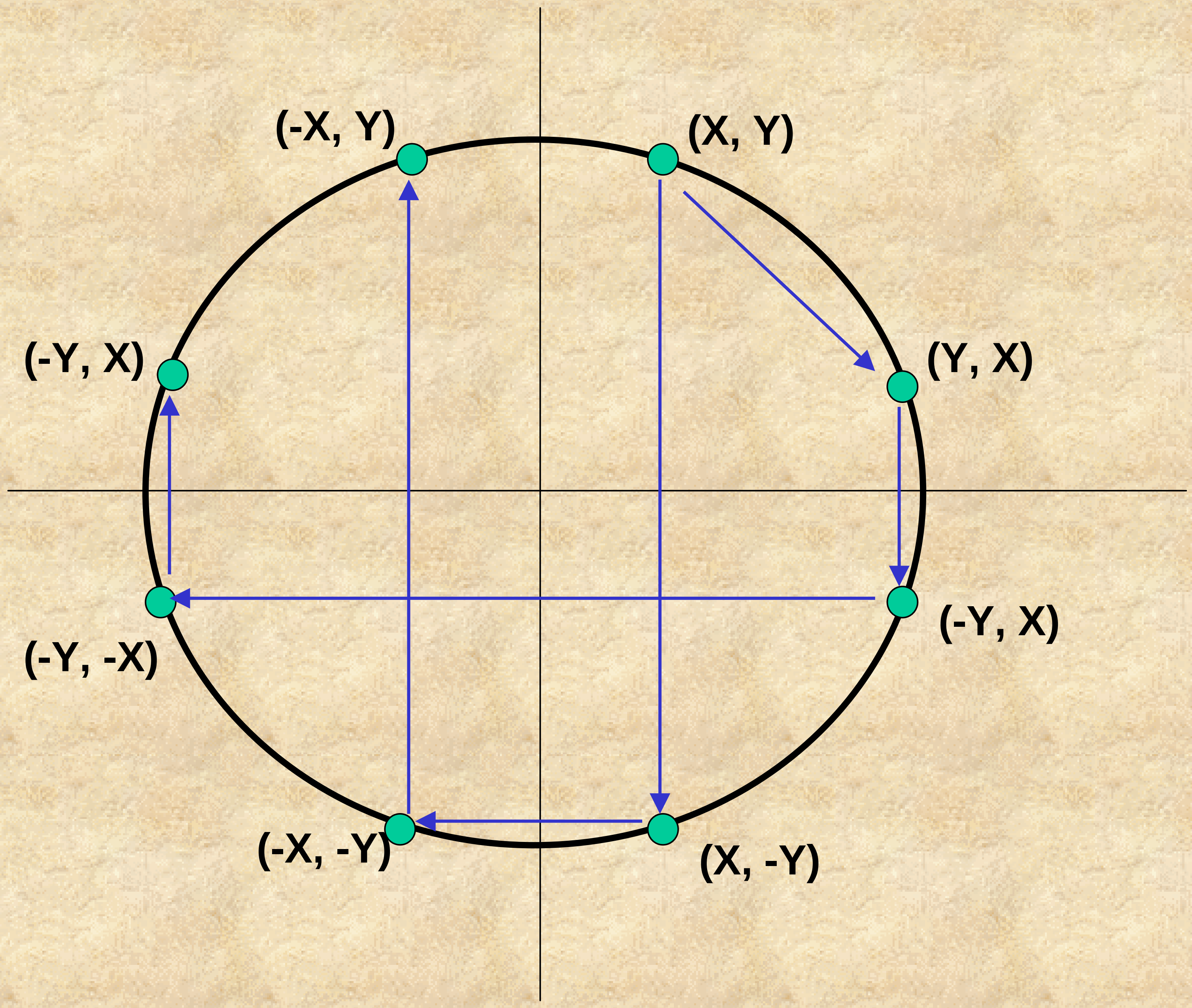
**Explicit equation: $y = \pm \mathrm{sqrt}(R^2 - x^2)$**

