# S-Clip E2: A New Concept of Clipping Algorithms


http://www.VaclavSkala.eu Vaclav Skala skala@kiv.zcu.cz

**University of West Bohemia**
Faculty of Applied Sciences
Plzen, Czech Republic

**VŠB-Technical University Ostrava**
Faculty of Electrical Engineering and Computer Science
Ostrava, Czech Republic


## 1. Introduction

A clipping operation is a fundamental operation within all computer graphics systems and it has been well explored as many algorithms have been developed. However there are still other ways how a graphical primitive can be clipped against a window. The type of the window has a strong influence on the final algorithm complexity. The window can be orthogonal and axis aligned, 4-sided orthogonal non-axis aligned convex or non-convex. Clipped primitives are mostly lines, line segments and polygons of the above mentioned types. It leads to algorithms with *O(lg N)* [Skala 1994] up to *O(NM)* complexities in general. In some cases, when a clipping window is constant and many primitives are to be clipped, a clipping algorithm with *O(1)* complexity [Skala 1996] can be used. Algorithms are mostly based on Cohen-Sutherland, Cyrus-Beck, Liang-Barsky and Sutherland-Hodgman algorithms [Theoharis, et al. 2008]. Some of them are intended for hardware implementation but all operate in the Euclidean space.

Points in computer graphics are represented in homogeneous coordinates as $\mathbf{x}=[x,y:w]^T$, i.e. actually in the projective extension of the Euclidean space. A line $p$ in the implicit form is represented by a vector $\mathbf{a}=[a,b:c]^T$, i.e. $ax+by+cw=0$ , $\mathbf{a}^T\mathbf{x} = 0$. Due to the principle of duality in $E^2$ points and lines, join and intersection etc. are dual. It means that any theorem remains valid if the meaning of the symbols is swapped. This leads to nice applications, e.g.

$$\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{vmatrix} = [a,b:c]^T$$

where $p$ is the line given by two points in homogeneous coordinates. Due to the principle of duality we can determine an intersection point of two lines. So we need only one code to compute both.

$$\mathbf{x} = \mathbf{p}_1 \times \mathbf{p}_2 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix} = [x,y:w]^T$$

## 2. Proposed Smart Clip Algorithm

The approach presented is based on a basic principle – test first and then compute. Unlike Cohen-Sutherland's algorithm it evaluates a position of the given line with respect to the corners of the clipping window.

### 2.1. Line clipping

Let us consider a line $p$ given by two points $\mathbf{x}_A$ and $\mathbf{x}_B$ as $\mathbf{p} = \mathbf{x}_A \times \mathbf{x}_B$ clipped by an orthogonal axis aligned window. The clipping window is given by corners $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ .

The position of the line $p$ with respect to the corners of the clipping window is determined by a code vector c. The binary code of the window vertices for the line $p$ can be determined as

**for** i := 1 **to** N-1 **do** /* $\mathbf{c} = [c_3,c_2,c_1,c_0]^T$ */

{ $\xi_i := \mathbf{p}^T \mathbf{x}_i$ ; **if** $\xi_i \geq 0$ **then** $c_i = 1$ **else** $c_i = 0$ }

Algorithm 1

It should be noted, that the vector $c$ is constructed differently from the Cohen-Sutherland's algorithm as the window corner's position against the line $p$ is now coded.

This is actually a classification of corners by a half space given by the line $p$. The code $\boldsymbol{c}$ is actually the index to a table in which indices of intersected window edges are stored. Note that there is no geometrical meaning for some values $\boldsymbol{c}$ and "N/A" value is assigned. Value "-1" means no intersection at all [Skala 2004]. $TAB_1$, $TAB_2$ stores indices of the edges intersected by the line.

| code | $TAB_1$ | $TAB_2$ | code | $TAB_1$ | $TAB_2$ |
|---|---|---|---|---|---|
| 0000 | -1 | -1 | 1111 | -1 | -1 |
| 0001 | 3 | 0 | 1110 | 0 | 3 |
| 0010 | 0 | 1 | 1101 | 1 | 0 |
| 0011 | 3 | 1 | 1100 | 1 | 3 |
| 0100 | 1 | 2 | 1011 | 2 | 1 |
| 0101 | N/A | N/A | 1010 | N/A | N/A |
| 0110 | 0 | 2 | 1001 | 2 | 0 |
| 0111 | 3 | 2 | 1000 | 2 | 3 |

