





WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)  
[Wikipedia store](#)

Interaction  
[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact page](#)

Tools  
[What links here](#)  
[Related changes](#)  
[Upload file](#)  
[Special pages](#)  
[Permanent link](#)  
[Page information](#)  
[Wikidata item](#)  
[Cite this page](#)

Print/export  
[Create a book](#)  
[Download as PDF](#)  
[Printable version](#)


Languages   
[Español](#)  
 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#)

[More](#) ▾



# Line clipping

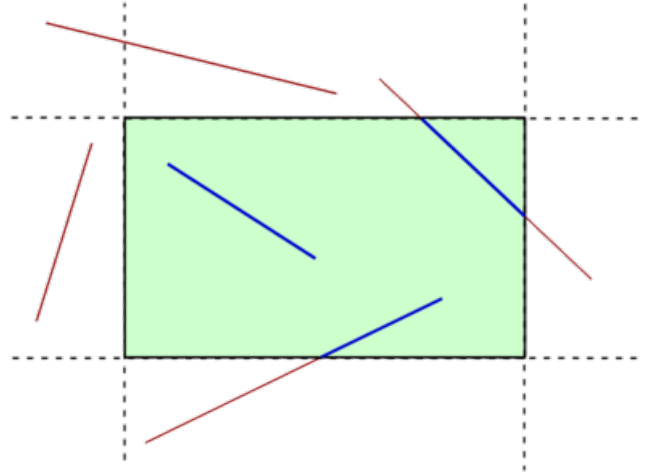
From Wikipedia, the free encyclopedia  
(Redirected from [Fast clipping](#))

In [computer graphics](#), '**line clipping**' is the process of removing lines or portions of lines outside of an area of interest. Typically, any line or part thereof which is outside of the viewing area is removed.

There are two common algorithms for line clipping: Cohen–Sutherland and Liang–Barsky.

## Contents [hide]

- [1 Cohen–Sutherland](#)
- [2 Liang–Barsky](#)
- [3 Cyrus–Beck](#)
- [4 Nicholl–Lee–Nicholl](#)
- [5 Fast clipping](#)
- [6 \*O\*\(lg \*N\*\) algorithm](#)
- [7 Skala](#)
- [8 See also](#)
- [9 References](#)



## Cohen–Sutherland [[edit](#)]

*Main article:* [Cohen–Sutherland](#)

In computer graphics, the Cohen–Sutherland algorithm (named after Danny Cohen and Ivan Sutherland) is a line clipping algorithm. The algorithm divides a 2D space into 9 regions, of which only the middle part (viewport) is visible.

In 1967, flight simulation work by Danny Cohen led to the development of the Cohen–Sutherland computer graphics two- and three-dimensional line clipping algorithms, created with Ivan Sutherland.<sup>[1]</sup>

## Liang–Barsky [[edit](#)]

*Main article:* [Liang–Barsky](#)

The Liang–Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping box to determine the intersections between the line and the clipping box. With these intersections it knows which portion of the line should be drawn. This algorithm is significantly more efficient than Cohen–Sutherland, but Cohen–Sutherland does trivial accepts and rejects much faster, so it should be considered instead if most of the lines you need to clip would be completely in or out of the clip window.

## Cyrus–Beck [[edit](#)]

*Main article:* [Cyrus–Beck algorithm](#)

Very similar to Liang–Barsky algorithm. The difference is that Liang–Barsky is a simplified Cyrus–Beck variation that was optimized for a rectangular clip window.

The Cyrus–Beck algorithm is of *O*(*N*) complexity, and it is primarily intended for a clipping a line in the parametric form against a convex polygon in 2 dimensions or against a convex polyhedron in 3 dimensions.<sup>[1]</sup>

## Nicholl–Lee–Nicholl [[edit](#)]

*Main article:* [Nicholl–Lee–Nicholl](#)

The Nicholl–Lee–Nicholl algorithm is a fast line clipping algorithm that reduces the chances of clipping a single line segment multiple times, as may happen in the Cohen–Sutherland algorithm. The clipping window is divided

into a number of different areas, depending on the position of the initial point of the line to be clipped.

## Fast clipping <sup>[edit]</sup>

This algorithm has similarities with Cohen-Sutherland. The start and end positions are classified by which portion of the 9 area grid they occupy. A large switch statement jumps to a specialized handler for that case. In contrast, Cohen-Sutherland may have to iterate several times to handle the same case.<sup>[2]</sup>

## $O(\lg N)$ algorithm <sup>[edit]</sup>

This algorithm classifies vertices against the given line in the implicit form  $\mathbf{p}: ax+by+c=0$ . As the polygon is assumed to be convex and vertices are ordered clockwise or anti-clockwise binary search can be applied and leads to a  $O(\lg N)$  run time complexity.<sup>[3]</sup>

## Skala <sup>[edit]</sup>

This algorithm is based on [homogeneous coordinates](#) and [duality](#).<sup>[4]</sup> It can be used for line or line segment clipping against a rectangular window as well as against a convex polygon. The algorithm is based on classifying a vertex of the clipping window against a half-space given by a line  $\mathbf{p}: ax+by+c=0$ . The result of the classification determines the edges intersected by the line  $\mathbf{p}$ . The algorithm is simple, easy to implement and extensible to a convex window as well. The line or line segment  $\mathbf{p}$  can be computed from points  $\mathbf{x}_1$ ,  $\mathbf{x}_2$  given in homogeneous coordinates directly using the cross product as

$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2 = [x_1, y_1, w_1] \times [x_2, y_2, w_2]$  or as

$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2 = [x_1, y_1, 1] \times [x_2, y_2, 1]$

## See also <sup>[edit]</sup>

- Clipping (computer graphics)

## References <sup>[edit]</sup>

- ↑ Cyrus, M., Beck, J.: Generalized Two and Three Dimensional Clipping, Computers&Graphics, Vol.3, No.1, pp.23-28, 1978
- ↑ Mark S. Sobkow, Paul Pospisil and Yee-Hong Yang. A Fast Two-Dimensional Line Clipping Algorithm via Line Encoding//Computer & Graphics, Vol. 11, No. 4, pp. 459–467, 1987
- ↑ **Skala, V.:**  *$O(\lg N)$  Line Clipping Algorithm in E2*, Computers & Graphics, Pergamon Press, Vol.18, No.4, 1994
- ↑ Skala, V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, ISSN 0178-2789, Vol.21, No.11, pp.905-914, Springer Verlag, 2005

Categories: [Clipping \(computer graphics\)](#)

This page was last modified on 18 June 2015, at 20:12.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

