# Aim

To implement Polygon Drawing and Flood Fill Algorithm using Python with NumPy as the pixel buffer.

# Code:

# import matplotlib.pyplot as plt

# import numpy as np

# from collections import deque

# W, H = 300, 300

# board = np.ones((H, W, 3), dtype=np.uint8) \* 255

# def connect\_points(points):

# for i in range(len(points)):

# x1, y1 = points[i]

# x2, y2 = points[(i + 1) % len(points)]

# line\_plot(x1, y1, x2, y2)

# def line\_plot(x1, y1, x2, y2):

# dx, dy = abs(x2 - x1), abs(y2 - y1)

# x, y = x1, y1

# stepx = 1 if x2 > x1 else -1

# stepy = 1 if y2 > y1 else -1

# if dx > dy:

# err = dx // 2

# while x != x2:

# board[y, x] = [0, 0, 0]

# err -= dy

# if err < 0:

# y += stepy

# err += dx

# x += stepx

# board[y, x] = [0, 0, 0]

# else:

# err = dy // 2

# while y != y2:

# board[y, x] = [0, 0, 0]

# err -= dx

# if err < 0:

# x += stepx

# err += dy

# y += stepy

# board[y, x] = [0, 0, 0]

# def fill\_area(sx, sy, target, new):

# tgt = np.array(target, dtype=np.uint8)

# if sx < 0 or sx >= W or sy < 0 or sy >= H:

# return

# if not np.array\_equal(board[sy, sx], tgt):

# return

# q = deque([(sx, sy)])

# while q:

# cx, cy = q.popleft()

# if cx < 0 or cx >= W or cy < 0 or cy >= H:

# continue

# if np.array\_equal(board[cy, cx], tgt):

# board[cy, cx] = new

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

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

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

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

# shape = [(60, 60), (240, 60), (190, 210), (110, 240), (60, 160)]

# connect\_points(shape)

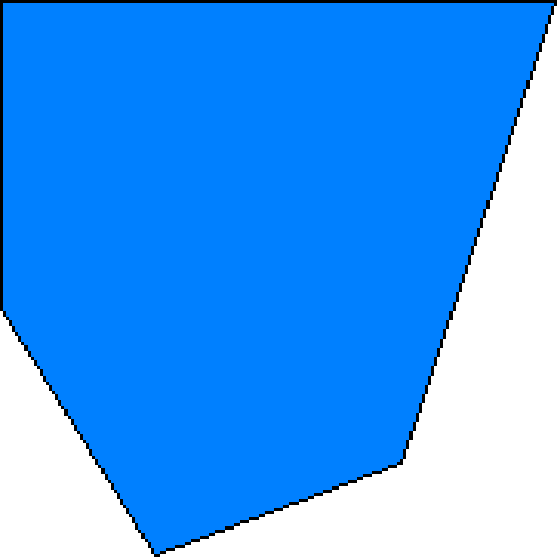
# fill\_area(150, 120, [255, 255, 255], [0, 128, 255])

# plt.imshow(board)

# plt.axis('off')

# plt.show()

# Output

****

**Conclusion**

This experiment demonstrates how basic raster graphics algorithms can be implemented at the pixel level using NumPy arrays. We successfully implemented polygon drawing and flood fill to render and color shapes, simulating low-level computer graphics.