POLYGON CLIPPING

A polygon is generally stored as a collection of vertices. A clipping algorithm takes one collection, and outputs a new collection. A clipped polygon, is also a polygon. The clipped polygon often will have more vertices than the unclipped one, but it can also have the same number, or less. If the unclipped polygon lies completely outside the clipping boundary, the clipped polygon even has zero vertices.

TYPES OF POLYGON:

CONVEX: If the line joining the 2 vertices that lie inside the polygon is also completely inside the polygon then the polygon is said to be convex.

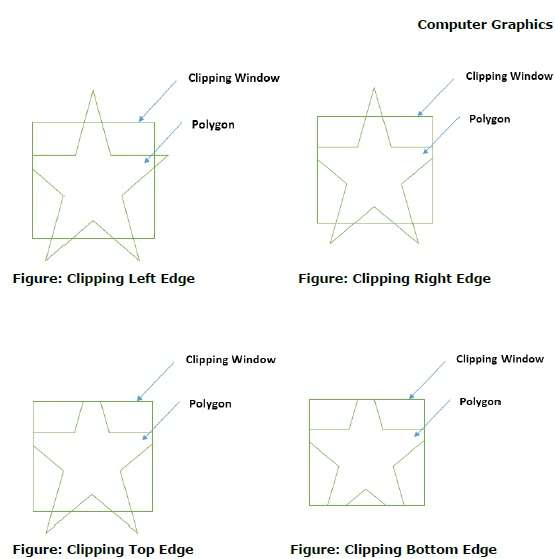
CONCAVE: If the line joining the 2 vertices that lie inside the polygon is partially outside the polygon then the polygon is said to be convex.

POLYGON CLIPPING ALGORITHMS:

1. Sutherland-Hodgman algorithm.
2. Weiler–Atherton algorithm.

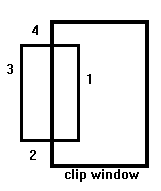
SUTHERLAND -HODGMAN ALGORITHM:

Sutherland-Hodgman algorithm was introduced in 1974. It uses a divide-and-conquer strategy to attack the problem. It clips a polygon against a single infinite clip edge. Four clip edges, each defining one boundary of the clip rectangle, successively clip a polygon against a clip rectangle.



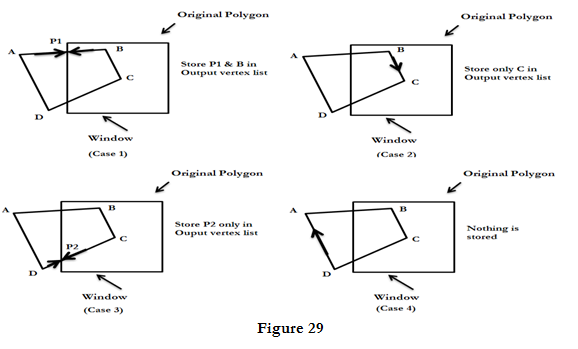
PROCEDURE TO CLIP A POLYGON:  
The edge (of clipping area) is extended infinitely to create a boundary and all the vertices are clipped using this boundary. The new list of vertices generated is passed to the next edge of the clip polygon in clockwise fashion until all the edges have been used.

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. **Both vertices are inside :**Only the second vertex is added to the output list.
2. **First vertex is outside while second one is inside :**Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list.
3. **First vertex is inside while second one is outside :**Only the point of intersection of the edge with the clip boundary is added to the output list.
4. **Both vertices are outside :**No vertices are added to the output list.



## 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 (X, Y) of the intersection of the edge with the left side of the window is:

1. X = xmin
2. Y = 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:

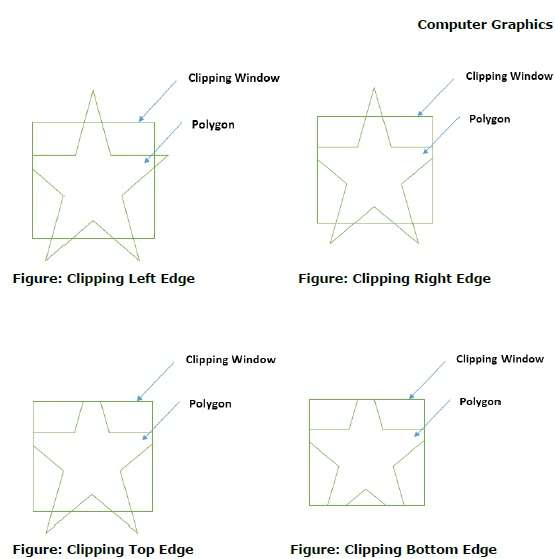
1. X = xmin
2. Y = slope\*(xmax-x1) + y1, where the slope = (y2-y1)/(x2-x1)

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

1. X = x1 + (ymax - y1) / slope
2. Y = ymax

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

1. X = x1 + (ymin - y1) / slope
2. Y = ymin



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.

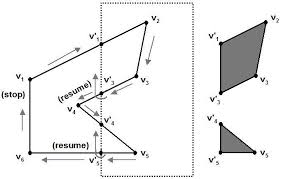
WEILER-ATHERTON POLYGON CLIPPING:

It is powerful but somewhat more complex clipping algorithm developed by weiler and Atherton to properly clip a concave polygon. This algorithm defines the Polygon to be clipped as a subject Polygon and the clipping region is the clip Polygon.

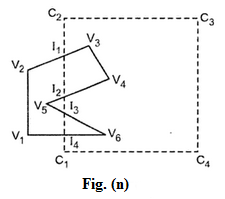
The algorithm describes both the subject and the clip polygon by a circular list of vertices. The boundaries of the subject polygon and the clip polygon may or may not intersect. If they intersect, then the intersections occur in pairs. One of the intersections occurs when a subject polygon edge enters the inside of the clip Polygon and one when it leaves.

