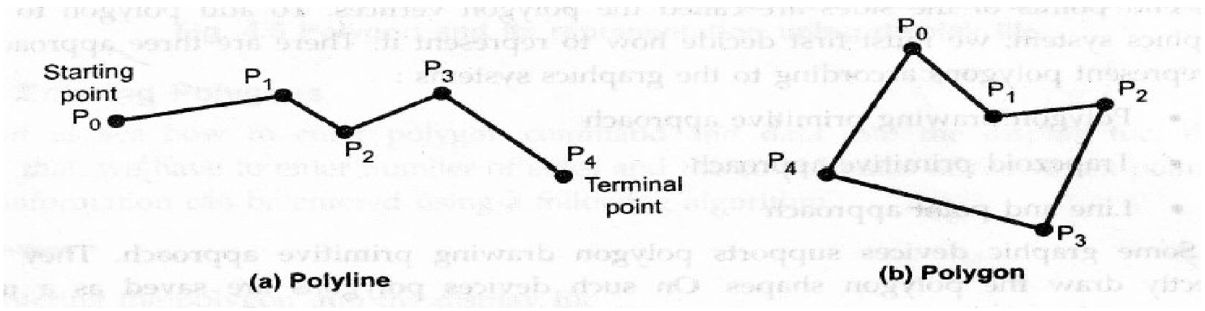
Fill Area Primitives

Introduction

Polyline

- A chain of connected line segments
- Specified by giving the vertices (nodes) P0, P1, P2…. and so on.
- First vertex → initial or starting point
- Last vertex →final or terminal point
- When starting and terminal point of any polyline is same i.e. when polyline is closed then it is called polygon.



Types of Polygon

Convex Polygon

- If the line segment joining any two interior points of the polygon lies completely inside the polygon then it is called convex polygon.

Concave Polygon

- If the line segment joining any two interior points of the polygon lies not completely inside the polygon then it is called concave polygon.

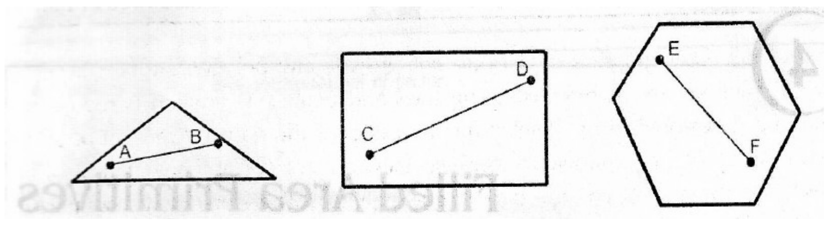


Figure : Convex Polygon

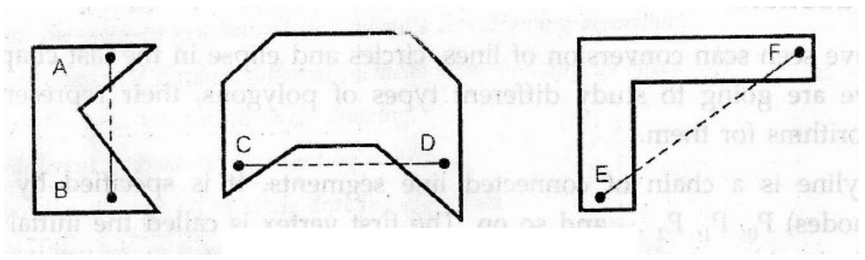


Figure : Concave Polygon

Polygon Filling

- It is process of highlighting all the pixels which lie inside the polygon with any color other than background color.
- It is the process of coloring the area of a polygon. An area or region is defined as a collection of pixels.

Two Basic approaches to fill the polygon

- One way to fill a polygon is to start from a given "seed", point known to be inside the polygon and highlight outward from this point i.e. neighboring pixels until we encounter the boundary pixels. This approach is called **seed fill** because color flows from the seed pixel until reaching the polygon boundary. Suitable for complex objects.
- Another approach to fill the polygon is to apply the inside test i.e. to check whether the pixel is inside the polygon or outside the polygon and then highlight pixels which lie inside the polygon. This approach is known as **scan-line algorithm**. It avoids the need for a seed pixel but it requires some additional computation. Suitable for simple objects, polygons, circles.

