7.4 Polygon clipping

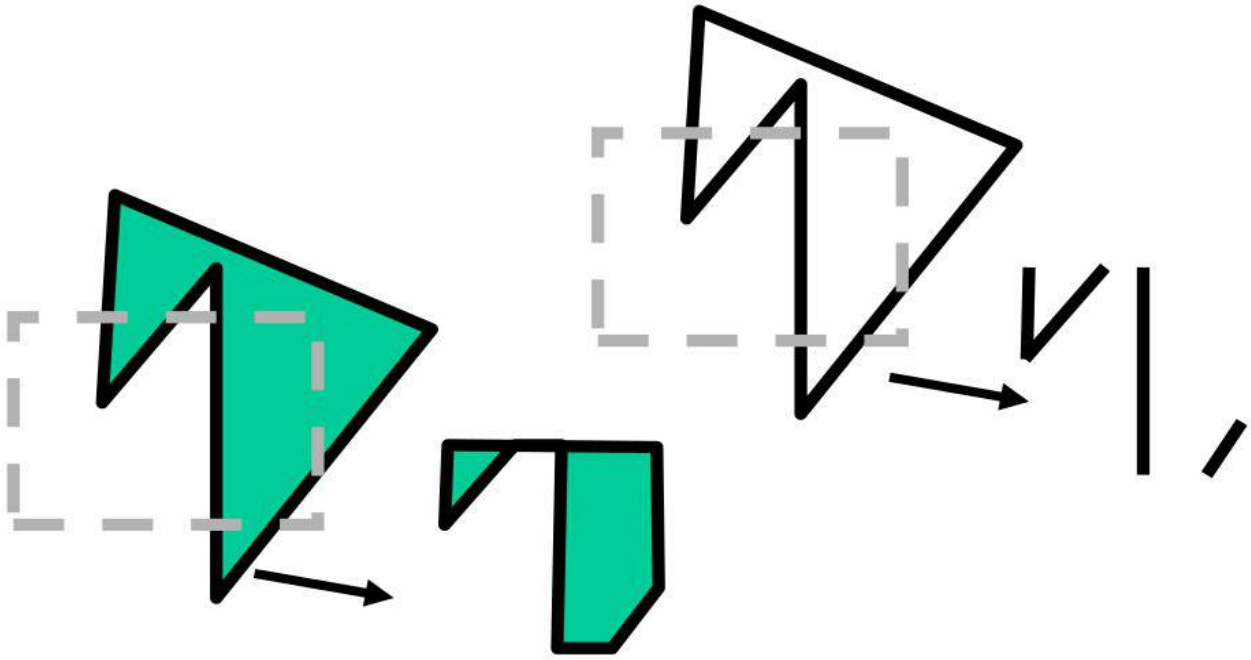
Clipping polygons is more complex than clipping the individual lines

Input: polygon

Output: original polygon, new polygon, or nothing

Since polygons are bounded by line segments, can we just use line clipping?

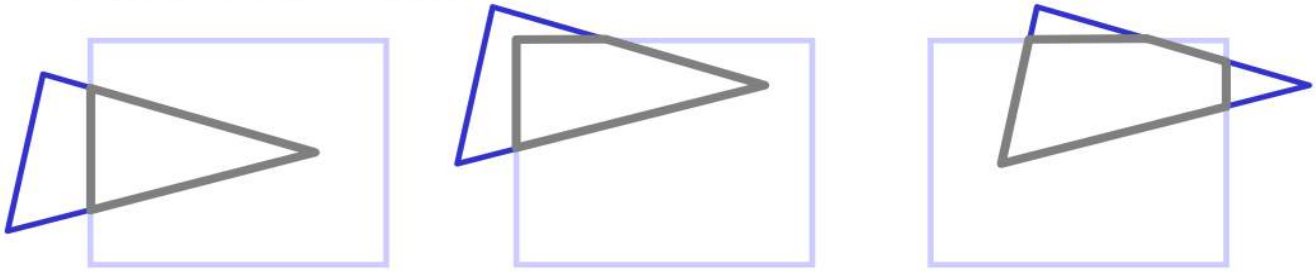
Why can't we just clip the lines of a polygon?



Why Is Clipping Hard?

What happens to a triangle during clipping?

Possible outcomes:

