<u>Assignment No:</u>

Title/ Problem Statement:

Write C++/Java program to draw a convex polygon and fill it with desired color using Seed fill algorithm.

Prerequisites:

Objectives:

1. To understand Seed Fill algorithms used for computer graphics.

Theory :

Fill Algorithms •
Given the edges defining a polygon, and a color for the polygon, we need to fill all the pixels inside the polygon.

- Three different algorithms: –

1. Scan-line fill
2. Boundary fill
3. Flood fill

Filled Area Primitives

Two basic approaches to area filling on raster systems:

– Determine the overlap intervals for scan lines that cross the area (scan-line)
– Start from an interior position and point outward from this point until the boundary condition reached (fill method)

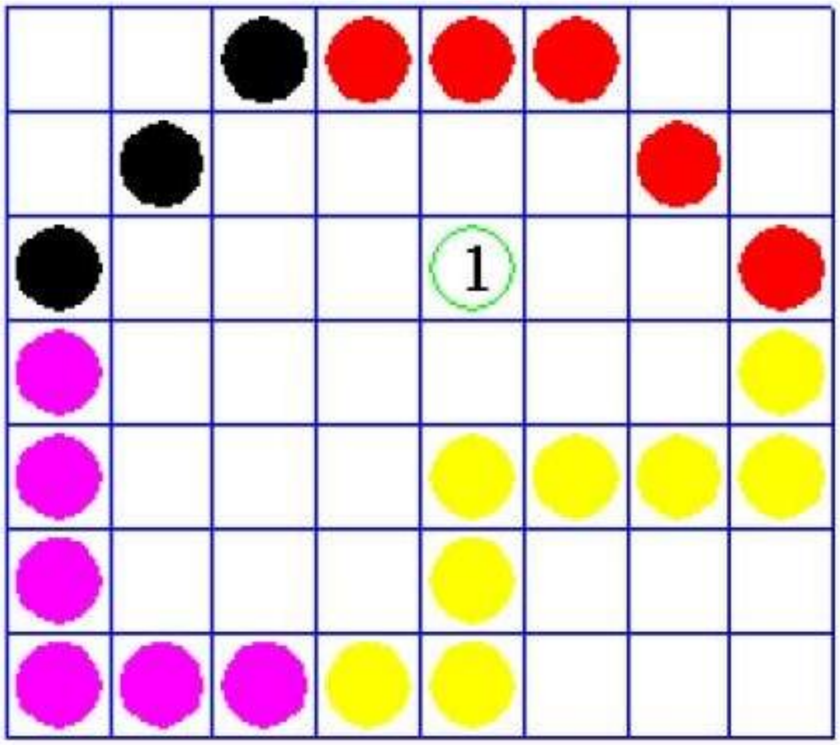
- Scan-line: simple objects, polygons, circles,
- Fill-method: complex objects, interactive fill.

Flood Fill Algorithm

Sometimes we come across an object where we want to fill the area and its boundary with different colors. We can paint such objects with a specified interior color instead of searching for particular boundary color as in boundary filling algorithm.

Instead of relying on the boundary of the object, it relies on the fill color. 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.

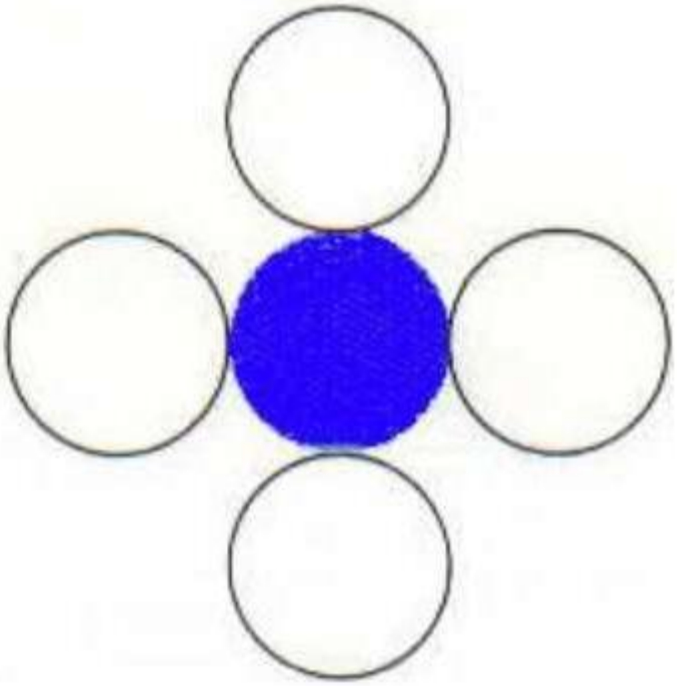
Once again, this algorithm relies on the Four-connect or Eight-connect method of filling in the pixels. But instead of looking for the boundary color, it is looking for all adjacent pixels that are a part of the interior.