**Implicit equation: $F(x,y) = x^2 + y^2 - R^2 = 0$**

**Note: Implicit equations used extensively for advanced modeling (e.g., liquid metal creature from "Terminator 2")**

**Use of Symmetry: Only need to calculate one octant, can get points in the other 7 as follows:**

```
Draw_circle(x,y)
        Plotpoint (x,y) ;           Plotpoint (x,-y) ;
        Plotpoint (-x,y) ;          Plotpoint (-x, -y) ;
        Plotpoint (y,x) ;           Plotpoint (y, -x) ;
        Plotpoint (-y, x) ;         Plotpoint ( -y, -x) ;
end
```


(-X, Y)
(X, Y)
(-Y, X)
(Y, X)
(-Y, -X)
(-Y, X)
(-X, -Y)
(X, -Y)

# MIDPOINT CIRCLE ALGORITHM

**Will calculate for the second octant. Use draw_circle procedure to calculate the rest. Now will choose between pixel S and SE.**

**$F(x,y) = x^2 + y^2 - R^2 = 0$**
**$F(x,y) > 0$ if point is outside the circle**
**$F(x,y) < 0$ if point inside the circle.**

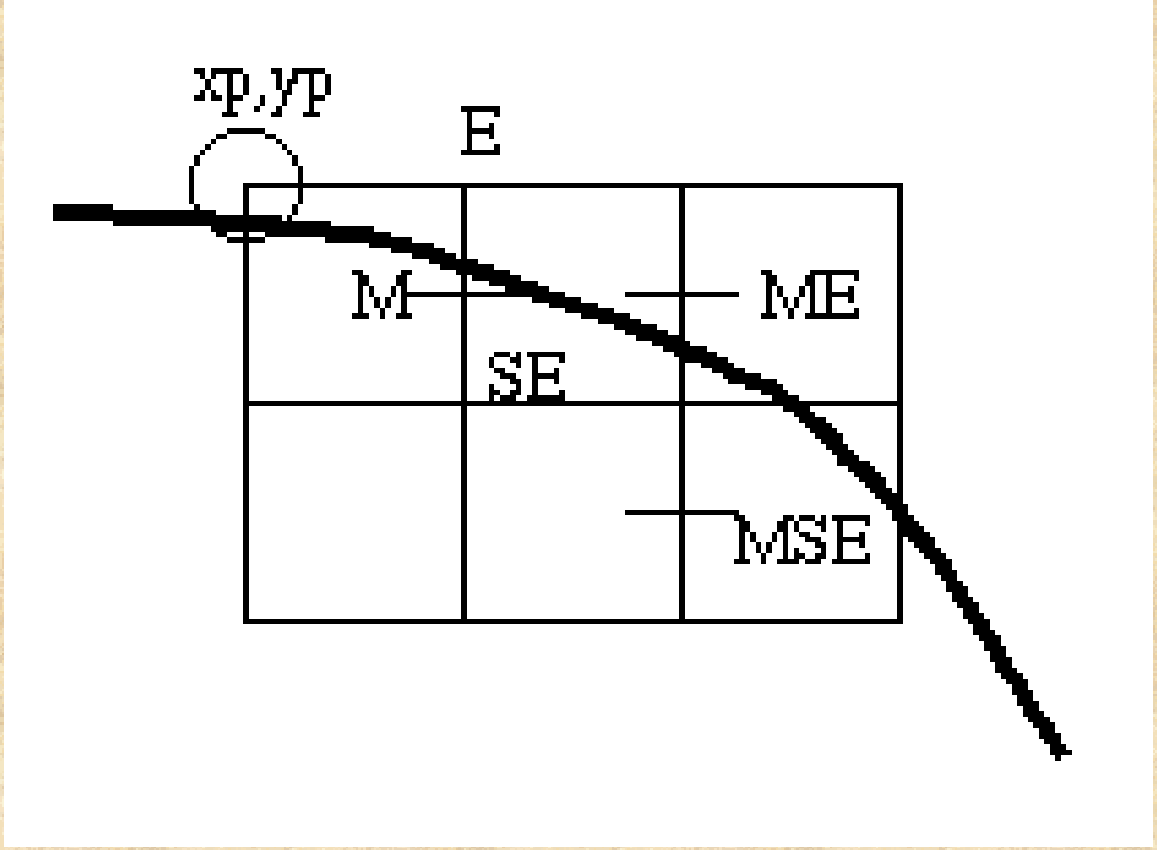
**Again, use $d_{old} = F(M)$ ; $F(M) = F(X_p+1, Y_p-1/2) = (X_p+1)^2 + (Y_p-1/2)^2 - R^2$**

