Computer Graphics

Scan Conversion of Line,Circle and Ellipse

Scan Conversion

- What is Scan Conversion?
- Rasterization.
- Fill out the pixels.
- Line,Arc,Circle,Ellipse,Curve,Polygon,Text.

Line

- Drawing a straight line is very easy then why do we need an algorithm?
- What is line?
- Display Device-Matrix of Pixels,We have to find out a finite set of pixels that form a line.
- Addressable pixels.

Goals

- Straight line should be straight-Rasterization
- Line should start and end accurately.-End points.
- Line should have constant intensity,brightness throughout-pixels placed with gap.
- Line should be drawn as quick as possible.

General method

- Equation of line:
- $y=mx+b \longrightarrow 1$
- m=slope of line;
- (0,b) is the y-intercept.
- Pick any values of x and substitute in equation 1.
- For any line-starting point (x1,y1) end point (x2,y2);

General method

- Increase x values in unit to get corresponding y value with the help of knowing slope.
- Slope-floating point.
- round(y);
- Eg: b=1;m=3/5; substitute in 1.
- Take x=0 to 5

  when x=0 y=1

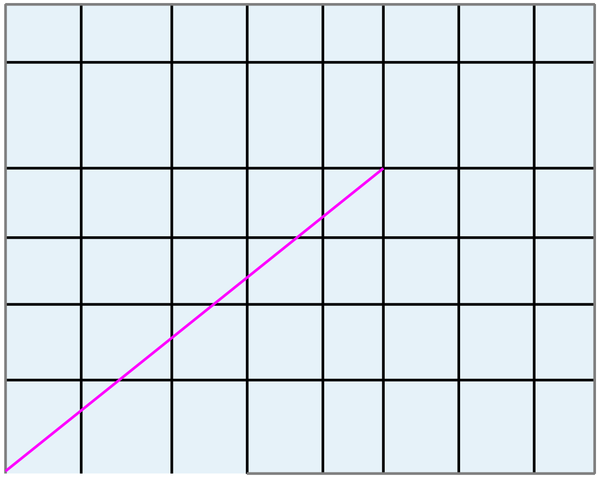
(y=(3/5*0)+1)

General method

X-Value	Round(y)	Y-Value	To Plot
0	Round(1)	1	(0,1)
1	Round(8/5)	2	(1,2)
2	Round(11/5)	2	(2,2)
3	Round(14/5)	3	(3,3)
4	Round(17/5)	3	(4,3)
5	Round(20/5)	4	(5,4)

General method

- Line drawn in graph sheet- here the line doesnt pass through any interscetion.



- We have to find out which particular pixel is to be illuminated.

- Issuses

  1.Floatiing point Multiplication and division is expensive

  2.Round function is needed

  3.Can get gaps in line. (slope > 1)

DDA-Digital Diffrential Analyzer Algorithm

- Incremental Algorithm
  - Unit Increment in either x or y axis.
  - Basic equation of line: y=mx+c ;
  - We have to know whether its x increment or y increment.

- Case 1:

  If slope(m) <1 (+ve)

- Left to right: Increment in X

X	Y
x=x+1	y=y+m

- Right to Left : Decrement in X

X	Y
x=x-1	y=y-m

- Case 2:

  if Slope(m)>1 (+ve)

- Left to right : Increment in y axis

X	Y
$x=x+1/m$	$y=y+1$

- Right to Left:Decrement in y axis.

X	Y
$x=x-1/m$	$y=y-1$

- Case 3:

  if slope(m)<1 (-ve)

  Here, x decreases y increases

X	Y
x=x-1	y=y+m

- If Slope(m)>1 (-ve)
- Here, x increases y decreases

X	Y
x=x+1/m	y=y-1

Algorithm for DDA

1. Get the input points.
2. Calculate dx and dy;
3. Calculate Lenght: if abs(dx)>=abs(dy)

   L=abs(dx) else L=abs(dy)
4. Calculate Increment Factor :xnew=dx/L ; ynew=dy/L;
5. Plot(x1,y1)
6. While (i<=L)

```
   {  x1=x1+xnew;
      y1=y1+Ynew;
      plot(x1,y1);
      i=i+1;
   }
end while
```

7. Finish

Problems to solve

1. (0,0) (4,8)
2. (0,0)(-6,-6)
3. (2,3) (12,8)

Bresenham's line algorithm

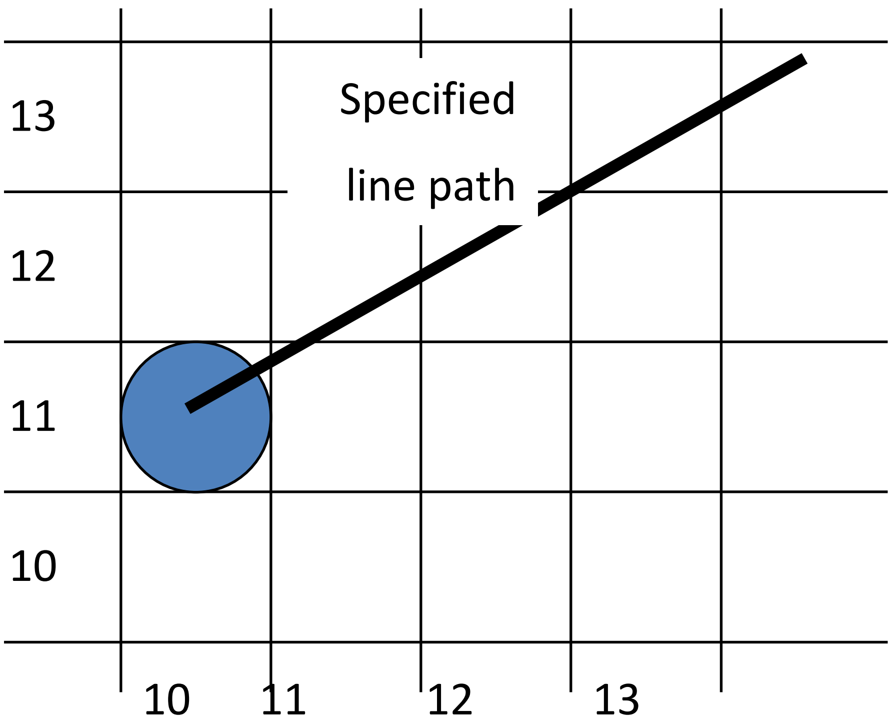
Accurate and efficient

Uses only incremental integer calculations

The method is described for a line segment with a positive slope less than one

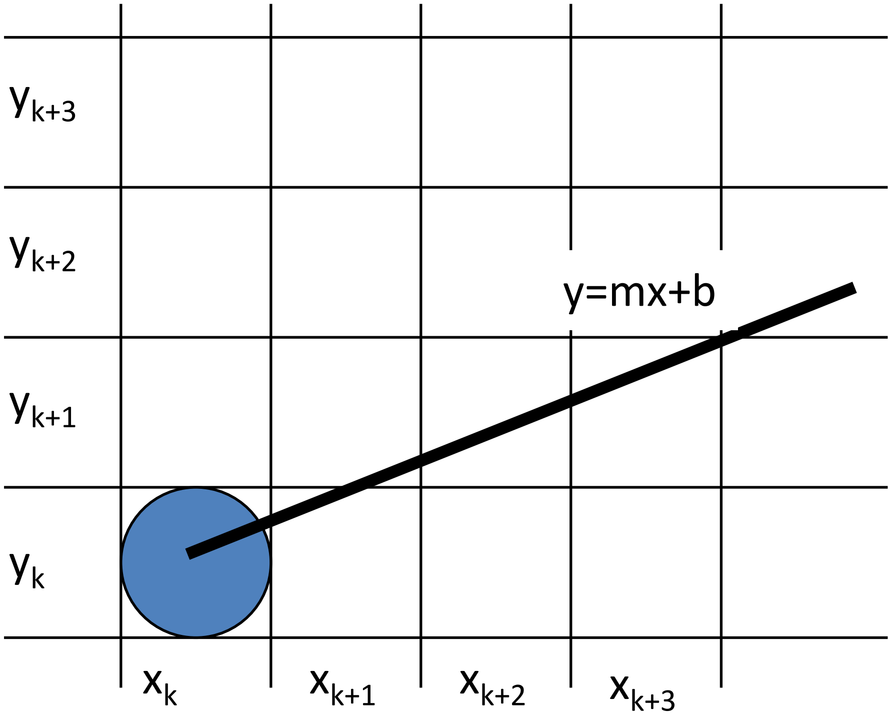
The method generalizes to line segments of other slopes by considering the symmetry between the various octants and quadrants of the xy plane

Bresenham's line algorithm



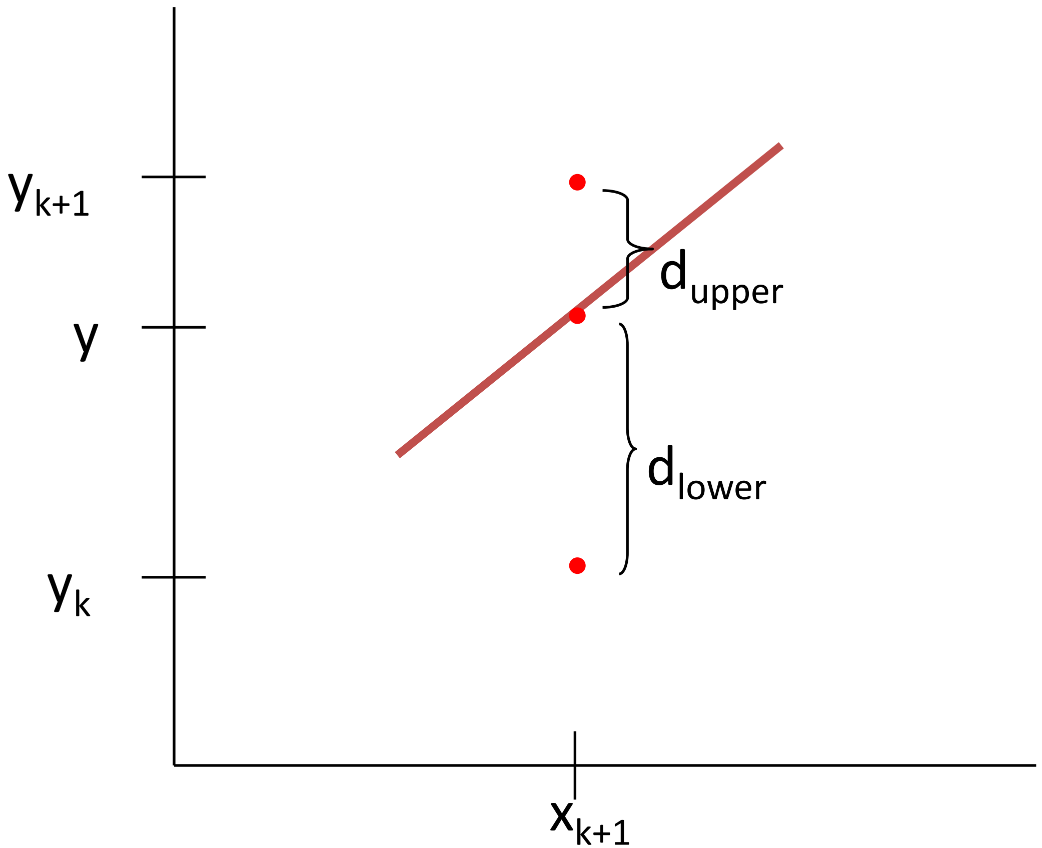
(11,11) or (11,12)
Decide what is the next pixel position

Illustrating Bresenham's Approach



For the pixel position $x_{k+1}=x_k+1$, which one we should choose: (x_{k+1},y_k) or (x_{k+1}, y_{k+1})

Bresenham's Approach



$y=m(x_k + 1)+b$

$d_{lower}=y-y_k$
$=m(x_k + 1)+b-y_k$

$d_{upper}=(y_k+1)-y$
$= y_k+1 -m(x_k + 1)-b$

- $d_{lower}- d_{upper}= 2m(x_k + 1)-2y_k+2b-1$
- Rearrange it to have integer calculations:

  $m=\Delta y/\Delta x$

  Decision parameter: $p_k= \Delta x(d_{lower}- d_{upper})=2\Delta y.x_k - 2\Delta x. y_k + c$

The Decision Parameter

Decision parameter: $p_k = \Delta x(d_{lower} - d_{upper}) = 2\Delta y.x_k - 2\Delta x.\ y_k + c$

p_k has the same sign with $d_{lower} - d_{upper}$ since $\Delta x > 0$.

c is constant and has the value $c = 2\Delta y + \Delta x(2b-1)$

- c is independent of the pixel positions and is eliminated from decision parameter p_k.

If $d_{lower} < d_{upper}$ then p_k is negative.

