UNIT 5: FILLED AREA ALGORIHMS

UNIT STRUCTURE

5.1 Learning Objectives

5.2 Introduction

5.3 Scan line polygon fill algorithm

5.3.1 Global edge table

5.3.2 Active edge table

5.4 Inside outside test

5.4.1 Odd-Even rule

5.4.2 Non-Zero winding number rule

5.5 Boundary fill

5.5.1 4-Connected area fill algorithm

5.5.2 8-Connected area fill algorithm

5.6 Flood fill algorithm

5.7 Let Us Sum Up

5.8 Further Reading

5.9 Answer to Check Your Progress

5.10 Model Questions

5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- How to fill an area of an object.
- What is area filled algorithm.
- How a scan line is used in area filled algorithm.
- Scan line polygon filling algorithm.
- How to detect a point whether it is an interior or exterior to the polygon.
- Learn the working principle of the odd-even rule used to identify interior points of a polygon.
- Working principle of winding number rule used in the inside-outside test.

- How does the boundary fill algorithm fill the interior points of a polygon with a specified colour?
- Working principle of the 4-connected and 8-connected approach used in the boundary-filled algorithm.
- Drawbacks of the boundary fill algorithm.
- Flood fill algorithm.
- How flood fill algorithm overcomes the drawbacks of the boundary fill algorithm.

5.2 INTRODUCTION

Output primitives of most of the graphics packages consist of solid colour or patterned polygon areas. Due to the linear boundaries of a polygon, polygon area primitives are easier than other alternatives. Filling a polygonal area can be done by calculating the overlap intervals of scan lines that traverse it. Another technique to fill an area is to start at a specific place within a defined interior region and fill colour to the surrounding pixels until a specified boundary line or conditions are found. The scan line method is used to fill simple regions such as circles, rectangles, polygons, ellipses, and other simple curve areas in basic graphics software. Sophisticated graphics packages and interactive painting systems commonly use the fill method from an interior location to determine more complex borders.

5.3 SCAN-LINE POLYGON FILL ALGORITHM

Scan-line polygon filling algorithm is used for the solid filling of polygon areas. Figure 1 shows some scan lines passing through a polygon area. Each horizontal scan line intersects polygon edges at some pixel coordinates. The polygon fill algorithm locates the intersecting points of the scan lines that make with the polygon edges. Considering the scan lines from top to bottom order, intersecting point coordinates are arranged in ascending order concerning the x coordinates i.e. from left to right and make pair of intersecting points.

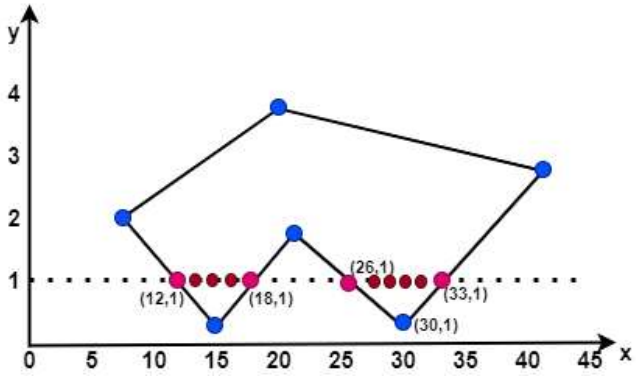


Figure 1Interiorpixels along a scan line passing through polygon edges

The corresponding frame buffer positions between the pairs will be filled with specified colours. Figure 1 shows a dotted scan line across the polygon area. The scan line intersects the polygon edges at coordinates (12, 1), (18,1), (26, 1), and (33, 1), which are arranged in sorted order on the value of the x-coordinate. The pair of consecutive coordinates are (12, 1), (18, 1), and (26, 1), (33, 1).

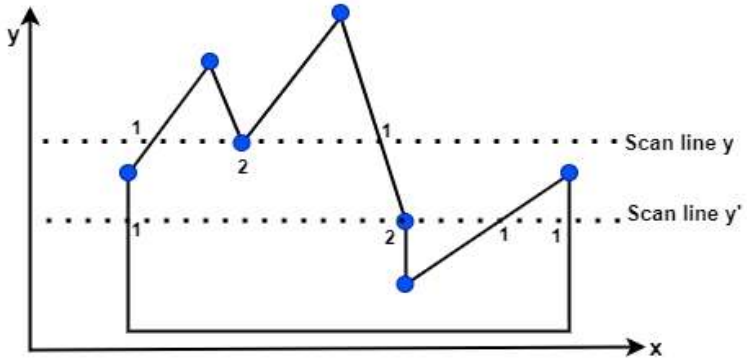


Figure 1 Intersection points at polygon edgeswith common vertices

The pixel positions between the pair coordinates are now can be filled with solid colours or any specified colour. A scan line as shown in figure 1 can identify the interior pixels correctly. But for some cases, it is required to perform some additional processing is to determine the correct interior points. In the following figure 2, the scan lines y and y' pass through the polygon edges. For the scan line y', two intersecting edges share a common vertex and the polygon edges are in opposite directions. For the scan line y, two intersecting edges share a common point, where both the edges are lies in the same direction, i.e. both are above the scan line. For the common