**$d >= 0$ choose SE**
**next midpoint: $M_{new}$; Increment + 1 in X, -1 in y; which gives $d_{new}$**
**$d < 0$ choose E**
**next midpoint: $M_{new}$; Increment + 1 in X ; which gives = $d_{new}$**

$$(\Delta d)_E = d_{new} - d_{old} = F(X_p+2, Y_p-1/2) - F(X_p+1, Y_p-1/2)$$
$$= 2X_p + 3;$$

$$(\Delta d)_{SE} = F(X_p+2, Y_p-3/2) - F(X_p+1, Y_p-1/2)$$
$$= 2X_p - 2Y_p + 5 ;$$

$$d_{start} = F(X_0+1, Y_0 - 1/2) = F(1, R-1/2)$$
$$= 1 + (R-1/2)^2 - R^2 = 1 + R^2 - R + 1/4 - R^2 = 5/4 - R$$



To get rid of the fraction, Let $h = d - 1/4$ => $h_{start} = 1 - R$

Comparison is: $h < -1/4$.

Since h is initialized to & incremented by integers, so we can just do with: $h < 0$

## The Midpoint Circle algorithm: (Version 1)

```
x=0;
y=R;
h = 1 – R;
DrawCircle(x, y);
while (y > x)
    if h < 0          /* select E */
        h = h + 2x + 3;
    else              /* select SE */
        h = h + 2(x-y) +5;
        y = y -1;
    endif
    x = x +1;
    DrawCircle(x, y);
end_while
```

**Problems with this?**

**Requires at least**
**1 multiply and 3 adds per pixel.**

**Why?**
**Because $(\Delta d)_E$ and $(\Delta d)_{SE}$ are linear functions and not constants.**

**All we have to do is calculate the differences for: $(\Delta d)_E$ and $(\Delta d)_{SE}$ (check if these will be constants). Say, $(\Delta d)^2_E$ and $(\Delta d)^2_{SE}$.**

**If we chose E, then we calculate $(\Delta d)^2_{E/E}$ and $(\Delta d)^2_{E/SE}$ based on this. Same if we chose SE. Calculate $(\Delta d)^2_{SE/E}$ and $(\Delta d)^2_{SE/SE}$**

**If we chose E, go from $(X_p, Y_p)$ to $(X_p+1, Y_p)$**

$(\Delta d)_{Eold} = 2X_p + 3$, $(\Delta d)_{Enew} = 2X_p + 5$.

Thus $(\Delta d)^2_{E/E} = 2$.

$(\Delta d)_{SEold} = 2X_p - 2Y_p + 5$,

$(\Delta d)_{SEnew} = 2(X_p+1) - 2Y_p + 5$

Thus $(\Delta d)^2_{E/SE} = 2$.

**If we chose SE, go from $(X_p, Y_p)$ to $(X_p+1, Y_p-1)$**

$(\Delta d)_{Eold} = 2X_p + 3$, $(\Delta d)_{Enew} = 2X_p + 5$.

Thus $(\Delta d)^2_{SE/E} = 2$.

$(\Delta d)_{SEold} = 2X_p - 2Y_p + 5$,

$(\Delta d)_{SEnew} = 2(X_p+1) - 2(Y_p-1) + 5$

Thus $(\Delta d)^2_{SE/E} = 4$.

