**Batch:** C2 **Roll No.:** 110

**Experiment No. 06**

**TITLE**: Write a program to demonstrate the Polygon CLIPPING algorithm

**AIM:**
Write a program to demonstrate the Polygon CLIPPING algorithm
(Implement using Sutherland Hodgeman polygon clipping algorithm)
VLab
https://cse18-iiith.vlabs.ac.in/exp/clipping-polygon/

**Expected OUTCOME of Experiment:**
Sutherland Hodgeman polygon clipping algorithm

**Books/ Journals/ Websites referred:**
https://www.geeksforgeeks.org/weiler-atherton-polygon-clipping-algorithm/

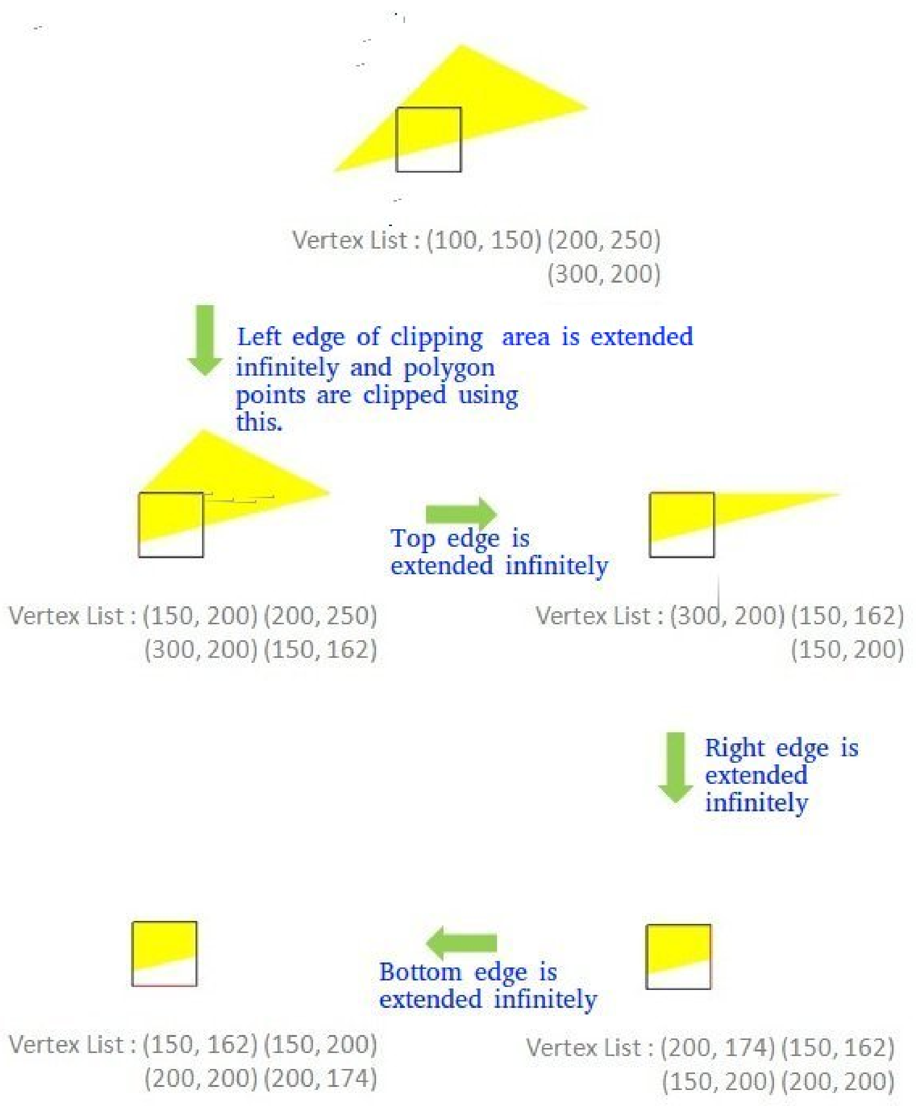
**Algorithm/ Pseudocode for each process:**

## Sutherland-Hodgeman Polygon Clipping Algorithm :

- Read coordinates of all vertices of the polygon.
- Read coordinates of the clipping window
- Consider the left edge of the window
- Compare the vertices of each edge of the polygon, individually with the clipping plane
- Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary discussed earlier.
- Repeat the steps 4 and 5 for remaining edges of the clipping

window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.

- Stop.