vertex of the polygon edges, the pairing of pixel points will not work properly, and thus scan line filling cannot be completed successfully. Thus some additional processing steps need to be performed for those that have connecting edges on the opposite side of the scan line. The existence of the connecting edges on opposite sides of a scan line can be identified by traversing the polygon boundary in a clockwise or anticlockwise direction. By observing the y coordinate of each endpoint of polygon edges, these connected vertices can be identified. If the y coordinate of two consecutive endpoints is monotonically decreased or increased, then the middle vertex needs to be counted as a single intersection point for any scan line passing through it. When the scan line found an intersecting point it starts its counting and it will be continued up to the last intersecting point. For odd count, the polygon fill algorithm starts to fill the pixels until the count becomes an even number. From the even count to the next odd count, all pixels will be considered exterior points. In the connecting point where both the polygon edges are lies in the same direction, the count will be considered two times consecutively. In figure 2, the scan liney intersects the first polygon edge at point 1 and set the counter value 1. The algorithm starts to fill the colour across the scan line up to the next intersecting point. The next intersecting point 2 in the scan line is a connecting point where two polygon edges are connected and lie on the same side of the scan line. So the counter will increase its value twice consecutively, i.e. for the counter value 2, the algorithm will stop its colour fill activity. But for the next increment of the counter value is an odd number, so the polygon fill will start its colour to the next even number of the counter. Thus the polygon fill algorithm overcomes the drawbacks and fills the interior points of the polygon with the specified colour. The scan line y passes through a connecting vertex, where both the connecting edges are lies in the opposite direction. So at this point 4, the counting will be considered once.

Using the concept of a sorted edge table, the performance of the Polygon fill algorithm can increase. The sorted edge table contains all the information about a scan line required to perform the fill activity including the polygon boundary. During the traversing of the polygon edges, clockwise

or anticlockwise direction, the bucket sort algorithm is used to store the y coordinate.

5.3.1 Global Edge Table

To compute the scan line intersection points made with the polygon edges two data structures namely - Global Edge table (GET) and Active Edge Table (AET) are used. The GET consists of an array of sizes Y_{min} to Y_{max}. The first entry of the array is pointed to the node with Y_{min} as shown in figure 4.

- GET Contains all edges, sorted by Y coordinates in ascending order, i.e. Y_{min} to Y_{max}.
- An array of a bucket is maintained to keep the track of each scan line in the polygon.
- Each bucket has to be maintained as a linked list with increasing X coordinates order where X is the X value at the Y_{min} of the scan line.
- Each entry contains Y_{max}, X_{min} (X at Y_{min}), and 1/m. Where m is the slope of the polygon edge.

Y_{max}	X_{min}(at Y_{min})	1/m

5.3.2 Active Edge Table

Active Edge Table (AET) gives the final intersecting points made by the scan line with the polygon edges with help of information maintained in the Global Edge Table. Active Edge Table starts with a NULL entry. The linked node with the Y_{min} will be added to the AET from the GET.

- Y=minimum, Y value in global edge table (GET) index of the first non-empty bucket.
- Set the active edge table (AET) to be empty
- Repeat until AET and GET are empty.
  - Move edges in GET with Y_{min} = Yinto AET for new edges.
  - Sort AET on X

- Use a list of X coordinates to fill spans
- Remove entries in AET with Y_{max} = Y
- Add 1 to y coordinate to go to the nextscan line
- Update X values of remaining entries in AET
- For NULL entry in GET, set X at Y_{min} = X + 1/m, and Y_{min} and 1/m will remain the same.

Example1: Let us consider a polygon ABCD with some specified coordinates A(2, 1), B(2, 4), C(4, 6), and D(4, 3) as shown in figure 3. Compute the coordinates of the intersecting points made by the scan lines with the polygon edges.

