

Computer Graphics

Unit 1 – Part4

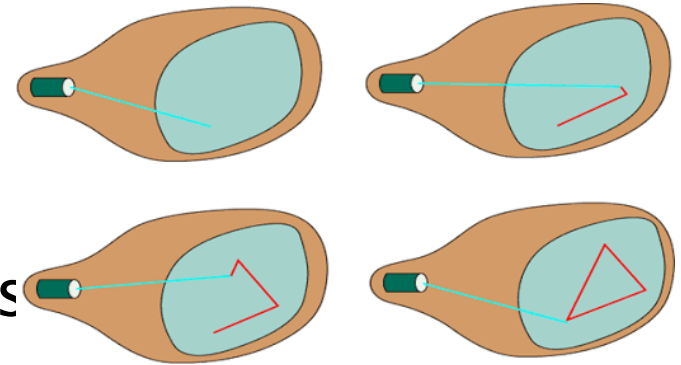
–By Manjula. S

Graphics Architectures

- ▶ Graphics architectures have evolved over decades to reach the current stage.
- ▶ **Categories of Graphics architecture**
 - Early Graphics Architectures (EGA)
 - Display Processor Architectures (DPA)
 - Pipeline Architectures

1. Early Graphics Architecture(EGA)

- ▶ Used computers with standard Von Newmann architecture.
- ▶ Had single processing unit could process only one instruction at a time.
- ▶ Calligraphic CRT display was used
- ▶ Job of host
 - To run application program.
 - compute end points of line segments in image.
 - To send these points to the display unit at a rate high enough to avoid flicker.



Early Graphics Architecture(EGA)

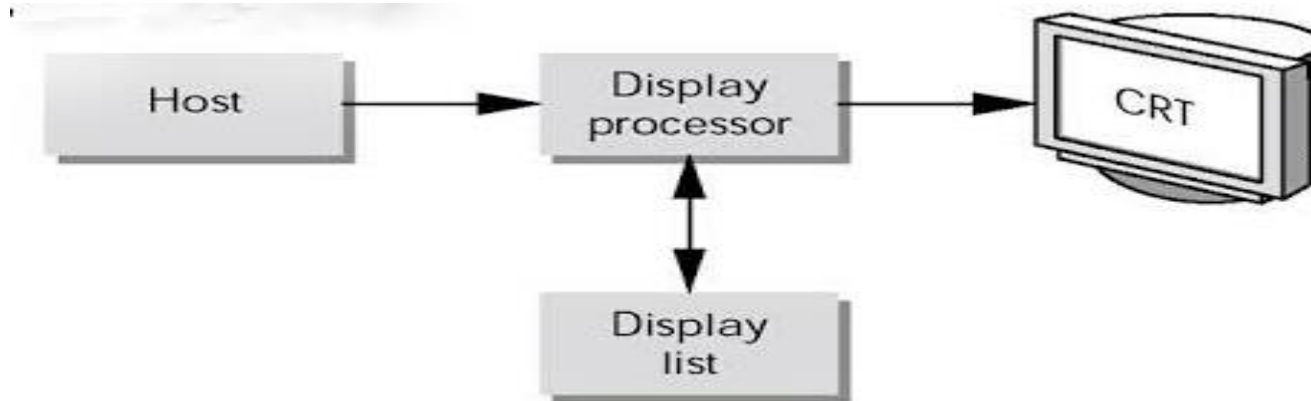
► Advantages

- Simple architecture
- Very less expensive

► Disadvantages

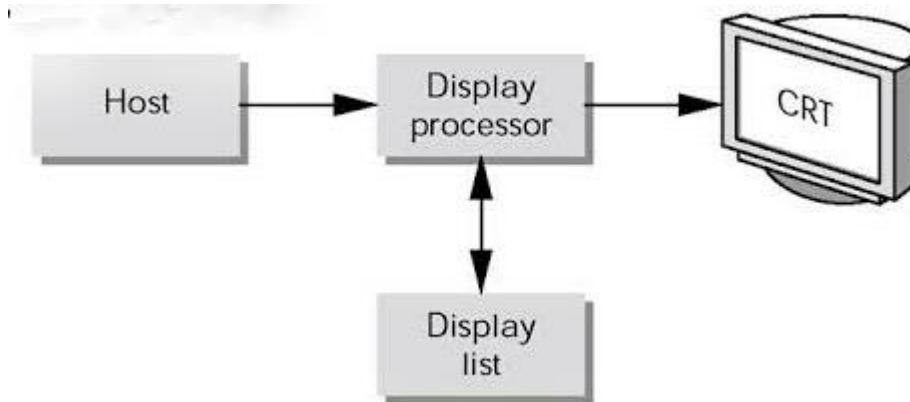
- Very slow
- Not suitable to produce all types of images.
Hence EGA were replaced by DPA.

Display Processor Architecture



- ▶ It had the host computer , display processor, display list(display file) and the output device.
- ▶ The host computer would assemble the instructions (which can generate the image) once and sends it to the display processor

2. Display Processor Architecture



- ▶ The display processor would store these assembled instructions in its own memory called display list.
- ▶ The display processor would then execute the display list continuously independently of the host at a rate which is sufficient to avoid flicker.

2. Display Processor Architecture

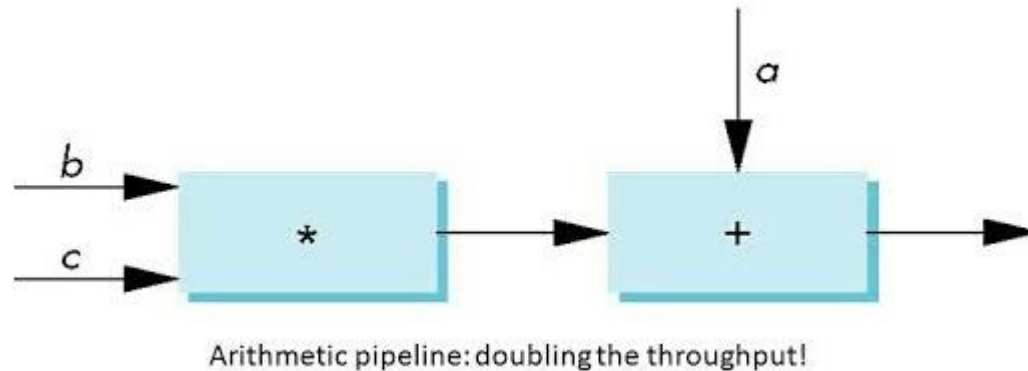
► Advantages

- It frees the host computer from repeatedly assembling the instructions.
- Hence the host is free to perform other tasks.