Seed Fill Algorithm

- **Principle**: "Start at a sample pixel called as a seed pixel from the area, fill the specified color value until the polygon boundary".
- Classified as:
  1. Flood fill algorithm
  2. Boundary fill algorithm
- Two types of areas or regions
  1. Boundary-defined region
  2. Interior-defined region

Boundary-defined region

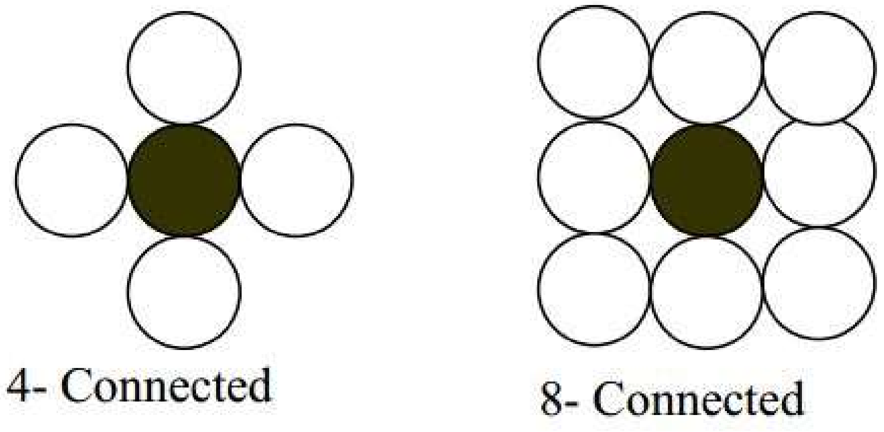
- Pixels that mark the boundary of the region are called as boundary defined region pixels.
- Have unique color that is not same as interior pixels.
- Algorithm that are used to fill boundary-defined regions are called as *boundary fill algorithms or edge fill algorithm.*

Interior-defined region

- A collection of same color continuous pixels.
- Algorithms used to fill interior-defined region are called as *flood-fill algorithms.*

Boundary-fill Algorithm

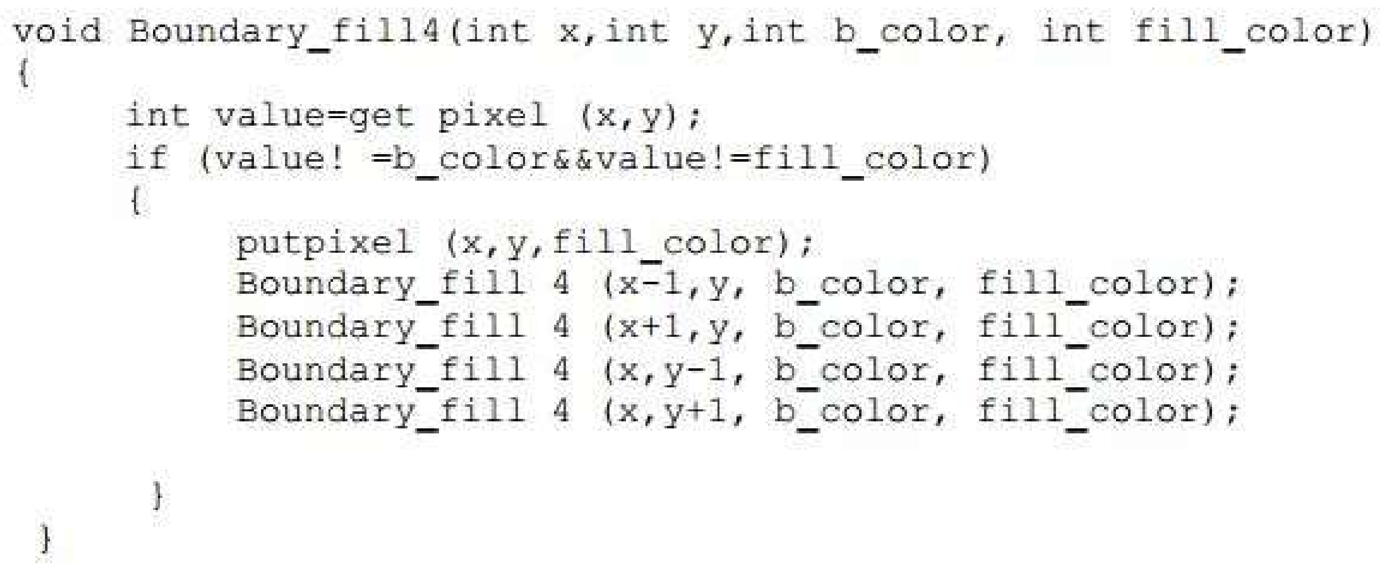
- In this method, edges of the polygons are drawn first **in a single color**.
- This algorithm picks a point inside an object and starts to fill until it hits the boundary of the object. The color of the boundary and the fill color that should be different for this algorithm to work.
- An area filling starts from a random pixel called as seed, inside a region and continues painting towards the boundary. This process continues outwards pixel by pixel until the boundary color is reached.
- A boundary-fill procedure accepts as input the co-ordinates of an interior point (x, y), a fill color, and a boundary color. Starting from (x, y), the procedure tests neighboring positions to determine whether they are of boundary color. If not, they are painted with the fill color, and their neighbors are tested.
- There are two methods for proceeding to the neighboring pixels from the seed pixel:
    - 4 adjacent pixel may be tested (4-connected)
        - 4 neighboring pixels i.e., left, right, up and down are tested.
    - 8 adjacent pixel may be tested (8-connected)
        - 8 neighboring pixels are tested i.e., left, right, up, down, and diagonal pixels are also tested.