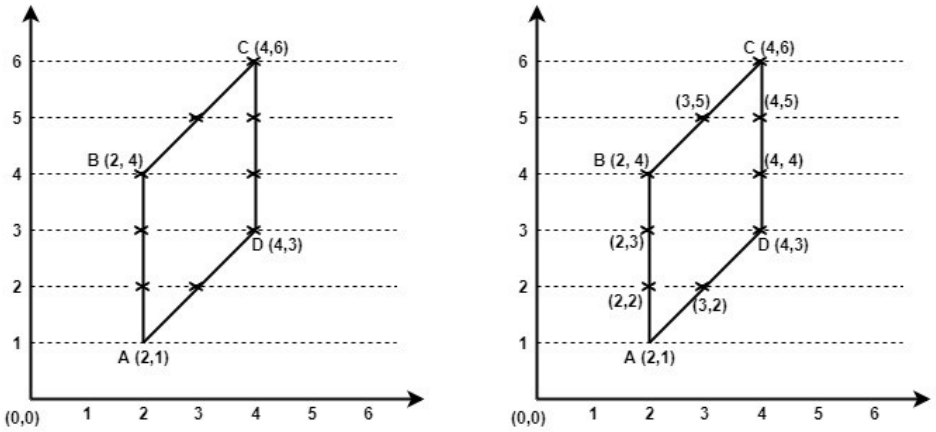


Figure 1: Illustrates the intersecting points made by scan lines with the polygon edges

Solution:The global edge table for the polygon will be-

Sort the vertices in ascending order of Y coordinate i.e. Y_{min} to Y_{max},

I.e. A (2, 1), D(4, 3), B(2, 4), C(4, 6)

Now for the Global edge table entry,

1. The vertex A(2, 1) is connected with D(4, 3) and B(2, 4), So the list entry will for the vertices A(2, 1), D(4, 3)

X1=2 and x2=4

Y1=1 and y2=3

So, Y_{max} = 3 and X_{min} where Y is minimum = 2 and

$$m = \frac{y2-y1}{x2-x1} = \frac{3-1}{4-2} = \frac{2}{2}$$

3	2	1

2. Among the vertex A(2, 1) and B(2, 4)

   Among the vertex A, B

X1=2 and x2=2

Y1=1 andy2=4

So, Y_{max} = 4 and X_{min} where Y is minimum = 2 and

$$m = \frac{y2-y1}{x2-x1} = \frac{4-1}{2-2} = \frac{3}{0}$$

4	2	0

3. Among the vertex D(4, 3), C(4, 6)

X1=4 and x2=4

Y1=3 and y2=6

So, Y_{max} = 6 and X_{min} where Y is minimum = 4 and

$$m = \frac{y2-y1}{x2-x1} = \frac{6-3}{4-4} = \frac{3}{0}$$

6	4	0

4. Among the vertex B(2, 4), C(4, 6)

X1=2 and x2=4

Y1=4 and y2=6

So, Y_{max} = 6 and X_{min} where Y is minimum = 2 and

$$m = \frac{y2-y1}{x2-x1} = \frac{6-4}{4-2} = \frac{2}{2}$$

6	2	1

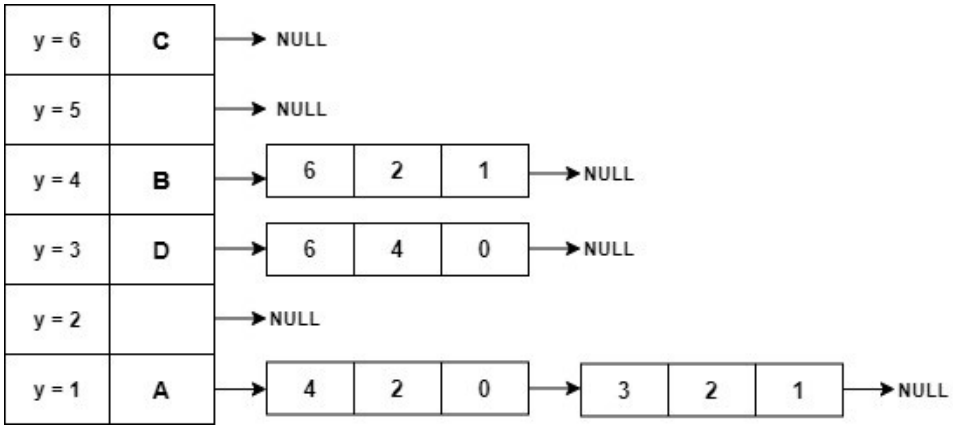
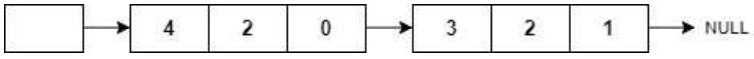


Figure 1: Global Edge Table

Active Edge Table (AET)

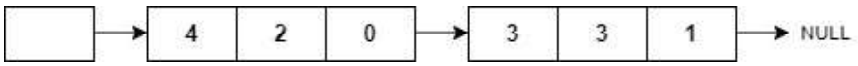
Active Edge Table always starts with a Null entry.

