CSE4203: Computer Graphics
Chapter – 8 (part - A)
Graphics Pipeline

Mohammad Imrul Jubair

Outline

– Rasterization

– The Graphics Pipeline

– Line Drawing Algorithm

Rasterization (1/2)

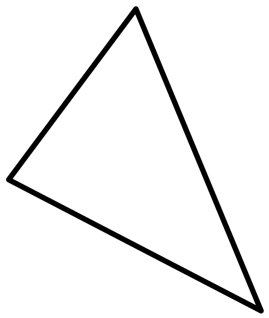
- The previous several chapters have established the mathematical skeleton for object-order rendering.
  - drawing objects one by one onto the screen
- Each geometric object is considered in turn and find the pixels that it could have an effect on.

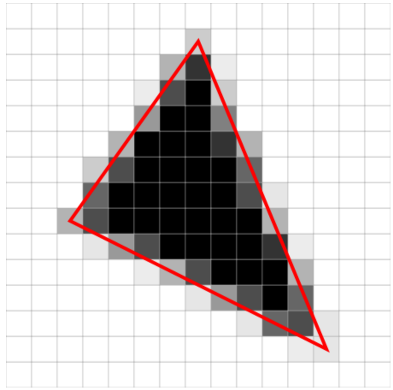
Rasterization (2/2)

- The process of finding all the pixels in an image that are occupied by a geometric primitive is called **rasterization**.

Graphics Pipeline (1/5)

- The sequence of operations that is required, starting with objects and ending by updating pixels in the image, is known as the **graphics pipeline.**





Graphics Pipeline (2/5)

- Two quite different examples of graphics pipelines with very different goals are the
  - hardware pipelines used to support interactive rendering via APIs like OpenGL and Direct3D
  - the software pipelines used in film production, supporting APIs like *RenderMan (by Pixar).*

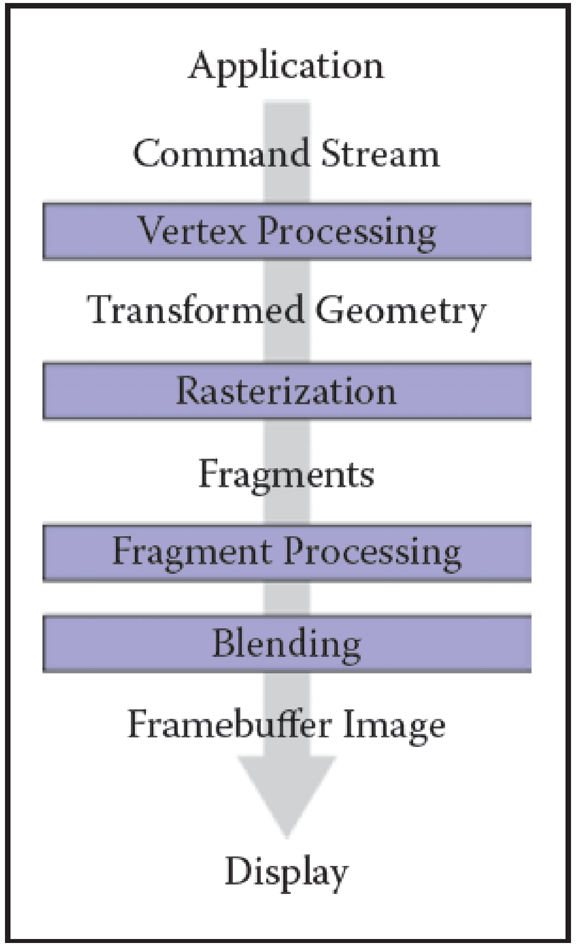
Graphics Pipeline (3/5)

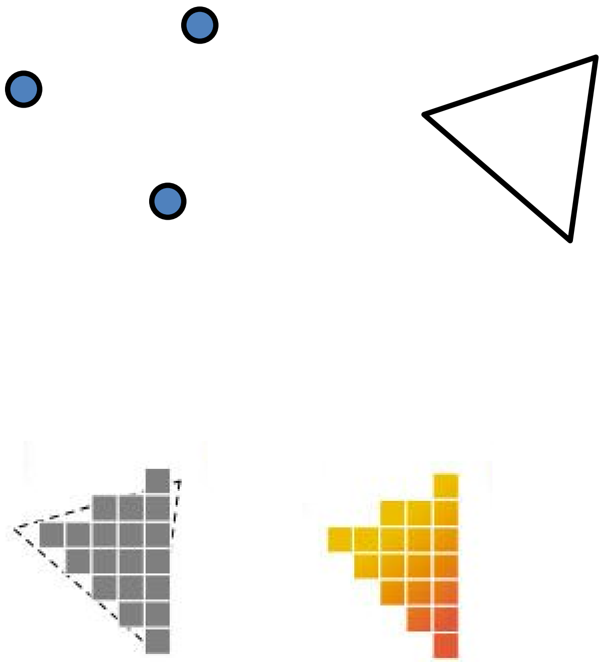
- Hardware pipelines:
  - run fast enough to react in real time for games, visualizations, and user interfaces.
- Software pipelines:
  - render the highest quality animation and visual effects possible and scale to enormous scenes
  - but take much more time to do so

Graphics Pipeline (4/5)

- Remarkable amount is shared among most pipelines
- This chapter attempts to focus on these common fundamentals

Graphics Pipeline (5/5)





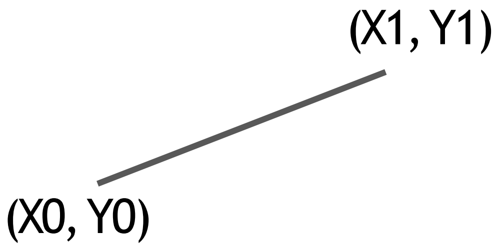
Bresenham’s Line Drawing Algorithm

Scenario (1/2)

Given,

Start point (X0,Y0)

End point (X1,Y1)

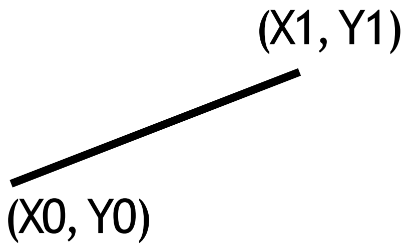


Scenario (2/2)

Given,

Start point (X0,Y0)

End point (X1,Y1)



Assume,

$X1 \geq X0$

And,

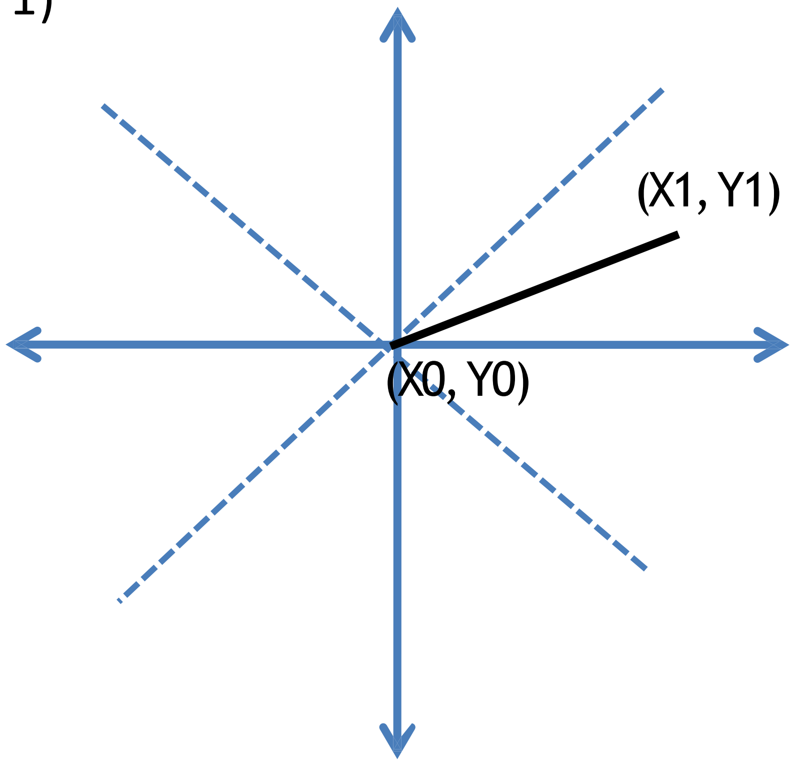
Slope, $m \leq 1$

Scenario (2/2)

Given,

Start point (X0, Y0)

End point (X1, Y1)



Assume,

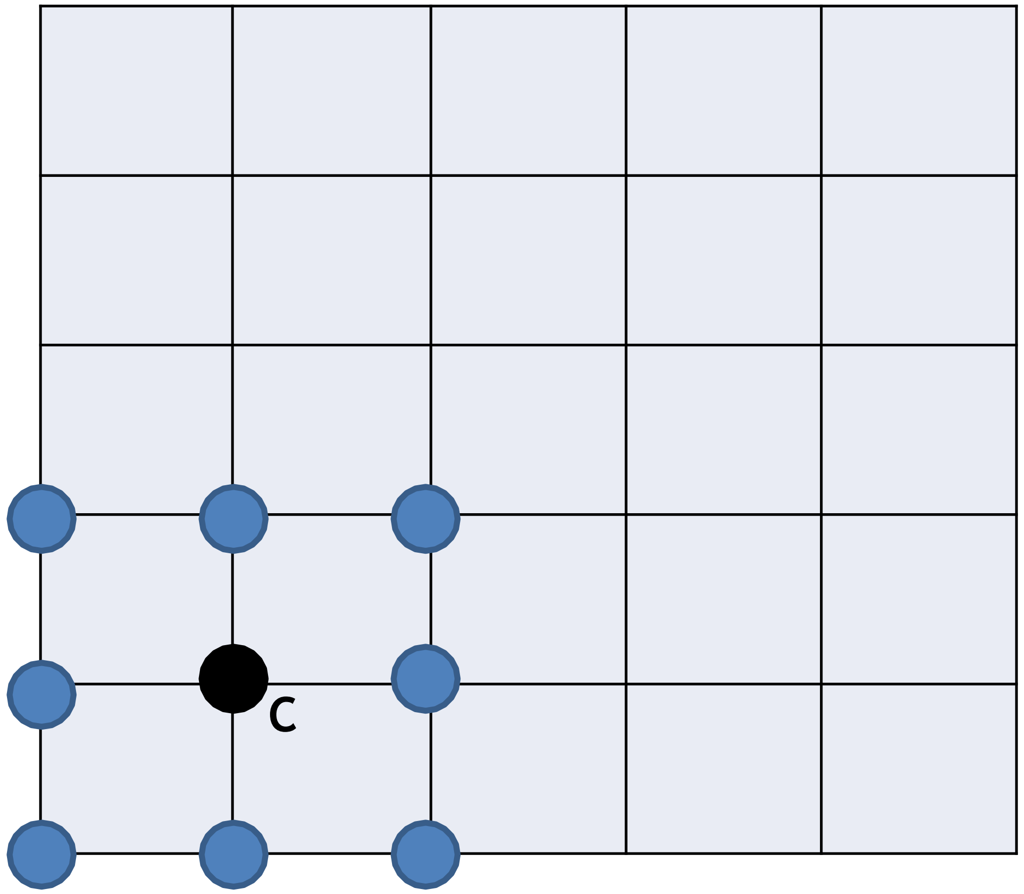
$X1 \geq X0$

And,

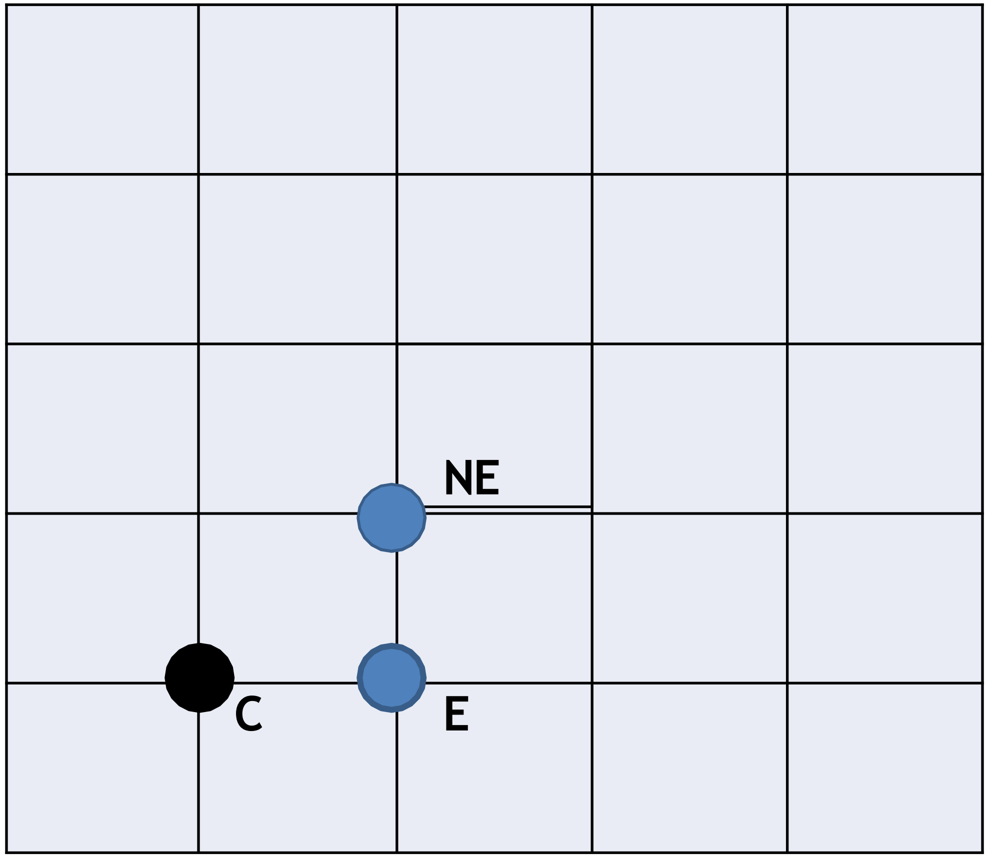
Slope, $m \leq 1$

[in 1st octant]

