Polygon Clipping

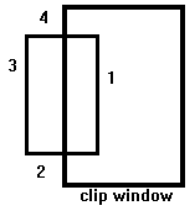
Polygon clipping differs from line clipping in several respects.

1. The input to the clipper is a polygon, which for simplicity we will view as a list of $n \geq 3$ vertices $(v_0, v_1, \ldots, v_{n-1})$.
2. The output from the clipper is one or more polygons.
3. The clipping process may generate vertices that do not lie on any of the edges of the orginal polygon.
4. Complex polygons (that is, non-convex) may lead to strange artifacts.

1. Sutherland-Hodgman Polygon Clipping

Four Types of Edges

As the algorithm goes around the edges of the window, clipping the polygon, it encounters four types of edges. All four edge types are illustrated by the polygon in the following figure. For each edge type, zero, one, or two vertices are added to the output list of vertices that define the clipped polygon.



The four types of edges are:

1. Edges that are totally inside the clip window. - add the second inside vertex point
2. Edges that are leaving the clip window. - add the intersection point as a vertex
3. Edges that are entirely outside the clip window. - add nothing to the vertex output list
4. Edges that are entering the clip window. - save the intersection and inside points as vertices

How To Calculate Intersections

Assume that we're clipping a polgon's edge with vertices at (x1,y1) and (x2,y2) against a clip window with vertices at (xmin, ymin) and (xmax,ymax).
The location (IX, IY) of the intersection of the edge with the left side of the window is:

i. IX = xmin
ii. IY = slope*(xmin-x1) + y1, where the slope = (y2-y1)/(x2-x1)

The location of the intersection of the edge with the right side of the window is:

i. IX = xmax
ii. IY = slope*(xmax-x1) + y1, where the slope = (y2-y1)/(x2-x1)

The intersection of the polygon's edge with the top side of the window is:

i. IX = x1 + (ymax - y1) / slope
ii. IY = ymax

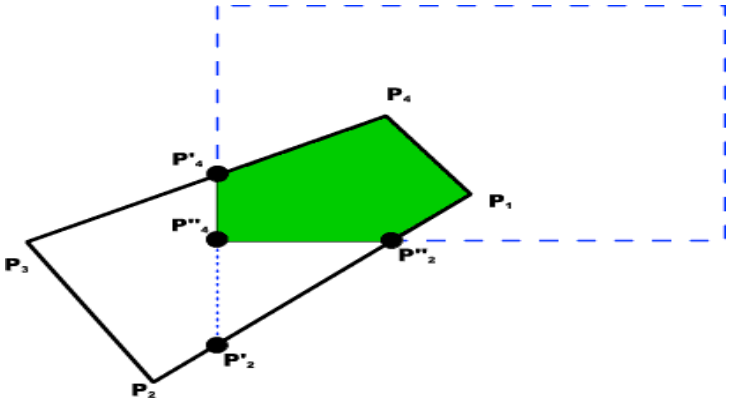
Finally, the intersection of the edge with the bottom side of the window is:

i. IX = x1 + (ymin - y1) / slope
ii. IY = ymin

Steps of Sutherland-Hodgman's polygon-clipping algorithm

- Polygons can be clipped against each edge of the window one at a time. Windows/edge intersections, if any, are easy to find since the X or Y coordinates are already known.
- Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges.
- Note that the number of vertices usually changes and will often increases.
- We are using the Divide and Conquer approach.

Sutherland - Hodgeman : An Example




```
in -> [clip left]  -> [clip right]   -> [clip bottom]    -> [clip top] -> out
P1 - in to in --> P1  - in to in --> P1  - in to in --> P1   - in to in --> P1
P2 - in to out -> P'2 - in to in --> P'2 - in to out -> P''2 - in to in --> P''2
P3 - out to out -> x
P4 - out to in -> P'4 - in to in --> P'4 - out to in -> P''4 - in to in --> P''4
                                                      -> P'4 - in to in --> P'4
               -> P4  - in to in --> P4  - in to in --> P4   - in to in --> P4
```

Some Problems With This Algorithm

1. This algorithm does not work if the clip window is not convex.
2. If the polygon is not also convex, there may be some dangling edges.