Here the Y_{min} = 1, so the iteration will start from

i = 1 

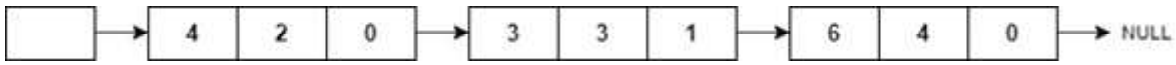
Thus the first intersecting coordinate is (2, 1) and (2, 1)

i = 2, is a NULL entry

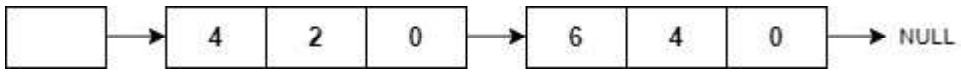


Thus the second intersecting coordinate is (2, 2) and (3, 2)

i = 3, the node connected with D has to be added to the previous node list as shown below.

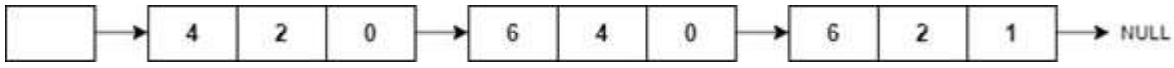


Now the linked list consists of 3 nodes,from which we cannot make pairs of coordinates positions. So the node with Y_{max} equivalent with the y value 3 will be deleted. Finally, the list becomes

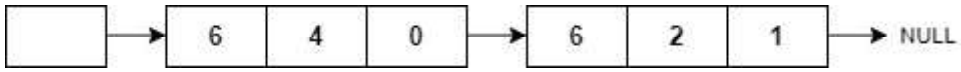


Thus the third intersecting coordinate will be (2, 3) and (4, 3)

i = 4, the node connected with B has to be added to the previous node list as shown below.

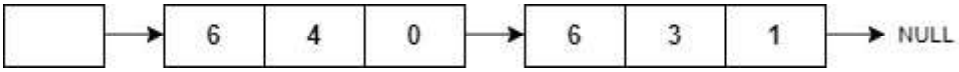


Now the linked list consists of 3 nodes, from which we cannot make pairs of coordinate's positions. So the node with Y_{max} equivalent with the y value 4 will be deleted. Finally, the list becomes



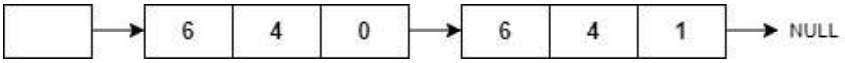
Thus the fourth intersecting coordinate will be (2, 4) and (4, 4)

i = 5, is a NULL entry



Thus the fifth intersecting coordinate is (4, 5) and (3, 5)

i = 6, is a NULL entry



Thus the sixth intersecting coordinate is (4, 6) and (4, 6)

Here the Y_{max} equivalent with the y value 6, so we can delete this two-node, and finally, the list contains the Null node, from where it was started. At the NULL node, stop the further iteration and terminate the processing.

Hence the intersecting coordinates observed during the polygon fill algorithm made by the scan lines with the polygon edges are (2, 1), (2, 1), (2, 2), (3, 2), (2, 3), (4, 3), (2, 4), (4, 4), (4, 5), (3, 5), (4, 6) and (4, 6). The intersecting points are depicted in figure 3.



CHECK YOUR PROGRESS

Q 1: Explain the data structure used by the scan line polygon fill algorithm. Determine the content of the active edge table to fill the polygon with the vertices A(2, 4) B(2, 7) C(4, 9), and D(4, 6).

Q 2: Expect at boundary edges, adjacent pixels are likely to have some characteristics. This property is referred to as ___.

(a) Coherence

(b) Special coherence

(c) Spatial coherence

(d) None of the above

Q 3: The characteristics of pixels on a given scan line change only where a polygon ___ scan line

(a) Intersects

(b) Unions

(c) Differences

(d) All of the above

The coherence property of a scan line can be that the properties of one scene can be linked to the attributes of another section of the scene, allowing the relationship to be exploited to speed up processing.Incremental calculations are frequently used in coherence methods, either along a single scan line or between succeeding scan lines.