- Fill method that applied and tests its 4 neighboring pixel is called 4-connected.
- Fill method that applied and tests its 8 neighboring pixel is called 8-connected.

The following procedures illustrate the recursive method for filling 4-connected region and 8-connected with color specified in parameter fill color (fill_color) up to a boundary color specified with parameter boundary color (b_color):

Boundary fill 4-Connected

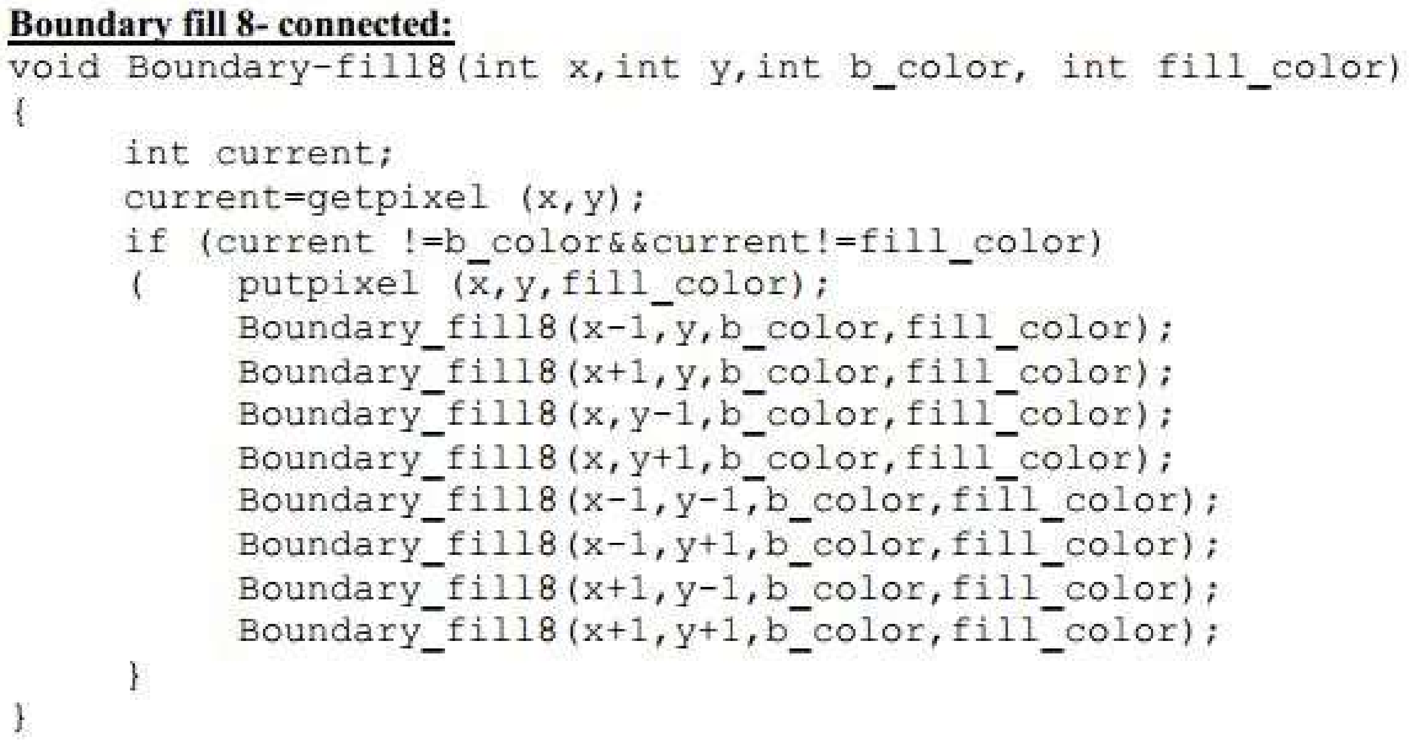


```
void Boundary_fill4(int x,int y,int b_color, int fill_color)
{
     int value=get pixel (x,y);
     if (value! =b_color&&value!=fill_color)
     {
          putpixel (x,y,fill_color);
          Boundary_fill 4 (x-1,y, b_color, fill_color);
          Boundary_fill 4 (x+1,y, b_color, fill_color);
          Boundary_fill 4 (x,y-1, b_color, fill_color);
          Boundary_fill 4 (x,y+1, b_color, fill_color);