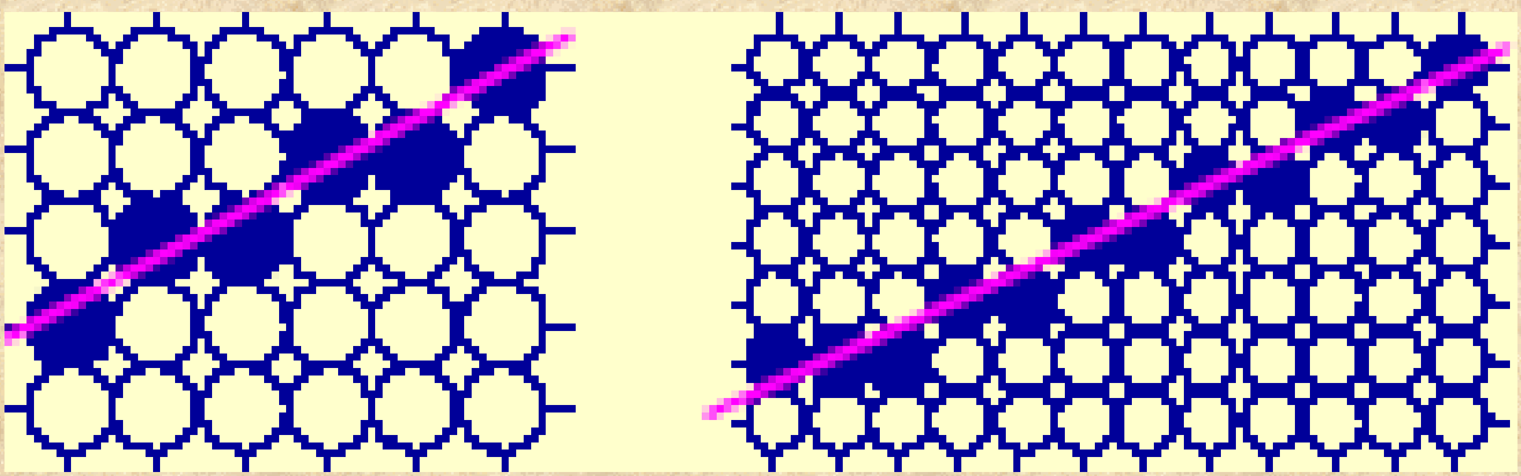
**So, at each step, we not only increment h, but we also increment $(\Delta d)_E$ and $(\Delta d)_{SE}$. What are $(\Delta d)_{E\text{-}start}$ and $(\Delta d)_{SE\text{-}start}$ ?**

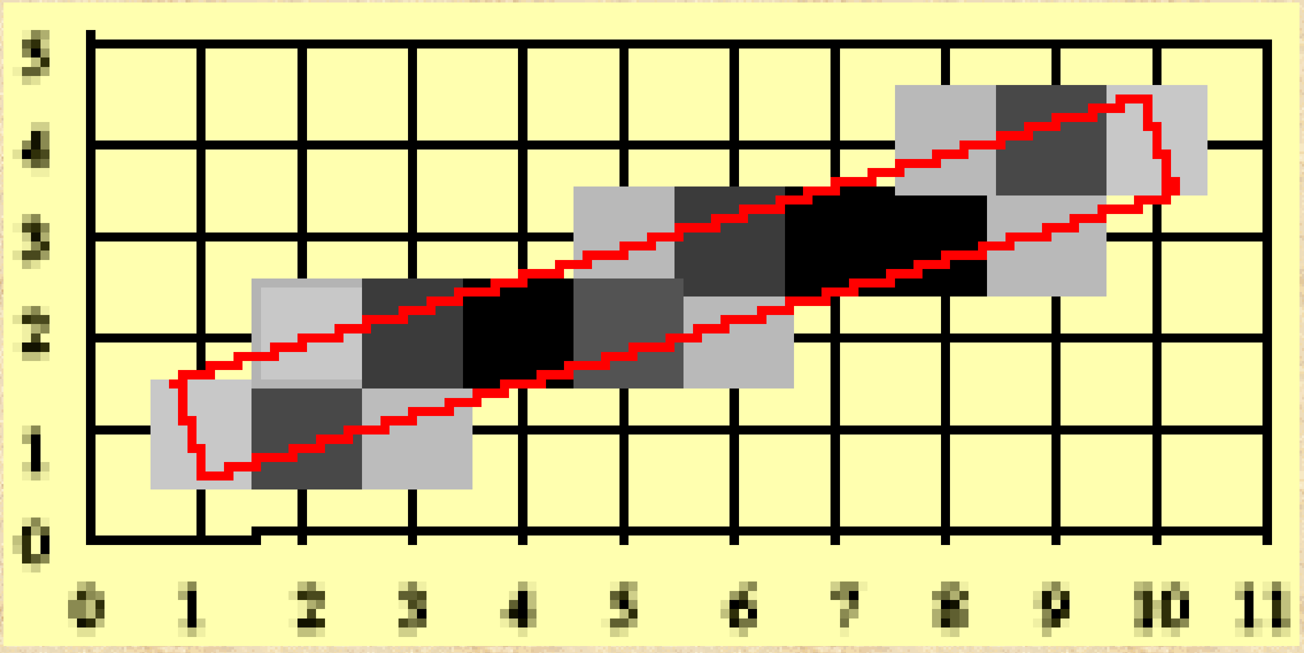
$(\Delta d)_{E\text{-}start} = 2^*(0) + 3 = 3$ ; $(\Delta d)_{SE\text{-}start} = 2^*(0) - 2^*(R) + 5$