5.4 INSIDE-OUTSIDE TEST

In computer graphics, filling the area of an object such as a polygon, circle, ellipse, any bounded region, etc. is a common requirement. This section covers the topic of area filling for standard polygon shapes. Mostly the polygon shapes in geometry are quite simple and easy to identify the interior and exterior points. Common polygon shapes such as triangle, quadrilateral, parallelogram, hexagon, and octagon do not contain any intersecting edges. But in computer graphics, it may have some polygon where edges may intersect and form some irregular patterns as shown in figures 5 and 6. In such cases, it is very difficult to identify the interior or exterior portion of the polygon. Computer graphics packages are used two methods namely the odd-even rule and non-zero winding number rule to identify the interior and exterior regions of a polygon.

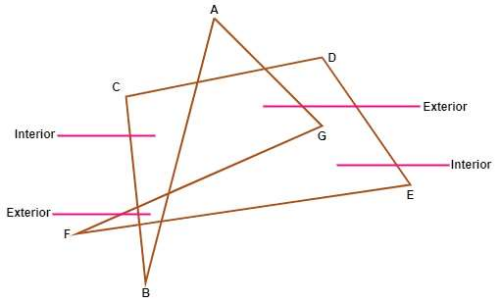


Figure 5 Odd Even rule

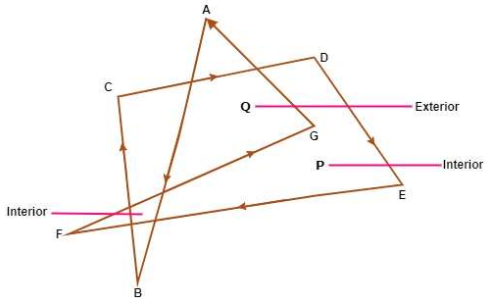


Figure 1 Nonzero Winding number rule

5.4.1 Odd-Even Rule

Odd-even rule or odd parity rule is used to check a region of a polygon, whether it is interior or exterior to the polygon boundary. The Odd-even rule is used to draw a line from a point P from the

polygon towards the outside. The number of polygon edges crossed by the line along its path determines the region status. If the polygon edges crossed by the line are odd, then the point P is considered an interior point, otherwise, P is an exterior point. To obtain an accurate edge count, it must be ensured that the line should not pass through any connecting vertices of polygon edges. Figure 5 depicts the odd-even rule used to identify the interior and exterior points for the given polygon.

5.4.2 Non-Zero Winding Number Rule

The non-zero winding number rule is another approach for defining interior sections of a polygon. This method counts the number of edges winding around a given point, in the scene, in an anticlockwise direction. That counting of the polygon edges is known as the winding number. The points with non-zero winding numbers will be defined as the interior points. For the application of the non-zero winding number rule to an object, first set the winding number count to zero (0), and draw a line from any point on the object extent to the outside coordinates. To get a better result, the line should not pass through a connecting vertex of polygon edges. It has to count the number of polygon edges cross by the line along its line path from a point P to a distant point beyond the polygon boundary in any direction. For every edge with anticlockwise direction or traverse right to left are intersected by the line, add 1 to the counter, and for every edge with clockwise direction intersected by the line add - 1 to the counter. After calculation of the number of edges intersected by the line in the clockwise or anticlockwise direction, the final counter value, i.e. the winding number gives us the relative position of the point P. P is defined as an interior point for a non-zero winding number. Otherwise, P will be considered as an exterior point. Figure 6, depicts interior and exterior points on a polygon defined using the winding number rule.

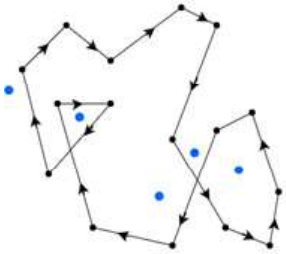
For standard polygon shape, both the odd-even rule and winding number rule results in the same, but for some complex or irregular polygon containing a self-intersecting set of edges, may show the different result as shown in figure 5 and 6.

The non-zero winding number rule is more versatile than the odd-even rule, so some graphics system uses the non-zero winding number rule to implement area filling.

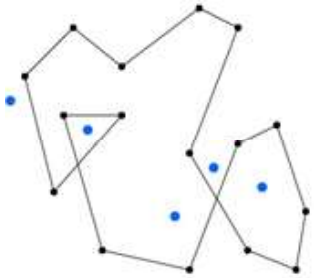


EXERCISE

Question1: Test the dot (highlighted positions) positions, whether they are interior or exterior to the given polygon using non zero winding number rule.



