Computer Graphics 05101301

Prof. Hina Chokshi, Sr. Assistant Professor
Parul Institute of Computer Application - BCA

CHAPTER-2

Graphic Primitives

Line Drawing Algorithms

They are clasified into…..

- Digital Differential Analyzer (DDA) Line Drawing Algorithm
- Bresenham Line Drawing Algorithm
- Mid Point Line Drawing Algorithm

Image source : seekimages

DDA Algorithm

DDA Algorithm is the simplest line drawing algorithm.

Given-

Starting coordinates = (X_0, Y_0)

Ending coordinates = (X_n, Y_n)

1) Find dx,dy,m=dy/dx, dx=x2-x1 dy=y2-y1
2) Find which is greater value dx or dy? Intermediate points between (x0,y0) and (xn,yn) = dx/dy
3) Check m/slope value: m=1, m>1 or m<1, find next subsequent point coordinate values

Definition

Current point is pk then next point will be pk+1(x,y).

Compare new (x,y) coord value with last coord value. If they are same then stop

4) Then connect all intermediate points and plot those points across graph

The points generation using DDA Algorithm involves the following steps-

Step-01:

Calculate ΔX, ΔY and M from the given input.
These parameters are calculated as-
$\Delta X = X_n - X_0$
$\Delta Y = Y_n - Y_0$
$M = \Delta Y / \Delta X$

Parul® University

Definition

Step-02:

Find the number of steps or points in between the starting and ending coordinates.

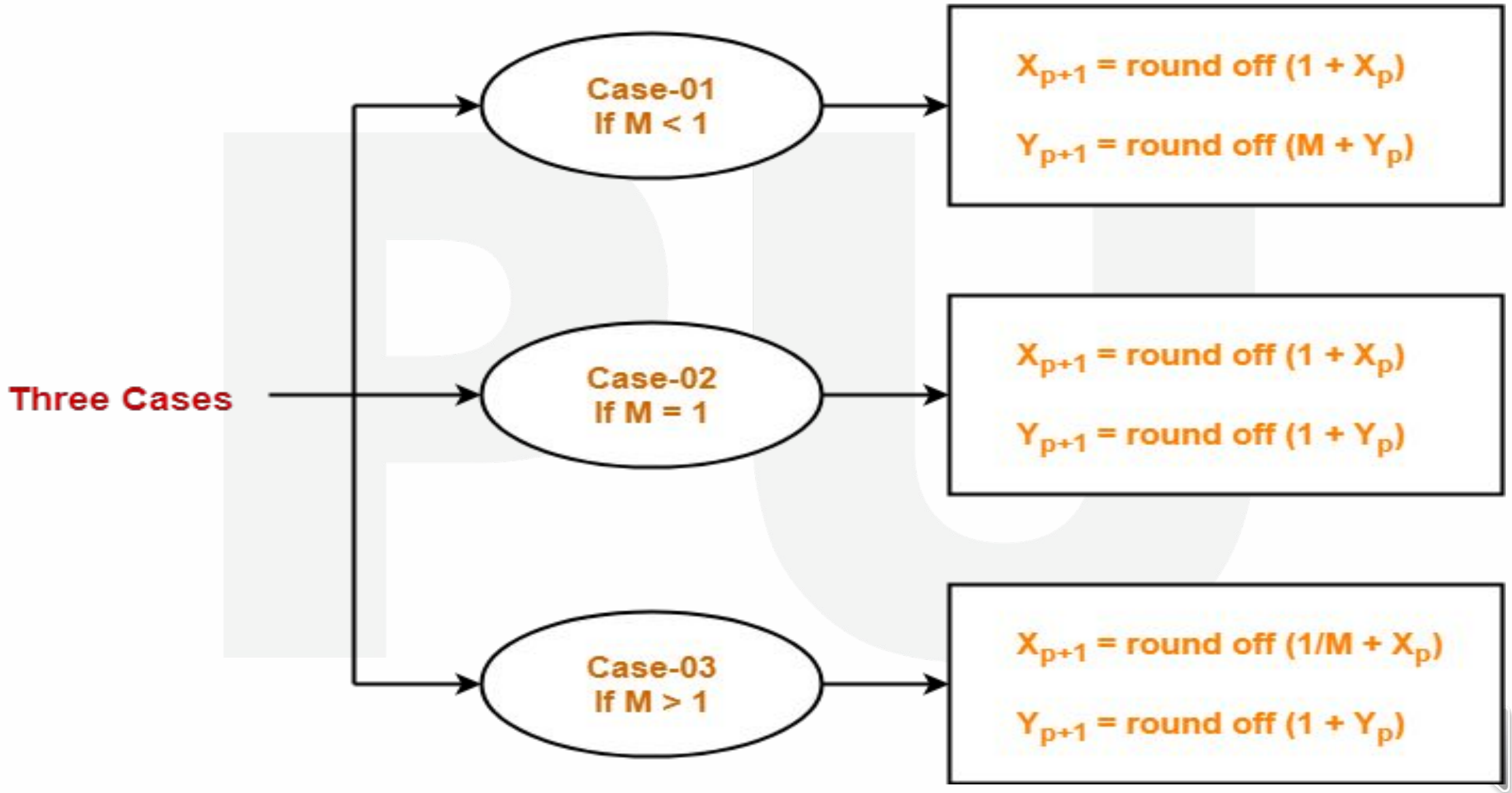
```
 if (absolute (ΔX) > absolute (ΔY))
Steps = absolute (ΔX);
else
Steps = absolute (ΔY);
```

Step-03:

Suppose the current point is (X_p, Y_p) and the next point is (X_{p+1}, Y_{p+1}).

Find the next point by following the below three cases-

Definition



X_{p+1} = round off $(1 + X_p)$

Y_{p+1} = round off $(M + Y_p)$

X_{p+1} = round off $(1 + X_p)$

Y_{p+1} = round off $(1 + Y_p)$

X_{p+1} = round off $(1/M + X_p)$

Y_{p+1} = round off $(1 + Y_p)$

Definition

Step-04:

Keep repeating Step-03 until the end point is reached or the number of generated new points (including the starting and ending points) equals to the steps count.

Parul® University

Advantages of DDA Algorithm-

The advantages of DDA Algorithm are-

It is a simple algorithm.

It is easy to implement.

It avoids using the multiplication operation which is costly in terms of time complexity.

Disadvantages of DDA Algorithm-

The disadvantages of DDA Algorithm are-

There is an extra overhead of using round off( ) function.

Using round off( ) function increases time complexity of the algorithm.

Resulted lines are not smooth because of round off( ) function.

The points generated by this algorithm are not accurate.

Mid Point Line Drawing Algorithm

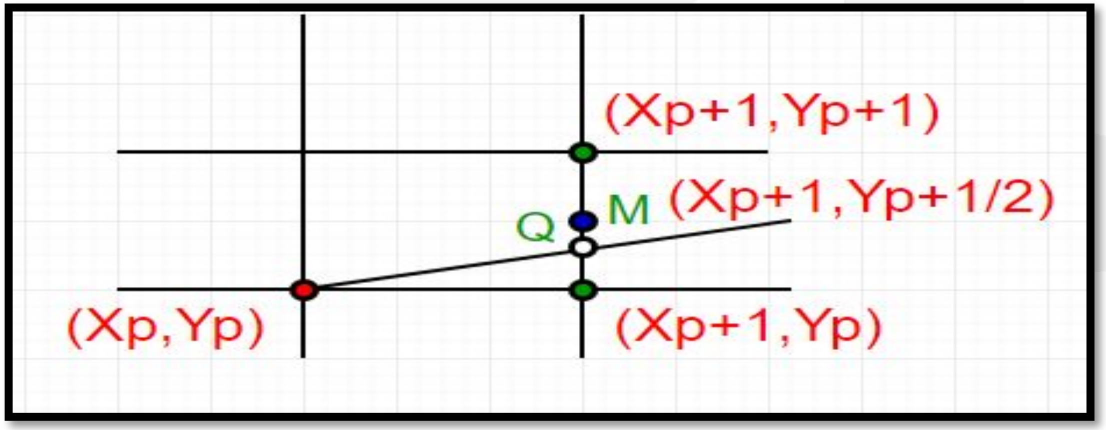
- Given coordinate of two points A(x1, y1) and B(x2, y2) such that x1 < x2 and y1 < y2. The task to find all the intermediate points required for drawing line AB on the computer screen of pixels. Note that every pixel has integer coordinates.
- For any given/calculated previous pixel $P(X_p,Y_p)$, there are two candidates for the next pixel closest to the line, $E(X_p+1, Y_p)$ and $NE(X_p+1, Y_p+1)$ (**E** stands for East and **NE** stands for North-East).