Table 1

```
procedure CLIP_L; /* input: x_A , x_B   ; CONVEX polygon */
/* x_A=[x_A,y_A : w_A]^T x_B=[x_B, y_B :w_B]^T , w_A, w_B > 0*/
/* x_k=[x_k, y_k : w_k]^T , w_k > 0          N - number of window edges*/
{     /* ax+by+c = 0; p = [a,b : c]^T */
/*1*/   p := x_A × x_B;
/*2*/   for k:=0 to N-1 do /* x_k=[x_k, y_k: w_k]^T */
/*3*/       if p^T x_k ≥ 0 then c_k:=1 else c_k:=0;
/*4*/       if  c = [0...0]^T or c = [1....1]^T  then EXIT;
/*5*/       i:= TAB_1[c];  j:= TAB_2[c];
/*6*/       x_A :=  p × e_i ; x_B :=  p × e_j ;
/*7*/       output (x_A , x_B )
} /*CLIP_L*/;
```

Algorithm 2

***A simple, elegant solution!***

Now, we know the edges intersected by the given line $p$. The intersection points can be computed as

$$\mathbf{x}_A = \mathbf{p} \times \mathbf{e}_{i1} \qquad \mathbf{x}_B = \mathbf{p} \times \mathbf{e}_{i2}$$

where: $\mathbf{e}_{i*}$ mean clipping window edges given by the Tab.1.

It can be seen that this algorithm is quite general as it does not need the clipping window edges aligned with the axes and it is applicable for clipping by a convex window as well. In the case of an orthogonal axis aligned window and with the corners given in the Euclidean space, i.e. $w_k = 1$ , the code can be even more optimized. If clipping is a part of the graphical pipeline, then clipping itself is made in the normalized space, i.e. in the interval <-1,1>×<-1,1>.. It means that the transformation pipeline **R** (**clip**) **S** is actually modified to the "normalized" clip given by the transformation:

**R N** (**normalized clip**) $\mathbf{N}^{-1}$ **S = R'** (**normalized clip**) **S'**

where: **R'** and **S'** are cumulative transformation matrices. Now, the vectors of edges are $\mathbf{e}_i = [\pm 1, 0: 0]^T$ or $\mathbf{e}_j = [0, \pm 1: 0]^T$ .

It means that multiplications in the if $\boldsymbol{p}^T\boldsymbol{x}_k \geq 0$ statement are ***not needed***. The cost of multiplication is actually hidden to the cumulative matrices, but it is made only once independently from a number of primitives processed.

Analyzing the algorithm, it can be seen that the proposed concept and algorithm above can be used for any 4-sided clipping window, not necessarily orthogonal, for axis unaligned windows and for **N-sided convex polygon clipping** as well.

The Tab.1 can be easily generated for a convex polygon by an algorithm as follows:

```
/*no intersection at all */
TAB_1[0000]:=-1; TAB_2[0000]:=-1;
TAB_1[1111]:=-1; TAB_2[1111]:=-1;
/* N = 2^k, where: k is number of windows edges */
for c:=1 to N-2 do      /* c = 0 and c = N-1 solved already */
{ /* take the code c and extended by c_0 from the left */
/* to avoid modulo operation over indices */
/* q = [c_0 ,c_3,c_2,c_1,c_0]^T      */
    i := 1;
    for k := N-1 to 1do
    {   if i ≤ 2 then
        {   if q_{k+1} ≠ q_k then { TAB_i[c] :=  e_k;  i := i+1; }
        }
        else { TAB_1[c] := N/A;    TAB_2[c] := N/A;}
    }
}
```

Algorithm 3 Table generation for a general convex case

### 2.2. Line segment clipping

Line segment clipping is a bit more complicated as the position of end points of the line segment must be considered. It leads to a more complicated structure of computation, but with low computational expense, see [Skala 2005].

In many cases we are more interested in a parametric form of the line $p$ and in the resulting line segment. If the line segment is given in a parametric form as $p$: $\mathbf{x}(t) = \mathbf{x}_A + \mathbf{s}\,t$ then the solution is quite simple.