How it works (1/9)

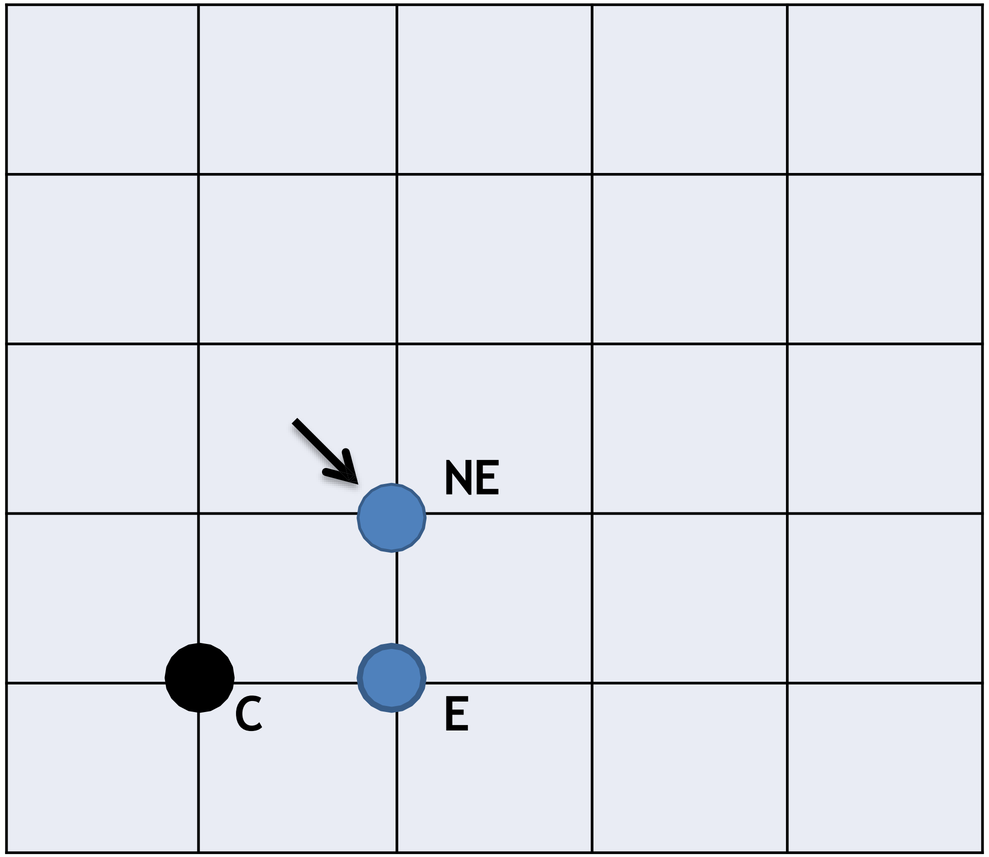


How it works (2/9)



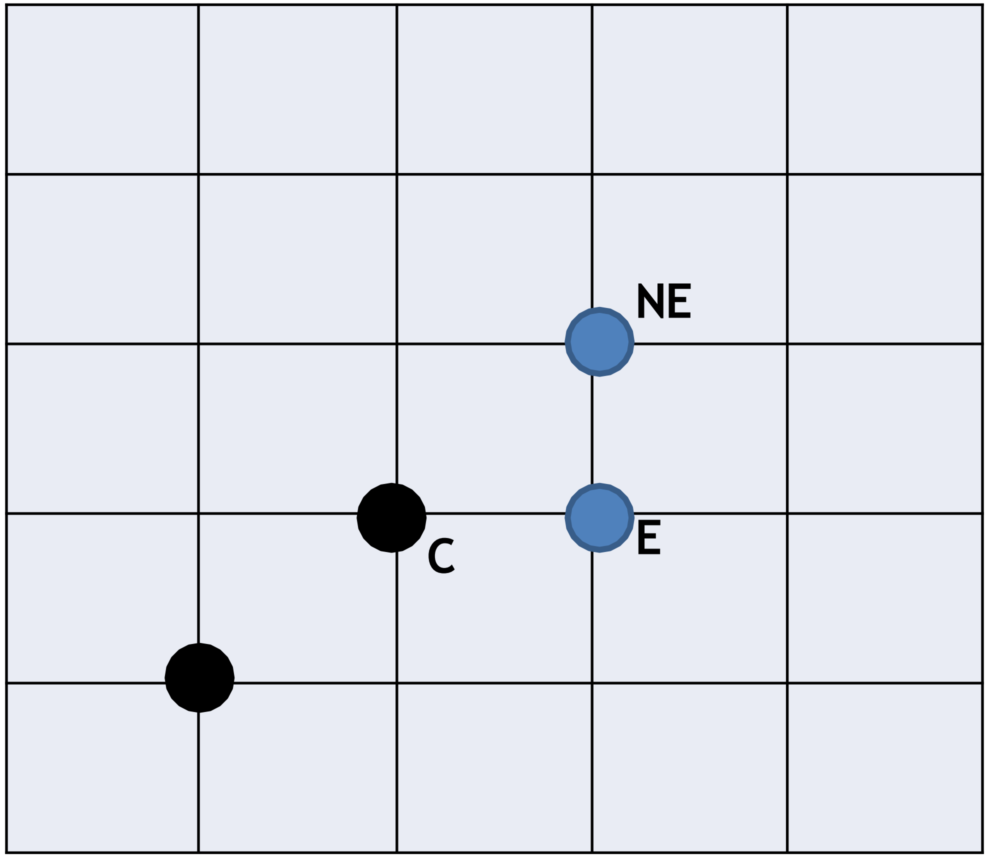
Next pixel is chosen (from E or NE) to build the line successively

How it works (3/9)



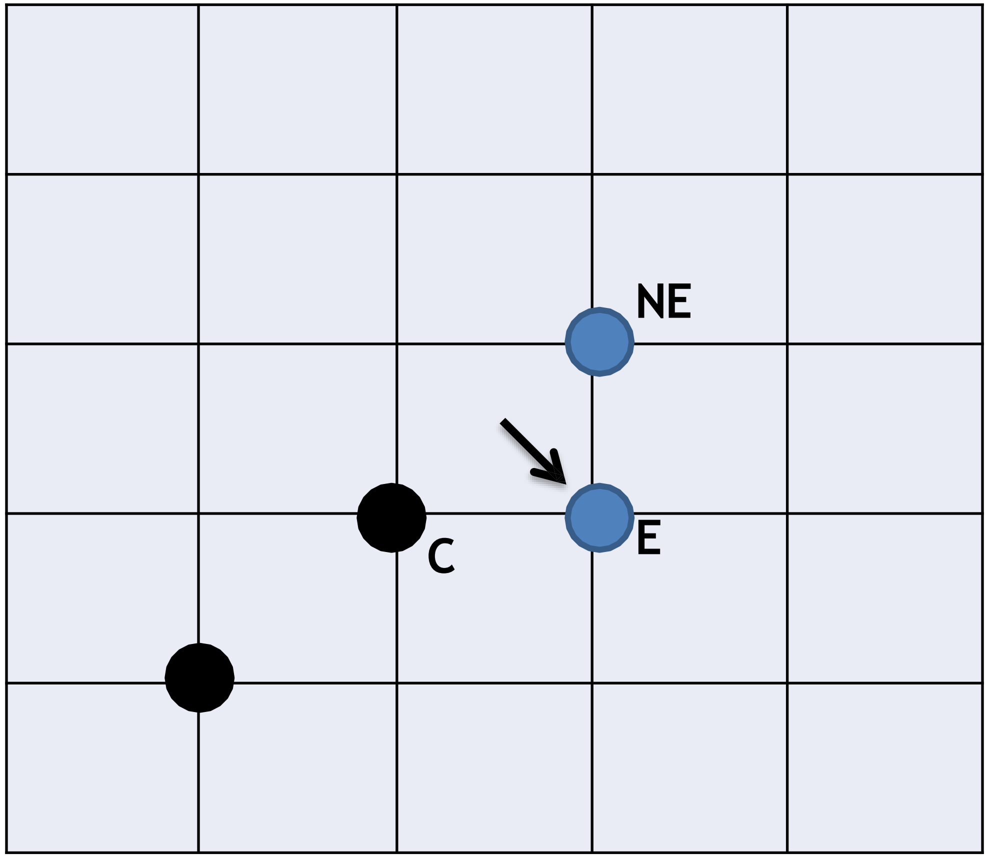
Next pixel is chosen (from E or NE) to build the line successively

How it works (4/9)



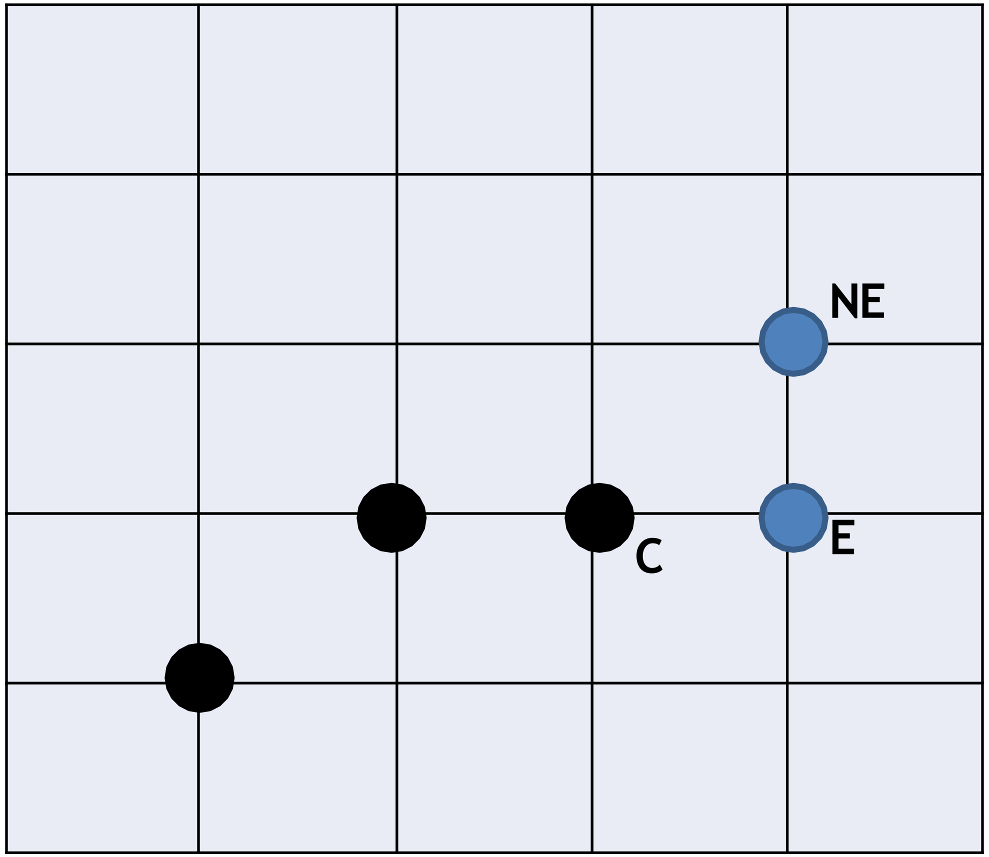
Next pixel is chosen (from E or NE) to build the line successively

How it works (5/9)



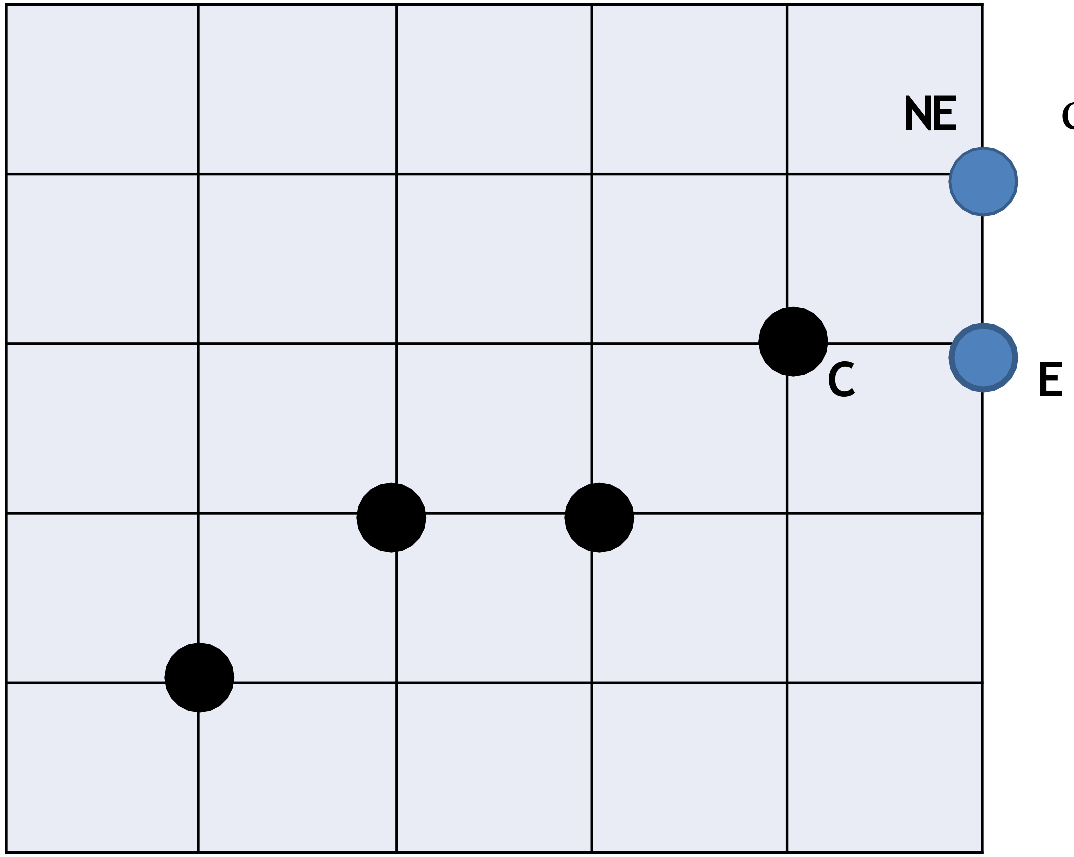
Next pixel is chosen (from E or NE) to build the line successively