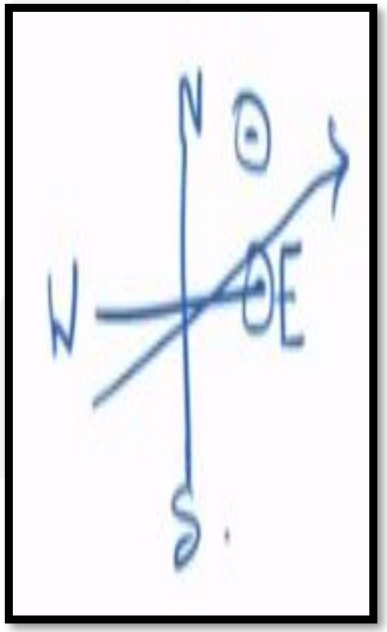
Parul® University

DIGITAL LEARNING CONTENT

Mid Point Line Drawing Algorithm

- In Mid-Point algorithm we do following.
- Find middle of two possible next points. Middle of E(X_p+1, Y_p) and NE(X_p+1, Y_p+1) is M(X_{p+1}, Y_p+1/2).
- If M is above the line, then choose E as next point.
- If M is below the line, then choose NE as next point.





Mid Point Line Drawing Algorithm

How to find if a point is above a line or below a line?

Below are some assumptions to keep algorithm simple.

We draw line from left to right.

x1 < x2 and y1< y2

Slope of the line is between 0 and 1. We draw a line from lower left to upper right.

Let us consider a line

1) y = mx + B. ,

2) Y=mx + C,

3) f(x,y)=ax+by+c ---------------(i)

Y=mx+B

Mid Point Line Drawing Algorithm

Slope= m = dy/dx

We can re-write the equation as :

y = (dy/dx)x + B or

(dy)x + B(dx) - y(dx) = 0

Let **F(x, y)** = (dy)x - y(dx) + B(dx) -----(1)

a=dy , b=-dx , c =b.(dx)

Mid Point Line Drawing Algorithm

Let we are given two end points of a line (under above assumptions)

)If at M, we have F(M) =0, which means all the points are on the line ie F(x,y)=0

2)If at M we get F(M)>0 , which means the line AB is ABOVE the midpoint then select (xp+1,yp+1)

3) If at M we get F(M)<0 , which means the line AB is BELOW the midpoint then select (xp+1,yp). Replacing the value of M in these equations will be time consuming so find the decision parameter

Mid Point Line Drawing Algorithm

Decision Parameter/Variable

d=F(x,y)

This relationship is used to determine the relative position of M

$$M = (X_{p+1}, Y_p+1/2)$$

So at M, our **decision parameter d** is,

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

$F(M)=a(X_p+1) + b(Y_p+1/2) + c$

We assign $d_{old}=d$

Based on the sign of d, we can select East(E) or northeast(NE)

Mid Point Line Drawing Algorithm

Case 1: If E is chosen then for next point : $f(x,y)=ax + by +c$

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

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

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

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

Difference (Or delta) of two distances:

Delta d = dnew – dold

$= a(X_p+2)- a(X_{p+1})+ b(Y_{p+1}/2)- b(Y_{p+1}/2)+ c-c$

$= a(X_p) +2a – a(X_p) – a$

Delta d= a =dy

Therefore, $(\text{Delta d})_E= a= dy$

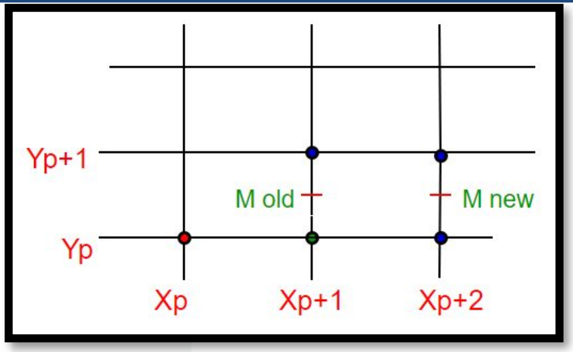
Mid Point Line Drawing Algorithm

Case 2: If NE is chosen then for next point :

dnew = $F(X_p+2, Y_p+3/2)$

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

dold = $a(X_{p+1}) + b(Y_{p+1}/2) + c$



Difference (Or delta) of two distances:

$(DELd)_{NE}$ = dnew -dold

$= a(X_p+2)- a(X_{p+1})+ b(Y_p+3/2)- b(Y_{p+1}/2)+ c-c$

$= a(X_p) + 2a – a(X_p) – a + b(Y_p) + 3/2b – b(Y_p) -1/2b$

$(DELd)_{NE} = a + b$

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

Mid Point Line Drawing Algorithm

To find out the initial value of the decision parameter (d_{start}), let's consider the starting point as (X0,Y0), so M=(X0+1,Y0+1/2)

Calculation For initial value of decision parameter d0:

d0 = F(X0+1 , Y0+1/2)

= a(X0 + 1) + b(Y0 + 1/2) +c

= aX0+ bY0 + c + a + b/2

= F(X0,Y0) + a + b/2

Mid Point Line Drawing Algorithm

= a + b/2 (as F(X0, Y0) = 0 ie points lie on the line)

d0 = dy – dx/2. (as a = dy, b = -dx)

Now to remove the fraction part , we need to multiply all equations by 2, therefore

(d)start=2dy-dx

(Delta d)$_E$ =2dy

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

Advantages of Mid Point Line Drawing Algorithm

The advantages of Mid-Point Line Drawing Algorithm are-

Accuracy of finding points is a key feature of this algorithm.

It is simple to implement.

It uses basic arithmetic operations.

It takes less time for computation.

The resulted line is smooth as compared to other line drawing algorithms.

Disadvantages of Mid Point Line Drawing_Algorithm

- The disadvantages of Mid-Point Line Drawing Algorithm are-
- This algorithm may not be an ideal choice for complex graphics and images.
- In terms of accuracy of finding points, improvement is still needed.
- There is no any remarkable improvement made by this algorithm.

Bresenhems Line Drawing Algorithm

- The Bresenham algorithm is another incremental scan conversion algorithm.
- The big advantage of this algorithm is that, it uses only integer calculations.
- Moving across the x axis in unit intervals and at each step choose between two different y coordinates.

Parul® University

DIGITAL LEARNING CONTENT

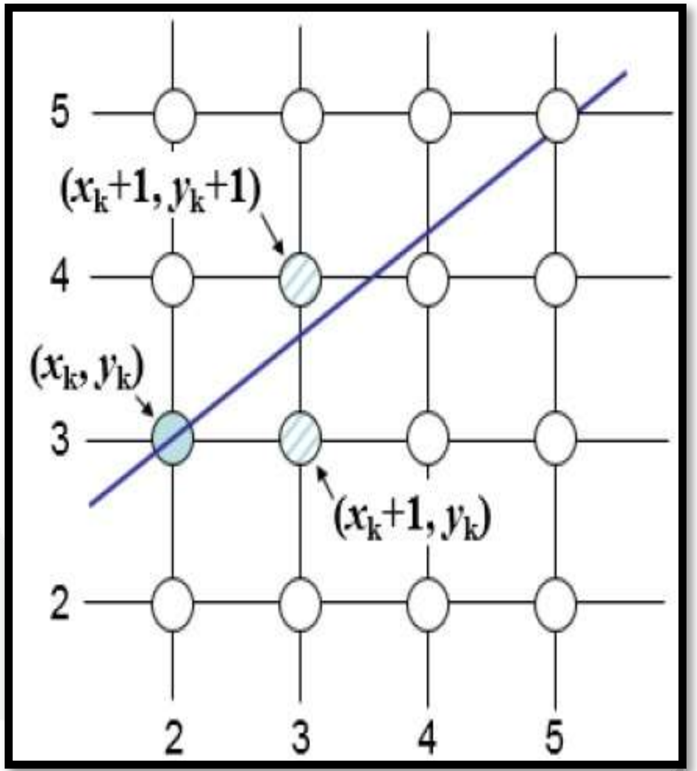
Bresenhems Line Drawing Algorithm

Given-

Starting coordinates = (X_0, Y_0)

Ending coordinates = (X_n, Y_n)

The points generation using Bresenham Line Drawing Algorithm involves the following steps-



Bresenhems Line Drawing Algorithm

Step-01:

Calculate ΔX and ΔY from the given input.