## The MidPoint Circle Algorithm (Version 2):

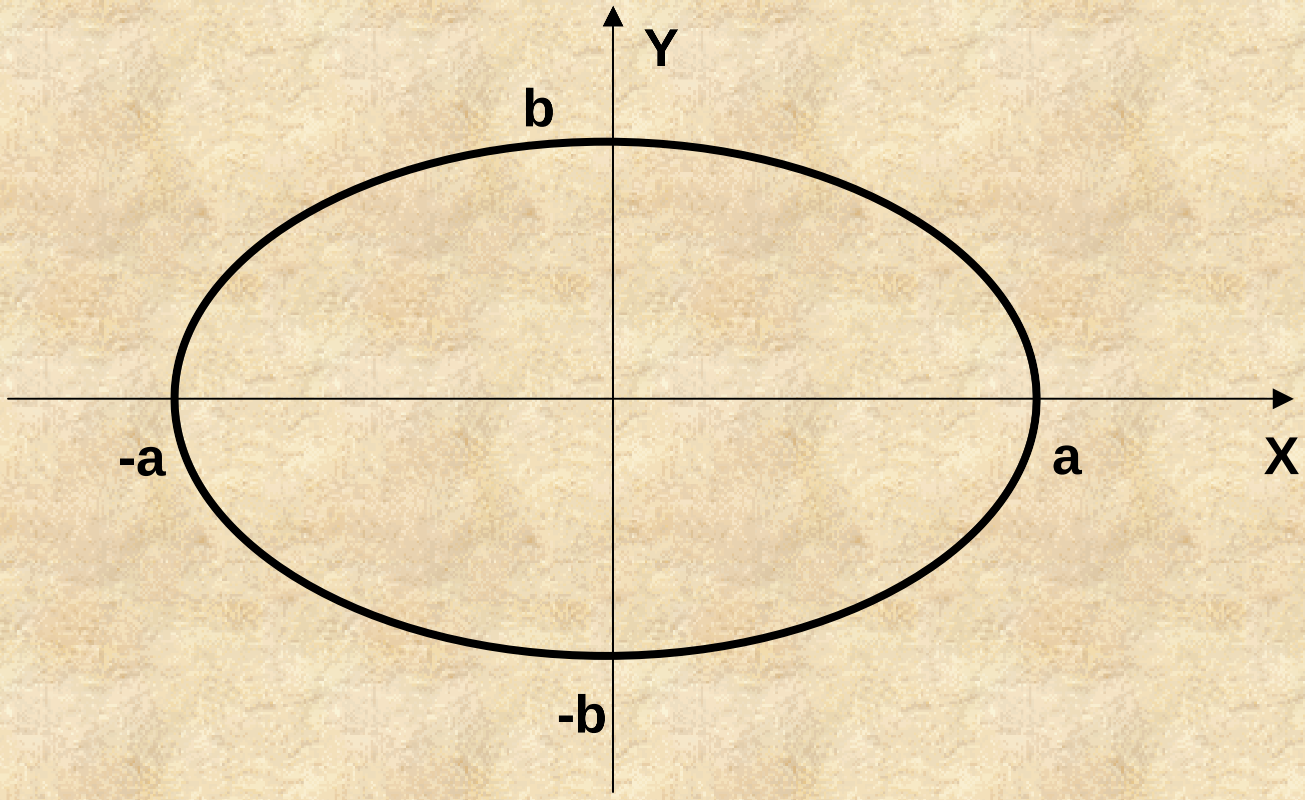
```
x = 0;                y = radius;
h = 1 – R ;
deltaE = 3 ;          deltaSE = -2*R +5 ;
DrawCircle(x, y);

while (y > x)
        if h < 0      /* select E */
                h = h +deltaE ;
                deltaE= deltaE + 2;      deltaSE= deltaSE + 2

        else          /* select SE */
                h = h + deltaSE ;
                deltaE= deltaE + 2 ;     deltaSE= deltaSE + 4
                y = y – 1 ;
        endif
        x = x+1 ;
  DrawCircle(x, y) ;
 end_while
```