How it works (6/9)



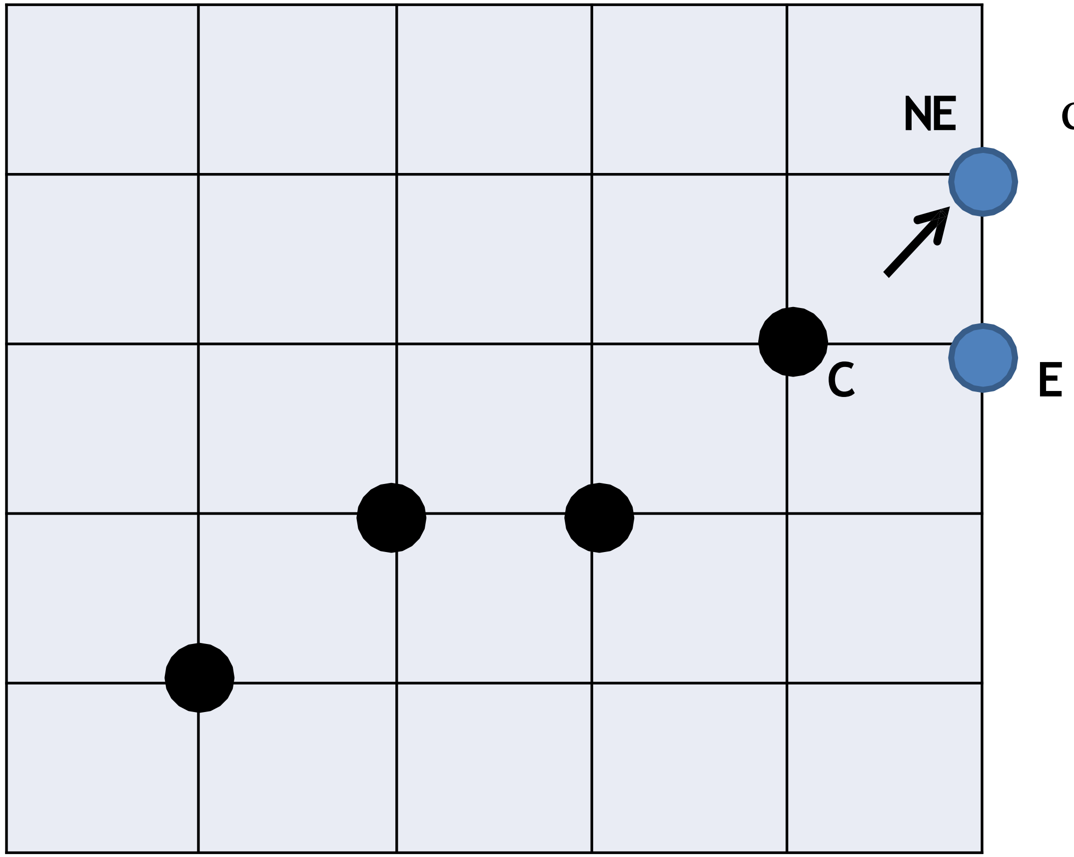
Next pixel is chosen (from E or NE) to build the line successively

How it works (7/9)



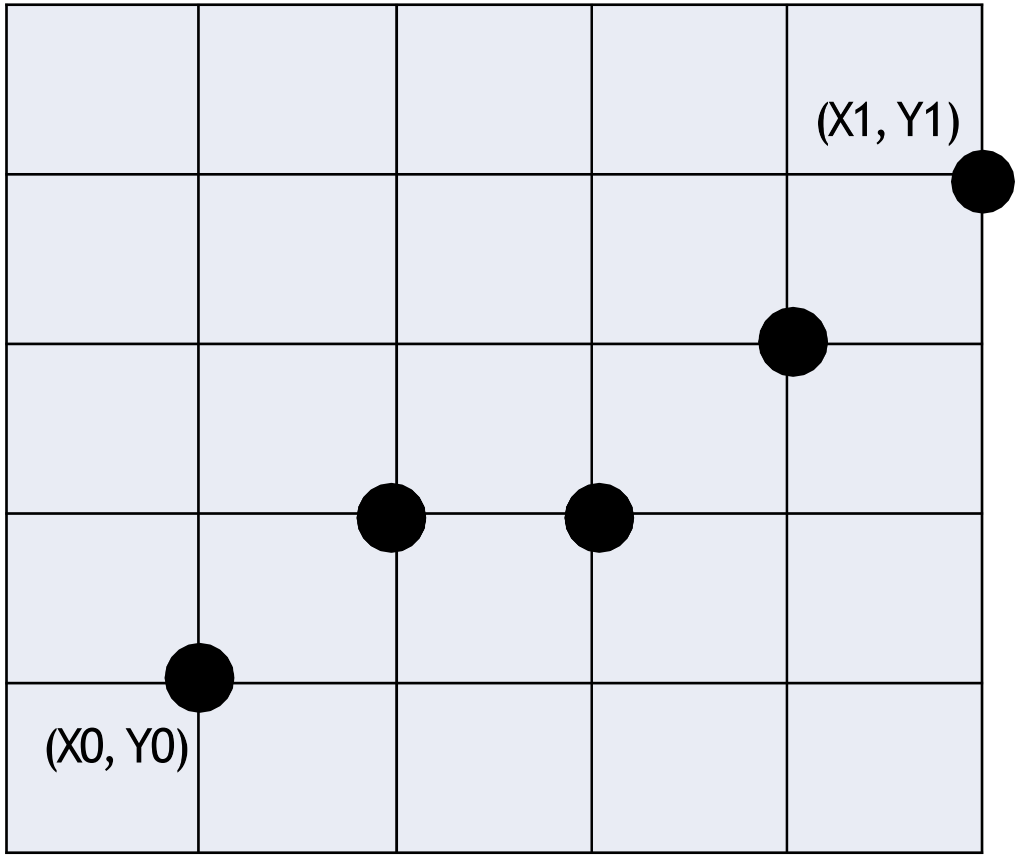
Next pixel is chosen (from E or NE) to build the line successively

How it works (8/9)



Next pixel is chosen (from E or NE) to build the line successively

How it works (9/9)



Next pixel is chosen (from E or NE) to build the line successively

Implicit Equation of a Line (1/5)

$$Y = mX + B$$

$$or, Y = \frac{dy}{dx} * X + B$$

$$or, Ydx = Xdy + Bdx$$

$$or, Xdy - Ydx + Bdx = 0$$

$$or, aX + bY + c = 0 \;\; [here, a = dy, b = -dx\ , c = Bdx]$$

$$F(X, Y) = aX + bY + c = 0$$

Implicit Equation of a Line (2/5)

$$Y = mX + B$$

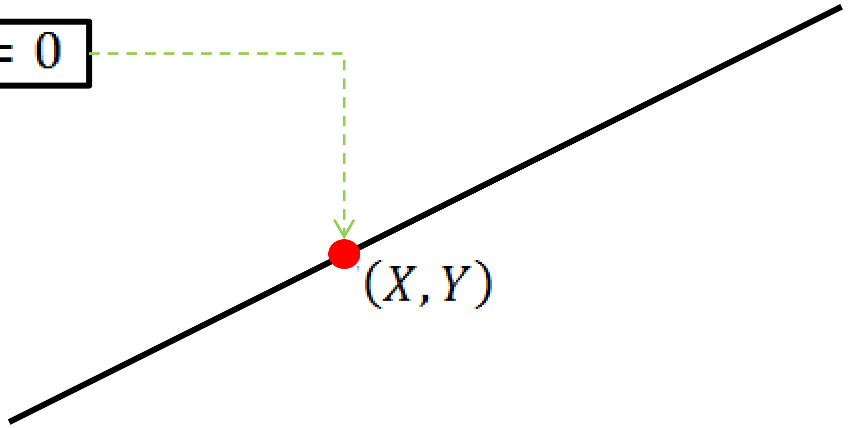
$$or, Y = \frac{dy}{dx} * X + B$$

$$or, Ydx = Xdy + Bdx$$

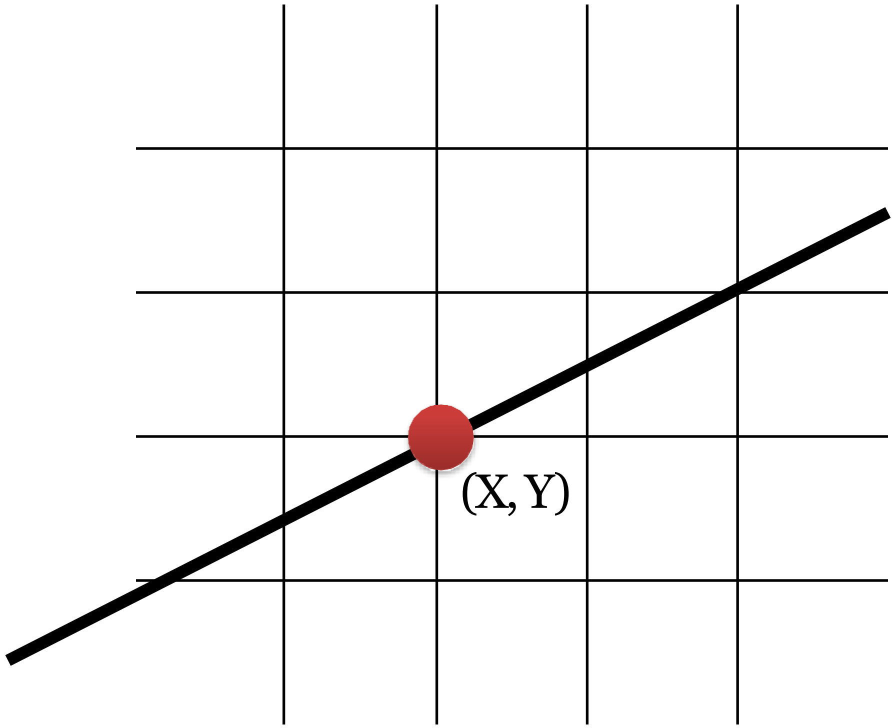
$$or, Xdy - Ydx + Bdx = 0$$

$$or, aX + bY + c = 0 \ \ [here, a = dy, b = -dx, c = Bdx]$$

$$\boxed{F(X,Y) = aX + bY + c = 0}$$

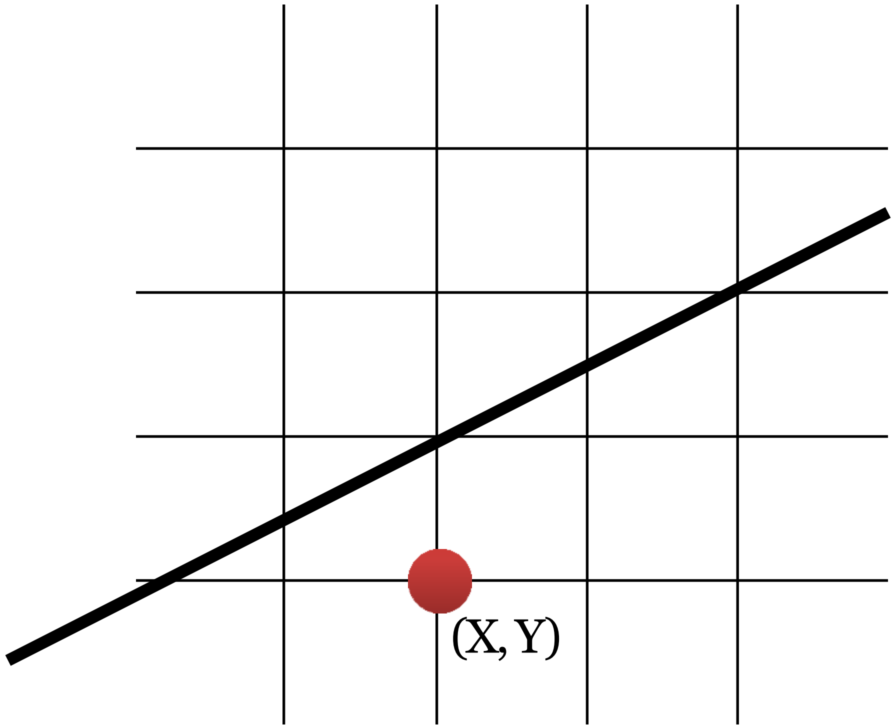


Implicit Equation of a Line (3/5)