These parameters are calculated as-

$\Delta X = X_n - X_0$

$\Delta Y = Y_n - Y_0$

Step-02:

Calculate the decision parameter P_k.
It is calculated as-
$P_k = 2\Delta Y - \Delta X$

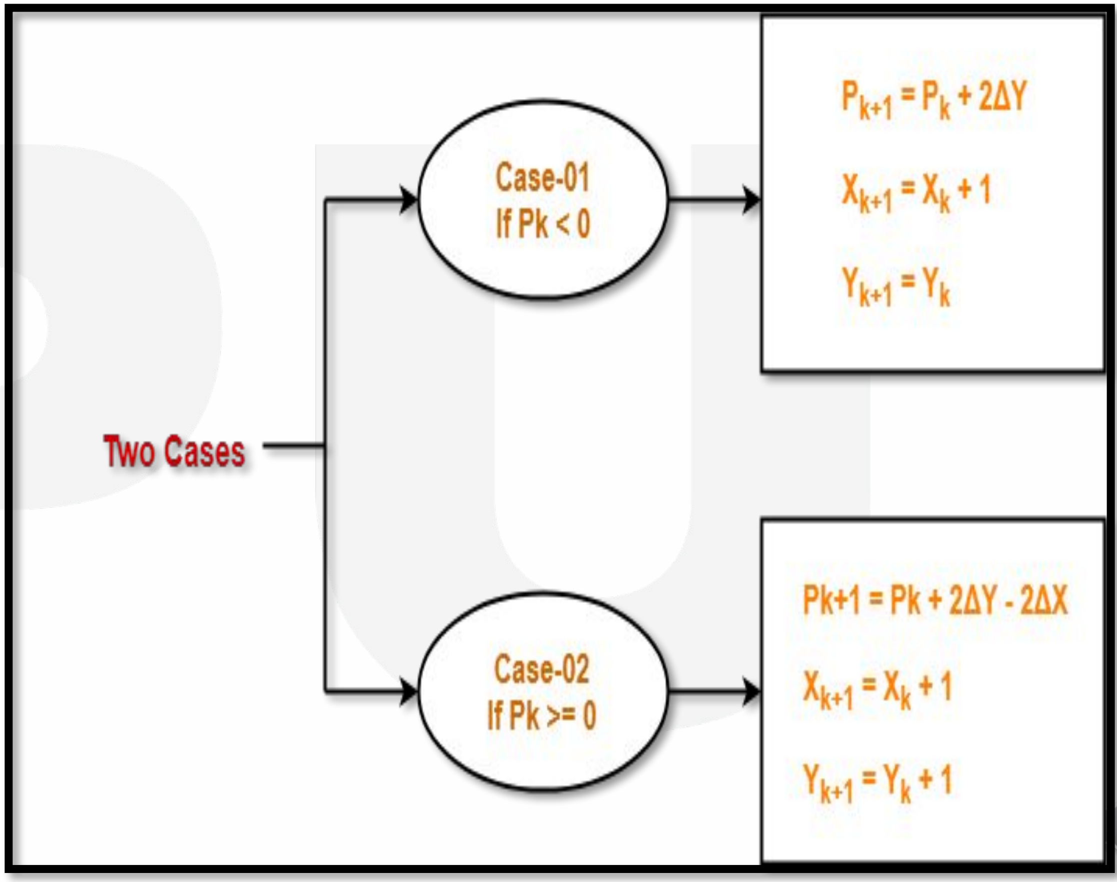
Bresenhems Line Drawing Algorithm

Step-03:

Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).

Find the next point depending on the value of decision parameter P_k.

Follow the below two cases-



Bresenhems Line Drawing Algorithm

Step-04:

Keep repeating Step-03 until the end point is reached or number of iterations equals to (ΔX-1) times.

Advantages of Bresenhems Line Drawing Algorithm

The advantages of Bresenham Line Drawing Algorithm are-

It is easy to implement.

It is fast and incremental.

It executes fast but less faster than DDA Algorithm.

The points generated by this algorithm are more accurate than DDA Algorithm.

It uses fixed points only.

Dis-advantages of Bresenhems Line Drawing Algorithm

The disadvantages of Bresenham Line Drawing Algorithm are-

Though it improves the accuracy of generated points but still the resulted line is not smooth.

This algorithm is for the basic line drawing.

Circle Drawing Algorithm

- Drawing a circle on the screen is a little complex than drawing a line.
- There are two popular algorithms for generating a circle − **Bresenham's Algorithm** and **Midpoint Circle Algorithm**.
- These algorithms are based on the idea of determining the subsequent points required to draw the circle
- The equation of circle is X2+Y2=r2, where r is radius.

Mid Point Circle Drawing Algorithm

Given-

Centre point of Circle = (X_0, Y_0)

Radius of Circle = R

The points generation using Mid Point Circle Drawing Algorithm involves the following steps-

Step-01:

Assign the starting point coordinates (X_0, Y_0) as-

$X_0 = 0$

$Y_0 = R$

Mid Point Circle Drawing Algorithm

Step-02:

Calculate the value of initial decision parameter P_0 as-

$P_0 = 1 – R$

Step-03:

Suppose the current point is (X_k, Y_k) and the next point is (X_{k+1}, Y_{k+1}).

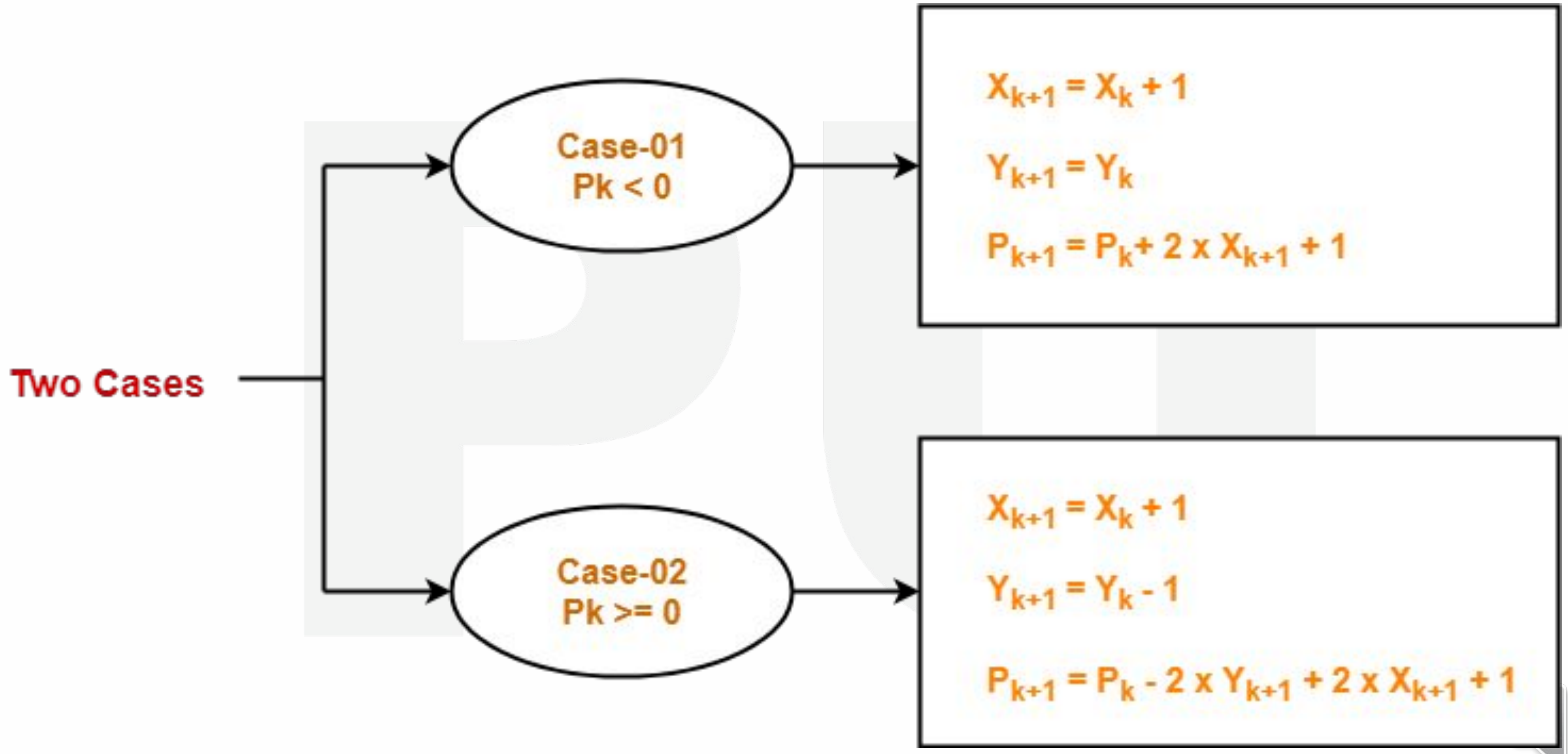
Find the next point of the first octant depending on the value of decision parameter P_k.