Two cases arises:

* 1. Outside to inside: save the intersection point and continue in clockwise direction along the subject polygon boundary.
  2. Inside to outside: save the intersection point and move in clockwise direction along the clip polygon boundary.



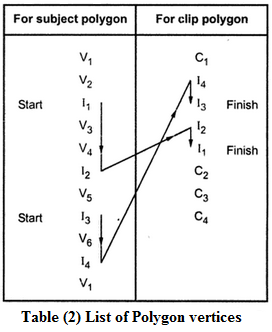
The algorithm describes both the subject and the clip polygon by a circular list of vertices. The boundaries of the subject polygon and the clip polygon may or may not intersect. If they intersect, then the intersections occur in pairs. One of the intersections occurs when a subject polygon edge enters the inside of the clip Polygon and one when it leaves. As shown in the figure (n),



there are four intersection vertices I1, I2 ,I3 and I4. In these intersections I1 and 13 are entering intersections, and I2 and I3 are leaving intersections. The clip polygon vertices are marked as C1, C2, C3 and C4.

In this algorithm two separate vertices lists are made one for **clip polygon** and one for **subject polygon** including intersection points. The Table 2 shows these two lists for polygons shown in figure (n).

The algorithm starts at an entering intersection (I1) and follows the subject polygon vertex list in the downward direction (i.e. I1, V3, V4, l2). At the occurrence of leaving intersection the algorithm follows the clip polygon vertex list from the leaving intersection vertex in the downward direction (i.e. I2, I1). At the occurrence of the entering intersection the algorithm follows the subject polygon vertex list from the entering intersection vertex.



This process is repeated until we get the starting vertex. This Process we have to repeat for all remaining entering intersections which are not included in the previous traversing of vertex list. In our example, entering vertex I3 was not included in the first traversing of vertex list, Therefore, we have to go for another vertex traversal from vertex I3.

The above two vertex traversal gives two clipped inside polygons. There are: - **I1, V3, V4, I2, I1 and I3, V6, I4, I3**

TEXT CLIPPING:

There are several techniques that can be used to provide text clipping in a graphics package. The clipping technique used will depend on the methods used to generate characters and the requirements of a particular application.

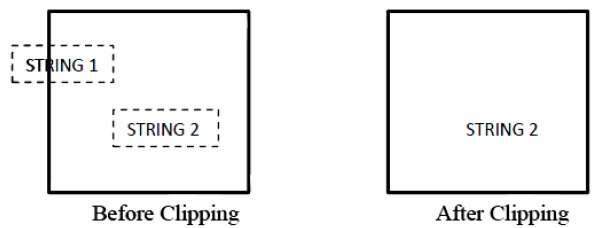
There are three methods for text clipping which are listed below −

* All or none string clipping
* All or none character clipping
* Text clipping

ALL OR NONE STRING CLIPPING:

If all of the string is inside the clip window, we keep it. Otherwise, the string is discarded. This procedure is implemented by considering a bounding rectangle around the text pattern. The boundary positions of the rectangle are then compared to the window boundaries, and the string is rejected if there is any overlap. This method produces the fastest text clipping.

The following figure shows all or none string clipping −

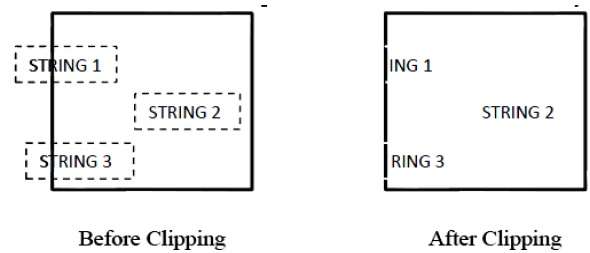


In all or none string clipping method, either we keep the entire string or we reject entire string based on the clipping window. As shown in the above figure, STRING2 is entirely inside the clipping window so we keep it and STRING1 being only partially inside the window, we reject.

ALL OR NONE CHARACTER CLIPPING :

An alternative to rejecting an entire character string that overlaps a window boundary is to use the all-or-none character-clipping strategy. Here we discard only those characters that are not completely inside the window. In this case the boundary limits of individual characters are compared to the window. Any character that either overlaps or is outside a window boundary is clipped.

The following figure shows all or none character clipping −



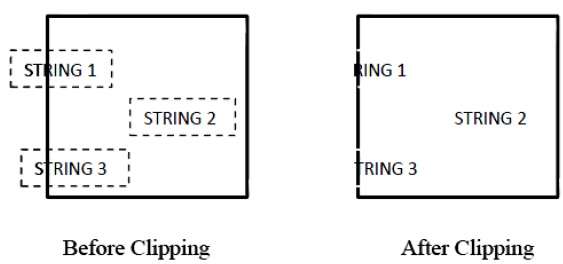
This clipping method is based on characters rather than entire string. In this method if the string is entirely inside the clipping window, then keep it. If it is partially outside the window, then −

* Reject only the portion of the string being outside
* If the character is on the boundary of the clipping window, then discard that entire character and keep the rest string.

TEXT CLIPPING :

Text clipping is used to clip the components of of individual characters. Characters are treated in much the same way that we treated lines. If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window.

The following figure shows text clipping −



This clipping method is based on characters rather than the entire string. In this method if the string is entirely inside the clipping window, then we keep it. If it is partially outside the window, then

* You reject only the portion of string being outside.
* If the character is on the boundary of the clipping window, then we discard only that portion of character that is outside of the clipping window.