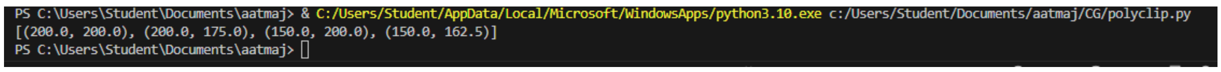

Vertex List : (100, 150) (200, 250) (300, 200)
Left edge of clipping area is extended infinitely and polygon points are clipped using this.
Vertex List : (150, 200) (200, 250) (300, 200) (150, 162)
Top edge is extended infinitely
Vertex List : (300, 200) (150, 162) (150, 200)
Right edge is extended infinitely
Vertex List : (200, 174) (150, 162) (150, 200) (200, 200)
Bottom edge is extended infinitely
Vertex List : (150, 162) (150, 200) (200, 200) (200, 174)

**Implementation details:**

```
polygon = [(100,150), (200,250), (300,200)]
clipping = [(150,150), (150,200), (200,200), (200,150)] #works only
for squares. else edit isin
```

```
def isintwo(p1):
    x=p1[0]
    y=p1[1]
    if(x in range(clipping[0][0],clipping[2][0]+1) and y in
range(clipping[0][1],clipping[2][1]+1) ):
            return True
    return False
def isin(p1,case):
    if(case==0):
        #left
        return p1[0]>clipping[0][0]
    if(case==1):
        #top
        return p1[1]<clipping[1][1]
    if(case==2):
        #right
        return p1[0]<clipping[1][1]
    if(case==3):
        #bottom
        return p1[1]>clipping[0][0]

def intersect(p1,p2,p3,p4,case):

    x= ( (p1[0]*p2[1]-p1[1]*p2[0])*(p3[0]-p4[0]) -
(p1[0]-p2[0])*(p3[0]*p4[1]-p3[1]*p4[0]) ) /
((p1[0]-p2[0])*(p3[1]-p4[1]) - (p1[1] - p2[1])*(p3[0]-p4[0]))
    y= ( (p1[0]*p2[1]-p1[1]*p2[0])*(p3[1]-p4[1]) -
(p1[1]-p2[1])*(p3[0]*p4[1]-p3[1]*p4[0]) ) /
((p1[0]-p2[0])*(p3[1]-p4[1]) - (p1[1] - p2[1])*(p3[0]-p4[0]))
    ans=[] # of points
    if(isin(p1,case) and isin(p2,case)):
       #print("case1")
       ans.append(p2)
    elif(isin(p1,case) and not isin(p2,case)):
        #print("case2")
        ans.append((x,y))
    elif(not isin(p1,case) and isin(p2,case)):
        #print("case3")
        ans.append((x,y))
        ans.append(p2)
    else:
        ans = -1
        #print(p1)
    return ans

answer = polygon
for i in range(0,4):
    answer_temp=[]
    for j in range(len(answer)):

         try:
             val = intersect(answer[j],answer[(j+1)%len(answer)
],clipping[i%4],clipping[(i+1) %4],i)
         except:
             val=(answer[j],answer[(j+1)%len(answer)])
         if(val != -1):
            for vali in val:
               answer_temp.append(vali)

    answer = list(set(answer_temp))

print(answer)
```

**Output(s) (Screen Shot):**



```
PS C:\Users\Student\Documents\aatmaj> & C:/Users/Student/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Users/Student/Documents/aatmaj/CG/polyclip.py
[(200.0, 200.0), (200.0, 175.0), (150.0, 200.0), (150.0, 162.5)]
PS C:\Users\Student\Documents\aatmaj> 
```

**[(200.0, 200.0), (200.0, 175.0), (150.0, 200.0), (150.0, 162.5)]**

**Conclusion and discussion:**
Thus we have implemented the sutherland hodgeman algo and understood its working. The cohen hogeman polygon clipping algorithm relies on the points and their intersections with the lines. It does not work well for convex polygons.

**Date: 27 sept 2023**

**Signature of faculty in-charge**

## Post lab

Explain Wiler-Atherton algorithm, Implement.

```
1. First make a list of all intersection points namely i1,
i2, i3, ...
 2. Classify those intersection points as entering or exiting.
 3. Now, make two lists, one for the clipping polygon, and the other
```

for the clipped polygon.

4. Fill both the lists up in such a way that the intersection points lie between the correct vertices of each of the polygon. That is the clipping polygon list is filled up with all the vertices of the clipping polygon along with the intersecting points lying between the corresponding vertices.
5. Now, start at the 'to be clipped' polygon's list.
6. Choose the first intersection point which has been labelled as an entering point. Follow the points in the list (looping back to the top of the list, in case the list ends) and keep on pushing them into a vector or something similar of the sorts. Keep on following the list until an exiting intersection point is found.
7. Now switch the list to the 'polygon that is clipping' list, and find the exiting the intersection that was previously encountered. Now keep on following the points in this list (similar to how we followed the previous list) until the entering intersection point is found (the one that was found in the previous 'to be clipped' polygon's list).
8. This vector now formed by pushing all the encountered points in the two lists, is now the clipped polygon (one of the many clipped polygons if any of the clipping polygons is concave).
9. Repeat this clipping procedure (i.e. from step 5) until all the entering intersection points have been visited once.