If $F(X, Y) = 0$, the point (X,Y) on lying on the line

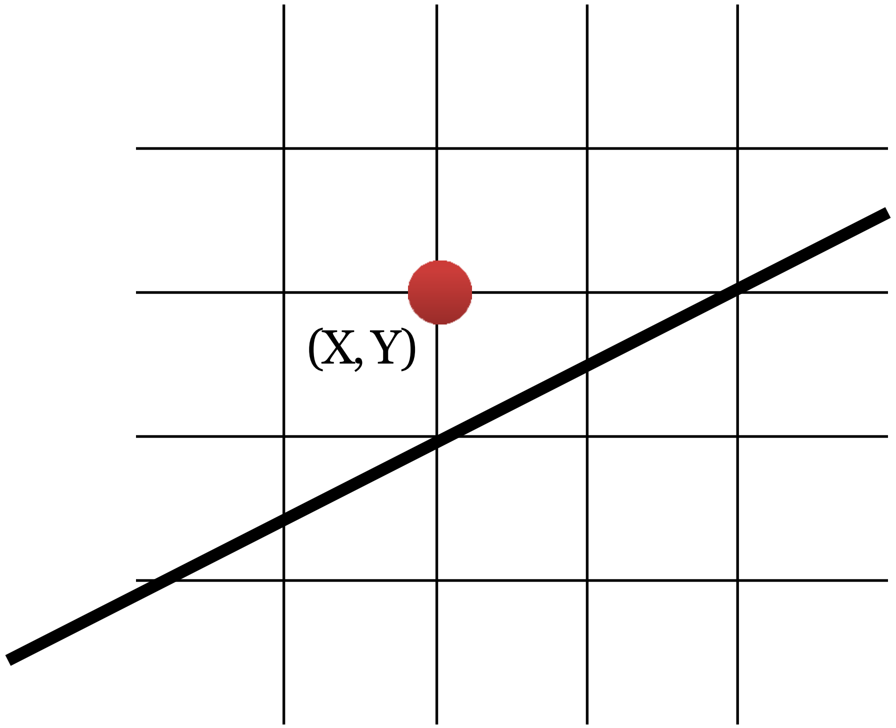
Implicit Equation of a Line (4/5)



If $F(X,Y)=0$, the point (X,Y) on lying on the line

If $F(X,Y) > 0$, the point (X,Y) is under the line

Implicit Equation of a Line (5/5)

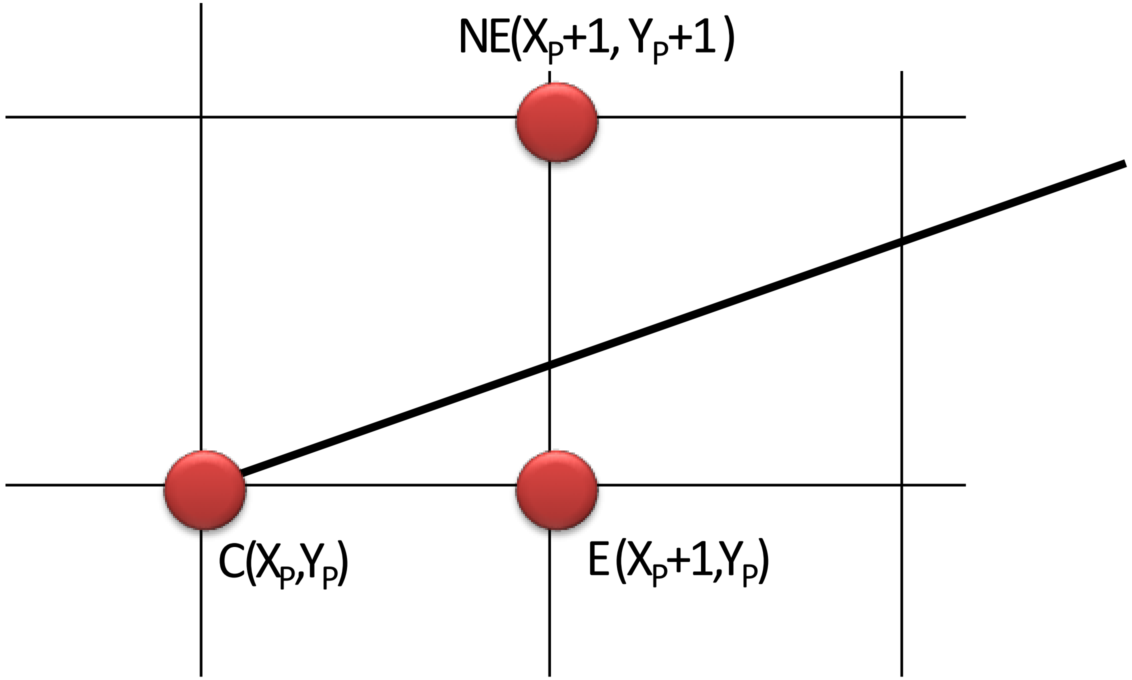


If $F(X,Y) = 0$, the point (X,Y) on lying on the line

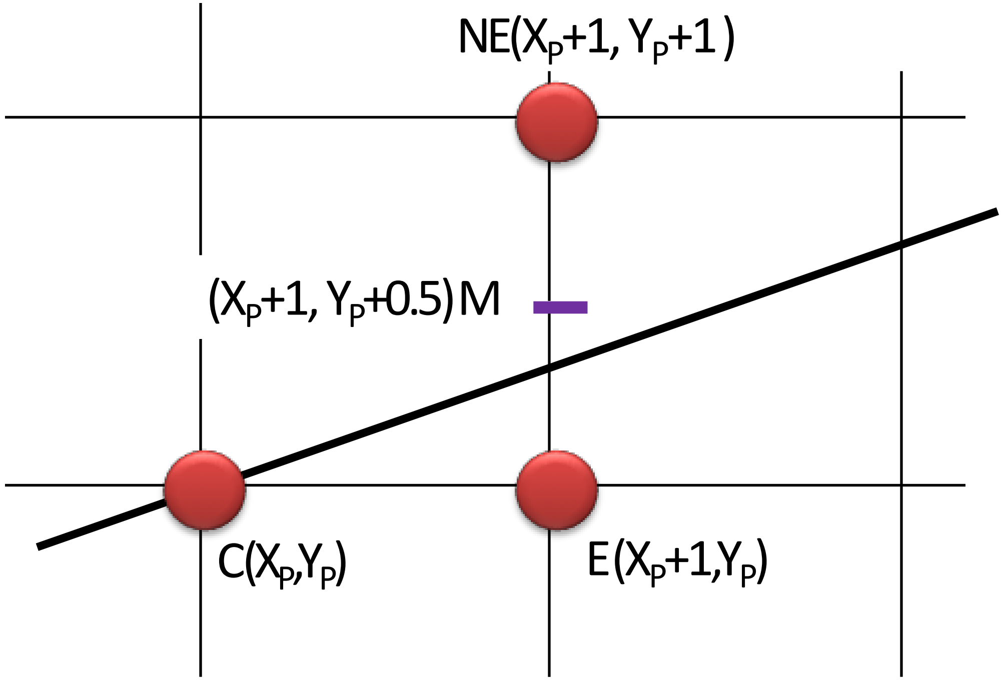
If $F(X,Y) > 0$, the point (X,Y) is under the line

If $F(X,Y) < 0$, the point (X,Y) is above the line

Midpoint Criteria (1/7)



Midpoint Criteria (2/7)



So, **d = F (M)**

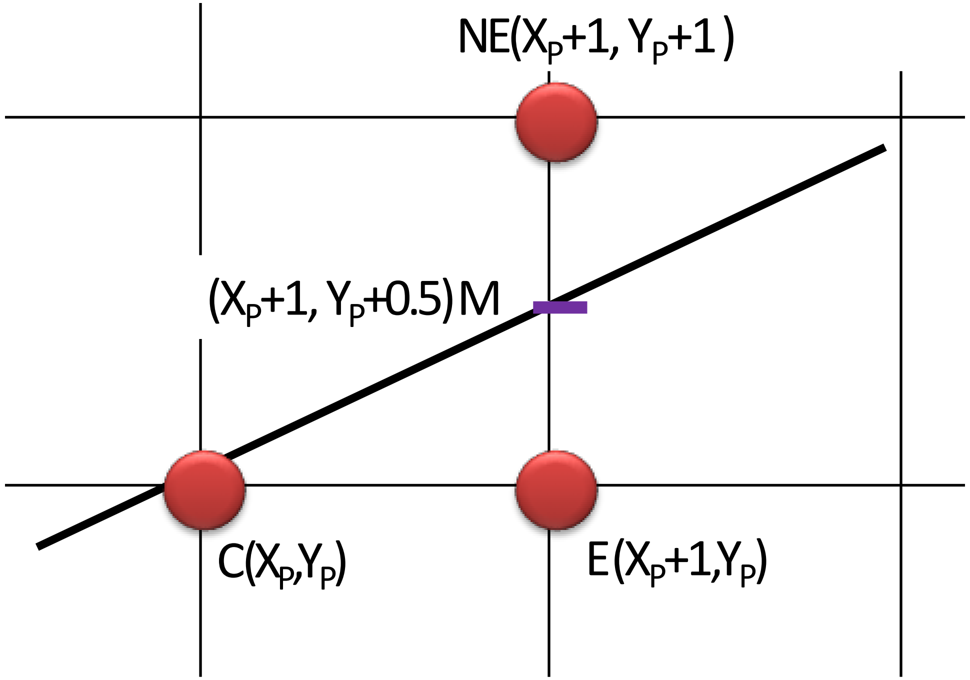
d is called 'decision variable'

So, $d = F(M)$

$= F(X_P+1, Y_P+0.5)$

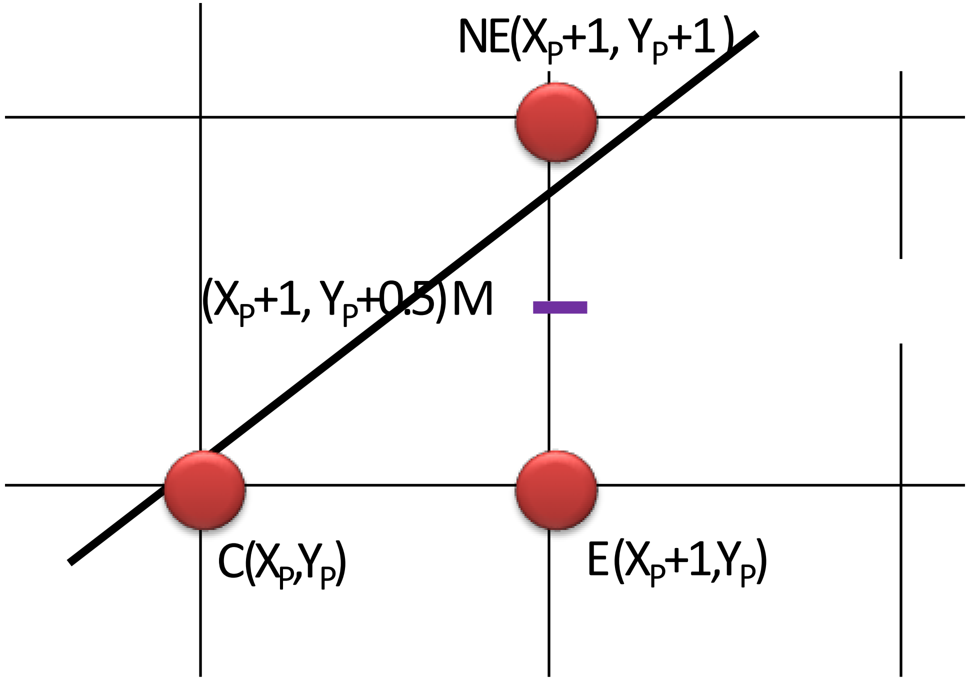
$= a(X_P+1) + b(Y_P+0.5) + c$

Midpoint Criteria (3/7)



if **d = 0**, then midpoint is on the line

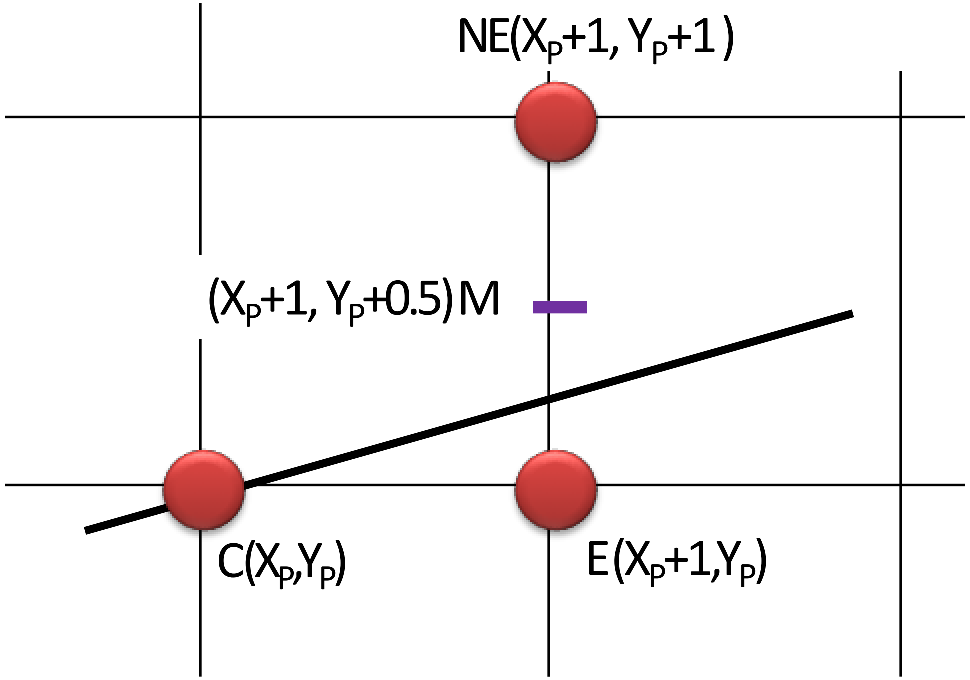
Midpoint Criteria (4/7)



if **d = 0**, then midpoint is on the line

If **d > 0**, then midpoint M is below the line

Midpoint Criteria (5/7)