      }
 }
```

Note: 'getpixel' function gives the color of specified pixel and 'putpixel' function draws the pixel with specified color.

Boundary fill 8-Connected



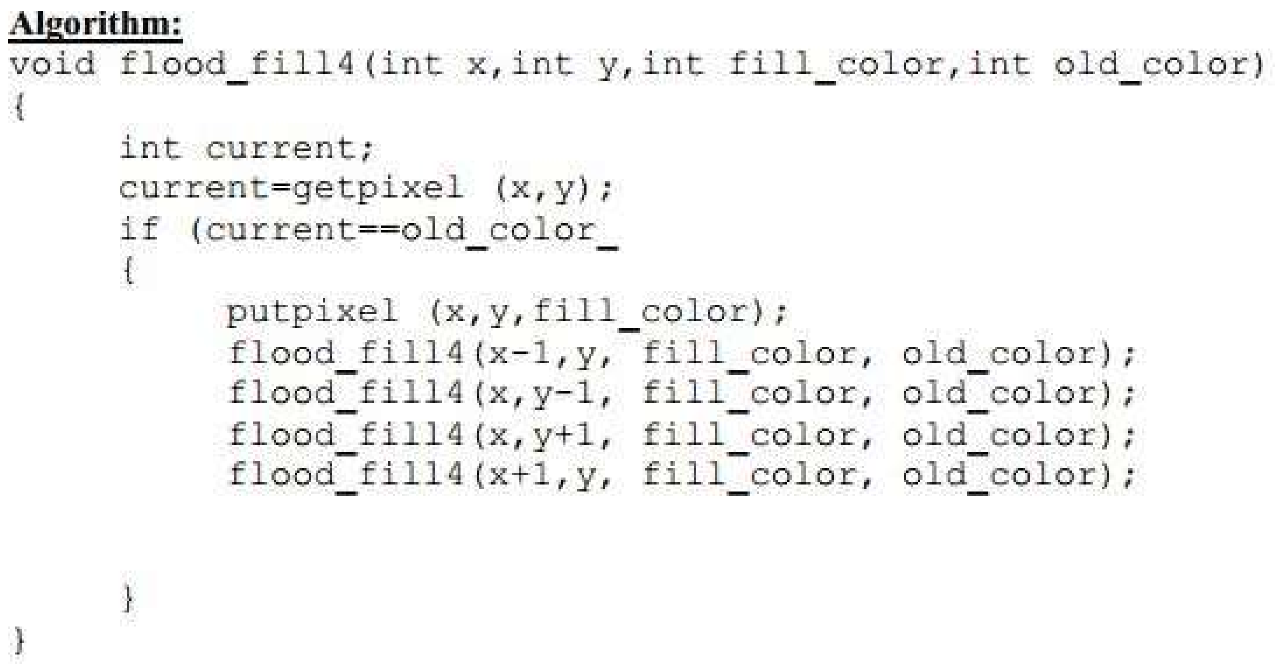
Boundary fill 8- connected:

```
void Boundary-fill8(int x,int y,int b_color, int fill_color)
{
     int current;
     current=getpixel (x,y);
     if (current !=b_color&&current!=fill_color)
     {    putpixel (x,y,fill_color);
          Boundary_fill8(x-1,y,b_color,fill_color);
          Boundary_fill8(x+1,y,b_color,fill_color);
          Boundary_fill8(x,y-1,b_color,fill_color);
          Boundary_fill8(x,y+1,b_color,fill_color);
          Boundary_fill8(x-1,y-1,b_color,fill_color);
          Boundary_fill8(x-1,y+1,b_color,fill_color);
          Boundary_fill8(x+1,y-1,b_color,fill_color);
          Boundary_fill8(x+1,y+1,b_color,fill_color);
     }
}
```

