Introduction to Computer Graphics

(C S 6 0 2)

Lecture 15
Clipping-II

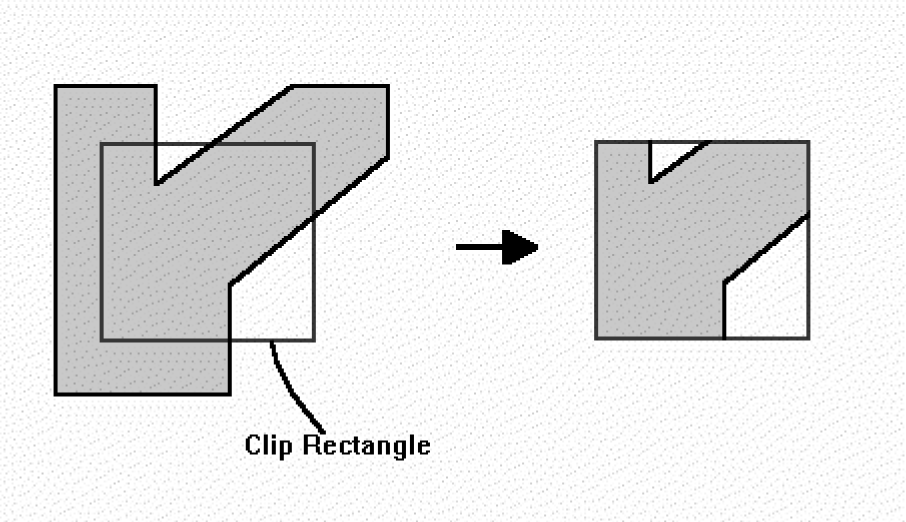
Introduction

15.1 Polygon Clipping

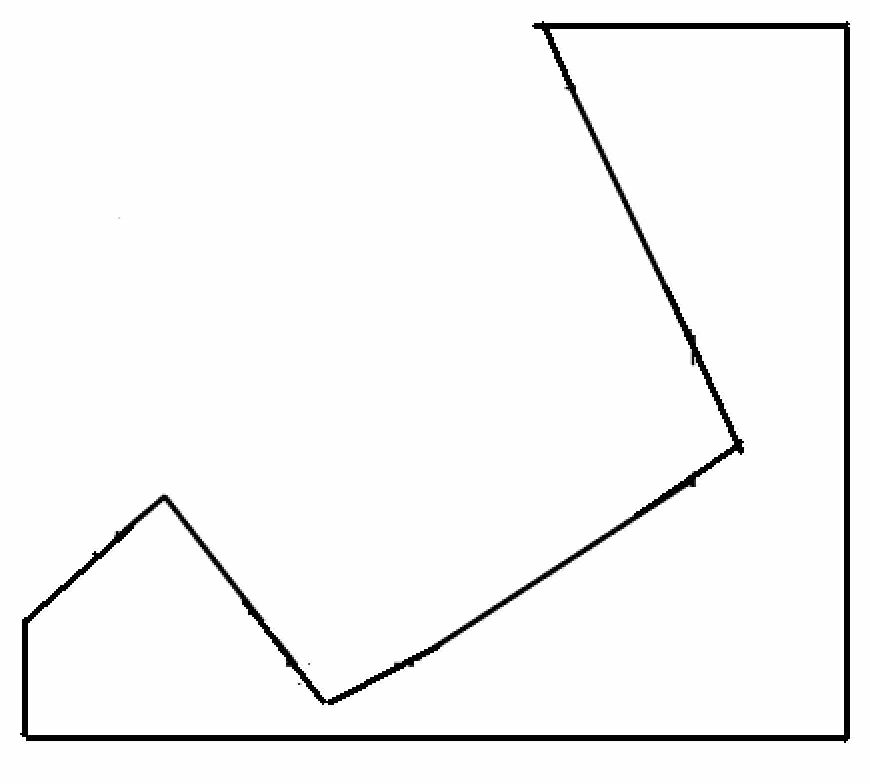
A polygon is usually defined by a sequence of vertices and edges. If the polygons are un-filled, line-clipping techniques are sufficient however, if the polygons are filled, the process in more complicated. A polygon may be fragmented into several polygons in the clipping process, and the original colour associated with each one. The Sutherland-Hodgeman clipping algorithm clips any polygon against a convex clip polygon. The Weiler-Atherton clipping algorithm will clip any polygon against any clip polygon. The polygons may even have holes.