► Disadvantages

- Not suitable to produce real time 3D graphics images. Hence it was replaced by pipeline architecture.

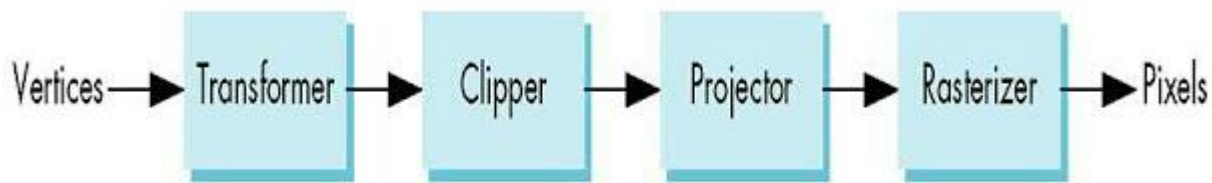
3. Pipeline Architecture



- ▶ Adder and multiplier.
- ▶ To calculate $a + (b * c)$
- ▶ With many values of a,b,c: multiplier can pass on the results of its calculations to the adder and can start its next multiplication while adder carries out second step of calculation on the first set of data.
- ▶ So, the rate at which data flows through the system, the throughput of the system is doubled.

3. Pipeline Architecture

- ▶ Currently used and the most prominent graphics architecture.
- ▶ Makes use of VLSI circuits to generate images.
- ▶ The major steps involved to form an image
 - Vertex processing
 - Clipping and primitive assembly
 - Rasterization
 - Fragment processing



Pipeline Architecture: Geometric pipeline

Pipeline Architecture

a) Vertex processing

- ▶ First step of Pipeline Architecture.
- ▶ Each vertex is processed independently.
- ▶ Carries out 2 functions
 - co-ordinate transformation
 - computing color of each vertex
- ▶ Successive change of co-ordinate system is represented as a single matrix.

Pipeline Architecture

Continued..

- ▶ Transformation can be done by multiplying OR concatenating individual matrices to a single matrix.
- ▶ After multiple stages of transformation, the geometry is transformed by a projection matrix.
- ▶ It also specifies color, properties of the object and light source in the scene.
- ▶ The result of this stage is a set of vertices that specify the object or group of objects.

Pipeline Architecture

b) Clipping and Primitive Assembly

- ▶ Second step of Pipeline Architecture.
- ▶ Output of vertex processing is given as input to this stage.
- ▶ This stage is mainly involved in clipping of the primitive.
- ▶ Clipping Just as a real camera cannot “see” the whole world, the virtual camera can only see part of the world or object space. Objects that are not within this volume are said to be clipped out of the scene.
- ▶ Within this stage, we must assemble sets of vertices into primitives(line segments or polygons) before clipping takes place.
- ▶ The output of this stage is a set of primitives whose projections can appear in the image.

Pipeline Architecture

c) Rasterization

- ▶ Third step of Pipeline Architecture.
- ▶ Converts 2D objects into pixels.
- ▶ The primitives that are output by the clipper are processed to generate pixels in the frame buffer.
- ▶ The output of the rasterizer is a set of fragments for each primitive.
- ▶ Fragments are “potential pixels” having a location in frame buffer, have Color and depth attributes.

Pipeline Architecture

d) Fragment processing

- ▶ Final step of Pipeline Architecture.
- ▶ Fragments are processed to determine the color of the corresponding pixel in the frame buffer.
- ▶ Color of a fragment can be altered by texture mapping.

DDA Algorithm to draw a Line

Digital Differential Analyzer (DDA) Algorithm:

We know the equation of a straight line is

$y = mx + c$ where m is the slope and c is the intercept.

the slope between (x_1, y_1) and (x_2, y_2) is

$$m = (y_2 - y_1) / (x_2 - x_1)$$

or $m = Dy/Dx$

now, if the points are (x_k, y_k) and (x_{k+1}, y_{k+1}) then the slope m is

$$m = (y_{k+1} - y_k) / (x_{k+1} - x_k)$$

DDA Algorithm to draw a Line

- Drawing a line starting from (x_s, y_s) and ending at (x_e, y_e) . So initially we have to calculate the slope of the straight line, using the starting and ending point.

$$m = (y_e - y_s) / (x_e - x_s)$$

- For the different values of m , different calculations are needed.
- **Case 1:** For the value of $(m < 1)$

Then the x coordinate will change in unit intervals

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

$$m = (y_{k+1} - y_k) / (x_{k+1} - x_k)$$

$$\text{but } (x_{k+1} - x_k) = 1$$

$$\text{so } m = (y_{k+1} - y_k)$$

$$\text{or } y_{k+1} = y_k + m$$

DDA Algorithm to draw a Line

➤ **Case 2:** For the value of ($m > 1$)

Then the y coordinate will change in unit intervals

$$\text{so } y_{k+1} = y_k + 1$$

$$m = (y_{k+1} - y_k) / (x_{k+1} - x_k)$$

$$\text{but } (y_{k+1} - y_k) = 1$$

$$\text{so } m = 1 / (x_{k+1} - x_k)$$

$$\text{or } x_{k+1} = (1/m) + x_k$$

➤ **Case 3:** For the value of ($m = 1$)