Flood Fill Algorithm

- Flood fill algorithm is applicable when we want to fill an area that is not defined within a single color boundary.
- If fill area is bounded with different color, we can paint that area by replacing a specified interior color instead of searching of boundary color value.
- In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original color exists, the algorithm is completed.
- We start from a specified interior pixel (x, y) and reassign all pixel values that are currently set to a given interior color with desired fill-color.
- Using either a 4-connected or 8-connected approach, we can step through pixel positions until all interior point have been filled.
- The following procedure illustrates the recursive method for filling 4 -connected region using flood-fill algorithm.

4 connected flood fill algorithm



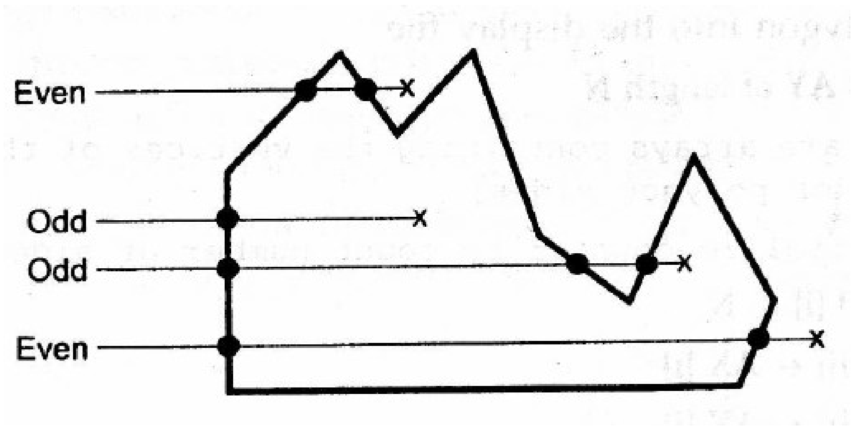
Algorithm:

```
void flood_fill4(int x,int y,int fill_color,int old_color)
{
      int current;
      current=getpixel (x,y);
      if (current==old_color_
      {
           putpixel (x,y,fill_color);
           flood_fill4(x-1,y, fill_color, old_color);
           flood_fill4(x,y-1, fill_color, old_color);
           flood_fill4(x,y+1, fill_color, old_color);
           flood_fill4(x+1,y, fill_color, old_color);


      }
}
```