An algorithm that clips a polygon must deal with many different cases. The case is particularly note worthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of the polygon must be tested against each edge of the clip rectangle; new edges must be added, and existing edges must be discarded, retained, or divided. Multiple polygons may result from clipping a single polygon. We need an organized way to deal with all these cases.

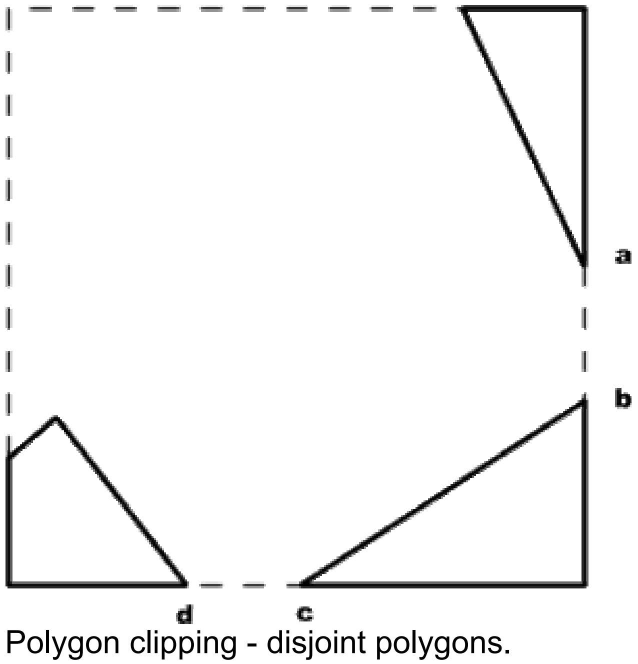
The following example illustrates a simple case of polygon clipping.



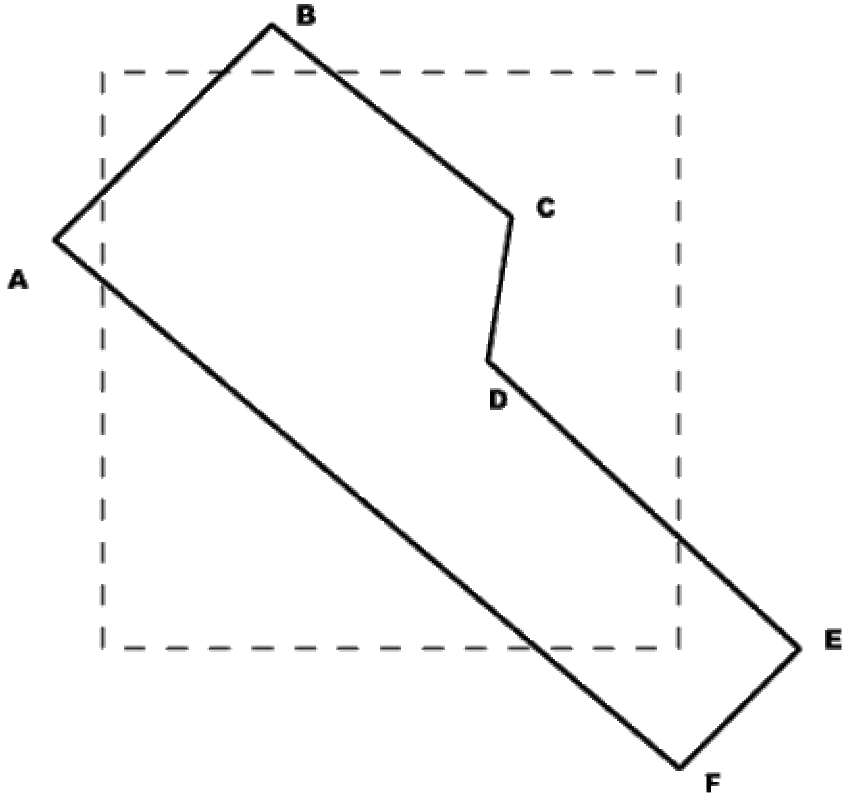
Given below are some examples to elaborate further.