Then the x and y both coordinate will change in unit intervals

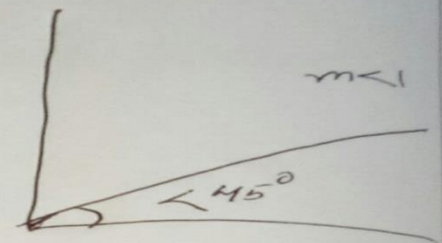
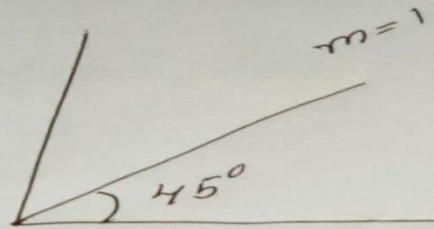
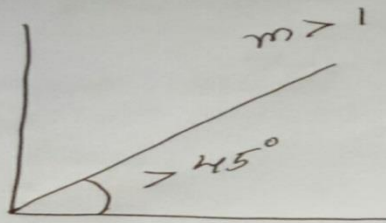
$$\text{so } x_{k+1} = x_k + 1 \text{ and}$$

$$y_{k+1} = y_k + 1$$

DDA Algorithm to draw a Line

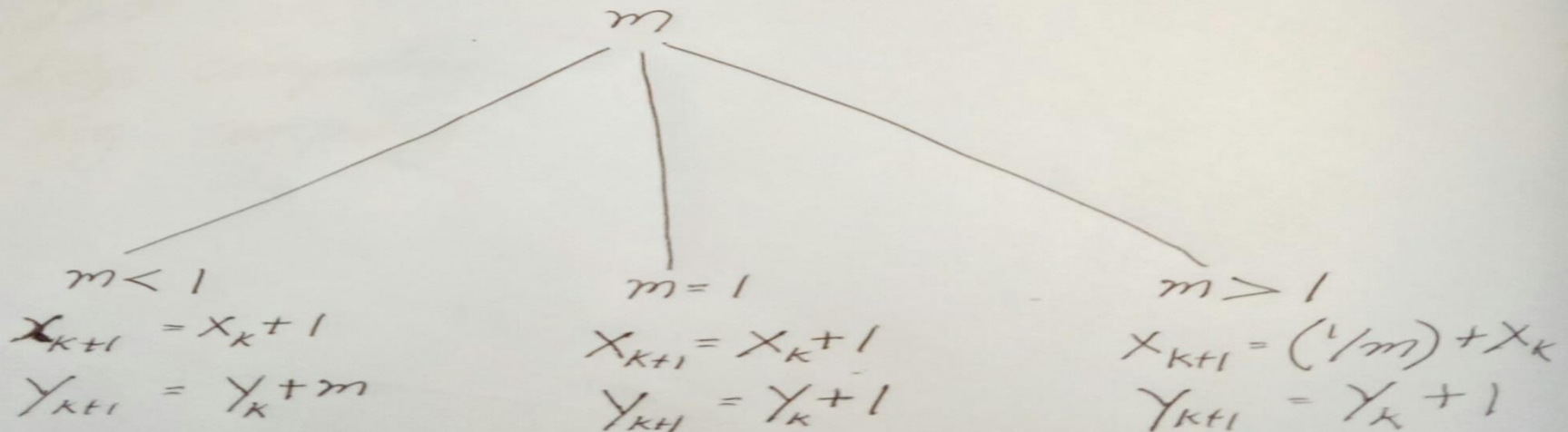
```
DDA (xs, ys, xe, ye : integer)
variable used dx, dy, steps : integer
              xinc, yinc, x, y : real
begin
    dx := xe - xs
    dy := ye - ys
    if |dx| > |dy| then
        steps := |dx|
    else
        steps := |dy|
    end if
    xinc = dx / steps
    yinc = dy / steps
    x := xs
    y := ys
    setPixel(Round(x), Round(y))
    for i := 0 to step - 1 do
        x := x + xinc
        y := y + yinc
        setPixel(Round(x), Round(y))
    end for
end
```

DDA Algorithm to draw a Line



$m = \text{slope}$

$$m = (y_e - y_s) / (x_e - x_s) = \frac{\Delta y}{\Delta x} = \frac{dy}{dx}$$



DDA Algorithm to draw a Line

Draw a straight line from (0, 0) to (4, 5)

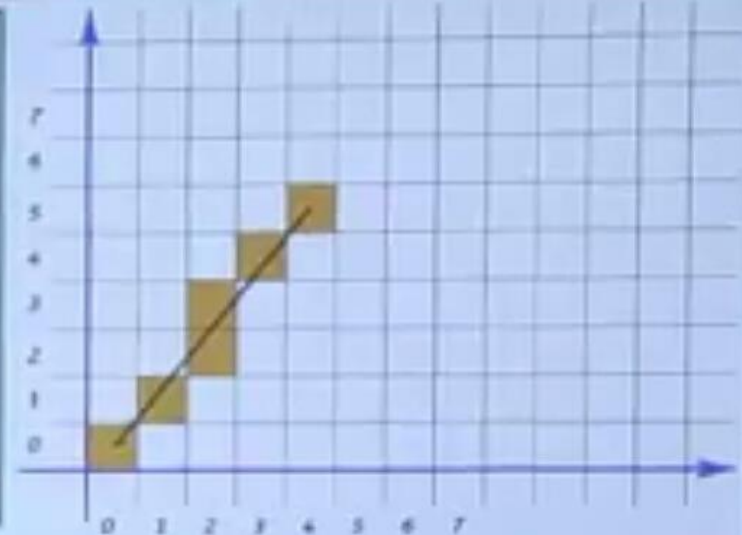
$$m = Dy / Dx = (y_{k+1} - y_k) / (x_{k+1} - x_k)$$

$m = (5 - 0) / (4 - 0) = 5/4 = 1.25 > 1$ (The slope is greater than 1)

as $m > 1$ then y moves in unit interval and x moves $x_k + (1/m)$

so the next point is $(x_{k+1}, y_{k+1}) = (x_k + (1/m), y_k + 1)$

$$m = 5/4 \text{ so } 1/m = 4/5 = 0.8$$



x	y	x-plot	y-plot	(x, y)	Actual value of x
0	0	0	0	(0, 0)	$x = 0$
0.8	1	1	1	(1, 1)	$\text{Ceil}(0.8)$
1.6	2	2	2	(2, 2)	$\text{Ceil}(1.6)$
2.4	3	2	3	(2, 3)	$\text{Floor}(2.4)$
3.2	4	3	4	(3, 4)	$\text{Floor}(3.2)$
4	5	4	5	(4, 5)	$x = 4$

DDA Algorithm to draw a Line

- ▶ DDA Algorithm is an incremental method of scan conversion of line.
- ▶ In this method calculation performed at each step but by using results of previous steps.
- ▶ It allow us to detect the change in the value of x & y. so plotting of same point twice is not possible.