- Plot the lower pixel (East)

Otherwise

- Plot the upper pixel (North East)

Succesive decision parameter

At step k+1

$p_{k+1}= 2\Delta y.x_{k+1} - 2\Delta x.\ y_{k+1} + c$

Subtracting two subsequent decision parameters yields:

$p_{k+1}-p_k= 2\Delta y.(x_{k+1}-x_k) - 2\Delta x.\ (y_{k+1}-y_k)$

$x_{k+1}=x_k+1$ so

$p_{k+1}= p_k + 2\Delta y - 2\Delta x.\ (y_{k+1}-y_k)$

$y_{k+1}-y_k$ is either 0 or 1 depending on the sign of p_k

First parameter p_0

$p_0=2\ \Delta y - \Delta x$

Bresenham's Line-Drawing Algorithm for I m I < 1

1. Input the two endpoints and store the left endpoint in (x_0, y_0).

2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.

3. Calculate constants **Δx, Δy, $2\Delta y$,** and **$2\Delta y - 2\Delta x$**, and obtain the starting value for the decision parameter as

$$p_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at k = 0, perform the following test:

   If $p_k < 0$, the next point to plot is (x_{k+1}, y_k) and

   $$p_{k+1} = p_k + 2\Delta y$$

   Otherwise, the next point to plot is (x_{k+1}, y_{k+1}) and

   $$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repeat step 4 **Δx** -1 times.

Trivial Situations: Do not need Bresenham

- $m = 0 \quad \Rightarrow \quad \text{horizontal line}$
- $m = \pm 1 \quad \Rightarrow \quad \text{line } \; y = \pm x$
- $m = \infty \quad \Rightarrow \quad \text{vertical line}$

Example

Draw the line with endpoints (20,10) and (30, 18).

Δx=30-20=10*, Δy=18-10=8,*

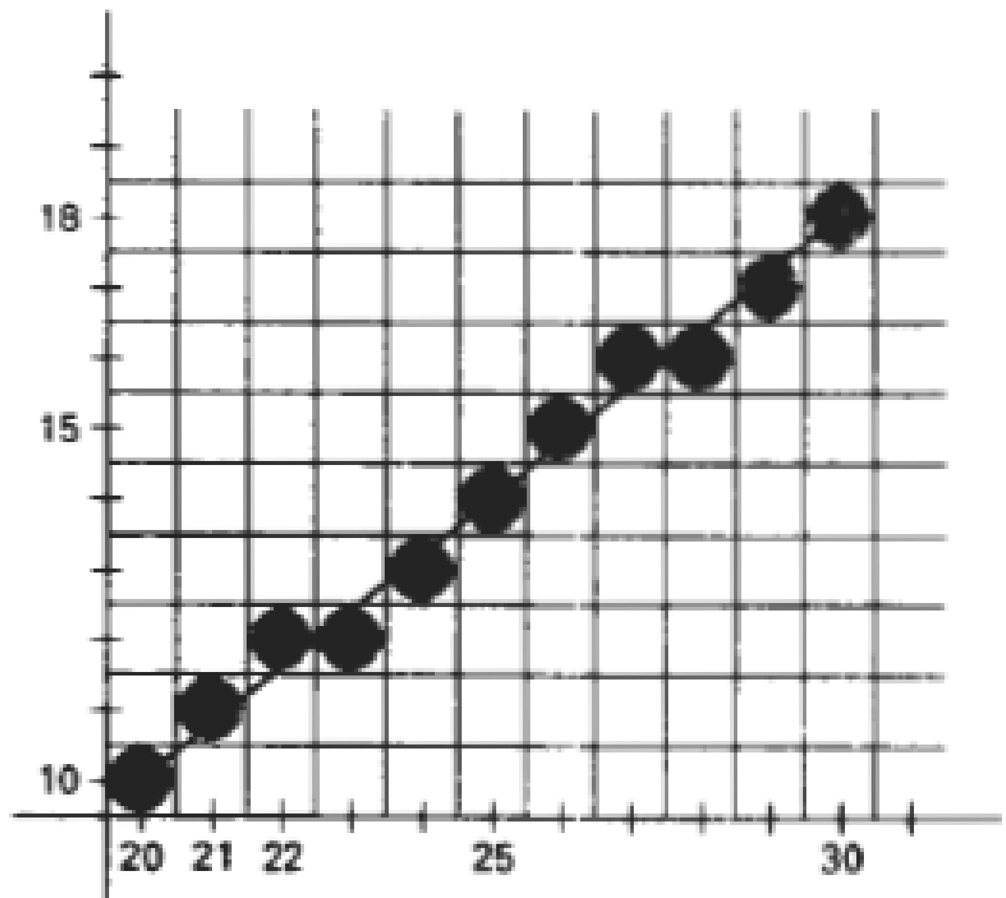
***p_0= 2Δy* – Δx=16-10=6**

2Δy=16, and ***2Δy*** - **2Δx=-4**

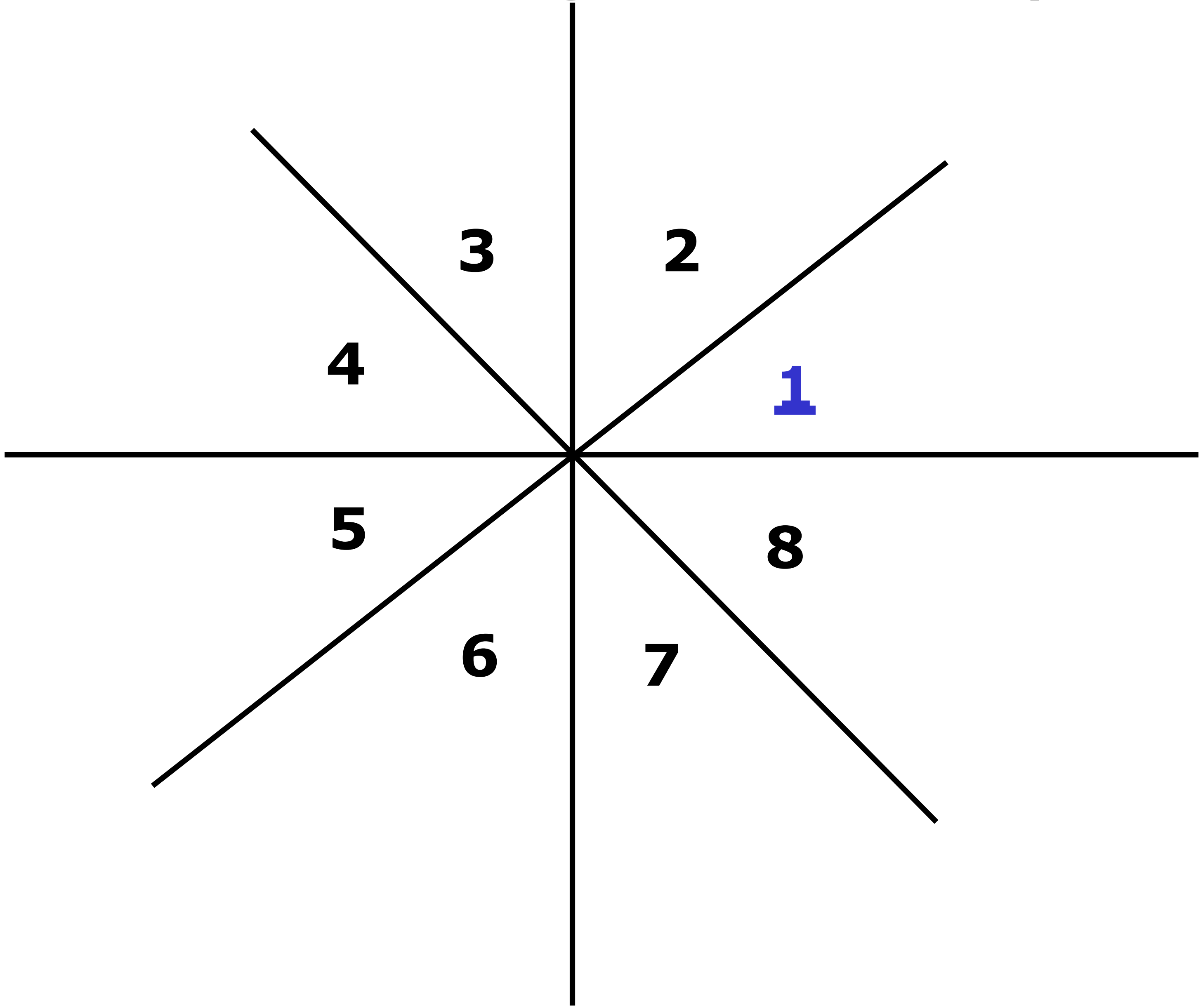
Plot the initial position at (20,10), then

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	−2	(23, 12)	7	−2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	−2	(23, 12)	7	−2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)



Octants covering the 2-D space



MIDPOINT LINE ALGORITHM

Incremental Algorithm (Assume first octant)

Given the choice of the current pixel, which one do we choose next : E or NE?

Equations:

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

2. $F(x,y) = a*x + b*y + c = 0$

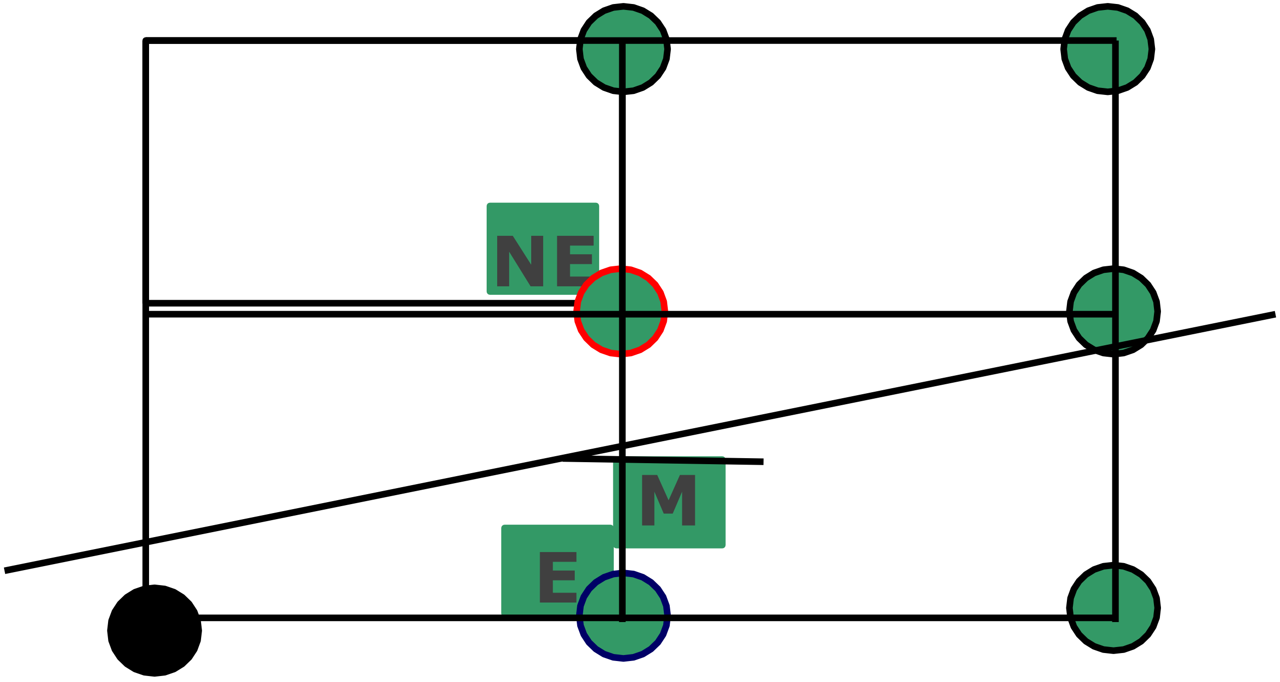
Gives: $F(x,y) = dy*x - dx*y + B*dx = 0$

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

Criteria:

Evaluate the mid-point, M, w.r.t. the equation of the line.