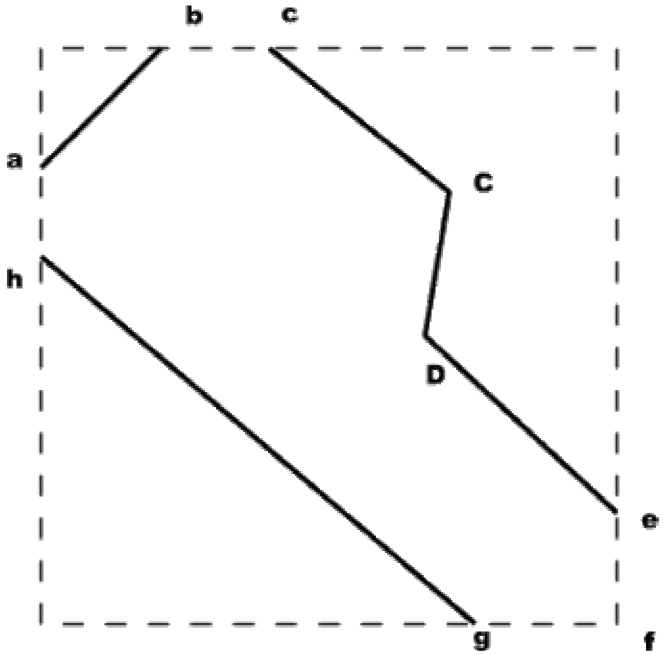
Polygon clipping - disjoint polygons.



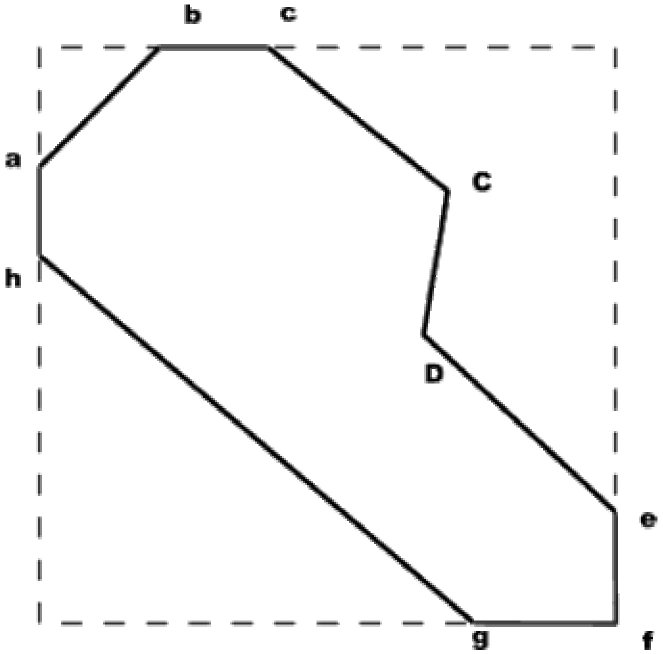
Polygon clipping - disjoint polygons.



Polygon clipping - open polygons.



Polygon clipping - open polygons.



Polygon clipping - open polygons.

15.2 Sutherland and Hodgman's polygon-clipping algorithm

Sutherland and Hodgman's polygon-clipping algorithm uses a divide-and-conquer strategy: It solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip 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.

Note the difference between this strategy for a polygon and the Cohen-Sutherland algorithm for clipping a line: The polygon clipper clips against four edges in succession, whereas the line clipper tests outcode to see which edge is crossed, and clips only when necessary.

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 increase.
- We are using the Divide and Conquer approach.

Here is a STEP-BY-STEP example of polygon clipping.