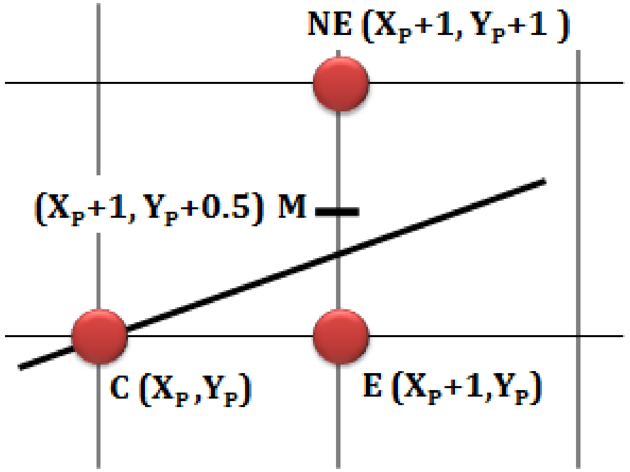


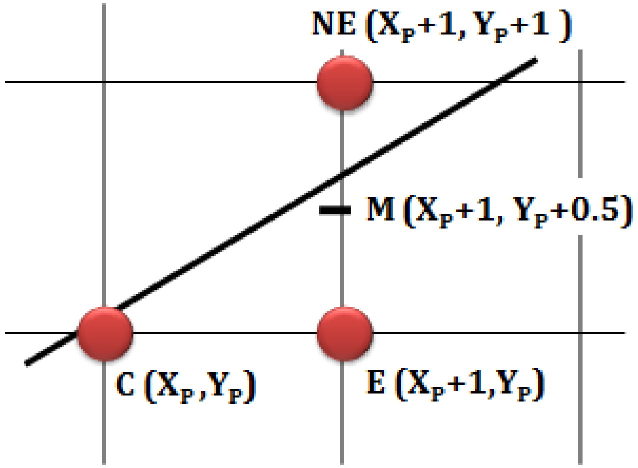
if **d = 0**, then midpoint is on the line

If **d > 0**, then midpoint M is below the line

If **d < 0**, then midpoint M is above the line

Midpoint Criteria (6/7)

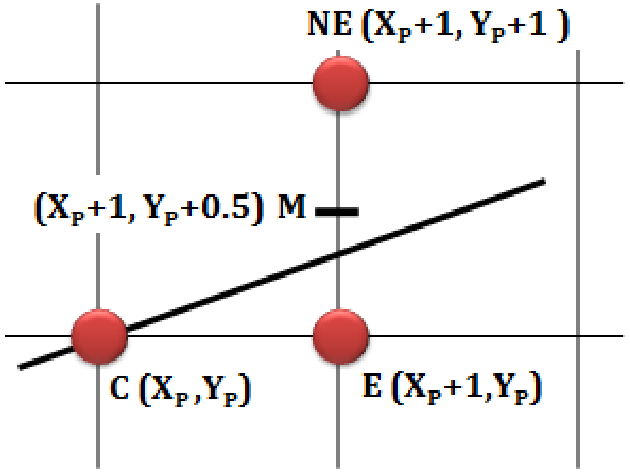




If **d >0**, then midpoint ***M*** is below the line

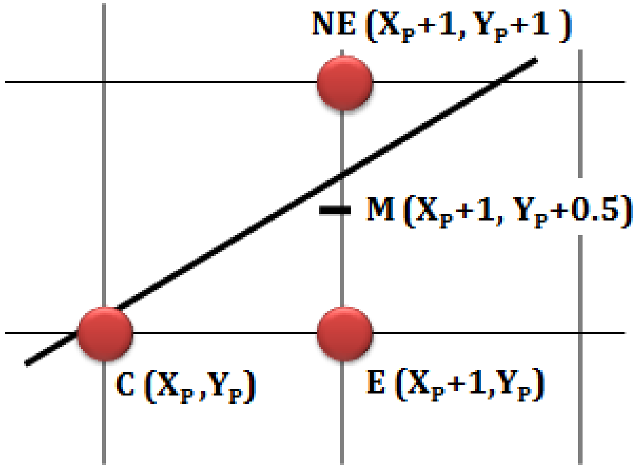
If **d ≤0**, then midpoint ***M*** is above the line

Midpoint Criteria (7/7)



If **d ≤0**, then midpoint M is above the line, and **E** is closer to line,

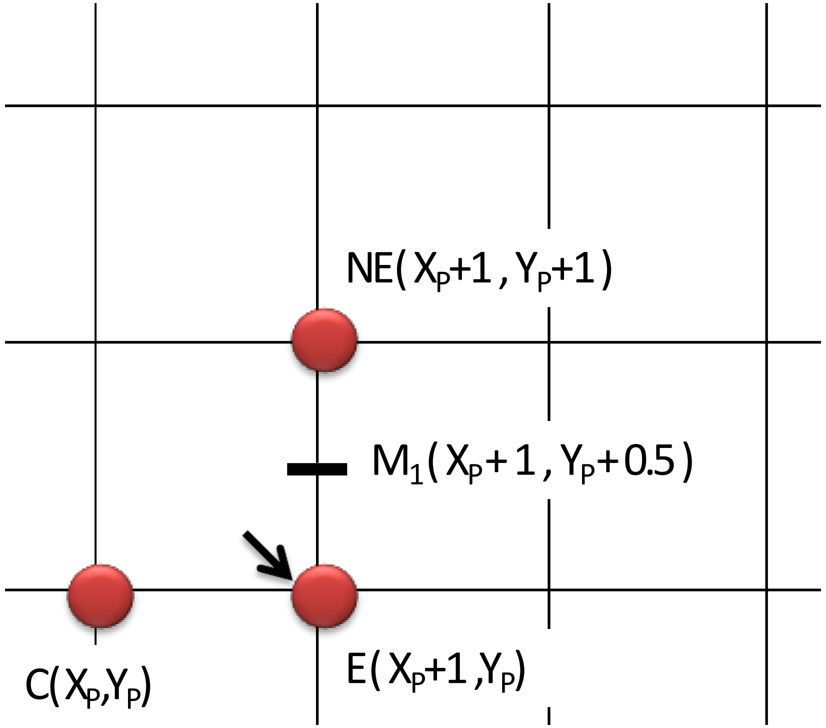
So, **E** is selected



If **d >0**, then midpoint M is below the line, and **NE** is closer to line,

So, **NE** is selected

Successive Updating for E (1/4)

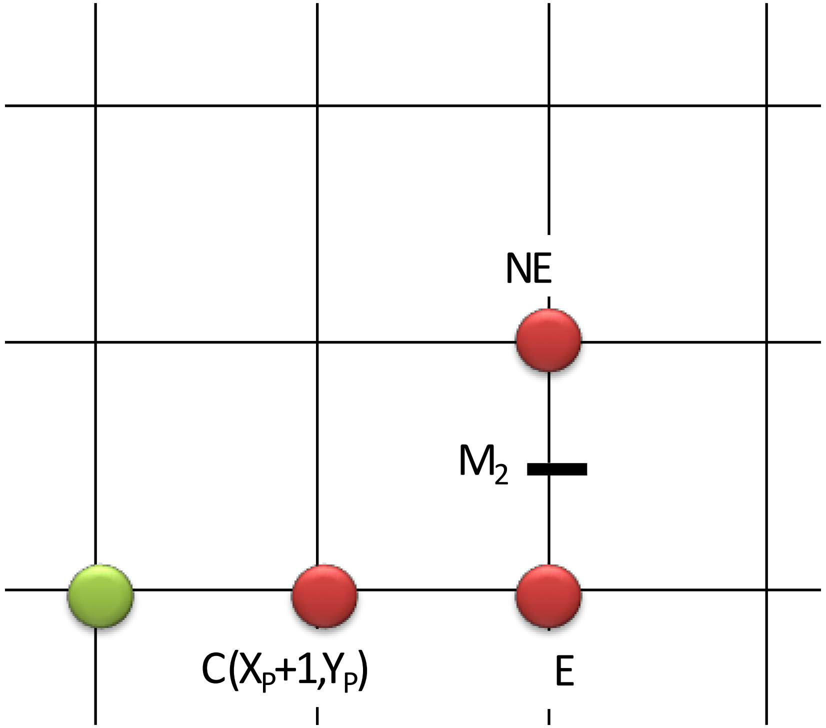


$d_1 = F(M_1)$

$= F(X_P+1,\ Y_P+0.5)$

$= a(X_P+1) + b(Y_P+0.5) + c$

Successive Updating for E (2/4)



$d_1 = F(M_1)$

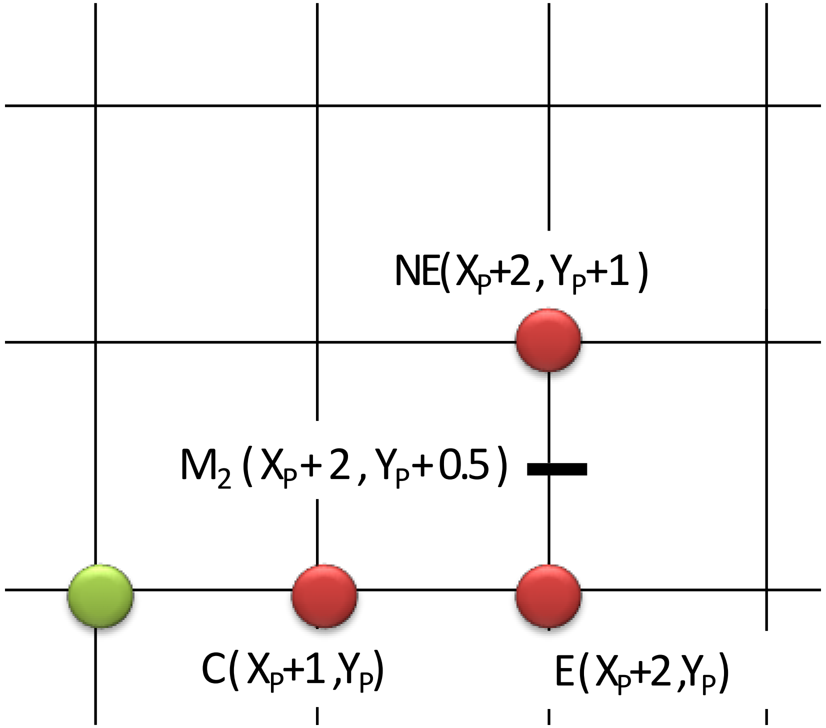
$= F(X_P+1,\ Y_P+0.5)$

$= a(X_P+1) + b(Y_P+0.5) + c$

IF $d_1 \leq 0$, *select E* $(X_P = X_P+1,\ Y_P)$

$d_2 = F(M_2)$

Successive Updating for E (3/4)



$d_1 = F(M_1)$
$= F(X_P+1, Y_P+0.5)$
$= a(X_P+1) + b(Y_P+0.5) + c$

IF $d_1 \leq 0$, *select E* $(X_P = X_P+1, Y_P)$

$d_2 = F(M_2)$
$= F(X_P+2, Y_P+0.5)$
$= a(X_P+2) + b(Y_P+0.5) + c$
$= aX_P + 2a + bY_P + 0.5b + c$
$= aX_P + a + bY_P + 0.5b + c + a$
$= [\ a(X_P+1) + b(Y_P+0.5) + c\ ] + a$
$= d_1 + a$

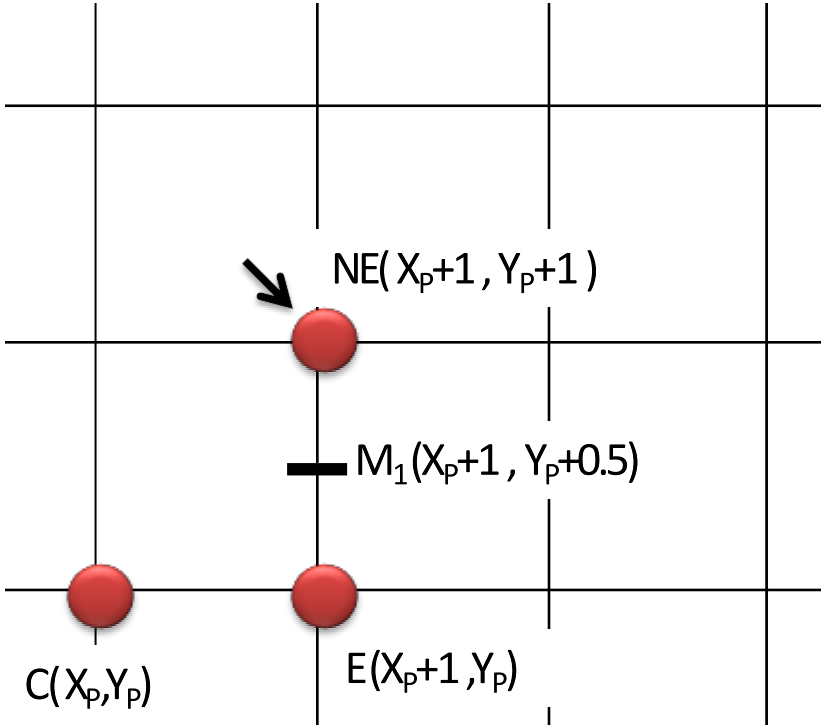
Successive Updating for E (4/4)