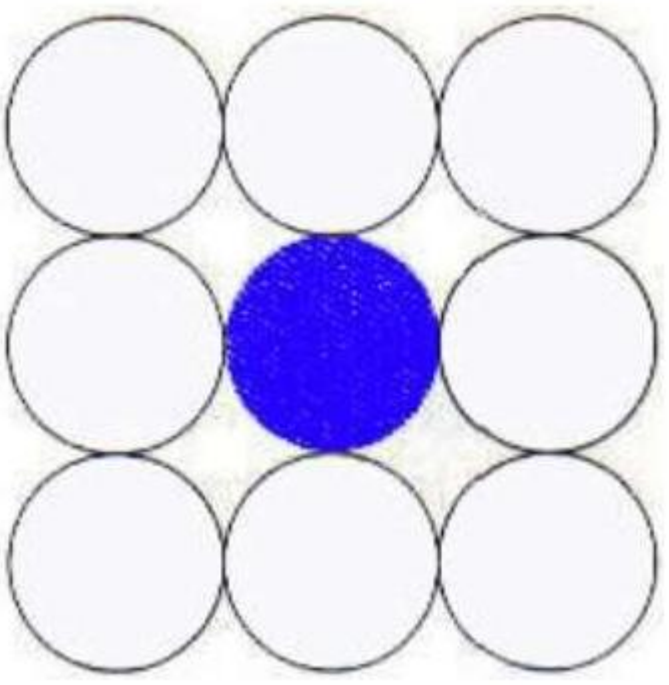
Pixel-Defined Regions

Definition: Region R is the set of all pixels having color C that are connected to a given pixel S.

4-adjacent: pixels that lie next to each other horizontally or vertically, NOT diagonally.
8-adjacent: pixels that lie next to each other horizontally, vertically OR diagonally.
4-connected: if there is unbroken path of 4-adjacent pixels connecting them.



8-connected: unbroken path of 8-adjacent pixels connecting them.



Recursive Flood-Fill Algorithm

- Recursive algorithm
- Starts from initial pixel of color, intColor
- Recursively set 4-connected neighbors to newColor
- **Flood-Fill:** floods region with newColor

- **Basic idea:**
- start at "seed" pixel (x, y)
- If (x, y) has color intColor, change it to newColor
- Do same recursively for all 4 neighbors

Recursive Flood-Fill Algorithm

Note: getPixel(x,y) used to interrogate pixel color at (x, y)

```
void floodFill(short x, short y, short intColor)
  {
        if(getPixel(x, y) == intColor)
                {
                 setPixel(x, y);
                floodFill(x – 1, y, intColor); // left pixel
                floodFill(x + 1, y, intColor); // right pixel
                floodFill(x, y + 1, intColor); // down pixel
                floodFill(x, y – 1, intColor); // fill up
                }
}
```

Recursive Flood-Fill Algorithm

- This version defines region using intColor
- Can also have version defining region by boundary
- Recursive flood-fill is somewhat blind and some pixels may be retested several times before algorithm terminates
- Region coherence is likelihood that an interior pixel mostly likely adjacent to another interior pixel Coherence can be used to improve algorithm performance
- A run is a group of adjacent pixels lying on same
- Exploit runs(adjacent, on same scan line) of pixels

Conclusion: Thus we have studied inscribed and Circumscribed circles in the triangle.