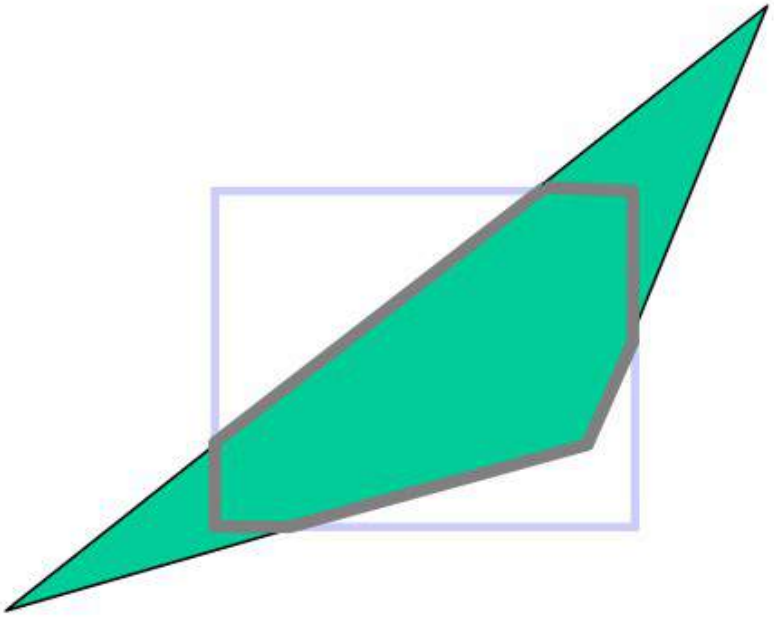


triangle ⇨ triangle triangle ⇨ quad triangle ⇨ 5-gon

How many edges can a clipped triangle have?

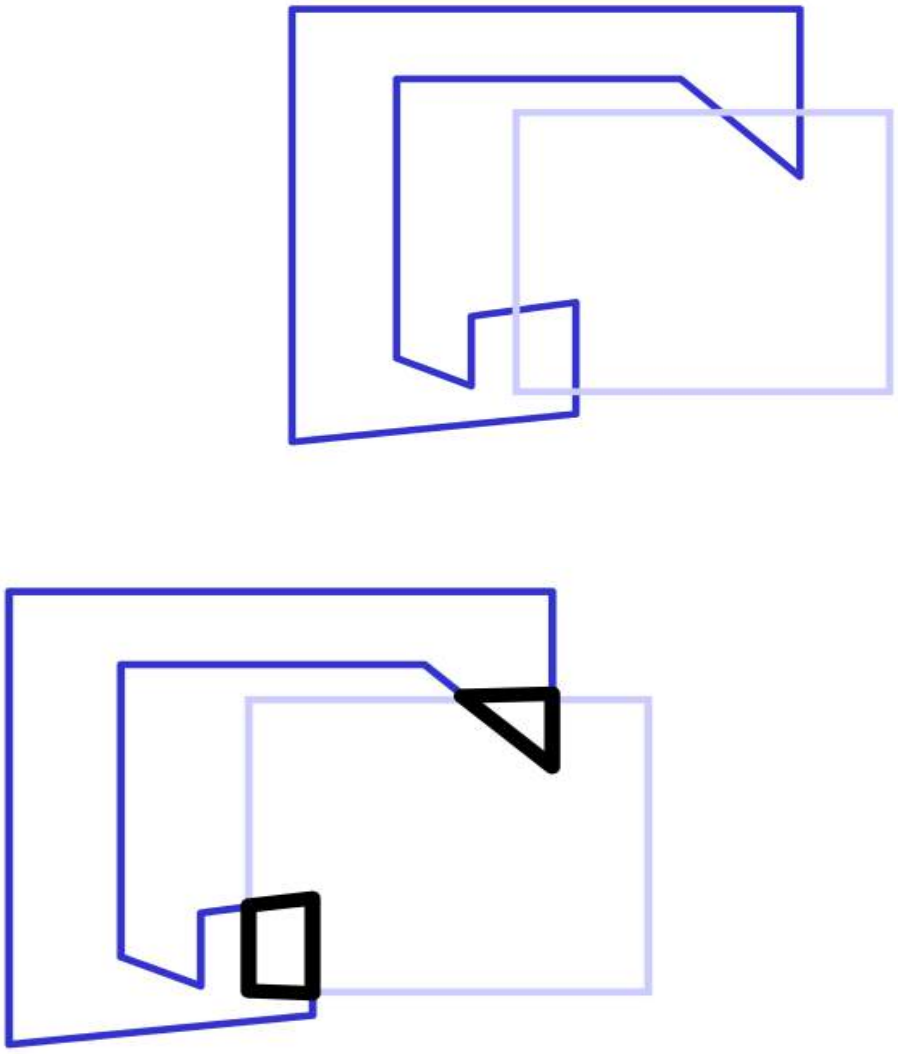
How many edges?

Seven…



Why Is Clipping Hard?

A really tough case:

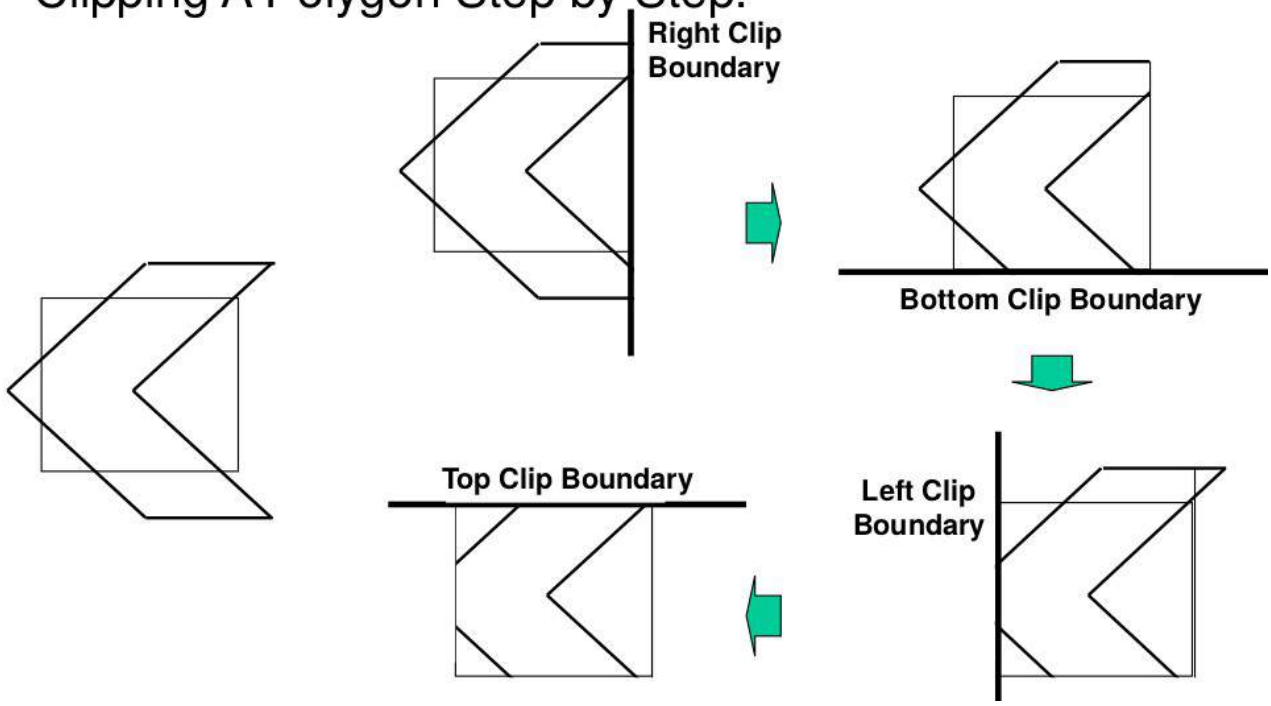


concave polygon ⇨ multiple polygons

Sutherland-Hodgeman algorithm (A divide-and-conquer strategy)

- Polygons can be clipped against each edge of the window one at a time. 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.

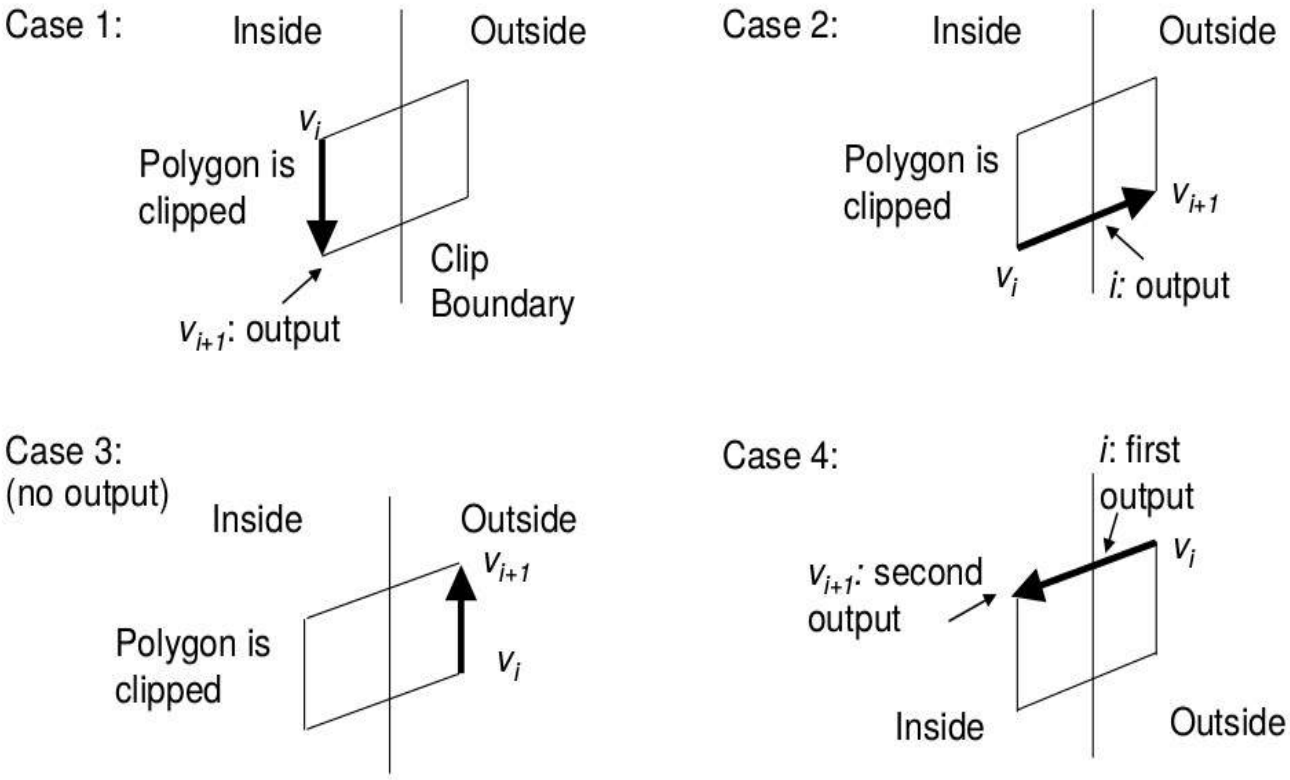
Clipping A Polygon Step by Step:



Sutherland-Hodgeman Algorithm

Note the difference between this strategy and the Cohen-Sutherland algorithm for clipping a line: the polygon clipper clips against each window edge in succession, whereas the line clipper is a recursive algorithm.

Given a polygon with n vertices, $v_1, v_2, \ldots, v_n$, the algorithm clips the polygon against a single, infinite clip edge and outputs another series of vertices defining the clipped polygon. In the next pass, the partially clipped polygon is then clipped against the second clip edge, and so on. Let us consider the polygon edge from vertex v_i to vertex v_{i+1}. Assuming that the start point v_i has been dealt with in the previous iteration, four cases will appear.

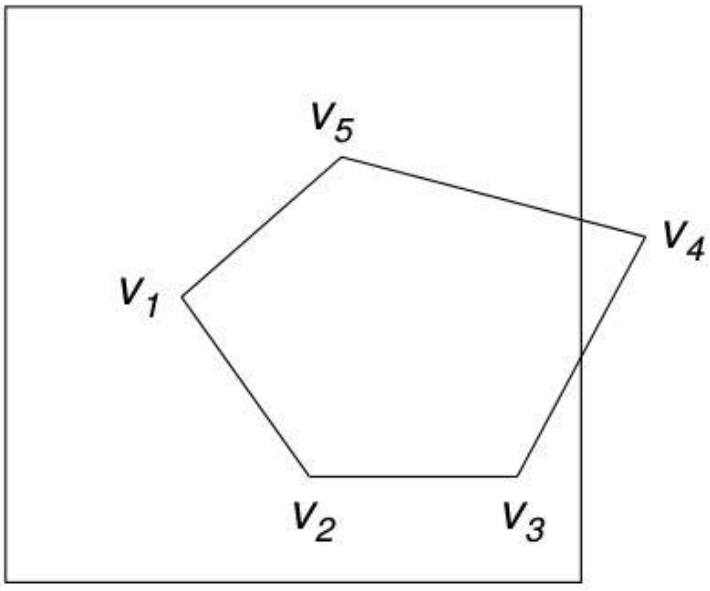


Sutherland-Hodgeman Clipping

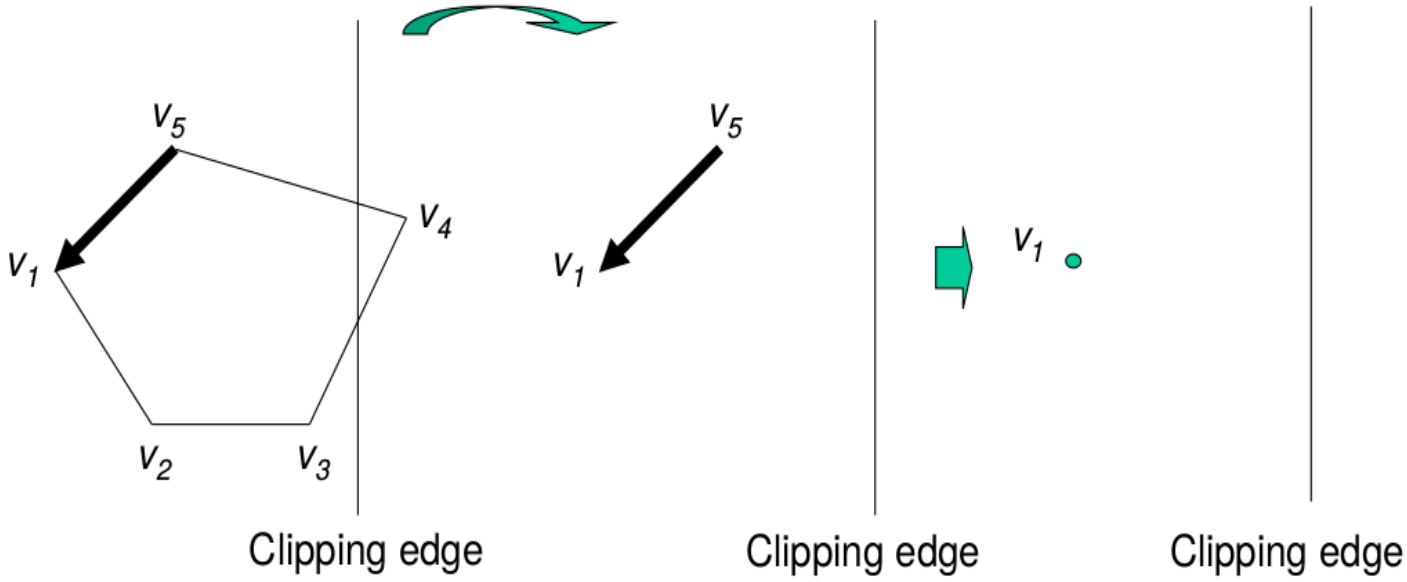
Four cases:

- *s* inside plane and *p* inside plane
  - Add *p* to output
  - Note: *s* has already been added
- *s* inside plane and *p* outside plane
  - Find intersection point *i*
  - Add *i* to output
- *s* outside plane and *p* outside plane
  - Add nothing
- *s* outside plane and *p* inside plane
  - Find intersection point *i*
  - Add *i* to output, followed by *p*

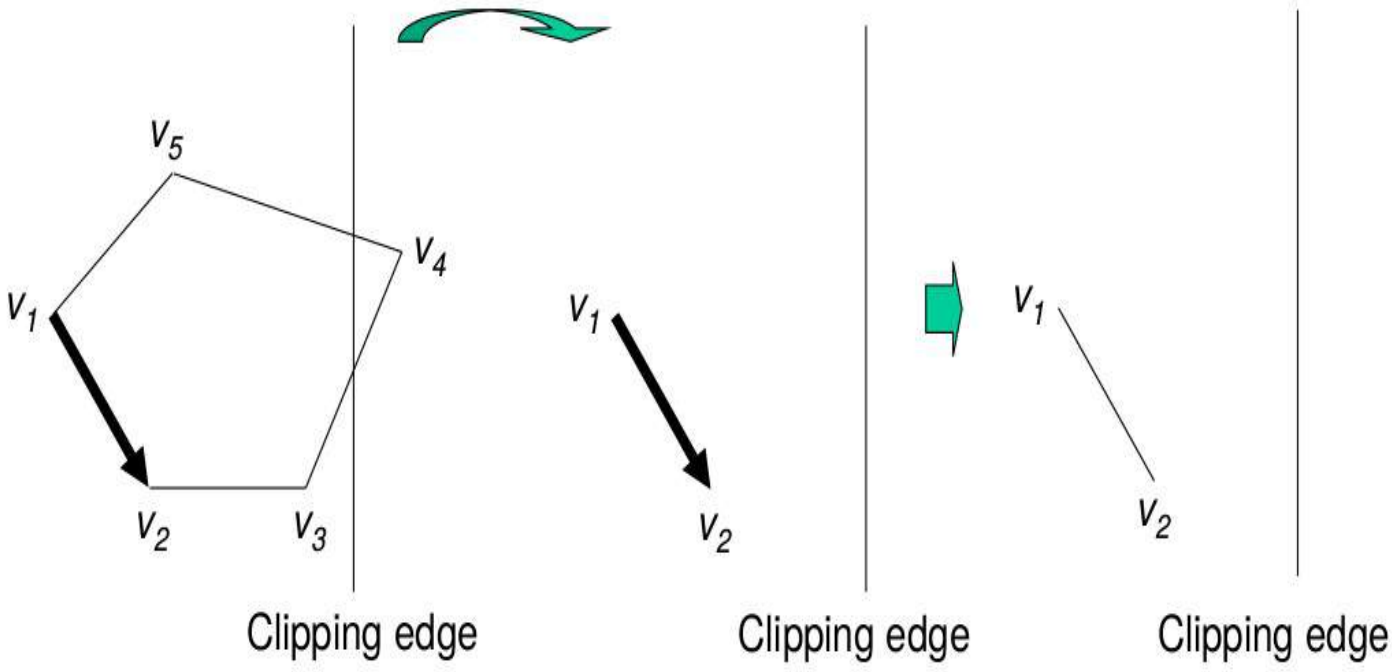
An example for the polygon clipping



