Polygon clipping with parallel axis boundaries

Jose Commins, March 2003

Revised edition April 2021

email: axora@axora.net

ABSTRACT

Clipping convex polygons to rectangular regions presents challenges when implementing the algorithms in computer software or hardware.

This paper presents a solution to such challenges by using the properties of parallel axis boundaries to offer the following advantages:

- No duplicate vertices or degenerate edges are produced.
- No round-off or disjoint errors when connecting polygons share the same vertices in any orientation, beneficial when using integer math.
- Fast visibility/clip tests.
- Steps can be efficiently pipelined in one pass per axis.

1. Introduction.

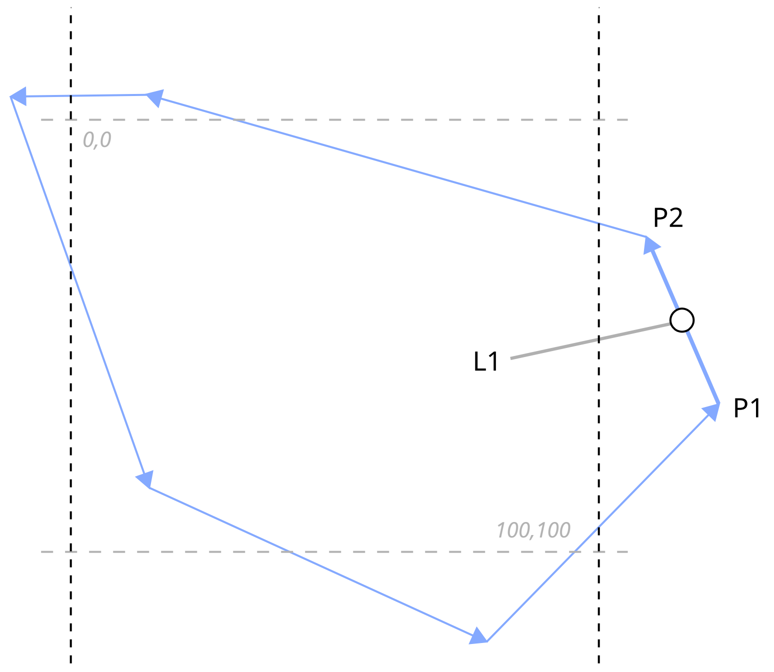
When clipping convex polygons to a rectangular region most algorithms present a combination of outcomes that require special consideration; handling of the input and output polygon vertices in respect to the clipping boundary can produce duplicate points or degenerate edges. Disjoint edges are also an issue between opposite-oriented polygons sharing the same coordinates, due to scaling of similar triangle intersection calculations exhibiting round-off bias.

This paper provides a solution to these problems by harnessing the properties of a line's orientation within parallel axis boundaries, coupling a fast reject-clip-accept test that also serves in solving the outlined challenges in one pass per-axis.

2. Algorithm operation.

2.1. Out of bounds rejection.

Example two-dimensional anticlockwise polygon [*Figure 1*], taking line *L1*:



[Figure 1]

Lines are evaluated according to their order in regards to the axis boundary:

- When P1 > P2
  - If P2 > axis max boundary, line is out of bounds.
  - If P1 < axis min boundary, line is out of bounds.
- When P2 > P1
  - If P1 > axis max boundary, line is out of bounds.
  - If P2 < axis min boundary, line is out of bounds.

Example: *L1* in [*Figure 1*] : P1 > P2 on the (highlighted) X axis, thus is out of bounds as P2 > axis max boundary.