Question2: Test the dot (highlighted positions) positions are interior or exterior to the polygon using odd even rule.



5.5 BOUNDARY FILL ALGORITHM

Another algorithm used to fill the area of an object is the boundary fill algorithm. In this approach, area filling starts with a point inside a region and paints from the interior towards the outside boundary. If the object border is defined by a single colour, the fill algorithm will proceed pixel by pixel until it reaches the boundary colour. In computer graphics painting

packages where the interior points may be detected easily, the boundary fill approach is commonly utilized. An artist can draw a figure's outline and choose fill colour or pattern as well as an interior point. The system may then paint the figures inside.

Input parameters required for the boundary fill algorithm are the coordinate of an interior point (x, y), a fill colour, and a boundary colour. Starting with the inside point coordinate, the system will examine its surrounding pixels and finish filling the area if they match the boundary colour. If not, the pixels will be painted with specified colour and recursively the neighbour will be tested. This process will continue until all pixels up to the boundary are not coloured.

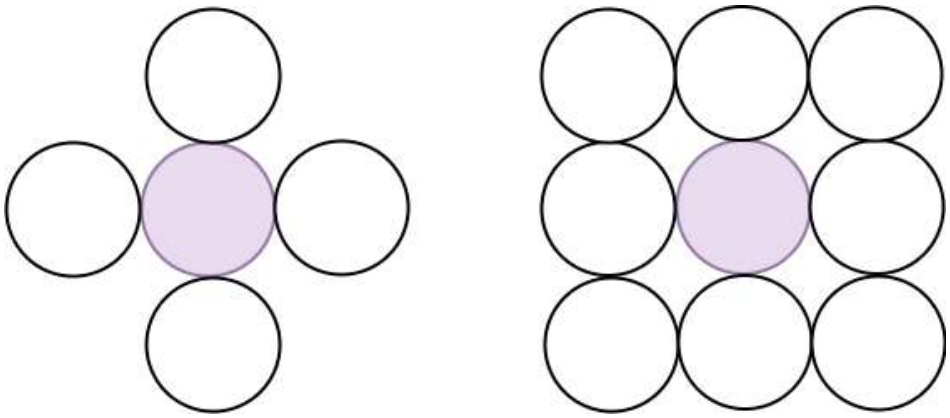


Figure 1: 4 - Connected and 8 - connected Boundary fill algorithm

Figure 7 depicts the two most popular and efficient boundary fill algorithms. In the first method, the system tests the four neighbours, above, below, left, and right of the specified interior pixel point. This process is known as the 4-connected area filling approach. The other method checks the four diagonal pixels including the left, right, above, and below of the current pixel as shown in figure 7, i.e. in this method total of 8 pixels has to be tested by the system to proceed outwards towards the object boundary. If the colours of these neighbour pixels are not matched with the specified boundary colour, the process will continue until it encounters the boundary colour. The fill method using this approach is known as the 8-connected. The 8-connected approach is used in comparatively complex figures and performs better results in comparison to the 4-connected approach. The

recursive boundary fill algorithm has some drawbacks such as if one or more of the interior pixels are coloured with the boundary specified colour, then the boundary fill algorithm will fail to paint the interior pixels. In this approach colour of the neighbour pixels of the current pixel is tested recursively whether they matched the boundary colour or not. So if it found any pixel with the specified boundary colour, the remaining pixels will be unfilled. To avoid this situation, the colour of the interior pixel may be reset before applying the boundary fill algorithm.

5.5.1 4- Connected Area Fill Algorithm

Step 1: set the interior pixel (seed point)coordinate (x, y), its default colour (DC), fill colour (FC), Boundary colour (BC).

Step 2: Specify the polygon boundary value.

Step 3: Test the seed colour, if it is in default colour, then repeat the following steps and till the system encountered boundary specified colour.

If getpixel(x, y) = DC, then repeat step 4 and 5

Step 4: Change the DC value of the seed point with the FC.

SetPixel(x, y, FC)

Step 5: For 4 neighbour pixel (above, below, lef, and right) of the seed point, recursively call the function -

```
FloodFill(x, y+1, FC, DC)     //above the seed point
FloodFill(x, y-1, FC, DC)     //below the seed point
FloodFill(x-1, y, FC, DC)     //left of the seed point
FloodFill(x+1, y, FC, DC)     //right of the seed point
```

Step 6: Stop

5.5.2 8-Connected Area Fill Algorithm

Step 1: set the interior pixel (seed point)coordinate (x, y), its default colour (DC), fill colour (FC), Boundary colour (BC).