Every iteration after selecting E,

we can successively update our decision variable with-

$$d_{NEW} = d_{OLD} + a$$
$$= d_{OLD} + dy$$

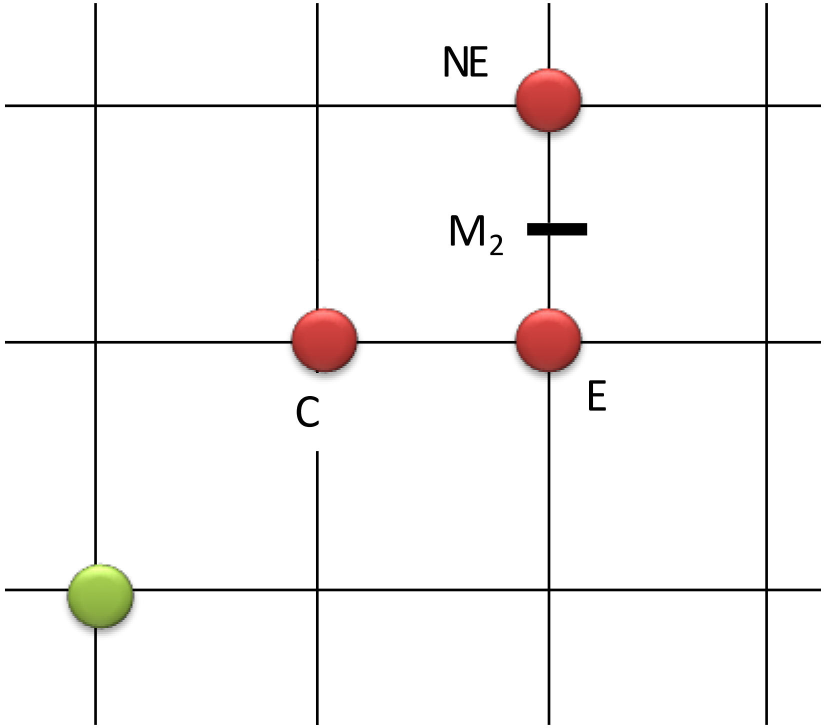
Successive Updating for NE (1/4)



$d_1 = F(M_1)$
$= F(X_P+1, Y_P+0.5)$
$= a(X_P+1) + b(Y_P+0.5) + c$

IF $d_1 > 0$, *select NE (X_P=X_P+1, Y_P=Y_P +1)*

Successive Updating for NE (2/4)



$d_1 = F(M_1)$

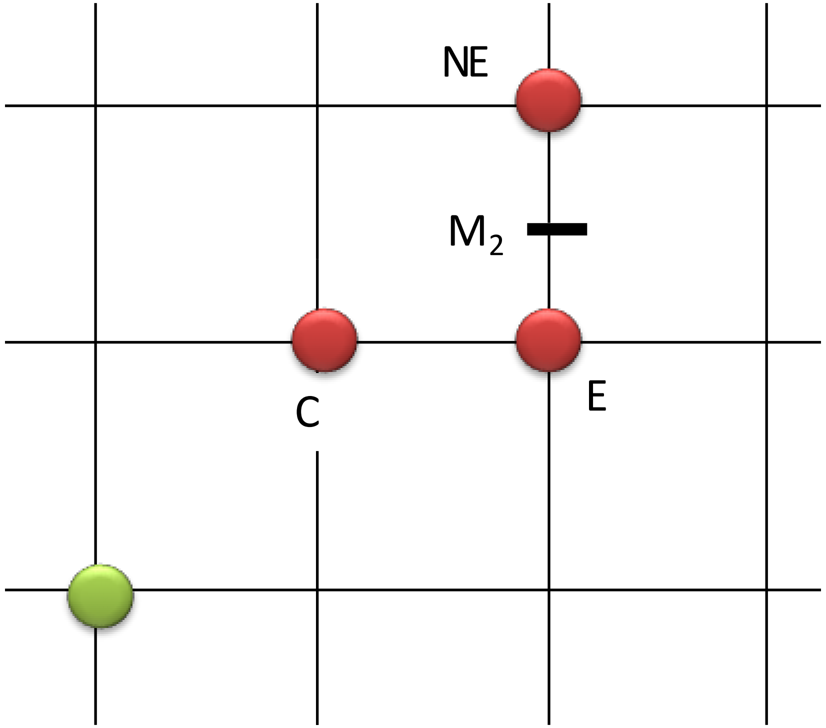
$= F(X_P+1, Y_P+0.5)$

$= a(X_P+1) + b(Y_P+0.5) + c$

IF $d_1 > 0$, *select NE* $(X_P=X_P+1,\ Y_P=Y_P+1)$

$d_2 = F(M_2)$

Successive Updating for NE (3/4)



$d_1 = F(M_1)$
$= F(X_P+1, Y_P+0.5)$
$= a(X_P+1) + b(Y_P+0.5) + c$

IF $d_1 > 0$, *select NE* ($X_P = X_P + 1$, $Y_P = Y_P + 1$)

$d_2 = F(M_2)$
$= F(X_P+2, Y_P+1.5)$

[…. Perform the intermediate steps…]

$= d_1 + (a + b)$

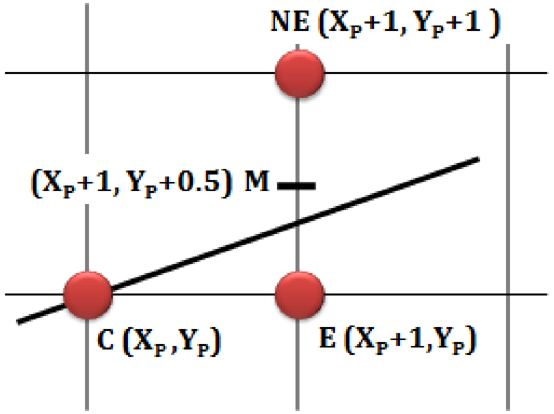
Successive Updating for NE (4/4)

Every iteration after selecting NE,

we can successively update our decision variable with-

$$d_{NEW} = d_{OLD} + (a + b)$$
$$= d_{OLD} + (dy - dx)$$

Midpoint Criteria with Successive Updating (1/1)

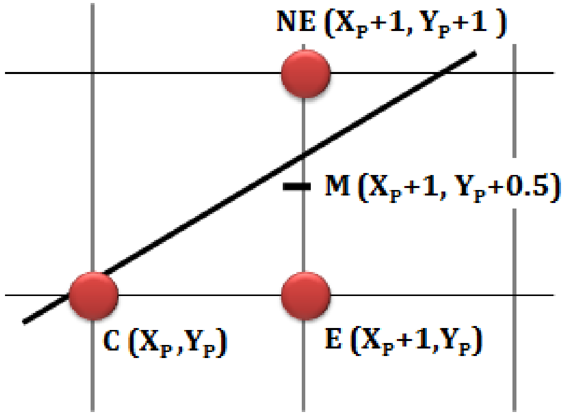


<u>If $d \leq 0$, then:</u>

- midpoint M is above the line,
- E is closer to line, E is selected

Do:

$d = d + \Delta E$, *Where, ΔE = dy*



<u>If $d > 0$, then:</u>

- midpoint M is below the line,
- NE is closer to line, NE is selected

Do:

$d = d + \Delta NE$, *Where, $\Delta NE = dy - dx$*

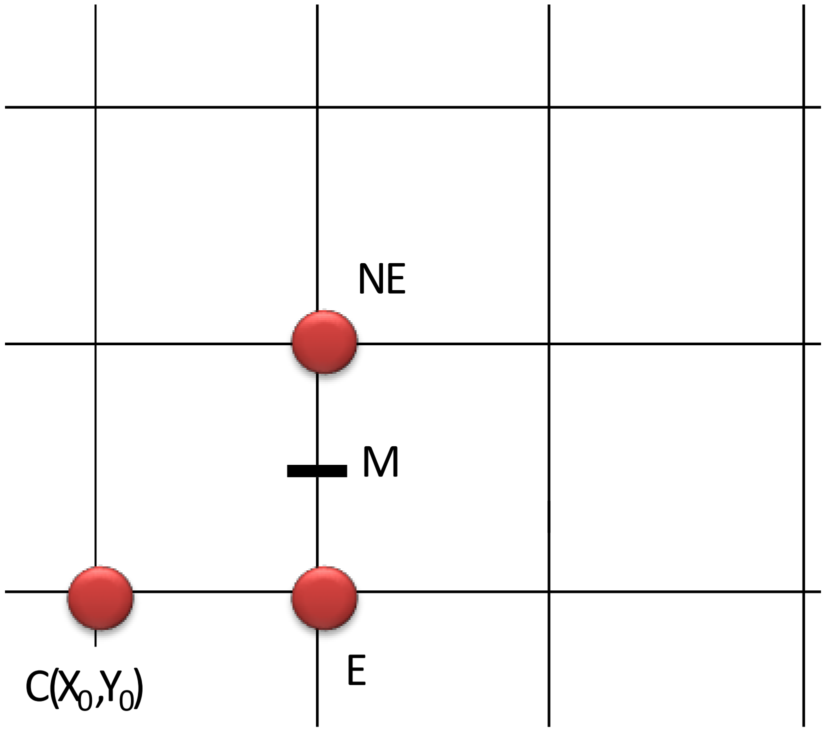
Bresenham's Midpoint Algorithm (1/2)

```
while (x <=x1)
        if d <=0 /* Choose E*/
                d =d + ΔE;

        else /* Choose NE */
                y =y+1
                d =d + ΔNE
        Endif
        x =x+1

        PlotPoint(x, y)
end while
```

Bresenham’s Midpoint Algorithm (2/2)

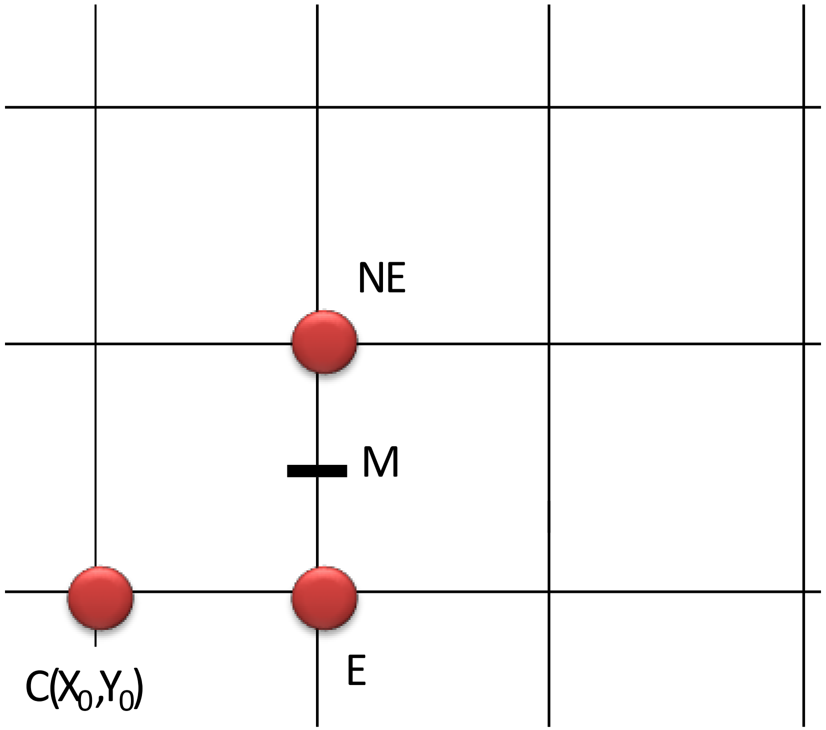
```
while (x <=x1)
    if d <=0   /* 'd' is not initialized!*/
        d =d + ΔE;

    else /* Choose NE */
        y =y+1
        d =d + ΔNE
    Endif
    x =x+1

    PlotPoint(x, y)
end while
```

Initializing the Decision Variable (1/3)



$$
\begin{aligned}
d_{INIT} &= F(M) \\
&= F(X_0+1,\ Y_0+0.5) \\
&= a(X_0+1) + b(Y_0+0.5) + c \\
&= aX_0 + a + bY_0 + 0.5b + c \\
&= aX_0 + bY_0 + c + a + 0.5b \\
&= (aX_0 + bY_0 + c) + a + 0.5b \\
&= F(X_0, Y_0) + a + 0.5b \\
&= a + 0.5b \\
&= dy - 0.5dx
\end{aligned}
$$

Initializing the Decision Variable (2/3)



$$
\begin{aligned}
d_{INIT} &= F(M) \\
&= F(X_0+1,\ Y_0+0.5) \\
&= a(X_0+1) + b(Y_0+0.5) + c \\
&= aX_0 + a + bY_0 + 0.5b + c \\
&= aX_0 + bY_0 + c + a + 0.5b \\
&= (aX_0 + bY_0 + c) + a + 0.5b \\
&= F(X_0, Y_0) + a + 0.5b \\
&= a + 0.5b \\
&= dy - \mathbf{0.5}dx
\end{aligned}
$$

(there is floating point. floating point operation is slower than integer operation)

Initializing the Decision Variable (3/3)

$$d_{INIT} = dy - 0.5dx = 2dy - dx$$
$$\Delta E = 2dy$$
$$\Delta NE = 2(dy - dx)$$

2 is multiplied with d_{INIT} to remove the floating point.

- Observe that, ΔE and ΔNE also multiplied by 2 as those two will be added with d_{INIT} depending on condition.
- Only the sign of the decision variable *d* is needed to select E or NE pixel, not their values.