Advantages:

- ▶ It is the simplest algorithm and it does not require special skills for implementation.
- ▶ It is a faster method for calculating pixel positions than the direct use of equation $y=mx+b$.

Disadvantages:

- ▶ It takes lot of computation time because at each and every step it has to round off.
- ▶ Floating point calculations and rounding operations are expensive and time consuming.
- ▶ Accumulation of round off error may drift the long line segment.

Bresenham Line Algorithm

- ▶ Faster than DDA
- ▶ An accurate and efficient raster line generating algorithm developed by Bresenham.
- ▶ It involves only integer addition, subtraction, multiplication operation.
- ▶ These operations can be performed very rapidly so lines can be generated quickly as compared to DDA.
- ▶ In this algorithm, moving across the x-axis in unit interval and at Y-axis choose between two different coordinates.

The Bresenham Line Algorithm

BRESENHAM'S LINE DRAWING ALGORITHM (for $|m| < 1.0$)

1. Input the two line end-points, storing the left end-point in (x_0, y_0)
2. Plot the point (x_0, y_0)
3. Calculate the constants Δx , Δy , $2\Delta y$, and $(2\Delta y - \Delta x)$ and get the first 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:

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

Bresenham Line Algorithm

the equation of the straight line is

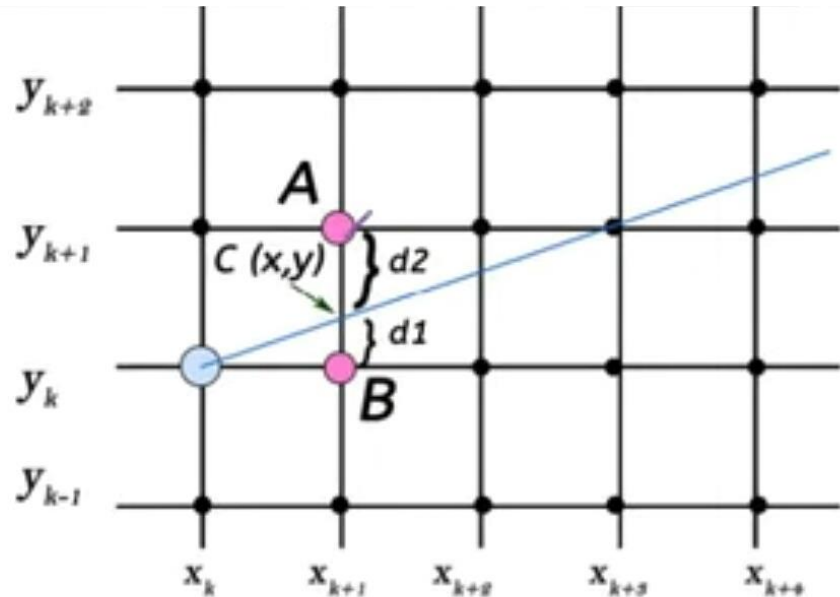
$$y = mx + c$$

now for the value of y for

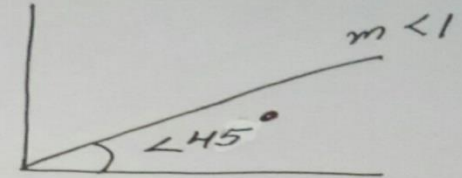
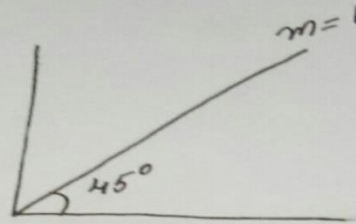
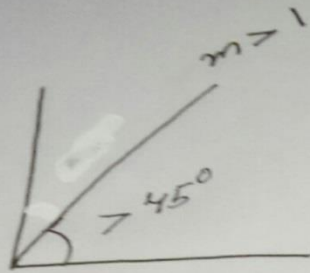
$$x = x_k + 1;$$

$$y = m(x_k + 1) + c \dots\dots(i)$$

In the diagram, the starting point is Blue Dot (x_k, y_k) . There are two points A & B are there on the next grid line. The line intersects the grid line at point C (x, y) . The $AC = d2$ and $BC = d1$. With the values of $d1$ and $d2$, the next point will be selected. Here a new term is introduced, the decision parameter(p_k). This parameter will take the decision, which grid line will be selected, y_k or y_{k+1}