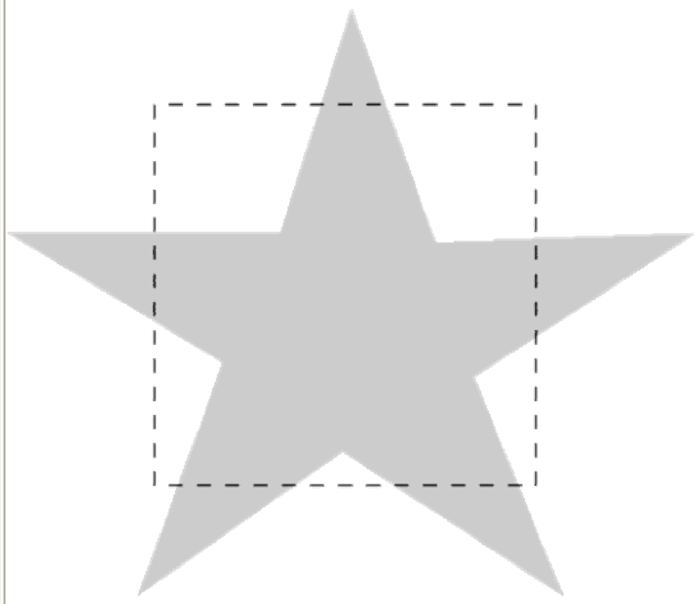
Four Cases of polygon clipping against one edge

The clip boundary determines a visible and invisible region. The edges from vertex i to vertex i+1 can be one of four types:

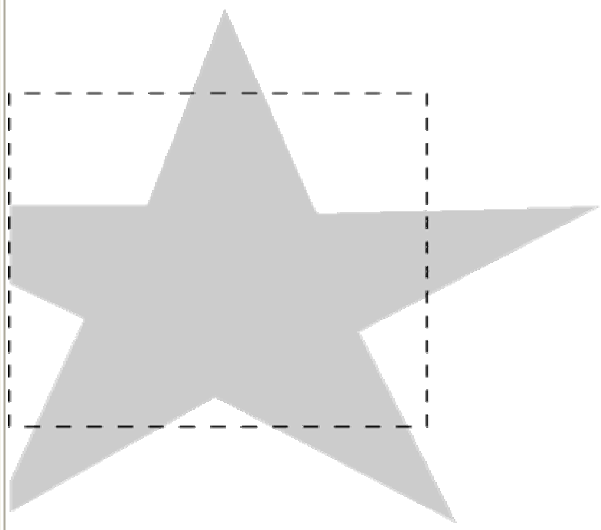
- Case 1 : Wholly inside visible region - save endpoint
- Case 2 : Exit visible region - save the intersection
- Case 3 : Wholly outside visible region - save nothing
- Case 4 : Enter visible region - save intersection and endpoint

Because clipping against one edge is independent of all others, it is possible to arrange the clipping stages in a pipeline. The input polygon is clipped against one edge and any points that are kept are passed on as input to the next stage of the pipeline. In this way four polygons can be at different stages of the clipping process simultaneously. This is often implemented in hardware.