2. Weiler-Atherton Polygon Clipping algorithm

A more formal statement of the algorithm is [3]

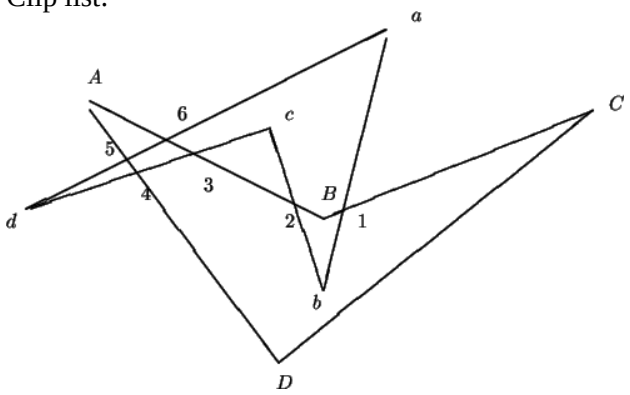
- **Determine the intersections of the subject and clip polygons** - Add each intersection to the SP and CP vertex lists. Tag each intersection vertex and establish a bidirectional link between the SP and CP lists for each intersection vertex.
- **Process nonintersecting polygon borders** - Establish two holding lists: one for boundaries which lie inside the CP and one for boundaries which lie outside. Ignore CP boundaries which are outside the SP. CP boundaries inside the SP form holes in the SP. Consequently. a copy of the CP boundary goes on both the inside and the outside holding list. Place the boundaries on the appropriate holding list.
- **Create two intersection vertex lists** - One, the entering list, contains only the intersections for the SP edge entering the inside of the CP. The other, the leaving list, contains only the intersections for the SP edge leaving the inside of the CP. The intersection type will alternate along the boundary. Thus, only one determination is required for each pair of intersections.

- **Perform the actual clipping** -
  Polygons inside the CP are found using the following procedure.
  - Remove an intersection vertex from the entering list. If the list is empty, the process is complete.
  - Follow the SP vertex list until an intersection is found. Copy the SP list upto this point to the inside holding list.
  - Using the link, jump to the CP vertex list.
  - Follow the CP vertex list until an intersection is found. Copy the CP vertex list upto this point to the inside holding list.
  - Jump back to the SP vertex list.
  - Repeat until the starting point is again reached. At this point, the new inside polygon has been closed.

Weiler-Atherton Polygon Clipper

- Assume the vertices of the subject polygon are listed in clockwise order (interior is on the right)
- Start at an entering intersection
- Follow the edge(s) of the polygon being clipped until an exiting intersection is encountered
- Turn right at the exiting intersection and following clip window edge until intersection is found
- Turn right and follow the subject polygon
- Continue until vertex already visited is reached
- If entire polygon has not be processed, repeat
- Consider the subject polygon with vertices a, b, c, d and the clip polygon with vertices A, B, C, D

- Insert the intersections in both vertex lists
  - Subject list: $a, 1, b, 2, c, 3, 4, d, 5, 6$
  - Clip list: $A, 6, 3, 2, B, 1, C, D, 4, 5$



- Starting at vertex *a* of the clip polygon, find 1 is first entering intersection
- Traversing the subject, find 2 is exiting intersection
- ``Jump" to vertex 2 in clip polygon, follow until vertex 1 (which has be visited)
  $1, b, 2, B$
- Output clipped list
- Jump back to subject list, restarting at *c*, find 3 is entering intersection
- Traversing the subject, find 4 is exiting intersection
- Jump to vertex 4 in clip polygon, follow until vertex 5 (which is entering)
- Jump to subject, at vertex 5, find 6 is exiting
- Jump to clip, at vertex 6, find 3 is visited
  $3, 4, 5, 6$
- Output clipped list
- All entering intersections have been visited