Follow the below two cases-

Parul® University

DIGITAL LEARNING CONTENT

Mid Point Circle Drawing Algorithm



$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k$

$P_{k+1} = P_k + 2 \times X_{k+1} + 1$

$X_{k+1} = X_k + 1$

$Y_{k+1} = Y_k - 1$

$P_{k+1} = P_k - 2 \times Y_{k+1} + 2 \times X_{k+1} + 1$

Mid Point Circle Drawing Algorithm

Step-04:

If the given centre point (X_0, Y_0) is not (0, 0), then do the following and plot the point-

$X_{plot} = X_c + X_0$

$Y_{plot} = Y_c + Y_0$

Here, (X_c, Y_c) denotes the current value of X and Y coordinates.

Step-05:

Keep repeating Step-03 and Step-04 until $X_{plot} >= Y_{plot}$.

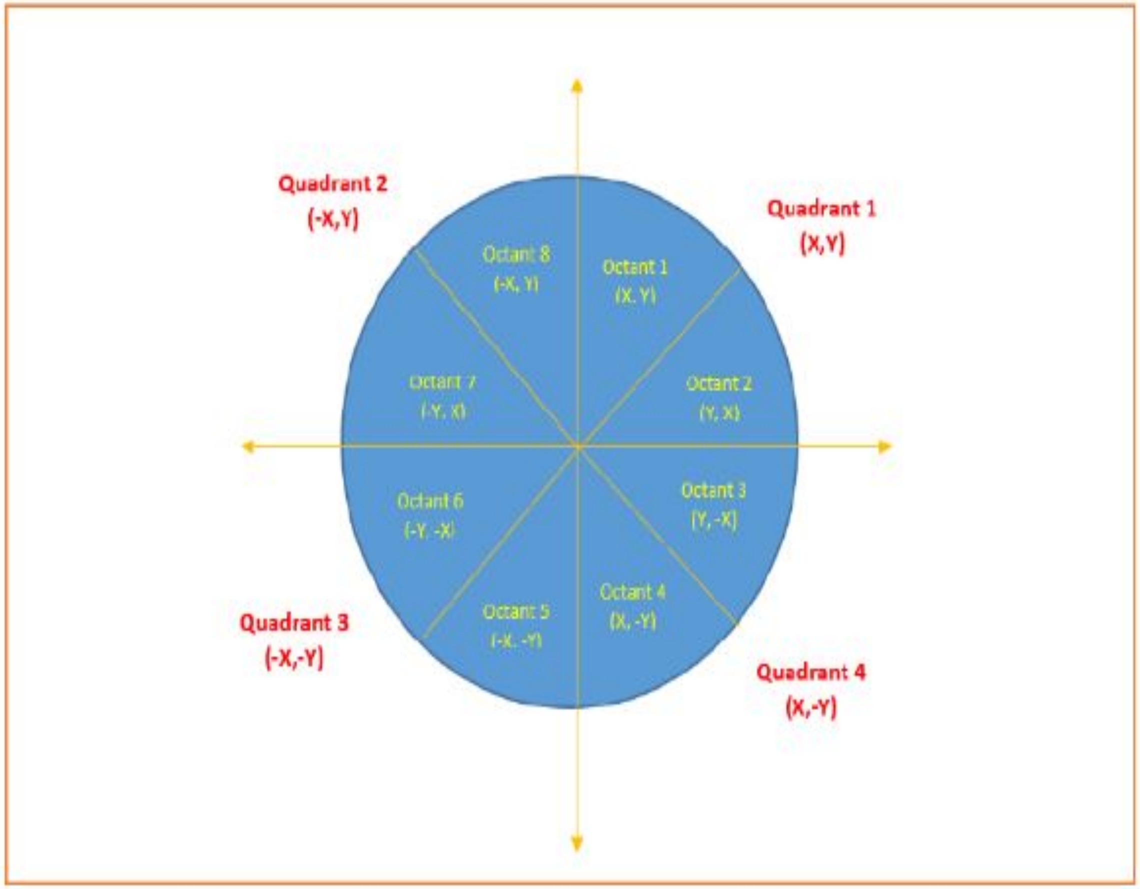
Mid Point Circle Drawing Algorithm

Step-06:

Step-05 generates all the points for one octant.

To find the points for other seven octants, follow the eight symmetry property of circle.

This is depicted by the following figure-



Advantages of Mid Point Circle Drawing Algorithm

The advantages of Mid Point Circle Drawing Algorithm are-

It is a powerful and efficient algorithm.

The entire algorithm is based on the simple equation of circle $X^2 + Y^2 = R^2$.

It is easy to implement from the programmer's perspective.

This algorithm is used to generate curves on raster displays.

Disadvantages of Mid Point Circle Drawing Algorithm

The disadvantages of Mid Point Circle Drawing Algorithm are-

Accuracy of the generating points is an issue in this algorithm.

The circle generated by this algorithm is not smooth.

This algorithm is time consuming.

Polygon Filling Algorithm

1. Scan line Fill Algorithm
2. Seed Fill Algorithm

   -Boundary Fill Algorithm

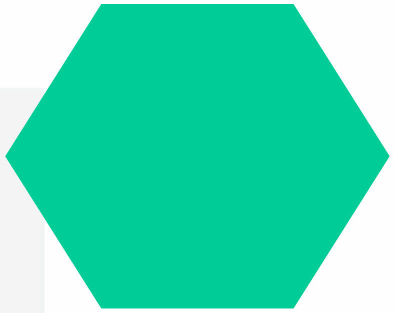
   -Flood Fill Algorithm

Polygon Filling Algorithm

Types of filling

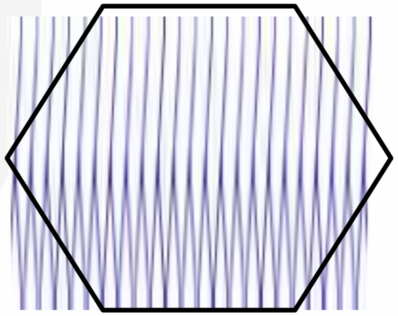
- **Solid-fill**

All the pixels inside the polygon's boundary are illuminated.



- **Pattern-fill**

The polygon is filled with an arbitrary predefined pattern.



Polygon Representation

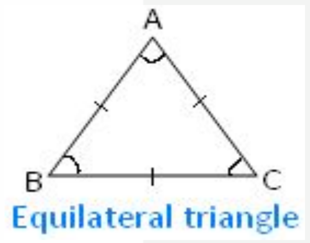
The polygon can be represented by listing its n vertices in an ordered list.

$$P = \{(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_n, y_n)\}.$$

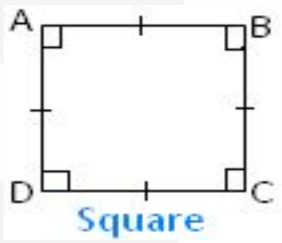
The polygon can be displayed by drawing a line between (x_1, y_1), and (x_2, y_2), then a line between (x_2, y_2), and (x_3, y_3), and so on until the end vertex. In order to close up the polygon, a line between (x_n, y_n), and (x_1, y_1) must be drawn.

Types of Polygons: Regular and Irregular

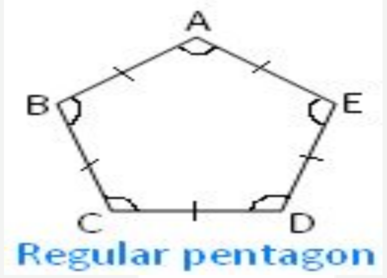
Regular Polygon :A polygon which has all its sides of equal length and all its angles of equal measures is called a **regular polygon**.



Equilateral triangle

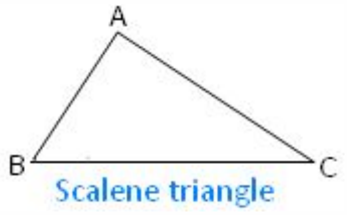


Square



Regular pentagon

Irregular Polygon :A polygon which has all its sides of unequal length and all its angles of unequal measures is called an irregular polygon.



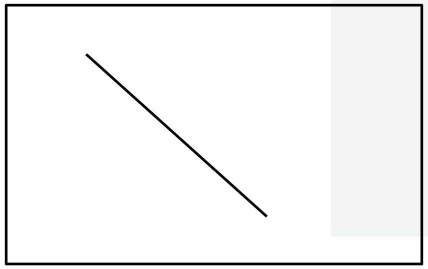
Scalene triangle

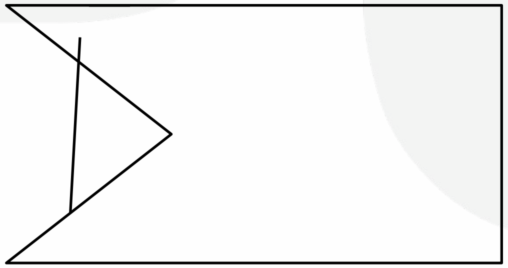
Types of Polygons: Regular and Irregular

Convex Polygon - For any two points P_1, P_2 inside the polygon, all points on the line segment which connects P_1 and P_2 are inside the polygon.

