**Graphics and multimedia**

**Polygon Clipping using Sutherland–Hodgman Algorithm**

**EXPERIMENT** : 4

**ROLL NO** : 231701012

**NAME** : DIVYA DHARRSHAN T

**--------------------------------------------------------------------------------------------------------------------------------------**

**AIM**

To write a program that clips a polygon to a specified rectangular window using the **Sutherland–Hodgman Polygon Clipping Algorithm** and displays the clipped polygon.

**Procedure**

1. **Input:**
   * Vertices of the polygon.
   * Clipping window boundaries (left, right, top, bottom).
2. **Process:**
   * Clip the polygon edges one by one against each boundary of the window (LEFT, RIGHT, BOTTOM, TOP).
   * For each clipping edge, retain only the portion of the polygon that lies inside.
   * Use intersection points when polygon edges cross the clipping boundary.
3. **Output:**
   * Display both the original polygon and the clipped polygon along with the clipping window.

**PROGRAM**

import matplotlib.pyplot as plt

LEFT, RIGHT, BOTTOM, TOP = 0, 1, 2, 3

def inside(p, edge, clip\_win):

x, y = p

xmin, xmax, ymin, ymax = clip\_win

if edge == LEFT:

return x >= xmin

elif edge == RIGHT:

return x <= xmax

elif edge == BOTTOM:

return y >= ymin

elif edge == TOP:

return y <= ymax

def intersect(p1, p2, edge, clip\_win):

xmin, xmax, ymin, ymax = clip\_win

x1, y1 = p1

x2, y2 = p2

if edge == LEFT:

x = xmin

y = y1 + (y2 - y1) \* (xmin - x1) / (x2 - x1)

elif edge == RIGHT:

x = xmax

y = y1 + (y2 - y1) \* (xmax - x1) / (x2 - x1)

elif edge == BOTTOM:

y = ymin

x = x1 + (x2 - x1) \* (ymin - y1) / (y2 - y1)

elif edge == TOP:

y = ymax

x = x1 + (x2 - x1) \* (ymax - y1) / (y2 - y1)

return (x, y)

def clip\_polygon(polygon, clip\_win):

output = polygon

for edge in [LEFT, RIGHT, BOTTOM, TOP]:

input\_list = output

output = []

if not input\_list:

break

s = input\_list[-1]

for p in input\_list:

if inside(p, edge, clip\_win):

if not inside(s, edge, clip\_win):

output.append(intersect(s, p, edge, clip\_win))

output.append(p)

elif inside(s, edge, clip\_win):

output.append(intersect(s, p, edge, clip\_win))

s = p

return output

def draw\_polygon(points, color, label):

x, y = zip(\*(points + [points[0]]))

plt.plot(x, y, color=color, label=label)

clip\_window = (100, 300, 100, 300)

polygon = [(50, 150), (200, 50), (350, 150), (350, 300), (250, 350), (150, 300)]

clipped\_poly = clip\_polygon(polygon, clip\_window)

plt.figure(figsize=(8, 8))

draw\_polygon(polygon, 'blue', "Original Polygon")

draw\_polygon([(clip\_window[0], clip\_window[2]),

(clip\_window[1], clip\_window[2]),

(clip\_window[1], clip\_window[3]),

(clip\_window[0], clip\_window[3])],

'black', "Clipping Window")

draw\_polygon(clipped\_poly, 'red', "Clipped Polygon")

plt.legend()

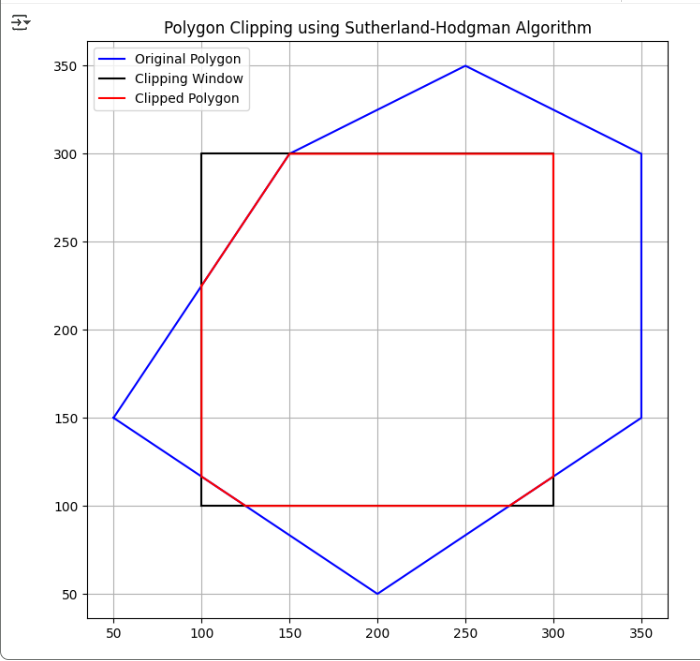
plt.title("Polygon Clipping using Sutherland-Hodgman Algorithm")

plt.grid(True)

plt.axis("equal")

plt.show()

**OUTPUT**



**Result**

The polygon was successfully clipped against the given rectangular clipping window using the **Sutherland–Hodgman Polygon Clipping Algorithm** and the clipped polygon was displayed.