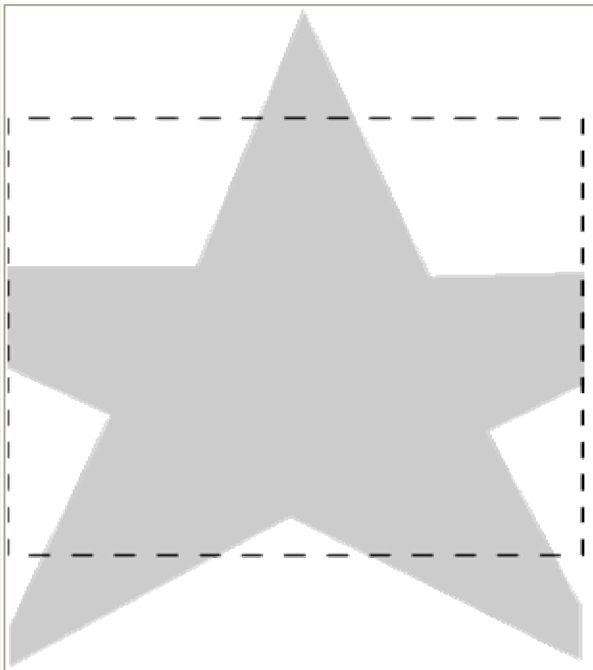
Example No # 1 Clipping a Polygon



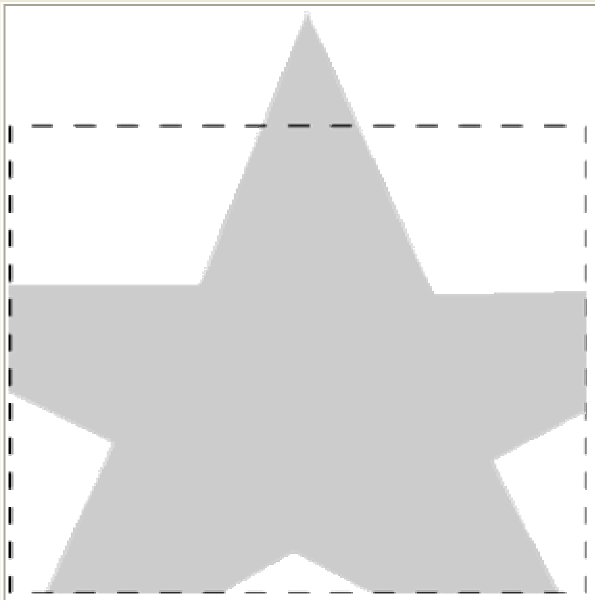
Original polygon



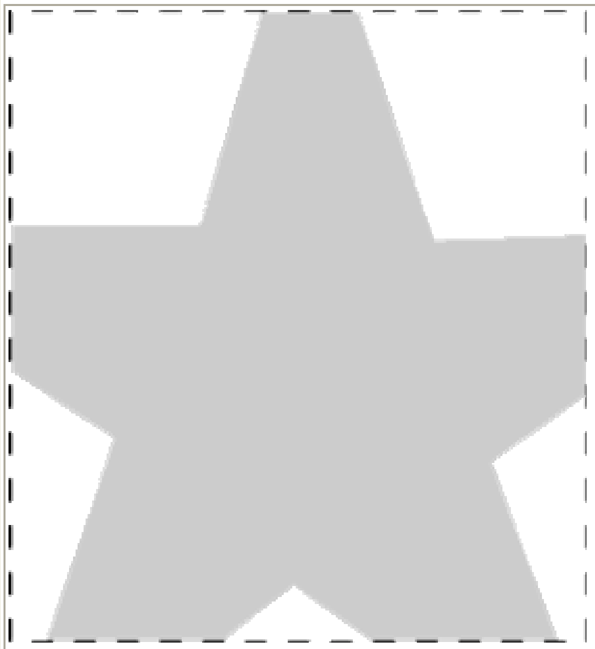
Clip Left



Clip Right



Clip Bottom

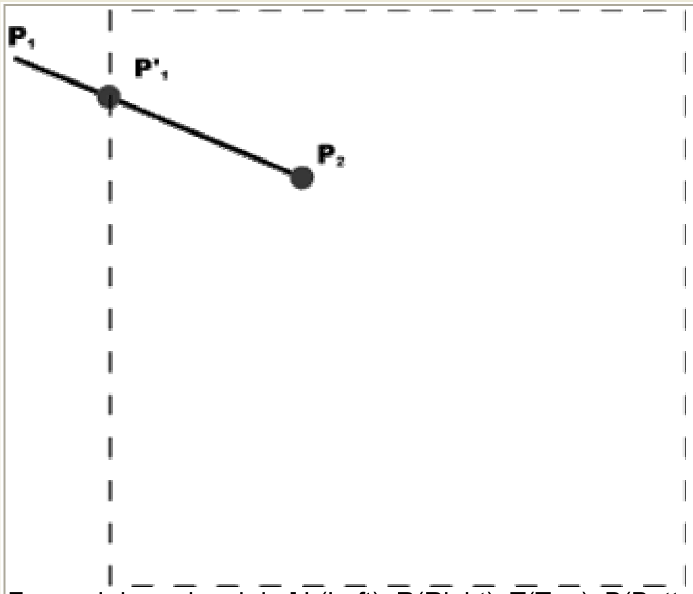


Clip Top

Example No #2 Clipping a Rectangle

If Clipping Rectangle is denoted by dashed lines and Line is defined by using points P1 and P2