Bresenham's Midpoint Algorithm (1/1)

Given:

Start point (x0,y0) End point (x1, y1)

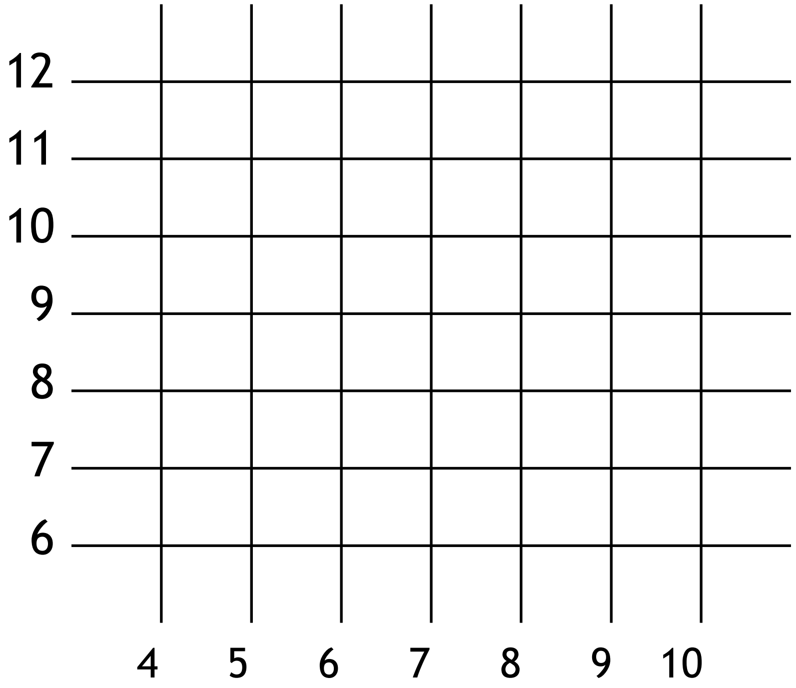
Initialization:

```
x = x0, y = y0;
dx = x1-x0;  dy = y1-y0;
d = 2dy –dx;
ΔE = 2dy;  ΔNE = 2(dy –dx);

PlotPoint(x, y);
```

Loop:

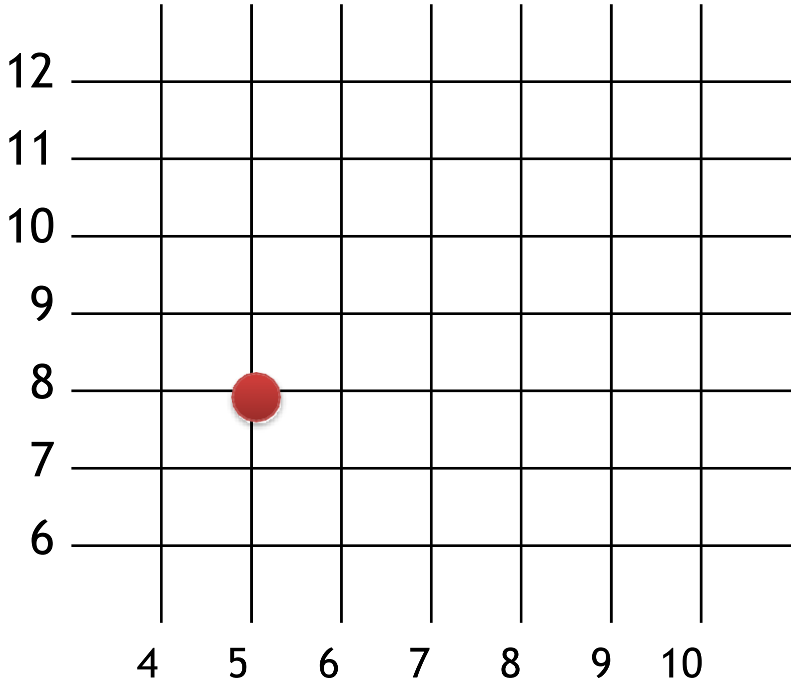
```
while (x <=x1)
    if d <=0 /* Choose E */
        d = d + ΔE;

    else /* Choose NE */
        y = y+1;
        d = d + ΔNE;
    Endif
    x = x+1;
    PlotPoint(x, y);
end while
```

Example (1/10)



Start point $(5, 8)$
End point $(9, 11)$

Example (2/10)



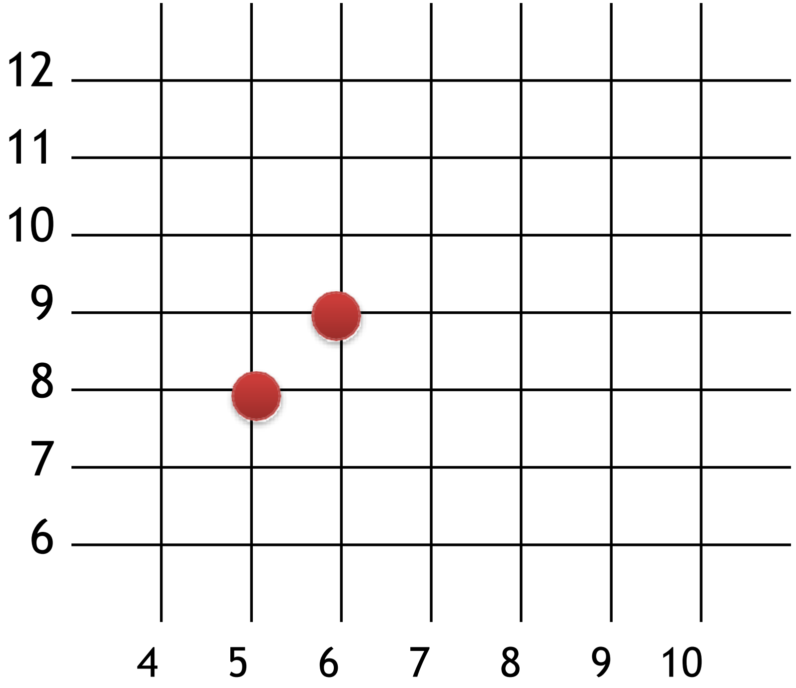
Start point (5, 8)
End point (9, 11)

dy = 3, dx = 4
d = 2dy – dx = 2
ΔE = 2dy = 6
ΔNE = 2(dy – dx) = -2

d = 2

d	2			
(X, Y)				

Example (3/10)

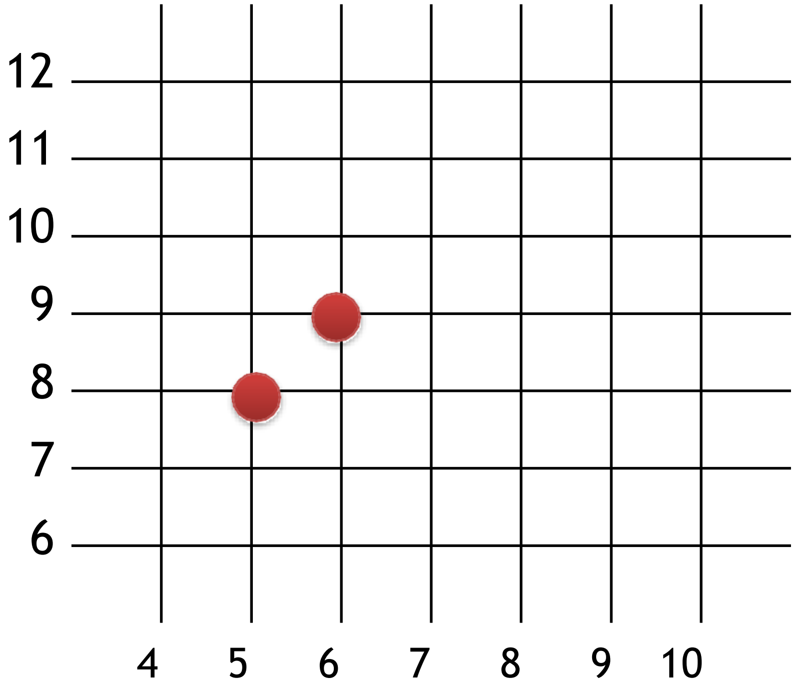


$\Delta E = 2dy = 6$
$\Delta NE = 2(dy - dx) = -2$

d	**2**			
(X, Y)	NE(6, 9)			

d > 0, NE

Example (4/10)



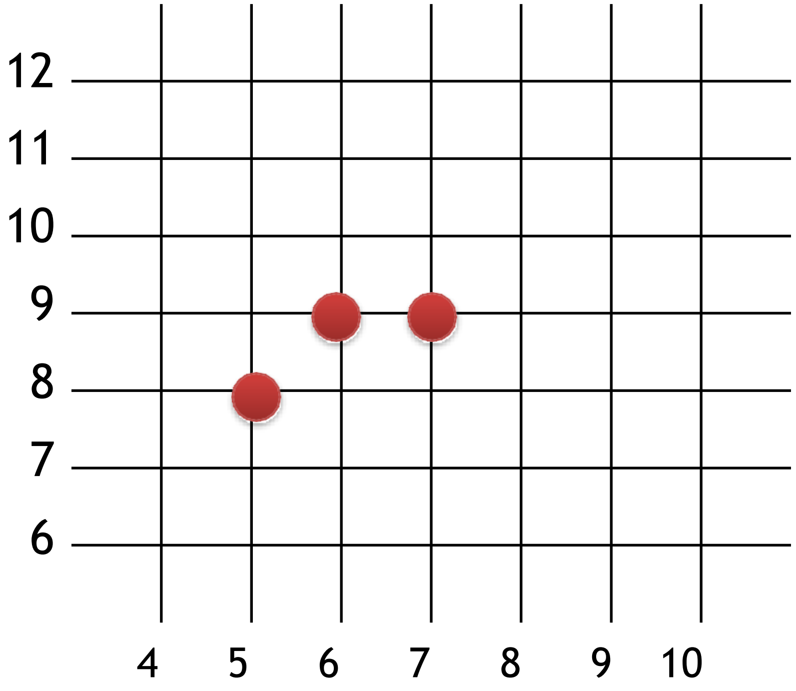
$\Delta E = 2dy = 6$

$\Delta NE = 2(dy - dx) = -2$

$d = 2 + \Delta NE$

d	2	0		
(X, Y)	NE(6, 9)			

Example (5/10)

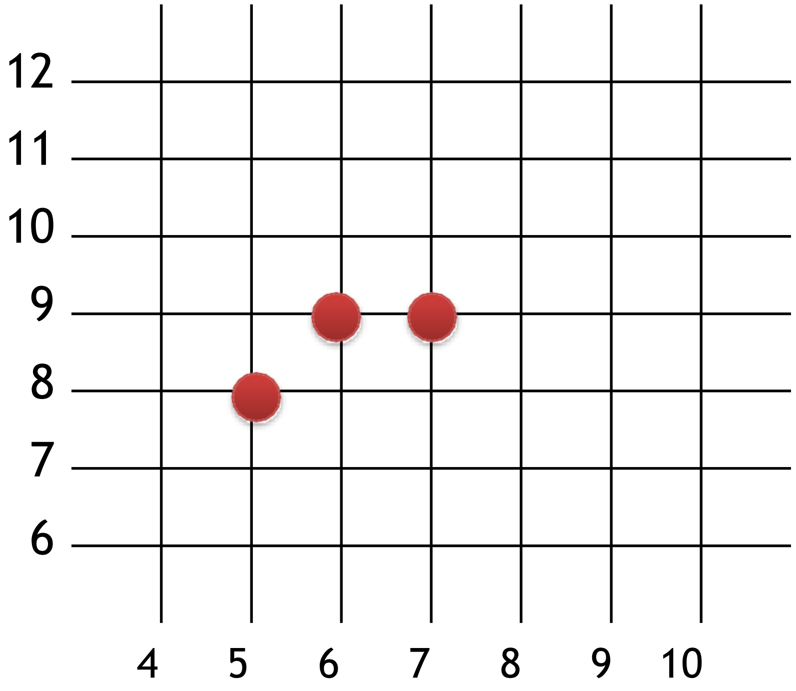


$\Delta E = 2dy = 6$
$\Delta NE = 2(dy - dx) = -2$

d	2	0		
(X, Y)	NE(6, 9)	E(7,9)		

d <= 0, E

Example (6/10)

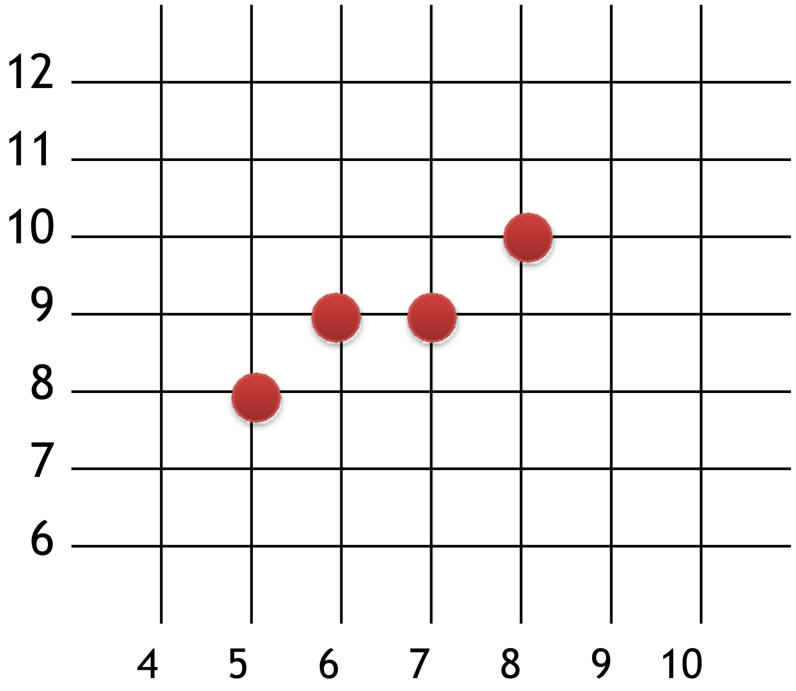


$\Delta E = 2dy = 6$
$\Delta NE = 2(dy - dx) = -2$

$d = 0 + \Delta E$

d	2	0	6	
(X, Y)	NE (6, 9)	E (7,9)		