Concave Polygon

- Aline segment connecting any two points may or may not lie inside the polygon.ie A polygon which is not convex.


Convex Polygon

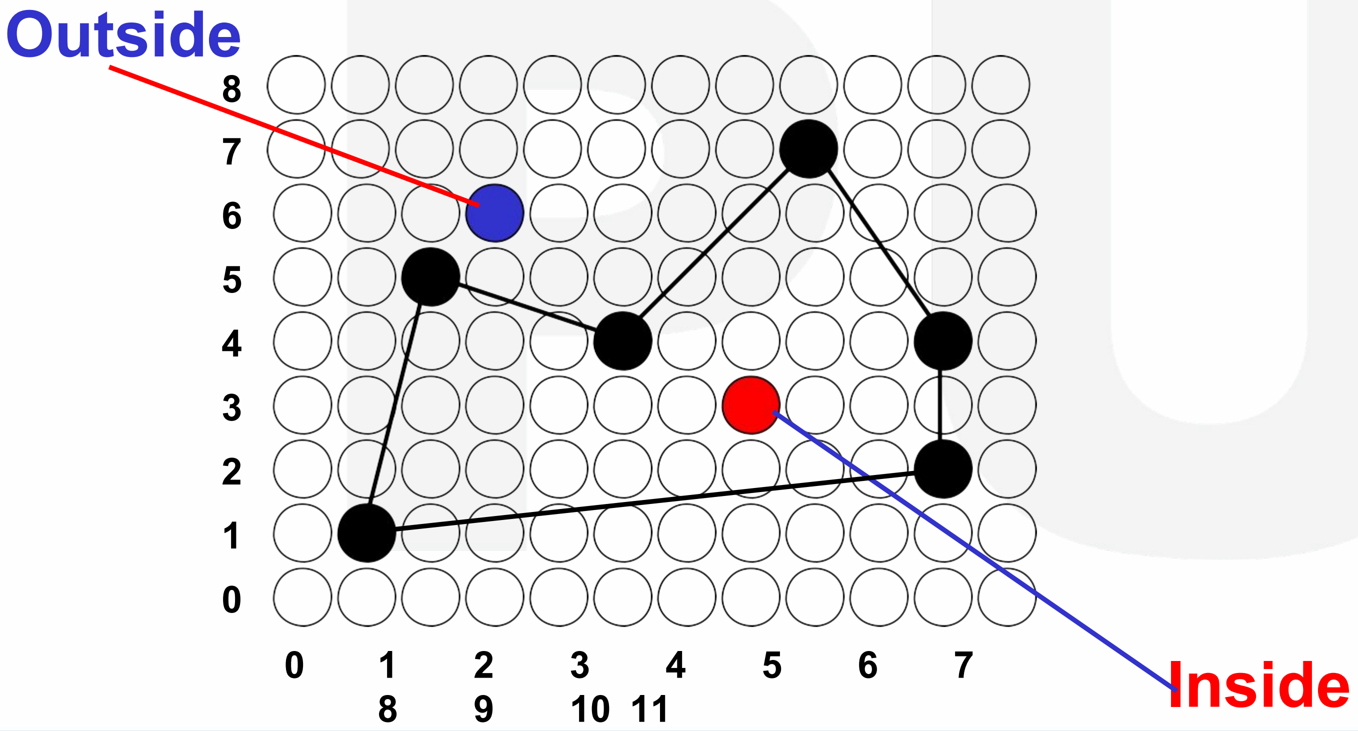

Concave Polygon

Inside-Outside Tests

- when filling polygons we should decide whether a particular point is interior or exterior to a polygon.
- A rule called the **odd-parity** (or the **odd-even rule**) is applied to test whether a point is interior or not.
- To apply this rule, we conceptually draw a line starting from the particular point and extending to a distance point outside the coordinate extends of the object in any direction such that **no polygon vertex intersects** with **the line**.

Inside-Outside Tests

The point is considered to be **interior** if the number of intersections between the line and the polygon edges is **odd**. Otherwise, The point is exterior point.

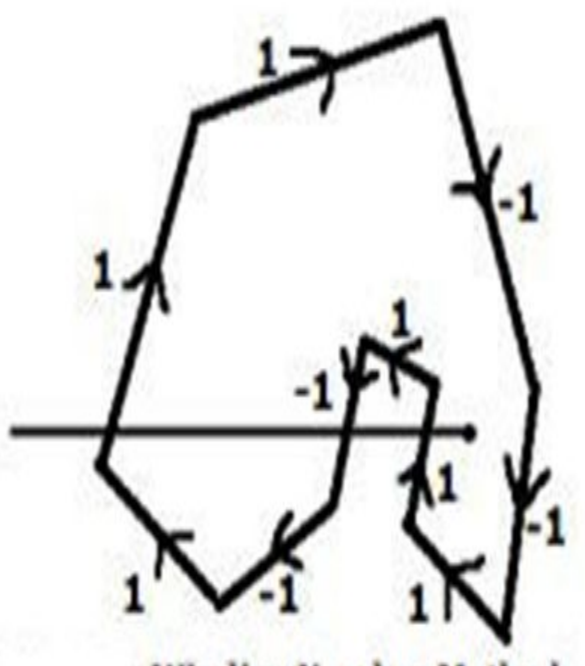


WINDING NUMBER METHOD FOR INSIDE OUT SIDE

If the winding number is **nonzero** then *P* is defined to be an **interior** point **Else** *P* is taken to be an **exterior** point**.**

1. **Sum of edge is nonzero means Inside the polygon**

2. **Sum of edge is zero means outside the polygon**

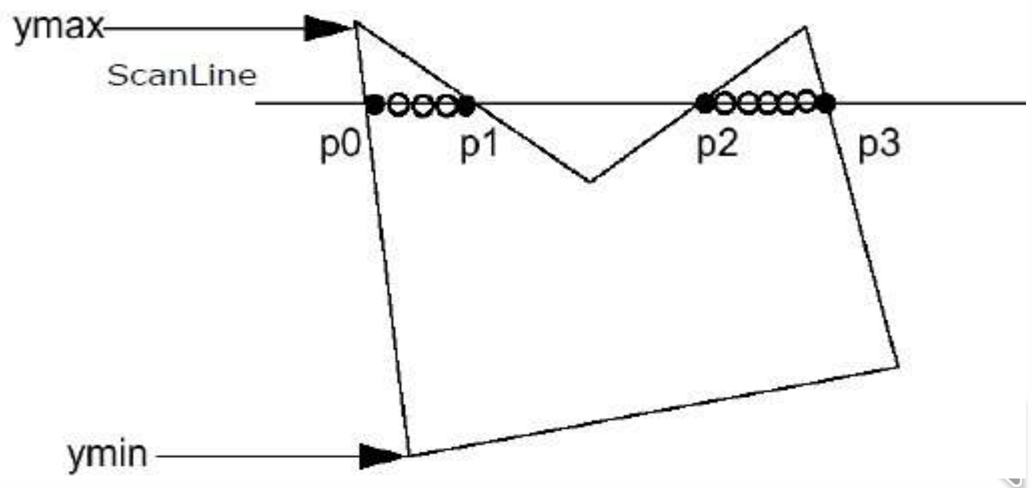


Winding Number Method

The Scan-Line Polygon Fill Algorithm

This algorithm works by intersecting scanline with polygon edges and fills the polygon between pairs of intersections. The following steps depict how this algorithm works.

Step 1 – Find out the Ymin and Ymax from the given polygon.



Scan Line Drawing Algorithm

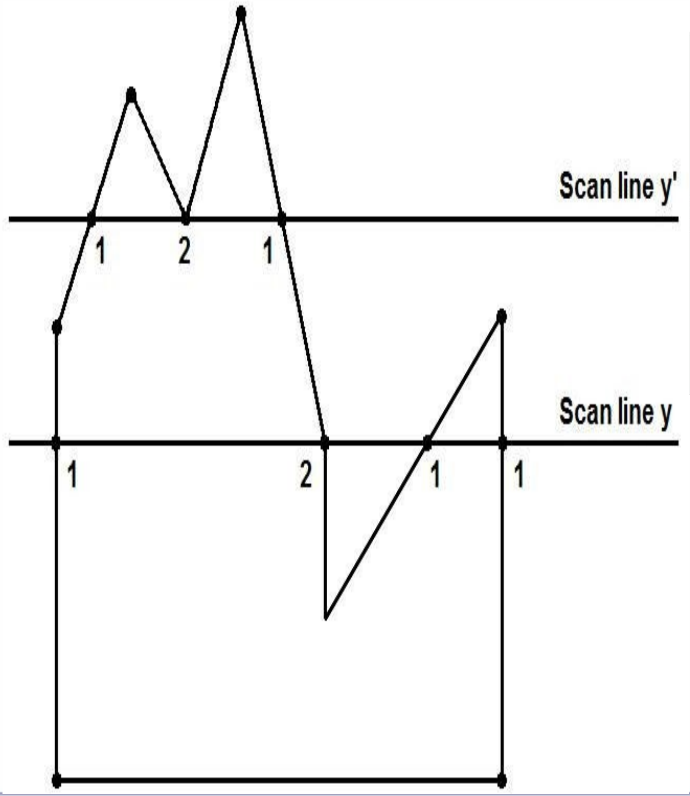
- **Step 2** – ScanLine intersects with each edge of the polygon from Ymin to Ymax. Name each intersection point of the polygon. As per the figure shown above, they are named as p0, p1, p2, p3.