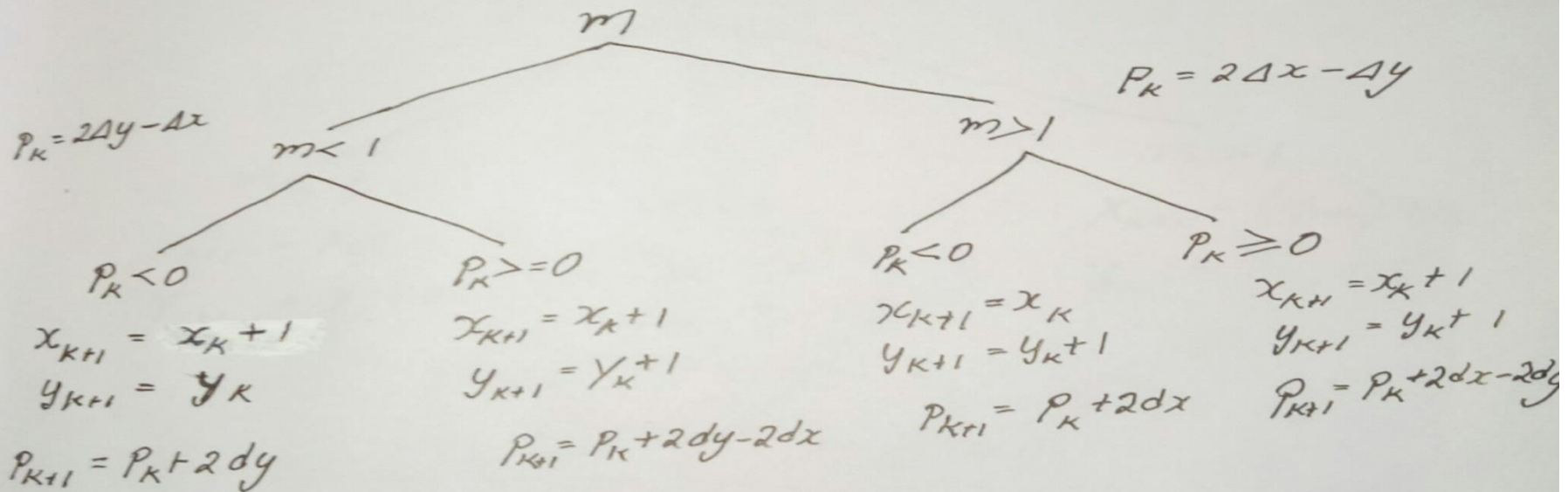
Bresenham Line Algorithm



$m = \text{slope}$

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{dy}{dx} = \frac{\Delta y}{\Delta x}$$

Decision Variable $P_K = 2dy - dx$ or $2\Delta y - \Delta x$



Bresenham Line Algorithm

Problem on Bresenham's Line Drawing Algorithm

Draw a straight line from (35, 40) to (43, 45)

$$m = \Delta y / \Delta x$$

Formulas

Value of m	P_k	P_{k+1}	$(P_k > 0)$	$(P_k < 0)$
$m < 1$	$P_k = 2\Delta y - \Delta x$	$P_{k+1} = P_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$	(x_{k+1}, y_{k+1})	(x_{k+1}, y_k)
$m > 1$	$P_k = 2\Delta x - \Delta y$	$P_{k+1} = P_k + 2\Delta x - 2\Delta y(x_{k+1} - x_k)$	(x_{k+1}, y_{k+1})	(x_k, y_{k+1})

$$m = (45 - 40) / (43 - 35) = 5/8 = 0.6 < 1$$

Here the slope m is less than 1 and positive, so following formulas will be used.

Bresenham Line Algorithm

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

$$P_{k+1} = P_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

$$\Delta x = 8 \quad \Delta y = 5$$

$$2\Delta x = 16 \quad 2\Delta y = 10$$

Initial Decision parameter:

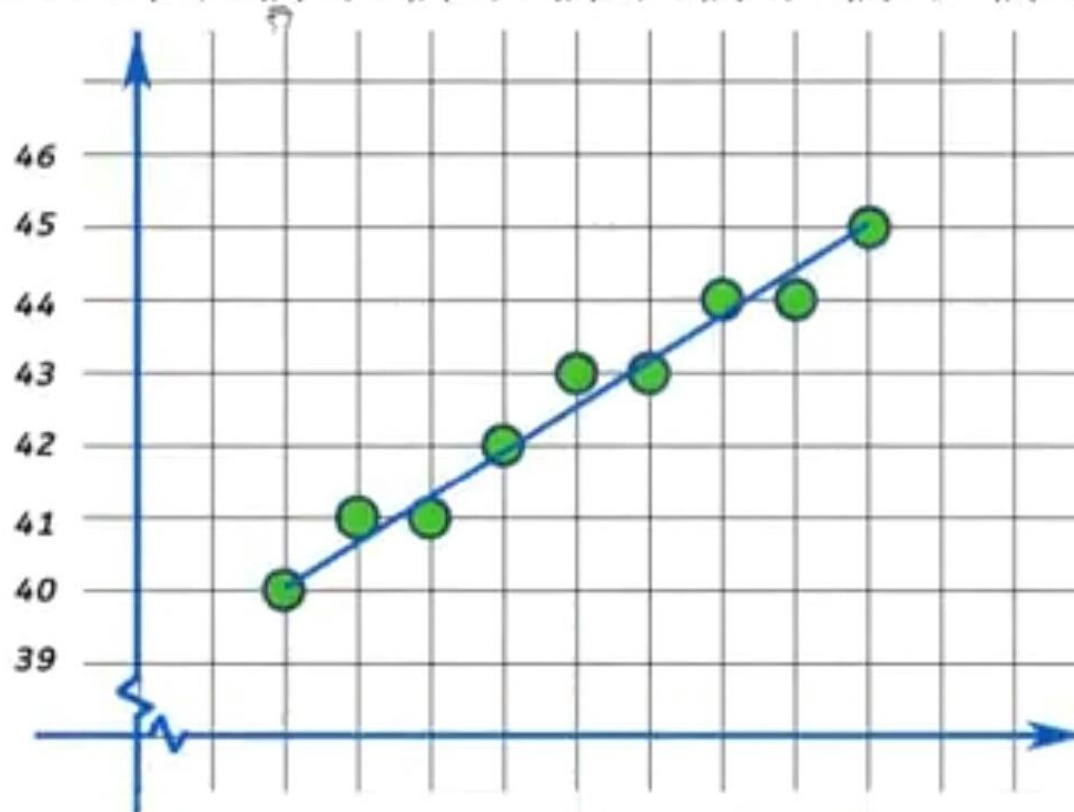
$$P_0 = 2\Delta y - \Delta x = 10 - 8 = 2 > 0 \quad (x_{k+1}, y_{k+1})$$

Iteration	(x_k, y_k)	P_k	(x_{k+1}, y_{k+1})	P_{k+1}
0	(35, 40)	2	(36, 41)	$2 + 10 - 16(41 - 40) = -4$
1	(36, 41)	-4	(37, 41)	$-4 + 10 - 16(41 - 41) = 6$
2	(37, 41)	6	(38, 42)	$6 + 10 - 16(42 - 41) = 0$
3	(38, 42)	0	(39, 43)	$0 + 10 - 16(43 - 42) = -6$
4	(39, 43)	-6	(40, 43)	$-6 + 10 - 16(43 - 43) = 4$
5	(40, 43)	4	(41, 44)	$4 + 10 - 16(44 - 43) = -2$
6	(41, 44)	-2	(42, 44)	$-2 + 10 - 16(44 - 44) = 8$
7	(42, 44)	8	(43, 45)	-----