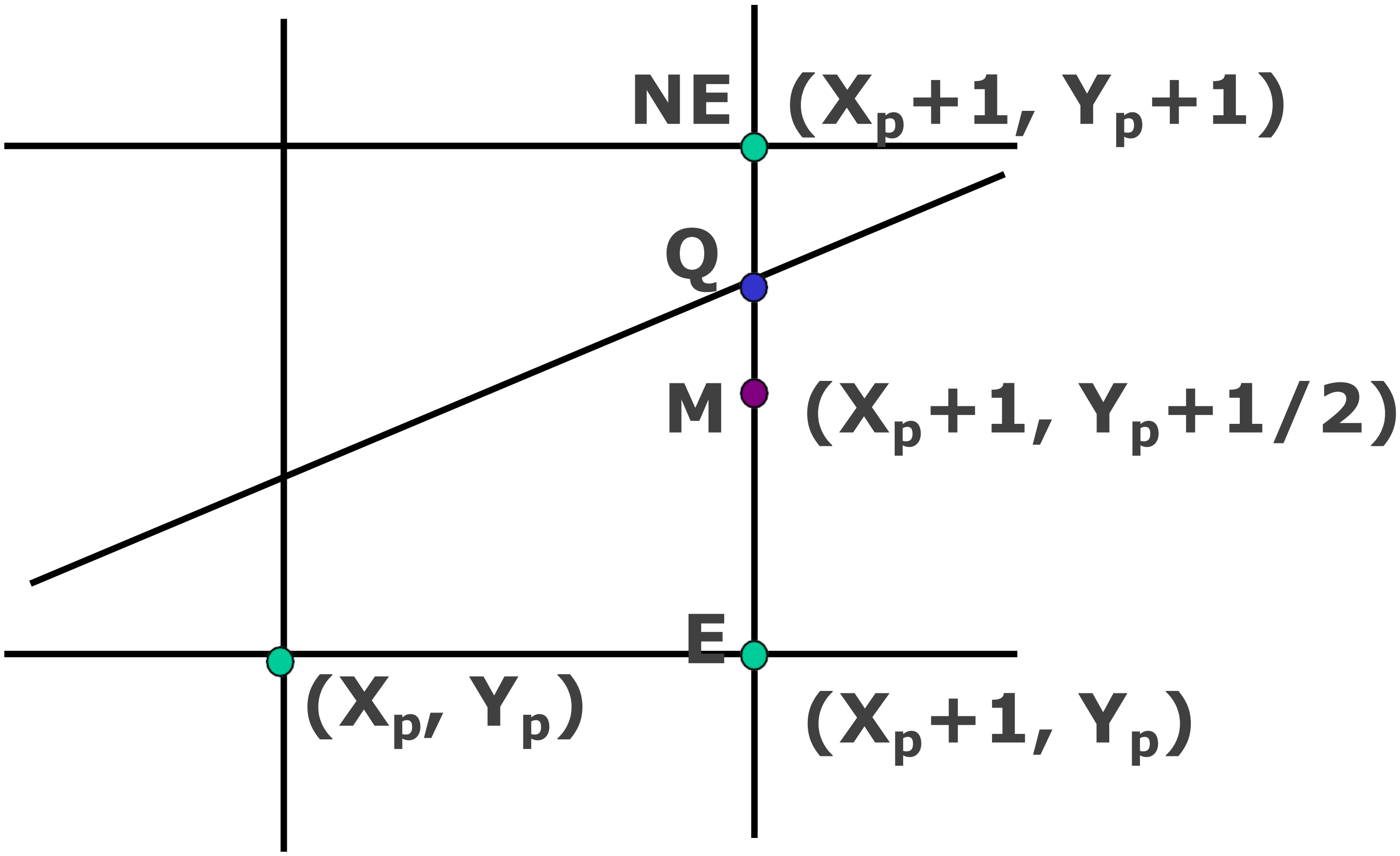
Choice: E or NE?



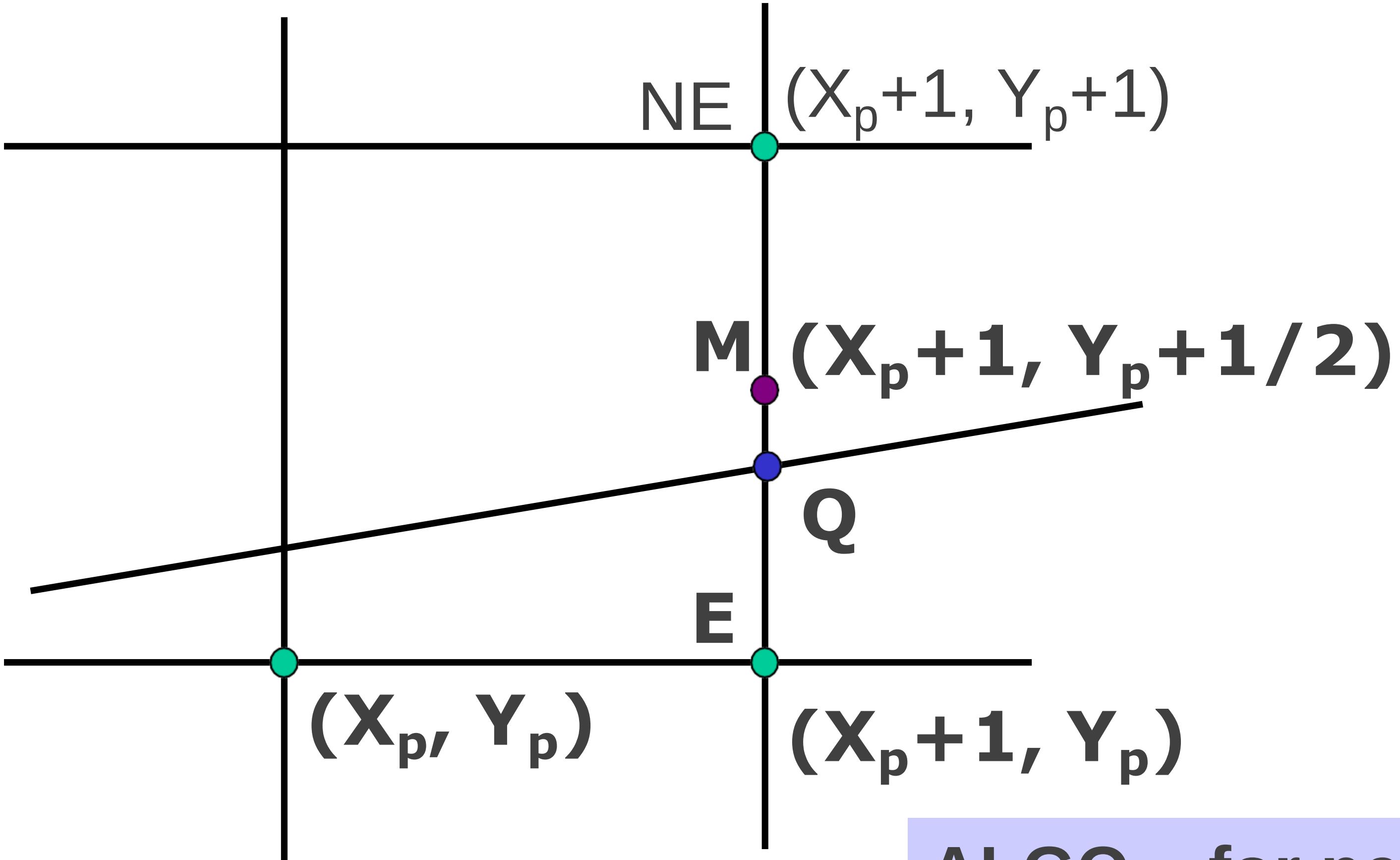
F(x,y) = dy*x - dx*y + B*dx =0

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

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



Q is above M,
hence select NE pixel as your next choice



Q is below M, hence select E pixel as your next choice

ALGO – for next choice:

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,

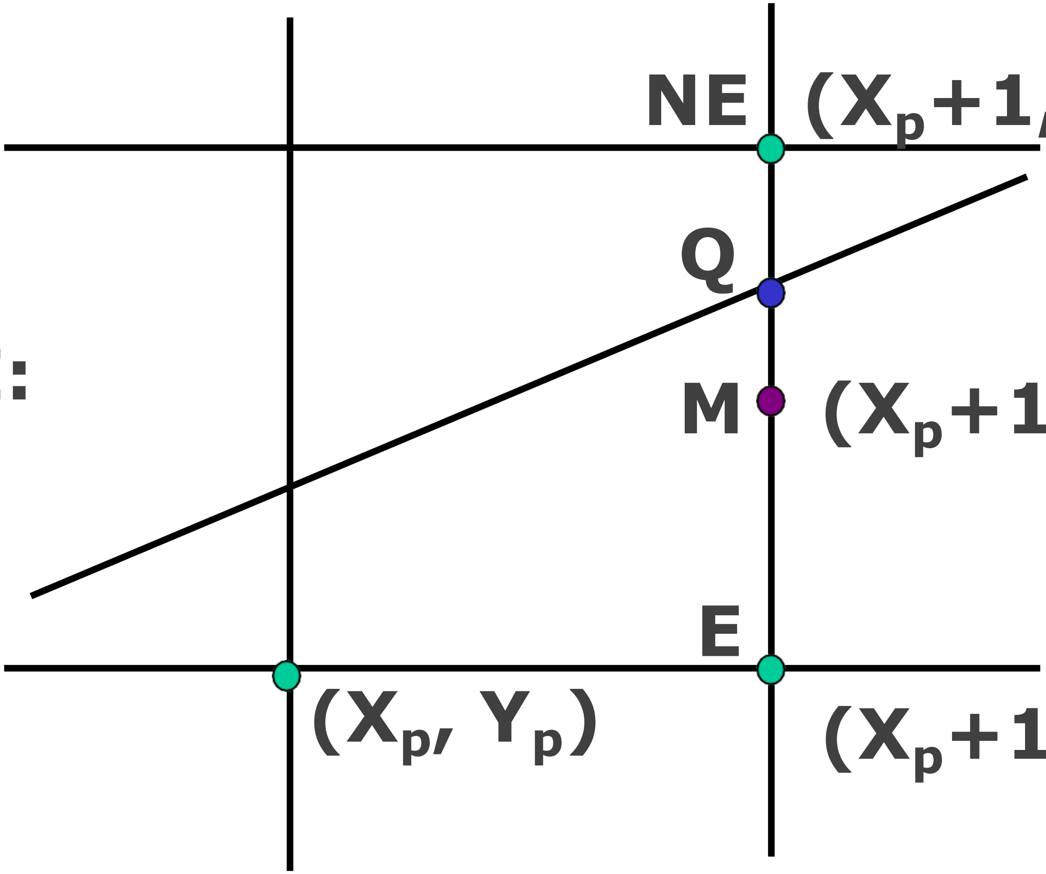
Set $d_{old} = d$;

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

Case I. Chosen E:

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

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



Case II.
Chosen NE:

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

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

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
else /* *if* $d <= 0$ */ 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 NORTH-EAST:

increment M by 1 in both x and y
$d_{new} = F(M_{new}) = F(X_p + 2, Y_p + 3/2)$
$(\Delta d)_{NE} = d_{new} - d_{old} = a + b = dy - dx$
$(\Delta d)_{NE} = dy - dx$

What is d_{start}?

$$d_{start} = F(x_0 + 1, y_0 + 1/2)$$
$$= ax_0 + a + by_0 + b/2 + c$$
$$= F(x_0, y_0) + a + b/2$$
$$= dy - dx/2$$

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

$x = x_0;$ $y = y_0;$

$dy = y_1 - y_0;$ $dx = x_1 - x_0;$

$d = 2dy - dx;$

$(\Delta d)_E = 2dy;$

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

Plot_Point(x,y)

The Midpoint Line Algorithm (Contd.)

```
while (x < x1)
        if (d <= 0)      /* Choose E */
              d = d + (Δd)E ;
        else             /* Choose NE */
              d = d + (Δd)NE ;
              y = y + 1
        endif
        x = x + 1 ;
        Plot_Point(x, y) ;
end while
```

Example:

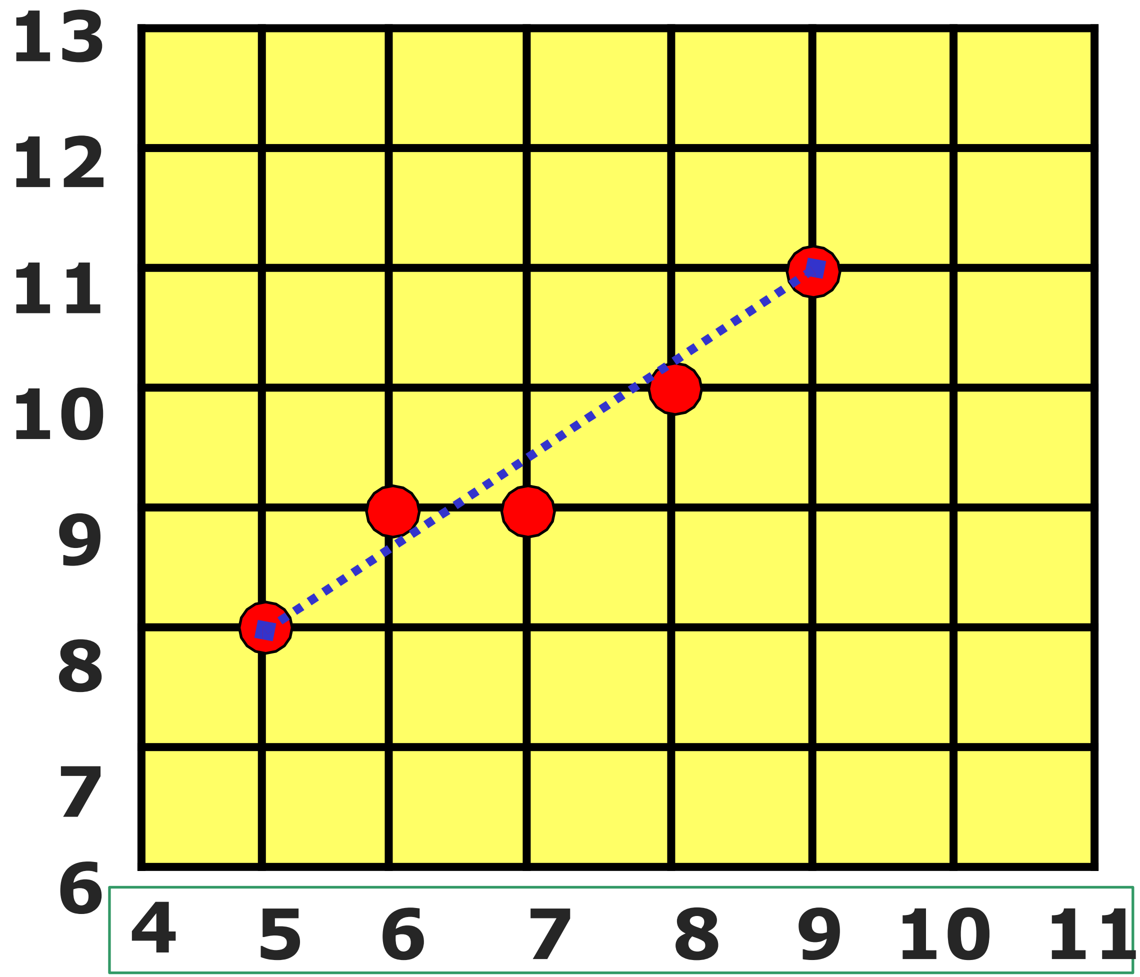
Starting point: (5, 8)
Ending point: (9, 11)

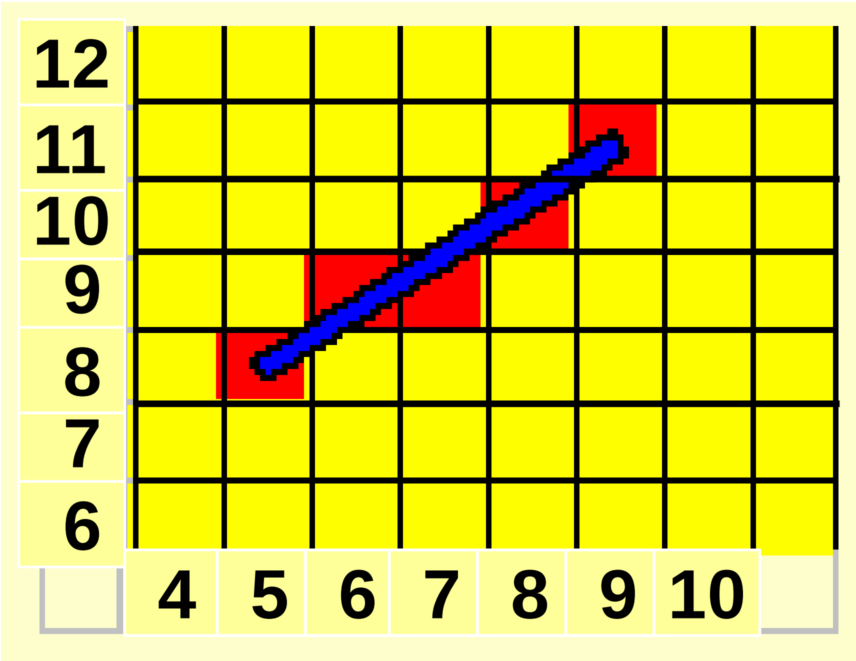
Successive steps:

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

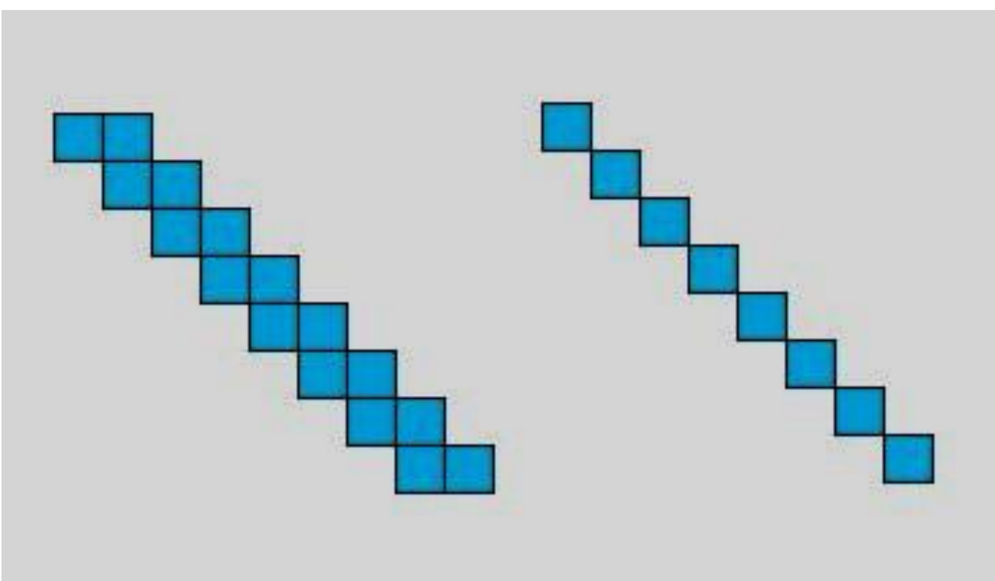
INIT: dy = 3; dx = 4; d_{start}=2;

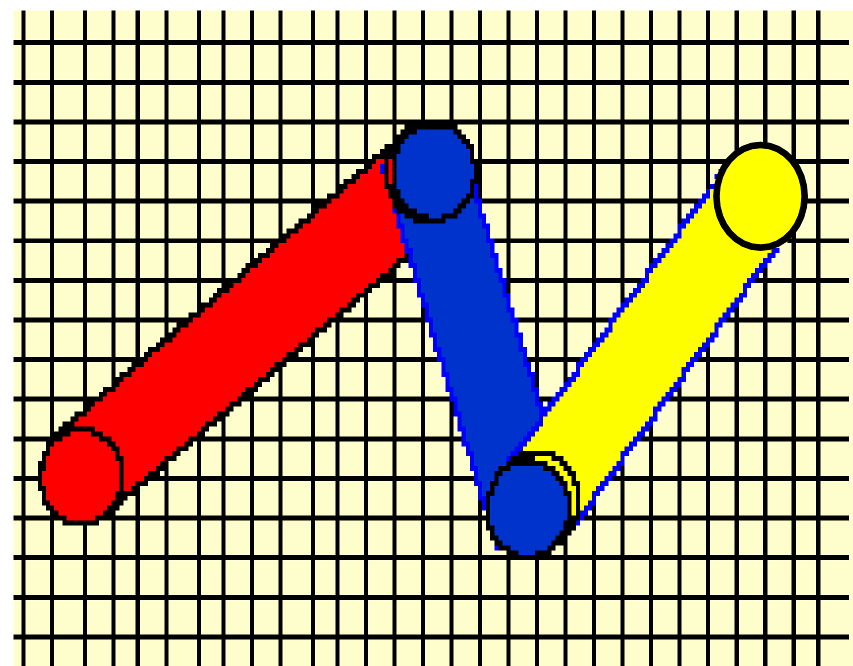
$(\Delta d)_E$ = 6; $(\Delta d)_{NE}$ = -2;



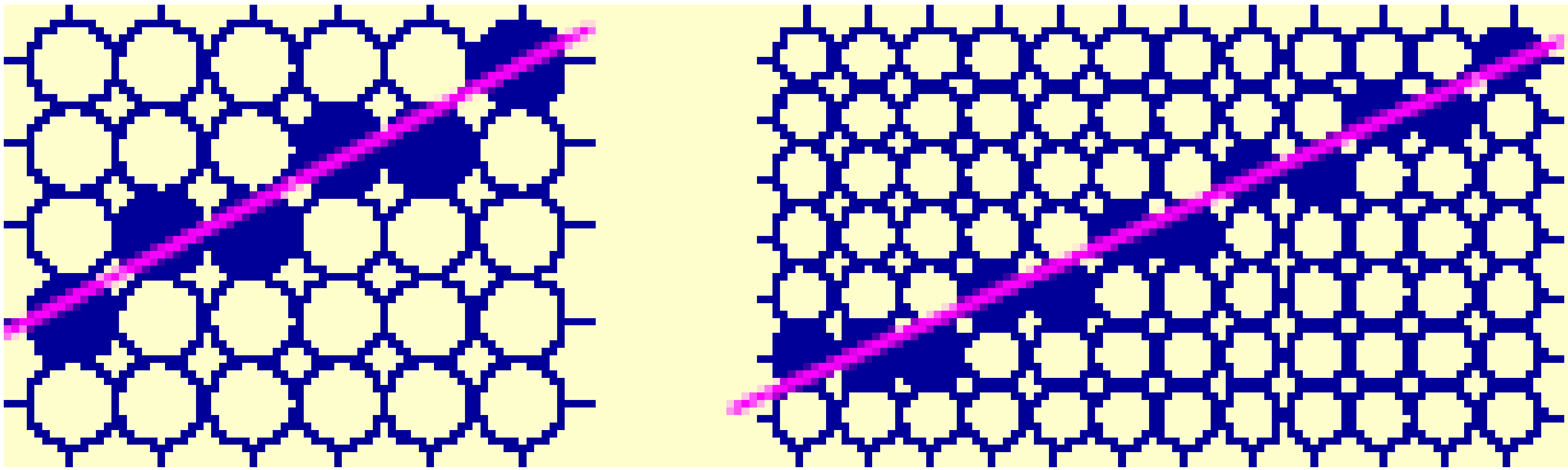


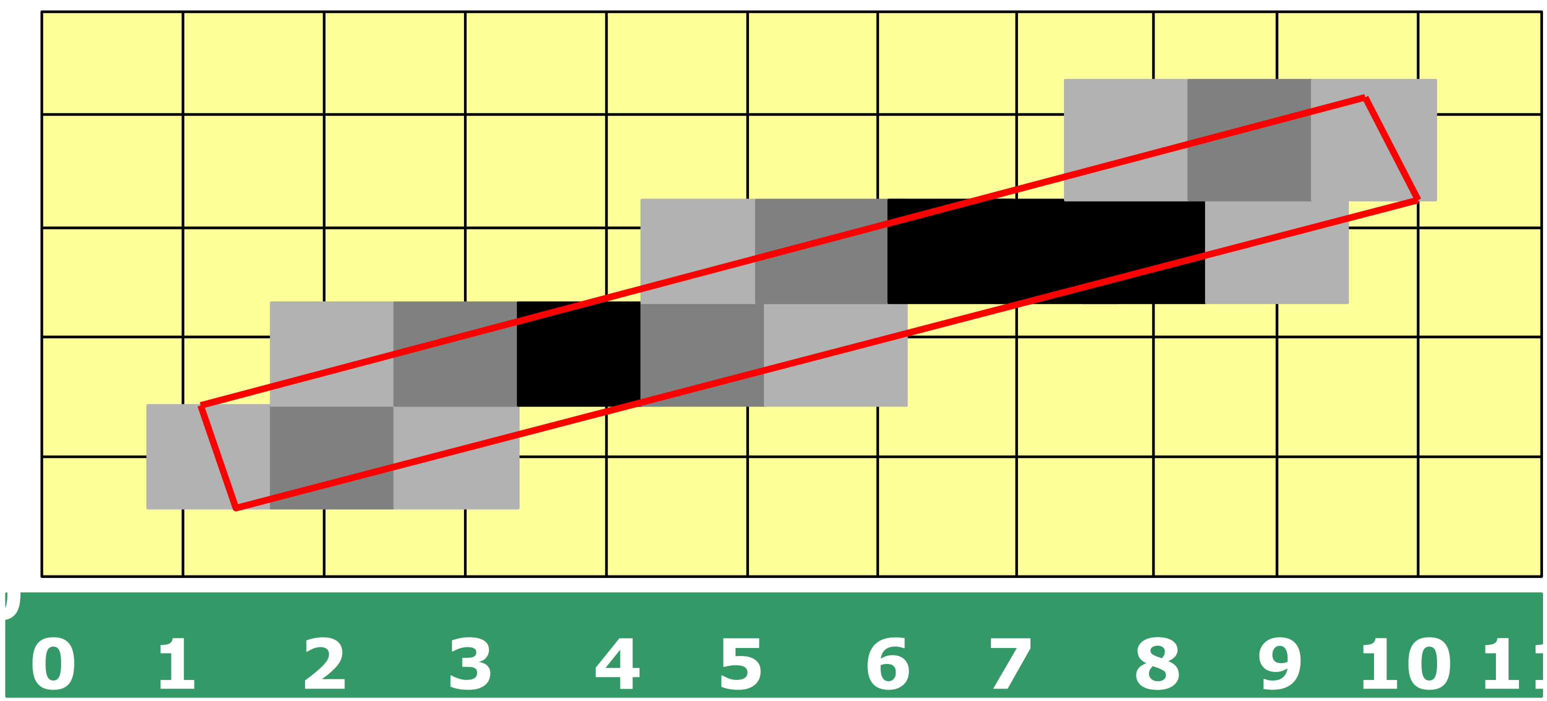
Issues: Staircasing, Fat lines, end-effects and end-point ordering.