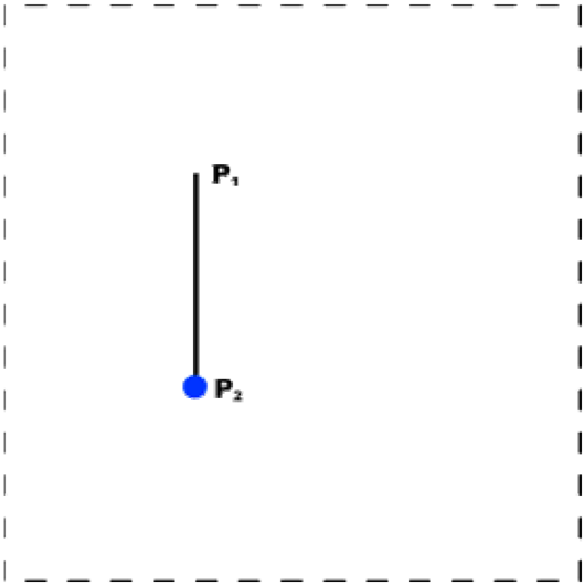
Case i



For each boundary b in [L(Left), R(Right), T(Top), B(Bottom)]
If P_1 outside and P_2 inside.
Output:

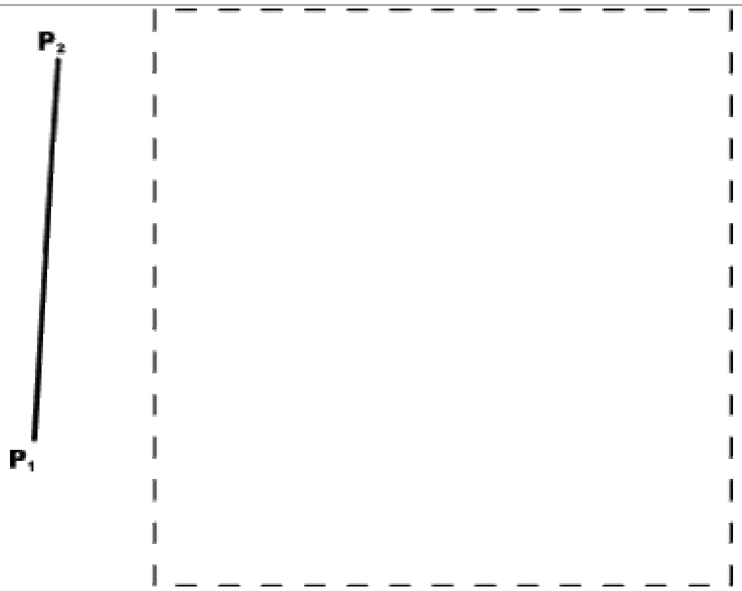
1. intersection Point (**P1'**)
2. Point $\mathbf{P_2}$

Case ii



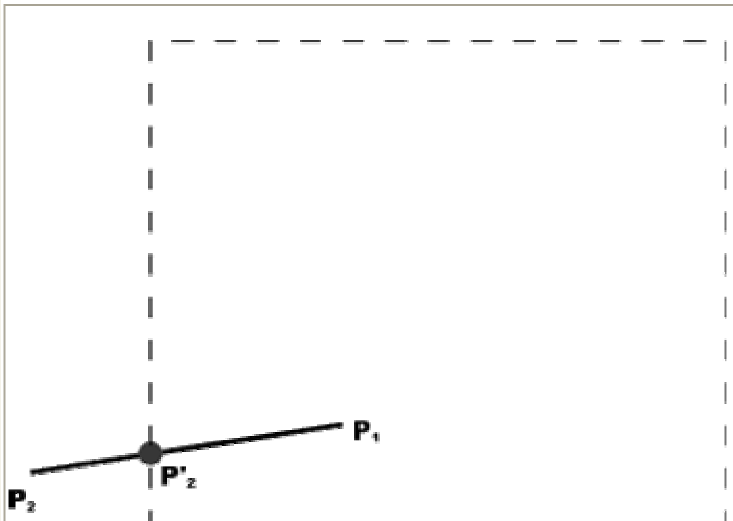
For each boundary b in [L(Left), R(Right), T(Top), B(Bottom)]
If P_1 inside and P_2 inside
Output:
Point $\mathbf{P_2}$

Case iii



For each boundary b in [L(Left), R(Right), T(Top), B(Bottom)]
If P1 outside and P2 outside
Do nothing

Case iv



For each boundary b in [L(Left), R(Right), T(Top), B(Bottom)]
If P_1 inside and P_2 outside (We are going from P1 to P2)
Output:

Point of intersection (**P2’**) only.