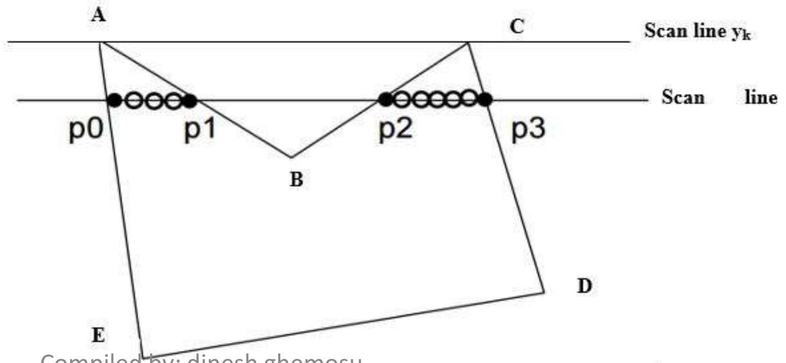
Inside and Outside Test of Polygon

- Area filling algorithms and other graphics processes often need to identify interior and exterior region of objects.
- To determine whether or not a point is inside of a polygon, even-odd method.
  - **Even-odd method(Odd-parity rule)**
    - In this method, we draw a line segment from a point X(point in question) and to a point known to be outside the polygon.
    - Now count the intersections of the line segment with the polygon boundary.
    - If there are an odd number of intersections then the point X is inside; otherwise it is outside.



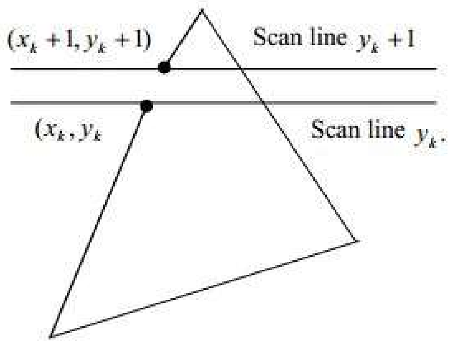
Scan Line Algorithm

- A scan-line algorithm fills horizontal pixels across scan lines, instead of proceeding to 4-connected or 8-connected neighboring points.
- This algorithm works by intersecting scanline with polygon edges and fills the polygon between pairs of intersections.
- **Approach:**
  - ➢ For each scan-line crossing a polygon, this algorithm computes the intersection points of the scan line with the polygon edges.
  - ➢ These intersection points are then sorted from left to right (i.e., increasing value of x coordinate), and the corresponding positions between each intersection pair are filled by the specified color (shown in figure below).
  - ➢ This procedure is repeated until complete polygon is filled.



To find intersection points

- Find the slope m for each polygon boundary (i.e., each edge) using vertex coordinates. If A(x_1, y_1) and B(x_2, y_2) then slope for polygon boundary AB can be calculated as follows;
  - $m = (y_2-y_1)/(x_2-x_1)$
- We can find the intersection point on the lower scan line if the intersection point for current scan line is known. Let (x_k, y_k) be the intersection point for the current scan line y_k.
  - Scanning starts from top to bottom, y-coordinates between the two scan line changes by 1
    - i.e., $y_{k+1} = y_k - 1$.
  - And x coordinate can be calculated as,
    - $x_{k+1} = x_k - 1/m$



Filling Rectangles

- Scan line method can be used to fill a rectangle.

```
for (y=$y_{min}$ to y= $y_{max}$ of the rectangle)          //by scan line
    for(x = $x_{min}$ to x= $x_{max}$)                     //by pixels
        putpixel(x, y, fill_color)
```

Filling Curved Boundary Areas

- Scan line fill of regions with curved boundaries requires more work than polygon filling, since intersection calculations now involve non-linear boundaries.
- For simple curves such as circles or ellipses, performing a scan line fill is straightforward process.
- We only need to calculate the two scan-line intersections on opposite sides of the curve.
- Then simply fill the horizontal spans of pixel between the boundary points on opposite side of curve.
- Symmetries between quadrants are used to reduce the boundary calculation.