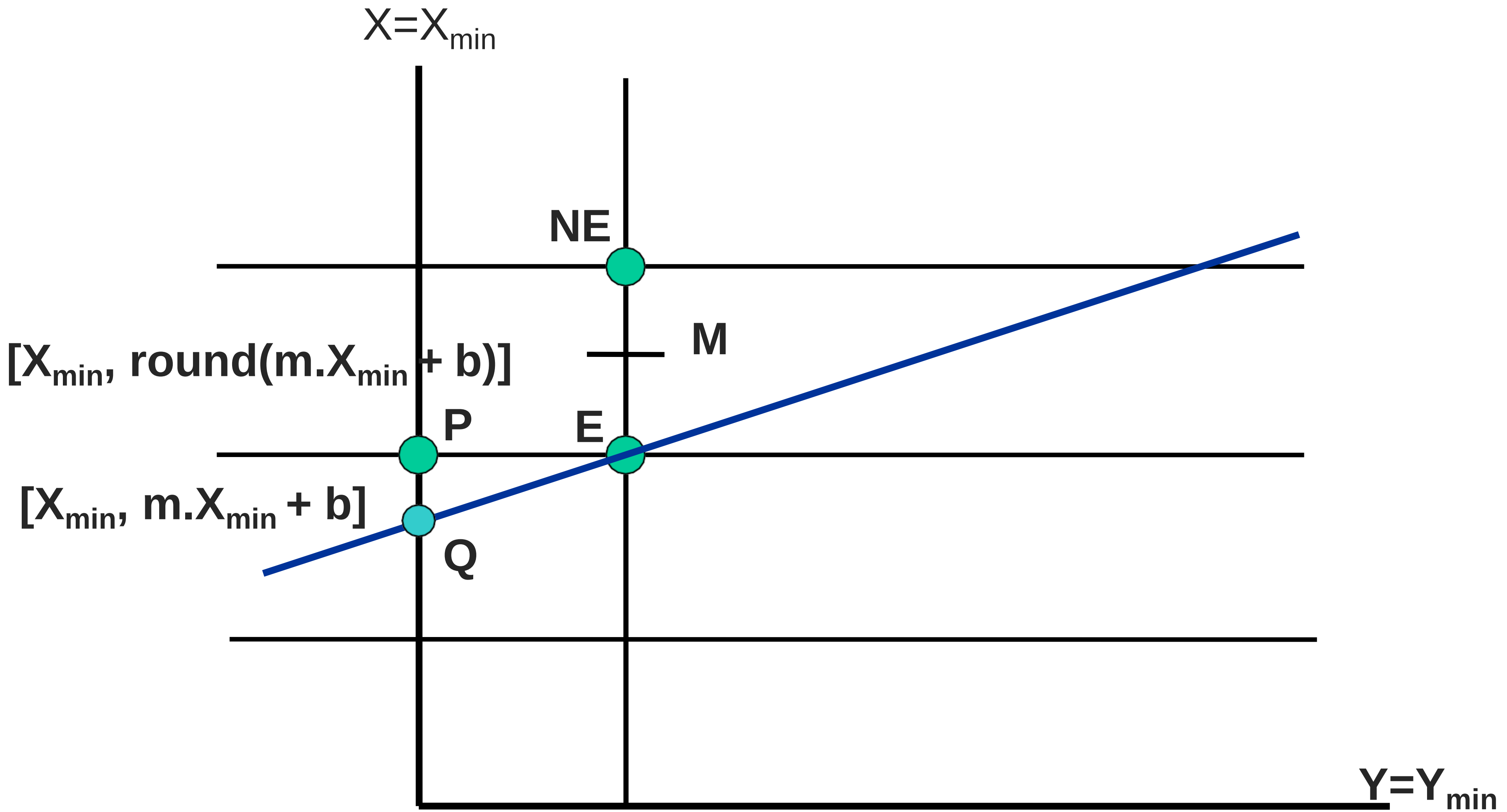




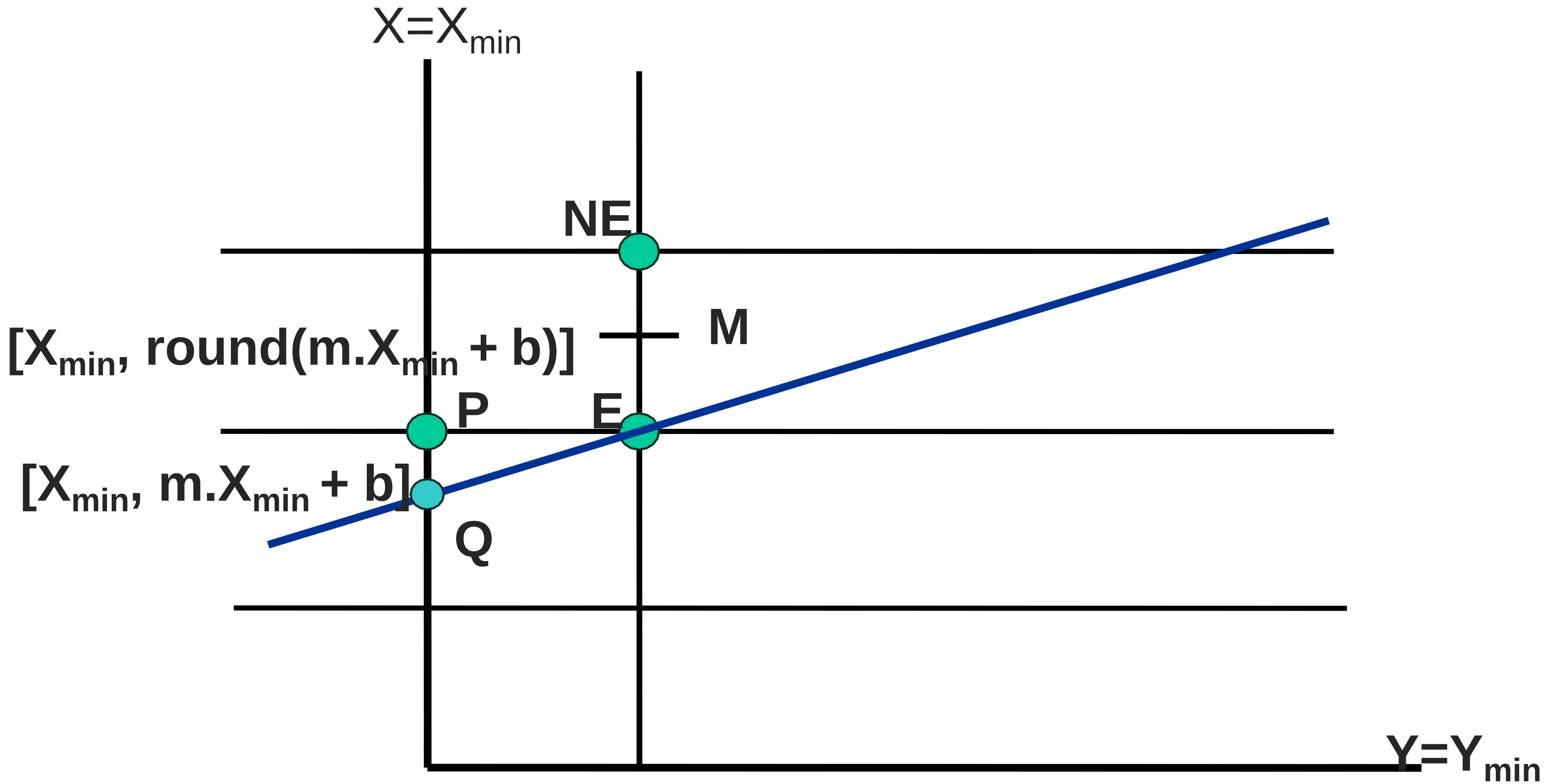
ANTI-ALIASING





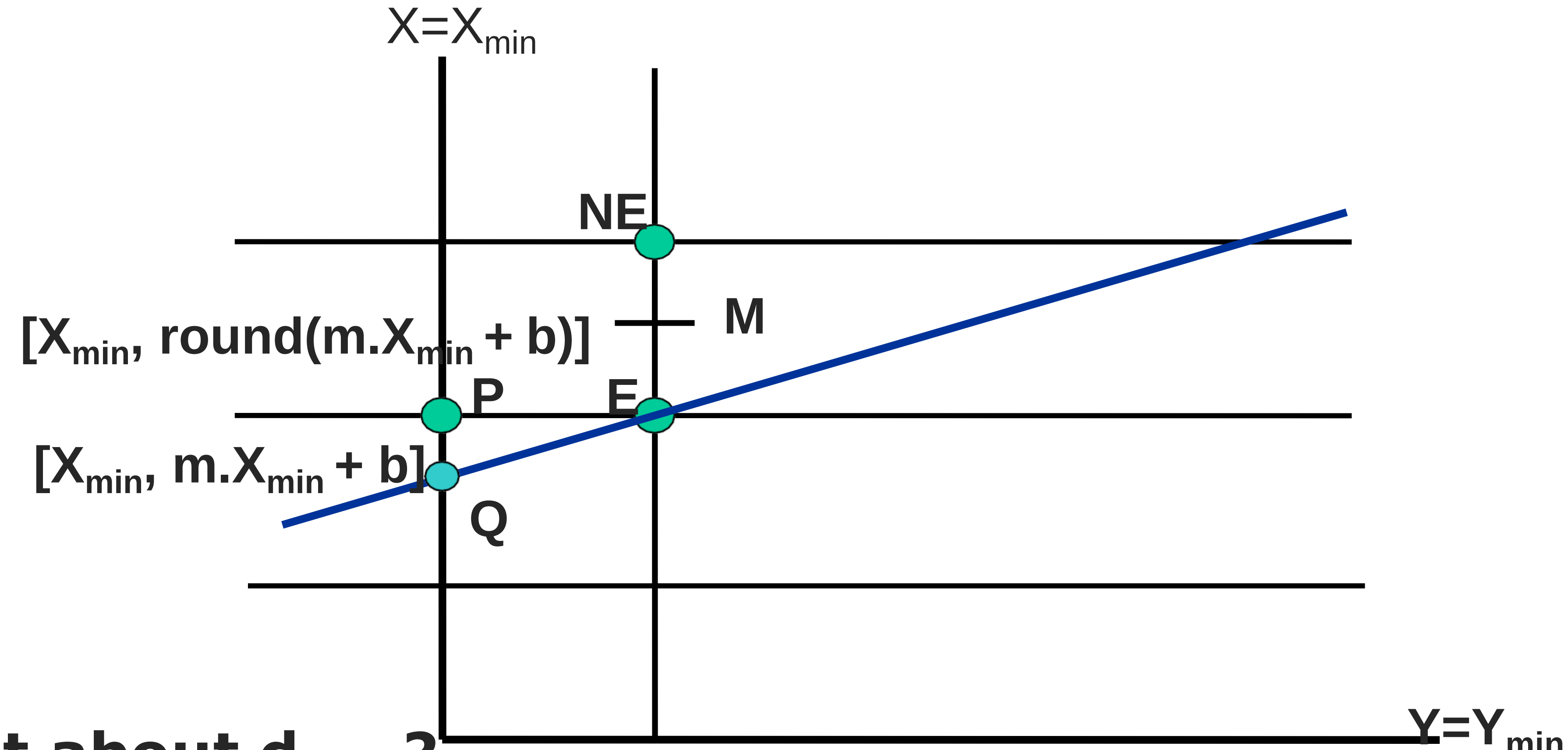


Intersection of a line with a vertical edge of the clip rectangle



No problem in this case to round off the starting point, as that would have been a point selected by mid-point criteria too.

