**Ex. No. :2 Date: 9/9/2025**

.



**Register No.: 231701039 Name: Prasanna R**

Scan Conversion of Polygon and Area Filling using Flood-Fill Algorithm

AIM:

To implement:

1. Scan conversion (drawing) of a polygon.
2. Flood-fill algorithm to fill the polygonal area in a graphics window.

# Procedure:

**Polygon Scan Conversion:**

1. Input the number of vertices and their coordinates.
2. Connect the vertices in order to draw polygon edges using line drawing algorithms.

# Flood-Fill Algorithm:

1. Select a point inside the polygon as the seed point.
2. Recursively check neighboring pixels.
3. Replace target color with fill color if conditions match.

***Program:***

import matplotlib.pyplot as plt import numpy as np

from collections import deque

width, height = 300, 300

canvas = np.ones((height, width, 3), dtype=np.uint8) \* 255 # White canvas

def draw\_polygon(vertices): for i in range(len(vertices)):

x1, y1 = vertices[i]

x2, y2 = vertices[(i+1)%len(vertices)] bresenham\_line(x1, y1, x2, y2)

.



def bresenham\_line(x1, y1, x2, y2): dx = abs(x2 - x1)

dy = abs(y2 - y1) x, y = x1, y1

sx = 1 if x2 > x1 else -1 sy = 1 if y2 > y1 else -1 if dx > dy:

err = dx / 2 while x != x2:

canvas[y, x] = [0, 0, 0] err -= dy

if err < 0: y += sy

err += dx x += sx

canvas[y, x] = [0, 0, 0] else:

err = dy / 2 while y != y2:

canvas[y, x] = [0, 0, 0] err -= dx

if err < 0: x += sx

err += dy y += sy

canvas[y, x] = [0, 0, 0]

'''def flood\_fill(x, y, target\_color, fill\_color):

if x < 0 or x >= width or y < 0 or y >= height: return

if not np.array\_equal(canvas[y, x], target\_color): return

canvas[y, x] = fill\_color

flood\_fill(x+1, y, target\_color, fill\_color) flood\_fill(x-1, y, target\_color, fill\_color) flood\_fill(x, y+1, target\_color, fill\_color) flood\_fill(x, y-1, target\_color, fill\_color)

'''

def flood\_fill\_iter(x, y, target\_color, fill\_color): target = np.array(target\_color, dtype=np.uint8) fill = np.array(fill\_color, dtype=np.uint8)

if x < 0 or x >= width or y < 0 or y >= height: return

.



if not np.array\_equal(canvas[y, x], target): return

q = deque([(x, y)]) while q:

cx, cy = q.popleft()

if cx < 0 or cx >= width or cy < 0 or cy >= height: continue

if np.array\_equal(canvas[cy, cx], target): canvas[cy, cx] = fill\_color q.append((cx+1, cy))

q.append((cx-1, cy))

q.append((cx, cy+1))

q.append((cx, cy-1))

# Main Execution

vertices = [(50, 50), (250, 50), (200, 200), (100, 250), (50, 150)]

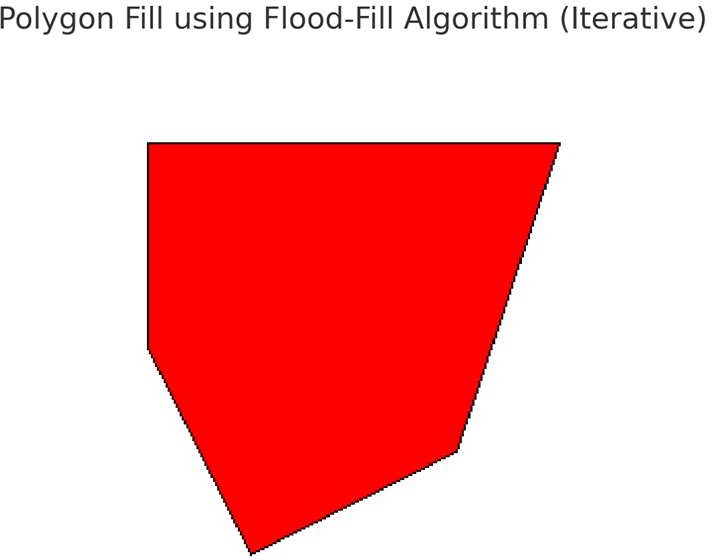
draw\_polygon(vertices)

flood\_fill\_iter(150, 100, [255, 255, 255], [255, 0, 0])

plt.imshow(canvas)

plt.title("Polygon Fill using Flood-Fill Algorithm") plt.axis('off')

plt.show()



# Result:

A polygon will be drawn and filled with red color using the flood-fill algorithm.