Let us consider clipping by a 4-sided convex window with anticlockwise orientation. The first step is clipping of the line on which the line segment lies by the Algorithm 1. Now, we know which edges, $e_i$ and $e_j$ intersected by the line $p$. A parameter $t$ is determined by equations for intersected edge $e_i$ , resp. $e_j$ as

$$e_i: \mathbf{a}_i^T \mathbf{x} = 0 \qquad \text{and} \qquad p: \mathbf{x}(t) = \mathbf{x}_A + \mathbf{s}\,t$$

where: $\mathbf{x}_A = [x, y: w]^T$ and $\mathbf{s} = [x_B - x_A, y_B - y_A: w_B - w_A]^T$

Note, that computation is made in the projective space using homogeneous coordinates. The line $p$ is considered as a linear interpolation with monotonically non-linear parameterization [Skala 2011]. Solving those equations we get

$$t = [\mathbf{a}^T\mathbf{x}_A : \mathbf{a}^T\mathbf{s}]^T$$

where: $t$ is a scalar value expressed in homogeneous coordinates. For both intersected edges we get two values $t$, i.e. $t_{min}$ and $t_{max}$. The resulting segment is determined as $\langle t_{min} , t_{max} \rangle \cap \langle 0 , 1 \rangle$ that is a trivial operation. If the orientation of the clipping window is known, no ordering of $t$ values is needed.

### 2.3. Non-convex Window

The presented concept and algorithm are also valid for non-convex line and line segment clipping with some modifications as the values $t$ have to be ordered as there might be more than one segment, i.e. several intervals $\langle t_{min} , t_{max} \rangle$ can occur. Also the intersection computation of $\langle t_{min} , t_{max} \rangle \cap \langle 0 , 1 \rangle$ is to be done for all $k/2$ resulting segments. Therefore, the algorithm is of $O(N)+O(k \lg k)$ complexity, where $k$ is a number of segments.

## 3. Results

The presented concept and algorithms were experimentally verified. The proposed algorithm is slightly faster than the Cohen-Sutherland and about 2 times faster than Cyrus-Beck algorithms if all points are given in the Euclidean coordinates. If points are given in homogeneous coordinates or the window is in a general position the proposed algorithm is significantly faster.

## 4. Conclusions

A new approach is presented to line and line segment clipping by a convex polygon in $E^2$. The algorithm is simple, easy to implement and convenient for GPU application, too. The presented concept is more general, offers higher robustness and reduces division operations. It is comparable and competitive, at least with other relevant algorithms.

A significant advantage of the proposed concept is the use of a separation function, which is not only more robust, but enables avoidance of complicated cases difficult to resolve [Skala 1989].


## Acknowledgment

The author would like to thank to colleagues at the University of West Bohemia, Plzen, VSB-Technical University, Ostrava in Czech Republic, Shandong University and Zhejiang University in China for their comments and suggestions. Research was supported by MSMT CR projects ME10060, LH12181.


## References


CYRUS, M., BECK, J. 1978. Generalized Two and Three Dimensional Clipping, *Computers&Graphics*, Vol.3, No.1, Pergamon Press, pp.23-28.

BUI, D.H., SKALA, V. 1998. Fast Algorithms for Line Segment and Line Clipping in E2, *The Visual Computer*, No.1, Vol.14, Springer Verlag, pp. 31-37.

LIANG, D.H., BARSKY, B. 1984. A new concept and method for Line Clipping, ACM, *TOG*,3(I), pp.l-22.

SKALA,V., 1989. Algorithms for 2D Line Clipping, *Eurographics89 Proceedings*, Elsevier, pp.355-366.

SKALA, V. 1994. O(lgN) Line Clipping Algorithm in $E^2$, *Computers&Graphics*, *Elsevier*, Vol.18, No.4, pp.517-524.

SKALA, V. 1996. Line Clipping in $E^2$ with O(1) Processing Complexity, *Computers&Graphics*, Vol.20, No.4, pp.523-530.

BUI, D.H., SKALA, V. 1998. Fast Algorithms for Line Segment and Line Clipping in $E^2$, *The Visual Computer*, No.1, Vol.14, Springer Verlag, pp. 31-37.

SKALA, V., BUI, D.H. 2000. Two New Algorithms for Line Clipping in $E^2$ and Their *Comparison, Machine Graphics&Vision*, No.1/2, Vol.9, Poland Academy of Sciences, Poland, pp.297-306.

SKALA, V. 2004. A New Line Clipping Algorithm with Hardware Acceleration. *CGI2004 Proceedings*, pp. 270-273.

SKALA, V. 2005. A new approach to line and line segment clipping in homogeneous coordinates, *The Visual Computer*, Vol.21, No.11, Springer Verlag, pp.905-914.

SKALA, V. 2010. Duality and Intersection Computation in Projective Space Using GPU Support, *WSEAS Trans. on Mathematics*, Vol.9, No.6, pp.407-416.

SKALA, V. 2012. Projective Geometry and Duality for Graphics, Games and Visualization, course notes, 2012, *ACM SIGGRAPH*.

THEOHARIS, T.,PAPAIOANNOU, G., PLATIS,N., PATRIKALAKIS, M.N. 2008. Graphics and Visualization, *A.K.Peters.*