Pipeline Clipping Approach
An array, *s* records the most recent point that was clipped for each clip-window boundary. The main routine passes each vertex *p* to the *clipPoint* routine for clipping against the first window boundary. If the line defined by endpoints p and s (boundary) crosses this window boundary, the intersection is calculated and passed to the next clipping stage. If *p* is inside the window, it is passed to the next clipping stage. Any point that survives clipping against all window boundaries is then entered into the output array of points. The array *firstPoint* stores for each window boundary the first point flipped against that boundary. After all polygon vertices have been processed, a closing routine clips lines defined by the first and last points clipped against each boundary.

Shortcoming of Sutherlands -Hodgeman Algorithm
Convex polygons are correctly clipped by the Sutherland-Hodegeman algorithm, but concave polygons may be displayed with extraneous lines. This occurs when the clipped polygon should have two or more separate sections. But since there is only one output vertex list, the last vertex in the list is always joined to the first vertex. There are several things we could do to correct display concave polygons. For one, we could split the concave polygon into two or more convex polygons and process each convex polygon separately.

Another approach to check the final vertex list for multiple vertex points along any clip window boundary and correctly join pairs of vertices. Finally, we could use a more general polygon clipper, such as wither the Weiler-Atherton algorithm or the Weiler algorithm described in the next section.

15.3 Weiler-Atherton Polygon Clipping

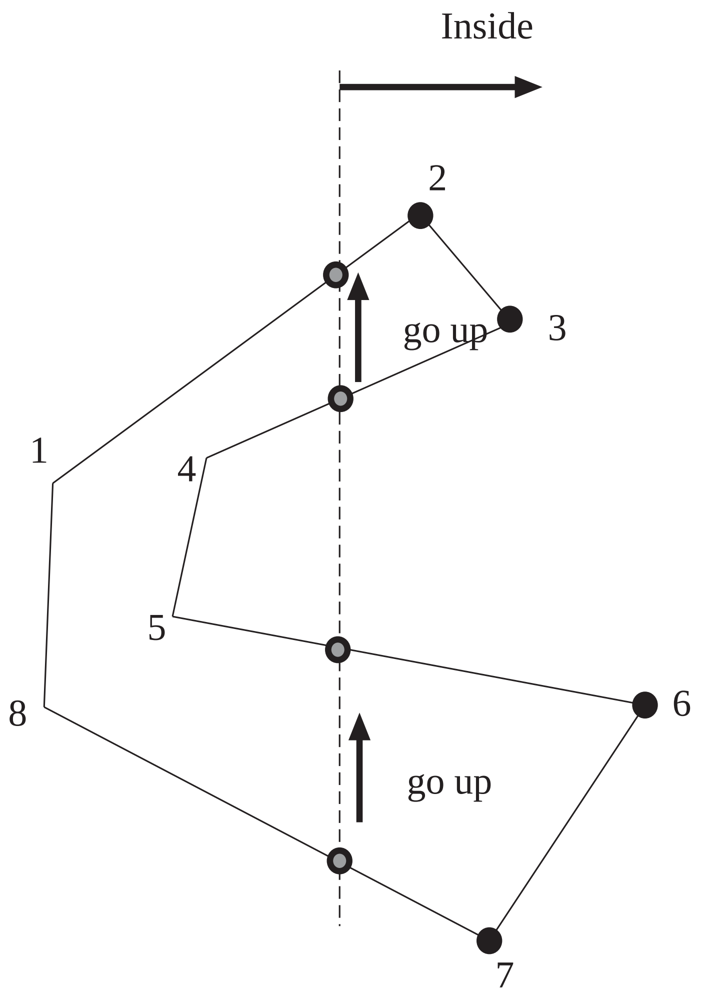
In this technique, the vertex-processing procedures for window boundaries are modified so that concave polygons are displayed correctly. This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon-clipping regions.

The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction(clockwise or counterclockwise) and whether the pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside-to-outside pair. For clockwise processing of polygon vertices, we use the following rules:

- For an outside-top inside pair of vertices, follow the polygon boundary

- For an inside-to-outside pair of vertices, follow the window boundary in a clockwise direction

In following figure, the processing direction in the Wieler-Atherton algorithm and the resulting clipped polygon is shown for a rectangular clipping window.


Inside
2
go up
3
1
4
5
6
8
go up
7