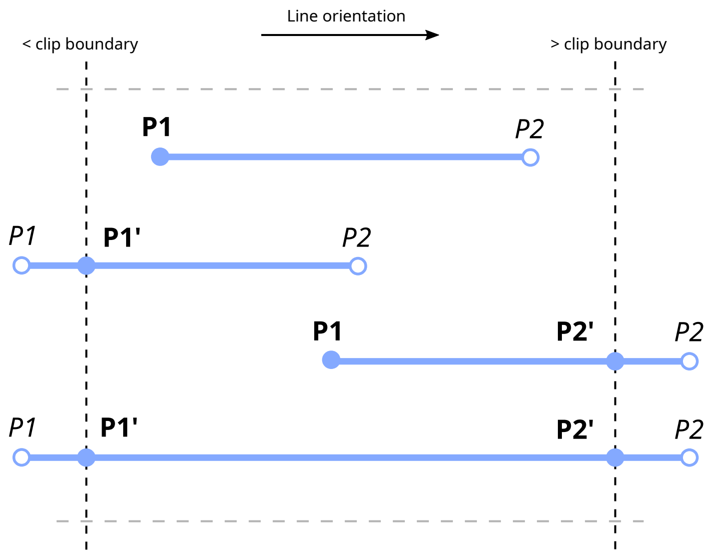
2.2. Clipping.

The algorithm subsequently proceeds using the orientation resolved by the out of bounds rejection phase to determine the amount and order of points to output in the clipping process.

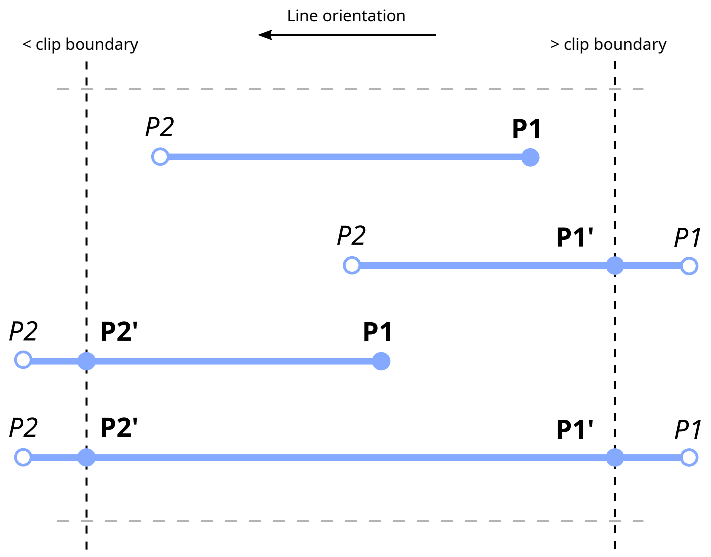
[*Figure 2*] and [*Figure 3*] refer to the coordinate output conditions according to a line's orientation with regards to an axis clipping boundary:

- *P* in *italic* : discard point.
- **P** in **bold** : add original point to output polygon.
- **P'** in **bold** (derivative) : add clipped point to output polygon.

Conditions for output according to orientation and axis boundary:

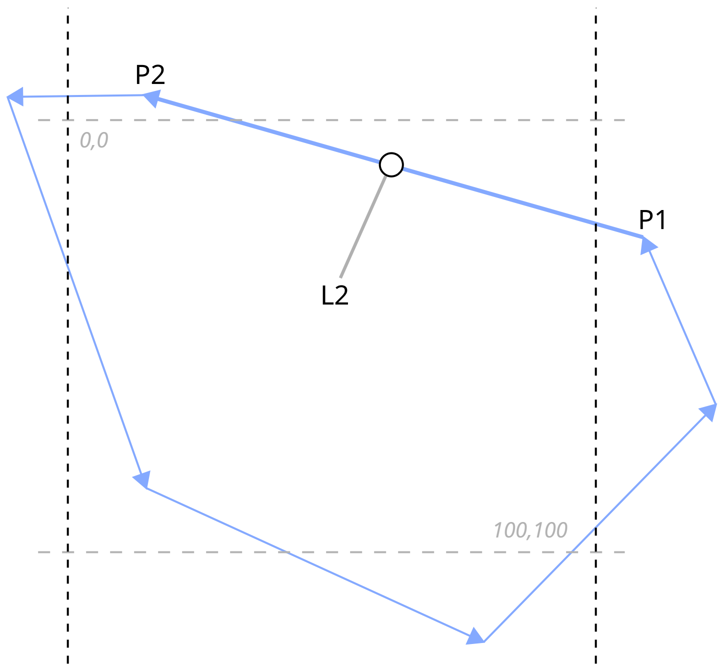


[Figure 2, *P2 > P1 boundary clipping scenarios*]