Example (7/10)



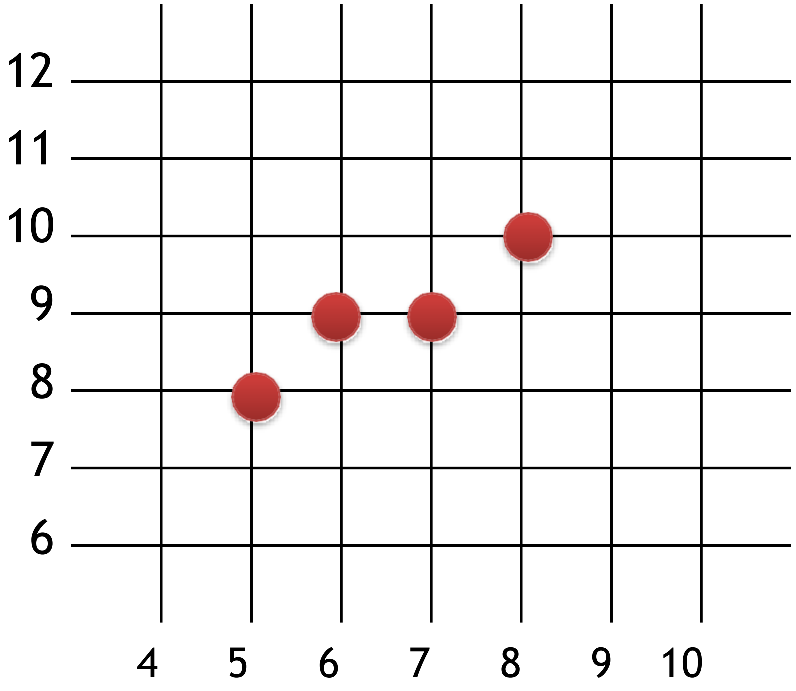
$\Delta E = 2dy = 6$

$\Delta NE = 2(dy - dx) = -2$

d	2	0	6	
(X, Y)	NE (6, 9)	E (7,9)	NE (8,10)	

d > 0, NE

Example (8/10)

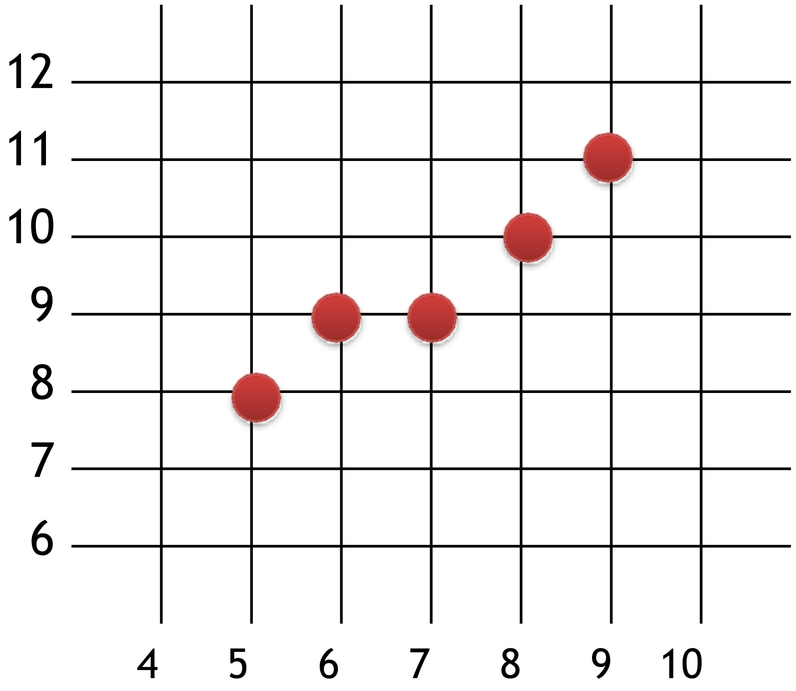


$\Delta E = 2dy = 6$
$\Delta NE = 2(dy - dx) = -2$

$d = 6 + \Delta NE$

d	2	0	6	4
(X, Y)	NE (6, 9)	E (7,9)	NE (8,10)	

Example (9/10)

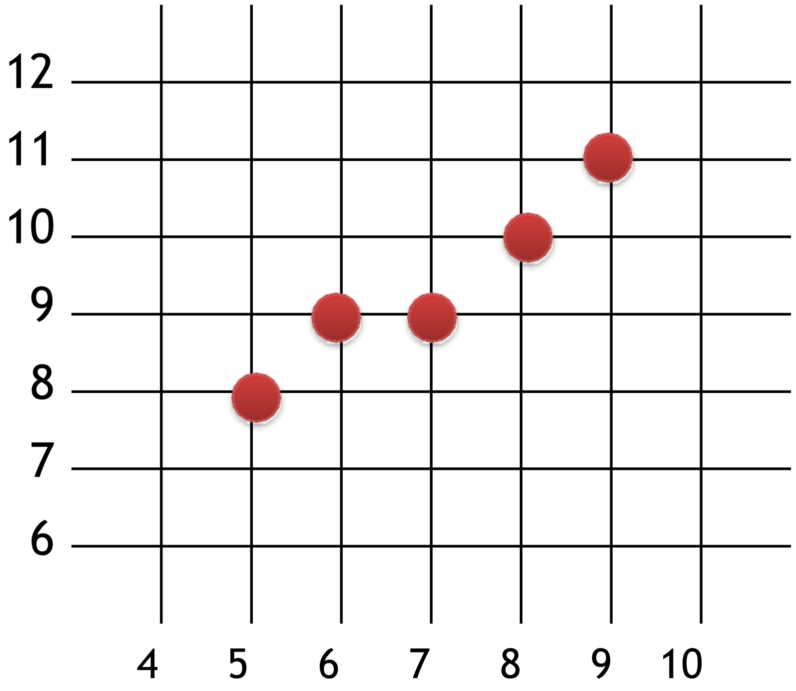


$\Delta E = 2dy = 6$
$\Delta NE = 2(dy - dx) = -2$

d	2	0	6	4
(X, Y)	NE (6, 9)	E (7,9)	NE (8,10)	NE (9,11)

$d > 0$, NE

Example (10/10)



$\Delta E = 2dy = 6$
$\Delta NE = 2(dy - dx) = -2$

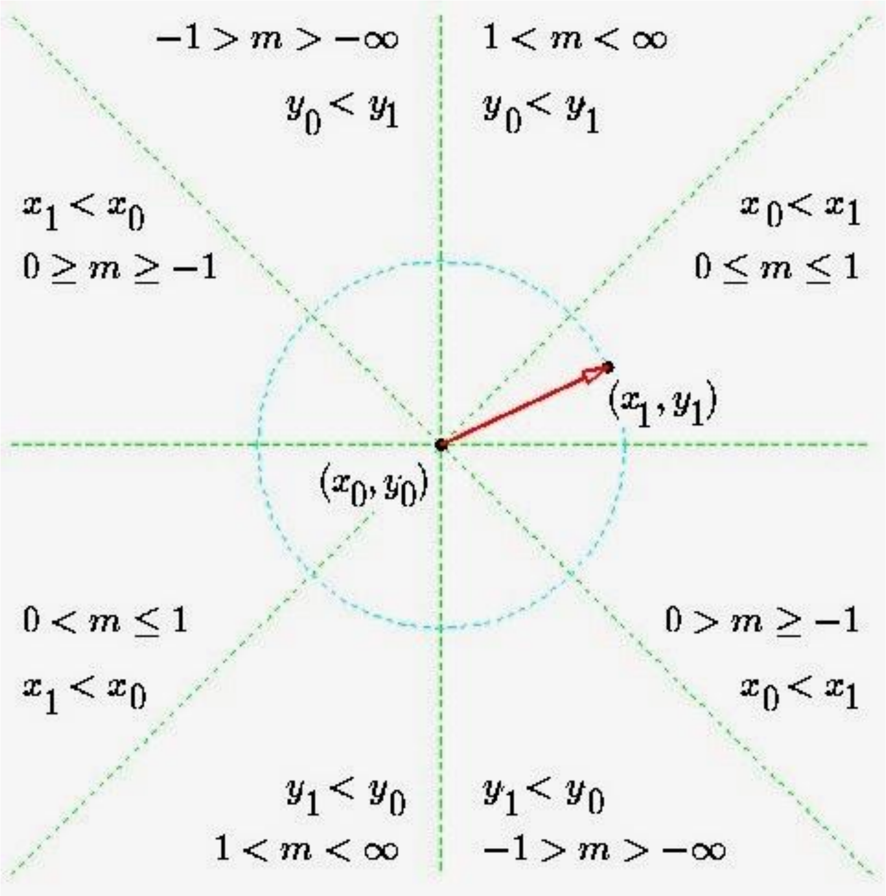
Start point (5, 8)
End point (9, 11)

d	2	0	6	4
(X, Y)	NE (6, 9)	E (7,9)	NE (8,10)	NE (9,11)

d > 0, NE

Rest of the Octant (1/2)

- Find which octant, based on slopes
- See the relations between start and end points



Rest of the Octant (2/2)

- Modify the algorithm accordingly -

(1) plot(x, y)	(2) swap(x, y); plot(y, x)
(5) x=−x; y=−y; plot(−x, −y)	(6) x=−x; y=−y; swap(x, y); plot(−y, −x)

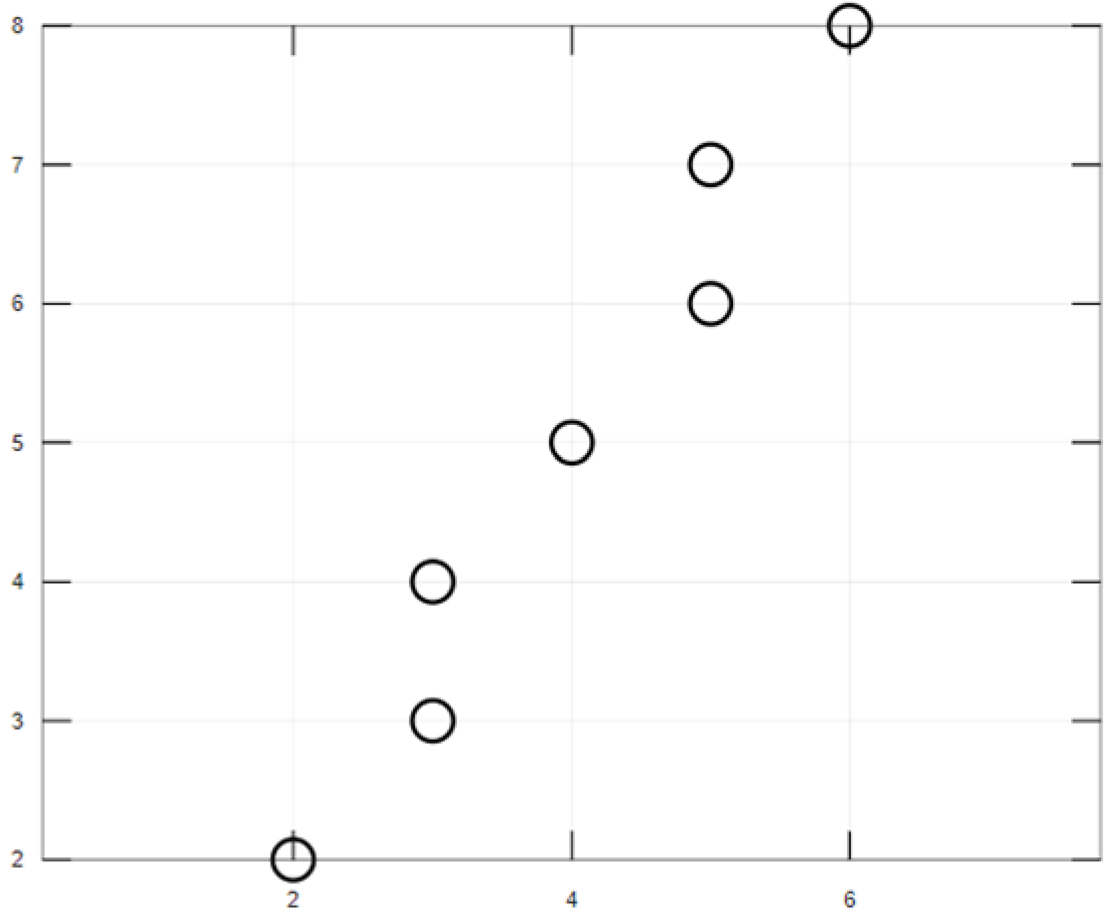
(3) x=−x; swap(x, y); plot(−y, x)	(4) x=−x; plot(−x, y)
(7) y=−y; swap(x, y); plot(y, −x)	(8) y=−y; plot(x, −y)

```
//example:

if (m>1 && y1<y0) //oct == 6
    x0 = -x0;
    x1 = -x1;
    y0 = -y0;
    y1 = -y1;

[x0, y0] = swap(x0, y0);
[x1, y1] = swap(x1, y1);

//line drawing algorithm

plot(-y, -x);
```

Code (1/1)

- https://github.com/imruljubair/bresenhamsAlgorithm

```
Octant:2
moves x   y  d
-----------
      2   2  2
NE   3   3 -2
E    3   4  6
NE   4   5  2
NE   5   6 -2
E    5   7  6
NE   6   8  2
```



Practice Problem

- Rewrite the midpoint algorithm that works for all the octant.
- Perform the midpoint algorithm for a line with two points (5, 8) and (-9, -11).