Step 2: Specify the polygon boundary value.

Step 3: Test the seed colour, if it is in default colour, then repeat the

following steps and till the system encountered boundary specified colour.

If getpixel(x, y) = DC, then repeat step 4 and 5

Step 4: Change the DC value of the seed point with the FC.

SetPixel(x, y, FC)

Step 5: For 4 neighbour pixel (above, below, left, and right) of the seed point, recursively call the function -

```
FloodFill(x, y+1, FC, DC)    //above the seed point
FloodFill(x, y-1, FC, DC)    //below the seed point
FloodFill(x-1, y, FC, DC)    //left of the seed point
FloodFill(x+1, y, FC, DC)    //right of the seed point
FloodFill(x+1, y+1, FC, DC)  //right top of the seed point
FloodFill(x-1, y+1, FC, DC)  //right left of the seed point
FloodFill(x-1, y-1, FC, DC)  //bottomleft of the seed point
FloodFill(x+1, y-1, FC, DC)  //bottomright of the seed point
```

Step 6: Stop

The 4-connected and 8-connected area fill approaches needs to maintain a stack so as to store the neighbour pixels' information. To overcome this situation another approach has come into the picture. The horizontal pixels across a scan line are filled using this method. The stack initial for each scan line has to maintain in the scan line approach, instead of storing the information regarding 4 or 8 neighbour pixels of the current position. Starting at the initial interior point, fill the continuous span of pixels on the scan line with this approach. Thus the stack starting positions for the next scan line will be located next and the process will continue up to the completion of the area fill for the specified object.



CHECK YOUR PROGRESS

Q 4: In.....algorithm edges can be specified in different colour.

A. flood fill

B. boundary fill

C. DDA fill

D. BSA fill

Q 5: Boundary fill algorithm uses .....call to the fill function.

A. recursive

B. static

C. one

D. none of these

Q 6: Which statement is true about the 4 connected regions?

A. it is faster than 8 connected region

B. sometimes it leaves region unfilled.

C. both a and b

D. neither a or b

Q 7: Active edge table is used in following algorithm

A. boundary fill

B. region fill

C. scan line

D. edge fill

5.6. FLOOD FILL ALGORITHM

The flood fill algorithm is used for area fill when the object boundary contains more than one colour. In this approach, the interior points can paint by replacing a specified colour with the required fill colour instead of searching for the boundary colour. The flood fill algorithm is not concerned about the object boundary colour. Here the process starts from an interior point say (x, y), and reassigns all pixel values that are currently set to a given interior colour with the desired fill colour. In figure 8, the object boundary has four different colours in its boundary such as blue, yellow, orange, and purple. The interior points have a default colour of white. Now, assume that the interior points have to fill with the target colour red. Without testing the boundary colour, the method will start its area filled with the target colour from any interior pixel (x, y). If the colour of the specified

coordinate is white, i.e. the default colour, then it will be replaced by the target colour. To fill the other remaining interior points, the 4 - connected or 8 - connected approach can be utilized. If the interior pixels have more than one colour, then all pixel colours have to reassign to a uniform or solid colour before applying the flood fill algorithm. The interior pixels are coloured using the 4 - connected or 8 - connected area fill approach until all points have been repainted. The 8-connected approach helps to paint the diagonal points (x-1, y-1), (x-1, y+1), (x+1, y+1), and (x+1, y-1) of the current pixel (x,y).

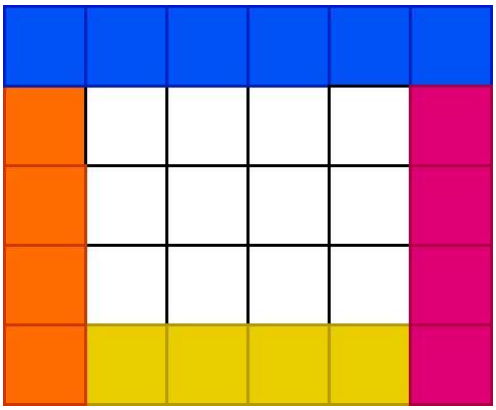


Figure 2: Area covered with a multi-coloured boundary

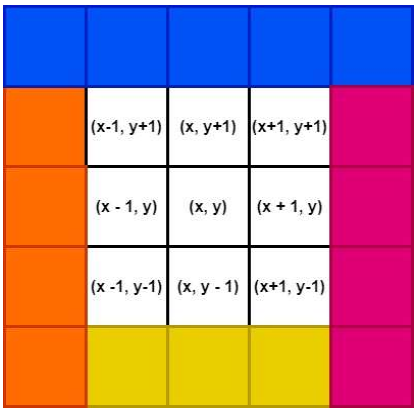


Figure 2Neighbourcoordinates positions for the current pixel (x, y)