[Figure 3, *P1 > P2 boundary clipping scenarios*]

Example: clipping line *L2* in [*Figure 4*] polygon by the (highlighted) X axis:



[Figure 4]

In this example L2 is P1 > P2, thus [*Figure 3*] applies. P1 crosses the rightmost clip boundary, P2 is within both axis boundaries: only P1' (clipped P1 derivative) is output.

Note that P1 or its derivative is always output; this is convenient when implementing the algorithm by replacing P1 if clipped with P1' before it is output. P2' is only output when P2 is clipped.

The combination of point output according to orientation and replacing P1 if a clip occurs ensures that no duplicate points are produced.

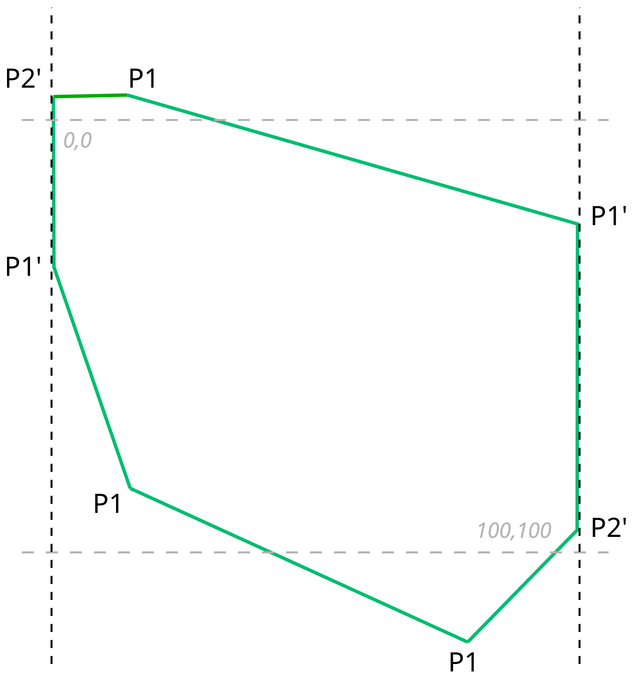
2.3. Edge continuity.

As the algorithm processes lines using their orientation, edge continuity is preserved as shared edges possess the same similar triangles when clipped; thus integer math can be used without exhibiting disjoint edge artifacts.

For example, if two polygons share points but use them in a different order (clockwise to anticlockwise for example) the resultant clipped polygons are not subject to edge gaps as the algorithm exhibits the same round-off bias regardless of line orientation.

3. Result of algorithm on example polygon clipped to the X axis.

[*Figure 5*] is the result of the anticlockwise oriented polygon in [*Figure 1*] clipped to the X axis. No redundant points or degenerate edges are produced.



[Figure 5, *polygon from Figure 1 after clipping to the (highlighted) X axis*]

4. Example code.

An example written in ANSI C/99: 'TwoAxisPolygonClip.c' is provided, implementing the algorithm to efficiently clip an arbitrary n-sided convex polygon to a rectangular region.

5. References.

Sutherland, I.E., and Hodgman, G.W., "Reentrant Polygon Clipping," CACM, 17(1), January 1974, 32-42

Nicholl, T.M., D.T. Lee, and R.A. Nicholl, "An Efficient New Algorithm for 2-D Line Clipping: Its Development and Analysis," SIGGRAPH 87, 253-262