# ANTI-ALIASING





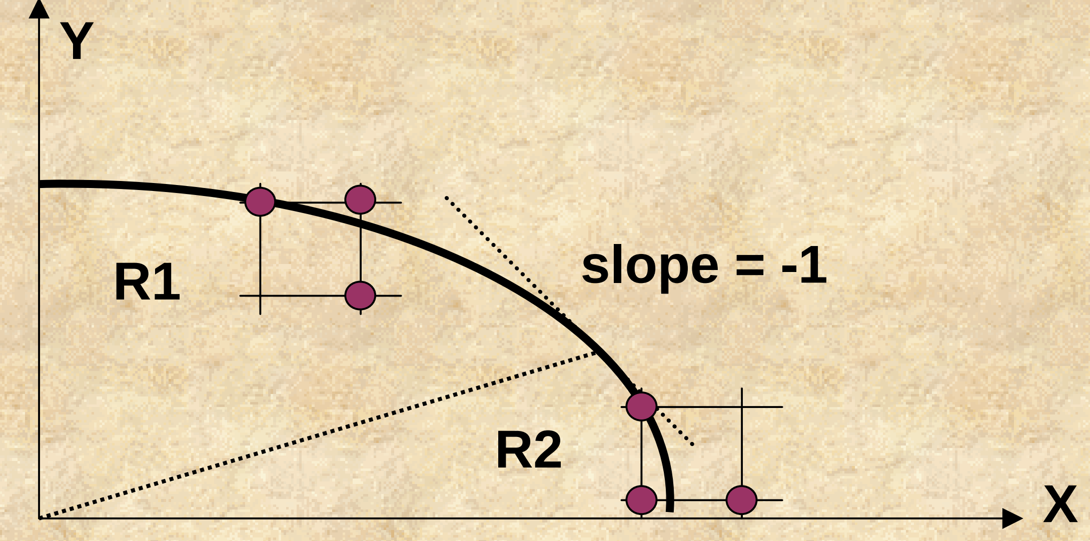
## SCAN CONVERTING ELLIPSES



**Equation of Ellipse centered at origin:**

$$F(X,Y) = b^2X^2 + a^2Y^2 - a^2b^2 = 0$$

**Length of the major axis: 2a; and minor axis: 2b.**



**Draw pixels in two regions R1 and R2, to fill up the first Quadrant. Points in other quadrants are obtained using symmetry.**

**We need to obtain the point on the contour where the slope of the curve is -1. This helps to demarcate regions R1 and R2. The choice of pixels in R1 is between E and SE, whereas in R2, it is S and SE.**

$$grad[f(X,Y)] = \frac{\partial f}{\partial X}\hat{i} + \frac{\partial f}{\partial Y}\hat{j}$$

$$= 2b^2X\,\hat{i} + 2a^2Y\,\hat{j}$$

$$In\ \mathrm{R1}, \left|\frac{\partial f}{\partial Y}\right| > \left|\frac{\partial f}{\partial X}\right|$$

and

$$in\ \mathrm{R2}, \left|\frac{\partial f}{\partial X}\right| > \left|\frac{\partial f}{\partial Y}\right|$$