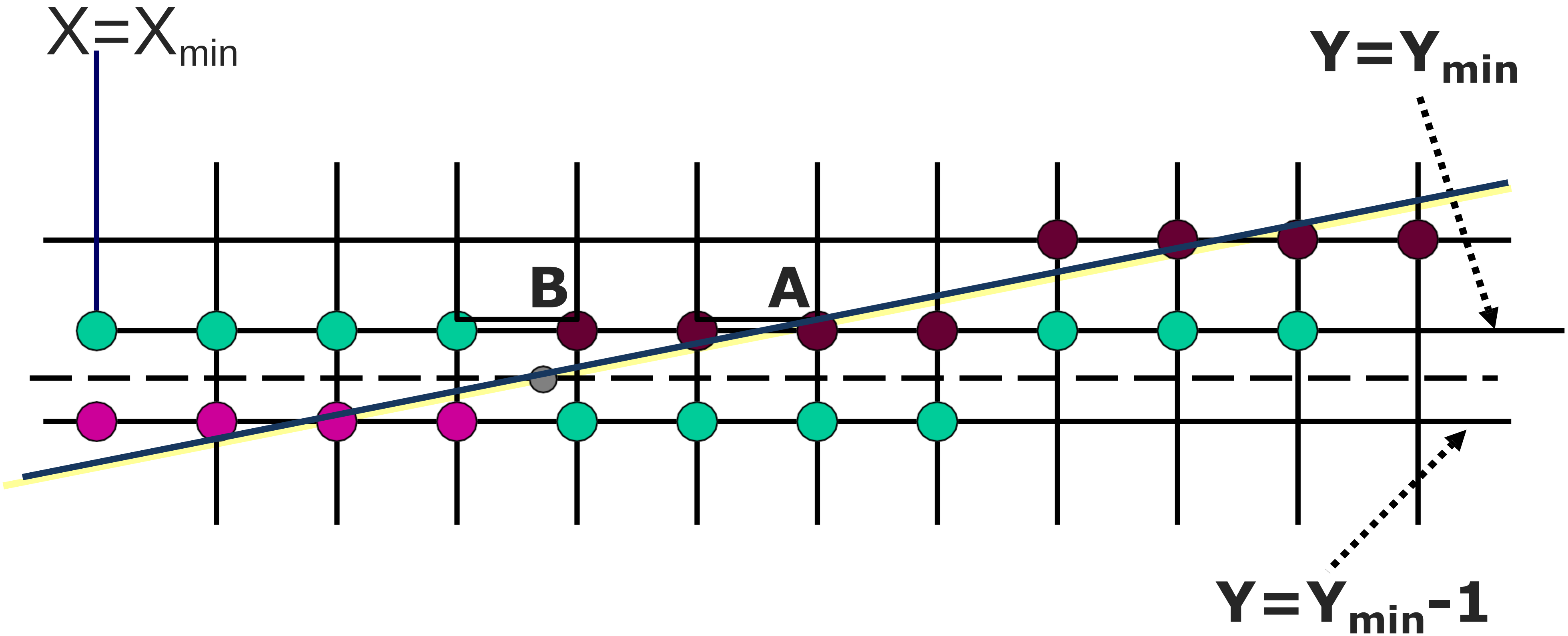
Select P by rounding the intersection point coordinates at Q.



What about d_{start}?

If you initialize the algorithm from P, 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).



Intersection of a shallow line with a horizontal edge of the clip rectangle

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

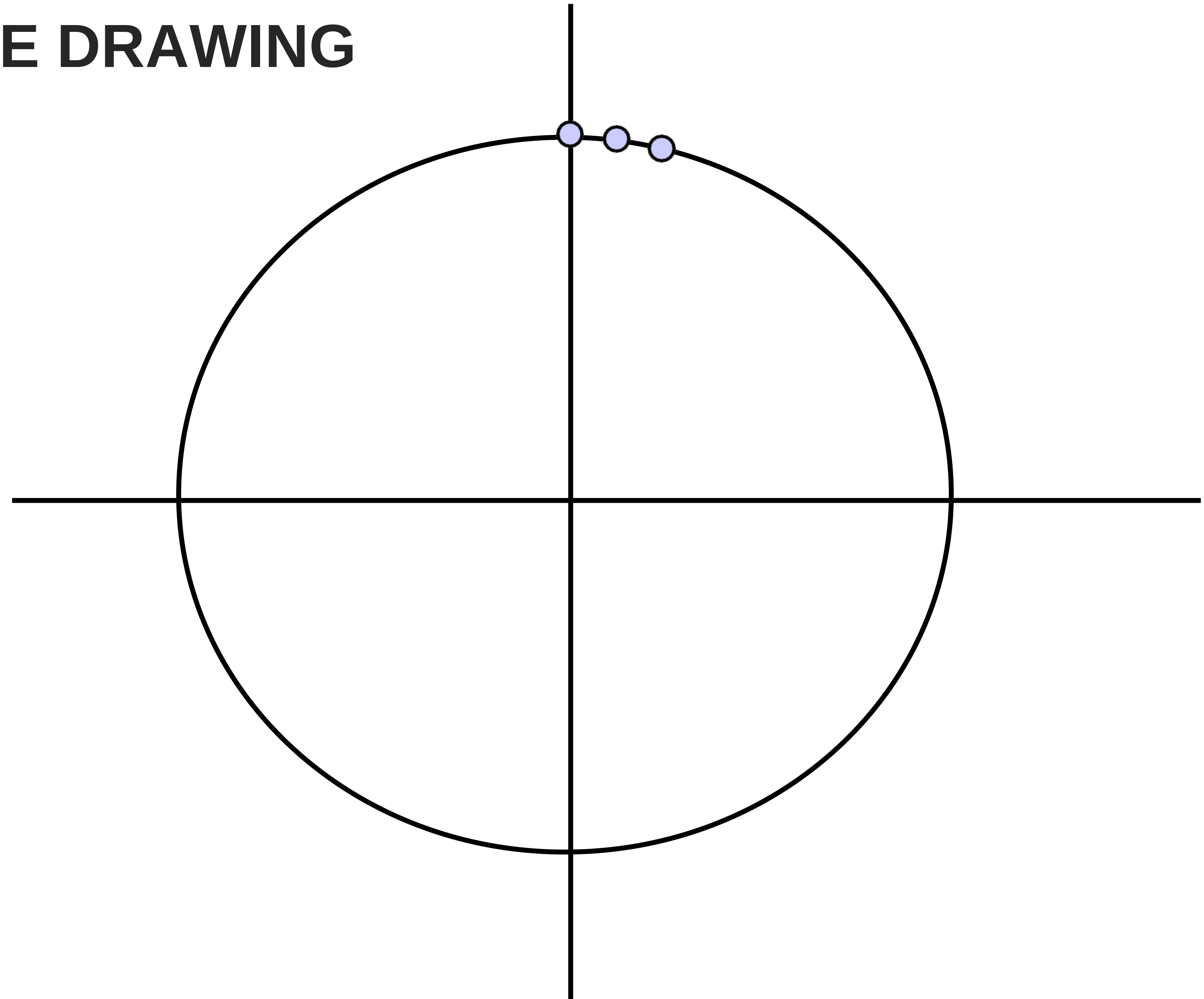
To get B, as a part of the clipped line:

Obtain intersection of line with (Y_{min} - 1/2) and then round off, as

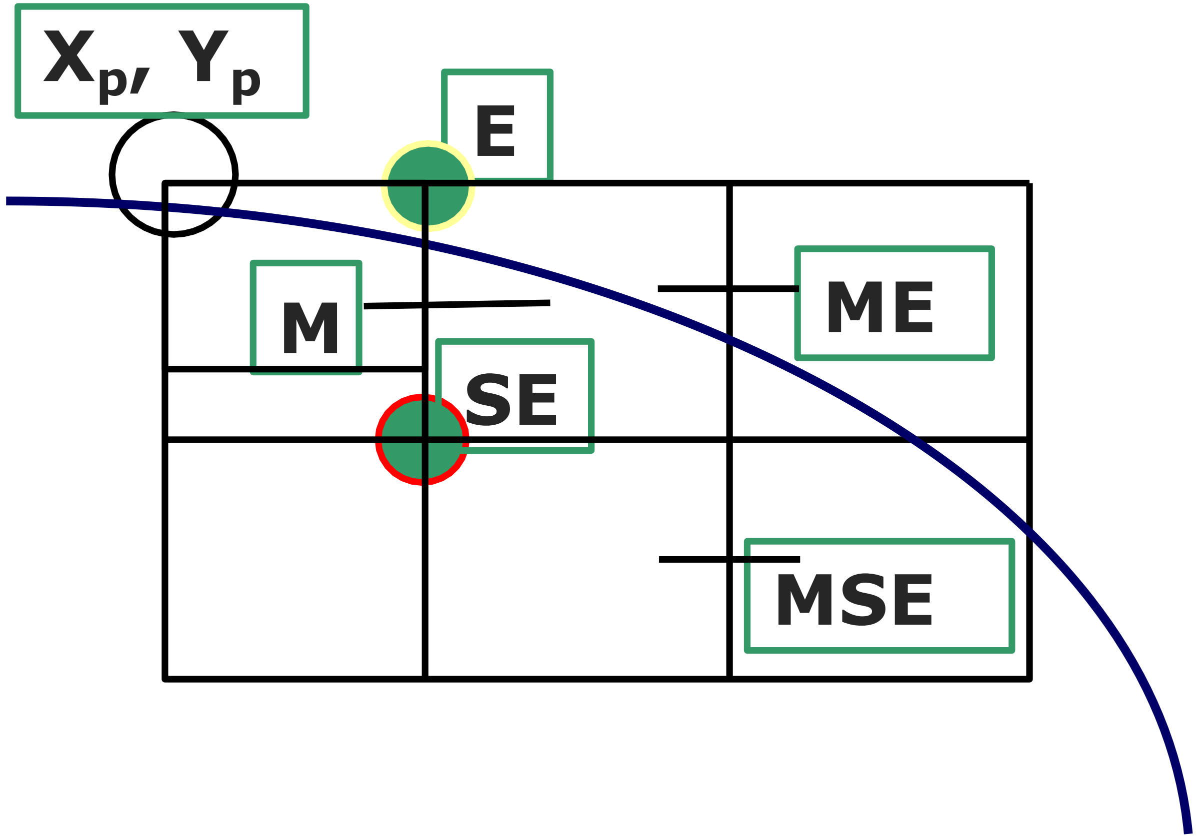
$$\mathbf{B = [round(X|_{Y_{min}-1/2}), Y_{min}]}$$

CIRCLE DRAWING

CIRCLE DRAWING



Assume second octant



Now the choice is between pixels E and SE.

CIRCLE DRAWING

Only considers circles centered at the origin with integer radii.

Can apply translations to get non-origin centered circles.

Explicit equation: $y = +/- \text{sqrt}(R^2 - x^2)$

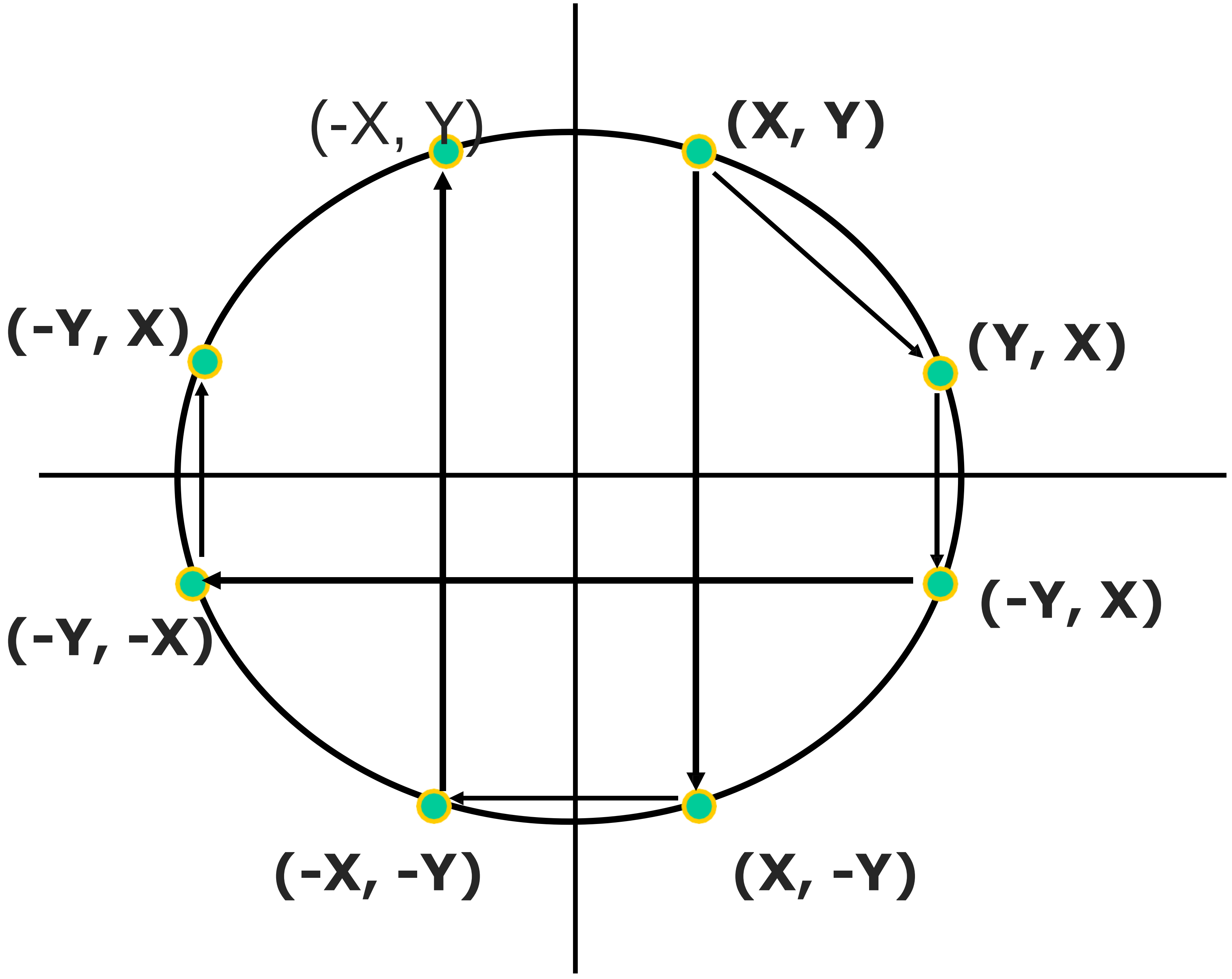
Implicit equation: $F(x,y)= x^2 + y^2 - R^2 =0$