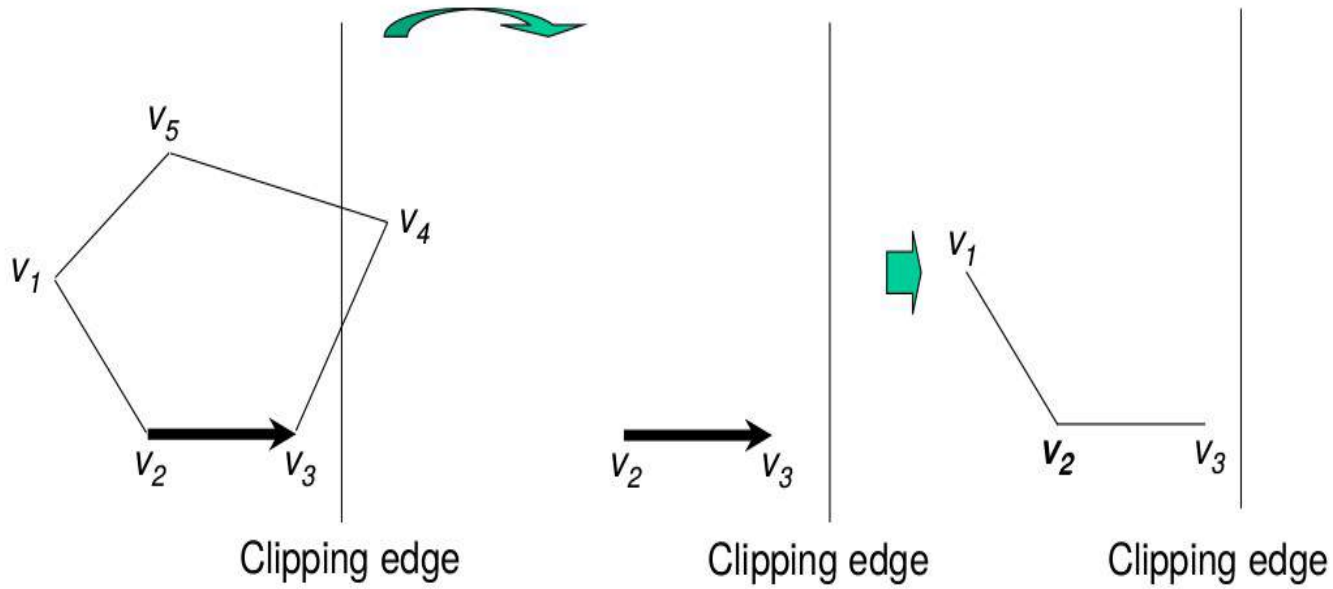
As we said, the Sutherland-Hodgeman algorithm clips the polygon against one clipping edge at a time. We start with the right edge of the clip rectangle. In order to clip the polygon against the line, each edge of the polygon have to be considered. Starting with the edge, represented by a pair of vertices, v_5v_1:



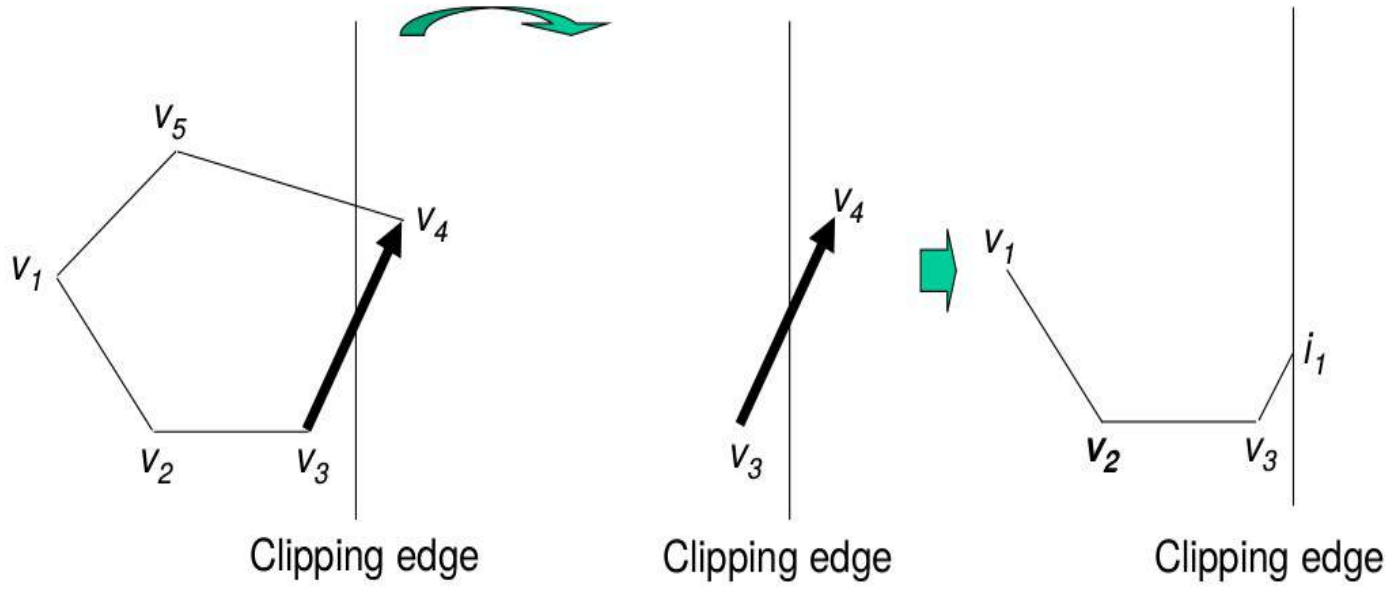
Now v_1v_2:



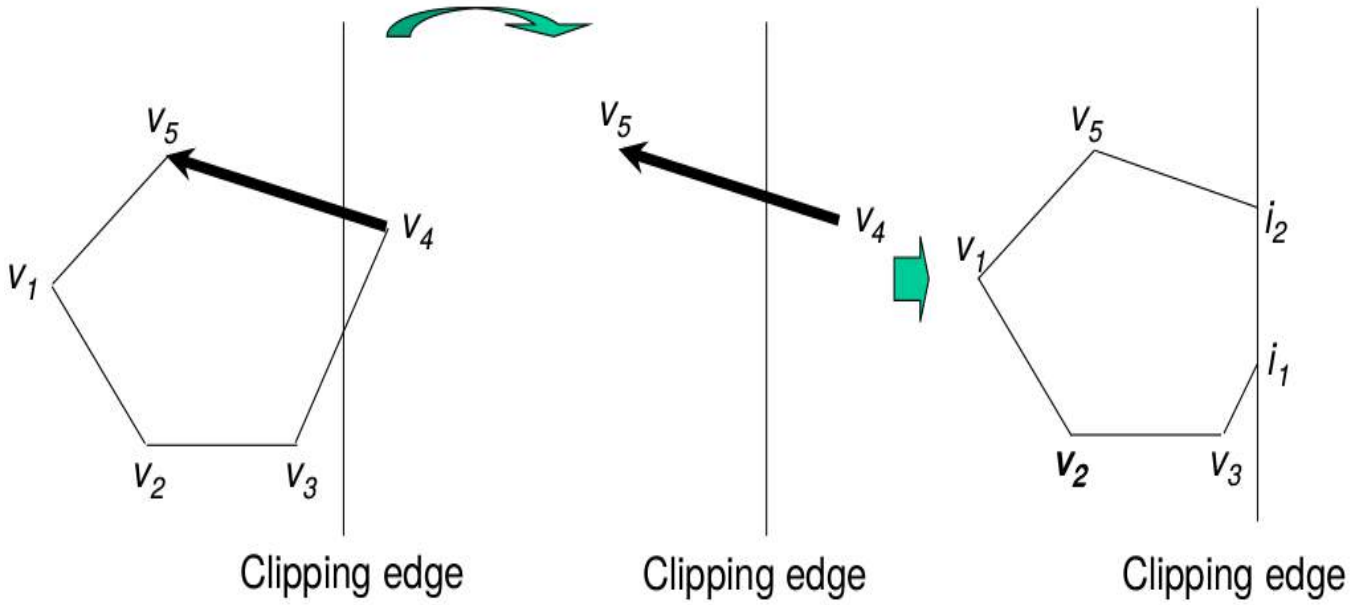
Now v_2v_3:



Now v_3v_4:



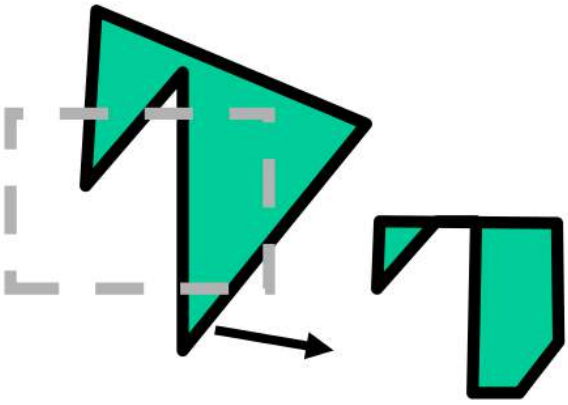
Now v_4v_5:



After these, we have to clip the polygon against the other three edges of the window in a similar way.

Problem with Sutherland-Hodgeman

Concavities can end up linked:



Weiler-Atherton creates separate polygons in cases like this.

Weiler-Atherton Polygon Clipping

To find the edges for a clipped polygon, we follow a path (either clockwise or counterclockwise) around the fill area that detours along a clipping-window boundary whenever a polygon edge crosses to the outside of that boundary. The direction of a detour at a clipping-window border is the same as the processing direction for the polygon edges.

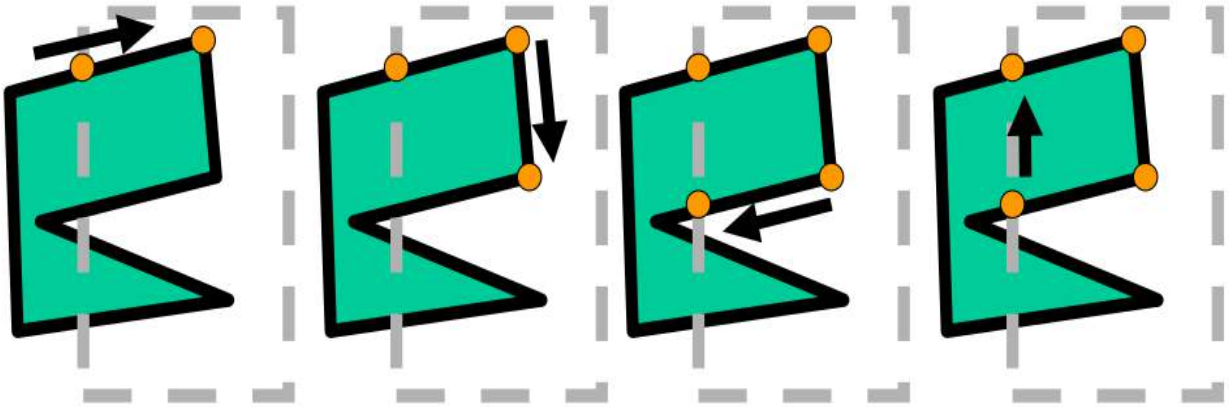
For a counterclockwise traversal of the polygon vertices, we apply the following **Weiler-Atherton** procedures:

1. Process the edges of the polygon in a counterclockwise order until an inside-outside pair of vertices is encountered for one of the clipping boundaries; that is, the first vertex of the polygon edge is inside the clip region and the second vertex is outside the clip region.
2. Follow the window boundaries in a counterclockwise direction from the exit-intersection point to another intersection point with the polygon. If this is a previously processed point, proceed to the next step. If this is a new intersection point, continue processing polygon edges in a counterclockwise order until a previously processed vertex is encountered.

3. Form the vertex list for this section of the clipped polygon.
4. Return to the exit-intersection point and continue processing the polygon edges in a counterclockwise order.

Note: this may generate more than one polygon!

Example:

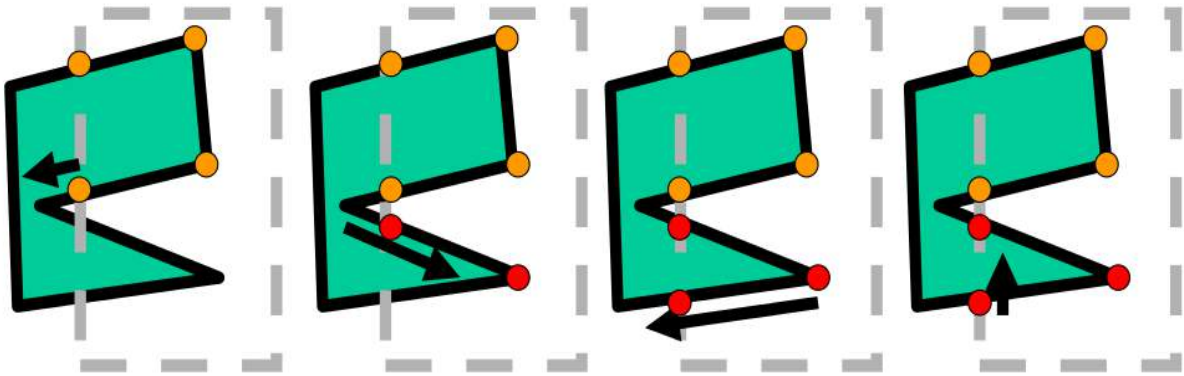


add clip pt.
and end pt.

add end pt.

add clip pt.
cache old dir.

follow clip edge until
a) new crossing found
b) reach pt. already added

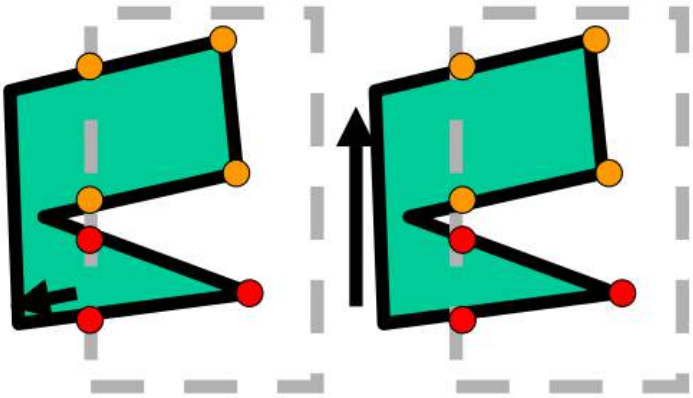


continue from
cached location

add clip pt.
and end pt.

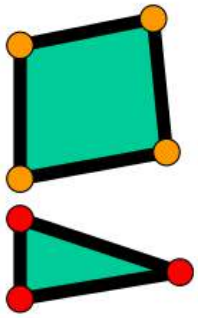
add clip pt.
cache dir.

follow clip edge until
a) new crossing found
b) reach pt. already added



continue from cached location

nothing added finished

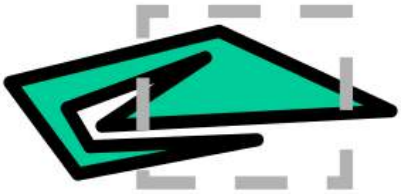


Final result:
Two *unconnected* polygons

Difficulties with Weiler-Atherton polygon clipping

What if the polygon re-crosses edge?

How many “cached” crossings?



Your geometry step must be able to *create* new polygons instead of 1-in-1-out