**At the region boundary point on the ellipse:**

$$\left|\frac{\partial f}{\partial Y}\right| = \left|\frac{\partial f}{\partial X}\right|$$

**Based on this condition, we obtain the criteria when the next mid-point moves from R1 to R2:**

$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$

**When the above condition occurs, we switch from R1 to R2.**

**Analysis in region R1:** **Let current pixel be $(X_p, Y_p)$**

$d_{old} = F(M);$

$F(M) = F(X_p+1, Y_p-1/2) = b^2(X_p+1)^2 + a^2(Y_p-1/2)^2 - a^2b^2$

**For choice E (d<0):**

$d_{new} = F(X_p+2, Y_p-1/2)$

$= b^2(X_p+2)^2 + a^2(Y_p-1/2)^2 - a^2b^2$

$= d_{old} + b^2(2X_p + 3);$

Thus $(\Delta d)_{E1} = b^2(2X_p + 3);$

**For choice SE ($d \geq 0$):**

$d_{new} = F(X_p+2, Y_p-3/2)$

$= b^2(X_p+2)^2 + a^2(Y_p-3/2)^2 - a^2b^2$

$= d_{old} + b^2(2X_p + 3) + a^2(-2Y_p + 2);$

Thus $(\Delta d)_{SE1} = b^2(2X_p + 3) + a^2(-2Y_p + 2);$

**Initial Condition:** **In region R1, first point is (0, b).**

$(d_{init})_{R1} = F(1, b-1/2) = b^2 + a^2(1/4-b)$ ;

**Switch to Region R2, when:** $b^2(X_p + 1) \geq a^2(Y_p - 1/2)$

**Let the last point in R1 be $(X_k, Y_k)$.**

$F(M) = F(X_k+1/2, Y_k-1) = b^2(X_k +1/2)^2 + a^2(Y_k -1)^2 - a^2b^2 = (d_{init})_{R2}$

**For choice SE (d<0):**

$d_{new} = F(X_K+3/2, Y_k - 2)$
$= b^2(X_k +3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2$
$= d_{old} + b^2(2X_k + 2) + a^2(-2Y_k + 3)$;

Thus $(\Delta d)_{SE2} = b^2(2X_k + 2) + a^2(-2Y_k + 3)$;

**For choice S ($d \geq 0$):**

$d_{new} = F(X_K+1/2, Y_k - 2)$
$= b^2(X_k +1/2)^2 + a^2(Y_k - 2)^2 - a^2b^2$
$= d_{old} + a^2(-2Y_k + 3)$;

Thus $(\Delta d)_{S2} = a^2(-2Y_k + 3)$;

**Stop iteration, when $Y_k = 0$;**

**Home assignment:**
**Work out the second order differences and modify the algorithm,**

```
void MidPointEllipse (int a, int b, int value);
{
            double d2; int X = 0; int Y = 0; sa = sqr(a); sb = sqr(b);
            double d1 = sb – sa*b + 0.25*sa;
            EllipsePoints(X, Y, value);        /* 4-way symmetrical pixel plotting */
            while  ( sa*(Y-0.5) > sb*(X+1))    /*Region R1 */
            {           if (d1 < 0)
                                    d1 += sb*((X<<1) + 3);
                          else
                                    { d1 += sb*((X<<1) + 3) + sa*(-(Y<<1) + 2);        Y-- ; }
                        X++ ; EllipsePoints(X, Y, value);
            }

            double d2 = sb*sqr(X+0.5) + sa*sqr(Y-1) - sa*sb;
            while  ( Y > 0)   /*Region R2 */
            {           if (d2 < 0)
                                    { d2 += sb*((X<<1) + 2) + sa*(-(Y<<1) + 3); X++; }
                          else
                                    d2 += sa*(-(Y<<1) + 3);
                         Y-- ; EllipsePoints(X, Y, value);
            }
}
```