Bresenham Line Algorithm

These are the points between (35, 40) and (43, 45)

Points are: (35, 40), (36, 41), (37, 41), (38, 42), (39, 43), (40, 43), (41, 44), (42, 44), (43, 45);



Bresenham Circle Drawing Algorithm

Explanation :

- In this algorithm, we sample at unit interval and determine the nearest pixel on circle path each step.
- In this first quadrant of circle from $X = 0$ to $X = Y$, the slope of the curve varies from 0 to -1.
- So, we take unit step in positive X-direction over this octant and using a decision parameter we can determine which of the two possible Y – position is closer to the circle path at each step.
- And as above step we find the Y – position in other seven octant.

Bresenham Circle Drawing Algorithm

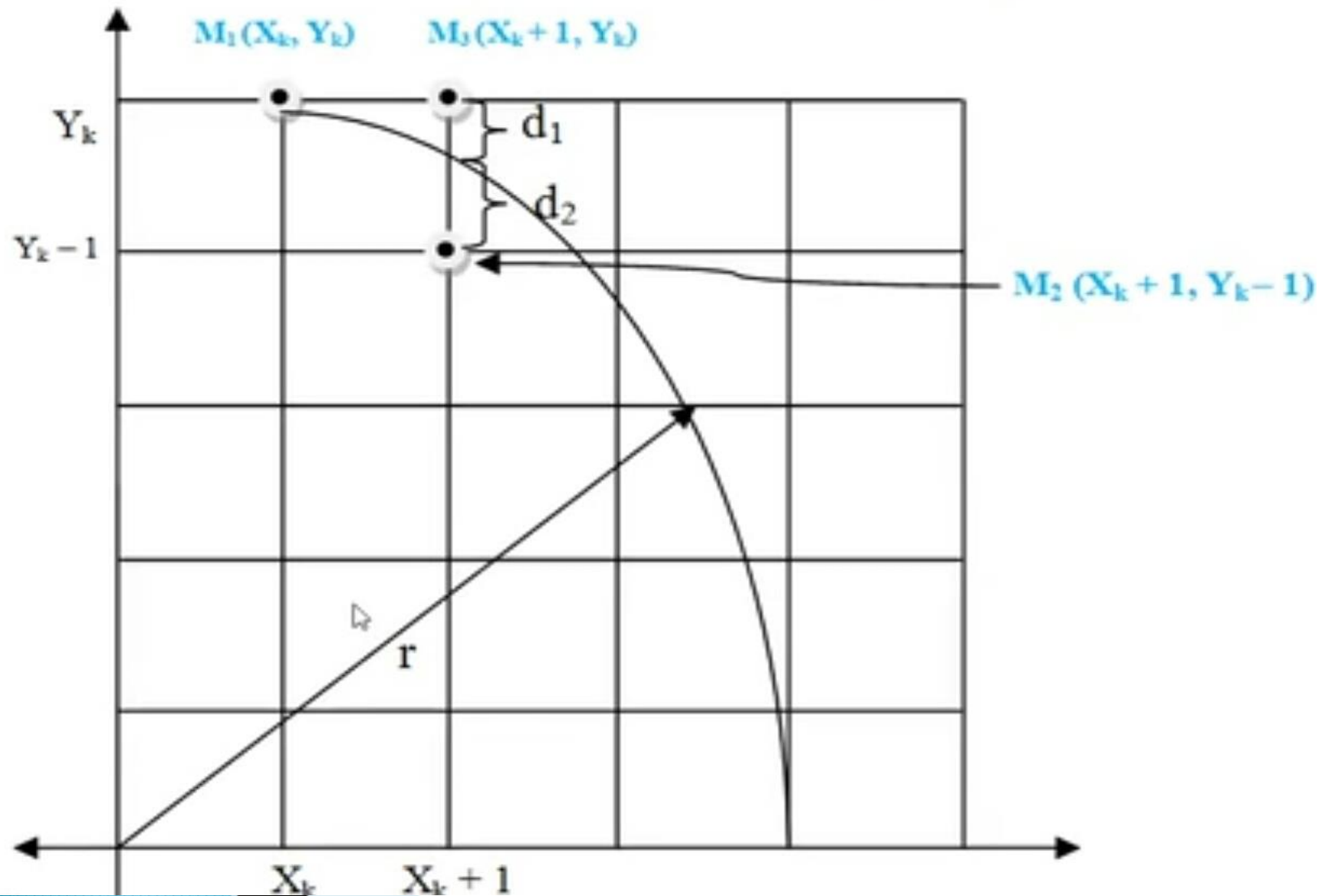
- Circle function is,

$$f_{\text{circle}}(X, Y) = X^2 + Y^2 - r^2$$

- Any point (X, Y) on the boundary of the circle with radius r satisfies the equation $f_{\text{circle}}(X, Y) = 0$
- In general the relative position of any point (X, Y) can be determined by sign of the circle function

$$f_{\text{circle}}(X, Y) \begin{cases} < 0, (x, y) \text{ inside circle} \\ = 0, (x, y) \text{ on the circle} \\ > 0, (x, y) \text{ outside circle} \end{cases}$$

Bresenham Circle Drawing Algorithm



Bresenham Circle Drawing Algorithm

- After point M_1 , do we choose M_2 or M_3 ?

d_1 = distance of M_3 from circle

d_2 = distance of M_2 from circle

$$\therefore d_1 = (X_k + 1)^2 + Y_k^2 - r^2 \quad [\text{always +ve}]$$

$$\therefore d_2 = (X_k + 1)^2 + (Y_k - 1)^2 - r^2 \quad [\text{always -ve}]$$

- Decision parameter $P_k = d_1 + d_2$

$$\begin{aligned} P_k &= (X_k + 1)^2 + Y_k^2 - r^2 + (X_k + 1)^2 + (Y_k - 1)^2 - r^2 \\ &= 2(X_k + 1)^2 + Y_k^2 + (Y_k - 1)^2 - 2r^2 \quad \text{-----}(1) \end{aligned}$$