- **Step 3** – Sort the intersection point in the increasing order of X coordinate i.e. (p0, p1), (p1, p2), and (p2, p3).

- **Step 4** – Fill all those pair of coordinates that are inside polygons and ignore the alternate pairs.

The Scan-Line Polygon Fill Algorithm

Dealing with Vertices



If the edges having a common intersection point lies on the same side of the scan line then the intersection point is considered two.

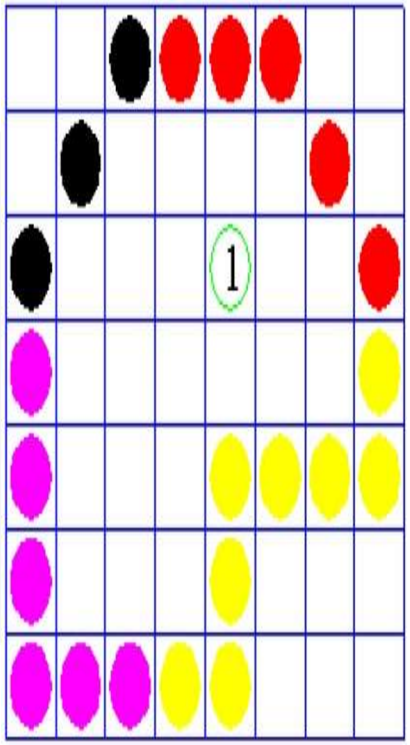
But if the edges having a common intersection point lies on the different side of the scan line then the intersection point is considered one.

Here please count number of intersection 1.

Problem:
Calulating the Intersection Points is a slow process.

Flood Fill Algorithm

- Sometimes we come across an object where we want to fill the area and its boundary with different colors.
- In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original interior color exist, the algorithm is completed.
- This algorithm relies on the Four-connect or Eight-connect method of filling in the pixels.
- But instead of looking for the boundary color, it is looking for all adjacent pixels that are a part of the interior.



Boundary Fill Algorithm

- Another approach to area filling is to start at a point inside a region and paint the interior outward toward the boundary.
- If the boundary is specified in a single color, the fill algorithm processed outward pixel by pixel until the boundary color is encountered.
- A boundary-fill procedure accepts as input the coordinate of the interior **point (x,y)**, a **fill color**, and a **boundary color**.

Boundary Fill Algorithm

The following steps illustrate the **recursive** boundary-fill algorithm:

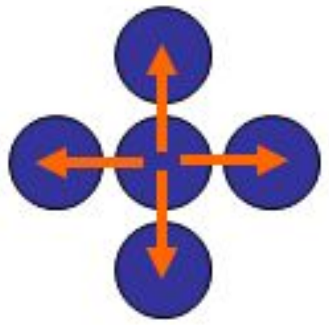
1)Start from an interior point.

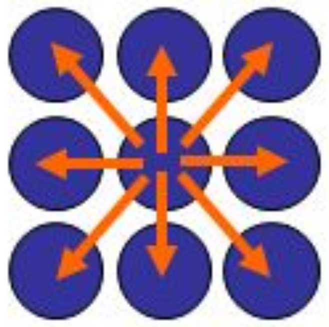
2) If the current pixel is **not already** filled and if it is not an edge point, then set the pixel with the fill color, and store its neighboring pixels (**4** or **8- connected**) in the stack for processing. Store only neighboring pixel that is **not already** filled and is not an edge point.

3)Select the next pixel from the stack, and continue with step **2**.

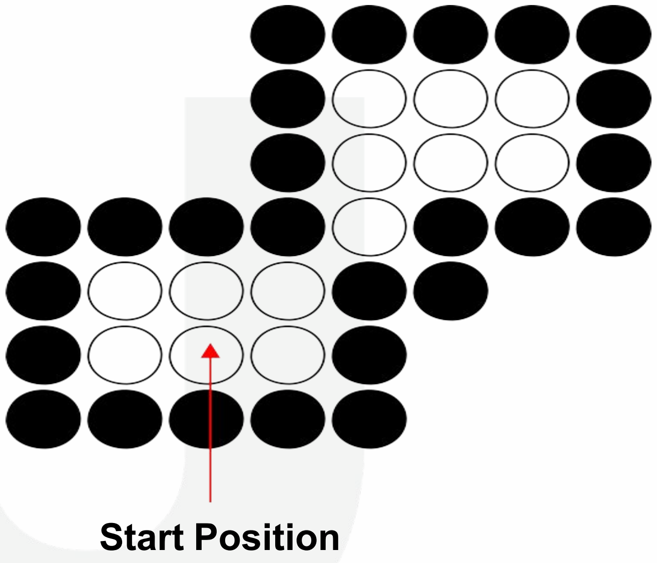
Boundary Fill Algorithm

The order of pixels that should be added to stack using **4-connected** is above, below, left, and right. For **8-connected** is above, below, left, right, above- left, above-right, below-left, and below-right.

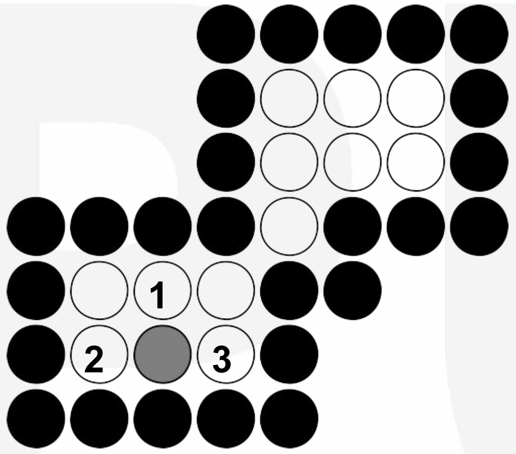




Boundary Fill Algorithm : 4-connected (Example)



Boundary Fill Algorithm : 4-connected (Example)

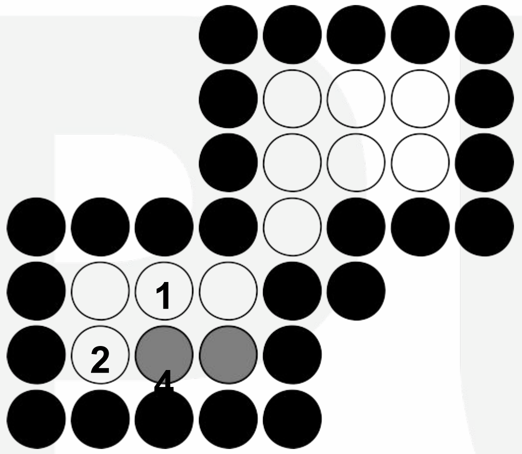


3

2

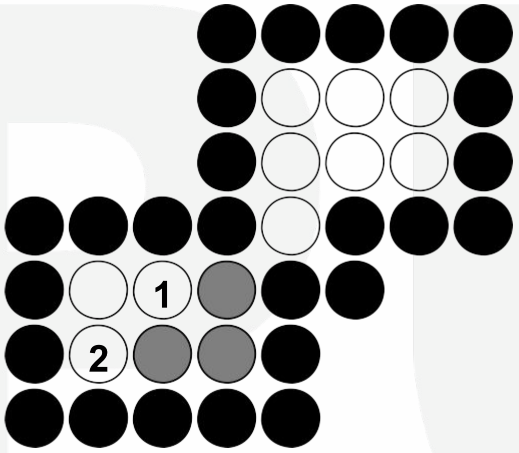
1

Boundary Fill Algorithm : 4-connected (Example)