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. One can get points in the other 7 octants as follows:

```
Draw_circle(x, y)

begin
    Plotpoint (x, y);  Plotpoint (y, x);

    Plotpoint (x, -y); Plotpoint (-y, x);

    Plotpoint (-x, -y) ; Plotpoint (-y, -x);

    Plotpoint (-x, y); 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 points for the second octant.

Use *draw_circle* procedure to calculate the rest.

Now the choice is between pixels E 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}.

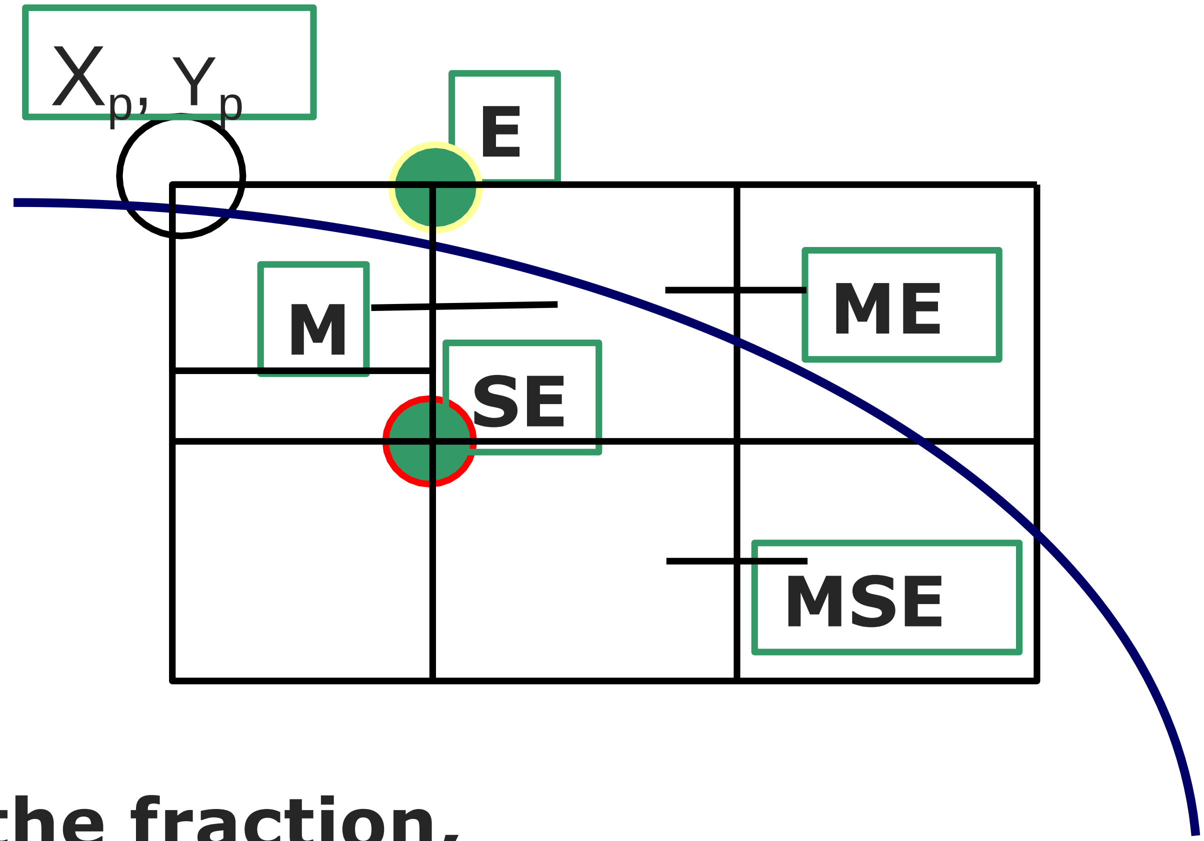
d < 0 choose E ;   next midpoint: M_{new};

Increment + 1 in X;   which gives = d_{new}.

$$
\begin{aligned}
(\Delta d)_{SE} &= d_{new} - d_{old} \\
&= F(X_p + 2, Y_p - 3/2) - F(X_p + 1, Y_p - 1/2) \\
&= 2X_p - 2Y_p + 5 \ ;
\end{aligned}
$$

$$
\begin{aligned}
(\Delta d)_{E} &= d_{new} - d_{old} \\
&= F(X_p + 2, Y_p - 1/2) - F(X_p + 1, Y_p - 1/2) \\
&= 2X_p + 3;
\end{aligned}
$$

$$
\begin{aligned}
d_{start} &= F(X_0 + 1, Y_0 - 1/2) \quad = F(1, R - 1/2) \\
&= 1 + (R - 1/2)^2 - R^2 = 1 + R^2 - R + 1/4 - R^2 \\
&= 5/4 - R
\end{aligned}
$$



To get rid of the fraction,
Let h = d - ¼ => h_{start} = 1 - R

Comparison is: h < -1/4.

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

The Midpoint Circle algorithm:

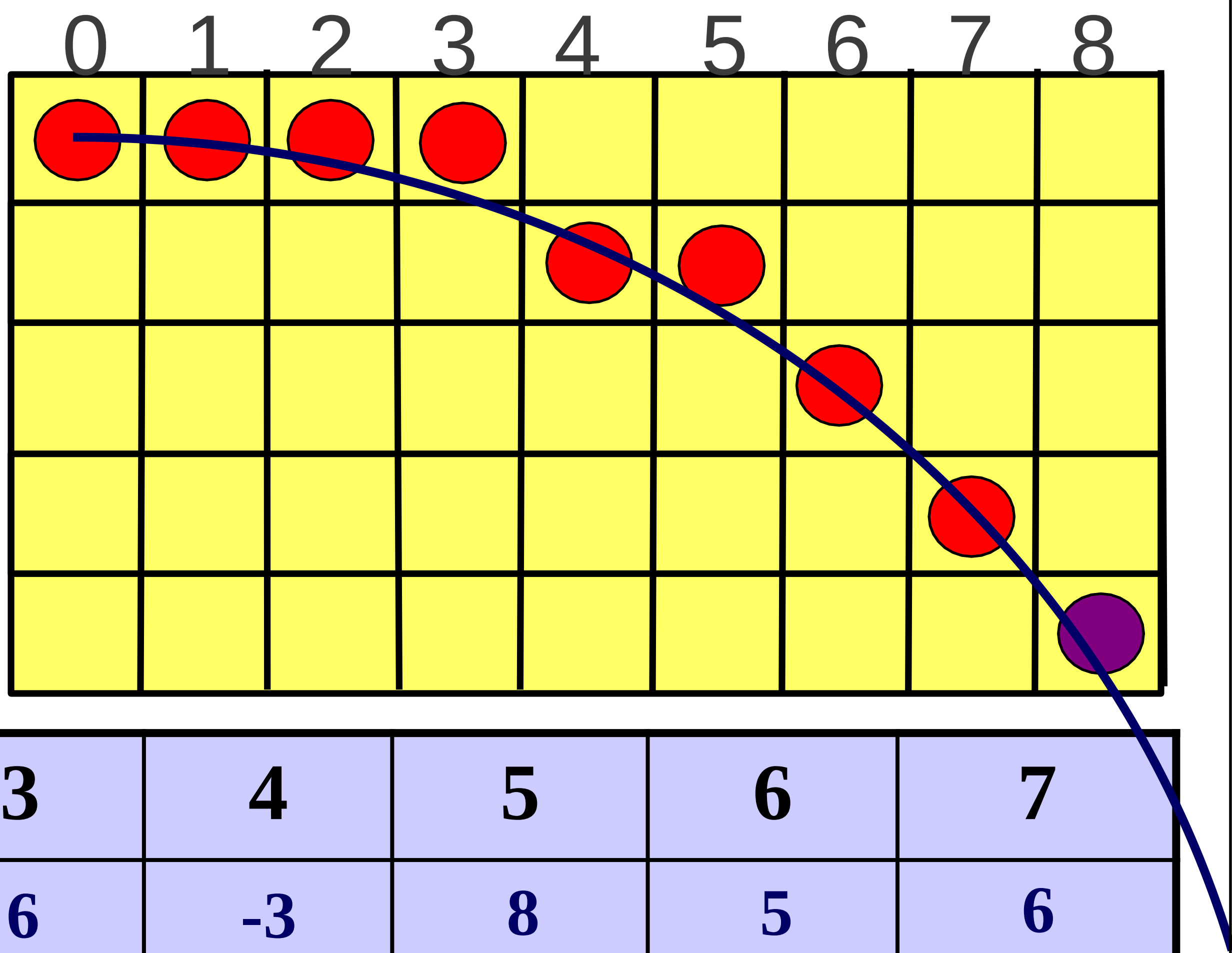
(Version 1)