- So, if $P_k < 0$ then circle is closer to M_3 (above point),

$$(X_{k+1}, Y_k) = (X_k + 1, Y_k)$$

- Otherwise, $P_k \geq 0$ then circle is closer to M_2 (below point),

$$(X_{k+1}, Y_k) = (X_k + 1, Y_k - 1)$$

Bresenham Circle Drawing Algorithm

- To find next decision parameter, P_{k+1}

$$P_{k+1} = 2(X_{k+1} + 1)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 - 2r^2 \text{ -----(2)}$$

- Difference between equation (2) - (1)

$$\begin{aligned} P_{k+1} - P_k &= 2(X_{k+1} + 1)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 - 2r^2 \\ &\quad - 2(X_k + 1)^2 - Y_k^2 - (Y_k - 1)^2 + 2r^2 \\ &= 2(X_k + 1 + 1)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 \\ &\quad - 2(X_k + 1)^2 - Y_k^2 - (Y_k - 1)^2 \quad (\text{where } X_{k+1} = X_k + 1) \end{aligned}$$

$$\begin{aligned} &= 2(X_k + 2)^2 + Y_{k+1}^2 + (Y_{k+1} - 1)^2 \\ &\quad - 2(X_k + 1)^2 - Y_k^2 - (Y_k - 1)^2 \end{aligned}$$

$$\begin{aligned} &= 2(X_k^2 + 4X_k + 4) + Y_{k+1}^2 + Y_{k+1}^2 - 2Y_{k+1} + 1 \\ &\quad - 2(X_k^2 + 2X_k + 1) - Y_k^2 - Y_k^2 + 2Y_k - 1 \end{aligned}$$

$$\begin{aligned} &= 2X_k^2 + 8X_k + 8 + 2Y_{k+1}^2 - 2Y_{k+1} \\ &\quad - 2X_k^2 - 4X_k - 2 - 2Y_k^2 + 2Y_k \end{aligned}$$

$$= 4X_k + 6 + 2Y_{k+1}^2 - 2Y_{k+1} - 2Y_k^2 + 2Y_k$$

$$P_{k+1} = P_k + 4X_k + 2(Y_{k+1}^2 - Y_k^2) - 2(Y_{k+1} - Y_k) + 6$$

Bresenham Circle Drawing Algorithm

- If $P_k < 0$ (M_3 is select) then $Y_{k+1} = Y_k$



$$P_{k+1} = P_k + 4X_k + 2(Y_k^2 - Y_k^2) - 2(Y_k - Y_k) + 6$$

$$P_{k+1} = P_k + 4X_k + 6$$

- If $P_k \geq 0$ (M_2 is select) then $Y_{k+1} = Y_k - 1$

$$\begin{aligned} P_{k+1} &= P_k + 4X_k + 2[(Y_k - 1)^2 - Y_k^2] - 2[(Y_k - 1) - Y_k] + 6 \\ &= P_k + 4X_k + 2[Y_k^2 - 2Y_k + 1 - Y_k^2] - 2[Y_k - 1 - Y_k] + 6 \\ &= P_k + 4X_k - 4Y_k + 2 + 2 + 6 \\ &= P_k + 4X_k - 4Y_k + 10 \end{aligned}$$

$$P_{k+1} = P_k + 4(X_k - Y_k) + 10$$

Bresenham Circle Drawing Algorithm

- Here initial decision parameter P_0 ,

$$\begin{aligned}P_0 &= d_1 + d_2 \\&= [1^2 + \cancel{r^2} - \cancel{r^2}] + [1^2 + (r-1)^2 - r^2] \\&= 1 + 1 + \cancel{r^2} - 2r + 1 - \cancel{r^2} \\P_0 &= 3 - 2r\end{aligned}$$

Bresenham Circle Drawing Algorithm

Algorithm:

Step-1: Input radius r and circle centre (X_c, Y_c) obtained the first point $(X_0, Y_0) = (0, r)$

Step-2: Calculate the initial value of decision parameter as

$$P_0 = 3 - 2r$$

Step-3: At each X_k position, starting at $k = 0$, perform, the following test:

If $P_k < 0$, the next point is $(X_k + 1, Y_k)$

$$P_{k+1} = P_k + 4X_k + 6$$

otherwise the next point is $(X_k + 1, Y_k - 1)$

$$P_{k+1} = P_k + 4(X_k - Y_k) + 10$$

Bresenham Circle Drawing Algorithm

Algorithm:

Step-4: Determine the symmetry points in other seven octant

Step-5: Move each pixel position (X, Y) into circular path

$$X = X + X_c \quad \text{and} \quad Y = Y + Y_c$$

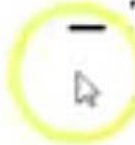
Step-6: Repeat step 3 to 5 until $X \geq Y$.



Bresenham Circle Drawing Algorithm

Example-1: Given a circle radius $r = 10$, we demonstrate the Bresenham circle drawing algorithm by determining position along the circle octant in the first quadrant from $X=0$ to $X=Y$.