4
2
1

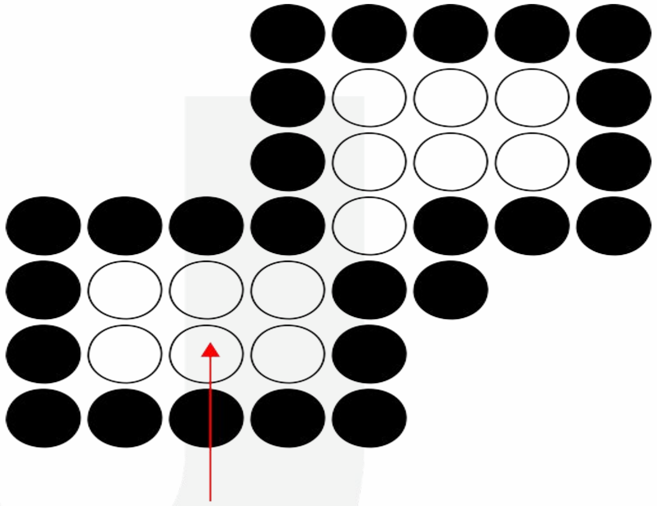
Boundary Fill Algorithm : 4-connected (Example)



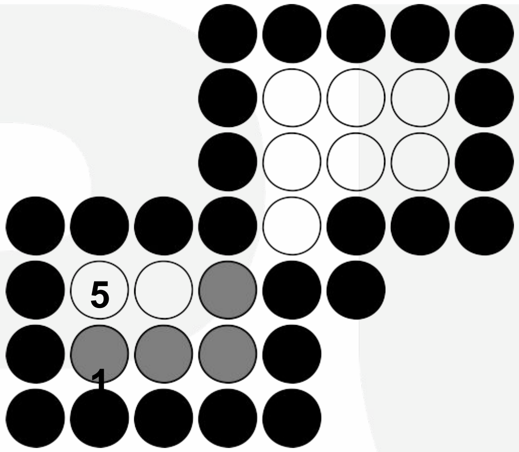
2

1

Boundary Fill Algorithm : 4-connected (Example)



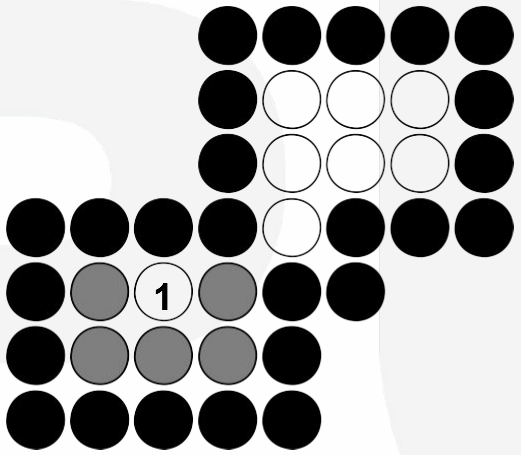
Boundary Fill Algorithm : 4-connected (Example)



5

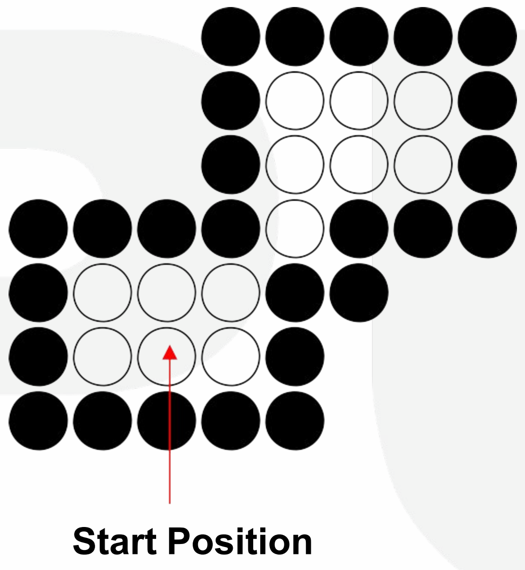
1

Boundary Fill Algorithm : 4-connected (Example)

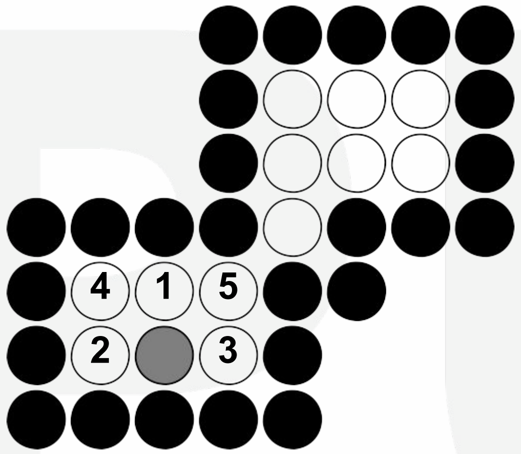


1

Boundary Fill Algorithm(8-connected Example)



Boundary Fill Algorithm(8-connected Example)



5

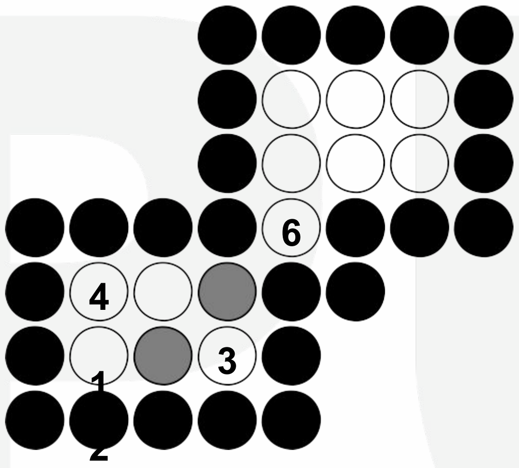
4

3

2

1

Boundary Fill Algorithm(8-connected Example)



6

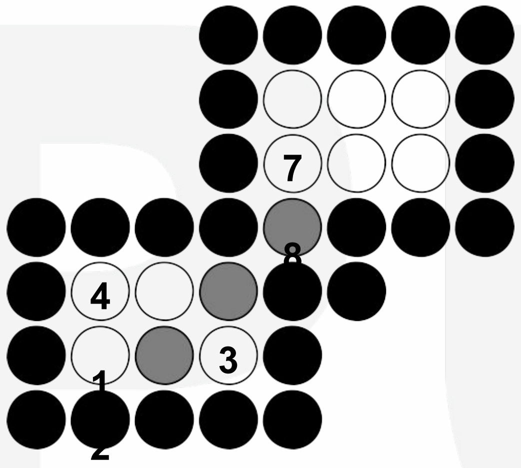
4

3

2

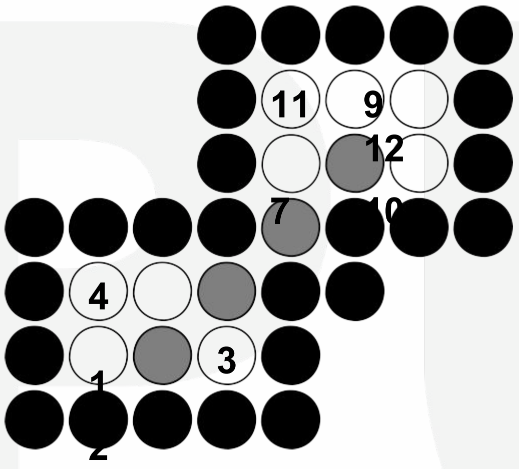
1

Boundary Fill Algorithm(8-connected Example)



8
7
4
3
2
1

Boundary Fill Algorithm(8-connected Example)



12

11

10

9

7

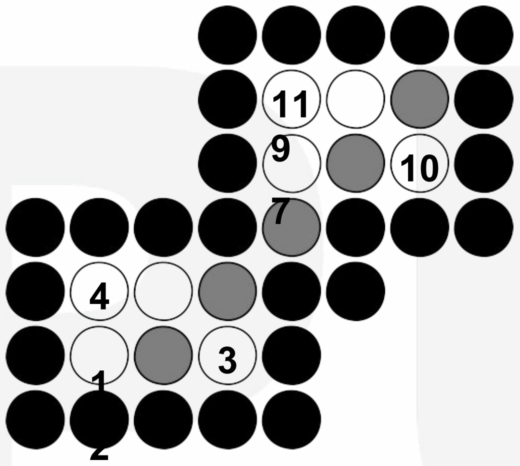
4

3

2

1

Boundary Fill Algorithm(8-connected Example)