```
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
```

Example:
R = 10;

Initial Values:
h = 1 – R = -9;
X = 0; Y = 10;
2X = 0;
2Y = 20.



K	1	2	3	4	5	6	7
h	-6	-1	6	-3	8	5	6
2X	0	2	4	6	8	10	12
2Y	20	20	20	20	18	18	16
X, Y	(1, 10)	(2, 10)	(3, 10)	(4, 9)	(5, 9)	(6, 8)	(7, 7)

Problem in this?

Requires atleast 1 multiplication and 3 addition operation per pixel.

Why? $(\Delta d)_E$,$(\Delta d)_{SE}$ are linear and not constant

Solution?

Check for $(\Delta d^2)_E$ and $(\Delta d^2)_{SE}$ are constant

If we choose E, then we calculate $(\Delta d^2)_{E/E}$ and

$(\Delta d^2)_{E/SE}$, . Same if we choose SE, then

calculate $(\Delta d^2)_{SE/E}$ and $(\Delta d^2)_{SE/SE}$.

If we choose E ;

go from (X_p , Y_p) to $(X_p + 1, Y_p)$

$$(\Delta d)_{E\text{-}old} = 2X_p + 3$$

$$(\Delta d)_{E\text{-}new} = 2X_p + 5$$

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

$$(\Delta d)_{SE\text{-}old} = 2X_p - 2Y_p + 5$$

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

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

If we choose SE ;

go from (X_p , Y_p) to $(X_p + 1, Y_p - 1)$

$$(\Delta d)_{E\text{-}old} = 2X_p + 3$$

$$(\Delta d)_{E\text{-}new} = 2X_p + 5$$

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

$$(\Delta d)_{SE\text{-}old} = 2X_p - 2Y_p + 5$$

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

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

What about $(\Delta d)_{E\text{-start}}$, $(\Delta d)_{SE\text{-start}}$?

$$(\Delta d)_{E\text{-start}} = 2X_p + 3$$

$$(\Delta d)_{E\text{-start}} = 2(0) + 3=3;$$

$$(\Delta d)_{SE\text{-start}} = 2X_p - 2Y_p + 5$$

$$(\Delta d)_{SE\text{-start}} = 2\,(0) - 2(R\ ) + 5$$

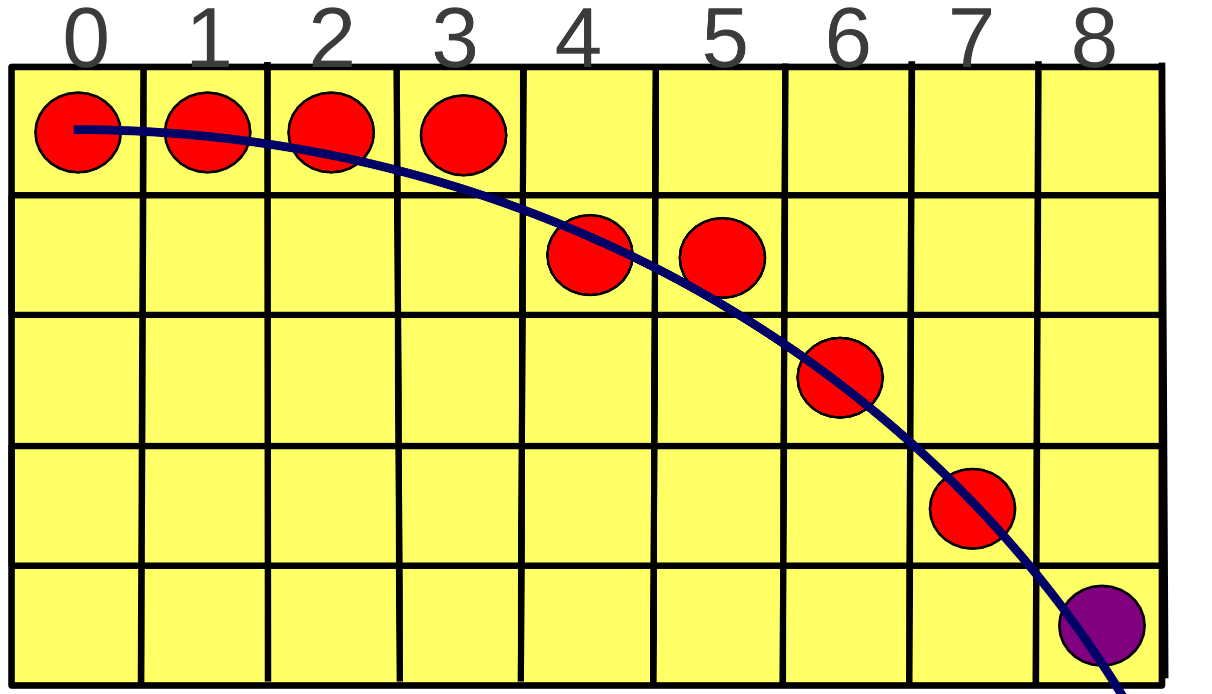
The Midpoint Circle algorithm:

(Version **2**)

```
x = 0;  y = R;
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
```

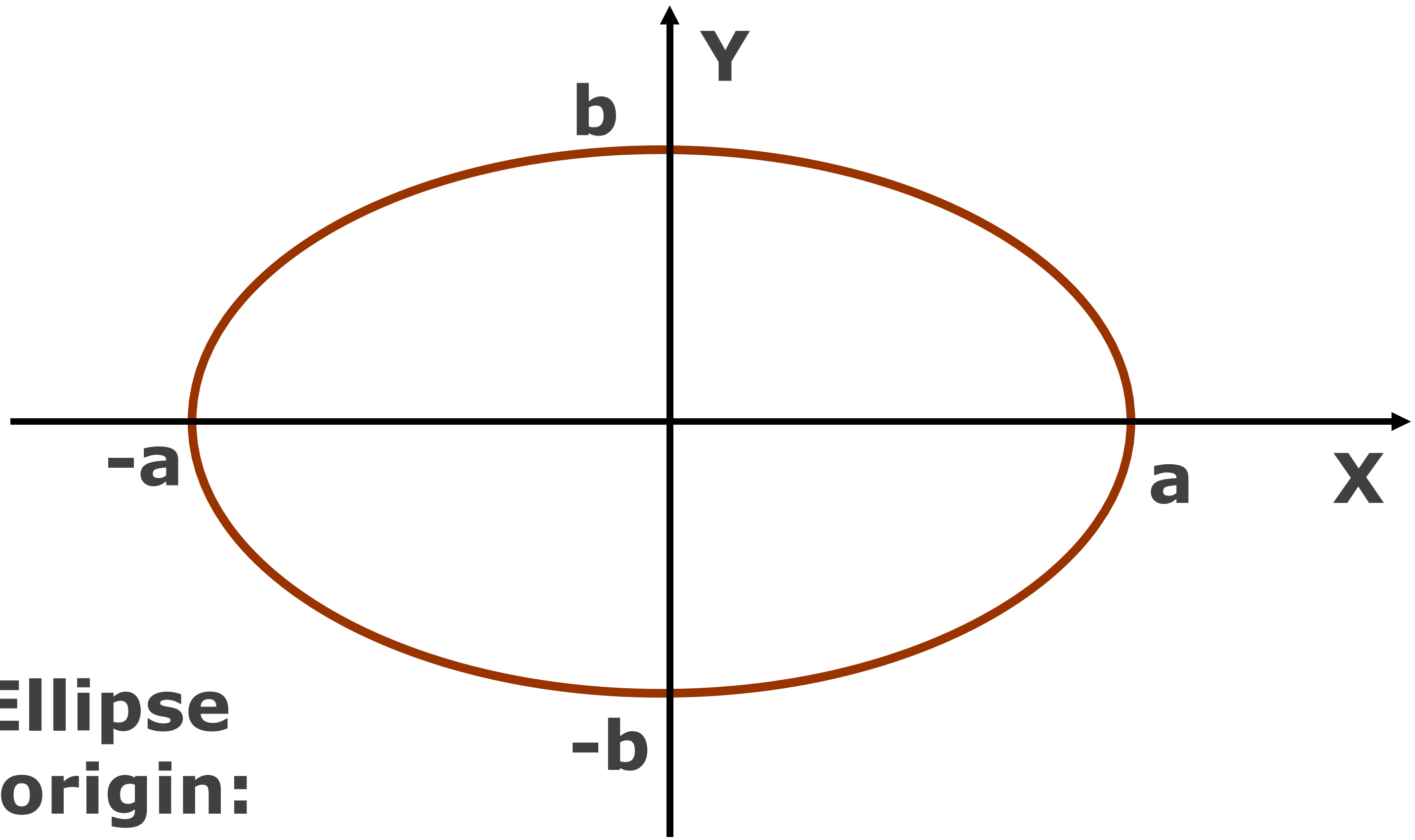
Example:
R = 10;
Initial Values:
h = 1 – R = -9;
X = 0; Y = 10;
Δ_E = 3;
Δ_{SE} = -15.



K	1	2	3	4	5	6	7
h	-6	-1	6	-3	8	5	6
Δ_E	5	7	9	11	13	15	17
Δ_{SE}	-13	-11	-9	-5	-3	1	5
X, Y	(1, 10)	(2, 10)	(3, 10)	(4, 9)	(5, 9)	(6, 8)	(7, 7)

ELLIPSE DRAWING

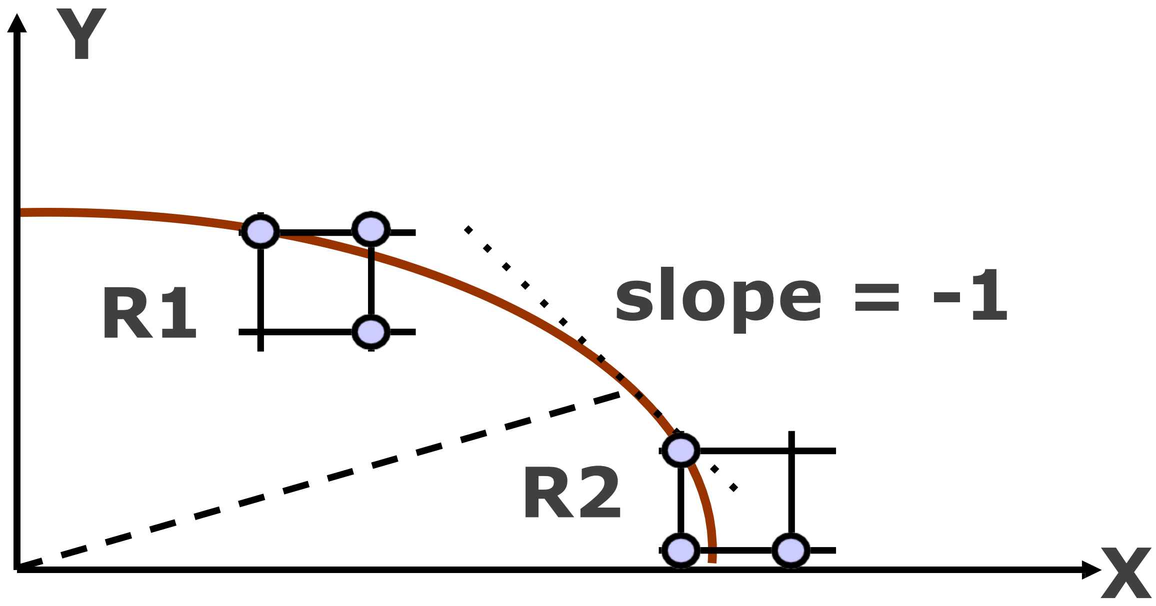
SCAN CONVERTING ELLIPSES



Equation of Ellipse centered at origin:

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

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.

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

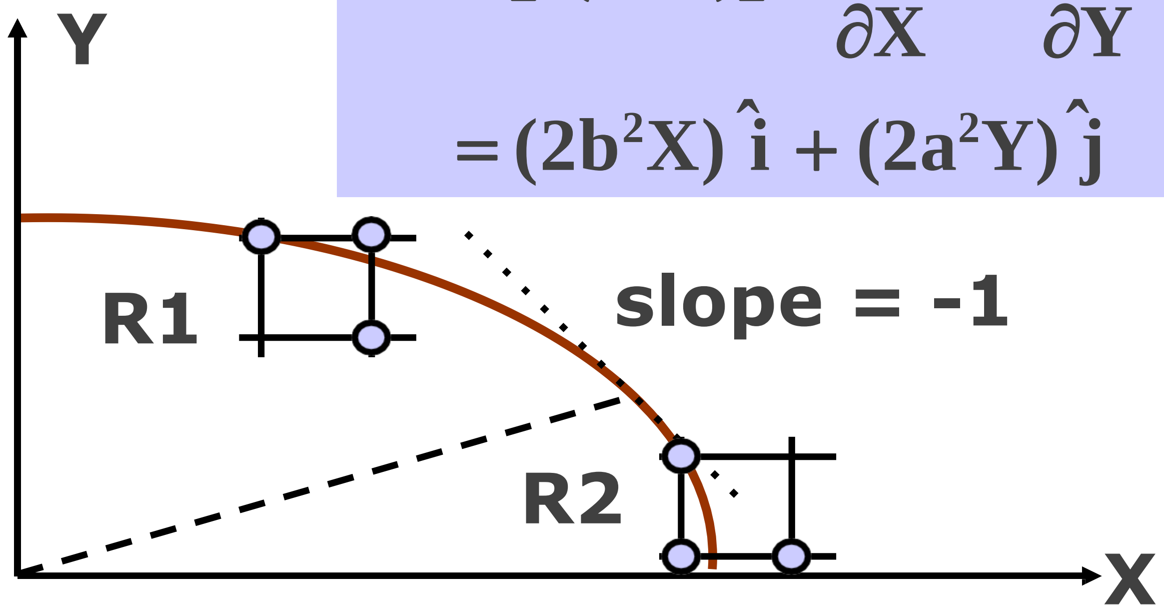
$$\operatorname{grad}\left[f(X,Y)\right] = \frac{\partial f}{\partial X}\hat{i} + \frac{\partial f}{\partial Y}\hat{j}$$

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

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

and

in 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 :

$$b^2 (Xp + 1) \geq a^2 (Yp - 1/2)$$

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

Analysis in region R1:

Let the current pixel be (X_p, Y_p);

$d_{old} = F(M_1)$;

$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$

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

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).

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

Problem with a fractional (floating point) value for $(d_{init})_{R1}$?

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).

$F(M_2) = F(X_k + 1/2, Y_k - 1)$

$= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2$

$= (d_{init})_{R2}$

$$F(M_2) = d_{old} = F(X_k + 1/2, Y_k - 1)$$

$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2$$

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 $Y_k = 0$;

```
void MidPointEllipse (int a, int b, int value);
{
        double d2; int X = 0; int Y = b;
        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)             /*Select E */
                        d1 += sb*((X<<1) + 3);
                  else                   /*Select SE */
                        { 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)          /*Select SE */
                         { d2 += sb*((X<<1) + 2) +
                                   sa*(-(Y<<1) + 3);
                                   X++; }

                  else                  /*Select S */
                         d2 += sa*(-(Y<<1) + 3);

               Y-- ; EllipsePoints(X, Y, value);
        }
}
```