Solution:

– The initial decision parameter P_0
 $\Rightarrow P_0 = 3 - 2r = 3 - 20 = -17 \quad \therefore P_0 < 0 \Rightarrow (X_1, Y_1) = (1, 10)$

– For the circle centred on the coordinator origin, the initial point is $(X_0, Y_0) = (0, 10)$ and initial increment terms for calculating the decision parameter are

$$\begin{aligned}\Rightarrow P_1 &= P_0 + 4X_0 + 6 \\ &= -17 + 0 + 6 \\ &= -11\end{aligned}$$

$$\therefore P_1 < 0 \Rightarrow (X_2, Y_2) = (2, 10)$$

Bresenham Circle Drawing Algorithm

$$\begin{aligned}\Rightarrow P_2 &= P_1 + 4X_1 + 6 \\ &= -11 + 4 + 6 \\ &= -1\end{aligned}$$

$$\therefore P_2 < 0 \Rightarrow (X_3, Y_3) = (3, 10)$$

$$\begin{aligned}\Rightarrow P_3 &= P_2 + 4X_2 + 6 \\ &= -1 + 8 + 6 \\ &= 13\end{aligned}$$

$$\therefore P_3 > 0 \Rightarrow (X_4, Y_4) = (4, 9)$$

$$\begin{aligned}\Rightarrow P_4 &= P_3 + 4(X_3 - Y_3) + 10 \\ &= 13 - 28 + 10 \\ &= -5\end{aligned}$$

$$\therefore P_4 < 0 \Rightarrow (X_5, Y_5) = (5, 9)$$

Bresenham Circle Drawing Algorithm

$$\begin{aligned}\Rightarrow P_5 &= P_4 + 4X_4 + 6 \\ &= -5 + 16 + 6 \\ &= 17\end{aligned}$$

$$\therefore P_5 > 0 \Rightarrow (X_6, Y_6) = (6, 8)$$

$$\begin{aligned}\Rightarrow P_6 &= P_5 + 4(X_5 - Y_5) + 10 \\ &= 17 - 16 + 10 \\ &= 11\end{aligned}$$

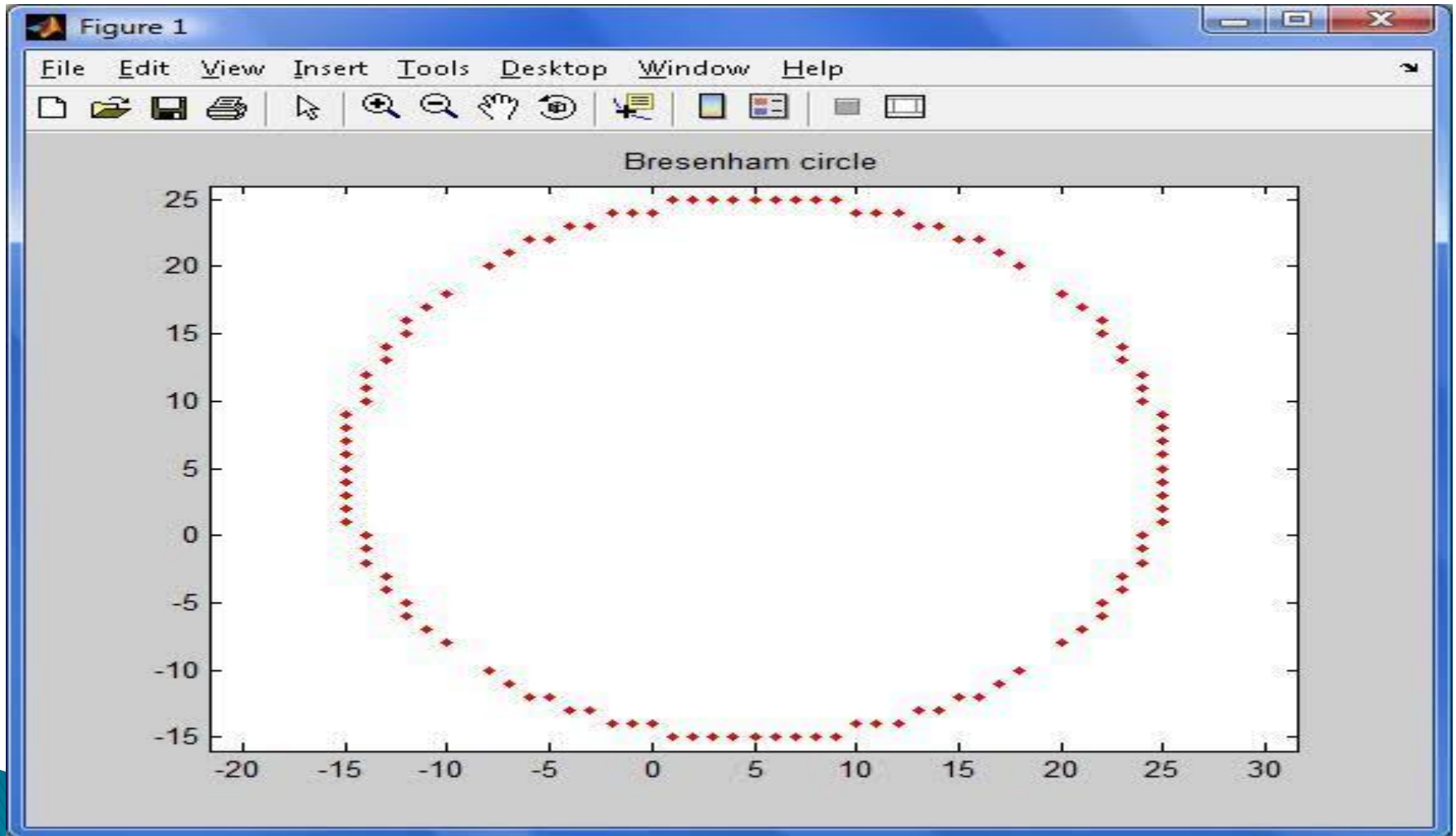
$$\therefore P_6 > 0 \Rightarrow (X_7, Y_7) = (7, 7)$$

Bresenham Circle Drawing Algorithm

➤ Final summery table:

K	d_k	New Point (X_{k+1}, Y_{k+1})
0	-17	(1, 10)
1	-11	(2, 10)
2	-1	(3, 10)
3	13	(4, 9)
4	-5	(5, 9)
5	17	(6, 8)
6	11	(7, 7)

Bresenham Circle Drawing Algorithm



Bresenham Circle Drawing Algorithm

Advantages:

- ▶ Easy to implement.
- ▶ It is fast and incremental.
- ▶ It uses only integer calculations.
- ▶ The performance of this algorithm is faster compared to DDA.

Disadvantages:

- ▶ It is used only to draw basic line.
- ▶ It is not meant for smooth lines.

Thank You