11

10

9

7

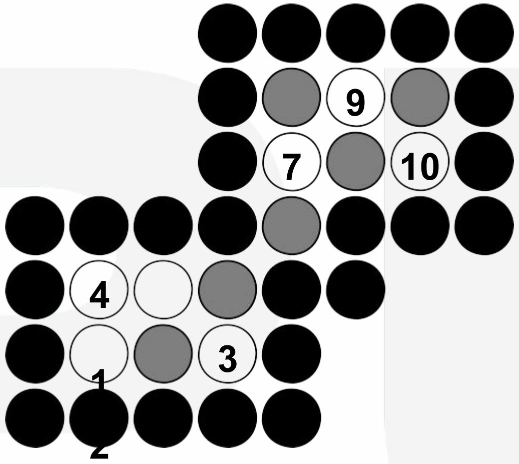
4

3

2

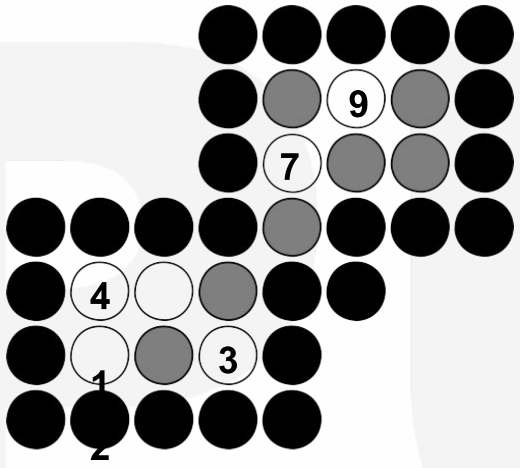
1

Boundary Fill Algorithm(8-connected Example)



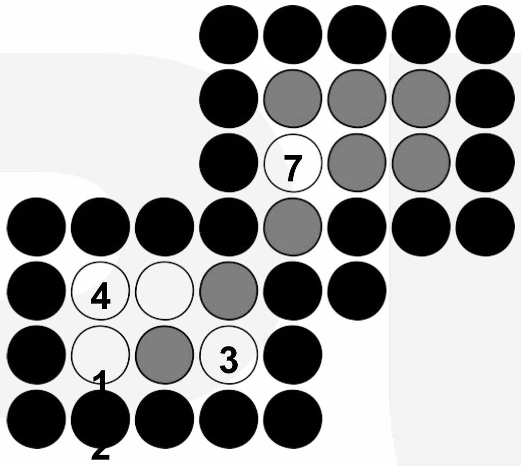
10
9
7
4
3
2
1

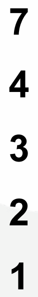
Boundary Fill Algorithm(8-connected Example)



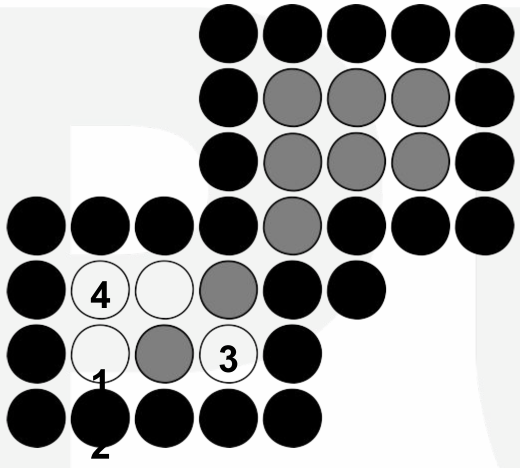
9
7
4
3
2
1

Boundary Fill Algorithm(8-connected Example)





Boundary Fill Algorithm(8-connected Example)



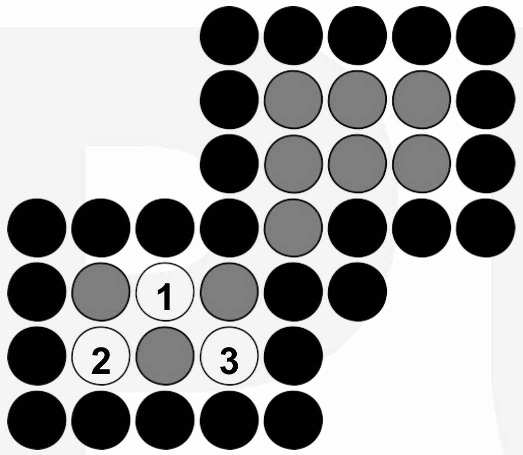
4

3

2

1

Boundary Fill Algorithm(8-connected Example)



3

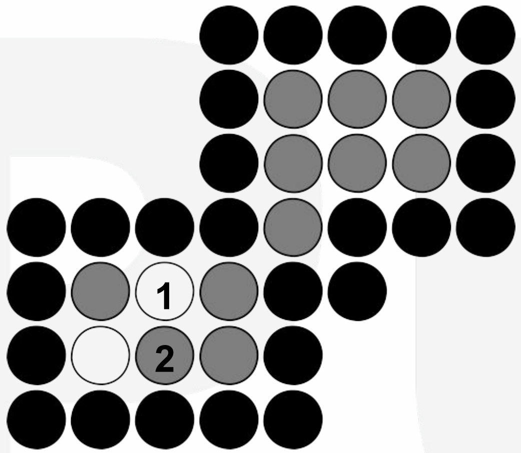
2

1

Parul® University

DIGITAL LEARNING CONTENT

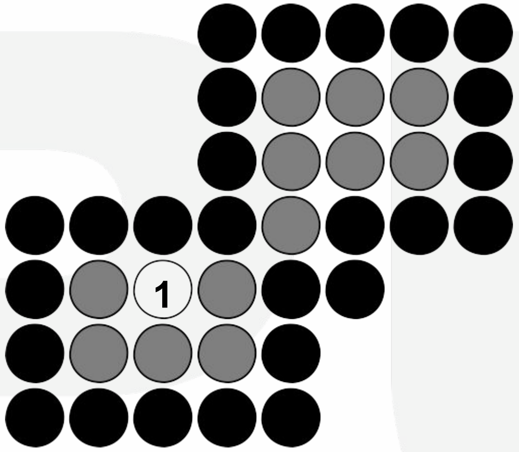
Boundary Fill Algorithm(8-connected Example)



2

1

Boundary Fill Algorithm(8-connected Example)

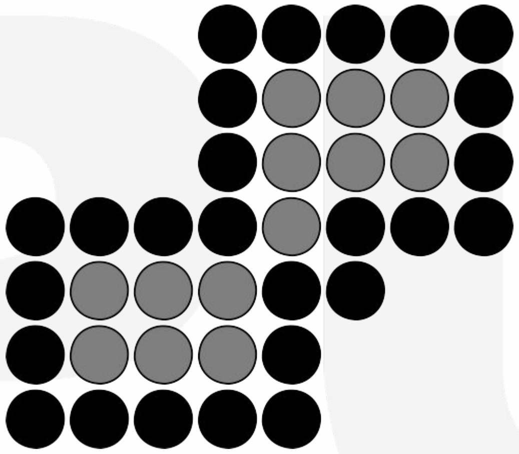


1

Parul® University

DIGITAL LEARNING CONTENT

Boundary Fill Algorithm(8-connected Example)



Boundary Fill Algorithm

In boundary fill algorithm we start from a seed pixel and start fill the color until boundary color does not encountered. In this algorithm if color of seed pixel is already fill color then we have to make this pixel as background color and then start fill color.

Functions used in this algorithm

Getpixel( )- This function is used to fetch color of a pixel on the screen.

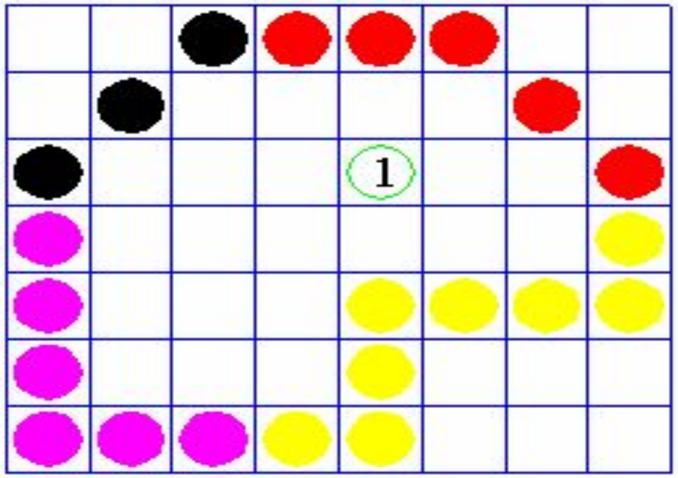
Putpixel( ) – To give color on a particular pixel.

Flood Fill Algorithm

Sometimes we want to fill in (recolor) an area that is not defined within a single color boundary.

We paint such areas by replacing a specified interior color instead of searching for a boundary color value.

This approach is called a **flood-fill algorithm**.



Flood Fill Algorithm

We start from a specified interior pixel (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color.

If the area has **more than one** interior color, we can first **reassign pixel values** so that all interior pixels have the same color.

Using either **4-connected** or **8-connected** approach, we then step through pixel positions until all interior pixels have been repainted.

In this algorithm we match color of a pixel with background color of the object. We stop color fill until we does not find a pixel where color is not same as background color.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in