5.7 LET US SUM UP

- The polygon fill algorithm locates the intersecting points of the scan lines that make with the polygon edges.
- If the y coordinate of two consecutive endpoints is monotonically decreased or increased, then the middle vertex needs to be counted as a single intersection point for any scan line passing through it.
- In the connecting point where both the polygon edges are lies in the same direction, the count will be considered two times consecutively.
- Using the concept of a sorted edge table, the performance of the Polygon fill algorithm can increase.
- Active Edge Table (AET) gives the final intersecting points made by the scan line with the polygon edges.

- The coherence property of a scan line can be that the properties of one scene can be linked to the attributes of another section of the scene, allowing the relationship to be exploited to speed up processing.
- In the odd-even rule, if the polygon edges cross by the line are odd, then the point P is considered an interior point, otherwise, the P is an exterior point.
- The points with non-zero winding numbers will be defined as the interior points.
- The non-zero winding number rule is more versatile than the odd-even rule.
- In the 4 - connected approach, the system tests the four neighbours, above, below, left, and right of the specified interior pixel point.
- In the 4 - connected approach and the 8 - connected approach colour of the neighbour pixels of the current pixel is tested recursively whether they matched the boundary colour or not.
- In the 4-connected and 8-connected area fill approach, it needs to maintain a stack, storing neighbour pixels' information.
- The flood fill algorithm is used for area fill when the object boundary contains more than one colour.
- The flood fill algorithm is not concerned about the object boundary colour.
- If the interior pixels have more than one colour, then all pixel colours have to reassign to a uniform or solid colour before applying the flood fill algorithm.

5.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: 2. A 3. A 4. A 5. A 6. C 7. C

5.9 MODEL QUESTIONS

Short Questions (Answer each question in around 150 words).

Q 1: Choose the correct options from the followings:

i. Scan-line coherence property is used in

A. line drawing algorithm
B. scan line polygon filling algorithm
C. flood fill polygon filling algorithm
D. none of above

ii. In the scan line algorithm the intersection points are

A. sorted in ascending order
B. sorted in descending order
C. not sorted
D. deleted

iii. _____ is a method for testing a pixel inside of a polygon.

A. even-odd method
B. winding number method
C. A and B
D. None of these

iv. _____ is a basic approach used to fill the polygon.

A. seed fill
B. scan fill
C. A and B
D. None of these

v. The seed fill algorithm for filling polygon is classified as _____ fill algorithm and _____ fill algorithm.

A. flood, boundary
B. even, odd
C. edge, flood
D. boundary, scan

vi. Polygon filling algorithms those fill interior-defined regions are called _____ algorithms.

A. flood fill
B. boundary fill
C. scan line
D. edge fill

vii. Polygon filling algorithms those fill boundary-defined regions are called _____ algorithms.

A. flood fill
B. boundary fill
C. edge line
D. A and B

viii. In a boundary fill algorithm for filling polygon, boundary defined regions may be either _____ connected or_____ connected.

A. 2,4
B. 4,8
C. 8,16
D. 8,6

ix. The getpixel function gives the _____ of the specified pixel.

A. intensity
B. colour
C. Size
D. Shape

x. The putpixel function draws the pixel specified_____.

A. intensity
B. colour
C. Size
D. Shape

xi. Seed fill algorithm for filling polygon is _____ algorithm.

A. recursive
B. non-recursive
C. A and B
D. None of these

xii. Scan line algorithm for filling polygon is the _____ algorithm.

A. recursive
B. non-recursive
C. A and B
D. None of these

Q 2: Answer the following questions:

i. What is a polygon filling algorithm?

ii. Explain the working principle of the scan line polygon fill algorithm.
iii. Explain the concept of the boundary fill algorithm.
iv. Write the differences between the 4-connected and 8-connected approaches used in the boundary algorithm.
v. Discuss the advantages and disadvantages of the boundary fill algorithm.
vi. Discuss the advantages of the flood fill algorithm.
vii. Explain the advantages of the scan line polygon fill algorithm.
viii. Write the steps involved in the 4 - Connected and 8 - Connected approaches used in the area fill algorithm.

*** ***** ***